SBT Repositories

Overview

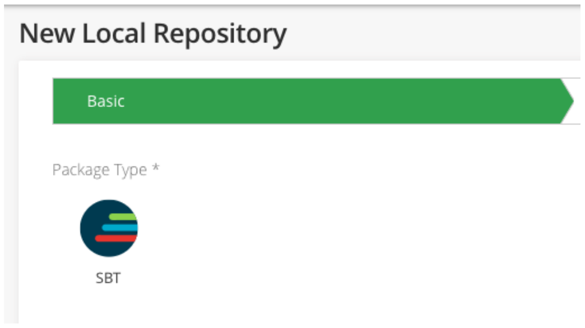
Artifactory provides integration with sbt, allowing you to configure it to resolve dependencies from, and deploy build output to sbt repositories. All you need to do is make minor modifications to your `build.sbt` configuration file.

Page Contents

Configuration

Local Repositories

A local sbt repository is used as a target to which you can deploy the output of your `build.sbt` script.
To create an sbt repository, from the **Administration** module, under **Repositories | Repositories | Local**, set the **Package Type** to **SBT**.

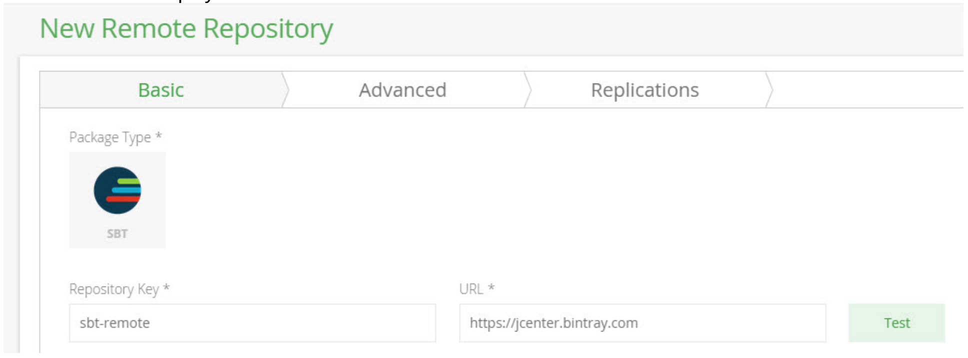


Remote Repositories

A remote repository defined in Artifactory serves as a caching proxy for a registry managed at a remote URL.

Artifacts (such as JAR files) requested from a remote repository are cached on demand. You can remove downloaded artifacts from the remote repository cache, however, you can not manually deploy artifacts to a remote SBT repository.

