Open for Innovation
KNIME®

KNIME Server on Kubernetes

KNIME AG, Zurich, Switzerland
Version 4.9 (last updated on 2019-06-18)

Table of Contents

Introduction

KNIME Server is the enterprise software for team based collaboration, automation, management, and deployment of data science workflows, data, and guided analytics. Non experts are given access to data science via KNIME WebPortal or can use REST APIs to integrate workflows as analytic services to applications and IoT systems. A full overview is available here.

For an overview of use cases, see our solutions page. Presentations at KNIME Summits about usage of the KNIME Server can be found here.

Further reading

If you are looking for detailed explanations around the additional configuration options for KNIME Server, you can check the KNIME Server Administration Guide.

If you are looking to install KNIME Server, you should first consult the KNIME Server Installation Guide.

For guides on connecting to KNIME Server from KNIME Analytics Platform, or using KNIME WebPortal please refer to the following guides:

- KNIME Explorer User Guide
- KNIME WebPortal User Guide

There are additional resources such as the KNIME Server Advanced Setup Guide and KNIME Server Preview Functionality Guide.

Deployment on Kubernetes

KNIME Server can be deployed on Kubernetes. There are several options:

- [KNIME Server Small]
- [KNIME Server Medium]
- [KNIME Server Large]

For a full list of product offerings including KNIME Analytics Platform, see here.

When deploying on Kubernetes there are several things that you will need to consider.

*

Prerequisites

The person responsible for the deployment of KNIME Server should be familiar with Kubernetes functionality surrounding launching, and editing HELM templates. KNIME Server administration on Kubernetes is primarily handled by changing variables in templates used to control the deployment.

KNIME Server BYOL is a single container deployment launched from an image supplied by KNIME. KNIME Server Large is a multi-container deployment launched from multiple instances supplied by KNIME, and external providers. It is possible to build the images from scratch. This is covered in the section https://docs.knime.com/2019-06/common/index.pdf

Kubernetes resources

Launching an instance requires a VPC and subnet. The default security group will enable HTTP access on port 80, and HTTPS access on port 443.

The KNIME Server administrator must have permissions to launch templates.

Pre-installed software (Kubernetes Specific)

No Kubernetes specific software is included in the image builds.

Pre-installed software

For convenience we have installed and pre-configured:

- OpenJDK 8 (required)
- Anaconda Python (see Update Python configuration)
- R
- Chrony (To ensure system clock is syncronised)
- Postfix (To enable KNIME Server to send email notifications)
- iptables (Redirects of requests on port 80, 443 to TomEE running on port 8080, 8443)

Extensions installed with Executors

To install additional extensions, see the section Install additional extensions.

Extensions all come from the following update sites:

- https://update.knime.com/analytics-platform/4.0
- https://update.knime.com/community-contributions/trusted/4.0

The following extensions are installed:

- org.knime.features.activelearning.feature.group
- org.knime.features.cloud.aws.mlservices.feature.group
- org.knime.features.cloud.aws.athena.feature.group
- org.knime.features.cloud.aws.feature.group
- org.knime.features.cloud.aws.redshift.feature.group
- org.knime.features.cloud.aws.redshift.driver.feature.group
- org.knime.product.desktop
- org.knime.features.audio.feature.group
- org.knime.features.arima.feature.group
- org.knime.features.cloud.azure.feature.group
- org.knime.features.chem.types.feature.group
- org.knime.features.bigdata.connectors.feature.group
- org.knime.features.ext.chem.tools.feature.group
- org.knime.features.parquet.feature.group
- org.knime.features.datageneration.feature.group
- org.knime.features.database.feature.group
- org.knime.features.dl.keras.feature.group
- org.knime.features.dl.tensorflow.feature.group
- org.knime.features.dl.onnx.feature.group
- org.knime.features.ext.dl4j.feature.group
- org.knime.features.distmatrix.feature.group
- com.knime.features.enterprise.slave.feature.group
- org.knime.features.expressions.feature.group
- org.knime.features.kafka.feature.group
- org.knime.features.bigdata.spark.feature.group
- org.knime.features.bigdata.fileformats.feature.group

- org.knime.features.bigdata.spark.local.feature.group
- org.knime.features.ext.h2o.mojo.spark.feature.group
- org.knime.features.ext.chromium.feature.group
- org.knime.features.ext.exttool.feature.group
- org.knime.features.exttool.feature.group
- org.knime.features.base.filehandling.feature.group
- org.knime.features.scm.git.feature.group
- org.knime.features.ext.h2o.feature.group
- org.knime.features.ext.h2o.mojo.feature.group
- org.knime.features.ext.h2o.spark.feature.group
- org.knime.features.ext.birt.feature.group
- org.knime.features.ext.lucene.feature.group
- org.knime.features.r.feature.group
- org.knime.features.ext.itemset/latest[org.knime.features.ext.itemset.feature.group
- org.knime.features.js.views.feature.group
- org.knime.features.js.views.labs.feature.group
- org.knime.features.ext.jfreechart.feature.group
- org.knime.features.jpmml.feature.group
- org.knime.features.ext.jython.feature.group
- org.knime.features.mli.feature.group
- org.knime.features.database.connectors.sqlserver.driver.feature.group
- org.knime.features.database.extensions.sqlserver.driver.feature.group
- org.knime.features.microsoft.r.feature.group
- org.knime.features.base.pmml.feature.group
- org.knime.features.mongodb.feature.group
- org.knime.features.neighborgram.feature.group
- org.knime.features.network.feature.group
- org.knime.features.network.distmatrix.feature.group
- org.knime.features.base.widedata.feature.group
- org.knime.features.ext.osm.feature.group

