aws

Automated Install Guide

Wickr Enterprise

Wickr Enterprise: Automated Install Guide

Services or capabilities described in AWS documentation might vary by Region. To see the differences applicable to the AWS European Sovereign Cloud Region, see the AWS European Sovereign Cloud User Guide.

Table of Contents

What is Wickr Enterprise?

Wickr Enterprise is an end-to-end encrypted, self-hosted service that helps organizations and government agencies to communicate securely through one-to-one and group messaging, voice and video calling, file sharing, and screen sharing. Customers can use Wickr Enterprise to overcome data retention obligations associated with consumer-grade messaging apps, and safely facilitate collaboration. Advanced security and administrative controls help organizations meet legal and regulatory requirements, and build custom solutions for data security challenges.

Information can be logged to a private, customer-controlled data store for retention and auditing purposes. Customers have comprehensive administrative control over data, which includes setting permissions, configuring ephemeral messaging options, and defining security groups. Administrators can also securely automate workflows with Wickr bots. Wickr Enterprise integrates with additional services such as Active Directory and single sign-on (SSO) with OpenID Connect (OIDC). To begin configuring Wickr Enterprise, see Getting started with Wickr Enterprise.

> **Note**
>
> If you do not already have the Wickr Enterprise deployment package, see Contact Us for business inquiries.

Getting started with Wickr Enterprise

Topics

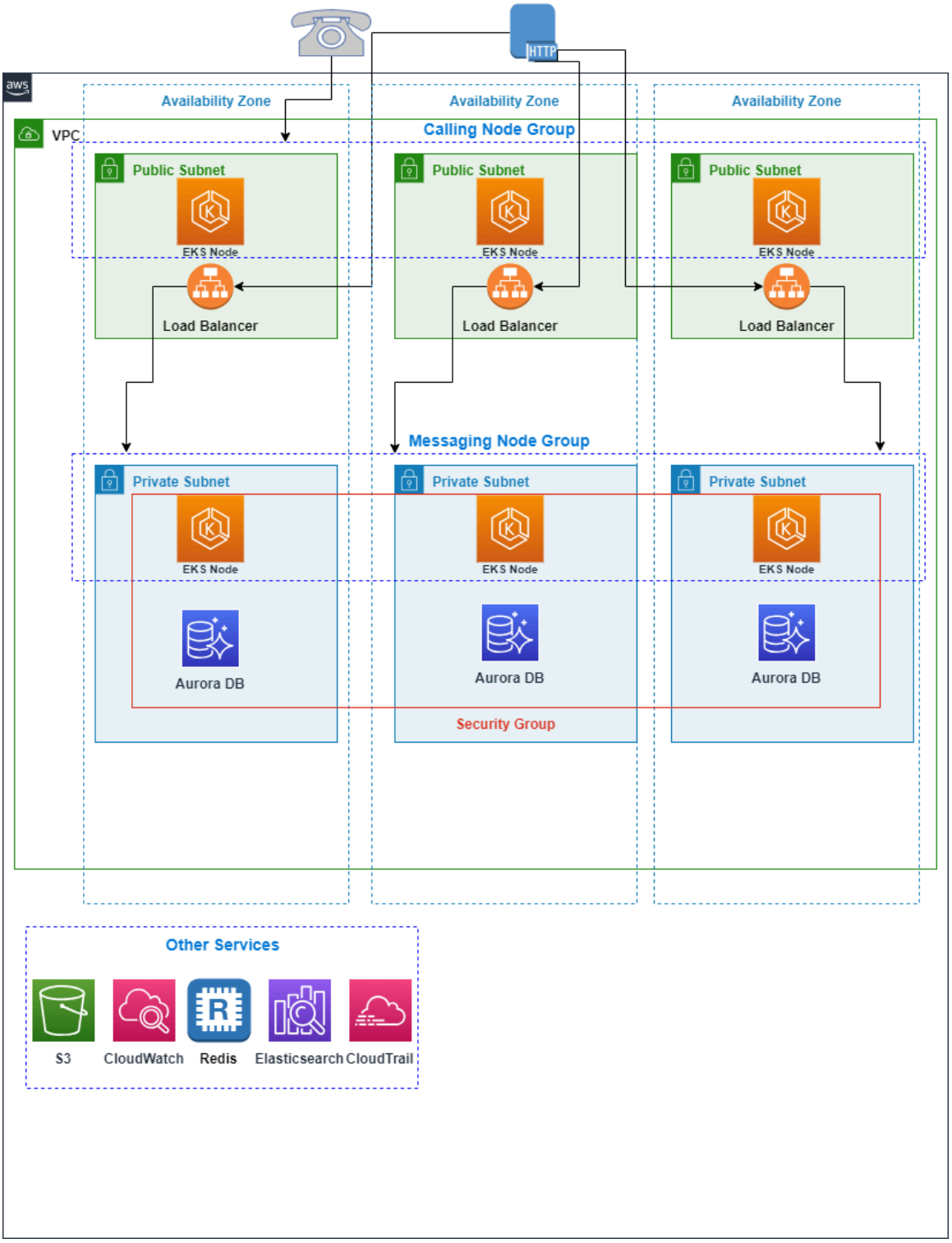
Requirements

Before you start, verify that the following requirements are met:

- Download Node.js 16+
- AWS CLI configured with credentials for your account.

  These will be sourced either from your config file at `~/.aws/config` or using the `AWS_` environment variables.
- Install kubectl. For more information, see Installing or updating kubectl in the *Amazon EKSUser Guide*.
- Install kots CLI. For more information, see Installing the kots CLI.
- Ports to allowlist: 443/TCP for HTTPS and TCP Calling traffic; 16384-19999/UDP for UDP Calling traffic; TCP/8443

Architecture



Install dependencies

You can add all dependencies to the default package with the following command:

```
npm install
```

Configure

AWS Cloud Development Kit (AWS CDK) uses context values to control the configuration of the application. Wickr Enterprise uses CDK context values to provide control over settings such as the domain name of your Wickr Enterprise installation or the number of days to retain RDS backups. For more information, see Runtime context in the *AWS Cloud Development Kit (AWS CDK) Developer Guide*.

There are multiple ways to set context values, but we recommend editing the values in `cdk.context.json` to fit your particular use case. Only context values that begin with `wickr/` are related to the Wickr Enterprise deployment; the rest are CDK-specific context values. To keep the same settings the next time you make an update through the CDK, save this file.

At a minimum, you must set `wickr/licensePath`, `wickr/domainName`, and either `wickr/acm:certificateArn` or `wickr/route53:hostedZoneId` and `wickr/route53:hostedZoneName`.

With a public hosted zone

If you have a Route 53 public hosted zone in your AWS account, we recommend using the following settings to configure your CDK context:

- `wickr/domainName` - The domain name to use for this Wickr Enterprise deployment. If you use a Route 53 public hosted zone, DNS records and ACM certificates for this domain name will be automatically created.
- `wickr/route53:hostedZoneName` - Route 53 hosted zone name in which to create DNS records.
- `wickr/route53:hostedZoneId` - Route 53 hosted zone ID in which to create DNS records.

This method creates an ACM certificate on your behalf, along with the DNS records pointing your domain name to the load balancer in front of your Wickr Enterprise deployment.

Without a public hosted zone

If you don't have a Route 53 public hosted zone in your account, an ACM certificate must be created manually and imported into the CDK using the `wickr/acm:certificateArn` context value.

- `wickr/domainName` - The domain name to use for this Wickr Enterprise deployment. If you use a Route 53 public hosted zone, DNS records and ACM certificates for this domain name will be automatically created.
- `wickr/acm:certificateArn` - The ARN of an ACM certificate to use on the load balancer. This value must be supplied if a Route 53 public hosted zone isn't available in your account.

Importing a certificate to ACM

You can import an externally obtained certificate with the following command:

```
aws acm import-certificate \
   --certificate fileb://path/to/cert.pem \
   --private-key fileb://path/to/key.pem \
   --certificate-chain fileb://path/to/chain.pem
```

The output will be the Certificate ARN, which should be used for the value of the `wickr/acm:certificateArn` context setting. It's important that the uploaded certificate is valid for the `wickr/domainName`, or HTTPS connections will be unable to validate. For more information, see Importing a certificate in the *AWS Certificate Manager User Guide*.

Create DNS records

Because there is no public hosted zone available, DNS records must be created manually after the deployment is finished to point to the load balancer in front of your Wickr Enterprise deployment.

Deploying into an existing VPC

If you require the use of an existing VPC you can use one. However, the VPC must be configured to meet the specifications necessary for EKS. For more information, see View Amazon EKS networking requirements for VPC and subnets in the *Amazon EKS User Guide*, and ensure that the VPC to be used meets these requirements.

Additionally, it is highly recommended to ensure you have VPC endpoints for the following services:

- CLOUDWATCH

- CLOUDWATCH_LOGS
- EC2
- EC2_MESSAGES
- ECR
- ECR_DOCKER
- ELASTIC_LOAD_BALANCING
- KMS
- SECRETS_MANAGER
- SSM
- SSM_MESSAGES

To deploy resources into an existing VPC, set the following context values:

- `wickr/vpc:id` - The VPC ID to deploy resources into (e.g. `vpc-412beef`).
- `wickr/vpc:cidr` - The IPv4 CIDR of the VPC (e.g. `172.16.0.0/16`).
- `wickr/vpc:publicSubnetIds` - A comma-separated list of public subnets in the VPC. The Application Load Balancer and calling EKS worker nodes will be deployed in these subnets (e.g. `subnet-6ce9941,subnet-1785141,subnet-2e7dc10`).
- `wickr/vpc:privateSubnetIds` - A comma-separated list of private subnets in the VPC. The EKS worker nodes and bastion server will be deployed in these subnets (e.g. `subnet-f448ea8,subnet-3eb0da4,subnet-ad800b5`).
- `wickr/vpc:isolatedSubnetIds` - A comma-separated list of isolated subnets in the VPC. The RDS database will be deployed in these subnets (e.g. `subnet-d1273a2,subnet-33504ae,subnet-0bc83ac`).
- `wickr/vpc:availabilityZones` - A comma-separated list of availability zones for the subnets in the VPC (e.g. `us-east-1a,us-east-1b,us-east-1c`).

