Conda Repositories

Overview

From version 6.3, Artifactory natively supports Conda repositories for Python, R, Ruby, Lua, Scala, Java, JavaScript, C/ C++, FORTRAN and additional programming languages, giving you full control of your deployment and resolution process of Conda packages.

Conda repositories in Artifactory offer the following benefits:

- Secure and private local Conda repositories with fine-grained access control.
- The ability to proxy remote Conda resources and cache downloaded Conda packages to keep you independent of the network and the remote resource.
- Virtual Conda repositories that support a single URL through which to manage the resolution and deployment of all your Conda packages.
- Metadata calculation of the Conda packages hosted in the Artifactory local repositories.
- Version management: Archiving older versions of the packages uploaded to local repositories.

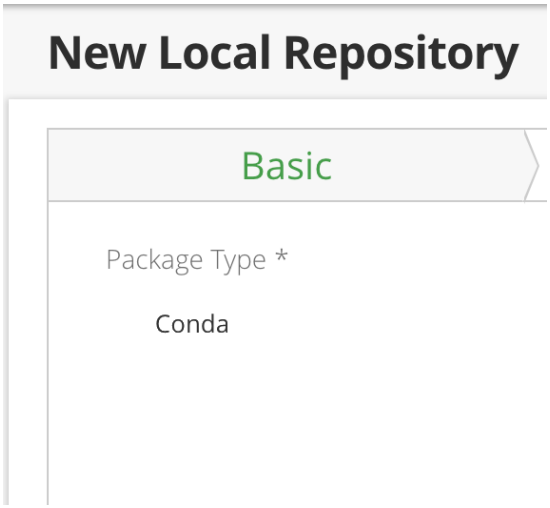
> **Minimal supported Conda client version**
>
> Artifactory supports Conda Client version 4.3.0 and above. We recommend using the latest version.

Page Contents

Configuration

Local Repositories

To enable calculation of Conda metadata, select **Conda** as the **Package Type** when you create your local repository.



Local Repository Layout

The local Conda repository in Artifactory gives you the flexibility of deploying your packages in a layout of your choice. When deploying Conda binaries into nested paths, it is important to ensure that the channel URL inside your `.condarc` file correctly reflects the path of the packages. The Conda client automatically appends the host machine platform as a subdirectory in the channel URL. When you deploy these packages, you need to meet this requirement and upload your packages into the relevant subdirectories. For example, consider the following valid package path in Artifactory:

```
conda-local/osx-64/my-conda-package.tar.bz2
```

Based on this layout, the corresponding channel URL in your `.condarc` file is:

```
<YOUR_ARTIFACTORY_URL>/artifactory/api/conda/conda-local
```

In the above example, Conda will be appending the platform automatically (i.e osx-64).

Another example shows a more detailed deployment path such as:

```
<YOUR_ARTIFACTORY_URL>/artifactory/conda-local/my/own/layout/osx-64/my-conda-package.tar.bz2
```

In this example, the channel URL should be set as follows:

```
<YOUR_ARTIFACTORY_URL>/artifactory/api/conda/conda-local/my/own/layout
```

Best Practice for Scaling Up

The Conda repository metadata is maintained on the package level, meaning that the parent directory of a Conda package is also the parent of the Conda `repodata.json` metadata file. To scale with maximum efficiency, refrain from deploying large amounts of packages into a single parent directory when possible. To enable Artifactory metadata calculation processes to provide maximal performance, try to plan your local repository layouts in a way that supports modularity by avoiding directories with large amounts of artifacts as their direct children.

