RocketMap Documentation

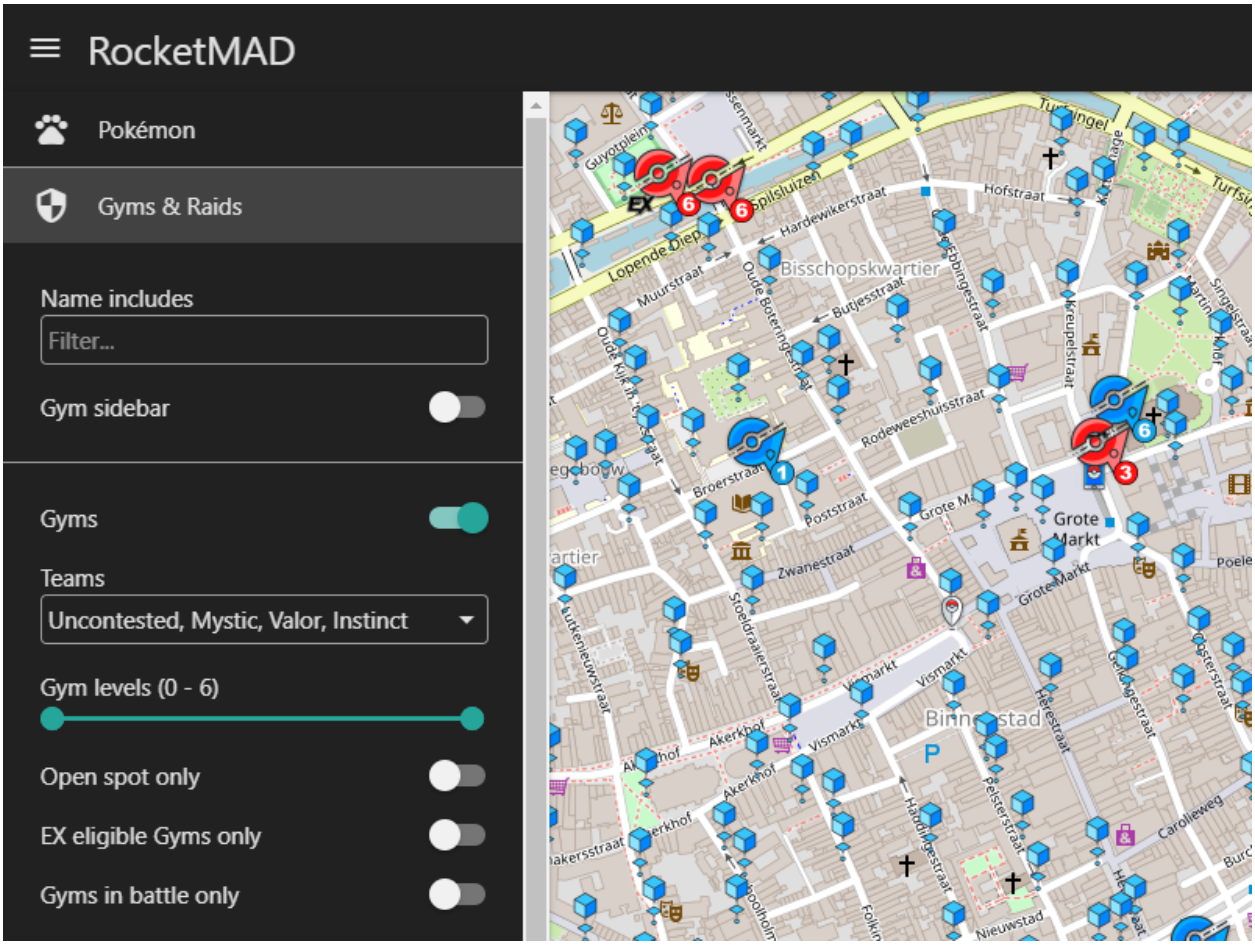
Release 4.0.0

RocketMap Team

Sep 25, 2023

About

RocketMAD gives you a map with nearby Pokémon, PokéStops, Gyms, and much more in the form of a web-app as well as a native phone application. It visualizes data scanned by MAD. RocketMAD does not work as standalone and the code has been heavily modified to only work with MAD. This frontend is based on RocketMap.



[Wanting to install and run RocketMAD for the first time? Start here!]

[Official GitHub] [Discord Support] [GitHub Issues]

CHAPTER 1

Contributing to this Wiki

Note: This article is about contributing pages and edits to this wiki. For contributing to RocketMAD itself see *App Development*.

You can fork this documentation from the main RocketMAD GitHub repository and open pull requests for changes.

1.1 Adding or Editing Pages

A few guidelines to help keep things clean and organized...

Keep filenames short and to the point, for example: `installation.rst`

Always begin your new page with a title:

```
Awesome Wiki Page
#################
```

Titles will be shown at the top of a page and in the site navigation. A title should describe a page in a glance. The rest of the file is written in ReST or Markdown structured text. Here is a cheatsheet for RST formatting, and one for markdown.

Once done editing your page, add it under one of the `toctree` sections in `index.rst`.

Now to preview your changes, open a terminal, go into the `docs` directory and use `make clean-auto auto`. This will start a local webserver with live updates pages as you save them.

Finally, when you are finished, submit your changes as a Pull Request to be reviewed.

CHAPTER 2

App Development

Note: This article is about contributing to the development codebase. For contributing to the wiki see *Contributing to this Wiki*.

Warning: These instructions will help you get started contributing code to the `master` branch. If you just want to **use the map** you should follow the *Basic Installation* instructions.

Development requires several tools to get the job done. Python, obviously, needs to be installed. We also utilize NodeJS and Grunt for front-end asset compilation. The *Basic Installation* instructions have the relevant information about getting node installed. Follow that.

2.1 Node and Grunt

Grunt is a tool to automatically run tasks against the code. We use grunt to transform the Javascript and CSS before it's run, and bundle it up for distribution.

If you want to change the Javascript or CSS, you must install and run Grunt to see your changes

2.1.1 Compiling Assets

After Grunt is installed successfully, you need to run it when you change Javascript or CSS.

Simply type

```
npm run watch
```

on the command-line. This runs a default grunt "task" that performs a number of subtasks, like transforming JS with Babel, minifying, linting, and placing files in the right place. It will also automatically start a "watch" which will automatically rebuild files as you modify them. You can stop grunt-watch with CTRL+C.

If you'd like to just build assets once, you can run:

```
npm run build
```

2.2 The "/dist" directory

Files in the "static/dist/" subdirectories should not be edited. These will be automatically overwritten by Grunt.

To make your changes you want to edit e.g. `static/js/map/map.js`

CHAPTER 3

Basic Installation

These instructions cover an installation from the master branch in git.

3.1 Prerequisites

The following prerequisites are required to run RocketMAD:

- MAD MySQL database
- Python 3.6+
- pip3
- node and npm

3.2 Downloading the Application

To run a copy from the latest `master` branch in `git` you can clone the repository:

```
git clone https://github.com/cecpk/RocketMAD.git
```

3.3 Installing Modules

At this point you should have the following:

- Python 3.6+
- pip3
- RocketMAD application folder

First, open up your shell and change to the directory of RocketMAD.

You can verify your installation like this:

```
python3 --version
pip3 --version
```

The output should look something like:

```
$ python3 --version
Python 3.6.0
$ pip3 --version
pip 9.0.1 from /usr/lib/python3/dist-packages (python 3.6)
```

Now you can install all the Python dependencies, make sure you're still in the directory of RocketMAD:

```
pip3 install -r requirements.txt
```

Note: It's highly recommended to use a virtual environment. A virtual environment is a Python environment such that the Python interpreter, libraries and scripts installed into it are isolated from those installed in other virtual environments.

3.3.1 Building Front-End Assets

In order to run from a git clone, you must compile the front-end assets with node. Make sure you have node installed for your platform.

Once node/npm is installed, open a shell and validate your install:

```
node --version
npm --version
```

The output should look something like:

```
$ node --version
v10.15.3
$ npm --version
6.0.1
```

Once node/npm is installed, you can install the node dependencies and build the front-end assets:

```
npm install
npm run build
```

3.4 Basic Launching

Once those have run, you should be able to start using the application, make sure you're in the directory of RocketMAD then:

```
python3 runserver.py --help
```

Read through the available options and set all the required CLI flags to start your own server. At a minimum you will need to provide a location.

The most basic config you could use would look something like this:

```
python3 runserver.py -l "a street address or lat/lng coords here"
```

Once your setup is running, open your browser to http://localhost:5000 and your pokemon will begin to show up! Happy hunting!

3.5 Things to Know

- All of these flags can be set inside of a configuration file to avoid clutter in the command line. Go here to see how.
- A full list of all commands are available here.
- A few tools to help you along the way are located here.

3.6 Updating the Application

RocketMAD is a very active project and updates often. You can follow the latest changes to see what's changing.

You can update with a few quick commands:

```
git pull
pip3 install -r requirements.txt --upgrade
npm run build
```

Watch the latest changes on Discord to know when updating will require commands other than above.

IMPORTANT Some updates will include database changes that run on first startup. You should run only **one** `runserver.py` command until you are certain that the DB has been updated. You will know almost immediately that your DB needs updating if **Detected database version x, updating to x** is printed in the console. This can take a while so please be patient. Once it's done, you can start all your instances like you normally would.

CHAPTER 4

Command Line (Outdated)

```
usage: runserver.py [-h] [-cf CONFIG] [-scf SHARED_CONFIG] [-a AUTH_SERVICE]
                    [-u USERNAME] [-p PASSWORD] [-w WORKERS]
                    [-asi ACCOUNT_SEARCH_INTERVAL]
                    [-ari ACCOUNT_REST_INTERVAL] [-ac ACCOUNTCSV]
                    [-hlvl HIGH_LVL_ACCOUNTS] [-bh] [-wph WORKERS_PER_HIVE]
                    [-l LOCATION] [-alt ALTITUDE] [-altv ALTITUDE_VARIANCE]
                    [-uac] [-j] [-al] [-st STEP_LIMIT] [-gf GEOFENCE_FILE]
                    [-gef GEOFENCE_EXCLUDED_FILE] [-sd SCAN_DELAY]
                    [--spawn-delay SPAWN_DELAY] [-enc] [-cs] [-ck CAPTCHA_KEY]
                    [-cds CAPTCHA_DSK] [-mcd MANUAL_CAPTCHA_DOMAIN]
                    [-mcr MANUAL_CAPTCHA_REFRESH]
                    [-mct MANUAL_CAPTCHA_TIMEOUT] [-ed ENCOUNTER_DELAY]
                    [-ignf IGNORELIST_FILE] [-encwf ENC_WHITELIST_FILE]

                    [-nostore] [-apir API_RETRIES]
                    [-wwhtf WEBHOOK_WHITELIST_FILE]
                    [-ld LOGIN_DELAY] [-lr LOGIN_RETRIES] [-mf MAX_FAILURES]
                    [-me MAX_EMPTY] [-bsr BAD_SCAN_RETRY]
                    [-msl MIN_SECONDS_LEFT] [-dc] [-H HOST] [-P PORT]
                    [-L LOCALE] [-c] [-m MOCK] [-ns] [-os] [-sc] [-nfl]
                    -k GMAPS_KEY [--skip-empty] [-C] [-cd] [-np] [-ng] [-nr]
                                 [-nk] [-ss [SPAWNPOINT_SCANNING]] [-speed] [-spin]
                    [-ams ACCOUNT_MAX_SPINS] [-kph KPH] [-hkph HLVL_KPH]
                    [-ldur LURE_DURATION] [-px PROXY] [-pxsc]
                    [-pxt PROXY_TEST_TIMEOUT] [-pxre PROXY_TEST_RETRIES]
                    [-pxbf PROXY_TEST_BACKOFF_FACTOR]
                    [-pxc PROXY_TEST_CONCURRENCY] [-pxd PROXY_DISPLAY]
                    [-pxf PROXY_FILE] [-pxr PROXY_REFRESH]
                    [-pxo PROXY_ROTATION] --db-name DB_NAME --db-user DB_USER
                    --db-pass DB_PASS [--db-host DB_HOST] [--db-port DB_PORT]
                    [--db-threads DB_THREADS] [-DC] [-DCw DB_CLEANUP_WORKER]
                    [-DCp DB_CLEANUP_POKEMON] [-DCg DB_CLEANUP_GYM]
                    [-DCs DB_CLEANUP_SPAWNPOINT] [-DCf DB_CLEANUP_FORTS]
                    [-wh WEBHOOKS] [-gi] [-DC]
```

(continues on next page)

(continued from previous page)

```
                [--wh-types {pokemon,gym,raid,egg,tth,gym-info,pokestop,lure,captcha}]
                [--wh-threads WH_THREADS] [-whc WH_CONCURRENCY]
                [-whr WH_RETRIES] [-whct WH_CONNECT_TIMEOUT]
                [-whrt WH_READ_TIMEOUT] [-whbf WH_BACKOFF_FACTOR]
                [-whlfu WH_LFU_SIZE] [-whfi WH_FRAME_INTERVAL]
                [--ssl-certificate SSL_CERTIFICATE]
                [--ssl-privatekey SSL_PRIVATEKEY] [-ps [logs]]
                [-slt STATS_LOG_TIMER] [-sn STATUS_NAME] [-hk HASH_KEY]
                [-novc] [-vci VERSION_CHECK_INTERVAL]
                [-odt ON_DEMAND_TIMEOUT] [--disable-blacklist] [--disable-blacklist]
                [-tp TRUSTED_PROXIES] [--api-version API_VERSION]
                [--no-file-logs] [--log-path LOG_PATH]
                [--log-filename LOG_FILENAME] [--dump] [-exg]

                [-v | --verbosity VERBOSE] [-Rh RARITY_HOURS]
                [-Rf RARITY_UPDATE_FREQUENCY]
```

Args that start with '–' (eg. -a) can also be set in a config file (config/config.ini or specified via -cf or -scf). The recognized syntax for setting (key, value) pairs is based on the INI and YAML formats (e.g. key=value or foo=TRUE). For full documentation of the differences from the standards please refer to the ConfigArgParse documentation. If an arg is specified in more than one place, then commandline values override environment variables which override config file values which override defaults.

```
optional arguments:
  -h, --help            show this help message and exit
                        [env var: POGOMAP_HELP]
  -cf CONFIG, --config CONFIG
                        Set configuration file
  -scf SHARED_CONFIG, --shared-config SHARED_CONFIG
                        Set a shared config
  -a AUTH_SERVICE, --auth-service AUTH_SERVICE
                        Auth Services, either one for all accounts or one per
                        account: ptc or google. Defaults all to ptc.
                        [env var: POGOMAP_AUTH_SERVICE]
  -u USERNAME, --username USERNAME
                        Usernames, one per account.
                        [env var: POGOMAP_USERNAME]
  -p PASSWORD, --password PASSWORD
                        Passwords, either single one for all accounts or one
                        per account. [env var: POGOMAP_PASSWORD]
  -w WORKERS, --workers WORKERS
                        Number of search worker threads to start. Defaults to
                        the number of accounts specified.
                        [env var: POGOMAP_WORKERS]
  -asi ACCOUNT_SEARCH_INTERVAL, --account-search-interval ACCOUNT_SEARCH_INTERVAL
                        Seconds for accounts to search before switching to a
                        new account. 0 to disable.
                        [env var: POGOMAP_ACCOUNT_SEARCH_INTERVAL]
  -ari ACCOUNT_REST_INTERVAL, --account-rest-interval ACCOUNT_REST_INTERVAL
                        Seconds for accounts to rest when they fail or are
                        switched out. [env var: POGOMAP_ACCOUNT_REST_INTERVAL]
  -ac ACCOUNTCSV, --accountcsv ACCOUNTCSV
                        Load accounts from CSV file containing
                        "auth_service,username,password" lines.
                        [env var: POGOMAP_ACCOUNTCSV]
  -hlvl HIGH_LVL_ACCOUNTS, --high-lvl-accounts HIGH_LVL_ACCOUNTS
```

