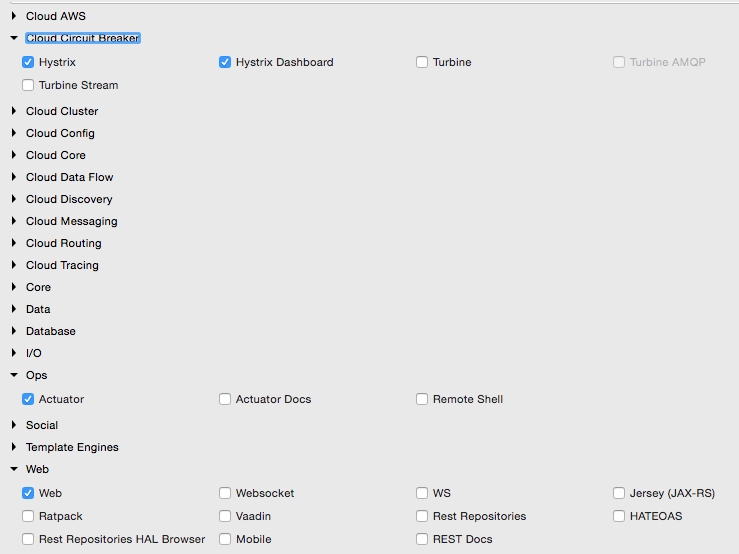
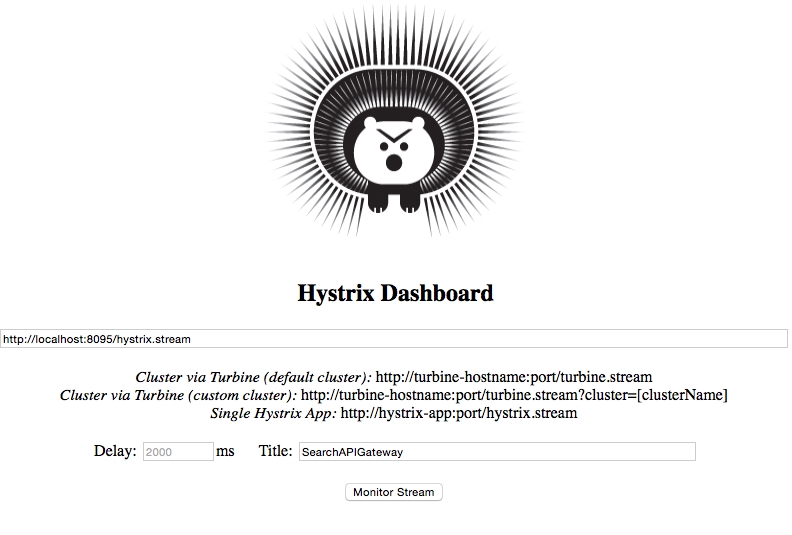
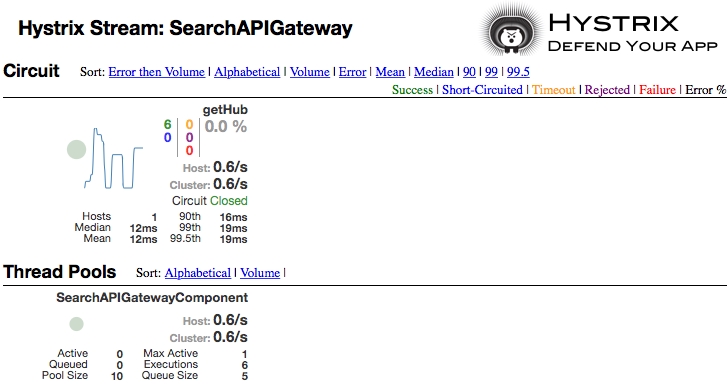
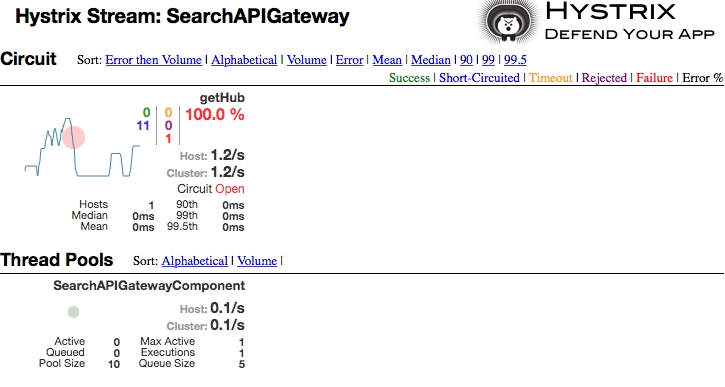
Lab 23 - Hystrix