Remote Repositories

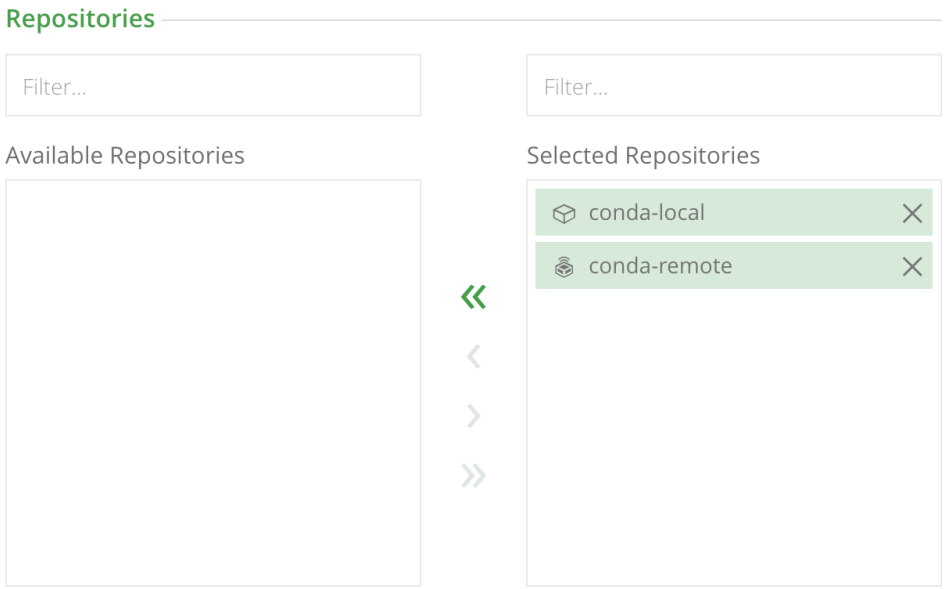
You can create a Conda remote repository to proxy and cache remote repositories or other Artifactory instances.

Note that the index files for remote Conda repositories are stored and renewed according to the Retrieval Cache Period setting on your remote repository.

Virtual Repositories

A virtual repository in Artifactory aggregates packages from both local and remote repositories allowing you to access both locally hosted Conda packages and remote proxied Conda libraries from a single URL defined for the virtual repository.

To create a virtual Conda repository, set **Conda** as the **Package Type**, and select the underlying local and remote Conda repositories to include under the **Repositories** section.



