Spring Cloud Skipper Reference Guide

Mark Pollack, Ilayaperumal Gopinathan, Janne Valkealahti, Gunnar Hillert,
Sabby Anandan, Vinicius Carvalho, Jay Bryant

Table of Contents

Preface

Chapter 1. About the Documentation

The documentation for this release is available in HTML.

The latest copy of the Spring Cloud Skipper reference guide can be found here.

Chapter 2. Getting Help

If you are having trouble with Spring Cloud Skipper, we would like to help!

- Ask a question. We monitor stackoverflow.com for questions tagged with `spring-cloud-skipper`.
- Reach out to us on gitter.
- Report bugs with Spring Cloud Skipper at github.com/spring-cloud/spring-cloud-skipper/issues.

All of Spring Cloud Skipper is open source, including the documentation! If you find problems with the docs or if you want to improve them, please get involved.

Spring Cloud Skipper Overview

Skipper is a lightweight tool that lets you discover Spring Boot applications and manage their lifecycle on multiple Cloud Platforms. You can use Skipper standalone or integrate it with Continuous Integration pipelines to help implement the practice of Continuous Deployment.

Skipper consists of a server application that exposes an HTTP API. A shell application provides easy-to-use commands to interact with the server. The server uses a relational database to store state. Documentation to call the HTTP API is available in the REST API Guide.

Applications in Skipper are bundled as packages that contain a templated configuration file and a default set of values that are used to fill in the template. You can override these defaults when installing or upgrading a package. Skipper provides a means to orchestrate the upgrade/rollback procedure of applications between different versions, taking the minimal set of actions to bring the system to the desired state.

Skipper's design is influenced by a large number of projects in the Kubernetes ecosystem that perform resource templating and/or orchestration, hence the nautically inspired project name Skipper. In particular, Helm's approach to present the user with a familiar `apt-get` or `brew` like installation experience was a big influence.

Chapter 3. Features

The main features are:

- Define multiple platform accounts where Spring Boot applications can be deployed. Supported platforms are Local, Cloud Foundry, and Kubernetes.
- Substitute variables in Mustache-templated files that describe how to deploy applications to a platform.
- Search Package Repositories for existing applications.
- Upgrade/Rollback a package based on a simple workflow.
- Store the history of resolved template files (AKA 'application manifests') that represent the final description of what has been deployed to a platform for a specific release.
- Use a standalone interactive shell or an HTTP API.

Chapter 4. Concepts

The main concepts are **Platforms**, **Packages**, **Repositories**, **Releases**, and **Release Workflows**.

Platforms are where your apps run. Skipper 1.0 supports deploying applications to platforms by using the Spring Cloud Deployer family of libraries. Doing so lets Skipper deploy Spring Boot applications to Cloud Foundry, Kubernetes, and your local machine. You can configure a single Skipper server to deploy to multiple platforms, with each platform identified by a unique name.

The Spring Cloud Deployer libraries for Apache YARN, Apache Mesos, Redhat Openshift, and Hashicorp Nomad were not bundled with Skipper in 1.0. Donovan Muller has provided support for Redhat Openshift.

Packages define the basic recipe for describing what to install on a platform. A package can define a single application or it can define a group of applications. It contains descriptive metadata, the location of the Spring Boot uber jar, and default application or deployment properties. The location of the uber jar can be a Maven repository, docker registry, file location, or HTTP location. A package is a collection of YAML files that are zipped up into a file with a naming convention of `name-version.zip` (for example: `myapp-1.0.3.zip`).

Repositories are where package metadata and zip files are hosted. Repositories can either be 'local' or 'remote'. A remote repository is one that is only accessible over HTTP. Any arbitrary web app that serves up files off a file system can be used to host a remote repository as long as certain directory and file naming conventions are followed. A local repository is managed by the Skipper server and backed by a relational database. Skipper lets you search for packages that are hosted in repositories.

Releases are created in Skipper after you install, upgrade, or rollback a package. A release has a unique name that you provide to perform release operations such as upgrading, rolling back, and deleting. The release contains the fully resolved template files, also known as **application manifests**, that represent the final description of what has been deployed to the platform. You can also get the status and application manifest for a specific release.

Release Workflows are the steps taken to upgrade or rollback an application from one version to another. In Skipper terms, it is how we go from one Release to another on a Platform.

An upgrade may keep the same version but update application properties.

Getting Started

This section describes the minimal steps to install Skipper on your local machine in addition to using Skipper to installing a sample application. It is the “three-second tour”. After completing this section, you can move on to the Three minute Tour. When you are ready to dive deeper, head on over to the “three-hour tour” section, Using Skipper. (Well, it is not really three hours….)

Chapter 5. System Requirements

The Skipper server is a Spring Boot application. Both the server and the shell are based on Java 8. The server uses an RDBMS to store state. An embedded H2 database is used if you do not provide a Data Source configuration through Spring Boot configuration properties. Supported databases are H2, HSQLDB, MySQL, Oracle, Postgresql, DB2, and SqlServer. Schemas are created on server startup

Chapter 6. Installing Skipper

This section covers installing Skipper on your **local machine**, as it is the easiest way to get started. The section Installation discusses installing on Cloud Foundry and Kubernetes. It also shows additional options for installing on your local machine.

- Download the Skipper server and shell apps by using the following commands in a terminal session:

```
wget http://repo.spring.io/0-snapshot/org/springframework/cloud/spring-cloud-skipper-server/2.5.0-SNAPSHOT/spring-cloud-skipper-server-2.5.0-SNAPSHOT.jar

wget http://repo.spring.io/0-snapshot/org/springframework/cloud/spring-cloud-skipper-shell/2.5.0-SNAPSHOT/spring-cloud-skipper-shell-2.5.0-SNAPSHOT.jar
```

- Launch the server and shell apps by using the following commands in a terminal session:

```
java -jar spring-cloud-skipper-server-2.5.0-SNAPSHOT.jar


java -jar spring-cloud-skipper-shell-2.5.0-SNAPSHOT.jar
```

The default port that the server listens on is 7577. That is `SKPR` on a telephone keypad. :)

There is also a docker image hosted on dockerhub

Now install some apps!

Chapter 7. A Three-second Tour

The default configuration of Skipper deploys apps to the local machine. The default configuration also has one local repository, named `local`, where you can upload packages. You can get a list of the package repositories by using the command `repo list`, as shown (with its output) in the following example:

```
skipper:>repo list
╔════════════╤═══════════════════════════════════════════════════════════╤═════╤═════╗
║    Name    │                           URL                             │Local│Order║
╠════════════╪═══════════════════════════════════════════════════════════╪═════╪═════╣
║local       │https://10.55.13.45:7577                                   │true │1    ║
╚════════════╧═══════════════════════════════════════════════════════════╧═════╧═════╝
```

Search for the available packages using the `package search` or its alias `package list` command. The following example shows the `package search` command and typical output for it:

```
skipper:>package search
╔═════════════════╤═══════╤════════════════════════════════════════════════════╗
║      Name       │Version│                    Description                     ║
╠═════════════════╪═══════╪════════════════════════════════════════════════════╣
║helloworld       │1.0.1  │The app has two endpoints, /about and /greeting in
Portuguese.  Maven resource. ║
║helloworld       │1.0.0  │The app has two endpoints, /about and /greeting in
English.  Maven resource.    ║
║helloworld-docker│1.0.1  │The app has two endpoints, /about and /greeting in
Portuguese.  Docker resource.║
║helloworld-docker│1.0.0  │The app has two endpoints, /about and /greeting in
English.  Docker resource.   ║
╚═════════════════╧═══════╧════════════════════════════════════════════════════╝
```

Install the Maven-based Hello World application by using the `package install` command. Since this application picks a random port for the HTTP server by default, we specify the Spring Boot property `server.port`, prefixed with `spec.applicationProperties`. The prefix is due to the internal format of the template file. The following example shows the whole command with its output:

```
skipper:>package install --release-name helloworld-local --package-name helloworld
--package-version 1.0.0 --properties spec.applicationProperties.server.port=8099
Released helloworld-local. Now at version v1.
```

You can now curl the `greeting` endpoint, as follows:

```
$ curl http://localhost:8099/greeting
Hello World!
```

The release name, `helloworld-local`, is used for subsequent commands, such as `release status`, `release upgrade` or `release delete`.

To see the status of the release, use the `release status` command, as shown (with its output) in the following example:

```
skipper:>release status --release-name helloworld-local
╔═══════════════════════════════╤═══════════════════════════════════════════════════════════════════════════════
═══════════════════════════════════════════════════════════════════════════════
═══════════════════════════════════════╗
║Last Deployed   │Fri Oct 27 16:17:53 IST 2017
║
║Status          │DEPLOYED
║
║Platform Status │All applications have been successfully deployed.
║
║                │[helloworld-local.helloworld-v1], State = [helloworld-
local.helloworld-v1-0=deployed]║
╚═══════════════════════════════╧═══════════════════════════════════════════════════════════════════════════════
═══════════════════════════════════════════════════════════════════════════════
═══════════════════════════════════════╝
```

Now we can upgrade the release. The `1.0.1` package refers to a newly released application that changed the default value of the greeting to be in `Portuguese`. The following example shows a typical `release upgrade` command with its output:

```
skipper:>release upgrade --release-name helloworld-local --package-name helloworld
--package-version 1.0.1  --properties spec.applicationProperties.server.port=8100
helloworld-local has been upgraded.  Now at version v2.
```

The preceding example command deploys the new version of the application, waits until it is healthy, and then destroys the old version of the application. You can then see the status of the application by using the `release status` command, as follows:

```
skipper:>release status --release-name helloworld-local
╔═══════════════╤══════════════════════════════════════════════════════════════════════════════════════════════════════════════╗
║Last Deployed  │Fri Oct 27 16:20:07 IST 2017
║
║Status         │DEPLOYED
║
║Platform Status│All applications have been successfully deployed.
║
║               │[helloworld-local.helloworld-v2], State = [helloworld-local.helloworld-v2-0=deployed]║
╚═══════════════╧══════════════════════════════════════════════════════════════════════════════════════════════════════════════╝
```

You can now curl the `greeting` endpoint at the new port and see that the application has been updated, as follows:

```
$ curl http://localhost:8100/greeting
Olá Mundo!
```

To delete the release, use the `delete` command, as shown (with its output) in the following example:

```
skipper:>release delete --release-name helloworld-local
helloworld-local has been deleted.
```

> This example, where the upgrade changed only a property of the application, is not realistic. A more realistic example is the case where code has changed so that the updated application behaves differently.

You can also deploy the other packages named `helloworld-docker` to the local machine.

The examples in this section have shown the most basic operations. Other interesting commands such as `manifest get`, `release rollback`, `release list`, and `release history` are covered in the Three minute Tour.

Three minute Tour

Picking up from where the A Three-second Tour left off, this section walks through the additional commands and other features of Skipper. Each section walks through the same set of operations, but for a different platform:

- Local Machine
- Cloud Foundry
- Kuberenetes

Chapter 8. Local Machine

Start up the server and shell as in the three-second tour.

Now you can install and then update the Hello World application. Start by running the `package install` command, as shown (with its output) in the following example:

```
skipper:>package install --release-name helloworldlocal --package-name helloworld
--package-version 1.0.0 --properties spec.applicationProperties.server.port=8099
Released helloworldlocal. Now at version v1.
```

You can now curl the `greeting` endpoint, as shown (with its output) in the following example:

```
$ curl http://localhost:8099/greeting
Hello World!
$ curl http://localhost:8099/about
Hello World v1.0.0.RELEASE
```

We use a YAML file to update the release. This application contains a Spring Boot `@ConfigurationProperty`, named `helloworld.greeting`, so we set that along with a standard Spring Boot property: `endpoints.sensitive=false`. We also bump the memory up to 2G, make the Boot actuator endpoint not sensitive, and set the port to 8100.

The `helloworld-upgrade-local.yml` file contains the following code:

```
spec:
  applicationProperties:
    server.port: 8100
    endpoints.sensitive: false
    helloworld.greeting: yo
  deploymentProperties:
    spring.cloud.deployer.memory: 2048m
```

The following example shows the `release upgrade` command, with its output:

```
skipper:>release upgrade --release-name helloworldlocal --package-name helloworld
--package-version 1.0.1 --file /home/mpollack/helloworld-upgrade-local.yml
helloworldlocal has been upgraded.  Now at version v2.
```

The `--package-version 1.0.1` command line option is also used to upgrade to a newer version of the package.

The current upgrade strategy is simple: If the new app is healthy, the old app is removed. There is no rolling upgrade option. All new apps are deployed and checked for health. Then any previous versions are removed. More flexible upgrade strategies are planned in a future release of Skipper.

You can now curl the `greeting` endpoint and the `about` endpoint, as shown (with its output) in the following example:

```
$ curl http://localhost:8100/greeting
yo
$ curl http://localhost:8100/about
Hello World v1.0.1.RELEASE
```

You can also view the endpoints in your browser.

The `list` command shows you the current `DEPLOYED` and `DELETED` releases for every release name. In this case there, is just one entry, as you can see with the `release list` command, as follows:

```
skipper:>release list
╔═══════════════╤═══════╤════════════╤════════╤══════════╤═════════╤════════
════════╤══════════════════════╤═══════════════════╤═══════════════════╤════
═══════════════════════════════════════════════════════════════════════════
════╗
║     Name      │Version│Last updated │ Status │ Package  │ Package │Platform
│                 Platform Status                       ║
║               │       │             │        │   Name   │ Version │  Name
│                                                       ║
╠═══════════════╪═══════╪════════════╪════════╪══════════╪═════════╪════════
════════╪══════════════════════╪═══════════════════╪═══════════════════╪════
═══════════════════════════════════════════════════════════════════════════
════╣
║helloworldlocal│2      │Fri Oct 27   │DEPLOYED│helloworld│1.0.1    │default
│[helloworldlocal.helloworld-v2], State =               ║
║               │       │16:39:03 IST │        │          │         │
│[helloworldlocal.helloworld-v2-0=deployed]             ║
║               │       │2017         │        │          │         │
│                                                       ║
╚═══════════════╧═══════╧════════════╧════════╧══════════╧═════════╧════════
════════╧══════════════════════╧═══════════════════╧═══════════════════╧════
═══════════════════════════════════════════════════════════════════════════
════╝
```

You can get the full history of the release by using the `history` command, as shown (with its output) in the following example:

```
skipper:>release history --release-name helloworldlocal
╔═══════╤══════════════════════════╤═══════════╤══════════════╤═══════════════╤═══════════════╗
║Version│       Last updated       │  Status   │ Package Name │ Package Version │ Description ║
╠═══════╪══════════════════════════╪═══════════╪══════════════╪═══════════════╪═══════════════╣
║2      │Fri Oct 27 16:39:03 IST 2017│DEPLOYED │helloworld    │1.0.1          │Upgrade complete║
║1      │Fri Oct 27 16:37:59 IST 2017│DELETED  │helloworld    │1.0.0          │Delete complete ║
╚═══════╧══════════════════════════╧═══════════╧══════════════╧═══════════════╧═══════════════╝
```

To see what changed, you can look at the Skipper manifest for each release by using the `manifest get` command, as shown (with its output) in the following example:

```
skipper:>manifest get --release-name helloworldlocal --release-version 2

---
# Source: helloworld.yml
apiVersion: skipper.spring.io/v1
kind: SpringCloudDeployerApplication
metadata:
  name: helloworld
  type: demo
spec:
  resource: maven://org.springframework.cloud.samples:spring-cloud-skipper-samples-
helloworld:1.0.1.RELEASE
  applicationProperties:
    server.port: 8100
    endpoints.sensitive: false
    helloworld.greeting: yo
  deploymentProperties:
    spring.cloud.deployer.memory: 2048m
    spring.cloud.deployer.count: 1
```

The following example shows the `manifest get` command and its output for version 1:

```
skipper:>manifest get --release-name helloworldlocal --release-version 1

---
# Source: helloworld.yml
apiVersion: skipper.spring.io/v1
kind: SpringCloudDeployerApplication
metadata:
  name: helloworld
  type: demo
spec:
  resource: maven://org.springframework.cloud.samples:spring-cloud-skipper-samples-
helloworld:1.0.0.RELEASE
  applicationProperties:
    server.port: 8099
  deploymentProperties:
```

(A `manifest diff` command is coming in a future release.)

Now we can use the `rollback` command to deploy an older version of the application. Since we have the manifest for that version, we have all we need to redeploy an earlier release, as shown (with its output) in the following example:

```
skipper:>release rollback --release-name helloworldlocal --release-version 1
helloworldlocal has been rolled back.  Now at version v3.
```

The history now shows a new `v3` version, even though it is identical in terms of app behavior to the `v1` version.

The `release history` command shows all the versions that have been deployed, as shown (with its output) in the following example:

```
skipper:>release history --release-name helloworldlocal
╔═══════╤═══════════════════════════╤═══════════╤══════════════╤═══════════════╤═══════════════╗
║Version│       Last updated        │  Status   │ Package Name │ Package Version │ Description ║
╠═══════╪═══════════════════════════╪═══════════╪══════════════╪═══════════════╪═══════════════╣
║3      │Fri Oct 27 16:42:47 IST 2017│DEPLOYED │helloworld    │1.0.0          │Upgrade complete║
║2      │Fri Oct 27 16:39:03 IST 2017│DELETED  │helloworld    │1.0.1          │Delete complete ║
║1      │Fri Oct 27 16:37:59 IST 2017│DELETED  │helloworld    │1.0.0          │Delete complete ║
╚═══════╧═══════════════════════════╧═══════════╧══════════════╧═══════════════╧═══════════════╝
```

You can now curl the `greeting` endpoint and see the output of each endpoint, as follows:

```
$ curl http://localhost:8099/greeting
Hello World!
$ curl http://localhost:8099/about
Hello World v1.0.0.RELEASE
```

Chapter 9. Cloud Foundry

First, follow the instructions in the section Installing on Cloud Foundry to deploy the Skipper Server to Cloud Foundry.

When you start the Skipper shell, by default, it tries to look for the Skipper server on the same (local) machine. To specify the Skipper server that is running on Cloud Foundry, provide the `serverUrl` when launching the shell or use the `config` command after the shell has started. The following example provides the `serverUrl`:

```
java -jar spring-cloud-skipper-shell-2.5.0-SNAPSHOT.jar
--spring.cloud.skipper.client.serverUri=https://mlp-skipper.cfapps.io/api
```

The following example uses `config`:

```
skipper:>skipper config --uri https://mlp-skipper.cfapps.io/api
Successfully targeted https://mlp-skipper.cfapps.io/api
```

The `repo list` command shows the `experimental` and `local` repositories, since they are configured by default. The `local` repository is where you can upload new packages. The `experimental` repository has a few "hello world" applications to help get you started. The following example shows the `repo list` command and the output of our example:

```
skipper:>repo list
╔════════════════════════════════╤═══════════════════════════════════════════════════════════════════════════
═══════════════════════════════════════════════════════════════════════════╤══════════════╤══════════════╗
║     Name     │                              URL
│ Local │ Order ║
╠════════════════════════════════╪═══════════════════════════════════════════════════════════════════════════
═══════════════════════════════════════════════════════════════════════════╪══════════════╪══════════════╣
║experimental │https://skipper-repository.cfapps.io/repository/experimental │false │0
║
║local        │https://d4d6d1b6-c7e5-4226-69ec-01d4:7577                    │true  │1
║
╚════════════════════════════════╧═══════════════════════════════════════════════════════════════════════════
═══════════════════════════════════════════════════════════════════════════╧══════════════╧══════════════╝
```

> Above example assumes that `experimental` repository has been added to the server configuration. More about working with repositories can be found from Repositories.

The following example shows the `package search` command and the output of our example:

```
skipper:>package search
╔═══════════════════╤═══════╤═════════════════════════════════════════════════════════════════════════════════╗
║       Name        │Version│                                   Description                                   ║
╠═══════════════════╪═══════╪═════════════════════════════════════════════════════════════════════════════════╣
║helloworld         │1.0.1  │The app has two endpoints, /about and /greeting in Portuguese.  Maven resource.  ║
║helloworld         │1.0.0  │The app has two endpoints, /about and /greeting in English.  Maven resource.     ║
║helloworld-docker  │1.0.1  │The app has two endpoints, /about and /greeting in Portuguese.  Docker resource. ║
║helloworld-docker  │1.0.0  │The app has two endpoints, /about and /greeting in English.  Docker resource.    ║
╚═══════════════════╧═══════╧═════════════════════════════════════════════════════════════════════════════════╝
```

The command `platform list` shows the platforms with which the server has been configured, as shown (with its output) in the following example:

```
╔════╤════════════╤═════════════════════════════════════════════════════════════════════════╗
║Name│    Type    │                                Description                              ║
╠════╪════════════╪═════════════════════════════════════════════════════════════════════════╣
║pws │cloudfoundry│org = [scdf-ci], space = [space-mark], url = [https://api.run.pivotal.io]║
╚════╧════════════╧═════════════════════════════════════════════════════════════════════════╝
```

In the preceding example, there is only one Cloud Foundry platform.

Now we can install the Hello World app (specifically, the maven based artifact). The following example shows the `package install` command (with its output) that we use to install the `Hello World` application:

```
skipper:>package install --release-name helloworldpcf --package-name helloworld
--package-version 1.0.0 --platform-name pws --properties
spec.deploymentProperties.spring.cloud.deployer.cloudfoundry.route=helloworldpcf.cfapp
s.io
Released helloworldpcf. Now at version v1.
```

The `spring.cloud.deployer.cloudfoundry.route=helloworldpcf.cfapps.io` deployment property is set so that, when different versions of this application are deployed, they have the same HTTP route.

Because the default value of that shell option is `default`he `--platform-name pws`, we used the command option. When installing Skipper, you can register a platform under the name `default`, but it is a best practice to specify the target platform name.

You can monitor the progress of the deployment by using the `release status` command, as shown (with its output) in the following example:

```
skipper:>release status --release-name helloworldpcf
╔═══════════════╤═════════════════════════════════════════════════╗
║Last Deployed  │Thu Jan 18 13:18:44 EST 2018                     ║
║Status         │DEPLOYED                                         ║
║Platform Status│The applications are being deployed.             ║
║               │[helloworldpcf-helloworld-v1], State = [partial] ║
╚═══════════════╧═════════════════════════════════════════════════╝
```

Eventually, the Platform Status says, `All applications have been successfully deployed.`

The `DEPLOYED` status in the preceding example indicates that Skipper has told the platform to deploy. Skipper does not keep track of the intermediate states 'deploying' or 'deleting'. The platform status provides finer-grained status information.

The `cf apps` command now has a new listing for this deployed application, as shown (with its output) in the following example:

```
$ cf apps
Getting apps in org scdf-ci / space space-mark as mpollack@gopivotal.com...
OK

name                          requested state   instances   memory   disk   urls
helloworldpcf-helloworld-v1   started           1/1         1G       1G
helloworldpcf.cfapps.io
```

You can now curl the `greeting` endpoint and the `about` endpoint, as shown in the following example:

```
$ curl https://helloworldpcf.cfapps.io/greeting
Hello World!
$ curl https://helloworldpcf.cfapps.io/about
Hello World v1.0.0.RELEASE
```

The name of the application is based on the `<release-name>-<package-name>-v<incrementing-counter>` convention.

Also note that we specified a route for this application that is different than the application's name. The deployment property `spring.cloud.deployer.cloudfoundry.route` is set to something that does not change across the deployment of different versions of this application — in this case, `helloworldpcf.cfapps.io`.

The package provides a means to template the application version, application properties, and deployment properties that are used to deploy the application to Cloud Foundry. The `manifest get` command shows the final YAML file which is passed off to the Spring Cloud Deployer Library, as shown (with its output) in the following example:

```
skipper:>manifest get --release-name helloworldpcf

---
# Source: helloworld.yml
apiVersion: skipper.spring.io/v1
kind: SpringCloudDeployerApplication
metadata:
  name: helloworld
  type: demo
spec:
  resource: maven://org.springframework.cloud.samples:spring-cloud-skipper-samples-
helloworld:1.0.0.RELEASE
  applicationProperties:
  deploymentProperties:
    spring.cloud.deployer.cloudfoundry.route: helloworldpcf.cfapps.io
```

The manifest format is inspired by the Kubernetes Resource file format. By looking at the manifest, you can see which Maven artifact was used and which properties were set before the final push to Cloud Foundry. A future release of Skipper will use the metadata values to support searching for releases based on those values.

Since it is somewhat awkward to specify multiple flattened-out YAML values for the `--properties` argument in the shell, you can also specify the location of a YAML file when installing or upgrading. In the next example, we use a YAML file, named `helloworld-upgrade.yml`, to update the release. This application contains a Spring Boot `@ConfigurationProperty` named `helloworld.greeting`, so we set that, along with a standard Spring Boot property: `endpoints.sensitive=false`. We also bump the memory up to 2G from the default 1G. The contents of the `helloworld-upgrade.yml` file follows:

```
spec:
  applicationProperties:
    endpoints.sensitive: false
    helloworld.greeting: yo
  deploymentProperties:
    spring.cloud.deployer.cloudfoundry.route: helloworldpcf.cfapps.io
    spring.cloud.deployer.memory: 2048m
```

Now you can run the `release upgrade` command, as shown (with its output) in the following example:

```
skipper:>release upgrade --release-name helloworldpcf --package-name helloworld
--package-version 1.0.0 --file /home/mpollack/helloworld-upgrade.yml
helloworldpcf has been upgraded.  Now at version v2.
```

The preceding example starts another instance of the hello world application, and Skipper determines when it can stop the instance of the previous instance. If you do not specify `--package-version`, it picks the latest version of the `helloworld` package. You do not need to specify the `--platform-name`, as it is always where the current application was deployed.

The following example shows the `cf apps` command and its output:

```
$ cf apps
Getting apps in org scdf-ci / space space-mark as mpollack@gopivotal.com...
OK

name                          requested state   instances   memory   disk   urls
helloworldpcf-helloworld-v1   started           1/1         1G       1G
helloworldpcf.cfapps.io
helloworldpcf-helloworld-v2   stopped           0/1         2G       1G
helloworldpcf.cfapps.io
```

The following example shows the `cf routes` command and its output:

```
$ cf routes
Getting routes for org scdf-ci / space space-mark as mpollack@gopivotal.com ...

space        host                          domain      port   path   type   apps
service
space-mark   helloworldpcf                 cfapps.io
helloworldpcf-helloworld-v1,helloworldpcf-helloworld-v2
```

At this point, Skipper is checking the health of the new application. The default health checks whether the HTTP port of the application is open. There is a customization in Skipper that influences the way the health check is performed. The `spring.cloud.skipper.server.strategies.healthcheck.timeoutInMillis` property is the maximum

time the upgrade process waits for a healthy app. The default value is 5 minutes. Skipper fails the deployment if it is not healthy within that time. The `spring.cloud.skipper.server.strategies.healthcheck.sleepInMillis` property is how long to sleep between health checks.

The current upgrade strategy is very simple: If the new app is healthy, the old app is removed. There is not a rolling upgrade option, all new apps are deployed, checked for health, and then previous versions removed. More flexible upgrade strategies are planned in a future release.

You can now curl the `greeting` endpoint and the `about` endpoint, as shown in the following example:

```
$ curl https://helloworldpcf.cfapps.io/greeting
yo
$ curl https://helloworldpcf.cfapps.io/about
Hello World v1.0.0.RELEASE
```

The `release list` command shows the current `DEPLOYED` and `DELETED` releases for every release name. In the following example from the sample application, there is only one entry, as shown in the following example:

```
╔═══════════════╤═════════╤════════════════════╤═════════╤═════════════╤═════════════╤══
═══════════════╤════════════════════════╤════════════════════════╤═══════════════════╤═
═════════════════════════════════════════════════════════════════════════════════╗
║    Name       │Version│    Last updated    │ Status │   Package   │   Package   │
Platform │          Platform Status          ║
║               │        │                    │        │    Name     │   Version   │
Name    │                                   ║
╟───────────────┼─────────┼────────────────────┼─────────┼─────────────┼─────────────┼──
───────────────┼────────────────────────┼────────────────────────┼───────────────────┼─
─────────────────────────────────────────────────────────────────────────────────╢
║helloworldpcf│2       │Thu Jan 18          │DEPLOYED│helloworld  │1.0.0        │pws
│[helloworldpcf-helloworld-v2], State =  ║
║               │        │13:26:50 EST 2018  │        │             │             │
│[helloworldpcf-helloworld-v2-0=deployed]║
╚═══════════════╧═════════╧════════════════════╧═════════╧═════════════╧═════════════╧══
═══════════════╧════════════════════════╧════════════════════════╧═══════════════════╧═
═════════════════════════════════════════════════════════════════════════════════╝
```

You can get the full history of the release byusing the `release history` command, as shown (with its output) in the following example:

```
skipper:>release history --release-name helloworldpcf
╔═══════╤══════════════════════════╤═══════════╤══════════════╤═══════════════╤═══════════════╗
║Version│       Last updated       │  Status   │Package Name  │Package Version│  Description  ║
╠═══════╪══════════════════════════╪═══════════╪══════════════╪═══════════════╪═══════════════╣
║2      │Thu Jan 18 13:26:50 EST 2018│DEPLOYED │helloworld    │1.0.0          │Upgrade complete║
║1      │Thu Jan 18 13:18:44 EST 2018│DELETED  │helloworld    │1.0.0          │Delete complete ║
╚═══════╧══════════════════════════╧═══════════╧══════════════╧═══════════════╧═══════════════╝
```

A more typical upgrade process is not to change application properties but to change the version of the application because the code has changed. In the following example, we now upgrade the release to use a new Maven artifact, version 1.0.1, which also corresponds to version 1.0.1 of the `helloworld` Skipper package. In this case, we do not add any additional properties other than the route. The following example shows the `release upgrade` command (with its update) to deploy version 1.0.1:

```
skipper:>release upgrade --release-name helloworldpcf --package-name helloworld
--package-version 1.0.1 --properties
spec.deploymentProperties.spring.cloud.deployer.cloudfoundry.route=helloworldpcf.cfapp
s.io
helloworldpcf has been upgraded.  Now at version v3.
```

Note that the current release's property values, such as using 2G or the greeting being `yo` are not carried over. A future release will introduce a `--reuse-properties` command that will carry the current release properties over to the next release to be made. You can monitor the status of the upgrade by using the `status` command, as shown (with its output) in the following example:

```
skipper:>release status --release-name helloworldpcf
╔═══════════════╤═══════════════════════════════════════════════════════╗
║Last Deployed  │Thu Jan 18 13:49:42 EST 2018                           ║
║Status         │UNKNOWN                                                ║
║Platform Status│The applications are being deployed.                   ║
║               │[helloworldpcf-helloworld-v3], State = [partial]       ║
╚═══════════════╧═══════════════════════════════════════════════════════╝
```

Now a `curl` command shows the following output:

```
curl https://helloworldpcf.cfapps.io/greeting
Olá Mundo!
$ curl https://helloworldpcf.cfapps.io/about
Hello World v1.0.1.RELEASE
```

Our release history is now as follows:

```
skipper:>release history --release-name helloworldpcf
╔═══════╤══════════════════════════╤══════════╤══════════════╤═══════════════╤═══════════════╗
║Version│       Last updated       │  Status  │Package Name  │Package Version│  Description  ║
╠═══════╪══════════════════════════╪══════════╪══════════════╪═══════════════╪═══════════════╣
║3      │Thu Jan 18 13:49:42 EST 20│DEPLOYED  │helloworld    │1.0.1          │Upgrade comple-│
║       │18                        │          │              │               │te             ║
╟───────┼──────────────────────────┼──────────┼──────────────┼───────────────┼───────────────╢
║2      │Thu Jan 18 13:26:50 EST 20│DELETED   │helloworld    │1.0.0          │Delete complete║
╟───────┼──────────────────────────┼──────────┼──────────────┼───────────────┼───────────────╢
║1      │Thu Jan 18 13:18:44 EST 20│DELETED   │helloworld    │1.0.0          │Delete complete║
╚═══════╧══════════════════════════╧══════════╧══════════════╧═══════════════╧═══════════════╝
```

Next, we use the `rollback` command to deploy an older version of the application. Since we have the manifest for that version, we have all we need to redeploy an earlier release. The following example shows the `release rollback` command and its output:

```
skipper:>release rollback --release-name helloworldpcf --release-version 2
helloworldpcf has been rolled back.  Now at version v4.
```

The history now shows a new `v4` version, even though it is identical in terms of app behavior to the `v2` version, as follows:

```
skipper:>release history --release-name helloworldpcf
╔═══════╤════════════════════════════╤════════╤════════════╤═══════════════╤
════════════════╗
║Version│        Last updated        │ Status │Package Name│Package Version│
Description    ║
╠═══════╪════════════════════════════╪════════╪════════════╪═══════════════╪
════════════════╣
║4      │Thu Jan 18 13:51:43 EST 2018│DEPLOYED│helloworld  │1.0.0
│Upgrade complete║
║3      │Thu Jan 18 13:49:42 EST 2018│DELETED │helloworld  │1.0.1
│Delete complete ║
║2      │Thu Jan 18 13:26:50 EST 2018│DELETED │helloworld  │1.0.0
│Delete complete ║
║1      │Thu Jan 18 13:18:44 EST 2018│DELETED │helloworld  │1.0.0
│Delete complete ║
╚═══════╧════════════════════════════╧════════╧════════════╧═══════════════╧
════════════════╝
```

The `curl` commands shows the following output:

```
$ curl https://helloworldpcf.cfapps.io/greeting
yo
$ curl https://helloworldpcf.cfapps.io/about
Hello World v1.0.0.RELEASE
```

Chapter 10. Kuberenetes

In this example, we run the Skipper server on the local machine and deploy to minikube, which also runs on the local machine.

> The upgrade approach in 1.02 does not correctly handle the routing of HTTP traffic between versions, so the following representation may not be exactly accurate.

The Spring Cloud Deployer for Kubernetes creates a service, a replication controller, and a pod for the app (or, optionally, a deployment). This is not an issue for apps that communicate over Messaging middleware and will be addressed in a future release.

