Jama Connect® Kubernetes-Native Deployment

Contents

Overview

Kubernetes-Native deployment of Jama Connect delivers the application as a Helm chart into your existing Kubernetes cluster, allowing you to configure, operate, and upgrade it using the same platform tools and processes that you already know and trust.

A Kubernetes application is installed into an existing Kubernetes cluster using standard Kubernetes resources and tooling.

This means that Jama Connect:

- Is delivered as a Helm chart
- Defines all the services, containers, and configuration
- Deploys standard Kubernetes objects, including:
  - Deployments
  - Services
  - Ingress
  - StatefulSet
  - Jobs
  - ServiceAccounts
  - ConfigMaps
  - Secrets
  - PersistentVolumeClaims

Your platform team owns the cluster and core add-ons (networking, ingress controller, storage, secrets, monitoring, and RBAC). Jama Software assumes that the platform already exists.

WHAT IS HELM?

Helm is the Kubernetes package manager that enables you to ship your application as a chart. You log in to the Helm repository, configure a `values.yaml` file for your environment, and install or upgrade with simple Helm commands. All key configuration settings (networking, scaling, storage, external services, security, resource limits) live in the `values.yaml` file.

BENEFITS

You can use your existing Kubernetes platform

- Jama Connect runs on your own on-premises clusters.

- You are no longer constrained by vendor-managed Kubernetes or locked-in infrastructure.

Standard tools, no proprietary installer

- Install, upgrade, and operate with Helm and kubectl.

Security, compliance, and transparency

- All resources are standard Kubernetes objects for easy review.

Operational consistency

- Customer operations and IT teams can use familiar workflows such as scaling, monitoring health, troubleshooting, and upgrades.
- Helm-based rolling upgrades avoid downtime by letting Kubernetes handle pod restarts, traffic routing, and readiness checks.

Installation and deployment workflows

This information depicts the workflows for installing and deploying Jama Connect in a Kubernetes environment.

INSTALLATION WORKFLOW

Whether your environment is internet-enabled or airgap, the installation process consists of three stages:

- Planning your installation
- Preparing your environment
- Installing and upgrading

DEPLOYMENT PROCESS WORKFLOW

These are the steps involved in deploying Kubernetes:

1. Verify that a Kubernetes cluster and core platform services exist.
2. Configure `values.yaml` (including Jama Connect license and other configuration settings).
3. Log into the Helm repository.
4. Install application and components into a dedicated namespace.
5. Allow Kubernetes to create services, pods, and storage.

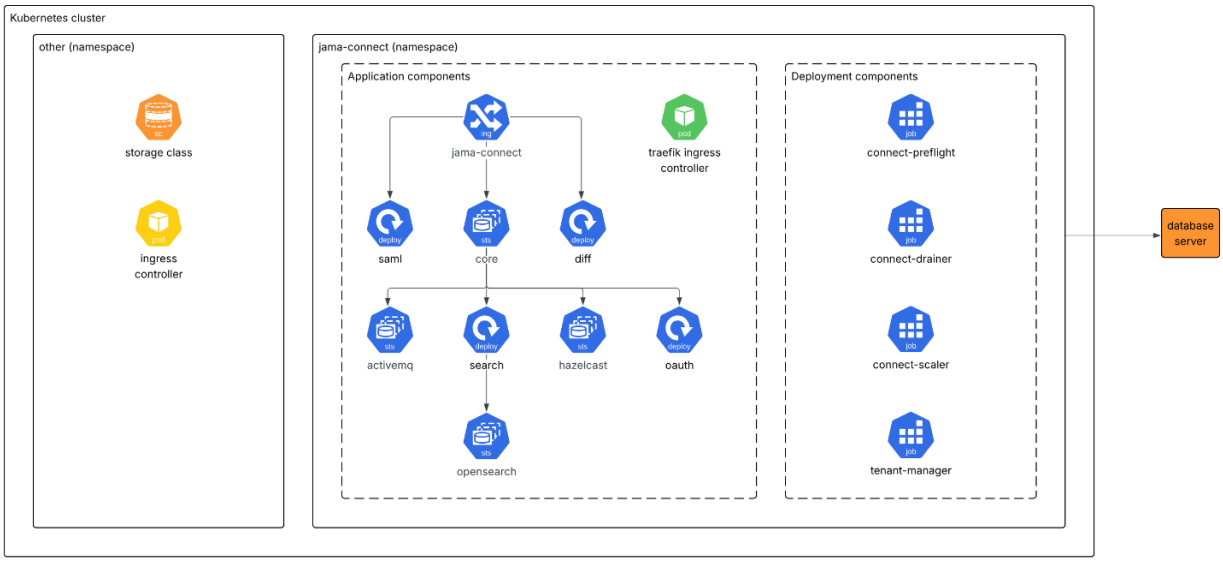
Jama Connect becomes accessible via the ingress controller and operates with standard Kubernetes tooling.

> **NOTE:** For airgap environments, the same flow is used with an offline Helm bundle and locally imported container images.

Architectural components

These diagrams show the high-level logical architecture of Jama Connect in a Kubernetes deployment.

This diagram identifies the application components, deployment components, and customer-provided infrastructure within the Kubernetes cluster, including the external database server, and components that can be optionally enabled or provided.



This color-coding legend summarizes the role of customer-provided components, deployment jobs, and core application components that run and manage Jama Connect.

Color coding

- Component provided by Jama Connect.
- Component **optionally** provided by Jama Connect. It can be disabled if customer provides its own equivalent component.
- Required component provided by the customer.
- Component **optionally** provided by the customer. It is not required, if customer enables equivalent component provided by Jama Connect.

Customer provided components

- **storage class:** allocates persistent storage for components that require it.
- **ingress controller:** implements routing rules defined in ingress component.
- **database server:** MySQL or MSSQL.

Deployment components

- **connect-preflight:** verifies database connection before deployment.
- **connect-drainer:** drains scheduled jobs before deployment.
- **connect-scaler:** scales down replicas of core statefulsets to zero before deployment.
- **tenant-manager:** requests provisioning of a tenant, tenant upgrades and scheduling of jobs after a deployment.

Planning your installation

Plan a Kubernetes Jama Connect installation by reviewing the software, infrastructure, resource sizing, database requirements, supported software, and authentication options needed to prepare your environment before installation.

IDENTIFY JAMA CONNECT VERSION

Review the Jama Connect Release Notes and decide which version to install from our Supported Versions list. For example: 9.35.2.

REQUIRED SOFTWARE FOR INSTALLATION

- Kubectl access
- Helm CLI with support for OCI repositories

APPLICATION REQUIREMENTS

Provided by Jama Software — A welcome email that includes:

- Artifactory credentials
- Jama Connect license

Provided by customer

- An existing Kubernetes cluster
- Supported Kubernetes versions — 1.34
- A DNS hostname for accessing Jama Connect
- An ingress controller OR the Traefik ingress controller included in our Helm chart (controller runs on node ports 80 and 443; the Traefik ingress controller must be enabled)
- Virtual memory setting (vm.max_map_count) set to 262144 or higher OR our Helm chart configured to set the virtual memory. For information on how to set up the virtual memory, see “Configure custom memory settings for OpenSearch (KOTS)” in the *Installing and Upgrading Jama Connect KOTS Guide*.
- Nodes with the topology.kubernetes.io/zone labels unless the default topologySpreadConstraints is changed
- A dedicated namespace
- Allowed users and capabilities required by containers (see Prepare security context constraints)
- CPU, memory, and disk — By default, our application requires:

- 11 GB for storing container images per release
- 4705 m of CPU, with limits set to 13250 m
- 16.32 GB of memory, with limits set to 32.85 GB
- A CSI driver able to provision 26.84 GB of disk space for persistent volume claims

If our HA sample values file is used, our application:

- Requests 18915 m of CPU, with limits set to 61550 m
- Requests 83.67 GB of memory, with limits set to 168.27 GB
- Requires a CSI driver able to provision 74.09 GB of disk space for persistent volume claims

- (Airgap only) A private registry to host the Jama Connect container images.

Important considerations

- To allow for upgrades, provision enough space for multiple releases.
- To fit your own expected usage, adjust the default values for persistent volume claims.

Database server requirements

The database must be hosted on a server separate from the Jama Connect application. This server can host other databases, but running other applications on the same server as the database is not supported.

Minimum

- 4–8 CPU
- 16–24 GB RAM

Recommended

- 8 CPU
- 24 GB RAM
- Dedicated volumes for data

Database software

- MySQL 8.4.x
- Microsoft SQL Server 2022

Musts

- Database is hosted on a server separate from the Jama Connect application.
- Database server can host other databases, but no other applications.
- Accessible by admin with permissions.
- Uses only supported software and environments.
- Databases must be able to accept a minimum of 300 concurrent connections.

Not supported

- Azure database
- MariaDB
- Custom configurations of Jama Connect databases (for example, query optimization and additional indexes that aren't shipped with Jama Connect)

Resource sizing for database server

For optimal performance, estimate your database server needs before you install Jama Connect.

Important considerations

- Total system RAM for your database server can vary if you use memory-intensive workflows such as reuse, exporting, moving items, integrations, and batch updates. Database sizing is based on your usage patterns and your platform. You must have a minimum of 4–8 cores and 16–24 GB of memory. Consult with your database admin when determining database size.
- The memory allocation allows for minimum headroom. If you need to run additional software for monitoring and analysis, consider the system requirements for that software.

Use the information in this table to determine the resources needed for your database server.

Database server	Small	Medium	Large	Enterprise
Active items in system	≤ 600, 000	≤ 2 million	2–4 million	4 million+
Active projects	≤ 100	≤ 500	≤ 1, 000	1,000+
Concurrent users	≤ 50	≤ 500	≤ 1, 000	1,000+
CPU	4	8	16	Contact Support

Database server	Small	Medium	Large	Enterprise
Total systems of RAM	16 GB	32 GB	64 GB	Contact Support

If your usage approaches the Enterprise threshold, contact Support for customized recommendations and advanced, multi-server setup.

Supported software

Make sure your environment uses only supported software.

BROWSERS

- Edge Chromium
- Firefox*
- Google Chrome*
- Safari*

** Versions released over the past 12 months are supported.*

IMPORTANT:

- Browser zoom is supported only at 100%. Use of browser extensions/addons or enabling Compatibility View is not supported while using Jama Connect.
- To prevent session issues, use the application in a single browser window.

WORD PROCESSOR AND SPREADSHEET PROGRAM

- Office 365 for Mac
- Office 365 for Windows (used for exports and reports)

AUTHENTICATION

- LDAP
- SAML

Preparing your environment

Prepare your environment for a Kubernetes deployment by completing the following tasks before you install Jama Connect.

- Preparing your database server
- Install and configure MySQL

- Install and configure Microsoft SQL Server
- Prepare custom memory setting for OpenSearch
- Preparing the Helm chart
- Preparing the Helm chart
- Preparing the ingress controller
- Preparing zone labels
- Create a namespace
- Prepare security context constraints
- Prepare artifacts for airgap installation
- Preparing a dataset

Preparing your database server

Use the following information to connect the application server to the database server.

Information	Requirements
Type/vendor	Database must be one of the following: • MySQL 8.4.x (recommended) — See Install and configure MySQL. • Microsoft SQL Server 2022 — See Install and configure Microsoft SQL Server.
Database hostname	Example: jama.companydb.com
Listening ports	The Kubernetes cluster must be allowed to communicate remotely with the database server over the listening ports. Default ports are: • MySQL = 3306 • Microsoft SQL Server = 1433
Database schema name	The database owner must be able to create: • A new database schema • Tables inside an existing database schema of the given name Database name must follow these rules: • Start with a letter (a–z)

Information	Requirements
	• Contain any number of characters: a–z, 0–9 or an underscore ("_") • Letters must be lowercase
Username	Jamauser
Password	
Connections	Database must be able to accept a minimum of 300 concurrent connections.
SAML database username	samluser
OAuth database username	oauthuser
Quartz database username	quartzuser

IMPORTANT: Usernames and passwords for each database schema must match what's entered in the preparation SQL scripts.

Install and configure MySQL

MySQL is the recommended database server. Follow these steps to install and configure it.

Important considerations

- You must have full database admin permissions to the server that hosts the MySQL database.
- For the Jama Connect installation to succeed, you must create three or four database schemas, as described here.

