aws

Guida per l'utente

Amazon ECR

Versione API 2015-09-21

Amazon ECR: Guida per l’utente

Table of Contents

Che cos'è Amazon Elastic Container Registry?

Amazon Elastic Container Registry (Amazon ECR) è AWS un servizio di registro di immagini di container gestito sicuro, scalabile e affidabile. Amazon ECR supporta repository privati con autorizzazioni basate sulle risorse tramite IAM. AWS In questo modo utenti o EC2 istanze Amazon specifici possono accedere ai repository e alle immagini dei container. È possibile utilizzare la CLI preferita per inviare, estrarre e gestire le immagini Docker, le immagini OCI (Open Container Initiative) e i manufatti compatibili con OCI.

> **Note**
>
> Amazon ECR supporta anche repository di immagini di container pubblici. Per maggiori informazioni, consulta Che cos'è Amazon ECR Public nella Guida per l'utente di Amazon ECR Public.

Il team dei servizi per i AWS container mantiene una tabella di marcia pubblica su. GitHub Contiene informazioni su ciò a cui stanno lavorando i team e consente a tutti AWS i clienti di fornire un feedback diretto. Per ulteriori informazioni, consulta Roadmap dei container AWS.

Concetti e componenti di Amazon ECR

Amazon ECR è un servizio di registro di container Docker completamente gestito fornito da. AWS Consente di archiviare, gestire e distribuire le immagini dei container Docker in modo sicuro e affidabile. Questi concetti e componenti collaborano per fornire un servizio di registro dei container Docker sicuro, scalabile e affidabile all'interno di AWS, che consente di gestire e distribuire in modo efficiente le applicazioni containerizzate.

Ecco alcuni concetti e componenti chiave di Amazon ECR:

Registry

Un registro Amazon ECR è un archivio privato fornito a ciascun AWS account, in cui è possibile creare uno o più repository. Questi repository consentono di archiviare e distribuire immagini Docker, immagini Open Container Initiative (OCI) e altri elementi compatibili con OCI all'interno del proprio ambiente. AWS Per ulteriori informazioni, consulta Registro privato Amazon ECR.

Token di autorizzazione

Per inviare ed estrarre le immagini, il tuo client deve prima autenticarsi in un registro privato Amazon ECR come utente AWS . Per ulteriori informazioni, consulta Autenticazione del registro privato in Amazon ECR.

Repository

Un repository in Amazon ECR è una raccolta logica in cui è possibile archiviare immagini Docker, immagini Open Container Initiative (OCI) e altri artefatti compatibili con OCI. All'interno di un unico registro Amazon ECR, puoi avere più repository per organizzare le immagini dei tuoi container. Per ulteriori informazioni, consulta Repository Amazon ECR privati.

Politica del repository

Puoi controllare l'accesso ai repository e al contenuto presente in questi attraverso le policy del repository. Per ulteriori informazioni, consulta Politiche di archivio privato in Amazon ECR.

Immagine

Puoi inviare ed estrarre le immagini del container ai tuoi repository. Puoi utilizzare queste immagini localmente sul tuo sistema di sviluppo oppure nelle definizioni di attività Amazon ECS e nelle specifiche pod Amazon EKS. Per ulteriori informazioni, consultare Utilizzo delle immagini Amazon ECR con Amazon ECS e Utilizzo delle immagini Amazon ECR con Amazon EKS.

Politica del ciclo di vita

Le policy del ciclo di vita di Amazon ECR consentono di gestire il ciclo di vita delle immagini definendo regole per l'eliminazione e la scadenza di immagini vecchie o inutilizzate. Per ulteriori informazioni, consulta Automatizza la pulizia delle immagini utilizzando le politiche del ciclo di vita in Amazon ECR.

Scansione di immagini

Amazon ECR offre una funzionalità integrata di scansione delle immagini che aiuta a identificare le vulnerabilità del software nelle immagini dei container. Per ulteriori informazioni, consulta Scansiona le immagini per individuare le vulnerabilità del software in Amazon ECR.

Controllo degli accessi

Amazon ECR utilizza IAM per controllare l'accesso ai tuoi repository. Puoi creare utenti, gruppi e ruoli IAM con autorizzazioni specifiche per inviare, estrarre o gestire i repository Amazon ECR. Per ulteriori informazioni, consulta Sicurezza in Amazon Elastic Container Registry.

Replica tra account e regioni

Amazon ECR supporta la replica delle immagini su più AWS account e regioni per una maggiore disponibilità e una latenza ridotta. Per ulteriori informazioni, consulta Private image replication in Amazon ECR.

Encryption (Crittografia)

Amazon ECR supporta la crittografia lato server delle immagini Docker inutilizzate. AWS KMS Per ulteriori informazioni, consulta Protezione dei dati in Amazon ECR.

AWS Command Line Interface Integration

AWS CLI Fornisce comandi per interagire con i repository Amazon ECR, ad esempio per creare, elencare, inviare ed estrarre immagini.

Console di gestione AWS

Amazon ECR può essere gestito anche tramite Console di gestione AWS, fornendo un'interfaccia Web intuitiva per lavorare con i tuoi repository e le tue immagini.

AWS CloudTrail

Amazon ECR si integra con AWS CloudTrail, consentendoti di registrare e controllare le chiamate API effettuate ad Amazon ECR per scopi di sicurezza e conformità. Per ulteriori informazioni, consulta Registrazione delle azioni Amazon ECR con AWS CloudTrail.

Amazon CloudWatch

Amazon ECR fornisce parametri e log che possono essere monitorati utilizzando Amazon CloudWatch, consentendoti di monitorare le prestazioni e l'utilizzo dei tuoi repository Amazon ECR. Per ulteriori informazioni, consulta Parametri del repository Amazon ECR.

Firma gestita

La firma gestita genera automaticamente firme crittografiche quando le immagini vengono inviate ad Amazon ECR, semplificando la firma delle immagini dei container. Per ulteriori informazioni, consulta Firma gestita.

Casi d'uso comuni in Amazon ECR

Amazon ECR è un servizio di registro di container Docker completamente gestito offerto da. AWS Fornisce un repository sicuro e scalabile per l'archiviazione e la distribuzione delle immagini dei container Docker, rendendolo un componente essenziale nelle implementazioni di applicazioni

containerizzate. Amazon ECR semplifica il processo di creazione, distribuzione ed esecuzione di applicazioni containerizzate su vari AWS servizi e ambienti locali.

Ecco alcuni casi d'uso chiave per Amazon ECR:

Archiviazione e distribuzione delle immagini dei container

Amazon ECR funge da repository centralizzato per l'archiviazione e la distribuzione di immagini di container Docker all'interno di un'organizzazione o per il consumo pubblico. Gli sviluppatori possono inviare le immagini dei contenitori ad Amazon ECR e quindi recuperarle da qualsiasi ambiente di elaborazione interno AWS, come Amazon EC2 o Amazon AWS Fargate EKS. Per ulteriori informazioni, consulta Repository Amazon ECR privati.

Integrazione continua e implementazione continua (CI/CD)

Amazon ECR si integra perfettamente con altri CI/CD strumenti AWS CodeBuild AWS CodePipeline, consentendo la creazione, il test e la distribuzione automatizzati di applicazioni containerizzate. Le immagini dei container possono essere inviate automaticamente ad Amazon ECR come parte della CI/CD pipeline, garantendo una distribuzione coerente e affidabile in ambienti diversi.

Architettura dei microservizi

Amazon ECR è ideale per le architetture di microservizi, in cui le applicazioni sono suddivise in servizi più piccoli e disaccoppiati confezionati come contenitori. Ogni microservizio può avere la propria immagine del contenitore archiviata in Amazon ECR, che consente lo sviluppo, la distribuzione e la scalabilità indipendenti dei singoli servizi.

Implementazioni ibride e multi-cloud

Amazon ECR supporta la capacità di estrarre immagini di container da altri registri di container, come Docker Hub o registri di terze parti. Ciò consente alle organizzazioni di mantenere un modello di distribuzione coerente in ambienti ibridi o multi-cloud, utilizzando Amazon ECR come repository centrale per le immagini dei container.

Controllo e sicurezza degli accessi

Amazon ECR fornisce meccanismi di controllo degli accessi dettagliati, che consentono alle organizzazioni di controllare chi può inviare o estrarre immagini di container dal registro. Si integra inoltre con AWS Identity and Access Management le funzioni di autenticazione e autorizzazione, garantendo un accesso sicuro alle immagini dei container. Per ulteriori informazioni, consulta Sicurezza in Amazon Elastic Container Registry.

Scansione delle vulnerabilità delle immagini

Amazon ECR offre la scansione automatica delle immagini dei container per individuare vulnerabilità del software e potenziali errori di configurazione, contribuendo a mantenere un ambiente container sicuro e conforme. Per ulteriori informazioni, consulta Scansiona le immagini per individuare le vulnerabilità del software in Amazon ECR.

Registro privato dei container

Per le organizzazioni con requisiti di sicurezza o conformità rigorosi, Amazon ECR può essere utilizzato come registro di container privato, garantendo che le immagini sensibili dei container non siano esposte a registri pubblici e siano accessibili solo all'interno dell'ambiente dell'organizzazione. AWS Per ulteriori informazioni, consulta Registro privato Amazon ECR.

Distribuzione di applicazioni distribuita a livello globale con Amazon ECR Replication

Sfruttando la capacità di replica di Amazon ECR, puoi centralizzare le immagini delle tue applicazioni Web containerizzate in un repository principale, abilitando la distribuzione automatizzata su più AWS regioni, garantendo distribuzioni globali coerenti con bassa latenza in tutto il mondo e riducendo il carico operativo. Per ulteriori informazioni, consulta Private image replication in Amazon ECR

Pulizia automatica delle immagini obsolete dei container

Le policy del ciclo di vita di Amazon ECR consentono la pulizia automatica delle immagini obsolete dei container in base a regole definite come età, numero o tag, ottimizzando i costi di storage, mantenendo un registro organizzato, migliorando la sicurezza e la conformità e semplificando i flussi di lavoro di sviluppo attraverso l'automazione. Per ulteriori informazioni, consulta Automatizza la pulizia delle immagini utilizzando le politiche del ciclo di vita in Amazon ECR

Caratteristiche di Amazon ECR

Amazon ECR offre le seguenti caratteristiche:

- Le policy relative al ciclo di vita consentono di gestire il ciclo di vita delle immagini nei repository. Definisci le regole che determinano la pulizia delle immagini inutilizzate. È possibile testare le regole prima di applicarle al repository. Per ulteriori informazioni, consulta Automatizza la pulizia delle immagini utilizzando le politiche del ciclo di vita in Amazon ECR.
- La scansione delle immagini aiuta a identificare le vulnerabilità del software nelle immagini del container. Ogni repository può essere configurato per la scansione su invio. In tal modo viene

eseguita la scansione di ogni nuova immagine inserita nel repository. È quindi possibile recuperare i risultati della scansione delle immagini. Per ulteriori informazioni, consulta Scansiona le immagini per individuare le vulnerabilità del software in Amazon ECR.

- La replica tra regioni e account consente di disporre più facilmente delle immagini quando è necessario. La configurazione è come un'impostazione del registro ed è per ogni specifica regione. Per ulteriori informazioni, consulta Impostazioni del registro privato in Amazon ECR.
- Le regole di cache pull-through forniscono un modo per memorizzare nel registro privato Amazon ECR la cache dei repository di un registro upstream. Utilizzando una regola di cache pull-through, Amazon ECR si rivolgerà periodicamente al registro upstream per garantire che l'immagine memorizzata nella cache nel registro privato Amazon ECR sia aggiornata. Per ulteriori informazioni, consulta Sincronizzazione di un registro upstream con un registro privato Amazon ECR.
- I modelli di creazione di repository ti consentono di definire le impostazioni per i repository creati da Amazon ECR per tuo conto durante le azioni di pull through cache, create on push o replica. Puoi specificare l'immutabilità dei tag, la configurazione della crittografia, le politiche dei repository, le politiche del ciclo di vita e i tag delle risorse per i repository creati automaticamente. Per ulteriori informazioni, consulta Modelli per controllare gli archivi creati durante un'azione di pull through cache, di creazione in modalità push o di replica.
- La firma gestita genera automaticamente firme crittografiche quando le immagini vengono inviate ad Amazon ECR, semplificando la firma delle immagini dei container. Per ulteriori informazioni, consulta Firma gestita.

Nozioni di base su Amazon ECR

Se utilizzi Amazon Elastic Container Service (Amazon ECS) o Amazon Elastic Kubernetes Service (Amazon EKS), tieni presente che la configurazione per questi due servizi è simile alla configurazione per Amazon ECR perché Amazon ECR è un'estensione di entrambi i servizi.

Quando utilizzi Amazon ECR, utilizza una versione di Amazon ECR AWS CLI che supporti le funzionalità più recenti di Amazon ECR. AWS Command Line Interface Se non vedi il supporto per una funzionalità di Amazon ECR nel AWS CLI, esegui l'upgrade alla versione più recente di AWS CLI. Per informazioni sull'installazione della versione più recente di AWS CLI, consulta Installare o aggiornare alla versione più recente di AWS CLI nella Guida per l'AWS Command Line Interface utente.

Per informazioni su come inviare un'immagine del contenitore a un repository Amazon ECR privato utilizzando AWS CLI e Docker, consulta. Spostamento di un'immagine durante il suo ciclo di vita in Amazon ECR

Prezzi di Amazon ECR

Con Amazon ECR, paghi in base alla quantità di dati archiviati nei tuoi repository, al trasferimento di dati da immagini push e pull e alle azioni relative alle immagini che scegli di eseguire, come la firma e la replica delle immagini. Per ulteriori informazioni, consulta la pagina dei prezzi di Amazon ECR.

Spostamento di un'immagine durante il suo ciclo di vita in Amazon ECR

Se utilizzi Amazon ECR per la prima volta, segui i passaggi seguenti con la CLI Docker e poi AWS CLI per creare un'immagine di esempio, autenticarti nel registro predefinito e creare un repository privato. Quindi invia un'immagine e recupera un'immagine dal repository privato. Quando hai finito con l'immagine di esempio, elimina l'immagine campione e il repository.

Per utilizzare il Console di gestione AWS anziché il AWS CLI, consultathe section called "Creazione di un repository per archiviare immagini".

Per ulteriori informazioni sugli altri strumenti disponibili per la gestione AWS delle risorse, inclusi i diversi AWS SDKs toolkit IDE e gli strumenti da riga di PowerShell comando di Windows, consulta http://aws.amazon.com/tools/.

Prerequisiti

Se non hai la versione più recente AWS CLI e Docker installata e pronta all'uso, segui la procedura seguente per installare entrambi questi strumenti.

Installa il AWS CLI

Per utilizzarlo AWS CLI con Amazon ECR, installa la AWS CLI versione più recente. Per informazioni, consulta Installazione dell' AWS Command Line Interface nella Guida per l'utente di AWS Command Line Interface .

Installa Docker

Docker è disponibile per diversi sistemi operativi, compresa la maggior parte delle distribuzioni Linux, ad esempio Ubuntu, e persino per macOS e Windows. Per ulteriori informazioni sull'installazione di Docker sul tuo specifico sistema operativo, consulta la guida all'installazione di Docker.

Per l'utilizzo di Docker non è necessario un sistema di sviluppo locale. Se utilizzi EC2 già Amazon, puoi avviare un'istanza Amazon Linux 2023 e installare Docker per iniziare.

Se hai già installato Docker, passa a Fase 1: creazione di un'immagine Docker.

Per installare Docker su un' EC2 istanza Amazon utilizzando un'AMI Amazon Linux 2023

1. Avvia un'istanza con l'ultima AMI Amazon Linux 2023. Per ulteriori informazioni, consulta Launching an Instance nella Amazon EC2 User Guide.
2. Connettiti alla tua istanza. Per ulteriori informazioni, consulta Connect to Your Linux Instance nella Amazon EC2 User Guide.
3. Aggiorna i pacchetti installati e la cache dei pacchetti sulla tua istanza.

```
sudo yum update -y
```

4. Installa il pacchetto Docker Community Edition più recente.

```
sudo yum install docker
```

5. Avvia il servizio Docker.

```
sudo service docker start
```

6. Aggiungi `ec2-user` al gruppo `docker` in modo da poter eseguire comandi Docker senza utilizzare `sudo`.

```
sudo usermod -a -G docker ec2-user
```

7. Esci e ripeti l'accesso per trovare il nuovo gruppo di autorizzazioni `docker`. A questo scopo, puoi chiudere la finestra del terminale SSH corrente e riconnetterti all'istanza in una nuova finestra. La nuova sessione SSH avrà le autorizzazioni del gruppo `docker` appropriate.
8. Verifica che `ec2-user` possa eseguire i comandi Docker senza `sudo`.

```
docker info
```

> **Note**
>
> In alcuni casi, l'assegnazione delle autorizzazioni necessarie a `ec2-user` per accedere al daemon Docker può richiedere il riavvio dell'istanza. Prova a riavviare l'istanza se visualizzi questo errore:

```
Cannot connect to the Docker daemon. Is the docker daemon running on this
 host?
```

Fase 1: creazione di un'immagine Docker

In questo passaggio, crei un'immagine Docker di una semplice applicazione Web e la testi sul tuo sistema locale o sull' EC2 istanza Amazon.

Per creare un'immagine Docker di una semplice applicazione Web

1. Crea un file denominato `Dockerfile`. Un Dockerfile è un file manifest che descrive l'immagine di base da utilizzare per l'immagine Docker, nonché gli elementi da installare ed eseguire su di essa. Per ulteriori informazioni sui Dockerfile, consulta la documentazione di riferimento sui Dockerfile.

   ```
   touch Dockerfile
   ```

2. Modifica il `Dockerfile` appena creato e aggiungi i seguenti contenuti.

   ```
   FROM public.ecr.aws/amazonlinux/amazonlinux:latest

   # Install dependencies
   RUN yum update -y && \
    yum install -y httpd

   # Install apache and write hello world message
   RUN echo 'Hello World!' > /var/www/html/index.html

   # Configure apache
   RUN echo 'mkdir -p /var/run/httpd' >> /root/run_apache.sh && \
    echo 'mkdir -p /var/lock/httpd' >> /root/run_apache.sh && \
    echo '/usr/sbin/httpd -D FOREGROUND' >> /root/run_apache.sh && \
    chmod 755 /root/run_apache.sh

   EXPOSE 80

   CMD /root/run_apache.sh
   ```

Questo Dockerfile utilizza l'immagine pubblica di Amazon Linux 2 ospitata su Amazon ECR Public. Le istruzioni `RUN` aggiornano le cache dei pacchetti, installano alcuni pacchetti software per il server Web e infine scrivono il contenuto "Hello World!" nella root del documento del server Web. L'istruzione `EXPOSE` espone la porta 80 nel container, mentre l'istruzione `CMD` avvia il server Web.

3. Crea l'immagine Docker dal tuo Dockerfile.

> **Note**
>
> In alcune versioni di Docker, il seguente comando potrebbe richiedere il percorso completo al Dockerfile anziché il percorso relativo mostrato di seguito.

```
docker build -t hello-world .
```

4. Elenca l'immagine del container.

```
docker images --filter reference=hello-world
```

Output:

```
REPOSITORY          TAG                 IMAGE ID            CREATED
 SIZE
hello-world         latest              e9ffedc8c286        4 minutes ago
 194MB
```

5. Esegui l'immagine appena creata. L'opzione `-p 80:80` mappa la porta 80 esposta sul container alla porta 80 sul sistema host. Per ulteriori informazioni sul comando docker run, consulta la documentazione di riferimento di Docker run.

```
docker run -t -i -p 80:80 hello-world
```

> **Note**
>
> L'output dal server Web Apache viene visualizzato nella finestra del terminale. Puoi ignorare il messaggio "`Could not reliably determine the fully qualified domain name`".

6. Apri un browser e accedi al server su cui è in esecuzione Docker e che ospita il tuo container.

   - Se utilizzi un' EC2 istanza, questo è il valore DNS pubblico per il server, che è lo stesso indirizzo che usi per connetterti all'istanza con SSH. Assicurati che il gruppo di sicurezza per l'istanza consenta il traffico in entrata sulla porta 80.
   - Se Docker è in esecuzione in locale, accedi con il browser a http://localhost/.
   - Se lo utilizzi docker-machine su un computer Windows o Mac, trova l'indirizzo IP della macchina VirtualBox virtuale che ospita Docker con il docker-machine ip comando, sostituendolo *`machine-name`* con il nome della macchina docker che stai utilizzando.

     ```
     docker-machine ip machine-name
     ```

   Visualizzerai una pagina Web con il tuo contenuto "Hello World!" dichiarazione.

7. Interrompi il container Docker digitando Ctrl+c.

Fase 2: Creare un repository

Ora che hai un'immagine da inviare ad Amazon ECR, devi creare un repository in cui memorizzarla. In questo esempio, viene creato un archivio denominato `hello-repository` in cui successivamente inviare l'immagine `hello-world:latest`. Per creare un repository, eseguire il comando seguente:

```
aws ecr create-repository \
    --repository-name hello-repository \
    --region region
```

Fase 3: Effettua l'autenticazione nel registro predefinito

Dopo aver installato e configurato AWS CLI, autentica la CLI Docker nel registro predefinito. In questo modo, il comando docker può inviare ed estrarre le immagini con Amazon ECR. AWS CLI Fornisce un get-login-password comando per semplificare il processo di autenticazione.

Per autenticare Docker in un registro Amazon ECR con get-login-password, esegui il comando. aws ecr get-login-password Quando si passa il token di autenticazione al comando docker login, usare il valore `AWS` per il nome utente e specificare l'URI di registro Amazon ECR a cui si desidera autenticare. Se si esegue l'autenticazione a più registri, è necessario ripetere il comando per ogni registro di sistema.

> **Important**
>
> Se viene visualizzato un errore, installare o eseguire l'upgrade alla versione più recente dell' AWS CLI. Per ulteriori informazioni, consulta Installazione dell' AWS Command Line Interface nella Guida per l'utente dell'AWS Command Line Interface .

- get-login-password (AWS CLI)

```
aws ecr get-login-password --region region | docker login --username AWS --password-
stdin aws_account_id.dkr.ecr.region.amazonaws.com
```

- Comando Get () ECRLoginAWS Tools for Windows PowerShell

```
(Get-ECRLoginCommand).Password | docker login --username AWS --password-
stdin aws_account_id.dkr.ecr.region.amazonaws.com
```

Fase 4: invio di un'immagine ad Amazon ECR

Ora puoi inviare la tua immagine al repository Amazon ECR creato nella sezione precedente. Utilizza la docker CLI per inviare immagini dopo aver soddisfatto i seguenti prerequisiti:

- È installata la versione minima docker di: 1.7.
- Il token di autorizzazione Amazon ECR è stato configurato condocker login.
- Il repository Amazon ECR è stato creato e l'utente ha accesso per eseguire l'invio al repository stesso.

Dopo che tali prerequisiti sono stati soddisfatti, puoi inviare l'immagine al repository appena creato nel registro predefinito del tuo account.

Per assegnare un tag a un'immagine e inviarla ad Amazon ECR

1. Elencare le immagini memorizzate localmente per identificare l'immagine a cui aggiungere il tag e inviare.

   ```
   docker images
   ```

   Output:

   ```
   REPOSITORY          TAG                 IMAGE ID            CREATED
    VIRTUAL SIZE
   hello-world         latest              e9ffedc8c286        4 minutes ago
    241MB
   ```

2. Aggiungere un tag all'immagine da inviare al tuo repository.

   ```
   docker tag hello-world:latest aws_account_id.dkr.ecr.region.amazonaws.com/hello-
   repository
   ```

3. Invia l'immagine.

   ```
   docker push aws_account_id.dkr.ecr.region.amazonaws.com/hello-repository:latest
   ```

   Output:

   ```
   The push refers to a repository [aws_account_id.dkr.ecr.region.amazonaws.com/hello-
   repository] (len: 1)
   e9ae3c220b23: Pushed
   a6785352b25c: Pushed
   0998bf8fb9e9: Pushed
   0a85502c06c9: Pushed
   latest: digest: sha256:215d7e4121b30157d8839e81c4e0912606fca105775bb0636EXAMPLE
    size: 6774
   ```

Fase 5: estrazione di un'immagine da Amazon ECR

Dopo che l'immagine è stata inserita nel tuo repository Amazon ECR, puoi recuperarla da altre posizioni. Utilizza la docker CLI per estrarre le immagini dopo aver soddisfatto i seguenti prerequisiti:

- È installata la versione minima docker di: 1.7.
- Il token di autorizzazione Amazon ECR è stato configurato condocker login.
- Il repository Amazon ECR è stato creato e l'utente ha accesso per eseguire l'estrazione dal repository stesso.

Dopo che tali prerequisiti sono stati soddisfatti, si può estrarre l'immagine. Per estrarre l'immagine di esempio da Amazon ECR, eseguire il comando seguente:

```
docker pull aws_account_id.dkr.ecr.region.amazonaws.com/hello-repository:latest
```

Output:

```
latest: Pulling from hello-repository
0a85502c06c9: Pull complete
0998bf8fb9e9: Pull complete
a6785352b25c: Pull complete
e9ae3c220b23: Pull complete
Digest: sha256:215d7e4121b30157d8839e81c4e0912606fca105775bb0636EXAMPLE
Status: Downloaded newer image for aws_account_id.dkr.region.amazonaws.com/hello-
repository:latest
```

Fase 6: eliminazione di un'immagine

Se non hai più bisogno di un'immagine in uno dei tuoi repository, puoi eliminarla. Per eliminare un'immagine, specifica il repository in cui si trova e un `imageDigest` valore `imageTag` or per l'immagine. L'esempio seguente elimina un'immagine dal `hello-repository` repository con il tag image. `latest` Per eliminare l'immagine di esempio dal repository, esegui il comando seguente:

```
aws ecr batch-delete-image \
      --repository-name hello-repository \
      --image-ids imageTag=latest \
      --region region
```

Fase 7: eliminazione di un repository

Se non è più necessario un intero archivio di immagini, è possibile eliminarlo. L'esempio seguente utilizza il `--force` flag per eliminare un archivio che contiene immagini. Per eliminare un repository che contiene immagini (e tutte le immagini in esso contenute), esegui il comando seguente:

```
aws ecr delete-repository \
      --repository-name hello-repository \
      --force \
      --region region
```

Ottimizzazione delle prestazioni per Amazon ECR

Puoi utilizzare i seguenti consigli su impostazioni e strategie per ottimizzare le prestazioni quando usi Amazon ECR.

Scegli Docker 1.10 e versioni successive per cogliere i vantaggi degli upload di livelli simultanei

Le immagini Docker sono composte da livelli, ovvero fasi intermedie di compilazione dell'immagine. Ciascuna riga in un Dockerfile porta alla creazione di un nuovo livello. Quando utilizzi Docker 1.10 e versioni successive, per impostazione predefinita Docker invia il maggior numero possibile di livelli come upload simultanei a Amazon ECR, pertanto gli upload risultano più veloci.

Utilizza un'immagine base più piccola

Le immagini predefinite disponibili tramite Docker Hub possono contenere molte dipendenze di cui la tua applicazione non ha bisogno. Potresti pertanto utilizzare un'immagine più piccola creata e gestita da altri nella community Docker oppure creare la tua immagine base utilizzando l'immagine scratch minima di Docker. Per ulteriori informazioni, consulta Creare un'immagine base nella documentazione Docker.

Posiziona le dipendenze che sono meno soggette a variazione all'inizio nel tuo Dockerfile

Docker memorizza i livelli nella cache velocizzando i tempi di compilazione. Se nulla è cambiato in un livello dall'ultima compilazione, Docker utilizza la versione nella cache invece di ricompilare il livello. Tuttavia, ciascun livello dipende dai livelli che lo hanno preceduto. Se un livello cambia, Docker ricompila non solo quel livello, ma anche tutti i livelli che vengono dopo.

Per ridurre al minimo il tempo necessario per ricompilare un file Docker e per ricaricare i livelli, è utile posizionare le dipendenze che cambiano con minore frequenza all'inizio nel Dockerfile. Posiziona quelle che cambiano rapidamente (come il codice sorgente dell'applicazione) più avanti nello stack.

Concatena i comandi per evitare lo storage di file non necessari

I file intermedi creati su un livello rimangono parte di quel livello anche se vengono eliminati in un livello successivo. Considera il seguente esempio:

```
WORKDIR /tmp
RUN wget http://example.com/software.tar.gz
RUN wget tar -xvf software.tar.gz
```

```
RUN mv software/binary /opt/bin/myapp
RUN rm software.tar.gz
```

In questo esempio i livelli creati dal primo e dal secondo comando RUN contengono il file .tar.gz originale e tutti i suoi componenti non compressi anche se il file .tar.gz viene eliminato dal quarto comando RUN. Questi comandi possono essere concatenati in un unico comando RUN affinché i file non necessari non facciano parte dell'immagine Docker finale:

```
WORKDIR /tmp
RUN wget http://example.com/software.tar.gz &&\
    wget tar -xvf software.tar.gz &&\
    mv software/binary /opt/bin/myapp &&\
    rm software.tar.gz
```

Utilizza l'endpoint regionale più vicino

Puoi ridurre la latenza nell'estrazione delle immagini da Amazon ECR assicurandoti di utilizzare l'endpoint regionale più vicino al luogo in cui la tua applicazione è in esecuzione. Se la tua applicazione è in esecuzione su un' EC2 istanza Amazon, puoi utilizzare il seguente codice shell per ottenere la regione dalla zona di disponibilità dell'istanza:

```
REGION=$(curl -s http://169.254.169.254/latest/meta-data/placement/availability-zone
 |\
    sed -n 's/\(\d*\)[a-zA-Z]*$/\1/p')
```

La regione può essere passata ai AWS CLI comandi utilizzando il --region parametro o impostata come regione predefinita per un profilo utilizzando il aws configure comando. È inoltre possibile impostare la regione quando si effettuano chiamate utilizzando l' AWS SDK. Per ulteriori informazioni, consulta la documentazione sugli SDK per il tuo linguaggio di programmazione specifico.

Effettuare richieste ai registri Amazon ECR

Puoi inviare, estrarre, eliminare, visualizzare e gestire immagini OCI, immagini Docker e artefatti compatibili con OCI nei registri privati di Amazon ECR utilizzando endpoint solo IPv4 o endpoint dual-stack (and). IPv4 IPv6 Per effettuare richieste dalle reti, puoi utilizzare dual-stack o endpoint. IPv4 IPv4 Per effettuare richieste da una IPv6 rete, utilizza un endpoint dual-stack. Per ulteriori informazioni su come effettuare richieste ai registri pubblici di Amazon ECR utilizzando IPv4 endpoint dual-stack, consulta Effettuare richieste ai registri pubblici Amazon ECR. Non sono previsti costi aggiuntivi per l'accesso ad Amazon ECR tramite IPv6. Per ulteriori informazioni sui prezzi, consulta i prezzi di Amazon Elastic Container Registry.

Gli endpoint Amazon ECR sono designati da attributi che non supportano IPv4 solo endpoint o endpoint dual-stack. Questi attributi possono includere:

- Regione: ogni endpoint è specifico per una regione.
- Tipo: la selezione dell'endpoint dipende dal fatto che si utilizzi l' AWS SDK o le interfacce a riga di comando compatibili con OCI e Docker.
- Sicurezza: in alcune regioni Amazon ECR offre endpoint conformi a FIPS. Per ulteriori informazioni su un elenco di endpoint Amazon ECR conformi a FIPS, consulta Federal Information Processing Standard (FIPS) 140-3.

Per ulteriori informazioni sugli endpoint di servizio supportati dal client dual-stack IPv4, Docker e OCI, che gestisce le chiamate API Amazon ECR dalla AWS CLI, consulta Service endpoints. AWS SDKs

Inizia a fare richieste su IPv6

Per effettuare una richiesta a un registro Amazon ECR IPv6, devi utilizzare un endpoint dual-stack. Prima di accedere a un registro Amazon ECR IPv6, verifica i seguenti requisiti:

- Il client e la rete devono supportare IPv6.
- Amazon ECR supporta i seguenti tipi di richieste su IPv6:
  - Richieste client OCI e Docker:

    ```
    <registry-id>.dkr-ecr.<aws-region>.on.aws
    ```

  - AWS Richieste API:

`ecr.<aws-region>.api.aws`

- È necessario aggiornare tutte le politiche AWS Identity and Access Management (IAM) o di registro che utilizzano il filtraggio degli indirizzi IP di origine per includere gli intervalli di IPv6 indirizzi. Per ulteriori informazioni, consulta Utilizzo IPv6 degli indirizzi nelle politiche IAM.
- Quando si utilizza IPv6, i registri di accesso al server visualizzano gli `Remote IP` indirizzi in IPv6 formato. Aggiorna gli strumenti, gli script e il software esistenti per analizzare questi indirizzi IPv6 IP in formato.

> **Note**
>
> Se riscontri problemi relativi alla presenza di IPv6 indirizzi nei file di registro, contatta. Supporto AWS

Test di compatibilità degli indirizzi IP

Se utilizzi use Linux/Unix o Mac OS X, puoi verificare se puoi accedere a un endpoint dual-stack IPv6 utilizzando il `curl` comando come mostrato nell'esempio seguente:

Example

```
curl --verbose https://ecr.us-west-2.api.aws
```

Vengono restituite informazioni simili a quelle indicate nell'esempio seguente. Se sei connesso tramite IPv6 l'indirizzo IP connesso sarà un indirizzo. IPv6

```
* About to connect() to ecr.us-west-2.api.aws port 443 (#0)
* Trying IPv6 address... connected
* Connected to ecr.us-west-2.api.aws (IPv6 address) port 443 (#0)
> Host: ecr.us-west-2.api.aws
* Request completely sent off
```

Se si utilizza Microsoft Windows 7 o Windows 10, è possibile verificare se è possibile accedere a un endpoint dual-stack tramite IPv4 o IPv6 utilizzando il `ping` comando come illustrato nell'esempio seguente.

```
ping ecr.us-west-2.api.aws
```

Effettuare richieste utilizzando endpoint dual-stack IPv6

Puoi effettuare chiamate API Amazon ECR IPv6 utilizzando endpoint dual-stack. La funzionalità e le prestazioni delle operazioni dell'API Amazon ECR rimangono costanti indipendentemente dal fatto che utilizzi IPv4 o IPv6.

Quando usi AWS Command Line Interface (AWS CLI) and AWS SDKs, puoi IPv6 abilitarlo utilizzando un parametro o un flag per passare a un endpoint dual-stack o specificando direttamente l'endpoint dual-stack nel tuo file di configurazione per sovrascrivere l'endpoint Amazon ECR predefinito. Puoi anche apportare modifiche alla configurazione utilizzando un comando impostato su true nel profilo predefinito. `use_dualstack_endpoint` Per ulteriori informazioni su`use_dualstack_endpoint`, consulta Endpoint Dual-stack e FIPS.

Example Apportare modifiche alla configurazione utilizzando un comando

```
aws configure set default.ecr.use_dualstack_endpoint true
```

Example Effettuare richieste anziché IPv6 utilizzare AWS CLI

```
aws ecr describe-repositories --region us-west-2 --endpoint-url https://
ecr.us-west-2.api.aws
```

Utilizzo degli endpoint Amazon ECR dalla CLI docker

Dopo aver effettuato l'accesso al tuo repository Amazon ECR e taggato l'immagine, puoi inviare ed estrarre immagini OCI e immagini Docker da e verso i registri Amazon ECR. Gli esempi seguenti mostrano i comandi docker push e docker pull con entrambi gli endpoint dual-stack.

Example Invio di immagini docker tramite endpoint IPv4

```
docker push <registry-id>.dkr.ecr.us-west-1.amazonaws.com/my-repository:tag
```

Example Invio di immagini docker utilizzando un endpoint dual-stack

```
docker push <registry-id>.dkr-ecr.us-west-1.on.aws/my-repository:tag
```

Example Estrazione di immagini docker utilizzando l'endpoint IPv4

```
docker pull <registry-id>.dkr.ecr.us-west-1.amazonaws.com/my-repository:tag
```

Example Estrazione di immagini docker utilizzando un endpoint dual-stack

```
docker pull <registry-id>.dkr-ecr.us-west-1.on.aws/my-repository:tag
```

Utilizzo IPv6 degli indirizzi nelle politiche IAM

Prima di accedere a un registro utilizzando IPv6, assicurati che le politiche del tuo utente IAM e del registro Amazon ECR che utilizzano il filtro degli indirizzi IP includano intervalli di IPv6 indirizzi. Se le politiche di filtraggio degli indirizzi IP non vengono aggiornate per gestire IPv6 gli indirizzi, i client potrebbero erroneamente perdere o accedere al registro quando iniziano a utilizzarlo. IPv6 Per ulteriori informazioni sulla gestione delle autorizzazioni di accesso con IAM, consulta Identity and Access Management per Amazon Elastic Container Registry.

Le policy IAM che filtrano gli indirizzi IP utilizzano gli operatori di condizione degli indirizzi IP. Il seguente esempio di politica del registro mostra come identificare l'`54.240.143.*`intervallo di IPv4 indirizzi consentiti utilizzando gli operatori delle condizioni degli indirizzi IP. A tutti gli indirizzi IP al di fuori di questo intervallo viene negato l'accesso al registro (`exampleregistry`). Poiché tutti IPv6 gli indirizzi non rientrano nell'intervallo consentito, questa politica impedisce l'accesso IPv6 agli indirizzi`exampleregistry`.

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Sid": "IPAllow",
      "Effect": "Allow",
      "Principal": "*",
      "Action": "ecr:*",
      "Resource": "arn:aws:ecr:*:*:repository/exampleregistry/*",
      "Condition": {
         "IpAddress": {"aws:SourceIp": "54.240.143.0/24"}
      }
    }
  ]
}
```

Per consentire sia gli IPv4 intervalli di indirizzi IPv6 (`2001:DB8:1234:5678::/64`) che (), modificate l'elemento Condition della politica di registro come mostrato nell'esempio seguente. `54.240.143.0/24` Puoi utilizzare questo formato a `Condition` blocchi per aggiornare sia le politiche degli utenti che quelle del registro di sistema IAM.

```
      "Condition": {
        "IpAddress": {
           "aws:SourceIp": [
             "54.240.143.0/24",
              "2001:DB8:1234:5678::/64"
           ]
         }
       }
```

> **Important**
>
> Prima di IPv6 utilizzarlo, devi aggiornare tutte le politiche relative agli utenti e al registro IAM che utilizzano il filtraggio degli indirizzi IP. Non è consigliabile utilizzare il filtro degli indirizzi IP nelle politiche del registro.

Puoi rivedere le tue policy utente IAM utilizzando la console IAM all'indirizzo https://console.aws.amazon.com/iam/. Per ulteriori informazioni su IAM, consulta la Guida per l'utente di IAM.

Registro privato Amazon ECR

Il registro privato Amazon ECR ospita le immagini di container in un'architettura altamente disponibile e scalabile. È possibile utilizzare il registro privato per gestire i repository di immagini privati costituiti da immagini e artefatti Docker e Open Container Initiative (OCI). Ciascun account AWS dispone di un registro Amazon ECR privato predefinito. Per ulteriori informazioni sui registri pubblici Amazon ECR, consulta Registri pubblici nella Guida per l'utente di Amazon Elastic Container Registry.

Concetti dei registri privati

- L'URL del registro privato predefinito è `https://aws_account_id.dkr.ecr.region.amazonaws.com`.
- Per impostazione predefinita l'account ha accesso in lettura e in scrittura ai repository contenuti nel registro privato. Tuttavia, gli utenti richiedono le autorizzazioni per effettuare chiamate ad Amazon ECR APIs e inviare o estrarre immagini da e verso i tuoi repository privati. Amazon ECR fornisce diverse policy gestite per controllare l'accesso degli utenti a diversi livelli. Per ulteriori informazioni, consulta Esempi di policy basate su Identità di Amazon Elastic Container Registry.
- Devi autenticare il tuo client Docker nel tuo registro privato in modo da poter utilizzare i comandi docker push e docker pull per inviare ed estrarre immagini dai repository in quel registro. Per ulteriori informazioni, consulta Autenticazione del registro privato in Amazon ECR.
- I repository privati possono essere controllati sia tramite le policy di accesso degli utenti sia con le policy relative ai repository stessi. Per ulteriori informazioni sulle policy dei repository, consulta Politiche di archivio privato in Amazon ECR.
- I repository del tuo registro privato possono essere replicati tra AWS regioni nel tuo registro privato e su account separati configurando la replica per il tuo registro privato. Per ulteriori informazioni, consulta Private image replication in Amazon ECR.

Autenticazione del registro privato in Amazon ECR

È possibile utilizzare il Console di gestione AWS AWS CLI, il o il AWS SDKs per creare e gestire archivi privati. Puoi utilizzare questi metodi anche per eseguire alcune operazioni sulle immagini, come elencarle o eliminarle. Questi client utilizzano metodi di AWS autenticazione standard. Anche se tecnicamente puoi utilizzare l'API Amazon ECR per inviare ed estrarre immagini, è più probabile che utilizzerai la CLI Docker o una libreria Docker specifica per la lingua.

La CLI Docker non supporta i metodi di autenticazione IAM nativi. È necessario eseguire ulteriori passaggi affinché Amazon ECR possa autenticare e autorizzare le richieste di invio ed estrazione di Docker.

Sono disponibili i metodi di autenticazione del registro illustrati nel dettaglio nelle sezioni seguenti.

Utilizzo dell'assistente delle credenziali Amazon ECR

Amazon ECR fornisce un supporto per le credenziali Docker che semplifica l'archiviazione e l'utilizzo delle credenziali Docker durante l'invio e l'estrazione delle immagini in Amazon ECR. Per i passaggi di installazione e configurazione, consulta Amazon ECR Docker Credential Helper.

> **Note**
>
> Al momento l'assistente credenziali Amazon ECR Docker non supporta l'autenticazione a più fattori (MFA) con più fattori.

Utilizzo di un token di autorizzazione

L'ambito di autorizzazione di un token di autorizzazione corrisponde a quello dell'entità principale IAM utilizzata per recuperare il token di autenticazione. Un token di autenticazione viene utilizzato per accedere a qualsiasi registro Amazon ECR a cui l'entità principale IAM ha accesso ed è valido per 12 ore. Per ottenere un token di autorizzazione, è necessario utilizzare l'operazione GetAuthorizationTokenAPI per recuperare un token di autorizzazione con codifica Base64 contenente il nome utente e una password codificata. `AWS` Il AWS CLI `get-login-password` comando semplifica questa operazione recuperando e decodificando il token di autorizzazione che è quindi possibile reindirizzare a un comando per l'autenticazione. docker login

Per autenticare Docker su un registro privato Amazon ECR con get-login

- Per autenticare Docker in un registro Amazon ECR con get-login-password, esegui il comando. aws ecr get-login-password Quando si passa il token di autenticazione al comando docker login, usare il valore `AWS` per il nome utente e specificare l'URI di registro Amazon ECR a cui si desidera autenticare. Se si esegue l'autenticazione a più registri, è necessario ripetere il comando per ogni registro di sistema.

> **Important**
>
> Se viene visualizzato un errore, installare o eseguire l'upgrade alla versione più recente dell' AWS CLI. Per ulteriori informazioni, consulta Installazione dell' AWS Command Line Interface nella Guida per l'utente dell'AWS Command Line Interface .

- get-login-password (AWS CLI)

```
aws ecr get-login-password --region region | docker login --username AWS --
password-stdin aws_account_id.dkr.ecr.region.amazonaws.com
```

- Comando Get () ECRLoginAWS Tools for Windows PowerShell

```
(Get-ECRLoginCommand).Password | docker login --username AWS --password-
stdin aws_account_id.dkr.ecr.region.amazonaws.com
```

Utilizzo dell'autenticazione API HTTP

Amazon ECR supporta l'API HTTP del registro Docker. Tuttavia, poiché Amazon ECR è un registro privato, devi fornire un token di autorizzazione con ogni richiesta HTTP. È possibile aggiungere un'intestazione di autorizzazione HTTP utilizzando l'`-H`opzione for curl e passare il token di autorizzazione fornito dal get-authorization-token AWS CLI comando.

Per effettuare l'autenticazione con l'API HTTP di Amazon ECR

1. Recupera un token di autorizzazione con AWS CLI e impostalo su una variabile di ambiente.

```
TOKEN=$(aws ecr get-authorization-token --output text --query
 'authorizationData[].authorizationToken')
```

2. Per effettuare l'autenticazione nell'API, passa la variabile `$TOKEN` all'opzione `-H` di curl. Ad esempio, il comando seguente elenca i tag immagine in un repository Amazon ECR. Per ulteriori informazioni, consulta API HTTP del registro Docker nella documentazione di riferimento.

```
curl -i -H "Authorization: Basic $TOKEN"
 https://aws_account_id.dkr.ecr.region.amazonaws.com/v2/amazonlinux/tags/list
```

L'output è il seguente:

```
HTTP/1.1 200 OK
Content-Type: text/plain; charset=utf-8
Date: Thu, 04 Jan 2018 16:06:59 GMT
Docker-Distribution-Api-Version: registry/2.0
Content-Length: 50
Connection: keep-alive

{"name":"amazonlinux","tags":["2017.09","latest"]}
```

Impostazioni del registro privato in Amazon ECR

Amazon ECR utilizza le impostazioni del registro privato per configurare le funzionalità a livello di registro. Le impostazioni del registro privato sono configurate separatamente per ogni regione. Puoi utilizzare le impostazioni del registro privato per configurare le seguenti funzionalità.

- Autorizzazioni di registro: una politica di autorizzazioni del registro fornisce il controllo sulla replica e consente di utilizzare le autorizzazioni della cache. Per ulteriori informazioni, consulta Autorizzazioni di registro private in Amazon ECR.
- Regole pull through cache: una regola pull through cache viene utilizzata per memorizzare nella cache le immagini da un registro upstream nel registro privato Amazon ECR. Per ulteriori informazioni, consulta Sincronizzazione di un registro upstream con un registro privato Amazon ECR.
- Configurazione di replica: la configurazione di replica viene utilizzata per controllare se i repository vengono copiati su o. Regioni AWS Account AWS Per ulteriori informazioni, consulta Private image replication in Amazon ECR
- Modelli di creazione di repository: un modello di creazione di repository viene utilizzato per definire le impostazioni standard da applicare quando Amazon ECR crea nuovi repository per tuo conto. Ad esempio, i repository creati mediante un'azione pull through cache, la creazione in modalità push o la replica. Per ulteriori informazioni, consulta Modelli per controllare gli archivi creati durante un'azione di pull through cache, di creazione in modalità push o di replica.
- Configurazione della scansione: per impostazione predefinita, il registro è abilitato per la scansione di base. Puoi abilitare la scansione avanzata che offre una modalità di scansione automatica e continua che esegue la scansione delle vulnerabilità del sistema operativo e dei pacchetti del

linguaggio di programmazione. Per ulteriori informazioni, consulta Scansiona le immagini per individuare le vulnerabilità del software in Amazon ECR.

- Esclusione degli aggiornamenti pull-time: è possibile configurare le esclusioni degli aggiornamenti pull-time per evitare che l'ultima ora di pull venga aggiornata per immagini specifiche quando vengono scaricate. Ciò è utile per le immagini utilizzate a scopo di test o per CI/CD scopi in cui non si desidera che il pull time influenzi le decisioni relative alle politiche relative al ciclo di vita. Per ulteriori informazioni, consulta Esclusioni relative agli aggiornamenti a tempo pieno.

Autorizzazioni di registro private in Amazon ECR

Amazon ECR utilizza una policy di registro per concedere le autorizzazioni a un principale AWS a livello di registro privato.

L'ambito viene impostato scegliendo la versione della politica del registro. Esistono due versioni con un diverso ambito dei criteri di registro: versione 1 (V1) e versione 2 (V2). V2 è l'ambito esteso della politica di registro che include tutte le autorizzazioni ECR. Per l'elenco completo delle azioni API, consulta la Amazon ECR API Guide. La versione V2 è l'ambito predefinito della politica di registro. Per ulteriori informazioni sulla visualizzazione o l'impostazione dell'ambito dei criteri di registro, vederePassaggio all'ambito esteso della politica di registro. Per informazioni sulle impostazioni generali per il tuo registro privato Amazon ECR, consultaImpostazioni del registro privato in Amazon ECR.

Le versioni sono dettagliate come segue.

- V1 — Per la versione 1, Amazon ECR applica solo le seguenti autorizzazioni a livello di registro privato.
  - `ecr:ReplicateImage`: concede l'autorizzazione a un altro account, denominato registro di origine, per replicare le immagini nel proprio registro. Questa operazione viene utilizzata solo per la replica tra account.
  - `ecr:BatchImportUpstreamImage`— Concede l'autorizzazione per recuperare l'immagine esterna e importarla nel registro privato.
  - `ecr:CreateRepository` – Concede l'autorizzazione per creare un repository in un registro privato. Questa autorizzazione è necessaria se il repository che archivia le immagini replicate o memorizzate nella cache non esiste già.
- V2 — Per la versione 2, Amazon ECR consente tutte le azioni ECR nella policy e applica la policy del registro in tutte le richieste ECR.

È possibile utilizzare la console o la CLI per visualizzare o modificare l'ambito della politica del registro.

> **Note**
>
> Sebbene sia possibile aggiungere l'`ecr:*`azione a una politica di registro privata, è consigliabile aggiungere solo le azioni specifiche richieste in base alla funzionalità che si sta utilizzando anziché utilizzare un carattere jolly.

Argomenti

Esempi di policy di registro privato per Amazon ECR

I seguenti esempi mostrano le dichiarazioni di policy di autorizzazione del registro che è possibile utilizzare per controllare le autorizzazioni degli utenti per il registro Amazon ECR.

> **Note**
>
> In ogni esempio, se l'`ecr:CreateRepository`azione viene rimossa dalla politica del registro, la replica può comunque avvenire. Tuttavia, per la replica corretta, è necessario creare repository con lo stesso nome all'interno dell'account.

Esempio: consenti a tutti i principali IAM in un account di origine di replicare tutti i repository

La seguente politica di autorizzazione del registro consente a tutti i principali IAM (utenti e ruoli) in un account di origine di replicare tutti i repository.

Tenere presente quanto segue:

- Importante: quando specifichi un Account AWS ID come principale in una policy, concedi l'accesso a tutti gli utenti e i ruoli IAM all'interno di quell'account, non solo all'utente root. Ciò fornisce un ampio accesso all'intero account.
- Considerazioni sulla sicurezza: le autorizzazioni a livello di account concedono l'accesso a tutte le entità IAM nell'account specificato. Per un accesso più restrittivo, specifica i singoli utenti e ruoli IAM o utilizza le istruzioni condizionali per limitare ulteriormente l'accesso.

JSON

```
{
    "Version":"2012-10-17",
    "Statement":[
        {
            "Sid":"ReplicationAccessCrossAccount",
            "Effect":"Allow",
            "Principal":{
                "AWS":"arn:aws:iam::111122223333:root"
            },
            "Action":[
                "ecr:CreateRepository",
                "ecr:ReplicateImage"
            ],
            "Resource": [
                "arn:aws:ecr:us-west-2:444455556666:repository/*"
            ]
        }
    ]
}
```

Esempio: consenti i principali IAM da più account

La seguente politica sulle autorizzazioni del registro contiene due istruzioni. Ogni istruzione consente a tutti i principali IAM (utenti e ruoli) di un account di origine di replicare tutti i repository.

JSON

```
{
    "Version":"2012-10-17",
```

```
    "Statement":[
        {
            "Sid":"ReplicationAccessCrossAccount1",
            "Effect":"Allow",
            "Principal":{
                "AWS":"arn:aws:iam::111122223333:root"
            },
            "Action":[
                "ecr:CreateRepository",
                "ecr:ReplicateImage"
            ],
            "Resource": [
                "arn:aws:ecr:us-west-2:123456789012:repository/*"
            ]
        },
        {
            "Sid":"ReplicationAccessCrossAccount2",
            "Effect":"Allow",
            "Principal":{
                "AWS":"arn:aws:iam::444455556666:root"
            },
            "Action":[
                "ecr:CreateRepository",
                "ecr:ReplicateImage"
            ],
            "Resource": [
                "arn:aws:ecr:us-west-2:123456789012:repository/*"
            ]
        }
    ]
}
```

Esempio: consenti a tutti i principali IAM in un account di origine di replicare tutti i repository con prefisso. **`prod-`**

La seguente politica di autorizzazione del registro consente a tutti i principali IAM (utenti e ruoli) di un account di origine di replicare tutti i repository che iniziano con. `prod-`

JSON

```
{
```

```
    "Version":"2012-10-17",
    "Statement":[
        {
            "Sid":"ReplicationAccessCrossAccount",
            "Effect":"Allow",
            "Principal":{
                "AWS":"arn:aws:iam::111122223333:root"
            },
            "Action":[
                "ecr:CreateRepository",
                "ecr:ReplicateImage"
            ],
            "Resource": [
                "arn:aws:ecr:us-west-2:444455556666:repository/prod-*"
            ]
        }
    ]
}
```

Passaggio all'ambito esteso della politica di registro

> **Important**
>
> Per i nuovi utenti, i registri vengono configurati automaticamente per utilizzare la politica del V2 registro al momento della creazione. Non devi intraprendere alcuna azione. Amazon ECR non consiglia di tornare alla precedente politica di registro. V1

È possibile utilizzare la console o la CLI per visualizzare o modificare l'ambito della politica del registro.

Console di gestione AWS

Utilizza i seguenti passaggi per visualizzare le impostazioni del tuo account. Per visualizzare o aggiornare l'ambito della politica del registro, consulta la procedura CLI in questa pagina.

Attiva la politica di registro avanzata per il tuo registro privato

1. Apri la console Amazon ECR su private-registry/repository https://console.aws.amazon.com/ecr/

2. Dalla barra di navigazione, scegli la regione.
3. Nel riquadro di navigazione, scegli Registro privato, Funzionalità e impostazioni, quindi scegli Autorizzazioni.
4. Nella pagina Autorizzazioni, per la politica del registro, visualizza la tua politica JSON. Se hai la politica V1, viene visualizzato un banner con le istruzioni per l'aggiornamento alla V2. Scegli Abilita .

   Viene visualizzato un banner che indica che l'ambito della politica del registro è stato aggiornato alla V2.
5. Puoi anche configurare facoltativamente le autorizzazioni con la CLI. Per ulteriori informazioni, consulta Impostazioni del registro privato in Amazon ECR.

> **Note**
>
> Per visualizzare o aggiornare l'ambito della politica del registro, consulta la procedura CLI in questa pagina.

AWS CLI

Amazon ECR genera la policy del registro V2. Utilizza i seguenti passaggi per visualizzare o aggiornare l'ambito della politica del registro. Non è possibile visualizzare o modificare l'ambito dei criteri di registro nella console

- Per recuperare la politica del registro attualmente in uso.

```
aws ecr get-account-setting --name REGISTRY_POLICY_SCOPE
```

Il nome del parametro è un campo obbligatorio. Se non fornisci il nome, riceverai il seguente errore:

```
aws: error: the following arguments are required: --name
```

Visualizza l'output del comando Registry Policy. Nell'output di esempio seguente, la versione dei criteri di registro è la V1.

```
{
    "name": "REGISTRY_POLICY_SCOPE",
```

```
    "value": "V1"
}
```

È possibile modificare la versione dei criteri di registro da V1 aV2. La versione V1 non è l'ambito consigliato dei criteri di registro.

```
aws ecr put-account-setting --name REGISTRY_POLICY_SCOPE --value value
```

Ad esempio, utilizzare il comando seguente per eseguire l'aggiornamento alla V2.

```
aws ecr put-account-setting --name REGISTRY_POLICY_SCOPE --value V2
```

Visualizza l'output del comando di policy del registro. Nell'output di esempio seguente, la versione dei criteri di registro è stata aggiornata alla V2.

```
{
    "name": "REGISTRY_POLICY_SCOPE",
    "value": "V2"
}
```

Concessione delle autorizzazioni di registro per la replica tra account in Amazon ECR

Il tipo di policy cross-account viene utilizzato per concedere le autorizzazioni a un AWS principale, permettendo la replica dei repository da un registro di origine al registro dell'utente. Per impostazione predefinita, hai l'autorizzazione per configurare la replica tra regioni all'interno del tuo registro. È necessario configurare le policy di registro solo se si concede a un altro account l'autorizzazione per replicare il contenuto nel registro.

Una policy di registro deve concedere l'autorizzazione per il operazione API `ecr:ReplicateImage`. Questa API è un'API Amazon ECR interna in grado di replicare immagini tra regioni o account. È inoltre possibile concedere l'autorizzazione per il `ecr:CreateRepository`, che consente ad Amazon ECR di creare repository nel registro se non esistono già. Se l'autorizzazione `ecr:CreateRepository` non viene fornita, nel registro deve essere creato manualmente un repository con lo stesso nome del repository di origine. Se nessuno dei due viene creato, la replica ha esito negativo. Eventuali azioni non riuscite `CreateRepository` o relative `ReplicateImage` all'API vengono visualizzate in. CloudTrail

Per configurare una policy delle autorizzazioni per la replica (Console di gestione AWS)

1. Apri la console Amazon ECR all'indirizzo https://console.aws.amazon.com/ecr/.
2. Dalla barra di navigazione, scegli la regione in cui configurare la policy del registro.
3. Nel riquadro di navigazione, scegli Registro privato, scegli Caratteristiche e impostazioni, quindi scegli Autorizzazioni.
4. Alla pagina Registry permissions (Autorizzazioni di registro), scegli Generate statement (Genera istruzione).
5. Per definire la tua istruzione di policy usando il generatore di policy, completa i seguenti passaggi.

   a. Per Tipo di policy, scegli Replica - cross account.

   b. Per ID dichiarazione, inserisci un ID di dichiarazione univoco. Questo campo viene utilizzato come `Sid` sulle policy di registro.

   c. Per Account, inserisci l'account IDs per ogni account a cui desideri concedere le autorizzazioni. Quando specificate più account IDs, separateli con una virgola.
6. Scegli Save (Salva).

Per configurare una policy delle autorizzazioni per la replica (AWS CLI)

1. Creare un file denominato `registry_policy.json` e popolarlo con una policy di registro.

   JSON

```
{
    "Version":"2012-10-17",
    "Statement":[
        {
            "Sid":"ReplicationAccessCrossAccount",
            "Effect":"Allow",
            "Principal":{
                "AWS":"arn:aws:iam::111122223333:root"
            },
            "Action":[
                "ecr:CreateRepository",
                "ecr:ReplicateImage"
            ],
            "Resource": [
```

```
                "arn:aws:ecr:us-west-2:444455556666:repository/*"
            ]
        }
    ]
}
```

2. Creare le policy di registro utilizzando il file delle policy.

```
aws ecr put-registry-policy \
      --policy-text file://registry_policy.json \
      --region us-west-2
```

3. Recuperare le policy di registro da confermare.

```
aws ecr get-registry-policy \
      --region us-west-2
```

Concessione delle autorizzazioni di registro per il pull through cache in Amazon ECR

Le autorizzazioni del registro privato di Amazon ECR possono essere utilizzate per definire le autorizzazioni delle singole entità IAM per utilizzare la cache pull-through. Se un'entità IAM dispone di più autorizzazioni concesse da una policy IAM di quelle concesse dalla policy delle autorizzazioni del registro, la policy IAM ha la precedenza.

Per creare una policy delle autorizzazioni del registro privato (Console di gestione AWS)

1. Apri la console Amazon ECR all'indirizzo https://console.aws.amazon.com/ecr/.
2. Dalla barra di navigazione, scegli la regione in cui configurare l'istruzione delle autorizzazioni del registro privato.
3. Nel riquadro di navigazione, scegli Registro privato, scegli Caratteristiche e impostazioni, quindi scegli Autorizzazioni.
4. Alla pagina Registry permissions (Autorizzazioni di registro), scegli Generate statement (Genera istruzione).
5. Per ogni istruzione delle policy di autorizzazione della cache pull-through che si desidera creare, procedi come segue.

a. Per Policy type (Tipo di policy), scegli Pull through cache policy (Policy della cache pull-through).
b. Per Statement id (ID istruzione), inserisci un nome per la policy dell'istruzione della cache pull-through.
c. Per Entità IAM, specifica gli utenti, i gruppi o i ruoli da includere nella policy.
d. Per lo spazio dei nomi Cache, seleziona la regola pull through cache a cui associare la policy.
e. Per Repository names (Nomi dei repository), specifica il nome di base del repository per cui applicare la regola. Ad esempio, se si desidera specificare il repository Amazon Linux su Amazon ECR Public, il nome del repository sarà `amazonlinux`.

Repository Amazon ECR privati

Un repository privato Amazon ECR contiene immagini Docker, immagini Open Container Initiative (OCI) e artefatti compatibili con OCI. Puoi creare, monitorare ed eliminare repository di immagini e impostare autorizzazioni per controllare chi può accedervi utilizzando le operazioni dell'API Amazon ECR o la sezione Repositories della console Amazon ECR. Amazon ECR si integra anche con la CLI Docker, in modo da poter inviare ed estrarre immagini dai tuoi ambienti di sviluppo ai tuoi repository.

Argomenti

Concetti del repository privato

- Per impostazione predefinita il tuo account ha accesso in lettura e in scrittura ai repository contenuti nel registro predefinito (*aws_account_id*.dkr.ecr.*region*.amazonaws.com). Tuttavia, gli utenti richiedono le autorizzazioni per effettuare chiamate ad Amazon ECR APIs e inviare o estrarre immagini da e verso i tuoi repository. Amazon ECR fornisce diverse policy gestite per controllare l'accesso degli utenti a diversi livelli. Per ulteriori informazioni, consulta Esempi di policy basate su Identità di Amazon Elastic Container Registry.
- I repository possono essere controllati tramite le policy di accesso degli utenti oppure con le policy relative ai singoli repository. Per ulteriori informazioni, consulta Politiche di archivio privato in Amazon ECR.
- I nomi dei repository possono supportare i namespace, utili per raggruppare repository simili. Ad esempio, se vi sono diversi team che utilizzano lo stesso registro, il Team A potrebbe utilizzare il namespace `team-a` mentre il Team B potrebbe utilizzare il namespace `team-b`. In questo modo, ogni team ha la propria immagine chiamata `web-app` con ogni immagine anteposta dallo spazio dei nomi del team. Questa configurazione consente di utilizzare simultaneamente queste immagini su ogni team senza interferenze. L'immagine del team A è `team-a/web-app` e l'immagine del team B è `team-b/web-app`.

- Le immagini possono essere replicate in altri repository tra regioni del proprio registro e tra account. È possibile eseguire questa operazione specificando una configurazione di replica nelle impostazioni del registro. Per ulteriori informazioni, consulta Impostazioni del registro privato in Amazon ECR.

Creazione di un repository privato Amazon ECR per archiviare immagini

> **Important**
>
> La crittografia lato server a doppio livello con AWS KMS (DSSE-KMS) è disponibile solo nelle regioni. AWS GovCloud (US)

Crea un repository privato Amazon ECR, quindi utilizza il repository per archiviare le immagini dei contenitori. Per creare un repository privato usando la Console di gestione AWS, segui i seguenti passaggi.

Per creare un repository (Console di gestione AWS)

1. Apri la console Amazon ECR nei https://console.aws.amazon.com/ecr/repository.
2. Dalla barra di navigazione, scegliere la regione in cui creare il repository.
3. Scegli Archivi privati, quindi scegli Crea repository.
4. Per Repository name (Nome repository), immettere un nome univoco per il repository. Il nome del repository può essere specificato autonomamente (ad esempio `nginx-web-app`). In alternativa, può esservi anteposto uno spazio dei nomi per raggruppare il repository in una categoria (ad esempio `project-a/nginx-web-app`).

   > **Note**
   >
   > Il nome del repository può contenere un massimo di 256 caratteri. Il nome deve iniziare con una lettera e può contenere solo lettere minuscole, numeri, trattini, trattini bassi, punti e barre. L'uso di un doppio trattino, un doppio trattino basso o una doppia barra non è supportato.

5. Per l'immutabilità dei tag Image, scegli una delle seguenti impostazioni di modifica dei tag per il repository.

   - Mutabile: scegliete questa opzione se desiderate che i tag delle immagini vengano sovrascritti. Consigliato per i repository che utilizzano azioni pull through cache per garantire che Amazon ECR possa aggiornare le immagini memorizzate nella cache. Inoltre, per disabilitare gli aggiornamenti dei tag per alcuni tag mutabili, inserisci i nomi dei tag o usa i caratteri jolly (*) per abbinare più tag simili nella casella di testo Esclusione tag mutabili.
   - Immutabile: scegli questa opzione se vuoi evitare che i tag delle immagini vengano sovrascritti e si applica a tutti i tag e le esclusioni presenti nel repository quando inserisci un'immagine con un tag esistente. Amazon ECR restituisce un `ImageTagAlreadyExistsException` messaggio se tenti di inviare un'immagine con un tag esistente. Inoltre, per abilitare gli aggiornamenti dei tag per alcuni tag immutabili, inserisci i nomi dei tag o usa i caratteri jolly (*) per abbinare più tag simili nella casella di testo Esclusione tag immutabile.

   > **Note**
   >
   > Le impostazioni di mutabilità dei tag individuali non sono supportate.

6. Per la configurazione della crittografia, scegli tra AES-256 o. AWS KMS Per ulteriori informazioni, consulta Crittografia dei dati a riposo.

   a. Se AWS KMS è selezionata, scegli tra crittografia a livello singolo e crittografia a doppio strato. Sono previsti costi aggiuntivi per l'utilizzo AWS KMS della crittografia a doppio livello. Per ulteriori informazioni, consulta i prezzi dei servizi Amazon ECR.

   b. Per impostazione predefinita, viene AWS scelta la chiave gestita con l'alias`aws/ecr`. Questa chiave viene creata nel tuo account la prima volta che crei un repository con la AWS KMS crittografia abilitata. Seleziona Chiave gestita dal cliente (avanzata) per scegliere la tua AWS KMS chiave. La AWS KMS chiave deve trovarsi nella stessa regione del cluster. Seleziona Crea una AWS KMS chiave per accedere alla AWS KMS console e creare la tua chiave.

7. Per le impostazioni di scansione delle immagini, sebbene sia possibile specificare le impostazioni di scansione a livello di repository per la scansione di base, è consigliabile specificare la configurazione di scansione a livello di registro privato. La configurazione delle impostazioni di scansione a livello di registro privato consente di scegliere tra la scansione avanzata o la

scansione di base e consente inoltre di definire filtri per specificare quali archivi devono essere analizzati.

8. Scegli Create (Crea).

Per creare un repository (AWS CLI)

1. È possibile creare un repository utilizzando il AWS CLI comando. aws ecr create-repository

```
aws ecr create-repository \
            --repository-name hello-repository \
            --region region
```

2. Se hai definito un modello per la creazione di un repository, puoi creare un repository inserendo la tua immagine utilizzando i familiari comandi push di Amazon ECR con il nome del repository desiderato. Amazon ECR creerà automaticamente il repository per te utilizzando le impostazioni predefinite del modello di creazione del repository. Se non hai ancora definito un modello per la creazione di un repository, la tua richiesta al tuo repository di immagini inesistente avrà esito negativo.

```
docker push aws_account_id.dkr.ecr.region.amazonaws.com/prefix/my-new-
repository:tag
```

Fasi successive

Per visualizzare i passaggi per inviare un'immagine al tuo repository, seleziona il repository e scegli Visualizza i comandi push. Per ulteriori informazioni su come inserire un'immagine nel repository, consulta Invio di un'immagine a un repository privato Amazon ECR.

Visualizzazione dei contenuti e dei dettagli di un repository privato in Amazon ECR

Dopo aver creato un repository privato, puoi visualizzare i dettagli sul repository in: Console di gestione AWS

- Quali immagini sono archiviate in un repository
- I dettagli su ogni immagine memorizzata nel repository, incluse le dimensioni e il digest SHA per ciascuna immagine

- La frequenza di scansione specificata per il contenuto del repository
- Se al repository è associata una regola di cache pull-through attiva
- L'impostazione di crittografia per il repository

> **Note**
>
> A partire dalla versione Docker 1.9, il client Docker comprime i livelli delle immagini prima di inviarli a un registro Docker V2. L'output del comand docker images mostra la dimensione dell'immagine non compressa. Pertanto, tieni presente che Docker potrebbe restituire un'immagine più grande dell'immagine visualizzata in Console di gestione AWS.

Per visualizzare le informazioni relative al repository (Console di gestione AWS)

1. Apri la console Amazon ECR nei https://console.aws.amazon.com/ecr/repository.
2. Dalla barra di navigazione, scegliere la regione in cui si trova il repository da visualizzare.
3. Nel riquadro di navigazione, selezionare Repositories (Repository).
4. Nella pagina Repositories (Repository), seleziona la scheda Private (Privato) e quindi il repository da visualizzare.
5. Nella pagina prodotto del repository, la console viene impostata di default sulla visualizzazione Images (Immagini). Utilizzare il menu di navigazione per visualizzare altre informazioni sul repository.
   - Selezionare Summary (Riepilogo) per visualizzare i dettagli del repository e i dati del pull count per il repository.
   - Selezionare la scheda Images (Immagini) per visualizzare le informazioni sui tag di immagine contenuti nel repository. Per visualizzare ulteriori informazioni sull'immagine, selezionare il tag dell'immagine. Per ulteriori informazioni, consulta Visualizzazione dei dettagli delle immagini in Amazon ECR.

     Se vi sono immagini senza tag che desideri eliminare, puoi selezionare la casella a sinistra dei repository da eliminare e scegliere Delete (Elimina). Per ulteriori informazioni, consulta Eliminazione di un'immagine in Amazon ECR.
   - Selezionare la scheda Permissions (Autorizzazioni) per visualizzare le policy dei repository applicate al repository stesso. Per ulteriori informazioni, consulta Politiche di archivio privato in Amazon ECR.

- Scegliere Lifecycle Policy (Policy ciclo di vita) per visualizzare le regole delle policy del ciclo di vita applicate al repository. Qui viene anche visualizzata la cronologia degli eventi del ciclo di vita. Per ulteriori informazioni, consulta Automatizza la pulizia delle immagini utilizzando le politiche del ciclo di vita in Amazon ECR.
- Selezionare la scheda Tags (Tag) per visualizzare i tag dei metadati applicati al repository.

Eliminazione di un repository privato in Amazon ECR

Al termine dell'utilizzo di un repository, puoi eliminarlo. Quando elimini un repository in Console di gestione AWS, vengono eliminate anche tutte le immagini in esso contenute; questa operazione non può essere annullata.

> **Important**
>
> Vengono eliminate anche le immagini presenti negli archivi eliminati. Questa operazione non può essere annullata.

Per eliminare un repository (Console di gestione AWS)

1. Apri la console Amazon ECR nei https://console.aws.amazon.com/ecr/repository.
2. Dalla barra di navigazione, scegliere la regione in cui si trova il repository da eliminare.
3. Nel riquadro di navigazione, selezionare Repositories (Repository).
4. Nella pagina Repositories (Repository), selezionare la scheda Private (Privato), quindi selezionare il repository da eliminare e scegliere Delete (Elimina).
5. Nella ***repository_name*** finestra Elimina, verifica che i repository selezionati debbano essere eliminati e scegli Elimina.

Politiche di archivio privato in Amazon ECR

Amazon ECR utilizza autorizzazioni basate sulle risorse per controllare l'accesso ai repository. Le autorizzazioni basate sulle risorse consentono di specificare quali utenti o ruoli hanno accesso a un repository e quali azioni possono eseguire sul repository. Per impostazione predefinita, solo l' AWS account che ha creato il repository ha accesso al repository. È possibile applicare una politica di repository che consenta un accesso aggiuntivo al repository.

Argomenti

Policy del repository e policy IAM

Le policy del repository Amazon ECR sono un sottoinsieme delle policy IAM che vengono definite e specificamente utilizzate per controllare l'accesso ai singoli repository Amazon ECR. Le policy IAM sono generalmente utilizzate per applicare le autorizzazioni per l'intero servizio Amazon ECR, ma possono anche essere utilizzate per controllare l'accesso alle risorse specifiche.

Le policy dei repository Amazon ECR e le policy IAM vengono entrambe utilizzate per determinare quali azioni possono essere eseguite da un utente o un ruolo specifico su un repository. Se a un utente o a un ruolo è consentito eseguire un'operazione tramite una policy del repository ma l'autorizzazione gli viene negata da una policy IAM (o viceversa), anche l'operazione viene negata. A un utente o a un ruolo deve essere concessa l'autorizzazione per un'operazione tramite una policy del repository o una policy IAM, ma non entrambe le opzioni per consentire l'operazione.

> **Important**
>
> Amazon ECR richiede che gli utenti dispongano dell'autorizzazione per effettuare chiamate all'API `ecr:GetAuthorizationToken` tramite una policy IAM prima che possano autenticarsi in un registro ed eseguire l'invio o l'estrazione delle immagini da un repository Amazon ECR. Amazon ECR fornisce diverse policy IAM gestite per controllare l'accesso degli utenti a vari livelli. Per ulteriori informazioni, consulta Esempi di policy basate su Identità di Amazon Elastic Container Registry.

È possibile utilizzare questi tipi di policy per controllare l'accesso ai repository, come illustrato negli esempi seguenti.

Questo esempio illustra una policy del repository Amazon ECR che consente a un utente specifico di descrivere il repository e le immagini all'interno del repository.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "ECRRepositoryPolicy",
            "Effect": "Allow",
            "Principal": {"AWS": "arn:aws:iam::111122223333:user/username"},
            "Action": [
                "ecr:DescribeImages",
                "ecr:DescribeRepositories"
            ],
            "Resource": "*"
        }
    ]
}
```

Questo esempio illustra una policy IAM che consente di raggiungere lo stesso obiettivo precedente, definendo l'ambito della policy per un repository (specificato dall'ARN completo del repository) utilizzando il parametro a livello di risorsa. Per maggiori informazioni sul formato Amazon Resource Name (ARN), consulta Resources.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "AllowDescribeRepoImage",
            "Effect": "Allow",
            "Action": [
                "ecr:DescribeImages",
                "ecr:DescribeRepositories"
            ],
            "Resource": ["arn:aws:ecr:us-
east-1:111122223333:repository/repository-name"]
        }
    ]
```

```
}
```

Esempi di policy relative agli archivi privati in Amazon ECR

> **Important**
>
> Gli esempi di policy di repository riportati in questa pagina sono concepiti per essere applicati ai repository privati di Amazon ECR. Non funzioneranno correttamente se utilizzati direttamente con un principale IAM, a meno che non vengano modificati per specificare il repository di Amazon ECR come risorsa. Per ulteriori informazioni sull'impostazione delle policy dei repository, consulta Impostazione di una dichiarazione sulla politica di archiviazione privata in Amazon ECR.

Le policy del repository Amazon ECR sono un sottoinsieme delle policy IAM che vengono definite e specificamente utilizzate per controllare l'accesso ai singoli repository Amazon ECR. Le policy IAM sono generalmente utilizzate per applicare le autorizzazioni per l'intero servizio Amazon ECR, ma possono anche essere utilizzate per controllare l'accesso alle risorse specifiche. Per ulteriori informazioni, consulta Policy del repository e policy IAM .

I seguenti esempi di policy dei repository mostrano le dichiarazioni di autorizzazione che puoi utilizzare per controllare l'accesso ai repository privati di Amazon ECR.

> **Important**
>
> Amazon ECR richiede che gli utenti dispongano dell'autorizzazione per effettuare chiamate all'API `ecr:GetAuthorizationToken` tramite una policy IAM prima che possano autenticarsi in un registro ed eseguire l'invio o l'estrazione delle immagini da un repository Amazon ECR. Amazon ECR fornisce diverse policy IAM gestite per controllare l'accesso degli utenti a vari livelli. Per ulteriori informazioni, consulta Esempi di policy basate su Identità di Amazon Elastic Container Registry.

Esempio: consentire uno o più utenti

La policy del repository seguente consente a uno o più utenti di eseguire il push e il pull delle immagini da e verso un repository.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "AllowPushPull",
            "Effect": "Allow",
            "Principal": {
                "AWS": [
                    "arn:aws:iam::111122223333:user/push-pull-user-1",
                    "arn:aws:iam::111122223333:user/push-pull-user-2"
                ]
            },
            "Action": [
                "ecr:BatchGetImage",
                "ecr:BatchCheckLayerAvailability",
                "ecr:CompleteLayerUpload",
                "ecr:GetDownloadUrlForLayer",
                "ecr:InitiateLayerUpload",
                "ecr:PutImage",
                "ecr:UploadLayerPart"
            ],
            "Resource": "*"
        }
    ]
}
```

Esempio: abilita un altro account

La seguente policy di repository consente a un account specifico di inviare immagini.

> **Important**
>
> L'account a cui si concedono le autorizzazioni deve avere abilitata la regione in cui si sta creando la policy del repository; in caso contrario si verificherà un errore.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "AllowCrossAccountPush",
            "Effect": "Allow",
            "Principal": {
                "AWS": "arn:aws:iam::111122223333:root"
            },
            "Action": [
                "ecr:BatchCheckLayerAvailability",
                "ecr:CompleteLayerUpload",
                "ecr:InitiateLayerUpload",
                "ecr:PutImage",
                "ecr:UploadLayerPart"
            ],
            "Resource": "*"
        }
    ]
}
```

La seguente politica di archiviazione consente ad alcuni utenti di estrarre immagini (*pull-user-1*e*pull-user-2*) fornendo al contempo l'accesso completo a un'altra (). *admin-user*

> **Note**
>
> Per politiche di repository più complicate che attualmente non sono supportate in Console di gestione AWS, è possibile applicare la politica con il set-repository-policy AWS CLI comando.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "AllowPull",
```

```
            "Effect": "Allow",
            "Principal": {
                "AWS": [
                    "arn:aws:iam::111122223333:user/pull-user-1",
                    "arn:aws:iam::111122223333:user/pull-user-2"
                ]
            },
            "Action": [
                "ecr:BatchGetImage",
                "ecr:GetDownloadUrlForLayer"
            ],
            "Resource": "*"
        },
        {
            "Sid": "AllowAll",
            "Effect": "Allow",
            "Principal": {
                "AWS": "arn:aws:iam::111122223333:user/admin-user"
            },
            "Action": [
                "ecr:*"
            ],
            "Resource": "*"
        }
    ]
}
```

Esempio: vieta a tutti

La seguente policy di repository nega a tutti gli utenti in tutti gli account la possibilità di estrarre immagini.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "DenyPull",
            "Effect": "Deny",
            "Principal": "*",
```

```
            "Action": [
                "ecr:BatchGetImage",
                "ecr:GetDownloadUrlForLayer"
            ],
            "Resource": "*"
        }
    ]
}
```

Esempio: limitazione dell'accesso a indirizzi IP specifici

Nell'esempio seguente vengono negate autorizzazioni a qualunque utente per eseguire qualsiasi operazione Amazon ECR quando applicata a un repository da uno specifico intervallo di indirizzi.

La condizione in questa dichiarazione identifica l'`54.240.143.*`intervallo di indirizzi IP consentiti per la versione 4 (IPv4) del protocollo Internet.

Il `Condition` blocco utilizza `NotIpAddress` le condizioni e la chiave di `aws:SourceIp` condizione, che è una chiave di condizione a AWS livello di ampiezza. Per ulteriori informazioni su queste chiavi di condizioni, consulta Chiavi di contesto delle condizioni globali AWS. I `aws:sourceIp` IPv4 valori utilizzano la notazione CIDR standard. Per ulteriori informazioni, consulta Operatori di condizione con indirizzo IP nella Guida per l'utente di IAM.

JSON

```
{
    "Version":"2012-10-17",
    "Id": "ECRPolicyId1",
    "Statement": [
        {
            "Sid": "IPAllow",
            "Effect": "Deny",
            "Principal": "*",
            "Action": "ecr:*",
            "Resource": "*",
            "Condition": {
                "NotIpAddress": {
                    "aws:SourceIp": "54.240.143.0/24"
                }
            }
```

```
        }
    ]
}
```

Esempio: consentire un servizio AWS

La seguente politica di repository consente AWS CodeBuild l'accesso alle azioni dell'API Amazon ECR necessarie per l'integrazione con quel servizio. Quando utilizzi l'esempio seguente, è necessario utilizzare le chiavi di condizione `aws:SourceArn` e `aws:SourceAccount` per l'ambito delle risorse che possono assumere tali autorizzazioni. Per ulteriori informazioni, consulta l'esempio di Amazon ECR CodeBuild nella Guida per l'AWS CodeBuild utente.

JSON

```
{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Sid":"CodeBuildAccess",
         "Effect":"Allow",
         "Principal":{
            "Service":"codebuild.amazonaws.com"
         },
         "Action":[
            "ecr:BatchGetImage",
            "ecr:GetDownloadUrlForLayer"
         ],
         "Resource": "*",
         "Condition":{
            "ArnLike":{
               "aws:SourceArn":"arn:aws:codebuild:us-
east-1:123456789012:project/project-name"
            },
            "StringEquals":{
               "aws:SourceAccount":"123456789012"
            }
         }
      }
   ]
}
```

Impostazione di una dichiarazione sulla politica di archiviazione privata in Amazon ECR

Puoi aggiungere una dichiarazione sulla politica di accesso a un repository Console di gestione AWS seguendo i passaggi seguenti. Puoi aggiungere più dichiarazioni di policy per ciascun repository. Per esempi di policy, consulta Esempi di policy relative agli archivi privati in Amazon ECR.

> **Important**
>
> Amazon ECR richiede che gli utenti dispongano dell'autorizzazione per effettuare chiamate all'API `ecr:GetAuthorizationToken` tramite una policy IAM prima che possano autenticarsi in un registro ed eseguire l'invio o l'estrazione delle immagini da un repository Amazon ECR. Amazon ECR fornisce diverse policy IAM gestite per controllare l'accesso degli utenti a vari livelli. Per ulteriori informazioni, consulta Esempi di policy basate su Identità di Amazon Elastic Container Registry.

Per impostare una dichiarazioni di policy per i repository

1. Apri la console Amazon ECR nei https://console.aws.amazon.com/ecr/repository.
2. Sulla barra di navigazione seleziona la regione che contiene il repository sul quale impostare una dichiarazione di policy.
3. Nel riquadro di navigazione, selezionare Repositories (Repository).
4. Alla pagina Repositories (Repository), scegliere il repository sul quale impostare una dichiarazione di policy per visualizzare il contenuto del repository.
5. Dalla visualizzazione dell'elenco immagini del repository, nel pannello di navigazione, selezionare Permissions (Autorizzazioni), Edit (Modifica).

   > **Note**
   >
   > Se non si visualizza l'opzione Permissions (Autorizzazioni) nel pannello di navigazione, assicurarsi di essere nella visualizzazione dell'elenco immagini del repository.

6. Nella pagina Edit permissions (Modifica autorizzazioni), scegli Add statement (Aggiungi istruzione).
7. Nel campo Statement name (Nome istruzione) inserisci un nome per l'istruzione.

8. Per Effect (Effetto), scegliere se la dichiarazione della policy restituisce un consenso o una negazione esplicita.
9. In Principal seleziona l'ambito a cui applicare la dichiarazione di policy. Per ulteriori informazioni, consulta Elementi delle policy JSON AWS : principale nella Guida per l'utente di IAM.
   - Puoi applicare la dichiarazione a tutti gli AWS utenti autenticati selezionando la casella di controllo Everyone (*).
   - Per Service principal (Principal del servizio), specificare il nome del principal del servizio (ad esempio, `ecs.amazonaws.com`) per applicare la dichiarazione a un determinato servizio.
   - Per AWS Account IDs, specifica un numero di AWS account (ad esempio,`111122223333`) per applicare l'estratto conto a tutti gli utenti di un AWS account specifico. È possibile specificare più account utilizzando un elenco delimitato da virgole.

     Important

     L'account a cui si concedono le autorizzazioni deve avere abilitata la regione in cui si sta creando la policy del repository; in caso contrario si verificherà un errore.

   - Per le entità IAM, seleziona i ruoli o gli utenti del tuo AWS account a cui applicare l'informativa.

     Note

     Per politiche di repository più complicate che attualmente non sono supportate in Console di gestione AWS, puoi applicare la politica con il set-repository-policy AWS CLI comando.

10. In Actions (Operazioni), scegli l'ambito delle operazioni API di Amazon ECR a cui applicare la dichiarazione di policy dall'elenco delle singole operazioni API.
11. Al termine, seleziona Save (Salva) per impostare la policy.
12. Ripeti i passaggi precedenti per ogni policy di repository da aggiungere.

Taggare un repository privato in Amazon ECR

Per aiutarti a gestire i tuoi repository Amazon ECR, puoi assegnare i tuoi metadati a repository Amazon ECR nuovi o esistenti utilizzando i tag di risorsa. AWS Ad esempio, puoi definire un set

di tag per i repository Amazon ECR del tuo account che consentono di monitorare il proprietario di ciascun repository.

Nozioni di base sui tag

I tag non hanno alcun significato semantico per Amazon ECR e vengono interpretati rigorosamente come una stringa di caratteri. I tag non vengono assegnati in automatico alle risorse. Puoi modificare chiavi e valori di tag e rimuovere tag da una risorsa in qualsiasi momento. Puoi impostare il valore di un tag su una stringa vuota, ma non su null. Se aggiungi un tag con la stessa chiave di un tag esistente a una risorsa specifica, il nuovo valore sovrascrive quello precedente. Se elimini una risorsa, verranno eliminati anche tutti i tag a essa associati.

Puoi lavorare con i tag utilizzando la console Amazon ECR AWS CLI, e l'API Amazon ECR.

Utilizzando AWS Identity and Access Management (IAM), puoi controllare quali utenti del tuo AWS account sono autorizzati a creare, modificare o eliminare i tag. Per informazioni sui tag nelle politiche IAM, consultathe section called “Uso del controllo degli accessi basato su tag”.

Tagging delle risorse per la fatturazione

I tag che aggiungi ai repository Amazon ECR sono utili quando rivedi l'allocazione dei costi dopo averli abilitati nel report su costi e utilizzo. Per ulteriori informazioni, consulta Report di utilizzo di Amazon ECR.

Per visualizzare il costo delle risorse combinate, puoi organizzare le informazioni di fatturazione in base alle risorse con gli stessi valori di chiave di tag. Puoi ad esempio applicare tag a numerose risorse con un nome di applicazione specifico, quindi organizzare le informazioni di fatturazione per visualizzare il costo totale dell'applicazione in più servizi. Per ulteriori informazioni sulla configurazione di un report di allocazione dei costi mediante i tag, consulta Report di allocazione dei costi mensili nella Guida per l'utente di AWS Billing .

> **Note**
>
> Se hai appena abilitato la reportistica, i dati relativi al mese corrente saranno disponibili per la visualizzazione dopo 24 ore.

Aggiungere tag a un repository privato in Amazon ECR

È possibile aggiungere tag a un repository privato.

Per informazioni sui nomi e sulle migliori pratiche per i tag, consulta Limiti e requisiti per la denominazione dei tag e Best practice nella Tagging AWS Resources User Guide.

Aggiungere tag a un repository ()Console di gestione AWS

1. Apri la console Amazon ECR all'indirizzo https://console.aws.amazon.com/ecr/.
2. Seleziona la regione da utilizzare nella barra di navigazione.
3. Nel riquadro di navigazione, selezionare Repositories (Repository).
4. Nella pagina Repository, seleziona la casella di controllo accanto al repository che desideri taggare.
5. Dal menu Azione, seleziona Tag del repository.
6. Nella pagina Tag del repository seleziona Aggiungi tag, Aggiungi tag.
7. Nella pagina Modifica tag specifica la chiave e il valore di ogni tag, quindi scegli Salva.

Aggiungere tag a un repository (AWS CLI o API)

Puoi aggiungere o sovrascrivere uno o più tag utilizzando AWS CLI o un'API.

- AWS CLI - tag-risorsa
- Azione API - TagResource

I seguenti esempi mostrano come aggiungere tag utilizzando AWS CLI.

Esempio 1: etichettare un repository

Il comando seguente contrassegna un repository.