(continues on next page)

(continued from previous page)

```
                        Load high level accounts from CSV file containing
                        "auth_service,username,password" lines.
                        [env var: POGOMAP_HIGH_LVL_ACCOUNTS]
-bh, --beehive          Use beehive configuration for multiple accounts, one
                        account per hex. Make sure to keep -st under 5, and -w
                        under the total amount of accounts available.
                        [env var: POGOMAP_BEEHIVE]
-wph WORKERS_PER_HIVE, --workers-per-hive WORKERS_PER_HIVE
                        Only referenced when using --beehive. Sets number of
                        workers per hive. Default value is 1.
                        [env var: POGOMAP_WORKERS_PER_HIVE]
-l LOCATION, --location LOCATION
                        Location, can be an address or coordinates.
                        [env var: POGOMAP_LOCATION]
-alt ALTITUDE, --altitude ALTITUDE
                        Default altitude in meters.
                        [env var: POGOMAP_ALTITUDE]
-altv ALTITUDE_VARIANCE, --altitude-variance ALTITUDE_VARIANCE
                        Variance for --altitude in meters
                        [env var: POGOMAP_ALTITUDE_VARIANCE]
-uac, --use-altitude-cache
                        Query the Elevation API for each step, rather than
                        only once, and store results in the database.
                        [env var: POGOMAP_USE_ALTITUDE_CACHE]
-j, --jitter            Apply random -5m to +5m jitter to location.
                        [env var: POGOMAP_JITTER]
-al, --access-logs      Write web logs to access.log.
                        [env var: POGOMAP_ACCESS_LOGS]
-st STEP_LIMIT, --step-limit STEP_LIMIT
                        Steps. [env var: POGOMAP_STEP_LIMIT]
-gf GEOFENCE_FILE, --geofence-file GEOFENCE_FILE
                        Geofence file to define outer borders of the scan
                        area. [env var: POGOMAP_GEOFENCE_FILE]
-gef GEOFENCE_EXCLUDED_FILE, --geofence-excluded-file GEOFENCE_EXCLUDED_FILE
                        File to define excluded areas inside scan area.
                        Regarded this as inverted geofence. Can be combined
                        with geofence-file.
                        [env var: POGOMAP_GEOFENCE_EXCLUDED_FILE]
-sd SCAN_DELAY, --scan-delay SCAN_DELAY
                        Time delay between requests in scan threads.
                        [env var: POGOMAP_SCAN_DELAY]
--spawn-delay SPAWN_DELAY
                        Number of seconds after spawn time to wait before
                        scanning to be sure the Pokemon is there.
                        [env var: POGOMAP_SPAWN_DELAY]
-enc, --encounter       Start an encounter to gather IVs and moves.
                        [env var: POGOMAP_ENCOUNTER]
-cs, --captcha-solving
                        Enables captcha solving.
                        [env var: POGOMAP_CAPTCHA_SOLVING]
-ck CAPTCHA_KEY, --captcha-key CAPTCHA_KEY
                        2Captcha API key. [env var: POGOMAP_CAPTCHA_KEY]
-cds CAPTCHA_DSK, --captcha-dsk CAPTCHA_DSK
                        Pokemon Go captcha data-sitekey.
                        [env var: POGOMAP_CAPTCHA_DSK]
-mcd MANUAL_CAPTCHA_DOMAIN, --manual-captcha-domain MANUAL_CAPTCHA_DOMAIN
                        Domain to where captcha tokens will be sent. [env var:
```

(continues on next page)

(continued from previous page)

```
                        POGOMAP_MANUAL_CAPTCHA_DOMAIN]
  -mcr MANUAL_CAPTCHA_REFRESH, --manual-captcha-refresh MANUAL_CAPTCHA_REFRESH
                        Time available before captcha page refreshes. [env
                        var: POGOMAP_MANUAL_CAPTCHA_REFRESH]
  -mct MANUAL_CAPTCHA_TIMEOUT, --manual-captcha-timeout MANUAL_CAPTCHA_TIMEOUT
                        Maximum time captchas will wait for manual captcha
                        solving. On timeout, if enabled, 2Captcha will be used
                        to solve captcha. Default is 0.
                        [env var: POGOMAP_MANUAL_CAPTCHA_TIMEOUT]
  -ed ENCOUNTER_DELAY, --encounter-delay ENCOUNTER_DELAY
                        Time delay between encounter pokemon in scan threads.
                        [env var: POGOMAP_ENCOUNTER_DELAY]
  -ignf IGNORELIST_FILE, --ignorelist-file IGNORELIST_FILE
                        File containing a list of Pokemon IDs to ignore, one
                        line per ID. Spawnpoints will be saved, but ignored
                        Pokemon won't be encountered, sent to webhooks or
                        saved to the DB. [env var: POGOMAP_IGNORELIST_FILE]
  -encwf ENC_WHITELIST_FILE, --enc-whitelist-file ENC_WHITELIST_FILE
                        File containing a list of Pokemon IDs to encounter for
                        IV/CP scanning. One line per ID.
                        [env var: POGOMAP_ENC_WHITELIST_FILE]
  -nostore, --no-api-store
                        Don't store the API objects used by the high level
                        accounts in memory. This will increase the number of
                        logins per account, but decreases memory usage.
                        [env var: POGOMAP_NO_API_STORE]
  -apir API_RETRIES, --api-retries API_RETRIES
                        Number of times to retry an API request.
                        [env var: POGOMAP_API_RETRIES]
  -wwhtf WEBHOOK_WHITELIST_FILE, --webhook-whitelist-file WEBHOOK_WHITELIST_FILE
                        File containing a list of Pokemon IDs or names to be sent
                        to webhooks. [env var: POGOMAP_WEBHOOK_WHITELIST_FILE]
  -ld LOGIN_DELAY, --login-delay LOGIN_DELAY
                        Time delay between each login attempt.
                        [env var: POGOMAP_LOGIN_DELAY]
  -lr LOGIN_RETRIES, --login-retries LOGIN_RETRIES
                        Number of times to retry the login before refreshing a
                        thread. [env var: POGOMAP_LOGIN_RETRIES]
  -mf MAX_FAILURES, --max-failures MAX_FAILURES
                        Maximum number of failures to parse locations before
                        an account will go into a sleep for -ari/--account-
                        rest-interval seconds. [env var: POGOMAP_MAX_FAILURES]
  -me MAX_EMPTY, --max-empty MAX_EMPTY
                        Maximum number of empty scans before an account will
                        go into a sleep for -ari/--account-rest-interval
                        seconds.Reasonable to use with proxies.
                        [env var: POGOMAP_MAX_EMPTY]
  -bsr BAD_SCAN_RETRY, --bad-scan-retry BAD_SCAN_RETRY
                        Number of bad scans before giving up on a step.
                        Default 2, 0 to disable.
                        [env var: POGOMAP_BAD_SCAN_RETRY]
  -msl MIN_SECONDS_LEFT, --min-seconds-left MIN_SECONDS_LEFT
                        Time that must be left on a spawn before considering
                        it too late and skipping it. For example 600 would
                        skip anything with < 10 minutes remaining. Default 0.
                        [env var: POGOMAP_MIN_SECONDS_LEFT]
  -dc, --display-in-console
```

(continues on next page)

(continued from previous page)

```
                      Display Found Pokemon in Console.
                      [env var: POGOMAP_DISPLAY_IN_CONSOLE]
-H HOST, --host HOST  Set web server listening host. [env var: POGOMAP_HOST]
-P PORT, --port PORT  Set web server listening port. [env var: POGOMAP_PORT]
-L LOCALE, --locale LOCALE
                      Locale for Pokemon names (default: en, check
                      static/dist/locales for more).
                      [env var: POGOMAP_LOCALE]
-c, --china           Coordinates transformer for China.
                      [env var: POGOMAP_CHINA]
-m MOCK, --mock MOCK  Mock mode - point to a fpgo endpoint instead of using
                      the real PogoApi, ec: http://127.0.0.1:9090
                      [env var: POGOMAP_MOCK]
-ns, --no-server      No-Server Mode. Starts the searcher but not the
                      Webserver. [env var: POGOMAP_NO_SERVER]
-os, --only-server    Server-Only Mode. Starts only the Webserver without
                      the searcher. [env var: POGOMAP_ONLY_SERVER]
-sc, --search-control
                      Enables search control.
                      [env var: POGOMAP_SEARCH_CONTROL]
-nfl, --no-fixed-location
                      Disables a fixed map location and shows the search bar
                      for use in shared maps. [env var:
                      POGOMAP_NO_FIXED_LOCATION]
-k GMAPS_KEY, --gmaps-key GMAPS_KEY
                      Google Maps Javascript API Key.
                      [env var: POGOMAP_GMAPS_KEY]
--skip-empty          Enables skipping of empty cells in normal scans -
                      requires previously populated database (not to be used
                      with -ss) [env var: POGOMAP_SKIP_EMPTY]
-C, --cors            Enable CORS on web server. [env var: POGOMAP_CORS]
-cd, --clear-db       Deletes the existing database before starting the
                      Webserver. [env var: POGOMAP_CLEAR_DB]
-np, --no-pokemon     Disables Pokemon from the map (including parsing them
                      into local db.) [env var: POGOMAP_NO_POKEMON]
-ng, --no-gyms        Disables Gyms from the map (including parsing them
                      into local db). [env var: POGOMAP_NO_GYMS]
-nr, --no-raids       Disables Raids from the map (including parsing them
                      into local db). [env var: POGOMAP_NO_RAIDS]
-nk, --no-pokestops   Disables PokeStops from the map (including parsing
                      them into local db). [env var: POGOMAP_NO_POKESTOPS]
-ss [SPAWNPOINT_SCANNING], --spawnpoint-scanning [SPAWNPOINT_SCANNING]
                      Use spawnpoint scanning (instead of hex grid). Scans
                      in a circle based on step_limit when on DB.
                      [env var: POGOMAP_SPAWNPOINT_SCANNING]
-ssct SS_CLUSTER_TIME, --ss-cluster-time SS_CLUSTER_TIME
                      Time threshold in seconds for spawn point clustering
                      (0 to disable). [env var: POGOMAP_SS_CLUSTER_TIME]
-speed, --speed-scan  Use speed scanning to identify spawn points and then
                      scan closest spawns. [env var: POGOMAP_SPEED_SCAN]
-spin, --pokestop-spinning
                      Spin Pokestops with 50% probability.
                      [env var: POGOMAP_POKESTOP_SPINNING]
-ams ACCOUNT_MAX_SPINS, --account-max-spins ACCOUNT_MAX_SPINS
                      Maximum number of Pokestop spins per hour.
                      [env var: POGOMAP_ACCOUNT_MAX_SPINS]
-kph KPH, --kph KPH   Set a maximum speed in km/hour for scanner movement.
```

(continues on next page)

(continued from previous page)

```
                        0 to disable. Default: 35. [env var: POGOMAP_KPH]
  -hkph HLVL_KPH, --hlvl-kph HLVL_KPH
                        Set a maximum speed in km/hour for scanner movement,
                        for high-level (L30) accounts. 0 to disable.
                        Default: 25. [env var: POGOMAP_HLVL_KPH]
  -ldur LURE_DURATION, --lure-duration LURE_DURATION
                        Change duration for lures set on pokestops. This is
                        useful for events that extend lure duration.
                        [env var: POGOMAP_LURE_DURATION]
  -px PROXY, --proxy PROXY
                        Proxy url (e.g. socks5://127.0.0.1:9050)
                        [env var: POGOMAP_PROXY]
  -pxsc, --proxy-skip-check
                        Disable checking of proxies before start.
                        [env var: POGOMAP_PROXY_SKIP_CHECK]
  -pxt PROXY_TEST_TIMEOUT, --proxy-test-timeout PROXY_TEST_TIMEOUT
                        Timeout settings for proxy checker in seconds.
                        [env var: POGOMAP_PROXY_TEST_TIMEOUT]
  -pxre PROXY_TEST_RETRIES, --proxy-test-retries PROXY_TEST_RETRIES
                        Number of times to retry sending proxy test requests
                        on failure. [env var: POGOMAP_PROXY_TEST_RETRIES]
  -pxbf PROXY_TEST_BACKOFF_FACTOR, --proxy-test-backoff-factor PROXY_TEST_BACKOFF_
↪FACTOR
                        Factor (in seconds) by which the delay until next
                        retry will increase.
                        [env var: POGOMAP_PROXY_TEST_BACKOFF_FACTOR]
  -pxc PROXY_TEST_CONCURRENCY, --proxy-test-concurrency PROXY_TEST_CONCURRENCY
                        Async requests pool size.
                        [env var: POGOMAP_PROXY_TEST_CONCURRENCY]
  -pxd PROXY_DISPLAY, --proxy-display PROXY_DISPLAY
                        Display info on which proxy being used (index or
                        full). To be used with -ps.
                        [env var: POGOMAP_PROXY_DISPLAY]
  -pxf PROXY_FILE, --proxy-file PROXY_FILE
                        Load proxy list from text file (one proxy per line),
                        overrides -px/--proxy. [env var: POGOMAP_PROXY_FILE]
  -pxr PROXY_REFRESH, --proxy-refresh PROXY_REFRESH
                        Period of proxy file reloading, in seconds. Works only
                        with -pxf/--proxy-file. (0 to disable).
                        [env var: POGOMAP_PROXY_REFRESH]
  -pxo PROXY_ROTATION, --proxy-rotation PROXY_ROTATION
                        Enable proxy rotation with account changing for search
                        threads (none/round/random).
                        [env var: POGOMAP_PROXY_ROTATION]
  -wh WEBHOOKS, --webhook WEBHOOKS
                        Define URL(s) to POST webhook information to. [env
                        var: POGOMAP_WEBHOOK]
  -gi, --gym-info       Get all details about gyms (causes an additional API
                        hit for every gym). [env var: POGOMAP_GYM_INFO]
  -DC, --enable-clean   Enable DB cleaner. [env var: POGOMAP_ENABLE_CLEAN]
  --wh-types {pokemon,gym,raid,egg,tth,gym-info,pokestop,lure,captcha}
                        Defines the type of messages to send to webhooks.
                        [env var: POGOMAP_WH_TYPES]
  --wh-threads WH_THREADS
                        Number of webhook threads; increase if the webhook
                        queue falls behind. [env var: POGOMAP_WH_THREADS]
  -whc WH_CONCURRENCY, --wh-concurrency WH_CONCURRENCY
```

