Biosphere Capital

Right Plant Systems Management Guide

Prepared by

Catalyst.Net Ltd

December 2021

Commercial in confidence

catalyst
open source technologists

Level 6, Catalyst House, 150 Willis Street, Wellington 6011
PO Box 11053, Manners Street, Wellington 6142, New Zealand
+64 4 499 2267 // enquiries@catalyst.net.nz // www.catalyst.net.nz

Contents

Architecture **2**

Running Configuration **2**
- OS and Supporting Software . . . 2
  - SSH . . . 2
  - Code . . . 2
  - Observing running configuration . . . 2

Developer README **3**
  - Add shapefiles for database population . . . 3
  - Add spreadsheet data for database population . . . 3
- Running application for development . . . 3
  - Initial build . . . 3
  - Initialise database . . . 3
  - Run web application . . . 3
- Available commands . . . 4
- Creating zones for habitat images . . . 5
- Setting up and running the application for production . . . 6
  - Setting up certificates . . . 6

Architecture

Right Plant is built on a stack of a React frontend build in JavaScript, a simple Django REST API as a backend and PostGIS for data storage and persistence.

All the components are configured to be deployed using `docker-compose`.

Running Configuration

OS and Supporting Software

The Right Plant VM lives in the Hamilton (nz-hlz-1) region of Catalyst Cloud, with an IP address of 103.197.61.141.

It runs a minimal version of 20.04, which will be supported until April 2025. Patching the server is as simple as `sudo apt update; sudo apt dist-upgrade -y` and a system reboot if required. It has not been configured for automatic upgrades.

SSH

SSH is available on port 22 for Catalyst management purposes, and 43212 for general user access. SSH Keys are required to log into the server, password authentication should not be enabled.

Code

The code and all configuration can be found on the VM in `/opt/rightplant`.

Code is deployed as a git repository - any changes made will be tracked by git. It is not possible to push changes to the Catalyst stored git repository.

Observing running configuration

Because Right Plant is deployed using Docker, the root user can access the docker daemon and execute CLI commands, such as `docker ps` to view running containers and `docker logs <container-name>` to view logs from the containers.

In general, running `docker <command> --help` will get you useful information about how to interact with the docker CLI.

Developer README

This section has been lifted from the `README.md` file, that can be found in the git repository. It is aimed at developers and how to develop and run Right Plant.

Add shapefiles for database population

Please unzip and add the following shapefiles to the `./backend/right_tree/api/data/resources` directory. It should include all the files required by the shapefile and use naming conventions as follows:

Ecological Districts Shapefile:

```
backend/right_tree/api/data/resources/ecological_districts/
- DOC_EcologicalDistricts_2021_08_02.cpg
- DOC_EcologicalDistricts_2021_08_02.dbf
- DOC_EcologicalDistricts_2021_08_02.prj
- DOC_EcologicalDistricts_2021_08_02.sbn
- DOC_EcologicalDistricts_2021_08_02.sbx
- DOC_EcologicalDistricts_2021_08_02.shp
- DOC_EcologicalDistricts_2021_08_02.shp.xml
- DOC_EcologicalDistricts_2021_08_02.shx
```

Ecological Districts Shapefile:

```
backend/right_tree/api/data/resources/fundamental_soil_layers/
- fundamental-soil-layers-new-zealand-soil-classification.cpg
- fundamental-soil-layers-new-zealand-soil-classification.dbf
- fundamental-soil-layers-new-zealand-soil-classification.prj
- fundamental-soil-layers-new-zealand-soil-classification.shp
- fundamental-soil-layers-new-zealand-soil-classification.shx
- fundamental-soil-layers-new-zealand-soil-classification.xml
```

Christchurch Zone Shapefile:

```
backend/right_tree/api/data/resources/chch_zone/
- Greater_Christchurch_Area.cpg
- Greater_Christchurch_Area.shp
- Greater_Christchurch_Area.dbf
- Greater_Christchurch_Area.shx
- Greater_Christchurch_Area.prj
```

Add spreadsheet data for database population

The plant spreadsheet should be renamed as `plant_data.xlsx` and placed in the `./backend/right_tree/api/data/resources` directory.

Running application for development

Initial build

Builds the Django backend docker image. This may need to be re-run if any new dependencies are added.

```
./dev build
```

Initialise database

Creates `right_tree` database and installs `postgis` extensions.

```
./dev init_database
```

Run web application

Starts up the applications including the frontend, backend and database.

```
./dev start
```

Once running the components can be accessed as follows:

Application	Hosted
React Frontend	http://localhost:3000
Django Backend	http://localhost:8000
Database	postgis://localhost:5432

Available commands

Other commands can be run using the following.

```
./dev <command>
```

A summary of available commands are outlined below. Note that if the command requires the application to be running (`Requires Run`) please execute `./dev start` in another terminal before running that command.