RECOMMENDED SETTINGS AND SAMPLE CONFIG FILE

The following recommended settings require 8 GB of memory allocated to MySQL Server for a typical installation and 16 GB for an enterprise installation.

When creating a database, use the default character set and collation (utf8mb4 and utf8mb4_0900_ai_ci).

These settings can be added to your my.cnf file (Linux) or your my.ini file (Windows).

Property	Typical installation	Enterprise installation
`max_allowed_packet`	1 GB	1 GB

Property	Typical installation	Enterprise installation
tmp_table_size	2 GB	2 GB
max_heap_table_size	2 GB	2 GB
table_open_cache	512	512
innodb_buffer_pool_size	2 GB	12 GB
innodb_log_file_size	256 MB	256 GB
innodb_log_buffer_size	12 MB	12 MB
innodb_thread_concurrency	16	16
max_connections	151	351
wait_timeout	259200	259200

Here is a sample text config file at an enterprise level. You must add the following values for your environment:

```
bind-address=0.0.0.0
key_buffer_size=16M
max_allowed_packet=1G
thread_stack=192K
thread_cache_size=8
tmp_table_size=2G
max_heap_table_size=2G
table_open_cache=512
innodb_buffer_pool_size=12G
innodb_log_file_size=256M
innodb_log_buffer_size=12M
innodb_thread_concurrency=16
max_connections=351
wait_timeout=259200
```

To install and configure MySQL:

1. Make sure that the InnoDB engine is enabled.
2. Download and install a supported version of MySQL.
3. On the MySQL database server, create an empty Jama Connect schema / database that uses UTF8:

   ```
   CREATE DATABASE jama character set utf8mb4;
   ```

4. On the MySQL database server, create two additional database schemas and a user ("jamauser") with the ability to access, create, and update tables within the database:

```
CREATE DATABASE saml;
CREATE DATABASE oauth;
CREATE USER 'oauthuser'@'%' IDENTIFIED BY 'password';
CREATE USER 'samluser'@'%' IDENTIFIED BY 'password';
GRANT ALL PRIVILEGES ON jama.* TO 'jamauser'@'%';
GRANT ALL PRIVILEGES ON oauth.* TO 'oauthuser'@'%';
GRANT ALL PRIVILEGES ON saml.* TO 'samluser'@'%';
```

5. Create a database schema for Quartz to support horizontal scaling of core pods:

```
CREATE DATABASE quartz;
CREATE USER 'quartzuser'@'%' IDENTIFIED BY 'password';
GRANT ALL PRIVILEGES ON quartz.* TO 'quartzuser'@'%'
```

6. Restart the database server.

MySQL is now installed and configured on your database server.

Install and configure Microsoft SQL Server

If you are using Microsoft SQL Server for your database, follow these steps to install and configure it.

Important considerations

- You must have full database admin permissions to the server that hosts the SQL Server database.
- Install Microsoft SQL Server 2022 for the database server.
- For the Jama Connect installation to succeed, you must create an empty Jama Connect database and two or three additional database schemas (mentioned below). Otherwise, the system continues to attempt to connect to the databases and produces log failures. After you create the database schemas, you must restart Jama Connect.
- Jama Connect requires that the MSSQL COMPATIBILITY_LEVEL value is 160 or greater.
- The SQL query script in this task is run only for a fresh install and only before installing Jama Connect. Don’t run the script on an existing installation.
- When you upgrade Microsoft SQL Server (for example, from Microsoft SQL Server 2017 to 2019 or 2022), existing databases keep their previous compatibility level by default. After the database instance upgrade, the compatibility level must be manually configured.

  To confirm the current value:

  ```
  SELECT compatibility_level FROM sys.databases WHERE name = <DATABASENAM
  E>;
  ```

  To modify the value:

```
ALTER DATABASE SET COMPATIBILITY_LEVEL = 160; -- SQL Server 2022
```

For more information on compatibility level, see Alter Database (Transact-SQL) compatibility level.

To install and configure Microsoft SQL Server:

1. Connect to the SQL Server using a SQL management application (such as SQL Server Management Studio).
2. Replace the database passwords in the installation script below.
3. Copy and store the passwords you create here. You will need them later to configure the application.
4. In a new query window, run this SQL query script.

   IMPORTANT: This script must be run prior to Jama Connect installation, or the installation might fail. Modify the passwords in this script before execution. Passwords must be enclosed in single quotes.

```
/*
This script must be run prior to Jama Connect installation or installat
ion may fail to complete.
Modify the passwords in this script before execution. Passwords must be
enclosed in single quotes.
*/
USE master;
CREATE LOGIN jamauser with password = 'password';
CREATE LOGIN samluser with password = 'password';
CREATE LOGIN oauthuser with password = 'password';
GO
USE master;
CREATE DATABASE jama;
GO
ALTER DATABASE jama SET READ_COMMITTED_SNAPSHOT ON WITH ROLLBACK IMMEDI
ATE
GO
ALTER DATABASE jama CONVERT TO CHARACTER SET latin1 [COLLATE =
'latin1_general_CI_AI'];
GO
USE jama;
EXEC ('CREATE SCHEMA oauth');
EXEC ('CREATE SCHEMA saml');
GO
USE jama;
CREATE USER jamauser for LOGIN jamauser;
CREATE USER samluser for LOGIN samluser with DEFAULT_SCHEMA=saml;
CREATE USER oauthuser for LOGIN oauthuser with DEFAULT_SCHEMA=oauth;
GO
```

```
EXEC sp_addrolemember N'db_owner', jamauser;
EXEC sp_addrolemember N'db_owner', samluser;
EXEC sp_addrolemember N'db_owner', oauthuser;
```

The script creates a new empty Jama Connect database, adds two empty schemas to the database, and adds two database logins and database users to support the multi-auth functionality in Jama Connect.

5. Create a database schema for Quartz to support horizontal scaling of core pods:

```
USE master;
CREATE LOGIN quartzuser with password = 'password';
GO
USE jama;
EXEC ('CREATE SCHEMA quartz');
GO
USE jama;
CREATE USER quartzuser for LOGIN quartzuser with DEFAULT_SCHEMA=quartz;
GO
EXEC sp_addrolemember N'db_owner', quartzuser;
GO
```

6. Confirm that these actions were successful:

   - **Script completed** — Check the Query Execution results for errors.
   - **Users created** — Run the following SQL script in a new query window. The results include jamauser, samluser, and oauthuser in the "Name" column of the result panes.

     ```
     USE jama
     SELECT * from master.sys.sql_logins
     SELECT * from Jama.sys.sysusers
     ```

   - **Users granted the DB_owner role** — Run the following SQL script in a new query window. The results show that the db_owner role is granted to jamauser, samluser, and oauthuser.

     ```
     USE jama
     SELECT DP1.name AS DatabaseRoleName,
     isnull (DP2.name, 'No members') AS DatabaseUserName
     FROM sys.database_role_members AS DRM
     RIGHT OUTER JOIN sys.database_principals AS DP1
     ON DRM.role_principal_id = DP1.principal_id
     LEFT OUTER JOIN sys.database_principals AS DP2
     ON DRM.member_principal_id = DP2.principal_id
     WHERE DP1.type = 'R'
     ORDER BY DP1.name;
     ```

7. Keep the database from locking users' accounts while they are logging in or working in Jama Connect (you must have db_owner permissions):

```
ALTER DATABASE jama SET READ_COMMITTED_SNAPSHOT ON WITH ROLLBACK IMMEDI
ATE;
```

8. Make sure the flag was successfully enabled:

   ```
   SELECT is_read_committed_snapshot_on FROM sys.databases WHERE name='jam
   a';
   ```

   If the returned value is 1, the flag is on.

Microsoft SQL Server is now installed and configured on your database server.

Prepare custom memory setting for OpenSearch

OpenSearch uses an `mmapfs` directory by default to store its indices. The default operating system limits on mmap counts are often too low, which can result in out-of-memory exceptions. OpenSearch requires `vm.max_map_count ≥ 262144` on each node that runs OpenSearch pods.

> **IMPORTANT:** Execute these steps on each node where OpenSearch is expected to run, unless the Helm chart is already configured to automatically set the virtual memory.

To prepare a custom memory setting for OpenSearch:

- As an admin, run the following script:

  ```
  echo "vm.max_map_count=262144" | sudo tee -a /etc/sysctl.conf

  sudo sysctl -p

  sudo sysctl -a | grep max_map_count
  ```

The system responds with: `vm.max_map_count=262144`.

Preparing the Helm chart

Jama Connect is configured through a standard Helm values file.

> **BEST PRACTICE:** Use the example Helm values in this GitHub repository.

For lists of the required and optional values for the Helm values file, see Appendix A: Configuring the Helm chart.

Preparing the ingress controller

If you are providing your own ingress controller and/or load balancer, consider adjusting the following settings on them:

- **Maximum request body size** — Set this value high enough to support the largest files you expect to upload.

- **Idle timeout** — We recommend that you set this to 30 minutes to support long-running Jama Connect operations.

Helm values related to ingress can be found in Configure ingress in Appendix A: Configuring the Helm chart.

If you use our bundled ingress controller, it is automatically configured with our recommended settings. See Traefik ingress controller in Appendix A: Configuring the Helm chart.

Preparing zone labels

You must create a topology.kubernetes.io/zone label for your Kubernetes nodes.

If your deployment requires a different scheduling strategy, you can override the default `topologySpreadConstraints` with Helm values. See Preparing a dataset.

EXAMPLE

This example assumes that you have three nodes, and that each node is in a different zone.

Run these commands to add the label, replacing `<node1-name>`, `<node2-name>`, `<node3-name>`, `<zone1>`, `<zone2>` and `<zone3>` accordingly.

```
kubectl label nodes <node1-name-> topology.kubernetes.io/zone=<zone1>
kubectl label nodes <node2-name> topology.kubernetes.io/zone=<zone2>
kubectl label nodes <node3-name> topology.kubernetes.io/zone=<zone3>
```

NOTE: If running OpenShift, run these commands but replace `kubectl` with `oc`.

Create a namespace

For safer deployments and separation of resources, install each Jama Connect instance in its own namespace.

To create a namespace:

1. Run the following command to create a namespace called `jama-connect`:

   ```
   kubectl create namespace jama-connect
   ```

2. If running OpenShift, run this command:

   ```
   oc new-project jama-connect
   ```

Commands in the next tasks assume that this namespace already exists.

Prepare security context constraints

IMPORTANT: This task is only required if your cluster runs OpenShift.

Jama Connect uses the following users to run containers and access the persistent volumes: `91`, `481`, `1000`, and `1001`. In addition, the Hazelcast component requires the `NET_RAW` capability to start. Before you install Jama Connect, verify that your cluster allows these users and this capability.

Example

In OpenShift, a `SecurityContextConstraints` resource like the following can be saved to a file called `scc-jama-connect.yml`.

```
apiVersion: security.openshift.io/v1
kind: SecurityContextConstraints
metadata:
  name: jama-connect
allowedCapabilities:
  - NET_RAW # Hazelcast
requiredDropCapabilities:
  - ALL
allowPrivilegeEscalation: false
runAsUser:
  type: RunAsAny
seLinuxContext:
  type: MustRunAs
fsGroup:
  type: MustRunAs
  ranges:
    - min: 91 # tomcat
      max: 91
    - min: 481 # opensearch
      max: 481
    - min: 1000 # saml
      max: 1001 # others
supplementalGroups:
  type: RunAsAny
volumes:
  - configMap
  - csi
  - downwardAPI
  - emptyDir
  - ephemeral
  - persistentVolumeClaim
```

```
    - projected
    - secret
```

To create security context constraints:

1. Create the resource:

   ```
   oc apply -f scc-jama-connect.yml
   ```

2. Link the `SecurityContextConstraints` resource with the service accounts from the `jama-connect` namespace:

   ```
   oc adm policy add-scc-to-group jama-connect system:serviceaccounts:jama
   -connect
   ```

3. Add the following to your `values.yaml` file:

   ```
   core:
     hazelcast:
       application:
         env:
           HZ_NETWORK_FAILUREDETECTOR_ICMP_ENABLED: 'false'
       securityContext:
         capabilities:
           add:
             - NET_RAW
   ```

The security context restraints are configured so that your cluster allows the users and capability to run containers and access the persistent volumes.