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

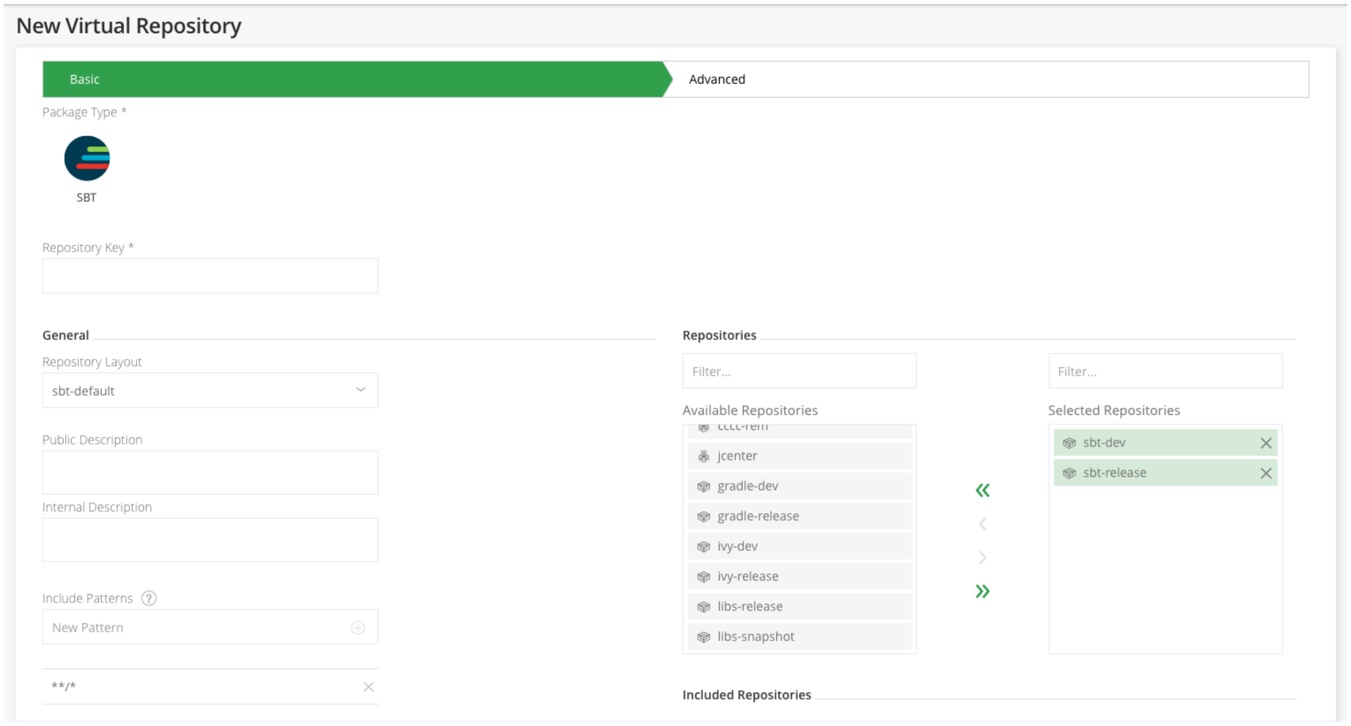
1. In the **Administration** module, under **Repositories | Repositories | Remote**, click **New Remote Repository**.
2. In the New Repository dialog, set the **Package Type** to **SBT**, set the **Repository Key** value, and specify the URL to the remote registry in the **URL** field as displayed below:


Repository Key *: sbt-remote
URL *: https://jcenter.bintray.com
3. Click **Save & Finish**.

The parameters needed to configure remote sbt repositories are identical to those used for Maven repositories. For more details, please refer to Remot e Repositories.

Virtual Repositories

A Virtual Repository defined in Artifactory aggregates packages from both local and remote repositories.
This allows you to access both locally hosted JARS and remote proxied sbt registries from a single URL defined for the virtual repository.
To define a virtual sbt repository, from the **Administration** module, go to **Repositories** | **Repositories** | **Virtual**, set the **Package Type** to sbt**,** and sel ect the underlying local and remote sbt repositories to include in the **Basic** settings tab.


New Virtual Repository

Basic | Advanced

Package Type *

SBT

Repository Key *

General

Repository Layout

sbt-default

Public Description

Internal Description

Include Patterns

New Pattern

**/*

Repositories

Filter...

Available Repositories

- cccc-rem
- jcenter
- gradle-dev
- gradle-release
- ivy-dev
- ivy-release
- libs-release
- libs-snapshot

Filter...

