Pivotal

PRODUCT DOCUMENTATION

Spring Cloud Data Flow for Pivotal Cloud Foundry® Documentation

Version 1.1.4

Published: 2 April 2019

Spring Cloud® Data Flow for PCF

Note: Spring Cloud Data Flow for PCF v1.1 is no longer supported. The support period for v1.1 has expired. To stay up to date with the latest software and security updates, upgrade to a supported version.

About Spring Cloud Data Flow

Spring Cloud Data Flow is an open-source toolkit for data integration and real-time data processing (see the project home page). The Spring Cloud Data Flow (SCDF) server deploys pipelines composed of Spring Cloud Stream or Spring Cloud Task applications. A domain-specific language (DSL) makes it easy to describe the applications in a pipeline and how they are connected.

The SCDF server exposes a REST API for composing and deploying data pipelines and a dashboard with a graphical pipeline editor. It also includes a shell for working with the API from the command line.

About Spring Cloud Data Flow for PCF

Spring Cloud Data Flow for Pivotal Cloud Foundry (PCF) automates the deployment of SCDF and its dependent services so that developers can use Apps Manager to deploy their own SCDF instances. They can then access their Data Flow server, either via the SCDF dashboard UI or using their PCF credentials with the SCDF shell.

Key Features

Spring Cloud Data Flow for PCF includes the following key features:

- Addition of the Spring Cloud Data Flow server to the Marketplace as a managed service
- Automatic integration with dependent PCF services:
  - MySQL for PCF (or a user-provided relational database service) for apps, pipelines, and task history
  - RabbitMQ for PCF (or a user-provided RabbitMQ or Kafka service) for event messaging
  - Redis for PCF (or a user-provided Redis service) for capturing analytics data
- Integration of Data Flow server with PCF's UAA security model

Product Snapshot

The following table provides version and version-support information about Spring Cloud Data Flow for PCF.

Element	Details
Version	v1.1.4
Release date	September 7, 2018
Software component version	v1.5.1.RELEASE
Compatible Ops Manager version(s)	v1.10.x, v1.11.x, v1.12.x, v2.0.x, and v2.1.x
Compatible Elastic Runtime version(s)	v1.10.x, v1.11.x, v1.12.x, v2.0.x, and v2.1.x
IaaS support	All supported by PCF

Requirements

Spring Cloud Data Flow for PCF has the following requirements:

- Java Cloud Foundry buildpack version 3.8 or later (see the Prerequisites section of Installation for more information)
- MySQL for PCF v2, or an alternative relational database service
- RabbitMQ for PCF, or an alternative RabbitMQ or Kafka service
- Redis for PCF, or an alternative Redis service
- Cloud Foundry Command Line Interface (cf CLI)

Optional

To use the Spring Cloud Data Flow shell interface with Spring Cloud Data Flow for PCF service instances, install the following cf CLI plugins:

- Spring Cloud Data Flow for PCF cf CLI plugin. To install the plugin, run the following command:

```
$ cf install-plugin -r CF-Community "spring-cloud-dataflow-for-pcf"
```

  - Java Runtime Environment (JRE). Required to run the Data Flow shell. You can download the JRE from the Java website.
- Service Instance Logging cf CLI plugin. To install the plugin, run the following command:

```
$ cf install-plugin -r CF-Community "Service Instance Logging"
```

Please provide any bugs, feature requests, or questions to the Pivotal Cloud Foundry Feedback list.

Spring Cloud Data Flow Release Notes

Release notes for Spring Cloud Data Flow for PCF

v1.1.4

Release Date: September 7, 2018

Fixes included in this release:

- Fixed an issue that prevented the “run-tests” errand from completing when the Spring Cloud Data Flow tile was configured to not allow internet access.

v1.1.3

Release Date: August 6, 2018

Enhancements included in this release:

- A new “Do not allow public Internet access from service instances” option in the Data Flow tile configuration prevents service instances from accessing the Internet.
- The “run-tests” errand now complies with the UAA password policy when creating its temporary test user.

Fixes included in this release:

- The Java buildpack setting configured in the Data Flow tile or for a particular service instance is now honored.

v1.1.2

Release Date: July 16, 2018

Enhancements included in this release:

- Service instances now allow `spring.cloud.dataflow.applicationProperties.*` to be set via `-c` properties to `cf` create and update commands

v1.1.1

Release Date: June 25, 2018

Fixes included in this release:

- The broker now proceeds with an update operation for a service instance that previously failed to update.

v1.1.0

Release Date: June 18, 2018

Features included in this release:

- Service instances are now based on Spring Cloud Data Flow 1.5.1.RELEASE.
- Service instances now accept Maven configuration properties provided by Spring Cloud Data Flow OSS.
- The “run-tests” lifecycle errand no longer requires access to the Internet.

Installing and Configuring Spring Cloud® Data Flow for PCF

Follow the below steps to install Spring Cloud Data Flow for Pivotal Cloud Foundry (PCF).

Prerequisites

Spring Cloud Data Flow for PCF is built using Spring Boot 1.5, which requires version 3.8 or later of the Java Cloud Foundry buildpack. The default Java buildpack—the buildpack at the lowest position of all Java buildpacks—on your PCF installation must therefore be at version 3.8 or later.

You can use the Cloud Foundry Command Line Interface tool (cf CLI) to see the version of the Java buildpack that is currently installed.

```
$ cf buildpacks
Getting buildpacks...

buildpack              position   enabled   locked   filename
java_buildpack_offline  1          true      false    java-buildpack-offline-v3.8.1.zip
ruby_buildpack          2          true      false    ruby_buildpack-cached-v1.6.19.zip
nodejs_buildpack        3          true      false    nodejs_buildpack-cached-v1.5.15.zip
go_buildpack            4          true      false    go_buildpack-cached-v1.7.10.zip
```

If the default Java buildpack is older than version 3.8, you can download a newer version from Pivotal Network and update Pivotal Cloud Foundry by following the instructions in the Managing Custom Buildpacks topic. To ensure that the newer buildpack is the default Java buildpack, you may delete or disable the older buildpack or make sure that the newer buildpack is in a lower position.

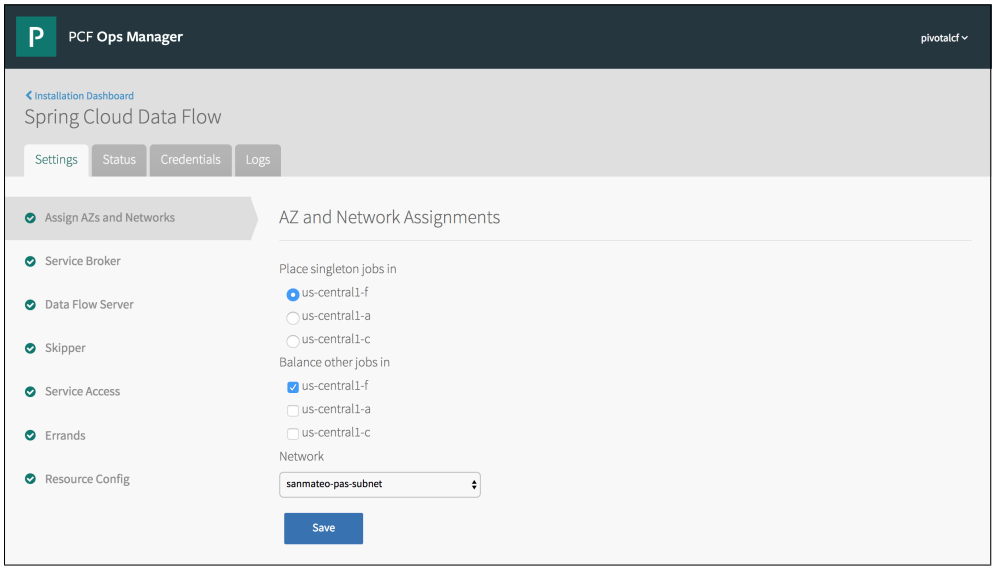
Dependent Services

Spring Cloud Data Flow relies on other PCF services for its service broker, for the Data Flow server created for each service instance, and for the Spring Cloud Skipper application created for each service instance. You can configure these in the tile settings. When following the installation steps below, review all of these settings, ensuring that you have previously installed the specified services and that the specified service plans are active.

> **Important**: If any of the services or service plans configured in the Data Flow tile settings are not available in your PCF deployment, the tile installation may fail.

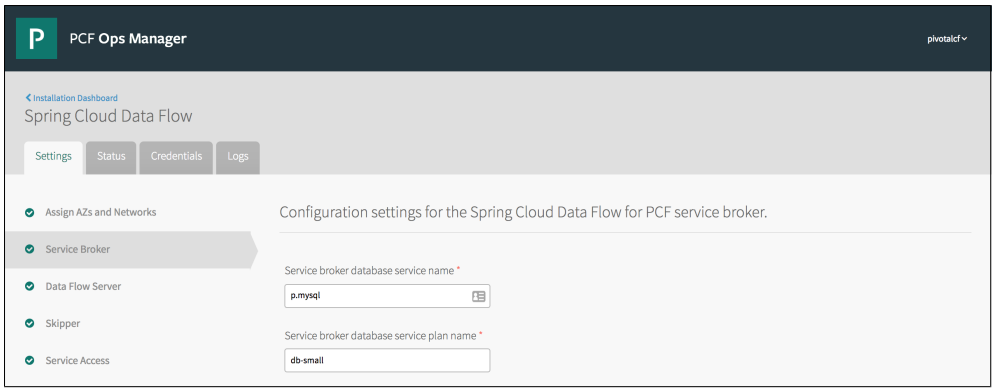
Installation Steps

1. Download the product file from Pivotal Network.
2. Navigate to the Ops Manager Installation Dashboard and click **Import a Product** to upload the product file.
3. Under the **Import a Product** button, click **+** next to the version number of Spring Cloud® Data Flow for PCF. This adds the tile to your staging area.
4. Click the newly added **Spring Cloud Data Flow** tile. In the **Settings** tab, click **Assign AZs and Networks**.

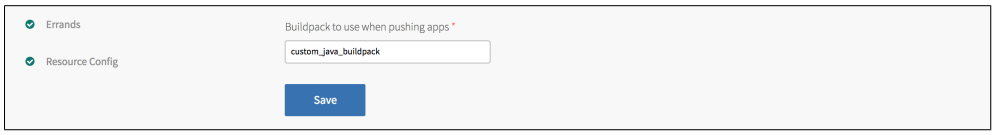


   Select the availability zones for the tile to use. In the **Network** section, select the PAS (or ERT) network.

5. In the **Settings** tab, click **Service Broker**.

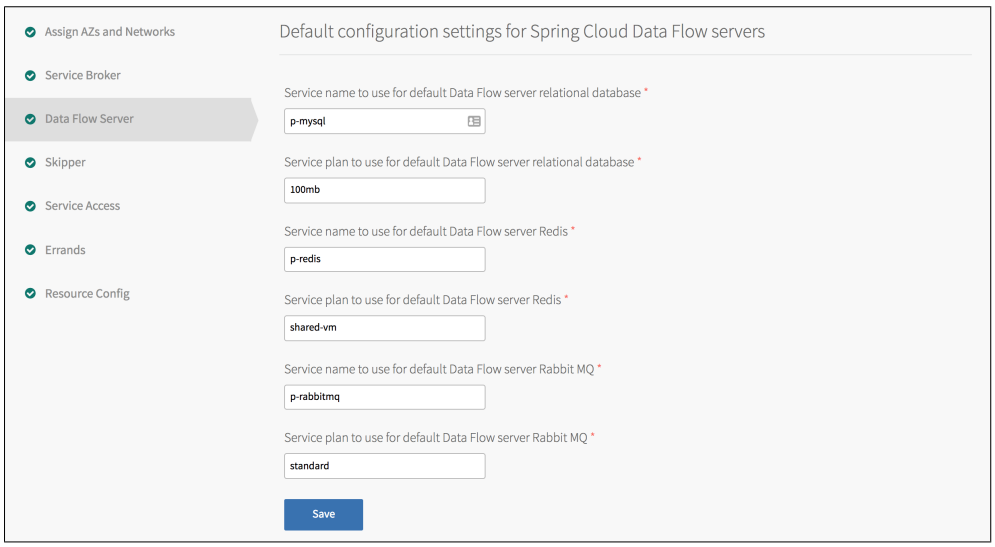


Enter the relational database service name and plan name for the Spring Cloud Data Flow tile to use for storing its service broker's service instance data.



