Conan Repositories

Overview

Artifactory introduces advanced artifact management to the world of C/C++ through support for local repositories that work directly with the **Conan** client to manage Conan packages and dependencies. As a repository to which builds can be uploaded, and from which dependencies can be downloaded, Artifactory offers many benefits to C/C++ developers using Conan:

1. Secure, private repositories for C/C++ packages with fine-grained access control according to projects or development teams
2. Automatic layout and storage of C/C++ packages for all platforms configured in the Conan client
3. The ability to provision C/C++ dependencies from Artifactory to the Conan command line tool from local repositories.
4. Enterprise features such as high availability, repository replication for multi-site development, different options for massively scalable storage

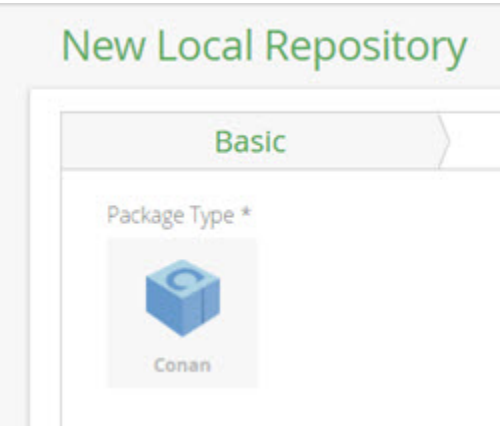
...and much more.

For more details on building Conan packages and working with the Conan client, please refer to the **Conan documentation**.

Configuration

Local Repositories

To enable calculation of C/C++ package metadata, set **Conan** to be the **Package Type** when you create your local repository.



Make sure to also select `conan-default` as the repository layout.

