KNIME Server on AWS Marketplace

KNIME AG, Zurich, Switzerland
Version 4.17 (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 AWS

KNIME Server can be launched through the AWS Marketplace.

- KNIME Server Medium (PAYG)

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

KNIME Server Medium is a single AMI instance, and is most easily launched via the AWS console. If you are familiar with the AWS CLI, 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 AWS functionality surrounding configuring EC2 instances. KNIME Server administration requires basic Linux system administration skills, such as editing text files via the CLI, and starting/stopping systemd services.

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

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

AWS 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. SSH access to administer the server on port 22.

Other AWS services that may optionally be used are:

- AWS KMS: To support EBS/EFS volume encryption.
- AWS EFS: Elastic File System can be used to support Server failover setup.
- AWS ELB: Load-balancer can optionally be used to support Server failover setup.
- AWS ASG: Auto-scaling group used in Server failover setup, or to support distributed KNIME Executors for KNIME Server Large.

Pre-installed software (AWS Specific)

For convenience we have installed and pre-configured:

- AWS CLI.
- boto3.

Supported regions

More detailed information about supported regions are available here.

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/5.3
- https://update.knime.com/community-contributions/trusted/5.3

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.

Release Notes

Relevant changes the KNIME Server software can be found in the Release notes and the corresponding Changelog.

Relevant changes for KNIME Analytics Platform can be found in the Changelogs.

Below we list all AWS specific changes to the AMIs submitted to AWS Marketplace.

4.9.0

- Updated OS packages

- Updated Anaconda Python package versions

- Added Boto3 to the Anaconda environment(s)

- Added symbolic link (/opt/knime/knime-latest) to point to the KNIME Executor /opt/knime/knime-4.0.0, to make disk swap updates easier.

- Changed knime-server.config to reference the KNIME Executor via the symbolic link /opt/knime-latest, to make disk swap updates easier.

- Added new extensions (AWS ML Services, ONNX, MLI, Plotly, Erlwood Base, Continental, Slack, Marvin)

4.8.2

- Updated from Ubuntu 16.04 LTS to 18.04 LTS.

- Updated OS packages.

- Updated Anaconda Python package versions (default Python is now version 3.6).

- Included Chrony to ensure server clock is correctly syncronised to AWS internal resources.

- Changed location of Server and Executor (/etc/fstab) in /opt.

- Added name tag to Server repo block device to support easier updates.

- Minor bugfixes.

4.7.2

- New image: KNIME Server Small (incl. 30 day free trial)

- Updated OS packages.

- Added Anaconda Python.

- Minor bugfixes.

4.6.4

- Updated OS packages

- Updated System packages.

- Updated Python package versions.

- Minor bugfixes.

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 (AWS)

KNIME Server Small and KNIME Server Medium run as a single EC2 instance in a single subnet of a VPC. Use of an elastic IP is preferred since it simplifies the update/upgrade procedure.
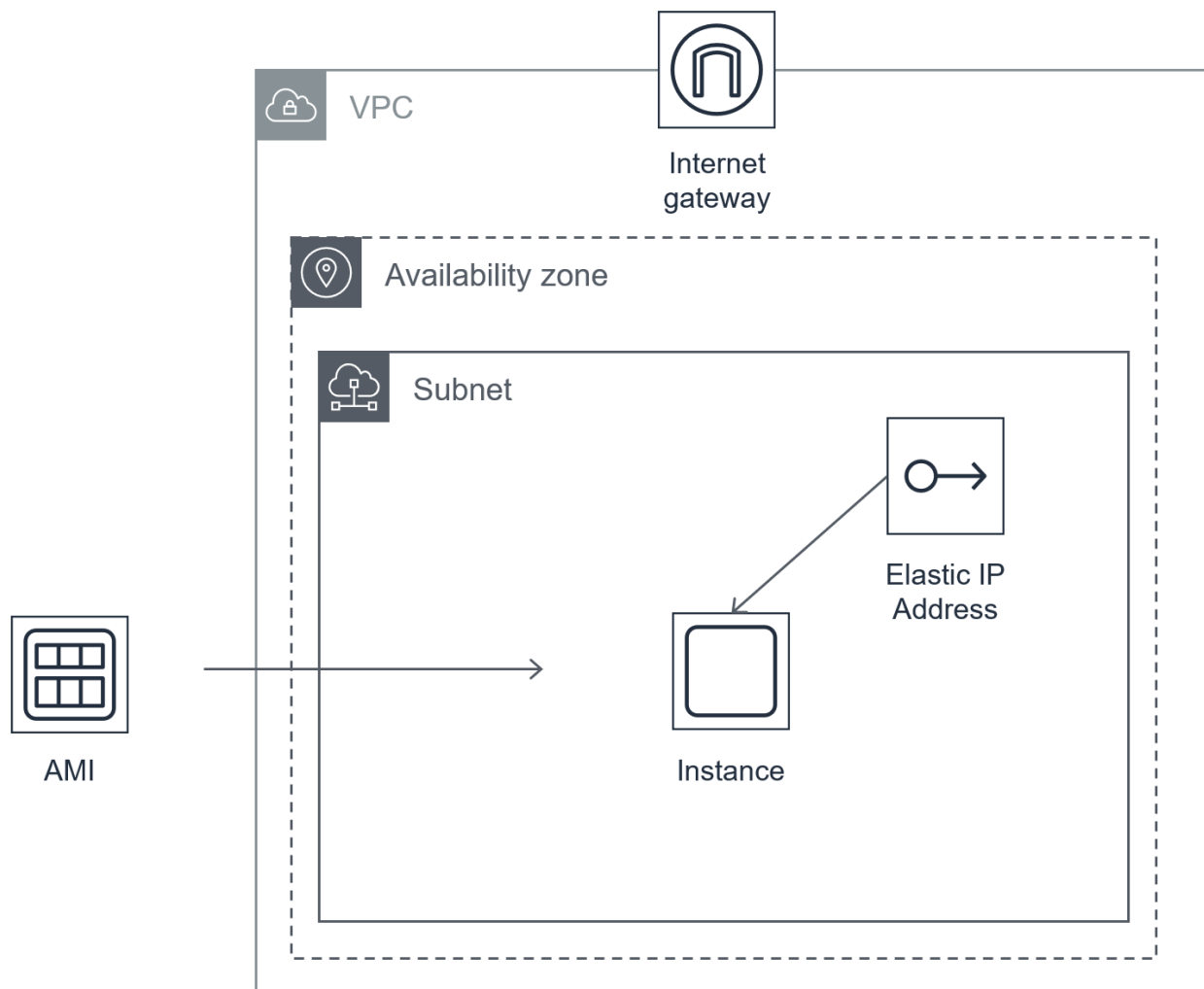
KNIME Server Large (AWS)

KNIME Server Large supports configurations that allow for resilient architectures, to support scalability and high-availability. For detailed discussions on those topics see the

Recommended AWS deployment (KNIME Server Small/Medium/Large (via BYOL)) section. For reference we've shown here one of the more complex architectures to support scaling KNIME Executors, and failover of the server instance to a new availability zone.

