|  |
| --- |
|  |

**MUCP WEB SERVER Installation MANUAL**

**by**

**Kirodh Boodhraj**

**Handed over to DFFE on 2 October 2025**

|  |  |
| --- | --- |
| Executive Summary | |
| Title: | MUCP Web Server Installation Manual (Kirodh Boodhraj) |
| Abstract: | How to install the MUCP tool for the web. |
| Keywords: |  |
| Objective: | Manual written for DFFE and future users. |
| Purpose: |  |
| Methodology: |  |
| Conclusions: |  |
| Limitations or Risks: |  |
| Benefits: |  |
| Contract Context: | This document is the result of a research effort funded by NRM and CSIR PG internal funding. |

|  |  |
| --- | --- |
| Approval Page | |
| Project Name | MUCP |
| Project No |  |
| Title |  |
| Prepared for |  |

|  |  |  |  |
| --- | --- | --- | --- |
| Authored for | CSIR | Signature | Date |
| By | Kirodh Boodhraj | Kirodh | 29 Sept. 25 |
| Designation |  | | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Change History | | | | |
| Change Control No | Date | Name | Description |
| Version 1 | 29 Sept. 25 |  | Original release. |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

Contents

[Abbreviations 5](#_Toc210040928)

[1 Introduction 6](#_Toc210040929)

[2 Specifications for the VM 7](#_Toc210040930)

[2.1 Recommended VM Specifications 7](#_Toc210040931)

[3 Get the code 8](#_Toc210040932)

[3.1 Install Git (if not already installed) 8](#_Toc210040933)

[3.2 Clone the MUCP Tool Repository 8](#_Toc210040934)

[3.3 Navigate into the Project Folder 9](#_Toc210040935)

[4 Setting up the environment 10](#_Toc210040936)

[4.1 Installing Docker 10](#_Toc210040937)

[4.1.1 Initial VM Setup 10](#_Toc210040938)

[4.1.2 Install Docker 10](#_Toc210040939)

[4.1.3 (Optional) Manage Docker as a Non-Root User 11](#_Toc210040940)

[4.2 Generating SSL Certificates for MUCP with certbot and LetsEncrypt 11](#_Toc210040941)

[4.2.1 Generate Certificates (Certbot with Docker + NGINX) 11](#_Toc210040942)

[4.2.2 Copy Certificates into the MUCP Project 12](#_Toc210040943)

[4.3 Navigate into the Project Directory 12](#_Toc210040944)

[4.4 Configure the Static Files Directory 13](#_Toc210040945)

[4.5 Set Up the Environment File (.env file) 13](#_Toc210040946)

[4.6 Make Executables Ready 14](#_Toc210040947)

[4.7 Configure Trusted Origins 14](#_Toc210040948)

[4.8 Setting up the nginx.conf file 15](#_Toc210040949)

[4.8.1 Location of Nginx Configuration File 15](#_Toc210040950)

[4.8.2 Updating the Django Container Name 15](#_Toc210040951)

[4.8.3 Updating the DNS/Domain Name 16](#_Toc210040952)

[4.8.4 Certificate Directory 16](#_Toc210040953)

[5 Initial Program Setup 18](#_Toc210040954)

[5.1 Running the MUCP Tool with Docker 18](#_Toc210040955)

[5.1.1 Initially Running the Docker Containers 18](#_Toc210040956)

[5.1.2 Creating a Superuser 18](#_Toc210040957)

[5.2 Loading the Default MUCP Data 19](#_Toc210040958)

[6 Running and accessing the viewer on your web browser 20](#_Toc210040959)

[7 Support and debugging 21](#_Toc210040960)

Abbreviations

|  |  |
| --- | --- |
| CSIR | Council for Scientific and Industrial Research |
| DFFE | Department of Forestry, Fisheries and Environment |
| MUCP | Management Unit Control Plan |
|  |  |
|  |  |
|  |  |
|  |  |

# Introduction

Le Maitre et al. (2012) developed a generic species and area prioritization model (MUCP Tool) for use in prioritising invasive alien plant control operations in South Africa using spatial data and Forsyth et al. (2012) . It schedules treatments of invasions in the catchment taking into account the current state of the invasions, benefits of clearing, treatments that are required and the resources provided in its budget. The MUCP tool does not generate a detailed annual schedule of annual operations but the DEA-Natural Resources Management (NRM) programme has an Annual Plan of Operations tool which serves that purpose.

This manual will guide the user on installing the Web version of the MUCP tool on a Web Server using Linux operating system installed on a VM.

Outlined process consists of:

* Specifications for the VM
* Get the code
* Setting up the environment (installing Docker, certificates and running the containers)
* Initial Program Setup: Setting up the program with a superuser, performing migrations and collecting static files (if necessary)
* Running and accessing the viewer on your web browser

# Specifications for the VM

Linux VM Specification for MUCP

The MUCP tool is designed to run in a Linux environment with Docker. Setting up a dedicated Linux Virtual Machine (VM) ensures a stable, consistent environment for development and production.

