



BEST PRACTICES FOR **ARTIFACTORY BACKUPS** AND **DISASTER RECOVERY**

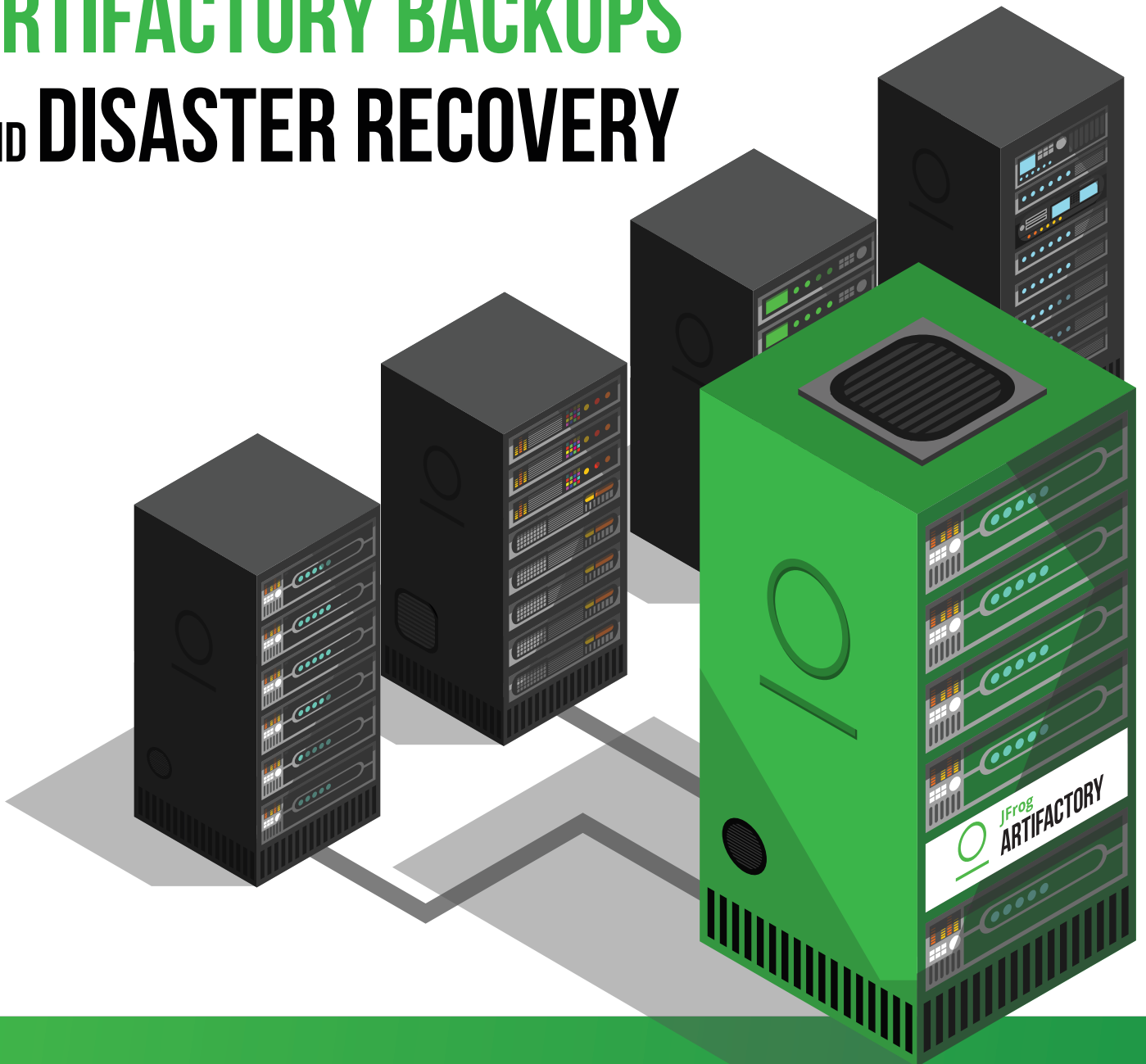


TABLE OF CONTENT

INTRODUCTION	3
How Artifactory stores your binaries and what's so special about it?	3
1. BACKING UP SMALL ARTIFACTORY INSTANCES	4
2. BACKING UP LARGE ARTIFACTORY INSTANCES	6
Periodic Database Snapshots	6
ADDITIONAL WAYS TO KEEP YOUR BINARIES SAFE	7
Filestore Sharding	7
Disaster Recovery (DR)	7
Manually setting up DR	7
DR with JFrog Mission Control	8
Moving data in a way that won't clog your network	8
Configuring Disaster Recovery	8
Initializing DR	9
Synchronizing Repositories	9
Simple migration with downtime	9
CONCLUSION	11

INTRODUCTION

Right at the heart of the DevOps pipeline, JFrog Artifactory is the central hub for all of your binary needs. In production, every minute is valuable. Whether it's to deploy your latest packages or to cache openly available packages, it is vital that you have all of your binaries available at all times. The challenge is that there is no such thing as an indestructible computer or a flawless piece of software, and this is why we must make sure to have a backup plan, literally.

This white paper describes several methods for tackling these concerns, in hopes that one will work best for your organization.

HOW ARTIFACTORY STORES YOUR BINARIES AND WHAT'S SO SPECIAL ABOUT IT?

The classic way to protect your binaries is by using recurring backups of your files, and having them available for use in case anything goes down. Artifactory has specific ways to backup your binaries so that you may import them back into a new instance and keep all your references. As described in the following section, the way Artifactory stores your binaries is a bit different than your usual storage, so that has to be taken into consideration for these tasks.

Artifactory stores both binaries and their metadata. The metadata is stored in a Derby database (by default), and includes information such as the checksum, repository, path, created time, and so on. The actual binaries are, however, stored separately. Depending on how you [configure your filestore](#), the files will be stored in one or multiple locations, using their [SHA1 checksum value](#) as the file name and the first two characters of the SHA1 value as the folder name. For example, with a default Artifactory installation you'll find the following structure in the `$ArtifactoryHome/data/filestore`.



