

A&AI Developer Environment Setup - Updated for Beijing!

This guide will illustrate setting up an A&AI development environment in Ubuntu 16.04.

DRAFT - this guide is in process of being updated - thanks for your patience 😊

For this exercise, I set up a new instance of Ubuntu in Virtualbox and gave it 16G RAM, 200GB dynamically allocated storage, and 3 processors.

1. install openjdk 8
 - a. `sudo apt install openjdk-8-jdk`
2. Install single node hadoop/janusgraph
 - a. `$ wget http://github.com/JanusGraph/JanusGraph/releases/download/v0.2.0/janusgraph-0.2.0-hadoop2.zip`
 - b. `$ unzip janusgraph-0.2.0-hadoop2.zip`
 - c. `$ cd janusgraph-0.2.0-hadoop2/`
 - d. `$ ./bin/janusgraph.sh start`, make sure you are not a root user as elasticsearch cannot be run as root, response is like:

Forking Cassandra...

Running `nodetool statusthrift`... OK (returned exit status 0 and printed string "running").

Forking Elasticsearch...

Connecting to Elasticsearch (127.0.0.1:9200)..... OK (connected to 127.0.0.1:9200).

Forking Gremlin-Server...

Connecting to Gremlin-Server (127.0.0.1:8182).... OK (connected to 127.0.0.1:8182).

Run `gremlin.sh` to connect.
 - e. you can verify whether everything is running by executing
`./bin/janusgraph.sh status`
3. Install haproxy (If you are on Mac OS X, Here's the link to HAProxy setting with Docker [Setting up HAProxy for MAC OS X user](#))
 - a. `$ sudo apt-get -y install haproxy`
 - b. `$ <path-to-haproxy>/haproxy -v`
HA-Proxy version 1.6.3 2015/12/25
Copyright 2000-2015 Willy Tarreau <willy@haproxy.org>
 - c. Install this [haproxy.cfg](#) file in `/etc/haproxy`
 - d. Download [aai.pem](#)
 - e. `$ sudo cp aai.pem /etc/ssl/private/aai.pem`
 - f. `$ sudo chmod 640 /etc/ssl/private/aai.pem`
 - g. `$ sudo chown root:ssl-cert /etc/ssl/private/aai.pem`
 - h. `sudo mkdir /usr/local/etc/haproxy`
 - i. Add these hostnames to the loopback interface in `/etc/hosts`:

i. 127.0.0.1 localhost [aai-traversal.api.simpledemo.openecomp.org](#) [aai-resources.api.simpledemo.openecomp.org](#)
 - j. `$ sudo service haproxy restart`
4. Set up repos. First, follow the initial setup instructions in [Setting Up Your Development Environment](#)
 - a. `$ mkdir -p ~/LF/AAI`
 - b. `$ cd ~/LF/AAI`
 - c. `$ git clone ssh://<username>@gerrit.onap.org:29418/aai/aai-common`
 - d. `$ git clone ssh://<username>@gerrit.onap.org:29418/aai/traversal`
 - e. `$ git clone ssh://<username>@gerrit.onap.org:29418/aai/resources`
 - f. `$ git clone ssh://<username>@gerrit.onap.org:29418/aai/logging-service`
 - g. If you did not originally create a `settings.xml` file when setting up the dev environment, you may get an error on some of the repos saying that oparent is unresolvable. Using the example `settings.xml` file should solve this problem: [Setting Up Your Development Environment#MavenExampleSettings.xml](#)
5. Build aai-common, traversal, and resources
 - a. `$ cd ~/LF/AAI/aai-common`
 - b. `$ mvn -DskipTests clean install`
Should result in BUILD SUCCESS
 - c. `$ cd ~/LF/AAI/resources`
 - d. `$ mvn -DskipTests clean install`
Should result in BUILD SUCCESS
 - e. `$ cd ~/LF/AAI/logging-service`
 - f. `$ mvn -DskipTests clean install`
Should result in BUILD SUCCESS
 - g. `$ cd ~/LF/AAI/traversal`
I had to add the following to `traversal/pom.xml` to get traversal to build, this may not be necessary:


```
<repositories>
  <repository>
    <id>maven-restlet</id>
    <name>Restlet repository</name>
    <url>https://maven.restlet.com</url>
  </repository>
</repositories>
```

