i2 Overwatch

i2 Overwatch transforms an existing deployment of i2 Analyst's Notebook into an enhanced solution that effectively supports the end-to-end intelligence lifecycle, building a mission-critical intelligence platform for military operations.

This documentation guides you through installing and deploying i2 Overwatch, and how to set up your own local map server.

Installing i2 Overwatch

You can install i2 Overwatch using an Installation Manager.

Before you begin

Before you begin the installation, you should review the i2 Overwatch release notes to ensure your setup meets the prerequisites and system requirements.

Procedure

To install the i2 Overwatch plug-in, follow these steps:

1. Extract the product files from your downloaded distribution.
2. From Windows Explorer, navigate to the root of the distribution and run the `Setup.exe` file to start the installation process.
3. Follow the prompts in the Installation Manager, and select the setup type, for example, **Typical** or **Complete**.
   - **Typical**: Installs i2 Overwatch along with its documentation.
   - **Complete**: Installs the i2 Overwatch plug-in.
4. Continue through the prompts to complete the installation.

Errors

If you don't have the expected version of Analyst's Notebook and the expected Fixpack/Test-fix installed, you will see an error message.

Before re-running the installation, ensure you have the following software installed:

1. i2 Analyst's Notebook 10.1.0
2. Analyst's Notebook 10.1.0 Test Fix 1

Additionally, if you plan to use Overwatch with specific i2 products, please verify the following:

- To run Overwatch over iBase, ensure you have iBase 10.1.1 installed.
- To run Overwatch using an Analysis Hub or Studio offering, ensure you have i2 Analyze 4.4.5 installed.

Installation and application data folders

You can install i2 Overwatch in a suggested folder, or you can choose a different folder.

During the installation of i2 Overwatch, the installer recommends a default installation folder of `C:\Program Files\i2 Overwatch`. However, you have the option to choose a different location

if preferred. Regardless of the chosen folder, the application will automatically store its data in the designated application data folder, which is determined by your version of Microsoft Windows. Note that this folder is typically hidden from view in Windows.

Installing i2 Overwatch from the command line

i2 Overwatch is installed using a Microsoft Windows Installer. You can use the msi command line options to install Overwatch components.

Before you begin

Before you install Overwatch from the command line, ensure you have all the Overwatch prerequisites installed.

Procedure

To install the i2 Overwatch using the command line:

1. Open a command prompt with administrator privileges.
2. Navigate to the location of the Overwatch msi file. **Note**: You can also provide the absolute file path to the msi file.
3. Enter the command that specifies the components you would like to install in the following format:

   `msiexec /i "i2 Overwatch.msi" <ADDITIONAL OPTIONS>`

   Where the additional options refer to the standard windows installer command line options, for example;

   INSTALLLEVEL - specify the install level of a feature set. Using this option ensures that you get all options at a level and all the options available at lower levels, ensuring all prerequisite features are present.

   ADDLOCAL - specify specific features to install. For a list of the Microsoft specific options use `msiexec /h`.

   Examples:

   Install Overwatch silently:

   ```
   msiexec /i "i2 Overwatch.msi" /qn
   ```

Setting up i2 Overwatch for your i2 deployment

There are a number of different tasks that are required to set up i2 Overwatch, depending on your i2 deployment

Standalone i2 Analyst's Notebook (ANB)

i2 Overwatch comes with a template that is used by default for standalone Analyst's Notebook deployments. If you require your own template, you can use the semantic type library `Example Material\Semantic Type Library\OverwatchSemanticLibrary.mtc` provided in the installation folder. You can find out more about semantic types in Analyst's Notebook, here.

i2 Analyze

i2 Overwatch comes with an example `Example Material\i2 Analyze\OverwatchSchema.xml` schema that you can choose to build on, or use as a base template for your own schema. It is important

to note that for your deployment to fully use the Overwatch features, you must use the Overwatch semantic type library. The Overwatch semantic Type library ships within the i2 Analyze schema. Learn about assigning semantic types in i2 Analyze here.

i2 iBase

i2 Overwatch comes with an iBase `Example Material\i2 iBase\Overwatch Template.adt` file that contains all the entities and links that Overwatch requires. You can use and build on this template, or you can create your own new template from the `Example Material\Semantic Type Library\OverwatchSemanticLibrary.mtc` provided in the Overwatch installation. See the iBase documentation for more information about the semantic type library here.

Deploying a local map server

You can set up a local OpenStreetMap (OSM) tile server to use with i2 Overwatch.