```
aws ecr tag-resource \
     --resource-arn arn:aws:ecr:region:account_id:repository/repository_name \
     --tags Key=stack,Value=dev
```

Esempio 2: etichettare un repository con più tag

Il comando seguente aggiunge tre tag a un repository.

```
aws ecr tag-resource \
     --resource-arn arn:aws:ecr:region:account_id:repository/repository_name \
```

```
    --tags Key=key1,Value=value1 Key=key2,Value=value2 Key=key3,Value=value3
```

Esempio 3: elenco dei tag per un repository

Il comando seguente elenca i tag associati a un repository.

```
aws ecr list-tags-for-resource \
    --resource-arn arn:aws:ecr:region:account_id:repository/repository_name
```

Esempio 4: creare un repository e aggiungere un tag

Il comando seguente crea un repository denominato `test-repo` e aggiunge un tag con chiave `team` e valore `devs`.

```
aws ecr create-repository \
    --repository-name test-repo \
    --tags Key=team,Value=devs
```

Eliminazione di tag da un repository privato in Amazon ECR

È possibile eliminare i tag da un archivio privato.

Per eliminare un tag da un archivio privato ()Console di gestione AWS

1. Apri la console Amazon ECR all'indirizzo https://console.aws.amazon.com/ecr/.
2. Seleziona la regione da utilizzare nella barra di navigazione.
3. Nella pagina Repository, seleziona la casella di controllo accanto al repository da cui desideri rimuovere un tag.
4. Dal menu Azione, seleziona Tag del repository.
5. Nella pagina Tag del repository seleziona Modifica.
6. Nella pagina Modifica tag del repository, seleziona Rimuovi per ogni tag da eliminare, quindi scegli Salva.

Per eliminare un tag da un archivio privato ()AWS CLI

È possibile eliminare uno o più tag utilizzando AWS CLI o un'API.

- AWS CLI - untag-resource

- Azione API - UntagResource

L'esempio seguente mostra come eliminare un tag da un repository utilizzando. AWS CLI

```
aws ecr untag-resource \
     --resource-arn arn:aws:ecr:region:account_id:repository/repository_name \
     --tag-keys tag_key
```

Immagini private in Amazon ECR

Amazon ECR archivia immagini Docker, immagini Open Container Initiative (OCI) e artefatti compatibili con OCI in repository privati. Puoi utilizzare la CLI di Docker, il tuo client preferito, per inviare ed estrarre immagini dai tuoi repository.

Con il supporto Amazon ECR per OCI v1.1, puoi archiviare e gestire gli artefatti di riferimento definiti dall'API OCI Referrers. Gli artefatti includono firme, Software Bill of Materials (SBoM), grafici Helm, risultati di scansione e attestazioni. Un insieme di elementi per un'immagine del contenitore viene trasferito con quel contenitore e archiviato come immagine separata che viene considerata un'immagine utilizzata per il repository.

Le Eliminazione di firme e altri elementi da un repository privato Amazon ECR pagine Firma immagini in Amazon ECR e forniscono esempi di come utilizzare gli elementi relativi alla firma. Per ulteriori informazioni sulla firma delle immagini dei contenitori, consulta Firmare le immagini dei contenitori nella Guida per gli sviluppatori.AWS Signer

Argomenti

Invio di un'immagine a un repository privato Amazon ECR

Puoi inviare immagini Docker, elenchi manifesto e immagini Open Container Initiative (OCI) e artefatti compatibili ai tuoi repository privati.

Amazon ECR offre anche un modo per replicare le immagini in altri repository. Specificando una configurazione di replica nelle impostazioni del registro privato, puoi eseguire la replica tra regioni nel tuo registro e su diversi account. Per ulteriori informazioni, consulta Impostazioni del registro privato in Amazon ECR.

> **Note**
>
> Se si inserisce un'immagine attualmente archiviata, tale immagine verrà ripristinata e rimossa automaticamente dall'archivio. Per ulteriori informazioni sull'archiviazione e il ripristino delle immagini, vedere. Archiviazione di un'immagine in Amazon ECR

Argomenti

Autorizzazioni IAM per il trasferimento di un'immagine a un repository privato Amazon ECR

Gli utenti necessitano delle autorizzazioni IAM per inviare immagini agli archivi privati di Amazon ECR. Seguendo la migliore pratica di concessione del privilegio minimo, puoi concedere l'accesso a un repository specifico. È inoltre possibile concedere l'accesso a tutti i repository.

Un utente deve autenticarsi in ogni registro Amazon ECR a cui desidera inviare le immagini richiedendo un token di autorizzazione. Amazon ECR fornisce diverse policy AWS gestite per controllare l'accesso degli utenti a vari livelli. Per ulteriori informazioni, consulta AWS politiche gestite per Amazon Elastic Container Registry.

Puoi anche creare politiche IAM personalizzate. La seguente policy IAM concede le autorizzazioni necessarie per inviare un'immagine a un repository specifico. Per limitare le autorizzazioni per un repository specifico, utilizza l'Amazon Resource Name (ARN) completo del repository.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "ecr:CompleteLayerUpload",
                "ecr:UploadLayerPart",
                "ecr:InitiateLayerUpload",
                "ecr:BatchCheckLayerAvailability",
                "ecr:PutImage",
                "ecr:BatchGetImage"
            ],
            "Resource": "arn:aws:ecr:us-
east-1:111122223333:repository/repository-name"
        },
        {
            "Effect": "Allow",
            "Action": "ecr:GetAuthorizationToken",
            "Resource": "*"
        }
    ]
}
```

La seguente policy IAM concede le autorizzazioni necessarie per inviare un'immagine a tutti i repository.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
```

```
            "Action": [
                "ecr:CompleteLayerUpload",
                "ecr:GetAuthorizationToken",
                "ecr:UploadLayerPart",
                "ecr:InitiateLayerUpload",
                "ecr:BatchCheckLayerAvailability",
                "ecr:PutImage"
            ],
             "Resource": "arn:aws:ecr:us-west-2:111122223333:repository/*"
        }
    ]
}
```

Trasferimento di un'immagine Docker a un repository privato Amazon ECR

Puoi inviare le immagini del container a un repository Amazon ECR con il comando docker push.

Amazon ECR supporta anche la creazione e l'invio di elenchi di manifest Docker utilizzati per immagini multiarchitettura. Per informazioni, consulta Trasferimento di un'immagine multiarchitettura a un repository privato Amazon ECR.

Per inviare un'immagine Docker a un repository Amazon ECR

Il repository Amazon ECR deve esistere prima di inviare l'immagine oppure è necessario che sia stato definito un modello di creazione del repository. Per ulteriori informazioni, consultare Creazione di un repository privato Amazon ECR per archiviare immagini e Modelli per controllare gli archivi creati durante un'azione di pull through cache, di creazione in modalità push o di replica.

1. Autentica il tuo client Docker nel registro Amazon ECR al quale desideri inviare l'immagine. Devi ottenere i token di autenticazione per ciascun registro utilizzato. I token hanno una validità di 12 ore. Per ulteriori informazioni, consulta Autenticazione del registro privato in Amazon ECR.

   Per autenticare Docker in un registro Amazon ECR, esegui il comando aws ecr get-login-password. Quando si passa il token di autenticazione al comando docker login, usare il valore `AWS` per il nome utente e specificare l'URI di registro Amazon ECR a cui si desidera autenticare. Se si esegue l'autenticazione a più registri, è necessario ripetere il comando per ogni registro di sistema.

> **Important**
>
> Se viene visualizzato un errore, installare o eseguire l'upgrade alla versione più recente dell' AWS CLI. Per ulteriori informazioni, consulta Installazione dell' AWS Command Line Interface nella Guida per l'utente dell'AWS Command Line Interface .

```
aws ecr get-login-password --region <region> | docker login --username AWS --password-stdin <aws_account_id>.dkr.ecr.<region>.amazonaws.com
```

2. Se il tuo repository di immagini non è ancora presente nel registro che intendi visitare e hai definito un modello per la creazione del repository, puoi inviare l'immagine utilizzando il prefisso del modello di creazione del repository e il nome del repository desiderato. ECR creerà automaticamente il repository per te utilizzando le impostazioni predefinite del modello di creazione del repository.

   Se non hai definito un modello di creazione del repository corrispondente, dovrai creare un repository. Per ulteriori informazioni, consulta Modelli per controllare gli archivi creati durante un'azione di pull through cache, di creazione in modalità push o di replica o Creazione di un repository privato Amazon ECR per archiviare immagini.

3. Identifica l'immagine locale da inviare. Esegui il comando docker images per elencare le immagini container nel tuo sistema.

   ```
   docker images
   ```

   È possibile identificare un'immagine con il *repository:tag* valore o l'ID dell'immagine nell'output del comando risultante.

4. Assegna un tag alla tua immagine con la combinazione di registro, repository, e nome tag immagine opzionale Amazon ECR da utilizzare. Il formato del registro è *aws_account_id*.dkr.ecr.*region*.amazonaws.com. Il nome del repository deve corrispondere a quello del repository che hai creato per la tua immagine. Se ometti il tag dell'immagine, presupponiamo che sia `latest`.

   L'esempio seguente contrassegna un'immagine locale con l'ID *e9ae3c220b23* come*aws_account_id*.dkr.ecr.*region*.amazonaws.com/my-repository:tag.

```
docker tag e9ae3c220b23 aws_account_id.dkr.ecr.region.amazonaws.com/my-
repository:tag
```

5. Invia l'immagine con il comando docker push:

```
docker push aws_account_id.dkr.ecr.region.amazonaws.com/my-repository:tag
```

6. (Opzionale) Assegna eventuali tag aggiuntivi alla tua immagine e invia questi tag ad Amazon ECR ripetendo Step 4 e Step 5.

Trasferimento di un'immagine multiarchitettura a un repository privato Amazon ECR

Puoi inviare immagini multiarchitettura a un repository Amazon ECR creando e inviando elenchi di manifest Docker. Un elenco manifesto è un elenco di immagini che viene creato specificando uno o più nomi di immagini. Nella maggior parte dei casi, l'elenco dei manifesti viene creato da immagini che svolgono la stessa funzione ma sono per sistemi operativi o architetture diversi. L'elenco manifesto non è obbligatorio. Per ulteriori informazioni, consulta Docker manifest.

Un elenco manifesto può essere estratto o è possibile farvi riferimento in una definizione di attività Amazon ECS o specifiche del pod Amazon EKS come altre immagini Amazon ECR.

Prerequisiti

- Nella CLI Docker, attiva le funzionalità sperimentali. Per informazioni sulle funzionalità sperimentali, consulta Funzionalità sperimentali nella documentazione Docker.
- Il repository Amazon ECR deve esistere prima di eseguire l'invio dell'immagine. Per ulteriori informazioni, consulta the section called “Creazione di un repository per archiviare immagini”.
- Le immagini devono essere inviate al tuo repository prima di creare il manifesto Docker. Per informazioni su come inviare un'immagine, consulta Trasferimento di un'immagine Docker a un repository privato Amazon ECR.

Per eseguire l'invio di un'immagine Docker multi-architettura in un repository Amazon ECR

1. Autentica il tuo client Docker nel registro Amazon ECR al quale desideri inviare l'immagine. Devi ottenere i token di autenticazione per ciascun registro utilizzato. I token hanno una validità di 12 ore. Per ulteriori informazioni, consulta Autenticazione del registro privato in Amazon ECR.

Per autenticare Docker in un registro Amazon ECR, esegui il comando aws ecr get-login-password. Quando si passa il token di autenticazione al comando docker login, usare il valore `AWS` per il nome utente e specificare l'URI di registro Amazon ECR a cui si desidera autenticare. Se si esegue l'autenticazione a più registri, è necessario ripetere il comando per ogni registro di sistema.

> **Important**
>
> Se viene visualizzato un errore, installare o eseguire l'upgrade alla versione più recente dell' AWS CLI. Per ulteriori informazioni, consulta Installazione dell' AWS Command Line Interface nella Guida per l'utente dell'AWS Command Line Interface .

```
aws ecr get-login-password --region <region> | docker login --username AWS --
password-stdin <aws_account_id>.dkr.ecr.<region>.amazonaws.com
```

2. Elenca le immagini nel tuo repository, confermando i tag immagine.

```
aws ecr describe-images --repository-name my-repository
```

3. Crea l'elenco manifesto Docker. Il comando `manifest create` verifica che le immagini di riferimento siano già presenti nel repository e crea il manifest localmente.

```
docker manifest create aws_account_id.dkr.ecr.region.amazonaws.com/my-
repository aws_account_id.dkr.ecr.region.amazonaws.com/my-
repository:image_one_tag aws_account_id.dkr.ecr.region.amazonaws.com/my-
repository:image_two
```

4. (Facoltativo) Ispezionare l'elenco dei manifest Docker. Ciò consente di confermare le dimensioni e il digest per ogni manifest immagine a cui si fa riferimento nell'elenco dei manifest.

```
docker manifest inspect aws_account_id.dkr.ecr.region.amazonaws.com/my-repository
```

5. Eseguire l'invio dell'elenco manifesto Docker nel repository Amazon ECR.

```
docker manifest push aws_account_id.dkr.ecr.region.amazonaws.com/my-repository
```

Trasferimento di un grafico Helm a un repository privato Amazon ECR

Puoi inviare gli artefatti dell'Open Container Initiative (OCI) a un repository Amazon ECR. Per vedere un esempio di questa funzionalità, utilizza i seguenti passaggi per inviare un grafico Helm ad Amazon ECR.

Per informazioni sull'utilizzo dei grafici Helm ospitati da Amazon ECR con Amazon EKS, consulta. Installazione di un grafico Helm su un cluster Amazon EKS

Per inviare un grafico Helm a un repository Amazon ECR

1. Installa la versione più recente del client Helm. Questi passaggi sono stati scritti utilizzando la versione `3.18.6` di Helm. Per la compatibilità con le versioni di Kubernetes supportate da Amazon EKS, usa Helm versione 3.9 o successiva. Per ulteriori informazioni, consulta l'argomento relativo all'installazione di Helm.
2. Utilizza i seguenti passaggi per creare un grafico Helm di prova. Per ulteriori informazioni, consulta Documenti Helm - Nozioni di base.

   a. Creare un grafico Helm denominato `helm-test-chart` e cancellare il contenuto della directory `templates`.

   ```
   helm create helm-test-chart
   rm -rf ./helm-test-chart/templates/*
   ```

   b. Crea un file nella cartella. `ConfigMap` `templates`

   ```
   cd helm-test-chart/templates
   cat <<EOF > configmap.yaml
   apiVersion: v1
   kind: ConfigMap
   metadata:
     name: helm-test-chart-configmap
   data:
     myvalue: "Hello World"
   EOF
   ```

3. Creazione pacchetto del grafico. L'output conterrà il nome del file del grafico in pacchetto utilizzato quando si invia il grafico Helm.

   ```
   cd ../..
   ```

```
helm package helm-test-chart
```

Output

```
Successfully packaged chart and saved it to: /Users/username/helm-test-
chart-0.1.0.tgz
```

4. Creare un repository per archiviare il grafico Helm. Il nome del repository deve corrispondere al nome utilizzato nel grafico Helm al passaggio 2. Per ulteriori informazioni, consulta Creazione di un repository privato Amazon ECR per archiviare immagini.

```
aws ecr create-repository \
     --repository-name helm-test-chart \
     --region us-west-2
```

5. Autentica il tuo client Helm nel registro Amazon ECR al quale desideri inviare il grafico Helm. Devi ottenere i token di autenticazione per ciascun registro utilizzato. I token hanno una validità di 12 ore. Per ulteriori informazioni, consulta Autenticazione del registro privato in Amazon ECR.

```
aws ecr get-login-password \
     --region us-west-2 | helm registry login \
     --username AWS \
     --password-stdin aws_account_id.dkr.ecr.region.amazonaws.com
```

6. Inviare il grafico Helm utilizzando il comando helm push. L'output deve includere l'URI del repository Amazon ECR e il digest SHA.

```
helm push helm-test-chart-0.1.0.tgz
 oci://aws_account_id.dkr.ecr.region.amazonaws.com/
```

7. Descrivi il tuo grafico Helm.

```
aws ecr describe-images \
     --repository-name helm-test-chart \
     --region us-west-2
```

Nell'output, verificare che il parametri `artifactMediaType` indichi il tipo di artefatto corretto.

```
{
    "imageDetails": [
```

```
        {
            "registryId": "aws_account_id",
            "repositoryName": "helm-test-chart",
            "imageDigest":
 "sha256:dd8aebdda7df991a0ffe0b3d6c0cf315fd582cd26f9755a347a52adEXAMPLE",
            "imageTags": [
                "0.1.0"
            ],
            "imageSizeInBytes": 1620,
            "imagePushedAt": "2021-09-23T11:39:30-05:00",
            "imageManifestMediaType": "application/vnd.oci.image.manifest.v1+json",
            "artifactMediaType": "application/vnd.cncf.helm.config.v1+json"
        }
    ]
}
```

8. (Facoltativo) Per ulteriori passaggi, installa Helm `ConfigMap` e inizia a usare Amazon EKS. Per ulteriori informazioni, consulta Installazione di un grafico Helm su un cluster Amazon EKS.

Eliminazione di firme e altri elementi da un repository privato Amazon ECR

Puoi utilizzare il client ORAS per elencare ed eliminare firme e altri artefatti di tipo di riferimento da un repository privato Amazon ECR. L'eliminazione di firme e altri elementi di riferimento è simile a come viene eliminata un'immagine (vedi). Eliminazione di un'immagine in Amazon ECR Ecco come elencare gli artefatti ed eliminare le firme:

Per gestire gli artefatti dell'immagine utilizzando l'ORAS CLI

1. Installa e configura il client ORAS.

   Per informazioni sull'installazione e la configurazione del client ORAS, vedere Installazione nella documentazione ORAS.

2. Per elencare gli artefatti disponibili per un'immagine Amazon ECR, usa`oras discover`, seguito dal nome dell'immagine:

   ```
   oras discover 111222333444.dkr.ecr.us-east-1.amazonaws.com/oci:helloworld
   ```

   L'output dovrebbe avere questo aspetto:

```
111222333444.dkr.ecr.us-east-1.amazonaws.com/
oci@sha256:88c0c54329bfdc1d94d6f58cd3fcb1226d46f58670f44a8c689cb3c9b37b6925
### application/vnd.cncf.notary.signature
    ### sha256:387c10c1598ee18aae81dcfc86d0d06d116e46461d1c3cda8927e69c48108c42
    ### sha256:6527bcec87adf1d55460666183b9d0968b3cd4e4bc34602d485206a219851171
```

3. Per eliminare una firma utilizzando la CLI ORAS, dato l'esempio precedente, esegui il comando seguente:

```
oras manifest delete 111222333444.dkr.ecr.us-east-1.amazonaws.com/
oci@sha256:387c10c1598ee18aae81dcfc86d0d06d116e46461d1c3cda8927e69c48108c42
```

   L'output dovrebbe avere questo aspetto:

```
Are you sure you want to delete the manifest "111222333444.dkr.ecr.us-
east-1.amazonaws.com/
oci@sha256:387c10c1598ee18aae81dcfc86d0d06d116e46461d1c3cda8927e69c48108c42" and
 all tags associated with it? [y/N] y
```

4. Premi y. L'artefatto deve essere eliminato.

Per risolvere i problemi di eliminazione degli artefatti

Se l'eliminazione di una firma, come quella appena mostrata, non riesce, viene visualizzato un output simile al seguente.

```
Error response from registry: failed to delete 111222333444.dkr.ecr.us-
east-1.amazonaws.com/
oci@sha256:387c10c1598ee18aae81dcfc86d0d06d116e46461d1c3cda8927e69c48108c42:
unsupported: Requested image referenced by manifest list:
 [sha256:005e2c97a6373e483799fa4ff29ac64a42dd10f08efcc166d6775f9b74943b5b]
```

Questo errore può verificarsi quando si elimina un'immagine inviata prima del lancio di OCI 1.1. Come indicato nell'errore, è necessario eliminare il manifesto che fa riferimento all'immagine prima di poter eliminare l'immagine come segue:

1. Per eliminare il manifesto associato alla firma che desideri eliminare, digita:

```
 oras manifest delete 111222333444.dkr.ecr.us-east-1.amazonaws.com/
oci@sha256:005e2c97a6373e483799fa4ff29ac64a42dd10f08efcc166d6775f9b74943b5b
```

   L'output dovrebbe avere questo aspetto:

```
Are you sure you want to delete the manifest
 "sha256:005e2c97a6373e483799fa4ff29ac64a42dd10f08efcc166d6775f9b74943b5b" and all
 tags associated with it? [y/N] y
```

2. Premi y. Il manifesto deve essere eliminato.
3. Una volta eliminato il manifesto, puoi eliminare la firma:

```
 oras manifest delete 111222333444.dkr.ecr.us-east-1.amazonaws.com/
oci@sha256:387c10c1598ee18aae81dcfc86d0d06d116e46461d1c3cda8927e69c48108c42
```

   L'output dovrebbe essere simile a questo. Premi y.

```
Are you sure you want to delete the manifest
 "sha256:387c10c1598ee18aae81dcfc86d0d06d116e46461d1c3cda8927e69c48108c42" and all
 tags associated with it? [y/N] y
Deleted [registry] 111222333444.dkr.ecr.us-east-1.amazonaws.com/
oci@sha256:387c10c1598ee18aae81dcfc86d0d06d116e46461d1c3cda8927e69c48108c42
```

4. Per verificare che la firma sia stata eliminata, digita:

```
oras discover 111222333444.dkr.ecr.us-east-1.amazonaws.com/oci:helloworld
```

   L'output dovrebbe avere questo aspetto:

```
111222333444.dkr.ecr.us-east-1.amazonaws.com/
oci@sha256:88c0c54329bfdc1d94d6f58cd3fcb1226d46f58670f44a8c689cb3c9b37b6925
```

```
### application/vnd.cncf.notary.signature
    ### sha256:6527bcec87adf1d55460666183b9d0968b3cd4e4bc34602d485206a219851171
```

Visualizzazione dei dettagli delle immagini in Amazon ECR

Dopo aver inviato un'immagine al tuo repository, puoi visualizzare le relative informazioni. I dettagli inclusi sono i seguenti:

- URI immagine
- Tag di immagine
- Tipo di supporto di un artefatto
- Tipo di manifesto delle immagini
- Stato della scansione
- Le dimensioni dell'immagine in MB
- Data del push dell'immagine al repository
- Lo stato di replica

Per visualizzare i dettagli dell'immagine (Console di gestione AWS)

1. Apri la console Amazon ECR nei https://console.aws.amazon.com/ecr/repository.
2. Dalla barra di navigazione, scegliere la regione in cui si trova il repository che contiene la tua immagine.
3. Nel riquadro di navigazione, selezionare Repositories (Repository).
4. Nella pagina Repositories (Repository), selezionare il repository da visualizzare.
5. Nella *repository_name* pagina Repositories:, scegli l'immagine di cui visualizzare i dettagli.

Estrazione di un'immagine nel tuo ambiente locale da un repository privato Amazon ECR

Se desideri eseguire un'immagine Docker disponibile in Amazon ECR, puoi estrarla nel tuo ambiente locale con il comando docker pull. È possibile eseguire questa operazione dal registro predefinito o da un registro associato a un altro AWS account.

Per utilizzare un'immagine Amazon ECR in una definizione di attività Amazon ECS, consulta Utilizzo delle immagini Amazon ECR con Amazon ECS.

> **Important**
>
> Non è possibile estrarre un'immagine archiviata. Le immagini archiviate devono essere ripristinate prima di poter essere recuperate. Per ulteriori informazioni sull'archiviazione e il ripristino delle immagini, vedere. Archiviazione di un'immagine in Amazon ECR

> **Important**
>
> Amazon ECR richiede che gli utenti dispongano dell'autorizzazione per effettuare chiamate all'API `ecr:GetAuthorizationToken` tramite una policy IAM prima che possano autenticarsi in un registro ed eseguire l'invio o l'estrazione delle immagini da un repository Amazon ECR. Amazon ECR fornisce diverse policy AWS gestite per controllare l'accesso degli utenti a vari livelli. Per informazioni sulle politiche AWS gestite per Amazon ECR, consultaAWS politiche gestite per Amazon Elastic Container Registry.

Per estrarre un'immagine Docker da un repository Amazon ECR

1. Autentica il tuo client Docker nel registro Amazon ECR dal quale desideri estrarre la tua immagine. Devi ottenere i token di autenticazione per ciascun registro utilizzato. I token hanno una validità di 12 ore. Per ulteriori informazioni, consulta Autenticazione del registro privato in Amazon ECR.
2. (Opzionale) Identifica l'immagine da estrarre.
   - Puoi elencare i repository in un registro con il comando aws ecr describe-repositories:

     ```
     aws ecr describe-repositories
     ```

     Il registro di esempio riportato sopra ha un repository chiamato. `amazonlinux`
   - Puoi descrivere le immagini in un repository con il comando aws ecr describe-images:

     ```
     aws ecr describe-images --repository-name amazonlinux
     ```

Il precedente repository di esempio ha un'immagine taggata come `latest` e `2016.09`, con il digest dell'immagine `sha256:f1d4ae3f7261a72e98c6ebefe9985cf10a0ea5bd762585a43e0700ed99863807`.

3. Estrai l'immagine con il comando docker pull. Il formato del nome dell'immagine deve essere *`registry`*`/`*`repository`*` [:`*`tag`*`]` per effettuare l'estrazione tramite tag o *`registry`*`/`*`repository`*`[@`*`digest`*`]` per effettuare l'estrazione tramite digest.

```
docker pull aws_account_id.dkr.ecr.us-west-2.amazonaws.com/amazonlinux:latest
```

> **Important**
>
> Se ricevi un errore *`repository-url`*` not found: does not exist or no pull access`, potrebbe essere necessario autenticare il tuo client Docker con Amazon ECR. Per ulteriori informazioni, consulta Autenticazione del registro privato in Amazon ECR.

Estrazione dell'immagine del contenitore Amazon Linux

L'immagine di container Linux è costruita con gli stessi componenti software inclusi nell'Amazon Linux AMI. L'immagine del contenitore Amazon Linux è disponibile per l'uso in qualsiasi ambiente come immagine di base per i carichi di lavoro Docker. Se utilizzi l'AMI Amazon Linux per applicazioni in Amazon EC2, puoi containerizzare le tue applicazioni con l'immagine del contenitore Amazon Linux.

Puoi utilizzare l'immagine del contenitore Amazon Linux nel tuo ambiente di sviluppo locale e quindi inviare l'applicazione all' AWS utilizzo di Amazon ECS. Per ulteriori informazioni, consulta Utilizzo delle immagini Amazon ECR con Amazon ECS.

L'immagine del container Amazon Linux è disponibile in Amazon ECR Public e in Docker Hub. Per il supporto per l'immagine del contenitore Amazon Linux, vai ai forum degli AWS sviluppatori.

Per estrarre l'immagine del container Amazon Linux da Amazon ECR Public

1. Autentica il cliente Docker nel registro Amazon Linux Public. I token di autenticazione sono validi 12 ore. Per ulteriori informazioni, consulta Autenticazione del registro privato in Amazon ECR.

> **Note**
>
> I ecr-public comandi sono disponibili in AWS CLI a partire dalla versione `1.18.1.187`, tuttavia consigliamo di utilizzare comunque la versione più recente di AWS CLI. Per ulteriori informazioni, consulta Installazione dell' AWS Command Line Interface nella Guida per l'utente dell'AWS Command Line Interface .

```
aws ecr-public get-login-password --region us-east-1 | docker login --username AWS
 --password-stdin public.ecr.aws
```

L'output è il seguente:

```
Login succeeded
```

2. Estrai l'immagine del container Amazon Linux con il comando docker pull. Per visualizzare l'immagine del container Amazon Linux nella galleria di Amazon ECR Public, consulta la galleria di Amazon ECR Public - amazonlinux.

```
docker pull public.ecr.aws/amazonlinux/amazonlinux:latest
```

3. (Opzionale) Esegui il container a livello locale.

```
docker run -it public.ecr.aws/amazonlinux/amazonlinux /bin/bash
```

Per estrarre l'immagine del container Amazon Linux da Docker Hub

1. Estrai l'immagine del container Amazon Linux con il comando docker pull.

```
docker pull amazonlinux
```

2. (Opzionale) Esegui il container a livello locale.

```
docker run -it amazonlinux:latest /bin/bash
```

Eliminazione di un'immagine in Amazon ECR

Quando hai finito di utilizzare un'immagine, puoi eliminarla dal tuo repository. Quando hai finito di utilizzare un repository, puoi eliminare l'intero repository e tutte le immagini al suo interno. Per ulteriori informazioni, consulta Eliminazione di un repository privato in Amazon ECR.

In alternativa all'eliminazione manuale delle immagini, puoi creare policy relative al ciclo di vita del repository che forniscono un maggiore controllo sulla gestione del ciclo di vita delle immagini nei tuoi repository. Le policy relative al ciclo di vita automatizzano questo processo per te. Per ulteriori informazioni, consulta Automatizza la pulizia delle immagini utilizzando le politiche del ciclo di vita in Amazon ECR.

> **Note**
>
> Se il tuo repository contiene una combinazione di immagini, alcune delle quali sono state inviate prima che Amazon ECR supportasse OCI v1.1, alcune firme avranno indici di immagini o elenchi di manifesti che puntano ad esse. Di conseguenza, quando si elimina un'immagine precedente a OCI v1.1, potrebbe essere necessario eliminare manualmente l'elenco dei manifesti che fa riferimento all'immagine per eliminare l'artefatto.

Per eliminare un'immagine (Console di gestione AWS)

1. Apri la console Amazon ECR nei https://console.aws.amazon.com/ecr/repository.
2. Sulla barra di navigazione seleziona la regione in cui si trova l'immagine da eliminare.
3. Nel riquadro di navigazione, selezionare Repositories (Repository).
4. Nella pagina Repositories (Repository) seleziona il repository che contiene l'immagine da eliminare.
5. Nella ***repository_name*** pagina Repository:, seleziona la casella a sinistra dell'immagine da eliminare e scegli Elimina.
6. Nella finestra di dialogo Delete image(s) (Elimina immagini) verifica che le immagini selezionate debbano essere realmente eliminate, quindi scegli Delete (Elimina).

Per eliminare un'immagine (AWS CLI)

1. Elenca le immagini nel tuo repository. Le immagini con tag assegnati avranno sia un digest di immagine che un elenco di tag associati. Le immagini senza tag assegnati avranno solo un digest delle immagini.

```
aws ecr list-images \
     --repository-name my-repo
```

2. (Opzionale) Elimina i tag indesiderati per l'immagine specificando il tag associato all'immagine che desideri eliminare. Quando elimini l'ultimo tag da un'immagine, l'immagine viene anche eliminata.

```
aws ecr batch-delete-image \
     --repository-name my-repo \
     --image-ids imageTag=tag1 imageTag=tag2
```

3. Eliminare un'immagine con o senza tag assegnati specificando il digest dell'immagine. Quando elimini un'immagine facendo riferimento al suo digest, l'immagine e tutti i relativi tag vengono eliminati.

```
aws ecr batch-delete-image \
     --repository-name my-repo \
     --image-ids imageDigest=sha256:4f70ef7a4d29e8c0c302b13e25962d8f7a0bd304EXAMPLE
```

   Per eliminare più immagini, puoi specificare più tag immagine o digest immagine nella richiesta.

```
aws ecr batch-delete-image \
     --repository-name my-repo \
     --image-ids imageDigest=sha256:4f70ef7a4d29e8c0c302b13e25962d8f7a0bd304EXAMPLE
 imageDigest=sha256:f5t0e245ssffc302b13e25962d8f7a0bd304EXAMPLE
```

Archiviazione di un'immagine in Amazon ECR

Cos'è la classe di archiviazione ECR?

La classe di storage di archiviazione Amazon ECR è una nuova classe di storage che offre storage a lungo termine a basso costo per le immagini dei container. Amazon ECR offre due classi di storage:

- Classe di storage standard ECR: la classe di storage predefinita per le immagini attive a cui si accede regolarmente.
- Classe di archiviazione ECR: una classe di archiviazione a basso costo per immagini a cui si accede raramente ma che devono essere conservate per motivi di conformità o per riferimento a lungo termine. La classe di archiviazione offre risparmi sui costi per grandi quantità di immagini rispetto alla classe di archiviazione Standard per la conservazione delle immagini a lungo termine. Per informazioni dettagliate sui prezzi, consulta i prezzi di Amazon ECR.

Per archiviare le immagini, hai due opzioni. Innanzitutto, puoi configurare le regole del ciclo di vita per archiviare automaticamente le immagini in base a:

- Tempo trascorso da quando l'immagine è stata inviata
- Tempo trascorso dall'ultima volta che l'immagine è stata estratta
- Numero di immagini nel repository

Puoi anche configurare le impostazioni per eliminare definitivamente le immagini dopo che sono state archiviate per un periodo specificato. Per ulteriori informazioni, consulta Automatizza la pulizia delle immagini utilizzando le politiche del ciclo di vita in Amazon ECR.

Puoi anche archiviare le immagini utilizzando la console Amazon ECR o AWS CLI. Per ulteriori informazioni, consulta Archiviazione di un'immagine.

Quando devi utilizzare nuovamente un'immagine archiviata, puoi ripristinarla nella classe di storage ECR Standard. Puoi aspettarti che ECR ripristini l'immagine entro 20 minuti. Le immagini ripristinate si comportano come immagini appena inviate e sono immediatamente disponibili per l'uso una volta completato il ripristino. Le immagini ripristinate sono soggette alle politiche di scansione, replica e ciclo di vita del repository. Per ulteriori informazioni, consulta Ripristino di un'immagine.

Archiviazione di un'immagine

Puoi archiviare le immagini manualmente utilizzando la console Amazon ECR o automaticamente utilizzando AWS CLI le policy del ciclo di vita. Quando un'immagine viene archiviata:

- L'immagine viene spostata nella classe di archiviazione.
- Le immagini archiviate non possono essere recuperate. Le richieste di recupero dell'immagine archiviata falliranno e restituiranno un errore 404.

- Sebbene l'immagine non possa essere estratta, può comunque essere descritta utilizzando il describe-images comando o elencata utilizzando il list-images comando. Lo stato dell'immagine verrà visualizzato come`ARCHIVED`.
- Le immagini archiviate hanno una durata minima di archiviazione di 90 giorni. Non è possibile configurare politiche del ciclo di vita che eliminino le immagini archiviate da meno di 90 giorni. Se devi eliminare immagini archiviate per meno di 90 giorni, devi utilizzare l'batch-delete-imageAPI, ma ti verrà addebitata la durata minima di archiviazione di 90 giorni.
- L'immagine viene visualizzata in una scheda Immagini archiviate nella vista del repository (questa scheda viene visualizzata solo se almeno un'immagine è archiviata nell'archivio).
- L'immagine può essere ripristinata come immagine attiva selezionandola manualmente da ripristinare o inserendo nuovamente l'immagine nel repository.
- L'immagine verrà eliminata se il repository dispone di politiche relative al ciclo di vita che eliminano l'immagine con criteri come il tempo di archiviazione.

Console di gestione AWS

Per archiviare un'immagine

1. Apri la console Amazon ECR nei https://console.aws.amazon.com/ecr/repository.
2. Dalla barra di navigazione, scegli la regione che contiene il repository con l'immagine che desideri archiviare.
3. Nel riquadro di navigazione, selezionare Repositories (Repository).
4. Nella pagina Repository, scegli il repository contenente l'immagine che desideri archiviare.
5. Seleziona l'immagine che desideri archiviare. Vedrai i dettagli dell'immagine.
6. Per archiviare l'immagine, seleziona il pulsante Archivia e seleziona Conferma quando richiesto.
7. Se questa è la prima immagine archiviata nel repository, viene visualizzata una nuova scheda Immagini archiviate con l'immagine appena archiviata. Se ci sono altre immagini archiviate, questa immagine verrà aggiunta a quella scheda.

AWS CLI

Per archiviare un'immagine

- Utilizzate il update-image-storage-class comando per archiviare un'immagine aggiornando la relativa classe di archiviazione a`ARCHIVE`:

```
aws ecr update-image-storage-class \
    --repository-name my-repository \
    --image-id
 imageDigest=sha256:4f70ef7a4d29e8c0c302b13e25962d8f7a0bd304EXAMPLE \
    --target-storage-class ARCHIVE
```

Per archiviare un'immagine utilizzando le politiche del ciclo di vita

- È possibile configurare le regole di archiviazione per i repository utilizzando le politiche del ciclo di vita per archiviare automaticamente le immagini. Le policy relative al ciclo di vita consentono di archiviare automaticamente le immagini in base a criteri quali:
  - Tempo trascorso da quando l'immagine è stata inviata
  - Tempo trascorso dall'ultima volta che l'immagine è stata estratta
  - Numero massimo di immagini da mantenere attive

  Puoi anche configurare le politiche del ciclo di vita per eliminare definitivamente le immagini dopo che sono state archiviate per un periodo specificato. Per ulteriori informazioni ed esempi di politiche del ciclo di vita con azioni di archiviazione, consulta. Automatizza la pulizia delle immagini utilizzando le politiche del ciclo di vita in Amazon ECR

  > **Note**
  >
  > Le immagini archiviate hanno una durata minima di archiviazione di 90 giorni. Non è possibile configurare politiche del ciclo di vita che eliminino le immagini archiviate da meno di 90 giorni. Se devi eliminare immagini archiviate per meno di 90 giorni, devi utilizzare l'batch-delete-imageAPI, ma ti verrà addebitata la durata minima di archiviazione di 90 giorni.

Quando descrivi le immagini utilizzando il describe-images comando, le immagini archiviate hanno un di. `image-status` `ARCHIVED` È possibile filtrare le immagini `image-status` per visualizzare solo le immagini archiviate o solo le immagini attive.

Ripristino di un'immagine

Quando ripristini un'immagine archiviata, questa viene spostata dalla classe di archiviazione ECR Archive alla classe di archiviazione ECR Standard. Le immagini ripristinate vengono addebitate in base alle tariffe di archiviazione standard. Il processo di ripristino esegue azioni simili a quelle che si verificano quando viene creata una nuova immagine:

- L'immagine diventa disponibile per l'estrazione al termine del ripristino. Il ripristino richiede in genere fino a 20 minuti, anche se può essere completato più rapidamente.
- Se la scansione su push è abilitata per il repository, l'immagine ripristinata verrà scansionata. Tieni presente che i risultati della scansione precedente all'archiviazione dell'immagine non saranno disponibili.
- Se la replica è configurata per il repository, l'immagine ripristinata verrà replicata se la replica era abilitata al momento del ripristino.
- L'immagine ripristinata viene visualizzata nell'elenco delle immagini attive.

Il ripristino di un'immagine richiede in genere fino a 20 minuti, anche se può essere completato più rapidamente. Durante il processo di ripristino, l'immagine rimane nello stato di archiviazione e non può essere recuperata fino al completamento del ripristino.

Console di gestione AWS

Per ripristinare un'immagine archiviata

1. Apri la console Amazon ECR nei https://console.aws.amazon.com/ecr/repository.
2. Dalla barra di navigazione, scegli la regione che contiene l'archivio con l'immagine archiviata che desideri ripristinare.
3. Nel riquadro di navigazione, selezionare Repositories (Repository).
4. Nella pagina Repository, scegli il repository contenente l'immagine archiviata.
5. Scegli la scheda Immagini archiviate.
6. Seleziona l'immagine archiviata che desideri ripristinare.
7. Scegli Ripristina e conferma l'azione di ripristino.

8. Attendi il completamento del ripristino. L'immagine apparirà nell'elenco delle immagini attive una volta completato il ripristino.

AWS CLI

Per ripristinare un'immagine archiviata

- Utilizzate il update-image-storage-class comando per ripristinare un'immagine archiviata aggiornando la relativa classe di archiviazione a: `STANDARD`

```
aws ecr update-image-storage-class \
    --repository-name my-repository \
    --image-id
 imageDigest=sha256:4f70ef7a4d29e8c0c302b13e25962d8f7a0bd304EXAMPLE \
    --target-storage-class STANDARD
```

Quando descrivi le immagini utilizzando il describe-images comando, le immagini che vengono ripristinate hanno un `image-status` di`ACTIVATING`. È possibile filtrare le immagini in base `image-status` `ACTIVATING` al valore utilizzato per visualizzare le immagini attualmente in fase di ripristino.

Un metodo alternativo per ripristinare un'immagine archiviata consiste nel reinserire l'immagine nell'archivio. Quando invii un'immagine che è attualmente archiviata, tale immagine verrà immediatamente ripristinata e rimossa dall'archivio.

Ritaggare un'immagine in Amazon ECR

Con le immagini Docker Image Manifest V2 Schema 2, puoi usare l'opzione `--image-tag` del comando put-image per inserire nuovamente un tag in un'immagine esistente. Puoi inserire nuovamente il tag senza estrarre o inviare l'immagine con Docker. Per le immagini più grandi, questo processo consente di risparmiare una notevole quantità di larghezza di banda di rete e il tempo richiesto per reinserire un tag a un'immagine.

Per inserire nuovamente un tag in un'immagine (AWS CLI)

Per rietichettare un'immagine con AWS CLI

1. Utilizza il comando batch-get-image per ottenere il manifesto dell'immagine al fine di inserire nuovamente il tag nell'immagine e scriverla su un file. In questo esempio, il manifesto di

un'immagine con il tag*latest*, nel repository*amazonlinux*, viene scritto in una variabile di ambiente denominata. *MANIFEST*

```
MANIFEST=$(aws ecr batch-get-image --repository-name amazonlinux --image-ids
 imageTag=latest --output text --query 'images[].imageManifest')
```

2. Utilizza l'opzione `--image-tag` del comando put-image per inserire il manifesto dell'immagine in Amazon ECR con un nuovo tag. In questo esempio, l'immagine è etichettata come*2017.03*.

> **Note**
>
> Se l'`--image-tag`opzione non è disponibile nella tua versione di AWS CLI, esegui l'aggiornamento alla versione più recente. Per ulteriori informazioni, consulta Installazione dell' AWS Command Line Interface nella Guida per l'utente dell'AWS Command Line Interface .

```
aws ecr put-image --repository-name amazonlinux --image-tag 2017.03 --image-
manifest "$MANIFEST"
```

3. Verifica che il nuovo tag dell'immagine sia collegato all'immagine. Nell'output seguente, l'immagine presenta i tag `latest` e `2017.03`.

```
aws ecr describe-images --repository-name amazonlinux
```

L'output è il seguente:

```
{
    "imageDetails": [
        {
            "imageSizeInBytes": 98755613,
            "imageDigest":
 "sha256:8d00af8f076eb15a33019c2a3e7f1f655375681c4e5be157a26EXAMPLE",
            "imageTags": [
                "latest",
                "2017.03"
            ],
            "registryId": "aws_account_id",
            "repositoryName": "amazonlinux",
            "imagePushedAt": 1499287667.0
```

```
        }
    ]
}
```

Per inserire nuovamente un tag in un'immagine (AWS Tools for Windows PowerShell)

Per rietichettare un'immagine con AWS Tools for Windows PowerShell

1. Utilizzate il Get-ECRImageBatch cmdlet per ottenere la descrizione dell'immagine da rietichettare e scriverla in una variabile di ambiente. In questo esempio, un'immagine con il tag, *latest*, nel repository*amazonlinux*, viene scritta nella variabile di ambiente,. *$Image*

   > **Note**
   >
   > Se non è Get-ECRImageBatch cmdlet disponibile sul sistema, consulta Configurazione di AWS Tools for Windows PowerShell nella Guida per l'AWS Strumenti per PowerShell utente.

   ```
   $Image = Get-ECRImageBatch -ImageId @{ imageTag="latest" } -
   RepositoryName amazonlinux
   ```

2. Scrivi il manifesto dell'immagine nella variabile di *$Manifest* ambiente.

   ```
   $Manifest = $Image.Images[0].ImageManifest
   ```

3. Utilizza l'`-ImageTag`opzione di Write-ECRImage cmdlet inserire il manifesto dell'immagine in Amazon ECR con un nuovo tag. In questo esempio, l'immagine è etichettata come*2017.09*.

   ```
   Write-ECRImage -RepositoryName amazonlinux -ImageManifest $Manifest -
   ImageTag 2017.09
   ```

4. Verifica che il nuovo tag dell'immagine sia collegato all'immagine. Nell'output seguente, l'immagine presenta i tag `latest` e `2017.09`.

   ```
   Get-ECRImage -RepositoryName amazonlinux
   ```

   L'output è il seguente:

```
ImageDigest                                                             ImageTag
-----------                                                             --------
sha256:359b948ea8866817e94765822787cd482279eed0c17bc674a7707f4256d5d497 latest
sha256:359b948ea8866817e94765822787cd482279eed0c17bc674a7707f4256d5d497 2017.09
```

Impedire la sovrascrittura dei tag di immagine in Amazon ECR

È possibile impedire che i tag delle immagini vengano sovrascritti attivando l'immutabilità dei tag in un repository. Dopo aver attivato l'immutabilità dei tag, l' `ImageTagAlreadyExistsException`errore viene restituito se si inserisce un'immagine con un tag già presente nel repository. L'immutabilità dei tag influisce su tutti i tag. Non puoi rendere immutabili alcuni tag mentre altri no.

È possibile utilizzare gli AWS CLI strumenti Console di gestione AWS e per impostare la mutabilità dei tag di immagine per un nuovo repository o per un repository esistente. Per creare un repository utilizzando i passaggi della console, consulta. Creazione di un repository privato Amazon ECR per archiviare immagini

Impostazione della mutabilità del tag dell'immagine ()Console di gestione AWS

Per impostare la mutabilità dei tag di immagine

1. Apri la console Amazon ECR nei https://console.aws.amazon.com/ecr/repository.
2. Dalla barra di navigazione, scegliere la regione in cui si trova il repository da modificare.
3. Nel riquadro di navigazione, scegli Repositories in Registro privato.

   Se non vedi Repository, scegli Registro privato per espandere il menu, quindi scegli Repository.
4. Nella pagina Archivi privati, scegli il pulsante di opzione prima del nome del repository per il quale desideri impostare le impostazioni di mutabilità del tag dell'immagine.
5. Scegli Azioni, quindi scegli Archivio in Modifica.
6. Per l'immutabilità dei tag Image, scegli una delle seguenti impostazioni di modifica dei tag per il repository.

   - Mutabile: scegliete questa opzione se desiderate che i tag delle immagini vengano sovrascritti. Consigliato per i repository che utilizzano azioni pull through cache per garantire che Amazon ECR possa aggiornare le immagini memorizzate nella cache. Inoltre, per disabilitare gli

aggiornamenti dei tag per alcuni tag mutabili, inserisci i nomi dei tag o usa i caratteri jolly (*) per abbinare più tag simili nella casella di testo Esclusione tag mutabili.

- Immutabile: scegli questa opzione se vuoi evitare che i tag delle immagini vengano sovrascritti e si applica a tutti i tag e le esclusioni presenti nel repository quando inserisci un'immagine con un tag esistente. Amazon ECR restituisce un `ImageTagAlreadyExistsException` messaggio se tenti di inviare un'immagine con un tag esistente. Inoltre, per abilitare gli aggiornamenti dei tag per alcuni tag immutabili, inserisci i nomi dei tag o usa i caratteri jolly (*) per abbinare più tag simili nella casella di testo Esclusione tag immutabile.

7. Per Image scan settings (Impostazioni di scansione delle immagini), mentre è possibile specificare le impostazioni di scansione a livello di repository per la scansione di base, la best practice è specificare la configurazione di scansione a livello di registro privato. Specificare le impostazioni di scansione nel registro privato che consentono di abilitare la scansione avanzata o la scansione di base, nonché di definire i filtri per specificare quali repository vengono scansionati. Per ulteriori informazioni, consulta Scansiona le immagini per individuare le vulnerabilità del software in Amazon ECR.
8. Per Encryption settings (Impostazioni di crittografia), si tratta di un campo di sola visualizzazione in quanto le impostazioni di crittografia per un repository non possono essere modificate una volta creato il repository.
9. Scegliere Save (Salva) per aggiornare le impostazioni del repository.

Impostazione della mutabilità dei tag di immagine ()AWS CLI

Per creare un repository con i tag immutabili configurati

Utilizza uno dei seguenti comandi per creare un nuovo repository di immagini con i tag immutabili configurati.

- create-repository (AWS CLI) con mutabilità dei tag di immagine

```
aws ecr create-repository --repository-name name --image-tag-mutability IMMUTABLE --
region us-east-2
```

- create-repository (AWS CLI) con filtri di esclusione della mutabilità dei tag di immagine

```
aws ecr create-repository --repository-name name --image-tag-
mutability IMMUTABLE_WITH_EXCLUSION --image-tag-mutability-exclusion-filters
 filterType=WILDCARD,filter=filter-text --region us-east-2
```

- Nuovo- ECRRepository (AWS Tools for Windows PowerShell) con mutabilità dei tag di immagine

```
New-ECRRepository -RepositoryName name -ImageTagMutability IMMUTABLE -Region us-
east-2 -Force
```

- Nuovo- ECRRepository (AWS Tools for Windows PowerShell) con filtri di esclusione della mutabilità dei tag di immagine

```
New-ECRRepository -RepositoryName name -ImageTagMutability IMMUTABLE_WITH_EXCLUSION
 -ImageTagMutabilityExclusionFilter @{FilterType=WILDCARD Filter=filter-text} -
Region us-east-2 -Force
```

Per aggiornare le impostazioni di mutabilità dei tag di immagine per un repository

Utilizzare uno dei seguenti comandi per aggiornare le impostazioni di mutabilità dei tag immagine per un repository esistente.

- put-image-tag-mutability(AWS CLI) con mutabilità del tag di immagine

```
aws ecr put-image-tag-mutability --repository-name name --image-tag-
mutability IMMUTABLE --region us-east-2
```

- put-image-tag-mutability(AWS CLI) con filtri di esclusione della mutabilità dei tag di immagine

```
aws ecr put-image-tag-mutability --repository-name name --image-tag-
mutability IMMUTABLE_WITH_EXCLUSION --image-tag-mutability-exclusion-filters
 filterType=WILDCARD,filter=latest --region us-east-2
```

- Write- ECRImage TagMutability (AWS Tools for Windows PowerShell) con mutabilità del tag di immagine

```
Write-ECRImageTagMutability -RepositoryName name -ImageTagMutability IMMUTABLE -
Region us-east-2 -Force
```

- Write- ECRImage TagMutability (AWS Tools for Windows PowerShell) con i filtri di esclusione della mutabilità dei tag di immagine

```
Write-ECRImageTagMutability -RepositoryName name -
ImageTagMutability IMMUTABLE_WITH_EXCLUSION -ImageTagMutabilityExclusionFilter
 @{FilterType=WILDCARD Filter=latest}
```

Supporto del formato manifesto dell'immagine del contenitore in Amazon ECR

Amazon ECR supporta i seguenti formati manifesto per le immagini dei container:

- Docker Image Manifest V2 Schema 1 (utilizzato con Docker versione 1.9 e precedenti)
- Docker Image Manifest V2 Schema 2 (utilizzato con Docker versione 1.10 e successive)
- Specifiche Open Container Initiative (OCI) (v1.0 e v1.1)

Il supporto di Docker Image Manifest V2 Schema 2 offre le seguenti funzionalità:

- La capacità di utilizzare più tag per un'unica immagine.
- Supporto dell'archiviazione delle immagini per i container Windows.

Conversione del manifesto delle immagini Amazon ECR

Quando invii ed estrai immagini da Amazon ECR, il client del motore del container (ad esempio, Docker) comunica con il registro per concordare un formato manifesto che sia comprensibile dal client e dal registro per utilizzare l'immagine.

Quando invii un'immagine ad Amazon ECR con Docker versione 1.9 o precedente, il formato manifesto dell'immagine viene archiviato come Docker Image Manifest V2 Schema 1. Quando invii un'immagine ad Amazon ECR con Docker versione 1.10 o successiva, il formato manifesto dell'immagine viene archiviato come Docker Image Manifest V2 Schema 2.

Quando estrai un'immagine da Amazon ECR in base al tag, Amazon ECR restituisce il formato manifesto dell'immagine archiviato nel repository. Il formato viene restituito solo se è supportato dal client. Se il formato manifesto dell'immagine memorizzato non è supportato dal client, Amazon ECR converte il manifesto dell'immagine in un formato compreso. Ad esempio, se un client Docker 1.9 richiede un manifesto dell'immagine che è memorizzato come Docker Image Manifest V2 Schema 2, Amazon ECR restituisce il manifesto nel formato Docker Image Manifest V2 Schema 1. La tabella

che segue descrive le conversioni disponibili supportate da Amazon ECR quando un'immagine viene estratta in base al tag:

Schema richiesto in base al client	Inviato a ECR come V2, schema 1	Inviato a ECR come V2, schema 2	Inviato a ECR come OCI
V2, schema 1	Nessuna conversione richiesta	Convertito in V2, schema 1	Nessuna conversione disponibile
V2, schema 2	Nessuna conversio ne disponibile, il client utilizza V2, schema 1	Nessuna conversione richiesta	Convertito in V2, schema 2
OCI	Nessuna conversione disponibile	Convertito in OCI	Nessuna conversione richiesta

Important

Se si estrae un'immagine in base al digest, non è disponibile alcuna traduzione. Il cliente deve comprendere il formato manifesto delle immagini archiviato in Amazon ECR. Se richiedi un'immagine Docker Image Manifest V2 Schema 2 per digest su un client Docker 1.9 o precedente, l'estrazione dell'immagine non va a buon fine. Per ulteriori informazioni, consulta Compatibilità del registro nella documentazione Docker.

In questo esempio, se richiedi la stessa immagine in base al tag, Amazon ECR converte il formato manifesto dell'immagine in un formato che il client possa comprendere. L'immagine viene estratta con esito positivo.

Utilizzo delle immagini Amazon ECR con Amazon ECS

Puoi utilizzare i tuoi repository privati Amazon ECR per ospitare immagini di container e artefatti da cui le tue attività Amazon ECS possono eseguire il pull. Affinché ciò funzioni, l'agente container Amazon ECS o Fargate deve disporre delle autorizzazioni per creare`ecr:BatchGetImage`, e. `ecr:GetDownloadUrlForLayer` `ecr:GetAuthorizationToken` APIs

Autorizzazioni IAM richieste

La tabella seguente mostra il ruolo IAM da utilizzare, per ogni tipo di avvio, che fornisce le autorizzazioni necessarie per il pull delle tue attività da un repository privato Amazon ECS. Amazon ECS fornisce policy IAM gestite che includono le autorizzazioni richieste.

Tipo di lancio	Ruolo IAM	AWS politica IAM gestita
Amazon ECS su istanze Amazon EC2	Utilizza il ruolo IAM dell'istanza del contenitore, associato all' EC2 istanza Amazon registrat a nel tuo cluster Amazon ECS. Per maggiori informazioni, consulta Ruolo IAM dell'istanza di container nella Guida per lo sviluppatore di Amazon Elastic Container Service.	`AmazonEC2ContainerServiceforEC2Role` Per ulteriori informazioni, consulta AmazonEC2ContainerServiceforEC2Role nella Guida per lo sviluppatore di Amazon Elastic Container Service.
Amazon ECS su Fargate	Usa il ruolo IAM di esecuzione attività a cui fai riferimento nella definizione dell'attività Amazon ECS. Per ulteriori informazioni, consulta Ruolo IAM per l'esecuzione attività nella Guida per lo sviluppatore di Amazon Elastic Container Service.	`AmazonECSTaskExecutionRolePolicy` Per ulteriori informazioni, consulta AmazonECSTaskExecutionRolePolicy nella Guida per lo sviluppatore di Amazon Elastic Container Service.
Amazon ECS su istanze esterne	Utilizza il ruolo IAM dell'istanza di container associato al server on-premise o alla macchina virtuale (VM) registrata nel tuo cluster Amazon ECS. Per ulteriori informazioni, consulta Ruolo Amazon ECS dell'istanza di container nella Guida per lo	`AmazonEC2ContainerServiceforEC2Role` Per ulteriori informazioni, consulta AmazonEC2ContainerServiceforEC2Role nella Guida per lo sviluppatore di Amazon Elastic Container Service.

Tipo di lancio	Ruolo IAM	AWS politica IAM gestita
	sviluppatore di Amazon Elastic Container Service.	

Important

Le politiche IAM AWS gestite contengono autorizzazioni aggiuntive che potresti non richiedere per il tuo utilizzo. In questo caso, queste sono le autorizzazioni minime richieste per eseguire il pull da un repository privato Amazon ECR.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "ecr:BatchGetImage",
                "ecr:GetDownloadUrlForLayer",
                "ecr:GetAuthorizationToken"
            ],
            "Resource": "*"
        }
    ]
}
```

Specifica di un'immagine Amazon ECR in una definizione attività Amazon ECS

Quando crei una definizione attività Amazon ECS, puoi specificare un'immagine di container ospitata in un repository privato Amazon ECR. Nella definizione attività accertati di utilizzare la denominazione completa di `registry/repository:tag` per le immagini Amazon ECR. Ad esempio, *`aws_account_id`*`.dkr.ecr.`*`region`*`.amazonaws.com/`*`my-repository`*`:`*`latest`*.

Lo snippet di definizione dell'attività seguente mostra la sintassi da utilizzare per specificare un'immagine container ospitata in Amazon ECR nella definizione dell'attività Amazon ECS.

```
{
    "family": "task-definition-name",
    ...
    "containerDefinitions": [
        {
            "name": "container-name",
            "image": "aws_account_id.dkr.ecr.region.amazonaws.com/my-
repository:latest",
            ...
        }
    ],
    ...
}
```

Utilizzo delle immagini Amazon ECR con Amazon EKS

Puoi usare le tue immagini Amazon ECR con Amazon EKS.

Quando si fa riferimento a un'immagine da Amazon ECR, è necessario utilizzare la denominazione completa `registry/repository:tag` per l'immagine. Ad esempio, *aws_account_id*`.dkr.ecr.`*region*`.amazonaws.com/`*my-repository*`:`*latest*.

Autorizzazioni IAM richieste

Se disponi di carichi di lavoro Amazon EKS ospitati su nodi gestiti, nodi autogestiti oppure AWS Fargate, consulta quanto segue:

- Carichi di lavoro Amazon EKS ospitati su nodi gestiti o autogestiti: è richiesto il ruolo IAM (`NodeInstanceRole`) del nodo di lavoro Amazon EKS. Il ruolo IAM del nodo worker Amazon EKS deve contenere le seguenti autorizzazioni delle policy IAM per Amazon ECR.

  JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
```

```
            "Effect": "Allow",
            "Action": [
                "ecr:BatchCheckLayerAvailability",
                "ecr:BatchGetImage",
                "ecr:GetDownloadUrlForLayer",
                "ecr:GetAuthorizationToken"
            ],
            "Resource": "*"
        }
    ]
}
```

> **Note**
>
> Se hai utilizzato `eksctl` o i CloudFormation modelli in Getting Started with Amazon EKS per creare i gruppi di cluster e nodi di lavoro, queste autorizzazioni IAM vengono applicate al ruolo IAM del nodo di lavoro per impostazione predefinita.

- Carichi di lavoro Amazon EKS ospitati su AWS Fargate: utilizza il ruolo di esecuzione del pod Fargate, che fornisce ai tuoi pod l'autorizzazione a estrarre immagini da repository Amazon ECR privati. Per ulteriori informazioni, consulta Creazione di un ruolo di esecuzione pod Fargate.

Installazione di un grafico Helm su un cluster Amazon EKS

I grafici Helm ospitati in Amazon ECR possono essere installati sui cluster Amazon EKS.

Prerequisiti

- Installa la versione più recente del client Helm. Questi passaggi sono stati scritti utilizzando la versione `3.9.0` di Helm. Per ulteriori informazioni, consulta l'argomento relativo all'installazione di Helm.
- Hai almeno la versione `1.23.9` o `2.6.3` del AWS CLI installata sul computer. Per ulteriori informazioni, consulta Installare o aggiornare la versione più recente della AWS CLI.
- Hai inviato un grafico Helm al tuo repository Amazon ECR. Per ulteriori informazioni, consulta Trasferimento di un grafico Helm a un repository privato Amazon ECR.
- Hai configurato `kubectl` affinché lavori con Amazon EKS. Per ulteriori informazioni, consulta Crea un `kubeconfig` per Amazon EKS nella Guida per l'utente di Amazon EKS. Se i comandi seguenti vanno a buon fine per il cluster, la configurazione è corretta.

```
kubectl get svc
```

Per installare un grafico Helm su un cluster Amazon EKS

1. Autentica il tuo client Helm nel registro Amazon ECR che esegue l'hosting del tuo grafico Helm. Devi ottenere i token di autenticazione per ciascun registro utilizzato. I token hanno una validità di 12 ore. Per ulteriori informazioni, consulta Autenticazione del registro privato in Amazon ECR.

   ```
   aws ecr get-login-password \
        --region us-west-2 | helm registry login \
        --username AWS \
        --password-stdin aws_account_id.dkr.ecr.region.amazonaws.com
   ```

2. Installare il grafico. *helm-test-chart*Sostituiscilo con il tuo repository e *0.1.0* con il tag del grafico Helm.

   ```
   helm install ecr-chart-demo oci://aws_account_id.dkr.ecr.region.amazonaws.com/helm-test-chart --version 0.1.0
   ```

   L'output dovrebbe avere questo aspetto:

   ```
   NAME: ecr-chart-demo
   LAST DEPLOYED: Tue May 31 17:38:56 2022
   NAMESPACE: default
   STATUS: deployed
   REVISION: 1
   TEST SUITE: None
   ```

3. Verifica l'installazione del grafico.

   ```
   helm list -n default
   ```

   Output di esempio:

   ```
   NAME            NAMESPACE       REVISION        UPDATED
        STATUS          CHART                   APP VERSION
   ecr-chart-demo  default         1               2022-06-01 15:56:40.128669157 +0000
    UTC deployed        helm-test-chart-0.1.0   1.16.0
   ```

4. (Facoltativo) Vedi il grafico Helm installato `ConfigMap`.

```
kubectl describe configmap helm-test-chart-configmap
```

5. Al termine dell'operazione, puoi rimuovere la versione del grafico dal cluster.

```
helm uninstall ecr-chart-demo
```

Firma immagini in Amazon ECR

Amazon ECR si integra con AWS Signer per fornire due modi per firmare le immagini dei container: firma gestita (automatica, consigliata) e firma manuale (lato client). Puoi archiviare sia le immagini del container sia le firme nei tuoi repository privati.

Scegli un metodo di firma

Amazon ECR supporta due metodi per firmare le immagini dei container:

Firma gestita (consigliata)

La firma gestita genera automaticamente firme crittografiche quando le immagini vengono inviate ad Amazon ECR. Questo metodo semplifica la configurazione. La firma gestita è l'approccio consigliato per la maggior parte degli utenti. Per ulteriori informazioni, consulta Firma gestita.

Firma manuale

La firma manuale utilizza la CLI AWS Signer e il plug-in Notation per firmare le immagini prima di inviarle ad Amazon ECR. Questo metodo offre un maggiore controllo sul processo di firma ed è utile quando è necessario firmare immagini al di fuori del flusso di lavoro push o quando è necessario un controllo preciso sulle operazioni di firma. Per ulteriori informazioni, consulta Firma manuale.

Considerazioni

Quando si utilizza la firma di immagini Amazon ECR, è necessario considerare quanto segue:

- Le firme archiviate nel repository vengono conteggiate nella quota di servizio relativa al numero massimo di immagini per repository. Ogni firma conta come 1 artefatto rispetto alle immagini per quota del repository. Per ulteriori informazioni, consulta Service Quotas di Amazon ECR..
- Quando in un repository sono presenti elementi di riferimento, le policy del ciclo di vita di Amazon ECR li elimineranno automaticamente entro 24 ore dall'eliminazione dell'immagine oggetto.

Firma gestita

La firma gestita di Amazon ECR firma automaticamente le immagini dei container generando firme crittografiche con AWS Signer quando le immagini vengono inviate ad Amazon ECR. Ciò elimina la necessità di installare e configurare strumenti lato client e consente di gestire centralmente la firma come configurazione del registro.

Prerequisiti

Per configurare la firma gestita, crei una configurazione di firma con Amazon ECR che faccia riferimento a uno o più profili di firma Signer e, facoltativamente, filtri di repository che limitano i repository con le immagini firmate. Una volta configurata, la firma gestita di Amazon ECR firma automaticamente le immagini man mano che vengono inviate utilizzando l'identità dell'entità che invia l'immagine.

Prima di poter configurare la firma gestita, devi disporre di quanto segue:

- Un profilo di firma per i firmatari: crea almeno un profilo di firma per i firmatari. Un profilo di firma è una risorsa AWS Signer unica che puoi utilizzare per eseguire operazioni di firma in Amazon ECR. I profili di firma consentono di firmare e verificare elementi del codice, come immagini di container e pacchetti di distribuzione. AWS Lambda Ogni profilo di firma indica la piattaforma di firma a cui firmare, un ID della piattaforma e altre informazioni specifiche della piattaforma. Per esempio, un profilo di firma ARN ha questo aspetto:. `arn:partition:signer:region:account-id:/signing-profiles/profile-name`
- Autorizzazioni IAM: il principale IAM che invia l'immagine deve disporre delle autorizzazioni IAM necessarie per accedere al profilo di firma Signer pertinente e al relativo repository ECR. È necessario modificare la politica basata sull'identità per il principale IAM in modo da includere le autorizzazioni sia per le operazioni del repository ECR che per le operazioni di firma Signer. La seguente politica di esempio mostra le autorizzazioni richieste:

```
{
   "Version": "2012-10-17",
   "Statement": [
   {
       "Sid":"UploadSignaturePermissions",
       "Effect":"Allow",
       "Action":[
           "ecr:CompleteLayerUpload",
```

```
            "ecr:UploadLayerPart",
            "ecr:InitiateLayerUpload",
            "ecr:BatchCheckLayerAvailability",
            "ecr:PutImage"
        ],
        "Resource":"arn:aws:ecr:region:account-id:repository/repository-name"
    },
    {
        "Sid": "SignPermissions",
        "Effect": "Allow",
        "Action": [
            "signer:SignPayload"
        ],
        "Resource": "arn:aws:signer:region:account-id:/signing-profiles/signing-profile-
name"
    }
    ]
}
```

Con la firma gestita di Amazon ECR, puoi creare più regole di firma (fino a 10 per registro) per creare limiti di sicurezza più solidi. Ad esempio, potresti eseguire più pipeline di compilazione e voler limitare i repository che ciascuna pipeline può firmare. All'interno di ogni regola, configuri un profilo di firma e specifichi i filtri dei nomi del repository. Quando viene inviata una nuova immagine, Amazon ECR corrisponde alla regola di firma e al profilo di firma che possono firmare l'immagine. Se ci sono più corrispondenze, Amazon ECR genera più firme.

> **Note**
>
> Se verifichi le firme manualmente, devi comunque installare la CLI di Notation.

> **Note**
>
> La firma gestita di Amazon ECR è disponibile in tutte le AWS regioni in cui è disponibile la firma di immagini di container con AWS Signer.

Nozioni di base

Segui questi passaggi per configurare la firma gestita. Fornisci ad Amazon ECR un riferimento a un profilo di firma del firmatario e, facoltativamente, filtri che limitano i repository con le immagini firmate.

Console di gestione AWS

Utilizza i seguenti passaggi per configurare la firma gestita utilizzando. Console di gestione AWS

1. Apri la console Amazon ECR. Nel riquadro di navigazione a sinistra, seleziona Registro privato, Caratteristiche e impostazioni, Firma gestita.
2. Nella pagina Regole di firma, seleziona Crea regola.
3. Nella pagina Profilo di firma, in Seleziona un profilo AWS firmatario, scegli Crea nuovo profilo AWS firmatario, inserisci un nome di profilo e, facoltativamente, modifica il periodo di validità della firma. Quindi seleziona Avanti.
4. Nella pagina Filtri, in Seleziona i repository, inserisci un filtro per il nome del repository. Quindi seleziona Avanti.
5. Nella pagina Rivedi e crea, verifica i filtri del profilo del AWS firmatario e del nome del repository che hai inserito. Se tutto sembra corretto, seleziona Salva.

AWS CLI

Usa i seguenti AWS CLI comandi per configurare la firma gestita.

- Crea una regola di firma

  Crea una configurazione di firma con il tuo profilo di firma ARN. Crea un file JSON con i seguenti contenuti:

```
{
    "rules": [
        {
            "signingProfileArn": "arn:aws:signer:region:account-id:/signing-
profiles/profile-name",
            "repositoryFilters": [
                {
                    "filter": "test*",
                    "filterType": "WILDCARD_MATCH"
                }
```

```
            ]
        }
    ]
}
```

Quindi, esegui il comando riportato di seguito:

```
aws ecr --region region \
    put-signing-configuration \
    --signing-configuration file://signing-config.json
```

Dovresti vedere la risposta dell'API contenente la configurazione di firma.

- Visualizza la tua configurazione di firma

  Recupera la configurazione di firma:

  ```
  aws ecr --region region \
      get-signing-configuration
  ```

  Dovresti vedere la risposta dell'API contenente la configurazione di firma.

- Controlla lo stato della firma dell'immagine

  Invia un'immagine al tuo repository. Ad esempio:

  ```
  docker pull ubuntu

  IMAGE_NAME="account-id.dkr.ecr.region.amazonaws.com/repository-name"
  IMAGE_TAG="${IMAGE_NAME}:test-1"

  docker tag ubuntu $IMAGE_TAG
  docker push $IMAGE_TAG
  ```

  Dopo aver premuto, usa il tag dell'immagine per controllare lo stato della firma:

  ```
  aws ecr --region region \
      describe-image-signing-status \
      --repository-name repository-name \
      --image-id imageTag=test-1
  ```

Se il nome del repository corrisponde al filtro del repository definito nella configurazione di firma, dovresti vedere lo stato della firma nella risposta dell'API. Se lo stato è positivo, dovresti vedere una firma inserita nel tuo repository.

- Elimina la configurazione di firma

  Eliminare la configurazione di firma:

  ```
  aws ecr --region region \
      delete-signing-configuration
  ```

  Dovresti vedere la risposta API contenente la configurazione di firma eliminata.

Considerazioni

Le seguenti limitazioni e funzionalità si applicano alla firma gestita:

- La firma tra regioni non è supportata: i profili di firma devono trovarsi nella stessa regione del registro Amazon ECR. Non puoi utilizzare un profilo di firma di una regione per firmare immagini in un registro situato in un'altra regione.
- È supportata la firma su più account: i profili di firma possono trovarsi in account diversi rispetto al registro Amazon ECR. Ciò consente alle organizzazioni di gestire centralmente i profili di firma, consentendo al contempo agli sviluppatori di altri account di utilizzarli. Per ulteriori informazioni, consulta Configurare la firma su più account per Signer nella Guida per gli AWS Signer sviluppatori.
- Le firme non possono essere firmate: non puoi firmare le firme da solo. È possibile firmare solo le immagini dei contenitori.

Verifica della firma

Dopo aver firmato le immagini dei contenitori, puoi verificare le firme per assicurarti che le immagini non siano state manomesse e provengano da una fonte attendibile. Amazon ECR supporta diversi metodi per la verifica delle firme:

Verifica gestita con Amazon EKS

Amazon EKS offre un'integrazione nativa per la verifica automatica delle firme. Quando configuri la verifica delle firme nei cluster Amazon EKS, il servizio verifica automaticamente le firme delle immagini prima di consentire l'esecuzione dei contenitori. Per ulteriori informazioni sulla configurazione della verifica delle firme, consulta Convalida delle firme delle immagini dei container durante la distribuzione nella Amazon EKS User Guide.

Controller di ammissione Lambda per Amazon ECS

Amazon ECS fornisce hook per il ciclo di vita del servizio che consentono di eseguire logiche personalizzate durante le distribuzioni dei servizi. Questi hook possono attivare AWS Lambda funzioni in punti specifici del processo di distribuzione, consentendoti di convalidare le firme delle immagini dei container prima di consentire l'avvio dei servizi. Per ulteriori informazioni, consulta Verify container image signatures for Amazon ECS nella AWS Signer Developer Guide.

Verifica manuale con Notation CLI

È possibile verificare le firme manualmente utilizzando la CLI di Notation. Questo metodo richiede l'installazione e la configurazione della CLI di Notation sul computer locale o nell'ambiente di verifica. Per istruzioni dettagliate sulla verifica di un'immagine utilizzando la CLI di Notation, consulta Verificare un'immagine localmente dopo l'accesso alla Developer Guide.AWS Signer

Configura l'autenticazione per il client Notation

Se utilizzi la firma manuale o verifichi le firme manualmente utilizzando la CLI di Notation, devi configurare il client Notation in modo che possa autenticarsi su Amazon ECR. Se hai installato Docker sullo stesso host in cui installi il client Notation, Notation riutilizzerà lo stesso metodo di autenticazione utilizzato per il client Docker. Il Docker `login` e `logout` i comandi consentiranno a Notation `sign` e ai `verify` comandi di utilizzare le stesse credenziali e non è necessario autenticare Notation separatamente. Per ulteriori informazioni sulla configurazione del client Notation per l'autenticazione, consulta Autenticazione con registri conformi a OCI nella documentazione di Notary Project.

Se non utilizzi Docker o un altro strumento che utilizza le credenziali Docker, ti consigliamo di utilizzare l'assistente di gestione credenziali Docker di Amazon ECR come archivio di credenziali. Per ulteriori informazioni su come installare e configurare l'assistente credenziali di Amazon ECR, consulta Assistente di gestione credenziali Docker di Amazon ECR.

Firma manuale

La firma manuale utilizza la CLI AWS Signer e il plug-in Notation per firmare le immagini prima di inviarle ad Amazon ECR. Questo metodo offre un maggiore controllo sul processo di firma ed è utile quando è necessario firmare immagini al di fuori del flusso di lavoro push o quando è necessario un controllo preciso sulle operazioni di firma.

Per istruzioni dettagliate sulla firma delle immagini dei contenitori utilizzando la CLI di Notation AWS Signer e, consulta Firmare le immagini dei contenitori in Signer e gli argomenti correlati nella AWS Signer Guida per gli sviluppatori.

Prerequisiti

Prima di iniziare, verifica che siano stati soddisfatti i seguenti requisiti preliminari.

- Installazione e configurazione della versione più recente di AWS CLI. Per ulteriori informazioni, consulta Installazione o aggiornamento della versione più recente della AWS CLI nella Guida per l'utente di AWS Command Line Interface .
- Installa la Notation CLI e AWS Signer il plugin per Notation. Per ulteriori informazioni, consulta Prerequisiti per la firma delle immagini di container nella Guida per gli sviluppatori di AWS Signer .
- Conserva un'immagine del container da firmare in un repository privato Amazon ECR. Per ulteriori informazioni, consulta Invio di un'immagine a un repository privato Amazon ECR.

Scansiona le immagini per individuare le vulnerabilità del software in Amazon ECR

La scansione delle immagini di Amazon ECR aiuta a identificare le vulnerabilità del software nelle immagini dei container. Sono disponibili i seguenti tipi di scansione.

Important

Il passaggio tra le versioni di scansione avanzata, scansione di base e scansione di base migliorata farà sì che le scansioni precedentemente stabilite non siano più disponibili. Dovrai configurare nuovamente le scansioni. Tuttavia, se si torna alla versione di scansione precedente, le scansioni stabilite saranno disponibili.

Note

Le immagini archiviate non possono essere scansionate. Le immagini archiviate devono essere ripristinate prima di poter essere scansionate. Per ulteriori informazioni sull'archiviazione e il ripristino delle immagini, vedere. Archiviazione di un'immagine in Amazon ECR

- Scansione migliorata: Amazon ECR si integra con Amazon Inspector per fornire una scansione automatica e continua dei tuoi repository. Le immagini di container vengono analizzate alla ricerca di vulnerabilità sia per i sistemi operativi che per i pacchetti del linguaggio di programmazione. Man mano che compaiono nuove vulnerabilità, i risultati della scansione vengono aggiornati e Amazon Inspector emette un evento EventBridge per avvisarti. La scansione avanzata offre quanto segue:
  - Vulnerabilità dei pacchetti del sistema operativo e dei linguaggi di programmazione
  - Due frequenze di scansione: scansione in modalità push e scansione continua
- Scansione di base: Amazon ECR offre due versioni di scansione di base che utilizzano il database Common Vulnerabilities and Exposures ()CVEs:.
  - AWS scansione nativa di base: utilizza la tecnologia AWS nativa, che ora è GA e consigliata. Tutti i nuovi registri dei clienti sono inseriti in questa versione migliorata per impostazione predefinita.

- Scansione di base Clair: utilizza il progetto Clair open source. Clair è ora obsoleto. Per informazioni dettagliate, vedi Deprecazione di Clair.

Con la scansione di base, configuri i repository per la scansione tramite push o puoi eseguire scansioni manuali e Amazon ECR fornisce un elenco dei risultati della scansione. La scansione di base fornisce quanto segue:

- Scansioni del sistema operativo
- Due frequenze di scansione: manuale e scansione a pressione

> **Important**
>
> La nuova versione di Amazon ECR Basic Scanning non utilizza gli `imageScanStatus` attributi `imageScanFindingsSummary` and della risposta dell'`DescribeImages`API per restituire i risultati della scansione. Utilizza invece l' `DescribeImageScanFindings`API. Per ulteriori informazioni, consulta `DescribeImageScanFindings`.

Filtri per scegliere quali repository scansionare in Amazon ECR

Quando configuri la scansione delle immagini per il tuo registro privato, puoi utilizzare i filtri per scegliere quali repository sottoporre a scansione.

Quando viene utilizzata la scansione di base, è possibile specificare la scansione sui filtri push per specificare quali repository sono impostati per eseguire una scansione delle immagini quando ne vengono inserite di nuove. Tutti gli archivi che non corrispondono a una scansione di base sul filtro push verranno impostati sulla frequenza di scansione manuale, il che significa che per eseguire una scansione è necessario attivarla manualmente.

Quando viene utilizzata la scansione avanzata, è possibile specificare filtri separati per la scansione su push e la scansione continua. Tutti i repository che non corrispondono a un filtro di scansione avanzato avranno la scansione disattivata. Se si utilizza la scansione avanzata e si specificano filtri separati per la scansione su push e la scansione continua in cui più filtri corrispondono allo stesso repository, Amazon ECR applica il filtro di scansione continua sul filtro della scansione su push per quel repository.

Filtra i caratteri jolly

Quando viene specificato un filtro, un filtro senza caratteri jolly corrisponderà a tutti i nomi del repository che contengono il filtro. Un filtro con caratteri jolly (*) corrisponde a qualsiasi nome del repository in cui il carattere jolly sostituisce zero o più caratteri nel nome del repository.

La tabella seguente fornisce esempi in cui i nomi del repository sono espressi sull'asse orizzontale e i filtri di esempio sono specificati sull'asse verticale.

	prod	repo-prod	prod-repo	repo-prod-repo	prodrepo
prod	Sì	Sì	Sì	Sì	Sì
*prod	Sì	Sì	No	No	No
prod*	Sì	No	Sì	No	Sì
prod	Sì	Sì	Sì	Sì	Sì
prod*repo	No	No	Sì	No	Sì

Scansiona le immagini per individuare le vulnerabilità del sistema operativo e dei pacchetti del linguaggio di programmazione in Amazon ECR

La scansione avanzata di Amazon ECR è un'integrazione con Amazon Inspector che fornisce la scansione delle vulnerabilità delle immagini dei container. Le immagini di container vengono analizzate alla ricerca di vulnerabilità sia per i sistemi operativi che per i pacchetti del linguaggio di programmazione. Puoi visualizzare direttamente i risultati della scansione sia con Amazon ECR che con Amazon Inspector. Per ulteriori informazioni su Amazon Inspector, consulta la sezione Scansione delle immagini dei container con Amazon Inspector nella Guida per l'utente di Amazon Inspector.

Con la scansione avanzata, è possibile scegliere quali repository sono configurati per la scansione automatica e continua e quali sono configurati per la scansione su push. Questo viene fatto impostando i filtri di scansione.

Considerazioni per le scansioni avanzate

Considera quanto segue prima di abilitare la scansione avanzata di Amazon ECR.

- Amazon ECR non prevede costi aggiuntivi per l'utilizzo di questa funzione ma Amazon Inspector prevede un costo per la scansione delle immagini. Questa funzionalità è disponibile nelle regioni in cui è supportato Amazon Inspector. Per ulteriori informazioni, consulta:
  - Prezzi di Amazon Inspector: prezzi di Amazon Inspector.
  - Regioni supportate da Amazon Inspector: regioni ed endpoint.
- La scansione avanzata di Amazon ECR mostra come vengono utilizzate le immagini su Amazon EKS e Amazon ECS. Puoi vedere quando le immagini sono state utilizzate l'ultima volta e identificare quanti cluster utilizzano ciascuna immagine. Queste informazioni aiutano a dare priorità alla correzione delle vulnerabilità per le immagini utilizzate attivamente. È possibile determinare rapidamente quali cluster potrebbero essere interessati dalle nuove vulnerabilità scoperte. Per ulteriori informazioni su come richiedere queste informazioni e visualizzare la risposta, vedere. `DescribeImageScanFindings`
- Amazon Inspector supporta la scansione per sistemi operativi specifici. Per un elenco completo, consulta la sezione Sistemi operativi supportati - Scansione Amazon ECR nella Guida per l'utente di Amazon Inspector.
- Amazon Inspector utilizza un ruolo IAM collegato al servizio, che fornisce le autorizzazioni necessarie per fornire una scansione avanzata dei repository. Il ruolo IAM collegato al servizio viene creato automaticamente da Amazon Inspector quando è attivata la scansione avanzata del registro privato. Per ulteriori informazioni, consulta la sezione Utilizzo di ruoli collegati ai servizi per Amazon Inspector nella Guida per l'utente di Amazon Inspector.
- Quando attivi inizialmente la scansione avanzata per il tuo registro privato, Amazon Inspector riconosce solo le immagini inviate ad Amazon ECR negli ultimi 14 giorni, in base al timestamp dell'immagine. Le immagini più vecchie avranno lo stato di scansione `SCAN_ELIGIBILITY_EXPIRED`. Se desideri che queste immagini vengano scansionate da Amazon Inspector, devi reinserirle nel tuo repository.
- Quando per il registro privato Amazon ECR è abilitata la scansione avanzata, i repository corrispondenti ai filtri di scansione vengono scansionati solo utilizzando la scansione avanzata. I repository che non corrispondono a un filtro avranno una frequenza di scansione `Off`, ma non saranno sottoposti a scansione. Le scansioni manuali che utilizzano la scansione avanzata non sono supportate. Per ulteriori informazioni, consulta Filtri per scegliere quali repository scansionare in Amazon ECR.

- Se si specificano filtri separati per la scansione su push e la scansione continua in cui più filtri corrispondono allo stesso repository, Amazon ECR applica il filtro di scansione continua sul filtro della scansione su push per quel repository.
- Quando la scansione avanzata è attivata, Amazon ECR invia un evento a EventBridge quando viene modificata la frequenza di scansione di un repository. Amazon Inspector emette eventi EventBridge quando viene completata una scansione iniziale e quando viene creato, aggiornato o chiuso un risultato di scansione di immagini.

Modifica della durata di scansione migliorata per le immagini in Amazon Inspector

Dopo aver abilitato la scansione avanzata, Amazon ECR esegue continuamente la scansione delle immagini appena inviate per la durata configurata. Per impostazione predefinita, Amazon Inspector monitora i tuoi repository fino all'eliminazione delle immagini o alla disattivazione della scansione avanzata. Puoi configurare sia la durata della data di push (fino a Lifetime) sia la durata della nuova scansione nella console Amazon Inspector in base alle esigenze del tuo ambiente. Quando la durata della scansione di un repository è scaduta, lo stato della scansione viene visualizzato come. `SCAN_ELIGIBILITY_EXPIRED` Per ulteriori informazioni sulla configurazione delle impostazioni della durata della nuova scansione per Amazon ECR in Amazon Inspector, consulta Configuring the Amazon ECR re-scan duration nella Amazon Inspector User Guide.

Autorizzazioni IAM necessarie per una scansione avanzata in Amazon ECR

La scansione avanzata di Amazon ECR richiede un ruolo IAM collegato al servizio Amazon Inspector e che il responsabile IAM che abilita e utilizza la scansione avanzata disponga delle autorizzazioni per chiamare Amazon Inspector necessario per la scansione. APIs Il ruolo IAM collegato al servizio Amazon Inspector viene creato automaticamente da Amazon Inspector quando è attivata la scansione avanzata del registro privato. Per ulteriori informazioni, consulta la sezione Utilizzo di ruoli collegati ai servizi per Amazon Inspector nella Guida per l'utente di Amazon Inspector.

La seguente policy IAM concede le autorizzazioni necessarie per l'attivazione e l'utilizzo della scansione avanzata. Include l'autorizzazione necessaria ad Amazon Inspector per creare il ruolo IAM collegato al servizio e le autorizzazioni API di Amazon Inspector necessarie per attivare e disattivare la scansione avanzata e recuperare i risultati della scansione.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "inspector2:Enable",
                "inspector2:Disable",
                "inspector2:ListFindings",
                "inspector2:ListAccountPermissions",
                "inspector2:ListCoverage"
            ],
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": "iam:CreateServiceLinkedRole",
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "iam:AWSServiceName": [
                        "inspector2.amazonaws.com"
                    ]
                }
            }
        }
    ]
}
```

Configurazione della scansione avanzata delle immagini in Amazon ECR

Configura la scansione avanzata per regione per il tuo registro privato.

Verifica di disporre delle autorizzazioni IAM appropriate per configurare la scansione avanzata. Per informazioni, consulta Autorizzazioni IAM necessarie per una scansione avanzata in Amazon ECR.

Console di gestione AWS

Per attivare la scansione avanzata per il registro privato

1. Apri la console Amazon ECR nei https://console.aws.amazon.com/ecr/repository.
2. Dalla barra di navigazione, scegli la regione in cui impostare la configurazione della scansione.
3. Nel pannello di navigazione, scegli Registro privato, quindi scegli Impostazioni.
4. Nella pagina Scanning configuration (Configurazione della scansione), per Scan type (Tipo di scansione) scegli Enhanced scanning (Scansione avanzata).

   Per impostazione predefinita, quando è selezionata l'opzione Scansione avanzata, tutti gli archivi vengono sottoposti a scansione continua.
5. Per scegliere repository specifici da scansionare continuamente, deseleziona la casella Scansione continua di tutti gli archivi, quindi definisci i filtri:

   > **Important**
   >
   > I filtri senza caratteri jolly corrisponderanno a tutti i nomi del repository che contengono il filtro. I filtri con caratteri jolly (*) corrispondono a qualsiasi nome di un repository in cui il carattere jolly sostituisce zero o più caratteri nel nome del repository. Per vedere esempi di come si comportano i filtri, consulta. the section called “Filtra i caratteri jolly”

   a. Inserisci un filtro basato sui nomi dei repository, quindi scegli Aggiungi filtro.

   b. Decidi quali repository scansionare quando viene inviata un'immagine:

      - Per scansionare tutti i repository in modalità push, seleziona Scansiona in push tutti i repository.
      - Per scegliere repository specifici da scansionare al momento del push, inserisci un filtro basato sui nomi dei repository, quindi scegli Aggiungi filtro.
6. Scegli Save (Salva).
7. Ripeti questi passaggi in ogni regione in cui desideri attivare la scansione avanzata.

