Advanced DevOps Practical Report

Sanket More
D15A 30

Case Study:-
Building and Managing Multi-Cloud Infrastructure
● Concepts Used: Terraform, Kubernetes, and AWS S3.
● Problem Statement: "Using Terraform, create and manage multi-cloud resources, including an AWS S3 bucket and a Kubernetes cluster on AWS. Deploy a sample application on the Kubernetes cluster."
● Tasks:
  ○ Write a Terraform script to create an S3 bucket on AWS and a Kubernetes cluster on AWS.
  ○ Deploy a simple application (e.g., a Node.js app) on the Kubernetes cluster.
  ○ Verify that the S3 bucket and the Kubernetes cluster are working as expected.

Case Study Overview: Building and Managing Multi-Cloud Infrastructure

In the rapidly evolving tech landscape, organizations are increasingly adopting multi-cloud strategies to enhance flexibility, minimize vendor lock-in, and optimize costs. This case study focuses on the practical implementation of such a strategy using Terraform, a powerful Infrastructure as Code (IaC) tool, alongside Kubernetes and Amazon Web Services (AWS).

Objectives

1. **Resource Creation**:
   - **AWS S3 Bucket**: Provision an S3 bucket to serve as a storage solution for assets such as application data, logs, or static files.
   - **Kubernetes Cluster on AWS**: Set up a managed Kubernetes cluster using AWS EKS (Elastic Kubernetes Service) to facilitate scalable and efficient application deployment.
2. **Application Deployment**:

    - Deploy a simple Node.js application that serves as a demonstration of the cluster's capabilities. This application will handle basic HTTP requests and can be configured to interact with the S3 bucket for data storage.
3. **Verification**:
    - Validate the operational status of both the S3 bucket and the Kubernetes cluster. This includes ensuring that the application is accessible and that it can successfully store and retrieve data from S3.

Tools and Technologies

- **Terraform**:
    - Terraform allows for the codification of infrastructure, enabling version control and reproducibility. This approach minimizes human error and streamlines the process of setting up cloud resources.
- **Kubernetes**:
    - A container orchestration platform that automates the deployment, scaling, and management of containerized applications. Kubernetes provides the necessary framework to manage the lifecycle of applications in a cloud-native environment.
- **AWS S3**:
    - Amazon S3 is a highly durable and available object storage service designed for scalability and security. It is ideal for storing application data, backups, and static content.

Problem Statement

The primary challenge is to create an automated process for provisioning and managing cloud resources using Terraform. This includes writing a Terraform script to:

- Create an AWS S3 bucket for data storage.
- Set up a Kubernetes cluster using AWS EKS.
- Deploy a simple Node.js application onto the Kubernetes cluster, ensuring it interacts with the S3 bucket as intended.

The success of this project will be measured by the successful creation of the resources and the operational deployment of the Node.js application, demonstrating a cohesive multi-cloud infrastructure.

Implementation Steps

1. **Terraform Script Development**:
    - Write Terraform scripts to define the infrastructure components, including AWS provider configuration, S3 bucket resources, and EKS cluster specifications.
    - Implement output variables to easily retrieve and use resource information, such as the S3 bucket URL and cluster endpoint.
2. **Kubernetes Configuration**:
    - Use Kubernetes manifests to define the deployment configuration for the Node.js application, including necessary services and ingress rules.
    - Apply Kubernetes configurations using `kubectl` to deploy the application and manage its lifecycle.
3. **Application Development**:
    - Create a simple Node.js application that can handle HTTP requests. This application should be capable of performing CRUD operations with the S3 bucket, such as uploading files and retrieving stored data.
4. **Verification and Testing**:
    - After deployment, conduct tests to ensure that the Node.js application is running correctly in the Kubernetes cluster.

    - Verify that the application can successfully store and retrieve data from the S3 bucket, confirming that all components of the infrastructure are functioning as expected.

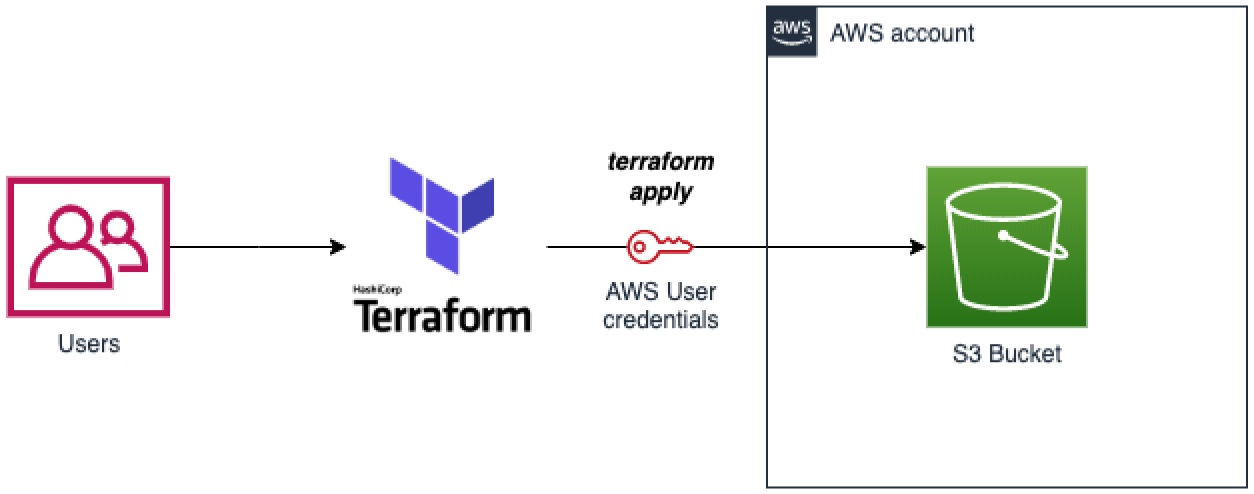
Expected Outcomes

By the conclusion of this case study, participants will have a thorough understanding of how to build and manage a multi-cloud infrastructure using Terraform, Kubernetes, and AWS. They will gain hands-on experience in:

- Writing and executing Terraform scripts to provision cloud resources.
- Deploying and managing containerized applications with Kubernetes.
- Interacting with AWS services like S3 within a cloud-native application context.

This case study serves not only as a practical implementation guide but also as a foundational step toward mastering multi-cloud architectures, preparing participants for real-world cloud engineering challenges.

Terraform script to create an S3 bucket on AWS



Terraform is an Infrastructure as Code (IaC) tool that enables you to define, provision, and manage cloud infrastructure using a declarative configuration language. You can automate the creation of cloud resources, including AWS S3 buckets, with Terraform.

Key Concepts:

- **Providers**: These are plugins that allow Terraform to interact with cloud platforms like AWS. The AWS provider helps you manage AWS services.
- **Resources**: These are the components you define in your Terraform code, such as S3 buckets.
- **State**: Terraform maintains the state of your infrastructure, which allows it to know what resources exist and manage them over time.

