Migrating Data from NFS

Overview

Requires an **Enterprise license**

Previous to version 5.0, an Artifactory HA installation stored binaries and configuration files on an NFS mount. This mount was used by the $CLUSTER_HOME folder to synchronize configuration and binary files between the cluster nodes. From version 5.0, you have the option of migrating your binaries to alternative storage which presents the following advantages:

- The filestore can be distributed between the cluster nodes or on a cloud storage provider (S3)
- Limitations of the network (such as file size limits) no longer affect the filestore
- The cluster nodes do not require access to one central location
- Once removed from the NFS, binaries are stored with redundancy in a clustered sharding configuration

This page is designated for users who have upgraded their Artifactory HA installation from version 4.x to version 5.x. During the upgrade process, all configuration files will have been migrated to the database, and will be synchronized and managed there henceforth, however, the data in these installations is still stored on an NFS mount under the *$CLUSTER_HOME/ha-data* folder which leaves you still reliant on the NFS. While you may continue operating in this mode, you also have the option of migrating your data to alternative storage and removing the NFS mount.

> **Migrating data is optional. NFS is still supported.**
>
> While migrating your data from NFS presents the advantages described above, this is optional. Artifactory 5 still supports an HA cluster storing its data on the NFS.

The instructions on this page describe how to move your binary data away from the *$CLUSTER_HOME/ha-data* folder on the NFS mount allowing you to remove the mount altogether. We will cover three main use cases:

Use Case	Initial State	Final State
1	**NFS:** All data is stored on the NFS	**Local FS:** All data is stored on each node's local file system
2	**NFS Eventual + S3:** NFS is used as the Eventual Binary Provider before copying data over to S3 for persistent object store	**Local FS Evenutal + S3:** Each node's local file system is used as the Eventual Binary Provider before copying data over to S3 for persistent object store
3	**NFS:** All data is stored on the NFS	**Local FS Evenutal + S3:** Each node's local file system is used as the Eventual Binary Provider before copying data over to S3 for persistent object store

For all these use cases, once the data has been migrated, you will be able to completely remove the NFS mount.

Page contents

Configuring the Migration

Before migrating your data away from the NFS, make sure all nodes in your HA cluster are up and running. Then, to configure migration of your data for the use cases described above, follow the procedure below:

1. Verify versions
2. Verify configuration files are synchronized
3. Edit the *ha-node.properties* file
4. Copy data to the new location
5. Configure binarystore.xml to match your setup
6. Test the configuration

Verifying Versions

Before proceeding with transferring your data, you need to verify that all cluster nodes are installed with exactly the same version which must be 5.0 and above. To verify the version running on each node in your HA cluster, in the **Admin** module under **Configuration | High Availability,** check the **Version** column of the table displaying your HA nodes.

Verify Configuration Files are Synchronized

When upgrading your HA cluster from version 4.x to version 5.x, an automatic conversion process synchronizes the configuration files for all the cluster nodes. This replaces the need for the *$CLUSTER_HOME/ha-etc* folder that was used in v4.x. Once you have verified that all nodes are running the same version, you should verify that all configuration files are synchronized between the nodes. For each node, navigate to its *$ARTIFACTORY_HOME/etc* folder and verify the following:

ha-node.properties	Each node should still have this file configured as described in Create ha-node.properties.
db.properties	This file was introduced in Artifactory 5.0 and it defines the connection to the database. The password specified in this file is encrypted by the key in the *master.key* file. It should be identical in each cluster node.
binarystore.xml	This file opens up the full set of options to configure your binary storage without the NFS. It will contain the binary provider configuration according to how you wish to store your binaries. For each of the use cases described above, you can find the corresponding binary provider configuration under Configure binarystore.xml.
master.key	This file contains the key used to encrypt and decrypt files that are used to synchronize the cluster nodes. It should be identical on each cluster node.

ha-node.properties	Each node should still have this file configured as described in Create ha-node.properties
db.properties	This file was introduced in Artifactory 5.0 and it defines the connection to the database. The password specified in this file is encrypted by the key in the *communication.key* file. It should be identical in each cluster node.
binarystore.xml	This file opens up the full set of options to configure your binary storage without the NFS. It will contain the binary provider configuration according to how you wish to store your binaries. For each of the use cases described above, you can find the corresponding binary provider configuration under Configure binarystore.xml.
communication.key	This file contains the key used to encrypt and decrypt files that are used to synchronize the cluster nodes. It should be identical on each cluster node.