Start the Skipper server with the `--spring.config.additional-location=skipper.yml` option. The YAML content follows:

```
spring:
  cloud:
    skipper:
      server:
        platform:
          kubernetes:
            accounts:
              minikube:
                namespace: default
```

The `repo list` command shows the `experimental` and `local` repositories, since they are configured by default, as follows:

```
skipper:>repo list
╔════════════╤═══════════════════════════════════════════════════════════════
═══════════════════════════════════════════════════════════╤═════╤═════╗
║    Name    │                            URL
│Local│Order║
╠════════════╪═══════════════════════════════════════════════════════════════
═══════════════════════════════════════════════════════════╪═════╪═════╣
║experimental│https://skipper-repository.cfapps.io/repository/experimental│false│0
║
║local       │https://d4d6d1b6-c7e5-4226-69ec-01d4:7577                   │true │1
║
╚════════════╧═══════════════════════════════════════════════════════════════
═══════════════════════════════════════════════════════════╧═════╧═════╝
```

The `package search` command shows the Name, the Version, and the Description, as follows:

```
skipper:>package search
╔═══════════════════╤═══════╤═══════════════════════════════════════════════════════════════════════╗
║       Name        │Version│                              Description                              ║
╠═══════════════════╪═══════╪═══════════════════════════════════════════════════════════════════════╣
║helloworld         │1.0.1  │The app has two endpoints, /about and /greeting in
Portuguese.  Maven resource. ║
║helloworld         │1.0.0  │The app has two endpoints, /about and /greeting in
English.  Maven resource.    ║
║helloworld-docker  │1.0.1  │The app has two endpoints, /about and /greeting in
Portuguese.  Docker resource.║
║helloworld-docker  │1.0.0  │The app has two endpoints, /about and /greeting in
English.  Docker resource.   ║
╚═══════════════════╧═══════╧═══════════════════════════════════════════════════════════════════════╝
```

The `platform list` command shows which platforms the server has been configured with — in this case, one Kubernetes namespace.

```
skipper:>platform list
╔════════╤═══════════╤══════════════════════════════════════════════════════════════════════════╗
║  Name  │   Type    │                                Description                               ║
╠════════╪═══════════╪══════════════════════════════════════════════════════════════════════════╣
║minikube│kubernetes│master url = [https://192.168.99.100:8443/], namespace =
[default], api version = [v1]║
╚════════╧═══════════╧══════════════════════════════════════════════════════════════════════════╝
```

Now we can install the Hello World app (specifically, the Docker-based artifact), as follows:

```
skipper:>package install --release-name helloworldk8s --package-name helloworld-docker
--package-version 1.0.0 --platform-name minikube --properties
spec.deploymentProperties.spring.cloud.deployer.kubernetes.createNodePort=32123
Released helloworldk8s. Now at version v1.
```

We use the `--platform-name minikube` command option, because the default value of that shell

option is `default`. You can register a platform under the `default` name when installing Skipper, but it is a best practice to specify the target platform name.

You can monitor the process by using the `release status` command, as follows:

```
skipper:>release status --release-name helloworldk8s
╔═══════════════╤═══════════════════════════════════════════════════════════════════════════════════════════════════════════
═══════════════════════════════════════════════════════════╗
║Last Deployed  │Wed Oct 25 17:34:24 EDT 2017
║
║Status         │DEPLOYED
║
║Platform Status│The applications are being deployed.
║
║               │[helloworldk8s-helloworld-docker-v1], State = [helloworldk8s-
helloworld-docker-v1-cch68=deploying]║
╚═══════════════╧═══════════════════════════════════════════════════════════════════════════════════════════════════════════
═══════════════════════════════════════════════════════════╝
```

Eventually, the Platform Status says, `All applications have been successfully deployed.`

Note that the `DEPLOYED` status in the preceding example indicates that Skipper has told the platform to deploy. Skipper does not keep track of the intermediate states ('deploying' or 'deleting').

A `kubectl pods` command now shows a new listing for this deployed application, as follows:

```
$ kubectl get pods
NAME                                    READY     STATUS    RESTARTS   AGE
helloworldk8s-helloworld-docker-v1-g8j39   0/1       Running   0          37s

$ kubectl get service
NAME                                 CLUSTER-IP   EXTERNAL-IP   PORT(S)          AGE
helloworldk8s-helloworld-docker-v1   10.0.0.202   <nodes>       8080:32123/TCP   41s
kubernetes                           10.0.0.1     <none>        443/TCP          57m
```

To get the URL of this app on minikube, use the `minikube service` command, as follows:

```
$ minikube service --url helloworldk8s-helloworld-docker-v1
https://192.168.99.100:32123
```

You can now curl the `greeting` endpoint and the `about` endpoint, as shown in the following example:

```
$ curl https://192.168.99.100:32123/greeting
Hello World!
$ curl https://192.168.99.100:32123/about
Hello World v1.0.0.RELEASE
```

The name of the application is based on the following convention: `<release-name>-<package-name>-v<incrementing-counter>`. Future releases will change this convention to correctly handle routing.

The package provides a means to template the application version, application properties, and deployment properties that are used to deploy the application to Kubernetes. The `manifest get` command shows the final YAML file, which is passed off to the Spring Cloud Deployer Library, as shown (with its output) in the following example:

```
skipper:>manifest get --release-name helloworldk8s

---
# Source: template.yml
apiVersion: skipper.spring.io/v1
kind: SpringCloudDeployerApplication
metadata:
  name: helloworld-docker
spec:
  resource: docker:springcloud/spring-cloud-skipper-samples-helloworld:1.0.0.RELEASE
  applicationProperties:
  deploymentProperties:
    spring.cloud.deployer.kubernetes.createNodePort: 32123
```

The format of the is inspired by the Kubernetes Resource file format. By looking at the manifest, you can see which Docker images were used and which properties were set before the final push to Kubernetes. A future release of Skipper will use the metadata values to support searching for releases based on those values.

Since it is somewhat awkward to specify multiple flattened out YAML values for the `--properties` argument in the shell, you can also specify the location of a YAML file when installing or upgrading. We use a YAML file when we update the release. This application contains a Spring Boot `@ConfigurationProperty` named `helloworld.greeting`, so we set that, along with a standard Spring Boot property: `endpoints.sensitive=false`. We also bump the memory down to 768m from the default 1G. The following listing shows all the settings:

```
spec:
  applicationProperties:
    endpoints.sensitive: false
    helloworld.greeting: yo
  deploymentProperties:
    spring.cloud.deployer.kubernetes.createNodePort: 32124
    spring.cloud.deployer.memory: 768m
```

The following example shows the `release upgrade` command and its output:

```
skipper:>release upgrade --release-name helloworldk8s --package-name helloworld-docker
--package-version 1.0.0 --file /home/mpollack/helloworld-upgrade-k8s.yml
helloworldk8s has been upgraded.  Now at version v2.
```

The preceding command starts another instance of the hello world application. If you do not specify `--package-version`, it picks the latest version of the `helloworld-docker` package. You do not need to specify the `--platform-name` as it is always where the current application was deployed.

The following example shows the `kubectl get all` command and its output:

```
$ kubectl get all
NAME                                          READY     STATUS    RESTARTS   AGE
po/helloworldk8s-helloworld-docker-v1-g8j39   1/1       Running   0          2m
po/helloworldk8s-helloworld-docker-v2-jz85l   0/1       Running   0          50s

NAME                                    DESIRED   CURRENT   READY     AGE
rc/helloworldk8s-helloworld-docker-v1   1         1         1         2m
rc/helloworldk8s-helloworld-docker-v2   1         1         0         50s

NAME                                     CLUSTER-IP   EXTERNAL-IP   PORT(S)
AGE
svc/helloworldk8s-helloworld-docker-v1   10.0.0.202   <nodes>       8080:32123/TCP
2m
svc/helloworldk8s-helloworld-docker-v2   10.0.0.154   <nodes>       8080:32124/TCP
51s
svc/kubernetes                           10.0.0.1     <none>        443/TCP
59m
```

At this point, Skipper is looking to see if the health endpoint of the Boot application is OK. The `spring.cloud.skipper.server.strategies.healthcheck.timeoutInMillis` property sets the maximum time the upgrade process waits for a healthy app. The default value is 5 minutes. Skipper fails the deployment if it is not healthy within that time. The `spring.cloud.skipper.server.strategies.healthcheck.sleepInMillis` property sets how long to sleep between health checks.

The current upgrade strategy is simple: If the new app is healthy, the old app is removed. There is not a rolling upgrade option. All new apps are deployed and checked for health. Then any previous versions are removed. Future releases will have more flexible upgrade strategies, along with the introduction of the Spring Cloud State Machine project to orchestrate the update process.

You can now curl the `greeting` endpoint and the `about` endpoint, as follows:

```
$ curl https://192.168.99.100:32124/greeting
yo
$ curl https://192.168.99.100:32124/about
Hello World v1.0.0.RELEASE
```

The `release list` command shows the current `DEPLOYED` and `DELETED` release for every release name. In the following example, there is only one entry:

```
skipper:>release list
╔═══════════════╤═══════╤═══════════════════════════╤══════════╤═══════════════╤═══════════════╤═══════════════╗
║     Name      │Version│       Last updated        │  Status  │ Package Name  │Package Version│Platform Name  │Platform Status║
╟───────────────┼───────┼───────────────────────────┼──────────┼───────────────┼───────────────┼───────────────┼───────────────╢
║helloworldk8s  │2      │Wed Oct 25 17:36:16 EDT 2017│DEPLOYED  │helloworld-docker│1.0.0          │minikube       │               ║
╚═══════════════╧═══════╧═══════════════════════════╧══════════╧═══════════════╧═══════════════╧═══════════════╧═══════════════╝
```

You can get the full history of the release using the `history` command, as follows:

```
skipper:>release history --release-name helloworldk8s
╔═══════╤═══════════════════════════╤══════════╤═══════════════╤═══════════════╤═══════════════╗
║Version│       Last updated        │  Status  │ Package Name  │Package Version│  Description  ║
╟───────┼───────────────────────────┼──────────┼───────────────┼───────────────┼───────────────╢
║2      │Wed Oct 25 17:36:16 EDT 2017│DEPLOYED  │helloworld-docker│1.0.0          │Upgrade complete║
╟───────┼───────────────────────────┼──────────┼───────────────┼───────────────┼───────────────╢
║1      │Wed Oct 25 17:34:24 EDT 2017│DELETED   │helloworld-docker│1.0.0          │Delete complete ║
╚═══════╧═══════════════════════════╧══════════╧═══════════════╧═══════════════╧═══════════════╝
```

A more typical upgrade process is not to change application properties but to change the version of the application because the code has changed. We can now upgrade the release to use a new Docker artifact, version 1.0.1, which also corresponds to version 1.0.1 of the `helloworld` Skipper package. In the following example, we do not add any additional properties other than `NodePort`:

```
skipper:>release upgrade --release-name helloworldk8s --package-name helloworld-docker
--package-version 1.0.1 --properties
spec.deploymentProperties.spring.cloud.deployer.kubernetes.createNodePort=32125
Released helloworldk8s. Now at version v3.
```

Note that the the current release's property values, such as using 2G RAM or the greeting being `yo`, are not carried over. A future release will introduce a `--reuse-properties` command option that will carry the current release properties over to the next release to be made. You can monitor the status of the upgrade by using the `status` command, as shown (with its output) in the following example:

```
skipper:>release status --release-name helloworldk8s
╔═══════════════╤═══════════════════════════════════════════════════════════════════════════════════════════════════════╗
║Last Deployed  │Wed Oct 25 17:41:33 EDT 2017                                                                            ║
║Status         │DEPLOYED                                                                                                ║
║Platform Status│All applications have been successfully deployed.                                                       ║
║               │[helloworldk8s-helloworld-docker-v3], State = [helloworldk8s-
helloworld-docker-v3-sb59j=deployed]  ║
╚═══════════════╧═══════════════════════════════════════════════════════════════════════════════════════════════════════╝
```

A `curl` command shows the following output:

```
$ curl https://192.168.99.100:32125/greeting
Olá Mundo!

$ curl https://192.168.99.100:32125/about
Hello World v1.0.1.RELEASE
```

The following example shows the `release history` command and its output:

```
skipper:>release history --release-name helloworldk8s
╔╤═══════════════╤═══════════════════════════════════════════════════════╤══════════
═══════╤════════════════════════════════════╤═══════════════════════════════╤═══════════
═══════════════════════╗
║│Version│         Last updated          │ Status │   Package Name    │Package
Version│   Description    ║
║├───────────────┼───────────────────────────────────────────────────────┼──────────
───────┼────────────────────────────────────┼───────────────────────────────┼───────────
───────────────────────╢
║│3      │Wed Oct 25 17:41:33 EDT 2017│DEPLOYED│helloworld-docker│1.0.1
│Upgrade complete║
║│2      │Wed Oct 25 17:36:16 EDT 2017│DELETED │helloworld-docker│1.0.0
│Delete complete ║
║│1      │Wed Oct 25 17:34:24 EDT 2017│DELETED │helloworld-docker│1.0.0
│Delete complete ║
╚╧═══════════════╧═══════════════════════════════════════════════════════╧══════════
═══════╧════════════════════════════════════╧═══════════════════════════════╧═══════════
═══════════════════════╝
```

Next, we use the `rollback` command to deploy an older version of the application. Since we have the manifest for that version, we have all we need to redeploy an earlier release. The following example shows the rollback command and its output:

```
skipper:>release rollback --release-name helloworldk8s --release-version 2
helloworldk8s has been rolled back.  Now at version v4.
```

The history now shows a new `v4` version, even though it is identical to the `v2` version, as shown in the following example:

```
skipper:>release history --release-name helloworldk8s
╔═════════╤══════════════════════════════════════════════════════════╤═══════
═══════╤═══════════════════════════════════╤════════════════════════════╤═══════
═══════════════════════╗
║Version│        Last updated        │ Status │   Package Name    │Package
Version│   Description   ║
╠═════════╪══════════════════════════════════════════════════════════╪═══════
═══════╪═══════════════════════════════════╪════════════════════════════╪═══════
═══════════════════════╣
║4      │Wed Oct 25 17:44:25 EDT 2017│DEPLOYED│helloworld-docker│1.0.0
│Upgrade complete║
║3      │Wed Oct 25 17:41:33 EDT 2017│DELETED │helloworld-docker│1.0.1
│Delete complete ║
║2      │Wed Oct 25 17:36:16 EDT 2017│DELETED │helloworld-docker│1.0.0
│Delete complete ║
║1      │Wed Oct 25 17:34:24 EDT 2017│DELETED │helloworld-docker│1.0.0
│Delete complete ║
╚═════════╧══════════════════════════════════════════════════════════╧═══════
═══════╧═══════════════════════════════════╧════════════════════════════╧═══════
═══════════════════════╝
```

The `curl` commands now shows the following:

```
$ curl https://192.168.99.100:32124/greeting
yo
$ curl https://192.168.99.100:32124/about
Hello World v1.0.0.RELEASE
```

Chapter 11. CF manifest based deployments

Following examples cover the scenarios of managing CF manifest based packages.

```
skipper:>platform list
╔═══════════════╤══════════════════════════╤═══════════════════════════════════════════════════════════════════════════════════════════════════════════╗
║  Name   │     Type     │                                   Description                                    ║
╠═══════════════╪══════════════════════════╪═══════════════════════════════════════════════════════════════════════════════════════════════════════════╣
║cf-dev  │cloudfoundry│org = [scdf-ci], space = [space-ilaya], url =
[https://api.run.pivotal.io]║
╚═══════════════╧══════════════════════════╧═══════════════════════════════════════════════════════════════════════════════════════════════════════════╝
```

Upload the log application packages available in the test directory under `spring-cloud-skipper-server-core`.

```
skipper:>package upload --repo-name local --path spring-cloud-skipper-server-
core/src/test/resources/repositories/binaries/test/log/logcf-1.0.0.zip
Package uploaded successfully:[logcf:1.0.0]
```

```
skipper:>package upload --repo-name local --path spring-cloud-skipper-server-
core/src/test/resources/repositories/binaries/test/log/logcf-1.0.1.zip
Package uploaded successfully:[logcf:1.0.1]
```

```
skipper:>package search
╔═══════════════════════════════════╤═══════════════╤═══════════════════════════════
═══════════════════════════════════════════════════════════════════════════════════
═════════════════════════════════════════╗
║      Name        │ Version │                                  Description
║
╠═══════════════════════════════════╪═══════════════╪═══════════════════════════════
═══════════════════════════════════════════════════════════════════════════════════
═════════════════════════════════════════╣
║helloworld        │1.0.0    │The app has two endpoints, /about and /greeting in
English.  Maven resource.    ║
║helloworld        │1.0.1    │The app has two endpoints, /about and /greeting in
Portuguese.  Maven resource. ║
║helloworld-docker │1.0.0    │The app has two endpoints, /about and /greeting in
English.  Docker resource.   ║
║helloworld-docker │1.0.1    │The app has two endpoints, /about and /greeting in
Portuguese.  Docker resource.║
║logcf             │1.0.0    │The log sink uses the application logger to output the
data for inspection.     ║
║logcf             │1.0.1    │The log sink uses the application logger to output the
data for inspection.     ║
╚═══════════════════════════════════╧═══════════════╧═══════════════════════════════
═══════════════════════════════════════════════════════════════════════════════════
═════════════════════════════════════════╝
```

Install the `logcf` package with the version `1.0.0`

```
skipper:>package install logcf --release-name a1 --platform-name cf-dev --package
-version 1.0.0
Released a1. Now at version v1.

skipper:>release list
╔════╤═══════╤══════════════════════════╤════════╤════════════╤═══════════════
═══════════╤═════════════╤═══════════════════════════════════════════════════
═══════════╤═════════════════════════════════════════════════════════════════
═══╗
║Name│Version│       Last updated        │ Status │Package Name│Package
Version│Platform Name│         Platform Status          ║
╠════╪═══════╪══════════════════════════╪════════╪════════════╪═══════════════
═══════════╪═════════════╪═══════════════════════════════════════════════════
═══════════╪═════════════════════════════════════════════════════════════════
═══╣
║a1  │1      │Thu Aug 09 12:29:02 IST 2018│DEPLOYED│logcf       │1.0.0
│cf-dev       │[a1-v1], State = [a1-v1-0=deployed]║
╚════╧═══════╧══════════════════════════╧════════╧════════════╧═══════════════
═══════════╧═════════════╧═══════════════════════════════════════════════════
═══════════╧═════════════════════════════════════════════════════════════════
═══╝

skipper:>release history a1
╔═══════╤════════════════════════════════════════════════════════════╤═══════
═══════╤══════════════════════════╤════════════════════════════════╤══════════
═════════════╗
║Version│       Last updated        │ Status │Package Name│Package Version│
Description   ║
╠═══════╪════════════════════════════════════════════════════════════╪═══════
═══════╪══════════════════════════╪════════════════════════════════╪══════════
═════════════╣
║1      │Thu Aug 09 12:29:02 IST 2018│DEPLOYED│logcf       │1.0.0
│Install complete║
╚═══════╧════════════════════════════════════════════════════════════╧═══════
═══════╧══════════════════════════╧════════════════════════════════╧══════════
═════════════╝

skipper:>manifest get a1
"apiVersion": "skipper.spring.io/v1"
"kind": "CloudFoundryApplication"
"spec":
  "resource": "maven://org.springframework.cloud.stream.app:log-sink-rabbit"
  "version": "1.3.0.RELEASE"
  "manifest":
    "memory": "1024"
    "disk-quota": "2048"
    "instances": "1"
    "services":
    - "rabbit"
    "timeout": "180"
```

```
$ cf apps
Getting apps in org scdf-ci / space space-ilaya as igopinathan@pivotal.io...
OK

name    requested state   instances   memory   disk   urls
a1-v1   started           1/1         1G       2G     a1-v1.cfapps.io
```

Upgrade the `logcf` package with the version `1.0.1`

```
skipper:>release upgrade --package-name logcf --package-version 1.0.1 --release-name
a1
a1 has been upgraded.  Now at version v2.

skipper:>release list
╔═════╤═══════╤════════════════════════════════════════════════════════════════════════╤
════════════════╤════════════════════════════════╤════════════════════════════════╤═════════
════════════════╤═════════════════════════════════════════════════════════════════════
═══╗
║Name│Version│         Last updated         │ Status │Package Name│Package
Version│Platform Name│            Platform Status            ║
╠═════╪═══════╪════════════════════════════════════════════════════════════════════════╪
════════════════╪════════════════════════════════╪════════════════════════════════╪═════════
════════════════╪═════════════════════════════════════════════════════════════════════
═══╣
║a1   │2      │Thu Aug 09 12:33:44 IST 2018│DEPLOYED│logcf       │1.0.1
│cf-dev       │[a1-v2], State = [a1-v2-0=deployed]║
╚═════╧═══════╧════════════════════════════════════════════════════════════════════════╧
════════════════╧════════════════════════════════╧════════════════════════════════╧═════════
════════════════╧═════════════════════════════════════════════════════════════════════
═══╝

skipper:>release history a1
╔═══════════════╤════════════════════════════════════════════════════════════════════════╤
══════╤════════════════════════════╤════════════════════════════════╤═════════════════
═════════════╗
║Version│         Last updated         │ Status │Package Name│Package Version│
Description    ║
╠═══════════════╪════════════════════════════════════════════════════════════════════════╪
══════╪════════════════════════════╪════════════════════════════════╪═════════════════
═════════════╣
║2      │Thu Aug 09 12:33:44 IST 2018│DEPLOYED│logcf       │1.0.1
│Upgrade complete║
║1      │Thu Aug 09 12:29:02 IST 2018│DELETED │logcf       │1.0.0
│Delete complete ║
╚═══════════════╧════════════════════════════════════════════════════════════════════════╧
══════╧════════════════════════════╧════════════════════════════════╧═════════════════
═════════════╝

skipper:>manifest get a1
```

```
"apiVersion": "skipper.spring.io/v1"
"kind": "CloudFoundryApplication"
"spec":
  "resource": "maven://org.springframework.cloud.stream.app:log-sink-rabbit"
  "version": "1.3.1.RELEASE"
  "manifest":
    "memory": "1024"
    "disk-quota": "2048"
    "instances": "1"
    "services":
    - "rabbit"
    "timeout": "180"
```

```
$ cf apps
Getting apps in org scdf-ci / space space-ilaya as igopinathan@pivotal.io...
OK

name     requested state   instances   memory   disk   urls
a1-v2    started           1/1         1G       2G     a1-v2.cfapps.io
```

Rollback the `logcf` package with the version `1.0.1`

```
skipper:>release rollback a1
a1 has been rolled back.  Now at version v3.

skipper:>release list
╔═════╤═══════╤════════════════════════════╤════════╤════════════╤═══════
═══════════════╤═════════════════════════════╤════════════════════════
════════════════╤═══════════════════════════════╗
║Name │Version│        Last updated        │ Status │Package Name│Package
Version│Platform Name│Platform Status║
╠═════╪═══════╪════════════════════════════╪════════╪════════════╪═══════
═══════════════╪═════════════════════════════╪════════════════════════
════════════════╪═══════════════════════════════╣
║a1   │3      │Thu Aug 09 12:39:17 IST 2018│DEPLOYED│logcf       │1.0.0
│cf-dev       │                ║
╚═════╧═══════╧════════════════════════════╧════════╧════════════╧═══════
═══════════════╧═════════════════════════════╧════════════════════════
════════════════╧═══════════════════════════════╝

skipper:>release history a1
╔═══════╤════════════════════════════╤════════════════════════════════
══════╤═════════════════════════════╤════════════════════════════════
══════════════╗
║Version│        Last updated        │ Status │Package Name│Package Version│
Description   ║
╠═══════╪════════════════════════════╪════════════════════════════════
══════╪═════════════════════════════╪════════════════════════════════
══════════════╣
```

```
| 3       | Thu Aug 09 12:39:17 IST 2018 | DEPLOYED | logcf        | 1.0.0
| Rollback complete |
| 2       | Thu Aug 09 12:33:44 IST 2018 | DELETED  | logcf        | 1.0.1
| Delete complete   |
| 1       | Thu Aug 09 12:29:02 IST 2018 | DELETED  | logcf        | 1.0.0
| Delete complete   |
```

```
skipper:>manifest get a1
"apiVersion": "skipper.spring.io/v1"
"kind": "CloudFoundryApplication"
"spec":
  "resource": "maven://org.springframework.cloud.stream.app:log-sink-rabbit"
  "version": "1.3.0.RELEASE"
  "manifest":
    "memory": "1024"
    "disk-quota": "2048"
    "instances": "1"
    "services":
    - "rabbit"
    "timeout": "180"


$ cf apps
Getting apps in org scdf-ci / space space-ilaya as igopinathan@pivotal.io...
OK

name    requested state   instances   memory   disk   urls
a1-v3   started           1/1         1G       2G     a1-v3.cfapps.io
```

Upgrade the `logcf` package into the latest `1.0.1` version and also update the manifest's memory to `2G`.

```
skipper:>release upgrade --package-name logcf --release-name a1 --properties
"spec.manifest.memory=2G"
a1 has been upgraded.  Now at version v4.

skipper:>release list
```

```
| Name | Version |         Last updated         |  Status  | Package Name | Package
Version | Platform Name |           Platform Status           |
```

```
║a1  │4      │Thu Aug 09 12:49:49 IST 2018│DEPLOYED│logcf      │1.0.1
│cf-dev      │[a1-v4], State = [a1-v4-0=deployed]║
╚═══════════╧═══════════════════════════════════════════════════════════════════════════════════════
═══════════════════╧═══════════════════════════╧════════════════════════════════════╧═══════════════
═══════════════════╧════════════════════════════════════════════════════════════════════════════════
═══╝

skipper:>release history a1
╔═══════════════════╤═══════════════════════════════════════════════════════════════════╤═══════════
═══════╤═════════════════════════════╤════════════════════════════════════╤═══════════════════════
══════════════╗
║Version│        Last updated        │ Status  │Package Name│Package Version│
Description   ║
╠═══════════════════╪═══════════════════════════════════════════════════════════════════╪═══════════
═══════╪═════════════════════════════╪════════════════════════════════════╪═══════════════════════
══════════════╣
║4      │Thu Aug 09 12:49:49 IST 2018│DEPLOYED│logcf      │1.0.1
│Upgrade complete║
║3      │Thu Aug 09 12:39:17 IST 2018│DELETED │logcf      │1.0.0
│Delete complete ║
║2      │Thu Aug 09 12:33:44 IST 2018│DELETED │logcf      │1.0.1
│Delete complete ║
║1      │Thu Aug 09 12:29:02 IST 2018│DELETED │logcf      │1.0.0
│Delete complete ║
╚═══════════════════╧═══════════════════════════════════════════════════════════════════╧═══════════
═══════╧═════════════════════════════╧════════════════════════════════════╧═══════════════════════
══════════════╝

skipper:>manifest get a1
"apiVersion": "skipper.spring.io/v1"
"kind": "CloudFoundryApplication"
"spec":
  "resource": "maven://org.springframework.cloud.stream.app:log-sink-rabbit"
  "version": "1.3.1.RELEASE"
  "manifest":
    "memory": "2G"
    "disk-quota": "2048"
    "instances": "1"
    "services":
    - "rabbit"
    "timeout": "180"

$ cf apps
Getting apps in org scdf-ci / space space-ilaya as igopinathan@pivotal.io...
OK

name    requested state   instances   memory   disk   urls
a1-v4   started           1/1         2G       2G     a1-v4.cfapps.io
```

Delete the release

```
skipper:>release delete a1
a1 has been deleted.
```

The following example shows how Skipper helps managing any application that can be deployed into CF using manifest In this case, we have a couple of python packages that print the greeting messages.

Upload the python packages from the `spring-cloud-skipper-server-core` test directory

```
skipper:>package upload --path spring-cloud-skipper-server-
core/src/test/resources/repositories/binaries/test/python/python-printer-1.0.0.zip
Package uploaded successfully:[python-printer:1.0.0]

skipper:>package upload --path spring-cloud-skipper-server-
core/src/test/resources/repositories/binaries/test/python/python-printer-1.0.1.zip
Package uploaded successfully:[python-printer:1.0.1]
```

Install the python package

```
skipper:>package install --package-name python-printer --package-version 1.0.0
--release-name printer --platform-name cf-dev
Released printer. Now at version v1.
```

```
skipper:>manifest get printer
"apiVersion": "skipper.spring.io/v1"
"kind": "CloudFoundryApplication"
"spec":
  "resource":
"https://github.com/ilayaperumalg/sandbox/raw/master/python/1.0.0/hello.py-1.0.0.zip"
  "version": "1.0.0"
  "manifest":
    "memory": "1024"
    "disk-quota": "1024"
    "instances": "1"
    "health-check-type": "process"
    "buildpack": "python_buildpack"
    "timeout": "180"
    "command": "python hello.py"
```

```
$ cf apps
Getting apps in org scdf-ci / space space-ilaya as igopinathan@pivotal.io...
OK

name         requested state   instances   memory   disk   urls
printer-v1   started           1/1         1G       1G     printer-v1.cfapps.io
```

```
$ cf logs printer-v1
Retrieving logs for app printer-v1 in org scdf-ci / space space-ilaya as
igopinathan@pivotal.io...

   2018-08-09T13:33:36.55+0530 [APP/PROC/WEB/0] OUT Hello!
   2018-08-09T13:33:41.55+0530 [APP/PROC/WEB/0] OUT Hello!
```

Upgrade the python package with the version `1.0.1`

```
skipper:>release upgrade printer --package-name python-printer --package-version 1.0.1
printer has been upgraded.  Now at version v2.

skipper:>manifest get printer
"apiVersion": "skipper.spring.io/v1"
"kind": "CloudFoundryApplication"
"spec":
  "resource":
"https://github.com/ilayaperumalg/sandbox/raw/master/python/1.0.1/hello.py-1.0.1.zip"
  "version": "1.0.1"
  "manifest":
    "memory": "1024"
    "disk-quota": "1024"
    "instances": "1"
    "health-check-type": "process"
    "buildpack": "python_buildpack"
    "timeout": "180"
    "command": "python vanakkam.py"
```

```
$ cf apps
Getting apps in org scdf-ci / space space-ilaya as igopinathan@pivotal.io...
OK

name         requested state   instances   memory   disk   urls
printer-v2   started           1/1         1G       1G     printer-v2.cfapps.io
```

```
$ cf logs printer-v2
Retrieving logs for app printer-v2 in org scdf-ci / space space-ilaya as
igopinathan@pivotal.io...

   2018-08-09T13:36:13.39+0530 [APP/PROC/WEB/0] OUT Vanakkam!
   2018-08-09T13:36:18.40+0530 [APP/PROC/WEB/0] OUT Vanakkam!
```

Using Skipper

This section is the "'three-hour tour'" of Skipper. It describes how to configure and use the main feature set of Skipper in detail. We will cover the shell, platforms, packages, and repositories.

Feel free to reach out on Gitter for help and ask questions on Stack Overflow. Issues can be filed on Github issues.

Chapter 12. Skipper Shell

The shell is based on the Spring Shell project. Two of the shell's best features are tab-completion and colorization of commands. Use the 'help' command or the `--help` argument when starting the shell to get help information. The output of using the `--help` argument follows:

```
Skipper Options:

  --spring.cloud.skipper.client.serverUri=<uri>                          Address of the
Skipper Server [default: http://localhost:7577].
  --spring.cloud.skipper.client.username=<USER>                          Username of the
Skipper Server [no default].
  --spring.cloud.skipper.client.password=<PASSWORD>                      Password of the
Skipper Server [no default].
  --spring.cloud.skipper.client.credentials-provider-command=<COMMAND> Runs an
external command, which must return an OAuth Access Token [no default].
  --spring.cloud.skipper.client.skip-ssl-validation=<true|false>       Accept any SSL
certificate (even self-signed) [default: no].

  --spring.shell.historySize=<SIZE>                Default size of the shell log file
[default: 3000].
  --spring.shell.commandFile=<FILE>                Skipper Shell read commands read
from the file(s) and then exits.

  --help                                           This message.
```

12.1. Shell Modes

The shell can be started in either interactive or non-interactive mode. In the case of the non-interactive mode, command line arguments are run as Skipper commands, and then the shell exits. If there are any arguments that do not have the prefix `spring.cloud.skipper.client`, they are considered as skipper commands to run.

Consider the following example:

```
java -jar spring-cloud-skipper-shell-2.5.0-SNAPSHOT.jar
--spring.cloud.skipper.client.serverUri=http://localhost:9123/api
```

The preceding example brings up the interactive shell and connects to `localhost:9123/api`. Now consider the following command:

```
$ java -jar spring-cloud-skipper-shell-2.5.0-SNAPSHOT.jar
--spring.cloud.skipper.client.serverUri=http://localhost:9123/api search
```

The preceding command connects to `localhost:9123/api`, runs the `search` command, and then exits.

A more common use case would be to update a package from within a CI job — for example, in a Jenkins Stage, as shown in the following example:

```
stage ('Build') {
    steps {
        checkout([
            $class: 'GitSCM',
            branches: [
                [name: "*/master"]
            ],
            userRemoteConfigs: [
                [url: "https://github.com/markpollack/skipper-samples.git"]
            ]
        ])
        sh '''
            VERSION="1.0.0.M1-$(date +%Y%m%d_%H%M%S)-VERSION"
            mvn org.codehaus.mojo:versions-maven-plugin:2.3:set
-DnewVersion="${VERSION}"
            mvn install
            java -jar /home/mpollack/software/skipper.jar upgrade --package-name
helloworld --release-name helloworld-jenkins --properties version=${VERSION}
        '''
    }
}
```

Chapter 13. Platforms

Skipper supports deploying to multiple platforms. The platforms included are Local, Cloud Foundry, and Kubernetes. For each platform, you can configure multiple accounts. Each `account name` must be globally unique across all platforms.

Usually, different `accounts` correspond to different orgs or spaces for Cloud Foundry and to different namespaces for a single Kubernetes cluster.

