KNIME Business Hub Installation Guide - Embedded Cluster

KNIME AG, Zurich, Switzerland

Version 1.14 (last updated on)

Table of Contents

Introduction

KNIME Business Hub is a customer-managed KNIME Hub instance.

Once you have a license for it and proceed with installation you will have access to Hub resources and will be able to customize specific features, as well as give access to these resources to your employees, organize them into Teams and give them the ability to manage specific resources.

Once you have access to a KNIME Business Hub instance available at your company, you can use KNIME Business Hub to perform a number of tasks such as:

- collaborate with your colleagues,

- test execution of workflows,

- create and share data apps, schedules, and API services

- keep track of changes with versioning.

The following is a guide for installing KNIME Business Hub into an embedded cluster.

To administrate a KNIME Business Hub instance please refer instead to the following guide:

- KNIME Business Hub Administration Guide

Installation planning

KNIME Business Hub supports the following types of installations:

- **Installation by downloading container images from public image registries:** This type of installation requires an environment with externally accessible networks. This is a standard installation process and is documented in the KNIME Business Hub installation section. Just follow the steps here, after making sure you fulfill the prerequisites in the next section.

- **Airgapped installation:** This type of installation is required for installing into air gap environments, or into an environment that, even though has a network with external access, is not allowed to download container images from public image registries. If your environment does not have access to external networks you can follow the steps in the airgapped installation section.

Kubernetes

Since KNIME Business Hub is a product based on microservices that are orchestrated and managed via Kubernetes, a Kubernetes cluster is needed when installing KNIME Business Hub.

KNIME Business Hub 1.14 supports the following versions of Kubernetes:

- 1.25
- 1.26
- 1.27
- 1.28
- 1.29
- 1.30

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 Business Hub.

System prerequisites

This guide covers the installation of KNIME Business Hub.

The following **Operating Systems** are supported:

- Ubuntu Server 20.04 LTS (Requires Docker version >= 19.03.10)
- Ubuntu Server 22.04 LTS (Requires Containerd version >= 1.5.10 or Docker version >= 20.10.17)
- Ubuntu Server 24.04 LTS (Requires Containerd package preinstalled on the host)
- RHEL 8.6, 8.8, 8.9, 8.10, 9.0, 9.2, 9.3, 9.4 (RHEL 8.x and 9.x require Containerd)

> ℹ️ For KNIME Business Hub version 1.13 or above installed on RHEL OS, **before updating enable the *RHEL Support* flag** in the *Advanced Postgres* configuration option in KOTS Admin Console as specified in this section of the KNIME Business Hub Update Guide.

- Amazon Linux 2

Note that only x86, x64 and x86_64 processors are supported.

Host package requirements

Host packages are bundled and installed by kURL without the need for external package repositories.

Some OS require additional packages. Find a list of the additional packages or steps in the following sections.

> ℹ️ Make sure to lock the packages versions **after** the installation is completed so that they are not updated automatically. This could break dependencies and bring the Hub instance in a unhealthy state. When performing an update, you will need to unlock the packages and lock them again when the deployment is terminated. Refer to OS documentation to learn how to lock packages.

> ℹ️ For installations via AWS please refer to the KNIME Business Hub on AWS Marketplace guide, for installations via Azure please refer to the KNIME Business Hub on Azure Marketplace guide.

RHEL 9 and Ubuntu Server 24.04 LTS

An exception exists for Red Hat Enterprise Linux (RHEL) 9 and Ubuntu Server 24.04 LTS.

For these OSs, the following packages are required:

Table 1. Add-on and relative packages required for RHEL 9

Add-on	Packages
* kURL Core	curl openssl tar fio
Collectd	bash glibc libcurl libcurl-minimal libgcrypt libgpg-error libmnl openssl-libs rrdtool systemd systemd-libs yajl
Containerd	container-selinux bash libseccomp libzstd systemd
Kubernetes	conntrack-tools ethtool glibc iproute iptables-nft socat util-linux
Longhorn	iscsi-initiator-utils nfs-utils
OpenEBS \versions 1.x and 2.x*	iscsi-initiator-utils
Rook	lvm2
Velero	nfs-utils

Additional steps for Ubuntu Server 24.04 LTS

Additionally to the above, the `containerd` package is required for Containerd add-on installations on Ubuntu 24.04.

To install on Ubuntu Server 24.04 LTS run these commands beforehand:

```
sudo apt update
sudo apt install socat -y
sudo apt install ebtables -y
sudo apt install conntrack -y
sudo apt install nfs-common -y
sudo apt install containerd -y
sudo apt install fio -y
```

Environments with restricted third-party package installation

For environments where third-party package installation is restricted due to policies, kURL offers a `skip-system-package-install` flag. See documentation here.

This flag allows for manual dependency management, bypassing automatic installations. If you use this flag you must ensure all necessary dependencies are pre-installed.

To identify these dependencies, perform a standard installation without the flag enabled and review the installation logs under `/var/log/kurl`, noting the packages listed there.

Hardware prerequisites

The following sections cover the prerequisities for single node and multi node installations.

Here are some recurrent terms used in the next sections:

- **Hub core:** Refers to Hub core services that are responsible for all functionalities of Business Hub besides execution. Including authentication services, UI, database services, etc.

- **Hub execution:** Refers to executor resources, e.g. executor pod resources.

- **Disk:** The attached storage size requirement for the persistent components of Business Hub. Used to provide storage to databases, store data files for KNIME workflows, etc.

> ℹ️ The CPU and Memory requirement amounts shown in the following chapters of this document refer to the "Total Capacity" of the nodes. Hence not all of these resources are allocatable for Kubernetes workloads, some needs to be available or reserved for system-related-services and should not be allocated, or reserved for Kubernetes related workloads. By default usually only some Memory is reserved from the total capacity, therefore if there are no reservation for CPU vCores, it is advised to either reserve or leave some vCores un-reserved by Kubernetes workloads. The reserved Memory for system-related-services is usually 100 MB by default for Kurl installations.

> ℹ️ The "vCore" term used in this guide stands for virtual cores. They are a measure of the processing power allocated to a virtual machine (VM) or a computing instance in cloud computing environments. KNIME makes no representation about how the vCores licensed for execution are correlated or mapped onto the physical system where the software is running. The same principle also extends to the vCores necessary for operating core services.

> ℹ️ The **CPU** must be **from 2011 or newer**, as older processors may lack required instruction sets such as SSE4.2 (needed by Keycloak) and AVX (needed by the event-dispatcher).

Single node installation

Here we provide recommended and minimal installation requirements for single node installations with both Basic and Standard plans, and recommended installation requirements for Enterprise plans.

KNIME's advice is to strive to meet the **recommended system requirements** for better performance and more scalability.

Find the advanced install option for installation on host machines with smaller sized root volumes in the Installation on hosts with undersized root volumes section.

Basic Plan

Using a **Basic** license automatically provides **4 vCores** for execution. Extra execution vCores can be purchased, but in such cases, the available vCores for the system must be calculated accordingly.

Recommended

- CPU: 16+ vCores total
  - Hub Core: 10 vCores
  - Hub Execution: 4 vCores (included in the license)
- Memory: 32 GB+
  - Hub Core: 20 GB
  - Hub Execution: 12 GB+ available memory to allocate for executors
- Disk: 500 GB+

Minimal

- CPU: 12 vCores total
  - Hub Core: 7 vCores

- ◦ Hub Execution: 4 vCores (included in the license)
- • Memory: 32 GB
    - ◦ Hub Core: 20 GB
    - ◦ Hub Execution: 12 GB available memory to allocate for executors
- • Disk: 500 GB

Examples for setting up executors based on available resources with Basic license:

> *Limitations:*
>
> - • *at least 1 vCore per executor*
> - • *at least 2 GB Memory per executor*

ℹ️   Allocatable RAM for executors in case of a 32 GB Ram installation is **12 GB**.

Examples with 4 execution vCores included in the Basic license for an installation with recommended system requirements:

- • Scenario 1:
    - ◦ 4 executors. Each executor running on 1 vCore and 3 GB of Memory.
- • Scenario 2:
    - ◦ 1 executor. Running with 4 vCores and 12 GB of Memory.

Standard Plan

Using a **Standard** license automatically provides **8 vCores** for execution. Extra execution vCores can be purchased, but in such cases, the available CPU cores for the system must be calculated accordingly.

Recommended

- • CPU: 20+ vCores total
    - ◦ Hub Core: 10 vCores
    - ◦ Hub Execution: 8 vCores (included in the license)
- • Memory: 64 GB+
    - ◦ Hub Core: 20 GB

- ◦ Hub Execution: 44 GB+ available memory to be allocated for executors
- Disk: 500GB+

Minimal

- CPU: 16 vCores total
  - ◦ Hub Core: 7 vCores
  - ◦ Hub Execution: 8 vCores (included in the license)
- Memory: 36 GB
  - ◦ Hub Core: 20 GB
  - ◦ Hub Execution: 16 GB available memory to be allocated for executors
- Disk: 500GB

Examples for setting up executors based on available resources with Standard license:

Limitations:

- *at least 1 vCore per executor*
- *at least 2 GB Memory per executor*

ℹ️ Allocatable RAM for executors in case of a 64 GB Ram installation is **44 GB**.

Examples with 8 execution vCores included in the Standard license for an installation with recommended resource requirements:

- Scenario 1:
  - ◦ 8 executors. Each executor running on 1 vCore and 2+ GB of Memory. (total of 44 GB RAM can be shared amongst executors)
- Scenario 2:
  - ◦ 2 executors. Each executor running with 4 vCores and 2+ GB of Memory. (total of 44 GB RAM can be shared amongst executors)
- Scenario 3:
  - ◦ 1 executor. Running with 8 vCores and 2+ GB of Memory. (total of 44 GB RAM can be shared amongst executors)

Enterprise Plan

Using a **Enterprise** license provides the possibility to set the desired amount of vCores.

Recommended

- CPU: 26+ vCores total
  - Hub Core: 10 vCores
  - Hub Execution: 16 vCores
- Memory: 64 GB+
  - Hub Core: 20 GB
  - Hub Execution: 44 GB+ available memory to be allocated for executors
- Disk: 500GB+

Examples for setting up executors based on available resources with Enterprise license:

Limitations:

- *at least 1 vCore per executor*
- *at least 2 GB Memory per executor*

> ℹ️ Allocatable RAM for executors in case of a 64 GB Ram installation is **44 GB**.