Artifactory offers a deduplication feature that will save you countless GBs or even TBs of space, using [checksum based storage](#).

This is why it's important to backup your filestore, as well as the database or metadata of these files. Depending on **the size of your instance** there are different approaches.

The following sections will describe the **different backup approaches** and ways to keep your binaries safe.

DEDUPLICATION

By referencing binaries by their checksum, pretty much like Git or Dropbox do, and not relying on filesystem paths same-content files are never stored more than once. This is one of the few ways you can optimize the storage of binaries.

CHECKSUM-BASED STORAGE

Artifactory was built from the ground up for optimal management of binaries with the capability to support any package format that emerged in the software development domain. One of the key features enabling these characteristics is Checksum-Based Storage. [Learn More >>](#)

1. BACKING UP SMALL ARTIFACTORY INSTANCES

System backups are a simple, built-in way of backing up your Artifactory instances, directly [configured from within the Artifactory UI](#).

User administrators can set daily, weekly, and even manual periodic backups using [cron expressions](#).

Once a system backup is created, it will be located in the `$ArtifactoryHome/backup/<backup_key>` directory, with a timestamp.

[The Artifactory System Import](#) can then be used to recreate an instance entirely if there is a failure.

The screenshot shows the 'Edit backup-daily Backup' configuration page in the Artifactory UI. It is divided into two main sections: 'Backup Settings' and 'Advanced'.

Backup Settings:

- Enabled:** A checkbox that is checked.
- Backup Key:** A text field containing 'backup-daily'.
- Cron Expression:** A text field containing '0 0 2 * MON-FRI'.
- Next Backup Time:** A text field showing 'Wed Oct 19 02:00:00 UTC 2016'.
- Server Path For Backup:** A text field with a 'Browse' button next to it.

Advanced:

- Send Mail to Admins if there are Backup Errors:** A checked checkbox.
- Exclude Builds:** An unchecked checkbox.
- Exclude New Repositories:** An unchecked checkbox.
- Verify enough disk space is available for backup:** An unchecked checkbox.
- Incremental:** A checked checkbox.
- Retention Period Hours:** A text field containing '0'.

