YUM Repositories

Overview

Artifactory is a fully-fledged RPM repository. As such, it enables:

1. RPM metadata calculation for RPMs hosted in Artifactory local repositories.
2. Provisioning RPMs directly from Artifactory to YUM clients.
3. Detailed RPM metadata views from Artifactory's web UI.
4. Providing GPG signatures that can be used by the YUM client to authenticate RPMs.

RPM Metadata for Hosted RPMs

The RPM metadata generated by Artifactory is identical to the basic-mode output of the Red Hat-based Linux command `createrepo`.

A folder named `repodata` is created in the configured location within a local repository with the following files in it:

File	Description
primary.xml.gz	Contains an XML file describing the primary metadata of each RPM archive.
filelists.xml.gz	Contains an XML file describing all the files contained within each RPM archive.
other.xml.gz	Contains an XML file describing miscellaneous information regarding each RPM archive.
repomd.xml	Contains information regarding all the other metadata files.

YUM Support is Platform Independent!

Artifactory's RPM metadata calculation is based on **pure Java.**

It does not rely on the existence of the `createrepo` binary or on running external processes on the host on which Artifactory is running.

Page Contents

Triggering RPM Metadata Updates

When enabled, the metadata calculation is triggered automatically by some actions, and can also be invoked manually by others. Either way, the metadata produced is served to YUM clients.

Automatic

RPM metadata is automatically calculated:

1. When deploying/removing/copying/moving an RPM file.

2. When performing content import (both system and repository imports).

Manual

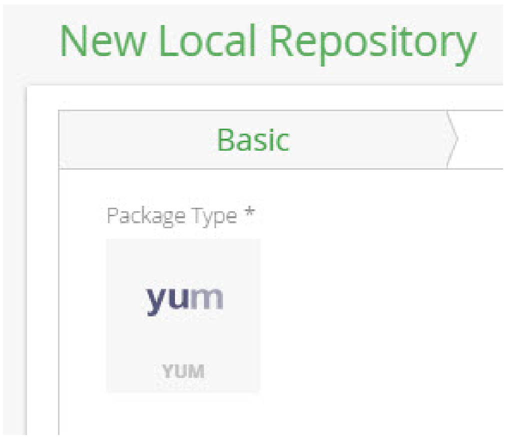
You can manually invoke RPM metadata calculation:

1. By selecting the local repository in the Tree Browser and clicking **Recalculate Index** in the **Actions** menu.
2. Via Artifactory's REST-API.

> Metadata calculation cleans up RPM metadata that already existed as a result of manual deployment or import. This includes RPM metadata stored as SQLite database files.

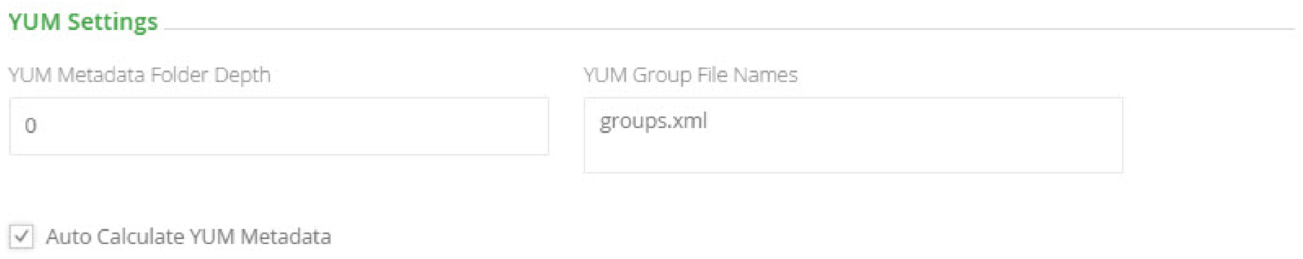
Configuration

To create an RPM local repository, select **RPM** as the **Package Type** when you create the repository.



Local Repositories

To enable automatic RPM metadata calculation on a local YUM repository, in the **YUMSettings** section of the **Basic** settings screen, set **Auto-calculate YUM Metadata**.



