DOCUDB

INSTALLATION GUIDE

WEB APPLICATION

NOTE: This guide assumes that you will use an Ubuntu system, preferably Ubuntu 16.04, for the production server.

STEP 0. UPDATE SERVER SYSTEM

Run the following commands on the system:

```
sudo apt-get update
sudo apt-get upgrade
```

STEP 1. INSTALL RUBY

Install some dependecies for Ruby:

```
sudo apt-get install git-core curl zlib1g-dev build-essential libssl-dev
libreadline-dev libyaml-dev libsqlite3-dev sqlite3 libxml2-dev libxslt1-dev
libcurl4-openssl-dev python-software-properties libffi-dev imagemagick php5-
imagick -y
```

Next, install rbenv, a version manager for Ruby. This will allow us to specify which version of Ruby to use, in this case Ruby 2.2.3

```
cd
git clone https://github.com/rbenv/rbenv.git ~/.rbenv
echo 'export PATH="$HOME/.rbenv/bin:$PATH"' >> ~/.bashrc
echo 'eval "$(rbenv init -)"' >> ~/.bashrc
exec $SHELL

git clone https://github.com/rbenv/ruby-build.git ~/.rbenv/plugins/ruby-build
echo 'export PATH="$HOME/.rbenv/plugins/ruby-build/bin:$PATH"' >> ~/.bashrc
exec $SHELL

rbenv install 2.2.3
rbenv global 2.2.3
ruby -v
```

The last step is to install Bundler and rehash after installation.

```
gem install bundler
rbenv rehash
```

STEP 2. INSTALL NGINX

Phusion is the company that develops Passenger and they recently put out an official Ubuntu package that ships with Nginx and Passenger pre-installed.

We'll be using that to setup our production server because it's very easy to setup.

```
sudo apt-key adv --keyserver hkp://keyserver.ubuntu.com:80 --recv-keys
561F9B9CACec40B2F7
sudo apt-get install -y apt-transport-https ca-certificates

# Add Passenger APT repository
sudo sh -c 'ho deb https://oss-binaries.phusionpassenger.com/apt/passenger
xenial main > /etc/apt/sources.list.d/passenger.list'
sudo apt-get update

# Install Passenger & Nginx
sudo apt-get install -y nginx-extras passenger
```

Start the Nginx webserver with the following command.

```
sudo service nginx start
```

Open up the server's IP address in your browser to make sure that nginx is up and running.

The service command also provides some other methods such as `restart` and `stop` that allow you to easily restart and stop your webserver.

Next, we need to update the Nginx configuration to point Passenger to the version of Ruby that we're using. You'll want to open up /etc/nginx/nginx.conf in your favorite editor. I like to use vim, so I'd run this command:

```
sudo vim /etc/nginx/nginx.conf
```

Find the following lines, and uncomment them:

```
##
# Phusion Passenger
##
# Uncomment it if you installed ruby-passenger or ruby-passenger-enterprise
##
```

```
passenger_root /usr/lib/ruby/vendor_ruby/phusion_passenger/locations.ini;
passenger_ruby /home/deploy/.rbenv/shims/ruby;
```

The `passenger_ruby` is the important line here. Make sure you only set this once and use the line from the example that pertains to the version of Ruby you installed..

Next, we need to add an Nginx host for our application. First, modify the default Nginx `sites-available` configuration file and comment out the following lines:

```
sudo vim /etc/nginx/sites-available/default
# listen 80 default_server;
# listen [::]:80 default_server ipv6only=on;
```

Then, we create a new file for our application and type in the following values:

```
sudo vim /etc/nginx/sites-available/docudb
server {
  listen 80 default_server;
  server_name dms.up-ovpd.ph;
  passenger_enabled on;
  rails_env production;
  client_max_body_size 0;
  root /home/sysadmin/docudb/current/public;
}
```

Afterwards, create a symlink from our newly created file to `sites-enabled`.

```
sudo ln -s /etc/nginx/sites-available/docudb /etc/nginx/sites-enabled/docudb
```

Finally, restart the Nginx service using `sudo service nginx restart`.

Now that we've restarted Nginx, the Rails application will be served up using the deploy user just how we want.

STEP 3. INSTALL MYSQL

All you need to do in order to install MySQL is to run the following command:

```
sudo apt-get install mysql-server mysql-client libmysqlclient-dev
```

Once MySQL is installed, it is preferable that we setup up the user and databases now so that we do not have to worry about it later. Refer to the database configuration file in the application located at `/config/database.yml` for the values to use. Run the following commands inside the MySQL shell:

NOTE: Please replace the values enclosed in <> with those found in `/config/database.yml` and `/config/application.yml`. They contain the sensitive database credentials needed here.

```
CREATE USER '<your_production_user>'@'localhost' IDENTIFIED BY
'<your_password>';
GRANT ALL PRIVILEGES ON * . * TO '<your_production_user>'@'localhost';
FLUSH PRIVILEGES;
```

Login to the MySQL shell using your newly created user and run the following commands:

```
CREATE DATABASE IF NOT EXISTS <your_database_name>;
USE <your_database_name>;
```

STEP 4. INSTALL ELASTICSEARCH

Elasticsearch is a platform for distributed search and analysis of data in real time. Its popularity is due to its ease of use, powerful features, and scalability.

To install Elasticsearch, first install Java as its dependency.

```
sudo apt-get install openjdk-7-jre -y
```

Next, we download and install Elasticsearch with the following commands:

```
wget https://download.elastic.co/elasticsearch/elasticsearch/elasticsearch-
1.7.2.deb
sudo dpkg -i elasticsearch-1.7.2.deb
```

Finally, setup the Elasticsearch service and run it.

```
sudo update-rc.d elasticsearch defaults
sudo service elasticsearch start
```

STEP 5. SETUP CAPISTRANO

NOTE: For Capistrano, make sure you do these steps on your development machine inside your rails app. For the purposes of this installation manual, these steps should have already been completed and this guide is only here for reference if some changes are to be made. As such, this part will proceed under the assumption that the necessary configuration files already exist and are prepared.

`/Capfile` contains the Capistrano library requirements.

`/config/deploy.rb` contains the deployment configurations, such as the repository URL, Ruby version, deployment path, and directories and files that persist through multiple versions.

`/config/deploy/production.rb` contains the server configurations, such as the server IP address, user, and SSH options.

Before deploying the server, we must first make sure to copy over these configuration files over to the server. While Capistrano will do most of the heavy lifting for us by pulling the source code from the specified repository to the server, these configuration files may contain sensitive information about the application and as such is usually not uploaded to the repository. Upload the files with the following commands:

NOTE: As always, remember to change the values enclosed in <> with the configuration you are using.

```
scp config/application.yml <server_ssh>:~/docudb/shared/config/application.yml
scp config/database.yml <server_ssh>:~/docudb/shared/config/database.yml
scp config/secrets.yml <server_ssh>:~/docudb/shared/config/secrets.yml
```

Finally, deploy the app to the server with the command `cap production deploy`.

STEP 6. CREATING THE ADMIN USER

As the application is intended to be used only for specific users, the sign up page is hidden and requires admin approval. As such, we will bypass this restriction by creating the first admin user through the command line. Afterwards, the admin user can simply login to the application and create additional users through the User Management module.

To create the first admin user, go to the application's current directory and run the following command:

```
cd ~/docudb/current/
RAILS_ENV=production bundle exec rails console
```

Inside the Rails console shell, run the following command:

```
User.create!( email: "<admin_email>", password: "<admin_password>",
      admin: true, is_approved: true )
```

This should create the admin user.