Meta Alligators Team 4 Deployment Report

The hypotheticalmeals code can be found on gitlab at the following link:
https://gitlab.com/maddierosenelson/hypothetical_meals

Running locally

To run the web server locally, the steps are as follows:

1) Make sure you have Node.js and Git installed. The links to download them are:
    a) https://git-scm.com/downloads
    b) https://nodejs.org/en/download/
2) Clone the repository into some directory using git clone
3) Make sure you are in the highest level directory of the repository and then run npm install
4) Change into the directory labeled client using cd ./client
5) Run npm install again now that you are in the client directory
6) Change back into the highest directory using cd ..
7) You can now deploy the server using npm run start:dev

The server will then be accessible by going to http://localhost:3000. The front end will listen on port 3000 of your local server and the back end will listen on port 3001.

MERN (MongoDB, Express, React, and Node.js)

The MERN stack stands for MongoDB, Express, React, and Node.js. These four technologies create the stack and allow a developer to easily create a real-world application in minimal time using JavaScript for the entire stack. The four technologies do the following:

- Node.js is an open source server environment that enables the use of JavaScript with the server.
- MongoDB is the responsible for the database portion of the stack and specifically, MongoDB uses a document-based database.
- Express is a web framework for Node.js
- React is the client portion of the stack and also written in Javascript to allow developers to build user interfaces.

Downloading the Appropriate Software

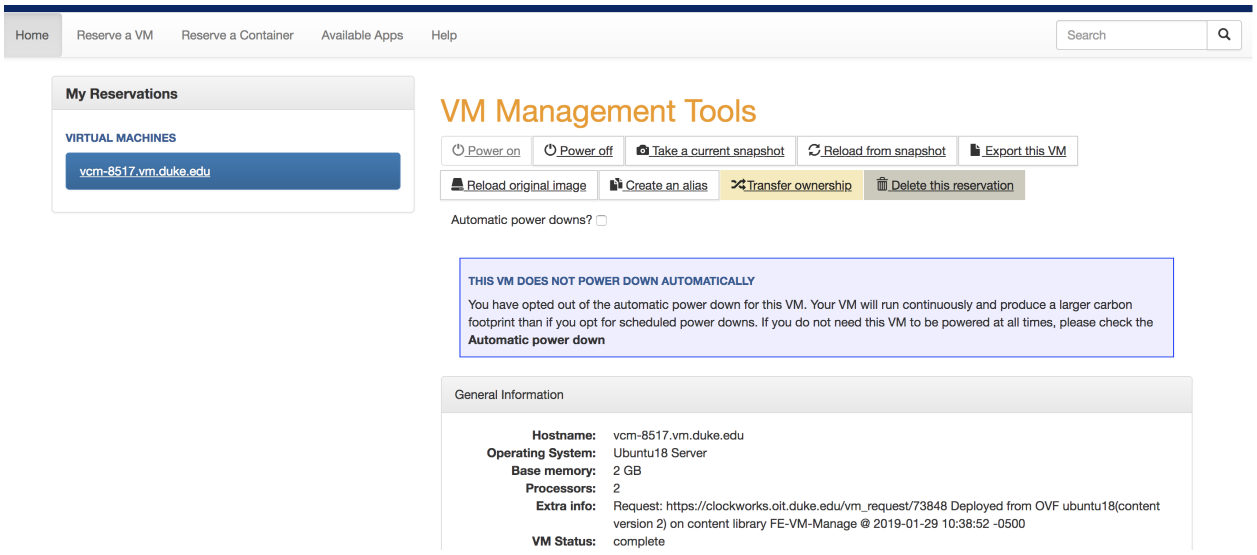
Before beginning using our hypothetical meals website, we must download the following items:

- Download Git here:https://git-scm.com/downloads
- Download node.js here: https://nodejs.org/en/download/.

Deploying The Application To A Production Server

Setting up the Virtual Machine

To deploy the code to a production server, we need to obtain a Virtual Machine. A simple way to do this is through the Duke University Virtual Computing Manager (https://vcm.duke.edu/). At this site, choose to "Reserve a VM" and follow Duke's VCM instructions for reserving a Ubuntu18 server. Next, login to your reservations to view your VM Management Tools. The page should look like the below image:



Automatic power downs? ☐

THIS VM DOES NOT POWER DOWN AUTOMATICALLY

You have opted out of the automatic power down for this VM. Your VM will run continuously and produce a larger carbon footprint than if you opt for scheduled power downs. If you do not need this VM to be powered at all times, please check the **Automatic power down**

General Information

Hostname:	vcm-8517.vm.duke.edu
Operating System:	Ubuntu18 Server
Base memory:	2 GB
Processors:	2
Extra info:	Request: https://clockworks.oit.duke.edu/vm_request/73848 Deployed from OVF ubuntu18(content version 2) on content library FE-VM-Manage @ 2019-01-29 10:38:52 -0500
VM Status:	complete

In the top left corner, you will find the VM address. In the above picture it is: vcm-8517.vm.duke.edu, but the number will vary for each VM. On this page it is important to opt-out of automatic power downs in order to ensure your application will be running on the production server continuously.

