Eureka Labs

Developer's Manual

Revision History

Revision history of this document

Version	Changes	Date
1.0	Initial Draft	4/20/2019
1.0	Final	5/6/2019

Table of Contents

1. Introduction

1.1 Purpose

The purpose of this document is to give direction for environment set up and specific information for re-installing and running the application, to develop new features, or to update current features.

1.2 Project Overview

Copy from other document

1.3 Document Overview

Section 1 - Introduction
Section 2 - Server Setup
Section 3 - Web Server Setup
Section 4 - Database Server Setup
Section 5 - Firewall Information
Section 6 - Setup for development
Section 7 - Glossary of Terms

1.4 Instructions

All commands to use will be *italicized in Courier New font.*
*All step shown are in an Ubuntu 18.04 operating system.

2. Server Setup

2.1 Description

The current environment uses two Ubuntu 18.04 LTS Linux servers. One of the servers is the web application server and the other is the database server. The application and database can be run on any version of Linux that can run NGINX web server and MongoDB database server.

2.2 Server Installation

In both installations used throughout this project, the server was a Virtual Machine (VM) and installation was already completed. Steps to follow upon first accessing the servers are listed in this section.

2.2.3 Server Updates

Steps:

- Update repositories
    - *sudo yum -y update*
- Update applications that need updating
    - *sudo yum -y upgrade*
- Turn off firewall
    - Our servers are behind external firewalls therefore we can turn off the local firewall
    - *sudo systemctl stop ufw*
    - *sudo systemctl disable ufw*
- Reboot the system
    - *sudo reboot*

Your new server will be up to date.

3. Web Server Setup

3.1 Web Server Introduction

The web server hosts the application and the web server. It also is the mechanism used to perform database backups. The web server will need several components to be able to run the application.

- Python 3
- Pip 3
- NGINX Web Server
- MongoDB Tools
- MongoDB Shell - optional
- Certbot

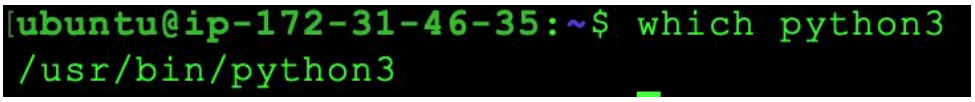
3.1.1 Python 3 Installation

Python is the programming language used to develop the application.

Steps:

- Install Python PPA
  - *sudo add-apt-repository ppa:deadsnakes/ppa*
- Update repositories
  - *sudo yum -y update*
- Install Python
  - *sudo yum -y install python3.**
- Verify which Python is in $PATH (installed)
  - *which python3*

Example



3.1.2 Pip 3 Installation

Pip is a package installation program for Python packages. This is how the application installs Python packages that are needed to run.

Steps:

- Install pip
  - *sudo apt -y install python3-pip*
- Verify installation
  - *pip3 --version*

Example

```
ubuntu@ip-172-31-46-35:~$ pip3 --version
pip 9.0.1 from /usr/lib/python3/dist-packages (python 3.6)
```

3.1.3 NGINX Installation

NGINX is the web server used for the application.

Steps:

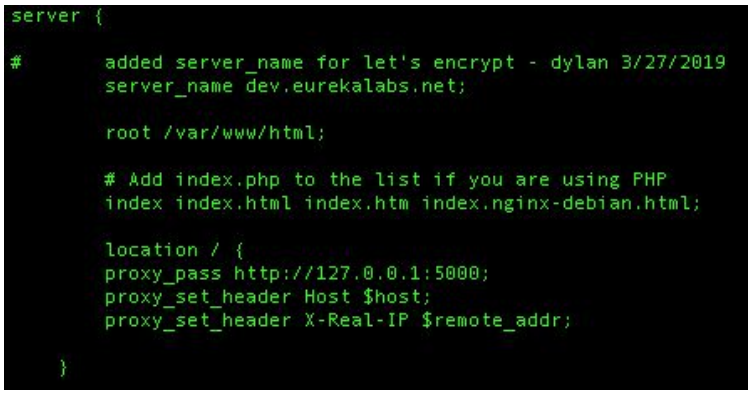
- Install NGINX
  - *sudo apt −y install nginx*

Configuration:

- Navigate to the `/etc/nginx` directory.
- In the nginx.conf file, we need to enable site files.
  - *sudo vim nginx.conf*
- Once in the configuration file, scroll down to the bottom of the http object until the two include lines are visible.
- If there is pound sign (#) in front of `include /etc/nginx/sites-enabled/*;`, remove it.
  Example:

```
#       include /etc/nginx/conf.d/*.conf;
        include /etc/nginx/sites-enabled/*;
}
```

- Save the file and exit.
- Change directories to the `sites-available` directory
  - `cd sites-available`
- Create a server file for the application
  - `cp default eurekalabs`
  - `vim eurekalabs`
- Change the server block.
  - The server name is important for the Let's Encrypt certificate that will be installed.
  - The location sub-block allows the Flask application to be routed to the web server.



