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.

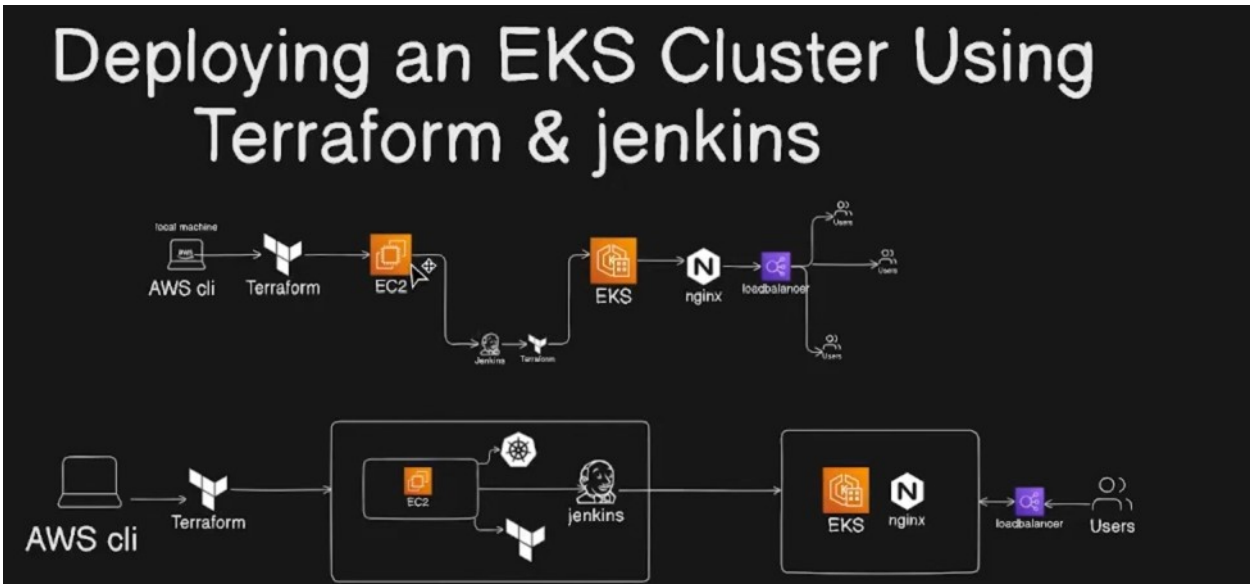
2. Utilize Terraform to define infrastructure as code, configuring the EKS cluster effortlessly.

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

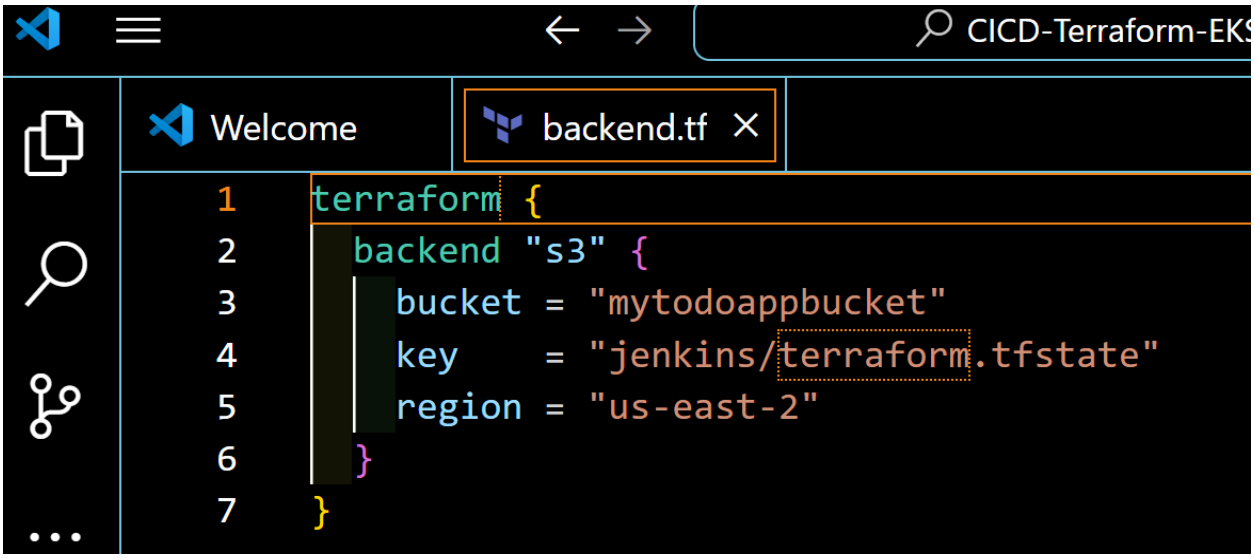


 # Pre-Requisites

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



 # tf init

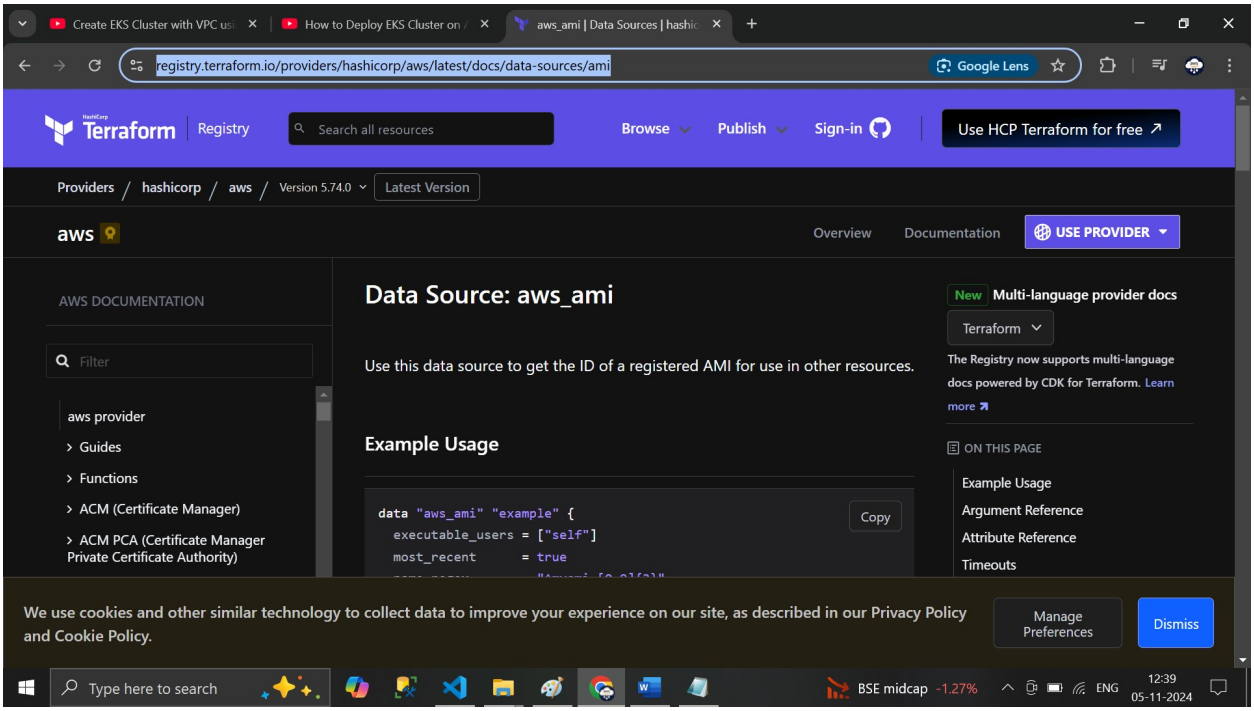
```
$ terraform init

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.
```

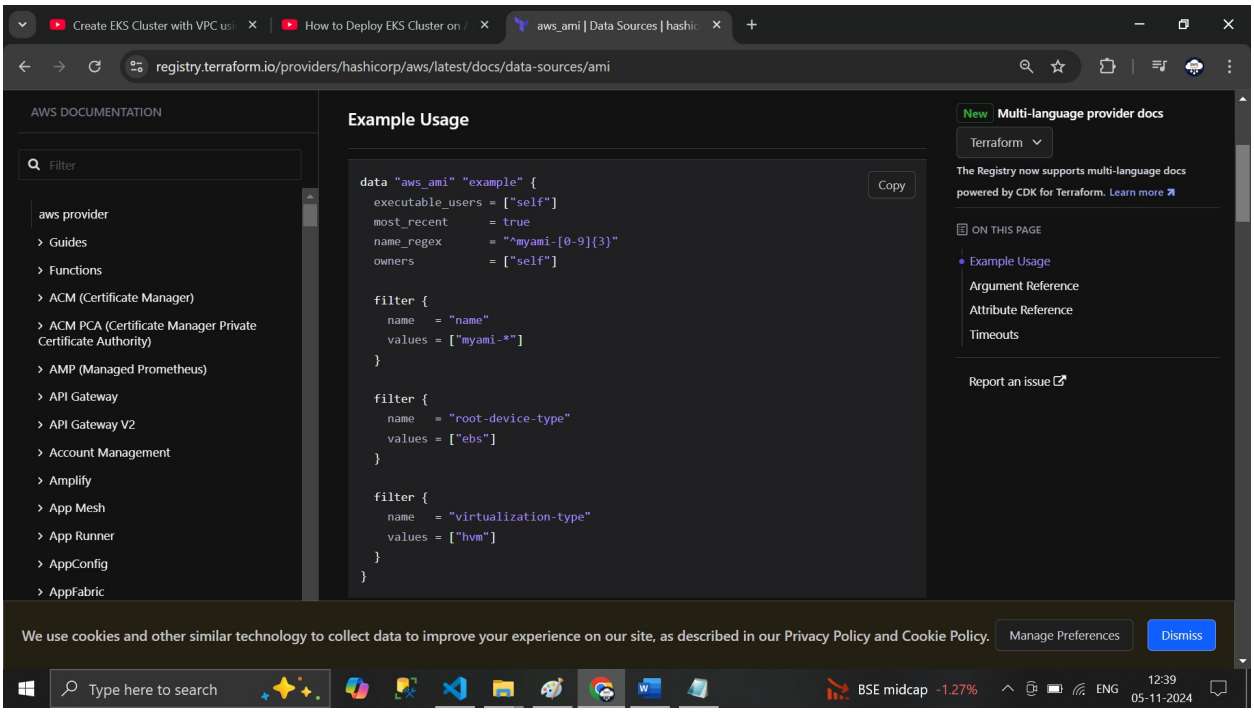
```
$ terraform fmt
backend.tf

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

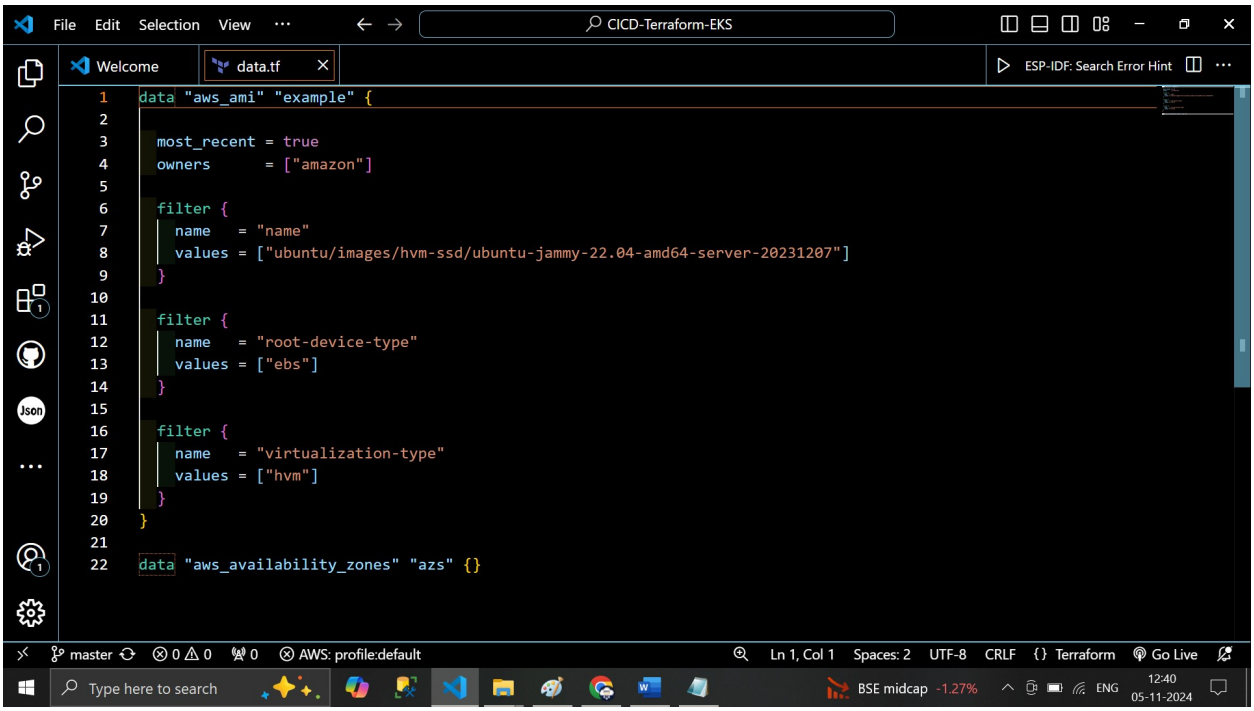
```
$ terraform plan

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

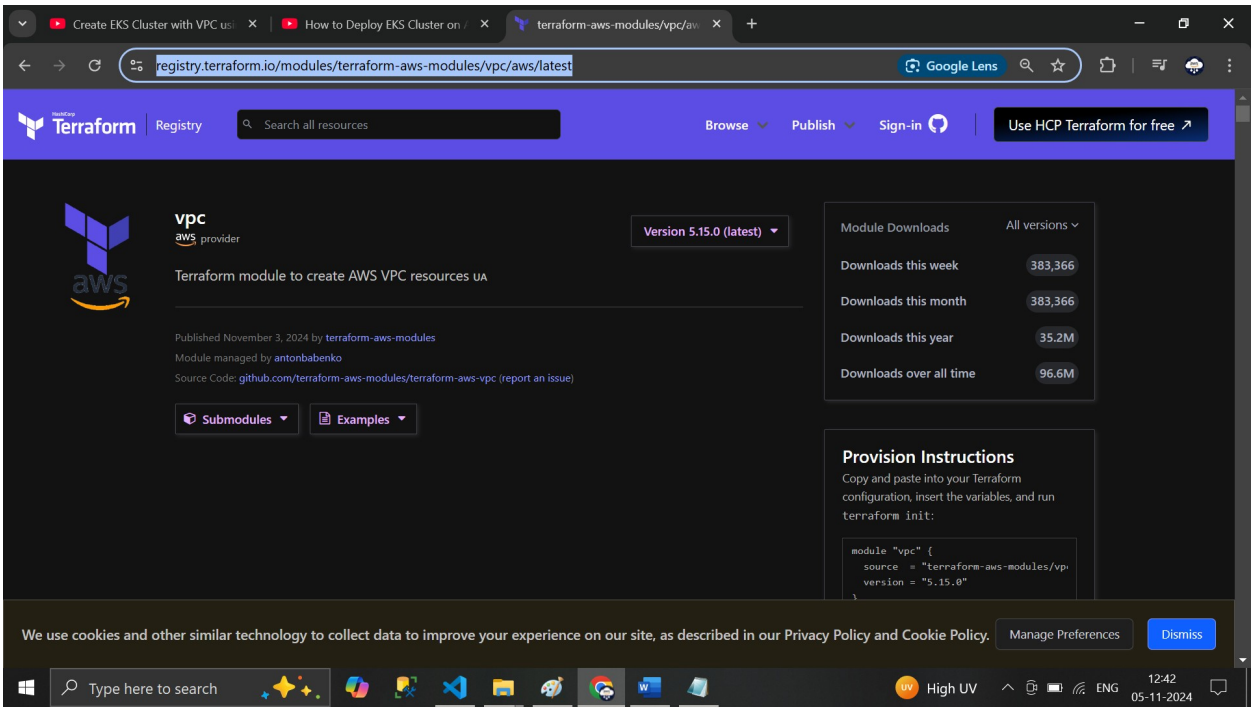




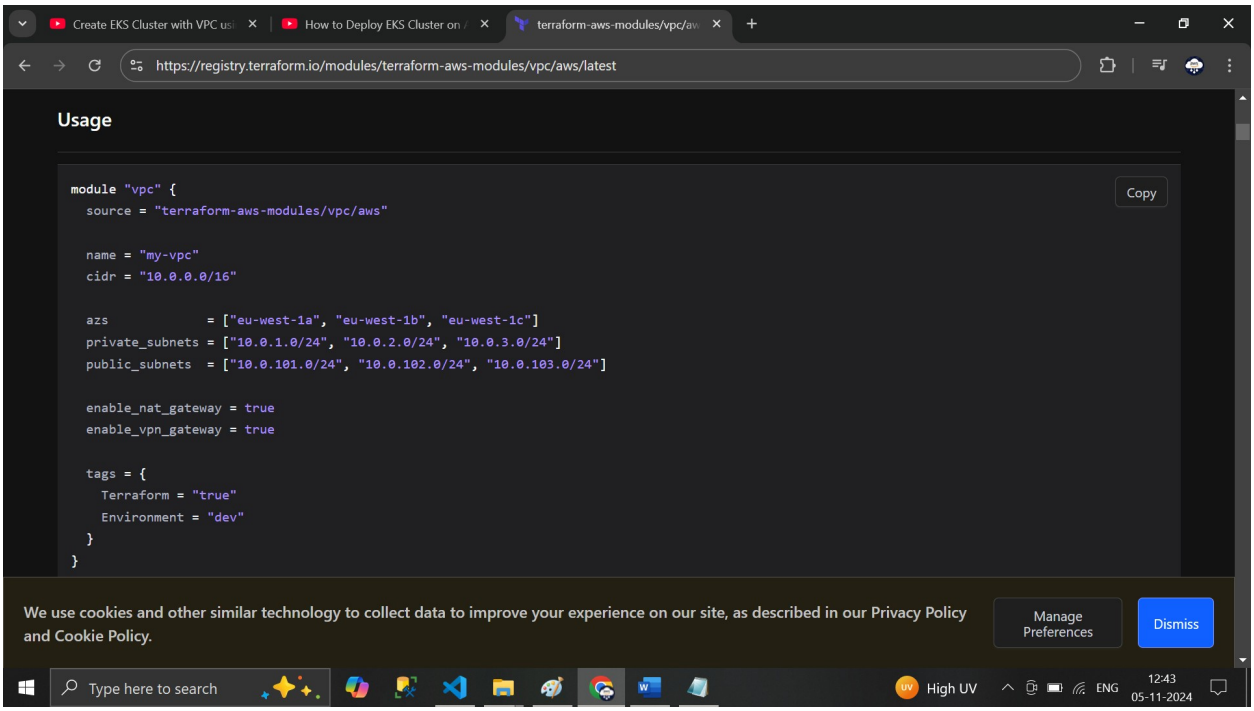


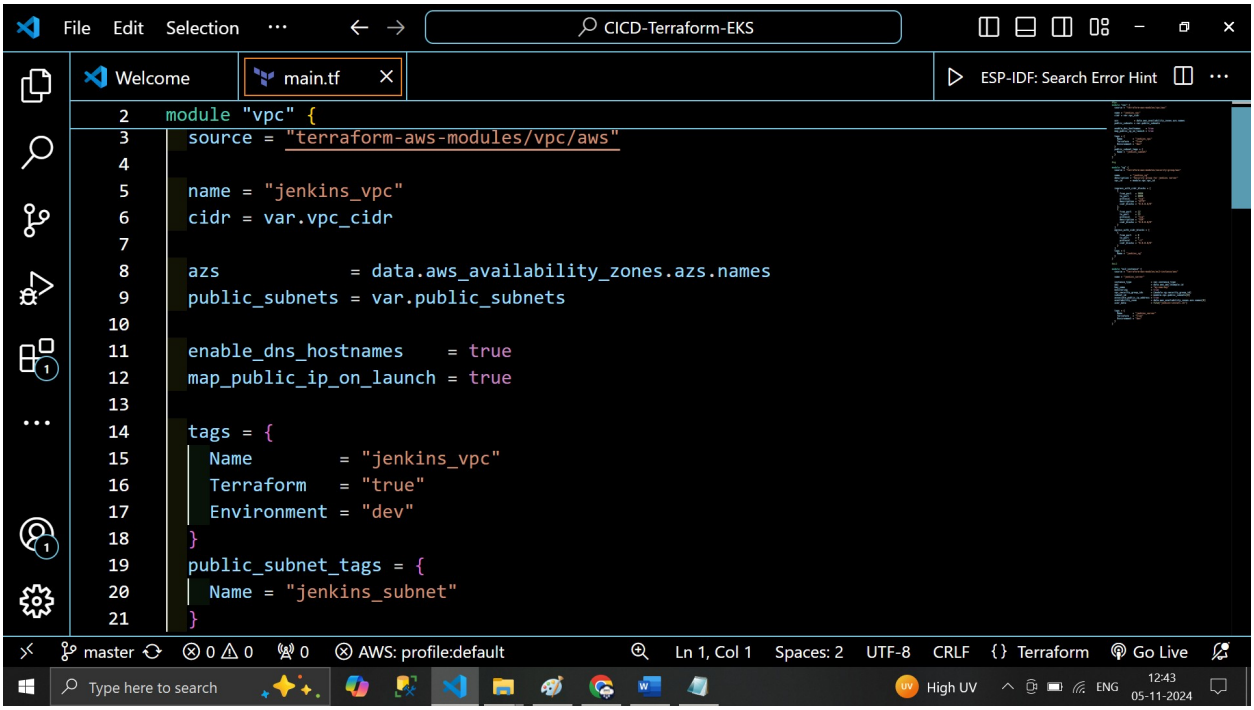
 # Go to tf doc 4 VPC module

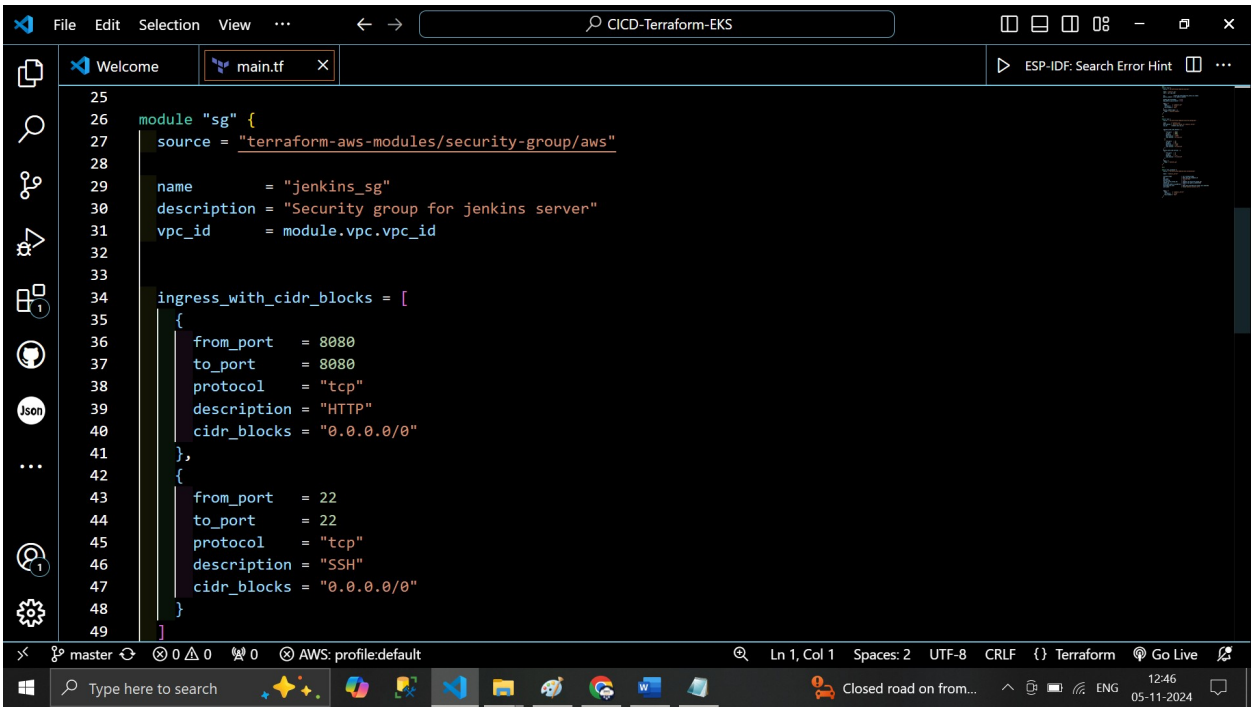
https://registry.terraform.io/modules/terraform-aws-modules/vpc/aws/latest

