Valhalla setup on Ubuntu

I have the latest Ubuntu 23.04 installed on my laptop. The tutorials I found were for Ubuntu 20.04. I first tried making the tutorial work for 23.04, but due to continuously arising conflicts and errors during the setup, I decided to create a virtual machine with the recommended Ubuntu 20.04.

```
vaishnavnegi@vaishnavnegi:~$ lsb_release -a
No LSB modules are available.
Distributor ID: Ubuntu
Description:    Ubuntu 23.10
Release:        23.10
Codename:       mantic
```

Laying the Groundwork

- For virtualization I installed Gnome Boxes. After that I downloaded the iso image for Ubuntu 20.04 from the official Ubuntu website. Gnome can be installed simply using the command:

```
$ sudo apt-get install gnome-boxes
```

- After this we can simply select the downloaded iso image in the Boxes' interface and create the required Virtual Machine.
  - After we have created the VM, we will need to install and prepare our fresh OS by installing some important packages, tools and dependencies. Run the following commands as below.

```
rsuvi@rsuvi-Standard-PC-Q35-ICH9-2009:~$ sudo apt-get update
[sudo] password for rsuvi:
Hit:1 http://de.archive.ubuntu.com/ubuntu focal InRelease
Hit:2 http://security.ubuntu.com/ubuntu focal-security InRelease
Hit:3 http://ppa.launchpad.net/kevinkreiser/prime-server/ubuntu focal InRelease
```

- This will give you access to the latest available Ubuntu packages through apt-get.

```
rsuvi@rsuvi-Standard-PC-Q35-ICH9-2009:~$ sudo apt-get install -y git wget curl ca-certificates gnupg2
Reading package lists... Done
Building dependency tree
Reading state information... Done
curl is already the newest version (7.68.0-1ubuntu2.20).
```

- **git**: to clone repositories from GitHub.
- **wget, curl**: to download things from any source.
- **ca-certificates, gnupg2**: to load and verify packages from third-party sources.

```
rsuvi@rsuvi-Standard-PC-Q35-ICH9-2009:~$ sudo apt-get install -y cmake build-essential
Reading package lists... Done
Building dependency tree
Reading state information... Done
cmake is already the newest version (3.16.3-1ubuntu1.20.04.1).
```

- **cmake:** to configure C/C++ projects, like Valhalla and prime-server.
- **g++:** c++ compiler to build Valhalla.
- **build-essential:** holds all the packages needed to build Debian/Ubuntu packages. Helps build from source

```
rsuvi@rsuvi-Standard-PC-Q35-ICH9-2009:~$ sudo apt-get install -y libsqlite3-mod-spatialite libsqlite3-dev libspatialite-dev \
>                         autoconf libtool pkg-config libczmq-dev libzmq5 \
>                         libcurl4-openssl-dev zlib1g-dev jq libgeos-dev liblz4-dev \
>                         libgeos++-dev libprotobuf-dev protobuf-compiler \
>                         libboost-all-dev libluajit-5.1-dev spatialite-bin unzip
Reading package lists... Done
Building dependency tree
Reading state information... Done
autoconf is already the newest version (2.69-11.1).
libtool is already the newest version (2.4.6-14).
pkg-config is already the newest version (0.29.1-0ubuntu4).
libczmq-dev is already the newest version (4.2.0-2).
libzmq5 is already the newest version (4.3.2-2ubuntu1).
libzmq5 set to manually installed.
libcurl4-openssl-dev is already the newest version (7.68.0-1ubuntu2.20).
unzip is already the newest version (6.0-25ubuntu1.1).
```

- These are the remaining dependencies for building the prime_server and Valhalla. Most of the above packages are present on a newly installed Ubuntu, but we do this just to be on the safe side.

Setting up prime_server dependency for Valhalla

- Kevin Kreiser et al. who also developed Valhalla, developed a load-balancing web server to deal with the highly heterogeneous workloads of routing engine APIs, where classic load-balancing would fail. `prime_server` is a C++ library that provides a set of tools for building scalable, high-performance servers for handling network communication. It is often used in the development of geospatial data processing and routing applications. The library is designed to efficiently handle large amounts of data and requests, making it suitable for applications dealing with geographic information systems (GIS), mapping, and navigation.
- I set up prime_server in my `Home` directory following the Quick Start guide on the official GitHub repository of prime_server. It first instructed us to install some dependencies as follows:

```
# grab some standard autotools stuff
sudo apt-get install autoconf automake pkg-config libtool make gcc g++ lcov
# grab curl (for url de/encode) and zmq for the awesomeness
sudo apt-get install libcurl4-openssl-dev libzmq3-dev libczmq-dev
```

- After this step, we must clone the git repository of the prime_server and open the terminal from inside it. The following steps will then install prime_server for us:

```
# dont forget submodules
git submodule update --init --recursive
# standard autotools:
./autogen.sh
./configure
make test -j8
sudo make install
```

- **autogen.sh:** generates the source code to a compilable data source.
- **configure:** checks if all dependencies are fulfilled and the compile-ready data set is suitable for the current system.
- **make test:** checks if everything compiled correctly and **-j** switch allows you to run the compilation/tests with the specified number of cores.
- **sudo make install:** installs the prime_server.

