**Configuration for 2 tier application**  
  
1. Launched aws instance with slight bigger space as it holds multiple software’s like Jenkins, Docker, aws cli setup, terraform (only executes if time permits) and I have done it manually through console and same can be done through aws cli and while launching the instance I can also use user data section to just to install all these software’s just using this for now.  
  
2. Install Software’s but before installation update the instance so it will take all the necessary updates for your software’s and installations. (Better connect with fast internet)

3. After software’s installed setup the Jenkins Controller server and updates the plugins and credentials needed for the applications to run from git .

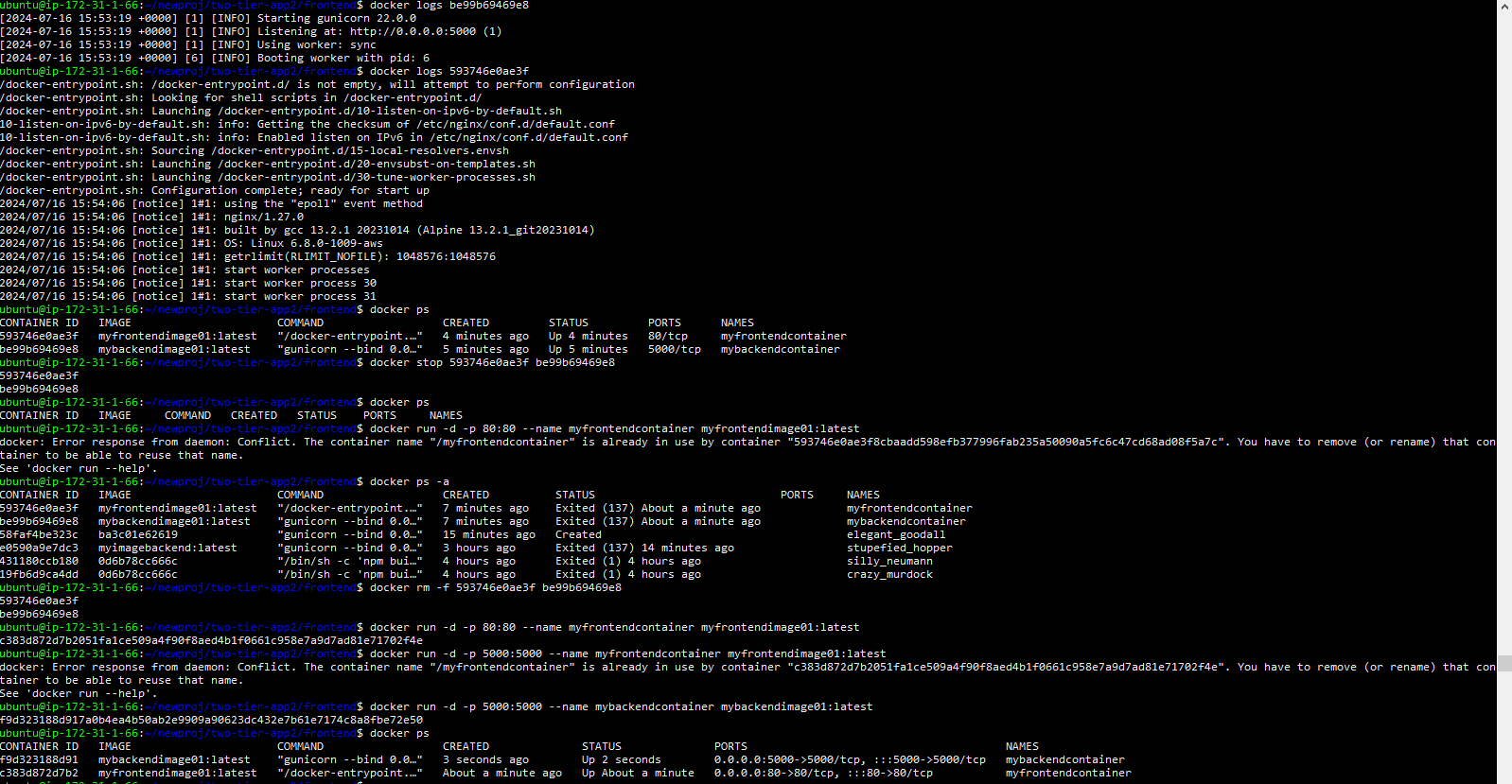
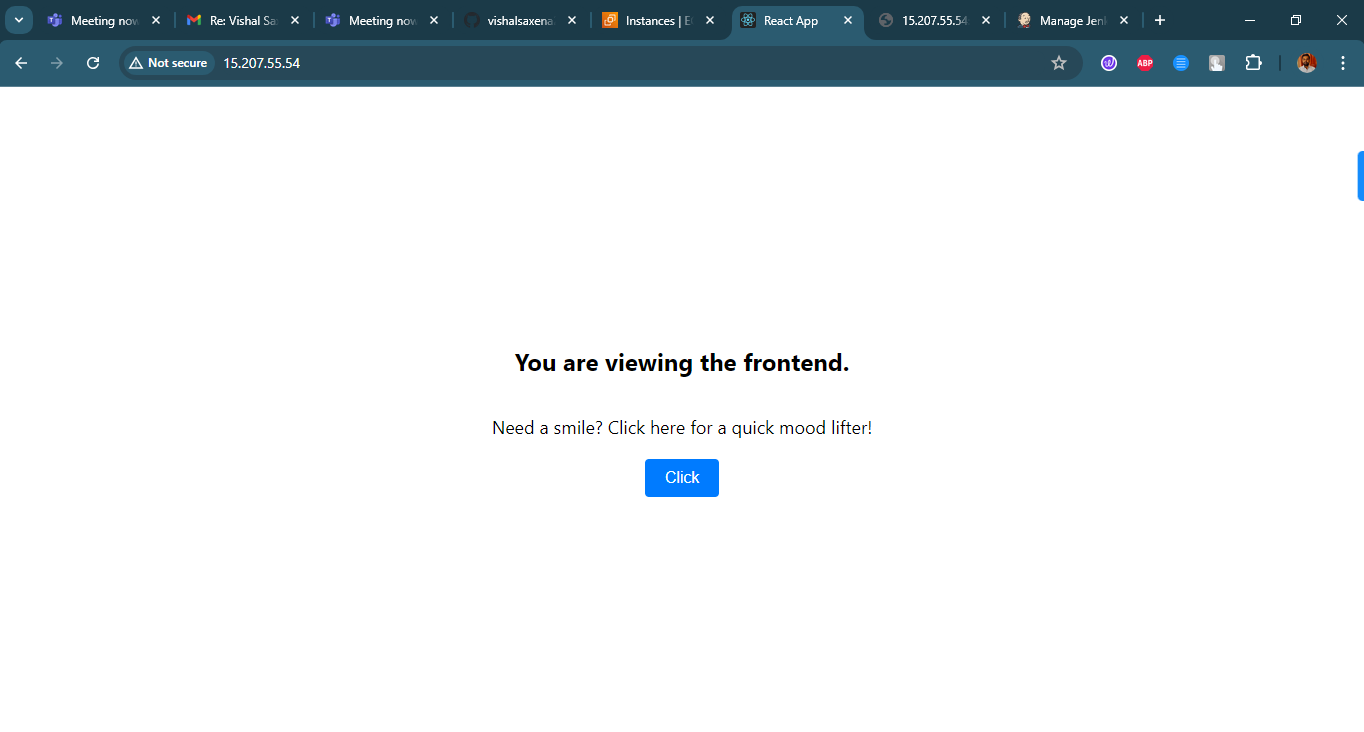
