**Server Handling - Documentation**

* **Server – Introduction:**

The VM is Linux Ubuntu machine. The Code base of the project is in newbackend\_rust directory. I have planned to create separate scripts for different types of restarting. It has two different components a small python component and a major rust component. Both of these work *independently together* during production.

* **Server – Structure:**

The entire server of Runamic consists of two components Python and RUST. The Rust part of the server is responsible for the major functions such as routing and thus the core algorithms lies here. The python part consist of the preprocessing of OSM data and some parts of the database communications (mainly related to POI and User table operations).   
More detailed documentation that would be useful for further development could be reached from the following link :

* **Steps To connect to the server:**

Step 1: Use putty to reach the server using the following SSH

If inside the ELIS network, use

*$ ssh* [*ugent\_elis\_name@tw06v061.ugent.be*](mailto:ugent_elis_name@tw06v061.ugent.be)

Password***:*** *ELIS unix password*

example :$ ssh [kthiruko@tw06v061.ugent.be](mailto:kthiruko@tw06v061.ugent.be).

If outside ELIS network, use the following SSH to get into ELIS network and then use the above procedure to connect to the server.

*$ ssh* [*ugent\_elis\_name@ssh.elis.ugent.be*](mailto:ugent_elis_name@ssh.elis.ugent.be)

**NOTE :** There is a development server that you can use for testing. This can be reached from the following : *$ ssh* [*ugent\_elis\_name@tw06v066.ugent.be*](mailto:ugent_elis_name@tw06v066.ugent.be)

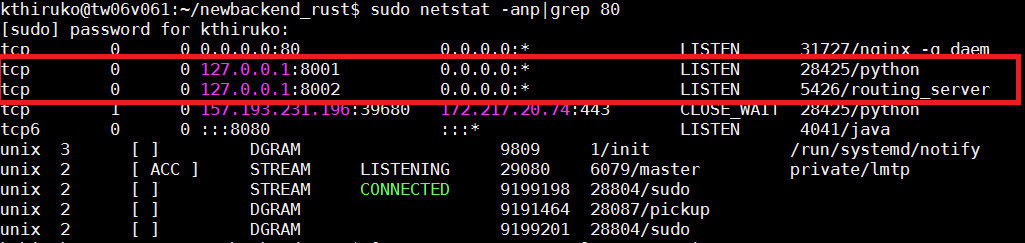
* **Steps to Check if the server is running:**

Step 1: Use the following the command to see if the server is running.

**$ sudo netstat –anp | grep 80**

If the server is running, in the list, **nginx –g daem**, ***routing\_server*** and ***python*** would be listed, as shown in figure.

***Note*** : As explained before, the entire backend server of Runamic comprises of two independent components namely the ***RUST*** and ***PYTHON*** components. Thus in the below example, *28425 / python* refers to the *python component* while *5426 / routing\_server* refers to the *RUST component*. The *Nginx –g daem* is the process that indicates the working of nginx, that is used as a reverse proxy.



* **Steps to stop a running server:**

Step 1: Check if the server running.

Step 2: From the list, find out process IDs of both *routing\_server* and *python* processes.

Step 3: Use the following command to Stop the running server.

**$ sudo kill -9 <process\_ID>**

In this example :$ sudo kill -9 28425.

* *Special Cases :*

There are certain special scenarios where we would require to delete only one component. In those cases only that specific component is killed, which would still keep the other component working.

* **Steps to restart the server:**

Step 1: Check if the server is still active. In case if the server is still running, it has to be stopped.

Step 2: Based on the category listed below, use the appropriate script to restart the server accordingly.

* **Restart entire server :**
  + To restart entire server, run the ***deploy.sh***script in *kthiruko/newbackend\_rust* directory.
* **Restart Python component alone :** 
  + To restart the Python component run the ***python\_deploy.sh***script in *kthiruko/newbackend\_rust* directory.
* **Restart RUST component alone :**
  + To restart the RUST component, run the ***rust\_deploy.sh***script in *kthiruko/newbackend\_rust* directory.
* **Steps to restart Nginx:**

Nginx is used as the reverse proxy for the server. To restart the nginx,

**$ sudo systemctl restart nginx**

* **Steps to add a new POI data to the Application:**

Note : ADDING a new POI to the server would currently CLEAR the DATABASE and repopulated with the new OSM data. SO it is highly recommended not to do coz it might clear out the user feedbacks for each edge.

Step 1: The POI json that has to be checked for the structure of it and only the right format of the POI data has to be used.

Step 2: The POI json is placed in the *kthiruko/newbackend\_rust/data/poisets.*

Step 3: From the *kthiruko/newbackend\_rust/data/* directory, **./update.sh** is used to add the new POI set to the Runamic appliation.

* **Data base Information - Postgres:**

We use a postgres database to store the data. To use postgres in the command line, use the following:

**$ sudo –u postgres psql**

Database\_Name: postgres

Password: idlab\_lopeningent

The schema for the server operations and data storage in Postgres is lopeningent2.

For more Detailed Documentations regarding the Database, Installation, server Structure and Request architecture, visit the following page:

<https://github.com/oSoc17/lopeningent_backend/wiki>