Open for Innovation
KNIME

KNIME Server on Azure Marketplace

KNIME AG, Zurich, Switzerland
Version 4.16 (last updated on 2023-03-16)

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 Server User Guide
- KNIME WebPortal User Guide

An additional resource is also the KNIME Server Advanced Setup Guide.

Deployment on Azure

KNIME Server can be launched through the Azure Marketplace.

- KNIME Server Medium

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

KNIME Server Medium is a single VM instance, and is most easily launched via the Azure portal. If you are familiar with the Azure CLI, or Powershell you may also use this deployment method.

For self-build deployments using a custom base image, you should consult the KNIME Server Installation Guide.

Prerequisites

The person responsible for the deployment of KNIME Server should be familiar with basic Azure functionality surrounding configuring Azure VMs. KNIME Server administration requires basic Linux system administration skills, such as editing text files via the CLI, and starting/stopping systemd services.

KNIME Server Medium, and BYOL are single VM images and contain all software requirements.

For self-build instances, please consult the standard KNIME Server Installation Guide.

Azure resources

Launching a VM requires a new (or available) resource group. Within the resource group a Public IP address, Network Security Group, Virtual Network, Network Interface and Virtual Machine are deployed.

The default Network Security Group will enable HTTP access on port 80, and HTTPS access on port 443. SSH access to administer the server defaults to port 22.

Pre-installed software (Azure Specific)

For convenience we have installed and pre-configured:

- Azure CLI

Pre-installed software

For convenience we have installed and pre-configured:

- OpenJDK 11 (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 Tomcat running on port 8080, 8443)

Extensions installed with Executors

To install additional extensions, see the section [install-extensions].

Extensions all come from the following update sites:

- https://update.knime.com/analytics-platform/5.2
- https://update.knime.com/community-contributions/trusted/5.2

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.google.cloud.storage.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.browser.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.buildworkflows.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

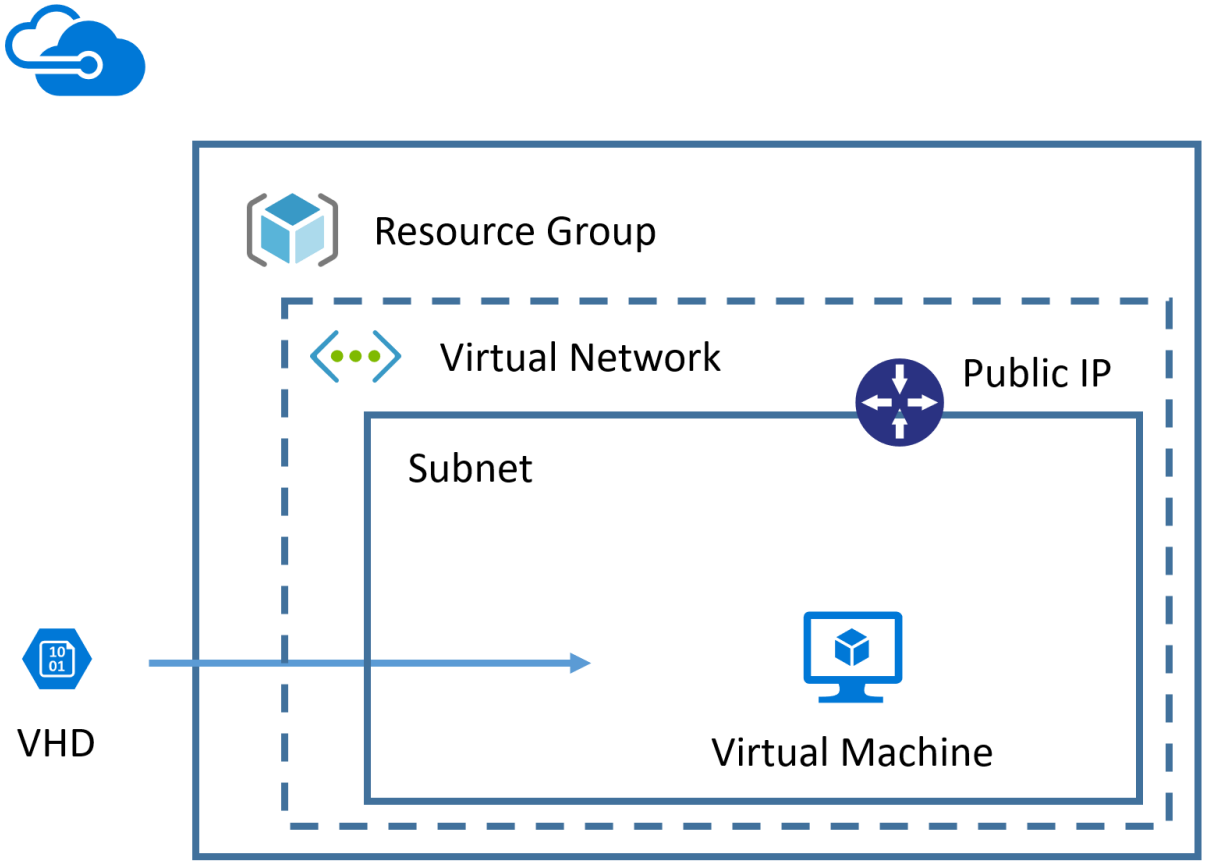
Optionally KNIME Server Large instances (currently available via BYOL license only) may choose to connect KNIME Server to an external LDAP/AD service. Full details are contained in KNIME Server Advanced Setup Guide.

