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.

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

Artifactory Community Edition for C/C++

Conan repositories are available in Artifactory CE.

> Learn more

From Xray 3.21.2 and above, Xray can scan Conan packages, for more information see Conan and C/C++ Support in Xray.

Conan cheat sheet, the C/C++ Package Manager

Page Contents

- Overview
- Configuration
  - Local Repositories
  - Remote Repositories
  - Virtual Repositories
- Using Conan with Artifactory
  - Adding Your Repository
  - Authenticating the Conan Client
  - Allowing Anonymous Access
  - Installing Dependencies
  - Uploading Packages
- Viewing Individual Conan Package Information
- Conan V2 Package Support
  - Conan Package V1 Backward Compatibility
  - System Requirements
  - Revision Indexing
  - Viewing Individual Conan V2 Package Information
  - Viewing Conan Package Revisions

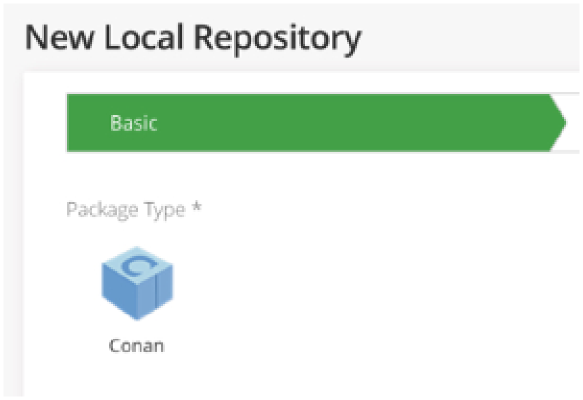
Integration Benefits

JFrog Artifactory and Conan Repositories

Configuration

Local Repositories

To enable calculation of C/C++ package metadata, from the **Administration** module, select **Repositories | Repositories | Local** and set **Conan** to be the **Package Type** when you create your local repository.



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

Remote Repositories

Deprecation Notice

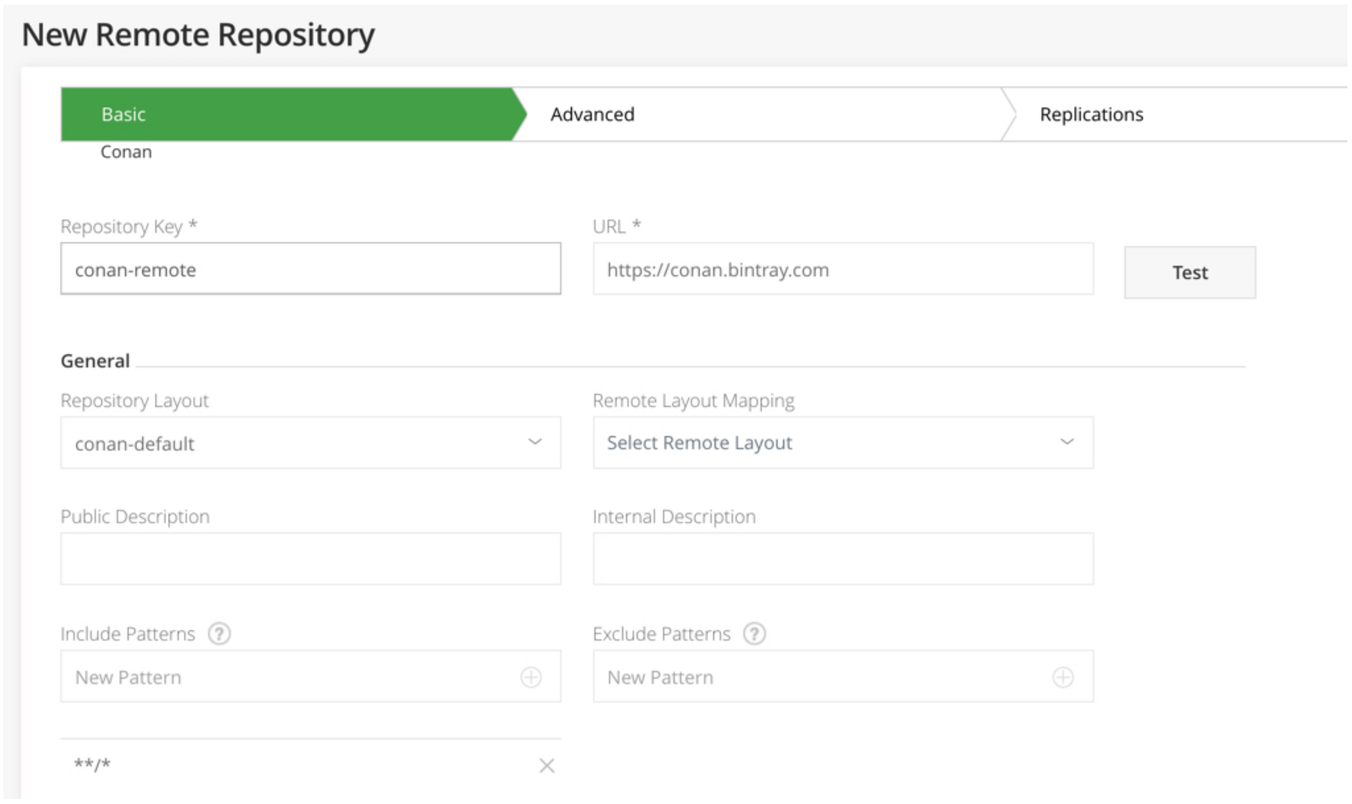
JFrog Bintray is being sunset. Please refer this blog post for more detail.

