

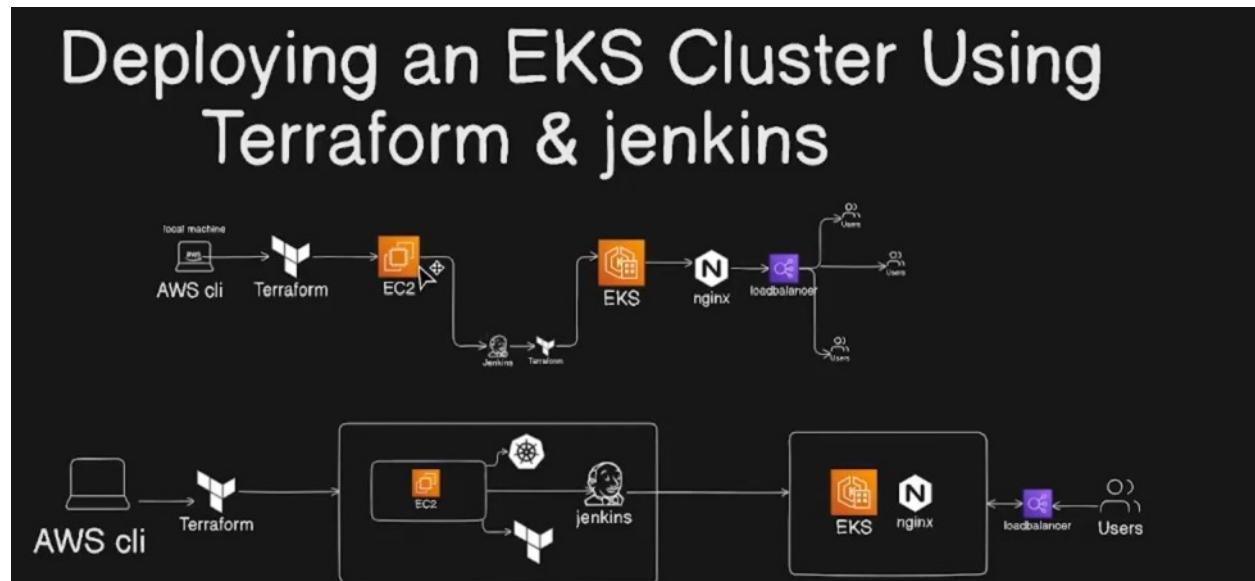
In world of cloud infrastructure and DevOps automation! seamless deployment of an EKS (Amazon Elastic Kubernetes Service) cluster on AWS using Terraform.

Automate the deployment of an Nginx application onto this EKS cluster, all orchestrated by a Jenkins CI/CD pipeline. 1. Set up your AWS environment for EKS cluster deployment.

2. Utilize Terraform to define infrastructure as code, configuring the EKS cluster effortlessly.
3. Automate the deployment of an Nginx application onto the EKS cluster using Kubernetes manifests.
4. Configure Jenkins CI/CD pipeline to automate the entire process, from code commit to deployment.

Terraform, Kubernetes, and Jenkins to automate the deployment of EKS clusters and applications, streamlining DevOps workflows and increasing efficiency.

Architecture

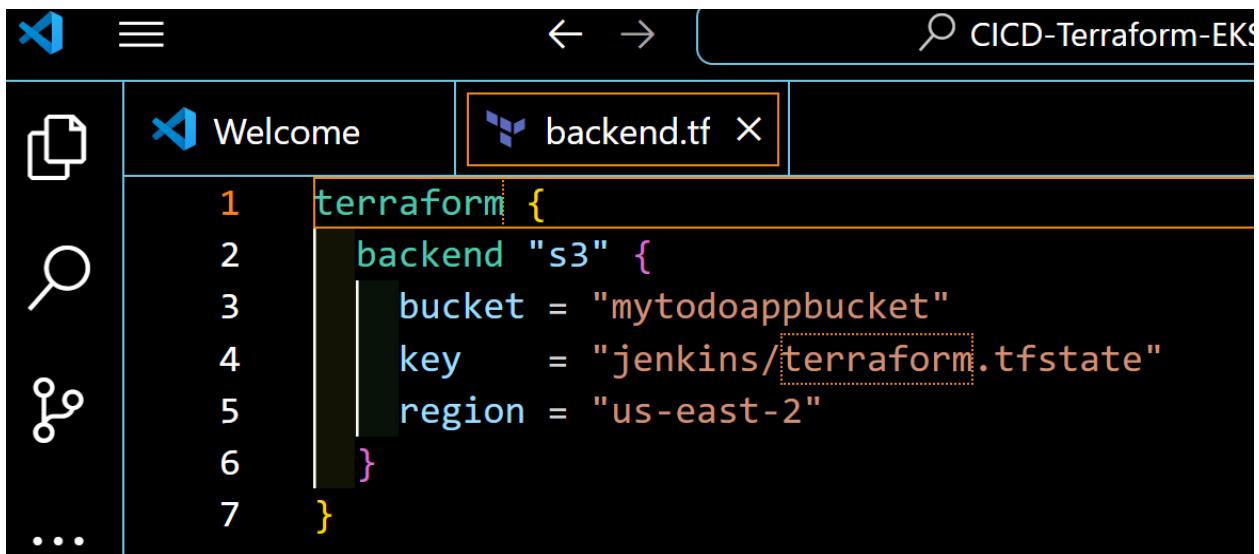


Pre-Requisites

1. Terraform
2. AWS Account
3. IAM User access Key
4. AWS CLI

Part 1 – Creating AWS EC2 + deploy Jenkins on it using Terraform

Backend



```
1 terraform {  
2   backend "s3" {  
3     bucket = "mytodoappbucket"  
4     key    = "jenkins/terraform.tfstate"  
5     region = "us-east-2"  
6   }  
7 }
```

```
# tf init  
  
$ terraform init  
Initializing the backend...  
Successfully configured the backend "s3"! Terraform will automatically  
use this backend unless the backend configuration changes.  
Initializing provider plugins...  
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.
```

```
$ terraform fmt  
backend.tf  
  
LAPTOP-JN110945 MINGW64 ~/OneDr  
$ terraform validate  
Success! The configuration is valid.
```

```
$ terraform plan  
No changes. Your infrastructure matches the configuration.  
Terraform has compared your real infrastructure against your configuration  
and found no differences, so no changes are needed.
```

go to Tf Doc

<https://registry.terraform.io/providers/hashicorp/aws/latest/docs/data-sources/ami>

The screenshot shows the Terraform Registry interface. The URL in the address bar is <https://registry.terraform.io/providers/hashicorp/aws/latest/docs/data-sources/ami>. The page title is "Data Source: aws_ami". The main content area contains a brief description: "Use this data source to get the ID of a registered AMI for use in other resources." Below this is a "Example Usage" section with the following Terraform code:

```
data "aws_ami" "example" {
  executable_users = ["self"]
  most_recent     = true
  name             = "myami-20240511"
}
```

On the right side, there is a sidebar with links: "Multi-language provider docs", "Terraform", "ON THIS PAGE" (with links to "Example Usage", "Argument Reference", "Attribute Reference", and "Timeouts"), and a "Manage Preferences" button.

This screenshot shows the same Terraform Registry page as above, but with the "Example Usage" code block expanded to show multiple filter blocks. The expanded code is:

```
data "aws_ami" "example" {
  executable_users = ["self"]
  most_recent     = true
  name             = "myami-[0-9]{3}"
  owners           = ["self"]

  filter {
    name  = "name"
    values = ["myami-*"]
  }

  filter {
    name  = "root-device-type"
    values = ["ebs"]
  }

  filter {
    name  = "virtualization-type"
    values = ["hvm"]
  }
}
```

The rest of the page layout is identical to the first screenshot, including the sidebar and footer.

```
data "aws_ami" "example" {
  most_recent = true
  owners      = ["amazon"]

  filter {
    name  = "name"
    values = ["ubuntu/images/hvm-ssd/ubuntu-jammy-22.04-amd64-server-20231207"]
  }

  filter {
    name  = "root-device-type"
    values = ["ebs"]
  }

  filter {
    name  = "virtualization-type"
    values = ["hvm"]
  }
}

data "aws_availability_zones" "azs" {}
```

Go to tf doc 4 VPC module

<https://registry.terraform.io/modules/terraform-aws-modules/vpc/aws/latest>

vpc
aws provider

Terraform module to create AWS VPC resources

Version 5.15.0 (latest)

Module Downloads

Downloads this week	383,366
Downloads this month	383,366
Downloads this year	35.2M
Downloads over all time	96.6M

Submodules Examples

Provision Instructions

Copy and paste into your Terraform configuration, insert the variables, and run `terraform init`:

```
module "vpc" {
  source = "terraform-aws-modules/terraform-aws-vpc"
  version = "5.15.0"
}
```

cp

The screenshot shows a web browser window with three tabs open: "Create EKS Cluster with VPC us", "How to Deploy EKS Cluster on", and "terraform-aws-modules/vpc/aws". The active tab is "terraform-aws-modules/vpc/aws". The URL in the address bar is <https://registry.terraform.io/modules/terraform-aws-modules/vpc/aws/latest>. The page content is titled "Usage" and contains a code snippet for creating a VPC module:

```
module "vpc" {
  source = "terraform-aws-modules/vpc/aws"

  name = "my-vpc"
  cidr = "10.0.0.0/16"

  azs      = ["eu-west-1a", "eu-west-1b", "eu-west-1c"]
  private_subnets = ["10.0.1.0/24", "10.0.2.0/24", "10.0.3.0/24"]
  public_subnets = ["10.0.101.0/24", "10.0.102.0/24", "10.0.103.0/24"]

  enable_nat_gateway = true
  enable_vpn_gateway = true

  tags = {
    Terraform = "true"
    Environment = "dev"
  }
}
```

At the bottom of the page, there is a cookie consent banner with buttons for "Manage Preferences" and "Dismiss". The browser's taskbar at the bottom shows various pinned icons and the date/time as 05-11-2024.

The screenshot shows a Windows desktop environment with a code editor window titled "CICD-Terraform-EKS". The editor is displaying a Terraform configuration file named "main.tf". The code defines a VPC module with specific parameters:

```
2 module "vpc" {
3   source = "terraform-aws-modules/vpc/aws"
4
5   name = "jenkins_vpc"
6   cidr = var.vpc_cidr
7
8   azs      = data.aws_availability_zones.azs.names
9   public_subnets = var.public_subnets
10
11  enable_dns_hostnames = true
12  map_public_ip_on_launch = true
13
14  tags = {
15    Name      = "jenkins_vpc"
16    Terraform = "true"
17    Environment = "dev"
18  }
19  public_subnet_tags = {
20    Name = "jenkins_subnet"
21 }
```

The code editor has a dark theme and includes a sidebar with various icons. The system tray at the bottom shows the date as 05-11-2024.

```
25
26 module "sg" {
27   source = "terraform-aws-modules/security-group/aws"
28
29   name      = "jenkins_sg"
30   description = "Security group for jenkins server"
31   vpc_id    = module.vpc.vpc_id
32
33
34   ingress_with_cidr_blocks = [
35     {
36       from_port  = 8080
37       to_port    = 8080
38       protocol   = "tcp"
39       description = "HTTP"
40       cidr_blocks = "0.0.0.0/0"
41     },
42     {
43       from_port  = 22
44       to_port    = 22
45       protocol   = "tcp"
46       description = "SSH"
47       cidr_blocks = "0.0.0.0/0"
48     }
49 ]
```

```
25
26 module "sg" {
27   source = "terraform-aws-modules/security-group/aws"
28
29   name      = "jenkins_sg"
30   description = "Security group for jenkins server"
31   vpc_id    = module.vpc.vpc_id
32
33
34   ingress_with_cidr_blocks = [
35     {
36       from_port  = 8080
37       to_port    = 8080
38       protocol   = "tcp"
39       description = "HTTP"
40       cidr_blocks = "0.0.0.0/0"
41     },
42     {
43       from_port  = 22
44       to_port    = 22
45       protocol   = "tcp"
46       description = "SSH"
47       cidr_blocks = "0.0.0.0/0"
48     }
49 ]
```

CICD-Terraform-EKS

```
25
26 module "sg" {
27   source = "terraform-aws-modules/security-group/aws"
28
29   name      = "jenkins_sg"
30   description = "Security group for jenkins server"
31   vpc_id    = module.vpc.vpc_id
32
33
34   ingress_with_cidr_blocks = [
35     {
36       from_port  = 8080
37       to_port    = 8080
38       protocol   = "tcp"
39       description = "HTTP"
40       cidr_blocks = "0.0.0.0/0"
41     },
42     {
43       from_port  = 22
44       to_port    = 22
45       protocol   = "tcp"
46       description = "SSH"
47       cidr_blocks = "0.0.0.0/0"
48     }
49 }
```

master AWS: profile/default

File Edit Selection View ... ← → CICD-Terraform-EKS ESP-IDF: Search Error Hint ...