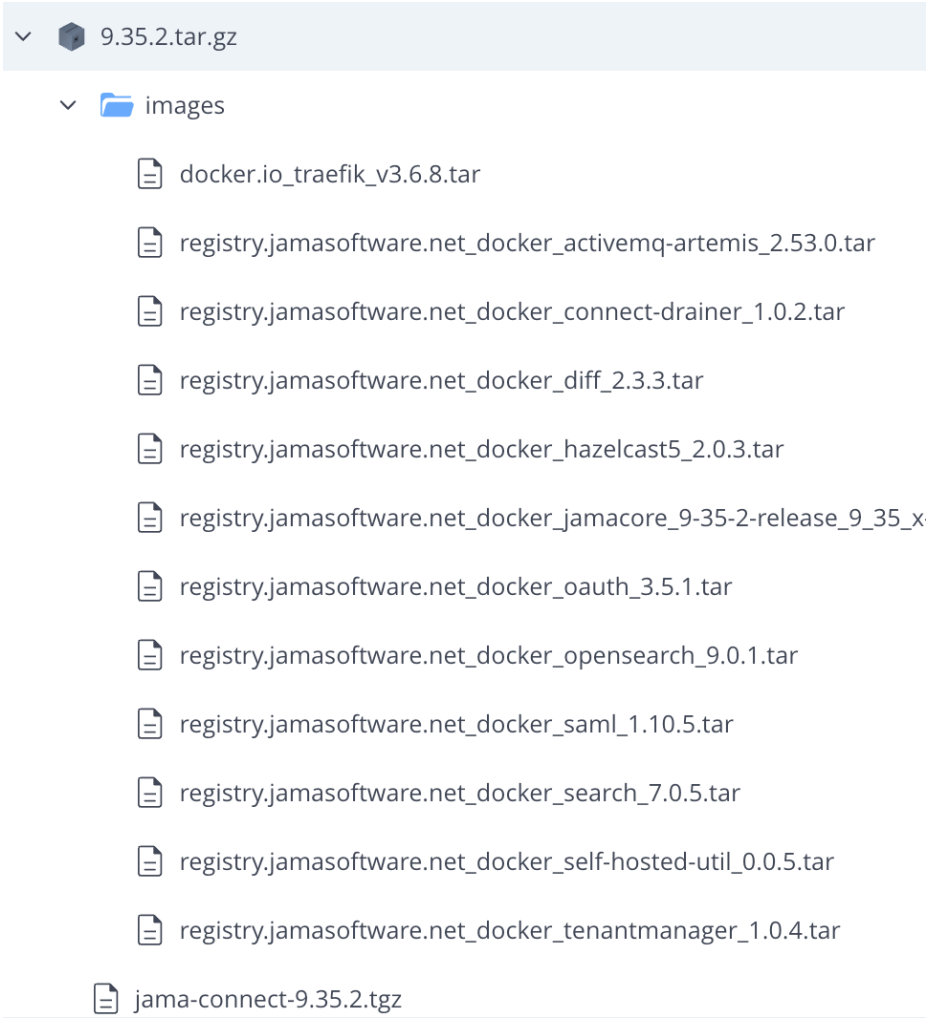
Prepare artifacts for airgap installation

Airgap installations must download the airgap bundle for the chosen Jama Connect version and make its content available to the cluster.

A *bundle* is a compressed tar file that contains:

- A `jama-connect-<version>.tgz` file. This is a Helm chart.
- An `images` folder with all the container images in tar format required by the Helm chart.

For example:



To prepare artifacts:

1. Download the bundle using one of these methods:

   From Artifactory:

   a. Go to https://registry.jamasoftware.net.

   b. Log in using the credentials provided by Jama Software.

   c. On the left side menu, navigate to **Artifactory > Artifacts**.

   d. Expand the airgap repository.

   e. Inside the jama-connect folder, navigate to and select the compressed file corresponding to the chosen version.

   f. Select **Download**.

Using curl:

- In a terminal with internet access, run the following command, replacing `<username>`, `<password>` and `<version>`:

  ```
  curl -u <username>:<password> -L -O "https://registry.jamasoftware.net/
  artifactory/airgap/jama-connect/<version>.tar.gz"
  ```

2. Extract the bundle with the following command, replacing `<version>`:

   ```
   tar -xzf <version>.tar.gz
   ```

3. Move the Helm chart from the bundle to a location accessible by Helm CLI.
4. Push all images from the bundle to a registry reachable by the cluster where Jama Connect will be installed.

Go to this GitHub repository for an example script using Docker.

Preparing a dataset

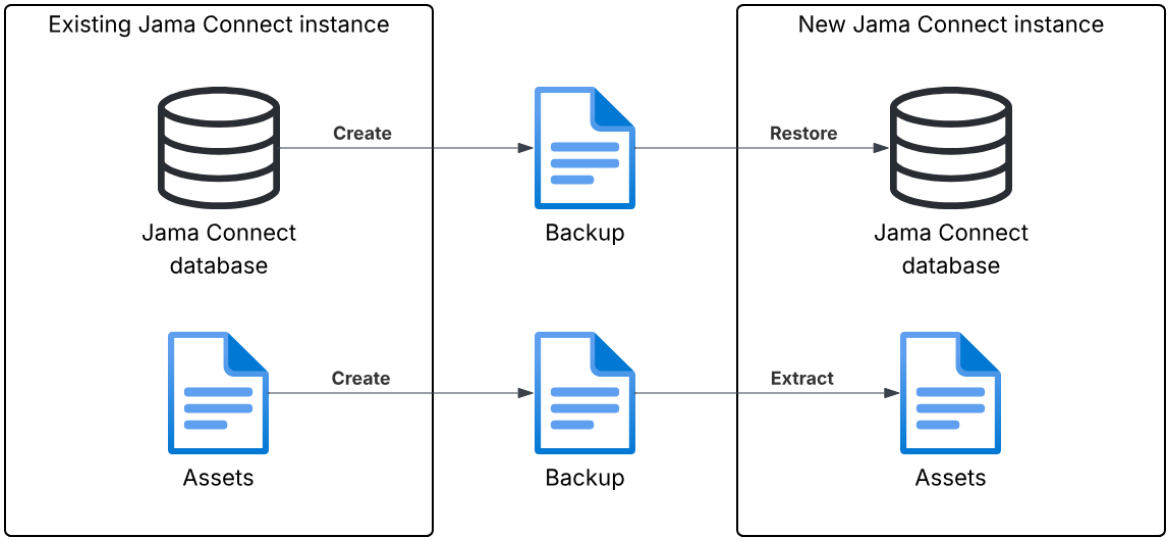
This information is only applicable if you are provisioning a specific dataset. By default, Jama Connect automatically provisions a basic dataset.

> **BEST PRACTICE:** Use a dataset created from an instance running Jama Connect 9.35.2 or later.

There are two primary approaches to provisioning a dataset:

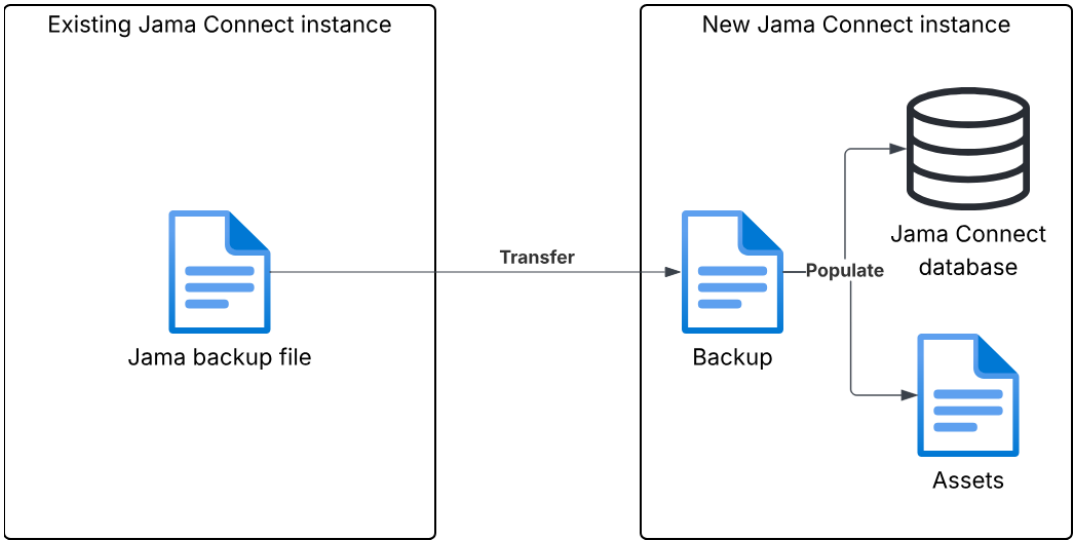
- **Use a pre-populated database** (Recommended) — Connect a new Jama Connect instance to a pre-populated database that was restored from a database backup.

  For large datasets, using a pre-populated database is generally faster and more reliable. For that reason, we recommend the pre-populated database approach for most environments.



- **Use a .jama backup file** — Restore from a `.jama` backup file.

  While `.jama` backups are a valid option, they typically require additional storage for file staging and extraction, and they often take longer to complete end-to-end.



Use a pre-populated database to configure a dataset

This recommended method for preparing a dataset involves connecting a new Jama Connect instance to a pre-populated database that was restored from a database backup.

Use the official backup tools provided by your database vendor to complete this task.

> **IMPORTANT:** Once the new instance is up and running, you must perform additional steps to finish the instance setup. See Complete your installation for dataset with pre-populated database.

The following example includes parameters that must be added to your Helm values:

```
core:
  application:
    storage:
      sharedVolume:
        request: 50Gi
  search:
    opensearch:
      application:
        storage:
          request: 30Gi
```

To prepare a dataset:

1. Create a backup of the database used by your current Jama Connect instance.
2. Create a new database instance from the database backup.
3. From the existing Jama Connect instance:
   a. Set an environment variable with your FAQ, which is used by subsequent commands.

      ```
      export TENANT_NAME=jama
      ```

   b. Save the tenant assets. If you want to reduce the backup size, exclude the `tempreports` directory.

      ```
      kubectl get pods -l app.kubernetes.io/name=core -o name | head -n 1
      | \
      xargs -I{} sh -c 'kubectl exec -c core {} -- tar -zcvf - -C /home/co
      ntour/tenant/${TENANT_NAME} \
      avatars attachments diagrams reports equations tempreports > ./asset
      s.tar.gz'
      ```

      The `assets.tar.gz` file is generated.

   c. Determine the amount of storage used by OpenSearch or Elasticsearch. If you run multiple OpenSearch/Elasticsearch pods, use the largest amount among them as your baseline.

      - **For OpenSearch pods:**

        ```
        kubectl get pods -l app.kubernetes.io/name=opensearch -o name | xarg
        s -I{} sh -c 'echo "=== {} ==="; \
        kubectl exec -c opensearch {} -- du -sh --apparent-size /usr/share/o
        pensearch/data'
        ```

      - **For Elasticsearch pods:**

        ```
        kubectl get pods -l app.kubernetes.io/name=elasticsearch -o name | x
        args -I{} sh -c 'echo "=== {} ==="; \
        kubectl exec -c elasticsearch {} -- du -sh --apparent-size /usr/shar
        e/opensearch/data'
        ```

4. In the Helm values of the new instance:
   a. Set the following to the size for OpenSearch and additional buffer for future growth:

      ```
      core.search.opensearch.application.storage.request
      ```

b. Set `core.application.storage.sharedVolume.request` to the uncompressed size of the tenant assets, plus additional buffer for future growth.

c. Set the configuration for your new database. See Configure database in Preparing a dataset.

Continue with the installation steps described in Installing and upgrading and Post-installation setup.

Use a .jama backup file to configure a dataset

Restoring from a `.jama` backup file is supported only during initial installation, before the database has been populated with any other dataset.

> **IMPORTANT:** Once the new instance is up and running, you must perform additional steps to finish the instance setup. See Complete your installation for dataset with a .jama backup file.

