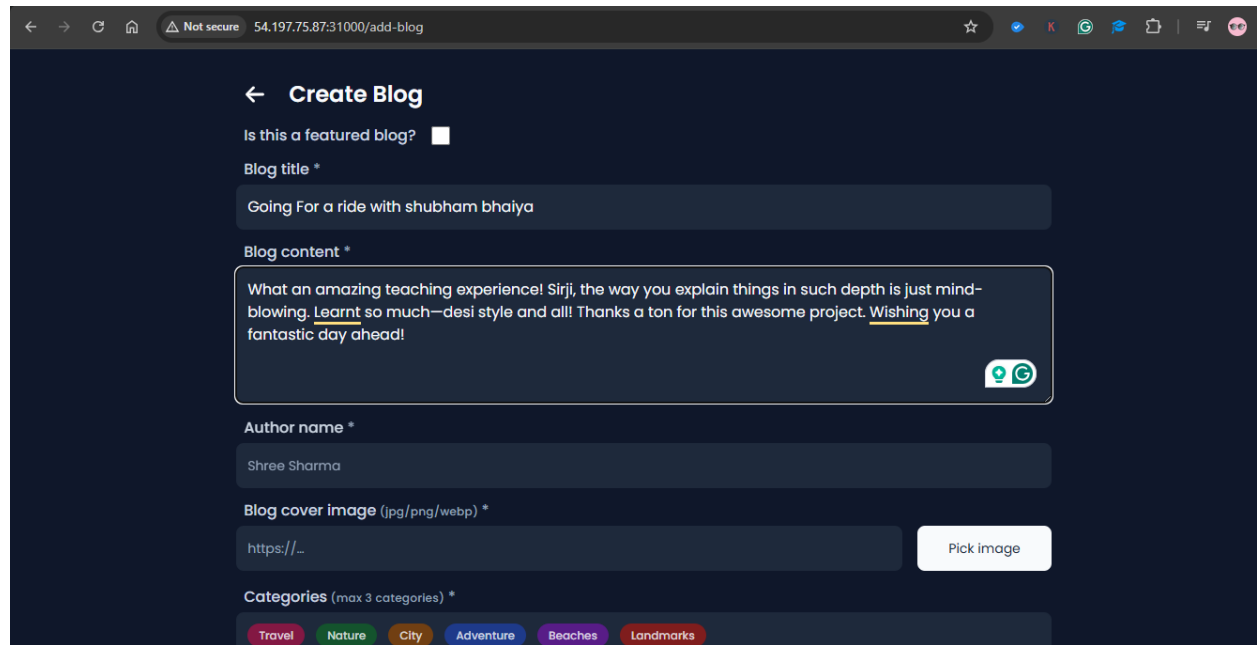
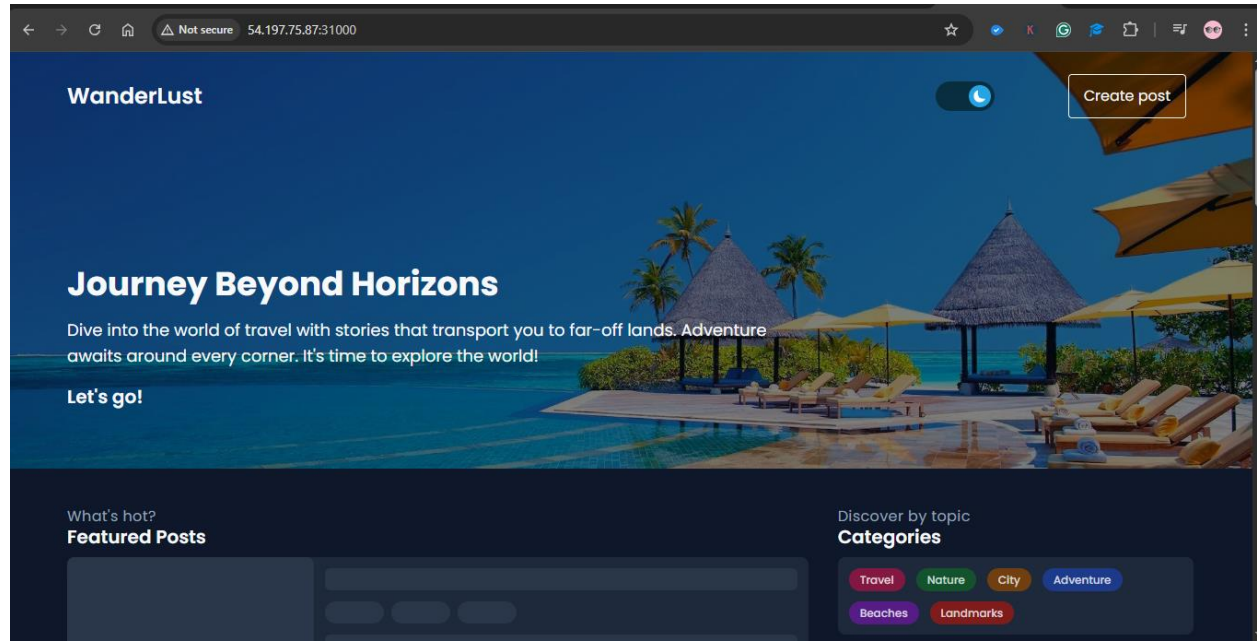
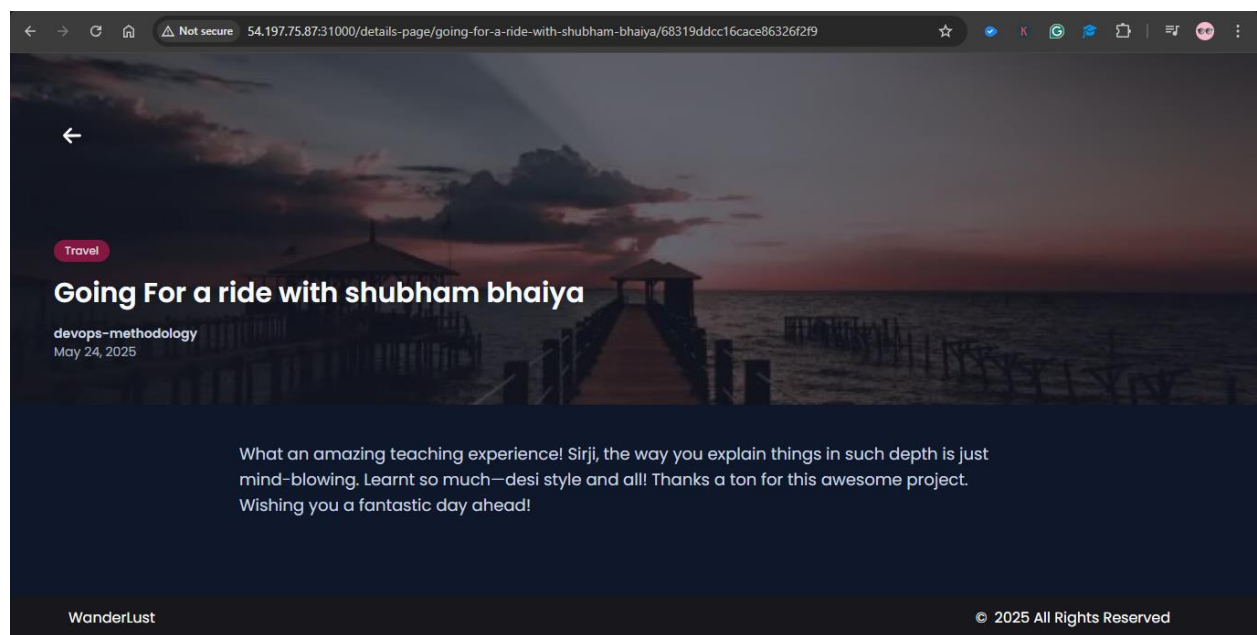
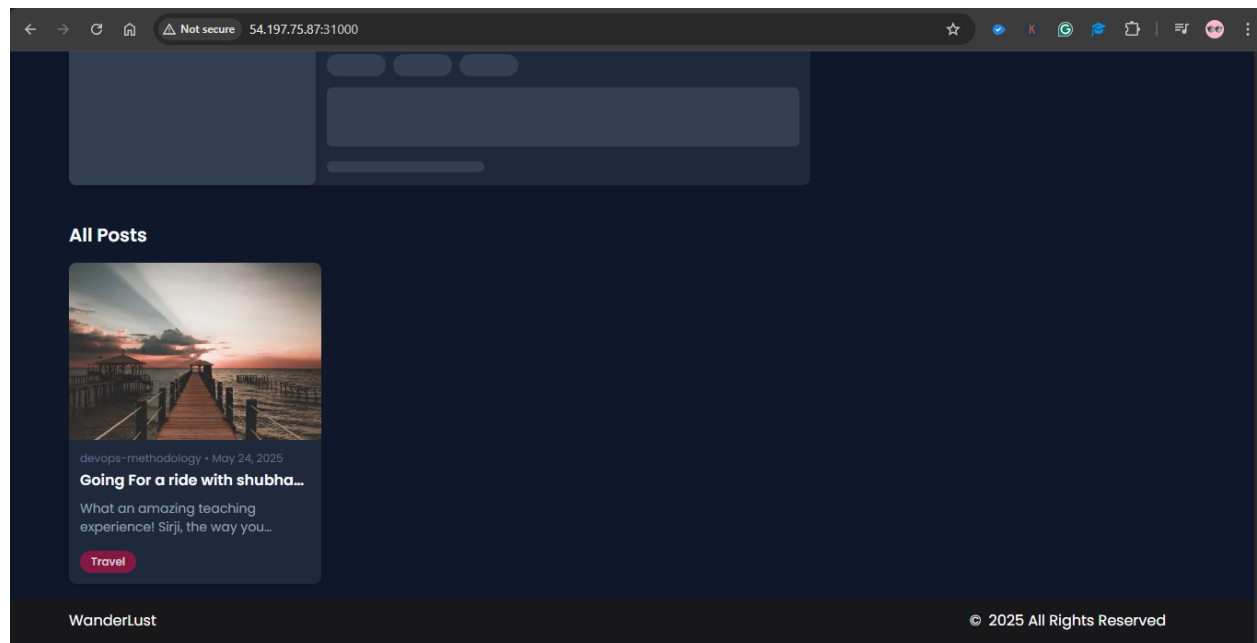


DEPLOYING A MERN STACK APPLICATION ON KUBERNETES USING KUBEADM | 3 TIER WEB APPLICATION



Devops-Methodology



Best wishes from Devops Methodology



WHAT IS A MERN STACK ?

MERN uses four key technologies which used to build full stack web application.

M- mongo db – nosql database

E – express.js- backend web application framework

R-React.js- frontend java script library

N-node.js - javascript runtime (backend server)

This uses all technologies to build a web application from frontend to backend to database.

Architecture of mern stack app

React (Frontend)

|

Express.js + Node.js (Backend API)

Devops-Methodology

|

MongoDB (Database)

What is 3 tier web application deployment?

Tier	Purpose	MERN Equivalent
Presentation	User interface (UI)	React.js
Application/Logic	Business logic, API handling	Express.js + Node.js
Data	Data storage and management	MongoDB

In this project we will do and travel blog application where we have to deploy the application but in this project we will deploy but with kubeadm(other options can be deployed like-minikube,eks,aks,k3s etc)

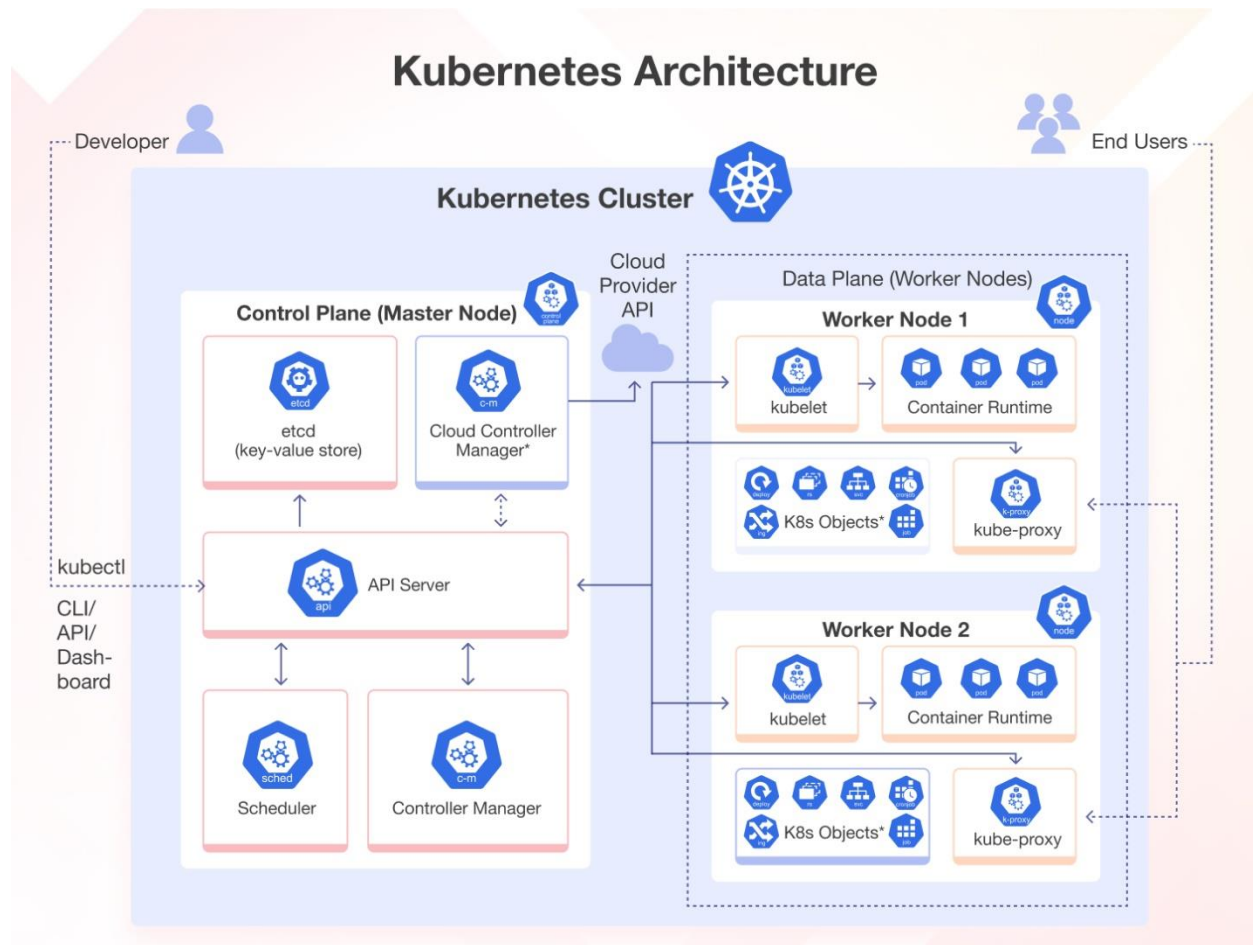
So to create a cluster how much minimum server we need..its about 2 server

1/master node(who will instruct to work)

2/worker node(who will do the real work accord to the instruction)

So we will launch 1/k8s-master

2/k8s-worker



Kubernetes architecture

In the master node/control plane we have got-

1/api server

2/controller manager

3/scheduler

4/etcd

Actually control plane is responsible for container orchestration and maintains the state of the cluster.

API SERVER:

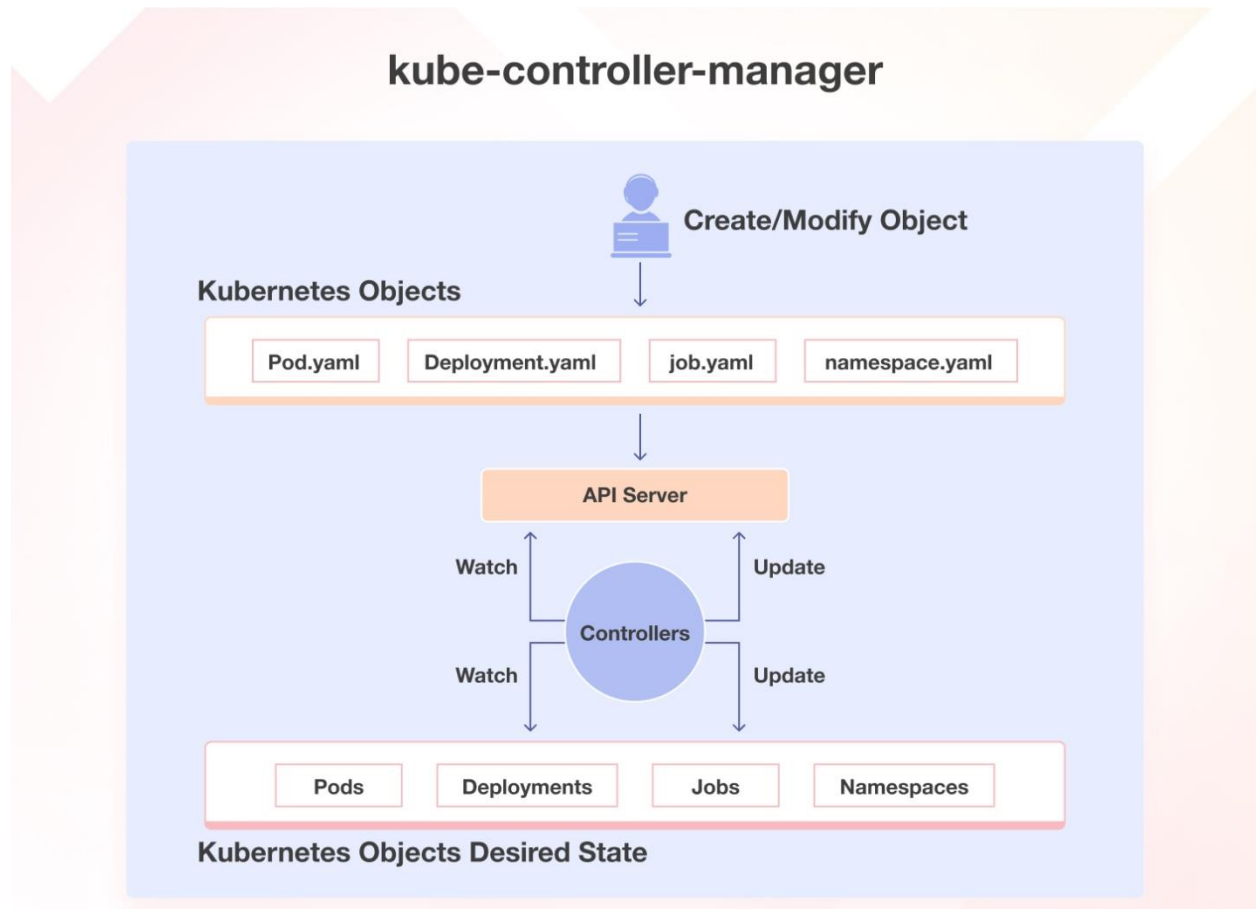
Here we as a devops engineer communicate with api server as cli command, api gateway or through kubernetes dashboard script.