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 (Azure)

KNIME Server Medium run as a single VM instance in a single subnet of a Virtual Network. Use of an Public IP is preferred since it simplifies the update/upgrade procedure.



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

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

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

Authenticating with Azure

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

Keys and rotation policies

When SSH access to KNIME Server is required, it is recommended to use an SSH keypair instead of username/password. This can be configured in the Azure portal at launch time.

You are responsible to manage this access key as per the recommendations set within your organisation.

Network Security Group access control lists

The default Network 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.

Data encryption configuration

Azure managed disks use SSE encryption of disks as default (see here). You may also wish to enable Azure Disk Encrypion (see here) for all KNIME Server volumes.

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

Tagging resources

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

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-tomcat>` **/opt/knime/knime-srv-4.11.x/apache-tomcat-9.0.x**

All data required by the Apache Tomcat installation. Including settings for authentication, security and logging. Backup of this directory is recommended.

`<knime-executor>` **/opt/knime/knime-4.2.x**

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.

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 Tomcat 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 Tomcat Server process, which is primarily responsible for sharing workflows and data.

Storage considerations

The Tomcat Server needs a minimum of 30 GB for the operating system, and application itself. Since the Tomcat 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

Tomcat Server

The Tomcat 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 KNIME Executors the Tomcat 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 Tomcat Server. A default installation assigns 2GB RAM to the Tomcat process, although it may be sensible to increase the RAM available to Tomcat 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 Tomcat 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

Tomcat 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 (Azure)

KNIME Server Medium are both sold via the Azure Marketplace with built in licenses for 5 named users, and a maximum of 4 cores for workflow execution. Additionally KNIME Server Medium allows 20 consumers to access the KNIME Server WebPortal via the web browser only. Please contact sales@knime.com if you require a larger number of users, consumers, or cores.

Azure VM Size selection

Typically workflow execution speed can benefit from additional available instance RAM. Therefore we recommend the 'Ev3' series, since they provide the best value access to RAM.

The Standard_E8_v3 instance has 64 Gb RAM available, and also 8 CPU cores, thus is the largest instance that KNIME Server Medium can make use of.

Full details of Azure VM sizes can be found here.

Disk selection

KNIME Server Small/Medium (Azure)

The default OS Disk volume is 30Gb Standard SSD or Premium SSD (depending on selected instance type), and in most cases it is not necessary to increase the volume size. The additional volume has a default size of 250Gb SSD which should be appropriate for many new installations.

Full details of Azure Disk options are available here.

To help guide the volume size, you will need to consider:

- the size and number of workflows expected to be stored
- the number of jobs executed and job retention duration
- number and size of additional files stored in the workflow repository

In case you need to add additional storage space later, please see the section Increasing Azure disk size.

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 defined in the Azure Portal. Questions regarding BYOL licensing should be directed to sales@knime.com.

