Configuring Eureka

David Liu edited this page on Mar 30, 2016 · 11 revisions

Read the Eureka-at-a-glance page to understand the concepts of the setup better.

Eureka comes with two components - **Eureka Client** and the **Eureka Server**. Your architecture which is using Eureka will typically have two applications

- **Application Client** which uses Eureka Client to make requests to the Application Service.
- **Application Service** which receives requests from Application Client and sends a response back.

The setups involve the following

- Eureka Server
- Eureka Client for the application client
- Eureka Client for the application service

Eureka can be run in both AWS and non-AWS environments.

If you are running in the cloud environment, you will need to pass in the java commandline property -Deureka.datacenter=cloud so that the Eureka Client/Server knows to initialize the information specific to AWS cloud.

Configuring Eureka Client

Prerequisites

- JDK 1.8 or higher

You have the following choices to get the Eureka client binaries. Always try to get the latest release as tend to have more fixes.

- You can download the Eureka Client binary by using this URL "http://search.maven.org/#search%7Cga%7C1%7Ceureka-client"
- You can add the eureka client as a maven dependency

```
<dependency>
 <groupId>com.netflix.eureka</groupId>
 <artifactId>eureka-client</artifactId>
 <version>1.1.16</version>
</dependency>
```

- You can build the client as specified here.

Configuration

The easiest way to configure Eureka client is by using property files. By default, Eureka client searches for the property file *eureka-client.properties* in the *classpath*. It further searches for environment specific overrides in the environment specific properties files. The environment is typically *test* or *prod* and is supplied by a *-Deureka.environment* java commandline switch to the eureka client (without the *.properties* suffix). Accordingly, the client also searches for *eureka-client-{test,prod}.properties*.

You can take a look at the examples here for default configurations. You can copy these configurations and edit for your need and place them in your class path. If you want to change the name of the properties file for some reason you can do so by specifying *-Deureka.client.props=* (without a suffix) in the java commandline switch, where is the name of the property file to search for for.

The properties in the files explain what they are for. At the minimum the following things need to be configured:

```
    Application Name (eureka.name)
    Application Port (eureka.port)
    Virtual HostName (eureka.vipAddress)
    Eureka Service Urls (eureka.serviceUrls)
```

For more advanced configurations, check out the options available in the following links.
https://github.com/Netflix/eureka/blob/master/eureka-client/src/main/java/com/netflix/appinfo/EurekaInstanceConfig.java
https://github.com/Netflix/eureka/blob/master/eureka-client/src/main/java/com/netflix/discovery/EurekaClientConfig.java

Configuring Eureka Server

Prerequisites

- JDK 1.8 or higher
- Tomcat 6.0.10 or higher

With Eureka server, you have the following choices to get the binaries

- You can build a WAR archive from the sources as specified here.
- You can download the WAR archive from mavencentral by using this URL "http://search.maven.org/#search%7Cga%7C1%7Ceureka-server"

Configuration

Eureka Server has two sets of configurations

- Eureka Client configuration as explained above.
- Eureka Server configuration.

The easiest way to configure Eureka Server is by using property files similar to the Eureka Client above. First, configure the Eureka client that is running with the server as specified above. Eureka server itself fires up a Eureka Client that it uses to find other Eureka Servers. Therefore, you need to first configure the Eureka Client for the Eureka Server as you would do with any other clients that connect to the Eureka service. The Eureka Server will use its Eureka Client configuration to identify peer eureka server that have the same name (ie) eureka.name

After configuring the Eureka Client, you may need to configure the Eureka Server if you are running in AWS. Eureka server by default searches for property file *eureka-server.properties* in the *classpath*. It further searches for environment specific overrides in the environment specific properties files. The environment is typically *test* or *prod* and is supplied by a *-Deureka.environment* java commandline switch to the eureka server (without the *.properties* suffix). Accordingly the server also searches for *eureka-server-{test,prod}.properties*.

Configuring for local development
When running eureka server for local development, there is typically a wait of ~3 minutes until it fulling boots up. This is due to the default server behaviour to search for peers to sync up and retries when it finds no available peers. This wait time can be reduced by setting the property `eureka.numberRegistrySyncRetries=0` .

Configuring for AWS
Additional configurations are required if you are running in AWS as explained here. For more advanced server configurations, refer to the options available here.

If you are building the WAR archive, you can edit the files under *eureka-server/conf* in place and the build takes care of placing the properties files under WEB-INF/classes before creating the archive.

If you are downloading the archive from maven, then you can merge in the edited property files under WEB-INF/classes yourself.

Running the demo application may help you to understand the configurations better.

Client/Server version compatibility

We use semantic versioning for eureka, and will maintain client/server protocol compatibility between minor version upgrades (i.e. the 1.x versions should have compatible protocol between client and server deployments). In general, it's always safer to have the servers be on a newer version than clients.