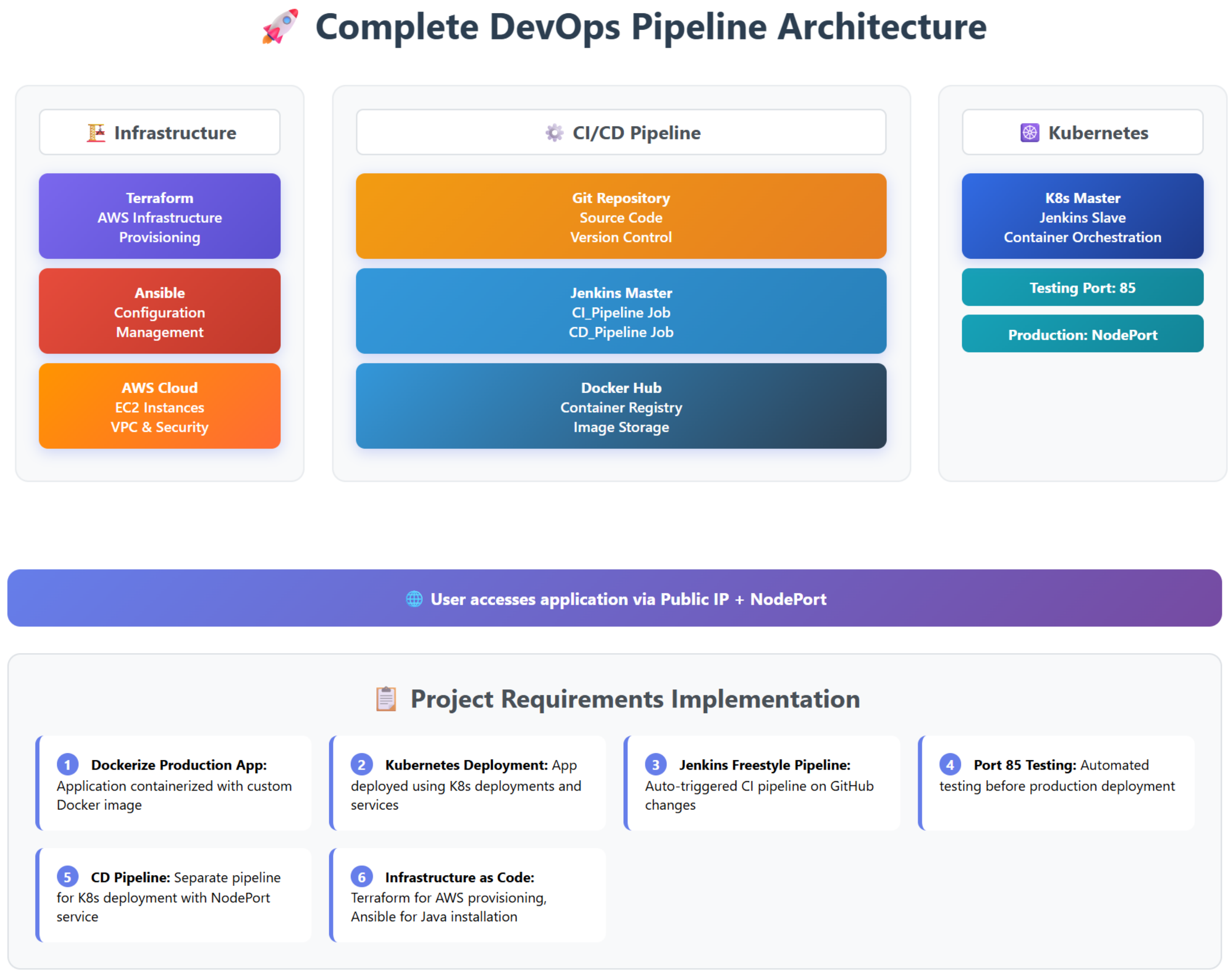


🚀 Complete DevOps Pipeline Architecture

🏗️ Infrastructure

Terraform
AWS Infrastructure Provisioning

Ansible
Configuration Management

AWS Cloud
EC2 Instances
VPC & Security

⚙️ CI/CD Pipeline

Git Repository
Source Code
Version Control

Jenkins Master
CI_Pipeline Job
CD_Pipeline Job

Docker Hub
Container Registry
Image Storage

☸️ Kubernetes

K8s Master
Jenkins Slave
Container Orchestration

Testing Port: 85

Production: NodePort

🌐 **User accesses application via Public IP + NodePort**

📋 Project Requirements Implementation

1 **Dockerize Production App:** Application containerized with custom Docker image

2 **Kubernetes Deployment:** App deployed using K8s deployments and services

3 **Jenkins Freestyle Pipeline:** Auto-triggered CI pipeline on GitHub changes

4 **Port 85 Testing:** Automated testing before production deployment

5 **CD Pipeline:** Separate pipeline for K8s deployment with NodePort service

6 **Infrastructure as Code:** Terraform for AWS provisioning, Ansible for Java installation

Create a Role with AdministratorAccess and attach to user till resource creation is done.