(continues on next page)

(continued from previous page)

```
                        Async requests pool size.
                        [env var: POGOMAP_WH_CONCURRENCY]
  -whr WH_RETRIES, --wh-retries WH_RETRIES
                        Number of times to retry sending webhook data on
                        failure. [env var: POGOMAP_WH_RETRIES]
  -whct WH_CONNECT_TIMEOUT, --wh-connect-timeout WH_CONNECT_TIMEOUT
                        Connect timeout (in seconds) for webhook requests.
                        [env var: POGOMAP_WH_CONNECT_TIMEOUT]
  -whrt WH_READ_TIMEOUT, --wh-read-timeout WH_READ_TIMEOUT
                        Read timeout (in seconds) for webhook requests.
                        [env var: POGOMAP_WH_READ_TIMEOUT]
  -whbf WH_BACKOFF_FACTOR, --wh-backoff-factor WH_BACKOFF_FACTOR
                        Factor (in seconds) by which the delay until next
                        retry will increase. [env var:
                        POGOMAP_WH_BACKOFF_FACTOR]
  -whlfu WH_LFU_SIZE, --wh-lfu-size WH_LFU_SIZE
                        Webhook LFU cache max size.
                        [env var: POGOMAP_WH_LFU_SIZE]
  -whfi WH_FRAME_INTERVAL, --wh-frame-interval WH_FRAME_INTERVAL
                        Minimum time (in ms) to wait before sending the next
                        webhook data frame.
                        [env var: POGOMAP_WH_FRAME_INTERVAL]
  --ssl-certificate SSL_CERTIFICATE
                        Path to SSL certificate file.
                        [env var: POGOMAP_SSL_CERTIFICATE]
  --ssl-privatekey SSL_PRIVATEKEY
                        Path to SSL private key file.
                        [env var: POGOMAP_SSL_PRIVATEKEY]
  -ps [logs], --print-status [logs]
                        Show a status screen instead of log messages. Can
                        switch between status and logs by pressing enter.
                        Optionally specify "logs" to startup in logging mode.
                        [env var: POGOMAP_PRINT_STATUS]
  -slt STATS_LOG_TIMER, --stats-log-timer STATS_LOG_TIMER
                        In log view, list per hr stats every X seconds
                        [env var: POGOMAP_STATS_LOG_TIMER]
  -sn STATUS_NAME, --status-name STATUS_NAME
                        Enable status page database update using STATUS_NAME
                        as main worker name. [env var: POGOMAP_STATUS_NAME]
  -spp STATUS_PAGE_PASSWORD, --status-page-password STATUS_PAGE_PASSWORD
                        Set the status page password. [env var:
                        POGOMAP_STATUS_PAGE_PASSWORD]
  -hk HASH_KEY, --hash-key HASH_KEY
                        Key for hash server [env var: POGOMAP_HASH_KEY]
  -novc, --no-version-check
                        Disable API version check. [env var:
                        POGOMAP_NO_VERSION_CHECK]
  -vci VERSION_CHECK_INTERVAL, --version-check-interval VERSION_CHECK_INTERVAL
                        Interval to check API version in seconds (Default: in
                        [60, 300]). [env var: POGOMAP_VERSION_CHECK_INTERVAL]
  -el ENCRYPT_LIB, --encrypt-lib ENCRYPT_LIB
                        Path to encrypt lib to be used instead of the shipped
                        ones. [env var: POGOMAP_ENCRYPT_LIB]
  -odt ON_DEMAND_TIMEOUT, --on-demand_timeout ON_DEMAND_TIMEOUT
                        Pause searching while web UI is inactive for this
                        timeout (in seconds). [env var:
                        POGOMAP_ON_DEMAND_TIMEOUT]
```

(continues on next page)

(continued from previous page)

```
  --disable-blacklist   Disable the global anti-scraper IP blacklist.
                        [env var: POGOMAP_DISABLE_BLACKLIST]
  -tp TRUSTED_PROXIES, --trusted-proxies TRUSTED_PROXIES
                        Enables the use of X-FORWARDED-FOR headers to identify
                        the IP of clients connecting through these trusted
                        proxies. [env var: POGOMAP_TRUSTED_PROXIES]
  --api-version API_VERSION
                        API version currently in use.
                        [env var: POGOMAP_API_VERSION]
  --no-file-logs        Disable logging to files. Does not disable --access-
                        logs. [env var: POGOMAP_NO_FILE_LOGS]
  --log-path LOG_PATH   Defines directory to save log files to.
                        [env var: POGOMAP_LOG_PATH]
  --log-filename LOG_FILENAME
                        Defines the log filename to be saved. Allows date
                        formatting, and replaces <SN> with the instance's
                        status name. Read the python time module docs for
                        details. Default: %Y%m%d_%H%M_<SN>.log. [env var:
                        POGOMAP_LOG_FILENAME]
  --dump                Dump censored debug info about the environment and
                        auto-upload to hastebin.com. [env var: POGOMAP_DUMP]
  -v                    Show debug messages from RocketMap and pgoapi. Can be
                        repeated up to 3 times.
  --verbosity VERBOSE   Show debug messages from RocketMap and pgoapi.
                        [env var: POGOMAP_VERBOSITY]

Database:
  --db-name DB_NAME     Name of the database to be used.
                        [env var: POGOMAP_DB_NAME]
  --db-user DB_USER     Username for the database. [env var: POGOMAP_DB_USER]
  --db-pass DB_PASS     Password for the database. [env var: POGOMAP_DB_PASS]
  --db-host DB_HOST     IP or hostname for the database.
                        [env var: POGOMAP_DB_HOST]
  --db-port DB_PORT     Port for the database. [env var: POGOMAP_DB_PORT]
  --db-threads DB_THREADS
                        Number of db threads; increase if the db queue falls
                        behind. [env var: POGOMAP_DB_THREADS]

Database Cleanup:
  -DC, --db-cleanup     Enable regular database cleanup thread.
                        [env var: POGOMAP_DB_CLEANUP]
  -DCw DB_CLEANUP_WORKER, --db-cleanup-worker DB_CLEANUP_WORKER
                        Clear worker status from database after X minutes of
                        inactivity. Default: 30, 0 to disable.
                        [env var: POGOMAP_DB_CLEANUP_WORKER]
  -DCp DB_CLEANUP_POKEMON, --db-cleanup-pokemon DB_CLEANUP_POKEMON
                        Clear pokemon from database X hours after they
                        disappeared. Default: 0, 0 to disable.
                        [env var: POGOMAP_DB_CLEANUP_POKEMON]
  -DCg DB_CLEANUP_GYM, --db-cleanup-gym DB_CLEANUP_GYM
                        Clear gym details from database X hours after last gym
                        scan. Default: 8, 0 to disable.
                        [env var: POGOMAP_DB_CLEANUP_GYM]
  -DCs DB_CLEANUP_SPAWNPOINT, --db-cleanup-spawnpoint DB_CLEANUP_SPAWNPOINT
                        Clear spawnpoint from database X hours after last
                        valid scan. Default: 720, 0 to disable.
                        [env var: POGOMAP_DB_CLEANUP_SPAWNPOINT]
```

(continues on next page)

(continued from previous page)

```
  -DCf DB_CLEANUP_FORTS, --db-cleanup-forts DB_CLEANUP_FORTS
                        Clear gyms and pokestops from database X hours after
                        last valid scan. Default: 0, 0 to disable.
                        [env var: POGOMAP_DB_CLEANUP_FORTS]

Dynamic Rarity:
  -Rh RARITY_HOURS, --rarity-hours RARITY_HOURS
                         Number of hours of Pokemon data to use to calculate
                         dynamic rarity. Decimals allowed. Default: 48. 0 to
                         use all data. [env var: POGOMAP_RARITY_HOURS]
   -Rf RARITY_UPDATE_FREQUENCY, --rarity-update-frequency RARITY_UPDATE_FREQUENCY
                         How often (in minutes) the dynamic rarity should be
                         updated. Decimals allowed. Default: 0. 0 to disable.
                         [env var: POGOMAP_RARITY_UPDATE_FREQUENCY]
```

CHAPTER 5

Configuration files (Outdated)

Configuration files can be used to organize server/scanner deployments. Any long-form command-line argument can be specified in a configuration file.

5.1 Default file

The default configuration file is *config/config.ini* underneath the project home. However, this location can be changed by setting the environment variable POGOMAP_CONFIG or using the -cf or –config flag on the command line. In the event that both the environment variable and the command line argument exists, the command line value will take precedence. Note that all relative pathnames are relative to the current working directory (often, but not necessarily where runserver.py is located).

5.2 Setting configuration key/value pairs

For command line values that take a single value they can be specified as:

```
keyname: value
e.g.   host: 0.0.0.0
```

For parameters that may be repeated:

```
keyname: [ value1, value2, ...]
e.g.   username: [ randomjoe, bonnieclyde ]

*for usernames and passwords, the first username must correspond to the first␣
↪password, and so on *
```

For command line arguments that take no parameters:

```
keyname: True
e.g.   fixed-location: True
```

5.3 Example config file

```
auth-service: ptc
username: [ username1, username2 ]
password: [ password1, password2 ]
location: seattle, wa
step-limit: 5
gmaps-key: MyGmapsKeyGoesHereSomeLongString
hash-kay: MyHashingKeyGoesHere
print-status: "status"
speed-scan: True
```

Running this config file as:

```
python runserver.py -cf myconfig.seattle
```

would be the same as running with the following command line:

```
python runserver.py -u randomjoe -p password1 -u bob -p password2 -l "seattle, wa" -
↪st 5 -k MyGmapsKeyGoesHereSomeLongString -ps
```

5.4 Shared config

If you run multiple instances, you can add settings to a shared configuration file rather than adding it to each unique instance configuration file individually. This is useful for settings that are always the same such as Google Maps API key, hashing key, etc. It makes managing multiple instances easier: instead of having to change the hashing key in every configuration file every month, you only need to change it in the shared config file. Add the shared settings to your shared-config.ini and call the shared config from the command line using the `-scf` flag when you start your instance.

```
python runserver.py -cf myconfig.ini -scf shared-config.ini
```

Using `--shared-config` in a configuration file does not work.

Remember to remove old keys and any other settings from your normal configs that may cause conflicts. If you set a value like the Google Maps API key in both configuration files, the value set in the regular `-cf` configuration file takes precedence.

5.5 Running multiple configs

One common way of running multiple locations is to use two configuration files each with common or default database values, but with different location specs. The first configuration running as both a scanner and a server, and in the second configuration file, use the *no-server* flag to not start the web interface for the second configuration. In the config file, this would mean including a line like:

no-server: True

5.5.1 Using `-ns` and `-os`

If RocketMap is not scanning enough areas, you can add additional areas by starting a 2nd RocketMap instance with the flag `-ns`. This starts the searchers without starting another webserver. You can run as many instances with `-ns` as your server can keep up with. ***HOWEVER:*** If all your instances are running `-ns` you will also want to start an

instance with `-os`. This will start only the webserver. This becomes useful if you begin to seperate your RM instances across several computers all linked to the same database.

CHAPTER 6

Common Questions and Answers (Outdated)

6.1 Should I use this as a way to make money?

No, it is gross to charge people for maps when the information should be provided by Niantic! We do not endorse paid maps, which is why this platform is opensource.

6.2 All Pokemon are set as ultra rare, what is going on?

We now use a dynamic rarity system to determine the rarity of pokemon, the rarity is calculated by what you scan so it unique to your area. The rarity is updated every hour so when you start an initial scan everything will probably say it is ultra rare for the first hour and then will adjust itself.

6.3 Does dynamic rarity mean that all future pokemon will get a rarity automatically?

Yes.

6.4 What do the spawn point colors mean?

- A **grey** dot represents a spawn point that is more than 5 minutes from spawning.
- A **light blue** dot represents a spawn point that will spawn in 5 minutes. **Light blue** changes to **dark blue** and finally into **purple** just before spawn time.
- A **green dot** represents a fresh spawn. This will transition to **yellow**, through **orange** and finally **red** (like a stop light) as it is about to despawn.

6.5 All I see are numbers! Where are the pokemon?

You are missing the sprite files. Please consult #faq on the RocketMap Discord.

6.6 Lures are 6 hours right now! Why is it saying they have already expired?

You need to add `-ldur 360` to change the lure assumption to 6 hours (360 minutes.)

6.7 Can I sign in with Google?

Yes you can! Pass the flag `-a google` (replacing `-a ptc`) to use Google authentication.

If you happen to have 2-step verification enabled for your Google account you will need to supply an app password for your password instead of your normal login password.

6.8 Which is the best scan option to use to find pokemon?

SpeedScan (`-speed`) is the most used scheduler: it's the only scheduler that currently supports finding the proper spawnpoint time and duration, and it also features a built-in speed limiter to avoid speed violations (i.e. softbans).

More information can be found here : Speed Scheduler

6.9 But I was happy using the default Hex or -ss...

Speed Scheduler combines both and is more efficient, -ss is not being actively maintained and doesn't work unless you already have spawnpoints and timers exported.

6.10 All pokemon disappear after only 1 minute, the map is broken!

One of Niantic's updates removed spawn timers from Pokémon (until there's little time left until they despawn). SpeedScan does an initial scan to determine all spawn points and their timers and automatically transitions into spawn scanning once it found them all. Seeing 1-minute timers during initial scan is perfectly normal.

6.11 What's the simplest command to start the map scanning?

./runserver.py -speed -l LOCATION -a GOOGLE/PTC -u USER -p PASS -k GOOGLEKEY -hk HASHINGHEY
You must replace the values for GOOGLE,PTC/LOCATION/USER/PASS/GOOGLEKEY/HASHINGKEY with your information.

6.12 Nice, what other stuff can I use in the command line?

There is a list here or a more up to date list can be found by running ./runserver.py -h

6.13 Woah I added a ton of cool stuff and now my command line is massive, any way to shorten it?

It is a lot simplier to use a config file

6.14 Can I scan for free or do I need to pay for a hashing key?

Using a hashing key is mandatory at this point. The API will not function without an active hashing key from Bossland. More Informatiion about Hashing Keys

6.15 Is there anything I can do to lower captchas?

Yes, you can enable pokestop spinning to level your workers to level two (spinning a single pokéstop), this reduces captchas a lot. You may also consider scanning a smaller area, using less workers or encountering less pokemon for IV.

6.16 How many workers do I need?

There is no simple answer to this, it all depends on your -st and more importantly how spawn dense that location is.For a rough guide you can use the formulas at the bottom of this page.

6.17 example.py isn't working right!

Seb deleted it, it was the only good thing left in our lives. Seb has murdered us all.

6.18 How do I setup port forwarding?

See this helpful guide

6.19 I edited my files/installed unfinished code and messed up, will you help me fix it?

No, the best course of action is to delete it all and start again, this time don't edit files unless you know what you are doing.

6.20 I used a PR and now everything is messed up! HELP ME!

No, remove everything and start from scratch. A Pull Request is merged when it meets the standards of the project.

6.21 "It's acting like the location flag is missing."

-l, never forget.