The mapping functionality within i2 products can be configured to use both online and local map servers.

One option for a local map server is an OpenStreetMap server with Open Street Maps data which covers many regions of the world.

Deploying a local OpenStreetMap server

As described at Switch2OSM an OpenStreetMap server can be deployed in a variety of ways.

1. Use pre-built OpenStreetMap image running on WSL. This option imports an Ubuntu distribution with an OpenStreetMap server pre-loaded with map data for the UK.
2. Use Docker. The installation process is straightforward, and the fastest path to getting a map server up and running.
3. Install on Ubuntu 24.04. More steps are required for installation, and require some Linux administration skills.
4. Use Windows Subsystem for Linux. This is equivalent to option 2, but allows for installation on a Windows computer.

Use pre-built OpenStreetMaps image running on WSL

A Windows 11 x64 bit Ubuntu distribution with a pre-installed OpenStreetMaps server is available.

1. Download the tar file to a temporary folder, such as `c:\temp\UbuntuOSM-UK.tar`
2. Import the tar file.

   ```
   wsl --import OpenStreetMaps %LOCALAPPDATA%\Packages\OpenStreetMaps C:\temp
   \UbuntuOSM-UK.tar
   ```
3. Wait for the import to complete.
4. Check that the import was successful by running the following command:

   ```
   wsl -l -v
   ```
5. Start the OpenStreetMaps image.

   ```
   wsl -d OpenStreetMaps
   ```

   You will be prompted for the super user password. The default password is `cyclone`.
6. Check for correct operation using a browser.

```
http://localhost:8080/sample_leaflet.html
```

7. OPTIONAL: Change the password for the `osm` username. The default password for the `osm` username is `cyclone`. Ensure you are logged in as the `osm` username, and choose a new password.

   ```
   passwd
   ```

   You will see the following prompts:

   ```
   Changing password for [username]
   (current) UNIX password:
   Enter new UNIX password:
   Retype new UNIX password:
   passwd: password updated successfully
   ```

Use Docker

You can use Docker to run the Open Street Map tile server on a Windows machine using the Windows Subsystem for Linux (WSL) or directly on an Ubuntu server. This approach allows you to quickly set up and run the tile server without needing to install all dependencies manually.

This example assumes that Docker is already deployed on a Windows machine and that Ubuntu 22.04 LTS is the distribution in use.

These steps are described in more detail on the Using a Docker container page at Switch2OSM.

1. Start the Ubuntu application on your Windows machine.
2. Download data for your region of interest to the home directory of the Ubuntu user (assumed to be *renderaccount* in this example).

   ```
   wget https://download.geofabrik.de/europe/united-kingdom-latest.osm.pbf
   ```

3. Create a docker volume for the data:

   ```
   docker volume create osm-data
   ```

4. Install it and import the data:

   ```
   docker run  -v /home/renderaccount/united-kingdom-latest.osm.pbf:/data/
   region.osm.pbf  -v osm-data:/data/database/ -v osm-tiles:/data/tiles/
    overv/openstreetmap-tile-server  import
   ```

5. Check for the following message at the end of the process:

   ```
   INFO:root:  Import complete
   ```

6. Start the tile server:

   ```
   docker run -p 8080:80 -v osm-data:/data/database --shm-size="192m" -v osm-
   tiles:/data/tiles/ -d overv/openstreetmap-tile-server run
   ```

   Note: The amount of shared memory for the docker container has been increased to 192 MB to ensure successful processing of the *united-kingdom-latest.osm.pbf* data. It might be necessary to further increase the amount of shared memory for other larger datasets.

7. Check to see that it is working by browsing to:

   ```
   http://your.server.ip.address:8080
   ```

8. If this installation is on your own computer, you can browse to:

   ```
   http://localhost:8080
   ```

Install on an Ubuntu 24.04 server

This topic provides detailed information about how to set up Open Street Maps on a Ubuntu 24.04 server.