KNIME Server Large can be run as a single EC2 instance, in which case all of the guidance from above may be used.
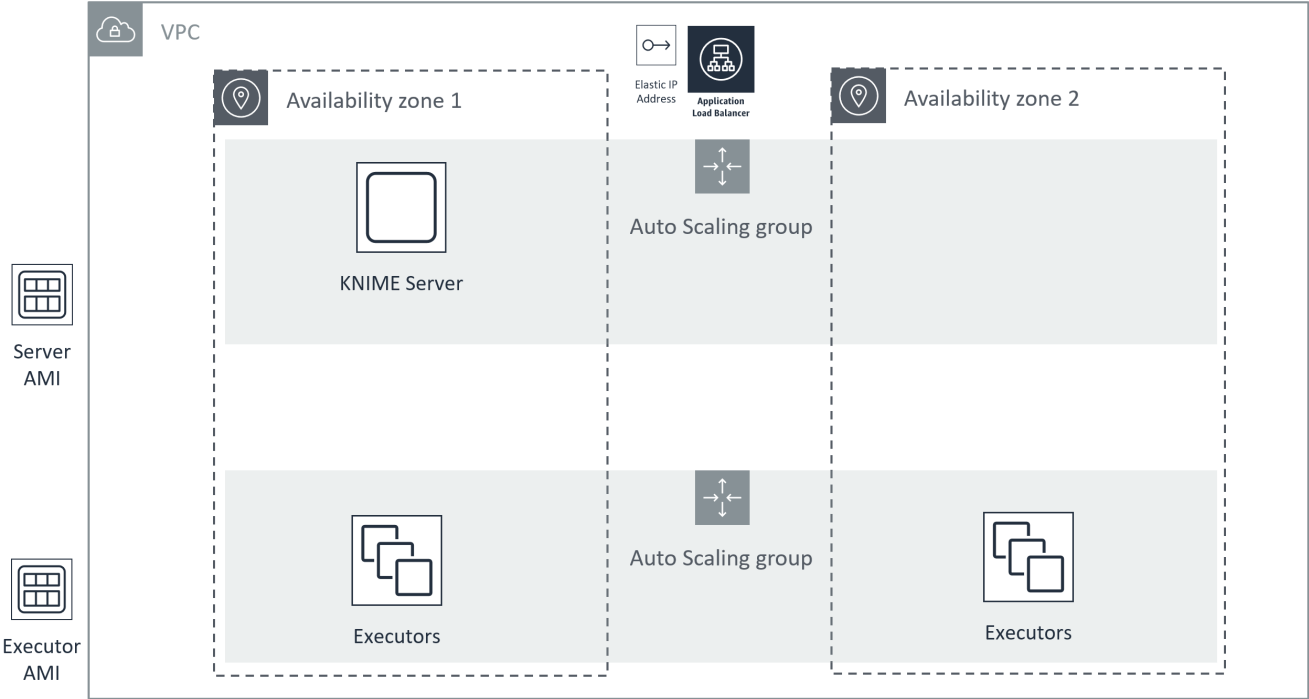
Security

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 AWS 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. The default setting does not generate any IAM roles and policies for KNIME Server and doesn't grant public access to any resources. Users can generate the IAM role for the EC2 instance of KNIME Server with necessary access based on their use cases, e.g. S3 access.

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 organization.

Stored secrets

Users do not need to create or manage any stored secrets when deploying the solution.

EC2 security groups and VPC access control lists
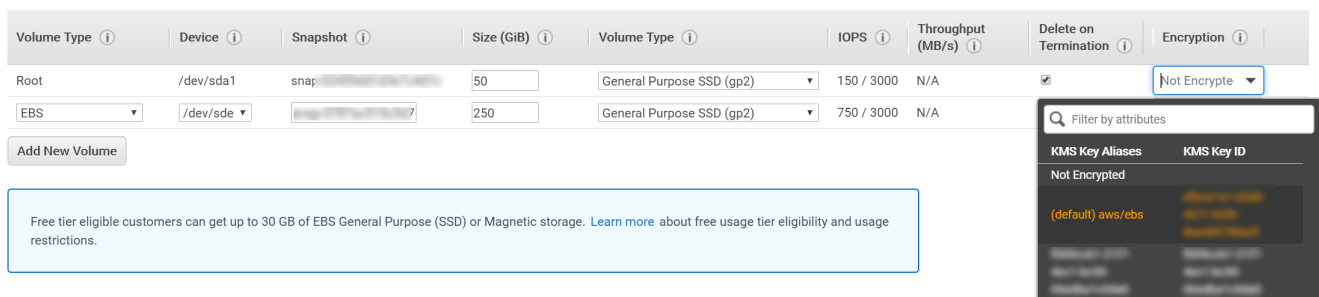
The default security group allows access to the KNIME Server via HTTP, and HTTPS on ports 80 and 443. Additionally advanced admin access via the SSH port 22 is enabled.

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.

The simplest way to enable this encryption is to choose to enable disk encryption with the default key, at the stage where you can choose the volume options.

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.

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 (AWS)

KNIME Server Small, and KNIME Server Medium are both sold via the AWS 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.

For a more general discussion of sizing KNIME Server, please see the section Sizing.

EC2 Instance selection

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

The R5a.2xlarge instance has 64 Gb RAM available, and also 8 CPU cores, thus is the largest instance that KNIME Server Small and KNIME Server Medium can make use of.

Full details of EC2 instance types can be found here.

Note on core counting

KNIME Executors will identify a core using the JVM. On AWS, by default a KNIME Executor core is equivalent to a vCPU. In some cases you may find that you get better per core workflow execution performance by choosing to optimize the number of threads per core. E.g. using an R5a.4xlarge instance, with threads per core set to 1. Making this settings change will obviously come at the expense of additional infrastructure costs (r5a.2xlarge vs. r5a.4xlarge). See the AWS documentation for full details.

EBS volume selection

The default root instance volume is 50Gb SSD (gp2), and in most cases it is not necessary to increase the volume size. The additional volume has a default size of 250Gb SSD (gp2) which should be appropriate for many new installations.

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 Workflow Repository EBS volume capacity.

KNIME Server Large

You may follow all recommendations for KNIME Server Small/Medium. However, when using a KNIME Server Large deployment with distributed KNIME Executors you may wish to use several homogeneous instance types for the KNIME Executors, in order to leverage the flexibility of scaling instance types depending on the required Executor workloads. You may also consider scaling in the KNIME Server instance, to a slightly smaller instance type (no less than 4 vcpus), since execution now happens on the KNIME Executors themselves. Since RAM is of less relevance for the KNIME Server instance, you may choose an 'm' instance rather than an 'r' instance.