Hardware pricing

Hardware pricing is defined by Azure. See Azure Pricing

Required services

- Azure Virtual Machines
- Azure Storage Disk (Managed Disk recommended)
- Data transfer in/out
- Optional: Azure Disk Snapshots

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

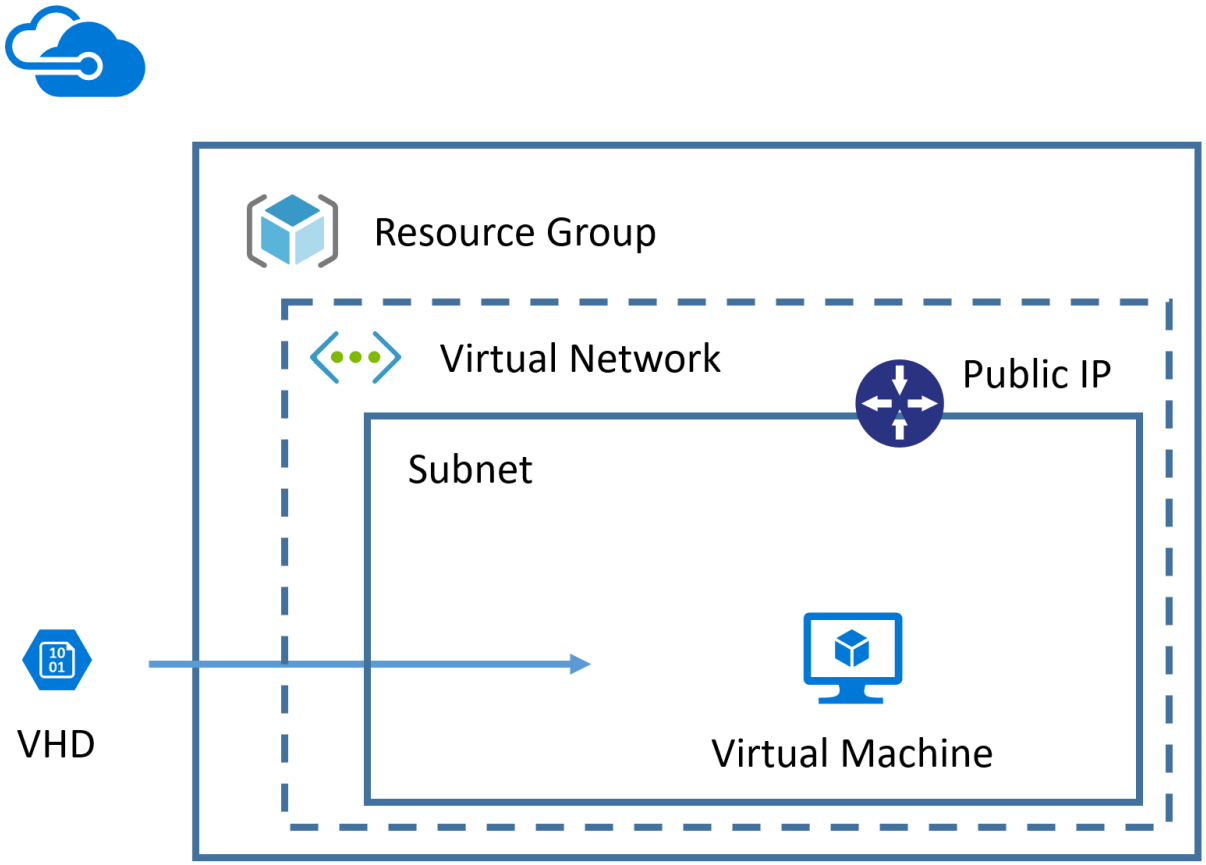
Access to the KNIME Server from the KNIME Analytics Platform is via the KNIME Explorer. Full documentation is available in the KNIME Server User Guide Use the mount point address: `https://<public-hostname>/knime`

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.