Terraform S3 Bucket Creation

1. Infrastructure as Code (IaC)

Terraform enables infrastructure management using code. This approach automates the provisioning of infrastructure and makes it easily reproducible. The code is stored in version control systems (like Git), making collaboration easier and tracking changes over time simple.

2. Provider and Resource

In Terraform, the **provider** block specifies the cloud platform, in this case, AWS, where resources will be managed. The **resource** block represents the specific infrastructure component you want to create—in this case, an S3 bucket.

3. Idempotency

Terraform configurations are idempotent, meaning if you run the same script multiple times, it won't create duplicate resources. Terraform will compare the current state with the desired state defined in your configuration and will only make necessary changes.

4. Version Control

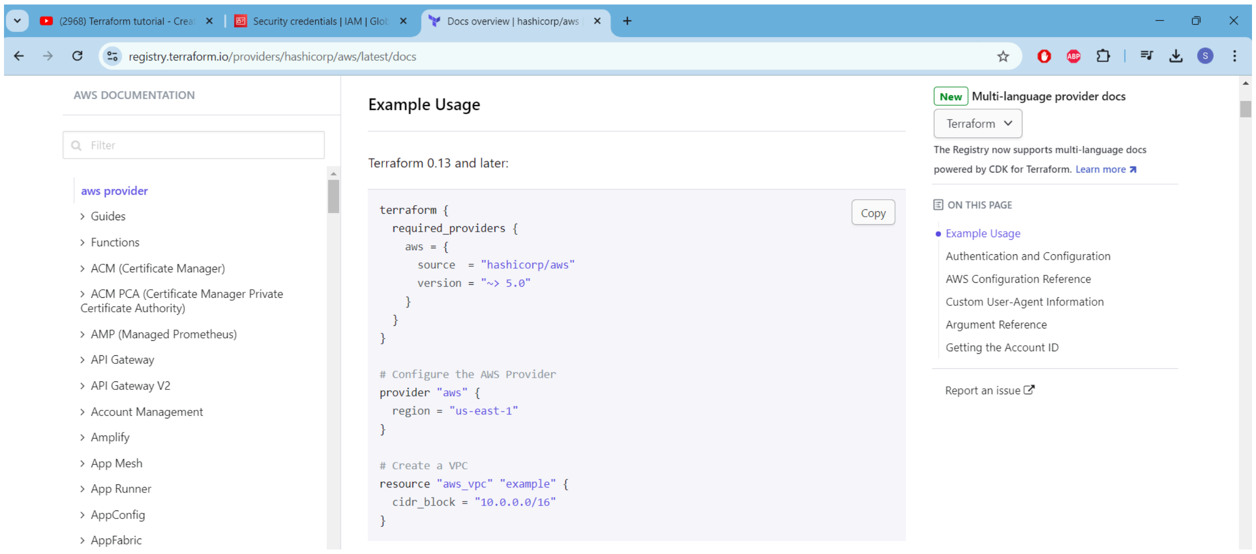
Terraform's state management feature ensures that any changes made to the infrastructure are tracked and managed properly. It stores the infrastructure state in a file (`terraform.tfstate`), which is used for tracking future changes.

5. Benefits of Using Terraform for AWS S3 Buckets

- **Consistency**: Terraform ensures that infrastructure is consistent across environments (e.g., dev, staging, production).
- **Reusability**: You can reuse the same Terraform configuration to deploy multiple S3 buckets.
- **Automation**: Infrastructure creation and management are automated, reducing manual effort.
- **Versioning & Change Management**: Terraform tracks all infrastructure changes in its state file, and you can view differences between desired and current states using `terraform plan`.

Implementation

Using the AWS Hashicorp reference to create main.tf

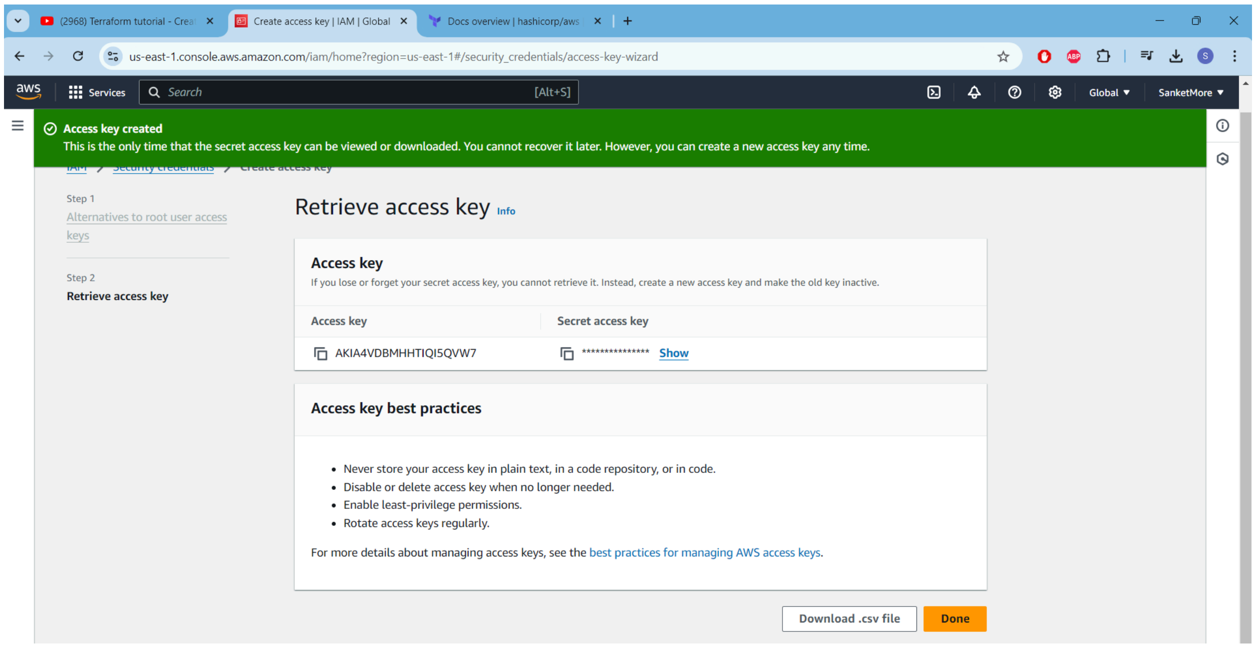

(2968) Terraform tutorial - Crea
Security credentials | IAM | Glol
Docs overview | hashicorp/aws
registry.terraform.io/providers/hashicorp/aws/latest/docs
AWS DOCUMENTATION
Filter
aws provider
Guides
Functions
ACM (Certificate Manager)
ACM PCA (Certificate Manager Private Certificate Authority)
AMP (Managed Prometheus)
API Gateway
API Gateway V2
Account Management
Amplify
App Mesh
App Runner
AppConfig
AppFabric
New Multi-language provider docs
Terraform
The Registry now supports multi-language docs powered by CDK for Terraform. Learn more
ON THIS PAGE
Example Usage
Authentication and Configuration
AWS Configuration Reference
Custom User-Agent Information
Argument Reference
Getting the Account ID
Report an issue
Copy