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 AWS Marketplace. See AWS Marketplace Pricing Questions regarding BYOL licensing should be directed to sales@knime.com.

Hardware pricing

Hardware pricing is defined by AWS. See AWS Pricing.

All hardware/service costs calculated using the AWS Cost Calculator.

KNIME Server Small

Required services (Server Small/Medium - incl. BYOL)

- EC2
- EBS volume
- Data transfer in/out
- Optional: EBS 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 AWS deployment (KNIME Server Small/Medium/Large (via BYOL))

Simple deployment

A simple deployment as per the sizing guidelines in the previous Sizing (AWS) section looks something like:
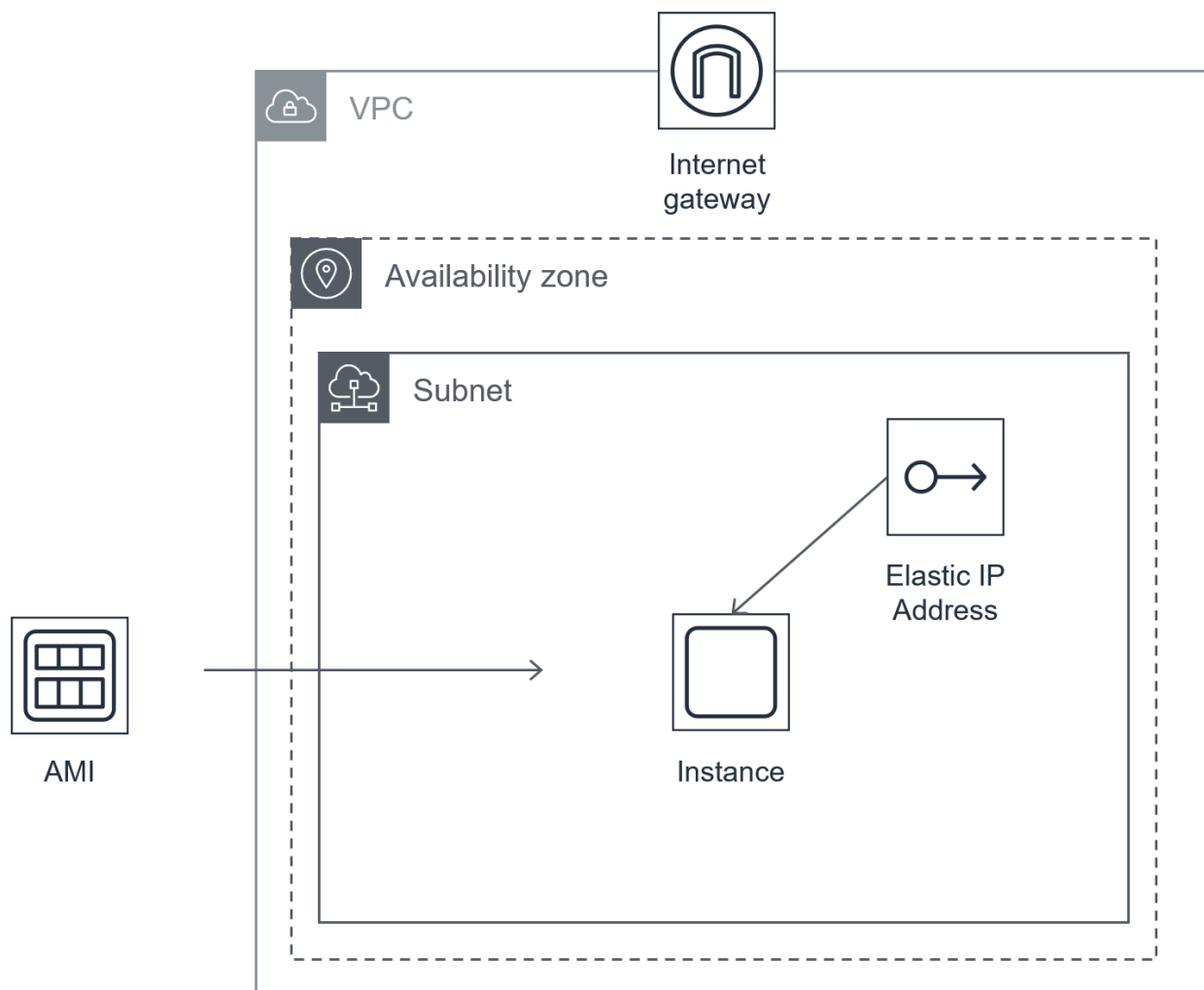

Simple Failover

A simple, non-automated deployment to ensure disaster recovery in case of e.g. an instance becoming corrupted, is to to take a snapshot of the instance (on a regular basis). Then in case that disaster recovery is required, simply switch the Elastic IP from the old instance to the new instance.
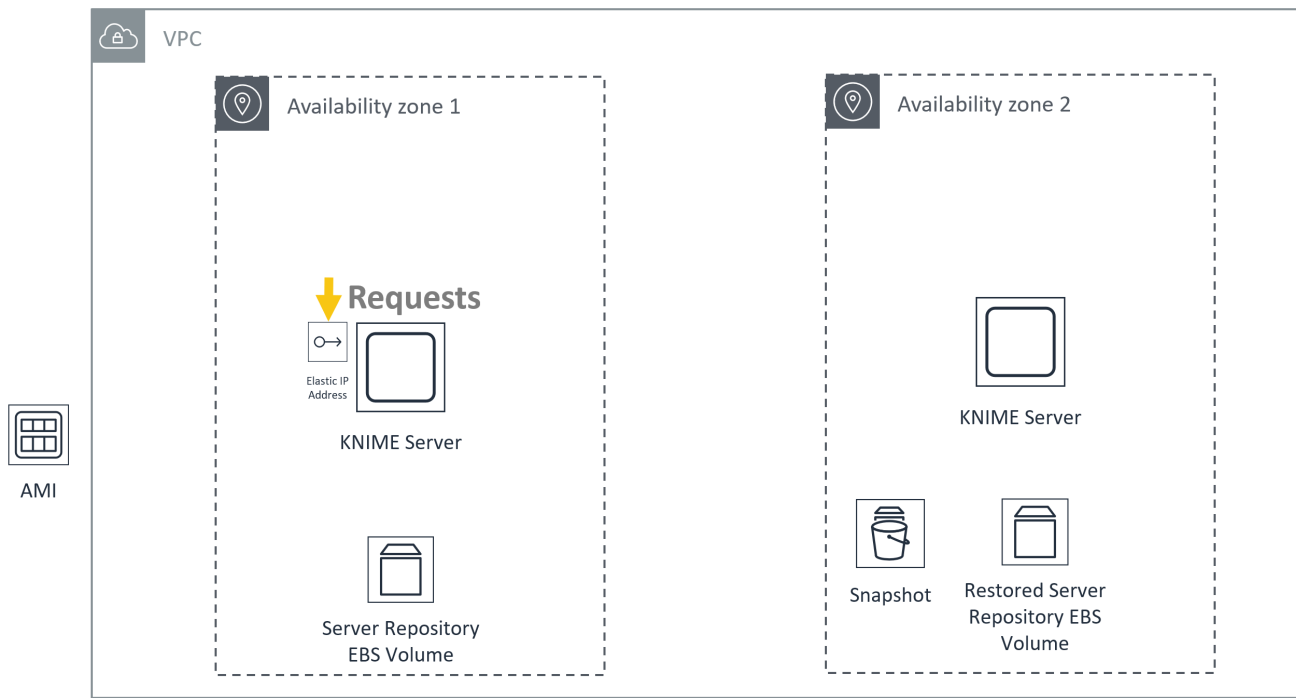
Cold-standby (EFS)

