Cargo Package Registry

Overview

From JFrog Artifactory 7.17.4, the Cargo registry is supported for the Rust programming language, giving you full control of your deployment and resolving Cargo packages. Cargo downloads your Rust package's dependencies, compiles your packages, makes distributable packages, and uploads them to crates.io, the Rust community's package registry. You can contribute to this book on GitHub.

About Rust Programming Language

Rust is a programming language designed for performance and safety, with an emphasis on safety concurrency. A crate is a compilation unit in Rust. Using Cargo, you can publish libraries on crates.io, organize large projects with a workspace, install binaries from crates.io and extend Cargo using custom commands.

Learn More

Cheatsheet for managing applications using Rust & Cargo

Cargo repositories in Artifactory offer the following benefits:

- Secure and private local Cargo repositories with fine-grained access control
- The ability to proxy remote Cargo resources and cache downloaded Cargo packages to keep you independent of the network and the remote resource
- Metadata calculation of the Cargo packages hosted in the Artifactory local repositories
- Version management: Archiving older versions of the packages uploaded to local repositories
- Source and binaries management

Supported Cargo Version

Artifactory supports Cargo version 1.49.0 and above.

Git Support for Cargo Repositories

As the Cargo Client requires the Cargo registry to be a Git repository, the following Git support has been applied in Artifactory:

- An internal `.git` folder exists for each Cargo repository to reflect the index `.git` files. This folder is recreated after every reindexing process.
- An internal `git` directory in the `Data` directory of Artifactory has been added for each Cargo repository. This is a local clone that will be recreated following each restart or repository init.

Page Contents

Integration Benefits

Cargo Registry

Configuration

Local Repositories

To enable calculation of Cargo metadata, in the Administration module, go to **Repositories** | **Repositories** | **Local** and select **Cargo** as the **Package Type** when you create your local repository.

Prerequisite

Prior to setting a local repository, you will need to configure a Custom Base URL for the Artifactory instance. For more information, see General Settings.

Local Repository Layout

You will need to maintain a specific path structure to manage the Cargo packages that are uploaded to Cargo local repositories.

Cargo Source packages are automatically uploaded by default to the relative path: `crates/{package_name}/{package_name}-{version}.crate`.

Yank and Un-yank Crates in Local Repositories

Artifactory supports yanking and un-yanking crates in local repositories.

Cargo Yank/Un-yank

```
cargo yank hello_world --vers 0.1.4 --token "Bearer (token)"
cargo yank hello_world --vers 0.1.4 --token "Bearer (token)" --undo
```

Yank/ Un-yank Smart Repositories

To synchronize yanking in Smart remote repositories/replications, the properties must be synced.

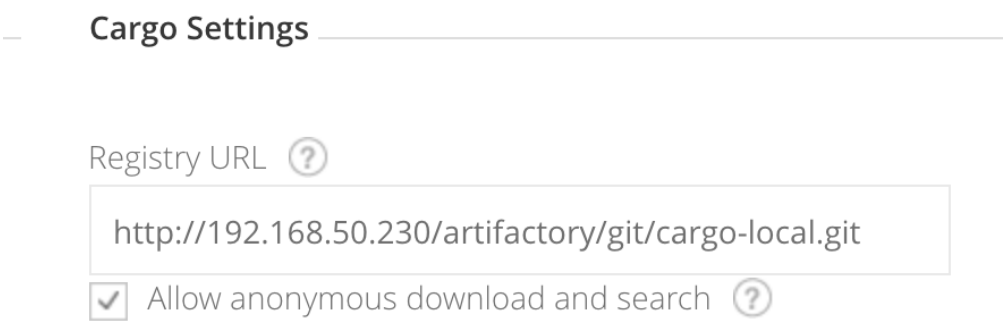
Remote Repositories

Prerequisite

Prior to setting a remote repository, you will need to configure a Custom Base URL for the Artifactory instance that is required to support. For more information, see General Settings.

