Project-2

Project: End-to-End DevOps Automation for Scalable Containerized Application Deployment

Company: Analytics Pvt Ltd

Role: DevOps Engineer

Description: Designed and implemented a complete DevOps automation pipeline for Analytics Pvt Ltd using Terraform, Ansible, Jenkins, Docker, and Kubernetes to enable scalable, high-availability application deployment. The solution automated infrastructure provisioning, CI/CD workflows, containerization, and Kubernetes-based production deployments.

Application Repo:

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

Problem Statement:

Analytics Pvt Ltd needed a reliable and scalable platform to:

- Automate application deployment and operations
- Handle increasing product demand
- Eliminate manual deployment processes
- Maintain consistent environments without modifying existing Docker containers
- Implement controlled release management with monthly production releases
- Enable seamless scaling of application workloads across multiple servers

The challenge was to implement this solution using modern DevOps practices while ensuring zero disruption to existing Docker images and development workflows.

Architecture Overview

```
Developer

↓

GitHub ( master)

↓ (Webhook Trigger)

Jenkins Pipeline (jenkins-controller)

├── Stage 1: Checkout Code

├── Stage 2: Build (Docker Image)

├── Stage 3: Push Image (Docker Hub)

└── Stage 4: Deploy to Kubernetes

↓

Docker Image (vikky9387/website)

↓

Kubernetes Cluster

├── k8s-master

├── k8s-worker1 → Pod (Replica 1)

└── k8s-worker2 → Pod (Replica 2)

↓

Application Running in Production
```

Server Mapping

Four EC2 instances were provisioned using Terraform:

EC2 Name	Purpose
jenkins-controller	Jenkins + Ansible
k8s-master	Kubernetes Control Plane
k8s-worker1	Kubernetes Worker Node
k8s-worker2	Kubernetes Worker Node

Tools Used

The project utilizes **GitHub** for version control, **Jenkins** for continuous integration and continuous deployment (CI/CD), **Docker** for containerization of the application, **Kubernetes** for container orchestration and management, **Terraform** for infrastructure provisioning, **Ansible** for configuration management and automation, and **AWS** as the cloud platform for hosting the complete infrastructure.

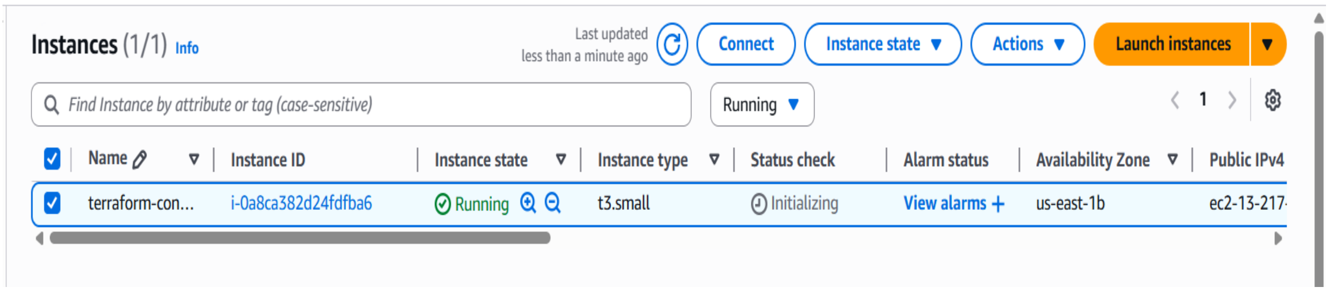
PHASE 1 — Launch Terraform Controller EC2

Create EC2 instance:

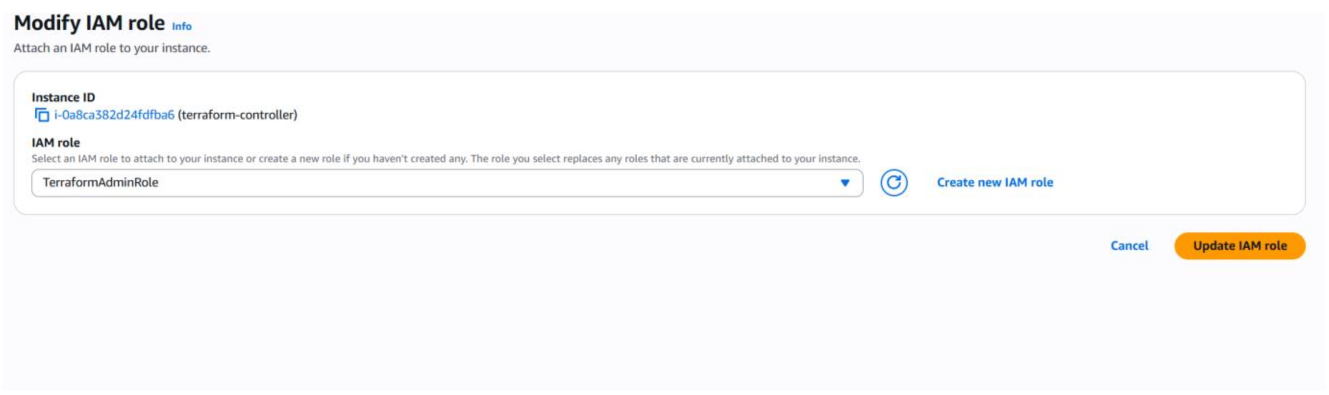
Launch an EC2 instance named Terraform controller and connect it through EC2 instance connect.

Attach IAM role with policies:

- AmazonEC2FullAccess
- AmazonVPCFullAccess
- IAMFullAccess


Instances (1/1) Info
Last updated less than a minute ago
Connect
Instance state ▼
Actions ▼
Launch instances ▼
Find Instance by attribute or tag (case-sensitive)
Running ▼
1

	Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public IPv4
✓	terraform-con...	i-0a8ca382d24fdfba6	Running	t3.small	Initializing	View alarms +	us-east-1b	ec2-13-217


Modify IAM role Info

Attach an IAM role to your instance.

Instance ID
i-0a8ca382d24fdfba6 (terraform-controller)

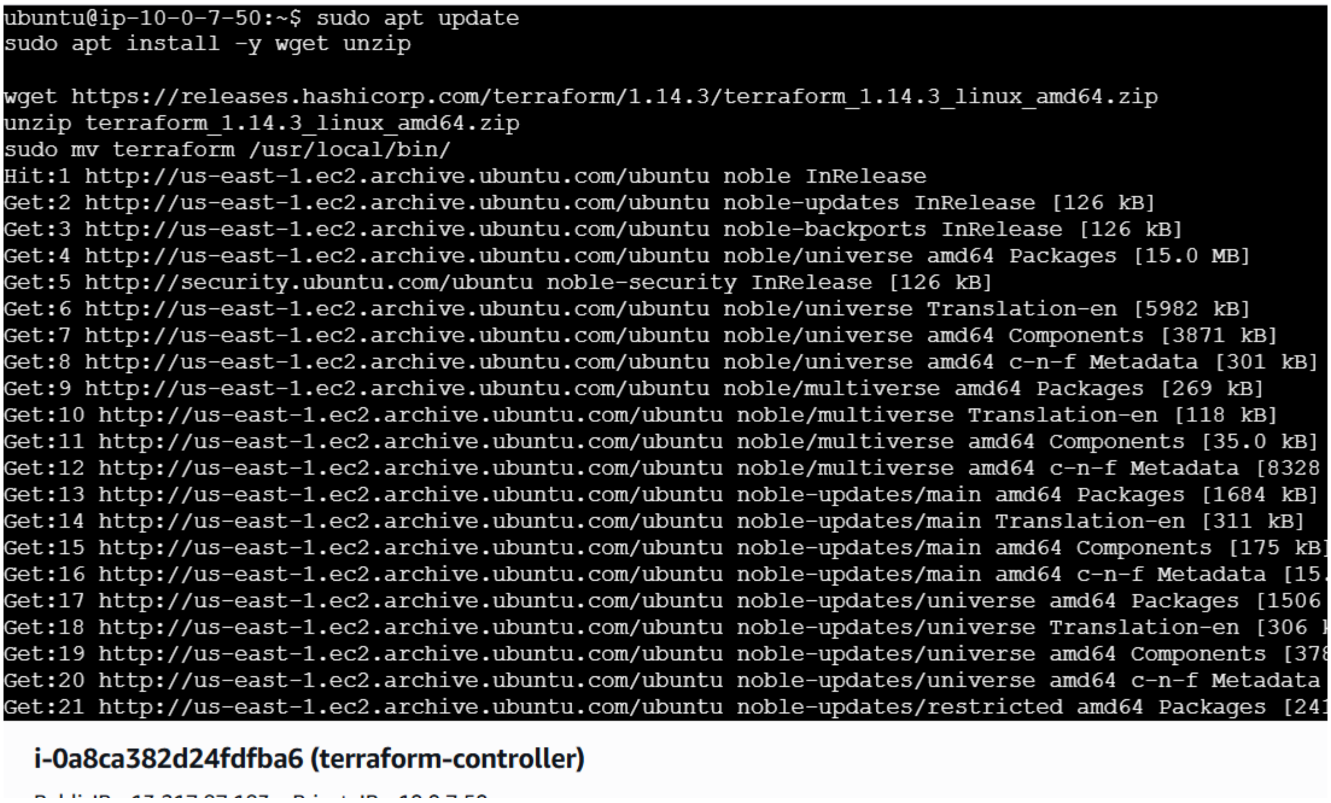
IAM role
Select an IAM role to attach to your instance or create a new role if you haven't created any. The role you select replaces any roles that are currently attached to your instance.

TerraformAdminRole

PHASE 2 — Install Terraform

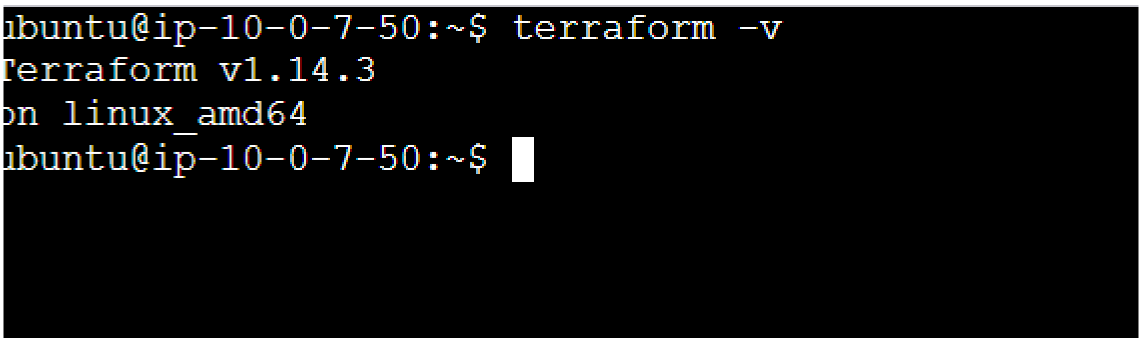
On terraform-controller:

```
sudo apt update
sudo apt install -y wget unzip

wget https://releases.hashicorp.com/terraform/1.14.3/terraform_1.14.3_linux_amd64.zip
unzip terraform_1.14.3_linux_amd64.zip
sudo mv terraform /usr/local/bin/

terraform -v
```



```
ubuntu@ip-10-0-7-50:~$ sudo apt update
sudo apt install -y wget unzip

wget https://releases.hashicorp.com/terraform/1.14.3/terraform_1.14.3_linux_amd64.zip
unzip terraform_1.14.3_linux_amd64.zip
sudo mv terraform /usr/local/bin/
Hit:1 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble InRelease
Get:2 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates InRelease [126 kB]
Get:3 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-backports InRelease [126 kB]
Get:4 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/universe amd64 Packages [15.0 MB]
Get:5 http://security.ubuntu.com/ubuntu noble-security InRelease [126 kB]
Get:6 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/universe Translation-en [5982 kB]
Get:7 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/universe amd64 Components [3871 kB]
Get:8 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/universe amd64 c-n-f Metadata [301 kB]
Get:9 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/multiverse amd64 Packages [269 kB]
Get:10 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/multiverse Translation-en [118 kB]
Get:11 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/multiverse amd64 Components [35.0 kB]
Get:12 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/multiverse amd64 c-n-f Metadata [8328
Get:13 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates/main amd64 Packages [1684 kB]
Get:14 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates/main Translation-en [311 kB]
Get:15 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates/main amd64 Components [175 kB
Get:16 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates/main amd64 c-n-f Metadata [15.
Get:17 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates/universe amd64 Packages [1506
Get:18 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates/universe Translation-en [306
Get:19 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates/universe amd64 Components [37
Get:20 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates/universe amd64 c-n-f Metadata
Get:21 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates/restricted amd64 Packages [24
```

i-0a8ca382d24fdfba6 (terraform-controller)



```
ubuntu@ip-10-0-7-50:~$ terraform -v
Terraform v1.14.3
on linux_amd64
ubuntu@ip-10-0-7-50:~$
```

PHASE 3 — Workspace Setup

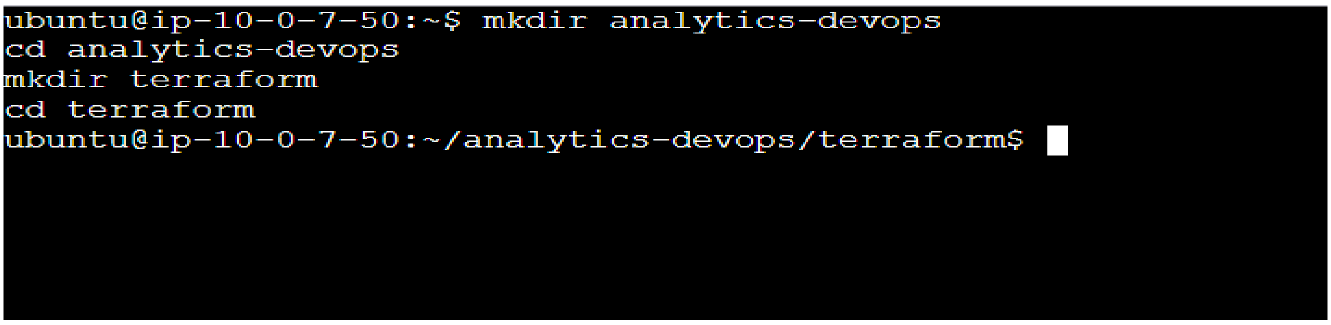
mkdir analytics-devops

cd analytics-devops

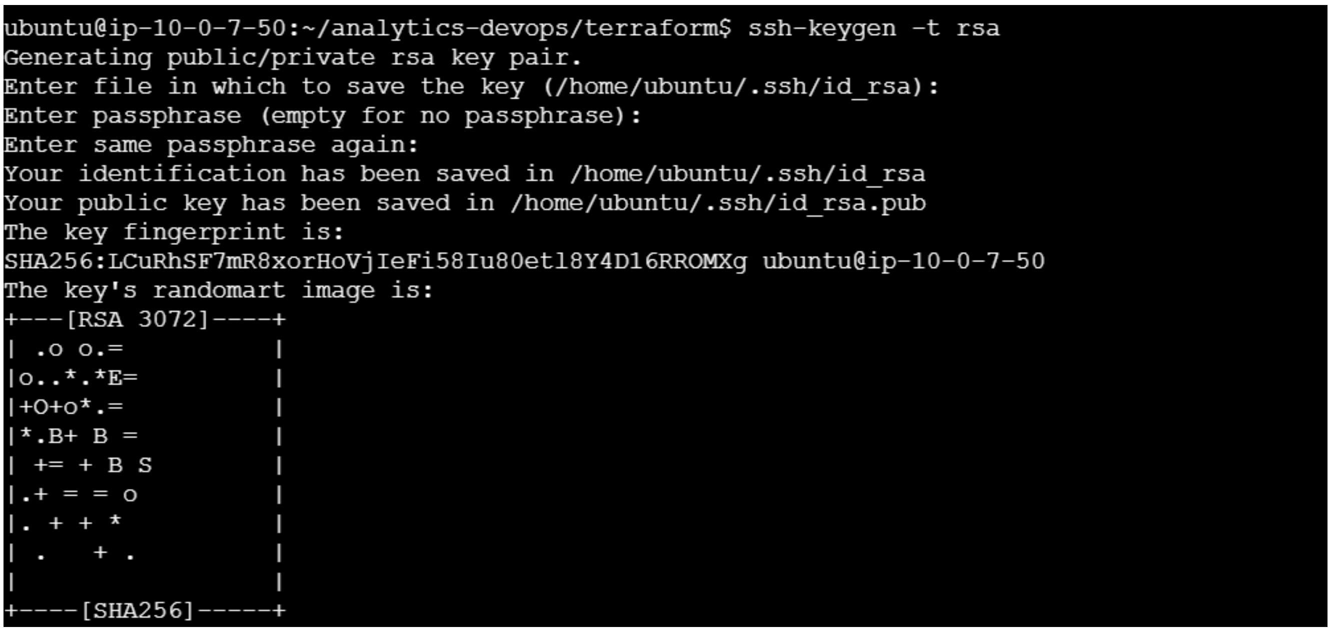
mkdir terraform

cd terraform

ssh-keygen -t rsa



```
ubuntu@ip-10-0-7-50:~$ mkdir analytics-devops
cd analytics-devops
mkdir terraform
cd terraform
ubuntu@ip-10-0-7-50:~/analytics-devops/terraform$
```


```
ubuntu@ip-10-0-7-50:~/analytics-devops/terraform$ ssh-keygen -t rsa
Generating public/private rsa key pair.
Enter file in which to save the key (/home/ubuntu/.ssh/id_rsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/ubuntu/.ssh/id_rsa
Your public key has been saved in /home/ubuntu/.ssh/id_rsa.pub
The key fingerprint is:
SHA256:LCuRhSF7mR8xorHoVjIeFi58Iu80etl8Y4D16RROMXg ubuntu@ip-10-0-7-50
The key's randomart image is:
+---[RSA 3072]----+
| .o o.=          |
|o..*.*E=         |
|+O+o*.=          |
|*.B+ B =         |
| += + B S        |
|.+ = = o         |
|. + + *          |
| .   + .         |
|                 |
+----[SHA256]-----+
```

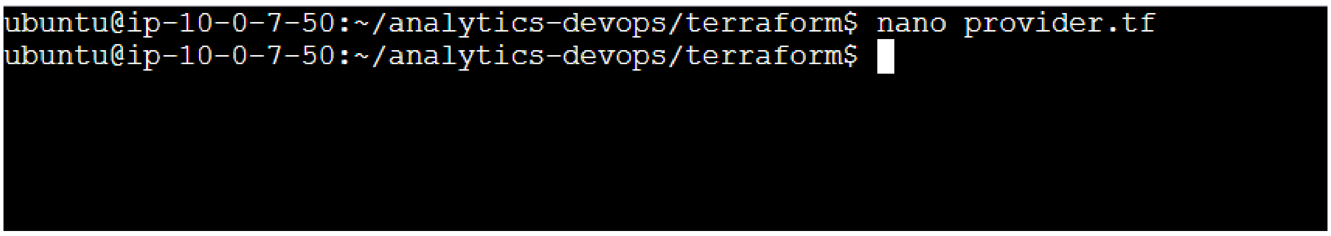
PHASE 4 — Terraform Configuration

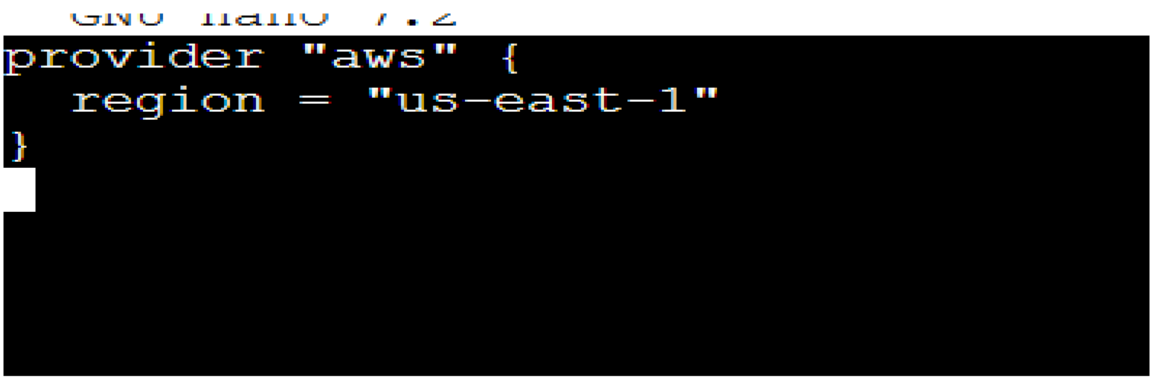
nano provider.tf

paste this:

```
provider "aws" {
  region = "us-east-1"
}
```

Save and exit


```
ubuntu@ip-10-0-7-50:~/analytics-devops/terraform$ nano provider.tf
ubuntu@ip-10-0-7-50:~/analytics-devops/terraform$
```

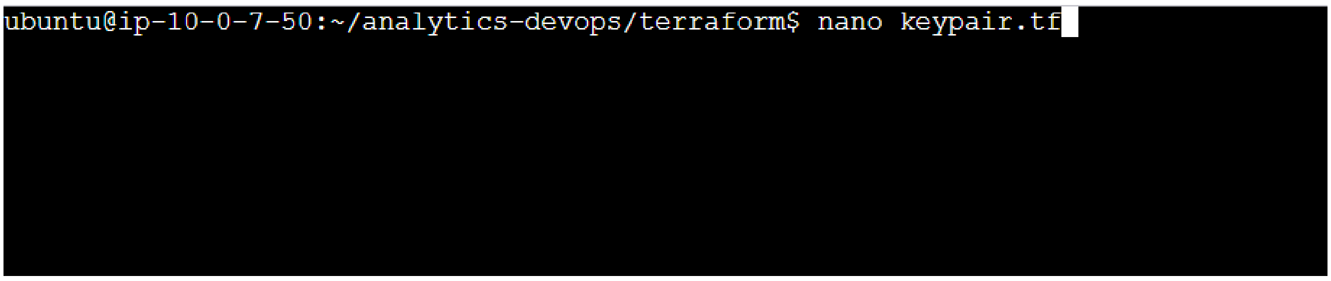

```
   GNU nano 7.2
provider "aws" {
  region = "us-east-1"
}
```

nano keypair.tf

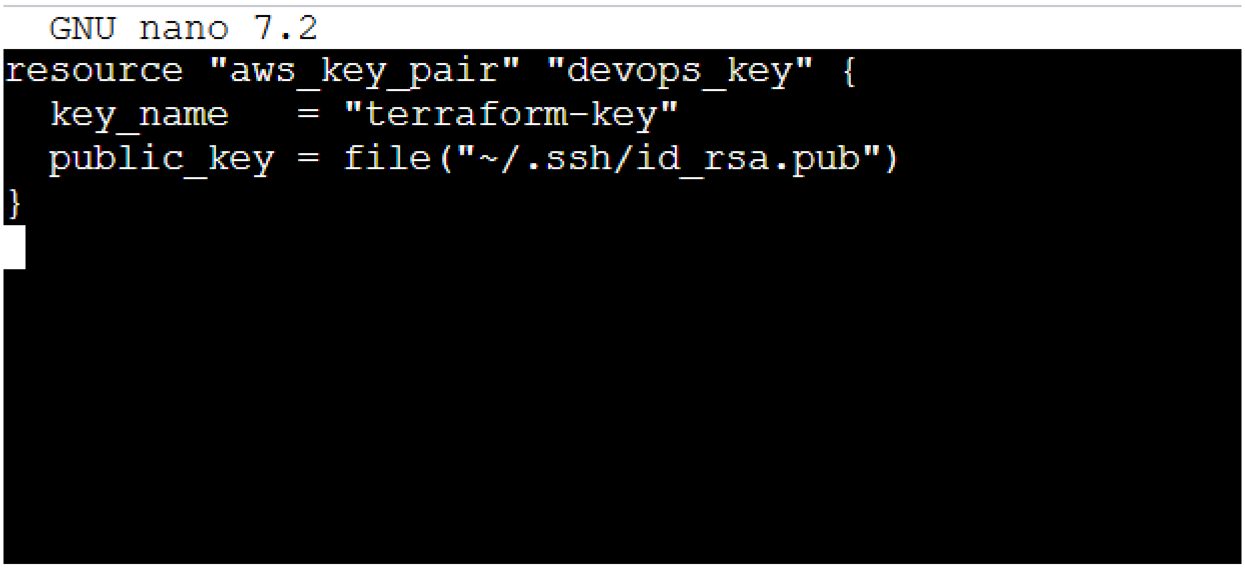
paste this:

```
resource "aws_key_pair" "devops_key" {
  key_name   = "terraform-key"
  public_key = file("~/.ssh/id_rsa.pub")
}
```

Save and exit



```
ubuntu@ip-10-0-7-50:~/analytics-devops/terraform$ nano keypair.tf
```



```
  GNU nano 7.2
resource "aws_key_pair" "devops_key" {
  key_name    = "terraform-key"
  public_key = file("~/.ssh/id_rsa.pub")
}
```

---

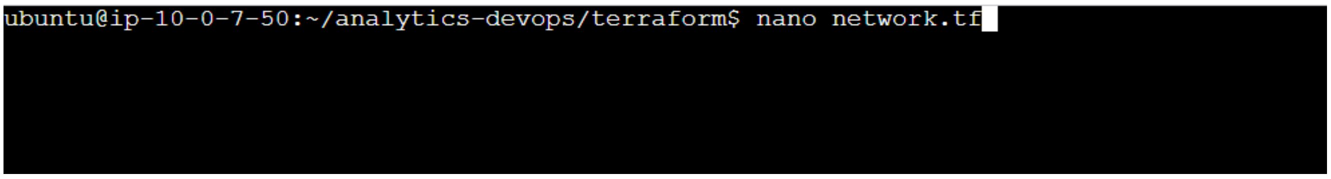
nano network.tf

paste this:

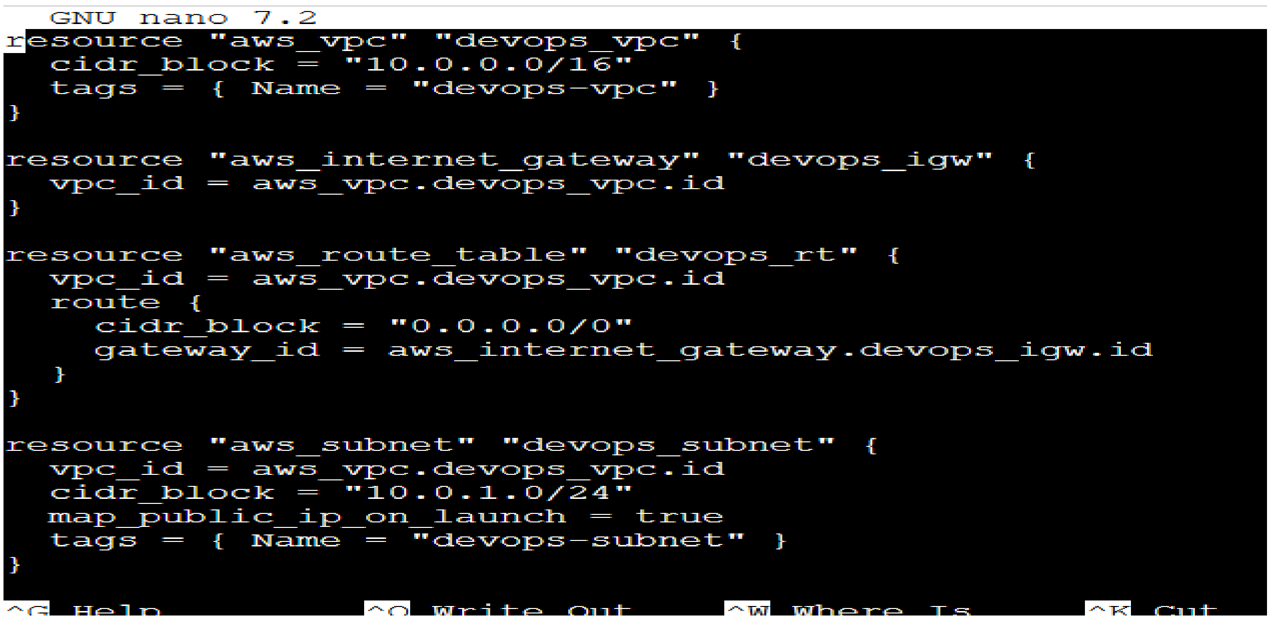
```
resource "aws_vpc" "devops_vpc" {
  cidr_block = "10.0.0.0/16"
  tags = { Name = "devops-vpc" }
}

resource "aws_internet_gateway" "devops_igw" {
  vpc_id = aws_vpc.devops_vpc.id
}
```

```
resource "aws_route_table" "devops_rt" {
  vpc_id = aws_vpc.devops_vpc.id
  route {
    cidr_block = "0.0.0.0/0"
    gateway_id = aws_internet_gateway.devops_igw.id
  }
}

resource "aws_subnet" "devops_subnet" {
  vpc_id = aws_vpc.devops_vpc.id
  cidr_block = "10.0.1.0/24"
  map_public_ip_on_launch = true
  tags = { Name = "devops-subnet" }
}

resource "aws_route_table_association" "devops_assoc" {
  subnet_id = aws_subnet.devops_subnet.id
  route_table_id = aws_route_table.devops_rt.id
}
```

Save and exit



```
ubuntu@ip-10-0-7-50:~/analytics-devops/terraform$ nano network.tf
```



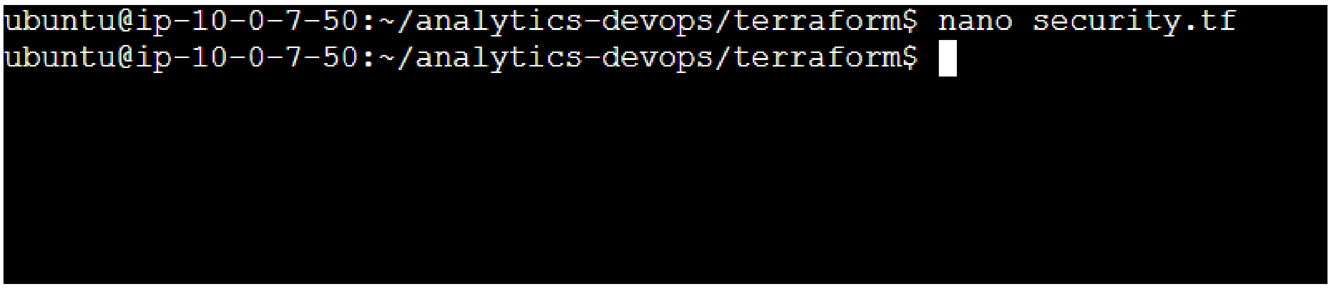
```
  GNU nano 7.2
resource "aws_vpc" "devops_vpc" {
  cidr_block = "10.0.0.0/16"
  tags = { Name = "devops-vpc" }
}

resource "aws_internet_gateway" "devops_igw" {
  vpc_id = aws_vpc.devops_vpc.id
}

resource "aws_route_table" "devops_rt" {
  vpc_id = aws_vpc.devops_vpc.id
  route {
    cidr_block = "0.0.0.0/0"
    gateway_id = aws_internet_gateway.devops_igw.id
  }
}

resource "aws_subnet" "devops_subnet" {
  vpc_id = aws_vpc.devops_vpc.id
  cidr_block = "10.0.1.0/24"
  map_public_ip_on_launch = true
  tags = { Name = "devops-subnet" }
}
^G Help          ^O Write Out     ^W Where Is      ^K Cut
```

nano security.tf

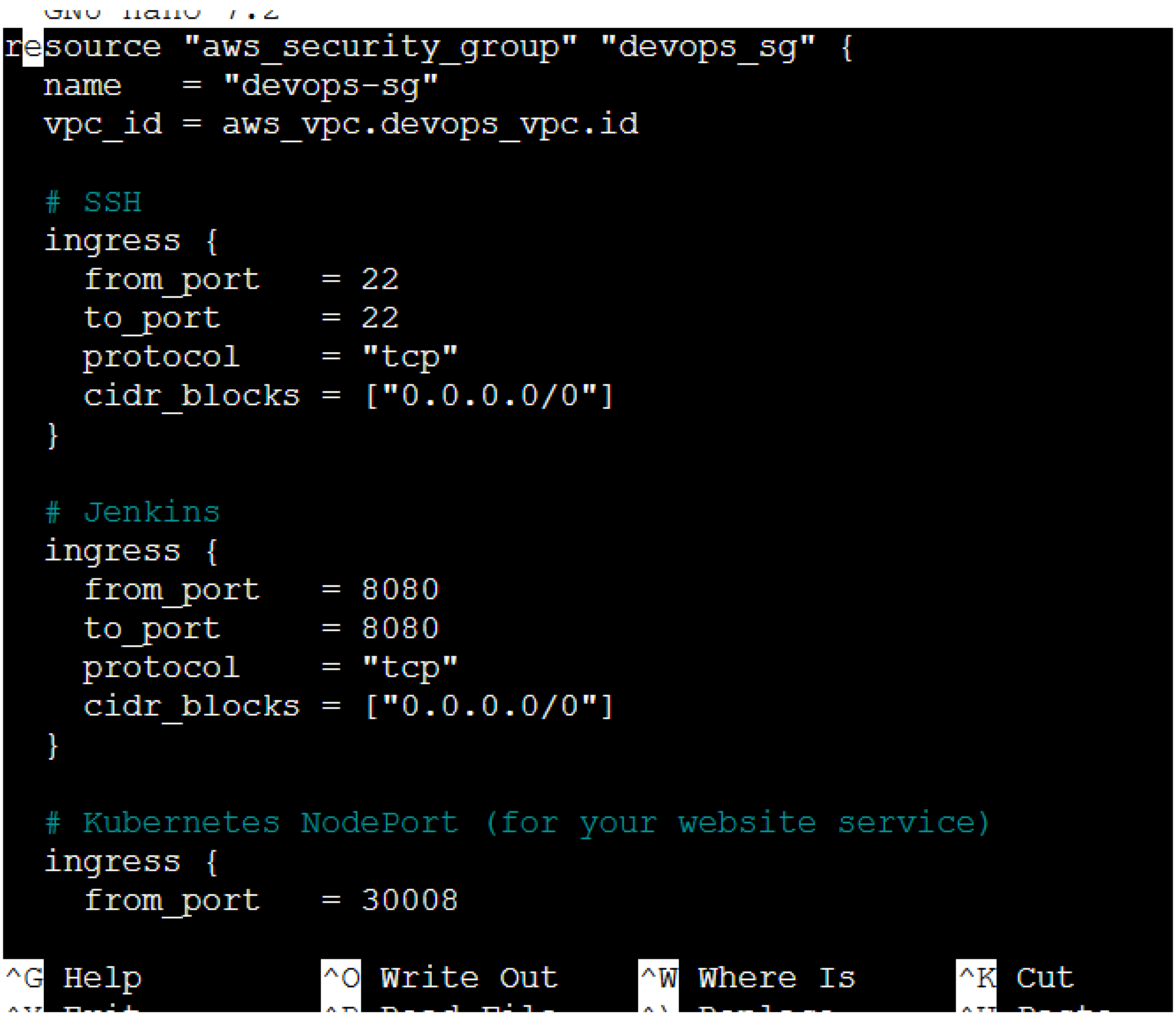
paste this:

```
resource "aws_security_group" "devops_sg" {
  name   = "devops-sg"
  vpc_id = aws_vpc.devops_vpc.id

  # SSH
  ingress {
    from_port   = 22
    to_port     = 22
    protocol    = "tcp"
    cidr_blocks = ["0.0.0.0/0"]
  }

  # Jenkins
  ingress {
    from_port   = 8080
    to_port     = 8080
    protocol    = "tcp"
    cidr_blocks = ["0.0.0.0/0"]
  }

  # Kubernetes NodePort (for your website service)
  ingress {
    from_port   = 30008
    to_port     = 30008
    protocol    = "tcp"
    cidr_blocks = ["0.0.0.0/0"]
  }

  # Kubernetes API
  ingress {
    from_port   = 6443
    to_port     = 6443
    protocol     = "tcp"
    cidr_blocks = ["0.0.0.0/0"]
  }

  # Internal cluster communication
  ingress {
    from_port = 0
    to_port   = 0
    protocol  = "-1"
    self      = true
  }

  egress {
    from_port   = 0
    to_port     = 0
    protocol    = "-1"
    cidr_blocks = ["0.0.0.0/0"]
  }

  tags = {
    Name = "devops-sg"
  }
}
```

Save and exit



```
ubuntu@ip-10-0-7-50:~/analytics-devops/terraform$ nano security.tf
ubuntu@ip-10-0-7-50:~/analytics-devops/terraform$
```