Example Usage

Terraform 0.13 and later:

```
terraform {
  required_providers {
    aws = {
      source  = "hashicorp/aws"
      version = "~> 5.0"
    }
  }
}

# Configure the AWS Provider
provider "aws" {
  region = "us-east-1"
}

# Create a VPC
resource "aws_vpc" "example" {
  cidr_block = "10.0.0.0/16"
}
```

Creating access keys so that the script can communicate with the AWS account


(2968) Terraform tutorial - Crea
Create access key | IAM | Global
Docs overview | hashicorp/aws
us-east-1.console.aws.amazon.com/iam/home?region=us-east-1#/security_credentials/access-key-wizard
aws
Services
Search
[Alt+S]
Global
SanketMore
IAM
Security credentials
Create access key
Step 1
Alternatives to root user access keys
Step 2
Retrieve access key
Info
Show
Download .csv file
Done

Access key created
This is the only time that the secret access key can be viewed or downloaded. You cannot recover it later. However, you can create a new access key any time.

Retrieve access key

Access key
If you lose or forget your secret access key, you cannot retrieve it. Instead, create a new access key and make the old key inactive.

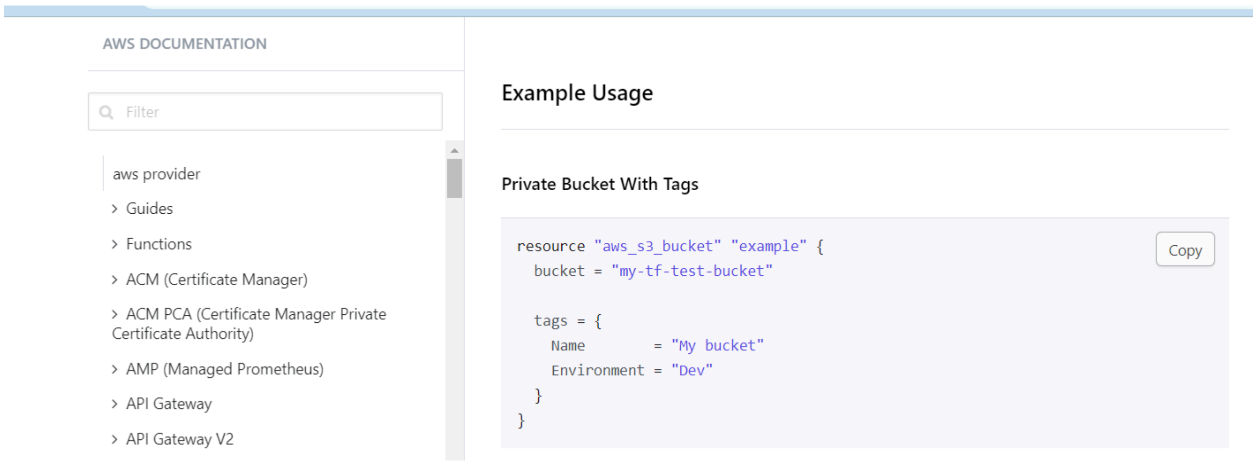
Access key	Secret access key
AKIA4VDBMHHTIQI5QVW7	***************

Access key best practices

- Never store your access key in plain text, in a code repository, or in code.
- Disable or delete access key when no longer needed.
- Enable least-privilege permissions.
- Rotate access keys regularly.

For more details about managing access keys, see the best practices for managing AWS access keys.

Using Hashicorp AWS reference for creating a bucket



AWS DOCUMENTATION

Filter

aws provider

› Guides

› Functions

› ACM (Certificate Manager)

› ACM PCA (Certificate Manager Private Certificate Authority)

› AMP (Managed Prometheus)

› API Gateway

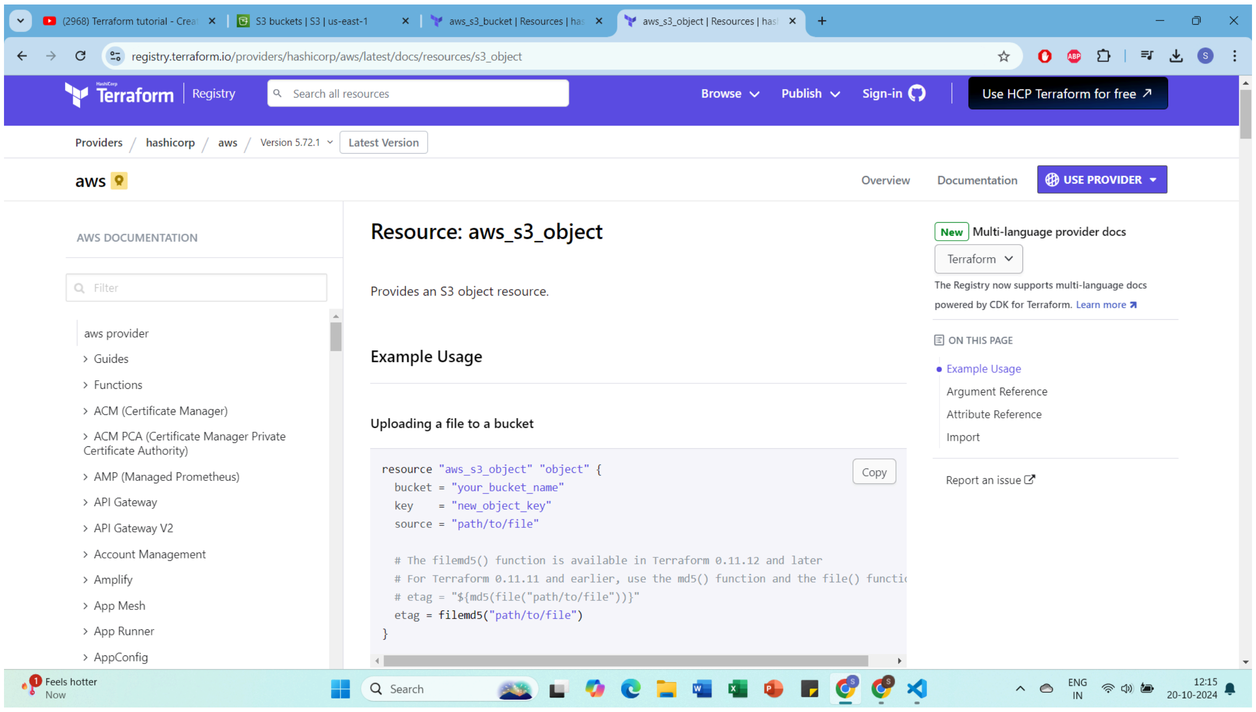
› API Gateway V2

Example Usage

Private Bucket With Tags

```
resource "aws_s3_bucket" "example" {
  bucket = "my-tf-test-bucket"

  tags = {
    Name        = "My bucket"
    Environment = "Dev"
  }
}
```

Hashicorp reference to add something the the created bucket



registry.terraform.io/providers/hashicorp/aws/latest/docs/resources/s3_object

Providers / hashicorp / aws / Version 5.72.1 Latest Version

aws

AWS DOCUMENTATION

aws provider

› Guides

› Functions

› ACM (Certificate Manager)

› ACM PCA (Certificate Manager Private Certificate Authority)

› AMP (Managed Prometheus)

› API Gateway

› API Gateway V2

› Account Management

› Amplify

› App Mesh

› App Runner

› AppConfig

Resource: aws_s3_object

Provides an S3 object resource.