For more information on interface VPC endpoints, see Access an AWS service using an interface VPC endpoint.

Other settings

For more information, see Context values.

Bootstrap

If this is your first time using CDK on this particular AWS account and Region, you must first *bootstrap* the account to begin using CDK.

```
npx cdk bootstrap
```

Deploy

This process will take around 45 minutes.

```
npx cdk deploy --all --require-approval=never
```

After it's complete, the infrastructure has been created and you can begin installing Wickr Enterprise.

Create DNS records

This step isn't required if you used a public hosted zone when configuring the CDK.

The output from the deployment process will include a value `WickrAlb.AlbDnsName`, which is the DNS name of the load balancer. The output will look like:

```
WickrAlb.AlbDnsName = Wickr-Alb-1Q5IBPJR4ZVZR-409483305.us-west-2.elb.amazonaws.com
```

In this case, the DNS name is `Wickr-Alb-1Q5IBPJR4ZVZR-409483305.us-west-2.elb.amazonaws.com`. That is the value that should be used when creating a CNAME or A/AAAA (ALIAS) record for your domain name.

If you don't have the output from the deployment, run the following command to display the load balancer DNS name:

```
aws cloudformation describe-stacks --stack-name WickrAlb \
   --query 'Stacks[0].Outputs[?OutputKey==`AlbDnsName`].OutputValue' \
   --output text
```

Generate KOTS Config

> **Warning**
>
> This file contains sensitive information about your installation. Do not share or save it publicly.

The Wickr Enterprise installer requires a number of configuration values about the infrastructure in order to install successfully. You can use a helper script to generate the configurations values.

```
./bin/generate-kots-config.ts > wickr-config.json
```

If you imported an external certificate into ACM in the first step, pass the `--ca-file` flag to this script, for example:

```
./bin/generate-kots-config.ts --ca-file path/to/chain.pem > wickr-config.json
```

If you receive an error saying the stack does not exist, set the `AWS_REGION` environment variable (`export AWS_REGION=us-west-2`) to your selected Region and try again. Or, if you set the context value `wickr/stackSuffix`, pass the suffix with the `--stack-suffix` flag.

Connecting to the Kubernetes cluster

The Amazon EKS API is accessible only through a bastion host that is created as a part of the deployment. As a result, all `kubectl` commands must either be run on the bastion host itself or be proxied through the bastion host.

Proxying connections through the bastion

The first time you're connecting to the cluster, you must update your local kubeconfig file using the `aws eks update-kubeconfig` command, and then set the `proxy-url` in your configuration. Then, each time you want to connect to the cluster, you start an SSM session with the bastion host to port forward to the proxy for API access.

One-time setup

There is an output value on the `WickrEks` CloudFormation stack with a name that begins with `WickrEnterpriseConfigCommand`. The value contains the full command needed to generate the kubectl configuration for your cluster. This output can be viewed with the following command:

```
aws cloudformation describe-stacks --stack-name WickrEks \
--query 'Stacks[0].Outputs[?starts_with(OutputKey,
 `WickrEnterpriseConfigCommand`)].OutputValue' \
--output text
```

This should output a command that begins with `aws eks update-kubeconfig`. Run this command.

Next, the Kubernetes configuration must be modified to proxy requests through the bastion host. This can be done using the following commands:

```
CLUSTER_ARN=$(aws cloudformation describe-stacks --stack-name WickrEks --query
 'Stacks[0].Outputs[?OutputKey==`WickrEnterpriseEksClusterArn`].OutputValue' --output
 text)
kubectl config set "clusters.${CLUSTER_ARN}.proxy-url" http://localhost:8888
```

If it worked correctly, you will see output like `'Property "clusters.arn:aws:eks:us-west-2:012345678912:cluster/WickrEnterprise5B8BF472-1234a41c4ec48b7b615c6789d93dcce.proxy-url" set.'`

Port forward to the bastion

To connect to the Amazon EKS cluster, you must start an SSM session to port forward requests to the proxy running on your bastion host. The command to do this is provided as the output `BastionSSMProxyEKSCommand` on the `WickrEks` stack. Run the following command to view the output value:

```
aws cloudformation describe-stacks --stack-name WickrEks \
--query 'Stacks[0].Outputs[?OutputKey==`BastionSSMProxyEKSCommand`].OutputValue' \
--output text
```

The command that it outputs will begin with `aws ssm start-session`. Run this command to start a local proxy running on port 8888 through which you can connect to the Amazon EKS cluster. If the port forward worked correctly, the output should say 'Waiting for connections...'. Keep this process running the entire time that you need to access the Amazon EKS cluster.

If everything is set up correctly, you will be able to run `kubectl get nodes` in another terminal to list the worker nodes in the Amazon EKS cluster:

```
kubectl get nodes
   NAME                             STATUS   ROLES    AGE     VERSION
   ip-10-0-111-216.ec2.internal   Ready     none   3d      v1.26.4-eks-0a21954
   ip-10-0-180-1.ec2.internal     Ready     none   2d23h   v1.26.4-eks-0a21954
   ip-10-0-200-102.ec2.internal   Ready     none   3d      v1.26.4-eks-0a21954
```

Installing Wickr Enterprise

After your connection to the Kubernetes cluster has been made, you can begin installing Wickr Enterprise using the `kubectl kots` plugin. You will need your KOTS License file (a `.yaml` file provided by Wickr) and your Config Values file, which were saved to the file `wickr-config.json` in the Generate KOTS Config section. For more information about Generate KOTS Config, see Generate KOTS Config.

Installing Wickr Enterprise manually

The following command will begin the installation of Wickr Enterprise:

```
kubectl kots install wickr-enterprise-ha \
    --license-file ./license.yaml \
    --config-values ./wickr-config.json  \
    --namespace wickr \
    --skip-preflights
```

You will be prompted to enter a password for the KOTS Admin Console. Save this password because you will need it to upgrade or change the configuration of your Wickr Enterprise installation in the future.

When the installation is complete, `kubectl kots` will open up a local port (usually `http://localhost:8080`), which provides access to the KOTS Admin Console. You can change or monitor the status of your Wickr Enterprise installation on this site, or begin setting up Wickr by visiting the domain name that you configured for your installation in your browser.

Installing Wickr Enterprise with Lambda

During the CDK deployment, a Lambda is created and invoked to complete the Wickr Enterprise installation on your behalf automatically. To invoke it manually, open the AWS console and find `WickrLambda-func*` lambda function, under test tab, select `test`, the input is irrelevant.

Post installation

There are two web consoles available for managing your Wickr Enterprise installation: the KOTS Admin Console and the Wickr Admin Console.

> **Note**
>
> Make any changes needed to reflect your organization's backup and logging policies (Amazon S3 settings, Elastic Load Balancing access logs, Amazon Virtual Private Cloud flow logs).

KOTS Admin Console

This interface is used for managing the deployed version of Wickr Enterprise. You can see the status of the installation, modify configurations, or perform upgrades. The KOTS Admin Console is accessible only through a Kubernetes port forward, which can be opened using the following command:

```
kubectl kots --namespace wickr admin-console
```

> **Note**
>
> You must first set up your bastion connection as described in the port forward to the bastion section. For more information on port forward to the bastion, see Proxying connections through the bastion.

When the port forward is successfully configured, the previous command will output the following:

```
- Press Ctrl+C to exit
- Go to http://localhost:8800 to access the Admin Console
```

Use the provided URL to access the KOTS Admin Console. The password to log in is the one you chose when running `kubectl kots install` during the installation. If you need to reset your password, see Resetting the KOTS Admin Console Password.

Wickr Admin Console

This interface is used for configuring your Wickr Enterprise installation to set up networks, users, and federation. It's accessible over HTTPS at the DNS name that you configured to point to your Load Balancer. If DNS was configured automatically with a public hosted zone, the domain name is the value of the `wickr/domainName` context value.

The default username is `admin`, with the password `Password123`. You will be required to change this password on first log in.

Context values

Context values are key-value pairs that can be associated with an app, stack, or construct. They can be supplied to your app from a file (usually either `cdk.json` or `cdk.context.json` in your project directory) or on the command line. CDK uses context values to control the configuration of the application. Wickr Enterprise uses CDK context values to provide control over settings such as the domain name of your Wickr Enterprise installation or the number of days to retain RDS backups.

There are multiple ways to set context values, but we recommend editing the values in `cdk.context.json` to fit your particular use case. Only context values that begin with `wickr/` are related to the Wickr Enterprise deployment.