Enable Ubuntu to download postgis

These steps to make the postgis repos visible were sourced from the trac.osgeo.org wiki

```
sudo apt install ca-certificates gnupg

curl  https://www.postgresql.org/media/keys/ACCC4CF8.asc | gpg --dearmor |
 sudo tee /etc/apt/trusted.gpg.d/apt.postgresql.org.gpg >/dev/null

sudo sh -c 'echo "deb http://apt.postgresql.org/pub/repos/apt $(lsb_release
 -cs)-pgdg main" > /etc/apt/sources.list.d/pgdg.list'
```

Install the required packages

```
sudo apt update

sudo apt upgrade

sudo apt install screen locate libapache2-mod-tile renderd git tar unzip
 wget bzip2 apache2 lua5.1 mapnik-utils python3-mapnik python3-psycopg2
 python3-yaml gdal-bin npm postgresql-14 postgresql-contrib postgis
 postgresql-14-postgis-3 postgresql-14-postgis-3-scripts osm2pgsql net-tools
 curl
```

Set up the PostgreSQL database

1. As the `postgres` user create a new PostgreSQL user called `_renderd` and then create the `gis` database.

   ```
   sudo service postgresql start

   sudo -u postgres -i

   createuser _renderd

   createdb -E UTF8 -O _renderd gis
   ```

2. While still working as the "postgres" user, set up PostGIS on the PostgreSQL database:

   ```
   psql
   ```

   This takes you to a "postgres=#" prompt.

   ```
   \c gis
   ```

   You should see the message, "You are now connected to database 'gis' as user 'postgres'".

3. Create extensions and tables.

   ```
   CREATE EXTENSION postgis;
   ```

   You see a response of CREATE EXTENSION

   ```
   CREATE EXTENSION hstore;
   ```

   You see a response of CREATE EXTENSION

   ```
   ALTER TABLE geometry_columns OWNER TO _renderd;
   ```

   You see a response of ALTER TABLE

```
ALTER TABLE spatial_ref_sys OWNER TO _renderd;
```

You see a response of ALTER TABLE

4. Quit Postfres, and then exit back to the non-root user.

```
\q
exit
```

Confirm that Mapnik has been installed

1. As the non-root user check for Mapnik. The second chevron prompt indicates success.

```
python3
>>> import mapnik
>>>
```

2. Exit python.

```
quit()
```

Configure the mapping stylesheet

1. Perform the following steps as the non-root user.

```
mkdir ~/src
cd ~/src
git clone https://github.com/gravitystorm/openstreetmap-carto
cd openstreetmap-carto
git pull --all
git switch --detach v5.9.0
```

2. Install the Carto compiler.

```
sudo npm install -g carto
carto -v
```

The response should be a number that is at least as high as: `1.2.0`

3. Convert the carto project.

```
carto project.mml > mapnik.xml
```

Loading data

Mapping data for the United Kingdom has been used as an example. Further data is available from sites like Geofabrik.

1. Create a data directory in the non-root user's home directory.

```
mkdir ~/data
cd ~/data
wget https://download.geofabrik.de/europe/united-kingdom-latest.osm.pbf
```

2. Allow other accounts to access the non-root user's home directory.

```
chmod o+rx ~
```

Note: If you don't want to grant this access you can change the location of the data, and adjust the following commands.

3. Use the `_renderd` Postgres account to load the data into the `gis` database.

```
sudo -u _renderd osm2pgsql -d gis --create --slim  -G --hstore --tag-
transform-script ~/src/openstreetmap-carto/openstreetmap-carto.lua -C
 2500 --number-processes 1 -S ~/src/openstreetmap-carto/openstreetmap-
carto.style ~/data/united-kingdom-latest.osm.pbf
```

This will take a while, perhaps 30 minutes depending on your hardware.

4. Create additional indexes.

   ```
   openstreetmap-carto/

   sudo -u _renderd psql -d gis -f indexes.sql
   ```

   You should see the `CREATE INDEX` response 16 times

5. Add database functions.

   ```
   sudo -u _renderd psql -d gis -f functions.sql
   ```

   You should see the `CREATE FUNCTION` response 3 times

6. Add Shapefile data.

   ```
   cd ~/src/openstreetmap-carto/

   mkdir data

   sudo chown _renderd data

   sudo -u _renderd scripts/get-external-data.py
   ```

7. Add fonts.

   ```
   cd ~/src/openstreetmap-carto/

   scripts/get-fonts.sh
   ```

   Note: You might see errors reporting that some fonts are not available, however these can be safely ignored.