- org.knime.features.optimization.feature.group
- org.knime.features.ext.perl.feature.group
- org.knime.features.ext.webservice.client.ppilot.feature.group
- org.knime.features.js.plotly.feature.group
- org.knime.features.base.pmml2.feature.group
- org.knime.features.base.pmml.translation.feature.group
- org.knime.features.python2.feature.group
- org.knime.features.quickform.legacy.feature.group
- org.knime.features.ext.r.bin.feature.group
- com.knime.features.gateway.explorer.feature.group
- com.knime.features.gateway.remote.feature.group
- com.knime.features.reporting.designer.feature.group
- org.knime.features.rest.feature.group
- org.knime.features.ext.md.feature.group
- org.knime.features.ext.parso.feature.group
- org.knime.features.semanticweb.feature.group
- com.knime.features.explorer.serverspace.feature.group
- org.knime.features.ext.spotfire.feature.group
- org.knime.features.stats2.feature.group
- org.knime.features.core.streaming.feature.group
- org.knime.features.ext.svg.feature.group
- org.knime.features.ext.tableau.bin.feature.group
- org.knime.features.ext.tableau.hyper.bin.feature.group
- org.knime.features.ext.tableau.hyper.feature.group
- org.knime.features.ext.tableau.feature.group
- com.knime.features.explorer.sharedspace.feature.group
- org.knime.features.ext.textprocessing.feature.group
- org.knime.features.ext.textprocessing.dl4j.feature.group
- org.knime.features.ext.socialmedia.feature.group
- org.knime.features.virtual.feature.group

- org.knime.features.ext.webservice.client.feature.group
- org.knime.features.webanalytics.feature.group
- org.knime.features.ext.weka_3.7.feature.group
- com.knime.features.bigdata.knosp.feature.group
- org.knime.features.xgboost.feature.group
- jp.co.infocom.cheminfo.marvin.feature.feature.group
- org.erlwood.features.core.base.feature.group
- com.continental.knime.feature.feature.group
- com.sjwebb.knime.slack.feature.feature.group

Optional external dependencies

Optionally KNIME Server Large instances (currently available via BYOL license only) may choose to connect KNIME Server to an external LDAP/AD service. Full details are contained in KNIME Server Advanced Setup Guide.

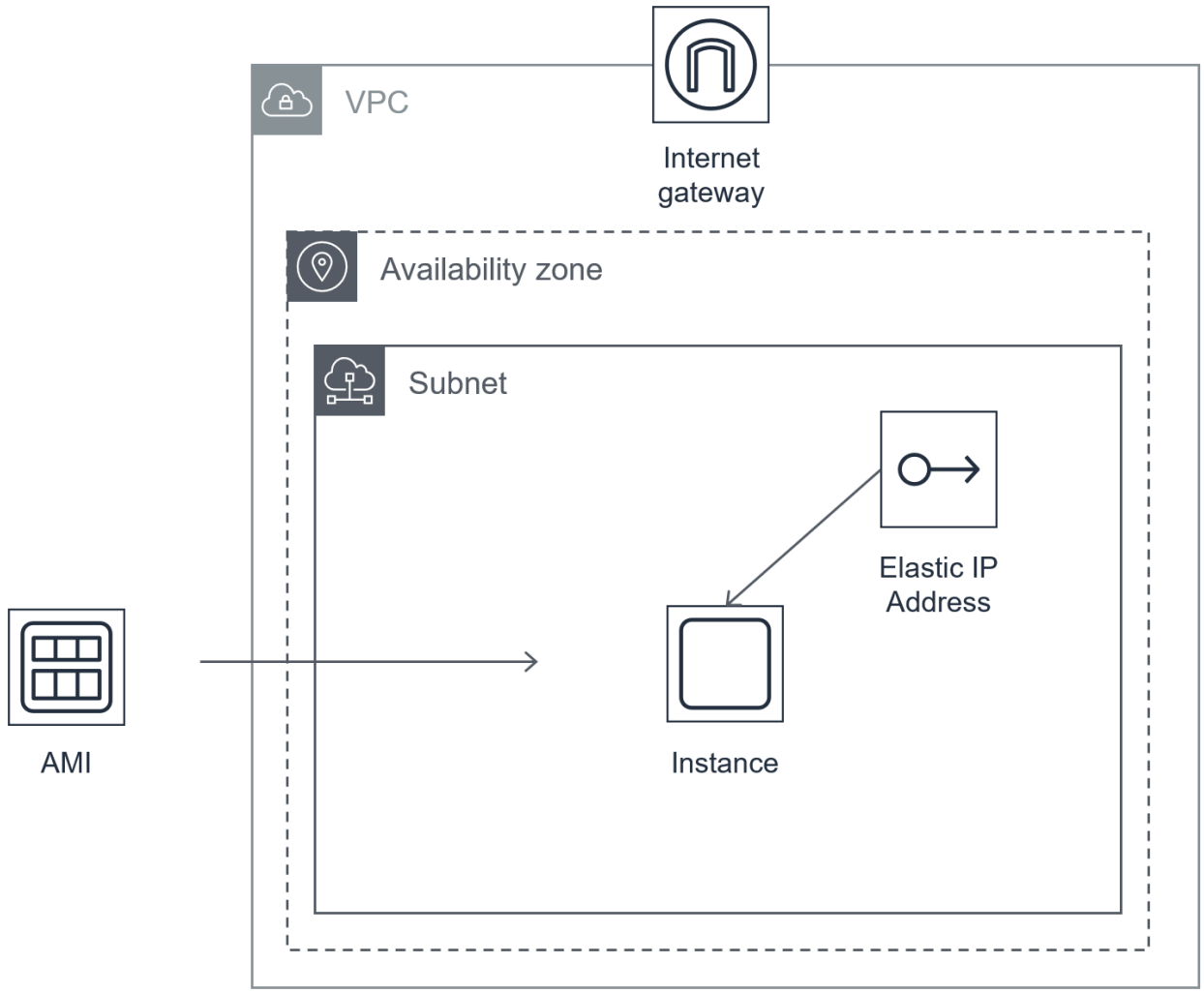
Unresolved directive in index.adoc - include::02_kubernetes_release_notes.adoc[]