This Terraform configuration provisions a complete AWS infrastructure setup for Jenkins CI/CD and Kubernetes cluster deployment.
Architecture Overview
The infrastructure creates:

1. A VPC with DNS support in the us-east-2 region
2. Two Public Subnets across different availability zones (us-east-2a, us-east-2b) for HA
3. One Jenkins Server for CI/CD operations which will servers as jenkins master and pass job to jenkins worker aka K8s master
4. Three Kubernetes Nodes for container orchestration (out of 3 k8s nodes 2 will be worker nodes and 1 will be master node)
5. Security Groups with appropriate access rules
6. Internet Gateway and routing for public access

Details about each resources:-

1. VPC: 10.0.0.0/16 CIDR block with DNS hostnames and support enabled
2. Public Subnet 1: 10.0.1.0/24 in us-east-2a
3. Public Subnet 2: 10.0.2.0/24 in us-east-2b
4. Internet Gateway: Provides internet access to public subnets
5. Route Table: Routes all traffic (0.0.0.0/0) through the internet gateway

6. Security Groups for our instances:-

a.Jenkins Security Group (Jenkins-sg):-

Inbound Rules:

HTTP (80): Web traffic
HTTPS (443): Secure web traffic
SSH (22): Remote access
Port 8080: Jenkins web interface

Outbound Rules: All traffic allowed

b. Kubernetes Security Group (K8s-sg):-

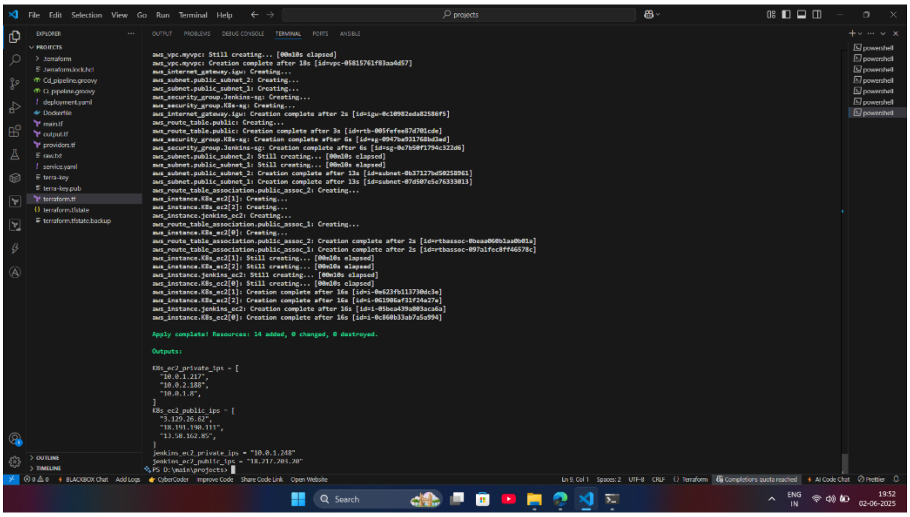
Inbound/Outbound Rules: All traffic allowed (temporary for installation)
Note: This is intentionally I put all traffic allow during setup phase and should be restricted in production.

EC2 Instances
1. Jenkins Server

a.Instance Type: t2.micro
b.AMI: ami-04f167a56786e4b09 (ubuntu)