You can create remote Cargo repositories to proxy and cache remote repositories or other Artifactory instances.
The **Registry URL** has been added to Cargo remote repositories, to reflect the index (git) location:

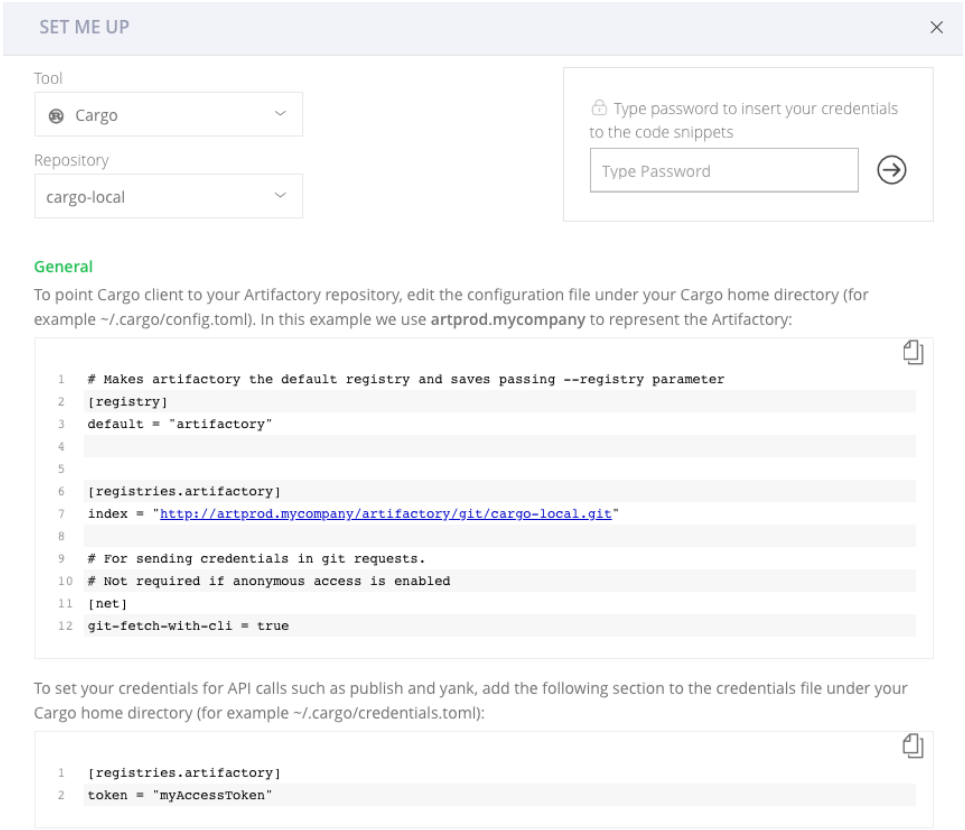
- For external repositories, the path should be the same as the URL. For example: `https://github.com/rust-lang/crates.io-index`
- For Smart remote repositories, the path should be `http(s)://art_url/artifactory/git/repo_name.git`. For example: `http://127.0.0.1/artifactory/git/cargo-local.git`.



Resolving Cargo Packages

To resolve Cargo packages:

1. In the Application module, navigate to **Artifactory | Artifacts**.
2. In the Artifact Tree Browser, select a Cargo repository and click **Set Me Up**.



3. Copy the Cargo code snippets to publish a Cargo package or to configure your Rust client to resolve the artifacts in the selected repository.

Resolving Cargo Packages Using the Rust Command Line

1. In the Application module, navigate to **Artifactory | Artifacts**.
2. In the Artifact Tree Browser, select a Cargo repository and click **Set Me Up**.
3. Install a package using the Cargo build or install commands.