Example Usage

Uploading a file to a bucket

```
resource "aws_s3_object" "object" {
  bucket = "your_bucket_name"
  key    = "new_object_key"
  source = "path/to/file"

  # The filemd5() function is available in Terraform 0.11.12 and later
  # For Terraform 0.11.11 and earlier, use the md5() function and the file() functi
  # etag = "${md5(file("path/to/file"))}"
  etag = filemd5("path/to/file")
}
```

New Multi-language provider docs

The Registry now supports multi-language docs powered by CDK for Terraform. Learn more

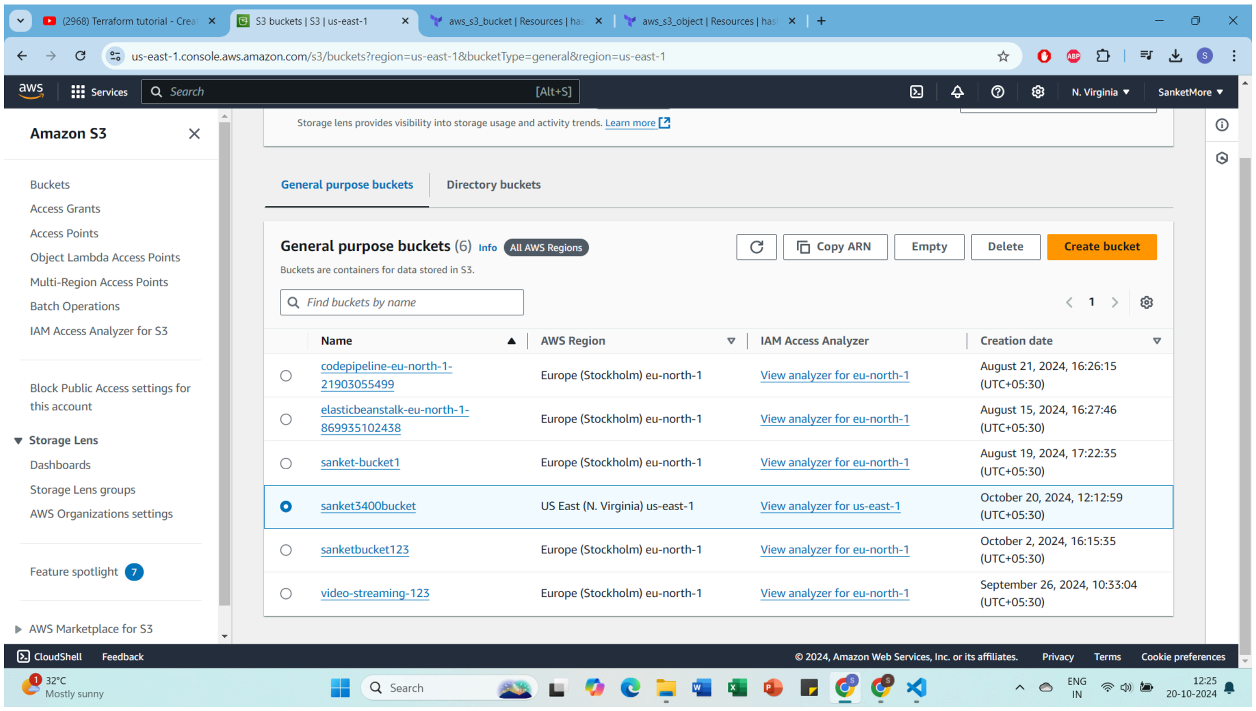
ON THIS PAGE

- Example Usage
- Argument Reference
- Attribute Reference
- Import

Report an issue

(2968) Terraform tutorial - Crea
S3 buckets | S3 | us-east-1
aws_s3_bucket | Resources | has
aws_s3_object | Resources | has
Terraform Registry
Search all resources
Browse
Publish
Sign-in
Use HCP Terraform for free
Overview
Documentation
USE PROVIDER
Terraform
Copy
Feels hotter
Now
Search
ENG IN
12:15
20-10-2024

Bucket created successfully

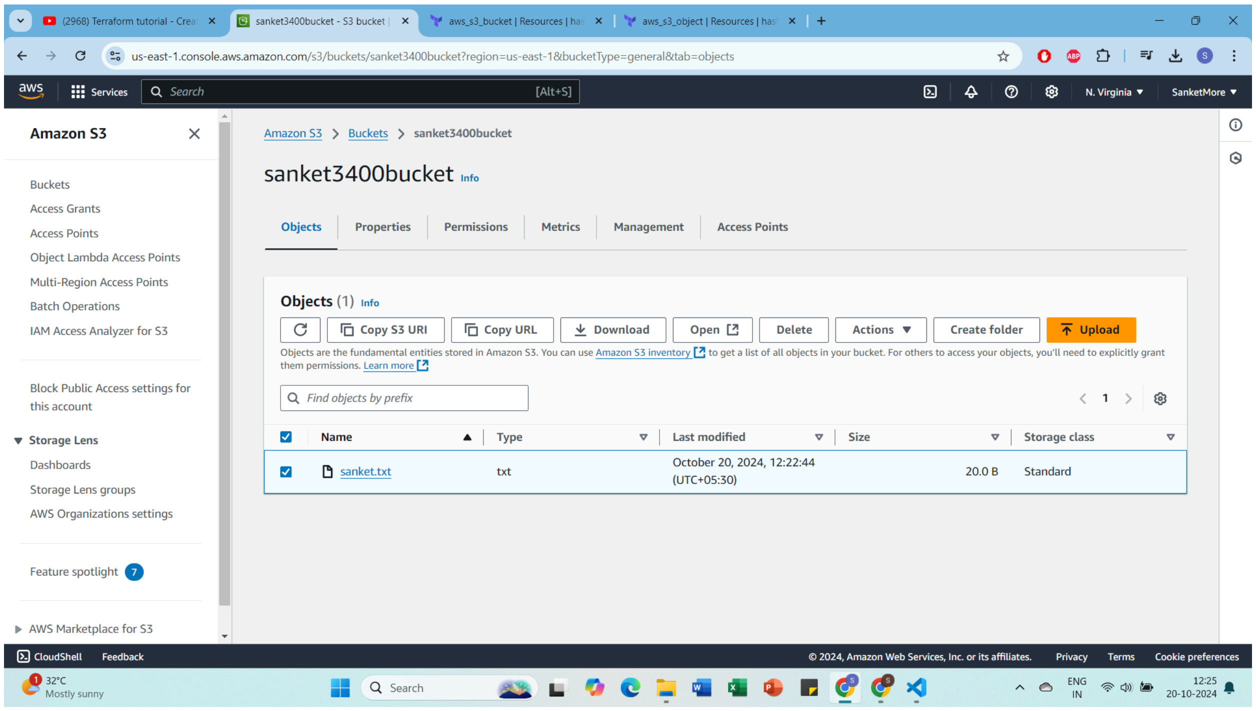


General purpose buckets (6)

Buckets are containers for data stored in S3.

