Spring Cloud Services

Spring Cloud Services (aka SCS)

A Pivotal Cloud Foundry *managed service* for deploying Spring Cloud infrastructure services on-demand in the cloud

https://docs.pivotal.io/spring-cloud-services

What are managed services?

- Cloud Foundry provides a Service Broker API which provides an mechanism for extending Cloud Foundry to support the *automatic provisioning of backing services in the cloud*, and the automatic binding these services to applications

- A managed service is an implementation of the Service Broker API for a specific type of backing service. Examples include databases, message brokers, cloud services, autoscaling services, monitoring services, etc..

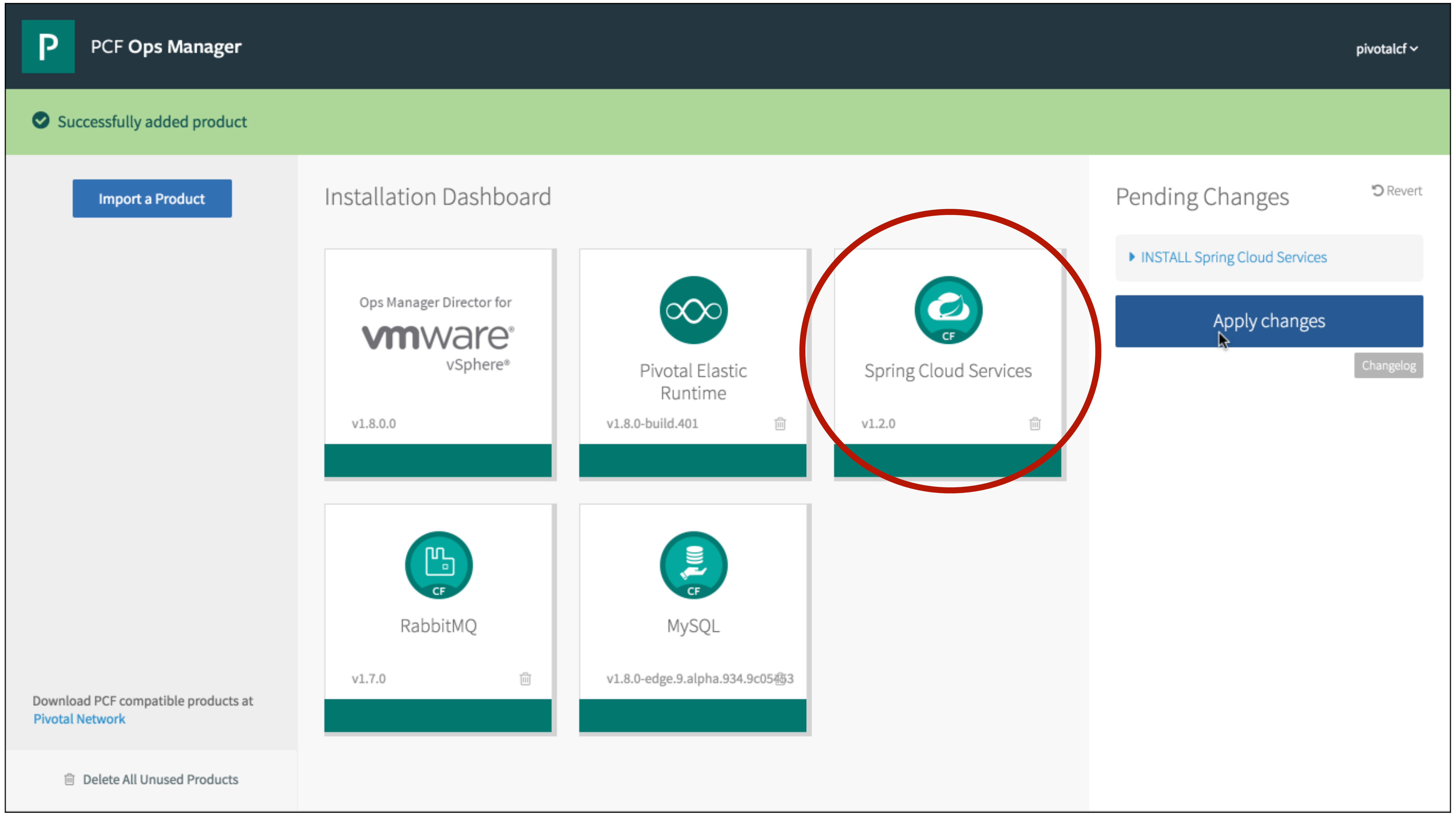
What Services are supported?

