Circuit Breaker Metric Aggregation

Table of Contents

Estimated Time: 25 minutes

Requirements

- Provided by e-mail last week
- Completion of Spring Cloud Config Lab, Service Discovery Lab, Circuit Breakers lab

---

What You Will Learn

- How to use Turbine in Pivotal Cloud Foundry to consume metric streams and view them with the `hystrix-dashboard`

Exercises

Start the `config-server`, `service-registry`, `fortune-service`, and `greeting-hystrix`

1) Make sure config-server, service-registry, fortune-service, and greeting-hystrix are running, as per the previous labs.

Create a `circuit-breaker-dashboard`

Looking at individual application instances in the Hystrix Dashboard is not very useful in terms of understanding the overall health of the system. Turbine is an application that aggregates all of the relevant `/hystrix.stream` endpoints into a combined `/turbine.stream` for use in the Hystrix Dashboard.

1) Create a Circuit Breaker Dashboard Service Instance

```
$ cf create-service p-circuit-breaker-dashboard standard circuit-breaker-dashb
oard
```

When creating a Circuit Breaker Service instance there are three items that get provisioned:

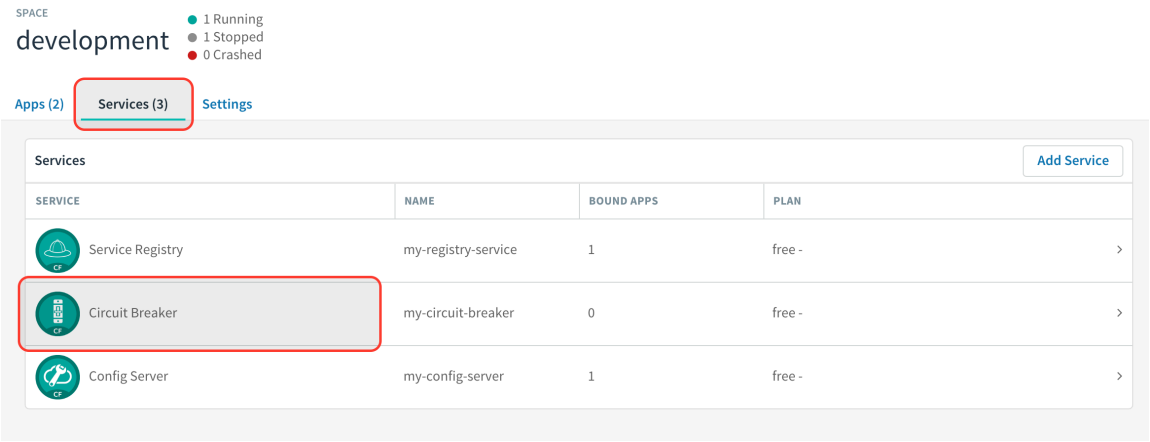
1. Hystrix Dashboard application instance
2. Turbine AMQP application instance
3. RabbitMQ Service Instance

This process takes some time and won't be immediately available for binding. Give it a couple of minutes.

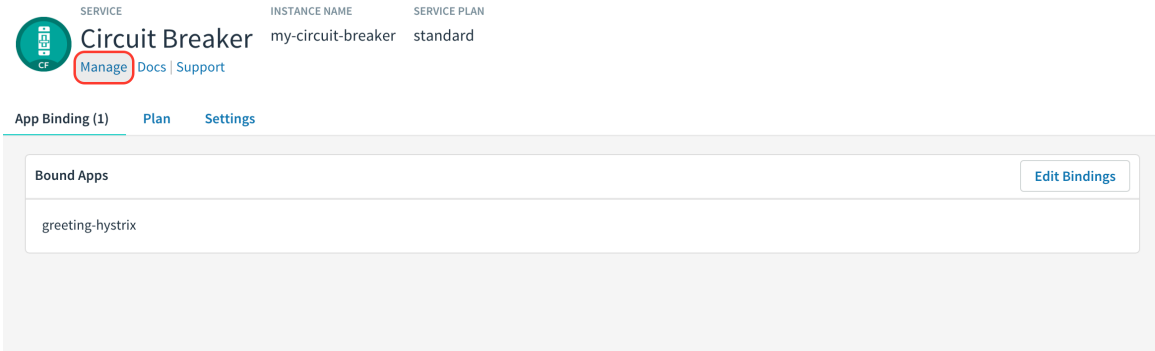
Note: In PCF, when applications register using the `route` registrationMethod method, every application has the same `hostname` (every app instance has the same URL for a given app; read here for more details on registration methods).