By moving the contents of the /srv directory from an EBS volume to AWS EFS, it becomes possible to setup a failover system that can ready the latest state of the KNIME Server Workflow Repository.
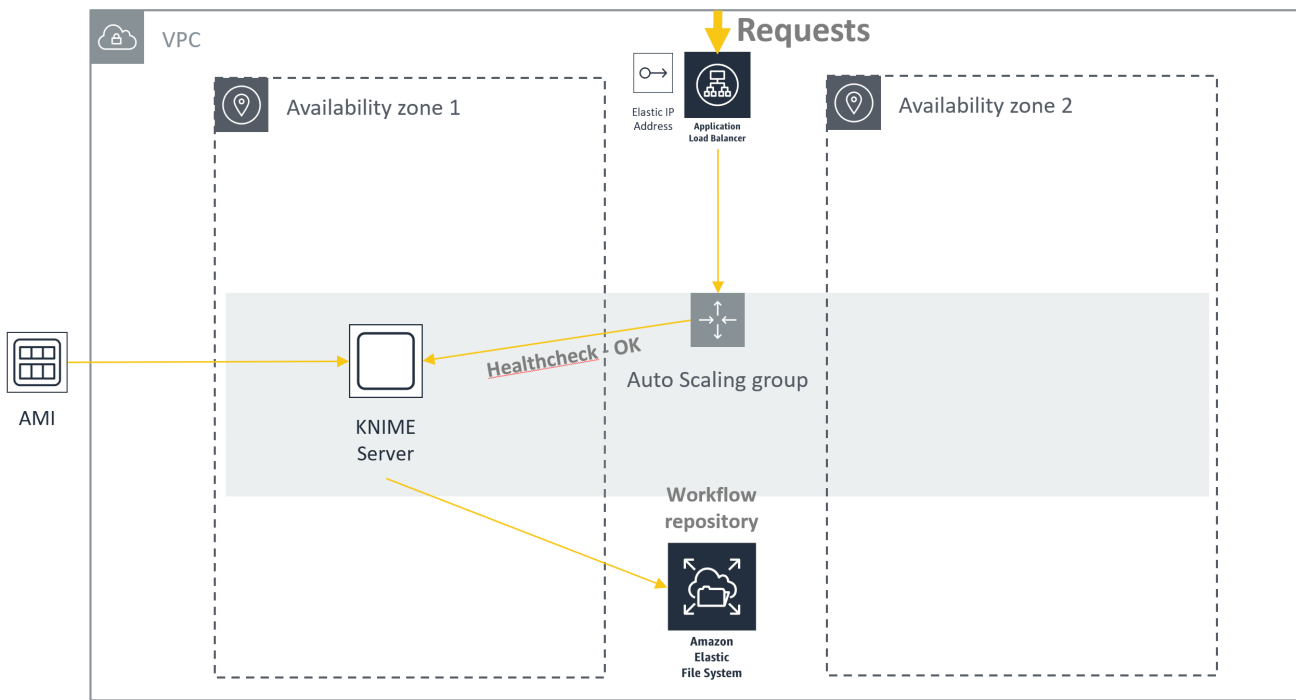
Failover will take a couple of minutes while the new instance boots and loads the repository contents.

Using this setup gives resilience to both instance failure, and availabity zone failure.

> ⚠️ While it is technically possible to connect two KNIME Server instances to the same workflow repository, doing so will result in data loss.

Cold-standby (EBS)

Deploying on AWS makes it very simple to provide cold-standby in case of an instance failure. In this case you may choose to either migrate the EBS volume of the workflow repository, or you need to take a regular snapshot of the data block devices.
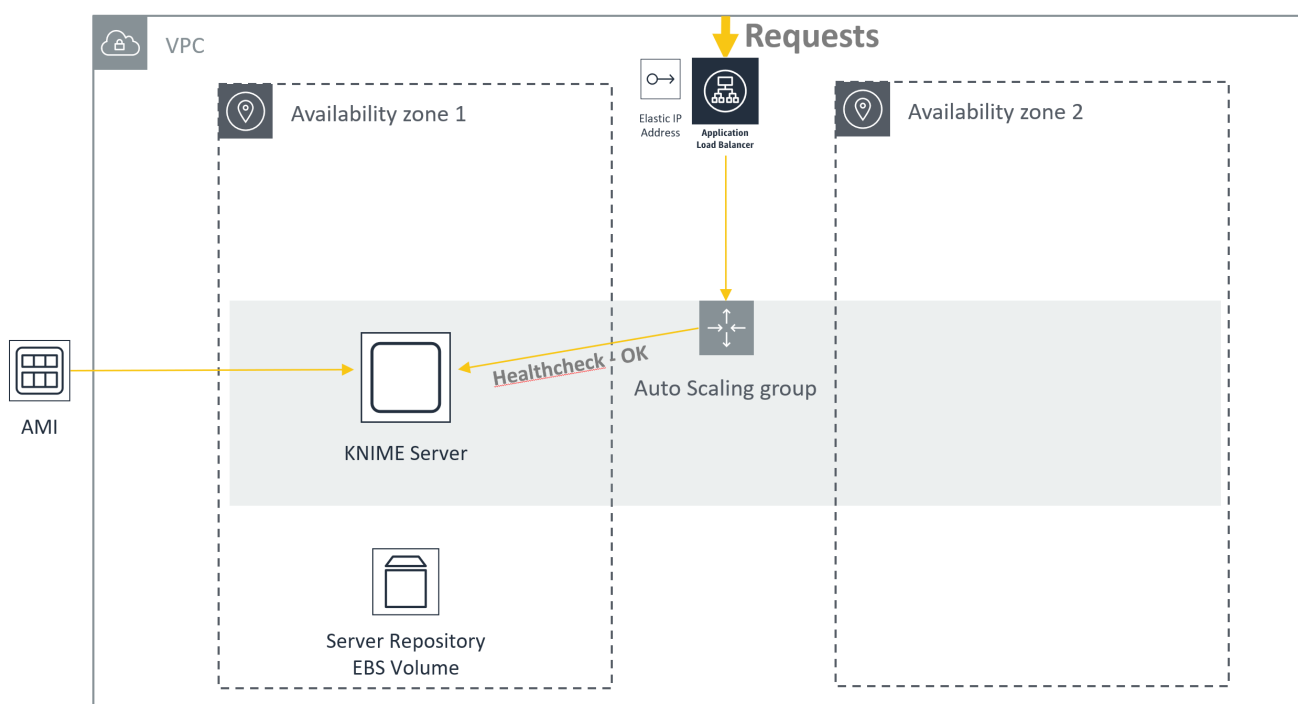

Make sure that the snapshots are available in the region that you want to launch the cold-standby server.