Examples with 16 execution vCores for an installation with recommended resource requirements:

- Scenario 1:
  - 8 executors. Each executor running with 2 vCore and 2 GB or more Memory. (total of 44 GB RAM can be shared amongst executors)
- Scenario 2:
  - 4 executors. Each executor running with 4 vCores and 2 GB or more Memory. (total of 44 GB RAM can be shared amongst executors)
- Scenario 3:
  - 2 executors. Each executor running with 8 vCores and 2 GB or more Memory. (total of 44 GB RAM can be shared amongst executors)

Multi-node installation

Here we provide installation requirements for multi node installations.

See the advanced install options section for configuring highly available clusters and installing on instances with smaller sized root volumes.

Also we distinguish two cases based on whether or not `nodeAffinity` and `nodeSelection` is used.

Read more about `nodeAffinity` and `nodeSelection` in the KNIME Business Hub Admin Guide.

Highly-available Multinode installation without `nodeAffinity` and `nodeSelection`

- Node count: 3+ (Shared between Hub Core services and Execution services)
- The cluster needs to have at least **three** nodes with at least:
    - 16 vCores and
    - 32 GB RAM per instance
- Disk: 100GB+ per instance for the root volume
- Additional Attached Disks: 1 or more additional, attached, unformatted disks are required for multinode installations to handle data replication between nodes
- See the advanced install options for configuring highly available clusters and installing on instances with smaller sized root volumes

Example

3 nodes. 16 vCores and 32 GB RAM each. (Shared between Hub Core services and Execution services)

Running the installation on the above resources will allow the user to allocate 8 vCores and 16 GB RAM for execution.

Highly-available Multinode installation with `nodeAffinity` and `nodeSelection`

- Node count: 3+ (Allocated for Hub Core services only)
- For Hub Core workloads the cluster needs to have at least **three** nodes with at least:
    - 12 vCores and
    - 32 GB RAM per instance.

- For execution workloads: additional nodes need to be added to the cluster depending on the performance needs for execution.

- Disk: 100GB+ per instance for the root volume

- Additional Attached Disks: 1 or more additional, attached, unformatted disks are required for multinode installations to handle data replication between nodes

- See the advanced install options for configuring highly available clusters and installing on instances with smaller sized root volumes

Example

3 nodes for Hub Core services.

1 node for Execution services.

3 nodes with nodeAffinity or nodeSelection set to "core", 12 vCores each, with 32 GB RAM.

At least 1 node with nodeAffinity or nodeSelection set to "execution",

- 8 vCores and 32 GB RAM (Basic)

- 12 vCores and 32 GB RAM (Standard)

- 16+ vCores and 64+ GB RAM (Enterprise)

> **i** Nodes annotated with nodeAffinity or nodeSelection set to "execution" can use approximately 75% of the total allocatable resources for execution resources, due to some resource being allocated to system-related-services, some to default Kubernetes workloads running on the node and some to certain kurl add-ons, achieving high availability.

Networking prerequisites

Network Ports:

- 80 (HTTP)
- 443 (HTTPS)
- 22 (SSH) ADMIN USE ONLY
- 6443 (kubectl) ADMIN USE ONLY
- 8800 (KOTS Admin Console) ADMIN USE ONLY

Security Warnings:

- Ports 22, 6443, and 8800 are vulnerable access points for a KNIME Hub installation. If a malicious actor gained access to any of those ports, they would be able to perform destructive actions on the cluster and/or gain access to sensitive configuration. Access to these ports must be restricted to only the IP address(es) of the machine(s) which will administer the installation.
- Security-Enhanced Linux (SELinux) is not currently supported. If enabled, the installer script will notify the user via a prompt and disable SELinux before proceeding.

The following domains need to be accessible from servers performing online installations:

Trusted Host	Domain
KNIME	`*.knime.com`
Replicated	See the Firewall Openings for Online Installations guide.

Also, in order to be able to pull Docker images the following Docker registries need to be accessible from servers performing online installations:

```
docker.elastic.co
docker.io
ghcr.io
quay.io
registry.k8s.io (for this domain image layers are pulled from the domain below)
*.pkg.dev
registry.opensource.zalan.do
proxy.replicated.com
cr.fluentbit.io
```

i Some features in KNIME Business Hub, e.g. the job viewer, use websockets. If an external proxy or load balancer is used it needs to be websocket compatible.

KNIME Business Hub installation

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

> **i** If your environment does not have access to external networks you can follow the steps in the airgapped installation section.

Connect to your host VM

The first 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
```

Increase the limit for allowed open files

In single node installations the limit for allowed open files needs to be increased on the host VM for Business Hub to function properly.

We recommend to increase the limit by running the following command on the host VM.

```
sudo sysctl fs.inotify.max_user_instances=8192
```

This value needs to be set again after VM restarts. Alternatively you can persist it by adding it to `/etc/sysctl.conf`.

If the limit was not increased you might encounter the following error, appearing after the Business Hub installation is completed. In this case the `istio-proxy` container in the `keycloak-proxy` and `s3-proxy` pods isn't able to start up with a "too many open files" error.

An example error log can be seen below. Replace `keycloak-proxy-7ff44f845f-jrh9s` with an appropriate pod id in your cluster.