AWS CLI

Usa il seguente AWS CLI comando per attivare la scansione avanzata per il tuo registro privato utilizzando il. AWS CLIÈ possibile specificare i filtri di scansione utilizzando l'oggetto `rules`.

- put-registry-scanning-configuration (AWS CLI)

  L'esempio seguente consente di attivare la scansione avanzata del registro privato. Per impostazione predefinita, quando non sono state specificate `rules`, Amazon ECR imposta la configurazione di scansione su una scansione continua per tutti i repository.

```
aws ecr put-registry-scanning-configuration \
     --scan-type ENHANCED \
     --region us-east-2
```

  L'esempio seguente consente di attivare la scansione avanzata del registro privato e specifica un filtro di scansione. Il filtro di scansione nell'esempio consente di attivare la scansione continua di tutti i repository con `prod` nel suo nome.

```
aws ecr put-registry-scanning-configuration \
     --scan-type ENHANCED \
     --rules '[{"repositoryFilters" : [{"filter":"prod","filterType" :
 "WILDCARD"}],"scanFrequency" : "CONTINUOUS_SCAN"}]' \
     --region us-east-2
```

  L'esempio seguente consente di attivare la scansione avanzata del registro privato e specifica più filtri di scansione. I filtri di scansione nell'esempio consentono di attivare la scansione continua di tutti i repository con `prod` nel nome e consentono di attivare la scansione su push solo per tutti gli altri repository.

```
aws ecr put-registry-scanning-configuration \
     --scan-type ENHANCED \
     --rules '[{"repositoryFilters" : [{"filter":"prod","filterType" :
 "WILDCARD"}],"scanFrequency" : "CONTINUOUS_SCAN"},{"repositoryFilters" :
 [{"filter":"*","filterType" : "WILDCARD"}],"scanFrequency" : "SCAN_ON_PUSH"}]' \
     --region us-west-2
```

EventBridge eventi inviati per una scansione avanzata in Amazon ECR

Quando la scansione avanzata è attivata, Amazon ECR invia un evento a EventBridge quando viene modificata la frequenza di scansione di un repository. Amazon Inspector invia eventi EventBridge quando viene completata una scansione iniziale e quando viene creato, aggiornato o chiuso un risultato di scansione dell'immagine.

Evento per una modifica della frequenza di scansione del repository

Quando la scansione avanzata è attivata per il registro, il seguente evento viene inviato da Amazon ECR quando si verifica una modifica con una risorsa che ha attivato la scansione avanzata. Ciò include la creazione di nuovi repository, la modifica della frequenza di scansione di un repository o la creazione o eliminazione di immagini nei repository con la scansione avanzata attivata. Per ulteriori informazioni, consulta Scansiona le immagini per individuare le vulnerabilità del software in Amazon ECR.

```
{
 "version": "0",
 "id": "0c18352a-a4d4-6853-ef53-0abEXAMPLE",
 "detail-type": "ECR Scan Resource Change",
 "source": "aws.ecr",
 "account": "123456789012",
 "time": "2021-10-14T20:53:46Z",
 "region": "us-east-1",
 "resources": [],
 "detail": {
  "action-type": "SCAN_FREQUENCY_CHANGE",
  "repositories": [{
    "repository-name": "repository-1",
    "repository-arn": "arn:aws:ecr:us-east-1:123456789012:repository/repository-1",
    "scan-frequency": "SCAN_ON_PUSH",
    "previous-scan-frequency": "MANUAL"
   },
   {
    "repository-name": "repository-2",
    "repository-arn": "arn:aws:ecr:us-east-1:123456789012:repository/repository-2",
    "scan-frequency": "CONTINUOUS_SCAN",
    "previous-scan-frequency": "SCAN_ON_PUSH"
   },
   {
    "repository-name": "repository-3",
    "repository-arn": "arn:aws:ecr:us-east-1:123456789012:repository/repository-3",
```

```
    "scan-frequency": "CONTINUOUS_SCAN",
    "previous-scan-frequency": "SCAN_ON_PUSH"
   }
  ],
  "resource-type": "REPOSITORY",
  "scan-type": "ENHANCED"
 }
}
```

Evento per una scansione iniziale dell'immagine (scansione avanzata)

Quando la scansione avanzata è attivata per il registro, Amazon Inspector invia il seguente evento al termine della scansione iniziale dell'immagine. Il parametro `finding-severity-counts` restituirà un valore per un livello di gravità solo se ne esiste uno. Ad esempio, se l'immagine non contiene risultati a livello `CRITICAL`, non viene restituito alcun conteggio critico. Per ulteriori informazioni, consulta Scansiona le immagini per individuare le vulnerabilità del sistema operativo e dei pacchetti del linguaggio di programmazione in Amazon ECR.

Modello di eventi:

```
{
  "source": ["aws.inspector2"],
  "detail-type": ["Inspector2 Scan"]
}
```

Output di esempio:

```
{
    "version": "0",
    "id": "739c0d3c-4f02-85c7-5a88-94a9EXAMPLE",
    "detail-type": "Inspector2 Scan",
    "source": "aws.inspector2",
    "account": "123456789012",
    "time": "2021-12-03T18:03:16Z",
    "region": "us-east-2",
    "resources": [
        "arn:aws:ecr:us-east-2:123456789012:repository/amazon/amazon-ecs-sample"
    ],
    "detail": {
        "scan-status": "INITIAL_SCAN_COMPLETE",
        "repository-name": "arn:aws:ecr:us-east-2:123456789012:repository/amazon/
amazon-ecs-sample",
```

```
        "finding-severity-counts": {
            "CRITICAL": 7,
            "HIGH": 61,
            "MEDIUM": 62,
            "TOTAL": 158
        },
        "image-digest":
 "sha256:36c7b282abd0186e01419f2e58743e1bf635808231049bbc9d77e5EXAMPLE",
        "image-tags": [
            "latest"
        ]
    }
}
```

Evento per un aggiornamento del risultato della scansione dell'immagine (scansione avanzata)

Quando la scansione avanzata è attivata per il registro, Amazon Inspector invia il seguente evento quando il rilevamento della scansione delle immagini viene creato, aggiornato o chiuso. Per ulteriori informazioni, consulta Scansiona le immagini per individuare le vulnerabilità del sistema operativo e dei pacchetti del linguaggio di programmazione in Amazon ECR.

Modello di eventi:

```
{
  "source": ["aws.inspector2"],
  "detail-type": ["Inspector2 Finding"]
}
```

Output di esempio:

```
{
    "version": "0",
    "id": "42dbea55-45ad-b2b4-87a8-afaEXAMPLE",
    "detail-type": "Inspector2 Finding",
    "source": "aws.inspector2",
    "account": "123456789012",
    "time": "2021-12-03T18:02:30Z",
    "region": "us-east-2",
    "resources": [
        "arn:aws:ecr:us-east-2:123456789012:repository/amazon/amazon-ecs-sample/
sha256:36c7b282abd0186e01419f2e58743e1bf635808231049bbc9d77eEXAMPLE"
    ],
    "detail": {
```

```
        "awsAccountId": "123456789012",
        "description": "In libssh2 v1.9.0 and earlier versions, the SSH_MSG_DISCONNECT
 logic in packet.c has an integer overflow in a bounds check, enabling an attacker to
 specify an arbitrary (out-of-bounds) offset for a subsequent memory read. A crafted
 SSH server may be able to disclose sensitive information or cause a denial of service
 condition on the client system when a user connects to the server.",
        "findingArn": "arn:aws:inspector2:us-east-2:123456789012:finding/
be674aaddd0f75ac632055EXAMPLE",
        "firstObservedAt": "Dec 3, 2021, 6:02:30 PM",
        "inspectorScore": 6.5,
        "inspectorScoreDetails": {
            "adjustedCvss": {
                "adjustments": [],
                "cvssSource": "REDHAT_CVE",
                "score": 6.5,
                "scoreSource": "REDHAT_CVE",
                "scoringVector": "CVSS:3.0/AV:N/AC:L/PR:N/UI:R/S:U/C:H/I:N/A:N",
                "version": "3.0"
            }
        },
        "lastObservedAt": "Dec 3, 2021, 6:02:30 PM",
        "packageVulnerabilityDetails": {
            "cvss": [
                {
                    "baseScore": 6.5,
                    "scoringVector": "CVSS:3.0/AV:N/AC:L/PR:N/UI:R/S:U/C:H/I:N/A:N",
                    "source": "REDHAT_CVE",
                    "version": "3.0"
                },
                {
                    "baseScore": 5.8,
                    "scoringVector": "AV:N/AC:M/Au:N/C:P/I:N/A:P",
                    "source": "NVD",
                    "version": "2.0"
                },
                {
                    "baseScore": 8.1,
                    "scoringVector": "CVSS:3.1/AV:N/AC:L/PR:N/UI:R/S:U/C:H/I:N/A:H",
                    "source": "NVD",
                    "version": "3.1"
                }
            ],
            "referenceUrls": [
                "https://access.redhat.com/errata/RHSA-2020:3915"
```

```
            ],
            "source": "REDHAT_CVE",
            "sourceUrl": "https://access.redhat.com/security/cve/CVE-2019-17498",
            "vendorCreatedAt": "Oct 16, 2019, 12:00:00 AM",
            "vendorSeverity": "Moderate",
            "vulnerabilityId": "CVE-2019-17498",
            "vulnerablePackages": [
                {
                    "arch": "X86_64",
                    "epoch": 0,
                    "name": "libssh2",
                    "packageManager": "OS",
                    "release": "12.amzn2.2",
                    "sourceLayerHash":
"sha256:72d97abdfae3b3c933ff41e39779cc72853d7bd9dc1e4800c5294dEXAMPLE",
                    "version": "1.4.3"
                }
            ]
        },
        "remediation": {
            "recommendation": {
                "text": "Update all packages in the vulnerable packages section to
their latest versions."
            }
        },
        "resources": [
            {
                "details": {
                    "awsEcrContainerImage": {
                        "architecture": "amd64",
                        "imageHash":
"sha256:36c7b282abd0186e01419f2e58743e1bf635808231049bbc9d77e5EXAMPLE",
                        "imageTags": [
                            "latest"
                        ],
                        "platform": "AMAZON_LINUX_2",
                        "pushedAt": "Dec 3, 2021, 6:02:13 PM",
                        "lastInUseAt": "Dec 3, 2021, 6:02:13 PM",
                        "inUseCount": 1,
                        "registry": "123456789012",
                        "repositoryName": "amazon/amazon-ecs-sample"
                    }
                },
```

```
                "id": "arn:aws:ecr:us-east-2:123456789012:repository/amazon/amazon-ecs-
sample/sha256:36c7b282abd0186e01419f2e58743e1bf635808231049bbc9d77EXAMPLE",
                "partition": "N/A",
                "region": "N/A",
                "type": "AWS_ECR_CONTAINER_IMAGE"
            }
        ],
        "severity": "MEDIUM",
        "status": "ACTIVE",
        "title": "CVE-2019-17498 - libssh2",
        "type": "PACKAGE_VULNERABILITY",
        "updatedAt": "Dec 3, 2021, 6:02:30 PM"
    }
}
```

Recupero dei risultati per scansioni avanzate in Amazon ECR

Puoi recuperare i risultati della scansione per l'ultima scansione dell'immagine migliorata completata, quindi aprirli in Amazon Inspector per visualizzare ulteriori dettagli. Le vulnerabilità del software scoperte sono elencate per gravità in base al database Common Vulnerabilities and Exposures (). CVEs

Per la risoluzione dei problemi relativi ad alcuni problemi comuni durante la scansione delle immagini, consulta Risoluzione dei problemi di scansione delle immagini in Amazon ECR.

Console di gestione AWS

Attieniti alla seguente procedura per recuperare i risultati della scansione delle immagini utilizzando la Console di gestione AWS.

Per recuperare i risultati della scansione delle immagini

1. Apri la console Amazon ECR all'indirizzo https://console.aws.amazon.com/ecr/.
2. Dalla barra di navigazione, scegli la regione in cui si trova il repository.
3. Nel riquadro di navigazione, selezionare Repositories (Repository).
4. Nella pagina Repositories (Repository), scegliere il repository che contiene l'immagine per cui recuperare i risultati della scansione.
5. Nella pagina Immagini, nella colonna Tag immagine, seleziona il tag dell'immagine per recuperare i risultati della scansione.

6. Per visualizzare maggiori dettagli nella console Amazon Inspector, scegli il nome della vulnerabilità nella colonna Nome.

AWS CLI

Usa il seguente AWS CLI comando per recuperare i risultati della scansione delle immagini utilizzando. AWS CLI Puoi specificare un'immagine utilizzando `imageTag` o `imageDigest`, entrambe ottenibili utilizzando il comando CLI list-images.

- describe-image-scan-findings (AWS CLI)

  Nell'esempio seguente viene utilizzato un tag di immagine.

```
aws ecr describe-image-scan-findings \
     --repository-name name \
     --image-id imageTag=tag_name \
     --region us-east-2
```

  Nell'esempio seguente viene utilizzato un digest di immagine.

```
aws ecr describe-image-scan-findings \
     --repository-name name \
     --image-id imageDigest=sha256_hash \
     --region us-east-2
```

Scansiona le immagini per individuare le vulnerabilità del sistema operativo in Amazon ECR

Amazon ECR offre due versioni di scansione di base che utilizzano il database Common Vulnerabilities and Exposures (): CVEs

- AWS scansione nativa di base: utilizza la tecnologia AWS nativa, che ora è GA e consigliata. Questa scansione di base migliorata è progettata per fornire ai clienti risultati di scansione migliori e un rilevamento delle vulnerabilità più accurato su un'ampia gamma di sistemi operativi diffusi. Ciò permette ai clienti di rafforzare ulteriormente la sicurezza delle immagini di container. Tutti i nuovi registri dei clienti sono inseriti in questa versione migliorata per impostazione predefinita.

- Scansione di base di Clair — La versione di scansione di base precedente, che utilizza il progetto Clair open source (vedi Clair su). GitHub Clair è ora obsoleto e non sarà più supportato a partire dal 2 febbraio 2026.

Sia la scansione AWS nativa che quella di base Clair sono supportate in tutte le regioni elencate in AWS Servizi per regione, ad eccezione del fatto che Clair non è supportato per quelle aggiunte dopo settembre 2024. Per ulteriori informazioni, consulta Deprecazione di Clair.

Amazon ECR utilizza la severity di un CVE dalla fonte di distribuzione upstream, se disponibile. Altrimenti, viene usato il punteggio CVSS (Common Vulnerability Scoring System). Il punteggio CVSS può essere utilizzato per ottenere il livello di gravità della vulnerabilità NVD. Per ulteriori informazioni, consulta NVD Vulnerability Severity Ratings.

Entrambe le versioni di Amazon ECR Basic Scanning supportano filtri per specificare quali repository scansionare in modalità push. Tutti i repository che non corrispondono a un filtro Scan on Push sono impostati sulla frequenza di scansione manuale, il che significa che è necessario avviare la scansione manualmente. Un'immagine può essere scansionata una volta ogni 24 ore. Le 24 ore includono la scansione iniziale su push, se configurata, e le eventuali scansioni manuali. Con la scansione di base, è possibile scansionare fino a 100.000 immagini ogni 24 ore in un determinato registro. Il limite di 100.000 include sia la scansione iniziale con scansione push che quella manuale, sia su Clair che sulla versione migliorata della scansione di base.

I risultati delle ultime scansioni completate delle immagini possono essere recuperati per ogni immagine. Una volta completata la scansione dell'immagine, Amazon ECR invia un evento ad Amazon EventBridge. Per ulteriori informazioni, consulta Eventi Amazon ECR e EventBridge.

Deprecazione di Clair

Clair in Amazon ECR è obsoleto. Clair sarà ancora disponibile per l'uso fino al 2 febbraio 2026. Tuttavia, ti consigliamo vivamente di passare l'utilizzo di Clair alla scansione di base AWS nativa il prima possibile. Ecco cosa dovresti sapere su Clair Deprecation:

- Clair non sarà supportato in nuove regioni man mano che verranno aggiunte e non sarà più supportato in nessuna regione a partire dal 2 febbraio 2026.
- Non potrai eseguire alcuna scansione Clair a partire dal 2 febbraio 2026 e le scansioni eseguite prima di allora non saranno disponibili dopo tale data. Dovrai attivare una nuova scansione delle immagini per rigenerare i risultati della scansione dopo il passaggio alla nuova versione.

- Prima del 2 febbraio 2026 potete passare dalla scansione Clair alla scansione di base nativa e viceversa.
- Se hai già configurato Clair, passerai automaticamente alla scansione di base nativa a partire dal 2 febbraio 2026, se non l'hai fatto prima.

AWS La scansione di base nativa offre le seguenti funzionalità aggiuntive rispetto alla scansione Clair:

- Quando la scansione di base nativa analizza le risorse, recupera più di 50 feed di dati per generare risultati su vulnerabilità ed esposizioni comuni (). CVEs Esempi di queste fonti includono avvisi di sicurezza dei fornitori, feed di dati e feed di intelligence sulle minacce, nonché il National Vulnerability Database (NVD) e MITRE.
- La scansione di base nativa aggiorna i dati sulle vulnerabilità dai feed di origine almeno una volta al giorno.
- I risultati della scansione e il rilevamento delle vulnerabilità sono disponibili su un'ampia gamma di sistemi operativi più diffusi (vedi sotto).

Per passare alla scansione di base migliorata, vedere le istruzioni all'indirizzoPassaggio alla scansione di base migliorata per le immagini in Amazon ECR.

Supporto del sistema operativo per la scansione di base e la scansione di base migliorata

Come best practice di sicurezza e per una copertura continua, si consiglia di continuare a utilizzare le versioni supportate di un sistema operativo. In conformità alla politica dei fornitori, i sistemi operativi fuori produzione non vengono più aggiornati con patch e, in molti casi, non vengono più rilasciati nuovi avvisi di sicurezza relativi a tali sistemi. Inoltre, alcuni fornitori rimuovono gli avvisi e i rilevamenti di sicurezza esistenti dai propri feed quando un sistema operativo interessato raggiunge la fine del supporto standard. Dopo che una distribuzione perde il supporto del fornitore, Amazon ECR potrebbe non supportare più la scansione alla ricerca di vulnerabilità. Qualsiasi risultato generato da Amazon ECR per un sistema operativo fuori produzione deve essere utilizzato solo a scopo informativo. Di seguito sono elencati i sistemi operativi e le versioni attualmente supportati.

Sistema operativo	Versione	AWS base nativa	Clair di base
Alpine Linux (Alpine)	3.19	Sì	Sì
Linux Alpine (Alpine)	3.20	Sì	Sì
Alpine Linux (Alpine)	3.21	Sì	No
Alpine Linux (Alpine)	3.22	Sì	No
Alpine Linux (Alpine)	3.23	Sì	No
AlmaLinux	8	Sì	No
AlmaLinux	9	Sì	No
AlmaLinux	10	Sì	No
Amazon Linux (2AL2)	AL2	Sì	Sì
Amazon Linux 2023 (AL2023)	AL2023	Sì	Sì
Server Debian (Bullseye)	11	Sì	Sì
Server Debian (Bookworm)	12	Sì	Sì

Sistema operativo	Versione	AWS base nativa	Clair di base
Server Debian (Trixie)	13	Sì	No
Fedora	41	Sì	No
openSUSE Leap	15.6	Sì	No
Oracle Linux (Oracle)	8	Sì	Sì
Oracle Linux (Oracle)	9	Sì	Sì
Sistema operativo Photon	4	Sì	No
Sistema operativo Photon	5	Sì	No
Red Hat Enterprise Linux (RHEL)	8	Sì	Sì
Red Hat Enterprise Linux (RHEL)	9	Sì	Sì
Red Hat Enterprise Linux (RHEL)	10	Sì	No
Rocky Linux	8	Sì	No
Rocky Linux	9	Sì	No
SUSE Linux Enterprise Server (SLES)	15.6	Sì	No

Sistema operativo	Versione	AWS base nativa	Clair di base
Ubuntu (Xenial)	16.04 (SEM)	Sì	Sì
Ubuntu (Bionico)	18.04 (SECONDI)	Sì	Sì
Ubuntu (focale)	20.04 (LITRI)	Sì	Sì
Ubuntu (Jammy)	22.04 (LITRI)	Sì	Sì
Ubuntu (Noble Numbat)	24.04	Sì	No
Ubuntu (Oracular Oriole)	24.10	Sì	No

Configurazione della scansione di base per le immagini in Amazon ECR

Per impostazione predefinita, Amazon ECR attiva la scansione di base per tutti i registri privati. Di conseguenza, a meno che tu non abbia modificato le impostazioni di scansione nel tuo registro privato, non è necessario attivare la scansione di base. La scansione di base utilizza il progetto open source Clair.

È possibile utilizzare i seguenti passaggi per definire una o più scansioni sui filtri push.

Per attivare la scansione di base per il registro privato

1. Apri la console Amazon ECR su private-registry/repository https://console.aws.amazon.com/ecr/
2. Dalla barra di navigazione, scegli la regione in cui impostare la configurazione della scansione.
3. Nel pannello di navigazione, scegli Registro privato, Scansione.
4. Nella pagina Scanning configuration (Configurazione della scansione), per Scan type (Tipo di scansione) scegli Basic scanning (Scansione di base).
5. Per impostazione predefinita, tutti i repository sono impostati per la scansione Manual (Manuale). Puoi scegliere di configurare la scansione su push specificando Scan on push filters (Filtri di

scansione su push). È possibile impostare la scansione su push per tutti i repository o i singoli repository. Per ulteriori informazioni, consulta Filtri per scegliere quali repository scansionare in Amazon ECR.

> **Note**
>
> Se la scansione su push è abilitata per un repository, le scansioni vengono eseguite anche sulle immagini che vengono ripristinate dopo essere state archiviate. Nessuna vecchia scansione sarà disponibile dall'immagine ripristinata.

Passaggio alla scansione di base migliorata per le immagini in Amazon ECR

Amazon ECR offre funzionalità avanzate di scansione delle immagini dei contenitori attraverso una versione migliorata della scansione di base che utilizza la tecnologia AWS nativa. Questa funzionalità consente di identificare le vulnerabilità del software nelle immagini dei container. La procedura seguente consente di passare a questa versione migliorata della scansione di base se si utilizza una versione precedente della scansione di base che utilizza la `CLAIR` tecnologia.

> **Important**
>
> Per i nuovi utenti, i registri vengono configurati automaticamente per utilizzare la tecnologia di `AWS_NATIVE` scansione al momento della creazione. Non devi intraprendere alcuna azione. Amazon ECR non consiglia di tornare alla tecnologia di scansione precedente `CLAIR`, che è ormai obsoleta. Deprecazione di ClairPer ulteriori dettagli, consulta

Console di gestione AWS

Per attivare una scansione di base migliorata per il registro privato

1. Apri la console Amazon ECR su private-registry/repository https://console.aws.amazon.com/ecr/
2. Dalla barra di navigazione, scegli la regione in cui impostare la configurazione della scansione.

3. Nel pannello di navigazione, scegli Registro privato, Caratteristiche e impostazioni, Scansione.
4. Nella pagina di configurazione della scansione, scegli Opt in (consigliato) per selezionare la versione migliorata della scansione di base.
5. Per impostazione predefinita, tutti gli archivi sono impostati per la scansione manuale. Puoi scegliere di configurare la scansione su push specificando Scan on push filters (Filtri di scansione su push). È possibile impostare la scansione su push per tutti i repository o i singoli repository. Per ulteriori informazioni, consulta Filtri per scegliere quali repository scansionare in Amazon ECR.

AWS CLI

Amazon ECR ha abilitato la scansione di base per tutti i registri privati. Utilizza i seguenti comandi di seguito per visualizzare il tipo di scansione di base corrente e per modificare il tipo di scansione di base.

- Per recuperare la versione del tipo di scansione di base attualmente in uso.

```
aws ecr get-account-setting --name BASIC_SCAN_TYPE_VERSION
```

Il nome del parametro è un campo obbligatorio. Se non fornisci il nome, riceverai il seguente errore:

```
aws: error: the following arguments are required: --name
```

Per modificare la versione del tipo di scansione di base da `CLAIR` a`AWS_NATIVE`. Dopo aver modificato la versione del tipo di scansione di base da `CLAIR` a`AWS_NATIVE`, non è consigliabile tornare a`CLAIR`.

```
aws ecr put-account-setting --name BASIC_SCAN_TYPE_VERSION --value value
```

Scansione manuale di un'immagine per individuare le vulnerabilità del sistema operativo in Amazon ECR

Se i tuoi repository non sono configurati per la scansione in modalità push, puoi avviare manualmente le scansioni delle immagini. Un'immagine può essere scansionata una volta ogni 24 ore. Le 24 ore includono la scansione iniziale su push, se configurata, e le eventuali scansioni manuali.

Per la risoluzione dei problemi relativi ad alcuni problemi comuni durante la scansione delle immagini, consulta Risoluzione dei problemi di scansione delle immagini in Amazon ECR.

Console di gestione AWS

Attieniti alla seguente procedura per avviare una scansione manuale dell'immagine utilizzando la Console di gestione AWS.

1. Apri la console Amazon ECR su private-registry/repository https://console.aws.amazon.com/ecr/
2. Dalla barra di navigazione, scegliere la regione in cui creare il repository.
3. Nel riquadro di navigazione, selezionare Repositories (Repository).
4. Nella pagina Repositories (Repository) selezionare il repository contente l'immagine da scansionare.
5. Nella pagina Images (Immagini) selezionare l'immagine da scansionare, quindi scegliere Scan (Scansione).

AWS CLI

- start-image-scan (AWS CLI)

  Nell'esempio seguente viene utilizzato un tag di immagine.

```
aws ecr start-image-scan --repository-name name --image-id imageTag=tag_name --
region us-east-2
```

  Nell'esempio seguente viene utilizzato un digest di immagine.

```
aws ecr start-image-scan --repository-name name --image-id imageDigest=sha256_hash
 --region us-east-2
```

AWS Tools for Windows PowerShell

- ECRImageScanFindingOttieni AWS Tools for Windows PowerShell- ()

  Nell'esempio seguente viene utilizzato un tag di immagine.

  ```
  Start-ECRImageScan -RepositoryName name -ImageId_ImageTag tag_name -Region us-east-2 -Force
  ```

  Nell'esempio seguente viene utilizzato un digest di immagine.

  ```
  Start-ECRImageScan -RepositoryName name -ImageId_ImageDigest sha256_hash -Region us-east-2 -Force
  ```

Recupero dei risultati per le scansioni di base in Amazon ECR

È possibile recuperare i risultati della scansione per l'ultima scansione dell'immagine di base completata. Le vulnerabilità software rilevate sono elencate per gravità in base al database Common Vulnerabilities and Exposures (). CVEs

Per la risoluzione dei problemi relativi ad alcuni problemi comuni durante la scansione delle immagini, consulta Risoluzione dei problemi di scansione delle immagini in Amazon ECR.

Console di gestione AWS

Attieniti alla seguente procedura per recuperare i risultati della scansione delle immagini utilizzando la Console di gestione AWS.

Per recuperare i risultati della scansione delle immagini

1. Apri la console Amazon ECR su private-registry/repository https://console.aws.amazon.com/ecr/
2. Dalla barra di navigazione, scegliere la regione in cui creare il repository.
3. Nel riquadro di navigazione, selezionare Repositories (Repository).
4. Nella pagina Repositories (Repository), scegliere il repository che contiene l'immagine per cui recuperare i risultati della scansione.
5. Nella pagina Immagini, nella colonna Tag immagine, seleziona il tag dell'immagine per recuperare i risultati della scansione.

AWS CLI

Utilizzare il AWS CLI comando seguente per recuperare i risultati della scansione delle immagini utilizzando il. AWS CLI Puoi specificare un'immagine utilizzando `imageTag` o `imageDigest`, entrambe ottenibili utilizzando il comando CLI list-images.

- describe-image-scan-findings (AWS CLI)

  Nell'esempio seguente viene utilizzato un tag di immagine.

```
aws ecr describe-image-scan-findings --repository-name name --image-id
 imageTag=tag_name --region us-east-2
```

  Nell'esempio seguente viene utilizzato un digest di immagine.

```
aws ecr describe-image-scan-findings --repository-name name --image-id
 imageDigest=sha256_hash --region us-east-2
```

AWS Tools for Windows PowerShell

- Ottieni- ECRImage ScanFinding ()AWS Tools for Windows PowerShell

  Nell'esempio seguente viene utilizzato un tag di immagine.

```
Get-ECRImageScanFinding -RepositoryName name -ImageId_ImageTag tag_name -
Region us-east-2
```

  Nell'esempio seguente viene utilizzato un digest di immagine.

```
Get-ECRImageScanFinding -RepositoryName name -ImageId_ImageDigest sha256_hash -
Region us-east-2
```

Risoluzione dei problemi di scansione delle immagini in Amazon ECR

Di seguito sono riportati gli errori comuni relativi alla scansione delle immagini. Puoi visualizzare errori di questo tipo nella console Amazon ECR visualizzando i dettagli dell'immagine o tramite l'API o AWS CLI utilizzando l' `DescribeImageScanFindings`API.

UnsupportedImageError

È possibile ricevere un errore `UnsupportedImageError` nel tentativo di eseguire una scansione di base di un'immagine creata utilizzando un sistema operativo per il quale Amazon ECR non supporta la scansione di base delle immagini. Amazon ECR supporta la scansione delle vulnerabilità dei pacchetti per le versioni principali delle distribuzioni Amazon Linux, Amazon Linux 2, Debian, Ubuntu, CentOS, Oracle Linux, Alpine e RHEL Linux. Una volta che una distribuzione perde il supporto del suo fornitore, Amazon ECR potrebbe non supportare più la scansione per individuare eventuali vulnerabilità. Amazon ECR non supporta la scansione delle immagini create a partire dall'immagine scratch Docker.

> **Important**
>
> Quando si utilizza la scansione avanzata, Amazon Inspector supporta la scansione per tipi di supporto e sistemi operativi specifici. Per un elenco completo, consulta la sezione Tipi di supporto e sistemi operativi supportati nella Guida per l'utente di Amazon Inspector.

Viene restituito un livello di gravità `UNDEFINED`

È possibile che venga visualizzato un rilevamento di scansione con un livello di gravità `UNDEFINED`. Di seguito sono riportate le cause comuni per questo:

- Alla vulnerabilità non è stata assegnata una priorità dall'origine CVE.
- Alla vulnerabilità è stata assegnata una priorità che Amazon ECR non riconosce.

Per determinare la gravità e la descrizione di una vulnerabilità, è possibile visualizzare il CVE direttamente dall'origine.

Informazioni sullo stato della scansione **SCAN_ELIGIBILITY_EXPIRED**

Quando la scansione avanzata con Amazon Inspector è abilitata per il tuo registro privato e stai visualizzando le vulnerabilità della scansione, è possibile che venga visualizzato uno stato di scansione di `SCAN_ELIGIBILITY_EXPIRED`. Le cause più comuni sono le seguenti.

- Quando inizialmente attivi la scansione avanzata per il tuo registro privato, Amazon Inspector riconosce solo le immagini inviate ad Amazon ECR negli ultimi 30 giorni, in base al timestamp di invio delle immagini. Le immagini più vecchie avranno lo stato di scansione `SCAN_ELIGIBILITY_EXPIRED`. Se desideri che queste immagini vengano scansionate da Amazon Inspector, devi reinserirle nel tuo repository.
- Se la Durata della nuova scansione ECR viene modificata nella console di Amazon Inspector e questo tempo è trascorso, lo stato di scansione dell'immagine viene modificato in `inactive` con un codice motivo `expired` e tutti i resultati associati all'immagine sono programmati per essere chiusi. Il risultato è che la console Amazon ECR elenca lo stato di scansione come `SCAN_ELIGIBILITY_EXPIRED`.

Sincronizzazione di un registro upstream con un registro privato Amazon ECR

Utilizzando le regole pull through cache, puoi sincronizzare il contenuto di un registro upstream con il tuo registro privato Amazon ECR.

Amazon ECR attualmente supporta la creazione di regole pull through cache per i seguenti registri upstream:

- Amazon ECR Public, registro delle immagini dei container Kubernetes e Quay (non richiede l'autenticazione)
- Docker Hub, Microsoft Azure Container Registry, GitHub Container Registry e GitLab Container Registry (richiede l'autenticazione con Gestione dei segreti AWS segreto)
- Amazon ECR (richiede l'autenticazione con ruolo AWS IAM)

Per GitLab Container Registry, Amazon ECR supporta la funzionalità pull through cache solo con GitLab l'offerta Software as a Service (SaaS). Per ulteriori informazioni sull'utilizzo GitLab dell'offerta SaaS, consulta GitLab .com.

Per i registri upstream che richiedono l'autenticazione con segreti (come Docker Hub), è necessario archiviare le credenziali in un luogo segreto. Gestione dei segreti AWS Puoi utilizzare la console Amazon ECR per creare segreti di Secrets Manager per ogni registro upstream autenticato. Per ulteriori informazioni sulla creazione di un segreto di Secrets Manager utilizzando la console Secrets Manager, vedereArchiviazione segreta delle credenziali del repository originale Gestione dei segreti AWS.

Per Amazon ECR, devi creare un ruolo IAM se i registri Amazon ECR upstream e downstream appartengono a un account diverso. AWS Per ulteriori informazioni sulla creazione di un ruolo IAM, consulta Le politiche IAM sono necessarie per il pull through della cache da ECR a ECR su più account.

Dopo aver creato una regola pull through cache per il registro upstream, estrai un'immagine da quel registro upstream utilizzando l'URI del tuo registro privato Amazon ECR. Amazon ECR, quindi, crea un repository e memorizza l'immagine nella cache nel tuo registro privato. Per le richieste pull successive dell'immagine memorizzata nella cache con un determinato tag, Amazon ECR verifica la presenza di una nuova versione dell'immagine con quel tag specifico e tenta di aggiornare l'immagine nel registro privato almeno una volta ogni 24 ore.

Modelli per la creazione di repository

Amazon ECR ha aggiunto il supporto per i modelli di creazione di repository, che ti consente di specificare le configurazioni iniziali per i nuovi repository creati da Amazon ECR per tuo conto utilizzando le regole pull through cache. Ogni modello contiene un prefisso dello spazio dei nomi del repository che viene utilizzato per abbinare i nuovi repository a un modello specifico. I modelli possono specificare la configurazione per tutte le impostazioni del repository, comprese le policy di accesso basate sulle risorse, l'immutabilità dei tag, la crittografia e le policy del ciclo di vita. Le impostazioni in un modello di creazione di repository vengono applicate solo durante la creazione del repository e non hanno alcun effetto sui repository esistenti o sui repository creati con altri metodi. Per ulteriori informazioni, consulta Modelli per controllare gli archivi creati durante un'azione di pull through cache, di creazione in modalità push o di replica.

Considerazioni sull'utilizzo delle regole pull through cache

Considera quanto segue quando utilizzi le regole pull through cache di Amazon ECR.

- La creazione di regole di cache pull-through non è supportata nelle seguenti regioni.
  - Cina (Pechino) (`cn-north-1`)
  - Cina (Ningxia) (`cn-northwest-1`)
  - AWS GovCloud (Stati Uniti orientali) () `us-gov-east-1`
  - AWS GovCloud (Stati Uniti occidentali) () `us-gov-west-1`
- AWS Lambda non supporta l'estrazione di immagini di container da Amazon ECR utilizzando una regola pull through cache.
- Quando si estraggono immagini utilizzando la cache pull-through, gli endpoint del servizio FIPS di Amazon ECR non sono supportati la prima volta che viene estratta un'immagine. Tuttavia, l'utilizzo degli endpoint del servizio FIPS Amazon ECR funziona sui pull successivi.
- Quando un'immagine memorizzata nella cache viene estratta dall'URI del registro privato di Amazon ECR, i recuperi delle immagini vengono avviati dagli indirizzi IP. AWS Ciò garantisce che l'estrazione dell'immagine non tenga conto delle quote relative alla frequenza di estrazione implementate dal registro upstream.
- Quando un'immagine memorizzata nella cache viene estratta attraverso l'URI del registro privato di Amazon ECR, quest'ultimo controlla il repository upstream almeno una volta ogni 24 ore per verificare se la versione dell'immagine memorizzata nella cache è la più recente. Se è presente un'immagine più recente nel registro upstream, Amazon ECR tenta di aggiornare l'immagine

memorizzata nella cache. Questo timer è basato sull'ultima estrazione dell'immagine memorizzata nella cache.

- Se per qualsiasi motivo Amazon ECR non è in grado di aggiornare l'immagine dal registro upstream e l'immagine viene estratta, l'ultima immagine memorizzata nella cache verrà comunque estratta.
- Quando crei il segreto di Secrets Manager contenente le credenziali del registro upstream, il nome del segreto deve utilizzare il prefisso `ecr-pullthroughcache/`. Il segreto, inoltre, deve trovarsi nello stesso account e nella stessa regione in cui è stata creata la regola di cache pull-through.
- Quando un'immagine multi-architettura viene estratta utilizzando una regola di cache pull-through, l'elenco manifesto e ogni immagine a cui fa riferimento nell'elenco manifesto vengono estratti nel repository Amazon ECR. Se si desidera estrarre solo un'architettura specifica, è possibile estrarre l'immagine utilizzando il digest dell'immagine o il tag associato all'architettura anziché il tag associato all'elenco manifesto.
- Amazon ECR utilizza un ruolo IAM collegato ai servizi che fornisce le autorizzazioni necessarie ad Amazon ECR per creare per tuo conto il repository, recuperare il valore segreto di Secrets Manager per l'autenticazione e inviare l'immagine memorizzata nella cache. Il ruolo IAM collegato ai servizi viene creato automaticamente quando viene creata una regola di cache pull-through. Per ulteriori informazioni, consulta Ruolo collegato ai servizi Amazon ECR per la cache pull-through.
- Per impostazione predefinita, il principale IAM che estrae l'immagine memorizzata nella cache dispone delle autorizzazioni concesse tramite la policy IAM. È possibile utilizzare la policy delle autorizzazioni del registro privato di Amazon ECR per definire ulteriormente le autorizzazioni di un'entità IAM. Per ulteriori informazioni, consulta Utilizzo delle autorizzazioni di registro.
- I repository Amazon ECR creati utilizzando il flusso di lavoro della cache pull-through vengono trattati come qualsiasi altro repository Amazon ECR. Sono supportate tutte le funzionalità del repository, come la replica e la scansione delle immagini.
- Quando Amazon ECR crea un nuovo repository per tuo conto utilizzando un'azione di cache pull-through, al repository vengono applicate le impostazioni predefinite indicate di seguito, a meno che non esista un modello di creazione repository corrispondente. Puoi utilizzare un modello di creazione repository per definire le impostazioni applicate ai repository creati da Amazon ECR per tuo conto. Per ulteriori informazioni, consulta Modelli per controllare gli archivi creati durante un'azione di pull through cache, di creazione in modalità push o di replica.
  - Immutabilità dei tag: l'immutabilità dei tag specifica se i tag di immagine possono essere sovrascritti. Per impostazione predefinita, i tag di immagine sono modificabili (possono essere sovrascritti). È possibile modificare il comportamento dei tag configurando i filtri di esclusione dei

tag nella casella di testo Esclusione tag mutabile quando è selezionato Mutabile o nella casella di testo di esclusione dei tag Immutabile quando è selezionato Immutabile.

- Crittografia: viene utilizzata la crittografia predefinita. `AES256`
- Autorizzazioni del repository: omesse, non viene applicata alcuna politica di autorizzazione del repository.
- Criteri relativi al ciclo di vita: omessi, non vengono applicati criteri relativi al ciclo di vita.
- Tag delle risorse: omessi, non viene applicato alcun tag di risorsa.

- L'attivazione dell'immutabilità dei tag di immagine per i repository utilizzando una regola di cache pull-through impedirà ad Amazon ECR di aggiornare le immagini utilizzando lo stesso tag.
- Quando un'immagine viene estratta utilizzando la regola pull through cache per la prima volta, potrebbe essere necessario un percorso verso Internet. In alcune circostanze è necessario un percorso verso Internet, quindi è meglio impostare un percorso per evitare errori. Pertanto, se hai configurato Amazon ECR per utilizzare un'interfaccia che AWS PrivateLink utilizza un endpoint VPC, devi assicurarti che il primo pull abbia un percorso verso Internet. Un modo per farlo consiste nel creare una sottorete pubblica nello stesso VPC, con un gateway Internet, e quindi indirizzare tutto il traffico in uscita verso Internet dalla sottorete privata alla sottorete pubblica. I successivi recuperi di immagini che utilizzano la regola pull through cache non richiedono questa operazione. Per ulteriori informazioni, consulta Opzioni di routing di esempio nella Guida per l'utente di Amazon Virtual Private Cloud.

Autorizzazioni IAM necessarie per sincronizzare un registro upstream con un registro privato Amazon ECR

Oltre alle autorizzazioni dell'API Amazon ECR necessarie per l'autenticazione in un registro privato e per l'invio e l'estrazione di immagini, sono necessarie le seguenti autorizzazioni aggiuntive per utilizzare efficacemente le regole di cache pull-through.

- `ecr:CreatePullThroughCacheRule` – Concede l'autorizzazione per creare una nuova regola di cache pull-through. Questa autorizzazione deve essere concessa tramite una policy IAM basata sull'identità.
- `ecr:BatchImportUpstreamImage`— Concede l'autorizzazione per recuperare l'immagine esterna e importarla nel registro privato. Questa autorizzazione può essere concessa utilizzando la policy delle autorizzazioni del registro privato, una policy IAM basata sull'identità o utilizzando la policy delle autorizzazioni del repository basate sulle risorse. Per ulteriori informazioni sull'uso delle autorizzazioni del repository, consulta Politiche di archivio privato in Amazon ECR.

- `ecr:CreateRepository` – Concede l'autorizzazione per creare un repository in un registro privato. Questa autorizzazione è necessaria se il repository che memorizza le immagini memorizzate nella cache non è già esistente. Questa autorizzazione può essere concessa da una policy IAM basata sull'identità o dalla policy delle autorizzazioni del registro privato.

Utilizzo delle autorizzazioni di registro

Le autorizzazioni del registro privato di Amazon ECR possono essere utilizzate per definire le autorizzazioni delle singole entità IAM per utilizzare la cache pull-through. Se un'entità IAM dispone di più autorizzazioni concesse da una policy IAM di quelle concesse dalla policy delle autorizzazioni del registro, la policy IAM ha la precedenza. Ad esempio, se a un utente sono state concesse autorizzazioni `ecr:*`, non occorrono altre autorizzazioni a livello di registro.

Per creare una policy delle autorizzazioni del registro privato (Console di gestione AWS)

1. Apri la console Amazon ECR all'indirizzo https://console.aws.amazon.com/ecr/.
2. Dalla barra di navigazione, scegli la regione in cui configurare l'istruzione delle autorizzazioni del registro privato.
3. Nel pannello di navigazione, seleziona Private registry (Registro privato), Registry permissions (Autorizzazioni di registro).
4. Alla pagina Registry permissions (Autorizzazioni di registro), scegli Generate statement (Genera istruzione).
5. Per ogni istruzione delle policy di autorizzazione della cache pull-through che si desidera creare, procedi come segue.

    a. Per Policy type (Tipo di policy), scegli Pull through cache policy (Policy della cache pull-through).

    b. Per Statement id (ID istruzione), inserisci un nome per la policy dell'istruzione della cache pull-through.

    c. Per Entità IAM, specifica gli utenti, i gruppi o i ruoli da includere nella policy.

    d. Per Repository namespace (Spazio dei nomi del repository), seleziona la regola della cache pull-through a cui associare la policy.

    e. Per Repository names (Nomi dei repository), specifica il nome di base del repository per cui applicare la regola. Ad esempio, se si desidera specificare il repository Amazon Linux su Amazon ECR Public, il nome del repository sarà `amazonlinux`.

Per creare una policy delle autorizzazioni del registro privato (AWS CLI)

Usa il seguente AWS CLI comando per specificare le autorizzazioni del registro privato utilizzando. AWS CLI

1. Crea un file locale denominato `ptc-registry-policy.json` con il contenuto della policy del registro. L'esempio seguente concede l'autorizzazione `ecr-pull-through-cache-user` per creare un repository ed eseguire il pull di un'immagine da Amazon ECR Public, che è l'origine upstream associata alla regola cache pull-through creata precedentemente.

   JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Sid": "PullThroughCacheFromReadOnlyRole",
      "Effect": "Allow",
      "Principal": {
        "AWS": "arn:aws:iam::111122223333:user/ecr-pull-through-cache-user"
      },
      "Action": [
        "ecr:CreateRepository",
        "ecr:BatchImportUpstreamImage"
      ],
      "Resource": "arn:aws:ecr:us-east-1:111122223333:repository/ecr-public/
*"
    }
  ]
}
```

> **Important**
>
> L'autorizzazione `ecr-CreateRepository` è necessaria solo se il repository che memorizza le immagini memorizzate nella cache non è già esistente. Ad esempio, se l'operazione di creazione del repository e le operazioni di estrazione dell'immagine vengono eseguite da principali IAM separati come un amministratore e uno sviluppatore.

2. Utilizzare il put-registry-policycomando per impostare la politica del registro.

```
aws ecr put-registry-policy \
     --policy-text file://ptc-registry.policy.json
```

Fasi successive

Quando sei pronto a cominciare a utilizzare le regole di cache pull-through, attieniti ai passaggi seguenti.

- Crea una regola di cache pull-through. Per ulteriori informazioni, consulta Creazione di una regola pull through cache in Amazon ECR.
- Crea un modello di creazione repository. Un modello di creazione repository ti consente di gestire la definizione delle impostazioni da utilizzare per i nuovi repository creati da Amazon ECR per tuo conto nel corso di un'operazione di estrazione di cache pull-through. Per ulteriori informazioni, consulta Modelli per controllare gli archivi creati durante un'azione di pull through cache, di creazione in modalità push o di replica.

Impostazione delle autorizzazioni per più account da ECR a ECR PTC

La funzionalità pull through cache da Amazon ECR ad Amazon ECR (da ECR a ECR) consente la sincronizzazione automatica delle immagini tra regioni, AWS account o entrambi. Con ECR to ECR PTC, puoi inviare immagini al registro Amazon ECR principale e configurare una regola pull through cache per memorizzare nella cache le immagini nei registri Amazon ECR downstream.

Le politiche IAM sono necessarie per il pull through della cache da ECR a ECR su più account

Per memorizzare nella cache le immagini tra i registri Amazon ECR su AWS account diversi, crea un ruolo IAM nell'account downstream e configura le policy in questa sezione per fornire le seguenti autorizzazioni:

- Amazon ECR necessita delle autorizzazioni per estrarre immagini dal registro Amazon ECR upstream per tuo conto. Puoi concedere queste autorizzazioni creando un ruolo IAM e poi specificandolo nella regola pull through cache.

- Il proprietario del registro a monte deve inoltre concedere al proprietario del registro di cache le autorizzazioni necessarie per inserire le immagini nelle politiche delle risorse.

Policy

- Creazione di un ruolo IAM per definire le autorizzazioni di pull through cache
- Creazione di una politica di fiducia per il ruolo IAM
- Creazione di una politica delle risorse nel registro Amazon ECR a monte

Creazione di un ruolo IAM per definire le autorizzazioni di pull through cache

L'esempio seguente mostra una politica di autorizzazioni che concede a un ruolo IAM l'autorizzazione a estrarre immagini dal registro Amazon ECR a monte per tuo conto. Quando Amazon ECR assume il ruolo, riceve le autorizzazioni specificate in questa politica.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "VisualEditor0",
            "Effect": "Allow",
            "Action": [
                "ecr:GetDownloadUrlForLayer",
                "ecr:GetAuthorizationToken",
                "ecr:BatchImportUpstreamImage",
                "ecr:BatchGetImage",
                "ecr:GetImageCopyStatus",
                "ecr:InitiateLayerUpload",
                "ecr:UploadLayerPart",
                "ecr:CompleteLayerUpload",
                "ecr:PutImage"
            ],
            "Resource": "*"
        }
    ]
}
```

Creazione di una politica di fiducia per il ruolo IAM

L'esempio seguente mostra una policy di fiducia che identifica il pull through cache di Amazon ECR come principale del AWS servizio che può assumere il ruolo.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "Service": "pullthroughcache.ecr.amazonaws.com"
            },
            "Action": "sts:AssumeRole"
        }
    ]
}
```

Creazione di una politica delle risorse nel registro Amazon ECR a monte

Il proprietario del registro Amazon ECR a monte deve inoltre aggiungere una politica di registro o una politica di repository per concedere al proprietario del registro a valle le autorizzazioni necessarie per eseguire le seguenti azioni.

```
{
    "Effect": "Allow",
    "Principal": {
        "AWS": "arn:aws:iam::444455556666:root"
    },
    "Action": [
        "ecr:BatchGetImage",
        "ecr:GetDownloadUrlForLayer",
        "ecr:BatchImportUpstreamImage",
        "ecr:GetImageCopyStatus"
    ],
    "Resource": "arn:aws:ecr:region:111122223333:repository/*"
}
```

Creazione di una regola pull through cache in Amazon ECR

Per ogni registro upstream contenente immagini che desideri memorizzare nella cache nel tuo registro privato Amazon ECR, devi creare una regola pull through cache.

Per i registri upstream che richiedono l'autenticazione con segreti, è necessario memorizzare le credenziali in un segreto di Secrets Manager. È possibile utilizzare un segreto esistente o crearne uno nuovo. Puoi creare il segreto di Secrets Manager nella console Amazon ECR o nella console Secrets Manager. Per creare un segreto di Secrets Manager utilizzando la console Secrets Manager anziché la console Amazon ECR, consultaArchiviazione segreta delle credenziali del repository originale Gestione dei segreti AWS.

Prerequisiti

- Verifica di disporre delle autorizzazioni IAM appropriate per creare regole pull through cache. Per informazioni, consulta Autorizzazioni IAM necessarie per sincronizzare un registro upstream con un registro privato Amazon ECR.
- Per i registri upstream che richiedono l'autenticazione con segreti: se desideri utilizzare un segreto esistente, verifica che il segreto di Secrets Manager soddisfi i seguenti requisiti:
  - Il nome del segreto inizia con. `ecr-pullthroughcache/` Visualizza Console di gestione AWS solo i segreti di Secrets Manager con il `ecr-pullthroughcache/` prefisso.
  - L'account e la regione in cui si trova il segreto devono corrispondere all'account e alla regione in cui si trova la regola pull through cache.

Per creare una regola di cache pull-through (Console di gestione AWS)

Nei passaggi seguenti viene illustrato come creare una regola di cache pull-through e un segreto di Secrets Manager tramite la console di Amazon ECR. Per creare un segreto utilizzando la console Secrets Manager, vedereArchiviazione segreta delle credenziali del repository originale Gestione dei segreti AWS.

Per Amazon ECR Public, il registro container Kubernetes o Quay

1. Apri la console Amazon ECR all'indirizzo https://console.aws.amazon.com/ecr/.
2. Dalla barra di navigazione, scegli la regione in cui configurare le impostazioni del registro privato.
3. Nel pannello di navigazione, seleziona Private registry (Registro privato), Pull through cache (Cache pull-through).

4. Nella pagina Pull through cache configuration (Configurazione della cache pull through), scegli Add rule (Aggiungi regola).
5. Nella pagina Passaggio 1: specifica un'origine, per Registro seleziona Amazon ECR Public, Kubernetes o Quay dall'elenco dei registri upstream, quindi seleziona Avanti.
6. Nella pagina Passaggio 2: specifica una destinazione, per Prefisso del repository Amazon ECR specifica il prefisso dello spazio dei nomi del repository da utilizzare per la memorizzazione nella cache delle immagini estratte dal registro pubblico di origine, quindi seleziona Avanti. Per impostazione predefinita, viene popolato uno spazio dei nomi ma è possibile specificare anche uno spazio dei nomi personalizzato.
7. Nella pagina Passaggio 3: rivedi e crea, esamina la configurazione della regola di cache pull-through, quindi seleziona Crea.
8. Ripeti il passaggio precedente per ogni cache pull-through da creare. Le regole della cache pull-through vengono create separatamente per ciascuna regione.

Per Docker Hub

1. Apri la console Amazon ECR all'indirizzo https://console.aws.amazon.com/ecr/.
2. Dalla barra di navigazione, scegli la regione in cui configurare le impostazioni del registro privato.
3. Nel pannello di navigazione, seleziona Private registry (Registro privato), Pull through cache (Cache pull-through).
4. Nella pagina Pull through cache configuration (Configurazione della cache pull through), scegli Add rule (Aggiungi regola).
5. Nel Passaggio 1: specifica un'origine, per Registro seleziona Docker Hub, Avanti.
6. Nella pagina Passaggio 2: configura l'autenticazione, per Credenziali Upstream devi archiviare le credenziali di autenticazione per Docker Hub in un segreto di Gestione dei segreti AWS . Puoi specificare un segreto esistente o utilizzare la console di Amazon ECR per crearne uno nuovo.

    a. Per utilizzare un segreto esistente, scegli Usa un AWS segreto esistente. Per Nome del segreto, utilizza il menu a discesa per selezionare il segreto esistente, quindi seleziona Avanti.

    > **Note**
    >
    > Visualizza Console di gestione AWS solo i segreti di Secrets Manager i cui nomi utilizzano il `ecr-pullthroughcache/` prefisso. Il segreto, inoltre, deve trovarsi

nello stesso account e nella stessa regione in cui è stata creata la regola di cache pull-through.

b. Per creare un nuovo segreto, seleziona Crea un segreto di AWS , effettua le seguenti operazioni e seleziona Avanti.

   i. Per Nome del segreto specifica un nome descrittivo per il segreto. I nomi dei segreti devono contenere un numero di caratteri Unicode compreso tra 1 e 512.

   ii. Per l'e-mail di Docker Hub, specifica la tua e-mail Docker Hub.

   iii. Per Token di accesso Docker Hub specifica il tuo token di accesso Docker Hub. Per ulteriori informazioni sulla creazione di un token di accesso Docker Hub, consulta Creazione e gestione di token di accesso nella documentazione di Docker.

7. Nella pagina Passaggio 3: specifica una destinazione, per Prefisso del repository Amazon ECR specifica lo spazio dei nomi del repository da utilizzare nel corso della memorizzazione nella cache delle immagini estratte dal registro pubblico di origine, quindi seleziona Avanti.

   Per impostazione predefinita, viene popolato uno spazio dei nomi ma è possibile specificare anche uno spazio dei nomi personalizzato.

8. Nella pagina Passaggio 4: rivedi e crea, esamina la configurazione della regola di cache pull-through, quindi seleziona Crea.

9. Ripeti il passaggio precedente per ogni cache pull-through da creare. Le regole della cache pull-through vengono create separatamente per ciascuna regione.

Per GitHub Container Registry

1. Apri la console Amazon ECR all'indirizzo https://console.aws.amazon.com/ecr/.

2. Dalla barra di navigazione, scegli la regione in cui configurare le impostazioni del registro privato.

3. Nel pannello di navigazione, seleziona Private registry (Registro privato), Pull through cache (Cache pull-through).

4. Nella pagina Pull through cache configuration (Configurazione della cache pull through), scegli Add rule (Aggiungi regola).

5. Nella Fase 1: Specificate una pagina sorgente, per Registro, scegliete GitHub Container Registry, Next.

6. Nella pagina Passaggio 2: Configurazione dell'autenticazione, per le credenziali Upstream, è necessario archiviare le credenziali di autenticazione per GitHub Container Registry in un luogo

segreto. Gestione dei segreti AWS Puoi specificare un segreto esistente o utilizzare la console di Amazon ECR per crearne uno nuovo.

a. Per utilizzare un segreto esistente, scegli Usa un segreto esistente. AWS Per Nome del segreto, utilizza il menu a discesa per selezionare il segreto esistente, quindi seleziona Avanti.

> **Note**
>
> Visualizza Console di gestione AWS solo i segreti di Secrets Manager i cui nomi utilizzano il `ecr-pullthroughcache/` prefisso. Il segreto, inoltre, deve trovarsi nello stesso account e nella stessa regione in cui è stata creata la regola di cache pull-through.

b. Per creare un nuovo segreto, seleziona Crea un segreto di AWS , effettua le seguenti operazioni e seleziona Avanti.

   i. Per Nome del segreto specifica un nome descrittivo per il segreto. I nomi dei segreti devono contenere un numero di caratteri Unicode compreso tra 1 e 512.

   ii. Per il nome utente del GitHub Container Registry, specifica il tuo nome utente del GitHub Container Registry.

   iii. Per il token di accesso al GitHub Container Registry, specifica il token di accesso al GitHub Container Registry. Per ulteriori informazioni sulla creazione di un token di GitHub accesso, consulta Gestire i token di accesso personali nella GitHub documentazione.

7. Nella pagina Passaggio 3: specifica una destinazione, per Prefisso del repository Amazon ECR specifica lo spazio dei nomi del repository da utilizzare nel corso della memorizzazione nella cache delle immagini estratte dal registro pubblico di origine, quindi seleziona Avanti.

   Per impostazione predefinita, viene popolato uno spazio dei nomi ma è possibile specificare anche uno spazio dei nomi personalizzato.

8. Nella pagina Passaggio 4: rivedi e crea, esamina la configurazione della regola di cache pull-through, quindi seleziona Crea.

9. Ripeti il passaggio precedente per ogni cache pull-through da creare. Le regole della cache pull-through vengono create separatamente per ciascuna regione.

Per Microsoft Azure Container Registry

1. Apri la console Amazon ECR all'indirizzo https://console.aws.amazon.com/ecr/.
2. Dalla barra di navigazione, scegli la regione in cui configurare le impostazioni del registro privato.
3. Nel pannello di navigazione, seleziona Private registry (Registro privato), Pull through cache (Cache pull-through).
4. Nella pagina Pull through cache configuration (Configurazione della cache pull through), scegli Add rule (Aggiungi regola).
5. Nella pagina Passaggio 1: specifica un'origine, procedi come segue.

   a. Per Registro seleziona Microsoft Azure Container Registry

   b. Per URL registro di origine, specifica il nome del registro dei container di Microsoft Azure, quindi seleziona Avanti.

   > **Important**
   >
   > Poiché il suffisso `.azurecr.io` viene compilato per tuo conto, devi specificare solo il prefisso.

6. Nella pagina Passaggio 2: configura l'autenticazione, per Credenziali Upstream devi archiviare le credenziali di autenticazione per Microsoft Azure Container Registry in un segreto di Gestione dei segreti AWS . Puoi specificare un segreto esistente o utilizzare la console di Amazon ECR per crearne uno nuovo.

   a. Per utilizzare un segreto esistente, scegli Usa un AWS segreto esistente. Per Nome del segreto, utilizza il menu a discesa per selezionare il segreto esistente, quindi seleziona Avanti.

   > **Note**
   >
   > Visualizza Console di gestione AWS solo i segreti di Secrets Manager i cui nomi utilizzano il `ecr-pullthroughcache/` prefisso. Il segreto, inoltre, deve trovarsi nello stesso account e nella stessa regione in cui è stata creata la regola di cache pull-through.

   b. Per creare un nuovo segreto, seleziona Crea un segreto di AWS , effettua le seguenti operazioni e seleziona Avanti.

i. Per Nome del segreto specifica un nome descrittivo per il segreto. I nomi dei segreti devono contenere un numero di caratteri Unicode compreso tra 1 e 512.

ii. Per Nome utente di Microsoft Azure Container Registry, specifica il tuo nome utente per Microsoft Azure Container Registry.

iii. Per Token di accesso di Microsoft Azure Container Registry, specifica il tuo token di accesso per Microsoft Azure Container Registry. Per ulteriori informazioni sulla creazione di un token di accesso per Microsoft Azure Container Registry, consulta Creazione token - portale nella documentazione di Microsoft Azure.

7. Nella pagina Passaggio 3: specifica una destinazione, per Prefisso del repository Amazon ECR specifica lo spazio dei nomi del repository da utilizzare nel corso della memorizzazione nella cache delle immagini estratte dal registro pubblico di origine, quindi seleziona Avanti.

   Per impostazione predefinita, viene popolato uno spazio dei nomi ma è possibile specificare anche uno spazio dei nomi personalizzato.

8. Nella pagina Passaggio 4: rivedi e crea, esamina la configurazione della regola di cache pull-through, quindi seleziona Crea.

9. Ripeti il passaggio precedente per ogni cache pull-through da creare. Le regole della cache pull-through vengono create separatamente per ciascuna regione.

Per GitLab Container Registry

1. Apri la console Amazon ECR all'indirizzo https://console.aws.amazon.com/ecr/.

2. Dalla barra di navigazione, scegli la regione in cui configurare le impostazioni del registro privato.

3. Nel pannello di navigazione, seleziona Private registry (Registro privato), Pull through cache (Cache pull-through).

4. Nella pagina Pull through cache configuration (Configurazione della cache pull through), scegli Add rule (Aggiungi regola).

5. Nella Fase 1: Specificate una pagina sorgente, per Registro, scegliete GitLab Container Registry, Next.

6. Nella pagina Passaggio 2: Configurazione dell'autenticazione, per le credenziali Upstream, è necessario archiviare le credenziali di autenticazione per GitLab Container Registry in un luogo segreto. Gestione dei segreti AWS Puoi specificare un segreto esistente o utilizzare la console di Amazon ECR per crearne uno nuovo.

a. Per utilizzare un segreto esistente, scegli Usa un segreto esistente. AWS Per Nome del segreto, utilizza il menu a discesa per selezionare il segreto esistente, quindi seleziona Avanti. Per ulteriori informazioni sulla creazione di un segreto di Secrets Manager utilizzando la console di Secrets Manager, consulta Archiviazione segreta delle credenziali del repository originale Gestione dei segreti AWS.

> **Note**
>
> Visualizza Console di gestione AWS solo i segreti di Secrets Manager i cui nomi utilizzano il `ecr-pullthroughcache/` prefisso. Il segreto, inoltre, deve trovarsi nello stesso account e nella stessa regione in cui è stata creata la regola di cache pull-through.

b. Per creare un nuovo segreto, seleziona Crea un segreto di AWS , effettua le seguenti operazioni e seleziona Avanti.

   i. Per Nome del segreto specifica un nome descrittivo per il segreto. I nomi dei segreti devono contenere un numero di caratteri Unicode compreso tra 1 e 512.

   ii. Per il nome utente del GitLab Container Registry, specifica il tuo nome utente del GitLab Container Registry.

   iii. Per il token di accesso al GitLab Container Registry, specifica il token di accesso al GitLab Container Registry. Per ulteriori informazioni sulla creazione di un token di accesso al GitLab Container Registry, consulta Token di accesso personali, token di accesso di gruppo o token di accesso al progetto nella documentazione. GitLab

7. Nella pagina Passaggio 3: specifica una destinazione, per Prefisso del repository Amazon ECR specifica lo spazio dei nomi del repository da utilizzare nel corso della memorizzazione nella cache delle immagini estratte dal registro pubblico di origine, quindi seleziona Avanti.

   Per impostazione predefinita, viene popolato uno spazio dei nomi ma è possibile specificare anche uno spazio dei nomi personalizzato.

8. Nella pagina Passaggio 4: rivedi e crea, esamina la configurazione della regola di cache pull-through, quindi seleziona Crea.

9. Ripeti il passaggio precedente per ogni cache pull-through da creare. Le regole della cache pull-through vengono create separatamente per ciascuna regione.

Per il registro privato di Amazon ECR all'interno del tuo account AWS

1. Apri la console Amazon ECR all'indirizzo https://console.aws.amazon.com/ecr/.
2. Dalla barra di navigazione, scegli la regione in cui desideri configurare le impostazioni del registro privato.
3. Nel pannello di navigazione, seleziona Private registry (Registro privato), Pull through cache (Cache pull-through).
4. Nella pagina Pull through cache configuration (Configurazione della cache pull through), scegli Add rule (Aggiungi regola).
5. Nella Fase 1: Specificate la pagina upstream, per Registro, selezionate Amazon ECR Private e Questo account. Per Regione, seleziona la regione per il registro Amazon ECR upstream, quindi scegli Avanti.
6. Nella pagina Step 2: Specificare gli spazi dei nomi, per Cache namespace, scegli se creare repository pull through cache con un prefisso specifico o nessun prefisso. Se si seleziona Un prefisso specifico, è necessario specificare un nome di prefisso da utilizzare come parte dello spazio dei nomi per la memorizzazione nella cache delle immagini dal registro upstream.
7. Per lo spazio dei nomi Upstream, scegli se estrarre da un prefisso specifico presente nel registro upstream. Se non si seleziona alcun prefisso, è possibile estrarlo da qualsiasi repository nel registro upstream. Specificate il prefisso del repository upstream, se richiesto, quindi scegliete Avanti.

   > **Note**
   >
   > Per ulteriori informazioni sulla personalizzazione della cache e dei namespace upstream, consulta. Personalizzazione dei prefissi del repository da ECR a ECR pull through cache

8. Nella pagina Passaggio 3: rivedi e crea, esamina la configurazione della regola di cache pull-through, quindi seleziona Crea.
9. Ripeti questi passaggi per ogni pull through cache che desideri creare. Le regole della cache pull-through vengono create separatamente per ciascuna regione.

Per il registro privato Amazon ECR da un altro account AWS

1. Apri la console Amazon ECR all'indirizzo https://console.aws.amazon.com/ecr/.
2. Dalla barra di navigazione, scegli la regione in cui configurare le impostazioni del registro privato.

3. Nel pannello di navigazione, seleziona Private registry (Registro privato), Pull through cache (Cache pull-through).
4. Nella pagina Pull through cache configuration (Configurazione della cache pull through), scegli Add rule (Aggiungi regola).
5. Nella Fase 1: Specificate la pagina upstream, per Registry, scegliete Amazon ECR Private and Cross account. Per Regione, seleziona la regione per il registro Amazon ECR upstream. Per Account, specifica l'ID AWS dell'account per il registro Amazon ECR a monte, quindi scegli Avanti.
6. Nella pagina Step 2: Specificare le autorizzazioni, per il ruolo IAM, seleziona un ruolo da utilizzare per l'accesso alla cache tra più account, quindi scegli Crea.

> **Note**
>
> Assicurati di selezionare il ruolo IAM che utilizza le autorizzazioni create in. Le politiche IAM sono necessarie per il pull through della cache da ECR a ECR su più account

7. Nella pagina Step 3: Specificare gli spazi dei nomi, per Cache namespace, scegli se creare repository pull through cache con un prefisso specifico o nessun prefisso. Se si seleziona Un prefisso specifico, è necessario specificare un nome di prefisso da utilizzare come parte dello spazio dei nomi per la memorizzazione nella cache delle immagini dal registro upstream.
8. Per lo spazio dei nomi Upstream, scegli se estrarre da un prefisso specifico presente nel registro upstream. Se non si seleziona alcun prefisso, è possibile estrarlo da qualsiasi repository nel registro upstream. Specificate il prefisso del repository upstream, se richiesto, quindi scegliete Avanti.

> **Note**
>
> Per ulteriori informazioni sulla personalizzazione della cache e dei namespace upstream, consulta. Personalizzazione dei prefissi del repository da ECR a ECR pull through cache

9. Nella pagina Passaggio 4: rivedi e crea, esamina la configurazione della regola di cache pull-through, quindi seleziona Crea.
10. Ripeti questi passaggi per ogni pull through cache che desideri creare. Le regole della cache pull-through vengono create separatamente per ciascuna regione.

Per creare una regola di cache pull-through (AWS CLI)

Usa il AWS CLI comando create-pull-through-cache-rule per creare una regola pull through cache per un registro privato Amazon ECR. Per i registri upstream che richiedono l'autenticazione con segreti, è necessario memorizzare le credenziali in un segreto di Secrets Manager. Per creare un segreto utilizzando la console Secrets Manager, vedereArchiviazione segreta delle credenziali del repository originale Gestione dei segreti AWS.

Gli esempi seguenti sono forniti per ogni registro upstream supportato.

Per Amazon ECR Public

L'esempio seguente crea una regola di cache pull-through per il registro pubblico di Amazon ECR. Specifica un prefisso di `ecr-public`, che fa sì che ciascun repository creato utilizzando la regola della cache pull-through abbia lo schema di denominazione di `ecr-public/`*`upstream-repository-name`*.

```
aws ecr create-pull-through-cache-rule \
     --ecr-repository-prefix ecr-public \
     --upstream-registry-url public.ecr.aws \
     --region us-east-2
```

Per Kubernetes Container Registry

L'esempio seguente crea una regola di cache pull through per il registro pubblico Kubernetes. Specifica un prefisso di `kubernetes`, che fa sì che ciascun repository creato utilizzando la regola della cache pull-through abbia lo schema di denominazione di `kubernetes/`*`upstream-repository-name`*.

```
aws ecr create-pull-through-cache-rule \
     --ecr-repository-prefix kubernetes \
     --upstream-registry-url registry.k8s.io \
     --region us-east-2
```

Per Quay

L'esempio seguente crea una regola di cache pull-through per il registro pubblico Quay. Specifica un prefisso di `quay`, che fa sì che ciascun repository creato utilizzando la regola della cache pull-through abbia lo schema di denominazione di `quay/`*`upstream-repository-name`*.

```
aws ecr create-pull-through-cache-rule \
```

```
    --ecr-repository-prefix quay \
    --upstream-registry-url quay.io \
    --region us-east-2
```

Per Docker Hub

L'esempio seguente crea una regola di cache pull-through per il registro Docker Hub. Specifica un prefisso di `docker-hub`, che fa sì che ciascun repository creato utilizzando la regola della cache pull-through abbia lo schema di denominazione di `docker-hub/`*`upstream-repository-name`*. Devi specificare nella sua interezza il nome della risorsa Amazon (ARN) del segreto contenente le credenziali di Docker Hub.

```
aws ecr create-pull-through-cache-rule \
    --ecr-repository-prefix docker-hub \
    --upstream-registry-url registry-1.docker.io \
    --credential-arn arn:aws:secretsmanager:us-east-2:111122223333:secret:ecr-
pullthroughcache/example1234 \
    --region us-east-2
```

Per Container Registry GitHub

L'esempio seguente crea una regola pull through cache per il GitHub Container Registry. Specifica un prefisso di `github`, che fa sì che ciascun repository creato utilizzando la regola della cache pull-through abbia lo schema di denominazione di `github/`*`upstream-repository-name`*. È necessario specificare l'Amazon Resource Name (ARN) completo del segreto contenente le credenziali del GitHub Container Registry.

```
aws ecr create-pull-through-cache-rule \
    --ecr-repository-prefix github \
    --upstream-registry-url ghcr.io \
    --credential-arn arn:aws:secretsmanager:us-east-2:111122223333:secret:ecr-
pullthroughcache/example1234 \
    --region us-east-2
```

Per Microsoft Azure Container Registry

L'esempio seguente crea una regola pull through cache per il Microsoft Azure Container Registry. Specifica un prefisso di `azure`, che fa sì che ciascun repository creato utilizzando la regola della cache pull-through abbia lo schema di denominazione di `azure/`*`upstream-repository-name`*. Devi specificare nella sua interezza il nome della risorsa Amazon (ARN) del segreto contenente le credenziali di Microsoft Azure Container Registry.

```
aws ecr create-pull-through-cache-rule \
     --ecr-repository-prefix azure \
     --upstream-registry-url myregistry.azurecr.io \
     --credential-arn arn:aws:secretsmanager:us-east-2:111122223333:secret:ecr-
pullthroughcache/example1234 \
     --region us-east-2
```

Per GitLab Container Registry

L'esempio seguente crea una regola pull through cache per il GitLab Container Registry. Specifica un prefisso di `gitlab`, che fa sì che ciascun repository creato utilizzando la regola della cache pull-through abbia lo schema di denominazione di `gitlab/`*`upstream-repository-name`*. È necessario specificare l'Amazon Resource Name (ARN) completo del segreto contenente le credenziali del GitLab Container Registry.

```
aws ecr create-pull-through-cache-rule \
     --ecr-repository-prefix gitlab \
     --upstream-registry-url registry.gitlab.com \
     --credential-arn arn:aws:secretsmanager:us-east-2:111122223333:secret:ecr-
pullthroughcache/example1234 \
     --region us-east-2
```

Per il registro privato di Amazon ECR all'interno del tuo account AWS

L'esempio seguente crea una regola pull through cache per il registro privato Amazon ECR per Cross-region all'interno dello stesso AWS account. Specifica un prefisso di `ecr`, che fa sì che ciascun repository creato utilizzando la regola della cache pull-through abbia lo schema di denominazione di `ecr/`*`upstream-repository-name`*.

```
aws ecr create-pull-through-cache-rule \
     --ecr-repository-prefix ecr \
     --upstream-registry-url aws_account_id.dkr.ecr.region.amazonaws.com \
     --region us-east-2
```

Per il registro privato Amazon ECR da un altro account AWS

L'esempio seguente crea una regola pull through cache per il registro privato Amazon ECR per Cross-region all'interno dello stesso AWS account. Specifica un prefisso di `ecr`, che fa sì che ciascun repository creato utilizzando la regola della cache pull-through abbia lo schema di denominazione di `ecr/`*`upstream-repository-name`*. È necessario specificare l'Amazon Resource Name (ARN)

completo del ruolo IAM con le autorizzazioni create in. Creazione di una regola pull through cache in Amazon ECR

```
aws ecr create-pull-through-cache-rule \
 --ecr-repository-prefix ecr \
 --upstream-registry-url aws_account_id.dkr.ecr.region.amazonaws.com \
 --custom-role-arn arn:aws:iam::aws_account_id:role/example-role \
 --region us-east-2
```

Fasi successive

Dopo aver creato le regole pull through cache, i passaggi successivi sono i seguenti:

- Crea un modello di creazione repository. Un modello di creazione repository ti consente di gestire la definizione delle impostazioni da utilizzare per i nuovi repository creati da Amazon ECR per tuo conto nel corso di un'operazione di estrazione di cache pull-through. Per ulteriori informazioni, consulta Modelli per controllare gli archivi creati durante un'azione di pull through cache, di creazione in modalità push o di replica.
- Convalida le regole di cache pull-through. Durante la convalida di una regola cache di pull-through, Amazon ECR stabilisce una connessione di rete con il registro upstream, verifica di poter accedere al segreto di Secrets Manager contenente le credenziali del registro upstream e controlla che l'autenticazione sia avvenuta correttamente. Per ulteriori informazioni, consulta Convalida delle regole pull through cache in Amazon ECR.
- Inizia a utilizzare le regole di cache pull-through. Per ulteriori informazioni, consulta Estrazione di un'immagine con una regola pull through cache in Amazon ECR.

Convalida delle regole pull through cache in Amazon ECR

Dopo aver creato una regola pull through cache, per i registri upstream che richiedono l'autenticazione è possibile verificare che la regola funzioni correttamente. Durante la convalida di una regola pull through cache, Amazon ECR stabilisce una connessione di rete con il registro upstream, verifica di poter accedere al segreto di Secrets Manager contenente le credenziali per il registro upstream e verifica che l'autenticazione sia avvenuta correttamente.

Prima di iniziare a utilizzare le regole pull through cache, verifica di disporre delle autorizzazioni IAM appropriate. Per ulteriori informazioni, consulta Autorizzazioni IAM necessarie per sincronizzare un registro upstream con un registro privato Amazon ECR.

Creazione di una regola di cache pull-through (Console di gestione AWS)

Nei seguenti passaggi viene illustrato come convalidare una regola di cache pull-through tramite la console di Amazon ECR.

1. Apri la console Amazon ECR all'indirizzo https://console.aws.amazon.com/ecr/.
2. Dalla barra di navigazione seleziona la regione in cui si trova la regola di cache pull-through da convalidare.
3. Nel pannello di navigazione, seleziona Private registry (Registro privato), Pull through cache (Cache pull-through).
4. Nella pagina Configurazione di cache pull-through, seleziona la regola cache di pull-through da convalidare. Utilizza, quindi, il menu a discesa Azioni e seleziona Visualizza dettagli.
5. Nella pagina dei dettagli della regola cache di pull-through, utilizza il menu a discesa Azioni e seleziona Verifica l'autenticazione. Amazon ECR mostrerà un banner con il risultato.
6. Ripeti questi passaggi per ogni regola di cache pull-through da convalidare.

Creazione di una regola di cache pull-through (AWS CLI)

Il AWS CLI comando validate-pull-through-cache-rule viene utilizzato per convalidare una regola pull through cache per un registro privato Amazon ECR. Nell'esempio seguente viene utilizzato il prefisso dello spazio dei nomi `ecr-public`. Sostituisci tale valore con il valore del prefisso per la convalida della regola di cache pull-through.

```
aws ecr validate-pull-through-cache-rule \
     --ecr-repository-prefix ecr-public \
     --region us-east-2
```

Nella risposta, il parametro `isValid` indica se la convalida è riuscita. `true` indica che Amazon ECR è riuscito a raggiungere il registro upstream e che l'autenticazione è riuscita. `false` indica che si è verificato un problema e che la convalida non è riuscita. Il parametro `failure` indica la causa.

Estrazione di un'immagine con una regola pull through cache in Amazon ECR

Gli esempi seguenti mostrano la sintassi del comando da utilizzare quando estrai un'immagine utilizzando una regola di cache pull-through. In caso di errore durante l'estrazione di un'immagine

upstream utilizzando una regola di cache pull-through, consulta Risoluzione dei problemi di pull through cache in Amazon ECR per gli errori più comuni e come risolverli.

Prima di iniziare a utilizzare le regole pull through cache, verifica di disporre delle autorizzazioni IAM appropriate. Per ulteriori informazioni, consulta Autorizzazioni IAM necessarie per sincronizzare un registro upstream con un registro privato Amazon ECR.

> **Note**
>
> I seguenti esempi utilizzano i valori di namespace del repository Amazon ECR predefiniti utilizzati. Console di gestione AWS Assicurati di utilizzare l'URI del repository privato di Amazon ECR configurato.

Per Amazon ECR Public

```
docker pull aws_account_id.dkr.ecr.region.amazonaws.com/ecr-public/repository_name/
image_name:tag
```

Registro dei container Kubernetes

```
docker pull aws_account_id.dkr.ecr.region.amazonaws.com/kubernetes/repository_name/
image_name:tag
```

Quay

```
docker pull aws_account_id.dkr.ecr.region.amazonaws.com/quay/repository_name/
image_name:tag
```

Docker Hub

Per le immagini ufficiali di Docker Hub:

```
docker pull aws_account_id.dkr.ecr.region.amazonaws.com/docker-hub/
library/image_name:tag
```

> **Note**
>
> Per le immagini ufficiali di Docker Hub, il prefisso `/library` deve essere incluso. Per tutti gli altri repository di Docker Hub, è necessario omettere il prefisso `/library`.

Per tutte le altre immagini di Docker Hub:

```
docker pull aws_account_id.dkr.ecr.region.amazonaws.com/docker-hub/repository_name/
image_name:tag
```

GitHub Registro dei contenitori

```
docker pull aws_account_id.dkr.ecr.region.amazonaws.com/github/repository_name/
image_name:tag
```

Microsoft Azure Container Registry

```
docker pull aws_account_id.dkr.ecr.region.amazonaws.com/azure/repository_name/
image_name:tag
```

GitLab Registro dei contenitori

```
docker pull aws_account_id.dkr.ecr.region.amazonaws.com/gitlab/repository_name/
image_name:tag
```

Archiviazione segreta delle credenziali del repository originale Gestione dei segreti AWS

Quando crei una regola di cache pull-through per un repository upstream che richiede l'autenticazione, devi memorizzare le credenziali in un segreto di Secrets Manager. L'utilizzo di un segreto di Secrets Manager potrebbe essere soggetto a un costo. Per ulteriori informazioni, consultare Prezzi di Gestione dei segreti AWS.

Le procedure seguenti illustrano come creare un segreto di Secrets Manager per ogni repository upstream supportato. Anziché creare il segreto utilizzando la console di Secrets Manager, per creare

il segreto puoi utilizzare anche il flusso di lavoro di creazione delle regole di cache pull-through nella console di Amazon ECR. Per ulteriori informazioni, consulta Creazione di una regola pull through cache in Amazon ECR.

Docker Hub

Creazione di un segreto di Secrets Manager per le credenziali di Docker Hub (Console di gestione AWS)

1. Apri la console Secrets Manager all'indirizzo https://console.aws.amazon.com/secretsmanager/.
2. Scegli Archivia un nuovo segreto.
3. Nella pagina Scegli tipo di segreto, procedi come segue.
   a. Per Secret type (Tipo di segreto), scegli Other type of secret (Altro tipo di segreto).
   b. In Coppie chiave/valore, crea due righe per le tue credenziali di Docker Hub. Puoi archiviare fino a 65536 byte nel segreto.
      i. Per la prima key/value coppia, specifica `username` come chiave e il tuo nome utente Docker Hub come valore.
      ii. Per la seconda key/value coppia, specifica `accessToken` come chiave e il token di accesso a Docker Hub come valore. Per ulteriori informazioni sulla creazione di un token di accesso di Docker Hub, consulta Creazione e gestione di token di accesso nella documentazione di Docker.
   c. Per Chiave di crittografia, mantieni il valore predefinito AWS KMS key aws/secretsmanager, quindi seleziona Avanti. L'utilizzo di questa chiave non prevede costi aggiuntivi. Per ulteriori informazioni, consulta Crittografia e decrittografia del segreto in Secrets Manager nella Guida per l'utente di Gestione dei segreti AWS .

   > **Important**
   >
   > Devi utilizzare la chiave di crittografia predefinita `aws/secretsmanager` per crittografare il tuo segreto. Per tale scopo, Amazon ECR non supporta l'utilizzo di una chiave gestita dal cliente (CMK).

4. Nella pagina Configura il segreto, procedi come segue.

a. Inserisci un Secret name (Nome del segreto) e una Description (Descrizione) descrittivi. I nomi dei segreti devono contenere un numero di caratteri Unicode compreso tra 1 e 512 e il prefisso `ecr-pullthroughcache/`.

> **Important**
>
> Amazon ECR visualizza Console di gestione AWS solo i segreti di Secrets Manager con nomi che utilizzano il `ecr-pullthroughcache/` prefisso.

b. (Facoltativo) Nella sezione Tags (Tag) aggiungere tag al segreto. Per le strategie di applicazione di tag, consulta Applicazione di tag ai segreti di Secrets Manager nella Guida per l'utente di Gestione dei segreti AWS . Non archiviare informazioni sensibili nei tag perché non sono crittografate.

c. (Facoltativo) In Permessi delle risorse, per aggiungere una policy delle risorse al tuo segreto, scegli Modifica delle autorizzazioni. Per ulteriori informazioni, consulta Collega una policy di autorizzazioni a un segreto di Secrets Manager nella Guida per l'utente di Gestione dei segreti AWS .

d. (Facoltativo) In Replica segreto, per replicare il tuo segreto su un altro Regione AWS, scegli Replica segreto. Puoi replicare il tuo segreto immediatamente o tornare e replicarlo in un secondo momento. Per ulteriori informazioni, consulta Replica di un segreto per altre regioni nella Guida per l'utente di Gestione dei segreti AWS .

e. Scegli Next (Successivo).

5. (Facoltativo) Nella pagina Configure rotation (Configura la rotazione), puoi attivare la rotazione automatica. Puoi anche disattivare la rotazione e poi riattivarla in un secondo momento. Per ulteriori informazioni, consulta Rotazione di segreti su Secrets Manager nella Guida per l'utente di Gestione dei segreti AWS . Scegli Next (Successivo).

6. Nella pagina Review (Revisione), rivedi i dettagli dei segreti e quindi scegli Store (Archivia).

   Secrets Manager ritorna all'elenco dei segreti. Se il segreto nuovo non viene visualizzato, scegli il pulsante aggiorna.

GitHub Container Registry

Per creare un segreto di Secrets Manager per le credenziali del GitHub Container Registry ()Console di gestione AWS

1. Apri la console Secrets Manager all'indirizzo https://console.aws.amazon.com/secretsmanager/.
2. Scegli Archivia un nuovo segreto.
3. Nella pagina Scegli tipo di segreto, procedi come segue.
   a. Per Secret type (Tipo di segreto), scegli Other type of secret (Altro tipo di segreto).
   b. Nelle coppie chiave/valore, create due righe per le vostre GitHub credenziali. Puoi archiviare fino a 65536 byte nel segreto.
      i. Per la prima key/value coppia, specificate `username` come chiave e il vostro GitHub nome utente come valore.
      ii. Per la seconda key/value coppia, specificate `accessToken` come chiave e il token di GitHub accesso come valore. Per ulteriori informazioni sulla creazione di un token di GitHub accesso, consulta Gestire i token di accesso personali nella GitHub documentazione.
   c. Per Chiave di crittografia, mantieni il valore predefinito AWS KMS key aws/secretsmanager, quindi seleziona Avanti. L'utilizzo di questa chiave non prevede costi aggiuntivi. Per ulteriori informazioni, consulta Crittografia e decrittografia del segreto in Secrets Manager nella Guida per l'utente di Gestione dei segreti AWS .

   > **Important**
   >
   > Devi utilizzare la chiave di crittografia predefinita `aws/secretsmanager` per crittografare il tuo segreto. Per tale scopo, Amazon ECR non supporta l'utilizzo di una chiave gestita dal cliente (CMK).

4. Nella pagina Configure secret (Configura il segreto), effettua le seguenti operazioni:
   a. Inserisci un Secret name (Nome del segreto) e una Description (Descrizione) descrittivi. I nomi dei segreti devono contenere un numero di caratteri Unicode compreso tra 1 e 512 e il prefisso `ecr-pullthroughcache/`.

> **Important**
>
> Amazon ECR visualizza Console di gestione AWS solo i segreti di Secrets Manager con nomi che utilizzano il `ecr-pullthroughcache/` prefisso.

b. (Facoltativo) Nella sezione Tags (Tag) aggiungere tag al segreto. Per le strategie di applicazione di tag, consulta Applicazione di tag ai segreti di Secrets Manager nella Guida per l'utente di Gestione dei segreti AWS . Non archiviare informazioni sensibili nei tag perché non sono crittografate.

c. (Facoltativo) In Permessi delle risorse, per aggiungere una policy delle risorse al tuo segreto, scegli Modifica delle autorizzazioni. Per ulteriori informazioni, consulta Collega una policy di autorizzazioni a un segreto di Secrets Manager nella Guida per l'utente di Gestione dei segreti AWS .

d. (Facoltativo) In Replica segreto, per replicare il tuo segreto su un altro Regione AWS, scegli Replica segreto. Puoi replicare il tuo segreto immediatamente o tornare e replicarlo in un secondo momento. Per ulteriori informazioni, consulta Replica di un segreto per altre regioni nella Guida per l'utente di Gestione dei segreti AWS .

e. Scegli Next (Successivo).

5. (Facoltativo) Nella pagina Configure rotation (Configura la rotazione), puoi attivare la rotazione automatica. Puoi anche disattivare la rotazione e poi riattivarla in un secondo momento. Per ulteriori informazioni, consulta Rotazione di segreti su Secrets Manager nella Guida per l'utente di Gestione dei segreti AWS . Scegli Next (Successivo).

6. Nella pagina Review (Revisione), rivedi i dettagli dei segreti e quindi scegli Store (Archivia).

   Secrets Manager ritorna all'elenco dei segreti. Se il segreto nuovo non viene visualizzato, scegli il pulsante aggiorna.

Microsoft Azure Container Registry

Creazione di un segreto di Secrets Manager per le credenziali di Microsoft Azure Container Registry (Console di gestione AWS)

1. Apri la console Secrets Manager all'indirizzo https://console.aws.amazon.com/secretsmanager/.
2. Scegli Archivia un nuovo segreto.

3. Nella pagina Scegli tipo di segreto, procedi come segue.

   a. Per Secret type (Tipo di segreto), scegli Other type of secret (Altro tipo di segreto).

   b. Per Coppie chiave/valore crea due righe per le tue credenziali di Microsoft Azure. Puoi archiviare fino a 65536 byte nel segreto.

      i. Per la prima key/value coppia, specificare `username` come chiave e il nome utente di Microsoft Azure Container Registry come valore.

      ii. Per la seconda key/value coppia, specificare `accessToken` come chiave e il token di accesso a Microsoft Azure Container Registry come valore. Per ulteriori informazioni sulla creazione di un token di accesso per Microsoft Azure, consulta Creazione token - portale nella documentazione di Microsoft Azure.

   c. Per Chiave di crittografia, mantieni il valore predefinito AWS KMS key aws/secretsmanager, quindi seleziona Avanti. L'utilizzo di questa chiave non prevede costi aggiuntivi. Per ulteriori informazioni, consulta Crittografia e decrittografia del segreto in Secrets Manager nella Guida per l'utente di Gestione dei segreti AWS .

   > **Important**
   >
   > Devi utilizzare la chiave di crittografia predefinita `aws/secretsmanager` per crittografare il tuo segreto. Per tale scopo, Amazon ECR non supporta l'utilizzo di una chiave gestita dal cliente (CMK).

4. Nella pagina Configure secret (Configura il segreto), effettua le seguenti operazioni:

   a. Inserisci un Secret name (Nome del segreto) e una Description (Descrizione) descrittivi. I nomi dei segreti devono contenere un numero di caratteri Unicode compreso tra 1 e 512 e il prefisso `ecr-pullthroughcache/`.

   > **Important**
   >
   > Amazon ECR visualizza Console di gestione AWS solo i segreti di Secrets Manager con nomi che utilizzano il `ecr-pullthroughcache/` prefisso.

   b. (Facoltativo) Nella sezione Tags (Tag) aggiungere tag al segreto. Per le strategie di applicazione di tag, consulta Applicazione di tag ai segreti di Secrets Manager nella Guida per l'utente di Gestione dei segreti AWS . Non archiviare informazioni sensibili nei tag perché non sono crittografate.

c. (Facoltativo) In Permessi delle risorse, per aggiungere una policy delle risorse al tuo segreto, scegli Modifica delle autorizzazioni. Per ulteriori informazioni, consulta Collega una policy di autorizzazioni a un segreto di Secrets Manager nella Guida per l'utente di Gestione dei segreti AWS .

d. (Facoltativo) In Replica segreto, per replicare il tuo segreto su un altro Regione AWS, scegli Replica segreto. Puoi replicare il tuo segreto immediatamente o tornare e replicarlo in un secondo momento. Per ulteriori informazioni, consulta Replica di un segreto per altre regioni nella Guida per l'utente di Gestione dei segreti AWS .

e. Scegli Next (Successivo).

5. (Facoltativo) Nella pagina Configure rotation (Configura la rotazione), puoi attivare la rotazione automatica. Puoi anche disattivare la rotazione e poi riattivarla in un secondo momento. Per ulteriori informazioni, consulta Rotazione di segreti su Secrets Manager nella Guida per l'utente di Gestione dei segreti AWS . Scegli Next (Successivo).

6. Nella pagina Review (Revisione), rivedi i dettagli dei segreti e quindi scegli Store (Archivia).

   Secrets Manager ritorna all'elenco dei segreti. Se il segreto nuovo non viene visualizzato, scegli il pulsante aggiorna.

GitLab Container Registry

Per creare un segreto di Secrets Manager per le credenziali del GitLab Container Registry ()Console di gestione AWS

1. Apri la console Secrets Manager all'indirizzo https://console.aws.amazon.com/secretsmanager/.
2. Scegli Archivia un nuovo segreto.
3. Nella pagina Scegli tipo di segreto, procedi come segue.

   a. Per Secret type (Tipo di segreto), scegli Other type of secret (Altro tipo di segreto).

   b. Nelle coppie chiave/valore, create due righe per le vostre GitLab credenziali. Puoi archiviare fino a 65536 byte nel segreto.

      i. Per la prima key/value coppia, specificate `username` come chiave e il nome utente del GitLab Container Registry come valore.

      ii. Per la seconda key/value coppia, specificate `accessToken` come chiave e il token di accesso al GitLab Container Registry come valore. Per ulteriori informazioni sulla

creazione di un token di accesso al GitLab Container Registry, consulta Token di accesso personali, token di accesso di gruppo o token di accesso al progetto nella documentazione. GitLab

c. Per Chiave di crittografia, mantieni il valore predefinito AWS KMS key aws/secretsmanager, quindi seleziona Avanti. L'utilizzo di questa chiave non prevede costi aggiuntivi. Per ulteriori informazioni, consulta Crittografia e decrittografia del segreto in Secrets Manager nella Guida per l'utente di Gestione dei segreti AWS .

> **Important**
>
> Devi utilizzare la chiave di crittografia predefinita `aws/secretsmanager` per crittografare il tuo segreto. Per tale scopo, Amazon ECR non supporta l'utilizzo di una chiave gestita dal cliente (CMK).

4. Nella pagina Configure secret (Configura il segreto), effettua le seguenti operazioni:

a. Inserisci un Secret name (Nome del segreto) e una Description (Descrizione) descrittivi. I nomi dei segreti devono contenere un numero di caratteri Unicode compreso tra 1 e 512 e il prefisso `ecr-pullthroughcache/`.

> **Important**
>
> Amazon ECR visualizza Console di gestione AWS solo i segreti di Secrets Manager con nomi che utilizzano il `ecr-pullthroughcache/` prefisso.

b. (Facoltativo) Nella sezione Tags (Tag) aggiungere tag al segreto. Per le strategie di applicazione di tag, consulta Applicazione di tag ai segreti di Secrets Manager nella Guida per l'utente di Gestione dei segreti AWS . Non archiviare informazioni sensibili nei tag perché non sono crittografate.

c. (Facoltativo) In Permessi delle risorse, per aggiungere una policy delle risorse al tuo segreto, scegli Modifica delle autorizzazioni. Per ulteriori informazioni, consulta Collega una policy di autorizzazioni a un segreto di Secrets Manager nella Guida per l'utente di Gestione dei segreti AWS .

d. (Facoltativo) In Replica segreto, per replicare il tuo segreto su un altro Regione AWS, scegli Replica segreto. Puoi replicare il tuo segreto immediatamente o tornare e replicarlo in un secondo momento. Per ulteriori informazioni, consulta Replica di un segreto per altre regioni nella Guida per l'utente di Gestione dei segreti AWS .

e. Scegli Next (Successivo).

5. (Facoltativo) Nella pagina Configure rotation (Configura la rotazione), puoi attivare la rotazione automatica. Puoi anche disattivare la rotazione e poi riattivarla in un secondo momento. Per ulteriori informazioni, consulta Rotazione di segreti su Secrets Manager nella Guida per l'utente di Gestione dei segreti AWS . Scegli Next (Successivo).
6. Nella pagina Review (Revisione), rivedi i dettagli dei segreti e quindi scegli Store (Archivia).

   Secrets Manager ritorna all'elenco dei segreti. Se il segreto nuovo non viene visualizzato, scegli il pulsante aggiorna.

Personalizzazione dei prefissi del repository da ECR a ECR pull through cache

Le regole pull through cache supportano sia il prefisso del repository ecr che il prefisso del repository upstream. Il prefisso del repository ecr è il prefisso dello spazio dei nomi del repository nel registro di cache Amazon ECR associato alla regola. Tutti i repository che utilizzano questo prefisso diventano repository abilitati alla memorizzazione nella cache per il registro upstream definito nella regola. Ad esempio, il prefisso di si applica a tutti i repository che iniziano con. `prod` `prod/` Per applicare un modello a tutti i repository del registro a cui non è associata una regola pull through cache, usa `ROOT` come prefisso.

> **Important**
>
> Si presuppone che `/` venga sempre applicato alla fine del prefisso. Se specifichi `ecr-public` come prefisso, Amazon ECR lo considera come `ecr-public/`.

Il prefisso del repository upstream corrisponde al nome del repository upstream. Per impostazione predefinita, è impostato su, il che consente la corrispondenza con `ROOT` qualsiasi repository upstream. Puoi impostare il prefisso del repository upstream solo quando il prefisso del repository Amazon ECR non ha un valore. `ROOT`

La tabella seguente mostra la mappatura tra i nomi dei repository di cache e i nomi dei repository upstream in base alle relative configurazioni di prefisso nelle regole pull through cache.

Spazio dei nomi della cache	Spazio dei nomi upstream	Relazione di mappatura (cache repository → repository upstream)

ecr-public	ROOT (impostazione predefinita)	`ecr-public/my-app/image1` → `my-app/image1`
		
		`ecr-public/my-app/image2` → `my-app/image2`

RADICE	RADICE	`my-app/image1` → `my-app/image1`
squadra-a	squadra-a	`team-a/myapp/image1` → `team-a/myapp/image1`
la mia app	app upstream	`my-app/image1` → `upstream-app/image1`

Risoluzione dei problemi di pull through cache in Amazon ECR

Di seguito sono riportati gli errori più comuni che potresti ricevere durante l'estrazione di un'immagine upstream utilizzando una regola di cache pull-through.

Il repository non esiste

Un errore che indica che il repository non esiste è spesso causato dal repository non esistente nel registro privato Amazon ECR o dall'autorizzazione `ecr:CreateRepository` non concessa al principale IAM che estrae l'immagine a monte. Per risolvere questo errore, è necessario verificare che l'URI del repository nel comando pull sia corretto, che le autorizzazioni IAM richieste siano concesse al principale IAM che estrae l'immagine upstream o che il repository per l'immagine upstream da inviare venga creato nel registro privato di Amazon ECR prima di estrarre l'immagine upstream. Per ulteriori informazioni sulle autorizzazioni IAM richieste, consulta Autorizzazioni IAM necessarie per sincronizzare un registro upstream con un registro privato Amazon ECR

Di seguito è illustrato un esempio di questo errore.