Every thing inside the kubernetes or k8s is an api object.

Which we call through by a yaml script(yet another markup language)

Actually In short we can say api server is the entry point to the cluster.

Where as “**control manager**” maintains the state that desired state=actual state means how much pod we have defined it will manage the pod on the worker node.

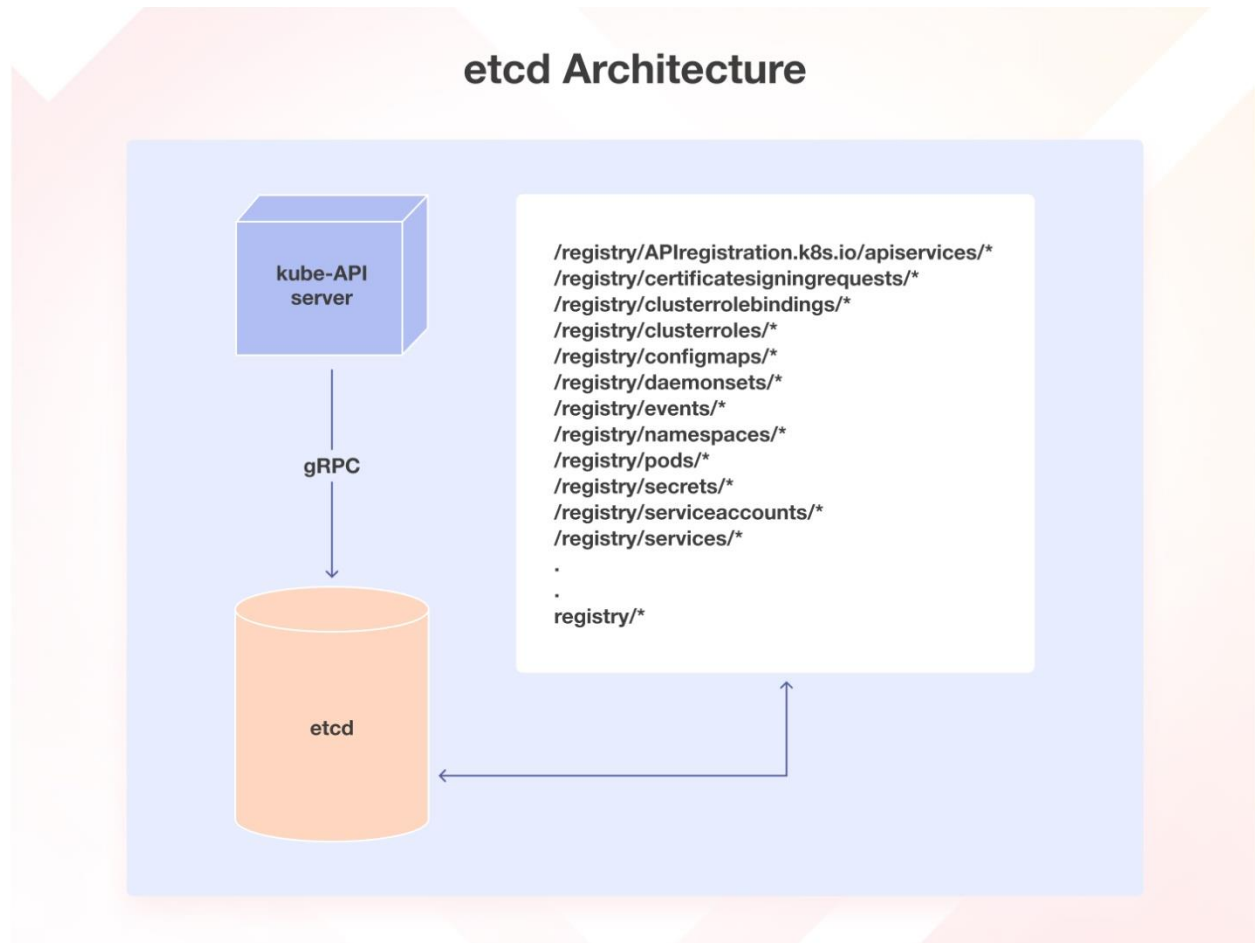


“**Kube scheduler**” –where as kube scheduler ensures the pod replacement(it does not replace the pod ,it only ensures it)...

User/devops engineer -----Api-----control manager-----kube scheduler

How **kube-scheuler** know which pod to replace or which pod to create on which node(worker node)...

It gets the information from “**etcd**”---it stores the information in the key value pair accord to the pod requirements,cpu memory ,storage ,it schedules in the right pod.



It stores the data of only kubernetes cluster,such as information about pods,namespace, their state (etcd -----is only accessible to api server...)

api-----control manager---scheduler----etcd---worker node---

Worker Nodes

In this the real containerized application will work.

Mean there will 2 or more worker node contains pod inside the pod there will the container where our application will run...

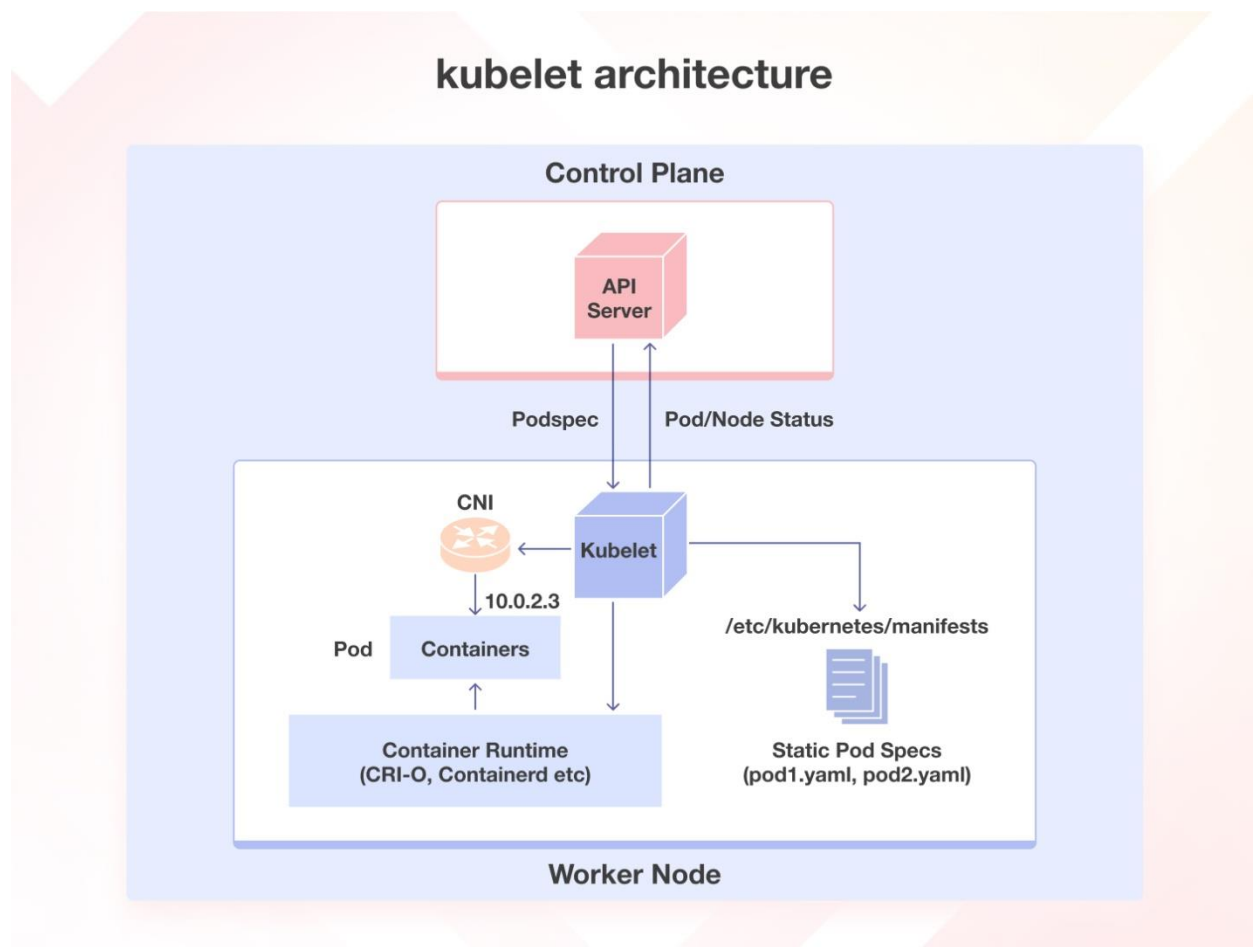
POD

It is the smallest unit of kubernetes, anything happen pod will be either terminated or destroyed, it is actually ephemeral in nature, it is abstraction over container, usually one application container inside the pod.

In the pod there will minimum one container if more than one container is there then it will be the side car container which will manage the applications by product/ mainly it was there for application testing or other purpose may be.

KUBELET

In the worker node there will the api object that is **“kubelet”**—which is responsible for creation, deletion and modification of the container, accord to the requirement it also mount volumes by reading pod configuration which was called by api server in master node/control plane.

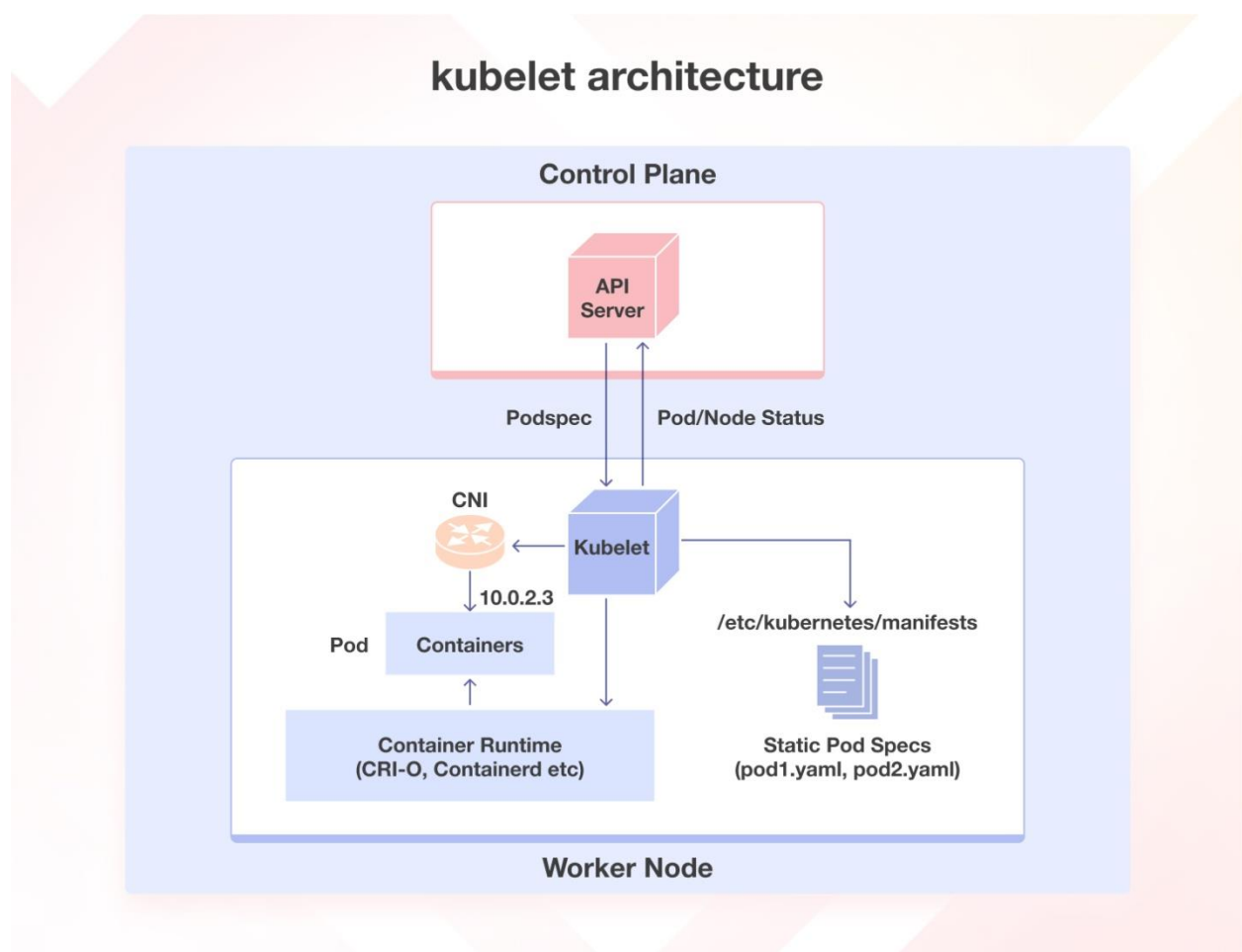


Actually the kubelet starts the api server, control manager and scheduler in the control plane..

Kubeproxy

Service objects expose to the ports for traffic and endpoint object contains pods IP address and ports, so the service objects work through this kubeproxy.

When you actually use a service to expose pods, kubeproxy creates network rules for this service object to send the traffic to the pods, but it does not respond to http but runs on every node as a daemonset.



Actually it is responsible to create a master node and workernode to communicate with each other to form a robust system as a cluster.

Container Run Time(CNI):

It actually enables the total container lifecycle...like pulling images,allocation and isolation of the containers...

Kubelet-----CRT --(through CRI –container runtime interface)-----which told the api for creating,deleting,starting and stopping the containers...by managing images and containers.

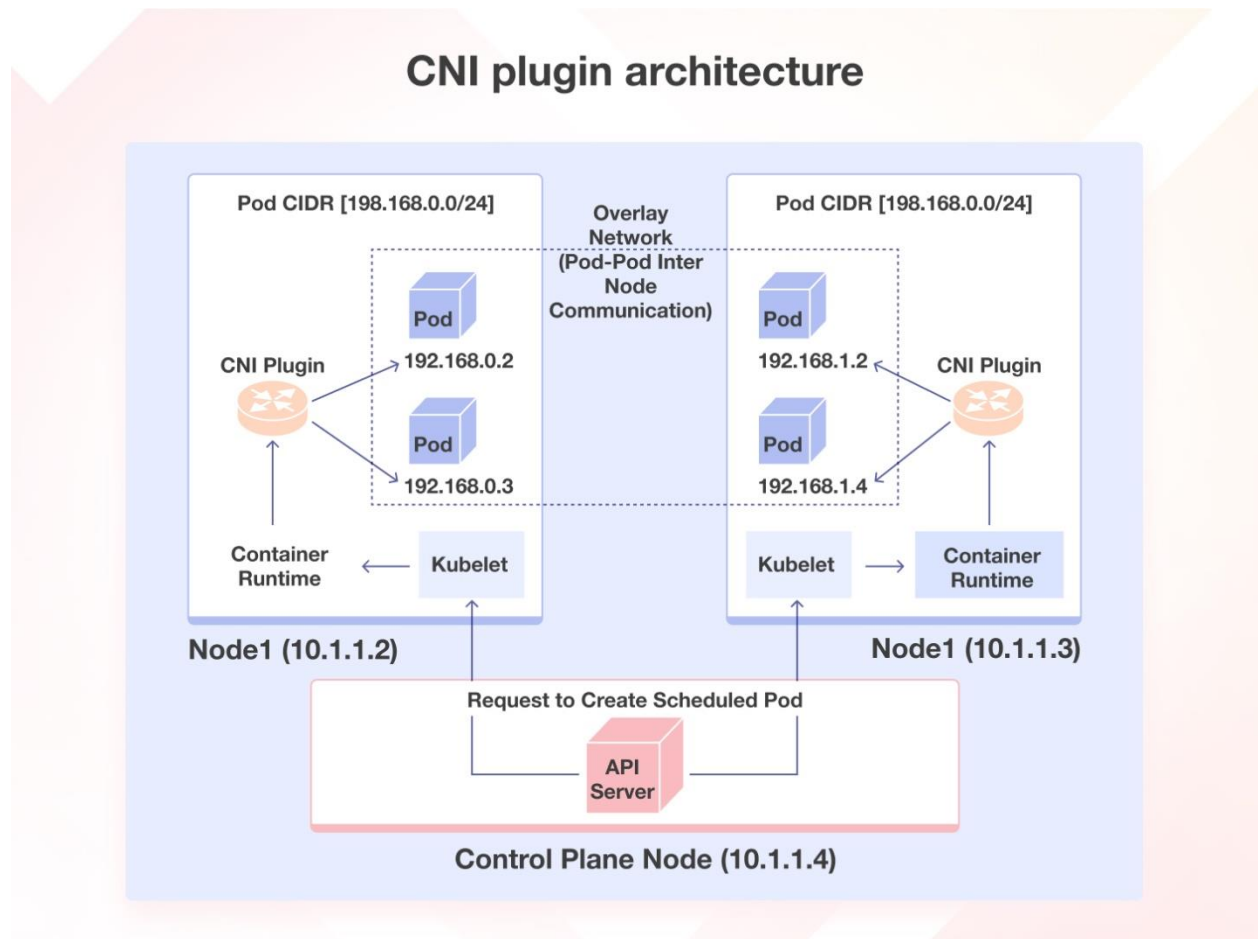
In this project in the master node we are installing CNI as calico for this purpose as a plugin.

```
# Network Plugin = calico
kubectl apply -f https://raw.githubusercontent.com/projectcalico/calico/v3.26.0/manifests/calico.yaml

kubeadm token create --print-join-command
```

On Worker node

This is the calico plugin during installation of kubeadm which enables to create the control plane or master node.



