INNOVATIVE
SOLUTION PVT.LTD.

Base IMIS

Deployment Manual

Integrated Municipal Information System (IMIS)

Innovative Solution Pvt. Ltd (ISPL)

Base IMIS Deployment Manual

Version 1.0.0

Table of Contents

Deploying Base IMIS on a server with the Linux operating system involves a series of steps and procedures. The steps mentioned below provide a guideline on the deployment process.

1.   BASIC REQUIREMENTS

The deployment manual is based on the specific system requirements mentioned below and any deviations will require adjustments to the deployment process. The specific system requirements mentioned below are recommended to ensure a smooth and error-free deployment process. The web-application and database are recommended to be deployed and maintained in two separate servers of identical specifications. However, both the web application and database can be maintained on the same server with minor modification to the deployment process.

1.1   Server Requirements

The minimum server requirements recommended for the successful deployment and implementation of IMIS are mentioned below:

Operating System

The deployment of Base IMIS is carried out on a Linux Server running Ubuntu 22.04 LTS "Jammy Jellyfish". This is the recommended server and operating system specifications.

Disk Space

The recommended minimum disk space for IMIS is 100 GB during the initial deployment. However, depending on the volume of the data and media files (images, GIS information, etc.) of the system, this requirement can be scaled up or scaled down as required.

RAM

The recommended minimum RAM is 8 GB during the initial deployment. However, depending on the size and complexity of the GIS data and spatial processing, expected traffic and number of users, this requirement can be further scaled up. Additionally, further scaling down the RAM is not recommended.

CPU

A minimum of 4 cores CPU is recommended for Base IMIS. However, depending on the size and complexity of the GIS data and spatial processing, expected traffic and number of users, this requirement can be further scaled up. Additionally, further scaling down the CPU cores is not recommended.

Network Bandwidth

A minimum network bandwidth of 100 Mbps speed with 500 GB to 1 TB monthly data transfer is recommended. However, depending on the number of users of the system and expected traffic, this requirement can be further scaled up or down as required.

1.2   Software Requirements

The software requirements for IMIS are mentioned below and its corresponding installation procedures are mentioned in sections below.

Web Server

The webserver recommended and currently used in this deployment manual is Nginx (V 1.22.0). However, the Apache web servers can also be used with modifications.

PHP

IMIS is developed using PHP version >= 8. This is the recommended version of PHP and any upgrades/downgrades require modifications to both the deployment process and source code.

Database

IMIS is designed and developed with PostgreSQL (V 14) database. For GIS data storage and processing, the PostGIS extension (V3) is used.

Geoserver

IMIS currently uses Geoserver (V2.21.0) for rendering and displaying spatial data maintained in the system. This specific version of Geoserver is recommended to ensure bugs/issues do not arise in the system.

1.3   Deployment Engineer Requirements

The recommended skillset for the deployment engineer conducting the deployment of IMIS is mentioned below. These skillsets are critical to ensure smooth and error free deployment and maintenance of IMIS.

- Linux Server with Ubuntu OS

- Docker Container Concepts

- Docker-Compose tool for multi-container applications

- Networking concepts and Port configurations

- Firewall concepts and rules

- PostgreSQL

- PostGIS extension of PostgreSQL

- Geoserver

- Shell scripting knowledge

Apart from the deployment engineer, the minimum recommended skillset for the development/ maintenance team is mentioned below:

- PHP language

- Laravel Framework

- PostgreSQL

- PostGIS extension of PostgreSQL

- NPM package manager

- Git version control

- Geoserver

- JavaScript

- jQuery

- Open Layers (OL)

- Technical Documentation

2. SECURED SERVER SETUP

The following steps are recommended to be carried out to ensure the server is secure and protected from external access/ intrusions and disruptions.

2.1 User Access Management

The root user is not recommended to be used for the server setup. Hence, a new user must be created. To add a new user and grant them superuser privileges, these steps are to be followed:

```
# adduser <<username>>
```

If root access is not available:

```
# sudo adduser <<username>>
```

Password prompt is displayed. Re-type the password to confirm password. Then fill out the user information as prompted. Press "Y" to continue.

```
Changing the user information for laravel
Enter the new value, or press ENTER for the default
        Full Name []:
        Room Number []:
        Work Phone []:
        Home Phone []:
        Other []:
```

Add the user "<<username>>" to the "sudo" group to grant them superuser privileges:

```
usermod -aG sudo <<username>>
```

If root access is not available:

```
sudo usermod -aG sudo <<username>>
```

2.2 Firewall Setup

To configure the firewall (ufw) to allow SSH connections:

See which applications are registered with ufw/firewall.

```
ufw app list
```

There should be "OpenSSH" in the list.

Allow OpenSSH through the firewall and enable the firewall.

[Note: Since the firewall blocks all connections except those explicitly allowed in the firewall rules, it is crucial to verify that the rules are accurate before enabling the firewall. Incorrect or missing rules could result in losing access to the server, so proceed with caution. For more information: *UFW Essentials* **guide.]**

```
ufw allow OpenSSH
ufw enable
```

Type Y and press ENTER to proceed. This will activate the firewall.

To check the status of the firewall and ensure that SSH connections are allowed, type:

```
ufw status
```

2.3   Secure SSH

The Secure SSH method for authentication is recommended for remove access and management of servers. The following steps are required to configure Secure SSH.

Use SSH key-based authentication:

Generate an SSH key pair on the local machine and copy the public key to ~/.ssh/authorized keys on the server.

Disable root login:

Edit the SSH configuration file (/etc/ssh/sshd_config) and set PermitRootLogin no.

[Note: This can be risky, so exercise caution and ensure that other login options are functioning correctly before proceeding.]

Change default SSH port:

Modify the SSH port in the SSH configuration file (Port <custom_port>) and restart the SSH service.

2.4 SSL Setup

SSL is recommended to ensure data security and prevent attacks. To secure your application with SSL, an SSL certification is required with the following certificates:

- private.pem (private key)
- fullchain.pem (full certificate)

SSL is required for both the application and the geoserver as well, thus two SSL certificates are required for IMIS. Additionally, to implement SSL, the corresponding domain/sub-domain names are also required for the web application and geoserver, that is mapped to the corresponding IP addresses of the servers. This procedure should be carried out at the end of the deployment process, after the deployment process is completed.

Configure Nginx for SSL in Docker

Modify Nginx configuration in Dockerfile to enable HTTPS. For more details refer to Section 6.10 section below.

Configure GeoServer to Use SSL

Configure GeoServer to use SSL; otherwise, it may create issues while displaying layers and styles. For more details refer to Section 6.11 below.

