Open for Innovation
KNIME

KNIME Edge Installation Guide

KNIME AG, Zurich, Switzerland
Version 1.3 (last updated on 2024-05-15)

Table of Contents

Introduction

This guide outlines the requirements, considerations, and steps for creating a KNIME Edge cluster.

Components of KNIME Edge

Short descriptions of the high-level architectural components which comprise KNIME Edge.

KNIME Edge control plane

Within the context of Kubernetes, the term "control plane" refers to a group of components that handle the management of a cluster's current state. In the case of KNIME Edge clusters, these clusters can be installed in various locations such as datacenters, manufacturing facilities, cloud providers, or environments with limited internet access. To ensure the edge clusters can function properly, they need to establish periodic communication with a control plane in order to retrieve the desired configuration for their deployments.

KNIME Business Hub has the capability to act as the control plane for a KNIME Edge cluster. They achieve this by utilizing a collection of KNIME workflows that serve as interactive data apps. These workflows enable the creation, updating, and deletion of edge deployments. The cluster's state is maintained by leveraging a PostgreSQL database, which is included in KNIME Business Hub.

KNIME Edge Operator

The KNIME Edge Operator has two responsibilities. Firstly, it takes on the responsibility of initializing the KNIME Edge stack, ensuring that all the necessary components are properly set up. Secondly, it plays a crucial role in reconciling deployments that have been configured through the control plane. This means that whenever a deployment is created, updated, or deleted, the operator takes action by reconfiguring the relevant Kubernetes resources within the cluster to reflect the desired changes.

KNIME Edge Inference Agent

The KNIME Edge Inference Agent provides a lightweight REST API on top of KNIME Analytics Platform which allows KNIME workflows to be built into container images so that they can be hosted in a Kubernetes cluster and scaled as needed behind a network load balancer. Only

one workflow can be hosted per container image, but multiple instances of the workflow can be loaded into memory. This approach helps maximize throughput and performance for KNIME workflow execution.

Inference Deployments

An Inference Deployment (`InferenceDeployment`) is a Kubernetes Custom Resource that contains the definition for a KNIME Edge deployment, including configuration options such as:

- The KNIME workflow to deploy
- Resource allocation
- Scaling configuration

Kong API Gateway

Kong is an open-source, third-party stack that provides KNIME Edge more flexibility and functionality in how external API requests are handled and proxied to inference deployments.

MinIO

MinIO is an embedded object store (which leverages the same API as the AWS S3 service). KNIME Edge uses MinIO to cache workflows, logs and other artifacts to optimize the amount of "fetching" required from KNIME Business Hub.

KNIME Edge Architecture

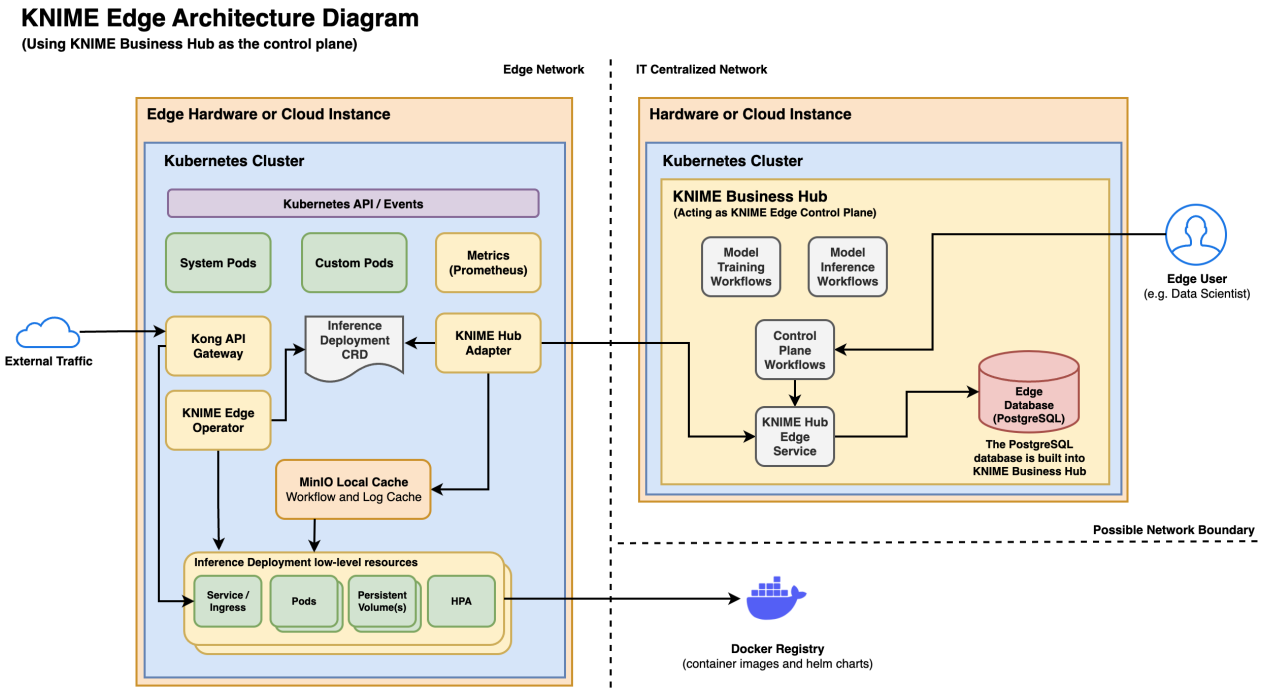
Below are high-level diagrams of the architectural components within KNIME Edge. In KNIME Edge 1.3+, KNIME Business Hub can be used as the control plane for KNIME Edge.

Using KNIME Business Hub as the control plane

There are several advantages to using KNIME Business Hub as the control plane for KNIME Edge:

- Installation & configuration of the KNIME Edge control plane is significantly easier due to the improved architecture of KNIME Business Hub.

- The PostgreSQL database for storing KNIME Edge configuration is included within KNIME Business Hub and requires no manual configuration.
- The KNIME workflows which act as the control plane are easier to deploy and use.



Installation Planning

Contact your customer care representative if you need assistance with installation planning.

Installation Method: Kurl vs. Helm

KNIME Edge supports two types of Kubernetes cluster installations, **embedded cluster** and **existing cluster**. For embedded clusters, Kurl is the recommended installation method. For existing clusters, Helm is the recommended installation method.

- **Embedded cluster installation via kurl**
  - Provides a guided installation of a Kubernetes cluster using kurl.
  - Future Kubernetes cluster version updates are performed by re-running the kurl installer. Version compatibility is detected and managed entirely by the installer.
  - Best for customers who are new to managing Kubernetes applications, but also great for any customers who want to benefit from automated cluster installation/upgrades.

    - Simplifies the cluster installation and upgrade process.
    - New nodes can be attached or removed from the cluster using kurl, but cluster autoscaling is not supported.
- **Existing cluster installation via Helm**
    - Requires the customer to provision and maintain their own cluster which adheres to the Replicated existing cluster installation requirements in addition to any requirements specified in this guide.
    - Requires the customer to ensure cluster version compatibility with KNIME Edge.
    - Best for customers who are experienced in Kubernetes and have multiple clusters running in production today. In particular, this option is great for customers whose IT departments have well-established policies and practices for maintaining Kubernetes clusters.
    - This option is more challenging in terms of initial configuration and requires more from the customer in terms of ongoing maintenance, but allows for advanced configurations.
    - With an existing cluster installation it is possible to take advantage of cluster autoscaling features from cloud providers such as EKS, GKE, or AKS.