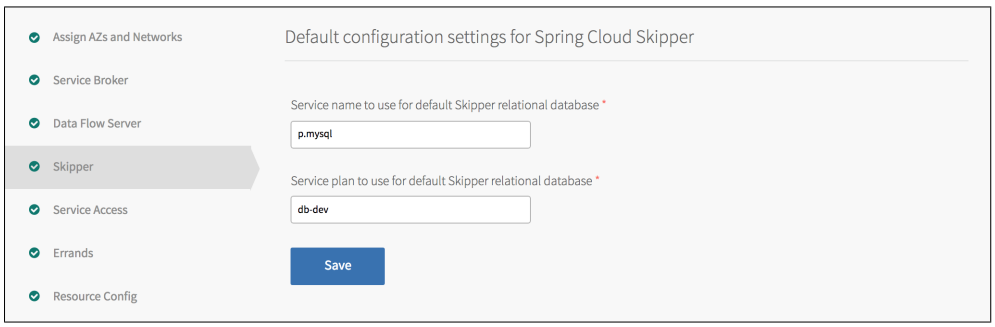
You can also configure the Java buildpack to use for deploying the broker and service instance applications (by default, the tile will use the highest-priority Java buildpack).

6. In the **Settings** tab, click **Data Flow Server**.



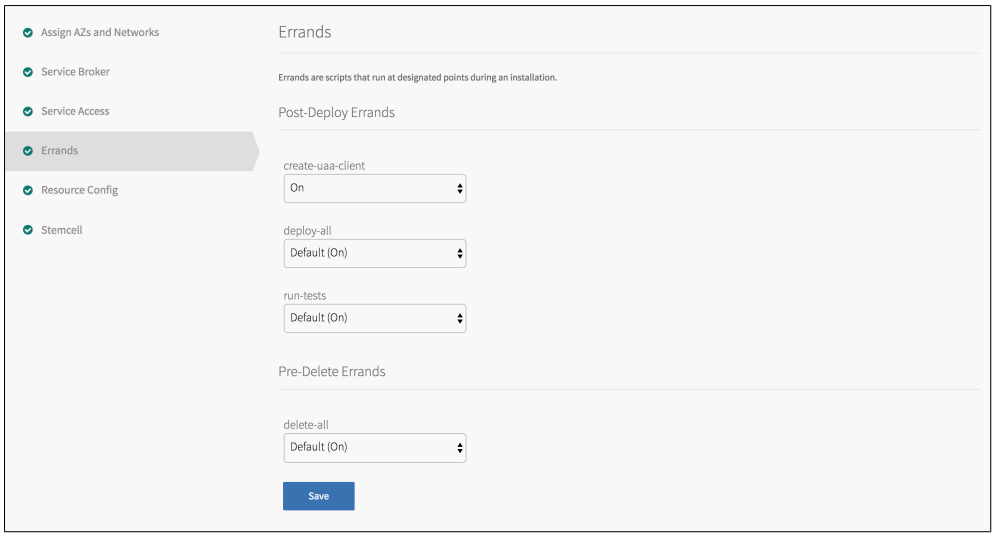
Configure the default services and service plans used by the Spring Cloud Data Flow server that is deployed for each Data Flow service instance. These values can be overridden when creating a Data Flow service instance.

7. Still in the **Settings** tab, click **Skipper**.



Configure the relational database service name and plan used by the Spring Cloud Skipper application deployed for each Data Flow service instance.

8. Still in the **Settings** tab, click **Errands**.



Spring Cloud Data Flow has four lifecycle errands. In PCF 2.2 and later, each errand can be set to always run (**On**) or to never run (**Off**). In PCF 2.1 and earlier, errands can also be set to run conditionally (**When Changed**). Pivotal recommends that all Spring Cloud Data Flow lifecycle errands be set to always run (**On**).

9. Return to the Ops Manager Installation Dashboard and click **Apply Changes** to install the Spring Cloud® Data Flow tile.

Tile Configuration

See below for information about configuring options in the Spring Cloud® Data Flow for PCF tile settings.

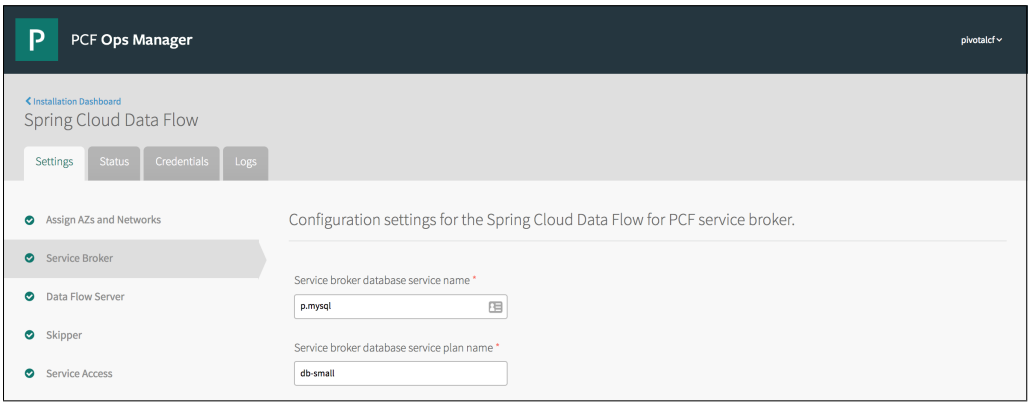
> **Note**: Ops Manager administrators can use Role-Based Access Control (RBAC) to manage which operators can make deployment changes, view credentials, and manage user roles in Ops Manager. Therefore, your role permissions might not allow you to perform every procedure in this operator guide. For more information about roles in Ops Manager, see Understand Roles in Ops Manager.

Tile Configuration Options

The Spring Cloud Data Flow tile includes settings for various options. You can configure these by visiting the Installation Dashboard of Pivotal Cloud Foundry® Operations Manager and clicking the Spring Cloud Data Flow tile.

Configure Broker Database Service and Plan

By default, the Spring Cloud Data Flow product uses the MySQL for PCF v2 product and its default `db-small` service plan to provision a database service instance for the Data Flow service broker. If you are using the MySQL for PCF v2 product and it does not have an active `db-small` plan or if you wish to use an alternative service, you must configure the Spring Cloud Data Flow product for the service and service plan you wish to use. You can configure an alternative relational database service and service plan in the **Service Broker** pane of the Spring Cloud Data Flow settings.



The broker's database service should be configured only during the Spring Cloud Data Flow tile installation process. If you have already completed the tile installation process, do not alter that setting (you may still configure the broker's database service plan after the tile installation).

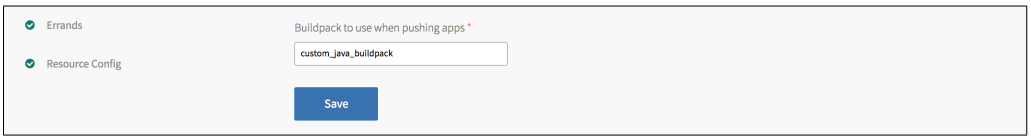
> **warning:** Configuring the service broker's dependent relational database service after the tile installation has finished can result in orphaned Data Flow service instances or in multiple data sets and corruption of the broker database's data.

See below for more information about the results of configuring the service broker's relational database.

If you	and change	then
install the tile	the service name or plan	the tile uses the specified service and plan. You may also use the default values of `p.mysql` and `db-small`.
update the tile	the service plan	the tile's relational database service instance is changed to use the specified plan. No data is lost.
update the tile	the service name	the service name change is ignored. If you have also changed the service plan, this may leave the broker in an inoperative state. To resolve this issue, revert your changes.

Configure Buildpack for Broker and Service Instances

By default, the Spring Cloud Data Flow service broker application and service instance backing applications use the buildpack chosen by the platform's buildpack detection; normally, this will be the highest-priority Java buildpack. To cause these applications to use a particular Java buildpack regardless of priority, you can set the name of the buildpack to use in the **Service Broker** pane of the Spring Cloud Data Flow tile settings.

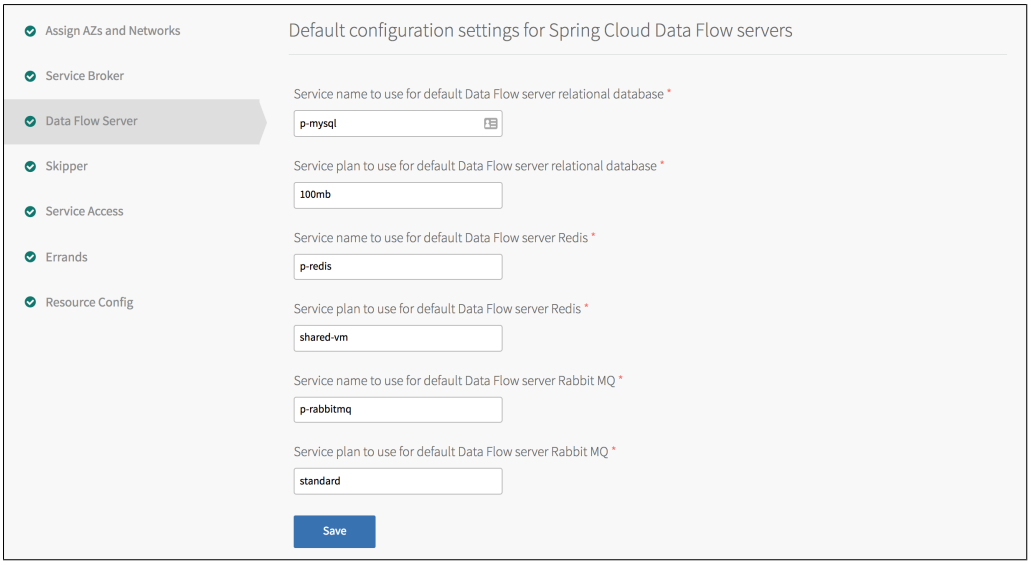


In the **Buildpack** field, enter the name of the desired Java buildpack. Click **Save**, return to the Installation Dashboard, and apply your changes. The broker application and service instance applications will now use the selected buildpack.

Configure Data Flow Server Dependent Services

By default, the Spring Cloud Data Flow product uses the MySQL for PCF v2, RabbitMQ for PCF, and Redis for PCF products to provision dependent service

instances for the Data Flow server application deployed for each Data Flow service instance. If you wish to use other services or to change the service plans used, you must configure the services and service plans you wish to use for the Data Flow server. You can configure alternate services or service plans in the **Data Flow Server** pane of the Spring Cloud Data Flow settings.