Setting up the web server

1. Update the renderd configuration file.

   ```
   sudo nano /etc/renderd.conf
   ```

2. Change the number of threads from '4' to '24'.

   ```
   [renderd]
   stats_file=/run/renderd/renderd.stats
   socketname=/run/renderd/renderd.sock
   ; num_threads=4
   num_threads=24
   tile_dir=/var/cache/renderd/tiles
   ```

3. Add the rendering style, and make sure to change *accountname* to the non-root user, such as *osm*.

   ```
   [s2o]
   URI=/hot/
   XML=/home/accountname/src/openstreetmap-carto/mapnik.xml
   HOST=localhost
   TILESIZE=256
   MAXZOOM=20
   ```

4. Save and exit nano.

5. Update the Apache mod-tile configuration.

   ```
   cd /etc/apache2/conf-available/

   sudo wget https://raw.githubusercontent.com/openstreetmap/mod_tile/python-
   implementation/etc/apache2/renderd.conf
   ```

```
sudo a2enconf renderd

sudo systemctl reload apache2
```

6. Change the Apache web server listening port from '80' to '8080' within the default site configuration.

   ```
   sudo nano /etc/apache2/sites-available/000-default.conf
   ```

   Change `<VirtualHost *:80>` to `<VirtualHost *:8080>`

   Save and exit nano.

7. Change the Apache web server listening port from `80` to `8080` within the ports configuration.

   ```
   sudo nano /etc/apache2/ports.conf
   ```

   Change `Listen 80` to `Listen 8080`

   Save and exit nano.

Set up logging for the renderd service

1. Edit the service definition.

   ```
   sudo nano /usr/lib/systemd/system/renderd.service
   ```

2. Add the logging directive to .

   ```
   [Section]
   Environment=G_MESSAGES_DEBUG=all
   ```

3. Save and exit the nano editor.
4. Reload the render daemon.

   ```
   sudo systemctl daemon-reload
   sudo systemctl restart renderd
   sudo systemctl restart apache2
   ```

Viewing tiles

We can use `sample_leaflet.html` to view a very simple map.

1. Get the sample file.

   ```
   cd /var/www/html

   sudo wget https://raw.githubusercontent.com/SomeoneElseOSM/mod_tile/
   switch2osm/extra/sample_leaflet.html
   ```

2. Edit `sample_leaflet.html`

   ```
   sudo nano sample_leaflet.html
   ```

   - Change the IP address and port matches your server address rather than `127.0.0.1`. e.g. localhost:8080
   - Replace:

     ```
     var map = L.map('map').setView([40.36629, 49.83335], 18);
     ```

     with

     ```
     var map = L.map('map').setView([51.5,0], 5);
     ```

3. Save and exit the nano editor.
4. Navigate to `http://localhost:8080/sample_leaflet.html`

Accessing the tile server triggers map tile generation, which might take a few seconds per tile to begin with. The rendered tiles are cached so subsequent performance is much faster.

Use Windows Subsystem for Linux

If not already installed, install the Windows Subsystem for Linux (WSL) on a Windows computer.

1. In an administrative Command Prompt window, run the following command:

   ```
   wsl --install
   ```
2. Restart the computer when prompted to complete the installation.
3. After you restart, open the distribution. Press **Start > wsl**.
4. You are prompted to create a new username and password for the Linux distribution.

For more information, see Set up your Linux username and password.

To configure the WSL Ubuntu image, access the image running on WSL and follow the steps for Install on an Ubuntu 24.04 server.

Using the tile server with your i2 product

You can use the tile server with your i2 product, such as i2 Analyst's Notebook, i2 Analysis Studio, or i2 Analysis Hub.

i2 Analyst's Notebook

i2 Analyst's Notebook uses the configuration specified in `C:\ProgramData\i2\Overwatch\GeospatialConfiguration.json`

By default this contains a mapConfig that defines a basemap at `http://localhost:8080/hot/`, (the defaults used in the preconfigured WSL image).

```
"baseMaps": [
 {
  "attribution": "",
  "displayName": "Local Tile Server",
  "id": "LocalTileServer",
  "url": "http://localhost:8080/hot/{z}/{x}/{y}.png",
  "minZoom": 4,
  "maxZoom": 10
 }
],
```

Note: The coordinate order of the `{z}/{x}/{y}` parameters in the URL differs from other map providers such as Esri.

If your tile server is configured with a different URL, then update the configuration of the basemap, save the file and restart i2 Analyst's Notebook.

i2 Analysis Studio or i2 Analysis Hub

i2 Analyze is the application server for both *i2 Analysis Studio* and *i2 Analysis Hub*.