YUM Metadata Folder Depth: 0
YUM Group File Names: groups.xml
[✓] Auto Calculate YUM Metadata

Field	Description

YUM Metadata Folder Depth	Informs Artifactory under which level of directory to search for RPMs and save the `repodata` directory.
	
	By default this value is 0 and refers to the repository's root folder. In this case, Artifactory searches the entire repository for RPMs and saves the `repodata` directory at *`$REPO-KEY/repodata`*.
	
	Using a different depth is useful in cases where generating metadata for a repository separates its artifacts by name, version and architecture. This will allow you to create multiple RPM repositories under the same Artifactory RPM repository.
	
	For example:
	If the repository layout is similar to that shown below and you want to generate RPM metadata for every artifact divided by name, set the `Depth` to `1` and the `repodata` directory is saved at *`REPO_ROOT/ARTIFACT_NAME/repodata`* :
	
	`REPO_ROOT/$ARTIFACT_NAME/$ARTIFACT_VERSION/$ARCHITECTURE/FILE_NAME`
	`- or -`
	`rpm-local/foo/1.0/x64/foo-1.0-x64.rpmm`
	
	When changing the configured depth of existing repository, packages indexed in the old depth might need to be re-indexed or moved to a new depth to be available in the new configured depth, and YUM clients might need to change their configuration to point to the new depth.depth.

YUM Group File Names	A comma-separated list of YUM group files associated with your RPM packages. Note that at each level (depth), the `repodata` directory in your repository may contain a different group file name, however each `repodata` directory may contain only 1 group metadata file (multiple groups should be listed as different tags inside the XML file. For more details, please refer to the YUM Documentation).
Auto-calculate YUM Metadata	When set, YUM metadata calculation is automatically triggered by the actions described above.

Metadata calculation is asynchronous and does not happen immediately when triggered, whether automatically or manually.

Artifactory invokes the actual calculation only after a certain "quiet period", so the creation of metadata normally occurs only 1-2 minutes after the calculation was triggered.

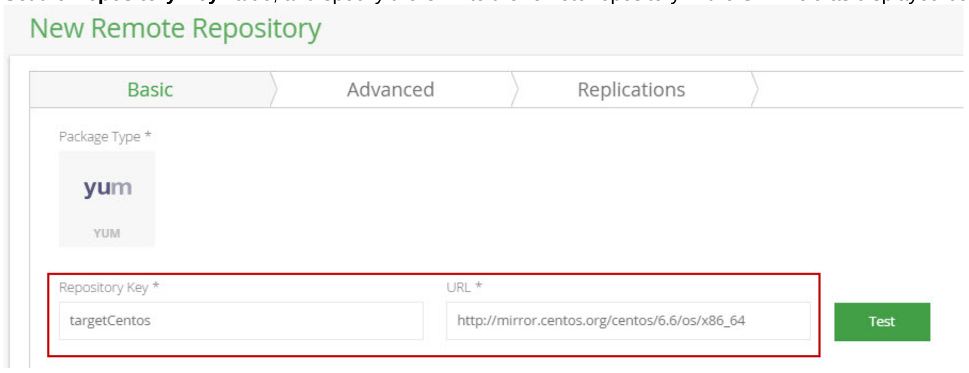
Remote Repositories

Artifactory remote repositories support RPMs out-of-the-box, and there is no need for any special configuration needed in order to work with RPMs in a remote repository.

All you need to do is point your YUM client at the remote repository, and you are ready to use YUM with Artifactory.

To define a remote repository to proxy an YUM remote repository, follow the steps below:

1. In the **Admin** module under **Repositories | Remote,** click "New" to create a new remote repository.
2. Set the **Repository Key** value, and specify the URL to the remote repository in the **URL** field as displayed below.