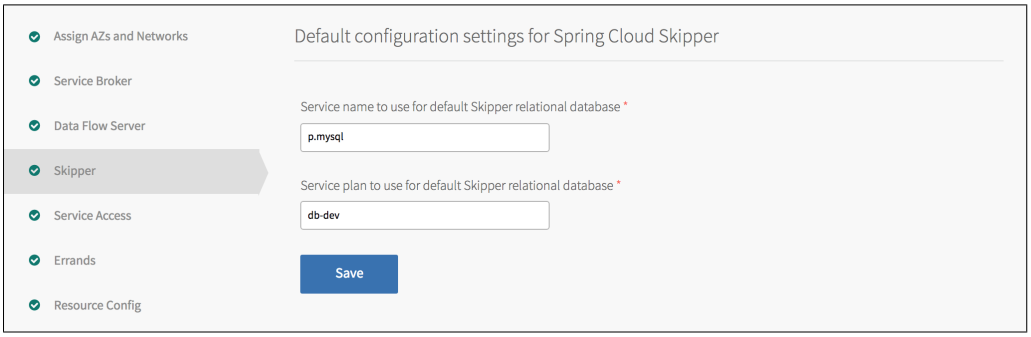


Disable Data Flow Server Internet Access

You can disable internet access for all Spring Cloud Data Flow service instances. To do this, select the “Disable public Internet access” checkbox in the **Data Flow Server** pane of the Spring Cloud Data Flow settings.

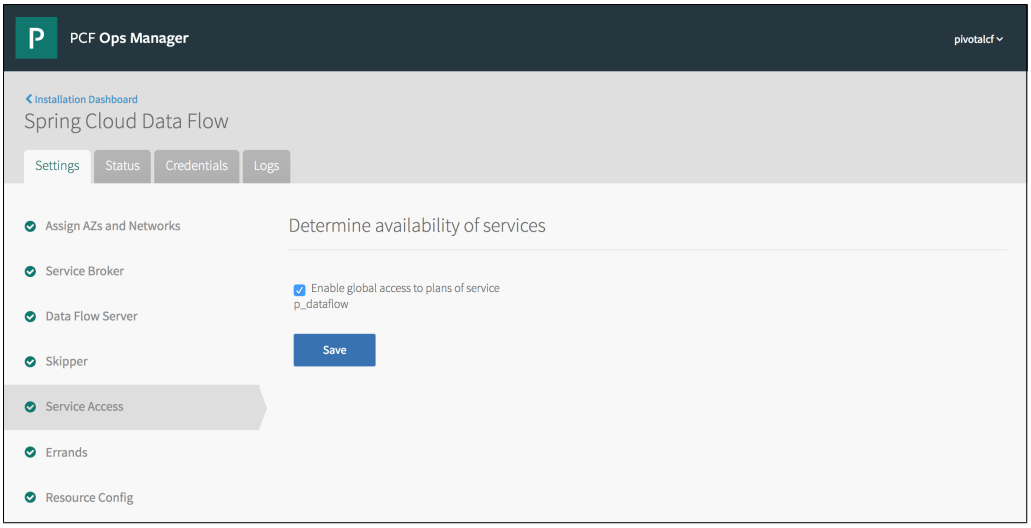
Configure Skipper Database Service and Plan

By default, the Spring Cloud Data Flow product uses the MySQL for PCF v2 product and its default `db-small` service plan to provision a database service instance for the Spring Cloud Skipper application deployed for each Data Flow service instance. If you are using the MySQL for PCF v2 product and it does not have an active `db-small` plan or if you wish to use an alternative service, you must configure the service and service plan you wish to use for Skipper. You can configure an alternative relational database service and service plan in the **Skipper** pane of the Spring Cloud Data Flow settings.



Enable or Disable Global Marketplace Access

By default, the Spring Cloud Data Flow product enables access to its service offering, called `p-dataflow`, across all orgs in the PCF deployment as part of the tile installation process. If you wish to manually grant service access to specific organizations, you can configure the default global access in the **Service Access** pane of the Spring Cloud Data Flow settings.



To disable the default of service access enabled for all orgs, clear the **Enable global access to plans of service p_dataflow** checkbox. Click **Save**, return to the Installation Dashboard, and apply your changes. You can now enable or disable access to the Data Flow service offering for specific orgs.

The Service Broker and Instances

See below for information about Spring Cloud® Data Flow's deployment model and other information which may be useful in administering Data Flow service instances or deployed applications.

> **Note**: Ops Manager administrators can use Role-Based Access Control (RBAC) to manage which operators can make deployment changes, view credentials, and manage user roles in Ops Manager. Therefore, your role permissions might not allow you to perform every procedure in this operator guide. For more information about roles in Ops Manager, see Understand Roles in Ops Manager.

The Service Broker

Spring Cloud Data Flow provides a Spring Cloud Data Flow server as a Managed Service on Pivotal Cloud Foundry (PCF). It uses Cloud Foundry's Service Broker API to manage this service. See below for information about Spring Cloud Data Flow's broker implementation.

The Spring Cloud Data Flow service broker's functionality is contained in the following Spring Boot application instance, which is deployed in the "system" organization to the "p-dataflow" space.

- **p-dataflow-[version]**: Implements the Service Broker API to act on provision, deprovision, bind, and unbind requests.

The broker relies on the MySQL for Pivotal Cloud Foundry v2 product for the following service instance.

- **p-dataflow-db**: A MySQL database used as a backing store for the service broker.

> **Note**: You can configure an alternate relational database service for the broker to use. See the Configure Broker Database Service and Plan section of the Tile Configuration topic.

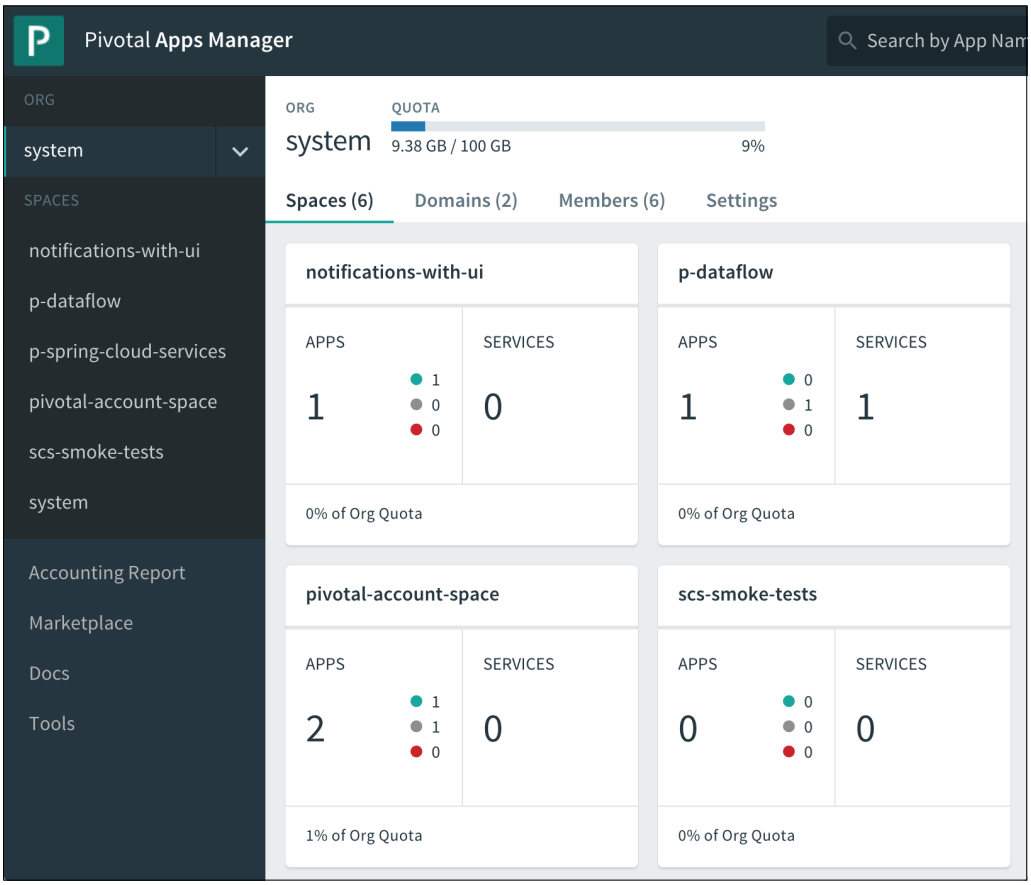
Service Broker Upgrades

The Spring Cloud Data Flow product upgrade process checks before redeploying the service broker to see whether the product's version has changed. If the version has not changed, the upgrade process will continue without redeploying the service broker.

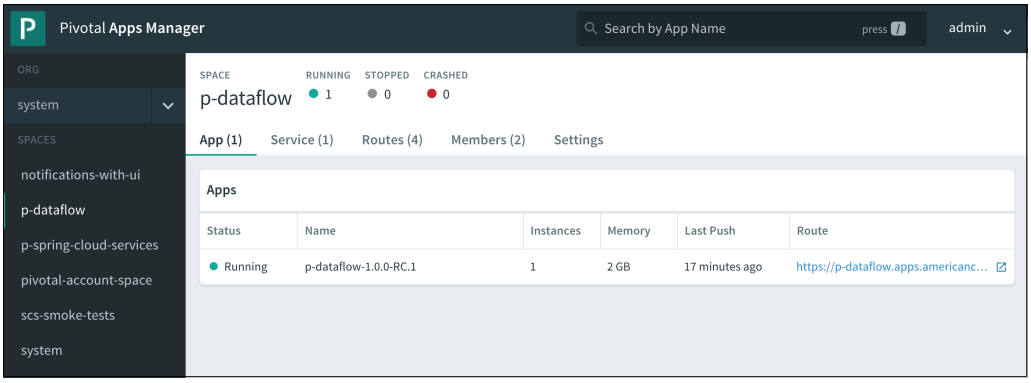
The service broker application is deployed using a blue-green deployment strategy. During an upgrade of the service broker, the broker will continue processing requests to provision, deprovision, bind, and unbind service instances, without downtime.

Access Via Apps Manager

To view the broker application in Pivotal Cloud Foundry® Apps Manager, log into Apps Manager as an admin user and select the "system" org.



The application is deployed in the “p-dataflow” space.



Get Broker Build Information

The Spring Cloud Data Flow broker provides build information using the Spring Boot Actuator `info` endpoint, which is mapped to `/info`. You can access this endpoint by appending `/info` to the path of the Spring Cloud Data Flow broker.

If the Spring Cloud Data Flow service broker application is located at the following URL:

```
https://p-dataflow.apps.wise.com
```

then you can access the `info` endpoint by visiting:

```
https://p-dataflow.apps.wise.com/info
```

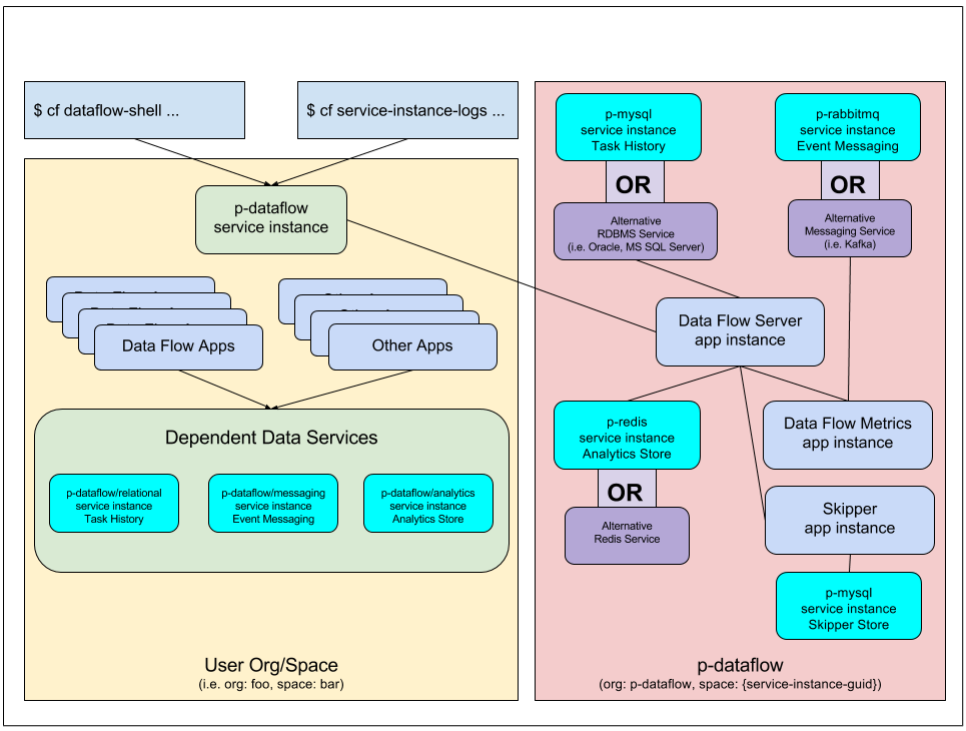
The service broker returns a JSON response, as in the following example.

```
{
  "git": {
    "commit": {
      "time": 1526673193000,
      "id": "740fb80"
    },
    "branch": "740fb80a6e1d83de033a4b81bf29c8b4cfa83b74"
  },
  "build": {
    "version": "1.1.0-build.19",
    "artifact": "scdf-for-pcf-service-broker",
    "name": "scdf-for-pcf-service-broker",
    "group": "io.pivotal.springcloud.dataflow",
    "time": 1526673593000
  }
}
```

The response contains information about the build of the service broker application, including the Maven project coordinates and build time. It also contains information about the Git repository for Spring Cloud Data Flow for PCF at build time.