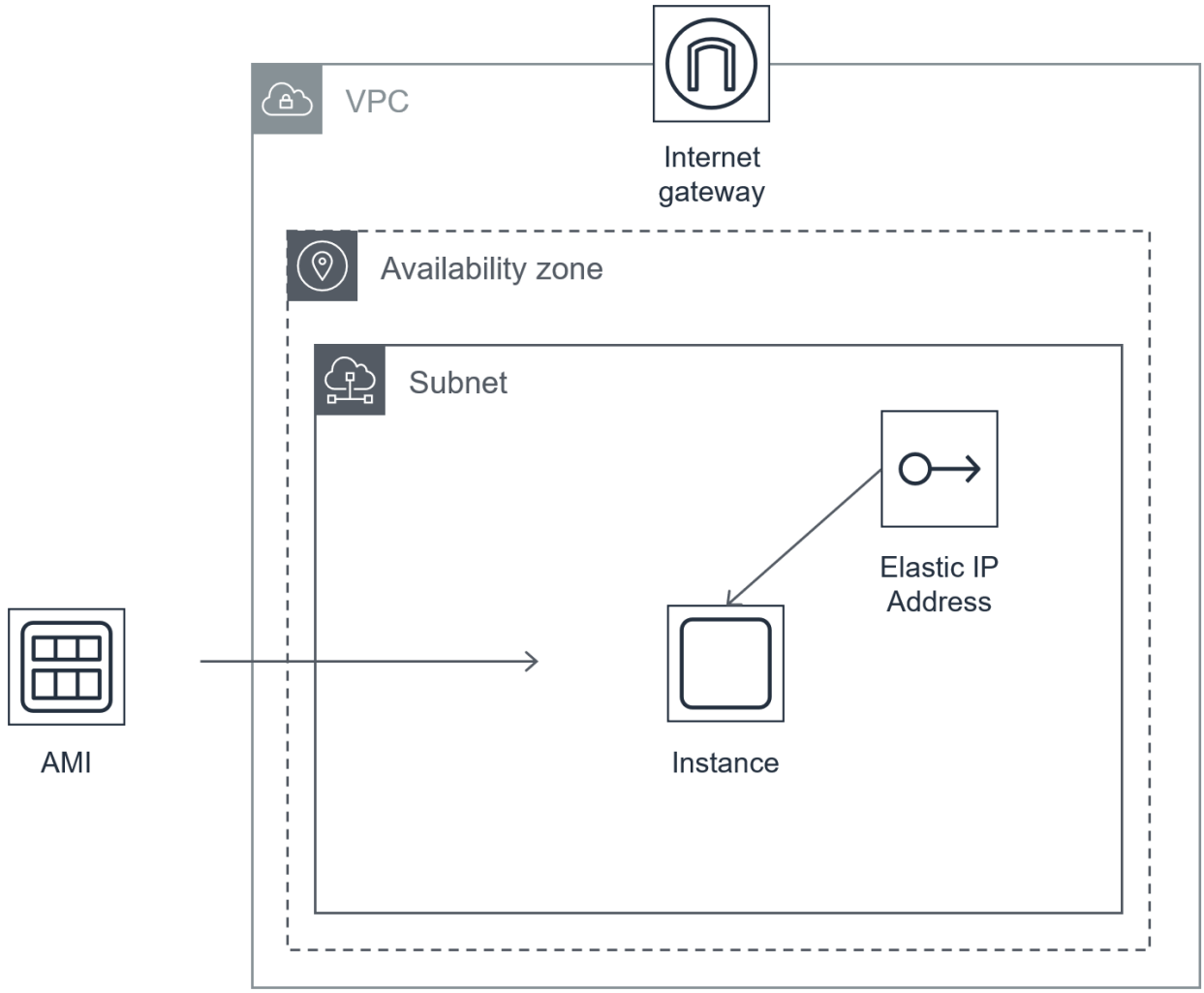
Recommended Azure deployment (KNIME Server Small/Medium)

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



Recommended Azure deployment (KNIME Server Large)

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



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 VM ID of the VM on the first launch of the KNIME Server. To recreate the password: obtain the VM ID of the VM from the Azure Portal console. When viewing the VM in the console, click on "JSON view" and copy the value in the "vmId" field.