Restart Nginx and GeoServer

Rebuild the Docker:

```
(Docker Compose version 2)
docker compose down
docker compose up -d --build


(Docker Compose version 1)
docker-compose down
docker-compose up -d --build
```

Restart the Tomcat service:

```
sudo systemctl restart tomcat
```

6

2.5   Data Backup

The 3-tier backup strategy is recommended to ensure protection of data and ensure quick recovery in case of data loss. The 3-tier backup strategy is mentioned below:

Tier 1: Primary Backup (On-Site Daily Backup)

The Primary Backup is to be carried out daily, which is stored on-site (local server or external storage device). This provides quick access to the data for immediate recovery.

Tier 2: Secondary Backup (Off-site or Cloud Backup)

The Secondary Backup is to be carried out weekly, which is stored in a remote location or a separate cloud server. This protects the data from local disasters and data loss from main servers. This ensures a copy of the data is present even when the main server and primary backups are lost.

Tier 3: Tertiary Backup (Archival)

The Tertiary Backup is to be carried out monthly, which is stored in a highly secure and often offline or low-access environment. This enables long term retention of historical data, geo-redundancy of data and resilience against large scale incidents.

3.   DOCKER SETUP

The Web-Application is deployed using Docker and its containerized approach. To manage multiple docker containers and its configurations, docker-compose has been implemented. The necessary steps to be followed to deploy the Web-Application are mentioned below. The docker image provided will automatically install the required software packages such as nginx server, PHP, Composer, Laravel, etc.

[Note: The web-application can also be deployed through the conventional process, however the docker approach is recommended to ensure a smooth deployment process.]

3.1   Docker Installation

First, update the existing list of packages:

```
sudo apt update
```

Next, install a few prerequisite packages which let `apt` use packages over HTTPS:

```
sudo apt install apt-transport-https ca-certificates curl software-properties-common
```

Then add the GPG key for the official Docker repository to the system:

```
sudo apt install apt-transport-https ca-certificates curl software-properties-common
```

Download the GPG key and save it to a keyring file:

```
curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo gpg --dearmor -o /usr/share/keyrings/docker-archive-keyring.gpg
```

Add the Docker repository to APT sources:

```
echo "deb [arch=$(dpkg --print-architecture) signed-by=/usr/share/keyrings/docker-archive-keyring.gpg] https://download.docker.com/linux/ubuntu $(lsb_release -cs) stable" | sudo tee /etc/apt/sources.list.d/docker.list > /dev/null
```

Update the existing list of packages again for the addition to be recognized:

```
sudo apt update
```

Make sure the Docker repo is being installed instead of the default Ubuntu repo:

```
sudo apt-cache policy docker-ce
```

A similar output will be shown, although the version number for Docker may be different:

Output of apt-cache policy docker-ce:

```
docker-ce:
  Installed: (none)
  Candidate: 5:24.0.5-1~ubuntu.22.04~jammy
  Version table:
     5:24.0.5-1~ubuntu.22.04~jammy 500
        500 https://download.docker.com/linux/ubuntu jammy/stable amd64 Packages
     5:24.0.4-1~ubuntu.22.04~jammy 500
        500 https://download.docker.com/linux/ubuntu jammy/stable amd64 Packages
```

Notice that `docker-ce` is not installed, but the candidate for installation is from the Docker repository for Ubuntu 22.04 (`jammy`).

Finally, install Docker:

```
sudo apt install docker-ce
```

Docker should now be installed, the daemon started, and the process enabled to start on boot. Check that it's running:

```
sudo systemctl status docker
```

If the docker is inactive, start docker using following command:

```
sudo systemctl start docker
```

9

3.2   Executing the Docker Command Without Sudo

To create the docker group and add the user:

Create a docker group.

```
sudo groupadd docker
```

Add the user to the docker group.

```
sudo usermod -aG docker ${USER}
```

To ensure that group membership is re-evaluated, it's necessary to either log out and log back in, or use the following command:

```
su -s ${USER}
```

Confirm that the user is now added to the docker group by typing:

```
groups
```

Add the user to the docker group.

```
sudo usermod -aG docker username
```

3.3   Docker Compose Installation

Use the following command to download:

```
mkdir -p ~/.docker/cli-plugins/

curl    -SL    https://github.com/docker/compose/releases/download/v2.3.3/docker-compose-linux-x86_64 -o ~/.docker/cli-plugins/docker-compose
```

Next, set the correct permissions so that the `docker compose` command is executable:

```
chmod +x ~/.docker/cli-plugins/docker-compose
```

To verify that the installation was successful, run:

```
docker compose version
```

[Note: For Docker Compose, use "docker compose" (spaces) in version 2 (recommended), while "docker-compose" (hyphens) was used in the deprecated version 1.]

A similar output will be shown:

```
Docker Compose version v2.3.3
```

Docker Compose is now successfully installed on the system. Refer to Annex 8.2 for the detailed overview of the Docker Folder.

In the next section, you'll see how to set up a `docker-compose.yml` file and get a containerized environment up and running with this tool.

3.4   Troubleshooting Permission Issues

Issue 1: Got permission denied while trying to connect to the Docker daemon socket at unix:///var/run/docker.sock: permission denied

Solution: Run the following command to give permission to docker.sock file:

```
sudo chmod 666 /var/run/docker.sock
```

Issue 2: WARNING: Error loading config file: /home/user/.docker/config.json - stat /home/user/.docker/config.json: permission denied

Solution: To fix this problem, either remove the `~/.docker/` directory (it is recreated automatically, but any custom settings are lost), or change its ownership and permissions using the following commands:

```
sudo     chown     "$USER":"$USER"     /home/"$USER"/.docker     -R
sudo chmod g+rwx "$HOME/.docker" -R
```

3.5   Docker Configuration – Template Repository

The docker configuration files are maintained in the GitHub organization's repository (docker-config.git). The following steps should be followed to create a copy of the repository and set up the required configurations.

Navigate to the folder where the project files will be stored:

```
# cd /<<folder-location>> /imis/

# cd /home/<<username>>/imis/
```

The provided repository maintains the code in a template format, thus click on the "*Use this template*" button to create a new repository template for the project. After the repository is created, pull the repository into the folder where the project files are stored. **[Note: Please make sure not to use the template file as your base file and make sure to create a new repository by clicking on the Use this template button.]**

Git clone or copy the docker compose image folder, use the following command:

```
# git clone https://github.com/<docker_repo_name>.git
```

Navigate to the folder where the project files will be stored:

```
# cd /<<folder-location>>/imis/imis-base-docker

# cd /home/<<username>>/imis/imis-base-docker
```

The folder structure for the repository is provided in Annex 7.2.