Architecture Overview

An overview of the general KNIME Server architecture is supplied. More detailed description of software architecture can be found in the KNIME Server Administration Guide.

KNIME Server Small/Medium (Kubernetes)

KNIME Server Small and KNIME Server Medium run as a single pod in a single subnet. You will need to use a since it simplifies the update/upgrade procedure.



KNIME Server Large (AWS)

Security

Detailed descriptions of general considerations for KNIME Server security configuration are described in the KNIME Server Administration Guide

Audit trail

KNIME Server log files are accesible via the KNIME Server AdminPortal, or by accessing the files in their standard locations, as described in the KNIME Server Administration Guide

Configurations specific to KNIME Server running on Kubernetes are described below.

Access to KNIME Server instance

Root credentials are not required to access the KNIME Server instance.

IAM Roles and policies

IAM roles/policies that allow access to launch EC2 instances, and manage EBS volumes are required to launch the KNIME Server. It is assumed that a VPC with an internet gateway is configured and available.

Authenticating with AWS

KNIME Server does not require to authenticate with any AWS provided services.

Keys and rotation policies

An SSH key for access to the KNIME Server instance is created or chosen, via the AWS Management Console, or CLI at instance launch. You are responsible to manage this key as per the recommendations set within your organisation.

EC2 security groups and VPC access control lists

The default security group allows access to the KNIME Server via HTTP, and HTTPS on ports 80 and 443. Additionally advanced admin access via the SSH port 22 is enabled.

No VPC access control lists are defined.

Data encryption configuration

It is recommended that you enable EBS encryption and EBS snapshot encryption for all KNIME Server volumes. Full details available in the AWS documentation.

Tagging resources

You may wish to tag the EC2 instances and volumes for KNIME Server in order to identify e.g. owner, cost centre, etc. See the AWS Tagging Strategy document.

RabbitMQ TLS/SSL

Data locations

We use a 'generic' approach to describe file locations, and then follow that with the default settings used on our cloud marketplace offerings.

`<knime-server-repository>` **/srv/knime_server**

This directory contains both permanent and ephemeral data for the KNIME Server, for example configuration information, workflows, temporary data. You may wish to backup the whole directory, or if you want a more fine grained approach, you should read on. Some parts of this directory are used by processes which would likely benefit from the provisioning of good IO performance. For details on disk choices including IOPS and size for cloud, see section Sizing.

`<apache-tomee>` **/opt/knime/knime-server-4.8.2/apache-tomee-plus-7.0.5**

All data required by the Apache TomEE installation. Including settings for authentication, security and logging. Backup of this directory is recommended. Note that this was /opt/apache-tomee-plus-7.0.5 for versions < 4.7.2

`<knime-executor>` **/opt/knime/knime-3.7.2**

A symbolic link to the latest KNIME Executor (located in the same directory). Contains the executor executable, and all installed extensions. Backup of this directory is recommended. Note that this was /opt/knime-latest for versions < 4.7.2

For a more detailed definition of what type of data is stored in each location, please read on.

Permanent Data

`<knime-server-repository>/config` **/srv/knime_server/config**

Configuration information specific to the KNIME Server. For example the knime-server.config, knime.ini and the customization profiles. Backup of this directory is recommended.

`<knime-server-repository>/extensions`

Contains some additional information required for extensions to the KNIME Server, such as the preview version of the Workflow Hub. Backup of this directory is recommended.

`<knime-server-repository>/jobs`

Stores information about running jobs. Practically the jobs are a copy of a workflow from the workflows directory, that then have additional runtime data. Backup of this

directory is recommended. Increasing IO provisioning for this directory will help speed up creation of new workflow jobs, or swapping jobs to/from memory.

`<knime-server-repository>/licenses`

Stores the license file required by the KNIME Server. Backup of this directory is recommended. If you need a license file, contact your KNIME customer care representative, or sales@knime.com

`<knime-server-repository>/trash`

The location of workflows or data files that have been moved to the KNIME Server 'Recycle Bin'. You may still want to backup this directory to ensure that restore of accidentally deleted files is possible.

`<knime-server-repository>/workflows`

The store of all workflows that are uploaded to the KNIME Server. Additional metadata such as the permissions on the workflows, and their OpenAPI specification are stored in this directory. Backup of this directory is recommended. Increasing IO provisioning for this directory will help speed up creation of new workflows, and creation of new jobs.