Ln 2, Col 1 Spaces: 2 UTF-8 CRLF { } Terraform Go Live

Type here to search

CICD-Terraform-EKS

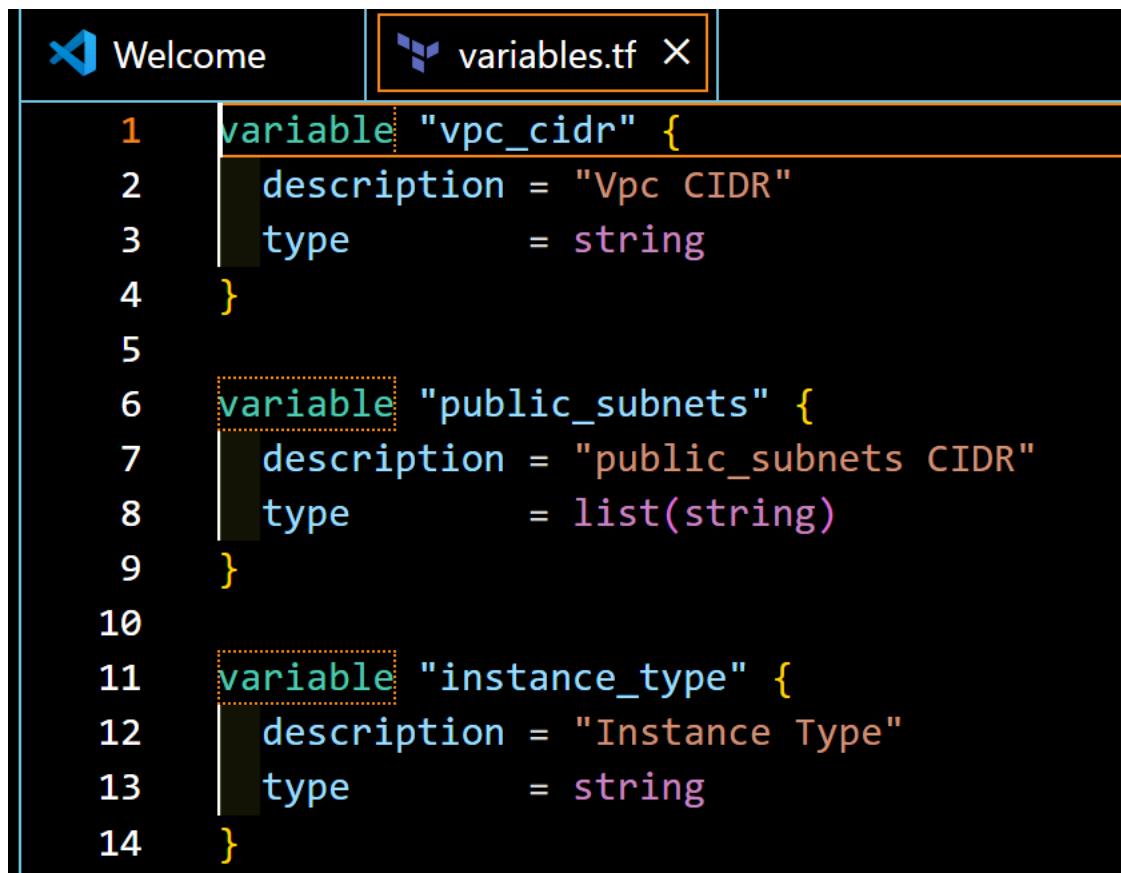
```
63 #ec2
64
65 module "ec2_instance" {
66   source = "terraform-aws-modules/ec2-instance/aws"
67
68   name = "jenkins_server"
69
70   instance_type      = var.instance_type
71   ami                = data.aws_ami.example.id
72   key_name          = "my-new-key"
73   monitoring         = true
74   vpc_security_group_ids = [module.sg.security_group_id]
75   subnet_id          = module.vpc.public_subnets[0]
76   associate_public_ip_address = true
77   availability_zone    = data.aws_availability_zones.azs.names[0]
78   user_data           = file("jenkins-install.sh")
79
80
81   tags = {
82     Name      = "jenkins_server"
83     Terraform = "true"
84     Environment = "dev"
85   }
86 }
```

master AWS: profile/default

File Edit Selection View ... ← → CICD-Terraform-EKS ESP-IDF: Search Error Hint ...

Ln 2, Col 1 Spaces: 2 UTF-8 CRLF { } Terraform Go Live

Type here to search



```
1 variable "vpc_cidr" {
2   description = "Vpc CIDR"
3   type        = string
4 }
5
6 variable "public_subnets" {
7   description = "public_subnets CIDR"
8   type        = list(string)
9 }
10
11 variable "instance_type" {
12   description = "Instance Type"
13   type        = string
14 }
```

```
# tf init
```

```
$ terraform init

Initializing the backend...
Initializing modules...

Initializing provider plugins...
- Reusing previous version of hashicorp/aws from the dependency lock file
- Using previously-installed hashicorp/aws v5.38.0

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.
```

```
# tf plan
```

```
$ terraform plan
data.aws_availability_zones.azs: Reading...
data.aws_ami.example: Reading...
data.aws_availability_zones.azs: Read complete after 1s [id=us-east-2]
data.aws_ami.example: Read complete after 1s [id=ami-05fb0b8c1424f266b]

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:

# module.vpc.aws_default_network_acl.this[0] will be created
+ resource "aws_default_network_acl" "this" {
    + arn                  = (known after apply)
    + default_network_acl_id = (known after apply)
    + id                   = (known after apply)
    + owner_id              = (known after apply)
    + tags                 = {
        + "Environment" = "dev"
        + "Name"        = "jenkins_vpc"
    }
}
```

```
# module.vpc.aws_default_network_acl.this[0] will be created
+ resource "aws_default_network_acl" "this" {
    + arn                  = (known after apply)
    + default_network_acl_id = (known after apply)
    + id                   = (known after apply)
    + owner_id              = (known after apply)
    + tags                 = {
        + "Environment" = "dev"
        + "Name"        = "jenkins_vpc"
        + "Terraform"   = "true"
    }
    + tags_all              = {
        + "Environment" = "dev"
        + "Name"        = "jenkins_vpc"
        + "Terraform"   = "true"
    }
    + vpc_id                = (known after apply)

    + egress {
        + action      = "allow"
        + from_port   = 0
    }
}
```

```
+ ingress {
+   action      = "allow"
+   cidr_block = "0.0.0.0/0"
+   from_port   = 0
+   protocol    = "-1"
+   rule_no     = 100
+   to_port     = 0
}
}

# module.vpc.aws_default_route_table.default[0] will be created
+ resource "aws_default_route_table" "default" {
+   arn                  = (known after apply)
+   default_route_table_id = (known after apply)
+   id                   = (known after apply)
+   owner_id             = (known after apply)
+   route                = (known after apply)
+   tags                 = {
+     "Environment" = "dev"
+     "Name"        = "jenkins_vpc"
+     "Terraform"   = "true"
+   }
+   tags_all             = {
+     "Environment" = "dev"
+   }
}
```

```
# module.vpc.aws_route.public_internet_gateway[0] will be created
+ resource "aws_route" "public_internet_gateway" {
+   destination_cidr_block = "0.0.0.0/0"
+   gateway_id              = (known after apply)
+   id                      = (known after apply)
+   id                      = (known after apply)
+   instance_id              = (known after apply)
+   instance_owner_id        = (known after apply)
+   network_interface_id     = (known after apply)
+   vpc_id                   = (known after apply)
+   route_table_id           = (known after apply)
+   state                    = (known after apply)

+   timeouts {
+     create = "5m"
+   }
}

# module.vpc.aws_route_table.public[0] will be created
+ resource "aws_route_table" "public" {
+   arn                  = (known after apply)
+   id                   = (known after apply)
+   owner_id             = (known after apply)
+   propagating_vgws     = (known after apply)
+   route                = (known after apply)
+   tags                 = [
```

```
]}
```

```
$ terraform apply --auto-approve
```

```
module.vpc.aws_subnet.public[0]: Creating...
module.vpc.aws_default_route_table.default[0]: Creating...
module.vpc.aws_default_network_acl.this[0]: Creating...
module.vpc.aws_default_route_table.default[0]: creation complete after 2s [id=rtb-09a97dd9a0a795]
module.vpc.aws_internet_gateway.this[0]: creation complete after 2s [id=igw-02c076e0d6252506d]
module.vpc.aws_route_table.public[0]: creation complete after 2s [id=rtb-09a97dd9a0a795]
module.vpc.aws_route_table_association.public[0]: creating...
module.vpc.aws_route_table_association.public[0]: creation complete after 1s [id=rtbassoc-03fd6f93e16c3064c]
```

Jenkins VPC



The screenshot shows the AWS VPC dashboard with the following details:

Name	VPC ID	State	IPv4 CIDR
jenkins_vpc	vpc-0fe28b784833ab601	Available	10.0.0.0/16
-	vpc-0c1438106e3372c5b	Available	172.31.0.0/16

The left sidebar includes links for VPC dashboard, EC2 Global View, Filter by VPC (Select a VPC), Virtual private cloud (Your VPCs, Subnets, Route tables, Internet gateways, Egress-only internet gateways, DHCP option sets, Elastic IPs, Managed prefix lists), and a bottom link for Managed prefix lists.

Subnets

VPC dashboard X

EC2 Global View []

Filter by VPC: Select a VPC

Virtual private cloud []

Your VPCs

Subnets

- Route tables
- Internet gateways
- Egress-only internet gateways
- DHCP option sets
- Elastic IPs
- Managed prefix lists

Subnets (4) Info

Name	Subnet ID	State	VPC
jenkins_subnet	subnet-0d4c0c1ea9b35ffc2	Available	vpc-0fe28b784
-	subnet-097dc371369326f28	Available	vpc-0c1438106
-	subnet-047973a4b2bacf572	Available	vpc-0c1438106

Select a subnet

IG

EC2 Global View []

Filter by VPC: Select a VPC

Virtual private cloud []

Your VPCs

Subnets

Route tables

Internet gateways

- Egress-only internet gateways
- DHCP option sets
- Elastic IPs
- Managed prefix lists

Internet gateways (2) Info

Name	Internet gateway ID	State	VPC ID
jenkins_vpc	igw-07463f645455bf795	Attached	vpc-0fe
-	igw-0e0d359bb0391c2d3	Attached	vpc-0c1

Select an internet gateway above

Go to security group cp

The screenshot shows a web browser window with three tabs open. The active tab displays Terraform code for creating a security group named 'web-server' with a single ingress rule allowing traffic from '10.10.0.0/16' on port 80. A 'Copy' button is visible in the top right corner of the code block.

```
module "web_server_sg" {  
  source = "terraform-aws-modules/security-group/aws//modules/http-80"  
  
  name      = "web-server"  
  description = "Security group for web-server with HTTP ports open within VPC"  
  vpc_id    = "vpc-12345678"  
  
  ingress_cidr_blocks = ["10.10.0.0/16"]  
}
```

Below the code, there is a section titled 'Security group with custom rules'. A cookie consent banner is present, with 'Manage Preferences' and 'Dismiss' buttons. The browser's taskbar at the bottom shows various pinned icons.

