**Deploying Deep Learning Model to a Web server**

**Overview:**Deploying a deep learning model on a web server involves setting up a backend API to serve model predictions, a frontend for user intera­­ction, and NGINX to manage requests efficiently.  
**My Use case:**  
I have trained a deep learning model on the CHESTMNIST dataset, achieving 94% accuracy and deployed it to a werserver.  
 **Architecture Overview:**

* **Frontend:** NextJS (Handles user interface)
* **Backend:** Fast API (Serves the deep learning model)
* **Model:** A trained Deep learning model.
* **Containerizing:** Docker (Manages dependencies)
* **Reverse proxy:** NGINX (Handles Routes and load balancing)
* **Deployment:** Digital Ocean (cloud hosting)

**Deployment Flow**

* User uploads chest Xray via frontend. **A screenshot of a computer

  Description automatically generated**
* Frontend sends image to the fast API backend.
* Fast Api processes the image, feeds it to the training model and return the probabilities of diseases.
* NGINX acts as a reverse proxy, forwarding requests and responses securely.
* Results are displayed on the frontend.

**A screenshot of a computer

Description automatically generated**

**Steps to Deploy the Trained Deep learning Model to Web server:**

**Step 1:**

* Create Repository in GitHub. As per below screenshot there is a new button on left side for creating the repository. **A screenshot of a black screen

  Description automatically generated**
* After creating clone, the repository by below command.  
  **command: git clone ‘your repository URL’**
* Created three folders frontend, backed and nginx.
* Created a NextJS application in frontend folder by below installation guide.  
  <https://nextjs.org/docs/app/getting-started/installation>
* Run the below command to start you next JS application.  
  command: npm run build  
   npm run dev
* You will be able to see your application running on localhost, after successfully running the above commands you will be able to see the localhost link where your application is running.

**Step 2:**

* Train your Deep learning model and save the weights in .pth format.
* Create a python file in backend folder with fast API.
* Upload your weights in .pth format, you will get this weights during training of your model.
* Below is the fast Api code which I have used in my deployment.A computer screen shot of a program code

  Description automatically generated

Step 3:

* Create the docker file in frontend folder.  
  Why do we create docker file?  
  When we are deploying any application on cloud server, cloud server is nothing but a machine, as we do all required installations like eg: npm install. these also need to be done on Server side to run our application. We cant do manually like we did in our local machine. Docker play a major role here, we give set of instructions in docker file, that what libraries are required to be installed to run the application on server side. Below is an example of screenshot of how do we create a docker file.  
  A screenshot of a computer program

  Description automatically generated
* Similarly create a docker file in backend folder as below for python related libraries to be installed on server side.  
  A screen shot of a computer

  Description automatically generated

**Step 4:**

* In NGINX folder create NGINX configuration, here we tell the server how to interact the communication between client and server.

Why NGINX?  
NGINX is a reverse proxy server it has a load balancing technique, it handles the traffic very efficient way. Below is the screenshot where we declare out NGINX configuration.  
A computer screen shot of a program code

Description automatically generated

* Create a docker file in this folder, in cloud we should mention the server where our application runs.  
  **A screen shot of a computer

  Description automatically generated**

**Step 5:**

* After creating all the above 3 folders now we need to run all the three docker files step by step.
* Create a docker compose file, to run these three docker files. As below. **A screenshot of a computer program

  Description automatically generated**

**Step 6:**

* Make sure docker app is open in your local machine.
* Run the below command.  
  **command: docker-compose up –build**
* After the successful run of above command, now your application will be deployed in localhost server.
* You will be able to interact with backend services.

**Step 7:**

* Now we will push all the work which we have done to cloud, So that public will be able to use our application.
* **Go to** [**https://cloud.digitalocean.com/**](https://cloud.digitalocean.com/)
* Create a Droplet as per below screenshot.  
  **A screenshot of a computer

  Description automatically generated**
* Open the droplet which you have created and click on console on right side**.  
  A screenshot of a computer

  Description automatically generated**
* **You will be able to see a terminal page as per below screenshot.  
  A screenshot of a computer

  Description automatically generated**
* Now this droplet is nothing bot a separate machine which running on cloud.
* Generate a SSH Key by below command, this is for making connection between your github repository and your cloud machine.  
  **command: ssh-keygen -t rsa -b 4096 -C "your email"**
* Enter the below command to view your created SSH key and copy it.  
  **command: cat ~/.ssh/id\_rsa.pub**
* **Got to github settngs and add the above SSH key, as per below screen shot.  
  A screenshot of a computer

  Description automatically generated**
* **Now clone the repository in cloud machine.  
  you can get the URL to be cloned from your repository as per below screenshot.  
  A screenshot of a computer

  Description automatically generated  
  command: git clone ‘url’**
* **After successful clone redirect to that folder where you have cloned as per below screenshot.  
  A computer screen with white text

  Description automatically generated**
* **Now run the below command.**Make sure docker is installed in your cloud. Do the below command before cloning.  
  sudo apt update

sudo apt install docker-ce docker-ce-cli containerd.io -y

After the above command enter the below command. **command: docker-compose up –build**

* After build is success full you will be able see your application running on your droplet ip address, you can find your droplet IP address from below. **A screenshot of a computer

  Description automatically generated**
* Go to chrome and run http:// + ipv4 in above mentioned screenshot.