Name	Description	Default
`wickr/licensePath`	The path to your KOTS license (a `.yaml` file provided by Wickr).	null
`wickr/domainName`	The domain name to use for this Wickr Enterprise deployment. If using a Route 53 public hosted zone, DNS records and ACM certificates for this domain name will be automatically created.	null
`wickr/route53:hostedZoneId`	Route 53 hosted zone ID in which to create DNS records.	null
`wickr/route53:hostedZoneName`	Route 53 hosted zone Name in which to create DNS records.	null
`wickr/acm:certificateArn`	ARN of an ACM certificate to use on the Load Balancer. This value must be supplied if a Route 53 public hosted	null

Name	Description	Default
	zone is not available in your account.	
wickr/caPath	Certificate path, only required when using self-signed certificates.	null
wickr/vpc:id	The ID of the VPC to deploy resources into. Only required when deploying into an existing VPC. If unset, a new VPC will be created.	null
wickr/vpc:cidr	IPv4 CIDR to associate with the created VPC. If deploying into an existing VPC, set this to the CIDR of the existing VPC.	172.16.0.0/16
wickr/vpc:availabilityZones	Comma-separated list of availability zones. Only required when deploying into an existing VPC.	null
wickr/vpc:publicSubnetIds	Comma-separated list of public subnet IDs. Only required when deploying into an existing VPC.	null
wickr/vpc:privateSubnetIds	Comma-separated list of private subnet IDs. Only required when deploying into an existing VPC.	null

Name	Description	Default
wickr/vpc:isolatedSubnetIds	Comma-separated list of isolated subnet IDs for the RDS database. Only required when deploying into an existing VPC.	null
wickr/rds:deletionProtection	Enable deletion protection on RDS instances.	true
wickr/rds:removalPolicy	Removal policy for RDS instances 'snapshot', 'destroy', or 'retain.'	snapshot
wickr/rds:readerCount	Number of reader instances to create in the RDS cluster.	1
wickr/rds:instanceType	Instance type to use for RDS instances.	r6g.xlarge
wickr/rds:backupRetentionDays	Number of days to retain backups.	7
wickr/eks:namespace	Default namespace for Wickr services in EKS.	wickr
wickr/eks:defaultCapacity	Number of EKS worker nodes for Messaging infrastructure.	3
wickr/eks:defaultCapacityCalling	Number of EKS worker nodes for Calling infrastructure.	2
wickr/eks:instanceTypes	Comma-separated list of instance types to use for Messaging EKS worker nodes.	m5.xlarge

Name	Description	Default
wickr/eks:instanceTypesCalling	Comma-separated list of instance types to use for Calling EKS worker nodes.	c5n.large
wickr/eks:enableAutoscaler	Toggles enabling the Cluster Autoscaler functionality for EKS.	true
wickr/s3:expireAfterDays	Number of days after which file uploads will be removed from the S3 bucket.	1095
wickr/eks:clusterVersion	Cluster versions, including Kubernetes version, kubectlLayer version, albController version, nodeGroupRelease version and more.	1.27
wickr/stackSuffix	A suffix to apply to CloudFormation stack names.	"
wickr/autoDeployWickr	Auto deploy the Wickr application with lambda.	true

Destroying resources

To delete everything created by this AWS CDK application, you must delete the `WickrRds` stack before all other stacks.

In order for the Amazon RDS resources to properly delete, deletion protection must be disabled, and the removal policy must be set to either `snapshot` or `destroy`. If these are not the current settings, modify the `wickr/rds:deletionProtection` and `wickr/rds:removalPolicy` values in your AWS CDK context and redeploy the Amazon RDS stack by running `npx cdk deploy -e WickrRds`.

Once the deletion protection and removal policy are properly set, run `cdk destroy` for the `WickrRds` stack:

```
npx cdk destroy WickrRds
```

When the `WickrRds` stack has finished destroying, the remaining CloudFormation stacks can be destroyed with the following command:

```
npx cdk destroy --all
```

Troubleshooting

Deleting the Wickr namespace

If you need to delete the `wickr` namespace to start over, it's important that you first back up any Service Accounts that were created by CDK within that namespace. These Service Accounts allow Wickr services to communicate with AWS APIs through IAM roles. Without them, tasks like file uploads through Amazon Simple Storage Service (Amazon S3) will no longer work.

Use the following command to back up the Service Accounts and delete and recreate the `wickr` namespace and the appropriate Service Accounts:

```
kubectl -n wickr get sa fileproxy -o yaml > fileproxy-sa.yaml && \
  kubectl delete ns wickr && \
  kubectl create ns wickr && \
  kubectl apply -f fileproxy-sa.yaml
```

Resetting the KOTS Admin Console password

You can reset your KOTS Admin Console password with the following command:

```
kubectl kots -n wickr reset-password
```

When you change this password you may also want to update the `wickr/kots` Secrets Manager secret as well, although it will generally not be used again by any automation.

Issues connecting to EKS cluster with bastion

If your connection to the EKS cluster through the bastion seems slow or is timing out occasionally, you may see the following error when running `kubectl` commands:

net/http: request canceled while waiting for connection (Client.Timeout exceeded while awaiting headers)

This issue can often be remedied by logging into the bastion host via SSM (see the `BastionSSMCommand` on the WickrEks stack) and restarting the `tinyproxy` service:

```
sudo systemctl restart tinyproxy
```

Custom installation

In the **Custom installation** section, you will learn how to install Wickr Enterprise.

Topics

Requirements

Before you start to install Wickr Enterprise, verify that the following requirements are met.

Hardware requirements

Wickr Enterprise requires a Kubernetes cluster to operate. It is possible to operate on a single node with Low Resource Mode enabled, but this is not recommended for general production use. In a Production deployment we recommend a minimum of three messaging worker nodes as well as a minimum of two calling worker nodes.

A worker node should have the following minimum specifications.

- 2 to 4 CPU cores
- 8 GB of Ram
- 200 GB of disk space

Minimum Hardware Requirements

A single worker node cluster running in Low Resource Mode requires a minimum of 3000m CPU and 5846Mi Ram. This does not include the kube-system pods.

Resource Requirements By Pod

Pod Name	Owner	CPU	Memory
admin-api	Wickr	100m	256Mi
directory	Wickr	100m	128Mi
expirer	Wickr	100m	128Mi
fileproxy	Wickr	100m	256Mi
oidc	Wickr	100m	128Mi
opensearch	Wickr	500m	100Mi
orville	Wickr	50m	128Mi
orville-redis	Wickr	50m	128Mi
push-device	Wickr	100m	128Mi
rabbitmq	Wickr	50m	256Mi
react	Wickr	100m	64Mi
receipts	Wickr	250m	128Mi
redis	Wickr	50m	128Mi
server-api	Wickr	250m	256Mi

Pod Name	Owner	CPU	Memory
switchboard	Wickr	250m	512Mi
kotsadm	KOTS	50m	50Mi
kotsadm-minio	KOTS	100m	512Mi
kotsadm-rqlite	KOTS	200m	1Gi
minio-operator	Internal S3	200m	256Mi
minio-tenant	Internal S3	100m	256Mi
mysql-primary	Internal MySQL	100m	512Mi
mysql-secondary	Internal MySQL	100m	512Mi

Storage Requirements

Wickr Enterprise requires a default StorageClass to utilize when creating Persistent Volume Claims. When deploying in an air-gapped environment or on premises you may need to configure one for your cluster. One available option is Longhorn. Recommended disk space requirements will vary based on the use of the Internal S3 option and the Internal Mysql option and the amount of space you wish to have available for file uploads.

- Internal Image caching: ~60 Gi
- RabbitMQ: 24 Gi Default / 8 Gi in Low Resource Mode
- Redis: 24 Gi Default / 8 Gi in Low Resource Mode
- OpenSearch: 24 Gi Default / 8 Gi in Low Resource Mode
- Internal Mysql: 80 Gi Default / 20Gi in Low Resource Mode
- Internal S3: 160 Gi Default / 2Gi in Low Resource Mode
- KOTS Minio: 4 Gi
- KOTS Rqlite: 1 Gi

Minimum Storage Size

- 377 Gi Default with Internal S3 and Internal Mysql
- 111 Gi in Low Resource Mode

Kubernetes Version Requirements

Wickr Enterprise relies on Replicated KOTS. Replicated, a commercial software distribution platform, provides a list of the currently supported versions of Kubernetes. For more information, see Kubernetes Version Compatibility.

Software requirements

Wickr Enterprise requires a Kubernetes cluster and KOTS to operate. Please refer to the KOTS documentation for supported OS and Kubernetes versions. For more information, see Minimum System Requirements.

Developer Host System

Operating System — The commands in this documentation are designed to work on Linux, MacOS, or Windows with WSL (Windows Subsystem for Linux) installed.

Internal Stateful Services

Wickr Enterprise can provide internal services for both MySQL database and S3 compatible storage however for general production use it is recommended you provide these services external from the Kubernetes cluster.

- MySQL 5.7 Database
  - Amazon RDS MySQL 5.7 or MySQL 5.7 database (External)
  - Mysql Bitnami Helm Chart (Internal)
  - File Storage
    - Amazon S3 or S3 compatible storage provider (External)
    - Minio Operator Helm Chart (Internal)

Network requirements

Wickr Enterprise requires a FQDN, SSL certificates, and specific open TCP and UDP ports.

- FQDN: A domain or sub-domain to be used by the Wickr Enterprise deployment.

- SSL certificate: An SSL certificate key pair signed by a public CA or a self signed certificate key pair. The certificate must list the FQDN in the Common Name and also as a SAN DNS entry. The certificate must also enable the serverAuth extendedKeyUsage extension.
- Online installs will need egress access to Replicated and third party resources. Replicated maintains a list of their IP addresses. For more information, see Replicated IP Addresses. Replicated also maintains a list of third party resources needed. For more information, see Firewall Openings for Online Installations.
- Air-gapped installs require access to a private container registry.

Messaging Nodes

Messaging nodes do not require a public IPV4 address and should be located in a private subnet. Message traffic will enter the cluster through the LoadBalancer or Ingress.

Calling Nodes

Calling nodes require a public IPV4 address so they must be in a public subnet. Call media is transferred via UDP by default. When TCP calling is enabled the TCP Proxy will accept connections on TCP 443 and will proxy them to the Orville service.

- TCP : 443 Calling TCP Proxy
- UDP : 16384-16484 Audio/Video Streams

Installation and Configuration access

Access to the KOTS Admin Console for installation and configuration is done through a Kubernetes port forward.

```
kubectl kots admin-console -n wickr
```

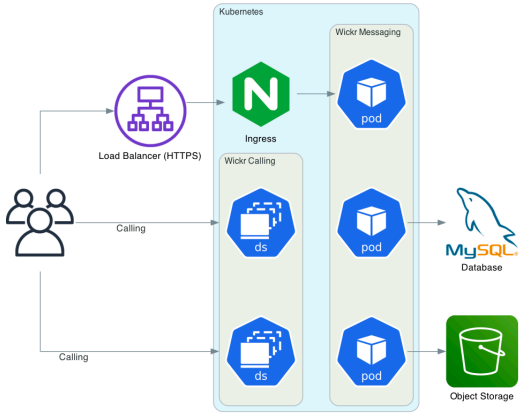
License Requirements

Installation will require a .yaml format license file, this will be provided to you by Wickr Support.