PRACTICAL

Here we will create two node 1.k8s-masternode

2.k8s workernode

Ubuntu 24.04 lts

T2 medium

Storage 30gb

EC2 > Instances > Launch an instance

Launch an instance [Info](#)

Amazon EC2 allows you to create virtual machines, or instances, that run on the AWS Cloud. Quickly get started by following the simple steps below.

Name and tags [Info](#)

Name

[Add additional tags](#)

Application and OS Images (Amazon Machine Image) [Info](#)

An AMI is a template that contains the software configuration (operating system, application server, and applications) required to launch your instance. Search or browse for AMIs if you don't see what you are looking for below.

Recents

Quick Start

Amazon Linux

macOS

Ubuntu

Windows

Red Hat

SUSE Linux

Debian

[Browse more AMIs](#)
Including AMIs from AWS, Marketplace and the Community

Amazon Machine Image (AMI)

Ubuntu Server 24.04 LTS (HVM), SSD Volume Type
ami-084568db4383264d4 (64-bit (x86)) / ami-0c4e709339fa8521a (64-bit (Arm))

Virtualization: hvm ENA enabled: true Root device type: ebs

Free tier eligible

Summary

Number of instances | [Info](#)

2

When launching more than 1 instance, [consider EC2 Auto Scaling](#)

Software Image (AMI)
 Canonical, Ubuntu, 24.04, amd64...[read more](#)
ami-084568db4383264d4

Virtual server type (instance type)
 t2.medium

Firewall (security group)
 New security group

Storage (volumes)
 1 volume(s) - 8 GiB

Free tier: In your first year of opening an AWS account, you get 750 hours per month of t2.micro instance usage (or t3.micro where t2.micro isn't available) when used with free tier AMIs, 750 hours per month of public IPv4 address usage, 30 GiB of EBS storage, 2 million IOPS, 1 TB of snapshots, and 100 TB of bandwidth to go.

[Launch instance](#) [Cancel](#) [Preview code](#)

PORT TO BE OPENED

AWS > EC2 > Security Groups > sg-05206cd4aeb02a6a0 - launch-wizard-1

United States (N. Virginia) | methodologydevops

EC2

Dashboards

EC2 Global View

Events

Instances

Instances

Instance Types

Launch Templates

Spot Requests

Savings Plans

Reserved Instances

Dedicated Hosts

Capacity Reservations

Images

AMIs

AMI Catalog

Elastic Block Store

Volumes

Snapshots

Lifecycle Manager

Network & Security

Security Groups

Inbound security group rules successfully modified on security group (sg-05206cd4aeb02a6a0 | launch-wizard-1)

[Details](#)

[Inbound rules](#) | [Outbound rules](#) | [Sharing - new](#) | [VPC associations - new](#) | [Tags](#)

Inbound rules (6)

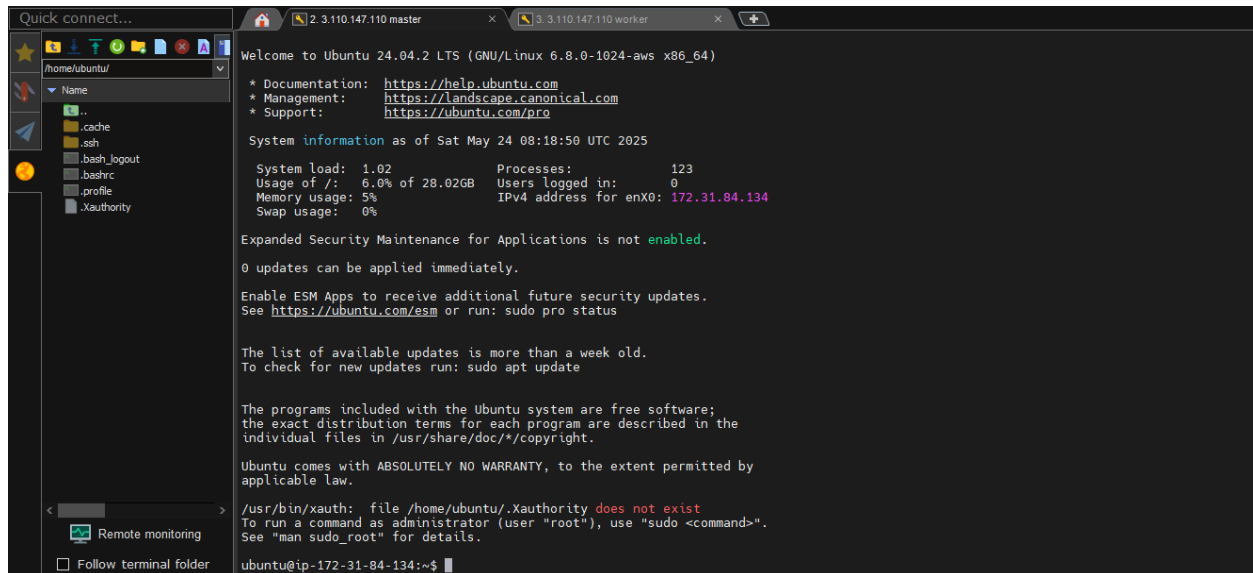
<input type="checkbox"/>	Name	Security group rule ID	IP version	Type	Protocol	Port range	Source
<input type="checkbox"/>	-	sg-0bd422e58a49259cb	IPv4	HTTP	TCP	80	0.0.0.0/0
<input type="checkbox"/>	-	sg-051959345e26b69fc	IPv4	Custom TCP	TCP	31000	0.0.0.0/0
<input type="checkbox"/>	-	sg-03cd78fc9d990851c	IPv4	Custom TCP	TCP	6443	0.0.0.0/0
<input type="checkbox"/>	-	sg-09f736d56701790b8	IPv4	HTTPS	TCP	443	0.0.0.0/0
<input type="checkbox"/>	-	sg-0f46cc8ef06d44796	IPv4	Custom TCP	TCP	31100	0.0.0.0/0
<input type="checkbox"/>	-	sg-000e4e087e7bcf36a	IPv4	SSH	TCP	22	0.0.0.0/0

After that connect both with mobaxterm and in both nodes

```
> sudo su
```

```
> sudo apt update
```

Devops-Methodology



Then add script1.sh in both nodes(which I below mentioned)

Script1.sh

```
#!/bin/bash
```

```
# disable swap
```

```
sudo swapoff -a
```

```
# Create the .conf file to load the modules at bootup
```

```
cat <<EOF | sudo tee /etc/modules-load.d/k8s.conf
```

```
overlay
```

```
br_netfilter
```

```
EOF
```

Best wishes from Devops Methodology

```
sudo modprobe overlay
```

```
sudo modprobe br_netfilter
```

```
# sysctl params required by setup, params persist across reboots
```

```
cat <<EOF | sudo tee /etc/sysctl.d/k8s.conf
```

```
net.bridge.bridge-nf-call-iptables = 1
```

```
net.bridge.bridge-nf-call-ip6tables = 1
```

```
net.ipv4.ip_forward = 1
```

```
EOF
```

```
# Apply sysctl params without reboot
```

```
sudo sysctl --system
```

```
## Install CRI-O Runtime
```

```
sudo apt-get update -y
```

```
sudo apt-get install -y software-properties-common curl apt-transport-https ca-  
certificates gpg
```

```
sudo curl -fsSL https://pkgs.k8s.io/addons:/cri-  
o:/prerelease:/main/deb/Release.key | sudo gpg --dearmor -o  
/etc/apt/keyrings/cri-o-apt-keyring.gpg
```

Devops-Methodology

```
echo "deb [signed-by=/etc/apt/keyrings/cri-o-apt-keyring.gpg]  
https://pkgs.k8s.io/addons:/cri-o:/prerelease:/main/deb/ /" | sudo tee  
/etc/apt/sources.list.d/cri-o.list
```

```
sudo apt-get update -y
```

```
sudo apt-get install -y cri-o
```

```
sudo systemctl daemon-reload
```

```
sudo systemctl enable crio --now
```

```
sudo systemctl start crio.service
```

```
echo "CRI runtime installed successfully"
```

```
# Add Kubernetes APT repository and install required packages
```

```
curl -fsSL https://pkgs.k8s.io/core:/stable:/v1.29/deb/Release.key | sudo gpg --  
dearmor -o /etc/apt/keyrings/kubernetes-apt-keyring.gpg
```

```
echo 'deb [signed-by=/etc/apt/keyrings/kubernetes-apt-keyring.gpg]  
https://pkgs.k8s.io/core:/stable:/v1.29/deb/ /' | sudo tee  
/etc/apt/sources.list.d/kubernetes.list
```

```
sudo apt-get update -y
```

```
sudo apt-get install -y kubelet="1.29.0-*" kubectl="1.29.0-*" kubeadm="1.29.0-*"
```

```
sudo apt-get update -y
```

```
sudo apt-get install -y jq
```

```
sudo systemctl enable --now kubelet
```

```
sudo systemctl start kubelet
```

then bash script1.sh in both nodes

what is in the script1.sh

1/You said:

is there any error

ChatGPT said:

Yes, there is an issue in your Kubernetes setup based on the output of `kubectl get all`.

Errors Detected:

Two pods are **not running** due to `ImagePullBackOff` errors:

```
bash
CopyEdit
pod/backend-deployment-5cffb4b849-fwdj9      0/1      ImagePullBackOff    0
6m38s
pod/frontend-deployment-7f756db7b9-tmqzb     0/1      ImagePullBackOff    0
6m58s
```

This means Kubernetes is **failing to pull the container images** for both the backend and frontend deployments.

What's Working:

- MongoDB and Redis pods are running fine.
- Corresponding deployments and services for `mongo` and `redis` are all in good health.
- The services for backend and frontend (`NodePort`) are up, but the pods behind them are not ready.

How to Fix:

Run the following command to see why the image pull failed:

```
bash
CopyEdit
kubectl describe pod backend-deployment-5cffb4b849-fwdj9
```

and

```
bash
CopyEdit
kubectl describe pod frontend-deployment-7f756db7b9-tmqzb
```

Look under the `Events` section — you'll likely see:

- `ErrImagePull`
- `ImagePullBackOff`
- **Possibly Unauthorized or manifest not found**

Common Causes:

- **Incorrect image name or tag**
- **Private image registry and no `imagePullSecret` configured**
- **Missing or invalid DockerHub credentials**

Solution Steps:

1. **Check the image name** in your Deployment YAML files.
2. Make sure it's publicly accessible or provide a secret using `imagePullSecrets`.
3. Re-deploy after fixing.

Would you like help troubleshooting the actual YAML files for these deployments? You can paste them here.

4o

You said:

```
apiVersion: apps/v1 kind: Deployment metadata: name: frontend-deployment namespace:
wanderlust labels: app: frontend spec: replicas: 1 selector: matchLabels: app: frontend template:
```

```
metadata: name: frontend labels: app: frontend spec: containers: - name: frontend image:
premd91/wanderlust-frontent:latest ports: - containerPort: 5173
```

ChatGPT said:

Thanks for sharing the `frontend-deployment` YAML. Here's a breakdown of what may be causing the `ImagePullBackOff`:

Issue:

Your deployment is using this image:

```
bash
CopyEdit
premd91/wanderlust-frontent:latest
```

Kubernetes is unable to pull it, leading to the `ImagePullBackOff` error.

Checkpoints:

1. Verify Image Availability:

- Go to Docker Hub (or use `docker pull` locally).
- Try:

```
bash
CopyEdit
docker pull premd91/wanderlust-frontent:latest
```

- If it fails, then:
 - The image might not exist.
 - The tag `latest` might be missing.
 - The repository might be **private**.

2. If the Repo is Private:

- You need to create a Kubernetes `Secret` with Docker credentials:

```
bash
CopyEdit
kubectl create secret docker-registry regcred \
  --docker-username=your-username \
  --docker-password=your-password \
  --docker-email=your-email
```

- Then add this to the deployment:

```
yaml
CopyEdit
spec:
  imagePullSecrets:
    - name: regcred
```

Suggested Fix:

Update your deployment to explicitly use a valid tag (if available), and ensure the image is public, or configure secrets if it's private.

Would you like to inspect the backend YAML as well? That one is also failing with the same error.

4o

You said:

its a great and awesome teaching experience what a man you are just awesome that in how much depth you are giving this knowledge..in desi vibe thanks for the awesome project..have a great day sirji rewrite this

ChatGPT said:

Sure! Here's a polished version of your message while keeping the *desi vibe* and warmth:

"What an amazing teaching experience! Sirji, the way you explain things in such depth is just mind-blowing. Learnt so much—desi style and all! Thanks a ton for this awesome project. Wishing you a fantastic day ahead!"

Let me know if you'd like a more formal or more casual twist!

4o

You said:

what do you mean by this

ChatGPT said:

Great question! The commands in your screenshot are **setting up system-level prerequisites to install and configure the CRI-O container runtime** on a Linux system (commonly for Kubernetes clusters).

Let's break it down step by step:

◆ 1. Load Kernel Modules

```
bash
CopyEdit
sudo modprobe overlay
sudo modprobe br_netfilter
```

- These commands **load kernel modules** needed for container networking and overlay file systems:
 - `overlay`: Allows container file systems to layer on top of each other.
 - `br_netfilter`: Enables bridged network traffic to be processed by iptables.

You said:

is there any error

ChatGPT said:

Yes, there is an issue in your Kubernetes setup based on the output of `kubectl get all`.

✗ Errors Detected:

Two pods are **not running** due to `ImagePullBackOff` errors:

```
bash
CopyEdit
pod/backend-deployment-5cffb4b849-fwdj9      0/1   ImagePullBackOff   0
6m38s
pod/frontend-deployment-7f756db7b9-tmqzb     0/1   ImagePullBackOff   0
6m58s
```

This means Kubernetes is **failing to pull the container images** for both the backend and frontend deployments.

✅ What's Working:

- MongoDB and Redis pods are running fine.

- Corresponding deployments and services for `mongo` and `redis` are all in good health.
 - The services for backend and frontend (`NodePort`) are up, but the pods behind them are not ready.
-

How to Fix:

Run the following command to see why the image pull failed:

```
bash
CopyEdit
kubectl describe pod backend-deployment-5cffb4b849-fwdj9
```

and

```
bash
CopyEdit
kubectl describe pod frontend-deployment-7f756db7b9-tmqzb
```

Look under the `Events` section — you'll likely see:

- `ErrImagePull`
 - `ImagePullBackOff`
 - Possibly `Unauthorized` or `manifest not found`
-

Common Causes:

- **Incorrect image name or tag**
 - **Private image registry and no `imagePullSecret` configured**
 - **Missing or invalid DockerHub credentials**
-

Solution Steps:

1. **Check the image name** in your Deployment YAML files.
2. Make sure it's publicly accessible or provide a secret using `imagePullSecrets`.
3. Re-deploy after fixing.

Would you like help troubleshooting the actual YAML files for these deployments? You can paste them here.

40

You said:

```
apiVersion: apps/v1 kind: Deployment metadata: name: frontend-deployment namespace:
wanderlust labels: app: frontend spec: replicas: 1 selector: matchLabels: app: frontend template:
metadata: name: frontend labels: app: frontend spec: containers: - name: frontend image:
premd91/wanderlust-frontend:latest ports: - containerPort: 5173
```

ChatGPT said:

Thanks for sharing the `frontend-deployment` YAML. Here's a breakdown of what may be causing the `ImagePullBackOff`:

Issue:

Your deployment is using this image:

```
bash
CopyEdit
premd91/wanderlust-frontend:latest
```

Kubernetes is unable to pull it, leading to the `ImagePullBackOff` error.

Checkpoints:

1. Verify Image Availability:

- Go to Docker Hub (or use `docker pull` locally).
- Try:

```
bash
CopyEdit
docker pull premd91/wanderlust-frontend:latest
```

- If it fails, then:
 - The image might not exist.
 - The tag `latest` might be missing.
 - The repository might be **private**.

