



CA Release Automation

With

JFrog Artifactory High Availability

Best Practices Guide

Author : Walter Guerrero – walter.guerrero@ca.com

Version: 2.5

Filename: CA-Release-Automation-Artifactory-HA-Best-Practices-Guide.docx

Date: 1/26/2015

Table of Contents

Disclaimer.....	3
Overview	4
Release Automation High Availability.....	4
Artifactory HA Architecture	5
Install JFrog Artifactory HA Software	6
Manual Installation	7
RPM Package Installation.....	8
Post-Installation Configuration	9
Setup NFS mounting point.....	10
NFS Server settings	10
NFS client settings.....	11
Configure Artifactory HA Cluster	11
Cluster Configuration	11
Database Configuration	13
Oracle	13
MS SQL Server	15
Artifactory HA (Demo) License.....	16
NGINX Proxy Load Balancer	16
HTTP Support	17
HTTPS Support	17
Create SSL Certificate.....	18
NGINX.CONF Entries	18
Additional Changes in Artifactory	19
Starting JFrog Artifactory	20
Install and Configure Release Automation	21
Updates to nolio-repo.properties	22
Import Certificate into "Nolio.jks"	23
Artifactory Repositories	24
Starting up all components.....	25
Useful Links	26

Disclaimer

It is recommended that the steps described in this best practice guide need to be conducted in a test system prior to attempting these steps in a production systems (JFrog Artifactory, CA Release Automation, Database instance, and NGINX).

Overview

This best practices guide is designed to provide you with *guidance and advice* in how to make the JFrog Artifactory High Availability cluster the default software package repository for CA Release Automation starting with release 5.x onwards.

It is important to note that the shipped version of Nexus or the available Nexus software package repository only supports an “Active/Passive” cluster (at the time of this writing). Given this limitation with the available cluster services provided by Nexus.

This document focuses on the “Active/Active” cluster provided by JFrog Artifactory, and that will be the concentration of this best practices document.

Please be advised to use the JFrog Artifactory High Availability cluster, you will need to obtain an “Enterprise License” from JFrog to activate this feature set (power pack option) in Artifactory.

Release Automation High Availability

The objective of this best practices document is to make the complete implementation of Release Automation high availability, which is composed of clusters services for the software package artifactory repository, databases, and the Release Automation components. Below is a typical high availability implementation.

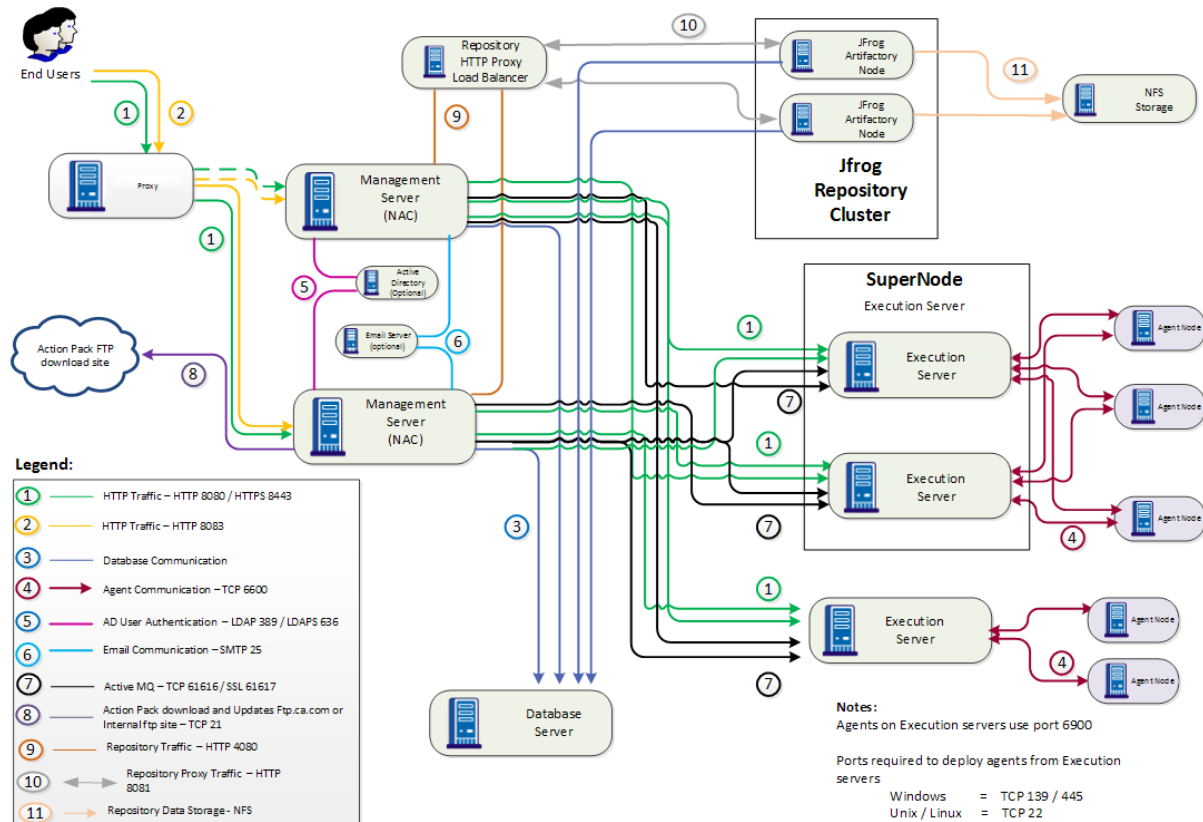


Figure 1: Typical high availability implementation

As you can see in the typical architecture above, not only is the Release Automation implementation a high availability implementation, but the JFrog Artifactory cluster is also a high availability implementation as well.

Artifactory HA Architecture

The JFrog Artifactory HA (Active/Active) is composed of two or more Artifactory servers that share a common database and Network File System (NFS), and these Artifactory servers have a HTTP/HTTPS load balancer as its working front end. As described in Figure 2.