KNIME Business Hub

Supported version(s): KNIME Business Hub 1.4.0+

KNIME Business Hub Standard (or higher) can be used as the control plane for KNIME Edge. If you're interested in KNIME Edge and want to understand more about the licensing, please contact our Customer Care team.

Kubernetes

Supported version(s): v1.21 to v1.30

A Kubernetes cluster must be available for KNIME Edge to function. In embedded cluster installations using kurl, the cluster is provisioned as part of the installation process. In all other configurations, the cluster must be provisioned (and upgraded) by the customer. See the Kubernetes documentation for details on cluster installation and usage.

Software Prerequisites

- kubectl: only required if installing into an existing cluster, or when remotely managing a cluster. When installing the embedded cluster with kURL kubectl is automatically installed on the host machine.
- Helm: only required if uninstalling KNIME Edge.
- **Recommended:** a Kubernetes integrated development environment (IDE) for easier inspection of Kubernetes resources within the KNIME Edge stack. There are many options available on the market such as Lens and OpenLens.

Networking Prerequisites

Trusted Host	Domain
KNIME	`https://registry.hub.knime.com`, `https://registry.hubdev.knime.com`,
Replicated	See the Firewall Openings for Online Installations guide.

Capacity planning & considerations

Memory per pod is the primary driver of infrastructure costs in a KNIME Edge cluster.

Warning: If insufficient memory is allocated to an Inference Agent pod, the pod will likely crash and Kubernetes will report an **OOMKilled** event for that pod.

When calculating needed capacity in Kubernetes for a KNIME Edge cluster, the **largest factor will be the expected number of Inference Deployment pods that will be running**. In Replicated installations, there are also minimum system requirements that should be added on top of expected capacity.

The base components of KNIME Edge require minimal resources to run, typically less than 1 CPU and around 1 GB of memory. The KNIME Edge Inference Agent pod(s) that are deployed for each Inference Deployment require additional resources. For most inference workflows, it is recommended that each corresponding KNIME Edge Inference Agent pod to be given an upper limit of **1 to 2 CPUs** and an upper limit of **2 GB of RAM**.

Note that each pod, once fully started, will typically average around **1.5 to 2 GB in active memory utilization**. This memory utilization is mostly static and does not typically increase or spike with request load.

CPU, by comparison, is heavily affected by request throughput. CPU usage will typically spike while requests are being processed and settle down to a lower level of utilization while waiting for new requests.

One useful method for gauging capacity requirements is to load test a single pod that utilizes the intended inference workflow and measure the number of requests handled and latency per request. By factoring in the expected number of concurrent Inference Deployments and number of pods running in each deployment, it's possible to estimate overall capacity needs.

It is recommended to add a ~20% buffer on top of estimated capacity for scalability. If a large-scale KNIME Edge cluster is being planned, the Considerations for large clusters page from the Kubernetes documentation may be useful.

Configure KNIME Business Hub

This section demonstrates how to configure KNIME Business Hub Standard 1.4.0+ instance (or higher) as the control plane for KNIME Edge.

Upload the KNIME Edge control plane workflows to KNIME Business Hub

To get started with KNIME Edge and KNIME Business Hub, follow these steps to install the KNIME Edge control plane workflows on the Hub. The control plane consists of a set of data apps that enable control and interaction with the remote KNIME Edge cluster.

Before proceeding, make sure you have a KNIME Community Hub account. If you don't have one, you can register at https://hub.knime.com/. Once you have an account, you can access the KNIME Edge control plane workflows in the KNIME Edge space.

There are two types of KNIME Edge control plane workflows:

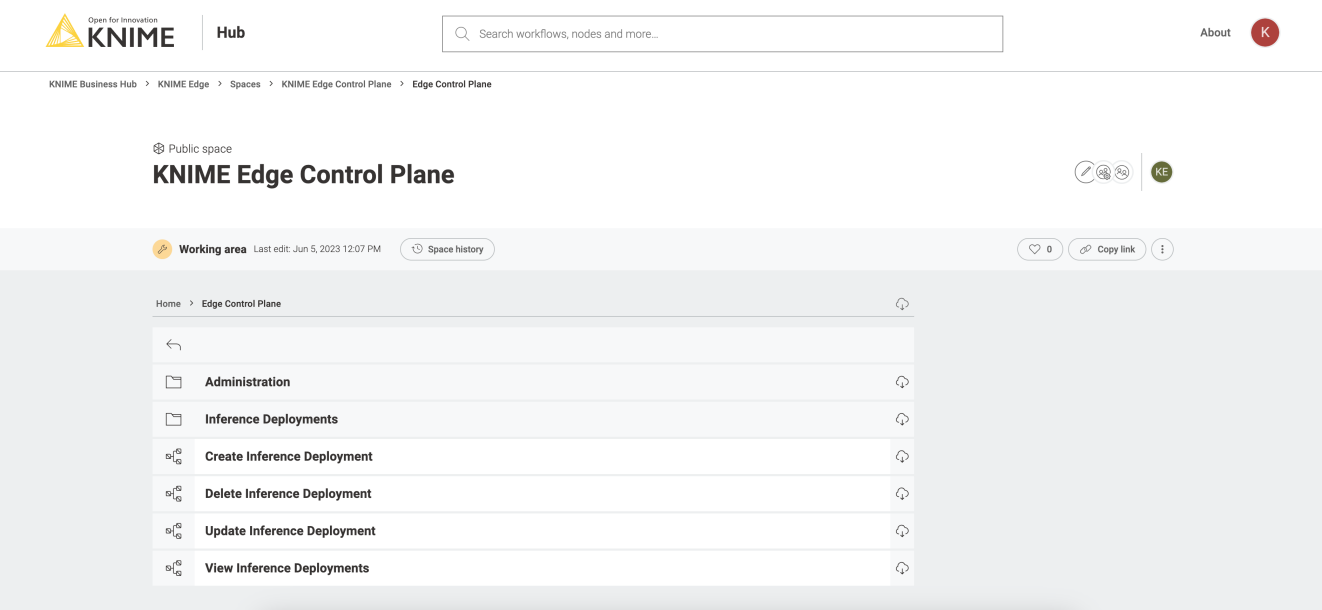
- **Administration workflows**
  - These workflows can be found in the "Administration" folder.
  - They include data apps for managing execution images and monitoring the status of connected clusters.
- **Inference deployment workflows**

- These workflows are located at the root of the KNIME Edge control plane workflows.
- They provide data apps for creating, reading, updating, and deleting inference deployments.

If you are an administrator of KNIME Business Hub, follow the steps below to upload the KNIME Edge control plane workflows and make them available to your team:

- **Create a new space and/or team in KNIME Business Hub.**
  - For example, you can create a team called "KNIME Edge" and a space named "KNIME Edge Control Plane".
- **Upload the KNIME Edge control plane workflows to the designated space.**
  - Make sure to maintain the default folder/workflow structure to avoid any unexpected errors during execution.
  - The "Inference Deployments" folder should contain both deployed workflows and those ready for deployment.
  - This folder serves as a central location for managing all workflows related to Inference deployments.
  - Once the workflows are uploaded, there is no additional configuration required to use the data apps. You can directly access and utilize them from the designated space.