Name	AWS Region	IAM Access Analyzer	Creation date
codepipeline-eu-north-1-21903055499	Europe (Stockholm) eu-north-1	View analyzer for eu-north-1	August 21, 2024, 16:26:15 (UTC+05:30)
elasticbeanstalk-eu-north-1-869935102438	Europe (Stockholm) eu-north-1	View analyzer for eu-north-1	August 15, 2024, 16:27:46 (UTC+05:30)
sanket-bucket1	Europe (Stockholm) eu-north-1	View analyzer for eu-north-1	August 19, 2024, 17:22:35 (UTC+05:30)
sanket3400bucket	US East (N. Virginia) us-east-1	View analyzer for us-east-1	October 20, 2024, 12:12:59 (UTC+05:30)
sanketbucket123	Europe (Stockholm) eu-north-1	View analyzer for eu-north-1	October 2, 2024, 16:15:35 (UTC+05:30)
video-streaming-123	Europe (Stockholm) eu-north-1	View analyzer for eu-north-1	September 26, 2024, 10:33:04 (UTC+05:30)

File added to the bucket successfully



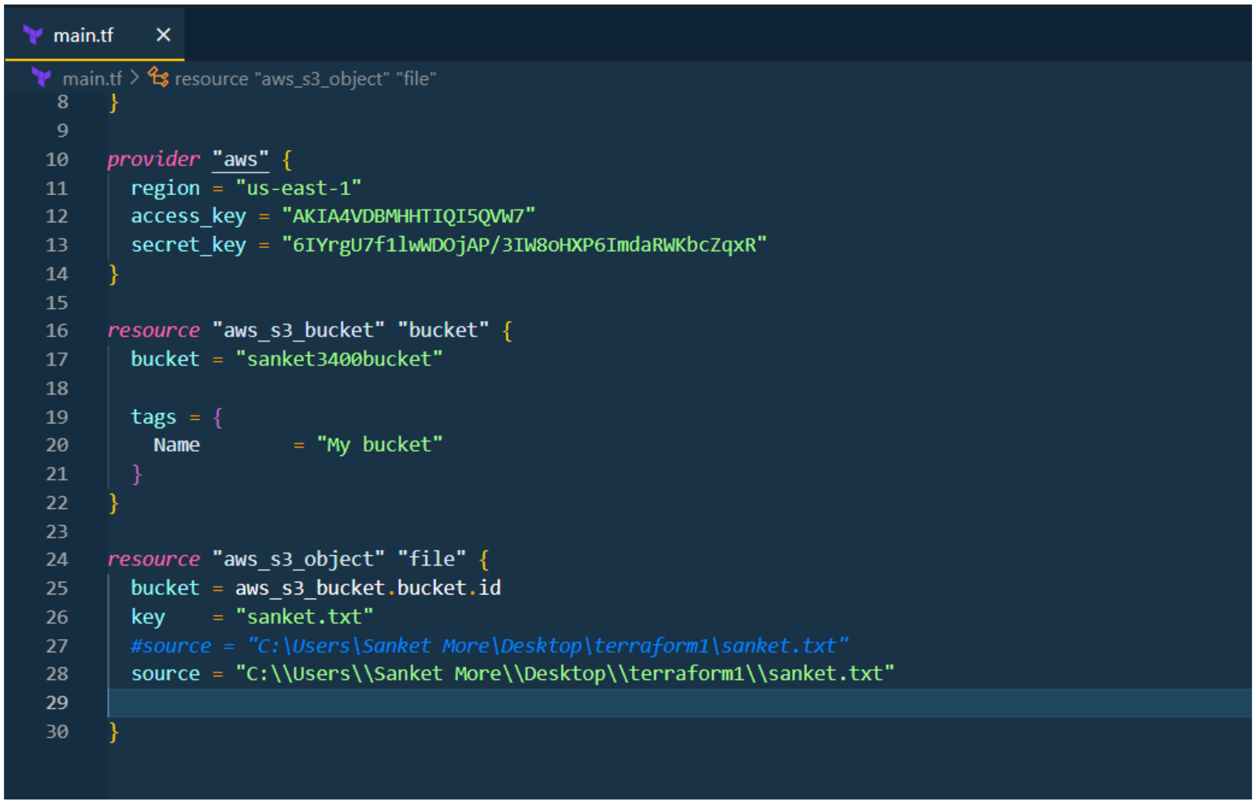
sanket3400bucket

Objects (1)

Objects are the fundamental entities stored in Amazon S3. You can use Amazon S3 inventory to get a list of all objects in your bucket. For others to access your objects, you'll need to explicitly grant them permissions. Learn more

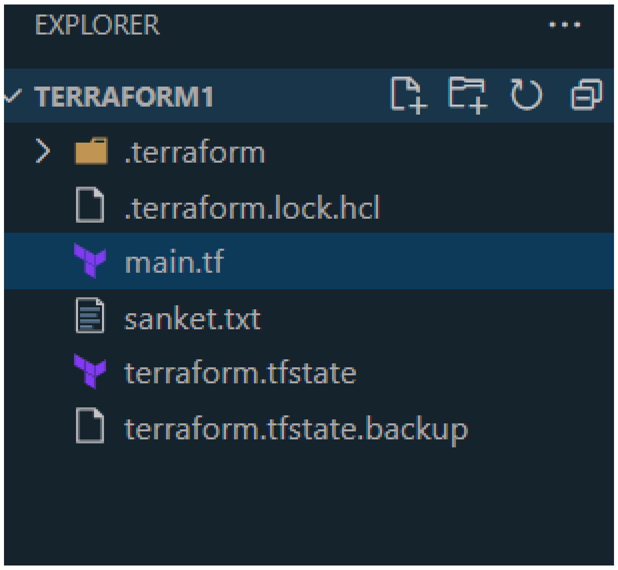
Name	Type	Last modified	Size	Storage class
sanket.txt	txt	October 20, 2024, 12:22:44 (UTC+05:30)	20.0 B	Standard