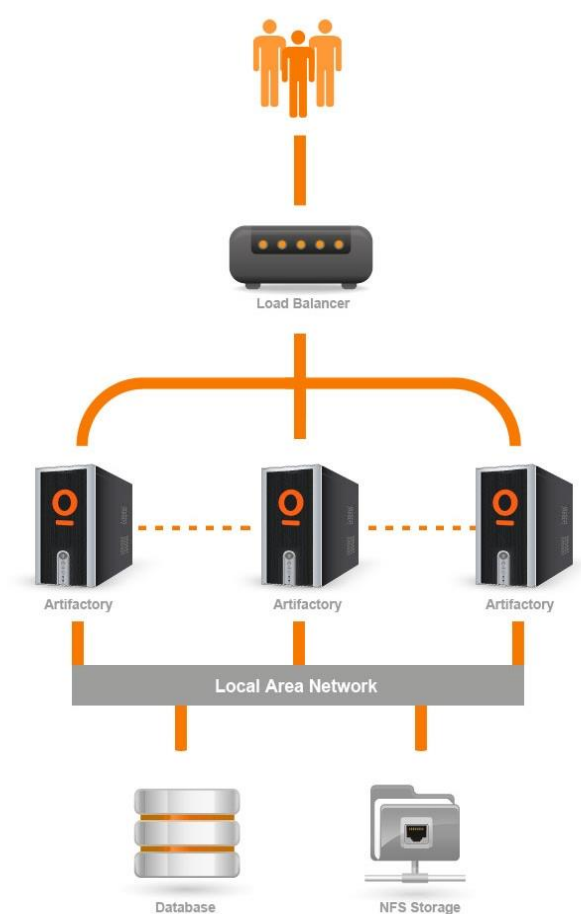


Figure 2: Typical Artifactory HA Topology

For optimum performance is recommended that all the components of the Artifactory HA cluster be connected via a fast internal LAN to support high system performance, as well as to allow for fast synchronization between the Artifactory HA nodes. It is very important that one of the Artifactory nodes be configured as the “master” node; the responsibilities of this

“master” node include the execution of cluster-wide tasks, such as the cleaning up of unreferenced binaries.

In this document, we are going to go over the different components that make up the JFrog Artifactory HA cluster and their installation and configuration; as well as its integration with CA Release Automation to become the default repository to be used by CA Release Automation.

This particular implementation of the JFrog Artifactory cluster is for an **Active/Active** cluster.

The implementation of the Artifactory Active/Active cluster is composed of the following sections.

- Install JFrog Artifactory HA in the cluster nodes
- Setup NFS mounting point
- Configure JFrog Artifactory HA cluster nodes
- Install and configure NGINX proxy load balancer¹
- Install and configure or update Release Automation

Support for high availability cluster (Active/Active) started with release 3.1 of JFrog Artifactory; Artifactory provides you with redundant network architecture. The Artifactory repository will continue to operate as long as one of the Artifactory HA clusters nodes is running.

Install JFrog Artifactory HA Software

This section of this guide describes how to obtain the Artifactory Pro package (which includes a demo license request). If you already have Artifactory Pro with the HA feature turn on and the cluster configure, please go to the “Install and Configure Release Automation” section in this document for the integration steps to follow.

You will need to make a request for the Artifactory HA package, along with the necessary demo license string by following the link below:

http://www.jfrog.com/home/v_artifactorypro_high_availability

This document was prepared utilizing Linux as the platform of choice, but you can also install all these packages utilizing Microsoft Windows Server 2008² 64-bit and above.

¹ This best practice guide was created utilizing NGINX load balancer, as it is the http load balancer used by large enterprises, especially by purchasing the NGINX+ commercial version with 24/7 customer support.

² Please follow the Windows installation instructions as listed in the JFrog documentation site [JFrog Artifactory Windows Installation](#)

Once you have received the link for the Artifactory installation package, you can download either the stand-alone or RPM package..

- artifactory-powerpack-standalone-3.x.x.zip³
- artifactory-powerpack-rpm-3.x.x.rpm -- exclusive for Linux installations

The following pre-installation instructions are focused on Linux installation. Please make sure that these steps are followed.

- Login to the designated JFrog Artifactory HA cluster nodes and NFS server as the root user or using the “*sudo* –”to become the root user as you are authorized.
- Create an Artifactory group by using the following command.
 - “*groupadd -g 550 artifactory*”
- Create the following user: artifactory by using the following command.
 - “*useradd -u 505-g 550 artifactory*”.
- Make sure that the system user ID and group ID are identical in all systems involved.
 - “*cat /etc/passwd*”
 - “*artifactory:x:505:550::/home/artifactory:/bin/bash*”
 - “*cat /etc/group*”
 - “*artifactory:x:550:*”
- Add a default password to the “*artifactory*” user, use the command, and follow the prompts.
 - “*passwd artifactory*”

We are going to provide with two different installation methods here, you can use the manual installation or the RPM package installation.

Manual Installation

Follow the instructions below to complete the manual installation of the JFrog Artifactory.

- Change directory to the location where the Artifactory power-pack zip file has been downloaded.
- Run the following command “*unzip artifactory-powerpack-standalone-3.x.x.zip*”
- Move the newly extracted directory: “*mv artifactory-powerpack-3.x.x /opt/artifactory*”
- Run the following script to install the Artifactory service:
\$ARTIFACTORY_HOME/bin/installService.sh
- The artifactory service should be installed as shown below