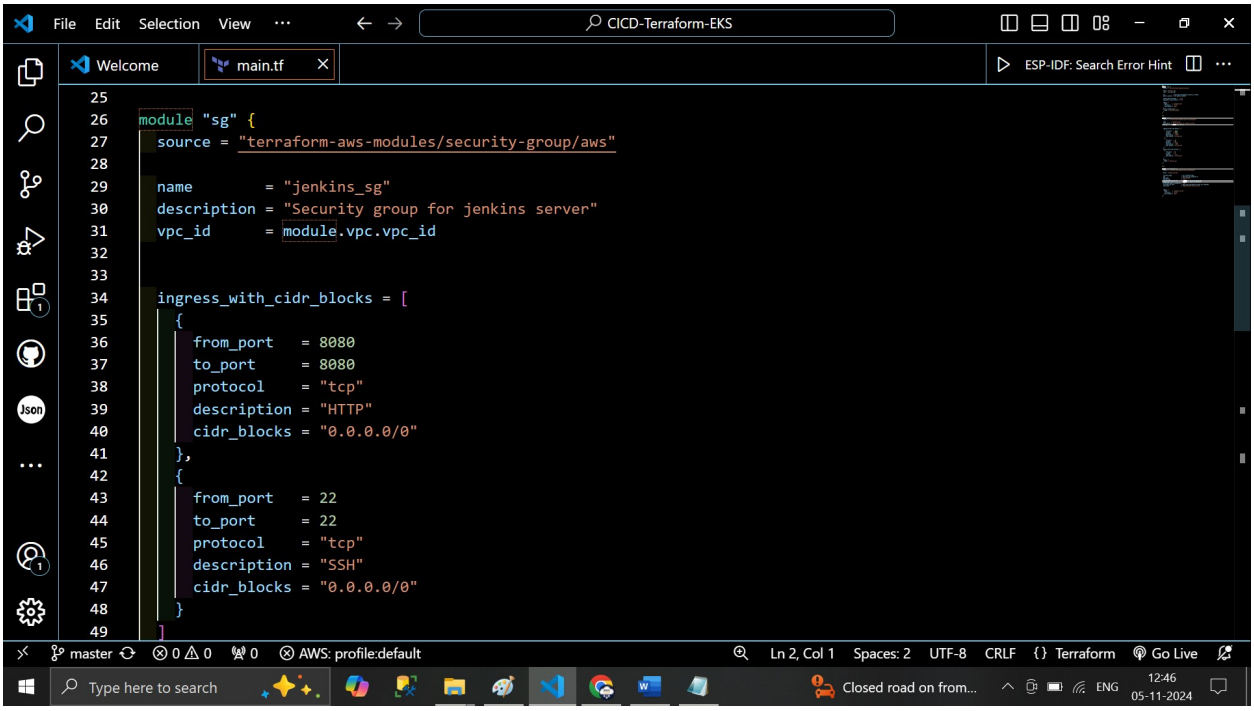


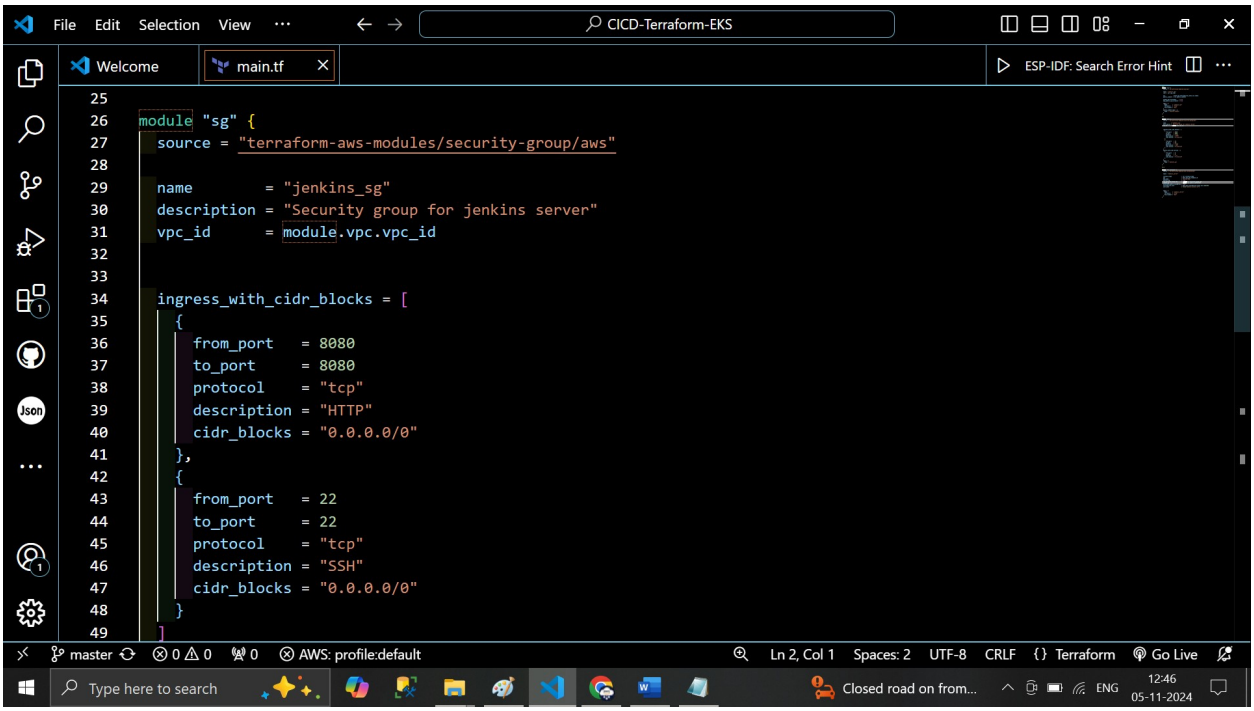
 # cp


Create EKS Cluster with VPC usi
How to Deploy EKS Cluster on
terraform-aws-modules/vpc/aw
https://registry.terraform.io/modules/terraform-aws-modules/vpc/aws/latest
Usage
Copy
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"
}
}
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
Type here to search
High UV
ENG
12:43
05-11-2024


File
Edit
Selection
CICD-Terraform-EKS
Welcome
main.tf
ESP-IDF: Search Error Hint
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 }
master
0 0
0
AWS: profile:default
Ln 1, Col 1
Spaces: 2
UTF-8
CRLF
Terraform
Go Live
Type here to search
High UV
ENG
12:43
05-11-2024


File Edit Selection View ···
CICD-Terraform-EKS
Welcome
main.tf
ESP-IDF: Search Error Hint
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"
}
]
master
AWS: profile:default
Ln 1, Col 1
Spaces: 2
UTF-8
CRLF
Terraform
Go Live
Type here to search
Closed road on from...
ENG
12:46
05-11-2024


File Edit Selection View ···
CICD-Terraform-EKS
Welcome
main.tf
ESP-IDF: Search Error Hint
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"
}
]
master
AWS: profile:default
Ln 2, Col 1
Spaces: 2
UTF-8
CRLF
Terraform
Go Live
Type here to search
Closed road on from...
ENG
12:46
05-11-2024


File Edit Selection View
CICD-Terraform-EKS
Welcome
main.tf
ESP-IDF: Search Error Hint
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
Ln 2, Col 1
Spaces: 2
UTF-8
CRLF
Terraform
Go Live
Type here to search
Closed road on from...
ENG
12:46
05-11-2024


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 = "jankins_server"
83 Terraform = "true"
84 Environment = "dev"
85 }
86 }
master
AWS: profile:default
Ln 2, Col 1
Spaces: 2
UTF-8
CRLF
Terraform
Go Live
Type here to search
Closed road on from...
ENG
12:47
05-11-2024

Welcome | variables.tf

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

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

variable "instance_type" {
  description = "Instance Type"
  type        = string
}
```

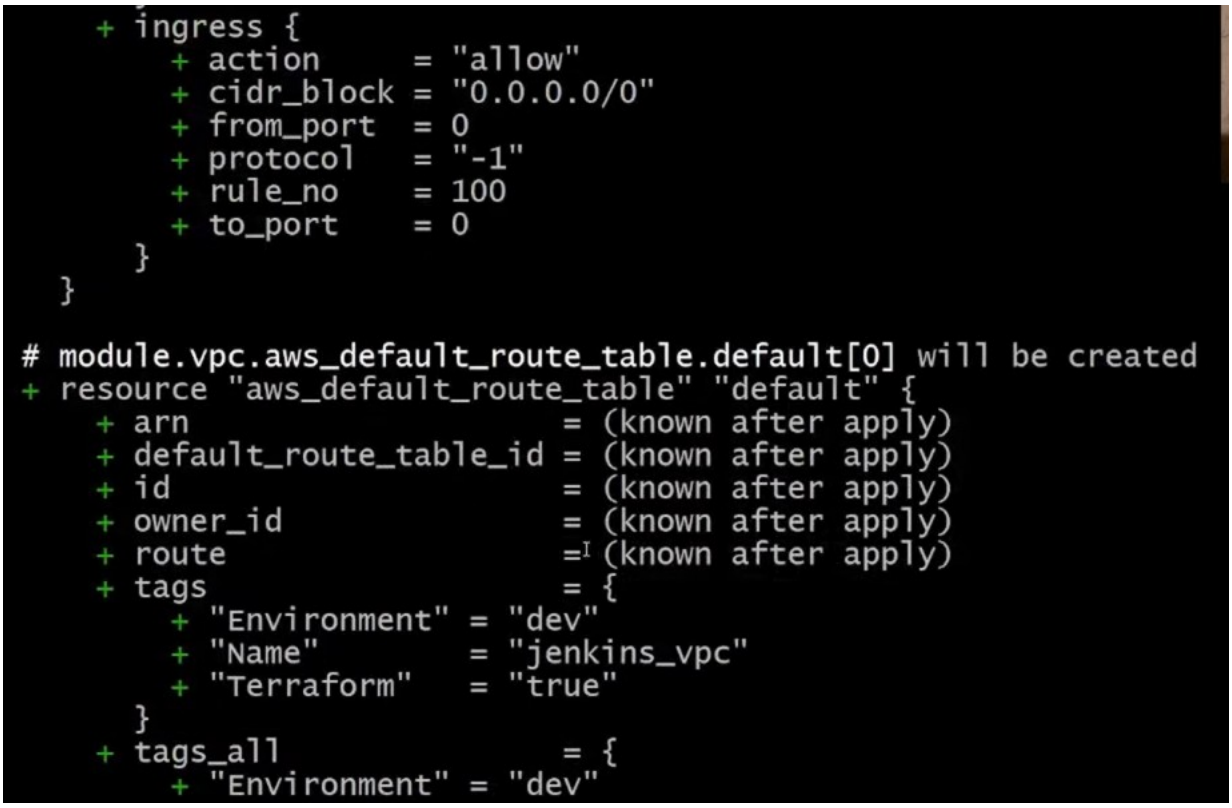
 # tf init

```
$ 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.
```

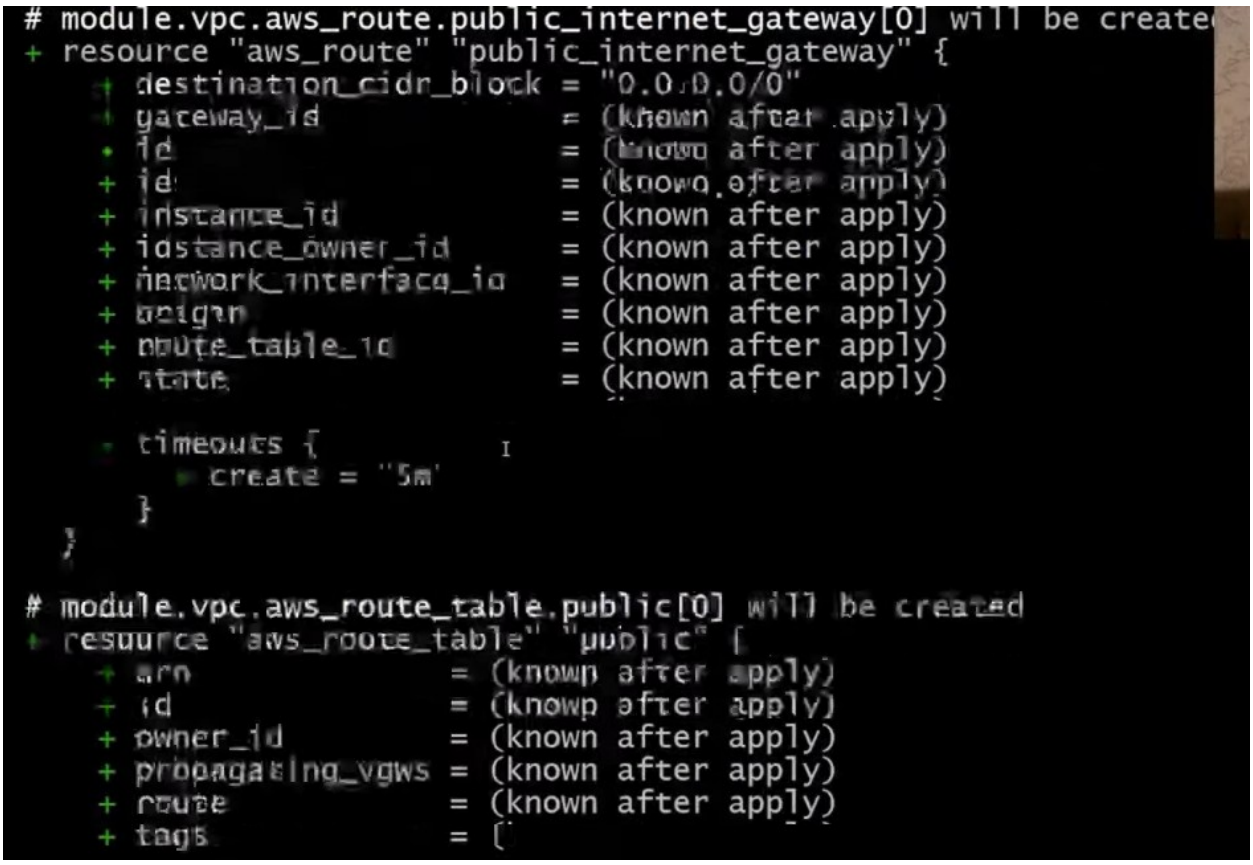
 # tf plan

```
$ 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"
```

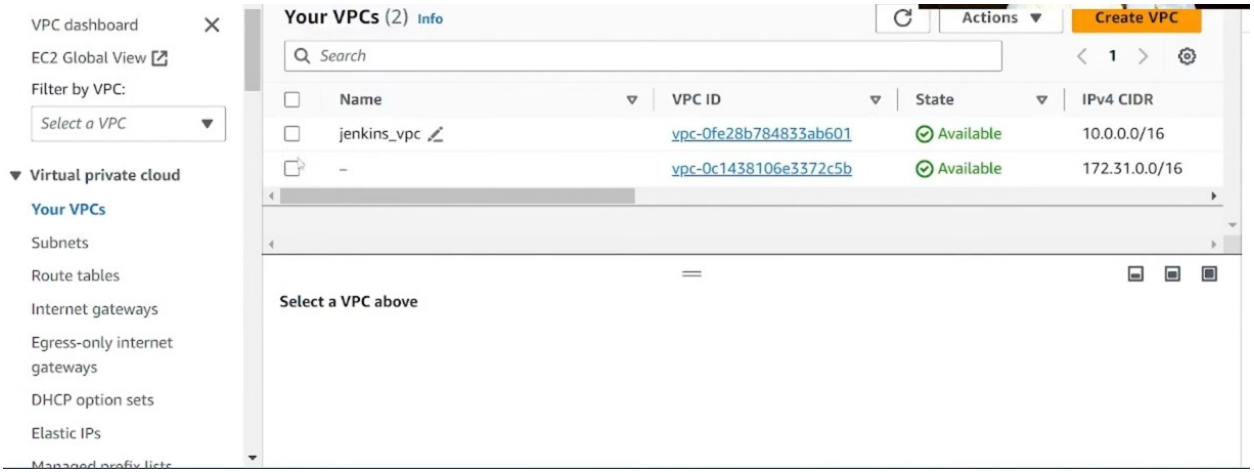


```
# module.vpc.aws_route.public_internet_gateway[0] will be create
+ resource "aws_route" "public_internet_gateway" {
    + destination_cidr_block = "0.0.0.0/0"
    + gateway_id             = (known after apply)
    + id                     = (known after apply)
    + instance_id            = (known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

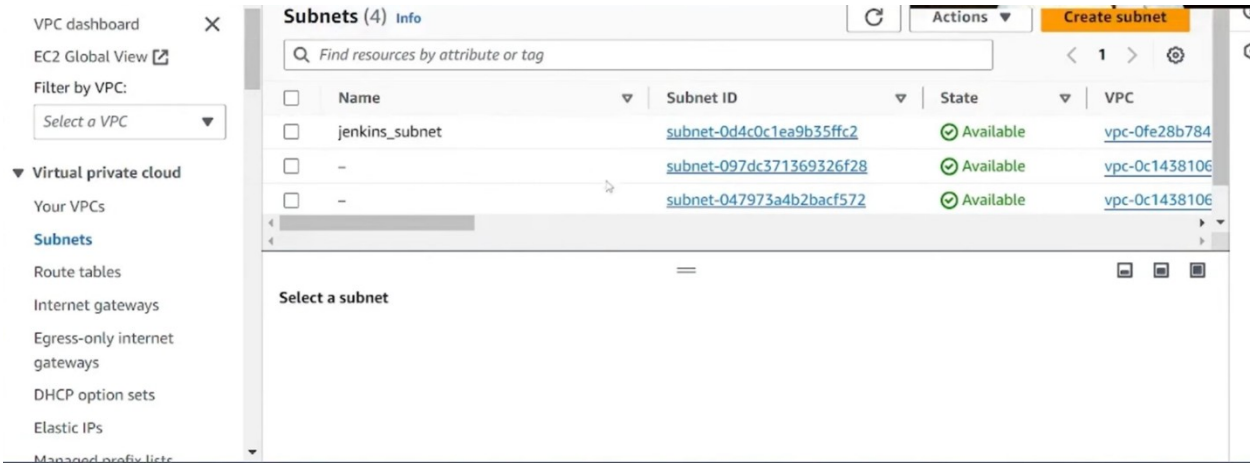
```
$ terraform apply --auto-approve
```

```
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 a
526789eb4]
module.vpc.aws_internet_gateway.this[0]: Creation complete after 2
795]
module.vpc.aws_route_table.public[0]: Creation complete after 2s [id=rtb-09a9fbb9d03a71cee
]
module.vpc.aws_route.public_internet_gateway[0]: Creating...
module.vpc.aws_default_security_group.this[0]: Creation complete after 3s [id=sg-02c076e0d
6252506d]
module.vpc.aws_route.public_internet_gateway[0]: Creation complete after 2s [id=r-rtb-09a9
fbb9d03a71cee1080289494]
module.vpc.aws_default_network_acl.this[0]: Creation complete after 4s [id=acl-0895d359c7d
e6010a]
module.vpc.aws_subnet.public[0]: Still creating... [10s elapsed]
module.vpc.aws_subnet.public[0]: Creation complete after 13s [id=subnet-0d4c0c1ea9b35ffc2]
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

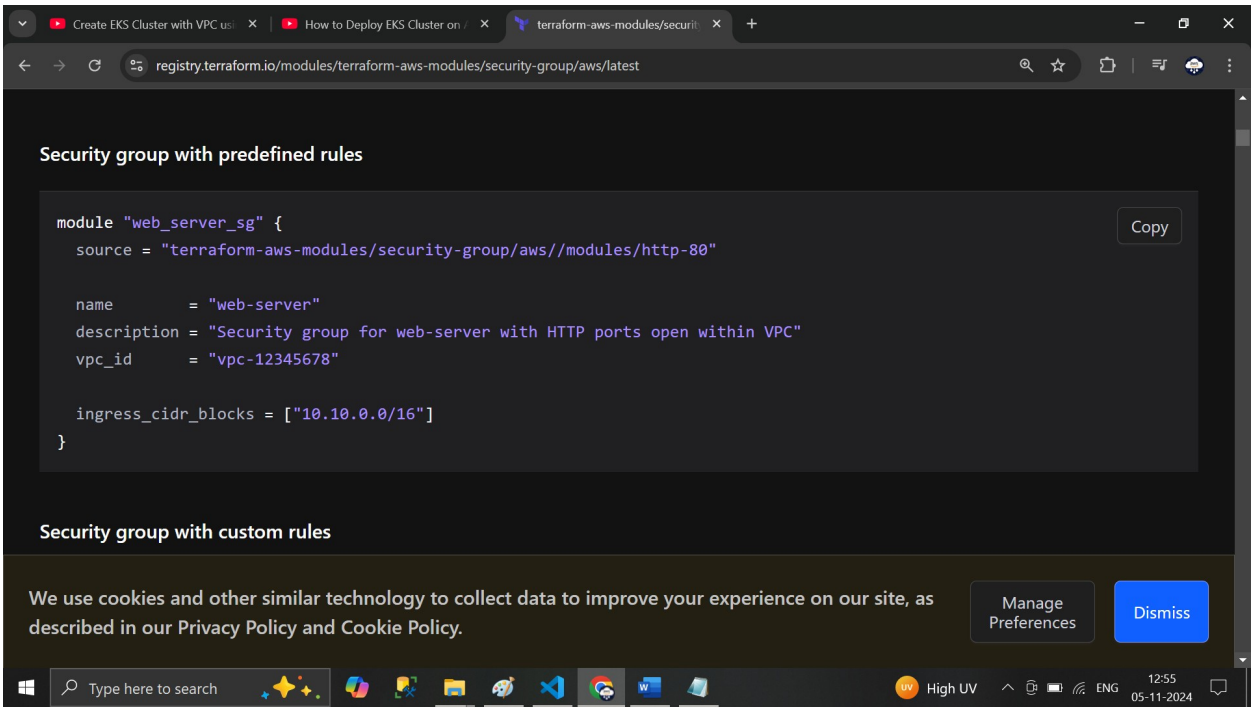


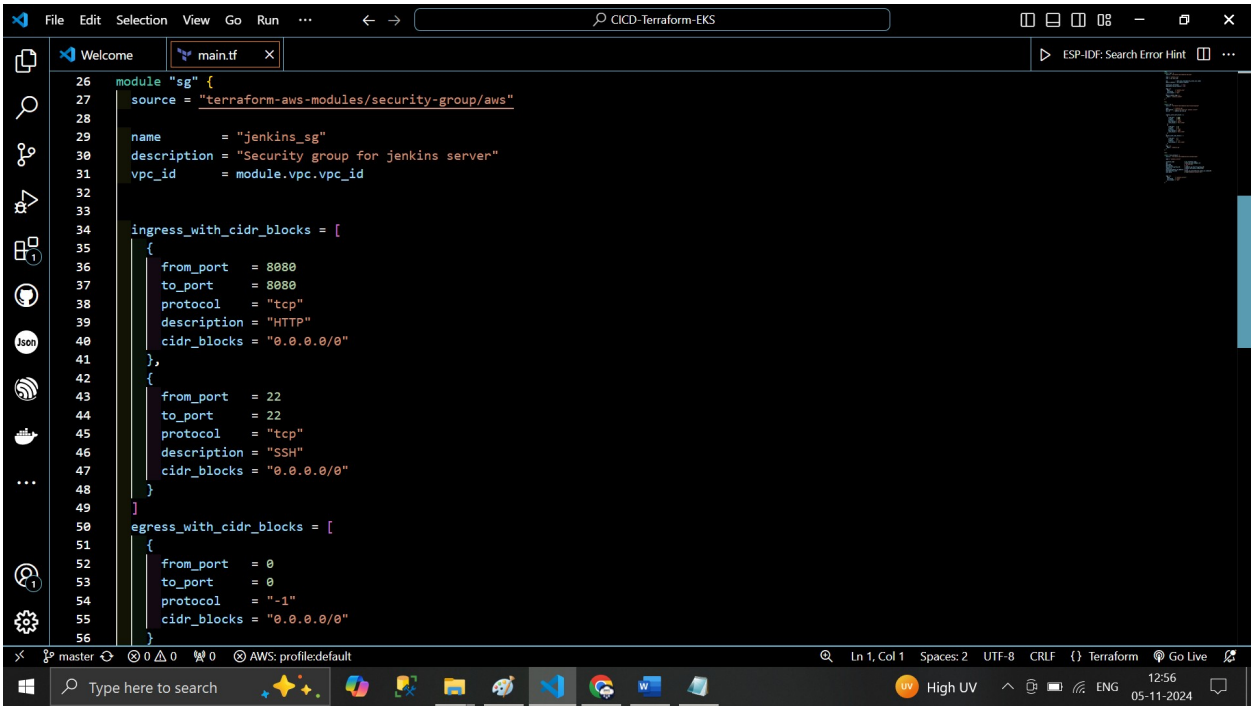
 # Subnets



 # IG

 # Go to security group cp


Create EKS Cluster with VPC usi
How to Deploy EKS Cluster on
terraform-aws-modules/securit
registry.terraform.io/modules/terraform-aws-modules/security-group/aws/latest
Security group with predefined rules
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"]
}
Copy
Security group with custom rules
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
Type here to search
High UV
ENG
12:55
05-11-2024


File Edit Selection View Go Run
CICD-Terraform-EKS
Welcome
main.tf
ESP-IDF: Search Error Hint
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"
}
master
AWS: profile:default
Ln 1, Col 1
Spaces: 2
UTF-8
CRLF
Terraform
Go Live
Type here to search
High UV
ENG
12:56
05-11-2024

 # 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.
```