A Remote Repository defined in Artifactory serves as a caching proxy for a registry managed at a remote URL such as https://center.conan.io, or even a Conan repository managed at a remote site by another instance of Artifactory.

Conan packages requested from a remote repository are cached on demand. You can remove Conan packages from the remote repository cache, however, you can not manually push Conan files to a remote Conan repository.

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

1. In the **Administration** module, under **Repositories | Repositories | Remote**, click **New Remote Repository**.
2. In the New Repository dialog, set the **Package Type** to **Conan**, 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 * conan-remote
URL * https://conan.bintray.com
Repository Layout: conan-default
Remote Layout Mapping: Select Remote Layout
Public Description
Internal Description
Include Patterns: New Pattern
Exclude Patterns: New Pattern
**/*

Resolving Conan Remote Packages

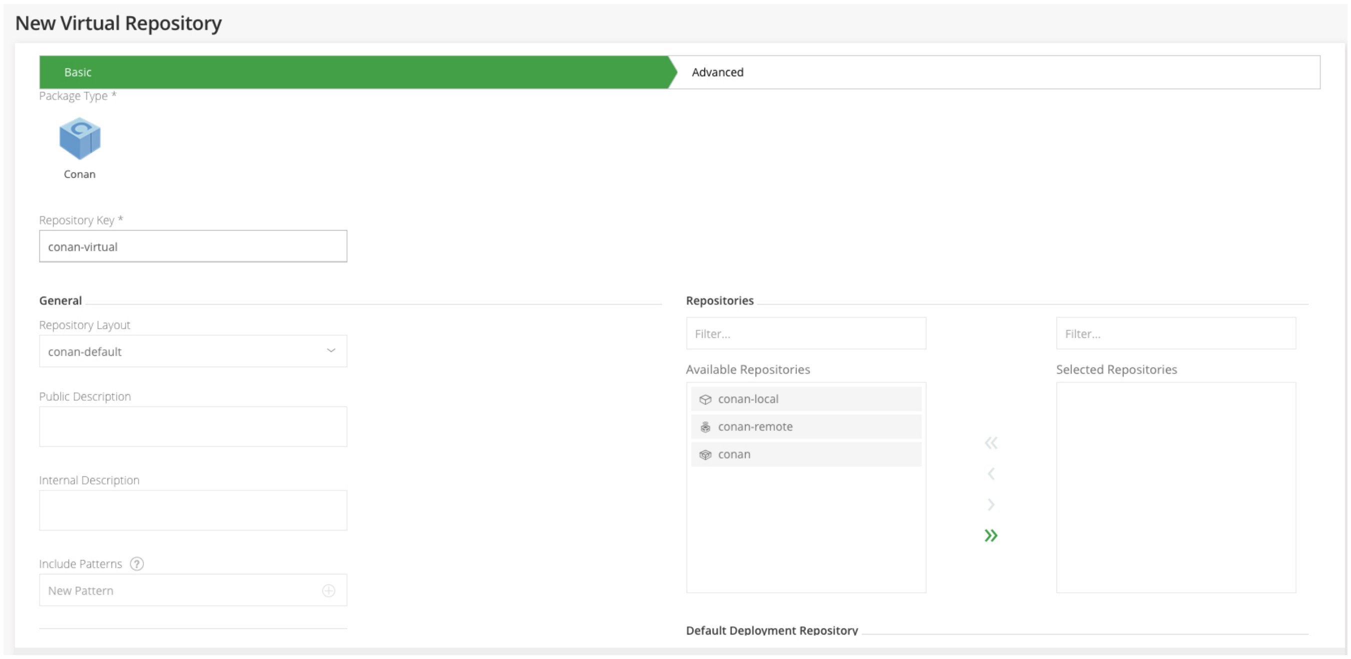
To resolve Conan remote packages, aggregate the remote repository in a virtual repository as they cannot be resolved directly from the remote repositories.