³ Please utilize the latest version of JFrog’s Artifactory, which at the writing of this document that was version 3.2.2.

```

[root@guewa01-U116126 opt]# ls
artifactory CA CertDelivery rh
[root@guewa01-U116126 opt]# ls -l
total 16
drwxr-xr-x. 8 root      root      4096 Jun 21 19:33 artifactory
drwxr-xr-x. 4 root      root      4096 May 2 15:44 CA
drwx----- 2 4294967294 4294967294 4096 Jan 17 2014 CertDelivery
drwxr-xr-x. 2 root      root      4096 May 17 2013 rh
[root@guewa01-U116126 opt]# cd artifactory
[root@guewa01-U116126 artifactory]# cd bin
[root@guewa01-U116126 bin]# ls
artifactory.bat  artifactory.default  artifactory.sh  installService.bat  recover.backup.sh  uninstallService.sh
artifactoryctl  artifactory-service.exe  configure.mysql.sh  installService.sh  uninstallService.bat
[root@guewa01-U116126 bin]# ./installService.sh

Installing artifactory as a Unix service that will run as user artifactory
Installing artifactory with home /opt/artifactory
Creating user artifactory...creating... DONE

Checking configuration link and files in /etc/opt/jfrog/artifactory...
Moving configuration dir /opt/artifactory/etc /opt/artifactory/etc.original...creating the link and updating dir... DONE
Creating environment file /etc/opt/jfrog/artifactory/default...creating... DONE
** INFO: Please edit the files in /etc/opt/jfrog/artifactory to set the correct environment
Especially /etc/opt/jfrog/artifactory/default that defines ARTIFACTORY_HOME, JAVA_HOME and JAVA_OPTIONS

Initializing artifactory service with chkconfig...artifactory      0:off  1:off  2:on   3:on   4:on   5:on   6:off
DONE

Setting file permissions... DONE

***** SUCCESS *****
Installation of Artifactory completed

Please check /etc/opt/jfrog/artifactory, /opt/artifactory/tomcat and /opt/artifactory folders
Please check /etc/init.d/artifactory startup script

you can now check installation by running:
> service artifactory check (or /etc/init.d/artifactory check)

Then activate artifactory with:
> service artifactory start (or /etc/init.d/artifactory start)

[root@guewa01-U116126 bin]# █

```

Figure 3: Results of running Artifactory's installService.sh

RPM Package Installation

If you desired to utilize the RPM package installation method, it is important to note the default location of all the resulting files.

File/Folder	Location	Ownership
Artifactory home	/var/opt/jfrog/artifactory	Artifactory
Artifactory etc	/etc/opt/jfrog/artifactory	Artifactory
Artifactory logs	/var/opt/jfrog/artifactory/logs	Artifactory
Artifactory env variables	/etc/opt/jfrog/artifactory/default	Artifactory
Tomcat home	/opt/jfrog/artifactory/tomcat	Artifactory (root for sub dirs)
Artifactory startup script	/etc/init.d/artifactory	Root
Artifactory binary	/opt/jfrog/artifactory	Root

Run the following command to install JFrog Artifactory RPM package

- Change to the directory where you have downloaded the Artifactory RPM package
 - “rpm -ivh ./artifactory-powerpack-rpm-3.x.x.rpm”


```
[root@guewa01-U117836 Downloads]# rpm -ivh ./artifactory-powerpack-rpm-3.2.2.rpm
Preparing... ##### [100%]
Checking if group artifactory exists...
Group artifactory exists.
Checking if user artifactory exists...
User artifactory exists.
Removing tomcat work directory
  1:artifactory ##### [100%]
Adding the artifactory service to auto-start

The installation of Artifactory has completed successfully.

PLEASE NOTE: You can recover a backup done with Artifactory RPM 3.0 and above using
'/opt/jfrog/artifactory/bin/recover.backup.sh'. For upgrading from previous
version of Artifactory please refer to the wiki http://wiki.jfrog.org/confluence/display/RTF/Upgrading+Artifactory
PLEASE NOTE: It is highly recommended to use Artifactory in conjunction with MySQL.
You can easily configure this setup using '/opt/jfrog/artifactory/bin/configure.mysql.sh'.
```

Figure 4: Results of Artifactory RPM Package Installation

The RPM package installation sets up all the necessary entries, including the startup script, the necessary parameter files.

Post-Installation Configuration

After the successful installation via the manual or RPM method, you will need to perform the following steps:

- Update the /etc/opt/jfrog/artifactory/default file
 - Add the following line: “export CLUSTER_HOME=\$ARTIFACTORY_HOME/cluster”
- Create directory “mkdir \$ARTIFACTORY_HOME/cluster”
- “chmod -R 775 \$ARTIFACTORY_HOME/cluster”
- Change the \$JAVA_HOME to the following: *export JAVA_HOME=/usr/java/jdk1.7.0_67*

```
[root@guewa01-U117836 artifactory]# cat default
#!/bin/sh

#Default values
export ARTIFACTORY_HOME=/var/opt/jfrog/artifactory
export ARTIFACTORY_USER=artifactory
export JAVA_HOME=/usr/java/jdk1.7.0_67
export CLUSTER_HOME=$ARTIFACTORY_HOME/cluster

export TOMCAT_HOME=/opt/jfrog/artifactory/tomcat
export ARTIFACTORY_PID=/var/opt/jfrog/run/artifactory.pid

export JAVA_OPTIONS="-server -Xms512m -Xmx2g -Xss256k -XX:PermSize=128m -XX:MaxPermSize=256m -XX:+UseG1GC"
[root@guewa01-U117836 artifactory]#
```

Figure 5: Contents of the Artifactory Default file

It is recommended that you install the Oracle HotSpot 1.7 64-bit JVM.

Please make sure that the \$JAVA_OPTIONS remain at those minimum levels, you could change those values to improve the performance of the Artifactory service, but these settings should be good for most installations.

Figure 5 shows the typical default values, what is being displayed are values based off the RPM package installation.

Setup NFS mounting point

The next component that we are going to be setting up is the shared NFS (Network File System) file system. This component is mandatory for the JFrog Artifactory HA cluster to support concurrent requests and file locking.

Starting with NFS v4 introduces “state”, where an NFS client notifies the NFS server of its intentions on a file. These intentions can be: locking, reading, writing, and other states. In addition to help keep the file “state” consistent, more sophisticated client and server reboot recovery mechanisms are built into the NFS v4 protocol. In NFS v4 locking is “lease-based”, where the NFS v4 client must maintain contact with the NFS v4 server to continue extending its open and lock leases.

The following conditions need to be met prior to making the NFS file system available to the JFrog Artifactory HA cluster nodes.

- Only NFS versions 3 and 4 are supported, we are recommending that you use **Version 4**.
- The NFS client-side needs to have the following entry: *lookupcache=none*
- Change the ownership and group of the “cluster” directory in the NFS server
 - “chgown -R artifactory /opt/artifactory/cluster”
 - “chgrp -R artifactory /opt/artifactory/cluster”

NFS Server settings

The following NFS settings are recommended in the NFS server:

- /etc/idmapd.conf

```
[Mapping]
Nobody-User = nobody
Nobody-Group = nobody
[Translation]
Method = nsswitch
```

- /etc/exports

```
/opt/artifactory/cluster <node1_ip>(rw, sync, root_squash)
```

```
/opt/artifactory/cluster <node2_ip>(rw, sync, root_squash)
```

NFS client settings

At each of the Artifactory HA's nodes.