Ephemeral data

`<knime-server-repository>/runtime`

Stores information required for locally running executors. This directory will not be used if the distributed executors feature is used. Backup of this directory is not required. It will be regenerated as required. Increasing IO provisioning for this directory will help workflow execution time, especially in the case of 'IO-bound' workflows.

`<knime-server-repository>/temp`

This directory is used as a temporary store for the TomEE process of the KNIME Server. E.g. when downloading large files from the server. Backup of this directory is not required.

Sizing

There is no 'one size fits all' answer to questions around sizing of deployments. The answer will vary depending on your typical workload, number of concurrent users, desired calculation time, and so on. We provide some recommendations to help get started.

KNIME Server Small/Medium

Compute considerations

The most compute intensive part of the KNIME Server, is executing workflows. As an example, if you expect 10 concurrent consumers to execute the same analysis workflow on the KNIME Server at approximately the same time. The workflow requires approximately 2GB of RAM, and executes in a reasonable amount of time using 2 cores. To run the workload on a single executor would require 20 GB RAM and 20 cores.

In addition you should reserve up to 4 cores, 4GB RAM for the TomEE Server process, which is primarily responsible for sharing workflows and data.

Storage considerations

The TomEE Server needs a minimum of 30 GB for the operating system, and application itself. Since the TomEE Server also hosts the KNIME Server Workflow Repository, a minimum of 250 GB additional storage is also recommended for storing workflows, and additional runtime information.

Storing a larger number of workflows, data, or jobs will naturally require more storage.

For more details on which disk locations store which kind of information, please see the section Data locations. The section also documents which storage locations may improve application performance as a result of increased IO provisioning.

KNIME Server Large

Since a typical deployment of KNIME Server Large will make use of the 'Distributed Executors' feature, there are some differences to the considerations needed when sizing a deployment.

Compute considerations

TomEE Server

The TomEE Server is responsible for managing interactions with the KNIME Server repository. Therefore, when serving a large number of workflows in parallel, or when keeping a large number of jobs in the workflow repository the size of this server may need to be increased. When using distributed executors the TomEE Server will consume four cores from the KNIME Server license. In the majority of setups it will be sufficient to reserve 4 cores to the TomEE Server. A default installation assigns 2GB RAM to the TomEE process, although it may be sensible to increase the RAM available to TomEE to 4-8 GB.

RabbitMQ

RabbitMQ is used as the message broker for Server-Executor communication. The amount of traffic through the queue is fairly limited. For that reason it is possible to reserve only 1 CPU core and 500Mb RAM to this service. In some deployments it may be desirable to install that software onto the same machine as the TomEE Server.

Executors

To support execution of a larger number of workflows it is possible to launch more than one executor. The minimum size of a single executor should be decided by considering the CPU and RAM requirements for executing a typical workflow, and the desired workflow parallelism.

Consider the following example. You expect 20 concurrent consumers to execute the same analysis workflow on the KNIME Server at approximately the same time. The workflow requires approximately 2GB of RAM, and executes in a reasonable amount of time using 2 cores. To run the workload on a single executor would require 40 GB RAM and 40 cores. There is a small RAM overhead for an executor process to run ~1-2GB.

If the number of users now doubled, it would be possible to either increase the size of the executor machine (double the size), or to launch a second executor, of the same size as the first executor.

One clear benefit of using a larger number of executors, is that this gives flexibility to add/remove executors in the case of changing executor demand. This needs to be weighed against the limited additional RAM requirement for running an executor.

Storage considerations

TomEE Server

KNIME Server Large has the same storage considerations as KNIME Server Small and Medium. See the section Storage considerations for full details.

RabbitMQ

RabbitMQ requires a minimum of 200 MB free space, and can typically be launched with a root volume of 50 GB.

Executors

KNIME Executors require a minimum of 30 GB for the operating system, and the application itself. An amount of temporary storage space is also required. Since execution of some KNIME workflows can be IO bound (especially when limited RAM is available), it is recommended that the executors have access to SSD-class storage.

Sizing (Kubernetes)

Single images for KNIME Server Small/Medium, as well as Server and Executor images for KNIME Server Large are available from a private KNIME repository. Contact cloud@knime.com for more information.

Container resource provisioning

Persistent storage

The section Data locations documents where data is stored, and can help with decisions about how to handle persistent data.

Costs

The costs of running a KNIME Server will vary depending on a number of factors. These include the desired workflow execution performance, the amount of data that is planned to be stored, the backup strategy, and planned failover setup.

Your KNIME customer care representative would be happy to offer you advice and guidance on how those choices will affect your setup. Below we provide some information on typical setups to give some idea of pricing.

Software pricing

The software pricing for the KNIME Server is listed on the KNIME website.

Questions regarding licensing should be directed to sales@knime.com.

Hardware pricing

Since costs will vary between data centers, and managed services, we have not estimated costs for hosting or subscribing to the hardware required to run a Kubernetes cluster capable of supporting KNIME Server. We refer you to the section Sizing (Kubernetes) for details on suggested sizing, and Recommended Kubernetes deployment (KNIME Server Small/Medium) for details on possible deployment options.

Deployment

Deployment of KNIME Server Small, and KNIME Server Medium is via a single virtual machine. KNIME Server Large offers several options for High-availability (HA) deployments.

KNIME Server installation

Topics surrounding KNIME Server installation are covered in detail with the KNIME Server Installation Guide. In addition we offer pre-configured images via Marketplaces which are covered in this guide.