6.22 I overridden watchdog and now all my accounts are flagged/banned.

Good Job! We recommend making new accounts. Current Tools are Here!

6.23 I get an error about PGoAPI version

You can get a warning which pauses your scanner due to a new API being forced. In that case check for announcements to see if a new version has been released, and update as it says.

In case your server does not start due to a PGoAPI version mismatch the problem is that you are trying to start your server with an different installed PGoAPI that the one it is built for, to update your PGoAPI installation to the required version do:

```
pip uninstall pgoapi
pip install --upgrade -r requirements.txt
```

Use `sudo` or `--user` if you don't have permissions to install modules.

6.24 I'm getting this error...

6.24.1 Python version

```
pip or python is not recognized as an internal or external command
```

Python/pip has not been added to the environment

```
Exception, e <- Invalid syntax.
```

This error is caused by Python 3. The project requires python 2.7

6.24.2 Gcc missing

```
error: command 'gcc' failed with exit status 1

# - or -

[...]failed with error code 1 in /tmp/pip-build-k3oWzv/pycryptodomex/
```

Your OS is missing the `gcc` compiler library. For Debian, run `apt-get install build-essentials`. For Red Hat, run `yum groupinstall 'Development Tools'`

6.24.3 KeyError

```
cells = map_dict['responses']['GET_MAP_OBJECTS']['map_cells']

KeyError: 'map_cells'
```

The account is banned.

6.24.4 Database error

```
InternalError(1054, u"unknown column 'cp' in 'field list'") or similar
```

Only one instance can run when the database is being modified or upgraded. Run ***ONE*** instance of RM with `-cd` to wipe your database, then run ***ONE*** instance of RM (without `-cd`) to setup your database.

6.24.5 Certificate errors

```
Unable to retrieve altitude from Google APIs: [SSL: CERTIFICATE_VERIFY_FAILED]␣
↪certificate verify failed (_ssl.c:579).
```

RedHat based distros (Fedora, CentOS) could have an old OpenSSL version that is not compatible to the latest `certifi` package, to fix it you need to:

```
pip uninstall certifi
pip install certifi==2015.4.28
```

Use `sudo` or `--user` if you are not using an account with root permission.

And remember that you should do this every time after updating the requirements of the project.

6.25 I have more questions!

Please read all wiki pages relating to the specific function you are questioning. If it does not answer your question, join us on the RocketMap Discord. Before asking questions in #help on Discord, make sure you've read #announcements and #faq.

6.26 Formulas?

st=step distancesd=scan delay [default: 10]w=# of workerst=desired scan time

6.26.1 Speed Scan

Workers for initial scan(speed scan):

```
cells = (((st * (st - 1)) * 3) + 1)
workers = cells / 20
```

an example for st 19: (((19 * 18) * 3) +1) / 20 = 51.35 so use 52 workers.You will not need as many accounts once initial scan is complete.

6.26.2 Hex Scan

time to scan: `(sd/w)*(3st^2-3st+1)` time to scan (using default scan delay): `(10/w)*(3st^2-3st+1)`

workers needed: `(sd/t)*(3st^2-3st+1)` workers needed (using default scan delay): `(10/t)*(3st^2-3st+1)`

CHAPTER 7

Community Tools (Outdated)

Some useful tools made by the community for the community

7.1 KinanCity

7.1.1 Java tool for creating PTC Accounts, based on Pallettown.

Used to generate any desired number of PTC accounts for use with RocketMap. Also allows for verification if you own your own domain.

7.2 Node.js CORS proxy server

7.2.1 A simple CORS proxy server that supports HTTP(S) request proxying.

This script allows you to verify accounts using a CORS proxy.

Note 1: HTTPS is required for use with the gmail verification script. Self-signed certificates work, but remember to add them to your trusted certificates on your OS.

Note 2: An update to the gmail verification script is planned to stay below the Google API limits (even if you refresh), retry on 503, automatically sleep when no proxies are available, and continue immediately when a proxy becomes available. For now, you'll have to refresh yourself.

7.3 PGM Multi Loc

7.3.1 Easily visualize locations on a map before scanning, and generate a customized launch script.

Add multiple scan locations on the map. Automatically convert an area to a beehive. Resize and move the location on the map. Disable individual hives to stop scanning a specific location.

Generate a customized launch script, with the ability to edit the templates used for the individual commands. Pass in a list of account information that contains usernames, passwords, proxies, etc.

7.4 PokeAlarm

7.4.1 PokeAlarm is a third party extension for Rocket Map that allows you to receive customized notifications

You can setup notifications via Twitter, Facebook Pages, Discord, Pushbullet, Slack, Telegram, and Twilio.

CHAPTER 8

Custom CSS styles (Outdated)

RocketMap supports the use of a custom CSS file to override the default selections. Use Google Chrome developer tools or Mozilla Firebug to easily find the right element selections.

8.1 Use of custom styles

Place your custom CSS into 'custom.css' in the folder 'static/css'. *Examples are found in 'static/css/custom.css.example'.*

8.2 CSS classes

There are four different classes added to style individual map templates.

- Use class .mapPage for styles in map.html.
- Use class .mobilePage for styles in mobile_list.html.
- Use class .statisticsPage for styles in statistics.html.
- Use class .statusPage for styles in status.html.

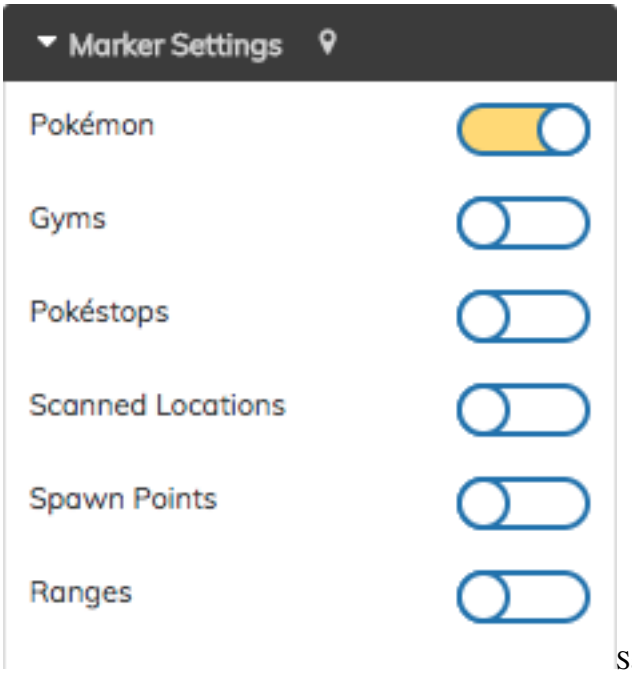
8.2.1 Examples

Examples below are included in custom.css.example!

Set font-size to 11px in map.html and use default style in all other pages.

```
.mapPage #nav h3 {font-size: 11px;}
```

Options menu form-control/switch-container, on/off switch style



Sample Image

```
.form-control input[type=text] {font-size: 11px !important;}
.onoffswitch-label, .onoffswitch-label:before {border-color: #256db6 !important;}
#nav .onoffswitch-checkbox:checked+.onoffswitch-label {border-color: #256db6;␣
↪background-color: #ffde4e;}
.select2-container {font-size: 11px;}
```

Options menu Reset button style

Sample Image

```
.mapPage button {border: 1px solid #256db6; background-color: #fff; color: #256db6 !
↪important; font-size: 11px;}
.mapPage button:hover {background-color: #ffde4e;}
```

See 'custom.css.example' for more information.

CHAPTER 9

Custom.js (Outdated)

RocketMap supports the use of a custom JavaScript file to run custom code instead of editing core files.

9.1 Warning of using code you don't understand

Never just put code in custom.js unless you understand what it does, someone could give you malicious JavaScript code that could result in credentials (accounts and keys) being stolen!

Please do not copy/paste code from strangers online!

9.2 Use of custom.js

Place your custom code into 'custom.js' in the folder 'static/js'. *Examples are found in 'static/js/custom.js.example'.*

9.2.1 Examples

Examples for a MOTD, google analytics and disabling scaling icons by rarity are included in custom.js.example! The example below is how to set default options usually done by editing core files.

Set a new map style by default. First we set a variable for it and pick which style we want as default.

```
const map_style = 'satellite'
```

Next we have to tell the script to store the value to set it.

```
Store.set('map_style', map_style)
```

Whenever you edit custom.js you will have to run `npm run build` to set the changes. When you load the map it will be set to satellite as default.

Setting options in this way forces that setting on page load, so even if a user changes the setting it will revert back to what you have set in custom.js every time, keep this in mind when forcing settings.

See 'custom.js.example' for more information.

CHAPTER 10

Access Your Map Anywhere (Outdated)

aka External Access to RocketMap

10.1 Introduction

This guide will show you how to make your map available on the internet, including yourself on the go. These instructions should **not** be used on a server you are giving out to other people for use, as it is not the most secure option available. **We are not responsible for damage caused to your software, equipment, or anything else, proceed at your own risk.**

This guide will most likely not match your router exactly, as different manufacturers have different software and configuration. This is meant to be a general guide as many routers have a lot in common, but if you can't follow these following instructions, try Googling something like "Port forwarding [*MY ROUTER MODEL*]"

10.2 Gathering Information

First we should find some information about your network. Press `Win+R` and type "cmd" on Windows to open a Command Prompt. On Linux and OS X, open a Terminal application.

On Windows, enter the following command:

```
ipconfig
```

On Linux and OS X, enter:

```
ifconfig
```

A lot of information will appear on your screen, but these two lines are what we're looking for:

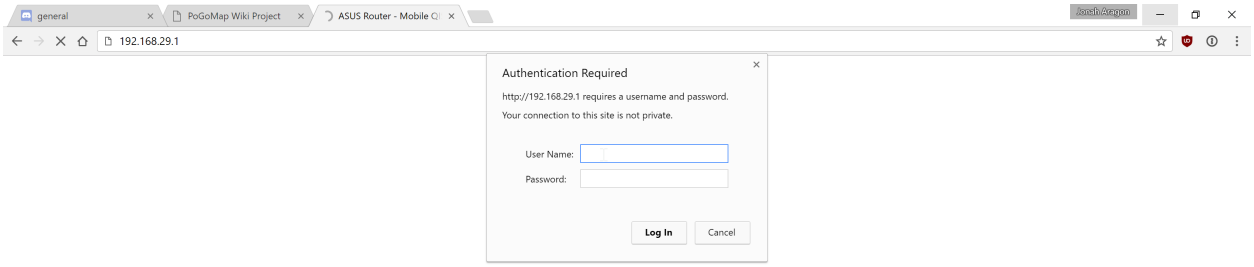
```
IPv4 Address. . . . . . . . . . . : 192.168.29.22
Default Gateway . . . . . . . . . : 192.168.29.1
```

Obviously, your numbers (IP Addresses) will most likely look different from mine. Jot those two down so we can use them later.

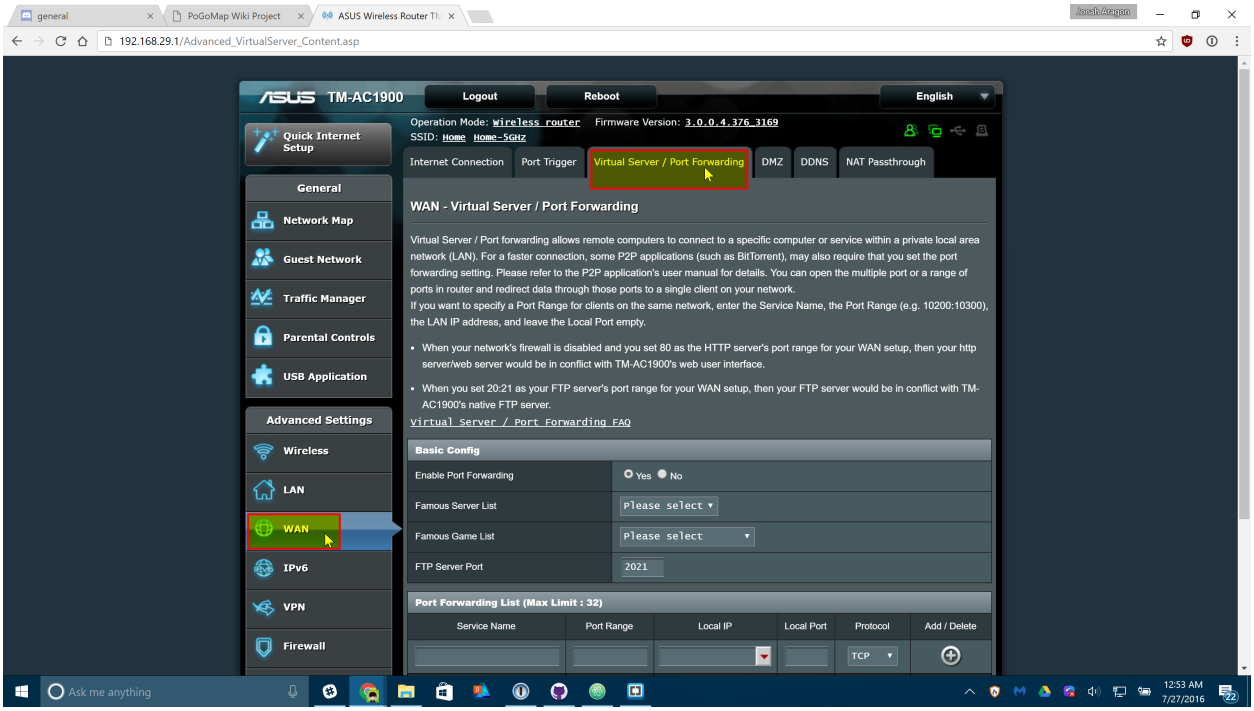
Let's also find your Public IP address, browse to mxtoolbox.com/whatismyip and note the IP Address it gives you there.

10.3 Port Forwarding

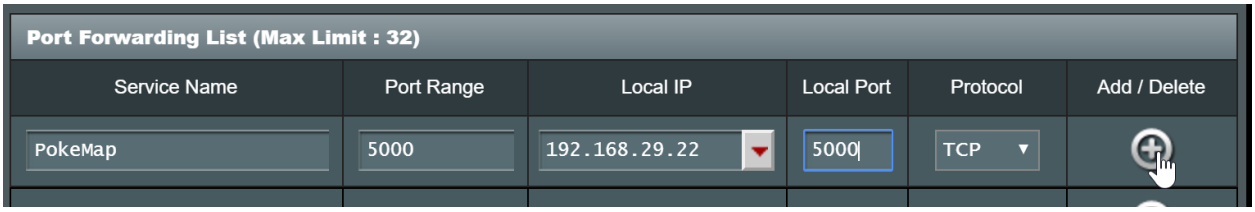
Open an internet browser and type in the "Default Gateway" IP address we just found in the last step. It may ask you to login at this point, enter the login credentials for your router. It is important to note that this generally isn't your wifi password, if you don't know your router login credentials, they are usually on the bottom of the router unless they've already been changed.



Now you should be on the homepage of your router's configuration. Find the Port Forwarding section of your router. On mine it was under "WAN" > "Virtual Server / Port Forwarding" but yours may differ.