When implementing this setup using the AWS Marketplace images, you will need to modify the AMI so that the admin password isn't reset on instance failover. If using the BYOL image and a KNIME Server Large license the recommended way to solve the issue is to perform authentication using LDAP/Active Directory. For KNIME Server Small/Medium where LDAP is not available, you will need to disable the password set script by renaming `/opt/init_db.sh`

You will need to implement user data scripts that manage attaching/detaching the EBS volume to be migrated to the failover instance, and if you need to migrate to a new availability zone you will also need to snapshot the instance. Due to this complexity, and the additional complexity to ensure availability zone failure resilience, cold-standby using AWS EFS is preferred in most cases.

Recommended AWS deployment (KNIME Server Large)

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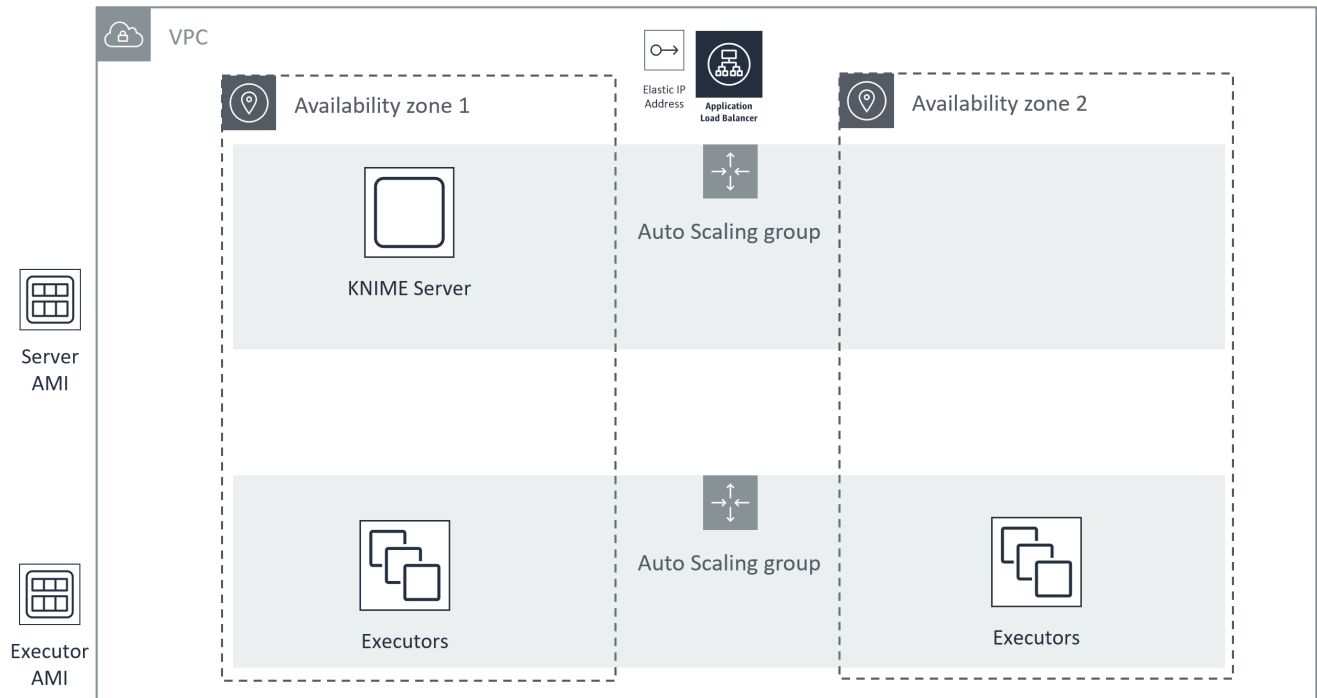
Default KNIME Server password

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 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 AWS (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: <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 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

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.

KNIME Executors with Auto Scaling Groups

Three components are needed for Auto Scaling KNIME Executors on AWS:

- A VPC that hosts KNIME Server, RabbitMQ and the KNIME Executors

- An AWS Auto Scaling Group that manages the startup and shutdown of KNIME Executors

> **i** KNIME Server should be deployed in a public subnet, while KNIME Executors should be deployed in a private subnet.

Setting up the VPC

A Virtual Private Cloud (VPC) within the AWS cloud platform provides a secure networking environment for KNIME Software. Each VPC provides its own networking domain which provides access to and protections from the outside networking world. The divisions are well defined, and layers of network security are provided.

When deploying the KNIME Server within a VPC it is usually deployed within a public subnet. This will allow connections to be made to the Server from the external internet. Security Groups can be used to control access to the KNIME Server (I.e. open ports to specific IP addresses or a range of IP addresses). There are scenarios where the KNIME Server can be deployed within a private subnet. The use cases for this deployment include: all access of the Server will be made within the VPC or the VPC is connected through a VPN to a company's private network.

KNIME Executors are generally deployed within private subnets. The reasoning here is that the outside world does not need to connect to Executors. Executors only communicate with the KNIME Server and RabbitMQ. When run within an Auto-scaling Group (ASG), the Executors should be run within subnets that are contained in different Availability Zones. This provides a level of resilience to the Executors. If networking within an AZ becomes unavailable, the ASG will automatically initiate new Executors in the still functioning AZ. While overall compute capacity may be lowered in the period while the new Executors are starting up, the service of Executors is not disrupted. And within a short period of time, the desired level of compute capacity will be regained.