- Spring Cloud Config Server
- Eureka Service Registry
- Hystrix Circuit Breaker Dashboard (with Turbine Metrics Aggregation)

Installation

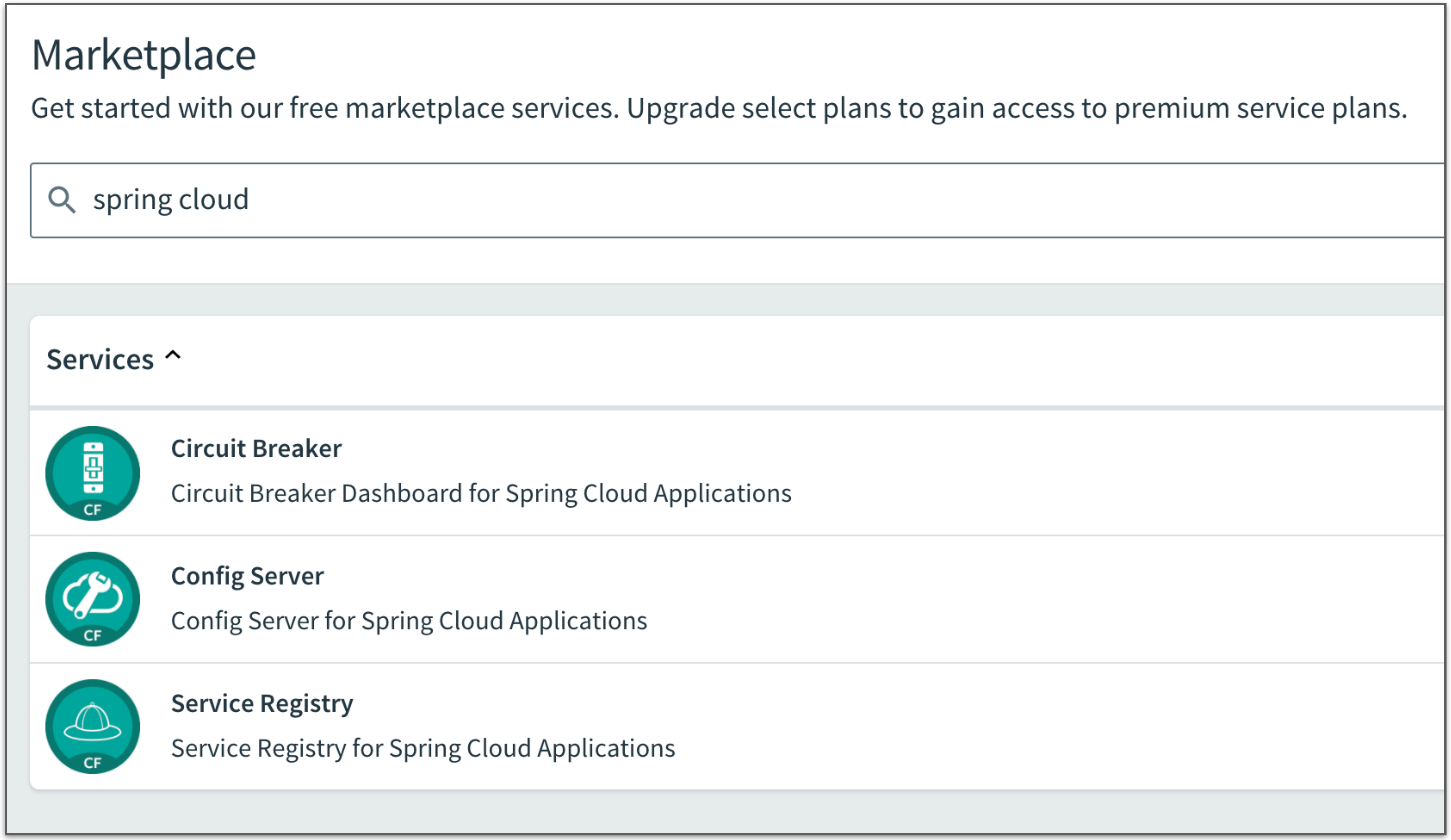
The Spring Cloud Services product is packaged as a pivotal Cloud Foundry “tile", installed by a PCF administrator as a Cloud Foundry extension

