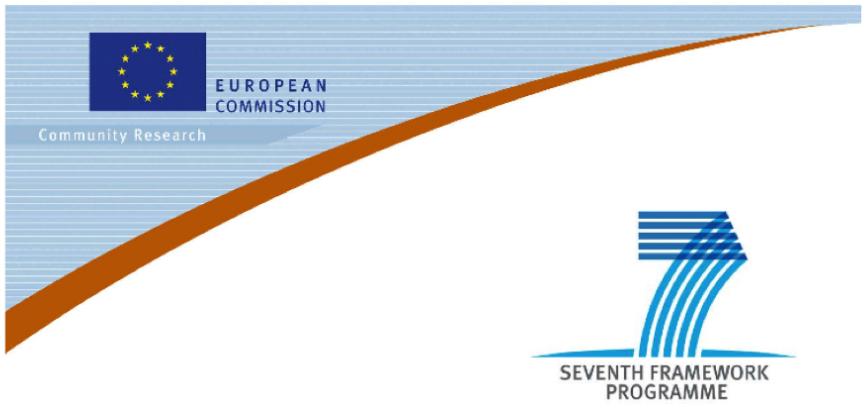

Private Public Partnership Project (PPP)

Large-scale Integrated Project (IP)

Installation & Administration Manual

Author: AEON Cloud Messaging GE development team (javier.garcia@atos.net)

Contents

AEON platform (Cloud Messaging GE) installation and configuration

To get a wider information about aeon, please, go to the README.md file.

Requires:

- Node.js (version 0.10)
- RabbitMQ (version 3.1.2)
- MongoDB

Setting up the environment:

To install the whole platform project, it is necessary to download the code from three different modules: the frontend, the rest and the events-manager module.

Setting up the EVENTS-MANAGER module.

Download the code:

```
$ git clone git@github.com:atos-ari-aeon/fiware-cloud-messaging-events-manager.git
```

Install necessary packages for node.js accordingly package.json:

```
$ cd ./fiware-cloud-messaging-events-manager/
$ npm install
```

Configure ./config/brokerconnetor_[environment].js with proper rabbitMQ values:

module.exports.url = “amqp://[user:password@]hostname[:port]”

Configure ./config/db_[environment].js with proper mongoDB values:

module.exports.host = “”;

module.exports.port = “”;

module.exports.db = “AEON”;

module.exports.username = “”;

module.exports.password = “”;

module.exports.collectionUsers = “AEONUsers”;

module.exports.collectionEntities = “AEONEntities”;

module.exports.collectionChannels = “AEONChannels”;

module.exports.collectionLogs = “AEONLogs”;

Run EVENTS-MANAGER:

```
$ node app.js &
```

Setting up the REST module.

Download the code:

```
$ git clone git@github.com:atos-ari-aeon/fiware-cloud-messaging-api.git
```

Install necessary packages for node.js accordingly package.json:

```
$ cd ./fiware-cloud-messaging-api/
$ npm install
```

Configure ./config/brokerconnetor_[environment].js with proper RabbitMQ values:

module.exports.url = “amqp://[user:password@]hostname[:port]”

Configure ./config/app_[environment].js with proper values:

module.exports.extHost = [externalHost]; //Rest API server IP

module.exports.host = [internalHost]; //Rest API server internal IP (if different)

module.exports.socket_server_host = [externalHost]; //Events-Manager server IP

module.exports.extHostGUI = [externalHost]; //Dashboard server IP

Configure ./config/db_[environment].js with proper mongoDB values:

module.exports.host = “”;

module.exports.port = “”;

module.exports.db = “AEON”;

module.exports.username = “”;

module.exports.password = “”;

module.exports.collectionUsers = “AEONUsers”;

module.exports.collectionEntities = “AEONEntities”;

module.exports.collectionChannels = “AEONChannels”;

module.exports.collectionLogs = “AEONLogs”;

Run REST:

```
$ node app.js &
```

Setting up the FRONTEND module.

Download the code:

```
$ git clone git@github.com:atos-ari-aeon/fiware-cloud-messaging-dashboard.git
```

Navigate to the specific folder:

```
$ cd ./fiware-cloud-messaging-dashboard/
```

Configure ./app/controllers/config.js with proper values:

RECAPTCHA_PUBLIC_KEY: 'captchaKey'

AEON_HOST: [externalHost], //Rest API server IP

AEON_PORT: [externalHost], //Rest API server PORT

Run FRONTEND:

```
$ node scripts/web-server.js &
```

To get the live demo running, it is necessary to create an entity and a channel after setting up an account. After that, go to the ./app/controllers/config.js and set the values:

LIVE_DEMO_PUBURL: "Channel publication url"