- Save and exit the file.
- Change directories to the *sites-enabled* directory
  - *cd ../sites-enabled*
- Create a symbolic link to the *eurekalabs* file.
  - It is best to use the full path when making a symbolic link.
  - *ln -s /etc/nginx/sites-available/eurekalabs*
- Set NGINX to start when the server is booted.
  - *sudo systemctl enable nginx*
- Restart NGINX.
  - *sudo systemctl restart nginx*

3.1.4 MongoDB Tools Installation

MongoDB Tools will allow you to execute database backups from the web server. In the current production set up, the database server only talks to the web server. If you are storing your database backups off site, you will need a way to get them off of the database server.

Steps:

- Install the MongoDB certificate to Apt.
  - Option 1 - download the certificate from *https://www.mongodb.org/static/pgp/server-4.0.asc*
    - *curl "https://www.mongodb.org/static/pgp/server-4.0.asc" >> mongokey.asc*
    - *sudo apt-key add mongokey.asc*
  - Option 2 - get the key directly from the key server.
    - *sudo apt-key adv --keyserver hkp://keyserver.ubuntu.com:80 --recv 9DA31620334BD75D9DCB49F368818C72E52529D4*
- Add the MongoDB repository to Apt.
  - *echo "deb [arch=amd64] https://repo.mongodb.org/apt/ubuntu bionic/mongodb-org/4.0 multiverse" | sudo tee /etc/apt/sources.list.d/mongodb-org-4.0.list*

- Update repositories.
    - *sudo apt -y update*
- Install MongoDB Tools.
    - *sudo apt -y install mongodb-org-tools*

3.1.5 Install MongoDB Shell (Optional)

The MongoDB shell will allow you to connect to the database to verify or change information. This does not affect the application. It is just a tool to use.

Steps:

- Complete MongoDB tools installation.
- Install MongoDB shell
    - *sudo apt -y install mongodb-org-shell*

3.1.6 Certbot Installation

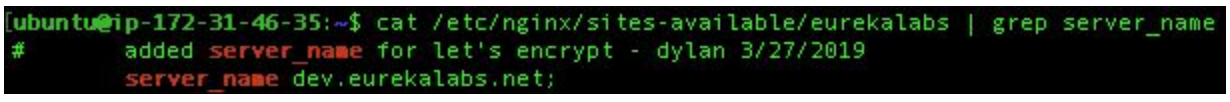
Let's Encrypt is an open source certificate authority. Certbot is a Python application that will use NGINX to verify the server's identity with Let's Encrypt and install a TLS certificate. Let's Encrypt certificates are only good for 90 days and will need to updated on a regular basis. The Certbot installation includes an auto-update function for the certificate.

Requirements:

- A DNS record needs to point to the server.

Steps:

- Install the Cerbot repository.
    - `sudo add-apt-repository ppa:certbot/certbot`
- Update repositories.
    - *sudo apt -y update*
- Install Certbot.
    - *sudo apt -y install python-cerbot-nginx*
- Verify that your NGINX server file has a server name set up.
    - *cat /etc/nginx/sites-enabled/eurekalabs | grep server_name*
    - The server name will be the domain requested from Let's Encrypt



- Run Certbot
    - All domain names that are being requested need to be in the *server_name* line of the NGINX server file. Add another *-d* and the domain name to the command for each domain.
    - *sudo certbot --nginx -d [domain name]*
- If this is your first time running Certbot, you will need to enter an email address.

```
ubuntu@ip-172-31-46-35:/etc/nginx/sites-available$ sudo certbot --nginx -d dev.eurekalabs.net
Saving debug log to /var/log/letsencrypt/letsencrypt.log
Plugins selected: Authenticator nginx, Installer nginx
Obtaining a new certificate
Performing the following challenges:
http-01 challenge for dev.eurekalabs.net
Using default addresses 80 and [::]:80 ipv6only=on for authentication.
Waiting for verification...
Cleaning up challenges
Deploying Certificate to VirtualHost /etc/nginx/sites-enabled/eurekalabs
```

