DevOps Certification Training

CASE STUDY - INTRODUCTION TO KUBERNETES

You have just joined a startup Ventura Software as a Devops Lead Engineer. The company relies on a Monolithic Architecture for its product. Recently, the senior management was hired. The new CTO insists on having a Microservice Architecture. The Development Team, is working on breaking the Monolith. Meanwhile, you have been asked to host a Test Application on Kubernetes, to understand how it works.

Following things have to be implemented:

1. Deploy an Apache2 deployment of 2 replicas
2. Sample code has been checked-in at the following Git-Hub repo:

https://github.com/hshar/website.git.

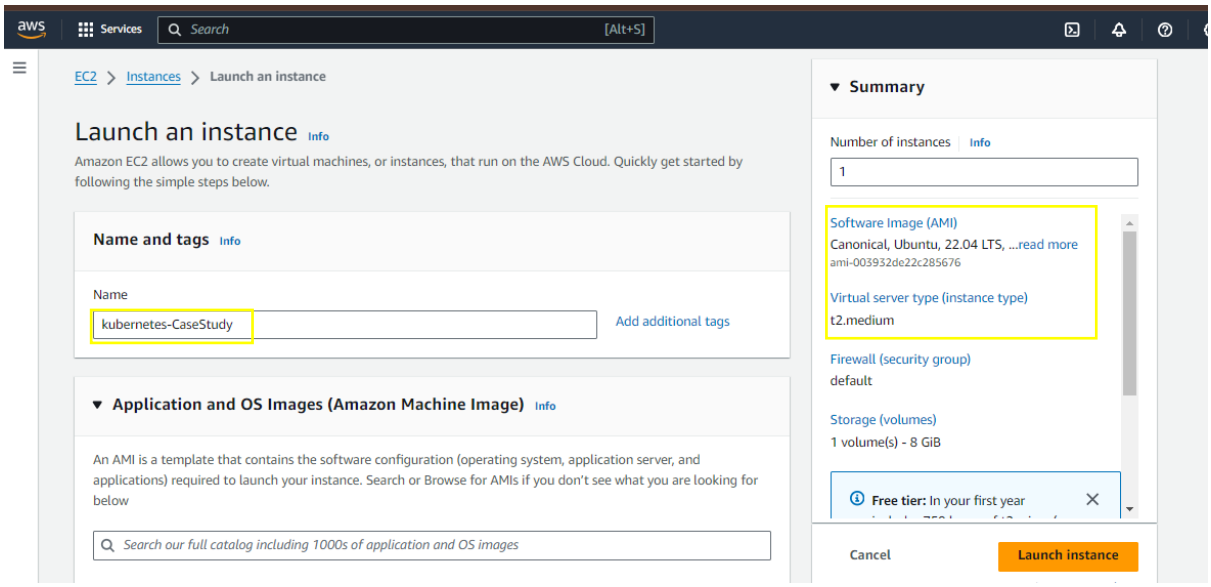
You have to containerize this code, and push it to Docker Hub. Once done, deploy it on Kubernetes with 2 replicas

3. Deploy Ingress with the following rules:

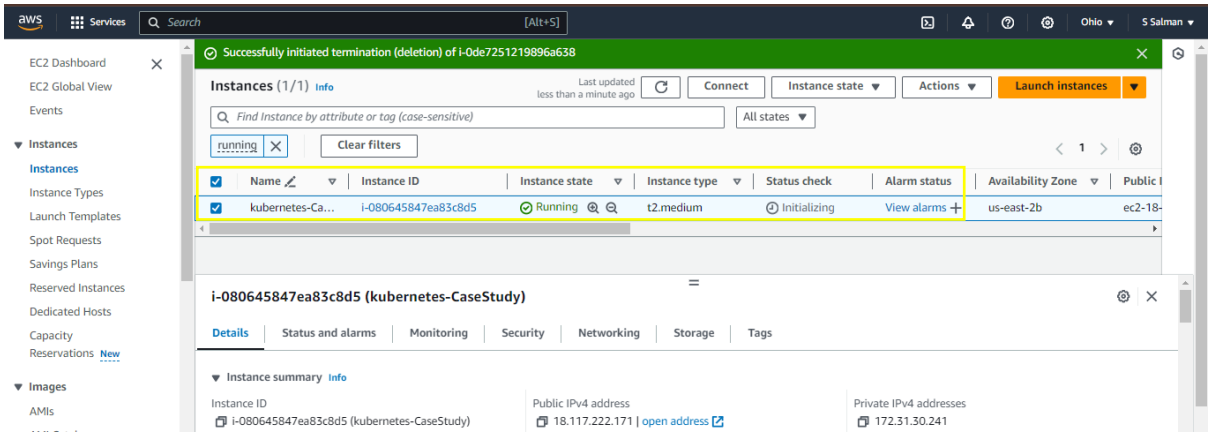
i) */apache* should point to the apache pods

ii) */custom* should point to the GitHub application

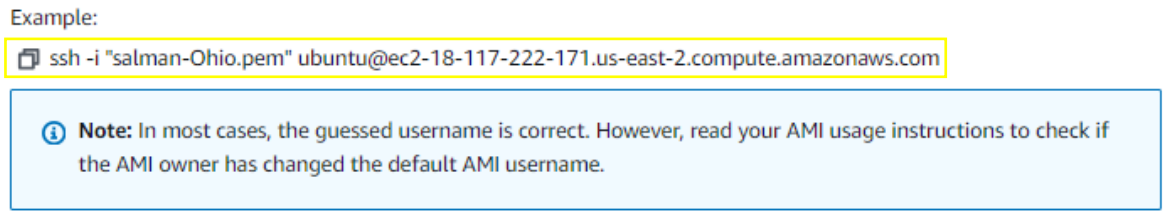
Launch t2.medium instances for Kubernetes Assignment