Architecture

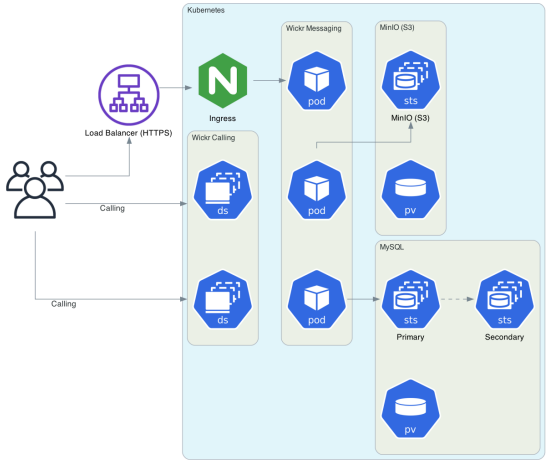
Recommended Production Architecture

The diagram below shows Wickr Enterprise configured as recommended for production, with both MySQL and Object Storage services situated outside of the Kubernetes cluster.



Internal or Test Architecture

The diagram below displays the configuration of Wickr Enterprise, utilizing the internal MYSQL and Object Storage services. Although it may satisfy the specific needs of certain deployments, it is not recommended for general production use.



Installation

1. Install kubectl and kots CLI.
2. Connect to the Kubernetes cluster.
3. Obtain Wickr Enterprise license file from Wickr Support.
4. Install Wickr Enterprise using the following command.

```
kubectl kots install wickr-enterprise-ha \
   --license-file ./license.yaml \
   --namespace wickr
```

> **Note**
>
> license.yaml represents your provided license file.

After initial installation the KOTS Admin Console will provide cluster level management and configuration options.

KOTS Admin Console

This interface is used for managing the deployed version of Wickr Enterprise. You can see the status of the installation, modify configurations, or perform upgrades of Wickr Enterprise. The KOTS Admin Console is accessible only through a Kubernetes port forward, which can be opened using the following command:

```
kubectl kots admin-console -n wickr
```

Ingress settings

Ingress Controller

Wickr Enterprise supports four ingress controller types:

- LoadBalancer (Default)
  - The loadbalancer object may require explicit configuration in fully on-prem installations, even though it is often provided by cloud providers.

  - Deploys the ingress controller (ingress-nginx) service with the LoadBalancer service type. This requires that the Kubernetes cluster is running on a platform which supports external load balancers.
- Existing ALB
  - Attaches the ingress controller to an existing ALB.
  - You will need to supply the existing Application Load Balancer Target Group ARN.
- Existing NLB
  - Attaches the ingress controller to an existing NLB.
  - You will need to supply the existing Network Load Balancer Target Group ARN.
- NodePort
  - The ingress controller (ingress-nginx) will be configured to use the NodePort service type, which opens a port on all nodes in the Kubernetes cluster and forwards traffic to the ingress. Client traffic can then be directed to these nodes either through DNS or some external load balancer.
  - You can choose a port range from 1-65535, or a random port from 30000-32767 will be used.
- Ingress
  - Bring your own ingress controller. This configuration will accept an ingress class name which the services will then use in their Ingress manifests. This implies that the ingress controller has some external connectivity already configured via some other load balancing mechanism.
  - Currently only the ingress-nginx controller is supported.

Wildcard Hostname

By default, Ingress routes will be defined with a host value of `*`. Disable this setting to use the defined hostname for the Wickr Enterprise Server. Wildcard Hostname is required for IP based hostnames.

Database settings

Wickr Enterprise requires a MySQL 8.0 database. If you are on MySQL 5.7, see Upgrade to MySQL 8.0 to upgrade. We recommend using a database which is external to your Kubernetes cluster, such as Amazon RDS, but you also have the option of deploying an **Internal** MySQL database inside of the Kubernetes cluster as a part of the installation.

External Database Settings

- **Hostname**: Hostname or IP address of the database server.
- **Reader Hostname**: Hostname or IP address of a read-only endpoint for the database server (if available).
- **Port**: The port which MySQL will be accessed on.
- **Database Name**: The name of the database created on the server.
- **Username**: The user which has permissions to access the database.
- **Password**: The password for that user.
- **CA Certificate**: A PEM certificate for connecting to the database over TLS.

> **Note**
>
> Ensure that your MySQL installation is using the default latin1 character set with latin1_swedish_ci collation. This can be accomplished by verifying that your MySQL server is started with the following flags:
> `"--character-set-server latin1", "--collation-server latin1_swedish_ci"`

Internal Database Settings

The internal database type will deploy two StatefulSets into your cluster for a MySQL primary and secondary with binary replication. The secondary does not receive any traffic and is available only for disaster recovery and backups.

Storage Size: Size (in gibibytes) of the Persistent Volumes for the database pods.

Increasing MySQL Storage Size

> **Note**
>
> The volume type of your StorageClass must support volume expansion in order to increase the storage size. For more information, see Volume expansion.

The MySQL services used in Wickr Enterprise are deployed as StatefulSet resources in Kubernetes. StatefulSets make many properties of the resource immutable, including the Persistent Volume Claim templates. As a workaround for the immutability of StatefulSets, the following actions must be performed to increase the size of volumes used by MySQL.

1. Edit the Persistent Volume Claims for `data-mysql-primary-0` and `data-mysql-secondary-0`.

   1. `kubectl -n wickr edit pvc data-mysql-primary-0.` Set `spec.resources.requests.storage` to the desired storage size.
   2. `kubectl -n wickr edit pvc data-mysql-secondary-0.` Set `spec.resources.requests.storage` to the desired storage size.

2. Delete the existing StatefulSets, but leave the Pods by passing the `--cascade=orphan` flag.

   `kubectl -n wickr delete statefulset --cascade=orphan mysql-primary mysql-secondary`.

3. In the KOTS UI, update the Storage Size setting to match the value you set in Step 1. Save and deploy this configuration.
4. Restart the StatefulSets to expand the volumes and bring the MySQL services back online.

   `kubectl -n wickr rollout restart statefulset mysql-primary mysql-secondary`.

Upgrade to MySQL 8.0

External Database (RDS)

To take Wickr Backend offline, complete the following steps.

1. Find the namespace of ingress `kubectl get deployments --all-namespaces`

   In the example below, the namespace is Wickr and replicas is 3.

```
NAMESPACE     NAME                       READY   UP-TO-DATE   AVAILABLE   AGE
...
wickr         ingress-nginx-controller   3/3     3            3           43h
...
```

2. Scale down ingress `kubectl scale deployment/ingress-nginx-controller --replicas=0 -n wickr`
3. Take a snapshot to backup DB. For more information, see Managing manual backups in the *Amazon Relational Database Service User Guide*.
4. Upgrade the engine version to MySQL 8.0.x (MySQL 8.4 is not supported). For more information, see Upgrading a DB instance engine version in the *Amazon Relational Database Service User Guide*.

To bring Wickr Backend online, scale back ingress `kubectl scale deployment/ingress-nginx-controller --replicas=3 -n wickr`

Internal Database

For more information, see Backup and Restore MySQL.

S3 File storage

Wickr Enterprise requires an S3 compatible storage service. We recommend using an S3 service which is external to your Kubernetes cluster, such as Amazon S3, but you also have the option of deploying an **Internal** S3 service inside of the Kubernetes cluster as a part of the installation.

External S3 Settings

- **Bucket Name**: The name of the S3 bucket where file uploads will be stored.
- **Region**: The AWS region of the S3 bucket.
- **Endpoint**: Set the endpoint that Wickr will use to interact with the S3 API. Defaults to the region's S3 service endpoint.
- **Fileproxy Service Account Name**: Amazon S3 only. The name of an existing Kubernetes Service Account to use for authenticating to S3 using IAM roles for Service Accounts.
- **External S3 Access Key**: This is your existing S3 Access key.
- **External S3 Secret Key**: This is your existing S3 Secret key.

Internal S3 Settings

The internal S3 type will deploy a default of 4 MinIO server pods that each contain 4 Persistent Volume Claims. The default configuration utilizes MinIO's Erasure Coding to increase fault tolerance.

- **Internal S3 server count**: The number of MinIO server pods to create, the default is 4 for a fault tolerant deployment. This value can be set as low as 1 for a development/test deployment.
- **Internal S3 volume count**: The number of MinIO volumes to create in each MinIO server pod, the default is 4 for a fault tolerant deployment. This value can be set as low as 1 for a development/test deployment.
- **Internal S3 volume size**: The size in GB of the MinIO volumes created in the MinIO server pods, the default is 10GB.
- A default Internal S3 deployment will use 4 servers with 4 PVCs. Each PVC is 10 Gi yielding 160 Gi Raw storage with 120 Gi Erasure Coded storage available to users.
- Minio Erasure Coding calculator is available. For more information, see Erasure Code Calculator.

Persistent volume claim settings

Wickr Enterprise requires Persistent Volume Claims to store stateful data. This setting allows you to specify the name of the name of the Storage Class you would like to use. If left blank Wickr will attempt to use the default Storage Class. Changing the Storage Class after Wickr has been deployed is not supported.

A default StorageClass for Persistent Volume Claims is often provided by cloud providers, however in fully onprem installations it may require explicit configuration using a third party service such as Longhorn.

TLS certificate settings

Upload a PEM certificate and private key for terminating TLS. The Subject Alternative Name on the certificate must match the hostname configured in the settings of your Wickr Enterprise deployment.

For the certificate chain field, concatenate any intermediate certificates (if required) with the root CA certificate **before uploading**.

Let's Encrypt

Select this option to automatically generate a certificate using Let's Encrypt. Certificates are issued using the HTTP-01 challenge through the cert-manager operator.

The HTTP-01 challenge requires that the desired DNS name resolves to the ingress point for your cluster (usually a Load Balancer), and traffic to TCP port 80 is open to the public. These certificates are short-lived and will be renewed regularly. It is necessary to keep port 80 open to allow the certificates to renew automatically.

> **Note**
>
> This section refers explicitly to the certificate used by the Wickr Enterprise application itself.

Pinned Certificate

Wickr Enterprise requires certificate pinning when using self-signed certificates or certificates not trusted by client devices. If the certificate presented by your Load Balancer is self-signed or is signed by a different CA than the Wickr Enterprise installation, upload the CA certificate here to have clients pin to it instead.