> **Note**: Fields such as those for Git repository information are for diagnostic purposes and intended to provide Pivotal Support with information to help in troubleshooting.

Service Instance Architecture



For each Spring Cloud Data Flow service instance created, the service broker provisions the following resources, all within the space from which the service instance was created ("the user space") unless noted otherwise.

- A new space within the "p-dataflow" org, named using the service instance GUID and containing:
  - A Data Flow server application.
  - A Data Flow metrics application.
  - A Spring Cloud Skipper package management application.
  - A relational database service, bound to the Data Flow server application.

    Note: This relational database service is a "p.mysql" service instance by default. You can configure an alternate relational database when you create the service instance.

  - A relational database service, bound to the Skipper package management application.
  - A messaging data service, bound to the Data Flow metrics application.

    Note: This messaging data service is a "p-rabbitmq" service instance by default. You can configure an alternate messaging service when you create the service instance.

  - A Redis database service, bound to the Data Flow server application.

    Note: This Redis database service is a "p-redis" service instance by default. You can configure an alternate Redis database when you create the service instance.

- The following resources are created in the originating user space where the service instance command was targeted at:
  - A "p-dataflow" service instance.
  - A "p-dataflow" relational database service instance (providing access to the relational database service created in the service instance's space within the "p-dataflow" org).
  - A "p-dataflow" messaging service instance (providing access to the messaging data service created in the service instance's space within the "p-dataflow" org).
  - A "p-dataflow" analytics service instance (providing access to the analytics service created in the service instance's space within the "p-dataflow" org).

Capacity Requirements

Below are the usage requirements of the Spring Cloud Data Flow service broker.

Application	Memory Allocated	Disk Allocation
Service Broker	2 GB	1 GB

The service broker is bound to a relational database service instance, which stores data relating to the broker's service instances. The relational database service to use is configurable, as described in The Service Broker section above.

Below are the usage requirements of the Data Flow server and metrics applications that back each Spring Cloud Data Flow service instance.

Backing Application	Memory Allocation / App Instance	Disk Allocation / App Instance
Data Flow Server	2 GB	2 GB
Data Flow Metrics	1 GB	1 GB

Backing Application	Memory Allocation / App Instance	Disk Allocation / App Instance
Spring Cloud Skipper	1 GB	2 GB

Spring Cloud Data Flow service instances are also backed by instances of other PCF services. These are either services from PCF data service products or custom services provided to a Data Flow service instance at create time. These data services include a relational database service, a messaging service, and an analytics database service instance for each Data Flow service instance created. The Skipper backing application uses a MySQL database service for its backing store.

Getting Started with Spring Cloud® Data Flow for PCF

See below for steps to get started using Spring Cloud Data Flow for PCF. The examples below use Spring Cloud Data Flow for PCF to quickly create a data pipeline.

> Consider installing the Spring Cloud Data Flow for PCF and Service Instance Logs cf CLI plugins. See the Using the Shell and Viewing Service Instance Logs topics.
>
> The examples in this topic use the Spring Cloud Data Flow for PCF cf CLI plugin.

Creating a Data Pipeline Using the Shell

Create a Spring Cloud Data Flow service instance (see the Creating an Instance section of the Managing Service Instances topic). If you use the default backing data services of MySQL for PCF v2, RabbitMQ for PCF, and Redis for PCF, you can then import the Spring Cloud Data Flow OSS "RabbitMQ + Maven" stream app starters.

Start the Data Flow shell using the `cf dataflow-shell` command added by the Spring Cloud Data Flow for PCF cf CLI plugin:

```
$ cf dataflow-shell data-flow
...
Welcome to the Spring Cloud Data Flow shell. For assistance hit TAB or type "help".
dataflow:>
```

Import the stream app starters using the Data Flow shell's `app import` command:

```
dataflow:>app import https://bit.ly/Celsius-SR2-stream-applications-rabbit-maven
Successfully registered 65 applications from [source.sftp, source.mqtt.metadata,
sink.mqtt.metadata,... processor.scriptable-transform.metadata]
```

With the app starters imported, you can use three–the `http` source, the `split` processor, and the `log` sink–to create a stream that consumes data via an HTTP POST request, processes it by splitting it into words, and outputs the results in logs.

Create the stream using the Data Flow shell's `stream create` command:

```
dataflow:>stream create --name words --definition "http | splitter --expression=payload.split(' ') | log"
Created new stream 'words'
```

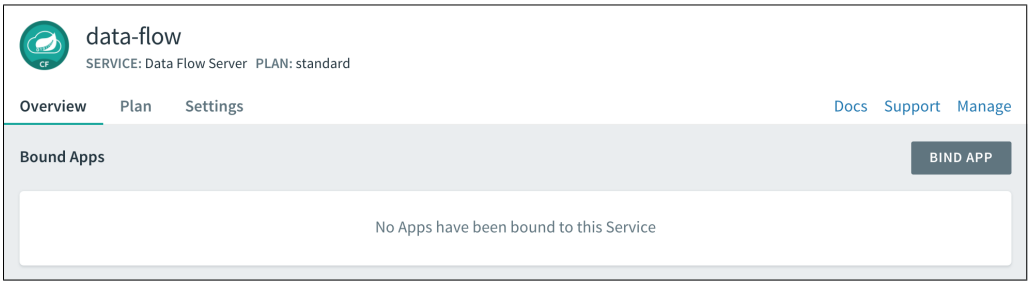
Next, deploy the stream, using the `stream deploy` command:

```
dataflow:>stream deploy words
Deployment request has been sent for stream 'words'
```

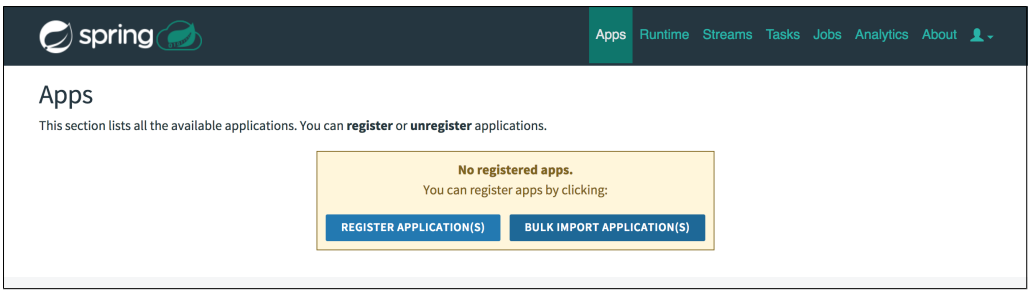
Creating a Data Pipeline Using the Dashboard

Create a Spring Cloud Data Flow service instance (see the Creating an Instance section of the Managing Service Instances topic). If you use the default backing data services of MySQL for PCF v2, RabbitMQ for PCF, and Redis for PCF, you can then import the Spring Cloud Data Flow OSS "RabbitMQ + Maven" stream app starters.

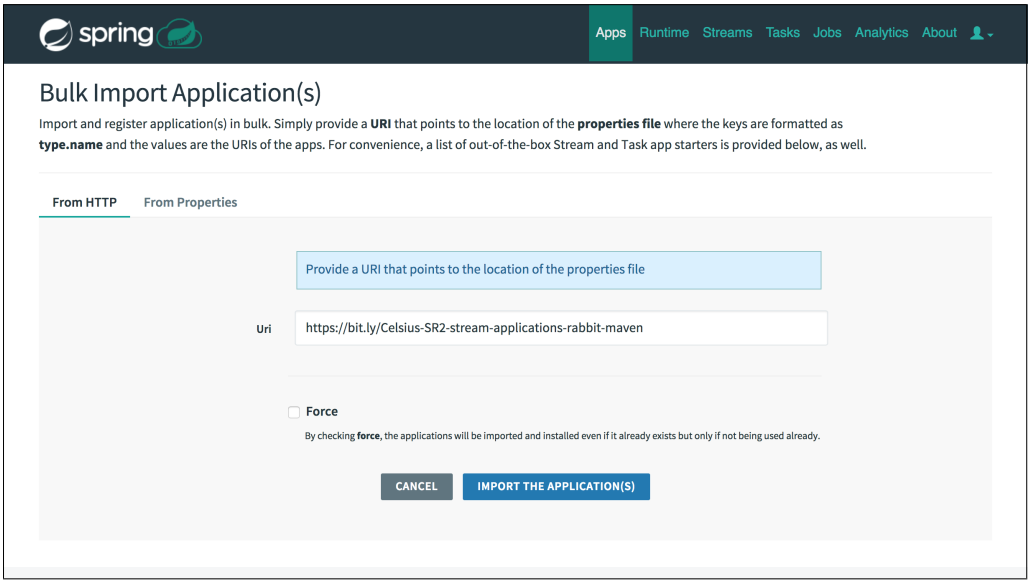
In Apps Manager, visit the Spring Cloud Data Flow service instance's page and click **Manage** to access its dashboard.



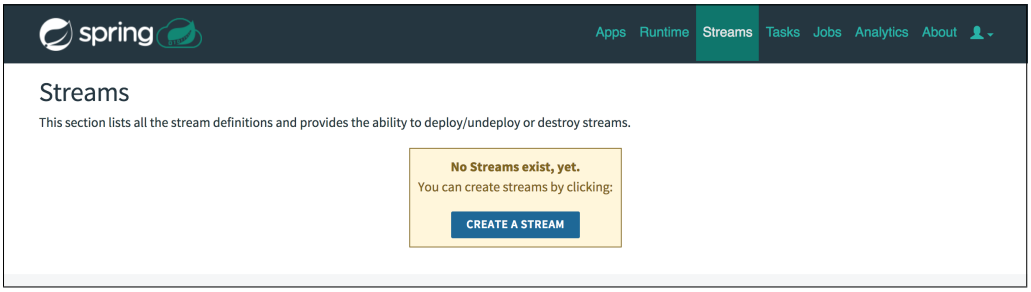
This will take you to the dashboard's **Apps** tab, where you can import applications.