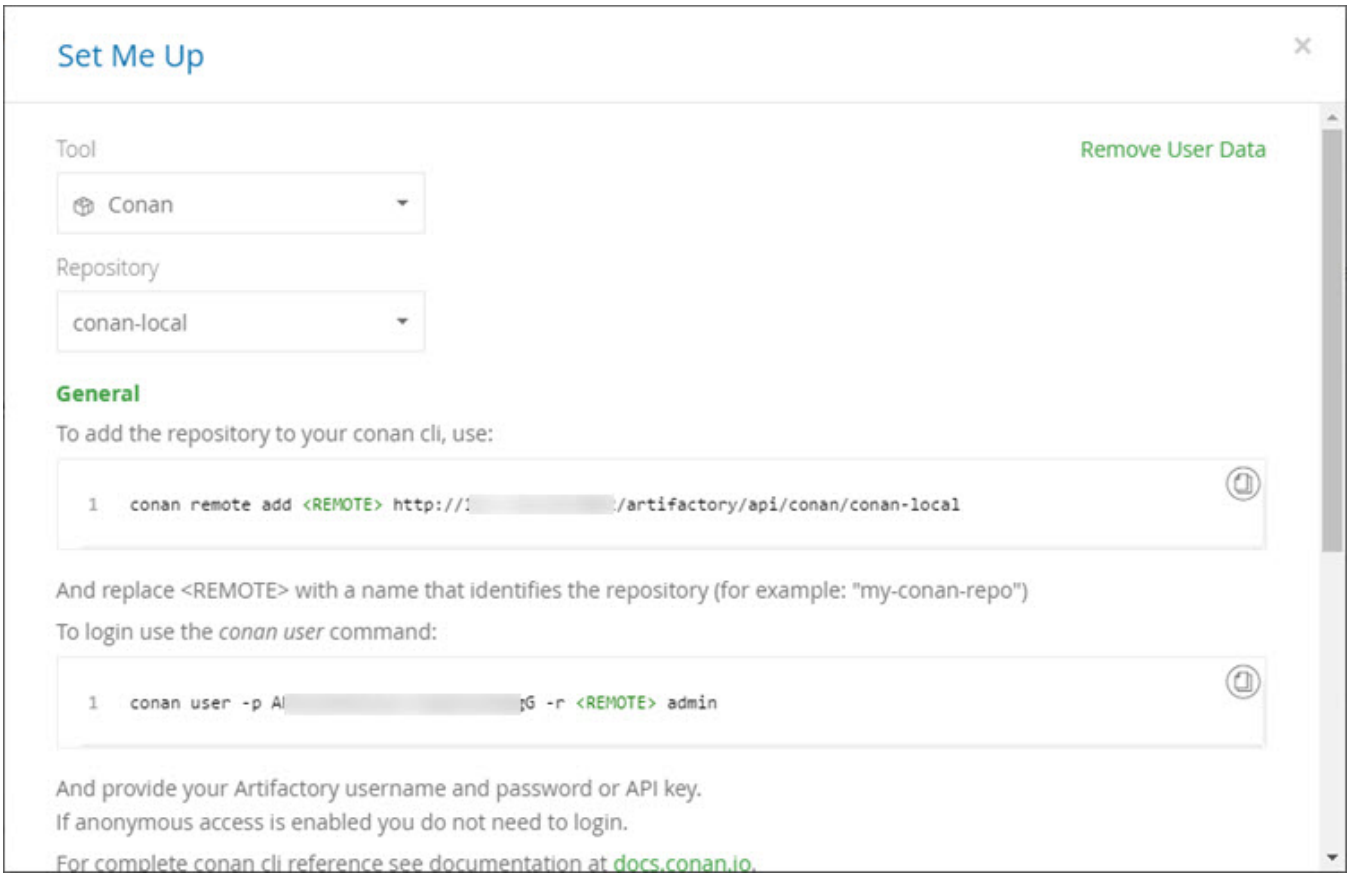
Using Conan with Artifactory

Once the Conan client is installed, you can access Conan repositories in Artifactory through its command line interface. You can only install packages from or export packages to your Artifactory local Conan repository using the Conan client.

Local vs. Remote

Don't let Conan terminology confuse you. For the purposes of this integration, the Conan "Remote" is actually the Artifactory local repository you have created for Conan packages.

Once you have created your Conan repository, select it in the Tree Browser and click **Set Me Up**, to see the code snippets you will need in order to use your repository as a source to install packages and as a target for export.



In the sections below, <REMOTE> is used to denote the logical name you set with which the Conan client can identify the Conan local repository in Artifactory.

Adding Your Repository

To use your local repository with Conan, you first need to add it as a Conan "Remote" to the client as follows:

```
conan remote add <REMOTE> http://<ARTIFACTORY_URL>/api/conan/<REPO_KEY>
```

Where:

<REPO_KEY> is the repository key.

> **Conan repositories must be prefixed with api/conan in the path**
>
> When accessing a Conan repository through Artifactory, the repository URL must be prefixed with **api/conan** in the path. This applies to all Conan commands including `conan install`.
>
> For example, if you are using Artifactory standalone or as a local service, you would access your Conan repositories using the following URL:
>
> *http://localhost:8081/artifactory/**api/conan**/<repository key>*
>
> Or, if you are using Artifactory SaaS, the URL would be:
>
> *https://<server name>.jfrog.io/<server name>/**api/conan**/<repository key>*

Authenticating the Conan Client

To authenticate the Conan client to Artifactory you need to log in using:

```
conan user -p <PASSWORD> -r <REMOTE> <USERNAME>
```

Accessing Artifactory anonymously

If Artifactory is configured for anonymous access, you may skip authenticating the Conan client.

Installing Dependencies

To install dependencies from Artifactory as defined in your *conanfile.txt* file use:

```
conan install . -r <REMOTE>
```