Enter the following information on that screen. Some routers may make you press a menu before you can see these text boxes:



Replace "Local IP" (sometimes called "Internal IP") with the IPv4 Address we found in the first step of this guide. Both port boxes should have the same number, 5000. If you only have one box for ports just enter "5000" in that one. Click the Add button to save your settings. If you have an option to choose from UDP or TDP, choose TCP.

10.4 Finishing Up

Your port should now be added! Next time you start your server, add `-H 0.0.0.0` to the list of flags so it's accessible from the internet!

10.5 Connecting

In any web browser not connected to your wifi, your phone for instance, browse to `http://11.22.33.44:5000/` **replacing** 11.22.33.44 with your external IP address, and you should be set!

Note: It is a known bug that Safari on iOS may not be able to load the map, if this happens to you download Chrome from the app store.

CHAPTER 11

Custom sprites support (Outdated)

If use of other sprites than provided in ‘static01.zip’ custom sprites should be placed in ‘static/icons’. Minimum pixel size of images is 48x48.

CHAPTER 12

Geofences (Outdated)

With the help of geofences you can define your search area even better. This feature let's you line out areas you are interested in without scanning overhead. Also you can exclude areas where no scan should happen due to the sake of security or respective issues. Lastly, with Geofences you can scan geometries which were not possible to define in before.

12.1 Speed

The included algorithm should be fast enough, but if you see your geofencing takes too long, there are some things you can do to improve geofencing times:

- Try to make the polygon simpler removing vertexes.
- Reduce step size to better fit your polygon.
- Install `matplotlib`.

12.2 Matplotlib

The geofence calculations can be faster if the powerful `matplotlib` python package is installed, in that case it will use it instead of the included algorithm to check if a point is inside an area. The real improvement varies a lot between setups but it should be faster anyway, in tests we have seen it ranging from 12% to 100%.

The install procedure is the same as any other python package: `pip install matplotlib`

You can see in the logs if RM is using `matplotlib` or not for the calculations. While this is a powerful tool, it also has its downsides that it may not be supported on certain devices, for example older Raspberry Pi.

12.3 How to use?

1. Define areas which you like geofence or exclude, from your standard hex. Best done via an online tool like this (export using Show Coordinates at the top) or this one (use the exp button on the left after creating a fence). The resulting format needs to match the content of `geofences/geofence.txt.example` or `geofences/excluded.txt.example`.
2. You may exactly use one file per instance for geofenced areas and one file for excluded areas. Put all areas into the correct file like it is done in the examples.
3. Activate geofencing by adding these file paths as flag arguments to either `-gf` / `--geofence-file` or `-gef`/ `--geofence-excluded-file` in CLI. This can be done via the configuration file, as well.
4. Best, place your scan location `-l` / `--location` on top of a valid area of your geofence file and adjust `-st` / `--step-limit` to a value which just exceeds the maximum desired scan radius by a bit.

12.3.1 Optional

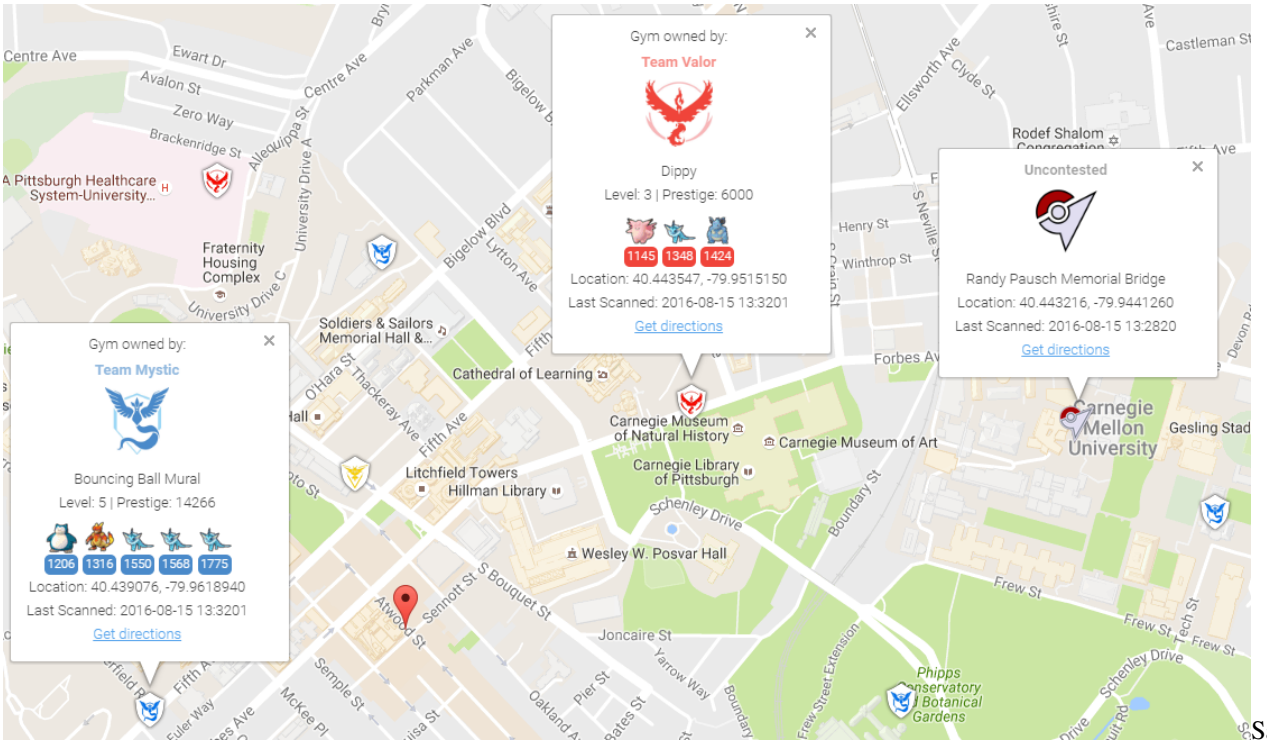
You can define geofences to form interesting areas like Pikachu faces or scanning along long paths with defining just a small corridor as geofence and rise `-st` accordingly.

12.3.2 Beehive

This feature explicitly works with beehive `-bh` scan function just as good. Please make sure that the very center of your scan area - meaning `-l` is right inside a valid geofence area.

CHAPTER 13

Detailed Gym Data (Outdated)



Sample Image

To collect all of the available information about gyms, you can enable gym info parsing.

Gym info parsing adds the gym's name and a list of all the Pokemon currently in the gym to the GUI. However, this comes at a slight cost: An extra API call is made for each gym to be updated, which slows the overall scan speed. Gym information is parsed intelligently, and only updates if something about the gym has changed since it was last updated.

13.1 MySQL Recommended

Because of the increased data being sent to the database, it is recommended to use MySQL when using this feature.

13.1.1 Note on Non-Latin Characters in MySQL

Using out of the box settings, MySQL uses a collation/encoding that does not support non-Latin characters. If gyms in your area are displaying with `????` instead of the correct characters, you need to update your database server's collation and encoding to use UTF8. To correct just the gym names, run:

```
ALTER TABLE `gymdetails` COLLATE='utf8_general_ci', CONVERT TO CHARSET utf8;
```

As gym details are refreshed, the correct characters will appear (to force this, you can empty `gymdetails`). More information regarding MySQL encoding is available here.

13.2 Enabling Gym Information Parsing

To enable gym parsing, run with the `-gi` argument:

```
python runserver.py -gi
```

Or update your config.ini:

```
gym-info                    # enables detailed gym info collection (default false)
```

13.3 Displaying the gym details sidebar in the front-end

To show the gym details sidebar on the front-end, `-gi` needs to be enabled on your webserver instance. To scan gyms for the gym details, `-gi` must also be enabled on your scanner instances.

13.4 New Webhook

When gym info parsing is enabled, gym details will be sent to webhooks. A sample webhook is below for reference:

```
{
        "message": {
                "id": "4b432a31c3c247e5b0f7656d09e2c050.11",
                "url": "http://lh3.ggpht.com/yBqXtFfq3nOlZmLc7DbgSIcXcyfvsWfY3VQs_
↪gBziPwjUx7xOfgvucz6uxP_Ri-ianoWFt5mgJ7_zpsa7VNK",
                "name": "Graduate School of Public Health Sculpture",
                "description": "Sculpture on the exterior of the Graduate School of␣
↪Public Health building.",
                "team": 1,
                "latitude": 40.442506,
                "longitude": -79.957962,
                "pokemon": [{
                        "num_upgrades": 0,
                        "move_1": 234,
```

(continues on next page)

(continued from previous page)


```
                        "move_2": 99,
                        "additional_cp_multiplier": 0,
                        "iv_defense": 11,
                        "weight": 14.138585090637207,
                        "pokemon_id": 63,
                        "stamina_max": 46,
                        "cp_multiplier": 0.39956727623939514,
                        "height": 0.7160492539405823,
                        "stamina": 46,
                        "pokemon_uid": 9278614152997308833,
                        "iv_attack": 12,
                        "trainer_name": "SportyGator",
                        "trainer_level": 18,
                        "cp": 138,
                        "iv_stamina": 8
                }, {
                        "num_upgrades": 0,
                        "move_1": 234,
                        "move_2": 87,
                        "additional_cp_multiplier": 0,
                        "iv_defense": 12,
                        "weight": 3.51259708404541,
                        "pokemon_id": 36,
                        "stamina_max": 250,
                        "cp_multiplier": 0.6121572852134705,
                        "height": 1.4966495037078857,
                        "stamina": 250,
                        "pokemon_uid": 6103380929145641793,
                        "iv_attack": 5,
                        "trainer_name": "Meckelangelo",
                        "trainer_level": 22,
                        "cp": 1353,
                        "iv_stamina": 15
                }, {
                        "num_upgrades": 9,
                        "move_1": 224,
                        "move_2": 32,
                        "additional_cp_multiplier": 0.06381925195455551,
                        "iv_defense": 13,
                        "weight": 60.0,
                        "pokemon_id": 31,
                        "stamina_max": 252,
                        "cp_multiplier": 0.5974000096321106,
                        "height": 1.0611374378204346,
                        "stamina": 252,
                        "pokemon_uid": 3580711458547635980,
                        "iv_attack": 10,
                        "trainer_name": "Plaidflamingo",
                        "trainer_level": 23,
                        "cp": 1670,
                        "iv_stamina": 11
                }]
        },
        "type": "gym_details"
}
```

CHAPTER 14

Using a MySQL Server (Outdated)

This is a guide for windows only currently. Preliminary Linux (Debian) instructions below (VII) Preliminary Docker (modern Linux OS w/ Docker & git installed) instructions below (VIII)

14.1 I. Prerequisites

1. Have already ran/operated the RocketMap using the default database setup.
2. Have the "develop" build of RocketMap. Available here.
3. Downloaded MariaDB

14.2 II. Installing MariaDB

1. Run the install file, for me this was: mariadb-10.1.16-winx64.msi
2. Click next
3. Check "I accept the terms in the License Agreement" and click next
4. If you wish to change the installation location do that. If not click next.
5. Decide whether or not you want a password on your root account, if you don't put a password on it this database cannot be accessed from remote machines, which isn't necessary for what were doing. But if you do want a password go to 5a, if not go to 5b.
    - **5a.** Input the password you want into "new root password", and again into the "confirm" textbox.
    - **5b.** Uncheck "modify password for database user 'root'.
6. Click next
7. All the default options are acceptable here. Hit next.

8. This screen wants to know if you can submit anonymous usage data, feel free to hit next or if you'd like to contribute data go ahead and check "enable the Feedback plugin and submit anonymous usage information" and then click next.
9. Click install. Administrator privileges required.
10. Congratulations you've installed MariaDB and it's all downhill from here.

14.3 III. Setting up your database

1. Go to your windows start menu and locate MariaDB.
2. Open "MySQL Client"
3. If you created a password in step 5a above enter it now and hit enter. If you didn't create a password simply hit enter.
4. One this command prompt screen you'll want to enter:

```
CREATE DATABASE rocketmapdb;
CREATE USER 'rocketmapuser'@'localhost' IDENTIFIED BY 'password';
GRANT ALL PRIVILEGES ON rocketmapdb . * TO 'rocketmapuser'@'localhost';
exit
```

   You can change `rocketmapdb` to whatever you want the name of the database to be. Also, be sure to change `'password'` to the password you want to use.

5. If the database creation was successful it will tell you "Query OK, 1 row affected". If it doesn't echo that back at you then you either received an error message, or it just created a blank line. I've detailed how to fix common errors, and the blank line below.
   - **Blank line:** You simply missed the ";" in the CREATE DATABASE command. Essentially you didn't close of the line, so the program thinks you still have more information to input. Simply insert a ; onto the blank line and hit enter and it should echo "Query OK" at you.
   - **Error: "ERROR 1064 (42000): You have an error in your SQL syntax"** Double check that you put in the CREATE DATABASE command exactly as it's typed above. If you had the blank line error, and then retyped the CREATE DATABASE line it will spit this at you because you actually typed `CREATE DATABASE rocketmapdb CREATE DATABASE rocketmapdb;`. Simply retry the `CREATE DATABASE rocketmapdb;` and don't forget your `;`.
   - **Error: "ERROR 1007 (HY000): Can't create database 'rocketmapdb'; database exists"** The rocketmapdb database already exists. If you're trying to start a fresh database you'll need to execute `DROP DATABASE rocketmapdb`, and then run `CREATE DATABASE rocketgomapdb`. If you want to keep the rocketmapdb but start a new one, change the name.
   - **(1045, u"Access denied for user 'rocketmapuser'@'localhost' (using password: YES)")** You might be using **password** as your password for the database user **rocketmapuser**. Simply run `ALTER USER 'rocketmapuser'@'localhost' IDENTIFIED BY 'password';` and replace `'password'` with the password you want to use.
6. Congratulations, your database is now setup and ready to be used.

14.4 IV. Setting up the Config.ini file

14.4.1 Config.ini

1. Open file explorer to where you've extracted your develop branch of RocketMap
2. Navigate to the "config" folder.
3. Right-click and open config.ini in your text editor of choice. I used Notepad++.
4. You're looking to fill in all the values in this file. If you've already ran and used the RocketMap like was required in step 1 of the prerequisites you should be familiar with most of this information, but I've broken it all down below. On every line that you change/add a value make sure you remove the # as that makes the program think it is a comment, which obviously ignores the values you input.
    - **Authentication Settings:** You'll need to pick between using google, or pokemon trainer club login information. The RocketMap initial setup recommends ptc.
        - Change "auth-service" to ptc or google, whichever you choose.
        - Change "username" to your respective username for the selected service.
        - Change "password" to your respective password for the username on the selected service.
    - **Database Settings:** This is the important section you will want to modify.
        - Change the "db-type" to "mysql"
        - Change "db-host" to "127.0.0.1"
        - Change "db-name:" to "rocketmapdb"
        - Change "db-user:" to "rocketmapuser"
        - Change "db-pass" to the password you chose in section III step 4, or leave it blank if you chose to roll with no password.
    - **Search Settings:** You only need to change this if you want to only run one location, or wish to disable gyms/pokemon/pokestops for all locations, or to have a universal thread count, scan delay, or step limit. I chose to not edit anything in the new config.ini.
    - **Misc:** This only has one setting and that's the google maps api key. If you don't have one, or don't know what that is please see this wiki page for the RocketMap project.
        - Change "gmaps-key:" to contain your google maps API key.
    - **Webserver Settings:** This is how your server knows where to communicate.
        - Change "host" to the host address you should already have setup.
        - Change "port" is whatever port you are running the map through, default is 5000.
5. Make sure you've removed all of the # from any line with a value you inputted. Indent the comments that are after the values as well, so they are on the following line below the variable they represent. For example:

```
# Database settings
db-type: mysql
# sqlite (default) or mysql
```

6. Go to File->Save as... and make sure you save this file into the same directory as the "config.ini.example", but obviously save it as "config.ini". Make sure it's saved as a .ini file type, and not anything else or it won't work.
7. You're now done configuring your config.ini file.