Before deploying a workflow as an inference deployment in KNIME Edge, it should be uploaded to the "Inference Deployments" folder within the "KNIME Edge Control Plane" space. Once uploaded, it will be visible as a selectable workflow in the Create Inference Deployment data app.

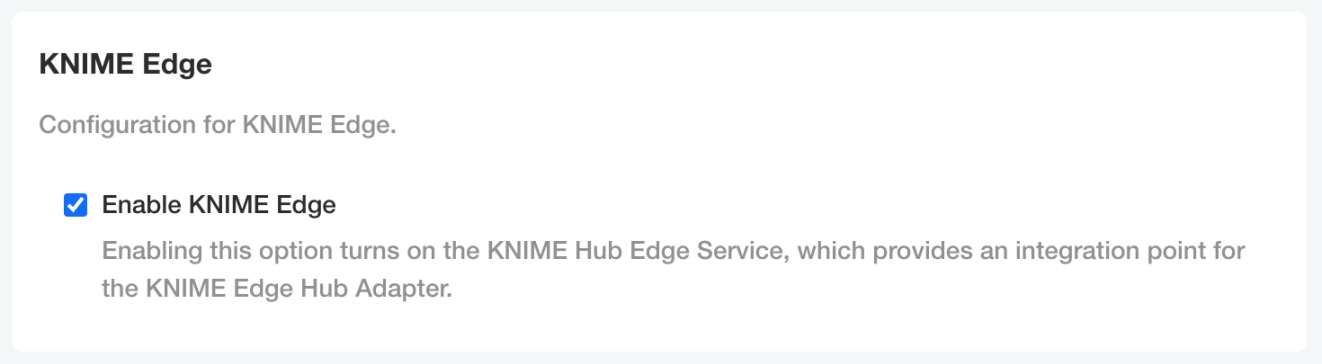


Configure the KNIME Hub Edge Service on KNIME Business Hub

If you are using KNIME Business Hub 1.4.0+ and you have KNIME Edge enabled on your Replicated (`.yaml`) license, an option for enabling KNIME Edge should display in the KNIME Business Hub KOTS Admin Console.

Contact your customer care representative if KNIME Edge needs to be enabled on your Replicated license.

This setting is visible in the KOTS Admin Console of KNIME Business Hub and will likely have to be enabled by the system administrator. Once enabled, deploy the changes to bring the KNIME Hub Edge Service online.



A note about authorization

The communication between the KNIME Edge control plane and any given KNIME Edge cluster occurs via REST API calls to the KNIME Hub Edge Service. The KNIME Edge Hub

Adapter handles the extraction of the authorization token which is used as a request header to authorize every call made to the KNIME Edge Service by the control plane workflows.

It's important to note that this authorization method only works when running data apps on KNIME Business Hub. The data apps cannot be executed locally in KNIME Analytics Platform and still successfully authenticate.

Installing KNIME Edge with KNIME Business Hub and Helm

Helm is an advanced installation method compared to the kurl installer and is the recommended installation method for KNIME Edge if you want to install KNIME Edge into an existing Kubernetes cluster. Installation via Helm gives you more configuration options compared to the installation via kurl.

Choosing a Kubernetes Namespace

Installing KNIME Edge into a namespace other than the `default` Kubernetes namespace is **highly recommended**, but optional. In kurl installations, KNIME Edge will install into the `edge` namespace. For more information see the Kubernetes Namespace documentation.

Using a Kubernetes namespace is recommended as it encapsulates all KNIME Edge resources within one scope and isolates them from any other resources running within the Kubernetes cluster. Scoping the KNIME Edge installation to a namespace also makes any potential manual maintenance operations safer since `kubectl` commands can be restricted to a specific namespace via the optional `[-n <namespace>]` flag.

This guide will reference to this optional, but recommended, namespace where applicable with `[-n <namespace>]`.

If desired, create a namespace using the following command:

```
kubectl create namespace <namespace>
```

Logging into the KNIME Artifact Registry

A valid KNIME Community Hub user (with elevated permissions) is required to access the `knime-edge` project, Docker images, and deployment charts in Harbor. Contact support@knime.com if help is needed accessing projects in Harbor.

The Docker images required for a KNIME Edge cluster are available in the KNIME Artifact Registry: https://registry.hub.knime.com, and in some cases, https://registry.hubdev.knime.com.

The `knime-edge` images and Helm charts can be found here.

Log into Harbor with a valid KNIME Community Hub user via the **OIDC Login** button.

Configuring Helm

In order to install a KNIME Edge cluster directly from the KNIME Artifact Registry, you need to add the KNIME Edge chart repository.

Authentication with the KNIME Artifact Registry and KNIME Edge chart repository requires a `username` and `password`. The `username` is the same as your Hub username. The `password` to specify is a CLI secret provided by Harbor. See the Using OIDC from the Docker or Helm CLI guide for information on how to retrieve the CLI secret (which acts as both the Helm and Docker password).

To add the Helm chart repository, run the following command (with `kubectl` already available and configured), and substitute the `username` and `password`.

```
helm repo add --username <username> --password <password> knime-edge \
https://registry.hub.knime.com/chartrepo/knime-edge
```

Adding a registry credential secret to Kubernetes

In order to pull Docker images directly from the KNIME Artifact Registry, a **Kubernetes Secret** with the name `regcred` needs to be created in the KNIME Edge cluster. Creating the secret is performed using the `kubectl` CLI.

Make sure the secret is created in the same **namespace** where **KNIME Edge** is intended to be deployed.


```
kubectl [-n <namespace>] create secret docker-registry regcred \
--docker-server=<your-registry-server> \
--docker-username=<your-name> \
--docker-password=<your-pword>
```

Verify by running:

```
kubectl [-n <namespace>] get secret regcred
```

Applying a KNIME Edge License

For the KNIME Edge cluster to work, a license is required. Further information about the licensing in KNIME Edge can be found in the KNIME Edge Licensing section of the KNIME Edge User Guide.

Once a license has been obtained, a **secret** with the name `knime-edge-license` that contains the license file needs to be created in the Kubernetes cluster.

Make sure the secret is created in the same **namespace** where **KNIME Edge** is intended to be deployed.

```
kubectl [-n <namespace>] create secret generic knime-edge-license --from
-file=license=<path-to-license-file>
```

KNIME Edge chart

The KNIME Edge Chart is a helm chart which defines the CRDs and dependencies needed by a KNIME Edge Cluster.

The parameters required to configure a KNIME Edge Cluster are specified within a standard Helm Values file, e.g. `values.yaml`, which is obtained from the `knime-edge` Helm repository.

The following steps outline the process of obtaining, understanding, customizing, and installing this chart.

Fetching the Helm values file for the KNIME Edge chart

Refresh Available Charts

```
helm repo update
```

Determine Version:

You can view all published versions of the KNIME Edge Helm charts in the Harbor registry. See the README in each chart version for the values file definition specific to that version. An overview of YAML techniques that can be used in the values.yaml file can be found in the Helm documentation.