At the bottom, there are two lists of repositories:

- Excluded Repositories:** A list containing 'repo1', 'debian-flat', 'debian-local', 'dockerv1', 'ext-release-local', 'ext-snapshot-local', and 'gem1'.
- Included Repositories:** A list containing 'dima', 'docker-dev-local2', 'docker-prod-local2', 'docker-hub-proxy', 'ifs-remote', 'p2-remote', and 'testtttt'.

Navigation arrows (back, forward, and search) are located between the two lists. At the very bottom, there is a checkbox labeled 'Back up to a Zip Archive (Slow and CPU intensive)'.

Additional advanced backup options include:

- **Incremental backups** - only backing up what the previous backup missed, saving time
- **Include/Exclude** specific repositories
- **Retention period** - time period to keep backups, if they are not incremental
- **Verify disc space** - check that there is enough space before performing backup
- **Exclude builds and new repositories** from backup

This type of Artifactory backup will create a new set of repository folders, that contain each artifact stored alongside with its metadata. This complete duplicate of data can take a toll on your storage if you are backing up large instances. You can mitigate this storage cost by backing up your filestore separately and performing a skeleton export of the database ([system export via REST API](#) using the Exclude Content = true option).

Ultimately, it is recommended to switch to a different backup method if your instance reaches 500GB-1TB of storage, or if you go over 1 Million Artifacts in your instance.

2. BACKING UP LARGE ARTIFACTORY INSTANCES

For instances with a large set of data, alternative routes are suggested. This is because large backups can take a significant amount of time to complete, which may even overlap your cron duration and cause missed backup intervals. The purpose of a backup is to make data available even in case of hardware failure, or perhaps get it ready for migration to a different version or instance. Spending too much time on backups is counterproductive, especially when you really need the backup!

PERIODIC DATABASE SNAPSHOTS

It's important to backup your filestore, as well as the database. Taking periodic snapshots of your external database will complete your coverage. Without the database backup, the filestore on its own is just a folder with files that are named after their checksum, making them impossible to identify in a timely manner (plus, you'd have to rename them all). When it's time to restore, you'll need to use the latest available snapshot. Copying the filestore and taking periodic snapshots of your external database should be done around the same time to avoid references of non-existent binaries or other empty values. However, taking snapshots of the external database should be done first before copying the filestore.

Completing a full system export with the content excluded is also a good way to backup data. This is the equivalent of a DB dump, where a collection of xml files that represent your binaries and repository locations are exported. It is similar to a system backup but without the binaries.

ADDITIONAL WAYS TO KEEP YOUR BINARIES SAFE

There are additional methods that can help you avoid losing data, as well as any downtime before an instance is recovered. These include, **redundancy** (storing multiple copies in different locations) and **disaster recovery** (restoring an instance when necessary).

FILESTORE SHARDING

Artifactory offers a [Sharding Binary Provider](#) that lets you manage your binaries in a sharded filestore. A sharded filestore is one that is implemented on a number of physical mounts (M), which store binary objects with redundancy (R), where $R \leq M$. This binary provider is not independent and will always be used as part of a more complex template chain of providers.

Sharding the filestores offers reasonable scalability, however be cautious of creating too many shards as additional shards do cause a performance impact (we generally don't recommend exceeding 10 shards at this time, although this may change in the future). The difference is that the process is initially more manual, which means that when the underlying storage approaches depletion, an additional mount will need to be added. The system will then invoke balancing mechanisms to regenerate the filestore redundancy according to the configuration parameters.

DISASTER RECOVERY (DR)

DR provides you with a solution to **easily recover from any event** that may cause irreversible damage and loss of data, as well as a graceful solution that enables taking Artifactory down for any other reason such as hardware maintenance on the server machine.

MANUALLY SETTING UP DR

Disaster recovery can be manually set up, however this would be time consuming and complex. Under the hood, this would include:

- **Matching up local repositories** on your master instance with corresponding repositories on the target instance.
- **Setting up all replication relationships** to move your critical data from the Master to the Data.