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

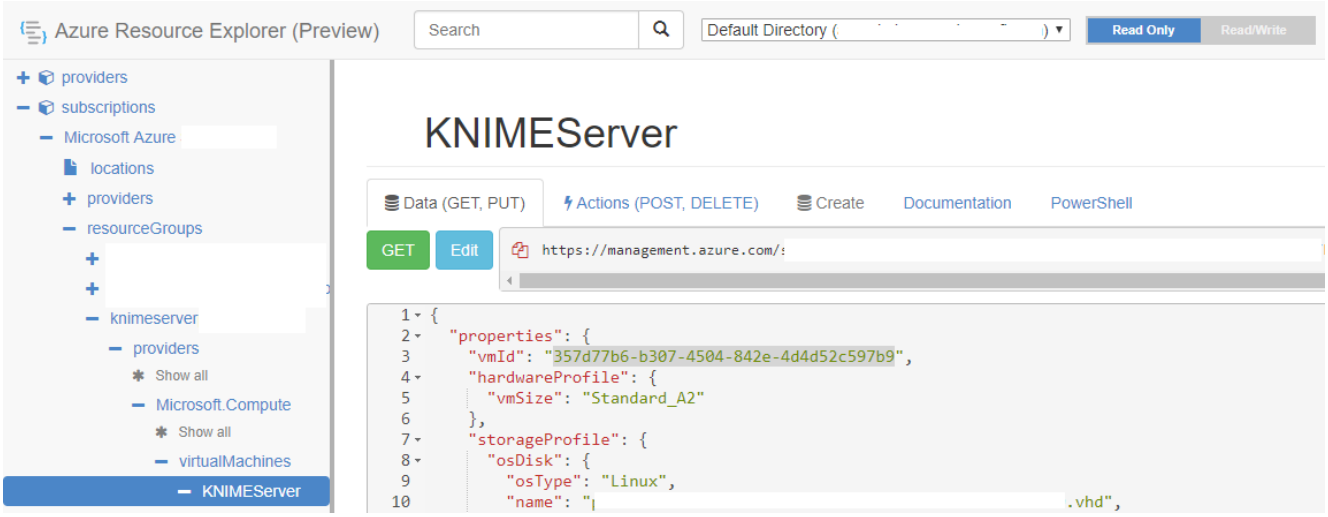
Note that this username/password is for the KNIME Server application, and is different to the KNIME Server VM username/password which can be optionally used for SSH access to the VM.

Applying license file to KNIME Server on Azure (BYOL)

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: `<default-password>`, 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.

To generate a license file, your KNIME contact will need the vmId of the VM. That can be found by browsing to https://resources.azure.com



More details on vmId can be found here.

You may also choose to login to the instance via SSH and issue the command to get the vmId:

```
curl -H Metadata:true "http://169.254.169.254/metadata/instance?api-version=2017-03-01&format=json" | jq -r '.compute.vmId'
```

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 20.04 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 11)
- 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 20.04 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 == Azure Resource Manager (ARM) template deployment

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

Azure Resource Manager (ARM) 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.

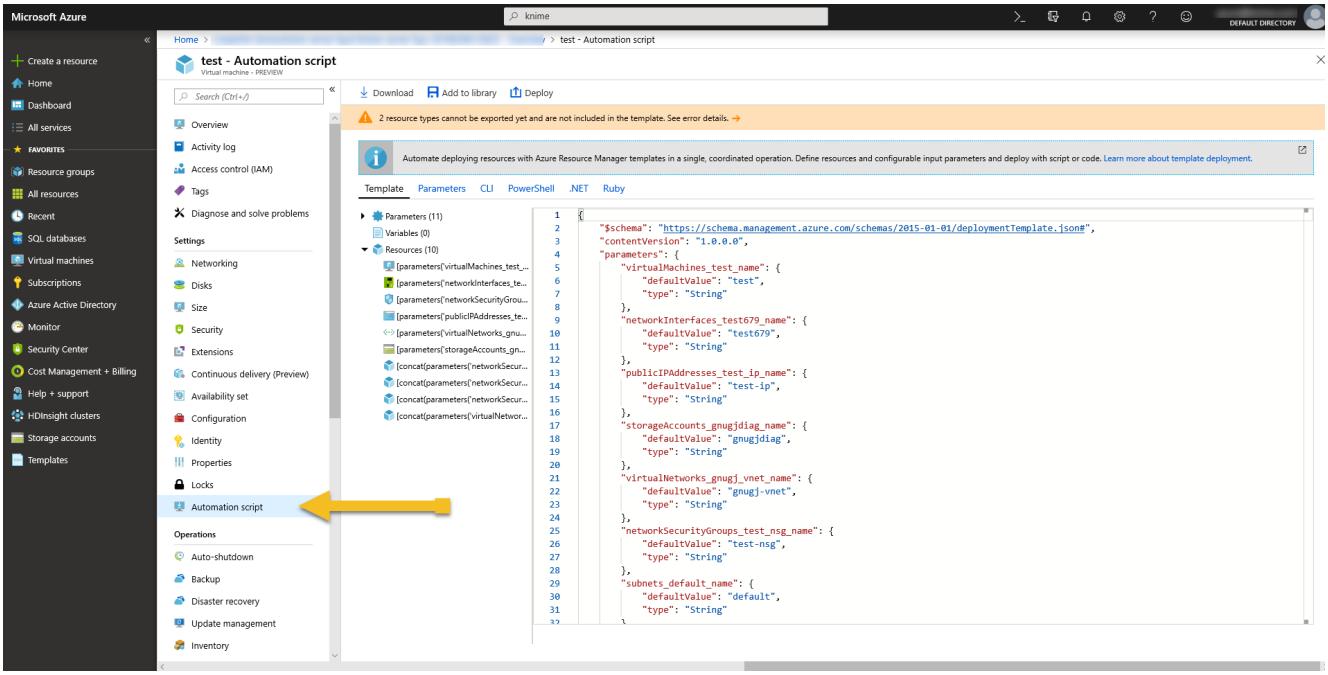
For an overview of the types of deployments possible with an ARM template see the Azure