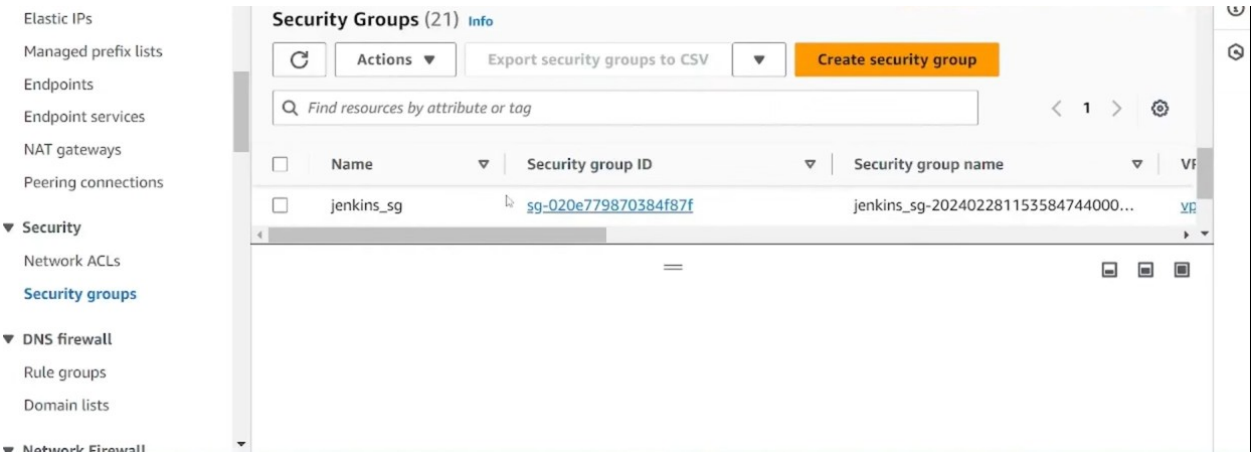
```
$ terraform fmt
main.tf

                    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-03e02fea526789e
b4]
module.vpc.aws_default_security_group.this[0]: Refreshing state... [id=sg-02c076e0d6252506
d]
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-09a9fbb9d03
a71cee1080289494]
module.vpc.aws_route_table_association.public[0]: Refreshing state... [id=rtbassoc-03fd6f9
3e16c3064c]
```

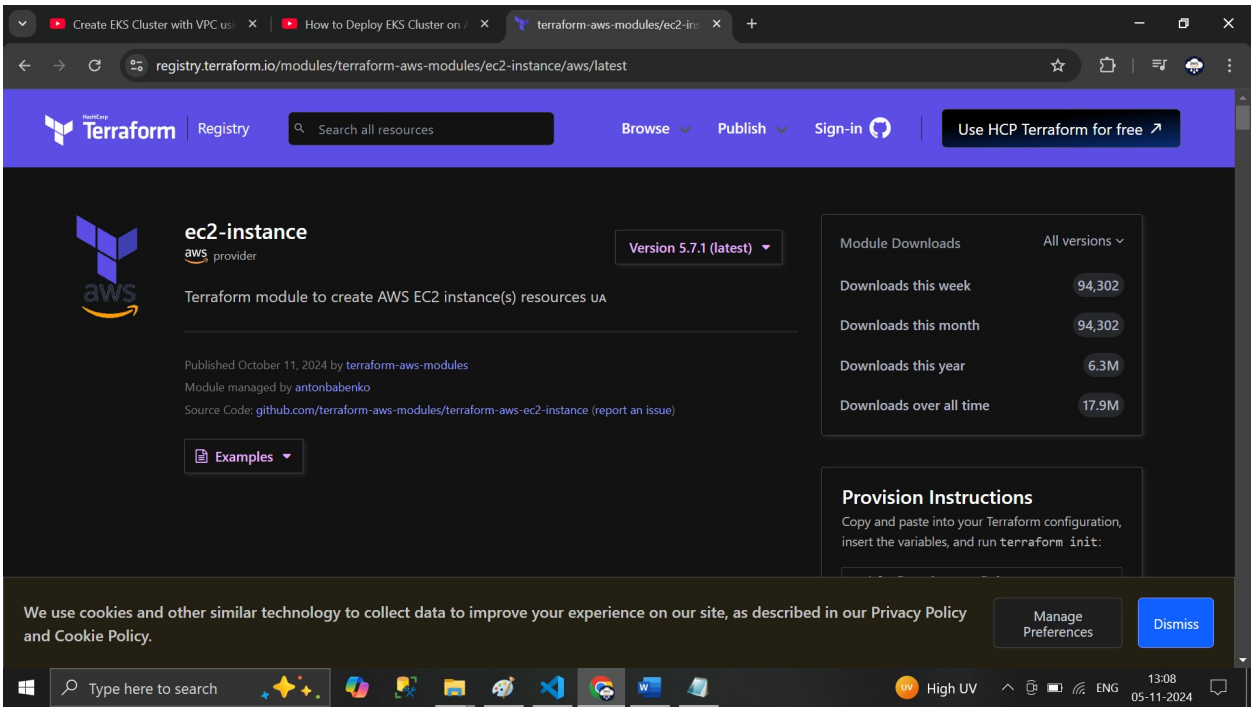
```
  # module.web_server_sg.module.sg.aws_security_group_rule.ingress
eated
  + resource "aws_security_group_rule" "ingress_with_self" {
      + description              = "Ingress Rule"
      + from_port                = -1
      + id                       = (known after apply)
      + prefix_list_ids          = []
      + 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-03e02fea526789e
b4]
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-02c076e0d6252506
d]
module.vpc.aws_subnet.public[0]: Refreshing state... [id=subnet-0d4c0c1ea9b35ffc2]
module.vpc.aws_route.public_internet_gateway[0]: Refreshing state... [id=r-rtb-09a9fbb9d03
a71cee1080289494]
module.vpc.aws_route_table_association.public[0]: Refreshing state... [id=rtbassoc-03fd6f9
3e16c3064c]
```