- /etc/fstab

```
<nfs_srv_ip>:/opt/artifactory/cluster /var/opt/jfrog/artifactory/cluster nfs  
defaults,lookupcache=none 0 0
```

- /etc/mtab

```
<nfs_srv_ip>:/opt/artifactory/cluster /var/opt/jfrog/artifactory/cluster nfs  
rw,lookupcache=none,vers=4,add=<nfs_srv_ip>,clientaddr=<node_x_ip> 0 0
```

It is important that the access to this NFS is tested and verified from each Artifactory HA node.

Configure Artifactory HA Cluster

The configuration of the Artifactory HA cluster will be broken down into two sections:

- Cluster configuration
- Database configuration

Cluster Configuration

It is important that in the \$CLUSTER_HOME, which for our example this is going to be “/var/opt/jfrog/artifactory/cluster”, the following directories need to be created.

- “ha-etc”
- “ha-data”
- “ha-backup”

Below is a comparison between the \$ARTIFACTORY_HOME and \$CLUSTER_HOME directories and their contents.

- \$ARTIFACTORY_HOME	- \$CLUSTER_HOME
- etc/	- ha-etc/
- ha-node.properties	- cluster.properties
- logback.xml	- storage.properties
- artifactory.lic	- artifactory.system.properties
- data/	- mimetypes.xml
- tmp/	- UI/
- artifactory.properties	- plugins/
- logs/	- ha-data/
- bin/	- filestore/
- misc/	- tmp/
- webapps/	- artifactory.properties
- tomcat/	- ha-backup/
- lib/	
- <jdbc driver>	

Figure 6: Comparison between \$ARTIFACTORY_HOME and \$CLUSTER_HOME

The following empty files need to be created in the \$CLUSTER_HOME

File	Purpose
ha-etc/cluster.properties	Configuration parameters shared by all the cluster nodes.
ha-etc/storage.properties	Identical to the standard “storage.properties”. This file replaces the “etc/storage.properties”, since all the cluster nodes will be using this new storage.properties file.

The typical contents of these files are (we are going to cover the storage.properties in more detail in the database configuration section).

- “cluster.properties” – this file contains an ASCII string that you have selected, this string is used to send secured messages between the nodes.

security.token=thisisjustanexample

The security token value can be an encrypted value, if so desired.

The following file “\$ARTIFACTORY_HOME/etc/ha-node.properties” needs to be created. This file needs to be owned by the artifactory user, be in the artifactory group, and have the following permissions 664.

This particular file will contain the following entries:

Property	Purpose
node.id	Unique descriptive name of the HA cluster node.

cluster.home	Location of the \$CLUSTER_HOME.
context.url	The context URL what will be used to communicate with this HA cluster node installation.
membership.port	The port that will be used by the cluster nodes to communicate. If no port value is provided, Artifactory will allocate a port automatically.
primary	[true false] This property indicates if this is the primary server. There has to be one (and only one) cluster node configure as the primary server.

Here is an example of the “ha-node.properties” contents:

```
node.id=nolio1
cluster.home=/var/opt/jfrog/artifactory/cluster
context.url=http://<node1_ip>:8081/artifactory
membership.port=10001
primary=true (for the primary node this will be set to true, for all the other nodes,
this will be set to false).
```

Database Configuration

To complete the setup of the Artifactory HA cluster, we need to setup the external database, the Artifactory cluster presently supports:

- Oracle
- MS SQL
- PostgreSQL
- MySQL

Our concentration is going to be Oracle and MS SQL external database support, since these are the database engines that are going to be encountered in high availability enterprise implementations.

Oracle

The recommended Oracle version to use is 10.x and above, it is recommended that a dedicated Oracle database schema and tablespaces be created.

- Launch SQL*Plus and connect as a user with DBA privileges
- Create a tablespace called “artifactory_tblspc”

```
CREATE TABLESPACE "ARTIFACTORY_TBLSPC" DATAFILE
'$ORACLE_HOME\ORADATA\<DB Instance>\artifactory01.dbf' SIZE 500M
AUTOEXTEND ON NEXT 250K MAXSIZE UNLIMITED LOGGING EXTENT MANAGEMENT
LOCAL SEGMENT SPACE MANAGEMENT AUTO
```

- Create a temporary tablespace called "artifactory_temp"

```
CREATE SMALLFILE TEMPORARY TABLESPACE "ARTIFACTORY_TEMP" TEMPFILE
'$ORACLE_HOME\ORADATA\<DB instance>\artfactory_temp01.dbf' SIZE 250M
EXTENT MANAGEMENT LOCAL UNIFORM SIZE 1M
```

- Create user "artifactory"

```
CREATE USER "ARTIFACTORY" PROFILE "DEFAULT" IDENTIFIED BY "<selected-
password>" DEFAULT TABLESPACE "ARTIFACTORY_TLSPC" TEMPORARY
TABLESPACE "ARTIFACTORY_TEMP" QUOTA UNLIMITED ON ARTIFACTORY_TLSPC
ACCOUNT UNLOCK;
```