```
$ kubectl logs -n hub keycloak-proxy-7ff44f845f-jrh9s -c istio-proxy
{"level":"fatal","time":"2023-03-01T22:07:13.738069Z","msg":"Agent start up errorfailed
to start workload secret manager too many open files"}
```

Install the embedded cluster for KNIME Business Hub

The command below executes a hands-free installation of all of the supporting Kubernetes architecture required to host KNIME Business Hub. 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-hub | 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.

> **i** During the installation of the Business Hub namespaces are created automatically. For example, the namespace where Hub is installed is the `knime` namespace by default in kURL installations. We will refer to it with the placeholder `<business-hub-namespace>` in the KNIME Business Hub documentation.

Once the process is complete, you should see something similar to the following output. The output will contain the KOTS Admin Console URL and password.

> **i** This output contains very important URLs, usernames, passwords, and commands for your instance. Ensure that you save this output somewhere secure before proceeding.

Next steps in the installation

Once the following supplemental steps above have been completed, you should be able to proceed with the remainder of the install.

The high-level remaining steps include:

- Logging into the KOTS Admin UI
- Uploading the Replicated license which will fetch the latest release of Business Hub
- Enter appropriate configuration parameters in the KOTS Admin Console
- Save the changes, allow the pre-flight checks to run, then click **Deploy** and Wait for the installation to complete.

Access the KOTS Admin Console

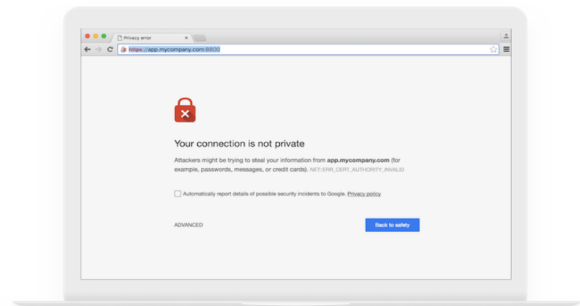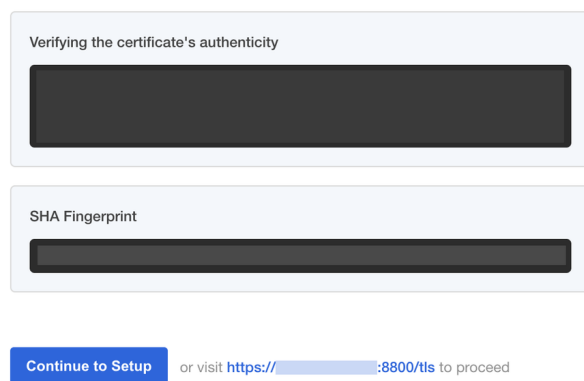Navigate to the KOTS Admin Console URL and take note of the password.
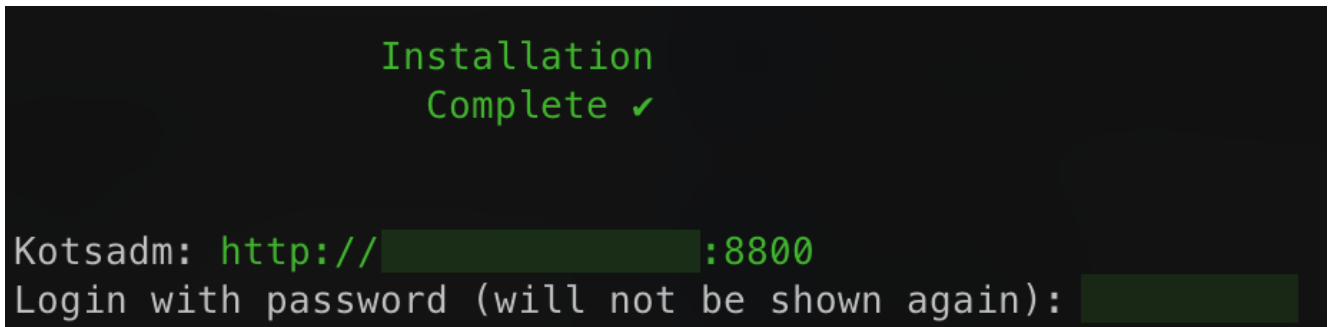
The first page that will display is a warning regarding Transport Layer Security (TLS) configuration. Follow the on-screen instructions to proceed.

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

> **i** For everything to work correctly and securely, setting up TLS is strongly recommended.

You should then see a prompt for a password. Enter the admin console password 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.

You may be prompted to install KNIME Business Hub into an "airgapped environment" if airgap installations are enabled on your license. This feature is particularly useful for installing KNIME Hub onto a machine that has no outbound internet access. Follow the instructions in the KNIME Business Hub airgapped installation section.

In most cases, it is fine to click the **Download KNIME Hub from the Internet** button if this prompt appears.

Install in airgapped environment

To install on an airgapped network, the images in KNIME Hub will be uploaded from the bundle you provide to the cluster.

Drag your airgap bundle here or choose a bundle to upload

This will be a .airgap file KNIME Hub provided. Please contact your account rep if you are unable to locate your .airgap file.

Optionally you can download KNIME Hub from the Internet

Configure the installation

If all prior steps were successful, you should now be prompted to configure your KNIME Business Hub installation. A number of settings will display for you to customize. Please note that all configuration settings in this view can be changed post-installation, except for the settings under "Initialization of KNIME Business Hub".

Global overview

Global

High-level settings for the KNIME Business Hub deployment.

KNIME Business Hub Deployment Name `Required`
The KNIME Business Hub deployment name is displayed in several places in the web UI. The value may contain spaces.

> KNIME Business Hub

Default value: **KNIME Business Hub**

KNIME Business Hub Mountpoint ID `Required`
Name of the Mountpoint ID when adding this KNIME Business Hub deployment in the KNIME AP. The value may not contain spaces (it is recommended to use dashes `-` instead).

> knime-business-hub

Default value: **knime-business-hub**

KNIME Business Hub License `Required`
Upload your KNIME Business Hub License `.xml` file. You have received this file from your account manager, in addition to the Replicated License `.yaml` file that you used during a previous step. Contact your KNIME account manager if you have not yet received a `.xml` file containing the KNIME Business Hub License.

Upload a file

> ○ Browse files for KNIME Business Hub License

☐ View Advanced Settings

- KNIME Business Hub Deployment Name - This will be the name of your deployment for KNIME Business Hub

- KNIME Business Hub Mountpoint ID - This is where your KNIME Business Hub Mountpoint will be.

- KNIME Business Hub License - You should have received this from your account manager. This is a different file than the Replicated `.yaml` license file. The KNIME

Business Hub license file is a `.xml` file that contains information needed for your installation. Upload it here by clicking *Browse files for KNIME Business Hub License* and navigating to the file on your local machine.

- (OPTIONAL) select *View Advanced Settings*: - This will add additional fields underneath the *Notifications* section. This are explained in the *Advanced* options section.

Initialization of KNIME Business Hub

During the very first installation of KNIME Business Hub a number of one-time initializations are made, like creating an admin user, team, space, and execution context. Changing fields in this section after installation won't have any effect on the deployed application. The admin user can change these after the installation in the browser.

Initialization of KNIME Business Hub

This section contains configuration options that are applied during the initial installation of KNIME Business Hub. These options will be hidden after the installation has completed, and any future changes will require an administrator to manually edit the affected resource(s).

KNIME Business Hub Admin Username `Required`
The username for the admin user of KNIME Business Hub.

> knimeadmin

Default value: **knimeadmin**

KNIME Business Hub Admin Password `Required`
The initial password for the admin user. This password can be changed in Keycloak post-installation.

> 👁️

Initial Team Name
The name for the initial KNIME Business Hub Team.

> Initial Team

Initial Team Description
The description for the initial KNIME Business Hub Team.

> Initial Team created by KNIME

Initial Space Name
The name of the initial KNIME Business Hub Space (public) which will be associated with the initial KNIME Business Hub Team.

> Initial Space

Initial Execution Context Name
The name for the initial KNIME Business Hub Execution Context. This Execution Context is provisioned with minimal CPU/memory resources, and the resource allocation can be increased post-installation via the KNIME Business Hub REST API.

> Initial Execution Context

Initial Execution Context Docker Image Repository
The image repository for the initial Execution Context.

> knime/knime-full

Default value: **registry.hub.knime.com/knime/knime-full**

Initial Execution Context Docker Image Tag
The image tag for the initial Execution Context.

> r-5.3.3-666

Default value: **r-5.2.5-592**

- KNIME Business Hub Admin Username - Choose a username for the KNIME Business Hub

- KNIME Business Hub Admin Password - Choose a password for the KNIME Business Hub

- Initial Team name - Choose a name for the initial team that KNIME Business Hub will create

- Initial Team description - Give a brief description of the Initial Team name

- Initial Space Name - This will be a public facing space name associated with the Initial Team Name

- Initial Execution Context Name - The name for the initial KNIME Execution Context Name

- Initial Execution Context Docker Image Repository - This is where the docker image link will go

- Initial Execution Context Docker Image Tag - This is where the initial Execution Context will go

> **i** There has been a change in the KNIME repository. In case it is pointing to *registry.hub.knime.com/knime/knime-full*, shorten it to *knime/knime-full* as shown in the screenshot. For a list of images and their tags from the KNIME repository, check the KNIME Business Hub Admin Guide.

The execution context has minimal resources (1CPU, 2GB memory) and a default executor provided by KNIME, to enable basic execution. For any production use of execution you should configure the execution context and assign more resources or use a different executor docker image.

Configure KNIME Business Hub URLs

URLs for KNIME Business Hub need to have the structure of:

- **Base URL**

    ◦ `<base-url>` (e.g. `hub.example.com`).

    ◦ The URL scheme (`http://` or `https://`) should not be included in the Base URL.

    ◦ The `<base-url>` must include the top-level domain (e.g. `.com`), and cannot be an IP address.

    ◦ This is the URL you use to view the KNIME Business Hub in your browser.

- Valid examples:

  - `hub.example.com`

  - `example.com`

- Invalid examples:

  - https://hub.example.com/

  - `example.com/hub`

  - `myhub`

  - `12.34.56.78`

- **Subdomains**

  - `apps.<base-url>`

  - `api.<base-url>`

  - `ws.<base-url>`

  - `auth.<base-url>`

  - `storage.<base-url>`

  - `registry.<base-url>`

The Base URL is the only URL that can be customized. The rest of the URLs are generated automatically.

- Webapp URL - The URL to the KNIME Business Hub webapp

- DataApps URL - The URL for the KNIME Business Hub DataApps

- API URL - The URL to the KNIME Business Hub API

- Websocket URL - The URL to the KNIME Business Hub Websocket

- Authentication URL - The URL to the KNIME Business Hub authentication endpoint

- Storage URL - The URL to the KNIME Business Hub persistent storage endpoint

- Registry URL - The URL to the KNIME Business Hub Embedded registry endpoint

If you are testing KNIME Business Hub without DNS configured, it is recommended to create `/etc/hosts` entries on your local machine pointing to the public IPv4 address of the instance running the cluster. This will redirect traffic from your local machine to the appropriate IPv4 address when you enter URLs such as `http://hub.example.com/` into your browser.

Notice that the values in `/etc/hosts` below are for `hub.example.com`. The values must match the config in the URLs section of the **Config** tab in the KOTS Admin Console, as

demonstrated above. You can always use `hub.example.com` as the Base URL for local test installations.

```
<public ip> hub.example.com
<public ip> api.hub.example.com
<public ip> auth.hub.example.com
<public ip> storage.hub.example.com
<public ip> apps.hub.example.com
<public ip> registry.hub.example.com
<public ip> ws.hub.example.com
```

On Windows machines you can find the `/etc/hosts` file in `<windows dir>\system32\drivers\etc\hosts`.

Branding

☑ **Enable Branding Options**

Custom Logo SVG File
The image will be scaled to fit 183px wide and 48px high logo area. Images with significantly more height than width may appear small when displayed.

Upload a file

⟳ Browse files for Custom Logo SVG File

If you would like to have branding options enabled check the *Enable Branding Options* box.

You will now be able to upload your own logo. Click *Browse files for Custom Logo SVG File* and navigate to select the file from you local machine.

If customizing the logo, the file being uploaded must be an .svg file in XML format such as the one below.

example.svg

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<!DOCTYPE svg PUBLIC "-//W3C//DTD SVG 1.1//EN"
"http://www.w3.org/Graphics/SVG/1.1/DTD/svg11.dtd">
<svg width="100%" height="100%" viewBox="0 0 183 48" version="1.1"
xmlns="http://www.w3.org/2000/svg" xmlns:xlink="http://www.w3.org/1999/xlink"
xml:space="preserve"
xmlns:serif="http://www.serif.com/" style="fill-rule:evenodd;clip-rule:evenodd;stroke-
linejoin:round;stroke-miterlimit:2;">
    <g transform="matrix(0.673983,0,0,0.673983,-2.4399,8.02946)">
        <text x="6.739px" y="43.245px" style="font-family:'Arial-BoldMT', 'Arial', sans-
serif;font-weight:700;font-size:54.619px;">EXAMPLE</text>
    </g>
</svg>
```

Execution Image Builder

Execution Image Builder

Configuration for the Execution Image Builder for building custom execution docker images and pushing them to the embedded registry or an external registry.

☐ **Enable Execution Image Builder**
Enable the use of Execution Image Builder. Note that the container for building an image has elevated privileges.

Here you can configure the Execution Image Builder to build custom execution Docker images and push them to the embedded registry or to an external registry.

If you check the *Enable Execution Image Builder* box, two additional fields will appear.

Execution Image Builder

Configuration for the Execution Image Builder for building custom execution docker images and pushing them to the embedded registry or an external registry.

☑ **Enable Execution Image Builder**
Enable the use of Execution Image Builder. Note that the container for building an image has elevated privileges.

☑ **Enable Execution Image Builder Cache**
Enable the use of the Execution Image Builder cache. Note that enabling the cache leads to more occupied disk space in the destination repository.

☐ **Skip TLS for Execution Image Builder**
Skips the TLS verification for pulling and pushing images.

By default *Enable Execution Image Builder Cache* will be selected. The option *Skip TLS for Execution Image Builder* skips the TLS verification for pulling and pushing images to the

registry.

Execution Contexts

Execution Contexts

KNIME Business Hub uses Execution Contexts for hosting and executing KNIME workflows.

☐ **Enable Custom Execution Image Pull Secret**
Enable the use of custom execution images that require Docker authentication to pull.

Executor Startup Timeout
The timeout in seconds after which the startup of an executor fails, is aborted and is tried again. Only needs to be changed if an executor image is used that takes a long time to start and never finishes startup within the configured time. Minimum value is 30.

60

Default value: **60**

By default *Enable Custom Execution Image Pull Secret* is unchecked. By checking it you will get another option that will appear:

Execution Image Pull Secret Name
The name of the Kubernetes Secret resource which contains the Docker image pull secret for custom execution image(s). The secret should exist in the namespace defined in the **Execution Context Namespace** config option.

regcred

Default value: **regcred**

- Execution Image Pull Secret Name - The name of the Kubernetes Secret resource which contains the Docker image pull secret for custom execution image(s).

- Executor Startup Timeout - The timeout in seconds after which the startup of an executor fails, is aborted and is tried again

Job Instrumentation Data

Job Instrumentation Data

KNIME Business Hub can collect instrumentation data on workflow executions.