Click on **connect**



Copy it




```
C:\Users\shaik\Downloads>ssh -i "salman-Ohio.pem" ubuntu@ec2-18-117-222-171.us-east-2.compute.amazonaws.com
The authenticity of host 'ec2-18-117-222-171.us-east-2.compute.amazonaws.com (18.117.222.171)' can't be established.
ED25519 key fingerprint is SHA256:f+PLENgXZSRdc0/pW7xMguEVxCf45BlviopmsxQyliA.
This key is not known by any other names
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added 'ec2-18-117-222-171.us-east-2.compute.amazonaws.com' (ED25519) to the list of known hosts.
Welcome to Ubuntu 22.04.4 LTS (GNU/Linux 6.5.0-1022-aws x86_64)
```

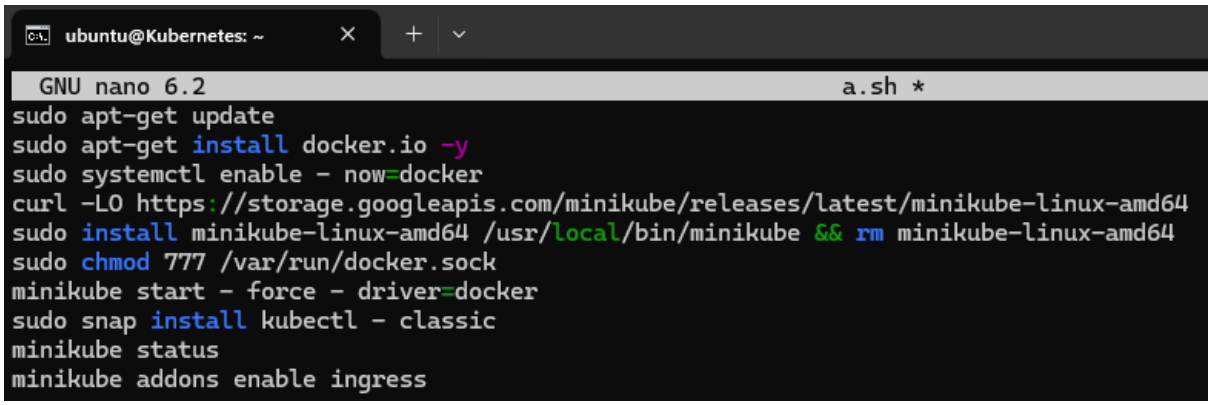
```
C:\WINDOWS\system32\cmd.
ubuntu@ip-172-31-30-241:~$ sudo hostnamectl set-hostname Kubernetes
ubuntu@ip-172-31-30-241:~$ exit
logout
Connection to ec2-18-117-222-171.us-east-2.compute.amazonaws.com closed.

C:\Users\shaik\Downloads>ssh -i "salman-Ohio.pem" ubuntu@ec2-18-117-222-171.us-east-2.compute.amazonaws.com
```

```
ubuntu@Kubernetes: ~
ubuntu@Kubernetes:~$ sudo apt-get update
```

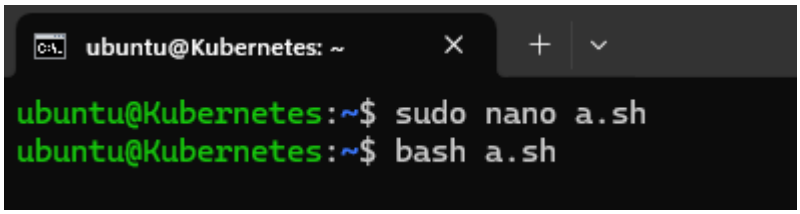
Script for installing dockers, Kubernetes, minikube

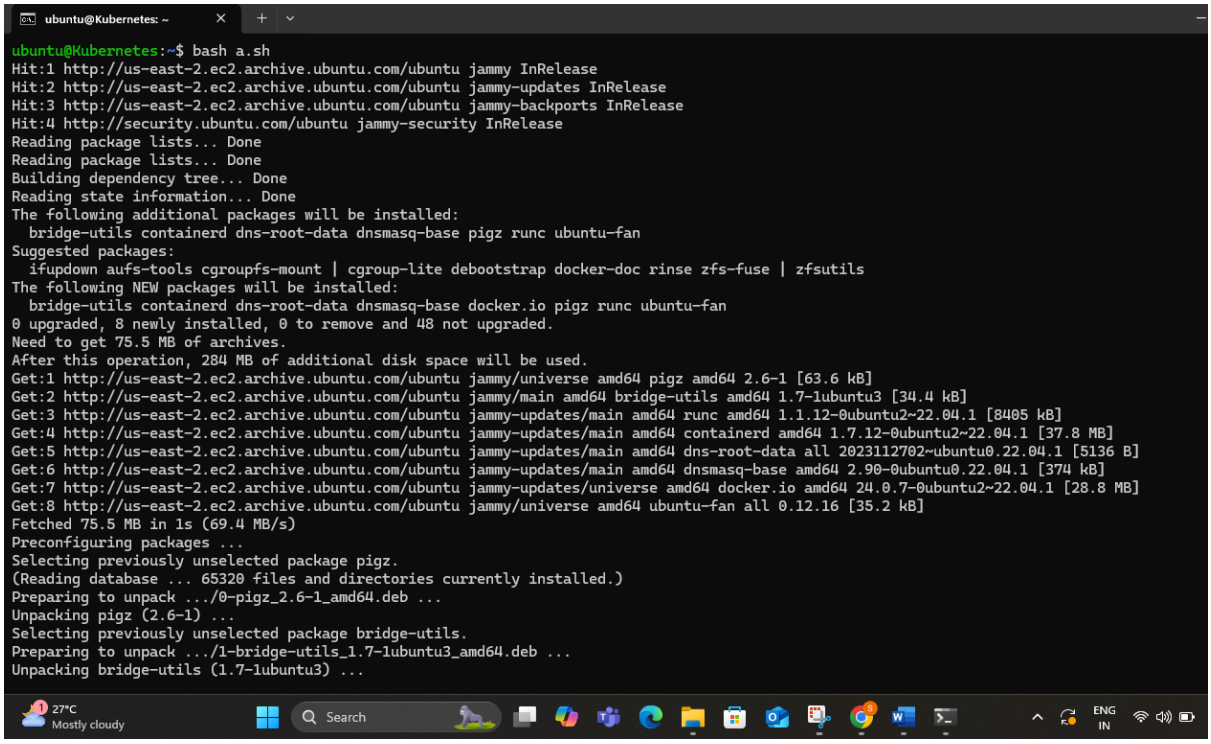
```
ubuntu@Kubernetes: ~
ubuntu@Kubernetes:~$ sudo nano a.sh
```