documentation Quickstart templates.

Creating an ARM template

The ARM template describes the Azure infrastructure that is required to be launched for the KNIME Server.

If you already have an existing deployment that you deployed through the Azure Portal, it is possible to export the ARM template from the Azure Portal by navigating to the Virtual Machine, and selecting `Automation Template` as shown in the below image.



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

arm_template_example

Example ARM template

The template itself is parametrized by the following file.

arm_template_parameters_example

Example ARM template parameters

Splitting the template from the parameters allows to describe multiple deployments, e.g. dev and prod using a single template, with a parameters file for each. That can help to make sure that the setup in the development environment is the same as the production environment.

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"
```

Availability Zone fault

Since KNIME Server Small/Medium runs in a single Availability Zone an Availability Zone fault will be detected by the application fault detection method described below.

Instance fault

A VM fault can be detected using the standard Azure techniques.

Storage capacity

You may monitor the storage capacity of the Azure disk volumes (OS Disk and Data disk) using standard techniques and services such as Azure Monitor. For more details see here.

We recommend triggering an alarm at <5% free space on either volume.

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 (Azure)

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

It is recommended to make use of the Azure Snapshot functionality. See the Azure documentation section on taking Azure disk snapshots.

Recovery (Azure)

To restore an Azure Disk Snapshot, see the Azure documentation section on restoring Azure disk snapshots.

Routine Maintenance

Starting KNIME Server

KNIME Server starts automatically when the instance starts using standard systemd commands. Once the Tomcat 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-tomcat.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-tomcat 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

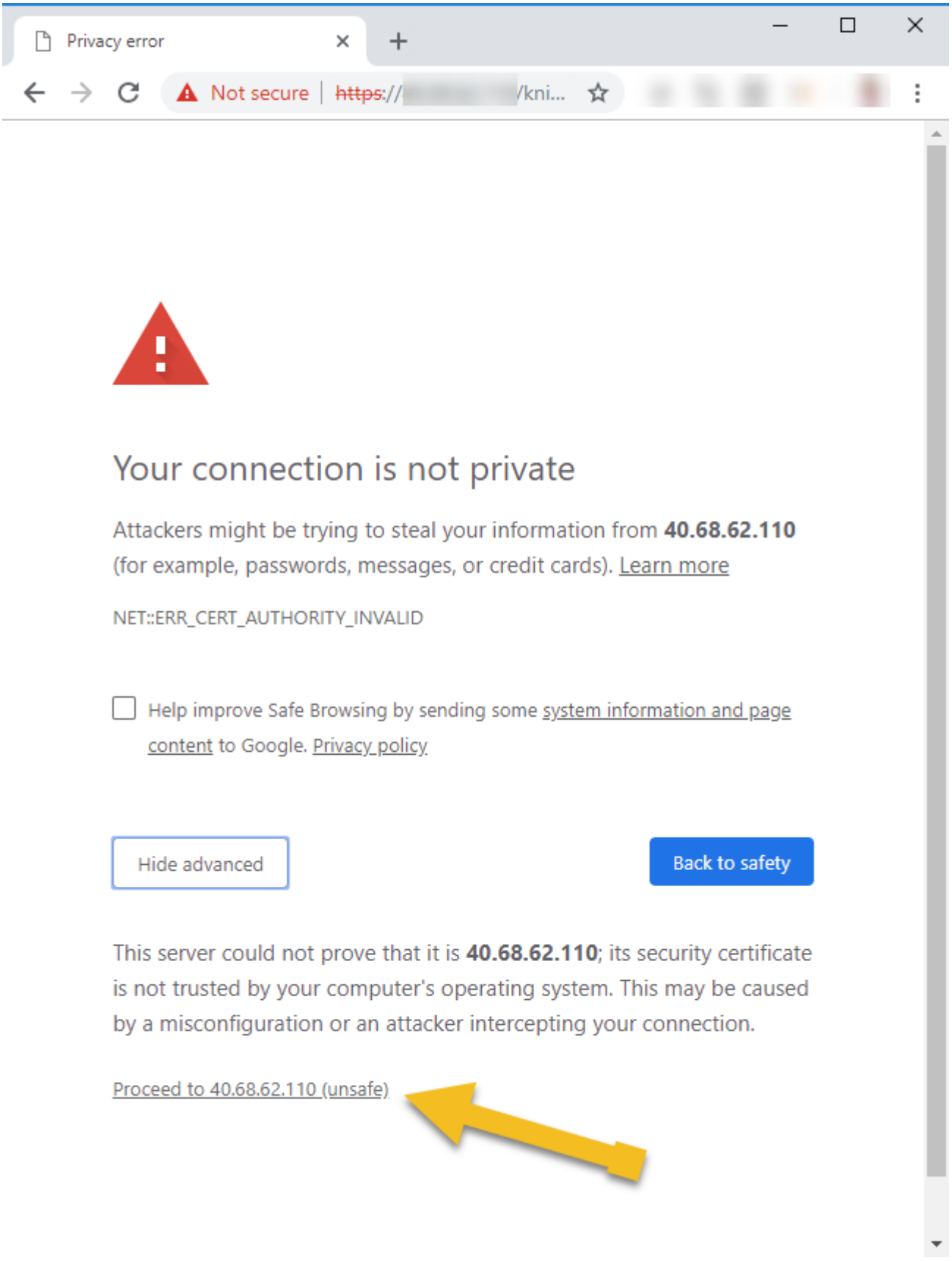