☑ **Do you want to collect job instrumentation data?**

Job Info Expiration
How long instrumentation data about a job is stored, in days.

30

Default value: **30**

- Select the box *Do you want to collect job instrumentation data?* to allow the KNIME Business Hub instance to collect data about workflows execution.

- Job Info Expiration - This number reflects how long the instrumentation data about a job is stored in days

Configure networking

In the section *Networking* of the Admin Console you can deploy an external load balancer and enable and configure the Transport Layer Security (TLS).

It is anyways possible to configure this section in a second step.

To have more information on how to configure the networking section read the KNIME Business Hub Admin Guide.

Networking

Configuration for networking options such as Transport Layer Security (TLS).

☐ **Enable TLS** Recommended
Enable Transport Layer Security (TLS). This option is highly recommended for any KNIME Business Hub deployed in a production environment. A certificate must be created for all URLs configured above, including a wildcard certificate for `hub.example.com` and `*.hub.example.com`.

Notifications

In this section you can configure the notification service, which notifies for various events.

By enabling the *Enable Notifications* box you get a new option that is checked by default.

- Enable HTML in Email Notifications - allows HTML in email notifications, or disable this option for plain text

- Mail Server Configuration - Enter configuration properties for the Jakarta Mail library

> i You can consult the Jakarta Mail API documentation for all possible parameters in the Jakarta Mail API documentation.

The default code value is as follows:

Name	Value	Description
`mail.from`	hub.email@example.com	Address from which all mails are sent, required
`mail.smtp.host`	`<host>`	SMTP server host address
`mail.smtp.port`	`25`	SMTP port, default 25
`mail.smtp.auth`	`true`	Set to `true` if the mail server requires authentication
`mail.smtp.user`	`<user>`	Username for SMTP authentication; optional
`mail.password`	`<password>`	Password for authentication; optional

Name	Value	Description
`mail.smtp.starttls.enable`	`false`	If `true`, enables the use of the STARTTLS command (if supported by the server) to switch the connection to a TLS-protected connection before issuing any login commands.
`mail.smtp.ssl.enable`	`false`	If set to `true`, use SSL to connect and use the SSL port by default.

Embedded Docker Registry

ℹ️ The Embedded Docker Registry is only available on the kURL embedded cluster.

To execute workflows on KNIME Business Hub the users will need to have access to an execution context. The execution context is based on a Docker image containing a KNIME executor with specific extensions and assigned vCores and RAM.

KNIME provides a public registry from which is possible to get standard Docker images to build execution context running on a specific executor version. However, there might be cases in which a team or user might need specific extensions that are not available in the Docker images provided in the KNIME public registry.

Enabling the Embedded Docker Registry will make it available at the following URL, defined in the KNIME Business Hub URLs section:

```
`registry.<base-url>`
```

e.g. `registry.hub.example.com`.

Once a custom Docker image is created, as shown in the Docker executor images section of the KNIME Business Hub Admin Guide, it can be pushed to the Embedded Docker Registry. The images present in the registry are then available to be used to create an execution context.

Embedded Registry

Configuration for an Embedded Registry for pushing and pulling custom KNIME execution docker images.

☑ Enable Embedded Registry

Use the credentials below to authenticate your Docker CLI for pushing images to the registry. An image pull secret for executors is created automatically in the cluster.

Embedded Registry Username

knime

Default value: **knime**

Embedded Registry Password Required

                                                                                                           👁‍🗨

- Embedded Registry Username - The default value for *Embedded Registry Username* is `knime`

- Embedded Registry Password - Choose a password for the Embedded Registry

> ℹ Please note that, in order to avoid important images, needed for the correct functioning of the Business Hub instance, to be auto-removed from the Embedded Docker Registry, the image garbage collection can be disabled as described in the Replicated documentation.

Preflight checks

The final step before installing is the preflight checks, which is a set of automated tests to help identify if KNIME Business Hub is ready for installation. It will check the Kubernetes distribution, Kubernetes version, resources available to the cluster, and other mission-critical settings.

It is highly recommended to never skip the pre-flight checks during installation or upgrades.

Preflight checks

Preflight checks validate that your cluster meets the minimum requirements. Required checks must pass in order to deploy the application. Optional checks are recommended to ensure that the application will work as intended.

Results   Re-run

✅ **Required Kubernetes Version**
Your cluster meets the recommended and required versions of Kubernetes.

✅ **KOTS Admin Console Version**
Your cluster meets the required version of KOTS.

✅ **Container Runtime**
Containerd runtime was found.

✅ **Check Kubernetes environment.**
KURL is a supported distribution.

✅ **Total CPU Cores in the cluster is 16 or greater.**
There are at least 16 cores in the cluster.

✅ **PostgreSQL configuration**
PostgreSQL configuration has no syntax errors.

✅ **KNIME Business Hub URL validation**
URL passes validation for alphanumeric and special characters.

✅ **KNIME Business Hub URL validation: IP address**
URL is not an IP address.

Deploy

Wait for the installation to complete

If the preflight checks all passed and you opted to continue, the only thing left to do is wait for a few minutes until KNIME Hub finishes installing! You should see the installation turn the **Ready** status (top left) to green after 15-20 minutes.

If you cannot access the KNIME Business Hub Webapp URL after the **Ready** status has turned green, the first troubleshooting step would be to check the **Config** tab in the KOTS Admin Console and ensure the URLs are configured properly.
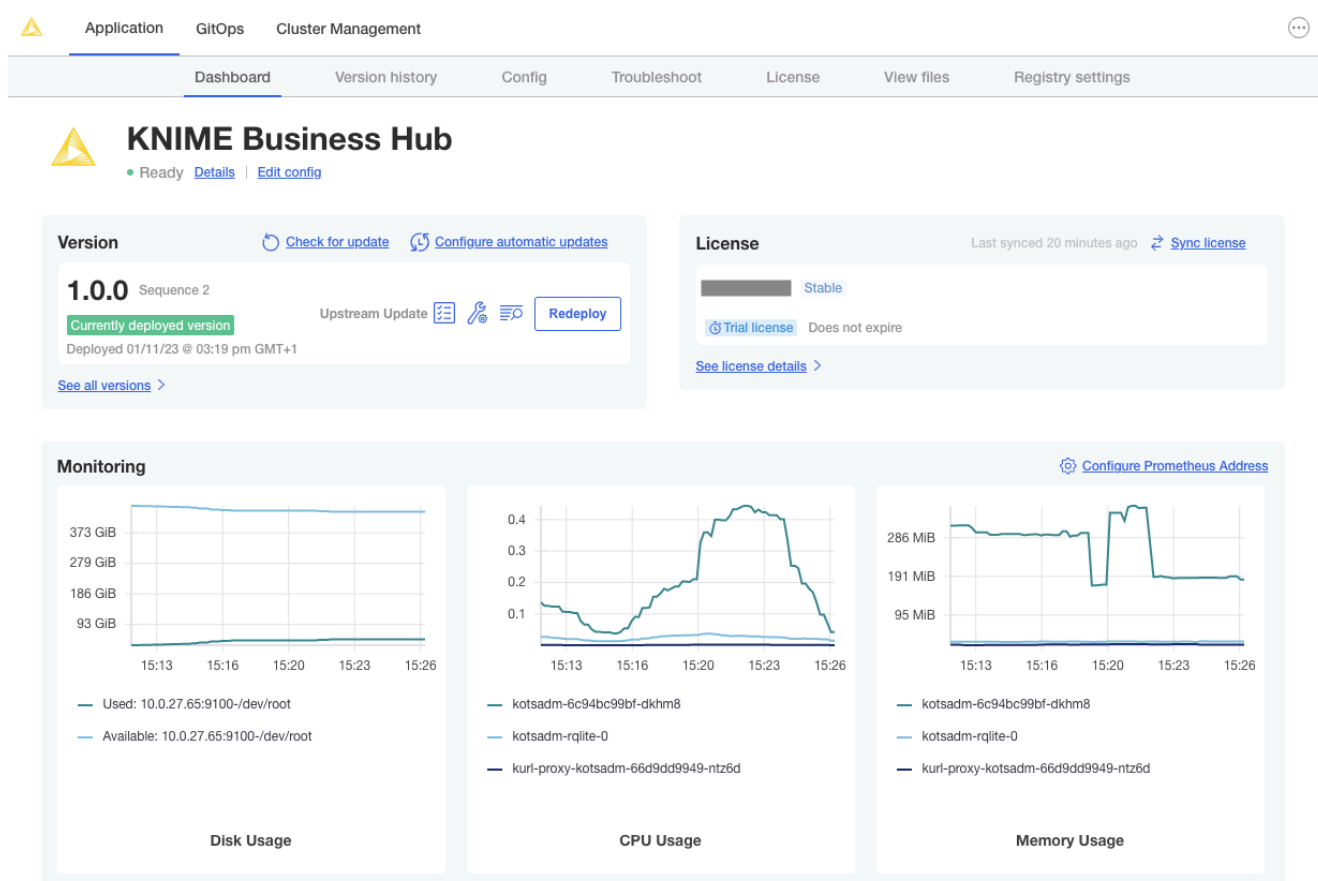
Navigating to the Webapp URL should display the KNIME Business Hub landing page.

KNIME | Hub

About   Sign in

Welcome to the
KNIME Hub

Solutions for data science: find workflows, nodes and components, and collaborate in spaces.

🔍 Search workflows, nodes and more...

0	**4 191**	**0**	**220**
Workflows	Nodes	Components	Extensions

KNIME Business Hub air gapped installation (single node)

Air gap install

The term air gap is used to indicate a computer network environment that has no external network access. This type of networking environment is useful for securing sensitive data, applications, and computing resources.

An air gap environment may be completely isolated or have a firewall between it and other networks with wider access. That wider access may be to the external world or within the wider organization.

Installing modern software in an air gap environment can be challenging. Containerization and public registries for container images work on the assumption that all software installs have access to the public internet. To get around this challenge, software installs have to package all the required artifacts and images into a bundle that can be copied to portable media for transport into the air gap environment.

The KNIME Business Hub supports installation into environments with externally accessible networks and into air gap environments. Of course air gap environments require an air gap install approach. But even networks with external access may require airgapped installations. For instance, many organizations do not allow downloading container images from public image registries. In this instance, an air gap install will be required.