The diagram below depicts the typical deployment of KNIME Server with KNIME Executors within a VPC in AWS.
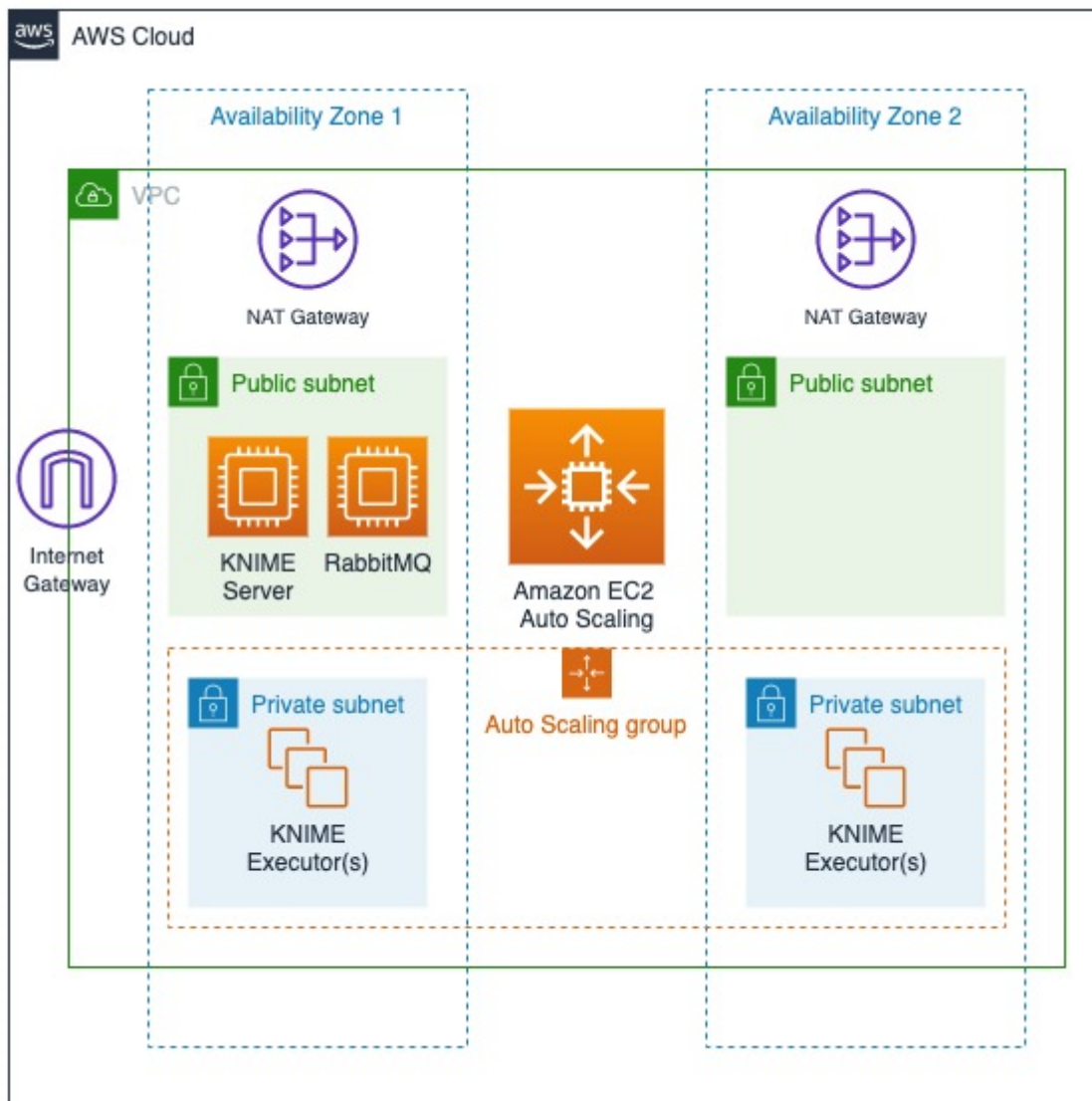
CloudFormation Templates

KNIME provides a CloudFormation template along with the AWS Marketplace AMIs to simplify the creation of the AutoScalingGroup.

executors.yaml

This stack creates the AutoScalingGroup that governs the KNIME Executors, as well as the security group for those instances; the AMIs for KNIME Executor instances are available on the AWS marketplace.

Open the yaml file in a text editor and edit the RegionMap which specifies the KNIME Executor AMI. Make sure to also set the correct AWS region.

Fill out the template as prompted in the CloudFormation console and create the stack. Select

the VPC which the server instance is in. In the Executor Configuration section you need to set the private IP that was assigned to the server instance; do not enter the public IP since the KNIME Executor instances are not allowed to communicate outside of the VPC. Set the target average CPU utilization for the scaling group.

> **i** Using a value here that is too low might cause ASG to scale up twice even though only one scale up is desired. A value of 90% should be an appropriate starting point. It is possible to tweak those settings at a later point in the AutoScalingGroups console.

In addition, you need to specify the minimum and maximum limits of how many executors should be started. In case you do not wish to scale dynamically, but rather use BYOL Executors, minimum and maximum should be set to the same value. For those scenarios, the ASG allows you to keep the correct number of executors running in case of failure.

After the stack was created the specified minimum number of KNIME Executor instances will be started, and they should automatically connect to RabbitMQ on the server.

After this step the Auto Scaling KNIME Executors are ready to use.

Terminate KNIME Executors

If you want to stop using this setup then simply delete the stack in the CloudFormation section of AWS.

> **i** This also means that all data on the server instance is lost, unless it has been backed up prior to the deletion.