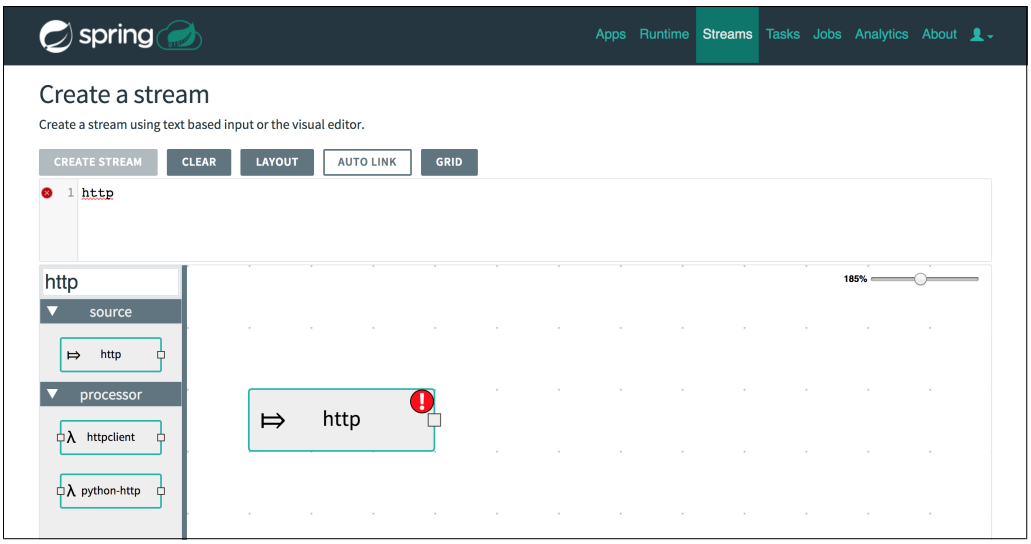
Click **Bulk Import Application(s)** to import the "RabbitMQ + Maven" stream app starters. In the **Uri** field, enter `https://bit.ly/Celsius-SR2-stream-applications-rabbit-maven`. Then click **Import The Application(s)**.



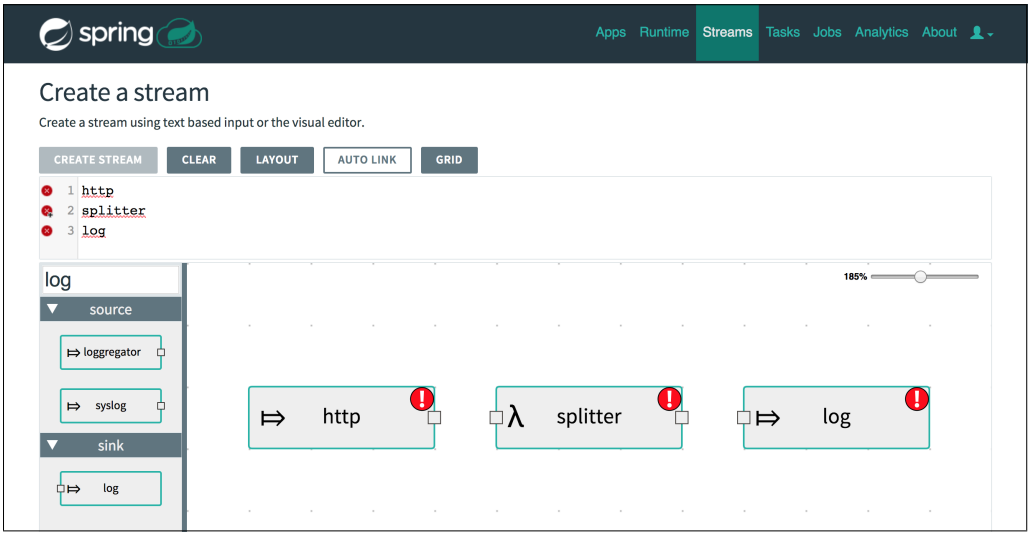
With the app starters imported, visit the **Streams** tab. You can use three of the imported starter applications—the `http` source, the `split` processor, and the `log` sink—to create a stream that consumes data via an HTTP POST request, processes it by splitting it into words, and outputs the results in logs.



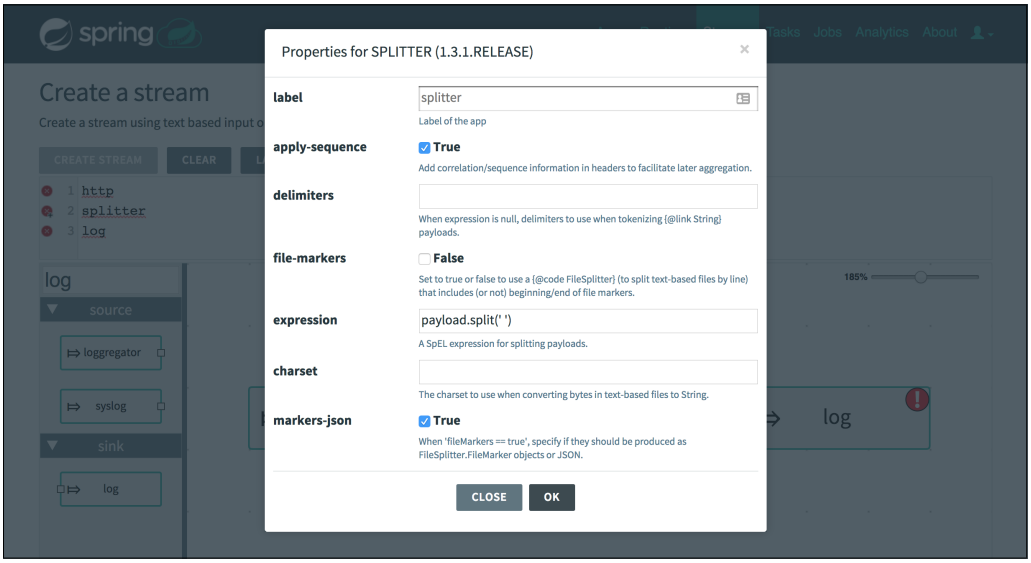
Click **Create A Stream** to enter the stream creation view. In the left sidebar, search for the `http` source application. Click it and drag it onto the canvas to begin defining a stream.



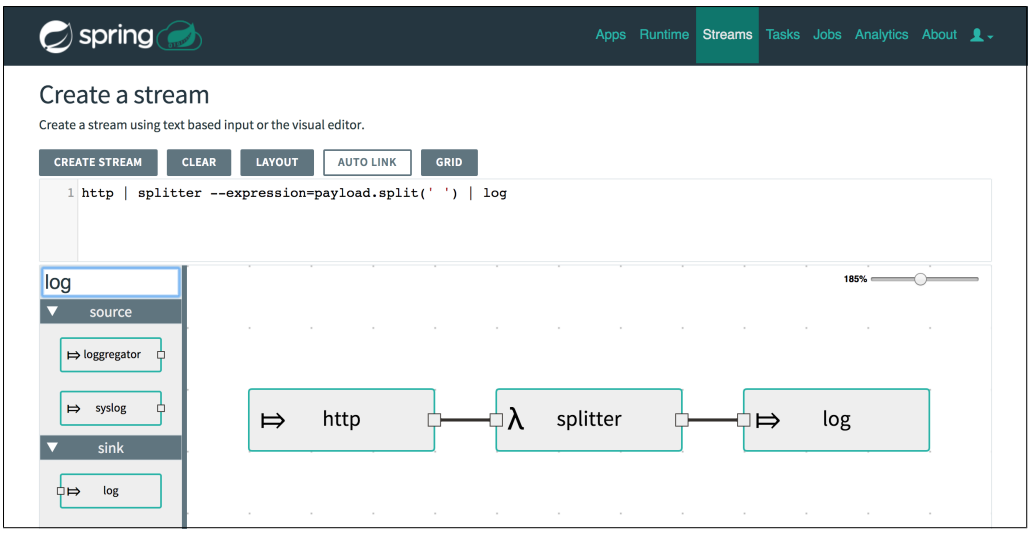
Search for and add the `splitter` processor application and `log` sink application.



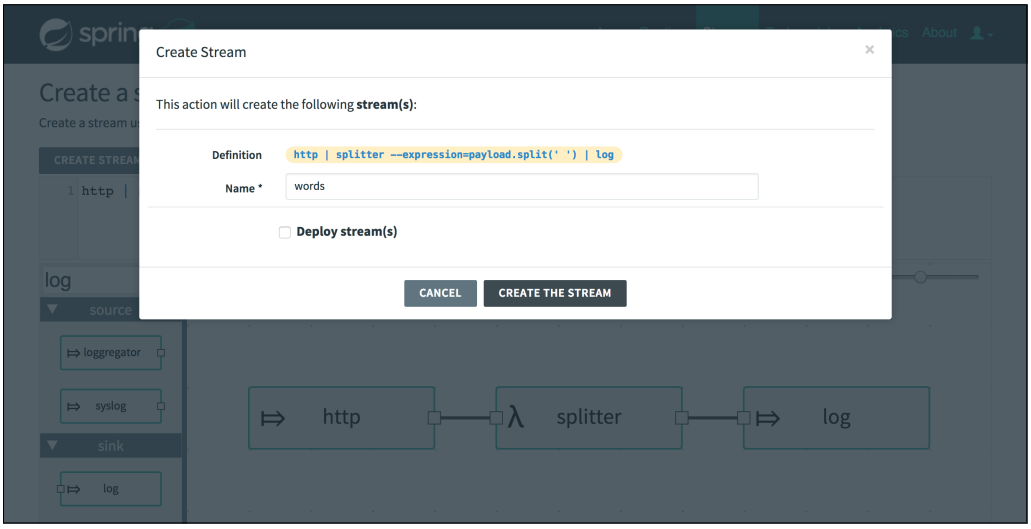
Click the `splitter` application, then click the gear icon beside it to edit its properties. In the **expression** field, enter `payload.split(' ')`. Click **OK**.



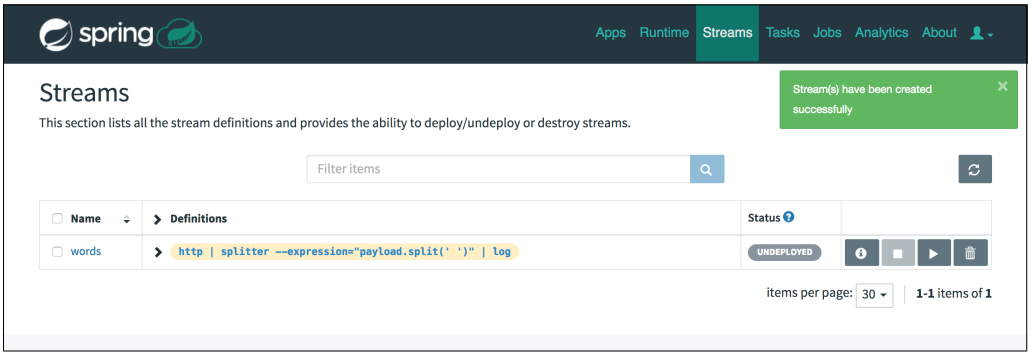
Click and drag between the output and input ports on the applications to connect them and complete the stream.