LIVE_DEMO_SUBURL: "Channel subscription url"

Deploy Cloud Messaging GE using Image

To deploy the Cloud Messaging GE using the image recipes, it is necessary to download them from the repo:

```
$ git clone https://github.com/atos-ari-aeon/fiware-cloud-messaging-platform.git
```

Navigate to the development environment folder:

```
$ cd ./fiware-cloud-messaging-platform/development_environment/AeonInstallAndConfig
```

Execute the script aeon.sh:

```
$ ./aeon.sh
```

When the script finishes, navigate to the aeonConfiguration folder:

```
$ cd /home/aeon/aeonConfiguration
```

Once here, it is necessary to configure the puppet files with your external and internal IPs. The files that need to be edited are:

```
$ vim /home/aeon/aeonConfiguration/aeon_dashboard/manifests/main.pp
$ vim /home/aeon/aeonConfiguration/aeon_api/manifests/main.pp
$ vim /home/aeon/aeonConfiguration/aeon_events_manager/manifests/main.pp
```

and the varibles to be set are $internalIP and $externalIP.

Finally, execute the puppet apply commands:

```
$ sudo puppet apply --modulepath /home/aeon/aeonConfiguration/aeon_api/modules /home/aeon/aeonConfiguration/aeon_api/manifests/main.pp
$ sudo puppet apply --modulepath /home/aeon/aeonConfiguration/aeon_events_manager/modules /home/aeon/aeonConfiguration/aeon_events_manager/manifests/main.pp
$ sudo puppet apply --modulepath /home/aeon/aeonConfiguration/aeon_dashboard/modules /home/aeon/aeonConfiguration/aeon_dashboard/manifests/main.pp
```

After finishing, the Cloud Messaging will be accesible.

Deploy Cloud Messaging GE using Docker

It is possible to deploy the Cloud Messaging GE using Docker. Here you have the steps:

Download the Cloud Messaging GE from the repo:

```
$ git clone https://github.com/atos-ari-aeon/fiware-cloud-messaging-platform.git
```

Navigate to the docker folder

```
$ cd ./fiware-cloud-messaging-platform/docker
```

Edit the docker-compose.yml file. Add your ip in the field "docker_host":

```
mongo:
 container_name: "mongo"
 build: env/mongodb
 ports:
 - "27017:27017"
rabbitmq:
 image: rabbitmq
 container_name: "rabbitmq"
 ports:
  - "5672:5672"
  - "15672:15672"
events:
 container_name: "events"
 build: ./aeon-events-manager
 ports:
  - "7789:7789"
 links:
  - mongo
  - rabbitmq
dashboard:
 container_name: "dashboard"
 build: ./aeon-dashboard
 ports:
  - "8080:8000"
rest:
 container_name: "rest"
 build: ./aeon-api
 ports:
  - "3000:3000"
 links:
  - mongo
  - rabbitmq
  - events
  - dashboard
 extra_hosts:
  - "docker_host: <YOUR_IP>"
```

Run the script file:

```
$ ./deploy_aeon.sh
```

This script will download the source code from the different repositories and will execute the docker-compose.yml file to run the Cloud Messaging GE.

Sanity Check Procedures

The Sanity Check Procedures are the steps that a System Administrator will take to verify that an installation is ready to be tested. This is therefore a preliminary set of tests to ensure that obvious or basic malfunctioning is fixed before proceeding to unit tests, integration tests and user validation.

End to End tests

It is possible to test the Cloud Messaging GE by accessing to its dashboard:

```
http://\<server_ip\>:8080
```

To test if everything is running properly, try to create a new user by clicking the "Sign Up" button. After the registration process, you will be redirected to the main page. Once here do the following:

- Create a new entity with a given name
- Create a new channel for that entity with a given name
- From the menu bar, open the publication and the subscription demo apps.
- In the subscription demo app, select the existing channel and press the "Subscribe" button.
- In the publication demo app, select the existing channal and press the "Attach" button.
- Send some position messages.
- In the subscription demo app, verify that the positions sent appears in the map.

Everything is working fine!