New Remote Repository

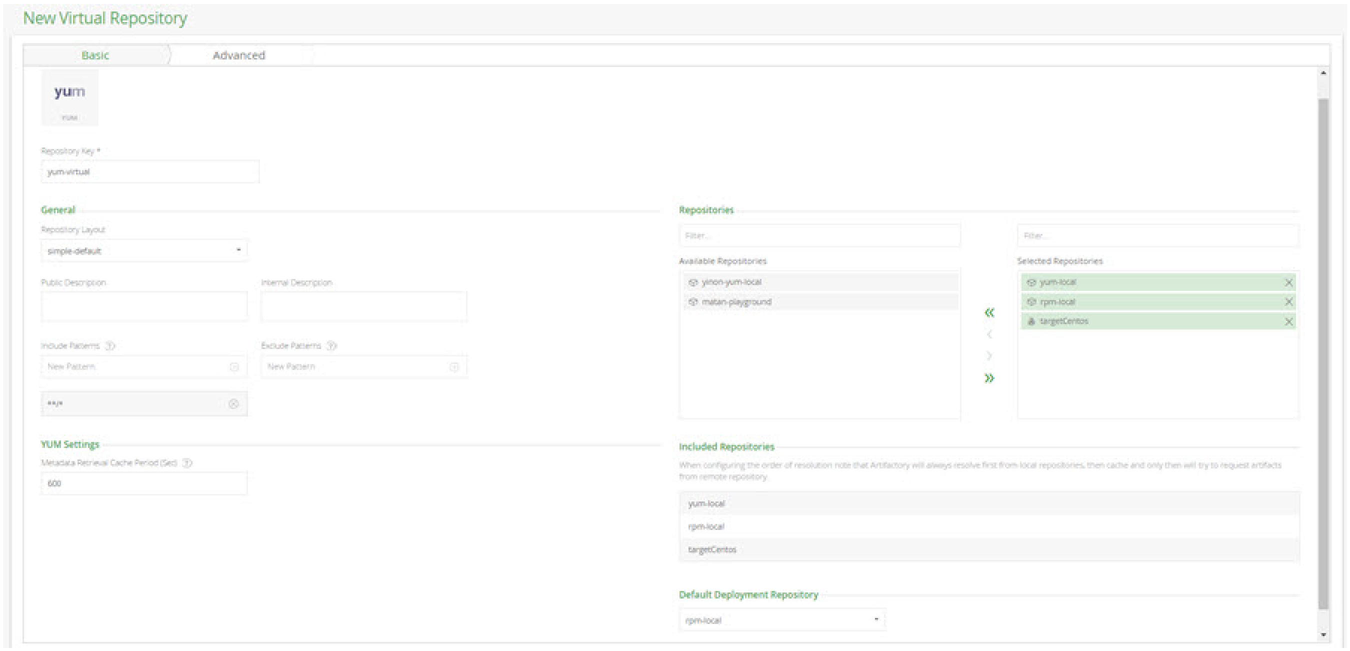
Repository Key *: targetCentos
URL *: http://mirror.centos.org/centos/6.6/os/x86_64

3. Click "Save & Finish"
4. Back in the **Artifacts** module, in the**Tree Browser,** select the repository. Note that in the Tree Browser, the repository name is appended with "-cache".
5. Click **Set Me Up** and copy the value of the **baseurl** tag.
6. Next, create the *`/etc/yum.repos.d/targetCentos.repo`* file and paste the following configuration into it:

```
[targetCentos]
name=targetCentos
baseurl=http://localhost:8081/artifactory/targetCentos/
enabled=1
gpgcheck=0
```

Virtual Repositories

A Virtual Repository defined in Artifactory aggregates packages from both local and remote repositories.
This allows you to access both locally hosted RPM packages and remote proxied RPM repositories from a single URL defined for the virtual repository.
To define a virtual YUM repository, create a virtual repository, set the **Package Type** to be **RPM,** and select the underlying local and remote RPM repositories to include in the **Basic** settings tab.