2. If the Repo is Private:

- You need to create a Kubernetes `Secret` with Docker credentials:

```
bash
CopyEdit
kubectl create secret docker-registry regcred \
  --docker-username=your-username \
  --docker-password=your-password \
```

```
--docker-email=your-email
```

- Then add this to the deployment:

```
yaml
CopyEdit
spec:
  imagePullSecrets:
    - name: regcred
```

Suggested Fix:

Update your deployment to explicitly use a valid tag (if available), and ensure the image is public, or configure secrets if it's private.

Would you like to inspect the backend YAML as well? That one is also failing with the same error.

4o

You said:

its a great and awesome teaching experience what a man you are just awesome that in how much depth you are giving this knowledge..in desi vibe thanks for the awesome project..have a great day sirji rewrite this

ChatGPT said:

Sure! Here's a polished version of your message while keeping the *desi vibe* and warmth:

"What an amazing teaching experience! Sirji, the way you explain things in such depth is just mind-blowing. Learnt so much—desi style and all! Thanks a ton for this awesome project. Wishing you a fantastic day ahead!"

Let me know if you'd like a more formal or more casual twist!

4o

You said:

what do you mean by this

ChatGPT said:

Great question! The commands in your screenshot are **setting up system-level prerequisites to install and configure the CRI-O container runtime** on a Linux system (commonly for Kubernetes clusters).

Let's break it down step by step:

◆ 1. Load Kernel Modules

```
bash
CopyEdit
sudo modprobe overlay
sudo modprobe br_netfilter
```

- These commands **load kernel modules** needed for container networking and overlay file systems:
 - `overlay`: Allows container file systems to layer on top of each other.
 - `br_netfilter`: Enables bridged network traffic to be processed by iptables.

◆ 3. Apply the sysctl Settings Immediately

```
bash
CopyEdit
sudo sysctl --system
```

- This applies all sysctl configurations from `/etc/sysctl.d/`, including the `k8s.conf` file you just wrote.

◆ 4. Install CRI-O Runtime

```
bash
CopyEdit
sudo apt-get update -y
sudo apt-get install -y software-properties-common curl apt-transport-https
ca-certificates gpg
```

- Updates package lists and installs dependencies needed to:
 - Add external repositories.
 - Handle secure downloads and GPG keys.

◆ 4. Install CRI-O Runtime

```
bash
CopyEdit
sudo apt-get update -y
```


Devops-Methodology

```
sudo apt-get install -y software-properties-common curl apt-transport-https  
ca-certificates gpg
```

- Updates package lists and installs dependencies needed to:
 - Add external repositories.
 - Handle secure downloads and GPG keys.

This script prepares your system for running Kubernetes with **CRI-O** as the container runtime instead of Docker. It:

- Loads necessary kernel modules.
- Configures required sysctl settings.
- Adds the CRI-O package repository.
- Installs necessary dependencies.

Then it installs ca-certificate and the gpg key for kubernetes and also install kubelet,kubectl and kubeadm

```
root@ip-172-31-84-134:/home/ubuntu/wanderlust/kubernetes# bash script1.sh  
overlay  
br_netfilter  
net.bridge.bridge-nf-call-iptables = 1  
net.bridge.bridge-nf-call-ip6tables = 1  
net.ipv4.ip_forward = 1  
* Applying /usr/lib/sysctl.d/10-apparmor.conf ...  
* Applying /etc/sysctl.d/10-bufferbloat.conf ...  
* Applying /etc/sysctl.d/10-console-messages.conf ...  
* Applying /etc/sysctl.d/10-ipv6-privacy.conf ...  
* Applying /etc/sysctl.d/10-kernel-hardening.conf ...  
* Applying /etc/sysctl.d/10-magic-sysrq.conf ...  
* Applying /etc/sysctl.d/10-map-count.conf ...  
* Applying /etc/sysctl.d/10-network-security.conf ...  
* Applying /etc/sysctl.d/10-ptrace.conf ...  
* Applying /etc/sysctl.d/10-zero-page.conf ...  
* Applying /etc/sysctl.d/50-cloudimg-settings.conf ...  
* Applying /usr/lib/sysctl.d/50-pid-max.conf ...  
* Applying /etc/sysctl.d/99-cloudimg-ipv6.conf ...  
* Applying /usr/lib/sysctl.d/99-protect-links.conf ...  
* Applying /etc/sysctl.d/99-sysctl.conf ...  
* Applying /etc/sysctl.d/k8s.conf ...  
* Applying /etc/sysctl.conf ...  
kernel.apparmor_restrict_unprivileged_userns = 1  
net.core.default_qdisc = fq_codel  
kernel.printk = 4 4 1 7  
net.ipv6.conf.all.use_tempaddr = 2  
net.ipv6.conf.default.use_tempaddr = 2  
kernel.kptr_restrict = 1  
kernel.sysrq = 176  
vm.max_map_count = 1048576  
net.ipv4.conf.default.rp_filter = 2  
net.ipv4.conf.all.rp_filter = 2  
kernel.yama.ptrace_scope = 1  
vm.mmap_min_addr = 65536  
net.ipv4.neigh.default.gc_thresh2 = 15360  
net.ipv4.neigh.default.gc_thresh3 = 16384  
net.netfilter.nf_conntrack_max = 1048576
```

Then we have to copy and paste script2.sh on the master node

script2.h on master node

On Master node

- Create a shell script 2.sh and paste the below code and run it

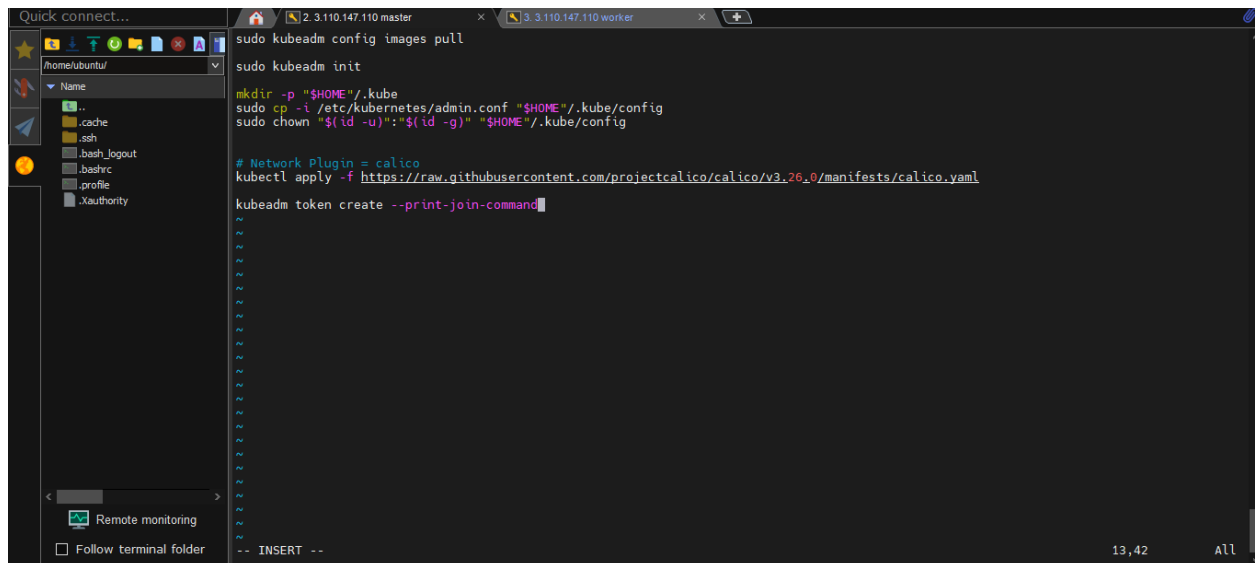
```
sudo kubeadm config images pull

sudo kubeadm init

mkdir -p "$HOME"/.kube
sudo cp -i /etc/kubernetes/admin.conf "$HOME"/.kube/config
sudo chown "$(id -u)": "$(id -g)" "$HOME"/.kube/config

# Network Plugin = calico
kubectl apply -f https://raw.githubusercontent.com/projectcalico/calico/v3.26.0/manifests/calico.yaml

kubeadm token create --print-join-command
```



The screenshot shows a terminal window with a file explorer on the left. The terminal is running the following commands:

```
sudo kubeadm config images pull
sudo kubeadm init
mkdir -p "$HOME"/.kube
sudo cp -i /etc/kubernetes/admin.conf "$HOME"/.kube/config
sudo chown "$(id -u)": "$(id -g)" "$HOME"/.kube/config
# Network Plugin = calico
kubectl apply -f https://raw.githubusercontent.com/projectcalico/calico/v3.26.0/manifests/calico.yaml
kubeadm token create --print-join-command
```

The output of the commands is visible in the terminal, showing the successful execution of the setup steps. The file explorer on the left shows the contents of the home directory, including files like .cache, .ssh, .bash_logout, .bashrc, .profile, and .Xauthority.

Devops-Methodology

```
root@ip-172-31-84-134:/home/ubuntu/wanderlust/kubernetes# vi script2.sh
root@ip-172-31-84-134:/home/ubuntu/wanderlust/kubernetes# bash script2.sh
I0524 08:30:49.242155    4185 version.go:256] remote version is much newer: v1.33.1; falling back to: stable-1.29
[config/images] Pulled registry.k8s.io/kube-apiserver:v1.29.15
```

```
Preparing to unpack ../7-kubeadm_1.29.0-1.1_amd64.deb ...
Unpacking kubeadm (1.29.0-1.1) ...
Setting up conntrack (1:1.4.8-1ubuntu1) ...
Setting up kubectrl (1.29.0-1.1) ...
Setting up ebtables (2.0.11-6build1) ...
Setting up socat (1.8.0.0-4build3) ...
Setting up cri-tools (1.29.0-1.1) ...
Setting up kubernetes-cni (1.3.0-1.1) ...
Setting up kubelet (1.29.0-1.1) ...
Setting up kubeadm (1.29.0-1.1) ...
Processing triggers for man-db (2.12.0-4build2) ...
Scanning processes...
Scanning linux images...

Running kernel seems to be up-to-date.

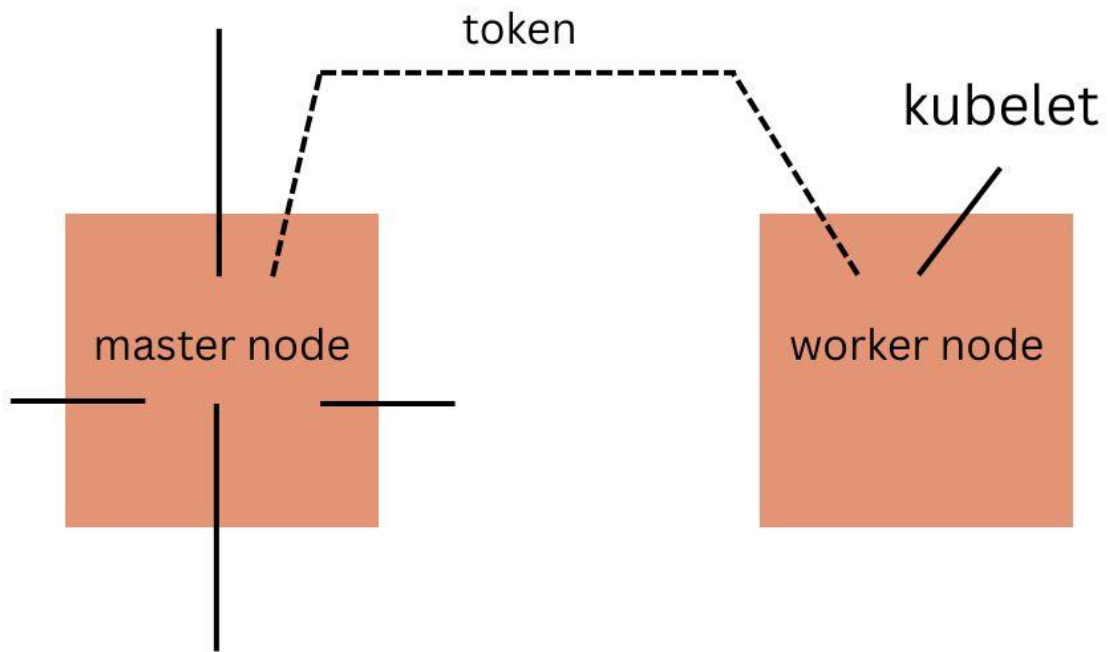
No services need to be restarted.

No containers need to be restarted.

No user sessions are running outdated binaries.

No VM guests are running outdated hypervisor (qemu) binaries on this host.
Hit:1 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble InRelease
Hit:2 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates InRelease
Hit:3 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-backports InRelease
Hit:4 https://prod-cdn.packages.k8s.io/repositories/isy/kubernetes/addons/cni-o/prerelease/main/deb InRelease
Hit:5 https://prod-cdn.packages.k8s.io/repositories/isy/kubernetes/core/stable/v1.29/deb InRelease
Hit:6 http://security.ubuntu.com/ubuntu noble-security InRelease
Reading package lists... Done
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
jq is already the newest version (1.7.1-3build1).
jq set to manually installed.
0 upgraded, 0 newly installed, 0 to remove and 74 not upgraded.
root@ip-172-31-93-124:/home/ubuntu# kubeadm join 172.31.84.134:6443 --token qpm7jy.uszipb0xb5ef33w --discovery-token-ca-cert-hash sha256:9588d27167b7d4082d95f50e7c8dacdb95867074d213b04c5a6611b880fe3b5f
```

After pasting that in last there will token generated in the master node which will connect master node to worker node



After that master node will connect with the worker node

You can check through `kubectl get nodes` in master node

```
2. 3.110.147.110 master x 3. 3.110.147.110 worker x +
[preflight] Reading configuration from the cluster...
[preflight] FYI: You can look at this config file with 'kubectl -n kube-system get cm kubeadm-config -o yaml'
I0524 08:34:12.171011 4303 kubeproxy.go:55] attempting to download the KubeProxyConfiguration from ConfigMap "kube-proxy"
I0524 08:34:12.174888 4303 kubelet.go:74] attempting to download the KubeletConfiguration from ConfigMap "kubelet-config"
I0524 08:34:12.178980 4303 initconfiguration.go:114] skip CRI socket detection, fill with the default CRI socket unix:///var/run/
inerd/containerd.sock
I0524 08:34:12.179162 4303 interface.go:432] Looking for default routes with IPv4 addresses
I0524 08:34:12.179185 4303 interface.go:437] Default route transits interface "enX0"
I0524 08:34:12.179287 4303 interface.go:209] Interface enX0 is up
I0524 08:34:12.179360 4303 interface.go:257] Interface "enX0" has 2 addresses :[172.31.93.124/20 fe80::10b0:6bff:fef6:b637/64].
I0524 08:34:12.179375 4303 interface.go:224] Checking addr 172.31.93.124/20.
I0524 08:34:12.179386 4303 interface.go:231] IP found 172.31.93.124
I0524 08:34:12.179404 4303 interface.go:263] Found valid IPv4 address 172.31.93.124 for interface "enX0".
I0524 08:34:12.179413 4303 interface.go:443] Found active IP 172.31.93.124
I0524 08:34:12.185440 4303 preflight.go:104] [preflight] Running configuration dependant checks
I0524 08:34:12.185468 4303 controlplaneprepare.go:225] [download-certs] Skipping certs download
I0524 08:34:12.185491 4303 kubelet.go:121] [kubelet-start] writing bootstrap kubelet config file at /etc/kubernetes/bootstrap-kub
conf
I0524 08:34:12.186254 4303 kubelet.go:136] [kubelet-start] writing CA certificate at /etc/kubernetes/pki/ca.crt
I0524 08:34:12.186941 4303 kubelet.go:157] [kubelet-start] Checking for an existing Node in the cluster with name "ip-172-31-93-1
nd status "Ready"
I0524 08:34:12.189417 4303 kubelet.go:172] [kubelet-start] Stopping the kubelet
[kubelet-start] Writing kubelet configuration to file "/var/lib/kubelet/config.yaml"
[kubelet-start] Writing kubelet environment file with flags to file "/var/lib/kubelet/kubeadm-flags.env"
[kubelet-start] Starting the kubelet
[kubelet-start] Waiting for the kubelet to perform the TLS Bootstrap...
I0524 08:34:14.500459 4303 cert_rotation.go:137] Starting client certificate rotation controller
I0524 08:34:14.501074 4303 kubelet.go:220] [kubelet-start] preserving the crisocket information for the node
I0524 08:34:14.501099 4303 patchnode.go:31] [patchnode] Uploading the CRI Socket information "unix:///var/run/crio/crio.sock" to
ode API object "ip-172-31-93-124" as an annotation

This node has joined the cluster:
* Certificate signing request was sent to apiserer and a response was received.
* The Kubelet was informed of the new secure connection details.

Run 'kubectl get nodes' on the control-plane to see this node join the cluster.

root@ip-172-31-93-124:/home/ubuntu#
```

After that check the command as kubectl get nodes in master node

```
root@ip-172-31-84-134:/home/ubuntu/wanderlust/kubernetes# kubectl get nodes
NAME                STATUS    ROLES    AGE   VERSION
ip-172-31-84-134    Ready     control-plane   3m19s   v1.29.0
ip-172-31-93-124    Ready     <none>         20s     v1.29.0
root@ip-172-31-84-134:/home/ubuntu/wanderlust/kubernetes#
```