- Select “Redirect”.
    - This will redirect all HTTP traffic to HTTPS. This will also configure your NGINX setup to do this.

```
Please choose whether or not to redirect HTTP traffic to HTTPS, removing HTTP access.
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
1: No redirect - Make no further changes to the webserver configuration.
2: Redirect - Make all requests redirect to secure HTTPS access. Choose this for
new sites, or if you're confident your site works on HTTPS. You can undo this
change by editing your web server's configuration.
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
Select the appropriate number [1-2] then [enter] (press 'c' to cancel): 2
```

- You are now finished with the certificate installation.

```
Redirecting all traffic on port 80 to ssl in /etc/nginx/sites-enabled/eurekalabs

- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
Congratulations! You have successfully enabled https://www.eurekalabs.net

You should test your configuration at:
https://www.ssllabs.com/ssltest/analyze.html?d=www.eurekalabs.net
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -

IMPORTANT NOTES:
 - Congratulations! Your certificate and chain have been saved at:
   /etc/letsencrypt/live/www.eurekalabs.net/fullchain.pem
   Your key file has been saved at:
   /etc/letsencrypt/live/www.eurekalabs.net/privkey.pem
   Your cert will expire on 2019-06-26. To obtain a new or tweaked
   version of this certificate in the future, simply run certbot again
   with the "certonly" option. To non-interactively renew *all* of
   your certificates, run "certbot renew"
 - If you like Certbot, please consider supporting our work by:

   Donating to ISRG / Let's Encrypt:   https://letsencrypt.org/donate
   Donating to EFF:                    https://eff.org/donate-le
```

- Restart NGINX.
    - *sudo systemctl restart nginx*

4. Database Server Setup

4.1 Database Introduction

MongoDB is our database. We chode MongoDB because it is easy to use, flexible, and can handle large amounts of data. Over the time of the application development, we discovered that we are not using MongoDB to its fullest potential. This installation is the simplest form of MongoDB.

Components for setup:

- Install MongoDB
- Configure MongoDB Network Interface
- Create database administration user
- Setup authentication
- Create database and database users
- Create collections

4.1.1 Install MongoDB

This installation will install MongoDB server, daemon, tools, and shell.

Steps:

- Install the MongoDB certificate to Apt.
    - Option 1 - download the certificate from *https://www.mongodb.org/static/pgp/server-4.0.asc*
        - *curl "https://www.mongodb.org/static/pgp/server-4.0.asc" >> mongokey.asc*
        - *sudo apt-key add mongokey.asc*
    - Option 2 - get the key directly from the key server.
        - *sudo apt-key adv --keyserver hkp://keyserver.ubuntu.com:80 --recv 9DA31620334BD75D9DCB49F368818C72E52529D4*
- Add the MongoDB repository to Apt.
    - *echo "deb [arch=amd64] https://repo.mongodb.org/apt/ubuntu bionic/mongodb-org/4.0 multiverse" | sudo tee /etc/apt/sources.list.d/mongodb-org-4.0.list*
- Update repositories.
    - *sudo apt -y update*
- Install MongoDB.
    - *sudo apt -y install mongodb-org*

4.1.2 Configure MongoDB Network Interface

MongoDB is initially setup to only talk to the localhost. We will change the MongoDB configuration file to use the network interface to make it accessible outside of itself.

Steps:

- Edit */etc/mongod.conf*.
    - *vim /etc/mongod.conf*
- Scroll to the *network interfaces* area.
- Change the *bindIp* line.
    - Spacing is very important in this file.
    - *bindIpAll: true*
- Restart MongoDB.
    - *sudo systemctl restart mongod*

4.1.3 Create Database Administration User

This user will be used to create the database and other users.