Repository Key * yum-virtual

General

Repository Layout: simple-default

Public Description

Internal Description

Include Patterns: New Pattern, **/*

Exclude Patterns: New Pattern

YUM Settings

Metadata Retrieval Cache Period (Sec): 600

Repositories

Available Repositories: yinon-yum-local, matan-playground

Selected Repositories: yum-local, rpm-local, targetCentos

Included Repositories

When configuring the order of resolution note that Artifactory will always resolve first from local repositories, then cache and only then will try to request artifacts from remote repository.

yum-local
rpm-local
targetCentos

Default Deployment Repository: rpm-local

To allow deploying packages to this repository, set the Default Deployment Repository.

Signing RPM Metadata

Artifactory supports using a GPG key to sign RPM metadata for authentication by the YUM client.

To generate a pair of GPG keys and upload them to Artifactory, please refer to GPG Signing.

Using Yum to Deploy RPM Packages

Once you have configured your local machine to install RPM packages from your YUM local repository, you may also deploy RPM packages to the same repository using the UI or using the REST API.

Through the REST API you also have the option to deploy by checksum or deploying from an archive.

For example, to deploy an RPM package into a repository called *`rpm-local`* `you could use the following:`

```
curl -u<USERNAME>:<PASSWORD> -XPUT http://localhost:8080/artifactory/rpm-local/<PATH_TO_METADATA_ROOT> -T
<TARGET_FILE_PATH>
```

where PATH_TO_METADATA_ROOT specifies the path from the repository root to the deploy folder.

YUM Groups

A YUM group is a set of RPM packages collected together for a specific purpose. For example, you might collect a set of "Development Tools" together as a YUM group.

A group is specified by adding a group XML file to same directory as the RPM packages included in it. The group file contains the metadata of the group including pointers to all the RPM files that make up the group.

Artifactory supports attaching a YUM Group file to the YUM calculation essentially mimicking the createrepo -g command.

A group file can also be created by running the following command:

```
sudo yum-groups-manager -n "My Group" --id=mygroup --save=mygroups.xml --mandatory yum glibc rpm
```

Attaching a YUM Group

The process of attaching YUM group metadata to a local repository is simple:

1. Create an XML file in the groups format used by YUM. You can either just type it out manually using any text editor, or run the `yum-groups-manager` command from `yum-utils`.
2. Deploy the created group file to the *`repodata`* folder.
   Artifactory will automatically perform the following steps:
   - Create the corresponding *`.gz`* file and deploy it next to the deployed group XML file.
   - Invoke a YUM calculation on the local repository.
   - Attach the group information (both the XML and the *`.gz`* file) to the *`repomd.xml`* file.
3. Make sure the group file names are listed in the **YUM Group File Names** field of the Packages tab. This tells Artifactory which files should be attached as repository group information.

YUM Group Commands

The following table lists some useful YUM group commands:

Command	**Description**
`yum groupinstall <Group ID>`	Install the YUM group. The group must be deployed to the root of the YUM local repository.
`yum groupremove <Group ID>`	Remove the RPM group
`yum groupupdate <Group ID>`	Update the RPM group. The group must be deployed to the root of the YUM local repository.
`yum groupinfo <Group ID>`	List the RPM packages within the group.
`yum grouplist | more`	List the YUM groups

Setting Group Properties

YUM group properties can be set in the *`/etc/yum.config`* file as follows:

Setting	**Allowed values**	**Description**
overwrite_groups	0 or 1	Determines YUM's behavior if two or more repositories offer package groups with the same name. If set to 1 then the group packages of the last matching repository will be used. If set to 0 then the groups from all matching repositories will be merged together as one large group.
groupremove_leaf_only	0 or 1	Determines YUM's behavior when the **groupremove** command is run. If set to 0 (default) then all packages in the group will be removed. If set to 1 then only those packages in the group that aren't required by another package will be removed.
enable_group_conditionals	0 or 1	Determines whether YUM will allow the use of conditionals packages. If set to 0 then conditionals are not allowed If set to 1 (default) package conditionals are allowed.
group_package_types	optional, default, mandatory	Tells YUM which type of packages in groups will be installed when `groupinstall` is called. Default is: default, mandatory