From version 5.0, Artifactory HA synchronizes configuration files from the primary to all secondary nodes, a change made to one of these files on the primary triggers the mechanism to synchronize the change to the other nodes.

Sync carefully

Since changes on one node are automatically synchronized to the other nodes, take care not to simultaneously modify the same file on two different nodes since changes you make on one node could overwrite the changes you make on the other one.

Edit the ha-node.properties File

Locate the ha-node.properties file in each node under the *$ARTIFACTORY_HOME/etc* and comment out or remove the following entries otherwise Artifactory will continue write according to the previous path you have configured to the shared file system.

```
artifactory.ha.data.dir=/var/opt/jfrog/artifactory-ha
artifactory.ha.backup.dir=/var/opt/jfrog/artifactory-backup
```

Copy Data to the New Location

Once you have verified your configuration files are correctly synchronized, you are ready to migrate your data. The sub-sections below describe how to migrate your data for the three use-cases described in the Overview above.

Use Case 1: NFS Local FS

For this use case, we first need to ensure that there is enough storage available on each node to accommodate the volume of data in my */data* folder and the desired redundancy. In general, you need to comply with the following formula:

```
Max storage * redundancy < total space available on all nodes
```

For example,

- If you expect the maximum storage in your environment to be **100 TB**
- Your redundancy is **2**
- You have **4 nodes** in your cluster,

Then each node should have at least **50 TB** of storage available.

For a redundancy of N, copy the data from your NFS to N of the nodes in your cluster.

For example, for a redundancy of 2, and assuming you have two nodes named "Node1" and "Node2" respectively, copy the *$CLUSTER_HOME/ha-data* folder to the *$ARTIFACTORY_HOME/data* folder on each of Node1 and Node2.

Optimize distribution of your files

Once you have copied your filestore to to each of the N nodes according to the desired redundancy, we recommend invoking the Optimize System Storage REST API endpoint in order to optimize the storage by balancing it storage amongst all nodes in the cluster.

Use Case 2: NFS Eventual + S3: Local FS Eventual + S3

This use case refers to using S3 as persistent storage, but is equally applicable to other cloud object store providers such as GCS, CEPH, OpenStack and other supported vendors.

In this use case, you only need to ensure that there are **no files in the *eventual* folder of your NFS**. If any files are still there, they should be moved to your cloud storage provider bucket, or to one of the nodes' ***eventual*** folder.

Use Case 3: NFS Local FS Eventual + S3

Migrating a filestore for a single installation to S3 is normally an automatic procedure handled by Artifactory, however, in the case of moving an HA filestore from the NFS, the automatic procedure does not work since the folder structure changes.

In this case, you need to copy the data under *$CLUSTER_HOME/ha-data* from your NFS to the bucket on your cloud storage provider (here too, other providers described in Use Case 2 are also supported) while making sure that there are no files left in the *_queue* or *_pre* folders of the eventual binary provider on your node's local file system.

Configure binarystore.xml

In this step you need to configure the binarystore.xml to match the setup you have selected in the use case. Note that the three use cases above use one of two final configurations:

All data is stored on the cluster node's local filesystem (labelled here as **Local FS**)

The cluster nodes use the cluster node's local filesystem as an eventual binary provider and data is persistently stored on S3 (labelled here as **Local FS Eventual + S3**)

Node downtime required

To modify the binarystore.xml file for a node, you first need to gracefully shut down the node, modify the file and then restart the node in order for your new configuration to take effect

Local FS

In this example, all data is stored on the nodes' file systems. For the sake of this example, we will assume that:

- We have 3 nodes
- We want redundancy = 1

To accomplish this setup, you need to:

- Copy the data from the *$CLUSTER_HOME/ha-data* on your NFS to the *$ARTIFACTORY_HOME/data* folder on two of the nodes.
- Once all data has been copied, you need to place the binarystore.xml under *$ARTIFACTORY_HOME/etc* of each cluster node.
- Finally, you need to gracefully restart each node for the changes to take effect.

Optimizing the redundant storage

After restarting your system, you can trigger optimization using the REST API so that all three nodes are utilized for redundancy. For details, please refer to Optimize System Storage.