Command	Description	Requires Run
`create_database`	Removes the existing database and data. Then it creates the `right_tree` database within a fresh postgis database instance.	No
`makemigrations`	Performs the django `makemigrations` command in the backend container.	Yes
`migrate`	Performs the django `migrate` command in the backend container.	Yes
`createsuperuser`	Performs the django `createsuperuser` command in the backend container.	Yes
`load_fixtures`	Performs the django `loaddata` command in the backend container. This loads all the fixtures found in the `/backend/right_tree/api/data/fixtures` directory.	Yes
`load_shapefiles`	Performs the custom `loadshapefiles` command in the backend container. This loads the ecological districts and soil layers shape files in c.	Yes
`create_plant_fixtures`	Performs the custom `createplantfixtures` command in the backend container. This loads the plant spreadsheet data from `/backend/right_tree/api/data/resources/plant_data.xlsx`. Requires the fixtures to be applied and shapefiles loaded.	Yes
`reset_plants`	Performs the custom `resetplants` command in the backend container. This removes all plant entries from the database.	Yes

Command	Description	Requires Run
`load_plant_fixtures`	Loads the `/backend/right_tree/api/data/fixtures/plants.json` fixture. Requires the `plants.json` file to be created (`./dev create_plant_fixtures`) and the plant table to be empty (`./dev reset_plants`).	Yes
`load_plants`	Creates plants fixtures and loads them into a fresh plant table in the database. Requires the fixtures to be applied and shapefiles loaded.	Yes
`load_sites_from_spreadsheet`	Loads site spreadsheet data the database initially (replaced with fixtures containing further information)	Yes
`populate_database`	Populates the `right_tree` database with base data (fixtures), provided shapefiles and plant spreadsheet data. Requires the database to be created.	No
`init_database`	Creates and populates the database	No
`reset_database`	Removes, recreates and populates the database	No
`build`	Builds required images	No
`start`	Runs all services including the frontend, backend and postgres database	No
`build`	Builds required images (frontend and backend) for development	No
`build_production`	Builds required images (frontend and backend) for production	No
`start_production`	Runs all services in production mode including the frontend, backend and postgres database	No
`renew_certificate`	Renews certificates for production	No
`process_svg_files`	Removes semi-colons from raw svg files to be compatible with the application	No

Creating zones for habitat images

1. Create png image from original svg with approprate crop.
2. Create zone polygons/rectangles on the original svg with divider lines anchor points as a guide
3. Copy zone polygons/rectangles to png image and size to fit (this is to ensure the only paths on the image the selectable ones)
4. Ensure all overlays have an almost transparent fill (lowest transparency value - in Inkscape this is 1) and no outline
5. Add a ‘label’ (not an id) to each overlay to match with a column name relating to the zone segment, this may be repeated. In Inkscape this is under ‘Object Properties’.
6. Save the png with overlays as an svg (it may either be inkscape or plain svg)
7. Place svg in relevant directory (`./frontend/src/assets/img/habitatSVG/`) in the frontend
8. Find and replace any instance of colons (:) in property names for the raw svg i.e. inkscape:label -> inkscapelabel. A helper script has been written to do this automatically please run `python process_svg.py`.

Setting up and running the application for production

1. Ensure the prerequisites are met as defined in [#Initial Setup]
2. Create an `.env` file (if not done prior) in the root directory using `default.env` as an example. Uncomment values relating to production and fill in the values as appropriate.
3. Build backend image sudo `./dev build_production`
4. Create the database sudo `./dev create_database`
5. Manually create postgres user with password and add the user to the `righttree` database with all permissions.

Create an interactive terminal into the postgres container

```
sudo docker-compose -f docker-compose.production.yaml up -d postgres
sudo docker exec -it postgres bash
```

Within the interactive terminal. Connect to the database, add the righttree_admin user and give permissions. Please use the same credentials as defined in .env.

```
psql -U postgres
\c righttree
CREATE USER righttree_admin;
ALTER USER righttree_admin with encrypted password 'YOUR PASSWORD';
GRANT ALL PRIVILEGES ON DATABASE righttree TO  righttree_admin;
```

Exit the container and stop postgres service:

```
[CTRL-D] - to exit psql THEN [CTRL-D] to exit container
sudo docker-compose -f docker-compose.production.yaml down
```

6. Populate the database using sudo `./dev populate_database`
7. Build optimised frontend build and collect together staticfiles sudo `./dev create_staticfiles`
8. Create a django superuser for access to the admin interface. Please use the same credentials as defined in .env sudo `./dev createsuperuser`
9. Run the production application using sudo `./dev start_production`

Setting up certificates

Create certificate using certbot and letsencrypt, choose option 1 and provide an appropriate email. Ensure port 80 and 443 are externally exposed for the domain before running this command. To retrieve a staging certificate, use the `--test-cert` flag.

```
sudo docker run -i --rm --name certbot -p 443:443 -p 80:80 \
  -v /etc/letsencrypt:/etc/letsencrypt/ certbot/certbot certonly \
  -d [YOUR DOMAIN] --logs-dir /etc/letsencrypt/logs
```