The installation usually involves the deployment of the service broker and registration of services into the PCF marketplace


PCF Ops Manager
pivotalcf
Successfully added product
Import a Product
Installation Dashboard
Ops Manager Director for vmware vSphere
v1.8.0.0
Pivotal Elastic Runtime
v1.8.0-build.401
Spring Cloud Services
v1.2.0
RabbitMQ
v1.7.0
MySQL
v1.8.0-edge.9.alpha.934.9c05453
Pending Changes
Revert
INSTALL Spring Cloud Services
Apply changes
Changelog
Download PCF compatible products at Pivotal Network
Delete All Unused Products

PCF Operations Manager

Can verify if a PCF instance has the Spring Cloud Services installed by looking for the presence of these services in the Cloud Foundry Marketplace



Marketplace

Get started with our free marketplace services. Upgrade select plans to gain access to premium service plans.

Circuit Breaker
Circuit Breaker Dashboard for Spring Cloud Applications

Config Server
Config Server for Spring Cloud Applications

Service Registry
Service Registry for Spring Cloud Applications

cf commands

PCF provides standard commands for provisioning any marketplace service, as follows:

```
cf create-service <service-name> <plan-name>
   <service-instance-name> [-c <optional-configuration>]
```

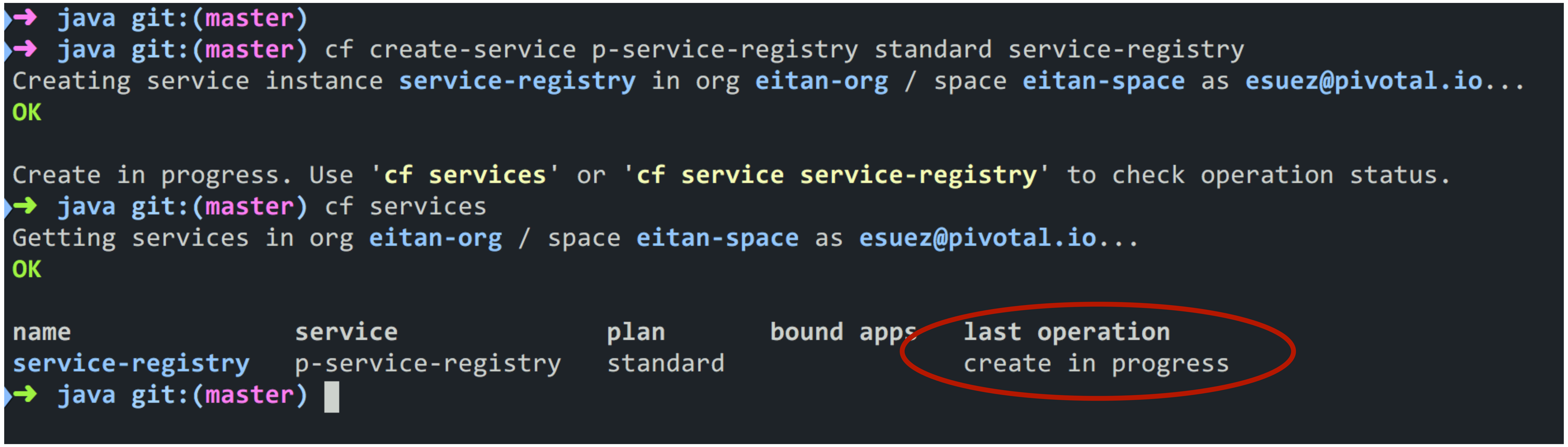
See the command *cf help create-service* for more information

Related commands:

- cf delete-service
- cf bind-service
- cf unbind-service

Asynchronous Provisioning

- The Spring Cloud Services are provisioned asynchronously
- It usually takes a couple of minutes until the service is deployed and ready
- After service creation, the provisioning status can be obtained with the command *cf services*, for example:



```
➜ java git:(master)
➜ java git:(master) cf create-service p-service-registry standard service-registry
Creating service instance service-registry in org eitan-org / space eitan-space as esuez@pivotal.io...
OK

Create in progress. Use 'cf services' or 'cf service service-registry' to check operation status.
➜ java git:(master) cf services
Getting services in org eitan-org / space eitan-space as esuez@pivotal.io...
OK

name               service              plan       bound apps   last operation
service-registry   p-service-registry   standard                create in progress
➜ java git:(master)
```