Steps:

- Start MongoDB
    - *mongo mongodb://[ip address/domain]*
    - *Ex. mongo mongodb://172.16.xxx.xxx)*
- Change to the Admin database
    - *use admin*
- Create the database administration user
    - Use this template:

```
db.createUser( {
user: “dbadmin”,
pwd: “password”,
roles: [ {
role: “userAdminAnyDatabase”, db: “admin” },
“readWriteAnyDatabase” ]
} )
```

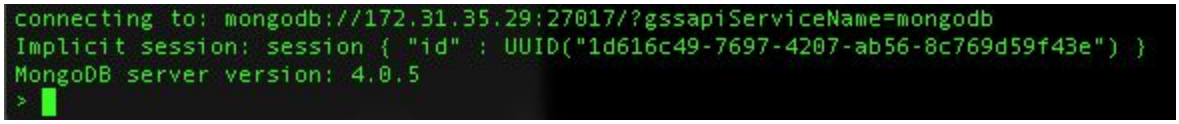
- Exit MongoDB.
    - *exit*

4.1.4 Setup Authentication

Authentication allows you to use username and password for accessing the database. There are more methods for authentication.

Steps:

- Edit */etc/mongod.conf*.
    - *vim /etc/mongod.conf*
- Scroll down to the *security* area and delete the "#" in front of *security.*
- Insert the following line:
    - *authorization: enabled*
    - Make sure that there are two spaces in front of this line.
    - Spacing is very important in this file.
- Save and exit the file
- Set MongoDB to start when the server is booted.
    - *sudo systemctl enable mongod*
- Restart MongoDB.
    - *sudo systemctl restart mongod*
- Verify authentication is working
    - *mongo mongodb://dbadmin:[password]@[ip address/domain]*
    - *Ex. mongo mongodb://dbadmin:password1@eurekalabs.net*
- Example result



4.1.5 Create Database and Database Users

There are five users in the database.

- dbadmin - already created
- dbbackup - for backup administration
- eurekaadmin - for Eureka Labs database administration
- eurekawrite - for read/write operations
- eurekaread - for read only operations

We will create the database in the process of creating these users.

Steps:

- Start MongoDB
    - *mongo mongodb://[ip address/domain]*
    - *Ex. mongo mongodb://172.16.xxx.xxx)*
- Change to the Admin database
    - *use admin*
- Create the database backup user
    - Use this template:

```
db.createUser( {
user: "dbbackup",
pwd: "password",
```

```
roles: [ {
role: "backup", db: "admin" }, {
role: "restore", db: "admin" },
"readWriteAnyDatabase" ]
} )
```

- Create `eureka-labs` database.
    - `use eureka-labs`
- Create eurekaadmin user.
    - You must be in the `eureka-labs` database. The previous step put you in the database.
    - Use this template

```
db.createUser( {
user: "eurekaadmin",
pwd: "password",
roles: [ {
role: "dbAdmin", db: "eureka-labs" }, {
role: "userAdmin", db: "eureka-labs" } ]
} )
```

- Create eurekawrite user.
    - You must be in the `eureka-labs` database
    - Use this template

```
db.createUser( {
user: "eurekawrite",
pwd: "password",
roles: [ {
role: "readWrite", db: "eureka-labs" } ]
} )
```

- Create eurekaread user.
    - You must be in the `eureka-labs` database
    - Use this template

```
db.createUser( {
user: "eurekaread",
pwd: "password",
roles: [ {
role: "read", db: "eureka-labs" } ]
} )
```

4.1.6 Create Collections

Collections are the equivalent of tables in SQL, but they are not as structured as tables. There are two collections for the application.

- User
- Lab

Requirements:

- Must login as *eurekaadmin*.

Steps:

- Start MongoDB.
  - *mongo mongodb://[username]:[password]@[ip address/domain]*
  - *Ex. mongo mongodb://eurekaadmin:password@172.16.xxx.xxx*
- Change to the *eureka-labs* database.
  - *use eureka-labs*
- Create the collections.
  - If you are recreating the database and are going to import data, this is the minimum that has to be done.
    - *db.createCollection("user")*
    - *db.createCollection("lab")*

4.2 Migrating Data From Existing Database

These instructions are to be used if you are creating a new database server and want data from the old server to be on the new server. You can use the export data instructions for backing up a database. You can also use the import data instructions for restoring database information from a backup.

4.2.1 Exporting Data / Database Backup

Requirements:

- Usernames and password for the backup user
- Database ip address or FQDN

Steps:

- Verify MongoDB tools are installed.
  - *dpkg --list | grep mongodb-org-tools*
  - If they are not installed, follow the directions in section 3.1.4.