> **i** The air gap install process described in this section is for a single node installation. In order to install KNIME Business Hub in a **multi-node, airgapped, highly available kURL cluster**, follow the instructions in the Highly-Available (HA) Embedded Cluster installation section instead.

Getting started

To get started with an air gap install you will need the link to the Download Portal of Replicated and a password that you will receive from your KNIME customer care representative.

Enter the Download Portal from a machine that has access to the internet and has also access to the installation machine.

Here, you need to select the type of installation that you need to perform. Select "Embedded

cluster" option so that the installation process will also install Kubernetes cluster along with the KNIME Business Hub.

From the download page you need to download:

- The Replicated license file (`.yaml`) with air gap install enabled.

- The airgap bundle for the KNIME Business Hub (*knime-hub Airgap Bundle*).

- Finally, you will need kURL bundle to install the kURL distribution of Kubernetes. This means you need to download the *Embedded Kubernetes installer* bundle.

Getting Kubernetes ready

Next step is to install a new Kubernetes cluster on your machine. KNIME supports the kURL distribution of Kubernetes.

This distribution is installed into a single machine and also supports extending to a cluster of machines.

Increase the limit for allowed open files

In single node installations the limit for allowed open files needs to be increased on the host VM for Business Hub to function properly.

We recommend to increase the limit by running the following command on the host VM.

```
sudo sysctl fs.inotify.max_user_instances=8192
```

This value needs to be set again after VM restarts. Alternatively you can persist it by adding it to `/etc/sysctl.conf`.

If the limit was not increased you might encounter the following error, appearing after the Business Hub installation is completed. In this case the `istio-proxy` container in the `keycloak-proxy` and `s3-proxy` pods isn't able to start up with a "too many open files" error.

An example error log can be seen below. Replace `keycloak-proxy-7ff44f845f-jrh9s` with an appropriate pod id in your cluster.

```
$ kubectl logs -n hub keycloak-proxy-7ff44f845f-jrh9s -c istio-proxy
{"level":"fatal","time":"2023-03-01T22:07:13.738069Z","msg":"Agent start up errorfailed
to start workload secret manager too many open files"}
```

Installing kURL

kURL is a Kubernetes distribution supported by KNIME Business Hub. If you do not have an existing Kubernetes cluster, then kURL can be used as the deployment cluster for KNIME Business Hub.

The kURL install bundle is required to install and deploy a Kubernetes instance to a single VM.

To download the kURL bundle go to the link to the Download Portal you received from your KNIME customer care representative, and download the Embedded Kubernetes Installer (`knime-hub.tar.gz`).

You will also need to create a file `patch.yaml` that looks like the following:

```
# This is an example patch file to override the Kurl MinIO object store's size. This
should
# be sized larger for air gap installs or when using the embedded registry to ensure
adequate
# disk is available. The default size is 10GB if not overridden.
apiVersion: cluster.kurl.sh/v1beta1
kind: Installer
metadata:
  name: "knime-hub-installer-patch-minio-size"
spec:
  MinIO:
    claimSize: "30Gi"
```

This is to ensure that enough storage is allocated for the MinIO object store used for staging/storing artifacts. By default, the Kurl MinIO object store is only 10GB, which is likely to fill up very quickly for airgap installations so you need to use the `patch.yaml` file to override the size setting at install.

Copy the downloaded bundle `knime-hub.tar.gz` and the `patch.yaml` to the target machine and run the following commands:

```
tar xvzf knime-hub.tar.gz
cat install.sh | sudo bash -s airgap installer-spec-file="./patch.yaml"
bash -l
```

The output of the command will contain the KOTS Admin Console URL and password. Look for these lines in the command output and copy the values:

```
Kotsadm: http://10.0.102.215:8800
Login with password (will not be shown again): <password>
```

The kotsadm password can be changed using the command:

```
kubectl kots reset-password default
```

The kotsadm password will be required when installing the KNIME Business Hub in this Kubernetes cluster.
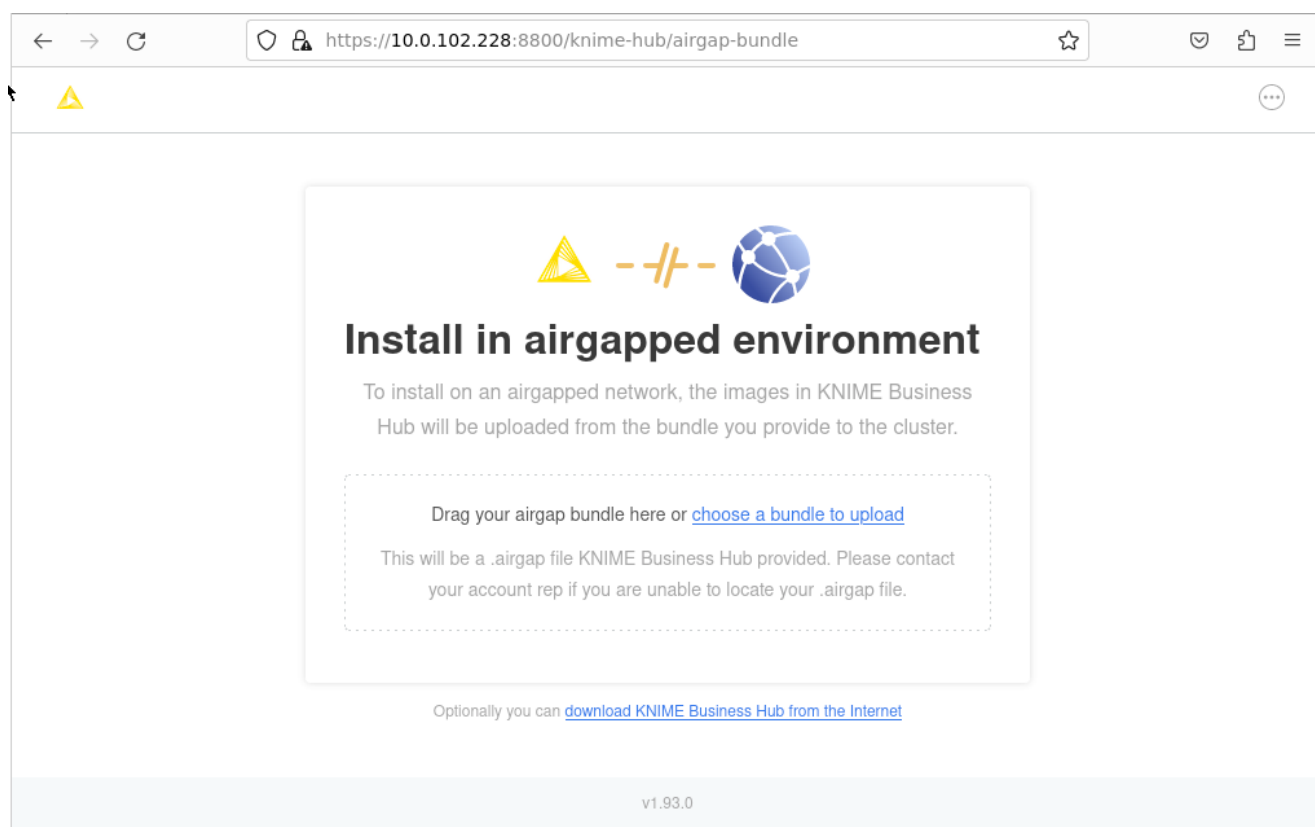
Installing KNIME Business Hub

Once your Kubernetes cluster is installed and configured with the KOTS extension, you are ready to install the KNIME Business Hub.

Now you need to proceed with the airgap install on a machine that:

- Hosts the web browser used for the install and has access to the internet

- Has access to the machine into which you are installing your KNIME Business Hub instance

On the machine that can access the installation machine, you can open the KOTS Admin Console URL and enter the kotsadm password. Follow the instructions for installing the KNIME Business via a browser. You will first need to upload the license and then the airgap bundle you downloaded from the Download Portal.

Once you select the KNIME Business Hub airgap bundle to use, the bundle will be uploaded to the install computer. The bundle is large and will take some time to upload.

Once the bundle has completed loading, the installer will extract all the required images from the bundle and store them in a local image registry. Again this will take some time. In your browser, you will see the progress of the extraction and load process. It will look like the screen shot below. Once the load into the registry is complete, the configuration page will be shown. Proceed as normal with the install at this point.
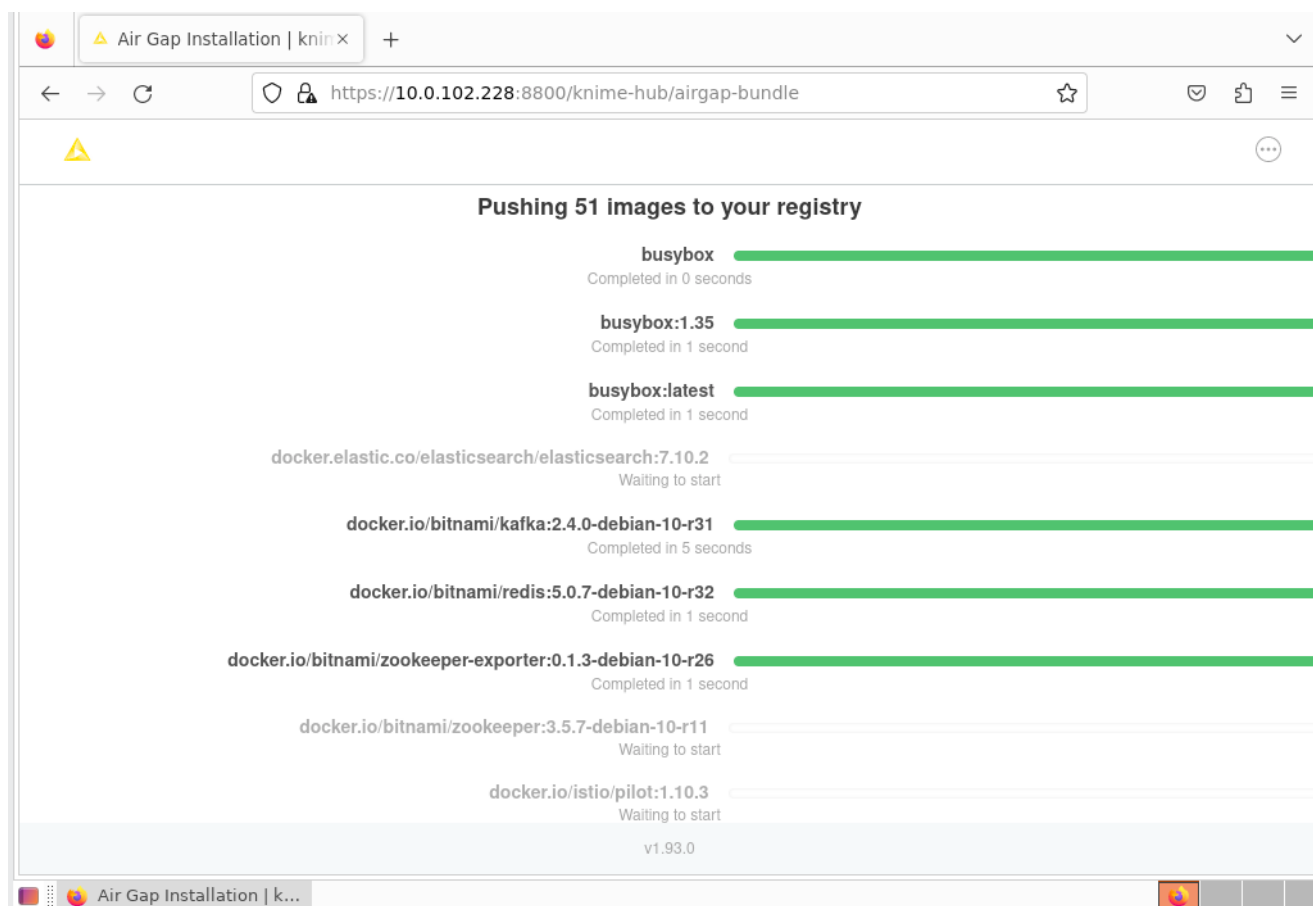
If all prior steps were successful, you should now be prompted to configure your KNIME Business Hub installation.