Script for installing dockers, Kubernetes, minikube



Execute the file





```
100 91.1M  100 91.1M    0     0  89.9M      0  0:00:01  0:00:01 --:--:-- 89.9M
😄  minikube v1.33.1 on Ubuntu 22.04 (xen/amd64)
✨  Automatically selected the docker driver. Other choices: none, ssh
📌  Using Docker driver with root privileges
👍  Starting "minikube" primary control-plane node in "minikube" cluster
🚜  Pulling base image v0.0.44 ...
💾  Downloading Kubernetes v1.30.0 preload ...
    > preloaded-images-k8s-v18-v1...:  342.90 MiB / 342.90 MiB  100.00% 73.19 M
    > gcr.io/k8s-minikube/kicbase...:  481.58 MiB / 481.58 MiB  100.00% 38.14 M
🔥  Creating docker container (CPUs=2, Memory=2200MB) ...
🐳  Preparing Kubernetes v1.30.0 on Docker 26.1.1 ...
    ▪ Generating certificates and keys ...
    ▪ Booting up control plane ...
    ▪ Configuring RBAC rules ...
🔗  Configuring bridge CNI (Container Networking Interface) ...
🔎  Verifying Kubernetes components...
    ▪ Using image gcr.io/k8s-minikube/storage-provisioner:v5
🌟  Enabled addons: storage-provisioner, default-storageclass
💡  kubectl not found. If you need it, try: 'minikube kubectl -- get pods -A'
🏄  Done! kubectl is now configured to use "minikube" cluster and "default" namespace by default
error: cannot install "kubectl", "-", "classic": invalid instance name: invalid snap name: "-"
minikube
type: Control Plane
host: Running
kubelet: Running
apiserver: Running
kubeconfig: Configured

💡  ingress is an addon maintained by Kubernetes. For any concerns contact minikube on GitHub.
You can view the list of minikube maintainers at: https://github.com/kubernetes/minikube/blob/master/OWNERS
    ▪ Using image registry.k8s.io/ingress-nginx/controller:v1.10.1
    ▪ Using image registry.k8s.io/ingress-nginx/kube-webhook-certgen:v1.4.1
    ▪ Using image registry.k8s.io/ingress-nginx/kube-webhook-certgen:v1.4.1
🔎  Verifying ingress addon...
```

```
💡  ingress is an addon maintained by Kubernetes. For any concerns contact minikube on GitHub.
You can view the list of minikube maintainers at: https://github.com/kubernetes/minikube/blob/master/OWNERS
    ▪ Using image registry.k8s.io/ingress-nginx/controller:v1.10.1
    ▪ Using image registry.k8s.io/ingress-nginx/kube-webhook-certgen:v1.4.1
    ▪ Using image registry.k8s.io/ingress-nginx/kube-webhook-certgen:v1.4.1
🔎  Verifying ingress addon...
🌟  The 'ingress' addon is enabled
ubuntu@Kubernetes:~$
```

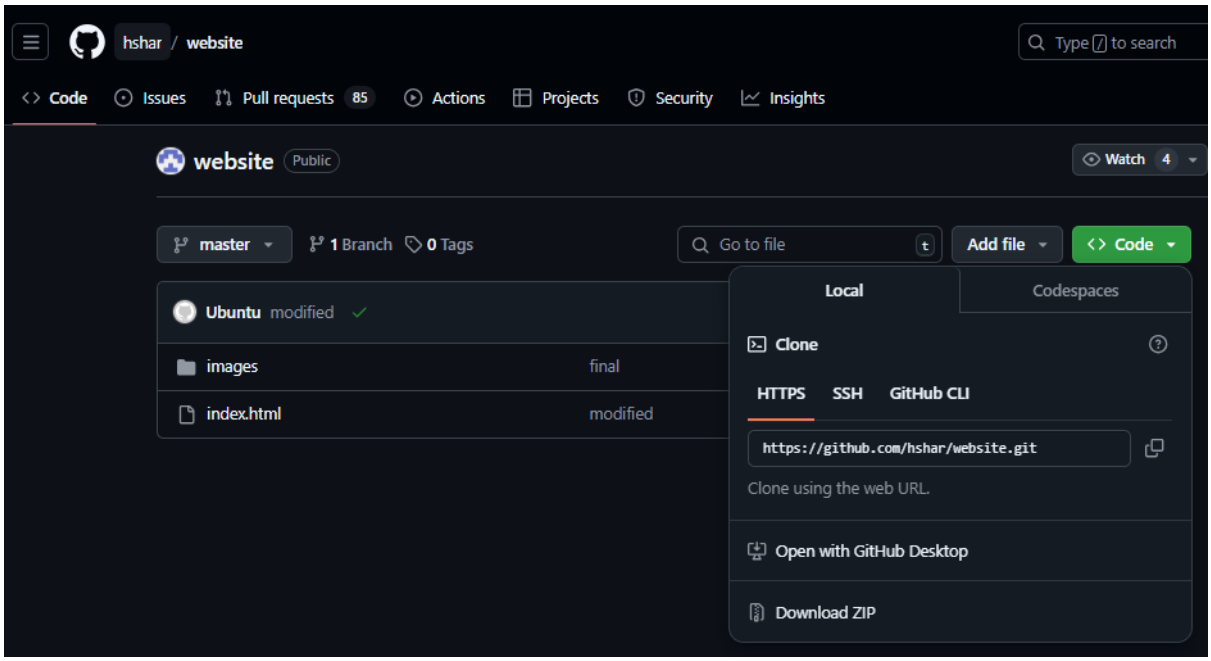
Its not installed kubectl so we need to install

```
ubuntu@Kubernetes: ~
ubuntu@Kubernetes:~$ sudo snap install kubectl --classic
kubectl 1.30.4 from Canonical✓ installed
ubuntu@Kubernetes:~$
```