Virtual Repositories

A Virtual Repositories defined in Artifactory aggregates Conan packages from both local and remote repositories that are included in the virtual repositories. Using virtual repositories can be very useful since users will continue to work with the virtual repository while the admin can manage the included repositories, replace the default deployment target and those changes will be transparent to the users.

To define a virtual Conan repository follow these steps:

1. Create a new Virtual Repository, in the **Administration** module, under **Repositories | Repositories | Virtual**, click **New Virtual Repository** and set **Conan** as the **Package Type**.
2. Set the **Repository Key** value.
3. Select the underlying local and remote Conan repositories to include under the **Repositories** section.
4. You can optionally also configure your **Default Deployment Repository**


New Virtual Repository

Package Type *

Conan

Repository Key *

conan-virtual

General

Repository Layout

conan-default

Public Description

Internal Description

Include Patterns

New Pattern

Repositories

Filter...

Filter...

Available Repositories

conan-local

conan-remote

conan

Selected Repositories

Default Deployment Repository

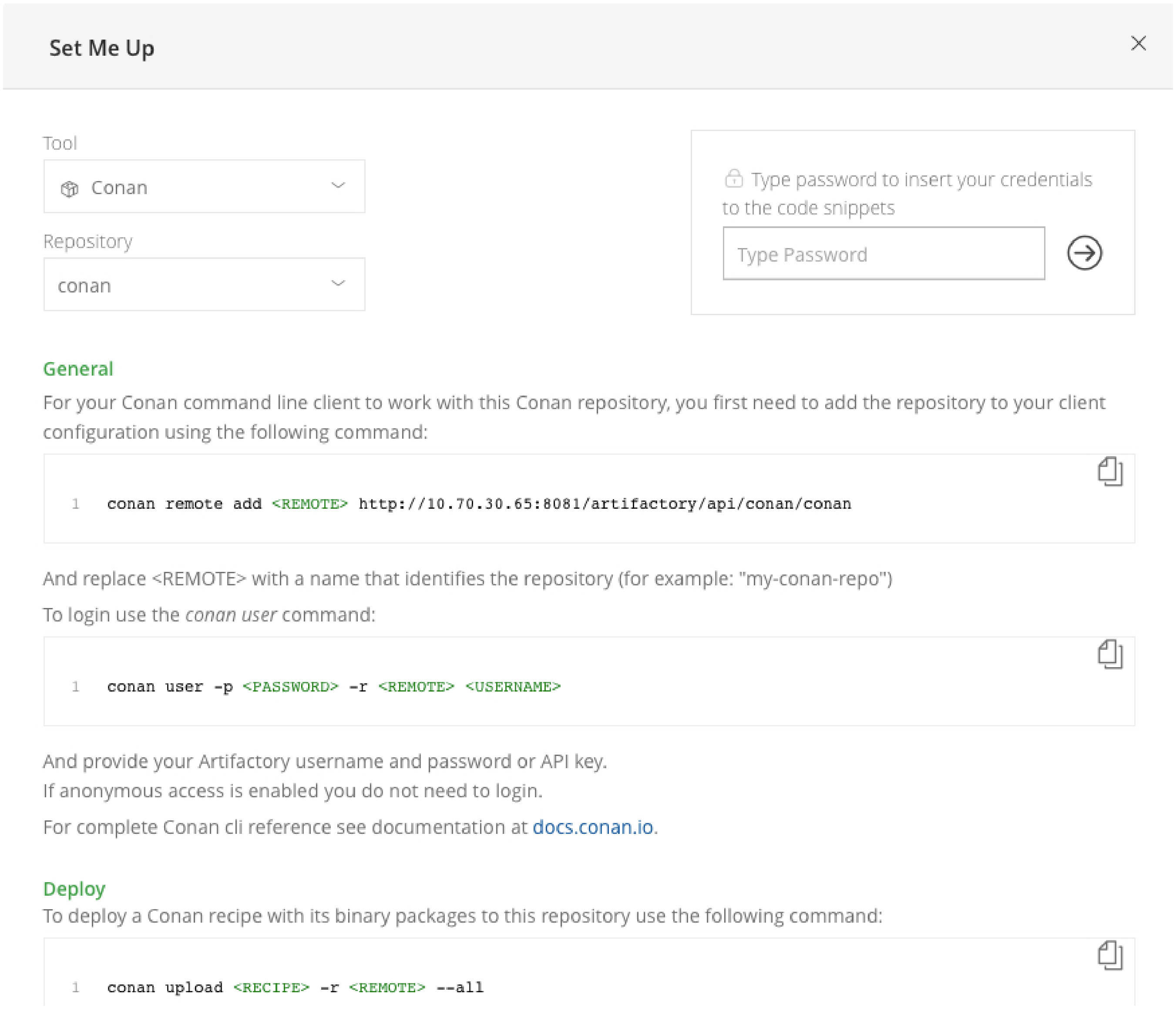
Using Conan with Artifactory

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 view in the **Application** module, **Artifactory** | **Artifacts** tab, 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.


Set Me Up
Tool
Conan
Repository
conan
Type password to insert your credentials to the code snippets
Type Password

General

For your Conan command line client to work with this Conan repository, you first need to add the repository to your client configuration using the following command:

```
conan remote add <REMOTE> http://10.70.30.65:8081/artifactory/api/conan/conan
```

And replace <REMOTE> with a name that identifies the repository (for example: "my-conan-repo")

To login use the *conan user* command:

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

And provide your Artifactory username and password or API key.
If anonymous access is enabled you do not need to login.

For complete Conan cli reference see documentation at docs.conan.io.

Deploy

To deploy a Conan recipe with its binary packages to this repository use the following command:

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

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 Cloud, the URL would be:

*`https://<server name>.jfrog.io/artifactory/`**`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

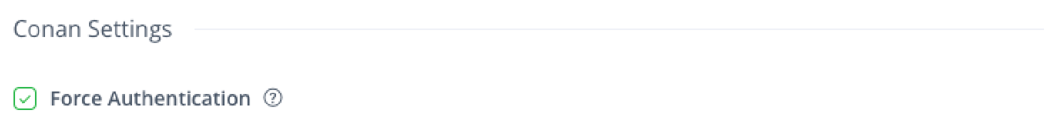
If Artifactory is configured for anonymous access, you may skip authenticating the Conan client.

Allowing Anonymous Access

Artifactory supports Conan repositories with **Allow Anonymous Access** enabled.

When **Allow Anonymous Access** is enabled, Artifactory will not query the Conan client for authentication parameters by default, so you need to indicate to Artifactory to request authentication parameters in a different way.

You can override the default behavior by setting the **Force Authentication** checkbox in the **New or Edit Repositor**y dialog.


Conan Settings

✓ Force Authentication

When set, Artifactory will first request authentication parameters from the Conan client before trying to access this repository.

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:

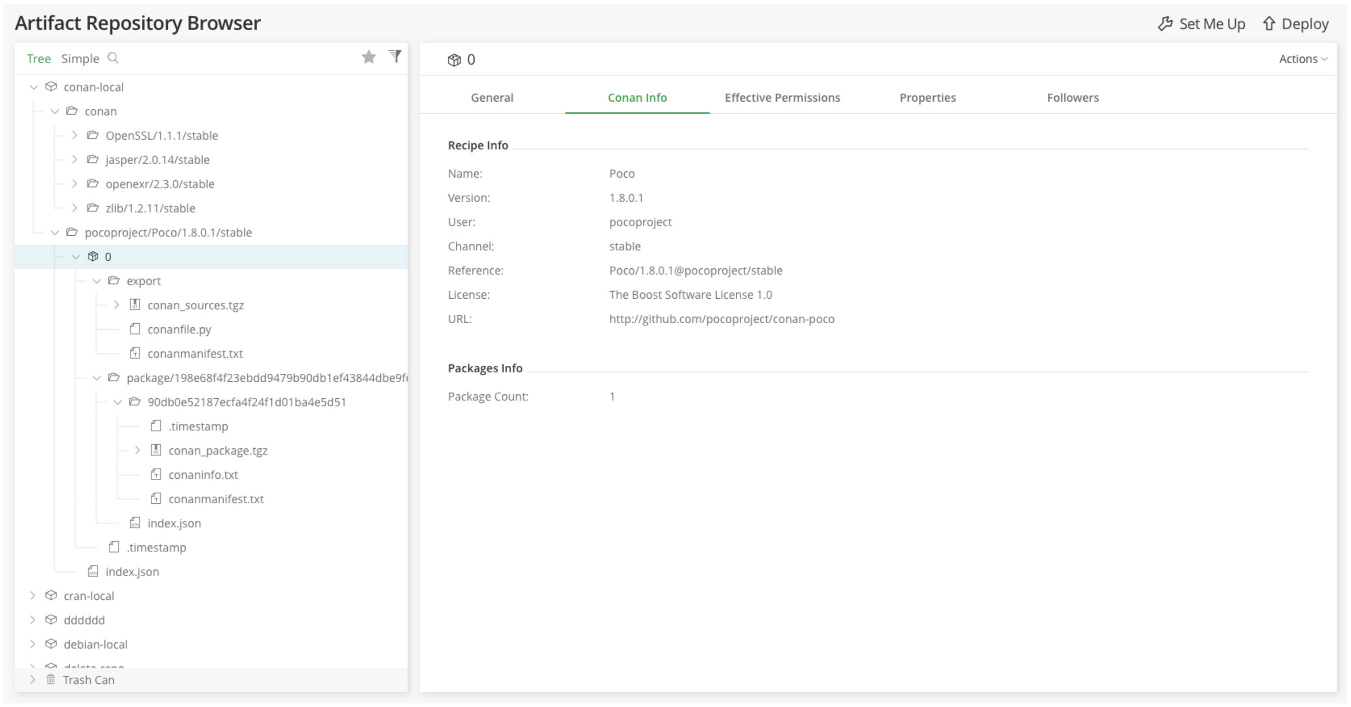
```
 conan upload <RECIPE> -r <REMOTE> --all
```

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 **Application** module, **Artifactory** | **Artifacts** tab, **Tree Browser**, and drill down to select the package file you want to inspect. The metadata is displayed in the **Conan Info** tab. 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.