The screenshot shows a Windows desktop environment with a code editor window titled 'CICD-Terraform-EKS'. The main editor pane displays Terraform code for a security group named 'jenkins_sg' with two ingress rules: one for port 8080 (HTTP) and one for port 22 (SSH). The code editor interface includes a sidebar with file navigation and a status bar at the bottom.

```
module "sg" {  
  source = "terraform-aws-modules/security-group/aws"  
  
  name      = "jenkins_sg"  
  description = "Security group for jenkins server"  
  vpc_id    = module.vpc.vpc_id  
  
  ingress_with_cidr_blocks = [  
    {  
      from_port  = 8080  
      to_port    = 8080  
      protocol   = "tcp"  
      description = "HTTP"  
      cidr_blocks = "0.0.0.0/0"  
    },  
    {  
      from_port  = 22  
      to_port    = 22  
      protocol   = "tcp"  
      description = "SSH"  
      cidr_blocks = "0.0.0.0/0"  
    }  
  ]  
  egress_with_cidr_blocks = [  
    {  
      from_port  = 0  
      to_port    = 0  
      protocol   = "-1"  
      cidr_blocks = "0.0.0.0/0"  
    }  
  ]  
}
```

```
# tf init
```

```
$ terraform init

Initializing the backend...
Initializing modules...

Initializing provider plugins...
- Reusing previous version of hashicorp/aws from the dependency lock file
- Using previously-installed hashicorp/aws v5.38.0

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.
```

```
$ terraform fmt
main.tf

MINGW64 ~/OneDri
$ terraform validate
Success! The configuration is valid.
```

```
$ terraform plan
data.aws_availability_zones.azs: Reading...
data.aws_ami.example: Reading...
module.vpc.aws_vpc.this[0]: Refreshing state... [id=vpc-0fe28b784833ab601]
data.aws_availability_zones.azs: Read complete after 1s [id=us-east-2]
data.aws_ami.example: Read complete after 1s [id=ami-05fb0b8c1424f266b]
module.vpc.aws_default_route_table.default[0]: Refreshing state... [id=rtb-03e02fea526789eb4]
module.vpc.aws_default_security_group.this[0]: Refreshing state... [id=sg-02c076e0d6252506d]
module.vpc.aws_default_network_acl.this[0]: Refreshing state... [id=acl-0895d359c7de6010a]
module.vpc.aws_internet_gateway.this[0]: Refreshing state... [id=igw-07463f645455bf795]
module.vpc.aws_route_table.public[0]: Refreshing state... [id=rtb-09a9fbb9d03a71cee]
module.vpc.aws_subnet.public[0]: Refreshing state... [id=subnet-0d4c0c1ea9b35ffc2]
module.vpc.aws_route.public_internet_gateway[0]: Refreshing state... [id=r-rtb-09a9fbb9d03a71cee1080289494]
module.vpc.aws_route_table_association.public[0]: Refreshing state... [id=rtbassoc-03fd6f93e16c3064c]
```

```
# module.web_server_sg.module.sg.aws_security_group_rule.ingress
eated
+ resource "aws_security_group_rule" "ingress_with_self" {
    + description          = "Ingress Rule"
    + from_port            = -1
    + id                   = (known after apply)
    + prefix_list_ids     = []
    + protocol             = "-1"
    + security_group_id   = (known after apply)
    + security_group_rule_id = (known after apply)
    + self                 = true
    + source_security_group_id = (known after apply)
    + to_port              = -1
    + type                 = "ingress"
}
}
```

Plan: 7 to add, 0 to change, 0 to destroy.

Note: You didn't use the `-out` option to save this plan, so Terraform can't guarantee to take exactly these actions if you run `"terraform apply"` now.

```
$ terraform apply --auto-approve
data.aws_availability_zones.azs: Reading...
data.aws_ami.example: Reading...
module.vpc.aws_vpc.this[0]: Refreshing state... [id=vpc-0fe28b784833ab601]
data.aws_availability_zones.azs: Read complete after 1s [id=us-east-2]
data.aws_ami.example: Read complete after 1s [id=ami-05fb0b8c1424f266b]
module.vpc.aws_default_route_table.default[0]: Refreshing state... [id=rtb-03e02fea526789eb4]
module.vpc.aws_default_network_acl.this[0]: Refreshing state... [id=acl-0895d359c7de6010a]
module.vpc.aws_internet_gateway.this[0]: Refreshing state... [id=igw-07463f645455bf795]
module.vpc.aws_route_table.public[0]: Refreshing state... [id=rtb-09a9fbb9d03a71cee]
module.vpc.aws_default_security_group.this[0]: Refreshing state... [id=sg-02c076e0d6252506d]
module.vpc.aws_subnet.public[0]: Refreshing state... [id=subnet-0d4c0c1ea9b35ffc2]
module.vpc.aws_route.public_internet_gateway[0]: Refreshing state... [id=r-rtb-09a9fbb9d03a71cee1080289494]
module.vpc.aws_route_table_association.public[0]: Refreshing state... [id=rtbassoc-03fd6f93e16c3064c]
```

SG in aws created

Name	Security group ID	Security group name
jenkins_sg	sg-020e779870384f87f	jenkins_sg-202402281153584744000...

Now Created AWS EC2 Module

A screenshot of a web browser displaying the Terraform Registry. The URL is registry.terraform.io/modules/terraform-aws-modules/ec2-instance/aws/latest. The page shows the 'ec2-instance' module by 'aws provider'. It includes a logo, a brief description ('Terraform module to create AWS EC2 instance(s) resources'), and a 'Version 5.7.1 (latest)' dropdown. To the right, there's a sidebar with 'Module Downloads' statistics: 'Downloads this week' (94,302), 'Downloads this month' (94,302), 'Downloads this year' (6.3M), and 'Downloads over all time' (17.9M). Below the main content, there's a 'Provision Instructions' section with a 'Copy' button, and a note about cookie usage with 'Manage Preferences' and 'Dismiss' buttons. The browser's taskbar at the bottom shows other open tabs related to EKS and Terraform.

cp

A screenshot of a web browser displaying the Terraform Registry. The URL is registry.terraform.io/modules/terraform-aws-modules/ec2-instance/aws/latest. The page shows the 'Single EC2 Instance' configuration. A code block contains the following Terraform code:

```
module "ec2_instance" {
  source  = "terraform-aws-modules/ec2-instance/aws"

  name = "single-instance"

  instance_type      = "t2.micro"
  key_name          = "user1"
  monitoring        = true
  vpc_security_group_ids = ["sg-12345678"]
  subnet_id         = "subnet-eddcddzz4"

  tags = {
    Terraform  = "true"
    Environment = "dev"
  }
}
```

The page also includes a note about cookie usage with 'Manage Preferences' and 'Dismiss' buttons. The browser's taskbar at the bottom shows other open tabs related to EKS and Terraform.

Existing Key pair

AMI Catalog

Elastic Block Store

- Volumes
- Snapshots
- Lifecycle Manager

Network & Security

- Security Groups
- Elastic IPs
- Placement Groups
- Key Pairs**
- Network Interfaces

Load Balancing

File Edit Selection View ... ← → 🔍 CICD-Terraform-EKS

Welcome | main.tf | ESP-IDF: Search Error Hint

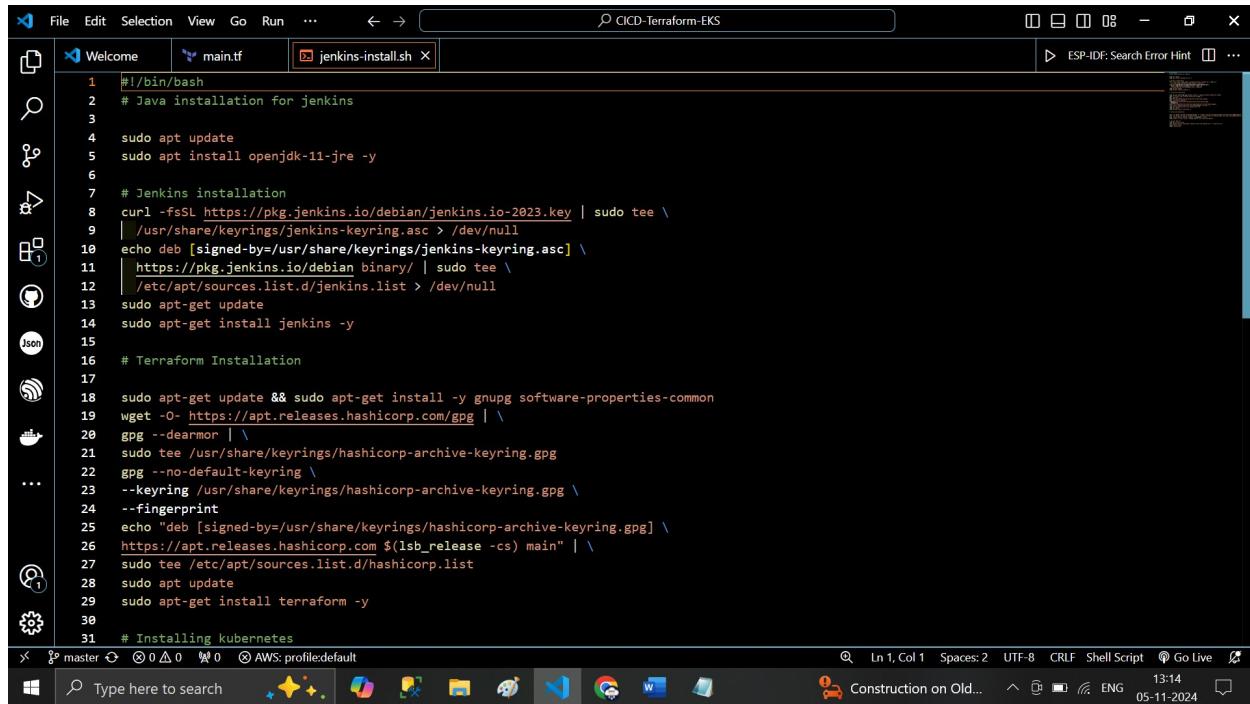
```

63 #ec2
64
65 module "ec2_instance" {
66   source = "terraform-aws-modules/ec2-instance/aws"
67
68   name = "jenkins_server"
69
70   instance_type      = var.instance_type
71   ami                = data.aws_ami.example.id
72   key_name          = "my-new-key"
73   monitoring         = true
74   vpc_security_group_ids = [module.sg.security_group_id]
75   subnet_id          = module.vpc.public_subnets[0]
76   associate_public_ip_address = true
77   availability_zone    = data.aws_availability_zones.azs.names[0]
78   user_data           = file("jenkins-install.sh")
79
80
81   tags = {
82     Name      = "jenkins_server"
83     Terraform = "true"
84     Environment = "dev"
85   }
86 }
```

master AWS: profile=default

Type here to search

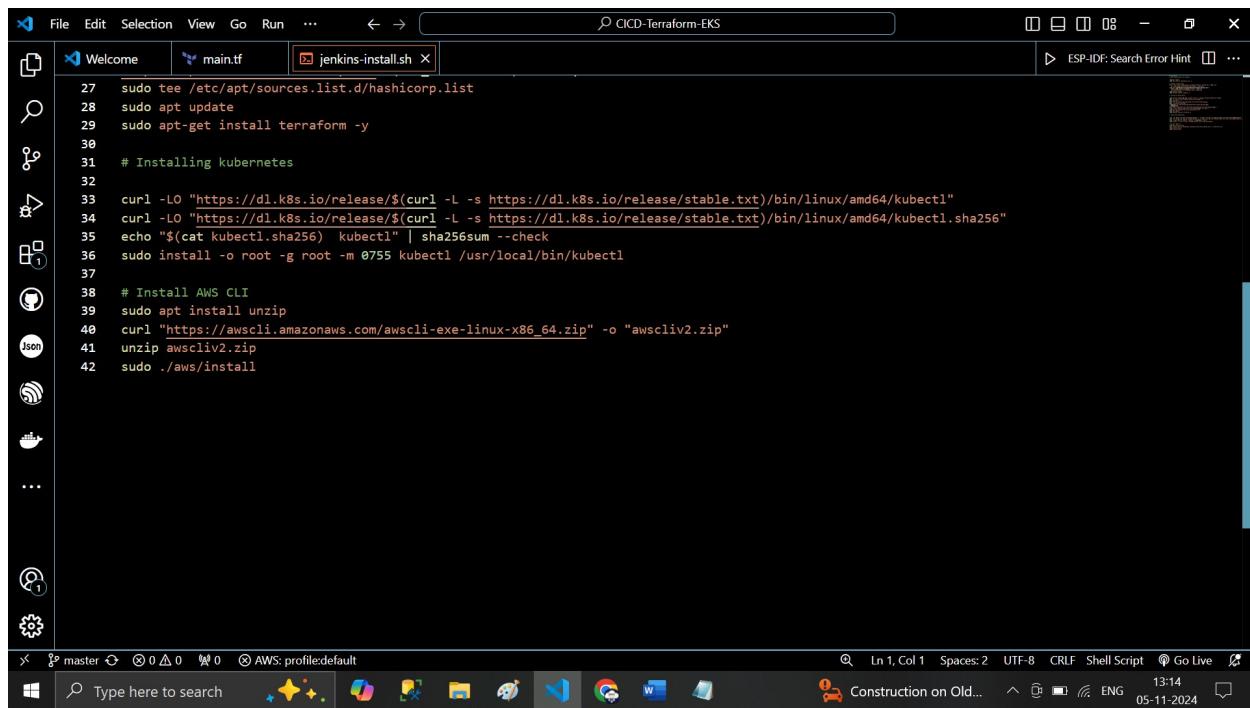
Creating a shell file name jenkins-install that will automate the installation of java, jenkins, tf &K8S



```
File Edit Selection View Go Run ... ← → ⌘ CICD-Terraform-EKS
Welcome main.tf jenkins-install.sh ESP-IDF: Search Error Hint ...
1#!/bin/bash
2# Java installation for jenkins
3
4sudo apt update
5sudo apt install openjdk-11-jre -y
6
7# Jenkins installation
8curl -fsSL https://pkg.jenkins.io/debian/jenkins.io-2023.key | sudo tee \
| /usr/share/keyrings/jenkins-keyring.asc > /dev/null
10echo deb [signed-by=/usr/share/keyrings/jenkins-keyring.asc] \
https://pkg.jenkins.io/debian binary/ | sudo tee \
/etc/apt/sources.list.d/jenkins.list > /dev/null
13sudo apt-get update
14sudo apt-get install jenkins -y
15
16# Terraform Installation
17
18sudo apt-get update && sudo apt-get install -y gnupg software-properties-common
19wget -O https://apt.releases.hashicorp.com/gpg | \
20gpg --dearmor | \
21sudo tee /usr/share/keyrings/hashicorp-archive-keyring.gpg
22gpg --no-default-keyring \
--keyring /usr/share/keyrings/hashicorp-archive-keyring.gpg \
--fingerprint
25echo "deb [signed-by=/usr/share/keyrings/hashicorp-archive-keyring.gpg] \
https://apt.releases.hashicorp.com $(lsb_release -cs) main" | \
27sudo tee /etc/apt/sources.list.d/hashicorp.list
28sudo apt update
29sudo apt-get install terraform -y
30
31# Installing kubernetes
```