```
Error response from daemon: repository 111122223333.dkr.ecr.us-east-1.amazonaws.com/
ecr-public/amazonlinux/amazonlinux not found: name unknown: The repository with
 name 'ecr-public/amazonlinux/amazonlinux' does not exist in the registry with id
 '111122223333'
```

L'immagine richiesta non è stata trovata

Un errore che indica che l'immagine non può essere trovata è spesso causato dall'immagine non esistente nel registro upstream o dall'autorizzazione `ecr:BatchImportUpstreamImage` non concessa al principale IAM che estrae l'immagine upstream ma il repository è già stato creato nel registro privato di Amazon ECR. Per risolvere questo errore, è necessario verificare che l'immagine upstream e il nome del tag immagine siano corretti e che esistano e che le autorizzazioni IAM richieste siano concesse al principale IAM che estrea l'immagine upstream. Per ulteriori informazioni sulle autorizzazioni IAM richieste, consulta Autorizzazioni IAM necessarie per sincronizzare un registro upstream con un registro privato Amazon ECR.

Di seguito è illustrato un esempio di questo errore.

```
Error response from daemon: manifest for 111122223333.dkr.ecr.us-
east-1.amazonaws.com/ecr-public/amazonlinux/amazonlinux:latest not found: manifest
 unknown: Requested image not found
```

403 Proibito quando si estrae da un repository Docker Hub

Quando estrai da un repository Docker Hub etichettato come Docker Official Image, devi includere `/library/` nell'URI utilizzato. Ad esempio, *`aws_account_id`*`.dkr.ecr.`*`region`*`.amazonaws.com/docker-hub/`**`library/`***`image_name:tag`*. Se ometti l'opzione `/library/` for Docker Hub Official images, verrà restituito un `403 Forbidden` errore quando tenti di estrarre l'immagine utilizzando una regola pull through cache. Per ulteriori informazioni, consulta Estrazione di un'immagine con una regola pull through cache in Amazon ECR.

Di seguito è illustrato un esempio di questo errore.

```
Error response from daemon: failed to resolve reference "111122223333.dkr.ecr.us-
west-2.amazonaws.com/docker-hub/amazonlinux:2023": pulling from host
 111122223333.dkr.ecr.us-west-2.amazonaws.com failed with status code [manifests
 2023]: 403 Forbidden
```

Private image replication in Amazon ECR

Puoi configurare il tuo registro privato Amazon ECR in modo da supportare la replica dei tuoi repository. Amazon ECR supporta sia la replica tra regioni che tra account. Affinché si verifichi la replica tra account, l'account di destinazione deve configurare una policy di autorizzazione del registro per consentire la replica dal registro. Per ulteriori informazioni, consulta Autorizzazioni di registro private in Amazon ECR.

Argomenti

Requisiti delle politiche di replica tra account

Affinché la replica ECR tra account funzioni correttamente, è necessario comprendere quale account necessita di quali politiche configurate. Questa sezione chiarisce i requisiti delle policy sia per gli account di origine che per quelli di destinazione.

Panoramica della configurazione delle politiche

La replica ECR tra account richiede la configurazione delle policy solo sull'account di destinazione. L'account di origine non richiede alcun repository o policy di registro speciali.

- Account di origine: configura le regole di replica nelle impostazioni del registro. Non sono richieste politiche aggiuntive sui repository di origine.
- Account di destinazione: configura una politica di autorizzazione del registro per consentire all'account di origine di replicare le immagini.

Requisiti della politica del registro di destinazione

L'account di destinazione deve configurare una politica di autorizzazioni del registro che conceda all'account di origine l'autorizzazione a eseguire le seguenti azioni:

- `ecr:ReplicateImage`- Consente all'account di origine di replicare le immagini nel registro di destinazione
- `ecr:CreateRepository`- Consente a ECR di creare automaticamente repository nel registro di destinazione se non esistono già

> **Important**
>
> Se non si concede l'`ecr:CreateRepository`autorizzazione, è necessario creare manualmente repository con gli stessi nomi nell'account di destinazione prima che la replica possa avere successo.

Esempio di politica del registro di destinazione:

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "AllowCrossAccountReplication",
            "Effect": "Allow",
            "Principal": {
                "AWS": "arn:aws:iam::111122223333:root"
            },
            "Action": [
                "ecr:ReplicateImage",
                "ecr:CreateRepository"
            ],
            "Resource": "*"
        }
    ]
}
```

Requisiti dell'account di origine

L'account di origine deve solo:

- Configurare le regole di replica nelle impostazioni del registro per specificare l'account e le regioni di destinazione
- Assicurati che il principale IAM che configura la replica disponga delle autorizzazioni ECR necessarie

Non sono richieste politiche aggiuntive sui repository di origine. I repository di origine non necessitano di politiche di repository che concedano autorizzazioni di replica.

Idee sbagliate comuni

Di seguito sono riportate le idee sbagliate più comuni sulle politiche di replica ECR tra account:

- Idea sbagliata: il repository di origine necessita di una policy che consenta all'account di destinazione di replicare le immagini.

  Realtà: i repository di origine non necessitano di politiche speciali per la replica.
- Idea sbagliata: sia gli account di origine che quelli di destinazione necessitano di politiche di registro.

  Realtà: solo l'account di destinazione necessita di una politica di autorizzazioni di registro.
- Idea sbagliata: le politiche del repository e le politiche del registro sono la stessa cosa.

  Realtà: le politiche del repository controllano l'accesso ai singoli repository, mentre le politiche del registro controllano le operazioni a livello di registro come la replica.

Risoluzione dei problemi di replica

Se la replica tra account non riesce, controlla quanto segue:

- Verifica che per l'account di destinazione sia configurata una politica di autorizzazione del registro
- Assicurati che la politica del registro includa entrambe `ecr:ReplicateImage` le `ecr:CreateRepository` azioni
- Verifica che l'ID dell'account di origine sia specificato correttamente nella politica del registro di destinazione
- Verifica che i repository di destinazione esistano (se non `ecr:CreateRepository` è concesso)
- Controlla CloudTrail i log per le chiamate non riuscite `CreateRepository` o `ReplicateImage` le chiamate API

Considerazioni per la replica di immagini private

Quando usi la replica di immagini private, tieni presenti le considerazioni seguenti:

- Viene replicato solo il contenuto del repository inviato o ripristinato in un repository dopo la configurazione della replica. Qualsiasi contenuto preesistente in un repository non sarà replicato. Se un'immagine viene ripristinata dopo l'attivazione della replica, verrà replicata. Se viene ripristinata prima dell'attivazione della replica, non verrà replicata.
- Il nome del repository rimarrà lo stesso in tutte le regioni e gli account una volta che la replica sarà stata effettuata. Amazon ECR non supporta la modifica del nome del repository durante la replica.
- La prima volta che configuri il registro privato per la replica, Amazon ECR crea un ruolo IAM collegato ai servizi per tuo conto. Il ruolo IAM collegato ai servizi concede al servizio di replica Amazon ECR l'autorizzazione necessaria per creare repository e replicare immagini nel registro. Per ulteriori informazioni, consulta Utilizzo di ruoli collegati ai servizi per Amazon ECR.
- Affinché si verifichi la replica tra account, la destinazione del registro privato deve concedere l'autorizzazione per consentire al registro di origine di replicare le immagini. Per questa operazione viene effettuato l'impostazione di una policy delle autorizzazioni del registro privato. Per ulteriori informazioni, consulta Autorizzazioni di registro private in Amazon ECR.
- Se le policy di autorizzazione per un registro privato vengono modificate per rimuovere un'autorizzazione, le repliche in corso precedentemente concesse potrebbero essere completate.
- Affinché si verifichi la replica tra regioni, sia l'account di origine che quello di destinazione devono essere abilitati alla regione prima di eseguire qualsiasi azione di replica all'interno o verso quella regione. Per ulteriori informazioni, consulta Gestione delle regioni AWS nella Riferimenti generali di Amazon Web Services.
- La replica tra regioni non è supportata tra le partizioni. AWS Ad esempio, un repository in `us-west-2` non può essere replicato in `cn-north-1`. Per ulteriori informazioni sulle AWS partizioni, vedere il formato ARN AWS nella Guida generale.
- La configurazione di replica per un registro privato può contenere fino a 25 destinazioni univoche per tutte le regole, con un massimo di 10 regole totali. Ogni regola può contenere fino a 100 filtri. Ciò consente di specificare regole separate per i repository contenenti immagini utilizzate per la produzione e il test, ad esempio.
- La configurazione di replica supporta il filtro dei repository di un registro privato che vengono replicati specificando un prefisso del repository. Per vedere un esempio, consulta Esempio: configurazione della replica tra regioni utilizzando un filtro repository.

- Un'azione di replica si verifica solo una volta per invio o ripristino dell'immagine. Ad esempio, se è stata configurata la replica tra regioni da `us-west-2` a `us-east-1` e da `us-east-1` a `us-east-2`, un'immagine inviata a `us-west-2` si replica solo in `us-east-1`, non si replica di nuovo in `us-east-2`. Questo comportamento si applica sia alla replica tra regioni e che tra account.
- La maggior parte delle immagini viene replicata in meno di 30 minuti, ma in rari casi la replica potrebbe richiedere più tempo.
- La replica del registro non esegue alcuna azione di eliminazione o archiviazione. Le immagini e gli archivi replicati possono essere eliminati o archiviati quando non vengono più utilizzati.
- Se l'immagine da replicare viene archiviata nella destinazione, verrà ripristinata nella destinazione.
- Quando un'immagine viene archiviata in una regione di origine, non verrà archiviata in una regione di destinazione specificata dalla configurazione di replica.
- le policy del repository, tra cui le policy IAM e le policy del ciclo di vita, non vengono replicate e non hanno alcun effetto se non sul repository per cui sono definite.
- Le impostazioni del repository non vengono replicate per impostazione predefinita, ma è possibile replicarle utilizzando i modelli di creazione del repository. Queste impostazioni includono la mutabilità dei tag, la crittografia, le autorizzazioni del repository e le politiche del ciclo di vita. Per ulteriori informazioni sui modelli di creazione di repository, vedere. Modelli per controllare gli archivi creati durante un'azione di pull through cache, di creazione in modalità push o di replica
- Se l'immutabilità dei tag è abilitata in un repository e viene replicata un'immagine che utilizza lo stesso tag di un'immagine esistente, l'immagine viene replicata ma non conterrà il tag duplicato. In questo modo all'immagine potrebbero non essere assegnati tag.

Esempi di replica di immagini private per Amazon ECR

Gli esempi seguenti riportano casi d'uso comuni per la replica di immagini private. Se configuri la replica utilizzando AWS CLI, puoi utilizzare gli esempi JSON come punto di partenza per creare il tuo file JSON. Se configuri la replica utilizzando il Console di gestione AWS, vedrai un codice JSON simile quando esamini la regola di replica nella pagina Rivedi e invia.

Esempio: configurazione della replica tra regioni in un'unica regione di destinazione

Di seguito viene illustrato un esempio per la configurazione della replica tra regioni all'interno di un unico registro. In questo esempio si presuppone che l'ID account sia `111122223333` e che si sta specificando questa configurazione di replica in una regione diversa da `us-west-2`.

```
{
    "rules": [
        {
            "destinations": [
                {
                    "region": "us-west-2",
                    "registryId": "111122223333"
                }
            ]
        }
    ]
}
```

Esempio: configurazione della replica tra regioni utilizzando un filtro repository

Di seguito viene illustrato un esempio per la configurazione della replica tra regioni per repository che corrispondono a un valore del nome di prefisso. In questo esempio si presuppone che l'ID account sia `111122223333` e che si sta specificando questa configurazione di replica in una regione diversa da `us-west-1` e che ci siano repository con prefisso `prod`.

```
{
 "rules": [{
  "destinations": [{
   "region": "us-west-1",
   "registryId": "111122223333"
  }],
  "repositoryFilters": [{
   "filter": "prod",
   "filterType": "PREFIX_MATCH"
  }]
 }]
}
```

Esempio: configurazione della replica tra regioni più regioni di destinazione

Di seguito viene illustrato un esempio per la configurazione della replica tra regioni all'interno di un unico registro. Questo esempio presuppone che l'ID del tuo account sia `111122223333` e che tu stia specificando questa configurazione di replica in una regione diversa da o. `us-west-1` `us-west-2`

```
{
    "rules": [
        {
            "destinations": [
                {
                    "region": "us-west-1",
                    "registryId": "111122223333"
                },
                {
                    "region": "us-west-2",
                    "registryId": "111122223333"
                }
            ]
        }
    ]
}
```

Esempio: configurazione della replica tra account

Di seguito viene illustrato un esempio per la configurazione della replica tra account per il registro. In questo esempio viene configurata la replica per l'account `444455556666` e per la regione `us-west-2`.

> **Important**
>
> Affinché si verifichi la replica tra account, l'account di destinazione deve configurare una policy delle autorizzazioni del registro per consentire la replica dal registro. Per ulteriori informazioni, consulta Autorizzazioni di registro private in Amazon ECR.

```
{
    "rules": [
        {
            "destinations": [
                {
                    "region": "us-west-2",
                    "registryId": "444455556666"
                }
            ]
        }
    ]
```

```
}
```

Esempio: specifica di più regole in una configurazione

Di seguito viene illustrato un esempio per la configurazione di più regole di replica per il registro. In questo esempio viene configurata la replica per l'account *111122223333* con una regola che replica i repository con un prefisso `prod` alla regione `us-west-2` e i repository con un prefisso `test` alla regione `us-east-2`. Una configurazione di replica può contenere fino a 10 regole, ognuna delle quali specifica fino a 25 destinazioni.

```
{
 "rules": [{
   "destinations": [{
    "region": "us-west-2",
    "registryId": "111122223333"
   }],
   "repositoryFilters": [{
    "filter": "prod",
    "filterType": "PREFIX_MATCH"
   }]
  },
  {
   "destinations": [{
    "region": "us-east-2",
    "registryId": "111122223333"
   }],
   "repositoryFilters": [{
    "filter": "test",
    "filterType": "PREFIX_MATCH"
   }]
  }
 ]
}
```

Esempio: rimozione di tutte le impostazioni di replica

Di seguito viene illustrato un esempio di rimozione di tutte le impostazioni di replica dal registro. Per rimuovere le impostazioni di replica, è necessario configurare un array di regole vuoto.

```
{
    "rules": []
```

```
}
```

> **Important**
>
> La rimozione delle impostazioni di replica non elimina gli archivi o le immagini precedentemente replicati. È necessario eliminare manualmente il contenuto replicato se non è più necessario.

Configurazione della replica di immagini private in Amazon ECR

Configura la replica per regione per il tuo registro privato. È possibile configurare la replica tra regioni o la replica tra account.

Per esempi di uso comune della replica, consulta Esempi di replica di immagini private per Amazon ECR.

Per configurare le impostazioni di replica del registro (Console di gestione AWS)

1. Apri la console Amazon ECR nei https://console.aws.amazon.com/ecr/repository.
2. Dalla barra di navigazione, scegli la regione in cui configurare le impostazioni di replica del registro.
3. Nel pannello di navigazione, seleziona Private registry (Registro privato).
4. Nella pagina del registro privato, scegli Impostazioni, quindi scegli Modifica in Configurazione della replica.
5. Alla pagina Replication (Replica), scegliere Add replication rule (Aggiunta di regola di replica).
6. Alla pagina Destination types (Tipi di destinazione), scegliere se abilitare la replica tra regioni, la replica tra account o entrambe, quindi scegliere Next (Successivo).
7. Se la replica tra regioni è abilitata, per Configure destination regions (Configurare regioni di destinazione) scegliere una o più Destination regions (Regioni di destinazione) e quindi scegliere Next (Successivo).
8. Se la replica tra account è abilitata, per Cross-account replication (Replica tra account) scegliere l'impostazione di replica tra account per il registro. Per Destination account (Account di destinazione), immettere l'ID account per l'account di destinazione e una o più Destination regions (Regioni di destinazione) a cui replicare. Scegliere Destination account + (Account di destinazione +) per configurare account aggiuntivi come destinazioni di replica.

> **Important**
>
> Affinché si verifichi la replica tra account, l'account di destinazione deve configurare una policy delle autorizzazioni del registro per consentire la replica dal registro. Per ulteriori informazioni, consulta Autorizzazioni di registro private in Amazon ECR.

9. (Facoltativo) Alla pagina Add filters (Aggiungi filtri), specificare uno o più filtri per la regola di replica, quindi scegliere Add (Aggiungi). Ripetere questo passaggio per ogni filtro da associare all'operazione di replica. Un filtro deve essere specificato come prefisso del nome del repository. Se non viene aggiunto alcun filtro, il contenuto di tutti i repository viene replicato. Scegliere Next (Successivo) una volta aggiunti tutti i filtri.
10. Alla pagina Review and submit (Verifica e invia), verificare la configurazione della regola di replica e quindi scegliere Submit rule (Invia regola).

Per configurare le impostazioni di replica del registro (AWS CLI)

1. Creare un file JSON contenente le regole di replica da definire per il registro. Una configurazione di replica può contenere fino a 10 regole, con un massimo di 25 destinazioni specifiche in tutte le regole e 100 filtri per ciascuna regola. Per configurare la replica tra regioni all'interno del proprio account, è necessario specificare il proprio ID account. Per ulteriori esempi, consulta Esempi di replica di immagini private per Amazon ECR.

```
{
 "rules": [{
  "destinations": [{
   "region": "destination_region",
   "registryId": "destination_accountId"
  }],
  "repositoryFilters": [{
   "filter": "repository_prefix_name",
   "filterType": "PREFIX_MATCH"
  }]
 }]
}
```

2. Crea una configurazione di replica per il tuo registro.

```
aws ecr put-replication-configuration \
```

```
    --replication-configuration file://replication-settings.json \
    --region us-west-2
```

3. Confermare le impostazioni del registro.

```
aws ecr describe-registry \
    --region us-west-2
```

Rimozione delle impostazioni di replica delle immagini private in Amazon ECR

Per rimuovere o disabilitare le impostazioni di replica per il registro privato, è necessario configurare una configurazione di replica vuota. Non è presente alcun comando di rimozione dedicato in. AWS CLI

Per rimuovere le impostazioni di replica del registro ()Console di gestione AWS

1. Apri la console Amazon ECR nei https://console.aws.amazon.com/ecr/repository.
2. Dalla barra di navigazione, scegli la regione da cui rimuovere le impostazioni di replica del registro.
3. Nel pannello di navigazione, seleziona Private registry (Registro privato).
4. Nella pagina del registro privato, scegli Impostazioni, quindi scegli Modifica in Configurazione della replica.
5. Rimuovi tutte le regole di replica esistenti scegliendo l'opzione di eliminazione per ogni regola.
6. Scegliete Salva per applicare la configurazione di replica vuota.

Per rimuovere le impostazioni di replica del registro ()AWS CLI

1. Crea un file JSON con un array di regole vuoto per rimuovere tutte le impostazioni di replica.

```
{
    "rules": []
}
```

2. Applica la configurazione di replica vuota al registro.

```
aws ecr put-replication-configuration \
```

```
    --replication-configuration file://empty-replication-settings.json \
    --region us-west-2
```

3. Conferma che le impostazioni di replica sono state rimosse.

```
aws ecr describe-registry \
    --region us-west-2
```

L'output dovrebbe mostrare un valore vuoto `replicationConfiguration` senza regole.

> **Important**
>
> La rimozione delle impostazioni di replica non elimina gli archivi o le immagini precedentemente replicati. È necessario eliminare manualmente il contenuto replicato se non è più necessario.

Modelli per controllare gli archivi creati durante un'azione di pull through cache, di creazione in modalità push o di replica

Utilizza i modelli di creazione di repository Amazon ECR per definire le impostazioni per i repository creati da Amazon ECR per tuo conto. Le impostazioni in un modello di creazione di repository vengono applicate solo durante la creazione del repository e non hanno alcun effetto sui repository esistenti o sui repository creati con altri metodi. Attualmente, i modelli di creazione di repository possono essere utilizzati per applicare le impostazioni durante la creazione di repository per queste funzionalità:

- Scorri la cache
- Crea in modalità push
- Replica

Funzionamento dei modelli di creazione di repository

A volte Amazon ECR deve creare un nuovo repository privato per tuo conto. Esempio:

- La prima volta che utilizzi una regola pull through cache per recuperare il contenuto di un repository upstream e archiviarlo nel tuo registro privato Amazon ECR.
- Quando invii un'immagine a un repository che non esiste ancora.
- Quando desideri che Amazon ECR replichi un repository in un'altra regione o account.

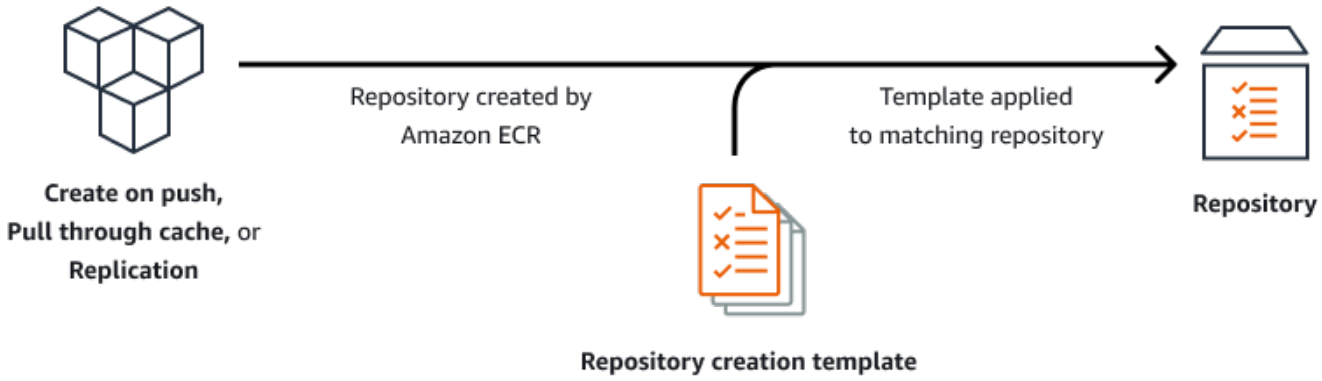
Quando non esiste un modello di creazione del repository che corrisponda alla regola pull through cache o al repository replicato, Amazon ECR utilizza le impostazioni predefinite per il nuovo repository. Queste impostazioni predefinite includono la disattivazione dell'immutabilità dei tag, l'utilizzo della crittografia `AES-256` e la mancata applicazione di policy di repository o del ciclo di vita.

Se non esiste un modello per la creazione di un repository che corrisponda al repository di destinazione per un push di immagini, Amazon ECR non creerà un repository con impostazioni predefinite.

L'utilizzo di un modello di creazione di repository ti offre la possibilità di definire le impostazioni che Amazon ECR applica ai nuovi repository creati tramite la cache pull through, le azioni di creazione

su push e di replica. Puoi definire l'immutabilità dei tag, la configurazione della crittografia, le autorizzazioni dei repository, la policy del ciclo di vita e i tag delle risorse per i nuovi repository.

Il diagramma seguente mostra il flusso di lavoro utilizzato da Amazon ECR quando viene utilizzato un modello di creazione repository.



Di seguito sono descritti in dettaglio i parametri di un modello di creazione repository.

Prefix

Il prefisso è il prefisso dello spazio dei nomi del repository da associare al modello. A tutti i repository creati utilizzando questo prefisso verranno applicate le impostazioni definite in questo modello. Ad esempio, il prefisso `prod` verrebbe applicato a tutti i repository che iniziano con `prod/`. Analogamente, il prefisso `prod/team` verrebbe applicato a tutti i repository che iniziano con `prod/team/`. In un registro contenente due modelli, se un modello ha il prefisso «prod» e l'altro ha il prefisso "/». prod/team", the template with the prefix "prod/team" will be applied to all repositories whose names start with "prod/team

Per applicare un modello a tutti i repository del registro a cui non è associato un modello di creazione, puoi utilizzare `ROOT` come prefisso.

> **Important**
>
> Si presuppone che `/` venga sempre applicato alla fine del prefisso. Se specifichi `ecr-public` come prefisso, Amazon ECR lo considera come `ecr-public/`. Quando utilizzi una regola di cache pull-through, il prefisso del repository specificato durante la creazione della regola è quello da specificare anche come prefisso del modello di creazione di repository.

Description

Questa descrizione del modello è facoltativa e viene utilizzata per descrivere lo scopo del modello di creazione del repository.

Applicato per

L'impostazione Applied for determina quali repository creati da Amazon ECRR verranno creati con questo modello. I valori validi sono `PULL_THROUGH_CACHE`, `CREATE_ON_PUSH` e `REPLICATION`. Questo succede, ad esempio, la prima volta che utilizzi una regola di cache pull-through per recuperare il contenuto di un repository upstream e archiviarlo nel registro privato di Amazon ECR. Quando non è presente un modello di creazione di repository che corrisponda alla regola di cache pull-through, Amazon ECR utilizza le impostazioni predefinite per il nuovo repository.

Ruolo di creazione del repository

Il ruolo di creazione del repository è un ARN del ruolo IAM che verrà assunto da Amazon ECR durante la creazione e la configurazione di repository tramite modelli di creazione di repository. Questo ruolo deve essere fornito quando si utilizzano i tag di repository and/or KMS nel modello, altrimenti la creazione del repository avrà esito negativo.

Mutabilità dei tag di immagine

L'impostazione di mutabilità dei tag da utilizzare per i repository creati utilizzando il modello. Se questo parametro viene omesso, verrà utilizzata l'impostazione predefinita MUTABLE, che consentirà la sovrascrittura dei tag immagine. Questa è l'impostazione consigliata da utilizzare per i modelli utilizzati per i repository creati dalle azioni di cache pull-through. Ciò garantisce che Amazon ECR possa aggiornare le immagini memorizzate nella cache quando i tag sono gli stessi.

Se è specificato IMMUTABLE, tutti i tag immagine all'interno del repository saranno immutabili, per cui non potranno essere sovrascritti.

Configurazione della crittografia

> **Important**
>
> La crittografia lato server a doppio livello con AWS KMS (DSSE-KMS) è disponibile solo nelle regioni. AWS GovCloud (US)

La configurazione della crittografia da utilizzare per i repository creati utilizzando il modello.

Se utilizzi il tipo di crittografia KMS, il contenuto del repository verrà crittografato utilizzando la crittografia lato server con la chiave AWS Key Management Service archiviata in AWS KMS. Quando lo utilizzi AWS KMS per crittografare i tuoi dati, puoi utilizzare la AWS KMS chiave AWS gestita predefinita per Amazon ECR o specificare la tua AWS KMS chiave, che hai già creato. Puoi inoltre scegliere di utilizzare la crittografia a livello singolo o doppio con. AWS KMS Per ulteriori informazioni, consulta Crittografia dei dati a riposo. Se utilizzi il tipo di crittografia KMS e lo utilizzi con la replica tra regioni, potresti aver bisogno di autorizzazioni aggiuntive. Per ulteriori informazioni, consulta Creazione di una politica chiave KMS per la replica.

Se utilizzi il tipo di crittografia AES256, Amazon ECR utilizza la crittografia lato server con chiavi di crittografia gestite da Amazon S3 che crittografano le immagini nel repository utilizzando un algoritmo di crittografia AES-256. Per ulteriori informazioni, consulta Protezione dei dati mediante la crittografia lato server con chiavi di crittografia gestite da Amazon S3 (SSE-S3) nella Guida per l'utente di Amazon Simple Storage Service.

Autorizzazioni del repository

La policy di repository da applicare ai repository creati utilizzando il modello. Una policy di repository utilizza autorizzazioni basate sulle risorse per controllare l'accesso a un repository. Le autorizzazioni basate sulle risorse ti consentono di specificare gli utenti o i ruoli IAM che hanno accesso a un repository e quali operazioni possono eseguire. Per impostazione predefinita, solo l' AWS account che ha creato il repository ha accesso a un repository. È possibile applicare un documento di policy per concedere o negare autorizzazioni aggiuntive al repository. Per ulteriori informazioni, consulta Politiche di archivio privato in Amazon ECR.

Policy del ciclo di vita del repository

La policy del ciclo di vita da utilizzare per i repository creati utilizzando il modello. Una policy del ciclo di vita offre un maggiore controllo sulla gestione del ciclo di vita delle immagini in un repository privato. Una policy del ciclo di vita contiene una o più regole, laddove ogni regola definisce un'operazione per Amazon ECR. Ciò consente di automatizzare la pulizia delle immagini dei container, facendo scadere le immagini in base all'età o al conteggio. Per ulteriori informazioni, consulta Automatizza la pulizia delle immagini utilizzando le politiche del ciclo di vita in Amazon ECR.

Tag delle risorse

I tag delle risorse sono metadati da applicare al repository per facilitarne la catalogazione e l'organizzazione. Ogni tag è composto da una chiave e da un valore opzionale, entrambi personalizzabili. Questa autorizzazione deve essere applicata alla politica del registro di destinazione se si utilizzano modelli di creazione di repository con replica interregionale.

Creazione di un modello di creazione di repository in Amazon ECR

Puoi creare un modello di creazione di repository per definire le impostazioni da utilizzare per i repository creati da Amazon ECR per tuo conto durante le azioni di pull through cache, create on push o replica. Una volta creato il modello di creazione repository, verranno applicate le impostazioni a tutti i nuovi repository creati. Ciò non ha alcun effetto sui repository creati in precedenza.

Quando configuri un repository con modelli, hai la possibilità di specificare chiavi KMS e tag di risorse. Se intendi utilizzare chiavi KMS, tag di risorsa o una combinazione di entrambi in uno o più modelli, devi:

- Crea una politica personalizzata per i modelli di creazione di repository.
- Crea un ruolo IAM per i modelli di creazione di repository.

Una volta configurato, puoi associare il ruolo personalizzato a modelli specifici nel registro.

Autorizzazioni IAM per la creazione di modelli di creazione di repository

Le seguenti autorizzazioni sono necessarie per consentire a un principale IAM di gestire i modelli di creazione repository. Questa autorizzazione deve essere concessa utilizzando una policy IAM basata sull'identità.

- `ecr:CreateRepositoryCreationTemplate`: concede l'autorizzazione a creare un modello di creazione repository.
- `ecr:UpdateRepositoryCreationTemplate`— Concede l'autorizzazione ad aggiornare un modello di creazione di repository.
- `ecr:DescribeRepositoryCreationTemplates`— Concede l'autorizzazione a elencare i modelli di creazione di repository in un registro.
- `ecr:DeleteRepositoryCreationTemplate`: concede l'autorizzazione a eliminare un modello di creazione repository.
- `ecr:CreateRepository`— Concede l'autorizzazione a creare un repository Amazon ECR.
- `ecr:PutLifecyclePolicy`: concede l'autorizzazione a creare una policy del ciclo di vita e applicarla a un repository. Questa autorizzazione è necessaria solo se il modello di creazione repository include una policy del ciclo di vita.

- `ecr:SetRepositoryPolicy`: concede l'autorizzazione a creare una policy di autorizzazioni per un repository. Questa autorizzazione è necessaria solo se il modello di creazione repository include una policy del repository.
- `iam:PassRole`— Concede l'autorizzazione a consentire a un'entità di trasferire un ruolo a un servizio o un'applicazione. Questa autorizzazione è necessaria per i servizi e le applicazioni che devono assumere un ruolo per eseguire azioni per conto dell'utente.

Crea una politica personalizzata per i modelli di creazione di repository

È possibile utilizzare il Console di gestione AWS per definire una policy che verrà successivamente associata a un ruolo IAM. Questo ruolo IAM può quindi essere utilizzato come ruolo di creazione di un repository durante la configurazione di un modello di creazione di repository.

Console di gestione AWS

Utilizzare l'editor di policy JSON per creare una policy personalizzata per i modelli di creazione di repository.

1. Accedi Console di gestione AWS e apri la console IAM all'indirizzo. https://console.aws.amazon.com/iam/
2. Nel riquadro di navigazione a sinistra, seleziona Policies (Policy).
3. Scegli Crea policy.
4. Nella sezione Editor di policy, scegli l'opzione JSON.
5. Inserisci la seguente policy nel campo JSON.

   JSON

```
{
        "Version":"2012-10-17",
        "Statement": [
            {
                "Effect": "Allow",
                "Action": [
                    "ecr:CreateRepository",
                    "ecr:ReplicateImage",
                    "ecr:TagResource"
                ],
                "Resource": "*"
```

```
            },
            {
                "Effect": "Allow",
                "Action": [
                    "kms:CreateGrant",
                    "kms:RetireGrant",
                    "kms:DescribeKey"
                ],
                "Resource": "*"
            }
        ]
}
```

6. Risolvi eventuali avvisi di sicurezza, errori o avvisi generali generati durante la convalida delle policy, quindi scegli Avanti.
7. Una volta terminata l'aggiunta delle autorizzazioni alla policy, scegli Successivo.
8. Nella pagina Verifica e crea, digita i valori per Nome policy e Descrizione (facoltativa) per la policy che si sta creando. Rivedi Autorizzazioni definite in questa policy per visualizzare le autorizzazioni concesse dalla policy.
9. Seleziona Crea policy per salvare la nuova policy.
10. Crea un ruolo per assegnare questo criterio per il modello di creazione, vedi. Crea un ruolo IAM per i modelli di creazione di repository

Crea un ruolo IAM per i modelli di creazione di repository

Puoi utilizzare il Console di gestione AWS per creare un ruolo che può essere utilizzato da Amazon ECR quando specifichi il ruolo di creazione del repository in un modello di creazione di repository che utilizza tag di repository o KMS in un modello.

Console di gestione AWS

Per creare un ruolo.

1. Accedi Console di gestione AWS e apri la console IAM all'indirizzo https://console.aws.amazon.com/iam/.
2. Nel riquadro di navigazione della console, selezionare Ruoli e Crea ruolo.
3. Scegli il tipo di ruolo Custom Trust Policy.

4. Nella sezione Politica di fiducia personalizzata, incolla la politica di fiducia personalizzata elencata di seguito:

   JSON

```
{
"Version":"2012-10-17",
"Statement": [
    {
        "Effect": "Allow",
        "Principal": {
            "Service": "ecr.amazonaws.com"
        },
        "Action": "sts:AssumeRole"
    }
    ]
}
```

5. Scegli Next (Successivo).
6. Dalla pagina Aggiungi autorizzazioni, seleziona la casella di controllo accanto alla politica personalizzata creata in precedenza dall'elenco delle politiche di autorizzazione e scegli Avanti.
7. In Nome ruolo, immetti un nome per il ruolo. I nomi dei ruoli devono essere univoci all'interno del tuo. Account AWS Quando il nome di un ruolo viene utilizzato in una policy o come parte di un ARN, il nome del ruolo fa distinzione tra maiuscole e minuscole. Quando un nome di ruolo viene visualizzato ai clienti nella console, ad esempio durante la procedura di accesso, il nome del ruolo non fa distinzione tra maiuscole e minuscole. Poiché varie entità possono fare riferimento al ruolo, non puoi modificare il nome del ruolo dopo averlo creato.
8. (Facoltativo) In Descrizione, inserisci una descrizione per il nuovo ruolo.
9. Rivedere il ruolo e scegliere Crea ruolo.

Crea un modello per la creazione di un repository

Dopo aver completato i prerequisiti necessari per i modelli, è possibile procedere alla creazione dei modelli per la creazione del repository.

Console di gestione AWS

Creazione di un modello di creazione repository (Console di gestione AWS)

1. Apri la console Amazon ECR all'indirizzo https://console.aws.amazon.com/ecr/.
2. Dalla barra di navigazione seleziona la regione in cui creare il modello di creazione repository.
3. Nel riquadro di navigazione seleziona Registro privato, Modelli di creazione repository.
4. Nella pagina Modelli di creazione repository seleziona Crea modello.
5. Nella pagina Passaggio 1: definisci il modello, per Dettagli del modello seleziona Un prefisso specifico per applicare il modello a un prefisso dello spazio dei nomi di un repository specifico o seleziona Qualsiasi prefisso nel registro ECR per applicare il modello a tutti i repository che non corrispondono a nessun altro modello della Regione.

   a. Se selezioni Un prefisso specifico per Prefisso, specifica il prefisso dello spazio dei nomi del repository a cui applicare il modello. Si presuppone che `/` venga sempre applicato alla fine del prefisso. Ad esempio, il prefisso `prod` verrebbe applicato a tutti i repository che iniziano con `prod/`. Analogamente, il prefisso `prod/team` verrebbe applicato a tutti i repository che iniziano con `prod/team/`.

   b. Se selezioni Qualsiasi prefisso nel registro ECR, il Prefisso verrà impostato su. `ROOT`.

6. Per Applied for, specifica a quali flussi di lavoro Amazon ECR verrà applicato questo modello. Le opzioni sono `PULL_THROUGH_CACHE`, `CREATE_ON_PUSH` e `REPLICATION`.
7. Per Descrizione del modello, specifica una descrizione facoltativa del modello, quindi seleziona Avanti.
8. Nella pagina Passaggio 2: aggiungi una configurazione di creazione del repository e specifica la configurazione delle impostazioni del repository da applicare ai repository creati utilizzando il modello.

   a. Per Mutabilità dei tag immagine, seleziona l'impostazione di mutabilità dei tag da utilizzare. Per ulteriori informazioni, consulta Impedire la sovrascrittura dei tag di immagine in Amazon ECR.

      - Mutabile: scegli questa opzione se desideri che i tag delle immagini vengano sovrascritti. Consigliato per i repository che utilizzano azioni pull through cache per garantire che Amazon ECR possa aggiornare le immagini memorizzate nella cache. Inoltre, per disabilitare gli aggiornamenti dei tag per alcuni tag mutabili, inserisci i

nomi dei tag o usa i caratteri jolly (*) per abbinare più tag simili nella casella di testo Esclusione tag mutabili.

- Immutabile: scegli questa opzione se vuoi evitare che i tag delle immagini vengano sovrascritti e si applica a tutti i tag e le esclusioni presenti nel repository quando inserisci un'immagine con un tag esistente. Amazon ECR restituisce un `ImageTagAlreadyExistsException` messaggio se tenti di inviare un'immagine con un tag esistente. Inoltre, per abilitare gli aggiornamenti dei tag per alcuni tag immutabili, inserisci i nomi dei tag o usa i caratteri jolly (*) per abbinare più tag simili nella casella di testo Esclusione tag immutabile.

b. Per la configurazione della crittografia, scegli l'impostazione di crittografia da utilizzare. Per ulteriori informazioni, consulta Crittografia dei dati a riposo.

   Quando è selezionato AES-256, Amazon ECR utilizza la crittografia lato server con chiavi di crittografia gestite da Amazon Simple Storage Service che crittografa i dati a riposo utilizzando un algoritmo di crittografia AES-256 standard di settore. Questa funzionalità è disponibile senza costi aggiuntivi.

   Quando è selezionato AWS KMS, Amazon ECR utilizza la crittografia lato server con chiavi archiviate in AWS Key Management Service (AWS KMS). Quando si utilizza AWS KMS per crittografare i dati, è possibile utilizzare la chiave AWS gestita predefinita, gestita da Amazon ECR, o specificare la propria AWS KMS chiave, denominata chiave gestita dal cliente.

   > **Note**
   >
   > Le impostazioni di crittografia per un repository non possono essere modificate una volta che il repository è stato creato.

c. Per Autorizzazioni del repository specifica la policy di autorizzazione di repository da applicare ai repository creati utilizzando questo modello. Facoltativamente, puoi utilizzare il menu a discesa per selezionare uno degli esempi JSON per i casi d'uso più comuni. Per ulteriori informazioni, consulta Politiche di archivio privato in Amazon ECR.

d. Per Policy del ciclo di vita dei repository specifica la policy di ciclo di vita dei repository da applicare ai repository creati utilizzando questo modello. Facoltativamente, puoi utilizzare il menu a discesa per selezionare uno degli esempi JSON per i casi d'uso più comuni. Per ulteriori informazioni, consulta Automatizza la pulizia delle immagini utilizzando le politiche del ciclo di vita in Amazon ECR.

e. Per i AWS tag Repository, specifica i metadati, sotto forma di coppie chiave-valore, da associare ai repository creati utilizzando questo modello, quindi scegli Avanti. Per ulteriori informazioni, consulta Taggare un repository privato in Amazon ECR.

f. Per il ruolo di creazione del repository, seleziona un ruolo IAM personalizzato dal menu a discesa da utilizzare per i modelli di creazione del repository quando utilizzi i tag del repository o KMS nel modello (vedi per i dettagli). Quindi scegli Avanti. Crea un ruolo IAM per i modelli di creazione di repository

9. Nella pagina Passaggio 3: rivedi e crea, rivedi le impostazioni specificate per il modello di creazione repository. Seleziona l'opzione Modifica per apportare le modifiche. Al termine, seleziona Crea.

AWS CLI

Il create-repository-creation-template AWS CLI comando viene utilizzato per creare un modello di creazione di repository per il registro privato.

Creazione di un modello di creazione repository (AWS CLI)

1. Utilizzate il AWS CLI per generare uno scheletro per il create-repository-creation-templatecomando.

```
aws ecr create-repository-creation-template \
    --generate-cli-skeleton
```

L'output del comando mostra la sintassi completa del modello di creazione del repository.

```
{
"appliedFor":[""], // string array, but valid are PULL_THROUGH_CACHE,
 CREATE_ON_PUSH, and REPLICATION
"prefix": "string",
    "description": "string",
    "imageTagMutability":
 "MUTABLE"|"IMMUTABLE"|"IMMUTABLE_WITH_EXCLUSION"|"MUTABLE_WITH_EXCLUSION",
    "imageTagMutabilityExclusionFilters": [
        "filterType": "WILDCARD",
        "filter": "string"
    ],
    "repositoryPolicy": "string",
    "lifecyclePolicy": "string"
```

```
"encryptionConfiguration": {
"encryptionType": "AES256"|"KMS",
        "kmsKey": "string"
    },
    "resourceTags": [
        {
"Key": "string",
            "Value": "string"
        }
    ],
    "customRoleArn": "string", // must be a valid IAM Role ARN
}
```

2. Crea un file denominato `repository-creation-template.json` con l'output del passaggio precedente. Questo modello imposta una chiave di crittografia KMS per qualsiasi repository creato in base `prod/*` a una politica di repository che consente di inviare e caricare immagini nei repository futuri, imposta una politica del ciclo di vita che farà scadere le immagini più vecchie di due settimane e imposta un ruolo personalizzato che consentirà a ECR di accedere alla chiave KMS e assegnare il tag di risorsa ai repository futuri. `examplekey`

```
{
"prefix": "prod",
    "description": "For repositories cached from my PTC rule and in my
 replication configuration that start with 'prod/'",
    "appliedFor": ["PULL_THROUGH_CACHE", "CREATE_ON_PUSH", "REPLICATION"],
    "encryptionConfiguration": {
"encryptionType": "KMS",
        "kmsKey": "arn:aws:kms:us-west-2:111122223333:key/a1b2c3d4-5678-90ab-
cdef-example11111"
    },
    "resourceTags": [
        {
"Key": "examplekey",
            "Value": "examplevalue"
        }
    ],
    "imageTagMutability": "IMMUTABLE_WITH_EXCLUSION",
    "imageTagMutabilityExclusionFilters": [
      {
      "filterType": "WILDCARD",
      "filter": "latest"
```

```
      },
      {
      "filterType": "WILDCARD",
      "filter": "beta*"
      }
    ]
    "repositoryPolicy": "{\"Version\":\"2012-10-17\",       \"Statement\":
[{\"Sid\":\"AllowPushPullIAMRole\",\"Effect\":\"Allow\",\"Principal\":{\"AWS\":
\"arn:aws:iam::111122223333:user\/IAMusername\"},\"Action\":[\"ecr:BatchGetImage
\",\"ecr:BatchCheckLayerAvailability\",\"ecr:CompleteLayerUpload\",
\"ecr:GetDownloadUrlForLayer\",\"ecr:InitiateLayerUpload\",\"ecr:PutImage\",
\"ecr:UploadLayerPart\"]}]}",
    "lifecyclePolicy": "{\"rules\":[{\"rulePriority\":1,\"description\":\"Expire
 images older than 14 days\",\"selection\":{\"tagStatus\":\"any\",\"countType
\":\"sinceImagePushed\",\"countUnit\":\"days\",\"countNumber\":14},\"action\":
{\"type\":\"expire\"}}]}",
    "customRoleArn": "arn:aws:iam::111122223333:role/myRole"
}
```

3. Utilizzate il seguente comando per creare un modello di creazione del repository. Assicuratevi di specificare il nome del file di configurazione creato nel passaggio precedente anziché `repository-creation-template.json` nel seguente esempio.

```
aws ecr create-repository-creation-template \
    --cli-input-json file://repository-creation-template.json
```

Aggiornamento di un modello per la creazione di un repository

Puoi modificare un modello di creazione di repository se devi modificarne le configurazioni. Una volta modificato il modello di creazione del repository, le nuove configurazioni verranno applicate al modello esistente.

> **Important**
>
> Ciò non ha alcun effetto sui repository creati in precedenza.

Console di gestione AWS

Per modificare un modello di creazione di un repository ()Console di gestione AWS

1. Apri la console Amazon ECR all'indirizzo https://console.aws.amazon.com/ecr/.
2. Dalla barra di navigazione, scegli la regione in cui si trova il modello di creazione del repository da modificare.
3. Nel riquadro di navigazione, scegli Registro privato, quindi scegli Impostazioni.
4. Dalla barra di navigazione, scegli i modelli di creazione del repository.
5. Nella pagina Modelli di creazione del repository, seleziona il modello di creazione del repository da modificare.
6. Dal menu a discesa Azioni, scegli Modifica.
7. Rivedi e aggiorna le impostazioni di configurazione.
8. Scegli aggiorna per applicare le nuove configurazioni del modello di creazione.

AWS CLI

Per modificare un modello di creazione di un repository ()AWS CLI

- Utilizzate il update-repository-creation-templatecomando per aggiornare un modello di creazione di repository esistente. È necessario specificare il `prefix` valore del modello. L'esempio seguente aggiorna un modello di creazione di repository con il `prod` prefisso.

```
aws ecr update-repository-creation-template \
     --prefix prod \
     --image-tag-mutability="IMMUTABLE_WITH_EXCLUSION" \
     --image-tag-mutability-exclusion-filters filterType=WILDCARD, filter=latest
```

L'output del comando visualizza i dettagli del modello di creazione del repository aggiornato.

Eliminazione di un modello per la creazione di un repository in Amazon ECR

Puoi eliminare un modello di creazione repository se non è più utilizzato. Una volta eliminato un modello di creazione del repository, tutti i repository appena creati con il prefisso associato durante un'azione di pull through cache o di replica erediteranno le impostazioni predefinite, a meno che non

venga trovato un altro modello corrispondente, vedere. Funzionamento dei modelli di creazione di repository

> **Important**
>
> Ciò non ha alcun effetto sui repository creati in precedenza.

Console di gestione AWS

Eliminazione di un modello di creazione repository (Console di gestione AWS)

1. Apri la console Amazon ECR all'indirizzo https://console.aws.amazon.com/ecr/.
2. Dalla barra di navigazione seleziona la regione in cui si trova il modello di creazione repository da eliminare.
3. Nel riquadro di navigazione seleziona Registro privato, Modelli di creazione repository.
4. Nella pagina Modelli di creazione di repository, seleziona il modello di creazione repository da eliminare.
5. Dal menu a discesa Operazioni, seleziona Elimina.

AWS CLI

Eliminazione di un modello di creazione repository (AWS CLI)

- Utilizza il delete-repository-creation-templatecomando.html per eliminare un modello di creazione di repository esistente. È necessario specificare il `prefix` valore del modello. L'esempio seguente elimina un modello di creazione di un repository con il `prod` prefisso.

```
aws ecr delete-repository-creation-template \
     --prefix prod
```

L'output del comando visualizza i dettagli del modello di creazione del repository eliminato.

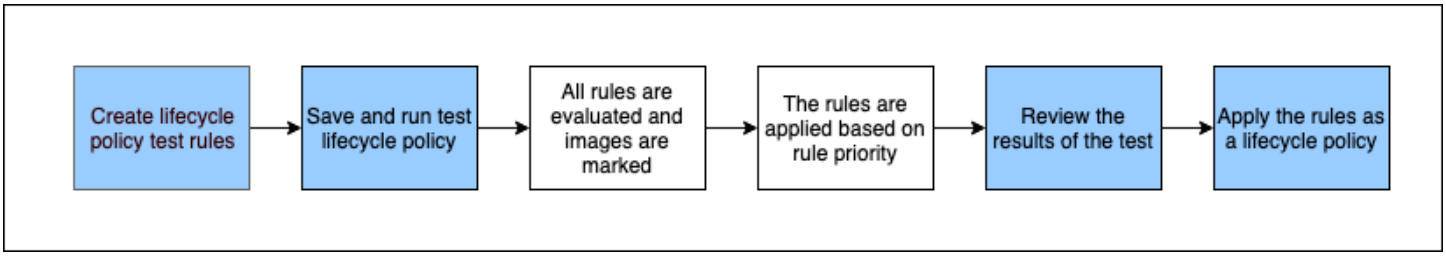
Automatizza la pulizia delle immagini utilizzando le politiche del ciclo di vita in Amazon ECR

Le policy del ciclo di vita di Amazon ECR offrono un maggiore controllo sulla gestione del ciclo di vita delle immagini in un repository privato. Una policy sul ciclo di vita contiene una o più regole e ogni regola definisce un'azione per Amazon ECR. In base ai criteri di scadenza della politica del ciclo di vita, le immagini possono essere archiviate o scadute in base ai criteri specificati nella politica del ciclo di vita entro 24 ore. Quando Amazon ECR esegue un'azione basata su una politica del ciclo di vita, tale azione viene acquisita come evento in. AWS CloudTrail Per ulteriori informazioni, consulta Registrazione delle azioni Amazon ECR con AWS CloudTrail.

Funzionamento delle policy del ciclo di vita

Una policy del ciclo di vita è costituita da una o più regole che determinano quali immagini in un repository devono scadere. Quando si considera l'utilizzo delle policy del ciclo di vita, è importante utilizzare la relativa anteprima per confermare quali immagini la policy del ciclo di vita fa scadere prima di applicarla a un repository. Una volta applicata una policy del ciclo di vita a un repository, le immagini interessate dovrebbero scadere entro 24 ore dal raggiungimento dei criteri di scadenza. Quando Amazon ECR esegue un'azione basata su una policy del ciclo di vita, questa viene acquisita come evento in AWS CloudTrail. Per ulteriori informazioni, consulta Registrazione delle azioni Amazon ECR con AWS CloudTrail.

Il seguente diagramma mostra un flusso di lavoro di policy del il ciclo di vita.



1. Creare una o più regole di test.
2. Salvare le regole di test ed eseguire l'anteprima.
3. Il valutatore delle policy del ciclo di vita passa in rassegna tutte le regole e contrassegna le immagini interessate da ogni regola.
4. Lo strumento di valutazione delle politiche del ciclo di vita applica quindi le regole, in base alla priorità delle regole, e mostra quali immagini nel repository sono impostate per essere scadute o

archiviate. Un numero di priorità della regola più basso significa una priorità più alta. Ad esempio, una regola con priorità 1 ha la precedenza su una regola con priorità 2.

5. Controlla i risultati del test, assicurandoti che le immagini contrassegnate come scadute o archiviate corrispondano a quelle desiderate.
6. Applicare le regole di test come policy del ciclo di vita per il repository.
7. Una volta creata la politica del ciclo di vita, dovresti aspettarti che le immagini scadano o vengano archiviate entro 24 ore dal soddisfacimento dei criteri di scadenza.

Regole per la valutazione delle policy del ciclo di vita

Il valutatore della policy del ciclo di vita è responsabile dell'analisi del JSON di testo normale della policy del ciclo di vita, della valutazione di tutte le regole e quindi dell'applicazione di tali regole in base alla priorità della regola alle immagini nel repository. Di seguito viene illustrata la logica del valutatore delle policy del ciclo di vita in modo più dettagliato. Per alcuni esempi, consulta Esempi di politiche del ciclo di vita in Amazon ECR.

- Quando in un repository sono presenti elementi di riferimento, le policy del ciclo di vita di Amazon ECR scadono o archiviano automaticamente tali artefatti entro 24 ore dall'eliminazione o dall'archiviazione dell'immagine oggetto.
- Tutte le regole vengono valutate contemporaneamente, a prescindere dalla priorità della regola. Dopo che tutte le regole sono state valutate, vengono applicate in base alla priorità della regola.
- Un'immagine è scaduta o archiviata esattamente in base a una o zero regole.
- Un'immagine che soddisfa i requisiti di tagging di una regola non può essere scaduta o archiviata in base a una regola con una priorità inferiore.
- Le regole non possono mai contrassegnare le immagini contrassegnate da regole di priorità più elevata, ma possono comunque identificarle come se non fossero scadute o archiviate.
- L'insieme di tutte le regole che selezionano una classe di archiviazione specifica deve contenere un set unico di prefissi.
- È consentita una sola regola: la selezione di una classe di archiviazione specifica è consentita per selezionare immagini senza tag.
- Se un'immagine è referenziata da un elenco di manifesti, non può essere scaduta o archiviata senza che l'elenco dei manifesti venga prima eliminato o archiviato.
- La scadenza è sempre ordinata per `pushed_at_time` o fa scadere sempre `transitioned_at_time` le immagini più vecchie prima di quelle nuove. Se un'immagine è

stata archiviata e poi ripristinata in qualsiasi momento nel passato, `last_activated_at` viene utilizzata l'immagine al posto di. `pushed_at_time`

- Una regola della policy del ciclo di vita può specificare `tagPatternList` o `tagPrefixList`, ma non entrambi. Tuttavia, una policy del ciclo di vita può contenere più regole in cui regole diverse utilizzano sia elenchi di modelli che di prefissi. Un'immagine viene abbinata correttamente se tutti i tag del `tagPrefixList` valore `tagPatternList` or corrispondono a uno qualsiasi dei tag dell'immagine.
- I parametri `tagPatternList` o `tagPrefixList` possono essere utilizzati solo se `tagStatus` è `tagged`.
- Quando si utilizza `tagPatternList`, un'immagine viene abbinata correttamente se corrisponde al filtro jolly. Ad esempio, se `prod*` viene applicato un filtro di, corrisponderebbe ai tag di immagine il cui nome inizia con`prod`, o. `prod` `prod1` `production-team1` Allo stesso modo, se `*prod*` viene applicato un filtro di, corrisponderebbe ai tag di immagine il cui nome contiene `prod` o. `repo-production` `prod-team`

  > **Important**
  >
  > Esiste un limite massimo di quattro caratteri jolly (`*`) per stringa. Ad esempio, `["*test*1*2*3", "test*1*2*3*"]` è valido ma `["test*1*2*3*4*5*6"]` non è valido.

- Quando viene utilizzata`tagPrefixList`, un'immagine viene abbinata correttamente se tutti i filtri jolly del `tagPrefixList` valore corrispondono a uno qualsiasi dei tag dell'immagine.
- Il `countUnit` parametro viene utilizzato solo se `countType` è`sinceImagePushed`, `sinceImagePulled` o. `sinceImageTransitioned`
- Con`countType = imageCountMoreThan`, le immagini vengono ordinate dalla più recente alla più vecchia in base a `pushed_at_time` e quindi tutte le immagini superiori al numero specificato vengono scadute o archiviate.
- Con`countType = sinceImagePushed`, tutte le immagini più vecchie del `pushed_at_time` numero di giorni specificato in base al numero di giorni specificato `countNumber` sono scadute o archiviate.
- Con`countType = sinceImagePulled`, `countNumber` vengono archiviate tutte `last_recorded_pulltime` le immagini più vecchie del numero di giorni specificato in base a. Se un'immagine non è mai stata estratta, `pushed_at_time` viene utilizzata l'immagine al posto di. `last_recorded_pulltime` Se un'immagine è stata archiviata e poi ripristinata in qualsiasi

momento in passato, ma non è mai stata estratta da quando l'immagine è stata ripristinata, `last_activated_at` viene utilizzata l'immagine al posto di. `last_recorded_pulltime`

- `ConcountType = sinceImageTransitioned`, tutte le immagini archiviate più vecchie del numero di giorni specificato in base `countNumber` a sono scadute. `last_archived_at`
- La scadenza è sempre ordinata per `pushed_at_time` e fa scadere sempre le immagini più vecchie prima di quelle nuove.

Creazione di un'anteprima della politica del ciclo di vita in Amazon ECR

È possibile utilizzare un'anteprima dei criteri del ciclo di vita per vedere l'impatto di un criterio del ciclo di vita su un archivio di immagini prima di applicarlo. Una best practice consiste nell'eseguire un'anteprima prima di applicare una policy del ciclo di vita a un repository.

> **Note**
>
> Se utilizzi la replica di Amazon ECR per creare copie di un repository in diverse regioni o account, tieni presente che una politica del ciclo di vita può intervenire solo sui repository nella regione in cui è stata creata. Pertanto, se è stata attivata la replica, potrebbe essere necessario prendere in considerazione la creazione di una policy del ciclo di vita in ogni Regione e account su cui si stanno replicando i repository.

Per creare un'anteprima di una policy del ciclo di vita (Console di gestione AWS)

1. Apri la console Amazon ECR nei https://console.aws.amazon.com/ecr/repository.
2. Sulla barra di navigazione seleziona la regione che contiene il repository sul quale eseguire un'anteprima della policy del ciclo di vita.
3. Nel riquadro di navigazione, in Registro privato, seleziona Repository.
4. Nella pagina Repository privati, seleziona un repository e utilizza il menu a discesa Operazioni per scegliere Policy del ciclo di vita.
5. Nella pagina delle regole della policy del ciclo di vita del repository, scegli Modifica regole di test, Crea regola.
6. Immetti i seguenti dettagli per ogni regola della policy del ciclo di vita di prova.

a. In Rule priority (Priorità regola), digita un numero per la priorità della regola. La priorità delle regole determina l'ordine in cui vengono applicate le regole delle policy relative al ciclo di vita. Un numero più basso indica una priorità più alta. Ad esempio, una regola con priorità 1 ha la precedenza su una regola con priorità 2.

b. In Rule description (Descrizione regola), digita una descrizione della regola della policy del ciclo di vita.

c. Per Stato immagine, scegli Taggata (corrispondenza con caratteri jolly), Taggata (corrispondenza dei prefissi), Non taggata o Qualsiasi.

> **Important**
>
> Se specifichi più tag, vengono selezionate solo le immagini che hanno tutti i tag specificati.

d. Se hai scelto Taggata (corrispondenza con caratteri jolly) per Stato immagine, allora per Specifica i tag per la corrispondenza con caratteri jolly, puoi specificare un elenco di tag di immagine con un carattere jolly (*) su cui intervenire con la tua policy del ciclo di vita. Ad esempio, se le immagini sono taggate come `prod`, `prod1`, `prod2` e così via, sarà necessario utilizzare `prod*` per specificarle tutte. Se specifichi più tag, vengono selezionate solo le immagini che hanno tutti i tag specificati.

> **Important**
>
> Esiste un limite massimo di quattro caratteri jolly (*) per stringa. Ad esempio, `["*test*1*2*3", "test*1*2*3*"]` è valido ma `["test*1*2*3*4*5*6"]` non è valido.

e. Se hai scelto Taggata (corrispondenza dei prefissi) per Stato immagine, allora per Specifica i tag per la corrispondenza dei prefissi, puoi specificare un elenco di tag di immagine su cui intervenire con la tua policy del ciclo di vita.

f. Per i criteri di corrispondenza, scegliete Giorni dalla creazione dell'immagine, Giorni dall'ultima ora di estrazione registrata, Giorni dall'archiviazione dell'immagine o Numero immagini, quindi specificate un valore.

g. Per Rule action, scegliete Scadenza o Archivia.

h. Scegli Save (Salva).

7. Per creare ulteriori regole per la policy del ciclo di vita di prova, ripeti le operazioni da 5 a 7.

8. Per eseguire l'anteprima della policy del ciclo di vita, seleziona Save and run test (Salva ed esegui test).
9. In Image matches for test lifecycle rules (Corrispondenze dell'immagine per le regole del ciclo di vita), verifica l'impatto dell'anteprima della tua policy del ciclo di vita.
10. Se i risultati ti soddisfano, seleziona Apply as lifecycle policy (Applica come policy del ciclo di vita) per creare una policy del ciclo di vita con le regole specificate. Dovresti aspettarti che, dopo aver applicato una politica del ciclo di vita, le immagini interessate scadano o vengano archiviate entro 24 ore.
11. Se non sei soddisfatto dei risultati dell'anteprima, puoi eliminare una o più regole del ciclo di vita del test e creare una o più regole per sostituirle e ripetere il test.

Creazione di una politica del ciclo di vita per un repository in Amazon ECR

Utilizza una policy del ciclo di vita per creare un set di regole con scadenza o archiviare le immagini del repository non utilizzate. Dopo aver creato una politica sul ciclo di vita, le immagini interessate vengono scadute o archiviate entro 24 ore.

> **Note**
>
> Se utilizzi la replica di Amazon ECR per creare copie di un repository in diverse regioni o account, tieni presente che una politica del ciclo di vita può intervenire solo sui repository nella regione in cui è stata creata. Pertanto, se è stata attivata la replica, potrebbe essere necessario prendere in considerazione la creazione di una policy del ciclo di vita in ogni Regione e account su cui si stanno replicando i repository.

Prerequisito

Procedura consigliata: crea un'anteprima della politica del ciclo di vita per verificare che le immagini scadute o archiviate in base alle regole della policy sul ciclo di vita corrispondano alle tue intenzioni. Per istruzioni, consulta Creazione di un'anteprima della politica del ciclo di vita in Amazon ECR.

Per creare una policy del ciclo di vita (Console di gestione AWS)

1. Apri la console Amazon ECR nei https://console.aws.amazon.com/ecr/repository.

2. Sulla barra di navigazione seleziona la regione che contiene il repository per il quale creare una policy del ciclo di vita.
3. Nel riquadro di navigazione, in Registro privato, seleziona Repository.
4. Nella pagina Repository privati, seleziona un repository e utilizza il menu a discesa Operazioni per scegliere Policy del ciclo di vita.
5. Nella pagina della policy del ciclo di vita del repository, scegli Crea regola.
6. Immetti i seguenti dettagli per la regola della policy del ciclo di vita.

    a. In Rule priority (Priorità regola), digita un numero per la priorità della regola. La priorità delle regole determina l'ordine in cui vengono applicate le regole delle policy relative al ciclo di vita. Un numero di priorità della regola più basso significa una priorità più alta. Ad esempio, una regola con priorità 1 ha la precedenza su una regola con priorità 2.

    b. In Rule description (Descrizione regola), digita una descrizione della regola della policy del ciclo di vita.

    c. Per Stato immagine, scegli Taggata (corrispondenza con caratteri jolly), Taggata (corrispondenza dei prefissi), Non taggata o Qualsiasi.

    > **Important**
    >
    > Se specifichi più tag, vengono selezionate solo le immagini che hanno tutti i tag specificati.

    d. Se hai scelto Taggata (corrispondenza con caratteri jolly) per Stato immagine, allora per Specifica i tag per la corrispondenza con caratteri jolly, puoi specificare un elenco di tag di immagine con un carattere jolly (*) su cui intervenire con la tua policy del ciclo di vita. Ad esempio, se le immagini sono taggate come `prod`, `prod1`, `prod2` e così via, sarà necessario utilizzare `prod*` per specificarle tutte. Se specifichi più tag, vengono selezionate solo le immagini che hanno tutti i tag specificati.

    > **Important**
    >
    > Esiste un limite massimo di quattro caratteri jolly (*) per stringa. Ad esempio, `["*test*1*2*3", "test*1*2*3*"]` è valido ma `["test*1*2*3*4*5*6"]` non è valido.

e. Se hai scelto Taggata (corrispondenza dei prefissi) per Stato immagine, allora per Specifica i tag per la corrispondenza dei prefissi, puoi specificare un elenco di tag di immagine su cui intervenire con la tua policy del ciclo di vita.

f. Per i criteri di corrispondenza, scegliete Giorni dalla creazione dell'immagine, Giorni dall'ultima ora di estrazione registrata, Giorni dall'archiviazione dell'immagine o Numero immagini, quindi specificate un valore.

g. Per Rule action, scegliete Scadenza o Archivia.

h. Scegli Save (Salva).

7. Per creare ulteriori regole per la policy del ciclo di vita, ripeti le operazioni da 5 a 7.

Per creare una policy del ciclo di vita (AWS CLI)

1. Ottieni il nome del repository per il quale creare la policy del ciclo di vita.

```
aws ecr describe-repositories
```

2. Creare un file locale denominato `policy.json` con il contenuto della policy del ciclo di vita. Per esempi di policy del ciclo di vita, consulta Esempi di politiche del ciclo di vita in Amazon ECR.

3. Creare una policy del ciclo di vita specificando il nome del repository e facendo riferimento al file JSON della policy del ciclo di vita creato.

```
aws ecr put-lifecycle-policy \
      --repository-name repository-name \
      --lifecycle-policy-text file://policy.json
```

Esempi di politiche del ciclo di vita in Amazon ECR

Di seguito sono riportati alcuni esempi di policy relative al ciclo di vita che mostrano la sintassi.

Per ulteriori informazioni sulle proprietà delle politiche, vedere. Proprietà delle policy del ciclo di vita in Amazon ECR Per istruzioni sulla creazione di una politica del ciclo di vita utilizzando il AWS CLI, vedere. Per creare una policy del ciclo di vita (AWS CLI)

Modello di policy del ciclo di vita

I contenuti della politica del ciclo di vita vengono valutati prima di essere associati a un repository. Di seguito viene mostrato un modello di sintassi JSON per la policy del ciclo di vita.

```
{
        "rules": [
            {
                "rulePriority": integer,
                "description": "string",
                "selection": {
                    "tagStatus": "tagged"|"untagged"|"any",
                    "tagPatternList": list<string>,
                    "tagPrefixList": list<string>,
                    "storageClass": "standard"|"archive",
                    "countType":
 "imageCountMoreThan"|"sinceImagePushed"|"sinceImagePulled"|"sinceImageTransitioned",
                    "countUnit": "string",
                    "countNumber": integer
                },
                "action": {
                    "type": "expire"|"transition",
                    "targetStorageClass": "archive"
                }
            }
        ]
    }
```

Filtraggio per età delle immagini

L'esempio seguente mostra la sintassi della policy del ciclo di vita per una policy che prevede la scadenza delle immagini con un tag che inizia con `prod` utilizzando un `tagPatternList` di `prod*` anch'esso più vecchio di `14` giorni.

```
{
    "rules": [
        {
            "rulePriority": 1,
            "description": "Expire images older than 14 days",
            "selection": {
                "tagStatus": "tagged",
                "tagPatternList": ["prod*"],
```

```
                "countType": "sinceImagePushed",
                "countUnit": "days",
                "countNumber": 14
            },
            "action": {
                "type": "expire"
            }
        }
    ]
}
```

Filtraggio per numero di immagini

L'esempio seguente mostra la sintassi della policy del ciclo di vita per una policy che mantiene una sola immagine non taggata e fa scadere tutte le altre.

```
{
    "rules": [
        {
            "rulePriority": 1,
            "description": "Keep only one untagged image, expire all others",
            "selection": {
                "tagStatus": "untagged",
                "countType": "imageCountMoreThan",
                "countNumber": 1
            },
            "action": {
                "type": "expire"
            }
        }
    ]
}
```

Filtraggio per più regole

I seguenti esempi utilizzano più regole in una policy del ciclo di vita. Vengono forniti un repository e una policy del ciclo di vita di esempio insieme a una spiegazione del risultato.

Esempio A

Contenuto del repository:

- Immagine A, Elenco tag: ["beta-1", "prod-1"], Invio: 10 giorni fa
- Immagine B, Elenco tag: ["beta-2", "prod-2"], Invio: 9 giorni fa
- Immagine C, Elenco tag: ["beta-3"], Invio: 8 giorni fa

Testo della policy del ciclo di vita:

```
{
    "rules": [
        {
            "rulePriority": 1,
            "description": "Rule 1",
            "selection": {
                "tagStatus": "tagged",
                "tagPatternList": ["prod*"],
                "countType": "imageCountMoreThan",
                "countNumber": 1
            },
            "action": {
                "type": "expire"
            }
        },
        {
            "rulePriority": 2,
            "description": "Rule 2",
            "selection": {
                "tagStatus": "tagged",
                "tagPatternList": ["beta*"],
                "countType": "imageCountMoreThan",
                "countNumber": 1
            },
            "action": {
                "type": "expire"
            }
        }
    ]
}
```

La logica di questa policy del ciclo di vita è:

- La regola 1 identifica le immagini taggate con il prefisso `prod`. Dovrebbe contrassegnare le immagini, a iniziare da quella meno recente, fino a quando rimangono una o meno immagini corrispondenti. Contrassegna l'immagine A per la scadenza.
- La regola 2 identifica le immagini taggate con il prefisso `beta`. Dovrebbe contrassegnare le immagini, a iniziare da quella meno recente, fino a quando rimangono una o meno immagini corrispondenti. Contrassegna entrambe le immagini A e B per la scadenza. Tuttavia, l'immagine A è già stata vista dalla regola 1 e, se l'immagine B scadesse, violerebbe la regola 1, pertanto viene saltata.
- Risultato: l'immagine A scade.

Esempio B

Questo è lo stesso repository dell'esempio precedente, ma l'ordine di priorità delle regole è cambiato per mostrare questo altro esito.

Contenuto del repository:

- Immagine A, Elenco tag: ["beta-1", "prod-1"], Invio: 10 giorni fa
- Immagine B, Elenco tag: ["beta-2", "prod-2"], Invio: 9 giorni fa
- Immagine C, Elenco tag: ["beta-3"], Invio: 8 giorni fa

Testo della policy del ciclo di vita:

```
{
    "rules": [
        {
            "rulePriority": 1,
            "description": "Rule 1",
            "selection": {
                "tagStatus": "tagged",
                "tagPatternList": ["beta*"],
                "countType": "imageCountMoreThan",
                "countNumber": 1
            },
            "action": {
                "type": "expire"
            }
        },
        {
```

```
            "rulePriority": 2,
            "description": "Rule 2",
            "selection": {
                "tagStatus": "tagged",
                "tagPatternList": ["prod*"],
                "countType": "imageCountMoreThan",
                "countNumber": 1
            },
            "action": {
                "type": "expire"
            }
        }
    ]
}
```

La logica di questa policy del ciclo di vita è:

- La regola 1 identifica le immagini taggate con il prefisso `beta`. Dovrebbe contrassegnare le immagini, a iniziare da quella meno recente, fino a quando rimangono una o meno immagini corrispondenti. Vede tutte e tre le immagini e contrassegna l'immagine A e l'immagine B per la scadenza.
- La regola 2 identifica le immagini taggate con il prefisso `prod`. Dovrebbe contrassegnare le immagini, a iniziare da quella meno recente, fino a quando rimangono una o meno immagini corrispondenti. Non vede alcuna immagine, perché tutte le immagini disponibili sono già viste dalla regola 1, pertanto non vengono contrassegnate altre immagini.
- Risultato: l'immagine A e l'immagine B scadono.

Filtraggio per più tag in una stessa regola

I seguenti esempi specificano la sintassi della policy del ciclo di vita per più modelli di tag in una sola regola. Vengono forniti un repository e una policy del ciclo di vita di esempio insieme a una spiegazione del risultato.

Esempio A

Quando vengono specificati più modelli di tag in una sola regola, le immagini devono corrispondere a tutti i modelli di tag elencati.

Contenuto del repository:

- Immagine A, Elenco tag: ["alpha-1"], Invio: 12 giorni fa
- Immagine B, Elenco tag: ["beta-1"], Invio: 11 giorni fa
- Immagine C, Elenco tag: ["alpha-2", "beta-2"], Invio: 10 giorni fa
- Immagine D, Elenco tag: ["alpha-3"], Invio: 4 giorni fa
- Immagine E, Elenco tag: ["beta-3"], Invio: 3 giorni fa
- Immagine F, Elenco tag: ["alpha-4", "beta-4"], Invio: 2 giorni fa

```
{
    "rules": [
        {
            "rulePriority": 1,
            "description": "Rule 1",
            "selection": {
                "tagStatus": "tagged",
                "tagPatternList": ["alpha*", "beta*"],
                "countType": "sinceImagePushed",
                "countNumber": 5,
                "countUnit": "days"
            },
            "action": {
                "type": "expire"
            }
        }
    ]
}
```

La logica di questa policy del ciclo di vita è:

- La regola 1 identifica le immagini taggate con il prefisso `alpha` e `beta`. Vede le immagini C ed F. Dovrebbe contrassegnare le immagini che hanno più di cinque giorni, in questo caso l'immagine C.
- Risultato: l'immagine C scade.

Esempio B

Il seguente esempio mostra che i tag non sono esclusivi.

Contenuto del repository:

- Immagine A, Elenco tag: ["alpha-1", "beta-1", "gamma-1"], Invio: 10 giorni fa

- Immagine B, Elenco tag: ["alpha-2", "beta-2"], Invio: 9 giorni fa
- Immagine C, Elenco tag: ["alpha-3", "beta-3", "gamma-2"], Invio: 8 giorni fa

```
{
    "rules": [
        {
            "rulePriority": 1,
            "description": "Rule 1",
            "selection": {
                "tagStatus": "tagged",
                "tagPatternList": ["alpha*", "beta*"],
                "countType": "imageCountMoreThan",
                "countNumber": 1
            },
            "action": {
                "type": "expire"
            }
        }
    ]
}
```

La logica di questa policy del ciclo di vita è:

- La regola 1 identifica le immagini taggate con il prefisso `alpha` e `beta`. Vede tutte le immagini. Dovrebbe contrassegnare le immagini, a iniziare da quella meno recente, fino a quando rimangono una o meno immagini corrispondenti. Contrassegna le immagini A e B per la scadenza.
- Risultato: l'immagine A e l'immagine B scadono.

Filtro su tutte le immagini

I seguenti esempi di policy del ciclo di vita specificano tutte le immagini con diversi filtri. Vengono forniti un repository e una policy del ciclo di vita di esempio insieme a una spiegazione del risultato.

Esempio A

Quanto segue mostra la sintassi della policy del ciclo di vita per una policy che si applica a tutte le regole ma mantiene una sola immagine e fa scadere tutte le altre.

Contenuto del repository:

- Immagine A, Elenco tag: ["alpha-1"], Invio: 4 giorni fa
- Immagine B, Elenco tag: ["beta-1"], Invio: 3 giorni fa
- Immagine C, Elenco tag: [], Invio: 2 giorni fa
- Immagine D, Elenco tag: ["alpha-2"], Invio: 1 giorno fa

```
{
    "rules": [
        {
            "rulePriority": 1,
            "description": "Rule 1",
            "selection": {
                "tagStatus": "any",
                "countType": "imageCountMoreThan",
                "countNumber": 1
            },
            "action": {
                "type": "expire"
            }
        }
    ]
}
```

La logica di questa policy del ciclo di vita è:

- La regola 1 identifica tutte le immagini. Vede le immagini A, B, C e D. Dovrebbe far scadere tutte le immagini tranne quelle più recenti. Contrassegna le immagini A, B e C per la scadenza.
- Risultato: le immagini A, B e C scadono.

Esempio B

Il seguente esempio mostra una policy del ciclo di vita che combina tutti i tipi di regole in un'unica policy.

Contenuto del repository:

- Immagine A, Elenco tag: ["alpha-1", "beta-1"], Invio: 4 giorni fa
- Immagine B, Elenco tag: [], Invio: 3 giorni fa
- Immagine C, Elenco tag: ["alpha-2"], Invio: 2 giorni fa

- Immagine D, Elenco tag: ["git hash"], Invio: 1 giorno fa
- Immagine E, Elenco tag: [], Invio: 1 giorno fa

```
{
    "rules": [
        {
            "rulePriority": 1,
            "description": "Rule 1",
            "selection": {
                "tagStatus": "tagged",
                "tagPatternList": ["alpha*"],
                "countType": "imageCountMoreThan",
                "countNumber": 1
            },
            "action": {
                "type": "expire"
            }
        },
        {
            "rulePriority": 2,
            "description": "Rule 2",
            "selection": {
                "tagStatus": "untagged",
                "countType": "sinceImagePushed",
                "countUnit": "days",
                "countNumber": 1
            },
            "action": {
                "type": "expire"
            }
        },
        {
            "rulePriority": 3,
            "description": "Rule 3",
            "selection": {
                "tagStatus": "any",
                "countType": "imageCountMoreThan",
                "countNumber": 1
            },
            "action": {
                "type": "expire"
            }
```

```
        }
    ]
}
```

La logica di questa policy del ciclo di vita è:

- La regola 1 identifica le immagini taggate con il prefisso `alpha`. Identifica le immagini A e C. Mantiene l'immagine più recente e contrassegna il resto per la scadenza. Contrassegna l'immagine A per la scadenza.
- La regola 2 identifica le immagini non taggate. Identifica le immagini B ed E. Contrassegna tutte le immagini più vecchie di un giorno per la scadenza. Contrassegna l'immagine B per la scadenza.
- La regola 3 identifica tutte le immagini. Identifica le immagini A, B, C, D ed E. Mantiene l'immagine più recente e contrassegna il resto per la scadenza. Tuttavia, non può contrassegnare le immagini A, B, C o E perché sono identificate da regole di priorità superiore. Contrassegna l'immagine D per la scadenza.
- Risultato: le immagini A, B e D scadono.

Esempi di archivio

Gli esempi seguenti mostrano le politiche del ciclo di vita che archiviano le immagini anziché eliminarle.