Selected Repositories

- sbt-dev
- sbt-release

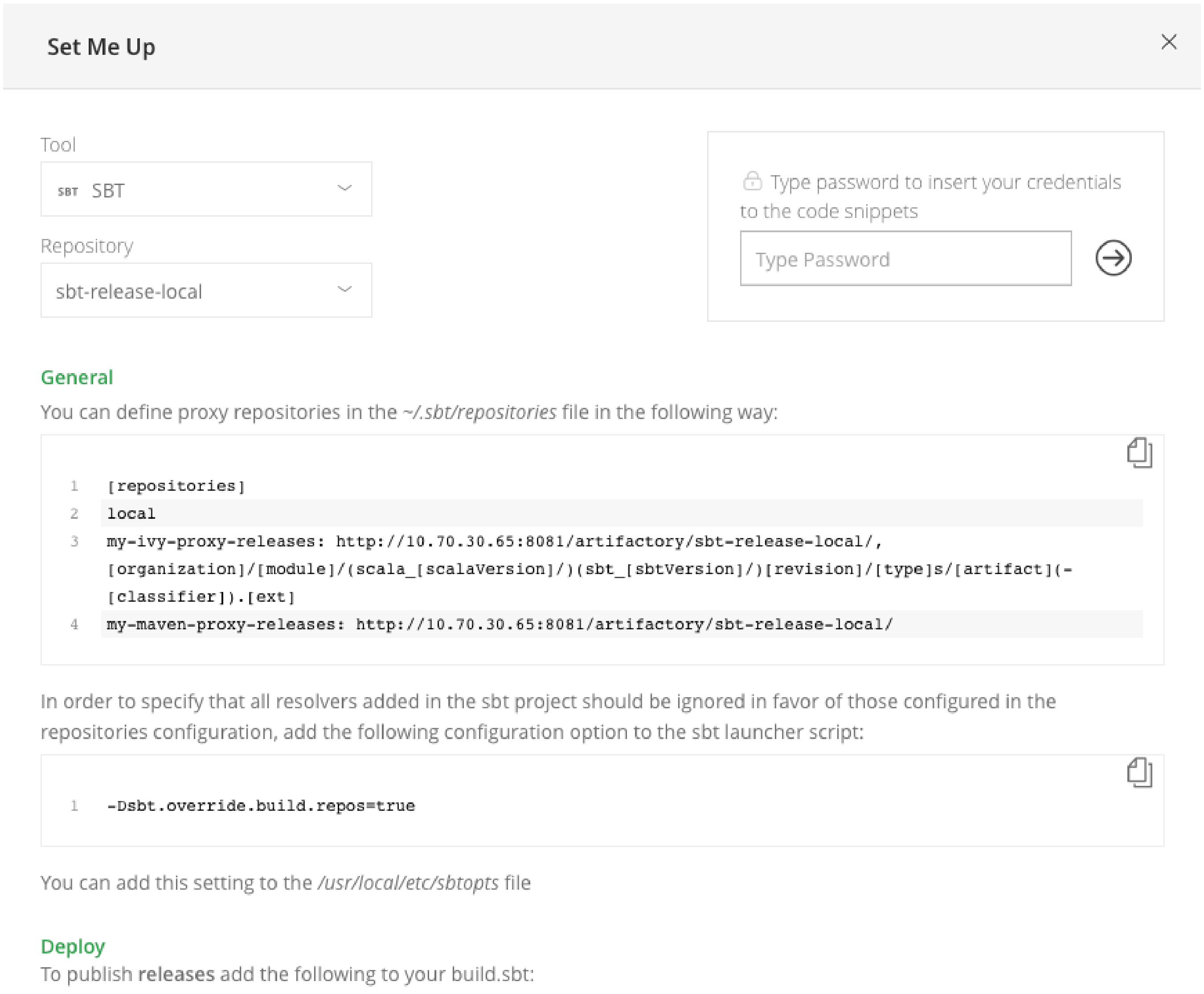
Included Repositories

Click **Save & Finish** to create the repository.

The parameters needed to configure virtual sbt repositories are identical to those used for Maven repositories. For more details, please refer to Virtual Repositories.

Configuring SBT

To configure sbt to resolve and deploy artifacts through sbt repositories defined in Artifactory, simply select one of the sbt repositories in the **Application** module, go to | **Artifactory** | **Artifacts** | **Artifact Repository Browser** and click **Set Me Up**. Artifactory will display code snippets you can use in the relevant sbt files.