The following example includes parameters that must be added to your Helm values:

```
global:
  core:
    enableHorizontalScaling: false
core:
  replicaCounts:
    jobs: 1
  nodeSelector: {}
  application:
    storage:
      core:
        path: '/data/restore'
      sharedVolume:
        request: 50Gi
  tenant-manager:
    application:
      env:
        RESTORE_BACKUP_FILE: '/data/restore/mybackup.jama'
  search:
    opensearch:
      application:
        storage:
          request: 30Gi
```

To prepare a dataset:

1. Make sure the database for the new Jama Connect instance has at least the same size as the existing database.
2. From the existing Jama Connect instance:
   a. Generate a `.jama` backup. See Appendix B: Back up to .jama or XML file for details.
   b. Retrieve the backup file. See FAQ for details.
   c. Determine the amount of storage used by OpenSearch. If you run multiple OpenSearch/Elasticsearch pods, use the largest amount among them as your baseline. You can use the following commands as a reference to estimate the size.
      - **For OpenSearch pods:**

        ```
        kubectl get pods -l app.kubernetes.io/name=opensearch -o name | xarg
        s -I{} sh -c 'echo "=== {} ==="; \

        kubectl exec -c opensearch {} -- du -sh --apparent-size /usr/share/o
        pensearch/data'
        ```

      - **If you have Elasticsearch pods:**

        ```
        kubectl get pods -l app.kubernetes.io/name=elasticsearch -o name | x
        args -I{} sh -c 'echo "=== {} ==="; \

        kubectl exec -c elasticsearch {} -- du -sh --apparent-size /usr/shar
        e/opensearch/data'
        ```

3. Copy the backup file to the nodes where the new Jama Connect instance will be installed.

   By default, the restore process looks for backup files in `/data/restore`, which is mounted into the FAQ using a `hostPath` volume. You can customize this location in your Helm values. If needed, set `core.application.storage.core.path` to the directory you want to use.

   To move the backup file to a single node:

   a. Disable horizontal scaling for core pods in your Helm values so only one core pod runs:

      ```
      global.core.enableHorizontalScaling: false.
      ```

   b. Set `core.nodeSelector` in your Helm values so the core pod is scheduled on a specific node.

4. If setting `global.core.enableHorizontalScaling: true`, set `core.replicaCounts.jobs: 1` because only one core-jobs pod is supported when restoring a `.jama` backup file.
5. Make sure the backup file is readable by all users. For example:

   ```
   chmod 644 /data/restore/mybackup.jama
   ```

6. Determine the amount of storage required to restore your `.jama` backup file. Use the formula `2C + U + B`, where

   `C` = size of the `.jama` file (compressed).

   `U` = total uncompressed size of the `.jama` contents.

   `B` = additional buffer for safety.

   For example, if your `.jama` file is 4Gi compressed, 30Gi uncompressed and uses 4Gi of buffer, the result is `2 × 4 + 30 + 4 = 42Gi`.

7. In the Helm values that the new instance will use:

   a. Set `core.tenant-manager.application.env.RESTORE_BACKUP_FILE` to the backup file's absolute path.

   b. Set `core.application.storage.sharedVolume.request` to the size calculated to restore the backup file.

   c. If installing Jama Connect 9.35.x or later, set `core.search.opensearch.application.storage.request` to the size identified for OpenSearch, plus additional buffer for future growth.

Continue with the installation steps described in Installing and upgrading and Post-installation setup.

Installing and upgrading

Complete these tasks to install and upgrade Jama Connect in an internet or airgap environment.

- Install Jama Connect in an existing Kubernetes cluster (internet)
- Install Jama Connect in an existing Kubernetes cluster (airgap)
- Post-installation setup
- Upgrade Jama Connect (internet)
- Upgrade Jama Connect (airgap)
- Troubleshooting installation and upgrades

Install Jama Connect in an existing Kubernetes cluster (internet)

Use this workflow to install Jama Connect in an existing Kubernetes environment, including the required configuration steps and access to the Docker images needed for deployment.

Our Docker registry URL is https://registry.jamasoftware.net/docker.

To install Jama Connect in an internet environment:

1. Store the Artifactory credentials in a Kubernetes-Native secret by running the following command. Replace `<username>` and `<password>` with your credentials.

   ```
   kubectl create secret docker-registry jama-registry-creds \
     --docker-server='https://registry.jamasoftware.net/docker' \
     --docker-username='<username>' \
     --docker-password='<password>' \
     -n jama-connect
   ```

   If running OpenShift, use `oc` instead of `kubectl`.

2. Create a `values.yaml` file (see Preparing the Helm chart).

3. Install the Helm chart:

   a. Run the following command to log into our Helm registry.

      ```
      helm registry login oci://registry.jamasoftware.net/helm/jama-connect
      ```

   b. At the prompt, enter your Artifactory credentials for username and password.

   c. Replace `<version>` with the Jama Connect version chosen and run the following command to install it:

      ```
      helm install jama-connect oci://registry.jamasoftware.net/helm/jama-
      connect -f values.yaml -n jama-connect --version <version>
      ```

   d. During the installation, a random password is generated for the Jama Connect root user. Once the installation finishes, a command to retrieve the password will be displayed. If required, the command can be displayed again by running:

      ```
      helm get notes jama-connect -n jama-connect
      ```

      If you used a

      Use a pre-populated database to configure a dataset, the password is the same as in the source Jama Connect instance.

   e. Monitor the pods until they are running:

      ```
      kubectl get pods -n jama-connect
      ```

   f. Validate application access through the configured Jama Connect URL. While provisioning is in progress, the application might display a message that the tenant does not exist or is closed.

g. Continue with the “after installation” instructions if you If you Preparing a dataset.

If you prepared a dataset, continue with one of these tasks:

- Complete your installation for dataset with pre-populated database
- Complete your installation for dataset with a .jama backup file

Install Jama Connect in an existing Kubernetes cluster (airgap)

Use this workflow to install Jama Connect in an existing Kubernetes cluster in an airgap environment.

To install Jama Connect in an airgap environment:

1. Store registry credentials in a Kubernetes-Native secret with the following command, replacing `<registry>`, `<username>`, and `<password>` with the URL and credentials required to access the registry:

   ```
   kubectl create secret docker-registry jama-registry-creds \
     --docker-server='<registry>' \
     --docker-username='<username>' \
     --docker-password='<password>' \
     -n jama-connect
   ```

   If running OpenShift, use `oc` instead of `kubectl`.
2. Create a `values.yaml` file (see Preparing the Helm chart).
3. Install the Helm chart:
   a. Run the following command after replacing `<version>` to install the local Helm chart `tgz` file.

      ```
      helm install jama-connect jama-connect-<version>.tgz -f values.yaml
      -n jama-connect
      ```

      During the installation, a random password is generated for the Jama Connect root user. Once the installation finishes, a command to retrieve the password is displayed. If required, the command can be displayed again by running:

      ```
      helm get notes jama-connect -n jama-connect
      ```

      If you used a pre-populated database to configure a dataset, the password is the same as in the source Jama Connect instance.

   b. Monitor the pods until they are running:

      ```
      kubectl get pods -n jama-connect
      ```

c. Validate application access through the configured Jama Connect URL. During the provisioning process, the application might display a message that the tenant does not exist or is closed.

If you prepared a dataset, continue with one of these tasks:

- Complete your installation for dataset with pre-populated database
- Complete your installation for dataset with a .jama backup file

Post-installation setup

Once the new instance is up and running, you must finish setting up your instance with the dataset.

Follow the instructions for the type of dataset you configured:

- Complete your installation for dataset with pre-populated database
- Complete your installation for dataset with a .jama backup file

Complete your installation for dataset with pre-populated database

Once the new instance is up and running, perform the following actions to finish setting up your instance with the dataset.

To complete your instance setup with the dataset:

1. Set an environment variable with your FAQ, which is used by subsequent commands.

   ```
   export TENANT_NAME=jama
   ```

2. Restore the tenant assets from the `assets.tar.gz` that was generated previously:

   ```
   kubectl get pods -n jama-connect -l app.kubernetes.io/name=core -o json
   path='{.items[0].metadata.name}' | \

   xargs -r -I{} sh -c 'kubectl exec -n jama-connect -i -c core {} -- tar
   -xzvf - -C /home/contour/tenant/${TENANT_NAME} \

   < ./assets.tar.gz'
   ```

3. Update the application's base URL to match the information assigned to the instance via the `global.url` Helm value:

   a. Log in as root.

   b. Select the **Organizations** tab.

   c. Select **Change URL**.

   d. Monitor the logs until you see `Completed url update`.

```
kubectl logs -n jama-connect -l app.kubernetes.io/name=core -c core
--prefix=true -f
```

4. If you want to use SAML on the new instance with a different configuration than the previous instance, reconfigure SAML authentication.
5. Perform a full reindex:
   a. Log in as root.
   b. Select the **Index/Search** tab.
   c. Select **Index Items**.

Your instance setup with a dataset for a pre-populated database is now complete.

Complete your installation for dataset with a .jama backup file

Once the new instance is up and running, perform the following actions to finish setting up your instance with the dataset.

To complete your instance setup with the dataset:

1. If you want to use SAML on the new instance with a different configuration than the previous instance, enable SAML in your new instance. It is disabled by default.
2. Set `core.tenant-manager.application.env.RESTORE_BACKUP_FILE` to an empty string so that the core pod unmounts the hostPath volume.
3. If using `global.core.enableHorizontalScaling: true`, set `core.replicaCounts.jobs: 3` to have multiple core-jobs now that the .jama backup file has been restored.
4. Run a Helm upgrade to apply the value changes, if needed.
5. Delete the `/data/restore` directory from the node(s) because it is no longer required.

Your instance setup using a dataset created with a .jama backup file is now complete.

Upgrade Jama Connect (internet)

Use this workflow to upgrade Jama Connect in an internet Kubernetes environment.

Important considerations

- Values file used by the current Helm release, plus any changes required for the target version. Current values can be retrieved by running:

  ```
  helm get values jama-connect -n jama-connect -o yaml
  ```

- You have planned a maintenance window since upgrades require downtime.

To upgrade Jama Connect (internet):

1. For a dry run, execute the command, replacing `<version>` with the target Jama Connect version:

```
helm upgrade jama-connect oci://registry.jamasoftware.net/helm/jama-connect \
  -n jama-connect \
  -f values.yaml \
  --version <version> \
  --dry-run
```

2. If there were no errors, execute the same Helm upgrade command without the `--dry-run` parameter.

```
helm upgrade jama-connect oci://registry.jamasoftware.net/helm/jama-connect \
  -n jama-connect \
  -f values.yaml \
  --version <version> \
```

3. Monitor the pods until they are running:

```
kubectl get pods -n jama-connect
```

4. Use the configured Jama Connect URL to validate access to the application.

The application might display a message that the tenant is being upgraded.

Upgrade Jama Connect (airgap)

Use this workflow to upgrade Jama Connect in an airgap Kubernetes environment.

Important considerations

- Values file used by the current Helm release, plus any changes required for the target version. Current values can be retrieved by running:

```
helm get values jama-connect -n jama-connect -o yaml
```

- You have planned a maintenance window since upgrades require downtime.
- For airgap environments, you must prepare airgap artifacts.

To upgrade Jama Connect (airgap):

1. For a dry run, install the local Helm chart `tgz` file with this command, replacing `<version>` with the target Jama Connect version:

```
helm upgrade jama-connect jama-connect-<version>.tgz -f values.yaml -n
jama-connect --dry-run
```

2. If there were no errors, execute the same Helm upgrade command without the `--dry-run` parameter.

```
helm upgrade jama-connect jama-connect-<version>.tgz -f values.yaml -n
jama-connect
```

3. Monitor the pods until they are running:

```
kubectl get pods -n jama-connect
```

4. Use the configured Jama Connect URL to validate access to the application.

The application might show a message saying that the tenant is being upgraded.

Troubleshooting installation and upgrades

Use this information to troubleshoot any issues when you install or upgrade Jama Connect.

VERIFY PREFLIGHT CHECKS

- Before installing or upgrading Jama Connect, our Helm chart validates that the database server is accessible using the credentials provided.
- If installation or upgrade failed with an error similar to:

```
Error: INSTALLATION FAILED: failed pre-install: 1 error occurred:
        * job connect-preflight failed: BackoffLimitExceeded
```

- Run the following command to check for errors in the preflight checks:

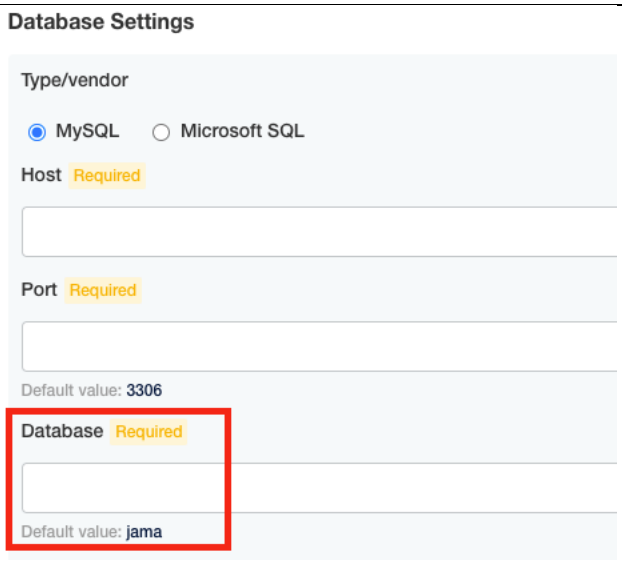
```
kubectl logs -l app.kubernetes.io/name=connect-preflight --all-containe
rs=true -n jama-connect
```

CREATE SUPPORT BUNDLE

Use the scripts in this GitHub repository to collect diagnostic information for troubleshooting.