(Advanced) All Chart Versions can be listed with the following command:

```
helm search repo knime-edge/knime-edge-operator --versions --devel
```

Values file configuration

Generate the file `values.yaml` to configure parameters for the KNIME Edge cluster:

For the latest stable release, run:

```
helm show values knime-edge/knime-edge-operator > edge-values.yaml
```

(Advanced) For a specific version, run:

```
helm show values knime-edge/knime-edge-operator --version <chart_version> > edge-
values.yaml
```

Now open the `edge-values.yaml` in a text editor and follow the instructions in the file: Comment strings in the `edge-values.yaml` file will specify what fields *must* be provided prior to installation and what values can be overridden for advanced configurations.

Installing the KNIME Edge helm chart

Once you have updated the values file with configuration relevant to your environment, you can install KNIME Edge to your Kubernetes cluster with the following command.

```
helm install [-n <namespace>] knime-edge knime-edge/knime-edge-operator -f edge-
values.yaml
```

(Advanced) If using a specific Chart version, install with:

```
helm install [-n <namespace>] knime-edge knime-edge/knime-edge-operator --version
<chart_version> -f edge-values.yaml
```

Installing KNIME Edge with KNIME Business Hub and Kurl

Kurl is the recommended installation method for KNIME Edge if you want to install KNIME Edge into an embedded Kubernetes cluster. It will set up a Kubernetes cluster and install KNIME Edge into it.

For the commands demonstrated below, replace anything shown in `<brackets>` with real values.

Connect to your host VM

The next step is to connect to your host VM and update it. If you are connecting via SSH, ensure that the machine you are using is permitted to connect to port 22 of the instance. Also ensure that the user you connect to the instance with has permissions to run commands as the superuser (i.e. `sudo`).

```
# Connect to your instance. This process/command may differ depending on OS.
ssh -i "some-identity-key.pem" ubuntu@<instance-ip-address>

# Update the instance. This process/command may differ depending on OS.
sudo apt-get update && sudo apt-get upgrade
```

Install the embedded cluster for KNIME Edge

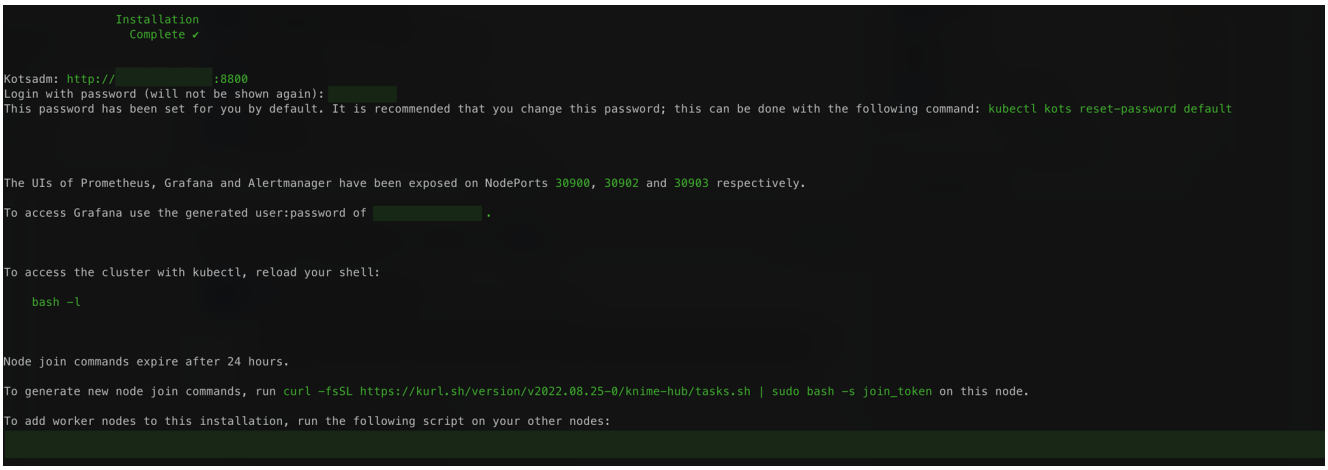
The command below executes a hands-free installation of all of the supporting Kubernetes architecture required to host KNIME Edge. It will take circa 10-15 minutes to run in its entirety and will output a significant amount of logs as the process installs all necessary dependencies.

```
curl -sSL https://kurl.sh/knime-edge | sudo bash
```

For more advanced installation options with kURL please consult the kURL documentation. Please note: if you execute this command with any additional flags or environment variables set then please note them down in a document. The same flags and environment variables

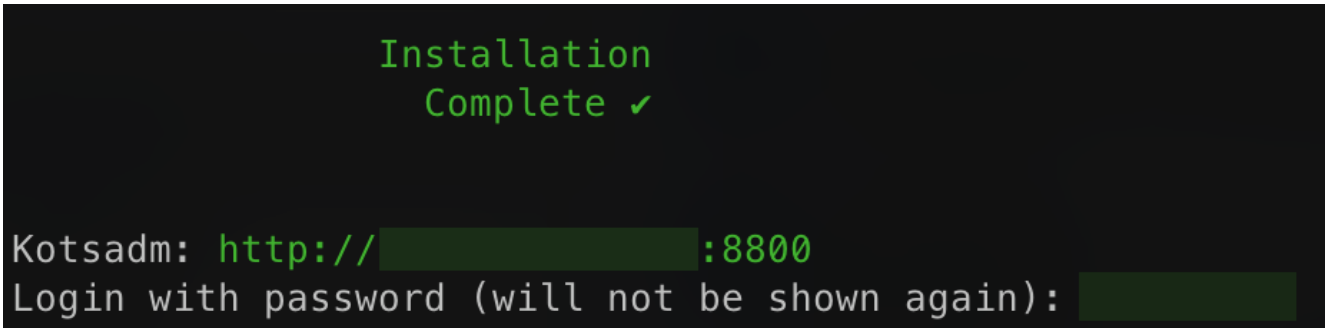
need to be present again when you update the kubernetes cluster version or KOTS Admin Console.

Once the process is complete, you should see something similar to the following output. This output contains very important URLs, usernames, passwords, and commands for your instance. Ensure that you save this output somewhere secure before proceeding.