Testing the deployment

Simple testing of the deployment can be done by logging into KNIME Server WebPortal via the web browser. Certain functionality is only available to test via the KNIME Analytics Platform.

Connecting via the browser

Once you have launched the KNIME Server AMI, the resulting instance will automatically start the KNIME Server. The KNIME Server WebPortal is available in the browser at `https://<public-hostname>/knime`

Connecting via the Analytics Platform

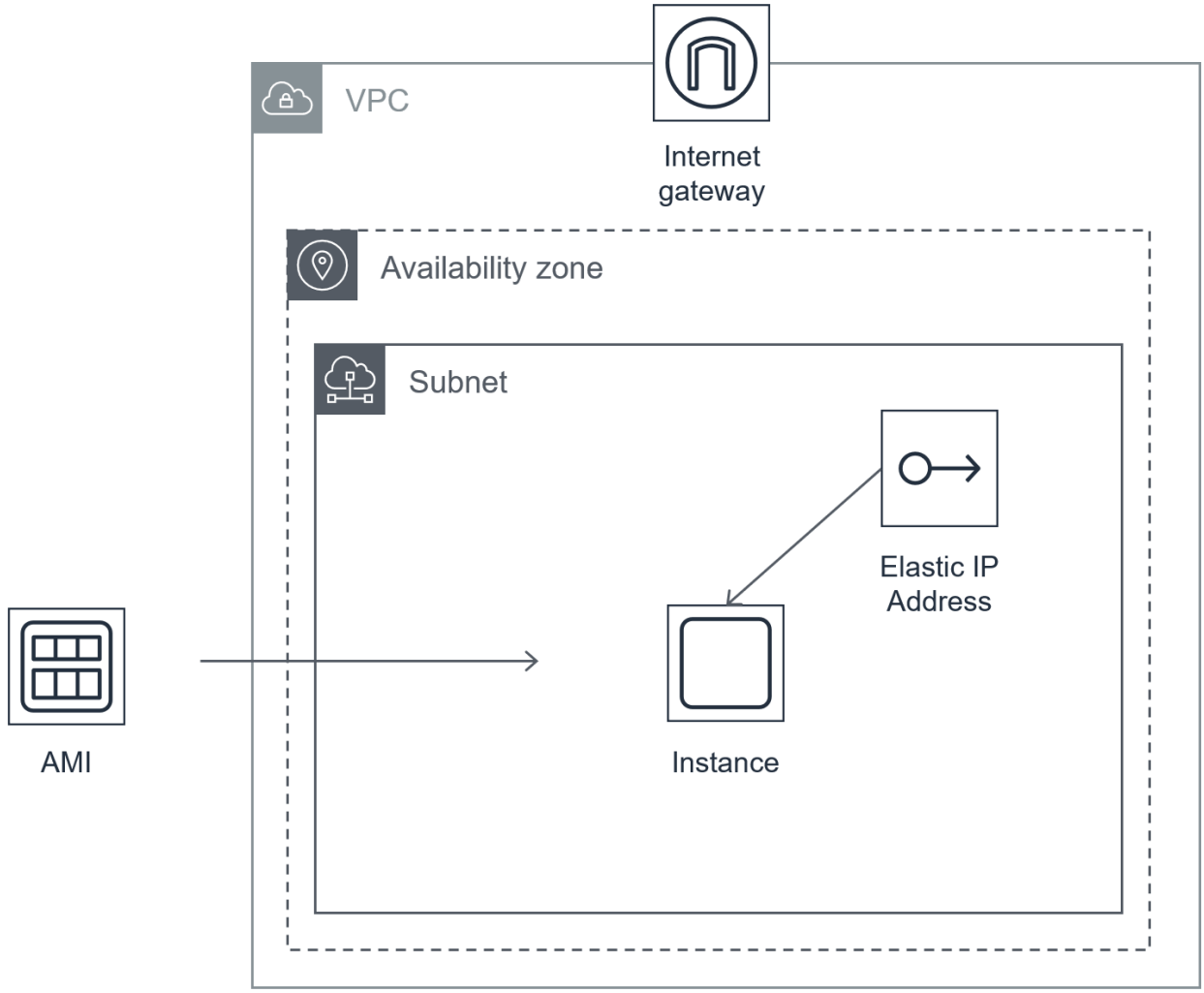
Access to the KNIME Server from the KNIME Analytics Platform is via the KNIME Explorer. Full documentation is available in the KNIME Explorer User Guide Use the mountpoint address: `https://<public-hostname>/knime`

Testing workflow execution

Click on any workflow from the WebPortal repository tree, and wait for the page to load. If the 'Start' button appears then workflow execution is working as expected. For automated testing strategies, see the section: Operations.

Recommended Kubernetes deployment (KNIME Server Small/Medium)

A typical deployment as per the sizing guidelines in the previous Sizing section looks something like:



Deployment with plain Docker

In some circumstances it may be desirable to use plain Docker to manage a single container deployment.

```
`docker run -d knime/knime-server:latest`
```

Deployment with docker-compose

Issue the following command from the directory containing the docker-compose file.

```
`docker-compose up`
```

Deployment with Kubernetes

```
`kubectl`
```

N.B. It is possible to use the docker-compose file, and Kompose to launch on Kubernetes. But it is generally preferred to use the supplied HELM charts since they allow more transparent control of your deployment.

