This document illustrates the steps required to orchestrate Google PubSub binder based streams in SCDF running on PCF + GCP environment.

Server

1) If we are to provide automatic pubsub binding to all the stream apps orchestrated by SCDF, it is necessary to supply the pubsub SI instance name like the following.

```
SPRING_CLOUD_DEPLOYER_CLOUDFOUNDRY_STREAM_SERVICES: pubsub
```

(where `pubsub` *is the service-instance name)*

2) The GCP broker used to expect the "projectName" to be supplied by the client applications, so we would have to supply the following env-var in the manifest.

```
SPRING_APPLICATION_JSON:
"{\"spring.cloud.dataflow.applicationProperties.stream.spring.cloud.s
tream.pubsub.binder.projectName\":\"spring-cloud-dataflow-148214\"}"
```

It *may not be necessary* if the latest release of GCP service-broker is used.

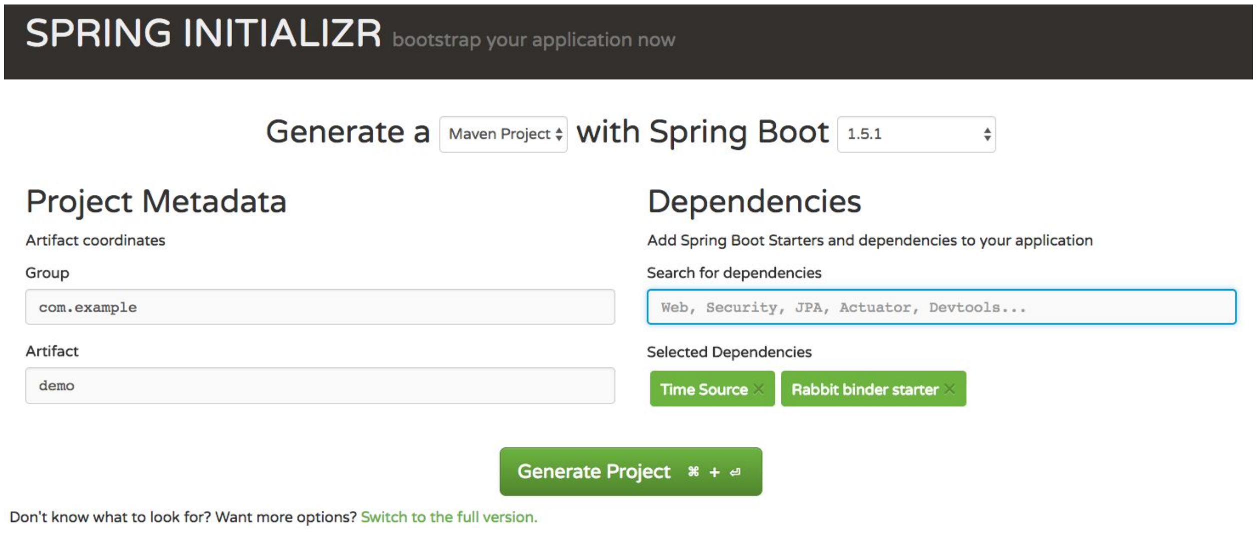
Apps

The pubsub binder has not been officially released so far - we only have SNAPSHOT builds. Since there's no publicly consumable artifact of this binder implementation, we do not generate OOTB apps with this binder implementation. We only generate it for rabbit, kafka9, and kafka10. So, it is necessary to build OOTB apps with this binder implementation on your own.

To do that, you can download the necessary applications from http://start-scs.cfapps.io/ and update the binder dependency manually.

For "`time | log`" example, you'd do the following.

1) Download apps


SPRING INITIALIZR bootstrap your application now

Generate a Maven Project with Spring Boot 1.5.1

Project Metadata

Artifact coordinates

Group: com.example

Artifact: demo

Dependencies

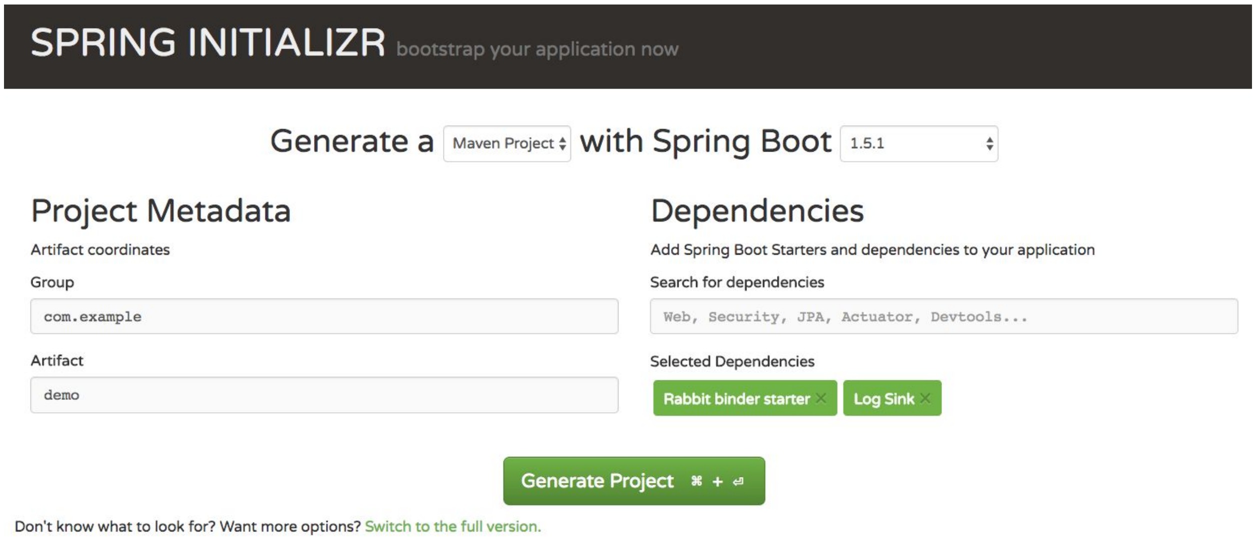
Add Spring Boot Starters and dependencies to your application

Search for dependencies: Web, Security, JPA, Actuator, Devtools...