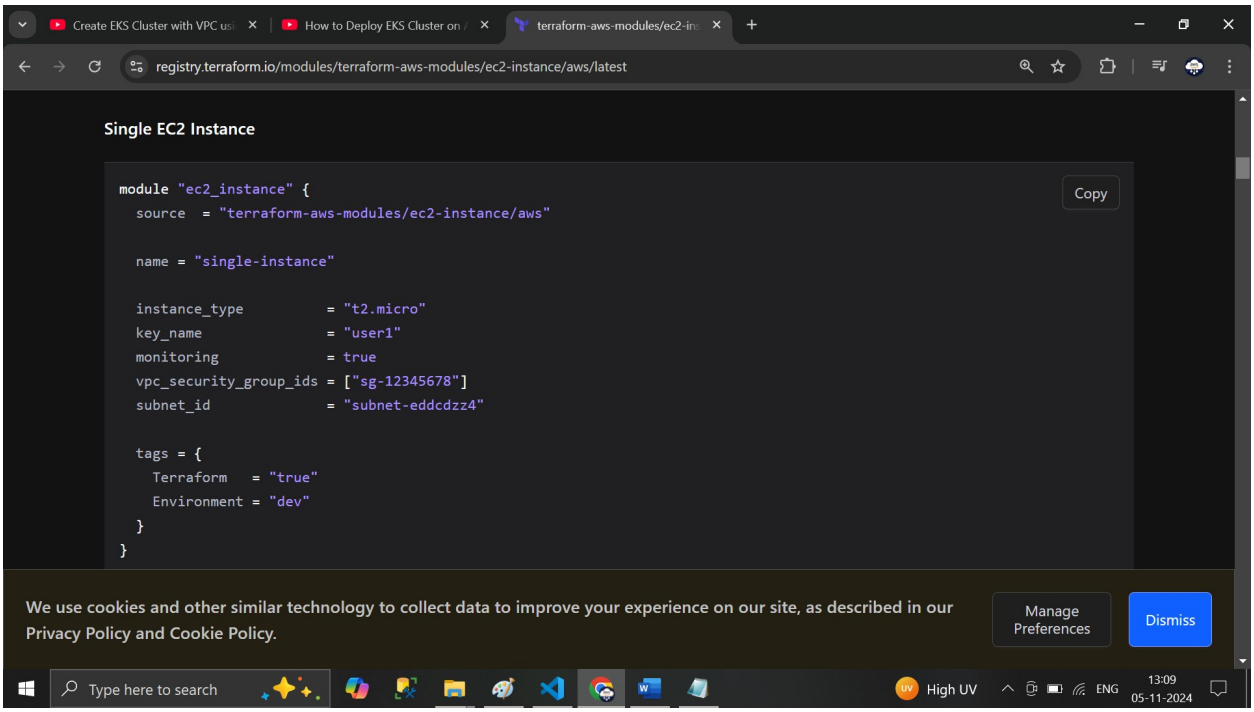
 # SG in aws created



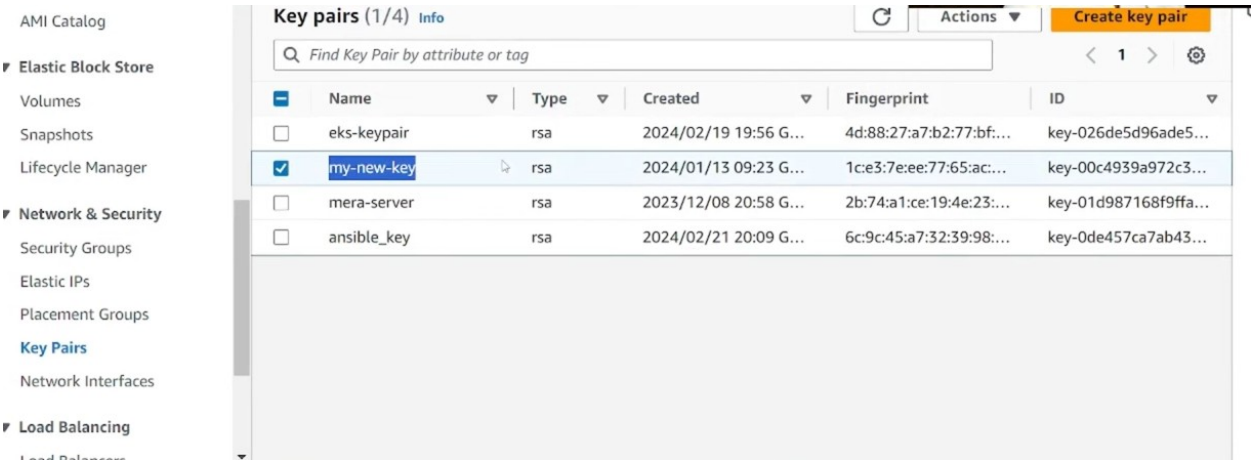
 # Now Created AWS EC2 Module

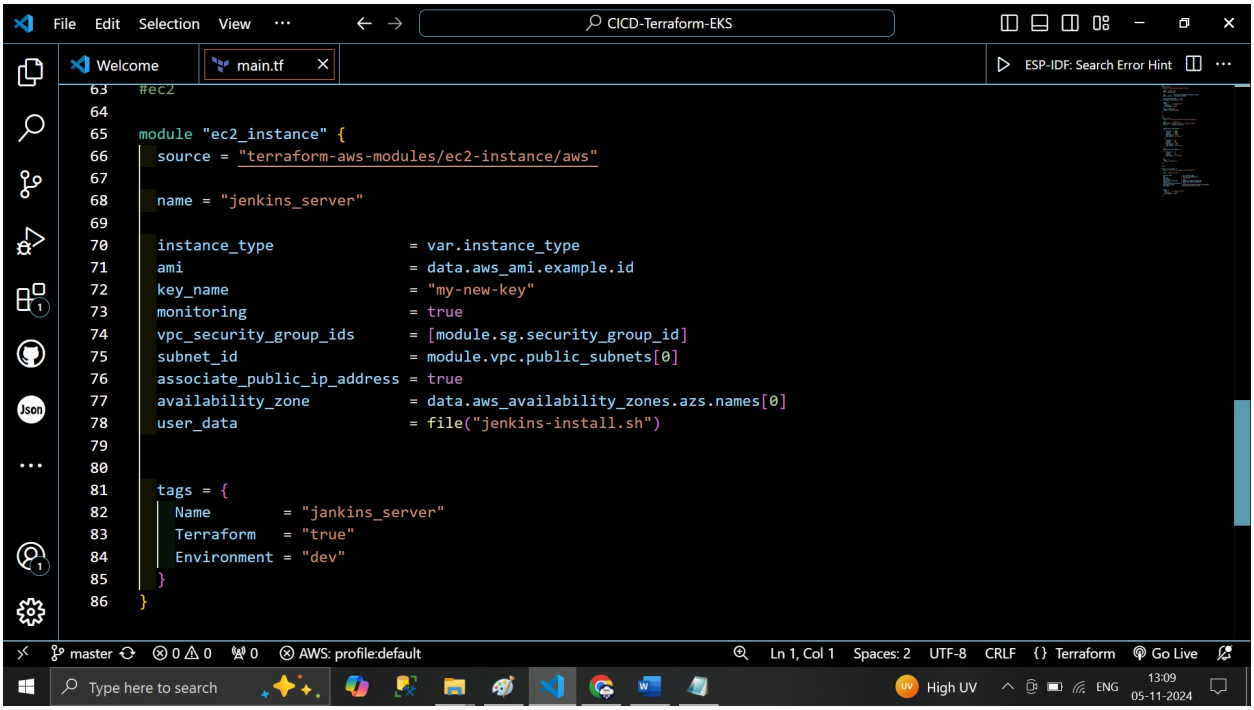


 # cp

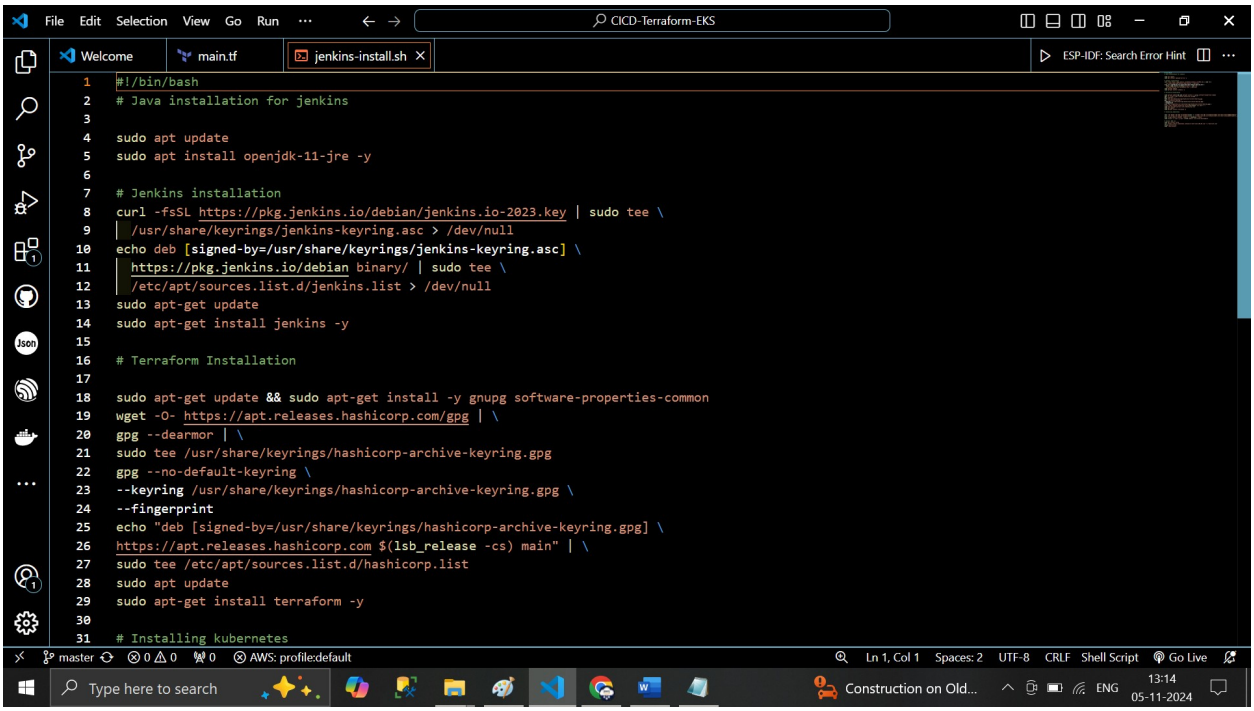


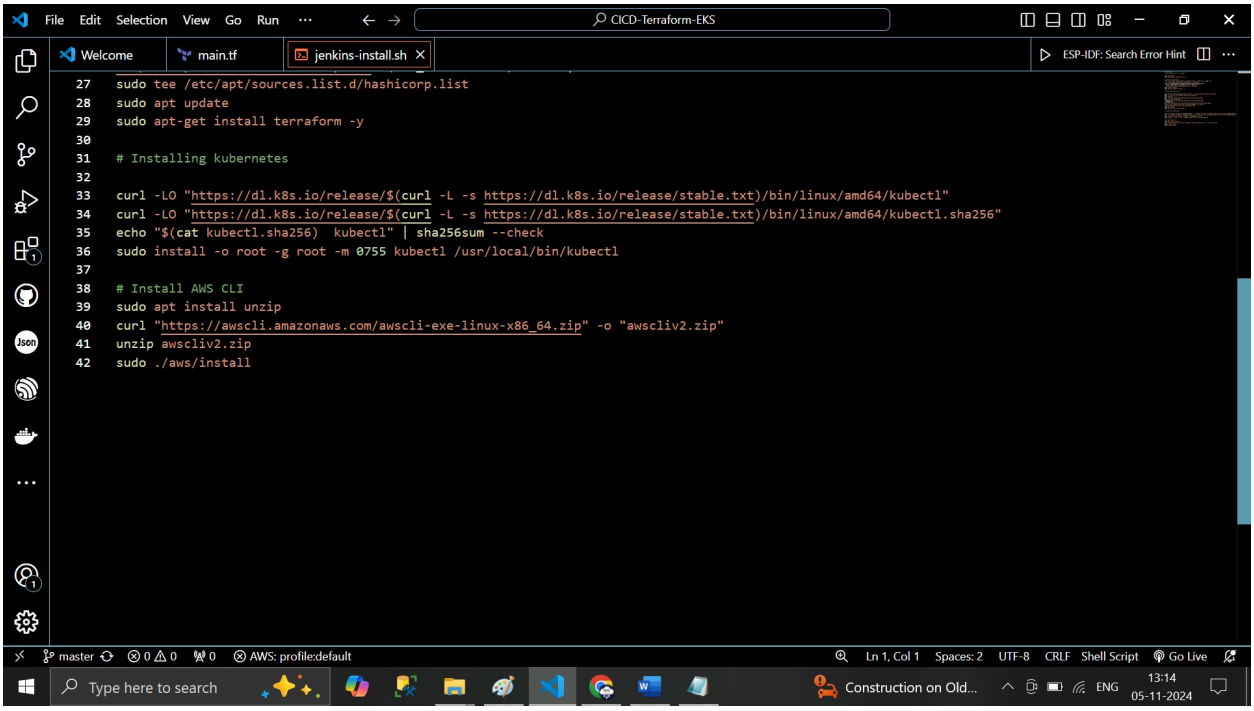
 # Existing Key pair



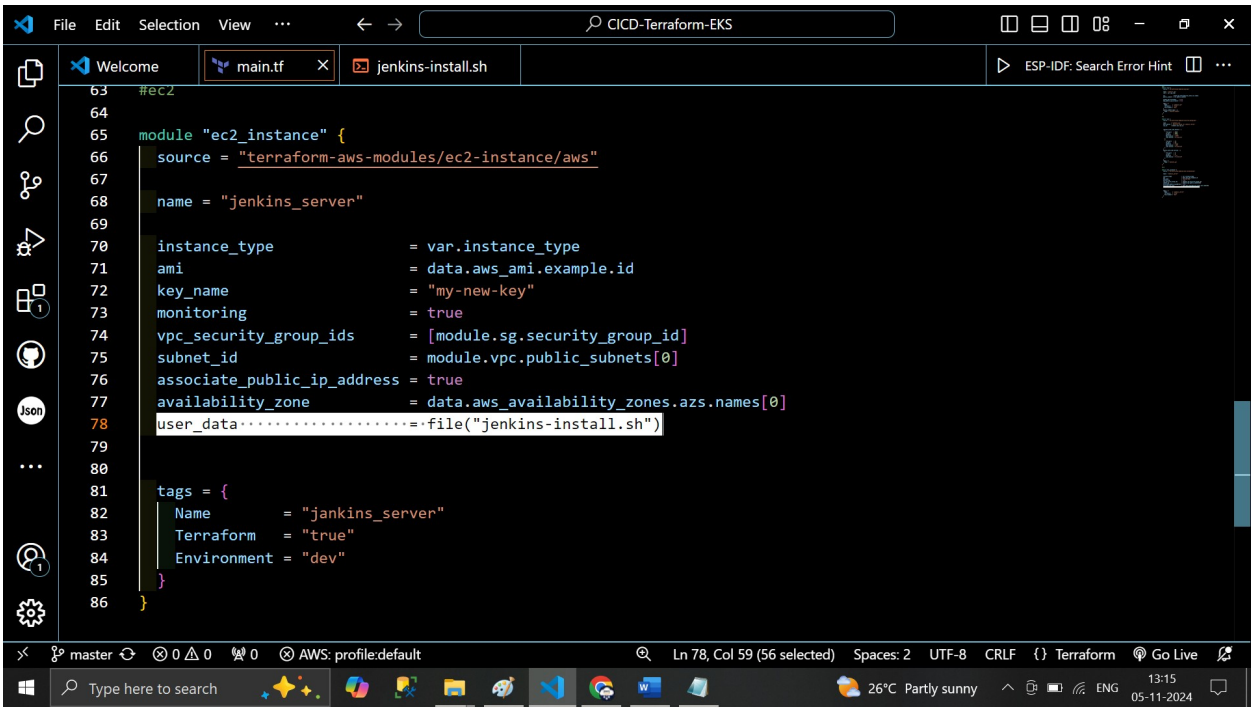


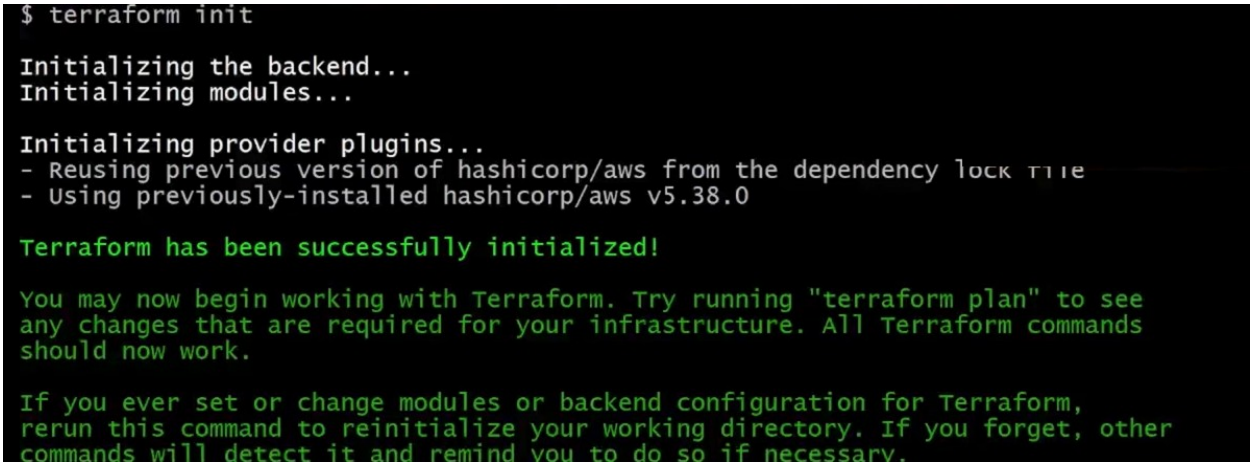
 # Creating a shell file name jekins-install that will automate the installation of java, jekins, tf &K8S

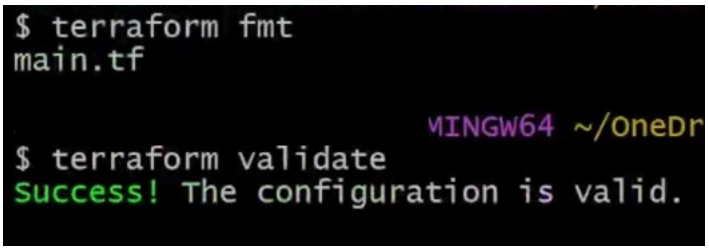


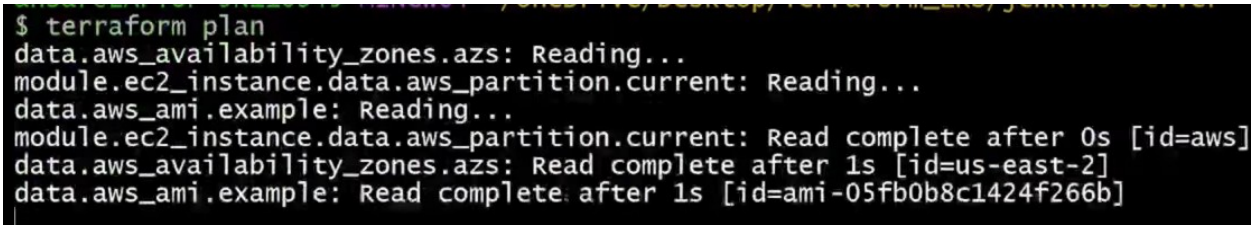


This Jenkins—install.sh is placed in main.tf file under ec2 installation automated


File Edit Selection View
CICD-Terraform-EKS
Welcome
main.tf
jenkins-install.sh
ESP-IDF: Search Error Hint
#ec2
module "ec2_instance" {
source = "terraform-aws-modules/ec2-instance/aws"
name = "jenkins_server"
instance_type = var.instance_type
ami = data.aws_ami.example.id
key_name = "my-new-key"
monitoring = true
vpc_security_group_ids = [module.sg.security_group_id]
subnet_id = module.vpc.public_subnets[0]
associate_public_ip_address = true
availability_zone = data.aws_availability_zones.azs.names[0]
user_data = file("jenkins-install.sh")
tags = {
Name = "jenkins_server"
Terraform = "true"
Environment = "dev"
}
}
master
AWS: profile:default
Ln 78, Col 59 (56 selected)
Spaces: 2
UTF-8
CRLF
Terraform
Go Live
Type here to search
26°C Partly sunny
ENG
13:15
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]

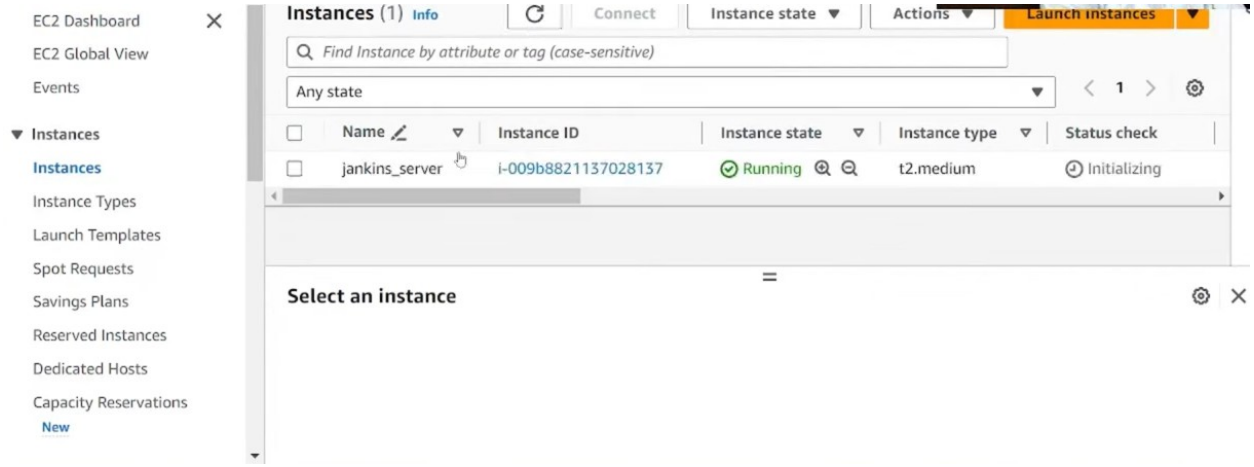
```
+ placement_partition_number           = (known after apply)
+ primary_network_interface_id         = (known after apply)
+ private_dns                          = (known after apply)
+ private_ip                           = (known after apply)
+ public_dns                           = (known after 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

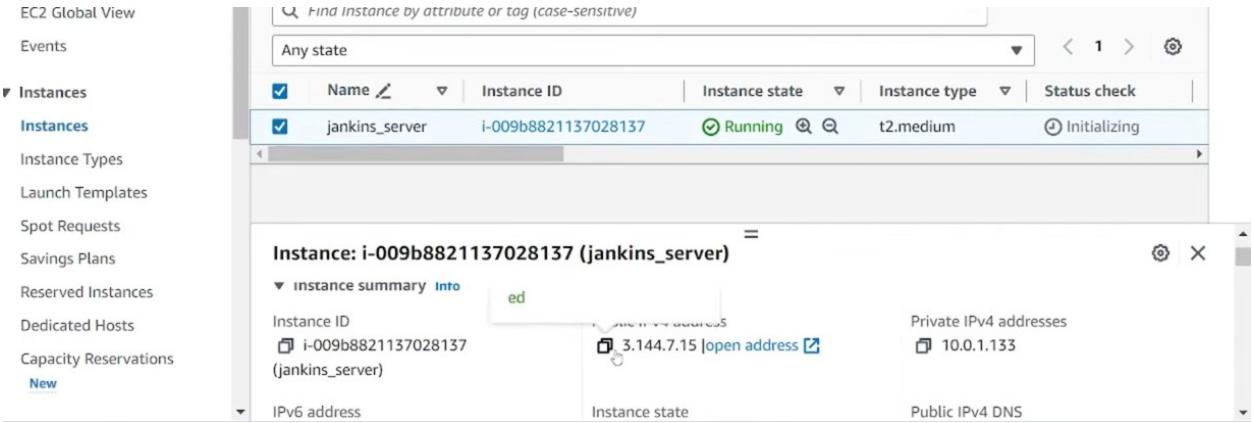
```
+ secondary_private_ips                 = (known after apply)
+ security_groups                       = (known after apply)
+ source_dest_check                     = true
+ spot_instance_request_id              = (known after apply)
+ subnet_id                             = (known after apply)
+ 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 {}
```

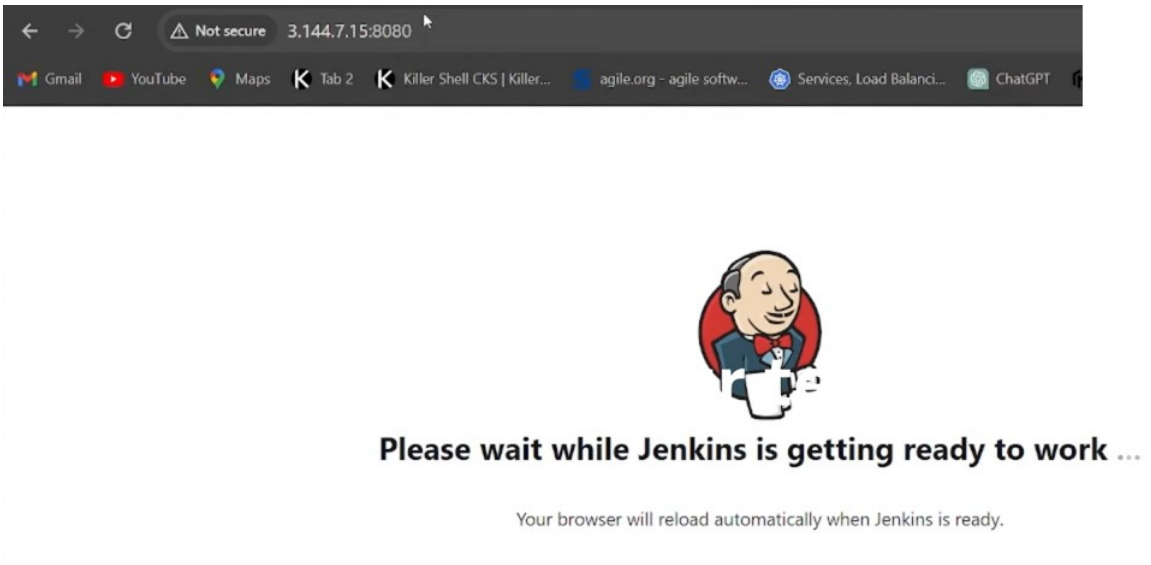
```
$ terraform apply --auto-approve
data.aws_availability_zones.azs: Reading...
module.ec2_instance.data.aws_partition.current: Reading...
data.aws_ami.example: Reading...
module.ec2_instance.data.aws_partition.current: Read complete after 0s [id=aws]
data.aws_availability_zones.azs: Read complete after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

```
grule-3341211921]
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-0cd0
0f1150665f648]
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]
```

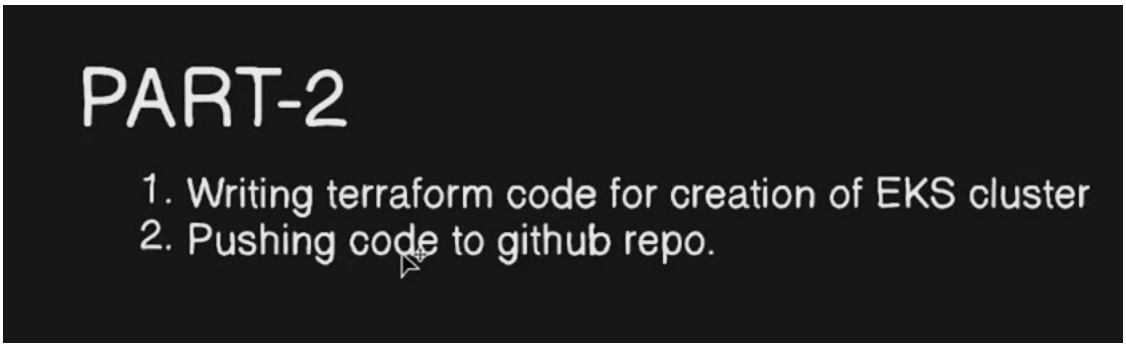


 # Check for IP Jenkins working env

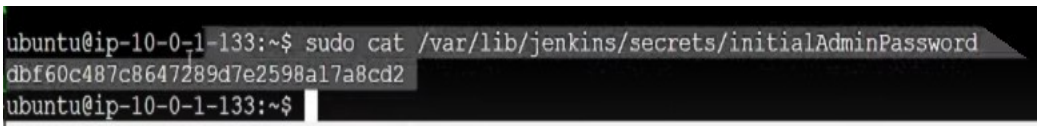



Not secure
3.144.7.15:8080
Gmail
YouTube
Maps
Tab 2
Killer Shell CKS | Killer...
agile.org - agile softw...
Services, Load Balanci...
ChatGPT
Please wait while Jenkins is getting ready to work ...
Your browser will reload automatically when Jenkins is ready.

 # 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


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.

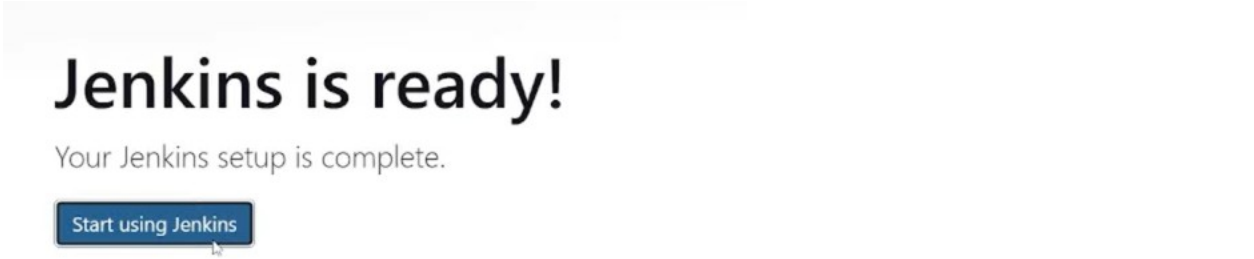
Select plugins to install

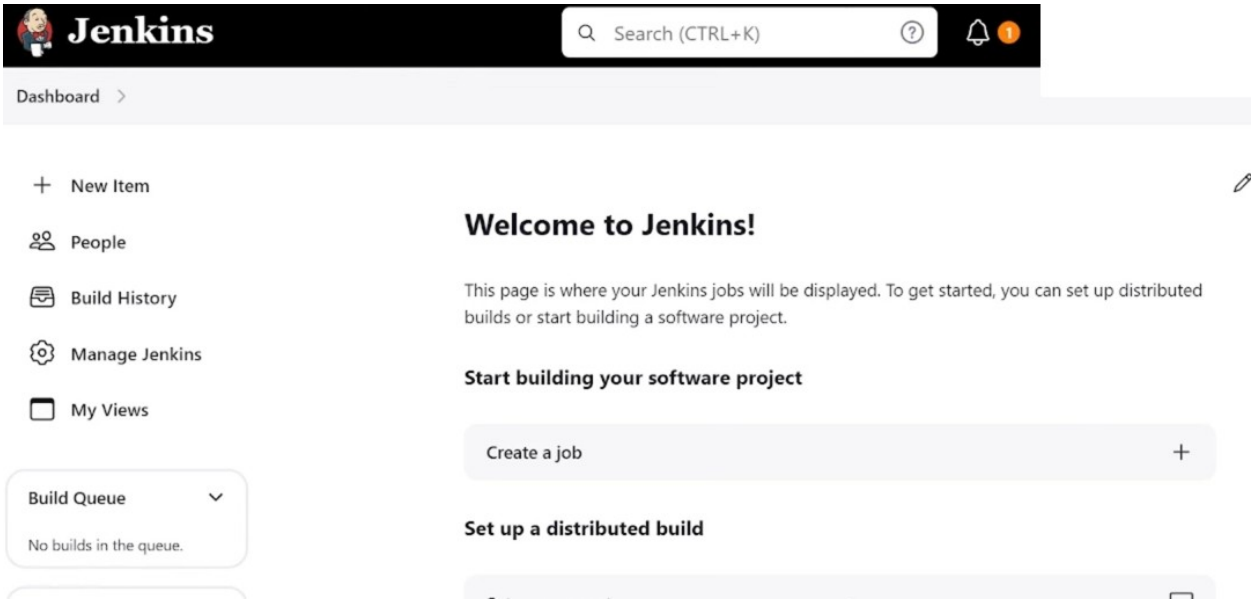
Select and install plugins most suitable for your needs.

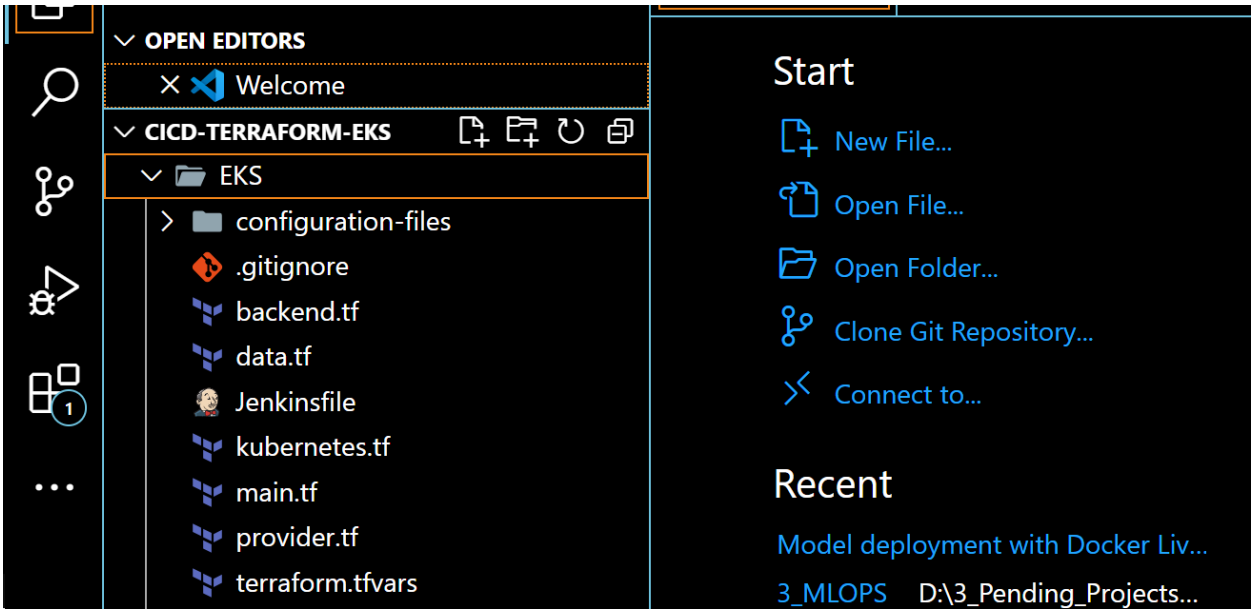
Getting Started

✓ Folders	✓ OWASP Markup	✓ Build Timeout	Credentials Binding	** JSON Path API ** Structs
Timestamper	Workspace Cleanup	Ant	Gradle	** Token Macro Build Timeout ** bouncycastle API ** Instance Identity ** JavaBeans Activation Framework (JAF) API ** JavaMail API ** Credentials ** Plain Credentials ** Gson API ** Trilead API
Pipeline	GitHub Branch Source	Pipeline: GitHub Groovy Libraries	Pipeline: Stage View	
Git	SSH Build Agents	Matrix Authorization Strategy	PAM Authentication	
LDAP	Email Extension	Mailer	Dark Theme	** - required dependency