The main.tf code the create the bucket and add an file to it

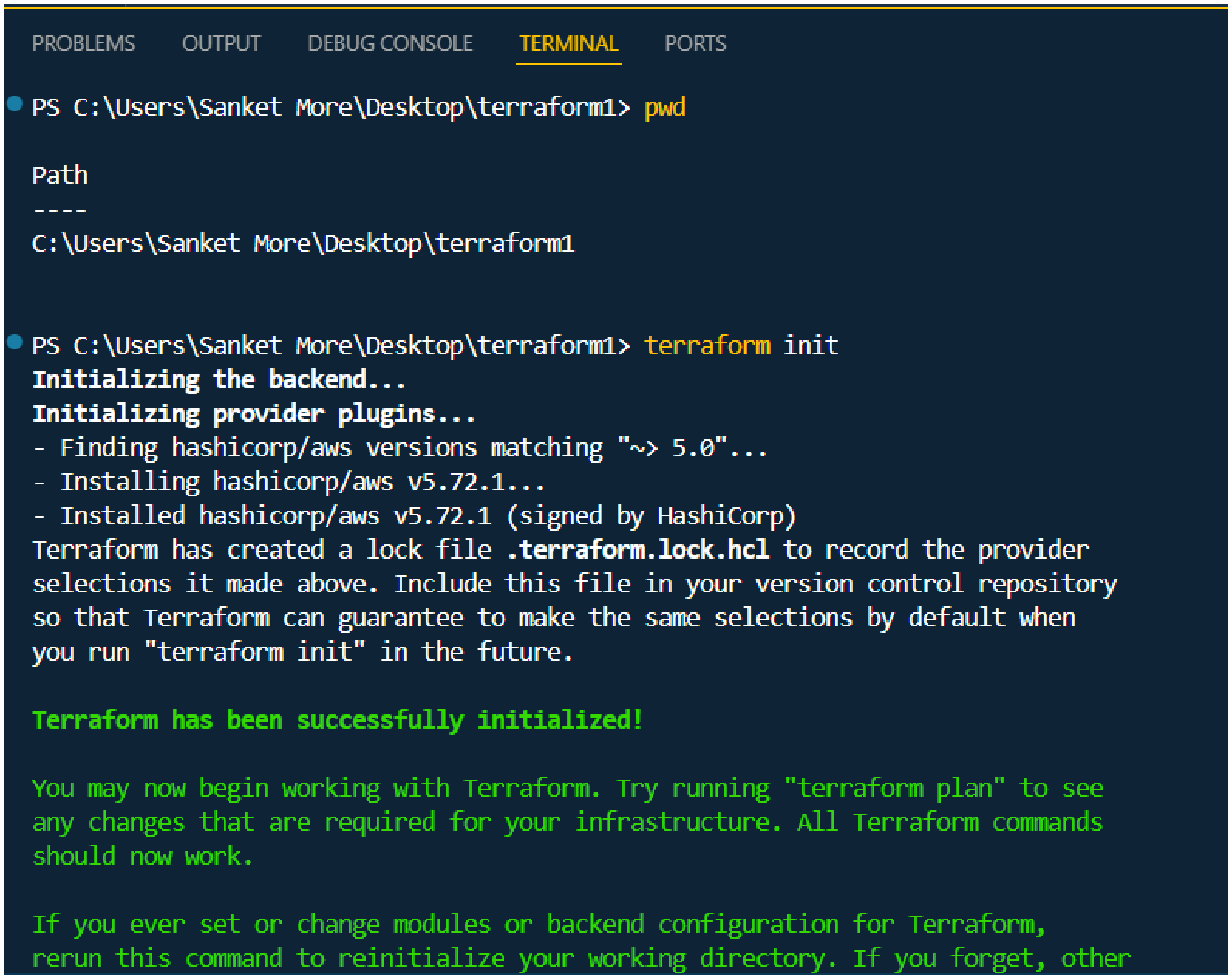


```
}

provider "aws" {
  region = "us-east-1"
  access_key = "AKIA4VDBMHHTIQI5QVW7"
  secret_key = "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"
}

resource "aws_s3_bucket" "bucket" {
  bucket = "sanket3400bucket"

  tags = {
    Name          = "My bucket"
  }
}

resource "aws_s3_object" "file" {
  bucket = aws_s3_bucket.bucket.id
  key    = "sanket.txt"
  #source = "C:\Users\Sanket More\Desktop\terraform1\sanket.txt"
  source = "C:\\Users\\Sanket More\\Desktop\\terraform1\\sanket.txt"

}
```