Archiviazione di immagini più vecchie di un determinato numero di giorni

L'esempio seguente mostra una politica del ciclo di vita che archivia immagini con tag che iniziano con e `prod` che risalgono a più di 30 giorni:

```
{
    "rules": [
        {
            "rulePriority": 1,
            "description": "Archive production images older than 30 days",
            "selection": {
                "tagStatus": "tagged",
                "tagPatternList": ["prod*"],
                "countType": "sinceImagePushed",
                "countUnit": "days",
                "countNumber": 30
            },
```

```
            "action": {
                "type": "transition",
                "targetStorageClass": "archive"
            }
        }
    ]
}
```

Archiviazione di immagini non recuperate in un numero di giorni specificato

L'esempio seguente mostra una politica del ciclo di vita che archivia le immagini che non sono state recuperate da 90 giorni:

```
{
    "rules": [
        {
            "rulePriority": 1,
            "description": "Archive images not pulled in 90 days",
            "selection": {
                "tagStatus": "any",
                "countType": "sinceImagePulled",
                "countUnit": "days",
                "countNumber": 90
            },
            "action": {
                "type": "transition",
                "targetStorageClass": "archive"
            }
        }
    ]
}
```

Combinazione di regole di archiviazione ed eliminazione

L'esempio seguente mostra una politica del ciclo di vita che archivia le immagini più vecchie di 30 giorni e quindi elimina definitivamente le immagini archiviate per più di 365 giorni:

> **Note**
>
> Le immagini archiviate hanno una durata minima di archiviazione di 90 giorni. Non è possibile configurare politiche del ciclo di vita che eliminino le immagini archiviate da meno di 90 giorni.

Se devi eliminare immagini archiviate per meno di 90 giorni, devi utilizzare l'batch-delete-imageAPI, ma ti verrà addebitata la durata minima di archiviazione di 90 giorni.

```
{
    "rules": [
        {
            "rulePriority": 1,
            "description": "Archive images older than 30 days",
            "selection": {
                "tagStatus": "any",
                "countType": "sinceImagePushed",
                "countUnit": "days",
                "countNumber": 30
            },
            "action": {
                "type": "transition",
                "targetStorageClass": "archive"
            }
        },
        {
            "rulePriority": 2,
            "description": "Delete images archived for more than 365 days",
            "selection": {
                "tagStatus": "any",
                "storageClass": "archive",
                "countType": "sinceImageTransitioned",
                "countUnit": "days",
                "countNumber": 365
            },
            "action": {
                "type": "expire"
            }
        }
    ]
}
```

Proprietà delle policy del ciclo di vita in Amazon ECR

Le politiche del ciclo di vita hanno le seguenti proprietà.

Per visualizzare esempi di politiche relative al ciclo di vita, vedere. Esempi di politiche del ciclo di vita in Amazon ECR Per istruzioni sulla creazione di una politica del ciclo di vita utilizzando il, vedere. AWS CLIPer creare una policy del ciclo di vita (AWS CLI)

Priorità delle regole

`rulePriority`

Tipo: Integer

Obbligatorio: sì

Imposta l'ordine in base al quale le regole vengono applicate, dal valore più basso a quello più alto. Viene applicata per prima una regola della politica del ciclo di vita con una priorità di1, successivamente una regola con priorità di 2 e così via. Quando aggiungi delle regole a una policy del ciclo di vita, queste devono avere un valore univoco per `rulePriority`. Non è necessario che i valori siano sequenziali tra le regole di una policy. Una regola con valore `tagStatus` di `any` deve avere il valore più elevato per `rulePriority` ed essere valutata per ultima.

Description

`description`

Tipo: string

Obbligatorio: no

(Opzionale) Descrive lo scopo di una regola nella policy del ciclo di vita.

Stato tag

`tagStatus`

Tipo: string

Obbligatorio: sì

Determina se la regola della policy del ciclo di vita che vuoi aggiungere specifica un tag per un'immagine. Le opzioni accettabili sono `tagged`, `untagged` o `any`. Se specifichi `any`,

per tutte le immagini la regola viene valutata rispetto ad esse. Se si specificatagged, è necessario specificare anche un tagPrefixList valore o un tagPatternList valore. Se si specificauntagged, è necessario omettere entrambi tagPrefixList etagPatternList.

Elenco di modelli di tag

tagPatternList

Tipo: list[string]

Obbligatorio: sì, se tagStatus è impostato su Taggato e tagPrefixList non è specificato

Quando si crea una policy del ciclo di vita per le immagini con tag, è consigliabile utilizzare tagPatternList per specificare che i tag devono scadere. È necessario specificare un elenco separato da virgole di modelli per i tag di immagini che possono contenere caratteri jolly (*) sul quale lavorare con la policy del ciclo di vita. Ad esempio, se le immagini sono taggate come prod, prod1, prod2 e così via, utilizzerai l'elenco di modelli di tag prod* per specificarle tutte. Se specifichi più tag, vengono selezionate solo le immagini che hanno tutti i tag specificati.

> **Important**
>
> Esiste un limite massimo di quattro caratteri jolly (*) per stringa. Ad esempio, ["*test*1*2*3", "test*1*2*3*"] è valido ma ["test*1*2*3*4*5*6"] non è valido.

Elenco prefissi tag

tagPrefixList

Tipo: list[string]

Obbligatorio: sì, se tagStatus è impostato su taggato e tagPatternList non è specificato

Utilizzato solo se è stato specificato "tagStatus": "tagged" e non si sta specificando un tagPatternList. Devi specificare un elenco separato da virgole di prefissi per i tag delle immagini sul quale operare con la tua policy del ciclo di vita. Ad esempio, se le tue immagini sono taggate come prod, prod1, prod2 e così via, utilizzerai il prefisso di tag prod per specificarle tutte. Se specifichi più tag, vengono selezionate solo le immagini che hanno tutti i tag specificati.

Classe di storage

`storageClass`

Tipo: stringa

Obbligatorio: sì, se lo è `countType` `sinceImageTransitioned`

La regola selezionerà solo le immagini di questa classe di archiviazione. Quando si utilizza un valore `countType` di `imageCountMoreThansinceImagePushed`, o`sinceImagePulled`, l'unico valore supportato è`standard`. Quando si utilizza un tipo di conteggio di`sinceImageTransitioned`, questo è obbligatorio e l'unico valore supportato è`archive`. Se lo ometti, `standard` verrà utilizzato il valore di.

Tipo di conteggio

`countType`

Tipo: string

Obbligatorio: sì

Specifica un tipo di conteggio da applicare alle immagini.

Se `countType` è impostato su `imageCountMoreThan`, devi specificare inoltre `countNumber` per creare una regola che impone un limite al numero di immagini presenti nel tuo repository. Se `countType` è impostato su `sinceImagePushedsinceImagePulled`, o`sinceImageTransitioned`, si specifica `countUnit` e si specifica anche un limite di tempo `countNumber` per le immagini presenti nel repository.

Unità conteggio

`countUnit`

Tipo: stringa

Obbligatorio: sì, solo se `countType` è impostato su `sinceImagePushedsinceImagePulled`, o `sinceImageTransitioned`

Specifica un unità di conteggio di `days` per definirla come unità di tempo, oltre a `countNumber`, che corrisponde al numero di giorni.

Deve essere specificato solo quando `countType` è `sinceImagePushedsinceImagePulled`, o`sinceImageTransitioned`; si verificherà un errore se si specifica un'unità di conteggio quando `countType` è presente un qualsiasi altro valore.

Count number (Numero conteggio)

`countNumber`

Tipo: Integer

Obbligatorio: sì

Specifica un numero conteggio. I valori accettabili sono numeri interi positivi (`0` non è un valore accettato).

Se il `countType` utilizzato è `imageCountMoreThan`, allora il valore è il numero massimo di immagini che desideri conservare nel tuo repository. Se il `countType` utilizzato è `sinceImagePushed`, allora il valore è il limite di età massimo per le tue immagini. Se `countType` utilizzato è`sinceImagePulled`, il valore è il numero massimo di giorni trascorsi dall'ultima volta che l'immagine è stata estratta. Se `countType` utilizzato è`sinceImageTransitioned`, il valore è il numero massimo di giorni trascorsi dall'archiviazione dell'immagine.

Azione

`type`

Tipo: stringa

Obbligatorio: sì

Specifica un tipo di azione. I valori supportati sono `expire` (per eliminare le immagini) e `transition` (per spostare le immagini nell'archivio).

`targetStorageClass`

Tipo: stringa

Obbligatorio: sì, se lo `type` è `transition`

La classe di storage a cui si desidera applicare la politica del ciclo di vita per la transizione dell'immagine. `archive`è l'unico valore supportato.

Esclusioni relative agli aggiornamenti a tempo pieno

Amazon ECR aggiorna il `LastRecordedPullTime` timestamp di ogni estrazione ad eccezione delle estrazioni effettuate da Inspector. AWS Le esclusioni relative agli aggiornamenti pull-time consentono di specificare un ruolo IAM ARNs che non deve aggiornare i tempi di recupero delle immagini quando estraggono immagini, come quelle eseguite da scanner di terze parti (come Crowdstrike, Snyk e Trivy). Ciò è utile per le immagini utilizzate a scopo di test o per CI/CD scopi in cui non si desidera che il pull time influenzi le decisioni relative alle politiche relative al ciclo di vita.

Quando un ruolo nell'elenco di esclusione recupera un'immagine, il tempo di recupero rimane invariato. Qualsiasi altro ruolo continua ad aggiornare il pull time (comportamento attuale). Puoi configurare fino a 100 esclusioni per account.

Gestione delle esclusioni relative agli aggiornamenti a tempo pieno

Per gestire le esclusioni degli aggiornamenti pull-time, sono necessarie le seguenti autorizzazioni IAM:

- `ecr:CreatePullTimeUpdateExclusion`— Concede l'autorizzazione ad aggiungere un ruolo ARN all'elenco di esclusione.
- `ecr:DeletePullTimeUpdateExclusion`— concede l'autorizzazione a rimuovere l'ARN di un ruolo dall'elenco di esclusione.
- `ecr:ListPullTimeUpdateExclusions`— Concede l'autorizzazione a elencare tutti i ruoli ARNs nell'elenco di esclusione.

> **Note**
>
> Non è necessaria `iam:PassRole` l'autorizzazione. Amazon ECR non si assume il ruolo di eseguire un'azione; utilizza solo la configurazione di esclusione ARNs per determinare se il tempo di pull dell'immagine deve essere aggiornato.

Puoi gestire le esclusioni degli aggiornamenti pull-time utilizzando la console Amazon ECR o la CLI. AWS

Console di gestione AWS

Per gestire le esclusioni relative agli aggiornamenti pull-time ()Console di gestione AWS

1. Apri la console Amazon ECR su private-registry/repository https://console.aws.amazon.com/ecr/
2. Dalla barra di navigazione, scegli la regione.
3. Nel riquadro di navigazione, scegli Registro privato, Caratteristiche e impostazioni, quindi scegli Esclusioni dagli aggiornamenti a tempo pieno.
4. Per aggiungere un'esclusione, scegli Aggiungi esclusione, inserisci il ruolo ARN, quindi scegli Aggiungi.
5. Per rimuovere un'esclusione, seleziona il ruolo ARN dall'elenco e scegli Elimina.
6. Per visualizzare tutte le esclusioni, l'elenco mostra tutti i ruoli configurati. ARNs

AWS CLI

Per creare un'esclusione per gli aggiornamenti a tempo pieno

- Usa il create-pull-time-update-exclusion comando per aggiungere un ruolo ARN all'elenco di esclusione:

```
aws ecr create-pull-time-update-exclusion \
    --role-arn arn:aws:iam::123456789012:role/scanner-role
```

Il comando restituisce l'ARN del ruolo e il timestamp di creazione:

```
{
   "roleArn": "arn:aws:iam::123456789012:role/scanner-role",
   "createdAt": 1745531331.0
}
```

Per eliminare un'esclusione di aggiornamento pull-time

- Usa il delete-pull-time-update-exclusion comando per rimuovere l'ARN di un ruolo dall'elenco di esclusione:

```
aws ecr delete-pull-time-update-exclusion \
```

```
    --role-arn arn:aws:iam::123456789012:role/scanner-role
```

Il comando restituisce il ruolo ARN che è stato eliminato:

```
{
  "roleArn": "arn:aws:iam::123456789012:role/scanner-role"
}
```

Per elencare le esclusioni relative agli aggiornamenti a tempo pieno

1. Usa il list-pull-time-update-exclusions comando per elencare tutti i ruoli ARNs nell'elenco di esclusione:

   ```
   aws ecr list-pull-time-update-exclusions
   ```

   Se non è configurata alcuna esclusione, il comando restituisce un elenco vuoto:

   ```
   {
      "pullTimeUpdateExclusions": []
   }
   ```

   Se le esclusioni sono configurate, il comando restituisce l'elenco dei ruoli: ARNs

   ```
   {
      "pullTimeUpdateExclusions": [
           "arn:aws:iam::123456789012:role/security-role"
       ]
   }
   ```

2. Per impaginare i risultati, usa i parametri `--max-results` and`--next-token`:

   ```
   aws ecr list-pull-time-update-exclusions \
       --max-results 4
   ```

   Il comando restituisce fino al numero di risultati specificato e `nextToken` se sono disponibili più risultati:

   ```
   {
      "pullTimeUpdateExclusions": [
   ```

```
        "arn:aws:iam::123456789012:role/security-role1",
        "arn:aws:iam::123456789012:role/security-role2",
        "arn:aws:iam::123456789012:role/security-role3",
        "arn:aws:iam::123456789012:role/security-role4"
    ],
    "nextToken": "ukD72mdD/mC8b5xV3susmJzzaTgp3hKwR9nRUW1yZZ79..."
}
```

Per recuperare la pagina successiva di risultati, usa la `nextToken` risposta precedente:

```
aws ecr list-pull-time-update-exclusions \
    --max-results 4 \
    --next-token ukD72mdD/mC8b5xV3susmJzzaTgp3hKwR9nRUW1yZZ79...
```

Considerazioni relative alle esclusioni relative agli aggiornamenti a tempo pieno

Quando utilizzi le esclusioni relative agli aggiornamenti pull-time, tieni presente quanto segue:

- La dimensione predefinita della pagina per le esclusioni dalle inserzioni è 100. È possibile utilizzare l'impaginazione con parametri `maxResults` e `nextToken`.
- Sono accettati solo ruoli IAM validi ARNs nel formato ARN corretto.
- Se provi a creare un'esclusione già esistente, riceverai un `ExclusionAlreadyExistsException` errore. Se provi a eliminare un'esclusione che non esiste, riceverai un `ExclusionNotFoundException` errore.

Sicurezza in Amazon Elastic Container Registry

La sicurezza del cloud AWS è la massima priorità. In qualità di AWS cliente, puoi beneficiare di un data center e di un'architettura di rete progettati per soddisfare i requisiti delle organizzazioni più sensibili alla sicurezza.

La sicurezza è una responsabilità condivisa tra AWS te e te. Il modello di responsabilità condivisa descrive questo come sicurezza del cloud e sicurezza nel cloud:

- Sicurezza del cloud: AWS è responsabile della protezione dell'infrastruttura che gestisce AWS i servizi nel AWS cloud. AWS ti fornisce anche servizi che puoi utilizzare in modo sicuro. I revisori di terze parti testano e verificano regolarmente l'efficacia della sicurezza come parte dei programmi di conformitàAWS. Per ulteriori informazioni sui programmi di conformità che si applicano ad Amazon ECR, consulta Servizi AWS coperti dal programma di conformità.
- Sicurezza nel cloud: la tua responsabilità è determinata dal AWS servizio che utilizzi. L'utente è anche responsabile di altri fattori, tra cui la riservatezza dei dati, i requisiti della propria azienda e le leggi e normative vigenti.

Questa documentazione aiuta a comprendere come applicare il modello di responsabilità condivisa quando si utilizza Amazon ECR. Gli argomenti seguenti descrivono come configurare Amazon ECR per soddisfare gli obiettivi di sicurezza e conformità. Scopri anche come utilizzare altri AWS servizi che ti aiutano a monitorare e proteggere le tue risorse Amazon ECR.

Argomenti

Identity and Access Management per Amazon Elastic Container Registry

AWS Identity and Access Management (IAM) è un servizio Servizio AWS che aiuta un amministratore a controllare in modo sicuro l'accesso alle AWS risorse. Gli amministratori IAM controllano chi è

autenticato (accesso effettuato) e autorizzato (dispone di autorizzazioni) a utilizzare risorse Amazon ECR. IAM è un software Servizio AWS che puoi utilizzare senza costi aggiuntivi.

Argomenti

Destinatari

Il modo in cui utilizzi AWS Identity and Access Management (IAM) varia in base al tuo ruolo:

- Utente del servizio: richiedi le autorizzazioni all'amministratore se non riesci ad accedere alle funzionalità (consulta Risoluzione dei problemi di Identity and Access Management per Amazon Elastic Container Registry)
- Amministratore del servizio: determina l'accesso degli utenti e invia le richieste di autorizzazione (consulta Come funziona Amazon Elastic Container Registry con IAM)
- Amministratore IAM: scrivi policy per gestire l'accesso (consulta Esempi di policy basate su Identità di Amazon Elastic Container Registry)

Autenticazione con identità

L'autenticazione è il modo in cui accedi AWS utilizzando le tue credenziali di identità. Devi autenticarti come utente IAM o assumendo un ruolo IAM. Utente root dell'account AWS

Puoi accedere come identità federata utilizzando credenziali provenienti da una fonte di identità come AWS IAM Identity Center (IAM Identity Center), autenticazione Single Sign-On o credenziali. Google/

Facebook Per ulteriori informazioni sull'accesso, consulta Come accedere all' Account AWS nella Guida per l'utente di Accedi ad AWS .

Per l'accesso programmatico, AWS fornisce un SDK e una CLI per firmare crittograficamente le richieste. Per ulteriori informazioni, consulta AWS Signature Version 4 per le richieste API nella Guida per l'utente IAM.

Account AWS utente root

Quando si crea un Account AWS, si inizia con un'identità di accesso denominata utente Account AWS root che ha accesso completo a tutte Servizi AWS le risorse. Si consiglia vivamente di non utilizzare l'utente root per le attività quotidiane. Per le attività che richiedono le credenziali come utente root, consulta Attività che richiedono le credenziali dell'utente root nella Guida per l'utente di IAM.

Utenti e gruppi IAM

Un utente IAM è una identità che dispone di autorizzazioni specifiche per una singola persona o applicazione. Consigliamo di utilizzare credenziali temporanee invece di utenti IAM con credenziali a lungo termine. Per ulteriori informazioni, consulta Richiedere agli utenti umani di utilizzare la federazione con un provider di identità per accedere AWS utilizzando credenziali temporanee nella Guida per l'utente IAM.

Un gruppo IAM specifica una raccolta di utenti IAM e semplifica la gestione delle autorizzazioni per gestire gruppi di utenti di grandi dimensioni. Per ulteriori informazioni, consulta Casi d'uso per utenti IAM nella Guida per l'utente di IAM.

Ruoli IAM

Un ruolo IAM è un'identità con autorizzazioni specifiche che fornisce credenziali temporanee. Puoi assumere un ruolo passando da un ruolo utente a un ruolo IAM (console) o chiamando un'operazione AWS CLI o AWS API. Per ulteriori informazioni, consulta Metodi per assumere un ruolo nella Guida per l'utente IAM.

I ruoli IAM sono utili per l'accesso federato degli utenti, le autorizzazioni utente IAM temporanee, l'accesso tra account, l'accesso tra servizi e le applicazioni in esecuzione su Amazon. EC2 Per maggiori informazioni, consultare Accesso a risorse multi-account in IAM nella Guida per l'utente IAM.

Gestione dell'accesso tramite policy

Puoi controllare l'accesso AWS creando policy e collegandole a identità o risorse. AWS Una policy definisce le autorizzazioni quando è associata a un'identità o a una risorsa. AWS valuta queste politiche quando un preside effettua una richiesta. La maggior parte delle politiche viene archiviata AWS come documenti JSON. Per maggiori informazioni sui documenti delle policy JSON, consulta Panoramica delle policy JSON nella Guida per l'utente IAM.

Utilizzando le policy, gli amministratori specificano chi ha accesso a cosa definendo quale principale può eseguire azioni su quali risorse e in quali condizioni.

Per impostazione predefinita, utenti e ruoli non dispongono di autorizzazioni. Un amministratore IAM crea le policy IAM e le aggiunge ai ruoli, che gli utenti possono quindi assumere. Le policy IAM definiscono le autorizzazioni indipendentemente dal metodo utilizzato per eseguirle.

Policy basate sull'identità

Le policy basate su identità sono documenti di policy di autorizzazione JSON che è possibile collegare a un'identità (utente, gruppo o ruolo). Tali policy controllano le operazioni autorizzate per l'identità, nonché le risorse e le condizioni in cui possono essere eseguite. Per informazioni su come creare una policy basata su identità, consultare Definizione di autorizzazioni personalizzate IAM con policy gestite dal cliente nella Guida per l'utente IAM.

Le policy basate su identità possono essere policy in linea (con embedding direttamente in una singola identità) o policy gestite (policy autonome collegate a più identità). Per informazioni su come scegliere tra una policy gestita o una policy inline, consulta Scegliere tra policy gestite e policy in linea nella Guida per l'utente di IAM.

Policy basate sulle risorse

Le policy basate su risorse sono documenti di policy JSON che è possibile collegare a una risorsa. Gli esempi includono le policy di trust dei ruoli IAM e le policy dei bucket di Amazon S3. Nei servizi che supportano policy basate sulle risorse, gli amministratori dei servizi possono utilizzarli per controllare l'accesso a una risorsa specifica. In una policy basata sulle risorse è obbligatorio specificare un'entità principale.

Le policy basate sulle risorse sono policy inline che si trovano in tale servizio. Non è possibile utilizzare le policy AWS gestite di IAM in una policy basata sulle risorse.

Altri tipi di policy

AWS supporta tipi di policy aggiuntivi che possono impostare le autorizzazioni massime concesse dai tipi di policy più comuni:

- Limiti delle autorizzazioni: imposta il numero massimo di autorizzazioni che una policy basata su identità ha la possibilità di concedere a un'entità IAM. Per ulteriori informazioni, consulta Limiti delle autorizzazioni per le entità IAM nella Guida per l'utente di IAM.
- Politiche di controllo del servizio (SCPs): specificano le autorizzazioni massime per un'organizzazione o un'unità organizzativa in. AWS Organizations Per ulteriori informazioni, consultare Policy di controllo dei servizi nella Guida per l'utente di AWS Organizations .
- Politiche di controllo delle risorse (RCPs): imposta le autorizzazioni massime disponibili per le risorse nei tuoi account. Per ulteriori informazioni, consulta Politiche di controllo delle risorse (RCPs) nella Guida per l'AWS Organizations utente.
- Policy di sessione: policy avanzate passate come parametro quando si crea una sessione temporanea per un ruolo o un utente federato. Per maggiori informazioni, consultare Policy di sessione nella Guida per l'utente IAM.

Più tipi di policy

Quando a una richiesta si applicano più tipi di policy, le autorizzazioni risultanti sono più complicate da comprendere. Per scoprire come si AWS determina se consentire o meno una richiesta quando sono coinvolti più tipi di policy, consulta Logica di valutazione delle policy nella IAM User Guide.

Come funziona Amazon Elastic Container Registry con IAM

Prima di utilizzare IAM per gestire l'accesso ad Amazon ECR, è necessario comprendere quali funzioni IAM sono disponibili per l'uso con Amazon ECR. Per avere una visione di alto livello di come Amazon ECR e altri AWS servizi funzionano con IAM, consulta AWS Services That Work with IAM nella IAM User Guide.

Argomenti

- Policy basate su Identità Amazon ECR
- Policy basate sulle risorse Amazon ECR
- Autorizzazione basata sui tag Amazon ECR
- Ruoli IAM Amazon ECR

Policy basate su Identità Amazon ECR

Con le policy basate sull'identità di IAM, è possibile specificare quali operazioni e risorse sono consentite o respinte, nonché le condizioni in base alle quali le operazioni sono consentite o respinte. Amazon ECR supporta specifiche operazioni, risorse e chiavi di condizione. Per informazioni su tutti gli elementi utilizzati in una policy JSON, consulta Documentazione di riferimento degli elementi delle policy JSON IAM nella Guida per l'utente IAM.

Azioni

Gli amministratori possono utilizzare le policy AWS JSON per specificare chi ha accesso a cosa. In altre parole, quale entità principale può eseguire operazioni su quali risorse e in quali condizioni.

L'elemento `Action` di una policy JSON descrive le operazioni che è possibile utilizzare per consentire o negare l'accesso in una policy. Includere le operazioni in una policy per concedere le autorizzazioni a eseguire l'operazione associata.

Le operazioni delle policy in Amazon ECR utilizzano il seguente prefisso prima dell'operazione: `ecr:`. Ad esempio, per concedere a qualcuno l'autorizzazione per creare un repository ECR mediante l'operazione API `CreateRepository` Amazon ECR, includi l'operazione `ecr:CreateRepository` nella policy. Le istruzioni della policy devono includere un elemento `Action` o `NotAction`. Amazon ECR definisce un proprio set di operazioni che descrivono le attività che puoi eseguire con quel servizio.

Per specificare più azioni in una sola istruzione, separa ciascuna di esse con una virgola come mostrato di seguito:

```
"Action": [
      "ecr:action1",
      "ecr:action2"
```

È possibile specificare più azioni tramite caratteri jolly (*). Ad esempio, per specificare tutte le azioni che iniziano con la parola `Describe`, includi la seguente azione:

```
"Action": "ecr:Describe*"
```

Per visualizzare un elenco di operazioni Amazon ECR, consulta Operazioni, risorse e chiavi di condizione per Amazon Elastic Container Registry nella Guida per l'utente di IAM.

Resources

Gli amministratori possono utilizzare le policy AWS JSON per specificare chi ha accesso a cosa. In altre parole, quale entità principale può eseguire operazioni su quali risorse e in quali condizioni.

L'elemento JSON `Resource` della policy specifica l'oggetto o gli oggetti ai quali si applica l'operazione. Come best practice, specifica una risorsa utilizzando il suo nome della risorsa Amazon (ARN). Per le azioni che non supportano le autorizzazioni a livello di risorsa, si utilizza un carattere jolly (*) per indicare che l'istruzione si applica a tutte le risorse.

```
"Resource": "*"
```

La risorsa del repository Amazon ECR dispone del seguente ARN:

```
arn:${Partition}:ecr:${Region}:${Account}:repository/${Repository-name}
```

Per ulteriori informazioni sul formato di ARNs, consulta Amazon Resource Names (ARNs) e AWS Service Namespaces.

Ad esempio, per specificare il repository `my-repo` nella regione `us-east-1` nell'istruzione, utilizza il seguente ARN:

```
"Resource": "arn:aws:ecr:us-east-1:123456789012:repository/my-repo"
```

Per specificare tutti i repository che appartengono a un account specifico, utilizza il carattere jolly (*):

```
"Resource": "arn:aws:ecr:us-east-1:123456789012:repository/*"
```

Per specificare più risorse in una singola istruzione, separale con virgole. ARNs

```
"Resource": [
      "resource1",
      "resource2"
```

Per visualizzare un elenco dei tipi di risorse Amazon ECR e relativi ARNs, consulta Resources Defined by Amazon Elastic Container Registry nella IAM User Guide. Per informazioni sulle operazioni con cui è possibile specificare l'ARN di ogni risorsa, consultare Operazioni definite da Amazon Elastic Container Registry.

Chiavi di condizione

Gli amministratori possono utilizzare le policy AWS JSON per specificare chi ha accesso a cosa. In altre parole, quale entità principale può eseguire operazioni su quali risorse e in quali condizioni.

L'elemento `Condition` specifica quando le istruzioni vengono eseguite in base a criteri definiti. È possibile compilare espressioni condizionali che utilizzano operatori di condizione, ad esempio uguale a o minore di, per soddisfare la condizione nella policy con i valori nella richiesta. Per visualizzare tutte le chiavi di condizione AWS globali, consulta le chiavi di contesto delle condizioni AWS globali nella Guida per l'utente IAM.

Amazon ECR definisce il proprio set di chiavi di condizione e supporta anche l'uso di alcune chiavi di condizione globali. Per vedere tutte le chiavi di condizione AWS globali, consulta AWS Global Condition Context Keys nella Guida per l'utente IAM.

La maggior parte delle operazioni di Amazon ECR supporta le chiavi di condizione `aws:ResourceTag` e `ecr:ResourceTag`. Per ulteriori informazioni, consulta Uso del controllo degli accessi basato su tag.

Per visualizzare un elenco di chiavi di condizione Amazon ECR, consulta Chiave di condizione definita da Amazon Elastic Container Registry nella Guida per l'utente di IAM. Per informazioni su operazioni e risorse con cui è possibile utilizzare una chiave di condizione, consultare Operazioni definite da Amazon Elastic Container Registry.

Esempi

Per visualizzare esempi di policy basate su identità Amazon ECR, consultare Esempi di policy basate su Identità di Amazon Elastic Container Registry.

Policy basate sulle risorse Amazon ECR

Le policy basate su risorse sono documenti di policy JSON che specificano le operazioni che possono essere eseguite da un'entità principale specificata su una risorsa Amazon ECR e in base a quali condizioni. Amazon ECR supporta policy di autorizzazione basate sulle risorse per i repository Amazon ECR. Le policy basate su risorse consentono di concedere l'autorizzazione all'utilizzo ad altri account per ogni risorsa. Puoi inoltre utilizzare una policy basata su risorse per consentire a un servizio AWS di accedere ai repository Amazon ECR.

Per consentire l'accesso a più account, è possibile specificare un intero account o entità IAM in un altro account come entità principale in una policy basata su risorse. L'aggiunta di un'entità principale

multi-account a una policy basata sulle risorse rappresenta solo una parte della relazione di trust. Quando il principale e la risorsa si trovano in AWS account diversi, è inoltre necessario concedere all'entità principale l'autorizzazione ad accedere alla risorsa. Concedi l'autorizzazione collegando una policy basata sull'identità all'entità. Tuttavia, se una policy basata sulle risorse concede l'accesso a una persona principale nello stesso account, non sono necessarie autorizzazioni aggiuntive per il repository Amazon ECR nella policy basata sull'identità. Per ulteriori informazioni, consulta Differenza tra i ruoli IAM e le policy basate su risorse nella Guida per l'utente IAM.

Il servizio Amazon ECR supporta solo un tipo di policy basata su risorse detta policy di repository, che è collegata a un repository. Questa policy definisce quali entità principali (account, utenti, ruoli e utenti federati) possono eseguire operazioni sul repository. Per informazioni su come collegare una policy basata su risorse a un repository, consulta Politiche di archivio privato in Amazon ECR.

> **Note**
>
> In una politica di repository Amazon ECR, l'elemento della policy `Sid` supporta caratteri e spaziature aggiuntivi non supportati nelle policy IAM.

Esempi

Per visualizzare esempi di policy basate su risorse Amazon ECR, consulta Esempi di policy relative agli archivi privati in Amazon ECR.

Autorizzazione basata sui tag Amazon ECR

Puoi collegare i tag alle risorse Amazon ECR o inoltrarli in una richiesta ad Amazon ECR. Per controllare l'accesso basato su tag, fornire informazioni sui tag nell'elemento condizione di una policy utilizzando le chiavi di condizione `ecr:ResourceTag/`*`key-name`*, `aws:RequestTag/`*`key-name`* o `aws:TagKeys`. Per ulteriori informazioni sul tagging delle risorse di Amazon ECR, consultare Taggare un repository privato in Amazon ECR.

Per visualizzare una policy basata sulle identità di esempio per limitare l'accesso a una risorsa basata su tag su tale risorsa, consulta Uso del controllo degli accessi basato su tag.

Ruoli IAM Amazon ECR

Un ruolo IAM è un'entità all'interno del tuo account che dispone di autorizzazioni specifiche. AWS

Utilizzo di credenziali temporanee con Amazon ECR

È possibile utilizzare credenziali temporanee per effettuare l'accesso con la federazione, assumere un ruolo IAM o un ruolo multi-account. Ottieni credenziali di sicurezza temporanee chiamando operazioni AWS STS API come AssumeRoleo. GetFederationToken

Amazon ECR supporta l'uso di credenziali temporanee.

Ruoli collegati ai servizi

I ruoli collegati ai AWS servizi consentono ai servizi di accedere alle risorse di altri servizi per completare un'azione per conto dell'utente. I ruoli collegati ai servizi sono visualizzati nell'account IAM e sono di proprietà del servizio. Un amministratore IAM può visualizzare le autorizzazioni per i ruoli collegati ai servizi, ma non può modificarle.

Amazon ECR supporta i ruoli collegati ai servizi. Per ulteriori informazioni, consulta Utilizzo di ruoli collegati ai servizi per Amazon ECR.

Esempi di policy basate su Identità di Amazon Elastic Container Registry

Per impostazione predefinita, gli utenti e i ruoli non dispongono dell'autorizzazione per creare o modificare risorse Amazon ECR. Per concedere agli utenti l'autorizzazione a eseguire azioni sulle risorse di cui hanno bisogno, un amministratore IAM può creare policy IAM.

Per informazioni su come creare una policy basata su identità IAM utilizzando questi documenti di policy JSON di esempio, consulta Creazione di policy IAM (console) nella Guida per l'utente di IAM.

Per dettagli sulle azioni e sui tipi di risorse definiti da Amazon ECR, incluso il formato di ARNs per ogni tipo di risorsa, consulta Azioni, risorse e chiavi di condizione per Amazon Elastic Container Registry nel Service Authorization Reference.

Per informazioni su come creare una policy basata su identità IAM utilizzando questi documenti di policy JSON di esempio, consulta Creazione di policy nella scheda JSON nella Guida per l'utente IAM.

Argomenti

- Best practice delle policy
- Utilizzo della console Amazon ECR
- Consenti agli utenti di visualizzare le loro autorizzazioni
- Accesso a un repository Amazon ECR

Best practice delle policy

Le policy basate su identità determinano se qualcuno può creare, accedere o eliminare risorse Amazon ECR nel tuo account. Queste azioni possono comportare costi aggiuntivi per l' Account AWS. Quando si creano o modificano policy basate sull'identità, seguire queste linee guida e raccomandazioni:

- Inizia con le politiche AWS gestite e passa alle autorizzazioni con privilegi minimi: per iniziare a concedere autorizzazioni a utenti e carichi di lavoro, utilizza le politiche AWS gestite che concedono le autorizzazioni per molti casi d'uso comuni. Sono disponibili nel tuo. Account AWS Ti consigliamo di ridurre ulteriormente le autorizzazioni definendo politiche gestite dai AWS clienti specifiche per i tuoi casi d'uso. Per maggiori informazioni, consulta Policy gestite da AWS o Policy gestite da AWS per le funzioni dei processi nella Guida per l'utente di IAM.
- Applicazione delle autorizzazioni con privilegio minimo - Quando si impostano le autorizzazioni con le policy IAM, concedere solo le autorizzazioni richieste per eseguire un'attività. È possibile farlo definendo le azioni che possono essere intraprese su risorse specifiche in condizioni specifiche, note anche come autorizzazioni con privilegio minimo. Per maggiori informazioni sull'utilizzo di IAM per applicare le autorizzazioni, consulta Policy e autorizzazioni in IAM nella Guida per l'utente di IAM.
- Condizioni d'uso nelle policy IAM per limitare ulteriormente l'accesso - Per limitare l'accesso ad azioni e risorse è possibile aggiungere una condizione alle policy. Ad esempio, è possibile scrivere una condizione di policy per specificare che tutte le richieste devono essere inviate utilizzando SSL. Puoi anche utilizzare le condizioni per concedere l'accesso alle azioni del servizio se vengono utilizzate tramite uno specifico Servizio AWS, ad esempio CloudFormation. Per maggiori informazioni, consultare la sezione Elementi delle policy JSON di IAM: condizione nella Guida per l'utente di IAM.
- Utilizzo dello strumento di analisi degli accessi IAM per convalidare le policy IAM e garantire autorizzazioni sicure e funzionali - Lo strumento di analisi degli accessi IAM convalida le policy nuove ed esistenti in modo che aderiscano al linguaggio (JSON) della policy IAM e alle best practice di IAM. Lo strumento di analisi degli accessi IAM offre oltre 100 controlli delle policy e consigli utili per creare policy sicure e funzionali. Per maggiori informazioni, consultare Convalida delle policy per il Sistema di analisi degli accessi IAM nella Guida per l'utente di IAM.
- Richiedi l'autenticazione a più fattori (MFA): se hai uno scenario che richiede utenti IAM o un utente root nel Account AWS tuo, attiva l'MFA per una maggiore sicurezza. Per richiedere la MFA quando vengono chiamate le operazioni API, aggiungere le condizioni MFA alle policy. Per maggiori informazioni, consultare Protezione dell'accesso API con MFA nella Guida per l'utente di IAM.

Per maggiori informazioni sulle best practice in IAM, consulta Best practice di sicurezza in IAM nella Guida per l'utente di IAM.

Utilizzo della console Amazon ECR

Per accedere alla console Amazon Elastic Container Registry, è necessario disporre di un set di autorizzazioni minimo. Queste autorizzazioni devono consentirti di elencare e visualizzare i dettagli sulle risorse Amazon ECR nel tuo AWS account. Se crei una policy basata sull'identità più restrittiva rispetto alle autorizzazioni minime richieste, la console non funzionerà nel modo previsto per le entità (utenti o ruoli) associate a tale policy.

Per garantire che tali entità possano ancora utilizzare la console Amazon ECR, aggiungi la policy `AmazonEC2ContainerRegistryReadOnly` AWS gestita alle entità. Per ulteriori informazioni, consulta Aggiunta di autorizzazioni a un utente nella Guida per l'utente di IAM:

Per vedere le autorizzazioni per questa policy, consulta AmazonElasticContainerRegistryPublicReadOnly nella Guida di riferimento sulle policy gestite da AWS .

Non è necessario consentire autorizzazioni minime di console per gli utenti che effettuano chiamate solo verso AWS CLI o l' AWS API. Al contrario, è possibile accedere solo alle operazioni che soddisfano l'operazione API che stai cercando di eseguire.

Consenti agli utenti di visualizzare le loro autorizzazioni

Questo esempio mostra in che modo è possibile creare una policy che consente agli utenti IAM di visualizzare le policy inline e gestite che sono collegate alla relativa identità utente. Questa politica include le autorizzazioni per completare questa azione sulla console o utilizzando l'API o a livello di codice. AWS CLI AWS

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "ViewOwnUserInfo",
            "Effect": "Allow",
            "Action": [
                "iam:GetUserPolicy",
                "iam:ListGroupsForUser",
                "iam:ListAttachedUserPolicies",
                "iam:ListUserPolicies",
```

```
                "iam:GetUser"
            ],
            "Resource": ["arn:aws:iam::*:user/${aws:username}"]
        },
        {
            "Sid": "NavigateInConsole",
            "Effect": "Allow",
            "Action": [
                "iam:GetGroupPolicy",
                "iam:GetPolicyVersion",
                "iam:GetPolicy",
                "iam:ListAttachedGroupPolicies",
                "iam:ListGroupPolicies",
                "iam:ListPolicyVersions",
                "iam:ListPolicies",
                "iam:ListUsers"
            ],
            "Resource": "*"
        }
    ]
}
```

Accesso a un repository Amazon ECR

In questo esempio, vuoi concedere a un utente del tuo AWS account l'accesso a uno dei tuoi repository Amazon ECR,. `my-repo` Si desidera anche consentire all'utente di inviare, estrarre ed elencare le immagini.

JSON

```
{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Sid":"GetAuthorizationToken",
         "Effect":"Allow",
         "Action":[
            "ecr:GetAuthorizationToken"
         ],
         "Resource":"*"
      },
      {
```

```
        "Sid":"ManageRepositoryContents",
        "Effect":"Allow",
        "Action":[
               "ecr:BatchCheckLayerAvailability",
               "ecr:GetDownloadUrlForLayer",
               "ecr:GetRepositoryPolicy",
               "ecr:DescribeRepositories",
               "ecr:ListImages",
               "ecr:DescribeImages",
               "ecr:BatchGetImage",
               "ecr:InitiateLayerUpload",
               "ecr:UploadLayerPart",
               "ecr:CompleteLayerUpload",
               "ecr:PutImage"
        ],
        "Resource":"arn:aws:ecr:us-east-1:123456789012:repository/my-repo"
      }
   ]
}
```

Uso del controllo degli accessi basato su tag

L'azione dell'`CreateRepository`API Amazon ECR consente di specificare i tag quando si crea il repository. Per ulteriori informazioni, consulta Taggare un repository privato in Amazon ECR.

Per consentire agli utenti di applicare tag ai repository durante la creazione, essi devono disporre delle autorizzazioni per utilizzare l'operazione che crea la risorsa, ad esempio `ecr:CreateRepository`. Se i tag vengono specificati nell'azione di creazione delle risorse, Amazon esegue autorizzazioni aggiuntive per l'azione `ecr:CreateRepository` per verificare se gli utenti dispongono delle autorizzazioni per creare tag.

Puoi utilizzare il controllo degli accessi basato su tag tramite le policy IAM. Di seguito vengono mostrati gli esempi.

La policy seguente consente a un utente di creare o di applicare un tag a un repository come `key=environment,value=dev`.

JSON

```
{
```

```
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "AllowCreateTaggedRepository",
            "Effect": "Allow",
            "Action": [
                "ecr:CreateRepository"
            ],
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "aws:RequestTag/environment": "dev"
                }
            }
        },
        {
            "Sid": "AllowTagRepository",
            "Effect": "Allow",
            "Action": [
                "ecr:TagResource"
            ],
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "aws:RequestTag/environment": "dev"
                }
            }
        }
    ]
}
```

La seguente politica consentirebbe a un utente di estrarre immagini da tutti i repository a meno che non siano contrassegnate come. `key=environment,value=prod`

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [{
            "Effect": "Allow",
            "Action": [
```

```
                "ecr:BatchGetImage",
                "ecr:GetDownloadUrlForLayer"
            ],
            "Resource": "*"
        },
        {
            "Effect": "Deny",
            "Action": [
                "ecr:BatchGetImage",
                "ecr:GetDownloadUrlForLayer"
            ],
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "ecr:ResourceTag/environment": "prod"
                }
            }
        }
    ]
}
```

AWS politiche gestite per Amazon Elastic Container Registry

Una politica AWS gestita è una politica autonoma creata e amministrata da. AWS AWS le politiche gestite sono progettate per fornire autorizzazioni per molti casi d'uso comuni, in modo da poter iniziare ad assegnare autorizzazioni a utenti, gruppi e ruoli.

Tieni presente che le policy AWS gestite potrebbero non concedere le autorizzazioni con il privilegio minimo per i tuoi casi d'uso specifici, poiché sono disponibili per tutti i clienti. AWS Si consiglia pertanto di ridurre ulteriormente le autorizzazioni definendo policy gestite dal cliente specifiche per i propri casi d'uso.

Non è possibile modificare le autorizzazioni definite nelle politiche gestite. AWS Se AWS aggiorna le autorizzazioni definite in una politica AWS gestita, l'aggiornamento ha effetto su tutte le identità principali (utenti, gruppi e ruoli) a cui è associata la politica. AWS è più probabile che aggiorni una policy AWS gestita quando ne Servizio AWS viene lanciata una nuova o quando diventano disponibili nuove operazioni API per i servizi esistenti.

Per ulteriori informazioni, consultare Policy gestite da AWS nella Guida per l'utente di IAM.

Amazon ECR fornisce diverse policy gestite che puoi collegare alle identità IAM o alle istanze Amazon EC2 . Queste policy gestite consentono diversi livelli di controllo sull'accesso alle risorse Amazon ECR e alle operazioni API. Per ulteriori informazioni su ciascuna operazione API citata in queste policy, consulta Actions (Operazioni) nella documentazione di riferimento alle API Amazon Elastic Container Registry.

Argomenti

- Amazon EC2 ContainerRegistryFullAccess
- Amazon EC2 ContainerRegistryPowerUser
- Amazon EC2 ContainerRegistryPullOnly
- Amazon EC2 ContainerRegistryReadOnly
- AWSECRPullThroughCache_ServiceRolePolicy
- ECRReplicationServiceRolePolicy
- ECRTemplateServiceRolePolicy
- Aggiornamenti di Amazon ECR alle politiche AWS gestite

Amazon EC2 ContainerRegistryFullAccess

È possibile allegare la policy `AmazonEC2ContainerRegistryFullAccess` alle identità IAM. Questa policy concede l'accesso amministrativo alle risorse di Amazon ECR e concede a un'identità IAM (ad esempio un utente, un gruppo o un ruolo) l'accesso ai servizi con AWS cui è integrato Amazon ECR per utilizzare tutte le funzionalità di Amazon ECR. L'utilizzo di questa policy consente l'accesso a tutte le funzionalità di Amazon ECR disponibili in. Console di gestione AWS

Per visualizzare le autorizzazioni per questa politica, consulta Amazon EC2 ContainerRegistryFullAccess nel AWS Managed Policy Reference.

Amazon EC2 ContainerRegistryPowerUser

È possibile allegare la policy `AmazonEC2ContainerRegistryPowerUser` alle identità IAM. Questa policy concede le autorizzazioni amministrative che consentono agli utenti IAM di leggere e scrivere nei repository, ma non permette né di eliminare i repository né di modificare i documenti di policy ad essi applicati.

Per visualizzare le autorizzazioni per questa politica, consulta Amazon EC2 ContainerRegistryPowerUser nel AWS Managed Policy Reference.

Amazon EC2 ContainerRegistryPullOnly

È possibile allegare la policy `AmazonEC2ContainerRegistryPullOnly` alle identità IAM. Questa politica concede l'autorizzazione a estrarre immagini di container da Amazon ECR. Se il registro è abilitato per la cache pull-through, consentirà anche ai pull di importare un'immagine da un registro upstream.

Per visualizzare le autorizzazioni per questa politica, consulta Amazon EC2 ContainerRegistryPullOnly nel AWS Managed Policy Reference.

Amazon EC2 ContainerRegistryReadOnly

È possibile allegare la policy `AmazonEC2ContainerRegistryReadOnly` alle identità IAM. Questa policy concede le autorizzazioni che consentono l'accesso in sola lettura ad Amazon ECR. Ciò include la possibilità di elencare repository e immagini all'interno dei repository. Include anche la possibilità di estrarre immagini da Amazon ECR con la CLI di Docker.

Per visualizzare le autorizzazioni per questa politica, consulta Amazon EC2 ContainerRegistryReadOnly nel AWS Managed Policy Reference.

`AWSECRPullThroughCache_ServiceRolePolicy`

Non è possibile attribuire la policy IAM gestita `AWSECRPullThroughCache_ServiceRolePolicy` alle entità IAM. Questa policy è collegata a un ruolo collegato al servizio che consente ad Amazon ECR di inviare immagini nei repository attraverso il flusso di lavoro della cache pull-through. Per ulteriori informazioni, consulta Ruolo collegato ai servizi Amazon ECR per la cache pull-through.

`ECRReplicationServiceRolePolicy`

Non è possibile attribuire la policy IAM gestita `ECRReplicationServiceRolePolicy` alle entità IAM. Questa policy è associata a un ruolo collegato ai servizi che consente ad Amazon ECR di eseguire operazioni per tuo conto. Per ulteriori informazioni, consulta Utilizzo di ruoli collegati ai servizi per Amazon ECR.

`ECRTemplateServiceRolePolicy`

Non è possibile attribuire la policy IAM gestita `ECRTemplateServiceRolePolicy` alle entità IAM. Questa policy è associata a un ruolo collegato ai servizi che consente ad Amazon ECR di eseguire operazioni per tuo conto. Per ulteriori informazioni, consulta Utilizzo di ruoli collegati ai servizi per Amazon ECR.

Aggiornamenti di Amazon ECR alle politiche AWS gestite

Visualizza i dettagli sugli aggiornamenti delle politiche AWS gestite per Amazon ECR dal momento in cui questo servizio ha iniziato a tracciare queste modifiche. Per gli avvisi automatici sulle modifiche apportate a questa pagina, sottoscrivere il feed RSS nella pagina di Cronologia dei documenti di Amazon ECR.

Modifica	Descrizione	Data
Ruolo collegato ai servizi Amazon ECR per la cache pull-through: aggiornamento a una policy esistente	Amazon ECR ha aggiunto nuove autorizzazioni alla policy `AWSECRPullThroughCache_ServiceRolePolicy` . Queste autorizzazioni consentono ad Amazon ECR di estrarre immagini dal registro privato ECR. Ciò è necessario quando si utilizza una regola pull through cache per memorizzare nella cache le immagini da un altro registro privato Amazon ECR.	12 marzo 2025
Amazon EC2 Container RegistryPullOnly — Nuova politica	Amazon ECR ha aggiunto una nuova policy che concede autorizzazioni pull-only ad Amazon ECR.	10 ottobre 2024
ECRTemplateService RolePolicy: nuova policy	Amazon ECR ha aggiunto una nuova policy. Questa policy è associata al ruolo collegato al `ECRTemplateServiceRolePolicy` servizio per la funzionalità di creazione di modelli di repository.	20 giugno 2024

Modifica	Descrizione	Data
AWSECRPullThroughCache_ServiceRolePolicy: aggiornamento di una policy esistente	Amazon ECR ha aggiunto nuove autorizzazioni alla policy `AWSECRPullThroughCache_ServiceRolePolicy` . Queste autorizzazioni consentono ad Amazon ECR di recuperare i contenuti crittografati di un segreto di Secrets Manager. Ciò è necessario quando utilizzi una regola di cache pull-through per memorizzare nella cache le immagini da un registro upstream che richiede l'autenticazione.	15 novembre 2023
AWSECRPullThroughCache_ServiceRolePolicy: nuova policy	Amazon ECR ha aggiunto una nuova policy. Questa policy è associata al ruolo `AWSServiceRoleForECRPullThroughCache` collegato al servizio per la funzione di cache pull-through.	29 novembre 2021
ECRReplicationServiceRolePolicy: nuova policy	Amazon ECR ha aggiunto una nuova policy. Questa policy è associata al ruolo `AWSServiceRoleForECRReplication` collegato al servizio per la funzione di replica.	4 dicembre 2020

Modifica	Descrizione	Data
Amazon EC2 Container RegistryFullAccess: aggiornamento a una politica esistente	Amazon ECR ha aggiunto nuove autorizzazioni alla policy `AmazonEC2ContainerRegistryFullAccess` . Queste autorizzazioni consentono alle entità principali di creare il ruolo collegato ai servizi Amazon ECR.	4 dicembre 2020
Amazon EC2 Container RegistryReadOnly: aggiornamento a una politica esistente	Amazon ECR ha aggiunto nuove autorizzazioni alla policy `AmazonEC2ContainerRegistryReadOnly` che consentono alle entità principali di leggere le policy del ciclo di vita, elencare i tag e descrivere i risultati della scansione delle immagini.	10 dicembre 2019
Amazon EC2 Container RegistryPowerUser: aggiornamento a una politica esistente	Amazon ECR ha aggiunto nuove autorizzazioni alla policy `AmazonEC2ContainerRegistryPowerUser` . Consentono alle entità principali di leggere le policy del ciclo di vita, elencare i tag e descrivere i risultati della scansione delle immagini.	10 dicembre 2019

Modifica	Descrizione	Data
Amazon EC2 Container RegistryFullAccess: aggiornamento a una politica esistente	Amazon ECR ha aggiunto nuove autorizzazioni alla policy `AmazonEC2ContainerRegistryFullAccess` . Consentono ai responsabili di cercare eventi di gestione o eventi AWS CloudTrail Insights acquisiti da CloudTrail.	10 Novembre 2017
Amazon EC2 Container RegistryReadOnly: aggiornamento a una politica esistente	Amazon ECR ha aggiunto nuove autorizzazioni alla policy `AmazonEC2ContainerRegistryReadOnly` . Consentono alle entità principali di descrivere le immagini Amazon ECR.	11 ottobre 2016
Amazon EC2 Container RegistryPowerUser: aggiornamento a una politica esistente	Amazon ECR ha aggiunto nuove autorizzazioni alla policy `AmazonEC2ContainerRegistryPowerUser` . Consentono alle entità principali di descrivere le immagini Amazon ECR.	11 ottobre 2016

Modifica	Descrizione	Data
Amazon EC2 Container RegistryReadOnly — Nuova politica	Amazon ECR ha aggiunto una nuova policy che concede autorizzazioni di sola lettura ad Amazon ECR. Queste autorizzazioni prevedono la possibilità di elencare repository e immagini all'inter no dei repository. Prevedono anche la possibilità di estrarre immagini da Amazon ECR con la CLI di Docker.	21 dicembre 2015
Amazon EC2 Container RegistryPowerUser — Nuova politica	Amazon ECR ha aggiunto una nuova politica che concede autorizzazioni amministrative che consentono agli utenti di leggere e scrivere negli archivi ma non consente loro di eliminare gli archivi o modificare i documenti relativi alle policy a loro applicati.	21 dicembre 2015
Amazon EC2 Container RegistryFullAccess — Nuova politica	Amazon ECR ha aggiunto una nuova policy. Questa policy consente l'accesso completo ad Amazon ECR.	21 dicembre 2015
Amazon ECR ha iniziato a monitorare le modifiche	Amazon ECR ha iniziato a tracciare le modifiche per le policy AWS gestite.	24 giugno 2021

Utilizzo di ruoli collegati ai servizi per Amazon ECR

Amazon Elastic Container Registry (Amazon ECR) AWS Identity and Access Management utilizza ruoli collegati ai servizi (IAM) per fornire le autorizzazioni necessarie per utilizzare le funzionalità di

replica e pull through cache. Un ruolo collegato ai servizi è un tipo di ruolo IAM univoco collegato direttamente ad Amazon ECR. Il ruolo collegato ai servizi è predefinito da Amazon ECR. Include tutte le autorizzazioni richieste dal servizio per supportare le funzionalità di replica e cache pull-through per il registro privato. Dopo avere configurato la replica o la cache pull-through per il registro, viene creato automaticamente un ruolo collegato al servizio per conto dell'utente. Per ulteriori informazioni, consulta Impostazioni del registro privato in Amazon ECR.

Un ruolo collegato ai servizi semplifica la configurazione della replica e della cace pull-trough con Amazon ECR. Ciò avviene perché, utilizzandolo, non sarà più necessario aggiungere manualmente tutte le autorizzazioni necessarie. Amazon ECR definisce le autorizzazioni del ruolo associato ai servizi e, salvo diversamente definito, solo Amazon ECR può assumere il ruolo. Le autorizzazioni definite includono policy di attendibilità e di autorizzazioni. Le policy di autorizzazioni non possono essere collegate a nessun'altra entità IAM.

Puoi eliminare il ruolo collegato ai servizi corrispondente solo dopo avere disabilitato la replica o la cache pull-trough nel registro. Ciò garantisce che l'utente non rimuova inavvertitamente le autorizzazioni richieste da Amazon ECR per queste funzionalità.

Per informazioni sugli altri servizi che supportano i ruoli collegati ai servizi consulta Servizi AWS che funzionano con IAM. In questa pagina collegata, cerca i servizi che hanno Yes (Sì) nella colonna Service-Linked Role (Ruolo collegato ai servizi). Scegli un link Yes (Sì) per visualizzare la documentazione relativa al ruolo collegato ai servizi per tale servizio.

Argomenti

Regioni supportate per i ruoli collegati ai servizi di Amazon ECR

Amazon ECR supporta l'utilizzo di ruoli collegati ai servizi in tutte le regioni in cui il servizio Amazon ECR è disponibile. Per ulteriori informazioni sulla disponibilità delle regioni di Amazon ECR, consulta Regioni ed endpoint AWS.

Ruolo collegato ai servizi Amazon ECR per la replica

Amazon ECR utilizza un ruolo collegato al servizio denominato che AWSServiceRoleForECRReplicationconsente ad Amazon ECR di replicare le immagini su più account.

Autorizzazioni del ruolo collegato ai servizi per Amazon ECR

Il ruolo AWSService RoleFor ECRReplication collegato al servizio prevede che i seguenti servizi assumano il ruolo:

- `replication.ecr.amazonaws.com`

La seguente policy delle autorizzazioni del ruolo `ECRReplicationServiceRolePolicy` consente ad Amazon ECR di eseguire le operazioni seguenti su tutte le risorse:

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "ecr:CreateRepository",
                "ecr:ReplicateImage"
            ],
            "Resource": "*"
        }
    ]
}
```

> **Note**
>
> La `ReplicateImage` è un'API interna utilizzata da Amazon ECR per la replica e non può essere chiamata direttamente.

Per consentire a un'entità IAM (come un utente, un gruppo o un ruolo) di creare, modificare o eliminare un ruolo collegato ai servizi devi configurare le relative autorizzazioni. Per ulteriori informazioni, consulta Autorizzazioni del ruolo collegato ai servizi nella Guida per l'utente di IAM.

Creazione di un ruolo collegato ai servizi per Amazon ECR

Non devi creare manualmente il ruolo collegato al servizio Amazon ECR. Quando configuri le impostazioni di replica per il tuo registro nell'API Console di gestione AWS AWS CLI, nell'o nell' AWS API, Amazon ECR crea il ruolo collegato al servizio per te.

Se elimini questo ruolo collegato ai servizi e devi ricrearlo di nuovo, puoi utilizzare lo stesso processo per ricreare il ruolo nel tuo account. Quando si configurano le impostazioni di replica per il registro, Amazon ECR crea il ruolo collegato ai servizi per l'utente.

Modifica di un ruolo collegato ai servizi per Amazon ECR

Amazon ECR non consente la modifica manuale del ruolo collegato al AWSService RoleFor ECRReplication servizio. Dopo aver creato un ruolo collegato al servizio, non è possibile modificarne il nome, perché potrebbero farvi riferimento diverse entità. È possibile tuttavia modificarne la descrizione utilizzando IAM. Per ulteriori informazioni, consulta Modifica di un ruolo collegato al servizio nella Guida per l'utente di IAM.

Eliminazione di un ruolo collegato ai servizi per Amazon ECR

Se non è più necessario utilizzare una caratteristica o un servizio che richiede un ruolo collegato ai servizi, ti consigliamo di eliminare il ruolo. In questo modo non hai un'entità non utilizzata che non viene monitorata o gestita attivamente. Tuttavia, prima di poter eliminare manualmente il ruolo collegato ai servizi, è necessario rimuovere la configurazione della replica per il registro in ogni regione.

> **Note**
>
> Se provi a eliminare risorse mentre il servizio Amazon ECR utilizza ancora i ruoli, l'operazione di eliminazione potrebbe non riuscire. In questo caso, attendi alcuni minuti e riprova.

Per eliminare le risorse Amazon ECR utilizzate da AWSService RoleFor ECRReplication

1. Apri la console Amazon ECR all'indirizzo https://console.aws.amazon.com/ecr/.

2. Dalla barra di navigazione, scegli la regione in cui configurare le impostazioni di replica.
3. Nel pannello di navigazione, seleziona Private registry (Registro privato).
4. Nella pagina Private registry (Registro privato), nella sezione Replication configuration (Configurazione di replica) scegli Edit (Modifica).
5. Per eliminare tutte le regole di replica, scegli Delete all (Elimina tutto). Questo passaggio richiede una conferma.

Per eliminare manualmente il ruolo collegato ai servizi mediante IAM

Utilizza la console IAM AWS CLI, o l' AWS API per eliminare il ruolo collegato al AWSServiceRoleForECRReplicationservizio. Per ulteriori informazioni, consulta Eliminazione del ruolo collegato al servizio nella Guida per l'utente di IAM.

Ruolo collegato ai servizi Amazon ECR per la cache pull-through

Amazon ECR utilizza un ruolo collegato al servizio denominato AWSServiceRoleForECRPullThroughCacheche autorizza Amazon ECR a eseguire azioni per tuo conto per completare le azioni pull through cache. Per ulteriori informazioni sulla cache pull-through, consulta Modelli per controllare gli archivi creati durante un'azione di pull through cache, di creazione in modalità push o di replica.

Autorizzazioni del ruolo collegato ai servizi per Amazon ECR

Il ruolo AWSServiceRoleForECRPullThroughCachecollegato al servizio prevede che il seguente servizio assuma il ruolo.

- `pullthroughcache.ecr.amazonaws.com`

Dettagli delle autorizzazioni

La policy delle autorizzazioni `AWSECRPullThroughCache_ServiceRolePolicy` è attribuita al ruolo collegato ai servizi. Questa policy gestita concede ad Amazon ECR l'autorizzazione all'esecuzione delle operazioni seguenti. Per ulteriori informazioni, consulta `AWSECRPullThroughCache_ServiceRolePolicy`.

- `ecr`— Consente al servizio Amazon ECR di estrarre e inviare immagini a un archivio privato.
- `secretsmanager:GetSecretValue`— Consente al servizio Amazon ECR di recuperare i contenuti crittografati di un Gestione dei segreti AWS segreto. Ciò è necessario quando utilizzi una

regola di cache pull-through per memorizzare nella cache le immagini da un registro upstream che richiede l'autenticazione nel tuo registro privato. Questa autorizzazione è valida solo per i segreti con il prefisso di nome `ecr-pullthroughcache/`.

La policy `AWSECRPullThroughCache_ServiceRolePolicy` contiene il JSON seguente.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "ECR",
            "Effect": "Allow",
            "Action": [
                "ecr:GetAuthorizationToken",
                "ecr:BatchCheckLayerAvailability",
                "ecr:InitiateLayerUpload",
                "ecr:UploadLayerPart",
                "ecr:CompleteLayerUpload",
                "ecr:PutImage",
                "ecr:BatchGetImage",
                "ecr:BatchImportUpstreamImage",
                "ecr:GetDownloadUrlForLayer",
                "ecr:GetImageCopyStatus"
            ],
            "Resource": "*"
        },
        {
            "Sid": "SecretsManager",
            "Effect": "Allow",
            "Action": [
                "secretsmanager:GetSecretValue"
            ],
            "Resource": "arn:aws:secretsmanager:*:*:secret:ecr-pullthroughcache/
*",
            "Condition": {
                "StringEquals": {
                    "aws:ResourceAccount": "${aws:PrincipalAccount}"
                }
            }
        }
```

```
    ]
}
```

Per consentire a un'entità IAM (come un utente, un gruppo o un ruolo) di creare, modificare o eliminare un ruolo collegato ai servizi devi configurare le relative autorizzazioni. Per ulteriori informazioni, consulta Autorizzazioni del ruolo collegato ai servizi nella Guida per l'utente IAM.

Creazione di un ruolo collegato ai servizi per Amazon ECR

Non devi creare manualmente il ruolo collegato al servizio Amazon ECR per la cache pull-trough. Quando crei una regola pull through cache per il tuo registro privato nell'API Console di gestione AWS, nell' AWS CLI o nell' AWS API, Amazon ECR crea il ruolo collegato al servizio per te.

Se elimini questo ruolo collegato ai servizi e devi ricrearlo di nuovo, puoi utilizzare lo stesso processo per ricreare il ruolo nel tuo account. Quando crei una regola di cache pull-through per il tuo registro privato, Amazon ECR crea il ruolo collegato ai servizi per te se non esiste già.

Modifica di un ruolo collegato ai servizi per Amazon ECR

Amazon ECR non consente la modifica manuale del ruolo collegato al AWSServiceRoleForECRPullThroughCacheservizio. Dopo aver creato un ruolo collegato ai servizi, non puoi modificarne il nome, perché potrebbero farvi riferimento diverse entità. È possibile tuttavia modificarne la descrizione utilizzando IAM. Per ulteriori informazioni, consulta Modifica di un ruolo collegato al servizio nella Guida per l'utente di IAM.

Eliminazione di un ruolo collegato ai servizi per Amazon ECR

Se non è più necessario utilizzare una caratteristica o un servizio che richiede un ruolo collegato ai servizi, ti consigliamo di eliminare il ruolo. In questo modo non hai un'entità non utilizzata che non viene monitorata o gestita attivamente. Tuttavia, prima di poter eliminare manualmente il ruolo collegato ai servizi, è necessario eliminare le regole della cache pull-through per il registro in ogni regione.

> **Note**
>
> Se provi a eliminare risorse mentre il servizio Amazon ECR utilizza ancora il ruolo, l'operazione di eliminazione potrebbe non riuscire. In questo caso, attendi alcuni minuti e riprova.

Per eliminare le risorse Amazon ECR utilizzate dal ruolo collegato ai servizi AWSServiceRoleForECRPullThroughCache

1. Apri la console Amazon ECR all'indirizzo https://console.aws.amazon.com/ecr/.
2. Nella barra di navigazione, seleziona la regione in cui vengono create le regole della cache pull-through.
3. Nel pannello di navigazione, seleziona Private registry (Registro privato).
4. Nella pagina Private registry (Registro privato), nella sezione Pull through cache configuration (Configurazione di cache pull through) scegli Edit (Modifica).
5. Per ogni regola di cache pull through che hai creato, seleziona la regola e quindi scegli Delete rule (Elimina regola).

Per eliminare manualmente il ruolo collegato ai servizi mediante IAM

Utilizza la console IAM AWS CLI, o l' AWS API per eliminare il ruolo collegato al AWSServiceRoleForECRPullThroughCacheservizio. Per ulteriori informazioni, consulta Eliminazione del ruolo collegato al servizio nella Guida per l'utente di IAM.

Ruolo collegato al servizio Amazon ECR per i modelli di creazione di repository

Amazon ECR utilizza un ruolo collegato al servizio denominato AWSServiceRoleForECRTemplateche autorizza Amazon ECR a eseguire azioni per tuo conto per completare le azioni dei modelli di creazione di repository.

Autorizzazioni del ruolo collegato ai servizi per Amazon ECR

Il ruolo AWSServiceRoleForECRTemplatecollegato al servizio si affida al seguente servizio per l'assunzione del ruolo.

- `ecr.amazonaws.com`

Dettagli delle autorizzazioni

La policy delle autorizzazioni `ECRTemplateServiceRolePolicy` è attribuita al ruolo collegato ai servizi. Questa policy gestita concede ad Amazon ECR l'autorizzazione a eseguire azioni di creazione di repository per tuo conto.

La policy `ECRTemplateServiceRolePolicy` contiene il JSON seguente.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
     "Sid": "CreateRepositoryWithTemplate",
            "Effect": "Allow",
            "Action": [
                "ecr:CreateRepository"
            ],
            "Resource": "*"
        }
    ]
}
```

Per consentire a un'entità IAM (come un utente, un gruppo o un ruolo) di creare, modificare o eliminare un ruolo collegato ai servizi devi configurare le relative autorizzazioni. Per ulteriori informazioni, consulta Autorizzazioni del ruolo collegato ai servizi nella Guida per l'utente IAM.

Creazione di un ruolo collegato ai servizi per Amazon ECR

Non è necessario creare manualmente il ruolo collegato al servizio Amazon ECR per il modello di creazione del repository. Quando crei una regola del modello di creazione di un repository per il tuo registro privato nell'API Console di gestione AWS AWS CLI, nell'o nell' AWS API, Amazon ECR crea il ruolo collegato al servizio per te.

Se elimini questo ruolo collegato ai servizi e devi ricrearlo di nuovo, puoi utilizzare lo stesso processo per ricreare il ruolo nel tuo account. Quando crei una regola per la creazione di un repository per il tuo registro privato, Amazon ECR crea nuovamente il ruolo collegato al servizio se non esiste già.

Modifica di un ruolo collegato ai servizi per Amazon ECR

Amazon ECR non consente la modifica manuale del ruolo collegato al AWSServiceRoleForECRTemplateservizio. Dopo aver creato un ruolo collegato ai servizi, non puoi modificarne il nome, perché potrebbero farvi riferimento diverse entità. È possibile tuttavia modificarne la descrizione utilizzando IAM. Per ulteriori informazioni, consulta Modifica di un ruolo collegato al servizio nella Guida per l'utente di IAM.

Eliminazione di un ruolo collegato ai servizi per Amazon ECR

Se non è più necessario utilizzare una caratteristica o un servizio che richiede un ruolo collegato ai servizi, ti consigliamo di eliminare il ruolo. In questo modo non hai un'entità non utilizzata che non viene monitorata o gestita attivamente. Tuttavia, devi eliminare le regole di creazione del repository per il registro in ogni regione prima di poter eliminare manualmente il ruolo collegato al servizio.

> **Note**
>
> Se provi a eliminare risorse mentre il servizio Amazon ECR utilizza ancora il ruolo, l'operazione di eliminazione potrebbe non riuscire. In questo caso, attendi alcuni minuti e riprova.

Per eliminare le risorse Amazon ECR utilizzate dal ruolo collegato ai servizi AWSServiceRoleForECRTemplate

1. Apri la console Amazon ECR all'indirizzo https://console.aws.amazon.com/ecr/.
2. Dalla barra di navigazione, scegli la regione in cui vengono create le regole di creazione del repository.
3. Nel pannello di navigazione, seleziona Private registry (Registro privato).
4. Nella pagina del registro privato, nella sezione Modelli di creazione del repository, scegli Modifica.
5. Per ogni regola di creazione del repository che hai creato, seleziona la regola e quindi scegli Elimina regola.

Per eliminare manualmente il ruolo collegato ai servizi mediante IAM

Utilizza la console IAM AWS CLI, o l' AWS API per eliminare il ruolo collegato al AWSServiceRoleForECRTemplateservizio. Per ulteriori informazioni, consulta Eliminazione del ruolo collegato al servizio nella Guida per l'utente di IAM.

Risoluzione dei problemi di Identity and Access Management per Amazon Elastic Container Registry

Utilizza le informazioni seguenti per diagnosticare e risolvere i problemi comuni che possono verificarsi durante l'utilizzo di Amazon ECR e IAM.

Argomenti

Non sono autorizzato a eseguire un'operazione in Amazon ECR

Se ricevi un errore che indica che non sei autorizzato a eseguire un'operazione, le tue policy devono essere aggiornate per poter eseguire l'operazione.

L'errore di esempio seguente si verifica quando l'utente IAM `mateojackson` prova a utilizzare la console per visualizzare i dettagli relativi a una risorsa *`my-example-widget`* fittizia ma non dispone di autorizzazioni `ecr:`*`GetWidget`* fittizie.

```
User: arn:aws:iam::123456789012:user/mateojackson is not authorized to perform:
 ecr:GetWidget on resource: my-example-widget
```

In questo caso, la policy per l'utente `mateojackson` deve essere aggiornata per consentire l'accesso alla risorsa *`my-example-widget`* utilizzando l'azione `ecr:`*`GetWidget`*.

Se hai bisogno di aiuto, contatta il tuo AWS amministratore. L'amministratore è la persona che ti ha fornito le credenziali di accesso.

Non sono autorizzato a eseguire iam: PassRole

Se ricevi un errore che indica che non sei autorizzato a eseguire l'operazione `iam:PassRole`, dovrai aggiornare le policy in modo da poter passare un ruolo ad Amazon ECR.

Alcuni Servizi AWS consentono di trasferire un ruolo esistente a quel servizio invece di creare un nuovo ruolo di servizio o un ruolo collegato al servizio. Per eseguire questa operazione, è necessario disporre delle autorizzazioni per trasmettere il ruolo al servizio.

L'errore di esempio seguente si verifica quando un utente IAM denominato `marymajor` cerca di utilizzare la console per eseguire un'operazione in Amazon ECR. Tuttavia, l'azione richiede che il servizio disponga delle autorizzazioni concesse da un ruolo di servizio. Mary non dispone delle autorizzazioni per trasmettere il ruolo al servizio.

```
User: arn:aws:iam::123456789012:user/marymajor is not authorized to perform:
 iam:PassRole
```

In questo caso, le policy di Mary devono essere aggiornate per poter eseguire l'operazione `iam:PassRole`.

Se hai bisogno di aiuto, contatta il tuo AWS amministratore. L'amministratore è la persona che ti ha fornito le credenziali di accesso.

Voglio consentire a persone esterne a me di accedere Account AWS alle mie risorse Amazon ECR

È possibile creare un ruolo con il quale utenti in altri account o persone esterne all'organizzazione possono accedere alle tue risorse. È possibile specificare chi è attendibile per l'assunzione del ruolo. Per i servizi che supportano politiche basate sulle risorse o liste di controllo degli accessi (ACLs), puoi utilizzare tali politiche per consentire alle persone di accedere alle tue risorse.

Per maggiori informazioni, consulta gli argomenti seguenti:

- Per sapere se Amazon ECR supporta queste caratteristiche, consultare Come funziona Amazon Elastic Container Registry con IAM.
- Per scoprire come fornire l'accesso alle risorse di tua proprietà, consulta Fornire l'accesso a un utente IAM in Account AWS un altro Account AWS di tua proprietà nella IAM User Guide.
- Per scoprire come fornire l'accesso alle tue risorse a terze parti Account AWS, consulta Fornire l'accesso a soggetti Account AWS di proprietà di terze parti nella Guida per l'utente IAM.
- Per informazioni su come fornire l'accesso tramite la federazione delle identità, consulta Fornire l'accesso a utenti autenticati esternamente (federazione delle identità) nella Guida per l'utente IAM.
- Per informazioni sulle differenze di utilizzo tra ruoli e policy basate su risorse per l'accesso multi-account, consulta Accesso a risorse multi-account in IAM nella Guida per l'utente di IAM.

Protezione dei dati in Amazon ECR

Il modello di responsabilità AWS condivisa modello si applica alla protezione dei dati in Amazon Elastic Container Service. Come descritto in questo modello, AWS è responsabile della protezione dell'infrastruttura globale che gestisce tutti i Cloud AWS. L'utente è responsabile del controllo dei contenuti ospitati su questa infrastruttura. L'utente è inoltre responsabile della configurazione della protezione e delle attività di gestione per i Servizi AWS utilizzati. Per maggiori informazioni sulla privacy dei dati, consulta le Domande frequenti sulla privacy dei dati. Per informazioni sulla protezione dei dati in Europa, consulta il post del blog relativo al AWS Modello di responsabilità condivisa e GDPR nel AWS Blog sulla sicurezza.

Ai fini della protezione dei dati, consigliamo di proteggere Account AWS le credenziali e configurare i singoli utenti con AWS IAM Identity Center or AWS Identity and Access Management (IAM). In tal modo, a ogni utente verranno assegnate solo le autorizzazioni necessarie per svolgere i suoi compiti. Suggeriamo, inoltre, di proteggere i dati nei seguenti modi:

- Utilizza l'autenticazione a più fattori (MFA) con ogni account.
- SSL/TLS Da utilizzare per comunicare con AWS le risorse. È richiesto TLS 1.2 ed è consigliato TLS 1.3.
- Configura l'API e la registrazione delle attività degli utenti con AWS CloudTrail. Per informazioni sull'utilizzo dei CloudTrail percorsi per acquisire AWS le attività, consulta Lavorare con i CloudTrail percorsi nella Guida per l'AWS CloudTrail utente.
- Utilizza soluzioni di AWS crittografia, insieme a tutti i controlli di sicurezza predefiniti all'interno Servizi AWS.
- Utilizza i servizi di sicurezza gestiti avanzati, come Amazon Macie, che aiutano a individuare e proteggere i dati sensibili archiviati in Amazon S3.
- Se hai bisogno di moduli crittografici convalidati FIPS 140-3 per accedere AWS tramite un'interfaccia a riga di comando o un'API, usa un endpoint FIPS. Per ulteriori informazioni sugli endpoint FIPS disponibili, consulta il Federal Information Processing Standard (FIPS) 140-3.

Ti consigliamo di non inserire mai informazioni riservate o sensibili, ad esempio gli indirizzi e-mail dei clienti, nei tag o nei campi di testo in formato libero, ad esempio nel campo Nome. Ciò include quando lavori con Amazon ECS o altro Servizi AWS utilizzando la console, l'API o AWS SDKs. AWS CLI I dati inseriti nei tag o nei campi di testo in formato libero utilizzati per i nomi possono essere utilizzati per i la fatturazione o i log di diagnostica. Quando si fornisce un URL a un server esterno, suggeriamo vivamente di non includere informazioni sulle credenziali nell'URL per convalidare la richiesta al server.

Argomenti

- Crittografia dei dati a riposo

Crittografia dei dati a riposo

> **Important**
>
> La crittografia lato server a doppio livello con AWS KMS (DSSE-KMS) è disponibile solo nelle regioni. AWS GovCloud (US)

Amazon ECR memorizza le immagini nei bucket Amazon S3 gestiti da Amazon ECR. Per impostazione predefinita, Amazon ECR utilizza la crittografia lato server con chiavi di crittografia gestite da Amazon S3 che crittografano i dati a riposo utilizzando un algoritmo di crittografia AES-256. Ciò non richiede alcuna operazione da parte tua e viene offerto senza costi aggiuntivi. Per ulteriori informazioni, consulta Protezione dei dati mediante la crittografia lato server con chiavi di crittografia gestite da Amazon S3 (SSE-S3) nella Guida per l'utente di Amazon Simple Storage Service.

Per un maggiore controllo sulla crittografia per i tuoi repository Amazon ECR, puoi utilizzare la crittografia lato server con chiavi KMS archiviate in (). AWS Key Management Service AWS KMS Quando si utilizza AWS KMS per crittografare i dati, è possibile utilizzare l'impostazione predefinita Chiave gestita da AWS, gestita da Amazon ECR, o specificare la propria chiave KMS (denominata chiave gestita dal cliente). Per ulteriori informazioni, consulta Protezione dei dati utilizzando la crittografia lato server con chiavi KMS archiviate in AWS KMS (SSE-KMS) nella Guida per l'utente di Amazon Simple Storage Service.

Puoi scegliere di applicare due livelli di crittografia alle tue immagini Amazon ECR utilizzando la crittografia lato server a doppio livello con (). AWS KMS DSSE-KMS DSSE-KMSl'opzione è simile aSSE-KMS, ma applica due singoli livelli di crittografia anziché un livello. Per ulteriori informazioni, vedere Utilizzo della crittografia lato server a due livelli con AWS KMS chiavi (DSSE-KMS).

Ogni repository Amazon ECR dispone di una configurazione di crittografia che viene impostata al momento della creazione del repository. È possibile utilizzare configurazioni di crittografia diverse su ciascun repository. Per ulteriori informazioni, consulta Creazione di un repository privato Amazon ECR per archiviare immagini.

Quando viene creato un repository con la AWS KMS crittografia abilitata, viene utilizzata una chiave KMS per crittografare il contenuto del repository. Inoltre, Amazon ECR aggiunge una AWS KMS sovvenzione alla chiave KMS con l'archivio Amazon ECR come beneficiario principale.

Di seguito vengono fornite informazioni di alto livello su come Amazon ECR è integrato con AWS KMS per crittografare e decrittografare i repository:

1. Durante la creazione di un repository, Amazon ECR invia una DescribeKeychiamata a per AWS KMS convalidare e recuperare l'Amazon Resource Name (ARN) della chiave KMS specificata nella configurazione di crittografia.
2. Amazon ECR invia due CreateGrantrichieste per AWS KMS creare sovvenzioni sulla chiave KMS per consentire ad Amazon ECR di crittografare e decrittografare i dati utilizzando la chiave dati.
3. Quando si invia un'immagine, viene effettuata una GenerateDataKeyrichiesta AWS KMS che specifica la chiave KMS da utilizzare per crittografare il livello di immagine e il manifesto.
4. AWS KMS genera una nuova chiave dati, la cripta con la chiave KMS specificata e invia la chiave di dati crittografata da archiviare con i metadati del livello di immagine e il manifesto dell'immagine.
5. Quando si estrae un'immagine, viene effettuata una richiesta Decrypt a AWS KMS, specificando la chiave di dati crittografata.
6. AWS KMS decrittografa la chiave dati crittografata e invia la chiave dati decrittografata ad Amazon S3.
7. La chiave dati viene utilizzata per decrittografare il livello immagine prima che il livello immagine venga estratto.
8. Quando un repository viene eliminato, Amazon ECR invia due RetireGrantrichieste per AWS KMS ritirare le sovvenzioni create per il repository.

Considerazioni

I seguenti punti devono essere considerati quando si utilizza la crittografia AWS KMS basata (SSE-KMSoDSSE-KMS) con Amazon ECR.

- Se crei il tuo repository Amazon ECR con crittografia KMS e non specifichi una chiave KMS, Amazon ECR utilizza un alias Chiave gestita da AWS con l'alias per impostazione predefinita. `aws/ecr` Questa chiave KMS viene creata nel tuo account la prima volta che crei un repository con la crittografia KMS abilitata.
- La configurazione della crittografia del repository non può essere modificata dopo la creazione di un repository.
- Quando si utilizza la crittografia KMS con la propria chiave KMS, la chiave deve esistere nella stessa regione del repository.

- Le autorizzazioni che Amazon ECR crea per tuo conto non devono essere revocate. Se revochi la concessione che autorizza Amazon ECR a utilizzare AWS KMS le chiavi del tuo account, Amazon ECR non può accedere a questi dati, crittografare le nuove immagini inserite nel repository o decrittografarle quando vengono estratte. Quando revochi una concessione per Amazon ECR, la modifica avviene immediatamente. Per revocare i diritti di accesso, eliminare il repository piuttosto che revocare la concessione. Quando un repository viene eliminato, Amazon ECR ritira le concessioni per tuo conto.
- L'utilizzo delle chiavi comporta AWS KMS un costo. Per ulteriori informazioni, consultare Prezzi di AWS Key Management Service.
- L'utilizzo della crittografia lato server a due livelli comporta un costo. Per ulteriori informazioni, consulta i prezzi di Amazon ECR

Autorizzazioni IAM richieste

Quando crei o elimini un repository Amazon ECR con crittografia lato server utilizzando AWS KMS, le autorizzazioni richieste dipendono dalla chiave KMS specifica in uso.

Autorizzazioni IAM richieste Chiave gestita da AWS per l'utilizzo di Amazon ECR

Per impostazione predefinita, quando AWS KMS la crittografia è abilitata per un repository Amazon ECR ma non viene specificata alcuna chiave KMS, viene utilizzata la per Chiave gestita da AWS Amazon ECR. Quando la chiave KMS AWS gestita per Amazon ECR viene utilizzata per crittografare un repository, qualsiasi principale autorizzato a creare un repository può anche abilitare la crittografia sul repository. AWS KMS Tuttavia, l'entità principale IAM che elimina il repository deve disporre dell'autorizzazione `kms:RetireGrant`. Ciò consente il ritiro delle sovvenzioni che sono state aggiunte alla chiave al momento della creazione del repository. AWS KMS

La seguente policy IAM di esempio può essere aggiunta come policy inline a un utente per assicurarsi che questi disponga delle autorizzazioni minime necessarie per eliminare un repository per cui è attivata la crittografia. La chiave KMS utilizzata per crittografare il repository può essere specificata utilizzando il parametro della risorsa.

JSON

```
{
    "Version":"2012-10-17",
    "Id": "ecr-kms-permissions",
```

```
    "Statement": [
        {
            "Sid": "AllowAccessToRetireTheGrantsAssociatedWithTheKey",
            "Effect": "Allow",
            "Action": [
                "kms:RetireGrant"
            ],
            "Resource": "arn:aws:kms:us-
west-2:111122223333:key/b8d9ae76-080c-4043-92EXAMPLE"
        }
    ]
}
```

Autorizzazioni IAM richieste quando si utilizza una chiave gestita dal cliente

Quando si crea un repository con AWS KMS crittografia abilitata utilizzando una chiave gestita dal cliente, sono necessarie le autorizzazioni sia per la politica delle chiavi KMS che per la politica IAM per l'utente o il ruolo che crea il repository.

Quando crei la tua chiave KMS, puoi utilizzare la chiave predefinita creata dalla policy AWS KMS o puoi specificare un'icona personalizzata. Per garantire che la chiave gestita dal cliente rimanga gestibile dal proprietario dell'account, la politica chiave per la chiave KMS dovrebbe consentire tutte le AWS KMS azioni all'utente root dell'account. Puoi aggiungere ulteriori autorizzazioni con ambito alle policy chiave, ma almeno all'utente root devono essere concesse autorizzazioni per gestire la chiave KMS. Per consentire l'utilizzo della chiave KMS solo per le richieste che provengono da Amazon ECR, puoi utilizzare la chiave kms: ViaService condition con il valore. `ecr.<region>.amazonaws.com`

L'esempio seguente di policy chiave fornisce all' AWS account (utente root) che possiede la chiave KMS l'accesso completo alla chiave KMS. Per ulteriori informazioni su questa policy chiave di esempio, consulta Consente l'accesso all' AWS account e abilita le politiche IAM nella AWS Key Management Service Developer Guide.

JSON

```
{
    "Version":"2012-10-17",
    "Id": "ecr-key-policy",
    "Statement": [
        {
```

```
            "Sid": "EnableIAMUserPermissions",
            "Effect": "Allow",
            "Principal": {
                "AWS": "arn:aws:iam::111122223333:root"
            },
            "Action": "kms:*",
            "Resource": "*"
        }
    ]
}
```

L'utente IAM, il ruolo IAM o l' AWS account che crea i tuoi repository deve disporre dell'`kms:DescribeKey`autorizzazione e `kms:CreateGrant` `kms:RetireGrant`, oltre alle autorizzazioni Amazon ECR necessarie.

> **Note**
>
> L'autorizzazione `kms:RetireGrant` deve essere aggiunta alla policy IAM dell'utente o del ruolo che crea il repository. Le autorizzazioni `kms:CreateGrant` e `kms:DescribeKey` possono essere aggiunte alla policy delle chiave per la chiave KMS o alla policy IAM dell'utente o del ruolo che creano il repository. Per ulteriori informazioni su come funzionano le AWS KMS autorizzazioni, consulta Autorizzazioni AWS KMS API: riferimento alle azioni e alle risorse nella Guida per gli sviluppatori.AWS Key Management Service

La seguente policy IAM di esempio può essere aggiunta come policy inline a un utente per assicurarsi che questi disponga delle autorizzazioni minime necessarie per creare un repository per cui è attivata la crittografia ed eliminare il repository al termine. La AWS KMS key utilizzata per crittografare il repository può essere specificata utilizzando il parametro delle risorse.

JSON

```
{
    "Version":"2012-10-17",
    "Id": "ecr-kms-permissions",
    "Statement": [
        {
            "Sid":
 "AllowAccessToCreateAndRetireTheGrantsAssociatedWithTheKeyAsWellAsDescribeTheKey",
```

```
            "Effect": "Allow",
            "Action": [
                "kms:CreateGrant",
                "kms:RetireGrant",
                "kms:DescribeKey"
            ],
            "Resource": "arn:aws:kms:us-
west-2:111122223333:key/b8d9ae76-080c-4043-92EXAMPLE"
        }
    ]
}
```

Consenti a un utente di elencare le chiavi KMS nella console durante la creazione di un repository

Quando si utilizza la console Amazon ECR per creare un repository, è possibile concedere le autorizzazioni per consentire a un utente di elencare le chiavi KMS gestite dal cliente nella regione quando si abilita la crittografia per il repository. L'esempio di policy IAM seguente illustra le autorizzazioni necessarie per elencare le chiavi e gli alias KMS quando si utilizza la console.

JSON

```
{
  "Version":"2012-10-17",
  "Statement": {
    "Effect": "Allow",
    "Action": [
      "kms:ListKeys",
      "kms:ListAliases",
      "kms:DescribeKey"
    ],
    "Resource": "*"
  }
}
```

Monitoraggio dell'interazione di Amazon ECR con AWS KMS

Puoi utilizzarlo AWS CloudTrail per tenere traccia delle richieste a cui Amazon ECR invia per tuo AWS KMS conto. Le voci di registro contenute nel CloudTrail registro contengono una chiave di contesto di crittografia per renderle più facilmente identificabili.

Contesto di crittografia di Amazon ECR

Un contesto di crittografia è un set di coppie chiave-valore che contiene dati arbitrari non segreti. Quando si include un contesto di crittografia in una richiesta di crittografia dei dati, associa AWS KMS crittograficamente il contesto di crittografia ai dati crittografati. lo stesso contesto di crittografia sia necessario per decrittografare i dati.

Nelle sue richieste GenerateDataKeye Decrypt a, AWS KMS Amazon ECR utilizza un contesto di crittografia con due coppie nome-valore che identificano il repository e il bucket Amazon S3 utilizzati. Questo viene mostrato nell'esempio seguente. I nomi non variano, ma i valori del contesto di crittografia combinati saranno diversi per ogni valore.