master ⌂ ⑧ 0 △ 0 ⌂ 0 AWS: profiledefault

Type here to search Construction on Old... 13:14 05-11-2024



```
File Edit Selection View Go Run ... ← → ⌘ CICD-Terraform-EKS
Welcome main.tf jenkins-install.sh ESP-IDF: Search Error Hint ...
27sudo tee /etc/apt/sources.list.d/hashicorp.list
28sudo apt update
29sudo apt-get install terraform -y
30
31# Installing kubernetes
32
33curl -LO "https://dl.k8s.io/release/$(curl -L -s https://dl.k8s.io/release/stable.txt)/bin/linux/amd64/kubectl"
34curl -LO "https://dl.k8s.io/release/$(curl -L -s https://dl.k8s.io/release/stable.txt)/bin/linux/amd64/kubectl.sha256"
35echo "$(cat kubectl.sha256) kubectl" | sha256sum --check
36sudo install -o root -g root -m 0755 kubectl /usr/local/bin/kubectl
37
38# Install AWS CLI
39sudo apt install unzip
40curl "https://awscli.amazonaws.com/awscli-exe-linux-x86_64.zip" -o "awscliv2.zip"
41unzip awscliv2.zip
42sudo ./aws/install
```

master ⌂ ⑧ 0 △ 0 ⌂ 0 AWS: profiledefault

Type here to search Construction on Old... 13:14 05-11-2024

This Jenkins—install.sh is placed in main.tf file under ec2 installation automated

```

File Edit Selection View ... ← → ⌂ CICD-Terraform-EKS
Welcome main.tf jenkins-install.sh ⌂ ESP-IDF: Search Error Hint ...
63 #ec2
64
65 module "ec2_instance" {
66   source = "terraform-aws-modules/ec2-instance/aws"
67
68   name = "jenkins_server"
69
70   instance_type      = var.instance_type
71   ami                = data.aws_ami.example.id
72   key_name           = "my-new-key"
73   monitoring         = true
74   vpc_security_group_ids = [module.sg.security_group_id]
75   subnet_id          = module.vpc.public_subnets[0]
76   associate_public_ip_address = true
77   availability_zone  = data.aws_availability_zones.azs.names[0]
78   user_data... = file("jenkins-install.sh")
79
80
81   tags = {
82     Name      = "jenkins_server"
83     Terraform = "true"
84     Environment = "dev"
85   }
86 }

```

master ↻ 0 △ 0 ⏪ 0 ⌂ AWS: profile:default Q Ln 78, Col 59 (56 selected) Spaces: 2 UTF-8 CRLF ⌂ Terraform ⌂ Go Live ⌂ Type here to search ⌂ 26°C Partly sunny ⌂ 13:15 ENG 05-11-2024

```

$ terraform init

Initializing the backend...
Initializing modules...

Initializing provider plugins...
- Reusing previous version of hashicorp/aws from the dependency lock file
- Using previously-installed hashicorp/aws v5.38.0

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.

```

```

$ terraform fmt
main.tf

MINGW64 ~/OneDr
$ terraform validate
Success! The configuration is valid.

```

```

$ terraform plan
data.aws_availability_zones.azs: Reading...
module.ec2_instance.data.aws_partition.current: Reading...
data.aws_ami.example: Reading...
module.ec2_instance.data.aws_partition.current: Read complete after 0s [id=aws]
data.aws_availability_zones.azs: Read complete after 1s [id=us-east-2]
data.aws_ami.example: Read complete after 1s [id=ami-05fb0b8c1424f266b]

```

```

+ placement_partition_number          = (known after apply)
+ primary_network_interface_id       = (known after apply)
+ private_dns                       = (known after apply)
+ private_ip                        = (known after apply)
+ public_dns                        = (known after apply)
+ public_ip                         = (known after apply)
+ secondary_private_ips             = (known after apply)
+ security_groups                   = (known after apply)
+ source_dest_check                 = true
+ spot_instance_request_id          = (known after apply)
+ subnet_id                         = (known after apply)
+ tags                             = {
    + "Environment" = "dev"
    + "Name"        = "jenkins-server"
    + "Terraform"   = "true"
}
+ tags_all                          = {
    + "Environment" = "dev"
    + "Name"        = "jenkins-server"
    + "Terraform"   = "true"
}
+ tenancy                           = (known after apply)
+ user_data                         = "db5c567bf3859e5f2d7676cae3f1a3f5f3b12b6b"
+ user_data_base64                  = (known after apply)
+ user_data_replace_on_change       = false
+ volume_tags                       = {
```

File Encrypted

```

+ secondary_private_ips              = (known after apply)
+ security_groups                   = (known after apply)
+ source_dest_check                 = true
+ spot_instance_request_id          = (known after apply)
+ subnet_id                         = (known after apply)
+ tags                             = {
    + "Environment" = "dev"
    + "Name"        = "jenkins-server"
    + "Terraform"   = "true"
}
+ tags_all                          = {
    + "Environment" = "dev"
    + "Name"        = "jenkins-server"
    + "Terraform"   = "true"
}
+ tenancy                           = (known after apply)
+ user_data                         = "db5c567bf3859e5f2d7676cae3f1a3f5f3b12b6b"
+ user_data_base64                  = (known after apply)
+ user_data_replace_on_change       = false
+ volume_tags                       = {
    + "Name" = "jenkins_server"
}
+ vpc_security_group_ids            = (known after apply)
+ credit_specification {}
```

```
$ terraform apply --auto-approve
data.aws_availability_zones.azs: Reading...
module.ec2_instance.data.aws_partition.current: Reading...
data.aws_ami.example: Reading...
module.ec2_instance.data.aws_partition.current: Read complete after 0s [id=aws]
data.aws_availability_zones.azs: Read complete after 1s [id=us-east-2]
data.aws_ami.example: Read complete after 2s [id=ami-05fb0b8c1424f266b]
```

```

groute-354121192]
module.sg.aws_security_group_rule.ingress_with_cidr_blocks[0]: Creation complete after 2s [id=sgrule-3083431863]
module.sg.aws_security_group_rule.ingress_with_cidr_blocks[1]: Creation complete after 3s [id=sgrule-1648177788]
module.vpc.aws_subnet.public[0]: still creating... [10s elapsed]
module.vpc.aws_subnet.public[0]: Creation complete after 13s [id=subnet-0ba85794f2c53f813]
module.vpc.aws_route_table_association.public[0]: Creating...
module.ec2_instance.aws_instance.this[0]: Creating...
module.vpc.aws_route_table_association.public[0]: Creation complete after 1s [id=rtbassoc-0cd00f1150665f648]
module.ec2_instance.aws_instance.this[0]: still creating... [10s elapsed]
module.ec2_instance.aws_instance.this[0]: Still creating... [20s elapsed]
module.ec2_instance.aws_instance.this[0]: Still creating... [30s elapsed]
module.ec2_instance.aws_instance.this[0]: Creation complete after 37s [id=i-009b8821137028137]

```

The screenshot shows the AWS EC2 Instances page. On the left, there's a sidebar with navigation links like EC2 Dashboard, EC2 Global View, Events, Instances (selected), Instance Types, Launch Templates, Spot Requests, Savings Plans, Reserved Instances, Dedicated Hosts, Capacity Reservations, and a New button. The main area displays a table with one row for the instance 'jenkins_server'. The table columns include Name, Instance ID, Instance state, Instance type, and Status check. The instance is listed as 'Running' with an 't2.medium' type and an 'Initializing' status check.

Instances (1) Info				
G Connect Instance state ▾ Actions ▾ Launch instances ▾				
<input type="text" value="Find Instance by attribute or tag (case-sensitive)"/>				
Any state				
<input type="checkbox"/>	Name ↴	Instance ID	Instance state	Instance type
<input type="checkbox"/>	jenkins_server	i-009b8821137028137	Running	t2.medium
Status check				

Select an instance

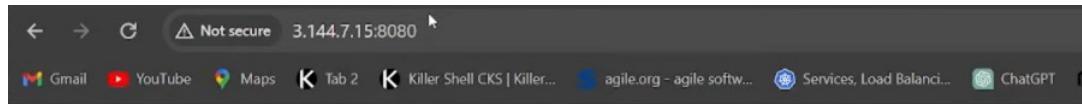
Check for IP Jenkins working env

This screenshot shows the detailed view for the instance 'jenkins_server'. It includes a summary table with fields like Instance ID, Private IPv4 address, and Public IPv4 DNS. Below the table, there are sections for Instance summary, Network interfaces, and Block device mapping.

Any state				
<input checked="" type="checkbox"/>	Name ↴	Instance ID	Instance state	Instance type
<input checked="" type="checkbox"/>	jenkins_server	i-009b8821137028137	Running	t2.medium
Status check				

Instance: i-009b8821137028137 (jenkins_server)

Instance summary Info		
Instance ID	Private IPv4 address	Public IPv4 DNS
i-009b8821137028137 (jenkins_server)	3.144.7.15 open address	10.0.1.133
IPv6 address	Instance state	Public IPv4 DNS



End of Part 1

PART-2

1. Writing terraform code for creation of EKS cluster
2. Pushing code to github repo.

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

```
I
```

```
ubuntu@ip-10-0-1-133:~$ sudo cat /var/lib/jenkins/secrets/initialAdminPassword
dbf60c487c8647289d7e2598a17a8cd2
ubuntu@ip-10-0-1-133:~$
```

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

```
.....
```

Plugins extend Jenkins with additional features to support many different needs.

Install suggested plugins

Install plugins the Jenkins community finds most useful.

Select plugins to install

Select and install plugins most suitable for your needs.

Getting Started

The screenshot shows the Jenkins 'Getting Started' page. At the top, there are several checkboxes for configuration options: 'Folders', 'OWAS', 'Markup', 'Build Timeout', 'Credentials Binding', and 'JSON Path API'. Below these are two main sections: 'Install suggested plugins' (blue background) and 'Select plugins to install' (white background). The 'Install suggested plugins' section contains the text: 'Install plugins the Jenkins community finds most useful.' The 'Select plugins to install' section contains the text: 'Select and install plugins most suitable for your needs.' To the right of these sections is a sidebar titled 'API' which lists various Jenkins APIs with their descriptions. The sidebar includes links for 'Token Macro', 'Build Timeout', 'bouncycastle API', 'Instance Identity', 'JavaBeans Activation Framework (JAF) API', 'JavaMail API', 'Credentials', 'Plain Credentials', 'Gson API', 'Trilead API', and a note about required dependencies.

API	Description
Token Macro	Macro for building tokens.
Build Timeout	API for setting build timeouts.
bouncycastle API	API for using the Bouncy Castle library.
Instance Identity	API for managing instance identities.
JavaBeans Activation Framework (JAF) API	API for JavaBeans Activation Framework.
JavaMail API	API for JavaMail.
Credentials	API for managing credentials.
Plain Credentials	API for plain text credentials.
Gson API	API for Gson.
Trilead API	API for Trilead.
- required dependency	Required dependencies.