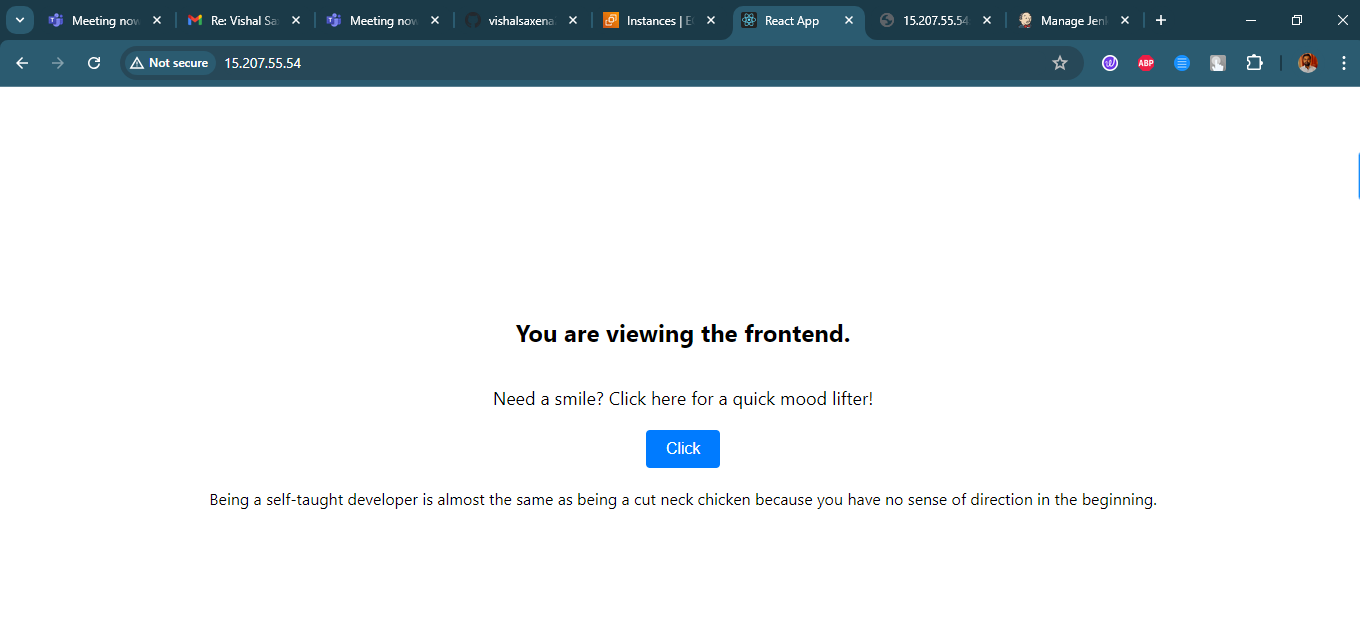
4. After all this setup create a pipeline job on your Jenkins and run using agents or run on the controller machine as I have bigger machine I am running in my controller machine itself but it’s not advice to run it on the **controller the best choice is to bit the docker image and use it as agent** which can be efficient I am not doing here due to the time constraint only.