A few changes are required to the default configuration files, which are mentioned below.

Docker Configuration File Changes

The template provided in the repository is a ready-to-deploy configuration setup, however the different paths to different configurations are dependent on the server setup.

a. Filename: docker-compose.yml

Under the volumes heading, update the name of the project folder. By default, the folder name is set as base-imis. This is required to be done on all the services, as this variable links the project folder with the docker container folder.

If SSL is present, ensure the ssl folder with the corresponding files are present inside the docker configuration folder and the corresponding path is correct.

Additionally, if the default ports are not being used, the corresponding updates must also be carried out.

b. Filename: dockerfiles/nginx/default.conf

12

Under the server block, update the server_name variable as per the requirements.

4. APPLICATION SETUP

4.1 Application Source Code

The application source code is also maintained in the same GitHub organization under the Web-app Source Code repository. This source code must be cloned inside the "src" folder present in the Docker configuration template repository. This folder will contain the project files.

```
#cd imis-base-docker/src
```

Git pull the new repository into the "src" folder.

```
# git clone https://github.com/<application_repo_name>.git
```

Create the .env file by creating a copy of the existing example env file:

```
cp env.example .env
```

Ensure the necessary changes are carried out to the variables set in the environment file. By default, the env file will not have the values set, but will have placeholders in the format [variable_name].

Build the project and keep it running using the following command:

```
# docker compose up -d --build (Docker Compose version 2)

# docker-compose up -d --build (Docker Compose version 1)
```

For Docker Compose, use "docker compose" (spaces) in version 2 (recommended), while "docker-compose" (hyphens) is used in the deprecated version 1.

[Note: Be sure to type the above command instead of copying it, as copying might not register hyphens properly. There is a double hyphen before "build."]

4.2 Dependencies Installation

Install Composer dependencies:

```
# docker compose run --rm composer install
```

If a problem arises in the lock file, grant permission.

```
# chmod -R 777 composer.lock
```

If dependency issues arise, run the command below to help get dependencies installed without modification; however, it should be used with caution.

```
# docker compose run --rm composer install --ignore-platform-reqs
```

After installing the dependencies, you need to set the proper permissions on the files.

```
# chmod -R 777 stub
# chmod -R 777 storage
# chmod -R 777 app/Console/Commands
```

Install the JavaScript dependencies specified in the package.json file by executing

```
# docker compose run --rm npm install
```

This command runs the npm run watch command inside the Docker container:

```
# docker compose run --rm npm run watch
```

Cache configuration:

```
# docker compose run --rm php artisan config:cache
# docker compose run --rm php artisan route:cache
# docker compose run --rm php artisan view:cache

# Or run optimize:clear

#docker compose run --rm artisan optimize:clear
```

4.3 File Permissions

Two folders need to be created manually to store building survey KML files sent via mobile application and the emptying receipts and house images as well. They can be done by running the following commands:

```
# cd into project file
# cd storage/app/public
# mkdir building-survey-kml
# mkdir emptyings
# cd emptyings
# mkdir houses
# mkdir receipts
```

Then, provide the following permissions to enable write access:

```
# cd storage/app/public
# chmod -R 777  building-survey-kml
# chmod -R 777 emptyings
```

Go to Annex 8.4 for git commands that could be useful for the maintenance process and Annex 8.5 for useful laravel commands.

5. DATABASE SETUP

5.1 Postgres Installation

PostgreSQL VERSION = 14

Enable port 5432 of the server:

```
sudo ufw allow 8080
```

To install PostgreSQL, first refresh the server's local package index:

```
sudo apt update
```

Check which version of postgres is available in default repositories.

```
sudo apt-cache search postgresql | grep postgresql
```

Then, install the Postgres package along with a -contrib package that adds some additional utilities and functionality:

```
sudo apt install postgresql-${VERSION} postgresql-contrib
```

Press Y when prompted to confirm installation. Press Enter to accept any defaults to continue if prompted to restart any services. On successful installation, the PostgreSQL service starts automatically and can be verified as below.

```
systemctl status postgresql
```

5.2 User and Permission Management

After installing and ensuring PostgreSQL is running, create a user and a dB, and grant permissions of the dB to the user.

Connect to the PostgreSQL instance as the postgres user:

```
sudo -u postgres psql
```

Create a database in the postgresql terminal:

```
create database [database_name];
```

Create a user with encrypted password for postgresql:

```
create user [user_name]with password 'mypass';
```

Grant the created user permissions on the database:

```
grant all privileges on database [database_name] to [user_name];
```

[Note: The username and db name are taken in small letters. testDB changes into mydatabase and myUser turns into myuser]

To exit from PostgreSQL instance as the postgres user type:

```
\q
```

5.3   Configure PostgreSQL

Configure PostgreSQL in postgresql.conf file which is generally location in: /etc/postgresql/${VERSION}/main/postgresql.conf

PostgreSQL VERSION in use = 14

To open the file use nano command: (If root access is not available use sudo command)

```
cd /etc/postgresql/${VERSION}/main/

nano postgresql.conf

OR

nano /etc/postgresql/${VERSION}/main/postgresql.conf
```

Uncomment the following line in the postgresql.conf file:

```
listen_addresses = 'localhost'
```

and change it to:

```
listen_addresses = '*'
```

Open the pg_hba.conf file and add the following line to the end of the file: (Generally file location is: /etc/postgresql/${VERSION}/main/pg_hba.conf)

To open the file use nano command: (If root access is not available use sudo command)

```
cd /etc/postgresql/${VERSION}/main

nano pg_hba.conf

OR

nano /etc/postgresql/${VERSION}/main/pg_hba.conf
```

Add the following line to the end of the file:

```
host    all     all     server_ip/32   md5
```

Replace server_ip with the IPv4 address.

Once these steps are completed, the user will have full access to the database.

5.4   PostGIS Extension Installation

The PostGIS Extension is required for spatial data processing and analysis.