Create First Admin User

Username

Password

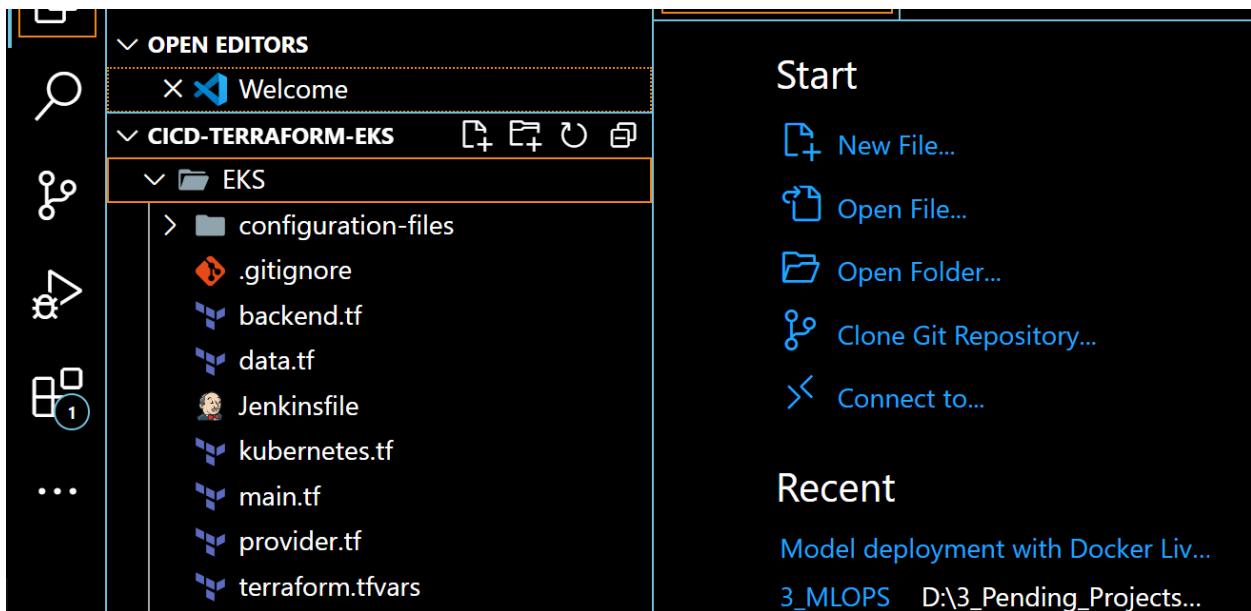
Confirm password

Jenkins is ready!

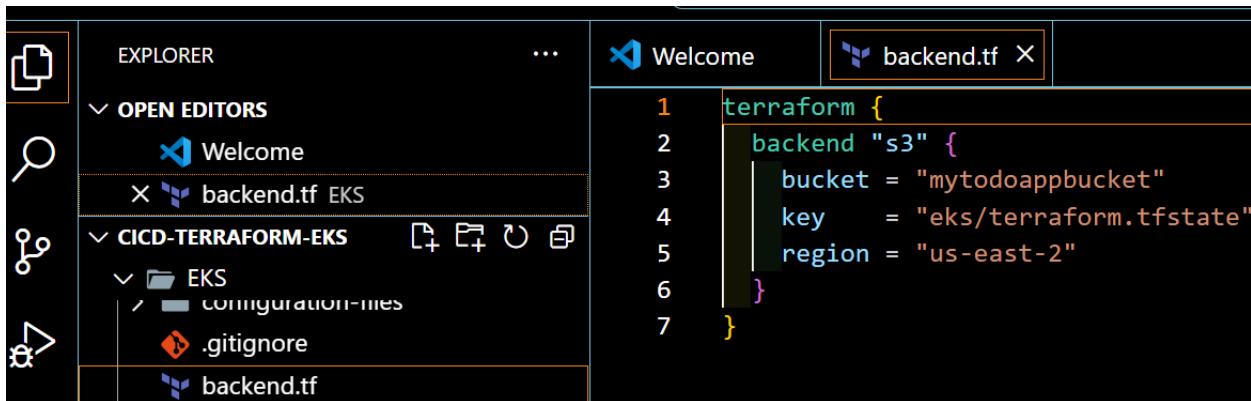
Your Jenkins setup is complete.

[Start using Jenkins](#)

The screenshot shows the Jenkins dashboard. At the top, there's a navigation bar with the Jenkins logo, a search bar, and a notifications icon. Below the bar, the page title is "Welcome to Jenkins!". A sub-header says, "This page is where your Jenkins jobs will be displayed. To get started, you can set up distributed builds or start building a software project." A prominent button labeled "Start building your software project" is centered. On the left side, there's a sidebar with links: "New Item", "People", "Build History", "Manage Jenkins", and "My Views". A "Build Queue" section indicates "No builds in the queue.". At the bottom right, there's a "Create a job" button and a "+" sign.



Creating EKS cluster



Main.tf

The screenshot shows the Visual Studio Code interface with the title bar "CICD-Terraform-EKS". The left sidebar displays the project structure under "OPEN EDITORS" for "CICD-TERRAFORM-EKS". The "main.tf" file is open in the center editor, showing Terraform code for creating a VPC module. The code defines a module named "vpc" with various configuration parameters like azs, public_subnets, private_subnets, enable_dns_hostnames, enable_nat_gateway, single_nat_gateway, and tags.

```
#Vpc
module "vpc" {
  source = "terraform-aws-modules/vpc/aws"

  name = "eks_cluster_vpc"
  cidr = var.vpc_cidr

  azs     = data.aws_availability_zones.azs.names
  public_subnets = var.public_subnets
  private_subnets = var.private_subnets

  enable_dns_hostnames = true
  enable_nat_gateway   = true
  single_nat_gateway   = true

  tags = [
    "kubernetes.io/cluster/my-eks-cluster" = "shared"
  ]
  public_subnet_tags = [
    "kubernetes.io/cluster/my-eks-cluster" = "shared"
    "kubernetes.io/role/elb"               = 1
  ]
  private_subnet_tags = [
    "kubernetes.io/cluster/my-eks-cluster" = "shared"
    "kubernetes.io/role/private_elb"       = 1
  ]
}
```

The screenshot shows the Visual Studio Code interface with the title bar "CICD-Terraform-EKS". The left sidebar displays the project structure under "OPEN EDITORS" for "CICD-Terraform-EKS". The "variables.tf" file is open in the center editor, defining four variables: "vpc_cidr", "public_subnets", "private_subnets", and "instance_types", each with a string type and a description.

```
variable "vpc_cidr" {
  description = "Vpc CIDR"
  type        = string
}

variable "public_subnets" {
  description = "public_subnets CIDR"
  type        = list(string)
}

variable "private_subnets" {
  description = "private_subnets CIDR"
  type        = list(string)
}

variable "instance_types" {
  description = "Node Instances"
  type        = list(string)
}
```

```
$ terraform init

Initializing the backend...
Initializing modules...

Initializing provider plugins...
- Finding hashicorp/aws versions matching ">= 5.20.0"...
- Installing hashicorp/aws v5.38.0...
- Installed hashicorp/aws v5.38.0 (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.
```

```
$ terraform validate
Success! The configuration is valid.
```

```
$ terraform plan
data.aws_availability_zones.azs: Reading...
```

```
$ terraform plan
data.aws_availability_zones.azs: Reading...
data.aws_availability_zones.azs: Read complete after 1s [id=us-east-2]

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:

# module.vpc.aws_default_network_acl.this[0] will be created
+ resource "aws_default_network_acl" "this" {
    + arn                  = (known after apply)
    + default_network_acl_id = (known after apply)
    + id                   = (known after apply)
    + owner_id             = (known after apply)
    + tags                 = {
        + "Name"            = "eks_cluster_vpc-default"
        + "kubernetes.io/cluster/my-eks-cluster" = "shared"
    }
    + tags_all             = {
        + "Name"            = "eks_cluster_vpc-default"
        + "kubernetes.io/cluster/my-eks-cluster" = "shared"
    }
    + vpc_id               = (known after apply)
    + egress {
        + action            = "allow"
    }
}
```

```
# Terraform eks module Doc
```

A screenshot of a Windows desktop environment. In the center is a Microsoft Edge browser window displaying the Terraform AWS EKS module code. The code defines a module named 'eks' with a source of 'terraform-aws-modules/eks/aws' and a version of '~> 20.0'. It includes configuration for EFA support, managed node groups (with instance types like p5.48xlarge), and pre-bootstrapping user data. A 'Copy' button is visible in the top right corner of the code editor. Below the browser is a taskbar with various pinned icons and a system tray showing the date and time (05-11-2024) and battery status.

```
module "eks" {  
  source  = "terraform-aws-modules/eks/aws"  
  version = "~> 20.0"  
  
  # Truncated for brevity ...  
  
  # Adds the EFA required security group rules to the shared  
  # security group created for the node group(s)  
  enable_efa_support = true  
  
  eks_managed_node_groups = {  
    example = {  
      instance_types = ["p5.48xlarge"]  
  
      # Exposes all EFA interfaces on the launch template created by the node group(s)  
      # This would expose all 32 EFA interfaces for the p5.48xlarge instance type  
      enable_efa_support = true  
  
      pre_bootstrap_user_data = <<-EOT  
        # Mount NVME instance store volumes since they are typically  
        # available on instance types that support EFA  
        setup-local-disks raid0  
      EOT  
  
      # EFA should only be enabled when connecting 2 or more nodes  
      # Do not use EFA on a single node workload  
      min_size     = 2  
      max_size     = 10  
      desired_size = 2  
    }  
  }  
}
```

We use cookies and other similar technology to collect data to improve your experience on our site, as described in our Privacy Policy and Cookie Policy.

Manage Preferences Dismiss

A screenshot of a Windows desktop environment, identical to the one above, showing the same browser window with the Terraform code. The code defines a module named 'eks' with a source of 'terraform-aws-modules/eks/aws' and a version of '~> 20.0'. It includes configuration for EFA support, managed node groups (with instance types like p5.48xlarge), and pre-bootstrapping user data. A 'Copy' button is visible in the top right corner of the code editor. Below the browser is a taskbar with various pinned icons and a system tray showing the date and time (05-11-2024) and battery status.

```
instance_types = ["p5.48xlarge"]  
  
# Exposes all EFA interfaces on the launch template created by the node group(s)  
# This would expose all 32 EFA interfaces for the p5.48xlarge instance type  
enable_efa_support = true  
  
pre_bootstrap_user_data = <<-EOT  
  # Mount NVME instance store volumes since they are typically  
  # available on instance types that support EFA  
  setup-local-disks raid0  
EOT  
  
# EFA should only be enabled when connecting 2 or more nodes  
# Do not use EFA on a single node workload  
min_size     = 2  
max_size     = 10  
desired_size = 2  
}
```

We use cookies and other similar technology to collect data to improve your experience on our site, as described in our Privacy Policy and Cookie Policy.