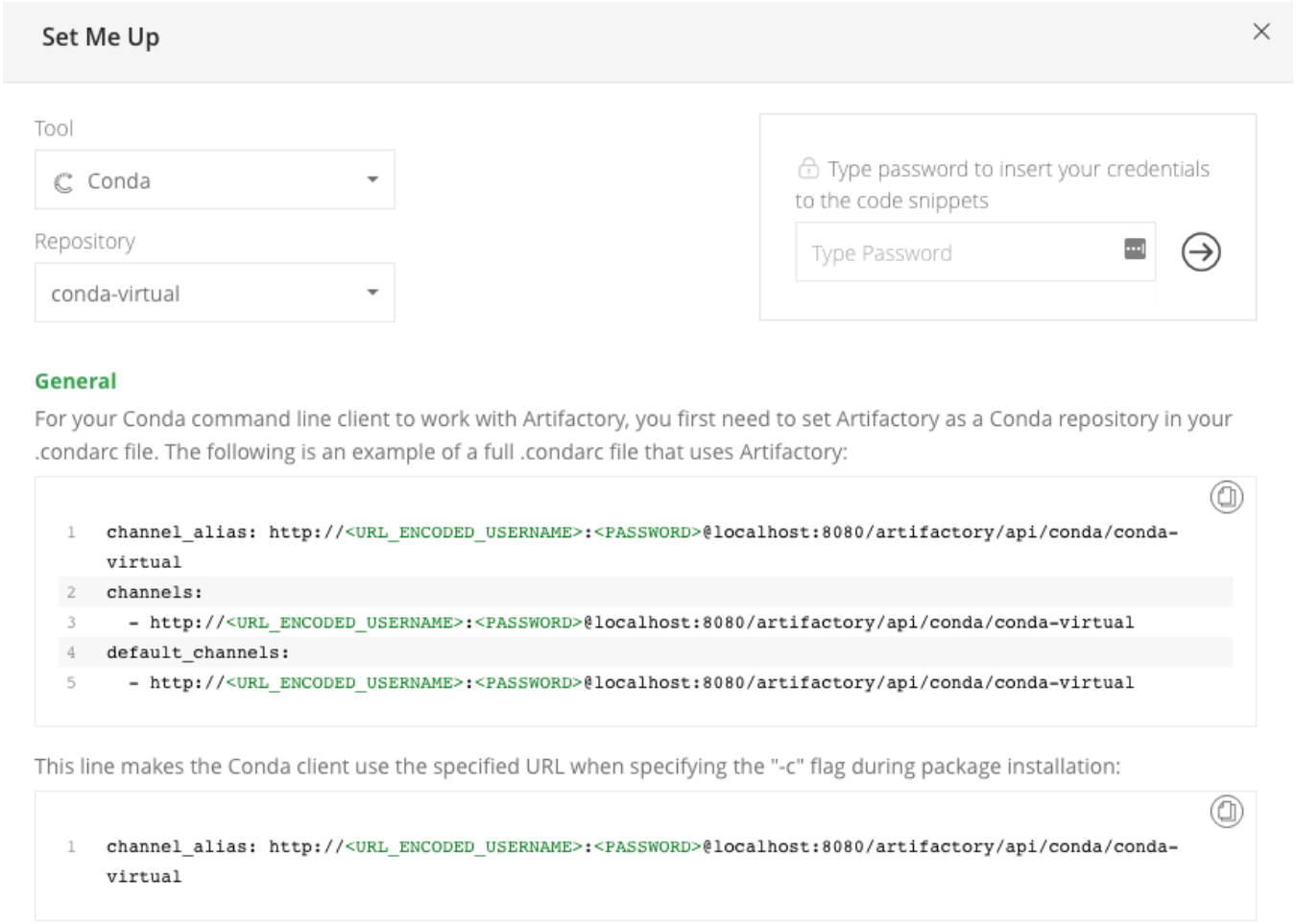
Virtual Repository Metadata

Artifactory maintains your aggregated virtual repository metadata in a clever way that keeps your metadata up-to-date by reflecting changes in the aggregated **local** repositories in real-time. For aggregated **remote** repositories, the virtual repository metadata is renewed on-demand, in minimal intervals of 10 minutes by default. The renewal period is controlled using the Retrieval Cache Period parameter of your virtual repository.

Resolving Conda Packages

Resolving Conda Packages in the UI

When a Conda repository is selected in the Artifacts Tree Browser, click **Set Me Up** to view the code snippets you can use to publish a Conda package or to configure your Conda client to resolve artifacts using the selected repository.



Resolving Conda Packages using the Conda Client

1. Perform the process of settings up your `.condarc` file according to the instructions in the **SET ME UP** screen for Conda.

2. Install a package from your Artifatory Conda repository:

```
conda install <PACKAGE_NAME>
```

3. Install a package from a specific sub-channel inside your Conda repository:

```
conda install -c <CHANNEL_NAME> <PACKAGE_NAME>
```

4. Search for a package in your Artifactory Conda repository:

```
conda search <PACKGE_NAME>
```

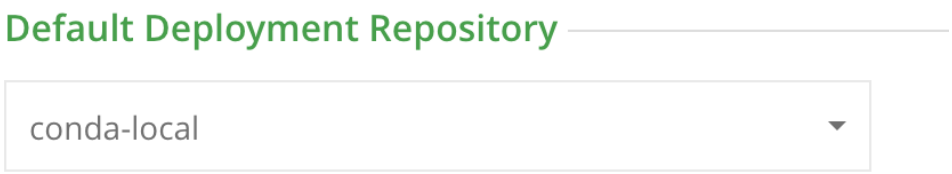
Deploying Conda Packages

You can deploy packages to a local or virtual Conda repository using the **Deploy** feature in the UI or using an HTTP client of your choice.

> **Metadata Updates**
>
> The Conda metadata is automatically calculated and updated when adding, removing, copying or moving Conda packages. The calculation is only invoked after a package-related action is completed.
> It may sometimes take up to 30 seconds to complete as the process is asynchronous and its performance depends on the overall system load.
> Although rarely required, you may wish to invoke metadata calculation on the entire repository. This can be done using the **Recalculate Index** option after you right-click the repository in the Tree Browser, or via the REST API.