Next, you can create an alias at the bottom of this page with the name of the URL that you want your application to be hosted on. For the purpose of this deployment, the alias was set to:

metaalligators.colab.duke.edu

This alias will be the servername used in following steps.
Next, open a terminal window and ssh into your vcm using the following command.

```
ssh vcm@vcm-8517.vm.duke.edu
```
*Note: the number will be different for your VM.

You will then be prompted to enter the password which can be found on the VM Management Tools (picture above)

If successful, you should now be in your virtual machine.

Downloading the Code to the Virtual Machine

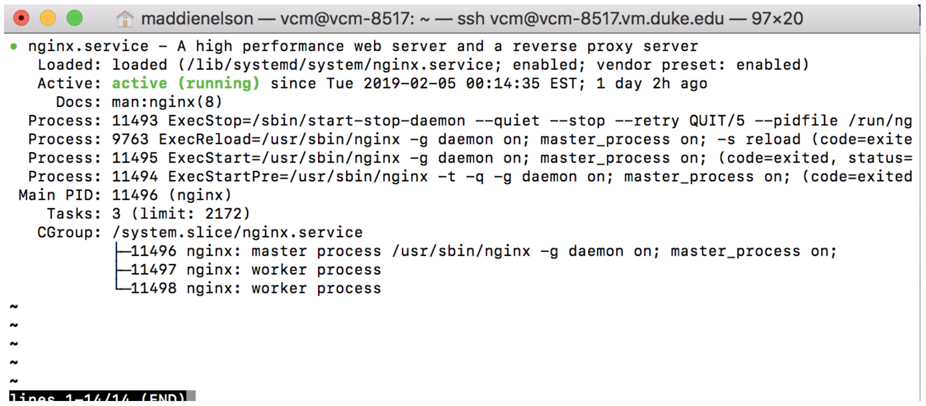
While SSHed into your VM, you must now create a folder where you would like to clone the hypothetical meals repository and then clone the repository from:
https://gitlab.com/maddierosenelson/hypothetical_meals.
Once the code has been cloned, navigate into the hypotheticalmeals folder and run:
npm install
followed by
cd client && npm install.
Next, we must run the command **npm run build** to create the static HTML files of the client portion of our application.
In the file backend/server.js we direct our application page to the client/build/index.html and we have the application use the express.static() function.

Generating Certificates and SSL

Because we are using two ports, one for our server, and one for our client, and proxying from one to the other, we need to use Nginx to support this communication and proxying. To setup Nginx and this certification process we run the following commands.
We run **sudo apt-get update** to ensure that we have the most recent package listings and then we will run **sudo apt-get install nginx** to install nginx.
For security reasons, we must have a firewall and we can configure our firewall, and allow Nginx to access it. To see what applications can access our service we can type the following:
sudo ufw app list
Because we want our service to be as secure as possible, we will choose to only use Nginx HTTPS which will ensure that the traffic is SSL encrypted. To enable the Nginx HTTPS we write the following command: **sudo ufw allow 'Nginx HTTPS' .** To ensure that this change was correctly made, or to check your firewall at any time, you can use **sudo ufw status**

Web Server

Now that we have enabled and checked our firewall, the web server should be running. We can check this with **systemctl status nginx** which will (hopefully) output something like the following:


maddienelson — vcm@vcm-8517: ~ — ssh vcm@vcm-8517.vm.duke.edu — 97×20

```
● nginx.service - A high performance web server and a reverse proxy server
   Loaded: loaded (/lib/systemd/system/nginx.service; enabled; vendor preset: enabled)
   Active: active (running) since Tue 2019-02-05 00:14:35 EST; 1 day 2h ago
     Docs: man:nginx(8)
  Process: 11493 ExecStop=/sbin/start-stop-daemon --quiet --stop --retry QUIT/5 --pidfile /run/ng
  Process: 9763 ExecReload=/usr/sbin/nginx -g daemon on; master_process on; -s reload (code=exite
  Process: 11495 ExecStart=/usr/sbin/nginx -g daemon on; master_process on; (code=exited, status=
  Process: 11494 ExecStartPre=/usr/sbin/nginx -t -q -g daemon on; master_process on; (code=exited
 Main PID: 11496 (nginx)
    Tasks: 3 (limit: 2172)
   CGroup: /system.slice/nginx.service
           ├─11496 nginx: master process /usr/sbin/nginx -g daemon on; master_process on;
           ├─11497 nginx: worker process
           └─11498 nginx: worker process
~
~
~
~
~
lines 1-14/14 (END)
```