/

```
GRANT "CONNECT" TO "ARTIFACTORY";
```

/

```
GRANT "RESOURCE" TO "ARTIFACTORY";
```

/

- Start the process of downloading the ojdbc6.jar or ojdbc7.jar files from the following location: [Oracle JDBC Drivers](#). You will have to login to the Oracle OTN (free account).
 - It is recommended that you select the Database version 12.1.0.x version, since this particular JDBC version supports all the database versions from the 12.2.0 to 10.2.0.x
- Select the "JDBC Drivers, SQLJ and JPublisher Downloads" link
- Select the "Oracle Database 12c Release (12.1.0.2), (12.1.0.1)" link
- Accept the OTN License Agreement
- Download the ojdbc7.jar for Java 7 JDK or ojdbc6.jar for Java 6 JDK
- Copy the ojdbc7.jar or ojdbc6.jar to the "/opt/jfrog/artifactory/tomcat/lib" directory. *This needs to be done in each JFrog Artifactory node*
- Change the ownership, group, and execution of the ojdbc7.jar
 - "cd /opt/jfrog/artifactory/tomcat/lib"
 - "chown root ojdbc7.jar"
 - "chgrp root ojdbc7.jar"
 - "chmod 775 ojdbc7.jar"
- Place the ojdbc7.jar in the "\$JAVA_HOME/jre/lib/ext", i.e:
 - "cp ojdbc7.jar /usr/java/jdk1.7.0_67/jre/lib/ext"
- Copy the \$ARTIFACTORY_HOME/misc/db/oracle.properties to \$CLUSTER_HOME/ha-etc/storage.properties
- Make the following changes to the newly copied storage.properties file

type=oracle

```
driver=oracle.jdbc.OracleDriver
url=jdbc:oracle:thin:@<oracle_host>:1521:<ORACLE_SID or ORACLE_SERVICE>
username=artifactory
password=<assigned password for artifactory>
```

The above updates will usually work correctly, if the Oracle TNS listener has not been defined with SID entries for the database instances to connect to. This is not always the case with Oracle 10, 11, and 12c. As a workaround to this issue, you can perform the following changes:

```
type=oracle
driver=oracle.jdbc.OracleDriver
#url=jdbc:oracle:thin:@guewa01-u127435.ca.com:1521:pdbreauto50.ca.com
url=jdbc:oracle:thin:@(DESCRIPTION=(ADDRESS=(PROTOCOL=TCP)(HOST=<oracle_host>)(PORT=1521))(CONNECT_DATA=(SERVICE_NAME=<oracle_service>)))
username=artifactory
password=<assigned password for artifactory>
```

The more permanent solution is for the TNS listener to be updated with the correct SID listing, please work with your Oracle DBA to make this possible, if so desired.

MS SQL Server

Only MS SQL 2008 and above database server instances are supported by the JFrog Artifactory HA clusters. It is recommended that a dedicated MS SQL server database be created and configure Artifactory to interact with this database type.

- Create a new MS SQL server user called “artifactory”.
- Create a new MS SQL server database called “artifactory”.
- Select the “Collation” for this new database to “SQL_Latin1_General_CP1_CI_AS”
- Please download the SQL JDBC 4-3 driver from the following link [SQL JDBC 4.3](#)
- Extract the SQL JDBC 4.3 jar file and place the extracted JAR file at the following location “\$ARTIFACTORY_HOME/tomcat/lib/”. This needs to be done in each JFrog Artifactory HA cluster node.
- Now we need to make the following changes to the “\$CLUSTER_HOME/etc/storage.properties”

```
type=mssql
driver=com.microsoft.sqlserver.jdbc.SQLServerDriver
url=jdbc:sqlserver://<MS SQL Server
name>:1433;databaseName=artifactory;sendStringParametersAsUnicode=false;applicationName=Artifactory Binary Repository
username=artifactory
password=<assigned password for artifactory>
```

Artifactory HA (Demo) License

If you have already received the JFrog Artifactory HA (Demo) license, which you should have received with the location of the artifactory-pro package. Please follow these steps

- Create “\$ARTIFACTORY_HOME/etc/artifactory.lic”
- Change group and ownership to artifactory
- Change its execution parameter to 770
- Copy the license string that you received into the artifactory.lic that you just created.

You can also add the HA license via the web browser interface.

- Bring up Artifactory: http://artifactory_node1:8081/artifactory
- You will see the Artifactory Pro License panel
- Add the license string and press “Save”
- You will be asked to restart the server for the license to take effect

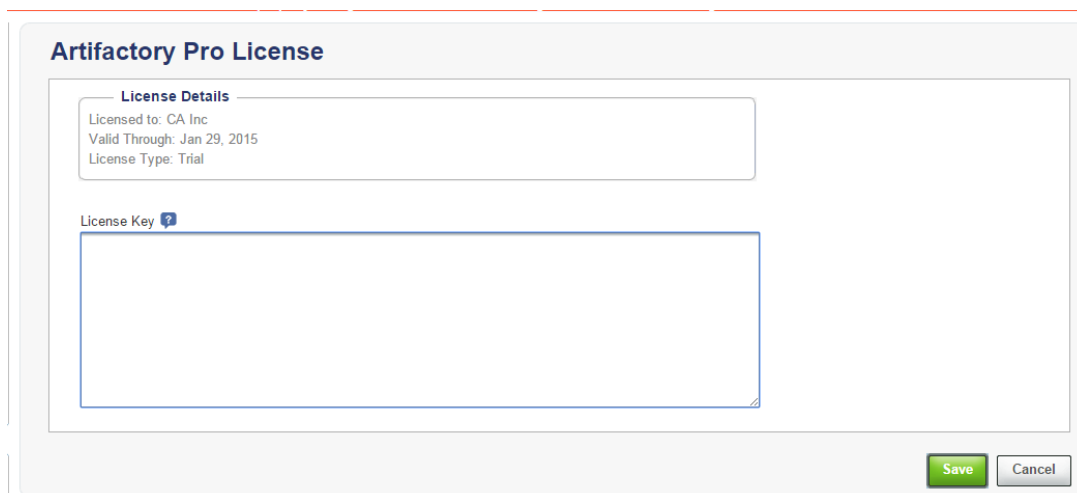


Figure 7: Artifactory Pro License Panel Example

NGINX Proxy Load Balancer

For the proper interaction with the Artifactory HA cluster it is necessary to install an http proxy load balancer, in this scenario we are going to be using the NGINX http proxy load balancer, since it is very stable and provide us with an extensive feature set, as well as the standard in enterprise implementations.

You will need to install the “NGINX” proxy server from the following location: [NGINX Install Packages](#). You will need to use the “yum” utility as per the instructions in the link provided.

You will install the desired version based on the supported platform that CA Release Automation supports, for example for Red Hat/CentOS you will select the desired installation package based off the available versions: 5.x, 6.x, or 7.x.

The version NGINX used during the writing of this document was NGINX v1.6.1.

HTTP Support

After the NGINX package has been installed, you will need to make the following changes to support HTTP.