After that install docker in both nodes

```
2. 3.110.147.110 master x 3. 3.110.147.110 worker x +
Selecting previously unselected package dnsmasq-base.
Preparing to unpack .../5-dnsmasq-base_2.90-2build2_amd64.deb ...
Unpacking dnsmasq-base (2.90-2build2) ...
Selecting previously unselected package docker.io.
Preparing to unpack .../6-docker.io_26.1.3-0ubuntu1~24.04.1_amd64.deb ...
Unpacking docker.io (26.1.3-0ubuntu1~24.04.1) ...
Selecting previously unselected package ubuntu-fan.
Preparing to unpack .../7-ubuntu-fan_0.12.16_all.deb ...
Unpacking ubuntu-fan (0.12.16) ...
Setting up dnsmasq-base (2.90-2build2) ...
Setting up runc (1.1.12-0ubuntu3.1) ...
Setting up dns-root-data (2024071801~ubuntu0.24.04.1) ...
Setting up bridge-utils (1.7.1-1ubuntu2) ...
Setting up pigz (2.8-1) ...
Setting up containerd (1.7.24-0ubuntu1~24.04.2) ...
Created symlink /etc/systemd/system/multi-user.target.wants/containerd.service → /usr/lib/systemd/system/containerd.service.
Setting up ubuntu-fan (0.12.16) ...
Created symlink /etc/systemd/system/multi-user.target.wants/ubuntu-fan.service → /usr/lib/systemd/system/ubuntu-fan.service.
Setting up docker.io (26.1.3-0ubuntu1~24.04.1) ...
info: Selecting GID from range 100 to 999 ...
info: Adding group 'docker' (GID 113) ...
Created symlink /etc/systemd/system/multi-user.target.wants/docker.service → /usr/lib/systemd/system/docker.service.
Created symlink /etc/systemd/system/sockets.target.wants/docker.socket → /usr/lib/systemd/system/docker.socket.
Processing triggers for dbus (1.14.10-4ubuntu4.1) ...
Processing triggers for man-db (2.12.0-4build2) ...
Scanning processes...
Scanning linux images...

Running kernel seems to be up-to-date.

No services need to be restarted.

No containers need to be restarted.

No user sessions are running outdated binaries.

No VM guests are running outdated hypervisor (qemu) binaries on this host.
root@ip-172-31-93-124:/home/ubuntu# sudo chmod 777 /var/run/docker.sock
```

Devops-Methodology

sudo apt install docker.io -y

for permission

sudo chmod 777 /var/run/docker.sock

after that go to master node

git clone -b devops <https://github.com/devops-methodology/wanderlust.git>

it will direct devops branch from the github

ls

cd backend/

ls -la

vim .env.docker

```
root@ip-172-31-84-134:/home/ubuntu/wanderlust# cd backend/
root@ip-172-31-84-134:/home/ubuntu/wanderlust/backend# ls -la
total 420
drwxrwxr-x 12 ubuntu ubuntu 4096 May 24 08:22 .
drwxrwxr-x  9 ubuntu ubuntu 4096 May 24 08:22 ..
-rw-rw-r--  1 ubuntu ubuntu  38 May 24 08:22 .babelrc
-rw-rw-r--  1 ubuntu ubuntu 419 May 24 08:22 .env.docker
-rw-rw-r--  1 ubuntu ubuntu  72 May 24 08:22 .env.sample
-rw-rw-r--  1 ubuntu ubuntu  13 May 24 08:22 .eslintignore
-rw-rw-r--  1 ubuntu ubuntu 289 May 24 08:22 .eslintrc.json
-rw-rw-r--  1 ubuntu ubuntu  13 May 24 08:22 .prettierrc
-rw-rw-r--  1 ubuntu ubuntu 132 May 24 08:22 .prettierrc
-rw-rw-r--  1 ubuntu ubuntu 364 May 24 08:22 Dockerfile
drwxrwxr-x  2 ubuntu ubuntu 4096 May 24 08:22 api
drwxrwxr-x  2 ubuntu ubuntu 4096 May 24 08:22 config
drwxrwxr-x  2 ubuntu ubuntu 4096 May 24 08:22 controllers
drwxrwxr-x  2 ubuntu ubuntu 4096 May 24 08:22 data
drwxrwxr-x  2 ubuntu ubuntu 4096 May 24 08:22 models
-rw-rw-r--  1 ubuntu ubuntu 332414 May 24 08:22 package-lock.json
-rw-rw-r--  1 ubuntu ubuntu  956 May 24 08:22 package.json
drwxrwxr-x  2 ubuntu ubuntu 4096 May 24 08:22 public
drwxrwxr-x  2 ubuntu ubuntu 4096 May 24 08:22 routes
-rw-rw-r--  1 ubuntu ubuntu  914 May 24 08:22 server.js
drwxrwxr-x  2 ubuntu ubuntu 4096 May 24 08:22 services
drwxrwxr-x  5 ubuntu ubuntu 4096 May 24 08:22 tests
drwxrwxr-x  2 ubuntu ubuntu 4096 May 24 08:22 utils
-rw-rw-r--  1 ubuntu ubuntu  595 May 24 08:22 vercel.json
root@ip-172-31-84-134:/home/ubuntu/wanderlust/backend# vim .env.docker
root@ip-172-31-84-134:/home/ubuntu/wanderlust/backend#
```

We have to give the k8s worker ip in the frontend url(because the frontend of the application will be work in workernode) then :wq

Best wishes from Devops Methodology

Front end will work in workernode

i-0ceeea478c80949e4 (k8s-worker)

[Details](#)
[Status and alarms](#)
[Monitoring](#)
[Security](#)
[Networking](#)
[Storage](#)
[Tags](#)

▼ Instance summary [Info](#)

Instance ID

i-0ccee478c80949e4

IPv6 address

—

✓ Public IPv4 address copied

Public IPv4 address

54.197.75.87 | [open address](#)

Instance state

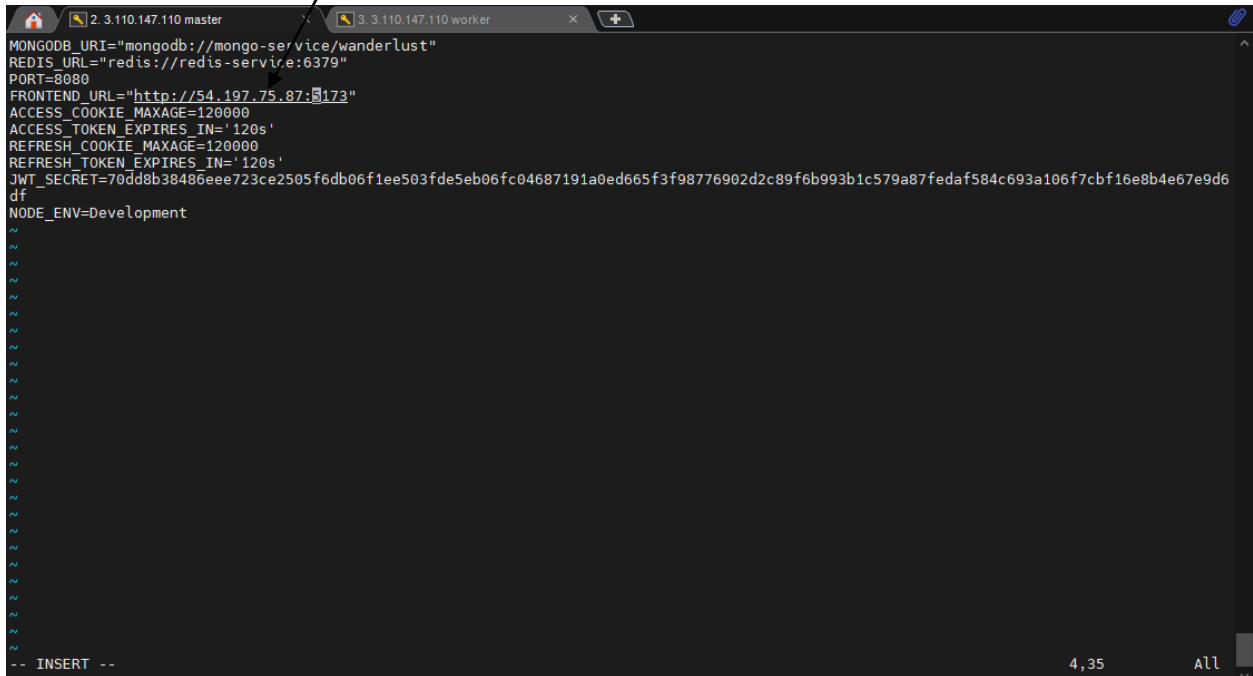
✓ Running

Private IPv4 addresses

172.31.93.124

Public DNS

ec2-54-197-75-87-0

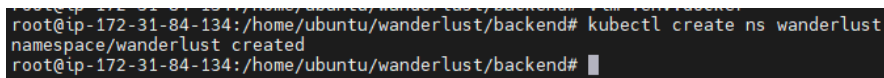


```
MONGODB_URI="mongodb://mongo-service/wanderlust"
REDIS_URL="redis://redis-service:6379"
PORT=8080
FRONTEND_URL="http://54.197.75.87:8173"
ACCESS_COOKIE_MAXAGE=120000
ACCESS_TOKEN_EXPIRES_IN='120s'
REFRESH_COOKIE_MAXAGE=120000
REFRESH_TOKEN_EXPIRES_IN='120s'
JWT_SECRET=70dd8b38486eee723ce2505f6db06f1ee503fde5eb06fc04687191a0ed665f3f98776902d2c89f6b993b1c579a87fedaf584c693a106f7cbf16e8b4e67e9d6df
NODE_ENV=Development
-- INSERT --
```

After that we have to create a namespace named as wanderlust

Why we are creating namespace

Because in a cluster a lot of resources will create we will confuse whom to export whom to work with so we can't organize the things, so that's why we are going to create folder in cluster so that the resources within the group will not overlap with each other and will be in isolation process for which we can easily work according to the services needed (it will group the resources)



```
root@ip-172-31-84-134:/home/ubuntu/wanderlust/backend# kubectl create ns wanderlust
namespace/wanderlust created
root@ip-172-31-84-134:/home/ubuntu/wanderlust/backend#
```

After that check

Kubectl create ns wanderlust

Kubectl get ns


```
root@ip-172-31-84-134:/home/ubuntu/wanderlust/backend# kubectl get ns
NAME                STATUS   AGE
default             Active   19m
kube-node-lease     Active   19m
kube-public         Active   19m
kube-system         Active   19m
wanderlust          Active   74s
root@ip-172-31-84-134:/home/ubuntu/wanderlust/backend#
```

We will see there is a default name space

We have to write another command

`kubectl config set-context --current --namespace wanderlust`

why we are doing this as we don't want to write repeatedly `-n namespace` during any `kubectl get nodes,pods` or `logs` if not in every command we have to write `-n namespace` which is not a good practice.


```
root@ip-172-31-84-134:/home/ubuntu/wanderlust/backend# kubectl config set-context --current --namespace wanderlust
Context "kubernetes-admin@kubernetes" modified.
```

Then write the command as

`kubectl get pods -n kube-system -o wide | grep -i core`

why we are write this command we can check that the dns is working on the master node only

```
root@ip-172-31-84-134:/home/ubuntu/wanderlust/backend# kubectl get pods -n kube-system -o wide | grep -i core
coredns-76f75df574-8rw25      1/1      Running   0          22m      192.168.135.67   ip-172-31-84-134   <none>      <none>
coredns-76f75df574-gk55s     1/1      Running   0          22m      192.168.135.66   ip-172-31-84-134   <none>      <none>
root@ip-172-31-84-134:/home/ubuntu/wanderlust/backend#
```



The core dns is in master node so we have edit in core dns because and replace

Devops-Methodology

```
2. 3.110.147.110 master x 3. 3.110.147.110 worker x +
# Please edit the object below. Lines beginning with a '#' will be ignored,
# and an empty file will abort the edit. If an error occurs while saving this file will be
# reopened with the relevant failures.
#
apiVersion: apps/v1
kind: Deployment
metadata:
  annotations:
    deployment.kubernetes.io/revision: "1"
  creationTimestamp: "2025-05-24T08:31:17Z"
  generation: 1
  labels:
    k8s-app: kube-dns
  name: coredns
  namespace: kube-system
  resourceVersion: "606"
  uid: dddf2f11-a86e-4b63-a211-77ce5018242e
spec:
  progressDeadlineSeconds: 600
  replicas: 4
  revisionHistoryLimit: 10
  selector:
    matchLabels:
      k8s-app: kube-dns
  strategy:
    rollingUpdate:
      maxSurge: 25%
      maxUnavailable: 1
    type: RollingUpdate
  template:
    metadata:
      creationTimestamp: null
      labels:
        k8s-app: kube-dns
    spec:
      affinity:
        podAntiAffinity:
          -- INSERT --
```

20,14 Top

Replace the replica as 4

So that dns will resolve in worker node also

```
2. 3.110.147.110 master x 3. 3.110.147.110 worker x +
schedulerName: default-scheduler
securityContext: {}
serviceAccount: coredns
serviceAccountName: coredns
terminationGracePeriodSeconds: 30
tolerations:
- key: CriticalAddonsOnly
  operator: Exists
- effect: NoSchedule
  key: node-role.kubernetes.io/control-plane
volumes:
- configMap:
    defaultMode: 420
    items:
    - key: Corefile
      path: Corefile
    name: coredns
  name: config-volume
status:
  availableReplicas: 2
  conditions:
  - lastTransitionTime: "2025-05-24T08:31:49Z"
    lastUpdateTime: "2025-05-24T08:31:49Z"
    message: Deployment has minimum availability.
    reason: MinimumReplicasAvailable
    status: "True"
    type: Available
  - lastTransitionTime: "2025-05-24T08:31:22Z"
    lastUpdateTime: "2025-05-24T08:31:52Z"
    message: ReplicaSet "coredns-76f75df574" has successfully progressed.
    reason: NewReplicaSetAvailable
    status: "True"
    type: Progressing
  observedGeneration: 1
  readyReplicas: 2
  replicas: 2
  updatedReplicas: 2
root@ip-172-31-84-134:/home/ubuntu/wanderlust/backend#
```

Best wishes from Devops Methodology

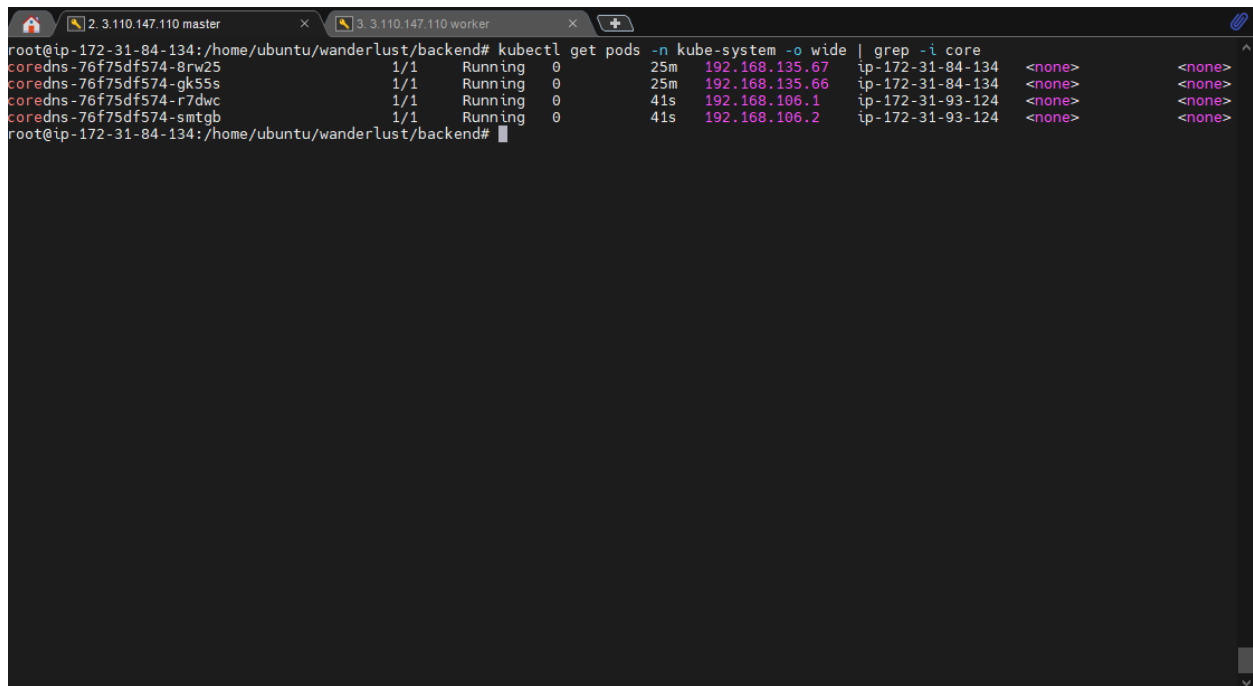
Devops-Methodology

You can check now the dns is pointing to both worker node and master node

DNS translates human-friendly domain names (like `google.com`) into IP addresses (like `142.250.195.78`) so that browsers and apps can load websites and services.

Instead of remembering numbers, we use names — and DNS handles the behind-the-scenes conversion.