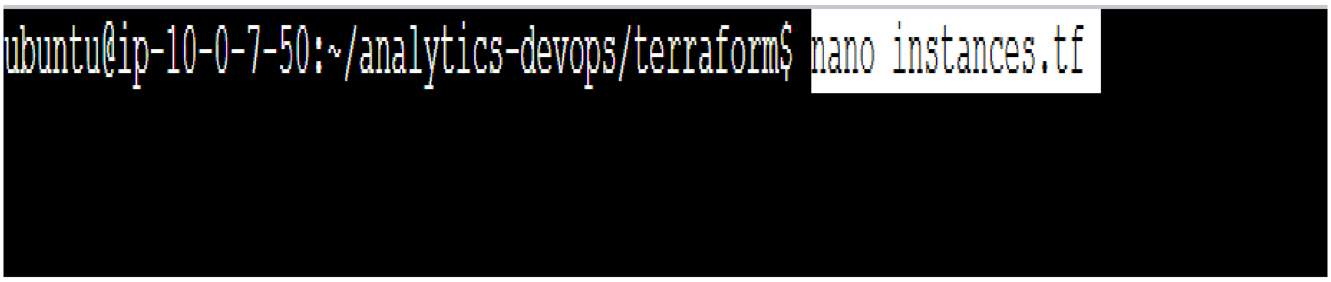
```
resource "aws_security_group" "devops_sg" {
  name   = "devops-sg"
  vpc_id = aws_vpc.devops_vpc.id

  # SSH
  ingress {
    from_port   = 22
    to_port     = 22
    protocol    = "tcp"
    cidr_blocks = ["0.0.0.0/0"]
  }

  # Jenkins
  ingress {
    from_port   = 8080
    to_port     = 8080
    protocol    = "tcp"
    cidr_blocks = ["0.0.0.0/0"]
  }

  # Kubernetes NodePort (for your website service)
  ingress {
    from_port   = 30008
```

nano instances.tf

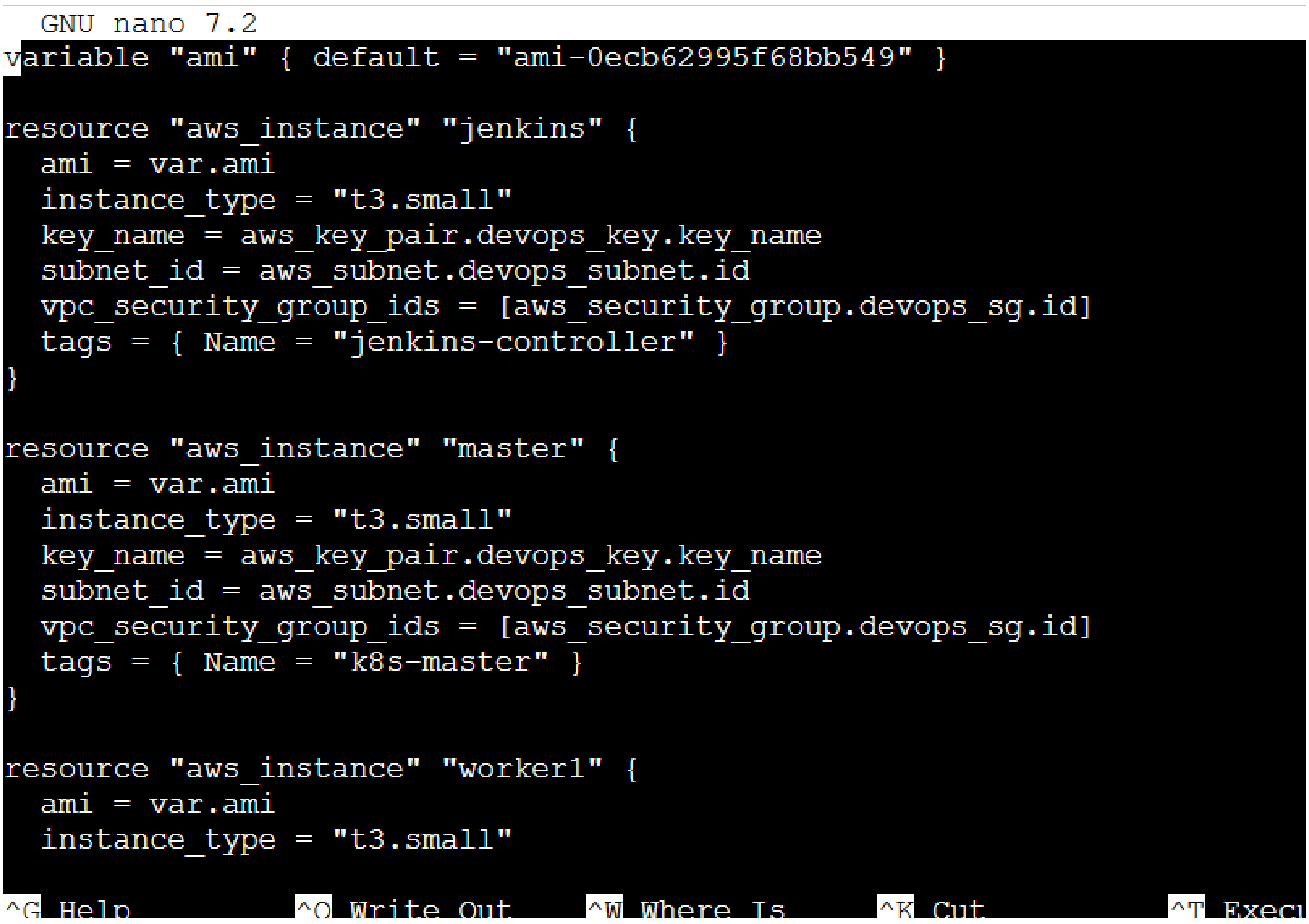
paste this:

```
variable "ami" { default = "ami-0ecb62995f68bb549" }


resource "aws_instance" "jenkins" {
  ami = var.ami
  instance_type = "t3.small"
  key_name = aws_key_pair.devops_key.key_name
  subnet_id = aws_subnet.devops_subnet.id
  vpc_security_group_ids = [aws_security_group.devops_sg.id]
  tags = { Name = "jenkins-controller" }
}


resource "aws_instance" "master" {
  ami = var.ami
  instance_type = "t3.small"
  key_name = aws_key_pair.devops_key.key_name
  subnet_id = aws_subnet.devops_subnet.id
  vpc_security_group_ids = [aws_security_group.devops_sg.id]
  tags = { Name = "k8s-master" }
}

resource "aws_instance" "worker1" {
  ami = var.ami
  instance_type = "t3.small"
  key_name = aws_key_pair.devops_key.key_name
  subnet_id = aws_subnet.devops_subnet.id
  vpc_security_group_ids = [aws_security_group.devops_sg.id]
  tags = { Name = "k8s-worker1" }
}

resource "aws_instance" "worker2" {
  ami = var.ami
  instance_type = "t3.small"
  key_name = aws_key_pair.devops_key.key_name
  subnet_id = aws_subnet.devops_subnet.id
  vpc_security_group_ids = [aws_security_group.devops_sg.id]
  tags = { Name = "k8s-worker2" }
}
```

Save and exit



```
ubuntu@ip-10-0-7-50:~/analytics-devops/terraform$ nano instances.tf
```



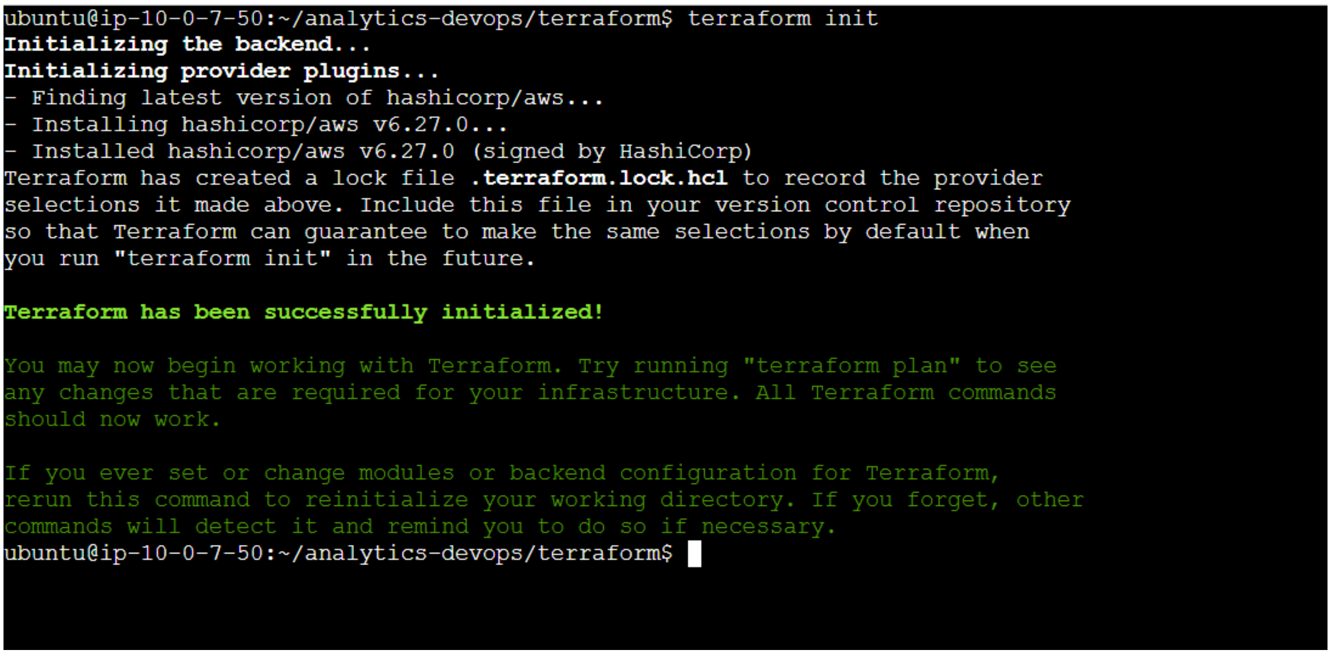
```
  GNU nano 7.2
variable "ami" { default = "ami-0ecb62995f68bb549" }

resource "aws_instance" "jenkins" {
  ami = var.ami
  instance_type = "t3.small"
  key_name = aws_key_pair.devops_key.key_name
  subnet_id = aws_subnet.devops_subnet.id
  vpc_security_group_ids = [aws_security_group.devops_sg.id]
  tags = { Name = "jenkins-controller" }
}

resource "aws_instance" "master" {
  ami = var.ami
  instance_type = "t3.small"
  key_name = aws_key_pair.devops_key.key_name
  subnet_id = aws_subnet.devops_subnet.id
  vpc_security_group_ids = [aws_security_group.devops_sg.id]
  tags = { Name = "k8s-master" }
}

resource "aws_instance" "worker1" {
  ami = var.ami
  instance_type = "t3.small"

^G Help          ^O Write Out      ^W Where Is       ^K Cut            ^T Execu
```

PHASE 5 — Deploy Infrastructure

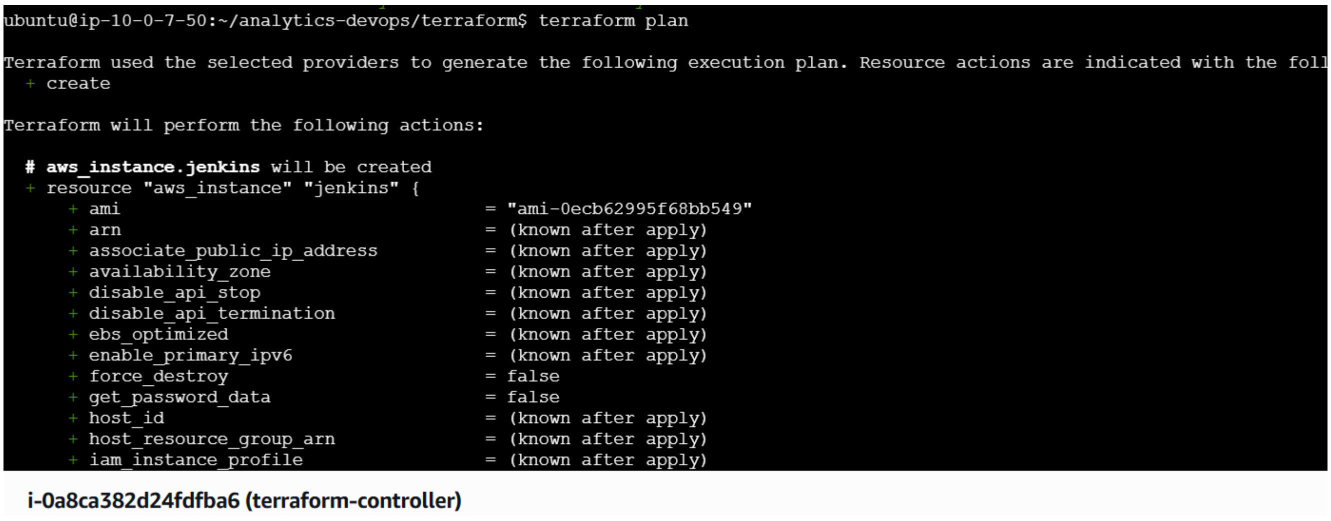
terraform init

terraform plan

terraform apply

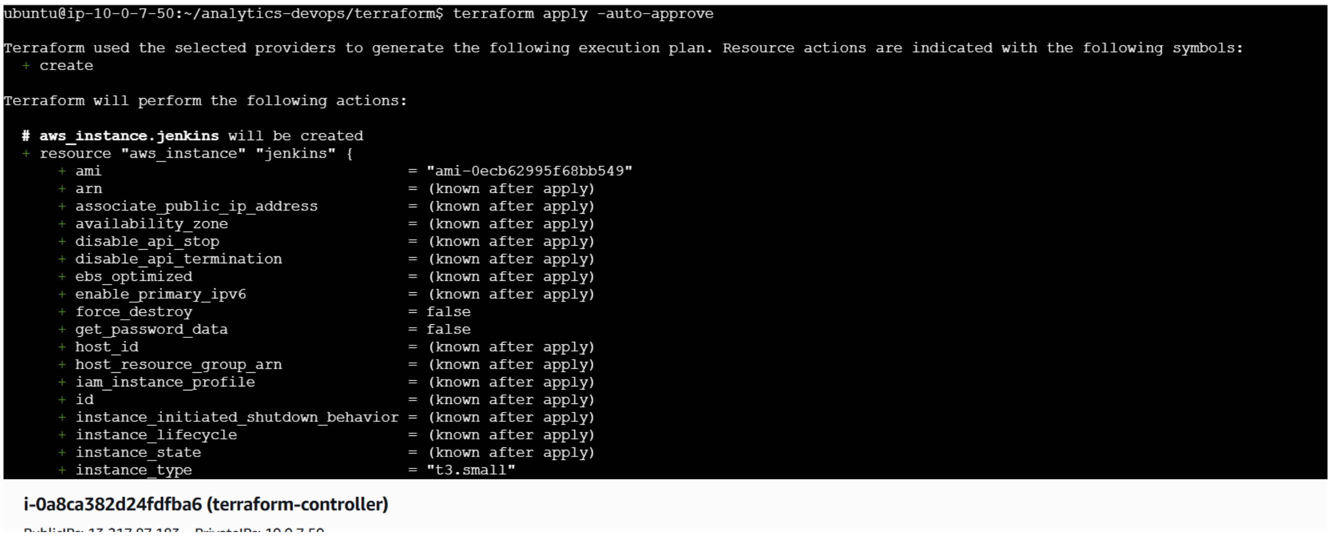


```
ubuntu@ip-10-0-7-50:~/analytics-devops/terraform$ terraform init
Initializing the backend...
Initializing provider plugins...
- Finding latest version of hashicorp/aws...
- Installing hashicorp/aws v6.27.0...
- Installed hashicorp/aws v6.27.0 (signed by HashiCorp)
Terraform has created a lock file .terraform.lock.hcl to record the provider
selections it made above. Include this file in your version control repository
so that Terraform can guarantee to make the same selections by default when
you run "terraform init" in the future.

Terraform has been successfully initialized!

You may now begin working with Terraform. Try running "terraform plan" to see
any changes that are required for your infrastructure. All Terraform commands
should now work.

If you ever set or change modules or backend configuration for Terraform,
rerun this command to reinitialize your working directory. If you forget, other
commands will detect it and remind you to do so if necessary.
ubuntu@ip-10-0-7-50:~/analytics-devops/terraform$
```



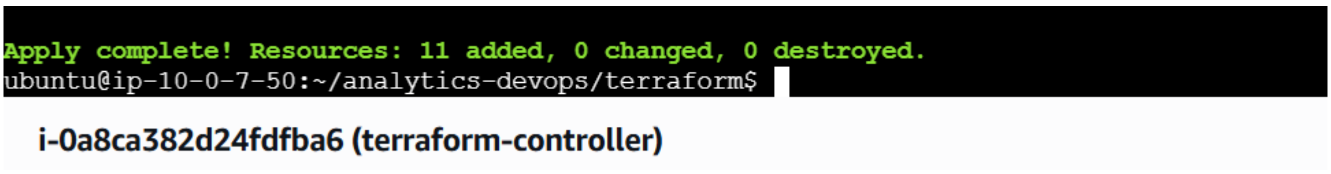
```
ubuntu@ip-10-0-7-50:~/analytics-devops/terraform$ terraform plan

Terraform used the selected providers to generate the following execution plan. Resource actions are indicated with the foll
  + create

Terraform will perform the following actions:

  # aws_instance.jenkins will be created
  + resource "aws_instance" "jenkins" {
      + ami                                  = "ami-0ecb62995f68bb549"
      + arn                                  = (known after apply)
      + associate_public_ip_address          = (known after apply)
      + availability_zone                    = (known after apply)
      + disable_api_stop                     = (known after apply)
      + disable_api_termination              = (known after apply)
      + ebs_optimized                        = (known after apply)
      + enable_primary_ipv6                  = (known after apply)
      + force_destroy                        = false
      + get_password_data                    = false
      + host_id                              = (known after apply)
      + host_resource_group_arn              = (known after apply)
      + iam_instance_profile                 = (known after apply)
```

i-0a8ca382d24fdfba6 (terraform-controller)



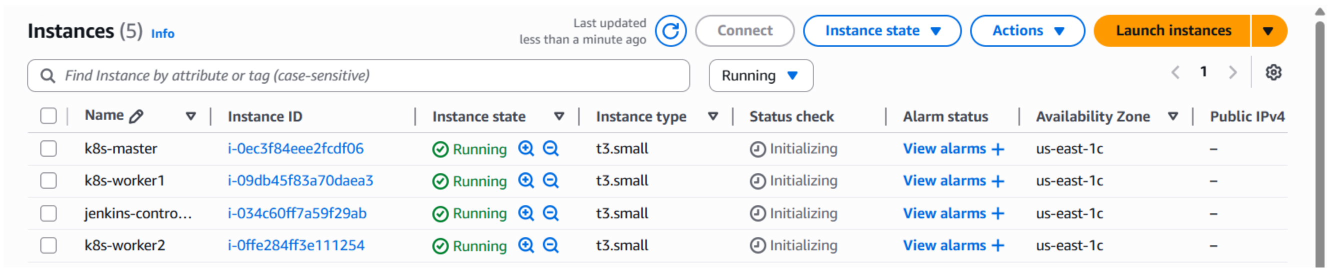
```
ubuntu@ip-10-0-7-50:~/analytics-devops/terraform$ terraform apply -auto-approve

Terraform used the selected providers to generate the following execution plan. Resource actions are indicated with the following symbols:
  + create

Terraform will perform the following actions:

  # aws_instance.jenkins will be created
  + resource "aws_instance" "jenkins" {
      + ami                                  = "ami-0ecb62995f68bb549"
      + arn                                  = (known after apply)
      + associate_public_ip_address          = (known after apply)
      + availability_zone                    = (known after apply)
      + disable_api_stop                     = (known after apply)
      + disable_api_termination              = (known after apply)
      + ebs_optimized                        = (known after apply)
      + enable_primary_ipv6                  = (known after apply)
      + force_destroy                        = false
      + get_password_data                    = false
      + host_id                              = (known after apply)
      + host_resource_group_arn              = (known after apply)
      + iam_instance_profile                 = (known after apply)
      + id                                   = (known after apply)
      + instance_initiated_shutdown_behavior = (known after apply)
      + instance_lifecycle                   = (known after apply)
      + instance_state                       = (known after apply)
      + instance_type                        = "t3.small"
```

i-0a8ca382d24fdfba6 (terraform-controller)



```
Apply complete! Resources: 11 added, 0 changed, 0 destroyed.
ubuntu@ip-10-0-7-50:~/analytics-devops/terraform$
```

i-0a8ca382d24fdfba6 (terraform-controller)



Instances (5) Info
Last updated less than a minute ago
Connect
Instance state
Actions
Launch instances
Find Instance by attribute or tag (case-sensitive)
Running
1

Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public IPv4
k8s-master	i-0ec3f84eee2fcdf06	Running	t3.small	Initializing	View alarms +	us-east-1c	–
k8s-worker1	i-09db45f83a70daea3	Running	t3.small	Initializing	View alarms +	us-east-1c	–
jenkins-contro...	i-034c60ff7a59f29ab	Running	t3.small	Initializing	View alarms +	us-east-1c	–
k8s-worker2	i-0ffe284ff3e111254	Running	t3.small	Initializing	View alarms +	us-east-1c	–

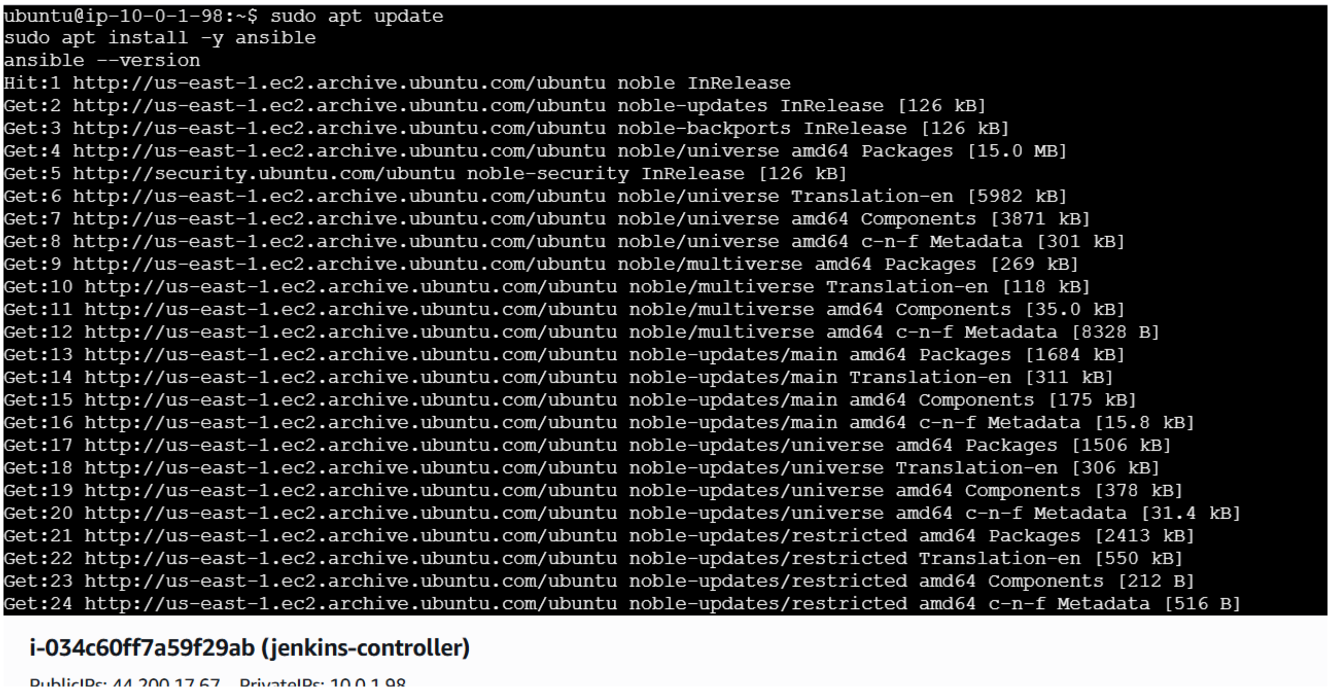
PHASE 6 — Configuration Management with Ansible

Install Ansible on Jenkins Controller

sudo apt update

sudo apt install -y ansible

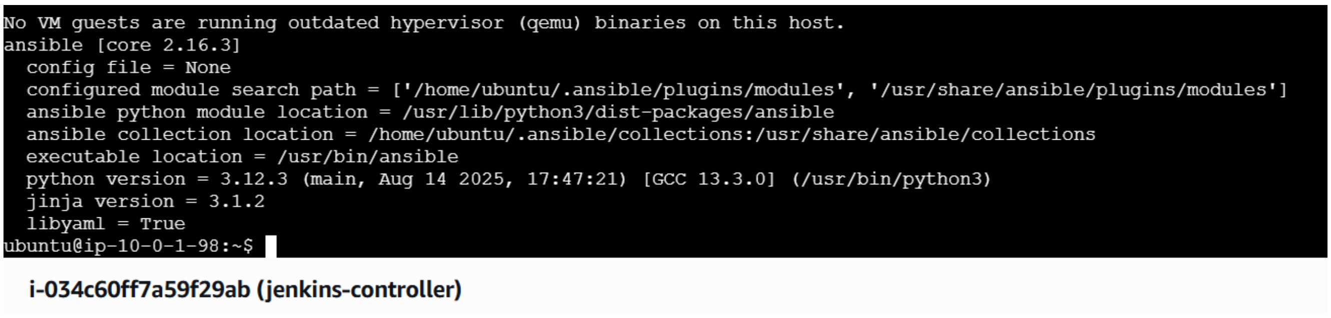
ansible –version



```
ubuntu@ip-10-0-1-98:~$ sudo apt update
sudo apt install -y ansible
ansible --version
Hit:1 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble InRelease
Get:2 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates InRelease [126 kB]
Get:3 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-backports InRelease [126 kB]
Get:4 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/universe amd64 Packages [15.0 MB]
Get:5 http://security.ubuntu.com/ubuntu noble-security InRelease [126 kB]
Get:6 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/universe Translation-en [5982 kB]
Get:7 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/universe amd64 Components [3871 kB]
Get:8 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/universe amd64 c-n-f Metadata [301 kB]
Get:9 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/multiverse amd64 Packages [269 kB]
Get:10 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/multiverse Translation-en [118 kB]
Get:11 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/multiverse amd64 Components [35.0 kB]
Get:12 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/multiverse amd64 c-n-f Metadata [8328 B]
Get:13 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates/main amd64 Packages [1684 kB]
Get:14 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates/main Translation-en [311 kB]
Get:15 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates/main amd64 Components [175 kB]
Get:16 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates/main amd64 c-n-f Metadata [15.8 kB]
Get:17 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates/universe amd64 Packages [1506 kB]
Get:18 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates/universe Translation-en [306 kB]
Get:19 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates/universe amd64 Components [378 kB]
Get:20 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates/universe amd64 c-n-f Metadata [31.4 kB]
Get:21 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates/restricted amd64 Packages [2413 kB]
Get:22 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates/restricted Translation-en [550 kB]
Get:23 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates/restricted amd64 Components [212 B]
Get:24 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates/restricted amd64 c-n-f Metadata [516 B]
```

i-034c60ff7a59f29ab (jenkins-controller)

PublicIPs: 44.200.17.67   PrivateIPs: 10.0.1.98



```
No VM guests are running outdated hypervisor (qemu) binaries on this host.
ansible [core 2.16.3]
  config file = None
  configured module search path = ['/home/ubuntu/.ansible/plugins/modules', '/usr/share/ansible/plugins/modules']
  ansible python module location = /usr/lib/python3/dist-packages/ansible
  ansible collection location = /home/ubuntu/.ansible/collections:/usr/share/ansible/collections
  executable location = /usr/bin/ansible
  python version = 3.12.3 (main, Aug 14 2025, 17:47:21) [GCC 13.3.0] (/usr/bin/python3)
  jinja version = 3.1.2
  libyaml = True
ubuntu@ip-10-0-1-98:~$
```

i-034c60ff7a59f29ab (jenkins-controller)

Create Inventory

mkdir ~/ansible

cd ~/ansible

nano hosts

Paste this:

[jenkins]

44.200.17.67

[k8s_master]

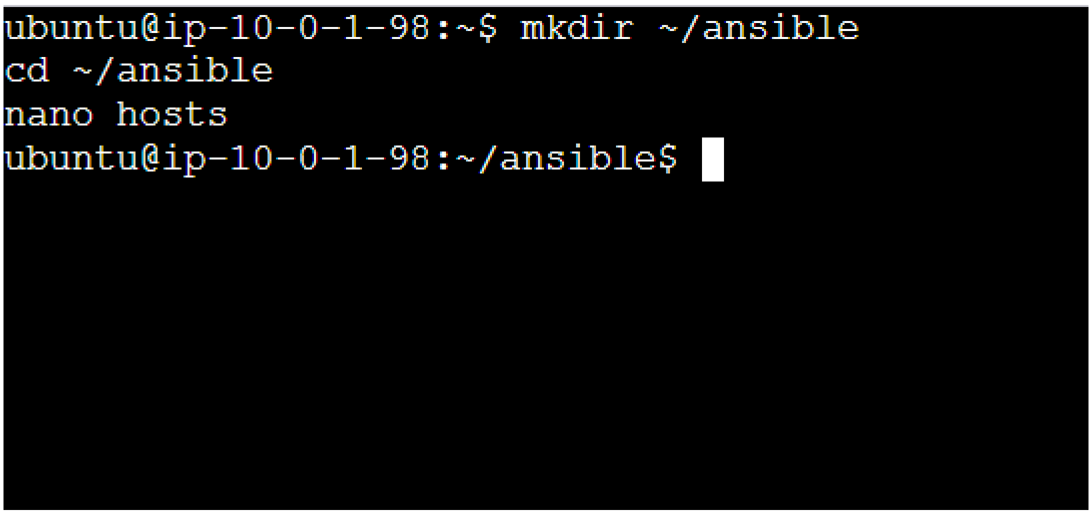
44.204.209.45

[k8s_workers]

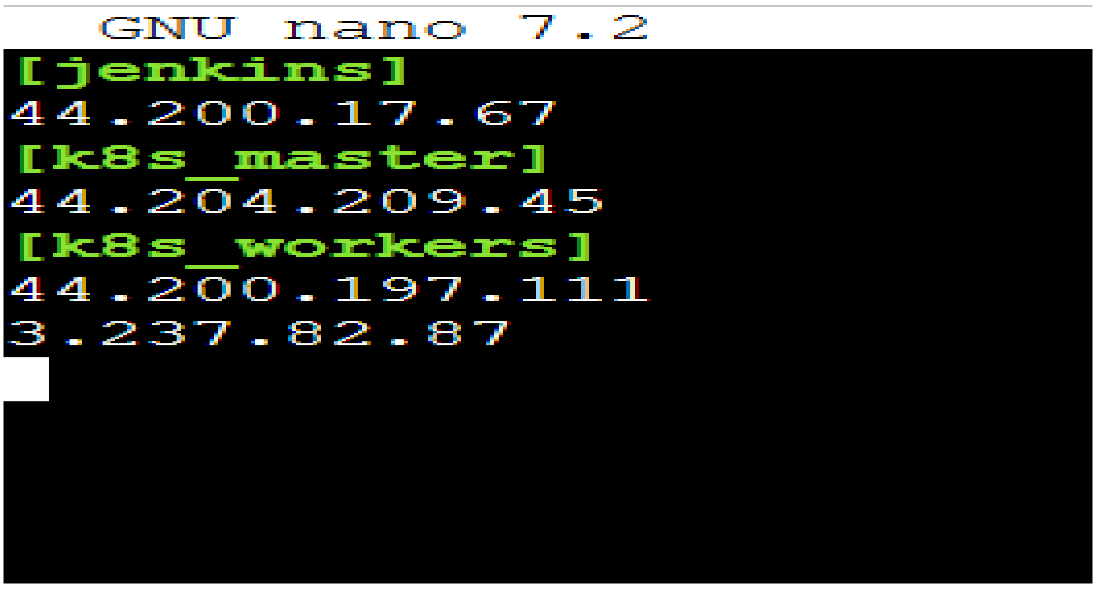
44.200.197.111

3.237.82.87

Save and exit



```
ubuntu@ip-10-0-1-98:~$ mkdir ~/ansible
cd ~/ansible
nano hosts
ubuntu@ip-10-0-1-98:~/ansible$
```



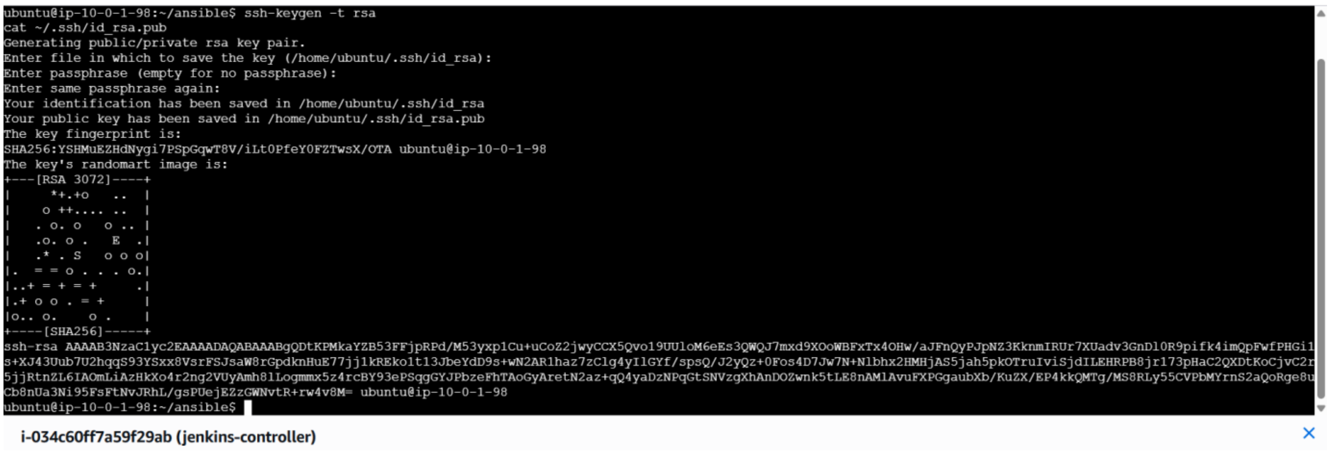
```
GNU nano 7.2
[jenkins]
44.200.17.67
[k8s_master]
44.204.209.45
[k8s_workers]
44.200.197.111
3.237.82.87
```

PHASE 7 - SECURE SSH TRUST SETUP

On Controller

ssh-keygen -t rsa

cat ~/.ssh/id_rsa.pub



```
ubuntu@ip-10-0-1-98:~/ansible$ ssh-keygen -t rsa
cat ~/.ssh/id_rsa.pub
Generating public/private rsa key pair.
Enter file in which to save the key (/home/ubuntu/.ssh/id_rsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/ubuntu/.ssh/id_rsa
Your public key has been saved in /home/ubuntu/.ssh/id_rsa.pub
The key fingerprint is:
SHA256:YSHMuEZHdNygi7PSpGqwT8V/iLt0PfeY0FZTwsX/OTA ubuntu@ip-10-0-1-98
The key's randomart image is:
+---[RSA 3072]----+
|      *+.+o   ..  |
|     o ++.... ..  |
|    . o. o   o .. |
|    .o. o .   E  .|
|    .* . S   o o o|
|.  = = o . . . o.|
|..+ = + = +      .|
|.+ o o . = +      |
|o.. o.     o .    |
+----[SHA256]-----+
ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAABgQDtKPMkaYZB53FFjpRPd/M53yxp1Cu+uCoZ2jwyCCX5Qvo19UUloM6eEs3QWQJ7mxd9XOoWBFxTx4OHw/aJFnQyPJpNZ3KknmIRUr7XUadv3GnDl0R9pifk4imQpFwfPHGil
s+XJ43Uub7U2hqqS93YSxx8VsrFSJsaW8rGpdknHuE77jj1kREko1t13JbeYdD9s+wN2AR1haz7zClg4yIlGYf/spsQ/J2yQz+0Fos4D7Jw7N+Nlbhx2HMHjAS5jah5pkOTruIviSjdILEHRPB8jr173pHaC2QXDtKoCjvC2r
5jjRtnZL6IAOmLiAzHkXo4r2ng2VUyAmh81Logmmx5z4rcBY93ePSqgGYJPbzeFhTAoGyAretN2az+qQ4yaDzNPqGtSNVzgXhAnDOZwnk5tLE8nAMlAvuFXPGgaubXb/KuZX/EP4kkQMTg/MS8RLy55CVPbMYrnS2aQoRge8u
Cb8nUa3Ni95FsFtNvJRhL/gsPUejEZzGWNvtR+rw4v8M= ubuntu@ip-10-0-1-98
ubuntu@ip-10-0-1-98:~/ansible$
```

i-034c60ff7a59f29ab (jenkins-controller)

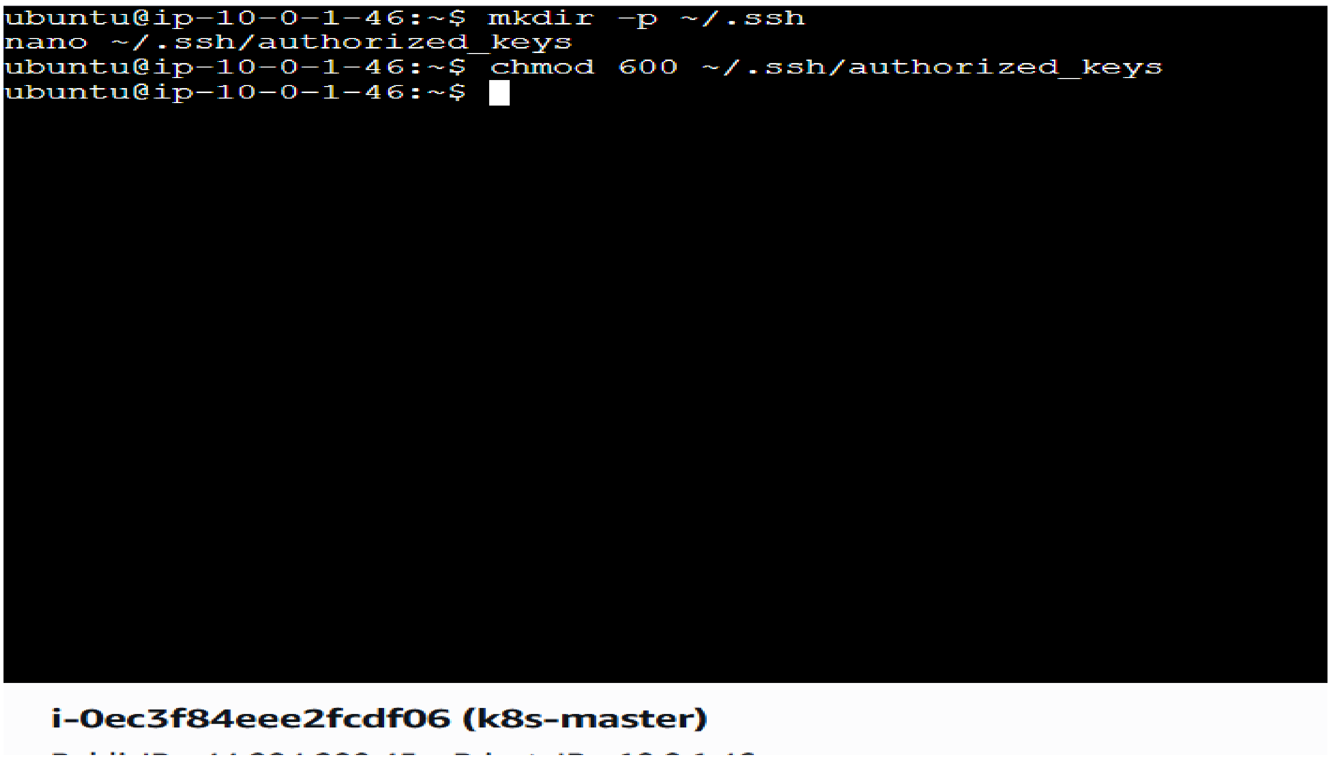
On Each Node (via EC2 Instance Connect)

```
mkdir -p ~/.ssh

nano ~/.ssh/authorized_keys
```

# paste public key

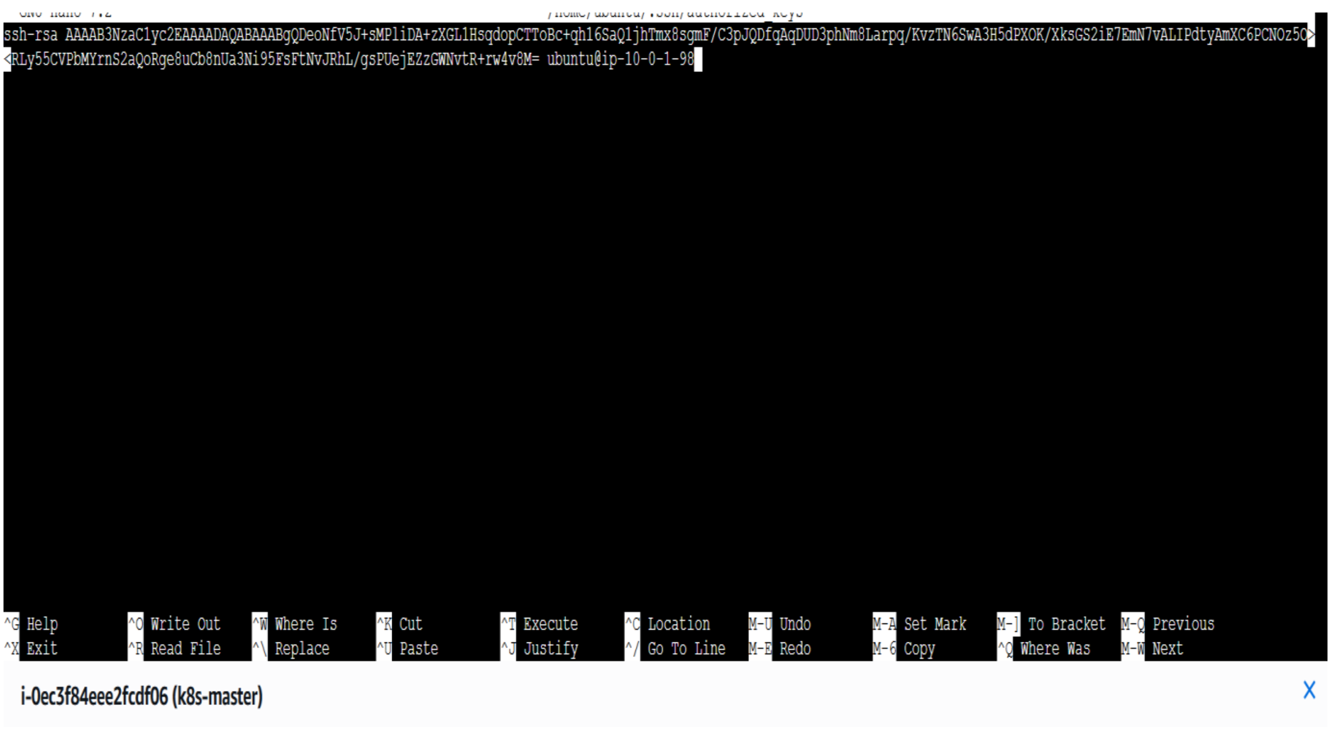
Save and exit

```
chmod 600 ~/.ssh/authorized_keys
```