2. Kubernetes Cluster

a. Node Count: 3 instances
b. Instance Type: t3.medium
c. AMI: ami-04f167a56786e4b09 (Amazon Linux 2)
d. Naming: k8s-node-1, k8s-node-2, k8s-node-3



```
aws_vpc.myvpc: Still creating... [00m10s elapsed]
aws_vpc.myvpc: Creation complete after 18s [id=vpc-05815761f83aa4d57]
aws_internet_gateway.igw: Creating...
aws_subnet.public_subnet_2: Creating...
aws_subnet.public_subnet_1: Creating...
aws_security_group.Jenkins-sg: Creating...
aws_security_group.K8s-sg: Creating...
aws_internet_gateway.igw: Creation complete after 2s [id=igw-0c10982eda82586f5]
aws_route_table.public: Creating...
aws_route_table.public: Creation complete after 3s [id=rtb-005fefee87d701cde]
aws_security_group.K8s-sg: Creation complete after 6s [id=sg-0947ba931768bd3ed]
aws_security_group.Jenkins-sg: Creation complete after 6s [id=sg-0e7b50f1794c322d6]
aws_subnet.public_subnet_2: Still creating... [00m10s elapsed]
aws_subnet.public_subnet_1: Still creating... [00m10s elapsed]
aws_subnet.public_subnet_2: Creation complete after 13s [id=subnet-0b37127bd50258961]
aws_subnet.public_subnet_1: Creation complete after 13s [id=subnet-07d507e5e76333013]
aws_route_table_association.public_assoc_2: Creating...
aws_instance.K8s_ec2[1]: Creating...
aws_instance.K8s_ec2[2]: Creating...
aws_instance.jenkins_ec2: Creating...
aws_route_table_association.public_assoc_1: Creating...
aws_instance.K8s_ec2[0]: Creating...
aws_route_table_association.public_assoc_2: Creation complete after 2s [id=rtbassoc-0beaa060b1aa0b01a]
aws_route_table_association.public_assoc_1: Creation complete after 2s [id=rtbassoc-097a1fec8ff46578c]
aws_instance.K8s_ec2[1]: Still creating... [00m10s elapsed]
aws_instance.K8s_ec2[2]: Still creating... [00m10s elapsed]
aws_instance.jenkins_ec2: Still creating... [00m10s elapsed]
aws_instance.K8s_ec2[0]: Still creating... [00m10s elapsed]
aws_instance.K8s_ec2[1]: Creation complete after 16s [id=i-0e623fb113730dc3e]
aws_instance.K8s_ec2[2]: Creation complete after 16s [id=i-061906af31f24a27e]
aws_instance.jenkins_ec2: Creation complete after 16s [id=i-05bea439a803aca6a]
aws_instance.K8s_ec2[0]: Creation complete after 16s [id=i-0c860b33ab7a5a994]

Apply complete! Resources: 14 added, 0 changed, 0 destroyed.

Outputs:

K8s_ec2_private_ips = [
  "10.0.1.217",
  "10.0.2.188",
  "10.0.1.8",
]
K8s_ec2_public_ips = [
  "3.129.26.62",
  "18.191.190.111",
  "13.58.162.85",
]
jenkins_ec2_private_ips = "10.0.1.248"
jenkins_ec2_public_ips = "18.217.203.20"
PS D:\main\projects>
```

# Ansible Jenkins & Kubernetes Cluster Setup

This Ansible configuration automates the installation and configuration of Jenkins CI/CD server and a Kubernetes cluster across multiple hosts.

The playbook manages three types of hosts in hosts file:
- localhost: Jenkins CI/CD server
- KMaster: Kubernetes master node (1 node)
- KWorker: Kubernetes worker nodes (2 nodes)

cat /etc/ansible/hosts

KMaster (Kubernetes Master)
- host-1: `10.0.1.217` - Controls the Kubernetes cluster

KWorker (Kubernetes Workers)
- host-2: `10.0.2.188` - Worker node in subnet 2
- host-3: `10.0.1.8` - Worker node in subnet 1

Global Variables
- User: `ubuntu` - SSH user for all remote hosts
- SSH Key: `/etc/ansible/terra-key` - Private key for authentication

## Playbook Structure

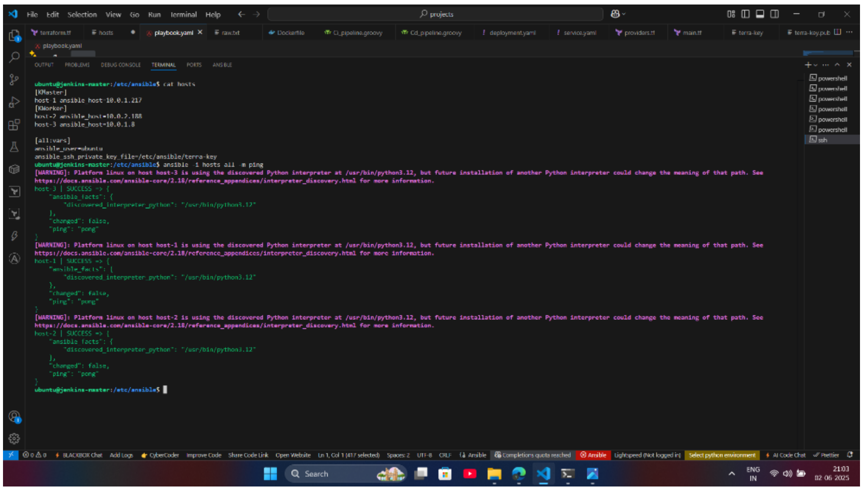
1. Jenkins Installation (localhost)
Target: localhost
Roles:
- `java` - Installs Java runtime
- `jenkins` - Installs and configures Jenkins