So `mongodb-service` can know about master node and worker node as `mongodb-service` is exposed to the application port.



```
root@ip-172-31-84-134:/home/ubuntu/wanderlust/backend# kubectl get pods -n kube-system -o wide | grep -i core
coredns-76f75df574-8rw25      1/1      Running    0          25m      192.168.135.67    ip-172-31-84-134    <none>      <none>
coredns-76f75df574-gk55s      1/1      Running    0          25m      192.168.135.66    ip-172-31-84-134    <none>      <none>
coredns-76f75df574-r7dwc      1/1      Running    0          41s      192.168.106.1     ip-172-31-93-124    <none>      <none>
coredns-76f75df574-smtgb      1/1      Running    0          41s      192.168.106.2     ip-172-31-93-124    <none>      <none>
```

Devops-Methodology

```
root@ip-172-31-84-134:/home/ubuntu/wanderlust/backend# ls
Dockerfile  config      data      package-lock.json  public  server.js  tests  vercel.json
api         controllers models  package.json       routes  services  utils
root@ip-172-31-84-134:/home/ubuntu/wanderlust/backend# cat Dockerfile
# Stage 1
FROM node:21 AS backend-builder

# setup the working dir
WORKDIR /app

# code
COPY . .

# packages install
RUN npm i

# tests
RUN npm run test

# Stage 2
FROM node:21-slim

# setup the working dir
WORKDIR /app

# copy the above stage as compressed
COPY --from=backend-builder /app .

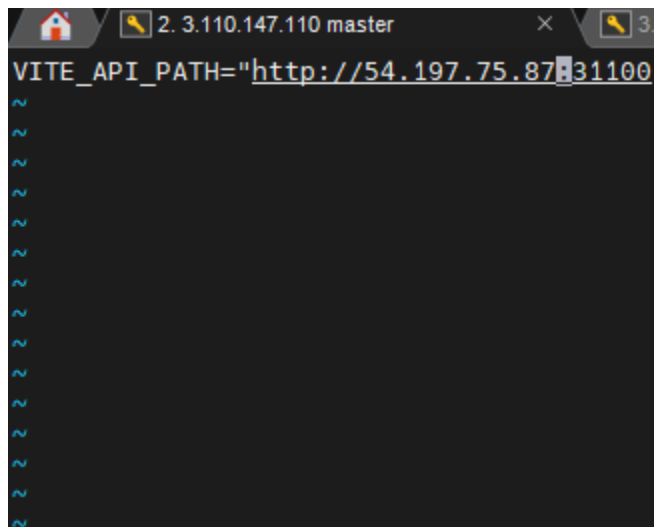
COPY .env.docker .env

# Port
EXPOSE 8080

# App
CMD ["npm", "start"]
root@ip-172-31-84-134:/home/ubuntu/wanderlust/backend#
```

Then go to the backend vi .env.dock

There also change the ip and copy of the workernode ip and paste in that(it's the environment variables we are changing because accord to that system will work)



```
2. 3.110.147.110 master
VITE_API_PATH="http://54.197.75.87:31100"
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
```

Then we have to build the image and push it to docker hub and then there k8s will pull the image and after the deployment it will create the container that is application and we can see our wanderlust website.

`docker build -t premd9/frontend-wanderlust:latest .`

`docker build -t premd9/backend-wanderlust:latest .`

Best wishes from Devops Methodology

Devops-Methodology

```
root@ip-172-31-84-134:/home/ubuntu/wanderlust/backend# ls
Dockerfile  config  data  package-lock.json  public  server.js  tests  vercel.json
api  controllers  models  package.json  routes  services  utils
root@ip-172-31-84-134:/home/ubuntu/wanderlust/backend# cd ..
root@ip-172-31-84-134:/home/ubuntu/wanderlust# cd frontend/
root@ip-172-31-84-134:/home/ubuntu/wanderlust/frontend# ls
Dockerfile  components.json  jest.config.ts  package.json  public/  tailwind.config.js  tsconfig.node.json  vercel.json
README.md  index.html  package-lock.json  postcss.config.js  src/  tsconfig.json  tsconfig.prod.json  vite.config.ts
root@ip-172-31-84-134:/home/ubuntu/wanderlust/frontend# ls
Dockerfile  components.json  jest.config.ts  package.json  public  tailwind.config.js  tsconfig.node.json  vercel.json
README.md  index.html  package-lock.json  postcss.config.js  src  tsconfig.json  tsconfig.prod.json  vite.config.ts
root@ip-172-31-84-134:/home/ubuntu/wanderlust/frontend# vi .env.docker
root@ip-172-31-84-134:/home/ubuntu/wanderlust/frontend# docker build -t premd91/frontend-wanderlust:latest.
```

```
README.md  index.html  package-lock.json  postcss.config.js  src  tsconfig.json  tsconfig.prod.json  vite.config.ts
root@ip-172-31-84-134:/home/ubuntu/wanderlust/frontend# vi .env.docker
root@ip-172-31-84-134:/home/ubuntu/wanderlust/frontend# docker build -t premd91/frontend-wanderlust:latest .
DEPRECATED: The legacy builder is deprecated and will be removed in a future release.
            Install the buildx component to build images with BuildKit:
            https://docs.docker.com/go/buildx/

Sending build context to Docker daemon   918kB
Step 1/11 : FROM node:21 AS frontend-builder
21: Pulling from library/node
c6cf28de8a06: Extracting [=====>] 48.23MB/49.58MB
891494355808: Download complete
6582c62583ef: Download complete
bf2c3e352f3d: Download complete
2b542796ccab: Download complete
4684276225df: Download complete
08750d53428e: Download complete
5d3106ce01c3: Download complete
█
```

Devops-Methodology

```
---> Removed intermediate container 0fc757d9d997
---> 59445f6cf769
Step 5/11 : COPY . .
---> 4597e5157560
Step 6/11 : FROM node:21-slim
21-slim: Pulling from library/node
09f376ebb190: Pull complete
44f5fe9df22b: Pull complete
9f1c458e7f2d: Pull complete
2e82ac84e151: Pull complete
30e51537f486: Pull complete
Digest: sha256:dfc05dee209a1d7adf2ef189bd97396daad4e97c6eaa85778d6f75205ba1b0fb
Status: Downloaded newer image for node:21-slim
---> 539323daf922
Step 7/11 : WORKDIR /app
---> Running in 6ff758730b18
---> Removed intermediate container 6ff758730b18
---> eb9e34eb125f
Step 8/11 : COPY --from=frontend-builder /app .
---> 91fa4bae73b4
Step 9/11 : COPY .env.docker .env.local
---> d2f1e62edb6b
Step 10/11 : EXPOSE 5173
---> Running in b02dfe80b49d
---> Removed intermediate container b02dfe80b49d
---> 9b239b04f0e5
Step 11/11 : CMD ["npm", "run", "dev", "--", "--host"]
---> Running in b6306ee3e530
---> Removed intermediate container b6306ee3e530
---> 5819db2d698c
Successfully built 5819db2d698c
Successfully tagged premd91/frontend-wanderlust:latest
root@ip-172-31-84-134:/home/ubuntu/wanderlust/frontend#
```

```
root@ip-172-31-84-134:/home/ubuntu/wanderlust/frontend# docker images
REPOSITORY          TAG          IMAGE ID      CREATED        SIZE
premd91/frontend-wanderlust  latest      5819db2d698c  53 seconds ago  491MB
<none>              <none>      4597e5157560  About a minute ago  1.48GB
node                21-slim     539323daf922  12 months ago  203MB
node                21         5c8697e82213  12 months ago  1.1GB
root@ip-172-31-84-134:/home/ubuntu/wanderlust/frontend#
```

```
root@ip-172-31-84-134:/home/ubuntu/wanderlust/backend# ls
Dockerfile  config  data  package-lock.json  public  server.js  tests  vercel.json
api         controllers  models  package.json      routes  services  utils
root@ip-172-31-84-134:/home/ubuntu/wanderlust/backend# docker build -t premd91/backend-wanderlust:latest .
```

Devops-Methodology

```
2. 3.110.147.110 master x 3. 3.110.147.110 worker x +
app      93.33    100    50    93.33
server.js 93.33    100    50    93.33    30
app/config 78.57    100    66.66  78.57
db.js     70      100    66.66  70      7-8,14
utils.js  100      100    100    100
app/controllers 38.09  34.54  47.05  38.09
auth-controller.js 8.59    0      0      8.59    13-43,52-94,104-113,118-172,181-232,242-251,256-312,321-370,379-385
posts-controller.js 100     95     100    100    16
app/models 100     100    100    100
post.js   100     100    100    100
user.js   100     100    100    100
app/routes 100     100    100    100
auth.js   100     100    100    100
posts.js  100     100    100    100
app/services 62.5    50     100    62.5
redis.js  62.5    50     100    62.5    9-13,18
app/tests/utils 100     100    100    100
helper-objects.js 100     100    100    100
app/utils 62.5    40     100    57.14
cache-posts.js 43.75  37.5   100    30.76    12-17,23-24,30-31
constants.js 100     100    100    100
cookie_options.js 100     100    100    100
middleware.js 66.66  50     100    62.5    8-9,13
-----
Test Suites: 1 failed, 1 passed, 2 total
Tests:       9 failed, 28 passed, 37 total
Snapshots:   0 total
Time:        50.202 s
Ran all test suites.
---> Removed intermediate container 5344a2c968ac
---> 81d9e7215679
Step 6/11 : FROM node:21-slim
---> 539323daf922
Step 7/11 : WORKDIR /app
---> Using cache
---> eb9e34eb125f
Step 8/11 : COPY --from=backend-builder /app .
```

```
---> Removed intermediate container 5344a2c968ac
---> 81d9e7215679
Step 6/11 : FROM node:21-slim
---> 539323daf922
Step 7/11 : WORKDIR /app
---> Using cache
---> eb9e34eb125f
Step 8/11 : COPY --from=backend-builder /app .
---> 59bed0f41c5c
Step 9/11 : COPY env.docker .env
---> 53391c1bad2b
Step 10/11 : EXPOSE 8080
---> Running in 2f058f0b2824
---> Removed intermediate container 2f058f0b2824
---> dd30027a6832
Step 11/11 : CMD ["npm", "start"]
---> Running in 80acfc9c944e
---> Removed intermediate container 80acfc9c944e
---> b8e8255d71b8
Successfully built b8e8255d71b8
Successfully tagged premd91/backend-wanderlust:latest
root@ip-172-31-84-134:/home/ubuntu/wanderlust/backend#
```

You can check by the command

docker images

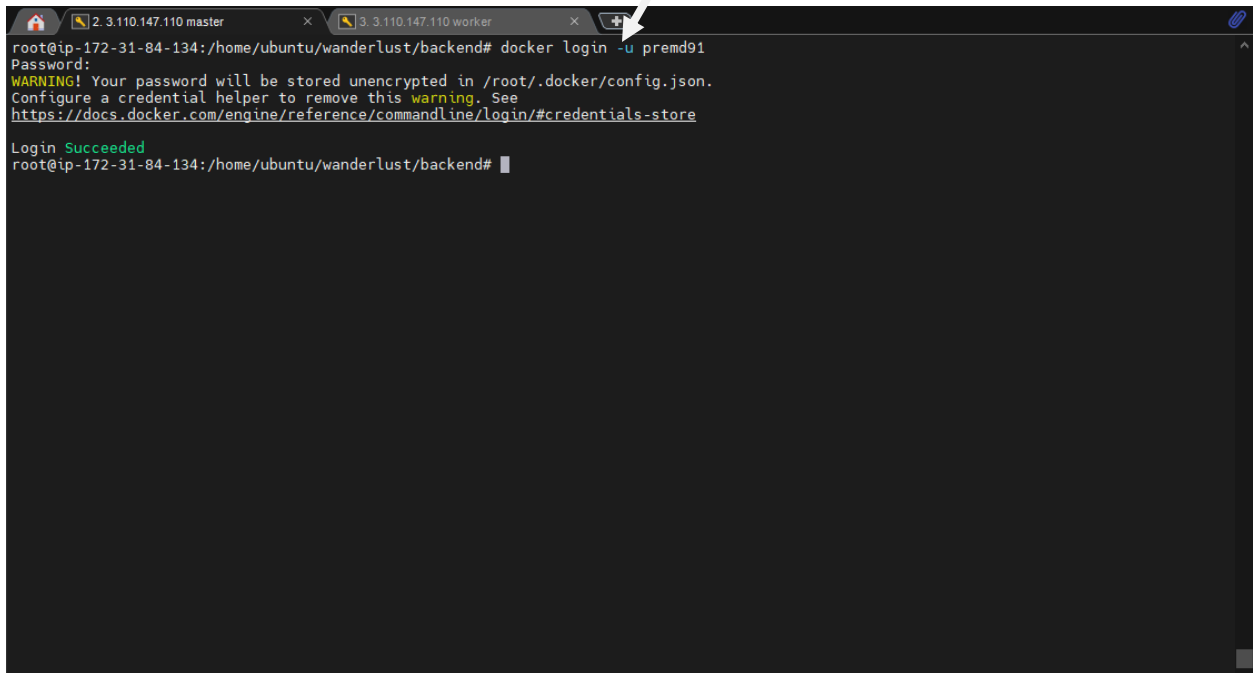
```
Successfully built b8e8255d71b8
Successfully tagged premd91/backend-wanderlust:latest
root@ip-172-31-84-134:/home/ubuntu/wanderlust/backend# docker images
REPOSITORY          TAG          IMAGE ID          CREATED          SIZE
premd91/backend-wanderlust  latest      b8e8255d71b8     30 seconds ago  273MB
<none>              <none>      81d9e7215679     43 seconds ago  1.2GB
premd91/frontend-wanderlust latest      5819db2d698c     5 minutes ago   491MB
<none>              <none>      4597e5157560     5 minutes ago   1.48GB
node                 21-slim     539323daf922     12 months ago   203MB
node                 21          5c8697e82213     12 months ago   1.1GB
root@ip-172-31-84-134:/home/ubuntu/wanderlust/backend#
```

Then we have to push

Devops-Methodology

Before that create “pat” which is the authorization token and will be generated by going to dockerhub>account>security>generate token

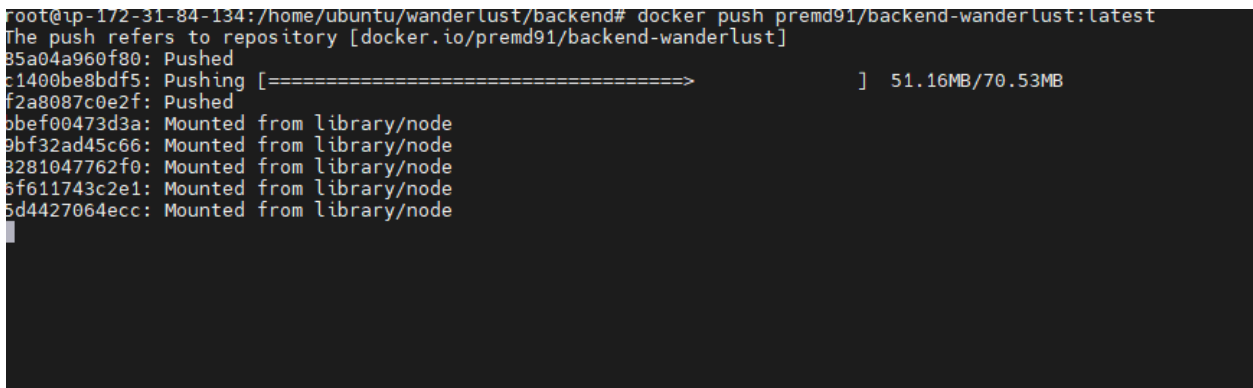
After that docker push



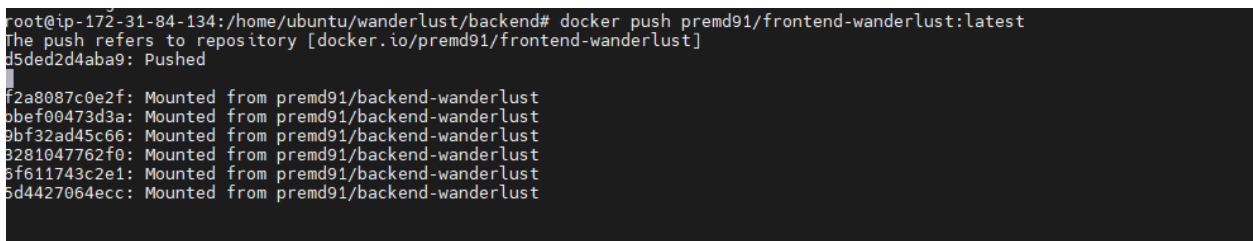
```
root@ip-172-31-84-134:/home/ubuntu/wanderlust/backend# docker login -u premd91
Password:
WARNING! Your password will be stored unencrypted in /root/.docker/config.json.
Configure a credential helper to remove this warning. See
https://docs.docker.com/engine/reference/commandline/login/#credentials-store

Login Succeeded
root@ip-172-31-84-134:/home/ubuntu/wanderlust/backend#
```

Then docker push backend and frontend



```
root@ip-172-31-84-134:/home/ubuntu/wanderlust/backend# docker push premd91/backend-wanderlust:latest
The push refers to repository [docker.io/premd91/backend-wanderlust]
b5a04a960f80: Pushed
c1400be8bdf5: Pushing [=====] 51.16MB/70.53MB
f2a8087c0e2f: Pushed
0bef00473d3a: Mounted from library/node
9bf32ad45c66: Mounted from library/node
3281047762f0: Mounted from library/node
5f611743c2e1: Mounted from library/node
5d4427064ecc: Mounted from library/node
```