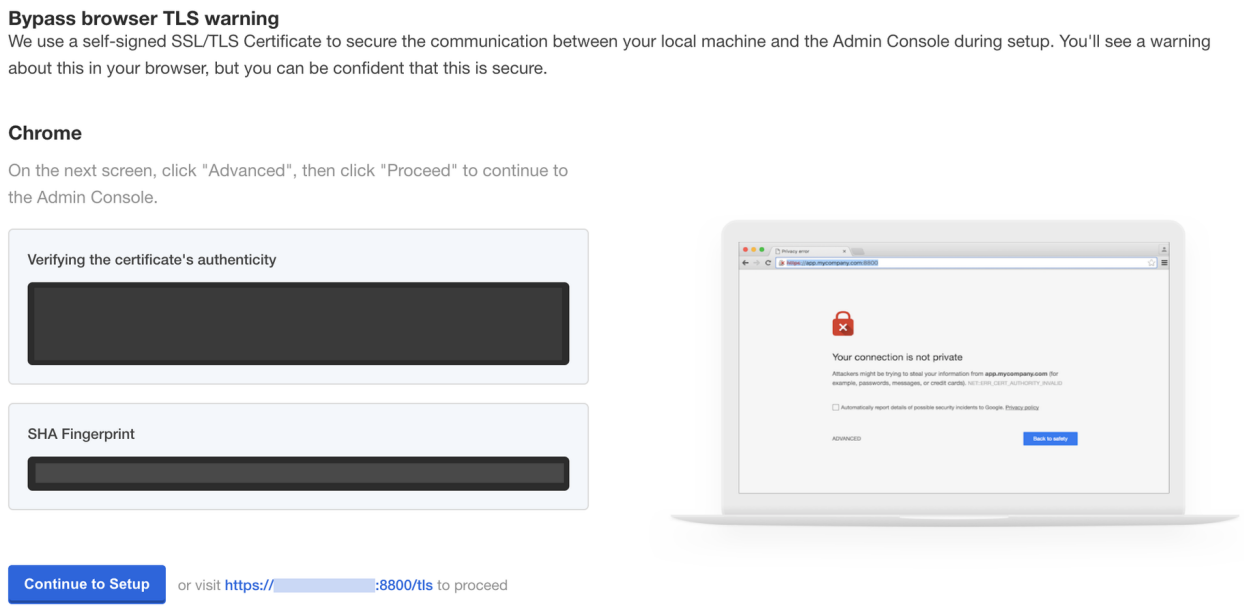


Access the KOTS Admin Console

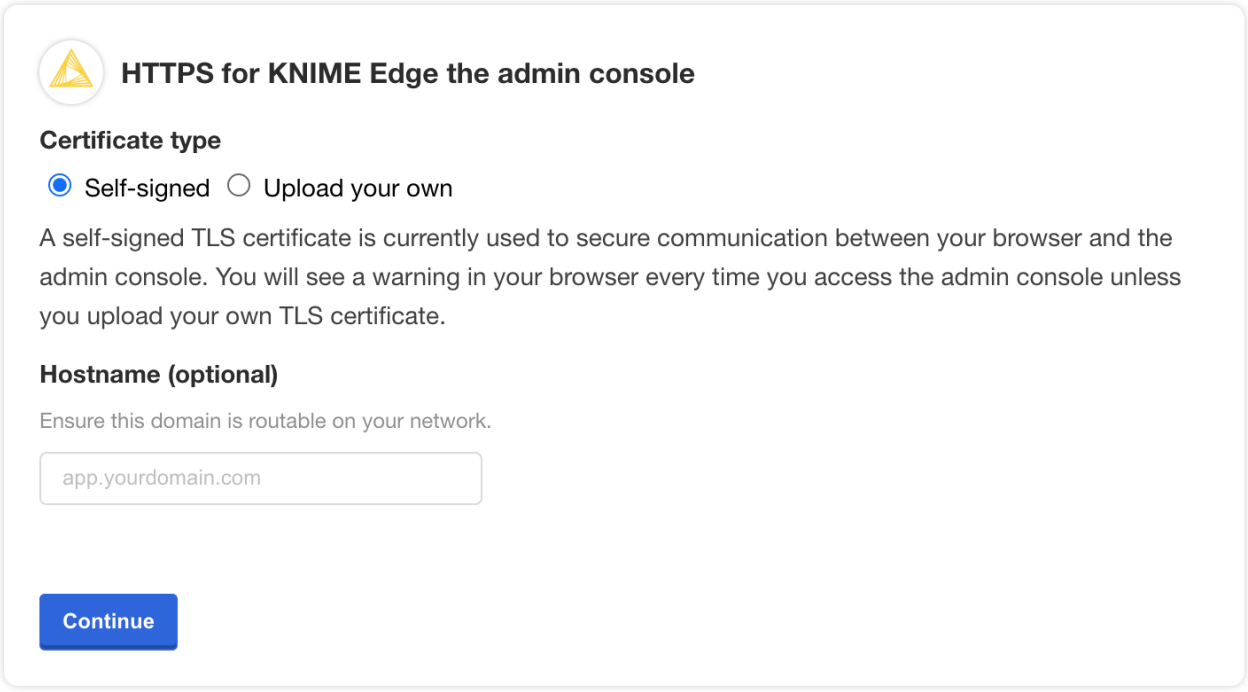
Navigate to the KOTS Admin Console URL provided in the embedded cluster installation output and take note of the password.



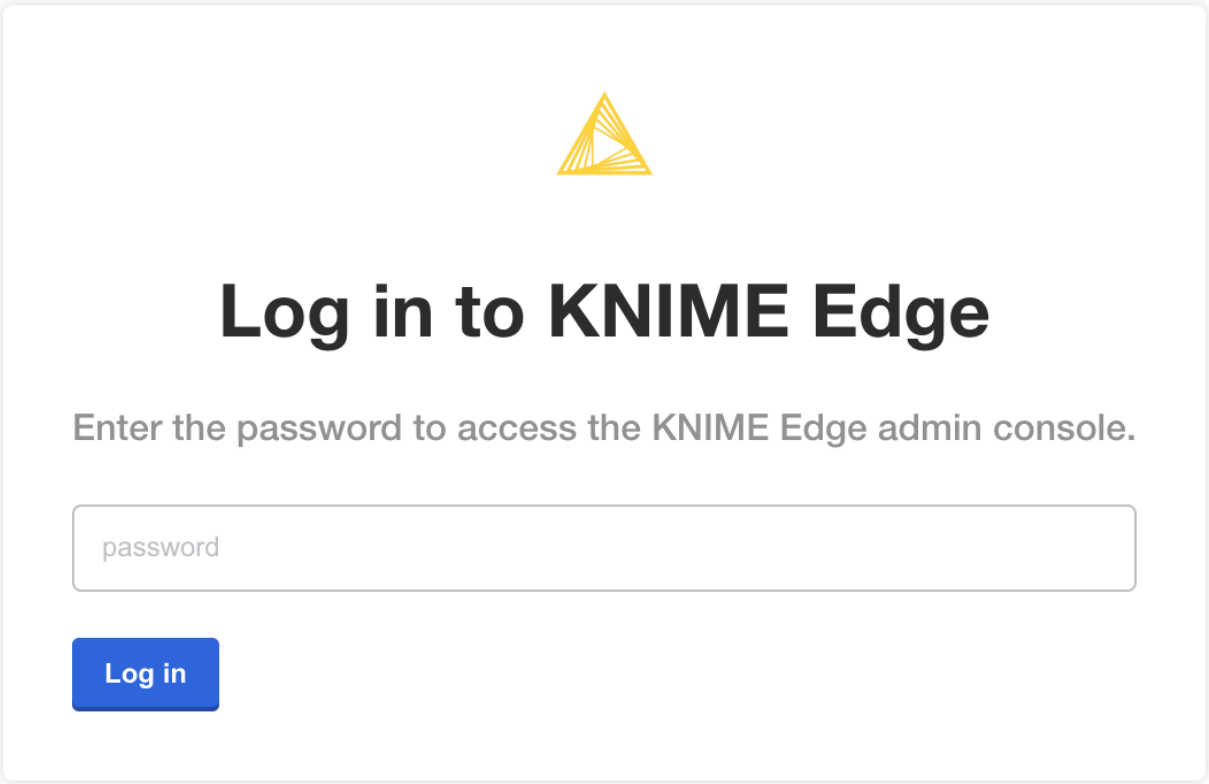
The first page that will display is a warning regarding Transport Layer Security (TLS) configuration. Follow the on-screen instructions to proceed.



You will then be prompted to provide your own TLS cert to secure traffic to the admin console, if desired.