- `/etc/nginx/config.d/nginx.conf`

```
upstream balanced_load {
    ip_hash;
    server artifactorynode1:8081;
    server artifactorynode2:8081;
}

server {
    listen 4080 ;
    server_name global-artifactory;
    client_max_body_size 2048M;
    access_log /var/log/nginx/artifactory-balancing.log;
    error_log /var/log/nginx/artifactory-balancing.error.log;

    location /artifactory/ {
        proxy_pass http://balanced_load;
        proxy_http_version 1.1;
        proxy_redirect default;
        port_in_redirect on;
    }
}
```

Save the changes, you can start the “nginx” service with the following command:

```
service nginx start
Starting nginx: [ OK ]
```

HTTPS Support

Support for HTTP SSL for JFrog Artifactory is accomplished by setting up a HTTPS proxy, given that this document concentrates in the NGINX HTTP proxy; we are going to provide the additional instructions to accomplish this task.

If you would like to use the Apache Foundation HTTP proxy, please follow this link: [Artifactory with Apache HTTPS](#).

Create SSL Certificate

We are going to create a self-signed certificate in the system where the NGINX load balancer is running.

- Create a dir at the following location “/etc/nginx”
 - Run “mkdir /etc/nginx/ssl”
- Please follow the creation of the self-signed certificate at the following link [Create self-signed certificate.](#)
- If you have signed certificates, you can also use them as well. You can either place them at the directory created above or your own location.

NGINX.CONF Entries

The following entries are provided for the SSL functional to be activated in the NGINX load balancer proxy.

The SSL entries that need to be added to the /etc/nginx/config.d/nginx.conf file.

```
upstream balanced_load {
    ip_hash;
    server artifactorynode1:8081;
    server artifactorynode2:8081;
}

server {
    listen 443 ;
    server_name global-artifactory;
    client_max_body_size 2048M;
    access_log /var/log/nginx/artifactory-balancing.log;
    error_log /var/log/nginx/artifactory-balancing.error.log;

    # SSL settings
    ssl on;
    ssl_certificate /etc/nginx/ssl/balanced_load.crt;
    ssl_certificate_key /etc/nginx/ssl/balanced_load.key;

    ssl_session_timeout 5m;

    ssl_protocols SSLv3 TLSv1;
    ssl_ciphers ALL:!ADH:!EXPORTS56:RC4+RSA:+HIGH:+MEDIUM:+LOW:+SSLv3:+EXP;
    ssl_prefer_server_ciphers on;

    location /artifactory/ {
        proxy_redirect off;
        proxy_set_header Host $host:$server_port;
        proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
        proxy_set_header X-Real-IP $remote_addr;
        proxy_set_header X-Forwarded-SSL on;
```

```

        proxy_set_header X-Forwarded-Proto $scheme;
        proxy_pass http://balanced_load;
        proxy_read_timeout 90;
        proxy_http_version 1.1;
    }
}

```

Save the changes, you can start the “nginx” service with the following command:

```

service nginx start
Starting nginx: [ OK ]

```

Additional Changes in Artifactory

After making the necessary changes to the nginx.conf for the NGINX http proxy server, it is necessary that the following files in the Artifactory servers (nodes) be updated.

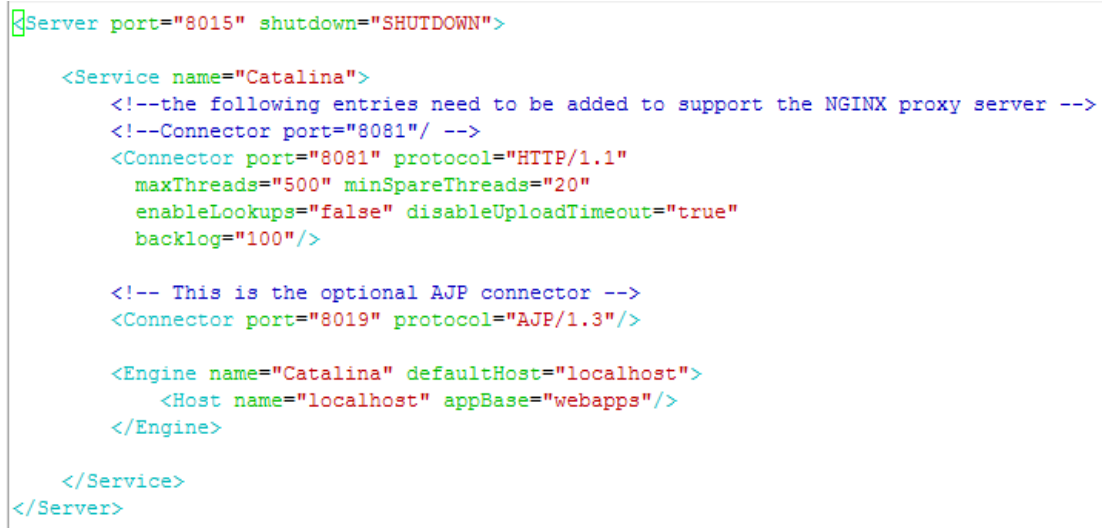
The ‘\$ARTIFACTORY_HOME/tomcat/conf/server.xml’ file needs to be updated in the XML tag below.

```

<Connector port="8081" protocol="HTTP/1.1"
maxThreads="500" minSpareThreads="20"
enableLookups="false" disableUploadTimeout="true"
backlog="100"/>

```

As shown in the Figure 8 below, you can see a complete example of how the additional XML tag will look in the server.xml file.



```

<Server port="8015" shutdown="SHUTDOWN">

  <Service name="Catalina">
    <!--the following entries need to be added to support the NGINX proxy server -->
    <!--Connector port="8081"/ -->
    <Connector port="8081" protocol="HTTP/1.1"
      maxThreads="500" minSpareThreads="20"
      enableLookups="false" disableUploadTimeout="true"
      backlog="100"/>

    <!-- This is the optional AJP connector -->
    <Connector port="8019" protocol="AJP/1.3"/>

    <Engine name="Catalina" defaultHost="localhost">
      <Host name="localhost" appBase="webapps"/>
    </Engine>
  </Service>
</Server>

```

Figure 8: Example of the server.xml changes

```
<Valve className="org.apache.catalina.valves.RemoteIpValve" protocolHeader="x-forwarded-proto"/>
```

Figure 9: Updated contents of artifactory.xml

Starting JFrog Artifactory

- Login to the Artifactory servers, in our case, this is going to be the following two nodes: nolio1 and nolio2.
- Run the following command “service artifactory start”
- Check that the \$ARTIFACTORY_HOME/logs/artifactory.log contains the entry as shown in Figure 10.