Keeping track of the millions of artifacts and making sure their metadata is correctly replicated will take up a considerable amount of time. There can't be any errors here as in the event that Disaster Recovery needs to kick in, no time can be wasted on "empty" artifacts.



DR WITH JFROG MISSION CONTROL

Disaster recovery is designed to continue providing service as quickly as possible if a whole Artifactory installation goes down (for example, all servers in an HA installation go down due to an electrical malfunction in the site). In this case, requests can be rerouted to the Target instance, and, while this is not automatic, it is achieved with a single click of a button in Mission Control.

Don't confuse setting up Artifactory in a High Availability configuration with setting up Disaster Recovery. A high availability configuration uses two or more redundant Artifactory servers to ensure users continue to get service if one or more of the Artifactory servers goes down (whether due to some hardware fault, maintenance, or any other reason) as long as at least one of the servers is operational. Once HA is set up, service continues automatically with no intervention required by the administrator.

MOVING DATA IN A WAY THAT WON'T CLOG YOUR NETWORK

Synchronizing network is resource intensive. For this reason, Mission Control does not move data all at once, but rather individually, with time intervals.

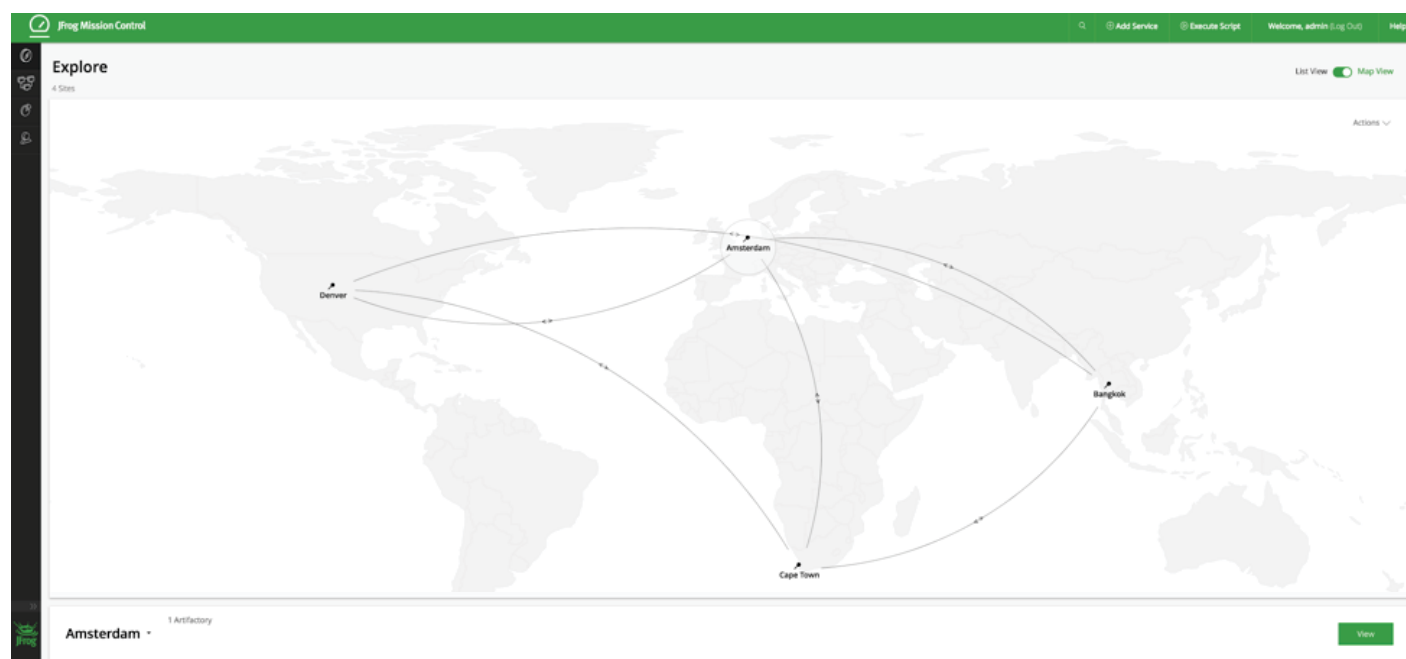
CONFIGURING DISASTER RECOVERY

JFrog Mission Control lets you configure [complete system replication](#) between **Master instances** and corresponding **Target instances**. The Master instance in this case is your production instance, and the Target instance will work as a replication target for DR.