Install the extension (postgis-V3):

```
sudo apt install postgis postgresql-14-postgis-3
```

Then create an extension in db either through the pgAdmin interface or run the following command:

```
CREATE EXTENSION postgis;
```

Check if the extension has been installed successfully:

```
SELECT PostGIS_version();
```

Restart the PostgreSQL service:

```
systemctl restart postgresql
```

5.5 Blank Database Import

A blank database of the IMIS is maintained in the GitHub organization under the repository deployment_documentation. This blank database is imported into PostgreSQL through pgAdmin. The database server (PostgrSQL) must be connected through pgAdmin initially:

Connect to database server through pgadmin

Step 1 : Download and install pgAdmin.
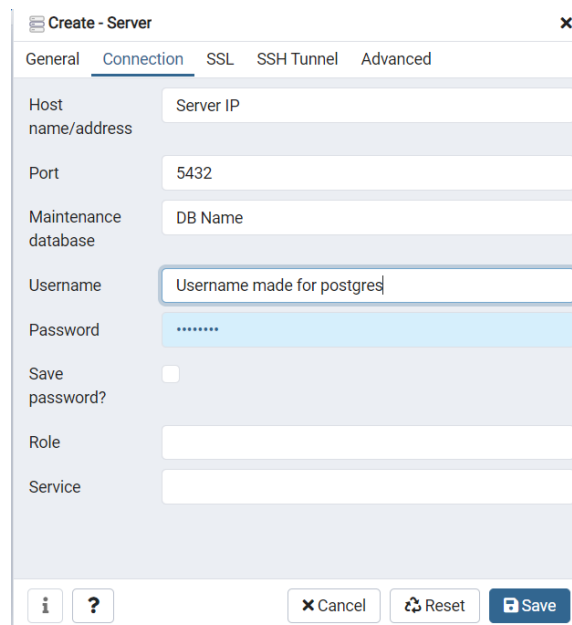
Step 2 : Start pgAdmin.

Step 3 : In the pgAdmin window, click the Servers tab.

Step 4 : Right-click on the Servers node and select Create > Server.

Step 5 : In the Create Server dialog, enter the following information:

- o Name: The name of the server.
- o Host name: The hostname or IP address of the PostgreSQL server.
- o Port: The port number of the PostgreSQL server.
- o Username: The username of the PostgreSQL user that you want to connect with.
- o Password: The password of the PostgreSQL user.

Step 6 : You may need to configure the SSH Tunnel settings if your PostgreSQL server is not directly accessible and requires an SSH tunnel to establish a connection securely. Follow the steps below:

- o Click the SSH Tunnel tab.

- o **Tunnel host:** Enter the hostname or IP address of the SSH server (the server you will use to create the SSH tunnel).
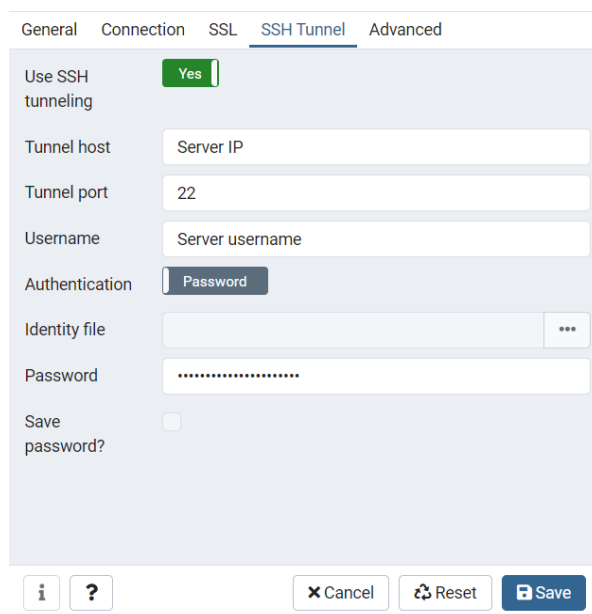
- o **Tunnel port:** Enter the port number on the SSH server where SSH is running. The default port for SSH is 22.

Step 7 : Click the Save button.

Step 8 : The server will be added to the pgAdmin window. Right-click on the server's name and select Connect.

Step 9 : You will be prompted to enter the password for the PostgreSQL user. Enter the password and click the OK button.

Step 10 : You should now be connected to the PostgreSQL server through pgAdmin.

Once pgAdmin is setup with the database server, the blank database is imported to the server.

Importing Blank Database

Step 1 : Download the blank database from the GitHub repository.

Step 2 : Open pgAdmin and connect to your PostgreSQL server.

Step 3 : Navigate to your server's "Databases" folder.

Step 4 : Right-click on the target database where you want to import the schema and data, and then select "Restore..." from the context menu.

Step 5 : Under the "Format" section, choose "Custom or Tar" from the dropdown menu. This format allows you to import both schema and data together. Browse the file by clicking on the "..." button in the "Filename" field.

Step 6 : Select the correct format of the file you are restoring, dump or backup.

Step 7 : Go to "Data Options" tab, under the "Do not save" section, check the box labeled **"Owner"**. (Optional)

Step 8 : Click "Restore".

Step 9 : If prompted, enter the password for the database superuser.

Step 10 : Update the .env file in the web-application server with the corresponding values of the database, such as database name, credentials, etc.

The restoration process will start, and pgAdmin will import the schema and data into the target database. The time it takes to complete depends on the size of the database dump. Explore the tables and other database objects to verify that the schema and data have been imported.

Note: When restoring a PostgreSQL database, permission issues could arise if the original owner specified in the backup does not exist on the target system. To overcome this issue, Exclude Ownership Restoration as mentioned in **Step 7 :**. This corresponds to the `--no-owner` flag, telling pgAdmin not to set ownership of the objects to the original user from the backup.

5.6   Database Seeder

The default look-up values for various dropdowns and default roles, permissions and user access can be setup through the seeder. Run the database seeder command:

```
docker-compose run –rm db:seed
```

6.   GEOSERVER SETUP

6.1   Installing JAVA (openjdk v "11.0.15")

Update the package list:

```
sudo apt update
```

Check if Java is already installed:

```
java -version
```

Install the default Java Runtime Environment (JRE), which will install the JRE from OpenJDK 11:

```
sudo apt install default-jre
```

Check if Java is installed:

```
java -version
```

[Note: Ubuntu offers AppArmor as an alternative to SELinux. While SELinux is available on Ubuntu, it is rather in an experimental stage and most likely will break your system if set to enforcing mode. In case you must use SELinux, make sure to disable AppArmor first. Also set SELinux first to permissive mode and check your logs for potential issues before you enable enforcing mode.]

6.2   Installing Tomcat

For security purposes, Tomcat should run under a separate, unprivileged user. Run the following command to create a user called tomcat:

```
sudo useradd -m -d /opt/tomcat -U -s /bin/false tomcat
```

By supplying /bin/false as the user's default shell, you ensure that it's not possible to login as tomcat.

Navigate to the /tmp directory:

```
cd /tmp
```

Download the archive using wget by running the following command:

```
VERSION=9.0.64

wget      https://downloads.apache.org/tomcat/tomcat-9/v${VERSION}/bin/apache-tomcat-${VERSION}.tar.gz
```

Check the Tomcat download page to see if a newer version is available.