For finding the **nodes, run** the **below-given command**

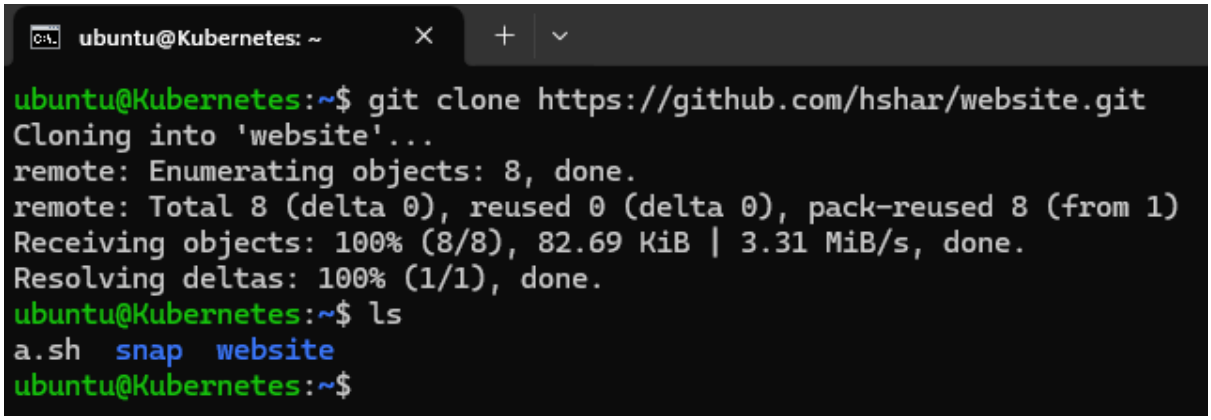
```
ubuntu@Kubernetes: ~
ubuntu@Kubernetes:~$ kubectl get nodes
NAME       STATUS   ROLES           AGE   VERSION
minikube   Ready    control-plane   10m   v1.30.0
ubuntu@Kubernetes:~$
```

Clone this **repository** in the **“Git Hub Account”. Click** on **“Code” & Copy the URL**



Type the **below-given command** in the **EC2 Instance** & **press enter** from the **keyboard**

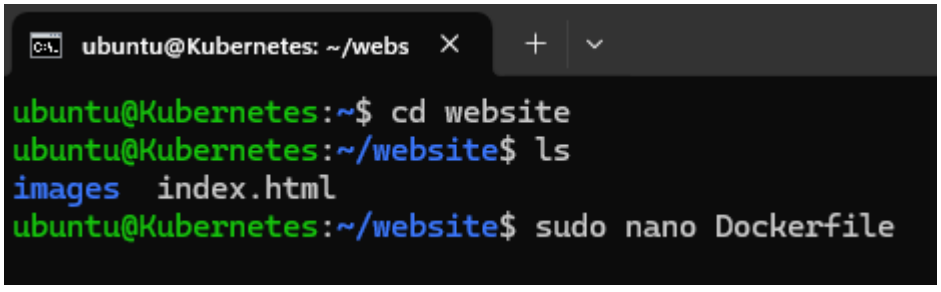
It will be **started cloning** the **website** from **Git.**



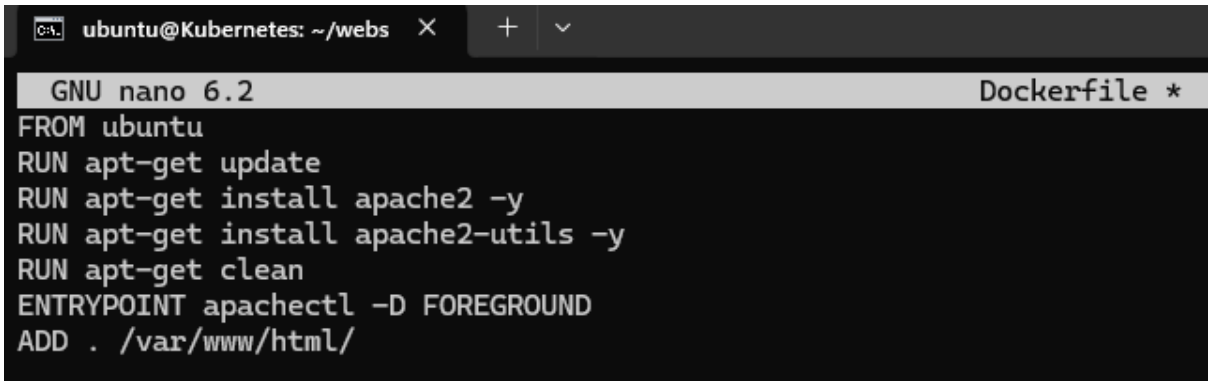
Type the **below-given command** to **show how many directories** are **present**

Write a Docker file & Push the Image to Docker Hub

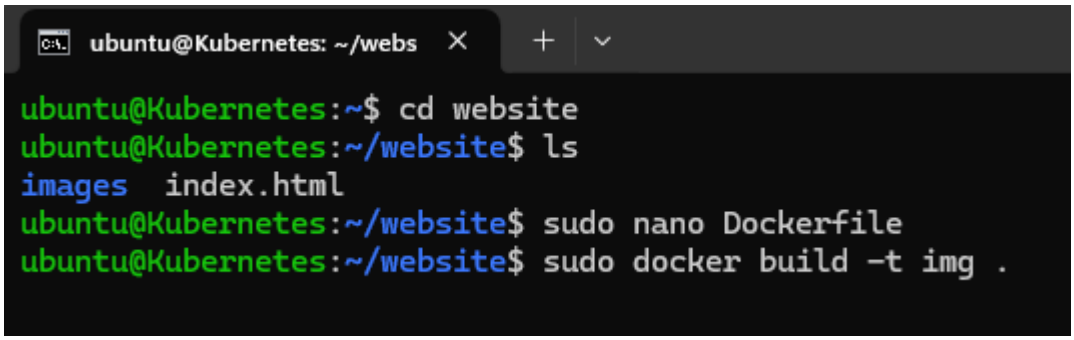
Type the **below-given command** to **create** a **Dockerfile**