- h. `mvn -DskipTests clean install`
Should result in BUILD SUCCESS
6. Janus setup
 - a. Modify both `janus-cached.properties` and `janus-realtime.properties` to the following (for all MS's that will connect to the local Cassandra backend)


```
storage.backend=cassandra
storage.hostname=localhost
storage.cassandra.keyspace=onap; or different keyspace name of your choosing
```

 - update `~/LF/AAI/resources/aai-resources/src/main/resources/etc/appprops/janusgraph-cached.properties`
 - update `~/LF/AAI/resources/aai-resources/src/main/resources/etc/appprops/janusgraph-realtime.properties`
 - update `~/LF/AAI/traversal/aai-traversal/src/main/resources/etc/appprops/janusgraph-cached.properties`
 - update `~/LF/AAI/traversal/aai-traversal/src/main/resources/etc/appprops/janusgraph-realtime.properties`
 - a. **NOTE:** The first thing that would need to be done is adding the schema to the local instance. (this will need to be done whenever using a new keyspace or after wiping the data).
 - b. Here's the command I used, and it worked:


```
$ cd ~/LF/AAI/resources; java -DAJSC_HOME=aai-resources -DBUNDLECONFIG_DIR=src/main/resources/ -Dloader.main=org.onap.aai.dgen.GenTester -jar aai-resources/target/aai-resources-1.2.0-SNAPSHOT.jar
```

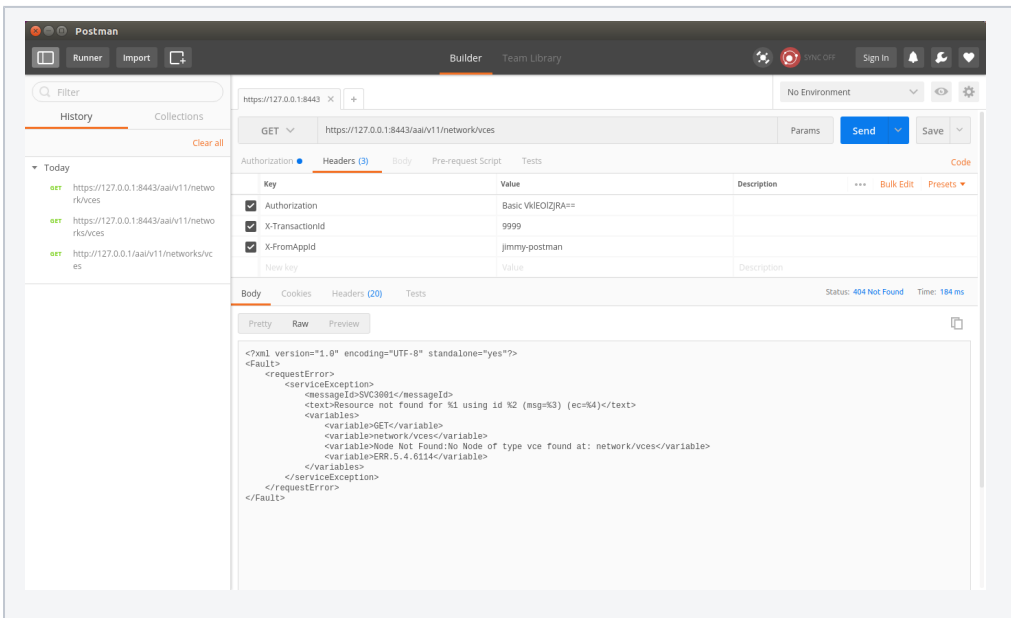
 verify the **version of built jar**
7. Start the "resources" microservice
 - a. Resources runs on port 8447. Go to the resources directory


```
$ cd ~/LF/AAI/resources
```
 - b. Set the debug port to 9447


```
$ export MAVEN_OPTS="-Xms1024m -Xmx5120m -XX:PermSize=2024m -Xdebug -Xnoagent -Djava.compiler=NONE -Xrunjdwp:transport=dt_socket,address=9447,server=y,suspend=n"
```
 - c. Start the microservice


```
$ java -DAJSC_HOME=aai-resources -DBUNDLECONFIG_DIR=src/main/resources/ -jar aai-resources/target/aai-resources-1.2.0-SNAPSHOT.jar
```