Detailed steps for managing the SSL certificate for KNIME Server can be found in the KNIME Server Administration Guide

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 4.16 AMIs are based on Ubuntu Server 20.04 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 KNIME Server 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.

SSH access to KNIME Server on Azure

Access to the KNIME Server instance via SSH follows the general instructions provided by Azure.

Connection to the KNIME Server instance is always using the user `username` which is specified at instance launch time, and with either the SSH key specified at instance launch time (recommended) or the password.

An example SSH connection string for key-based login is:

```
ssh -i <keyfile.pem> <username>@<hostname>
```

An example SSH connection string for password-based login is (upon login you will be prompted for the password):

```
ssh ubuntu@<hostname>
```

All relevant KNIME Server installation and runtime files are owned by the `knime` user. In order to make changes to these files, it is required to assume the identity of the `knime` user:

```
sudo su knime
```

Increasing Azure disk size

It is possible to increase the size of the workflow repository disk size (default size: 250 Gb) after an instance has been launched. Follow the instructions here.

Key rotation

Managing SSH keys for accessing the KNIME Server is detailed here.

Emergency Maintenance

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

See section Routine Maintenance for details.

Emergency Maintenance (Azure)

In case KNIME Server is not available due to degraded performance of an Availability Zone (AZ), VM fault, etc. It is possible to restore a snapshot and launch a new instance.

Availability Zone recovery

Availability Zone recovery is managed by launching a new instance into an unaffected Availability Zone, using a recent snapshot.

Then attach the elastic IP from the affected instance to the new instance.

Region recovery

Region recovery is managed by launching a new instance into an unaffected region, using a recent snapshot.

Then attach the elastic IP from the affected instance to the new instance.

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 Product Code, VM ID, and Azure Account ID.

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