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"
```

AZ fault

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

See the section Cold-standby (EFS) to see an example of an architecture resilient to Availability Zone fault.

Instance fault

An instance fault can be detected using AWS Cloudwatch to monitor the instance health. See the section Cold-standby (EFS) to see an example of an architecture resilient to instance fault.

Storage capacity

You may monitor the storage capacity of the two EBS volumes (root and data) using standard techniques and services such as AWS CloudWatch. 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 (AWS)

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

Recovery (AWS)

To restore an AWS EBS Snapshot, see the AWS documentation section on restoring EBS volumes.

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.
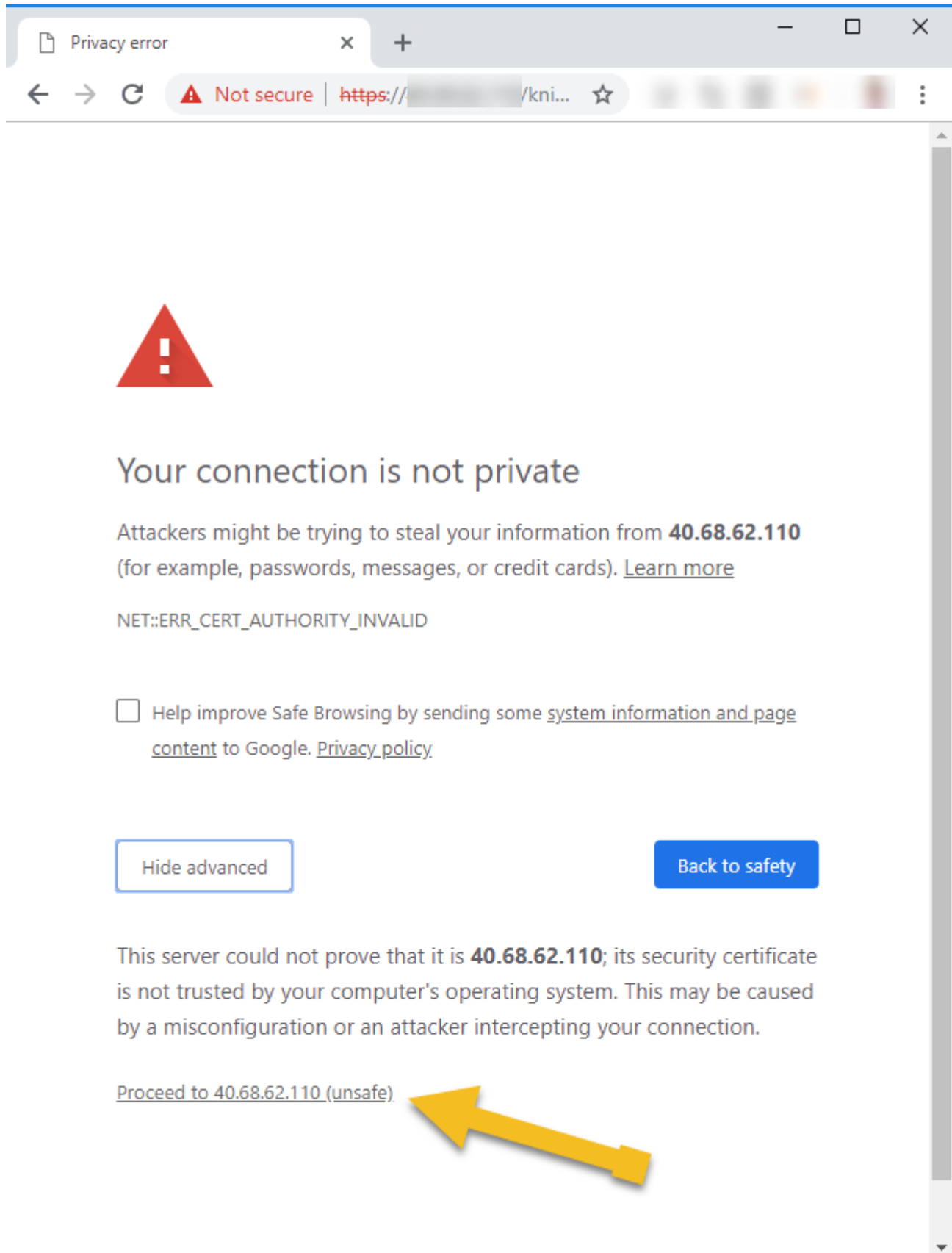
Update Python configuration

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 4.17 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.

Disk swap update (AWS)

- Login to the existing server, and stop the KNIME Server service. See section Stopping KNIME Server.

- Create a snapshot of the workflow repository block device.

  - Find the block device that needs snapshotting (in the AWS EC2 Console)
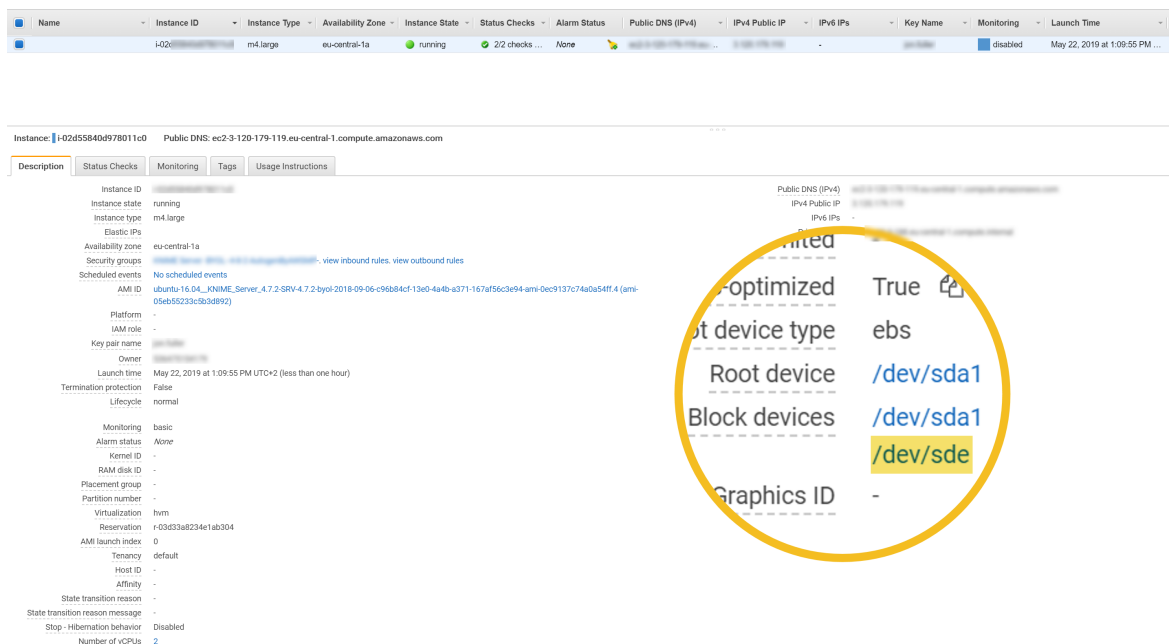
  - Create a snapshot of the block device
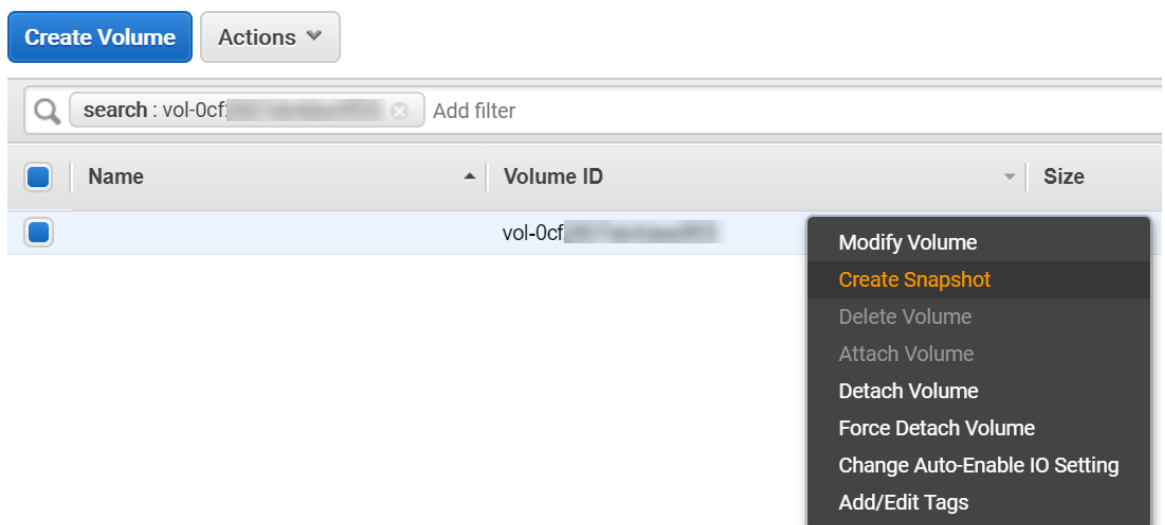
  - Give the snapshot a name

- Launch a new instance of KNIME Server from the AWS Marketplace, using the previously created snapshot.

  ◦ Launch new instance on AWS Console

  ◦ Assign snapshot to new instance

- For updates from marketplace images version 4.7.2 it is required to follow the instructions in the following section.
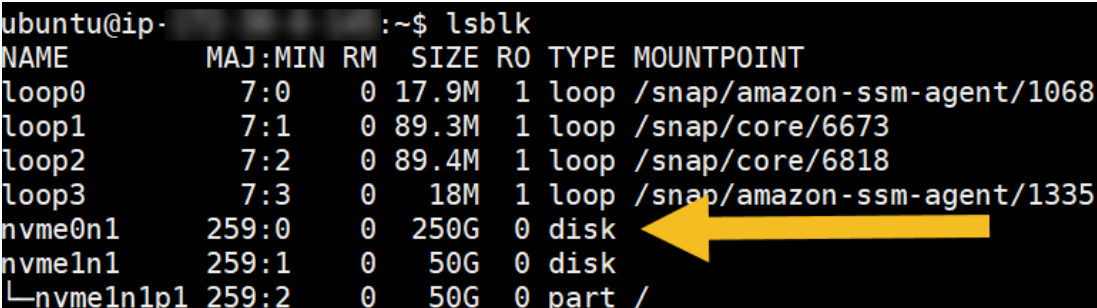
Note on upgrading to 4.8.2+

To better support newer EC2 instance types such as the m5, c5 and t3 series of instances which use NVMe block devices a couple of changes were required. That means that an additional manual step is required when updating from an older instance type.

Failure to perform this step will mean that the Tomcat management page at `https://<hostname>` will be accessible, but the KNIME Server at `https://hostname/knime` will not be accessible.

Details on NVMe devices on AWS are available here.

`lsblk`