To ensure that this is running, you can find your ip address using:
ip addr show eth0 | grep inet | awk '{ print $2; }' | sed 's/\/.*$//'

You can then navigate to your IP address in your browser and you should see the default Nginx landing page that says welcome to nginx! An important note here is that if you also have apache installed, the two may conflict at times and therefore to stop apache you can use the following command
sudo /etc/init.d/apache2 stop from the root directory.

To manage the nginx processes, use the following commands:

- Stopping your webserver: **sudo systemctl stop nginx**
- Starting your webserver: **sudo systemctl start nginx**
- Start and then Start your webserver: **sudo systemctl restart nginx**

<u>Installing Certbot and Getting The SSL Certificate</u>
To generate the certbot we will access the certbot repository with the following command:
sudo add-apt-repository ppa:certbot/certbot
After entering this command you will press enter to accept and then update the package list using
sudo apt-get update
The package we will use with the Certbot and Nginx is the Certbot's Nginx package which we can get with
sudo apt-get install python-certbot-nginx

<u>Configuring the Nginx</u>
To configure the Nginx we need to ensure that it has the correct server block in the config. We do this by accessing our default config file with the command: **sudo nano /etc/nginx/sites-available/default**
In this file we will be replacing the underscore of our servername with the alias we have created, in our case this is: metaalligators.colab.duke.edu
Any time we are configuring the nginx file, we should verify the syntax of our changes using the command **sudo nginx -t.**

<u>Allowing HTTPS Through the Firewall</u>
As mentioned before, we want to check the status of our firewall by typing: **sudo ufw status.** To ensure that we do not allow simple HTTP calls we can run the following commands: **sudo ufw allow 'Nginx Full'** followed by **sudo ufw delete allow 'Nginx HTTP'.** This will delete the HTTP option from our list and ensure that our server does not accept them.

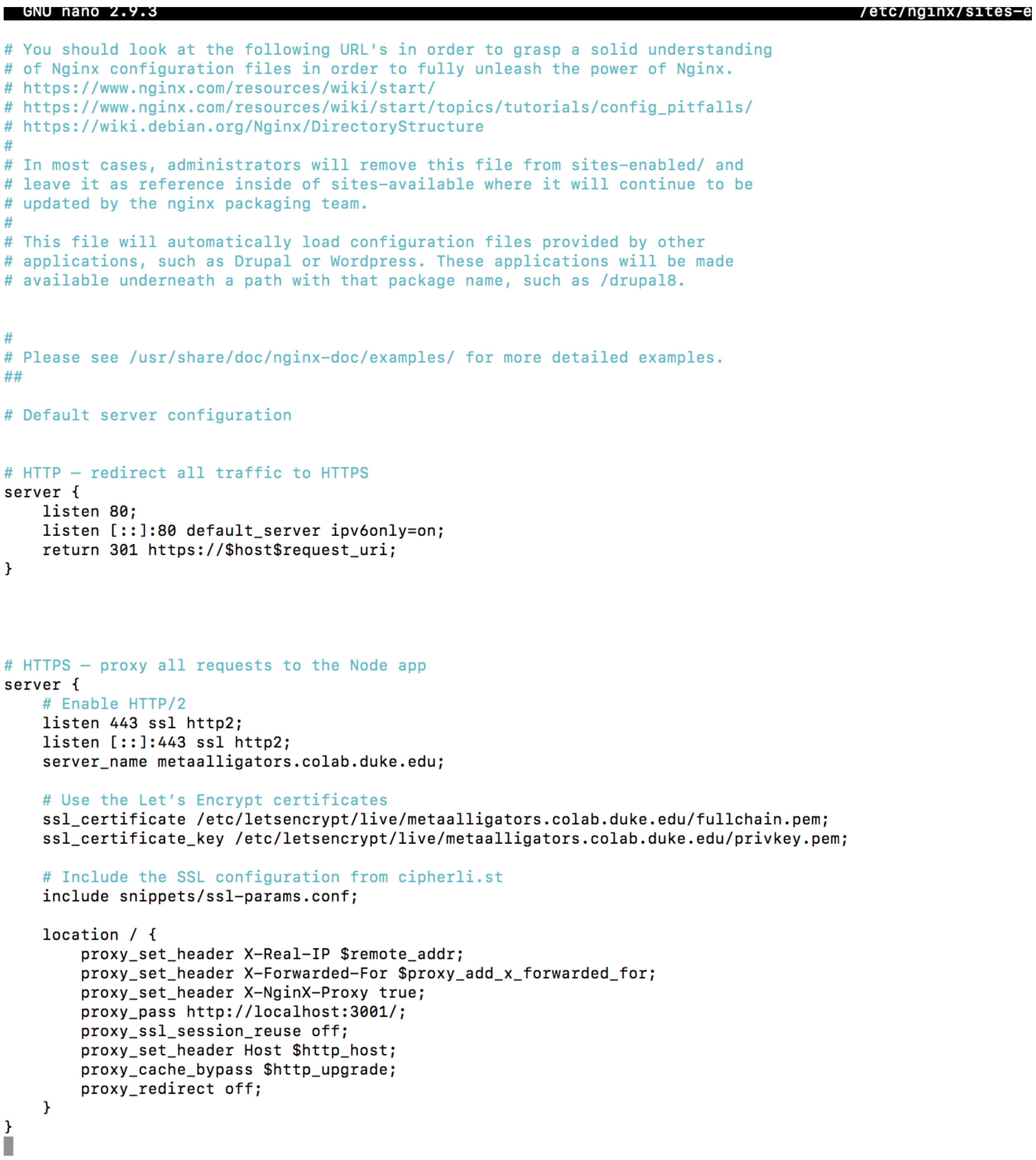
<u>Obtaining an SSL Certificate</u>
As discussed, we will use Nginx and Certbot for our SSL certificates. To obtain this we use the command **sudo certbot --nginx -d metaalligators.colab.duke.edu.** We use -d to specify the names we'd like the certificate to be valid for. We are prompted to enter an email address when we first run certbot and we will also be asked if we would like to redirect HTTP traffic to HTTPS or not. Select option 2, which is to redirect the traffic.

```
Output
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):
```