```
"encryptionContext": {
    "aws:s3:arn": "arn:aws:s3:::us-west-2-starport-manifest-bucket/EXAMPLE1-90ab-cdef-fedc-ba987BUCKET1/
sha256:a7766145a775d39e53a713c75b6fd6d318740e70327aaa3ed5d09e0ef33fc3df",
    "aws:ecr:arn": "arn:aws:ecr:us-west-2:111122223333:repository/repository-name"
}
```

Puoi utilizzare il contesto di crittografia per identificare queste operazioni crittografiche nei record e nei log di controllo, come AWS CloudTrailAmazon CloudWatch Logs, e come condizione per l'autorizzazione nelle politiche e nelle concessioni.

Il contesto di crittografia di Amazon ECR è costituito da due coppie nome-valore.

- aws:s3:arn – La prima coppia nome-valore identifica il bucket. La chiave è `aws:s3:arn`. L'Amazon Resource Name (ARN) del bucket Amazon S3 è il valore.

  ```
  "aws:s3:arn": "ARN of an Amazon S3 bucket"
  ```

  Ad esempio, se l'ARN del bucket è `arn:aws:s3:::us-west-2-starport-manifest-bucket/EXAMPLE1-90ab-cdef-fedc-ba987BUCKET1/sha256:a7766145a775d39e53a713c75b6fd6d318740e70327aaa3ed5d09e0ef33fc3df`, il contesto di crittografia include la seguente coppia.

  ```
  "arn:aws:s3:::us-west-2-starport-manifest-bucket/EXAMPLE1-90ab-cdef-fedc-ba987BUCKET1/sha256:a7766145a775d39e53a713c75b6fd6d318740e70327aaa3ed5d09e0ef33fc3df"
  ```

- aws:ecr:arn – La seconda coppia nome-valore identifica l'Amazon Resource Name (ARN) del repository. La chiave è `aws:ecr:arn`. Il valore rappresenta l'ARN del repository.

```
"aws:ecr:arn": "ARN of an Amazon ECR repository"
```

Ad esempio, se l'ARN del repository è `arn:aws:ecr:us-west-2:111122223333:repository/repository-name`, il contesto di crittografia include la seguente coppia.

```
"aws:ecr:arn": "arn:aws:ecr:us-west-2:111122223333:repository/repository-name"
```

risoluzione dei problemi

Quando si elimina un repository Amazon ECR con la console, se il repository viene eliminato correttamente ma Amazon ECR non è in grado di ritirare le concessioni aggiunte alla chiave KMS per il repository, viene visualizzato il seguente errore.

```
The repository [{repository-name}] has been deleted successfully but the grants created
 by the kmsKey [{kms_key}] failed to be retired
```

Quando ciò si verifica, puoi ritirare tu stesso le AWS KMS sovvenzioni per il repository.

Ritirare manualmente le AWS KMS sovvenzioni per un deposito

1. Elenca le concessioni per la AWS KMS chiave utilizzata per il repository. Il valore `key-id` viene incluso nell'errore visualizzato dalla console. Puoi anche utilizzare il `list-keys` comando per elencare sia le chiavi KMS gestite dal Chiavi gestite da AWS cliente che quelle gestite dal cliente in una regione specifica del tuo account.

   ```
   aws kms list-grants \
        --key-id b8d9ae76-080c-4043-9237-c815bfc21dfc
        --region us-west-2
   ```

   L'output include un `EncryptionContextSubset` con l'Amazon Resource Name (ARN) del repository. Questo può essere utilizzato per determinare quale concessione aggiunta alla chiave sia quella che si desidera ritirare. Il valore `GrantId` viene utilizzato quando si ritira la concessione nella fase successiva.
2. Ritira ogni concessione per la AWS KMS chiave aggiunta al repository. Sostituisci il valore per *`GrantId`* con l'ID della sovvenzione dall'output del passaggio precedente.

```
aws kms retire-grant \
     --key-id b8d9ae76-080c-4043-9237-c815bfc21dfc \
     --grant-id GrantId \
     --region us-west-2
```

Convalida della conformità per Amazon Elastic Container Registry

Per sapere se un Servizio AWS programma rientra nell'ambito di specifici programmi di conformità, consulta Servizi AWS la sezione Scope by Compliance Program Servizi AWS e scegli il programma di conformità che ti interessa. Per informazioni generali, consulta Programmi di AWS conformità Programmi di di .

È possibile scaricare report di audit di terze parti utilizzando AWS Artifact. Per ulteriori informazioni, consulta Scaricamento dei report in AWS Artifact .

La vostra responsabilità di conformità durante l'utilizzo Servizi AWS è determinata dalla sensibilità dei dati, dagli obiettivi di conformità dell'azienda e dalle leggi e dai regolamenti applicabili. Per ulteriori informazioni sulla responsabilità di conformità durante l'utilizzo Servizi AWS, consulta AWS la documentazione sulla sicurezza.

Sicurezza dell'infrastruttura in Amazon Elastic Container Registry

In quanto servizio gestito, Amazon Elastic Container Registry è protetto dalla sicurezza di rete AWS globale. Per informazioni sui servizi AWS di sicurezza e su come AWS protegge l'infrastruttura, consulta AWS Cloud Security. Per progettare il tuo AWS ambiente utilizzando le migliori pratiche per la sicurezza dell'infrastruttura, vedi Infrastructure Protection in Security Pillar AWS Well-Architected Framework.

Utilizzi chiamate API AWS pubblicate per accedere ad Amazon ECR attraverso la rete. I client devono supportare quanto segue:

- Transport Layer Security (TLS). È richiesto TLS 1.2 ed è consigliato TLS 1.3.
- Suite di cifratura con Perfect Forward Secrecy (PFS), ad esempio Ephemeral Diffie-Hellman (DHE) o Elliptic Curve Ephemeral Diffie-Hellman (ECDHE). La maggior parte dei sistemi moderni, come Java 7 e versioni successive, supporta tali modalità.

Puoi richiamare queste operazioni API da qualsiasi posizione di rete, ma Amazon ECR non supporta le policy di accesso basate sulle risorse che possono includere limitazioni sull'indirizzo IP di origine. Puoi anche utilizzare le policy di Amazon ECR per controllare l'accesso da endpoint Amazon Virtual Private Cloud (Amazon VPC) specifici o specifici. VPCs In effetti, questo isola l'accesso alla rete a una determinata risorsa Amazon ECR solo dal VPC specifico all'interno della rete. AWS Per ulteriori informazioni, consulta Endpoint VPC con interfaccia Amazon ECR ()AWS PrivateLink.

Endpoint VPC con interfaccia Amazon ECR ()AWS PrivateLink

Puoi migliorare la posizione di sicurezza del VPC configurando Amazon ECR in modo che utilizzi un endpoint VPC di interfaccia. Gli endpoint VPC sono basati su una tecnologia che consente di AWS PrivateLink accedere in modo privato ad Amazon APIs ECR tramite indirizzi IP privati (sia che). IPv4 IPv6 AWS PrivateLink limita tutto il traffico di rete tra il tuo VPC e Amazon ECR alla rete Amazon. Non è richiesto un gateway Internet, un dispositivo NAT o un gateway privato virtuale.

Per ulteriori informazioni sugli AWS PrivateLink endpoint VPC, consulta la sezione Endpoints VPC nella Amazon VPC User Guide.

Considerazioni sugli endpoint VPC di Amazon ECR

Prima di configurare gli endpoint VPC per Amazon ECR, tenere presente le considerazioni riportate di seguito:

- Per consentire alle tue attività Amazon ECS ospitate su EC2 istanze Amazon di estrarre immagini private da Amazon ECR, crea gli endpoint VPC di interfaccia per Amazon ECS. Per ulteriori informazioni, consulta Interface VPC Endpoints (AWS PrivateLink) nella Amazon Elastic Container Service Developer Guide.
- Le attività Amazon ECS con hosting su Fargate che estraggono le immagini del container da Amazon ECR possono limitare l'accesso al VPC specifico utilizzato dalle attività e all'endpoint VPC utilizzato dal servizio aggiungendo le chiavi di condizione al ruolo IAM per l'attività. Per ulteriori informazioni, consulta Autorizzazioni IAM facoltative per attività Fargate che estraggono le immagini Amazon ECR su endpoint di interfaccia nella Guida per lo sviluppatore di Amazon Elastic Container.
- Il gruppo di sicurezza collegato all'endpoint VPC deve consentire le connessioni in entrata sulla porta 443 dalla sottorete privata del VPC.
- Gli endpoint VPC di Amazon ECR supportano la connettività IPv4 dual-stack (e). IPv6 Quando crei un endpoint VPC dual-stack, può gestire il traffico sia su indirizzi IP privati che su IPv4 indirizzi IP privati. IPv6

- Gli endpoint VPC supportano i repository Amazon ECR Public tramite l'endpoint AWS API SDK negli Stati Uniti orientali (Virginia settentrionale).
- Gli endpoint VPC supportano solo il DNS AWS fornito tramite Amazon Route 53. Se si desidera utilizzare il proprio DNS, è possibile usare l'inoltro condizionale sul DNS. Per ulteriori informazioni, consulta Set opzioni DHCP nella Guida per l'utente di Amazon VPC.
- Se i container dispongono di connessioni esistenti ad Amazon S3, le relative connessioni potrebbero essere interrotte per un breve periodo quando aggiungi l'endpoint gateway Amazon S3. Se si desidera evitare l'interruzione, creare un nuovo VPC che usi l'endpoint del gateway Amazon S3, quindi migrare il cluster Amazon ECS e i relativi container in un nuovo VPC.
- Quando un'immagine viene estratta utilizzando una regola di cache pull-through per la prima volta, se hai configurato Amazon ECR per l'utilizzo di un endpoint VPC di interfaccia tramite AWS PrivateLink , allora dovrai creare una sottorete pubblica nello stesso VPC, con un gateway NAT, quindi instradare tutto il traffico in uscita verso Internet dalla sottorete privata al gateway NAT per far funzionare il pull. Le operazioni successive di estrazione dell'immagine non richiedono questo passaggio. Per ulteriori informazioni, consulta Scenario: accesso a Internet da una sottorete privata nella Guida per l'utente di Amazon Virtual Private Cloud.
- Per i carichi di lavoro che richiedono moduli crittografici convalidati FIPS 140-3, Amazon ECR supporta gli endpoint FIPS per gli endpoint VPC.

Considerazioni per le immagini Windows

Le immagini basate sul sistema operativo Windows includono artefatti con limitazioni di licenza che impediscono la distribuzione. Per impostazione predefinita, quando si inviano immagini Windows a un repository Amazon ECR, i livelli che includono questi artefatti non vengono inviati in quanto vengono considerati livelli estranei. Quando gli artefatti vengono forniti da Microsoft, i livelli estranei vengono recuperati dall'infrastruttura di Microsoft Azure. Per questo motivo, per consentire ai container di estrarre questi livelli estranei da Azure sono necessari passaggi aggiuntivi oltre alla creazione degli endpoint VPC.

È possibile sovrascrivere questo comportamento quando si inviano le immagini Windows ad Amazon ECR utilizzando il flag `--allow-nondistributable-artifacts` nel daemon Docker. Quando è abilitato, questo flag invia i livelli concessi in licenza ad Amazon ECR, consentendo di estrarre queste immagini da Amazon ECR tramite l'endpoint VPC senza richiedere un ulteriore accesso ad Azure.

> **Important**
>
> L'utilizzo del flag `--allow-nondistributable-artifacts` non preclude l'obbligo dell'utente di rispettare i termini della licenza dell'immagine di base del container Windows; non è possibile pubblicare contenuti di Windows per la ridistribuzione pubblica o di terze parti. L'uso all'interno del proprio ambiente è consentito.

Per abilitare l'uso di questo flag per l'installazione Docker, è necessario modificare il file di configurazione del daemon Docker che, a seconda dell'installazione Docker, può in genere essere configurato nel menu delle impostazioni o delle preferenze nella sezione Docker Engine o modificando direttamente il file `C:\ProgramData\docker\config\daemon.json`.

Di seguito è illustrato un esempio della configurazione necessaria: Sostituisci il valore con l'URI del repository a cui stai inviando le immagini.

```
{
    "allow-nondistributable-artifacts": [
        "111122223333.dkr.ecr.us-west-2.amazonaws.com"
    ]
}
```

Dopo aver modificato il file di configurazione del daemon Docker, è necessario riavviare il daemon Docker prima di tentare di inviare l'immagine. Conferma che l'invio ha funzionato verificando che il livello base sia stato inviato nel repository.

> **Note**
>
> I livelli base per le immagini Windows sono grandi. Le dimensioni del livello si tradurranno in tempi più lunghi per l'invio e costi di archiviazione aggiuntivi in Amazon ECR per queste immagini. Per questi motivi, si consiglia di utilizzare questa opzione solo quando è strettamente necessaria per ridurre i tempi di costruzione e i costi di storage in corso. Ad esempio, l'immagine `mcr.microsoft.com/windows/servercore` è di circa 1,7 GiB di dimensioni quando viene compressa in Amazon ECR.

Creare gli endpoint VPC per Amazon ECR

Per creare gli endpoint VPC per Amazon ECR Service, utilizza la Creazione di un endpoint di interfaccia nella Amazon VPC User Guide.

Gli endpoint VPC di Amazon ECR supportano la connettività IPv4 dual-stack (e). IPv6 Quando crei un endpoint VPC dual-stack, gestisce automaticamente il traffico su IPv4 entrambi gli indirizzi IP e privati. IPv6 L'endpoint indirizzerà il traffico utilizzando la versione IP appropriata in base alla configurazione di rete del client e alle funzionalità dell'endpoint.

Se disponi di endpoint VPC esistenti IPv4 solo e desideri migrare verso endpoint dual-stack, puoi modificare gli endpoint esistenti per supportare la connettività dual-stack o creare nuovi endpoint dual-stack. Quando crei o modifichi gli endpoint, assicurati che il VPC e le sottoreti supportino la versione IP che desideri utilizzare. Dopo aver creato gli endpoint dual-stack, gli endpoint supporteranno entrambi e. IPv4 IPv6

Le attività di Amazon ECS ospitate su EC2 istanze Amazon richiedono sia gli endpoint Amazon ECR che l'endpoint gateway Amazon S3.

Le attività Amazon ECS con hosting su Fargate che utilizzano la versione della piattaforma `1.4.0` o versioni successive richiedono sia gli endpoint VPC Amazon ECR sia gli endpoint gateway Amazon S3.

Le attività di Amazon ECS ospitate su Fargate che utilizzano una **`1.3.0`** versione della piattaforma o precedente richiedono solo com.amazonaws. ***`region`***.ecr.dkr Endpoint VPC Amazon ECR ed endpoint gateway Amazon S3.

> **Note**
>
> L'ordine in cui vengono creati gli endpoint non è rilevante.

com.amazonaws. ***`region`***.ecr.dkr

Questo endpoint viene utilizzato per il registro Docker. APIs I comandi del client Docker come `push` e `pull` utilizzano questo endpoint.

Quando si crea questo endpoint, è necessario abilitare un nome host DNS privato. Per eseguire questa operazione, accertarsi che l'opzione Enable Private DNS Name (Abilita nome DNS privato) sia selezionata nella console Amazon VPC quando si crea l'endpoint VPC.

Per le connessioni conformi a FIPS 140-3, usa il nome dell'endpoint FIPS com.amazonaws. ***region***.ecr-fips.dkr

com.amazonaws. ***region***.ecr.api

> **Note**
>
> Lo specificato *region* rappresenta l'identificatore della regione per una AWS regione supportata da Amazon ECR, ad esempio `us-east-2` per la regione Stati Uniti orientali (Ohio).

Per le connessioni conformi allo standard FIPS 140-3, usa i nomi degli endpoint FIPS: com.amazonaws. *region*.ecr-fips.dkr e com.amazonaws. ***region***.ecr-fips.api.

Questo endpoint viene utilizzato per le chiamate all'API Amazon ECR. Operazioni API come `DescribeImages` e `CreateRepository` vanno su questo endpoint.

Quando viene creato questo endpoint, è possibile abilitare un nome host DNS privato. Abilita questo nome host selezionando Abilita nome DNS privato nella console VPC quando crei l'endpoint VPC. Se abiliti un nome host DNS privato per l'endpoint VPC, aggiorna l'SDK o AWS CLI alla versione più recente in modo che non sia necessario specificare un URL dell'endpoint quando si utilizza l'SDK o non sia necessario. AWS CLI

Per le connessioni conformi a FIPS 140-3, usa il nome dell'endpoint FIPS com.amazonaws. ***region***.ecr-fips.api.

Se abiliti un nome host DNS privato e utilizzi un SDK o una AWS CLI versione rilasciata prima del 24 gennaio 2019, devi utilizzare il parametro per specificare gli endpoint dell'interfaccia. `--endpoint-url` Nell'esempio seguente viene illustrato il formato per l'URL dell'endpoint.

```
aws ecr create-repository --repository-name name --endpoint-url https://
api.ecr.region.amazonaws.com
```

Se non si abilita un nome host DNS privato per l'endpoint VPC, è necessario utilizzare il parametro `--endpoint-url` specificando l'ID dell'endpoint VPC per l'endpoint di interfaccia. Nell'esempio seguente viene illustrato il formato per l'URL dell'endpoint.

```
aws ecr create-repository --repository-name name --endpoint-url
 https://VPC_endpoint_ID.api.ecr.region.vpce.amazonaws.com
```

Per le connessioni conformi a FIPS 140-3, utilizza l'URL dell'endpoint FIPS:

```
aws ecr create-repository --repository-name name --endpoint-url https://api.ecr-fips.region.amazonaws.com
```

Creare l'endpoint gateway Amazon S3

Affinché le attività Amazon ECS possano estrarre immagini private da Amazon ECR, è necessario creare un endpoint gateway per Amazon S3. L'endpoint gateway è obbligatorio perché Amazon ECR utilizza Amazon S3 per archiviare i livelli di immagine. Quando i container scaricano immagini da Amazon ECR, devono accedere a Amazon ECR per ottenere il manifest dell'immagine e quindi ad Amazon S3 per scaricare i livelli effettivi dell'immagine. Di seguito è riportato l'Amazon Resource Name (ARN) del bucket Amazon S3 che contiene i livelli per ogni immagine Docker.

```
arn:aws:s3:::prod-region-starport-layer-bucket/*
```

> **Note**
>
> Se crei un endpoint VPC dual-stack per Amazon ECR, devi anche creare un gateway o un endpoint di interfaccia Amazon S3 dual-stack. Per i dettagli, consulta la documentazione di S3.

Utilizzare la procedura Creazione di un endpoint gateway nella Amazon VPC User Guide per creare il seguente endpoint del gateway Amazon S3 per Amazon ECR. Quando crei l'endpoint, assicurati di selezionare le tabelle di routing per il VPC.

com.amazonaws. ***region***.s3

L'endpoint del gateway Amazon S3 usa un documento di policy IAM per limitare l'accesso al servizio. La policy Full Access (Accesso completo) può essere utilizzata poiché qualsiasi limitazione relativa ai ruoli IAM dell'attività o ad altre policy utente IAM viene comunque applicata. Se si desidera limitare l'accesso del bucket Amazon S3 alle autorizzazioni minime richieste necessarie per utilizzare Amazon ECR, consulta Autorizzazioni minime del bucket Amazon S3 per Amazon ECR.

Autorizzazioni minime del bucket Amazon S3 per Amazon ECR

L'endpoint del gateway Amazon S3 usa un documento di policy IAM per limitare l'accesso al servizio. Per consentire solo le autorizzazioni minime del bucket Amazon S3 per Amazon ECR, limita l'accesso al bucket Amazon S3 utilizzato da Amazon ECR quando crei il documento di policy IAM per l'endpoint.

La tabella seguente descrive le autorizzazioni della policy del bucket Amazon S3 richieste da Amazon ECR.

Autorizzazione	Description
`arn:aws:s3:::prod-` *`region`* `-starport-layer-bucket/*`	Fornisce l'accesso al bucket Amazon S3 contenente i livelli per ogni immagine Docker. Rappresenta l'identificatore di regione per una regione AWS supportata da Amazon ECR, ad esempio `us-east-2` per la regione Stati Uniti orientali (Ohio).

Esempio

L'esempio seguente spiega come fornire l'accesso ai bucket Amazon S3 richiesti per le operazioni Amazon ECR.

```
{
  "Statement": [
    {
      "Sid": "Access-to-specific-bucket-only",
      "Principal": "*",
      "Action": [
        "s3:GetObject"
      ],
      "Effect": "Allow",
      "Resource": ["arn:aws:s3:::prod-region-starport-layer-bucket/*"]
    }
  ]
}
```

Crea l'endpoint Logs CloudWatch

Le attività di Amazon ECS che utilizzano il tipo di avvio Fargate che utilizzano un VPC senza un gateway Internet e che utilizzano anche il driver di registro per inviare informazioni di registro CloudWatch a Logs richiedono **`awslogs`** la creazione del file com.amazonaws. ***`region`***Endpoint VPC con interfaccia.logs per Logs. CloudWatch Per ulteriori informazioni, consulta Using CloudWatch Logs with interface VPC endpoint nella CloudWatch Amazon Logs User Guide.

Creazione di una policy di endpoint per l'endpoint VPC di Amazon ECR

Una policy endpoint VPC è una policy della risorsa IAM che viene collegata a un endpoint durante la creazione o la modifica dell'endpoint. Se non alleghi una policy quando crei un endpoint, ti AWS allega una policy predefinita che consente l'accesso completo al servizio. Una policy endpoint non esclude né sostituisce policy dell'utente o policy specifiche del servizio. Si tratta di una policy separata per controllare l'accesso dall'endpoint al servizio specificato. Le policy endpoint devono essere scritte in formato JSON. Per ulteriori informazioni, consultare Controllo degli accessi ai servizi con endpoint VPC nella Guida per l'utente di Amazon VPC.

Ti consigliamo di creare un'unica policy di risorse IAM e di collegarla a entrambi gli endpoint VPC di Amazon ECR.

Di seguito è riportato un esempio di una policy endpoint per l'API di Amazon ECR. Questa policy consente a un ruolo IAM specifico di estrarre immagini da Amazon ECR.

```
{
 "Statement": [{
  "Sid": "AllowPull",
  "Principal": {
   "AWS": "arn:aws:iam::1234567890:role/role_name"
  },
  "Action": [
   "ecr:BatchGetImage",
   "ecr:GetDownloadUrlForLayer",
                  "ecr:GetAuthorizationToken"
  ],
  "Effect": "Allow",
  "Resource": "*"
 }]
}
```

La seguente policy endpoint di esempio impedisce l'eliminazione di un repository specificato.

```
{
 "Statement": [{
   "Sid": "AllowAll",
   "Principal": "*",
   "Action": "*",
   "Effect": "Allow",
   "Resource": "*"
  },
  {
   "Sid": "PreventDelete",
   "Principal": "*",
   "Action": "ecr:DeleteRepository",
   "Effect": "Deny",
   "Resource": "arn:aws:ecr:region:1234567890:repository/repository_name"
  }
 ]
}
```

L'esempio di policy endpoint seguente combina i due esempi precedenti in un'unica policy.

```
{
 "Statement": [{
   "Sid": "AllowAll",
   "Effect": "Allow",
   "Principal": "*",
   "Action": "*",
   "Resource": "*"
  },
  {
   "Sid": "PreventDelete",
   "Effect": "Deny",
   "Principal": "*",
   "Action": "ecr:DeleteRepository",
   "Resource": "arn:aws:ecr:region:1234567890:repository/repository_name"
  },
  {
   "Sid": "AllowPull",
   "Effect": "Allow",
   "Principal": {
    "AWS": "arn:aws:iam::1234567890:role/role_name"
   },
   "Action": [
    "ecr:BatchGetImage",
```

```
    "ecr:GetDownloadUrlForLayer",
                        "ecr:GetAuthorizationToken"
   ],
   "Resource": "*"
  }
 ]
}
```

Per modificare la policy endpoint VPC per Amazon ECR

1. Apri la console Amazon VPC all'indirizzo https://console.aws.amazon.com/vpc/.
2. Nel pannello di navigazione, seleziona Endpoint.
3. Se non hai già creato gli endpoint VPC per Amazon ECR, consulta Creare gli endpoint VPC per Amazon ECR.
4. Selezionare l'endpoint VPC di Amazon ECR a cui aggiungere una policy e scegliere la scheda Policy (Policy) nella metà inferiore della schermata.
5. Scegli Edit Policy (Modifica policy) e apporta le modifiche alla policy.
6. Selezionare Save (Salva) per salvare la policy.

Sottoreti condivise

Non puoi creare, descrivere, modificare o eliminare gli endpoint VPC nelle sottoreti condivise con te. Tuttavia, puoi utilizzare gli endpoint VPC in sottoreti condivise con te.

Prevenzione del confused deputy tra servizi

Il problema confused deputy è un problema di sicurezza in cui un'entità che non dispone dell'autorizzazione per eseguire un'azione può costringere un'entità maggiormente privilegiata a eseguire l'azione. Inoltre AWS, l'impersonificazione tra diversi servizi può portare al confuso problema del vicesceriffo. La rappresentazione tra servizi può verificarsi quando un servizio (il servizio chiamante) effettua una chiamata a un altro servizio (il servizio chiamato). Il servizio chiamante può essere manipolato per utilizzare le proprie autorizzazioni e agire sulle risorse di un altro cliente, a cui normalmente non avrebbe accesso. Per evitare ciò, AWS fornisce strumenti per poterti a proteggere i tuoi dati per tutti i servizi con entità di servizio a cui è stato concesso l'accesso alle risorse dell'account.

Ti consigliamo di utilizzare le chiavi di contesto delle condizioni globali `aws:SourceArn` o `aws:SourceAccount` nelle policy delle risorse per limitare le autorizzazioni con cui Amazon ECR fornisce un altro servizio alla risorsa. Utilizzare `aws:SourceArn` se si desidera consentire l'associazione di una sola risorsa all'accesso tra servizi. Utilizzare `aws:SourceAccount` se si desidera consentire l'associazione di qualsiasi risorsa in tale account all'uso tra servizi.

Il modo più efficace per proteggersi dal problema "confused deputy" è quello di utilizzare la chiave di contesto della condizione globale `aws:SourceArn` con l'ARN completo della risorsa. Se non si conosce l'ARN completo della risorsa o si scelgono più risorse, utilizzare la chiave di contesto della condizione globale `aws:SourceArn` con caratteri jolly (*) per le parti sconosciute dell'ARN. Ad esempio, `arn:aws:`*`servicename`*`:`*`region`*`:`*`123456789012`*`:*`.

Se il valore `aws:SourceArn` non contiene l'ID account, ad esempio un ARN di un bucket Amazon S3, è necessario utilizzare entrambe le chiavi di contesto delle condizioni globali per limitare le autorizzazioni.

Il valore di `aws:SourceArn` deve essere ResourceDescription.

L'esempio seguente mostra come utilizzare le chiavi di contesto della `aws:SourceArn` condizione `aws:SourceAccount` globale in una policy di repository Amazon ECR per consentire AWS CodeBuild l'accesso alle azioni dell'API Amazon ECR necessarie per l'integrazione con quel servizio, evitando al contempo il confuso problema del vice.

JSON

```
{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Sid":"CodeBuildAccess",
         "Effect":"Allow",
         "Principal":{
            "Service":"codebuild.amazonaws.com"
         },
         "Action":[
            "ecr:BatchGetImage",
            "ecr:GetDownloadUrlForLayer"
         ],
         "Resource": "*",
         "Condition":{
```

```
            "ArnLike":{
               "aws:SourceArn":"arn:aws:codebuild:us-
east-1:123456789012:project/project-name"
            },
            "StringEquals":{
               "aws:SourceAccount":"123456789012"
            }
         }
      }
   ]
}
```

Monitoraggio Amazon ECR

Puoi monitorare l'utilizzo dell'API Amazon ECR con Amazon CloudWatch, che raccoglie ed elabora i dati grezzi da Amazon ECR in metriche leggibili quasi in tempo reale. Queste statistiche vengono registrate per un periodo di due settimane in modo da poter accedere alle informazioni storiche e avere una prospettiva sull'utilizzo delle API. I dati metrici di Amazon ECR vengono inviati automaticamente CloudWatch in periodi di un minuto. Per ulteriori informazioni CloudWatch, consulta la Amazon CloudWatch User Guide.

Amazon ECR fornisce i parametri basati sull'utilizzo dell'API per le azioni di autorizzazione, invio ed estrazione di immagini.

Il monitoraggio è una parte importante per mantenere l'affidabilità, la disponibilità e le prestazioni di Amazon ECR e delle tue AWS soluzioni. Ti consigliamo di raccogliere i dati di monitoraggio dalle risorse che compongono la tua AWS soluzione in modo da poter eseguire più facilmente il debug di un errore multipunto, se si verifica. Prima di iniziare il monitoraggio di Amazon ECR è tuttavia opportuno creare un piano di monitoraggio che includa le risposte alle seguenti domande:

- Quali sono gli obiettivi del monitoraggio?
- Di quali risorse si intende eseguire il monitoraggio?
- Con quale frequenza sarà eseguito il monitoraggio di queste risorse?
- Quali strumenti di monitoraggio verranno utilizzati?
- Chi eseguirà i processi di monitoraggio?
- Chi deve ricevere una notifica quando si verifica un problema?

La fase successiva consiste nello stabilire una baseline per le prestazioni normali di Amazon ECR nell'ambiente, misurando le prestazioni in diversi momenti e con condizioni di carico differenti. Quando monitori Amazon ECR, archivia i dati di monitoraggio cronologici per poterli confrontare con i nuovi dati sulle prestazioni e per poter identificare i normali modelli di prestazioni e le anomalie e ideare metodi per risolvere i problemi.

Argomenti

- Parametri del repository Amazon ECR
- Eventi Amazon ECR e EventBridge
- Registrazione delle azioni Amazon ECR con AWS CloudTrail

Visualizzazione delle quote di servizio e impostazione degli allarmi

Puoi utilizzare la CloudWatch console per visualizzare le quote di servizio e confrontare l'utilizzo attuale con le quote di servizio. Puoi anche impostare gli allarmi per ricevere una notifica quando ti avvicini a una quota.

Per visualizzare una quota di servizio e impostare facoltativamente un allarme

1. Apri la console all'indirizzo. CloudWatch https://console.aws.amazon.com/cloudwatch/
2. Nel riquadro di navigazione, seleziona Parametri.
3. Nella scheda All metrics (Tutti i parametri), scegli Usage (Utilizzo), quindi seleziona By AWS Resource (Per risorsa).

   Viene visualizzato l'elenco dei parametri di utilizzo delle quote di servizio.
4. Seleziona la casella di controllo accanto a uno dei parametri.

   Il grafico mostra l'utilizzo corrente di quella AWS risorsa.
5. Per aggiungere la quota di servizio al grafico, procedere come indicato di seguito:

   a. Seleziona la scheda Graphed metrics (Parametri nel grafico).

   b. Scegli Math expression (Espressione matematica), Start with an empty expression (Inizia con un'espressione vuota). Quindi nella nuova riga, in Details (Dettagli), immettere **`SERVICE_QUOTA(m1)`**.

      Una nuova riga viene aggiunta al grafico, visualizzando la quota di servizio per la risorsa rappresentata nel parametro.
6. Per visualizzare l'utilizzo corrente come una percentuale della quota, aggiungere una nuova espressione o modificare l'espressione SERVICE_QUOTA corrente. Per la nuova espressione, utilizzare **`m1/60/SERVICE_QUOTA(m1)*100`**
7. (Facoltativo) Per impostare un allarme che avvisa se ci si avvicina alla quota di servizio, procedere nel modo seguente:

a. Nella riga **`m1/60/SERVICE_QUOTA(m1)*100`**, in Actions (Operazioni), scegliere l'icona di allarme. L'aspetto è simile quello di una campana.

   Viene visualizzata la pagina di creazione dell'allarme.

b. In Conditions (Condizioni), assicurarti che Threshold type (Tipo di soglia) sia Static (Statico) e Whenever Expression1 is (Ogni volta che Expression1 è) sia impostato su Greater (Maggiore). In than (di), immetti **`80`**. Viene creato un allarme che passo nello stato ALARM quando l'utilizzo supera l'80% della quota.

c. Scegli Next (Successivo).

d. Nella pagina successiva, selezionare un argomento Amazon SNS o crearne uno nuovo. Questo argomento riceve una notifica quando l'allarme passa nello stato ALLARME. Quindi scegli Successivo.

e. Nella pagina successiva, immetti un nome e una descrizione per l'allarme, quindi scegli Next (Avanti).

f. Scegli Crea allarme.

Parametri di utilizzo Amazon ECR

Puoi utilizzare le metriche di CloudWatch utilizzo per fornire visibilità sull'utilizzo delle risorse da parte del tuo account. Utilizza queste metriche per visualizzare l'utilizzo corrente del servizio su CloudWatch grafici e dashboard.

I parametri di utilizzo di Amazon ECR corrispondono alle quote di AWS servizio. È possibile configurare gli allarmi che avvisano quando l'uso si avvicina a una quota di servizio. Per ulteriori informazioni sulle quote di servizio per Amazon ECR, consulta Service Quotas di Amazon ECR..

Amazon ECR pubblica i seguenti parametri nello spazio dei nomi `AWS/Usage`.

Metrica	Description
`CallCount`	Il numero di chiamate di operazione API dal tuo account. Le risorse sono definite dalle dimensioni associate al parametro. La statistica più utile per questo parametro è `SUM`, che rappresenta la somma dei valori di tutti i collaboratori durante il periodo definito.

Le seguenti dimensioni vengono utilizzate per perfezionare i parametri di utilizzo pubblicati da Amazon ECR.

Dimensione	Description
`Service`	Il nome del AWS servizio che contiene la risorsa. Per i parametri di utilizzo di Amazon ECR, il valore per questa dimensione è `ECR`.
`Type`	Il tipo di entità che viene segnalato. Attualmente, l'unico valore valido per i parametri di utilizzo Amazon ECR è `API`.
`Resource`	Il tipo di risorsa in esecuzione. Attualmente, Amazon ECR restituisce informazioni sull'utilizzo dell'API per le seguenti operazioni API. • `GetAuthorizationToken` • `BatchCheckLayerAvailability` • `InitiateLayerUpload` • `UploadLayerPart` • `CompleteLayerUpload` • `PutImage` • `BatchGetImage` • `GetDownloadUrlForLayer`
`Class`	La classe della risorsa monitorata. Attualmente, Amazon ECR non utilizza la dimensione della classe.

Report di utilizzo di Amazon ECR

AWS fornisce uno strumento di reporting gratuito chiamato Cost Explorer che consente di analizzare il costo e l'utilizzo delle risorse Amazon ECR.

Utilizza Cost Explorer per visualizzare i grafici relativi all'utilizzo e ai costi. Puoi visualizzare i dati degli ultimi 13 mesi e prevedere le spese per i successivi tre mesi. Puoi utilizzare Cost Explorer per vedere l'andamento della spesa sulle risorse AWS nel tempo, identificare le aree che necessitano di

maggiore attenzione e vedere le tendenze che puoi utilizzare per comprendere i costi. Puoi anche specificare intervalli di tempo per i dati e visualizzare i dati temporali per mese o per giorno.

I dati di misurazione nel report su costi e utilizzo mostrano l'utilizzo in tutti i repository Amazon ECR. Per ulteriori informazioni, consulta Tagging delle risorse per la fatturazione.

Per ulteriori informazioni sulla creazione di un report sui AWS costi e sull'utilizzo, consulta il report AWS sui costi e sull'utilizzo nella Guida per l'AWS Billing utente.

Parametri del repository Amazon ECR

Amazon ECR invia ad Amazon i parametri del pull count dei repository. CloudWatch I dati metrici di Amazon ECR vengono inviati automaticamente CloudWatch in periodi di 1 minuto. Per ulteriori informazioni CloudWatch, consulta la Amazon CloudWatch User Guide.

Argomenti

- Abilitazione delle CloudWatch metriche
- Parametri e dimensioni disponibili
- Visualizzazione dei parametri di Amazon ECR tramite la console CloudWatch

Abilitazione delle CloudWatch metriche

Amazon ECR invia automaticamente i parametri del repository per tutti i repository. Non è necessario intraprendere alcuna procedura manuale.

Parametri e dimensioni disponibili

Nelle seguenti sezioni sono elencate le metriche e le dimensioni che Amazon ECR invia ad Amazon. CloudWatch

Parametri Amazon ECR

Amazon ECR fornisce dei parametri per monitorare i repository. È possibile misurare il numero di pull.

Lo spazio dei nomi `AWS/ECR` include i parametri descritti di seguito.

`RepositoryPullCount`

Il numero totale di pull per le immagini nel repository.

Dimensioni valide: `RepositoryName`.

Statistiche valide: Average (Media), Minimum (Minimo), Maximum (Massimo), Sum (Somma), Sample Count (Conteggio campione). La statistica più utile è somma.

Unità: numero intero

Dimensioni per i parametri Amazon ECR

I parametri di Amazon ECR utilizzano lo spazio dei nomi `AWS/ECR` e forniscono i parametri per le seguenti dimensioni.

`RepositoryName`

Questa dimensione filtra i dati richiesti per tutte le immagini del container in un repository specificato.

Visualizzazione dei parametri di Amazon ECR tramite la console CloudWatch

Puoi visualizzare i parametri del repository Amazon ECR sulla console. CloudWatch La CloudWatch console offre una visualizzazione dettagliata e personalizzabile delle tue risorse. Per ulteriori informazioni, consulta la Amazon CloudWatch User Guide.

Eventi Amazon ECR e EventBridge

Amazon ti EventBridge consente di automatizzare AWS i tuoi servizi e di rispondere automaticamente a eventi di sistema come problemi di disponibilità delle applicazioni o modifiche delle risorse.
Gli eventi AWS relativi ai servizi vengono forniti quasi EventBridge in tempo reale. Puoi scrivere regole semplici che indichino quali eventi sono considerati di interesse per te e includere le azioni automatizzate da intraprendere quando un evento corrisponde a una regola. Le azioni che possono essere attivate automaticamente includono le seguenti:

- Aggiungere eventi ai gruppi di log in CloudWatch Logs
- Invocare una funzione AWS Lambda
- Richiamo del comando Amazon EC2 Run
- Inoltro dell'evento a Amazon Kinesis Data Streams

- Attivazione di una macchina a stati AWS Step Functions
- Notifica di un argomento Amazon SNS o di una coda Amazon SQS

Per ulteriori informazioni, consulta la sezione Getting Started with Amazon EventBridge nella Amazon EventBridge User Guide.

Esempi di eventi da Amazon ECR

Di seguito sono riportati eventi di esempio di Amazon ECR. Gli eventi vengono emessi secondo il principio del massimo sforzo.

Evento per il push di un'immagine completata

Il seguente evento viene inviato al termine di ogni push dell'immagine. Per ulteriori informazioni, consulta Trasferimento di un'immagine Docker a un repository privato Amazon ECR.

```
{
    "version": "0",
    "id": "13cde686-328b-6117-af20-0e5566167482",
    "detail-type": "ECR Image Action",
    "source": "aws.ecr",
    "account": "123456789012",
    "time": "2019-11-16T01:54:34Z",
    "region": "us-west-2",
    "resources": [],
    "detail": {
        "result": "SUCCESS",
        "repository-name": "my-repository-name",
        "image-digest":
 "sha256:7f5b2640fe6fb4f46592dfd3410c4a79dac4f89e4782432e0378abcd1234",
        "action-type": "PUSH",
        "image-tag": "latest"
    }
}
```

Evento per un'azione di cache pull-through

Il seguente evento viene inviato quando viene tentata un'azione di cache pull-through. Per ulteriori informazioni, consulta Sincronizzazione di un registro upstream con un registro privato Amazon ECR.

```
{
```

```
    "version": "0",
    "id": "85fc3613-e913-7fc4-a80c-a3753e4aa9ae",
    "detail-type": "ECR Pull Through Cache Action",
    "source": "aws.ecr",
    "account": "123456789012",
    "time": "2023-02-29T02:36:48Z",
    "region": "us-west-2",
    "resources": [
        "arn:aws:ecr:us-west-2:123456789012:repository/docker-hub/alpine"
    ],
    "detail": {
        "rule-version": "1",
        "sync-status": "SUCCESS",
        "ecr-repository-prefix": "docker-hub",
        "repository-name": "docker-hub/alpine",
        "upstream-registry-url": "public.ecr.aws",
        "image-tag": "3.17.2",
        "image-digest":
 "sha256:4aa08ef415aecc80814cb42fa41b658480779d80c77ab15EXAMPLE",
    }
}
```

Evento per la scansione completa di un'immagine (scansione di base)

Quando la scansione di base è abilitata per il registro, viene inviato il seguente evento al termine di ogni scansione dell'immagine. Il parametro `finding-severity-counts` restituirà un valore per un livello di gravità solo se ne esiste uno. Ad esempio, se l'immagine non contiene risultati a livello `CRITICAL`, non viene restituito alcun conteggio critico. Per ulteriori informazioni, consulta Scansiona le immagini per individuare le vulnerabilità del sistema operativo in Amazon ECR.

> **Note**
>
> Per informazioni dettagliate sugli eventi emessi da Amazon Inspector quando è abilitata la scansione avanzata, consulta EventBridge eventi inviati per una scansione avanzata in Amazon ECR.

```
{
    "version": "0",
    "id": "85fc3613-e913-7fc4-a80c-a3753e4aa9ae",
    "detail-type": "ECR Image Scan",
```

```
    "source": "aws.ecr",
    "account": "123456789012",
    "time": "2019-10-29T02:36:48Z",
    "region": "us-east-1",
    "resources": [
        "arn:aws:ecr:us-east-1:123456789012:repository/my-repository-name"
    ],
    "detail": {
        "scan-status": "COMPLETE",
        "repository-name": "my-repository-name",
        "finding-severity-counts": {
        "CRITICAL": 10,
        "MEDIUM": 9
      },
        "image-digest":
 "sha256:7f5b2640fe6fb4f46592dfd3410c4a79dac4f89e4782432e0378abcd1234",
        "image-tags": []
    }
}
```

Evento per una notifica di modifica su una risorsa con scansione avanzata abilitata (scansione avanzata)

Quando la scansione avanzata è abilitata per il registro, il seguente evento viene inviato da Amazon ECR quando si verifica una modifica con una risorsa che ha abilitato la scansione avanzata. Ciò include la creazione di nuovi repository, la modifica della frequenza di scansione di un repository o la creazione o eliminazione di immagini nei repository con la scansione avanzata abilitata. Per ulteriori informazioni, consulta Scansiona le immagini per individuare le vulnerabilità del software in Amazon ECR.

```
{
 "version": "0",
 "id": "0c18352a-a4d4-6853-ef53-0ab8638973bf",
 "detail-type": "ECR Scan Resource Change",
 "source": "aws.ecr",
 "account": "123456789012",
 "time": "2021-10-14T20:53:46Z",
 "region": "us-east-1",
 "resources": [],
 "detail": {
  "action-type": "SCAN_FREQUENCY_CHANGE",
  "repositories": [{
```

```
    "repository-name": "repository-1",
    "repository-arn": "arn:aws:ecr:us-east-1:123456789012:repository/repository-1",
    "scan-frequency": "SCAN_ON_PUSH",
    "previous-scan-frequency": "MANUAL"
   },
   {
    "repository-name": "repository-2",
    "repository-arn": "arn:aws:ecr:us-east-1:123456789012:repository/repository-2",
    "scan-frequency": "CONTINUOUS_SCAN",
    "previous-scan-frequency": "SCAN_ON_PUSH"
   },
   {
    "repository-name": "repository-3",
    "repository-arn": "arn:aws:ecr:us-east-1:123456789012:repository/repository-3",
    "scan-frequency": "CONTINUOUS_SCAN",
    "previous-scan-frequency": "SCAN_ON_PUSH"
   }
  ],
  "resource-type": "REPOSITORY",
  "scan-type": "ENHANCED"
 }
}
```

Evento per l'eliminazione di un'immagine

Il seguente evento viene inviato quando un'immagine viene eliminata. Per ulteriori informazioni, consulta Eliminazione di un'immagine in Amazon ECR.

```
{
    "version": "0",
    "id": "dd3b46cb-2c74-f49e-393b-28286b67279d",
    "detail-type": "ECR Image Action",
    "source": "aws.ecr",
    "account": "123456789012",
    "time": "2019-11-16T02:01:05Z",
    "region": "us-west-2",
    "resources": [],
    "detail": {
        "result": "SUCCESS",
        "repository-name": "my-repository-name",
        "image-digest":
 "sha256:7f5b2640fe6fb4f46592dfd3410c4a79dac4f89e4782432e0378abcd1234",
        "action-type": "DELETE",
```

```
        "image-tag": "latest"
    }
}
```

Evento per un'azione di archiviazione di immagini

Il seguente evento viene inviato quando un'immagine viene archiviata. Il `target-storage-class` campo verrà impostato su. `ARCHIVE` L'evento include i tipi di media manifest e artifact per identificare il tipo di contenuto da archiviare.

```
{
    "version": "0",
    "id": "4f5ec4d5-4de4-7aad-a046-EXAMPLE",
    "detail-type": "ECR Image Action",
    "source": "aws.ecr",
    "account": "123456789012",
    "time": "2019-08-06T00:58:09Z",
    "region": "us-east-1",
    "resources": [],
    "detail": {
        "action-type": "UPDATE_STORAGE_CLASS",
        "target-storage-class": "ARCHIVE",
        "image-digest":
 "sha256:f98d67af8e53a536502bfc600de3266556b06ed635a32d60aa7a5fe6d7e609d7",
        "repository-name": "ubuntu",
        "result": "SUCCESS",
        "manifest-media-type": "application/vnd.oci.image.manifest.v1+json",
        "artifact-media-type": "application/vnd.oci.image.config.v1+json"
    }
}
```

Evento relativo a un'azione di ripristino dell'immagine

Il seguente evento viene inviato quando viene ripristinata un'immagine archiviata. Il `target-storage-class` campo verrà impostato su. `STANDARD` L'evento include un `last-activated-at` campo che mostra quando l'immagine è stata ripristinata l'ultima volta.

```
{
    "version": "0",
    "id": "7b8fc5e6-5ef5-8bbe-b157-EXAMPLE",
    "detail-type": "ECR Image Action",
    "source": "aws.ecr",
```

```
    "account": "123456789012",
    "time": "2019-08-06T01:15:22Z",
    "region": "us-east-1",
    "resources": [],
    "detail": {
        "action-type": "UPDATE_STORAGE_CLASS",
        "target-storage-class": "STANDARD",
        "image-digest":
 "sha256:f98d67af8e53a536502bfc600de3266556b06ed635a32d60aa7a5fe6d7e609d7",
        "repository-name": "ubuntu",
        "result": "SUCCESS",
        "manifest-media-type": "application/vnd.oci.image.manifest.v1+json",
        "artifact-media-type": "application/vnd.oci.image.config.v1+json",
        "last-activated-at": "2025-10-10T19:13:02.74Z"
    }
}
```

Evento relativo a un'azione di ripristino del referente

L'evento seguente viene inviato quando viene ripristinato un referrer archiviato (elemento di riferimento come un SBOM, una firma o un attestato). Nota che serve a distinguerlo dalle normali azioni `detail-type` relative `ECR Referrer Action` all'immagine. I `artifact-media-type` campi `manifest-media-type` e identificano il tipo specifico di referente da ripristinare. In questo esempio, viene ripristinato un elemento SBOM.

```
{
    "version": "0",
    "id": "8c9gd6f7-6fg6-9ccf-c268-EXAMPLE",
    "detail-type": "ECR Referrer Action",
    "source": "aws.ecr",
    "account": "123456789012",
    "time": "2019-08-06T01:20:45Z",
    "region": "us-east-1",
    "resources": [],
    "detail": {
        "action-type": "UPDATE_STORAGE_CLASS",
        "target-storage-class": "STANDARD",
        "image-digest":
 "sha256:f98d67af8e53a536502bfc600de3266556b06ed635a32d60aa7a5fe6d7e609d7",
        "repository-name": "sbom",
        "result": "SUCCESS",
        "manifest-media-type": "application/vnd.cncf.oras.artifact.manifest.v1+json",
        "artifact-media-type": "text/sbom+json",
```

```
        "last-activated-at": "2025-10-10T19:13:02.74Z"
    }
}
```

Evento per una replica dell'immagine completata

Il seguente evento viene inviato al termine di ogni replica dell'immagine. Per ulteriori informazioni, consulta Private image replication in Amazon ECR.

```
{
  "version": "0",
  "id": "c8b133b1-6029-ee73-e2a1-4f466b8ba999",
  "detail-type": "ECR Replication Action",
  "source": "aws.ecr",
  "account": "123456789012",
  "time": "2024-05-08T20:44:54Z",
  "region": "us-east-1",
  "resources": [
    "arn:aws:ecr:us-east-1:123456789012:repository/docker-hub/alpine"
  ],
  "detail": {
    "result": "SUCCESS",
    "repository-name": "docker-hub/alpine",
    "image-digest":
 "sha256:7f5b2640fe6fb4f46592dfd3410c4a79dac4f89e4782432e0378abcd1234",
    "source-account": "123456789012",
    "action-type": "REPLICATE",
    "source-region": "us-west-2",
    "image-tag": "3.17.2"
  }
}
```

Evento relativo a una replica dell'immagine non riuscita

Il seguente evento viene inviato quando una replica dell'immagine fallisce. Il `result` campo conterrà `FAILED` e nei dettagli dell'evento potrebbero essere incluse ulteriori informazioni sull'errore.

```
{
  "version": "0",
  "id": "d9c244c2-7130-ff84-f3b2-5g577c9cb000",
  "detail-type": "ECR Replication Action",
  "source": "aws.ecr",
```

```
  "account": "123456789012",
  "time": "2024-05-08T20:45:12Z",
  "region": "us-east-1",
  "resources": [
    "arn:aws:ecr:us-east-1:123456789012:repository/my-app"
  ],
  "detail": {
    "result": "FAILED",
    "repository-name": "my-app",
    "image-digest":
 "sha256:8g6c3751gf7gc5g47603ege4511d5a80ead5g90f5893543f1489bde2345",
    "source-account": "123456789012",
    "action-type": "REPLICATE",
    "source-region": "us-west-2",
    "image-tag": "latest"
  }
}
```

Registrazione delle azioni Amazon ECR con AWS CloudTrail

Amazon ECR è integrato con AWS CloudTrail, un servizio che fornisce un registro delle azioni intraprese da un utente, un ruolo o un AWS servizio in Amazon ECR. CloudTrail acquisisce le seguenti azioni Amazon ECR come eventi:

- Tutte le chiamate API, incluse le chiamate dalla console Amazon ECR
- Tutte le operazioni intraprese a causa delle impostazioni di crittografia nei repository
- Tutte le operazioni eseguite a causa delle regole delle policy relative al ciclo di vita, incluse le operazioni riuscite e non riuscite

> **Important**
>
> A causa dei limiti di dimensione dei singoli CloudTrail eventi, per le azioni relative alle politiche del ciclo di vita in cui 10 o più immagini sono scadute, Amazon ECR invia più eventi a. CloudTrail Inoltre, Amazon ECR include un massimo di 100 tag per immagine.

Quando viene creato un trail, puoi abilitare la distribuzione continua di CloudTrail eventi a un bucket Amazon S3, inclusi gli eventi per Amazon ECR. Se non configuri un percorso, puoi comunque visualizzare gli eventi più recenti nella CloudTrail console nella cronologia degli eventi. Utilizzando

queste informazioni, puoi determinare la richiesta effettuata ad Amazon ECR, l'indirizzo IP da cui è stata effettuata la richiesta, l'autore della richiesta, il momento in cui è stata effettuata e altri dettagli.

Per ulteriori informazioni, consulta la Guida per l'utente AWS CloudTrail.

Informazioni su Amazon ECR in CloudTrail

CloudTrail è abilitato sul tuo AWS account al momento della creazione dell'account. Quando si verifica un'attività in Amazon ECR, tale attività viene registrata in un CloudTrail evento insieme ad altri eventi di AWS servizio nella cronologia degli eventi. Puoi visualizzare, cercare e scaricare eventi recenti nel tuo AWS account. Per ulteriori informazioni, consulta Visualizzazione degli eventi con la cronologia degli CloudTrail eventi.

Per una registrazione continua degli eventi nel tuo AWS account, inclusi gli eventi per Amazon ECR, crea un percorso. Un trail consente di CloudTrail inviare file di log a un bucket Amazon S3. Quando si crea un percorso nella console, è possibile applicarlo a una singola regione o a tutte le regioni geografiche. Il trail registra gli eventi nella AWS partizione e consegna i file di log al bucket Amazon S3 specificato. Inoltre, puoi configurare altri AWS servizi per analizzare e agire in base ai dati sugli eventi raccolti nei log. CloudTrail Per ulteriori informazioni, consulta:

- Creare un percorso per il tuo account AWS
- AWS integrazioni di servizi con log CloudTrail
- Configurazione delle notifiche Amazon SNS per CloudTrail
- Ricezione di file di CloudTrail registro da più regioni e ricezione di file di CloudTrail registro da più account

Tutte le azioni dell'API Amazon ECR vengono registrate CloudTrail e documentate nell'Amazon Elastic Container Registry API Reference. Quando esegui attività comuni, nei file di CloudTrail registro vengono generate sezioni per ogni azione API che fa parte di tale attività. Ad esempio, quando si crea un repository`GetAuthorizationToken`, `CreateRepository` le `SetRepositoryPolicy` sezioni vengono generate nei file di CloudTrail registro. Quando esegui il push di un'immagine in un repository, vengono generate le sezioni `InitiateLayerUpload`, `UploadLayerPart`, `CompleteLayerUpload` e `PutImage`. Quando esegui il pull di un'immagine, vengono generate le sezioni `GetDownloadUrlForLayer` e `BatchGetImage`. Quando si archivia o si ripristina un'immagine, viene generata una `UpdateImageStorageClass` sezione. Quando OCI i client che supportano la OCI 1.1 specifica recuperano l'elenco di referrer, o artefatti di riferimento, per

un'immagine utilizzando l'API Referrers, viene emesso un evento. `ListImageReferrers` CloudTrail Per esempi di attività comuni, consulta CloudTrail esempi di immissione di log.

Ogni evento o voce di log contiene informazioni sull'utente che ha generato la richiesta. Le informazioni di identità consentono di determinare quanto segue:

- Se la richiesta è stata effettuata con le credenziali utente o root.
- Se la richiesta è stata effettuata con le credenziali di sicurezza temporanee per un ruolo o un utente federato.
- Se la richiesta è stata effettuata da un altro servizio AWS

Per ulteriori informazioni, consulta l'elemento CloudTrail `userIdentity`.

Informazioni sulle voci del file di log Amazon ECR

Un trail è una configurazione che consente la distribuzione di eventi come file di log in un bucket Amazon S3 specificato dall'utente. CloudTrail i file di registro contengono una o più voci di registro. Un evento rappresenta una singola richiesta proveniente da qualsiasi fonte e include informazioni sull'azione richiesta, la data e l'ora dell'azione, i parametri della richiesta e altre informazioni. CloudTrail i file di registro non sono una traccia ordinata delle chiamate API pubbliche, quindi non vengono visualizzati in un ordine specifico.

CloudTrail esempi di immissione di log

Di seguito sono riportati esempi di immissione di CloudTrail log per alcune attività comuni di Amazon ECR.

Questi esempi sono stati formattati per migliorare la leggibilità. In un file di CloudTrail registro, tutte le voci e gli eventi sono concatenati in un'unica riga. Inoltre, questo esempio è limitato a una singola voce Amazon ECR. In un vero file di CloudTrail registro, puoi vedere voci ed eventi provenienti da più servizi. AWS

> **Important**
>
> L'origine IPAddress è l'indirizzo IP da cui è stata effettuata la richiesta. Per le azioni che provengono dalla console di servizio, l'indirizzo riportato è relativo alla risorsa sottostante, non al server Web della console. Per i servizi in AWS, viene visualizzato solo il nome DNS.

Valutiamo comunque l'autenticazione con l'IP di origine del client anche se è stato modificato in base al nome DNS del AWS servizio.

Argomenti

Esempio: crea operazione del repository

L'esempio seguente mostra una voce di CloudTrail registro che illustra l'azione. `CreateRepository`


```
{
    "eventVersion": "1.04",
    "userIdentity": {
        "type": "AssumedRole",
        "principalId": "AIDACKCEVSQ6C2EXAMPLE:account_name",
        "arn": "arn:aws:sts::123456789012:user/Mary_Major",
        "accountId": "123456789012",
        "accessKeyId": "AKIAIOSFODNN7EXAMPLE",
        "sessionContext": {
            "attributes": {
                "mfaAuthenticated": "false",
                "creationDate": "2018-07-11T21:54:07Z"
            },
            "sessionIssuer": {
                "type": "Role",
                "principalId": "AIDACKCEVSQ6C2EXAMPLE",
                "arn": "arn:aws:iam::123456789012:role/Admin",
                "accountId": "123456789012",
                "userName": "Admin"
            }
```

```
        }
    },
    "eventTime": "2018-07-11T22:17:43Z",
    "eventSource": "ecr.amazonaws.com",
    "eventName": "CreateRepository",
    "awsRegion": "us-east-2",
    "sourceIPAddress": "203.0.113.12",
    "userAgent": "console.amazonaws.com",
    "requestParameters": {
        "repositoryName": "testrepo"
    },
    "responseElements": {
        "repository": {
            "repositoryArn": "arn:aws:ecr:us-east-2:123456789012:repository/testrepo",
            "repositoryName": "testrepo",
            "repositoryUri": "123456789012.dkr.ecr.us-east-2.amazonaws.com/testrepo",
            "createdAt": "Jul 11, 2018 10:17:44 PM",
            "registryId": "123456789012"
        }
    },
    "requestID": "cb8c167e-EXAMPLE",
    "eventID": "e3c6f4ce-EXAMPLE",
    "resources": [
        {
            "ARN": "arn:aws:ecr:us-east-2:123456789012:repository/testrepo",
            "accountId": "123456789012"
        }
    ],
    "eventType": "AwsApiCall",
    "recipientAccountId": "123456789012"
}
```

Esempio: azione dell' AWS KMS**`CreateGrant`**API durante la creazione di un repository Amazon ECR

L'esempio seguente mostra una voce di CloudTrail registro che illustra l' AWS KMS `CreateGrant`azione da eseguire durante la creazione di un repository Amazon ECR con crittografia KMS abilitata. Per ogni repository creato con la crittografia KMS abilitata, dovresti vedere due voci di registro. `CreateGrant` CloudTrail

```
{
    "eventVersion": "1.05",
    "userIdentity": {
```

```
        "type": "IAMUser",
        "principalId": "AIDAIEP6W46J43IG7LXAQ",
        "arn": "arn:aws:iam::123456789012:user/Mary_Major",
        "accountId": "123456789012",
        "accessKeyId": "AKIAIOSFODNN7EXAMPLE",
        "userName": "Mary_Major",
        "sessionContext": {
            "sessionIssuer": {

            },
            "webIdFederationData": {

            },
            "attributes": {
                "mfaAuthenticated": "false",
                "creationDate": "2020-06-10T19:22:10Z"
            }
        },
        "invokedBy": "AWS Internal"
    },
    "eventTime": "2020-06-10T19:22:10Z",
    "eventSource": "kms.amazonaws.com",
    "eventName": "CreateGrant",
    "awsRegion": "us-west-2",
    "sourceIPAddress": "203.0.113.12",
    "userAgent": "console.amazonaws.com",
    "requestParameters": {
        "keyId": "4b55e5bf-39c8-41ad-b589-18464af7758a",
        "granteePrincipal": "ecr.us-west-2.amazonaws.com",
        "operations": [
            "GenerateDataKey",
            "Decrypt"
        ],
        "retiringPrincipal": "ecr.us-west-2.amazonaws.com",
        "constraints": {
            "encryptionContextSubset": {
                "aws:ecr:arn": "arn:aws:ecr:us-west-2:123456789012:repository/testrepo"
            }
        }
    },
    "responseElements": {
        "grantId": "3636af9adfee1accb67b83941087dcd45e7fadc4e74ff0103bb338422b5055f3"
    },
    "requestID": "047b7dea-b56b-4013-87e9-a089f0f6602b",
```

```
    "eventID": "af4c9573-c56a-4886-baca-a77526544469",
    "readOnly": false,
    "resources": [
        {
            "accountId": "123456789012",
            "type": "AWS::KMS::Key",
            "ARN": "arn:aws:kms:us-west-2:123456789012:key/4b55e5bf-39c8-41ad-
b589-18464af7758a"
        }
    ],
    "eventType": "AwsApiCall",
    "recipientAccountId": "123456789012"
}
```

Esempio: operazione di invio immagine

L'esempio seguente mostra una voce di CloudTrail registro che mostra un'immagine push che utilizza l'azione. `PutImage`

> **Note**
>
> Quando inserite un'immagine`InitiateLayerUpload`, vedrete anche `UploadLayerPart` i `CompleteLayerUpload` riferimenti nei CloudTrail log.

```
{
    "eventVersion": "1.04",
    "userIdentity": {
    "type": "IAMUser",
    "principalId": "AIDACKCEVSQ6C2EXAMPLE:account_name",
    "arn": "arn:aws:sts::123456789012:user/Mary_Major",
    "accountId": "123456789012",
    "accessKeyId": "AKIAIOSFODNN7EXAMPLE",
  "userName": "Mary_Major",
  "sessionContext": {
   "attributes": {
    "mfaAuthenticated": "false",
    "creationDate": "2019-04-15T16:42:14Z"
   }
  }
 },
 "eventTime": "2019-04-15T16:45:00Z",
```

```
 "eventSource": "ecr.amazonaws.com",
 "eventName": "PutImage",
 "awsRegion": "us-east-2",
 "sourceIPAddress": "AWS Internal",
 "userAgent": "AWS Internal",
 "requestParameters": {
  "repositoryName": "testrepo",
  "imageTag": "latest",
  "registryId": "123456789012",
  "imageManifest": "{\n   \"schemaVersion\": 2,\n   \"mediaType\": \"application/
vnd.docker.distribution.manifest.v2+json\",\n   \"config\": {\n      \"mediaType\":
 \"application/vnd.docker.container.image.v1+json\",\n      \"size\": 5543,\n
 \"digest\": \"sha256:000b9b805af1cdb60628898c9f411996301a1c13afd3dbef1d8a16ac6dbf503a
\"\n   },\n   \"layers\": [\n      {\n         \"mediaType\": \"application/
vnd.docker.image.rootfs.diff.tar.gzip\",\n         \"size\": 43252507,\n
 \"digest\": \"sha256:3b37166ec61459e76e33282dda08f2a9cd698ca7e3d6bc44e6a6e7580cdeff8e
\"\n      },\n      {\n         \"mediaType\": \"application/
vnd.docker.image.rootfs.diff.tar.gzip\",\n         \"size\": 846,\n         \"digest
\": \"sha256:504facff238fde83f1ca8f9f54520b4219c5b8f80be9616ddc52d31448a044bd
\"\n      },\n      {\n         \"mediaType\": \"application/
vnd.docker.image.rootfs.diff.tar.gzip\",\n         \"size\": 615,\n         \"digest
\": \"sha256:ebbcacd28e101968415b0c812b2d2dc60f969e36b0b08c073bf796e12b1bb449\"\n
      },\n      {\n         \"mediaType\": \"application/
vnd.docker.image.rootfs.diff.tar.gzip\",\n         \"size\": 850,\n         \"digest
\": \"sha256:c7fb3351ecad291a88b92b600037e2435c84a347683d540042086fe72c902b8a
\"\n      },\n      {\n         \"mediaType\": \"application/
vnd.docker.image.rootfs.diff.tar.gzip\",\n         \"size\": 168,\n         \"digest\":
 \"sha256:2e3debadcbf7e542e2aefbce1b64a358b1931fb403b3e4aeca27cb4d809d56c2\"\n      },
\n      {\n         \"mediaType\": \"application/vnd.docker.image.rootfs.diff.tar.gzip
\",\n         \"size\": 37720774,\n         \"digest\":
 \"sha256:f8c9f51ad524d8ae9bf4db69cd3e720ba92373ec265f5c390ffb21bb0c277941\"\n
      },\n      {\n         \"mediaType\": \"application/
vnd.docker.image.rootfs.diff.tar.gzip\",\n         \"size\": 30432107,\n
 \"digest\": \"sha256:813a50b13f61cf1f8d25f19fa96ad3aa5b552896c83e86ce413b48b091d7f01b
\"\n      },\n      {\n         \"mediaType\": \"application/
vnd.docker.image.rootfs.diff.tar.gzip\",\n         \"size\": 197,\n         \"digest
\": \"sha256:7ab043301a6187ea3293d80b30ba06c7bf1a0c3cd4c43d10353b31bc0cecfe7d
\"\n      },\n      {\n         \"mediaType\": \"application/
vnd.docker.image.rootfs.diff.tar.gzip\",\n         \"size\": 154,\n         \"digest
\": \"sha256:67012cca8f31dc3b8ee2305e7762fee20c250513effdedb38a1c37784a5a2e71\"\n
      },\n      {\n         \"mediaType\": \"application/
vnd.docker.image.rootfs.diff.tar.gzip\",\n         \"size\": 176,\n         \"digest
\": \"sha256:3bc892145603fffc9b1c97c94e2985b4cb19ca508750b15845a5d97becbd1a0e
\"\n      },\n      {\n         \"mediaType\": \"application/
```

```
vnd.docker.image.rootfs.diff.tar.gzip\",\n         \"size\": 183,\n         \"digest
\": \"sha256:6f1c79518f18251d35977e7e46bfa6c6b9cf50df2a79d4194941d95c54258d18\"\n
      },\n      {\n         \"mediaType\": \"application/
vnd.docker.image.rootfs.diff.tar.gzip\",\n         \"size\": 212,\n         \"digest
\": \"sha256:b7bcfbc2e2888afebede4dd1cd5eebf029bb6315feeaf0b56e425e11a50afe42\"\n
      },\n      {\n         \"mediaType\": \"application/
vnd.docker.image.rootfs.diff.tar.gzip\",\n         \"size\": 212,\n         \"digest\":
 \"sha256:2b220f8b0f32b7c2ed8eaafe1c802633bbd94849b9ab73926f0ba46cdae91629\"\n      }\n
   ]\n}"
 },
 "responseElements": {
  "image": {
   "repositoryName": "testrepo",
   "imageManifest": "{\n   \"schemaVersion\": 2,\n   \"mediaType\": \"application/
vnd.docker.distribution.manifest.v2+json\",\n   \"config\": {\n      \"mediaType\":
 \"application/vnd.docker.container.image.v1+json\",\n      \"size\": 5543,\n
 \"digest\": \"sha256:000b9b805af1cdb60628898c9f411996301a1c13afd3dbef1d8a16ac6dbf503a
\"\n   },\n   \"layers\": [\n      {\n         \"mediaType\": \"application/
vnd.docker.image.rootfs.diff.tar.gzip\",\n         \"size\": 43252507,\n
 \"digest\": \"sha256:3b37166ec61459e76e33282dda08f2a9cd698ca7e3d6bc44e6a6e7580cdeff8e
\"\n      },\n      {\n         \"mediaType\": \"application/
vnd.docker.image.rootfs.diff.tar.gzip\",\n         \"size\": 846,\n         \"digest
\": \"sha256:504facff238fde83f1ca8f9f54520b4219c5b8f80be9616ddc52d31448a044bd
\"\n      },\n      {\n         \"mediaType\": \"application/
vnd.docker.image.rootfs.diff.tar.gzip\",\n         \"size\": 615,\n         \"digest
\": \"sha256:ebbcacd28e101968415b0c812b2d2dc60f969e36b0b08c073bf796e12b1bb449\"\n
      },\n      {\n         \"mediaType\": \"application/
vnd.docker.image.rootfs.diff.tar.gzip\",\n         \"size\": 850,\n         \"digest
\": \"sha256:c7fb3351ecad291a88b92b600037e2435c84a347683d540042086fe72c902b8a
\"\n      },\n      {\n         \"mediaType\": \"application/
vnd.docker.image.rootfs.diff.tar.gzip\",\n         \"size\": 168,\n         \"digest\":
 \"sha256:2e3debadcbf7e542e2aefbce1b64a358b1931fb403b3e4aeca27cb4d809d56c2\"\n      },
\n      {\n         \"mediaType\": \"application/vnd.docker.image.rootfs.diff.tar.gzip
\",\n         \"size\": 37720774,\n         \"digest\":
 \"sha256:f8c9f51ad524d8ae9bf4db69cd3e720ba92373ec265f5c390ffb21bb0c277941\"\n
      },\n      {\n         \"mediaType\": \"application/
vnd.docker.image.rootfs.diff.tar.gzip\",\n         \"size\": 30432107,\n
 \"digest\": \"sha256:813a50b13f61cf1f8d25f19fa96ad3aa5b552896c83e86ce413b48b091d7f01b
\"\n      },\n      {\n         \"mediaType\": \"application/
vnd.docker.image.rootfs.diff.tar.gzip\",\n         \"size\": 197,\n         \"digest
\": \"sha256:7ab043301a6187ea3293d80b30ba06c7bf1a0c3cd4c43d10353b31bc0cecfe7d
\"\n      },\n      {\n         \"mediaType\": \"application/
vnd.docker.image.rootfs.diff.tar.gzip\",\n         \"size\": 154,\n         \"digest
\": \"sha256:67012cca8f31dc3b8ee2305e7762fee20c250513effdedb38a1c37784a5a2e71\"\n
```

```
      },\n      {\n         \"mediaType\": \"application/
vnd.docker.image.rootfs.diff.tar.gzip\",\n         \"size\": 176,\n         \"digest
\": \"sha256:3bc892145603fffc9b1c97c94e2985b4cb19ca508750b15845a5d97becbd1a0e
\"\n      },\n      {\n         \"mediaType\": \"application/
vnd.docker.image.rootfs.diff.tar.gzip\",\n         \"size\": 183,\n         \"digest
\": \"sha256:6f1c79518f18251d35977e7e46bfa6c6b9cf50df2a79d4194941d95c54258d18\"\n
      },\n      {\n         \"mediaType\": \"application/
vnd.docker.image.rootfs.diff.tar.gzip\",\n         \"size\": 212,\n         \"digest
\": \"sha256:b7bcfbc2e2888afebede4dd1cd5eebf029bb6315feeaf0b56e425e11a50afe42\"\n
      },\n      {\n         \"mediaType\": \"application/
vnd.docker.image.rootfs.diff.tar.gzip\",\n         \"size\": 212,\n         \"digest\":
 \"sha256:2b220f8b0f32b7c2ed8eaafe1c802633bbd94849b9ab73926f0ba46cdae91629\"\n      }\n
   ]\n}",
   "registryId": "123456789012",
   "imageId": {
    "imageDigest":
 "sha256:98c8b060c21d9adbb6b8c41b916e95e6307102786973ab93a41e8b86d1fc6d3e",
    "imageTag": "latest"
   }
  }
 },
 "requestID": "cf044b7d-5f9d-11e9-9b2a-95983139cc57",
 "eventID": "2bfd4ee2-2178-4a82-a27d-b12939923f0f",
 "resources": [{
  "ARN": "arn:aws:ecr:us-east-2:123456789012:repository/testrepo",
  "accountId": "123456789012"
 }],
 "eventType": "AwsApiCall",
 "recipientAccountId": "123456789012"
}
```


Esempio: operazione di estrazione immagine

L'esempio seguente mostra una voce di CloudTrail registro che mostra un'immagine pull che utilizza l'azione. `BatchGetImage`

> **Note**
>
> Quando esegui il pull di un'immagine, se non hai ancora l'immagine localmente, vedrai anche i riferimenti `GetDownloadUrlForLayer` nei log CloudTrail .

```
{
    "eventVersion": "1.04",
    "userIdentity": {
    "type": "IAMUser",
    "principalId": "AIDACKCEVSQ6C2EXAMPLE:account_name",
    "arn": "arn:aws:sts::123456789012:user/Mary_Major",
    "accountId": "123456789012",
    "accessKeyId": "AKIAIOSFODNN7EXAMPLE",
  "userName": "Mary_Major",
  "sessionContext": {
   "attributes": {
    "mfaAuthenticated": "false",
    "creationDate": "2019-04-15T16:42:14Z"
   }
  }
 },
 "eventTime": "2019-04-15T17:23:20Z",
 "eventSource": "ecr.amazonaws.com",
 "eventName": "BatchGetImage",
 "awsRegion": "us-east-2",
 "sourceIPAddress": "ecr.amazonaws.com",
 "userAgent": "ecr.amazonaws.com",
 "requestParameters": {
  "imageIds": [{
   "imageTag": "latest"
  }],
  "acceptedMediaTypes": [
   "application/json",
   "application/vnd.oci.image.manifest.v1+json",
   "application/vnd.oci.image.index.v1+json",
   "application/vnd.docker.distribution.manifest.v2+json",
   "application/vnd.docker.distribution.manifest.list.v2+json",
   "application/vnd.docker.distribution.manifest.v1+prettyjws"
  ],
  "repositoryName": "testrepo",
  "registryId": "123456789012"
 },
 "responseElements": null,
 "requestID": "2a1b97ee-5fa3-11e9-a8cd-cd2391aeda93",
 "eventID": "c84f5880-c2f9-4585-9757-28fa5c1065df",
 "resources": [{
  "ARN": "arn:aws:ecr:us-east-2:123456789012:repository/testrepo",
  "accountId": "123456789012"
```

```
 }],
 "eventType": "AwsApiCall",
 "recipientAccountId": "123456789012"
}
```

Esempio: operazione delle policy relative al ciclo di vita delle immagini

L'esempio seguente mostra una voce di CloudTrail registro che mostra quando un'immagine è scaduta a causa di una regola del ciclo di vita. Questo tipo di evento può essere individuato applicando un filtro `PolicyExecutionEvent` per il campo del nome dell'evento.

Quando si testa un'anteprima di una politica del ciclo di vita, Amazon ECR genera una voce di CloudTrail registro con il campo del nome dell'evento di`DryRunEvent`, con la stessa struttura di. `PolicyExecutionEvent` Modificando il nome dell'evento in`DryRunEvent`, puoi invece filtrare gli eventi di dry run.

> **Important**
>
> A causa dei limiti di dimensione dei singoli CloudTrail eventi, per le azioni relative alle politiche del ciclo di vita in cui 10 o più immagini sono scadute, Amazon ECR invia più eventi a. CloudTrail Inoltre, Amazon ECR include un massimo di 100 tag per immagine.

```
{
    "eventVersion": "1.05",
    "userIdentity": {
        "accountId": "123456789012",
        "invokedBy": "AWS Internal"
    },
    "eventTime": "2020-03-12T20:22:12Z",
    "eventSource": "ecr.amazonaws.com",
    "eventName": "PolicyExecutionEvent",
    "awsRegion": "us-west-2",
    "sourceIPAddress": "AWS Internal",
    "userAgent": "AWS Internal",
    "requestParameters": null,
    "responseElements": null,
    "eventID": "9354dd7f-9aac-4e9d-956d-12561a4923aa",
    "readOnly": true,
    "resources": [
        {
```

```
            "ARN": "arn:aws:ecr:us-west-2:123456789012:repository/testrepo",
            "accountId": "123456789012",
            "type": "AWS::ECR::Repository"
        }
    ],
    "eventType": "AwsServiceEvent",
    "recipientAccountId": "123456789012",
    "serviceEventDetails": {
        "repositoryName": "testrepo",
        "lifecycleEventPolicy": {
            "lifecycleEventRules": [
                {
                    "rulePriority": 1,
                    "description": "remove all images > 2",
                    "lifecycleEventSelection": {
                        "tagStatus": "Any",
                        "tagPrefixList": [],
                        "countType": "Image count more than",
                        "countNumber": 2
                    },
                    "action": "expire"
                }
            ],
            "lastEvaluatedAt": 0,
            "policyVersion": 1,
            "policyId": "ceb86829-58e7-9498-920c-aa042e33037b"
        },
        "lifecycleEventImageActions": [
            {
                "lifecycleEventImage": {
                    "digest":
"sha256:ddba4d27a7ffc3f86dd6c2f92041af252a1f23a8e742c90e6e1297bfa1bc0c45",
                    "tagStatus": "Tagged",
                    "tagList": [
                        "alpine"
                    ],
                    "pushedAt": 1584042813000
                },
                "rulePriority": 1
            },
            {
                "lifecycleEventImage": {
                    "digest":
"sha256:6ab380c5a5acf71c1b6660d645d2cd79cc8ce91b38e0352cbf9561e050427baf",
```

```
                    "tagStatus": "Tagged",
                    "tagList": [
                        "centos"
                    ],
                    "pushedAt": 1584042842000
                },
                "rulePriority": 1
            }
        ],
        "lifecycleEventFailureDetails": [
            {
                "lifecycleEventImage": {
                    "digest":
 "sha256:9117e1bc28cd20751e584b4ccd19b1178d14cf02d134b04ce6be0cc51bff762a",
                    "tagStatus": "Untagged",
                    "tagList": [],
                    "pushedAt": 1584042844000
                },
                "rulePriority": 1,
                "failureCode": "ImageReferencedByManifestList",
                "failureReason": "Requested image referenced by manifest list:
 [sha256:4b27c83d44a18c31543039d9e8b2786043ec6c8d00804d5800c5148d6b6f65bc]"
            }
        ]
    }
}
```

Esempio: azione di archiviazione delle immagini

L'esempio seguente mostra una voce di CloudTrail registro che dimostra che un'immagine viene archiviata utilizzando l'`UpdateImageStorageClass`azione con `targetStorageClass` set to. `ARCHIVE`

```
{
    "eventVersion": "1.11",
    "userIdentity": {
        "type": "IAMUser",
        "principalId": "AIDACKCEVSQ6C2EXAMPLE:account_name",
        "arn": "arn:aws:sts::123456789012:user/Mary_Major",
        "accountId": "123456789012",
        "accessKeyId": "AKIAIOSFODNN7EXAMPLE",
  "userName": "Mary_Major",
  "sessionContext": {
```

```
   "attributes": {
    "mfaAuthenticated": "false",
    "creationDate": "2019-04-15T16:42:14Z"
   }
  }
 },
 "eventTime": "2019-04-15T16:45:00Z",
 "eventSource": "ecr.amazonaws.com",
 "eventName": "UpdateImageStorageClass",
 "awsRegion": "us-east-2",
 "sourceIPAddress": "AWS Internal",
 "userAgent": "AWS Internal",
 "requestParameters": {
  "repositoryName": "testrepo",
  "imageId": {
   "imageDigest":
 "sha256:98c8b060c21d9adbb6b8c41b916e95e6307102786973ab93a41e8b86d1fc6d3e"
  },
  "targetStorageClass": "ARCHIVE",
  "registryId": "123456789012"
 },
 "responseElements": {
  "image": {
   "registryId": "123456789012",
   "repositoryName": "testrepo",
   "imageId": {
    "imageDigest":
 "sha256:98c8b060c21d9adbb6b8c41b916e95e6307102786973ab93a41e8b86d1fc6d3e"
   },
   "imageStatus": "ARCHIVED"
  }
 },
 "requestID": "cf044b7d-EXAMPLE",
 "eventID": "2bfd4ee2-EXAMPLE",
 "readOnly": false,
 "resources": [{
  "ARN": "arn:aws:ecr:us-east-2:123456789012:repository/testrepo",
  "accountId": "123456789012"
 }],
 "eventType": "AwsApiCall",
 "managementEvent": true,
 "recipientAccountId": "123456789012",
 "eventCategory": "Management"
```

```
}
```

Esempio: azione di ripristino dell'immagine

Gli esempi seguenti mostrano le voci di CloudTrail registro che dimostrano il ripristino di un'immagine. Quando si ripristina un'immagine archiviata, vengono generati due eventi:

1. Un evento di chiamata API all'avvio del ripristino
2. Un evento di servizio al termine dell'operazione di ripristino asincrona

Evento di chiamata API (avvio del ripristino)

L'esempio seguente mostra la chiamata API iniziale per ripristinare un'immagine utilizzando l'`UpdateImageStorageClass`azione con `targetStorageClass` set to`STANDARD`. La risposta mostra lo stato dell'immagine come`ACTIVATING`.

```
{
    "eventVersion": "1.11",
    "userIdentity": {
        "type": "IAMUser",
        "principalId": "AIDACKCEVSQ6C2EXAMPLE:account_name",
        "arn": "arn:aws:sts::123456789012:user/Mary_Major",
        "accountId": "123456789012",
        "accessKeyId": "AKIAIOSFODNN7EXAMPLE",
  "userName": "Mary_Major",
  "sessionContext": {
   "attributes": {
    "mfaAuthenticated": "false",
    "creationDate": "2019-04-15T16:42:14Z"
   }
  }
 },
 "eventTime": "2019-04-15T16:45:00Z",
 "eventSource": "ecr.amazonaws.com",
 "eventName": "UpdateImageStorageClass",
 "awsRegion": "us-east-2",
 "sourceIPAddress": "AWS Internal",
 "userAgent": "AWS Internal",
 "requestParameters": {
  "repositoryName": "testrepo",
  "imageId": {
```

```
    "imageDigest":
 "sha256:98c8b060c21d9adbb6b8c41b916e95e6307102786973ab93a41e8b86d1fc6d3e"
   },
   "targetStorageClass": "STANDARD",
   "registryId": "123456789012"
 },
 "responseElements": {
   "image": {
    "registryId": "123456789012",
    "repositoryName": "testrepo",
    "imageId": {
     "imageDigest":
 "sha256:98c8b060c21d9adbb6b8c41b916e95e6307102786973ab93a41e8b86d1fc6d3e"
    },
    "imageStatus": "ACTIVATING"
   }
 },
 "requestID": "cf044b7d-EXAMPLE",
 "eventID": "2bfd4ee2-EXAMPLE",
 "readOnly": false,
 "resources": [{
   "ARN": "arn:aws:ecr:us-east-2:123456789012:repository/testrepo",
   "accountId": "123456789012"
 }],
 "eventType": "AwsApiCall",
 "managementEvent": true,
 "recipientAccountId": "123456789012",
 "eventCategory": "Management"
}
```

Evento di servizio (completamento del ripristino)

L'esempio seguente mostra l'evento di servizio generato al termine dell'operazione di ripristino asincrona. Questo tipo di evento può essere individuato applicando un filtro `ImageActivationEvent` per il campo del nome dell'evento. La `serviceEventDetails` sezione contiene il risultato del ripristino e lo stato finale dell'immagine.

```
{
    "eventVersion": "1.11",
    "userIdentity": {
        "accountId": "123456789012",
        "invokedBy": "AWS Internal"
    },
```

```
    "eventTime": "2020-03-12T20:22:12Z",
    "eventSource": "ecr.amazonaws.com",
    "eventName": "ImageActivationEvent",
    "awsRegion": "us-west-2",
    "sourceIPAddress": "AWS Internal",
    "userAgent": "AWS Internal",
    "requestParameters": null,
    "responseElements": null,
    "eventID": "9354dd7f-EXAMPLE",
    "readOnly": true,
    "resources": [
        {
            "ARN": "arn:aws:ecr:us-west-2:123456789012:repository/testrepo",
            "accountId": "123456789012",
            "type": "AWS::ECR::Repository"
        }
    ],
    "eventType": "AwsServiceEvent",
    "managementEvent": true,
    "recipientAccountId": "123456789012",
    "serviceEventDetails": {
        "repositoryName": "testrepo",
        "imageDigest":
 "sha256:98c8b060c21d9adbb6b8c41b916e95e6307102786973ab93a41e8b86d1fc6d3e",
        "targetStorageClass": "STANDARD",
        "result": "SUCCESS",
        "imageStatus": "ACTIVE"
    },
    "eventCategory": "Management"
}
```

Esempio: azione Image Referrers

L'esempio seguente mostra una voce di AWS CloudTrail registro che mostra quando un client OCI 1.1 conforme recupera un elenco di referrer, o artefatti di riferimento, per un'immagine utilizzando l'API. `Referrers`

```
{
    "eventVersion": "1.08",
    "userIdentity": {
        "type": "AssumedRole",
        "principalId": "AIDACKCEVSQ6C2EXAMPLE:account_name",
        "arn": "arn:aws:sts::123456789012:user/Mary_Major",
```

```
        "accountId": "123456789012",
        "accessKeyId": "AKIAIOSFODNN7EXAMPLE",
        "sessionContext": {
            "sessionIssuer": {
                "type": "Role",
                "principalId": "AIDACKCEVSQ6C2EXAMPLE",
                "arn": "arn:aws:iam::123456789012:role/Admin",
                "accountId": "123456789012",
                "userName": "Admin"
            },
            "webIdFederationData": {},
            "attributes": {
                "creationDate": "2024-10-08T16:38:39Z",
                "mfaAuthenticated": "false"
            },
            "ec2RoleDelivery": "2.0"
        },
        "invokedBy": "ecr.amazonaws.com"
    },
    "eventTime": "2024-10-08T17:22:51Z",
    "eventSource": "ecr.amazonaws.com",
    "eventName": "ListImageReferrers",
    "awsRegion": "us-east-2",
    "sourceIPAddress": "ecr.amazonaws.com",
    "userAgent": "ecr.amazonaws.com",
    "requestParameters": {
        "registryId": "123456789012",
        "repositoryName": "testrepo",
        "subjectId": {
            "imageDigest":
"sha256:000b9b805af1cdb60628898c9f411996301a1c13afd3dbef1d8a16ac6dbf503a"
        },
        "nextToken": "urD72mdD/mC8b5-EXAMPLE"
    },
    "responseElements": null,
    "requestID": "cb8c167e-EXAMPLE",
    "eventID": "e3c6f4ce-EXAMPLE",
    "readOnly": true,
    "resources": [
        {
            "accountId": "123456789012",
            "ARN": "arn:aws:ecr:us-east-2:123456789012:repository/testrepo"
        }
    ],
```

```
    "eventType": "AwsApiCall",
    "managementEvent": true,
    "recipientAccountId": "123456789012",
    "eventCategory": "Management"
}
```

Utilizzo di Amazon ECR con un SDK AWS

AWS i kit di sviluppo software (SDKs) sono disponibili per molti linguaggi di programmazione più diffusi. Ogni SDK fornisce un'API, esempi di codice e documentazione che facilitano agli sviluppatori la creazione di applicazioni nel loro linguaggio preferito.

Documentazione sugli SDK	Esempi di codice
AWS SDK per C++	AWS SDK per C++ esempi di codice
AWS CLI	AWS CLI esempi di codice
AWS SDK per Go	AWS SDK per Go esempi di codice
AWS SDK per Java	AWS SDK per Java esempi di codice
AWS SDK per JavaScript	AWS SDK per JavaScript esempi di codice
AWS SDK per Kotlin	AWS SDK per Kotlin esempi di codice
AWS SDK per .NET	AWS SDK per .NET esempi di codice
AWS SDK per PHP	AWS SDK per PHP esempi di codice
AWS Strumenti per PowerShell	AWS Strumenti per PowerShell esempi di codice
AWS SDK per Python (Boto3)	AWS SDK per Python (Boto3) esempi di codice
AWS SDK per Ruby	AWS SDK per Ruby esempi di codice
AWS SDK per Rust	AWS SDK per Rust esempi di codice
AWS SDK per SAP ABAP	AWS SDK per SAP ABAP esempi di codice
AWS SDK per Swift	AWS SDK per Swift esempi di codice

Esempio di disponibilità

Non riesci a trovare quello che ti serve? Richiedi un esempio di codice utilizzando il link Fornisci un feedback nella parte inferiore di questa pagina.

Esempi di codice per l'utilizzo di Amazon ECR AWS SDKs

I seguenti esempi di codice mostrano come usare Amazon ECR con un kit di sviluppo AWS software (SDK).

Nozioni di base: esempi di codice che mostrano come eseguire le operazioni essenziali all'interno di un servizio.

Le azioni sono estratti di codice da programmi più grandi e devono essere eseguite nel contesto. Sebbene le azioni mostrino come richiamare le singole funzioni del servizio, è possibile visualizzarle contestualizzate negli scenari correlati.

Per un elenco completo di guide ed esempi di codice per sviluppatori AWS SDK, consulta. Utilizzo di Amazon ECR con un SDK AWS Questo argomento include anche informazioni su come iniziare e dettagli sulle versioni precedenti dell'SDK.

Nozioni di base

Hello Amazon ECR

Gli esempi di codice seguenti mostrano come iniziare a utilizzare Amazon ECR.

Java

SDK per Java 2.x

> **Note**
>
> C'è altro su. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.ecr.EcrClient;
import software.amazon.awssdk.services.ecr.model.EcrException;
import software.amazon.awssdk.services.ecr.model.ListImagesRequest;
import software.amazon.awssdk.services.ecr.paginators.ListImagesIterable;
```

```
public class HelloECR {

    public static void main(String[] args) {
        final String usage = """
            Usage:    <repositoryName>

            Where:
               repositoryName - The name of the Amazon ECR repository.
            """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String repoName = args[0];
        EcrClient ecrClient = EcrClient.builder()
            .region(Region.US_EAST_1)
            .build();

        listImageTags(ecrClient, repoName);
    }
    public static void listImageTags(EcrClient ecrClient, String repoName){
        ListImagesRequest listImagesPaginator = ListImagesRequest.builder()
            .repositoryName(repoName)
            .build();

        ListImagesIterable imagesIterable =
 ecrClient.listImagesPaginator(listImagesPaginator);
        imagesIterable.stream()
            .flatMap(r -> r.imageIds().stream())
            .forEach(image -> System.out.println("The docker image tag is: "
 +image.imageTag()));
    }
}
```

- Per informazioni dettagliate sull'API, consulta listImages nella documentazione di riferimento dell'API AWS SDK for Java 2.x .