- Export the data.
  - *mongodump --uri mongodb://dbbackup:[password]@[ip address/domain name] --archive=[file name]*

```
ubuntu@ip-172-31-46-35:~$ mongodump --uri mongodb://dbbackup:[redacted]@db1.eurekalabs.net --archive=dbarchive.mongo
2019-04-21T03:00:55.337+0000    writing admin.system.users to archive 'dbarchive.mongo'
2019-04-21T03:00:55.361+0000    done dumping admin.system.users (5 documents)
2019-04-21T03:00:55.361+0000    writing admin.system.version to archive 'dbarchive.mongo'
2019-04-21T03:00:55.384+0000    done dumping admin.system.version (2 documents)
2019-04-21T03:00:55.384+0000    writing eureka-labs.lab_v2 to archive 'dbarchive.mongo'
2019-04-21T03:00:55.401+0000    writing eureka-labs.lab to archive 'dbarchive.mongo'
2019-04-21T03:00:55.417+0000    writing eureka-labs.user to archive 'dbarchive.mongo'
2019-04-21T03:00:55.483+0000    done dumping eureka-labs.user (10 documents)
2019-04-21T03:00:55.485+0000    done dumping eureka-labs.lab_v2 (12 documents)
2019-04-21T03:00:55.485+0000    done dumping eureka-labs.lab (10 documents)
ubuntu@ip-172-31-46-35:~$
```

- You can zip the file if you will be storing it for later use by adding `--gzip` to the previous command

4.2.2 Importing Data / Restoring Database

The database needs to be setup with the correct users at a minimum. Mongorestore will not overwrite any existing records. It will write new records if they do not exist.

Requirements:

- Database created
- Users created

Steps:

- Verify MongoDB tools are installed.
  - `dpkg --list | grep mongodb-org-tools`
  - If they are not installed, follow the directions in section 3.1.4.

```
ubuntu@ip-172-31-35-29:~$ dpkg --list | grep mongodb-org-tools
ii  mongodb-org-tools            4.0.5                           amd64        MongoDB tools
ubuntu@ip-172-31-35-29:~$
```

- Import data
  - `mongorestore --uri mongodb://dbbackup:[password]@[ip address/domain name] --archive=[archive name]`
  - `Ex. mongorestore --uri mongodb://dbbackup:$DBBACKUP@db1.eurekalabs.net --archive=dbarchive.mongo`
  - If the archive is zipped, you can add the `--gzip` switch in front of the `--archive` switch.

```
ubuntu@ip-172-31-46-35:~$ mongorestore --uri mongodb://dbbackup:$DBBACKUP@db1.eurekalabs.net --archive=dbarchive.mongo
2019-04-21T03:21:52.058+0000    preparing collections to restore from
2019-04-21T03:21:52.126+0000    reading metadata for eureka-labs.user from archive 'dbarchive.mongo'
2019-04-21T03:21:52.127+0000    restoring eureka-labs.user from archive 'dbarchive.mongo'
2019-04-21T03:21:52.136+0000    reading metadata for eureka-labs.lab_v2 from archive 'dbarchive.mongo'
2019-04-21T03:21:52.136+0000    restoring eureka-labs.lab_v2 from archive 'dbarchive.mongo'
2019-04-21T03:21:52.141+0000    reading metadata for eureka-labs.lab from archive 'dbarchive.mongo'
2019-04-21T03:21:52.141+0000    restoring eureka-labs.lab from archive 'dbarchive.mongo'
2019-04-21T03:21:52.186+0000    error: multiple errors in bulk operation:
```

5. Firewalls

5.1 Firewall Information

This application has always sat behind external firewalls. It is recommended that firewalls be used to regulate traffic to/from the web server and to/from the database server. All traffic to the web server via port 80 or 443 should be allowed. The rest of the traffic should be denied with exceptions for necessary access to the servers.

5.1.1 Web Server Firewall Access

This describes the access needed to and from the web server

- Inbound
    - Port 22 - SSH
        - This should be allowed for the IP addresses or IP address range that will be used to manage the web server.
    - Port 53 - DNS
        - Used for name resolution
    - Port 80 - HTTP
        - all web traffic should be allowed.
    - Port 443 - HTTPS
        - All secure web traffic should be allowed.
    - Port 27017 - MongoDB
        - Traffic to the database should be restricted to only the database server or any other MongoDB source being used for development.
- Outbound
    - Should be exactly the same as inbound.

The application makes API calls to Google for Captcha and Sendgrid for email.

5.1.2 Database Firewall Access

This describes access needed for the database server.