In most situations, this setting is not required.

Certificate Providers

If you plan to purchase a certificate for use with Wickr Enterprise see below for a list of providers who's certificates are known to function correctly by default. If a provider is listed below their certificates have been validated with the software explicitly.

- Digicert
- RapidSSL

Generating a self-signed certificate

If you would like to create your own self signed certificate for use with Wickr Enterprise, the example command below contains all required flags for generation.

```
openssl req -x509 -newkey rsa:4096 -sha256 -days 365 -nodes -keyout $YOUR_DOMAIN.key -
out $YOUR_DOMAIN.crt -subj "/CN=$YOUR_DOMAIN" -addext "subjectAltName=DNS:$YOUR_DOMAIN"
 -addext "extendedKeyUsage = serverAuth"
```

If you would like to create an IP based self signed certificate, use the following command instead. In order to use the IP based certificate, ensure that the Wildcard Hostname field is enabled under Ingress settings. For more information, see Ingress settings.

```
openssl req -x509 -newkey rsa:4096 -sha256 -days 365 -nodes -keyout $YOUR_DOMAIN.key -
out $YOUR_DOMAIN.crt -subj "/CN=$YOUR_DOMAIN" -addext "subjectAltName=IP:$YOUR_DOMAIN"
 -addext "extendedKeyUsage = serverAuth"
```

Note

Replace $YOUR_DOMAIN in the example with the domain name or IP address you intend to use.

Calling settings

- **Require Calling Nodes**: When this setting is enabled, Wickr's calling services are only deployed on Kubernetes Nodes with the label `role=calling`. Disable this setting to deploy Calling and Messaging services on the same nodes, or for single node deployments.

  You will generally also want to disable the calling TCP Proxy when this setting is disabled, because the TCP Proxy service runs on port 443.
- **Enable TCP Proxy**: This setting controls whether or not the service for TCP fallback mode on calls is deployed. Disable this setting if you have other services running on 443/tcp or do not require TCP fallback mode for calls. This must be enabled for deployments planning to use Wickr Open Access.
- **Automatically discover server public IP addresses**: When this setting is enabled, the calling services will discover their public IP address by making HTTPS requests to https://ipv4.icanhazip.com/ and https://ipv6.icanhazip.com/. When disabled, you must enable the "Use host primary IP address for Calling traffic" or "Hostname override" setting otherwise the calling services will fail to start.
- **Use host primary IP address for Calling traffic**: Use the primary IP address of the Kubernetes nodes for calling services. This implies that all Wickr clients are able to connect to your Kubernetes nodes on the primary IP address of the node, as presented in `status.hostIP` from the Downward API.
- **Hostname override**: Provide a hostname or IP address to return as the connectivity point for Calling services. This setting should only be used when running a single calling server, because

the same value is returned for all replicas of the service. When a hostname override is set and the “use host primary IP address” setting is enabled, the host primary IP address setting takes precedence.

- **Calling Host Network Enabled**: By default, calling pods use the nodes' host network for connectivity. Disable this to expose a NodePort service for calling traffic. If calling ingress is enabled, make sure that an appropriate service is configured to allow ingress traffic. This must be disabled for STIG compliance.

Calling ingress settings

Wickr supports a calling ingress setting, allowing a client to connect to any calling node within the cluster and have the call route to the correct calling server. Wickr supports four calling ingress types:

- LoadBalancer (default)
  - The LoadBalancer will be provisioned by the cloud provider (fully on-premise installations will require additional configuration). After the LoadBalancer has been provisioned, the KOTS config must be updated again to provide the load balancer's hostname or IP addresses.
- NodePort
  - Exposes a NodePort service on each calling node that will serve as the entry point for calling traffic. A hostname resolving to one or more nodes or an IP address of one or more nodes must be provided. You can choose a port range from 30000-32767 for UDP and optionally TCP traffic.
- Existing NLB
  - Attaches the calling ingress service to an existing NLB. You will need to supply the target group ARN for UDP and optionally, TCP traffic.
- No Service
  - Select this if you don't need an additional Kubernetes service to allow ingress traffic. This will typically be used with the host network setting to route calling ingress traffic directly to your calling nodes.

Considerations

- To ensure backwards compatibility with older clients and federated networks without calling ingress, when calling ingress is enabled, the legacy calling mode is still available (direct

connection to the calling servers). If changing any default ports, ensure that you do not have any port collisions on the calling nodes.

- Dual-stack NLBs serving UDP traffic must have IPv6 backend targets. For more information, see Network Load Balancer Target Groups.
- If you require STIG compliance, you must disable the host network option for calling. If the nodes are configured in dual-stack mode, but the cluster is not, you may lose IPv6 connectivity (assuming an IPv4 cluster).
- Calling ingress requires pre-defined host names or IP Addresses. Scaling nodes or providing custom routing may require modifying configuration.
- Default calling ingress ports are 8443 for TCP and 16384 for UDP. Make sure firewalls and security groups allow traffic for these ports, or alternative ports if the defaults are overridden.

Reference architectures

Ingress with load balancer

This option exposes a single load balancer as the entry point for all calling traffic.

1. For **Calling Ingress Type**, choose either **Load Balancer** or **Existing NLB**. For more information about **Existing NLB**, reference the NLB stack in the Wickr Enterprise CDK Sample on GitHub.
2. Do one of the following, depending on the **Calling Ingress Type**:

   - For **Existing NLB**, provide the target group ARNs for UDP and TCP traffic and the hostname of the NLB.
   - For **Load Balancer**, provide the hostname after it is provisioned by Kubernetes.

   Alternatively, for either **Calling Ingress Type**, you can provide the IP addresses of the load balancer or a custom hostname that points to the load balancer.
3. (Optional) To combine messaging and calling traffic under a single NLB, choose **Existing NLB** in the **Ingress** section, and provide an HTTPS target group.

Ingress with NodePort

This option is useful if host networking is disabled and you don't want to expose an additional load balancer.

> **Note**
>
> Make sure your firewalls and security groups allow traffic for the NodePorts.

1. For **Calling Ingress Type**, choose **NodePort**.
2. Add the calling node hostnames or IP addresses.
3. Disable **Calling Host Network**.

Direct ingress with HostNetwork

This option does not expose any additional Kubernetes service and allows calling ingress traffic to connect directly over the calling nodes' host network. This approach is preferred if IPv6 connectivity is required.

1. For **Calling Ingress Type**, select **No Service**.
2. Add the calling node hostnames or IP addresses.
3. Enable **Calling Host Network**.

Kubernetes cluster autoscaler (optional)

Kubernetes Cluster Autoscaler is an optional configuration value for the Wickr Enterprise installation. It will aid in scaling your Kubernetes node groups in the event of increased traffic or other resource restrictions that could lead to poor performance.

The Wickr Enterprise installation supports 3 Cloud provider integrations: AWS, Google Cloud, and Azure. **Each Cloud provider has different requirements for this integration.** Please follow the instructions for your specific cloud provider below to enable this feature.

AWS

If you did not use the WickrEnterpriseCDK to install your Wickr Environment on AWS, you will need to take some additional steps to enable the Cluster Autoscaler.

1. Add the following tags to your Node Groups. This allows the Cluster Autoscaler to autodiscover the appropriate nodes.

1. `k8s.io/cluster-autoscaler/`**`clusterName`** `=` owned where **clusterName** is the name of your Kubernetes Cluster
2. `k8s.io/cluster-autoscaler-enabled = true`

2. Add a Kubernetes Service Account, in the kube-system namespace and associate it with an IAM policy that allows autoscaling and ec2 actions. For more information and detailed instructions, see Configuring a Kubernetes service account to assume an IAM role in the *Amazon EKS User Guide*.

   1. You'll need to use the 'kube-system' namespace when setting up the Service Account
   2. The following policy can be used for the Service Account:

      JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Action": [
                "autoscaling:DescribeAutoScalingGroups",
                "autoscaling:DescribeAutoScalingInstances",
                "autoscaling:DescribeLaunchConfigurations",
                "autoscaling:DescribeTags",
                "autoscaling:SetDesiredCapacity",
                "autoscaling:TerminateInstanceInAutoScalingGroup",
                "ec2:DescribeInstanceTypes",
                "ec2:DescribeInstances",
                "ec2:DescribeLaunchTemplateVersions"
               ],
            "Resource": "*",
            "Effect": "Allow"
        }
    ]
}
```

In the Replicated UI when configuring the Cluster Autoscaler, select **AWS** as your cloud provider and provide the name of the Service Account you created above to instruct the Cluster Autoscaler to utilize that service account.

Google cloud

It is highly recommended to use the built-in Autoscaling capabilities from GKE for both Autopilot and standard clusters. However, if you wish to proceed with this integration, the following requirements must be met before proceeding.

Requirements:

1. The Managed Instance Groups (MIG) must be created with Security Scope including at a minimum 'Read/Write' to Compute Engine Resources. This cannot be added to the MIG later currently.
2. Cluster must have Workload Identity Federation enabled. You can enable this on an existing cluster by running: `gcloud container clusters update ${CLUSTER_NAME} --workload-pool=${PROJECT_ID}.svc.id.goog`
3. A Google Cloud Platform (GCP) Service Account with access to the role `roles/compute.instanceAdmin.v1`. This can be created using these instructions:

```
# Create GCP Service Account
gcloud iam service-accounts create k8s-cluster-autoscaler

# Add role to GCP Service Account
gcloud projects add-iam-policy-binding ${PROJECT_ID} \
--member "serviceAccount:k8s-cluster-autoscaler@${PROJECT_ID}.iam.gserviceaccount.com"
 \
--role "roles/compute.instanceAdmin.v1"

# Link GCP Service Account to Kubernetes Service Account
gcloud iam service-accounts add-iam-policy-binding k8s-cluster-autoscaler@
${PROJECT_ID}.iam.gserviceaccount.com \
--role roles/iam.workloadIdentityUser \
--member "serviceAccount:${PROJECT_ID}.svc.id.goog[kube-system/cluster-autoscaler-gce-
cluster-autoscaler]"
```

Azure

Azure Kubernetes Service (AKS) provides integrated cluster autoscaling for most deployments and it is highly recommended to utilize those methods for cluster autoscaling. However, if your requirements are such that those methods do not work, we have provided a Kubernetes Cluster Autoscaler integration for Azure Kubernetes Service. To utilize this integration you will need to

gather the following information and put them in the configuration of the KOTS admin panel under **Cluster Autoscaler** after selecting **Azure** as your cloud provider.

Azure Authentication

Subscription Id: The subscription ID can be obtained via the Azure portal by following the official documentation. For more information, see Get subscription and tenant IDs in the Azure portal.

The following parameters can be obtained by creating an AD Service Principal using the az command line utility.