Platforms are defined by using Spring Boot's Externalized Configuration feature. To simplify the getting started experience, if a local platform account is not defined in your configuration, Skipper creates a `local` deployer implementation named `default`.

You can make use of the Encryption and Decryption features of Spring Cloud Config as one way to secure credentials.

Distinct from where Skipper deploys the application, you can also run the Skipper server itself on a platform. Installation on other platforms is covered in the Installation section.

The following example YAML file shows configuration of all three platforms:

```
spring:
  cloud:
    skipper:
      server:
        platform:
          local:
            accounts:
              localDevDebug:
                javaOpts: "-Xdebug"
          cloudfoundry:
            accounts:
              cf-dev:
                connection:
                  url: https://api.run.pivotal.io
                  org: scdf-ci
                  space: space-mark
                  domain: cfapps.io
                  username: <your-username>
                  password: <your-password>
                  skipSslValidation: false
                deployment:
                  deleteRoutes: false
          kubernetes:
            accounts:
              minikube:
                namespace: default
```

The properties available for each platform can be found in the following classes:

- LocalDeployerProperties.
- CloudFoundryDeploymentProperties for `deployment:` and CloudFoundryConnectionProperties for the `connection:`.
- KubernetesDeployerProperties

Chapter 14. Packages

Packages contain all the necessary information to install your application or group of applications. The approach to describing the applications is to use a YAML file that provides all the necessary information to help facilitate searching for your application hosted in a Package Registry and to install your application to a platform.

To make it easy to customize a package, the YAML files are templated. The final version of the YAML file, with all values substituted, is known as the release `manifest`. Skipper currently understands how to deploy applications based off a YAML file that contains the information needed for a *Spring Cloud Deployer* or *Cloud Foundry* implementation to deploy an application. It describes where to find the application (an HTTP, Maven or Docker location), application properties (think Spring Boot `@ConfigurationProperties`), and deployment properties (such as how much memory to use).

14.1. Package Format

A package is a collection of YAML files that are zipped up into a file with the following naming convention: `[PackageName]-[PackageVersion].zip` (for example: `mypackage-1.0.0.zip`).

A package can define a single application or a group of applications.

14.1.1. Single Application

The single application package file, `mypackage-1.0.0.zip`, when unzipped, should have the following directory structure:

```
mypackage-1.0.0
 ├──── package.yml
 ├──── templates
 │     └──── template.yml
 └──── values.yml
```

The `package.yml` file contains metadata about the package and is used to support Skipper's search functionality. The `template.yml` file contains placeholders for values that are specified in the `values.yml` file. When installing a package, placeholder values can also be specified, and they would override the values in the `values.yml` file. The templating engine that Skipper uses is JMustache. The YAML files can have either `.yml` or `.yaml` extensions.

The helloworld-1.0.0.zip or helloworld-docker-1.0.0.zip files are good examples to use as a basis to create your own package "'by hand'".

The source code for the `helloworld` sample can be found here.

14.1.2. Multiple Applications

A package can contain a group of applications bundled in it. In those cases, the structure of the package would resemble the following:

```
mypackagegroup-1.0.0
├──── package.yml
├──── packages
│     ├──── app1
│     │     ├──── package.yml
│     │     ├──── templates
│     │     │     └──── log.yml
│     │     └──── values.yml
│     └──── app2
│           ├──── package.yml
│           ├──── templates
│           │     └──── time.yml
│           └──── values.yml
└──── values.yml
```

In the preceding example, the `mypackagegroup` still has its own `package.yml` and `values.yml` to specify the package metadata and the values to override. All the applications inside the `mypackagegroup` are considered to be sub-packages and follow a package structure similar to the individual packages. These sub packages need to be specified inside the `packages` directory of the root package, `mypackagegroup`.

The ticktock-1.0.0.zip file is a good example to use as a basis for creating your own package 'by-hand'.

Packages with template kind *CloudFoundryApplication* currently doesn't support multiple applications format.

14.2. Package Metadata

The `package.yml` file specifies the package metadata. A sample package metadata would resemble the following:

```
# Required Fields
apiVersion: skipper.spring.io/v1
kind: SkipperPackageMetadata
name: mypackage
version: 1.0.0

# Optional Fields
packageSourceUrl: https://github.com/some-mypackage-project/v1.0.0.RELEASE
packageHomeUrl: https://some-mypackage-project/
tags: skipper, mypackage, sample
maintainer: https://github.com/maintainer
description: This is a mypackage sample.
```

Required Fields:

- `apiVersion`: The Package Index spec version this file is based on.
- `kinds`: What type of package system is being used.
- `name`: The name of the package.
- `version`: The version of the package.

Currently only supported *kind* is **SkipperPackageMetadata**.

Optional Fields:

- `packageSourceUrl`: The location of the source code for this package.
- `packageHomeUrl`: The home page of the package.
- `tags`: A comma-separated list of tags to be used for searching.
- `maintainer`: Who maintains this package.
- `description`: Free-form text describing the functionality of the package — generally shown in search results.
- `sha256`: The hash of the package binary (not yet enforced).
- `iconUrl`: The URL for an icon to show for this package.
- `origin`: Free-form text describing the origin of this package — for example, your company name.

Currently, the package search functionality is only a wildcard match against the name of the package.

A Package Repository exposes an `index.yml` file that contains multiple metadata documents and that uses the standard three dash notation `---` to separate the documents — for example, index.yml.

14.3. Package Templates

Currently, two type of applications are supported. One having `SpringCloudDeployerApplication` kind, which means the applications can be deployed into the target platforms only by using their corresponding Spring Cloud Deployer implementations (CF, Kubernetes Deployer, and so on). Other is having `CloudFoundryApplication` kind, which means the applications are directly deployed into *Cloud Foundry* using its manifest support.

14.3.1. Spring Cloud Deployer

The `template.yml` file has a package structure similar to that of the following example:

```
mypackage-1.0.0
 ├──── package.yml
 ├──── templates
 │     └──── template.yml
 └──── values.yml
```

Actual template file name doesn't matter and you can have multiple template files. These just need to be inside of a `templates` directory.

```
# template.yml
apiVersion: skipper.spring.io/v1
kind: SpringCloudDeployerApplication
metadata:
  name: mypackage
  type: sample
spec:
  resource: maven://org.mysample:mypackage
  resourceMetadata:  maven://org.mysample:mypackage:jar:metadata:{{spec.version}}
  version: {{spec.version}}
  applicationProperties:
    {{#spec.applicationProperties.entrySet}}
    {{key}}: {{value}}
    {{/spec.applicationProperties.entrySet}}
  deploymentProperties:
    {{#spec.deploymentProperties.entrySet}}
    {{key}}: {{value}}
    {{/spec.deploymentProperties.entrySet}}
```

The `apiVersion`, `kind`, and `spec.resource` are required.

The `spec.resource` and `spec.version` define where the application executable is located. The `spec.resourceMetadata` field defines where a Spring Boot Configuration metadata jar is located that contains the configuration properties of the application. This is either a Spring Boot uber jar hosted under a HTTP endpoint or a Maven or Docker repository. The template placeholder `{{spec.version}}` exists so that the version of a specific application can be easily upgraded without having to create a new package .zip file.

The `resource` is based on `http://` or `maven://` or `docker:`. The format for specifying a `resource` follows documented types in Resources.

14.3.2. Cloud Foundry

The `template.yml` file has a package structure similar to that of the following example:

```
mypackage-1.0.0
 ├──── package.yml
 ├──── templates
 │     └──── template.yml
 └──── values.yml
```

`template.yml` commonly has content similar to the following:

Actual template file name doesn't matter and you can have multiple template files. These just need to be inside of a `templates` directory.

```
# template.yml
apiVersion: skipper.spring.io/v1
kind: CloudFoundryApplication
spec:
  resource: maven://org.mysample:mypackage
  version: {{spec.version}}
  manifest:
    {{#spec.manifest.entrySet}}
    {{key}}: {{value}}
    {{/spec.manifest.entrySet}}
```

Where values could for example be something like:

```
# values.yml
spec:
  version: 1.0.0
  manifest:
    memory: 1024
    disk-quota: 1024
```

Possible values of a `spec.manifest` are:

Key	Value	Notes
`buildpack`	(String)	*buildpack* attribute as is.
`command`	(String)	*command* attribute as is.
`memory`	(String or Integer)	*memory* attribute as is if type is Integer, String is converted using same format in a CF, like `1024M` or `2G`. `1024` and `1024M` are equivalent.
`disk-quota`	(String or Integer)	*disk_quota* attribute as is if type is Integer, String is converted using same format in a CF, like `1024M` or `2G`. `1024` and `1024M` are equivalent.
`timeout`	(Integer)	*timeout* attribute as is.
`instances`	(Integer)	*instances* attribute as is.
`no-hostname`	(Boolean)	*no-hostname* attribute as is.
`no-route`	(Boolean)	*no-route* attribute as is.
`random-route`	(Boolean)	*random-route* attribute as is.
`health-check-type`	(String)	*health-check-type* having possible values of `port`, `process` or `http`.
`health-check-http-endpoint`	(String)	*health-check-http-endpoint* attribute as is.

Key	Value	Notes
`stack`	(String)	*stack* attribute as is.
`services`	(List<String>)	*services* attribute as is.
`domains`	(List<String>)	*domains* attribute as is.
`hosts`	(List<String>)	*hosts* attribute as is.
`env`	(Map<String,Object>)	*env* attribute as is.

Remember that when a value is given from a command-line, replacement happens as is defined in a template. Using a template format `{{#spec.manifest.entrySet}}` shown above, *List* would be given in format `spec.manifest.services=[service1, service2]` and *Map* would be given in format `spec.manifest.env={key1: value1, key2: value2}`.

The `resource` is based on `http://` or `maven://` or `docker:`. The format for specifying a `resource` follows documented types in Resources.

14.3.3. Resources

This section contains resource types currently supported.

HTTP Resources

The following example shows a typical spec for HTTP:

```
spec:
  resource: https://example.com/app/hello-world
  version: 1.0.0.RELEASE
```

There is a naming convention that must be followed for HTTP-based resources so that Skipper can assemble a full URL from the `resource` and `version` field and also parse the version number given the URL. The preceding `spec` references a URL at `example.com/app/hello-world-1.0.0.RELEASE.jar`. The `resource` and `version` fields should not have any numbers after the `-` character.

Docker Resources

The following example shows a typical spec for Docker:

```
spec:
  resource: docker:springcloud/spring-cloud-skipper-samples-helloworld
  version: 1.0.0.RELEASE
```

The mapping to docker registry names follows:

```
spec:
  resource: docker:<user>/<repo>
  version: <tag>
```

Maven Resources

The following example shows a typical spec for Maven:

```
spec:
  resource: maven://org.springframework.cloud.samples:spring-cloud-skipper-samples-
helloworld:1.0.0.RELEASE
  version: 1.0.0.RELEASE
```

The mapping to Maven artifact names follows

```
spec:
  resource: maven://<maven-group-name>:<maven-artifact-name>
  version:<maven-version>
```

There is only one setting to specify with Maven repositories to search. This setting applies across all platform accounts. By default, the following configuration is used:

```
maven:
  remoteRepositories:
    springRepo: https://repo.spring.io/libs-snapshot
```

You can specify other entries and also specify proxy properties. This is currently best documented here. Essentially, this needs to be set as a property in your launch properties or `manifest.yml` (when pushing to PCF), as follows:

```
# manifest.yml
...
env:
    SPRING_APPLICATION_JSON: '{"maven": { "remote-repositories": { "springRepo": {
"url": "https://repo.spring.io/libs-snapshot"} } } }'
...
```

The metadata section is used to help search for applications after they have been installed. This feature will be made available in a future release.

The `spec` contains the resource specification and the properties for the package.

The `resource` represents the resource URI to download the application from. This would typically be a Maven co-ordinate or a Docker image URL.

The `SpringCloudDeployerApplication` kind of application can have `applicationProperties` and `deploymentProperties` as the configuration properties.

The application properties correspond to the properties for the application itself.

The deployment properties correspond to the properties for the deployment operation performed by Spring Cloud Deployer implementations.

The `name` of the template file can be anything, as all the files under `templates` directory are loaded to apply the template configurations.

14.4. Package Values

The `values.yml` file contains the default values for any of the keys specified in the template files.

For instance, in a package that defines one application, the format is as follows:

```
version: 1.0.0.RELEASE
spec:
  applicationProperties:
    server.port: 9090
```

If the package defines multiple applications, provide the name of the package in the top-level YML section to scope the `spec` section. Consider the example of a multiple application package with the following layout:

```
ticktock-1.0.0/
├──── packages
│     ├──── log
│     │     ├──── package.yml
│     │     └──── values.yml
│     └──── time
│           ├──── package.yml
│           └──── values.yml
├──── package.yml
└──── values.yml
```

The top-level `values.yml` file might resemble the following:

```
#values.yml

hello: world

time:
  appVersion: 1.3.0.M1
  deployment:
    applicationProperties:
      log.level: WARN
      trigger.fixed-delay: 1
log:
  deployment:
    count: 2
    applicationProperties:
      log.level: WARN
      log.name: skipperlogger
```

The preceding `values.yml` file sets `hello` as a variable available to be used as a placeholder in the `packages\log\values.yml` file and the `packages\time\values.yml`. However, the YML section under `time:` is applied only to the `packages\time\values.yml` file and the YML section under `log:` is applied only to the `packages\log\values.yml` file.

14.5. Package Upload

After creating the package in the structure shown in the previous section, we can compress it in a zip file with the following naming scheme: [PackageName]-[PackageVersion].zip (for example, mypackage-1.0.0.zip).

For instance, the package directory would resemble the following before compression:

```
mypackage-1.0.0
 ├──── package.yml
 ├──── templates
 │     └──── template.yml
 └──── values.yml
```

The zip file can be uploaded into one of the local repositories of the Skipper server. By default, the Skipper server has a local repository with the name, `local`.

By using the Skipper shell, we can upload the package zip file into the Skipper server's local repository, as follows:

```
skipper:>package upload --path /path-to-package/mypackage-1.0.0.zip
Package uploaded successfully:[mypackage:1.0.0]
```

If no `--repo-name` is set, the `upload` command uses `local` as the repository to upload.

We can then use the `package list` or `package search` command to see that our package has been uploaded, as shown (with its output) in the following example:

```
skipper:>package list
╔═════════════════╤════════╤════════════════════════════════════════════════════════════════════════════
════════════════════════════════════════════════════════════════════════════════════════════════════════
══════════════════════════════════════════╗
║      Name       │Version │                                   Description
║
╠═════════════════╪════════╪════════════════════════════════════════════════════════════════════════════
════════════════════════════════════════════════════════════════════════════════════════════════════════
══════════════════════════════════════════╣
║helloworld       │1.0.0   │The app has two endpoints, /about and /greeting in
English.  Maven resource.     ║
║helloworld       │1.0.1   │The app has two endpoints, /about and /greeting in
Portuguese.  Maven resource.  ║
║helloworld-docker│1.0.0   │The app has two endpoints, /about and /greeting in
English.  Docker resource.    ║
║helloworld-docker│1.0.1   │The app has two endpoints, /about and /greeting in
Portuguese.  Docker resource.║
║mypackage        │1.0.0   │This is a mypackage sample
║
╚═════════════════╧════════╧════════════════════════════════════════════════════════════════════════════
════════════════════════════════════════════════════════════════════════════════════════════════════════
══════════════════════════════════════════╝
```

14.6. Creating Your Own Package

In this section, we create a package that can be deployed by using Spring Cloud Deployer implementations.

For this package, we are going to create a simple package and upload it to our local machine.

To get started creating your own package, create a folder following a naming convention of `[package-name]-[package-version]`. In our case, the folder name is `demo-1.0.0`. In this directory, create empty files named `values.yml` and `package.yml` and create a `templates` directory. In the `templates` directory, create an empty file named `template.yml`.

Go into the `package.yml` where we are going to specify the package metadata. For this app, we fill only the minimum values possible, as shown in the following example:

```
# package.yml

apiVersion: skipper.spring.io/v1
kind: SkipperPackageMetadata
name: demo
version: 1.0.0
description: Greets the world!
```

Ensure that your `name` and `version` matches the `name` and `version` in your folder name, or you get an error.

Next, open up your `templates/template.yml` file. Here, we are going to specify the actual information about your package and, most importantly, set default values. In the `template.yml`, copy the template for the kind `SpringCloudDeployerApplication` from the preceding sample. Your resulting `template.yml` file should resemble the following:

```
# templates/template.yml

apiVersion: skipper.spring.io/v1
kind: SpringCloudDeployerApplication
metadata:
  name: demo
spec:
  resource: maven://org.springframework.cloud.samples:spring-cloud-skipper-samples-
helloworld
  version: {{version}}
  applicationProperties:
    {{#spec.applicationProperties.entrySet}}
    {{key}}: {{value}}
    {{/spec.applicationProperties.entrySet}}
  deploymentProperties:
    {{#spec.deploymentProperties.entrySet}}
    {{key}}: {{value}}
    {{/spec.deploymentProperties.entrySet}}
```

The preceding example file specifies that our application name is `demo` and finds our package in Maven. Now we can specify the `version`, `applicationProperties`, and `deploymentProperties` in our `values.yml`, as follows:

```
# values.yml

# This is a YAML-formatted file.
# Declare variables to be passed into your templates
version: 1.0.0.RELEASE
spec:
  applicationProperties:
    server.port: 8100
```

The preceding example sets the `version` to `1.0.0.RELEASE` and also sets the `server.port=8100` as one of the application properties. When the Skipper Package reader resolves these values by merging the `values.yml` against the template, the resolved values resemble the following:

```
# hypothetical template.yml

apiVersion: skipper.spring.io/v1
kind: SpringCloudDeployerApplication
metadata:
  name: demo
spec:
  resource: maven://org.springframework.cloud.samples:spring-cloud-skipper-samples-
helloworld
  version: 1.0.0.RELEASE
  applicationProperties:
    server.port: 8100
  deploymentProperties:
```

The reason to use `values.yml` instead of entering the values directly is that it lets you overwrite the values at run time by using the `--file` or `--properties` flags.

We have finished making our file. Now we have to zip it up. The easiest way to do is by using the `zip -r` command on the command line, as follows:

```
$ zip -r demo-1.0.0.zip demo-1.0.0/
  adding: demo-1.0.0/ (stored 0%)
  adding: demo-1.0.0/package.yml (deflated 14%)
  adding: demo-1.0.0/templates/ (stored 0%)
  adding: demo-1.0.0/templates/template.yml (deflated 55%)
  adding: demo-1.0.0/values.yml (deflated 4%)
```

Armed with our zipped file and the path to it, we can head to Skipper and use the `upload` command, as follows:

```
skipper:>package upload --path /Users/path-to-your-zip/demo-1.0.0.zip
Package uploaded successfully:[demo:1.0.0]
```

Now you can search for it as shown previously and then install it, as follows

```
skipper:>package install --package-name demo --package-version 1.0.0 --release-name
demo
Released demo. Now at version v1.
```

Congratulations! You have now created, packaged, uploaded, and installed your own Skipper package!

Chapter 15. Repositories

Repositories store package metadata and host package .zip files. Repositores can be local or remote, were local means backed by Skipper's relational database and remote means a filesystem exposed over HTTP.

When registering a remote registry (for example, the `experimental` one that is currently not defined by default in addition to one named local `), use the following format:

```
spring
  cloud:
    skipper:
      server:
        package-repositories:
          experimental:
            url: https://skipper-repository.cfapps.io/repository/experimental
            description: Experimental Skipper Repository
            repoOrder: 0
          local:
            url: http://${spring.cloud.client.hostname}:7577
            local: true
            description: Default local database backed repository
            repoOrder: 1
```

For Skipper 2.x, `spring.cloud.skipper.server.package-repositories` structure has been changed from a list to a map where key is the repository name. Having a map format makes it easier to define and override configuration values.

The `repoOrder` determines which repository serves up a package if one with the same name is registered in two or more repositories.

The directory structure assumed for a remote repository is the registered `url` value followed by the package name and then the zip file name (for example, `skipper-repository.cfapps.io/repository/experimental/helloworld/helloworld-1.0.0.zip` for the package `helloworld` with a version of `1.0.0`). A file named `index.yml` is expected to be directly under the registered `url` — for example, skipper-repository.cfapps.io/repository/experimental/index.yml. This file contains the package metadata for all the packages hosted by the repository.

It is up to you to update the `index.yml` file "'by hand'" for remote repositories.

'Local' repositories are backed by Skipper's database. In the Skipper 1.0 release, they do not expose the index.yml or the .zip files under a filesystem-like URL structure as with remote repositories. This feature will be provided in the next version. However, you can upload packages to a local repository and do not need to maintain an index file. See the "Skipper Commands" section for information on creating local repositories.

A good example that shows using a Spring Boot web application with static resources to host a Repository can be found here. This application is currently running under skipper-

repository.cfapps.io/repository/experimental.

Installation

Chapter 16. Installing on a Local Platform

16.1. Local Platform configuration

The following example YAML file configures two local deployer accounts, named `localDev` and `localDevDebug`:

```
spring:
  cloud:
    skipper:
      server:
        platform:
          local:
            accounts:
              localDev:
                shutdownTimeout: 60
                javaOpts: "-Dtest=foo"
              localDevDebug:
                javaOpts: "-Xdebug"
```

The key-value pairs that follow the name of the account are `javaCmd`, `workingDirectoriesRoot`, `deleteFilesOnExit`, `envVarsToInherit`, `shutdownTimeout`, `javaOpts`, and `useSpringApplicationJson`. More information can be found in the JavaDocs for LocalDeployerProperties.

Chapter 17. Installing on Cloud Foundry

This section contains an example YAML file that configures two Cloud Foundry accounts, named `cf-dev` and `cf-qa`. This is useful on Cloud Foundry if you use the Spring Cloud Config Server to manage Skipper's configuration properties.

17.1. Cloud Foundry Configuration

You can modify the following sample YML snippet to fit your needs:

```
spring:
  cloud:
    skipper:
      server:
        platform:
          cloudfoundry:
            accounts:
              cf-dev:
                connection:
                  url: https://api.run.pivotal.io
                  org: myOrg
                  space: mySpace
                  domain: cfapps.io
                  username: cf-dev@example.com
                  password: drowssap
                  skipSslValidation: false
                deployment:
                  memory: 2048m
                  disk: 2048m
                  services: rabbit
                  deleteRoutes: false
              cf-qa:
                connection:
                  url: https://api.run.pivotal.io
                  org: myOrgQA
                  space: mySpaceQA
                  domain: cfapps.io
                  username: cf-qa@example.com
                  password: drowssap
                  skipSslValidation: true
                deployment:
                  memory: 1024m
                  disk: 1024m
                  services: rabbitQA
                  deleteRoutes: false
```

The `deleteRoutes` deployment setting is `false` so that "v2" of an application has the same route as "v1". Otherwise, undeploying "v1" removes the route.

You can also run the Skipper server locally and deploy to Cloud Foundry. In this case, it is more convenient to specify the configuration in a `skipper.yml` file and start the server with the `--spring.config.additional-location=skipper.yml` option.

If you use `cf push` to deploy Skipper, a Cloud Foundry manifest is more appropriate to use. You can modify the following sample manifest.yml to fit your needs:

```
applications:
- name: mlp-skipper
  host: mlp-skipper
  memory: 1G
  disk_quota: 1G
  timeout: 180
  instances: 1
  buildpack: java_buildpack
  path: spring-cloud-skipper-server.jar
env:
    SPRING_APPLICATION_NAME: mlp-skipper
    JBP_CONFIG_SPRING_AUTO_RECONFIGURATION: '{enabled: false}'
    SPRING_CLOUD_SKIPPER_SERVER_PLATFORM_CLOUDFOUNDRY_ACCOUNTS[pws]_CONNECTION_URL:
https://api.run.pivotal.io
    SPRING_CLOUD_SKIPPER_SERVER_PLATFORM_CLOUDFOUNDRY_ACCOUNTS[pws]_CONNECTION_ORG:
myOrgQA
    SPRING_CLOUD_SKIPPER_SERVER_PLATFORM_CLOUDFOUNDRY_ACCOUNTS[pws]_CONNECTION_SPACE:
mySpaceQA
    SPRING_CLOUD_SKIPPER_SERVER_PLATFORM_CLOUDFOUNDRY_ACCOUNTS[pws]_CONNECTION_DOMAIN:
cfapps.io

SPRING_CLOUD_SKIPPER_SERVER_PLATFORM_CLOUDFOUNDRY_ACCOUNTS[pws]_CONNECTION_USERNAME:
cf-qa@example.com

SPRING_CLOUD_SKIPPER_SERVER_PLATFORM_CLOUDFOUNDRY_ACCOUNTS[pws]_CONNECTION_PASSWORD:
drowssap

SPRING_CLOUD_SKIPPER_SERVER_PLATFORM_CLOUDFOUNDRY_ACCOUNTS[pws]_CONNECTION_SKIPSSLVALI
DATION: false

SPRING_CLOUD_SKIPPER_SERVER_PLATFORM_CLOUDFOUNDRY_ACCOUNTS[pws]_DEPLOYMENT_DELETEROUTE
S: false

SPRING_CLOUD_SKIPPER_SERVER_PLATFORM_CLOUDFOUNDRY_ACCOUNTS[pws]_DEPLOYMENT_SERVICES:
rabbitmq
services:
- mysqlboost
```

In the preceding manifest, we bound the application to the `mysqlboost` service. If you do not specify a service, the server uses an embedded database.

As of Skipper 2.0, you must disable Spring Auto-reconfiguration and set the profile to `cloud`.

You must set `SPRING_CLOUD_SKIPPER_SERVER_PLATFORM_CLOUDFOUNDRY_ACCOUNTS[pws]_DEPLOYMENT_DELETEROUTES: false` so that "v2" of an application has the same route as "v1". Otherwise, undeploying "v1" removes the route.

You must set `SPRING_CLOUD_SKIPPER_SERVER_PLATFORM_CLOUDFOUNDRY_ACCOUNTS[pws]_DEPLOYMENT_SERVICES` property that binds the specified services to each of the deployed applications.

You can find information on the deployment properties that you can configure in CloudFoundryDeploymentProperties.

When starting the Skipper shell on your local machine, it tries to connect to the Server at the default location of `localhost:7577/api`. Use the shell's `--spring.cloud.skipper.client.serverUri` command line option to specify the location of the server. You can alternatively use the `config` interactive shell command to set the server location, as follows:

```
server-unknown:>skipper config --uri https://mlp-skipper.cfapps.io/api
Successfully targeted https://mlp-skipper.cfapps.io/api
skipper:>
```

17.2. Database Connection Pool

As of Skipper 2.0, the Spring Cloud Connector library is no longer used to create the DataSource. The library java-cfenv is now used which allows you to set Spring Boot properties to configure the connection pool.

17.3. Maximum Disk Quota

By default, every application in Cloud Foundry starts with 1G disk quota and this can be adjusted to a default maximum of 2G. The default maximum can also be overridden up to 10G by using Pivotal Cloud Foundry's (PCF) Ops Manager GUI.

This configuration is relevant for Spring Cloud Skipper because every deployment is composed of applications (typically Spring Boot uber-jar's), and those applications are resolved from a remote maven repository. After resolution, the application artifacts are downloaded to the local Maven Repository for caching and reuse. With this happening in the background, the default disk quota (1G) can fill up rapidly, especially when we experiment with streams that are made up of unique applications. In order to overcome this disk limitation and depending on your scaling requirements, you may want to change the default maximum from 2G to 10G. Let's review the steps to change the default maximum disk quota allocation.

From PCF's Ops Manager, select the "Pivotal Elastic Runtime" tile and navigate to the "Application

Developer Controls" tab. Change the "Maximum Disk Quota per App (MB)" setting from 2048 (2G) to 10240 (10G). Save the disk quota update and click "Apply Changes" to complete the configuration override.

17.4. Managing Disk Use

Even when configuring Skipper to use 10G of space, there is the possibility of exhausting the available space on the local disk. To prevent this, `jar` artifacts downloaded from external sources, i.e., apps registered as `http` or `maven` resources, are automatically deleted whenever the application is deployed, whether or not the deployment request succeeds. This behavior is optimal for production environments in which container runtime stability is more critical than I/O latency incurred during deployment. In development environments deployment happens more frequently. Additionally, the `jar` artifact (or a lighter `metadata` jar) contains metadata describing application configuration properties which is used by various operations related to application configuration, more frequently performed during pre-production activities. To provide a more responsive interactive developer experience at the expense of more disk usage in pre-production environments, you can set the CloudFoundry deployer property `autoDeleteMavenArtifacts` to `false`.

If you deploy the Skipper by using the default `port` health check type, you must explicitly monitor the disk space on the server in order to avoid running out space. If you deploy the server by using the `http` health check type (see the next example), the server is restarted if there is low disk space. This is due to Spring Boot's Disk Space Health Indicator. You can configure the settings of the Disk Space Health Indicator by using the properties that have the `management.health.diskspace` prefix.

For version 1.7, we are investigating the use of Volume Services for the server to store `.jar` artifacts before pushing them to Cloud Foundry.

The following example shows how to deploy the `http` health check type to an endpoint called `/management/health`:

```
---
  ...
  health-check-type: http
  health-check-http-endpoint: /management/health
```

Chapter 18. Installing on Kubernetes

A docker image, named `springcloud/spring-cloud-skipper-server`, is available for Skipper server in dockerhub. You can use this image to run the Skipper server in Kubernetes.

18.1. Kuberenetes configuration

The following example YAML file configures two accounts, named `k8s-dev` and `k8sqa`, on a Kubernetes cluster.

```
spring:
  cloud:
    skipper:
      server:
        platform:
          kubernetes:
            accounts:
              k8s-dev:
                namespace: devNamespace
                cpu: 4
              k8s-qa:
                namespace: qaNamespace
                memory: 1024m
```

The accounts correspond to different namespaces. We are investigating how to support connecting to different Kubernetes clusters.

You can find more information on the deployment properties that you can configure in KubernetesDeployerProperties

Chapter 19. Database configuration

A relational database is used to store stream and task definitions as well as the state of tasks that have been run. Spring Cloud Skipper provides schemas for **H2**, **MySQL**, **Oracle**, **PostgreSQL**, **Db2**, and **SQL Server**. The schema is automatically created when the server starts.

By default, Spring Cloud Skipper offers an embedded instance of the **H2** database. The **H2** database is good for development purposes but is not recommended for production use.

H2 database in Server Mode is not supported, only Embedded Mode.

The JDBC drivers for **MySQL** (through the MariaDB driver), **PostgreSQL**, **SQL Server**, and embedded **H2** are available without additional configuration. If you are using any other database, then you need to put the corresponding JDBC driver jar on the classpath of the server.

The database properties can be passed as environment variables or command-line arguments to the Skipper Server.

19.1. MySQL

The following example shows how to define a MySQL database connection using MariaDB driver.

```
java -jar spring-cloud-skipper-server-2.5.0-SNAPSHOT.jar \
    --spring.datasource.url=jdbc:mysql://localhost:3306/mydb \
    --spring.datasource.username=<user> \
    --spring.datasource.password=<password> \
    --spring.datasource.driver-class-name=org.mariadb.jdbc.Driver
```

MySQL versions up to *5.7* can be used with a MariaDB driver. Starting from version *8.0* MySQL's own driver has to be used.

```
java -jar spring-cloud-skipper-server-2.5.0-SNAPSHOT.jar \
    --spring.datasource.url=jdbc:mysql://localhost:3306/mydb \
    --spring.datasource.username=<user> \
    --spring.datasource.password=<password> \
    --spring.datasource.driver-class-name=com.mysql.jdbc.Driver
```

Due to licensing restrictions we're unable to bundle the MySQL driver. You need to add it to server's classpath yourself.

19.2. MariaDB

The following example shows how to define a MariaDB database connection with command Line arguments

```
java -jar spring-cloud-skipper-server-server-2.5.0-SNAPSHOT.jar \
    --spring.datasource.url=jdbc:mariadb://localhost:3306/mydb?useMysqlMetadata=true \
    --spring.datasource.username=<user> \
    --spring.datasource.password=<password> \
    --spring.datasource.driver-class-name=org.mariadb.jdbc.Driver
```

Starting with MariaDB v2.4.1 connector release, it is required to also add `useMysqlMetadata=true` to the JDBC URL. This is a required workaround until the time when MySQL and MariaDB are considered to be two different databases.

MariaDB version *10.3* introduced a support for real database sequences which is yet another breaking change while toolings around these databases fully support MySQL and MariaDB as a separate database types. A workaround is to use an older hibernate dialect which doesn't try to use sequences.

```
java -jar spring-cloud-spring-cloud-skipper-server-server-2.5.0-SNAPSHOT.jar \
    --spring.datasource.url=jdbc:mariadb://localhost:3306/mydb?useMysqlMetadata=true \
    --spring.datasource.username=<user> \
    --spring.datasource.password=<password> \
    --spring.jpa.properties.hibernate.dialect=org.hibernate.dialect.MariaDB102Dialect
\
    --spring.datasource.driver-class-name=org.mariadb.jdbc.Driver
```

19.3. PostgreSQL

The following example shows how to define a PostgreSQL database connection with command line arguments:

```
java -jar spring-cloud-skipper-server-2.5.0-SNAPSHOT.jar \
    --spring.datasource.url=jdbc:postgresql://localhost:5432/mydb \
    --spring.datasource.username=<user> \
    --spring.datasource.password=<password> \
    --spring.datasource.driver-class-name=org.postgresql.Driver
```

19.4. SQL Server

The following example shows how to define a SQL Server database connection with command line arguments:

```
java -jar spring-cloud-skipper-server-2.5.0-SNAPSHOT.jar \
    --spring.datasource.url='jdbc:sqlserver://localhost:1433;databaseName=mydb' \
    --spring.datasource.username=<user> \
    --spring.datasource.password=<password> \
    --spring.datasource.driver-class-name=com.microsoft.sqlserver.jdbc.SQLServerDriver
```

19.5. Db2

The following example shows how to define a Db2 database connection with command line arguments:

```
java -jar spring-cloud-skipper-server-2.5.0-SNAPSHOT.jar \
    --spring.datasource.url=jdbc:db2://localhost:50000/mydb \
    --spring.datasource.username=<user> \
    --spring.datasource.password=<password> \
    --spring.datasource.driver-class-name=com.ibm.db2.jcc.DB2Driver
```

Due to licensing restrictions we're unable to bundle Db2 driver. You need to add it to server's classpath yourself.