Setting the Default Deployment Repository

To deploy Conda packages to a virtual Conda repository, make sure you have set the **Default Deployment Repository**.

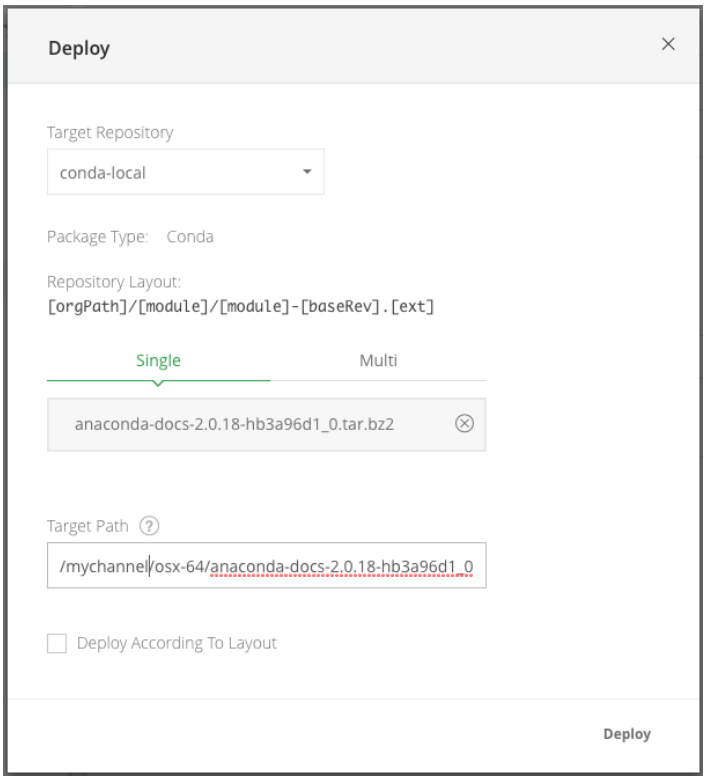


Deploying a package using the UI

You can drag and drop, or select a Conda package to upload in Deploy in the UI.

Deploying a Source Package

When deploying sources, the Target Path is automatically displayed and we recommend not changing this path. Changing the '`src/contrib`' path will result in Artifactory not identifying the package as a Conda package since Artifactory will not be able to index it.



Deploying a Binary Package

When deploying binaries, you'll need to configure the settings in the Conda Artifact section.

In the Conda Artifact section, configure these fields when deploying the Conda packages. It is mandatory to set these fields which are used to create the destination path of the deployed binary package.

- Distribution: Specifies the operating system.
- R Version: Indicates the R version used.

> **Target Path**
>
> The Target path is updated **after** the file is deployed and there is no need to change it.