Yum Authentication

Proxy Server Settings

If your organization uses a proxy server as an intermediary for Internet access, specify the `proxy` settings in */etc/yum.conf.* If the proxy server also requires authentication, you also need to specify the `proxy_username`, and `proxy_password` settings.

```
proxy=<proxy server url>
proxy_username=<user>
proxy_password=pass
```

If you use the yum plugin (`yum-rhn-plugin`) to access the ULN, specify the `enableProxy` and `httpProxy` settings in */etc/sysconfig/rhn/up2date.* In addition, If the proxy server requires authentication, you also need to specify the `enableProxyAuth`, `proxyUser`, and `proxyPassword` settings as shown below.

```
enableProxy=1
httpProxy=<proxy server url>
enableProxyAuth=1
proxyUser=<user>
proxyPassword=<password>
```

SSL Setting

YUM supports SSL from version 3.2.27.

To secure a repository with SSL, execute the following steps:

- Generate a private key and certificate using OpenSSL.
- Define your protected repository in a *.repo* file as follows:

```
[protected]
name = SSL protected repository
baseurl=<secure repo url>
enabled=1
gpgcheck=1
gpgKey=<URL to public key>
sslverify=1
sslclientcert=<path to .cert file>
sslclientkey=<path to .key file>
```

where:
gpgkey is a URL pointing to the ASCII-armored GPG key file for the repository . This option is used if YUM needs a public key to verify a package and the required key has not been imported into the RPM database.
If this option is set, YUM will automatically import the key from the specific URL. You will be prompted before the key is installed unless the **assumeyes** option is set.

Using Yum Variables

You can use and reference the following built-in variables in `yum` commands and in all YUM configuration files (i.e. */etc/yum.conf* and all *.repo* files in the */etc/yum.repos.d/* directory):

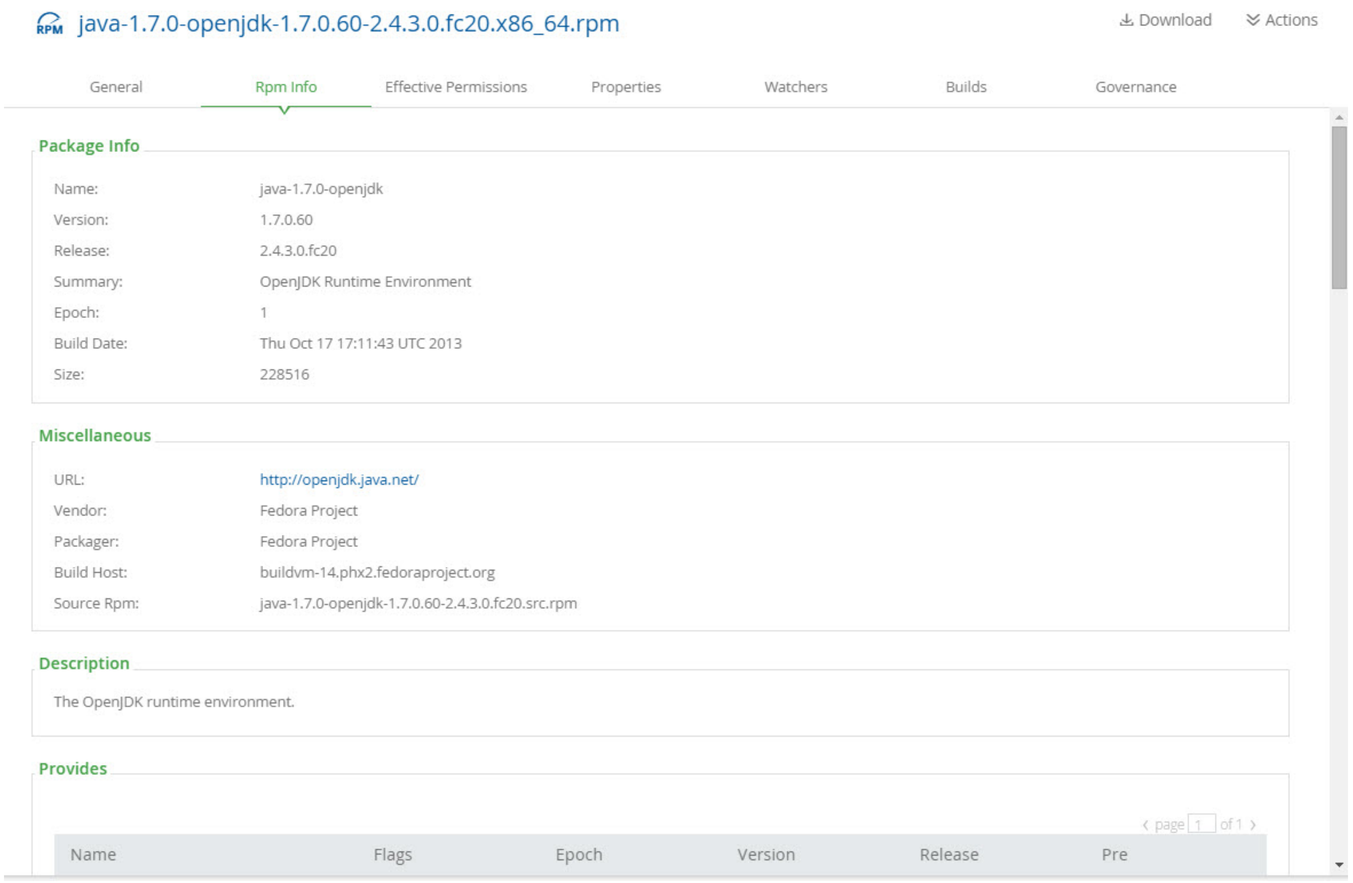
Variable	Description
$releasever	This is replaced with the package's version, as listed in `distroverpkg`. This defaults to the version of the `redhat-release` packa ge.
$arch	This is replaced with your system's architecture, as listed by `os.uname()` in Python.
$basearch	This is replaced with your base architecture. For example, if `$arch`=i686 then `$basearch`=i386