Provisioning

- Spring Cloud Services are created *on-demand*, in other words, "from scratch" when the *cf create-service* command is invoked.
- The services are deployed in the same way that one would deploy an application to Cloud Foundry: with a *cf push* command.
- The deployed services reside in a special PCF “org” and “space”.

cf plugin for SCS

A plugin is available for the cf CLI, that provides the following commands:

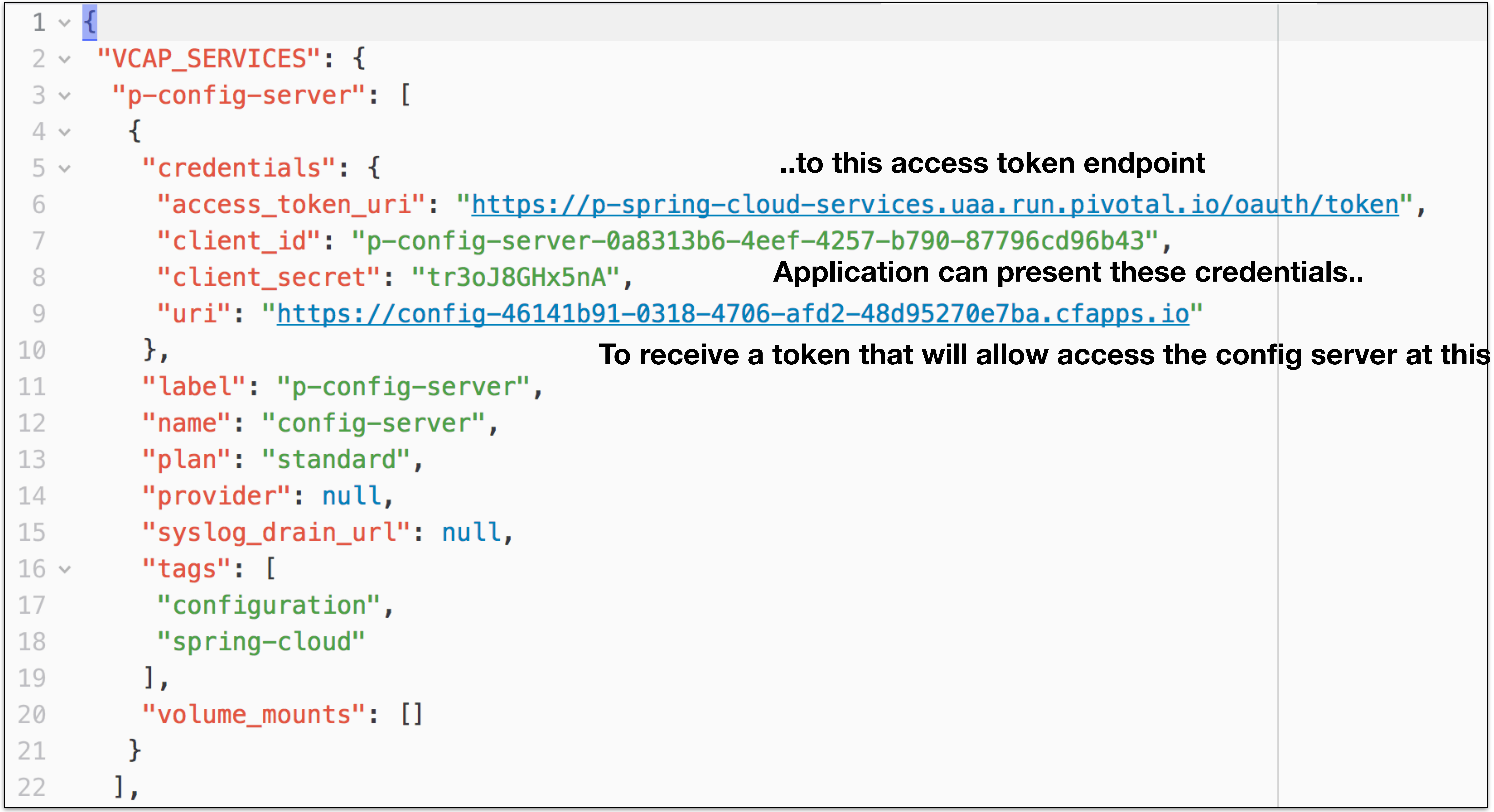
```
config-server-encrypt-value, csev        Encrypt a string using a Spring Cloud Services configuration server
service-registry-deregister, srdr        Deregister an application registered with a Spring Cloud Services service registry
service-registry-disable, srda           Disable an application registered with a Spring Cloud Services service registry so that it is unavailable for traffic
service-registry-enable, sren            Enable an application registered with a Spring Cloud Services service registry so that it is available for traffic
service-registry-info, sri               Display Spring Cloud Services service registry instance information
service-registry-list, srl               Display all applications registered with a Spring Cloud Services service registry
spring-cloud-service-restage, scs-restage   Restage a Spring Cloud Services service instance
spring-cloud-service-restart, scs-restart   Restart a Spring Cloud Services service instance
spring-cloud-service-start, scs-start    Start a Spring Cloud Services service instance
spring-cloud-service-stop, scs-stop      Stop a Spring Cloud Services service instance
spring-cloud-service-view, scs-view      Display health and status for a Spring Cloud Services service instance
```