Uploading Packages

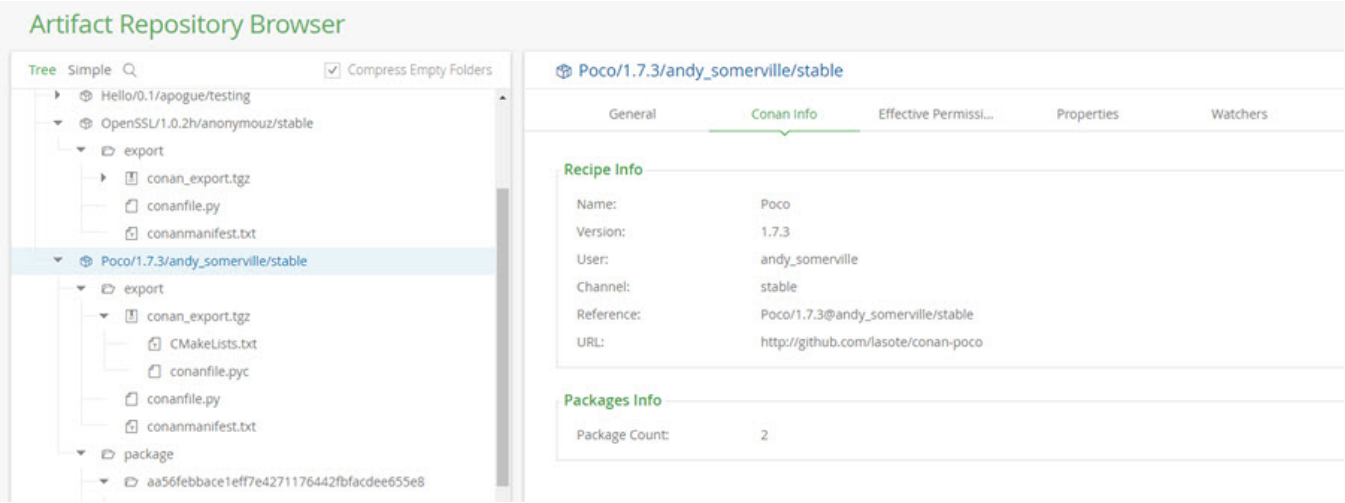
To upload packages to your Artifactory local Conan repository use:

```
conan upload <RECIPE> -r <REMOTE> --all
```

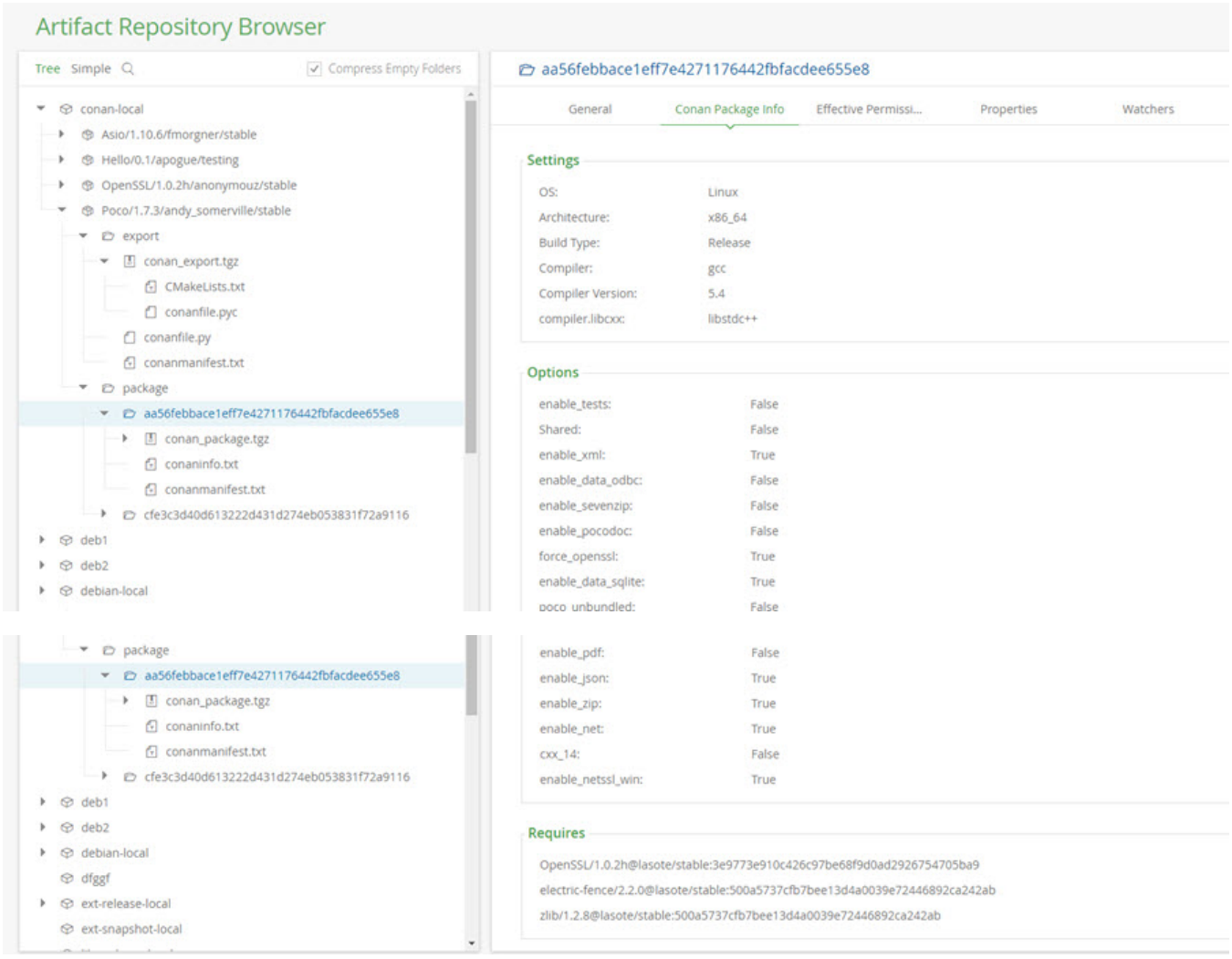
Where <RECIPE> specifies your Conan recipe reference formatted <NAME>/<VERSION>@<USER>/<CHANNEL>