Coming to the main course: Installing Valhalla

- Finally, we will clone the GitHub repo of Valhalla to move into it by changing directories. Right afterwards, we need to install its dependencies as shown below.

```
rsuvi@rsuvi-Standard-PC-Q35-ICH9-2009:~$ git clone https://github.com/valhalla/valhalla
Cloning into 'valhalla'...
remote: Enumerating objects: 102901, done.
remote: Counting objects: 100% (12406/12406), done.
remote: Compressing objects: 100% (1072/1072), done.
remote: Total 102901 (delta 11629), reused 11656 (delta 11267), pack-reused 90495
Receiving objects: 100% (102901/102901), 248.01 MiB | 7.74 MiB/s, done.
Resolving deltas: 100% (73417/73417), done.
rsuvi@rsuvi-Standard-PC-Q35-ICH9-2009:~$ cd valhalla
rsuvi@rsuvi-Standard-PC-Q35-ICH9-2009:~/valhalla$ ./scripts/install-linux-deps.sh
+ Ubuntu 20.04.6 LTS amd64                    --assume-yes
Hit:1 http://de.archive.ubuntu.com/ubuntu focal InRelease
Hit:2 http://security.ubuntu.com/ubuntu focal-security InRelease
```

- During the first build, I ran into the following error, due to inconsistency between the submodules of the cloned repo and the one we are trying to build:

```
CMake Error at src/CMakeLists.txt:164 (add_subdirectory):
  The source directory

    /home/rsuvi/valhalla/third_party/robin-hood-hashing

  does not contain a CMakeLists.txt file.
```

- It was easily resolved by running the following command. This will sync and give us the up-to-date versions of all needed and included submodules.

```
rsuvi@rsuvi-Standard-PC-Q35-ICH9-2009:~/valhalla$ git submodule update --init --recursive
Submodule 'OSM-binary' (https://github.com/scrosby/OSM-binary.git) registered for path 'thi
Submodule 'third_party/benchmark' (https://github.com/google/benchmark) registered for path
Submodule 'third_party/cpp-statsd-client' (https://github.com/vthiery/cpp-statsd-client) re
```

- Now you can build and install Valhalla, using the following sequence of commands.

```
rsuvi@rsuvi-Standard-PC-Q35-ICH9-2009:~/valhalla$ cmake -B build -DCMAKE_BUILD_TYPE=Release
-- The CXX compiler identification is GNU 9.4.0
-- The C compiler identification is GNU 9.4.0
-- Check for working CXX compiler: /usr/bin/c++
-- Check for working CXX compiler: /usr/bin/c++ -- works
```

- **cmake -B build -DCMAKE_BUILD_TYPE=Release**: configures the build for make (using config files) and **DCMAKE_BUILD_TYPE=Release** will compile the Release version.

```
rsuvi@rsuvi-Standard-PC-Q35-ICH9-2009:~/valhalla$ make -C build -j$(nproc)
make: Entering directory '/home/rsuvi/valhalla/build'
make[1]: Entering directory '/home/rsuvi/valhalla/build'
make[2]: Entering directory '/home/rsuvi/valhalla/build'
make[2]: Entering directory '/home/rsuvi/valhalla/build'
make[2]: Entering directory '/home/rsuvi/valhalla/build'
make[2]: Entering directory '/home/rsuvi/valhalla/build'
make[2]: Entering directory '/home/rsuvi/valhalla/build'
[  0%] Running cpp protocol buffer compiler on api.proto
Scanning dependencies of target valhalla-midgard
```

- **make -C build -j$(nproc)**: compiles and builds Valhalla with all available cores.

```
rsuvi@rsuvi-Standard-PC-Q35-ICH9-2009:~/valhalla$ sudo make -C build install
make: Entering directory '/home/rsuvi/valhalla/build'
make[1]: Entering directory '/home/rsuvi/valhalla/build'
make[2]: Entering directory '/home/rsuvi/valhalla/build'
make[2]: Leaving directory '/home/rsuvi/valhalla/build'
[  3%] Built target valhalla-midgard
make[2]: Entering directory '/home/rsuvi/valhalla/build'
make[2]: Leaving directory '/home/rsuvi/valhalla/build'
[ 10%] Built target valhalla-proto
make[2]: Entering directory '/home/rsuvi/valhalla/build'
```

- **sudo make -C build install**: installs Valhalla to the system (in the directory directory we configured with CMake)

*The above steps might require a fair amount of time to complete. So, patience was also required, so that we didn't kill the processes if they seemed stuck at some place for a while.