FAQ

Question	Answer
What is my tenant name?	The tenant name is what you specified as the database schema in your Helm values: `global.database.schema`. In KOTS, the tenant name is what you specified as the database name in the Config tab of KOTS admin:

	
How can I retrieve a Jama Connect backup file?	Jama backup files are located in a directory shared by the core pods, so you need to retrieve the backup using `kubectl`. 1. Identify your FAQ and set it as an environment variable by replacing `<tenant-name>` in the following command: `export TENANT_NAME=<tenant-name>` 2. List all backups: `kubectl get pods -l app.kubernetes.io/name=core -o jsonpath='{.items[0].metadata.name}' | \` `xargs -r -I{} sh -c 'kubectl exec --tty -c core {} -- ls -la \` `/home/contour/tenant/${TENANT_NAME}/backup'` 3. Replace `<backup-name>` two times in the following command with the name of the backup that you want to retrieve: `kubectl get pods -l app.kubernetes.io/name=core -o jsonpath='{.items[0].metadata.name}' \| \` `xargs -r -I{} sh -c 'kubectl exec -i -c core {} -- cat \` `/home/contour/tenant/${TENANT_NAME}/backup/<backup-name> > ./<backup-name>'`
What are the core pods?	The name of the core pods starts with the `core-` prefix. For example: `core-0`, `core-ingress-0`, `core-jobs-0`, and `core-reports-0`.
What are tenant assets?	Tenant assets are files uploaded or generated by the application. For example: avatars, attachments, diagrams, reports, and equations.

How do I back up and restore my instance?	This topic is directly related to creating a new instance from a dataset. See Preparing a dataset for details on the two approaches that are supported.

Appendix A: Configuring the Helm chart

Use the following topics to configure the Helm chart.

- Configure custom memory setting for OpenSearch
- Configure container image registry
- Configure ingress
- Configure persistent storage
- Configure database
- Configure license
- Optional configuration

> **BEST PRACTICE:** To help you configure the Helm chart, use the examples for basic values and HA values that are available in this GitHub repository.

If you have an existing KOTS stack and you want to copy some of its settings, see Appendix C: Mapping KOTS configuration to Helm values.

Configure custom memory setting for OpenSearch

Use the following information to configure custom memory settings for OpenSearch.

OS/NODE METHOD (PREFERRED)

Refer to Prepare custom memory setting for OpenSearch.

INIT CONTAINERS METHOD (OPTIONAL)

Our chart optionally sets and checks the memory setting at pod startup. If your cluster forbids privileged pods, you must use the OS/node method.

Init containers available:

- **Checker** — Always runs. Verifies the node's `vm.max_map_count` is at least the required value. If not, the pod fails to start.

- **Setter** — Privileged. Only runs when explicitly enabled. It attempts to raise the value on the node before OpenSearch starts.

To enable the Setter initContainer, add the following to your `values.yaml` file:

```
core:
  search:
    opensearch:
      application:
        virtualMemory:
          autoSet: true
```

AVAILABLE RELATED PARAMETERS

Parameter	Description	Default
`core.search.opensearch.application.virtualMemory.autoSet`	If true, run a privileged initContainer to set vm.max_map_count.	false
`core.search.opensearch.application.virtualMemory.maxMapCount`	Desired vm.max_map_count value. Must be at least 262144. Higher values are accepted.	262144
`core.search.opensearch.application.virtualMemory.timeoutSeconds`	Seconds the checker container waits before failing.	60

Configure container image registry

Parameter	Description	Example
`global.repo.host`	Container registry for images.	registry.jamasoftware.net/docker
`global.repo.secrets`	List of `imagePullSecrets` names within the same namespace to use for pulling images.	jama-registry-creds

Configure ingress

Parameter	Description	Default	Example
`global.url.domain`	Host name used as the base URL of Jama Connect. Make sure this domain name is routable on your network. If you change this host		example.org

	name later, make sure to also change the base URL through the Jama Connect system admin (root user).		
`global.url.httpPort`	Http port that will be used as part of the base URL of Jama Connect. Only used when global.url.secure is false. If you change this port later, make sure to also change the base URL through the Jama Connect system admin (root user).	80	
`global.url.httpsPort`	Https port that will be used as part of the base URL of Jama Connect. Only used when global.url.secure is true. If you change this port later, make sure to also change the base URL through the Jama Connect system admin (root user).	443	
`global.url.secure`	Specifies if the base URL of Jama Connect will be accessed through a secure protocol (https).	true	
`ingress.className`	Class name of the ingress controller to use. If empty, default ingress class for the cluster is used.		
`ingress.annotations`	Map of annotations for the ingress resource.		

CONFIGURE TLS FOR INGRESS

When you set global.url.secure to true, you can enable TLS termination at the ingress resource level. You can bring your own certificate, let the chart generate a self-signed one, or reuse an existing Kubernetes TLS secret.

To do so, use the following parameters. The Priority column indicates the order in which the parameters are considered.

Parameter	Description	Priority	Default
`ingress.tls.secretName`	Name of an existing `kubernetes.io/tls` secret to use.	1	

`ingress.tls.cert`	PEM-encoded certificate to store in the TLS secret. Must match `ingress.tls.key`.	2	
`ingress.tls.key`	PEM-encoded private key that matches `ingress.tls.cert`.	2	
`ingress.tls.generateSelfSigned`	If true, generate a self-signed certificate when no TLS secret exists yet.	3	false

On installation and upgrade:

- **Custom certificate** — If you set `ingress.tls.cert` and `ingress.tls.key`, the chart always creates or updates the TLS secret with these values, even if a secret already exists. To renew/rotate your certificate, simply update these values and run Helm upgrade.
- **Self-signed certificate** — If `ingress.tls.generateSelfSigned: true` and there is **no TLS secret yet**, the chart generates a self-signed certificate for `global.url.domain` and stores it in the TLS secret. If the TLS secret already exists, the existing certificate is reused, and a new one is **not** generated. To **force a new self-signed certificate**, delete the existing TLS secret (or change to a custom cert/key) and run Helm upgrade.
- **Reuse an existing TLS secret** — If you specify `ingress.tls.secretName` (or there is already a TLS secret with the chart's configured name) and you do **not** set `ingress.tls.cert / ingress.tls.key`, the chart uses that secret name in our ingress. This is useful when the certificate is managed externally.
- **No TLS secret** — If no certificate is provided, `generateSelfSigned` is false, and no matching TLS secret exists, the chart does not create a TLS secret.

To see the TLS secret names, if any, used by the ingress resources in a namespace, run:

```
kubectl get ingress -n <namespace> -o jsonpath='{.items[*].spec.tls[*].
secretName}{"\\n"}'
```

If this command returns nothing, no ingress in that namespace is currently configured with TLS.

OPTIONAL INGRESS CONFIGURATION

If your ingress controller requires it, you can configure the service resources used by the ingress resource:

Parameter	Description	Default	Example

`core.service.type`	Type of the core service resource.	ClusterIP	NodePort
`core.service.annotations`	Map of annotations for the core service resource.		alb.ingress.kubernetes.io/healthcheck-path: /
`core.diff.service.type`	Type of the diff service resource.	ClusterIP	NodePort
`core.diff.service.annotations`	Map of annotations for the diff service resource.		alb.ingress.kubernetes.io/healthcheck-path: /health
`core.saml.service.type`	Type of the SAML service resource.	ClusterIP	NodePort
`core.saml.service.annotations`	Map of annotations for the SAML service resource.		alb.ingress.kubernetes.io/healthcheck-path: /health

Additionally, we bundle a Traefik ingress controller inside our chart. To use it, see Traefik ingress controller.

Configure persistent storage

Parameter	Description	Default
`global.storageClass`	Name of the storage class to use when creating persistent volume claims. If empty, default storage class for the cluster is used.	
`core.application.storage.sharedVolume.storageClass`	Name of the storage class for the assets volume. If empty, fallbacks to `global.storageClass`.	
`core.application.storage.sharedVolume.request`	Size for the assets volume. Examples of assets that Connect saves: attachments, avatars, reports, backups, etc.	7 Gi
`core.application.storage.sharedVolume.accessMode`	Access mode for the assets volume. If planning to enable horizontal scaling of core pods across multiple nodes, set this to `ReadWriteMany`.	ReadWriteOnce
`core.activemq.application.storage.accessMode`	Access mode for ActiveMQ's persistent volumes. If high availability is desired for ActiveMQ, set this to `ReadWriteMany`.	ReadWriteOnce

`core.activemq.application.storage.request`	Size for each volume associated to an ActiveMQ pod.	
`core.search.opensearch.application.storage.request`	Size for each volume associated with an OpenSearch pod.	10 Gi

Configure database

Parameter	Description	Default	Example
`global.database.type`	Database type/vendor. Possible values: • mysql • mssql	mysql	
`global.database.host`	Database host.		db.example.com
`global.database.port`	Database port.	• "3306" for mysql • "1433" for mssql	
`global.database.schema`	Database schema.	jama	
`global.database.connectionParameters`	Database connection parameters.		
`global.database.driver`	Database driver. Possible values are: • com.mysql.jdbc.Driver • net.sourceforge.jtds.jdbc.Driver • com.microsoft.sqlserver.jdbc.SQLServerDriver (don't use unless specifically instructed)	• com.mysql.jdbc.Driver for mysql • net.sourceforge.jtds.jdbc.Driver for mssql	
`core.application.database.connect.username`	Database username.	jamauser	
`core.application.database.connect.password`	Database password.		

`core.oauth.application.database.schema`	OAuth database schema.	oauth	
`core.oauth.application.database.username`	OAuth database username.	oauthuser	
`core.oauth.application.database.password`	OAuth database password.		
`core.oauth.application.database.connectionParameters`	OAuth database connection parameters. Only used when using mysql.		
`core.saml.application.database.schema`	SAML database schema.	saml	
`core.saml.application.database.username`	SAML database username.	samluser	
`core.saml.application.database.password`	SAML database password.		
`core.saml.application.database.connectionParameters`	SAML database connection parameters. Only used when using mysql.		

Configure license

Parameter	Description
`core.tenant-manager.application.license`	Jama license

Optional configuration

Use the information in the following topics for optional configuration of your environment.

- Trusted certificates
- Horizontal scaling of core pods
- Traefik ingress controller
- Memory and CPU settings
- Wiris
- OpenSearch cluster
- High Availability

- Restore from .jama backup
- Enable JMX monitoring

Trusted certificates

You can use a PEM-formatted file containing any certificates that Jama needs to trust. The file might contain a concatenation of multiple PEM-formatted certificates to trust. For example, you might need this functionality to make an encrypted connection from Jama to your database, if the database is protected with a self-signed certificate.

Parameter	Description
`core.secrets.trustedCertificates.data`	Content of the PEM file.

When a TLS certificate is available for ingress (see "Configure TLS for ingress" in Configure ingress), the chart:

- Creates a dedicated secret for trusted certificates
- Stores the concatenation of:
  - Any PEM content provided in `core.secrets.trustedCertificates.data`
  - The ingress TLS certificate

This allows Jama Connect to trust both your own custom CAs or intermediate certificates and the same certificate that is used for HTTPS at ingress.

Horizontal scaling of core pods

Enable horizontal scaling of core pods to deploy specialized core pods instead of a single one.

Before enabling this feature, consider the following:

- If this is the first time you are installing Connect in the cluster, **DO NOT** enable this feature. Once Jama Connect is installed and working properly, you can enable horizontal scaling safely. If you are restoring a backup, restore it without horizontal scaling enabled.
- You must provide a new database schema and user for Quartz to use.
- You can configure the number of replicas for each role. Once you have increased the number of replicas, **DO NOT** decrease it.