Kotlin

SDK per Kotlin

> **Note**
>
> C'è dell'altro GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
import aws.sdk.kotlin.services.ecr.EcrClient
import aws.sdk.kotlin.services.ecr.model.ListImagesRequest
import kotlin.system.exitProcess

suspend fun main(args: Array<String>) {
    val usage = """
            Usage: <repositoryName>

            Where:
               repositoryName - The name of the Amazon ECR repository.

    """.trimIndent()

    if (args.size != 1) {
        println(usage)
        exitProcess(1)
    }

    val repoName = args[0]
    listImageTags(repoName)
}

suspend fun listImageTags(repoName: String?) {
    val listImages =
        ListImagesRequest {
            repositoryName = repoName
        }

    EcrClient.fromEnvironment { region = "us-east-1" }.use { ecrClient ->
        val imageResponse = ecrClient.listImages(listImages)
        imageResponse.imageIds?.forEach { imageId ->
            println("Image tag: ${imageId.imageTag}")
```

```
        }
    }
}
```

- Per informazioni dettagliate sull'API, consulta listImages nella documentazione di riferimento dell'API AWS SDK per Kotlin.

Python

SDK per Python (Boto3)

> **Note**
>
> C'è dell'altro GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
import boto3
import argparse
from boto3 import client


def hello_ecr(ecr_client: client, repository_name: str) -> None:
    """
    Use the AWS SDK for Python (Boto3) to create an Amazon Elastic Container
 Registry (Amazon ECR)
    client and list the images in a repository.
    This example uses the default settings specified in your shared credentials
    and config files.

    :param ecr_client: A Boto3 Amazon ECR Client object. This object wraps
                           the low-level Amazon ECR service API.
    :param repository_name: The name of an Amazon ECR repository in your account.
    """
    print(
        f"Hello, Amazon ECR! Let's list some images in the repository
 '{repository_name}':\n"
    )
    paginator = ecr_client.get_paginator("list_images")
    page_iterator = paginator.paginate(
```

```
        repositoryName=repository_name, PaginationConfig={"MaxItems": 10}
    )

    image_names: [str] = []
    for page in page_iterator:
        for schedule in page["imageIds"]:
            image_names.append(schedule["imageTag"])

    print(f"{len(image_names)} image(s) retrieved.")
    for schedule_name in image_names:
        print(f"\t{schedule_name}")


if __name__ == "__main__":
    parser = argparse.ArgumentParser(description="Run hello Amazon ECR.")
    parser.add_argument(
        "--repository-name",
        type=str,
        help="the name of an Amazon ECR repository in your account.",
        required=True,
    )
    args = parser.parse_args()

    hello_ecr(boto3.client("ecr"), args.repository_name)
```

- Per informazioni dettagliate sull'API, consulta listImages nella documentazione di riferimento dell'API AWS SDK per Python (Boto3).

Esempi di codice

- Esempi di base per l'utilizzo di Amazon ECR AWS SDKs
  - Hello Amazon ECR
  - Scopri le nozioni di base di Amazon ECR con un SDK AWS
  - Azioni per l'utilizzo di Amazon ECR AWS SDKs
    - Utilizzo CreateRepository con un AWS SDK o una CLI
    - Utilizzo DeleteRepository con un AWS SDK o una CLI
    - Utilizzo DescribeImages con un AWS SDK o una CLI
    - Utilizzo DescribeRepositories con un AWS SDK o una CLI
    - Utilizzo GetAuthorizationToken con un AWS SDK o una CLI

- Utilizzo GetRepositoryPolicy con un AWS SDK o una CLI
- Utilizzo ListImages con un AWS SDK o una CLI
- Utilizzare PushImageCmd con un SDK AWS
- Utilizzo PutLifeCyclePolicy con un AWS SDK o una CLI
- Utilizzo SetRepositoryPolicy con un AWS SDK o una CLI
- Utilizzo StartLifecyclePolicyPreview con un AWS SDK o una CLI

Esempi di base per l'utilizzo di Amazon ECR AWS SDKs

I seguenti esempi di codice mostrano come utilizzare le nozioni di base di Amazon Elastic Container Registry con AWS SDKs.

Esempi

- Hello Amazon ECR
- Scopri le nozioni di base di Amazon ECR con un SDK AWS
- Azioni per l'utilizzo di Amazon ECR AWS SDKs
  - Utilizzo CreateRepository con un AWS SDK o una CLI
  - Utilizzo DeleteRepository con un AWS SDK o una CLI
  - Utilizzo DescribeImages con un AWS SDK o una CLI
  - Utilizzo DescribeRepositories con un AWS SDK o una CLI
  - Utilizzo GetAuthorizationToken con un AWS SDK o una CLI
  - Utilizzo GetRepositoryPolicy con un AWS SDK o una CLI
  - Utilizzo ListImages con un AWS SDK o una CLI
  - Utilizzare PushImageCmd con un SDK AWS
  - Utilizzo PutLifeCyclePolicy con un AWS SDK o una CLI
  - Utilizzo SetRepositoryPolicy con un AWS SDK o una CLI
  - Utilizzo StartLifecyclePolicyPreview con un AWS SDK o una CLI

Hello Amazon ECR

Gli esempi di codice seguenti mostrano come iniziare a utilizzare Amazon ECR.

Java

SDK per Java 2.x

> **Note**
>
> C'è altro su. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.ecr.EcrClient;
import software.amazon.awssdk.services.ecr.model.EcrException;
import software.amazon.awssdk.services.ecr.model.ListImagesRequest;
import software.amazon.awssdk.services.ecr.paginators.ListImagesIterable;

public class HelloECR {

    public static void main(String[] args) {
        final String usage = """
            Usage:    <repositoryName>

            Where:
               repositoryName - The name of the Amazon ECR repository.
            """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String repoName = args[0];
        EcrClient ecrClient = EcrClient.builder()
            .region(Region.US_EAST_1)
            .build();

        listImageTags(ecrClient, repoName);
    }
    public static void listImageTags(EcrClient ecrClient, String repoName){
        ListImagesRequest listImagesPaginator = ListImagesRequest.builder()
            .repositoryName(repoName)
            .build();
```

```
        ListImagesIterable imagesIterable =
 ecrClient.listImagesPaginator(listImagesPaginator);
        imagesIterable.stream()
            .flatMap(r -> r.imageIds().stream())
            .forEach(image -> System.out.println("The docker image tag is: "
 +image.imageTag()));
    }
}
```

- Per informazioni dettagliate sull'API, consulta listImages nella documentazione di riferimento dell'API AWS SDK for Java 2.x .

Kotlin

SDK per Kotlin

> **Note**
>
> C'è dell'altro GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
import aws.sdk.kotlin.services.ecr.EcrClient
import aws.sdk.kotlin.services.ecr.model.ListImagesRequest
import kotlin.system.exitProcess

suspend fun main(args: Array<String>) {
    val usage = """
            Usage: <repositoryName>

            Where:
               repositoryName - The name of the Amazon ECR repository.

    """.trimIndent()

    if (args.size != 1) {
        println(usage)
        exitProcess(1)
    }
```

```
    val repoName = args[0]
    listImageTags(repoName)
}

suspend fun listImageTags(repoName: String?) {
    val listImages =
        ListImagesRequest {
            repositoryName = repoName
        }

    EcrClient.fromEnvironment { region = "us-east-1" }.use { ecrClient ->
        val imageResponse = ecrClient.listImages(listImages)
        imageResponse.imageIds?.forEach { imageId ->
            println("Image tag: ${imageId.imageTag}")
        }
    }
}
```

- Per informazioni dettagliate sull'API, consulta listImages nella documentazione di riferimento dell'API AWS SDK per Kotlin.

Python

SDK per Python (Boto3)

> **Note**
>
> C'è dell'altro GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
import boto3
import argparse
from boto3 import client


def hello_ecr(ecr_client: client, repository_name: str) -> None:
    """
```

```
    Use the AWS SDK for Python (Boto3) to create an Amazon Elastic Container
 Registry (Amazon ECR)
    client and list the images in a repository.
    This example uses the default settings specified in your shared credentials
    and config files.

    :param ecr_client: A Boto3 Amazon ECR Client object. This object wraps
                           the low-level Amazon ECR service API.
    :param repository_name: The name of an Amazon ECR repository in your account.
    """
    print(
        f"Hello, Amazon ECR! Let's list some images in the repository
 '{repository_name}':\n"
    )
    paginator = ecr_client.get_paginator("list_images")
    page_iterator = paginator.paginate(
        repositoryName=repository_name, PaginationConfig={"MaxItems": 10}
    )

    image_names: [str] = []
    for page in page_iterator:
        for schedule in page["imageIds"]:
            image_names.append(schedule["imageTag"])

    print(f"{len(image_names)} image(s) retrieved.")
    for schedule_name in image_names:
        print(f"\t{schedule_name}")


if __name__ == "__main__":
    parser = argparse.ArgumentParser(description="Run hello Amazon ECR.")
    parser.add_argument(
        "--repository-name",
        type=str,
        help="the name of an Amazon ECR repository in your account.",
        required=True,
    )
    args = parser.parse_args()

    hello_ecr(boto3.client("ecr"), args.repository_name)
```

- Per informazioni dettagliate sull'API, consulta listImages nella documentazione di riferimento dell'API AWS SDK per Python (Boto3).

Per un elenco completo delle guide per sviluppatori AWS SDK e degli esempi di codice, consultaUtilizzo di Amazon ECR con un SDK AWS. Questo argomento include anche informazioni su come iniziare e dettagli sulle versioni precedenti dell'SDK.

Scopri le nozioni di base di Amazon ECR con un SDK AWS

Gli esempi di codice seguenti mostrano come:

- Crea un repository Amazon ECR.
- Impostare le policy dei repository.
- Recupera il repository. URIs
- Ottenere i token di autorizzazione Amazon ECR.
- Impostare policy del ciclo di vita per i repository Amazon ECR.
- Eseguire il push di un'immagine Docker a un repository Amazon ECR.
- Verificare l'esistenza di un'immagine in un repository Amazon ECR.
- Elencare i repository Amazon ECR per l'account corrente e ottenere le relative informazioni dettagliate.
- Eliminare i repository Amazon ECR.

Java

SDK per Java 2.x

> **Note**
>
> C'è altro su. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

Esegue uno scenario interattivo che dimostra le funzionalità di Amazon ECR.

```
import software.amazon.awssdk.services.ecr.model.EcrException;
import
 software.amazon.awssdk.services.ecr.model.RepositoryPolicyNotFoundException;
```

```
import java.util.Scanner;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 *
 * This Java code example requires an IAM Role that has permissions to interact
 with the Amazon ECR service.
 *
 * To create an IAM role, see:
 *
 * https://docs.aws.amazon.com/IAM/latest/UserGuide/id_roles_create.html
 *
 * This Java scenario example requires a local docker image named echo-text.
 Without a local image,
 * this Java program will not successfully run. For more information including
 how to create the local
 * image, see:
 *
 * /scenarios/basics/ecr/README
 *
 */
public class ECRScenario {
    public static final String DASHES = new String(new char[80]).replace("\0",
 "-");
    public static void main(String[] args) {
        final String usage = """
            Usage: <iamRoleARN> <accountId>

            Where:
               iamRoleARN - The IAM role ARN that has the necessary permissions
 to access and manage the Amazon ECR repository.
               accountId - Your AWS account number.
            """;

        if (args.length != 2) {
            System.out.println(usage);
            return;
```

```
        }

        ECRActions ecrActions = new ECRActions();
        String iamRole = args[0];
        String accountId = args[1];
        String localImageName;

        Scanner scanner = new Scanner(System.in);
        System.out.println("""
             The Amazon Elastic Container Registry (ECR) is a fully-managed
Docker container registry
             service provided by AWS. It allows developers and organizations to
securely
             store, manage, and deploy Docker container images.
             ECR provides a simple and scalable way to manage container images
throughout their lifecycle,
             from building and testing to production deployment.\s

             The `EcrAsyncClient` interface in the AWS SDK for Java 2.x provides
a set of methods to
             programmatically interact with the Amazon ECR service. This allows
developers to
             automate the storage, retrieval, and management of container images
as part of their application
             deployment pipelines. With ECR, teams can focus on building and
deploying their
             applications without having to worry about the underlying
infrastructure required to
             host and manage a container registry.

           This scenario walks you through how to perform key operations for
this service.
           Let's get started...

           You have two choices:
           1 - Run the entire program.
           2 - Delete an existing Amazon ECR repository named echo-text (created
from a previous execution of
           this program that did not complete).
           """);

       while (true) {
           String input = scanner.nextLine();
           if (input.trim().equalsIgnoreCase("1")) {
```

```
                System.out.println("Continuing with the program...");
                System.out.println("");
                break;
            } else if (input.trim().equalsIgnoreCase("2")) {
                String repoName = "echo-text";
                ecrActions.deleteECRRepository(repoName);
                return;
            } else {
                // Handle invalid input.
                System.out.println("Invalid input. Please try again.");
            }
        }

        waitForInputToContinue(scanner);
        System.out.println(DASHES);

        System.out.println("""
           1. Create an ECR repository.

           The first task is to ensure we have a local Docker image named echo-
text.
           If this image exists, then an Amazon ECR repository is created.

           An ECR repository is a private Docker container repository provided
           by Amazon Web Services (AWS). It is a managed service that makes it
 easy
           to store, manage, and deploy Docker container images.\s
           """ );

        // Ensure that a local docker image named echo-text exists.
        boolean doesExist = ecrActions.isEchoTextImagePresent();
        String repoName;
        if (!doesExist){
            System.out.println("The local image named echo-text does not exist");
            return;
        } else {
            localImageName = "echo-text";
            repoName = "echo-text";
        }

        try {
            String repoArn = ecrActions.createECRRepository(repoName);
            System.out.println("The ARN of the ECR repository is " + repoArn);
```

```
        } catch (IllegalArgumentException e) {
            System.err.println("Invalid repository name: " + e.getMessage());
            return;
        } catch (RuntimeException e) {
            System.err.println("An error occurred while creating the ECR
repository: " + e.getMessage());
            e.printStackTrace();
            return;
        }
        waitForInputToContinue(scanner);

        System.out.println(DASHES);
        System.out.println("""
        2. Set an ECR repository policy.

        Setting an ECR repository policy using the `setRepositoryPolicy` function
is crucial for maintaining
        the security and integrity of your container images. The repository
policy allows you to
        define specific rules and restrictions for accessing and managing the
images stored within your ECR
        repository.
        """);
        waitForInputToContinue(scanner);
        try {
            ecrActions.setRepoPolicy(repoName, iamRole);

        } catch (RepositoryPolicyNotFoundException e) {
            System.err.println("Invalid repository name: " + e.getMessage());
            return;
        } catch (EcrException e) {
            System.err.println("An ECR exception occurred: " + e.getMessage());
            return;
        } catch (RuntimeException e) {
            System.err.println("An error occurred while creating the ECR
repository: " + e.getMessage());
            return;
        }
        waitForInputToContinue(scanner);

        System.out.println(DASHES);
        System.out.println("""
        3. Display ECR repository policy.
```

```
        Now we will retrieve the ECR policy to ensure it was successfully set.
        """);
        waitForInputToContinue(scanner);
        try {
            String policyText = ecrActions.getRepoPolicy(repoName);
            System.out.println("Policy Text:");
            System.out.println(policyText);

        } catch (EcrException e) {
            System.err.println("An ECR exception occurred: " + e.getMessage());
            return;
        } catch (RuntimeException e) {
            System.err.println("An error occurred while creating the ECR
repository: " + e.getMessage());
            return;
        }

        waitForInputToContinue(scanner);

        System.out.println(DASHES);
        System.out.println("""
        4. Retrieve an ECR authorization token.

        You need an authorization token to securely access and interact with the
Amazon ECR registry.
        The `getAuthorizationToken` method of the `EcrAsyncClient` is responsible
for securely accessing
        and interacting with an Amazon ECR repository. This operation is
responsible for obtaining a
        valid authorization token, which is required to authenticate your
requests to the ECR service.

        Without a valid authorization token, you would not be able to perform any
operations on the
        ECR repository, such as pushing, pulling, or managing your Docker images.

        """);
        waitForInputToContinue(scanner);
        try {
             ecrActions.getAuthToken();

        } catch (EcrException e) {
            System.err.println("An ECR exception occurred: " + e.getMessage());
            return;
```

```
        } catch (RuntimeException e) {
            System.err.println("An error occurred while retrieving the
authorization token: " + e.getMessage());
            return;
        }
        waitForInputToContinue(scanner);

        System.out.println(DASHES);
        System.out.println("""
        5. Get the ECR Repository URI.

        The URI  of an Amazon ECR repository is important. When you want to
deploy a container image to
        a container orchestration platform like Amazon Elastic Kubernetes Service
(EKS)
        or Amazon Elastic Container Service (ECS), you need to specify the full
image URI,
        which includes the ECR repository URI. This allows the container runtime
to pull the
        correct container image from the ECR repository.
       """);
        waitForInputToContinue(scanner);

        try {
            ecrActions.getRepositoryURI(repoName);

        } catch (EcrException e) {
            System.err.println("An ECR exception occurred: " + e.getMessage());
            return;

        } catch (RuntimeException e) {
            System.err.println("An error occurred while retrieving the URI: " +
e.getMessage());
            return;
        }
        waitForInputToContinue(scanner);

        System.out.println(DASHES);
        System.out.println("""
            6. Set an ECR Lifecycle Policy.

            An ECR Lifecycle Policy is used to manage the lifecycle of Docker
images stored in your ECR repositories.
```

```
            These policies allow you to automatically remove old or unused Docker
images from your repositories,
            freeing up storage space and reducing costs.

            This example policy helps to maintain the size and efficiency of the
container registry
            by automatically removing older and potentially unused images,
ensuring that the
            storage is optimized and the registry remains up-to-date.
            """);
        waitForInputToContinue(scanner);
        try {
            ecrActions.setLifeCyclePolicy(repoName);

        } catch (RuntimeException e) {
            System.err.println("An error occurred while setting the lifecycle
policy: " + e.getMessage());
            e.printStackTrace();
            return;
        }
        waitForInputToContinue(scanner);

        System.out.println(DASHES);
        System.out.println("""
        7. Push a docker image to the Amazon ECR Repository.

        The `pushImageCmd()` method pushes a local Docker image to an Amazon ECR
repository.
        It sets up the Docker client by connecting to the local Docker host using
the default port.
        It then retrieves the authorization token for the ECR repository by
making a call to the AWS SDK.

        The method uses the authorization token to create an `AuthConfig` object,
which is used to authenticate
        the Docker client when pushing the image. Finally, the method tags the
Docker image with the specified
        repository name and image tag, and then pushes the image to the ECR
repository using the Docker client.
        If the push operation is successful, the method prints a message
indicating that the image was pushed to ECR.
        """);
        waitForInputToContinue(scanner);
```

```
        try {
            ecrActions.pushDockerImage(repoName, localImageName);

        } catch (RuntimeException e) {
            System.err.println("An error occurred while pushing a local Docker
 image to Amazon ECR: " + e.getMessage());
            e.printStackTrace();
            return;
        }
        waitForInputToContinue(scanner);

        System.out.println(DASHES);
        System.out.println("8. Verify if the image is in the ECR Repository.");
        waitForInputToContinue(scanner);
        try {
            ecrActions.verifyImage(repoName, localImageName);

        } catch (EcrException e) {
            System.err.println("An ECR exception occurred: " + e.getMessage());
            return;
        } catch (RuntimeException e) {
            System.err.println("An error occurred " + e.getMessage());
            e.printStackTrace();
            return;
        }
        waitForInputToContinue(scanner);

        System.out.println(DASHES);
        System.out.println("9. As an optional step, you can interact with the
 image in Amazon ECR by using the CLI.");
        System.out.println("Would you like to view instructions on how to use the
 CLI to run the image? (y/n)");
        String ans = scanner.nextLine().trim();
        if (ans.equalsIgnoreCase("y")) {
            String instructions = """
            1. Authenticate with ECR - Before you can pull the image from Amazon
 ECR, you need to authenticate with the registry. You can do this using the AWS
 CLI:

                aws ecr get-login-password --region us-east-1 | docker login --
username AWS --password-stdin %s.dkr.ecr.us-east-1.amazonaws.com

            2. Describe the image using this command:
```

```
               aws ecr describe-images --repository-name %s --image-ids imageTag=
%s

            3. Run the Docker container and view the output using this command:

               docker run --rm %s.dkr.ecr.us-east-1.amazonaws.com/%s:%s
            """;

            instructions = String.format(instructions, accountId, repoName,
 localImageName, accountId, repoName, localImageName);
            System.out.println(instructions);
        }
        waitForInputToContinue(scanner);

        System.out.println(DASHES);
        System.out.println("10. Delete the ECR Repository.");
        System.out.println(
        """
        If the repository isn't empty, you must either delete the contents of the
 repository
        or use the force option (used in this scenario) to delete the repository
 and have Amazon ECR delete all of its contents
        on your behalf.
        """);
        System.out.println("Would you like to delete the Amazon ECR Repository?
(y/n)");
        String delAns = scanner.nextLine().trim();
        if (delAns.equalsIgnoreCase("y")) {
            System.out.println("You selected to delete the AWS ECR resources.");

            try {
                ecrActions.deleteECRRepository(repoName);

            } catch (EcrException e) {
                System.err.println("An ECR exception occurred: " +
 e.getMessage());
                return;
            } catch (RuntimeException e) {
                System.err.println("An error occurred while deleting the Docker
 image: " + e.getMessage());
                e.printStackTrace();
                return;
            }
        }
```

```
        System.out.println(DASHES);
        System.out.println("This concludes the Amazon ECR SDK scenario");
        System.out.println(DASHES);
    }

   private static void waitForInputToContinue(Scanner scanner) {
       while (true) {
           System.out.println("");
           System.out.println("Enter 'c' followed by <ENTER> to continue:");
           String input = scanner.nextLine();

           if (input.trim().equalsIgnoreCase("c")) {
               System.out.println("Continuing with the program...");
               System.out.println("");
               break;
           } else {
               // Handle invalid input.
               System.out.println("Invalid input. Please try again.");
           }
       }
   }
}
```

Una classe wrapper per i metodi dell'SDK di Amazon ECR.

```
import com.github.dockerjava.api.DockerClient;
import com.github.dockerjava.api.exception.DockerClientException;
import com.github.dockerjava.api.model.AuthConfig;
import com.github.dockerjava.api.model.Image;
import com.github.dockerjava.core.DockerClientBuilder;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.core.client.config.ClientOverrideConfiguration;
import software.amazon.awssdk.http.async.SdkAsyncHttpClient;
import software.amazon.awssdk.http.nio.netty.NettyNioAsyncHttpClient;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.ecr.EcrAsyncClient;
import software.amazon.awssdk.services.ecr.model.AuthorizationData;
import software.amazon.awssdk.services.ecr.model.CreateRepositoryRequest;
import software.amazon.awssdk.services.ecr.model.CreateRepositoryResponse;
import software.amazon.awssdk.services.ecr.model.DeleteRepositoryRequest;
```

```
import software.amazon.awssdk.services.ecr.model.DeleteRepositoryResponse;
import software.amazon.awssdk.services.ecr.model.DescribeImagesRequest;
import software.amazon.awssdk.services.ecr.model.DescribeImagesResponse;
import software.amazon.awssdk.services.ecr.model.DescribeRepositoriesRequest;
import software.amazon.awssdk.services.ecr.model.DescribeRepositoriesResponse;
import software.amazon.awssdk.services.ecr.model.EcrException;
import software.amazon.awssdk.services.ecr.model.GetAuthorizationTokenResponse;
import software.amazon.awssdk.services.ecr.model.GetRepositoryPolicyRequest;
import software.amazon.awssdk.services.ecr.model.GetRepositoryPolicyResponse;
import software.amazon.awssdk.services.ecr.model.ImageIdentifier;
import software.amazon.awssdk.services.ecr.model.Repository;
import
 software.amazon.awssdk.services.ecr.model.RepositoryPolicyNotFoundException;
import software.amazon.awssdk.services.ecr.model.SetRepositoryPolicyRequest;
import software.amazon.awssdk.services.ecr.model.SetRepositoryPolicyResponse;
import
 software.amazon.awssdk.services.ecr.model.StartLifecyclePolicyPreviewRequest;
import
 software.amazon.awssdk.services.ecr.model.StartLifecyclePolicyPreviewResponse;
import com.github.dockerjava.api.command.DockerCmdExecFactory;
import com.github.dockerjava.netty.NettyDockerCmdExecFactory;
import java.time.Duration;
import java.util.Base64;
import java.util.List;
import java.util.concurrent.CompletableFuture;
import java.util.concurrent.CompletionException;

public class ECRActions {
    private static EcrAsyncClient ecrClient;

    private static DockerClient dockerClient;

    private static Logger logger = LoggerFactory.getLogger(ECRActions.class);

    /**
     * Creates an Amazon Elastic Container Registry (Amazon ECR) repository.
     *
     * @param repoName the name of the repository to create.
     * @return the Amazon Resource Name (ARN) of the created repository, or an
 empty string if the operation failed.
     * @throws IllegalArgumentException     If repository name is invalid.
     * @throws RuntimeException             if an error occurs while creating the
 repository.
     */
```

```
   public String createECRRepository(String repoName) {
       if (repoName == null || repoName.isEmpty()) {
           throw new IllegalArgumentException("Repository name cannot be null or
empty");
       }

       CreateRepositoryRequest request = CreateRepositoryRequest.builder()
           .repositoryName(repoName)
           .build();

       CompletableFuture<CreateRepositoryResponse> response =
getAsyncClient().createRepository(request);
       try {
           CreateRepositoryResponse result = response.join();
           if (result != null) {
               System.out.println("The " + repoName + " repository was created
successfully.");
               return result.repository().repositoryArn();
           } else {
               throw new RuntimeException("Unexpected response type");
           }
       } catch (CompletionException e) {
           Throwable cause = e.getCause();
           if (cause instanceof EcrException ex) {
               if
("RepositoryAlreadyExistsException".equals(ex.awsErrorDetails().errorCode())) {
                   System.out.println("The Amazon ECR repository already exists,
moving on...");
                   DescribeRepositoriesRequest describeRequest =
DescribeRepositoriesRequest.builder()
                       .repositoryNames(repoName)
                       .build();
                   DescribeRepositoriesResponse describeResponse =
getAsyncClient().describeRepositories(describeRequest).join();
                   return
describeResponse.repositories().get(0).repositoryArn();
               } else {
                   throw new RuntimeException(ex);
               }
           } else {
               throw new RuntimeException(e);
           }
       }
   }
```

```
   /**
    * Deletes an ECR (Elastic Container Registry) repository.
    *
    * @param repoName the name of the repository to delete.
    * @throws IllegalArgumentException if the repository name is null or empty.
    * @throws EcrException if there is an error deleting the repository.
    * @throws RuntimeException if an unexpected error occurs during the deletion
process.
    */
   public void deleteECRRepository(String repoName) {
       if (repoName == null || repoName.isEmpty()) {
           throw new IllegalArgumentException("Repository name cannot be null or
empty");
       }

       DeleteRepositoryRequest repositoryRequest =
DeleteRepositoryRequest.builder()
           .force(true)
           .repositoryName(repoName)
           .build();

       CompletableFuture<DeleteRepositoryResponse> response =
getAsyncClient().deleteRepository(repositoryRequest);
       response.whenComplete((deleteRepositoryResponse, ex) -> {
           if (deleteRepositoryResponse != null) {
               System.out.println("You have successfully deleted the " +
repoName + " repository");
           } else {
               Throwable cause = ex.getCause();
               if (cause instanceof EcrException) {
                   throw (EcrException) cause;
               } else {
                   throw new RuntimeException("Unexpected error: " +
cause.getMessage(), cause);
               }
           }
       });

       // Wait for the CompletableFuture to complete
       response.join();
   }
```

```
   private static DockerClient getDockerClient() {
       String osName = System.getProperty("os.name");
       if (osName.startsWith("Windows")) {
           // Make sure Docker Desktop is running.
           String dockerHost = "tcp://localhost:2375"; // Use the Docker Desktop
default port.
           DockerCmdExecFactory dockerCmdExecFactory = new
NettyDockerCmdExecFactory().withReadTimeout(20000).withConnectTimeout(20000);
           dockerClient =
DockerClientBuilder.getInstance(dockerHost).withDockerCmdExecFactory(dockerCmdExecFactor
       } else {
           dockerClient = DockerClientBuilder.getInstance().build();
       }
       return dockerClient;
   }

   /**
    * Retrieves an asynchronous Amazon Elastic Container Registry (ECR) client.
    *
    * @return the configured ECR asynchronous client.
    */
   private static EcrAsyncClient getAsyncClient() {

       /*
        The `NettyNioAsyncHttpClient` class is part of the AWS SDK for Java,
version 2,
        and it is designed to provide a high-performance, asynchronous HTTP
client for interacting with AWS services.
        It uses the Netty framework to handle the underlying network
communication and the Java NIO API to
        provide a non-blocking, event-driven approach to HTTP requests and
responses.
        */
       SdkAsyncHttpClient httpClient = NettyNioAsyncHttpClient.builder()
           .maxConcurrency(50)  // Adjust as needed.
           .connectionTimeout(Duration.ofSeconds(60))  // Set the connection
timeout.
           .readTimeout(Duration.ofSeconds(60))  // Set the read timeout.
           .writeTimeout(Duration.ofSeconds(60))  // Set the write timeout.
           .build();

       ClientOverrideConfiguration overrideConfig =
ClientOverrideConfiguration.builder()
```

```
            .apiCallTimeout(Duration.ofMinutes(2))  // Set the overall API call
 timeout.
            .apiCallAttemptTimeout(Duration.ofSeconds(90))  // Set the individual
 call attempt timeout.
            .build();

        if (ecrClient == null) {
            ecrClient = EcrAsyncClient.builder()
                .region(Region.US_EAST_1)
                .httpClient(httpClient)
                .overrideConfiguration(overrideConfig)
                .build();
        }
        return ecrClient;
    }

    /**
     * Sets the lifecycle policy for the specified repository.
     *
     * @param repoName the name of the repository for which to set the lifecycle
 policy.
     */
    public void setLifeCyclePolicy(String repoName) {
        /*
           This policy helps to maintain the size and efficiency of the container
 registry
           by automatically removing older and potentially unused images,
           ensuring that the storage is optimized and the registry remains up-to-
date.
         */
        String polText = """
             {
             "rules": [
                 {
                     "rulePriority": 1,
                     "description": "Expire images older than 14 days",
                     "selection": {
                         "tagStatus": "any",
                         "countType": "sinceImagePushed",
                         "countUnit": "days",
                         "countNumber": 14
                     },
                     "action": {
                         "type": "expire"
```

```
                    }
                }
            ]
            }
            """;

        StartLifecyclePolicyPreviewRequest lifecyclePolicyPreviewRequest =
StartLifecyclePolicyPreviewRequest.builder()
            .lifecyclePolicyText(polText)
            .repositoryName(repoName)
            .build();

        CompletableFuture<StartLifecyclePolicyPreviewResponse> response =
getAsyncClient().startLifecyclePolicyPreview(lifecyclePolicyPreviewRequest);
        response.whenComplete((lifecyclePolicyPreviewResponse, ex) -> {
            if (lifecyclePolicyPreviewResponse != null) {
                System.out.println("Lifecycle policy preview started
successfully.");
            } else {
                if (ex.getCause() instanceof EcrException) {
                    throw (EcrException) ex.getCause();
                } else {
                    String errorMessage = "Unexpected error occurred: " +
ex.getMessage();
                    throw new RuntimeException(errorMessage, ex);
                }
            }
        });
        // Wait for the CompletableFuture to complete.
        response.join();
    }

    /**
     * Verifies the existence of an image in an Amazon Elastic Container Registry
(Amazon ECR) repository asynchronously.
     *
     * @param repositoryName The name of the Amazon ECR repository.
     * @param imageTag       The tag of the image to verify.
     * @throws EcrException             if there is an error retrieving the image
information from Amazon ECR.
     * @throws CompletionException      if the asynchronous operation completes
exceptionally.
     */
    public void verifyImage(String repositoryName, String imageTag) {
```

```
        DescribeImagesRequest request = DescribeImagesRequest.builder()
            .repositoryName(repositoryName)
            .imageIds(ImageIdentifier.builder().imageTag(imageTag).build())
            .build();

        CompletableFuture<DescribeImagesResponse> response =
 getAsyncClient().describeImages(request);
        response.whenComplete((describeImagesResponse, ex) -> {
            if (ex != null) {
                if (ex instanceof CompletionException) {
                    Throwable cause = ex.getCause();
                    if (cause instanceof EcrException) {
                        throw (EcrException) cause;
                    } else {
                        throw new RuntimeException("Unexpected error: " +
 cause.getMessage(), cause);
                    }
                } else {
                    throw new RuntimeException("Unexpected error: " +
 ex.getCause());
                }
            } else if (describeImagesResponse != null &&
!describeImagesResponse.imageDetails().isEmpty()) {
                System.out.println("Image is present in the repository.");
            } else {
                System.out.println("Image is not present in the repository.");
            }
        });

        // Wait for the CompletableFuture to complete.
        response.join();
    }

    /**
     * Retrieves the repository URI for the specified repository name.
     *
     * @param repoName the name of the repository to retrieve the URI for.
     * @return the repository URI for the specified repository name.
     * @throws EcrException        if there is an error retrieving the repository
 information.
     * @throws CompletionException if the asynchronous operation completes
 exceptionally.
     */
    public void getRepositoryURI(String repoName) {
```

```
        DescribeRepositoriesRequest request =
DescribeRepositoriesRequest.builder()
            .repositoryNames(repoName)
            .build();

        CompletableFuture<DescribeRepositoriesResponse> response =
getAsyncClient().describeRepositories(request);
        response.whenComplete((describeRepositoriesResponse, ex) -> {
            if (ex != null) {
                Throwable cause = ex.getCause();
                if (cause instanceof InterruptedException) {
                    Thread.currentThread().interrupt();
                    String errorMessage = "Thread interrupted while waiting for
asynchronous operation: " + cause.getMessage();
                    throw new RuntimeException(errorMessage, cause);
                } else if (cause instanceof EcrException) {
                    throw (EcrException) cause;
                } else {
                    String errorMessage = "Unexpected error: " +
cause.getMessage();
                    throw new RuntimeException(errorMessage, cause);
                }
            } else {
                if (describeRepositoriesResponse != null) {
                    if (!describeRepositoriesResponse.repositories().isEmpty()) {
                        String repositoryUri =
describeRepositoriesResponse.repositories().get(0).repositoryUri();
                        System.out.println("Repository URI found: " +
repositoryUri);
                    } else {
                        System.out.println("No repositories found for the given
name.");
                    }
                } else {
                    System.err.println("No response received from
describeRepositories.");
                }
            }
        });
        response.join();
    }

    /**
```

```
    * Retrieves the authorization token for Amazon Elastic Container Registry
(ECR).
    * This method makes an asynchronous call to the ECR client to retrieve the
authorization token.
    * If the operation is successful, the method prints the token to the
console.
    * If an exception occurs, the method handles the exception and prints the
error message.
    *
    * @throws EcrException     if there is an error retrieving the authorization
token from ECR.
    * @throws RuntimeException if there is an unexpected error during the
operation.
    */
   public void getAuthToken() {
       CompletableFuture<GetAuthorizationTokenResponse> response =
getAsyncClient().getAuthorizationToken();
       response.whenComplete((authorizationTokenResponse, ex) -> {
           if (authorizationTokenResponse != null) {
               AuthorizationData authorizationData =
authorizationTokenResponse.authorizationData().get(0);
               String token = authorizationData.authorizationToken();
               if (!token.isEmpty()) {
                   System.out.println("The token was successfully retrieved.");
               }
           } else {
               if (ex.getCause() instanceof EcrException) {
                   throw (EcrException) ex.getCause();
               } else {
                   String errorMessage = "Unexpected error occurred: " +
ex.getMessage();
                   throw new RuntimeException(errorMessage, ex); // Rethrow the
exception
               }
           }
       });
       response.join();
   }

   /**
    * Gets the repository policy for the specified repository.
    *
    * @param repoName the name of the repository.
```

```
    * @throws EcrException if an AWS error occurs while getting the repository
policy.
    */
   public String getRepoPolicy(String repoName) {
       if (repoName == null || repoName.isEmpty()) {
           throw new IllegalArgumentException("Repository name cannot be null or
empty");
       }

       GetRepositoryPolicyRequest getRepositoryPolicyRequest =
GetRepositoryPolicyRequest.builder()
           .repositoryName(repoName)
           .build();

       CompletableFuture<GetRepositoryPolicyResponse> response =
getAsyncClient().getRepositoryPolicy(getRepositoryPolicyRequest);
       response.whenComplete((resp, ex) -> {
           if (resp != null) {
               System.out.println("Repository policy retrieved successfully.");
           } else {
               if (ex.getCause() instanceof EcrException) {
                   throw (EcrException) ex.getCause();
               } else {
                   String errorMessage = "Unexpected error occurred: " +
ex.getMessage();
                   throw new RuntimeException(errorMessage, ex);
               }
           }
       });

       GetRepositoryPolicyResponse result = response.join();
       return result != null ? result.policyText() : null;
   }

   /**
    * Sets the repository policy for the specified ECR repository.
    *
    * @param repoName the name of the ECR repository.
    * @param iamRole  the IAM role to be granted access to the repository.
    * @throws RepositoryPolicyNotFoundException if the repository policy does
not exist.
    * @throws EcrException                      if there is an unexpected error
setting the repository policy.
    */
```

```
   public void setRepoPolicy(String repoName, String iamRole) {
       /*
         This example policy document grants the specified AWS principal the
permission to perform the
         `ecr:BatchGetImage` action. This policy is designed to allow the
specified principal
         to retrieve Docker images from the ECR repository.
        */
       String policyDocumentTemplate = """
            {
             "Version":"2012-10-17",
             "Statement" : [ {
               "Sid" : "new statement",
               "Effect" : "Allow",
               "Principal" : {
                 "AWS" : "%s"
               },
               "Action" : "ecr:BatchGetImage"
             } ]
           }
            """;

       String policyDocument = String.format(policyDocumentTemplate, iamRole);
       SetRepositoryPolicyRequest setRepositoryPolicyRequest =
SetRepositoryPolicyRequest.builder()
           .repositoryName(repoName)
           .policyText(policyDocument)
           .build();

       CompletableFuture<SetRepositoryPolicyResponse> response =
getAsyncClient().setRepositoryPolicy(setRepositoryPolicyRequest);
       response.whenComplete((resp, ex) -> {
           if (resp != null) {
               System.out.println("Repository policy set successfully.");
           } else {
               Throwable cause = ex.getCause();
               if (cause instanceof RepositoryPolicyNotFoundException) {
                   throw (RepositoryPolicyNotFoundException) cause;
               } else if (cause instanceof EcrException) {
                   throw (EcrException) cause;
               } else {
                   String errorMessage = "Unexpected error: " +
cause.getMessage();
                   throw new RuntimeException(errorMessage, cause);
```

```
                }
            }
        });
        response.join();
    }

    /**
     * Pushes a Docker image to an Amazon Elastic Container Registry (ECR)
repository.
     *
     * @param repoName  the name of the ECR repository to push the image to.
     * @param imageName the name of the Docker image.
     */
    public void pushDockerImage(String repoName, String imageName) {
        System.out.println("Pushing " + imageName + " to Amazon ECR will take a
few seconds.");
        CompletableFuture<AuthConfig> authResponseFuture =
getAsyncClient().getAuthorizationToken()
            .thenApply(response -> {
                String token =
response.authorizationData().get(0).authorizationToken();
                String decodedToken = new
String(Base64.getDecoder().decode(token));
                String password = decodedToken.substring(4);

                DescribeRepositoriesResponse descrRepoResponse =
getAsyncClient().describeRepositories(b -> b.repositoryNames(repoName)).join();
                Repository repoData =
descrRepoResponse.repositories().stream().filter(r ->
r.repositoryName().equals(repoName)).findFirst().orElse(null);
                assert repoData != null;
                String registryURL = repoData.repositoryUri().split("/")[0];

                AuthConfig authConfig = new AuthConfig()
                    .withUsername("AWS")
                    .withPassword(password)
                    .withRegistryAddress(registryURL);
                return authConfig;
            })
            .thenCompose(authConfig -> {
                DescribeRepositoriesResponse descrRepoResponse =
getAsyncClient().describeRepositories(b -> b.repositoryNames(repoName)).join();
```

```
                Repository repoData =
 descrRepoResponse.repositories().stream().filter(r ->
 r.repositoryName().equals(repoName)).findFirst().orElse(null);
                getDockerClient().tagImageCmd(imageName + ":latest",
 repoData.repositoryUri() + ":latest", imageName).exec();
                try {

 getDockerClient().pushImageCmd(repoData.repositoryUri()).withTag("echo-
text").withAuthConfig(authConfig).start().awaitCompletion();
                    System.out.println("The " + imageName + " was pushed to
 ECR");

                } catch (InterruptedException e) {
                    throw (RuntimeException) e.getCause();
                }
                return CompletableFuture.completedFuture(authConfig);
            });

        authResponseFuture.join();
    }

    // Make sure local image echo-text exists.
    public boolean isEchoTextImagePresent() {
        try {
            List<Image> images = getDockerClient().listImagesCmd().exec();
            boolean helloWorldFound = false;
            for (Image image : images) {
                String[] repoTags = image.getRepoTags();
                if (repoTags != null) {
                    for (String tag : repoTags) {
                        if (tag.startsWith("echo-text")) {
                            System.out.println(tag);
                            helloWorldFound = true;
                        }
                    }
                }
            }
            if (helloWorldFound) {
                System.out.println("The local image named echo-text exists.");
                return true;
            } else {
                System.out.println("The local image named echo-text does not
 exist.");
                return false;
```

```
            }
        } catch (DockerClientException ex) {
            logger.error("ERROR: " + ex.getMessage());
            return false;
        }
    }
}
```

- Per informazioni dettagliate sull'API, consulta i seguenti argomenti nella documentazione di riferimento dell'API AWS SDK for Java 2.x .
  - CreateRepository
  - DeleteRepository
  - DescribeImages
  - DescribeRepositories
  - GetAuthorizationToken
  - GetRepositoryPolicy
  - SetRepositoryPolicy
  - StartLifecyclePolicyPreview

Kotlin

SDK per Kotlin

> **Note**
>
> C'è dell'altro GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

Esegue uno scenario interattivo che dimostra le funzionalità di Amazon ECR.

```
import java.util.Scanner

/**
 * Before running this Kotlin code example, set up your development environment,
 including your credentials.
 *
```

```
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-kotlin/latest/developer-guide/setup.html
 *
 * This code example requires an IAM Role that has permissions to interact with
 the Amazon ECR service.
 *
 * To create an IAM role, see:
 *
 * https://docs.aws.amazon.com/IAM/latest/UserGuide/id_roles_create.html
 *
 * This code example requires a local docker image named echo-text. Without a
 local image,
 * this program will not successfully run. For more information including how to
 create the local
 * image, see:
 *
 * /scenarios/basics/ecr/README
 *
 */

val DASHES = String(CharArray(80)).replace("\u0000", "-")

suspend fun main(args: Array<String>) {
    val usage =
        """
        Usage: <iamRoleARN> <accountId>

        Where:
           iamRoleARN - The IAM role ARN that has the necessary permissions to
 access and manage the Amazon ECR repository.
           accountId - Your AWS account number.

        """.trimIndent()

    if (args.size != 2) {
        println(usage)
        return
    }

    var iamRole = args[0]
    var localImageName: String
    var accountId = args[1]
    val ecrActions = ECRActions()
```

```
    val scanner = Scanner(System.`in`)

    println(
        """
        The Amazon Elastic Container Registry (ECR) is a fully-managed Docker
container registry
        service provided by AWS. It allows developers and organizations to
securely
        store, manage, and deploy Docker container images.
        ECR provides a simple and scalable way to manage container images
throughout their lifecycle,
        from building and testing to production deployment.

        The `EcrClient` service client that is part of the AWS SDK for Kotlin
provides a set of methods to
        programmatically interact with the Amazon ECR service. This allows
developers to
        automate the storage, retrieval, and management of container images as
part of their application
        deployment pipelines. With ECR, teams can focus on building and deploying
their
        applications without having to worry about the underlying infrastructure
required to
        host and manage a container registry.

        This scenario walks you through how to perform key operations for this
service.
        Let's get started...

         You have two choices:
            1 - Run the entire program.
            2 - Delete an existing Amazon ECR repository named echo-text (created
from a previous execution of
            this program that did not complete).

        """.trimIndent(),
    )

    while (true) {
        val input = scanner.nextLine()
        if (input.trim { it <= ' ' }.equals("1", ignoreCase = true)) {
            println("Continuing with the program...")
            println("")
            break
```

```
        } else if (input.trim { it <= ' ' }.equals("2", ignoreCase = true)) {
            val repoName = "echo-text"
            ecrActions.deleteECRRepository(repoName)
            return
        } else {
            // Handle invalid input.
            println("Invalid input. Please try again.")
        }
    }

    waitForInputToContinue(scanner)
    println(DASHES)
    println(
        """
        1. Create an ECR repository.

        The first task is to ensure we have a local Docker image named echo-
text.
        If this image exists, then an Amazon ECR repository is created.

        An ECR repository is a private Docker container repository provided
        by Amazon Web Services (AWS). It is a managed service that makes it easy
        to store, manage, and deploy Docker container images.

        """.trimIndent(),
    )

    // Ensure that a local docker image named echo-text exists.
    val doesExist = ecrActions.listLocalImages()
    val repoName: String
    if (!doesExist) {
        println("The local image named echo-text does not exist")
        return
    } else {
        localImageName = "echo-text"
        repoName = "echo-text"
    }

    val repoArn = ecrActions.createECRRepository(repoName).toString()
    println("The ARN of the ECR repository is $repoArn")
    waitForInputToContinue(scanner)

    println(DASHES)
    println(
```

```
        """
        2. Set an ECR repository policy.

        Setting an ECR repository policy using the `setRepositoryPolicy` function
is crucial for maintaining
        the security and integrity of your container images. The repository
policy allows you to
        define specific rules and restrictions for accessing and managing the
images stored within your ECR
        repository.

        """.trimIndent(),
    )
    waitForInputToContinue(scanner)
    ecrActions.setRepoPolicy(repoName, iamRole)
    waitForInputToContinue(scanner)

    println(DASHES)
    println(
        """
        3. Display ECR repository policy.

        Now we will retrieve the ECR policy to ensure it was successfully set.
        """.trimIndent(),
    )
    waitForInputToContinue(scanner)
    val policyText = ecrActions.getRepoPolicy(repoName)
    println("Policy Text:")
    println(policyText)
    waitForInputToContinue(scanner)

    println(DASHES)
    println(
        """
        4. Retrieve an ECR authorization token.

        You need an authorization token to securely access and interact with the
Amazon ECR registry.
        The `getAuthorizationToken` method of the `EcrAsyncClient` is responsible
for securely accessing
        and interacting with an Amazon ECR repository. This operation is
responsible for obtaining a
        valid authorization token, which is required to authenticate your
requests to the ECR service.
```

```
       Without a valid authorization token, you would not be able to perform any
operations on the
       ECR repository, such as pushing, pulling, or managing your Docker images.


       """.trimIndent(),
   )
   waitForInputToContinue(scanner)
   ecrActions.getAuthToken()
   waitForInputToContinue(scanner)

   println(DASHES)
   println(
       """
       5. Get the ECR Repository URI.

       The URI  of an Amazon ECR repository is important. When you want to
deploy a container image to
       a container orchestration platform like Amazon Elastic Kubernetes Service
(EKS)
       or Amazon Elastic Container Service (ECS), you need to specify the full
image URI,
       which includes the ECR repository URI. This allows the container runtime
to pull the
       correct container image from the ECR repository.

       """.trimIndent(),
   )
   waitForInputToContinue(scanner)
   val repositoryURI: String? = ecrActions.getRepositoryURI(repoName)
   println("The repository URI is $repositoryURI")
   waitForInputToContinue(scanner)

   println(DASHES)
   println(
       """
       6. Set an ECR Lifecycle Policy.

       An ECR Lifecycle Policy is used to manage the lifecycle of Docker images
stored in your ECR repositories.
       These policies allow you to automatically remove old or unused Docker
images from your repositories,
       freeing up storage space and reducing costs.
```

```
        """.trimIndent(),
    )
    waitForInputToContinue(scanner)
    val pol = ecrActions.setLifeCyclePolicy(repoName)
    println(pol)
    waitForInputToContinue(scanner)

    println(DASHES)
    println(
        """
        7. Push a docker image to the Amazon ECR Repository.

        The `pushImageCmd()` method pushes a local Docker image to an Amazon ECR
repository.
        It sets up the Docker client by connecting to the local Docker host using
the default port.
        It then retrieves the authorization token for the ECR repository by
making a call to the AWS SDK.

        The method uses the authorization token to create an `AuthConfig` object,
which is used to authenticate
        the Docker client when pushing the image. Finally, the method tags the
Docker image with the specified
        repository name and image tag, and then pushes the image to the ECR
repository using the Docker client.
        If the push operation is successful, the method prints a message
indicating that the image was pushed to ECR.

        """.trimIndent(),
    )

    waitForInputToContinue(scanner)
    ecrActions.pushDockerImage(repoName, localImageName)
    waitForInputToContinue(scanner)

    println(DASHES)
    println("8. Verify if the image is in the ECR Repository.")
    waitForInputToContinue(scanner)
    ecrActions.verifyImage(repoName, localImageName)
    waitForInputToContinue(scanner)

    println(DASHES)
```

```
    println("9. As an optional step, you can interact with the image in Amazon
 ECR by using the CLI.")
    println("Would you like to view instructions on how to use the CLI to run the
 image? (y/n)")
    val ans = scanner.nextLine().trim()
    if (ans.equals("y", true)) {
        val instructions = """
        1. Authenticate with ECR - Before you can pull the image from Amazon ECR,
 you need to authenticate with the registry. You can do this using the AWS CLI:

            aws ecr get-login-password --region us-east-1 | docker login --
username AWS --password-stdin $accountId.dkr.ecr.us-east-1.amazonaws.com

        2. Describe the image using this command:

           aws ecr describe-images --repository-name $repoName --image-ids
 imageTag=$localImageName

        3. Run the Docker container and view the output using this command:

           docker run --rm $accountId.dkr.ecr.us-east-1.amazonaws.com/$repoName:
$localImageName
        """
        println(instructions)
    }
    waitForInputToContinue(scanner)

    println(DASHES)
    println("10. Delete the ECR Repository.")
    println(
        """
        If the repository isn't empty, you must either delete the contents of the
 repository
        or use the force option (used in this scenario) to delete the repository
 and have Amazon ECR delete all of its contents
        on your behalf.

        """.trimIndent(),
    )
    println("Would you like to delete the Amazon ECR Repository? (y/n)")
    val delAns = scanner.nextLine().trim { it <= ' ' }
    if (delAns.equals("y", ignoreCase = true)) {
        println("You selected to delete the AWS ECR resources.")
        waitForInputToContinue(scanner)
```

```
        ecrActions.deleteECRRepository(repoName)
    }

    println(DASHES)
    println("This concludes the Amazon ECR SDK scenario")
    println(DASHES)
}

private fun waitForInputToContinue(scanner: Scanner) {
    while (true) {
        println("")
        println("Enter 'c' followed by <ENTER> to continue:")
        val input = scanner.nextLine()
        if (input.trim { it <= ' ' }.equals("c", ignoreCase = true)) {
            println("Continuing with the program...")
            println("")
            break
        } else {
            // Handle invalid input.
            println("Invalid input. Please try again.")
        }
    }
}
```

Una classe wrapper per i metodi dell'SDK di Amazon ECR.

```
import aws.sdk.kotlin.services.ecr.EcrClient
import aws.sdk.kotlin.services.ecr.model.CreateRepositoryRequest
import aws.sdk.kotlin.services.ecr.model.DeleteRepositoryRequest
import aws.sdk.kotlin.services.ecr.model.DescribeImagesRequest
import aws.sdk.kotlin.services.ecr.model.DescribeRepositoriesRequest
import aws.sdk.kotlin.services.ecr.model.EcrException
import aws.sdk.kotlin.services.ecr.model.GetRepositoryPolicyRequest
import aws.sdk.kotlin.services.ecr.model.ImageIdentifier
import aws.sdk.kotlin.services.ecr.model.RepositoryAlreadyExistsException
import aws.sdk.kotlin.services.ecr.model.SetRepositoryPolicyRequest
import aws.sdk.kotlin.services.ecr.model.StartLifecyclePolicyPreviewRequest
import com.github.dockerjava.api.DockerClient
import com.github.dockerjava.api.command.DockerCmdExecFactory
import com.github.dockerjava.api.model.AuthConfig
import com.github.dockerjava.core.DockerClientBuilder
import com.github.dockerjava.netty.NettyDockerCmdExecFactory
```

```
import java.io.IOException
import java.util.Base64

class ECRActions {
    private var dockerClient: DockerClient? = null

    private fun getDockerClient(): DockerClient? {
        val osName = System.getProperty("os.name")
        if (osName.startsWith("Windows")) {
            // Make sure Docker Desktop is running.
            val dockerHost = "tcp://localhost:2375" // Use the Docker Desktop
 default port.
            val dockerCmdExecFactory: DockerCmdExecFactory =

 NettyDockerCmdExecFactory().withReadTimeout(20000).withConnectTimeout(20000)
            dockerClient =
 DockerClientBuilder.getInstance(dockerHost).withDockerCmdExecFactory(dockerCmdExecFactor
        } else {
            dockerClient = DockerClientBuilder.getInstance().build()
        }
        return dockerClient
    }


    /**
     * Sets the lifecycle policy for the specified repository.
     *
     * @param repoName the name of the repository for which to set the lifecycle
 policy.
     */
    suspend fun setLifeCyclePolicy(repoName: String): String? {
        val polText =
            """
             {
             "rules": [
                 {
                     "rulePriority": 1,
                     "description": "Expire images older than 14 days",
                     "selection": {
                         "tagStatus": "any",
                         "countType": "sinceImagePushed",
                         "countUnit": "days",
                         "countNumber": 14
                     },
```

```
                    "action": {
                        "type": "expire"
                    }
                }
            ]
            }

            """.trimIndent()
        val lifecyclePolicyPreviewRequest =
            StartLifecyclePolicyPreviewRequest {
                lifecyclePolicyText = polText
                repositoryName = repoName
            }

        // Execute the request asynchronously.
        EcrClient.fromEnvironment { region = "us-east-1" }.use { ecrClient ->
            val response =
ecrClient.startLifecyclePolicyPreview(lifecyclePolicyPreviewRequest)
            return response.lifecyclePolicyText
        }
    }


    /**
     * Retrieves the repository URI for the specified repository name.
     *
     * @param repoName the name of the repository to retrieve the URI for.
     * @return the repository URI for the specified repository name.
     */
    suspend fun getRepositoryURI(repoName: String?): String? {
        require(!(repoName == null || repoName.isEmpty())) { "Repository name
cannot be null or empty" }
        val request =
            DescribeRepositoriesRequest {
                repositoryNames = listOf(repoName)
            }

        EcrClient.fromEnvironment { region = "us-east-1" }.use { ecrClient ->
            val describeRepositoriesResponse =
ecrClient.describeRepositories(request)
            if (!describeRepositoriesResponse.repositories?.isEmpty()!!) {
                return
describeRepositoriesResponse?.repositories?.get(0)?.repositoryUri
            } else {
```

```
                println("No repositories found for the given name.")
                return ""
            }
        }
    }


    /**
     * Retrieves the authorization token for Amazon Elastic Container Registry
(ECR).
     *
     */
    suspend fun getAuthToken() {
        EcrClient.fromEnvironment { region = "us-east-1" }.use { ecrClient ->
            // Retrieve the authorization token for ECR.
            val response = ecrClient.getAuthorizationToken()
            val authorizationData = response.authorizationData?.get(0)
            val token = authorizationData?.authorizationToken
            if (token != null) {
                println("The token was successfully retrieved.")
            }
        }
    }


    /**
     * Gets the repository policy for the specified repository.
     *
     * @param repoName the name of the repository.
     */
    suspend fun getRepoPolicy(repoName: String?): String? {
        require(!(repoName == null || repoName.isEmpty())) { "Repository name
cannot be null or empty" }

        // Create the request
        val getRepositoryPolicyRequest =
            GetRepositoryPolicyRequest {
                repositoryName = repoName
            }
        EcrClient.fromEnvironment { region = "us-east-1" }.use { ecrClient ->
            val response =
ecrClient.getRepositoryPolicy(getRepositoryPolicyRequest)
            val responseText = response.policyText
            return responseText
```

```
        }
    }


    /**
     * Sets the repository policy for the specified ECR repository.
     *
     * @param repoName the name of the ECR repository.
     * @param iamRole the IAM role to be granted access to the repository.
     */
    suspend fun setRepoPolicy(
        repoName: String?,
        iamRole: String?,
    ) {
        val policyDocumentTemplate =
            """
             {
              "Version":"2012-10-17",
              "Statement" : [ {
                "Sid" : "new statement",
                "Effect" : "Allow",
                "Principal" : {
                  "AWS" : "$iamRole"
                },
                "Action" : "ecr:BatchGetImage"
              } ]
            }

            """.trimIndent()
        val setRepositoryPolicyRequest =
            SetRepositoryPolicyRequest {
                repositoryName = repoName
                policyText = policyDocumentTemplate
            }

        EcrClient.fromEnvironment { region = "us-east-1" }.use { ecrClient ->
            val response =
ecrClient.setRepositoryPolicy(setRepositoryPolicyRequest)
            if (response != null) {
                println("Repository policy set successfully.")
            }
        }
    }
```

```
   /**
    * Creates an Amazon Elastic Container Registry (Amazon ECR) repository.
    *
    * @param repoName the name of the repository to create.
    * @return the Amazon Resource Name (ARN) of the created repository, or an
empty string if the operation failed.
    * @throws RepositoryAlreadyExistsException if the repository exists.
    * @throws EcrException         if an error occurs while creating the
repository.
    */
   suspend fun createECRRepository(repoName: String?): String? {
       val request =
           CreateRepositoryRequest {
               repositoryName = repoName
           }

       return try {
           EcrClient.fromEnvironment { region = "us-east-1" }.use { ecrClient ->
               val response = ecrClient.createRepository(request)
               response.repository?.repositoryArn
           }
       } catch (e: RepositoryAlreadyExistsException) {
           println("Repository already exists: $repoName")
           repoName?.let { getRepoARN(it) }
       } catch (e: EcrException) {
           println("An error occurred: ${e.message}")
           null
       }
   }

   suspend fun getRepoARN(repoName: String): String? {
       // Fetch the existing repository's ARN.
       val describeRequest =
           DescribeRepositoriesRequest {
               repositoryNames = listOf(repoName)
           }
       EcrClient.fromEnvironment { region = "us-east-1" }.use { ecrClient ->
           val describeResponse =
ecrClient.describeRepositories(describeRequest)
           return describeResponse.repositories?.get(0)?.repositoryArn
       }
   }
```

```
    fun listLocalImages(): Boolean = try {
        val images = getDockerClient()?.listImagesCmd()?.exec()
        images?.any { image ->
            image.repoTags?.any { tag -> tag.startsWith("echo-text") } ?: false
        } ?: false
    } catch (ex: Exception) {
        println("ERROR: ${ex.message}")
        false
    }


    /**
     * Pushes a Docker image to an Amazon Elastic Container Registry (ECR)
repository.
     *
     * @param repoName the name of the ECR repository to push the image to.
     * @param imageName the name of the Docker image.
     */
    suspend fun pushDockerImage(
        repoName: String,
        imageName: String,
    ) {
        println("Pushing $imageName to $repoName will take a few seconds")
        val authConfig = getAuthConfig(repoName)

        EcrClient.fromEnvironment { region = "us-east-1" }.use { ecrClient ->
            val desRequest =
                DescribeRepositoriesRequest {
                    repositoryNames = listOf(repoName)
                }

            val describeRepoResponse = ecrClient.describeRepositories(desRequest)
            val repoData =
                describeRepoResponse.repositories?.firstOrNull
{ it.repositoryName == repoName }
                    ?: throw RuntimeException("Repository not found: $repoName")

            val tagImageCmd = getDockerClient()?.tagImageCmd("$imageName",
"${repoData.repositoryUri}", imageName)
            if (tagImageCmd != null) {
                tagImageCmd.exec()
            }
            val pushImageCmd =
                repoData.repositoryUri?.let {
```

```
                    dockerClient?.pushImageCmd(it)
                        // ?.withTag("latest")
                        ?.withAuthConfig(authConfig)
                }

            try {
                if (pushImageCmd != null) {
                    pushImageCmd.start().awaitCompletion()
                }
                println("The $imageName was pushed to Amazon ECR")
            } catch (e: IOException) {
                throw RuntimeException(e)
            }
        }
    }


    /**
     * Verifies the existence of an image in an Amazon Elastic Container Registry
(Amazon ECR) repository asynchronously.
     *
     * @param repositoryName The name of the Amazon ECR repository.
     * @param imageTag       The tag of the image to verify.
     */
    suspend fun verifyImage(
        repoName: String?,
        imageTagVal: String?,
    ) {
        require(!(repoName == null || repoName.isEmpty())) { "Repository name
cannot be null or empty" }
        require(!(imageTagVal == null || imageTagVal.isEmpty())) { "Image tag
cannot be null or empty" }

        val imageId =
            ImageIdentifier {
                imageTag = imageTagVal
            }
        val request =
            DescribeImagesRequest {
                repositoryName = repoName
                imageIds = listOf(imageId)
            }

        EcrClient.fromEnvironment { region = "us-east-1" }.use { ecrClient ->
```

```
            val describeImagesResponse = ecrClient.describeImages(request)
            if (describeImagesResponse != null && !
describeImagesResponse.imageDetails?.isEmpty()!!) {
                println("Image is present in the repository.")
            } else {
                println("Image is not present in the repository.")
            }
        }
    }


    /**
     * Deletes an ECR (Elastic Container Registry) repository.
     *
     * @param repoName the name of the repository to delete.
     */
    suspend fun deleteECRRepository(repoName: String) {
        if (repoName.isNullOrEmpty()) {
            throw IllegalArgumentException("Repository name cannot be null or
 empty")
        }

        val repositoryRequest =
            DeleteRepositoryRequest {
                force = true
                repositoryName = repoName
            }

        EcrClient.fromEnvironment { region = "us-east-1" }.use { ecrClient ->
            ecrClient.deleteRepository(repositoryRequest)
            println("You have successfully deleted the $repoName repository")
        }
    }

    // Return an AuthConfig.
    private suspend fun getAuthConfig(repoName: String): AuthConfig {
        EcrClient.fromEnvironment { region = "us-east-1" }.use { ecrClient ->
            // Retrieve the authorization token for ECR.
            val response = ecrClient.getAuthorizationToken()
            val authorizationData = response.authorizationData?.get(0)
            val token = authorizationData?.authorizationToken
            val decodedToken = String(Base64.getDecoder().decode(token))
            val password = decodedToken.substring(4)
```

```
            val request =
                DescribeRepositoriesRequest {
                    repositoryNames = listOf(repoName)
                }

            val descrRepoResponse = ecrClient.describeRepositories(request)
            val repoData = descrRepoResponse.repositories?.firstOrNull
 { it.repositoryName == repoName }
            val registryURL: String =
 repoData?.repositoryUri?.split("/")?.get(0) ?: ""

            return AuthConfig()
                .withUsername("AWS")
                .withPassword(password)
                .withRegistryAddress(registryURL)
        }
    }
}
```

- Per informazioni dettagliate sull'API, consulta i seguenti argomenti nella documentazione di riferimento dell'API AWS SDK per Kotlin.
  - CreateRepository
  - DeleteRepository
  - DescribeImages
  - DescribeRepositories
  - GetAuthorizationToken
  - GetRepositoryPolicy
  - SetRepositoryPolicy
  - StartLifecyclePolicyPreview

Python

SDK per Python (Boto3)

> **Note**
>
> C'è dell'altro GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

Esegui uno scenario interattivo al prompt dei comandi.

```
class ECRGettingStarted:
    """
    A scenario that demonstrates how to use Boto3 to perform basic operations
 using
    Amazon ECR.
    """

    def __init__(
        self,
        ecr_wrapper: ECRWrapper,
        docker_client: docker.DockerClient,
    ):
        self.ecr_wrapper = ecr_wrapper
        self.docker_client = docker_client
        self.tag = "echo-text"
        self.repository_name = "ecr-basics"
        self.docker_image = None
        self.full_tag_name = None
        self.repository = None

    def run(self, role_arn: str) -> None:
        """
        Runs the scenario.
        """
        print(
            """
The Amazon Elastic Container Registry (ECR) is a fully-managed Docker container
 registry
service provided by AWS. It allows developers and organizations to securely
store, manage, and deploy Docker container images.
```

```
ECR provides a simple and scalable way to manage container images throughout
 their lifecycle,
from building and testing to production deployment.

The `ECRWrapper' class is a wrapper for the Boto3 'ecr' client. The 'ecr' client
 provides a set of methods to
programmatically interact with the Amazon ECR service. This allows developers to
automate the storage, retrieval, and management of container images as part of
 their application
deployment pipelines. With ECR, teams can focus on building and deploying their
applications without having to worry about the underlying infrastructure required
 to
host and manage a container registry.

This scenario walks you through how to perform key operations for this service.
Let's get started...
        """
        )
        press_enter_to_continue()
        print_dashes()
        print(
            f"""
* Create an ECR repository.

An ECR repository is a private Docker container repository provided
by Amazon Web Services (AWS). It is a managed service that makes it easy
to store, manage, and deploy Docker container images.
        """
        )
        print(f"Creating a repository named {self.repository_name}")
        self.repository =
 self.ecr_wrapper.create_repository(self.repository_name)
        print(f"The ARN of the ECR repository is
 {self.repository['repositoryArn']}")
        repository_uri = self.repository["repositoryUri"]
        press_enter_to_continue()
        print_dashes()

        print(
            f"""
* Build a Docker image.

Create a local Docker image if it does not already exist.
A Python Docker client is used to execute Docker commands.
```

```
You must have Docker installed and running.
            """
        )
        print(f"Building a docker image from 'docker_files/Dockerfile'")
        self.full_tag_name = f"{repository_uri}:{self.tag}"
        self.docker_image = self.docker_client.images.build(
            path="docker_files", tag=self.full_tag_name
        )[0]
        print(f"Docker image {self.full_tag_name} successfully built.")
        press_enter_to_continue()
        print_dashes()

        if role_arn is None:
            print(
                """
* Because an IAM role ARN was not provided, a role policy will not be set for
 this repository.
            """
            )
        else:
            print(
                """
* Set an ECR repository policy.

Setting an ECR repository policy using the `setRepositoryPolicy` function is
 crucial for maintaining
the security and integrity of your container images. The repository policy allows
 you to
define specific rules and restrictions for accessing and managing the images
 stored within your ECR
repository.
        """
            )

            self.grant_role_download_access(role_arn)
            print(f"Download access granted to the IAM role ARN {role_arn}")
            press_enter_to_continue()
            print_dashes()

            print(
                """
* Display ECR repository policy.

Now we will retrieve the ECR policy to ensure it was successfully set.
```

```
            """
            )

            policy_text =
 self.ecr_wrapper.get_repository_policy(self.repository_name)
            print("Policy Text:")
            print(f"{policy_text}")
            press_enter_to_continue()
            print_dashes()

        print(
            """
* Retrieve an ECR authorization token.

You need an authorization token to securely access and interact with the Amazon
 ECR registry.
The `get_authorization_token` method of the `ecr` client is responsible for
 securely accessing
and interacting with an Amazon ECR repository. This operation is responsible for
 obtaining a
valid authorization token, which is required to authenticate your requests to the
 ECR service.

Without a valid authorization token, you would not be able to perform any
 operations on the
ECR repository, such as pushing, pulling, or managing your Docker images.
        """
        )

        authorization_token = self.ecr_wrapper.get_authorization_token()
        print("Authorization token retrieved.")
        press_enter_to_continue()
        print_dashes()
        print(
            """
* Get the ECR Repository URI.

The URI  of an Amazon ECR repository is important. When you want to deploy a
 container image to
a container orchestration platform like Amazon Elastic Kubernetes Service (EKS)
or Amazon Elastic Container Service (ECS), you need to specify the full image
 URI,
which includes the ECR repository URI. This allows the container runtime to pull
 the
```

```
correct container image from the ECR repository.
        """
        )
        repository_descriptions = self.ecr_wrapper.describe_repositories(
            [self.repository_name]
        )
        repository_uri = repository_descriptions[0]["repositoryUri"]
        print(f"Repository URI found: {repository_uri}")
        press_enter_to_continue()
        print_dashes()

        print(
            """
* Set an ECR Lifecycle Policy.

An ECR Lifecycle Policy is used to manage the lifecycle of Docker images stored
 in your ECR repositories.
These policies allow you to automatically remove old or unused Docker images from
 your repositories,
freeing up storage space and reducing costs.

This example policy helps to maintain the size and efficiency of the container
 registry
by automatically removing older and potentially unused images, ensuring that the
storage is optimized and the registry remains up-to-date.
            """
        )
        press_enter_to_continue()
        self.put_expiration_policy()
        print(f"An expiration policy was added to the repository.")
        print_dashes()

        print(
            """
* Push a docker image to the Amazon ECR Repository.

The Docker client uses the authorization token is used to authenticate the when
 pushing the image to the
ECR repository.
        """
        )
        decoded_authorization =
 base64.b64decode(authorization_token).decode("utf-8")
        username, password = decoded_authorization.split(":")
```

```
        resp = self.docker_client.api.push(
            repository=repository_uri,
            auth_config={"username": username, "password": password},
            tag=self.tag,
            stream=True,
            decode=True,
        )
        for line in resp:
            print(line)

        print_dashes()

        print("* Verify if the image is in the ECR Repository.")
        image_descriptions = self.ecr_wrapper.describe_images(
            self.repository_name, [self.tag]
        )
        if len(image_descriptions) > 0:
            print("Image found in ECR Repository.")
        else:
            print("Image not found in ECR Repository.")
        press_enter_to_continue()
        print_dashes()

        print(
            "* As an optional step, you can interact with the image in Amazon ECR
 by using the CLI."
        )
        if q.ask(
            "Would you like to view instructions on how to use the CLI to run the
 image? (y/n)",
            q.is_yesno,
        ):
            print(
                f"""
1. Authenticate with ECR - Before you can pull the image from Amazon ECR, you
 need to authenticate with the registry. You can do this using the AWS CLI:

   aws ecr get-login-password --region us-east-1 | docker login --username AWS
 --password-stdin {repository_uri.split("/")[0]}

2. Describe the image using this command:
```

```
   aws ecr describe-images --repository-name {self.repository_name} --image-ids
 imageTag={self.tag}

3. Run the Docker container and view the output using this command:

   docker run --rm {self.full_tag_name}
"""
            )

        self.cleanup(True)

    def cleanup(self, ask: bool):
        """
        Deletes the resources created in this scenario.
        :param ask: If True, prompts the user to confirm before deleting the
 resources.
        """
        if self.repository is not None and (
            not ask
            or q.ask(
                f"Would you like to delete the ECR repository
 '{self.repository_name}? (y/n) "
            )
        ):
            print(f"Deleting the ECR repository '{self.repository_name}'.")
            self.ecr_wrapper.delete_repository(self.repository_name)

        if self.full_tag_name is not None and (
            not ask
            or q.ask(
                f"Would you like to delete the local Docker image
 '{self.full_tag_name}? (y/n) "
            )
        ):
            print(f"Deleting the docker image '{self.full_tag_name}'.")
            self.docker_client.images.remove(self.full_tag_name)

    def grant_role_download_access(self, role_arn: str):
        """
        Grants the specified role access to download images from the ECR
 repository.

        :param role_arn: The ARN of the role to grant access to.
        """
```

```
        policy_json = {
            "Version":"2012-10-17",
            "Statement": [
                {
                    "Sid": "AllowDownload",
                    "Effect": "Allow",
                    "Principal": {"AWS": role_arn},
                    "Action": ["ecr:BatchGetImage"],
                }
            ],
        }

        self.ecr_wrapper.set_repository_policy(
            self.repository_name, json.dumps(policy_json)
        )


    def put_expiration_policy(self):
        """
        Puts an expiration policy on the ECR repository.
        """
        policy_json = {
            "rules": [
                {
                    "rulePriority": 1,
                    "description": "Expire images older than 14 days",
                    "selection": {
                        "tagStatus": "any",
                        "countType": "sinceImagePushed",
                        "countUnit": "days",
                        "countNumber": 14,
                    },
                    "action": {"type": "expire"},
                }
            ]
        }

        self.ecr_wrapper.put_lifecycle_policy(
            self.repository_name, json.dumps(policy_json)
        )



if __name__ == "__main__":
```

```
    parser = argparse.ArgumentParser(
        description="Run Amazon ECR getting started scenario."
    )
    parser.add_argument(
        "--iam-role-arn",
        type=str,
        default=None,
        help="an optional IAM role ARN that will be granted access to download
images from a repository.",
        required=False,
    )
    parser.add_argument(
        "--no-art",
        action="store_true",
        help="accessibility setting that suppresses art in the console output.",
    )
    args = parser.parse_args()
    no_art = args.no_art
    iam_role_arn = args.iam_role_arn
    demo = None
    a_docker_client = None
    try:
        a_docker_client = docker.from_env()
        if not a_docker_client.ping():
            raise docker.errors.DockerException("Docker is not running.")
    except docker.errors.DockerException as err:
        logging.error(
            """
        The Python Docker client could not be created.
        Do you have Docker installed and running?
        Here is the error message:
        %s
        """,
            err,
        )
        sys.exit("Error with Docker.")
    try:
        an_ecr_wrapper = ECRWrapper.from_client()
        demo = ECRGettingStarted(an_ecr_wrapper, a_docker_client)
        demo.run(iam_role_arn)

    except Exception as exception:
        logging.exception("Something went wrong with the demo!")
        if demo is not None:
```

```
            demo.cleanup(False)
```

ECRWrapper classe che racchiude le azioni di Amazon ECR.

```
class ECRWrapper:
    def __init__(self, ecr_client: client):
        self.ecr_client = ecr_client

    @classmethod
    def from_client(cls) -> "ECRWrapper":
        """
        Creates a ECRWrapper instance with a default Amazon ECR client.

        :return: An instance of ECRWrapper initialized with the default Amazon
 ECR client.
        """
        ecr_client = boto3.client("ecr")
        return cls(ecr_client)


    def create_repository(self, repository_name: str) -> dict[str, any]:
        """
        Creates an ECR repository.

        :param repository_name: The name of the repository to create.
        :return: A dictionary of the created repository.
        """
        try:
            response =
 self.ecr_client.create_repository(repositoryName=repository_name)
            return response["repository"]
        except ClientError as err:
            if err.response["Error"]["Code"] ==
 "RepositoryAlreadyExistsException":
                print(f"Repository {repository_name} already exists.")
                response = self.ecr_client.describe_repositories(
                    repositoryNames=[repository_name]
                )
                return self.describe_repositories([repository_name])[0]
            else:
                logger.error(
```

```
                    "Error creating repository %s. Here's why %s",
                    repository_name,
                    err.response["Error"]["Message"],
                )
                raise


    def delete_repository(self, repository_name: str):
        """
        Deletes an ECR repository.

        :param repository_name: The name of the repository to delete.
        """
        try:
            self.ecr_client.delete_repository(
                repositoryName=repository_name, force=True
            )
            print(f"Deleted repository {repository_name}.")
        except ClientError as err:
            logger.error(
                "Couldn't delete repository %s.. Here's why %s",
                repository_name,
                err.response["Error"]["Message"],
            )
            raise


    def set_repository_policy(self, repository_name: str, policy_text: str):
        """
        Sets the policy for an ECR repository.

        :param repository_name: The name of the repository to set the policy for.
        :param policy_text: The policy text to set.
        """
        try:
            self.ecr_client.set_repository_policy(
                repositoryName=repository_name, policyText=policy_text
            )
            print(f"Set repository policy for repository {repository_name}.")
        except ClientError as err:
            if err.response["Error"]["Code"] ==
"RepositoryPolicyNotFoundException":
                logger.error("Repository does not exist. %s.", repository_name)
                raise
```

```
            else:
                logger.error(
                    "Couldn't set repository policy for repository %s. Here's why
%s",
                    repository_name,
                    err.response["Error"]["Message"],
                )
                raise


    def get_repository_policy(self, repository_name: str) -> str:
        """
        Gets the policy for an ECR repository.

        :param repository_name: The name of the repository to get the policy for.
        :return: The policy text.
        """
        try:
            response = self.ecr_client.get_repository_policy(
                repositoryName=repository_name
            )
            return response["policyText"]
        except ClientError as err:
            if err.response["Error"]["Code"] ==
"RepositoryPolicyNotFoundException":
                logger.error("Repository does not exist. %s.", repository_name)
                raise
            else:
                logger.error(
                    "Couldn't get repository policy for repository %s. Here's why
%s",
                    repository_name,
                    err.response["Error"]["Message"],
                )
                raise


    def get_authorization_token(self) -> str:
        """
        Gets an authorization token for an ECR repository.

        :return: The authorization token.
        """
        try:
```

```
            response = self.ecr_client.get_authorization_token()
            return response["authorizationData"][0]["authorizationToken"]
        except ClientError as err:
            logger.error(
                "Couldn't get authorization token. Here's why %s",
                err.response["Error"]["Message"],
            )
            raise


    def describe_repositories(self, repository_names: list[str]) -> list[dict]:
        """
        Describes ECR repositories.

        :param repository_names: The names of the repositories to describe.
        :return: The list of repository descriptions.
        """
        try:
            response = self.ecr_client.describe_repositories(
                repositoryNames=repository_names
            )
            return response["repositories"]
        except ClientError as err:
            logger.error(
                "Couldn't describe repositories. Here's why %s",
                err.response["Error"]["Message"],
            )
            raise


    def put_lifecycle_policy(self, repository_name: str, lifecycle_policy_text:
str):
        """
        Puts a lifecycle policy for an ECR repository.

        :param repository_name: The name of the repository to put the lifecycle
policy for.
        :param lifecycle_policy_text: The lifecycle policy text to put.
        """
        try:
            self.ecr_client.put_lifecycle_policy(
                repositoryName=repository_name,
                lifecyclePolicyText=lifecycle_policy_text,
            )
```

```
            print(f"Put lifecycle policy for repository {repository_name}.")
        except ClientError as err:
            logger.error(
                "Couldn't put lifecycle policy for repository %s. Here's why %s",
                repository_name,
                err.response["Error"]["Message"],
            )
            raise


    def describe_images(
        self, repository_name: str, image_ids: list[str] = None
    ) -> list[dict]:
        """
        Describes ECR images.

        :param repository_name: The name of the repository to describe images
for.
        :param image_ids: The optional IDs of images to describe.
        :return: The list of image descriptions.
        """
        try:
            params = {
                "repositoryName": repository_name,
            }
            if image_ids is not None:
                params["imageIds"] = [{"imageTag": tag} for tag in image_ids]

            paginator = self.ecr_client.get_paginator("describe_images")
            image_descriptions = []
            for page in paginator.paginate(**params):
                image_descriptions.extend(page["imageDetails"])
            return image_descriptions
        except ClientError as err:
            logger.error(
                "Couldn't describe images. Here's why %s",
                err.response["Error"]["Message"],
            )
            raise
```

- Per informazioni dettagliate sull'API, consulta i seguenti argomenti nella documentazione di riferimento dell'API AWS SDK per Python (Boto3).
  - CreateRepository
  - DeleteRepository
  - DescribeImages
  - DescribeRepositories
  - GetAuthorizationToken
  - GetRepositoryPolicy
  - SetRepositoryPolicy
  - StartLifecyclePolicyPreview

Per un elenco completo delle guide per sviluppatori AWS SDK e degli esempi di codice, consulta. Utilizzo di Amazon ECR con un SDK AWS Questo argomento include anche informazioni su come iniziare e dettagli sulle versioni precedenti dell'SDK.

Azioni per l'utilizzo di Amazon ECR AWS SDKs

I seguenti esempi di codice mostrano come eseguire singole azioni Amazon ECR con AWS SDKs. Ogni esempio include un collegamento a GitHub, dove puoi trovare le istruzioni per la configurazione e l'esecuzione del codice.

Gli esempi seguenti includono solo le azioni più comunemente utilizzate. Per un elenco completo, consulta la documentazione di riferimento dell'API Amazon Elastic Container Registry.

Esempi

- Utilizzo CreateRepository con un AWS SDK o una CLI
- Utilizzo DeleteRepository con un AWS SDK o una CLI
- Utilizzo DescribeImages con un AWS SDK o una CLI
- Utilizzo DescribeRepositories con un AWS SDK o una CLI
- Utilizzo GetAuthorizationToken con un AWS SDK o una CLI
- Utilizzo GetRepositoryPolicy con un AWS SDK o una CLI
- Utilizzo ListImages con un AWS SDK o una CLI
- Utilizzare PushImageCmd con un SDK AWS

- Utilizzo PutLifeCyclePolicy con un AWS SDK o una CLI
- Utilizzo SetRepositoryPolicy con un AWS SDK o una CLI
- Utilizzo StartLifecyclePolicyPreview con un AWS SDK o una CLI

Utilizzo **`CreateRepository`** con un AWS SDK o una CLI

Gli esempi di codice seguenti mostrano come utilizzare `CreateRepository`.

Gli esempi di operazioni sono estratti di codice da programmi più grandi e devono essere eseguiti nel contesto. È possibile visualizzare questa operazione nel contesto nel seguente esempio di codice:

- Informazioni di base

CLI

AWS CLI

Esempio 1: come creare un repository.

L'esempio `create-repository` seguente crea un repository all'interno del namespace specificato nel registro predefinito per un account.

```
aws ecr create-repository \
    --repository-name project-a/sample-repo
```

Output:

```
{
    "repository": {
        "registryId": "123456789012",
        "repositoryName": "project-a/sample-repo",
        "repositoryArn": "arn:aws:ecr:us-west-2:123456789012:repository/project-
a/sample-repo"
    }
}
```

Per ulteriori informazioni, consulta Creazione di un repository nella Guida per l'utente di Amazon ECR.

Esempio 2: come creare un repository configurato con l'immutabilità dei tag delle immagini

L'esempio `create-repository` seguente crea un repository configurato per l'immutabilità dei tag nel registro predefinito per un account.

```
aws ecr create-repository \
    --repository-name project-a/sample-repo \
    --image-tag-mutability IMMUTABLE
```

Output:

```
{
    "repository": {
        "registryId": "123456789012",
        "repositoryName": "project-a/sample-repo",
        "repositoryArn": "arn:aws:ecr:us-west-2:123456789012:repository/project-
a/sample-repo",
        "imageTagMutability": "IMMUTABLE"
    }
}
```

Per ulteriori informazioni, consulta Mutabilità del tag immagine nella Guida per l'utente di Amazon ECR.

Esempio 3: come creare un repository configurato con una configurazione di scansione

L'esempio `create-repository` seguente crea un repository configurato per l'esecuzione di una scansione delle vulnerabilità all'esecuzione del push delle immagini nel registro predefinito per un account.

```
aws ecr create-repository \
    --repository-name project-a/sample-repo \
    --image-scanning-configuration scanOnPush=true
```

Output:

```
{
    "repository": {
        "registryId": "123456789012",
        "repositoryName": "project-a/sample-repo",
```

```
        "repositoryArn": "arn:aws:ecr:us-west-2:123456789012:repository/project-
a/sample-repo",
        "imageScanningConfiguration": {
            "scanOnPush": true
        }
    }
}
```

Per ulteriori informazioni, consulta Scansione delle immagini nella Guida per l'utente di Amazon ECR.

- Per i dettagli sull'API, consulta CreateRepository AWS CLICommand Reference.

Java

SDK per Java 2.x

> **Note**
>
> C'è altro su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
   /**
    * Creates an Amazon Elastic Container Registry (Amazon ECR) repository.
    *
    * @param repoName the name of the repository to create.
    * @return the Amazon Resource Name (ARN) of the created repository, or an
empty string if the operation failed.
    * @throws IllegalArgumentException     If repository name is invalid.
    * @throws RuntimeException             if an error occurs while creating the
repository.
    */
   public String createECRRepository(String repoName) {
       if (repoName == null || repoName.isEmpty()) {
           throw new IllegalArgumentException("Repository name cannot be null or
empty");
       }

       CreateRepositoryRequest request = CreateRepositoryRequest.builder()
           .repositoryName(repoName)
           .build();
```

```
        CompletableFuture<CreateRepositoryResponse> response =
getAsyncClient().createRepository(request);
        try {
            CreateRepositoryResponse result = response.join();
            if (result != null) {
                System.out.println("The " + repoName + " repository was created
successfully.");
                return result.repository().repositoryArn();
            } else {
                throw new RuntimeException("Unexpected response type");
            }
        } catch (CompletionException e) {
            Throwable cause = e.getCause();
            if (cause instanceof EcrException ex) {
                if
("RepositoryAlreadyExistsException".equals(ex.awsErrorDetails().errorCode())) {
                    System.out.println("The Amazon ECR repository already exists,
moving on...");
                    DescribeRepositoriesRequest describeRequest =
DescribeRepositoriesRequest.builder()
                        .repositoryNames(repoName)
                        .build();
                    DescribeRepositoriesResponse describeResponse =
getAsyncClient().describeRepositories(describeRequest).join();
                    return
describeResponse.repositories().get(0).repositoryArn();
                } else {
                    throw new RuntimeException(ex);
                }
            } else {
                throw new RuntimeException(e);
            }
        }
    }
```

- Per i dettagli sull'API, CreateRepositoryconsulta AWS SDK for Java 2.x API Reference.