Recommended Kubernetes deployment (KNIME Server Large)

A typical deployment as per the sizing guidelines in the previous Sizing section looks something like: image::03_simple-arch.png[]

Default KNIME Server password

We do not set a fixed default password for the KNIME Server since this is a known bad security practice. Therefore the default password is set to the instance ID of the AMI on the first launch of the KNIME Server. This can be found on the AWS console, or by issuing the command from the KNIME Server AMI instance:

```
`curl http://169.254.169.254/latest/meta-data/instance-id`
```

On first login, you are recommended to create a new admin user, and then remove the `knimeadmin` user.

Applying license file to KNIME Server on Kubernetes

For KNIME Server (BYOL) you will need to apply your license file. This can be done by visiting `https://<public-hostname>/knime` from your web browser.

Logging in using the admin username: knimeadmin, and password: <instance-id>, will redirect you to the license upload page. Here you can apply your license file. A valid license file will be immediately applied, and you can begin to use all KNIME Server functionality.

You can find the <instance-id> in the AWS Console (EC2 page) for the required instance. Or you may login to the instance via SSH and issue the command:

```
curl http://169.254.169.254/latest/meta-data/instance-id
```

Templated deployment

Especially in the case of KNIME Server Large, where there are potentially multiple instances that need to be deployed, with additional settings such as networking, security groups and access policies, we strongly recommend considering the use of templated deployments. Templated deployments can ensure that no important settings are missed, and that the deployment can be identically replicated when necessary.

Templating engine

In this documentation we describe a methodology using the 'native' templating tool for your cloud platform of choice, e.g. CloudFormation for AWS or Azure Resource Manager (ARM) for Azure. You may also wish to consider a third party tool such as Terraform. We don't show specific examples using Hashicorp Terraform, but translation from the native examples should be straightforward.

Templated VM build

In the case that you choose to build your own image, you almost certainly want to automate that process for the same reasons as wanting to automate the infrastructure deployment. It's possible to this using a tool such as Hashicorp Packer.

In addition to Marketplace images for KNIME Server (Small, Medium, Large, or BYOL) you have the option to install and configure KNIME Server from scratch. In this case the installation process is described in the KNIME Server Installation Guide. Additional configurations of KNIME Server are described in the KNIME Server Administration Guide.

In the case where you want to automate the build of a KNIME Server 'Golden Image' you should consider the above two documents as the required information. It will then be necessary to adapt your internal build process to follow this procedure. None of the tools mentioned are required, and you may choose to use alternatives.

We describe in brief detail the steps that we follow at KNIME to build the Azure Marketplace images. We use the tool Hashicorp Packer to automate the process.

The description is not intended to be an exhaustive list, but an overview of the kinds of things that you will need to consider.

Packer steps for KNIME Server Small/Medium

We follow steps such as:

- Define 'base' image. We choose Ubuntu 18.04 LTS.
- Apply latest OS patches
- Upload configuration files (preferences.epf, knime.ini, license.xml, autoinstall.xml, etc)
- Create VM user (knime)
- Install required dependency (Java JDK 8)
- Run automated KNIME Server installer
- Install optional dependencies (Python, R, Chrony, etc)
- Configure Port Forwarding/Firewall or Front-end webserver
- Cleanup image and generalize

Packer steps for KNIME Server Large

Follow the steps for KNIME Server Small/Medium to build the 'server' image. You may wish to disable the parts of the build that install the executor. Then follow steps such as:

- Define 'base' image. We choose Ubuntu 18.04 LTS.
- Apply latest OS patches
- Upload configuration files (knime.ini, executor launch script)
- Create VM user (knime)
- Download and extract KNIME Server Executor
- Install optional dependencies (Python, R, Chrony, etc)
- Cleanup image and generalize == AWS CloudFormation template deployment

Whilst it is easy and convenient to deploy the KNIME Server through the AWS Console, there are strong arguments for using a templated deployment using CloudFormation templates.

AWS CloudFormation templates allow to describe the full state of the final deployment such that it may be redeployed identically in the future. Similarly it is possible to alter a master template which allows for easy reuse of shared configurations, and customizations where required.

For an overview of the types of deployments possible with a CloudFormation template see the AWS documentation Quickstart templates.

Creating a CloudFormation template

The CloudFormation template describes the AWS infrastructure that is required to be launched for the KNIME Server.

Below is an example template that could be used as the starting point for a simple deployment.

arm_template_example

Example CloudFormation template

Operations

As part of any KNIME Server deployment you should consider monitoring your service for availability. KNIME Server has several endpoints that can be used to determine the system health.

Application fault

A simple REST call to the deployed KNIME Server should always return a 200 response with a payload similar to:

```
curl https://<public-hostname>/knime/rest
```

rest_response

```
{
  "@controls" : {
    "self" : {
      "href" : "https://<public-hostname>/knime/rest/",
      "method" : "GET"
    },
    "knime:v4" : {
      "href" : "https://<public-hostname>/knime/rest/v4",
      "title" : "KNIME Server API v4",
      "method" : "GET"
    }
  },
  "version" : {
    "major" : 4,
    "minor" : 8,
    "revision" : 0,
    "qualifier" : ""
  },
  "mountId" : "<public-hostname>",
  "@namespaces" : {
    "knime" : {
      "name" : "http://www.knime.com/server/rels#"
    }
  }
}
```

A different response indicates a configuration issue, or application fault.