Manage Preferences Dismiss

The screenshot shows a Windows desktop environment with a code editor window open. The window title is "CICD-Terraform-EKS". The main pane displays a Terraform configuration file named "main.tf". The code defines an EKS cluster with various parameters like source, cluster name, version, and node group details. Below the code editor is a taskbar with several pinned icons, including File Explorer, Microsoft Edge, and the Start button. The system tray at the bottom right shows the date and time as "05-11-2024".

```
33 module "eks" {
34   source          = "terraform-aws-modules/eks/aws"
35   cluster_name    = "my-eks-cluster"
36   cluster_version = "1.29"
37   cluster_endpoint_public_access = true
38   vpc_id          = module.vpc.vpc_id
39   subnet_ids      = module.vpc.private_subnets
40
41   eks_managed_node_groups = {
42     nodes = {
43       min_size      = 1
44       max_size      = 3
45       desired_size  = 2
46       instance_types = var.instance_types
47     }
48   }
49   tags = {
50     Environment = "dev"
51     Terraform   = "true"
52   }
53 }
54
55
56 data "aws_eks_cluster" "cluster" {
57   name = module.eks.cluster_id
58 }
59
60 data "aws_eks_cluster_auth" "cluster" {
61   name = module.eks.cluster_id
62 }
63
```

```
$ terraform plan
data.aws_availability_zones.azs: Reading...
data.aws_availability_zones.azs: Read complete after 1s [id=us-east-2]

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:

# module.vpc.aws_default_network_acl.this[0] will be created
+ resource "aws_default_network_acl" "this" {
    + arn          = (known after apply)
    + default_network_acl_id = (known after apply)
    + id           = (known after apply)
    + owner_id     = (known after apply)
    + tags         = {
        + "Name"          = "eks_cluster_vpc-default"
        + "kubernetes.io/cluster/my-eks-cluster" = "shared"
    }
    + tags_all     = {
        + "Name"          = "eks_cluster_vpc-default"
        + "kubernetes.io/cluster/my-eks-cluster" = "shared"
    }
    + vpc_id       = (known after apply)

    + egress {
        + action      = "allow"
    }
}

# module.eks.data.tls_certificate.this[0] will be read during apply
# (config refers to values not yet known)
<= data "tls_certificate" "this" {
    + certificates = (known after apply)
    + id           = (known after apply)
    + url          = (known after apply)
}

# module.eks.aws_cloudwatch_log_group.this[0] will be created
+ resource "aws_cloudwatch_log_group" "this" {
    + arn          = (known after apply)
    + id           = (known after apply)
    + log_group_class = (known after apply)
    + name         = "/aws/eks/my-eks-cluster/cluster"
    + name_prefix   = (known after apply)
    + retention_in_days = 90
    + skip_destroy  = false
}
```

```
$ terraform validate
Success! The configuration is valid.
```

Part 2 completed

Part 3

PART-3

1. Creating a jenkins pipeline - EKS cluster
2. Implementing a deployment files - kubectl
3. Deploying changes to aws

note Jenkins needs Aws Services for that access credentials from AWS is a must

The screenshot shows the Jenkins dashboard. At the top, there's a navigation bar with links for 'Dashboard', 'New Item', 'People', 'Build History', 'Manage Jenkins', and 'My Views'. Below the navigation bar, the main content area has a heading 'Welcome to Jenkins!'. It includes a 'Create a job' button with a '+' sign, a 'Start building your software project' link, and sections for 'Set up a distributed build' and 'Set up an agent'. On the left side, there's a 'Build Queue' section with a dropdown menu and a message stating 'No builds in the queue.'

manage Jenkins

The screenshot shows the 'Manage Jenkins' page. The left sidebar includes 'New Item', 'People', 'Build History', 'Manage Jenkins' (which is selected), and 'My Views'. A 'Build Queue' section indicates 'No builds in the queue.'. The main content area has a heading 'Manage Jenkins'. It features a warning message about running Jenkins on Java 11, which ends support on Sep 30, 2024. Buttons for 'Set up agent', 'Set up cloud', and 'Dismiss' are available. Below this, a 'Java 11 end of life in Jenkins' section provides more details and buttons for 'More Info' and 'Ignore'.

Credentails

Security



Security

Secure Jenkins; define who is allowed to access/use the system.



Credentials

Configure credentials



Credential Providers

Configure the credential providers and types



Users

Create/delete/modify users that can log in to this Jenkins.

Credentials

T P Store ↴

Domain

ID

Name

Stores scoped to Jenkins

P Store ↴

Domains



System

(glob*)

Icon: S M L

Username with password

Scope ?

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

Username ?

Treat username as secret ?

Password ?

Create

New credentials

Kind

Secret text

Scope [?](#)

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

1

Secret

Secret – Access Key

Secret

.....

ID [?](#)

AWS_ACCESS_KEY_ID

Description [?](#)

Access key ID	Secret access key
AKIARMRNV2OQYPTAZ2ZS	n0QMB9mhzt/IMjLQndAyy6dzmBFiKKnbd+3BFAF

Global credentials (unrestricted)

[+ Add Credentials](#)

Credentials that should be available irrespective of domain specification to requirements matching.

ID	Name	Kind	Description
 AWS_ACCESS_KEY_ID	AWS_ACCESS_KEY_ID	Secret text	
 AWS_SECRET_ACCESS_KEY	AWS_SECRET_ACCESS_KEY	Secret text	

Icon: S M L

Go to dashboard – Click on item – select pipeline

Enter an item name

» Required field

**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

OK Integrates 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.

» Required field

**Freestyle project**

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

**Pipeline**

OK Orchestrates long-running activities that can span multiple build agents. Suitable for building pipelines (formerly 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, builds, etc.

**OK**

Creates a container that stores nested items in it. Useful for grouping things together. Unlike view, which is just a

pipeline Script – Groovy Language

Configure

Definition
Pipeline script

Script ?
try sample Pipeline... ▾

```
1
```

I

Configure

Definition
Pipeline script

Script ?
try sample Pipeline... ▾

```
1 * pipeline{
2     agent any
3     environment {
4         AWS_ACCESS_KEY_ID = credentials('aws-access-key-id')
5     }
6 }
```

Checkout SCM – generate script – click pipeline syntax

Configure

General

Advanced Project Options

Pipeline

Script ?
try sample Pipeline... ▾

```
1 * pipeline{
2     agent any
3     environment {
4         AWS_ACCESS_KEY_ID = credentials('AWS_ACCESS_KEY_ID')
5         AWS_SECRET_ACCESS_KEY = credentials('AWS_SECRET_ACCESS_KEY')
6         AWS_DEFAULT_REGION = 'us-east-2'
7     }
8     stages{
9         stage('Checkout SCM'){
10            steps{
11                script{
12                    ...
13                }
14            }
15        }
16    }
17 }
```

Use Groovy Sandbox ?

Save Apply

Dashboard > terraform_cicd_eks > Pipeline Syntax

Snippet Generator

This **Snippet Generator** will help you learn the Pipeline Script code which can be used to define various steps. Pick a step you are interested in from the list, configure it, click **Generate Pipeline Script**, and you will see a Pipeline Script statement that would call the step with that configuration. You may copy and paste the whole statement into your script, or pick up just the options you care about. (Most parameters are optional and can be omitted in your script, leaving them at default values.)

Declarative Directive Generator

Declarative Online Documentation

Steps Reference

Global Variables Reference

Online Documentation

Examples Reference

IntelliJ IDEA GDSL

Steps

Sample Step

```
checkout: Check out from version control
```

checkout ?

SCM

```
Git
```

paste Your Repo

Repositories ?

Repository URL ?

! Please enter Git repository.

Credentials ?

- none -

+ Add ▾

h

<https://github.com/Siddhartha082/Deploy-EKS-Cluster-using-Terraform--Automate-Jenkins--AWS-EC2>

#cp

Include in polling? ?

Include in changelog? ?

Generate Pipeline Script

```
checkout scmGit(branches: [[name: '/master']], extensions: [], userRemoteConfigs: [[url: 'https://github.com/Siddhartha082/CICD_Terraform_EKS.git']])
```

paste it in script

```
4     AWS_ACCESS_KEY_ID = credentials('AWS_ACCESS_KEY_ID')
5     AWS_SECRET_ACCESS_KEY = credentials('AWS_SECRET_ACCESS_KEY')
6     AWS_DEFAULT_REGION = 'us-east-2'
7   }
8   stages{
9     stage('Checkout SCM'){
10       steps{
11         script{
12           checkout scmGit(branches: [[name: '/master']], extensions: [], userRemoteConfig:
13           }
14         }
15       }
16     }
17 }
```

Use Groovy Sandbox ?

Pipeline Syntax

Save

Apply

Build now

Dashboard > terraform_cicd_eks >

Status

</> Change Build scheduled

▷ Build Now

Add description

Disable Project

⚙ Configure

trash Delete Pipeline

🔍 Full Stage View

Stage View

No data available. This Pipeline has not yet run.

🔗 Rename

ⓘ Pipeline Syntax

Permalinks

 Configure

 Delete Pipeline

 Full Stage View

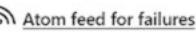
 Rename

 Pipeline Syntax

 Build History trend ▼

 Filter... /

 #1
Feb 29, 2024, 10:19 AM

 Atom feed for all  Atom feed for failures

Stage View

Average stage times:
(Average full run time: ~7s)

#1
Feb 29
15:49
No Changes

Success
Checkout SCM


4s

Permalinks

Going to 2nd Stage

Dashboard > terraform_cicd_eks > 

 Configure

 Delete Pipeline

 Full Stage View

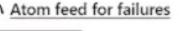
 Rename

 Pipeline Syntax

 Build History trend ▼

 Filter... /

 #1
Feb 29, 2024, 10:19 AM

 Atom feed for all  Atom feed for failures

Stage View

Average stage times:
(Average full run time: ~7s)

#1
Feb 29
15:49
No Changes

Checkout SCM

4s

4s

Permalinks

Click on Configure

```
| stage('Initializing Terraform'){
|   steps{
|     script{
|       dir('EKS'){
|         sh 'terraform init'
|       }
|     }
|   }
| }
```