To do so follow the steps in the Configure the installation section.

Advanced installation types

This section covers advanced installation topics for detailed networking requirements, setting up highly-available (HA) clusters and other environmental considerations.

Highly-Available (HA) Embedded Cluster installation

A highly-available (HA) cluster consists of running multiple primary nodes which share the responsibility of acting as the **control-plane**, meaning any primary instance can ensure that all nodes in the cluster are properly managed and work is evenly distributed across them.

In an HA configured cluster where three or more nodes are running, any node can become unavailable without impacting the overall stability and health of the cluster. Furthermore, any processes running on a node that becomes unavailable will be automatically moved to an available node, allowing the cluster to automatically **self heal**.

Additionally, for a cluster to be highly-available, all data must be replicated dynamically between all nodes to ensure any migrated processes have access to all needed data. This is enabled by configuring the Rook volume provider add-on onto nodes during the installation process. Rook requires additional, unformatted block devices (disks) be attached to each node which it leverages for volume management. Additional information can be found here.

Installation overview

In a basic HA scenario, three or more nodes are installed where each node is configured to act as a **primary node**. A primary node is a node that additionally runs processes for the Kubernetes control-plane. A **secondary node** is a node that only runs non control-plane processes.

Having all nodes configured as primary nodes ensures any node can become unavailable without affecting the stability of the Kubernetes cluster.

If more than three nodes are planned to be configured in a cluster, a minimum of three of them must be primary nodes. All additional nodes can be installed as secondary nodes if desired.

When installing the Kurl embedded Kubernetes cluster in an HA configuration, the installation process is fully run on an initial instance, creating the first **primary node**. Upon completion of the install, output will be generated and printed to the console which includes a "**join** command" that can be run on each subsequent instance to configure it as a new node and cluster it with existing nodes.

Note all relevant ports that should be open for nodes to communicate with each other in the advanced Networking requirements section.

Installing the first node

When installing the `kURL` embedded cluster, the command line script needs to be modified to pass in additional parameters to configure the node to enable additional HA components.

Run the following install command with the following flags to initialize the first node.

```
curl -sSL https://kurl.sh/knime-hub-stable-ha | sudo bash -s ha ekco-enable-internal-load-balancer
```

The installer may prompt the user to indicate when additional configurations are being enabled or installed, but will otherwise proceed as normal.

This install command is similar to the one for the single-node install, but has two additional parameters (listed after the `-s` flag) to enable the `ha` configuration option and the `internal-load-balancer` which makes the Kubernetes control-plane API a highly-available endpoint.

Installing additional nodes

1. After installing on the first node, the terminal prints instructions to access the KOTS Admin Console and a join command with a token.

   > **i** Be aware that the master node join commands expire after 2 hours, and the worker node join commands expire after 24 hours. You can generate new node join commands by running `cat ./tasks.sh | sudo bash -s join_token ha airgap` on an existing primary node.

2. Copy the downloaded bundle `knime-hub-stable.ha.tar.gz` to each target machine and extract the archive on each machine:

   ```
   # Extract the contents of the tarball
   tar xvzf knime-hub-stable-ha.tar.gz
   ```

3. Run the respective join command from the terminal output of the first node installation. Use the command for either a secondary node or another primary node, on each machine.

> **i** Only to **join** the **first primary node** you will also need the `patch.yaml` file. This must not be used for any of the additionally joined primary nodes. To do so, copy to the machine the file `patch.yaml` from the first node installation and add `installer-spec-file="./patch.yaml"` at the end of the join command that you will find in the terminal output of the first node installation.

> **i** The Rook addon to be able to replicate data to all nodes needs to have at least 3 nodes available during cluster installation. Until at least 3 nodes are installed, the OpenEBS addon is used to provision Local Persistent Volumes to Stateful workloads.

4. When the third node joins, the system prompts for automatic migration from OpenEBS to Rook.

5. Once all nodes are added, continue the installation via the KOTS Admin Console.

After the installation is complete, you can view each node status under *Cluster Management* tab in the KOTS Admin Console.

Installing HA cluster in an air gap environment

In order to install KNIME Business Hub into a **multi-node, airgapped, highly avaialable kURL cluster** follow these steps:

1. From the Download portal of Replicated, on a machine with internet access and access to the machine into which you are installing your KNIME Business Hub instance, download the airgapped kURL bundle from https://kurl.sh/knime-hub-stable-ha.

2. Create a `patch.yaml` that looks like the following:

```
# This is an example patch file to override the Kurl MinIO object store's size.
This should
# be sized larger for air gap installs or when using the embedded registry to
ensure adequate
# disk is available. The default size is 10GB if not overridden.
apiVersion: cluster.kurl.sh/v1beta1
kind: Installer
metadata:
  name: "knime-hub-installer-patch-minio-size"
spec:
  MinIO:
    claimSize: "30Gi"
```

Install the cluster on the first (primary) node

1. Copy the downloaded bundle `knime-hub-stable.ha.tar.gz` and the `patch.yaml` to the target machine and run the following commands to install the first node:

```
# Extract the contents of the tarball
tar xvzf knime-hub-stable-ha.tar.gz
```

2. Run the install command with the `airgap` and `ha` flags, and specify the `patch.yaml` file as the installer spec file. The command should look like this:

```
# This command runs the `install.sh` script with `sudo` in an air-gapped
environment, enabling high availability (HA) and the EKCO internal load balancer.
It uses the `patch.yaml` file for configuration. Ensure `install.sh` and
`patch.yaml` exist and have proper permissions.
cat install.sh | sudo bash -s airgap installer-spec-file="./patch.yaml" ha ekco-
enable-internal-load-balancer
# Starts a new login shell to apply environment changes made during the
installation process.
bash -l
```

3. The output of the command will contain the KOTS Admin Console URL and password.

Join additional nodes

1. After installing on the first node, the terminal prints instructions to access the KOTS Admin Console and a join command with a token.

> **i** Be aware that the master node join commands expire after 2 hours, and the worker node join commands expire after 24 hours. You can generate new node join commands by running `cat ./tasks.sh | sudo bash -s join_token ha airgap` on an existing primary node.

2. Copy the downloaded bundle `knime-hub-stable.ha.tar.gz` to each target machine and extract the archive on each machine:

```
# Extract the contents of the tarball
tar xvzf knime-hub-stable-ha.tar.gz
```

3. Run the respective join command from the terminal output of the first node installation. Use the command for either a secondary node or another primary node, on each machine.

> **i** Only to **join** the **first primary node** you will also need the `patch.yaml` file. This must not be used for any of the additionally joined primary nodes. To do so, copy to the machine the file `patch.yaml` from the first node installation and add `installer-spec-file="./patch.yaml"` at the end of the join command that you will find in the terminal output of the first node installation.

> **i** The Rook addon to be able to replicate data to all nodes needs to have at least 3 nodes available during cluster installation. Until at least 3 nodes are installed, the OpenEBS addon is used to provision Local Persistent Volumes to Stateful workloads.

4. When the third node joins, the system prompts for automatic migration from OpenEBS to Rook.

5. Once all nodes are added, continue the installation via the KOTS Admin Console.

After the installation is complete, you can view each node status under *Cluster Management* tab in the KOTS Admin Console.

> **i** For more information on the installation of air gapped clusters, see the KNIME Business Hub airgapped installation section.

Networking requirements

Firewall openings for online installations

The following domains need to be accessible from servers performing online kURL installs. IP addresses for these services can be found in replicatedhq/ips.

Host	Description
amazonaws.com	tar.gz packages are downloaded from Amazon S3 during embedded cluster installations. The IP ranges to allowlist for accessing these can be scraped dynamically from the AWS IP Address Ranges documentation.

Host	Description
k8s.gcr.io	Images for the Kubernetes control plane are downloaded from the Google Container Registry repository used to publish official container images for Kubernetes. For more information on the Kubernetes control plane components, see the Kubernetes documentation.
k8s.kurl.sh	Kubernetes cluster installation scripts and artifacts are served from kurl.sh. Bash scripts and binary executables are served from kurl.sh. This domain is owned by Replicated, Inc which is headquartered in Los Angeles, CA.

No outbound internet access is required for airgapped installations.

Host firewall rules

The kURL install script will prompt to disable firewalld. Note that firewall rules can affect communications between containers on the **same** machine, so it is recommended to disable these rules entirely for Kubernetes. Firewall rules can be added after or preserved during an install, but because installation parameters like pod and service CIDRs can vary based on local networking conditions, there is no general guidance available on default requirements. See Advanced Options for installer flags that can preserve these rules.

