Deploying in AWS - The complete procedure:

Stage0: Contents:

Stage1: Creating an AWS instance:

- Navigate to your Aws ec2 instance dashboard.

- Click on the Launch Instance option.

- Select Ubuntu x86 from the AMIs available
- Now select the desired Instance type, in our case it c5.4xlarge

- Now select the security group **“launch-wizard-1”** from the existing ones. (if you want to customize your own options you can do so. But please make sure to have some idea on managing the requests. **Launch-wizard-1** group allows all kinds of traffic .)

- Now select the storage, you have to change the default 8GB root to 20GB. Do not add any extra EBS storage, 20GB root will be sufficient.
- If you proceed further you will be shown to generate some key-pair you can skip to the next step, key-pairs are just for tagging and identification of an instance

- You can create your own .pem file, but make sure to have that file stored somewhere locally. The pem file acts as a key to our ec2 instance. We will attach the existing .pem file **freetier.pem** along with this document you can use that as well.

- Now review the instance details and click launch. After few seconds our instance will be up and running.

Stage2: SSH and SCP into our AWS instance: (Both windows and Linux)

SSH:

- Press the following command in your terminal or command prompt.
    - ssh -i {path to .pem file} ubuntu@{public ipv4 address}

  .pem file should be locally stored in your local machine. Ipv4 address can be easily obtained from the ec2 instance dashboard by clicking on the instance id.
- If you find any error saying that the .pem file is public, execute the following command and try again the above one
    - sudo chmod 400 {path to .pem file}
- If you need the AWS doc procedure to connect , select the instance in your Ec2 console and click on **connect** button and you will be gone to the new page name **Connect to instance**, now select the **SSH client** and follow the steps.

SCP:

- Scp is used for transferring files between remote servers. In our scenario we often need to add and remove images into our server.

  The following syntax holds for transferring files from a local machine to the server.

scp -i {path to .pem file} -r {path to file you want to send} ubuntu@{public ipv4 address}:{path to the directory on server}

For example
scp -i ./Documents/freetier.pem -r ./Pictures/oil ubuntu@23.333.45.67:/home/ubuntu/backend/data/demo

Stage3: Environment Setup in AWS instance.

- Follow the steps mentioned in the backend environment setup (quickstart.pdf). You will be good to go. Please do not forget to install the mongo db server.

Stage4: Hosting our app to the web.

1. Gunicorn:

- In the virtual environment created you have to install the gunicorn module by pressing the command,

  ```
  pip install gunicorn
  ```

  Please don't forget to activate the environment created before pressing any ` commands.

- Systemd is a boot manager for Linux. We are using it to restart gunicorn if the EC2 restarts or reboots for some reason. We create a <projectname>.service file in the /etc/systemd/system folder and specify what would happen to the gunicorn when the system reboots.

  - Execute the command.
    - `sudo vim /etc/systemd/system/gbackend.service`
  - Copy and Paste these service lines into the above file. We will attach the file too.

    ```
    [Unit]
    Description=Gunicorn instance for Perceptive Analytics Model
    After=network.target

    [Service]
    User=ubuntu
    Group=www-data
    WorkingDirectory=/home/ubuntu/backend
    ExecStart=/home/ubuntu/env/bin/gunicorn --timeout 3600 --graceful-timeout 3600 --access-logfile /home/ubuntu/access.txt --error-logfile /home/ubuntu/error.txt --log-file /home/ubuntu/g_log.txt -b localhost:8000 app:app
    Restart=always

    [Install]
    WantedBy=multi-user.target
    ```

    Exit that file by clicking **ESC** and then press **:wq**

2. Nginx:

- Install the Nginx by using the commands
  - sudo apt-get update
  - sudo apt-get install nginx

- Open the default file of Nginx file by using the command
  `sudo vim /etc/nginx/sites-available/default`

  And do the following changes in the document.

  1. Add the upstream named backend and specify the localhost, (if you need why, dig into the documentation of Nginx)

     Paste these lines of code in the file above the **server** block in the file you have just opened.

     ```
     upstream backend {
         server 127.0.0.1:8000;
     }
     ```
  2. Now add the created upstream **proxy_pass** into the **location /** block should look like this.

     ```
     location / {
          # First attempt to serve request as file, then
          # as directory, then fall back to displaying a
     404.
          proxy_pass http://backend;
     }
     ```

  3. Exit that file by clicking ESC and then press :wq
  4. Now open the Nginx config file by command
     sudo vim /etc/nginx/nginx.conf
  5. Now paste these lines in that file under **Basic Settings comment**
     ```
     proxy_read_timeout 3600;
     proxy_connect_timeout 3600;
     proxy_send_timeout 3600;

     client_max_body_size 100M;
     client_body_timeout 3600;
     client_header_timeout 3600;
     ```

  6. Exit that file by clicking ESC and then press :wq

Now you have successfully set all the configurations of both gunicorn and Nginx

3. The Last Step:

Once you have completed all the steps mentioned above, you have to restart the demon and configure minor changes as mentioned below.