2. Kubernetes Master Setup
Target: KMaster group
Roles:
- `java` - Java runtime for Kubernetes components
- `docker` - Container runtime
- `kubernetes` - Master node configuration

3. Kubernetes Workers Setup
Target: KWorker group
Roles:
- `package-update` - Updates system packages
- `docker` - Container runtime
- `kubernetes` - Worker node configuration

SSH Access: Verify connectivity to all hosts:-

ansible -i hosts all -m ping



```
ubuntu@jenkins-master:/etc/ansible$ cat hosts
[KMaster]
host-1 ansible_host=10.0.1.217
[KWorker]
host-2 ansible_host=10.0.2.188
host-3 ansible_host=10.0.1.8

[all:vars]
ansible_user=ubuntu
ansible_ssh_private_key_file=/etc/ansible/terra-key
ubuntu@jenkins-master:/etc/ansible$ ansible -i hosts all -m ping
[WARNING]: Platform linux on host host-3 is using the discovered Python interpreter at /usr/bin/python3.12, but future installation of another Python interpreter could change the meaning of that path. See
https://docs.ansible.com/ansible-core/2.18/reference_appendices/interpreter_discovery.html for more information.
host-3 | SUCCESS => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python3.12"
    },
    "changed": false,
    "ping": "pong"
}
[WARNING]: Platform linux on host host-1 is using the discovered Python interpreter at /usr/bin/python3.12, but future installation of another Python interpreter could change the meaning of that path. See
https://docs.ansible.com/ansible-core/2.18/reference_appendices/interpreter_discovery.html for more information.
host-1 | SUCCESS => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python3.12"
    },
    "changed": false,
    "ping": "pong"
}
[WARNING]: Platform linux on host host-2 is using the discovered Python interpreter at /usr/bin/python3.12, but future installation of another Python interpreter could change the meaning of that path. See
https://docs.ansible.com/ansible-core/2.18/reference_appendices/interpreter_discovery.html for more information.
host-2 | SUCCESS => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python3.12"
    },
    "changed": false,
    "ping": "pong"
}
ubuntu@jenkins-master:/etc/ansible$
```

Required Ansible Roles

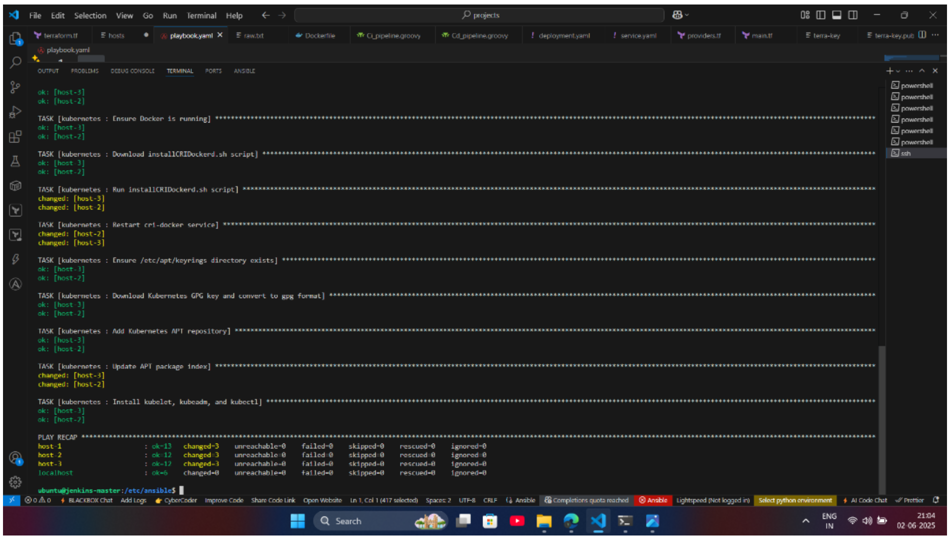
Create the following role directories in your Ansible project:
https://github.com/Rakshitsen/ansible-roles-setup.git

```
roles/
├──── java/
│     └──── tasks/main.yml
├──── jenkins/
│     └──── tasks/main.yml
├──── docker/
│     └──── tasks/main.yml
├──── kubernetes/
│     └──── tasks/main.yml
└──── package-update/
    └──── tasks/main.yml
```