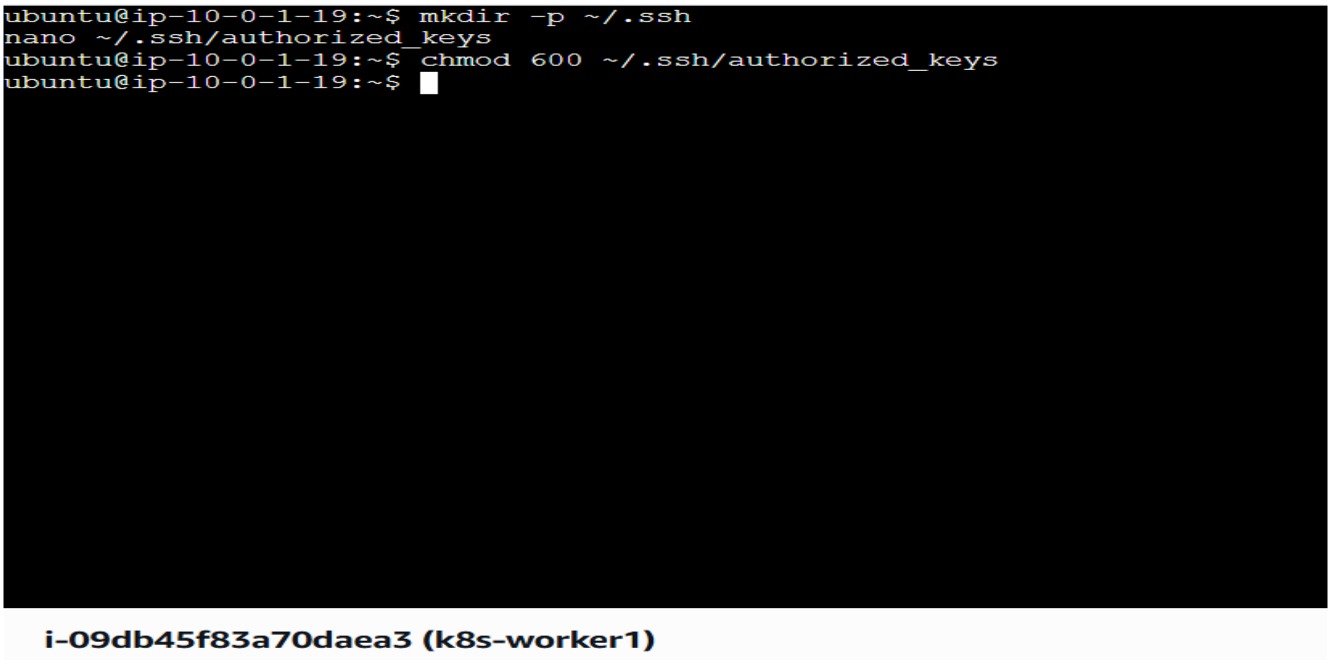
```
ubuntu@ip-10-0-1-46:~$ mkdir -p ~/.ssh
nano ~/.ssh/authorized_keys
ubuntu@ip-10-0-1-46:~$ chmod 600 ~/.ssh/authorized_keys
ubuntu@ip-10-0-1-46:~$ 
```

i-0ec3f84eee2fcdf06 (k8s-master)



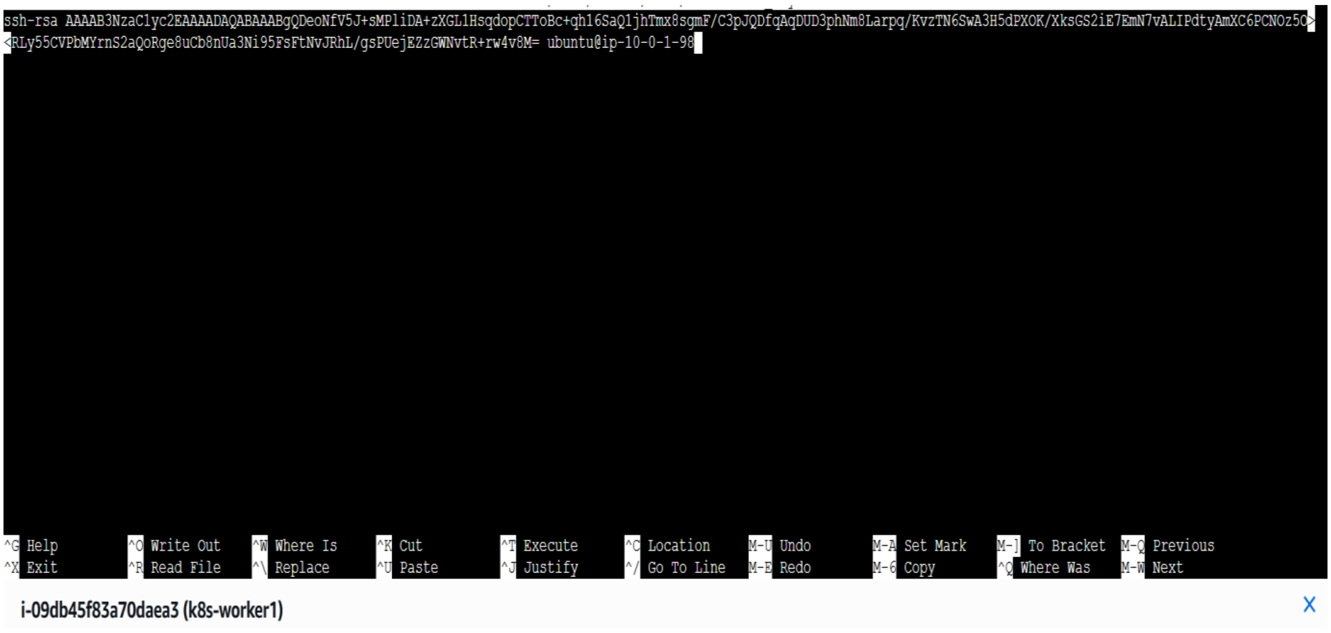
```
  GNU nano 7.2                                   /home/ubuntu/.ssh/authorized_keys
ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAABgQDeoNfV5J+sMPliDA+zXGL1HsqdopCTToBc+qh16SaQ1jhTmx8sgmF/C3pJQDfqAqDUD3phNm8Larpq/KvzTN6SwA3H5dPXOK/XksGS2iE7EmN7vALIPdtyAmXC6PCNOz5O>
<RLy55CVPbMYrnS2aQoRge8uCb8nUa3Ni95FsFtNvJRhL/gsPUejEZzGWNvtR+rw4v8M= ubuntu@ip-10-0-1-98

^G Help       ^O Write Out  ^W Where Is   ^K Cut        ^T Execute    ^C Location   M-U Undo      M-A Set Mark  M-] To Bracket M-Q Previous
^X Exit       ^R Read File  ^\ Replace    ^U Paste      ^J Justify    ^/ Go To Line M-E Redo      M-6 Copy      ^Q Where Was  M-W Next
```

i-0ec3f84eee2fcdf06 (k8s-master)



```
ubuntu@ip-10-0-1-19:~$ mkdir -p ~/.ssh
nano ~/.ssh/authorized_keys
ubuntu@ip-10-0-1-19:~$ chmod 600 ~/.ssh/authorized_keys
ubuntu@ip-10-0-1-19:~$
```

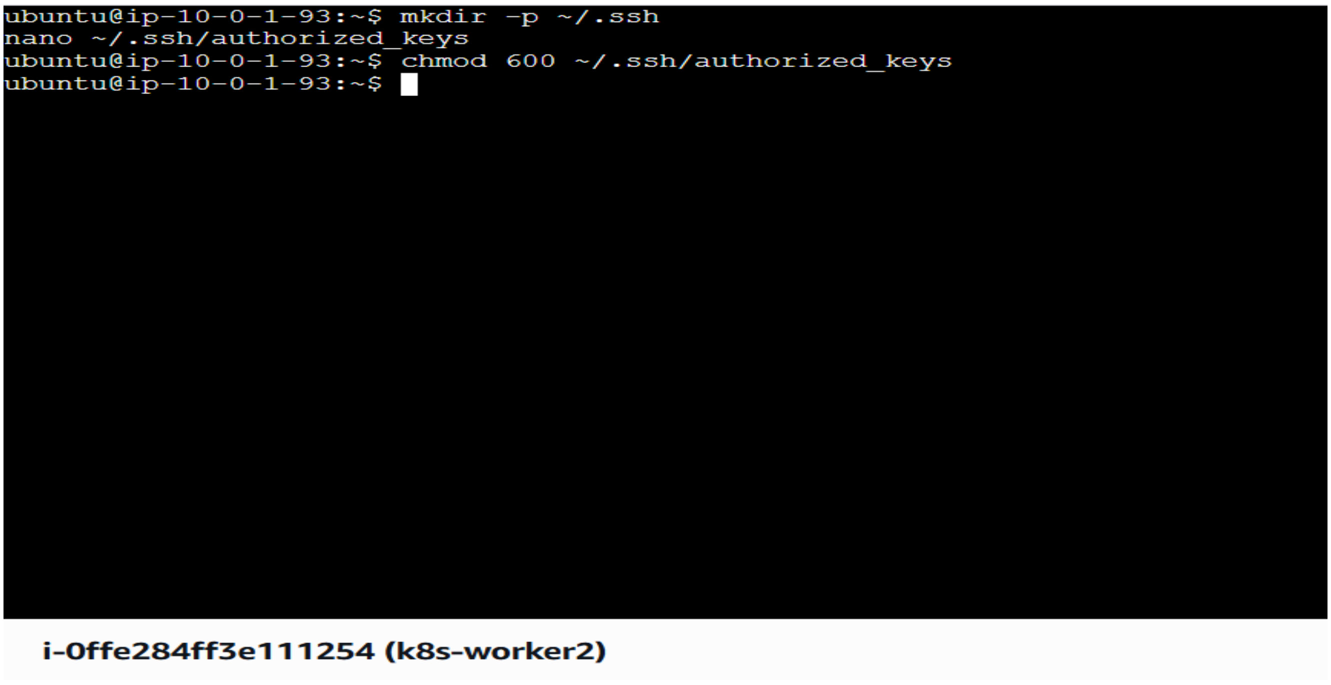
i-09db45f83a70daea3 (k8s-worker1)



```
ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAABgQDeoNfV5J+sMPliDA+zXGL1HsqdopCTToBc+qh16SaQ1jhTmx8sgmF/C3pJQDfqAqDUD3phNm8Larpq/KvzTN6SwA3H5dPXOK/XksGS2iE7EmN7vALIPdtyAmXC6PCNOz5O>
<RLy55CVPbMYrnS2aQoRge8uCb8nUa3Ni95FsFtNvJRhL/gsPUejEZzGWNvtR+rw4v8M= ubuntu@ip-10-0-1-98
```

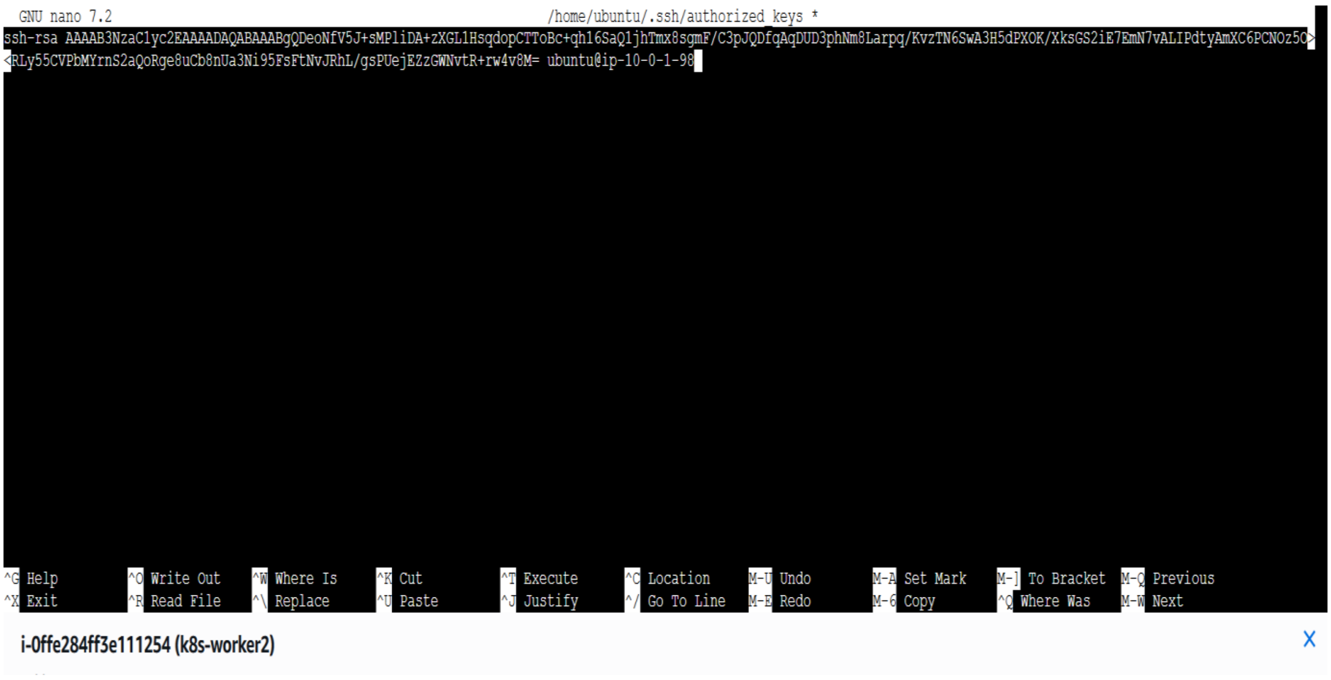
```
^G Help       ^O Write Out  ^W Where Is   ^K Cut        ^T Execute    ^C Location   M-U Undo      M-A Set Mark  M-] To Bracket M-Q Previous
^X Exit       ^R Read File  ^\ Replace    ^U Paste      ^J Justify    ^/ Go To Line M-E Redo      M-6 Copy      ^Q Where Was  M-W Next
```

i-09db45f83a70daea3 (k8s-worker1)



```
ubuntu@ip-10-0-1-93:~$ mkdir -p ~/.ssh
nano ~/.ssh/authorized_keys
ubuntu@ip-10-0-1-93:~$ chmod 600 ~/.ssh/authorized_keys
ubuntu@ip-10-0-1-93:~$
```

i-0ffe284ff3e111254 (k8s-worker2)

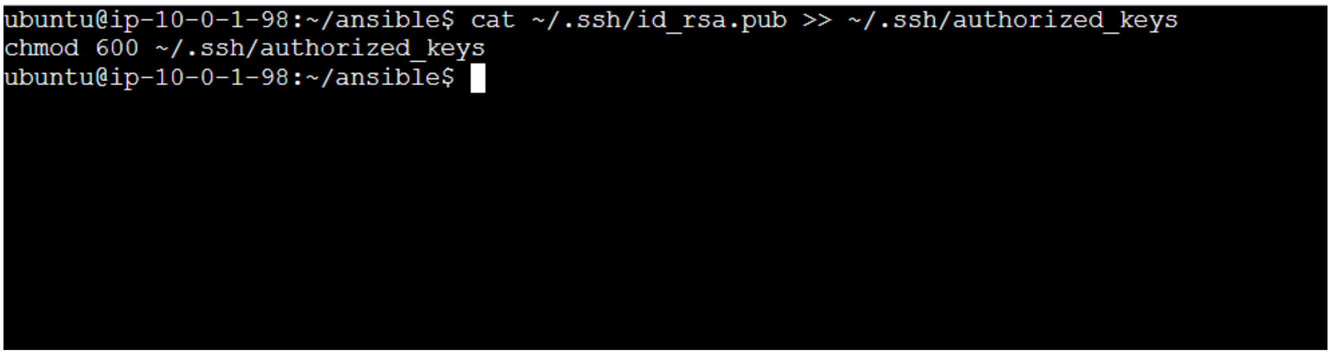


```
  GNU nano 7.2                                        /home/ubuntu/.ssh/authorized_keys *
ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAABgQDeoNfV5J+sMPliDA+zXGL1HsqdopCTToBc+qh16SaQ1jhTmx8sgmF/C3pJQDfqAqDUD3phNm8Larpq/KvzTN6SwA3H5dPXOK/XksGS2iE7EmN7vALIPdtyAmXC6PCNOz5O>
<RLy55CVPbMYrnS2aQoRge8uCb8nUa3Ni95FsFtNvJRhL/gsPUejEZzGWNvtR+rw4v8M= ubuntu@ip-10-0-1-98

^G Help        ^O Write Out   ^W Where Is    ^K Cut         ^T Execute     ^C Location    M-U Undo       M-A Set Mark   M-] To Bracket M-Q Previous
^X Exit        ^R Read File   ^\ Replace     ^U Paste       ^J Justify     ^/ Go To Line  M-E Redo       M-6 Copy       ^Q Where Was   M-W Next
```

Controller Self-Trust

cat ~/.ssh/id_rsa.pub >> ~/.ssh/authorized_keys

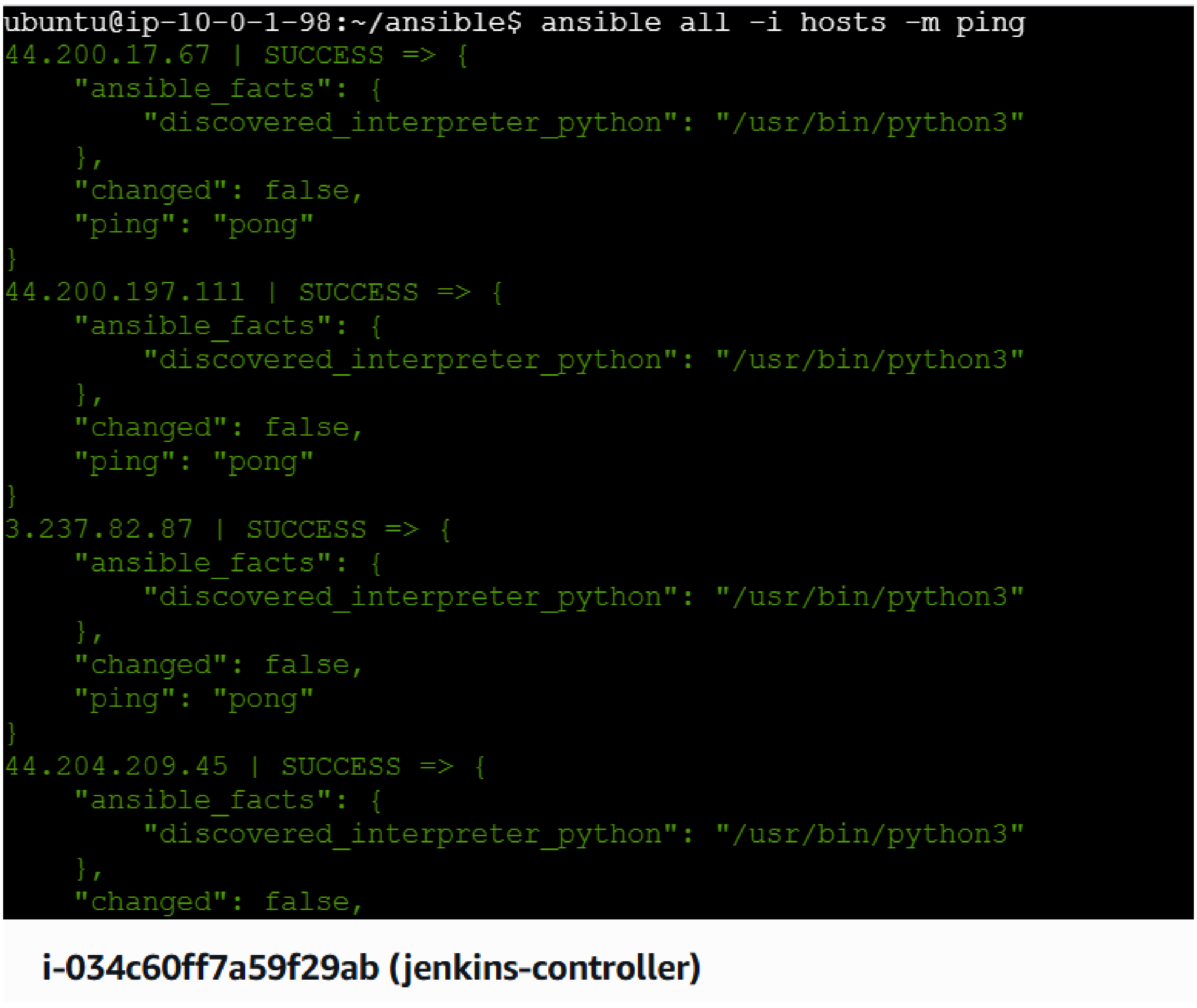
chmod 600 ~/.ssh/authorized_keys



```
ubuntu@ip-10-0-1-98:~/ansible$ cat ~/.ssh/id_rsa.pub >> ~/.ssh/authorized_keys
chmod 600 ~/.ssh/authorized_keys
ubuntu@ip-10-0-1-98:~/ansible$
```

Verify

ansible all -i hosts -m ping



```
ubuntu@ip-10-0-1-98:~/ansible$ ansible all -i hosts -m ping
44.200.17.67 | SUCCESS => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python3"
    },
    "changed": false,
    "ping": "pong"
}
44.200.197.111 | SUCCESS => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python3"
    },
    "changed": false,
    "ping": "pong"
}
3.237.82.87 | SUCCESS => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python3"
    },
    "changed": false,
    "ping": "pong"
}
44.204.209.45 | SUCCESS => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python3"
    },
    "changed": false,
```

i-034c60ff7a59f29ab (jenkins-controller)

PHASE 8 — Automated Installation Using Ansible

Server	Installation
jenkins-controller	Java + Jenkins
k8s-master	Docker + kubeadm + kubelet + kubectl
k8s-worker1	Docker + kubeadm + kubelet
k8s-worker2	Docker + kubeadm + kubelet

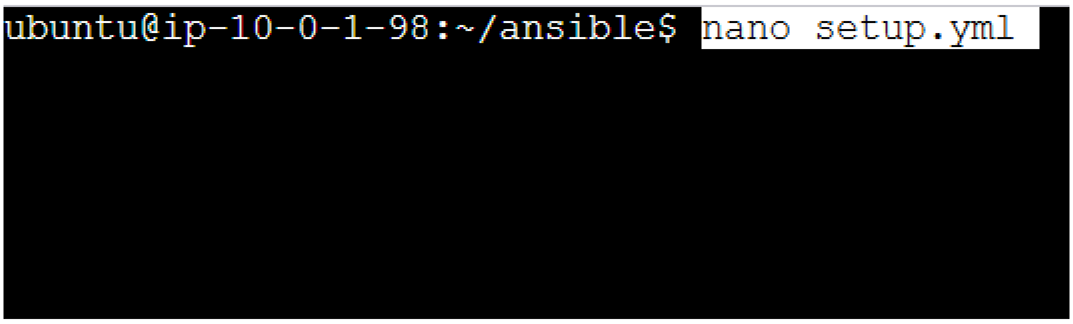
Step 1 — Create Playbook on Jenkins controller

```
cd ~/ansible

nano setup.yml
```

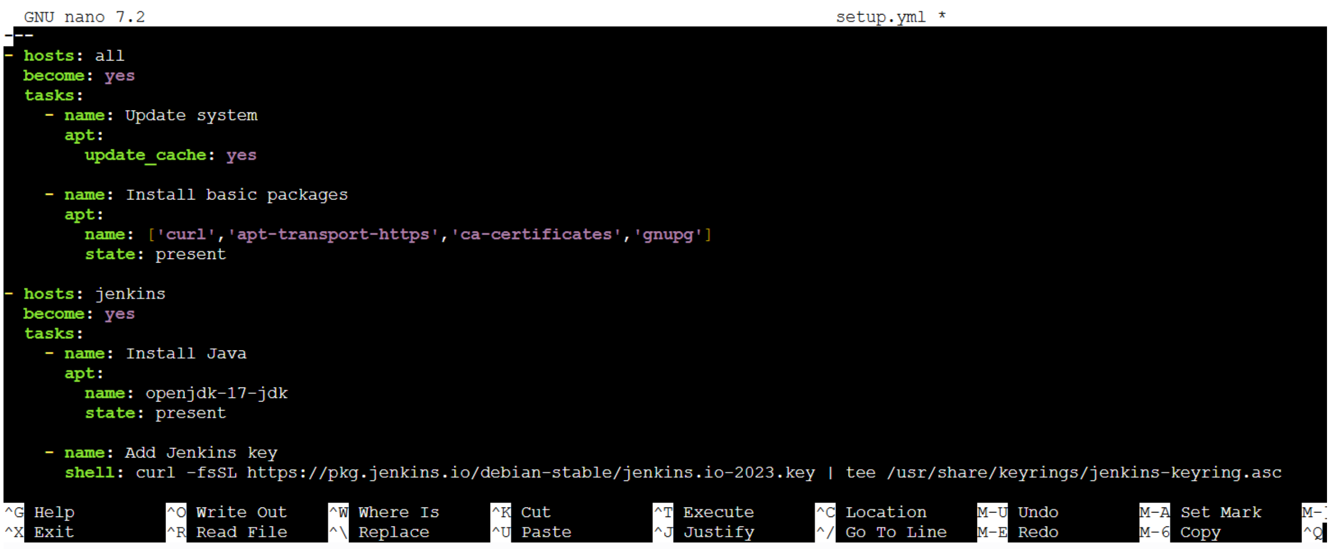
PASTE THIS:

```
---
- hosts: all
  become: yes
  tasks:
    - name: Update system
      apt:
        update_cache: yes

    - name: Install basic packages
      apt:
        name: ['curl','apt-transport-https','ca-certificates','gnupg']
        state: present

- hosts: jenkins
  become: yes
  tasks:
    - name: Install Java
      apt:
        name: openjdk-17-jdk
        state: present

    - name: Add Jenkins key
      shell: curl -fsSL https://pkg.jenkins.io/debian-stable/jenkins.io-2023.key | tee
/usr/share/keyrings/jenkins-keyring.asc

    - name: Add Jenkins repo
      apt_repository:
        repo: deb [signed-by=/usr/share/keyrings/jenkins-keyring.asc] https://pkg.jenkins.io/debian-
stable binary/
        state: present
    - name: Install Jenkins
      apt:
        name: jenkins
        state: present

    - name: Start Jenkins
      service:
        name: jenkins
        state: started
        enabled: yes

- hosts: k8s_master,k8s_workers
  become: yes
  tasks:
    - name: Install Docker
      apt:
        name: docker.io
        state: present

    - name: Enable Docker
      service:
        name: docker
        state: started
        enabled: yes

    - name: Add Kubernetes GPG key
      shell: |
        curl -fsSL https://pkgs.k8s.io/core:/stable:/v1.29/deb/Release.key \
        | gpg --dearmor -o /usr/share/keyrings/kubernetes-archive-keyring.gpg

    - name: Add Kubernetes repository
      shell: |
        echo "deb [signed-by=/usr/share/keyrings/kubernetes-archive-keyring.gpg] \
        https://pkgs.k8s.io/core:/stable:/v1.29/deb/ /" \
        > /etc/apt/sources.list.d/kubernetes.list

    - name: Install Kubernetes packages
      apt:
        name: ['kubelet','kubeadm','kubectl']
        state: present
        update_cache: yes

    - name: Hold Kubernetes packages
      shell: apt-mark hold kubelet kubeadm kubectl
```

Save and exit



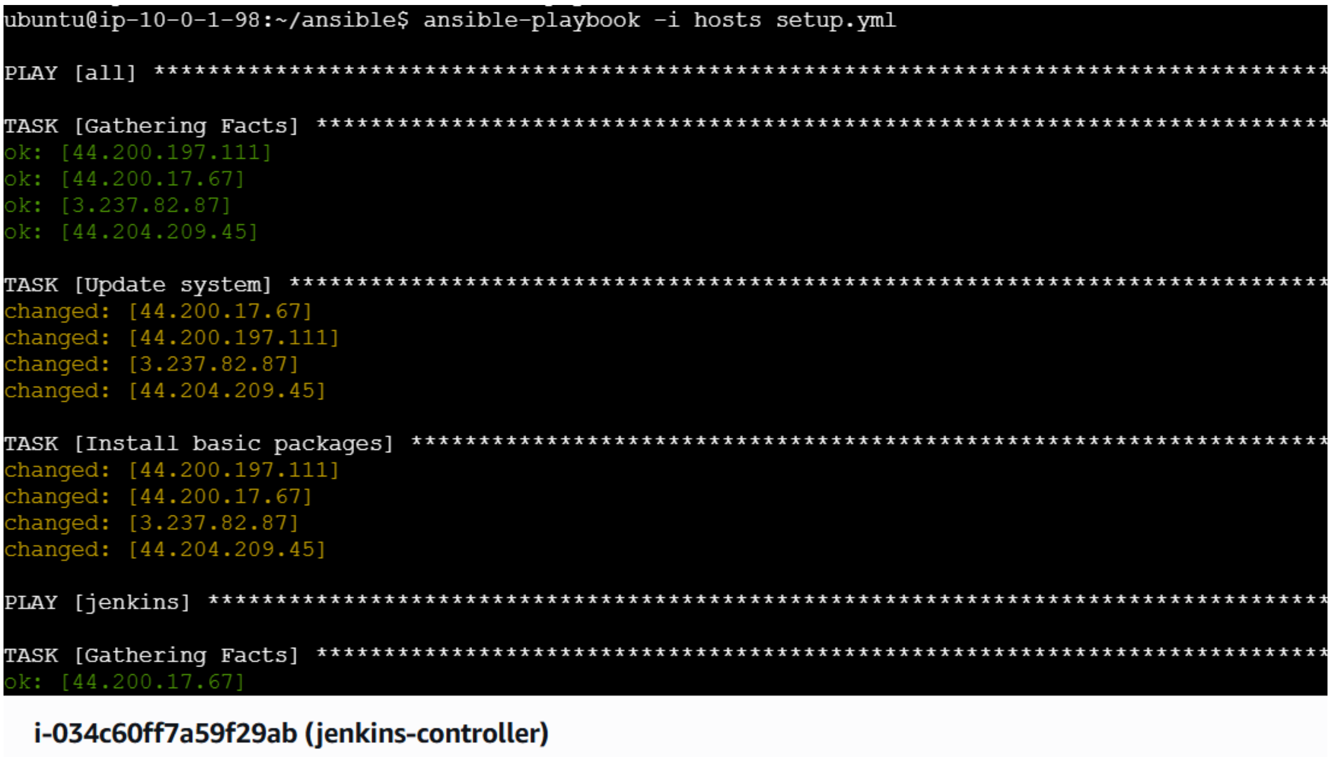
```
ubuntu@ip-10-0-1-98:~/ansible$ nano setup.yml
```



```
  GNU nano 7.2                                                        setup.yml *
---
- hosts: all
  become: yes
  tasks:
    - name: Update system
      apt:
        update_cache: yes

    - name: Install basic packages
      apt:
        name: ['curl','apt-transport-https','ca-certificates','gnupg']
        state: present

- hosts: jenkins
  become: yes
  tasks:
    - name: Install Java
      apt:
        name: openjdk-17-jdk
        state: present

    - name: Add Jenkins key
      shell: curl -fsSL https://pkg.jenkins.io/debian-stable/jenkins.io-2023.key | tee /usr/share/keyrings/jenkins-keyring.asc

^G Help       ^O Write Out  ^W Where Is   ^K Cut        ^T Execute    ^C Location   M-U Undo      M-A Set Mark  M-]
^X Exit       ^R Read File  ^\ Replace    ^U Paste      ^J Justify    ^/ Go To Line M-E Redo      M-6 Copy      ^Q
```

Step 2 — Run Automation

```
ansible-playbook -i hosts setup.yml
```



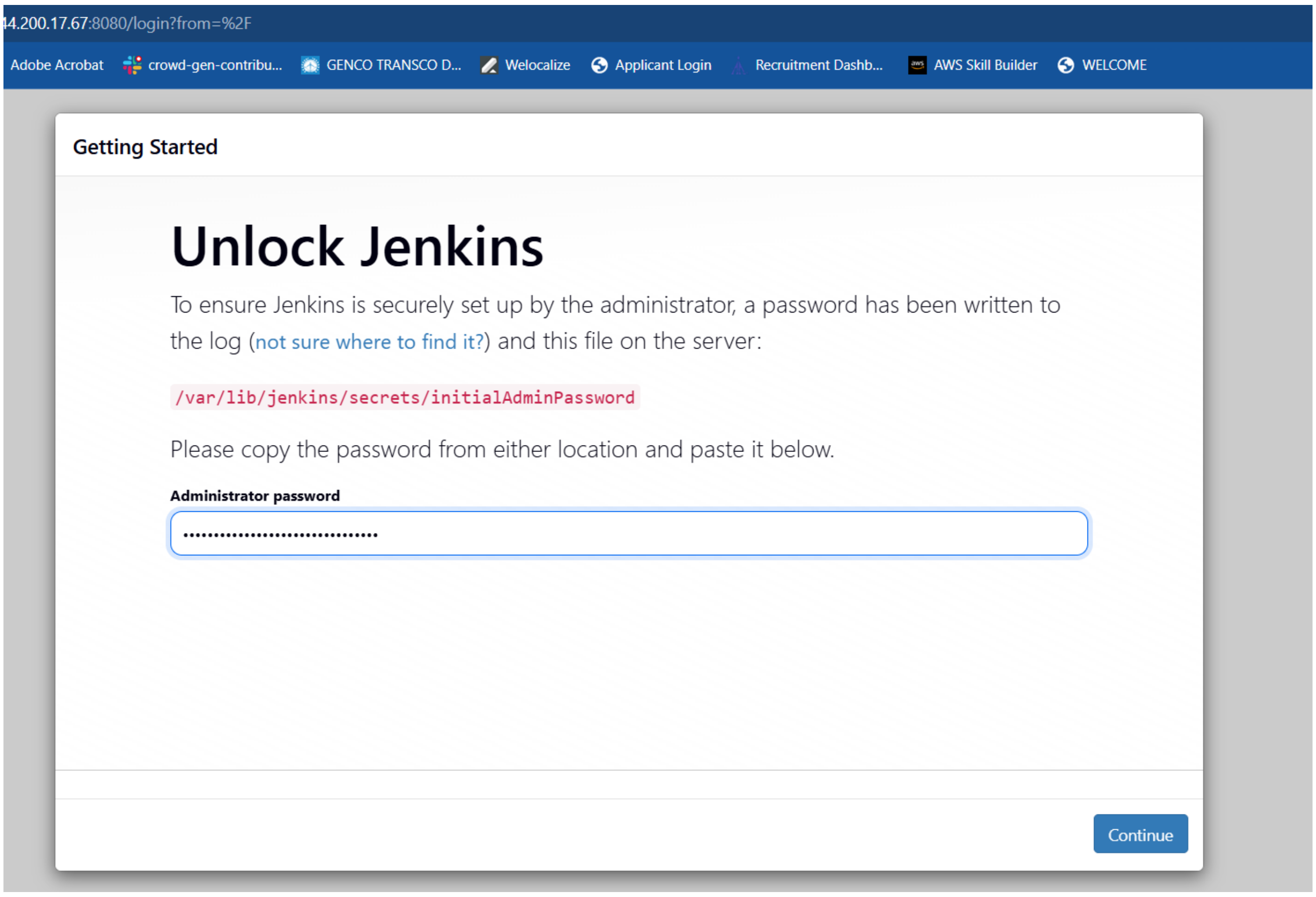
```
ubuntu@ip-10-0-1-98:~/ansible$ ansible-playbook -i hosts setup.yml

PLAY [all] *********************************************************************

TASK [Gathering Facts] *********************************************************
ok: [44.200.197.111]
ok: [44.200.17.67]
ok: [3.237.82.87]
ok: [44.204.209.45]

TASK [Update system] ***********************************************************
changed: [44.200.17.67]
changed: [44.200.197.111]
changed: [3.237.82.87]
changed: [44.204.209.45]

TASK [Install basic packages] **************************************************
changed: [44.200.197.111]
changed: [44.200.17.67]
changed: [3.237.82.87]
changed: [44.204.209.45]

PLAY [jenkins] *****************************************************************

TASK [Gathering Facts] *********************************************************
ok: [44.200.17.67]
```

i-034c60ff7a59f29ab (jenkins-controller)

Step 3 — Verify

Jenkins:

http:// 44.200.17.67:8080



44.200.17.67:8080/login?from=%2F

Getting Started

Unlock Jenkins

To ensure Jenkins is securely set up by the administrator, a password has been written to the log (not sure where to find it?) and this file on the server:

`/var/lib/jenkins/secrets/initialAdminPassword`

Please copy the password from either location and paste it below.