- `ls`

4. The verification:

Run the command

`curl localhost`

If everything was implemented as described you can see a greeting message, {login: yeah}

- In case of an error, if you need to have a look at the logs of Nginx and gunicorn
  The log files are at the below location

  1. /var/log/nginx/error.log - Error log of Nginx
  2. /var/log/nginx/access.log - Access log of Nginx
  3. /home/ubuntu/access.txt - Access log of gunicorn
  4. /home/ubuntu/error.txt - Error log of gunicorn
  5. /home/ubuntu/g_log.txt - log of processing of gunicorn (sometimes shows info about errors also)

Stage5: Attaching an Elastic IP

You can attach an elastic IP to an ec2 instance for keeping the IP address constant after quick reboots. Follow the below steps to attach an elastic IP

1. Navigate to the elastic IP which can be seen on the left dashboard of the EC2 console. (network and security section)

2. Click on the Allocate Elastic IP address on the top right corner.

3. Select amazon pool of IPv4 addresses. And click on allocate.

4. After allocating click on actions of the selected/created IP address and click on the associate elastic IP address. It will ask you for the ipv4 of existing ec2 instances.

5. Check box Allow this Elastic IP address to be reassociated and click on Associate

Basic Linux commands.

- Bash Shell Commands
  - cd
    - Description: change directory
    - Syntax : cd <path to directory>
  - ls
    - Description: list all files in the current directory
    - Syntax: ls
  - pwd
    - Description: Shows the current location of the directory
    - Syntax: pwd
  - mkdir
    - Description: make a directory
    - Syntax : mkdir <name of directory>
  - mv
    - Description: move a file or directory
    - Syntax : mv <source of file/directory> <destination>
    - Options : -r , (recursive) if you want to move a directory and all its contents ,Ex : mv -r dir1 ../dir2 moves ./dir1 into ../dir2
  - cp - Same like above mv, except it, doesn't remove the previous, but it just copies; cp = copy, mv = cut
  - rm
    - Description : Removes the file/directory
    - Syntax : rm <path to remove the file/directory>

        - Options : -rf, if you want to remove the directory,
          Ex : rm -rf <path-to-directory>

    - sudo
        - Description: Provides root administrator access when used before any command
        - Syntax : sudo <command>
        - Ex: sudo rm dir1, sudo systemctl start mongod

For more basic shell commands refer this page

- Basic Vim Commands -

Vim is a text editor opened if we open a file through the vim command

- Syntax - vim <file-name>

After entering the editor you will be in **Command mode**, this mode is where you can execute vim commands(not bash) to make some operations on the file. To go to **insert mode**, press "**i**", this mode is where you can edit the text.

Some basic ones are

- **ESC** - Switch to **Command mode**
- **i** - Switch to Insert Mode
- **:w** - save the file, only execute it in **Command mode**, if you type this in **insert mode** your file will be edited with **:w** characters at the pointer.
- **:q** - Quit the save or unchanged file, same as above execute in **Command mode**
- **:q!** - To quit the unsaved file (edited but not saved files) unlike above **:q** , execute it in **Command mode**

You can even execute the commands at a time Ex : "**:wq**" - saves the file and quits.

For more Vim commands refer this page

Managing AWS:

Starting and stopping an Instance:

Navigate to amazon aws console and select ec2. There you can see The number of instances created, number of running instances etc. Click on the instances button.

Now select the instance and you want to manipulate and click on instance Actions on the top, and then choose start or stop instance accordingly.

NOTE:
NEVER TERMINATE THE WORKING INSTANCE PA !!!!!!!.
IF you did, you have create a fresh instance again and do the environment setup

When you connect to the Instance by default you will be in **/home/ubuntu/**

Adding Images from local machine :

Here assume that you have a folder named **Bottle** with necessary Bottle images in that folder, and assume that you this folder in your Desktop

You need to move this **Bottle** folder to the /home/ubuntu/backend/data/demo/ in the Ec2

We can use scp command

scp -i <location-to-the-pem-file-of-the-instance> -r <location-to-**Bottle**> ubuntu@<Public-IP-v4-of-instance>:/home/ubuntu/backend/data/demo

Note : If you need to change the location of the class images folder from /home/ubuntu/backend/data/demo , you need to change the location in /imagedetect and /videodetect routes in app.py file.

SERVICES BEING USED:

1) AWS EC2 - C5.4xLarge :

$0.68 per hour. Please don't forget to stop the instance whenever you are not using the app.

2) AWS EBS:

$0.114 per GB-month storage === $0.114*20 = $2.28

3) AWS Elastic IP:
    - Elastic IP is actually free when connected to running instance , it only charges in these two condition :
        - When left unused or connected to instance which is not running - $0.005/hr
        - When more than one Elastic Ip is connected to the same instance - $0.005/hr