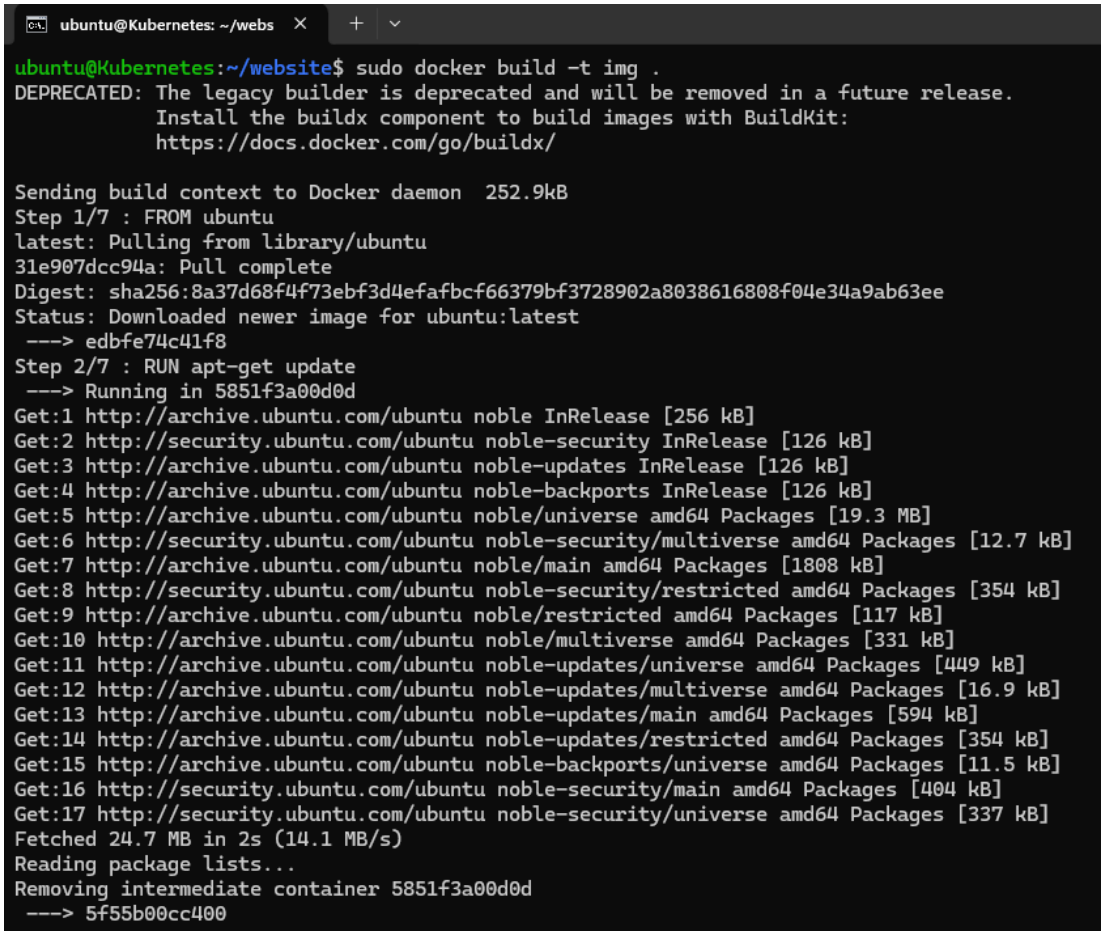
Type these **commands** to **run "apache2"** in the **containerized environment**



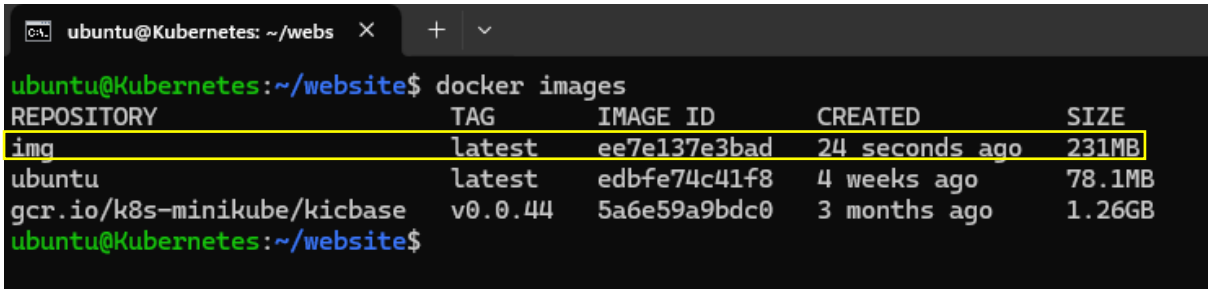
Run the **below-given command** to **build** an **image from** the **Dockerfile**



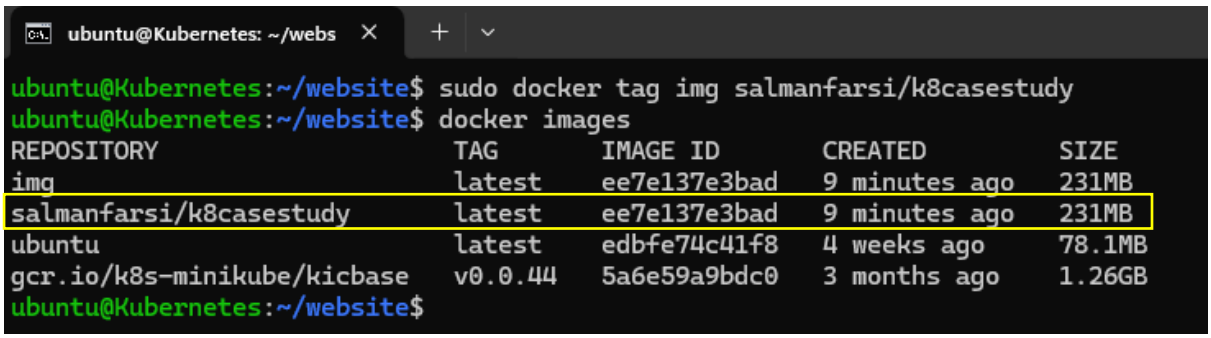
Run the below-given command to build an image from the Dockerfile