Figure 10: Results of artifactory.log

Once the Artifactory HA cluster is up and running, please login (default username to use is “admin” and the default password is “password” – *please change this password, after this change is made*) to the primary node directly, and make the following change to setup a “custom base URL”, this URL will point to the NGINX http proxy server.

You will need to go to the “Admin” tab, “General Configuration” section, and you will see the “Custom URL Base” field, it is in this field that you will enter the NGINX http proxy server URL that you will be using. For example: ‘<http://demo-server:4080/artifactory>’, if you are using the HTTP proxy settings, but for *HTTPS*, you would need to have these entries default to ‘<https://demo-server:443/artifactory>’.

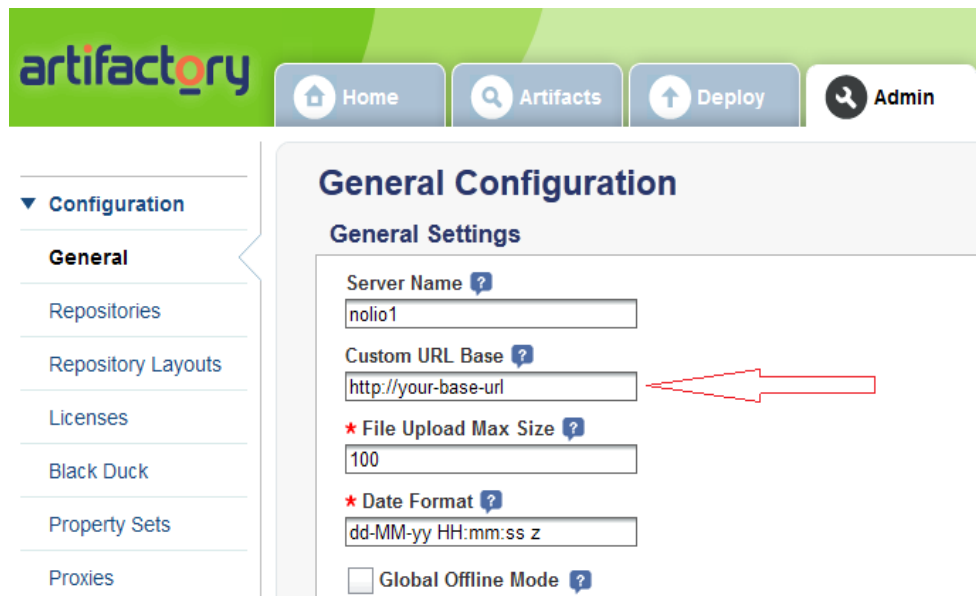


Figure 11: Custom URL Base entry

After the above changes have been made, please login back into the Artifactory HA cluster via the NGINX http defined proxy server, and navigate to the Admin→High Availability section, and you are going to see the nodes that make up the Artifactory HA cluster as shown below.

Configure High Availability									
All Artifactory Nodes									
ID	Start Time	URL	Membership Port	State	Role	Last Heartbeat	Version	Revision	Release Date
nolio1	12-08-14 15:32:03 -05:00	http://10.130.238.244:8081/artifactory	10001	Running	Primary	12-08-14 16:39:03 -05:00	3.2.2	30097	21-06-14 14:17:24 -05:00
nolio2	12-08-14 15:30:04 -05:00	http://10.130.211.20:8081/artifactory	10001	Running	Member	12-08-14 16:39:04 -05:00	3.2.2	30097	21-06-14 14:17:24 -05:00

Figure 12: Artifactory HA Nodes

Install and Configure Release Automation

After the Artifactory HA cluster’s components have been installed and configure, it is now time to install and configure the CA Release Automation Server. Please follow the installation instructions as defined in this link “[Installation Instructions](#) link”. This will also apply to high availability installation of Release Automation. The same rules apply as well.

Please keep in mind that you might already have a Release Automation (in High Availability, if so desired) installation running, and you can move to “Updates to nolio-repo.properties” section below.

Here are some screen shots of what the installation screens will look like.

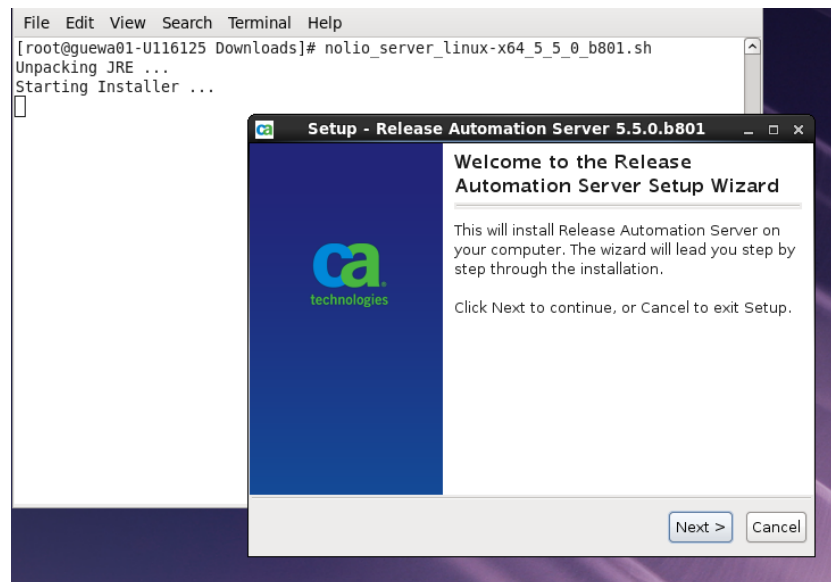


Figure 13: A typical Release Automation Installation Screen

After you have installed Release Automation, you will need to perform the following steps.