File structure of the folder

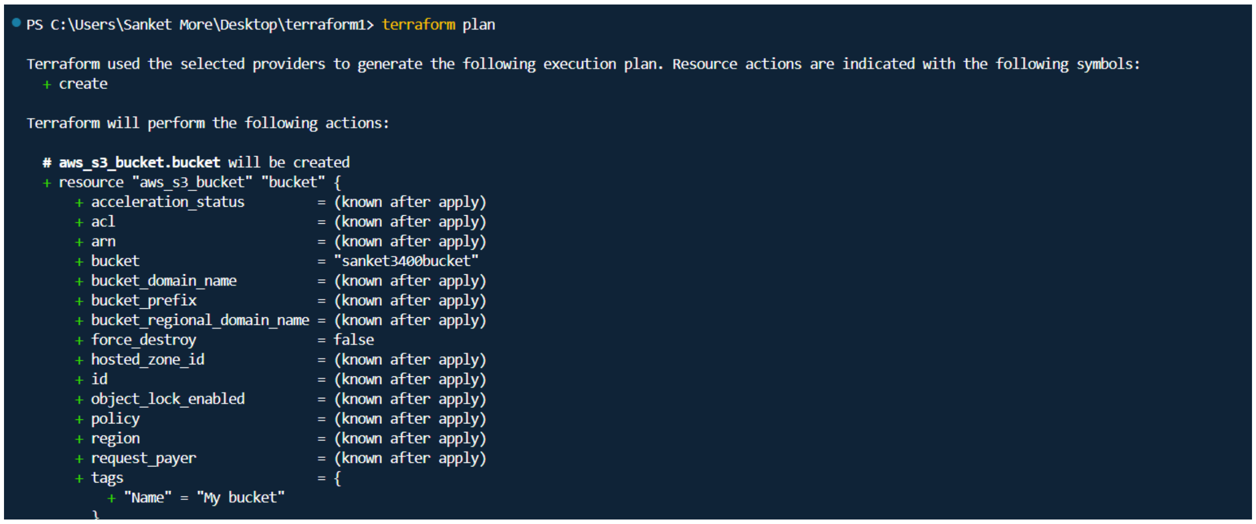


Initializing terraform

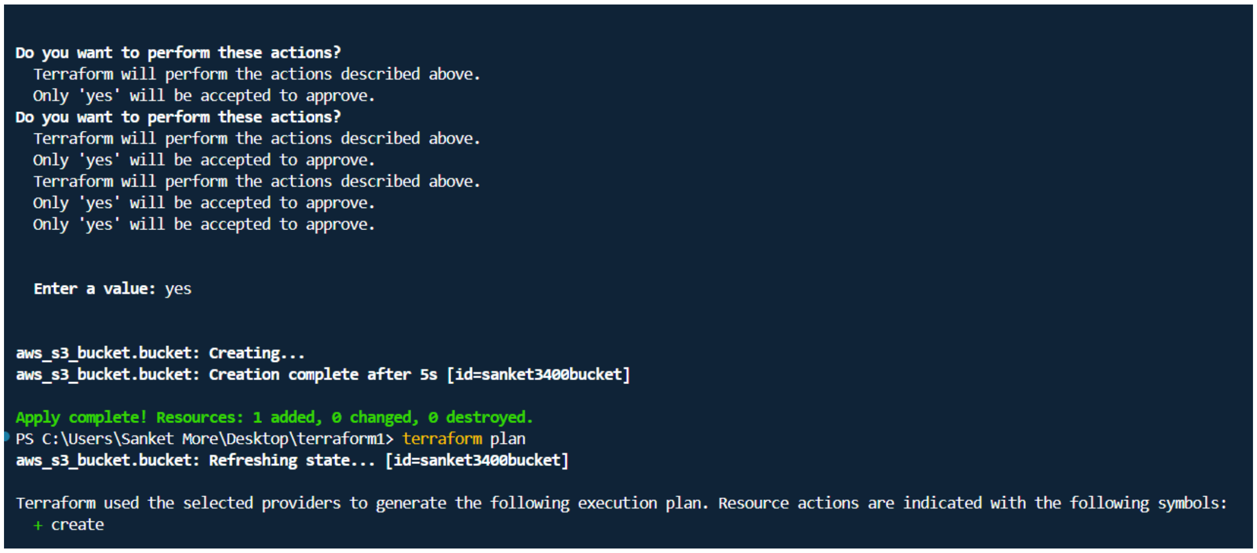


```
PS C:\Users\Sanket More\Desktop\terraform1> pwd

Path
----
C:\Users\Sanket More\Desktop\terraform1


PS C:\Users\Sanket More\Desktop\terraform1> terraform init
Initializing the backend...
Initializing provider plugins...
- Finding hashicorp/aws versions matching "~> 5.0"...
- Installing hashicorp/aws v5.72.1...
- Installed hashicorp/aws v5.72.1 (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
```

Planning the work terraform will be doing

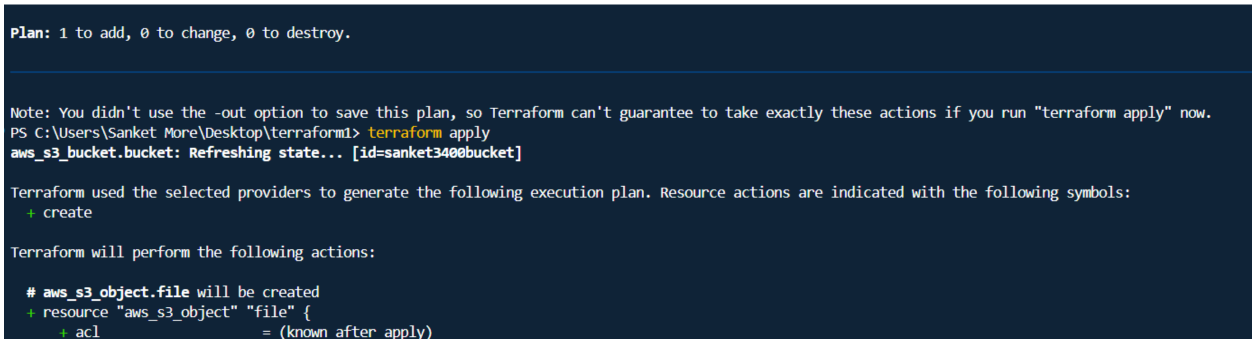


```
PS C:\Users\Sanket More\Desktop\terraform1> terraform plan

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

Terraform will perform the following actions:

  # aws_s3_bucket.bucket will be created
  + resource "aws_s3_bucket" "bucket" {
      + acceleration_status         = (known after apply)
      + acl                         = (known after apply)
      + arn                         = (known after apply)
      + bucket                      = "sanket3400bucket"
      + bucket_domain_name          = (known after apply)
      + bucket_prefix               = (known after apply)
      + bucket_regional_domain_name = (known after apply)
      + force_destroy               = false
      + hosted_zone_id              = (known after apply)
      + id                          = (known after apply)
      + object_lock_enabled         = (known after apply)
      + policy                      = (known after apply)
      + region                      = (known after apply)
      + request_payer               = (known after apply)
      + tags                        = {
          + "Name" = "My bucket"
        }
```



```
Do you want to perform these actions?
  Terraform will perform the actions described above.
  Only 'yes' will be accepted to approve.
Do you want to perform these actions?
  Terraform will perform the actions described above.
  Only 'yes' will be accepted to approve.
  Terraform will perform the actions described above.
  Only 'yes' will be accepted to approve.
  Only 'yes' will be accepted to approve.


  Enter a value: yes


aws_s3_bucket.bucket: Creating...
aws_s3_bucket.bucket: Creation complete after 5s [id=sanket3400bucket]

Apply complete! Resources: 1 added, 0 changed, 0 destroyed.
PS C:\Users\Sanket More\Desktop\terraform1> terraform plan
aws_s3_bucket.bucket: Refreshing state... [id=sanket3400bucket]

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

Refreshing the bucket after adding the text file



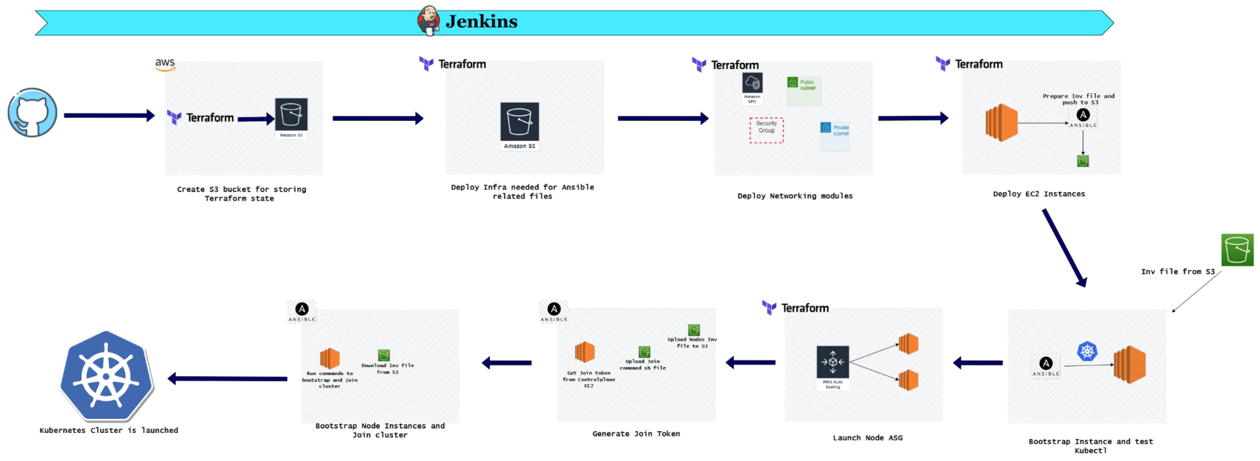
```
Plan: 1 to add, 0 to change, 0 to destroy.