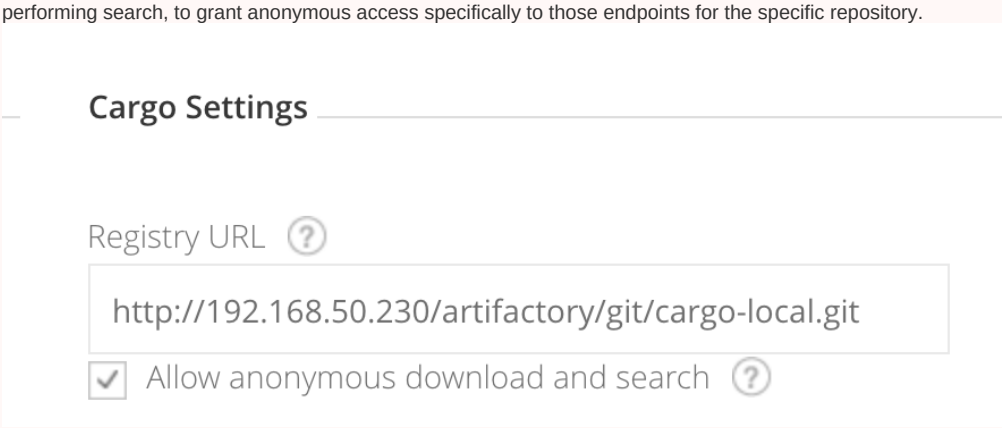
Cargo install
`cargo install crate`

Resolving Multiple Cargo Registries

To resolve multiple registries, add this optional flag `--registry (registryId)`.

Authentication: Allow Anonymous Downloads

The Cargo client does not send **any** authentication header when running install and search commands.
Select the "Allow anonymous download and search" to block anonymous requests but still allow anonymous Cargo client downloads and performing search, to grant anonymous access specifically to those endpoints for the specific repository.



Deploying Cargo Packages

You can deploy packages to a local Cargo repository using the Cargo Client, using the **Deploy** feature in the UI, or using a cURL request.

Deploying a Package Using the Cargo Client (Recommended)

To deploy a package, run the following Cargo `publish` command.

Cargo install
`cargo publish or cargo publish --registry (registry id)`

To override the credentials for that repository, run the following command.

Cargo install
`cargo publish --token "Basic (base64 of user:password)"or cargo publish --token "Bearer (access token)"`