```
az ad sp create-for-rbac –role="Contributor" –scopes="/subscriptions/subscription-id" –
output json
```

App ID:

Client Password:

Tenant ID:

Azure Cluster Autoscaler Configuration

In addition to the authentication requirements, the following fields are necessary for proper functioning of the cluster autoscaler. Commands for obtaining this information has been provided for convenience, however, they may require some modifications depending on your specific AKS configuration.

Azure Managed Node Resource Group: This value is the Managed Resource Group created by Azure when you established the AKS Cluster and not the Resource Group you defined. To obtain this value, you need the CLUSTER_NAME and RESOURCE_GROUP from when you created the cluster. Once you have those values, you can obtain this by running:

```
az aks show –resource-group ${RESOURCE_GROUP} –name ${CLUSTER_NAME} –query
 nodeResourceGroup -o tsv
```

Application Node Pool VMSS Name: This is the name of the Virtual Machine Scaling Set (VMSS) associated with your AKS Nodepool for the Wickr Application. This is the resource that will be scaling up or down based on the needs of your cluster. To obtain this value you can run the following **az** command:

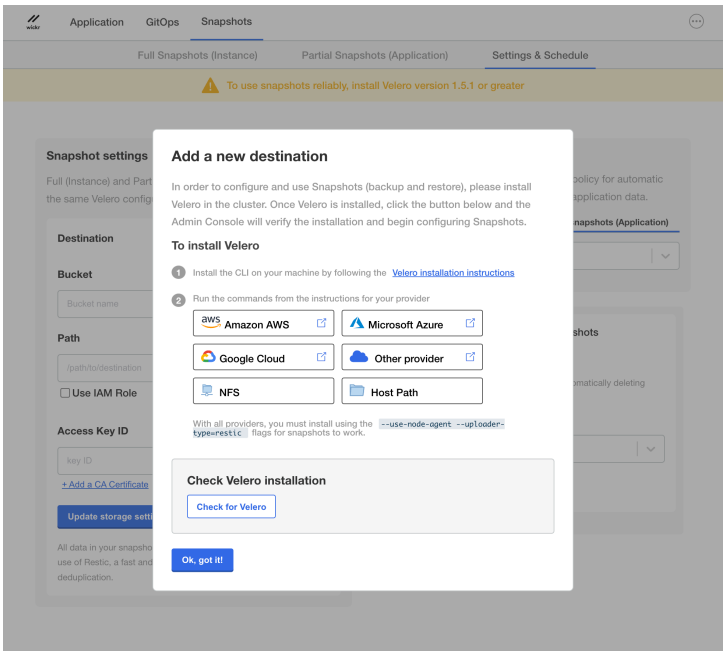
```
CLUSTER_NODEPOOL_NAME="(Your-NodePool-Name)"
CLUSTER_RESOURCE_GROUP="(Your-Managed-Node-Resource-Group-As-Defined-Above>)"
az vmss list -g ${CLUSTER_RESOURCE_GROUP} --query '[?tags."aks-managed-
poolName"==`'''${CLUSTER_NODEPOOL_NAME}'''`].{VMSS_name:name}' -o tsv
```

ACalling Node Pool VMSS Name (optional): This is the name of the VMSS associated with your calling Nodepool if you have one. To obtain this value, you can run a modified version of the command for Application Node Pool VMSS Name switching out the CLUSTER_NODEPOOL_NAME value for the name of the nodepool for your calling nodepool.

Backups

Wickr Enterprise utilizes Velero for Backup purposes. Velero provides the necessary tools for backing up and restoring Kubernetes cluster resources and persistent volumes, whether operating on a cloud provider or on-premises.

Velero backups with Minio: Currently Velero backups are only enabled for Minio in Low Resource Mode.



Installation using Velero documentation

- Install the Velero CLI. For more information, see Installing the Velero CLI.
- Install Velero on your cluster and configure storage based on your provider:

- AWS.
- GCP.
- Azure.
- Other providers.

Limitation

By default, no volumes are included in the backup. If any pods mount a volume that should be backed up, you must configure the backup with an annotation listing the specific volumes to include in the backup.

For each volume that requires a backup, add the backup.velero.io/backup-volumes annotation. The annotation name is backup.velero.io/backup-volumes and the value is a comma separated list of volumes to include in the backup. For more information, see Configure Snapshots.

Airgap installation

Wickr Enterprise and KOTS both support deployment into a fully airgapped Kubernetes cluster. You must provide access to a Private Docker Image Registry that is reachable from the airgapped Kubernetes cluster. The Private Docker Image Registry supplied to KOTS must be secured with username/password authentication to function correctly for this purpose. KOTS will utilize the Private Docker Image Registry to host all of the Wickr Enterprise images.

- Wickr Enterprise license.yaml with airgap enabled (Contact Wickr Sales or Customer Support Team)
- Wickr Enterprise wickr.airgap archive bundle (Contact Wickr Sales or Customer Support Team)
- Access to a Private Docker Image Registry .
- Access to a Kubernetes cluster deployed in the airgap environment.
- Kubectl installed.
- KOTS CLI installed.
- kotsadm.tar.gz downloaded.

Run the following commands to deploy KOTS and Wickr Enterprise on your airgapped kubernetes cluster. These commands upload the KOTS admin images and the the Wickr Enterprise images to

the Private Docker Image Registry. After the commands finish you will be prompted to access the KOTS Admin Console to complete the Wickr Enterprise installation as above.

```
kubectl kots admin-console push-images \
  ~/kotsadm.tar.gz $PRIVATE_REGISTRY_HOST \
  --registry-username $PRIVATE_REGISTRY_USER \
  --registry-password $PRIVATE_REGISTRY_PASSWORD

kubectl kots install wickr \
  --license-file ~/YOUR_LICENSE.yaml \
  --airgap-bundle ~/wickr.airgap \
  --kotsadm-registry $PRIVATE_REGISTRY_HOST \
  --registry-username $PRIVATE_REGISTRY_USER \
  --registry-password $PRIVATE_REGISTRY_PASSWORD
```

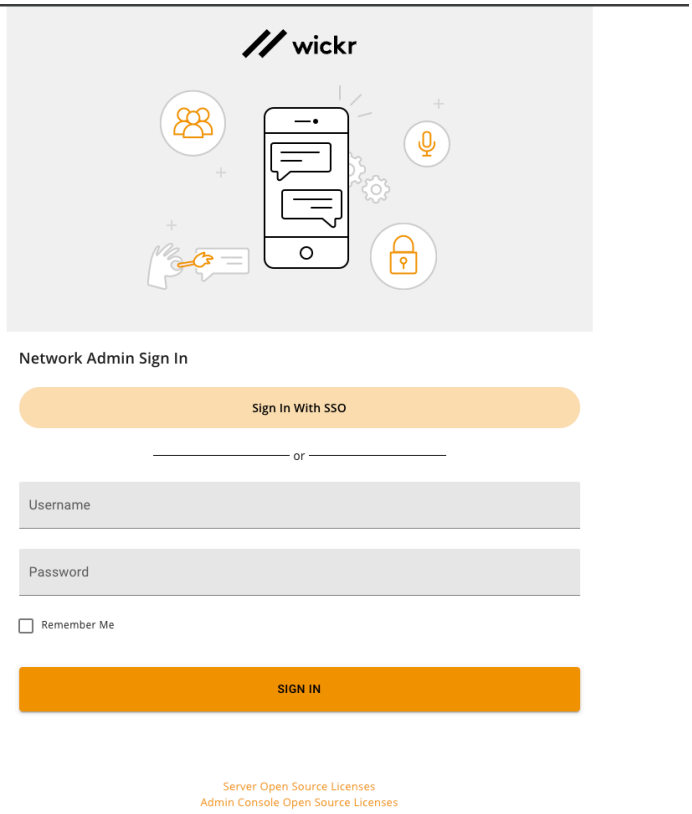
Mobile notification for airgap installs

Additional networking allow lists are necessary for push notifications from server backend to mobile clients. This requirement is due to how Apple iOS and Google Android implement this feature for offline and background devices. Refer to the documentation for these services and allow list the specified IP addresses and ports.

- iOS
- Android

Wickr admin console

The Wickr Admin Console interface is used for administering the Wickr Enterprise application itself. It can be used to set up networks, users, federation, and more. It's accessible over HTTPS at the DNS name that you configured to point to your Load Balancer. The default username is admin, with the password Password123. You will be required to change this password on first log in.



Security settings

AWS Wickr Enterprise provides configuration settings to enforce an enhanced security context for your deployment. This higher security standard is applied at the pod and container level, and is required for compliance with the Security Technical Implementation Guide (STIG).

Set the following configuration parameters to enforce the enhanced security context:

```
podSecurityContext:
  runAsNonRoot: true
  seccompProfile:
    type: RuntimeDefault
containerSecurityContext:
  allowPrivilegeEscalation: false
  capabilities:
    drop: ["ALL"]
```

> **Warning**
>
> For Opensearch, this security configuration disables the `fsgroup-volume` initContainer that updates permissions on the persistent storage, which can cause compatibility issues related to permissions.

FAQ

Q: My deployment fails with the following error in helm stderr:

```
Error: UPGRADE FAILED: cannot patch "enterprise-init" with kind Job:
Job.batch "enterprise-init" is invalid: spec.template: Invalid value: core.
```

A: This can happen when Debug Logging is enabled. Please disable debug logging, delete the problematic jobs, and try again.

Embedded cluster for Wickr Enterprise

The embedded cluster install option for Wickr Enterprise provides a small, efficient installation offering for the Wickr Enterprise product. It leverages the Replicated Embedded Cluster to provide a small Kubernetes installation using k0s on which Wickr Enterprise can be installed. Using this installation method minimizes the technical skill requirements as well as the overall hardware requirements for a Wickr Enterprise installation by providing an “all-in-one” solution at the cost of resiliency and high availability.

Topics

Getting started with Wickr Enterprise embedded cluster

To begin using the Wickr Enterprise embedded cluster option, contact support to receive a license. If you have an existing license and would like to utilize this option, contact support for assistance in updating your existing license and additional installation instructions.

Wickr Enterprise embedded cluster requirements

Before you start to install Wickr Enterprise embedded cluster, verify that the following requirements are met.

Network requirements

You will need to allow ingress to your Wickr server on the following ports:

- 443/TCP for HTTPS
- Calling TCP Proxy Only - The TCP proxy port configured for TCP Calling traffic in KOTS

- 16384-19999/UDP for UDP Calling traffic
- LAN Only - 30000/TCP for Accessing the KOTS Admin Console

System requirements

Before installation, make sure you have either a VM (Virtual Machine) or a physical machine running a Linux based Operating System (OS) with the following minimum resources available:

- 8 CPU cores
- 12 gigabytes (GB) of RAM
- 100 gigabytes (GB) of disk storage on the / (root) partition

The Wickr Enterprise embedded cluster has been tested on the following Linux OS systems but other Linux based OS options may be suitable as well:

- Red Hat Enterprise Linux 9.5
- Amazon Linux 2023
- Rocky Linux 9.5

Installation of Wickr Enterprise embedded cluster (standard)

Once you have the download instructions, download the Wickr Enterprise bundle to the destination machine and unpack it.

```
curl -f "https://replicated.app/embedded/wickr-enterprise-ha/stable/6.52" -H
 "Authorization: [redacted]" -o wickr-enterprise-ha-stable.tgz
tar xvf wickr-enterprise-ha-stable.tgz
```

You now should have two files, `wickr-enterprise-ha` and `license.yaml`. The `wickr-enterprise-ha` file is a binary file that includes all the necessary pieces for installation of the Embedded Cluster, whereas the `license.yaml` is your Wickr license that will be used to validate your installation.

A basic installation can be performed at this stage by executing the `wickr-enterprise-ha` file:

```
./wickr-enterprise-ha install --license license.yaml
```

Once the installation process begins, you are prompted to enter an Admin Console password. Enter a secure password and ensure you save it as you will need it when accessing the KOTS Admin Console to continue configuring your installation.

After the installation is complete, the output resembles the following:

```
   sudo ./wickr-enterprise-ha install --license license.yaml
? Set the Admin Console password (minimum 6 characters): *********
? Confirm the Admin Console password: *********
# Host files materialized!
# Host preflights succeeded!
# Node installation finished!
# Storage is ready!
# Embedded Cluster Operator is ready!
# Registry is ready!
# Application images are ready!
# Admin Console is ready!
Visit the Admin Console to configure and install wickr-enterprise-ha:
 http://192.168.1.100:30000
```

After the standard installation, proceed to the KOTS admin console URL provided in the output using a web browser. For this example, the URL is `http://192.168.1.100:30000`. However your URL will differ based on your networking configuration.

Multi-Node installation

Wickr Enterprise Embedded Cluster Multi-Node installations provide an option for Embedded Cluster users to separate the Wickr Calling and Wickr Messaging workloads on to different physical machines. To do this, Wickr Enterprise leverages the Replicated Embedded Cluster Multi-Node tooling.

Port requirements

The following ports must be open on all members of the cluster for the Multi-Node functionality to work correctly. These only need to be open between the nodes themselves, and not open to the wider internet.

- 53 TCP/UDP
- 2380/TCP
- 4789/UDP
- 6443/TCP
- 8080/TCP
- 9091/TCP
- 9443/TCP
- 10249/TCP
- 10250/TCP
- 10256/TCP
- 30000/TCP
- 50000/TCP

License requirements

The Wickr Embedded Cluster Multi-Node configuration options require additional license privileges. Contact Support to make sure your license supports this feature.

Creating an additional node during initial setup

When you initially configure the Wickr Enterprise Embedded Cluster, you can create an additional calling node during the setup process. Start by following the procedure described in Installation of Wickr Enterprise embedded cluster (standard) When you navigate to the KOTS admin panel, you will be prompted to create additional nodes.

> **Note**
>
> Currently, Embedded Cluster Multi-Node supports only 1 calling node and 1 messaging/controller node.

To begin, deselect the **Controller** role option and select the **Calling** role option. This populates additional instruction sets for configuring the new node. Run these instructions on the new node to configure it to join the cluster as a calling node.

Run instructions similar to the following examples on the new node:

1. Download the binary on the new node:

```
curl -k https://172.31.42.64:30000/api/v1/embedded-cluster/binary -o wickr-
enterprise-ha.tgz
```

2. Extract the binary:

```
tar -xvf wickr-enterprise-ha.tgz
```

3. Join the node to the cluster:

```
sudo ./wickr-enterprise-ha join 172.31.42.64:30000 AAAAAbbbbbbbbCCCCCCCzzzzz
```

After the join command completes successfully, the new node appears on the **Configure cluster** page with the **Calling** role assigned. Choose **Continue** to proceed to the Wickr Enterprise configuration page. Follow directions for embedded node configuration options outlined in KOTS admin console configuration.

Adding an additional node to an existing embedded cluster installation

To add a calling node to an existing Wickr Enterprise Embedded Cluster installation, navigate to the KOTS Admin Console. To do this, log in to the node through ssh or other mechanism and navigate to the installation directory that contains the `wickr-enterprise-ha` binary used for installation. Run `./wickr-enterprise-ha admin-console` to start the KOTS Admin Console. If this command does not return any output, the KOTS Admin Console is already running and can be accessed by navigating to port 30000 on the IP of the node in a web browser, for example: `https://127.0.0.1:30000/`.

Enter the KOTS Admin password when requested, then perform the following procedure to create an additional node:

1. Once logged in, navigate to the **Cluster Management** page on the top left of the KOTS Admin Console.
2. Choose **Add node**.
3. Deselect **Controller** under `Roles`.
4. Select **Calling** under `Roles`
5. Follow the instructions provided to run the commands on the new node you want to add.

6. When finished, choose **Close**
7. Your new node appears in the **Nodes** list with the **Calling** role.
8. Navigate to the **Application** page at the top-left of the KOTS Admin Console
9. Choose **Config** from the navigation bar at the top of the page.
10. Navigate to the **Calling** section in the left navigation panel.
11. Select **Require Calling Nodes** to allow use of the Calling node.
12. Scroll to the bottom of the page and choose **Save config**.
13. A popup appears, indicating the Config has been updated. Choose **Go to updated version**.
14. On the updated version page, the currently installed version is displayed. A new line item is listed under installed versions with the designation **Config Change**. Choose **Deploy** to deploy this new version and enable the new calling node.

KOTS admin console configuration

The KOTS admin console initially uses a self-signed certificate, which you'll need to allow as an exception in your browser. Once you accept this exception, you are welcomed by the Configuration Wizard for the KOTS admin console. This wizard guides you through additional configuration steps for configuring the KOTS Admin Console's behavior, including the option to add a custom certificate if necessary.

After initial configuration of the KOTS admin console is complete, you are prompted to enter your admin console password that you created during the installation process. Upon your first login you need to configure the cluster.

Choose **Continue** to proceed to the KOTS admin console for Wickr.

For a single node Embedded Cluster, choose **Continue** to proceed to the KOTS admin console for Wickr. For multi-node installations, see Multi-Node installation.

Once in the KOTS admin console, configure your installation according to your needs. When utilizing the embedded cluster offering there are some key configuration settings that should be set to ensure proper functionality of your Wickr Enterprise installation.

- **Hostname** - This is the hostname you use when communicating with the Wickr installation. Make sure you create appropriate DNS records for this domain to point to your Wickr Enterprise installation.

- Under **Advanced Options**, check the [ ] **Configure Ingress Controller** option to expose a configuration block for configuring the Kubernetes Ingress. In the **Ingress configuration block** , select **Single Node Embedded Cluster**, and then enter the "public" IP associated with your Wickr server in the text box labelled **Loadbalancer External IP (IPv4 Only)**.

  If you are unsure what this IP is, you can run the following command from the command line on the Wickr server to determine this value: `ip route get 1.1.1.1|awk '{print $7}'`
- Under **Advanced Options**, check the **Enable Low Resource Mode** option.
- Under **Calling**, if you are using a single node Embedded Cluster, make sure **Require Calling Nodes** is deselected. Otherwise, if you have added a Calling node during the initial setup, make sure **Require Calling Nodes** is selected.
- If you want an all in one solution that does not utilize an external Database or S3 Compatible storage for file sharing, select the **Internal** options for the following settings:
  - Database
  - S3 Storage Location

    The internal S3 storage location provides additional options for configuring storage capacity. It is recommended to start small and expand as necessary, since scaling down is not an option after provisioning.

Once you have configured all the necessary features, scroll to the bottom of the configuration page, and choose **Save Config**. This will initiate some preflight host checks. Once the preflight checks are complete, choose **Deploy** to begin the Wickr Enterprise Installation.

Now you are ready to begin configuring your Wickr Enterprise installation. For more information on configuring Wickr Enterprise, see What is Wickr Enterprise?.

Additional Common Installation Requirements

IP Hostname Installations

If your installation requires an IP based hostname, there are some additional configuration options. These instructions are specific for IP based hostnames, and it is recommended you follow the other instructions for the basic setup listed above.

In the KOTS admin panel, complete the following steps.

1. Set the **Hostname** to the IP you will be using.

2. Under **Certificates**, select **Upload a Certificate**. Then, generate a self-signed certificate following the instructions for an IP based certificate. For more information, see Generating a self-signed certificate.
3. Upload the `.crt` file for the Certificate and the `.key` file for the Private key
4. For the Certificate Chain, upload the `.crt` file again.
5. Check **Set a pinned certificate** checkbox.
6. Upload the `.crt` for the **Pinned Certificate**.
7. Under **Calling**, uncheck **Automatically discover server public IP addresses** and **Use host primary IP address for Calling traffic** checkboxes.
8. Under **Calling**, put the IP address of the hostname in the **Hostname Override** text box.
9. Under **Advanced Options**, check the **Configure Ingress Controller** checkbox. A new configuration section called **Ingress** appears below.
10. Under **Ingress**, select **Single Node Embedded Cluster**.
11. Under **Ingress**, enter the IP for the 'public' interface on the Wickr server. This may be different than the IP used as your hostname. See additional information on this value in the basic configuration steps.
12. Under **Ingress**, check **Use wildcard hostname**.

SELinux Enforcing Mode

If you require the use of SELinux in enforcing mode, modify the default data directory used to install the embedded cluster. It is recommended to use `/opt` as it has been tested to work with most SELinux policies for this use case.