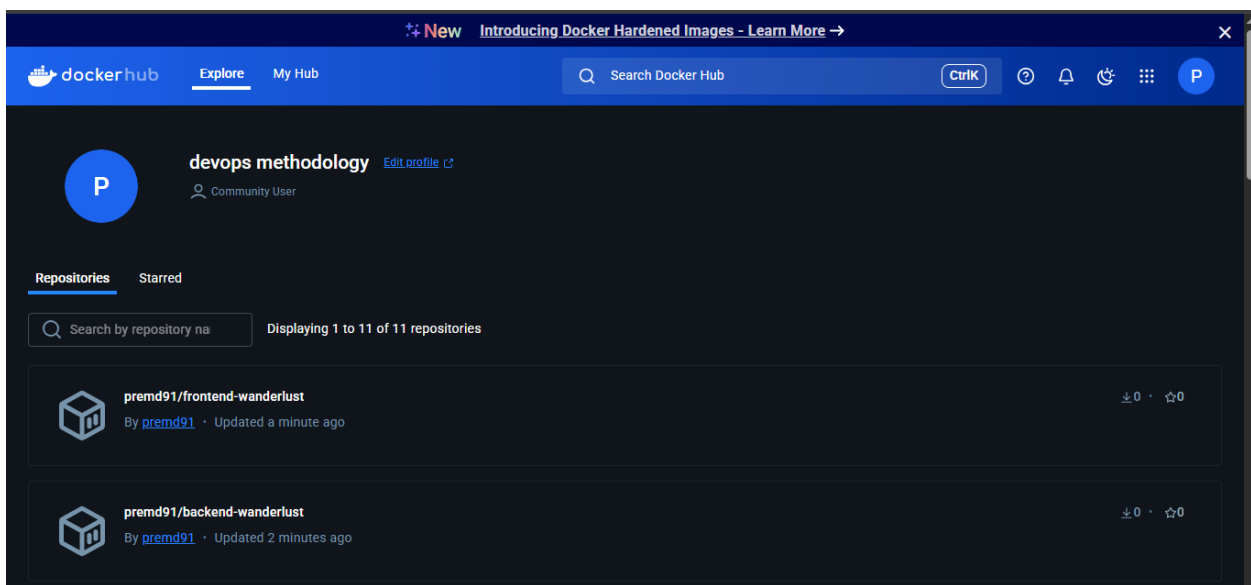


```
root@ip-172-31-84-134:/home/ubuntu/wanderlust/backend# docker push premd91/frontend-wanderlust:latest
The push refers to repository [docker.io/premd91/frontend-wanderlust]
5d5ded2d4aba9: Pushed
f2a8087c0e2f: Mounted from premd91/backend-wanderlust
0bef00473d3a: Mounted from premd91/backend-wanderlust
9bf32ad45c66: Mounted from premd91/backend-wanderlust
3281047762f0: Mounted from premd91/backend-wanderlust
5f611743c2e1: Mounted from premd91/backend-wanderlust
5d4427064ecc: Mounted from premd91/backend-wanderlust
```


Devops-Methodology

```
The push refers to repository [docker.io/premd91/backend-wanderlust]
85a04a960f80: Pushed
c1400be8bdf5: Pushed
f2a8087c0e2f: Pushed
bbef00473d3a: Mounted from library/node
9bf32ad45c66: Mounted from library/node
3281047762f0: Mounted from library/node
6f611743c2e1: Mounted from library/node
5d4427064ecc: Mounted from library/node
latest: digest: sha256:a7628e36a01a09998bd276b6a27c7df7567536dff304ec275cde13d34b4b0065 size: 1992
root@ip-172-31-84-134:/home/ubuntu/wanderlust/backend# docker push premd91/frontend-wanderlust:latest
The push refers to repository [docker.io/premd91/frontend-wanderlust]
d5ded2d4aba9: Pushed
12063f67bbbe: Pushed
f2a8087c0e2f: Mounted from premd91/backend-wanderlust
bbef00473d3a: Mounted from premd91/backend-wanderlust
9bf32ad45c66: Mounted from premd91/backend-wanderlust
3281047762f0: Mounted from premd91/backend-wanderlust
6f611743c2e1: Mounted from premd91/backend-wanderlust
5d4427064ecc: Mounted from premd91/backend-wanderlust
latest: digest: sha256:587999cf39470dd97738ac1340c481cb206a17e8ac4d6366f87764e539a0990b size: 1992
root@ip-172-31-84-134:/home/ubuntu/wanderlust/backend#
```

You can check in the docker hub both for backend and frontend



Next we have to deploy it

cd ../kubernetes

kubectl apply -f persistentVolume.yaml(because we have to create volume for the deployment)

then storage we want we have to claim it

so we will create persistent volume claim

kubectl apply -f persistentVolumeClaim.yaml

Best wishes from Devops Methodology

kubectl apply -f mongodb.yaml

kubectl apply -f redis.yaml

Before that in backend.yaml and frontend.yaml

We have to change the docker image which we have pushed to dockerhub

kubectl apply -f backend.yaml

kubectl apply -f frontend.yaml

```
root@ip-172-31-84-134:/home/ubuntu/wanderlust/kubernetes# ls
README.md  backend.yaml  kubeadm.md  persistentVolume.yaml  redis.yaml  script2.sh
assets     frontend.yaml  mongodb.yaml  persistentVolumeClaim.yaml  script1.sh
root@ip-172-31-84-134:/home/ubuntu/wanderlust/kubernetes# kubectl apply -f persistentVolume.yaml
persistentvolume/mongo-pv created
root@ip-172-31-84-134:/home/ubuntu/wanderlust/kubernetes# kubectl apply -f persistentVolumeClaim.yaml
persistentvolumeclaim/mongo-pvc created
root@ip-172-31-84-134:/home/ubuntu/wanderlust/kubernetes# kubectl apply -f mongodb.yaml
deployment.apps/mongo-deployment created
service/mongo-service created
root@ip-172-31-84-134:/home/ubuntu/wanderlust/kubernetes#
```

Then we can check by

Kubectl get pods

Devops-Methodology

```
root@ip-172-31-84-134:/home/ubuntu/wanderlust/kubernetes# kubectl get pods
NAME                                READY   STATUS    RESTARTS   AGE
mongo-deployment-9d8749988-nghgn    1/1     Running   0           24s
root@ip-172-31-84-134:/home/ubuntu/wanderlust/kubernetes# kubectl get pods -n wanderlust
NAME                                READY   STATUS    RESTARTS   AGE
mongo-deployment-9d8749988-nghgn    1/1     Running   0           34s
root@ip-172-31-84-134:/home/ubuntu/wanderlust/kubernetes#
```

```
root@ip-172-31-84-134:/home/ubuntu/wanderlust/kubernetes# ls
README.md  backend.yaml  kubeadm.md  persistentVolume.yaml  redis.yaml  script2.sh
assets     frontend.yaml  mongodb.yaml  persistentVolumeClaim.yaml  script1.sh
root@ip-172-31-84-134:/home/ubuntu/wanderlust/kubernetes# kubectl apply -f persistentVolume.yaml
persistentvolume/mongo-pv created
root@ip-172-31-84-134:/home/ubuntu/wanderlust/kubernetes# kubectl apply -f persistentVolumeClaim.yaml
persistentvolumeclaim/mongo-pvc created
root@ip-172-31-84-134:/home/ubuntu/wanderlust/kubernetes# kubectl apply -f mongodb.yaml
deployment.apps/mongo-deployment created
service/mongo-service created
root@ip-172-31-84-134:/home/ubuntu/wanderlust/kubernetes# kubectl get pods
NAME                                READY   STATUS    RESTARTS   AGE
mongo-deployment-9d8749988-nghgn    1/1     Running   0           24s
root@ip-172-31-84-134:/home/ubuntu/wanderlust/kubernetes# kubectl get pods -n wanderlust
NAME                                READY   STATUS    RESTARTS   AGE
mongo-deployment-9d8749988-nghgn    1/1     Running   0           34s
root@ip-172-31-84-134:/home/ubuntu/wanderlust/kubernetes# kubectl apply -f redis.yaml
deployment.apps/redis-deployment created
service/redis-service created
root@ip-172-31-84-134:/home/ubuntu/wanderlust/kubernetes# kubectl get pods
NAME                                READY   STATUS    RESTARTS   AGE
mongo-deployment-9d8749988-nghgn    1/1     Running   0           2m7s
redis-deployment-9fbf97b49-xxmjk    1/1     Running   0           5s
root@ip-172-31-84-134:/home/ubuntu/wanderlust/kubernetes#
```

Devops-Methodology

```
2. 3.110.147.110 master x 3. 3.110.147.110 worker x +
apiVersion: apps/v1
kind: Deployment
metadata:
  name: frontend-deployment
  namespace: wanderlust
  labels:
    app: frontend
spec:
  replicas: 1
  selector:
    matchLabels:
      app: frontend
  template:
    metadata:
      name: frontend
      labels:
        app: frontend
    spec:
      containers:
        - name: frontend
          image: premd91/wanderlust-frontend-beta:latest
          ports:
            - containerPort: 5173
---
apiVersion: v1
kind: Service
metadata:
  name: frontend-service
  namespace: wanderlust
-- INSERT --
```

21,57 Top

```
2. 3.110.147.110 master x 3. 3.110.147.110 worker x +
apiVersion: apps/v1
kind: Deployment
metadata:
  name: backend-deployment
  namespace: wanderlust
  labels:
    app: backend
spec:
  replicas: 1
  selector:
    matchLabels:
      app: backend
  template:
    metadata:
      name: backend
      labels:
        app: backend
    spec:
      containers:
        - name: backend
          image: premd91/wanderlust-backend-beta:latest
          ports:
            - containerPort: 8080
---
apiVersion: v1
kind: Service
metadata:
  name: backend-service
  namespace: wanderlust
-- INSERT --
```

21,56 Top

Best wishes from Devops Methodology

Devops-Methodology

The screenshot shows the AWS Management Console interface. On the left, the navigation menu is visible with categories like EC2, Images, Elastic Block Store, and Network & Security. The main content area displays the 'Inbound rules' for a specific security group (sg-05206cd4aeb02a6a0). A green notification banner at the top indicates that the inbound security group rules were successfully modified. Below this, there are tabs for 'Inbound rules', 'Outbound rules', 'Sharing - new', 'VPC associations - new', and 'Tags'. The 'Inbound rules' tab is active, showing a table with 6 rules. Each rule includes a checkbox, a name, a security group rule ID, IP version, type, protocol, port range, and source. The rules are: HTTP (port 80), Custom TCP (port 6443), Custom TCP (port 31000), Custom TCP (port 31100), and SSH (port 22). All rules have a source of 0.0.0.0/0.

<input type="checkbox"/>	Name	Security group rule ID	IP version	Type	Protocol	Port range	Source
<input type="checkbox"/>	-	sgr-0bd422e58a49259cb	IPv4	HTTP	TCP	80	0.0.0.0/0
<input type="checkbox"/>	-	sgr-051959345e26b69fc	IPv4	Custom TCP	TCP	31000	0.0.0.0/0
<input type="checkbox"/>	-	sgr-03cd78fc9d990851c	IPv4	Custom TCP	TCP	6443	0.0.0.0/0
<input type="checkbox"/>	-	sgr-09f736d56701790b8	IPv4	HTTPS	TCP	443	0.0.0.0/0
<input type="checkbox"/>	-	sgr-0f46cc8ef06d44796	IPv4	Custom TCP	TCP	31100	0.0.0.0/0
<input type="checkbox"/>	-	sgr-000e4e087e7bcf36a	IPv4	SSH	TCP	22	0.0.0.0/0

The screenshot shows the 'Edit inbound rules' page in the AWS Management Console. The page title is 'Edit inbound rules' with an 'Info' icon. Below the title, a message states: 'Inbound rules control the incoming traffic that's allowed to reach the instance.' The main content area is divided into two sections: 'Inbound rules' and 'Info'. The 'Inbound rules' section contains a table with columns for 'Security group rule ID', 'Type', 'Protocol', 'Port range', 'Source', and 'Description - optional'. Each row represents a rule, and there are buttons for 'Delete' and 'Add rule'. The 'Info' section contains a warning message: 'Rules with source of 0.0.0.0/0 or ::/0 allow all IP addresses to access your instance. We recommend setting security group rules to allow access from known IP addresses only.'

Security group rule ID	Type	Protocol	Port range	Source	Description - optional	Info
sgr-0bd422e58a49259cb	HTTP	TCP	80	Anywh...		Delete
sgr-03cd78fc9d990851c	Custom TCP	TCP	6443	Anywh...		Delete
sgr-09f736d56701790b8	HTTPS	TCP	443	Anywh...		Delete
sgr-000e4e087e7bcf36a	SSH	TCP	22	Anywh...		Delete
-	Custom TCP	TCP	31000	Anywh...	frontend	Delete
-	Custom TCP	TCP	31100	Anywh...		Delete

```
mongo-deployment-9d8749988-nghgn 1/1 Running 0 2m/s
redis-deployment-9fbf97b49-xxmjk 1/1 Running 0 5s
root@ip-172-31-84-134:/home/ubuntu/wanderlust/kubernetes# vi frontend.yaml
root@ip-172-31-84-134:/home/ubuntu/wanderlust/kubernetes# vi backend.yaml
root@ip-172-31-84-134:/home/ubuntu/wanderlust/kubernetes# vi frontend.yaml
root@ip-172-31-84-134:/home/ubuntu/wanderlust/kubernetes# kubectl apply -f frontend.yaml
deployment.apps/frontend-deployment created
service/frontend-service created
root@ip-172-31-84-134:/home/ubuntu/wanderlust/kubernetes# kubectl apply -f backend.yaml
deployment.apps/backend-deployment created
service/backend-service created
root@ip-172-31-84-134:/home/ubuntu/wanderlust/kubernetes#
```

Best wishes from Devops Methodology

I have some error that's why the application is not accessing in port no 31000

What was the error

```
root@ip-172-31-84-134:/home/ubuntu/wanderlust/kubernetes# kubectl get all
NAME                                READY    STATUS              RESTARTS   AGE
pod/backend-deployment-5cffb4b849-fwdj9    0/1      ImagePullBackOff    0           17s
pod/frontend-deployment-7f756db7b9-tmqzb    0/1      ErrImagePull        0           37s
pod/mongo-deployment-9d8749988-nghgn        1/1      Running             0           10m
pod/redis-deployment-9fbf97b49-xxmjk        1/1      Running             0           8m11s

NAME                                TYPE                CLUSTER-IP      EXTERNAL-IP      PORT(S)          AGE
service/backend-service             NodePort            10.97.195.48    <none>           8080:31100/TCP   17s
service/frontend-service            NodePort            10.108.248.71   <none>           5173:31000/TCP   37s
service/mongo-service               ClusterIP           10.102.92.89    <none>           27017/TCP        10m
service/redis-service               ClusterIP           10.109.120.127  <none>           6379/TCP         8m11s

NAME                                READY    UP-TO-DATE    AVAILABLE   AGE
deployment.apps/backend-deployment    0/1      1              0           17s
deployment.apps/frontend-deployment    0/1      1              0           37s
deployment.apps/mongo-deployment       1/1      1              1           10m
deployment.apps/redis-deployment       1/1      1              1           8m11s

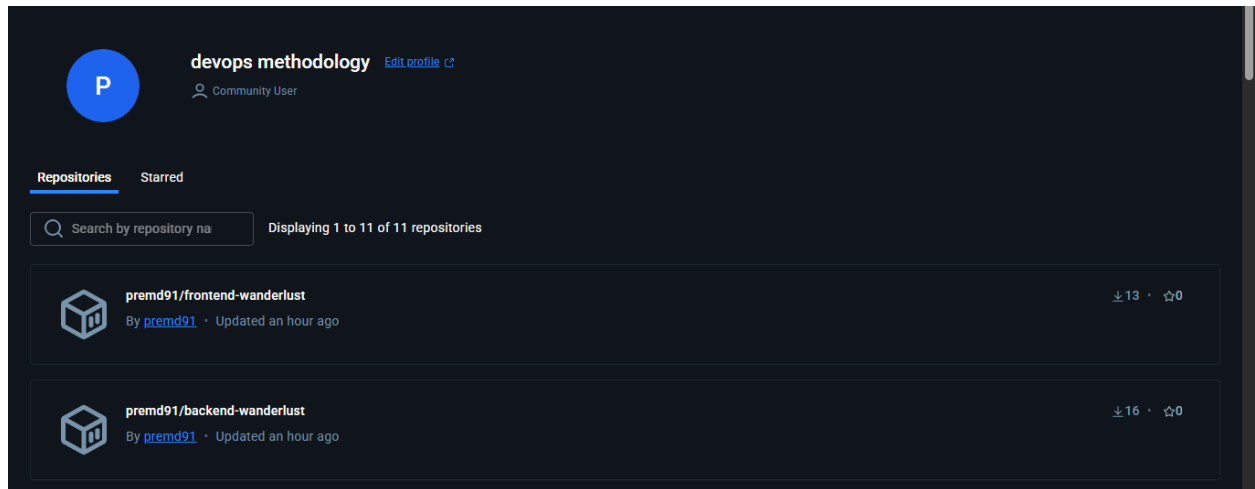
NAME                                DESIRED    CURRENT    READY    AGE
replicaset.apps/backend-deployment-5cffb4b849    1          1          0        17s
replicaset.apps/frontend-deployment-7f756db7b9    1          1          0        37s
replicaset.apps/mongo-deployment-9d8749988        1          1          1        10m
replicaset.apps/redis-deployment-9fbf97b49        1          1          1        8m11s
root@ip-172-31-84-134:/home/ubuntu/wanderlust/kubernetes#
```

Actually the image which I have build and tag it was not created the latest image for and in backend and frontend.yaml and I have updated as :latest which it can't pull the image as a devops we have to know the error but finally it was corrected