The following code block is an example of how your */etc/yum.conf* file might look:

```
[main]
cachedir=/var/cache/yum/$basearch/$releasever
keepcache=0
debuglevel=2
logfile=/var/log/yum.log
exactarch=1
obsoletes=1
gpgcheck=1
plugins=1
installonly_limit=3
[comments abridged]
```

Viewing Individual RPM Information

You can view all the metadata that annotates an RPM by choosing it in Artifactory's tree browser and selecting the **RPM Info** tab:



java-1.7.0-openjdk-1.7.0.60-2.4.3.0.fc20.x86_64.rpm

Package Info

Name:	java-1.7.0-openjdk
Version:	1.7.0.60
Release:	2.4.3.0.fc20
Summary:	OpenJDK Runtime Environment
Epoch:	1
Build Date:	Thu Oct 17 17:11:43 UTC 2013
Size:	228516

Miscellaneous

URL:	http://openjdk.java.net/
Vendor:	Fedora Project
Packager:	Fedora Project
Build Host:	buildvm-14.phx2.fedoraproject.org
Source Rpm:	java-1.7.0-openjdk-1.7.0.60-2.4.3.0.fc20.src.rpm

Description

The OpenJDK runtime environment.

Provides

page 1 of 1

Name	Flags	Epoch	Version	Release	Pre

Metadata Fields as Properties

The corresponding RPM metadata fields are automatically added as properties of an RPM artifact in YUM repositories accessed through Artifactory:

- rpm.metadata.name
- rpm.metadata.arch
- rpm.metadata.version
- rpm.metadata.release
- rpm.metadata.epoch
- rpm.metadata.group
- rpm.metadata.vendor
- rpm.metadata.summary

Properties can be used for searching and other functions. For more details please refer to Properties.

Watch the Screencast

Watch this short screencast to learn how easy it is to host RPMs in Artifactory.