Example

In this use case, the binarystore.xml used with the NFS before migration would look like the following if you are using one of the default file-system template.

```
<config version="1">
    <chain template="file-system"/>
</config>
```

After migrating the data, the new binarystore.xml placed on each cluster node you can use the cluster-file-system template.

```
<config version="2">
        <chain template="cluster-file-system"/>
</config>
```

While you don't need to configure anything else, this is what the *cluster-file-system* template looks like:

Redundancy leniency

We recommend adding the lenientLimit parameter to the below configuration under the *sharding-cluster* provider configuration:

```
<lenientLimit>1</lenientLimit>
```

Without this parameter, Artifactory won't accept artifact deployments while the number of live nodes in your cluster is lower than the specified redundancy.

```
<config version="2">
        <chain> <!--template="cluster-file-system"-->
        <provider id="cache-fs" type="cache-fs">
            <provider id="sharding-cluster" type="sharding-cluster">
                <sub-provider id="state-aware" type="state-aware"/>
                <dynamic-provider id="remote-fs" type="remote"/>
            </provider>
        </provider>
    </chain>

        <provider id="state-aware" type="state-aware">
        <zone>local</zone>
    </provider>

    <!-- Shard dynamic remote provider configuration -->
    <provider id="remote-fs" type="remote">
        <zone>remote</zone>
    </provider>

    <provider id="sharding-cluster" type="sharding-cluster">
        <readBehavior>crossNetworkStrategy</readBehavior>
        <writeBehavior>crossNetworkStrategy</writeBehavior>
        <redundancy>2</redundancy>
        <property name="zones" value="local,remote"/>
    </provider>

</config>
```

Local FS Eventual + S3

In this example, data is temporarily stored on the file system of each node using an Eventual binary provider, and is then passed on to your S3 object storage for persistent storage.

In this use case, the binarystore.xml used your NFS for cache and eventual with your object store on S3 before migration will look like the following if you are using the S3 template.

```
<config version="2">
        <chain template="s3"/>
</config>
```

After migrating your filestore to S3 (and stopping to use the NFS), your *binarystore.xml* should use the `cluster-s3` template as follows:

```
<config version="2">
        <chain template="cluster-s3"/>
</config>
```

The `cluster-s3` template looks like this:

Redundancy leniency

We recommend adding the lenientLimit parameter to the below configuration under the *sharding-cluster* provider configuration:

```
<lenientLimit>1</lenientLimit>
```

Without this parameter, Artifactory won't accept artifact deployments while the number of live nodes in your cluster is lower than the specified redundancy.

```
<config version="2">
        <chain> <!--template="cluster-s3"-->
            <provider id="cache-fs-eventual-s3" type="cache-fs">
                <provider id="sharding-cluster-eventual-s3" type="sharding-cluster">
                    <sub-provider id="eventual-cluster-s3" type="eventual-cluster">
                        <provider id="retry-s3" type="retry">
                            <provider id="s3" type="s3"/>
                        </provider>
                    </sub-provider>
                    <dynamic-provider id="remote-s3" type="remote"/>
                </provider>
            </provider>
        </chain>

        <provider id="sharding-cluster-eventual-s3" type="sharding-cluster">
            <readBehavior>crossNetworkStrategy</readBehavior>
            <writeBehavior>crossNetworkStrategy</writeBehavior>
            <redundancy>2</redundancy>
            <property name="zones" value="local,remote"/>
        </provider>

        <provider id="remote-s3" type="remote">
            <zone>remote</zone>
        </provider>

        <provider id="eventual-cluster-s3" type="eventual-cluster">
            <zone>local</zone>
        </provider>
        <provider id="s3" type="s3">
       <endpoint>http://s3.amazonaws.com</endpoint>
       <identity>[ENTER IDENTITY HERE]</identity>
       <credential>[ENTER CREDENTIALS HERE]</credential>
       <path>[ENTER PATH HERE]</path>
       <bucketName>[ENTER BUCKET NAME HERE]</bucketName>
    </provider>
</config>
```

Because you must configure the s3 provider with parameters specific to your account (but can leave all others with the recommended values), if you choose to use this template, your *binarystore.xml* configuration file should look like this:

```
<config version="2">

    <chain template="cluster-s3"/>

        <provider id="s3" type="s3">
       <endpoint>http://s3.amazonaws.com</endpoint>
       <identity>[ENTER IDENTITY HERE]</identity>
       <credential>[ENTER CREDENTIALS HERE]</credential>
       <path>[ENTER PATH HERE]</path>
       <bucketName>[ENTER BUCKET NAME HERE]</bucketName>
    </provider>

</config>
```

Testing Your Configuration

To test your configuration you can simply deploy an artifact to Artifactory and then inspect your persistent storage (whether on your node's file system on your cloud provider) and verify that the artifact has been stored correctly.