The image will be **successfully created. Now run** the **below-given command** to **view** the **images** in the **website directory**



Use the **below-given command** to **use docker credentials** the **image**

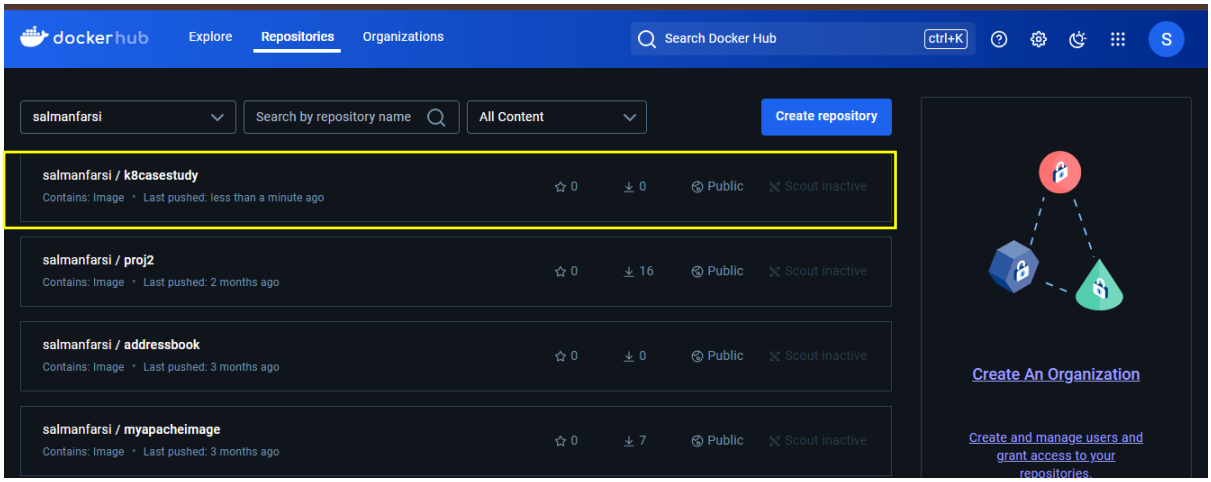


Docker Login

```
ubuntu@Kubernetes: ~/webs
ubuntu@Kubernetes:~/website$ docker login
Log in with your Docker ID or email address to push and pull images from Docker Hub. If you don't have a Docker ID, head
 over to https://hub.docker.com/ to create one.
You can log in with your password or a Personal Access Token (PAT). Using a limited-scope PAT grants better security and
 is required for organizations using SSO. Learn more at https://docs.docker.com/go/access-tokens/

Username: salmanfarsi
Password:
WARNING! Your password will be stored unencrypted in /home/ubuntu/.docker/config.json.
Configure a credential helper to remove this warning. See
https://docs.docker.com/engine/reference/commandline/login/#credentials-store

Login Succeeded
ubuntu@Kubernetes:~/website$
```

Pushing the Image to Docker Account

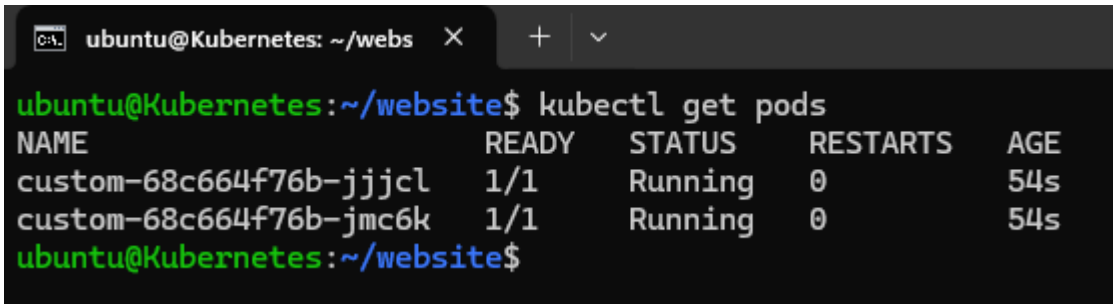
```
ubuntu@Kubernetes: ~/webs
ubuntu@Kubernetes:~/website$ docker push salmanfarsi/k8casestudy
Using default tag: latest
The push refers to repository [docker.io/salmanfarsi/k8casestudy]
8f79e5cf4fdb: Pushed
18d24e588286: Pushed
aef3e38a5bf7: Pushed
fd3314f966b4: Pushed
dccb0abec538: Pushed
f36fd4bb7334: Pushed
latest: digest: sha256:5ef5079a685bbb013d11981fb057cd75c1ff8c65b7adc6944b5dc0446bf1193b size: 1577
ubuntu@Kubernetes:~/website$
```



Run the below-given **command** to **create the custom image** with **2 replicas**

```
ubuntu@Kubernetes: ~/webs
ubuntu@Kubernetes:~/website$ kubectl create deployment custom --image=salmanfarsi/k8casestudy --replicas=2 --port=80
deployment.apps/custom created
ubuntu@Kubernetes:~/website$ kubectl get deploy -o wide
NAME     READY   UP-TO-DATE   AVAILABLE   AGE   CONTAINERS    IMAGES                      SELECTOR
custom   2/2     2            2           22s   k8casestudy   salmanfarsi/k8casestudy     app=custom
ubuntu@Kubernetes:~/website$
```