Extract the archive you downloaded by running:

```
sudo   tar   xzvf   apache-tomcat-${VERSION}.tar.gz   -C   /opt/tomcat   --strip-components=1
```

Now grant tomcat ownership over the extracted installation by running:

23

```
sudo        chown        -R        tomcat:tomcat        /opt/tomcat/
sudo chmod -R u+x /opt/tomcat/bin
```

[Note: Why not using Tomcat 10+? Tomcat 10.x which is based off Servlet API version 5.0 which in turn is part of Jakarta EE version 9. The javax.* package has been renamed to jakarta.* package since Jakarta EE version 9. This thus means that the deployed web application is actually not compatible with Jakarta EE version 9. Geoserver is based on the older JEE version where the javax.* package is still used.]

6.3   Configuring admin users

Tomcat users are defined in /opt/tomcat/conf/tomcat-users.xml. Open the file for editing with the following command:

```
sudo nano /opt/tomcat/conf/tomcat-users.xml
```

Add the following lines before the ending tag and change username and password as necessary:

```
<role                          rolename="manager-gui"                          />
<user  username="manager"  password="manager_pass"  roles="manager-gui"  />

<role                          rolename="admin-gui"                          />
<user username="admin" password="admin_pass" roles="manager-gui,admin-gui"
/>
```

By default, Tomcat is configured to restrict access to the admin pages, unless the connection comes from the server itself. To access those pages with the users just defined, the config files need to be edited for those pages.

To remove the restriction for the **Manager** page, open its config file for editing:

```
sudo nano /opt/tomcat/webapps/manager/META-INF/context.xml
```

Comment out the Valve definition, as shown:

```
<Context          antiResourceLocking="false"          privileged="true"          >
  <CookieProcessor
className="org.apache.tomcat.util.http.Rfc6265CookieProcessor"
```

24

```
           sameSiteCookies="strict"                                    />
<!--         <Valve     className="org.apache.catalina.valves.RemoteAddrValve"
     allow="127\.\d+\.\d+\.\d+|::1|0:0:0:0:0:0:0:1"              />              -->
  <Manager
sessionAttributeValueClassNameFilter="java\.lang\.(?:Boolean|Integer|Long|Numb
er|String)|org\.apache\.catalina\.filters\.Csr>
</Context>
```

Save and close the file, then repeat for **Host Manager**:

```
sudo nano /opt/tomcat/webapps/host-manager/META-INF/context.xml
```

6.4   Creating systemd service

The systemd service that will be created will keep Tomcat quietly running in the background. The systemd service will also restart Tomcat automatically in case of an error or failure.

Tomcat, being a Java application itself, requires the Java runtime to be present, which is installed with the JDK in step 1. Before the service is created, the location of Java is required:

```
sudo update-java-alternatives -l
```

Output

```
java-1.11.0-openjdk-amd64      1111      /usr/lib/jvm/java-1.11.0-openjdk-amd64
```

Note the path where Java resides, listed in the last column, which will be required to define the service.

The tomcat service is placed in a file name tomcat.service, under /etc/systemd/system. Create the file for editing by running:

```
sudo nano /etc/systemd/system/tomcat.service
```

Add the following lines:

```
[Unit]
Description=Tomcat
```

```
After=network.target

[Service]
Type=forking

User=tomcat
Group=tomcat

Environment="JAVA_HOME=/usr/lib/jvm/java-1.11.0-openjdk-amd64"
Environment="JAVA_OPTS=-Djava.security.egd=file:///dev/urandom"
Environment="CATALINA_BASE=/opt/tomcat"
Environment="CATALINA_HOME=/opt/tomcat"
Environment="CATALINA_PID=/opt/tomcat/temp/tomcat.pid"
Environment="CATALINA_OPTS=-Xms512M     -Xmx1024M     -server     -
XX:+UseParallelGC"

ExecStart=/opt/tomcat/bin/startup.sh
ExecStop=/opt/tomcat/bin/shutdown.sh

RestartSec=10
Restart=always

[Install]
WantedBy=multi-user.target
```

Here, the defined service will run Tomcat by executing the startup and shutdown scripts it provides. A few environment variables are also set to define its home directory (which is /opt/tomcat as before) and limit the amount of memory that the Java VM can allocate (in CATALINA_OPTS). Upon failure, the Tomcat service will restart automatically.

Reload the systemd daemon so that it becomes aware of the new service:

```
sudo systemctl daemon-reload
```

To enable Tomcat starting up with the system, run the following command:

```
sudo systemctl enable tomcat
```

Start the Tomcat service by typing:

```
sudo systemctl start tomcat
```

26

Look at its status to confirm that it started successfully:

```
sudo systemctl status tomcat
```

6.5   Accessing the Web Interface

Tomcat uses port 8080 to accept HTTP requests. Run the following command to allow traffic to that port:

```
sudo ufw allow 8080
```

Access Tomcat by navigating to the IP address of the server:

```
http://your_server_ip:8080
```

6.6   Installing Geoserver (2.21.0)

Download Geoserver package:

```
VERSION = 2.21.0

wget
https://sourceforge.net/projects/geoserver/files/GeoServer/${VERSION}/geoserver-${VERSION}-war.zip/download
```

A file named "download" will be downloaded.

Rename the download file to geoserver:

```
mv download geoserver
```

Install 'unzip'

```
sudo apt install unzip
```

Unzip the downloaded file:

```
unzip geoserver -d geoserver-${VERSION}
```

Go into the folder:

27

```
cd geoserver-${VERSION}
```

There should be a geoserver.war file inside this directory. Move the geoserver.war file into /opt/tomcat/webapps folder. After moving this file, there should be a geoserver folder automatically created in the /opt/tomcat/webapps directory.

```
mv geoserver.war /opt/tomcat/webapps/
```

Change the owner of the folder to tomcat:

```
chown tomcat:tomcat /opt/tomcat/ -R
```

6.7   Accessing Geoserver Interface

Access Tomcat by navigating to the IP address of the server:

```
http://your_server_ip:8080/geoserver/web
```

[Note: CORS error (Access to image at '##' from origin '##' has been blocked by CORS policy: No 'Access-Control-Allow-Origin' header is present on the requested resource.) can arise. If such error arises, uncomment the following lines of code in the file web.xml (located at: /tomcat/webapps/geoserver/WEB-INF/):

```
<filter>

<filter-name>cross-origin</filter-name>

<filter-class>org.apache.catalina.filters.CorsFilter</filter-class>

<init-param>

<param-name>cors.allowed.origins </param-name>

<param-value>*</param-value>

</init-param>

<init-param>

<param-name>cors.allowed.methods</param-name>

<param-value>GET,POST,PUT,DELETE,HEAD,OPTIONS</param-value>

</init-param>
```

```
<init-param>

<param-name>cors.allowed.headers</param-name>

<param-value>*</param-value> </init-param>

 </filter>


<filter-mapping>

<filter-name>Set Character Encoding</filter-name>

<url-pattern>/*</url-pattern>

 </filter-mapping>
```