 Should see something like this: Resources Microservice Started
8. Verify the resources microservice (this example uses Postman utility for Google Chrome)
 - a. Use basic auth, user = AAI, pw = AAI
 - b. Set the X-TransactionId header (in the example below, the value is 9999)
 - c. Set the X-FromAppId header (in the example below, the value is jimmy-postman)
 - d. Perform a GET of <https://127.0.0.1:8443/aai/v1/network/vces> (If you don't have HAproxy, you can use 8447 port instead of 8443)
 - e. You should see an error as below, 404 Not Found, ERR.5.4.6114. This indicates that the service is functioning normally:



9. Start the "traversal" microservice
 - a. Traversal runs on port 8446. Go to the traversal directory

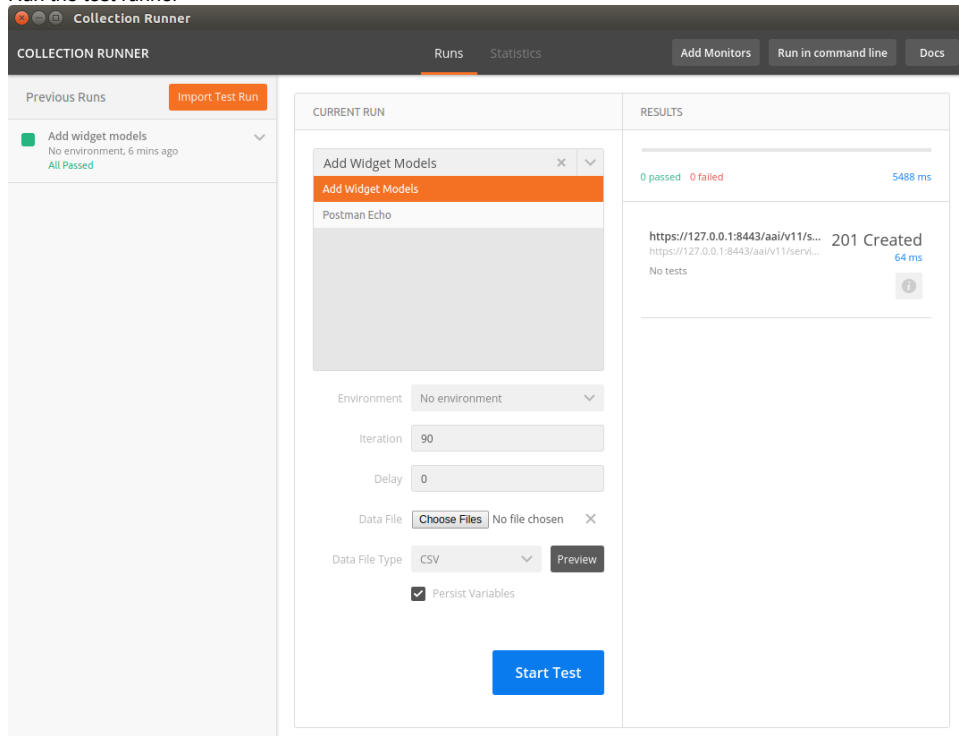

```
$ cd ~/LF/AAI/traversal
```
 - b. Set the debug port to 9446


```
$ export MAVEN_OPTS="-Xms1024m -Xmx5120m -XX:PermSize=2024m -Xdebug -Xnoagent -Djava.compiler=NONE -Xrunjdwp:transport=dt_socket,address=9446,server=y,suspend=n"
```
 - c. Start the microservice