Therefore, the traditional Turbine model of pulling metrics from all the distributed Hystrix enabled applications via HTTP doesn't work (it is unknown from the Turbine perspective if all metrics are properly being collected). The problem is solved with Turbine AMQP. Metrics are published through a message broker. PCF uses RabbitMQ.

Click on the ***Services*** tab and the ***Circuit Breaker*** entry to navigate to your service.



Then, click on the ***Manage*** link to determine when the `circuit-breaker` dashboard is ready.



Bind `greeting-hystrix` to `circuit-breaker-dashboard`

1) Bind the circuit-breaker-dashboard service to the `greeting-hystrix`.

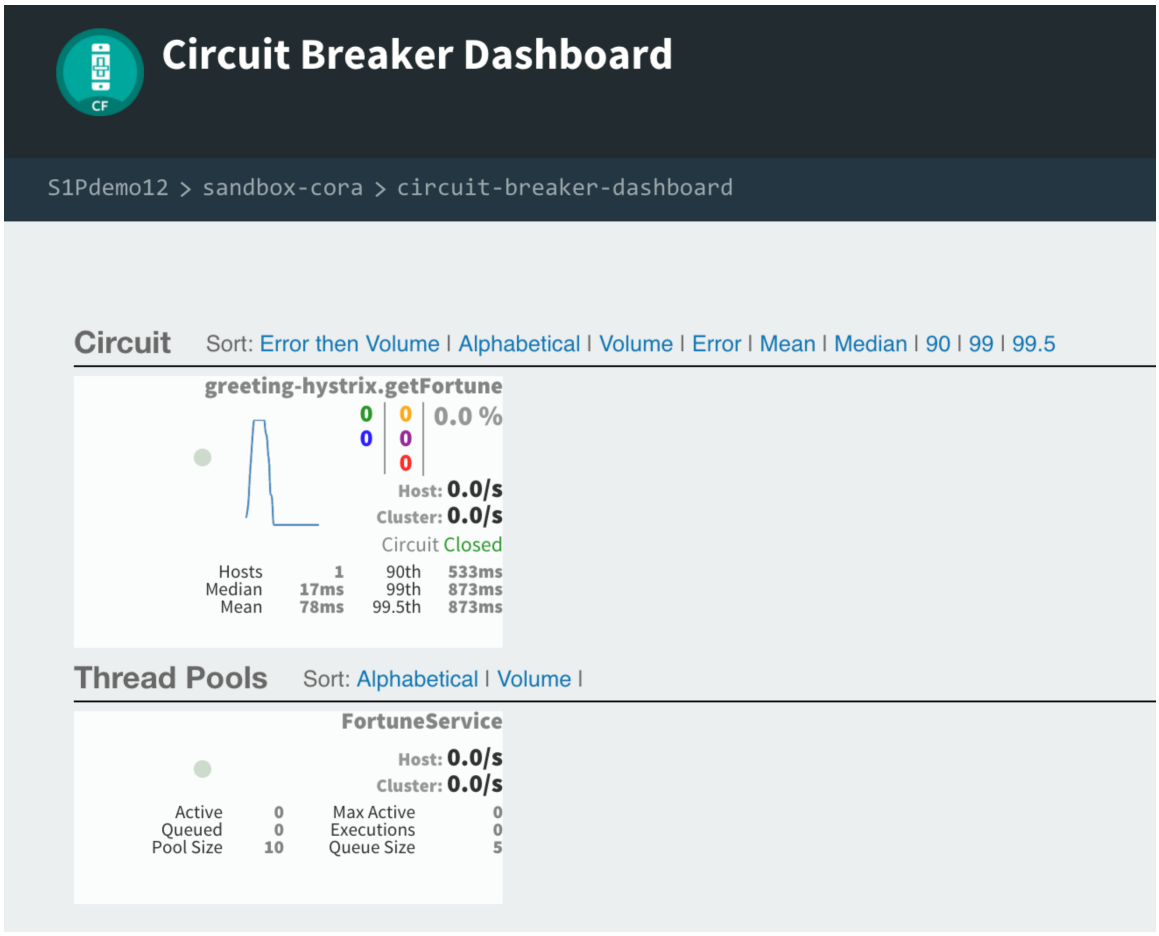
```
$ cf bind-service greeting-hystrix circuit-breaker-dashboard
```

2) Restart the `greeting-hystrix` app

```
$ cf restart greeting-hystrix
```

View `hystrix-dashboard` with turbine stream

1) Return to circuit breaker dashboard (via the ***Manage*** link for the `circuit-breaker-dashboard` service).