To configure use of the tile server, update your i2 Analyze configuration using the following steps:

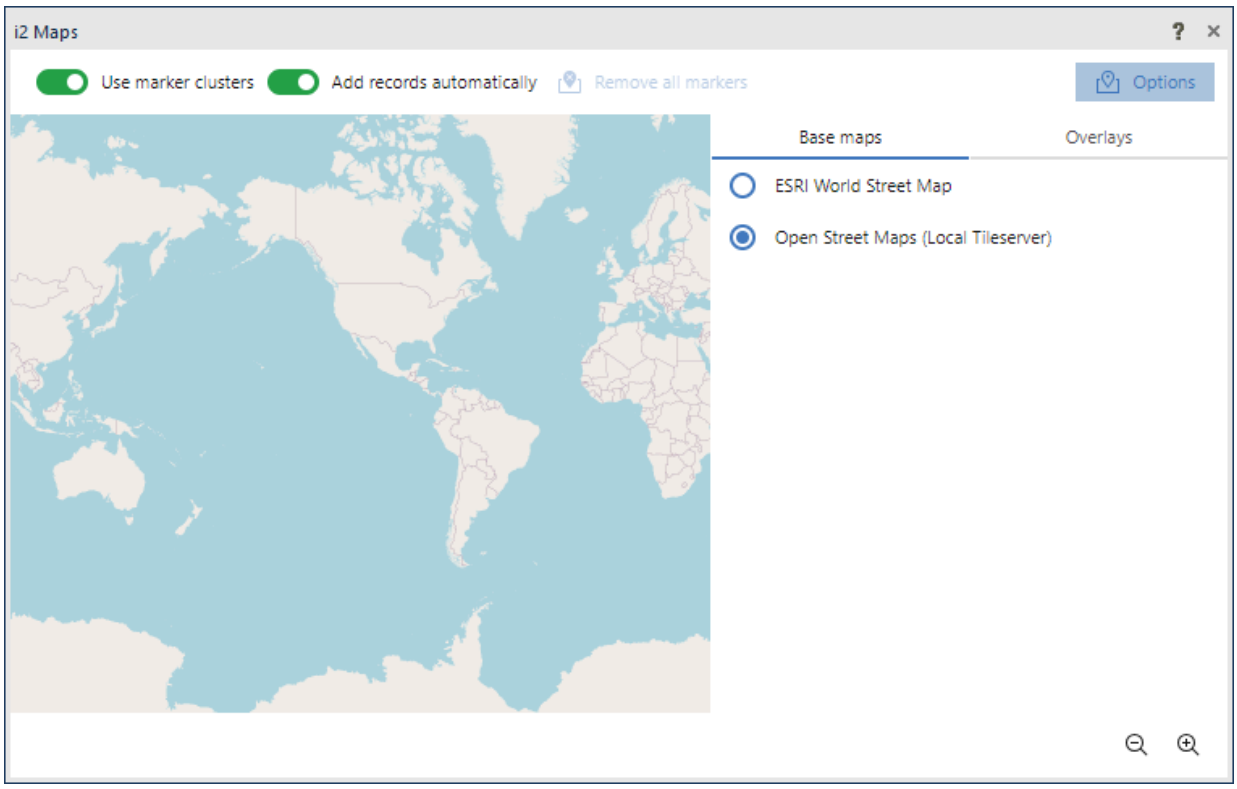
1. Add a new entry to the `baseMaps` object in the `geospatial-configuration.json` file replacing `your.server.ip.address` with the correct IP address of your tile server:

```
{
    "id": "OpenStreetMaps",
```