List of running processes

To check if the Cloud Messaging is running, three processes must be running:

```
<user>  4386  0.0  0.6 658976  6492 ? Sl  09:55  0:00 node scripts/web-server
<user>  5046  0.2  2.5 672980 26304 ? Sl  09:56  0:00 node app.js
<user>  5281  0.4  6.7 728852 68896 ? Sl  09:56  0:00 node app.js
```

Those three processes belong to the three modules that are part of the Cloud Messaging GE. Apart from them, it is necessary to have a RabbitMQ server and a MongoDB up and running:

```
root  4424  1.1  4.1 530056 41824 ?  Ssl  09:55  0:14 mongod
999  4240  0.0  0.0   4084   300 ?  Ss  09:55  0:00 tini -- rabbitmq-server
999  4258  0.1  6.6 130448 67508 ?  Sl  09:55  0:02 /usr/lib/erlang/erts-6.4.1/bin/beam -W w -A
    64 -P 1048576 -K true -- -root /usr/lib/erlang -progname erl -- -home /var/lib/rabbitmq -- -
    pa /usr/lib/rabbitmq/lib/rabbitmq_server-3.5.4/sbin/../ebin -noshell -noinput -s rabbit boot
     -sname rabbit@e3909f035dc1 -boot start_sasl -config /etc/rabbitmq/rabbitmq -kernel inet_def
    ault_connect_options [{nodelay,true}] -sasl errlog_type error -sasl sasl_error_logger tty -r
    abbit error_logger tty -rabbit sasl_error_logger tty -rabbit enabled_plugins_file "/etc/rabb
    itmq/enabled_plugins" -rabbit plugins_dir "/usr/lib/rabbitmq/lib/rabbitmq_server-3.5.4/sbin
    /../plugins" -rabbit plugins_expand_dir "/var/lib/rabbitmq/mnesia/rabbit@e3909f035dc1-plugin
    s-expand" -os_mon start_cpu_sup false -os_mon start_disksup false -os_mon start_memsup false
     -mnesia dir "/var/lib/rabbitmq/mnesia/rabbit@e3909f035dc1" -kernel inet_dist_listen_min 256
    72 -kernel inet_dist_listen_max 25672
```

Opened Ports

To get the needed ports used by the Cloud Messaging GE, we have executed the next command:

```
netstat -plnt
```

As a result, we identified the processed related to nodejs, rabbitmq and mongod:

```
Proto Recv-Q Send-Q Local Address           Foreign Address         State       PID/Program name
tcp        0      0 <server_ip>:3000        0.0.0.0:*               LISTEN      6913/nodejs
tcp        0      0 0.0.0.0:8080            0.0.0.0:*               LISTEN      6898/nodejs
tcp        0      0 0.0.0.0:7789            0.0.0.0:*               LISTEN      6898/nodejs
tcp        0      0 0.0.0.0:15672           0.0.0.0:*               LISTEN      18565/beam.smp
tcp        0      0 0.0.0.0:27017           0.0.0.0:*               LISTEN      1555/mongod
tcp        0      0 0.0.0.0:28017           0.0.0.0:*               LISTEN      1555/mongod
tcp6       0      0 :::5672                 :::*                    LISTEN      18565/beam.smp
```

Remote Service Access

To check if the Cloud Messaging GE is updated and up & running, it is necessary to invoke the service:

```
http://130.206.81.70:3000/version
```

The output of this execution should be the actual version of AEON.

```
{
  "code": 200,
  "desc": "ok",
  "result": [
    {
      "version": "0.2.2",
      "codename": "Bolt"
    }
  ]
}
```

Diagnosis Procedures

Resource Consumptions

The Cloud Messaging GE is prepared to be deployed in a cloud environment. It can be installed in a phisical machine with the following characteristics:

Machine Type	Physical Machine
CPU	Intel(R) Core(TM) i5-3337U CPU @ 1.80GHz
RAM	4 GB
HDD	500 GB
Operating System	Ubuntu 14.04

We have test the two main modules of the platform:

Idle services:

	CPU (%)	**RAM (MB)**
Rest Interface	0 %	70 MB
Events-Manager	0 %	65 MB

1000 publishing request from 3 clients

	CPU (%)	**RAM (MB)**
Rest Interface	7 %	100 MB
Events-Manager	2 %	70 MB

50000 publishing requests from 100 clients

	CPU (%)	**RAM (MB)**
Rest Interface	40 %	100 MB
Events-Manager	12 %	70 MB

I/O flows

Use the following commands to manage the Cloud Messaging modules respectively:

For the Events Manager:

```
$ sudo service aeon_events start
$ sudo service aeon_events stop
$ sudo service aeon_events restart
$ sudo service aeon_events status
```

For the REST API:

```
$ sudo service aeon_rest start
$ sudo service aeon_rest stop
$ sudo service aeon_rest restart
$ sudo service aeon_rest status
```

For the Front-end:

```
$ sudo service aeon_frontend start
$ sudo service aeon_frontend stop
$ sudo service aeon_frontend restart
$ sudo service aeon_frontend status
```

The Cloud Messaging GE logs file are stored in the folder /var/log/ under the files:

/var/log/aeon_events.log

/var/log/aeon_rest.log

/var/log/aeon_frontend.log