You will need to stop the Release Automation service by using the following commands:

- “\$RELAUTO_HOME/UpdateService/nolio_update_service stop”
- “\$RELAUTO_HOME/NolioAgent/nolio_agent.sh stop”
- “\$RELAUTO_HOME/nolio_server.sh stop”

Updates to nolio-repo.properties

Update the \$RELAUTO_HOME/conf/nolio-repo.properties, where the important changes will be the “type” property, which needs to default to “artifactory”.

```
# If you intend to use encrypted repository password, Please use the encrypt_password.bat/sh utility to
# encrypt the password.
type=artifactory
scheme=http
hostname=global-artifactory
port=4080
repositoryPath=/artifactory/nolio
actionRepositoryPath=/artifactory/nolio-actions
manifestRepositoryPath=/artifactory/nolio-manifests
username=admin
password=<password>
passwordEncrypted=false
deleteAnonymousUser=false
```

Figure 14: The artifactory values in the nolio-repo.properties

Figure 14 shows the values that will be need to added to the “nolio-repo.properties” for Release Automation to interact with the JFrog Artifactory HA cluster. Please update the following fields:

- Type
- Hostname
- Port
- Username
- Password
- passwordEncrypted should be set to true, if you are encrypting the password
- RepositoryPath
- ActionRepositoryPath
- ManifestRepositoryPath

The password can be encrypted with the provided Release Automation utility, if so desired.

The same changes need to be made to the NACs that make up the Release Automation high availability cluster, as this change is made for the master NAC, it also needs to be made to the files in the stand-by NAC as well.

Import Certificate into “Nolio.jks”

If the type value in the ‘nolio-repo.properties’ file is set to ‘HTTPS’, it is necessary that the “Nolio.jks” keystore be updated with the newly created certificate. This keystore is located in the ‘conf’ directory of the NAC servers and the Release Automation agents systems. Here are the instructions to follow

- Copy the newly created certificate to the NAC servers and the Release Automation agents.
- Place this certificate in the ‘\$RELAUTO_HOME/conf’ directory.
- Change directory to the \$RELAUTO_HOME/conf directory.
- Make a copy of the ‘Nolio.jks’ store.

- In the NAC servers, run the following command: “\$RELAUTO_HOME/jre/bin/keytool –import –file ./mycreated.cer –alias balanced_load –keystore ./nolio.jks”
- Copy the newly created certificate to the embedded agent in the NAC server, “\$RELAUTO_HOME/NolioAgent/conf/”
- Change directory to the ‘\$RELAUTO_HOME/NolioAgent/conf’
- Make a copy of the ‘Nolio.jks’ keystore.
- For the embedded agent in the NAC servers, run:
“”\$RELAUTO_HOME/jre/bin/keytool –import –file ./mycreated alias balanced_load –keystore ./nolio.jks”
- For stand-alone Release Automation agent systems, the same steps as for the embedded agent will apply.

Artifactory Repositories

You will need to manually create the following repositories in the Artifactory HA cluster:

- nolio
- nolio-actions
- nolio-manifests

Login to the Artifactory HA cluster via the NGINX http proxy that you have already setup, then go to Admin→Configure Repositories, and click the “New” button to create the repositories listed above.

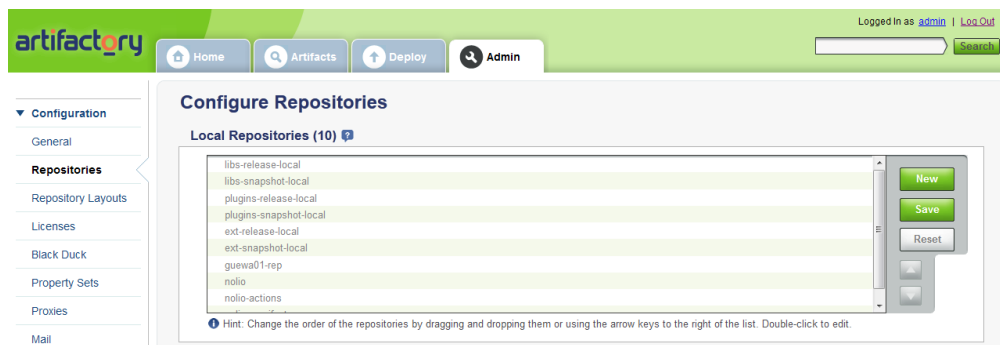


Figure 15: Creating Release Automation Repositories

The repositories need to be following the “maven-2-default”. Please use the following settings as an example for the Release Automation repositories to create.

★ Repository Key ?

nolio

Basic Settings Property Sets Replication Packages

Public Description ?

The default "nolio" repository

Internal Notes ?

This is the default "nolio" repository

Includes Pattern ?

**/*

Excludes Pattern ?

Repository Layout ?

maven-2-default

Checksum Policy ?

Verify against client checksums

Maven Snapshot Version Behavior ?

Unique

Max Unique Snapshots ?

0

☒ Handle Releases ? ☒ Handle Snapshots ?

☐ Blacked Out ? ☐ Suppress POM Consistency Checks ?

☒ Allow Content Browsing ?

Figure 16: Repository settings

These are the recommended values to use.

Starting up all components

After all the necessary changes have been accomplished, we need to make sure that the components are running in the following order.

- Data storage – this where the NFS partition is running.
- Database instance – the database instance is running.
- NGINX – make sure that all the changes to the NGINX http load balancer have been accomplished and the http load balancer is running,
- JFrog Artifactory HA – The JFrog Artifactory HA is running and you can access it via the NGINX load balanced URL.
- Release Automation – Bring up Release Automation after all the changes have been made.
- Update Release Automation Action Packs – Connect via the Release Automation ROC and update the Action Packs actions as necessary.

Now Release Automation is ready to be used with the JFrog Artifactory HA cluster repository.

Useful Links

<http://www.jfrog.com/confluence/display/RTF/Artifactory+High+Availability>

<http://www.jfrog.com/confluence/display/RTF/Changing+the+Default+Storage>

<http://nginx.org/en/>

<http://nginx.org/en/docs/>

<http://nginx.com/blog/nginx-load-balance-deployment-models/>

<https://wiki.ca.com/display/RA55/CA+Release+Automation+Home>

<https://wiki.ca.com/display/RA50/CA+Release+Automation+Home>