These commands can be useful for analysis and troubleshooting of provisioned services.

https://plugins.cloudfoundry.org/

Binding

- When binding an application to a Spring Cloud service, the service broker takes care to setup a secure channel for consuming the service
- The service will not accept https requests that do not bear an OAuth2 access token from an authorized client application
- Spring Cloud Services leverages Pivotal Cloud Foundry's OAuth2 server (the UAA) to generate credentials for the application on-demand
- These credentials are communicated via the environment variable VCAP_SERVICES to the consuming application
- This mechanism obviates the need to configure the consumer application with the URL and access credentials to the backing service



```
{
 "VCAP_SERVICES": {
  "p-config-server": [
   {
    "credentials": {
     "access_token_uri": "https://p-spring-cloud-services.uaa.run.pivotal.io/oauth/token",
     "client_id": "p-config-server-0a8313b6-4eef-4257-b790-87796cd96b43",
     "client_secret": "tr3oJ8GHx5nA",
     "uri": "https://config-46141b91-0318-4706-afd2-48d95270e7ba.cfapps.io"
    },
    "label": "p-config-server",
    "name": "config-server",
    "plan": "standard",
    "provider": null,
    "syslog_drain_url": null,
    "tags": [
     "configuration",
     "spring-cloud"
    ],
    "volume_mounts": []
   }
  ],
```

..to this access token endpoint

Application can present these credentials..