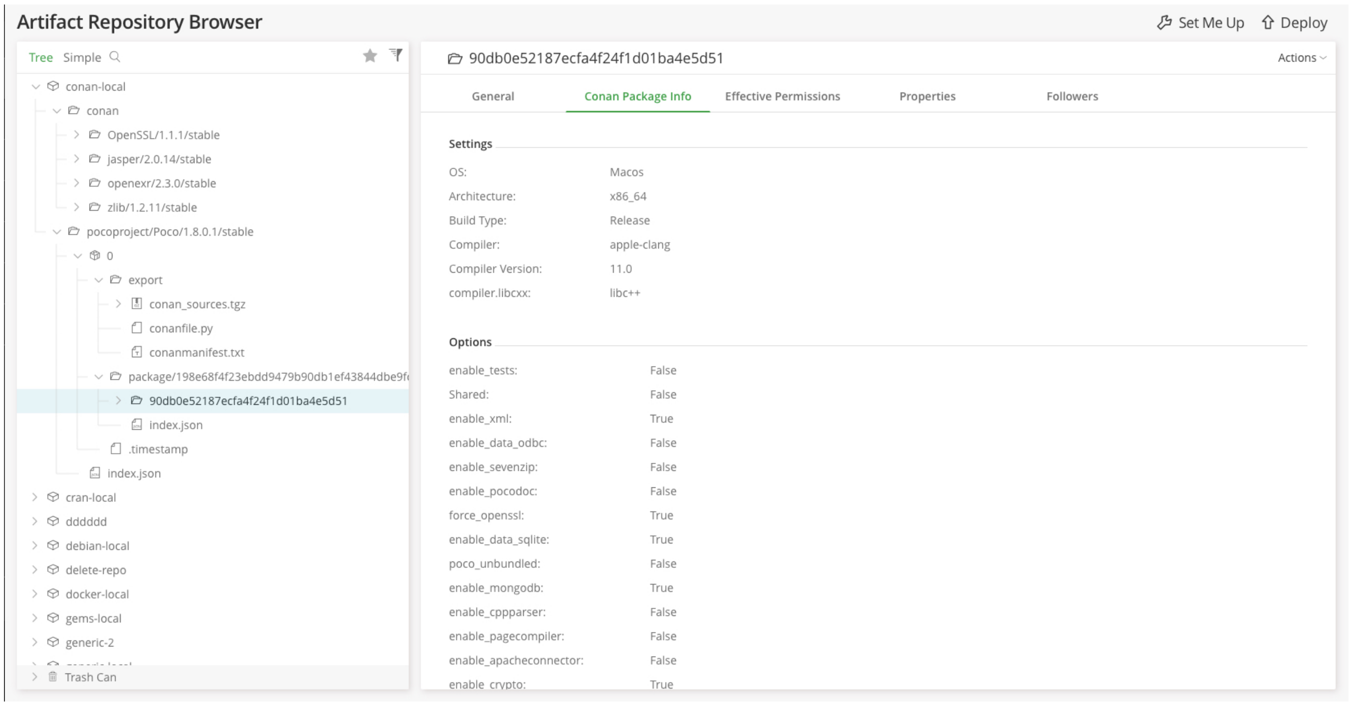
Recipe Info

Name:	Poco
Version:	1.8.0.1
User:	pocoproject
Channel:	stable
Reference:	Poco/1.8.0.1@pocoproject/stable
License:	The Boost Software License 1.0
URL:	http://github.com/pocoproject/conan-poco

Packages Info

Package Count:	1

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



Settings

OS:	Macos
Architecture:	x86_64
Build Type:	Release
Compiler:	apple-clang
Compiler Version:	11.0
compiler.libcxx:	libc++

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_mongodb:	True
enable_cppparser:	False
enable_pagecompiler:	False
enable_apacheconnector:	False
enable_crypto:	True

Conan V2 Package Support

Conan server API v2 is supported and introduces an extension of the binary layout to support Conan Package revisions. Revisions allow you to change your artifacts while keeping the same Conan reference and is intended to achieve package immutability, by preventing data from being overwritten on the server.
This example shows the layout with "9999" as the <packageId>, "1" is the Recipe Revision, and "4" is the Package Revision.

```
user/lib/1.0/channel/1/package/9999/4/*
```

Revisions Support By Conan Client

When the Revision features is enabled, the Conan client searches for the latest revision in Artifactory per reference, unless specified otherwise by the user. It is not mandatory to upgrade your conan client version to use Artifactory 6.9 however if you want to work with revisions you need to download use Conan client 1.13 with the revision mode enabled.

The Conan client, by default, searches for the latest revision in Artifactory per reference, unless specified otherwise by the user.

Conan Package V1 Backward Compatibility

Artifactory 6.9.0 provides backward compatibility for packages created with Conan server API v1 by automatically migrating the Conan server API v1 binary layout to the new format.
Migration to Conan server API V2 starts after the upgrade process is complete (in all nodes in case of High Availability) and all Conan API endpoints are blocked and cannot be accessed.
After migration, the default revision for all Conan server API v1 packages is set to "0" for both Recipe revision and Package revision, and endpoints will be accessible again.

By default, two threads are dedicated for the migration job. Before upgrading from versions previous to Artifactory 6.9, you can modify this setting in the `$JFROG_HOME/artifactory/etc/artifactory.system.properties` file (`$JFROG_HOME/artifactory/var/etc/artifactory/artifactory.system.properties` if modifying 7.x).
Note that you can allocate more threads during the migration process but need to restart Artifactory.

```
artifactory.conan.v2.migration.job.queue.workers = 2 (default)
```

System Requirements

- Artifactory version 6.9.0 and above.
- Artifactory Pro license and above or Artifactory CE for C/C++.
- Conan Client version 1.13.0 and above with the Revisions mode enabled. To enable the Revision mode on the Conan Client, see Conan Client.