14.5 V. Run it!

Now that we have our server setup and our config.ini filled out it's time to actually run the workers to make sure everything is in check. Remember from above if you commented out any parameters in the util.py file that all of those parameters need to be met and filled out when you run the runserver.py script. In our case we commented out location, and steps so we could individual choose where each worker scanned, and the size of the scan. I've put two code snippets below, one would be used if you didn't comment out anything and instead filled out the **[Search_Settings]** in section IV step 4 above. The other code snippet is what you would run if you commented out the same lines as I did in our running example.

```
python runserver.py
```

Left Search_Settings at default

```
python runserver.py -st 10 -l "[LOCATION]"
```

You should now be up and running. If you've encountered any errors it's most likely due to missing a parameter you commented out when you call runserver.py or you mis-typed something in your `config.ini`. However, if it's neither of those issues and something not covered in this guide hop into the RocketMap discord server, and go to the help channel. People there are great, and gladly assist people with troubleshooting issues.

14.6 Set up MySQL on a second computer, seperate from RM instances.

In this example, computer running RocketMap will be `Server A` while computer running MySQL will be `Server B`.

You will follow above directions for II and III on Server B and directions for IV on Server A. ***However:*** The following steps will be different.

Server B- III

You will need to grant your account permission to use the database outside of your database server.

```
  CREATE DATABASE rocketmapdb;
  CREATE USER 'rocketmapuser'@'%' IDENTIFIED BY 'password';
  GRANT ALL PRIVILEGES ON rocketmapdb . * TO 'rocketmapuser'@'%';
  exit
```

Server A- IV

You need to tell RocketMap where the database is!

```
**Database Settings:** This is the important section you will want to modify.
 - Change the "db-type" to "mysql"
 - Change "db-host" to [IP ADDDRESS OF SERVER B]
 - Change "db-name:" to "rocketmapdb"
 - Change "db-user:" to "rocketmapuser"
 - Change "db-pass" to the password you chose in section III step 4, or leave it␣
↪blank if you chose to roll with no password.
```

AND

```
 **Webserver Settings:** This is how your server knows where to communicate.
 - Change "host" to "0.0.0.0"
```

14.7 Linux Instructions

1. Visit https://downloads.mariadb.org/mariadb/repositories/ and download mariaDB
2. Login to your MySQL DB
   - mysql -p
   - Enter your password if you set one
3. Create the DB

```
CREATE DATABASE rocketmapdb;
CREATE USER 'rocketmapuser'@'localhost' IDENTIFIED BY 'password';
GRANT ALL PRIVILEGES ON rocketmapdb.* TO 'rocketmapuser'@'localhost';
```

4. Quit the MySQL command line tool `quit`
5. Edit the `config/config.ini` file

```
# Database settings
db-type: mysql          # sqlite (default) or mysql
db-host: 127.0.0.1      # required for mysql
db-name: rocketmapdb # required for mysql
db-user: rocketmapuser    # required for mysql
db-pass: YourPW         # required for mysql
```

14.8 Docker Settings w/ Let's Encrypt

Note: These are preliminary until better Docker support in the official container hosted on Docker Hub Note: These commands require git to be installed

```
docker build -t pokemap https://github.com/RocketMap/RocketMap.git:develop
docker run --name pokesql -e MYSQL_ROOT_PASSWORD=some-string -e MYSQL_DATABASE␣
↪pokemap -d mysql:5.6
```

only pain comes from mysql5.7 and beyond

```
docker run --name mainmap -d --link pokesql pokemap --auth-service=ptc \
  --username=youruser --password=yourpassword --db-type=mysql --db-host=pokesql \
  --db-name=pokemap --db-user=root --db-pass=some-string --gmaps-key=someapikey
```

all optional arguments except db type omitted, including –location (you can set it in the web ui!)

OPTIONAL: always scan Austin, TX (SQL benchmark?)

```
docker run --name scanagent -d --link pokesql pokemap --no-server \
 --auth-service=ptc --location="Austin, TX" --username=yourotheruser \
 --password=yourotherpassword --db-type=mysql --db-host=pokemap \
 --db-name=pokemap --db-user=root --db-pass=some-string \
 --gmaps-key=some-api-key
```

CHAPTER 15

Amazon ECS (Outdated)

Warning – Most cloud providers have been IP blocked from accessing the API

Amazon ECS is essentially managed docker allowed you to run multi-container environments easily with minimal configuration. In this guide we'll create an ECS Task that will run a single RocketMap container with a MariaDB container for persisting the data

15.1 Requirements

- AWS Account
- AWS ECS Cluster with at least one instance assigned
  - t2.micro type is sufficient for this setup

15.2 Process

In the AWS ECS console create a Task Definition with the JSON below. You will need to set the following values:

- `POGOM_USERNAME` - username for pokemongo
- `POGOM_PASSWORD` - password for pokemongo
- `POGOM_AUTH_SERVICE` - Define if you are using google or ptc auth
- `POGOM_LOCATION` - Location to search
- `POGOM_DB_USER` - Database user for MariaDB
- `POGOM_DB_PASS` - Database password for MariaDB

```
{
    "taskRoleArn": null,
    "containerDefinitions": [
```

(continues on next page)

(continued from previous page)

```
        {
            "volumesFrom": [],
            "memory": 128,
            "extraHosts": null,
            "dnsServers": null,
            "disableNetworking": null,
            "dnsSearchDomains": null,
            "portMappings": [
                {
                    "hostPort": 80,
                    "containerPort": 5000,
                    "protocol": "tcp"
                }
            ],
            "hostname": null,
            "essential": true,
            "entryPoint": null,
            "mountPoints": [],
            "name": "pokemongomap",
            "ulimits": null,
            "dockerSecurityOptions": null,
            "environment": [
                {
                    "name": "POGOM_DB_TYPE",
                    "value": "mysql"
                },
                {
                    "name": "POGOM_LOCATION",
                    "value": "Seattle, WA"
                },
                {
                    "name": "POGOM_DB_HOST",
                    "value": "database"
                },
                {
                    "name": "POGOM_NUM_THREADS",
                    "value": "1"
                },
                {
                    "name": "POGOM_DB_NAME",
                    "value": "pogom"
                },
                {
                    "name": "POGOM_PASSWORD",
                    "value": "MyPassword"
                },
                {
                    "name": "POGOM_GMAPS_KEY",
                    "value": "SUPERSECRET"
                },
                {
                    "name": "POGOM_AUTH_SERVICE",
                    "value": "ptc"
                },
                {
                    "name": "POGOM_DB_PASS",
                    "value": "somedbpassword"
```

(continues on next page)

(continued from previous page)

```
                },
                {
                    "name": "POGOM_DB_USER",
                    "value": "pogom"
                },
                {
                    "name": "POGOM_USERNAME",
                    "value": "MyUser"
                }
            ],
            "links": [
                "database"
            ],
            "workingDirectory": null,
            "readonlyRootFilesystem": null,
            "image": "frostthefox/rocketmap",
            "command": null,
            "user": null,
            "dockerLabels": null,
            "logConfiguration": null,
            "cpu": 1,
            "privileged": null
        },
        {
            "volumesFrom": [],
            "memory": 128,
            "extraHosts": null,
            "dnsServers": null,
            "disableNetworking": null,
            "dnsSearchDomains": null,
            "portMappings": [],
            "hostname": "database",
            "essential": true,
            "entryPoint": null,
            "mountPoints": [],
            "name": "database",
            "ulimits": null,
            "dockerSecurityOptions": null,
            "environment": [
                {
                    "name": "MYSQL_DATABASE",
                    "value": "pogom"
                },
                {
                    "name": "MYSQL_RANDOM_ROOT_PASSWORD",
                    "value": "yes"
                },
                {
                    "name": "MYSQL_PASSWORD",
                    "value": "somedbpassword"
                },
                {
                    "name": "MYSQL_USER",
                    "value": "pogom"
                }
            ],
            "links": null,
```

(continues on next page)

(continued from previous page)

```
            "workingDirectory": null,
            "readonlyRootFilesystem": null,
            "image": "mariadb:10.1.16",
            "command": null,
            "user": null,
            "dockerLabels": null,
            "logConfiguration": null,
            "cpu": 1,
            "privileged": null
        }
    ],
    "volumes": [],
    "family": "rocketmap"
}
```

If you would like to add workers you can easily do so by adding another container with the additional variable `POGOM_NO_SERVER` set to `true`. You have to let one of the RocketMap containers start first to create the database, an easy way to control this is to create a link from the worker to the primary one as it will delay the start.

Once the Task is running you'll be able to access the app via the Instances IP on port 80.

CHAPTER 16

Apache2 Reverse Proxy (Outdated)

If you do not want to expose RocketMap to the web directly or you want to place it under a prefix, follow this guide:

Assuming the following:

- You are running RocketMap on the default port 5000
- You've already made your machine available externally

1. Install Apache2 – plenty of tutorials on the web for this.
2. Enable the mods needed

```
sudo a2enmod proxy proxy_http proxy_connect ssl rewrite
```

3. Create a file /etc/apache2/sites-available/rocketmap.conf

```
sudo nano /etc/apache2/sites-available/rocketmap.conf`
```

copy pasta:

```
<VirtualHost *:80>

    ServerName rocketmap.yourdomain.com

    ProxyPass / http://127.0.0.1:5000/
    ProxyPassReverse / http://127.0.0.1:5000/

    RewriteCond %{HTTP_HOST} !^rocketmap\.yourdomain\.com$ [NC]
    RewriteRule ^/$ http://%{HTTP_HOST}/ [L,R=301]

    ErrorLog ${APACHE_LOG_DIR}/error.log
    CustomLog ${APACHE_LOG_DIR}/access.log combined

</VirtualHost>

<VirtualHost *:443>
```

(continues on next page)

(continued from previous page)

```
    ServerName rocketmap.yourdomain.com

    ProxyPass / http://127.0.0.1:5000/
    ProxyPassReverse / http://127.0.0.1:5000/

    RewriteCond %{HTTP_HOST} !^rocketmap\.yourdomain\.com$ [NC]
    RewriteRule ^/$ http://%{HTTP_HOST}/ [L,R=301]

    ErrorLog ${APACHE_LOG_DIR}/error.log
    CustomLog ${APACHE_LOG_DIR}/access.log combined

    SSLCertificateFile /var/www/ssl_keys/yourcert.crt
    SSLCertificateKeyFile /var/www/ssl_keys/yourkey.key
    SSLCertificateChainFile /var/www/ssl_keys/yourintermediatecert.crt

</VirtualHost>
```

If you want your maps at `rocketmap.yourdomain.com`, keep it just like it is If you want your maps at `yourdomain.com/go/` (note the trailing slash!)

```
(take out ServerName)
ProxyPass /go/ http://127.0.0.1:5000/
ProxyPassReverse /go/ http://127.0.0.1:5000/

RewriteCond %{HTTP_HOST} !^yourdomain\.com/go/$ [NC]
RewriteRule ^/go/$ http://%{HTTP_HOST}/go/ [L,R=301]
```

4. Test your Apache2 config: `sudo apachectl configtest`
5. Enable your new config: `sudo a2ensite rocketmap`
6. Reload your Apache2 service: `service apache2 reload`
7. You can now access it by going to: `http(s)://yourdomain.com/go` or `http(s)://rocketmap.yourdomain.com`

CHAPTER 17

Bluemix (Outdated)

Bluemix is IBM's PaaS, built on top of Cloud Foundry, and it's free tier allows you to have 24 up time! Oh boy!

17.1 Prerequisites

1. Clone the git repo via `https://github.com/RocketMap/RocketMap.git`
2. Create a Bluemix account
3. Install the Cloud Foundry CLI - Download here.

17.2 Create and Run the App on Bluemix

To do this, you can either use the GUI (click through, create a new python runtime and name it). Once it's created, you push the code by cding into the directory and running:

```
cf push <nameofapp>
```

To do the same thing via command line, you simply run that last command. It'll create the app and push the code for you.

Note that this first deploy will fail! We need to configure the environment variables for authentication to Pokemon Go and your Google API Key. To do that via the CLI:

```
cf set-env <nameofapp> AUTH_SERVICE <ptc|google>
cf set-env <nameofapp> USERNAME <username>
cf set-env <nameofapp> PASSWORD <password>
cf set-env <nameofapp> GMAPS_KEY <your google api key>
cf set-env <nameofapp> STEP_COUNT <step count>
cf set-env <nameofapp> LOCATION <the location you're spying on>
```

To set the environment variables via the GUI, you navigate to the "environment variables" tab in the app dashboard, click on "user defined," and enter them one by one.

Also make sure to paste your google api key in `config/credentials.json`.

Once the environment variables are set, and your credentials are set in `config/credentials.json`, re-push via `cf push <nameofapp>`.

17.3 An alternate way to set your credentials

Alternatively, instead of going the environment variable route, you can set up `config/config.ini`, and change the start command in `manifest.yml` to be

```
python runserver.py -se
```

which will pull the values from the config file instead of from the env vars.