Configure

Delete Pipeline

Full Stage View

Rename

Pipeline Syntax

Stage View

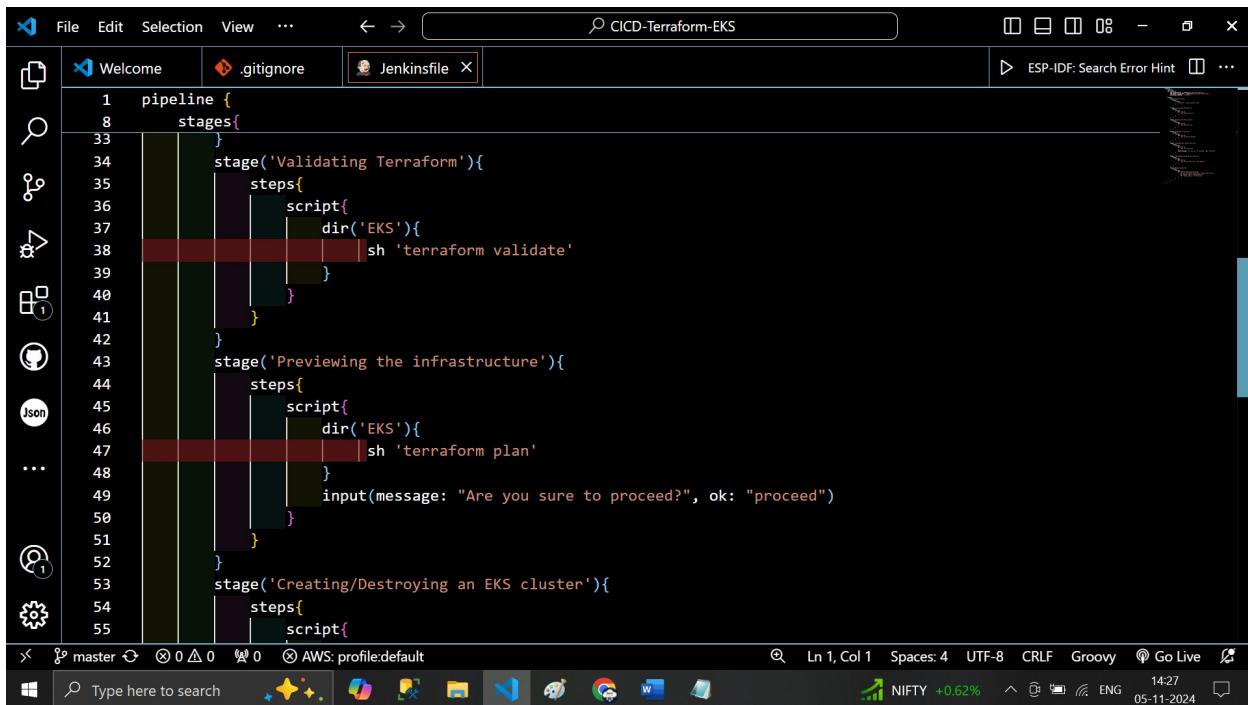


#1

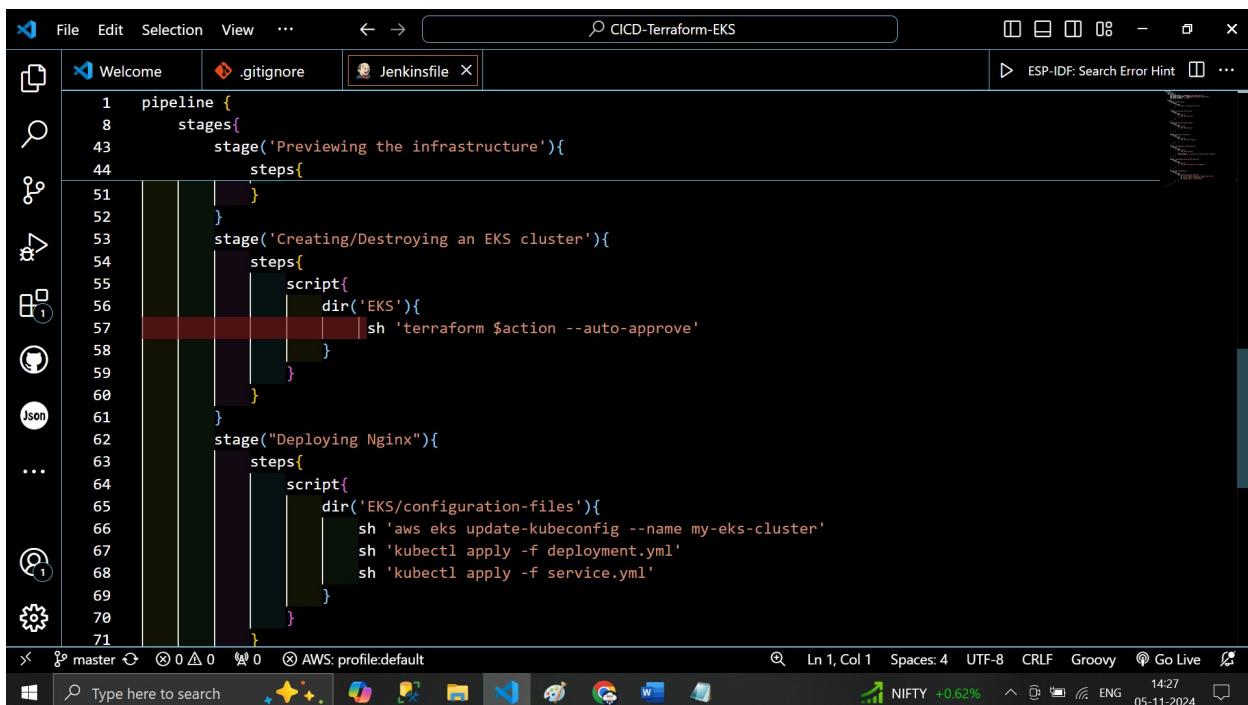
Console Output

```
[Pipeline] Start of Pipeline
[Pipeline] node
  Running on Jenkins in /var/lib/jenkins/workspace/terraform_cicd_eks
[Pipeline] {
[Pipeline] withCredentials
  Masking supported pattern matches of $AWS_ACCESS_KEY_ID or $AWS_SECRET_ACCESS_KEY
[Pipeline] {
[Pipeline] withEnv
[Pipeline] {
[Pipeline] stage
[Pipeline] { (Checkout SCM)
[Pipeline] script
[Pipeline] {
```

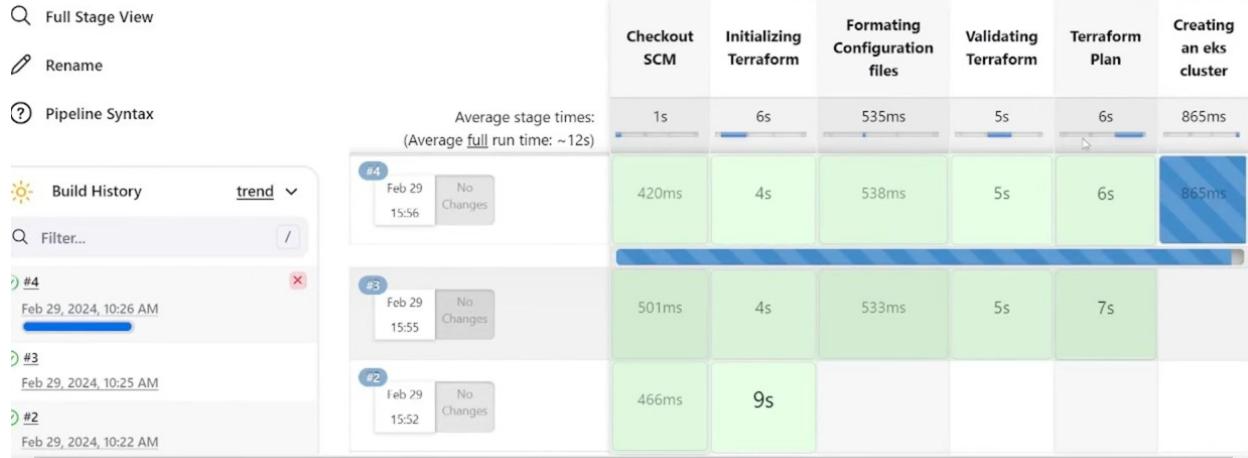
other stages



```
1 pipeline {
8   stages{
33     }
34     stage('Validating Terraform'){
35       steps{
36         script{
37           dir('EKS'){
38             sh 'terraform validate'
39           }
40         }
41       }
42     }
43     stage('Previewing the infrastructure'){
44       steps{
45         script{
46           dir('EKS'){
47             sh 'terraform plan'
48           }
49         }
50       }
51     }
52   }
53   stage('Creating/Destroying an EKS cluster'){
54     steps{
55       script{
56         dir('EKS'){
57           sh 'terraform $action --auto-approve'
58         }
59       }
60     }
61   }
62   stage("Deploying Nginx"){
63     steps{
64       script{
65         dir('EKS/configuration-files'){
66           sh 'aws eks update-kubeconfig --name my-eks-cluster'
67           sh 'kubectl apply -f deployment.yml'
68           sh 'kubectl apply -f service.yml'
69         }
70       }
71     }
72 }
```



```
1 pipeline {
8   stages{
43     stage('Previewing the infrastructure'){
44       steps{
51       }
52     }
53     stage('Creating/Destroying an EKS cluster'){
54       steps{
55         script{
56           dir('EKS'){
57             sh 'terraform $action --auto-approve'
58           }
59         }
60       }
61     }
62     stage("Deploying Nginx"){
63       steps{
64         script{
65           dir('EKS/configuration-files'){
66             sh 'aws eks update-kubeconfig --name my-eks-cluster'
67             sh 'kubectl apply -f deployment.yml'
68             sh 'kubectl apply -f service.yml'
69           }
70         }
71       }
72     }
73     stage("Deploying MySQL"){
74       steps{
75         script{
76           dir('EKS/mysql-deployment'){
77             sh 'aws eks update-kubeconfig --name my-eks-cluster'
78             sh 'kubectl apply -f deployment.yml'
79           }
80         }
81       }
82     }
83   }
84 }
```



Check in Aws EKS Cluster Created ??

aws Services Search [Alt+S] EKS > Clusters

Amazon Elastic Kubernetes Service

Clusters New

Amazon EKS Anywhere

Enterprise Subscriptions New

Related services

Amazon ECR AWS Batch

Documentation ↗ Submit feedback

Clusters (1) Info

Filter clusters

Cluster name	Status	Kubernetes version	Support type	Provider
my-eks-cluster	Creating	1.29	Standard support until March 23, 2025	EKS



EKS Cluster Activated

The screenshot shows the AWS EKS Clusters management interface. On the left, there's a sidebar with links for 'Clusters', 'Amazon EKS Anywhere', 'Related services' (Amazon ECR, AWS Batch), and 'Documentation'. The main area displays a table titled 'Clusters (1) info' with one entry: 'my-eks-cluster' (Status: Active, Kubernetes version: 1.29, Support type: Standard support until March 23, 2025, Provider: EKS). There are buttons for 'Delete' and 'Add cluster'.

To destroy the cluster in Jenkins- Go to configure- General –Project parameterized

The screenshot shows the Jenkins project configuration screen under the 'Configure' section. The 'General' tab is selected. A checkbox labeled 'This project is parameterized' is checked. A dropdown menu titled 'Add Parameter' is open, showing options like 'Boolean Parameter', 'Choice Parameter' (which is highlighted in blue), 'Credentials Parameter', 'File Parameter', 'Multi-line String Parameter', 'Password Parameter', and 'Run Parameter'. Other tabs visible include 'Advanced Project Options' and 'Pipeline'.

Configure

This project is parameterized ?

General

Advanced Project Options

Pipeline

Choice Parameter ?

Name ?
action

Choices ?

apply
destroy

! Requires Choices.

Save Apply

Go to Build with parameter (option – apply /destroy)

Dashboard > terraform_cicd_eks >

Status Changes Build with Parameters Configure Delete Pipeline Full Stage View Build Cancel Rename

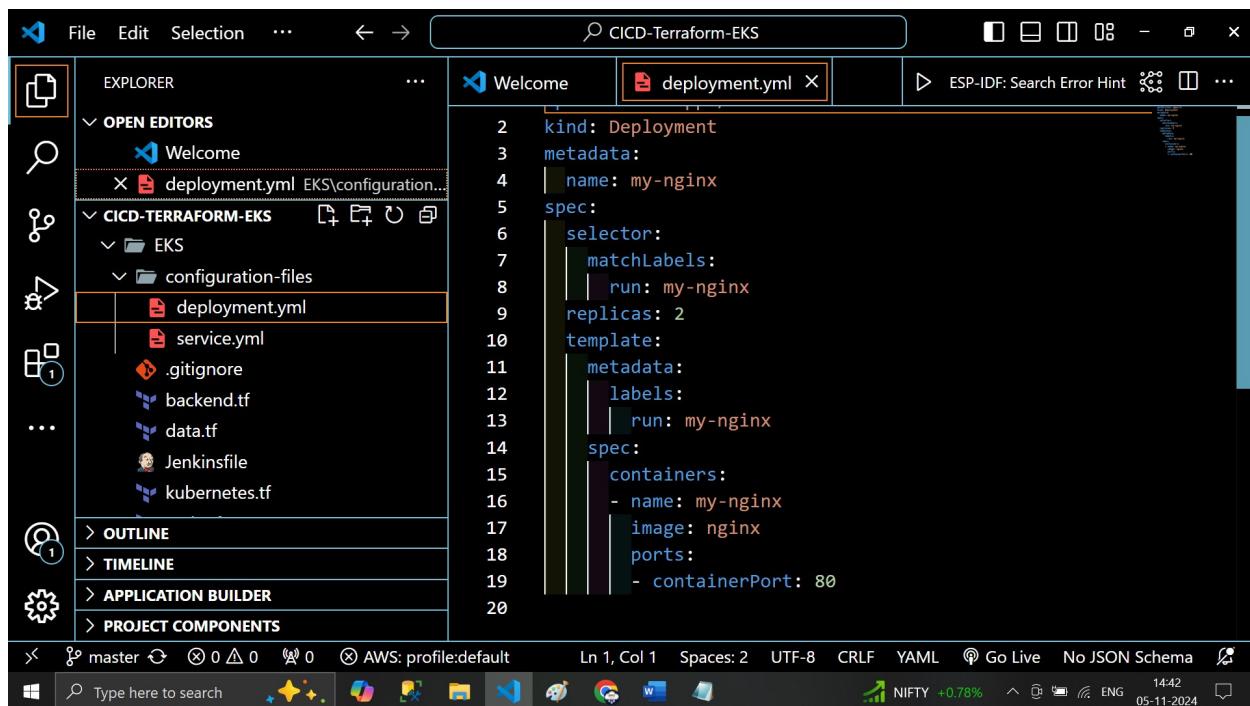
Pipeline terraform_cicd_eks

This build requires parameters:

action
apply

```
stage('Creating/Destroying an EKS cluster'){
    steps{
        script{
            dir('EKS'){
                sh 'terraform $action --auto-approve'
            }
        }
    }
}
```