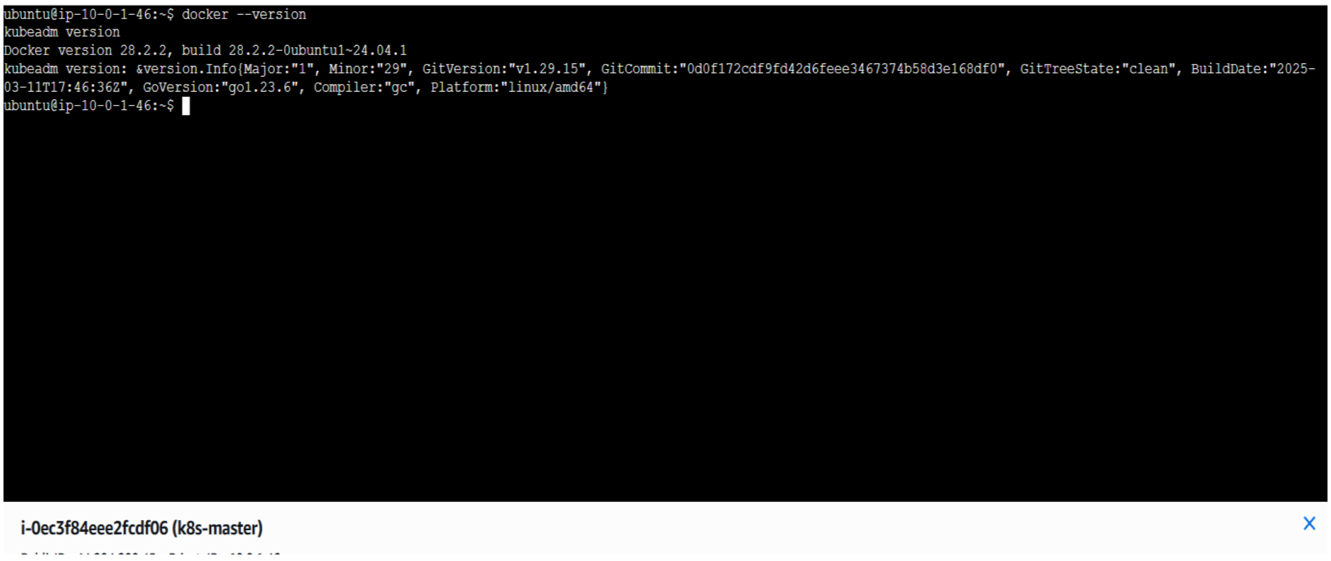
Administrator password

································

Continue

Kubernetes Nodes:

```
docker --version

kubeadm version
```



```
ubuntu@ip-10-0-1-46:~$ docker --version
kubeadm version
Docker version 28.2.2, build 28.2.2-0ubuntu1~24.04.1
kubeadm version: &version.Info{Major:"1", Minor:"29", GitVersion:"v1.29.15", GitCommit:"0d0f172cdf9fd42d6feee3467374b58d3e168df0", GitTreeState:"clean", BuildDate:"2025-
03-11T17:46:36Z", GoVersion:"go1.23.6", Compiler:"gc", Platform:"linux/amd64"}
ubuntu@ip-10-0-1-46:~$
```

i-0ec3f84eee2fcdf06 (k8s-master)

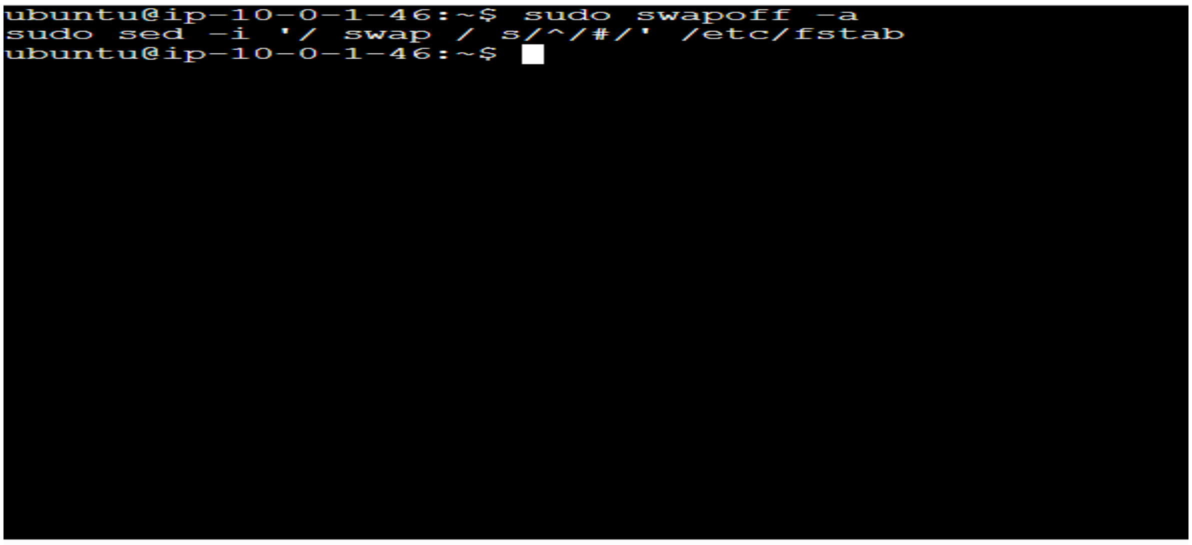
PHASE 9 - Kubernetes Cluster Setup — Production Grade

Cluster Architecture

Role	Node
Kubernetes Master	44.204.209.45
Worker Node 1	44.200.197.111
Worker Node 2	3.237.82.87

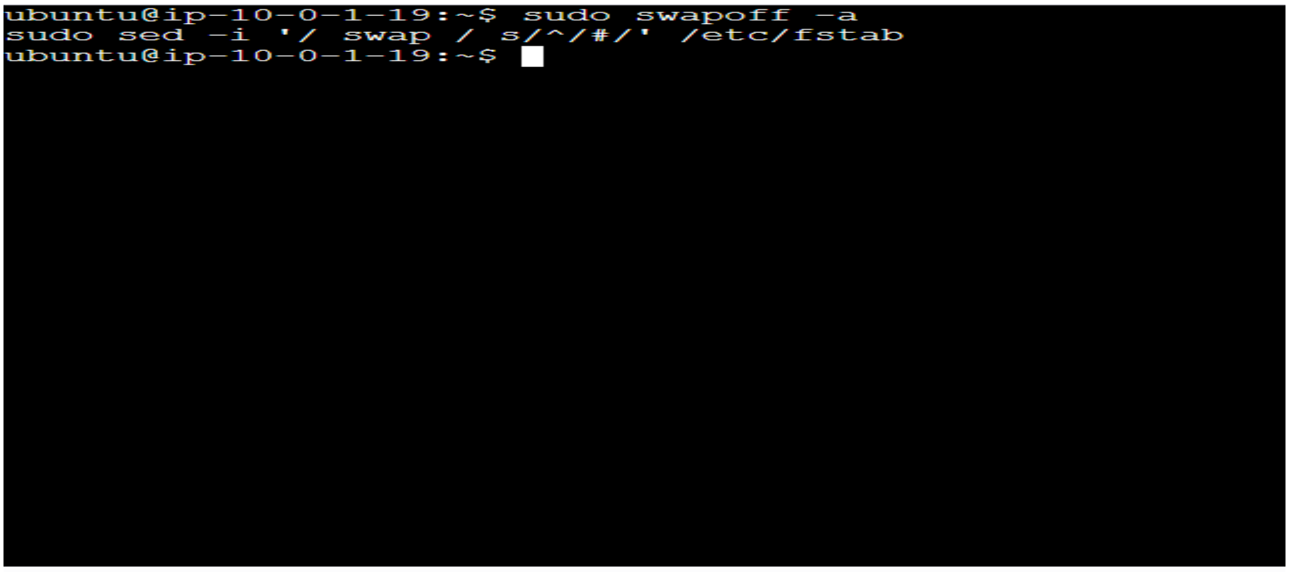
🔷 STEP 1 — Disable Swap (ALL NODES)

```
sudo swapoff -a

sudo sed -i '/ swap / s/^/#/' /etc/fstab
```



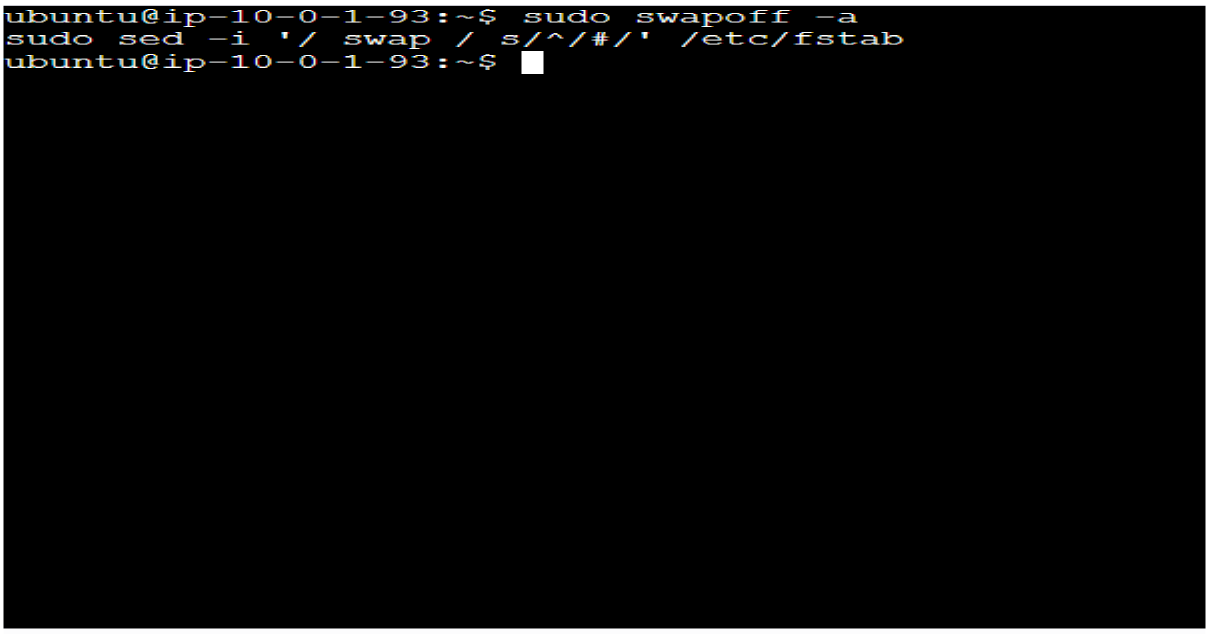
```
ubuntu@ip-10-0-1-46:~$ sudo swapoff -a
sudo sed -i '/ swap / s/^/#/' /etc/fstab
ubuntu@ip-10-0-1-46:~$
```

i-0ec3f84eee2fcdf06 (k8s-master)



```
ubuntu@ip-10-0-1-19:~$ sudo swapoff -a
sudo sed -i '/ swap / s/^/#/' /etc/fstab
ubuntu@ip-10-0-1-19:~$
```

i-09db45f83a70daea3 (k8s-worker1)

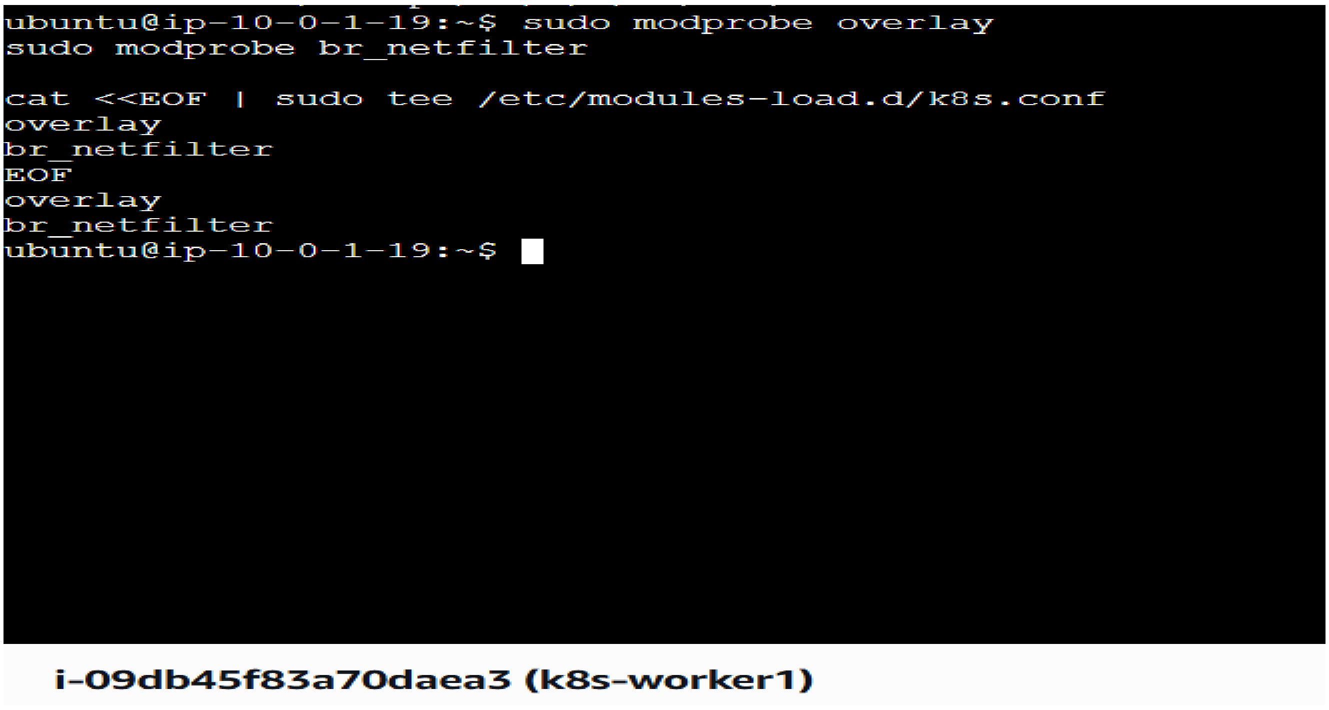


```
ubuntu@ip-10-0-1-93:~$ sudo swapoff -a
sudo sed -i '/ swap / s/^/#/' /etc/fstab
ubuntu@ip-10-0-1-93:~$
```

i-0ffe284ff3e111254 (k8s-worker2)

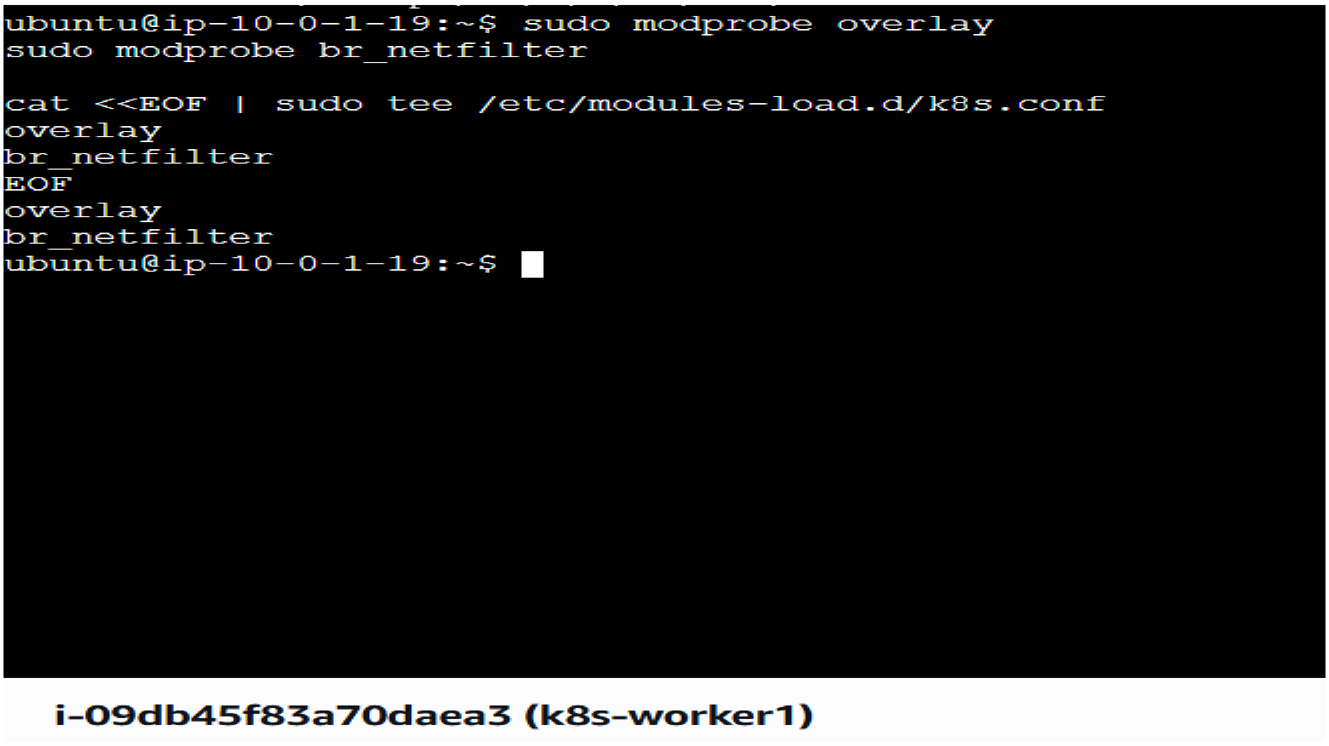
❖ STEP 2 — Load Kernel Modules (ALL NODES)

```
sudo modprobe overlay

sudo modprobe br_netfilter


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

overlay

br_netfilter

EOF
```



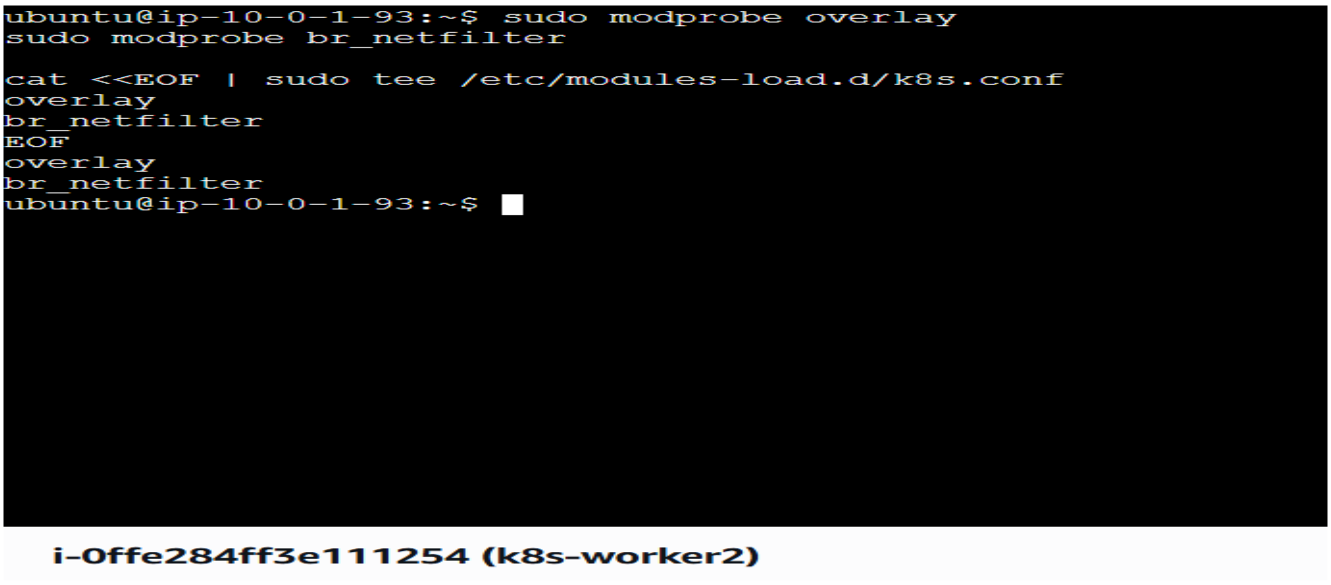
```
ubuntu@ip-10-0-1-19:~$ sudo modprobe overlay
sudo modprobe br_netfilter

cat <<EOF | sudo tee /etc/modules-load.d/k8s.conf
overlay
br_netfilter
EOF
overlay
br_netfilter
ubuntu@ip-10-0-1-19:~$
```

i-09db45f83a70daea3 (k8s-worker1)



```
ubuntu@ip-10-0-1-19:~$ sudo modprobe overlay
sudo modprobe br_netfilter

cat <<EOF | sudo tee /etc/modules-load.d/k8s.conf
overlay
br_netfilter
EOF
overlay
br_netfilter
ubuntu@ip-10-0-1-19:~$
```

i-09db45f83a70daea3 (k8s-worker1)



```
ubuntu@ip-10-0-1-93:~$ sudo modprobe overlay
sudo modprobe br_netfilter

cat <<EOF | sudo tee /etc/modules-load.d/k8s.conf
overlay
br_netfilter
EOF
overlay
br_netfilter
ubuntu@ip-10-0-1-93:~$ 
```

i-0ffe284ff3e111254 (k8s-worker2)

◈ STEP 3 — Configure Network Parameters (ALL NODES)

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

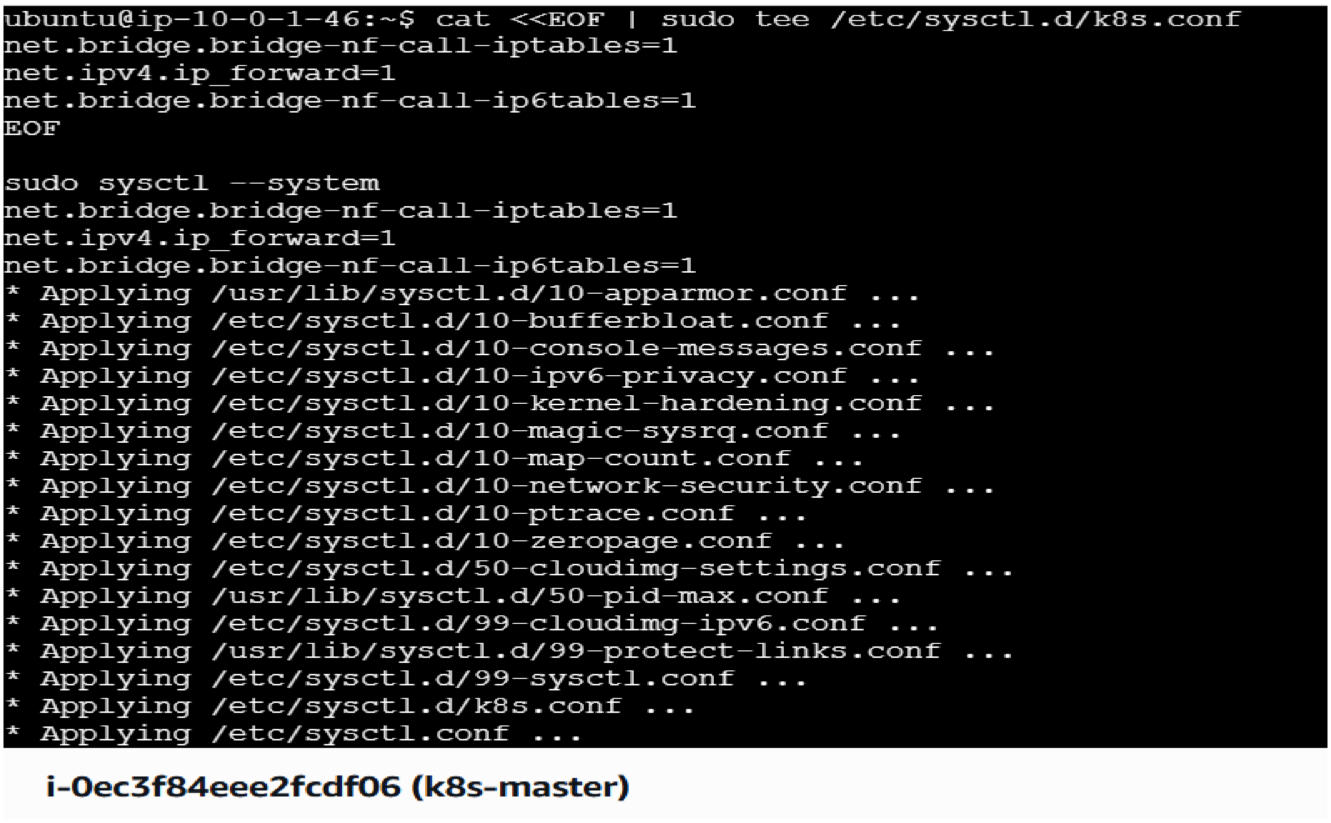
net.bridge.bridge-nf-call-iptables=1

net.ipv4.ip_forward=1

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

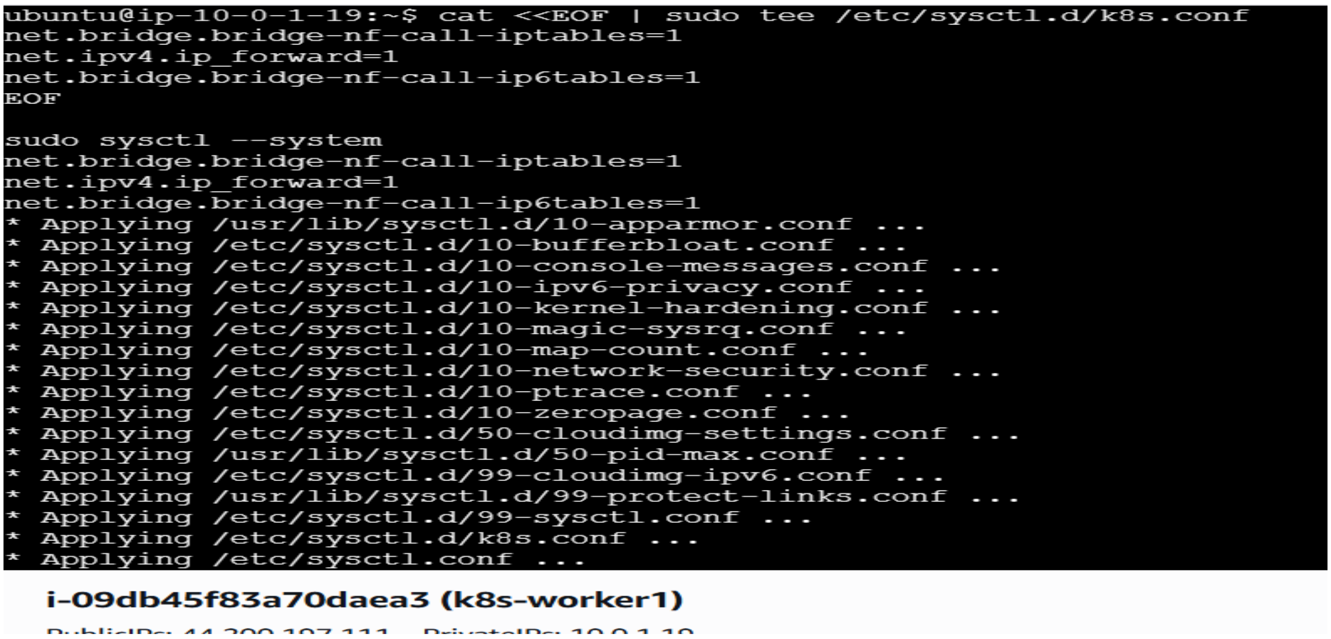
EOF

sudo sysctl –system



```
ubuntu@ip-10-0-1-46:~$ cat <<EOF | sudo tee /etc/sysctl.d/k8s.conf
net.bridge.bridge-nf-call-iptables=1
net.ipv4.ip_forward=1
net.bridge.bridge-nf-call-ip6tables=1
EOF

sudo sysctl --system
net.bridge.bridge-nf-call-iptables=1
net.ipv4.ip_forward=1
net.bridge.bridge-nf-call-ip6tables=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-zeropage.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 ...
```

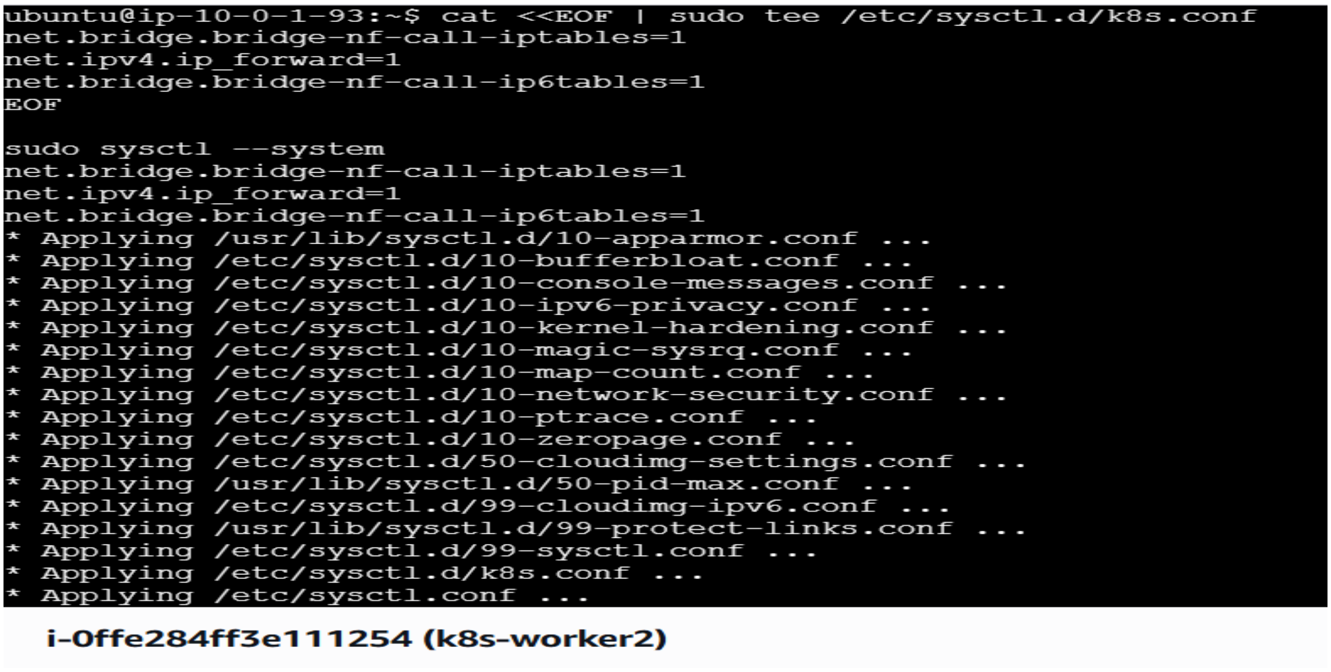
i-0ec3f84eee2fcdf06 (k8s-master)



```
ubuntu@ip-10-0-1-19:~$ cat <<EOF | sudo tee /etc/sysctl.d/k8s.conf
net.bridge.bridge-nf-call-iptables=1
net.ipv4.ip_forward=1
net.bridge.bridge-nf-call-ip6tables=1
EOF

sudo sysctl --system
net.bridge.bridge-nf-call-iptables=1
net.ipv4.ip_forward=1
net.bridge.bridge-nf-call-ip6tables=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-zeropage.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 ...
```

i-09db45f83a70daea3 (k8s-worker1)

PublicIPs: 44.200.187.111 PrivateIPs: 10.0.1.19

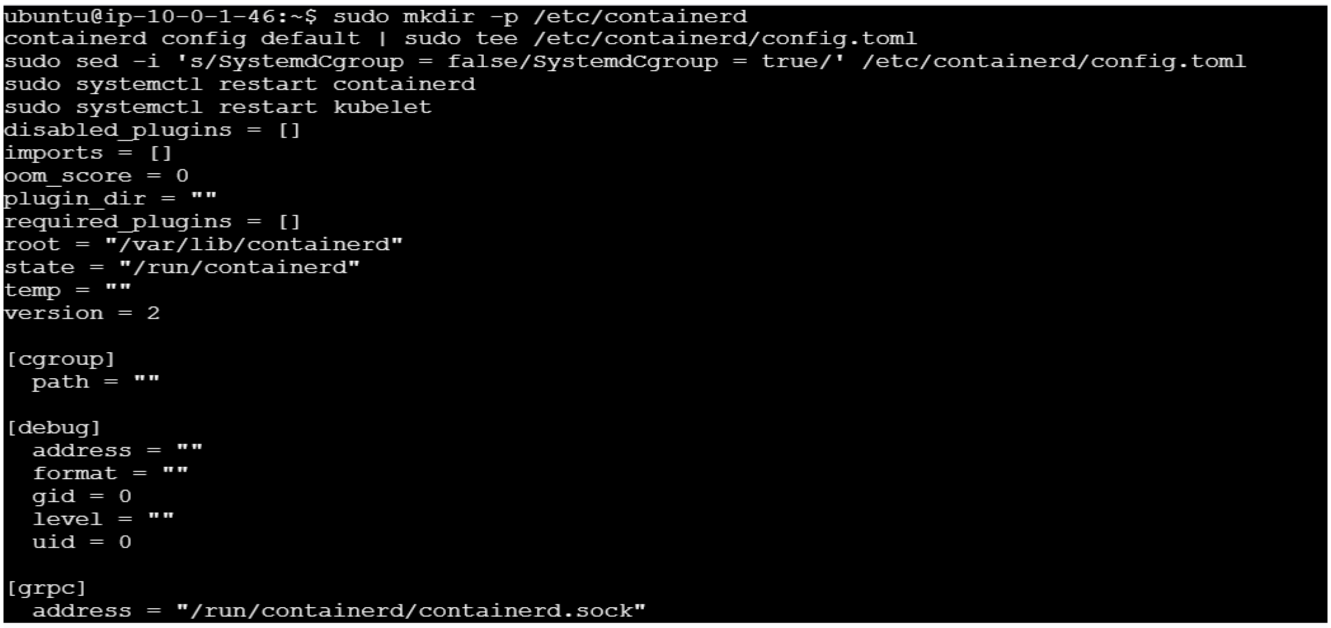


```
ubuntu@ip-10-0-1-93:~$ cat <<EOF | sudo tee /etc/sysctl.d/k8s.conf
net.bridge.bridge-nf-call-iptables=1
net.ipv4.ip_forward=1
net.bridge.bridge-nf-call-ip6tables=1
EOF

sudo sysctl --system
net.bridge.bridge-nf-call-iptables=1
net.ipv4.ip_forward=1
net.bridge.bridge-nf-call-ip6tables=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-zeropage.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 ...
```

i-0ffe284ff3e111254 (k8s-worker2)

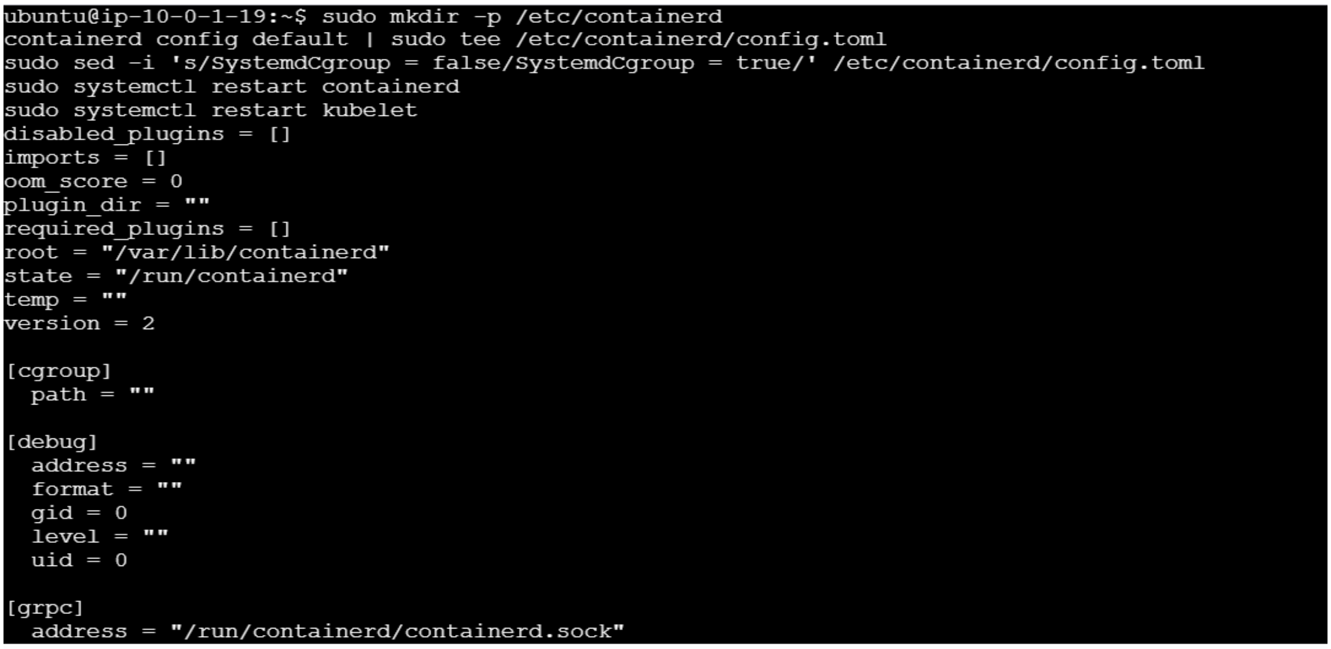
🔷 STEP 4 — Fix Container Runtime (ALL NODES)

```
sudo mkdir -p /etc/containerd
containerd config default | sudo tee /etc/containerd/config.toml
sudo sed -i 's/SystemdCgroup = false/SystemdCgroup = true/' /etc/containerd/config.toml
sudo systemctl restart containerd
sudo systemctl restart kubelet
```



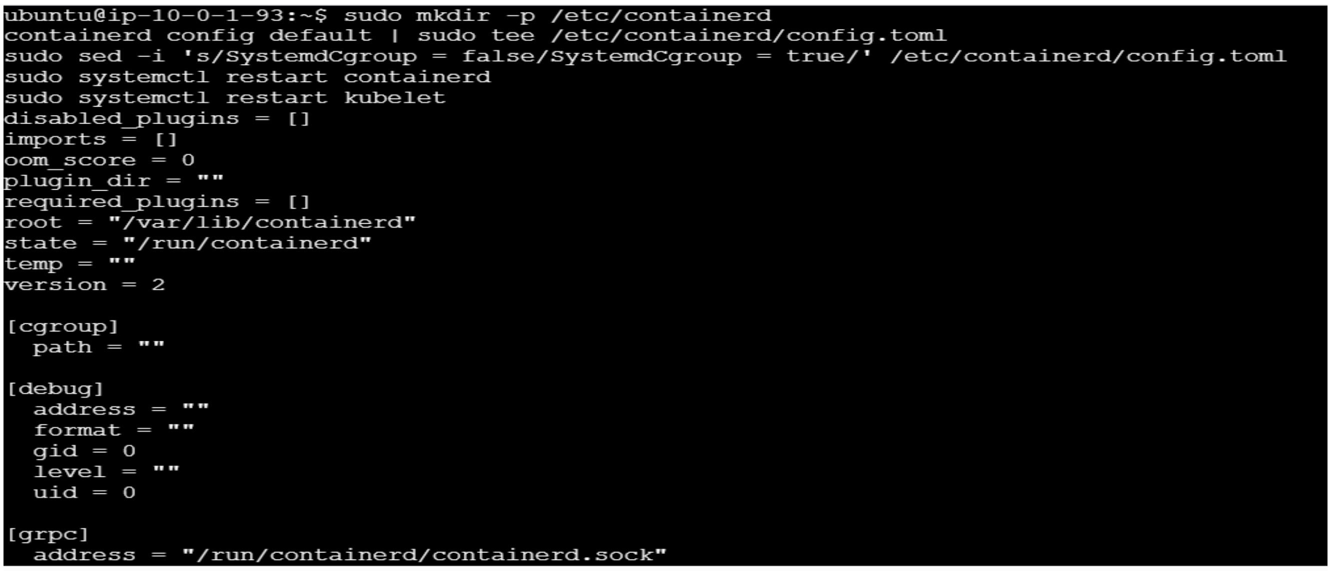
```
ubuntu@ip-10-0-1-46:~$ sudo mkdir -p /etc/containerd
containerd config default | sudo tee /etc/containerd/config.toml
sudo sed -i 's/SystemdCgroup = false/SystemdCgroup = true/' /etc/containerd/config.toml
sudo systemctl restart containerd
sudo systemctl restart kubelet
disabled_plugins = []
imports = []
oom_score = 0
plugin_dir = ""
required_plugins = []
root = "/var/lib/containerd"
state = "/run/containerd"
temp = ""
version = 2

[cgroup]
  path = ""

[debug]
  address = ""
  format = ""
  gid = 0
  level = ""
  uid = 0

[grpc]
  address = "/run/containerd/containerd.sock"
```

i-0ec3f84eee2fcdf06 (k8s-master)



```
ubuntu@ip-10-0-1-19:~$ sudo mkdir -p /etc/containerd
containerd config default | sudo tee /etc/containerd/config.toml
sudo sed -i 's/SystemdCgroup = false/SystemdCgroup = true/' /etc/containerd/config.toml
sudo systemctl restart containerd
sudo systemctl restart kubelet
disabled_plugins = []
imports = []
oom_score = 0
plugin_dir = ""
required_plugins = []
root = "/var/lib/containerd"
state = "/run/containerd"
temp = ""
version = 2

[cgroup]
  path = ""

[debug]
  address = ""
  format = ""
  gid = 0
  level = ""
  uid = 0

[grpc]
  address = "/run/containerd/containerd.sock"
```

i-09db45f83a70daea3 (k8s-worker1)



```
ubuntu@ip-10-0-1-93:~$ sudo mkdir -p /etc/containerd
containerd config default | sudo tee /etc/containerd/config.toml
sudo sed -i 's/SystemdCgroup = false/SystemdCgroup = true/' /etc/containerd/config.toml
sudo systemctl restart containerd
sudo systemctl restart kubelet
disabled_plugins = []
imports = []
oom_score = 0
plugin_dir = ""
required_plugins = []
root = "/var/lib/containerd"
state = "/run/containerd"
temp = ""
version = 2

[cgroup]
  path = ""

[debug]
  address = ""
  format = ""
  gid = 0
  level = ""
  uid = 0

[grpc]
  address = "/run/containerd/containerd.sock"
```

i-0ffe284ff3e111254 (k8s-worker2)

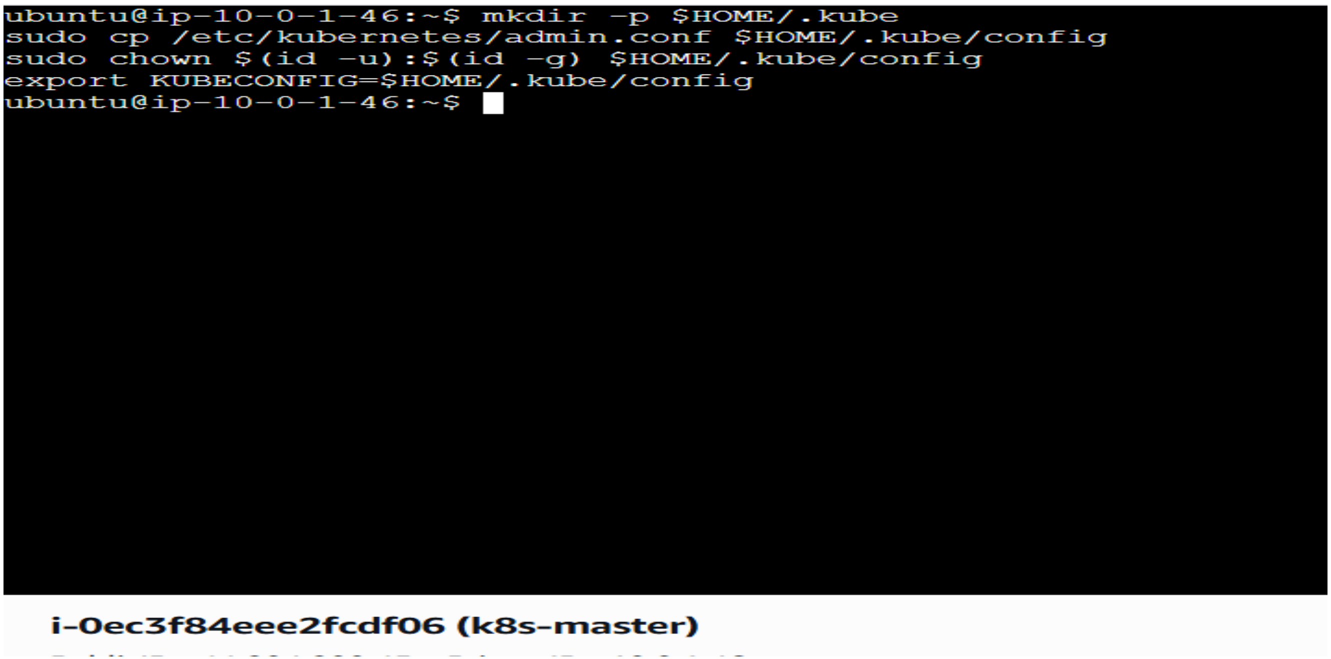
STEP 6 — Initialize Cluster (MASTER ONLY)

```
sudo kubeadm init --pod-network-cidr=192.168.0.0/16
```

Save the kubeadm join command printed at the end.