Parameter	Description	Default
`global.core.enableHorizontalScaling`	Enables horizontal scaling of core pods	false

`core.replicaCounts.ingress`	Number of core ingress pods.	1
`core.replicaCounts.jobs`	Number of core jobs pods.	1
`core.replicaCounts.reports`	Number of core reports pods.	1
`core.topologySpreadConstraints.maxSkew`	Describes the degree to which core pods might be unevenly distributed. You must specify this field, and the number must be greater than zero.	1
`core.topologySpreadConstraints.topologyKey`	The key of node labels. Nodes that have a label with this key and identical values are considered to be in the same topology.	topology.kubernetes.io/zone
`core.topologySpreadConstraints.whenUnsatisfiable`	Indicates how to deal with a core pod if it doesn't satisfy the spread constraint.	DoNotSchedule
`core.application.database.quartz.schema`	Quartz database schema.	quartz
`core.application.database.quartz.username`	Quartz database username.	quartzuser
`core.application.database.quartz.password`	Quartz database password.	

Traefik ingress controller

This table shows the main parameters that you can pass. For a full list, see the official Traefik documentation.

Parameter	Description	Default	Example
`traefik.enabled`	Specifies if the bundled Traefik ingress controller should be deployed. If true, our ingress resource is auto-configured to use it.	false	

`global.url.useTraefikNodePorts`	If true, the Traefik ingress controller overrides the http and https ports defined in global.url.	true	
`traefik.image.registry`	Registry host to pull Traefik images from.	docker.io	
`traefik.deployment.kind`	Kind of resource.	DaemonSet	Deployment
`traefik.service.type`	Type of the Traefik service resource.	NodePort	
`traefik.ports.web.nodePort`	HTTP port number. This overrides the value set for `global.url.httpPort` when `traefik.service.type` is set to `NodePort` and `global.url.useTraefikNodePorts` is true.	80	
`traefik.ports.websecure.nodePort`	HTTPS port number. This overrides the value set for `global.url.httpPort` when `traefik.service.type` is set to `NodePort` and `global.url.useTraefikNodePorts` is true.	443	
`traefik.resources.limits.cpu`	Max CPU for each Traefik pod.	1000 m	
`traefik.resources.limits.memory`	Max memory for each Traefik pod.	512 Mi	

Memory and CPU settings

In memory values, you MUST use a literal size, such as 2 G or 1000 M or 2000000 k, although sensible defaults are given. It is unlikely that you need to override them.

In CPU values, 1000 m is equal to one core.

In case of any doubt, don't make any changes and contact Support.

Parameter	Description	Default
`core.resources.limits.cpu`	Max CPU for each core pod	4000 m

core.resources.limits.memory	Max memory for each core pod	12 G
core.application.env.MAX_HEAP	Max memory for each core container	8 G
core.activemq.resources.limits.cpu	Max CPU for the activemq pod	1000 m
core.activemq.resources.limits.memory	Max memory for the activemq pod	1 G
core.activemq.application.java.xmx	Max memory for the activemq container	700M
core.diff.resources.limits.cpu	Max CPU for the diff pod	1000 m
core.diff.resources.limits.memory	Max memory for the diff pod	512 M
core.diff.application.env.MAX_HEAP	Max memory for the diff container	512 M
core.hazelcast.resources.limits.cpu	Max CPU for the Hazelcast pod	1000 m
core.hazelcast.resources.limits.memory	Max memory for the Hazelcast pod	7 G
core.hazelcast.application.env.MAX_HEAP	Max memory for the Hazelcast container	6 G
core.oauth.resources.limits.cpu	Max CPU for the oauth pod	1000 m
core.oauth.resources.limits.memory	Max memory for the oauth pod	1 G
core.oauth.application.env.MAX_HEAP	Max memory for the oauth container	1 G
core.saml.resources.limits.cpu	Max CPU for the saml pod	1000 m
core.saml.resources.limits.memory	Max memory for the saml pod	1 G
core.saml.application.env.MAX_HEAP	Max memory for the saml container	1 G
core.search.resources.limits.cpu	Max CPU for the search pod	2000 m

core.search.resources.limits.memory	Max memory for the search pod	3 G
core.search.application.env.MAX_HEAP	Max memory for the search container	2 G
core.search.opensearch.resources.limits.cpu	Max CPU for the OpenSearch pod	2000 m
core.search.opensearch.resources.limits.memory	Max memory for the OpenSearch pod	5 G
core.search.opensearch.application.env.MAX_HEAP	Max memory for the OpenSearch container	4 G

Wiris

Use these settings to configure a custom connection to your own wiris instance.
In case of any doubt, don't make any changes and contact Jama Support.

Parameter	Description	Default
core.application.env.WIRIS_CUSTOM_CONFIG_ENABLED	Enables custom Wiris connection	false
core.application.env.WIRIS_HOST	Wiris host	
core.application.env.WIRIS_PATH	Wiris path	/client/editor/render
core.application.env.WIRIS_PORT	Wiris port	
core.application.env.WIRIS_PROTOCOL	Wiris protocol	https
core.application.env.WIRIS_PROXY_ENABLED	Use proxy for Wiris	false
core.secrets.wirisProxy.username	Wiris proxy user	
core.secrets.wirisProxy.password	Wiris proxy password	

OpenSearch cluster

The properties that control settings to enable multiple OpenSearch pods are:

Parameter	Description	Default

`core.search.opensearch.replicaCount`	Number of pods for the OpenSearch component	1
`core.search.opensearch.topologySpreadConstraints.maxSkew`	Describes the degree to which OpenSearch pods might be unevenly distributed. You must specify this field, and the number must be greater than zero.	1
`core.search.opensearch.topologySpreadConstraints.topologyKey`	The key of node labels. Nodes that have a label with this key and identical values are considered to be in the same topology.	topology.kubernetes.io/zone
`core.search.opensearch.topologySpreadConstraints.whenUnsatisfiable`	Indicates how to deal with an OpenSearch pod if it doesn't satisfy the spread constraint.	DoNotSchedule

High Availability

To enable HA mode for Jama Connect, you must create a Kubernetes cluster with nodes on different availability zones and set a storage class that supports RWX for the tenant assets. See Configure persistent storage.

Besides that, you must configure the HA properties for each Jama Connect component:

- HA settings for Traefik ingress controller
- HA settings for opensearch
- HA settings for activemq
- HA settings for diff
- HA settings for hazelcast
- HA settings for oauth
- HA settings for saml
- HA settings for search

HA SETTINGS FOR TRAEFIK INGRESS CONTROLLER

When using Traefik ingress controller (traefik.enabled set to true) in high availability mode, you must:

1. Set up a load balancer that forwards requests to the Kubernetes nodes using the ports defined for Traefik in Traefik ingress controller. If your load balancer handles TLS

termination, you can disable TLS termination at ingress level by leaving ingress.tls.cert and ingress.tls.key empty.

2. Set the following properties:

Parameter	Description	Default
`global.url.useTraefikNodePorts`	This must be set to `false`, to avoid overriding the http and https ports defined in `global.url`.	true
`traefik.ports.web.forwardedHeaders.insecure`	This must be set to `true`, so that Traefik passes the incoming X-Forwarded-* headers to upstreams.	false
`traefik.ports.websecure.forwardedHeaders.insecure`	This must be set to `true`, so that Traefik passes the incoming X-Forwarded-* headers to upstreams.	false

HA SETTINGS FOR OPENSEARCH

The properties that control HA settings for OpenSearch pods are detailed in OpenSearch cluster.

HA SETTINGS FOR ACTIVEMQ

ActiveMQ uses a shared-store HA model in this chart. That means it supports only:

- `core.activemq.replicaCount: 1` for standalone mode.
- `core.activemq.replicaCount: 2` for HA mode (1 live broker + 1 backup broker).

IMPORTANT: Only values 1 and 2 are supported for ActiveMQ HA in this chart.

The properties that control HA settings for activemq component are:

Parameter	Description	Default
`core.activemq.replicaCount`	Number of pods for the activemq component. Set to 1 for standalone mode or 2 for HA mode. In HA mode, both brokers share the same data store, but only one broker is active at a time. The second broker remains in standby and takes over if the active broker fails.	1
`core.activemq.topologySpreadConstraints.maxSkew`	Describes the degree to which activemq pods might be unevenly distributed. You must specify this	1

	field, and the number must be greater than zero.	
`core.activemq.topologySpreadConstraints.topologyKey`	The key of node labels. Nodes that have a label with this key and identical values are considered to be in the same topology.	topology.kubernetes.io/zone
`core.activemq.topologySpreadConstraints.whenUnsatisfiable`	Indicates how to deal with an activemq pod if it doesn't satisfy the spread constraint.	DoNotSchedule
`core.activemq.application.storage.class`	The storage class to use for ActiveMQ persistent data. For HA, you must provide a storage class that can provision persistent volumes with ReadWriteMany access mode so that the shared broker data is accessible across availability zones.	''

HA SETTINGS FOR DIFF

The properties that control HA settings for diff component are:

Parameter	Description	Default
`core.diff.replicaCount`	Number of pods for diff component. To enable HA mode, this should be set to an odd number, usually 3.	1
`core.diff.topologySpreadConstraints.maxSkew`	Describes the degree to which diff pods might be unevenly distributed. You must specify this field, and the number must be greater than zero.	1
`core.diff.topologySpreadConstraints.topologyKey`	The key of node labels. Nodes that have a label with this key and identical values are considered to be in the same topology.	topology.kubernetes.io/zone
`core.diff.topologySpreadConstraints.whenUnsatisfiable`	Indicates how to deal with a diff pod if it doesn't satisfy the spread constraint.	DoNotSchedule

HA SETTINGS FOR HAZELCAST

The properties that control HA settings for Hazelcast component are:

Parameter	Description	Default
`core.hazelcast.replicaCount`	Number of pods for Hazelcast component. To enable HA mode, this should be set to an odd number, usually 3.	1
`core.hazelcast.topologySpreadConstraints.maxSkew`	Describes the degree to which Hazelcast pods might be unevenly distributed. You must specify this field, and the number must be greater than zero.	1
`core.hazelcast.topologySpreadConstraints.topologyKey`	The key of node labels. Nodes that have a label with this key and identical values are considered to be in the same topology.	topology.kubernetes.io/zone
`core.hazelcast.topologySpreadConstraints.whenUnsatisfiable`	Indicates how to deal with a Hazelcast pod if it doesn't satisfy the spread constraint.	DoNotSchedule

HA SETTINGS FOR OAUTH

The properties that control HA settings for oauth component are:

Parameter	Description	Default
`core.oauth.replicaCount`	Number of pods for oauth component. To enable HA mode, this should be set to an odd number, usually 3.	1
`core.oauth.topologySpreadConstraints.maxSkew`	Describes the degree to which oauth pods might be unevenly distributed. You must specify this field, and the number must be greater than zero.	1
`core.oauth.topologySpreadConstraints.topologyKey`	The key of node labels. Nodes that have a label with this key and identical values are considered to be in the same topology.	topology.kubernetes.io/zone
`core.oauth.topologySpreadConstraints.whenUnsatisfiable`	Indicates how to deal with an oauth pod if it doesn't satisfy the spread constraint.	DoNotSchedule

HA SETTINGS FOR SAML

The properties that control HA settings for saml component are:

Parameter	Description	Default
`core.saml.replicaCount`	Number of pods for saml component. To enable HA mode, this should be set to an odd number, usually 3.	1
`core.saml.topologySpreadConstraints.maxSkew`	Describes the degree to which saml pods might be unevenly distributed. You must specify this field, and the number must be greater than zero.	1
`core.saml.topologySpreadConstraints.topologyKey`	The key of node labels. Nodes that have a label with this key and identical values are considered to be in the same topology.	topology.kubernetes.io/zone
`core.saml.topologySpreadConstraints.whenUnsatisfiable`	Indicates how to deal with a saml pod if it doesn't satisfy the spread constraint.	DoNotSchedule

HA SETTINGS FOR SEARCH

The properties that control HA settings for search component are:

Parameter	Description	Default
`core.search.replicaCount`	Number of pods for search component. To enable HA mode, this should be set to an odd number, usually 3.	1
`core.search.topologySpreadConstraints.maxSkew`	Describes the degree to which search pods might be unevenly distributed. You must specify this field, and the number must be greater than zero.	1
`core.search.topologySpreadConstraints.topologyKey`	The key of node labels. Nodes that have a label with this key and identical values are considered to be in the same topology.	topology.kubernetes.io/zone

`core.search.topologySpreadConstraints.whenUnsatisfiable`	Indicates how to deal with a search pod if it doesn't satisfy the spread constraint.	DoNotSchedule

Restore from .jama backup

To restore from an existing Jama backup, use the following parameters:

Parameter	Description	Default	Example
`core.application.storage.core.path`	The restore process looks for backup files in this directory, which is mounted into the pods using a hostPath volume.	/data/restore	
`core.tenant-manager.application.env.RESTORE_BACKUP_FILE`	The backup file's absolute path.	''	/data/restore/myback up.jama

See Use a .jama backup file to configure a dataset for more details on this restore mechanism.