Viewing Individual Conan Package Information

Artifactory lets you view selected metadata of a Conan package directly from the UI.
In the **Artifacts** tab, select **Tree Browser** and drill down to select the package file you want to inspect. The metadata is displayed in the **Conan Info** ta b. The specific information displayed depends on the tree item you have selected. Selecting the root item of a package displays details of the Conan recipe used to upload the package.



If you select one of the packages, you get detailed Conan Package info including **Settings**, **Options** and dependencies ("**Requires**")


Artifact Repository Browser
Tree
Simple
Compress Empty Folders
conan-local
Asio/1.10.6/fmorgner/stable
Hello/0.1/apogue/testing
OpenSSL/1.0.2h/anonymouz/stable
Poco/1.7.3/andy_somerville/stable
export
conan_export.tgz
CMakeLists.txt
conanfile.pyc
conanfile.py
conanmanifest.txt
package
aa56febbace1eff7e4271176442fbfacdee655e8
conan_package.tgz
conaninfo.txt
conanmanifest.txt
cfe3c3d40d613222d431d274eb053831f72a9116
deb1
deb2
debian-local
dfggf
ext-release-local
ext-snapshot-local
aa56febbace1eff7e4271176442fbfacdee655e8
General
Conan Package Info
Effective Permissi...
Properties
Watchers
Settings
OS: Linux
Architecture: x86_64
Build Type: Release
Compiler: gcc
Compiler Version: 5.4
compiler.libcxx: libstdc++
Options
enable_tests: False
Shared: False
enable_xml: True
enable_data_odbc: False
enable_sevenzip: False
enable_pocodoc: False
force_openssl: True
enable_data_sqlite: True
poco_unbundled: False
enable_pdf: False
enable_json: True
enable_zip: True
enable_net: True
cxx_14: False
enable_netssl_win: True
Requires
OpenSSL/1.0.2h@lasote/stable:3e9773e910c426c97be68f9d0ad2926754705ba9
electric-fence/2.2.0@lasote/stable:500a5737cfb7bee13d4a0039e72446892ca242ab
zlib/1.2.8@lasote/stable:500a5737cfb7bee13d4a0039e72446892ca242ab