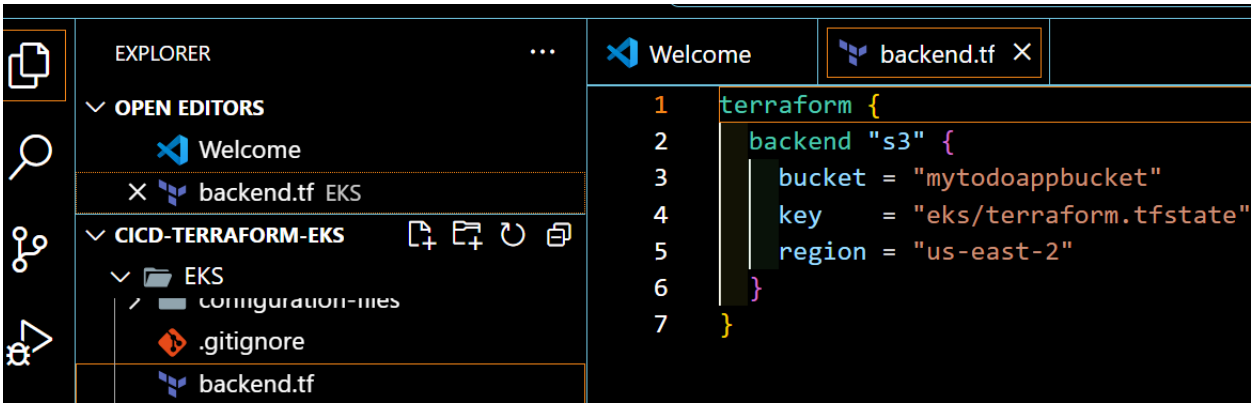

Create First Admin User
Username
admin
Password
Confirm password


Jenkins is ready!
Your Jenkins setup is complete.
Start using Jenkins

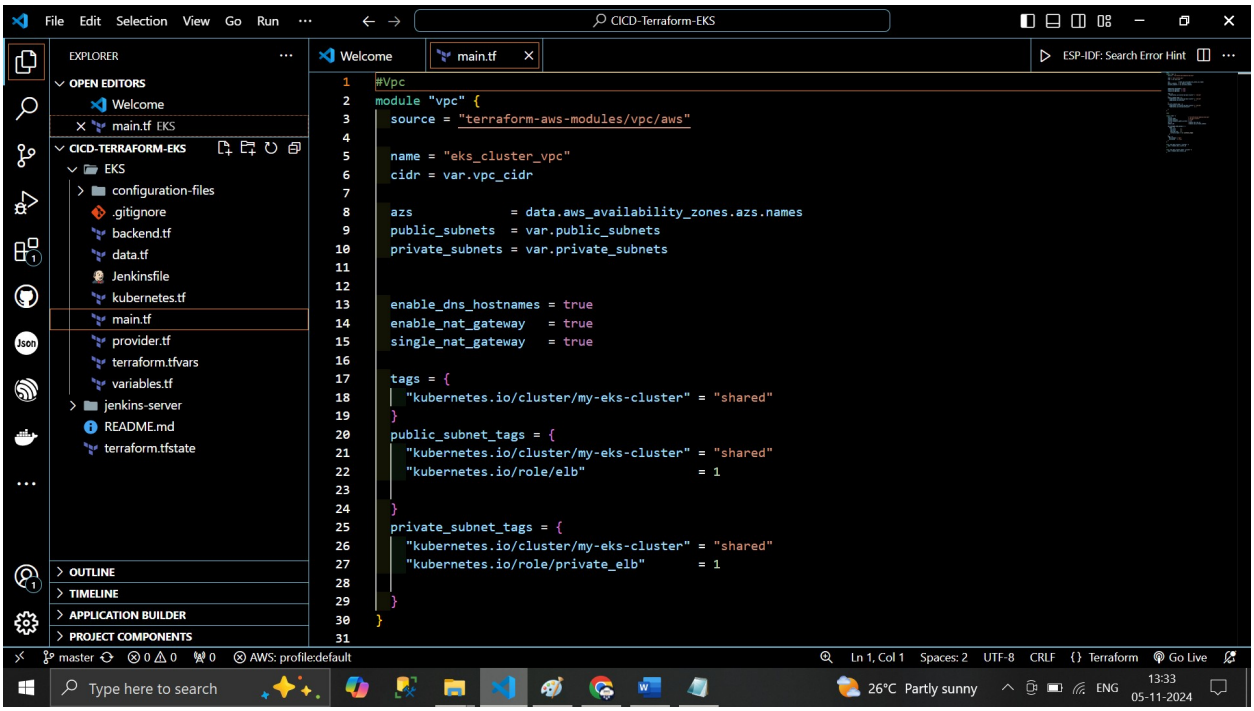

Jenkins
Search (CTRL+K)
Dashboard
New Item
People
Build History
Manage Jenkins
My Views
Build Queue
No builds in the queue.
Welcome to Jenkins!
This page is where your Jenkins jobs will be displayed. To get started, you can set up distributed builds or start building a software project.
Start building your software project
Create a job
Set up a distributed build

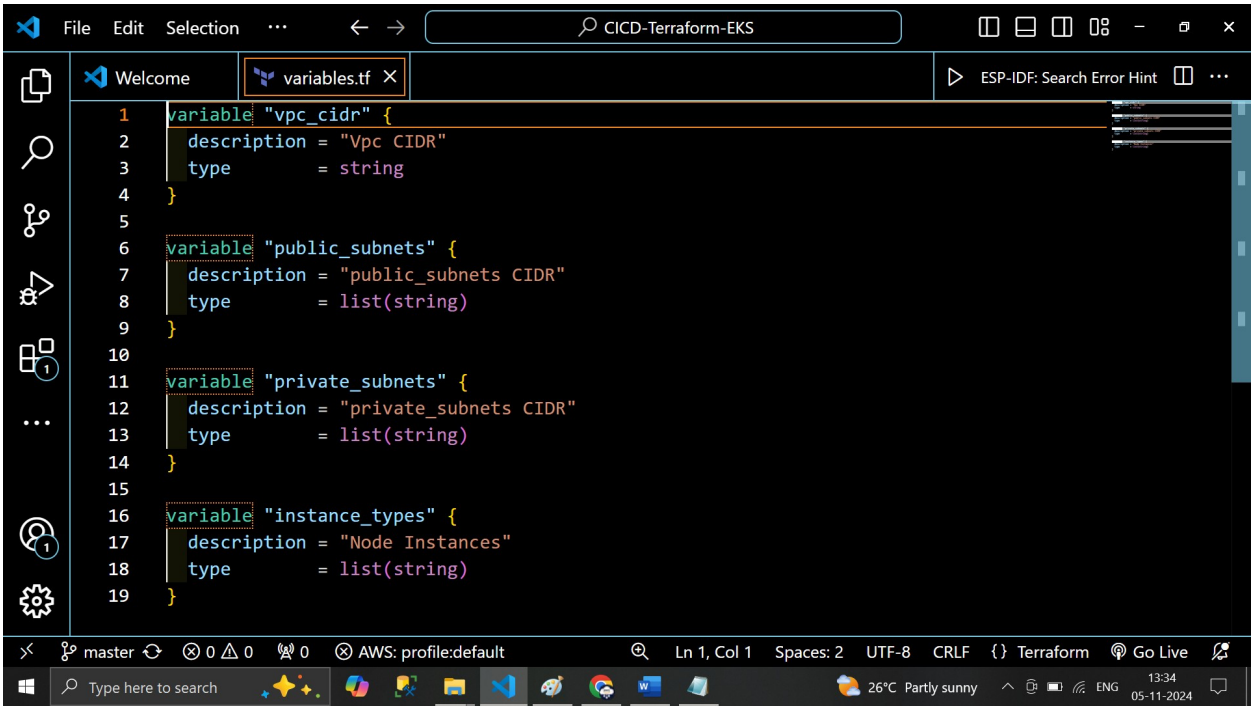


 # Creating EKS cluster



 # Main.tf


File Edit Selection View Go Run
CICD-Terraform-EKS
EXPLORER
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
OUTLINE
TIMELINE
APPLICATION BUILDER
PROJECT COMPONENTS
Welcome
main.tf
ESP-IDF: Search Error Hint
#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
}
}
master 0 0 0 AWS: profile:default
Ln 1, Col 1 Spaces: 2 UTF-8 CRLF Terraform Go Live
Type here to search
26°C Partly sunny
ENG
13:33
05-11-2024


File Edit Selection
CICD-Terraform-EKS
Welcome
variables.tf
ESP-IDF: Search Error Hint
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)
}
master 0 0 0 AWS: profile:default
Ln 1, Col 1 Spaces: 2 UTF-8 CRLF Terraform Go Live
Type here to search
26°C Partly sunny
ENG
13:34
05-11-2024

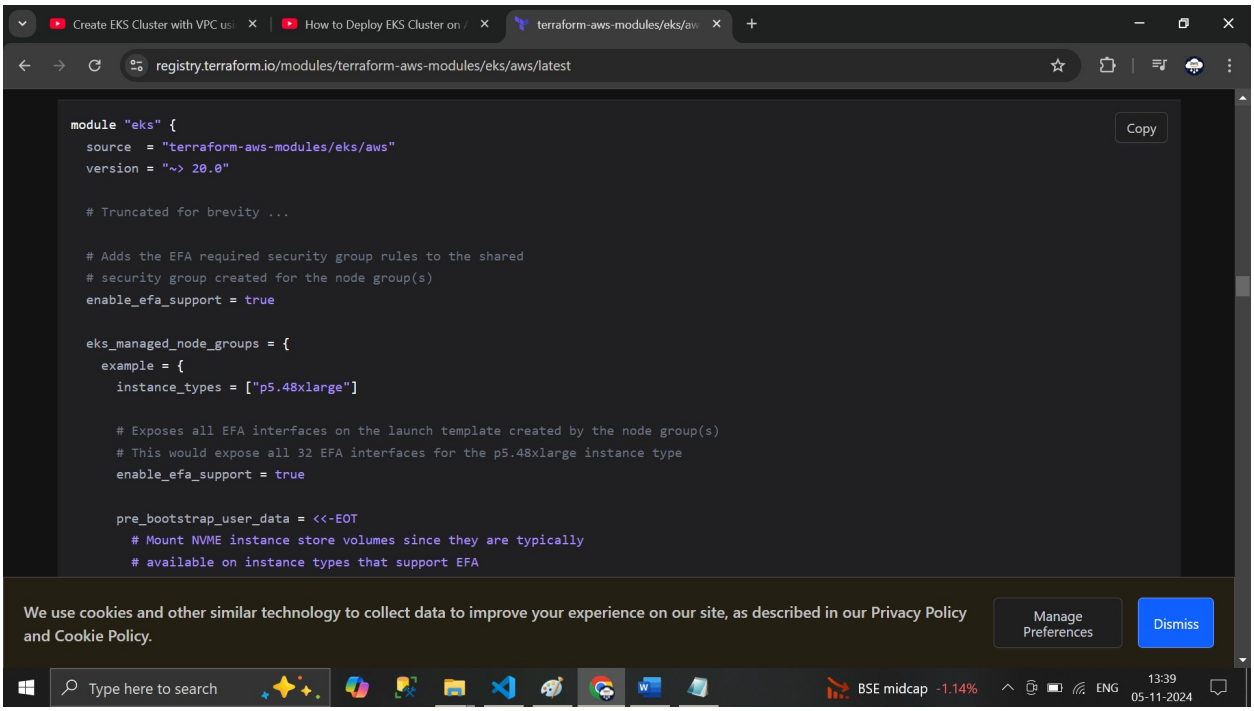
```
$ terraform init

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.
```

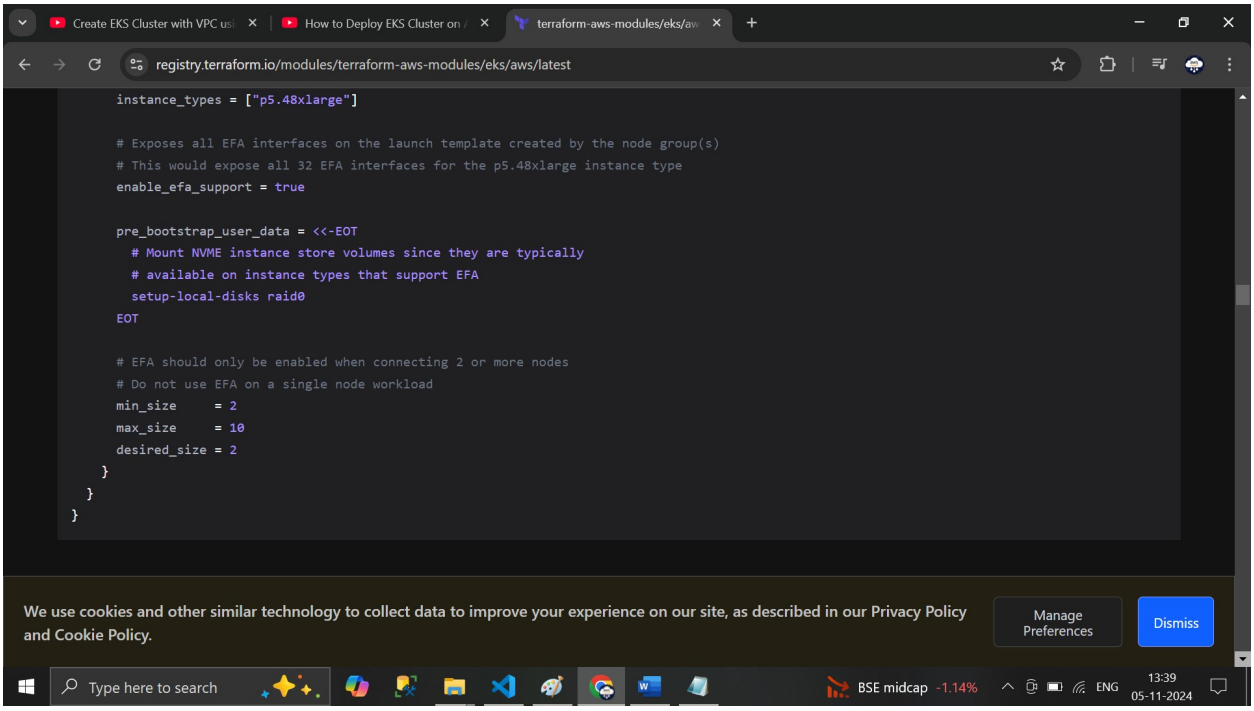
```
$ terraform validate
Success! The configuration is valid.
```

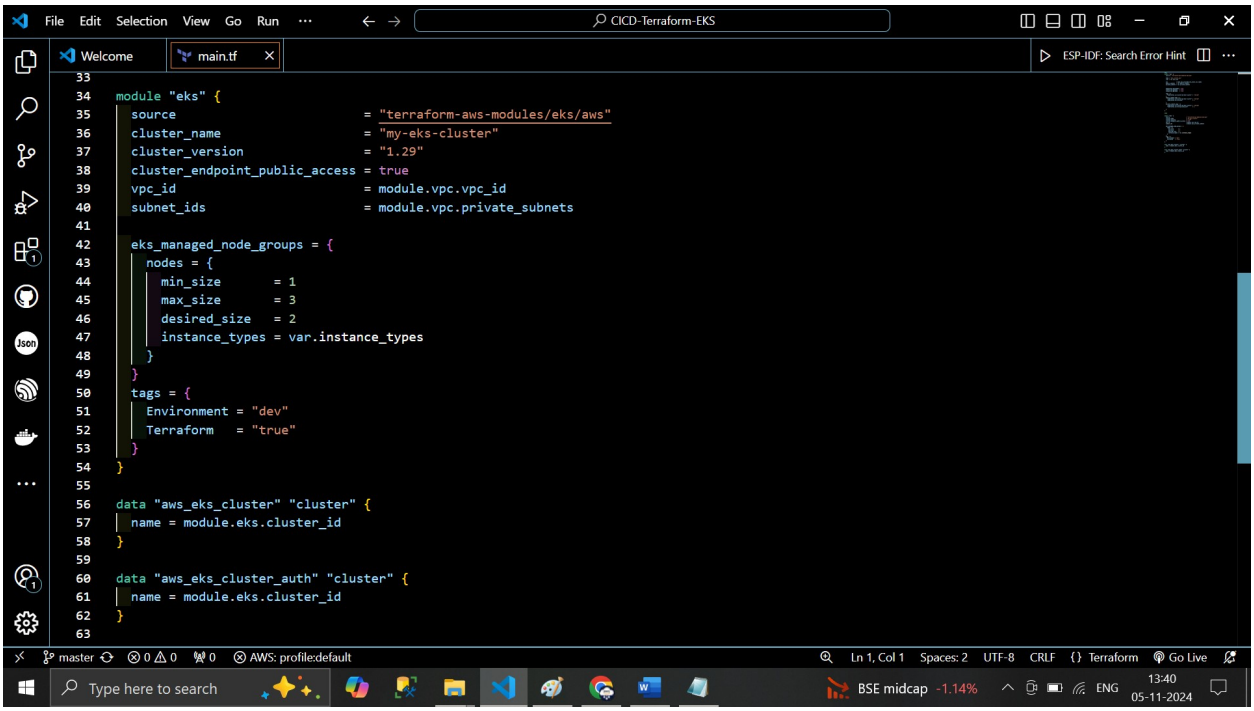
```
$ terraform plan
data.aws_availability_zones.azs: Reading...
```

```
$ 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 execu
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"
```

 # Terrafor eks module Doc


Create EKS Cluster with VPC usi
How to Deploy EKS Cluster on
terraform-aws-modules/eks/aw
registry.terraform.io/modules/terraform-aws-modules/eks/aws/latest
Copy
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
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
Type here to search
BSE midcap -1.14%
ENG
13:39
05-11-2024


Create EKS Cluster with VPC usi
How to Deploy EKS Cluster on
terraform-aws-modules/eks/aw
registry.terraform.io/modules/terraform-aws-modules/eks/aws/latest
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
Type here to search
BSE midcap -1.14%
ENG
13:39
05-11-2024


File Edit Selection View Go Run ···
CICD-Terraform-EKS
Welcome
main.tf
ESP-IDF: Search Error Hint
33
34 module "eks" {
35 source = "terraform-aws-modules/eks/aws"
36 cluster_name = "my-eks-cluster"
37 cluster_version = "1.29"
38 cluster_endpoint_public_access = true
39 vpc_id = module.vpc.vpc_id
40 subnet_ids = module.vpc.private_subnets
41
42 eks_managed_node_groups = {
43 nodes = {
44 min_size = 1
45 max_size = 3
46 desired_size = 2
47 instance_types = var.instance_types
48 }
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
master
0 0
0
AWS: profile:default
Ln 1, Col 1
Spaces: 2
UTF-8
CRLF
Terraform
Go Live
Type here to search
BSE midcap -1.14%
ENG
13:40
05-11-2024

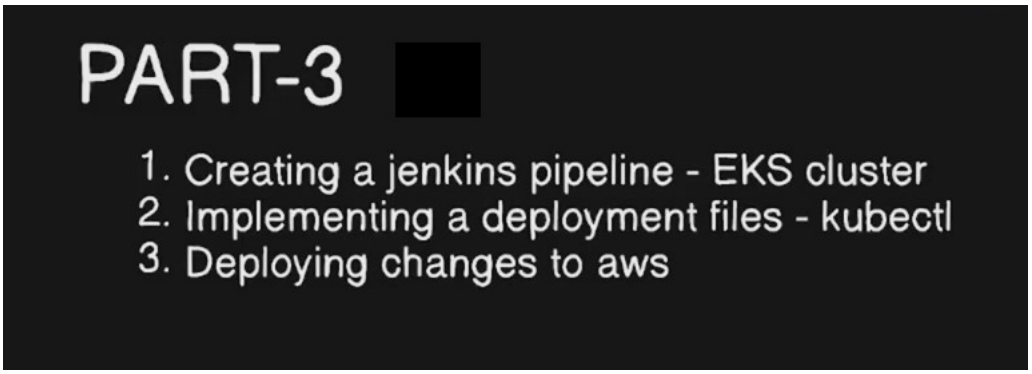
```
$ terraform plan
data.aws_availability_zones.azs: Reading...
data.aws_availability_zones.azs: Read complete after 1s [id=us-east-2

Terraform used the selected providers to generate the following execu
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
```

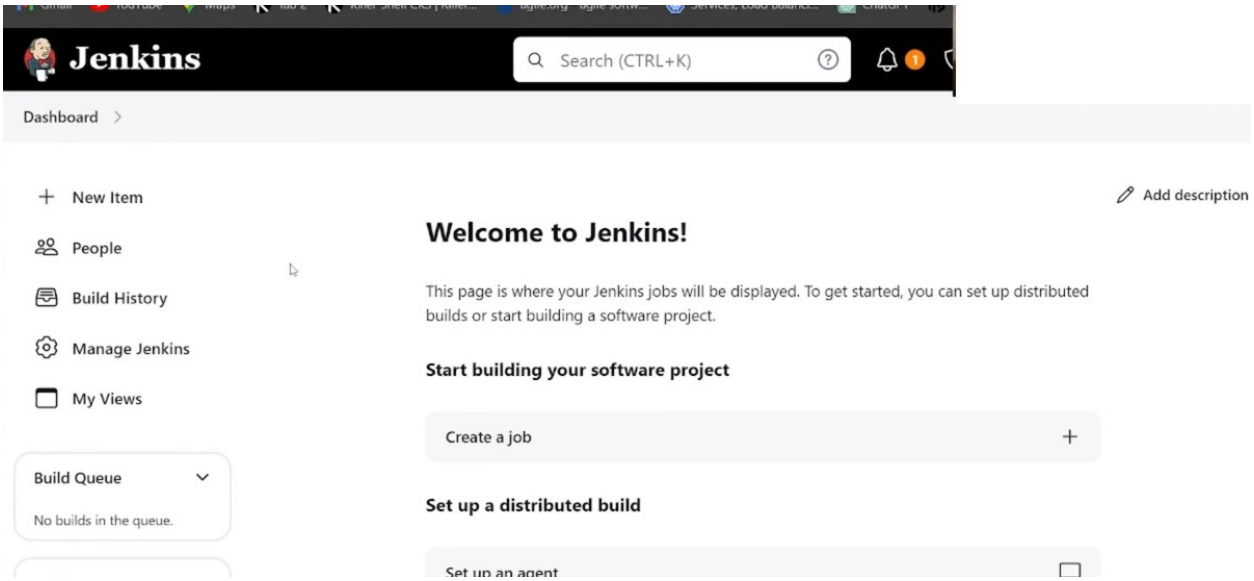
```
$ terraform validate
Success! The configuration is valid.
```