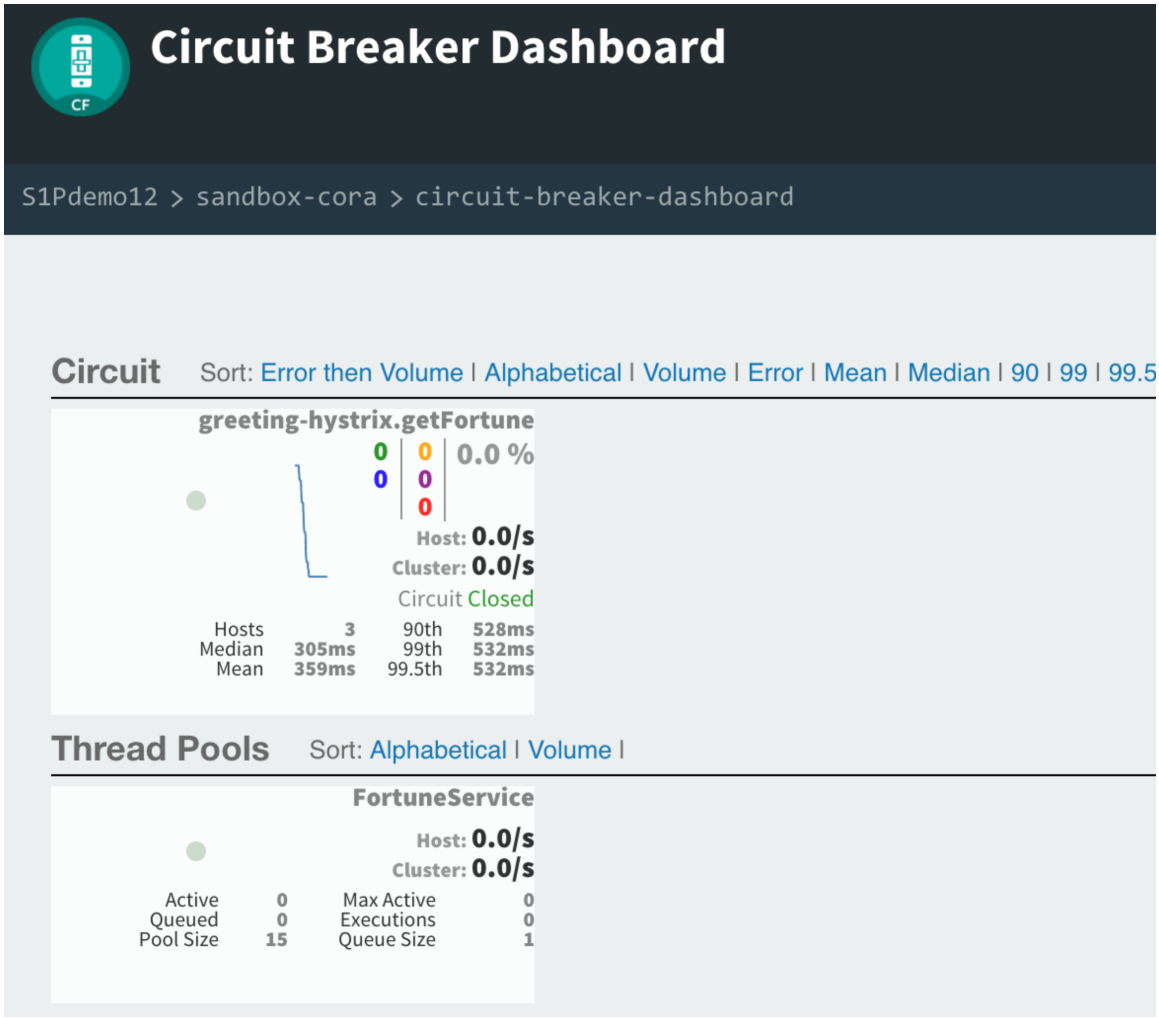
2) Refresh the `greeting-hystrix` / endpoint several times. Notice that Hosts=1, indicating the number of app instance streams being aggregated.

3) Scale the `greeting-hystrix` app

```
$ cf scale greeting-hystrix -i 3
```

4) Refresh the `greeting-hystrix` / endpoint several times again.

5) Check the dashboard again. Notice that Hosts changes to 3, indicating the number of app instance streams being aggregated.



What Just Happened?

The `greeting-hystrix` application is publishing metrics via AMQP to RabbitMQ (this can be discovered by looking at `VCAP_SERVICES`). Those metrics are then consumed and aggregated by Turbine. The Circuit Breaker Dashboard then consumes the Turbine endpoint. All of this detail has been abstracted away by using the PCF Circuit Breaker Dashboard Service.

Back to TOP