## Recommended VM Specifications

To run the MUCP tool efficiently, we recommend the following minimum VM specifications:

Operating System: Ubuntu 22.04 LTS (or newer)

CPU: minimum 4 vCPUs (4+ recommended for production use)

RAM: 6 GB (8+ GB recommended if handling large shapefiles or datasets)

Disk Space: 100 GB minimum (SSD recommended)

Network: Stable internet connection for package downloads and updates

For larger datasets (e.g., GIS shapefiles, spatial queries), consider scaling up to 8 vCPUs, 16 GB RAM, and 100 GB + storage.

# Get the code

Downloading the MUCP Tool Code

Before setting up the Python environment, you need to obtain the source code for the MUCP tool from the official repository.

## Install Git (if not already installed)

Windows:

Download Git from <https://git-scm.com/download/win>

git --version

Linux (Debian/Ubuntu):

sudo apt update

sudo apt install git

After installation, confirm Git is available by running:

git --version

You should see an output like git version 2.43.0.

## Clone the MUCP Tool Repository

Navigate to the folder where you want to download the project. For example:

cd ~/projects

Then run the following command to download the MUCP code:

git clone <https://gitlab.com/kirodh/mucp-viewer-django-web.git>

This will create a new folder named mucp-viewer-django containing all the project files.

## Navigate into the Project Folder

Once the repository has been cloned, move into the project directory:

cd mucp-viewer-django

# Setting up the environment

## Installing Docker

### Initial VM Setup

Update and install common tools:

sudo apt update && sudo apt upgrade -y

sudo apt install -y build-essential curl wget unzip git software-properties-common

### Install Docker

⚠️ Note that the Docker installation could change, please google how to install Docker for your specific version of Linux. This is a guideline below:

Remove any old versions (if installed):

sudo apt remove docker docker-engine docker.io containerd runc

Install required packages:

sudo apt install -y ca-certificates curl gnupg lsb-release

Add Docker’s official GPG key:

sudo mkdir -p /etc/apt/keyrings

curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo gpg --dearmor -o /etc/apt/keyrings/docker.gpg

Set up Docker repository:

echo \

"deb [arch=$(dpkg --print-architecture) signed-by=/etc/apt/keyrings/docker.gpg] https://download.docker.com/linux/ubuntu \

$(lsb\_release -cs) stable" | sudo tee /etc/apt/sources.list.d/docker.list > /dev/null

Install Docker Engine and Docker Compose plugin:

sudo apt update

sudo apt install -y docker-ce docker-ce-cli containerd.io docker-compose-plugin

Verify Docker installation:

sudo docker --version

sudo docker compose version

### (Optional) Manage Docker as a Non-Root User

By default, Docker requires sudo. To run Docker without sudo:

sudo groupadd docker # may already exist

sudo usermod -aG docker $USER

newgrp docker

## Generating SSL Certificates for MUCP with certbot and LetsEncrypt

To secure the MUCP tool with HTTPS, we use Let’s Encrypt SSL certificates via Certbot inside Docker. This ensures encrypted communication between users and the application.

### Generate Certificates (Certbot with Docker + NGINX)

⚠️ Make sure your domain (e.g., yourdomain.com) is already pointing to the VM’s public IP via DNS.

We will use the guide at:

<https://www.programonaut.com/setup-ssl-with-docker-nginx-and-lets-encrypt/>

Download or create the Certbot setup (from the guide above).

This usually includes a certbot/ directory with Docker Compose configuration for NGINX and Certbot.

Verify certificates were created

After successful execution, Certbot will generate files inside a certbot/ folder, usually under:

certbot/conf/live/<yourdomain.com>/

The key files are:

fullchain.pem (the certificate)

privkey.pem (the private key)

### Copy Certificates into the MUCP Project

Once certificates are generated, they must be copied into the MUCP project so Docker/NGINX can use them.

Move into the folder containing your certbot/ directory

For example:

Copy the entire certbot folder into the MUCP project

Assuming mucp-viewer-django is one level up:

sudo cp -r certbot ../mucp-viewer-django

## Navigate into the Project Directory

Move into the MUCP project folder:

cd mucp-viewer-django

All further setup commands will be run inside this directory.

## Configure the Static Files Directory

Django collects CSS, JavaScript, and images into a single static directory. You need to create this folder manually.

mkdir src/static

This ensures all static files are stored in mucp-viewer-django/src/static/ when you run the collectstatic command (already programmed in the startup script)

## Set Up the Environment File (.env file)

Environment variables are used to store sensitive project settings (like database credentials, secret keys, and API tokens).

Copy the sample environment file into a working .env file:

cp sample.env .env

Open the .env file in your preferred editor (e.g., nano, vim) such as:

nano .env

This is what the sample.env file looks like:

# .env - example (DO NOT COMMIT real secrets)

DJANGO\_SETTINGS\_MODULE=main.settings

DEBUG=0

DJANGO\_MANAGE=/app/src/manage.py

# Gunicorn

WEB\_CONCURRENCY=4

GUNICORN\_TIMEOUT=900

GUNICORN\_THREADS=4

Don’t change the Django settings module variable.