 # Part 2 completed

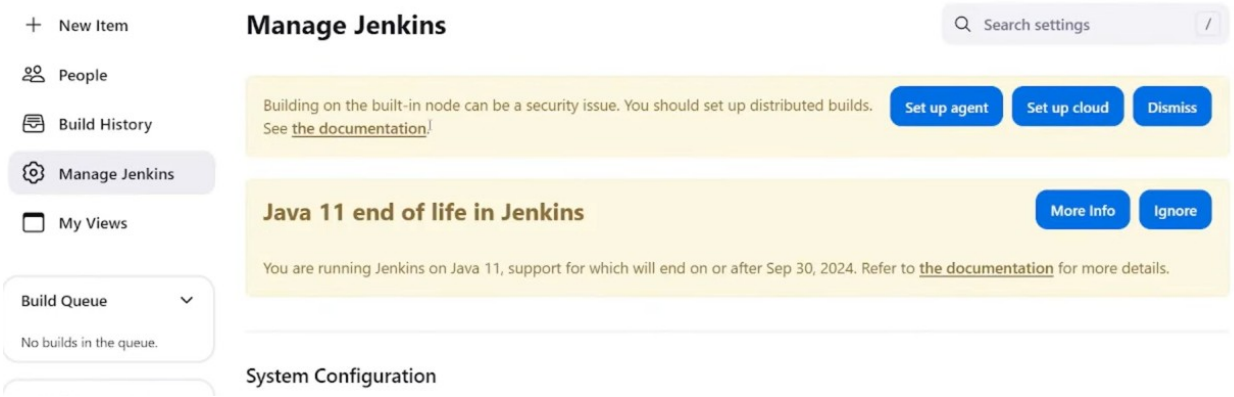
 # Part 3



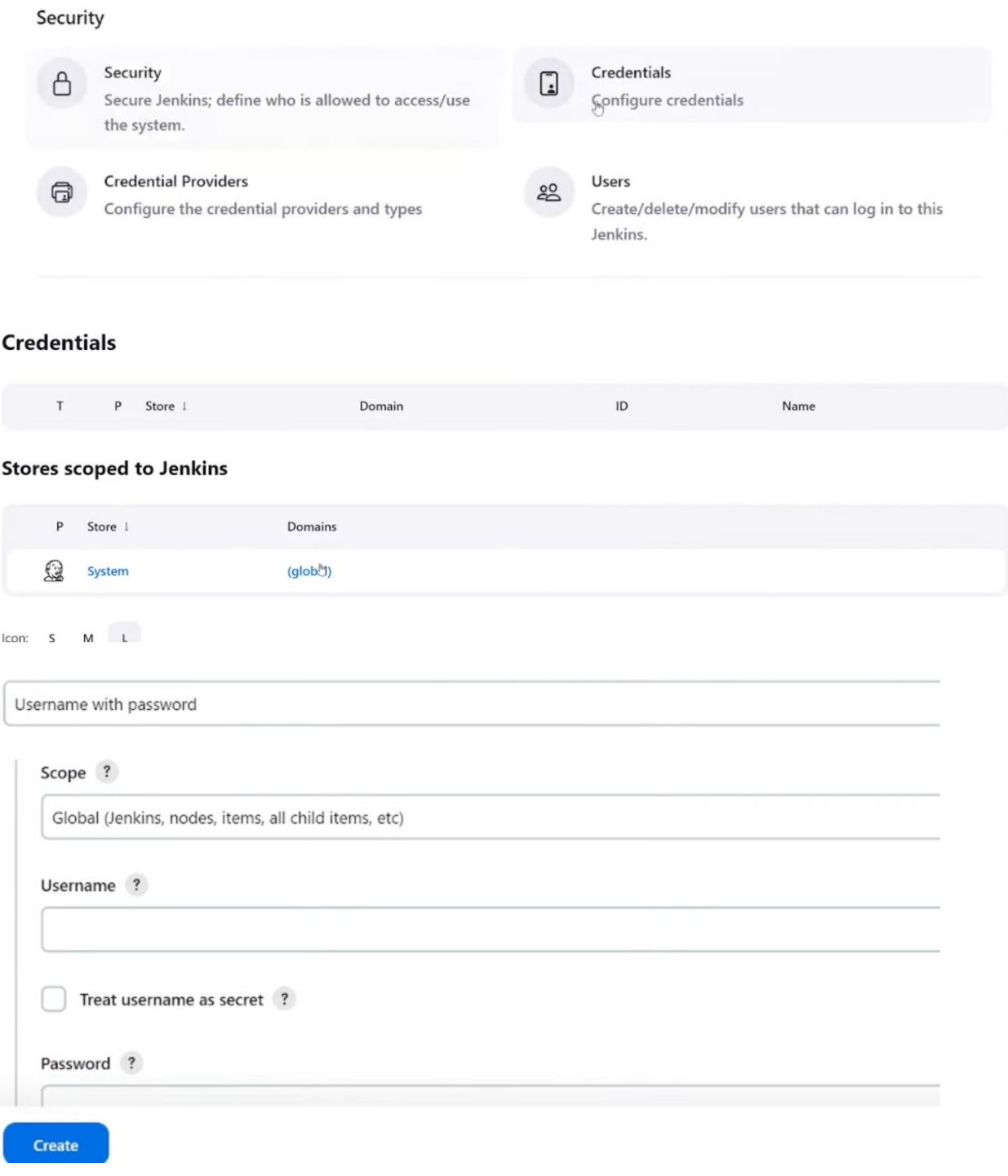
 # note Jenkins needs Aws Services for that access credentials from AWS is a must

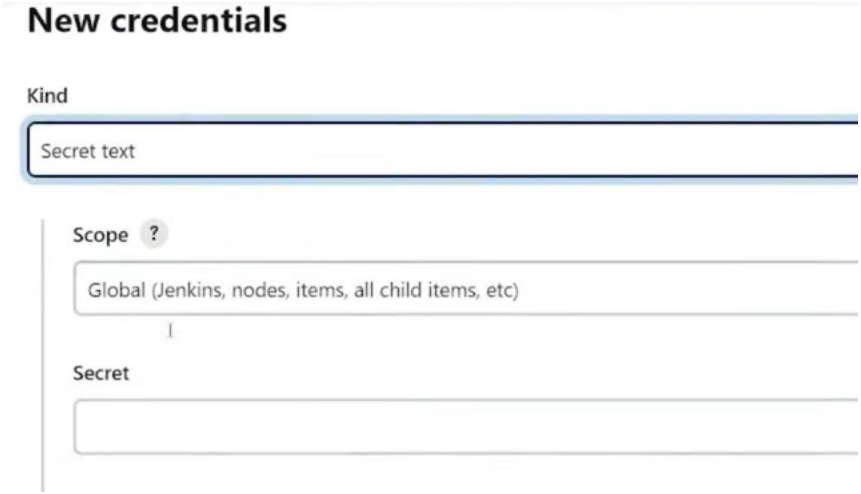


 # manage Jenkins



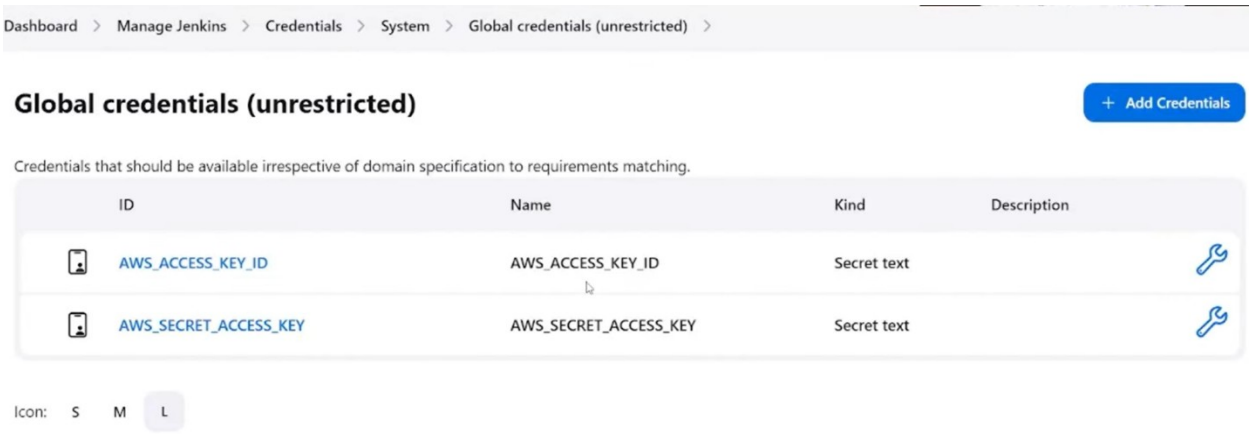
 # Credentails



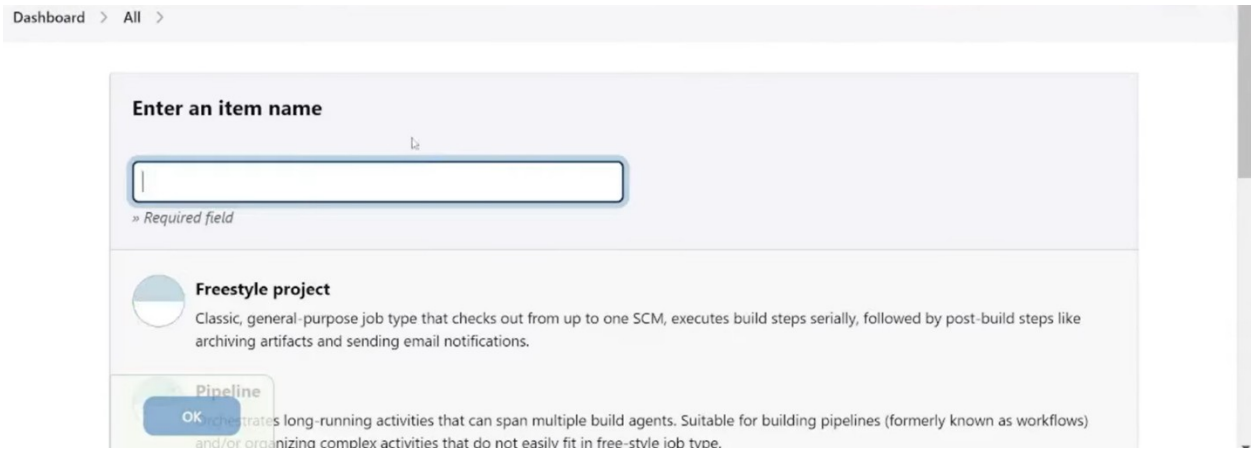


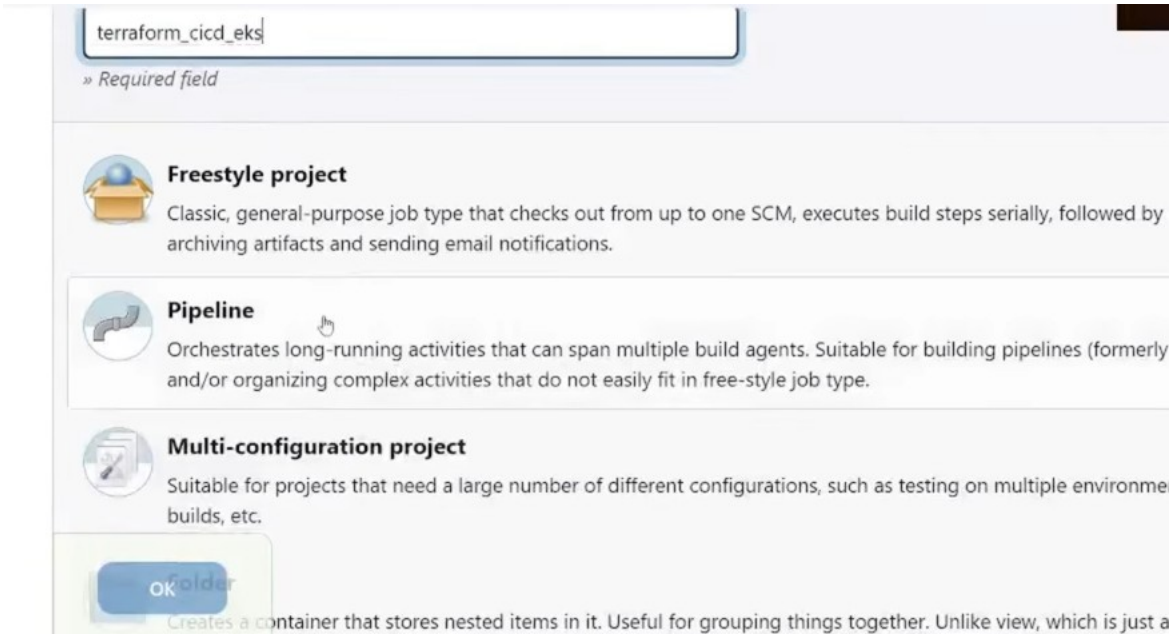
 # Secret – Access Key

Access key ID	Secret access key
AKIARMRNV2OQYPTAZ2ZS	n0QMB9mhzr/fMiJL[illegible]dAyy6dzmBFiKKnbd+3BFAF

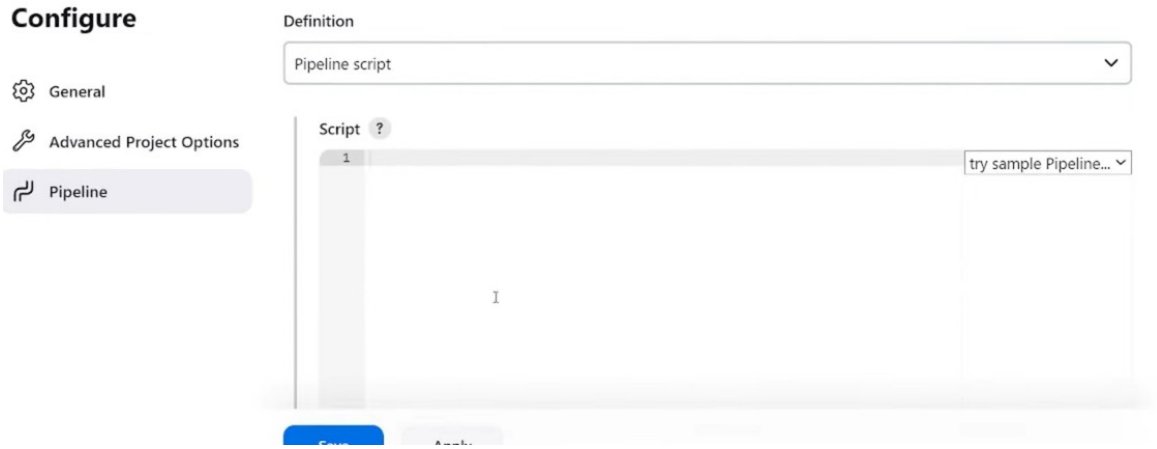


 # Go to dashboard – Click on item – select pipeline

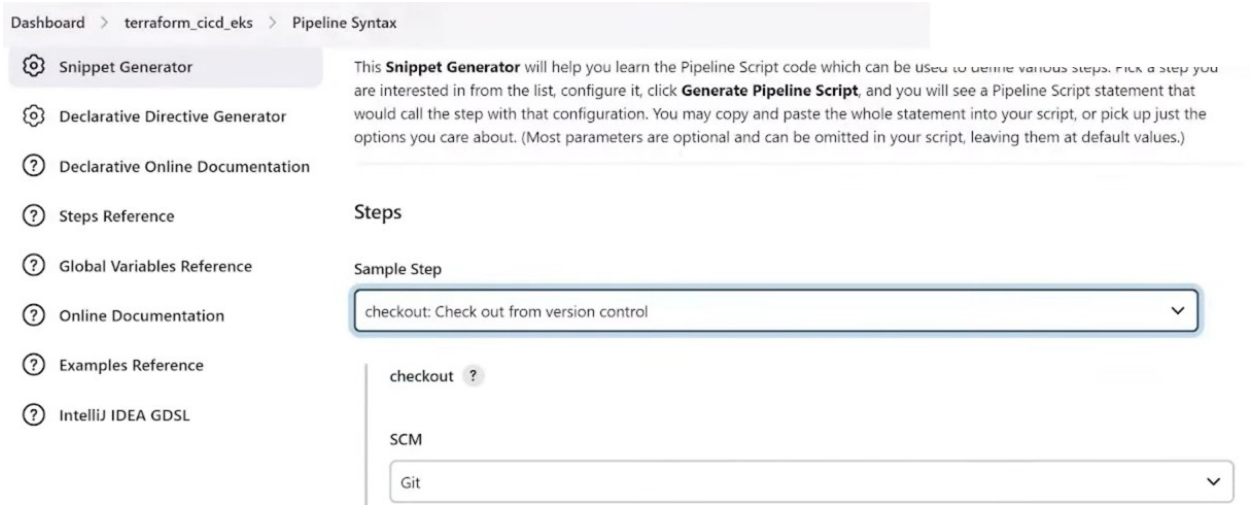




 # pipeline Script – Groovy Language



 # Checkout SCM – generate script – click pipeline synatax

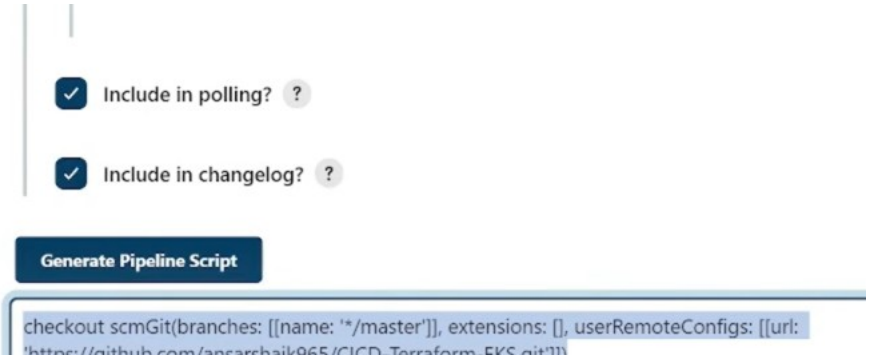


 # paste Your Repo

h

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

#cp



 # paste it in script

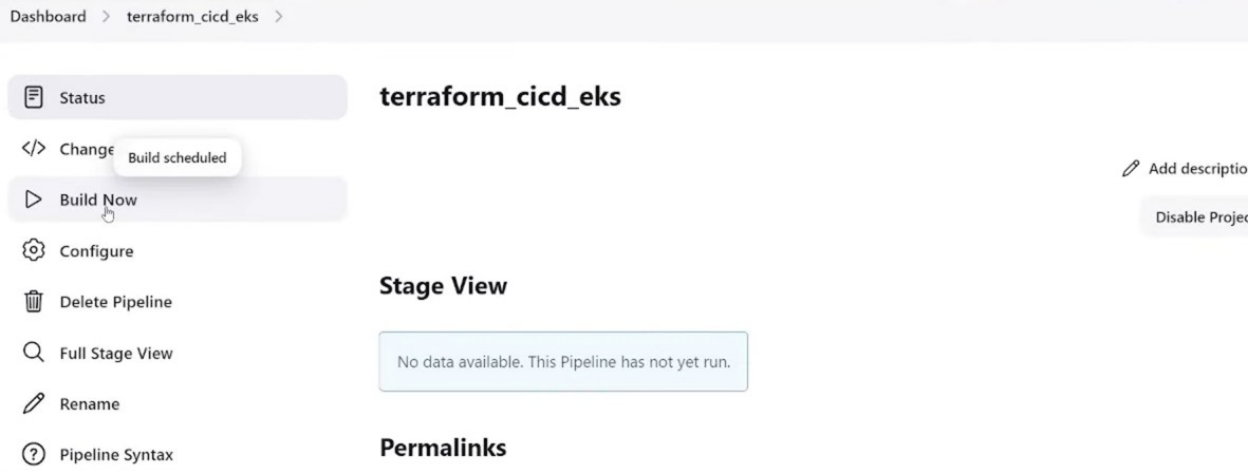
```
        AWS_ACCESS_KEY_ID = credentials('AWS_ACCESS_KEY_ID')
        AWS_SECRET_ACCESS_KEY = credentials('AWS_SECRET_ACCESS_KEY')
        AWS_DEFAULT_REGION = 'us-east-2'
    }
    stages{
        stage('Checkout SCM'){
            steps{
                script{
                    checkout scmGit(branches: [[name: '*/master']], extensions: [], userRemoteConfig
                }
            }
        }
    }
}
```

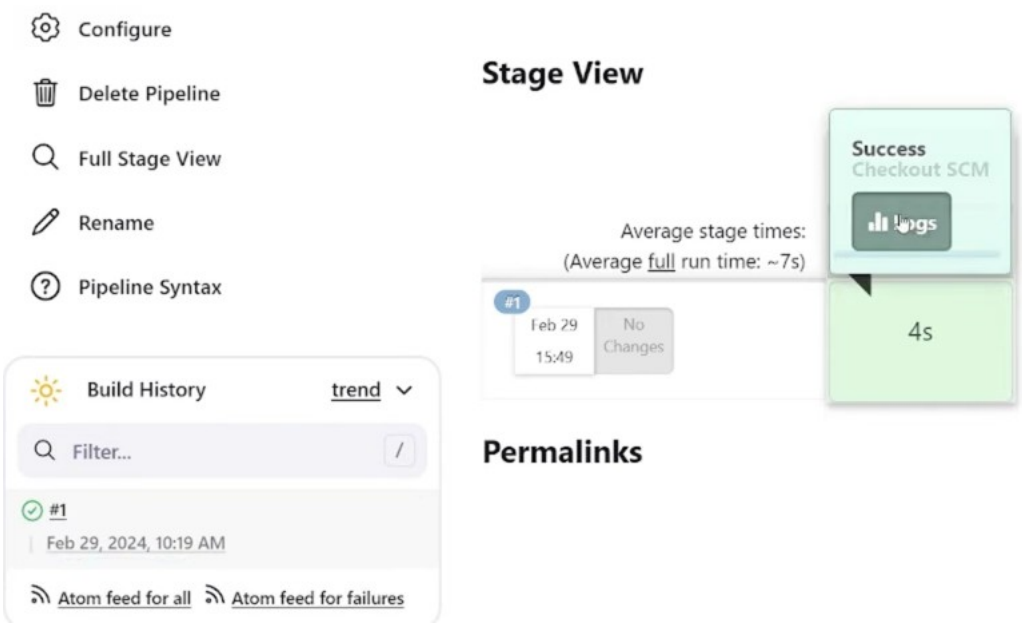
Use Groovy Sandbox ?