6.8   Installing Geoserver Plugins

There are few required geoserver plugins for IMIS that are mentioned below. The current Geoserver VERSION in use: 2.21.0

6.8.1 Authkey plugin

Download the plugin:

```
wget
https://sourceforge.net/projects/geoserver/files/GeoServer/2.21.0/extensions/geoserver-2.21.0-authkey-plugin.zip/download
```

Rename downloaded file "download" to "geoserver-cssr":

```
mv download geoserver-authkey
```

Unzip the file:

```
unzip geoserver-authkey -d  geoserver-2.21.0-authkey
```

CSS plugin

```
wget
https://sourceforge.net/projects/geoserver/files/GeoServer/2.21.0/extensions/geoser
ver-2.21.0-css-plugin.zip/download
```

Rename downloaded file "download" to "geoserver-css":

```
mv download geoserver-css
```

Unzip the file:

```
unzip geoserver-css -d  geoserver-2.21.0-css
```

6.8.2 Printing plugin

Download the plugin:

```
wget
https://sourceforge.net/projects/geoserver/files/GeoServer/2.21.0/extensions/geoser
ver-2.21.0-printing-plugin.zip/download
```

Rename downloaded file "download" to "geoserver-querylayer":

```
mv download geoserver-printing
```

Unzip the file:

```
unzip geoserver-printing -d  geoserver-2.21.0-printing
```

6.8.3 Querylayer plugin

Download the plugin:

```
wget
https://sourceforge.net/projects/geoserver/files/GeoServer/2.21.0/extensions/geoser
ver-2.21.0-querylayer-plugin.zip/download
```

Rename downloaded file "download" to "geoserver-querylayer":

```
mv download geoserver-querylayer
```

Unzip the file:

```
unzip geoserver-querylayer -d  geoserver-2.21.0-querylayer
```

6.8.4 WPS plugin

Download the plugin:

```
wget
https://sourceforge.net/projects/geoserver/files/GeoServer/2.21.0/extensions/geoser
ver-2.21.0-wps-plugin.zip/download
```

Rename downloaded file "download" to "geoserver-wps":

```
mv download geoserver-wps
```

Unzip the file:

```
unzip geoserver-wps -d geoserver-2.21.0-wps
```

6.8.5 YSLD plugin

Download the plugin:

```
wget
https://sourceforge.net/projects/geoserver/files/GeoServer/2.21.0/extensions/geoser
ver-2.21.0-ysld-plugin.zip/download --no-check-certificate
```

Rename downloaded file "download" to "geoserver-ysld" :

```
mv download geoserver-ysld
```

Unzip the file:

```
unzip geoserver-ysld -d geoserver-2.21.0-ysld
```

31

6.8.6 Installing the Extracted Plugins

Once all the plugins have been unzipped, copy all the extracted plugins into the following folder: (includes all the files and folders from auth, css and so on).

[Note: Extract the contents of the archive into the WEB-INF/lib directory in GeoServer. Make sure you do not create any sub-directories during the extraction process.]

```
cp        -r        geoserver-2.21.0-{authkey,css,printing,querylayer,wps,ysld}/*
/opt/tomcat/webapps/geoserver/WEB-INF/lib
```

Change the default password of the geoserver to a more secured password:

```
Default:
Username: admin
Password: geoserver
```

6.9   Workspace, Stores, Layers and Styles Setup

GeoServer includes several styles designed for IMIS. To set up these styles in IMIS, create a workspace and corresponding datastore connecting to the IMIS Database after installing GeoServer. The corresponding information related to the names and other attribute information related to the workspace, stores, layer and styles information including CSS, SLDs, and SQL queries are included and regularly maintained in the technical document section with the filename `**geoserver_document.md**` in the GitHub repository named web-app source code. To create a new workspace, store, layer, and styles in a geoserver, follow the steps as mentioned in section 7.6

SSL is recommended to ensure data security and prevent attacks. To secure your application with SSL, an SSL certification is required with the following certificates:

- private.pem (private key)

- fullchain.pem (full certificate)

SSL is required for both the application and the geoserver as well, thus two SSL certificates are required for IMIS. Additionally, to implement SSL, the corresponding domain/sub-domain names are also required for the web application and geoserver, that is mapped to the corresponding IP addresses of the servers. This procedure should be carried out at the end of the deployment process, after the deployment process is completed.

6.10   Configure SSL in Nginx (Docker)

Modify Nginx configuration in Dockerfile to enable HTTPS.

Generate ssl keys and keep them within the docker folder.

In docker-compose.yml:

Add ports 443 to nginx

```
container_name: nginx-imis
  ports:
    - 80:80
    - 443:443
```

Add volume of SSL folder to desired file path in nginx (in docker-compose.yml file):

```
volumes:
    - ./src/ imis-base:/var/www/html/imis:delegated
    - ./ssl:/etc/ssl/private:ro
```

In default.conf of nginx file, uncomment the following lines of code.

```
# SSL Certificate and Key
ssl_certificate /etc/ssl/private/fullchain.pem;
ssl_certificate_key /etc/ssl/private/privkey.pem;
```

IN default.conf of nginx, update following code for force redirection of port 80 to SSL port:

33

```
server {

    listen 80;

     index index.php index.html;

    server_name <<server IP>>;

    return 301 https://$request_uri; // domain name you want to point to

}
```

In nginx.dockerfile, uncomment the following line.

```
EXPOSE 443
```

Final default.conf for nginx:

```
server {

    listen 80;

    index index.php index.html;

    server_name <<server IP>>;

    return 301 https:// <<server domain>>;

}

server {

    listen 443 ssl;

    index index.php index.html;

    server_name <<domain_name>>;


    root /var/www/html/imis/public;


    # SSL Certificate and Key

     ssl_certificate /etc/ssl/private/fullchain.pem;

     ssl_certificate_key /etc/ssl/private/privkey.pem;


    location / {

        try_files $uri $uri/ /index.php?$query_string;
```

```
    }
    location ~ \.php$ {
        try_files $uri =404;
        fastcgi_split_path_info ^(.+\.php)(/.+)$;
        fastcgi_pass php:9000;
        fastcgi_index index.php;
        include fastcgi_params;
        fastcgi_param SCRIPT_FILENAME $document_root$fastcgi_script_name;
        fastcgi_param PATH_INFO $fastcgi_path_info;
    }
}
```