```
ubuntu@ip-          :~$ lsblk
NAME          MAJ:MIN RM   SIZE RO TYPE MOUNTPOINT
loop0             7:0    0 17.9M  1 loop /snap/amazon-ssm-agent/1068
loop1             7:1    0 89.3M  1 loop /snap/core/6673
loop2             7:2    0 89.4M  1 loop /snap/core/6818
loop3             7:3    0   18M  1 loop /snap/amazon-ssm-agent/1335
nvme0n1         259:0    0  250G  0 disk   ◄
nvme1n1         259:1    0   50G  0 disk
└─nvme1n1p1     259:2    0   50G  0 part /
```

From the screenshot you can identify the block device that will be in the form nvmeXnY. Here it is nvme0n1 that corresponds to the 250GB device.

Temporarily mount the workflow repository in order to set the volume label. Be sure to edit `nvmeXnY` to the value from the previous step.

`sudo mount -t /dev/nvmeXnY /test`

Next apply the 'knime-repo' volume label to the block device. Using exactly 'knime-repo' will mean that you won't need to perform this step in subsequent updates.

`sudo e2label /dev/nvmeXnY knime-repo`

If you use a volume label different to 'knime-repo' then you will also need to edit /etc/fstab to use the corresponding volume label to mount the block device to /srv.

Edit `/srv/knime_server/config/knime-server.config` to change the existing line to:

`com.knime.server.executor.knime_exe=/opt/knime/knime-3.7.2/knime`

> Restart the server to apply the changes.
>
> ```
> sudo shutdown -r now
> ```

SSH access to KNIME Server on AWS

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

Connection to the KNIME Server instance is always using the `ubuntu` user, and the SSH key specified at instance launch time. An example SSH connection string is:

```
ssh -i <keyfile.pem> 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
```

Usage instructions are always included as part of the AWS console `Usage Instructions` tab.

Changing instance type

Upgrading/downgrading software edition

KNIME Server is available in three different editions, Small, Medium, and Large. The full capability listing of each version can be found here. Should you wish to upgrade to add more features, or downgrade if there are features that you don't need, you can follow the instructions below to switch to the new version.

In addition to changing the available features, it is also possible to increase the number of licensed users for the software. Currently AWS Marketplace billed images only support 5-users, although it is possible to use more users with a BYOL license. If you would like to add more users to your AWS Marketplace billed image, then please contact support@knime.com.

KNIME Server BYOL

KNIME Server on AWS (BYOL) gives the option to apply any valid license file obtained from KNIME, or KNIME partner. Since all functionality, or the number of licensed users is tied to the license file, it is possible to exchange the license file and immediately benefit from the new functionality. See the section Applying license file to KNIME Server on AWS (BYOL) for full details on how to apply the license file.

Note on distributed KNIME Executors

The current KNIME Server on AWS (BYOL) image is a single AMI that can support KNIME Server Small, Medium or Large functionality. Should you wish to use KNIME Server Large and the distributed KNIME Executors functionality, which uses a server image, and a KNIME Executor image, you would need to either use the KNIME Server Large solution template from the AWS Marketplace, or contact support@knime.com for assistance to build this deployment yourself.

KNIME Server on AWS Small/Medium/Large

Upgrading/downgrading between versions is simply a case of following the Disk swap update (AWS) instructions. In this case the newly launched image should be the desired new edition, and the snapshot disk should be made from the existing installation workflow repository block device.

Scaling the KNIME Server instance

In the case that your KNIME Server instance is running at high CPU load permanently, you may wish to scale to a larger instance size. Conversely it may be the case that you started with an instance size that is too large. In both cases, it is possible to stop the KNIME Server and start it again using a different instance type.

To do so, you should follow the instructions here.

Increasing Workflow Repository EBS volume capacity

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

Changing the size of the root volume should not be required. The procedure is more difficult than increasing the size of the workflow repository EBS volume.

Key rotation

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

Service limits

Managing AWS service limits 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 (AWS)

In case KNIME Server is not available due to degraded performance of an Availability Zone (AZ), EC2 instance fault, etc. It is possible to restore a snapshot and launch a new instance.

AZ recovery

AZ recovery is managed by launching a new instance into an unaffected AZ, 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.

Troubleshooting

To troubleshoot problems that you might have with KNIME Server instance please refer to KNIME Server Administration Guide and to Amazon EC2 Troubleshoot documentation.

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, Instance ID, and AWS Account ID. We aim to respond to your question in under 48 hours.

Finding your product details.

Finding your Product Code, Instance ID and AWS Account ID:

The AWS documentation explains how to get access to the EC2 metadata that contains the information about your instance. You can also determine this information from the EC2 web management console.

Support costs

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