Instructions on how to setup Flask for Linode via Windows Shell

Reference: https://www.youtube.com/watch?v=goToXTC96Co

1. Look for the IP address of your Linode server under the **"Networking Tab"**
2. Open command prompt on your local desktop
3. Type in **ssh root@<ip address>** // ip address of your linode server. You may use Putty or some other terminal emulator.
4. Run **apt update && apt upgrade** to install any updates on your linode server
5. Run **hostnamectl set-hostname <name>** (Here I will use flask-server for the name)
6. Run **nano /etc/hosts** and in that file write the following below the 127.0.0.1 so it should look like:
    i. 127.0.0.1 localhost
    ii. <ip address> flask-server (This is the <name> # name is from step 4)
    b. Hit **CTRL-X** to exit
    c. Type **Y** for yes
    d. Hit **Enter**
7. Create a limited user by running the following commands
    a. **adduser <username>**
    b. **adduser <username> sudo**
8. Exit ssh session by typing **exit** into the shell
9. Login in again via command prompt as the limited user IE <username>@<ip address>
10. Next, we will need to setup the firewall
    i. sudo apt install ufw
    ii. sudo ufw default allow outgoing
    iii. sudo ufw default deny incoming
    iv. sudo ufw allow ssh
    v. sudo ufw allow 5000
    vi. sudo ufw enable
11. Now run **sudo ufw status** ← See allowances
12. Transfer over your flask files to server via FTP such as FileZilla or running **git clone <url>**
13. Now setup the python virtual environment (VE) on server
    a. Run **sudo apt install python3-pip**
    b. Run **sudo apt install python3-venv**
    c. Run **python3 -m venv project/to/project/<VE name>**
    d. **cd /path/to/project**
    e. Run **source venv/bin/activate**
    f. Get all the environment variables from the flask project
        i. Secret Key
        ii. Db URL
        iii. Email_user

iv. Email_pass
v. Etc.

g. Add the project variables to a **config.json** file
   i. run **sudo touch /etc/config.json**
   ii. run **sudo nano /etc/config.json**
   iii. In the **config.json** file add the following:

```
{
    "SECRET_KEY": < Secret Key >,
    "SQLALCHEMY_DATABASE_URI": < URI >,
    "SQLALCHEMY_TRACK_MODIFICATIONS": "FALSE",
    "MAIL_SERVER": "smtp.gmail.com",
    "MAIL_PORT": 587,
     "MAIL_USE_TLS": 1,
     "MAIL_USERNAME": < email >,
     "MAIL_PASSWORD": < pass >
}
```

h. Add the following lines of python code to your config.py file
   i.
```
with open("/etc/config.json") as config_file:
    config=json.load(config_file)
```
   ii. Load the json dictionary to the different parts of the config file

i. Test the Flask app by running
   i. **export FLASK_APP = app.py**
   ii. **flask run -- host=0.0.0.0** (Exposes flask app to the outside world)

j. Go to your local browser and type in **<ip address>:5000** and you should see your flask app

14. Head back to the Command Prompt and ensure your in your virtual environment **(step 13e)**
15. Run the following commands:
    a. **sudo apt install nginx**
    b. **pip install gunicorn**
16. **Nginx** will be the web server and handle the static requests such as load an image and **Gunicorn** will handle the python code
17. Run **sudo rm /etc/nginx/sites-enabled/default** (Removes the default nginx config file)
18. Create a new flask config file by running
    a. **sudo nano /etc/nginx/sites-enabled/flaskblog**
    b. Add the following lines:
       i. **server {**
       ii.
       iii. **listen 80;**
       iv. **server_name www.raj302.com;**
       v. **location /static {**

vi. **alias /home/raji/master-flask/app/static/;**
vii. **}**
viii.
ix. **location / {**
x. **proxy_pass http://localhost:8000;**
xi. **include /etc/nginx/proxy_params;**
xii. **proxy_redirect off;**
xiii. **}**
xiv. **}**

19. Open port 80 in UFW by running
    a. **sudo ufw allow http/tcp**
    b. **sudo ufw delete allow 5000**
    c. **sudo systemctl restart nginx**
20. Check to see if gunicorn is now by running
    a. **cd** into your main project folder
    b. run **gunicorn -w 3 run:app**
        i. **# of workers** = 2x# cores+1
        ii. **run:app** is basically saying from run.py use the app variable
            1. (app.py or could be run.py whatever the name of your flask initialization file is)
        iii. This will start gunicorn within your command prompt, but we need something that can run on its own
21. Setting up **gunicorn** to run on its own by running the following:
    a. **sudo apt install supervisor**
    b. **sudo nano /etc/supervisor/conf.d/simple-flask.conf** (config file)
    c. Add to the config file
        i. **[program: <name>] //This can be any name**
        ii. **directory= /home/user/path/to/project/folder**
        iii. **command= /home/user/path/to/project/folder/venv/bin/gunicorn -w 3 run:app**
        iv. **user=raji**
        v. **autostart=true**
        vi. **autorestart=true**
        vii. **stopasgroups=true**
        viii. **killasgroup=true**
        ix. **stderr_logfile = /var/log/simple_flask/flasklog.err.log //These can be anywhere you want.**
        x. **stdout_logfile=/var/log/simple_flask/flasklog.out.log**
    d. Make directories for log files by
        i. **sudo mkdir -p /var/log/simple_flask**
        ii. **sudo touch /var/log/simple_flask/flasklog.err.log**

iii. **sudo touch /var/log/simple_flask/flasklog.out.log**

e. **run sudo supervisorctl reload;**

22. If your flask project uses **flask_mail** or some other email package add the following rules to ufw by running either one of the two depending if your using gmail or some other email server to open either port.
    i. **sudo ufw allow 465**
    ii. **sudo ufw allow 587**