Deploying a Package using cURL

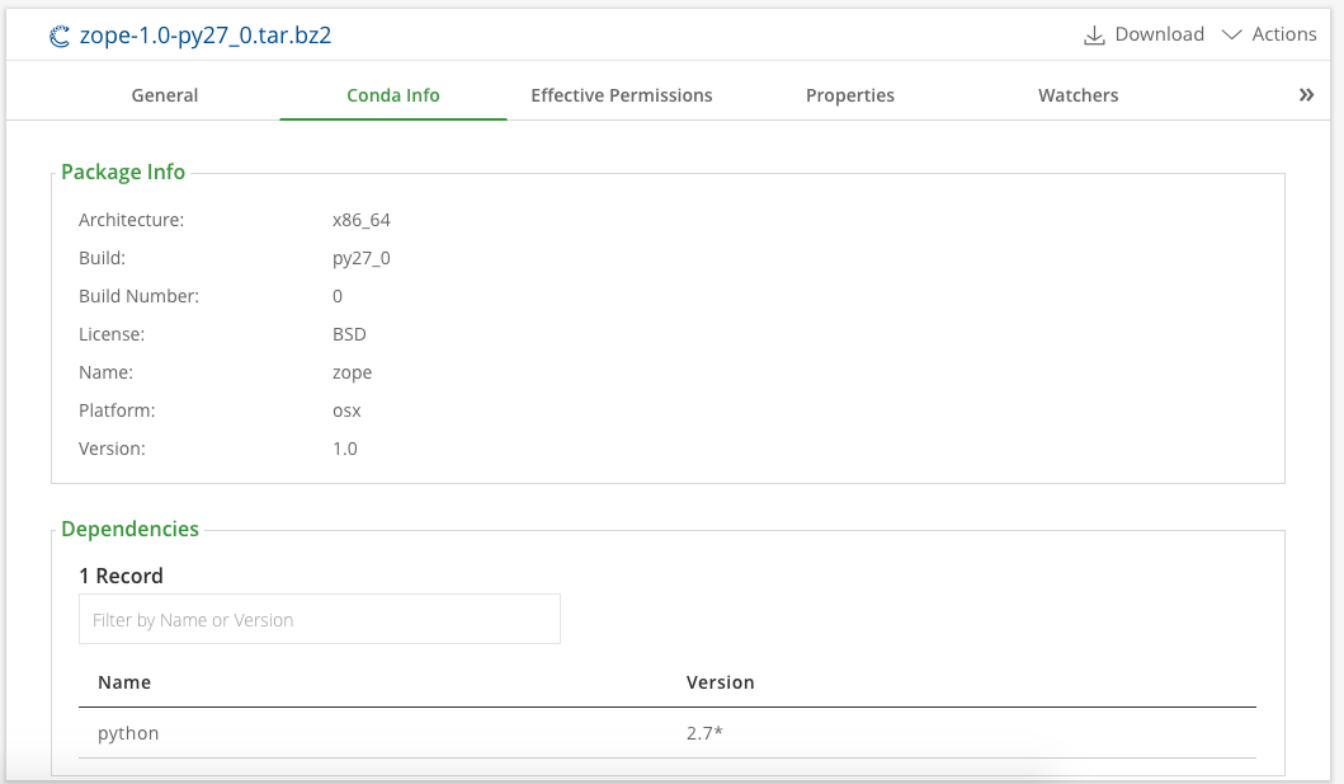
You can deploy a Conda package using cURL:

Deploy source package

```
curl -XPUT ${USERNAME}:${PASSWORD} "http://localhost:8080/artifactory/api/conda/conda-local/" -T my-package-1.0.0.tar.bz2
```

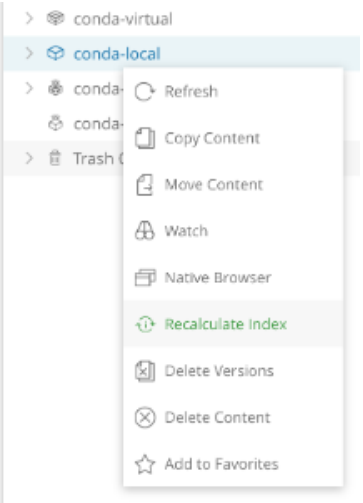
Viewing Individual Conda Package Information

Artifactory supports viewing selected Conda package metadata directly from the UI.
In the **Tree Browser**, select your virtual Conda repository and scroll down to find and select the package you want to inspect. The metadata is displayed in the **Conda Info** tab.



Reindexing a Conda Repository

You can trigger asynchronous reindexing of a local Conda repository either through the UI or using the REST API.
Through the UI, select your Conda repository in the **Tree Browser** and select **Recalculate Index** from the right-click menu as shown below. This requires Admin permissions.



To reindex a Conda repository through the REST API, please refer to Calculate Conda Repository Metadata.

Tuning Metadata Worker Threads

Conda Metadata Workers

Conda calculates metadata asynchronously based on repository storage events. The number of total worker threads that handle metadata calculation in parallel (specifically for Conda tasks) defaults to 5. In larger scales, you may modify this parameter by editing your $ARTIFACTORY_HOME/etc/artifactory.system.properties file and adding the following parameter:

```
artifactory.conda.metadata.calculation.workers=<NUMBER_OF_WORKERS>
```