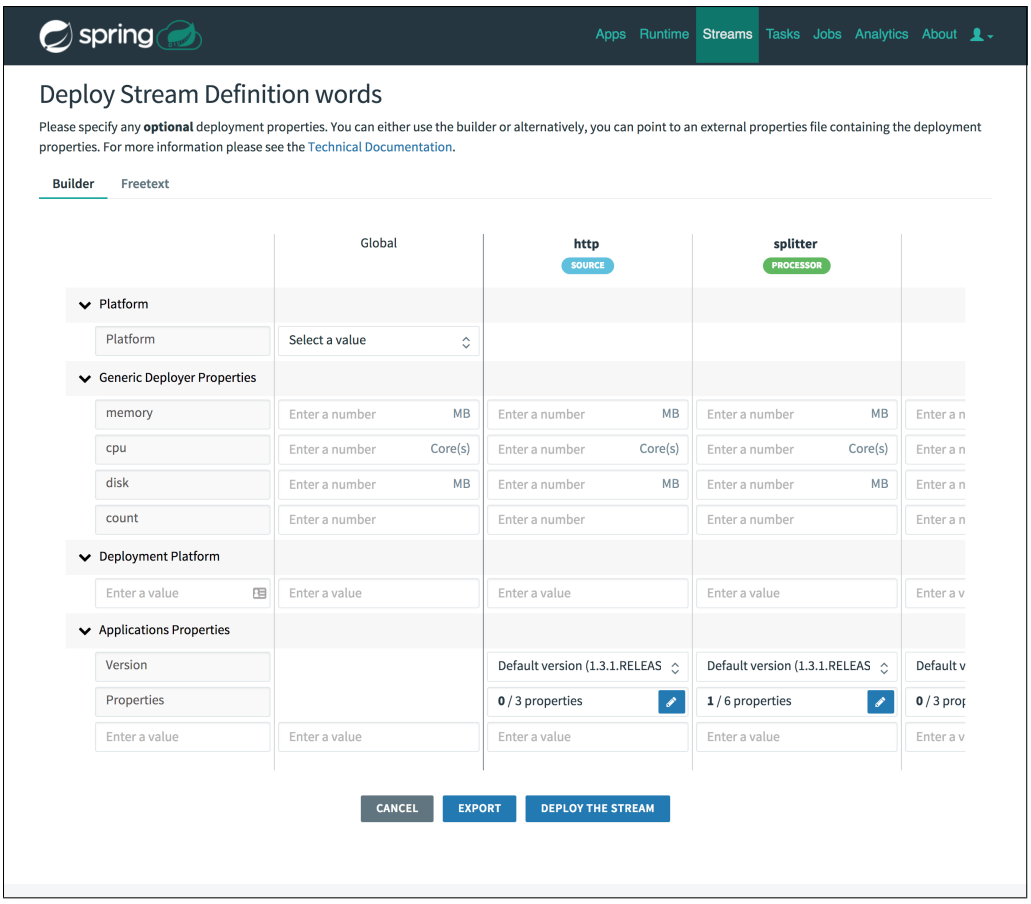
Click the **Create Stream** button. Type the name "words", then click **Create The Stream**.



The **Streams** tab now displays the new stream. Click the ▶ button to deploy the stream.



Click **Deploy The Stream**.



Using the Deployed Data Pipeline

You can run the `cf apps` command to see the applications deployed as part of the stream:

```
$ cf apps
Getting apps in org myorg / space dev as user...
OK

name                      requested state   instances   memory   disk   urls
RWSDZgk-words-http-v1     started           1/1         1G       1G     RWSDZgk-words-http-v1.apps.wise.com
RWSDZgk-words-log-v1      started           1/1         1G       1G     RWSDZgk-words-log-v1.apps.wise.com
RWSDZgk-words-splitter-v1 started           1/1         1G       1G     RWSDZgk-words-splitter-v1.apps.wise.com
```

Run the `cf logs` command on the `RWSDZgk-words-log-v1` application:

```
$ cf logs RWSDZgk-words-log-v1
Retrieving logs for app RWSDZgk-words-log-v1 in org myorg / space dev as user...
```

Then, in a separate command line, use the Data Flow shell (started using the `cf dataflow-shell` command) to send a POST request to the `RWSDZgk-words-http-v1` application:

```
$ cf dataflow-shell dataflow
...
Welcome to the Spring Cloud Data Flow shell. For assistance hit TAB or type "help".
dataflow:>http post --target http://RWSDZgk-words-http-v1.apps.wise.com --data "This is a test"
> POST (text/plain) http://RWSDZgk-words-http-v1.apps.wise.com This is a test
> 202 ACCEPTED
```

Watch for the processed data in the `RWSDZgk-words-log-v1` application's logs:

```
2018-06-07T16:47:08.80-0500 [APP/PROC/WEB/0] OUT 2018-06-07 21:47:08.808  INFO 16 --- [plitter.words-1] RWSDZgk-words-log-v1          : This
2018-06-07T16:47:08.81-0500 [APP/PROC/WEB/0] OUT 2018-06-07 21:47:08.810  INFO 16 --- [plitter.words-1] RWSDZgk-words-log-v1          : is
2018-06-07T16:47:08.82-0500 [APP/PROC/WEB/0] OUT 2018-06-07 21:47:08.820  INFO 16 --- [plitter.words-1] RWSDZgk-words-log-v1          : a
2018-06-07T16:47:08.82-0500 [APP/PROC/WEB/0] OUT 2018-06-07 21:47:08.822  INFO 16 --- [plitter.words-1] RWSDZgk-words-log-v1          : test
```

Managing Service Instances

See below for information about managing Data Flow service instances using the Cloud Foundry Command Line Interface tool (cf CLI). You can also manage Data Flow service instances using Pivotal Cloud Foundry® Apps Manager.

Creating an Instance

Important: If you are using the Redis for PCF product for the Spring Cloud Data Flow analytics store, you cannot create more Data Flow service instances than the setting of the Redis product's Service Instance Limit (the default Service Instance Limit for Redis for PCF is 5). See the Shared-VM Plan section of the Installing and Upgrading Redis for PCF topic in the Redis for PCF documentation for information about configuring this limit.

Begin by targeting the correct org and space.

```
$ cf target -o myorg -s development
api endpoint:   https://api.system.wise.com
api version:    2.75.0
user:           user
org:            myorg
space:          development
```

You can view plan details for the Data Flow product using `cf marketplace -s`.

```
$ cf marketplace
Getting services from marketplace in org myorg / space development as user...
OK

service            plans    description
p-dataflow         standard Deploys Spring Cloud Data Flow servers to orchestrate data pipelines
p-dataflow-mysql   proxy    Proxies to the Spring Cloud Data Flow MySQL service instance
p-dataflow-rabbitmq proxy   Proxies to the Spring Cloud Data Flow RabbitMQ service instance
p-dataflow-redis   proxy    Proxies to the Spring Cloud Data Flow Redis service instance

TIP:  Use 'cf marketplace -s SERVICE' to view descriptions of individual plans of a given service.

$ cf marketplace -s p-dataflow
Getting service plan information for service p-dataflow as user...
OK

service plan   description     free or paid
standard       Standard Plan   free
```

Setting the Buildpack

Each Data Flow service instance can be given the name of a buildpack to use for deploying stream and task apps. You can set the buildpack for the service instance using a `buildpack` parameter given to `cf create-service`. To create a service instance that uses a buildpack named `custom-java-buildpack` to deploy apps, you might run:

```
$ cf create-service p-dataflow standard data-flow -c '{"buildpack": "custom-java-buildpack"}'
```

Setting Dependent Services

Each Data Flow service instance uses three dependent data services. Defaults for these services can be configured in the tile settings, and these defaults can be overridden for each individual service instance at create time.

Note: The service offerings with the plan `proxy` are proxy services used by Spring Cloud Data Flow for PCF service instances. The Spring Cloud Data Flow service broker creates and deletes instances of these services automatically along with each Spring Cloud Data Flow service instance. Do not manually create or delete instances of these services.

General parameters used to configure dependent data services for a Data Flow service instance are listed below.

Parameter	Function
`relational-data-service.name`	The name of the service to use for a relational database that stores Spring Cloud Data Flow metadata and task history.
`relational-data-service.plan`	The name of the service plan to use for the relational database service.
`messaging-data-service.name`	The name of the service to use for a RabbitMQ or Kafka server that facilitates event messaging.
`messaging-data-service.plan`	The name of the service plan to use for the RabbitMQ or Kafka service.
`analytics-data-service.name`	The name of the service to use for a Redis server that stores analytics.
`analytics-data-service.plan`	The name of the service plan to use for the Redis server.
`skipper-relational.name`	The name of the service to use for a relational database used by the Skipper application.
`skipper-relational.plan`	The name of the service plan to use for a relational database used by the Skipper application.

Setting Maven Properties

Each Data Flow service instance can optionally specify Maven configuration properties. For the complete list of properties that can be specified, see the "Maven" section in the OSS Spring Cloud Data Flow documentation.

Maven configuration properties can be set for each Data Flow service instance using parameters given to `cf create-service`. To set the `maven.remote-repositories.repo1.url` property, you might use a command such as the following:

```
$ cf create-service p-dataflow standard data-flow -c '{"maven.remote-repositories.repo1.url": "https://repo.spring.io/libs-snapshot"}'
```

Creating the Instance

Create the service instance using `cf create-service`. To create a Data Flow service that uses a Redis Cloud service available from your PCF marketplace and sets the Maven `maven.remote-repositories.repo1.url` property to `https://repo.spring.io/release`, you might run:

```
$ cf create-service p-dataflow standard data-flow -c '{ "analytics-data-service": { "name": "rediscloud", "plan": "30mb" }, "maven.remote-repositories.repo1.url": "https://repo.spring.io/libs-snaps
Creating service instance data-flow in org myorg / space development as user...
OK

Create in progress. Use 'cf services' or 'cf service data-flow' to check operation status.
```

As the command output suggests, you can use the `cf services` or `cf service` commands to check the status of the service instance. When the service instance is ready, the `cf service` command will give a status of `create succeeded`:

```
$ cf service data-flow

Service instance: data-flow
Service: p-dataflow
Bound apps:
Tags:
Plan: standard
Description: Deploys Spring Cloud Data Flow servers to orchestrate data pipelines
Documentation url: http://cloud.spring.io/spring-cloud-dataflow/
Dashboard: https://p-dataflow.apps.wise.com/instances/f09e5c77-e526-4f49-86d6-721c6b8e2fd9/dashboard

Last Operation
Status: create succeeded
Message: Created
Started: 2017-07-20T18:24:14Z
Updated: 2017-07-20T18:26:17Z
```

Upgrading an Instance

After an upgrade of the Spring Cloud Data Flow for PCF product, you can use the `cf update-service` command to upgrade individual Data Flow service instances.

Begin by targeting the correct org and space.

```
$ cf target -o myorg -s development
api endpoint:   https://api.system.wise.com
api version:    2.75.0
user:           user
org:            myorg
space:          development
```

You can view all service instances in the space using `cf services`.

```
$ cf services
Getting services in org myorg / space development as user...
OK

name                                    service            plan      bound apps  last operation
data-flow                               p-dataflow         standard              create succeeded
mysql-b3e76c87-c5ae-47e4-a83c-5fabf2fc4f11     p-dataflow-mysql     proxy                 create succeeded
rabbitmq-b3e76c87-c5ae-47e4-a83c-5fabf2fc4f11  p-dataflow-rabbitmq  proxy                 create succeeded
redis-b3e76c87-c5ae-47e4-a83c-5fabf2fc4f11     p-dataflow-redis     proxy                 create succeeded
```

Upgrade the Data Flow service instance using `cf update-service`, passing the `-c` flag to set the `upgrade` parameter to `true`.

```
$ cf update-service data-flow -c '{"upgrade": true}'
Updating service instance data-flow as user...
OK

Update in progress. Use 'cf services' or 'cf service data-flow' to check operation status.
```

As the output from the `cf update-service` command suggests, you can use the `cf services` or `cf service` commands to check the status of the service instance. When the Data Flow service instance has been upgraded, the `cf service` command will give a status of `update succeeded`:

```
$ cf service data-flow
Showing info of service data-flow in org myorg / space dev as user...

name:           data-flow
service:        p-dataflow
bound apps:
tags:
plan:           standard
description:    Deploys Spring Cloud Data Flow servers to orchestrate data pipelines
documentation:
dashboard:      https://p-dataflow.apps.wise.com/instances/1cf8ff5b-4a65-469d-bee7-36e6541ac241/dashboard

Showing status of last operation from service data-flow...

status:    update succeeded
message:   Updated
started:   2018-06-19T19:26:09Z
updated:   2018-06-19T19:29:17Z
```

Deleting an Instance

Deleting a Data Flow service instance will result in deletion of all of its dependent service instances.