STEP 7 — Configure kubectl (MASTER ONLY)

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

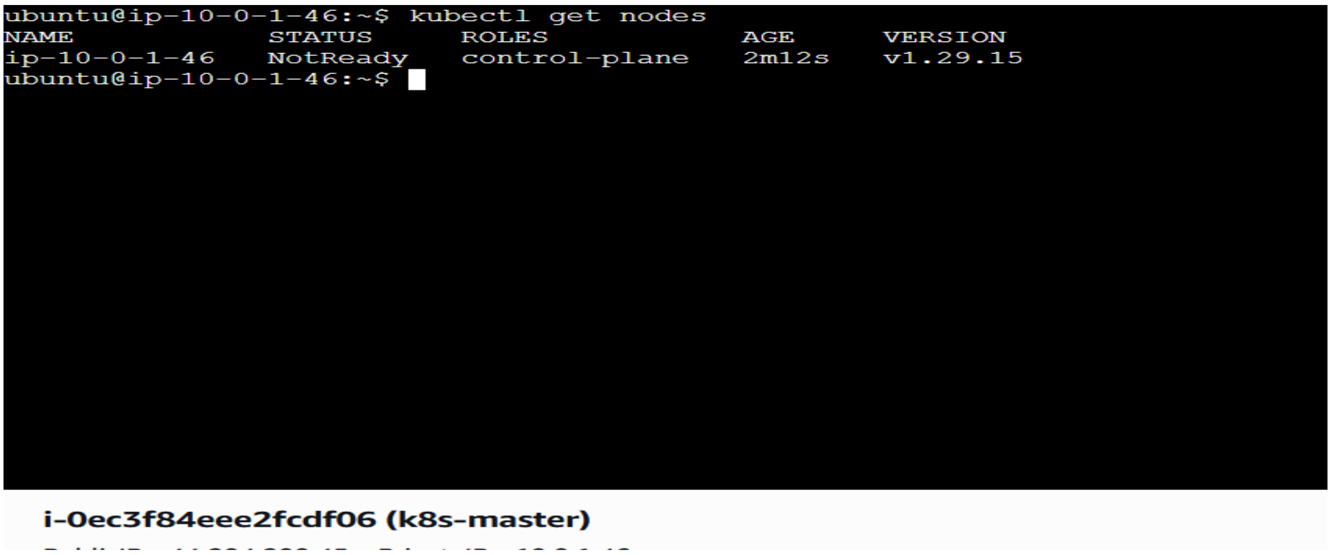


```
ubuntu@ip-10-0-1-46:~$ mkdir -p $HOME/.kube
sudo cp /etc/kubernetes/admin.conf $HOME/.kube/config
sudo chown $(id -u):$(id -g) $HOME/.kube/config
export KUBECONFIG=$HOME/.kube/config
ubuntu@ip-10-0-1-46:~$
```

i-0ec3f84eee2fcdf06 (k8s-master)

Verify:

```
kubectl get nodes
```



```
ubuntu@ip-10-0-1-46:~$ kubectl get nodes
NAME            STATUS     ROLES           AGE     VERSION
ip-10-0-1-46    NotReady   control-plane   2m12s   v1.29.15
ubuntu@ip-10-0-1-46:~$
```

i-0ec3f84eee2fcdf06 (k8s-master)

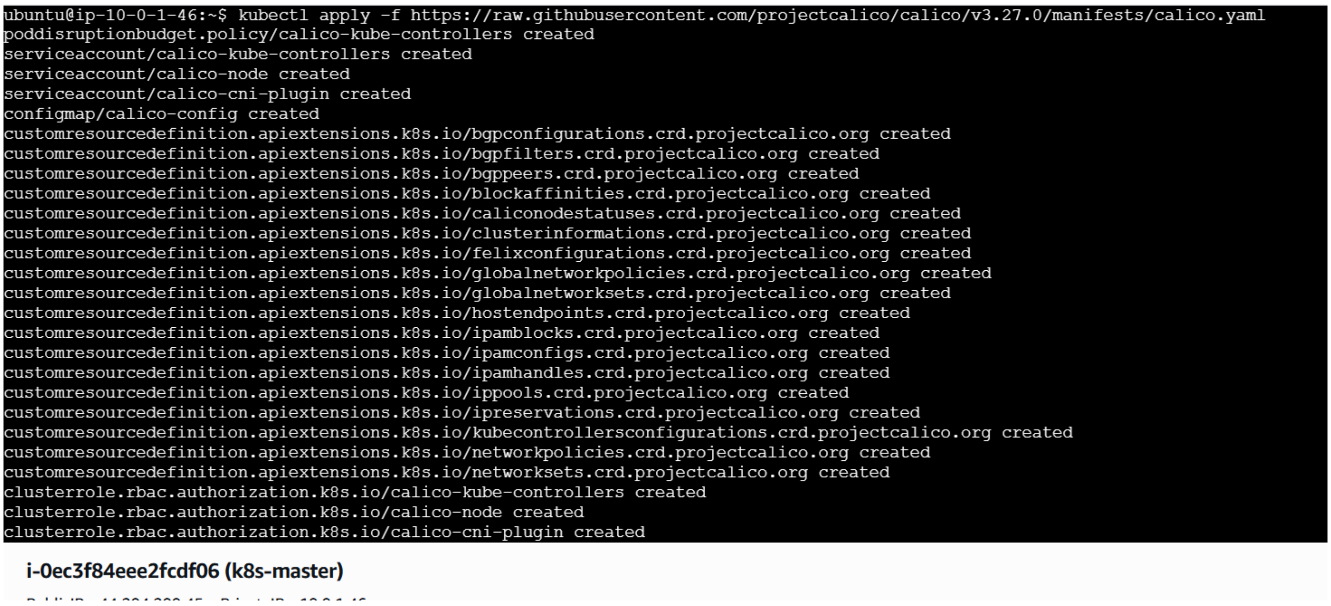
Expected:

k8s-master   NotReady

◈ STEP 8 — Install Network Plugin (Calico)

kubectl apply -f

https://raw.githubusercontent.com/projectcalico/calico/v3.27.0/manifests/calico.yaml



```
ubuntu@ip-10-0-1-46:~$ kubectl apply -f https://raw.githubusercontent.com/projectcalico/calico/v3.27.0/manifests/calico.yaml
poddisruptionbudget.policy/calico-kube-controllers created
serviceaccount/calico-kube-controllers created
serviceaccount/calico-node created
serviceaccount/calico-cni-plugin created
configmap/calico-config created
customresourcedefinition.apiextensions.k8s.io/bgpconfigurations.crd.projectcalico.org created
customresourcedefinition.apiextensions.k8s.io/bgpfilters.crd.projectcalico.org created
customresourcedefinition.apiextensions.k8s.io/bgppeers.crd.projectcalico.org created
customresourcedefinition.apiextensions.k8s.io/blockaffinities.crd.projectcalico.org created
customresourcedefinition.apiextensions.k8s.io/caliconodestatuses.crd.projectcalico.org created
customresourcedefinition.apiextensions.k8s.io/clusterinformations.crd.projectcalico.org created
customresourcedefinition.apiextensions.k8s.io/felixconfigurations.crd.projectcalico.org created
customresourcedefinition.apiextensions.k8s.io/globalnetworkpolicies.crd.projectcalico.org created
customresourcedefinition.apiextensions.k8s.io/globalnetworksets.crd.projectcalico.org created
customresourcedefinition.apiextensions.k8s.io/hostendpoints.crd.projectcalico.org created
customresourcedefinition.apiextensions.k8s.io/ipamblocks.crd.projectcalico.org created
customresourcedefinition.apiextensions.k8s.io/ipamconfigs.crd.projectcalico.org created
customresourcedefinition.apiextensions.k8s.io/ipamhandles.crd.projectcalico.org created
customresourcedefinition.apiextensions.k8s.io/ippools.crd.projectcalico.org created
customresourcedefinition.apiextensions.k8s.io/ipreservations.crd.projectcalico.org created
customresourcedefinition.apiextensions.k8s.io/kubecontrollersconfigurations.crd.projectcalico.org created
customresourcedefinition.apiextensions.k8s.io/networkpolicies.crd.projectcalico.org created
customresourcedefinition.apiextensions.k8s.io/networksets.crd.projectcalico.org created
clusterrole.rbac.authorization.k8s.io/calico-kube-controllers created
clusterrole.rbac.authorization.k8s.io/calico-node created
clusterrole.rbac.authorization.k8s.io/calico-cni-plugin created
```

i-0ec3f84eee2fcdf06 (k8s-master)

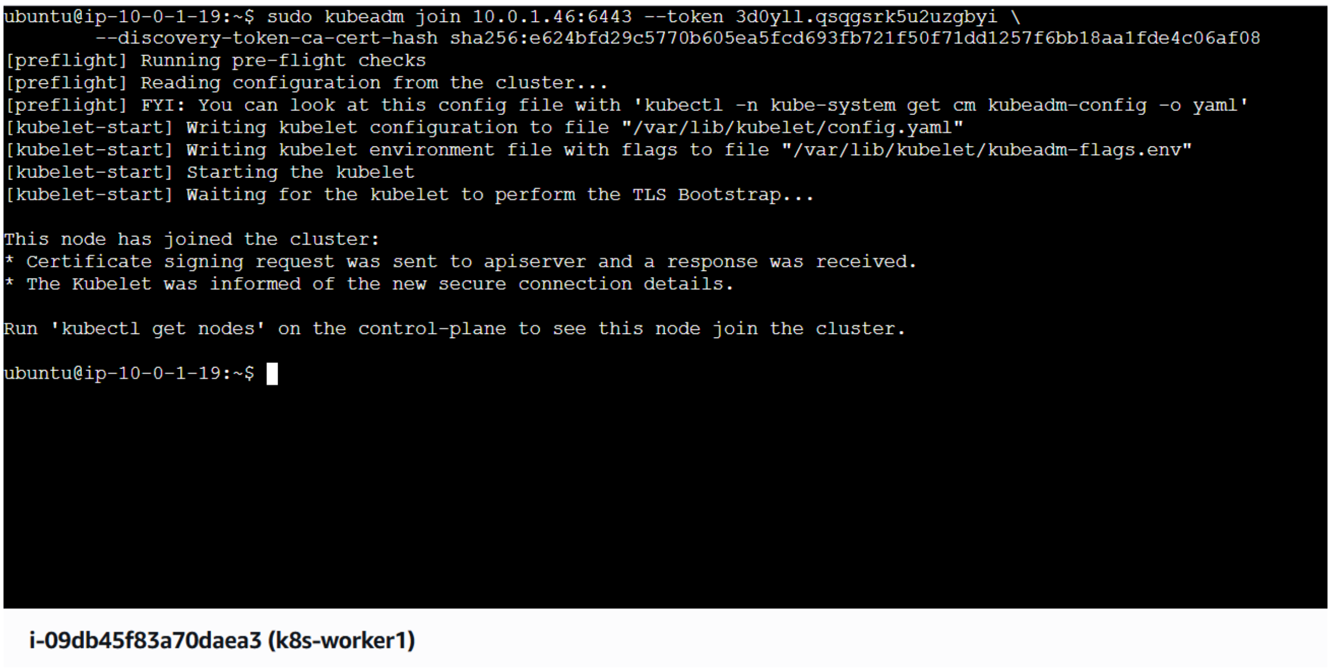
◈ STEP 9 — Join Worker Nodes

Run on each worker node:

sudo kubeadm join 10.0.1.46:6443 --token 3d0yll.qsqgsrk5u2uzgbyi \

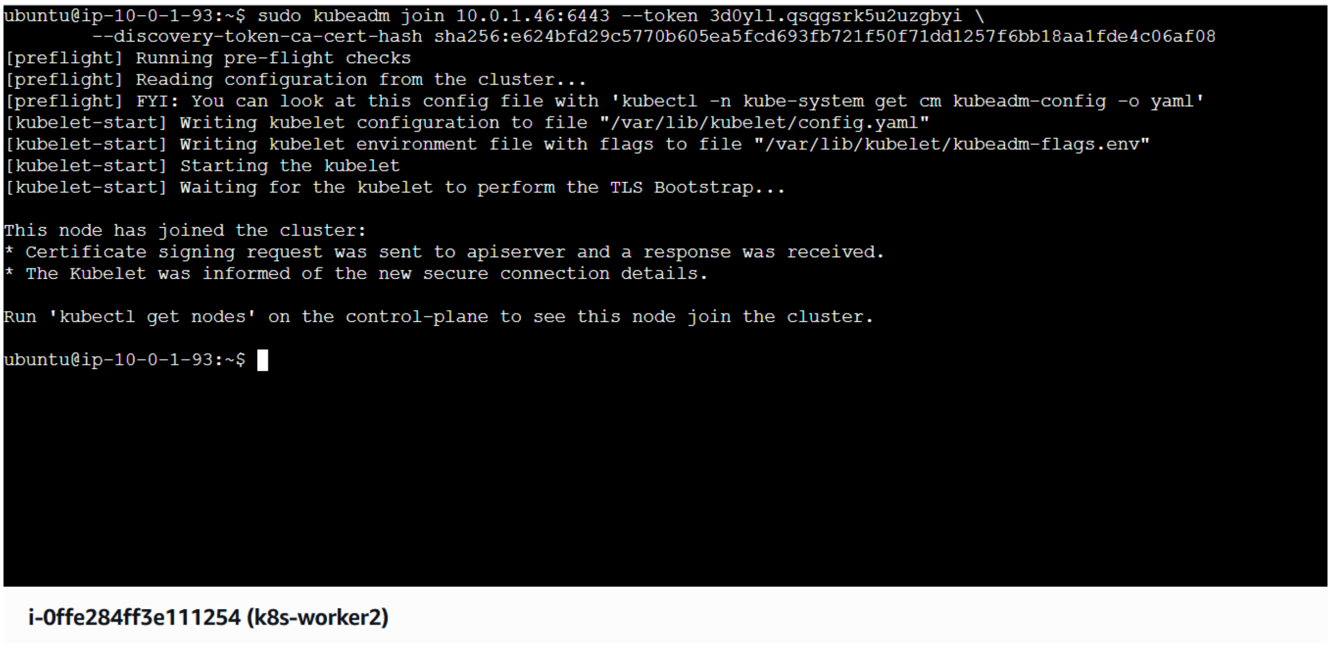
    --discovery-token-ca-cert-hash

sha256:e624bfd29c5770b605ea5fcd693fb721f50f71dd1257f6bb18aa1fde4c06af08



```
ubuntu@ip-10-0-1-19:~$ sudo kubeadm join 10.0.1.46:6443 --token 3d0yll.qsqgsrk5u2uzgbyi \
        --discovery-token-ca-cert-hash sha256:e624bfd29c5770b605ea5fcd693fb721f50f71dd1257f6bb18aa1fde4c06af08
[preflight] Running pre-flight checks
[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'
[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...

This node has joined the cluster:
* Certificate signing request was sent to apiserver 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.

ubuntu@ip-10-0-1-19:~$
```

i-09db45f83a70daea3 (k8s-worker1)



```
ubuntu@ip-10-0-1-93:~$ sudo kubeadm join 10.0.1.46:6443 --token 3d0yll.qsqgsrk5u2uzgbyi \
        --discovery-token-ca-cert-hash sha256:e624bfd29c5770b605ea5fcd693fb721f50f71dd1257f6bb18aa1fde4c06af08
[preflight] Running pre-flight checks
[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'
[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...

This node has joined the cluster:
* Certificate signing request was sent to apiserver 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.

ubuntu@ip-10-0-1-93:~$
```

i-0ffe284ff3e111254 (k8s-worker2)

◈ STEP 10 — Final Cluster Verification (MASTER)

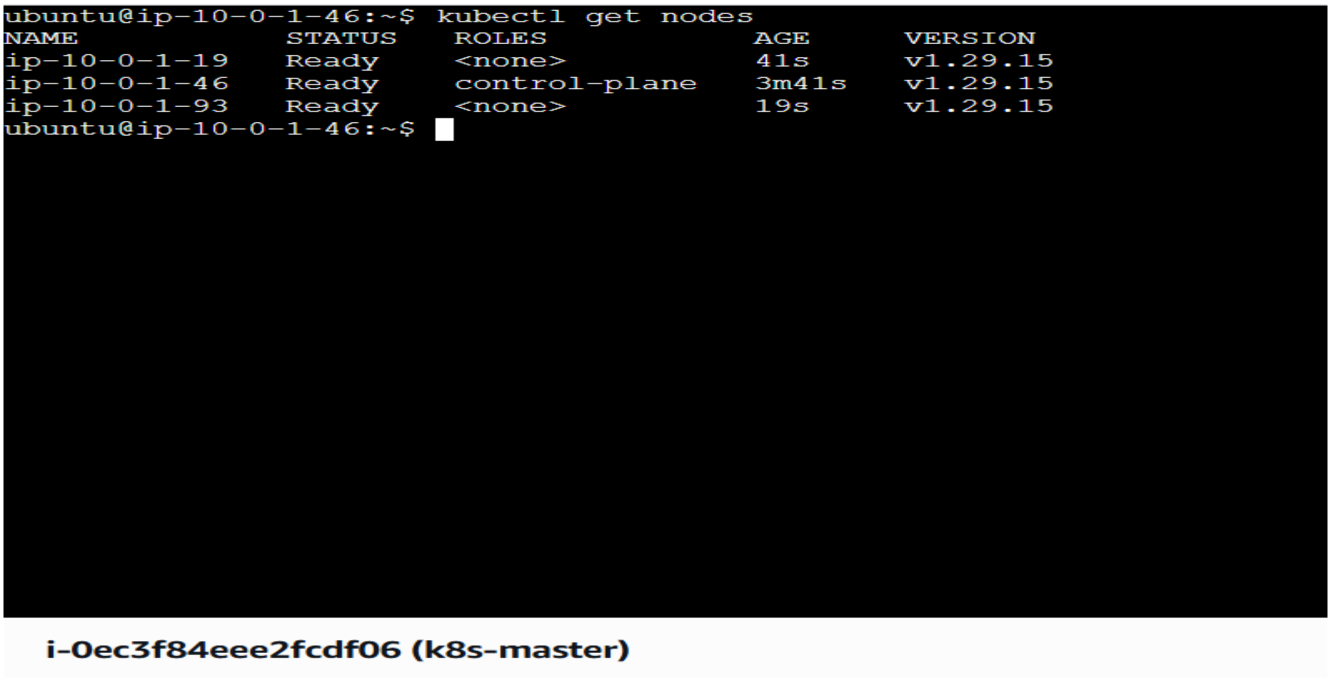
kubectl get nodes

Final expected state:

k8s-master   Ready

k8s-worker1   Ready

k8s-worker2   Ready



```
ubuntu@ip-10-0-1-46:~$ kubectl get nodes
NAME             STATUS    ROLES             AGE      VERSION
ip-10-0-1-19     Ready     <none>            41s      v1.29.15
ip-10-0-1-46     Ready     control-plane     3m41s    v1.29.15
ip-10-0-1-93     Ready     <none>            19s      v1.29.15
ubuntu@ip-10-0-1-46:~$
```

i-0ec3f84eee2fcdf06 (k8s-master)

PHASE 10 - Application Deployment + Jenkins CI/CD Pipeline

Pipeline Architecture

Developer Commit → GitHub → Jenkins Pipeline

↓

Docker Build → Docker Hub

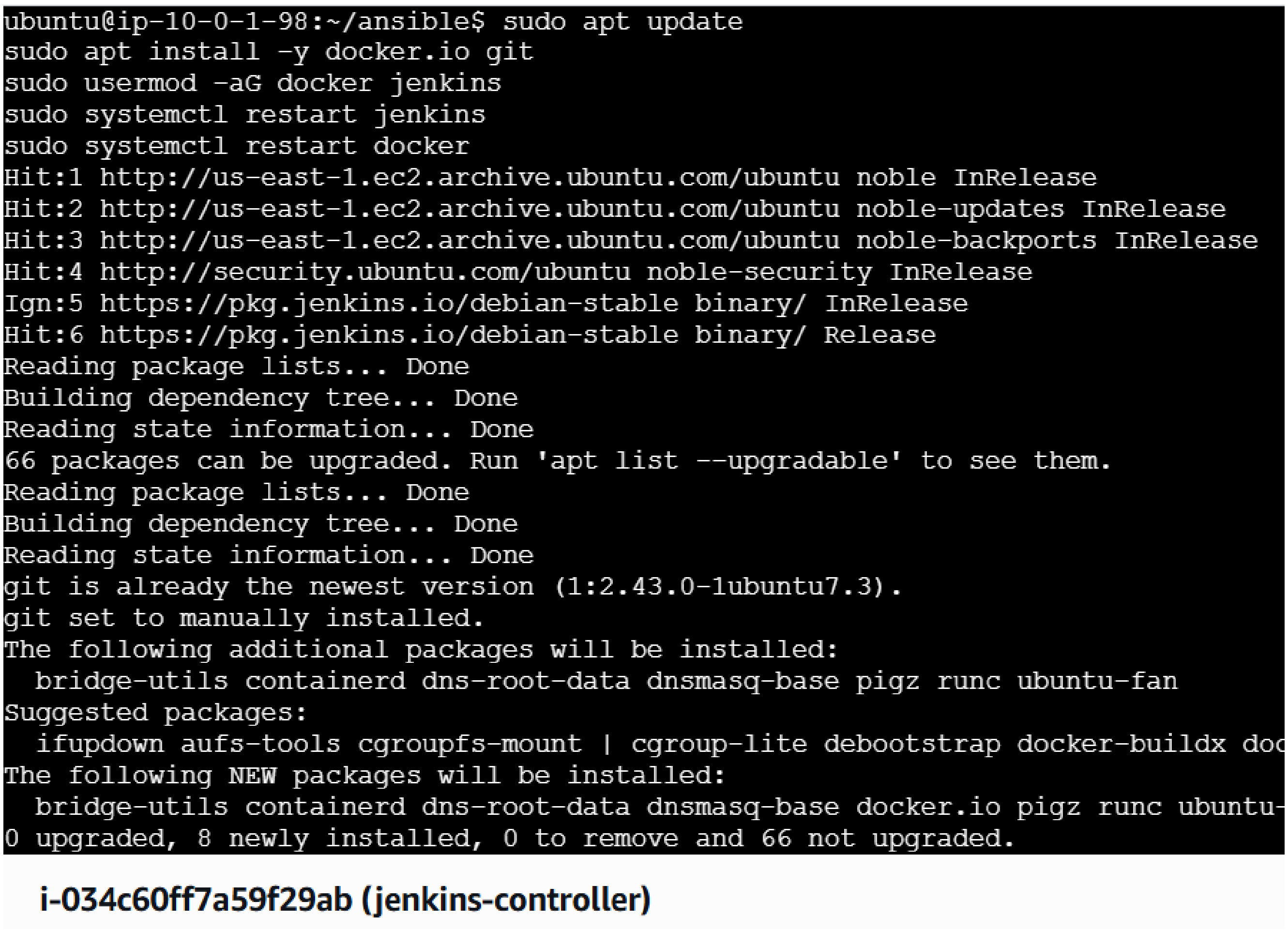
↓

Kubernetes Deployment (2 replicas)

STEP 1 — Jenkins Preparation

Install Required Software on Jenkins Server

```
sudo apt update

sudo apt install -y docker.io git

sudo usermod -aG docker jenkins

sudo systemctl restart jenkins

sudo systemctl restart docker
```



```
ubuntu@ip-10-0-1-98:~/ansible$ sudo apt update
sudo apt install -y docker.io git
sudo usermod -aG docker jenkins
sudo systemctl restart jenkins
sudo systemctl restart docker
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 http://security.ubuntu.com/ubuntu noble-security InRelease
Ign:5 https://pkg.jenkins.io/debian-stable binary/ InRelease
Hit:6 https://pkg.jenkins.io/debian-stable binary/ Release
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
66 packages can be upgraded. Run 'apt list --upgradable' to see them.
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
git is already the newest version (1:2.43.0-1ubuntu7.3).
git set to manually installed.
The following additional packages will be installed:
  bridge-utils containerd dns-root-data dnsmasq-base pigz runc ubuntu-fan
Suggested packages:
  ifupdown aufs-tools cgroupfs-mount | cgroup-lite debootstrap docker-buildx doc
The following NEW packages will be installed:
  bridge-utils containerd dns-root-data dnsmasq-base docker.io pigz runc ubuntu-
0 upgraded, 8 newly installed, 0 to remove and 66 not upgraded.
```

i-034c60ff7a59f29ab (jenkins-controller)

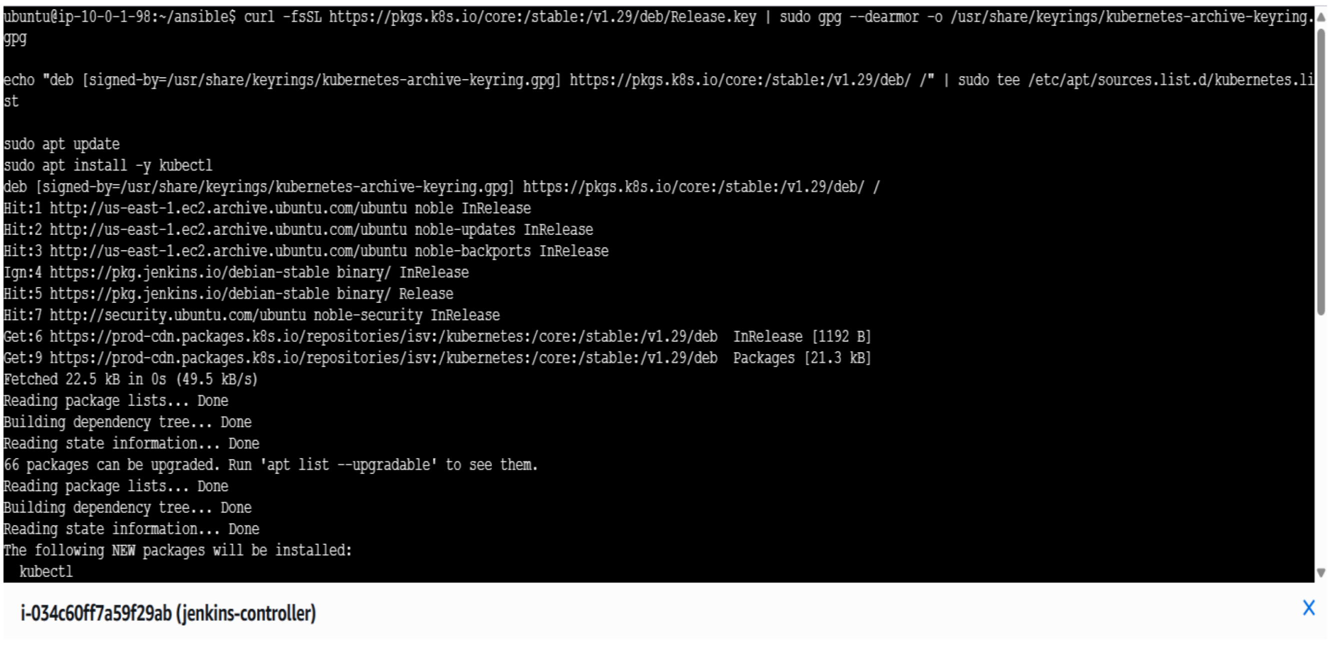
Install kubectl on Jenkins Server

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

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

sudo apt update

sudo apt install -y kubectl



```
ubuntu@ip-10-0-1-98:~/ansible$ curl -fsSL https://pkgs.k8s.io/core:/stable:/v1.29/deb/Release.key | sudo gpg --dearmor -o /usr/share/keyrings/kubernetes-archive-keyring.
gpg

echo "deb [signed-by=/usr/share/keyrings/kubernetes-archive-keyring.gpg] https://pkgs.k8s.io/core:/stable:/v1.29/deb/ /" | sudo tee /etc/apt/sources.list.d/kubernetes.li
st

sudo apt update
sudo apt install -y kubectl
deb [signed-by=/usr/share/keyrings/kubernetes-archive-keyring.gpg] https://pkgs.k8s.io/core:/stable:/v1.29/deb/ /
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
Ign:4 https://pkg.jenkins.io/debian-stable binary/ InRelease
Hit:5 https://pkg.jenkins.io/debian-stable binary/ Release
Hit:7 http://security.ubuntu.com/ubuntu noble-security InRelease
Get:6 https://prod-cdn.packages.k8s.io/repositories/isv:/kubernetes:/core:/stable:/v1.29/deb  InRelease [1192 B]
Get:9 https://prod-cdn.packages.k8s.io/repositories/isv:/kubernetes:/core:/stable:/v1.29/deb  Packages [21.3 kB]
Fetched 22.5 kB in 0s (49.5 kB/s)
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
66 packages can be upgraded. Run 'apt list --upgradable' to see them.
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following NEW packages will be installed:
  kubectl
```

i-034c60ff7a59f29ab (jenkins-controller)

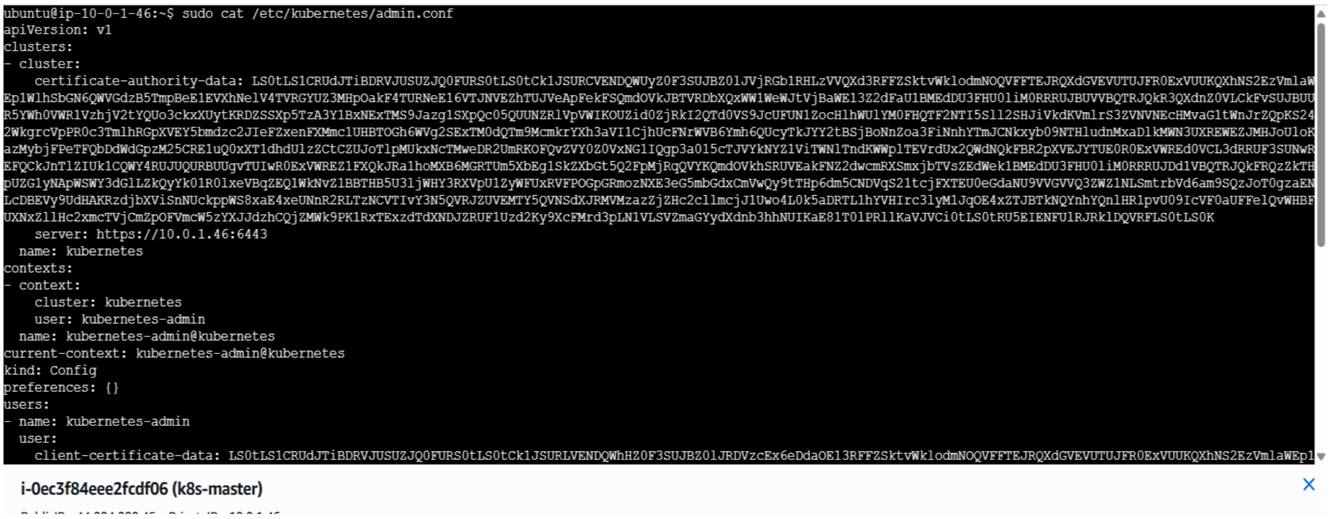
X

Provide Kubernetes Access to Jenkins

On k8s-master:

sudo cat /etc/kubernetes/admin.conf

Copy content.



```
ubuntu@ip-10-0-1-46:~$ sudo cat /etc/kubernetes/admin.conf
apiVersion: v1
clusters:
- cluster:
    certificate-authority-data: LS0tLS1CRUdJTiBDRVJUSUZJQ0FURS0tLS0tCk1JSURCVENDQWUyZ0F3SUJBZ0lJVjRGb1RHLzVVQXd3RFFZSktvWklodmNOQVFFTEJRQXdGVEVUTUJFR0ExVUUKQXhNS2EzVmlaW
Ep1WlhSbGN6QWVGdzB5TmpBeE1EVXhNelV4TVRGYUZ3MHpOakF4TURNeE16VTJNVEZhTUJVeApFekFSQmdOVkJBTVRDbXQxWW1WeWJtVjBaWE13Z2dFaU1BMEdDU3FHU0liM0RRRUJBUVVBQTRJQkR3QXdnZ0VLCkFvSUJBUU
R5YWh0VWR1VzhjV2tYQUo3ckxXUytKRDZSSXp5TzA3Y1BxNExTMS9Jazg1SXpQc05QUUNZRlVpVWIKOUZid0ZjRkI2QTd0VS9JcUFUN1ZocHlhWUlYM0FHQTF2NTI5Sll2SHJiVkdKVmlrS3ZVNVNEcHMvaG1tWnJrZQpKS24
2WkgrcVpPR0c3TmlhRGpXVEY5bmdzc2JIeFZxenFXMmc1UHBTOGh6WVg2SExTM0dQTm9McmkrYXh3aVI1CjhUcFNrWVB6Ymh6QUcyTkJYY2tBSjBoNnZoa3FiNnhYTmJCNkxyb09NTHludnMxaDlkMWN3UXREWEZJMHJoUloK
azMybjFPeTFQbDdWdGpzM25CRE1uQ0xXT1dhdUlzZCtCZUJoTlpMUkxNcTMweDR2UmRKOFQvZVY0Z0VxNG1IQgp3a015cTJVYkNYZ1ViTWNlTndKWWplTEVrdUx2QWdNQkFBR2pXVEJYTUE0R0ExVWREd0VCL3dRRUF3SUNwR
EFQCkJnTlZIUk1CQWY4RUJUQURBUUgvTUIwR0ExVWREZ1FXQkJRa1hoMXB6MGRTUm5XbEg1SkZXbGt5Q2FpMjRqQVYKQmdOVkhSUVVEakFNZ2dwcmRXSmxjbTVsZEdWek1BMEdDU3FHU0liM0RRRUJDd1VBQTRJQkFRQzZkTH
pUZG1yNApWSWY3dGlLZkQyYk01R0lxeVBqZEQ1WkNvZ1BBTHB5U3ljWHY3RXVpU1ZyWFUxRVFPOGpGRmozNXE3eG5mbGdxCmVwQy9tTHp6dm5CNDVqS21tcjFXTEU0eGdaNU9VVGVVQ3ZWZ1NLSmtrbVd6am9SQzJoT0gzaEN
LcDBEVy9UdHAKRzdjbXViSnNUckppWS8xaE4xeUNnR2RLTzNCVTIvY3N5QVRJZUVEMTY5QVNSdXJRMVMzazZjZHc2cllmcjJ1Uwo4L0k5aDRTL1hYVHIrc3lyM1JqOE4xZTJBTkNQYnhYQnlHR1pvU09IcVF0aUFFelQvWHBF
UXNxZllHc2xmcTVjCmZpOFVmcW5zYXJJdzhCQjZMWk9PK1RxTExzdTdXNDJZRUF1Uzd2Ky9XcFMrd3pLN1VLSVZmaGYydXdnb3hhNUIKaE81T0lPRllKaVJVCi0tLS0tRU5EIENFUlRJRklDQVRFLS0tLS0K
    server: https://10.0.1.46:6443
  name: kubernetes
contexts:
- context:
    cluster: kubernetes
    user: kubernetes-admin
  name: kubernetes-admin@kubernetes
current-context: kubernetes-admin@kubernetes
kind: Config
preferences: {}
users:
- name: kubernetes-admin
  user:
    client-certificate-data: LS0tLS1CRUdJTiBDRVJUSUZJQ0FURS0tLS0tCk1JSURLVENDQWhHZ0F3SUJBZ0lJRDVzcEx6eDdaOE13RFFZSktvWklodmNOQVFFTEJRQXdGVEVUTUJFR0ExVUUKQXhNS2EzVmlaWEp1
```

i-0ec3f84eee2fcdf06 (k8s-master)