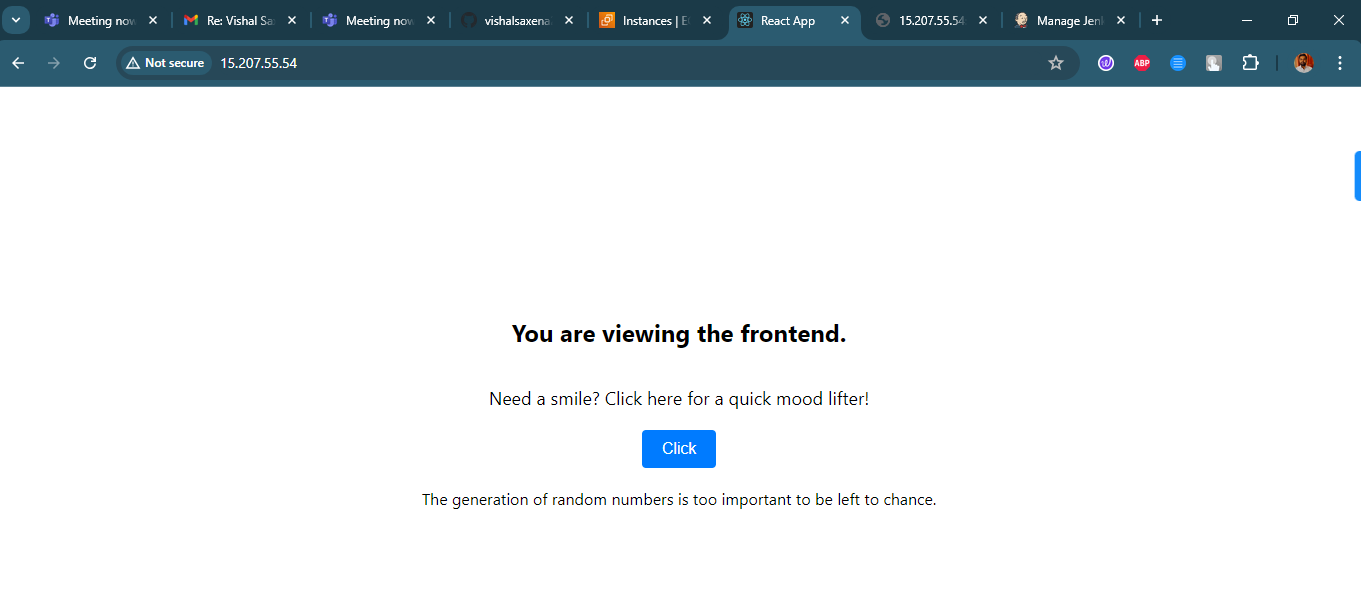
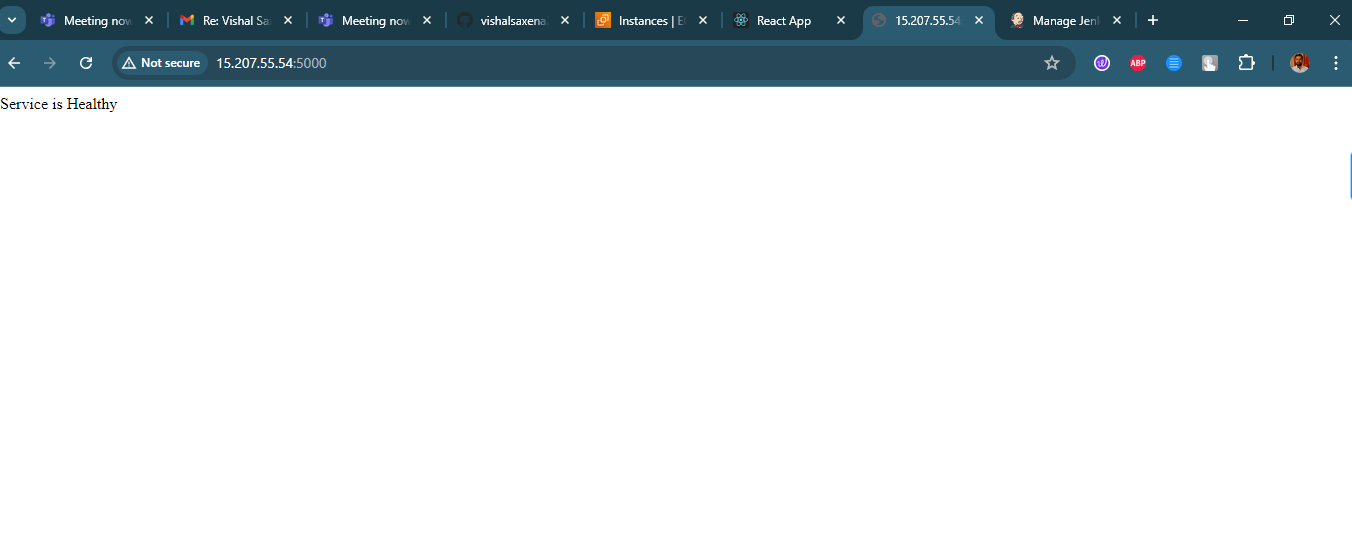
5. And yes make sure to open the necessary ports to run Jenkins.