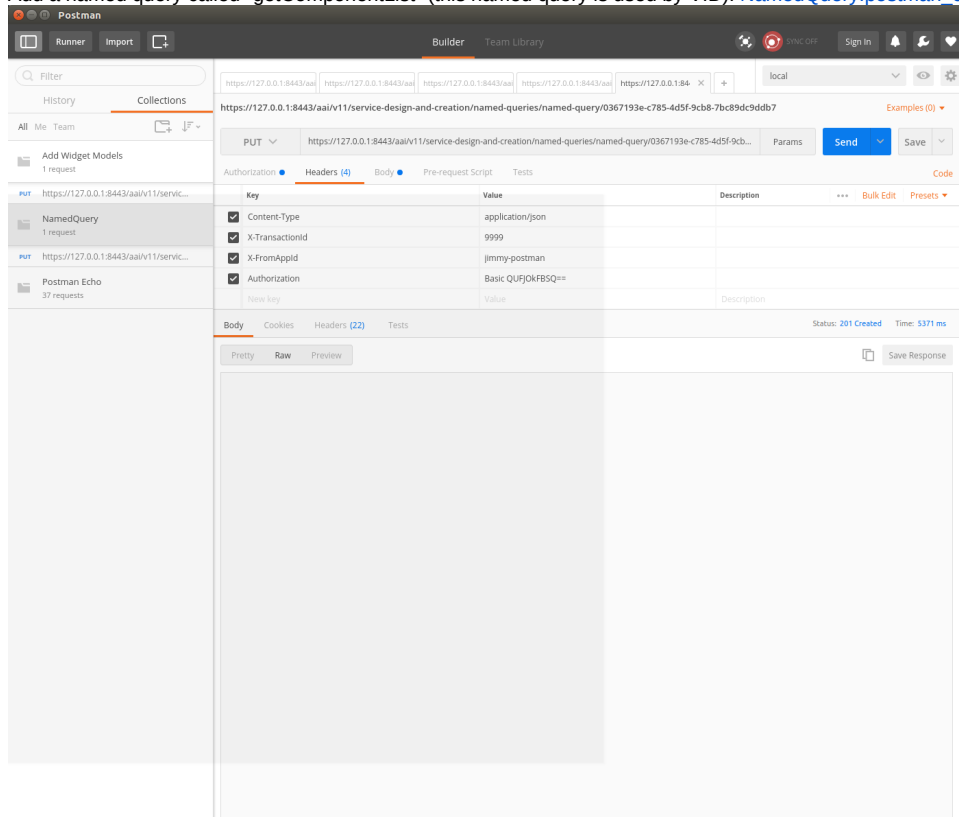

```
$ java -DAJSC_HOME=aai-traversal -DBUNDLECONFIG_DIR=src/main/resources/ -jar aai-traversal/target/aai-traversal-1.2.0-SNAPSHOT.jar
```

 Should see something like this: Traversal Microservice Started
10. Verify the traversal microservice by executing attached postman AAI API calls

- Set up the widget models
This will set up the postman to add widget models: [Add Widget Models.postman_collection.json](#)
- Create a runner using this file: [models.csv](#)
- Run the test runner