X

On Jenkins server:

sudo mkdir -p /var/lib/jenkins/.kube

sudo nano /var/lib/jenkins/.kube/config

 # paste admin.conf content

Save and exit

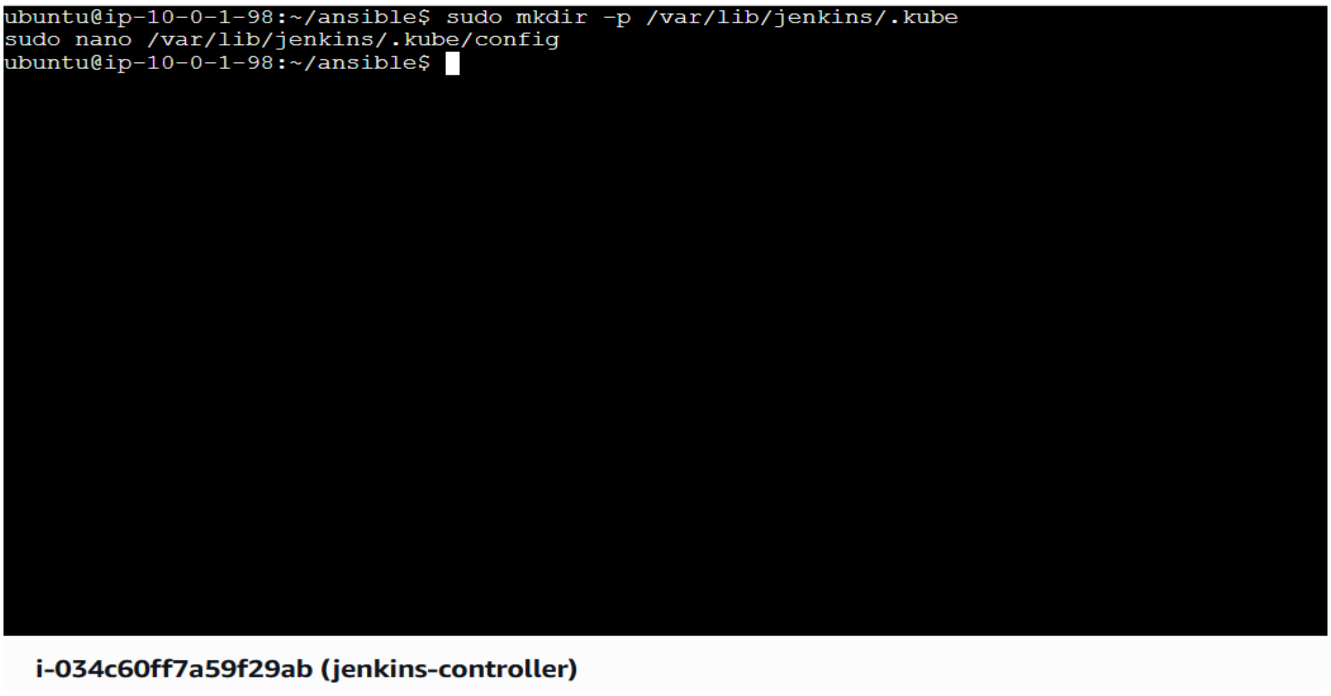
sudo chown -R jenkins:jenkins /var/lib/jenkins/.kube

sudo chmod 600 /var/lib/jenkins/.kube/config

sudo systemctl restart jenkins

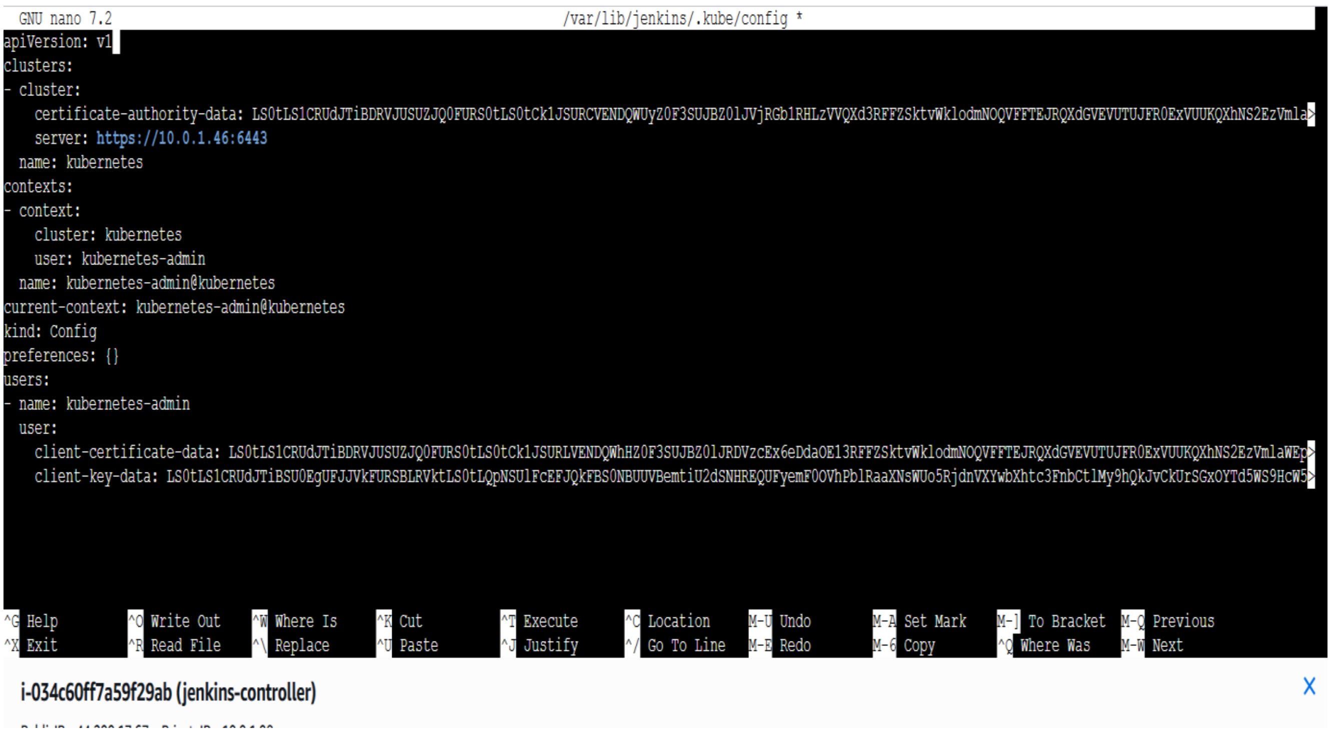
Verify:

sudo -u jenkins kubectl get nodes



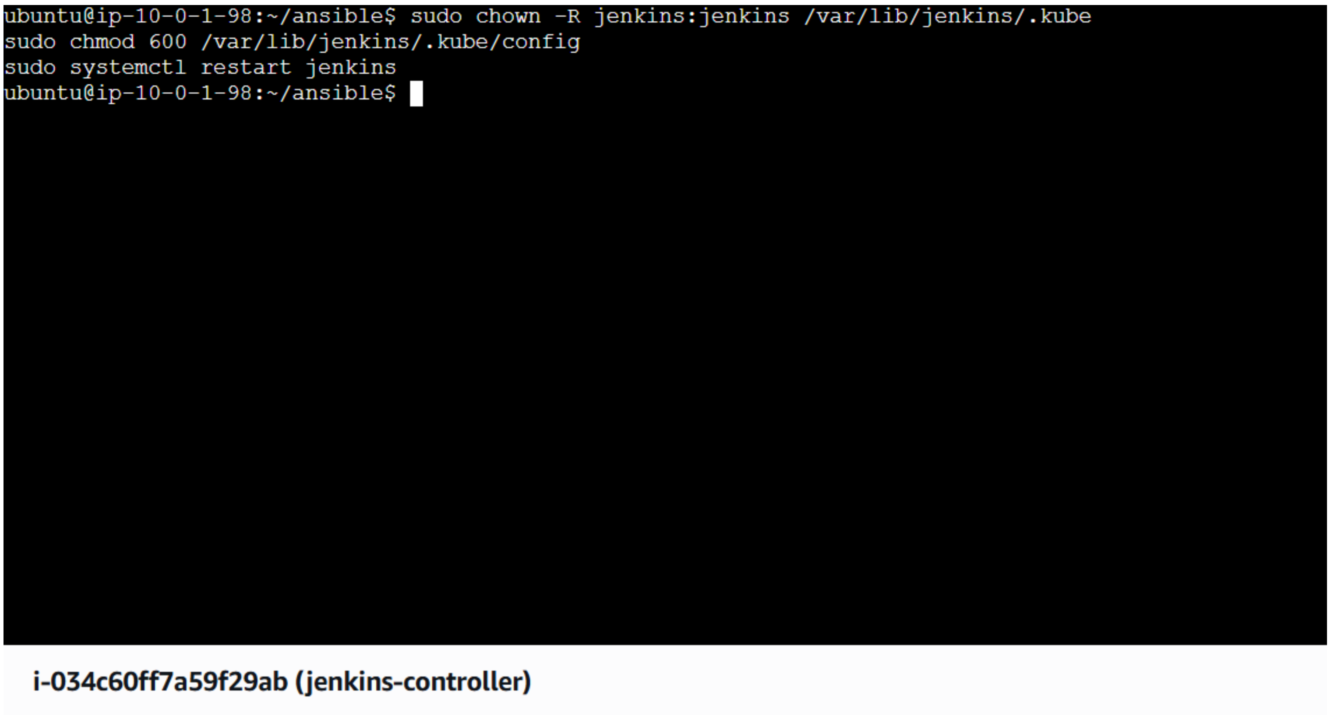
```
ubuntu@ip-10-0-1-98:~/ansible$ sudo mkdir -p /var/lib/jenkins/.kube
sudo nano /var/lib/jenkins/.kube/config
ubuntu@ip-10-0-1-98:~/ansible$
```

i-034c60ff7a59f29ab (jenkins-controller)



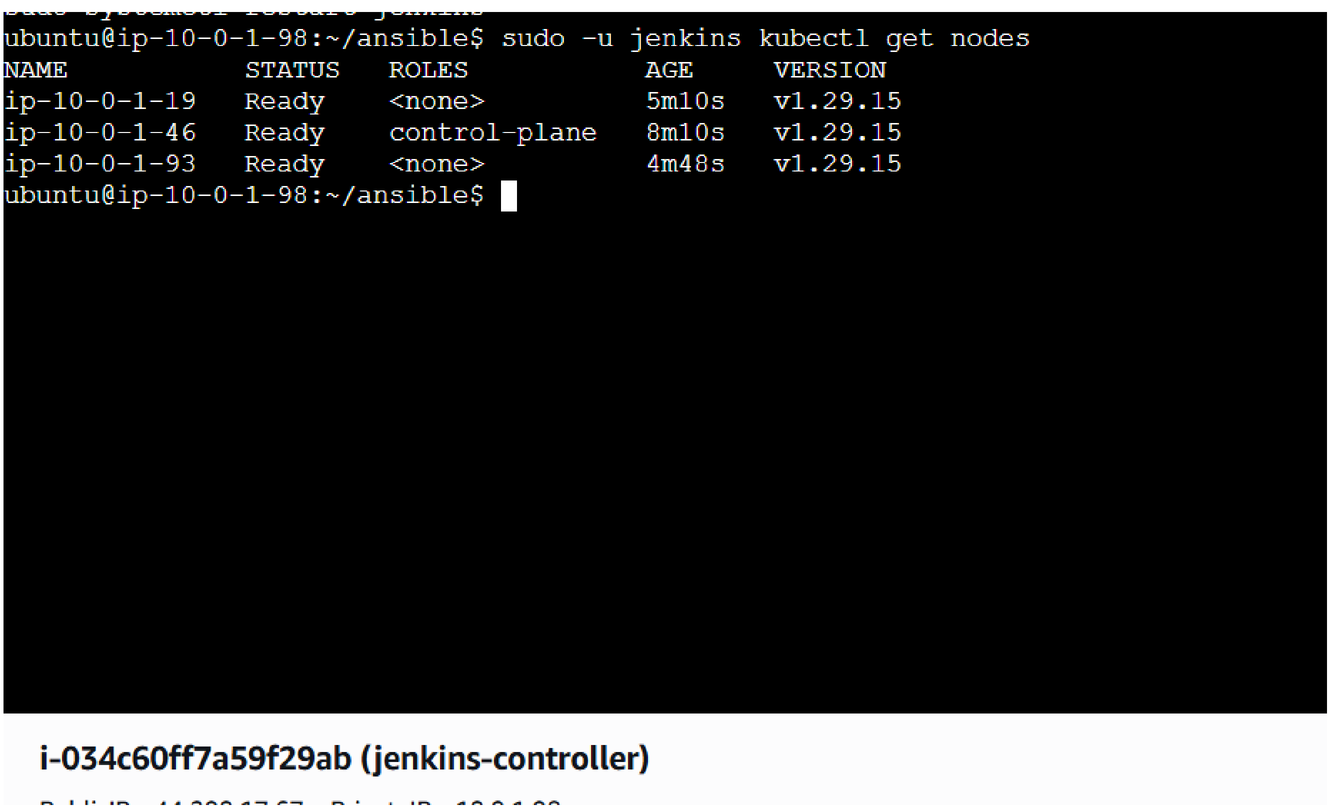
```
  GNU nano 7.2                                   /var/lib/jenkins/.kube/config *
apiVersion: v1
clusters:
- cluster:
    certificate-authority-data: LS0tLS1CRUdJTiBDRVJUSUZJQ0FURS0tLS0tCk1JSURCVENDQWUyZ0F3SUJBZ0lJVjRGb1RHLzVVQXd3RFFZSktvWklodmNOQVFFTEJRQXdGVEVUTUJFR0ExVUUKQXhNS2EzVmla>
    server: https://10.0.1.46:6443
  name: kubernetes
contexts:
- context:
    cluster: kubernetes
    user: kubernetes-admin
  name: kubernetes-admin@kubernetes
current-context: kubernetes-admin@kubernetes
kind: Config
preferences: {}
users:
- name: kubernetes-admin
  user:
    client-certificate-data: LS0tLS1CRUdJTiBDRVJUSUZJQ0FURS0tLS0tCk1JSURLVENDQWhHZ0F3SUJBZ0lJRDVzcEx6eDdaOE13RFFZSktvWklodmNOQVFFTEJRQXdGVEVUTUJFR0ExVUUKQXhNS2EzVmlaWEp>
    client-key-data: LS0tLS1CRUdJTiBSU0EgUFJJVkFURSBLRVktLS0tLQpNSUlFcEFJQkFBS0NBUUVBemtiU2dSNHREQUFyemF0OVhPblRaaXNsWUo5RjdnVXYwbXhtc3FnbCtlMy9hQkJvCkUrSGxOYTd5WS9HcW5>



^G Help        ^O Write Out   ^W Where Is    ^K Cut         ^T Execute     ^C Location    M-U Undo       M-A Set Mark   M-] To Bracket M-Q Previous
^X Exit        ^R Read File   ^\ Replace     ^U Paste       ^J Justify     ^/ Go To Line  M-E Redo       M-6 Copy       ^Q Where Was   M-W Next
```

i-034c60ff7a59f29ab (jenkins-controller)



```
ubuntu@ip-10-0-1-98:~/ansible$ sudo chown -R jenkins:jenkins /var/lib/jenkins/.kube
sudo chmod 600 /var/lib/jenkins/.kube/config
sudo systemctl restart jenkins
ubuntu@ip-10-0-1-98:~/ansible$
```

i-034c60ff7a59f29ab (jenkins-controller)



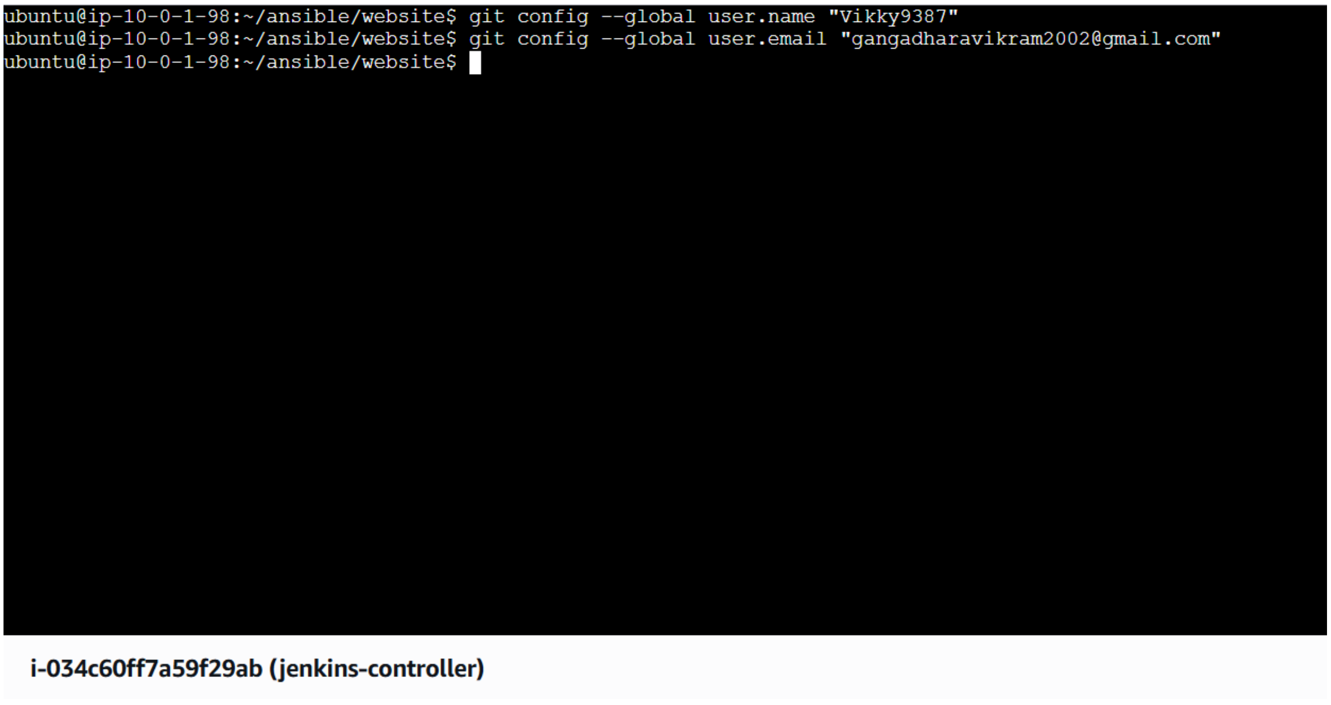
```
ubuntu@ip-10-0-1-98:~/ansible$ sudo -u jenkins kubectl get nodes
NAME            STATUS   ROLES           AGE     VERSION
ip-10-0-1-19    Ready    <none>          5m10s   v1.29.15
ip-10-0-1-46    Ready    control-plane   8m10s   v1.29.15
ip-10-0-1-93    Ready    <none>          4m48s   v1.29.15
ubuntu@ip-10-0-1-98:~/ansible$
```

i-034c60ff7a59f29ab (jenkins-controller)

STEP 2 — GitHub Integration

Configure Git

```
git config --global user.name "Vikky9387"
git config --global user.email gangadharavikram2002@gmail.com
```



```
ubuntu@ip-10-0-1-98:~/ansible/website$ git config --global user.name "Vikky9387"
ubuntu@ip-10-0-1-98:~/ansible/website$ git config --global user.email "gangadharavikram2002@gmail.com"
ubuntu@ip-10-0-1-98:~/ansible/website$
```

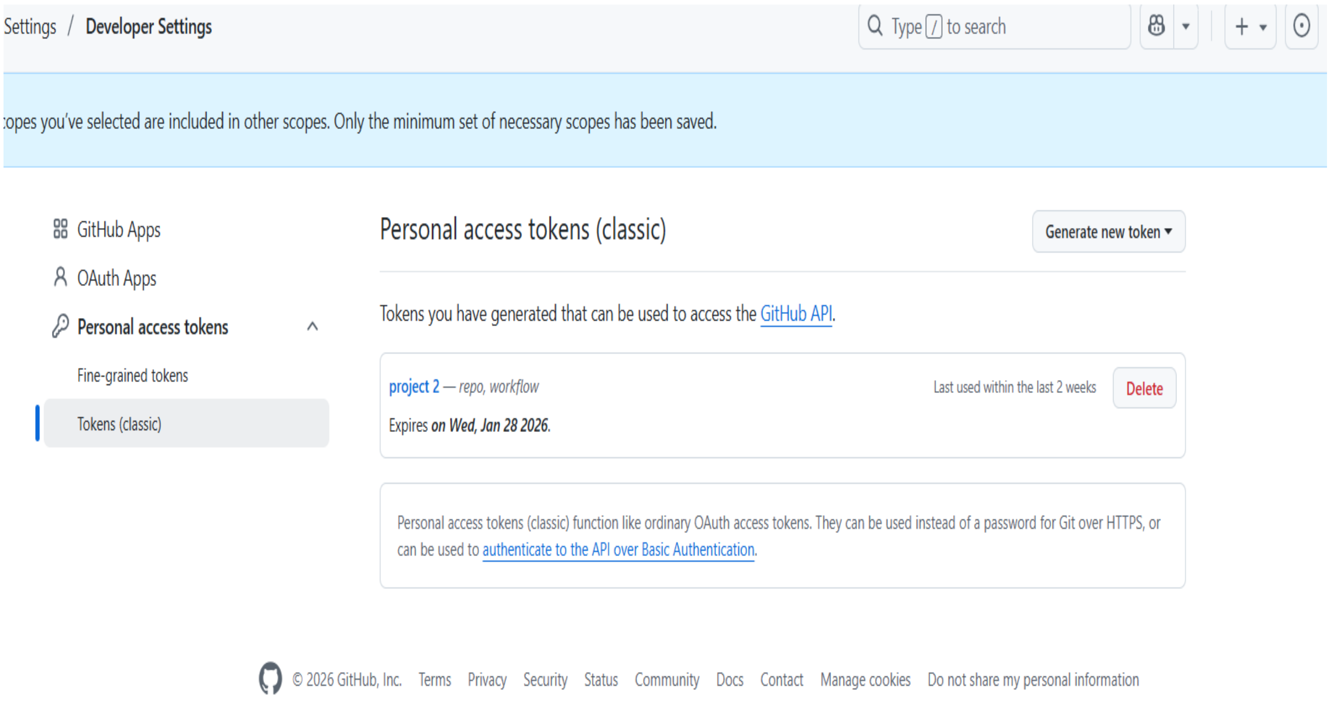
i-034c60ff7a59f29ab (jenkins-controller)

GitHub Authentication

GitHub → Settings → Developer Settings → Personal Access Token

Scopes: repo, workflow

Use the token as the Git password when pushing.



:opes you've selected are included in other scopes. Only the minimum set of necessary scopes has been saved.

Personal access tokens (classic)

Tokens you have generated that can be used to access the GitHub API.

project 2 — *repo, workflow* Last used within the last 2 weeks Delete

Expires *on Wed, Jan 28 2026.*

Personal access tokens (classic) function like ordinary OAuth access tokens. They can be used instead of a password for Git over HTTPS, or can be used to authenticate to the API over Basic Authentication.

© 2026 GitHub, Inc. Terms Privacy Security Status Community Docs Contact Manage cookies Do not share my personal information

---

STEP 3 - Connect Jenkins to GitHub

Part A — Prepare Your GitHub Repository

Fork the project

Open in browser:

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

Click **Fork** → select your GitHub account.

Now your repository becomes:

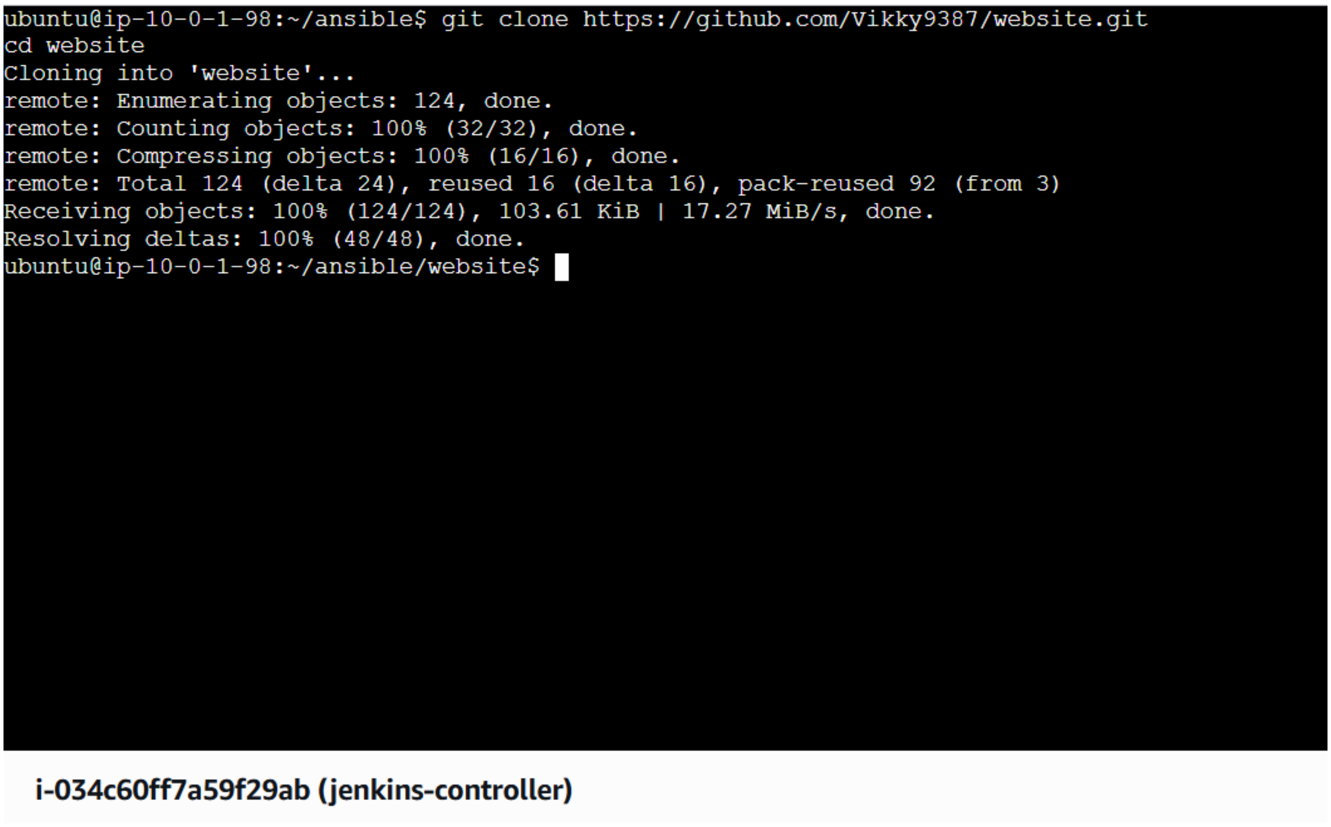
https://github.com/Vikky9387/website.git

Clone it on Jenkins server

Clone:

git clone https://github.com/Vikky9387/website.git

cd website



```
ubuntu@ip-10-0-1-98:~/ansible$ git clone https://github.com/Vikky9387/website.git
cd website
Cloning into 'website'...
remote: Enumerating objects: 124, done.
remote: Counting objects: 100% (32/32), done.
remote: Compressing objects: 100% (16/16), done.
remote: Total 124 (delta 24), reused 16 (delta 16), pack-reused 92 (from 3)
Receiving objects: 100% (124/124), 103.61 KiB | 17.27 MiB/s, done.
Resolving deltas: 100% (48/48), done.
ubuntu@ip-10-0-1-98:~/ansible/website$
```

i-034c60ff7a59f29ab (jenkins-controller)

STEP 4 — Docker Setup

nano Dockerfile

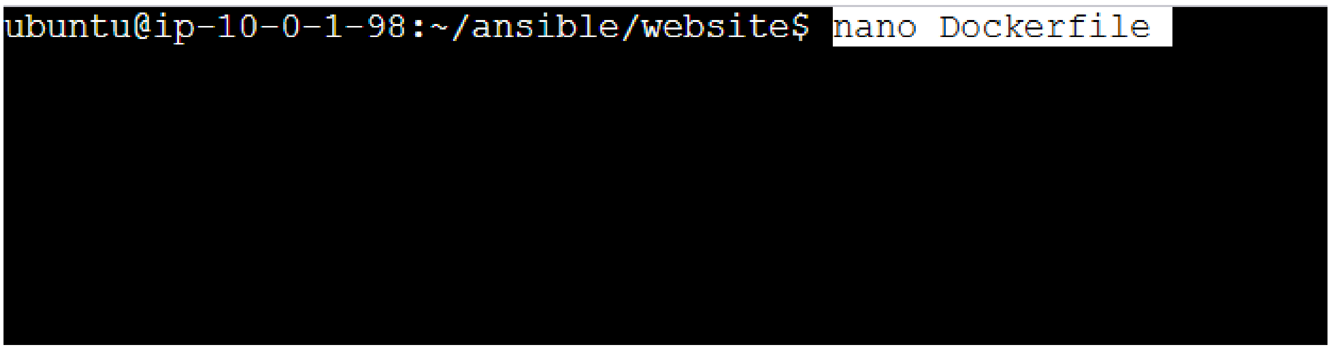
paste this:

```
FROM nginx:latest
COPY . /usr/share/nginx/html
```

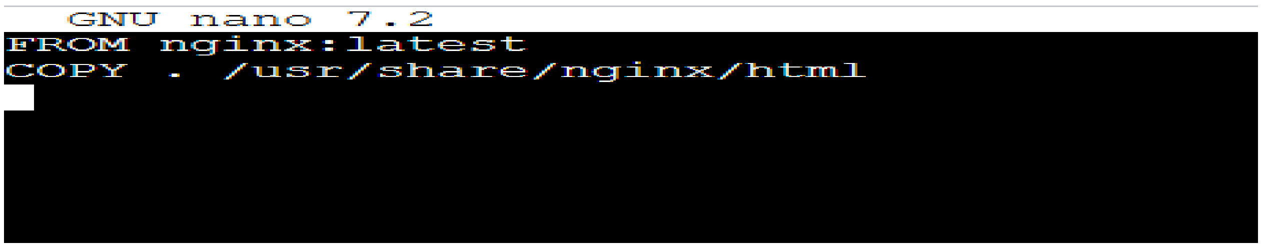
Save and exit

Commit:

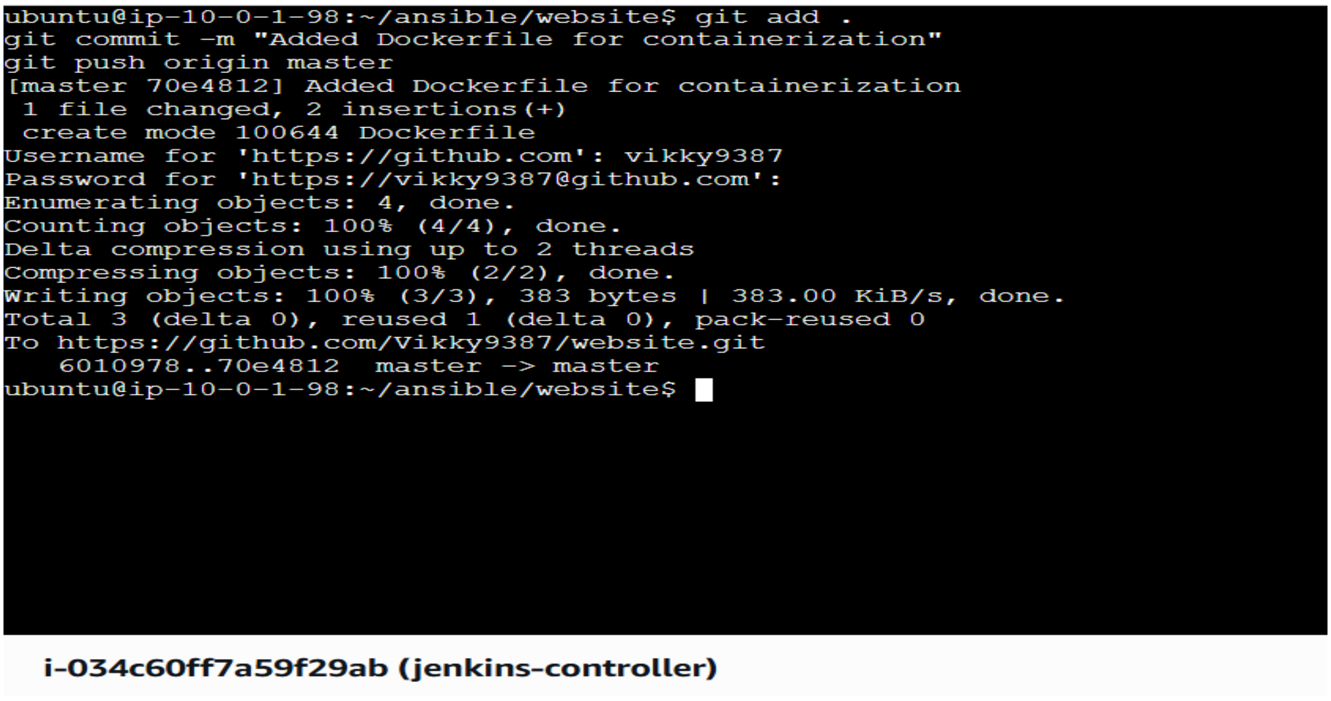
```
git add .
git commit -m "Added Dockerfile for containerization"
git push origin master
```



```
ubuntu@ip-10-0-1-98:~/ansible/website$ nano Dockerfile
```



```
  GNU nano 7.2
FROM nginx:latest
COPY . /usr/share/nginx/html
```



```
ubuntu@ip-10-0-1-98:~/ansible/website$ git add .
git commit -m "Added Dockerfile for containerization"
git push origin master
[master 70e4812] Added Dockerfile for containerization
 1 file changed, 2 insertions(+)
 create mode 100644 Dockerfile
Username for 'https://github.com': vikky9387
Password for 'https://vikky9387@github.com':
Enumerating objects: 4, done.
Counting objects: 100% (4/4), done.
Delta compression using up to 2 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 383 bytes | 383.00 KiB/s, done.
Total 3 (delta 0), reused 1 (delta 0), pack-reused 0
To https://github.com/Vikky9387/website.git
   6010978..70e4812  master -> master
ubuntu@ip-10-0-1-98:~/ansible/website$
```

i-034c60ff7a59f29ab (jenkins-controller)

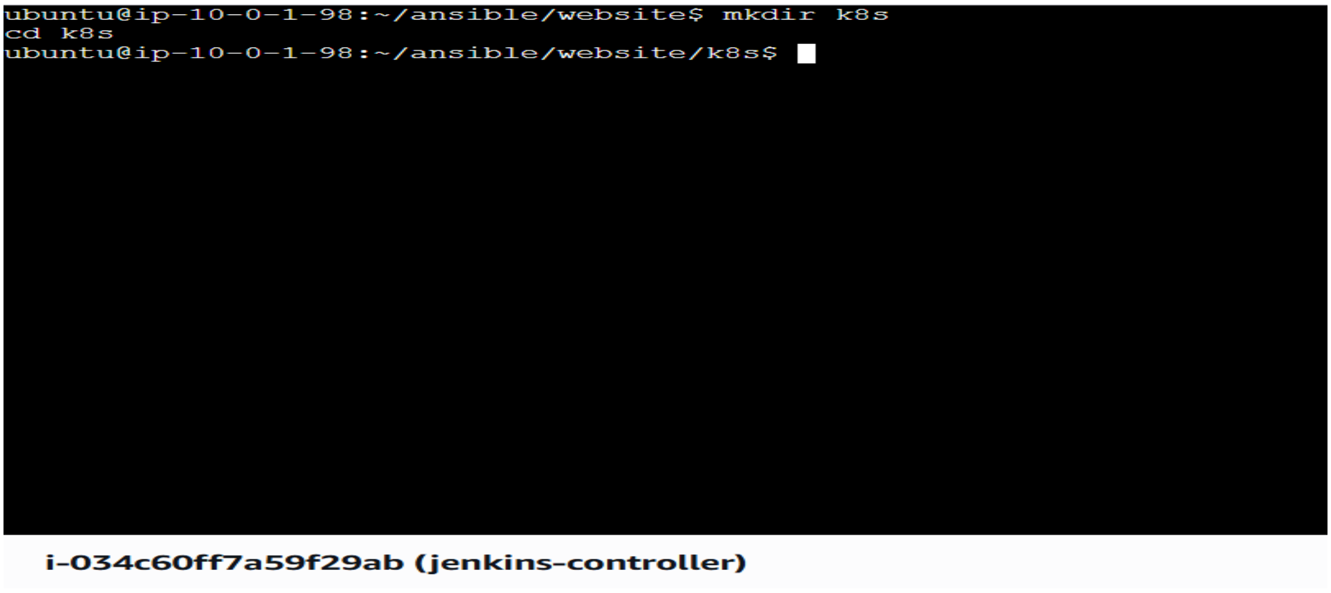
STEP 5 — Kubernetes Manifests

Go to your project directory:

```
cd ~/ansible/website
```

Create folder:

```
mkdir k8s

cd k8s
```



```
ubuntu@ip-10-0-1-98:~/ansible/website$ mkdir k8s
cd k8s
ubuntu@ip-10-0-1-98:~/ansible/website/k8s$
```

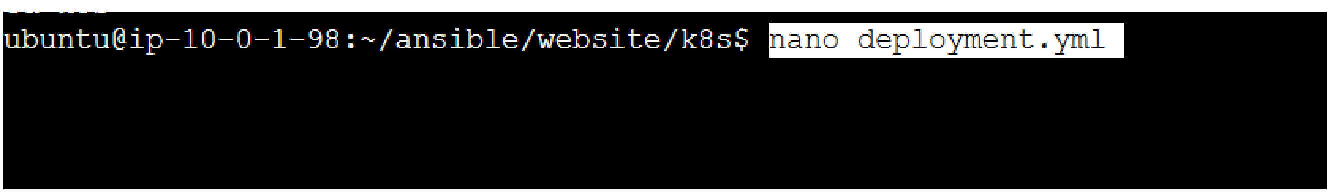
i-034c60ff7a59f29ab (jenkins-controller)

nano deployment.yml

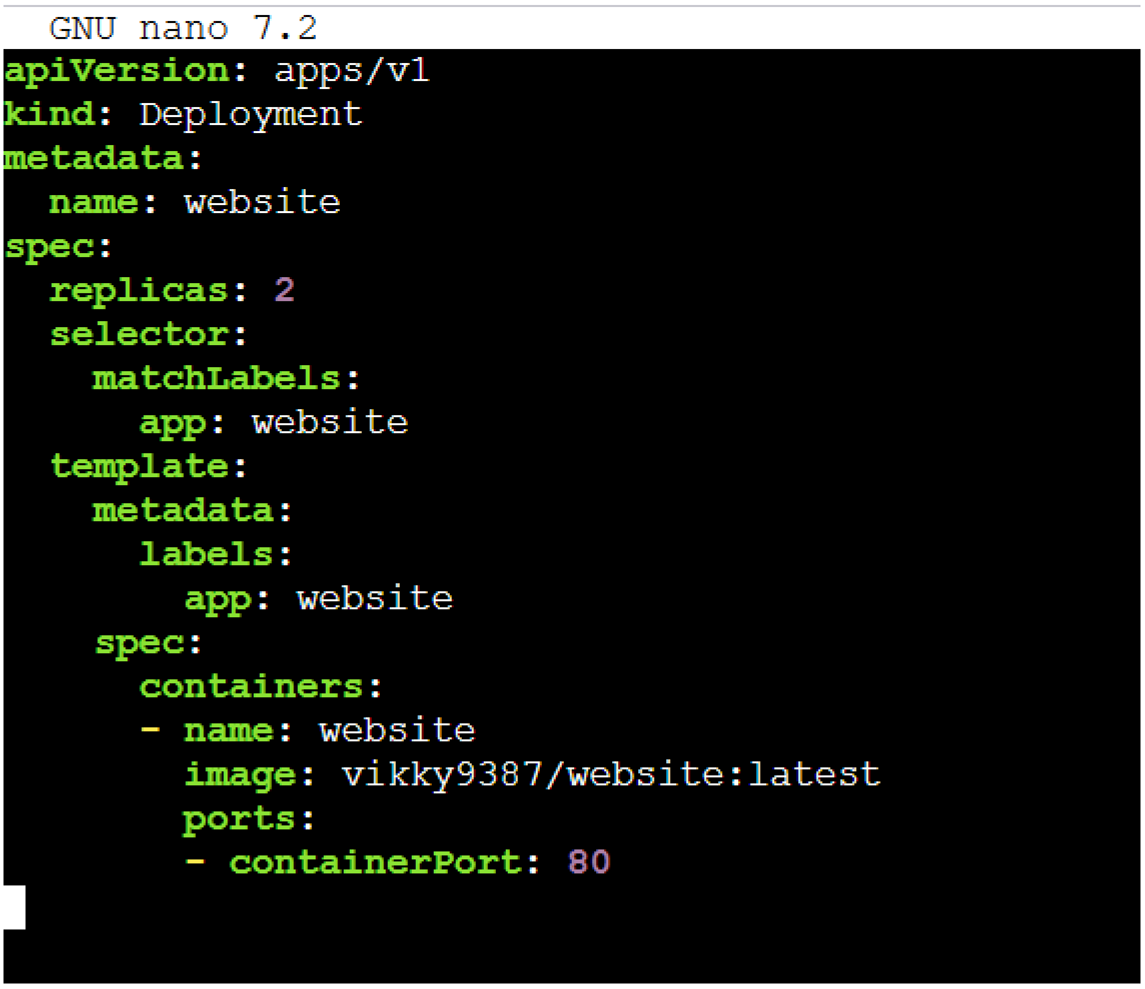
paste this:

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: website
spec:
  replicas: 2
  selector:
    matchLabels:
      app: website
  template:
    metadata:
      labels:
        app: website
    spec:
      containers:
      - name: website
        image: vikky9387/website:latest
        ports:
        - containerPort: 80
```

Save and exit



```
ubuntu@ip-10-0-1-98:~/ansible/website/k8s$ nano deployment.yml
```



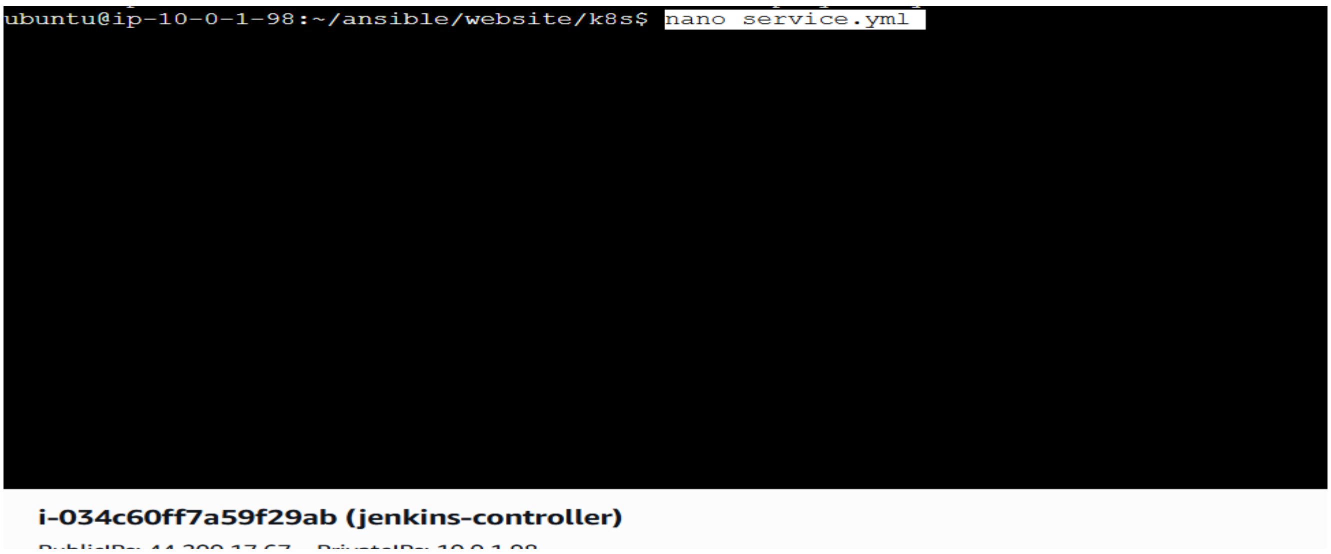
```
  GNU nano 7.2
apiVersion: apps/v1
kind: Deployment
metadata:
  name: website
spec:
  replicas: 2
  selector:
    matchLabels:
      app: website
  template:
    metadata:
      labels:
        app: website
    spec:
      containers:
      - name: website
        image: vikky9387/website:latest
        ports:
        - containerPort: 80
```

nano service.yml

paste this:

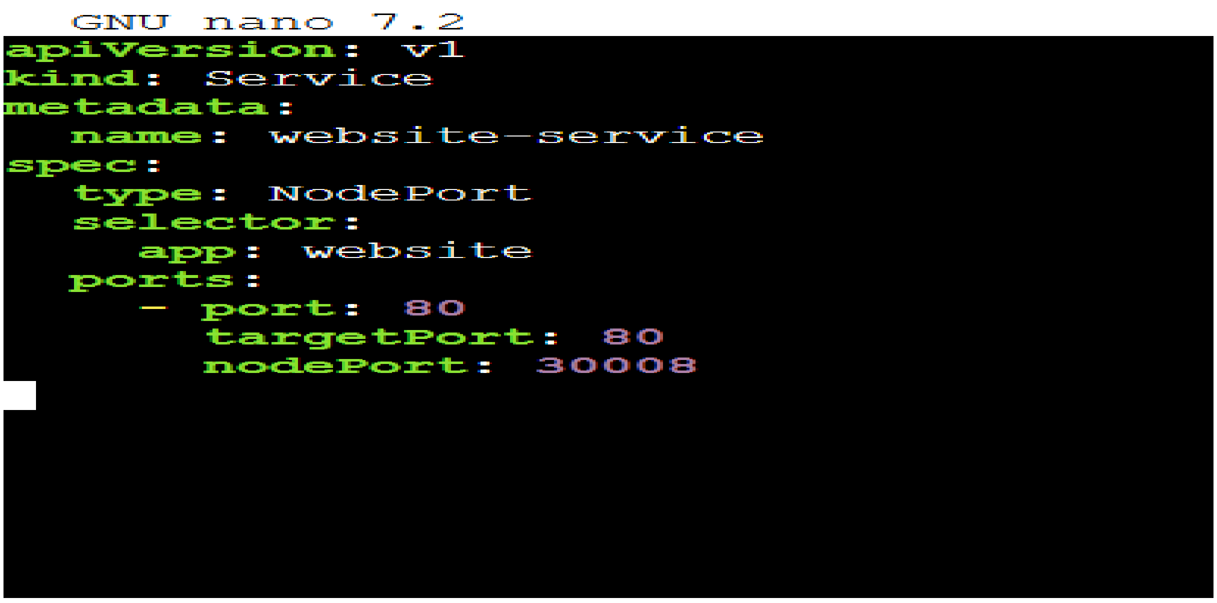
```yaml
apiVersion: v1
kind: Service
metadata:
  name: website-service
spec:
  type: NodePort
  selector:
    app: website
  ports:
    - port: 80
      targetPort: 80
      nodePort: 30008
```

save & exit



```
ubuntu@ip-10-0-1-98:~/ansible/website/k8s$ nano service.yml
```

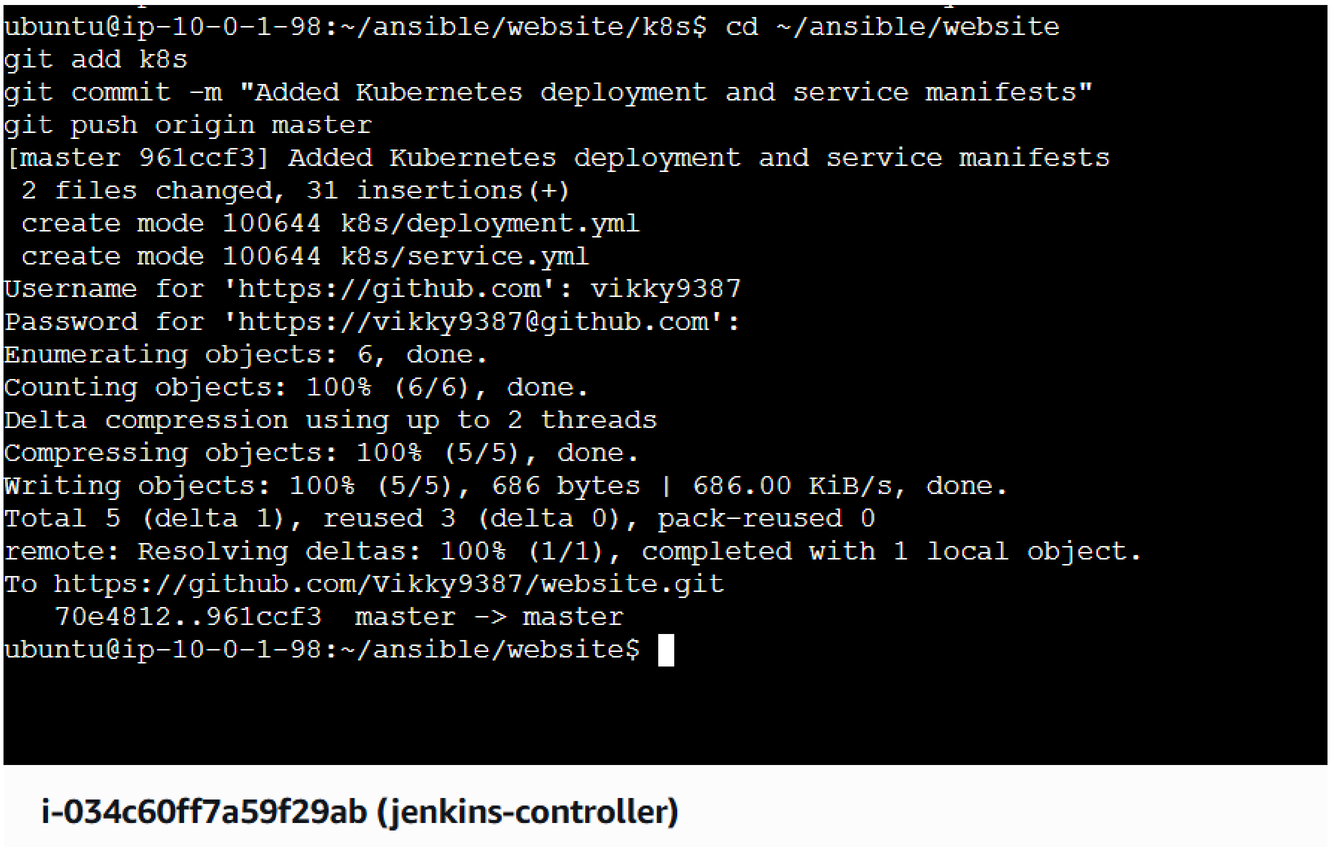
i-034c60ff7a59f29ab (jenkins-controller)



```
  GNU nano 7.2
apiVersion: v1
kind: Service
metadata:
  name: website-service
spec:
  type: NodePort
  selector:
    app: website
  ports:
    - port: 80
      targetPort: 80
      nodePort: 30008
```

Commit & Push

```
cd ~/ansible/website

git add k8s

git commit -m "Added Kubernetes deployment and service manifests"

git push origin master
```



```
ubuntu@ip-10-0-1-98:~/ansible/website/k8s$ cd ~/ansible/website
git add k8s
git commit -m "Added Kubernetes deployment and service manifests"
git push origin master
[master 961ccf3] Added Kubernetes deployment and service manifests
 2 files changed, 31 insertions(+)
 create mode 100644 k8s/deployment.yml
 create mode 100644 k8s/service.yml
Username for 'https://github.com': vikky9387
Password for 'https://vikky9387@github.com':
Enumerating objects: 6, done.
Counting objects: 100% (6/6), done.
Delta compression using up to 2 threads
Compressing objects: 100% (5/5), done.
Writing objects: 100% (5/5), 686 bytes | 686.00 KiB/s, done.
Total 5 (delta 1), reused 3 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (1/1), completed with 1 local object.
To https://github.com/Vikky9387/website.git
   70e4812..961ccf3  master -> master
ubuntu@ip-10-0-1-98:~/ansible/website$
```

i-034c60ff7a59f29ab (jenkins-controller)

Prepare Jenkins

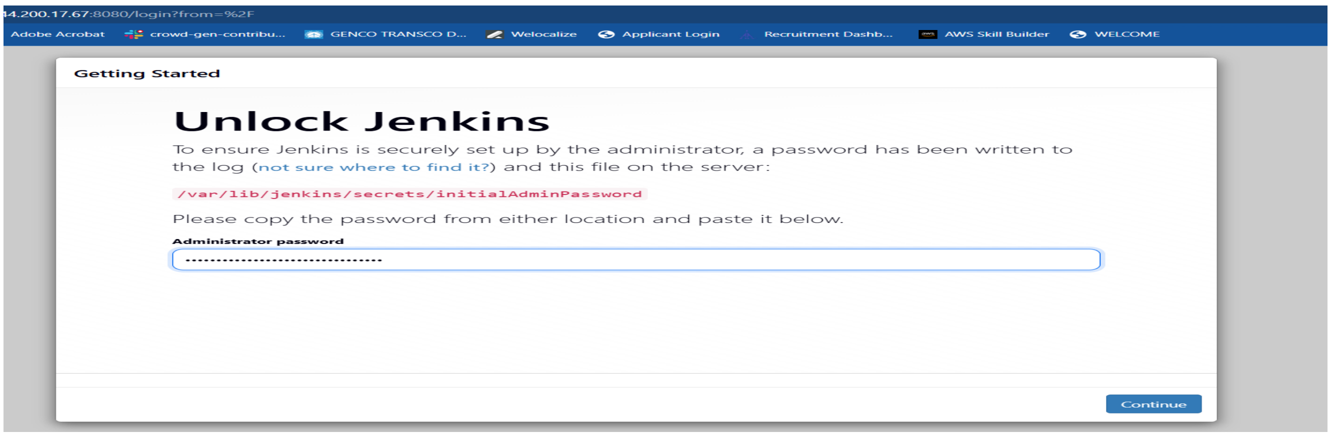
◆ Open Jenkins

http:// 44.200.17.67:8080

Unlock Jenkins:

```
sudo cat /var/lib/jenkins/secrets/initialAdminPassword
```

Install suggested plugins.



Getting Started

Unlock Jenkins

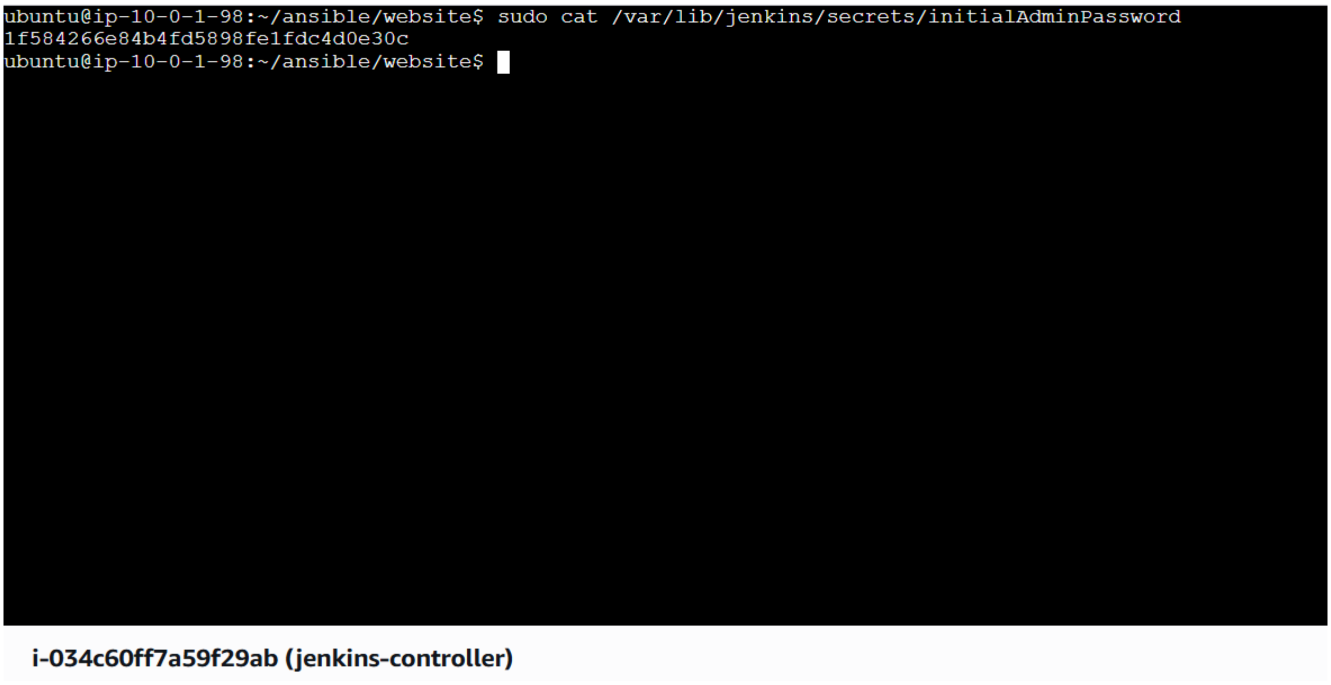
To ensure Jenkins is securely set up by the administrator, a password has been written to the log (not sure where to find it?) and this file on the server:

/var/lib/jenkins/secrets/initialAdminPassword

Please copy the password from either location and paste it below.

Administrator password

································



```
ubuntu@ip-10-0-1-98:~/ansible/website$ sudo cat /var/lib/jenkins/secrets/initialAdminPassword
1f584266e84b4fd5898fe1fdc4d0e30c
ubuntu@ip-10-0-1-98:~/ansible/website$
```

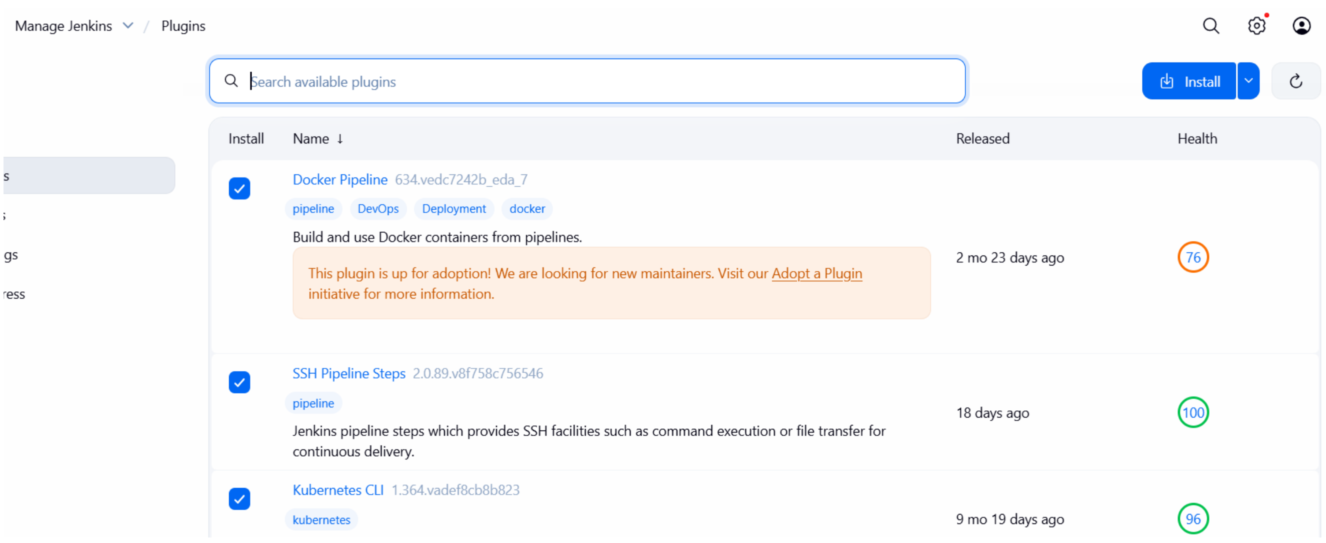
i-034c60ff7a59f29ab (jenkins-controller)

◈ Install Required Plugins

From Manage Jenkins → Plugins:

- Git
- Docker Pipeline
- Kubernetes CLI
- SSH Pipeline

Restart Jenkins.

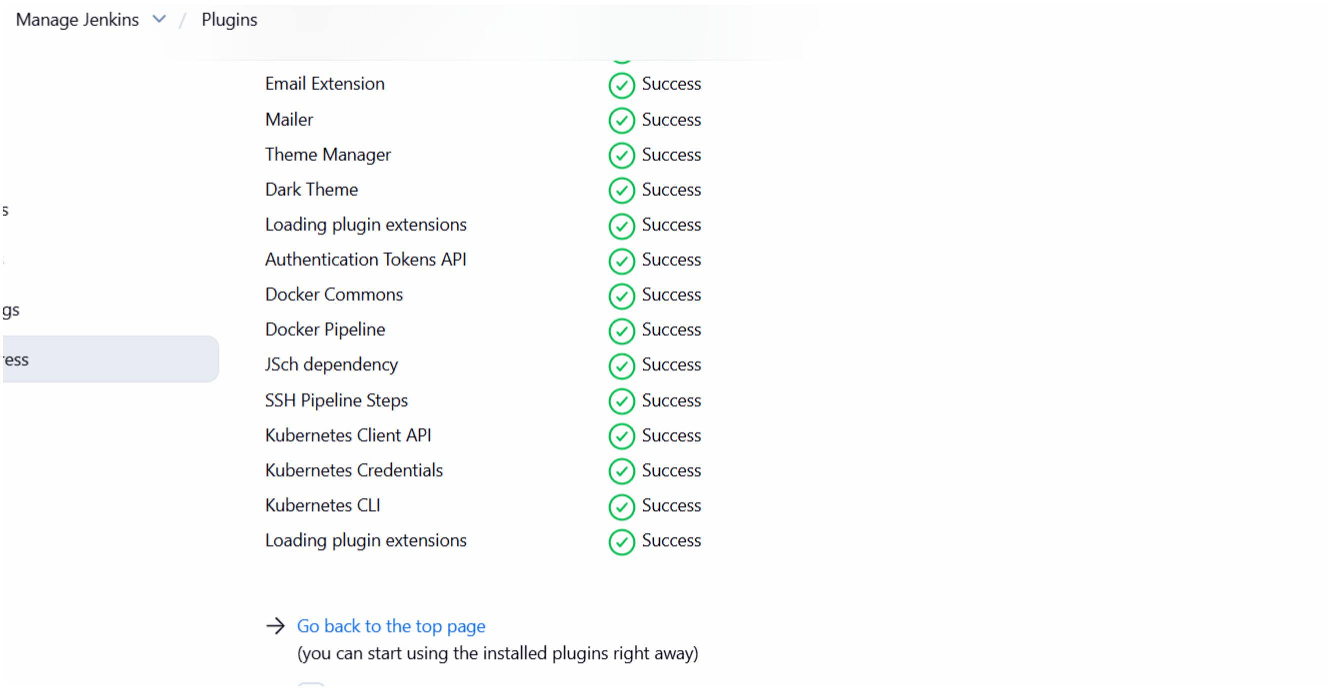


Manage Jenkins / Plugins

Search available plugins

Install

Install	Name ↓	Released	Health
✓	Docker Pipeline 634.vedc7242b_eda_7 · pipeline · DevOps · Deployment · docker · Build and use Docker containers from pipelines. This plugin is up for adoption! We are looking for new maintainers. Visit our Adopt a Plugin initiative for more information.	2 mo 23 days ago	76
✓	SSH Pipeline Steps 2.0.89.v8f758c756546 · pipeline · Jenkins pipeline steps which provides SSH facilities such as command execution or file transfer for continuous delivery.	18 days ago	100
✓	Kubernetes CLI 1.364.vadef8cb8b823 · kubernetes	9 mo 19 days ago	96


Manage Jenkins / Plugins

s

gs

ess

Email Extension	✓ Success
Mailer	✓ Success
Theme Manager	✓ Success
Dark Theme	✓ Success
Loading plugin extensions	✓ Success
Authentication Tokens API	✓ Success
Docker Commons	✓ Success
Docker Pipeline	✓ Success
JSch dependency	✓ Success
SSH Pipeline Steps	✓ Success
Kubernetes Client API	✓ Success
Kubernetes Credentials	✓ Success
Kubernetes CLI	✓ Success
Loading plugin extensions	✓ Success

→ Go back to the top page
(you can start using the installed plugins right away)

STEP 6 — Jenkins Credentials

Generate Docker Hub Access Token

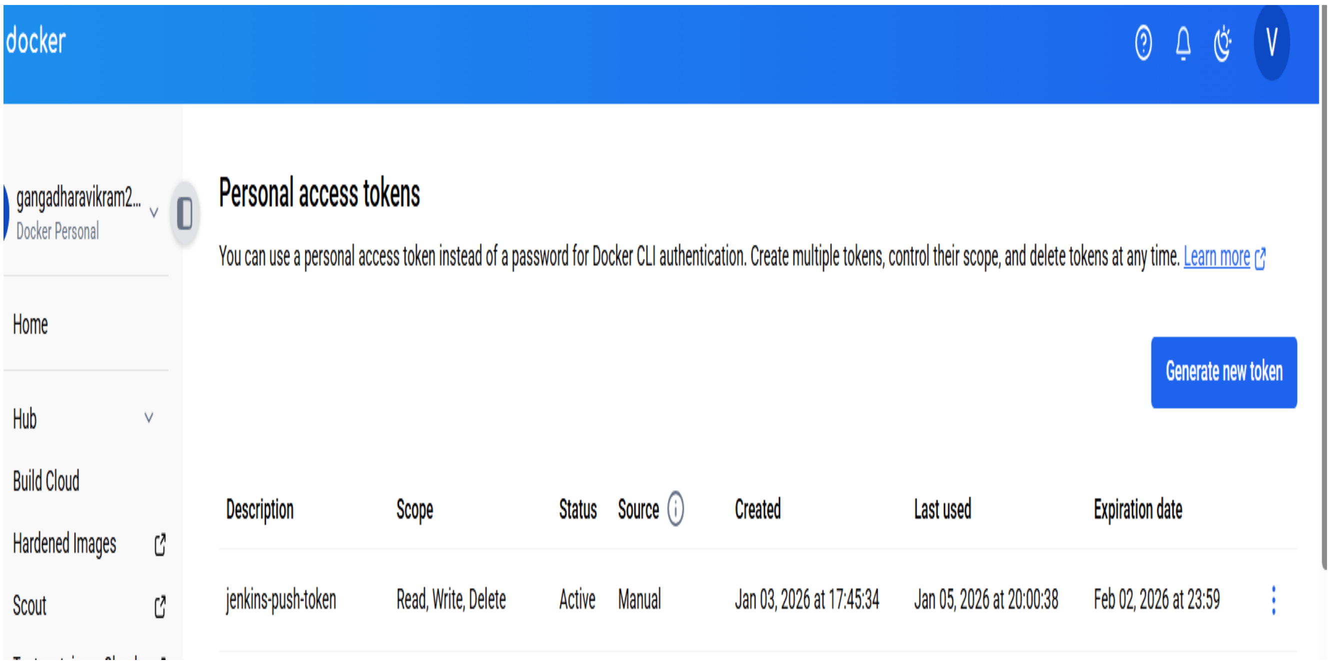
Open browser → go to:

https://hub.docker.com/settings/security

Click New Access Token

Name: Jenkins-token

Copy the token.



Personal access tokens

You can use a personal access token instead of a password for Docker CLI authentication. Create multiple tokens, control their scope, and delete tokens at any time. Learn more

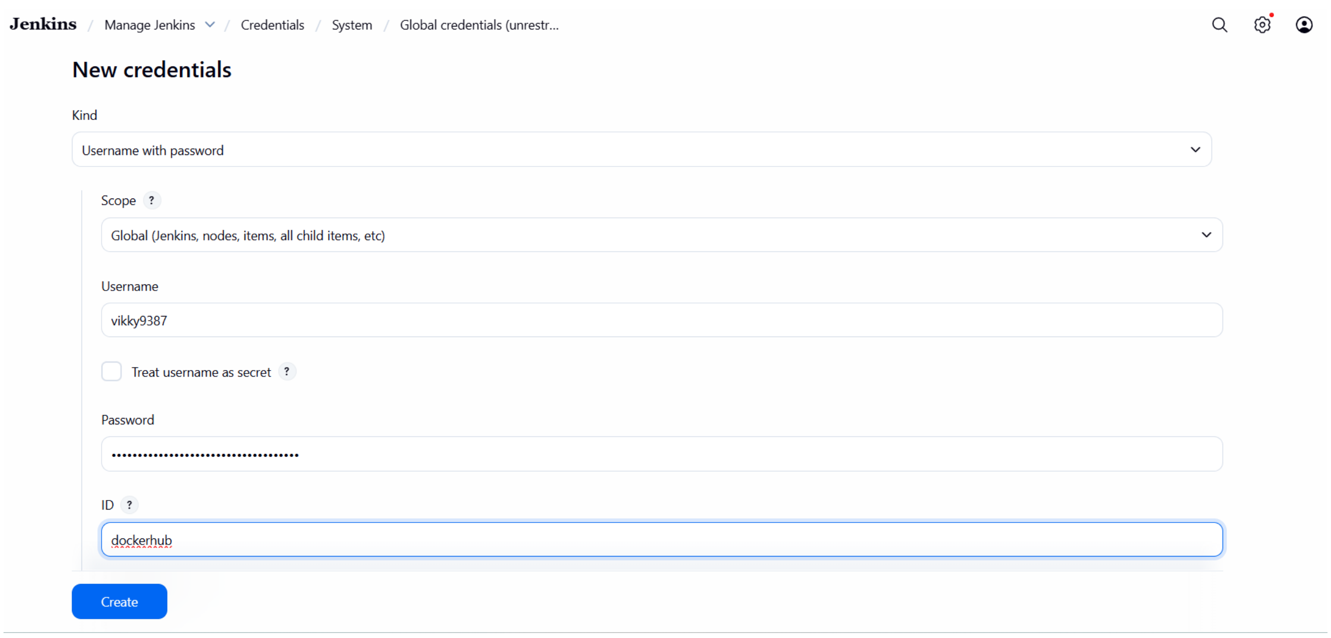
Generate new token

Description	Scope	Status	Source	Created	Last used	Expiration date
jenkins-push-token	Read, Write, Delete	Active	Manual	Jan 03, 2026 at 17:45:34	Jan 05, 2026 at 20:00:38	Feb 02, 2026 at 23:59

◈ **Dashboard → Manage Jenkins → Credentials → System → Global → Add Credentials**

Field	**Value**
Kind	Username with password
ID	dockerhub
Username	*your Docker Hub username*
Password	*your Docker Hub token*

Save



Jenkins / Manage Jenkins / Credentials / System / Global credentials (unrestr...

New credentials

Kind: Username with password

Scope: Global (Jenkins, nodes, items, all child items, etc)

Username: vikky9387

☐ Treat username as secret

Password: ••••••••••••••••••••••••••••••••••••

ID: dockerhub

Create

Add GitHub credentials

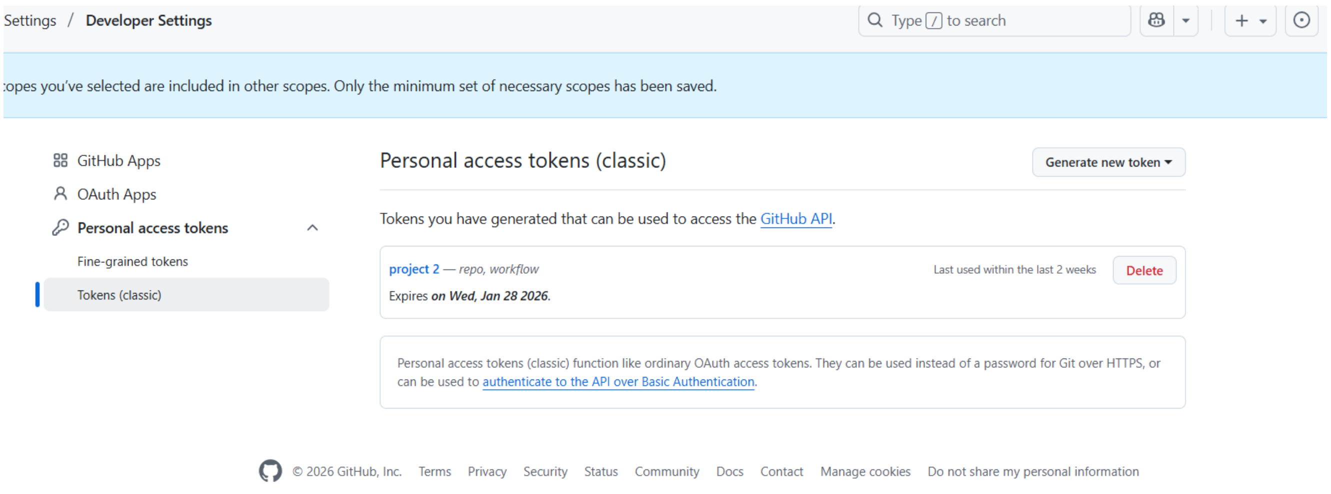
Create GitHub token:

GitHub → Settings → Developer settings → Personal access tokens → Tokens (classic) → Generate

Select:

- repo
- workflow

Copy token.



Settings / Developer Settings

:opes you've selected are included in other scopes. Only the minimum set of necessary scopes has been saved.

Personal access tokens (classic)

Tokens you have generated that can be used to access the GitHub API.

project 2 — *repo, workflow* Last used within the last 2 weeks

Expires *on Wed, Jan 28 2026*.

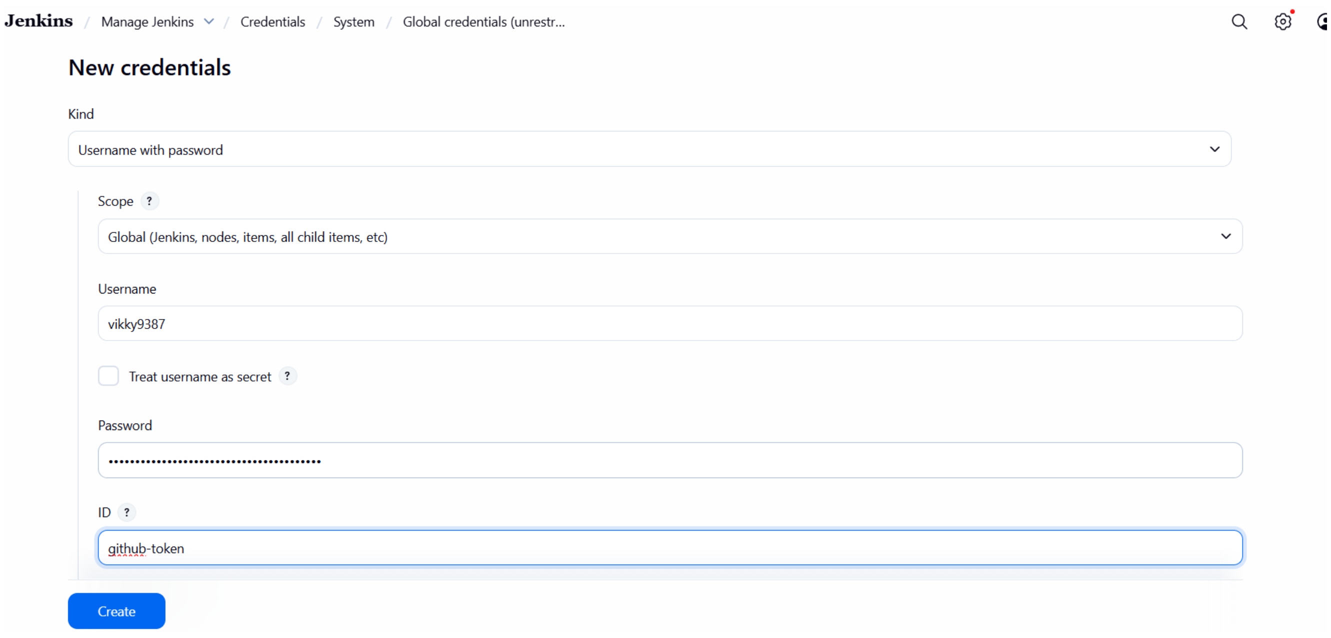
Personal access tokens (classic) function like ordinary OAuth access tokens. They can be used instead of a password for Git over HTTPS, or can be used to authenticate to the API over Basic Authentication.

Type / to search
GitHub Apps
OAuth Apps
Personal access tokens
Fine-grained tokens
Tokens (classic)
Generate new token
Delete
© 2026 GitHub, Inc. Terms Privacy Security Status Community Docs Contact Manage cookies Do not share my personal information

In Jenkins:

Add new credential:

Field	**Value**
ID	github-token
Username	your GitHub username
Password	*paste GitHub token*



Jenkins / Manage Jenkins / Credentials / System / Global credentials (unrestr...

New credentials

Kind: Username with password

Scope: Global (Jenkins, nodes, items, all child items, etc)

Username: vikky9387

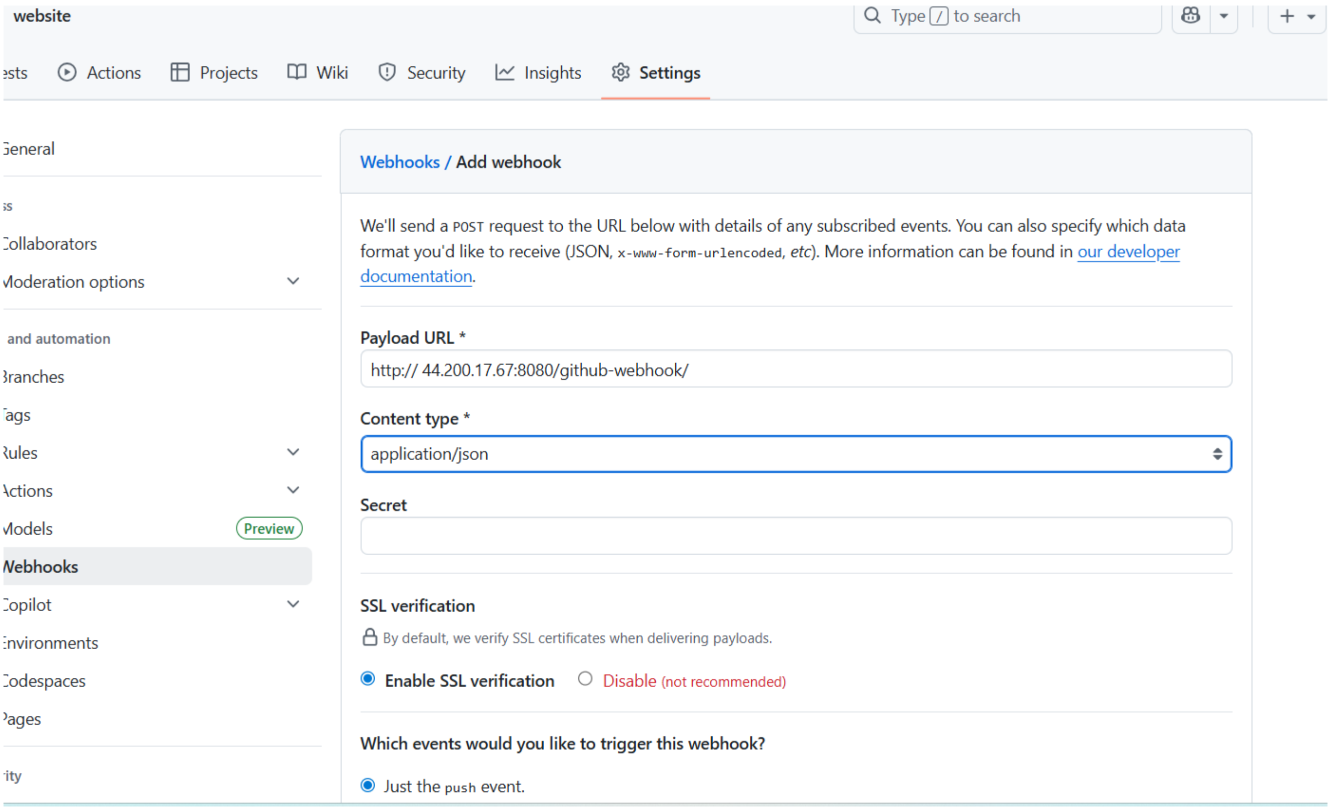
Treat username as secret

Password: ••

ID: github-token

GitHub Webhook

http://44.200.17.67:8080/github-webhook/



Webhooks / Add webhook

We'll send a POST request to the URL below with details of any subscribed events. You can also specify which data format you'd like to receive (JSON, `x-www-form-urlencoded`, *etc*). More information can be found in our developer documentation.

Payload URL *
http:// 44.200.17.67:8080/github-webhook/

Content type *
application/json

Secret

SSL verification
By default, we verify SSL certificates when delivering payloads.
[✓ Enable SSL verification] Disable (not recommended)

Which events would you like to trigger this webhook?
[✓ Just the `push` event.]

Create Jenkins Pipeline Job

Jenkins Dashboard → **New Item**

Name:

Website-CI-CD

Type: **Pipeline**

Pipeline Configuration

Build Trigger: ☑ GitHub hook trigger for GITScm polling

Scroll to **Pipeline** section:

Definition → **Pipeline script from SCM**

SCM: **Git**

Repository URL:

https://github.com/Vikky9387/website.git

Credentials: select **github-token**

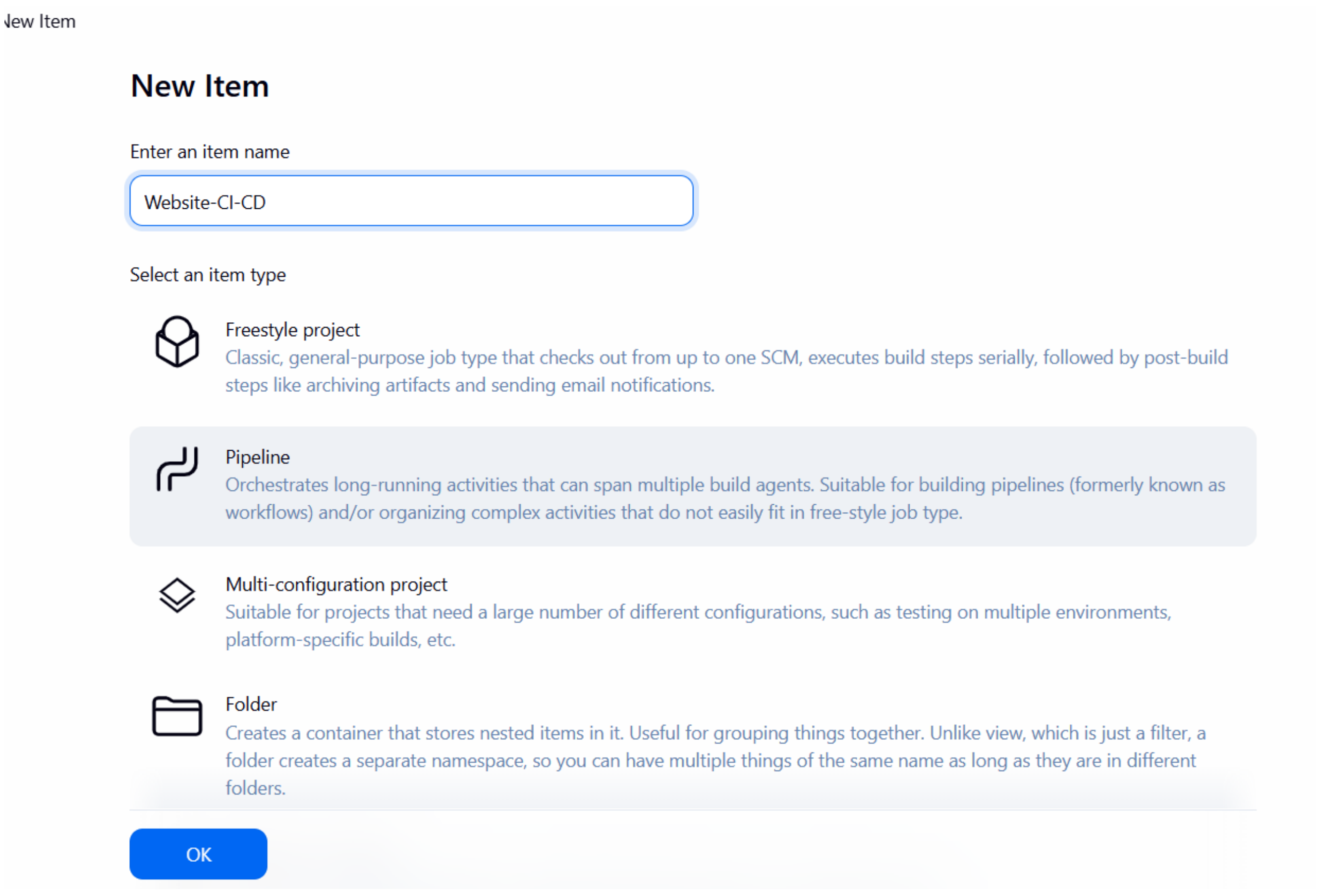
Branch:

*/master

Script Path:

Jenkinsfile

Save.


New Item

New Item

Enter an item name

Website-CI-CD

Select an item type

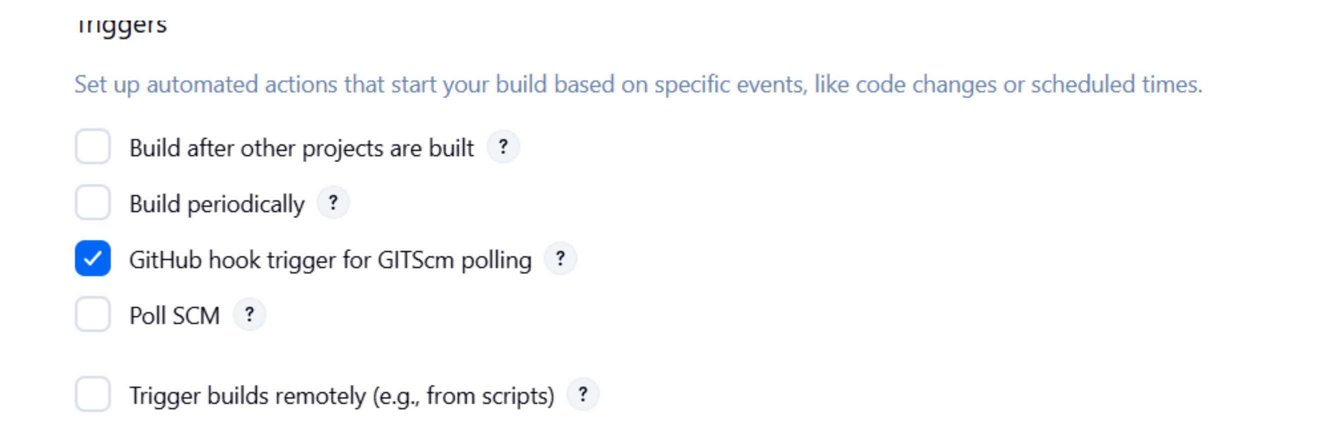
Freestyle project
Classic, general-purpose job type that checks out from up to one SCM, executes build steps serially, followed by post-build steps like archiving artifacts and sending email notifications.

Pipeline
Orchestrates long-running activities that can span multiple build agents. Suitable for building pipelines (formerly known as workflows) and/or organizing complex activities that do not easily fit in free-style job type.

Multi-configuration project
Suitable for projects that need a large number of different configurations, such as testing on multiple environments, platform-specific builds, etc.

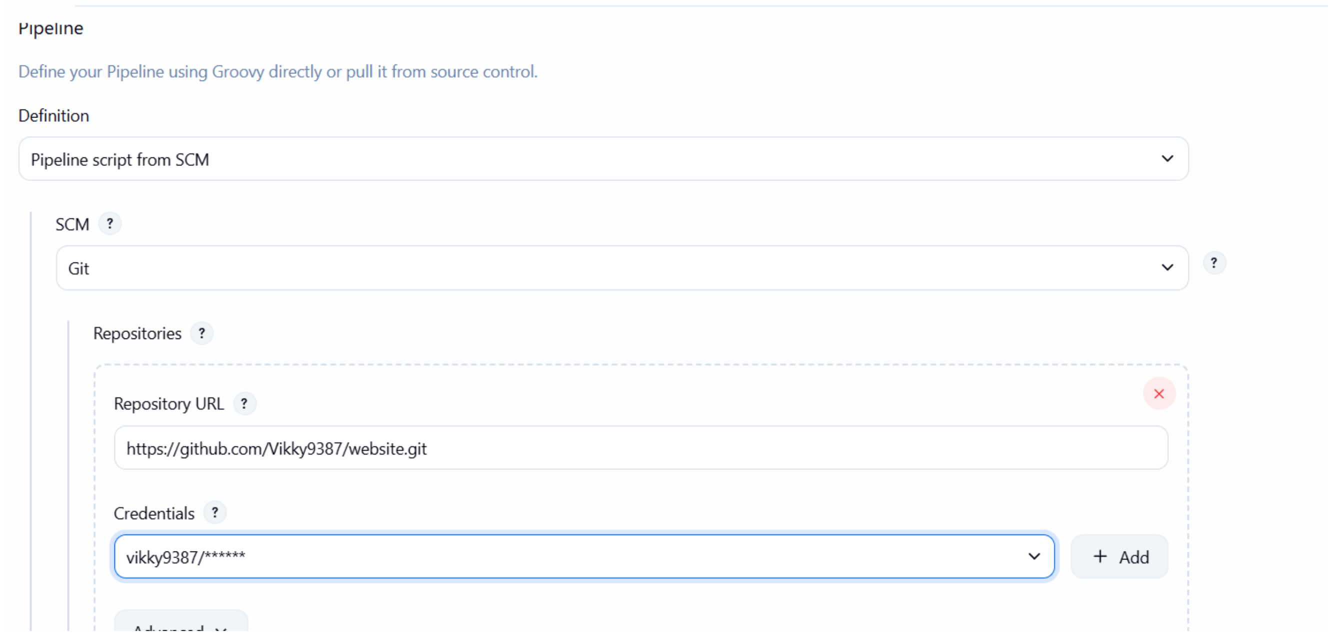
Folder
Creates a container that stores nested items in it. Useful for grouping things together. Unlike view, which is just a filter, a folder creates a separate namespace, so you can have multiple things of the same name as long as they are in different folders.

OK


Triggers

Set up automated actions that start your build based on specific events, like code changes or scheduled times.

- ☐ Build after other projects are built ?
- ☐ Build periodically ?
- ☑ GitHub hook trigger for GITScm polling ?
- ☐ Poll SCM ?
- ☐ Trigger builds remotely (e.g., from scripts) ?


Pipeline

Define your Pipeline using Groovy directly or pull it from source control.

Definition

Pipeline script from SCM

SCM ?

Git

Repositories ?

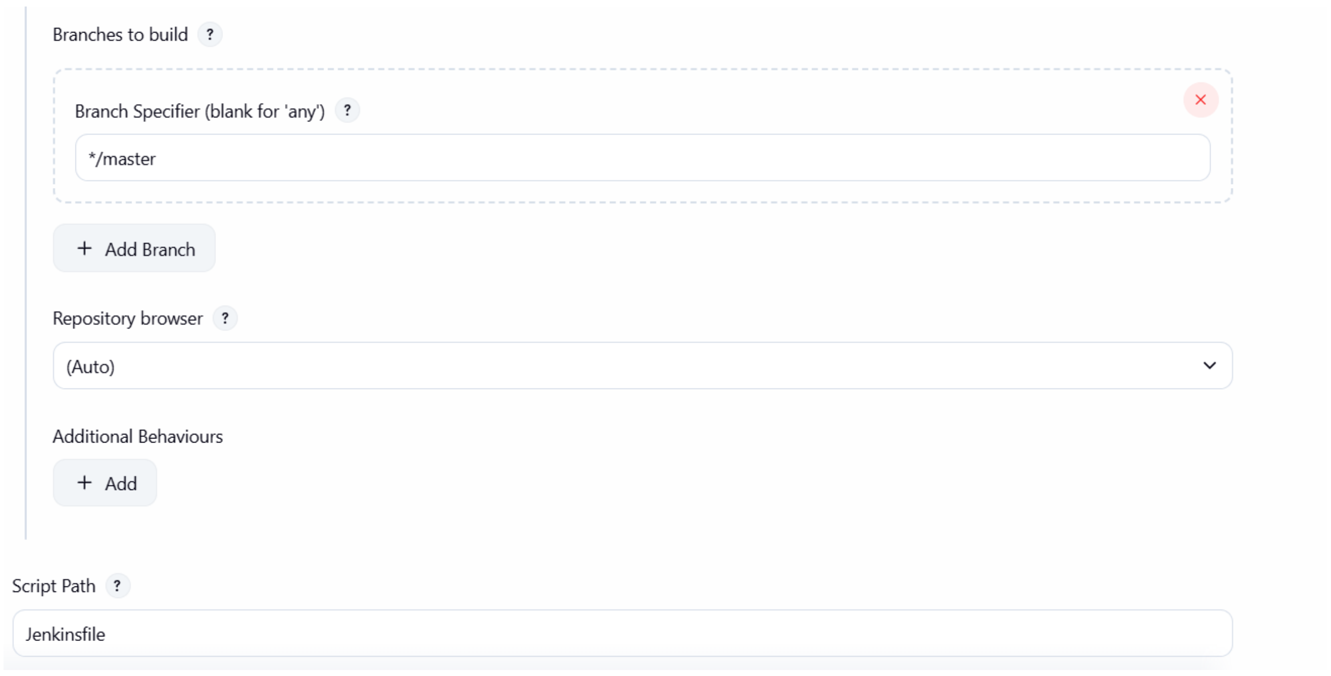
Repository URL ?

https://github.com/Vikky9387/website.git

Credentials ?

vikky9387/******

+ Add


Branches to build

Branch Specifier (blank for 'any')

*/master

Add Branch

Repository browser

(Auto)

Additional Behaviours

Add

Script Path

Jenkinsfile

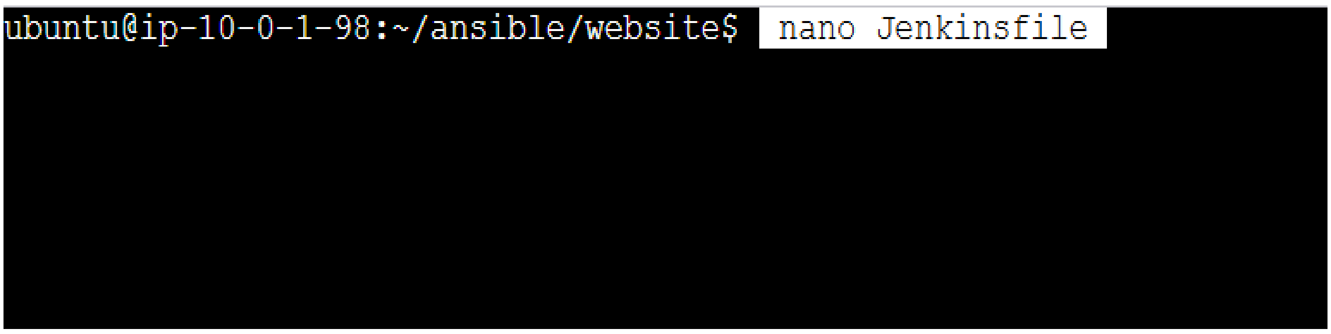
STEP 7 —Add Jenkinsfile to Repository

nano Jenkinsfile

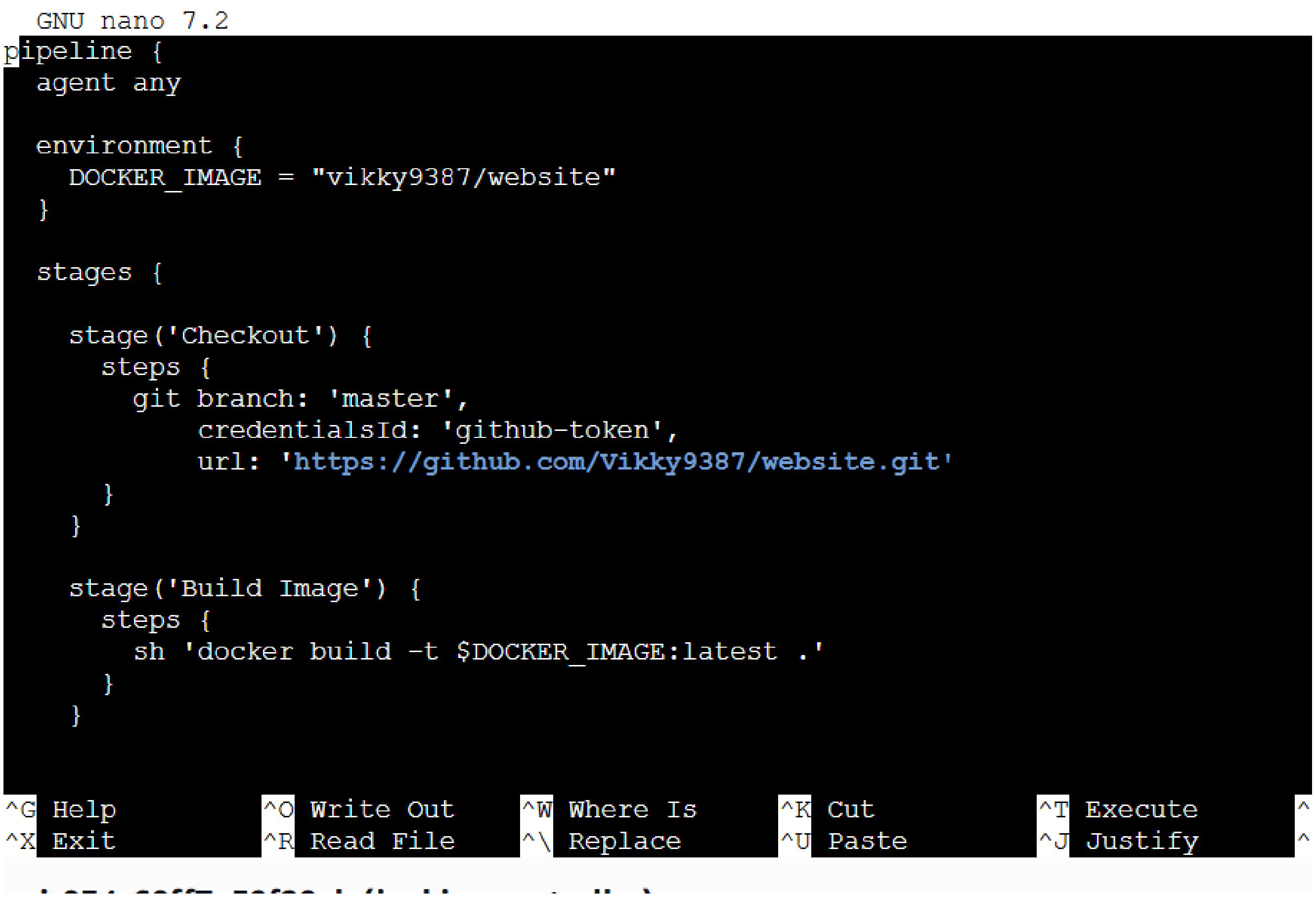
paste this:

```
pipeline {
 agent any

 environment {
  DOCKER_IMAGE = "vikky9387/website"
 }

 stages {

  stage('Checkout') {
   steps {
    git branch: 'master',
        credentialsId: 'github-token',
        url: 'https://github.com/Vikky9387/website.git'
   }
  }

  stage('Build Image') {
      steps {
        sh 'docker build -t $DOCKER_IMAGE:latest .'
      }
    }

    stage('Push Image') {
      steps {
        withCredentials([usernamePassword(credentialsId: 'dockerhub', usernameVariable: 'USER', 
passwordVariable: 'PASS')]) {
          sh '''
            echo $PASS | docker login -u $USER --password-stdin
            docker push $DOCKER_IMAGE:latest
          '''
        }
      }
    }

    stage('Deploy to Kubernetes') {
      steps {
        sh '''
          kubectl apply -f k8s/deployment.yml
          kubectl apply -f k8s/service.yml
        '''
      }
    }
  }
}
```

Save and exit



```
ubuntu@ip-10-0-1-98:~/ansible/website$ nano Jenkinsfile
```



```
  GNU nano 7.2
pipeline {
  agent any

  environment {
    DOCKER_IMAGE = "vikky9387/website"
  }

  stages {

    stage('Checkout') {
      steps {
        git branch: 'master',
            credentialsId: 'github-token',
            url: 'https://github.com/Vikky9387/website.git'
      }
    }

    stage('Build Image') {
      steps {
        sh 'docker build -t $DOCKER_IMAGE:latest .'
      }
    }

^G Help          ^O Write Out     ^W Where Is      ^K Cut           ^T Execute       ^
^X Exit          ^R Read File     ^\ Replace       ^U Paste         ^J Justify       ^
```

Commit & push.

git add .

git commit -m "Added Jenkins pipeline"

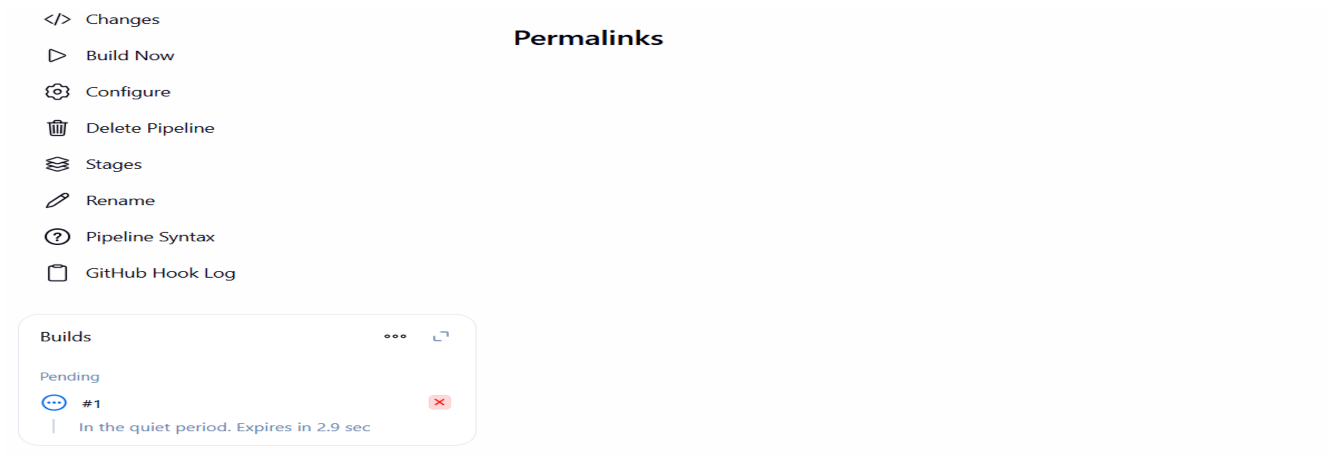
git push origin master

STEP 8 — Run Pipeline

In Jenkins:

Open Website-CI-CD → build triggers automatically

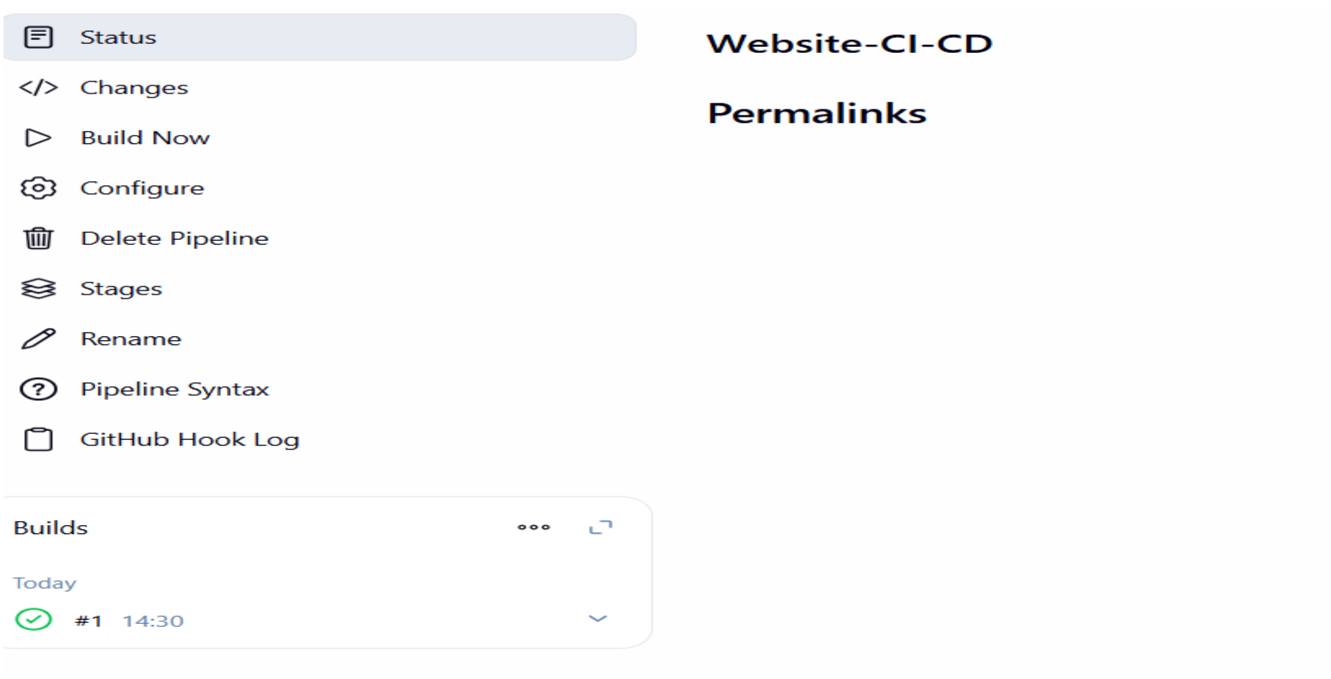
Watch console output.



Pending

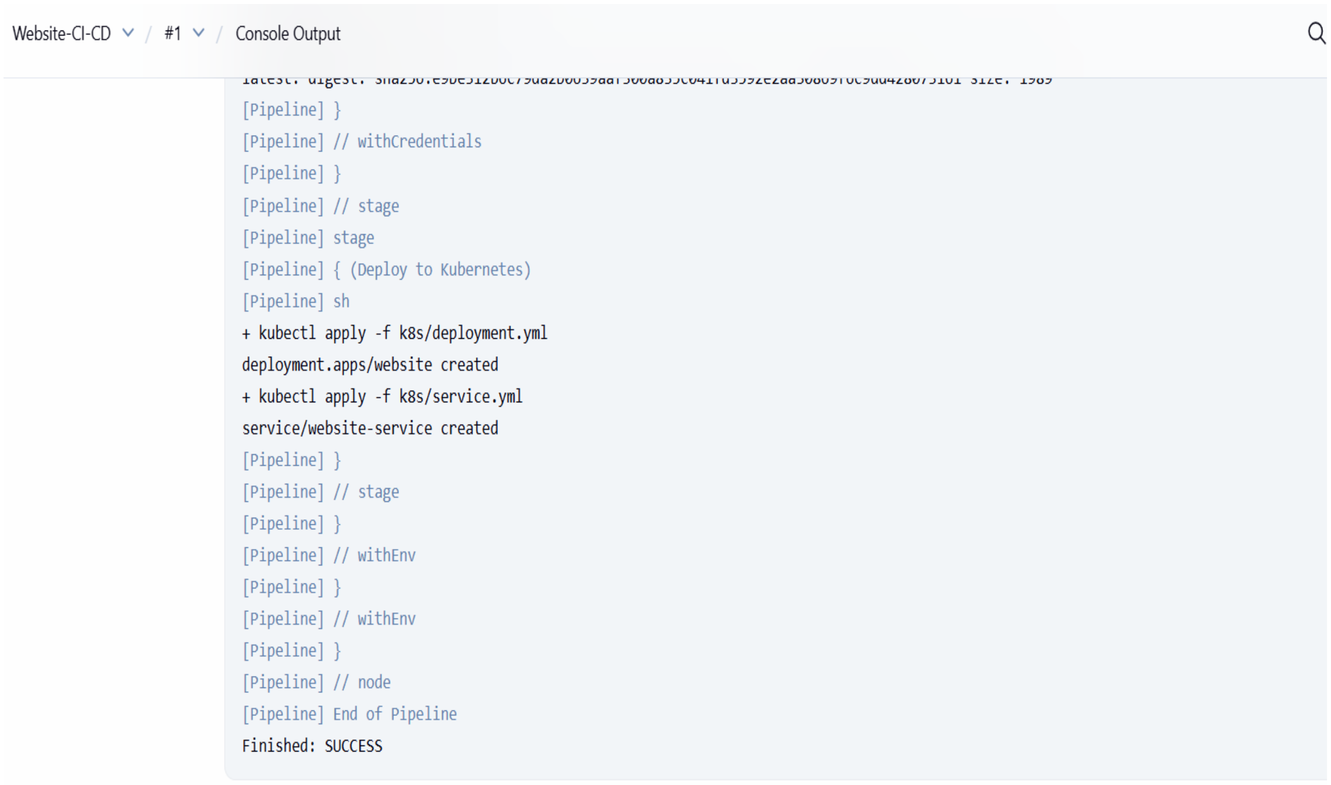
#1

In the quiet period. Expires in 2.9 sec

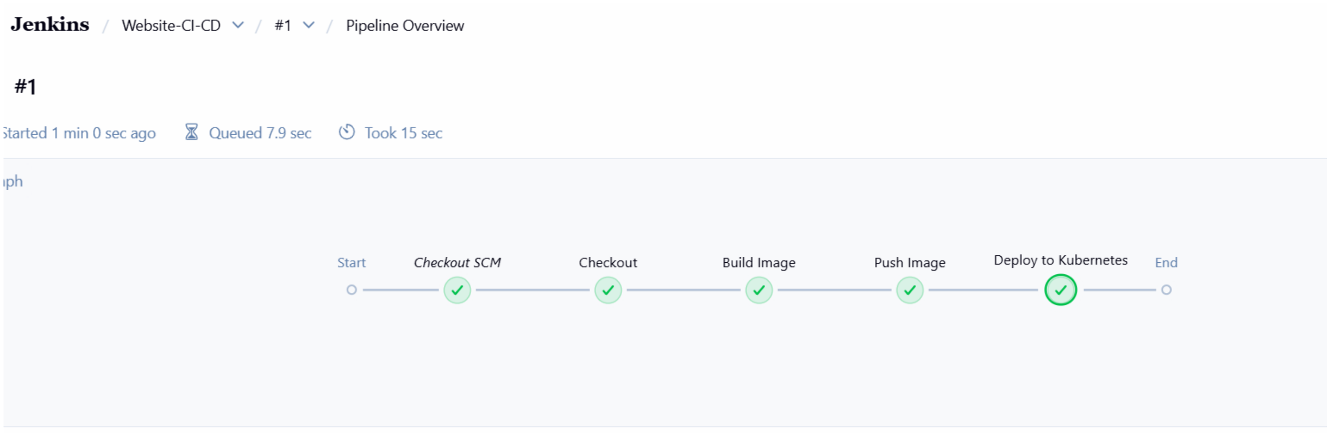


Website-CI-CD

Permalinks



```
latest: digest: sha256:e9be312b0c79da2b6039aa1300a835c041fd5592e2aa5080910c9dd42807310l size: 1989
[Pipeline] }
[Pipeline] // withCredentials
[Pipeline] }
[Pipeline] // stage
[Pipeline] stage
[Pipeline] { (Deploy to Kubernetes)
[Pipeline] sh
+ kubectl apply -f k8s/deployment.yml
deployment.apps/website created
+ kubectl apply -f k8s/service.yml
service/website-service created
[Pipeline] }
[Pipeline] // stage
[Pipeline] }
[Pipeline] // withEnv
[Pipeline] }
[Pipeline] // withEnv
[Pipeline] }
[Pipeline] // node
[Pipeline] End of Pipeline
Finished: SUCCESS
```

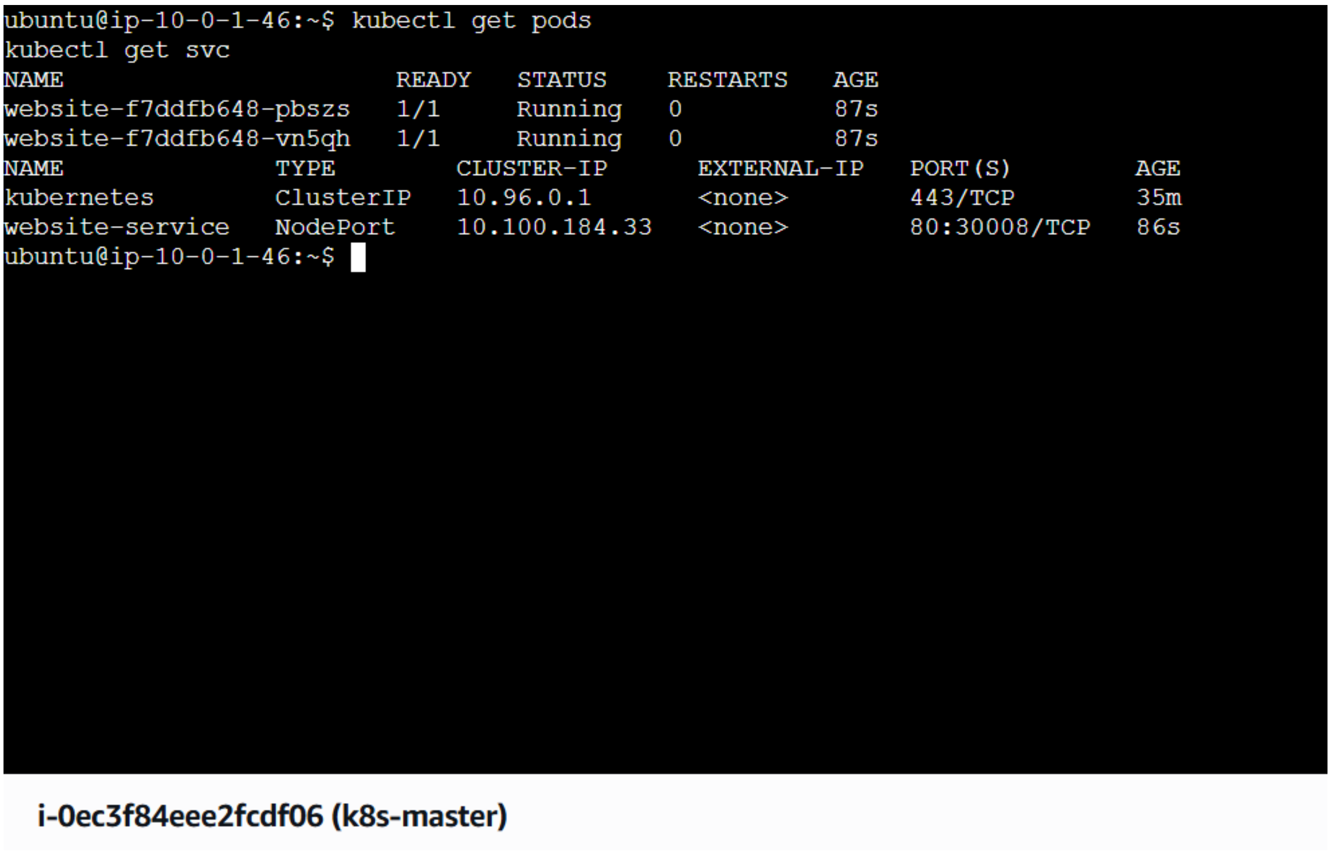


Verification

On your k8s-master:

kubectl get pods

kubectl get svc

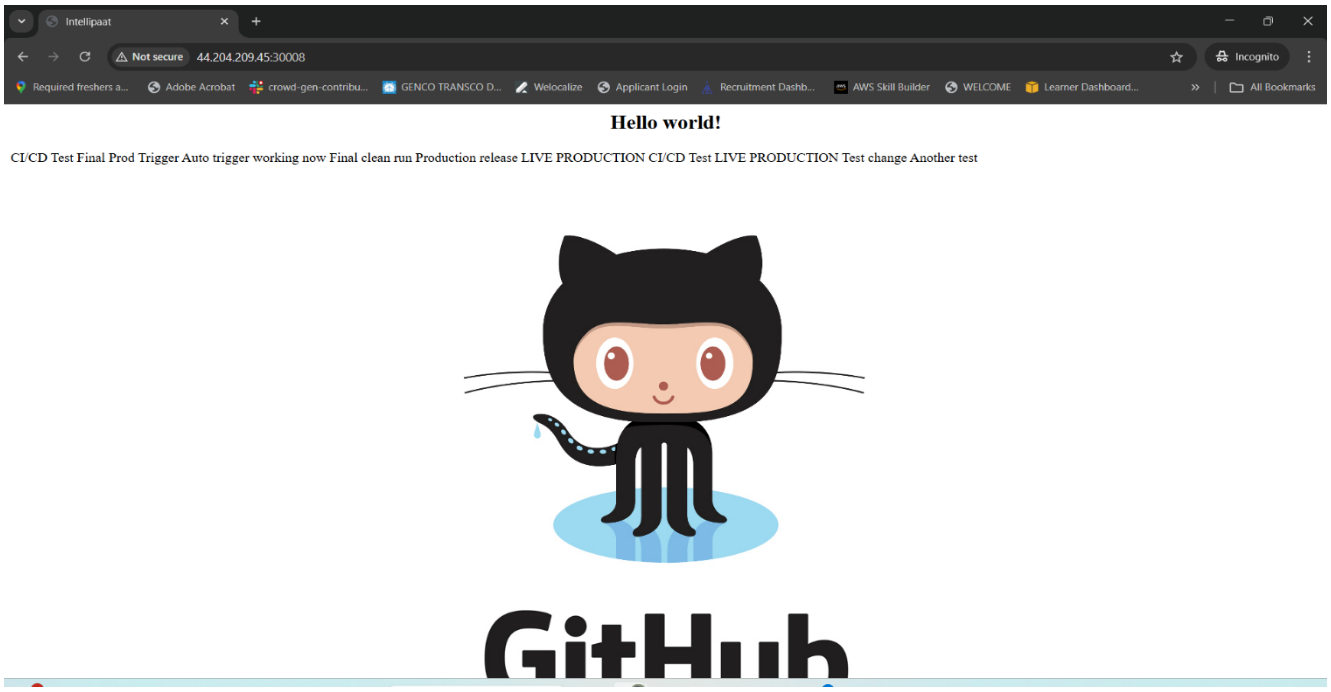


```
ubuntu@ip-10-0-1-46:~$ kubectl get pods
kubectl get svc
NAME                        READY   STATUS    RESTARTS   AGE
website-f7ddfb648-pbszs     1/1     Running   0          87s
website-f7ddfb648-vn5qh     1/1     Running   0          87s
NAME               TYPE        CLUSTER-IP      EXTERNAL-IP   PORT(S)        AGE
kubernetes         ClusterIP   10.96.0.1       <none>        443/TCP        35m
website-service    NodePort    10.100.184.33   <none>        80:30008/TCP   86s
ubuntu@ip-10-0-1-46:~$
```

i-0ec3f84eee2fcdf06 (k8s-master)

Then open in browser:

http://44.204.209.45:30008



Hello world!

CI/CD Test Final Prod Trigger Auto trigger working now Final clean run Production release LIVE PRODUCTION CI/CD Test LIVE PRODUCTION Test change Another test

Final Result

✓ Fully automated CI/CD pipeline

✓ Zero manual deployment

✓ Docker + Kubernetes production workflow

✓ Meets all DevOps lifecycle requirements

Project Implemented By:

Vikram Gangadhara.