JFROG CLI CHEAT SHEET

WHAT IS JFROG CLI?

JFrog CLI is an open-source project, written in Golang. It is a compact and smart client that provides a simple interface to automate access to JFrog products, such as JFrog Artifactory, Xray and Distribution.

CLI works with the JFrog Platform making your scripts more efficient and reliable by enabling parallel work, deployment, resolution and build info upload with simple and easy to use commands (running JFrog REST API behind the scenes). For example, with a single command you can upload a full directory to a repository in Artifactory, or download files from different Artifactory servers.

HOW TO DOWNLOAD JFROG CLI

JFrog CLI is available with the following installations types:

DEBIAN

```
...sudo apt install -y jfrog-cli
```

RPM

```
...yum install -y jfrog-cli
```

HOMEBREW

```
brew install jfrog-cli
```

CURL

```
curl -fL https:getcli.jfrog.io | sh
```

NPM

```
npm install -g jfrog-cli-go
```

DOCKER

```
docker run docker.bintray.io/...
```

GO

```
...go get github.com/jfrog.jfrog-cli...
```

CHOCOLATEY

```
choco install jfrog-cli
```

WHY USE JFROG CLI

1. JFrog CLI is a useful compact client, which was developed in order to enhance and simplify command-line interactions with JFrog products. Here are some examples of where it comes in handy with the JFrog Platform:
2. Securely protect your server. When JFrog CLI is configured to use username and password or API key, it automatically generates an access token to authenticate with Artifactory. The generated access token is valid only for one hour, and is refreshed automatically before it expires.
3. Customize repository cleanup. You can use JFrog CLI to delete artifacts that match specific file spec patterns, as well as query delete candidates using AQL (Artifactory Query Language).
4. Create a script and automate REST APIs. JFrog CLI allows you to automate all of the above and more, by using the upload, download, delete, move and copy commands; all of which can be enhanced using wildcards or regular expressions with placeholders. This gives you space to maneuver when looking for a specific solution.
5. Create build integration with any CI/CD server. JFrog CLI extends the build integration capability to any tool by integrating with any development ecosystem. This allows you to build from any tool, collect all relevant information, and deploy the build information along with the artifacts to Artifactory.
6. Back-up your file-systems. JFrog CLI can upload and download symlinks into your Artifactory repository, ensuring your linux file-system backups also include the configured symlinks, and that your entire system can be restored.

BUILD INTEGRATION EXAMPLE

JFrog CLI includes integration with different package types such as Maven, Gradle, Pypi, Docker and more.

The following commands will run a Maven build, resolve dependencies and deploy build artifacts from and to Artifactory, while collecting the build-info and storing it in Artifactory:

1. Configure the CLI connection to your Artifactory server(s)

```
$ jfrog rt c
```

2. Setting Maven repositories

```
$ jfrog rt mvn-config
```

3. Running maven build

```
$ jfrog rt mvn clean install -f path/to/pom-file --build-name test
--build-number 1.0.0
```

4. Publishing Build-Info

```
$ jfrog rt bp test 1.0.0
```

10 COMMONLY USED COMMANDS

Download command

```
$  jfrog rt dl <Repo_Name>/<file_name
```

For example, $ jfrog rt dl my-local-repo/cool-froggy.zip
Downloads the cool-froggy.zip file from my local repo. Use star (*) to download all files under a specific path or repo.

Upload command

```
$ jfrog rt u <file_name> <Repo_Name>
```

For example, $ jfrog rt u froggy.tgz my-local-repo
Uploads the file froggy to my-local-repo.

Running Curl

```
$ jfrog rt curl -XGET /api/build
```

Executes the cUrl client, to send a GET request to the /api/build endpoint to the default Artifactory server.

Scanning a publish builds

```
jfrog rt bs<Build_Name> <Build_Number>
```

For example, $ jfrog rt bs my-build-name 18

Promote a build

```
jfrog rt bpr <Build_Name> <Build_Number> <Target_repos>
```

For example, $ jfrog rt bpr my-build-name 18 target-repository

Creating users

```
$ jfrog rt users-create --csv path/to/users.csv
```

Creates new users according to details defined in the path/to/users.csv file.

Creating or updating an unsigned Release Bundle

```
$ jfrog rt rbc --spec=/path/to/<release_bundle_spec_json> <RB_Name> <number>
```

For example, $ jfrog rt rbc --spec=/path/to/rb-spec.json myApp 1.0.0
Creates and updates an unsigned Release Bundle on JFrog Distribution

Building Go Packages

```
$ jfrog rt go build
```

Builds Go packages using the Go client.

Creating Access Tokens

```
$ jfrog rt atc johnny-cache
```

Creates an access token for the user with the johnny-cache username.

Publishing Build-Info

```
$ jfrog rt bp <Build_Name> <Build_Number>
```

For example, $ jfrog rt bp my-build-name 12 will publish all build information to my-build-name number 12.
Publishes build information to Artifactory.