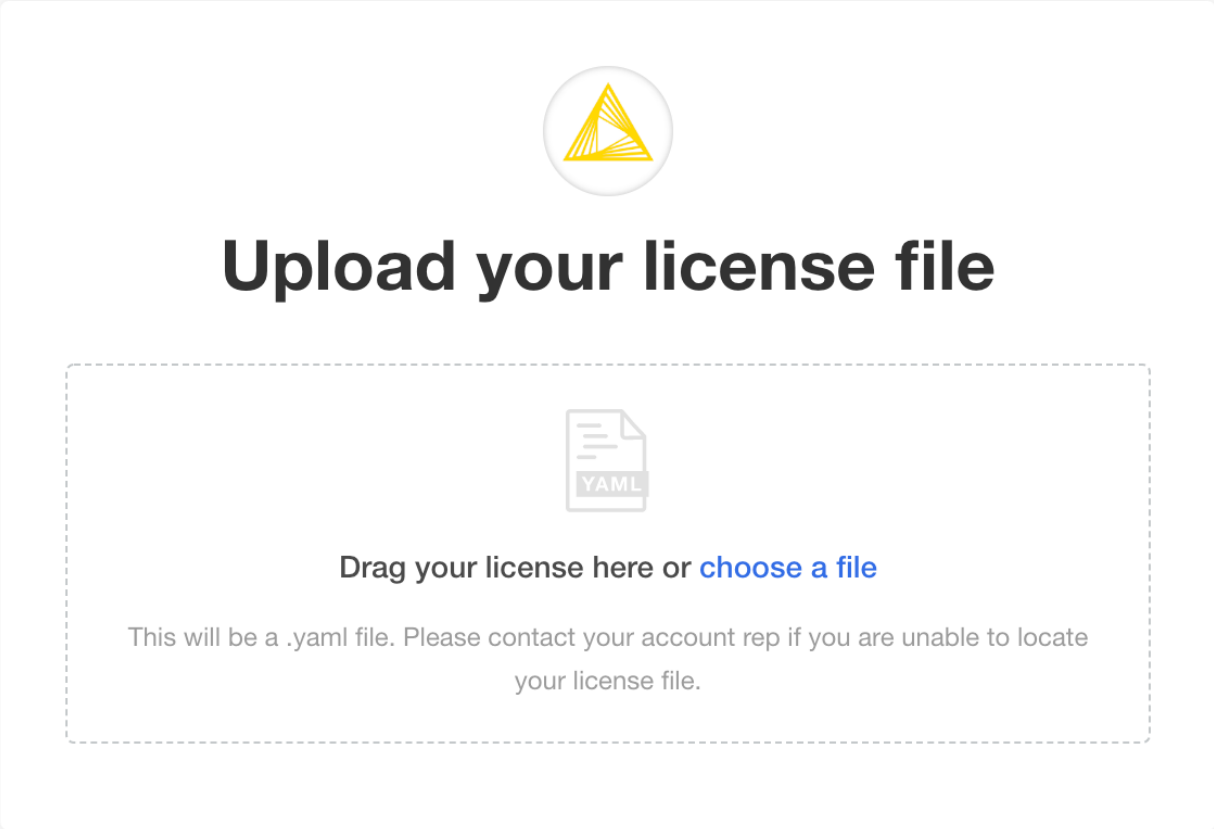


You should then see a prompt for a password. Enter the admin console password from the embedded cluster installation output to proceed (this password can be changed later).



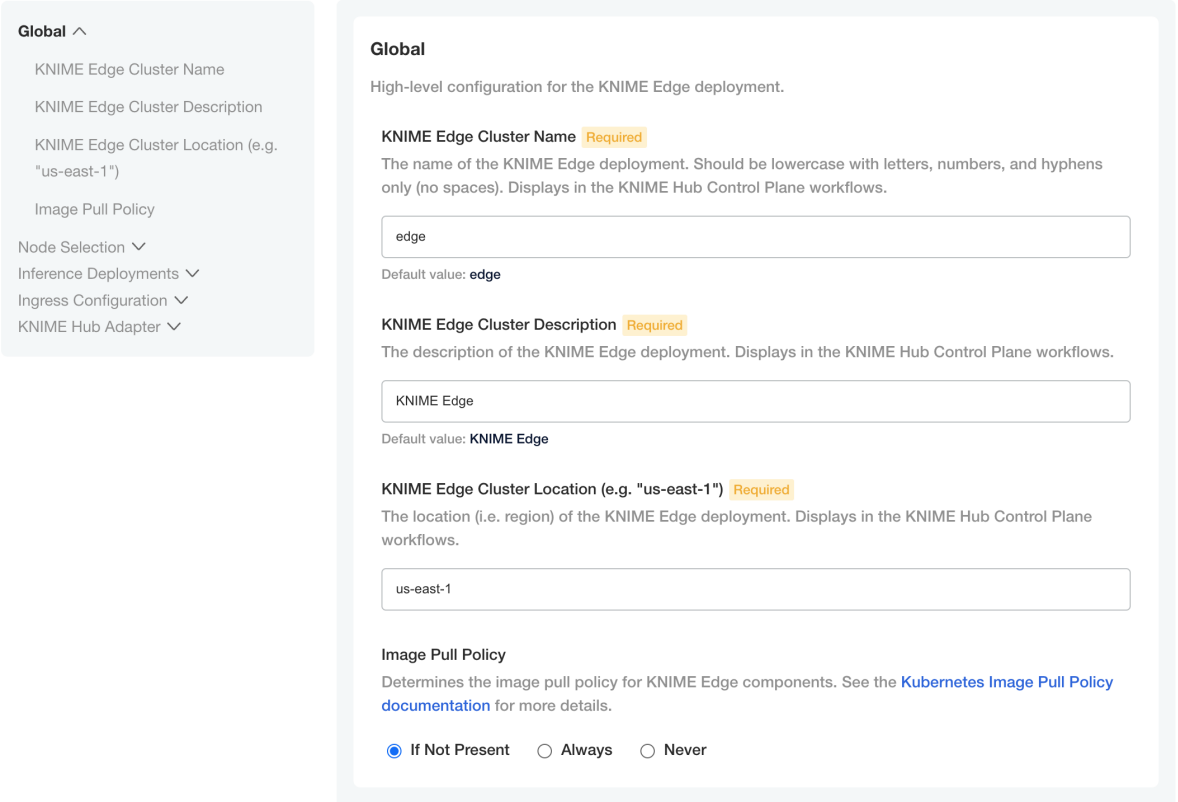
Provide a Replicated `.yaml` license file

After logging in, you should be prompted for a license file. This is the Replicated license file that your KNIME customer care representative has provided to you and has a `.yaml` extension. Please contact your customer care representative if you need assistance with your license.



Configure the installation

If all prior steps were successful, you should now be prompted to configure your KNIME Edge installation. A number of settings will display for you to customize.