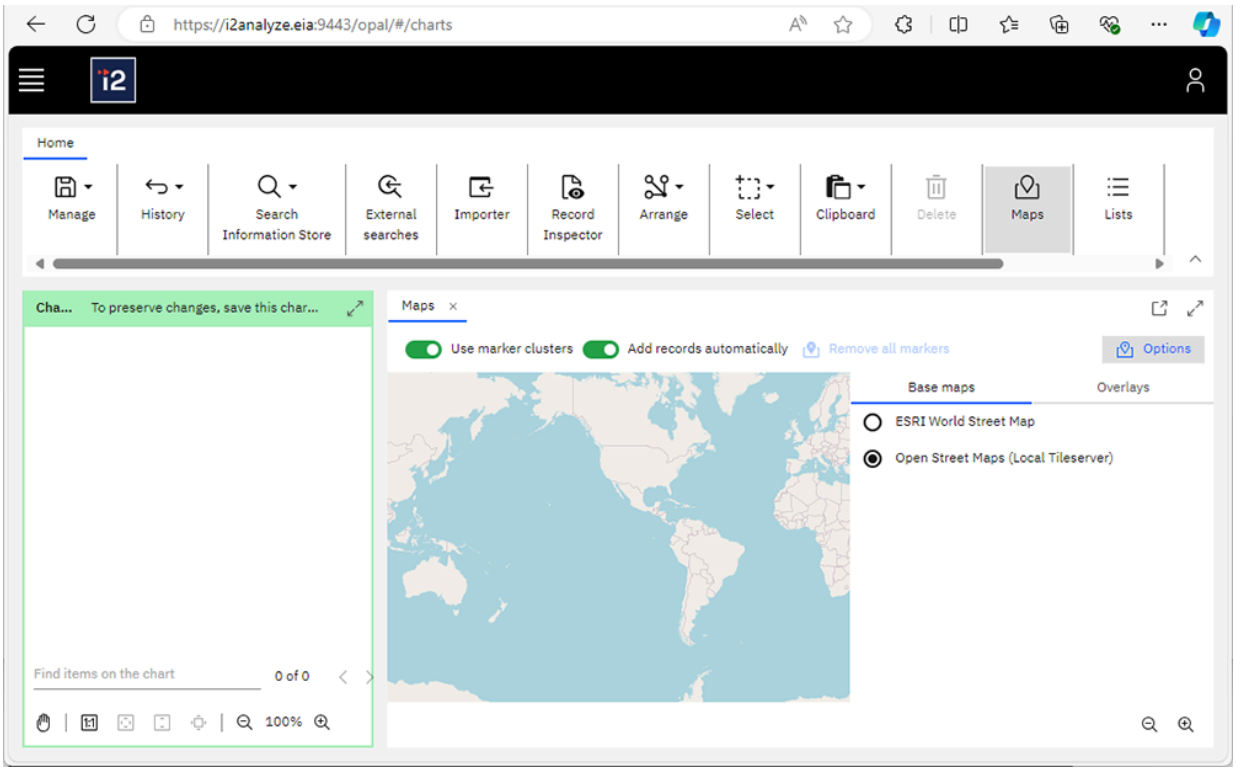
```
    "displayName": "Open Street Maps (Local Tileserver)",
    "url": "http://your.server.ip.address/hot/{z}/{x}/{y}.png",
    "attribution": "Sources: © OpenStreetMap contributors"
}
```

Note: The coordinate order of the `{z}/{x}/{y}` parameters in the URL differs from other map providers such as Esri.

1. Open *i2 Analyst's Notebook* and login to your i2 Analyze server.
2. Open i2 Maps, click the **Options** button, and then select `Open Street Maps (Local Tileserver)` on the `Base maps` tab.



1. Similarly set the **Options** in the web client:



Troubleshooting

Sometimes the tile server may not respond as expected. If you are experiencing issues with the tile server, follow these steps to troubleshoot.

In the event that http://localhost:80/sample_leaflet times out, or fails to respond, perform the following checks.

Confirm that the web server is running

```
service apache2 status
```

You should see a response similar to the following:

```
# apache2.service - The Apache HTTP Server
    Loaded: loaded (/usr/lib/systemd/system/apache2.service; enabled;
 preset: enabled)
    Active: active (running) since Wed 2024-12-11 10:43:14 AWST; 23min ago
    Docs: https://httpd.apache.org/docs/2.4/
    Process: 152 ExecStart=/usr/sbin/apachectl start (code=exited, status=0/
SUCCESS)
Main PID: 233 (apache2)
    Tasks: 55 (limit: 38320)
    Memory: 31.4M ()
    CGroup: /system.slice/apache2.service
            ##233 /usr/sbin/apache2 -k start
            ##235 /usr/sbin/apache2 -k start
            ##237 /usr/sbin/apache2 -k start
```

If the web server is not running correctly then restart it

```
sudo service apache2 restart
```

If tiles are not being rendered check the status of the `renderd` daemon

```
service renderd status
```

You should see a response similar to the following:

```
# renderd.service - Daemon that renders map tiles using mapnik
    Loaded: loaded (/usr/lib/systemd/system/renderd.service; enabled;
 preset: enabled)
    Active: active (running) since Wed 2024-12-11 10:43:14 AWST; 31min ago
    Docs: man:renderd
Main PID: 166 (renderd)
    Tasks: 6 (limit: 38320)
    Memory: 1.6G ()
    CGroup: /system.slice/renderd.service
            ##166 /usr/bin/renderd -f
```

If the `renderd` daemon is not running correctly then restart it, followed by restarting `apache2`

```
sudo service renderd restart

sudo service apache2 restart
```