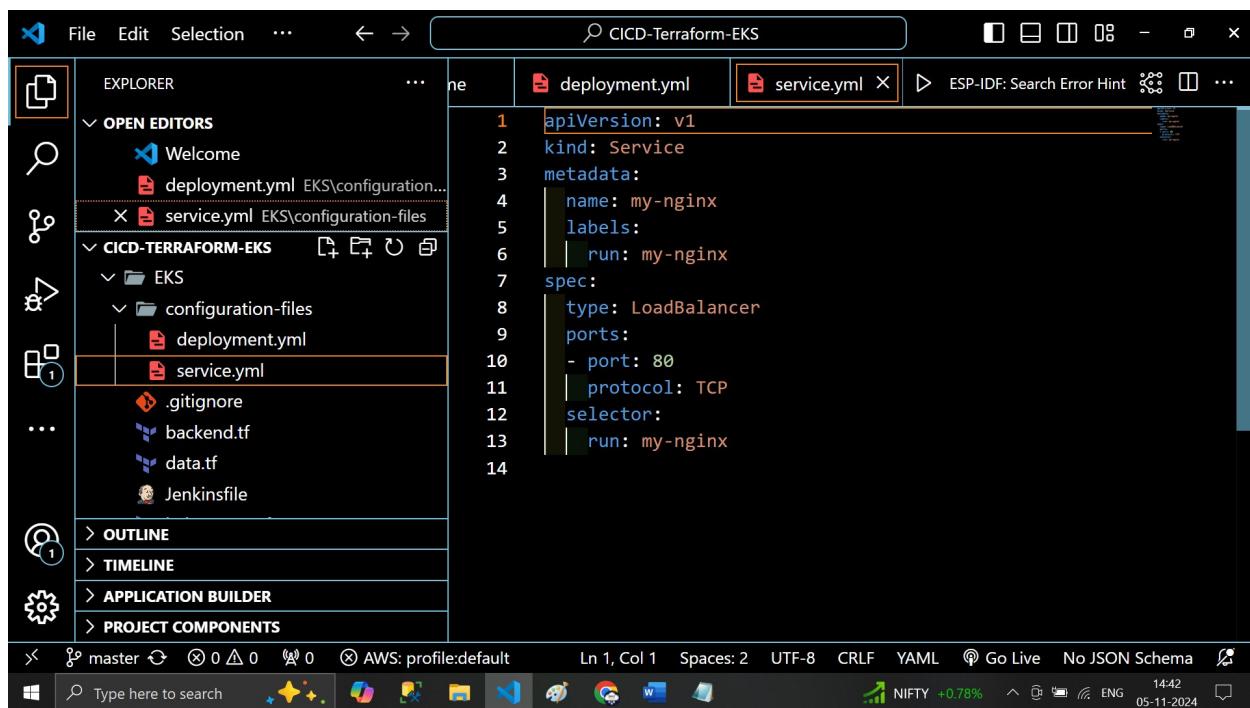
now Deploy Nginx app



The screenshot shows the Visual Studio Code interface with the title bar "CICD-Terraform-EKS". The Explorer sidebar on the left shows a project structure under "CICD-TERRAFORM-EKS/EKS/configuration-files". The "deployment.yml" file is selected and highlighted in orange. The main editor pane displays the YAML content for a Deployment resource:

```
kind: Deployment
metadata:
  name: my-nginx
spec:
  selector:
    matchLabels:
      run: my-nginx
  replicas: 2
  template:
    metadata:
      labels:
        run: my-nginx
    spec:
      containers:
        - name: my-nginx
          image: nginx
          ports:
            - containerPort: 80
```

The status bar at the bottom shows "AWS: profile:default", "NIFTY +0.78%", and the date "05-11-2024".



The screenshot shows the Visual Studio Code interface with the title bar "CICD-Terraform-EKS". The Explorer sidebar on the left shows a project structure under "CICD-TERRAFORM-EKS/EKS/configuration-files". The "service.yml" file is selected and highlighted in orange. The main editor pane displays the YAML content for a Service resource:

```
apiVersion: v1
kind: Service
metadata:
  name: my-nginx
  labels:
    run: my-nginx
spec:
  type: LoadBalancer
  ports:
    - port: 80
      protocol: TCP
  selector:
    run: my-nginx
```

The status bar at the bottom shows "AWS: profile:default", "NIFTY +0.78%", and the date "05-11-2024".

Push the directory to Git repo

```

$ cd EKS/
LAPTOP-JN110945 MINGW64 ~/OneDrive/Desktop/Terraform_EKS/EKS (master)
$ git status
On branch master
Untracked files:
  (use "git add <file>..." to include in what will be committed)
    configuration-files/
nothing added to commit but untracked files present (use "git add" to
LAPTOP-JN110945 MINGW64 ~/OneDrive/Desktop/Terraform_EKS/EKS (master)
$ git add configuration-files/
r@LAPTOP-JN110945 MINGW64 ~/OneDrive/Desktop/Terraform_EKS/EKS (master)
$ git commit -m "configuration file is added"
[master 050c68c] configuration file is added
 2 files changed, 35 insertions(+)
 create mode 100644 EKS/configuration-files/deployment.yml
 create mode 100644 EKS/configuration-files/service.yml
LAPTOP-JN110945 MINGW64 ~/OneDrive/Desktop/Terraform_EKS/EKS (master)
$ git push origin master

```

The screenshot shows a GitHub repository page for 'Deploy-EKS-Cluster-using-Terraform--Automate-Jenkins--AWS-EC2'. The 'Code' tab is selected, showing a list of files in the 'main' branch. The commit history shows a single commit from 'Terraform using Jenkins' with the message 'configuration file is added'. The commit was made 1 hour ago.

Name	Last commit message	Last commit date
..		1 hour ago
configuration-files	Terraform using Jenkins	1 hour ago
Jenkinsfile	Terraform using Jenkins	1 hour ago
backend.tf	Terraform using Jenkins	1 hour ago
data.tf	Terraform using Jenkins	1 hour ago
kubernetes.tf	Terraform using Jenkins	1 hour ago
main.tf	Terraform using Jenkins	1 hour ago
provider.tf	Terraform using Jenkins	1 hour ago
terraform.tfvars	Terraform using Jenkins	1 hour ago
variables.tf		1 hour ago

Add one more stage

Status

terraform_cicd_eks

Changes

Build with Parameters

Configure

Delete Pipeline

Full Stage View

Rename

Pipeline Syntax

Average stage times:
(Average full run time: ~2min 48s)

Checkout SCM	Initializing Terraform	Formatting Configuration files	Validating Terraform	Terraform Plan	Creating an eks cluster
1s	5s	535ms	5s	7s	3min

#7

```

}
stage("Deploying Nginx"){
    steps{
        script{
            dir('EKS/configuration-files'){
                sh 'aws eks update-kubeconfig --name my-eks-cluster'
                sh 'kubectl apply -f deployment.yml'
                sh 'kubectl apply -f service.yml'
            }
        }
    }
}

```

Error

user Does not have API Permission to deploy this app on EKS cluster

Stage View

Average stage times:
(Average full run time: ~2min 48s)

	Checkout SCM	Initializing Terraform	Formatting Configuration files	Validating Terraform	Terraform Plan	Creating an eks cluster	Deploying Nginx
#8 Feb 29 16:32 1 commit	777ms	4s	531ms	5s	7s (paused for 11s)	7s	648ms failed
#7 Feb 29 16:16 No Changes	424ms	4s	538ms	5s	7s (paused for 11s) aborted	127ms aborted	
#6							

in EKS main folder – main.tf I have made some Changes

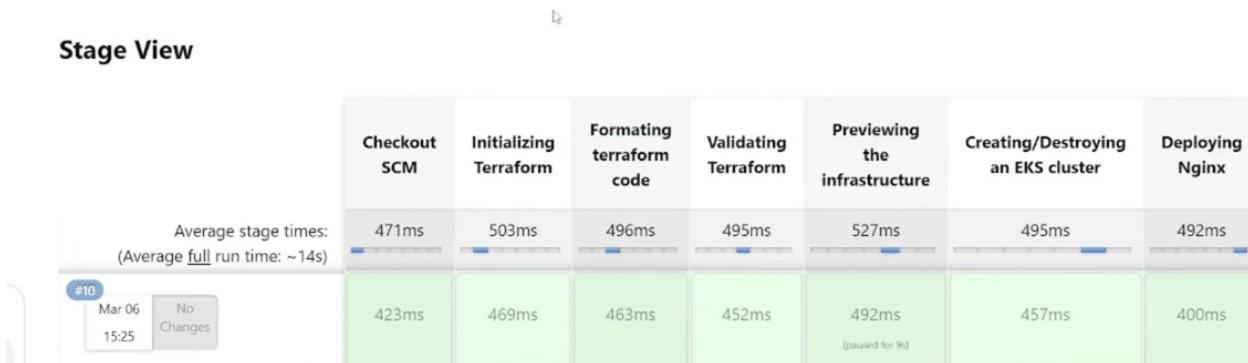
The screenshot shows a code editor interface with a sidebar and a main editor area. The sidebar on the left lists a project structure:

- OPEN EDITORS: Welcome, main.tf EKS
- CICD-TERRAFORM-EKS:
 - EKS:
 - configuration-files:
 - .gitignore
 - backend.tf
 - data.tf
 - Jenkinsfile
 - kubernetes.tf
 - main.tf
 - provider.tf
 - terraform.tfvars
 - variables.tf
 - jenkins-server
 - README.md
 - terraform.tfstate

The main editor area displays the content of the main.tf file:

```
34 module "eks" {  
42   eks_managed_node_groups = {  
49     }  
50     tags = {  
51       Environment = "dev"  
52       Terraform   = "true"  
53     }  
54   }  
55  
56   data "aws_eks_cluster" "cluster" {  
57     name = module.eks.cluster_id  
58   }  
59  
60   data "aws_eks_cluster_auth" "cluster" {  
61     name = module.eks.cluster_id  
62   }  
63 }
```

Now Build with parameter – apply



Check in aws Load balancer – cp the link & paste it in Browser

Network & Security

- Security Groups
- Elastic IPs
- Placement Groups
- Key Pairs
- Network Interfaces

Load Balancing

Load Balancers

- Target Groups
- Trust Stores [New](#)

Auto Scaling

- Auto Scaling Groups

Load balancers (1)

Elastic Load Balancing scales your load balancer capacity automatically in response to changes in incoming traffic.

Name	DNS name	Status	VPC ID
a83938684dd96420f8...	a83938684dd96420f89ee...	-	vpc-029b86eda1

0 load balancers selected

Select a load balancer above.

cp dns

Network & Security

- Security Groups
- Elastic IPs
- Placement Groups
- Key Pairs
- Network Interfaces

Load Balancing

Load Balancers

- Target Groups
- Trust Stores [New](#)

Auto Scaling

- Auto Scaling Groups

a83938684dd96420f89eea27055114c4

Details

Load balancer type	Status	VPC	Date created
Classic	Loading...	vpc-029b86eda16924f97	March 6, 2024, 15:19 (UTC+05:30)
Scheme	Hosted zone	Z3AADJGX6KTTL2	
Internet-facing			

DNS name [Info](#)

[a83938684dd96420f89eea27055114c4-1933308030.us-east-2.elb.amazonaws.com \(A Record\)](#)

h

Elastic IPs

Placement Groups

Key Pairs

Network Interfaces

Load Balancing

Load Balancers

Target Groups

Trust Stores [New](#)

Auto Scaling

Auto Scaling Groups

DNS name copied

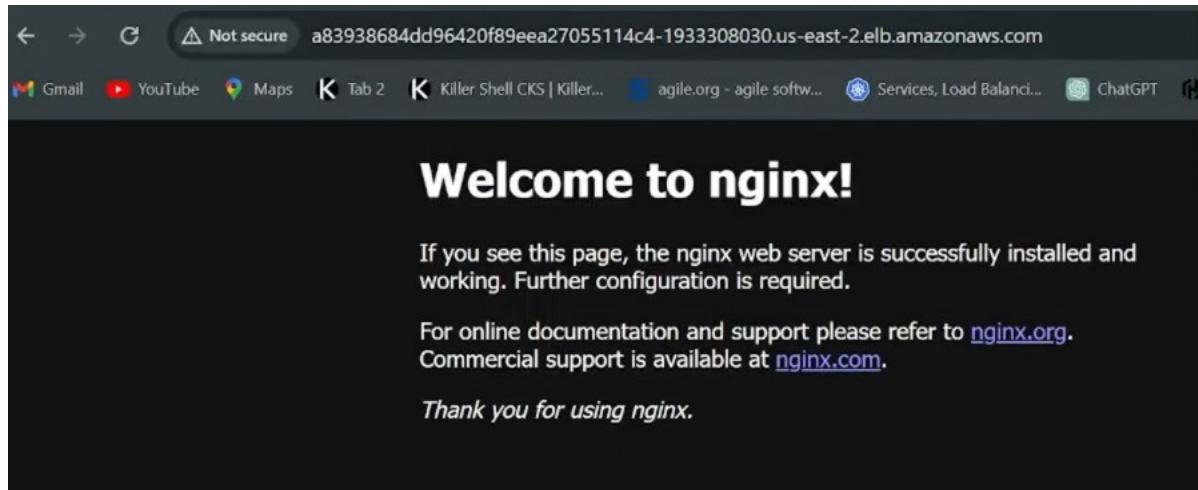
[a83938684dd96420f89eea27055114c4-1933308030.us-east-2.elb.amazonaws.com \(A Record\)](#)

This Classic Load Balancer can be migrated to a next generation load balancer. Migration wizard uses your load balancer's current configurations to create a new load balancer. [Learn more](#)

Distribution of targets by Availability Zone (AZ)

For each enabled Availability Zone, you can view the number of registered instances and their current health states. Selecting any values here will apply the corresponding filter to the Target Instances table.

final O/P



Welcome to nginx!

If you see this page, the nginx web server is successfully installed and working. Further configuration is required.

For online documentation and support please refer to nginx.org.
Commercial support is available at nginx.com.

Thank you for using nginx.