A Custom Deployment will be **created** with **Two Pods**

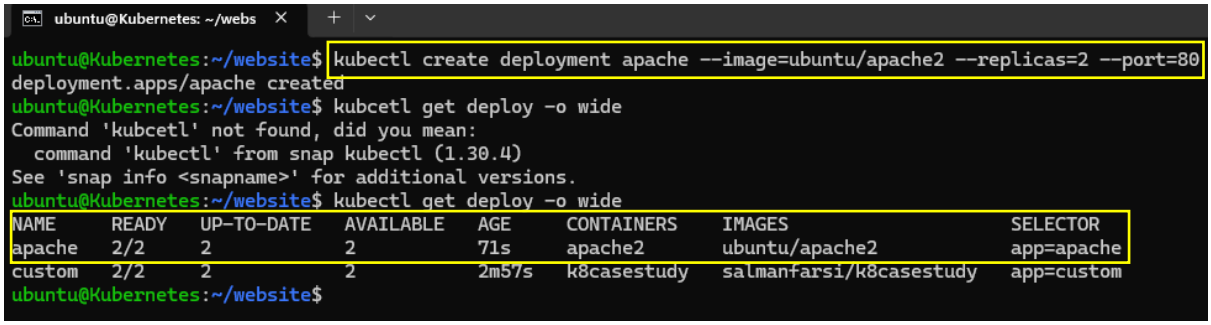


Run the below-given **command** to **create** the **deployment** with **2 replicas:**

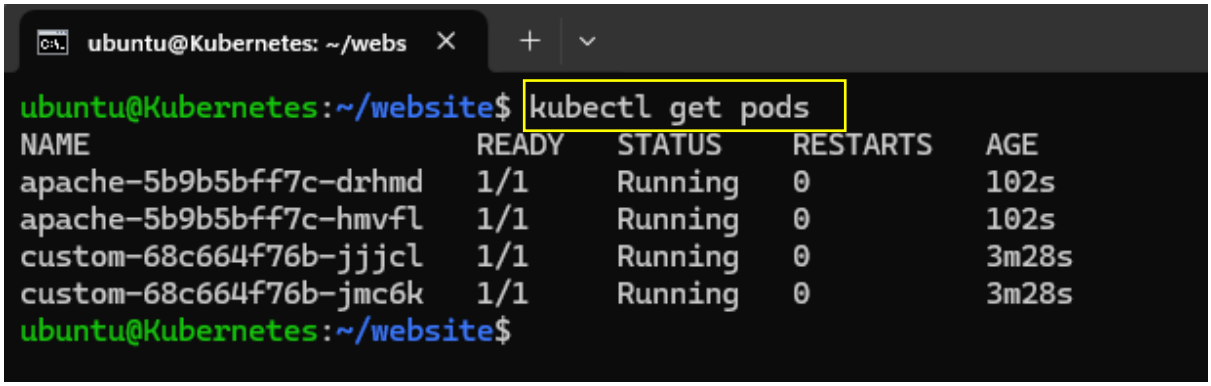
A deployment will be **successfully created** on **port 80.**

When you run the **below-given commands**

Two Apache pods will be **successfully created**



When you run the **below-given command**



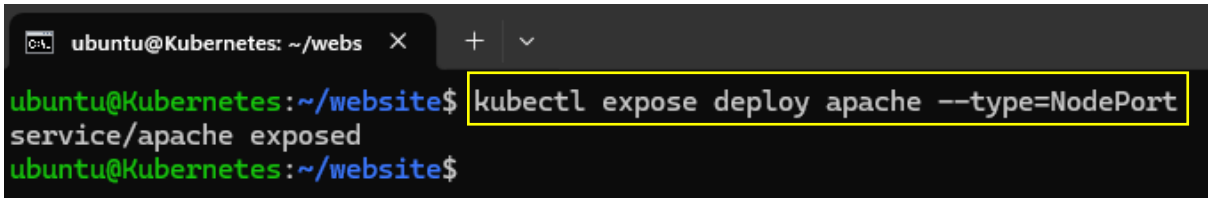
a. */apache* should point to the Apache pods

b. */custom* should point to the GitHub application.

Follow these steps;

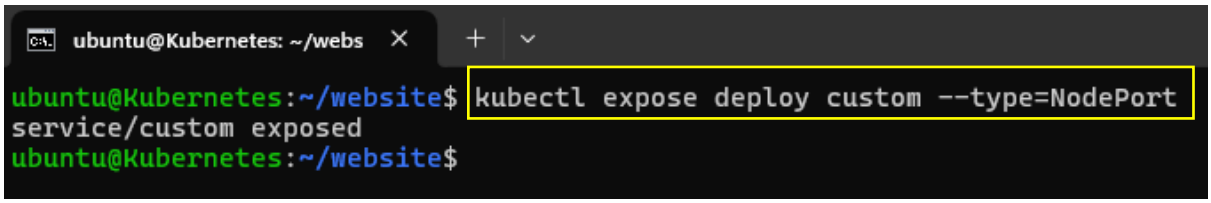
A. Expose Both the Replicas on "NodePort" for Creating a Service

First, we will **expose "apache"** on **"Node Port". Run** the **below-given command**



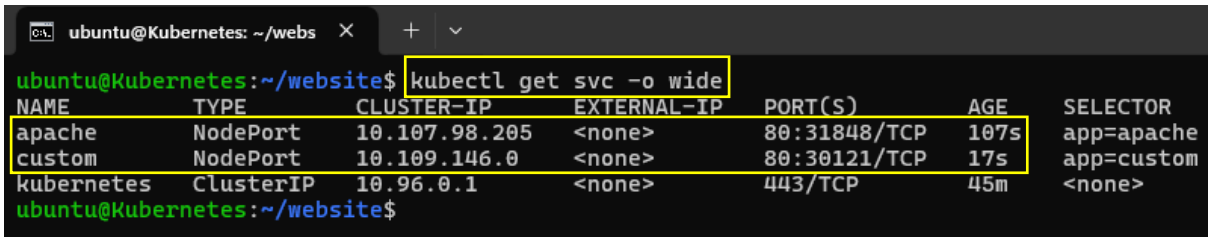
The apache service has been **successfully created** on **"NodePort".**

Second, we will **expose** the **"custom" on "NodePort". Use** the **below-given command**