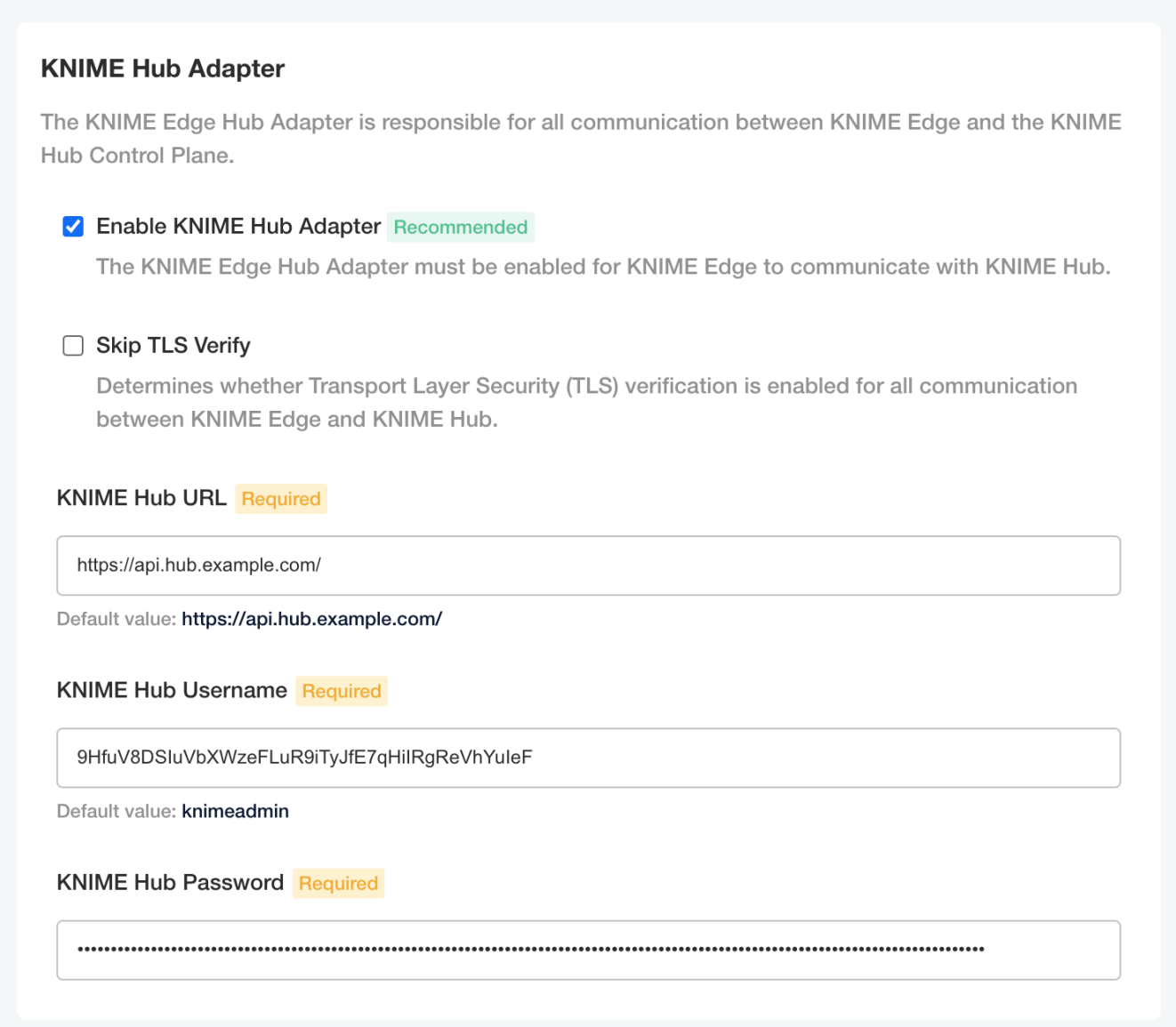


Please pay special attention to the **KNIME Hub Adapter** section, which contains configuration required for KNIME Edge to connect to the KNIME Business Hub control plane.

The KNIME Business Hub instance which is intended to be used as the control plane for KNIME Edge should have DNS configured and TLS enabled prior to installing KNIME Edge. Please note that KNIME Business Hub Standard is the minimum tier for KNIME Edge.

The KNIME Hub URL should be prefixed with `api.`, e.g. `https://api.hub.example.com/`. This is the address that KNIME Edge will use when communicating with the KNIME Business Hub API.

The username and password when connecting to KNIME Business Hub should be an application password generated from within the KNIME Business Hub. The user generating the app password should have admin access to the space in which KNIME Edge workflows are deployed. See the Application passwords section of the **KNIME Business Hub User Guide** for more information on generating app passwords.



Confirming a KNIME Edge cluster is operational

Verify Installation of KNIME Edge Cluster

Upon initial installation, the KNIME Edge Operator and KNIME Edge Hub Adapter pods should be up and running. The Kubectl CLI can be used to check the status (the pod ID will be different in your cluster):

```
> kubectl [-n namespace] get pod <edge-name>-knime-edge-operator-6574d6fd79-stlss

NAME                                              READY   STATUS    RESTARTS   AGE
# KNIME Edge Operator
<edge-name>-knime-edge-operator-6574d6fd79-stlss  1/1     Running   0          7m43s
```

```
> kubectl [-n namespace] get pod hub-adapter-697768bc8c-g9zjw

NAME                                          READY   STATUS    RESTARTS   AGE
# KNIME Edge Hub Adapter
hub-adapter-697768bc8c-g9zjw                  1/1     Running   0          7m40s
```

If the KNIME Edge Hub Adapter does not exist, there might be an issue in the KNIME Edge Operator. To inspect the logs of the KNIME Edge Operator pod, run:

```
> kubectl [-n namespace] logs <edge-name>-knime-edge-operator-6574d6fd79-stlss
```

Additionally to checking that the pods are running, querying the endpoint should return an empty list for `inferenceDeployments`:

```
% curl -sL http://localhost:8081/ | jq
{
  "inferenceDeployments": []
}
```

The KNIME Edge Operator creates additional pods in the KNIME Edge cluster which should all be either in the Running or Completed state. For more information on these pods, see the section below.

Interpreting the pods in the cluster

The Kubectl CLI can be used to investigate the pods in a KNIME Edge cluster. Below is sample output from the `kubectl [-n <namespace>] get pods` command (where `-n <namespace>` defines the namespace) with comments added to help identify the various pods and their roles:

```
> kubectl [-n <namespace>] get pods

NAME                                          READY   STATUS    RESTARTS   AGE

# Registers Inference Deployments as Kubernetes Endpoints
endpoint-discovery-depl-6457b745b7-pkb24      1/1     Running   0          7m40s

# Logging Service
fluentd-ds-6q8l6                              1/1     Running   0          7m41s
fluentd-ds-562zj                              1/1     Running   0          7m41s

# KNIME Edge Operator
knime-edge-knime-edge-operator-6574d6fd79-stlss   1/1     Running   0          25m

# Kong
knime-edge-kong-5bb658458f-znwsg              2/2     Running   0          25m

# Workflow and Log Cache
minio-b9b9547dd-xbvs2                         1/1     Running   0          7m41s

# Inference Deployment
sentiment-predictor-7d56b6cd47-lp8gp          1/1     Running   0          6m31s

# KNIME Edge Hub Adapter
hub-adapter-697768bc8c-g9zjw                  1/1     Running   0          7m40s

# Kong Gateway Loadbalancer
svclb-knime-edge-kong-proxy-bgqzc             2/2     Running   0          25m
svclb-knime-edge-kong-proxy-jg8fg             2/2     Running   0          25m

# Started by a cronjob to delete old logs
minio-log-cleaner-cronjob-1632700800-ssqsx    0/1     Completed 0          19m
```

Testing the KNIME Edge Hub Adapter

The Hub Adapter is an essential piece in the KNIME Edge architecture as it communicates with KNIME Hub. If, e.g., Inference Deployments created in the control plane on KNIME Hub are not showing up in the KNIME Edge cluster, a not properly working Hub Adapter might be the issue.