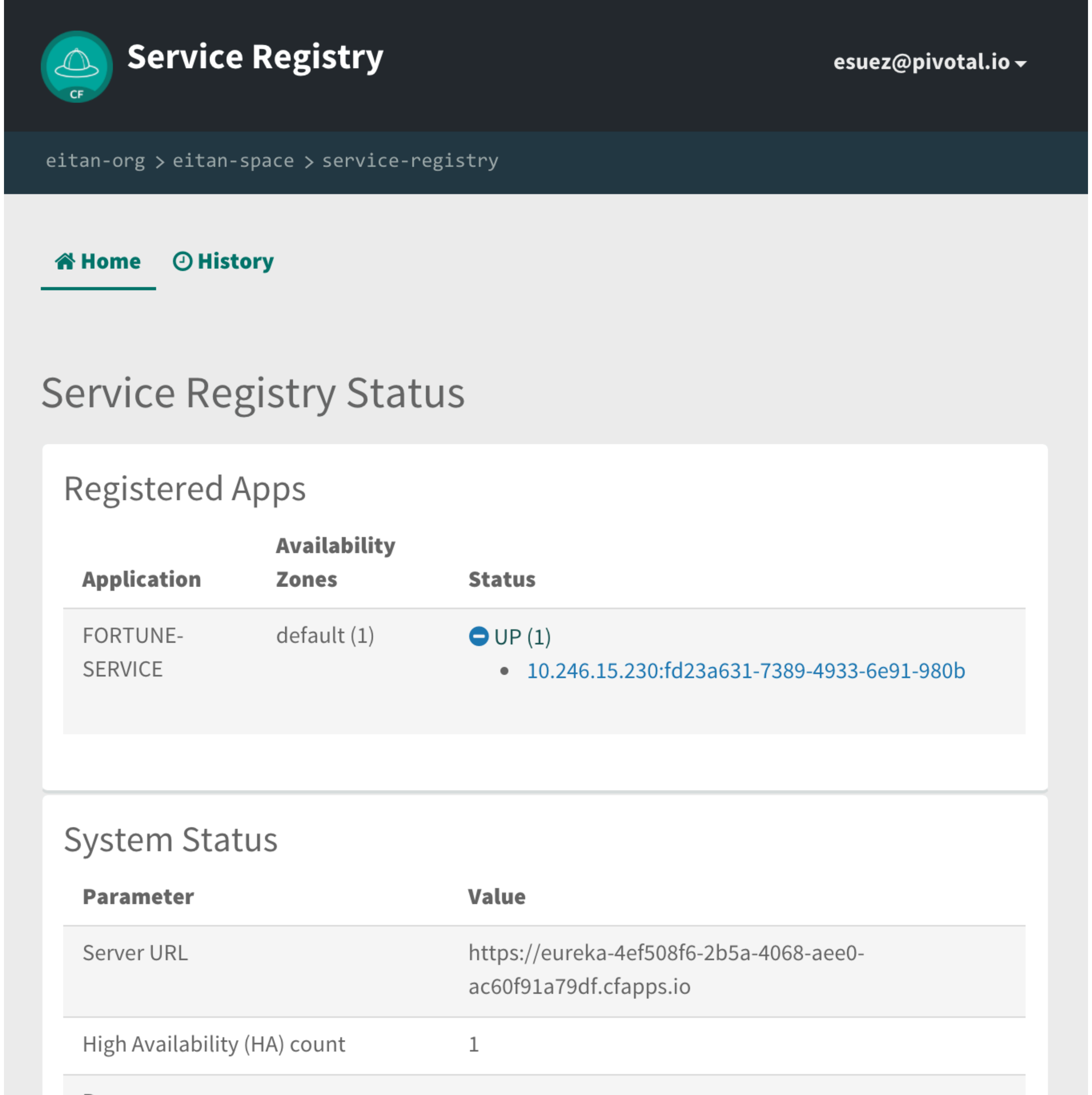
To receive a token that will allow access the config server at this url

Provisioning a Service Registry

Example:

```
cf create-service p-service-registry standard service-registry
```

After service has been provisioned, the service registry dashboard is accessible directly from the Pivotal Cloud Foundry *Apps Manager*



Service Registry Status

Registered Apps

Application	Availability Zones	Status
FORTUNE-SERVICE	default (1)	UP (1) • 10.246.15.230:fd23a631-7389-4933-6e91-980b

System Status

Parameter	Value
Server URL	https://eureka-4ef508f6-2b5a-4068-aee0-ac60f91a79df.cfapps.io
High Availability (HA) count	1

Configuration Options: Service Registry

- *count*: number of instances to provision
- *peers*: a mechanism to stand up a cluster of eureka instances that span more than one PCF space, org, or even a separate PCF instance running in a different data center

Provisioning a Config Server

- Example:

```
cf create-service p-config-server standard config-server
    -c config.json
```

- Example minmal *config.json* file contents:

```
{ "git": { "uri": "https://github.com/<username>/config-repo.git" } }
```