CHAPTER 18

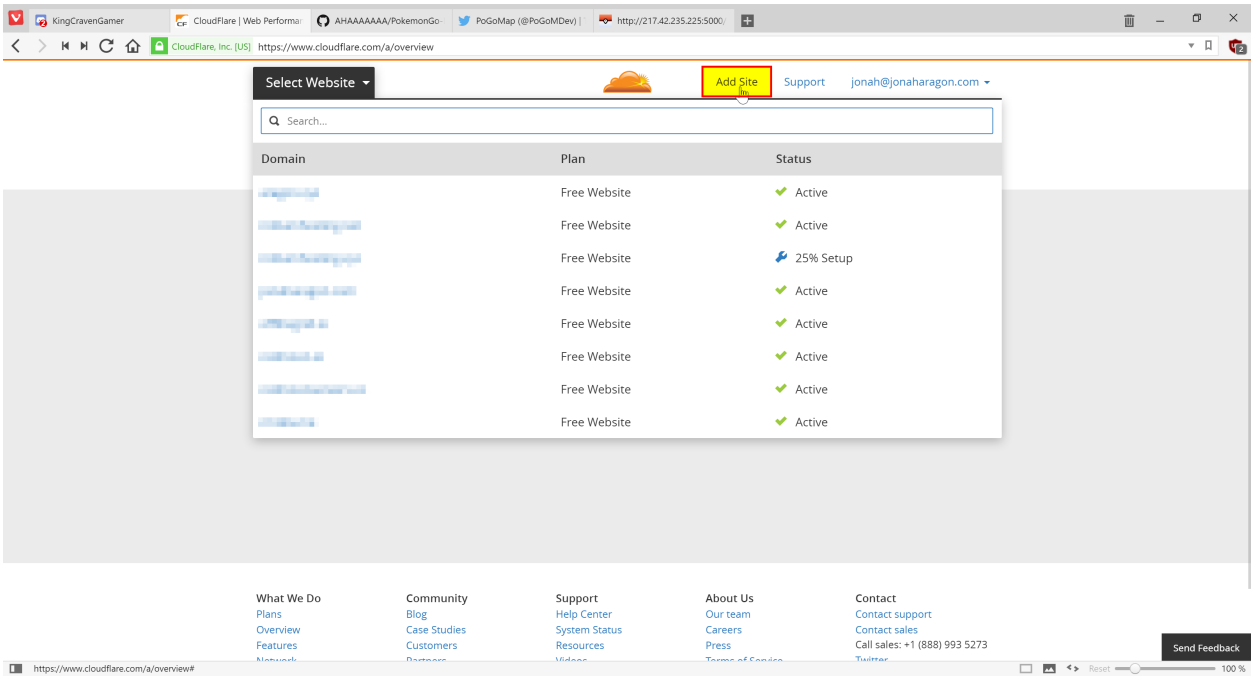
Cloudflare

18.1 Prerequisites

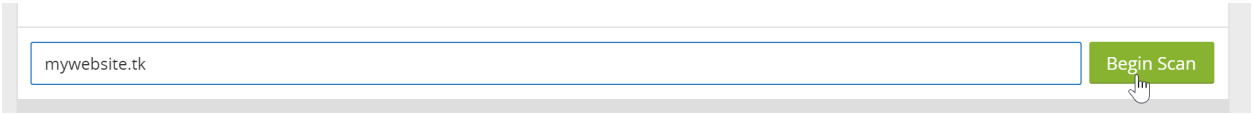
- A domain name (free ones can be had from places like Freenom)
- Your server port forwarded and **running on Port 80**

18.2 Getting Started

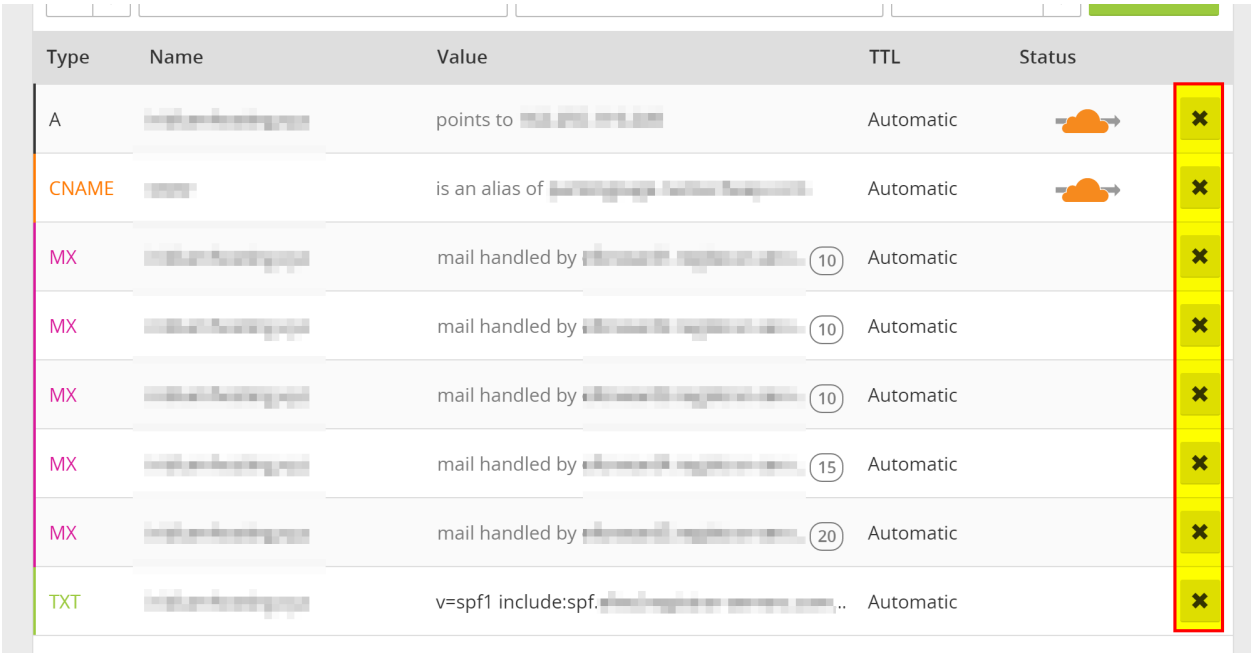
First, sign up for an account at cloudflare.com. Get signed in and click "Add Site" at the top of the page to get started.



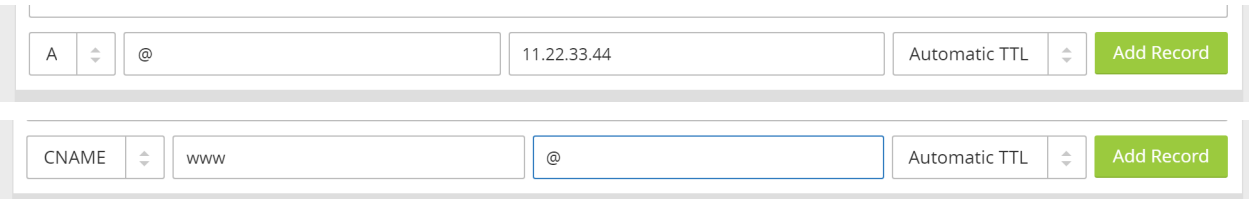
Enter your website in the text field and press "Begin Scan," when that finishes click Continue Setup next to your Domain.



If there is already information in this table, click the "X"s to clear them all out unless you already know what they are.



When the information is cleared, add two new records with the text boxes at the top with the following configuration (replace 11.22.33.44 with your server's IP address):



They should be added in the table automatically when you press "Add Record." **Make sure they have the Orange Cloud next to them** to stay secure. When this is done, press "Continue" at the bottom of the page. On the next page select the "Free Website" option if it isn't already preselected and "Continue" again.

Now it will give you two Nameservers on a page similar to this (although yours will most likely be different). Copy both the bold ones and put them in your domain settings at your registrar.

Current Nameservers	Change Nameservers to:
dns1.registrar-servers.com	**brad.ns.cloudflare.com**
dns2.registrar-servers.com	**pam.ns.cloudflare.com**

Instructions to change your nameservers differ between domain registrars, here's instructions for a few common ones:

- Namecheap
- GoDaddy
- Freenom

Once you have them set, navigate back to CloudFlare and press the Green Continue button once more, and you should be set! It may take **up to** 24 hours for it to switch to CloudFlare, but most of the time it happens much faster than that.

While we wait for it to switch, we should change some settings.

18.3 Settings to Change

Click the "Crypto" tab at the top and change SSL from Full to Flexible if it isn't already.

Navigate to "Firewall" and change "Security Level" to "High." If you are running this from your house consider setting it to "I'm under attack" for the highest amount of DDoS security against your server.

18.4 Finishing Up

Everything should now be configured correctly! Simply visit `http://yourdomain.com/` in your browser and your map should load, all while hiding your IP address from everybody else.

CHAPTER 19

DigitalOcean (Outdated)

Warning – Most cloud providers have been IP blocked from accessing the API

19.1 Prerequisites

- A DigitalOcean account - Using this link will grant $10 in credit, enough for running your server for up to 2 months.
- A Google Maps API key
- A new Pokemon Club account

19.2 Installation

Create a Droplet in your DigitalOcean control panel with Ubuntu 16.04, any Droplet size will work.

Check the "User Data" box lower on the page and enter the following:

```
#!/bin/bash

apt-get -y update
apt-get -y install python python-pip git
git clone https://github.com/RocketMap/RocketMap.git /root/PoGoMap
cd /root/PoGoMap
pip install -r requirements.txt
python runserver.py -u [USERNAME] -p [PASSWORD] -st 10 -k [Google Maps API key] -l
↪"[LOCATION]" -H 0.0.0.0 -P 80
```

Important: Be sure to replace `[USERNAME]`, `[PASSWORD]`, [API Key], `and` [LOCATION]' with your Pokemon Trainers Club Username and Password, your Google Maps API Key, and your location (Latitude and Longitude), respectively. You will be able to change locations later on the site.

Once you have that, create your Droplet. Setup will take a few minutes initially, but once it's done your map will be accessible at `http://[YOURDROPLET]/`, replacing that of course with your Droplet's IP address.

19.3 Starting the server

On the first boot the server will start automatically so this step isn't necessary, however if you have to restart the Droplet for any reason, you can start PoGoMap with the following two commands:

```
cd /root/PoGoMap
python runserver.py -u [USERNAME] -p [PASSWORD] -st 10 -k [Google Maps API key] -l
↪"[LOCATION]" -H 0.0.0.0 -P 80
```

Credit: JonahAragon

CHAPTER 20

Docker (Outdated)

Docker is a great way to run "containerized" applications easily and without installing tons of stuff into your computer.

If you are not familiar or don't feel confortable with Python, pip or any of the other the other stuff involved in launching a RocketMap server, Docker is probably the easiest approach for you.

20.1 Prerequisites

- Docker
- Google Maps API Key

20.2 Introduction

The quickest way to get RocketMap up and running with docker is quite simple. However you need to setup an external mysql database to make it work so be sure to read the tutorial until the "Advanced Docker Setup"

20.3 Simple Docker Setup

20.3.1 Starting the server

In order to start the map, you've got to run your docker container with a few arguments, such as authentication type, account, password, desired location and steps. If you don't know which arguments are necessary, you can use the following command to get help:

```
docker run --rm frostthefox/rocketmap -h
```

To be able to access the map in your machine via browser, you've got to bind a port on your host machine to the one wich will be exposed by the container (default is 5000). The following docker run command is an example of to launch a container with a very basic setup of the map, following the instructions above:

```
docker run -d --name pogomap -p 5000:5000 \
  frostthefox/rocketmap \
    -a ptc -u username -p password \
    -k 'your-google-maps-key' \
    -l 'lat, lon' \
    -st 5
```

If you would like to see what are the server's outputs (console logs), you can run:

```
docker logs -f pogomap
```

Press `ctrl-c` when you're done.

20.3.2 Stopping the server

In the step above we launched our server in a container named `pogomap`. Therefore, to stop it as simple as running:

```
docker stop pogomap
```

After stopping a named container, if you would like to launch a new one re-using such name, you have to remove it first, or else it will not be allowed:

```
docker rm pogomap
```

20.3.3 Local access

Given that we have bound port 5000 in your machine to port 5000 in the container, which the server is listening to, in order to access the server from your machine you just got to access `http://localhost:5000` in your preferred browser.

20.3.4 External access

If external access is necessary, there are plenty of ways to expose you server to the web. In this guide we are going to approach this using a ngrok container, which will create a secure introspected tunnel to your server. This is also very simple to do with Docker. Simply run the following command:

```
docker run -d --name ngrok --link pogomap \
  wernight/ngrok \
    ngrok http pogomap:5000
```

After the ngrok container is launched, we need to discover what domain you've been assigned. The following command can be used to obtain the domain:

```
docker run --rm --link ngrok \
  appropriate/curl \
    sh -c "curl -s http://ngrok:4040/api/tunnels | grep -o 'https\?:\/\/[a-zA-Z0-9\.
↪]\+'"
```

That should output something like:

```
http://random-string-here.ngrok.io
https://random-string-here.ngrok.io
```

Open that URL in your browser and you're ready to rock!

20.4 Updating Versions

In order to update your RocketMap docker image, you should stop/remove all the containers running with the current (outdated) version (refer to "Stopping the server"), pull the latest docker image version, and restart everything. To pull the latest image, use the following command:

```
docker pull frostthefox/rocketmap
```

If you are running a ngrok container, you've got to stop it as well. To start the server after updating your image, simply use the same commands that were used before, and the containers will be launched with the latest version.

20.5 Running on docker cloud

If you want to run RocketMap on a service that doesn't support arguments like docker cloud or ECS, you'll need to pass settings via variables below is an example:

```
  docker run -d -P \
    -e "POGOM_AUTH_SERVICE=ptc" \
    -e "POGOM_USERNAME=UserName" \
    -e "POGOM_PASSWORD=Password" \
    -e "POGOM_LOCATION=Seattle, WA" \
    -e "POGOM_GMAPS_KEY=SUPERSECRET" \
    frostthefox/rocketmap
```

20.6 Advanced Docker Setup

In this session, we are going to approach a docker setup that allows data persistence. To do so, we are going to use the docker image for MySQL as our database, and have our server(s) connect to it. This could be done by linking docker containers. However, linking is considered a legacy feature, so we are going to use the docker network approach. We are also going to refer to a few commands that were used in the "Simple Docker Setup" session, which has more in-depth explanation about such commands, in case you need those.

20.6.1 Creating the Docker Network

The first step is very simple, we are going to use the following command to create a docker network called `pogonw`:

```
docker network create pogonw
```

20.6.2 Launching the database

Now that we have the network, we've gotta launch the database into it. As noted in the introduction, docker containers are disposable. Sharing a directory in you machine with the docker container will allow the MySQL server to use

such directory to store its data, which ensures the data will remain there after the container stops. You can create this directory wherever you like. In this example we going to create a dir called `/path/to/mysql/` just for the sake of it.

```
mkdir /path/to/mysql/
```

After the directory is created, we can lauch the MySQL container. Use the following command to launch a container named `db` into our previously created network, sharing the directory we just created:

```
docker run --name db --net=pogonw -v /path/to/mysql/:/var/lib/mysql -e MYSQL_ROOT_
↪PASSWORD=yourpassword  -d mysql:5.6.32
```

The launched MySQL server will have a single user called `root` and its password will be `yourpassword`. However, there is no database/schema that we can use as the server will be empty on the first run, so we've gotta create one for RocketMap. This will be done by executing a MySQL command in the server. In order to connect to the server, execute this command:

```
docker exec -i db mysql -pyourpassword -e 'CREATE DATABASE pogodb'
```

That will do the trick. If you want to make sure the database was created, execute the following command and check if `pogodb` is listed:

```
docker exec -i db mysql -pyourpassword -e 'SHOW DATABASES'
```

20.6.3 Relaunching the database

If the `db` container is not running, simply execute the same command that was used before to launch the container and the MySQL server will be up and running with all the previously stored data. You won't have to execute any MySQL command to create the database.

20.6.4 Launching the RocketMap server

Now that we have a persistent database up and running, we need to launch our RocketMap server. To do so, we are going to use a slightly modified version of the docker run command from the "Simple Docker Setup" session. This time we need to launch our server inside the created network and pass the necessary database infos to it. Here's an example:

```
docker run -d --name pogomap --net=pogonw -p 5000:5000 \
  frostthefox/rocketmap \
    -a ptc -u username -p password \
    -k 'your-google-maps-key' \
    -l 'lat, lon' \
    -st 5 \
    --db-host db \
    --db-port 3306 \
    --db-name pogodb \
    --db-user root \
    --db-pass yourpassword
```

This will launch a container named `pogomap`. Just like before, in order to check the server's logs we can use:

```
docker logs -f pogomap
```

If you want more detailed logs, add the `--verbose` flag to the end of the docker run command.