6.11 Configure SSL in GeoServer

Configure GeoServer to use SSL; otherwise, it may create issues while displaying layers and styles. As geoserver is installed conventionally without docker, update the nginx configuration file of the database server as follows:

```
server {
    listen 443 ssl;
    server_name <<domain_name>>;


    ssl_certificate <<path_to_file>>/fullchain.crt;
    ssl_certificate_key <<path_to_file>>/private.key;


    location / {
        proxy_pass http://localhost:8080;
        proxy_set_header Host $host;
        proxy_set_header X-Real-IP $remote_addr;
        proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
        proxy_set_header X-Forwarded-Proto $scheme;
```

```
    }


    access_log <<path_to_file>>/subdomain-ssl-access.log;

    error_log <<path_to_file>>/subdomain-ssl-error.log;

}


server {

    listen <port number>;

    server_name <<domain_name>>;


    return 301 https://$host$request_uri;

}
```

Restart Nginx and GeoServer

Rebuild the Docker:

```
(Docker Compose version 2)

docker compose down

docker compose up -d --build

(Docker Compose version 1)

docker-compose down

docker-compose up -d --build
```

Restart the Tomcat service:

```
sudo systemctl restart tomcat
```

7. ANNEX

7.1   Useful Linux Commands

lsb_release -a:

Displays information about the Linux distribution. Output:

```
Distributor ID:   Ubuntu
Description:       Ubuntu 22.04.2 LTS
Release:           22.04
Codename:          jammy
```

df -H:

Shows disk space usage in a human-readable format. Reports the used, available, percentage used, and mount point of every disk attached to the system.

free -th:

Checks the current available RAM using a human-friendly format. It displays the total amount of RAM and swap available on the system.

lscpu:

Provides detailed information about the CPU, such as CPU op-mode, vendor id, model name, CPU family, etc. It fetches the CPU architecture's information from sysfs and /proc/cpuinfo.

nproc:

Displays the total number of core processors installed in the Linux system and current processes. Useful for system analysis.

htop:

Displays detailed information about processes used by different users, including priority, nice value, virtual memory, shared memory, etc.

7.2 Docker Folder Overview

i. Dockerfiles: Folder containing docker files and configs for the docker images included in the docker-compose files. Contains the following at the moment:

 a. Php - config file and docker file

 b. Composer - docker file

 c. Nginx - config file and docker file

 The files with "root" in their name are used when the root user is used as the user to deploy the application. Recommended using a different user created solely for the deployment.

ii. Src: Folder where your Laravel project should be cloned to.

iii. Docker-compose.root.yml: Docker-compose file that is used when the root user is used to deploy the application.

iv. Docker-compose.yml: The docker-compose file where all the docker images and commands are written.

7.3 Useful Docker Commands

Command	Action
su laravel	Change user to laravel
docker ps -a	List all the running containers
docker-compose up -d --build	Build the project and keep it running
docker-compose down	Bring down the running containers
docker-compose run --rm composer install	Equivalent to composer install
docker-compose run --rm composer update	Equivalent to composer update

38

docker-compose run --rm artisan {artisan-command}	Equivalent to php artisan {artisan-command} E.g. docker-compose run –rm artisan migrate
docker-compose run --rm npm install	Equivalent to npm install
docker-compose run --rm npm update	Equivalent to npm update
docker-compose run --rm npm run production	Equivalent to npm run production
docker logs nginx-imis	Get nginx logs
docker exec -it php-imis sh	Enter php container shell
docker system df	displays information regarding the amount of disk space used by the docker daemon
docker system prune --volumes	remove or delete unused objects or data, it might be images, container, volume, or network as these objects are not removed unless we explicitly remove those objects, however, we need to specify the '–volumes' option to remove volumes in Docker 17.06.

7.4 Useful Git Commands

Useful commands:

Installation process: If Git is not pre-installed, install Git with the command:

```
sudo apt-get install git
```

Navigate to the desired installation folder and clone the git repository:

```
git clone https://github.com/<repo_name>.git
```

To clone a specific branch from a remote repository

```
git clone -b <branchname> <remote-repo-url>
```

To list all branches, including both local branches and remote branches

```
git branch -a
```

To list all remote repositories along with their corresponding URLs

```
git remote -v
```

7.4.1 Maintenance process:

Step 1 :   Pull code

To update your local repository with changes from the remote repository.

```
git pull
```

Step 2 :   Add code to git

The '*git add .*' command stages all changes in the current directory (and its subdirectories) so that they are ready to be committed.

```
git add .
```

Step 3 :   Commit code

It's essential to write clear and meaningful commit messages to help others understand the purpose of the changes.

```
git commit -m "commit_message"
```

Step 4 :   Push code

This step sends your committed changes from your local repository to the remote repository.

git push origin master

The *'origin'* refers to the remote repository's name (commonly set to the URL of the repository), and *'master'* is the name of the branch you want to push to.

7.5   Laravel-related operations

Composer Install and Update: Run composer install to add or update all the composer packages and required dependencies by running:

docker-compose run --rm composer install

docker-compose run --rm composer update

docker-compose run --rm composer dump-autoload

Generate App Key: Laravel requires an application key for encryption. Generate app key by running

docker-compose run --rm artisan generate:key

Run NPM Install: Install the JavaScript dependencies specified in the package.json file by executing

docker-compose run --rm npm install

Run NPM Watch: This command runs the npm run watch command inside the Docker container:

docker-compose run --rm npm run watch

Cache configuration:

docker-compose run --rm php artisan config:cache
docker-compose run --rm php artisan route:cache
docker-compose run --rm php artisan view:cache

Please note that the provided steps assume that you have set up Docker, have the necessary permissions, and have the repository properly configured.

7.6   Creating New Style & Layers in Geoserver

To create a new workspace, store, layer, and styles in a geoserver, follow these steps:

Step 1 : Create Workspace

      i.   In the GeoServer web interface, navigate to Data > Workspaces.

     ii.   Click the Add new workspace button.

    iii.   Enter a name and namespace URI for the workspace.

    iv.   Click the Create button.

Step 2 : **Create Store**

      i.   In the GeoServer web interface, navigate to Data > Stores.

     ii.   Click the Add new store button.

    iii.   Select the type of store you want to create.

    iv.   Enter the necessary configuration information for the store.

     v.   Click the Create button.