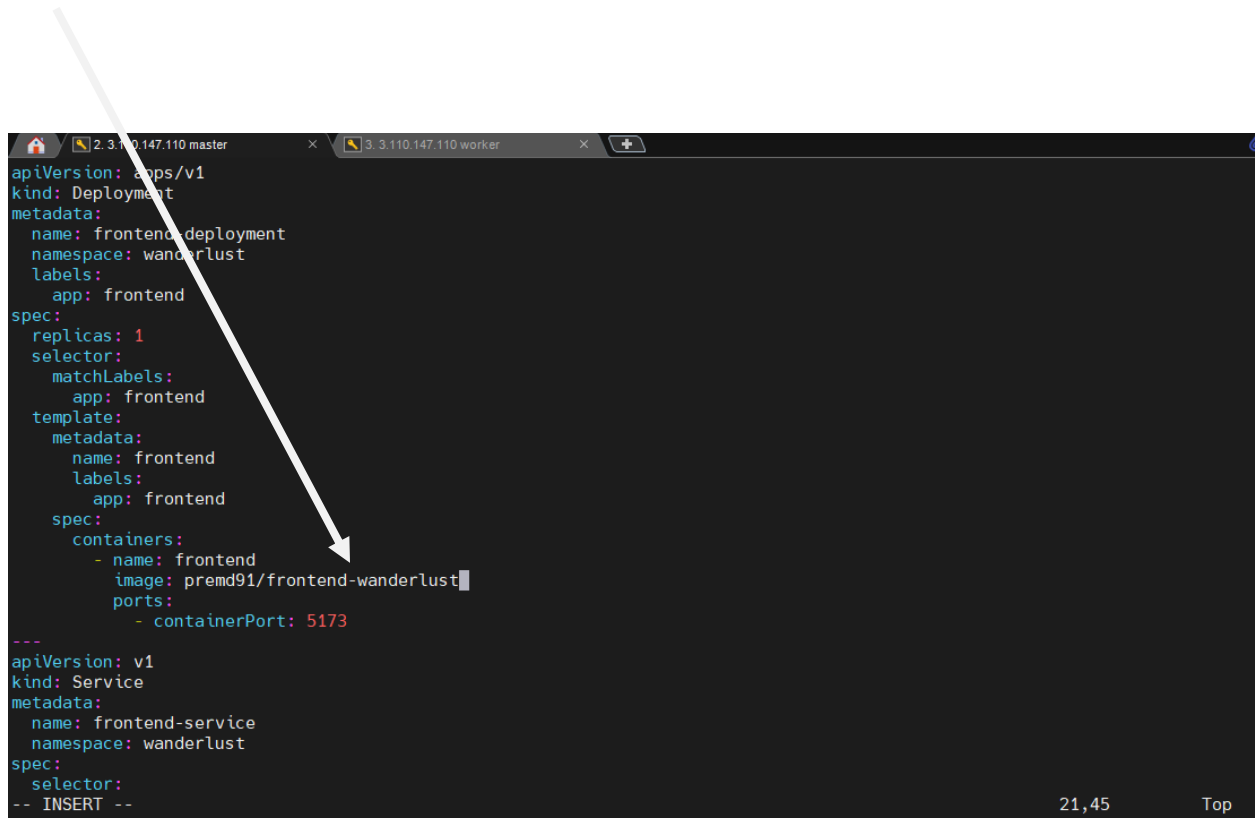
```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: backend-deployment
  namespace: wanderlust
  labels:
    app: backend
spec:
  replicas: 1
  selector:
    matchLabels:
      app: backend
  template:
    metadata:
      name: backend
      labels:
        app: backend
    spec:
      containers:
        - name: backend
          image: premd91/wanderlust-backend-beta:latest
          ports:
            - containerPort: 8080
---
apiVersion: v1
kind: Service
metadata:
  name: backend-service
  namespace: wanderlust
spec:
  type: NodePort
  backend.yaml" 38L, 667B
```

That's why the error was coming

But in dockerhub the latest was not added so I just eliminate it



Here's the correct format



After that check my container is creating

```
root@ip-172-31-84-134:/home/ubuntu/wanderlust/kubernetes# kubectl get all
NAME                                READY    STATUS              RESTARTS   AGE
pod/backend-deployment-67b5d55b65-p5lk6    0/1     ContainerCreating   0           5s
pod/backend-deployment-c6994bb47-7wm9g      0/1     ImagePullBackOff    0           4m35s
pod/frontend-deployment-84dd648ccc-wsbq6    1/1     Running             0           15s
pod/mongo-deployment-9d8749988-nghgn        1/1     Running             0           32m
pod/redis-deployment-9fbf97b49-xxmjk        1/1     Running             0           30m

NAME                                TYPE                CLUSTER-IP      EXTERNAL-IP   PORT(S)          AGE
service/backend-service             NodePort            10.97.195.48    <none>        8080:31100/TCP   22m
service/frontend-service            NodePort            10.108.248.71   <none>        5173:31000/TCP   22m
service/mongo-service               ClusterIP           10.102.92.89    <none>        27017/TCP        32m
service/redis-service               ClusterIP           10.109.120.127  <none>        6379/TCP         30m

NAME                                READY    UP-TO-DATE    AVAILABLE   AGE
```

After sometime it shows the container was created so

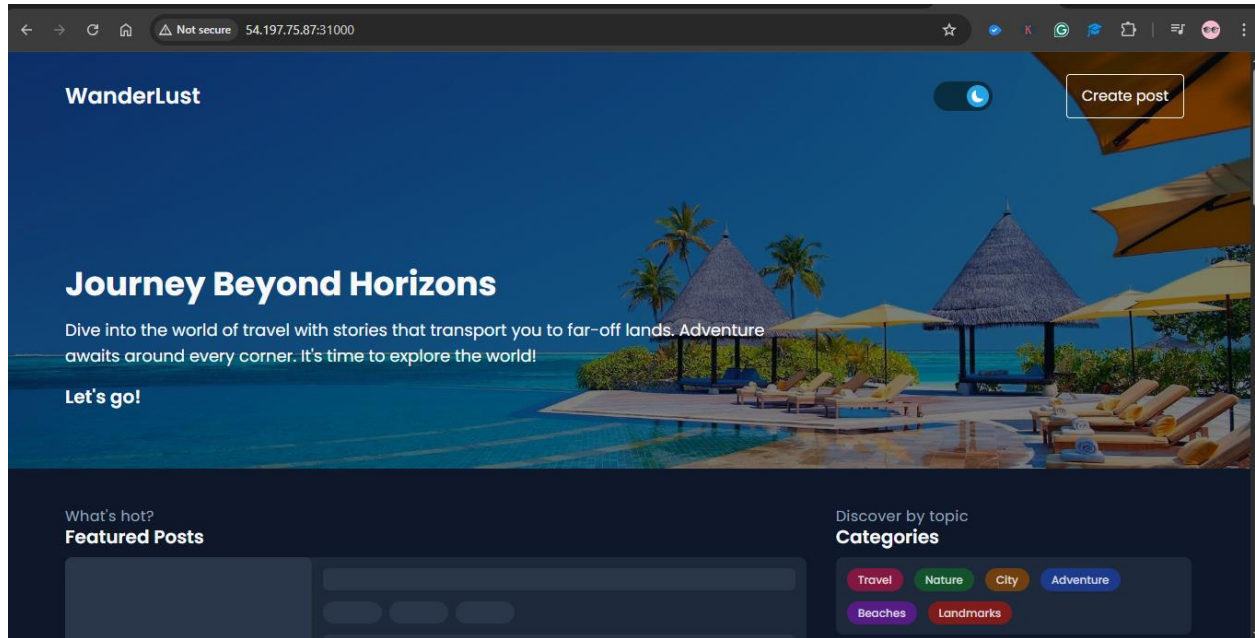
```
root@ip-172-31-84-134:/home/ubuntu/wanderlust/kubernetes# kubectl get all
NAME                                READY    STATUS    RESTARTS   AGE
pod/backend-deployment-67b5d55b65-p5lk6    1/1     Running   0           30s
pod/frontend-deployment-84dd648ccc-wsbq6    1/1     Running   0           40s
pod/mongo-deployment-9d8749988-nghgn        1/1     Running   0           33m
pod/redis-deployment-9fbf97b49-xxmjk        1/1     Running   0           30m

NAME                                TYPE                CLUSTER-IP      EXTERNAL-IP   PORT(S)          AGE
service/backend-service             NodePort            10.97.195.48    <none>        8080:31100/TCP   23m
service/frontend-service            NodePort            10.108.248.71   <none>        5173:31000/TCP   23m
service/mongo-service               ClusterIP           10.102.92.89    <none>        27017/TCP        33m
service/redis-service               ClusterIP           10.109.120.127  <none>        6379/TCP         30m

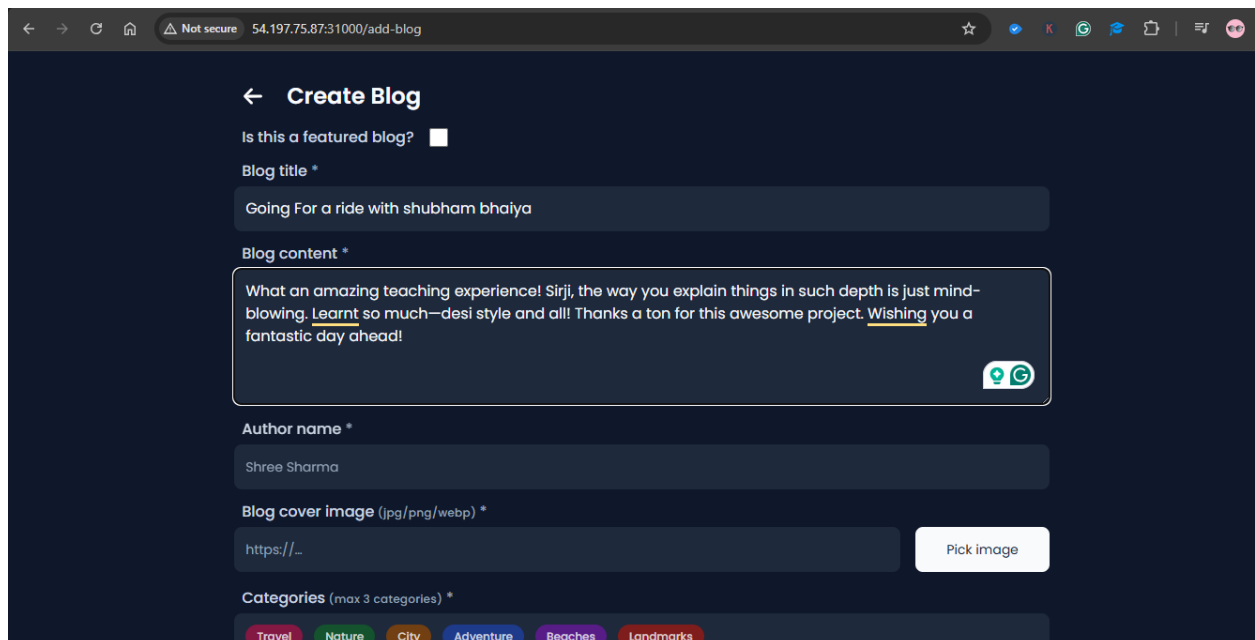
NAME                                READY    UP-TO-DATE    AVAILABLE   AGE
deployment.apps/backend-deployment    1/1     1              1           23m
deployment.apps/frontend-deployment    1/1     1              1           23m
deployment.apps/mongo-deployment       1/1     1              1           33m
deployment.apps/redis-deployment       1/1     1              1           30m

NAME                                DESIRED    CURRENT    READY    AGE
replicaset.apps/backend-deployment-54f669f69d    0          0          0        11m
replicaset.apps/backend-deployment-5cffb4b849    0          0          0        23m
replicaset.apps/backend-deployment-67b5d55b65    1          1          1        30s
replicaset.apps/backend-deployment-c6994bb47     0          0          0         5m
replicaset.apps/frontend-deployment-7f6f849fdc    0          0          0        5m12s
replicaset.apps/frontend-deployment-7f756db7b9    0          0          0        23m
replicaset.apps/frontend-deployment-84dd648ccc    1          1          1        40s
replicaset.apps/frontend-deployment-9f7d74586     0          0          0        12m
replicaset.apps/mongo-deployment-9d8749988        1          1          1        33m
replicaset.apps/redis-deployment-9fbf97b49        1          1          1        30m
root@ip-172-31-84-134:/home/ubuntu/wanderlust/kubernetes#
```

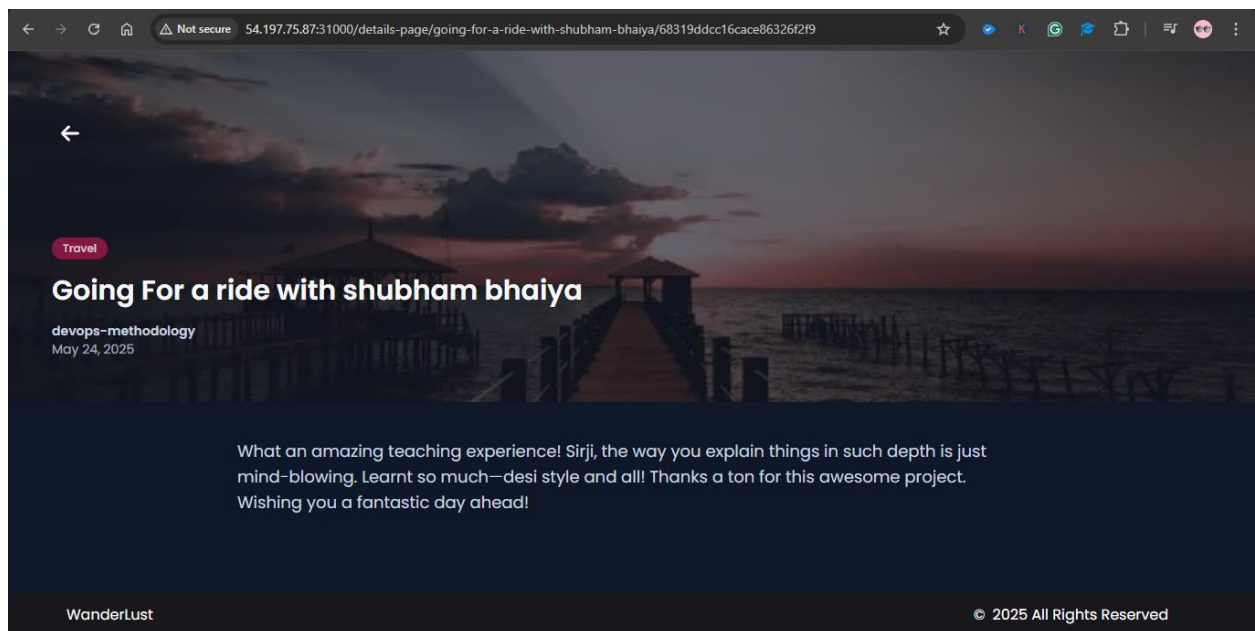
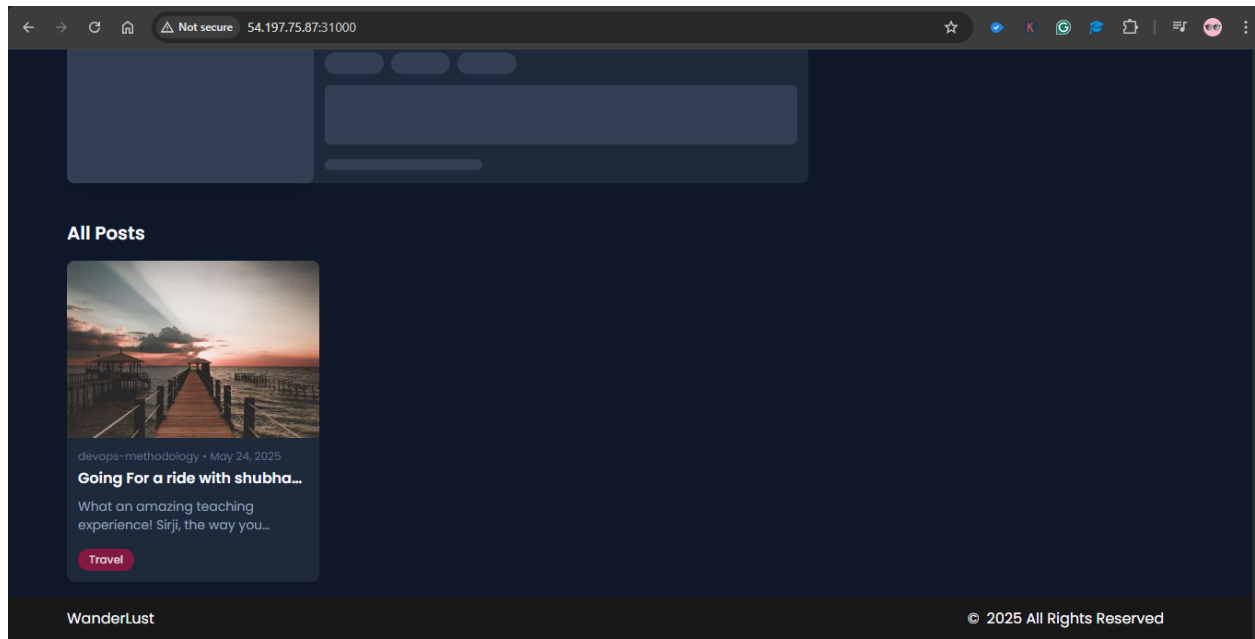

I have just access workernode ip:31000(for the frontend)



Also create the blog for devops wale bhaiya urff shubham bhaiya



Devops-Methodology



Best wishes from Devops Methodology

Workernodeip:31100(for backend)