# Now time to run ansible playbook commands

1. ansible-playbook –syntax-check playbook.yaml

2. ansible-playbook playbook.yaml (all tasks written in playbook file start executing)



```
ok: [host-3]
ok: [host-2]

TASK [kubernetes : Ensure Docker is running] ************************************************
ok: [host-3]
ok: [host-2]

TASK [kubernetes : Download installCRIDockerd.sh script] ************************************
ok: [host-3]
ok: [host-2]

TASK [kubernetes : Run installCRIDockerd.sh script] *****************************************
changed: [host-3]
changed: [host-2]

TASK [kubernetes : Restart cri-docker service] **********************************************
changed: [host-2]
changed: [host-3]

TASK [kubernetes : Ensure /etc/apt/keyrings directory exists] *******************************
ok: [host-3]
ok: [host-2]

TASK [kubernetes : Download Kubernetes GPG key and convert to gpg format] *******************
ok: [host-3]
ok: [host-2]

TASK [kubernetes : Add Kubernetes APT repository] *******************************************
ok: [host-3]
ok: [host-2]

TASK [kubernetes : Update APT package index] ************************************************
changed: [host-3]
changed: [host-2]

TASK [kubernetes : Install kubelet, kubeadm, and kubectl] ***********************************
ok: [host-3]
ok: [host-2]

PLAY RECAP **********************************************************************************
host-1                     : ok=13   changed=3    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
host-2                     : ok=12   changed=3    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
host-3                     : ok=12   changed=3    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
localhost                  : ok=6    changed=0    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0

ubuntu@jenkins-master:/etc/ansible$
```

Although k8s install on all three nodes but still we can't consider which one is master nodes and which are worker nodes for this I manually select k8s-ec2-1 as master and run **this commands for master only are as follows:-**

Validate : to check everything is here :-

- docker -v
- cri-dockerd --version
- kubeadm version -o short
- kubelet --version
- kubectl version --client

- sudo kubeadm init --cri-socket unix:///var/run/cri-dockerd.sock --ignore-preflight-errors=all

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

## below installs calico networking driver

- kubectl apply -f https://raw.githubusercontent.com/projectcalico/calico/v3.24.1/manifests/calico.yaml

# Validate: kubectl get nodes

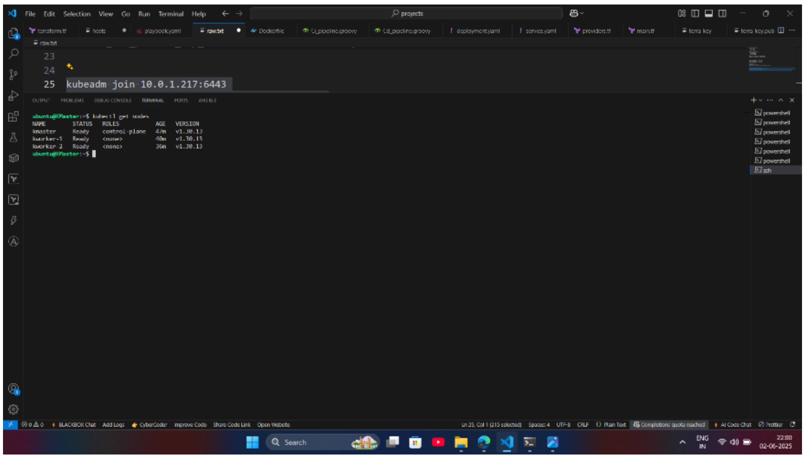
Master is ready now , it was turn of both worker nodes to connect with master.