Your certificate is now set up!

More Configuration of the Default File

Return to the default file using: **sudo nano /etc/nginx/sites-enabled/default** and ensure that your file has the following information.


```
  GNU nano 2.9.3                                                                                /etc/nginx/sites-e

# You should look at the following URL's in order to grasp a solid understanding
# of Nginx configuration files in order to fully unleash the power of Nginx.
# https://www.nginx.com/resources/wiki/start/
# https://www.nginx.com/resources/wiki/start/topics/tutorials/config_pitfalls/
# https://wiki.debian.org/Nginx/DirectoryStructure
#
# In most cases, administrators will remove this file from sites-enabled/ and
# leave it as reference inside of sites-available where it will continue to be
# updated by the nginx packaging team.
#
# This file will automatically load configuration files provided by other
# applications, such as Drupal or Wordpress. These applications will be made
# available underneath a path with that package name, such as /drupal8.


#
# Please see /usr/share/doc/nginx-doc/examples/ for more detailed examples.
##

# Default server configuration


# HTTP — redirect all traffic to HTTPS
server {
    listen 80;
    listen [::]:80 default_server ipv6only=on;
    return 301 https://$host$request_uri;
}




# HTTPS — proxy all requests to the Node app
server {
    # Enable HTTP/2
    listen 443 ssl http2;
    listen [::]:443 ssl http2;
    server_name metaalligators.colab.duke.edu;

    # Use the Let's Encrypt certificates
    ssl_certificate /etc/letsencrypt/live/metaalligators.colab.duke.edu/fullchain.pem;
    ssl_certificate_key /etc/letsencrypt/live/metaalligators.colab.duke.edu/privkey.pem;

    # Include the SSL configuration from cipherli.st
    include snippets/ssl-params.conf;

    location / {
        proxy_set_header X-Real-IP $remote_addr;
        proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
        proxy_set_header X-NginX-Proxy true;
        proxy_pass http://localhost:3001/;
        proxy_ssl_session_reuse off;
        proxy_set_header Host $http_host;
        proxy_cache_bypass $http_upgrade;
        proxy_redirect off;
    }
}
```

After doing this, it is smart to once again start Nginx and also ensure that Apache is stopped using the following commands: **sudo /etc/init.d/apache2 stop** and **sudo systemctl start nginx**

<u>We Are Ready To Run!</u>

Now, navigate to the hypotheticalmeals folder you cloned from gitlab. After ensuring that you have correctly done **npm install** in the root and in the client folder, use the following command to run your application on the production server.

npm run start:dev

This application is now accessible from any computer at https://metaalligators.colab.duke.edu. Enjoy!

To login to the application as an admin, use the following login:
Email: riley@admin.com
Password: testtest