19.6. Oracle

The following example shows how to define a Oracle database connection with command line arguments:

```
java -jar spring-cloud-skipper-server-2.5.0-SNAPSHOT.jar \
    --spring.datasource.url=jdbc:oracle:thin:@localhost:1521/MYDB \
    --spring.datasource.username=<user> \
    --spring.datasource.password=<password> \
    --spring.datasource.driver-class-name=oracle.jdbc.OracleDriver
```

Due to licensing restrictions we're unable to bundle Oracle driver. You need to add it to server's classpath yourself.

Security

By default, the Spring Cloud Skipper server is unsecured and runs on an unencrypted HTTP connection. You can secure your REST endpoints by enabling HTTPS and requiring clients to authenticate using OAuth 2.0

By default, the REST endpoints (administration, management and health) do not require authenticated access.

Chapter 20. Enabling HTTPS

By default, the REST endpoints use plain HTTP as a transport. You can switch to HTTPS by adding a certificate to your configuration, as shown in the following `skipper.yml` example:

```
server:
  port: 8443                                    ①
  ssl:
    key-alias: yourKeyAlias                     ②
    key-store: path/to/keystore                 ③
    key-store-password: yourKeyStorePassword    ④
    key-password: yourKeyPassword               ⑤
    trust-store: path/to/trust-store            ⑥
    trust-store-password: yourTrustStorePassword ⑦
```

① As the default port is `7577`, you may choose to change the port to a more common HTTPs-typical port.

② The alias (or name) under which the key is stored in the keystore.

③ The path to the keystore file. Classpath resources may also be specified, by using the classpath prefix: `classpath:path/to/keystore`

④ The password of the keystore.

⑤ The password of the key.

⑥ The path to the truststore file. Classpath resources may also be specified, by using the classpath prefix: `classpath:path/to/trust-store`

⑦ The password of the trust store.

You can reference the YAML file using the following parameter: `--spring.config.additional-location=skipper.yml`

If HTTPS is enabled, it completely replaces HTTP as the protocol over which the REST endpoints interact. Plain HTTP requests then fail. Therefore, you must make sure that you configure the Skipper shell accordingly.

20.1. Using Self-Signed Certificates

For testing purposes or during development, it might be convenient to create self-signed certificates. To get started, run the following command to create a certificate:

```
$ keytool -genkey -alias skipper -keyalg RSA -keystore skipper.keystore \
          -validity 3650 -storetype JKS \
          -dname "CN=localhost, OU=Spring, O=Pivotal, L=Holualoa, ST=HI, C=US"  ①
          -keypass skipper -storepass skipper
```

① `CN` is the only important parameter here. It should match the domain you are trying to access,

e.g. `localhost`.

Then add the following to your `skipper.yml` file:

```
server:
  port: 8443
  ssl:
    enabled: true
    key-alias: skipper
    key-store: "/your/path/to/skipper.keystore"
    key-store-type: jks
    key-store-password: skipper
    key-password: skipper
```

That is all you need for the Skipper Server. Once you start the server, you should be able to access it at https://localhost:8443/. As this is a self-signed certificate, you should hit a warning in your browser. You need to ignore that.

20.2. Self-Signed Certificates and the Shell

By default, self-signed certificates are an issue for the shell. Additional steps are necessary to make the shell work with self-signed certificates. Two options are available:

- Add the self-signed certificate to the JVM truststore
- Skip certificate validation

20.2.1. Add the Self-signed Certificate to the JVM Truststore

In order to use the JVM truststore option, we need to export the previously created certificate from the keystore:

```
$ keytool -export -alias skipper -keystore skipper.keystore -file skipper_cert
-storepass skipper
```

Next, we need to create a truststore which the Shell uses:

```
$ keytool -importcert -keystore skipper.truststore -alias skipper -storepass skipper
-file skipper_cert -noprompt
```

Now you can launch the Skipper shell by using the following JVM arguments:

```
$ java -Djavax.net.ssl.trustStorePassword=skipper \
       -Djavax.net.ssl.trustStore=/path/to/skipper.truststore \
       -Djavax.net.ssl.trustStoreType=jks \
       -jar spring-cloud-skipper-shell-2.5.0-SNAPSHOT.jar
```

If you run into trouble establishing a connection over SSL, you can enable additional logging by setting the `javax.net.debug` JVM argument to `ssl`.

Remember to target the Skipper server with a config command similar to the following:

```
skipper:>skipper config --uri https://localhost:8443/api
```

20.2.2. Skip Certificate Validation

Alternatively, you can bypass the certification validation by providing the following optional command-line parameter: `--spring.cloud.skipper.client.skip-ssl-validation=true`.

When you set this command-line parameter, the shell accepts any (self-signed) SSL certificate.

If possible, you should avoid using this option. Disabling the trust manager defeats the purpose of SSL and makes your site vulnerable to man-in-the-middle attacks.

Chapter 21. OAuth 2.0 Security

OAuth 2.0 lets you integrate Spring Cloud Skipper into Single Sign-on (SSO) environments. You can use the following OAuth2 Grant Types:

- Password: Used by the shell (and the REST integration), so you can login with a username and a password
- Client Credentials: Retrieve an Access Token directly from your OAuth provider and pass it to the Skipper server in the `Authorization` HTTP header.

The REST endpoints can be accessed in two ways:

- Basic Authentication: Uses the *Password Grant Type* to authenticate with your OAuth2 service.
- Access Token: Uses the *Client Credentials Grant Type*

When you set up authentication, we strongly recommended enabling HTTPS as well, especially in production environments.

You can turn on OAuth2 authentication by setting environment variables or by adding the following block to `skipper.yml`:

```
security:
  oauth2:
    client:
      client-id: myclient                                                    ①
      client-secret: mysecret
      access-token-uri: http://127.0.0.1:9999/oauth/token
      user-authorization-uri: http://127.0.0.1:9999/oauth/authorize
    resource:
      user-info-uri: http://127.0.0.1:9999/me
```

```
spring:
  security:
    oauth2:                                                          ①
      client:
        registration:
          uaa:                                                       ②
            client-id: myclient
            client-secret: mysecret
            redirect-uri: '{baseUrl}/login/oauth2/code/{registrationId}'
            authorization-grant-type: authorization_code
            scope:
            - openid                                                 ③
        provider:
          uaa:
            jwk-set-uri: http://uaa.local:8080/uaa/token_keys
            token-uri: http://uaa.local:8080/uaa/oauth/token
            user-info-uri: http://uaa.local:8080/uaa/userinfo        ④
            user-name-attribute: user_name                           ⑤
            authorization-uri: http://uaa.local:8080/uaa/oauth/authorize
      resourceserver:
        opaquetoken:
          introspection-uri: http://uaa.local:8080/uaa/introspect ⑥
          client-id: dataflow
          client-secret: dataflow
  cloud:
    skipper:
      security:
        authorization:
          provider-role-mappings:                                    ⑦
            uaa:
              map-oauth-scopes: true
              role-mappings:
                ROLE_VIEW: skipper.view
                ROLE_CREATE: skipper.create
                ROLE_MANAGE: skipper.manage
```

① Providing this property activates OAuth2 security

② The provider id. It is possible to specify more than 1 provider

③ As the UAA is an OpenID provider, you must at least specify the `openid` scope. If your provider also provides additional scopes to control the role assignments, you must specify those scopes here as well

④ OpenID endpoint. Used to retrieve user information such as the username. Mandatory.

⑤ The JSON property of the response that contains the username

⑥ Used to introspect and validate a directly passed-in token. Mandatory.

⑦ Role mappings for authorization. You can verify that basic authentication is working properly by using `curl`, as follows:

```
`curl -u myusername:mypassword http://localhost:7577/`
```

As a result, you should see a list of available REST endpoints.

Besides Basic Authentication, you can also provide an Access Token to access the REST API. To make that happen, retrieve an OAuth2 Access Token from your OAuth2 provider and then pass that Access Token to the REST API by using the `Authorization` HTTP header, as follows:

```
curl -H "Authorization: Bearer <ACCESS_TOKEN>" http://localhost:7577/
```

21.1. OAuth REST Endpoint Authorization

Spring Cloud Skipper supports the following roles:

- **VIEW**: For anything that relates to retrieving state.
- **CREATE**: For anything that involves creating, deleting, or mutating the state of the system.
- **MANAGE**: For boot management endpoints.

The rules regarding which REST endpoints require which roles are specified in the `application.yml` of the `spring-cloud-skipper-server-core` module.

Nonetheless, you can override those, if desired. The configuration takes the form of a YAML **list** (as some rules may have precedence over others). Consequently, you need to copy/paste the whole list and tailor it to your needs (as there is no way to merge lists). Always refer to your version of `application.yml`, as the snippet reproduced below may be outdated. The default rules are as follows:

```
            # About
            - GET /api/about                        => hasRole('ROLE_VIEW')
            # AppDeployerDatas
            - GET /api/appDeployerDatas             => hasRole('ROLE_VIEW')
            # Deployers
            - GET /api/deployers                    => hasRole('ROLE_VIEW')
            ## Releases
            - GET /api/releases                     => hasRole('ROLE_VIEW')
            # Status
            - GET /api/release/status/**            => hasRole('ROLE_VIEW')
```

```
        # Manifest

        - GET /api/release/manifest/**       => hasRole('ROLE_VIEW')

        # Upgrade

        - POST /api/release/upgrade          => hasRole('ROLE_CREATE')

        # Rollback

        - POST /api/release/rollback/**      => hasRole('ROLE_CREATE')

        # Delete

        - DELETE /api/release/**             => hasRole('ROLE_CREATE')

        # History

        - GET /api/release/history/**        => hasRole('ROLE_VIEW')

        # List

        - GET /api/release/list              => hasRole('ROLE_VIEW')
        - GET /api/release/list/**           => hasRole('ROLE_VIEW')

        # Packages

        - GET /api/packages                  => hasRole('ROLE_VIEW')

        # Upload

        - POST /api/package/upload           => hasRole('ROLE_CREATE')

        # Install

        - POST /api/package/install          => hasRole('ROLE_CREATE')
        - POST /api/package/install/**       => hasRole('ROLE_CREATE')

        # Delete

        - DELETE /api/package/**             => hasRole('ROLE_CREATE')

        # PackageMetaData

        - GET /api/packageMetadata           => hasRole('ROLE_VIEW')
        - GET /api/packageMetadata/**        => hasRole('ROLE_VIEW')

        # Repositories

        - GET /api/repositories              => hasRole('ROLE_VIEW')
        - GET /api/repositories/**           => hasRole('ROLE_VIEW')
```

```
            # Boot Endpoints

            - GET  /actuator/**                 => hasRole('ROLE_MANAGE')
```

The format of each line is as follows:

`HTTP_METHOD URL_PATTERN '⇒' SECURITY_ATTRIBUTE`

where

- HTTP_METHOD is one http method, capital case.
- URL_PATTERN is an Ant-style URL pattern.
- SECURITY_ATTRIBUTE is a SpEL expression (see docs.spring.io/spring-security/site/docs/current/reference/htmlsingle/#el-access)
- Each of those parts is separated by one or several white space characters (spaces, tabs, and others).

Be mindful that the above is indeed a YAML list, not a map (thus the use of '-' dashes at the start of each line) that lives under the `spring.cloud.skipper.security.authorization.rules` key.

21.1.1. Users and Roles

Spring Cloud Skipper does not make any assumptions of how roles are assigned to users. Due to the fact that the determination of security roles is very environment-specific, Spring Cloud Data Skipper, by default, assigns *all roles* to authenticated OAuth2 users by using the `DefaultAuthoritiesExtractor` class.

You can customize that behavior by providing your own Spring bean definition that extends Spring Security OAuth's `AuthoritiesExtractor` interface. In that case, the custom bean definition takes precedence over the default one provided by Spring Cloud Skipper.

21.2. OAuth Authentication Using the Spring Cloud Skipper Shell

If your OAuth2 provider supports the Password Grant Type, you can start the Skipper shell with the following command:

```
$ java -jar spring-cloud-skipper-shell-2.5.0-SNAPSHOT.jar \
  --spring.cloud.skipper.client.serverUrl=http://localhost:7577 \
  --spring.cloud.skipper.client.username=my_username \
  --spring.cloud.skipper.client.password=my_password
```

When authentication for Spring Cloud Skipper is enabled, the underlying OAuth2 provider **must** support the Password OAuth2 Grant Type if you want to use the hell.

From within the Skipper shell, you can also provide credentials by using the following command:

```
skipper:> skipper config --uri https://localhost:7577/api --username my_username
--password my_password
```

Once successfully targeted, you should see the following output:

```
Successfully targeted http://localhost:7577/api
skipper:>
```

21.3. OAuth2 Authentication Examples

This section provides examples of some common security arrangements for Skipper:

- Local OAuth2 Server
- Authentication Using UAA
- [skipper-security-authentication-using-github]

21.3.1. Local OAuth2 Server

With Spring Security OAuth, you can create your own OAuth2 Server by using the following annotations:

- `@EnableResourceServer`
- `@EnableAuthorizationServer`

You can find a working example application at https://github.com/ghillert/oauth-test-server/.

To do so, clone the project, build it, and start it. Then configure Spring Cloud Skipper with the respective Client ID and Client Secret.

Use this option only for development or demo purposes.

21.3.2. Authentication Using UAA

If you need to set up a production-ready OAuth provider, you may want to consider using the CloudFoundry User Account and Authentication (UAA) Server. While it is used by Cloud Foundry, it can also be used stand-alone. For more information see github.com/cloudfoundry/uaa.

Skipper Commands

Skipper commands fit into the following categories:

- Package Commands
- Release Commands
- Manifest Commands
- Platform commands
- Repository Commands
- Skipper Server Commands

More details about commands can be found from Generic Usage.

Chapter 22. Package Commands

Skipper's package commands include the following:

- Search
- Upload
- Install
- Delete

22.1. Search

This command searches existing packages.

NAME

package search - Search for packages.

SYNOPSYS

package search [[**--name**] string] [**--details**]

OPTIONS

--name string

wildcard expression to search for the package name
[Optional, default = **<none>]**

--details boolean

to set for more detailed package metadata
[Optional, default = **false]**

ALSO KNOWN AS

package list

The `search` or its alias `list` command shows all the packages available to be installed by the Skipper server, as shown (with output) in the following example:

```
skipper:>package search
╔═════════════════╤═════════╤═════════════════════════════════════════════════
═══════════════════════════════════════════════════════════════════════════════
══════════════════════════════════════════╗
║      Name       │ Version │                                   Description
║
╠═════════════════╪═════════╪═════════════════════════════════════════════════
═══════════════════════════════════════════════════════════════════════════════
══════════════════════════════════════════╣
║helloworld       │1.0.0    │The app has two endpoints, /about and /greeting in
English.  Maven resource.    ║
║helloworld       │1.0.1    │The app has two endpoints, /about and /greeting in
Portuguese.  Maven resource. ║
║helloworld-docker│1.0.1    │The app has two endpoints, /about and /greeting in
Portuguese.  Docker resource.║
║helloworld-docker│1.0.0    │The app has two endpoints, /about and /greeting in
English.  Docker resource.   ║
╚═════════════════╧═════════╧═════════════════════════════════════════════════
═══════════════════════════════════════════════════════════════════════════════
══════════════════════════════════════════╝
```

```
skipper:>package list
╔═════════════════╤═════════╤═════════════════════════════════════════════════
═══════════════════════════════════════════════════════════════════════════════
══════════════════════════════════════════╗
║      Name       │ Version │                                   Description
║
╠═════════════════╪═════════╪═════════════════════════════════════════════════
═══════════════════════════════════════════════════════════════════════════════
══════════════════════════════════════════╣
║helloworld       │1.0.0    │The app has two endpoints, /about and /greeting in
English.  Maven resource.    ║
║helloworld       │1.0.1    │The app has two endpoints, /about and /greeting in
Portuguese.  Maven resource. ║
║helloworld-docker│1.0.1    │The app has two endpoints, /about and /greeting in
Portuguese.  Docker resource.║
║helloworld-docker│1.0.0    │The app has two endpoints, /about and /greeting in
English.  Docker resource.   ║
╚═════════════════╧═════════╧═════════════════════════════════════════════════
═══════════════════════════════════════════════════════════════════════════════
══════════════════════════════════════════╝
```

The `search` command can use `--name` option to search for the package name containing the given option value, as shown (with output) in the following example:

```
skipper:>package search --name helloworld-
╔═════════════════════════════════╤═══════════════╤══════════════════════════════
═════════════════════════════════════════════════════════════════════════════════
═══════════════════════════════════════╗
║      Name       │ Version │                          Description
║
╠═════════════════════════════════╪═══════════════╪══════════════════════════════
═════════════════════════════════════════════════════════════════════════════════
═══════════════════════════════════════╣
║helloworld-docker│1.0.0   │The app has two endpoints, /about and /greeting in
English.  Docker resource.    ║
║helloworld-docker│1.0.1   │The app has two endpoints, /about and /greeting in
Portuguese.  Docker resource.║
╚═════════════════════════════════╧═══════════════╧══════════════════════════════
═════════════════════════════════════════════════════════════════════════════════
═══════════════════════════════════════╝
```

To search for more details of the packages, the `--details` option can be used, as shown (with output) in the following example:

```
skipper:>package search --name helloworld- --details
╔═══════════════════════════════╤═══════════════════════════════════════════════
═════════════════════════════════════════════════════════════════════════════════
═════════════════╗
║      Name       │                                Value
║
╠═══════════════════════════════╪═══════════════════════════════════════════════
═════════════════════════════════════════════════════════════════════════════════
═════════════════╣
║apiVersion       │v1
║
║origin           │A sample repository for using Skipper
║
║repositoryId     │1
║
║kind             │skipper
║
║name             │helloworld-docker
║
║version          │1.0.0
║
║packageSourceUrl │https://github.com/markpollack/skipper-sample-repository
║
║packageHomeUrl   │https://github.com/markpollack/skipper-sample-repository
║
║tags             │web, demo, docker, helloworld
║
║maintainer       │https://github.com/markpollack
║
```

```
 ‖ description      | The app has two endpoints, /about and /greeting in English.
Docker resource. ‖
 ‖ sha256           |
 ‖
 ‖ iconUrl          |
 ‖
 ╰──────────────────────────────────┴──────────────────────────────────────────────
──────────────────────────────────────────────────────────────────────────────────
──────────────────╯


 ╭──────────────────────────────────┬──────────────────────────────────────────────
──────────────────────────────────────────────────────────────────────────────────
──────────────────╮
 ‖       Name       |                                      Value
 ‖
 |╰──────────────────────────────────┼──────────────────────────────────────────────
──────────────────────────────────────────────────────────────────────────────────
──────────────────╯|
 ‖ apiVersion       | v1
 ‖
 ‖ origin           | A sample repository for using Skipper
 ‖
 ‖ repositoryId     | 1
 ‖
 ‖ kind             | skipper
 ‖
 ‖ name             | helloworld-docker
 ‖
 ‖ version          | 1.0.1
 ‖
 ‖ packageSourceUrl | https://github.com/markpollack/skipper-sample-repository
 ‖
 ‖ packageHomeUrl   | https://github.com/markpollack/skipper-sample-repository
 ‖
 ‖ tags             | web, demo, docker, helloworld
 ‖
 ‖ maintainer       | https://github.com/markpollack
 ‖
 ‖ description      | The app has two endpoints, /about and /greeting in Portuguese.
Docker resource. ‖
 ‖ sha256           |
 ‖
 ‖ iconUrl          |
 ‖
 ╰──────────────────────────────────┴──────────────────────────────────────────────
──────────────────────────────────────────────────────────────────────────────────
──────────────────╯
```

22.2. Upload

This command uploads a package .zip file, as shown (with output) in the following example:

NAME

package upload - Upload a package.

SYNOPSYS

package upload [**--path**] string [[**--repo-name**] string]

OPTIONS

--path string

the package to be uploaded
[Mandatory]

--repo-name string

the local repository name to upload to
[Optional, default = <none>]

```
skipper:>package upload --path /path-to-package/mypackage-1.0.0.zip
Package uploaded successfully:[mypackage:1.0.0]
```

If no `--repo-name` is set, the `upload` command uses `local` as the repository to upload.

22.3. Install

This command installs a package, as shown (with output) in the following example:

NAME

package install - Install a package.

SYNOPSYS

package install [**--package-name**] string [[**--package-version**] string] [[**--file**] file] [[**--properties**] string] [**--release-name**] string [[**--platform-name**] string]

OPTIONS

--package-name string

name of the package to install
[Mandatory]

--package-version string

version of the package to install, if not specified latest version will be used
[Optional, default = <none>]

--file file

specify values in a YAML file
[Optional, default = <none>]

--properties string

the comma separated set of properties to override during install
[Optional, default = <none>]

--release-name string

the release name to use
[Mandatory]

--platform-name string

the platform name to use
[Optional, default = default]

```
skipper:>package install --release-name helloworldlocal --package-name helloworld
--package-version 1.0.0 --properties spec.applicationProperties.server.port=8099
Released helloworldlocal. Now at version v1.
```

If no `package-version` is specified, then the latest package version by the given `package-name` is considered.

If no `platform-name` is specified, the platform name, `default`, is used.

The properties can either be provided through comma separated YAML string by using the `--properties` option or through a YAML file by using the `--file` option.

22.4. Delete

This command deletes a package.

NAME

package delete - Delete a package.

SYNOPSYS

package delete [**--package-name**] string

OPTIONS

--package-name string

the package name to be deleted
[Mandatory]

You can only delete a package that is in a local (database backed) repository, as shown (with output) in the following example:

```
skipper:>package delete --package-name helloworld
Can not delete package [helloworld], associated repository [experimental] is remote.
```

Chapter 23. Release Commands

Skipper's release commands include the following:

23.1. List

This command lists the latest deployed or failed release.

NAME

release list - List the latest version of releases with status of deployed or failed.

SYNOPSYS

release list [[**--release-name**] string]

OPTIONS

--release-name string

wildcard expression to search by release name
[Optional, default = **<none>]**

Listing the latest deployed or failed release, as shown (with output) in the following example:

```
skipper:>release list
╔╤══════════════════════════════╤═══════════════╤═════════════════════════════════
═══════════════╤════════════════╤══════════════════════╤══════════════════════════
═╤════════════════════════╤═══════════════════════════════════════════════════════
══════════════════════════════════════════════════════════════════════════════════
════════════╗
║│     Name      │Version│      Last updated       │ Status │  Package  │
Package    │  Platform  │                              Platform Status
║
║│               │       │                         │        │   Name    │
Version    │    Name    │
║
╟┼──────────────────────────────┼───────────────┼─────────────────────────────────
───────────────┼────────────────┼──────────────────────┼──────────────────────────
─┼────────────────────────┼───────────────────────────────────────────────────────
──────────────────────────────────────────────────────────────────────────────────
────────────╢
║│helloworldlocal│3      │Mon Oct 30 17:57:41 IST  │DEPLOYED│helloworld │1.0.0
│default     │[helloworldlocal.helloworld-v3], State =
║
║│               │       │2017                     │        │           │
│            │[helloworldlocal.helloworld-v3-0=deployed]
║
╚╧══════════════════════════════╧═══════════════╧═════════════════════════════════
═══════════════╧════════════════╧══════════════════════╧══════════════════════════
═╧════════════════════════╧═══════════════════════════════════════════════════════
══════════════════════════════════════════════════════════════════════════════════
════════════╝
```

23.2. Status

This command shows a release status.

NAME

release status - Status for a last known release version.

SYNOPSYS

release status [**--release-name**] string [[**--release-version**] integer]

OPTIONS

--release-name string

release name
[Mandatory]
[may not be null]

--release-version integer

the specific release version.
[Optional, default = <none>]

Shows the `status` of a specific release and version, as shown (with output) in the following example:

```
skipper:>release status --release-name helloworldlocal
╔═══════════════════════════════╤═══════════════════════════════════════════════════════════════════════════════
═══════════════════════════════════════════════════════════════════════════════
═══════════════════════════════╗
║Last Deployed   │Mon Oct 30 17:53:50 IST 2017
║
║Status          │DEPLOYED
║
║Platform Status │All applications have been successfully deployed.
║
║                │[helloworldlocal.helloworld-v2], State =
[helloworldlocal.helloworld-v2-0=deployed]║
╚═══════════════════════════════╧═══════════════════════════════════════════════════════════════════════════════
═══════════════════════════════════════════════════════════════════════════════
═══════════════════════════════╝
```

If no `--release-version` specified, the latest release version is used. The following example shows the command with the `--release-version` option:

```
skipper:>release status --release-name helloworldlocal --release-version 1
╔═══════════════════════════════════╦═══════════════════════════════════════════════════════════════════════════════
═══════════════════════════════════════════════════════════════════════════════
═══════╗
║ Last Deployed   ║ Mon Oct 30 17:52:57 IST 2017
║
║ Status          ║ DELETED
║
║ Platform Status ║ The applications are known to the system, but is not currently
deployed. ║
║                 ║ [helloworldlocal.helloworld-v1], State = [unknown]
║
╚═══════════════════════════════════╩═══════════════════════════════════════════════════════════════════════════════
═══════════════════════════════════════════════════════════════════════════════
═══════╝
```

23.3. Upgrade

This command upgrades a package.

NAME

release upgrade - Upgrade a release.

SYNOPSYS

release upgrade [**--release-name**] string [**--package-name**] string [[**--package-version**] string] [[**--file**] file] [[**--properties**] string] [[**--timeout-expression**] string]

OPTIONS

--release-name string

The name of the release to upgrade
[Mandatory]

--package-name string

the name of the package to use for the upgrade
[Mandatory]

--package-version string

the version of the package to use for the upgrade, if not specified latest version will be used
[Optional, default = <none>]

--file file

specify values in a YAML file
[Optional, default = <none>]

--properties string

the comma separated set of properties to override during upgrade
[Optional, default = <none>]

--timeout-expression string

the expression for upgrade timeout
[Optional, default = <none>]

```
--force force upgrade
    [Optional, default = false]
```

```
--app-names  string
    application names to force upgrade. If no specific list is provided, all
the apps in the packages are force upgraded
    [Optional, default = <none>]
```

Upgrades a package, as shown (with output) in the following example:

```
skipper:>release upgrade --release-name helloworldlocal --package-name helloworld
--package-version 1.0.0 --properties spec.applicationProperties.server.port=9090
helloworldpcf has been upgraded.  Now at version v2.
```

The manifest for this release would look like this:

```
"apiVersion": "skipper.spring.io/v1"
"kind": "SpringCloudDeployerApplication"
"metadata":
  "name": "helloworld"
  "type": "demo"
"spec":
  "resource": "maven://org.springframework.cloud.samples:spring-cloud-skipper-samples-
helloworld"
  "version": "1.0.0.RELEASE"
  "applicationProperties":
    "server.port": "9090"
  "deploymentProperties": !!null "null"
```

If no `package-version` is specified, the latest package version by the given `--package-name` option is considered. The properties can either be provided through comma separated YAML string by using the `--properties` option or through a YAML file by using the `--file` option.

An upgrade can be done by overriding the package version or by keeping the existing package version but overriding the properties. When overriding the package version, it needs to accompany with the corresponding properties as the existing properties are not carried over. In a future release, we plan to introduce a `--reuse-properties` command that will carry the current release properties over to the next release to be made.

For instance, if the package version is not changed but only other properties are changed, the manifest would add the new properties with the existing properties of the same package version.

```
skipper:>release upgrade --release-name helloworldlocal --package-name helloworld
--package-version 1.0.0 --properties spec.applicationProperties.log.level=DEBUG
helloworldpcf has been upgraded.  Now at version v3.
```

```
"apiVersion": "skipper.spring.io/v1"
"kind": "SpringCloudDeployerApplication"
"metadata":
  "name": "helloworld"
  "type": "demo"
"spec":
  "resource": "maven://org.springframework.cloud.samples:spring-cloud-skipper-samples-
helloworld"
  "version": "1.0.0.RELEASE"
  "applicationProperties":
    "server.port": "9090"
    "log.level": "DEBUG"
  "deploymentProperties": !!null "null"
```

Instead, if the upgrade is performed with a new package version as follows,

```
skipper:>release upgrade --release-name helloworldlocal --package-name helloworld
--package-version 1.0.1
helloworldpcf has been upgraded.  Now at version v3.
```

Since the package version is changed, the manifest wouldn't carry the properties from the existing release.

```
skipper:>manifest get helloworldlocal
"apiVersion": "skipper.spring.io/v1"
"kind": "SpringCloudDeployerApplication"
"metadata":
  "name": "helloworld"
  "type": "demo"
"spec":
  "resource": "maven://org.springframework.cloud.samples:spring-cloud-skipper-samples-
helloworld"
  "version": "1.0.1.RELEASE"
  "applicationProperties": !!null "null"
  "deploymentProperties": !!null "null"
```

When performing an update on a package that contains nested packages, use the name of the package as a prefix in the property string or as the first level in the YAML document. For example, the `ticktock` package that contains a `time` and a `log` application, a command to upgrade the `log` application would be as follows:

```
skipper:>release upgrade --release-name ticktockskipper --package-name ticktock --file
/home/mpollack/log-level-change.yml
```

where `log-level-change.yml` contains the following:

```
log:
  version: 1.1.1.RELEASE
  spec:
    applicationProperties:
      server.port: 9999
      endpoints.sensitive: false
      log.level: ERROR
```

Since it is a common use-case to change only the version of the application, the packages can list the version as a top-level property in the `values.yml` file. For example, in the test package `ticktock` (located here), `values.yml` contains the following:

```
version: 1.1.0.RELEASE
spec:
  applicationProperties:
    log.level: DEBUG
  deploymentProperties:
    memory: 1024m
```

You can then use the `--properties` option in the `upgrade` command, as shown in the following example:

```
skipper:>release upgrade --release-name ticktockskipper --package-name ticktock
--properties log.version=1.1.1.RELEASE
```

You can use `--timeout-expression` to alter *timeout* setting used to wait healthy applications when server is in state to do that. Global setting to override is `spring.cloud.skipper.server.strategies.healthcheck.timeoutInMillis` mentioned earlier. More about expression itself, see Timeout Expression.

```
skipper:>release upgrade --release-name ticktockskipper --package-name ticktock
--timeout-expression=30s
```

The `--force` option is used to deploy new instances of currently deployed applications. In other words, Skipper will upgrade the application again even if the manifest is unchanged. This behavior is needed in the case when configuration information is obtained by the application itself at startup time, for example from Spring Cloud Config Server. You can specify which applications for force upgrade by using the option `--app-names`. If you do not specify any application names, all the applications will be force upgraded. You can specify `--force` and `--app-names` options together with `--properties` or `--file` options.

Following example describes `force` upgrade:

First, install the package `ticktock` that has `time` and `log` apps.

```
skipper:>package upload --repo-name local --path spring-cloud-skipper-server-
core/src/test/resources/repositories/binaries/test/ticktock/ticktock-1.0.0.zip
Package uploaded successfully:[ticktock:1.0.0]

skipper:>package install --package-name ticktock --release-name a1
Released a1. Now at version v1.

skipper:>release list
╔═════╤════════╤════════════════════════════════════════════════════════════════════
════════════════╤═════════════════════════╤═══════════════════════════════
════════════════╤═══════════════════════════════════════════════════════════════
═══════════════════════╗
║Name│Version│          Last updated          │ Status  │Package Name│Package
Version│Platform Name│                Platform Status                ║
╠═════╪════════╪════════════════════════════════════════════════════════════════════
════════════════╪═════════════════════════╪═══════════════════════════════
════════════════╪═══════════════════════════════════════════════════════════════
═══════════════════════╣
║a1   │1       │Thu Sep 13 08:34:50 IST 2018│DEPLOYED│ticktock     │1.0.0
│default       │[a1.log-v1], State = [a1.log-v1-0=deployed]   ║
║     │        │                            │        │             │
│              │[a1.time-v1], State = [a1.time-v1-0=deployed]║
╚═════╧════════╧════════════════════════════════════════════════════════════════════
════════════════╧═════════════════════════╧═══════════════════════════════
════════════════╧═══════════════════════════════════════════════════════════════
═══════════════════════╝

skipper:>release history --release-name a1
╔════════════════╤═════════════════════════════════════════════════════════╤════════
═══════╤═════════════════════════╤═══════════════════════════════╤════════
═════════════╗
║Version│          Last updated          │ Status  │Package Name│Package Version│
Description    ║
╠════════════════╪═════════════════════════════════════════════════════════╪════════
═══════╪═════════════════════════╪═══════════════════════════════╪════════
═════════════╣
║1      │Thu Sep 13 08:34:50 IST 2018│DEPLOYED│ticktock     │1.0.0
│Install complete║
╚════════════════╧═════════════════════════════════════════════════════════╧════════
═══════╧═════════════════════════╧═══════════════════════════════╧════════
═════════════╝
```

Now, perform the upgrade even though there are no explicit differences between the `latest` and the `current` package.

```
skipper:>release upgrade --release-name a1 --package-name ticktock
Package to upgrade has no difference than existing deployed/deleted package. Not
upgrading.
```

If the upgrade needs to be forced for all the apps of `ticktock` (for both `time` and `log`)

```
skipper:>release upgrade --release-name a1 --package-name ticktock --force
a1 has been upgraded.  Now at version v2.
skipper:>release history --release-name a1
╔═══════╤════════════════════════════╤════════╤════════════╤═══════════════
═══╤════════════════════════╗
║Version│        Last updated        │ Status │Package Name│Package Version│
Description      ║
╠═══════╪════════════════════════════╪════════╪════════════╪═══════════════
═══╪════════════════════════╣
║2      │Thu Sep 13 08:35:53 IST 2018│UNKNOWN │ticktock    │1.0.0
│Upgrade install underway║
║1      │Thu Sep 13 08:34:50 IST 2018│DEPLOYED│ticktock    │1.0.0
│Install complete        ║
╚═══════╧════════════════════════════╧════════╧════════════╧═══════════════
═══╧════════════════════════╝

skipper:>release history --release-name a1
╔═══════╤════════════════════════════╤════════╤════════════╤═══════════════
═══╤════════════════╗
║Version│        Last updated        │ Status │Package Name│Package Version│
Description    ║
╠═══════╪════════════════════════════╪════════╪════════════╪═══════════════
═══╪════════════════╣
║2      │Thu Sep 13 08:35:53 IST 2018│DEPLOYED│ticktock    │1.0.0
│Upgrade complete║
║1      │Thu Sep 13 08:34:50 IST 2018│DELETED │ticktock    │1.0.0
│Delete complete ║
╚═══════╧════════════════════════════╧════════╧════════════╧═══════════════
═══╧════════════════╝
```

If the `force` upgrade needs to be done for a specific list of applications, then `--app-names` option can be used.

```
skipper:>release upgrade --release-name a1 --package-name ticktock --force --app-names
log
a1 has been upgraded.  Now at version v3.
skipper:>release history a1
╔═══════╤════════════════════════════╤═════════╤════════════╤═══════════════╤
════════════════╗
║Version│        Last updated        │ Status  │Package Name│Package Version│
Description   ║
╠═══════╪════════════════════════════╪═════════╪════════════╪═══════════════╪
════════════════╣
║3      │Thu Sep 13 08:36:51 IST 2018│DEPLOYED│ticktock    │1.0.0
│Upgrade complete║
║2      │Thu Sep 13 08:35:53 IST 2018│DELETED │ticktock    │1.0.0
│Delete complete ║
║1      │Thu Sep 13 08:34:50 IST 2018│DELETED │ticktock    │1.0.0
│Delete complete ║
╚═══════╧════════════════════════════╧═════════╧════════════╧═══════════════╧
════════════════╝
```

23.4. Rollback

This command rolls back the release.