Enable JMX monitoring

JMX can be enabled for the JVM-based components core, search, OpenSearch, and diff to expose standard management data (heap usage, thread counts, class loading, OS metrics, system properties, and any registered MBeans). Each component is configured independently and listens on its own fixed JMX port inside the pod.

JMX SETTINGS

Parameter	Description	Default
`core.application.env.ENABLE_JAVA_JMX_REMOTE`	Enables JMX on the core JVM.	false
`core.application.env.JAVA_JMX_REMOTE_PORT`	JMX port on the core pod.	9090
`core.search.application.env.ENABLE_JAVA_JMX_REMOTE`	Enables JMX on the search JVM.	false

core.search.application.env.JAVA_JMX_REMOTE_PORT	JMX port on the search pod.	9091
core.search.opensearch.application.env.ENABLE_JAVA_JMX_REMOTE	Enables JMX on the OpenSearch JVM.	false
core.search.opensearch.application.env.JAVA_JMX_REMOTE_PORT	JMX port on the OpenSearch pod.	9092
core.diff.application.env.ENABLE_JAVA_JMX_REMOTE	Enables JMX on the diff JVM.	false
core.diff.application.env.JAVA_JMX_REMOTE_PORT	JMX port on the diff pod.	9093

JMX RMI HOSTNAME

You can define a global RMI hostname to be used by core, search, OpenSearch, and diff pods.

By default, each pod uses its own IP, which might not be accessible externally. If needed, set this to a hostname that can be reached from outside the cluster.

Parameter	Description	Default
global.jmx.javaRmiHostName	RMI hostname to use for all JMX-enabled pods (core, search, OpenSearch and diff)	status.podIP

Appendix B: Back up to .jama or XML file

You can back up your data to a .jama or XML file for migrations and refreshes. Because this method has built-in automation, you can avoid manual changes that might impact the integrity of the data.

A .jama file backup includes the database and /data directory. An XML file backup includes only the database.

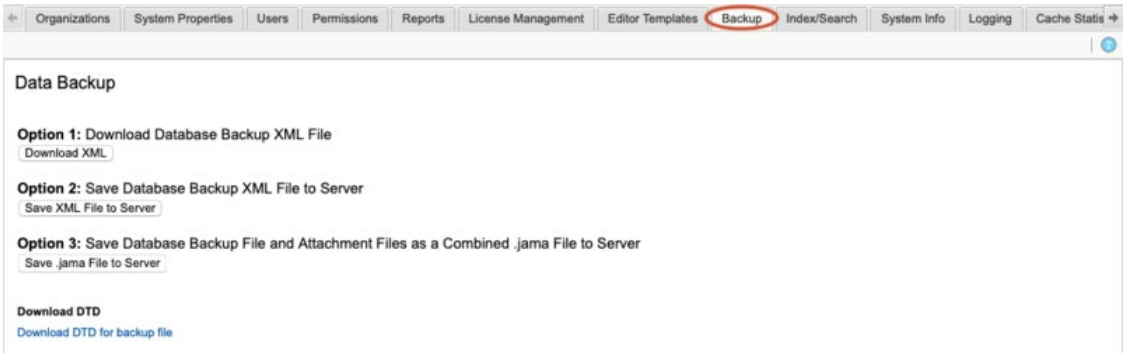
Important considerations

- If SAML is enabled, disable it before backing up your data. After you restore your instance of Jama Connect, you must re-enable SAML.
- Backups must be done manually; they can't be scheduled automatically.
- Make sure you have enough available disk space.

- Make sure Jama Connect is in maintenance mode before you create a backup.
- If you are using a version of Jama Connect prior to version 8, generate backups during a maintenance window and inform users, including API users, not to use the application.
- Regardless of what version you are using, all integrations must be disabled.

To back up to a .jama or XML file:

1. Log in to Jama Connect as the root user.
2. Select the **Backup** tab to see the backup options.



3. Choose a backup method (listed here in recommended order).

Method	Scenario	Select...
Save .jama file to the application server.	Migrating between versions later than 8.0.	**Save .jama File to server**
Save XML file to the application server.	• Migrating between different types of databases • Migrating version earlier than 8.0	**Save XML File to Server**
Download XML database backup to your workstation.	• If you can't access the application server • For smaller databases	**Download XML**
Download document type definition (DTD) to your workstation.	If all other methods fail	**Download DTD**

The backup process is complete.

Appendix C: Mapping KOTS configuration to Helm values

If you have an existing KOTS stack and want to copy some of the settings from the KOTS Config tab, use the following tables to identify the equivalent Helm value for each field.

Core Jama Application Settings

KOTS config field	Equivalent Helm chart value
Core Jama Application Settings > Requests CPU	`core.resources.requests.cpu`
Core Jama Application Settings > Max CPU	`core.resources.limits.cpu`
Core Jama Application Settings > Requests Memory	`core.resources.requests.memory`
Core Jama Application Settings > Max Memory	`core.application.env.MAX_HEAP`
Core Jama Application Settings > Max Memory for Container	`core.resources.limits.memory`
Core Jama Application Settings > Enable Horizontal Scaling	`global.core.enableHorizontalScaling`
Core Jama Application Settings > Minimum amount of ingress nodes	`core.replicaCounts.ingress`
Core Jama Application Settings > Minimum amount of job nodes	`core.replicaCounts.jobs`
Core Jama Application Settings > Minimum amount of report nodes	`core.replicaCounts.reports`
Core Jama Application Settings > Liveness Probe > Initial Delay	`core.container.livenessInitialDelay`
Core Jama Application Settings > Liveness Probe > Period	`core.container.livenessPeriod`
Core Jama Application Settings > Liveness Probe > Timeout	`core.container.livenessTimeout`
Core Jama Application Settings > Liveness Probe > Success Threshold	`core.container.livenessSuccessThreshold`
Core Jama Application Settings > Liveness Probe > Failure Threshold	`core.container.livenessFailureThreshold`

Core Jama Application Settings > Readiness Probe > Initial Delay	`core.container.readinessInitialDelay`
Core Jama Application Settings > Readiness Probe > Period	`core.container.readinessPeriod`
Core Jama Application Settings > Readiness Probe > Timeout	`core.container.readinessTimeout`
Core Jama Application Settings > Readiness Probe > Success Threshold	`core.container.readinessSuccessThreshold`
Core Jama Application Settings > Readiness Probe > Failure Threshold	`core.container.readinessFailureThreshold`

Ingress

KOTS config field	Equivalent Helm chart value
Kubernetes Configuration > Ingress Class Name	`ingress.className`
Web Server > Context path	`global.url.contextPath`
Web Server > Use TLS	`global.url.secure`
Web Server > Port	`global.url.httpPort` `If using the bundled Traefik:` `traefik.ports.web.nodePort`
SSL Settings > TLS port	`global.url.httpsPort` `If using the bundled Traefik:` `traefik.ports.websecure.nodePort`
Host Name > Host name	`global.url.domain`
Host Name > TLS Key Pair Source > Custom TLS Configuration TLS Configuration > Private Key TLS Configuration > Certificate	`ingress.tls.key` `ingress.tls.cert`
Host Name > TLS Key Pair Source > Generate New	`ingress.tls.generateSelfSigned`
Trusted Certificates > Upload certificate file	`core.secrets.trustedCertificates.data`

External SSL Settings > TLS managed externally	`traefik.ports.web.forwardedHeaders.insecure` `traefik.ports.websecure.forwardedHeaders.insecure`

Storage

KOTS config field	Equivalent Helm chart value
Storage > Storage Class	`global.storageClass`
Storage > Assets Size	`core.application.storage.sharedVolume.request`
Storage > Assets Storage Class	`core.application.storage.sharedVolume.storageClass`
Storage > Assets PVC Access Mode	`core.application.storage.sharedVolume.accessMode`

Database

KOTS config field	Equivalent Helm chart value
Database Settings > Type/vendor	`global.database.type`
Database Settings > Host	`global.database.host`
Database Settings > Port	`global.database.port`
Database Settings > Database	`global.database.schema`
Database Settings > User name	`core.application.database.connect.username`
Database Settings > Password	`core.application.database.connect.password`
Database Settings > SAML database schema	`core.saml.application.database.schema`
Database Settings > SAML user name	`core.saml.application.database.username`
Database Settings > SAML password	`core.saml.application.database.password`
Database Settings > OAuth database schema	`core.oauth.application.database.schema`
Database Settings > OAuth user name	`core.oauth.application.database.username`
Database Settings > OAuth password	`core.oauth.application.database.password`
Database Settings > Quartz database schema	`core.application.database.quartz.schema`

Database Settings > Quartz user name	`core.application.database.quartz.username`
Database Settings > Quartz password	`core.application.database.quartz.password`
Advanced DB Settings > Database connection parameters	`global.database.connectionParameters`
Advanced DB Settings > SAML database schema connection parameters	`core.saml.application.database.connectionParameters`
Advanced DB Settings > OAuth database schema connection parameters	`core.oauth.application.database.connectionParameters`
Advanced DB Settings > Database driver	`global.database.driver` Possible values: • `net.sourceforge.jtds.jdbc.Driver` • `com.microsoft.sqlserver.jdbc.SQLServerDriver`

OpenSearch Settings

KOTS config field	Equivalent Helm chart value
OpenSearch Settings > Max CPU	`core.search.opensearch.resources.limits.cpu`
OpenSearch Settings > Max Memory	`core.search.opensearch.application.env.MAX_HEAP`
OpenSearch Settings > Max Memory for Container	`core.search.opensearch.resources.limits.memory`
OpenSearch Settings > Volume Size	`core.search.opensearch.application.storage.request`
OpenSearch Settings > Amount of OpenSearch nodes	`core.search.opensearch.replicaCount`
OpenSearch Settings > Liveness Probe > Initial Delay	`core.search.opensearch.container.livenessInitialDelay`
OpenSearch Settings > Liveness Probe > Period	`core.search.opensearch.container.livenessPeriod`
OpenSearch Settings > Liveness Probe > Timeout	`core.search.opensearch.container.livenessTimeout`
OpenSearch Settings > Liveness Probe > Success Threshold	`core.search.opensearch.container.livenessSuccessThreshold`

OpenSearch Settings > Liveness Probe > Failure Threshold	`core.search.opensearch.container.livenessFailureThreshold`
OpenSearch Settings > Readiness Probe > Initial Delay	`core.search.opensearch.container.readinessInitialDelay`
OpenSearch Settings > Readiness Probe > Period	`core.search.opensearch.container.readinessPeriod`
OpenSearch Settings > Readiness Probe > Timeout	`core.search.opensearch.container.readinessTimeout`
OpenSearch Settings > Readiness Probe > Success Threshold	`core.search.opensearch.container.readinessSuccessThreshold`
OpenSearch Settings > Readiness Probe > Failure Threshold	`core.search.opensearch.container.readinessFailureThreshold`

Search Service Settings

KOTS config field	Equivalent Helm chart value
Search Service Settings > Max CPU	`core.search.resources.limits.cpu`
Search Service Settings > Max Memory For Service	`core.search.application.env.MAX_HEAP`
Search Service Settings > Max Memory for Container	`core.search.resources.limits.memory`
Search Service Settings > Number of primary shards	`core.search.application.env.DEFAULT_NUMBER_OF_SHARDS`
Search Service Settings > Number of replicas	`core.search.application.env.DEFAULT_NUMBER_OF_REPLICAS`
Search Service Settings > Liveness Probe > Initial Delay	`core.search.container.livenessInitialDelay`
Search Service Settings > Liveness Probe > Period	`core.search.container.livenessPeriod`
Search Service Settings > Liveness Probe > Timeout	`core.search.container.livenessTimeout`
Search Service Settings > Liveness Probe > Success Threshold	`core.search.container.livenessSuccessThreshold`
Search Service Settings > Liveness Probe > Failure Threshold	`core.search.container.livenessFailureThreshold`

