Deployment

Deployment:: Introduction to Production

We will be deploying the project on the cloud.

- Cloud is similar to the machine that we are using for development right now, it's just that it is located somewhere remotely and it has internet access with some openly available ports so that people from across the world access it.
- A Cloud machine is a machine that can be accessed publicly.
- Various service providers give us a machine-like cloud { AWS, GCP, Heroku }
- We will be studying AWS for now.

Steps we will be following for deployment.

- We will be purchasing or using a free tier machine and set it up.
- We will log in to the machine and install tools when required.
- We will pull our code that we have pushed either on Github, bitbucket, GitLab, etc.
- We will install the libraries and run them.
- We have to change/add the OAuth URL for the google dashboard.
- We have to link our **Codeial app** to the cloud machine to be publicly available.

EXTRA:
You can check out the link below to understand more about AWS -
https://docs.aws.amazon.com/

AWS:: Introduction + Launching Our First Machine

- We are going to use Amazon Web Services for purchasing or getting our first cloud machine, setting the code up, and making the website go live in production mode.
- Services - EC2 -> A computer that is available to us remotely with one or multiple harddisks attached to it. ECC { Elastic Cloud Computing } / RDS - {Relational Database Service } / S3 - { Simple Storage Service } A folder that is available to us remotely so that we can store the uploaded files separately / EMR - { Elastic Map Reduce } It is a tool for big data processing.
- T2 instances are a low-cost general-purpose instance type that provides a baseline level of CPU performance.
- Create a new key pair or the existing one for launching the instances.
- Key pair is the only access to the machine right now.

***EXTRA*: AWS has an infinite set of services. You can look at those services from the link below -**
https://aws.amazon.com/

***NOTE*: Key pair should be preserved as losing this will also let you lose access to the remote machine.**

SSH:: Setting up the Machine

- We need to access the machine and install the tools that are required.
- We will log in to the remote machine using our terminal.
- We need to permit the key that we have downloaded to be able to access the remote machine. The way to achieve this is using SSH { Secured shell }.
- To give permission, we need to enter the given command in the terminal (chmod 600 - {path of the downloaded key }).

- We have to log in to the remote machine using the command (ssh -i {file_path} machine_name along with IP address) in the terminal.
- We now need to install multiple things on the remote machine like { Node.js, Redis Server, Nginx, pm2}.
- We need to install the particular version of the node because the project we are about to deploy requires an older version hence for that, we need to install NVM.
- To exit the machine, we can use the command **exit** on the terminal.
- Use the following command to install NVM - { nvm install version_no }.
- We need to install MongoDB on the remote server along with its packages.
- We need to install the Redis server on the remote machine.
- We need to install the NGINX on the remote machine { NGINX is a remote proxy server }.
- PM2 - Whenever our app crashes in nodemon, the code execution gets stuck at that point. Although, in the case of PM2 if the app crashes it will attempt to restart the app. Hence, PM2 is good for production.
- We need to install PM2 on the remote server.

Pulling the Code

- We need to pull the code and clone the repository of the code to the main server.
- Clone the repository using the command { **git clone (URL)** } in the terminal.
- To fetch the packages automatically from the **package.json** using the command { npm install } in the terminal.
- We need to install Gulp to be able to access the app.
- We need to create the bash profile.
- We need to run the command { **gulp build** } in the terminal to access the app.
- We need to install nodemon using the command { **npm install -g nodemon** }.

- We have to export port 8000 to be open inside the security groups.

NGINX

- NGINX does reverse proxy. { Proxy means whenever we are accessing a machine there is a proxy server with address (a) and address of the machine is (b), we are accessing (a) and this (a) is giving access to (b). So here (a) that is NGINX is accessing the (b) that is node.js }.
- We cannot access server (b) directly.
- We need to configure the NGINX using the command { vi nginx.conf } in the terminal.
- NGINX will be listening on port 80 and it will be responding to requests from there. It will be sending forward all those requests to the Node.js server.
- NGINX code is present inside the { **etc }** folder.

Masking the Domain On Our IP

- We are now going to link the domain name { cordeial.com } with cloud AWS.
- We can use any host provider { Go Daddy } that you want to buy the domain name.

Summarizing it All

- We bought a machine from AWS for free.
- We installed all the libraries and packages { node.js, Redis, MongoDB, pm2, Nginx }that we need.
- We pulled the code.
- We installed the npm libraries.

- We figured out the Nginx server and linked the domain name to point to the Nginx server to the machine's IP address.

***EXTRA*: Style it better.**