NAME

release rollback - Rollback the release to a previous or a specific release.

SYNOPSYS

release rollback [**--release-name**] string [[**--release-version**] int] [[**--timeout-expression**] string]

OPTIONS

--release-name string

the name of the release to rollback
[Mandatory]

--release-version int

the specific release version to rollback to. Not specifying the value rolls back to the previous release.
[Optional, default = 0]

--timeout-expression string

the expression for rollback timeout
[Optional, default = <none>]

Rolls back the release to a specific version, as shown (with output) in the following example:

```
skipper:>release rollback --release-name helloworldlocal --release-version 1
helloworldlocal has been rolled back.  Now at version v3.
```

If no `--release-version` is specified, then the rollback version is the previous stable release (either in `DELETED` or `DEPLOYED` status).

You can use `--timeout-expression` to alter *timeout* setting used to wait healthy applications when server is in state to do that. Global setting to override is `spring.cloud.skipper.server.strategies.healthcheck.timeoutInMillis` mentioned earlier. More about expression itself, see Timeout Expression.

23.5. History

This command shows the history of a specific release.

NAME

release history - List the history of versions for a given release.

SYNOPSYS

release history [**--release-name**] string

OPTIONS

--release-name string

wildcard expression to search by release name
[Mandatory]
*[may not be null]

Showing the history of a specific release, as shown (with output) in the following example:

```
skipper:>release history --release-name helloworldlocal
╔═══════╤════════════════════════════╤════════╤════════════╤═══════════════╤═════════════════╗
║Version│        Last updated        │ Status │Package Name│Package Version│
Description   ║
╠═══════╪════════════════════════════╪════════╪════════════╪═══════════════╪═════════════════╣
║3      │Mon Oct 30 17:57:41 IST 2017│DEPLOYED│helloworld  │1.0.0
│Upgrade complete║
║2      │Mon Oct 30 17:53:50 IST 2017│DELETED │helloworld  │1.0.0
│Delete complete ║
║1      │Mon Oct 30 17:52:57 IST 2017│DELETED │helloworld  │1.0.0
│Delete complete ║
╚═══════╧════════════════════════════╧════════╧════════════╧═══════════════╧═════════════════╝
```

23.6. Delete

This command deletes a specific release's latest deployed revision.

NAME

release delete - Delete the release.

SYNOPSYS

release delete [**--release-name**] string [**--delete-package**]

OPTIONS

--release-name string

the name of the release to delete
[Mandatory]

--delete-package delete the release package

[Optional, default = **false]**

Deleting a specific release's latest deployed revision, undeploying the application or applications, as shown (with output) in the following example:

```
skipper:>release delete --release-name helloworldlocal
helloworldlocal has been deleted.
```

23.7. Cancel

This command attempts cancellation of existing release operation.

NAME

release cancel - Request a cancellation of current release operation.

SYNOPSYS

release cancel [**--release-name**] string

OPTIONS

--release-name string

the name of the release to cancel
[Mandatory]

This command can be used to attempt a cancel for a running release operation if it supports it and release is currently in state where any type of cancellation can be attempted. For example during an upgrade server will delete old applications if new applications are detected healtly. Before state is transitioned to deleting old applications, it is possible to request cancellation of whole upgrade procedure.

One other use case is that if new applications are failed and server will timeout waiting healtly applications, it's convenient to cancel operation without waiting full timeout to happen.

Here is an example how cancellation is attempted when upgraded applications fail:

```
skipper:>package install --package-name testapp --package-version 1.0.0 --release-name
mytestapp
Released mytestapp. Now at version v1.

skipper:>release history --release-name mytestapp
╔═══════╤══════════════════════════════╤═══════════╤══════════════╤═══════════════╤═══════════════╗
║Version│         Last updated         │  Status   │Package Name  │Package Version│
Description    ║
╠═══════╪══════════════════════════════╪═══════════╪══════════════╪═══════════════╪═══════════════╣
║1      │Thu May 17 11:18:07 BST 2018 │DEPLOYED   │testapp       │1.0.0          │Install complete
 ║
╚═══════╧══════════════════════════════╧═══════════╧══════════════╧═══════════════╧═══════════════╝


skipper:>release upgrade --package-name testapp --package-version 1.1.0 --release-name
mytestapp
mytestapp has been upgraded.  Now at version v2.
```

```
skipper:>release history --release-name mytestapp
╔═══════╤══════════════════════════╤═══════════╤══════════════╤═══════════════╤═══════════════════╗
║Version│       Last updated       │  Status   │ Package Name │ Package Version │ Description       ║
╠═══════╪══════════════════════════╪═══════════╪══════════════╪═══════════════╪═══════════════════╣
║2      │Thu May 17 11:18:52 BST 2018│UNKNOWN    │testapp       │1.1.0          │Upgrade install underway ║
║1      │Thu May 17 11:18:07 BST 2018│DEPLOYED   │testapp       │1.0.0          │Install complete   ║
╚═══════╧══════════════════════════╧═══════════╧══════════════╧═══════════════╧═══════════════════╝

skipper:>release status --release-name mytestapp
╔═══════════════╤═══════════════════════════════════════════════════════════════════════════════╗
║Last Deployed  │Thu May 17 11:18:52 BST 2018                                                   ║
║Status         │UNKNOWN                                                                        ║
║Platform Status│All apps have failed deployment.                                               ║
║               │[mytestapp.testapp-v2], State = [mytestapp.testapp-v2-0=failed]                ║
╚═══════════════╧═══════════════════════════════════════════════════════════════════════════════╝

skipper:>release cancel --release-name mytestapp
Cancel request for release mytestapp sent

skipper:>release history --release-name mytestapp
╔═══════╤══════════════════════════╤═══════════╤══════════════╤═══════════════╤═══════════════════╗
║Version│       Last updated       │  Status   │ Package Name │ Package Version │ Description       ║
╠═══════╪══════════════════════════╪═══════════╪══════════════╪═══════════════╪═══════════════════╣
║2      │Thu May 17 11:18:52 BST 2018│FAILED     │testapp       │1.1.0          │Cancelled after 39563 ms. ║
║1      │Thu May 17 11:18:07 BST 2018│DEPLOYED   │testapp       │1.0.0          │Install complete   ║
╚═══════╧══════════════════════════╧═══════════╧══════════════╧═══════════════╧═══════════════════╝
```

Chapter 24. Manifest Commands

Skipper's manifest has only one command: `get`.

24.1. Get

Thsi command shows a manifest.

NAME

manifest get - Get the manifest for a release

SYNOPSYS

manifest get [**--release-name**] string [[**--release-version**] integer]

OPTIONS

--release-name string

release name
[Mandatory]
[may not be null]

--release-version integer

specific release version.
[Optional, default = <none>]

The `manifest get` command shows the manifest used for a specific release, as shown (with output) in the following example:

```
skipper:>manifest get --release-name helloworldk8s

---
# Source: template.yml
apiVersion: skipper.spring.io/v1
kind: SpringCloudDeployerApplication
metadata:
  name: helloworld-docker
spec:
  resource: docker:springcloud/spring-cloud-skipper-samples-helloworld:1.0.0.RELEASE
  applicationProperties:
  deploymentProperties:
    spring.cloud.deployer.kubernetes.createNodePort: 32123
```

Chapter 25. Platform commands

Skipper's platform has only one command: `list`.

25.1. List

This command lists platforms.

NAME

platform list - List platforms

SYNOPSYS

platform list

The `platform list` command shows the list all the available deployment platform accounts, as shown (with output) in the following example:

```
skipper:>platform list
╔═══════════════╦═══════════════╦═══════════════════════════════════════════════════════════════
═══════════════════════════════════════════════════════════════════════════════════════════════
═══════════════════════════════════════════════════════════════════════════════════════════════
═══════════════════════════════════════════════════════════════════════════════════════════════
═══════════════════════════════════════════════════════════════════════════════════════════════
═══════════════════════╗
║   Name    ║     Type      ║
Description
                       ║
╠═══════════════╬═══════════════╬═══════════════════════════════════════════════════════════════
═══════════════════════════════════════════════════════════════════════════════════════════════
═══════════════════════════════════════════════════════════════════════════════════════════════
═══════════════════════════════════════════════════════════════════════════════════════════════
═══════════════════════════════════════════════════════════════════════════════════════════════
═══════════════════════╣
╠═══════════════╬═══════════════╬═══════════════════════════════════════════════════════════════
═══════════════════════╗
║default  ║local        ║ShutdownTimeout = [30], EnvVarsToInherit =
[TMP,LANG,LANGUAGE,LC_.*,PATH], JavaCmd =
                       ║
║         ║
 ║[/Library/Java/JavaVirtualMachines/jdk1.8.0_131.jdk/Contents/Home/jre/bin/java],
WorkingDirectoriesRoot = [/var/folders/t3/qf1wkpwj4lgd9gjccwk0wr7h0000gp/T], ║
║         ║               ║DeleteFilesOnExit = [true]
                       ║
║cf-dev   ║cloudfoundry ║org = [scdf-ci], space = [ilaya-space], url =
[https://api.run.pivotal.io]
                       ║
║minikube ║kubernetes   ║master url = [https://192.168.99.101:8443/], namespace =
[default], api version = [v1]
                       ║
╚═══════════════╩═══════════════╩═══════════════════════════════════════════════════════════════
═══════════════════════════════════════════════════════════════════════════════════════════════
═══════════════════════════════════════════════════════════════════════════════════════════════
═══════════════════════════════════════════════════════════════════════════════════════════════
═══════════════════════════════════════════════════════════════════════════════════════════════
═══════════════════════╝
```

Chapter 26. Repository Commands

Skipper’s repository commands include the following:

- List

26.1. List

This command list repositories.

NAME

repo list - List package repositories

SYNOPSYS

repo list

List repositories as shown (with output) in the following example:

```
skipper:>repo list
╔═════════════╤════════════════════════════════════════════════════════════
════════════╤═══════╤═════╗
║    Name     │                           URL
│Local│Order║
╠═════════════╪════════════════════════════════════════════════════════════
════════════╪═══════╪═════╣
║experimental │https://skipper-repository.cfapps.io/repository/experimental│false│0
║
║local        │https://10.55.13.45:7577                                      │true │1
║
╚═════════════╧════════════════════════════════════════════════════════════
════════════╧═══════╧═════╝
```

If a repository is local, it is backed by Skipper’s database and you can upload packages to the repository. If it is not local, it is a remote repository and you can only read packages. The packages in a remote repository are updated outside of Skipper’s control. The 1.0 release only polls the remote repository for contents upon server startup. Follow issue GH-262 for more on adding support for dynamic updating of remote repository metadata.

Chapter 27. Skipper Server Commands

Skipper's package commands include the following:

27.1. Config

This command configures the shell to reference the HTTP API endpoint of the Skipper Server.

NAME

skipper config - Configure the Spring Cloud Skipper REST server to use.

SYNOPSYS

skipper config **[--uri]** string] **[--username]** string] **[--password]** string] **[--credentials-provider-command]** string] **[--skip-ssl-validation]**

OPTIONS

--uri string

the location of the Spring Cloud Skipper REST endpoint
[Optional, default = localhost:7577/api]

--username string

the username for authenticated access to the Admin REST endpoint
[Optional, default = <none>]

--password string

the password for authenticated access to the Admin REST endpoint (valid only with a username)
[Optional, default = <none>]

--credentials-provider-command string

a command to run that outputs the HTTP credentials used for authentication
[Optional, default = <none>]

--skip-ssl-validation

accept any SSL certificate (even self-signed)
[Optional, default = <none>]

Configures shell as shown in the following example:

```
skipper:>skipper config --uri https://localhost:8443/api
```

When using OAuth, you can use the username and password options.

From within the Skipper Shell you can also provide credentials, as shown in the following example:

```
skipper:> skipper config --uri https://localhost:7577/api --username my_username
--password my_password
```

See the Security section for more information.

27.2. Info

This command shows server info.

NAME

skipper info - Show the Skipper server being used.

SYNOPSYS

skipper info

Show which server version is being used, as shown (with output) in the following example:

```
skipper:>info
Spring Cloud Skipper Server v1.0.0.2.5.0-SNAPSHOT
```

Chapter 28. Generic Usage

This section contains generic notes about commands.

28.1. Timeout Expression

- A regular long representation (using milliseconds as the default unit)
- The standard ISO-8601 format used by *java.util.Duration*
- A more readable format where the value and the unit are coupled (e.g. 10s means 10 seconds)

To specify a session timeout of 30 seconds, 30, PT30S and 30s are all equivalent. A read timeout of 500ms can be specified in any of the following form: 500, PT0.5S and 500ms.

You can also use any of the supported unit. These are:

- ns for nanoseconds
- ms for milliseconds
- s for seconds
- m for minutes
- h for hours
- d for days

Architecture

Skipper uses a basic client-server architecture. The server exposes a REST API that is used by the interactive shell. You can browse the API using familiar HTTP client tools. The server persists Package Metadata and Release state in a relational database.

Platforms are defined by using the following property prefix: `spring.cloud.skipper.server.platform`. For each of the supported platforms (`cloudfoundry`, 'kubernetes' and local), you can define multiple accounts. Each account maps onto an instance of a Spring Cloud Deployer implementation that is responsible for deploying the applications. The Installation shows more details, but it is important to note that the Skipper server is not tied to a deploying to a single platform. Wherever Skipper is running, it can be configured to deploy to any platform. For example, if Skipper is deployed on Cloud Foundry, you can still register accounts for Kubernetes and deploy apps to Kubernetes from Cloud Foundry.

The release workflow is currently a hard-coded workflow managed by the Spring Cloud State Machine project. The state of the State Machine is persisted in a relational database.

REST API Guide

This section covers the Spring Cloud Skipper REST API.

Chapter 29. Overview

Spring Cloud Skipper provides a REST API that lets you access all aspects of the server. The Spring Cloud Skipper shell is a first-class consumer of the API.

29.1. HTTP Verbs

Spring Cloud Skipper tries to adhere as closely as possible to standard HTTP and REST conventions in its use of HTTP verbs. The following table shows each verb and how Skipper uses it:

Verb	Usage
`GET`	Used to retrieve a resource.
`POST`	Used to create a new resource.
`PUT`	Used to update an existing resource, including partial updates. Also used for resources that imply the concept of `restarts`.
`DELETE`	Used to delete an existing resource.

29.2. HTTP Status Codes

Skipper adheres as closely as possible to standard HTTP and REST conventions in its use of HTTP status codes. The following table shows each status and its meaning in Skipper:

Status code	Usage
`200 OK`	The request completed successfully.
`201 Created`	A new resource has been created successfully. The resource's URI is available from the response's `Location` header.
`204 No Content`	An update to an existing resource has been applied successfully.
`400 Bad Request`	The request was malformed. The response body includes an error that provides further information.
`404 Not Found`	The requested resource does not exist.

29.3. Headers

Every response has the following header(s):

Name	Description
`Content-Type`	The `Content-Type` of the payload (for example `application/hal+json`).

29.4. Errors

Path	Type	Description
`error`	`String`	The HTTP error that occurred (for example, `Bad Request`).
`message`	`String`	A description of the cause of the error.
`path`	`String`	The path to which the request was made.
`status`	`Number`	The HTTP status code (for example `400`).
`timestamp`	`Number`	The time, in milliseconds, at which the error occurred.

29.5. Hypermedia

Spring Cloud Skipper uses hypermedia. As a result, resources include links to other resources in their responses. More specifically, responses are in Hypertext Application from resource to resource Language (HAL) format. Links can be found beneath the `_links` key. Consumers of the API should not create URIs themselves. Instead they should use the links in the resources to navigate.

Chapter 30. Resources

30.1. Index

The index provides the entry point into Spring Cloud Skipper's REST API.

30.1.1. Accessing the Index

You can use a `GET` request to access the index.

Request Structure

The following

```
GET /api HTTP/1.1
Host: localhost:7577
```

Example Request

```
$ curl 'http://localhost:7577/api' -i
```

Example Response

```
HTTP/1.1 200 OK
Vary: Origin
Vary: Access-Control-Request-Method
Vary: Access-Control-Request-Headers
Content-Type: application/hal+json
Content-Length: 1366

{
  "_links" : {
    "jpaRepositoryGuards" : {
      "href" : "http://localhost:7577/api/jpaRepositoryGuards"
    },
    "jpaRepositoryStateMachines" : {
      "href" : "http://localhost:7577/api/jpaRepositoryStateMachines"
    },
    "jpaRepositoryActions" : {
      "href" : "http://localhost:7577/api/jpaRepositoryActions"
    },
    "repositories" : {
      "href" : "http://localhost:7577/api/repositories{?page,size,sort}",
      "templated" : true
    },
    "jpaRepositoryStates" : {
```

```
      "href" : "http://localhost:7577/api/jpaRepositoryStates"
    },
    "releases" : {
      "href" : "http://localhost:7577/api/releases{?page,size,sort}",
      "templated" : true
    },
    "jpaRepositoryTransitions" : {
      "href" : "http://localhost:7577/api/jpaRepositoryTransitions"
    },
    "deployers" : {
      "href" : "http://localhost:7577/api/deployers{?page,size,sort}",
      "templated" : true
    },
    "packageMetadata" : {
      "href" :
 "http://localhost:7577/api/packageMetadata{?page,size,sort,projection}",
      "templated" : true
    },
    "about" : {
      "href" : "http://localhost:7577/api/about"
    },
    "release" : {
      "href" : "http://localhost:7577/api/release"
    },
    "package" : {
      "href" : "http://localhost:7577/api/package"
    },
    "profile" : {
      "href" : "http://localhost:7577/api/profile"
    }
  }
}
```

Links

The links are the main element of the index, as they let you traverse the API and invoke the desired functionality. The following table dsecribes the links:

Relation	Description
`repositories`	Exposes the 'package repository' repository.
`deployers`	Exposes the deployer repository.
`packageMetadata`	Exposes the package metadata repository.
`releases`	Exposes the release repository.
`profile`	Entrypoint to provide ALPS metadata that defines simple descriptions of application-level semantics.
`about`	Provides meta information about the server.
`release`	Exposes the release resource.

Relation	Description
package	Exposes the package resource.

30.2. Server

The Server resource exposes build and version information of the server.

30.2.1. Server info

A GET request returns meta information for Spring Cloud Skipper, including the following:

- Server name — typically spring-cloud-skipper-server
- Version of the server — for example, 2.5.0-SNAPSHOT

Request structure

```
GET /api/about HTTP/1.1
Accept: application/json
Host: localhost:7577
```

Example request

```
$ curl 'http://localhost:7577/api/about' -i \
    -H 'Accept: application/json'
```

Response structure

```
HTTP/1.1 200 OK
Content-Type: application/json
Content-Length: 260

{
  "versionInfo" : {
    "server" : {
      "name" : "Spring Cloud Skipper Server",
      "version" : "fake-server-version"
    },
    "shell" : {
      "name" : "Spring Cloud Skipper Shell",
      "version" : "fake-shell-version"
    }
  },
  "links" : [ ]
}
```

Response fields

Path	Type	Description
`versionInfo.server.name`	`String`	Spring Cloud Skipper Server dependency.
`versionInfo.server.version`	`String`	Spring Cloud Skipper Server dependency version.
`versionInfo.shell.name`	`String`	Spring Cloud Skipper Shell dependency.
`versionInfo.shell.version`	`String`	Spring Cloud Skipper Shell dependency version.
`links`	`Array`	Links.

30.3. Platforms

The Platforms (or Platform Deployer) resource is exported from the Spring Data Repository `DeployerRepository` and exposed by Spring Data REST.

30.3.1. Find All

A `GET` request returns a paginated list for all the Spring Cloud Skipper platform deployers.

Request structure

```
GET /api/deployers?page=0&size=10 HTTP/1.1
Host: localhost:7577
```

Example request

```
$ curl 'http://localhost:7577/api/deployers?page=0&size=10' -i
```

Response structure

```
HTTP/1.1 200 OK
Vary: Origin
Vary: Access-Control-Request-Method
Vary: Access-Control-Request-Headers
Content-Type: application/hal+json
Content-Length: 9142

{
  "_embedded" : {
    "deployers" : [ {
      "name" : "default",
      "type" : "local",
```

```
      "description" : "ShutdownTimeout = [30], EnvVarsToInherit =
[TMP,LANG,LANGUAGE,LC_.*,PATH,SPRING_APPLICATION_JSON], JavaCmd = [/opt/jdk8u232-
b09/jre/bin/java], WorkingDirectoriesRoot = [/tmp], DeleteFilesOnExit = [true]",
      "options" : [ {
        "id" : "spring.cloud.deployer.local.docker.network",
        "name" : "network",
        "type" : "java.lang.String",
        "description" : null,
        "shortDescription" : null,
        "defaultValue" : "bridge",
        "hints" : {
          "keyHints" : [ ],
          "keyProviders" : [ ],
          "valueHints" : [ ],
          "valueProviders" : [ ]
        },
        "deprecation" : null,
        "deprecated" : false
      }, {
        "id" : "spring.cloud.deployer.local.debug-suspend",
        "name" : "debug-suspend",
        "type" : "java.lang.String",
        "description" : null,
        "shortDescription" : null,
        "defaultValue" : null,
        "hints" : {
          "keyHints" : [ ],
          "keyProviders" : [ ],
          "valueHints" : [ {
            "value" : "y",
            "description" : null,
            "shortDescription" : null
          }, {
            "value" : "n",
            "description" : null,
            "shortDescription" : null
          } ],
          "valueProviders" : [ ]
        },
        "deprecation" : null,
        "deprecated" : false
      }, {
        "id" : "spring.cloud.deployer.local.working-directories-root",
        "name" : "working-directories-root",
        "type" : "java.nio.file.Path",
        "description" : "Directory in which all created processes will run and create
log files.",
        "shortDescription" : "Directory in which all created processes will run and
create log files.",
        "defaultValue" : null,
        "hints" : {
```

```
          "keyHints" : [ ],
          "keyProviders" : [ ],
          "valueHints" : [ ],
          "valueProviders" : [ ]
        },
        "deprecation" : null,
        "deprecated" : false
      }, {
        "id" : "spring.cloud.deployer.local.java-opts",
        "name" : "java-opts",
        "type" : "java.lang.String",
        "description" : "The Java Options to pass to the JVM, e.g -Dtest=foo",
        "shortDescription" : "The Java Options to pass to the JVM, e.g -Dtest=foo",
        "defaultValue" : null,
        "hints" : {
          "keyHints" : [ ],
          "keyProviders" : [ ],
          "valueHints" : [ ],
          "valueProviders" : [ ]
        },
        "deprecation" : null,
        "deprecated" : false
      }, {
        "id" : "spring.cloud.deployer.local.use-spring-application-json",
        "name" : "use-spring-application-json",
        "type" : "java.lang.Boolean",
        "description" : "Flag to indicate whether application properties are passed as
command line args or in a SPRING_APPLICATION_JSON environment variable. Default value
is {@code true}.",
        "shortDescription" : "Flag to indicate whether application properties are
passed as command line args or in a SPRING_APPLICATION_JSON environment variable.",
        "defaultValue" : true,
        "hints" : {
          "keyHints" : [ ],
          "keyProviders" : [ ],
          "valueHints" : [ ],
          "valueProviders" : [ ]
        },
        "deprecation" : null,
        "deprecated" : false
      }, {
        "id" : "spring.cloud.deployer.local.inherit-logging",
        "name" : "inherit-logging",
        "type" : "java.lang.Boolean",
        "description" : null,
        "shortDescription" : null,
        "defaultValue" : false,
        "hints" : {
          "keyHints" : [ ],
          "keyProviders" : [ ],
          "valueHints" : [ ],
```

```
          "valueProviders" : [ ]
        },
        "deprecation" : null,
        "deprecated" : false
      }, {
        "id" : "spring.cloud.deployer.local.debug-port",
        "name" : "debug-port",
        "type" : "java.lang.Integer",
        "description" : null,
        "shortDescription" : null,
        "defaultValue" : null,
        "hints" : {
          "keyHints" : [ ],
          "keyProviders" : [ ],
          "valueHints" : [ ],
          "valueProviders" : [ ]
        },
        "deprecation" : null,
        "deprecated" : false
      }, {
        "id" : "spring.cloud.deployer.local.delete-files-on-exit",
        "name" : "delete-files-on-exit",
        "type" : "java.lang.Boolean",
        "description" : "Whether to delete created files and directories on JVM
exit.",
        "shortDescription" : "Whether to delete created files and directories on JVM
exit.",
        "defaultValue" : true,
        "hints" : {
          "keyHints" : [ ],
          "keyProviders" : [ ],
          "valueHints" : [ ],
          "valueProviders" : [ ]
        },
        "deprecation" : null,
        "deprecated" : false
      }, {
        "id" : "spring.cloud.deployer.local.env-vars-to-inherit",
        "name" : "env-vars-to-inherit",
        "type" : "java.lang.String[]",
        "description" : "Array of regular expression patterns for environment
variables that should be passed to launched applications.",
        "shortDescription" : "Array of regular expression patterns for environment
variables that should be passed to launched applications.",
        "defaultValue" : null,
        "hints" : {
          "keyHints" : [ ],
          "keyProviders" : [ ],
          "valueHints" : [ ],
          "valueProviders" : [ ]
        },
```

```
        "deprecation" : null,
        "deprecated" : false
      }, {
        "id" : "spring.cloud.deployer.local.java-cmd",
        "name" : "java-cmd",
        "type" : "java.lang.String",
        "description" : "The command to run java.",
        "shortDescription" : "The command to run java.",
        "defaultValue" : null,
        "hints" : {
          "keyHints" : [ ],
          "keyProviders" : [ ],
          "valueHints" : [ ],
          "valueProviders" : [ ]
        },
        "deprecation" : null,
        "deprecated" : false
      }, {
        "id" : "spring.cloud.deployer.local.shutdown-timeout",
        "name" : "shutdown-timeout",
        "type" : "java.lang.Integer",
        "description" : "Maximum number of seconds to wait for application shutdown.
via the {@code /shutdown} endpoint. A timeout value of 0 specifies an infinite
timeout. Default is 30 seconds.",
        "shortDescription" : "Maximum number of seconds to wait for application
shutdown. via the {@code /shutdown} endpoint.",
        "defaultValue" : 30,
        "hints" : {
          "keyHints" : [ ],
          "keyProviders" : [ ],
          "valueHints" : [ ],
          "valueProviders" : [ ]
        },
        "deprecation" : null,
        "deprecated" : false
      }, {
        "id" : "spring.cloud.deployer.local.maximum-concurrent-tasks",
        "name" : "maximum-concurrent-tasks",
        "type" : "java.lang.Integer",
        "description" : "The maximum concurrent tasks allowed for this platform
instance.",
        "shortDescription" : "The maximum concurrent tasks allowed for this platform
instance.",
        "defaultValue" : 20,
        "hints" : {
          "keyHints" : [ ],
          "keyProviders" : [ ],
          "valueHints" : [ ],
          "valueProviders" : [ ]
        },
        "deprecation" : null,
```

```
        "deprecated" : false
      }, {
        "id" : "spring.cloud.deployer.local.port-range.high",
        "name" : "high",
        "type" : "java.lang.Integer",
        "description" : "Upper bound for computing applications's random port.",
        "shortDescription" : "Upper bound for computing applications's random port.",
        "defaultValue" : 61000,
        "hints" : {
          "keyHints" : [ ],
          "keyProviders" : [ ],
          "valueHints" : [ ],
          "valueProviders" : [ ]
        },
        "deprecation" : null,
        "deprecated" : false
      }, {
        "id" : "spring.cloud.deployer.local.port-range.low",
        "name" : "low",
        "type" : "java.lang.Integer",
        "description" : "Lower bound for computing applications's random port.",
        "shortDescription" : "Lower bound for computing applications's random port.",
        "defaultValue" : 20000,
        "hints" : {
          "keyHints" : [ ],
          "keyProviders" : [ ],
          "valueHints" : [ ],
          "valueProviders" : [ ]
        },
        "deprecation" : null,
        "deprecated" : false
      } ],
      "_links" : {
        "self" : {
          "href" : "http://localhost:7577/api/deployers/26c17de0-d151-424c-ab35-
afc8aa07bbc4"
        },
        "deployer" : {
          "href" : "http://localhost:7577/api/deployers/26c17de0-d151-424c-ab35-
afc8aa07bbc4"
        }
      }
    } ]
  },
  "_links" : {
    "self" : {
      "href" : "http://localhost:7577/api/deployers{&sort}",
      "templated" : true
    },
    "profile" : {
      "href" : "http://localhost:7577/api/profile/deployers"
```

```
    },
    "search" : {
      "href" : "http://localhost:7577/api/deployers/search"
    }
  },
  "page" : {
    "size" : 10,
    "totalElements" : 1,
    "totalPages" : 1,
    "number" : 0
  }
}
```

Response fields

Path	Type	Description
page	Object	Pagination properties
page.size	Number	The size of the page being returned
page.totalElements	Number	Total elements available for pagination
page.totalPages	Number	Total amount of pages
page.number	Number	Page number of the page returned (zero-based)
_embedded.deployers	Array	Array containing Deployer objects
_embedded.deployers[].name	String	Name of the deployer
_embedded.deployers[].type	String	Type of the deployer (e.g. 'local')
_embedded.deployers[].description	String	Description providing some deployer properties
_embedded.deployers[].options	Array	Array containing Deployer deployment properties
_embedded.deployers[].options[].id	String	Deployment property id
_embedded.deployers[].options[].name	String	Deployment property name
_embedded.deployers[].options[].type	String	Deployment property type
_embedded.deployers[].options[].description	String	Deployment property description
_embedded.deployers[].options[].shortDescription	String	Deployment property short description
_embedded.deployers[].options[].defaultValue	Varies	Deployment property default value

Path	Type	Description
`_embedded.deployers[].options[].hints`	`Object`	Object containing deployment property hints
`_embedded.deployers[].options[].hints.keyHints`	`Array`	Deployment property key hints
`_embedded.deployers[].options[].hints.keyProviders`	`Array`	Deployment property key hint providers
`_embedded.deployers[].options[].hints.valueHints`	`Array`	Deployment property value hints
`_embedded.deployers[].options[].hints.valueProviders`	`Array`	Deployment property value hint providers
`_embedded.deployers[].options[].deprecation`	`Null`	
`_embedded.deployers[].options[].deprecated`	`Boolean`	

30.4. Packages

The Packages resource is exported from the Spring Data Repository `PackageMetadata` and exposed by Spring Data REST.

30.4.1. Search

A `GET` request will return a paginated list for all Spring Cloud Skipper package metadata.