Search Service Settings > Readiness Probe > Initial Delay	`core.search.container.readinessInitialDelay`
Search Service Settings > Readiness Probe > Period	`core.search.container.readinessPeriod`
Search Service Settings > Readiness Probe > Timeout	`core.search.container.readinessTimeout`
Search Service Settings > Readiness Probe > Success Threshold	`core.search.container.readinessSuccessThreshold`
Search Service Settings > Readiness Probe > Failure Threshold	`core.search.container.readinessFailureThreshold`

ActiveMQ Service Settings

KOTS config field	Equivalent Helm chart value
ActiveMQ Service Settings > Max CPU	`core.activemq.resources.limits.cpu`
ActiveMQ Service Settings > Max Memory	`core.activemq.application.java.xmx`
ActiveMQ Service Settings > Max Memory for Container	`core.activemq.resources.limits.memory`
ActiveMQ Service Settings > Min Large Message Size in Bytes	`core.activemq.application.client.minLargeMessageSizeInBytes`
ActiveMQ Service Settings > Liveness Probe > Initial Delay	`core.activemq.container.livenessInitialDelay`
ActiveMQ Service Settings > Liveness Probe > Period	`core.activemq.container.livenessPeriod`
ActiveMQ Service Settings > Liveness Probe > Timeout	`core.activemq.container.livenessTimeout`
ActiveMQ Service Settings > Liveness Probe > Success Threshold	`core.activemq.container.livenessSuccessThreshold`
ActiveMQ Service Settings > Liveness Probe > Failure Threshold	`core.activemq.container.livenessFailureThreshold`
ActiveMQ Service Settings > Readiness Probe > Initial Delay	`core.activemq.container.readinessInitialDelay`
ActiveMQ Service Settings > Readiness Probe > Period	`core.activemq.container.readinessPeriod`

ActiveMQ Service Settings > Readiness Probe > Timeout	`core.activemq.container.readinessTimeout`
ActiveMQ Service Settings > Readiness Probe > Success Threshold	`core.activemq.container.readinessSuccessThreshold`
ActiveMQ Service Settings > Readiness Probe > Failure Threshold	`core.activemq.container.readinessFailureThreshold`

Diff Service Settings

KOTS config field	Equivalent Helm chart value
Diff Service Settings > Max CPU	`core.diff.resources.limits.cpu`
Diff Service Settings > Max Memory	`core.diff.application.env.MAX_HEAP` `core.diff.resources.limits.memory`
Diff Service Settings > Liveness Probe > Initial Delay	`core.diff.container.livenessInitialDelay`
Diff Service Settings > Liveness Probe > Period	`core.diff.container.livenessPeriod`
Diff Service Settings > Liveness Probe > Timeout	`core.diff.container.livenessTimeout`
Diff Service Settings > Liveness Probe > Success Threshold	`core.diff.container.livenessSuccessThreshold`
Diff Service Settings > Liveness Probe > Failure Threshold	`core.diff.container.livenessFailureThreshold`
Diff Service Settings > Readiness Probe > Initial Delay	`core.diff.container.readinessInitialDelay`
Diff Service Settings > Readiness Probe > Period	`core.diff.container.readinessPeriod`
Diff Service Settings > Readiness Probe > Timeout	`core.diff.container.readinessTimeout`
Diff Service Settings > Readiness Probe > Success Threshold	`core.diff.container.readinessSuccessThreshold`
Diff Service Settings > Readiness Probe > Failure Threshold	`core.diff.container.readinessFailureThreshold`

Hazelcast Service Settings

KOTS config field	Equivalent Helm chart value
Hazelcast Service Settings > Max CPU	`core.hazelcast.resources.limits.cpu`
Hazelcast Service Settings > Max Memory	`core.hazelcast.application.env.MAX_HEAP_SIZE`
Hazelcast Service Settings > Max Memory for Container	`core.hazelcast.resources.limits.memory`
Hazelcast Service Settings > Liveness Probe > Initial Delay	`core.hazelcast.container.livenessInitialDelay`
Hazelcast Service Settings > Liveness Probe > Period	`core.hazelcast.container.livenessPeriod`
Hazelcast Service Settings > Liveness Probe > Timeout	`core.hazelcast.container.livenessTimeout`
Hazelcast Service Settings > Liveness Probe > Success Threshold	`core.hazelcast.container.livenessSuccessThreshold`
Hazelcast Service Settings > Liveness Probe > Failure Threshold	`core.hazelcast.container.livenessFailureThreshold`
Hazelcast Service Settings > Readiness Probe > Initial Delay	`core.hazelcast.container.readinessInitialDelay`
Hazelcast Service Settings > Readiness Probe > Period	`core.hazelcast.container.readinessPeriod`
Hazelcast Service Settings > Readiness Probe > Timeout	`core.hazelcast.container.readinessTimeout`
Hazelcast Service Settings > Readiness Probe > Success Threshold	`core.hazelcast.container.readinessSuccessThreshold`
Hazelcast Service Settings > Readiness Probe > Failure Threshold	`core.hazelcast.container.readinessFailureThreshold`

Traefik Settings

KOTS config field	Equivalent Helm chart value
Traefik > Max CPU	`traefik.resources.limits.cpu`
Traefik > Max Memory for Container	`traefik.resources.limits.memory`

OAuth Service Settings

KOTS config field	Equivalent Helm chart value
OAuth Service Settings > Max CPU	`core.oauth.resources.limits.cpu`
OAuth Service Settings > Max Memory	`core.oauth.application.env.MAX_HEAP` `core.oauth.resources.limits.memory`
OAuth Service Settings > Liveness Probe > Initial Delay	`core.oauth.container.livenessInitialDelay`
OAuth Service Settings > Liveness Probe > Period	`core.oauth.container.livenessPeriod`
OAuth Service Settings > Liveness Probe > Timeout	`core.oauth.container.livenessTimeout`
OAuth Service Settings > Liveness Probe > Success Threshold	`core.oauth.container.livenessSuccessThreshold`
OAuth Service Settings > Liveness Probe > Failure Threshold	`core.oauth.container.livenessFailureThreshold`
OAuth Service Settings > Readiness Probe > Initial Delay	`core.oauth.container.readinessInitialDelay`
OAuth Service Settings > Readiness Probe > Period	`core.oauth.container.readinessPeriod`
OAuth Service Settings > Readiness Probe > Timeout	`core.oauth.container.readinessTimeout`
OAuth Service Settings > Readiness Probe > Success Threshold	`core.oauth.container.readinessSuccessThreshold`
OAuth Service Settings > Readiness Probe > Failure Threshold	`core.oauth.container.readinessFailureThreshold`

SAML Service Settings

KOTS config field	Equivalent Helm chart value
SAML Service Settings > Max CPU	`core.saml.resources.limits.cpu`
SAML Service Settings > Max Memory	`core.saml.application.env.MAX_HEAP` `core.saml.resources.limits.memory`
SAML Service Settings > Liveness Probe > Initial Delay	`core.saml.container.livenessInitialDelay`

SAML Service Settings > Liveness Probe > Period	`core.saml.container.livenessPeriod`
SAML Service Settings > Liveness Probe > Timeout	`core.saml.container.livenessTimeout`
SAML Service Settings > Liveness Probe > Success Threshold	`core.saml.container.livenessSuccessThreshold`
SAML Service Settings > Liveness Probe > Failure Threshold	`core.saml.container.livenessFailureThreshold`
SAML Service Settings > Readiness Probe > Initial Delay	`core.saml.container.readinessInitialDelay`
SAML Service Settings > Readiness Probe > Period	`core.saml.container.readinessPeriod`
SAML Service Settings > Readiness Probe > Timeout	`core.saml.container.readinessTimeout`
SAML Service Settings > Readiness Probe > Success Threshold	`core.saml.container.readinessSuccessThreshold`
SAML Service Settings > Readiness Probe > Failure Threshold	`core.saml.container.readinessFailureThreshold`

Wiris

KOTS config field	Equivalent Helm chart value
WIRIS Connection Settings > Use custom Wiris connection	`core.application.env.WIRIS_CUSTOM_CONFIG_ENABLED`
WIRIS Connection Settings > Wiris Host	`core.application.env.WIRIS_HOST`
WIRIS Connection Settings > Wiris Path	`core.application.env.WIRIS_PATH`
WIRIS Connection Settings > Wiris Port	`core.application.env.WIRIS_PORT`
WIRIS Connection Settings > Wiris Protocol	`core.application.env.WIRIS_PROTOCOL`
WIRIS Connection Settings > Use proxy for Wiris	`core.application.env.WIRIS_PROXY_ENABLED`

WIRIS Connection Settings > Wiris Proxy Host	`core.application.env.WIRIS_PROXY_PORT`
WIRIS Connection Settings > Wiris Proxy User	`core.secrets.wirisProxy.username`
WIRIS Connection Settings > Wiris Proxy Password	`core.secrets.wirisProxy.password`

Advanced Startup Settings

KOTS config field	Equivalent Helm chart value
Advanced Startup Settings > Enable JMX remote for core Jama application	`core.application.env.ENABLE_JAVA_JMX_REMOTE`
Advanced Startup Settings > JMX remote port number for core Jama application	`core.application.env.JAVA_JMX_REMOTE_PORT`
Advanced Startup Settings > Additional JVM options for core Jama application	`core.application.env.ADDITIONAL_JAVA_OPTIONS`
Advanced Startup Settings > Enable JMX remote for search service	`core.search.application.env.ENABLE_JAVA_JMX_REMOTE`
Advanced Startup Settings > JMX remote port number for search service	`core.search.application.env.JAVA_JMX_REMOTE_PORT`
Advanced Startup Settings > Additional JVM options for search service	`core.search.application.env.ADDITIONAL_JAVA_OPTIONS`
Advanced Startup Settings > Enable JMX remote for OpenSearch	`core.search.opensearch.application.env.ENABLE_JAVA_JMX_REMOTE`
Advanced Startup Settings > JMX remote port number for OpenSearch	`core.search.opensearch.application.env.JAVA_JMX_REMOTE_PORT`
Advanced Startup Settings > Additional JVM options for OpenSearch	`core.search.opensearch.application.env.ADDITIONAL_JAVA_OPTIONS`
Advanced Startup Settings > Enable JMX remote for diff service	`core.diff.application.env.ENABLE_JAVA_JMX_REMOTE`
Advanced Startup Settings > JMX remote port number for diff service	`core.diff.application.env.JAVA_JMX_REMOTE_PORT`

Advanced Startup Settings > Additional JVM options for diff service	`core.diff.application.env.ADDITIONAL_JAVA_OPTIONS`
Advanced Startup Settings > Additional JVM options for Hazelcast service	`core.hazelcast.application.env.ADDITIONAL_JAVA_OPTIONS`
Advanced Startup Settings > Java RMI server host name	`global.jmx.javaRmiHostName`

Miscellaneous

KOTS config field	Equivalent Helm chart value
Tenant Manager Settings > Enabled?	`core.tenant-manager.job.enabled`
Restore Jama Backup > Backup file	`core.tenant-manager.application.env.RESTORE_BACKUP_FILE`

KOTS only

The following KOTS config fields do not have a **direct** equivalent in Helm values.

- Kubernetes Configuration > Allow Master Nodes
- Kubernetes Configuration > Issuer Name
- Kubernetes Configuration > Is this a cluster scoped Issuer?
- Host Name > TLS Key Pair Source > Cluster Managed
- Host Name > TLS Key Pair Source > Reuse Admin Console TLS Configuration
- SSL Settings > TLSv1.2 TLSv1.3
- Jama Cloud > Authorization token
- Jama Cloud > SAML URL
- ActiveMQ Service Settings > Migration Job > Enable Migration Job
- ActiveMQ Service Settings > Migration Job > Max CPU for Migration Job
- ActiveMQ Service Settings > Migration Job > Max Memory for Migration Job
- ActiveMQ Service Settings > Migration Job > Max Memory for Migration Job Container
- OpenSearch Settings > Include OpenSearch in Replicated Snapshots

- AWS Resources > Enable EFS Storage Class
- AWS EFS Storage Class > Name
- AWS EFS Storage Class > File System Id
- AWS EFS Storage Class > Base Path