First, test if the Hub Adapter pod is running properly (the pod ID will be different in your cluster):

```
> kubectl [-n namespace] get pod hub-adapter-697768bc8c-g9zjw
```

It should return the status "Running".

```
NAME                              READY   STATUS    RESTARTS   AGE
hub-adapter-697768bc8c-g9zjw   1/1     Running   0          8m2s
```

To inspect the recent logs of the Hub Adapter pod, run:

```
> kubectl [-n namespace] logs --since=30s hub-adapter-697768bc8c-g9zjw
```

There are a few things to look for in the logs:

1. A line that says "Registration accepted" tells you that the KNIME Edge license is valid and set up correctly. Only if KNIME Hub accepts the registration, the Hub Adapter is able to pull information about Inference Deployments from KNIME Hub.
2. The Hub Adapter makes requests to different API endpoints on KNIME Hub. For each request, it logs the HTTP response status. a) 200: the request was successful. b) 404: the endpoint was not found. Make sure the KNIME Edge Service on KNIME Hub is running properly. See the Configure KNIME Business Hub section. c) 5xx: server error. Make sure the KNIME Edge Service on KNIME Hub is running properly. See the Configure KNIME Business Hub section.

Advanced Operations and Troubleshooting

Error: Failed to pull image

Error Signature

```
Failed to pull image "registry.hub.knime.com/knime-edge/knime-edge-operator:0.0.1-beta-
20210802-161605-000061-4fa252d":\
 rpc error: code = Unknown desc = failed to pull and unpack image
"registry.hub.knime.com/knime-edge/knime-edge-operator:0.0.1-beta-20210802-161605-
000061-4fa252d":\
 failed to resolve reference "registry.hub.knime.com/knime-edge/knime-edge-
operator:0.0.1-beta-20210802-161605-000061-4fa252d":\
 unexpected status code [manifests 0.0.1-beta-20210802-161605-000061-4fa252d]: 401
Unauthorized
```

Interpretation:

- 401 Unauthorized
- image URI
  `registry.hub.knime.com/knime-edge/knime-edge-operator:0.0.1-beta-20210802-`

`161605-000061-4fa252d`

- web URL https://registry.hub.knime.com/harbor/projects/2/repositories → `knime-edge` → `knime-edge-operator` → search for tag `0.0.1-beta-20210802-161605-000061-4fa252d`

1. Log in to KNIME Artifact Registry

Log in to https://registry.hub.knime.com to refresh the validity of the access token.

2. Verify that the secrets are configured correctly

2a. Verify that the `imagePullSecrets` are configured

Look in deployment for imagePullSecrets; verify that `regcred` is present:

```
 % kubectl [-n <namespace>] get deployments knime-edge-knime-edge-operator -o json | jq
'.spec.template.spec.imagePullSecrets'
[
  {
    "name": "regcred"
  }
]
```

2b. Verify the secret exists in the correct namespace

```
% kubectl [-n <namespace>] get secret regcred
NAME      TYPE                             DATA   AGE
regcred   kubernetes.io/dockerconfigjson   1      50m
```

2c. Verify the value of the secret

Warning: Executing the command below prints the secret value to stdout

Examine Secret values:

```
% kubectl [-n <namespace>] get secret regcred -o json | jq -r
'.data.".dockerconfigjson"' | base64 -d | jq
{
  "auths": {
    "registry.hub.knime.com": {
      "username": "<harbor_user>",
      "password": "<harbor_secret>",
      "email": "<harbor_email>",
      "auth": "<auth>"
    }
  }
}
```

(Advanced) Examine Secret values without printing to standard out:

```
% cd ~/.shh
% diff <(kubectl [-n <namespace>] get secret regcred -o json | jq -r
'.data.".dockerconfigjson"' | base64 -d | jq -r '.auths[].password') <(cat
./harbor_secret); echo $?
0
```

Error: Unable to Access Host

Possible Causes:

- Misconfigured Ingress, i.e. Kong was not enabled or Traefik was left enabled

Error Signature: HTTP - Empty reply from server

```
% curl -L http://localhost:8081/
curl: (52) Empty reply from server
```

For reference, a completely misconfigured cluster would return: `curl: (7) Failed to connect to localhost port 8081: Connection refused`

1. Check the Kubernetes Cluster Networking settings

```
kubectl [-n <namespace>] get ingress
kubectl [-n <namespace>] get services
kubectl [-n <namespace>] get endpoints
```

2. Check the Kong Ingress Controller settings

Correct ingress configuration requires a helm install with `kong.enabled: true`.

Check for the following in the `values.yaml` used to deploy the cluster (or explore Kubernetes configuration to determine whether the value was set):

```
...
kong.enabled:
  true
```

Alternatively, check the manifest used by Helm at cluster install time:

```
helm status -o yaml [-n <namespace>] knime-edge | less
```

Resolution:

If necessary, set to `true` with:

```
helm upgrade [-n <namespace>] knime-edge knime-edge/knime-edge-operator --version
<chart_version> --reuse-values --set kong.enabled=true
```

Upgrading an existing KNIME Edge chart

To update to a new chart version, follow the instructions for obtaining an updated values file and then run:

```
helm upgrade [-n <namespace>] knime-edge knime-edge/knime-edge-operator --version
<chart_version> -f edge-values.yaml
```

If any errors occur, refer to the advanced troubleshooting section in this guide.

Error: Helm Upgrade is invalid due to Required Value

Error Signature: `INSTALLATION FAILED: EdgeDeployment.edge.knime.com "edge" is invalid: ⋯ Required value`

```
% helm upgrade [-n <namespace] edge knime-edge/knime-edge-operator --version
<chart_version> -f edge-values.yaml
...
    Error: INSTALLATION FAILED: EdgeDeployment.edge.knime.com "edge" is invalid: \
    [spec.knimeServerDeleteLogRequestWF: Required value, \
    spec.knimeServerDeployWF: Required value, \
    spec.knimeServerGetLogRequestsWF: Required value, \
    spec.knimeServerPushMetricsWF: Required value, \
    spec.knimeServerRegisterWF: Required value, \
    spec.knimeServerUploadLogFileWF: Required value]
```

Interpretation: These are all keys which are required by the currently installed CRDs which are used by the operator. The following questions explore potential root causes.

1. Are these values new in this version? Check the Chart

It is possible that a newer version may have updated values; check for this by following the steps to obtain a new copy of the `values.yaml` file and searching it for the mentioned values.

2. Are these values not present in the Chart for this Version? Delete the CRDs

The following command will remove all KNIME Edge CRDs:

```
kubectl delete customresourcedefinition $(kubectl get customresourcedefinition
-o=jsonpath='{.items[?(@.spec.group=="edge.knime.com")].metadata.name}')
```

Explanation: Helm deliberately will not modify CRDs, so if the required values change between versions, Helm cannot automatically manage the update.

KNIME AG
Talacker 50
8001 Zurich, Switzerland
www.knime.com
info@knime.com