Pipeline Syntax

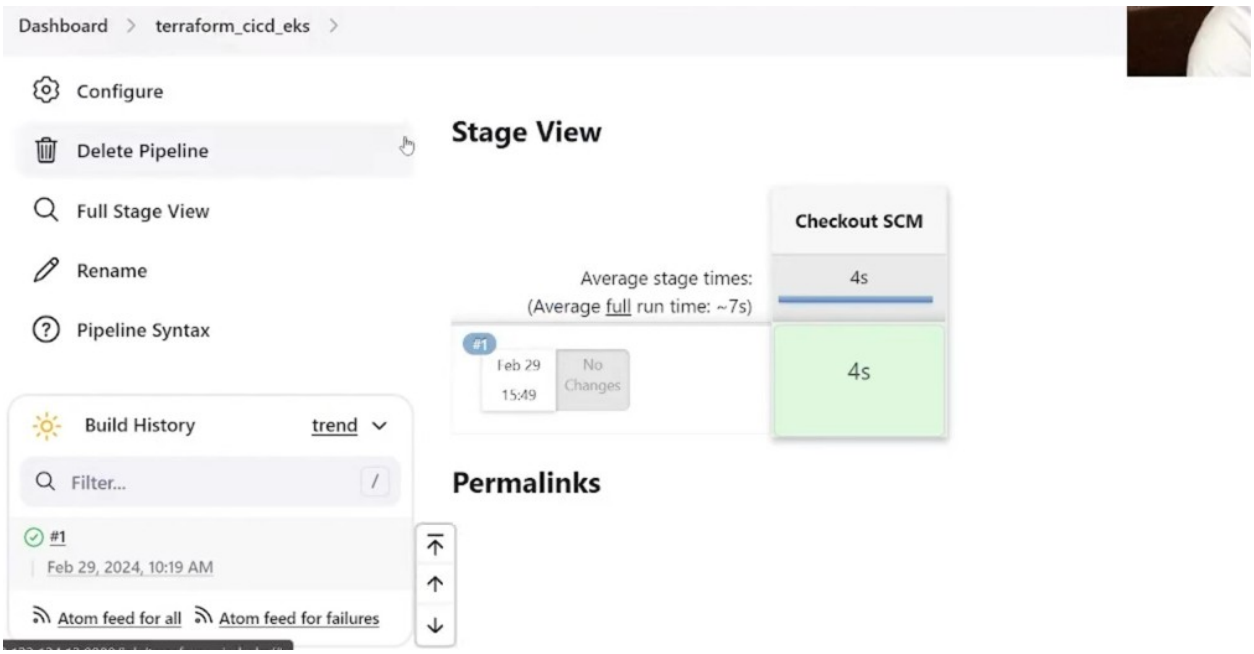
Save Apply

 # Build now



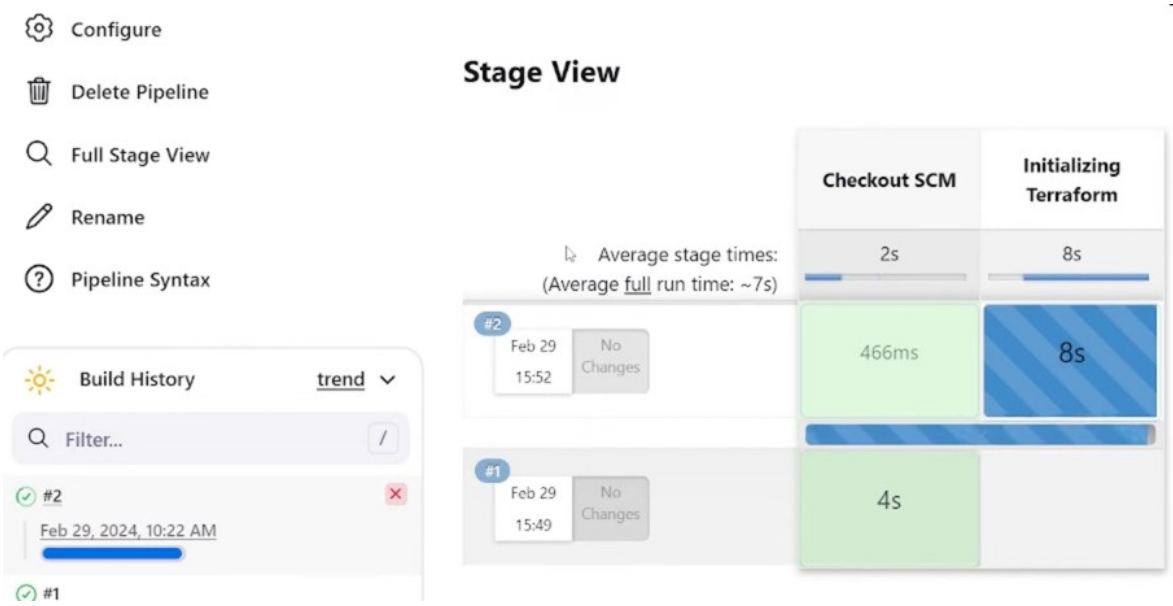


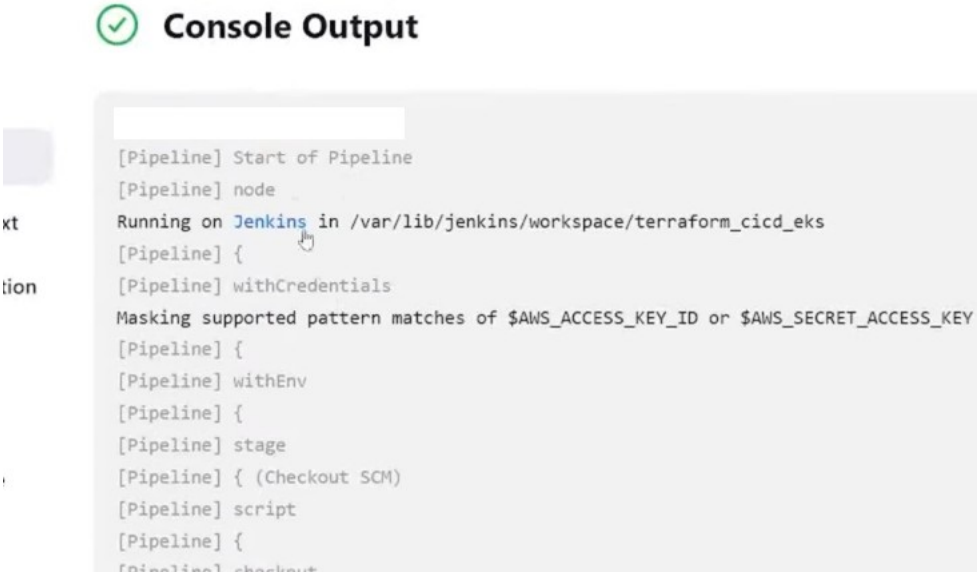
 # Going to 2nd Stage



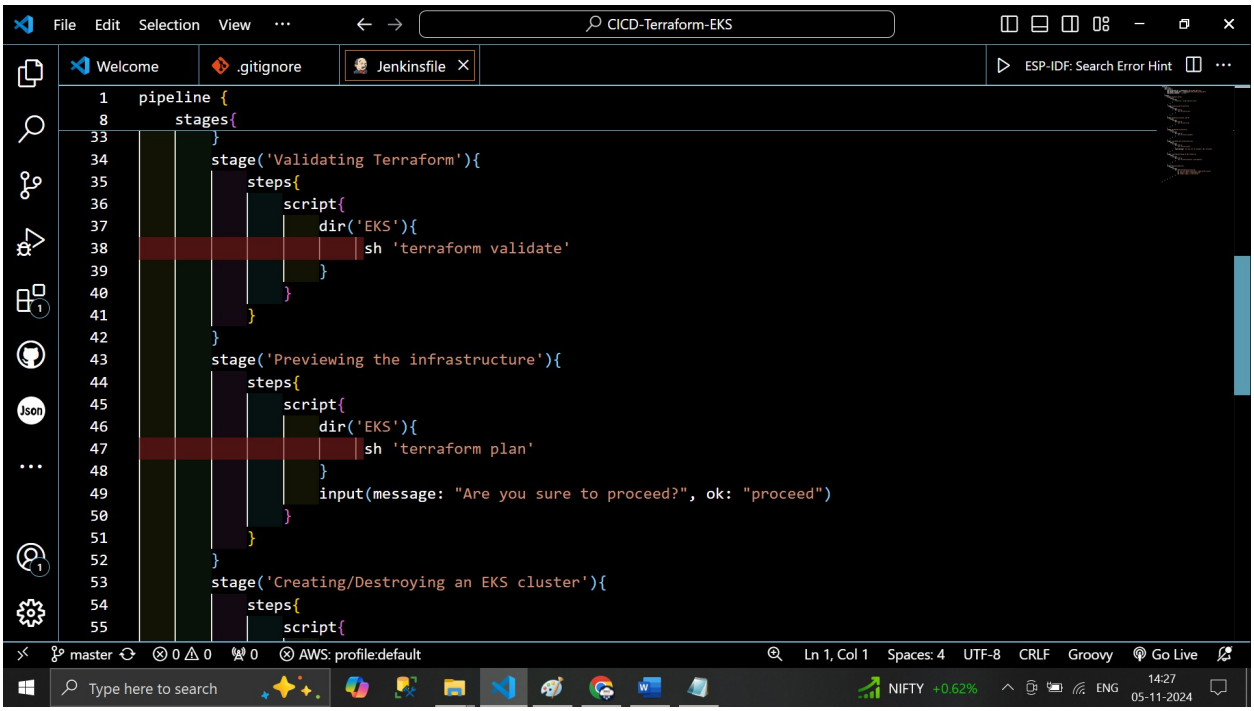
 # Click on Configure

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

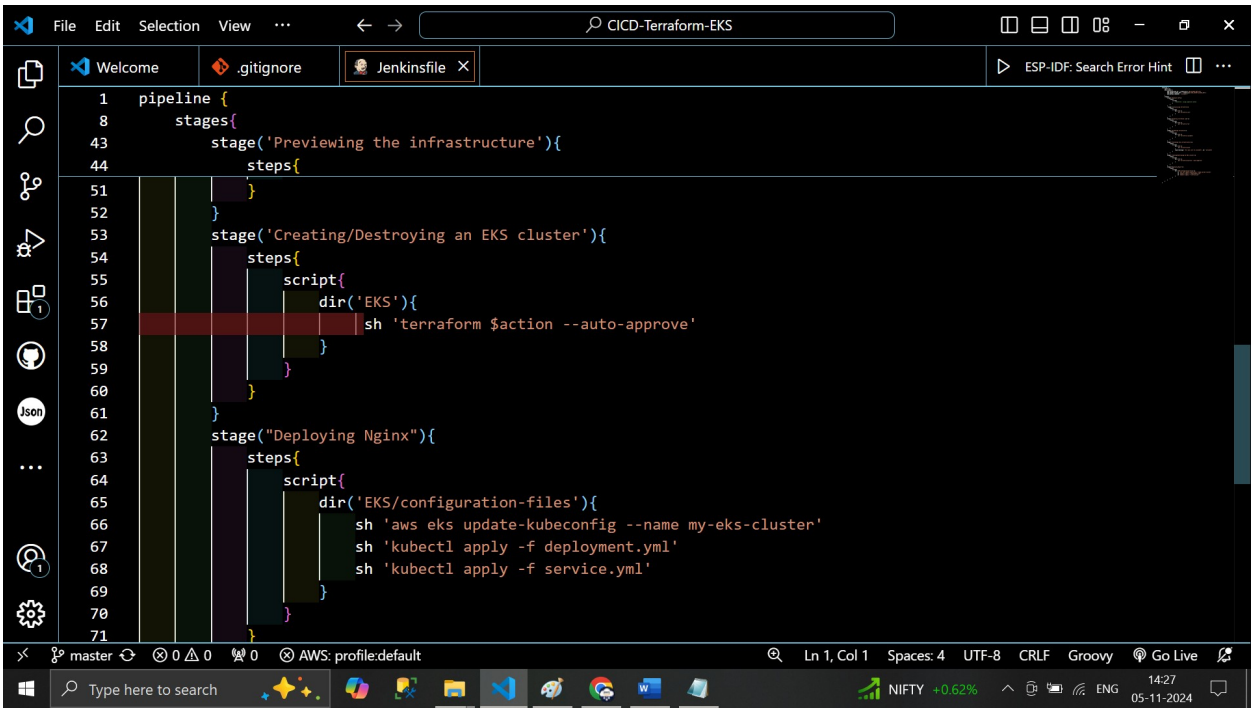




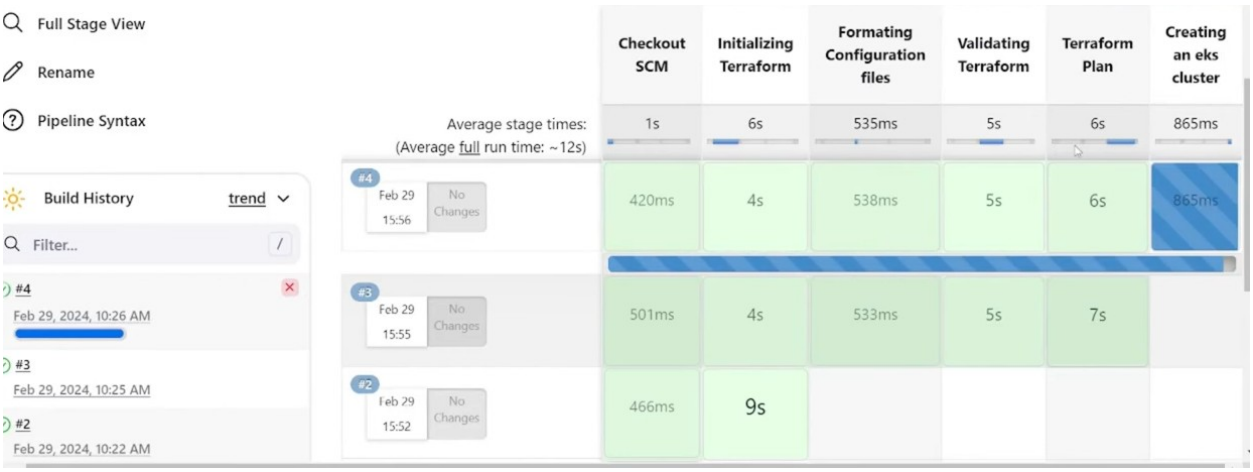
 # other stages



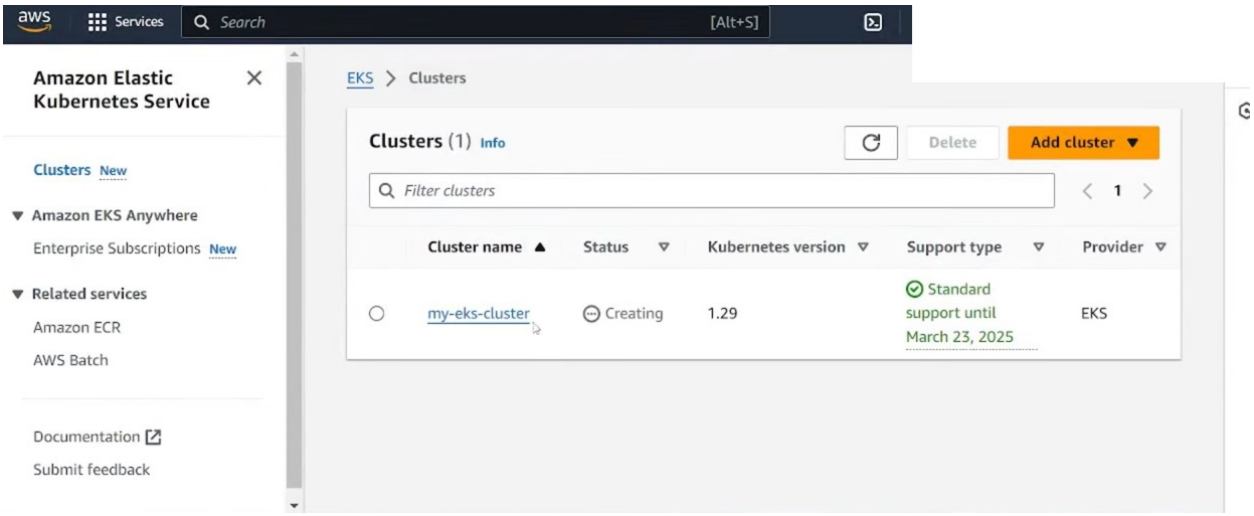
```
pipeline {
    stages{
        }
        stage('Validating Terraform'){
            steps{
                script{
                    dir('EKS'){
                        sh 'terraform validate'
                    }
                }
            }
        }
        stage('Previewing the infrastructure'){
            steps{
                script{
                    dir('EKS'){
                        sh 'terraform plan'
                    }
                    input(message: "Are you sure to proceed?", ok: "proceed")
                }
            }
        }
        stage('Creating/Destroying an EKS cluster'){
            steps{
                script{
```