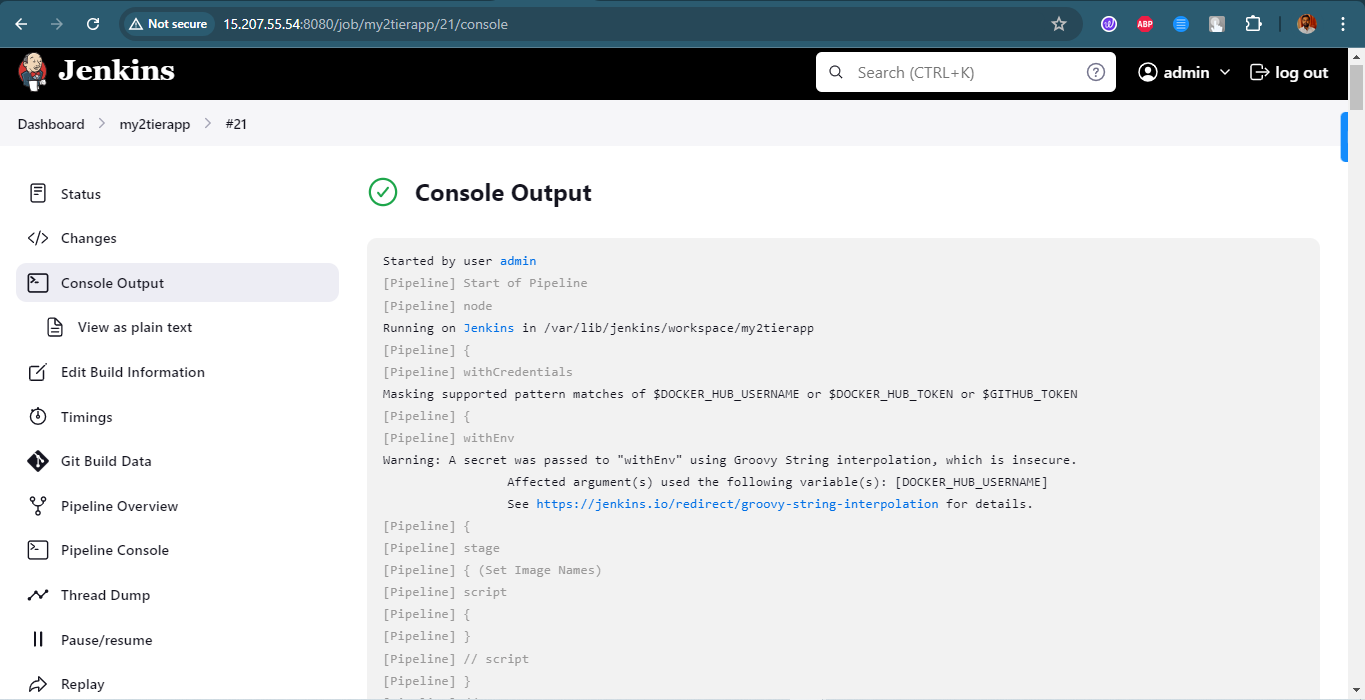
Jenkins url should look like this http://15.207.55.54:8080/  
get the pass from /var/lib/jenkins/secrets/initialAdminPassword  
**sudo cat /var/lib/jenkins/secrets/initialAdminPassword  
  