Kotlin

SDK per Kotlin

> **Note**
>
> C'è altro su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
    /**
     * Creates an Amazon Elastic Container Registry (Amazon ECR) repository.
     *
     * @param repoName the name of the repository to create.
     * @return the Amazon Resource Name (ARN) of the created repository, or an
empty string if the operation failed.
     * @throws RepositoryAlreadyExistsException if the repository exists.
     * @throws EcrException         if an error occurs while creating the
repository.
     */
    suspend fun createECRRepository(repoName: String?): String? {
        val request =
            CreateRepositoryRequest {
                repositoryName = repoName
            }

        return try {
            EcrClient.fromEnvironment { region = "us-east-1" }.use { ecrClient ->
                val response = ecrClient.createRepository(request)
                response.repository?.repositoryArn
            }
        } catch (e: RepositoryAlreadyExistsException) {
            println("Repository already exists: $repoName")
            repoName?.let { getRepoARN(it) }
        } catch (e: EcrException) {
            println("An error occurred: ${e.message}")
            null
        }
    }
```

- Per i dettagli sull'API, CreateRepositoryconsulta AWS SDK for Kotlin API reference.

Python

SDK per Python (Boto3)

> **Note**
>
> C'è altro su. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
class ECRWrapper:
    def __init__(self, ecr_client: client):
        self.ecr_client = ecr_client

    @classmethod
    def from_client(cls) -> "ECRWrapper":
        """
        Creates a ECRWrapper instance with a default Amazon ECR client.

        :return: An instance of ECRWrapper initialized with the default Amazon
 ECR client.
        """
        ecr_client = boto3.client("ecr")
        return cls(ecr_client)


    def create_repository(self, repository_name: str) -> dict[str, any]:
        """
        Creates an ECR repository.

        :param repository_name: The name of the repository to create.
        :return: A dictionary of the created repository.
        """
        try:
            response =
 self.ecr_client.create_repository(repositoryName=repository_name)
            return response["repository"]
        except ClientError as err:
            if err.response["Error"]["Code"] ==
 "RepositoryAlreadyExistsException":
```

```
                print(f"Repository {repository_name} already exists.")
                response = self.ecr_client.describe_repositories(
                    repositoryNames=[repository_name]
                )
                return self.describe_repositories([repository_name])[0]
            else:
                logger.error(
                    "Error creating repository %s. Here's why %s",
                    repository_name,
                    err.response["Error"]["Message"],
                )
                raise
```

- Per i dettagli sull'API, consulta CreateRepository AWSSDK for Python (Boto3) API Reference.

Per un elenco completo delle guide per sviluppatori AWS SDK e degli esempi di codice, consulta. Utilizzo di Amazon ECR con un SDK AWS Questo argomento include anche informazioni su come iniziare e dettagli sulle versioni precedenti dell'SDK.

Utilizzo **`DeleteRepository`** con un AWS SDK o una CLI

Gli esempi di codice seguenti mostrano come utilizzare `DeleteRepository`.

Gli esempi di operazioni sono estratti di codice da programmi più grandi e devono essere eseguiti nel contesto. È possibile visualizzare questa operazione nel contesto nel seguente esempio di codice:

- Informazioni di base

CLI

AWS CLI

Come eliminare un repository

L'esempio `delete-repository` seguente forza l'eliminazione del repository specificato nel registro predefinito per un account. Il flag `--force` è obbligatorio se il repository contiene immagini.

```
aws ecr delete-repository \
    --repository-name ubuntu \
    --force
```

Output:

```
{
    "repository": {
        "registryId": "123456789012",
        "repositoryName": "ubuntu",
        "repositoryArn": "arn:aws:ecr:us-west-2:123456789012:repository/ubuntu"
    }
}
```

Per ulteriori informazioni, consulta Eliminazione di un repository nella Guida per l'utente di Amazon ECR.

- Per i dettagli sull'API, consulta DeleteRepository AWS CLICommand Reference.

Java

SDK per Java 2.x

> **Note**
>
> C'è altro su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
    /**
     * Deletes an ECR (Elastic Container Registry) repository.
     *
     * @param repoName the name of the repository to delete.
     * @throws IllegalArgumentException if the repository name is null or empty.
     * @throws EcrException if there is an error deleting the repository.
     * @throws RuntimeException if an unexpected error occurs during the deletion
 process.
     */
    public void deleteECRRepository(String repoName) {
        if (repoName == null || repoName.isEmpty()) {
```

```
            throw new IllegalArgumentException("Repository name cannot be null or
empty");
        }

        DeleteRepositoryRequest repositoryRequest =
DeleteRepositoryRequest.builder()
            .force(true)
            .repositoryName(repoName)
            .build();

        CompletableFuture<DeleteRepositoryResponse> response =
getAsyncClient().deleteRepository(repositoryRequest);
        response.whenComplete((deleteRepositoryResponse, ex) -> {
            if (deleteRepositoryResponse != null) {
                System.out.println("You have successfully deleted the " +
repoName + " repository");
            } else {
                Throwable cause = ex.getCause();
                if (cause instanceof EcrException) {
                    throw (EcrException) cause;
                } else {
                    throw new RuntimeException("Unexpected error: " +
cause.getMessage(), cause);
                }
            }
        });

        // Wait for the CompletableFuture to complete
        response.join();
    }
```

- Per i dettagli sull'API, DeleteRepositoryconsulta AWS SDK for Java 2.x API Reference.

Kotlin

SDK per Kotlin

> **Note**
>
> C'è altro su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
    /**
     * Deletes an ECR (Elastic Container Registry) repository.
     *
     * @param repoName the name of the repository to delete.
     */
    suspend fun deleteECRRepository(repoName: String) {
        if (repoName.isNullOrEmpty()) {
            throw IllegalArgumentException("Repository name cannot be null or
 empty")
        }

        val repositoryRequest =
            DeleteRepositoryRequest {
                force = true
                repositoryName = repoName
            }

        EcrClient.fromEnvironment { region = "us-east-1" }.use { ecrClient ->
            ecrClient.deleteRepository(repositoryRequest)
            println("You have successfully deleted the $repoName repository")
        }
    }
```

- Per i dettagli sull'API, DeleteRepositoryconsulta AWS SDK for Kotlin API reference.

Python

SDK per Python (Boto3)

> **Note**
>
> C'è altro su. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
class ECRWrapper:
    def __init__(self, ecr_client: client):
        self.ecr_client = ecr_client
```

```
    @classmethod
    def from_client(cls) -> "ECRWrapper":
        """
        Creates a ECRWrapper instance with a default Amazon ECR client.

        :return: An instance of ECRWrapper initialized with the default Amazon
ECR client.
        """
        ecr_client = boto3.client("ecr")
        return cls(ecr_client)


    def delete_repository(self, repository_name: str):
        """
        Deletes an ECR repository.

        :param repository_name: The name of the repository to delete.
        """
        try:
            self.ecr_client.delete_repository(
                repositoryName=repository_name, force=True
            )
            print(f"Deleted repository {repository_name}.")
        except ClientError as err:
            logger.error(
                "Couldn't delete repository %s.. Here's why %s",
                repository_name,
                err.response["Error"]["Message"],
            )
            raise
```

- Per i dettagli sull'API, consulta DeleteRepository AWSSDK for Python (Boto3) API Reference.

Per un elenco completo delle guide per sviluppatori AWS SDK e degli esempi di codice, consulta. Utilizzo di Amazon ECR con un SDK AWS Questo argomento include anche informazioni su come iniziare e dettagli sulle versioni precedenti dell'SDK.

Utilizzo **`DescribeImages`** con un AWS SDK o una CLI

Gli esempi di codice seguenti mostrano come utilizzare `DescribeImages`.

Gli esempi di operazioni sono estratti di codice da programmi più grandi e devono essere eseguiti nel contesto. È possibile visualizzare questa operazione nel contesto nel seguente esempio di codice:

- Informazioni di base

CLI

AWS CLI

Come descrivere un'immagine in un repository

L'esempio `describe-images` seguente mostra i dettagli di un'immagine nel repository `cluster-autoscaler` con il tag `v1.13.6`.

```
aws ecr describe-images \
    --repository-name cluster-autoscaler \
    --image-ids imageTag=v1.13.6
```

Output:

```
{
    "imageDetails": [
        {
            "registryId": "012345678910",
            "repositoryName": "cluster-autoscaler",
            "imageDigest":
 "sha256:4a1c6567c38904384ebc64e35b7eeddd8451110c299e3368d2210066487d97e5",
            "imageTags": [
                "v1.13.6"
            ],
            "imageSizeInBytes": 48318255,
            "imagePushedAt": 1565128275.0
        }
    ]
}
```

- Per i dettagli sull'API, consulta DescribeImages AWS CLICommand Reference.

Java

SDK per Java 2.x

> **Note**
>
> C'è altro su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
   /**
    * Verifies the existence of an image in an Amazon Elastic Container Registry
(Amazon ECR) repository asynchronously.
    *
    * @param repositoryName The name of the Amazon ECR repository.
    * @param imageTag       The tag of the image to verify.
    * @throws EcrException             if there is an error retrieving the image
information from Amazon ECR.
    * @throws CompletionException      if the asynchronous operation completes
exceptionally.
    */
   public void verifyImage(String repositoryName, String imageTag) {
       DescribeImagesRequest request = DescribeImagesRequest.builder()
           .repositoryName(repositoryName)
           .imageIds(ImageIdentifier.builder().imageTag(imageTag).build())
           .build();

       CompletableFuture<DescribeImagesResponse> response =
getAsyncClient().describeImages(request);
       response.whenComplete((describeImagesResponse, ex) -> {
           if (ex != null) {
               if (ex instanceof CompletionException) {
                   Throwable cause = ex.getCause();
                   if (cause instanceof EcrException) {
                       throw (EcrException) cause;
                   } else {
                       throw new RuntimeException("Unexpected error: " +
cause.getMessage(), cause);
                   }
               } else {
                   throw new RuntimeException("Unexpected error: " +
ex.getCause());
```

```
                }
            } else if (describeImagesResponse != null && !
describeImagesResponse.imageDetails().isEmpty()) {
                System.out.println("Image is present in the repository.");
            } else {
                System.out.println("Image is not present in the repository.");
            }
        });

        // Wait for the CompletableFuture to complete.
        response.join();
    }
```

- Per i dettagli sull'API, DescribeImagesconsulta AWS SDK for Java 2.x API Reference.

Kotlin

SDK per Kotlin

> **Note**
>
> C'è altro su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
    /**
     * Verifies the existence of an image in an Amazon Elastic Container Registry
(Amazon ECR) repository asynchronously.
     *
     * @param repositoryName The name of the Amazon ECR repository.
     * @param imageTag       The tag of the image to verify.
     */
    suspend fun verifyImage(
        repoName: String?,
        imageTagVal: String?,
    ) {
        require(!(repoName == null || repoName.isEmpty())) { "Repository name
cannot be null or empty" }
        require(!(imageTagVal == null || imageTagVal.isEmpty())) { "Image tag
cannot be null or empty" }
```

```
        val imageId =
            ImageIdentifier {
                imageTag = imageTagVal
            }
        val request =
            DescribeImagesRequest {
                repositoryName = repoName
                imageIds = listOf(imageId)
            }

        EcrClient.fromEnvironment { region = "us-east-1" }.use { ecrClient ->
            val describeImagesResponse = ecrClient.describeImages(request)
            if (describeImagesResponse != null && !
describeImagesResponse.imageDetails?.isEmpty()!!) {
                println("Image is present in the repository.")
            } else {
                println("Image is not present in the repository.")
            }
        }
    }
```

- Per i dettagli sull'API, DescribeImagesconsulta AWS SDK for Kotlin API reference.

Python

SDK per Python (Boto3)

> **Note**
>
> C'è altro su. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
class ECRWrapper:
    def __init__(self, ecr_client: client):
        self.ecr_client = ecr_client

    @classmethod
    def from_client(cls) -> "ECRWrapper":
        """
```

```
        Creates a ECRWrapper instance with a default Amazon ECR client.

        :return: An instance of ECRWrapper initialized with the default Amazon
ECR client.
        """
        ecr_client = boto3.client("ecr")
        return cls(ecr_client)


    def describe_images(
        self, repository_name: str, image_ids: list[str] = None
    ) -> list[dict]:
        """
        Describes ECR images.

        :param repository_name: The name of the repository to describe images
for.
        :param image_ids: The optional IDs of images to describe.
        :return: The list of image descriptions.
        """
        try:
            params = {
                "repositoryName": repository_name,
            }
            if image_ids is not None:
                params["imageIds"] = [{"imageTag": tag} for tag in image_ids]

            paginator = self.ecr_client.get_paginator("describe_images")
            image_descriptions = []
            for page in paginator.paginate(**params):
                image_descriptions.extend(page["imageDetails"])
            return image_descriptions
        except ClientError as err:
            logger.error(
                "Couldn't describe images. Here's why %s",
                err.response["Error"]["Message"],
            )
            raise
```

- Per i dettagli sull'API, consulta DescribeImages AWSSDK for Python (Boto3) API Reference.

Per un elenco completo delle guide per sviluppatori AWS SDK e degli esempi di codice, consulta. Utilizzo di Amazon ECR con un SDK AWS Questo argomento include anche informazioni su come iniziare e dettagli sulle versioni precedenti dell'SDK.

Utilizzo **`DescribeRepositories`** con un AWS SDK o una CLI

Gli esempi di codice seguenti mostrano come utilizzare `DescribeRepositories`.

Gli esempi di operazioni sono estratti di codice da programmi più grandi e devono essere eseguiti nel contesto. È possibile visualizzare questa operazione nel contesto nel seguente esempio di codice:

- Informazioni di base

CLI

AWS CLI

Come descrivere i repository in un registro

Questo esempio descrive i repository nel registro predefinito di un account.

Comando:

```
aws ecr describe-repositories
```

Output:

```
{
    "repositories": [
        {
            "registryId": "012345678910",
            "repositoryName": "ubuntu",
            "repositoryArn": "arn:aws:ecr:us-west-2:012345678910:repository/
ubuntu"
        },
        {
            "registryId": "012345678910",
            "repositoryName": "test",
            "repositoryArn": "arn:aws:ecr:us-west-2:012345678910:repository/test"
        }
    ]
```

```
}
```

- Per i dettagli sull'API, consulta DescribeRepositories AWS CLICommand Reference.

Java

SDK per Java 2.x

> **Note**
>
> C'è altro su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
   /**
    * Retrieves the repository URI for the specified repository name.
    *
    * @param repoName the name of the repository to retrieve the URI for.
    * @return the repository URI for the specified repository name.
    * @throws EcrException        if there is an error retrieving the repository
information.
    * @throws CompletionException if the asynchronous operation completes
exceptionally.
    */
   public void getRepositoryURI(String repoName) {
       DescribeRepositoriesRequest request =
DescribeRepositoriesRequest.builder()
           .repositoryNames(repoName)
           .build();

       CompletableFuture<DescribeRepositoriesResponse> response =
getAsyncClient().describeRepositories(request);
       response.whenComplete((describeRepositoriesResponse, ex) -> {
           if (ex != null) {
               Throwable cause = ex.getCause();
               if (cause instanceof InterruptedException) {
                   Thread.currentThread().interrupt();
                   String errorMessage = "Thread interrupted while waiting for
asynchronous operation: " + cause.getMessage();
                   throw new RuntimeException(errorMessage, cause);
               } else if (cause instanceof EcrException) {
                   throw (EcrException) cause;
```

```
                } else {
                    String errorMessage = "Unexpected error: " +
cause.getMessage();
                    throw new RuntimeException(errorMessage, cause);
                }
            } else {
                if (describeRepositoriesResponse != null) {
                    if (!describeRepositoriesResponse.repositories().isEmpty()) {
                        String repositoryUri =
describeRepositoriesResponse.repositories().get(0).repositoryUri();
                        System.out.println("Repository URI found: " +
repositoryUri);
                    } else {
                        System.out.println("No repositories found for the given
name.");
                    }
                } else {
                    System.err.println("No response received from
describeRepositories.");
                }
            }
        });
        response.join();
    }
```

- Per i dettagli sull'API, DescribeRepositoriesconsulta AWS SDK for Java 2.x API Reference.

Kotlin

SDK per Kotlin

> **Note**
>
> C'è altro su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
/**
 * Retrieves the repository URI for the specified repository name.
 *
```

```
     * @param repoName the name of the repository to retrieve the URI for.
     * @return the repository URI for the specified repository name.
     */
    suspend fun getRepositoryURI(repoName: String?): String? {
        require(!(repoName == null || repoName.isEmpty())) { "Repository name
 cannot be null or empty" }
        val request =
            DescribeRepositoriesRequest {
                repositoryNames = listOf(repoName)
            }

        EcrClient.fromEnvironment { region = "us-east-1" }.use { ecrClient ->
            val describeRepositoriesResponse =
 ecrClient.describeRepositories(request)
            if (!describeRepositoriesResponse.repositories?.isEmpty()!!) {
                return
 describeRepositoriesResponse?.repositories?.get(0)?.repositoryUri
            } else {
                println("No repositories found for the given name.")
                return ""
            }
        }
    }
```

- Per i dettagli sull'API, DescribeRepositoriesconsulta AWS SDK for Kotlin API reference.

Python

SDK per Python (Boto3)

> **Note**
>
> C'è altro su. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
class ECRWrapper:
    def __init__(self, ecr_client: client):
        self.ecr_client = ecr_client

    @classmethod
```

```
    def from_client(cls) -> "ECRWrapper":
        """
        Creates a ECRWrapper instance with a default Amazon ECR client.

        :return: An instance of ECRWrapper initialized with the default Amazon
ECR client.
        """
        ecr_client = boto3.client("ecr")
        return cls(ecr_client)


    def describe_repositories(self, repository_names: list[str]) -> list[dict]:
        """
        Describes ECR repositories.

        :param repository_names: The names of the repositories to describe.
        :return: The list of repository descriptions.
        """
        try:
            response = self.ecr_client.describe_repositories(
                repositoryNames=repository_names
            )
            return response["repositories"]
        except ClientError as err:
            logger.error(
                "Couldn't describe repositories. Here's why %s",
                err.response["Error"]["Message"],
            )
            raise
```

- Per i dettagli sull'API, consulta DescribeRepositories AWSSDK for Python (Boto3) API Reference.

Rust

SDK per Rust

> **Note**
>
> C'è altro da fare. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
async fn show_repos(client: &aws_sdk_ecr::Client) -> Result<(),
 aws_sdk_ecr::Error> {
    let rsp = client.describe_repositories().send().await?;

    let repos = rsp.repositories();

    println!("Found {} repositories:", repos.len());

    for repo in repos {
        println!("  ARN:  {}", repo.repository_arn().unwrap());
        println!("  Name: {}", repo.repository_name().unwrap());
    }

    Ok(())
}
```

- Per i dettagli sulle API, consulta il riferimento DescribeRepositoriesall'API AWS SDK for Rust.

Per un elenco completo delle guide per sviluppatori AWS SDK e degli esempi di codice, consulta. Utilizzo di Amazon ECR con un SDK AWS Questo argomento include anche informazioni su come iniziare e dettagli sulle versioni precedenti dell'SDK.

Utilizzo **`GetAuthorizationToken`** con un AWS SDK o una CLI

Gli esempi di codice seguenti mostrano come utilizzare `GetAuthorizationToken`.

Gli esempi di operazioni sono estratti di codice da programmi più grandi e devono essere eseguiti nel contesto. È possibile visualizzare questa operazione nel contesto nel seguente esempio di codice:

- Informazioni di base

CLI

AWS CLI

Come ottenere un token di autorizzazione per il registro predefinito

L'esempio `get-authorization-token` seguente ottiene un token di autorizzazione per il registro predefinito.

```
aws ecr get-authorization-token
```

Output:

```
{
    "authorizationData": [
        {
            "authorizationToken": "QVdTOkN...",
            "expiresAt": 1448875853.241,
            "proxyEndpoint": "https://123456789012.dkr.ecr.us-
west-2.amazonaws.com"
        }
    ]
}
```

- Per i dettagli sull'API, consulta GetAuthorizationToken AWS CLICommand Reference.

Java

SDK per Java 2.x

> **Note**
>
> C'è altro su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
    /**
```

```
    * Retrieves the authorization token for Amazon Elastic Container Registry
(ECR).
    * This method makes an asynchronous call to the ECR client to retrieve the
authorization token.
    * If the operation is successful, the method prints the token to the
console.
    * If an exception occurs, the method handles the exception and prints the
error message.
    *
    * @throws EcrException     if there is an error retrieving the authorization
token from ECR.
    * @throws RuntimeException if there is an unexpected error during the
operation.
    */
   public void getAuthToken() {
       CompletableFuture<GetAuthorizationTokenResponse> response =
getAsyncClient().getAuthorizationToken();
       response.whenComplete((authorizationTokenResponse, ex) -> {
           if (authorizationTokenResponse != null) {
               AuthorizationData authorizationData =
authorizationTokenResponse.authorizationData().get(0);
               String token = authorizationData.authorizationToken();
               if (!token.isEmpty()) {
                   System.out.println("The token was successfully retrieved.");
               }
           } else {
               if (ex.getCause() instanceof EcrException) {
                   throw (EcrException) ex.getCause();
               } else {
                   String errorMessage = "Unexpected error occurred: " +
ex.getMessage();
                   throw new RuntimeException(errorMessage, ex); // Rethrow the
exception
               }
           }
       });
       response.join();
   }
```

- Per i dettagli sull'API, GetAuthorizationTokenconsulta AWS SDK for Java 2.x API Reference.

Kotlin

SDK per Kotlin

> **Note**
>
> C'è altro su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
    /**
     * Retrieves the authorization token for Amazon Elastic Container Registry
(ECR).
     *
     */
    suspend fun getAuthToken() {
        EcrClient.fromEnvironment { region = "us-east-1" }.use { ecrClient ->
            // Retrieve the authorization token for ECR.
            val response = ecrClient.getAuthorizationToken()
            val authorizationData = response.authorizationData?.get(0)
            val token = authorizationData?.authorizationToken
            if (token != null) {
                println("The token was successfully retrieved.")
            }
        }
    }
```

- Per i dettagli sull'API, GetAuthorizationTokenconsulta AWS SDK for Kotlin API reference.

Python

SDK per Python (Boto3)

> **Note**
>
> C'è altro su. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
class ECRWrapper:
    def __init__(self, ecr_client: client):
        self.ecr_client = ecr_client

    @classmethod
    def from_client(cls) -> "ECRWrapper":
        """
        Creates a ECRWrapper instance with a default Amazon ECR client.

        :return: An instance of ECRWrapper initialized with the default Amazon
 ECR client.
        """
        ecr_client = boto3.client("ecr")
        return cls(ecr_client)


    def get_authorization_token(self) -> str:
        """
        Gets an authorization token for an ECR repository.

        :return: The authorization token.
        """
        try:
            response = self.ecr_client.get_authorization_token()
            return response["authorizationData"][0]["authorizationToken"]
        except ClientError as err:
            logger.error(
                "Couldn't get authorization token. Here's why %s",
                err.response["Error"]["Message"],
            )
            raise
```

- Per i dettagli sull'API, consulta GetAuthorizationToken AWSSDK for Python (Boto3) API Reference.

Per un elenco completo delle guide per sviluppatori AWS SDK e degli esempi di codice, consulta. Utilizzo di Amazon ECR con un SDK AWS Questo argomento include anche informazioni su come iniziare e dettagli sulle versioni precedenti dell'SDK.

Utilizzo **GetRepositoryPolicy** con un AWS SDK o una CLI

Gli esempi di codice seguenti mostrano come utilizzare `GetRepositoryPolicy`.

Gli esempi di operazioni sono estratti di codice da programmi più grandi e devono essere eseguiti nel contesto. È possibile visualizzare questa operazione nel contesto nel seguente esempio di codice:

- Informazioni di base

CLI

AWS CLI

Come recuperare la policy del repository specificato

L'esempio `get-repository-policy` seguente mostra i dettagli della policy di repository per il repository `cluster-autoscaler`.

```
aws ecr get-repository-policy \
    --repository-name cluster-autoscaler
```

Output:

```
{
    "registryId": "012345678910",
    "repositoryName": "cluster-autoscaler",
    "policyText": "{\n  \"Version\" : \"2008-10-17\",\n  \"Statement\" :
 [ {\n    \"Sid\" : \"allow public pull\",\n    \"Effect\" : \"Allow\",\n
 \"Principal\" : \"*\",\n    \"Action\" : [ \"ecr:BatchCheckLayerAvailability\",
 \"ecr:BatchGetImage\", \"ecr:GetDownloadUrlForLayer\" ]\n  } ]\n}"
}
```

- Per i dettagli sull'API, consulta GetRepositoryPolicy AWS CLICommand Reference.

Java

SDK per Java 2.x

> **Note**
>
> C'è altro su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
   /**
    * Gets the repository policy for the specified repository.
    *
    * @param repoName the name of the repository.
    * @throws EcrException if an AWS error occurs while getting the repository
policy.
    */
   public String getRepoPolicy(String repoName) {
       if (repoName == null || repoName.isEmpty()) {
           throw new IllegalArgumentException("Repository name cannot be null or
empty");
       }

       GetRepositoryPolicyRequest getRepositoryPolicyRequest =
GetRepositoryPolicyRequest.builder()
           .repositoryName(repoName)
           .build();

       CompletableFuture<GetRepositoryPolicyResponse> response =
getAsyncClient().getRepositoryPolicy(getRepositoryPolicyRequest);
       response.whenComplete((resp, ex) -> {
           if (resp != null) {
               System.out.println("Repository policy retrieved successfully.");
           } else {
               if (ex.getCause() instanceof EcrException) {
                   throw (EcrException) ex.getCause();
               } else {
                   String errorMessage = "Unexpected error occurred: " +
ex.getMessage();
                   throw new RuntimeException(errorMessage, ex);
               }
           }
```

```
        });

        GetRepositoryPolicyResponse result = response.join();
        return result != null ? result.policyText() : null;
    }
```

- Per i dettagli sull'API, GetRepositoryPolicyconsulta AWS SDK for Java 2.x API Reference.

Kotlin

SDK per Kotlin

> **Note**
>
> C'è altro su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
    /**
     * Gets the repository policy for the specified repository.
     *
     * @param repoName the name of the repository.
     */
    suspend fun getRepoPolicy(repoName: String?): String? {
        require(!(repoName == null || repoName.isEmpty())) { "Repository name
cannot be null or empty" }

        // Create the request
        val getRepositoryPolicyRequest =
            GetRepositoryPolicyRequest {
                repositoryName = repoName
            }
        EcrClient.fromEnvironment { region = "us-east-1" }.use { ecrClient ->
            val response =
ecrClient.getRepositoryPolicy(getRepositoryPolicyRequest)
            val responseText = response.policyText
            return responseText
        }
    }
```

- Per i dettagli sull'API, GetRepositoryPolicyconsulta AWS SDK for Kotlin API reference.

Python

SDK per Python (Boto3)

> **Note**
>
> C'è altro su. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
class ECRWrapper:
    def __init__(self, ecr_client: client):
        self.ecr_client = ecr_client

    @classmethod
    def from_client(cls) -> "ECRWrapper":
        """
        Creates a ECRWrapper instance with a default Amazon ECR client.

        :return: An instance of ECRWrapper initialized with the default Amazon
 ECR client.
        """
        ecr_client = boto3.client("ecr")
        return cls(ecr_client)


    def get_repository_policy(self, repository_name: str) -> str:
        """
        Gets the policy for an ECR repository.

        :param repository_name: The name of the repository to get the policy for.
        :return: The policy text.
        """
        try:
            response = self.ecr_client.get_repository_policy(
                repositoryName=repository_name
            )
            return response["policyText"]
```

```
        except ClientError as err:
            if err.response["Error"]["Code"] ==
"RepositoryPolicyNotFoundException":
                logger.error("Repository does not exist. %s.", repository_name)
                raise
            else:
                logger.error(
                    "Couldn't get repository policy for repository %s. Here's why
%s",
                    repository_name,
                    err.response["Error"]["Message"],
                )
                raise
```

- Per i dettagli sull'API, consulta GetRepositoryPolicy AWSSDK for Python (Boto3) API Reference.

Per un elenco completo delle guide per sviluppatori AWS SDK e degli esempi di codice, consulta. Utilizzo di Amazon ECR con un SDK AWS Questo argomento include anche informazioni su come iniziare e dettagli sulle versioni precedenti dell'SDK.

Utilizzo **`ListImages`** con un AWS SDK o una CLI

Gli esempi di codice seguenti mostrano come utilizzare `ListImages`.

CLI

AWS CLI

Come elencare le immagini in un repository

L'esempio `list-images` seguente mostra un elenco delle immagini nel repository `cluster-autoscaler`.

```
aws ecr list-images \
    --repository-name cluster-autoscaler
```

Output:

```
{
```

```
    "imageIds": [
        {
            "imageDigest":
 "sha256:99c6fb4377e9a420a1eb3b410a951c9f464eff3b7dbc76c65e434e39b94b6570",
            "imageTag": "v1.13.8"
        },
        {
            "imageDigest":
 "sha256:99c6fb4377e9a420a1eb3b410a951c9f464eff3b7dbc76c65e434e39b94b6570",
            "imageTag": "v1.13.7"
        },
        {
            "imageDigest":
 "sha256:4a1c6567c38904384ebc64e35b7eeddd8451110c299e3368d2210066487d97e5",
            "imageTag": "v1.13.6"
        }
    ]
}
```

- Per i dettagli sull'API, consulta ListImages AWS CLICommand Reference.

Rust

SDK per Rust

> **Note**
>
> C'è altro da fare. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
async fn show_images(
    client: &aws_sdk_ecr::Client,
    repository: &str,
) -> Result<(), aws_sdk_ecr::Error> {
    let rsp = client
        .list_images()
        .repository_name(repository)
        .send()
        .await?;

    let images = rsp.image_ids();
```

```
    println!("found {} images", images.len());

    for image in images {
        println!(
            "image: {}:{}",
            image.image_tag().unwrap(),
            image.image_digest().unwrap()
        );
    }

    Ok(())
}
```

- Per i dettagli sulle API, consulta il riferimento ListImagesall'API AWS SDK for Rust.

Per un elenco completo delle guide per sviluppatori AWS SDK e degli esempi di codice, consulta. Utilizzo di Amazon ECR con un SDK AWS Questo argomento include anche informazioni su come iniziare e dettagli sulle versioni precedenti dell'SDK.

Utilizzare **`PushImageCmd`** con un SDK AWS

Gli esempi di codice seguenti mostrano come utilizzare `PushImageCmd`.

Java

SDK per Java 2.x

> **Note**
>
> C'è altro su. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
    /**
     * Pushes a Docker image to an Amazon Elastic Container Registry (ECR)
repository.
     *
     * @param repoName  the name of the ECR repository to push the image to.
     * @param imageName the name of the Docker image.
```

```
     */
    public void pushDockerImage(String repoName, String imageName) {
        System.out.println("Pushing " + imageName + " to Amazon ECR will take a
 few seconds.");
        CompletableFuture<AuthConfig> authResponseFuture =
 getAsyncClient().getAuthorizationToken()
            .thenApply(response -> {
                String token =
 response.authorizationData().get(0).authorizationToken();
                String decodedToken = new
 String(Base64.getDecoder().decode(token));
                String password = decodedToken.substring(4);

                DescribeRepositoriesResponse descrRepoResponse =
 getAsyncClient().describeRepositories(b -> b.repositoryNames(repoName)).join();
                Repository repoData =
 descrRepoResponse.repositories().stream().filter(r ->
 r.repositoryName().equals(repoName)).findFirst().orElse(null);
                assert repoData != null;
                String registryURL = repoData.repositoryUri().split("/")[0];

                AuthConfig authConfig = new AuthConfig()
                    .withUsername("AWS")
                    .withPassword(password)
                    .withRegistryAddress(registryURL);
                return authConfig;
            })
            .thenCompose(authConfig -> {
                DescribeRepositoriesResponse descrRepoResponse =
 getAsyncClient().describeRepositories(b -> b.repositoryNames(repoName)).join();
                Repository repoData =
 descrRepoResponse.repositories().stream().filter(r ->
 r.repositoryName().equals(repoName)).findFirst().orElse(null);
                getDockerClient().tagImageCmd(imageName + ":latest",
 repoData.repositoryUri() + ":latest", imageName).exec();
                try {

 getDockerClient().pushImageCmd(repoData.repositoryUri()).withTag("echo-
text").withAuthConfig(authConfig).start().awaitCompletion();
                    System.out.println("The " + imageName + " was pushed to
 ECR");

                } catch (InterruptedException e) {
                    throw (RuntimeException) e.getCause();
```

```
                }
                return CompletableFuture.completedFuture(authConfig);
            });

        authResponseFuture.join();
    }
```

- Per i dettagli sull'API, PushImageCmdconsulta AWS SDK for Java 2.x API Reference.

Kotlin

SDK per Kotlin

> **Note**
>
> C'è altro su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
   /**
    * Pushes a Docker image to an Amazon Elastic Container Registry (ECR)
repository.
    *
    * @param repoName the name of the ECR repository to push the image to.
    * @param imageName the name of the Docker image.
    */
   suspend fun pushDockerImage(
       repoName: String,
       imageName: String,
   ) {
       println("Pushing $imageName to $repoName will take a few seconds")
       val authConfig = getAuthConfig(repoName)

       EcrClient.fromEnvironment { region = "us-east-1" }.use { ecrClient ->
           val desRequest =
               DescribeRepositoriesRequest {
                   repositoryNames = listOf(repoName)
               }

           val describeRepoResponse = ecrClient.describeRepositories(desRequest)
```

```
            val repoData =
                describeRepoResponse.repositories?.firstOrNull
{ it.repositoryName == repoName }
                    ?: throw RuntimeException("Repository not found: $repoName")

            val tagImageCmd = getDockerClient()?.tagImageCmd("$imageName",
"${repoData.repositoryUri}", imageName)
            if (tagImageCmd != null) {
                tagImageCmd.exec()
            }
            val pushImageCmd =
                repoData.repositoryUri?.let {
                    dockerClient?.pushImageCmd(it)
                        // ?.withTag("latest")
                        ?.withAuthConfig(authConfig)
                }

            try {
                if (pushImageCmd != null) {
                    pushImageCmd.start().awaitCompletion()
                }
                println("The $imageName was pushed to Amazon ECR")
            } catch (e: IOException) {
                throw RuntimeException(e)
            }
        }
    }
```

- Per i dettagli sull'API, PushImageCmdconsulta AWS SDK for Kotlin API reference.

Per un elenco completo delle guide per sviluppatori AWS SDK e degli esempi di codice, consulta. Utilizzo di Amazon ECR con un SDK AWS Questo argomento include anche informazioni su come iniziare e dettagli sulle versioni precedenti dell'SDK.

Utilizzo **`PutLifeCyclePolicy`** con un AWS SDK o una CLI

Gli esempi di codice seguenti mostrano come utilizzare `PutLifeCyclePolicy`.

CLI

AWS CLI

Come creare una policy del ciclo di vita

L'esempio `put-lifecycle-policy` seguente crea una policy del ciclo di vita per il repository specificato nel registro predefinito per un account.

```
aws ecr put-lifecycle-policy \
    --repository-name "project-a/amazon-ecs-sample" \
    --lifecycle-policy-text "file://policy.json"
```

Contenuto di `policy.json`:

```
{
   "rules": [
       {
           "rulePriority": 1,
           "description": "Expire images older than 14 days",
           "selection": {
               "tagStatus": "untagged",
               "countType": "sinceImagePushed",
               "countUnit": "days",
               "countNumber": 14
           },
           "action": {
               "type": "expire"
           }
       }
   ]
}
```

Output:

```
{
   "registryId": "<aws_account_id>",
   "repositoryName": "project-a/amazon-ecs-sample",
   "lifecyclePolicyText": "{\"rules\":[{\"rulePriority\":1,\"description\":
\"Expire images older than 14 days\",\"selection\":{\"tagStatus\":\"untagged\",
\"countType\":\"sinceImagePushed\",\"countUnit\":\"days\",\"countNumber\":14},
\"action\":{\"type\":\"expire\"}}]}"
```

```
}
```

Per ulteriori informazioni, consulta Policy del ciclo di vita nella Guida per l'utente di Amazon ECR.

- Per i dettagli sull'API, consulta PutLifeCyclePolicy AWS CLICommand Reference.

Python

SDK per Python (Boto3)

> **Note**
>
> C'è altro su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
class ECRWrapper:
    def __init__(self, ecr_client: client):
        self.ecr_client = ecr_client

    @classmethod
    def from_client(cls) -> "ECRWrapper":
        """
        Creates a ECRWrapper instance with a default Amazon ECR client.

        :return: An instance of ECRWrapper initialized with the default Amazon
 ECR client.
        """
        ecr_client = boto3.client("ecr")
        return cls(ecr_client)


    def put_lifecycle_policy(self, repository_name: str, lifecycle_policy_text:
 str):
        """
        Puts a lifecycle policy for an ECR repository.

        :param repository_name: The name of the repository to put the lifecycle
 policy for.
        :param lifecycle_policy_text: The lifecycle policy text to put.
        """
```

```
        try:
            self.ecr_client.put_lifecycle_policy(
                repositoryName=repository_name,
                lifecyclePolicyText=lifecycle_policy_text,
            )
            print(f"Put lifecycle policy for repository {repository_name}.")
        except ClientError as err:
            logger.error(
                "Couldn't put lifecycle policy for repository %s. Here's why %s",
                repository_name,
                err.response["Error"]["Message"],
            )
            raise
```

Esempio che inserisce una policy relativa alla data di scadenza.

```
    def put_expiration_policy(self):
        """
        Puts an expiration policy on the ECR repository.
        """
        policy_json = {
            "rules": [
                {
                    "rulePriority": 1,
                    "description": "Expire images older than 14 days",
                    "selection": {
                        "tagStatus": "any",
                        "countType": "sinceImagePushed",
                        "countUnit": "days",
                        "countNumber": 14,
                    },
                    "action": {"type": "expire"},
                }
            ]
        }

        self.ecr_wrapper.put_lifecycle_policy(
            self.repository_name, json.dumps(policy_json)
        )
```

- Per i dettagli sull'API, consulta PutLifeCyclePolicy AWSSDK for Python (Boto3) API Reference.

Per un elenco completo delle guide per sviluppatori AWS SDK e degli esempi di codice, consulta. Utilizzo di Amazon ECR con un SDK AWS Questo argomento include anche informazioni su come iniziare e dettagli sulle versioni precedenti dell'SDK.

Utilizzo **`SetRepositoryPolicy`** con un AWS SDK o una CLI

Gli esempi di codice seguenti mostrano come utilizzare `SetRepositoryPolicy`.

Gli esempi di operazioni sono estratti di codice da programmi più grandi e devono essere eseguiti nel contesto. È possibile visualizzare questa operazione nel contesto nel seguente esempio di codice:

- Informazioni di base

CLI

AWS CLI

Come impostare la policy per un repository

L'esempio `set-repository-policy` seguente associa una policy di repository contenuta in un file al repository `cluster-autoscaler`.

```
aws ecr set-repository-policy \
    --repository-name cluster-autoscaler \
    --policy-text file://my-policy.json
```

Contenuto di `my-policy.json`:

```
{
    "Version":"2012-10-17",
    "Statement" : [
        {
            "Sid" : "allow public pull",
            "Effect" : "Allow",
            "Principal" : "*",
            "Action" : [
                "ecr:BatchCheckLayerAvailability",
                "ecr:BatchGetImage",
```

```
                "ecr:GetDownloadUrlForLayer"
            ]
        }
    ]
}
```

Output:

```
{
    "registryId": "012345678910",
    "repositoryName": "cluster-autoscaler",
    "policyText": "{\n  \"Version\" : \"2008-10-17\",\n  \"Statement\" :
 [ {\n    \"Sid\" : \"allow public pull\",\n    \"Effect\" : \"Allow\",\n
 \"Principal\" : \"*\",\n    \"Action\" : [ \"ecr:BatchCheckLayerAvailability\",
 \"ecr:BatchGetImage\", \"ecr:GetDownloadUrlForLayer\" ]\n  } ]\n}"
}
```

- Per i dettagli sull'API, consulta SetRepositoryPolicy AWS CLICommand Reference.

Java

SDK per Java 2.x

> **Note**
>
> C'è altro su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
    /**
     * Sets the repository policy for the specified ECR repository.
     *
     * @param repoName the name of the ECR repository.
     * @param iamRole  the IAM role to be granted access to the repository.
     * @throws RepositoryPolicyNotFoundException if the repository policy does
 not exist.
     * @throws EcrException                      if there is an unexpected error
 setting the repository policy.
     */
    public void setRepoPolicy(String repoName, String iamRole) {
        /*
```

```
         This example policy document grants the specified AWS principal the
permission to perform the
         `ecr:BatchGetImage` action. This policy is designed to allow the
specified principal
         to retrieve Docker images from the ECR repository.
        */
       String policyDocumentTemplate = """
           {
            "Version":"2012-10-17",
            "Statement" : [ {
              "Sid" : "new statement",
              "Effect" : "Allow",
              "Principal" : {
                "AWS" : "%s"
              },
              "Action" : "ecr:BatchGetImage"
            } ]
          }
           """;

       String policyDocument = String.format(policyDocumentTemplate, iamRole);
       SetRepositoryPolicyRequest setRepositoryPolicyRequest =
SetRepositoryPolicyRequest.builder()
           .repositoryName(repoName)
           .policyText(policyDocument)
           .build();

       CompletableFuture<SetRepositoryPolicyResponse> response =
getAsyncClient().setRepositoryPolicy(setRepositoryPolicyRequest);
       response.whenComplete((resp, ex) -> {
           if (resp != null) {
               System.out.println("Repository policy set successfully.");
           } else {
               Throwable cause = ex.getCause();
               if (cause instanceof RepositoryPolicyNotFoundException) {
                   throw (RepositoryPolicyNotFoundException) cause;
               } else if (cause instanceof EcrException) {
                   throw (EcrException) cause;
               } else {
                   String errorMessage = "Unexpected error: " +
cause.getMessage();
                   throw new RuntimeException(errorMessage, cause);
               }
           }
```

```
        });
        response.join();
    }
```

- Per i dettagli sull'API, SetRepositoryPolicyconsulta AWS SDK for Java 2.x API Reference.

Kotlin

SDK per Kotlin

> **Note**
>
> C'è altro su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
/**
 * Sets the repository policy for the specified ECR repository.
 *
 * @param repoName the name of the ECR repository.
 * @param iamRole the IAM role to be granted access to the repository.
 */
suspend fun setRepoPolicy(
    repoName: String?,
    iamRole: String?,
) {
    val policyDocumentTemplate =
        """
         {
          "Version":"2012-10-17",
          "Statement" : [ {
            "Sid" : "new statement",
            "Effect" : "Allow",
            "Principal" : {
              "AWS" : "$iamRole"
            },
            "Action" : "ecr:BatchGetImage"
          } ]
        }
```

```
            """.trimIndent()
        val setRepositoryPolicyRequest =
            SetRepositoryPolicyRequest {
                repositoryName = repoName
                policyText = policyDocumentTemplate
            }

        EcrClient.fromEnvironment { region = "us-east-1" }.use { ecrClient ->
            val response =
 ecrClient.setRepositoryPolicy(setRepositoryPolicyRequest)
            if (response != null) {
                println("Repository policy set successfully.")
            }
        }
    }
```

- Per i dettagli sull'API, SetRepositoryPolicyconsulta AWS SDK for Kotlin API reference.

Python

SDK per Python (Boto3)

> **Note**
>
> C'è altro su. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
class ECRWrapper:
    def __init__(self, ecr_client: client):
        self.ecr_client = ecr_client

    @classmethod
    def from_client(cls) -> "ECRWrapper":
        """
        Creates a ECRWrapper instance with a default Amazon ECR client.

        :return: An instance of ECRWrapper initialized with the default Amazon
 ECR client.
        """
        ecr_client = boto3.client("ecr")
```

```
        return cls(ecr_client)


    def set_repository_policy(self, repository_name: str, policy_text: str):
        """
        Sets the policy for an ECR repository.

        :param repository_name: The name of the repository to set the policy for.
        :param policy_text: The policy text to set.
        """
        try:
            self.ecr_client.set_repository_policy(
                repositoryName=repository_name, policyText=policy_text
            )
            print(f"Set repository policy for repository {repository_name}.")
        except ClientError as err:
            if err.response["Error"]["Code"] ==
"RepositoryPolicyNotFoundException":
                logger.error("Repository does not exist. %s.", repository_name)
                raise
            else:
                logger.error(
                    "Couldn't set repository policy for repository %s. Here's why
%s",
                    repository_name,
                    err.response["Error"]["Message"],
                )
                raise
```

Esempio che concede a un ruolo IAM l'accesso allo scaricamento.

```
    def grant_role_download_access(self, role_arn: str):
        """
        Grants the specified role access to download images from the ECR
repository.

        :param role_arn: The ARN of the role to grant access to.
        """
        policy_json = {
            "Version":"2012-10-17",
            "Statement": [
```

```
                {
                    "Sid": "AllowDownload",
                    "Effect": "Allow",
                    "Principal": {"AWS": role_arn},
                    "Action": ["ecr:BatchGetImage"],
                }
            ],
        }

        self.ecr_wrapper.set_repository_policy(
            self.repository_name, json.dumps(policy_json)
        )
```

- Per i dettagli sull'API, consulta SetRepositoryPolicy AWSSDK for Python (Boto3) API Reference.

Per un elenco completo delle guide per sviluppatori AWS SDK e degli esempi di codice, consulta. Utilizzo di Amazon ECR con un SDK AWS Questo argomento include anche informazioni su come iniziare e dettagli sulle versioni precedenti dell'SDK.

Utilizzo **`StartLifecyclePolicyPreview`** con un AWS SDK o una CLI

Gli esempi di codice seguenti mostrano come utilizzare `StartLifecyclePolicyPreview`.

Gli esempi di operazioni sono estratti di codice da programmi più grandi e devono essere eseguiti nel contesto. È possibile visualizzare questa operazione nel contesto nel seguente esempio di codice:

- Informazioni di base

CLI

AWS CLI

Come creare un'anteprima di una policy del ciclo di vita

L'esempio `start-lifecycle-policy-preview` seguente crea un'anteprima della policy del ciclo di vita definita in un file JSON per il repository specificato.

```
aws ecr start-lifecycle-policy-preview \
```

```
    --repository-name "project-a/amazon-ecs-sample" \
    --lifecycle-policy-text "file://policy.json"
```

Contenuto di `policy.json`:

```
{
   "rules": [
       {
           "rulePriority": 1,
           "description": "Expire images older than 14 days",
           "selection": {
               "tagStatus": "untagged",
               "countType": "sinceImagePushed",
               "countUnit": "days",
               "countNumber": 14
           },
           "action": {
               "type": "expire"
           }
       }
   ]
}
```

Output:

```
{
   "registryId": "012345678910",
   "repositoryName": "project-a/amazon-ecs-sample",
   "lifecyclePolicyText": "{\n    \"rules\": [\n        {\n
 \"rulePriority\": 1,\n            \"description\": \"Expire images older than 14
 days\",\n            \"selection\": {\n                \"tagStatus\": \"untagged
\",\n                \"countType\": \"sinceImagePushed\",\n
 \"countUnit\": \"days\",\n                \"countNumber\": 14\n            },\n
           \"action\": {\n                \"type\": \"expire\"\n            }\n
      }\n    ]\n}\n",
   "status": "IN_PROGRESS"
}
```

- Per i dettagli sull'API, consulta StartLifecyclePolicyPreview AWS CLICommand Reference.

Java

SDK per Java 2.x

> **Note**
>
> C'è altro su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
   /**
    * Verifies the existence of an image in an Amazon Elastic Container Registry
(Amazon ECR) repository asynchronously.
    *
    * @param repositoryName The name of the Amazon ECR repository.
    * @param imageTag       The tag of the image to verify.
    * @throws EcrException             if there is an error retrieving the image
information from Amazon ECR.
    * @throws CompletionException      if the asynchronous operation completes
exceptionally.
    */
   public void verifyImage(String repositoryName, String imageTag) {
       DescribeImagesRequest request = DescribeImagesRequest.builder()
           .repositoryName(repositoryName)
           .imageIds(ImageIdentifier.builder().imageTag(imageTag).build())
           .build();

       CompletableFuture<DescribeImagesResponse> response =
getAsyncClient().describeImages(request);
       response.whenComplete((describeImagesResponse, ex) -> {
           if (ex != null) {
               if (ex instanceof CompletionException) {
                   Throwable cause = ex.getCause();
                   if (cause instanceof EcrException) {
                       throw (EcrException) cause;
                   } else {
                       throw new RuntimeException("Unexpected error: " +
cause.getMessage(), cause);
                   }
               } else {
                   throw new RuntimeException("Unexpected error: " +
ex.getCause());
```

```
                }
            } else if (describeImagesResponse != null && !
describeImagesResponse.imageDetails().isEmpty()) {
                System.out.println("Image is present in the repository.");
            } else {
                System.out.println("Image is not present in the repository.");
            }
        });

        // Wait for the CompletableFuture to complete.
        response.join();
    }
```

- Per i dettagli sull'API, StartLifecyclePolicyPreviewconsulta AWS SDK for Java 2.x API Reference.

Kotlin

SDK per Kotlin

> **Note**
>
> C'è altro su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
    /**
     * Verifies the existence of an image in an Amazon Elastic Container Registry
(Amazon ECR) repository asynchronously.
     *
     * @param repositoryName The name of the Amazon ECR repository.
     * @param imageTag       The tag of the image to verify.
     */
    suspend fun verifyImage(
        repoName: String?,
        imageTagVal: String?,
    ) {
        require(!(repoName == null || repoName.isEmpty())) { "Repository name
cannot be null or empty" }
```

```
        require(!(imageTagVal == null || imageTagVal.isEmpty())) { "Image tag
 cannot be null or empty" }

        val imageId =
            ImageIdentifier {
                imageTag = imageTagVal
            }
        val request =
            DescribeImagesRequest {
                repositoryName = repoName
                imageIds = listOf(imageId)
            }

        EcrClient.fromEnvironment { region = "us-east-1" }.use { ecrClient ->
            val describeImagesResponse = ecrClient.describeImages(request)
            if (describeImagesResponse != null && !
describeImagesResponse.imageDetails?.isEmpty()!!) {
                println("Image is present in the repository.")
            } else {
                println("Image is not present in the repository.")
            }
        }
    }
```

- Per i dettagli sull'API, StartLifecyclePolicyPreviewconsulta AWS SDK for Kotlin API reference.

Per un elenco completo delle guide per sviluppatori AWS SDK e degli esempi di codice, consulta. Utilizzo di Amazon ECR con un SDK AWS Questo argomento include anche informazioni su come iniziare e dettagli sulle versioni precedenti dell'SDK.

Service Quotas di Amazon ECR.

Nella tabella seguente sono indicate le Service Quotas per Amazon Elastic Container Registry (Amazon ECR).

Nome	Predefinita	Adatta e	Description
Scansioni di immagini di base per 24 ore	Ogni regione supportata: 100.000	No	Il numero massimo di immagini che è possibile scansionare entro un periodo di 24 ore nell'account e nella regione correnti utilizzando la scansione di base. Questo limite include sia la scansione su richiesta che la scansione manuale.
Filtri per regola in una configurazione di replica	Ogni regione supportata: 100	No	Il numero massimo di filtri per regola in una configurazione di replica.
Immagine per repository	Ogni regione supportata: 100.000	Sì	Numero massimo di immagini per repository.
Parti di livello	Ogni regione supportata: 4.200	No	Il numero massimo di parti del livello. Questo è applicabile solo se utilizzi direttamente le operazioni API Amazon ECR per avviare caricamenti in più parti per le operazioni di invio delle immagini.

Nome	Predefinita	Adattabile	Description
Lunghezza della policy del ciclo di vita	Ogni regione supportata: 30.720	No	Numero massimo di caratteri in una policy del ciclo di vita.
Dimensione massima della parte del livello	Ogni regione supportata: 10	No	La dimensione massima (MiB) di una parte del livello. Questo è applicabile solo se utilizzi direttamente le operazioni API Amazon ECR per avviare caricamenti in più parti per le operazioni di invio delle immagini.
Dimensione massima del livello	Ogni regione supportata: 52.000	No	La dimensione massima (MiB) di un livello.
Dimensione minima della parte del livello	Ogni Regione supportata: 5	No	La dimensione minima (MiB) di una parte del livello. Questo è applicabile solo se utilizzi direttamente le operazioni API Amazon ECR per avviare caricamenti in più parti per le operazioni di invio delle immagini.
Regole di cache pull-through per registro	Ogni Regione supportata: 50	No	Il numero massimo di regole di cache pull-through.

Nome	Predefinita	Adatta e	Description
Frequenza delle richieste BatchCheckLayerAvailability	Ogni regione supportata: 1.000 al secondo	Sì	Il numero massimo di BatchCheckLayerAvailability richieste che è possibile effettuare al secondo nella regione corrente. Quando si esegue il push di un'immagine in un repository, ogni livello dell'immagine viene controllato per verificare se è stato già caricato. Se è stato caricato, il livello dell'immagine viene ignorato.
Frequenza delle BatchGetImage richieste	Ogni regione supportata: 2.000 al secondo	Sì	Il numero massimo di BatchGetImage richieste che è possibile effettuare al secondo nella regione corrente. Quando viene estratta un'immagine, l'BatchGetImage API viene chiamata una volta per recuperare il manifesto dell'immagine. Se richiedi un aumento della quota per questa API, controlla anche il tuo GetDownloadUrlForLayer utilizzo.

Nome	Predefinita	Adattabile	Description
Frequenza delle CompleteLayerUpload richieste	Ogni regione supportata: 100 al secondo	Sì	Il numero massimo di CompleteLayerUpload richieste che è possibile effettuare al secondo nella regione corrente. Quando viene inviata un'immagine, l'CompleteLayerUpload API viene chiamata una volta per ogni nuovo livello di immagine per verificare che il caricamento sia stato completato.
Frequenza delle richieste GetAuthorizationToken	Ogni regione supportata: 500 al secondo	Sì	Il numero massimo di GetAuthorizationToken richieste che è possibile effettuare al secondo nella regione corrente.

Nome	Predefinita	Adatta e	Description
Frequenza delle GetDownloadUrlForLayer richieste	Ogni regione supportata: 3.000 al secondo	Sì	Il numero massimo di GetDownloadUrlForLayer richieste che è possibile effettuare al secondo nella regione corrente. Quando un'immagine viene estratta, l'GetDownloadUrlForLayer API viene chiamata una volta per livello di immagine che non sia già memorizzato nella cache. Se richiedi un aumento della quota per questa API, controlla anche il tuo BatchGetImage utilizzo.

Nome	Predefinita	Adatta e	Description
Frequenza delle InitiateLayerUpload richieste	Ogni regione supportata: 100 al secondo	Sì	Il numero massimo di InitiateLayerUpload richieste che è possibile effettuare al secondo nella regione corrente. Quando viene inviata un'immagine, l' InitiateLayerUpload API viene chiamata una volta per livello di immagine che non sia già stato caricato. Il fatto che un livello di immagine sia stato caricato o meno è determinato dall'azione dell' BatchCheckLayerAvailability API.
Frequenza delle PutImage richieste	Ogni regione supportata: 10 al secondo	Sì	Il numero massimo di PutImage richieste che è possibile effettuare al secondo nella regione corrente. Quando viene inviata un'immagine e tutti i nuovi livelli di immagine sono stati caricati, l' PutImage API viene chiamata una sola volta per creare o aggiornare il manifesto dell'immagine e i tag associati all'immagine.

Nome	Predefinita	Adattabile	Description
Frequenza delle richieste UploadLayerPart	Ogni regione supportata: 500 al secondo	Sì	Il numero massimo di UploadLayerPart richieste che è possibile effettuare al secondo nella regione corrente. Quando viene inviata un'immagine, ogni nuovo livello di immagine viene caricato in parti e l' UploadLayerPart API viene chiamata una volta per ogni nuova parte del livello di immagine.
Frequenza delle scansioni delle immagini	Ogni regione supportata: 1	No	Il numero massimo di scansioni di immagini per immagine, al giorno.
Repository registrati	Ogni regione supportata: 100.000	Sì	Il numero massimo di repository che è possibile creare in questo account nella regione corrente.
Regole per la policy del ciclo di vita	Ogni Regione supportata: 50	No	Numero massimo di regole in una policy del ciclo di vita
Regole per la configurazione di repliche	Ogni regione supportata: 10	No	Il numero massimo di regole in una configurazione di replica.
Tag per immagine	Ogni regione supportata: 1.000	No	Il numero massimo di tag per immagine.

Nome	Predefinita	Adatta e	Description
Destinazioni univoche per tutte le regole in una configurazione di replica	Ogni regione supportata: 25	No	Il numero massimo di destinazioni univoche in tutte le regole di una configurazione di replica.

Gestione delle Service Quotas Amazon ECR in Console di gestione AWS

Amazon ECR si è integrato con Service Quotas, AWS un servizio che consente di visualizzare e gestire le quote da una posizione centrale. Per ulteriori informazioni, consulta Cos'è Service Quotas? nella Guida per l'utente di Service Quotas.

Service Quotas semplifica la ricerca del valore di tutte le quote di servizio Amazon ECR.

Per visualizzare le Service Quotas di Amazon ECR (Console di gestione AWS)

1. Apri la console Service Quotas all'indirizzo https://console.aws.amazon.com/servicequotas/.
2. Nel pannello di navigazione, scegliere servizi AWS .
3. Nell'elenco servizi AWS , cerca e seleziona Amazon Elastic Container Registry (Amazon ECR).

   Nell'elenco delle quote di servizio, puoi vedere il nome della quota di servizio, il valore applicato (se disponibile), la quota AWS predefinita e se il valore della quota è regolabile.
4. Per visualizzare ulteriori informazioni su una quota di servizio, ad esempio la descrizione, scegli il nome della quota.

Per richiedere un aumento delle quote, consultare Richiesta di aumento delle quote nella Guida dell'utente di Service Quotas.

Creazione di un CloudWatch allarme per monitorare le metriche di utilizzo dell'API

Amazon ECR fornisce parametri di CloudWatch utilizzo che corrispondono alle quote di AWS servizio per ciascuno dei APIs soggetti coinvolti nelle azioni di autenticazione del registro, image push e

image pull. Nella console Service Quotas puoi visualizzare l'utilizzo in un grafico e configurare gli allarmi che ti avvisano quando l'utilizzo si avvicina a una quota di servizio. Per ulteriori informazioni, consulta Parametri di utilizzo Amazon ECR.

Utilizza i seguenti passaggi per creare un CloudWatch allarme basato su uno dei parametri di utilizzo dell'API Amazon ECR.

Per creare un allarme basato sulle quote di utilizzo Amazon ECR (Console di gestione AWS)

1. Apri la console Service Quotas all'indirizzo https://console.aws.amazon.com/servicequotas/.
2. Nel pannello di navigazione, scegliere servizi AWS .
3. Nell'elenco servizi AWS , cerca e seleziona Amazon Elastic Container Registry (Amazon ECR).
4. Nell'elenco Service quotas (Quote di servizio) selezionare la quota di utilizzo Amazon ECR per cui si desidera creare un allarme.
5. Nella sezione Allarmi di Amazon CloudWatch Events, scegli Crea.
6. Per Soglia di allarme, scegli la percentuale del valore della quota applicata che desideri impostare come valore per l'allarme.
7. Per Nome allarme, immetti un nome per l'allarme e quindi scegli Crea.

Risoluzione dei problemi di Amazon ECR

Questo capitolo ti aiuta a trovare informazioni diagnostiche per Amazon ECR e fornisce passaggi per la risoluzione di problemi e messaggi di errore comuni.

Argomenti

Risoluzione dei problemi e dei comandi Docker durante l'utilizzo di Amazon ECR

In alcuni casi, l'esecuzione di un comando Docker su Amazon ECR potrebbe generare un messaggio di errore. Alcuni messaggi di errore comuni e possibili soluzioni sono spiegati di seguito.

Argomenti

I log Docker non contengono i messaggi di errore previsti

Per iniziare il debug di qualsiasi problema relativo a Docker, inizia attivando l'output di debug di Docker sul demone Docker in esecuzione sulle istanze host. Se utilizzi immagini estratte da Amazon ECR su istanze di container Amazon ECS, consulta Configuring verbose output dal daemon Docker nella Amazon Elastic Container Service Developer Guide.

Errore: "Filesystem Verification Failed" (Verifica del file system non riuscita) oppure "404: Image Not Found" (404: Immagine non trovata) durante l'estrazione di un'immagine da un repository Amazon ECR

Potresti ricevere l'errore `Filesystem verification failed` quando utilizzi il comando docker pull per estrarre un'immagine da un repository Amazon ECR con Docker 1.9 o versione successiva. Puoi visualizzare l'errore `404: Image not found` quando usi le versioni di Docker precedenti a 1.9.

Di seguito sono elencate alcune possibili cause e le relative spiegazioni.

Il disco locale è pieno

Se il disco locale sul quale esegui docker pull è pieno, il valore hash SHA-1 calcolato sul file locale può essere diverso dal valore calcolato da Amazon ECR. Controlla che sul tuo disco locale vi sia spazio libero sufficiente per conservare l'immagine Docker che stai estraendo. Puoi anche eliminare le immagini meno recenti per fare spazio a nuove immagini. Usa il comando docker images per vedere un elenco di tutte le immagini Docker scaricate in locale con le relative dimensioni.

Il client non è in grado di connettersi al repository remoto a causa di un errore di rete

Le chiamate a un repository Amazon ECR richiedono una connessione valida a Internet. Verifica le impostazioni di rete e che altri strumenti e applicazioni possano accedere alle risorse su Internet. Se stai docker pull eseguendo un' EC2 istanza Amazon in una sottorete privata, verifica che la sottorete abbia un percorso verso Internet. Utilizzare un server NAT (Network Address Translation) o un gateway NAT gestito.

Al momento, le chiamate a un repository Amazon ECR richiedono anche accesso di rete tramite il firewall aziendale a Amazon Simple Storage Service (Amazon S3). Se la tua organizzazione utilizza un firewall o un dispositivo NAT che consente gli endpoint del servizio, verifica che gli endpoint del servizio Amazon S3 per la regione corrente siano autorizzati.

Se utilizzi Docker dietro un proxy HTTP, puoi configurare Docker con le impostazioni proxy appropriate. Per ulteriori informazioni, consulta Proxy HTTP nella documentazione Docker.

Errore: "Filesystem Layer Verification Failed" (Verifica dei livelli del file system non riuscita) durante l'estrazione di immagini da Amazon ECR

Puoi visualizzare l'errore `image image-name not found` quando estrai le immagini utilizzando il comando docker pull. Se ispezioni i log di Docker, potresti trovare un errore simile al seguente:

```
filesystem layer verification failed for digest sha256:2b96f...
```

Questo errore indica che non è stato possibile scaricare uno o più livelli per la tua immagine. Di seguito sono elencate alcune possibili cause e le relative spiegazioni.

Stai utilizzando una versione di Docker meno recente

Questo errore può verificarsi in una piccola percentuale di casi quando utilizzi una versione di Docker precedente alla 1.10. Aggiorna il tuo client Docker alla versione 1.10 o versione successiva.

Il client ha riscontrato un errore di rete o del disco

Un problema di disco pieno o di rete può impedire a uno o più livelli di essere scaricati, come descritto precedentemente in merito al messaggio `Filesystem verification failed`. Segui le raccomandazioni suddette per assicurarti che il tuo file system non sia pieno e che tu abbia abilitato l'accesso ad Amazon S3 dalla tua rete.

Errore HTTP 403 o errore "no basic auth credentials" (Nessuna credenziale di autenticazione di base) quando effettui l'invio al repository

Ci sono volte in cui puoi visualizzare un errore `HTTP 403 (Forbidden)` oppure il messaggio di errore `no basic auth credentials` dal comando docker push o docker pull, anche sei ti sei autenticato correttamente in Docker utilizzando il comando aws ecr get-login-password. Di seguito sono elencate alcune cause note di questo problema:

Hai effettuato l'autenticazione per una regione diversa

Le richieste di autenticazione sono legate a regioni specifiche e non possono essere utilizzate in regioni diverse. Ad esempio, se ottieni un token di autorizzazione dagli Stati Uniti occidentali (Oregon), non puoi utilizzarlo per l'autenticazione dei tuoi repository negli Stati Uniti orientali (Virginia settentrionale). Per risolvere il problema, assicurati di aver recuperato un token di

autenticazione dalla stessa regione in cui esiste il repository. Per ulteriori informazioni, consulta the section called "Autenticazione del registro".

Hai autenticato il push in un repository per cui non disponi delle autorizzazioni

Non disponi delle autorizzazioni necessarie per eseguire il push al repository. Per ulteriori informazioni, consulta Politiche di archivio privato in Amazon ECR.

Il tuo token è scaduto

Il periodo di scadenza del token di autorizzazione predefinito per i token ottenuti tramite l'operazione `GetAuthorizationToken` è 12 ore.

Bug nella gestione credenziali `wincred`

Alcune versioni di Docker per Windows utilizzano un gestore di credenziali chiamato`wincred`, che non gestisce correttamente il comando di login Docker prodotto da aws ecr get-login-password (per ulteriori informazioni, consulta `CredsStorefail` with private repositories). Puoi eseguire il comando di login di Docker ottenuto, ma quando tenti di inviare o estrarre immagini, questi comandi non vanno a buon fine. Per risolvere questo bug, rimuovi lo schema `https://` dall'argomento del registro nel comando di login di Docker che risulta da aws ecr get-login-password. Di seguito viene mostrato un esempio di comando di login di Docker senza lo schema HTTPS.

```
docker login -u AWS -p <password> <aws_account_id>.dkr.ecr.<region>.amazonaws.com
```

Risoluzione dei problemi relativi ai messaggi di errore Amazon ECR

In alcuni casi, una chiamata API che hai avviato tramite la console Amazon ECR o AWS CLI esce con un messaggio di errore. Alcuni messaggi di errore comuni e possibili soluzioni sono spiegati di seguito.

HTTP 429: troppe richieste o ThrottleException

Potresti ricevere un `429: Too Many Requests` errore o un `ThrottleException` errore da una o più azioni o chiamate API di Amazon ECR. Ciò indica che stai chiamando un singolo endpoint in Amazon ECR più volte in un breve intervallo di tempo e che le tue richieste vengono limitate. Il throttling si verifica quando le chiamate a un singolo endpoint da uno stesso utente superano una certa soglia in un periodo di tempo.

A ogni operazione API in Amazon ECR è associata una limitazione di velocità. Ad esempio, la limitazione per l'operazione `GetAuthorizationToken` è di 20 transazioni al secondo (TPS), con un aumento consentito fino a 200 TPS. In ogni regione, ogni account riceve un bucket che può ospitare fino a 200 crediti `GetAuthorizationToken`. Questi crediti vengono riforniti a una velocità di 20 al secondo. Se il bucket ospita 200 crediti, puoi ottenere 200 transazioni API `GetAuthorizationToken` al secondo per un secondo, quindi mantenere 20 transazioni al secondo a tempo indeterminato. Per ulteriori informazioni sui limiti di tariffa per Amazon ECR APIs, consultaService Quotas di Amazon ECR..

Per gestire gli errori di throttling, implementa una funzione di nuovo tentativo con backoff incrementale nel tuo codice. Per ulteriori informazioni, consulta Retry behavior nella AWS SDKs and Tools Reference Guide. Un'altra opzione è richiedere un aumento del limite di velocità, cosa che puoi fare utilizzando la console Service Quotas. Per ulteriori informazioni, consulta Gestione delle Service Quotas Amazon ECR in Console di gestione AWS.

HTTP 403: "User [arn] is not authorized to perform [operation]" (HTTP 403: l'utente [arn] non è autorizzato a eseguire [operazione])

Puoi visualizzare il seguente errore quando tenti di eseguire un'operazione con Amazon ECR:

```
$ aws ecr get-login-password
A client error (AccessDeniedException) occurred when calling the GetAuthorizationToken
 operation:
    User: arn:aws:iam::account-number:user/username is not authorized to perform:
    ecr:GetAuthorizationToken on resource: *
```

Ciò indica che il tuo utente non ha ricevuto le autorizzazioni per utilizzare Amazon ECR oppure che quelle autorizzazioni non sono configurare correttamente. In particolare, se esegui delle operazioni su un repository Amazon ECR, verifica che l'utente abbia ricevuto le autorizzazioni per accedere a quel repository. Per ulteriori informazioni sulla creazione e le verifica delle autorizzazioni per Amazon ECR, consulta Identity and Access Management per Amazon Elastic Container Registry.

Errore HTTP 404: "Repository Does Not Exist" (HTTP 404: il repository non esiste)

Se specifichi un repository del Docker Hub che non esiste, Docker Hub lo crea automaticamente. Con Amazon ECR, i nuovi repository devono essere creati esplicitamente prima di poter essere utilizzati. Ciò impedisce la creazione accidentale di nuovi repository (ad esempio, per errori di digitazione) e

assicura inoltre che venga assegnata una policy d'accesso adeguata ai nuovi repository. Per ulteriori informazioni sulla creazione dei repository, consulta Repository Amazon ECR privati.

Errore: impossibile eseguire un accesso interattivo da un dispositivo non TTY

Se ricevi l'errore `Cannot perform an interactive login from a non TTY device`, i seguenti passaggi per la risoluzione dei problemi dovrebbero aiutarti.

- Verifica di utilizzare la AWS CLI versione 2 e di non avere una versione in conflitto della AWS CLI versione 1 sul tuo sistema. Per ulteriori informazioni, consulta Installare o aggiornare la versione più recente della AWS CLI.
- Verifica di aver configurato il tuo AWS CLI con credenziali valide. Per ulteriori informazioni, consulta Installare o aggiornare la versione più recente della AWS CLI.
- Verifica che la sintassi del AWS CLI comando sia corretta.

Utilizzo di Podman con Amazon ECR

L'utilizzo Podman con Amazon ECR consente alle organizzazioni di sfruttare la sicurezza e la semplicità di sfruttando al Podman contempo la scalabilità e l'affidabilità di Amazon ECR per la gestione delle immagini dei container. Seguendo i passaggi e i comandi descritti, sviluppatori e amministratori possono semplificare i flussi di lavoro dei container, migliorare la sicurezza e ottimizzare l'utilizzo delle risorse. Poiché la containerizzazione continua a guadagnare slancio, l'utilizzo di Podman Amazon ECR fornisce una soluzione robusta e flessibile per la gestione e la distribuzione di applicazioni containerizzate.

Utilizzo di Podman per l'autenticazione con Amazon ECR

Prima di interagire con Amazon ECR utilizzandoPodman, è richiesta l'autenticazione. Ciò può essere ottenuto eseguendo il `aws ecr get-login-password` comando per recuperare un token di autenticazione e quindi utilizzando tale token con il `podman login` comando per l'autenticazione con Amazon ECR.

```
aws ecr get-login-password --region <region> | podman login --username AWS --password-
stdin <aws_account_id>.dkr.ecr.<region>.amazonaws.com
```

Utilizzo dell'helper di credenziali Amazon ECR con Podman

Amazon ECR fornisce un supporto per le credenziali Docker che funziona con Podman. L'helper per le credenziali semplifica l'archiviazione e l'utilizzo delle credenziali Docker durante l'invio e l'estrazione di immagini in Amazon ECR. Per i passaggi di installazione e configurazione, consulta Amazon ECR Docker Credential Helper.

Important

Podman supporta solo parzialmente le specifiche. docker-creds-helper Podman supporta la `credHelpers` parola chiave nella configurazione Docker ma non la supporta. `credsStore` Per utilizzare l'helper di credenziali Amazon ECR con Podman, configura il file di configurazione Docker con il formato: `credHelpers`

```
{
    "credHelpers": {
```

```
        "public.ecr.aws": "ecr-login",
        "<aws_account_id>.dkr.ecr.<region>.amazonaws.com": "ecr-login"
    }
}
```

La seguente `credsStore` configurazione non è supportata da Podman:

```
{
    "credsStore": "ecr-login"
}
```

> **Note**
>
> L'helper di credenziali Amazon ECR Docker non supporta attualmente l'autenticazione a più fattori (MFA).

Estrarre immagini da Amazon ECR con Podman

Dopo una corretta autenticazione, le immagini dei container possono essere estratte da Amazon ECR utilizzando il `podman pull` comando con l'URI completo del repository Amazon ECR.

```
podman pull aws_account_id.dkr.ecr.region.amazonaws.com/repository_name:tag
```

Contenitori in esecuzione per Amazon ECR con Podman

Una volta estratta l'immagine desiderata, è possibile creare un'istanza di un contenitore utilizzando il comando. `podman run`

```
podman run -d aws_account_id.dkr.ecr.region.amazonaws.com/repository_name:tag
```

Invio di immagini ad Amazon ECR con Podman

Per inviare un'immagine locale ad Amazon ECR, l'immagine deve prima essere etichettata con l'URI del repository Amazon ECR`podman tag`, quindi il `podman push` comando può essere utilizzato per caricare l'immagine su Amazon ECR.

```
podman
 tag local_image:tag aws_account_id.dkr.ecr.region.amazonaws.com/repository_name:tag
podman push aws_account_id.dkr.ecr.region.amazonaws.com/repository_name:tag
```

Cronologia dei documenti

Nella tabella seguente vengono descritte le modifiche importanti apportate alla documentazione dall'ultima versione di Amazon ECR. Inoltre, aggiorniamo frequentemente la documentazione per tener conto del feedback inviatoci.

Modifica	Descrizione	Data
Firma gestita da ECR	Amazon ECR ora supporta la firma gestita delle immagini dei container per migliorare il livello di sicurezza ed eliminare il sovraccarico operativo legato alla configurazione della firma. La firma delle immagini dei container consente di verificare che le immagini provengano da fonti attendibili. Per ulteriori informazi oni, consulta Firma gestita.	21 novembre 2025
IPv6 supporto per AWS PrivateLink (endpoint VPC)	È stato aggiunto il supporto per la connettività dual-stack (IPv4 e IPv6) per gli endpoint VPC di Amazon ECR con tecnologia. AWS PrivateLink Ora puoi creare endpoint VPC dual-stack che gestiscono il traffico sia su indirizzi IP privati che su IPv4 indirizzi IP privati. IPv6 Per ulteriori informazioni, consulta Endpoint VPC con interfaccia Amazon ECR ()AWS PrivateLink.	21 novembre 2025
Funzione ECR Archive	Amazon ECR ha aggiunto il supporto per l'archivi azione delle immagini per la conservazione a lungo termine. Per ulteriori informazioni, consulta Archiviaz ione di un'immagine in Amazon ECR.	19 novembre 2025
Aggiornato per includere il supporto per i modelli di esclusione dell'immu tabilità dei tag di immagine	Amazon ECR ha aggiornato le funzionalità di tagging delle immagini per includere modelli di esclusion e dell'immutabilità dei tag di immagine durante la creazione e l'aggiornamento dei repository. Ora puoi specificare tag che possono essere aggiornati anche quando l'immutabilità dei tag è abilitata su un repositor y definendo modelli jolly (come `latest` «,, `dev-*`) per escludere tag specifici dalle regole di immutabil	23 luglio 2025

Modifica	Descrizione	Data
	ità mantenendo `v*.beta` l'immutabilità per tutti gli altri tag. Per ulteriori informazioni, consulta Creazione di un repository privato Amazon ECR per archiviare immagini.	
Scansione avanzata delle immagini aggiornat a per fornire informazioni sull'utilizzo delle immagini	Amazon ECR ha aggiornato le funzionalità di scansione avanzata delle immagini per includere la visibilità sul modo in cui le immagini vengono utilizzate su Amazon EKS e Amazon ECS. Per ulteriori informazi oni, consulta Scansione delle immagini per individua re le vulnerabilità del sistema operativo e dei pacchetti del linguaggio di programmazione in Amazon ECR.	16 giugno 2025
IPv6 supporto	È stato aggiunto il supporto per effettuare richieste ai registri Amazon ECR utilizzando endpoint sia IPv4 -only che dual-stack (and). IPv4 IPv6 Per ulteriori informazioni, consulta Effettuare richieste ai registri Amazon ECR.	30 aprile 2025
Aggiunto il supporto del registro privato Amazon ECR per l'estrazione della cache	Amazon ECR ha aggiunto il supporto per la creazione di regole pull through cache per il registro privato Amazon ECR. Per ulteriori informazioni, consultar e Sincronizzazione di un registro upstream con un registro privato Amazon ECR e Ruolo collegato ai servizi Amazon ECR per la cache pull-through.	12 marzo 2025
È stato aggiunto il supporto per l'imposta zione dell'ambito delle politiche di registro	Amazon ECR ha aggiunto il supporto per la configura zione dell'ambito delle politiche di registro per il registro privato. Per ulteriori informazioni, consulta Autorizzazioni di registro privato in Amazon ECR e Registro privato Amazon ECR.	23 dicembre 2024
Amazon EC2 Container RegistryPullOnly — Nuova politica	Amazon ECR ha aggiunto una nuova policy che concede autorizzazioni pull-only ad Amazon ECR.	10 ottobre 2024

Modifica	Descrizione	Data
Le operazioni tramite proxy client Docker/OCI negli eventi ora puntano a CloudTrail `ecr.amazonaws.com`	Il valore viene `ecr.amazonaws.com` sostituito `AWS Internal` nei campi User Agent (`userAgent`) e Source IP Address (`sourceIPAddress`) per gli eventi associati agli endpoint Client. CloudTrail Docker/OCI Per alcuni esempi, consulta Esempio: operazione di estrazione immagine e Esempio: operazione di invio immagine.	1 luglio 2024
È stata aggiunta la descrizione del nuovo ruolo collegato al servizio Amazon ECR per i modelli di creazione di repository.	Amazon ECR utilizza un ruolo collegato al servizio denominato AWSServiceRoleForECRTemplateche autorizza Amazon ECR a eseguire azioni per tuo conto per completare le azioni dei modelli di creazione di repository. Per ulteriori informazioni, consulta Ruolo collegato al servizio Amazon ECR per i modelli di creazione di repository.	20 giugno 2024
Aggiunto il ruolo collegato al `ECRTemplateServiceRolePolicy` servizio.	È stato aggiunto il ruolo collegato al `ECRTemplateServiceRolePolicy` servizio. Per ulteriori informazioni, consulta `ECRTemplateServiceRolePolicy`	20 giugno 2024
Aggiunta la replica tra regioni e più account nelle regioni della Cina.	Amazon ECR ha aggiunto il supporto alla regione Cina per filtrare i repository replicati. Per ulteriori informazioni, consulta Private image replication in Amazon ECR	15 maggio 2024
È stato aggiunto GitLab il registro dei contenitori per consultare le regole della cache	Amazon ECR ha aggiunto il supporto per la creazione di regole pull through cache per il registro dei GitLab container. Per ulteriori informazioni, consulta Sincronizzazione di un registro upstream con un registro privato Amazon ECR.	8 maggio 2024

Modifica	Descrizione	Data
Aggiornamento della policy del ciclo di vita di Amazon ECR per aggiungere il supporto per l'uso di caratteri jolly	Amazon ECR ha aggiunto il supporto per i caratteri jolly in una policy del ciclo di vita mediante l'uso del parametro `tagPatternList` in una regola della policy del ciclo di vita. Per ulteriori informazioni, consulta Automatizza la pulizia delle immagini utilizzando le politiche del ciclo di vita in Amazon ECR.	18 dicembre 2023
Modelli di creazione dei repository di Amazon ECR	Amazon ECR ha aggiunto il supporto per i modelli di creazione del repository. Per ulteriori informazioni, consulta Modelli per controllare gli archivi creati durante un'azione di pull through cache, di creazione in modalità push o di replica.	15 novembre 2023
Aggiunta la cache pull-through di Amazon ECR, supportata per registri upstream autenticati	Amazon ECR ha aggiunto il supporto per l'utilizzo di registri upstream che richiedono l'autenticazione per le regole di cache pull-through. Per ulteriori informazioni, consulta Sincronizzazione di un registro upstream con un registro privato Amazon ECR.	15 novembre 2023
AWSECRPullThroughCache_ServiceRolePolicy— Aggiornamento a una politica esistente	Amazon ECR ha aggiunto nuove autorizzazioni alla policy `AWSECRPullThroughCache_ServiceRolePolicy`. Queste autorizzazioni consentono ad Amazon ECR di recuperare i contenuti crittografati di un segreto di Secrets Manager. Ciò è necessario quando utilizzi una regola di cache pull-through per memorizzare nella cache le immagini da un registro upstream che richiede l'autenticazione.	15 novembre 2023
Firma delle immagini Amazon ECR	Amazon ECR e supporto AWS Signer aggiunto per la creazione e l'invio di firme di immagini di container utilizzando il client Notary. Per ulteriori informazioni, consulta Firma immagini in Amazon ECR.	6 giugno 2023

Modifica	Descrizione	Data
Registro del container Kubernetes aggiunto alle regole della cache pull-through	Amazon ECR ha aggiunto il supporto per la creazione di regole della cache pull-through per il registro dei container Kubernetes. Per ulteriori informazioni, consulta Sincronizzazione di un registro upstream con un registro privato Amazon ECR.	1 giugno 2023
Supporto della durata della scansione avanzata di Amazon ECR	Amazon Inspector ha aggiunto il supporto per l'impostazione della durata di monitoraggio dei repository quando è abilitata la scansione avanzata. Per ulteriori informazioni, consulta Modifica della durata di scansione migliorata per le immagini in Amazon Inspector.	28 giugno 2022
Amazon ECR invia i parametri del pull count del repository ad Amazon CloudWatch	Amazon ECR invia ad Amazon i parametri del pull count dei repository. CloudWatch Per ulteriori informazioni, consulta Parametri del repository Amazon ECR.	6 gennaio 2022
Supporto di replica espanso	Amazon ECR ha aggiunto il supporto per filtrare i repository replicati. Per ulteriori informazioni, consulta Private image replication in Amazon ECR.	21 settembre 2021
AWS politiche gestite per Amazon ECR	Amazon ECR ha aggiunto la documentazione delle politiche AWS gestite. Per ulteriori informazioni, consulta AWS politiche gestite per Amazon Elastic Container Registry.	24 giugno 2021
Replica tra regioni e tra account	Amazon ECR ha aggiunto il supporto per la configurazione delle impostazioni di replica per il tuo registro privato. Per ulteriori informazioni, consulta Impostazioni del registro privato in Amazon ECR.	8 dicembre 2020

Modifica	Descrizione	Data
Supporto artefatti OCI	Amazon ECR ha aggiunto il supporto per inviare ed estrarre gli artefatti Open Container Initiative (OCI). Un nuovo parametro `artifactMediaType` è stato aggiunto alla riposta API `DescribeImages` per indicare il tipo di artefatto. Per ulteriori informazioni, consulta Trasferimento di un grafico Helm a un repository privato Amazon ECR.	24 agosto 2020
Crittografia dei dati a riposo	Amazon ECR ha aggiunto il supporto per la configura zione della crittografia per i tuoi repository utilizzan do la crittografia lato server con le chiavi gestite dal cliente archiviate in AWS Key Management Service (AWS KMS). Per ulteriori informazioni, consulta Crittografia dei dati a riposo.	29 luglio 2020
Immagini multi-arc hitettura	Amazon ECR ha aggiunto il supporto per la creazione e l'invio di elenchi di manifesti di Docker che vengono utilizzati per le immagini multi-architettura. Per ulteriori informazioni, consulta Trasferimento di un'immagine multiarchitettura a un repository privato Amazon ECR.	28 aprile 2020
Parametri di utilizzo Amazon ECR	Amazon ECR ha aggiunto parametri di CloudWatc h utilizzo che forniscono visibilità sull'utilizzo delle risorse del tuo account. Hai anche la possibilità di creare CloudWatch allarmi sia dalla console Service Quotas che da quella di Service Quotas per ricevere avvisi quando l'utilizzo si avvicina alla quota di servizio applicata. CloudWatch Per ulteriori informazioni, consulta Parametri di utilizzo Amazon ECR.	28 febbraio 2020

Modifica	Descrizione	Data
Aggiornate Service Quotas di Amazon ECR.	Aggiornate le Service Quotas di Amazon ECR per includere le quote per le API. Per ulteriori informazioni, consulta Service Quotas di Amazon ECR..	19 febbraio 2020
Aggiunto il comando `get-login-password`	Aggiunto il supporto per get-login-password, che fornisce un metodo semplice e sicuro per recuperare un token di autorizzazione. Per ulteriori informazioni, consulta Utilizzo di un token di autorizzazione.	4 febbraio 2020
Scansione delle immagini	Aggiunto il supporto per la scansione delle immagini, che aiuta a identificare le vulnerabilità del software nelle immagini di container. Amazon ECR utilizza il database Common Vulnerabilities and Exposures (CVEs) del progetto open source CoreOS Clair e fornisce un elenco dei risultati della scansione. Per ulteriori informazioni, consulta Scansiona le immagini per individuare le vulnerabilità del software in Amazon ECR.	24 ottobre 2019
Policy degli endpoint VPC	Aggiunto il supporto per l'impostazione di una policy IAM sugli endpoint VPC dell'interfaccia Amazon ECR. Per ulteriori informazioni, consulta Creazione di una policy di endpoint per l'endpoint VPC di Amazon ECR.	26 settembre 2019
Mutabilità dei tag immagine	È stato aggiunto il supporto per la configurazione di un repository in modo che sia immutabile, per impedire che i tag immagine vengano sovrascritti. Per ulteriori informazioni, consulta Impedire la sovrascrittura dei tag di immagine in Amazon ECR.	25 luglio 2019

Modifica	Descrizione	Data
Endpoint VPC di interfaccia (AWS PrivateLink)	È stato aggiunto il supporto per la configurazione degli endpoint VPC dell'interfaccia con tecnologia. AWS PrivateLink Ciò consente di creare una connessione privata tra il VPC e Amazon ECR senza richiedere l'accesso tramite Internet, un'istanza NAT, una connessione VPN o Direct Connect. Per ulteriori informazioni, consulta Endpoint VPC con interfaccia Amazon ECR ()AWS PrivateLink.	25 gennaio 2019
Aggiunta di tag alle risorse	In Amazon ECR è stato implementato il supporto per aggiungere i tag di metadati ai repository. Per ulteriori informazioni, consulta Taggare un repository privato in Amazon ECR.	18 dicembre 2018
Modifica del nome Amazon ECR	Amazon Elastic Container Registry viene rinominato (in precedenza Amazon EC2 Container Registry).	21 novembre 2017
Policy del ciclo di vita	Le policy del ciclo di vita di Amazon ECR consentono di specificare la gestione del ciclo di vita delle immagini in un repository. Per ulteriori informazioni, consulta Automatizza la pulizia delle immagini utilizzando le politiche del ciclo di vita in Amazon ECR.	11 ottobre 2017
Supporto Amazon ECR per manifesto immagine Docker 2, schema 2	Amazon ECR supporta ora Docker Image Manifest V2 Schema 2 (utilizzato con la versione Docker 1.10 e più recente). Per ulteriori informazioni, consulta Supporto del formato manifesto dell'immagine del contenitore in Amazon ECR.	27 gennaio 2017

Modifica	Descrizione	Data
Disponibilità generale di Amazon ECR	Amazon Elastic Container Registry (Amazon ECR) è un servizio di registro Docker AWS gestito sicuro, scalabile e affidabile.	21 dicembre 2015

Le traduzioni sono generate tramite traduzione automatica. In caso di conflitto tra il contenuto di una traduzione e la versione originale in Inglese, quest'ultima prevarrà.