* Update the Search API Gateway service. Add the Hystrix dependency to the service. If developing from scratch, select the following libraries:
* 
* In the Spring Boot Application class, add @EnableCircuitBreaker. This command will tell Spring Cloud Hystrix to enable a circuit breaker for this application. It also exposes the /hystrix.stream endpoint for metrics collection.
* Add a component class to the Search API Gateway service with a method; in this case, this is getHubannotated with @HystrixCommand. This tells Spring that this method is prone to failure. Spring Cloud libraries wrap these methods to handle fault tolerance and latency tolerance by enabling circuit breaker. The Hystrix command typically follows with a fallback method. In case of failure, Hystrix automatically enables the fallback method mentioned and diverts traffic to the fallback method. As shown in the following code, in this case, getHub will fall back to getDefaultHub:
* @Component   
  class SearchAPIGatewayComponent {   
   @LoadBalanced  
   @Autowired   
   RestTemplate restTemplate;  
    
   @HystrixCommand(fallbackMethod = "getDefaultHub")  
   public String getHub(){  
   String hub = restTemplate.getForObject("http://search-service/search/hub", String.class);  
   return hub;  
   }  
   public String getDefaultHub(){  
   return "Possibily SFO";  
   }  
  }
* The getHub method of SearchAPIGatewayController calls the getHub method of SearchAPIGatewayComponent, as follows:
* @RequestMapping("/hubongw")   
  String getHub(){  
   logger.info("Search Request in API gateway for getting Hub, forwarding to search-service ");  
   return component.getHub();   
  }
* The last part of this exercise is to build a Hystrix Dashboard. For this, build another Spring Boot application. Include Hystrix, Hystrix Dashboard, and Actuator when building this application.
* In the Spring Boot Application class, add the @EnableHystrixDashboard annotation.
* Start the Search service, Search API Gateway, and Hystrix Dashboard applications. Point the browser to the Hystrix Dashboard application's URL. In this example, the Hystrix Dashboard is started on port 9999. So, open the URL http://localhost:9999/hystrix.
* A screen similar to the following screenshot will be displayed. In the Hystrix Dashboard, enter the URL of the service to be monitored.
* In this case, Search API Gateway is running on port 8095. Hence, the hystrix.stream URL will be http://localhost:8095/hytrix.stream, as shown:
* 
* The Hystrix Dashboard will be displayed as follows:
* 

### Tip

* Note that at least one transaction has to be executed to see the display. This can be done by hitting http://localhost:8095/hubongw.
* Create a failure scenario by shutting down the Search service. Note that the fallback method will be called when hitting the URL http://localhost:8095/hubongw.
* If there are continuous failures, then the circuit status will be changed to open. This can be done by hitting the preceding URL a number of times. In the open state, the original service will no longer be checked. The Hystrix Dashboard will show the status of the circuit as **Open**, as shown in the following screenshot. Once a circuit is opened, periodically, the system will check for the original service status for recovery. When the original service is back, the circuit breaker will fall back to the original service and the status will be set to **Closed**:
* 

### Note

* To know the meaning of each of these parameters, visit the Hystrix wiki at <https://github.com/Netflix/Hystrix/wiki/Dashboard>.