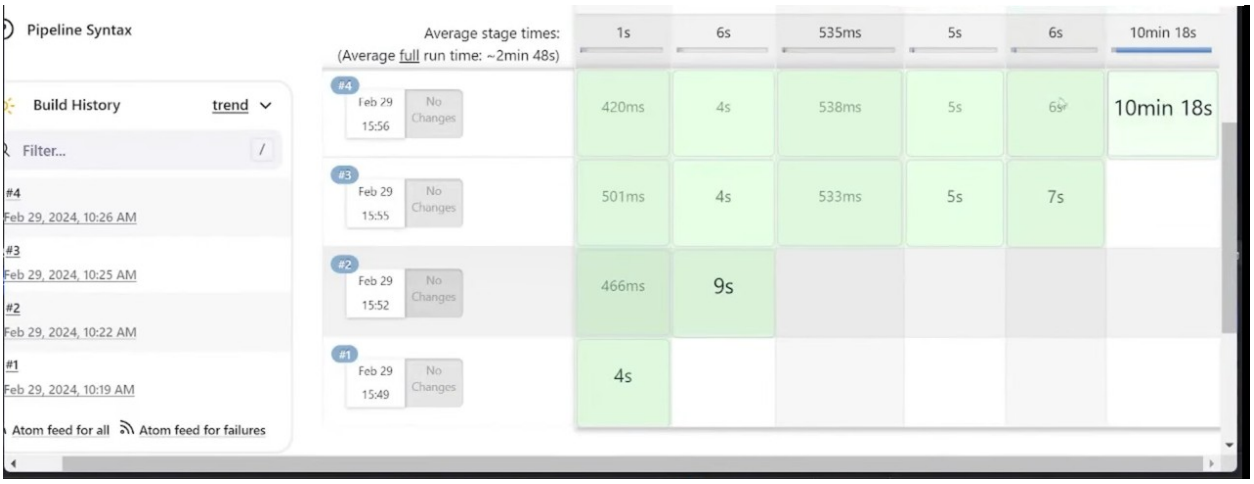


```
pipeline {
    stages{
        stage('Previewing the infrastructure'){
            steps{
            }
        }
        stage('Creating/Destroying an EKS cluster'){
            steps{
                script{
                    dir('EKS'){
                        sh 'terraform $action --auto-approve'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

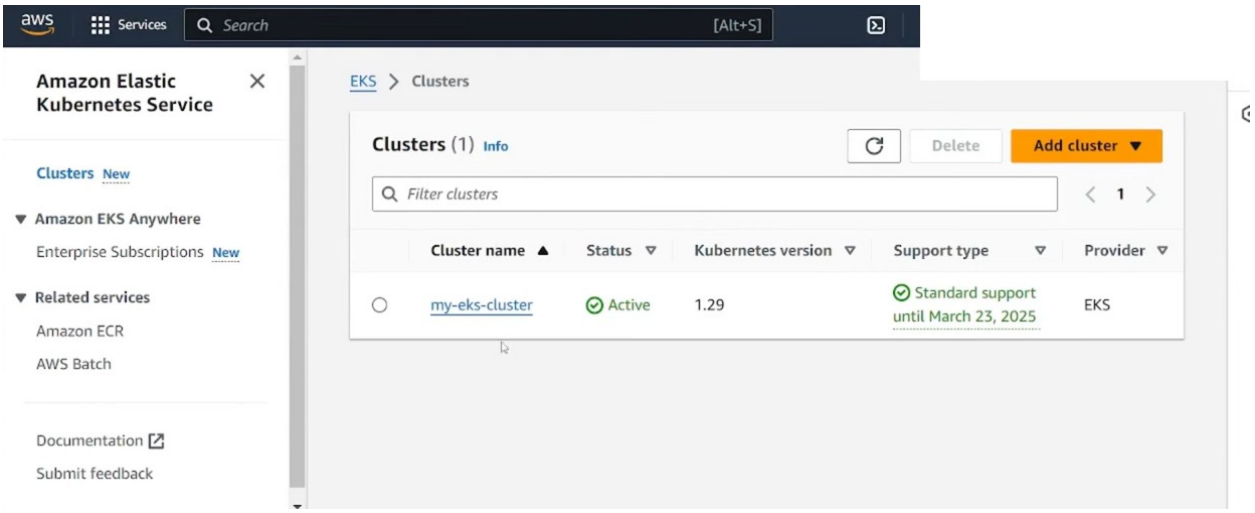


 # Check in Aws EKS Cluster Created ??

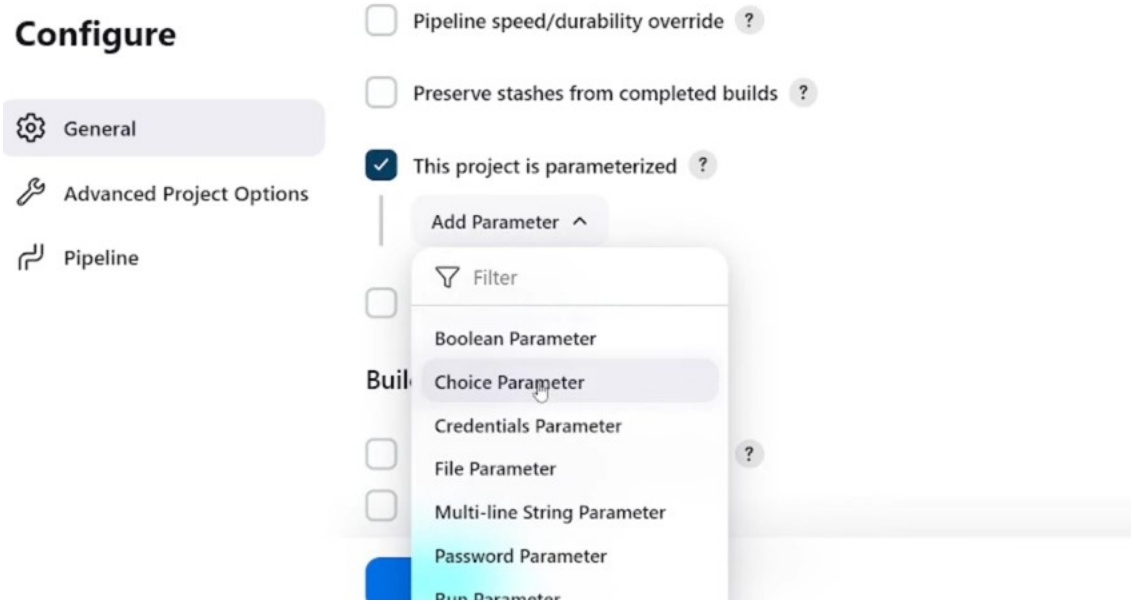


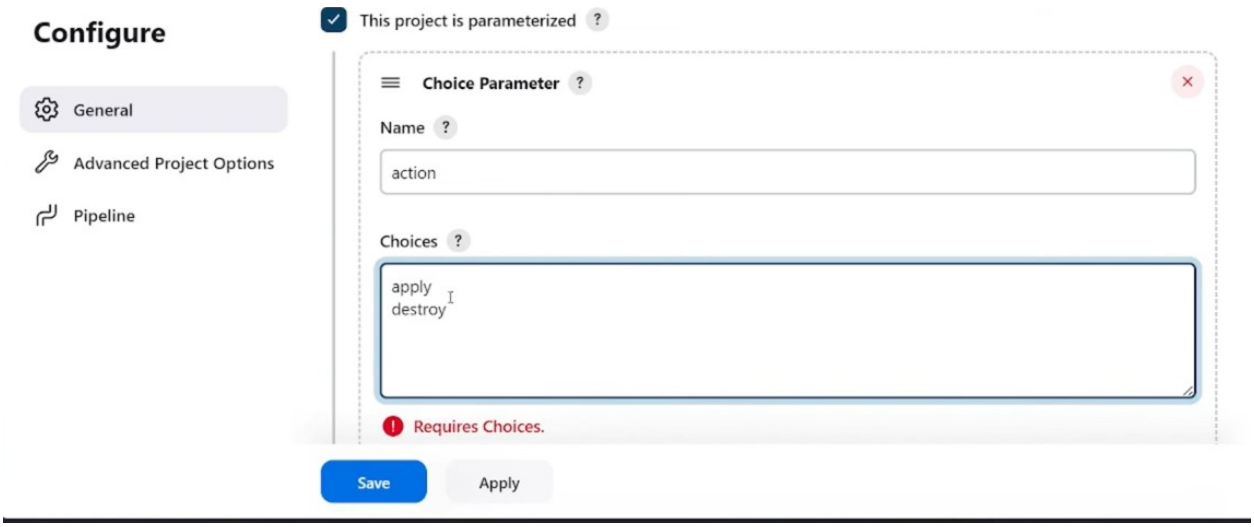


 # EKS Cluster Activated

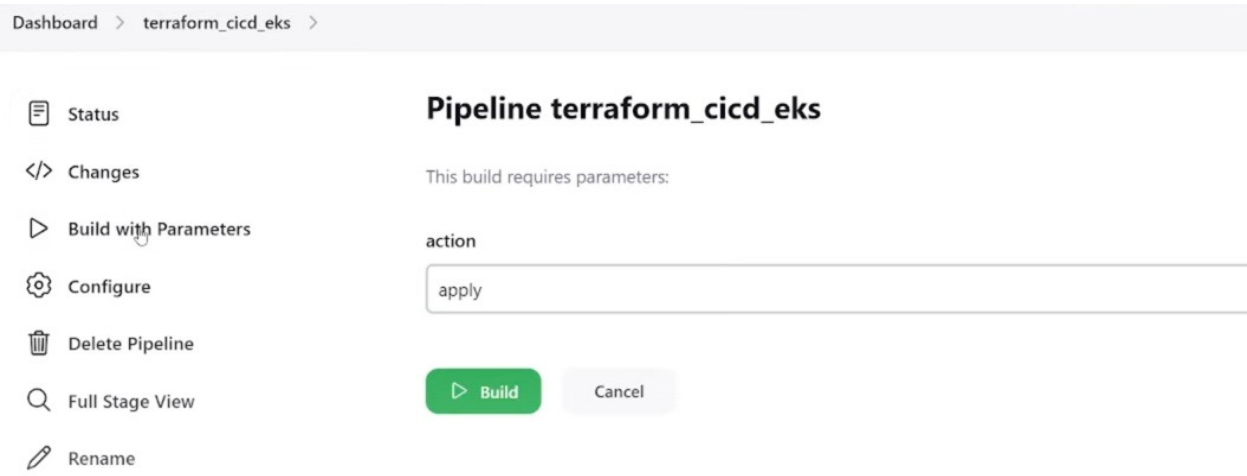


 # To destroy the cluster in Jenkins- Go to configure- General –Project parameterized



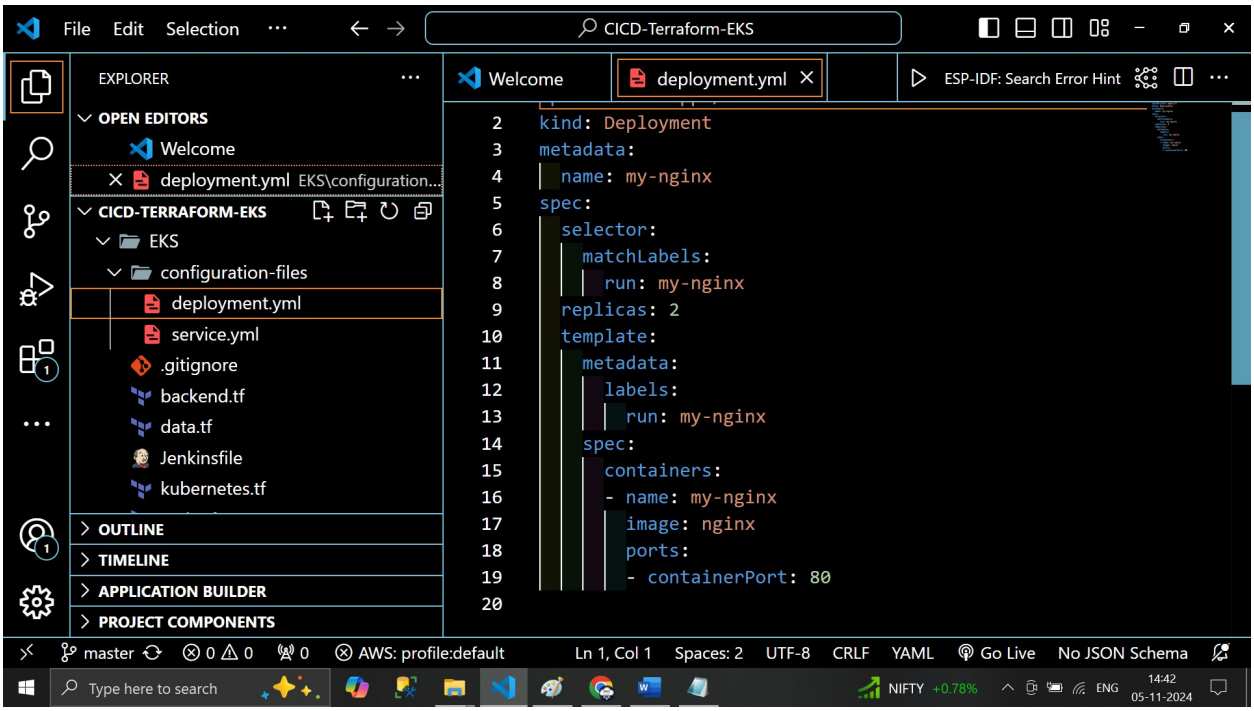


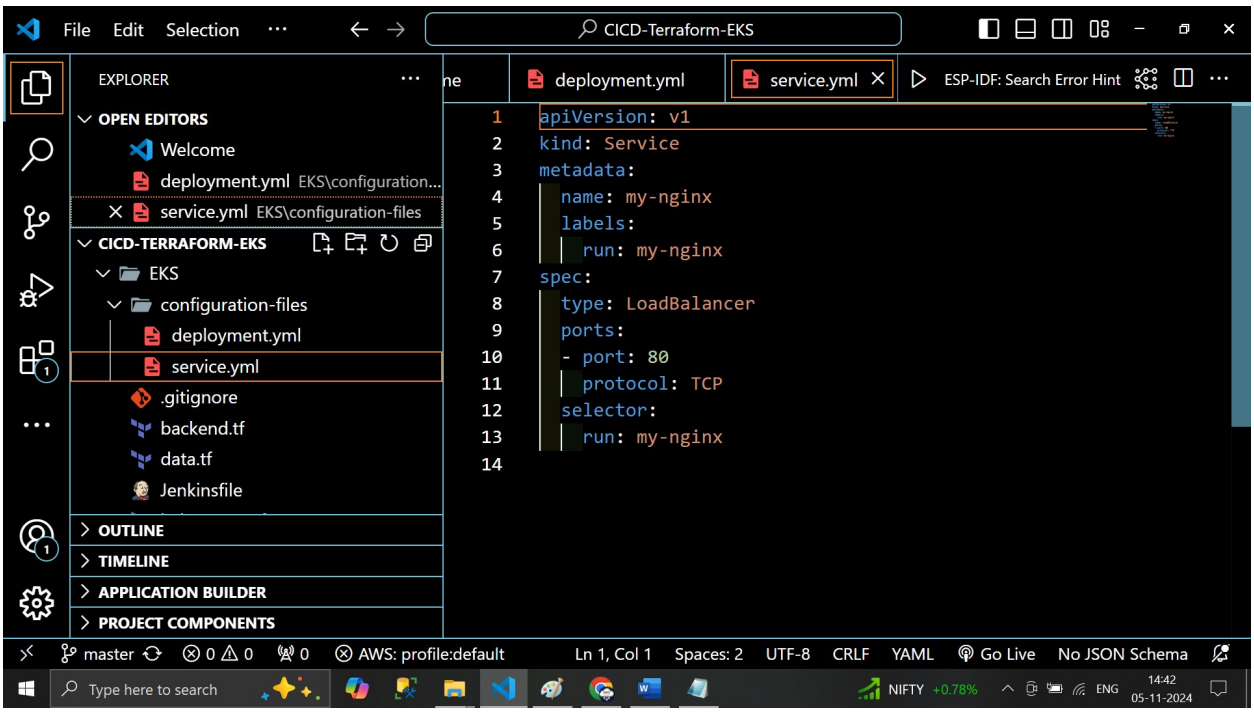
 # Go to Build with parameter (option – apply /destroy)




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

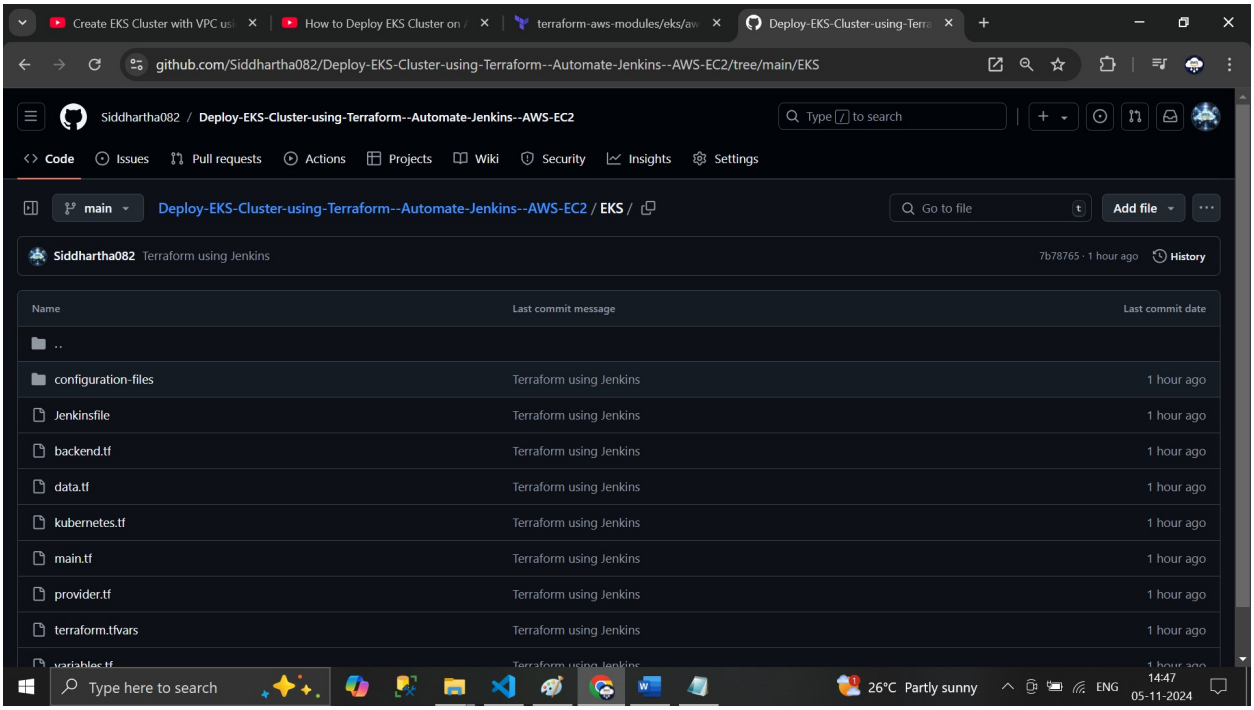
 # now Deploy Nginx app



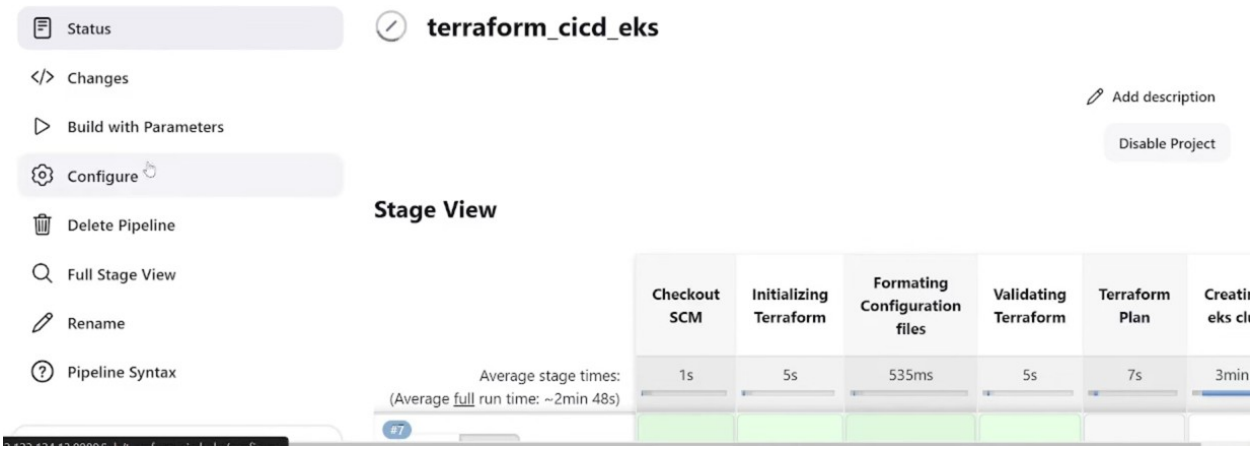


 # Pust the directory to Git repo

```
$ cd EKS/

      LAPTOP-JN110945 MINGW64 ~/OneDrive/Desktop/Terraform_EKS/EKS (ma
$ 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 (ma
$ git add configuration-files/

     r@LAPTOP-JN110945 MINGW64 ~/OneDrive/Desktop/Terraform_EKS/EKS (ma
$ 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 (ma
$ git push origin master
```



 # Add one more stage

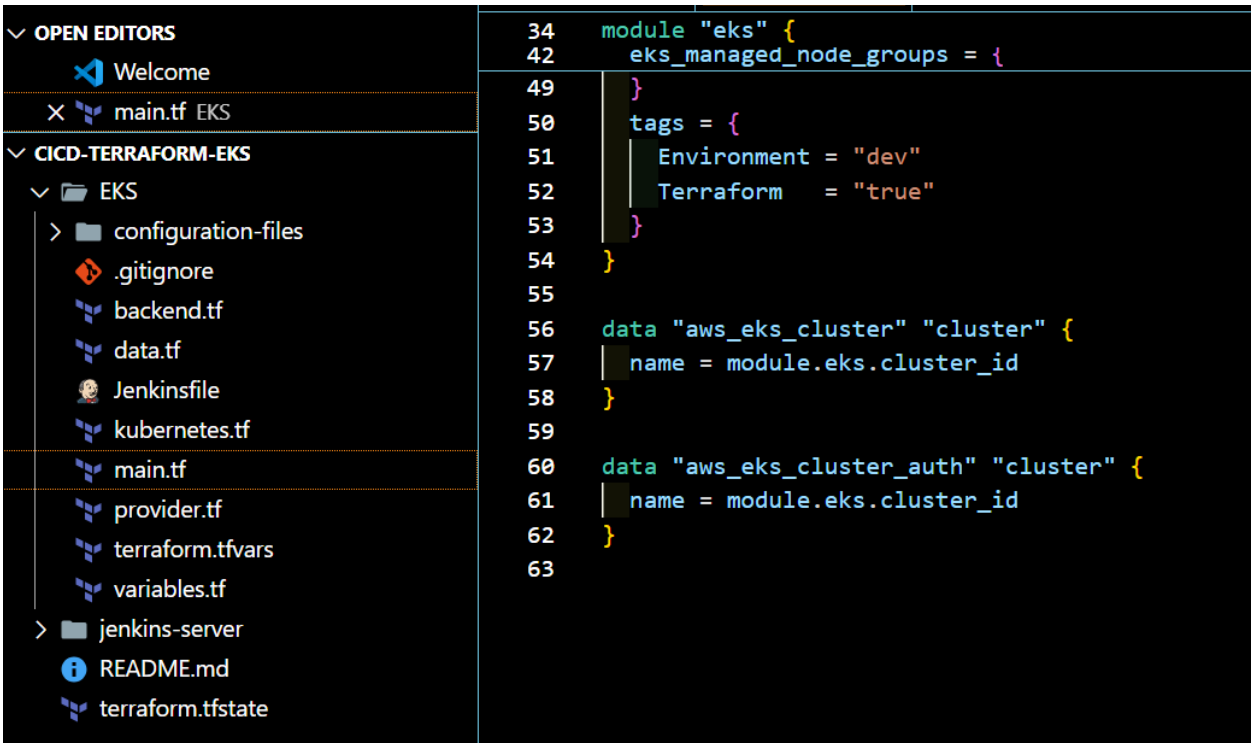



```
        }
        stage("Deploying Nginx"){
            steps{
                script{
                    dir('EKS/configuration-files'){
                        sh 'aws eks update-kubeconfig --name my-eks-cluster'
                        sh 'kubectl apply -f deployment.yml'
                   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

 # Error

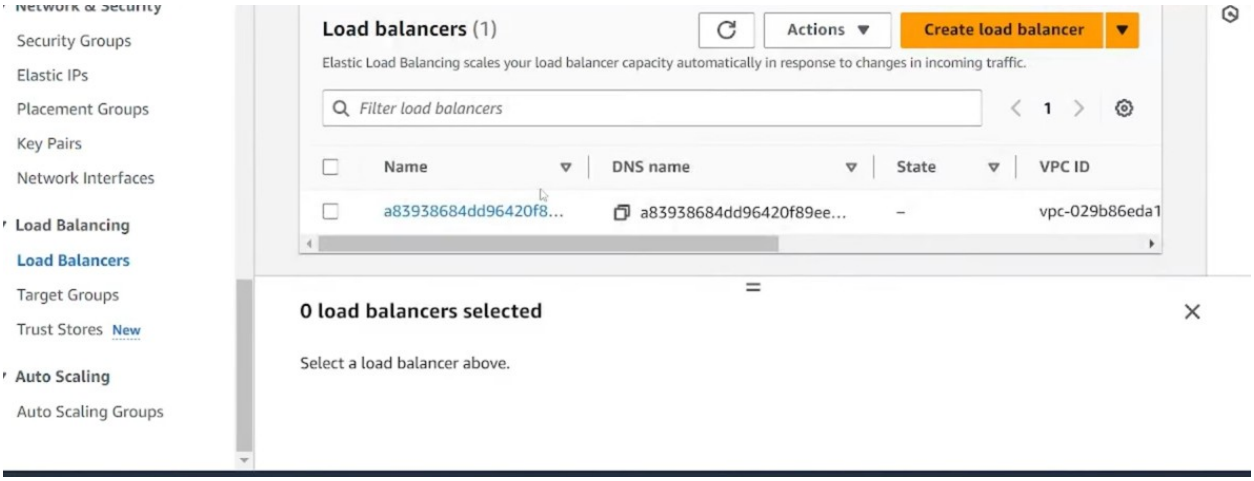
 # user Does not have API Permission to deploy this app on EKS cluster

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

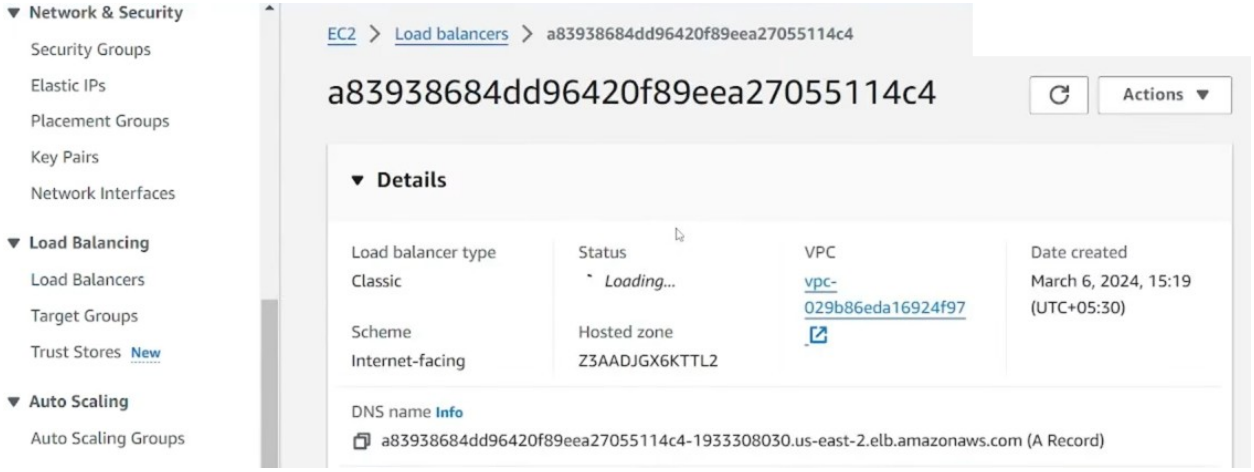


 # Now Build with parameter – apply

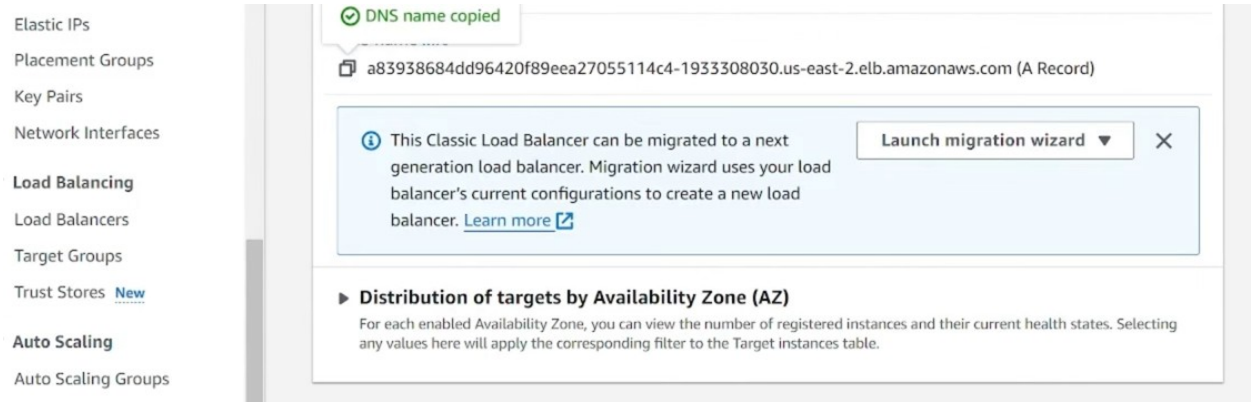
 # Check in aws Load balancer – cp the link & paste it in Browser



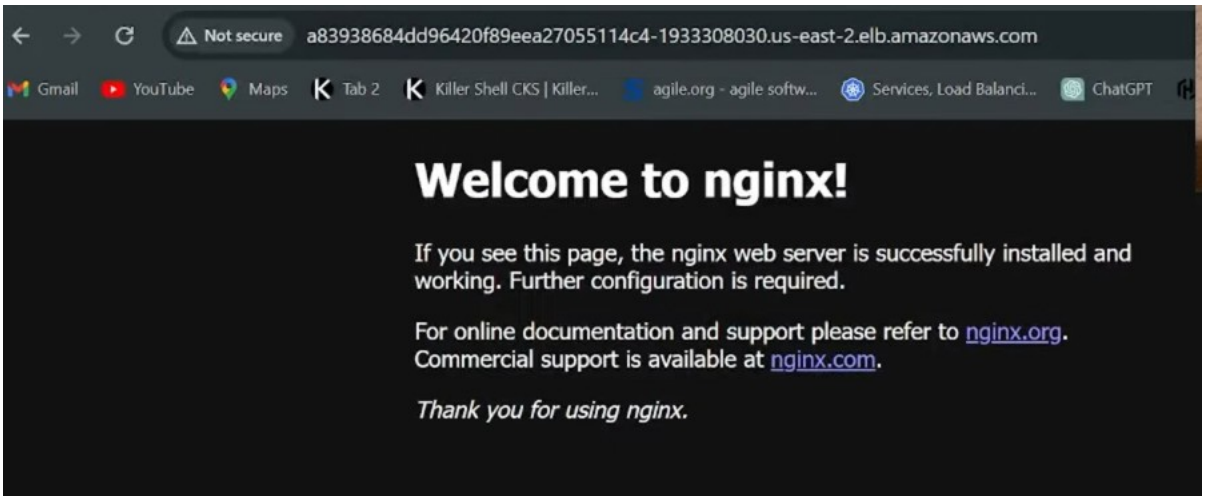
 # cp dns



h



 # final O/P


Not secure
a83938684dd96420f89eea27055114c4-1933308030.us-east-2.elb.amazonaws.com
Gmail
YouTube
Maps
Tab 2
Killer Shell CKS | Killer...
agile.org - agile softw...
Services, Load Balanci...
ChatGPT
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.