Set Me Up
Tool
SBT
Repository
sbt-release-local
Type password to insert your credentials to the code snippets
Type Password

General

You can define proxy repositories in the *~/.sbt/repositories* file in the following way:

```
[repositories]
local
my-ivy-proxy-releases: http://10.70.30.65:8081/artifactory/sbt-release-local/,
[organization]/[module]/(scala_[scalaVersion]/)(sbt_[sbtVersion]/)[revision]/[type]s/[artifact](-
[classifier]).[ext]
my-maven-proxy-releases: http://10.70.30.65:8081/artifactory/sbt-release-local/
```

In order to specify that all resolvers added in the sbt project should be ignored in favor of those configured in the repositories configuration, add the following configuration option to the sbt launcher script:

```
-Dsbt.override.build.repos=true
```

You can add this setting to the */usr/local/etc/sbtopts* file

Deploy

To publish **releases** add the following to your build.sbt:

Configuring Proxy Repositories

To configure a repository defined in Artifactory as a proxy repository for sbt, add the code snippet below to your `~/.sbt/repositories` file (`C:\Users\%USERNAME%\.sbt\repositories` on Windows).

```
[repositories]
local
my-ivy-proxy-releases: http://<host>:<port>/artifactory/<repo-key>/, [organization]/[module]/(scala_
[scalaVersion]/)(sbt_[sbtVersion]/)[revision]/[type]s/[artifact](-[classifier]).[ext]
my-maven-proxy-releases: http://<host>:<port>/artifactory/<repo-key>/
```

Where `<host>:<port>` are the host URL and port on which Artifactory is running.

For example, if you are running Artifactory on your local machine, on port 8081, and want to proxy Ivy repositories through a repository called `sbt-ivy-proxy`, and proxy Maven repositories through a repository called `sbt-maven-proxy` you would use:

```
[repositories]
local
my-ivy-proxy-releases: http://localhost:8081/artifactory/sbt-ivy-proxy/, [organization]/[module]/(scala_
[scalaVersion]/)(sbt_[sbtVersion]/)[revision]/[type]s/[artifact](-[classifier]).[ext]
my-maven-proxy-releases: http://localhost:8081/artifactory/sbt-maven-proxy/
```

Proxying Maven and Ivy repositories separately

We recommend using different repositories to proxy Maven and Ivy repositories as a best practice described in Proxying Ivy Repositoriesin the SBT Reference Manual.

To specify that all resolvers added in the sbt project should be ignored in favor of those configured in the repositories configuration, add the following configuration option to the sbt launcher script:

```
-Dsbt.override.build.repos=true
```

You can also add this setting to your */usr/local/etc/sbtopts* (*C:\Program Files (x86)\sbt\conf\sbtopts* on Windows)

For more details on sbt proxy repositories, please refer to Proxy Repositories in the SBT Reference Manual.

Configuring Artifact Resolution

To resolve artifacts through Artifactory, simply add the following code snippet to your *build.sbt* file:

```
resolvers += "Artifactory" at "http://<host>:<port>/artifactory/<repo-key>/"
```

Where `<host>:<port>` are the host URL and port on which Artifactory is running, and repo-key is the Artifactory repository through which you are resolving artifacts

Deploying Artifacts

To deploy SBT build artifacts to repositories in Artifactory, add the following code snippets to your `build.sbt` file.

For **releases,** add:

```
publishTo := Some("Artifactory Realm" at "http://<host>:<port>/artifactory/<repo-key>")
credentials += Credentials("Artifactory Realm", "<host>", "<USERNAME>", "<PASS>")
```

For **snapshots,** add:

```
publishTo := Some("Artifactory Realm" at "http://<host>:<port>/artifactory/<repo-key>;build.timestamp=" +
new java.util.Date().getTime)
credentials += Credentials("Artifactory Realm", "<host>", "<USERNAME>", "<PASS>")
```

Where `<host>:<port>` are the host URL and port on which Artifactory is running, and repo-key is the Artifactory repository to which you are deploying artifacts.

Sample Project

A sample sbt project that uses Artifactory is available on GitHub and can be freely forked.