Selected Dependencies: Time Source, Rabbit binder starter

Generate Project ⌘ + ⏎

Don't know what to look for? Want more options? Switch to the full version.


SPRING INITIALIZR bootstrap your application now

Generate a Maven Project with Spring Boot 1.5.1

Project Metadata

Artifact coordinates

Group: com.example

Artifact: demo

Dependencies

Add Spring Boot Starters and dependencies to your application

Search for dependencies: Web, Security, JPA, Actuator, Devtools...

Selected Dependencies: Rabbit binder starter, Log Sink

Generate Project ⌘ + ⏎

Don't know what to look for? Want more options? Switch to the full version.

2) Change the following dependency for the apps

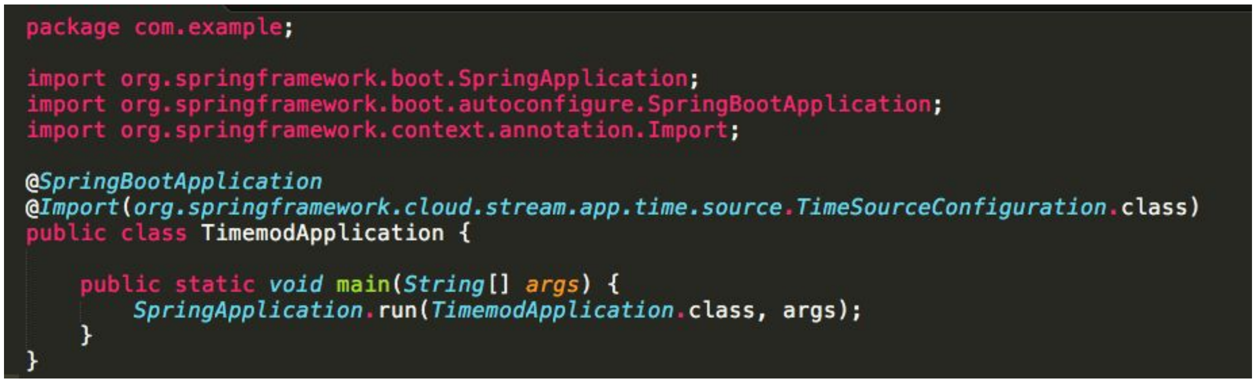
From:
```
<dependency>
  <groupId>org.springframework.cloud</groupId>
  <artifactId>spring-cloud-starter-stream-rabbit</artifactId>
</dependency>
```

To:
```
<dependency>
  <groupId>org.springframework.cloud</groupId>
  <artifactId>spring-cloud-stream-binder-pubsub</artifactId>
  <version>1.1.0.BUILD-SNAPSHOT</version>
</dependency>
```

3) When patching the OOTB apps, it is also necessary to add the respective @Configuration annotation.

For Time-source:
`@Import(org.springframework.cloud.stream.app.time.source.TimeSourceConfiguration.class)`

You'd add this in the Boot app itself. There's only one Java class in the downloaded project.



```java
package com.example;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.context.annotation.Import;

@SpringBootApplication
@Import(org.springframework.cloud.stream.app.time.source.TimeSourceConfiguration.class)
public class TimemodApplication {

    public static void main(String[] args) {
        SpringApplication.run(TimemodApplication.class, args);
    }
}
```

For Log-sink:
`@Import(org.springframework.cloud.stream.app.log.sink.LogSinkConfiguration.class)`

4) Build the apps locally

5) Register them in SCDF
In order to use the these custom built apps in PCF environment, you could either use maven repository or as remotely reachable http endpoints. I usually drop them in GH like this for example. You would then register in SCDF like the following, from the Spring Cloud Dataflow Shell:

```
app register --name s3jdbc --type task --uri
https://github.com/sabbyanandan/sandbox/raw/master/jars/ s3jdbc-0.0.1-SNAPSHOT.jar --force
```

ALTERNATIVELY, you can test the apps standalone *on PCF* before trying it with SCDF.

```
cf push time-source --no-start -p
~/Work/tmp/sandbox/time-source/target/timesource-0.0.1-SNAPSHOT.jar
-m 1024M
cf bs time-source google -c "{\"role\":\"pubsub.admin\"}"
cf set-env time-source SPRING_APPLICATION_JSON
"{\"spring.cloud.stream.bindings.output.destination\":\"ticktock\"}"
cf set-env time-source SPRING_CLOUD_STREAM_PUBSUB_BINDER_PROJECT_NAME
spring-cloud-dataflow-148214
cf start time-source


cf push log-sink --no-start -p
~/Work/tmp/sandbox/log-sink/target/logsink-0.0.1-SNAPSHOT.jar -m
1024M
cf bs log-sink google -c "{\"role\":\"pubsub.admin\"}"
cf set-env log-sink SPRING_APPLICATION_JSON
"{\"spring.cloud.stream.bindings.input.destination\":\"ticktock\"}"
cf set-env log-sink SPRING_CLOUD_STREAM_PUBSUB_BINDER_PROJECT_NAME
spring-cloud-dataflow-148214
cf start log-sink
```