It is also possible to test for executor availablity. This requires authenticating against the KNIME Server and calling the following REST endpoint.

```
curl -X GET "https://<public-hostname>/knime/rest/v4/repository/Examples/Test Workflows
(add your own for databases)/01 - Test Basic Workflow - Data
Blending:execution?reset=true&timeout=300000" -H "accept:application/vnd.mason+json"
```

Container fault

An container fault can be detected using the standard AWS techniques.

Security certificate expirations

Certificate expiration will be caught if the basic server check fails with an HTTP 400 status code.

Backup and Recovery

Backup

KNIME Server can be backed up subject to the information available in the KNIME Server Administration Guide.

Important data locations are detailed in the section Data locations

Typically the simplest backup solution is to take a snapshot of the OS volume, and a second snapshot of the data volume.

Recovery

When using the 'whole volume snapshot' backup method mentioned above, restoration of the system is best done by launching a new instance from the snapshot images.

Backup (Kubernetes)

Recovery (Kubernetes)

Routine Maintenance

Starting KNIME Server

KNIME Server starts automatically when the instance starts using standard systemd commands. Once the TomEE application has started successfully, it will automatically launch and executor. This means that in normal operation you will not need the below command.

In the case that you need to start a stopped KNIME Server, it may be started using the following command at the terminal:

```
sudo systemctl start knime-server.service
```

Stopping KNIME Server

Stop the KNIME Server by executing the command:

```
sudo systemctl stop knime-server.service
```

Restarting KNIME Server

Restart the KNIME Server by executing the command:

```
sudo systemctl restart knime-server.service
```

Bootnote, for versions older than KNIME Server 4.7

Note that starting, stopping and restarting differs from version 4.7 and older of KNIME Server, where knime-server.service was replaced with apache-tomee.service

Restarting the executor (KNIME Server Small/Medium/Large)

It is possible to restart the executor by issuing the following command:

```
sudo -u knime touch /srv/knime_server/rmirestart
```

This will launch a new executor, leaving the existing executor running. All existing jobs will continue to run on the old executor, and all new jobs will be launched on the new executor. That is helpful when updating executor preference files without needing to interrupt existing running jobs. When the rmirestart file is automatically deleted, the new executor has been launched.

It is possible to perform a hard kill on a running instance, by issuing the command:

```
sudo -u knime kill -9 <PID>
```

where <PID> is the process ID of the running executor. You can find the <PID> by running:

```
ps aux | grep knime
```

and looking for the process(es) that are not the apache-tomee instance.

Restarting the executor (KNIME Server Large - Distributed Executors)

In most cases you will want to restart the entire instance that is running the Executor. But in certain cases you may wish to do this by restarting the Executor application itself. That can be acheived either by stopping the executor process, and starting again from the terminal. Or by restarting the systemd service, if it is being used.

```
sudo systemctl restart knime-executor.service
```

Managing Certificates

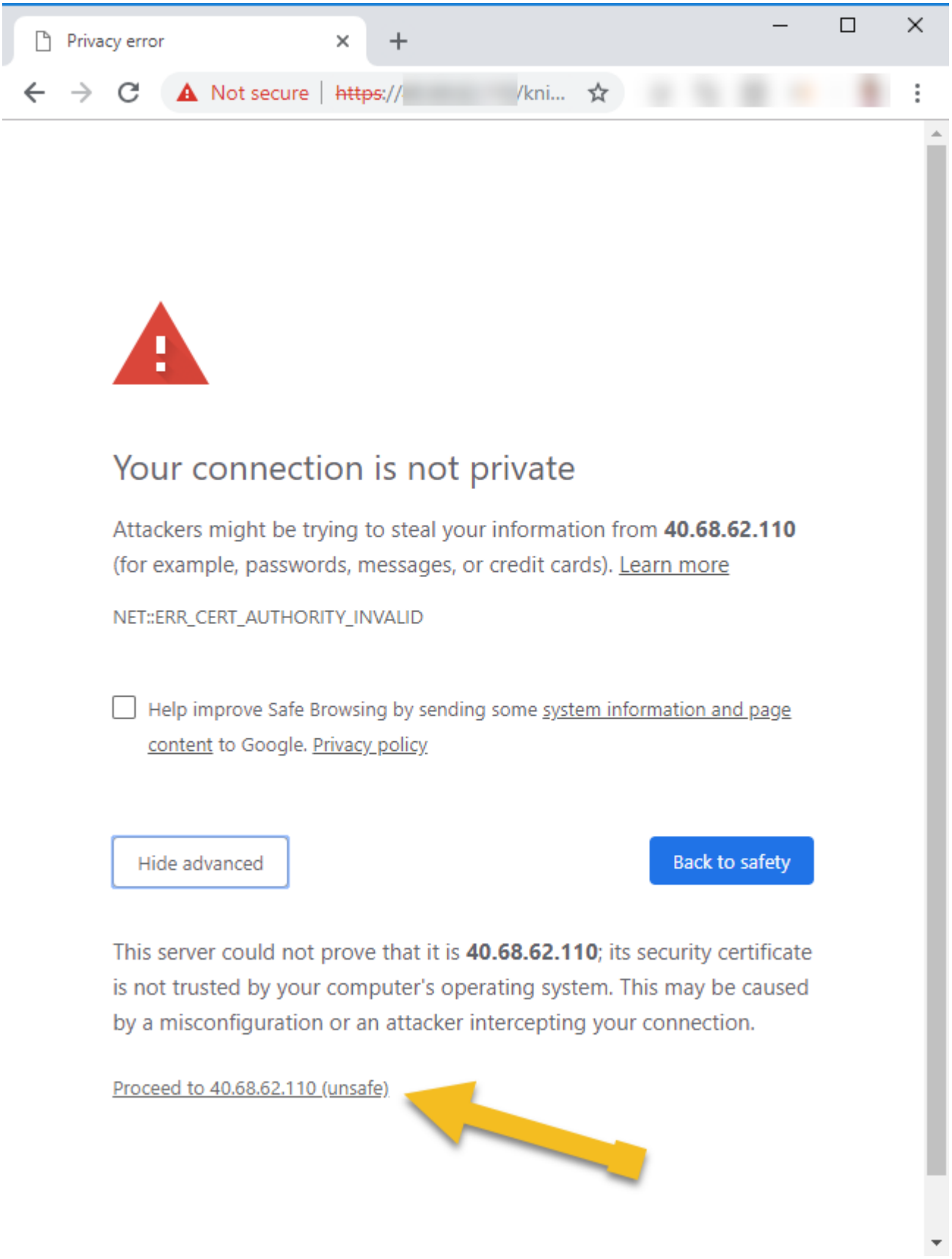
Detailed steps for managing the SSL certificate for KNIME Server can be found in the KNIME Server Administration Guide