- Inbound
    - Port 22 - SSH
        - This should be allowed for the IP addresses or IP address range that will be used to manage the web server.
    - Port 53 - DNS
        - This port will be used when making updates to the server, otherwise it can be turned off.

        - Port 443 - HTTPS
            - This port will be used when making updates to the server, otherwise it can be turned off.
        - Port 27017 - MongoDB
            - This port should only receive data from the web server.
- Outbound
    - Should have the same access as inbound.

There were issues installing MongoDB because it was behind a layer 7 firewall and access to that specific repository needed to be granted.

6. Application

6.1 Flask

6.1.1 Flask Tutorial

Visit this tutorial if you have questions on Flask and how it works:
http://flask.pocoo.org/docs/1.0/tutorial/

6.1.2 How We Use Flask

We use Flask with blueprints. This basically splits one large Flask application into two separate smaller ones namely Labs and Users. Each of these applications handle their own url routing and business logic and are then combined into one master application. The combining of the apps is shown in the app.py file.

6.2 File Structure

6.2.1 Lab

The Lab directory holds all of the python code concerning the lab flask application mentioned above. This includes forms, views, models, and helpers. Forms is how we create a copy of an html form in python which helps with validation and other useful features. Views is where we take url requests coming in and return the correct html document. We can also do our processing here like sending emails and updating the database. The models file is where we define the database objects we are working with. For a reference on this see https://en.wikipedia.org/wiki/Object-relational_mapping . The helpers file is where we put some useful setup functions that are needed throughout the application.

6.2.2 User

The User directory holds all of the python code concerning the user flask application mentioned above. This includes forms, views, and models. Forms is how we create a copy of an html form in python which helps with validation and other useful features. Views is where we take url requests coming in and return the correct html document. We can also do our processing here like sending emails and updating the database. The models file is where we define the database objects we are working with. For a reference on this see https://en.wikipedia.org/wiki/Object-relational_mapping .

6.2.3 Templates

The templates directory is where we keep all of the html files. The html files use a templating engine called jinja2 : http://jinja.pocoo.org/docs/2.10/templates/ . This helps us to insert data into html files to then present to users.

6.3 Environment Variables

6.3.1 Reference

These are the environment variables that need to be set in order for the program to run correctly. This is done for flexibility and security.
EUREKAWRITE - “This is the database password”
SECURITY_KEY - “This is for setting the password hashing”
SECURITY_PASSWORD_SALT - “This is the salt for the password hashing”
RECAPTCHA_PUBLIC_KEY - “This is the public key for the captcha”
RECAPTCHA_PRIVATE_KEY - “This is the private key for the captcha”

Recommended Settings Config: I would recommend creating a secrets.txt file on the database. In this file you can store all of your environmental variables. IMPORTANT: Don’t track this file in your git repository. From here you can open the file in python and then just read in the lines.

6.4 Running The Application

6.4.1 App.py

In order to run the application you need to run “python3 app.py” from the command line. On the server you are going to want to run this command as “nohup python3 app.py &” which will start the server in the background and tell it to not exit the command once you log out. This is ideal for running the server in production. If you are looking for some performance upgrades I would start with using a wsgi socket connected to nginx as a proxy. The other option is running multiple flask servers and using nginx as a load balancer.

7. Glossary of Terms

AWS	- Amazon Web Services.
Certbot	- Let's Encrypt Python application that will install and keep updated the TLS certificate.
DNS	- Domain Name System, the method in which alphanumeric URLs are converted to IP addresses.
FQDN	- Fully Qualified Domain Name - the name of the server. Ex. www.eurekalabs.net or db1.eurekalabs.net.
HTTP	- HyperText Transfer Protocol - the method and language used to present web pages for the application.
HTTPS	- HyperText Transfer Protocol Secure - the secure method of serving web pages.
Let's Encrypt	- a free, automated, and open certificate authority for issuing TLS certificates.
MongoDB	- The database that the application uses. There are several parts that are installed on the web server and the full install is on the database server.
NGINX	- web server application.
Pip	- is a package installer for Python. It pulls from the Python Package Index.
PPA	- Personal Package Archive - A third party archive for APT to get installation files.
Sendgrid	- a web based email platform used to send outgoing emails from the application.
TLS	- Transport Layer Security - this is used for serving HTTPS web pages.
Ubuntu	- the operating system used for the servers. Currently using 18.04 LTS.
Vim	- a linux text editing program.
VM	- Virtual Machine, a computer in a virtualized environment (AWS, VMWare).