If everything is fine, the server should be up and running now.

20.6.5 Launching workers

If you would like to launch a different worker sharing the same db, to scan a different area for example, it is just as easy. We can use the docker run command from above, changing the container's name, and the necessary account and coordinate infos. For example:

```
docker run -d --name pogomap2 --net=pogonw \
  frostthefox/rocketmap \
    -a ptc -u username2 -p password2 \
    -k 'your-google-maps-key' \
    -l 'newlat, newlon' \
    -st 5 \
    --db-host db \
    --db-port 3306 \
    --db-name pogodb \
    --db-user root \
    --db-pass yourpassword
    -ns
```

The difference here being: we are launching with the `-ns` flag, which means that this container will only run the searcher and not the webserver (front-end), because we can use the webserver from the first container. That also means we can get rid of `-p 5000:5000`, as we dont need to bind that port anymore.

If for some reason you would like this container to launch the webserver as well, simply remove the `-ns` flag and add back the `-p`, with a different pairing as your local port 5000 will be already taken, such as `-p 5001:5000`.

20.6.6 External Access

Just like before, we can use ngrok to provide external access to the webserver. The only thing we need to change in the command from the previous session is the `--link` flag, instead we need to launch ngrok in our network:

```
docker run -d --name ngrok --net=pogonw \
  wernight/ngrok \
    ngrok http pogomap:5000
```

To obtain the assigned domain from ngrok, we also need to execute the previous command in our network instead of using links:

```
docker run --rm --net=pogonw \
  appropriate/curl \
    sh -c "curl -s http://ngrok:4040/api/tunnels | grep -o 'https\?:\/\/[a-zA-Z0-9\.
↪]\+'"
```

20.6.7 Inspecting the containers

If at any moment you would like to check what containers are running, you can execute:

```
docker ps -a
```

If you would like more detailed information about the network, such as its subnet and gateway or even the ips that were assigned to each running container, you can execute:

```
docker network inspect pogonw
```

20.6.8 Setting up notifications

If you have a docker image for a notification webhook that you want to be called by the server/workers, such as PokeAlarm, you can launch such container in the 'pogonw' network and give it a name such as 'hook'. This guide won't cover how to do that, but once such container is configured and running, you can stop your server/workers and relaunch them with the added flags: `-wh`, `--wh-threads` and `--webhook-updates-only`. For example, if the hook was listening to port 4000, and we wanted 3 threads to post updates only to the hook:

```
docker run -d --name pogomap --net=pogonw -p 5000:5000 \
  frostthefox/rocketmap \
    -a ptc -u username -p password \
    -k 'your-google-maps-key' \
    -l 'lat, lon' \
    -st 5 \
    --db-host db \
    --db-port 3306 \
    --db-name pogodb \
    --db-user root \
    --db-pass yourpassword \
    -wh 'http://hook:4000' \
    --wh-threads 3 \
    --webhook-updates-only
```

CHAPTER 21

Nginx

If you do not want to expose RocketMAD to the web directly or you want to place it under a prefix, follow this guide:

Assuming the following:

- You are running RocketMAD on the default port 5000
- You've already made your machine available externally (for example, port forwarding)

1. Install nginx (I'm not walking you through that, google will assist) - http://nginx.org/en/linux_packages.html
2. In /etc/nginx/nginx.conf make sure the following is added:

```
include conf.d/*.conf;
```

3. Create a file /etc/nginx/conf.d/rocketmad.conf and place the following in it:

```
upstream app_server {
    # fail_timeout=0 means we always retry an upstream even if it failed
    # to return a good HTTP response.
    server 127.0.0.1:5000 fail_timeout=0;
}

server {
    listen 80;
    listen [::]:80;

    # Set the correct host(s) for your site.
    server_name my_domain.com www.my_domain.com;

    # Path to your RocketMAD folder.
    root /path/to/RM;

    location ~ /(config|geofences|logs) {
        return 403;
    }
```

(continues on next page)

(continued from previous page)

```
    location / {
        # Checks for static file, if not found proxy to app.
        try_files $uri @proxy_to_app;
    }

    location @proxy_to_app {
        proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
        proxy_set_header X-Forwarded-Proto $scheme;
        proxy_set_header Host $http_host;
        # We don't want nginx trying to do something clever with
        # redirects, we set the Host: header above already.
        proxy_redirect off;
        proxy_pass http://app_server;
    }
}
```

In case you want to use a subdirectory, e.g. http://my_domain.com/map, add the following instead of the above:

```
upstream app_server {
    # fail_timeout=0 means we always retry an upstream even if it failed
    # to return a good HTTP response.
    server 127.0.0.1:5000 fail_timeout=0;
}

server {
    listen 80;
    listen [::]:80;

    # Set the correct host(s) for your site.
    server_name my_domain.com www.my_domain.com;

    # Path to your RocketMAD folder.
    root /path/to/RM;

    location ~ /map/(config|geofences|logs) {
        return 403;
    }

    location /map {
        # Checks for static file, if not found proxy to app.
        try_files $uri @proxy_to_app;
    }

    location @proxy_to_app {
        proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
        proxy_set_header X-Forwarded-Proto $scheme;
        proxy_set_header Host $http_host;
        proxy_set_header SCRIPT_NAME /map;
        # We don't want nginx trying to do something clever with
        # redirects, we set the Host: header above already.
        proxy_redirect off;
        proxy_pass http://app_server;
    }
}
```

21.1 Add a free SSL certificate to your site:

1. Install certbot: https://certbot.eff.org/instructions
2. Run `certbot certonly -d my_domain.com www.my_domain.com` to generate certificates
3. Update your rocketmap.conf file, see example below
4. Certificates last for 3 months and can be renewed by running `certbot renew`

21.2 SSL example config

```
upstream app_server {
    # fail_timeout=0 means we always retry an upstream even if it failed
    # to return a good HTTP response.
    server 127.0.0.1:5000 fail_timeout=0;
}

server {
    listen 80;
    listen [::]:80;

    # Set the correct host(s) for your site.
    server_name my_domain.com www.my_domain.com;

    return 301 https://my_domain.com$request_uri;
}

server {
    listen 443 ssl http2;
    listen [::]:443 ssl http2;

    # Set the correct host(s) for your site.
    server_name my_domain.com www.my_domain.com;

    # Path to your RocketMAD folder.
    root /path/to/RM;

    ssl_certificate /etc/letsencrypt/live/my_domain.com/fullchain.pem;
    ssl_certificate_key /etc/letsencrypt/live/my_domain.com/privkey.pem;
    ssl_session_timeout 1d;
    ssl_session_cache shared:SSL:10m;
    ssl_session_tickets off;

    ssl_protocols TLSv1.2 TLSv1.3;
    ssl_ciphers ECDHE-ECDSA-AES128-GCM-SHA256:ECDHE-RSA-AES128-GCM-SHA256:ECDHE-ECDSA-
↪AES256-GCM-SHA384:ECDHE-RSA-AES256-GCM-SHA384:ECDHE-ECDSA-CHACHA20-POLY1305:ECDHE-
↪RSA-CHACHA20-POLY1305:DHE-RSA-AES128-GCM-SHA256:DHE-RSA-AES256-GCM-SHA384;
    ssl_prefer_server_ciphers off;

    add_header Strict-Transport-Security "max-age=63072000" always;

    location / {
        # Checks for static file, if not found proxy to app.
        try_files $uri @proxy_to_app;
    }
```

(continues on next page)

(continued from previous page)

```
    location @proxy_to_app {
        proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
        proxy_set_header X-Forwarded-Proto $scheme;
        proxy_set_header Host $http_host;
        # We don't want nginx trying to do something clever with
        # redirects, we set the Host: header above already.
        proxy_redirect off;
        proxy_pass http://app_server;
    }
}
```

Please be sure to change the ssl_certificate and ssl_certificate_key paths to point to your cert file and key.

21.3 Adding simple httpd Authentication. (Outdated)

This will guide you through setting up simple HTTP Authentication using nginx and reverse proxy protocols. These instructions are written for someone using a Debian/Ubuntu VPS. Your enviroment may have slightly different requirements, however the concepts as a whole should still stand. This guide assumes you have nginx installed and running, and a `conf.d/*.conf` file created, such as `/etc/nginx/conf.d/rocketmap.conf`, as the example above provides, and that you're running your service on port 5000, and want it to be accessable at http://your_ip/go/, although it supports other ports and locations.

`*` denotes a wildcard, and will be used to stand for your site's `*.conf` file, please **do not** literally type `sudo nano /etc/nginx/conf.d/*.conf`.

1. Create a .htpasswd file inside `/etc/nginx/`. Some suggested methods to create a .htpasswd file are below.
   - Linux users can use the apache2-tools package to create the files.-First, get the apache2-utils package

     ```
     sudo apt-get install apache2-utils
     ```

     -Then run the htpasswd command

     ```
     sudo htpasswd -c /etc/nginx/.htpasswd exampleuser
     ```

     This will prompt you for a new password for user exampleuser. Remove the `-c` tag for additional entries to the file. Opening the file with a text exitor such as nano should show one line for each user, with an encrypted password following, in the format of user:pass.
   - Manual generation of the file can be done using tools such as: http://www.htaccesstools.com/htpasswd-generator/. After manually generating the file, please place it in `/etc/nginx/`, or wherever your distro installs `nginx.conf` and the rest of your config files.
2. Open your `*.conf` file with a text editor, with a command such as `sudo nano /etc/nginx/conf.d/rocketmap.conf`. Add the following two lines underneath the domain path.

   ```
   auth_basic "Restricted";
   auth_basic_user_file /etc/nginx/.htpasswd;
   ```

   If your `*.conf` file matches the example provided above, you should have the following.

   ```
   server {
       listen 80;
   ```

(continues on next page)

(continued from previous page)

```
    location /go/ {
       auth_basic "Restricted";
       auth_basic_user_file /etc/nginx/.htpasswd;
       proxy_pass http://127.0.0.1:5000/;
    }
}
```

Now, we're going to go ahead and fill out the `*.conf` file with the rest of the information to make our service work, and shore up our nginx config, by appending the following between the authentication block and proxy_pass.

```
proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
proxy_set_header X-Forwarded-Proto http;
proxy_set_header Host $http_host;
proxy_redirect off;
```

Here is a fully completed example `*.conf`, with working httpd authentication. Notice, this example does not use SSL / 443, although the method can be adapted to it!

```
upstream pokemonmap {
   server 127.0.0.1:5000 fail_timeout=0
}
server {
   listen 80;
   server_name [sub.domain.com] [your_ip];

   location /go/ {
      auth_basic "Restricted";
      auth_basic_user_file /etc/nginx/.htpasswd;
      proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
      proxy_set_header X-Forwarded-Proto http;
      proxy_set_header Host $http_host;
      proxy_redirect off;
      proxy_pass http://[your_ip]:5000;
      break;
   }
}
```

3. Test your nginx configuration using the command `sudo nginx -t`. If this returns a positive result, restart the nginx service using `sudo service nginx restart`.
4. Verify your configuration is working by loading http://your_ip/go/ or http://sub.your.domain/go/, or however else you have it set in your `*.conf`. Please verify it's working before proceeding to step 5, or it will be much harder to troubleshoot!

   Troubleshooting:

   - **I can't reach my server at http://your_ip/go/!**

     Check http://your_ip:5000/. If you cannot see it there, your issue lies with your server, not with nginx! If you can see it there, but cannot see it at http://your_ip/go/, your issue lies with nginx. Please check your config files to make sure they are pointing at the right places, and that your `sudo nginx -t` checks out.

   - **nginx -t doesn't check out.**

     Check the error messages for which line is creating the error, and work your way backwards from there. Many times it's just a missed `;` or `}`.

5. Finally, we're going to modify our runserver.py command to operate with the `-H 127.0.0.1` flag, only allowing our webapp to be accessable from Localhost. As nginx is running on the local system, nginx will still be able to fetch the webapp, and serve it, through the proxy and authentication, to remote users, but remote users will not be able to connect directly to the webapp itself. If your runserver command is

   ```
   python runserver.py -u user -p pass -k key -l "my,coords" -st 10 -P 5000
   ```

   You are going to want to update it to the following:

   ```
   python runserver.py -u user -p pass -k key -l "my,coords" -st 10 -P 5000
   -H 127.0.0.1
   ```

   From there, we're going to want to check and see that you can get to your server, albeit through authentication, at http://your_ip/go/, and that you cannot get to your server at http://your_ip/go:5000/. If that works, you're all set up!

CHAPTER 22

Supervisord on Linux (Outdated)

22.1 Assuming:

- You are running on Linux
- You have installed supervisord
- You have seen a shell prompt at least a few times in your life
- You have configured your stuff properly in `config.ini`
- You understand worker separation
- You can tie your own shoelaces

22.2 The good stuff

cd into your root RocketMap folder. Then:

```
cd contrib/supervisord/
./install-reinstall.sh
```

When this completes, you will have all the required files. (this copies itself and the required files so that there is no conflict when doing a `git pull`. Now we are going to edit your local copy of gen-workers.sh:

```
cd ~/supervisor
nano gen-workers.sh
```

In this file, change the variables needed to suit your situation. Below is a snippet of the variables:

```
# Name of coords file. SEE coords-only.sh if you dont have one!
coords="coords.txt"
```

(continues on next page)

(continued from previous page)

```
# Webserver Location
initloc="Dallas, TX"
# Account name without numbers
pre="accountname"

# Variables
hexnum=1    # This is the beehive number you are creating. If its the first, or you
↪want to overwrite, dont change
worker=0    # This is the worker number. Generally 0 unless 2+ Hives
acct1=0     # The beginning account number for the hive is this +1
numacct=5   # This is how many accounts you want per worker
pass="yourpasshere" # The password you used for all the accounts
auth="ptc" # The auth you use for all the accounts
st=5        # Step Count per worker
sd=5        # Scan Delay per account
ld=1        # Login Delay per account
directory='/path/to/your/runserver/directory/' # Path to the folder containing
↪runserver.py **NOTICE THE TRAILING /**
```

As you saw above you will need to create a coords.txt (or whatever you decide to name it. I personally use city.stepcount.coords as my naming convention). We are going to use location_generator.py:

```
cd (your RocketMap main folder here)
python Tools/Hex-Beehive-Generator/location_generator.py -lat "yourlat" -lon "yourlon
↪" -st 5 -lp 4 -or "~/supervisor/coords.txt"
```

Now run the gen-workers.sh script

```
cd ~/supervisor
./gen-workers.sh
```

You should now have a bunch of .ini files in `~/supervisor/hex1/`

You can now do:

```
supervisord -c ~/supervisor/supervisord.conf
```

And you will have a working controllable hive! You should be able to see from the web as well at `http://localhost:5001` Read up on the supervisord link at the top if you want to understand more about supervisorctl and how to control from the web.