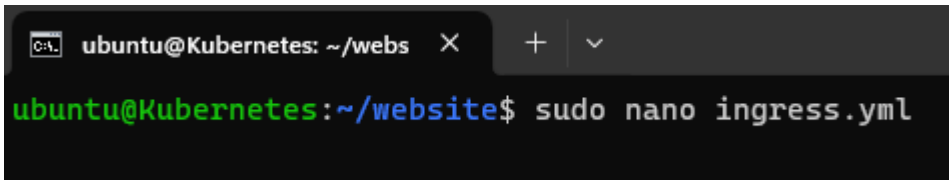
Run the **below-given command** to **find** the **service**

Both services are **successfully exposed** on **"NodePort**

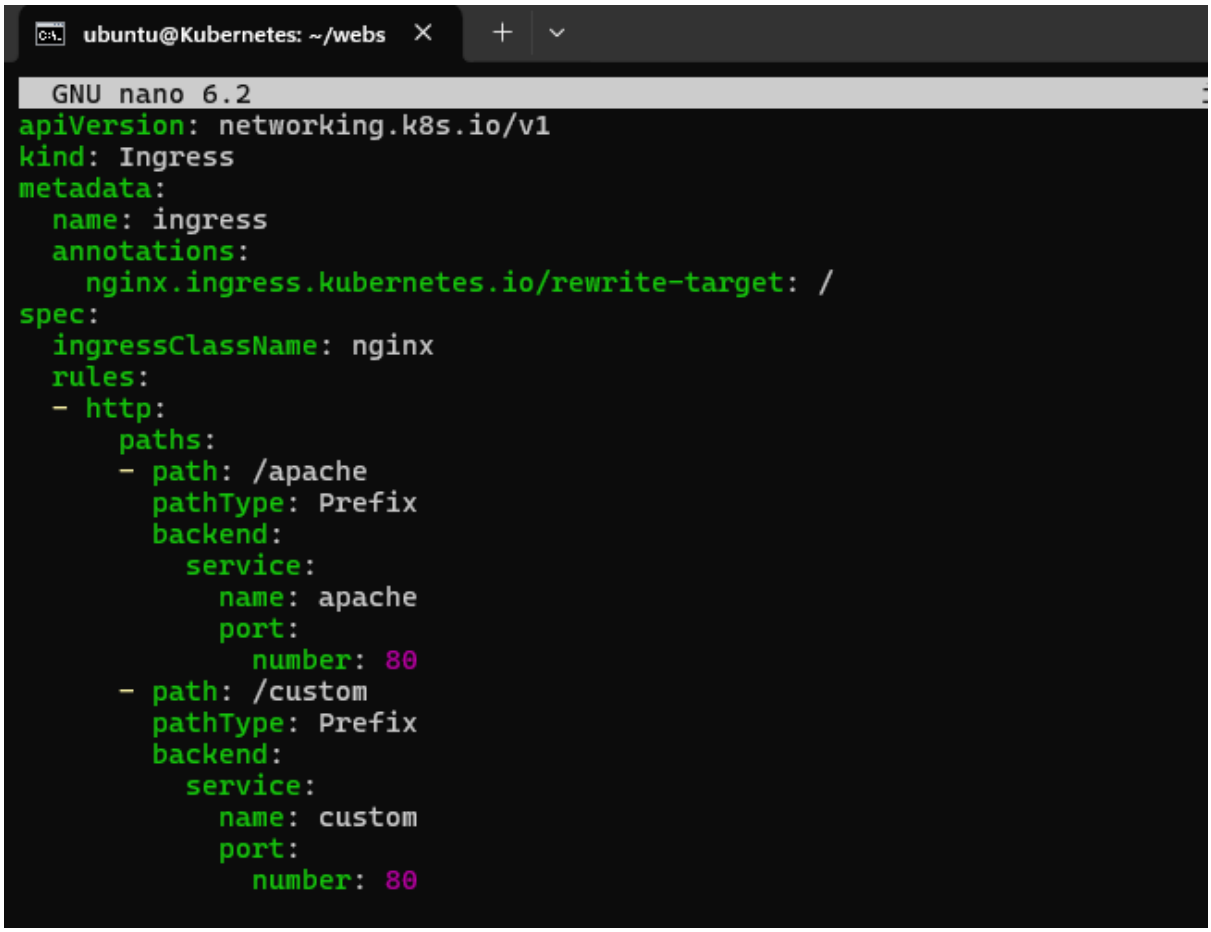


Create an ingress.yml file to point out the paths

Create an **"ingress.yml" file using** the below-given **command**



Paste the **below-given content** into the **“ingress.yml”** file

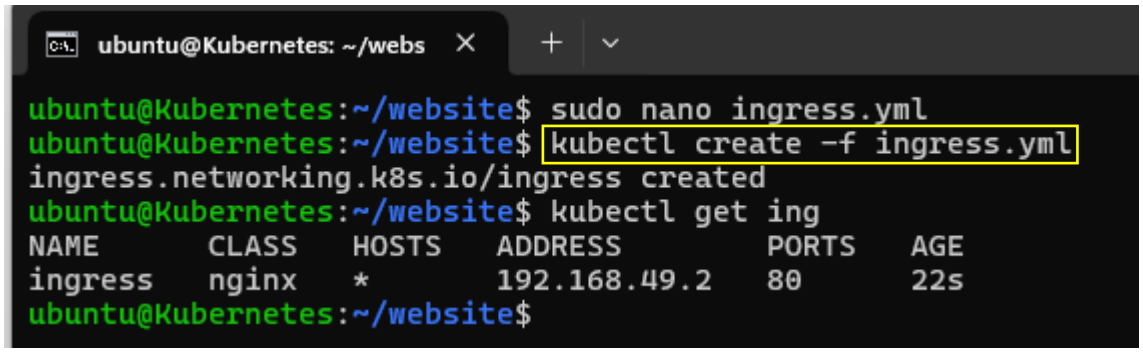


Create an **“ingress.yaml” file using** the **below-given command**

The “ingress” deployment will be **created**

Run the **below-given command to view the ingress**

Apply Port Forwarding to Open the Web Page on Separate Port



Paste the **below-given command** into **“EC2 Machine**

The port will be **successfully forwarded**