Begin by targeting the correct org and space.

```
$ cf target -o myorg -s development
api endpoint:   https://api.system.wise.com
api version:    2.75.0
user:           user
org:            myorg
space:          development
```

You can view all service instances in the space using `cf services`.

```
$ cf services
Getting services in org myorg / space development as user...
OK

name                                    service            plan      bound apps  last operation
data-flow                               p-dataflow         standard             create succeeded
mysql-b3e76c87-c5ae-47e4-a83c-5fabf2fc4f11     p-dataflow-mysql     proxy                create succeeded
rabbitmq-b3e76c87-c5ae-47e4-a83c-5fabf2fc4f11  p-dataflow-rabbitmq  proxy                create succeeded
redis-b3e76c87-c5ae-47e4-a83c-5fabf2fc4f11     p-dataflow-redis     proxy                create succeeded
```

Delete the Data Flow service instance using `cf delete-service`. When prompted, enter `y` to confirm the deletion.

```
$ cf delete-service data-flow

Really delete the service data-flow?>y
Deleting service data-flow in org myorg / space development as user...
OK

Delete in progress. Use 'cf services' or 'cf service data-flow' to check operation status.
```

The dependent service instances for the Data Flow server service instance are deleted first, and then the Data Flow server service instance itself is deleted.

As the output from the `cf delete-service` command suggests, you can use the `cf services` or `cf service` commands to check the status of the service instance.

When the Data Flow service instance and its dependent service instances have been deleted, the `cf services` command will no longer list the service instance:

```
$ cf services
Getting services in org myorg / space development as user...
OK

No services found
```

Using User-Provided Service Instances

By default, a Spring Cloud Data Flow service instance uses MySQL for PCF for its backing relational database service, RabbitMQ for PCF for its backing messaging service, and Redis for PCF for its backing analytics service. When creating a Spring Cloud Data Flow service instance, you can configure it to use Cloud Foundry user-provided services for its dependent data services instead. See below for information about configuring a Data Flow service instance to use an alternative dependent data service.

Dependent Service Parameters

You can specify that a Data Flow service instance should use a user-provided service instance by passing configuration parameters to the `cf create-service` command in JSON using the `-c` flag. The parameters used to configure alternative dependent data services for a Data Flow service instance are listed below.

Parameter	Function
`relational-data-service.user-provided`	A JSON object containing connection information for the relational database service used to store task history.
`messaging-data-service.user-provided`	A JSON object containing connection information for the messaging service used for event messaging.
`analytics-data-service.user-provided`	A JSON object containing connection information for the Redis service used for an analytics store.

See the following sections for information about using these parameters.

User-Provided Relational Database Service

You can use the `relational-data-service` parameter to supply the configuration for a user-provided relational database service. This parameter contains a JSON object `user-provided`, with fields for each of the connection values needed to provision a Cloud Foundry user-provided relational database service instance.

The relational database service can be the default MySQL for PCF v2 service or any other relational database service. It must provide a relational database.

An example of fields contained in the `user-provided` object for a relational database service is shown below.

```
{
 "uri":"mysql://kempercrabb:fake-password@123.234.456.879:3306/vigil",
 "username": "kempercrabb",
 "password": "fake-password",
 "dbname": "vigil",
 "host": "123.234.456.879",
 "port": 3306,
 "tags": ["mysql"]
}
```

To create a Spring Cloud Data Flow service instance using these connection values for the service instance's relational database service instance:

```
$ cf create-service p-dataflow standard data-flow -c '{"relational-data-service": { "user-provided": {"uri":"mysql://kempercrabb:fake-password@123.234.456.879:3306/vigil","username":"kempe
```

User-Provided Messaging Service

You can use the `messaging-data-service` parameter to supply the configuration for a user-provided messaging service. This parameter contains a JSON object `user-provided`, with fields for each of the connection values needed to provision a Cloud Foundry user-provided messaging service instance.

The messaging data service can be the default RabbitMQ for PCF service or another messaging service. It must provide either a RabbitMQ server or a Kafka server.

An example of fields contained in the `user-provided` object for a messaging service is shown below.

```
{
 "dashboard_url": "https://api.cloudamqp.com/console/a-GUID-would-go-here/details",
 "username": "buckstorm",
 "vhost": "buckstorm",
 "password": "fake-password",
 "ssl": false,
 "hostname": "stonehill.rmq.cloudamqp.com",
 "uri": "amqp://buckstorm:fake-password@stonehill.rmq.cloudamqp.com/buckstorm",
 "http_api_uri": "http://buckstorm:fake-password@stonehill.rmq.cloudamqp.com:1883/api",
 "tags": ["rabbitmq"]
}
```

To create a Spring Cloud Data Flow service instance using these connection values for the service instance's messaging service instance:

```
$ cf create-service p-dataflow standard data-flow -c '{"messaging-data-service": { "user-provided": {"dashboard_url":"https://api.cloudamqp.com/console/a-GUID-would-go-here/details","usern
```

User-Provided Analytics Service

You can use the `analytics-data-service` parameter to supply the configuration for a user-provided analytics service. This parameter contains a JSON object `user-provided`, with fields for each of the connection values needed to provision a Cloud Foundry user-provided Redis service instance.

The analytics service can be the default Redis for PCF service or any other Redis service. It must provide a Redis database.

An example of fields contained in the `user-provided` object for an analytics service is shown below.

```
{
 "host": "garrels.gce.cloud.redislabs.com",
 "uri": "redis://:fake-password@garrels.gce.cloud.redislabs.com:11781/boars-head",
 "port": 11781,
 "dbname": "boars-head",
 "password": "fake-password",
 "tags": ["redis"]
}
```

To create a Spring Cloud Data Flow service instance using these connection values for the service instance's analytics service instance:

```
$ cf create-service p-dataflow standard data-flow -c '{"analytics-data-service": { "user-provided": {"host":"garrels.gce.cloud.redislabs.com","uri":"redis://:fake-password@garrels.gce.cloud.redisl
```

Viewing Service Instance Logs

Spring Cloud Data Flow for PCF provides access to the logs generated by each Data Flow server service instance, including logs for each of the three backing applications (Data Flow server application, metrics application, and Skipper application) for each instance. You can view these logs either using the Service Instance Logs cf CLI plugin or by visiting the dashboard of the Spring Cloud Data Flow service broker.

Using the cf CLI Plugin

After installing the Service Instance Logs cf CLI plugin (see the instructions in the Installing section of the plugin's README), you can use the `service-logs` command to tail logs or dump recent logs for a service instance.

To tail logs for a Data Flow service instance, run `cf service-logs SERVICE_NAME`, where `SERVICE_NAME` is the name of the service instance:

```
$ cf service-logs data-flow
```

To dump recent logs for the instance, use the `--recent` flag:

```
$ cf service-logs --recent data-flow
```

If your Pivotal Cloud Foundry deployment uses a self-signed certificate, you must use the `--skip-ssl-validation` flag to disable the default validation of the platform's SSL certificate:

```
$ cf service-logs --skip-ssl-validation data-flow
```

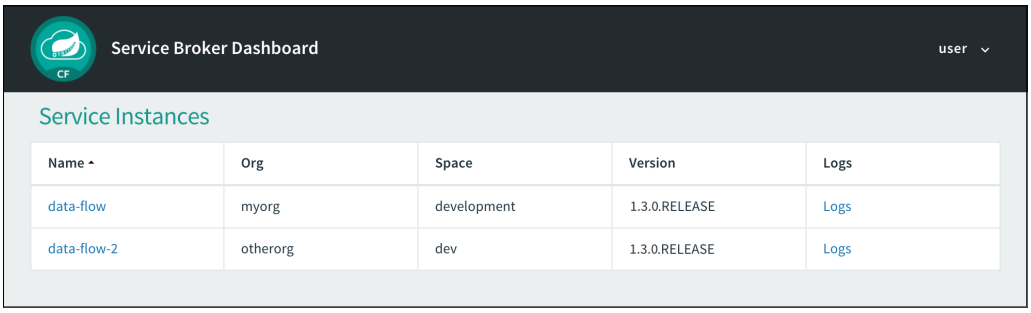
Using the Service Broker Dashboard

Note: To access the service broker dashboard, you must be a Space Developer in the broker application's space (this is typically the `system` org and `p-dataflow` space).

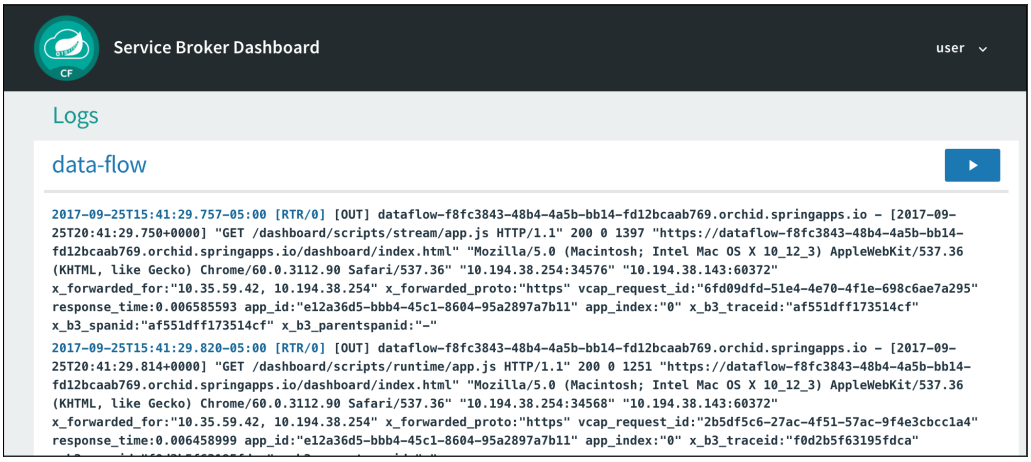
Visit the Spring Cloud Data Flow service broker's dashboard. You can access it at the following URL, where `apps.wise.com` is the application domain of your PCF deployment:

```
https://p-dataflow.apps.wise.com/
```

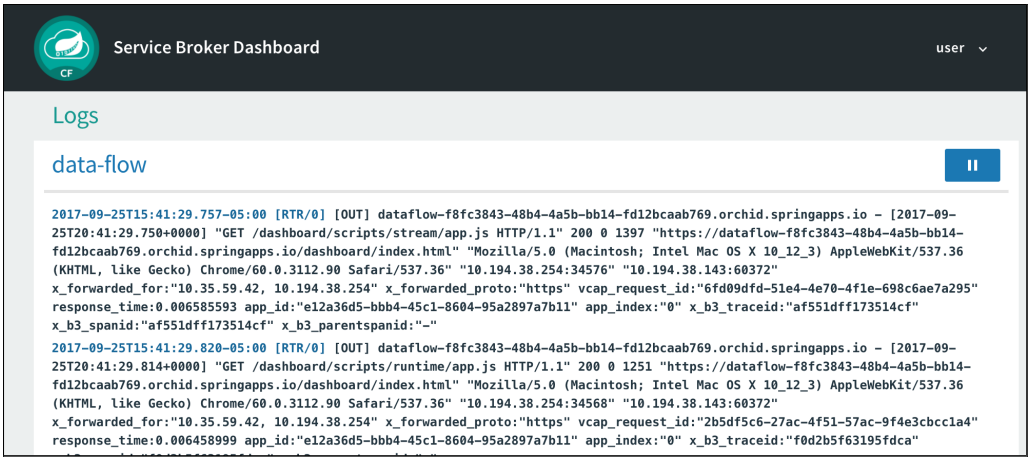
The dashboard shows the name, org, and space of each service instance, as well as a link to view logs for the instance.