Request structure

```
GET /api/packageMetadata?page=0&size=10 HTTP/1.1
Host: localhost:7577
```

Example request

```
$ curl 'http://localhost:7577/api/packageMetadata?page=0&size=10' -i
```

Response structure

```
HTTP/1.1 200 OK
Vary: Origin
Vary: Access-Control-Request-Method
Vary: Access-Control-Request-Headers
Content-Type: application/hal+json
Content-Length: 4657

{
  "_embedded" : {
```

```
    "packageMetadata" : [ {
      "apiVersion" : "skipper.spring.io/v1",
      "origin" : null,
      "repositoryId" : 2,
      "repositoryName" : "local",
      "kind" : "SkipperPackageMetadata",
      "name" : "log",
      "displayName" : null,
      "version" : "1.0.0",
      "packageSourceUrl" : "https://github.com/spring-cloud-stream-app-
starters/log/tree/v1.2.0.RC1",
      "packageHomeUrl" : "https://cloud.spring.io/spring-cloud-stream-app-starters/",
      "tags" : "logging, sink",
      "maintainer" : "https://github.com/sobychacko",
      "description" : "The log sink uses the application logger to output the data for
inspection.",
      "sha256" : null,
      "iconUrl" : null,
      "_links" : {
        "self" : {
          "href" : "http://localhost:7577/api/packageMetadata/3"
        },
        "packageMetadata" : {
          "href" : "http://localhost:7577/api/packageMetadata/3{?projection}",
          "templated" : true
        },
        "install" : {
          "href" : "http://localhost:7577/api/package/install/3"
        }
      }
    }, {
      "apiVersion" : "skipper.spring.io/v1",
      "origin" : null,
      "repositoryId" : 2,
      "repositoryName" : "local",
      "kind" : "SkipperPackageMetadata",
      "name" : "log",
      "displayName" : null,
      "version" : "1.0.0",
      "packageSourceUrl" : "https://github.com/spring-cloud-stream-app-
starters/log/tree/v1.2.0.RC1",
      "packageHomeUrl" : "https://cloud.spring.io/spring-cloud-stream-app-starters/",
      "tags" : "logging, sink",
      "maintainer" : "https://github.com/sobychacko",
      "description" : "The log sink uses the application logger to output the data for
inspection.",
      "sha256" : null,
      "iconUrl" : null,
      "_links" : {
        "self" : {
          "href" : "http://localhost:7577/api/packageMetadata/4"
```

```
        },
        "packageMetadata" : {
          "href" : "http://localhost:7577/api/packageMetadata/4{?projection}",
          "templated" : true
        },
        "install" : {
          "href" : "http://localhost:7577/api/package/install/4"
        }
      }
    }, {
      "apiVersion" : "skipper.spring.io/v1",
      "origin" : null,
      "repositoryId" : 2,
      "repositoryName" : "local",
      "kind" : "SkipperPackageMetadata",
      "name" : "log",
      "displayName" : null,
      "version" : "1.0.0",
      "packageSourceUrl" : "https://github.com/spring-cloud-stream-app-
starters/log/tree/v1.2.0.RC1",
      "packageHomeUrl" : "https://cloud.spring.io/spring-cloud-stream-app-starters/",
      "tags" : "logging, sink",
      "maintainer" : "https://github.com/sobychacko",
      "description" : "The log sink uses the application logger to output the data for
inspection.",
      "sha256" : null,
      "iconUrl" : null,
      "_links" : {
        "self" : {
          "href" : "http://localhost:7577/api/packageMetadata/5"
        },
        "packageMetadata" : {
          "href" : "http://localhost:7577/api/packageMetadata/5{?projection}",
          "templated" : true
        },
        "install" : {
          "href" : "http://localhost:7577/api/package/install/5"
        }
      }
    }, {
      "apiVersion" : "skipper.spring.io/v1",
      "origin" : null,
      "repositoryId" : 2,
      "repositoryName" : "local",
      "kind" : "SkipperPackageMetadata",
      "name" : "log",
      "displayName" : null,
      "version" : "1.0.0",
      "packageSourceUrl" : "https://github.com/spring-cloud-stream-app-
starters/log/tree/v1.2.0.RC1",
      "packageHomeUrl" : "https://cloud.spring.io/spring-cloud-stream-app-starters/",
```

```
      "tags" : "logging, sink",
      "maintainer" : "https://github.com/sobychacko",
      "description" : "The log sink uses the application logger to output the data for
inspection.",
      "sha256" : null,
      "iconUrl" : null,
      "_links" : {
        "self" : {
          "href" : "http://localhost:7577/api/packageMetadata/6"
        },
        "packageMetadata" : {
          "href" : "http://localhost:7577/api/packageMetadata/6{?projection}",
          "templated" : true
        },
        "install" : {
          "href" : "http://localhost:7577/api/package/install/6"
        }
      }
    } ]
  },
  "_links" : {
    "self" : {
      "href" : "http://localhost:7577/api/packageMetadata{&sort,projection}",
      "templated" : true
    },
    "profile" : {
      "href" : "http://localhost:7577/api/profile/packageMetadata"
    },
    "search" : {
      "href" : "http://localhost:7577/api/packageMetadata/search"
    }
  },
  "page" : {
    "size" : 10,
    "totalElements" : 4,
    "totalPages" : 1,
    "number" : 0
  }
}
```

Response fields

Path	Type	Description
`page`	`Object`	Pagination properties
`page.size`	`Number`	The size of the page being returned
`page.totalElements`	`Number`	Total elements available for pagination
`page.totalPages`	`Number`	Total amount of pages

Path	Type	Description
page.number	Number	Page number of the page returned (zero-based)
_embedded.packageMetadata	Array	Contains a collection of Package Metadata items
_embedded.packageMetadata[].apiVersion	String	The Package Index spec version this file is based on
_embedded.packageMetadata[].origin	Null	Indicates the origin of the repository (free form text)
_embedded.packageMetadata[].repositoryId	Number	The repository ID this Package belongs to
_embedded.packageMetadata[].repositoryName	String	The repository name this Package belongs to.
_embedded.packageMetadata[].kind	String	What type of package system is being used
_embedded.packageMetadata[].name	String	The name of the package
_embedded.packageMetadata[].displayName	Null	Display name of the release
_embedded.packageMetadata[].version	String	The version of the package
_embedded.packageMetadata[].packageSourceUrl	String	Location to source code for this package
_embedded.packageMetadata[].packageHomeUrl	String	The home page of the package
_embedded.packageMetadata[].tags	String	A comma separated list of tags to use for searching
_embedded.packageMetadata[].maintainer	String	Who is maintaining this package
_embedded.packageMetadata[].description	String	Brief description of the package
_embedded.packageMetadata[].sha256	Null	Hash of package binary that will be downloaded using SHA256 hash algorithm
_embedded.packageMetadata[].iconUrl	Null	Url location of a icon

30.4.2. Search summary

A `GET` request returns the list of available package metadata with the summary information of each package.

Request structure

```
GET /api/packageMetadata?projection=summary HTTP/1.1
Host: localhost:7577
```

Example request

```
$ curl 'http://localhost:7577/api/packageMetadata?projection=summary' -i
```

Response structure

```
HTTP/1.1 200 OK
Vary: Origin
Vary: Access-Control-Request-Method
Vary: Access-Control-Request-Headers
Content-Type: application/hal+json
Content-Length: 2296

{
  "_embedded" : {
    "packageMetadata" : [ {
      "version" : "1.0.0",
      "description" : "The log sink uses the application logger to output the data for
inspection.",
      "repositoryName" : "local",
      "iconUrl" : null,
      "name" : "log",
      "id" : "3",
      "_links" : {
        "self" : {
          "href" : "http://localhost:7577/api/packageMetadata/3"
        },
        "packageMetadata" : {
          "href" : "http://localhost:7577/api/packageMetadata/3{?projection}",
          "templated" : true
        },
        "install" : {
          "href" : "http://localhost:7577/api/package/install/3"
        }
      }
    }, {
      "version" : "1.0.0",
      "description" : "The log sink uses the application logger to output the data for
inspection.",
      "repositoryName" : "local",
      "iconUrl" : null,
      "name" : "log",
      "id" : "4",
      "_links" : {
        "self" : {
```

```
          "href" : "http://localhost:7577/api/packageMetadata/4"
        },
        "packageMetadata" : {
          "href" : "http://localhost:7577/api/packageMetadata/4{?projection}",
          "templated" : true
        },
        "install" : {
          "href" : "http://localhost:7577/api/package/install/4"
        }
      }
    }, {
      "version" : "1.0.0",
      "description" : "The log sink uses the application logger to output the data for
inspection.",
      "repositoryName" : "local",
      "iconUrl" : null,
      "name" : "log",
      "id" : "5",
      "_links" : {
        "self" : {
          "href" : "http://localhost:7577/api/packageMetadata/5"
        },
        "packageMetadata" : {
          "href" : "http://localhost:7577/api/packageMetadata/5{?projection}",
          "templated" : true
        },
        "install" : {
          "href" : "http://localhost:7577/api/package/install/5"
        }
      }
    } ]
  },
  "_links" : {
    "self" : {
      "href" :
"http://localhost:7577/api/packageMetadata{?page,size,sort,projection}",
      "templated" : true
    },
    "profile" : {
      "href" : "http://localhost:7577/api/profile/packageMetadata"
    },
    "search" : {
      "href" : "http://localhost:7577/api/packageMetadata/search"
    }
  },
  "page" : {
    "size" : 20,
    "totalElements" : 3,
    "totalPages" : 1,
    "number" : 0
  }
```

```
}
```

Response fields

Path	Type	Description
`page`	`Object`	Pagination properties
`page.size`	`Number`	The size of the page being returned
`page.totalElements`	`Number`	Total elements available for pagination
`page.totalPages`	`Number`	Total amount of pages
`page.number`	`Number`	Page number of the page returned (zero-based)
`_embedded.packageMetadata[].id`	`String`	Identifier of the package metadata
`_embedded.packageMetadata[].iconUrl`	`Null`	Url location of a icon
`_embedded.packageMetadata[].repositoryName`	`String`	The repository name this Package belongs to.
`_embedded.packageMetadata[].version`	`String`	The version of the package
`_embedded.packageMetadata[].name`	`String`	The name of the package
`_embedded.packageMetadata[].description`	`String`	Brief description of the package
`_embedded.packageMetadata[]._links.self.href`	`String`	self link
`_embedded.packageMetadata[]._links.packageMetadata.href`	`String`	link to full package metadata
`_embedded.packageMetadata[]._links.install.href`	`String`	link to install the package

30.4.3. Search with details

A `GET` request returns the details of a package using the `id` of the package.

Request structure

```
GET /api/packageMetadata/3 HTTP/1.1
Host: localhost:7577
```

Example request

```
$ curl 'http://localhost:7577/api/packageMetadata/3' -i
```

Response structure

```
HTTP/1.1 200 OK
Vary: Origin
Vary: Access-Control-Request-Method
Vary: Access-Control-Request-Headers
ETag: "0"
Content-Type: application/hal+json
Content-Length: 931

{
  "apiVersion" : "skipper.spring.io/v1",
  "origin" : null,
  "repositoryId" : 2,
  "repositoryName" : "local",
  "kind" : "SkipperPackageMetadata",
  "name" : "log",
  "displayName" : null,
  "version" : "1.0.0",
  "packageSourceUrl" : "https://github.com/spring-cloud-stream-app-
starters/log/tree/v1.2.0.RC1",
  "packageHomeUrl" : "https://cloud.spring.io/spring-cloud-stream-app-starters/",
  "tags" : "logging, sink",
  "maintainer" : "https://github.com/sobychacko",
  "description" : "The log sink uses the application logger to output the data for
inspection.",
  "sha256" : null,
  "iconUrl" : null,
  "_links" : {
    "self" : {
      "href" : "http://localhost:7577/api/packageMetadata/3"
    },
    "packageMetadata" : {
      "href" : "http://localhost:7577/api/packageMetadata/3{?projection}",
      "templated" : true
    },
    "install" : {
      "href" : "http://localhost:7577/api/package/install/3"
    }
  }
}
```

Response fields

Path	Type	Description
`apiVersion`	`String`	The Package Index spec version this file is based on
`origin`	`Null`	Indicates the origin of the repository (free form text)

Path	Type	Description
repositoryId	Number	The repository ID this Package belongs to.
repositoryName	String	The repository name this Package belongs to.
kind	String	What type of package system is being used
name	String	The name of the package
displayName	Null	The display name of the package
version	String	The version of the package
packageSourceUrl	String	Location to source code for this package
packageHomeUrl	String	The home page of the package
tags	String	A comma separated list of tags to use for searching
maintainer	String	Who is maintaining this package
description	String	Brief description of the package
sha256	Null	Hash of package binary that will be downloaded using SHA256 hash algorithm
iconUrl	Null	Url location of a icon

30.4.4. Search by Package Name

A GET request returns a list of all the Spring Cloud Skipper package metadata for the given package name.

Request structure

getPackageMetadataSearchFindByName

```
GET /api/packageMetadata/search/findByName?name=log HTTP/1.1
Host: localhost:7577
```

Example request

```
$ curl 'http://localhost:7577/api/packageMetadata/search/findByName?name=log' -i
```

Response structure

```
HTTP/1.1 200 OK
```

```
Vary: Origin
Vary: Access-Control-Request-Method
Vary: Access-Control-Request-Headers
Content-Type: application/hal+json
Content-Length: 5404

{
  "_embedded" : {
    "packageMetadata" : [ {
      "apiVersion" : "skipper.spring.io/v1",
      "origin" : null,
      "repositoryId" : 2,
      "repositoryName" : "local",
      "kind" : "SkipperPackageMetadata",
      "name" : "log",
      "displayName" : null,
      "version" : "1.0.0",
      "packageSourceUrl" : "https://github.com/spring-cloud-stream-app-
starters/log/tree/v1.2.0.RC1",
      "packageHomeUrl" : "https://cloud.spring.io/spring-cloud-stream-app-starters/",
      "tags" : "logging, sink",
      "maintainer" : "https://github.com/sobychacko",
      "description" : "The log sink uses the application logger to output the data for
inspection.",
      "sha256" : null,
      "iconUrl" : null,
      "_links" : {
        "self" : {
          "href" : "http://localhost:7577/api/packageMetadata/3"
        },
        "packageMetadata" : {
          "href" : "http://localhost:7577/api/packageMetadata/3{?projection}",
          "templated" : true
        },
        "install" : {
          "href" : "http://localhost:7577/api/package/install/3"
        }
      }
    }, {
      "apiVersion" : "skipper.spring.io/v1",
      "origin" : null,
      "repositoryId" : 2,
      "repositoryName" : "local",
      "kind" : "SkipperPackageMetadata",
      "name" : "log",
      "displayName" : null,
      "version" : "1.0.0",
      "packageSourceUrl" : "https://github.com/spring-cloud-stream-app-
starters/log/tree/v1.2.0.RC1",
      "packageHomeUrl" : "https://cloud.spring.io/spring-cloud-stream-app-starters/",
      "tags" : "logging, sink",
```

```
      "maintainer" : "https://github.com/sobychacko",
      "description" : "The log sink uses the application logger to output the data for
inspection.",
      "sha256" : null,
      "iconUrl" : null,
      "_links" : {
        "self" : {
          "href" : "http://localhost:7577/api/packageMetadata/4"
        },
        "packageMetadata" : {
          "href" : "http://localhost:7577/api/packageMetadata/4{?projection}",
          "templated" : true
        },
        "install" : {
          "href" : "http://localhost:7577/api/package/install/4"
        }
      }
    }, {
      "apiVersion" : "skipper.spring.io/v1",
      "origin" : null,
      "repositoryId" : 2,
      "repositoryName" : "local",
      "kind" : "SkipperPackageMetadata",
      "name" : "log",
      "displayName" : null,
      "version" : "1.0.0",
      "packageSourceUrl" : "https://github.com/spring-cloud-stream-app-
starters/log/tree/v1.2.0.RC1",
      "packageHomeUrl" : "https://cloud.spring.io/spring-cloud-stream-app-starters/",
      "tags" : "logging, sink",
      "maintainer" : "https://github.com/sobychacko",
      "description" : "The log sink uses the application logger to output the data for
inspection.",
      "sha256" : null,
      "iconUrl" : null,
      "_links" : {
        "self" : {
          "href" : "http://localhost:7577/api/packageMetadata/5"
        },
        "packageMetadata" : {
          "href" : "http://localhost:7577/api/packageMetadata/5{?projection}",
          "templated" : true
        },
        "install" : {
          "href" : "http://localhost:7577/api/package/install/5"
        }
      }
    }, {
      "apiVersion" : "skipper.spring.io/v1",
      "origin" : null,
      "repositoryId" : 2,
```

```
      "repositoryName" : "local",
      "kind" : "SkipperPackageMetadata",
      "name" : "log",
      "displayName" : null,
      "version" : "1.0.0",
      "packageSourceUrl" : "https://github.com/spring-cloud-stream-app-
starters/log/tree/v1.2.0.RC1",
      "packageHomeUrl" : "https://cloud.spring.io/spring-cloud-stream-app-starters/",
      "tags" : "logging, sink",
      "maintainer" : "https://github.com/sobychacko",
      "description" : "The log sink uses the application logger to output the data for
inspection.",
      "sha256" : null,
      "iconUrl" : null,
      "_links" : {
        "self" : {
          "href" : "http://localhost:7577/api/packageMetadata/6"
        },
        "packageMetadata" : {
          "href" : "http://localhost:7577/api/packageMetadata/6{?projection}",
          "templated" : true
        },
        "install" : {
          "href" : "http://localhost:7577/api/package/install/6"
        }
      }
    }, {
      "apiVersion" : "skipper.spring.io/v1",
      "origin" : null,
      "repositoryId" : 2,
      "repositoryName" : "local",
      "kind" : "SkipperPackageMetadata",
      "name" : "log",
      "displayName" : null,
      "version" : "1.0.0",
      "packageSourceUrl" : "https://github.com/spring-cloud-stream-app-
starters/log/tree/v1.2.0.RC1",
      "packageHomeUrl" : "https://cloud.spring.io/spring-cloud-stream-app-starters/",
      "tags" : "logging, sink",
      "maintainer" : "https://github.com/sobychacko",
      "description" : "The log sink uses the application logger to output the data for
inspection.",
      "sha256" : null,
      "iconUrl" : null,
      "_links" : {
        "self" : {
          "href" : "http://localhost:7577/api/packageMetadata/7"
        },
        "packageMetadata" : {
          "href" : "http://localhost:7577/api/packageMetadata/7{?projection}",
          "templated" : true
```

```
        },
        "install" : {
          "href" : "http://localhost:7577/api/package/install/7"
        }
      }
    } ]
  },
  "_links" : {
    "self" : {
      "href" : "http://localhost:7577/api/packageMetadata/search/findByName?name=log"
    }
  }
}
```

Response fields

Path	Type	Description
`_embedded.packageMetadata[].apiVersion`	`String`	The Package Index spec version this file is based on
`_embedded.packageMetadata[].origin`	`Null`	Indicates the origin of the repository (free form text)
`_embedded.packageMetadata[].repositoryId`	`Number`	The repository ID this Package belongs to.
`_embedded.packageMetadata[].repositoryName`	`String`	The repository name this Package belongs to.
`_embedded.packageMetadata[].kind`	`String`	What type of package system is being used
`_embedded.packageMetadata[].name`	`String`	The name of the package
`_embedded.packageMetadata[].displayName`	`Null`	The display name of the package
`_embedded.packageMetadata[].version`	`String`	The version of the package
`_embedded.packageMetadata[].packageSourceUrl`	`String`	Location to source code for this package
`_embedded.packageMetadata[].packageHomeUrl`	`String`	The home page of the package
`_embedded.packageMetadata[].tags`	`String`	A comma separated list of tags to use for searching
`_embedded.packageMetadata[].maintainer`	`String`	Who is maintaining this package
`_embedded.packageMetadata[].description`	`String`	Brief description of the package
`_embedded.packageMetadata[].sha256`	`Null`	Hash of package binary that will be downloaded using SHA256 hash algorithm

Path	Type	Description
`_embedded.packageMetadata[].iconUrl`	`Null`	Url location of a icon

30.4.5. Search by Package Name, Ignoring Case

A `GET` request returns a list for all Spring Cloud Skipper package metadata by the given package name ignoring case.

Request structure

```
GET /api/packageMetadata/search/findByNameContainingIgnoreCase?name=LO HTTP/1.1
Host: localhost:7577
```

Example request

```
$ curl
'http://localhost:7577/api/packageMetadata/search/findByNameContainingIgnoreCase?name=
LO' -i
```

Response structure

```
HTTP/1.1 200 OK
Vary: Origin
Vary: Access-Control-Request-Method
Vary: Access-Control-Request-Headers
Content-Type: application/hal+json
Content-Length: 2288

{
  "_embedded" : {
    "packageMetadata" : [ {
      "apiVersion" : "skipper.spring.io/v1",
      "origin" : null,
      "repositoryId" : 2,
      "repositoryName" : "local",
      "kind" : "SkipperPackageMetadata",
      "name" : "log",
      "displayName" : null,
      "version" : "1.0.0",
      "packageSourceUrl" : "https://github.com/spring-cloud-stream-app-
starters/log/tree/v1.2.0.RC1",
      "packageHomeUrl" : "https://cloud.spring.io/spring-cloud-stream-app-starters/",
      "tags" : "logging, sink",
      "maintainer" : "https://github.com/sobychacko",
      "description" : "The log sink uses the application logger to output the data for
inspection.",
```

```
      "sha256" : null,
      "iconUrl" : null,
      "_links" : {
        "self" : {
          "href" : "http://localhost:7577/api/packageMetadata/3"
        },
        "packageMetadata" : {
          "href" : "http://localhost:7577/api/packageMetadata/3{?projection}",
          "templated" : true
        },
        "install" : {
          "href" : "http://localhost:7577/api/package/install/3"
        }
      }
    }, {
      "apiVersion" : "skipper.spring.io/v1",
      "origin" : null,
      "repositoryId" : 2,
      "repositoryName" : "local",
      "kind" : "SkipperPackageMetadata",
      "name" : "log",
      "displayName" : null,
      "version" : "1.0.0",
      "packageSourceUrl" : "https://github.com/spring-cloud-stream-app-
starters/log/tree/v1.2.0.RC1",
      "packageHomeUrl" : "https://cloud.spring.io/spring-cloud-stream-app-starters/",
      "tags" : "logging, sink",
      "maintainer" : "https://github.com/sobychacko",
      "description" : "The log sink uses the application logger to output the data for
inspection.",
      "sha256" : null,
      "iconUrl" : null,
      "_links" : {
        "self" : {
          "href" : "http://localhost:7577/api/packageMetadata/4"
        },
        "packageMetadata" : {
          "href" : "http://localhost:7577/api/packageMetadata/4{?projection}",
          "templated" : true
        },
        "install" : {
          "href" : "http://localhost:7577/api/package/install/4"
        }
      }
    } ]
  },
  "_links" : {
    "self" : {
      "href" :
"http://localhost:7577/api/packageMetadata/search/findByNameContainingIgnoreCase?name=
LO"
```

```
    }
  }
}
```

Response fields

Path	Type	Description
`_embedded.packageMetadata[].apiVersion`	`String`	The Package Index spec version this file is based on
`_embedded.packageMetadata[].origin`	`Null`	Indicates the origin of the repository (free form text)
`_embedded.packageMetadata[].repositoryId`	`Number`	The repository ID this Package belongs to.
`_embedded.packageMetadata[].repositoryName`	`String`	The repository name this Package belongs to.
`_embedded.packageMetadata[].kind`	`String`	What type of package system is being used
`_embedded.packageMetadata[].name`	`String`	The name of the package
`_embedded.packageMetadata[].displayName`	`Null`	The display name of the package
`_embedded.packageMetadata[].version`	`String`	The version of the package
`_embedded.packageMetadata[].packageSourceUrl`	`String`	Location to source code for this package
`_embedded.packageMetadata[].packageHomeUrl`	`String`	The home page of the package
`_embedded.packageMetadata[].tags`	`String`	A comma separated list of tags to use for searching
`_embedded.packageMetadata[].maintainer`	`String`	Who is maintaining this package
`_embedded.packageMetadata[].description`	`String`	Brief description of the package
`_embedded.packageMetadata[].sha256`	`Null`	Hash of package binary that will be downloaded using SHA256 hash algorithm
`_embedded.packageMetadata[].iconUrl`	`Null`	Url location of a icon

30.5. Package

The Package resource maps onto the PackageController for uploading and installing packages.

30.5.1. Upload

The `upload` link uploads a package into a the `local` database backed repository.

Request structure

```
POST /api/package/upload HTTP/1.1
Content-Type: application/json;charset=UTF-8
Accept: application/json
Host: localhost:7577
```

Example request

```
$ curl 'http://localhost:7577/api/package/upload' -i -X POST \
    -H 'Content-Type: application/json;charset=UTF-8' \
    -H 'Accept: application/json'
```

Response structure

```
HTTP/1.1 201 Created
Content-Type: application/json
Content-Length: 805

{
  "apiVersion" : "skipper.spring.io/v1",
  "origin" : null,
  "repositoryId" : null,
  "repositoryName" : null,
  "kind" : "SkipperPackageMetadata",
  "name" : "log",
  "displayName" : null,
  "version" : "1.0.0",
  "packageSourceUrl" : "https://github.com/spring-cloud-stream-app-
starters/log/tree/v1.2.0.RC1",
  "packageHomeUrl" : "https://cloud.spring.io/spring-cloud-stream-app-starters/",
  "tags" : "logging, sink",
  "maintainer" : "https://github.com/sobychacko",
  "description" : "The log sink uses the application logger to output the data for
inspection.",
  "sha256" : null,
  "iconUrl" : null,
  "links" : [ {
    "rel" : "install",
    "href" : "http://localhost:7577/api/package/install"
  }, {
    "rel" : "install",
    "href" : "http://localhost:7577/api/package/install/{id}"
  } ]
}
```

Response fields

Path	Type	Description
apiVersion	String	The Package Index spec version this file is based on
origin	Null	Indicates the origin of the repository (free form text)
repositoryId	Null	The repository ID this Package belongs to.
repositoryName	Null	The repository nane this Package belongs to.
kind	String	What type of package system is being used
name	String	The name of the package
displayName	Null	The display name of the package
version	String	The version of the package
packageSourceUrl	String	Location to source code for this package
packageHomeUrl	String	The home page of the package
tags	String	A comma separated list of tags to use for searching
maintainer	String	Who is maintaining this package
description	String	Brief description of the package
sha256	Null	Hash of package binary that will be downloaded using SHA256 hash algorithm
iconUrl	Null	Url location of a icon

30.5.2. Install

The `install` link can install a package (identified by the `InstallRequest`) into the target platform.

Request structure

```
POST /api/package/install HTTP/1.1
Content-Type: application/json;charset=UTF-8
Accept: application/json
Host: localhost:7577
```

Example request

```
$ curl 'http://localhost:7577/api/package/install' -i -X POST \
    -H 'Content-Type: application/json;charset=UTF-8' \
    -H 'Accept: application/json'
```

Response structure

```
HTTP/1.1 201 Created
Content-Type: application/json
Content-Length: 2624

{
  "name" : "test",
  "version" : 1,
  "info" : {
    "status" : {
      "statusCode" : "DELETED",
      "platformStatus" : null
    },
    "firstDeployed" : null,
    "lastDeployed" : null,
    "deleted" : null,
    "description" : null
  },
  "pkg" : {
    "metadata" : {
      "apiVersion" : "skipper.spring.io/v1",
      "origin" : null,
      "repositoryId" : null,
      "repositoryName" : null,
      "kind" : "SkipperPackageMetadata",
      "name" : "log",
      "displayName" : null,
      "version" : "1.0.0",
      "packageSourceUrl" : "https://github.com/spring-cloud-stream-app-
starters/log/tree/v1.2.0.RC1",
      "packageHomeUrl" : "https://cloud.spring.io/spring-cloud-stream-app-starters/",
      "tags" : "logging, sink",
      "maintainer" : "https://github.com/sobychacko",
      "description" : "The log sink uses the application logger to output the data for
inspection.",
      "sha256" : null,
      "iconUrl" : null
    },
    "templates" : [ {
      "name" : "log.yml",
      "data" : "apiVersion: skipper.spring.io/v1\nkind:
SpringCloudDeployerApplication\nmetadata:\n  name: log\n  type: sink\nspec:\n
resource: maven://org.springframework.cloud.stream.app:log-sink-rabbit\n
resourceMetadata: maven://org.springframework.cloud.stream.app:log-sink-
```

```
rabbit:jar:metadata:{{version}}\n  version: {{version}}\n  applicationProperties:\n
server.port: 0\n    {{#spec.applicationProperties.entrySet}}\n    {{key}}: {{value}}\n
{{/spec.applicationProperties.entrySet}}\n  deploymentProperties:\n
{{#spec.deploymentProperties.entrySet}}\n    {{key}}: {{value}}\n
{{/spec.deploymentProperties.entrySet}}\n"
    } ],
    "dependencies" : [ ],
    "configValues" : {
      "raw" : "# Default values for {{name}}\n# This is a YAML-formatted file.\n#
Declare variables to be passed into your templates\nversion: 1.2.0.RC1\n"
    },
    "fileHolders" : [ ]
  },
  "configValues" : {
    "raw" : "config2: value2\nconfig1: value1\n"
  },
  "manifest" : {
    "data" : "\"apiVersion\": \"skipper.spring.io/v1\"\n\"kind\":
\"SpringCloudDeployerApplication\"\n\"metadata\":\n  \"name\": \"log\"\n  \"type\":
\"sink\"\n\"spec\":\n  \"resource\":
\"maven://org.springframework.cloud.stream.app:log-sink-rabbit\"\n
\"resourceMetadata\": \"maven://org.springframework.cloud.stream.app:log-sink-
rabbit:jar:metadata:1.2.0.RC1\"\n  \"version\": \"1.2.0.RC1\"\n
\"applicationProperties\":\n    \"server.port\": \"0\"\n  \"deploymentProperties\":
!!null \"null\"\n"
  },
  "platformName" : "default",
  "links" : [ {
    "rel" : "status",
    "href" : "http://localhost:7577/api/release/status/{name}"
  } ]
}
```

Response fields

Path	Type	Description
name	String	Name of the release
version	Number	Version of the release
info.status.statusCode	String	StatusCode of the release's status (UNKNOWN,DEPLOYED,DELETED,FAILED)
info.status.platformStatus	Null	Status from the underlying platform
info.firstDeployed	Null	Date/Time of first deployment
info.lastDeployed	Null	Date/Time of last deployment
info.deleted	Null	Date/Time of when the release was deleted

Path	Type	Description
info.description	Null	Human-friendly 'log entry' about this release
pkg.metadata.origin	Null	Indicates the origin of the repository (free form text)
pkg.metadata.apiVersion	String	The Package Index spec version this file is based on
pkg.metadata.repositoryId	Null	The repository ID this Package belongs to
pkg.metadata.repositoryName	Null	The repository name this Package belongs to.
pkg.metadata.kind	String	What type of package system is being used
pkg.metadata.name	String	The name of the package
pkg.metadata.displayName	Null	Display name of the release
pkg.metadata.version	String	The version of the package
pkg.metadata.packageSourceUrl	String	Location to source code for this package
pkg.metadata.packageHomeUrl	String	The home page of the package
pkg.metadata.tags	String	A comma separated list of tags to use for searching
pkg.metadata.maintainer	String	Who is maintaining this package
pkg.metadata.description	String	Brief description of the package
pkg.metadata.sha256	Null	Hash of package binary that will be downloaded using SHA256 hash algorithm
pkg.metadata.iconUrl	Null	Url location of a icon
pkg.templates[].name	String	Name is the path-like name of the template
pkg.templates[].data	String	Data is the template as string data
pkg.dependencies	Array	The packages that this package depends upon
pkg.configValues.raw	String	The raw YAML string of configuration values
pkg.fileHolders	Array	Miscellaneous files in a package, e.g. README, LICENSE, etc.
configValues.raw	String	The raw YAML string of configuration values
manifest.data	String	The manifest of the release

Path	Type	Description
`platformName`	`String`	Platform name of the release

30.5.3. Install with ID

The `install` link can install a package identified by its ID into the target platform.

Request structure

```
POST /api/package/install/1 HTTP/1.1
Content-Type: application/json;charset=UTF-8
Accept: application/json
Host: localhost:7577
```

Example request

```
$ curl 'http://localhost:7577/api/package/install/1' -i -X POST \
    -H 'Content-Type: application/json;charset=UTF-8' \
    -H 'Accept: application/json'
```

Response structure

```
HTTP/1.1 201 Created
Content-Type: application/json
Content-Length: 2633

{
  "name" : "myLogRelease2",
  "version" : 1,
  "info" : {
    "status" : {
      "statusCode" : "DELETED",
      "platformStatus" : null
    },
    "firstDeployed" : null,
    "lastDeployed" : null,
    "deleted" : null,
    "description" : null
  },
  "pkg" : {
    "metadata" : {
      "apiVersion" : "skipper.spring.io/v1",
      "origin" : null,
      "repositoryId" : null,
      "repositoryName" : null,
      "kind" : "SkipperPackageMetadata",
      "name" : "log",
```

```
      "displayName" : null,
      "version" : "1.0.0",
      "packageSourceUrl" : "https://github.com/spring-cloud-stream-app-
starters/log/tree/v1.2.0.RC1",
      "packageHomeUrl" : "https://cloud.spring.io/spring-cloud-stream-app-starters/",
      "tags" : "logging, sink",
      "maintainer" : "https://github.com/sobychacko",
      "description" : "The log sink uses the application logger to output the data for
inspection.",
      "sha256" : null,
      "iconUrl" : null
    },
    "templates" : [ {
      "name" : "log.yml",
      "data" : "apiVersion: skipper.spring.io/v1\nkind:
SpringCloudDeployerApplication\nmetadata:\n  name: log\n  type: sink\nspec:\n
resource: maven://org.springframework.cloud.stream.app:log-sink-rabbit\n
resourceMetadata: maven://org.springframework.cloud.stream.app:log-sink-
rabbit:jar:metadata:{{version}}\n  version: {{version}}\n  applicationProperties:\n
server.port: 0\n    {{#spec.applicationProperties.entrySet}}\n    {{key}}: {{value}}\n
{{/spec.applicationProperties.entrySet}}\n  deploymentProperties:\n
{{#spec.deploymentProperties.entrySet}}\n    {{key}}: {{value}}\n
{{/spec.deploymentProperties.entrySet}}\n"
    } ],
    "dependencies" : [ ],
    "configValues" : {
      "raw" : "# Default values for {{name}}\n# This is a YAML-formatted file.\n#
Declare variables to be passed into your templates\nversion: 1.2.0.RC1\n"
    },
    "fileHolders" : [ ]
  },
  "configValues" : {
    "raw" : "config2: value2\nconfig1: value1\n"
  },
  "manifest" : {
    "data" : "\"apiVersion\": \"skipper.spring.io/v1\"\n\"kind\":
\"SpringCloudDeployerApplication\"\n\"metadata\":\n  \"name\": \"log\"\n  \"type\":
\"sink\"\n\"spec\":\n  \"resource\":
\"maven://org.springframework.cloud.stream.app:log-sink-rabbit\"\n
\"resourceMetadata\": \"maven://org.springframework.cloud.stream.app:log-sink-
rabbit:jar:metadata:1.2.0.RC1\"\n  \"version\": \"1.2.0.RC1\"\n
\"applicationProperties\":\n    \"server.port\": \"0\"\n  \"deploymentProperties\":
!!null \"null\"\n"
  },
  "platformName" : "default",
  "links" : [ {
    "rel" : "status",
    "href" : "http://localhost:7577/api/release/status/{name}"
  } ]
}
```

Response fields

Path	Type	Description
name	String	Name of the release
version	Number	Version of the release
info.status.statusCode	String	StatusCode of the release's status (UNKNOWN,DEPLOYED,DELETED,FAILED)
info.status.platformStatus	Null	Status from the underlying platform
info.firstDeployed	Null	Date/Time of first deployment
info.lastDeployed	Null	Date/Time of last deployment
info.deleted	Null	Date/Time of when the release was deleted
info.description	Null	Human-friendly 'log entry' about this release
pkg.metadata.apiVersion	String	The Package Index spec version this file is based on
pkg.metadata.origin	Null	Indicates the origin of the repository (free form text)
pkg.metadata.repositoryId	Null	The repository ID this Package belongs to
pkg.metadata.repositoryName	Null	The repository name this Package belongs to.
pkg.metadata.kind	String	What type of package system is being used
pkg.metadata.name	String	The name of the package
pkg.metadata.displayName	Null	Display name of the release
pkg.metadata.version	String	The version of the package
pkg.metadata.packageSourceUrl	String	Location to source code for this package
pkg.metadata.packageHomeUrl	String	The home page of the package
pkg.metadata.tags	String	A comma separated list of tags to use for searching
pkg.metadata.maintainer	String	Who is maintaining this package
pkg.metadata.description	String	Brief description of the package
pkg.metadata.sha256	Null	Hash of package binary that will be downloaded using SHA256 hash algorithm
pkg.metadata.iconUrl	Null	Url location of a icon

Path	Type	Description
pkg.templates[].name	String	Name is the path-like name of the template
pkg.templates[].data	String	Data is the template as string data
pkg.dependencies	Array	The packages that this package depends upon
pkg.configValues.raw	String	The raw YAML string of configuration values
pkg.fileHolders	Array	Miscellaneous files in a package, e.g. README, LICENSE, etc.
configValues.raw	String	The raw YAML string of configuration values
manifest.data	String	The manifest of the release
platformName	String	Platform name of the release

30.6. Repositories

The Repositories resource is exported from the Spring Data Repository `RepositoryRepository` (yes, it's a funny name) and exposed by Spring Data REST.