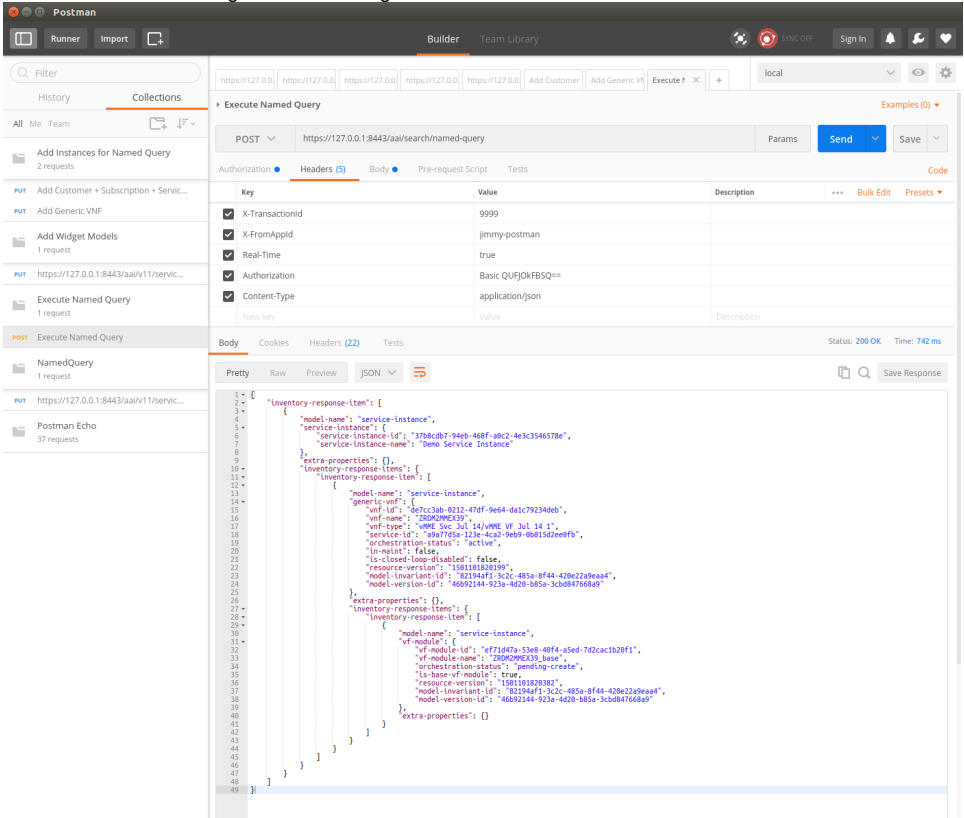


- Add a named query called "getComponentList" (this named query is used by VID): [NamedQuery.postman_collection.json](#)



- Add objects: [Add Instances for Named Query.postman_collection.json](#) (when using vXX in place of v11, replace the xmlns "[http://org.openecomp.aai.inventory/v11](#)" with "[http://org.onap.aai.inventory/vXX](#)" in the Body of the PUT request)

- f. Execute named-query: [Execute Named Query.postman_collection.json](#)
You should see something like the following:



11. Your A&AI instance is now running, both the resources and traversal microservices are working properly with a local janus graph.
12. Next: [Tutorial: Making and Testing a Schema Change in A&AI in Beijing Release](#)

Attachments

File	Modified
File aai.pem	Mar 13, 2018 by James Forsyth
File Add Instances for Named Query.postman_collection.json	Mar 13, 2018 by James Forsyth
File Add Widget Models.postman_collection.json	Mar 13, 2018 by James Forsyth
File Execute Named Query.postman_collection.json	Mar 13, 2018 by James Forsyth
File haproxy.cfg	Mar 13, 2018 by James Forsyth
PNG File image2017-7-26_11-6-11.png	Mar 13, 2018 by James Forsyth
PNG File image2017-7-26_16-17-19.png	Mar 13, 2018 by James Forsyth
PNG File image2017-7-26_16-23-12.png	Mar 13, 2018 by James Forsyth
PNG File image2017-7-26_16-58-5.png	Mar 13, 2018 by James Forsyth
File models.csv	Mar 13, 2018 by James Forsyth
File NamedQuery.postman_collection.json	Mar 13, 2018 by James Forsyth

[Download All](#)