Configuring and Testing Valhalla

Let's first test our Valhalla installation by typing the following command:

```
rsuvi@rsuvi-Standard-PC-Q35-ICH9-2009:~/valhalla$ valhalla_build_config
{
  "additional_data": {
    "elevation": "/data/valhalla/elevation/"
  },
  "httpd": {
    "service": {
      "drain_seconds": 28,
      "interrupt": "ipc:///tmp/interrupt",
      "listen": "tcp://*:8002",
```

```
rsuvi@rsuvi-Standard-PC-Q35-ICH9-2009:~/valhalla$ valhalla_build_admins
Configuration is required

valhalla_build_admins 3.4.0

valhalla_build_admins is a program that creates a administrative SQLite database from
one or multiple osm.pbf files. The admin db is used during graph building to enrich
nodes and edges.
```

```
rsuvi@rsuvi-Standard-PC-Q35-ICH9-2009:~/valhalla$ valhalla_build_tiles -h
valhalla_build_tiles 3.4.0

a program that creates the route graph
from one or multiple osm.pbf extract(s)
```

```
rsuvi@rsuvi-Standard-PC-Q35-ICH9-2009:~/valhalla$ valhalla_service
2023/11/13 23:59:39.898045 [ERROR] Usage: valhalla_service config/file.json [concurrency]
2023/11/13 23:59:39.898134 [ERROR] Usage: valhalla_service config/file.json action json_request
```

- Now we can set up a little experiment to test Valhalla. We'll start by installing some tools:

```
rsuvi@rsuvi-Standard-PC-Q35-ICH9-2009:~/valhalla$ sudo apt-get install -y curl jq unzip spatialite-bin
Reading package lists... Done
Building dependency tree
Reading state information... Done
spatialite-bin is already the newest version (4.3.0-3build1).
```

- **jq**: is a tool to deal with JSON structures.
- **unzip**: for unzipping files(used internally by Valhalla).
- **spatialite-bin**: for the spatialite support of Valhalla to build timezone support and admin areas.

Next we'll download the script files and set up a working directory.

```
rsuvi@rsuvi-Standard-PC-Q35-ICH9-2009:~$ cd ~/valhalla/scripts/
rsuvi@rsuvi-Standard-PC-Q35-ICH9-2009:~/valhalla/scripts$ mkdir valhalla_tiles
&& mkdir conf
rsuvi@rsuvi-Standard-PC-Q35-ICH9-2009:~/valhalla/scripts$ curl -O https://downl
oad.geofabrik.de/europe/albania-latest.osm.pbf
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100 41.4M  100 41.4M    0     0  4169k      0  0:00:10  0:00:10 --:--:-- 4472k
rsuvi@rsuvi-Standard-PC-Q35-ICH9-2009:~/valhalla/scripts$ valhalla_build_config
 --mjolnir-tile-dir ${PWD}/valhalla_tiles --mjolnir-tile-extract ${PWD}/valhall
a_tiles.tar --mjolnir-timezone ${PWD}/valhalla_tiles/timezones.sqlite --mjolnir
-admin ${PWD}/valhalla_tiles/admins.sqlite > ${PWD}/conf/valhalla.json
rsuvi@rsuvi-Standard-PC-Q35-ICH9-2009:~/valhalla/scripts$ valhalla_build_admins
 --config ./conf/valhalla.json albania-latest.osm.pbf
2023/11/14 00:06:05.348169 [INFO] Parsing files: albania-latest.osm.pbf
2023/11/14 00:06:05.348305 [INFO] Parsing relations...
2023/11/14 00:06:06.326687 [INFO] Finished with 13 admin polygons comprised of
```