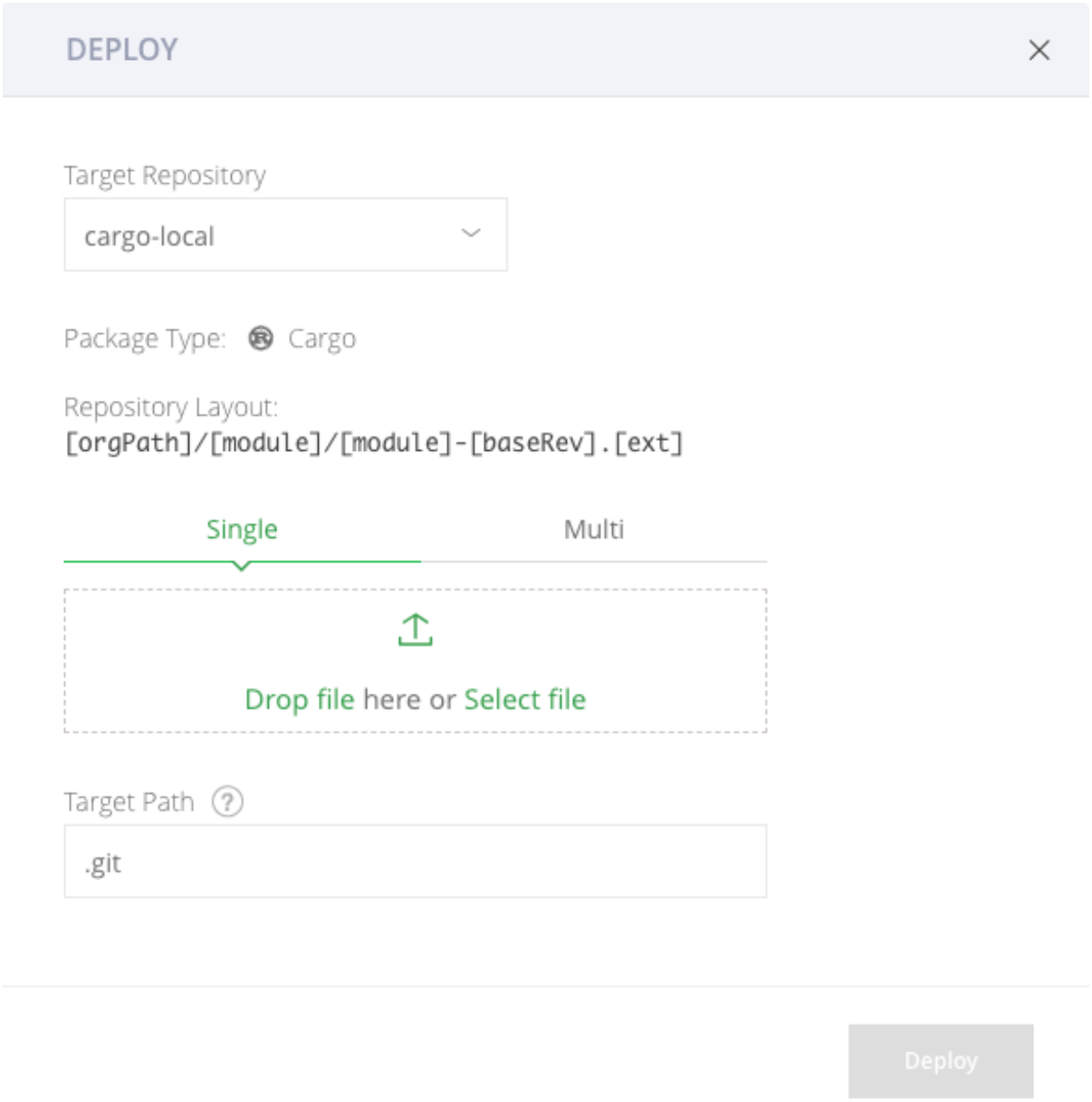
Deploying a Package Using the UI

You can either drag and drop, or select a Cargo package to upload in **Deploy** in the UI. Artifactory will automatically identify if the package is a source or binary package.

Target Path

It is important to be aligned with the following layout to support this feature.

```
crates/{package_name}/{package_name}-{version}.crate
```



Deploying a Package Using cURL

To deploy a package using a cURL request.

Deploying Cargo Crates
`curl -uadmin:password -XPUT "http://localhost:8082/artifactory/cargo-local/crates/package_1.0.0.crate" -T package_1.0.0.crate`

When deploying directly (PUT request to a specific path), make sure the target path is a valid Cargo path.

```
crates/{package_name}/{package_name}-{version}.crate
```

Note that deploying a package to a different path will not identify the package as Cargo packages, and will not invoke metadata indexing.

Viewing Individual Cargo Package Information

Artifactory lets you view selected metadata of a Cargo package directly from the UI.

In the Artifact Repository Browser, select your local Cargo repository and scroll down to find and select the package you want to inspect.

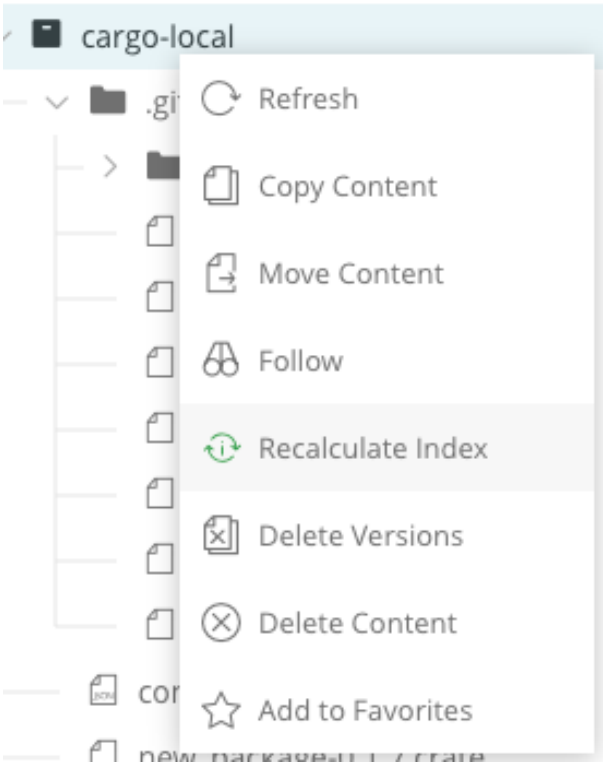
The metadata is displayed in the **Cargo Info** tab, or view in the **Packages** view.

Re-indexing a Cargo Repository

You can trigger an asynchronous re-indexing of a local Cargo repository either through the UI or using the REST API.

This will also reindex the git index and, as a result, will also index the remote repositories.

In the **Artifact Tree Browser**, select your Cargo repository, right-click and select **Recalculate Index** from the list. Requires Admin privileges.



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