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>**
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:
   * 1. 127.0.0.1 localhost
     2. <ip address> flask-server (This is the <name> # name is from step 4)
   1. Hit CTRL-X to exit
   2. Hit Y for yes
   3. Hit Enter
7. Create a limited user by running the following commands
   1. adduser <username>
   2. 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
    * 1. sudo apt install ufw
      2. sudo ufw default allow outgoing
      3. sudo ufw default deny incoming
      4. sudo ufw allow ssh
      5. sudo ufw allow 5000
      6. 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
    1. Run sudo apt install python3-pip
    2. Run sudo apt install python3-venv
    3. Run python3 -m venv project/to/project/<VE name>
    4. CD /path/to/project
    5. Run source venv/bin/activate
    6. Get all the environment variables from the flask project
       1. Secret Key
       2. Db URL
       3. Email\_user
       4. Email\_pass
       5. Etc.
    7. Add the project variables to a config.json file
       1. Here run sudo touch /etc/config.json
       2. sudo nano /etc/config.json
       3. In the config.json file add the following:
          1. {
          2. “SECRET\_KEY”:
          3. “SQLALCHEMY\_DATABASE\_URI”:
          4. “EMAIL\_USER”:
          5. “EMAIL\_PASS”:
          6. }
    8. Add the following lines of python code to your config.py file
       1. With open(“/etc/config.json”) as config\_file:
          1. config=json.load(config\_file)
       2. Load the json dictiobary to the different parts of the config file
    9. Test the Flask app by running
       1. export FLASK\_APP = app.py
       2. flask run -- host=0.0.0.0 (Exposes flask app to the outside world)
    10. 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:
    1. Sudo apt install nginx
    2. 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. Next run sudo rm /etc/nginx/sites-enabled/default (Removes the default nginx config file)
18. Create a new flask config file by running
    1. sudo nano /etc/nginx/sites-enabled/flaskblog
    2. Add the following lines:
       1. server{
          1. **listen 80;**
          2. **server\_name: ip\_address;**
          3. **location /static {**
          4. **alias /home/raji/simple-flask/templates/;**
          5. **}**
          6. **location / {**
          7. **proxy\_pass** [**http://localhost:8000**](http://localhost:8000)**;**
          8. **Include /etc/nginx/proxy\_params;**
          9. **Proxy\_redirect: off;**
          10. **}**
          11. **}**
    3. Open port 80 in UFW by running
       1. Sudo ufw allow http/tcp
       2. Sudo ufw delete allow 5000
       3. Sudo systemctl restart nginx
19. Start gunicorn now by running
    1. gunicorn -w 3 app:app
       1. # of workers = 2x# cores+1
       2. 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)
       3. This will start gunicorn within your command prompt but we need something that can run on its own
20. Setting up gunicorn to run on its own by running the following:
    1. Sudo apt install supervisor
    2. Sudo nano /etc/supervisor/conf.d/simple-flask.conf (config file)
    3. Add to the config file
       1. [program: <name>] //This can be any name
       2. directory= /home/user/path/to/project/folder
       3. command= /home/user/path/to/project/folder/venv/bin/gunicorn -w 3 run:app
       4. user=raji
       5. autostart=true
       6. autorestart=true
       7. stopasgroups=true
       8. killasgroup=true
       9. stderr\_logfile = /var/log/simple\_flask/flasklog.err.log
       10. stdout\_logfile=/var/log/simple\_flask/flasklog.out.log
    4. Make dirs. For log files by
       1. sudo mkdir -p /var/log/simple\_flask
       2. sudo touch /var/log/simple\_flask/flasklog.err.log
       3. sudo touch /var/log/simple\_flask/flasklog.out.log
    5. **run sudo supervisorctl reload;**