- **cd ~/valhalla/scripts/**: move onto the `scripts` directory inside the `valhalla` directory
- **valhalla_tiles**: holds processed Valhalla files.
- **conf**: holds the Valhalla config file.
- **curl -O https://download.geofabrik.de/europe/albania-latest.osm.pbf**: To build tiles for Valhalla, we download an OSM extract, for an arbitrary region from Geofabrik.
- Build your config file
- **valhalla_build_config --mjolnir-tile-dir ${PWD}/valhalla_tiles --mjolnir-tile-extract ${PWD}/valhalla_tiles.tar --mjolnir-timezone ${PWD}/valhalla_tiles/timezones.sqlite --mjolnir-admin ${PWD}/valhalla_tiles/admins.sqlite > ${PWD}/conf/valhalla.json**:
  The config file is the core part of the Valhalla setup and will hold all necessary (and optional) configurations and file paths, so Valhalla knows where to look for and where to store data.
  - **valhalla_build_config**: tool to build the config file.
  - **--mjolnir-tile-dir**: folder where the Valhalla tiles will be processed.
  - **--mjolnir-tile-extract**: file where the tarred Valhalla tiles will be stored.
  - **--mjolnir-timezone** : folder where the SQLite file holding the time zone areas is stored.
  - **--mjolnir-admin**: path where the SQLite file holding the admin areas will be stored.
  - **> ${PWD}/conf/valhalla.json**: output file path for the config file.

```
rsuvi@rsuvi-Standard-PC-Q35-ICH9-2009:~/valhalla/scripts$ valhalla_build_tiles -c ./conf/valhalla.json albania-latest.osm.pbf
2023/11/14 00:07:55.635303 [INFO] Running valhalla_build_tiles with 12 thread(s).
2023/11/14 00:07:55.635398 [INFO] Start stage = initialize End stage = cleanup
2023/11/14 00:07:55.637751 [INFO] Parsing files for ways: albania-latest.osm.pbf
2023/11/14 00:07:55.637956 [INFO] Parsing ways...
2023/11/14 00:08:03.250156 [INFO] Added 32 culdesac roundabouts from 105 candidates.
2023/11/14 00:08:03.250189 [INFO] Finished with 151786 routable ways containing 3314740 nodes
2023/11/14 00:08:03.266188 [INFO] Sorting osm access tags by way id...
2023/11/14 00:08:03.274275 [INFO] Finished
2023/11/14 00:08:03.276190 [INFO] Parsing files for relations: albania-latest.osm.pbf
2023/11/14 00:08:03.276277 [INFO] Parsing relations...
```

- The above command builds the routing tiles, i.e. the graph.

```
rsuvi@rsuvi-Standard-PC-Q35-ICH9-2009:~/valhalla/scripts$ find valhalla_tiles | sort -n | tar -cf "valhalla_tiles.tar" --no-recursion -T -
```

- And in the above step we aggregate all tiles into a tar file, which is more efficient for Valhalla's loading and caching processes.

- Now we can finally run Valhalla with our valhalla.json config file using the specified amount of cores as follows:

```
rsuvi@rsuvi-Standard-PC-Q35-ICH9-2009:~/valhalla/scripts$ valhalla_service ~/valhalla/scripts/conf/valhalla.json 2
2023/11/14 00:09:09.134648 [INFO] Tile extract successfully loaded with tile count: 235
2023/11/14 00:09:09.134676 [INFO] Tile extract successfully loaded with tile count: 235
2023/11/14 00:09:09.134648 [INFO] Tile extract successfully loaded with tile count: 235
2023/11/14 00:09:09.134675 [INFO] Tile extract successfully loaded with tile count: 235
2023/11/14 00:09:09.134822 [WARN] (stat): /data/valhalla/traffic.tar No such file or directory
2023/11/14 00:09:09.134829 [WARN] (stat): /data/valhalla/traffic.tar No such file or directory
2023/11/14 00:09:09.134838 [WARN] Traffic tile extract could not be loaded
2023/11/14 00:09:09.134825 [WARN] (stat): /data/valhalla/traffic.tar No such file or directory
2023/11/14 00:09:09.134858 [WARN] Traffic tile extract could not be loaded
2023/11/14 00:09:09.134881 [WARN] Traffic tile extract could not be loaded
2023/11/14 00:09:09.134825 [WARN] (stat): /data/valhalla/traffic.tar No such file or directory
2023/11/14 00:09:09.134903 [WARN] Traffic tile extract could not be loaded
0 2023/11/14 00:12:30.383873 POST /route HTTP/1.1
2023/11/14 00:12:30.384326 [INFO] Got Loki Request 0
2023/11/14 00:12:30.387537 [INFO] Got Thor Request 0
2023/11/14 00:12:30.387666 [INFO] algorithm::bidirectional_a*
2023/11/14 00:12:30.402960 [INFO] Got Odin Request 0
0 2023/11/14 00:12:30.466576 200 1813
^Z
[1]+  Stopped                 valhalla_service ~/valhalla/scripts/conf/valhalla.json 2
```

- While the service is running, we will open another terminal tab and write in the following command to test the service:

```
rsuvi@rsuvi-Standard-PC-Q35-ICH9-2009:~/valhalla/scripts$ curl http://localhost:8002/route \
> --data '{"locations":[
>               {"lat":41.318818,"lon":19.461336},
>               {"lat":41.321001,"lon":19.459598}
>           ],
>          "costing":"auto"
>         }' | jq '.'
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100  1855  100  1694  100   161  20166   1916 --:--:-- --:--:-- --:--:-- 22349
{
  "trip": {
    "locations": [
      {
        "type": "break",
        "lat": 41.318818,
        "lon": 19.461336,
        "original_index": 0
      },
      {
        "type": "break",
        "lat": 41.321001,
        "lon": 19.459598,
```