30.6.1. Find All

A `GET` request returns a paginated list for all Spring Cloud Skipper repositories.

Request structure

```
GET /api/repositories?page=0&size=10 HTTP/1.1
Host: localhost:7577
```

Example request

```
$ curl 'http://localhost:7577/api/repositories?page=0&size=10' -i
```

Response structure

```
HTTP/1.1 200 OK
Vary: Origin
Vary: Access-Control-Request-Method
Vary: Access-Control-Request-Headers
Content-Type: application/hal+json
Content-Length: 1321
```

```
{
  "_embedded" : {
    "repositories" : [ {
      "name" : "test",
      "url" : "classpath:/repositories/binaries/test",
      "sourceUrl" : null,
      "local" : false,
      "description" : "test repository with a few packages",
      "repoOrder" : null,
      "_links" : {
        "self" : {
          "href" : "http://localhost:7577/api/repositories/1"
        },
        "repository" : {
          "href" : "http://localhost:7577/api/repositories/1"
        }
      }
    }, {
      "name" : "local",
      "url" : "http://localhost:7577",
      "sourceUrl" : null,
      "local" : true,
      "description" : "Default local database backed repository",
      "repoOrder" : null,
      "_links" : {
        "self" : {
          "href" : "http://localhost:7577/api/repositories/2"
        },
        "repository" : {
          "href" : "http://localhost:7577/api/repositories/2"
        }
      }
    } ]
  },
  "_links" : {
    "self" : {
      "href" : "http://localhost:7577/api/repositories{&sort}",
      "templated" : true
    },
    "profile" : {
      "href" : "http://localhost:7577/api/profile/repositories"
    },
    "search" : {
      "href" : "http://localhost:7577/api/repositories/search"
    }
  },
  "page" : {
    "size" : 10,
    "totalElements" : 2,
    "totalPages" : 1,
    "number" : 0
```

```
  }
}
```

Response fields

Path	Type	Description
page	Object	Pagination properties
page.size	Number	The size of the page being returned
page.totalElements	Number	Total elements available for pagination
page.totalPages	Number	Total amount of pages
page.number	Number	Page number of the page returned (zero-based)
_embedded.repositories	Array	Contains a collection of Repositories
_embedded.repositories[].name	String	Name of the Repository
_embedded.repositories[].url	String	Url of the Repository
_embedded.repositories[].sourceUrl	Null	Source Url of the repository
_embedded.repositories[].description	String	Description of the Repository
_embedded.repositories[].local	Boolean	Is the repo local?
_embedded.repositories[].repoOrder	Null	Order of the Repository

30.6.2. Find By Name

A GET request returns a single Spring Cloud Skipper repositories.

Request structure

```
GET /api/repositories/search/findByName?name=local HTTP/1.1
Host: localhost:7577
```

Example request

```
$ curl 'http://localhost:7577/api/repositories/search/findByName?name=local' -i
```

Response structure

```
HTTP/1.1 200 OK
Vary: Origin
Vary: Access-Control-Request-Method
Vary: Access-Control-Request-Headers
ETag: "0"
Content-Type: application/hal+json
Content-Length: 366

{
  "name" : "local",
  "url" : "http://localhost:7577",
  "sourceUrl" : null,
  "local" : true,
  "description" : "Default local database backed repository",
  "repoOrder" : null,
  "_links" : {
    "self" : {
      "href" : "http://localhost:7577/api/repositories/2"
    },
    "repository" : {
      "href" : "http://localhost:7577/api/repositories/2"
    }
  }
}
```

Response fields

Path	Type	Description
`name`	`String`	Name of the Repository
`url`	`String`	URL of the Repository
`description`	`String`	Description of the Repository
`local`	`Boolean`	Is the repo local?
`repoOrder`	`Null`	Order of the Repository
`sourceUrl`	`Null`	Source URL of the repository

30.7. Releases

The `release` resource is exported from the Spring Data Repository `ReleaseRepository` and exposed by Spring Data REST.

30.7.1. Find all

A `GET` request returns a paginated list for all Spring Cloud Skipper releases.

Request structure

```
GET /api/releases?page=0&size=10 HTTP/1.1
Host: localhost:7577
```

Example request

```
$ curl 'http://localhost:7577/api/releases?page=0&size=10' -i
```

Response structure

```
HTTP/1.1 200 OK
Vary: Origin
Vary: Access-Control-Request-Method
Vary: Access-Control-Request-Headers
Content-Type: application/hal+json
Content-Length: 3354

{
  "_embedded" : {
    "releases" : [ {
      "name" : "test",
      "version" : 1,
      "info" : {
        "status" : {
          "statusCode" : "DELETED",
          "platformStatus" : null
        },
        "firstDeployed" : null,
        "lastDeployed" : null,
        "deleted" : null,
        "description" : null
      },
      "pkg" : {
        "metadata" : {
          "apiVersion" : "skipper.spring.io/v1",
          "origin" : null,
          "repositoryId" : null,
          "repositoryName" : null,
          "kind" : "SkipperPackageMetadata",
          "name" : "log",
          "displayName" : null,
          "version" : "1.0.0",
          "packageSourceUrl" : "https://github.com/spring-cloud-stream-app-
starters/log/tree/v1.2.0.RC1",
          "packageHomeUrl" : "https://cloud.spring.io/spring-cloud-stream-app-
starters/",
          "tags" : "logging, sink",
```

```
          "maintainer" : "https://github.com/sobychacko",
          "description" : "The log sink uses the application logger to output the data
for inspection.",
          "sha256" : null,
          "iconUrl" : null
        },
        "templates" : [ {
          "name" : "log.yml",
          "data" : "apiVersion: skipper.spring.io/v1\nkind:
SpringCloudDeployerApplication\nmetadata:\n  name: log\n  type: sink\nspec:\n
resource: maven://org.springframework.cloud.stream.app:log-sink-rabbit\n
resourceMetadata: maven://org.springframework.cloud.stream.app:log-sink-
rabbit:jar:metadata:{{version}}\n  version: {{version}}\n  applicationProperties:\n
server.port: 0\n    {{#spec.applicationProperties.entrySet}}\n    {{key}}: {{value}}\n
{{/spec.applicationProperties.entrySet}}\n  deploymentProperties:\n
{{#spec.deploymentProperties.entrySet}}\n    {{key}}: {{value}}\n
{{/spec.deploymentProperties.entrySet}}\n"
        } ],
        "dependencies" : [ ],
        "configValues" : {
          "raw" : "# Default values for {{name}}\n# This is a YAML-formatted file.\n#
Declare variables to be passed into your templates\nversion: 1.2.0.RC1\n"
        },
        "fileHolders" : [ ]
      },
      "configValues" : {
        "raw" : "config2: value2\nconfig1: value1\n"
      },
      "manifest" : {
        "data" : "\"apiVersion\": \"skipper.spring.io/v1\"\n\"kind\":
\"SpringCloudDeployerApplication\"\n\"metadata\":\n  \"name\": \"log\"\n  \"type\":
\"sink\"\n\"spec\":\n  \"resource\":
\"maven://org.springframework.cloud.stream.app:log-sink-rabbit\"\n
\"resourceMetadata\": \"maven://org.springframework.cloud.stream.app:log-sink-
rabbit:jar:metadata:1.2.0.RC1\"\n  \"version\": \"1.2.0.RC1\"\n
\"applicationProperties\":\n    \"server.port\": \"0\"\n  \"deploymentProperties\":
!!null \"null\"\n"
      },
      "platformName" : "default",
      "_links" : {
        "self" : {
          "href" : "http://localhost:7577/api/releases/8"
        },
        "release" : {
          "href" : "http://localhost:7577/api/releases/8"
        }
      }
    } ]
  },
  "_links" : {
    "self" : {
```

```
      "href" : "http://localhost:7577/api/releases{&sort}",
      "templated" : true
    },
    "profile" : {
      "href" : "http://localhost:7577/api/profile/releases"
    },
    "search" : {
      "href" : "http://localhost:7577/api/releases/search"
    }
  },
  "page" : {
    "size" : 10,
    "totalElements" : 1,
    "totalPages" : 1,
    "number" : 0
  }
}
```

Response fields

Path	Type	Description
page	Object	Pagination properties
page.size	Number	The size of the page being returned
page.totalElements	Number	Total elements available for pagination
page.totalPages	Number	Total amount of pages
page.number	Number	Page number of the page returned (zero-based)
_embedded.releases[].name	String	Name of the release
_embedded.releases[].version	Number	Version of the release
_embedded.releases[].info.status.statusCode	String	StatusCode of the release's status (UNKNOWN,DEPLOYED,DELETED,FAILED)
_embedded.releases[].info.status.platformStatus	Null	Status from the underlying platform
_embedded.releases[].info.firstDeployed	Null	Date/Time of first deployment
_embedded.releases[].info.lastDeployed	Null	Date/Time of last deployment
_embedded.releases[].info.deleted	Null	Date/Time of when the release was deleted
_embedded.releases[].info.description	Null	Human-friendly 'log entry' about this release

Path	Type	Description
_embedded.releases[].platformName	String	Platform name of the release
_embedded.releases[].pkg.metadata.apiVersion	String	The Package Index spec version this file is based on
_embedded.releases[].pkg.metadata.origin	Null	Indicates the origin of the repository (free form text)
_embedded.releases[].pkg.metadata.repositoryId	Null	The repository ID this Package belongs to
_embedded.releases[].pkg.metadata.repositoryName	Null	The repository name this Package belongs to.
_embedded.releases[].pkg.metadata.kind	String	What type of package system is being used
_embedded.releases[].pkg.metadata.name	String	The name of the package
_embedded.releases[].pkg.metadata.displayName	Null	Display name of the release
_embedded.releases[].pkg.metadata.version	String	The version of the package
_embedded.releases[].pkg.metadata.packageSourceUrl	String	Location to source code for this package
_embedded.releases[].pkg.metadata.packageHomeUrl	String	The home page of the package
_embedded.releases[].pkg.metadata.tags	String	A comma separated list of tags to use for searching
_embedded.releases[].pkg.metadata.maintainer	String	Who is maintaining this package
_embedded.releases[].pkg.metadata.description	String	Brief description of the package
_embedded.releases[].pkg.metadata.sha256	Null	Hash of package binary that will be downloaded using SHA256 hash algorithm
_embedded.releases[].pkg.metadata.iconUrl	Null	Url location of a icon
_embedded.releases[].pkg.templates[].name	String	Name is the path-like name of the template
_embedded.releases[].pkg.templates[].data	String	Data is the template as string data
_embedded.releases[].pkg.dependencies	Array	The packages that this package depends upon
_embedded.releases[].pkg.configValues.raw	String	The raw YAML string of configuration values
_embedded.releases[].pkg.fileHolders	Array	Miscellaneous files in a package, e.g. README, LICENSE, etc.

Path	Type	Description
`_embedded.releases[].configValues.raw`	`String`	The raw YAML string of configuration values
`_embedded.releases[].manifest.data`	`String`	The manifest of the release
`_embedded.releases[].platformName`	`String`	Platform name of the release

30.8. Release

The Release resource maps onto the ReleaseController for managing the lifecycle of a release.

30.8.1. List

List latest

The `list` link can list the latest version of releases with status of deployed or failed.

Request structure

```
GET /api/release/list HTTP/1.1
Host: localhost:7577
```

Example request

```
$ curl 'http://localhost:7577/api/release/list' -i
```

Response structure

```
HTTP/1.1 200 OK
Content-Type: application/hal+json
Content-Length: 2913

{
  "_embedded" : {
    "releases" : [ {
      "name" : "test",
      "version" : 1,
      "info" : {
        "status" : {
          "statusCode" : "DELETED",
          "platformStatus" : null
        },
        "firstDeployed" : null,
        "lastDeployed" : null,
        "deleted" : null,
        "description" : null
```

```
      },
      "pkg" : {
        "metadata" : {
          "apiVersion" : "skipper.spring.io/v1",
          "origin" : null,
          "repositoryId" : null,
          "repositoryName" : null,
          "kind" : "SkipperPackageMetadata",
          "name" : "log",
          "displayName" : null,
          "version" : "1.0.0",
          "packageSourceUrl" : "https://github.com/spring-cloud-stream-app-
starters/log/tree/v1.2.0.RC1",
          "packageHomeUrl" : "https://cloud.spring.io/spring-cloud-stream-app-
starters/",
          "tags" : "logging, sink",
          "maintainer" : "https://github.com/sobychacko",
          "description" : "The log sink uses the application logger to output the data
for inspection.",
          "sha256" : null,
          "iconUrl" : null
        },
        "templates" : [ {
          "name" : "log.yml",
          "data" : "apiVersion: skipper.spring.io/v1\nkind:
SpringCloudDeployerApplication\nmetadata:\n  name: log\n  type: sink\nspec:\n
resource: maven://org.springframework.cloud.stream.app:log-sink-rabbit\n
resourceMetadata: maven://org.springframework.cloud.stream.app:log-sink-
rabbit:jar:metadata:{{version}}\n  version: {{version}}\n  applicationProperties:\n
server.port: 0\n    {{#spec.applicationProperties.entrySet}}\n    {{key}}: {{value}}\n
{{/spec.applicationProperties.entrySet}}\n  deploymentProperties:\n
{{#spec.deploymentProperties.entrySet}}\n    {{key}}: {{value}}\n
{{/spec.deploymentProperties.entrySet}}\n"
        } ],
        "dependencies" : [ ],
        "configValues" : {
          "raw" : "# Default values for {{name}}\n# This is a YAML-formatted file.\n#
Declare variables to be passed into your templates\nversion: 1.2.0.RC1\n"
        },
        "fileHolders" : [ ]
      },
      "configValues" : {
        "raw" : "config2: value2\nconfig1: value1\n"
      },
      "manifest" : {
        "data" : "\"apiVersion\": \"skipper.spring.io/v1\"\n\"kind\":
\"SpringCloudDeployerApplication\"\n\"metadata\":\n  \"name\": \"log\"\n  \"type\":
\"sink\"\n\"spec\":\n  \"resource\":
\"maven://org.springframework.cloud.stream.app:log-sink-rabbit\"\n
\"resourceMetadata\": \"maven://org.springframework.cloud.stream.app:log-sink-
rabbit:jar:metadata:1.2.0.RC1\"\n  \"version\": \"1.2.0.RC1\"\n
```

```
\"applicationProperties\":\n    \"server.port\": \"0\"\n  \"deploymentProperties\":
!!null \"null\"\n"
      },
      "platformName" : "default",
      "_links" : {
        "status" : {
          "href" : "http://localhost:7577/api/release/status/{name}",
          "templated" : true
        }
      }
    } ]
  }
}
```

Response fields

Path	Type	Description
_embedded.releases[].name	String	Name of the release
_embedded.releases[].version	Number	Version of the release
_embedded.releases[].info.status.statusCode	String	StatusCode of the release's status (UNKNOWN,DEPLOYED,DELETED,FAILED)
_embedded.releases[].info.status.platformStatus	Null	Status from the underlying platform
_embedded.releases[].info.firstDeployed	Null	Date/Time of first deployment
_embedded.releases[].info.lastDeployed	Null	Date/Time of last deployment
_embedded.releases[].info.deleted	Null	Date/Time of when the release was deleted
_embedded.releases[].info.description	Null	Human-friendly 'log entry' about this release
_embedded.releases[].pkg.metadata.apiVersion	String	The Package Index spec version this file is based on
_embedded.releases[].pkg.metadata.origin	Null	Indicates the origin of the repository (free form text)
_embedded.releases[].pkg.metadata.repositoryId	Null	The repository ID this Package belongs to
_embedded.releases[].pkg.metadata.repositoryName	Null	The repository name this Package belongs to.
_embedded.releases[].pkg.metadata.kind	String	What type of package system is being used
_embedded.releases[].pkg.metadata.name	String	The name of the package
_embedded.releases[].pkg.metadata.displayName	Null	Display name of the release

Path	Type	Description
`_embedded.releases[].pkg.metadata.version`	`String`	The version of the package
`_embedded.releases[].pkg.metadata.packageSourceUrl`	`String`	Location to source code for this package
`_embedded.releases[].pkg.metadata.packageHomeUrl`	`String`	The home page of the package
`_embedded.releases[].pkg.metadata.tags`	`String`	A comma separated list of tags to use for searching
`_embedded.releases[].pkg.metadata.maintainer`	`String`	Who is maintaining this package
`_embedded.releases[].pkg.metadata.description`	`String`	Brief description of the package
`_embedded.releases[].pkg.metadata.sha256`	`Null`	Hash of package binary that will be downloaded using SHA256 hash algorithm
`_embedded.releases[].pkg.metadata.iconUrl`	`Null`	Url location of a icon
`_embedded.releases[].pkg.templates[].name`	`String`	Name is the path-like name of the template
`_embedded.releases[].pkg.templates[].data`	`String`	Data is the template as string data
`_embedded.releases[].pkg.dependencies`	`Array`	The packages that this package depends upon
`_embedded.releases[].pkg.configValues.raw`	`String`	The raw YAML string of configuration values
`_embedded.releases[].pkg.fileHolders`	`Array`	Miscellaneous files in a package, e.g. README, LICENSE, etc.
`_embedded.releases[].configValues.raw`	`String`	The raw YAML string of configuration values
`_embedded.releases[].manifest.data`	`String`	The manifest of the release
`_embedded.releases[].platformName`	`String`	Platform name of the release

List latest by name

The `list` link can list the latest version of releases with status of deployed or failed by the given release name.

Request structure

```
GET /api/release/list/test HTTP/1.1
Host: localhost:7577
```

Example request

```
$ curl 'http://localhost:7577/api/release/list/test' -i
```

Response structure

```
HTTP/1.1 200 OK
Content-Type: application/hal+json
Content-Length: 2913

{
  "_embedded" : {
    "releases" : [ {
      "name" : "test",
      "version" : 1,
      "info" : {
        "status" : {
          "statusCode" : "DELETED",
          "platformStatus" : null
        },
        "firstDeployed" : null,
        "lastDeployed" : null,
        "deleted" : null,
        "description" : null
      },
      "pkg" : {
        "metadata" : {
          "apiVersion" : "skipper.spring.io/v1",
          "origin" : null,
          "repositoryId" : null,
          "repositoryName" : null,
          "kind" : "SkipperPackageMetadata",
          "name" : "log",
          "displayName" : null,
          "version" : "1.0.0",
          "packageSourceUrl" : "https://github.com/spring-cloud-stream-app-
starters/log/tree/v1.2.0.RC1",
          "packageHomeUrl" : "https://cloud.spring.io/spring-cloud-stream-app-
starters/",
          "tags" : "logging, sink",
          "maintainer" : "https://github.com/sobychacko",
          "description" : "The log sink uses the application logger to output the data
for inspection.",
          "sha256" : null,
          "iconUrl" : null
        },
        "templates" : [ {
          "name" : "log.yml",
          "data" : "apiVersion: skipper.spring.io/v1\nkind:
```

```
SpringCloudDeployerApplication\nmetadata:\n  name: log\n  type: sink\nspec:\n
resource: maven://org.springframework.cloud.stream.app:log-sink-rabbit\n
resourceMetadata: maven://org.springframework.cloud.stream.app:log-sink-
rabbit:jar:metadata:{{version}}\n  version: {{version}}\n  applicationProperties:\n
server.port: 0\n    {{#spec.applicationProperties.entrySet}}\n    {{key}}: {{value}}\n
{{/spec.applicationProperties.entrySet}}\n  deploymentProperties:\n
{{#spec.deploymentProperties.entrySet}}\n    {{key}}: {{value}}\n
{{/spec.deploymentProperties.entrySet}}\n"
        } ],
        "dependencies" : [ ],
        "configValues" : {
          "raw" : "# Default values for {{name}}\n# This is a YAML-formatted file.\n#
Declare variables to be passed into your templates\nversion: 1.2.0.RC1\n"
        },
        "fileHolders" : [ ]
      },
      "configValues" : {
        "raw" : "config2: value2\nconfig1: value1\n"
      },
      "manifest" : {
        "data" : "\"apiVersion\": \"skipper.spring.io/v1\"\n\"kind\":
\"SpringCloudDeployerApplication\"\n\"metadata\":\n  \"name\": \"log\"\n  \"type\":
\"sink\"\n\"spec\":\n  \"resource\":
\"maven://org.springframework.cloud.stream.app:log-sink-rabbit\"\n
\"resourceMetadata\": \"maven://org.springframework.cloud.stream.app:log-sink-
rabbit:jar:metadata:1.2.0.RC1\"\n  \"version\": \"1.2.0.RC1\"\n
\"applicationProperties\":\n    \"server.port\": \"0\"\n  \"deploymentProperties\":
!!null \"null\"\n"
      },
      "platformName" : "default",
      "_links" : {
        "status" : {
          "href" : "http://localhost:7577/api/release/status/{name}",
          "templated" : true
        }
      }
    } ]
  }
}
```

Response fields

Path	Type	Description
`_embedded.releases[].name`	`String`	Name of the release
`_embedded.releases[].version`	`Number`	Version of the release
`_embedded.releases[].info.status.statusCode`	`String`	StatusCode of the release's status (UNKNOWN,DEPLOYED,DELETED,FAILED)

Path	Type	Description
_embedded.releases[].info.status.platformStatus	Null	Status from the underlying platform
_embedded.releases[].info.firstDeployed	Null	Date/Time of first deployment
_embedded.releases[].info.lastDeployed	Null	Date/Time of last deployment
_embedded.releases[].info.deleted	Null	Date/Time of when the release was deleted
_embedded.releases[].info.description	Null	Human-friendly 'log entry' about this release
_embedded.releases[].pkg.metadata.apiVersion	String	The Package Index spec version this file is based on
_embedded.releases[].pkg.metadata.origin	Null	Indicates the origin of the repository (free form text)
_embedded.releases[].pkg.metadata.repositoryId	Null	The repository ID this Package belongs to
_embedded.releases[].pkg.metadata.repositoryName	Null	The repository name this Package belongs to.
_embedded.releases[].pkg.metadata.kind	String	What type of package system is being used
_embedded.releases[].pkg.metadata.name	String	The name of the package
_embedded.releases[].pkg.metadata.displayName	Null	Display name of the release
_embedded.releases[].pkg.metadata.version	String	The version of the package
_embedded.releases[].pkg.metadata.packageSourceUrl	String	Location to source code for this package
_embedded.releases[].pkg.metadata.packageHomeUrl	String	The home page of the package
_embedded.releases[].pkg.metadata.tags	String	A comma separated list of tags to use for searching
_embedded.releases[].pkg.metadata.maintainer	String	Who is maintaining this package
_embedded.releases[].pkg.metadata.description	String	Brief description of the package
_embedded.releases[].pkg.metadata.sha256	Null	Hash of package binary that will be downloaded using SHA256 hash algorithm
_embedded.releases[].pkg.metadata.iconUrl	Null	Url location of a icon
_embedded.releases[].pkg.templates[].name	String	Name is the path-like name of the template
_embedded.releases[].pkg.templates[].data	String	Data is the template as string data

Path	Type	Description
`_embedded.releases[].pkg.dependencies`	`Array`	The packages that this package depends upon
`_embedded.releases[].pkg.configValues.raw`	`String`	The raw YAML string of configuration values
`_embedded.releases[].pkg.fileHolders`	`Array`	Miscellaneous files in a package, e.g. README, LICENSE, etc.
`_embedded.releases[].configValues.raw`	`String`	The raw YAML string of configuration values
`_embedded.releases[].manifest.data`	`String`	The manifest of the release
`_embedded.releases[].platformName`	`String`	Platform name of the release

30.8.2. Status

Get the status of a release

The `status` REST endpoint provides the status for the last known release version.

Request structure

```
GET /api/release/status/test HTTP/1.1
Host: localhost:7577
```

Example request

```
$ curl 'http://localhost:7577/api/release/status/test' -i
```

Response structure

```
HTTP/1.1 200 OK
Content-Type: application/hal+json
Content-Length: 313

{
  "status" : {
    "statusCode" : "DELETED",
    "platformStatus" : null
  },
  "firstDeployed" : null,
  "lastDeployed" : null,
  "deleted" : null,
  "description" : null,
  "_links" : {
    "manifest" : {
      "href" : "http://localhost:7577/api/release/manifest/{name}",
      "templated" : true
    }
  }
}
```

Response fields

Path	Type	Description
`status.statusCode`	`String`	StatusCode of the release's status (UNKNOWN,DEPLOYED,DELETED,FAILED)
`status.platformStatus`	`Null`	Status from the underlying platform
`firstDeployed`	`Null`	Date/Time of first deployment
`lastDeployed`	`Null`	Date/Time of last deployment
`deleted`	`Null`	Date/Time of when the release was deleted
`description`	`Null`	Human-friendly 'log entry' about this release

Status by version

The `status` REST endpoint can provide the status for a specific release version.

Request structure

```
GET /api/release/status/test/1 HTTP/1.1
Host: localhost:7577
```

Example request

```
$ curl 'http://localhost:7577/api/release/status/test/1' -i
```

Response structure

```
HTTP/1.1 200 OK
Content-Type: application/hal+json
Content-Length: 313

{
  "status" : {
    "statusCode" : "DELETED",
    "platformStatus" : null
  },
  "firstDeployed" : null,
  "lastDeployed" : null,
  "deleted" : null,
  "description" : null,
  "_links" : {
    "manifest" : {
      "href" : "http://localhost:7577/api/release/manifest/{name}",
      "templated" : true
    }
  }
}
```

Response fields

Path	Type	Description
status.statusCode	String	StatusCode of the release's status (UNKNOWN,DEPLOYED,DELETED,FAILED)
status.platformStatus	Null	Status from the underlying platform
firstDeployed	Null	Date/Time of first deployment
lastDeployed	Null	Date/Time of last deployment
deleted	Null	Date/Time of when the release was deleted
description	Null	Human-friendly 'log entry' about this release

30.8.3. Upgrade

Upgrade a release

The upgrade link upgrades an existing release with the configured package and config values from the `UpgradeRequest`.

Request structure

```
POST /api/release/upgrade HTTP/1.1
Content-Type: application/json;charset=UTF-8
Accept: application/json
Host: localhost:7577
```

Example request

```
$ curl 'http://localhost:7577/api/release/upgrade' -i -X POST \
    -H 'Content-Type: application/json;charset=UTF-8' \
    -H 'Accept: application/json'
```

Response structure

```
HTTP/1.1 201 Created
Content-Type: application/json
Content-Length: 2624

{
  "name" : "test",
  "version" : 1,
  "info" : {
    "status" : {
      "statusCode" : "DELETED",
      "platformStatus" : null
    },
    "firstDeployed" : null,
    "lastDeployed" : null,
    "deleted" : null,
    "description" : null
  },
  "pkg" : {
    "metadata" : {
      "apiVersion" : "skipper.spring.io/v1",
      "origin" : null,
      "repositoryId" : null,
      "repositoryName" : null,
      "kind" : "SkipperPackageMetadata",
      "name" : "log",
      "displayName" : null,
      "version" : "1.0.0",
      "packageSourceUrl" : "https://github.com/spring-cloud-stream-app-
```

```
starters/log/tree/v1.2.0.RC1",
      "packageHomeUrl" : "https://cloud.spring.io/spring-cloud-stream-app-starters/",
      "tags" : "logging, sink",
      "maintainer" : "https://github.com/sobychacko",
      "description" : "The log sink uses the application logger to output the data for
inspection.",
      "sha256" : null,
      "iconUrl" : null
    },
    "templates" : [ {
      "name" : "log.yml",
      "data" : "apiVersion: skipper.spring.io/v1\nkind:
SpringCloudDeployerApplication\nmetadata:\n  name: log\n  type: sink\nspec:\n
resource: maven://org.springframework.cloud.stream.app:log-sink-rabbit\n
resourceMetadata: maven://org.springframework.cloud.stream.app:log-sink-
rabbit:jar:metadata:{{version}}\n  version: {{version}}\n  applicationProperties:\n
server.port: 0\n    {{#spec.applicationProperties.entrySet}}\n    {{key}}: {{value}}\n
{{/spec.applicationProperties.entrySet}}\n  deploymentProperties:\n
{{#spec.deploymentProperties.entrySet}}\n    {{key}}: {{value}}\n
{{/spec.deploymentProperties.entrySet}}\n"
    } ],
    "dependencies" : [ ],
    "configValues" : {
      "raw" : "# Default values for {{name}}\n# This is a YAML-formatted file.\n#
Declare variables to be passed into your templates\nversion: 1.2.0.RC1\n"
    },
    "fileHolders" : [ ]
  },
  "configValues" : {
    "raw" : "config2: value2\nconfig1: value1\n"
  },
  "manifest" : {
    "data" : "\"apiVersion\": \"skipper.spring.io/v1\"\n\"kind\":
\"SpringCloudDeployerApplication\"\n\"metadata\":\n  \"name\": \"log\"\n  \"type\":
\"sink\"\n\"spec\":\n  \"resource\":
\"maven://org.springframework.cloud.stream.app:log-sink-rabbit\"\n
\"resourceMetadata\": \"maven://org.springframework.cloud.stream.app:log-sink-
rabbit:jar:metadata:1.2.0.RC1\"\n  \"version\": \"1.2.0.RC1\"\n
\"applicationProperties\":\n    \"server.port\": \"0\"\n  \"deploymentProperties\":
!!null \"null\"\n"
  },
  "platformName" : "default",
  "links" : [ {
    "rel" : "status",
    "href" : "http://localhost:7577/api/release/status/{name}"
  } ]
}
```

Response fields

Path	Type	Description
name	String	Name of the release
version	Number	Version of the release
info.status.statusCode	String	StatusCode of the release's status (UNKNOWN,DEPLOYED,DELETED,FAILED)
info.status.platformStatus	Null	Status from the underlying platform
info.firstDeployed	Null	Date/Time of first deployment
info.lastDeployed	Null	Date/Time of last deployment
info.deleted	Null	Date/Time of when the release was deleted
info.description	Null	Human-friendly 'log entry' about this release
pkg.metadata.apiVersion	String	The Package Index spec version this file is based on
pkg.metadata.origin	Null	Indicates the origin of the repository (free form text)
pkg.metadata.repositoryId	Null	The repository ID this Package belongs to.
pkg.metadata.repositoryName	Null	The repository name this Package belongs to.
pkg.metadata.kind	String	What type of package system is being used
pkg.metadata.name	String	The name of the package
pkg.metadata.displayName	Null	Display name of the release
pkg.metadata.version	String	The version of the package
pkg.metadata.packageSourceUrl	String	Location to source code for this package
pkg.metadata.packageHomeUrl	String	The home page of the package
pkg.metadata.tags	String	A comma separated list of tags to use for searching
pkg.metadata.maintainer	String	Who is maintaining this package
pkg.metadata.description	String	Brief description of the package
pkg.metadata.sha256	Null	Hash of package binary that will be downloaded using SHA256 hash algorithm
pkg.metadata.iconUrl	Null	Url location of a icon

Path	Type	Description
pkg.templates[].name	String	Name is the path-like name of the template
pkg.templates[].data	String	Data is the template as string data
pkg.dependencies	Array	The packages that this package depends upon
pkg.configValues.raw	String	The raw YAML string of configuration values
pkg.fileHolders	Array	Miscellaneous files in a package, e.g. README, LICENSE, etc.
configValues.raw	String	The raw YAML string of configuration values
manifest.data	String	The manifest of the release
platformName	String	Platform name of the release

30.8.4. Rollback

Rollback release using uri variables

The rollback link rolls back the release to a previous or a specific release.

This part of the api is deprecated, please use Rollback release using request object.