Configuration Options: Config Server

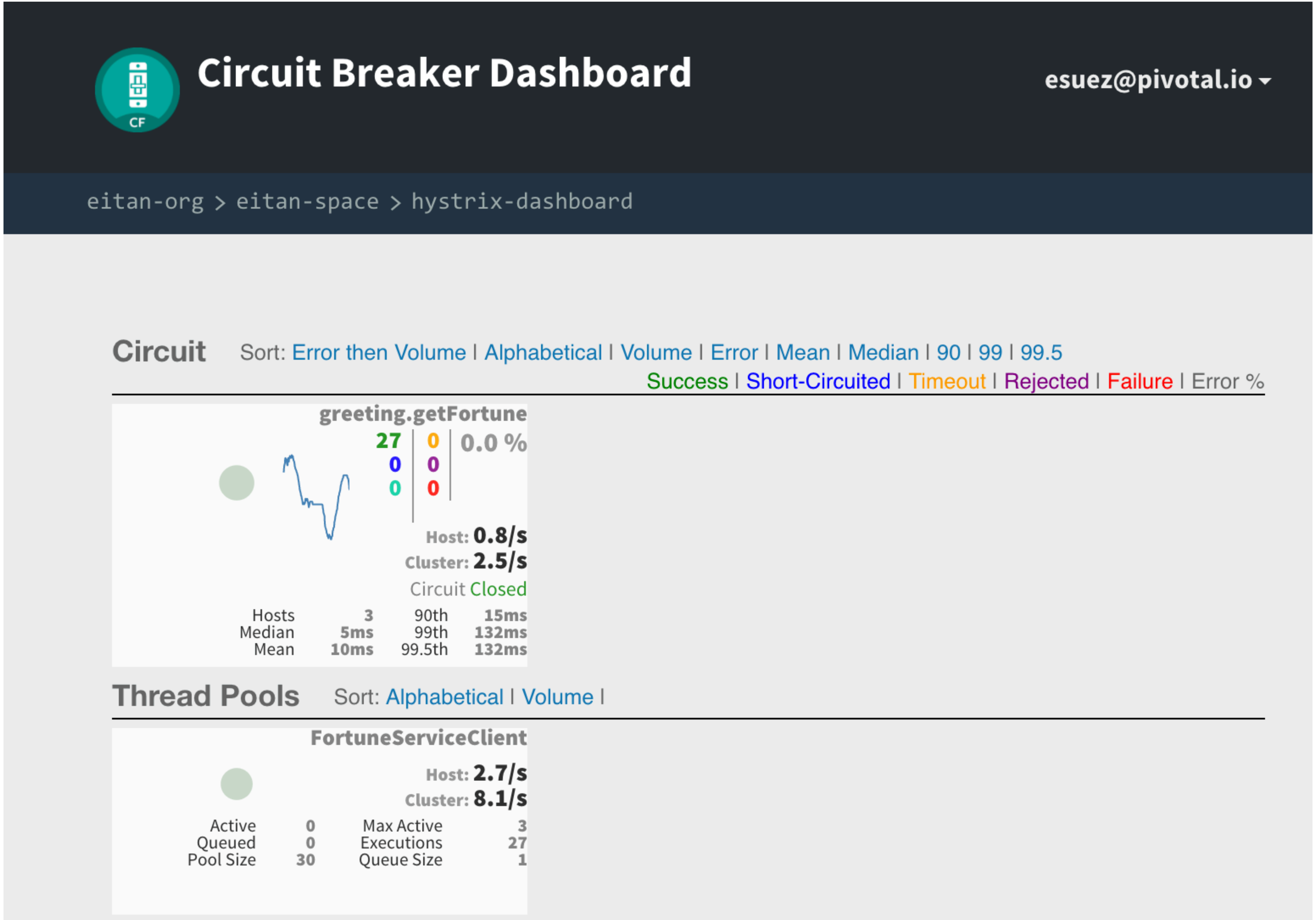
- *count*: number of instances to provision
- Supports multiple types of back-ends: git, vault.
- For git, supports repository access over http or ssh
- Also supports multiple source repositories

Provisioning a Hystrix Dashboard

Example:

```
cf create-service p-circuit-breaker–dashboard standard cb-dashboard
```

After service has been provisioned, the circuit breaker dashboard is accessible directly from the Pivotal Cloud Foundry *Apps Manager*



Circuit Breaker Dashboard

esuez@pivotal.io

eitan-org > eitan-space > hystrix-dashboard

Circuit Sort: Error then Volume | Alphabetical | Volume | Error | Mean | Median | 90 | 99 | 99.5

Success | Short-Circuited | Timeout | Rejected | Failure | Error %

greeting.getFortune

27	0	0.0 %
0	0	
0	0	

Host: 0.8/s
Cluster: 2.5/s
Circuit Closed

Hosts	3	90th	15ms
Median	5ms	99th	132ms
Mean	10ms	99.5th	132ms

Thread Pools Sort: Alphabetical | Volume |

FortuneServiceClient

Host: 2.7/s
Cluster: 8.1/s

Active	0	Max Active	3
Queued	0	Executions	27
Pool Size	30	Queue Size	1