```
mkdir /opt/wickr
./wickr-enterprise-ha install --license license.yaml --data-dir /opt/wickr --ignore-
host-preflights
```

The replicated embedded clusters default installation preflight checks will attempt to validate that SELinux is in permissive mode and fail if SELinux is in Enforcing. To bypass this, it is required to use the `--ignore-host-preflights` command line argument. When using the command line option, there is a prompt similiar to the one below. Enter **Yes** when prompted.

```
#  1 host preflight failed
```

```
•  SELinux must be disabled or run in permissive mode. To run SELinux in permissive
 mode, edit /etc/selinux/config, change the line
'SELINUX=enforcing' to 'SELINUX=permissive', save the file, and reboot. You can run
 getenforce to verify the change."

? Are you sure you want to ignore these failures and continue installing? Yes
```

AirGap installations

The embedded cluster installation option for Wickr Enterprise supports airgapped installations. Additional configuration and enablements for your license are required. Contact support if you are interested in using Wickr Enterprise embedded cluster in an airgapped environment.

When performing an airgap installation, the download instructions differ from the standard installation method. They should resemble the following:

```
curl -f "https://replicated.app/embedded/wickr-enterprise-ha/stable/6.52?airgap=true" -
H "Authorization: [redacted]" -o wickr-enterprise-ha-stable.tgz
```

Download the bundle to a machine that has internet access, then transfer it to your airgapped environment using your preferred data transportation method. Once the bundle is transferred, extract it as you would with any standard installation bundle. A third file `wickr-enterprise-ha.airgap`, containing all the associated Wickr Enterprise application service images will be included.

```
tar xvf wickr-enterprise-ha-stable.tgz
```

During installation, it is necessary to set the `--airgap-bundle` command line argument after extraction; otherwise, the process follows the standard installation procedure.

```
./wickr-enterprise-ha install --license license.yaml --airgap-bundle wickr-enterprise-
ha.airgap
```

Updating an airGapped embedded cluster

To update an AirGapped Embedded cluster, complete the following steps.

1. Download the new embedded cluster package from Replicated, and transfer it to the host machine using your standard data transfer methods for your airgapped environment. After the new bundle is on the host machine, extract the tarball:

```
tar xvf wickr-enterprise-ha-stable.tgz
```

2. Run the update using the new binary and airgap bundle:

```
./wickr-enterprise-ha update --airgap-bundle wickr-enterprise-ha.airgap
#  Application images are ready!
#  Finished!
```

3. Start the KOTS admin console, and login to the provided URL using your standard methods of accessing the KOTS admin console

```
./wickr-enterprise-ha admin-console
```

4. Once logged in to the KOTS Admin Console, find the **Latest Available Update** on the left under **Version** , and then press the **Go to Version history** button.
5. Choose **Deploy** for the new version under **Available Updates**. Walk through the screens:
    1. Change any configuration options, scroll down, and then choose **Next**.
    2. Verify no preflight checks failed, choose **Next: Confirm and deploy**.
    3. Choose **Deploy**.

Additional notes on the Wickr Enterprise embedded cluster

- **NAMESPACE**: Unlike most Wickr Enterprise installations, the embedded cluster installation installs the Wickr assets to the *kotsadm* namespace in kubernetes and not *wickr*. Modify any scripts or commands you have saved that use `-n wickr` for kubectl, helm or any other utility to use `-n kotsadm` instead.
- **Interacting with the Kubernetes Cluster**: From the host machine, use the `./wickr-enterprise-ha` binary to create a shell with appropriate variables set to interact with the Kubernetes installation by running **./wickr-enterprise-ha shell**. This will provide the kubectl utility within the shell's PATH and set the appropriate kube config to the local installation.

Troubleshooting Wickr embedded cluster installations

All instances of these troubleshooting steps assume you have shell access to the instance running the Wickr Embedded Cluster installation and have run the `./wickr-enterprise-ha shell` command to be able to interact with the Kubernetes installation directly.

General issues

Add Node Button Missing From Cluster Management Screen

Airgapped Installs

If you are on an airgap installation, please reach out to Wickr Support for assistance in correcting this behavior.

Standard Installs

If your license includes the Embedded Cluster Multi-Node entitlement, perform a license sync to get the latest version. If you are unsure or do not have this entitlement, please reach out to Wickr Support.

To perform a license sync, complete the following steps.

1. Navigate to the KOTS control panel.
2. On the **Dashboard** page, locate the license section in the upper right area of the page.
3. Within this section, in the top right corner, you should see a **Sync License** hyperlink. Select the hyperlink.
4. Once the license has been synced, the UI updates and **Last synced a few seconds ago** appears.
5. Choose **Redeploy** from the **Version** section of the KOTS dashboard page.
6. Once the redeploy finishes, navigate back to **Cluster management** and you can add nodes.

Upgrade issues

Upgrade Stuck on Upgrading Cluster

If your upgrade gets stuck on **Upgrading Cluster** this likely means that some pods are not being terminated appropriately. Log on to the instance and use the `./wickr-enteprise-ha shell` command to enter the shell environment for managing the kubernetes installation.

1. Identity the pods that are still running:

   ```
   kubectl -n kotsadm get pods | grep Running
   ```

2. `kubectl -n kotsadm delete pod name-of-running-pod`

   > **Note**
   >
   > If one of the running pods is `embedded-cluster-upgrade-XXXXXXXXXXXXXX-xxxxx` or `kotsadm-xxxxxxx` or similar, do not delete it as these pods are necessary for performing the upgrade.

3. Verify there are no remaining running pods.

   ```
   kubectl -n kotsadm get pods | grep Running
   ```

This procedure should allow the cluster upgrade to proceed by the Wickr upgrade.

Application Did Not Update During Cluster Upgrade And Cannot Deploy New Version

If the application remains on the old version after the upgrade, the new version may be in an inconsistent state.

Check the Kubernetes installation records:

1. Open the Kubernetes shell from the installer.

   ```
   ./wickr-enterprise-ha shell
   ```

2. Run the following kubectl command:

   ```
   kubectl get installations
   ```

3. Output will look something like this:

   ```
        [root@ip-172-31-6-72 ~]# kubectl get installations
   NAME              STATE      INSTALLERVERSION   CREATEDAT              AGE
   20251113170603    Obsolete   2.1.3+k8s-1.30     2025-11-13T17:06:05Z   22h
   20251113180133    Failed     2.6.0+k8s-1.31     2025-11-13T18:01:37Z   21h
   ```

4. Delete the failed installation.

```
kubectl delete installation 20251113180133
```

5. Attempt to run the upgrade again through KOTS Admin panel.

RabbitMQ Pod Failing with log lines `Error while waiting for Mnesia tables: {timeout_waiting_for_tables}`

The RabbitMQ secret and storage are out of sync. This usually happens when multiple RabbitMQ instances run and cause leader selection or quorum error. To fix this, delete the RabbitMQ service and its storage volumes, then redeploy.

To delete the failing RabbitMQ, complete the following steps.

1. Delete the RabbitMQ Statefulset.

   ```
   kubectl -n kotsadm delete statefulset rabbitmq —cascade=orphan
   ```

2. Delete the remaining RabbitMQ pods. If there are multiple RabbitMQ-X pods running, issue this command multiple times updating the RabbitMQ-X value to correspond to the additional pod names.

   ```
   kubectl -n kotsadm delete pod rabbitmq-0
   ```

3. Delete the corresponding PVCs. If there are multiple pods running, issue this command multiple times updating the data-RabbitMQ-X to correspond to the appropriate pods.

   ```
   kubectl -n kotsadm delete pvc data-rabbitmq-0
   ```

4. Check if there are any remaining pods, this should output nothing if successful.

   ```
   kubectl -n kotsadm get pods|grep -i rabbitmq
   ```

5. Check if there are any remaining PVCs, this should output nothing if successful.

   ```
   kubectl -n kotsadm get pvc|grep -i rabbitmq
   ```

6. Redeploy through KOTS Admin Panel.

For more troubleshooting information, see Troubleshooting.

Document history

The following table describes the documentation releases for Wickr Enterprise Automated Install Guide.

Change	Description	Date
Security settings	Security settings have been added. For more information, see Security settings.	August 26, 2025
Multi-Node installation	Multi-Node installation has been added. For more information, see Multi-Node installation.	August 26, 2025
Calling ingress settings	Calling ingress settings have been added. For more information, see Calling ingress settings.	August 26, 2025
Automatic deployment options	Automatic deployment options have been added. For more information, see Installing Wickr Enterprise.	February 23, 2024
Ports to allowlist	Port TCP/8443 has been added to the allowlist. For more information, see Requirements.	February 12, 2024
Destroying resources and Ports to allowlist	Instructions on how to destroy resources have been added. For more informati on, see Destroying resources. Additionally, ports to allowlist has been added. For more	August 17, 2023

	information, see Requirements.	
Initial release	Initial release of the Wickr Enterprise Automated Install Guide	August 4, 2023