Step 3 : **Create Layer**

      i.   In the GeoServer web interface, navigate to Data > Layers.

     ii.   Click the Add new layer button.

    iii.   Select the workspace and store that the layer will be associated with.

    iv.   Enter the name and title for the layer.

     v.   Select the data format for the layer.

    vi.   Click the Create button.

Step 4 : **Creating Styles**

      i.   In the GeoServer web interface, navigate to Styles.

     ii.   Click the Add new style button.

    iii.   Enter a name and title for the style.

    iv.   Add CSS or select the SLD file that you want to use for the style.

     v.   Click the Create button.

For more assistance, you can follow the GeoServer tutorials:

Getting Started with GeoServer:
https://docs.geoserver.org/main/en/user/gettingstarted/index.html

42

Managing Layers:

https://docs.geoserver.org/main/en/user/data/webadmin/layers.html

Managing Styles:

https://docs.geoserver.org/main/en/user/styling/webadmin/index.html

7.7 Connect to db through pgadmin

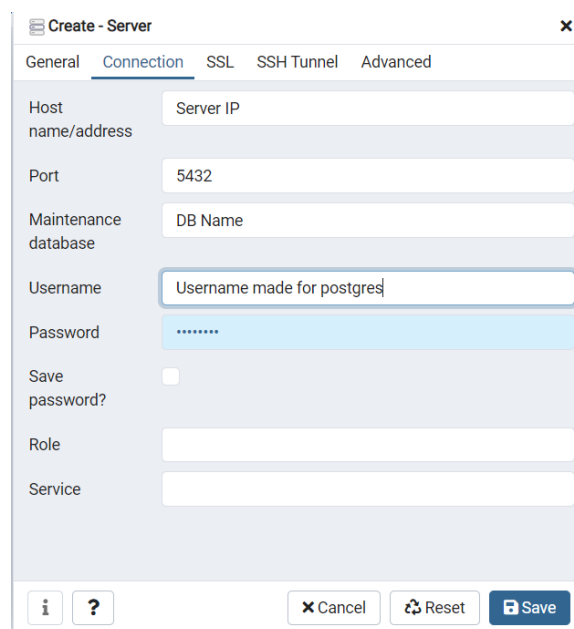**Step 11 :** Download and install pgAdmin.

Step 12 : Start pgAdmin.

Step 13 : In the pgAdmin window, click the Servers tab.

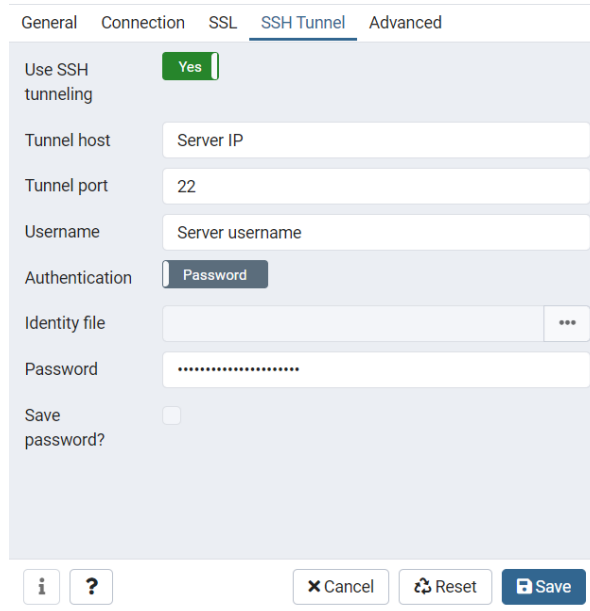**Step 14 :** Right-click on the Servers node and select Create > Server.

Step 15 : In the Create Server dialog, enter the following information:

- o Name: The name of the server.

- o Host name: The hostname or IP address of the PostgreSQL server.

- o Port: The port number of the PostgreSQL server.

- o Username: The username of the PostgreSQL user that you want to connect with.

- o Password: The password of the PostgreSQL user.

Step 16 : You may need to configure the SSH Tunnel settings if your PostgreSQL server is not directly accessible and requires an SSH tunnel to establish a connection securely. Follow the steps below:

43

o Click the SSH Tunnel tab.

o **Tunnel host:** Enter the hostname or IP address of the SSH server (the server you will use to create the SSH tunnel).

o **Tunnel port:** Enter the port number on the SSH server where SSH is running. The default port for SSH is 22.

Step 17 : Click the Save button.

Step 18 : The server will be added to the pgAdmin window. Right-click on the server's name and select Connect.

Step 19 : You will be prompted to enter the password for the PostgreSQL user. Enter the password and click the OK button.

Step 20 : You should now be connected to the PostgreSQL server through pgAdmin.

7.8 Logo and Copyright Text

i. To change logo in login page: public/img/logo.png

ii. To change text in login page: resources/views/app.blade.php >> div classes login-main-title, login-sub-title

iii. Copyright texts: resources/views/app.blade.php >> div class main footer

iv. Copyright texts in dashboard: resources/views/footer.blade.php

v. Copyright texts in maps: resources/views/index.blade.php

8. REFERENCES

HTTPS://WWW.DIGITALOCEAN.COM/COMMUNITY/TUTORIALS/INITIAL-SERVER-SETUP-WITH-UBUNTU-22-04

HTTPS://WWW.DIGITALOCEAN.COM/COMMUNITY/TUTORIALS/UFW-ESSENTIALS-COMMON-FIREWALL-RULES-AND-COMMANDS

HTTPS://WWW.DIGITALOCEAN.COM/COMMUNITY/TUTORIALS/HOW-TO-INSTALL-AND-USE-DOCKER-ON-UBUNTU-22-04

HTTPS://WWW.DIGITALOCEAN.COM/COMMUNITY/TUTORIALS/HOW-TO-INSTALL-AND-USE-DOCKER-COMPOSE-ON-UBUNTU-22-04

HTTPS://WWW.DIGITALOCEAN.COM/COMMUNITY/QUESTIONS/HOW-TO-FIX-DOCKER-GOT-PERMISSION-DENIED-WHILE-TRYING-TO-CONNECT-TO-THE-DOCKER-DAEMON-SOCKET

HTTPS://DOCS.GEOSERVER.ORG/STABLE/EN/USER/INSTALLATION/LINUX.HTML

HTTPS://WWW.DIGITALOCEAN.COM/COMMUNITY/TUTORIALS/HOW-TO-INSTALL-POSTGRESQL-ON-UBUNTU-22-04-QUICKSTART

HTTPS://TECHVIEWLEO.COM/HOW-TO-INSTALL-POSTGRESQL-DATABASE-ON-UBUNTU/