Click the **Logs** link to view logs for a particular service instance's backing application.



You can stream current logs for the instance by clicking the ▶ button.



Reading Aggregated Logs

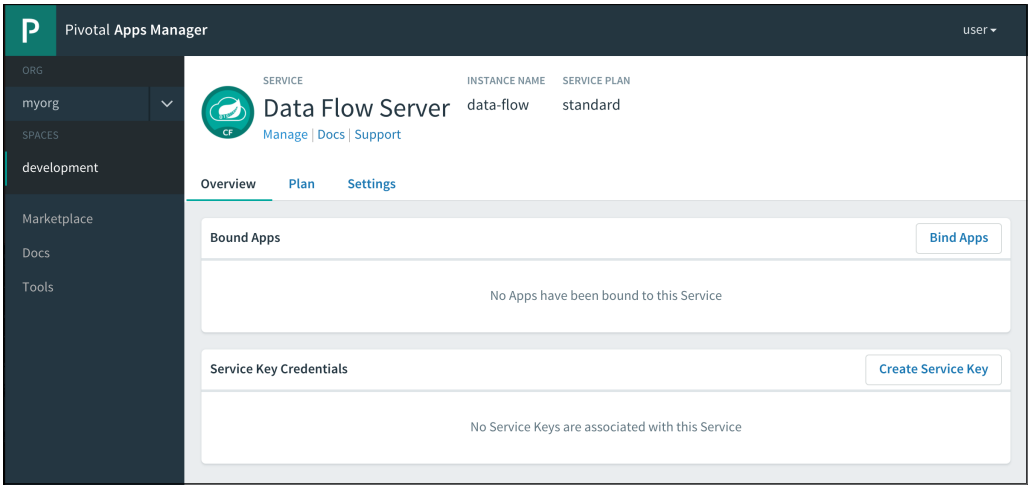
The logs retrieved by the Service Instance Logs cf CLI plugin aggregate logs from three backing applications: a Spring Cloud Data Flow server application, a metrics application, and a Spring Cloud Skipper application. The following excerpt shows logs after deploying a stream:

```
2018-02-09T11:12:02.45-0600 [RTR/dataflow 0] OUT dataflow-11f71dd3-f902-4e31-b631-fbbbfdb82459.apps.americancanyon.cf-app.com - [2018-02-09T17:12:02.429+0000] "GET /metrics/streams?names=httptest HTTP/1.1" 200 0 14765 "https://dataflow-11f71dd3-f9(

2018-02-09T11:12:02.45-0600 [RTR/metrics 0] OUT df-metrics-11f71dd3-f902-4e31-b631-fbbbfdb82459.apps.americancanyon.cf-app.com - [2018-02-09T17:12:02.444+0000] "GET /collector/metrics/streams HTTP/1.1" 200 0 22261 "-" "Apache-HttpClient/4.5.3 (Java/1

2018-02-09T11:12:02.60-0600 [APP/PROC/WEB/skipper 0] OUT 2018-02-09 17:12:02.603  INFO 15 --- [ry-client-nio-3] o.s.c.d.s.c.CloudFoundryAppDeployer     : Successfully computed status [deployed] for httptest-http-v1
2018-02-09T11:12:02.94-0600 [APP/PROC/WEB/skipper 0] OUT 2018-02-09 17:12:02.940  INFO 15 --- [ry-client-nio-3] o.s.c.d.s.c.CloudFoundryAppDeployer     : Successfully computed status [deployed] for astream-log-v4
2018-02-09T11:12:03.06-0600 [APP/PROC/WEB/skipper 0] OUT 2018-02-09 17:12:03.068  INFO 15 --- [ry-client-nio-1] o.s.c.d.s.c.CloudFoundryAppDeployer     : Successfully computed status [deployed] for astream-time-v4
```

The Data Flow server application's logs are identified as belonging to the `dataflow` application. The metrics application similarly is called `metrics`. The Spring Cloud Skipper application is called `skipper`.

Using the Dashboard

To find the dashboard, navigate in Pivotal Cloud Foundry® Apps Manager to the Data Flow service instance's space, click the listing for the service instance, and then click **Manage**.



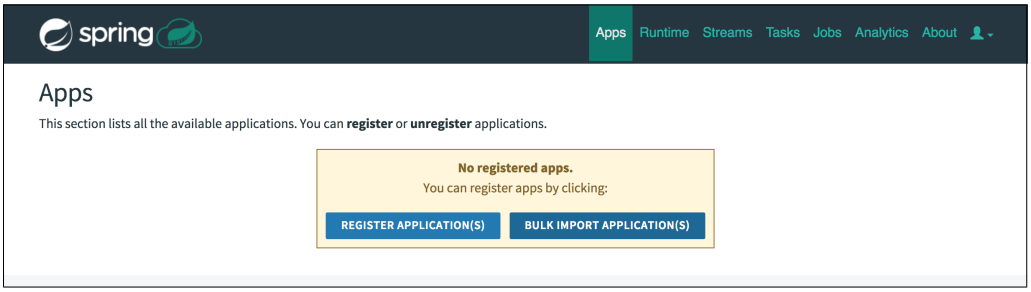
If you are using version 6.8.0 or later of the Cloud Foundry Command Line Interface tool (cf CLI), you can also use `cf service SERVICE_NAME`, where `SERVICE_NAME` is the name of the Data Flow service instance:

```
$ cf service data-flow
Showing info of service data-flow in org myorg / space dev as user...

name:          data-flow
service:       p-dataflow
bound apps:
tags:
plan:          standard
description:   Deploys Spring Cloud Data Flow servers to orchestrate data pipelines
documentation:
dashboard:     https://p-dataflow.apps.wise.com/instances/2c9a66d3-99d8-458c-860d-01e7ac64eb1d/dashboard

Showing status of last operation from service data-flow...

status:    create succeeded
message:   Created
started:   2018-05-17T19:02:04Z
updated:   2018-05-17T19:08:14Z
```

Visit the URL given for "Dashboard".

Dashboard Information

The dashboard provides an overview of registered applications, stream and task definitions, and batch jobs. It also provides controls for deploying streams, launching tasks, and restarting batch job executions.



For complete information about the dashboard and its provided functionality, see the OSS Spring Cloud Data Flow project's documentation.

Using the Shell

The open-source Spring Cloud Data Flow project provides a shell, which can be used to interact locally with a Data Flow service instance deployed on Pivotal Cloud Foundry (PCF).

You can use the shell with Spring Cloud Data Flow for PCF service instances in either of two ways:

- The Spring Cloud Data Flow for PCF Cloud Foundry CLI plugin (recommended)
- The open-source shell binary, with manually-configured command-line options

The Spring Cloud Data Flow for PCF Cloud Foundry CLI plugin was created to ease the use of the Spring Cloud Data Flow shell with Spring Cloud Data Flow service instances on PCF.

Note: To run the Spring Cloud Data Flow shell, you must have a Java Runtime Environment (JRE) installed. You can download the JRE from the Java website.

Note: Before connecting to a Data Flow service instance with the Spring Cloud Data Flow shell, be sure to log in to the PCF deployment using the Cloud Foundry Command Line Interface tool (cf CLI) and target the org and space of the Data Flow service instance. The Data Flow shell uses the cf CLI to authenticate to PCF.

Using the Cloud Foundry CLI Plugin

The plugin adds a `dataflow-shell` command, which attaches the Data Flow shell to a Spring Cloud Data Flow for PCF service instance. Given an existing service instance named `data-flow`, the following command will download the appropriate shell version and attach it to the service instance:

```
$ cf dataflow-shell data-flow
Attaching shell to dataflow service data-flow in org myorg / space dev as user...
Launching dataflow shell JAR
  ____                 ____ _               __
 / ___|_ __  _ __(_)_ __  / ___|___ _   _ __ |
 \___ \| '_ \| '__| | '_ \/ _` | |  | |/ _ \| |/ _` |
  ___) | |_) | |  | | | | (_| | |___| | (_) | | (_| |
 |____/| .__/|_|  |_|_| |_|\__,_| \____|_|\___/ \__,_|\__,_|
  ____|_|   _       __|___/                 __________
 | _ \ __ _| __ _ | ___|____      __ \\\\\\
 | | | |/ _` |/ _` | |_ |/ _ \\ \ /\ / // \\\\\\
 | |_| | (_| | (_| | _| | (_) \ V  V /  //////
 |____/ \__,_|\__,_|_|  |_|\___/ \_/\_/  /_/_/_/_/

1.5.0.RELEASE

Welcome to the Spring Cloud Data Flow shell. For assistance hit TAB or type "help".
dataflow:>
```

For more information about the Spring Cloud Data Flow shell, see its documentation.

Using the Shell Manually

If you would like to download and configure the Data Flow shell manually, download the shell JAR file from the Spring Releases Maven repository. Spring Cloud Data Flow for PCF can be used with version 1.5.0.RELEASE or later of the Spring Cloud Data Flow shell.

To target a Data Flow service instance's server with the shell, you must obtain the server's URL. Run `cf service SERVICE_NAME`, where `SERVICE_NAME` is the name of the service instance:

```
$ cf service data-flow

Service instance: data-flow
Showing info of service data-flow in org myorg / space dev as user...

name:           data-flow
service:        p-dataflow
bound apps:
tags:
plan:           standard
description:    Deploys Spring Cloud Data Flow servers to orchestrate data pipelines
documentation:
dashboard:      https://p-dataflow.apps.wise.com/instances/2f6ec0c6-c828-45bb-905a-4779ce50552b/dashboard

Showing status of last operation from service data-flow...

status:    create succeeded
message:   Created
started:   2018-05-17T14:43:38Z
updated:   2018-05-17T14:49:43Z
```

Visit the URL given for "Dashboard" and authenticate using your PCF credentials. When redirected to the service instance's dashboard, copy the dashboard's domain name (the URL minus the path following the domain).

For example, given the following URL:

```
https://dataflow-2f6ec0c6-c828-45bb-905a-4779ce50552b.apps.wise.com/dashboard/#/apps
```

Copy the following:

```
https://dataflow-2f6ec0c6-c828-45bb-905a-4779ce50552b.apps.wise.com
```

From the directory containing the shell JAR file, run the shell from the command line using a command as shown below.

```
$ java -jar JAR_NAME SKIP_VALIDATION --dataflow.uri=SERVER_URL --dataflow.credentials-provider-command="cf oauth-token" --dataflow.mode=skipper
```

In this command, replace the following placeholders as shown below.

- `JAR_NAME` with the name of the downloaded Spring Cloud Data Flow shell JAR file
- `SKIP_VALIDATION` with the flag and value `--dataflow.skip-ssl-validation=true` if your PCF installation is using self-signed SSL certificates; otherwise remove
- `SERVER_URL` with the Data Flow service instance's server URL (the domain copied above)

The complete command may look something like the following:

```
$ java -jar spring-cloud-dataflow-shell-1.5.0.RELEASE.jar \
--dataflow.skip-ssl-validation=true \
--dataflow.uri=https://dataflow-2f6ec0c6-c828-45bb-905a-4779ce50552b.apps.wise.com \
--dataflow.credentials-provider-command="cf oauth-token" --dataflow.mode=skipper
```

When you run the `java -jar` command, the shell will initialize and give a `dataflow:>` prompt.

```
Welcome to the Spring Cloud Data Flow shell. For assistance hit TAB or type "help".
dataflow:>
```

For a list of available shell commands, run `help` from within the shell. For information about the flags which can be provided to the shell on startup, see the OSS Spring Cloud Data Flow project's documentation.