Commands only for worker nodes:-

- sudo su
- modprobe br_netfilter
- echo 1 > /proc/sys/net/bridge/bridge-nf-call-iptables
- echo 1 > /proc/sys/net/ipv4/ip_forward

```
kubeadm join 10.128.15.231:6443  --token mks3y2.v03tyyru0gy12mbt \
--discovery-token-ca-cert-hash
sha256:3de23d42c7002be0893339fbe558ee75e14399e11f22e3f0b34351077b7c4b56 --cri-socket
unix:///var/run/cri-dockerd.sock
```

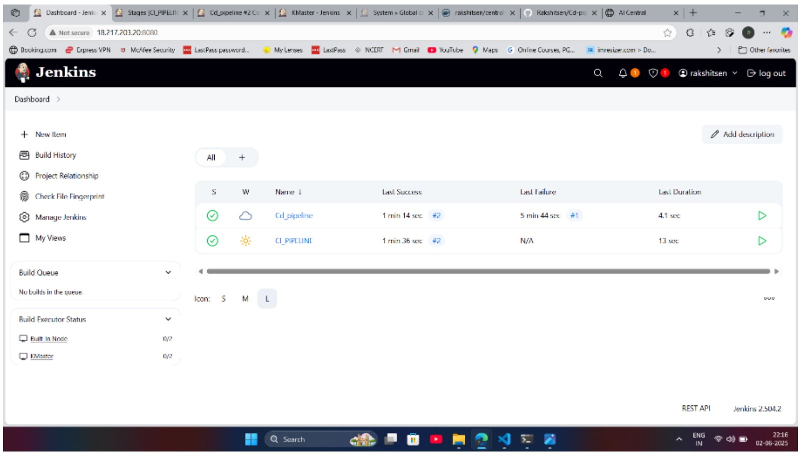
Done workers are conneted now go to master and run kubectl get nodes



```
kubeadm join 10.0.1.217:6443
```

```
ubuntu@KMaster:~$ kubectl get nodes
NAME        STATUS   ROLES           AGE   VERSION
kmaster     Ready    control-plane   47m   v1.30.13
kworker-1   Ready    <none>          40m   v1.30.13
kworker-2   Ready    <none>          36m   v1.30.13
ubuntu@KMaster:~$
```

# CI/CD process
start with jenkins setup:-



18.217.203.20:8080

Build Queue

No builds in the queue.

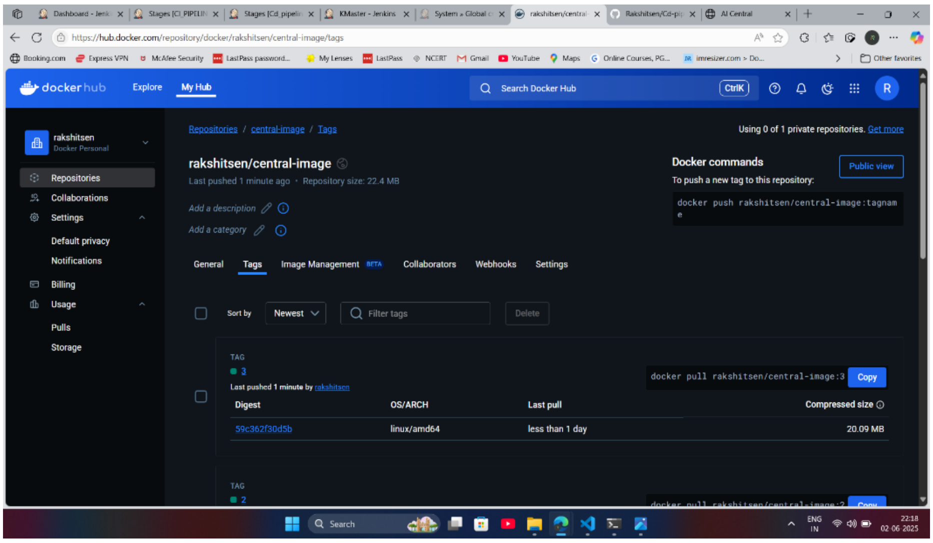
Build Executor Status

Built In Node	0/2
KMaster	0/2

S	W	Name ↓	Last Success	Last Failure	Last Duration
✓	☁	Cd_pipeline	1 min 14 sec #2	5 min 44 sec #1	4.1 sec
✓	☀	CI_PIPELINE	1 min 36 sec #2	N/A	13 sec

1. I add docker credential so that jenkins can push the image to docker hub
2. Create a new job in jenkins and select the Pipeline option name CI_Pipeline
3. CI_Pipeline: Git Repository → Jenkins Trigger → Build Process → Docker Hub
  - Trigger by Git webhook

- stages:
  - Clone repository
  - Build Docker image
  - Push to Docker Hub
  - Deploy test container (port 85)
  - Trigger CD pipeline on success
  - stop the run container (port 85)



rakshitsen/central-image

Last pushed 1 minute ago • Repository size: 22.4 MB

Docker commands

To push a new tag to this repository:

```
docker push rakshitsen/central-image:tagnam
e
```