Default Certificates

KNIME Server ships with a default SSL certificate. This allows for encrypted communication between client and server. However, since the certificate cannot be generated in advance for the server that you are running on, it will not be recognised as a valid certificate. Therefore, we recommend managing your own certificate as per the guidelines in the Managing Certificates.

When testing with the default certificate, modern browsers will issue a warning as below.

Choosing to ignore the warning, will allow you to access the KNIME WebPortal for testing.



Update Python configuration

We use Anaconda Python to define a default python environment. The current yaml file can be found in /home/knime/python/py36_knime.yaml.

For detailed documentation on managing Anaconda, please refer to the Anaconda documentation.

An example yaml file is shown below. We chose packages that we know are well used, or are required for the use of the KNIME Deep Learning package. Mostly we have pinned version numbers to ensure compatility. You may choose to unpin version numbers. Additionally you may wish to add a new Python package, in which case you can add the package to the yaml file and run the command:

```
sudo -u knime /home/knime/python/anaconda/bin/conda env update -f
/home/knime/python/py36_knime.yml --prune
```

py36_knime.yml, source,yaml

Apply Operating System patches

The KNIME Server 4.9 AMIs are based on Ubuntu Server 18.04 LTS. The OS should be regularly patched using the standard Ubuntu procedures.

After a Java JDK update, the KNIME Server must be restarted.

Update KNIME Server

Updates, and patches to the KNIME Server are announced on the KNIME Server Forum. You may subscribe to the topic.

Before applying a feature update (e.g. version 4.8.2 → 4.9.0) you should read the release notes and update guide. This will document any changes to parameters, features and settings that might affect your installation.

In the case of a patch update (e.g. version 4.8.1 → 4.8.2) there should be no changes to settings required.

There are two strategies to applying feature, or patch updates of KNIME Server. The first is to follow the instructions in the KNIME Server Update Guide via the terminal (in place update). The second is to migrate a snapshot of the workflow repository block device to a new KNIME Server instance (disk swap update).

In place update

To make a feature update you have the option to follow the instructions in the KNIME Server Update Guide.

Install additional extensions

We install a set of extensions that we think will be useful for many situations, see the Extensions installed with Executors. If you developed a workflow locally that uses an extension not in the list, you'll need to install the extension into the executor.

KNIME Server Small/Medium/BYOL

To install a new extension.

- Stop the KNIME Server `sudo systemctl stop knime-server`
- Find the extension on KNIME Hub by searching for the node of interest, and clicking the link to the extensions page.
- Determine the `<extension-identifier>` by modifying the URL, e.g. https://hub.knime.com/knime/extensions/**com.knime.features.gateway.remote**/latest → **com.knime.features.gateway.remote**.feature.group
- Run the command:

```
 sudo -u knime /opt/knime-latest/knime -application org.eclipse.equinox.p2.director
-nosplash
  -consolelog -r https://update.knime.org/analytics-
platform/4.0,https://update.knime.com/community-contributions/trusted/4.0
  -i <extension-identifier> -d /opt/knime/knime-latest
```

If you want to install an extension from another update site, you'll need to add that to the command. You'll also need to find the `<extension-identifier>` or 'Feature Group Name' from the list of installed extensions in the KNIME Analytics Platform.

KNIME Server Large (Distributed executors)

Instructions to be added shortly.

Key rotation

Emergency Maintenance

In case of KNIME Server REST API not being available, a restart of the TomEE Server is required.

In case of the REST API being available, but the execution API not working as intended, then the executor must first be restarted, and if that doesn't work, then restarting TomEE is required.

See section Routine Maintenance for details.

Emergency Maintenance (Kubernetes)

Support

KNIME Server Small support is provided by submitting questions in the KNIME Server forum.

KNIME Server Medium, and KNIME Server Large support is additionally supplied via contacting the support@knime.com email address. When contacting KNIME Support you will need to include your name and company. We aim to respond to your question in under 48 hours.

Support costs

If you require additional support, please contact sales@knime.com for further information.

KNIME AG
Technoparkstrasse 1
8005 Zurich, Switzerland
www.knime.com
info@knime.com