Request structure

```
POST /api/release/rollback/test/1 HTTP/1.1
Host: localhost:7577
```

Example request

```
$ curl 'http://localhost:7577/api/release/rollback/test/1' -i -X POST
```

Response structure

```
HTTP/1.1 201 Created
Content-Type: application/hal+json
Content-Length: 2650

{
  "name" : "test",
  "version" : 1,
  "info" : {
```

```
    "status" : {
      "statusCode" : "DELETED",
      "platformStatus" : null
    },
    "firstDeployed" : null,
    "lastDeployed" : null,
    "deleted" : null,
    "description" : null
  },
  "pkg" : {
    "metadata" : {
      "apiVersion" : "skipper.spring.io/v1",
      "origin" : null,
      "repositoryId" : null,
      "repositoryName" : null,
      "kind" : "SkipperPackageMetadata",
      "name" : "log",
      "displayName" : null,
      "version" : "1.0.0",
      "packageSourceUrl" : "https://github.com/spring-cloud-stream-app-
starters/log/tree/v1.2.0.RC1",
      "packageHomeUrl" : "https://cloud.spring.io/spring-cloud-stream-app-starters/",
      "tags" : "logging, sink",
      "maintainer" : "https://github.com/sobychacko",
      "description" : "The log sink uses the application logger to output the data for
inspection.",
      "sha256" : null,
      "iconUrl" : null
    },
    "templates" : [ {
      "name" : "log.yml",
      "data" : "apiVersion: skipper.spring.io/v1\nkind:
SpringCloudDeployerApplication\nmetadata:\n  name: log\n  type: sink\nspec:\n
resource: maven://org.springframework.cloud.stream.app:log-sink-rabbit\n
resourceMetadata: maven://org.springframework.cloud.stream.app:log-sink-
rabbit:jar:metadata:{{version}}\n  version: {{version}}\n  applicationProperties:\n
server.port: 0\n    {{#spec.applicationProperties.entrySet}}\n    {{key}}: {{value}}\n
{{/spec.applicationProperties.entrySet}}\n  deploymentProperties:\n
{{#spec.deploymentProperties.entrySet}}\n    {{key}}: {{value}}\n
{{/spec.deploymentProperties.entrySet}}\n"
    } ],
    "dependencies" : [ ],
    "configValues" : {
      "raw" : "# Default values for {{name}}\n# This is a YAML-formatted file.\n#
Declare variables to be passed into your templates\nversion: 1.2.0.RC1\n"
    },
    "fileHolders" : [ ]
  },
  "configValues" : {
    "raw" : "config2: value2\nconfig1: value1\n"
  },
```

```
  "manifest" : {
    "data" : "\"apiVersion\": \"skipper.spring.io/v1\"\n\"kind\":
\"SpringCloudDeployerApplication\"\n\"metadata\":\n  \"name\": \"log\"\n  \"type\":
\"sink\"\n\"spec\":\n  \"resource\":
\"maven://org.springframework.cloud.stream.app:log-sink-rabbit\"\n
\"resourceMetadata\": \"maven://org.springframework.cloud.stream.app:log-sink-
rabbit:jar:metadata:1.2.0.RC1\"\n  \"version\": \"1.2.0.RC1\"\n
\"applicationProperties\":\n    \"server.port\": \"0\"\n  \"deploymentProperties\":
!!null \"null\"\n"
  },
  "platformName" : "default",
  "_links" : {
    "status" : {
      "href" : "http://localhost:7577/api/release/status/{name}",
      "templated" : true
    }
  }
}
```

Response fields

Path	Type	Description
name	String	Name of the release
version	Number	Version of the release
info.status.statusCode	String	StatusCode of the release's status (UNKNOWN,DEPLOYED,DELETED,FAILED)
info.status.platformStatus	Null	Status from the underlying platform
info.firstDeployed	Null	Date/Time of first deployment
info.lastDeployed	Null	Date/Time of last deployment
info.deleted	Null	Date/Time of when the release was deleted
info.description	Null	Human-friendly 'log entry' about this release
pkg.metadata.apiVersion	String	The Package Index spec version this file is based on
pkg.metadata.origin	Null	Indicates the origin of the repository (free form text)
pkg.metadata.repositoryId	Null	The repository ID this Package belongs to.
pkg.metadata.repositoryName	Null	The repository name this Package belongs to.
pkg.metadata.kind	String	What type of package system is being used

Path	Type	Description
pkg.metadata.name	String	The name of the package
pkg.metadata.displayName	Null	Display name of the release
pkg.metadata.version	String	The version of the package
pkg.metadata.packageSourceUrl	String	Location to source code for this package
pkg.metadata.packageHomeUrl	String	The home page of the package
pkg.metadata.tags	String	A comma separated list of tags to use for searching
pkg.metadata.maintainer	String	Who is maintaining this package
pkg.metadata.description	String	Brief description of the package
pkg.metadata.sha256	Null	Hash of package binary that will be downloaded using SHA256 hash algorithm
pkg.metadata.iconUrl	Null	Url location of a icon
pkg.templates[].name	String	Name is the path-like name of the template
pkg.templates[].data	String	Data is the template as string data
pkg.dependencies	Array	The packages that this package depends upon
pkg.configValues.raw	String	The raw YAML string of configuration values
pkg.fileHolders	Array	Miscellaneous files in a package, e.g. README, LICENSE, etc.
configValues.raw	String	The raw YAML string of configuration values
manifest.data	String	The manifest of the release
platformName	String	Platform name of the release

Rollback release using request object

The rollback link rolls back the release to a previous or a specific release.

Request structure

```
POST /api/release/rollback HTTP/1.1
Content-Type: application/json;charset=UTF-8
Accept: application/json
Host: localhost:7577
```

Example request

```
$ curl 'http://localhost:7577/api/release/rollback' -i -X POST \
    -H 'Content-Type: application/json;charset=UTF-8' \
    -H 'Accept: application/json'
```

Response structure

```
HTTP/1.1 201 Created
Content-Type: application/json
Content-Length: 2624

{
  "name" : "test",
  "version" : 1,
  "info" : {
    "status" : {
      "statusCode" : "DELETED",
      "platformStatus" : null
    },
    "firstDeployed" : null,
    "lastDeployed" : null,
    "deleted" : null,
    "description" : null
  },
  "pkg" : {
    "metadata" : {
      "apiVersion" : "skipper.spring.io/v1",
      "origin" : null,
      "repositoryId" : null,
      "repositoryName" : null,
      "kind" : "SkipperPackageMetadata",
      "name" : "log",
      "displayName" : null,
      "version" : "1.0.0",
      "packageSourceUrl" : "https://github.com/spring-cloud-stream-app-
starters/log/tree/v1.2.0.RC1",
      "packageHomeUrl" : "https://cloud.spring.io/spring-cloud-stream-app-starters/",
      "tags" : "logging, sink",
      "maintainer" : "https://github.com/sobychacko",
      "description" : "The log sink uses the application logger to output the data for
inspection.",
      "sha256" : null,
      "iconUrl" : null
    },
    "templates" : [ {
      "name" : "log.yml",
      "data" : "apiVersion: skipper.spring.io/v1\nkind:
SpringCloudDeployerApplication\nmetadata:\n  name: log\n  type: sink\nspec:\n
```

```
resource: maven://org.springframework.cloud.stream.app:log-sink-rabbit\n
resourceMetadata: maven://org.springframework.cloud.stream.app:log-sink-
rabbit:jar:metadata:{{version}}\n  version: {{version}}\n  applicationProperties:\n
server.port: 0\n    {{#spec.applicationProperties.entrySet}}\n    {{key}}: {{value}}\n
{{/spec.applicationProperties.entrySet}}\n  deploymentProperties:\n
{{#spec.deploymentProperties.entrySet}}\n    {{key}}: {{value}}\n
{{/spec.deploymentProperties.entrySet}}\n"
    } ],
    "dependencies" : [ ],
    "configValues" : {
      "raw" : "# Default values for {{name}}\n# This is a YAML-formatted file.\n#
Declare variables to be passed into your templates\nversion: 1.2.0.RC1\n"
    },
    "fileHolders" : [ ]
  },
  "configValues" : {
    "raw" : "config2: value2\nconfig1: value1\n"
  },
  "manifest" : {
    "data" : "\"apiVersion\": \"skipper.spring.io/v1\"\n\"kind\":
\"SpringCloudDeployerApplication\"\n\"metadata\":\n  \"name\": \"log\"\n  \"type\":
\"sink\"\n\"spec\":\n  \"resource\":
\"maven://org.springframework.cloud.stream.app:log-sink-rabbit\"\n
\"resourceMetadata\": \"maven://org.springframework.cloud.stream.app:log-sink-
rabbit:jar:metadata:1.2.0.RC1\"\n  \"version\": \"1.2.0.RC1\"\n
\"applicationProperties\":\n    \"server.port\": \"0\"\n  \"deploymentProperties\":
!!null \"null\"\n"
  },
  "platformName" : "default",
  "links" : [ {
    "rel" : "status",
    "href" : "http://localhost:7577/api/release/status/{name}"
  } ]
}
```


Response fields

Path	Type	Description
`name`	`String`	Name of the release
`version`	`Number`	Version of the release
`info.status.statusCode`	`String`	StatusCode of the release's status (UNKNOWN,DEPLOYED,DELETED,FAILED)
`info.status.platformStatus`	`Null`	Status from the underlying platform
`info.firstDeployed`	`Null`	Date/Time of first deployment
`info.lastDeployed`	`Null`	Date/Time of last deployment

Path	Type	Description
info.deleted	Null	Date/Time of when the release was deleted
info.description	Null	Human-friendly 'log entry' about this release
pkg.metadata.apiVersion	String	The Package Index spec version this file is based on
pkg.metadata.origin	Null	Indicates the origin of the repository (free form text)
pkg.metadata.repositoryId	Null	The repository ID this Package belongs to.
pkg.metadata.repositoryName	Null	The repository name this Package belongs to.
pkg.metadata.kind	String	What type of package system is being used
pkg.metadata.name	String	The name of the package
pkg.metadata.displayName	Null	Display name of the release
pkg.metadata.version	String	The version of the package
pkg.metadata.packageSourceUrl	String	Location to source code for this package
pkg.metadata.packageHomeUrl	String	The home page of the package
pkg.metadata.tags	String	A comma separated list of tags to use for searching
pkg.metadata.maintainer	String	Who is maintaining this package
pkg.metadata.description	String	Brief description of the package
pkg.metadata.sha256	Null	Hash of package binary that will be downloaded using SHA256 hash algorithm
pkg.metadata.iconUrl	Null	Url location of a icon
pkg.templates[].name	String	Name is the path-like name of the template
pkg.templates[].data	String	Data is the template as string data
pkg.dependencies	Array	The packages that this package depends upon
pkg.configValues.raw	String	The raw YAML string of configuration values
pkg.fileHolders	Array	Miscellaneous files in a package, e.g. README, LICENSE, etc.

Path	Type	Description
`configValues.raw`	`String`	The raw YAML string of configuration values
`manifest.data`	`String`	The manifest of the release
`platformName`	`String`	Platform name of the release

30.8.5. Manifest

Get manifest

The `manifest` REST endpoint returns the manifest for the last known release version.

Request structure

```
GET /api/release/manifest/test HTTP/1.1
Content-Type: application/json;charset=UTF-8
Accept: application/json
Host: localhost:7577
```

Example request

```
$ curl 'http://localhost:7577/api/release/manifest/test' -i \
    -H 'Content-Type: application/json;charset=UTF-8' \
    -H 'Accept: application/json'
```

Response structure

```
HTTP/1.1 200 OK
Content-Type: application/json
Content-Length: 610

{
  "data" : "\"apiVersion\": \"skipper.spring.io/v1\"\n\"kind\":
\"SpringCloudDeployerApplication\"\n\"metadata\":\n  \"name\": \"log\"\n  \"type\":
\"sink\"\n\"spec\":\n  \"resource\":
\"maven://org.springframework.cloud.stream.app:log-sink-rabbit\"\n
\"resourceMetadata\": \"maven://org.springframework.cloud.stream.app:log-sink-
rabbit:jar:metadata:1.2.0.RC1\"\n  \"version\": \"1.2.0.RC1\"\n
\"applicationProperties\":\n    \"server.port\": \"0\"\n  \"deploymentProperties\":
!!null \"null\"\n",
  "links" : [ {
    "rel" : "status",
    "href" : "http://localhost:7577/api/release/status/{name}"
  } ]
}
```

Get manifest by version

The `manifest` REST endpoint can return the manifest for a specific release version.

Request structure

```
GET /api/release/manifest/test/1 HTTP/1.1
Host: localhost:7577
```

Example request

```
$ curl 'http://localhost:7577/api/release/manifest/test/1' -i
```

Response structure

```
HTTP/1.1 200 OK
Content-Type: application/hal+json
Content-Length: 636

{
  "data" : "\"apiVersion\": \"skipper.spring.io/v1\"\n\"kind\":
\"SpringCloudDeployerApplication\"\n\"metadata\":\n  \"name\": \"log\"\n  \"type\":
\"sink\"\n\"spec\":\n  \"resource\":
\"maven://org.springframework.cloud.stream.app:log-sink-rabbit\"\n
\"resourceMetadata\": \"maven://org.springframework.cloud.stream.app:log-sink-
rabbit:jar:metadata:1.2.0.RC1\"\n  \"version\": \"1.2.0.RC1\"\n
\"applicationProperties\":\n    \"server.port\": \"0\"\n  \"deploymentProperties\":
!!null \"null\"\n",
  "_links" : {
    "status" : {
      "href" : "http://localhost:7577/api/release/status/{name}",
      "templated" : true
    }
  }
}
```

30.8.6. Delete

Delete a release

You can use a `DELETE` request to delete an existing release. The delete operation does not uninstall the uploaded packages corresponding to the release.

Request structure

```
DELETE /api/release/test HTTP/1.1
Content-Type: application/json;charset=UTF-8
Accept: application/json
Host: localhost:7577
```

Example request

```
$ curl 'http://localhost:7577/api/release/test' -i -X DELETE \
    -H 'Content-Type: application/json;charset=UTF-8' \
    -H 'Accept: application/json'
```

Response structure

```
HTTP/1.1 200 OK
Content-Type: application/json
Content-Length: 2624

{
  "name" : "test",
  "version" : 1,
  "info" : {
    "status" : {
      "statusCode" : "DELETED",
      "platformStatus" : null
    },
    "firstDeployed" : null,
    "lastDeployed" : null,
    "deleted" : null,
    "description" : null
  },
  "pkg" : {
    "metadata" : {
      "apiVersion" : "skipper.spring.io/v1",
      "origin" : null,
      "repositoryId" : null,
      "repositoryName" : null,
      "kind" : "SkipperPackageMetadata",
      "name" : "log",
      "displayName" : null,
      "version" : "1.0.0",
      "packageSourceUrl" : "https://github.com/spring-cloud-stream-app-
starters/log/tree/v1.2.0.RC1",
      "packageHomeUrl" : "https://cloud.spring.io/spring-cloud-stream-app-starters/",
      "tags" : "logging, sink",
      "maintainer" : "https://github.com/sobychacko",
      "description" : "The log sink uses the application logger to output the data for
inspection.",
      "sha256" : null,
```

```
      "iconUrl" : null
    },
    "templates" : [ {
      "name" : "log.yml",
      "data" : "apiVersion: skipper.spring.io/v1\nkind:
SpringCloudDeployerApplication\nmetadata:\n  name: log\n  type: sink\nspec:\n
resource: maven://org.springframework.cloud.stream.app:log-sink-rabbit\n
resourceMetadata: maven://org.springframework.cloud.stream.app:log-sink-
rabbit:jar:metadata:{{version}}\n  version: {{version}}\n  applicationProperties:\n
server.port: 0\n    {{#spec.applicationProperties.entrySet}}\n    {{key}}: {{value}}\n
{{/spec.applicationProperties.entrySet}}\n  deploymentProperties:\n
{{#spec.deploymentProperties.entrySet}}\n    {{key}}: {{value}}\n
{{/spec.deploymentProperties.entrySet}}\n"
    } ],
    "dependencies" : [ ],
    "configValues" : {
      "raw" : "# Default values for {{name}}\n# This is a YAML-formatted file.\n#
Declare variables to be passed into your templates\nversion: 1.2.0.RC1\n"
    },
    "fileHolders" : [ ]
  },
  "configValues" : {
    "raw" : "config2: value2\nconfig1: value1\n"
  },
  "manifest" : {
    "data" : "\"apiVersion\": \"skipper.spring.io/v1\"\n\"kind\":
\"SpringCloudDeployerApplication\"\n\"metadata\":\n  \"name\": \"log\"\n  \"type\":
\"sink\"\n\"spec\":\n  \"resource\":
\"maven://org.springframework.cloud.stream.app:log-sink-rabbit\"\n
\"resourceMetadata\": \"maven://org.springframework.cloud.stream.app:log-sink-
rabbit:jar:metadata:1.2.0.RC1\"\n  \"version\": \"1.2.0.RC1\"\n
\"applicationProperties\":\n    \"server.port\": \"0\"\n  \"deploymentProperties\":
!!null \"null\"\n"
  },
  "platformName" : "default",
  "links" : [ {
    "rel" : "status",
    "href" : "http://localhost:7577/api/release/status/{name}"
  } ]
}
```


Response fields

Path	Type	Description
`name`	`String`	Name of the release
`version`	`Number`	Version of the release

Path	Type	Description
info.status.statusCode	String	StatusCode of the release's status (UNKNOWN,DEPLOYED,DELETED,FAILED)
info.status.platformStatus	Null	Status from the underlying platform
info.firstDeployed	Null	Date/Time of first deployment
info.lastDeployed	Null	Date/Time of last deployment
info.deleted	Null	Date/Time of when the release was deleted
info.description	Null	Human-friendly 'log entry' about this release
pkg.metadata.apiVersion	String	The Package Index spec version this file is based on
pkg.metadata.origin	Null	Indicates the origin of the repository (free form text)
pkg.metadata.repositoryId	Null	The repository ID this Package belongs to.
pkg.metadata.repositoryName	Null	The repository name this Package belongs to.
pkg.metadata.kind	String	What type of package system is being used
pkg.metadata.name	String	The name of the package
pkg.metadata.displayName	Null	Display name of the release
pkg.metadata.version	String	The version of the package
pkg.metadata.packageSourceUrl	String	Location to source code for this package
pkg.metadata.packageHomeUrl	String	The home page of the package
pkg.metadata.tags	String	A comma separated list of tags to use for searching
pkg.metadata.maintainer	String	Who is maintaining this package
pkg.metadata.description	String	Brief description of the package
pkg.metadata.sha256	Null	Hash of package binary that will be downloaded using SHA256 hash algorithm
pkg.metadata.iconUrl	Null	Url location of a icon
pkg.templates[].name	String	Name is the path-like name of the template
pkg.templates[].data	String	Data is the template as string data

Path	Type	Description
pkg.dependencies	Array	The packages that this package depends upon
pkg.configValues.raw	String	The raw YAML string of configuration values
pkg.fileHolders	Array	Miscellaneous files in a package, e.g. README, LICENSE, etc.
configValues.raw	String	The raw YAML string of configuration values
manifest.data	String	The manifest of the release
platformName	String	Platform name of the release

Delete a release and uninstall package

You can use a DELETE request to delete an existing release and uninstall the packages corresponding to the release, provided there are no other releases in active state use these packages.

Request structure

```
DELETE /api/release/test/package HTTP/1.1
Content-Type: application/json
Accept: application/json
Host: localhost:7577
```

Example request

```
$ curl 'http://localhost:7577/api/release/test/package' -i -X DELETE \
    -H 'Content-Type: application/json' \
    -H 'Accept: application/json'
```

Response structure

```
HTTP/1.1 200 OK
Content-Type: application/json
Content-Length: 2624

{
  "name" : "test",
  "version" : 1,
  "info" : {
    "status" : {
      "statusCode" : "DELETED",
      "platformStatus" : null
    },
```

```
    "firstDeployed" : null,
    "lastDeployed" : null,
    "deleted" : null,
    "description" : null
  },
  "pkg" : {
    "metadata" : {
      "apiVersion" : "skipper.spring.io/v1",
      "origin" : null,
      "repositoryId" : null,
      "repositoryName" : null,
      "kind" : "SkipperPackageMetadata",
      "name" : "log",
      "displayName" : null,
      "version" : "1.0.0",
      "packageSourceUrl" : "https://github.com/spring-cloud-stream-app-
starters/log/tree/v1.2.0.RC1",
      "packageHomeUrl" : "https://cloud.spring.io/spring-cloud-stream-app-starters/",
      "tags" : "logging, sink",
      "maintainer" : "https://github.com/sobychacko",
      "description" : "The log sink uses the application logger to output the data for
inspection.",
      "sha256" : null,
      "iconUrl" : null
    },
    "templates" : [ {
      "name" : "log.yml",
      "data" : "apiVersion: skipper.spring.io/v1\nkind:
SpringCloudDeployerApplication\nmetadata:\n  name: log\n  type: sink\nspec:\n
resource: maven://org.springframework.cloud.stream.app:log-sink-rabbit\n
resourceMetadata: maven://org.springframework.cloud.stream.app:log-sink-
rabbit:jar:metadata:{{version}}\n  version: {{version}}\n  applicationProperties:\n
server.port: 0\n    {{#spec.applicationProperties.entrySet}}\n    {{key}}: {{value}}\n
{{/spec.applicationProperties.entrySet}}\n  deploymentProperties:\n
{{#spec.deploymentProperties.entrySet}}\n    {{key}}: {{value}}\n
{{/spec.deploymentProperties.entrySet}}\n"
    } ],
    "dependencies" : [ ],
    "configValues" : {
      "raw" : "# Default values for {{name}}\n# This is a YAML-formatted file.\n#
Declare variables to be passed into your templates\nversion: 1.2.0.RC1\n"
    },
    "fileHolders" : [ ]
  },
  "configValues" : {
    "raw" : "config2: value2\nconfig1: value1\n"
  },
  "manifest" : {
    "data" : "\"apiVersion\": \"skipper.spring.io/v1\"\n\"kind\":
\"SpringCloudDeployerApplication\"\n\"metadata\":\n  \"name\": \"log\"\n  \"type\":
\"sink\"\n\"spec\":\n  \"resource\":
```

```
\"maven://org.springframework.cloud.stream.app:log-sink-rabbit\"\n
\"resourceMetadata\": \"maven://org.springframework.cloud.stream.app:log-sink-
rabbit:jar:metadata:1.2.0.RC1\"\n  \"version\": \"1.2.0.RC1\"\n
\"applicationProperties\":\n    \"server.port\": \"0\"\n  \"deploymentProperties\":
!!null \"null\"\n"
  },
  "platformName" : "default",
  "links" : [ {
    "rel" : "status",
    "href" : "http://localhost:7577/api/release/status/{name}"
  } ]
}
```

Response fields

Path	Type	Description
name	String	Name of the release
version	Number	Version of the release
info.status.statusCode	String	StatusCode of the release's status (UNKNOWN,DEPLOYED,DELETED,FAILED)
info.status.platformStatus	Null	Status from the underlying platform
info.firstDeployed	Null	Date/Time of first deployment
info.lastDeployed	Null	Date/Time of last deployment
info.deleted	Null	Date/Time of when the release was deleted
info.description	Null	Human-friendly 'log entry' about this release
pkg.metadata.apiVersion	String	The Package Index spec version this file is based on
pkg.metadata.origin	Null	Indicates the origin of the repository (free form text)
pkg.metadata.repositoryId	Null	The repository ID this Package belongs to.
pkg.metadata.repositoryName	Null	The repository name this Package belongs to.
pkg.metadata.kind	String	What type of package system is being used
pkg.metadata.name	String	The name of the package
pkg.metadata.displayName	Null	Display name of the release
pkg.metadata.version	String	The version of the package

Path	Type	Description
pkg.metadata.packageSourceUrl	String	Location to source code for this package
pkg.metadata.packageHomeUrl	String	The home page of the package
pkg.metadata.tags	String	A comma separated list of tags to use for searching
pkg.metadata.maintainer	String	Who is maintaining this package
pkg.metadata.description	String	Brief description of the package
pkg.metadata.sha256	Null	Hash of package binary that will be downloaded using SHA256 hash algorithm
pkg.metadata.iconUrl	Null	Url location of a icon
pkg.templates[].name	String	Name is the path-like name of the template
pkg.templates[].data	String	Data is the template as string data
pkg.dependencies	Array	The packages that this package depends upon
pkg.configValues.raw	String	The raw YAML string of configuration values
pkg.fileHolders	Array	Miscellaneous files in a package, e.g. README, LICENSE, etc.
configValues.raw	String	The raw YAML string of configuration values
manifest.data	String	The manifest of the release
platformName	String	Platform name of the release

30.8.7. Cancel

Cancel a release

You can use a POST request to cancel an existing release operation.

Request structure

```
POST /api/release/cancel HTTP/1.1
Content-Type: application/json;charset=UTF-8
Accept: application/json
Host: localhost:7577
```

Example request

```
$ curl 'http://localhost:7577/api/release/cancel' -i -X POST \
    -H 'Content-Type: application/json;charset=UTF-8' \
    -H 'Accept: application/json'
```

Response structure

```
HTTP/1.1 200 OK
Content-Type: application/json
Content-Length: 23

{
  "accepted" : true
}
```

Response fields

Path	Type	Description
`accepted`	`Boolean`	If cancel request was accepted

Appendices

Having trouble with Spring Cloud Skipper, We'd like to help!

- Ask a question - we monitor stackoverflow.com for questions tagged with `spring-cloud-skipper`.
- Report bugs with Spring Cloud Skipper at github.com/spring-cloud/spring-cloud-skipper/issues.

Appendix A: Building

To build the source, you need to install JDK 1.8.

The build uses the Maven wrapper so that you do not have to install a specific version of Maven.

The main build command is

```
$ ./mvnw clean install
```

To create the executables and avoid running the tests and generating JavaDocs, use the following command:

```
$ ./mvnw clean package -DskipTests -Dmaven.javadoc.skip=true
```

You can also install Maven (>=3.3.3) yourself and run the `mvn` command in place of `./mvnw` in the examples. If you do so, you also might need to add `-P spring` if your local Maven settings do not contain repository declarations for spring pre-release artifacts.

You might need to increase the amount of memory available to Maven by setting a `MAVEN_OPTS` environment variable with a value like `-Xmx512m -XX:MaxPermSize=128m`. We try to cover this in the `.mvn` configuration, so, if you find you have to increase memory to make a build succeed, please raise a ticket to get the settings added to source control.

A.1. Documentation

To generate only the REST Docs documentation, use the following command:

```
$ ./mvnw test -pl spring-cloud-skipper-server-core -Dtest=*Documentation*
```

To build the only the Asciidoctor documentation, use the following command:

```
$ ./mvnw package -DskipTests -Pfull -pl spring-cloud-skipper-docs
```

A.2. Custom Server Build

This chapter contains instructions how to create a custom server build and should cause exactly same packaged *uber-jar* compared to one from a Skipper build itself.

It is required to follow same *Spring Boot* main class structure used in Skipper itself. Example of it is shown below:

```
package com.example.customskipperserver;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import
org.springframework.boot.actuate.autoconfigure.ManagementWebSecurityAutoConfiguration;
import org.springframework.boot.autoconfigure.security.SecurityAutoConfiguration;
import org.springframework.boot.autoconfigure.session.SessionAutoConfiguration;
import
org.springframework.cloud.deployer.spi.cloudfoundry.CloudFoundryDeployerAutoConfigurat
ion;
import org.springframework.cloud.deployer.spi.kubernetes.KubernetesAutoConfiguration;
import org.springframework.cloud.deployer.spi.local.LocalDeployerAutoConfiguration;
import org.springframework.cloud.skipper.server.EnableSkipperServer;

@SpringBootApplication(exclude = {
                CloudFoundryDeployerAutoConfiguration.class,
                KubernetesAutoConfiguration.class,
                LocalDeployerAutoConfiguration.class,
                ManagementWebSecurityAutoConfiguration.class,
                SecurityAutoConfiguration.class,
                SessionAutoConfiguration.class
        })
@EnableSkipperServer
public class CustomSkipperServerApplication {

    public static void main(String[] args) {
        SpringApplication.run(CustomSkipperServerApplication.class, args);
    }
}
```

Working build file for *Maven* would look like something shown below:

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi=
"http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
https://maven.apache.org/xsd/maven-4.0.0.xsd">
    <modelVersion>4.0.0</modelVersion>

    <groupId>com.example</groupId>
    <artifactId>custom-skipper-server</artifactId>
    <version>0.0.1-SNAPSHOT</version>
    <packaging>jar</packaging>

    <name>custom-skipper-server</name>
    <description>Demo project for Spring Boot</description>

    <parent>
```

```
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-parent</artifactId>
        <version>1.5.9.RELEASE</version>
        <relativePath/> <!-- lookup parent from repository -->
    </parent>

    <properties>
        <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
        <project.reporting.outputEncoding>UTF-8</project.reporting.outputEncoding>
        <java.version>1.8</java.version>
        <spring-cloud.version>Dalston.SR5</spring-cloud.version>
        <spring-cloud-skipper.version>2.5.0-SNAPSHOT</spring-cloud-skipper.version>
        <!--
            reactor and flyway are managed by boot so only clean way with maven is to
            change version properties. trying to import boms in dependencyManagement
            would not actually change versions.
        -->
        <reactor.version>3.0.7.RELEASE</reactor.version>
        <flyway.version>5.0.5</flyway.version>
    </properties>

    <dependencies>
        <dependency>
            <groupId>org.springframework.cloud</groupId>
            <artifactId>spring-cloud-starter-skipper-server</artifactId>
        </dependency>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-test</artifactId>
            <scope>test</scope>
        </dependency>
    </dependencies>

    <dependencyManagement>
        <dependencies>
            <dependency>
                <groupId>org.springframework.cloud</groupId>
                <artifactId>spring-cloud-dependencies</artifactId>
                <version>${spring-cloud.version}</version>
                <type>pom</type>
                <scope>import</scope>
            </dependency>
            <dependency>
                <groupId>org.springframework.cloud</groupId>
                <artifactId>spring-cloud-skipper-dependencies</artifactId>
                <version>${spring-cloud-skipper.version}</version>
                <type>pom</type>
                <scope>import</scope>
            </dependency>
        </dependencies>
    </dependencyManagement>
```

```
    <build>
        <plugins>
            <plugin>
                <groupId>org.springframework.boot</groupId>
                <artifactId>spring-boot-maven-plugin</artifactId>
            </plugin>
        </plugins>
    </build>
</project>
```

Working build file for *Gradle* would look like something shown below:

```
buildscript {
    ext {
        springBootVersion = '1.5.9.RELEASE'
    }
    repositories {
        mavenCentral()
    }
    dependencies {
        classpath("org.springframework.boot:spring-boot-gradle-plugin:
${springBootVersion}")
    }
}

apply plugin: 'java'
apply plugin: 'eclipse'
apply plugin: 'org.springframework.boot'

group = 'com.example'
version = '0.0.1-SNAPSHOT'
sourceCompatibility = 1.8

repositories {
    mavenLocal()
    mavenCentral()
    maven { url "https://repo.springsource.org/libs-snapshot" }
    maven { url "https://repo.springsource.org/libs-release" }
    maven { url "https://repo.springsource.org/libs-milestone" }
}


ext {
    springCloudVersion = 'Dalston.SR5'
    springCloudSkipperVersion = '2.5.0-SNAPSHOT'
    reactorVersion = 'Aluminium-SR3'
    reactorNettyVersion = '0.6.6.RELEASE'
    objenesisVersion = '2.1'
```

```
}

dependencies {
    compile('org.springframework.cloud:spring-cloud-starter-skipper-server')
    testCompile('org.springframework.boot:spring-boot-starter-test')
}

dependencyManagement {
    imports {
        mavenBom "org.springframework.cloud:spring-cloud-dependencies:
${springCloudVersion}"
        mavenBom "org.springframework.cloud:spring-cloud-skipper-dependencies:
${springCloudSkipperVersion}"
        mavenBom "io.projectreactor:reactor-bom:${reactorVersion}"
    }
    dependencies {
        // latest reactor bom is still using reactor-netty:0.6.3.RELEASE
        // so we need to change it here because cf java client use
        // dedicated netty version while they should have been using
        // reactor boms assuming reactor boms would be up-to-date
        dependency "io.projectreactor.ipc:reactor-netty:${reactorNettyVersion}"
        // this is unfortunate mess with objenesis as there's versions 2.1 and 2.6
        // in build path and nobody manages version and maven vs. gradle is different
        dependency "org.objenesis:objenesis:${objenesisVersion}"
    }
}
```

A.3. Importing into eclipse

You can generate Eclipse project metadata by using the following command:

```
$ ./mvnw eclipse:eclipse
```

In Eclipse, the generated projects can be imported by selecting `Import existing projects` from the `File` menu.

Appendix B: Contributing

Spring Cloud is released under the non-restrictive Apache 2.0 license and follows a standard Github development process, using Github tracker for issues and merging pull requests into master. If you want to contribute even something trivial, please do not hesitate, but please do follow the guidelines spelled out in this section.

B.1. Sign the Contributor License Agreement

Before we accept a non-trivial patch or pull request, we need you to sign the contributor's agreement. Signing the contributor's agreement does not grant anyone commit rights to the main repository, but it does mean that we can accept your contributions. You will get an author credit if we do. Active contributors might be asked to join the core team and be given the ability to merge pull requests.

B.2. Code Conventions and Housekeeping

None of these conventions is essential for a pull request, but they all help. They can also be added after the original pull request but before a merge.

- Use the Spring Framework code format conventions. Follow these instructions for setting up the eclipse formatter in eclipse or IntelliJ. Note that checkstyle is enabled in the build.
- Make sure all new `.java` files have a simple Javadoc class comment with at least an `@author` tag identifying you and preferably at least a paragraph on what the class is for.
- Add the ASF license header comment to all new `.java` files. To do so, copy from existing files in the project.
- Add yourself as an `@author` to the `.java` files that you modify substantially (more than cosmetic changes).
- Add some Javadocs and, if you change the namespace, some XSD doc elements.
- A few unit tests would help a lot as well — someone has to do it, and your fellow developers appreciate it.
- If no-one else is using your branch, please rebase it against the current master (or other target branch in the main project).
- When writing a commit message, please follow these conventions. If you are fixing an existing issue, please add `Fixes gh-XXXX` at the end of the commit message (where XXXX is the issue number).