The Master and Target pairs you configure are displayed in the Manage module under DR Configuration.

>> [Learn more about replication](#) and using Artifactory to manage binaries across multi-site topologies.

The following illustrates a Full Mesh Topology configuration in Mission Control:



To avoid lengthy and resource intensive synchronization the relevant department in your organization may manually sync between Master and Target instances outside of both Mission Control and Artifactory before you initialize the DR process. This is called an external sync.

INITIALIZING DR

During what we call the Init step, Mission Control establishes the replication relationships between all local repositories on the Master instance and the corresponding repositories on the Target instance as well as backing up security settings and various configuration files from the Master instance to Mission Control. These are later pushed to the Target instance, though no data transfer occurs at this step, it will instead happen on the Synchronize step.

SYNCHRONIZING REPOSITORIES

During the synchronize step, Mission Control begins invoking replication from the Master instance to the Target instance so that each local repository on the Target instance is synchronized with the corresponding repository on the Master instance.

Now you're protected!

Once all repositories on your Target instance are synchronized with your Master instance, your system is DR protected. This means you can instantly invoke failover from your Master to your Target instance so that your users may transparently continue to get service from Artifactory.

SIMPLE MIGRATION WITH DOWNTIME

In some cases you will simply want to set up a copy of your instance, which can be done using backups. This method can also help you keep a “manual” DR instance that you periodically clone to or replicate to from your active cluster.

The benefits of executing a migration is that it's one of the simplest types of upgrades or instance cloning, since it only entails setting up a new instance to migrate data into, and requires no data in the new instance.





There are two methods that offer little to no downtime during this migration. The first method has a short downtime and requires the following steps:

1. Disable **Admin -> Advanced -> Maintenance -> Garbage collection** on both servers
2. Old server: Copy the **\$ARTIFACTORY_HOME/data/filestore** folder to the new server's filestore folder
3. Old server: Take server off the network to block new requests
4. Old server: Perform full system export with the "**Exclude Content**" option selected (no other options selected)
5. Old server: Shut down **<downtime step>**
6. Old Server: rsync from **\$ARTIFACTORY_HOME/data/filestore** folder to the new server's filestore folder one last time
7. New server: Perform full system import (Do **NOT** select the Exclude Content option).
8. New server: Turn on network traffic / switch DNS to new server.
9. New Server: Enable Garbage Collection again

The second method is more complicated than the first, but has almost no downtime. It requires the following steps:

1. Disable **Admin -> Advanced -> Maintenance -> Garbage Collection** on both servers
2. Old server: Copy the **\$ARTIFACTORY_HOME/data/filestore** folder to the new server's filestore folder
3. Old server: Perform full system export with the "Exclude Content" option selected (no other options selected)
4. New server: Perform full system import (Do NOT select the Exclude Content option).
5. Old Server: Set up all local repositories to replicate to the repositories on the new server with the "sync deletes" option turned off.
6. New server: Turn on network traffic / switch DNS to new server.
7. Old server: Execute all replication jobs
8. Old server: Shut down
9. New Server: Enable Garbage Collection again

Ultimately, the migration method you choose will depend on your preference, tolerance for downtime and even industry. For example, financial industries tend to lean towards filestore sharding for security purposes. The main difference between the two methods is the replication part that will allow you to move over any artifacts that were deployed during the export/import process.

CONCLUSION

As described in this whitepaper, there are multiple ways to protect your binaries. Depending on your setup, industry requirements, repository size and backup frequency (small incremental or disaster recovery), you can choose the right fit for your organization.

All of the methods described have a common goal in mind: minimize downtime in case of an unexpected event that can impact development and release time. As well as maximize developer productivity.