Create a Linode accout and IP

* Go to cloud/linode.com
* Click create
* Select OS: Ubuntu
* Select region: Near to user
* Linode plan: Based on Requirements
* Linode Lable: Photo-Editor
* Root Password: To login to system as root (remember it)
* Optional add ons: Backup for production server
* Click create
* Go to netwoking tag on server: Note down IP addresss

Install Linux bash shell to windows for using ssh (This is only one time)

* Go to windows icon
* Search “features”
* Click “turn widons on and off”
* Scroll down click “widowns subsystem for linux
* Click ok and restart
* Go to windows store
* Search ubuntu
* Download ubuntu
* Open strat menu
* Open ubuntu
* Enter user name and password for ubuntu server
* Now you can run ssh from git bash using commands

General setting

* Ssh root@IP address
* Enter root password: Password when we created sever
* root@localhost:~# apt update && apt upgrade (To update server)
* if asks for y/n enter y
* To set hostname
  + root@localhost:~# hostnamectl set-hostname <hostname> (i.e photo-editor)
  + root@localhost:~# hostname (to see hostname)
* To enter hostname to hostfile
  + root@localhost:~# nano /etc/hosts
  + under 127.00.1 localhost

IP address <hostname> (i.e photo-editor)

Ctr + x to exit, y to save and then enter to save in same file

* To add limited user. Root user has unlimted priviligies and to limit that we create imite that we create limted user
  + root@localhost:~# adduser <name of user>(i.e kaushalk)
  + Enter pass for limted user and remember it too
    - Fill some info about user. Like full name and all. We can skip if we want
  + root@localhost:~# adduser kaushalk sudo (To add that limited user to sudo group)
* exit and login as limited user
  + root@localhost:~# exit
  + $ ssh kaushalk@IP address
  + Passwod of root user

SSH key based atuthentication to access server without entering password

* To create key
  + $ ssh-keygen -b 4096
  + ask that where to save. Press enter for default location
  + asks for passphrase: Enter password If someone else is aceessing your commuter
* To copy ssh public key
  + Option 1
    - Create .ssh directory to remote machine
      * kaushalk@photo-editor:~# mkdir .ssh
    - To copy to that remote server
      * $ scp ~/.ssh/id\_rsa.pub [kaushalk@Ip:~/.ssh/authorised\_keys](mailto:kaushalk@Ip:~/.ssh/authorised_keys)
    - To see that key
      * kaushalk@photo-editor:~# cat ~/.ssh/authorised\_keys
    - To set permission
      * kaushalk@photo-editor:~# chmod 700 ~/.ssh/
      * kaushalk@photo-editor:~# chmod 600 ~/.ssh/\*
  + Option 2
    - ssh-copy-id kaushalk@IP
  + To check it is copied or not
    - kaushalk@photo-editor:~# ls .ssh
      * it should show “authorised\_keys”
* Now we should be able to enter withut password to server.
* We need to disallow shh on root user. To do that
  + kaushalk@photo-editor:~# Sudo nano /etc/ssh/ssh\_config
    - Change two things
      * PermitRootLogin no
      * PasswordAuthentication no
    - Ctr + x, y and enter
* To restart ssh
  + kaushalk@photo-editor:~# sudo systemctl restart sshd

To install firewall(uncomplecated firewall)

* kaushalk@photo-editor:~# sudo apt install ufw
* To setup rules
  + kaushalk@photo-editor:~# sudo ufw default allow outgoing
  + kaushalk@photo-editor:~# sudo ufw default deny incoming
* To set allow rules for certain ports
  + kaushalk@photo-editor:~# sudo ufw allow ssh
  + kaushalk@photo-editor:~# sudo ufw allow 5000 (for testing purpose)
  + kaushalk@photo-editor:~# sudo ufw enable
* To see status of ufw
  + kaushalk@photo-editor:~# sudo ufw status
* Now we can deploy our flask app to server. That can be done with git clone