The debug 0 means it is in production, while 1 means it is in debug mode.

The timeout is the web browser timeout.

The threads and web concurrency are the number of processes and workers the server will work with.

## Make Executables Ready

Some project scripts may need executable permissions. Run the following to ensure key files are executable:

chmod u+x src/manage.py

chmod u+x src/start.sh

This allows you to directly execute these scripts without permission errors.

## Configure Trusted Origins

For Django security, you must set trusted origins that match your domain.

Open the Django settings file:

nano src/main/settings.py

Locate the section for CSRF\_TRUSTED\_ORIGINS and update it with your DNS name, for example:

CSRF\_TRUSTED\_ORIGINS = [

"https://mucp.afis.co.za",

"http://mucp.afis.co.za",

]

Replace mucp.afis.co.za with your own domain if different.

Always include both https:// and http:// variants.

Save and exit the editor.

## Setting up the nginx.conf file

Once the Docker containers and SSL certificates are prepared, the next step is to configure Nginx to correctly serve the MUCP Django application. This section explains how to edit the Nginx configuration file with your container name and DNS information.

### Location of Nginx Configuration File

Inside the Docker setup, the Nginx configuration is typically mounted from:

mucp-viewer-django/nginx/nginx.conf

This file defines:

* The upstream connection to the Django Docker container
* The DNS/hostname used to access MUCP
* Static and media file serving rules
* SSL configuration using Let’s Encrypt certificates

### Updating the Django Container Name

In the upstream block of nginx.conf, update the container name so it matches the Django service defined in your docker-compose.yml.

upstream django\_upstream {

server django:8000; # replace "django" with the name of your Django service in docker-compose

keepalive 32;

}

Example: If your docker-compose.yml defines the service as mucp-django, then the line should be:

server mucp-django:8000;

### Updating the DNS/Domain Name

In both the HTTP redirect server and the HTTPS server, replace mucp.afis.co.za with the DNS name for which you generated certificates.

server {

listen 80;

server\_name mucp.afis.co.za; # replace with your DNS name

return 301 https://$host$request\_uri;

}

server {

listen 443 ssl http2;

server\_name mucp.afis.co.za; # replace with your DNS name

ssl\_certificate /etc/letsencrypt/live/mucp.afis.co.za/fullchain.pem;

ssl\_certificate\_key /etc/letsencrypt/live/mucp.afis.co.za/privkey.pem;

}

Example: If your DNS is mucp.example.org, then update as follows:

server\_name mucp.example.org;

ssl\_certificate /etc/letsencrypt/live/mucp.example.org/fullchain.pem;

ssl\_certificate\_key /etc/letsencrypt/live/mucp.example.org/privkey.pem;

### Certificate Directory

The SSL certificate and key paths are automatically generated by Certbot when you followed the steps in Section 6

/etc/letsencrypt/live/<your-domain>/fullchain.pem

/etc/letsencrypt/live/<your-domain>/privkey.pem

# Initial Program Setup

## Running the MUCP Tool with Docker

Once your environment is prepared and your configuration files are set up, the next step is to run the MUCP Django project using Docker. This section covers the initial container startup, superuser creation, and loading of the default MUCP data.

### Initially Running the Docker Containers

Navigate to the project folder if you are not already there:

cd mucp-viewer-django

Start the Django container in the background:

docker compose up -d Django

Start the NGINX container in the background:

docker compose up -d nginx

At this stage, the core containers for Django and NGINX are running. Use docker compose ps and logs commands to see if there are any errors.

The start.sh script will automatically do any migrations and collecting of static files for you before starting the Django production server. If this fails for some reason, please enter the Docker container and run these commands manually to determine the error.

### Creating a Superuser

In order to log into the Django administration interface, you need to create a superuser.

Access the Django container shell (django being the container name):

docker compose exec django bash

Once inside the container, run the following command to create a superuser:

python src/manage.py createsuperuser

Enter the following details when prompted:

* Username
* Email
* Password

(You can change these later in the Django admin interface if needed.)

Exit the Docker container:

exit

## Loading the Default MUCP Data

The MUCP tool comes with a set of default data that must be preloaded into the database. This step may take up to 10 minutes, depending on system performance.

Access the Django container shell again:

docker compose exec django bash

Navigate to the src directory:

cd src

Run the default data loading command:

python manage.py load\_default\_data

This command will insert the base species, project templates, and other required records into the database.

Once the process completes, exit the container:

exit

At this stage, the Django application is running in Docker, the superuser is created, and the system is preloaded with the required default data. You are now ready to log into the MUCP web interface and begin using the tool.

# Running and accessing the viewer on your web browser

If the above instruction were successful, the viewer will be on your DNS for example, <https://mucp.afis.co.za>

# Support and debugging

There are many resources out there for putting a Django program into production. Some include:

* ChatGPT and other advanced AI chatbots
* StackOverflow
* Youtube

Manual End