TAG 3

Last pushed 1 minute by rakshitsen

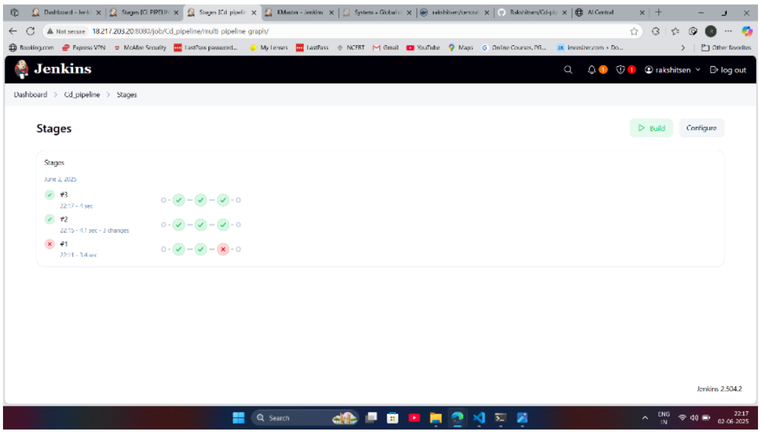
```
docker pull rakshitsen/central-image:3
```

Digest	OS/ARCH	Last pull	Compressed size
59c362f30d5b	linux/amd64	less than 1 day	20.09 MB

TAG 2

4. Create a another job in jenkins and select the Pipeline option name Cd_pipeline.

5. CI Success → Pull Latest Image → Production Deployment


18.217.203.20:8080/job/Cd_pipeline/multi-pipeline-graph/
Jenkins
rakshitsen
log out
Dashboard > Cd_pipeline > Stages
Build
Configure
Jenkins 2.504.2

Stages

Stages

June 2, 2025

- ✓ #3 22:17 - 4 sec
- ✓ #2 22:15 - 4.1 sec - 3 changes
- ✗ #1 22:11 - 3.4 sec

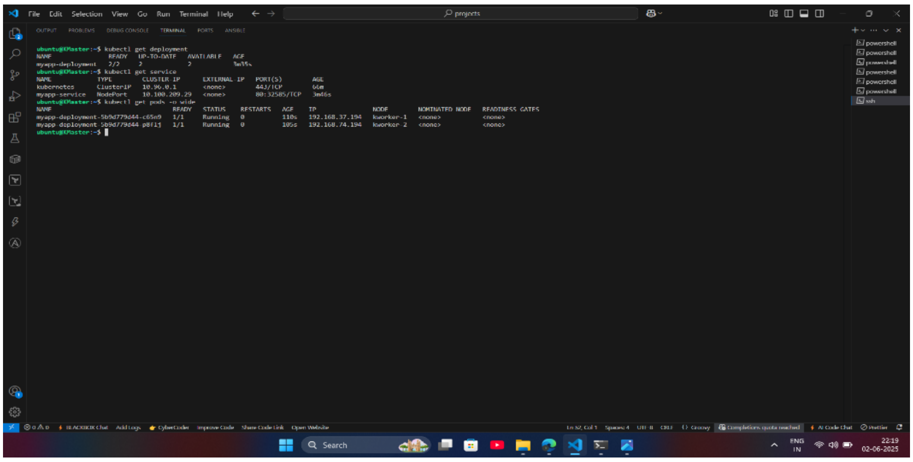
Workflow:

Trigger: Activated by successful CI pipeline

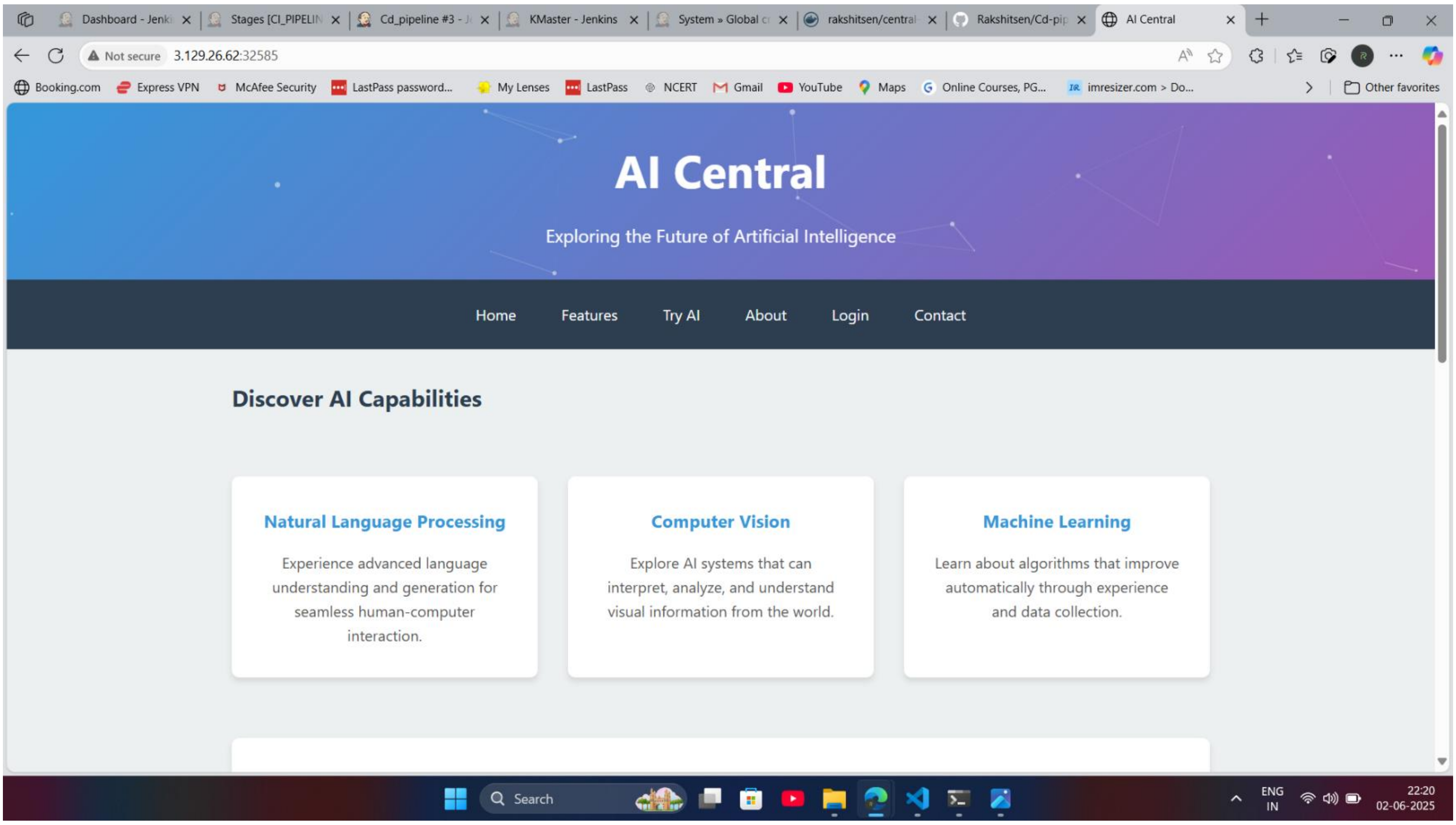
Image Pull: Fetches latest Docker image from Docker Hub

Deployment: Creates Kubernetes deployment

Service Exposure: Creates NodePort service for external access



```
ubuntu@KMaster:~$ kubectl get deployment
NAME               READY   UP-TO-DATE   AVAILABLE   AGE
myapp-deployment   2/2     2            2           3m35s
ubuntu@KMaster:~$ kubectl get service
NAME            TYPE        CLUSTER-IP       EXTERNAL-IP   PORT(S)        AGE
kubernetes      ClusterIP   10.96.0.1        <none>        443/TCP        66m
myapp-service   NodePort    10.100.209.29    <none>        80:32585/TCP   3m46s
ubuntu@KMaster:~$ kubectl get pods -o wide
NAME                                READY   STATUS    RESTARTS   AGE    IP               NODE        NOMINATED NODE   READINESS GATES
myapp-deployment-5b9d779d44-c65n9   1/1     Running   0          110s   192.168.37.194   kworker-1   <none>           <none>
myapp-deployment-5b9d779d44-p8flj   1/1     Running   0          105s   192.168.74.194   kworker-2   <none>           <none>
ubuntu@KMaster:~$
```



AI Central

Exploring the Future of Artificial Intelligence

Home   Features   Try AI   About   Login   Contact

Discover AI Capabilities

Natural Language Processing

Experience advanced language understanding and generation for seamless human-computer interaction.

Computer Vision

Explore AI systems that can interpret, analyze, and understand visual information from the world.

Machine Learning

Learn about algorithms that improve automatically through experience and data collection.