The following ports must be open between nodes for multi-node clusters:

Primary Nodes:

Protocol	Direction	Port Range	Purpose	Used By
TCP	Inbound	6443	Kubernetes API server	All
TCP	Inbound	2379-2380	etcd server client API	Primary
TCP	Inbound	10250	kubelet API	Primary
UDP	Inbound	8472	Flannel VXLAN	All
TCP	Inbound	6783	Weave Net control	All
UDP	Inbound	6783-6784	Weave Net data	All
TCP	Inbound	9090	Rook CSI RBD Plugin Metrics	All

Protocol	Direction	Port Range	Purpose	Used By
TCP	Inbound	31880	OpenEBS to Rook migration during setup	Primary

Secondary Nodes:

Protocol	Direction	Port Range	Purpose	Used By
TCP	Inbound	10250	kubelet API	Primary
UDP	Inbound	8472	Flannel VXLAN	All
TCP	Inbound	6783	Weave Net control	All
UDP	Inbound	6783-6784	Weave Net data	All
TCP	Inbound	9090	Rook CSI RBD Plugin Metrics	All

These ports are required for Kubernetes and Weave Net.

Available ports

In addition to the ports listed above that must be open between nodes, the following ports should be available on the host for components to start TCP servers accepting local connections.

Port	Purpose
2381	etcd health and metrics server
6781	weave network policy controller metrics server
6782	weave metrics server
10248	kubelet health server
10249	kube-proxy metrics server
9100	prometheus node-exporter metrics server
10257	kube-controller-manager health server

Port	Purpose
10259	kube-scheduler health server

Installation on hosts with undersized root volumes

By default, the Kurl embedded cluster uses OpenEBS for volume provisioning, which leverages the hosts disk for persistence. This location defaults to `/var/openebs/local`. Additionally, the host disk is used for caching container images and other artifacts.

If the host disk does not have sufficient capacity for installation, an additional disk can be mounted and configured for use.

Follow the recommended best practice for your hardware, infrastructure provider and Linux distribution to add a new disk and ensure a partition and filesystem have been created on it and that it is mounted.

The following steps can then be used to configure that disk for persistence with KNIME Business Hub.

Set up new directories and symlinks

The disk should be mounted to `/var/lib/replicated`. Once that is done, the following will create and link additional needed directories.

```
mkdir /var/lib/replicated/kubelet
ln -s /var/lib/replicated/kubelet /var/lib/kubelet

mkdir /var/lib/replicated/openebs
ln -s /var/lib/replicated/openebs /var/openebs

mkdir /var/lib/replicated/containerd
ln -s /var/lib/replicated/containerd /var/lib/containerd

mkdir /var/lib/replicated/kurl
```

Update installer script

Next, download the installer script, but don't execute it.

```
curl -sSL https://kurl.sh/knime-hub > kurl-installer.sh
```

Inside the installer script is a line starting with `Environment="KUBELET_CONFIG_ARGS=`, which specifies flags to be passed to the `kubelet` process that runs on the node. An additional flag (`--root-dir`) needs to be appended to these arguments to point to `/var/lib/replicated/kubelet`.

This line in the installer script can be manually updated, or the following `sed` command can be run to automatically apply the change.

```
sed -i 's/\/var\/lib\/kubelet\/config\.yaml/\/var\/lib\/kubelet\/config\.yaml --root-dir=\/var\/lib\/replicated\/kubelet/g' kurl-installer.sh
```

Once edited, the complete line should look like the following:

```
Environment="KUBELET_CONFIG_ARGS=--config=/var/lib/docker/kubelet/config.yaml --root-dir=/var/lib/replicated/kubelet"
```

Install Kurl with argument overrides

To complete the install, `chmod` is used to make the shell script executable, then the script is executed with any needed arguments.

The `kurl-install-directory="/var/lib/replicated/kurl"` argument must be specified. Other arguments can additionally be added as needed. Note that, unlike standard install command which uses `curl` to download the installation script and `pipe` it directly into a `bash` shell, the `-s` flag is not needed before specifying arguments when the script is directly executed.

```
chmod +x kurl-installer.sh
./kurl-installer.sh kurl-install-directory="/var/lib/replicated/kurl"
```

Post-installation steps

Version updates

If you save any changes in the **Config** tab of the KNIME Business Hub Admin Console, or check for updates and see a new version that you can upgrade to, then the new version will be visible in the **Version history** tab. New versions of KNIME Business Hub will not be deployed automatically unless automatic updates have been configured. Preflight checks will execute prior to deployment and the deployment itself can be triggered by clicking the **Deploy** button.

User registration

After initial installation, start the process of creating the first user by clicking the **Sign In** button.

Welcome to the
KNIME Hub
Solutions for data science: find workflows, nodes and components, and collaborate in spaces.

About Sign in

0	**4 191**	**0**	**220**
Workflows	Nodes	Components	Extensions
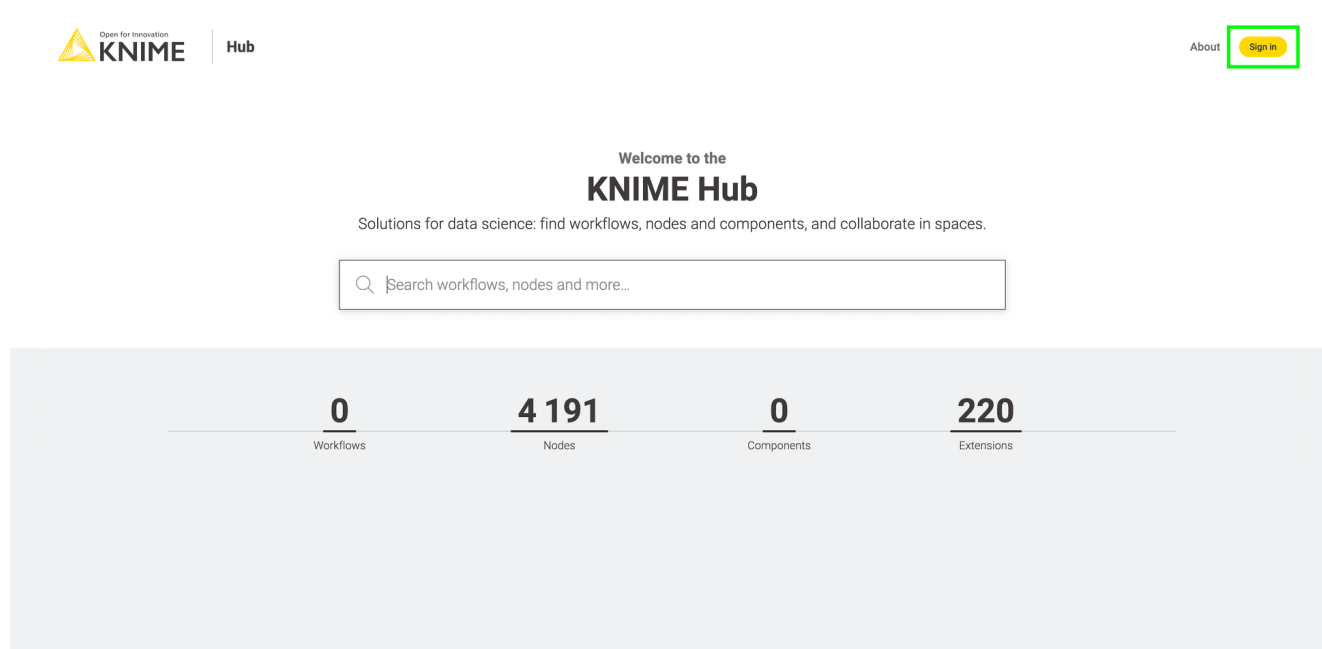
Next, click the **Register** button to proceed with creating a new account. You will be prompted for user information and will be logged in automatically.

Users can change their password by going to `auth.<base-url>/auth/realms/knime/account` (e.g. `http://auth.hub.example.com/auth/realms/knime/account`) and navigating to **Account Security** → **Signing In**.
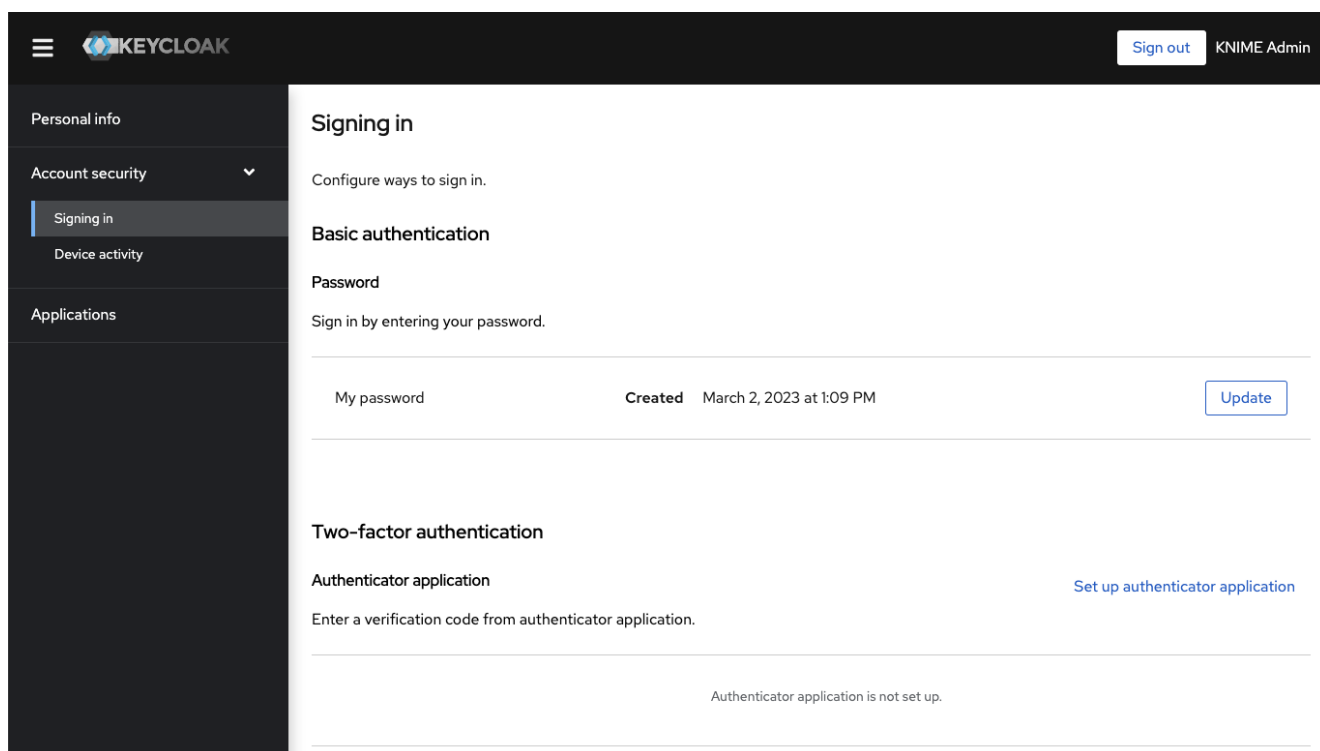
Keycloak setup

Keycloak is an open source software product that KNIME Business Hub leverages for authenticating users with Single-Sign On. Keycloak offers Identity Brokering and User Federation to external Identity Providers.