http://15.207.55.54:8080/  
  
Below is the git repo for the application fork it in your repo or you can create your own repo and push the code that’s an option.**https://github.com/vishalsaxena29/2TierReactApp-project.git **After installing plugin make sure to restart the Jenkins server to make all the settings work properly.**

Meanwhile clone the application code repo   
  
**Best practice is to run the application in your dev environment first just to check the functionality in my case I am running on my local system just to check how the application works , endpoints and other details.**

So as per my analysis the backend code runs on flask, python and with other dependencies and the endpoint which is showing in the app is first / and /api/getJoke   
  
test the front end application it’s a react app with npm

Test the docker images on your local aws ec2 controller before deploying it to aws eks cluster You can now view frontend in the browser.  


  
 You can now view Backend in the browser too.  
  
  
**Will share both the dockerfiles along with the project document**

Here is the jenkinsfile successfully build screenshot  


Jenkins File is attached

# For creating the infra you can use the terraform files which is attached to the email or you can use the following aws cli commands Following is the command based approach to create aws eks cluster IAM

Create a user “eks-admin” with AdministratorAccess

Create Security Credentials Access Key and Secret access key

# EC2

Create an ubuntu instance (region us-west-2)

ssh to the instance from local

### Install AWS CLI v2

curl "https://awscli.amazonaws.com/awscli-exe-linux-x86\_64.zip" -o "awscliv2.zip"

sudo apt install unzip

unzip awscliv2.zip

sudo ./aws/install -i /usr/local/aws-cli -b /usr/local/bin --update

### Setup your access by

aws configure

### Install Docker

sudo apt-get update

sudo apt install docker.io

docker ps

sudo chown $USER /var/run/docker.sock

### Install kubectl

curl -o kubectl <https://amazon-eks.s3.us-west-2.amazonaws.com/1.19.6/2021-01-05/bin/linux/amd64/kubectl>

chmod +x ./kubectl

sudo mv ./kubectl /usr/local/bin

kubectl version --short --client

### Install eksctl

curl --silent --location "https://github.com/weaveworks/eksctl/releases/latest/download/eksctl\_$(uname -s)\_amd64.tar.gz" | tar xz -C /tmp

sudo mv /tmp/eksctl /usr/local/bin

eksctl version

### Setup EKS Cluster

eksctl create cluster --name three-tier-cluster --region us-west-2 --node-type t2.medium --nodes-min 2 --nodes-max 2

aws eks update-kubeconfig --region us-west-2 --name three-tier-cluster

kubectl get nodes

### Run Manifests

kubectl create namespace two-tier-ns

kubectl apply -f .

Kubectl delete -f .

eksctl delete cluster --name my-cluster --region us-west-2

### Install AWS Load Balancer

curl -O <https://raw.githubusercontent.com/kubernetes-sigs/aws-load-balancer-controller/v2.5.4/docs/install/iam_policy.json>

aws iam create-policy     --policy-name AWSLoadBalancerControllerIAMPolicy     --policy-document file://iam\_policy.json

eksctl utils associate-iam-oidc-provider --region=us-west-2 --cluster=my-cluster --approve

eksctl create iamserviceaccount   --cluster=my-cluster   --namespace=kube-system   --name=aws-load-balancer-controller   --role-name AmazonEKSLoadBalancerControllerRole   --attach-policy-arn=arn:aws:iam::626072240565:policy/AWSLoadBalancerControllerIAMPolicy --approve --region=us-west-2

sudo snap install helm --classic

helm repo add eks <https://aws.github.io/eks-charts>

helm repo update eks

helm install aws-load-balancer-controller eks/aws-load-balancer-controller   -n kube-system   --set clusterName=my-cluster   --set serviceAccount.create=false   --set serviceAccount.name=aws-load-balancer-controller

kubectl get deployment -n kube-system aws-load-balancer-controller

kubectl apply -f full\_stack\_lb.yaml