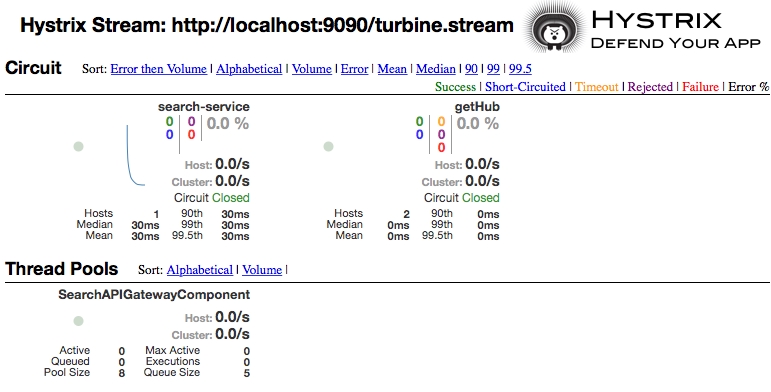
Lab 23 and a Half - Turbine

* The Turbine server can be created as just another Spring Boot application using Spring Boot Starter. Select Turbine to include the Turbine libraries.
* Once the application is created, add @EnableTurbine to the main Spring Boot Application class. In this example, both Turbine and Hystrix Dashboard are configured to be run on the same Spring Boot application. This is possible by adding the following annotations to the newly created Turbine application:
* @EnableTurbine  
  @EnableHystrixDashboard  
  @SpringBootApplication  
  public class TurbineServerApplication {
* Add the following configuration to the .yaml or property file to point to the instances that we are interested in monitoring:
* spring:  
   application:  
   name : turbineserver  
  turbine:  
   clusterNameExpression: new String('default')  
   appConfig : search-service,search-apigateway  
  server:  
   port: 9090  
  eureka:  
   client:  
   serviceUrl:  
   defaultZone: http://localhost:8761/eureka/
* The preceding configuration instructs the Turbine server to look up the Eureka server to resolve the search-service and search-apigateway services. The search-service and search-apigateways service IDs are used to register services with Eureka. Turbine uses these names to resolve the actual service host and port by checking with the Eureka server. It will then use this information to read /hystrix.stream from each of these instances. Turbine will then read all the individual Hystrix streams, aggregate all of them, and expose them under the Turbine server's /turbine.stream URL.
* The cluster name expression is pointing to the default cluster as there is no explicit cluster configuration done in this example. If the clusters are manually configured, then the following configuration has to be used:
* turbine:  
   aggregator:  
   clusterConfig: [comma separated clusternames]
* Change the Search service's SearchComponent to add another circuit breaker, as follows:
* @HystrixCommand(fallbackMethod = "searchFallback")  
   public List<Flight> search(SearchQuery query){
* Also, add @EnableCircuitBreaker to the main Application class in the Search service.
* Add the following configuration to bootstrap.properties of the Search service. This is required because all the services are running on the same host:
* Eureka.instance.hostname: localdomain1
* Similarly, add the following in bootstrap.properties of the Search API Gateway service. This is to make sure that both the services use different hostnames:
* eureka.instance.hostname: localdomain2
* In this example, we will run two instances of search-apigateway: one on localdomain1:8095 and another one on localdomain2:8096. We will also run one instance of search-service on localdomain1:8090.
* Run the microservices with command-line overrides to manage different host addresses, as follows:
* **java -jar -Dserver.port=8096 -Deureka.instance.hostname=localdomain2 -Dserver.address=localdomain2 target/chapter7.search-apigateway-1.0.jar**  
  **java -jar -Dserver.port=8095 -Deureka.instance.hostname=localdomain1 -Dserver.address=localdomain1 target/chapter7.search-apigateway-1.0.jar**  
  **java -jar -Dserver.port=8090 -Deureka.instance.hostname=localdomain1 -Dserver.address=localdomain1 target/chapter7.search-1.0.jar**
* Open Hystrix Dashboard by pointing the browser to http://localhost:9090/hystrix.
* Instead of giving /hystrix.stream, this time, we will point to /turbine.stream. In this example, the Turbine stream is running on 9090. Hence, the URL to be given in the Hystrix Dashboard is http://localhost:9090/turbine.stream.
* Fire a few transactions by opening a browser window and hitting the following two URLs: http://localhost:8095/hubongw and http://localhost:8096/hubongw.
* Once this is done, the dashboard page will show the **getHub** service.
* Run chapter7.website. Execute the search transaction using the website http://localhost:8001.
* After executing the preceding search, the dashboard page will show **search-service** as well. This is shown in the following screenshot:
* 

As we can see in the dashboard, **search-service** is coming from the Search microservice, and **getHub** is coming from Search API Gateway. As we have two instances of Search API Gateway, **getHub** is coming from two hosts, indicated by **Hosts 2**.