You can manage your Keycloak setup by going to `auth.<base-url>/auth/` (e.g. `http://auth.hub.example.com/auth/`), clicking Administration Console and logging in with the Keycloak admin credentials. These credentials are stored in a kubernetes secret called `credential-knime-keycloak` in the `<business-hub-namespace>` namespace.

You can perform the steps here to retrieve the Keycloak credentials at any time.

> **i** When you first log in to the Keycloak Administration Console you are shown the `master` realm. However, all KNIME Business Hub related configurations are in the `knime` realm. You can select the realm via the dropdown menu in the top left corner.

For configuring your Keycloak setup, e.g. for adding User Federation or an external Identity Providers, consult the Keycloak Server Administration Guide.

Integrating an external Identity Provider (IdP)

While Keycloak itself can be used as the user base, KNIME Business Hub can also federate to an external IdP. To do this the external IdP needs to support a protocol known to Keycloak, such as SAML v2.0, OpenID Connect v1.0, or OAuth v2.0. For configuring Keycloak to use an external IdP please see the Integrating identity providers section in Keycloaks own documentation on this topic.

> ℹ️ If you are using **Microsoft Entra ID** as external Identity Provider you can restrict who can register to your KNIME Business Hub instance by deactivating the option "Assignment required?" in Microsoft Entra ID to *Yes* and then allowing only a subset of users or groups to access the KNIME Business Hub instance as explained in the Microsoft Entra ID documentation.

Requirements

Keycloak and KNIME Business Hub have the following requirements for a user or external Identity Provider:

- Usernames can only include letters and numbers as first character. Letters, numbers, underscore, and @ as last character. Letters, numbers, space, dot, underscore, hyphen, and @ for remaining characters. The minimum length is 3 characters, the maximum is 64. Special characters like umlauts (äöü) are not allowed.

    - KNIME Business Hub uses this regex to validate usernames: `^[a-zA-Z0-9][a-zA-Z0-9-_.@ ]{1,62}[a-zA-Z0-9_@]$`.

- When a new user is added to keycloak, it needs to have values for the *Username*, *Email*, *First name* and *Last name* fields. If any of those are not filled in automatically you need to create a mapper for it in the Identity Provider settings. Otherwise the user will be prompted to fill in details themselves when they log in for the first time.

- Each user needs to have an email address, even if notifications are not configured in KNIME Business Hub.

Identity Provider Mapper

After adding an external IdP to your Keycloak instance you can create Identity Provider Mappers. These mappers can be used to fill in attributes if they are not getting automatically added, like *Email*. See the Mapping claims and assertions section in Keycloaks documentation.

In Keycloak, in the `knime` realm, go to the Identity Providers section, click your provider and go

to the Mappers tab. Click the *Add Mapper* button and configure the mapper. Some useful Mapper types:

- *Username Template Importer*: you can use this mapper to generate a username from multiple claims if a username claim is not available. Example template: `${CLAIM.given_name}.${CLAIM.family_name}`. `given_name` and `family_name` are the given claims here. Nesting is possible with e.g.`${CLAIM.user_info.given_name}`.

  - There are three transformers you can use here: `lowercase`, `uppercase`, and `localpart`.

- *Attribute Importer*: map an Identity Provider claim to a Keycloak attribute. Useful target attributes: `email` (*Email*), `given_name` (*First name*), and `family_name` (*Last name*). The Claim field is not a template like above, but accepts the claim name directly, e.g.`family_name`.

- Use the *Hardcoded Attribute* mapper if an attribute can't be constructed from a claim.

For finding available claims see the section below.

Tips for debugging Keycloak when integrating an external IdP

To create appropiate mappers in the IdP configuration one needs to know what the external IdP is actually returning to Keycloak in the token. This can be done by increasing the loglevel of the `org.keycloak.social.user_profile_dump` keycloak logger to `debug`. To do so you need to modify an instance of the `keycloaks.k8s.keycloak.org` CustomResource. You can get the current yaml definition of it by executing:

```
kubectl get keycloaks.k8s.keycloak.org -n knime knime-keycloak -o yaml
```

You can either save this output to a file, modify the file, and apply the changes to the cluster with:

```
kubectl apply -f <file>
```

Or you can edit it directly in the cluster using `kubectl edit`. This will open the default editor in your current shell. Make the modification, save and exit the editor, and it will automatically be applied to the cluster:

```
kubectl edit keycloaks.k8s.keycloak.org -n knime knime-keycloak
```

The modification that needs to be done is adding an additonal configuration option under

`spec.additionalOptions`:

```
spec:
  additionalOptions:
    - name: log-level
      value: INFO,org.keycloak.social.user_profile_dump:debug
```

Do not remove or change any of the other lines.

After this modification has been applied the `keycloak-operator` will automatically reconcile the `knime-keycloak-0` pod in the `<business-hub-namespace>` namespace. After the pod is ready again attempt a login with the configured external IdP in the KNIME Business Hub UI (e.g. http://hub.example.com) and look at the log output of the `knime-keycloak-0` pod. The interesting lines contain `org.keycloak.social.user_profile_dump` with the full JSON that was received from the external IdP:

```
2024-05-23 08:45:21,103 DEBUG [org.keycloak.social.user_profile_dump] (executor-thread-
2) User Profile JSON Data for provider oidc: {"sub":"02aebee3-aaba-4e96-86a7-
1d12531e8fb2","email":"admin@example.com","email_verified":false,"name":"Admin
Admin","preferred_username":"admin","zoneinfo":"Europe/Berlin"}
```

To turn the debug logging off again in Keycloak, remove the `additionalOptions` item that you added in the earlier step.

Update your KNIME Business Hub license

In order to deploy a new Business Hub license, please go to the Replicated console. There, navigate to the Config tab and find your current license file.

Click "select a different file", and choose the `.xml` file provided by your KNIME customer care representative. Afterwards, scroll all the way to the bottom to confirm the configuration change. Click "go to updated version" next. This brings you to the "version history", where you need to click on "deploy" to switch to the new license.

Connecting kubectl or other external tools to your cluster

Executing the following command on the Ubuntu instance in which KNIME Business Hub is installed will output the kubeconfig file which is required for accessing your cluster from another machine.

Sometimes the `KUBECONFIG` environment variable is not set automatically after installation. Running `bash -l` will reload the shell and likely solve the issue. Otherwise, you can run `kubectl config view --raw` which is equivalent to `cat $KUBECONFIG`.

```
cat $KUBECONFIG
```

Note that the `.clusters[0].cluster.server` property is almost certainly set to the private IPv4 address of the cluster (incorrect) and not the public IPv4 address (correct). Update the property to match the public IPv4 address of the Ubuntu instance hosting KNIME Business Hub.

```
apiVersion: v1
clusters:
- cluster:
    certificate-authority-data: ...
    server: https://<replace-with-public-ip>:6443
  name: kubernetes
contexts:
- context:
    cluster: kubernetes
    user: kubernetes-admin
  name: kubernetes-admin@kubernetes
current-context: kubernetes-admin@kubernetes
kind: Config
preferences: {}
users:
- name: kubernetes-admin
  user:
    client-certificate-data: ...
    client-key-data: ...
```

Uninstalling KNIME Business Hub

Uninstalling KNIME Business Hub is a **highly destructive action** that can have permanent implications. Please ensure you are connected to the right cluster and 100% sure you want to uninstall all resources related to KNIME Business Hub before proceeding. Please ensure you have retrieved all data or backups from the cluster that you want to preserve.

To completely remove a KNIME Business Hub instance, the following commands can be run from a terminal which has kubectl access to the cluster running KNIME Business Hub.

- Use the namespace where KNIME Business Hub was installed in place of `<business-hub-namespace>`

- Make sure both kubectl and Helm are installed to successfully run the commands

- Confirm that the proper Kubernetes context is set before executing

```
# Remove the KNIME Business Hub app from the KOTS Admin Console.
kubectl kots remove knime-hub -n default --force

# List all helm releases that will be deleted in a later step. This does not delete
anything yet.
# Ensure there are no helm releases that you deployed yourself.
helm ls -a -n <business-hub-namespace>

# Delete all helm releases in the hub namespace.
helm ls -a -n <business-hub-namespace> | awk 'NR > 1 { print  "-n "$2, $1}' | xargs -L1
helm delete;

# Delete the knime-hub-keycloak helm release, and wait 10seconds to give the knime-
keycloak-operator time to reconcile.
helm delete -n <business-hub-namespace> knime-hub-keycloak; kubectl delete
keycloakrealms.legacy.k8s.keycloak.org knime-realm -n <business-hub-namespace>; sleep
10;

# Delete all helm releases in the `<business-hub-namespace>` namespace.
helm ls -a -n <business-hub-namespace> | awk 'NR > 1 { print  "-n "$2, $1}' | xargs -L1
helm delete;

# Remove finalizer in Keycloak realm resource
kubectl patch keycloakrealms.keycloak.org -n <business-hub-namespace> knime-realm  -p
'{"metadata":{"finalizers":null}}' --type=merge

# Finally, delete all namespaces that were created by KNIME Business Hub.
kubectl delete namespace istio-system <business-hub-namespace>
```

If the commands above run successfully, all KNIME Business Hub resources will be removed from the cluster. You could then re-install KNIME Business Hub into the same cluster by

navigating to the KOTS Admin Console in a browser and following this installation guide again.

Removing kubernetes from a VM

You can uninstall all KOTS resources or remove everything related to kubernetes from a VM by following the documentation provided by Replicated under "Delete the Admin Console".