Revision Indexing

Use the Conan client to deploy your packages. The Conan client will by default request the latest revision of a Conan reference requested.
Upon deployment using the Conan client, a .timestamp file is created under each revision root for each Recipe and Package revision root.
This file contains the epoch time of deployment (in milliseconds, for example, 1547984992855) and is created only when using the Conan client.

Only Use the Conan Client to deploy Conan Packages

Do not deploy the Conan packages in the UI or via REST API deployment to prevent index consistency and failed resolutions.

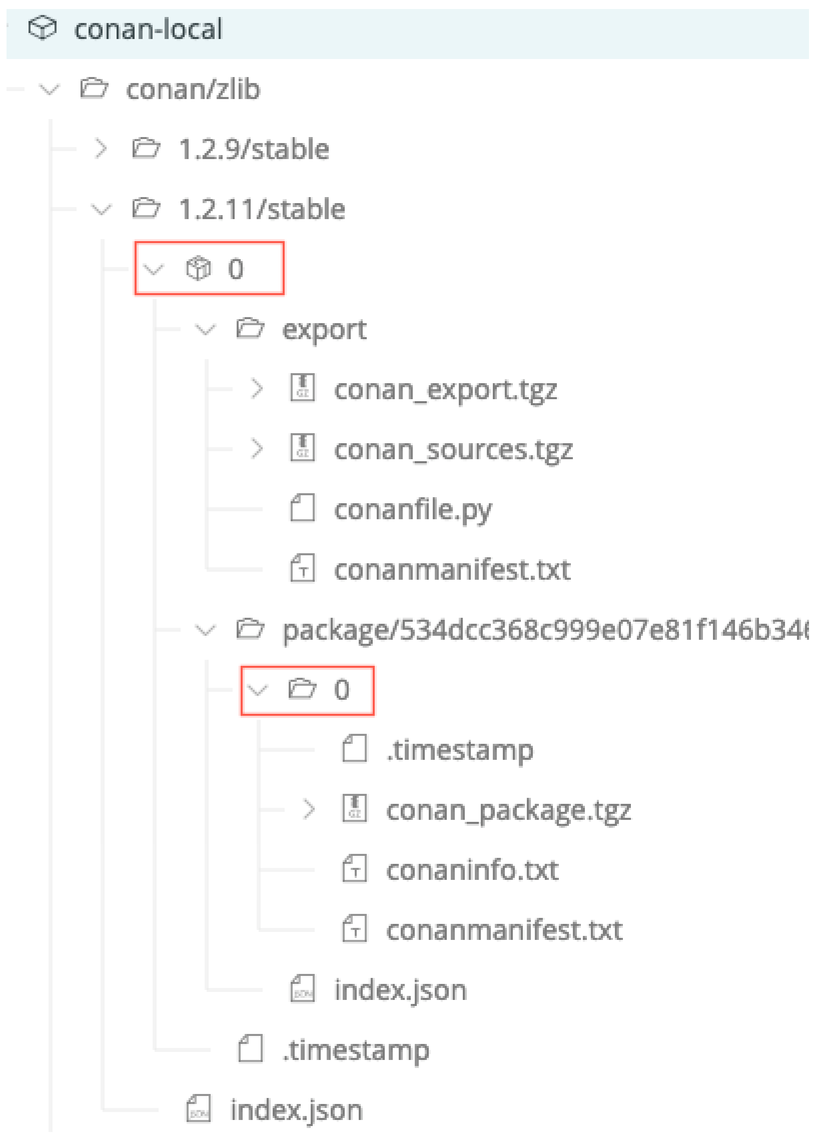
Viewing Individual Conan V2 Package Information

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** tab. 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.

Viewing Conan Package Revisions

From Artifactory 6.9.0, Conan v2 is supported and introduces a new Conan format layout to support the Revisions attribute.

The following example shows a package with the default revision "0 "for both Recipe and Package.



```
conan-local
  conan/zlib
    1.2.9/stable
    1.2.11/stable
      0
        export
          conan_export.tgz
          conan_sources.tgz
          conanfile.py
          conanmanifest.txt
        package/534dcc368c999e07e81f146b34
          0
            .timestamp
            conan_package.tgz
            conaninfo.txt
            conanmanifest.txt
          index.json
        .timestamp
      index.json
```