To deploy flask app to server

* Create requirements.txt file
  + $ pip freeze -> requirements.txt
* Copy that to our app
* To copy app to server
  + $ scp -r location of app(i.e Desktop/Flask\_blog) kaushalk@IP:~/
* To check on server
  + kaushalk@photo-editor:~# ls
    - will show your app(i.e Flask\_blog)
* To create venv to server
  + kaushalk@photo-editor:~# sudo apt install python3-pip
  + kaushalk@photo-editor:~# sudo apt install python3-venv
  + kaushalk@photo-editor:~# python3 -m venv Flask\_blog/venv
* To check
  + kaushalk@photo-editor:~# cd Flask\_blog
  + kaushalk@photo-editor: :/Flask\_blog# ls
    - will show venv file
* To activate
  + kaushalk@photo-editor:/Flask\_blog# source venv/bin/activate
    - will show (venv) kaushalk@photo-editor:~#
* (venv) kaushalk@photo-editor:~# pip install -r requirements.txt

Now we can run app. But in our app if we have used enviornment variable then sometimes some server will have problem of getting acees that. So we create config file in server for that.

* See youtube corey sacher video at time 35 to 43 minutes.

To run app for testing(on development server)

* (venv) kaushalk@photo-editor:/Flask\_blog# export FLASK\_APP=run.py
* (venv) kaushalk@photo-editor:/Flask\_blog# flask run –host=0.0.0.0
* To run type on url
  + Ip address:5000/

To run nginx and gunicorn

* Install
  + (venv) kaushalk@photo-editor:~# sudo apt install nginx
  + (venv) kaushalk@photo-editor:~# pip install gunicorn
* nginx handles web server, For access static files. And to handle python we use gunicorn
* config nginx
  + sudo rm /etc/nginx/sites-enabled/default (to remove default file)
  + sudo nano /etc/nginx/sites-enabled/flaskblog (to create new config file)
    - run following to it

server {

listen 80;

server\_name IP address;

location /static {

alias /home/kaushalk/Photo\_Editor/web\_app/static/stylsheet;

}

locaiton / {

proxy\_pass http:/localhost:8000;

Include /etc/nginx/proxy\_params;

proxy\_redirect off;

}

}

* To open that file on firewall
  + (venv) kaushalk@photo-editor:~# sudo ufw allow http/tcp (to allow port 8000)
  + (venv) kaushalk@photo-editor:~# sudo ufw delete allow 5000 (to disallow port 5000)
  + (venv) kaushalk@photo-editor:~# sudo ufw enable
  + (venv) kaushalk@photo-editor:~# sudo systemctl restart nginx
* Now we can get static file at our ip
  + IP adress/static/main.css
    - Will show our css file
* Now run gunicorn
  + (venv) kaushalk@photo-editor:~# gunicorn -w 3 run:app

-w is worker = (2 \* num\_cores) +1

To check num of cores

(venv) kaushalk@photo-editor:~# nporc –all

Will show you number of cores

run = name app to run

app = name of variable in app

* Now we can see our app in production server

We need to create software which runs our app

* To create that
  + (venv) kaushalk@photo-editor:~# sudo apt install supervisor
* Config
  + (venv) kaushalk@photo-editor:~# sudo /etc/supervisor/conf.d/photoeditor.conf

[program:photoeditor]

directory=/home/kaushalk/Photo\_Editor

command=/home/kaushalk/Photo\_Editor/venv/bin/gunicorn -w 3 run:app

user=kaushalk

autostart=ture

autorestart=true

stopasgroup=true

killasgroup=true

stderr\_logfile=/var/log/photoeditor/photoeditor.err.log

stdout\_logfile=/var/log/photoeditor/photoeditor.out.log

* To create log file
  + (venv) kaushalk@photo-editor:/Photo\_Editor# sudo mkdir -p /var/log/photoeditor (-p craetes direcory in chain if not exists)
  + (venv) kaushalk@photo-editor:/Photo\_Editor# sudo touch /var/log/photoeditor/photoeditor.err.log
  + (venv) kaushalk@photo-editor:/Photo\_Editor# sudo touch /var/log/photoeditor/photoeditor.out.log
* Now restart supervisor

(venv) kaushalk@photo-editor:/Photo\_Editor# sudo supervisorctl reload

To config size of file to load in nginx

* (venv) kaushalk@photo-editor:/Photo\_Editor# sudo nano /etc/nginx/nginx.conf
  + Write
    - client\_max\_body\_size 5M;
* restart server
  + sudo systemctl restart nginx