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.
PS C:\Users\Sanket More\Desktop\terraform1> terraform apply
aws_s3_bucket.bucket: Refreshing state... [id=sanket3400bucket]

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

Terraform will perform the following actions:

  # aws_s3_object.file will be created
  + resource "aws_s3_object" "file" {
      + acl                    = (known after apply)
```

Terraform Script to create a kubernetes cluster on AWS

AWS Infrastructure for Kubernetes: Kubernetes requires specific infrastructure components to function on AWS. At the core is the Amazon Elastic Kubernetes Service (EKS), which simplifies Kubernetes management. However, additional AWS resources like EC2 instances, VPCs, subnets, security groups, and IAM roles must be configured for a working K8s cluster.

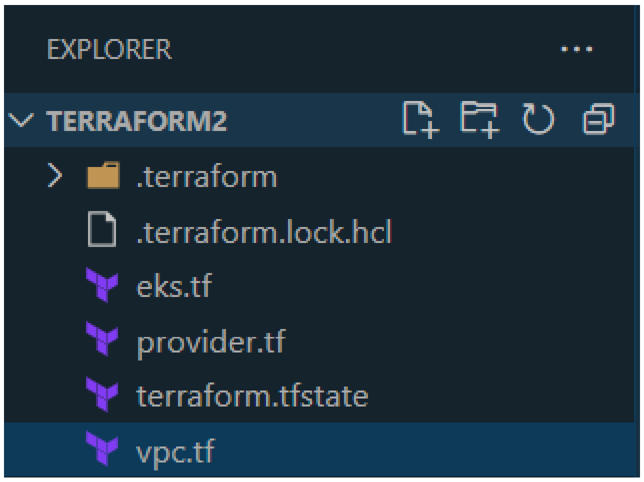
Key Steps in Provisioning:

1. **Infrastructure Definition**: The first step is defining the AWS infrastructure using Terraform's configuration files. You specify the required resources like VPCs, subnets, and security groups to ensure network connectivity for the Kubernetes cluster. EC2 instances (or managed services like EKS) are configured to serve as Kubernetes nodes.
2. **Networking Setup**: Kubernetes requires network setup to enable communication between pods and services. Terraform provisions a Virtual Private Cloud (VPC) with subnets across multiple availability zones to ensure high availability. Internet Gateways and NAT gateways allow external and internal network traffic.
3. **Node Group Provisioning**: Kubernetes nodes, which are typically EC2 instances in AWS, need to be provisioned. These nodes run your containerized workloads and communicate within the cluster. Using Terraform, you can define auto-scaling node groups to automatically adjust the number of nodes based on the workload demand.
4. **Security Configuration**: Security in Kubernetes involves both IAM roles and policies for controlling access, and security groups for managing network traffic. Terraform automates the creation of appropriate IAM roles and policies for the EKS control plane and worker nodes, ensuring secure communication between resources.
5. **Kubernetes Cluster Creation**: After setting up the AWS infrastructure, Terraform provisions the EKS cluster itself, which serves as the control plane for Kubernetes operations. Once the cluster is up and running, you can use tools like `kubectl` to manage workloads, deploy applications, and scale services.


Jenkins
aws
Terraform
Amazon S3
Create S3 bucket for storing Terraform state
Terraform
Amazon S3
Deploy Infra needed for Ansible related files
Terraform
Amazon VPC
Public subnet
Security Group
Private subnet
Deploy Networking modules
Terraform
Prepare Inv file and push to S3
ANSIBLE
Deploy EC2 Instances
Inv file from S3
ANSIBLE
Bootstrap Instance and test Kubectl
Terraform
AWS Auto Scaling
Launch Node ASG
ANSIBLE
Upload Nodes Inv file to S3
Upload Join command sh file
Get Join token from Controlplane EC2
Generate Join Token
ANSIBLE
Download Inv file from S3
Run commands to bootstrap and join cluster
Bootstrap Node Instances and Join cluster
Kubernetes Cluster is launched

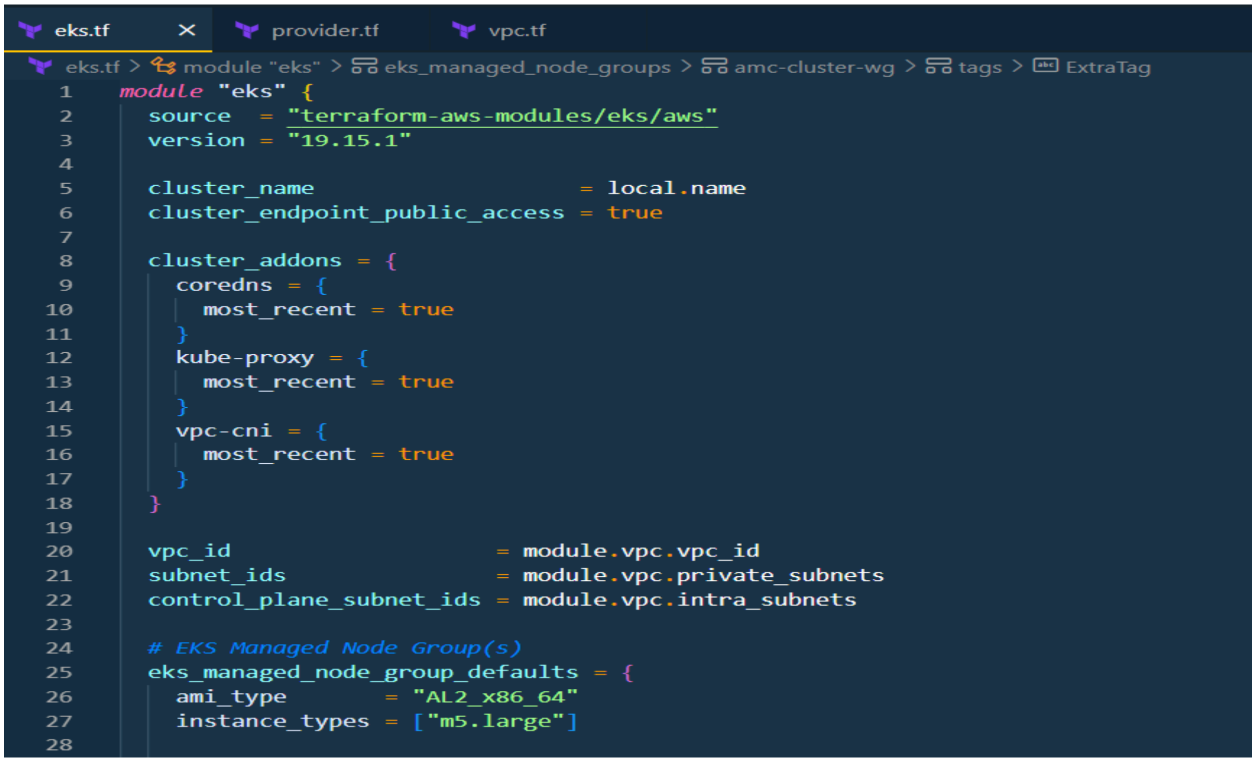
Implementation:-

File structure



```
.terraform
.terraform.lock.hcl
eks.tf
provider.tf
terraform.tfstate
vpc.tf
```

eks.tf file

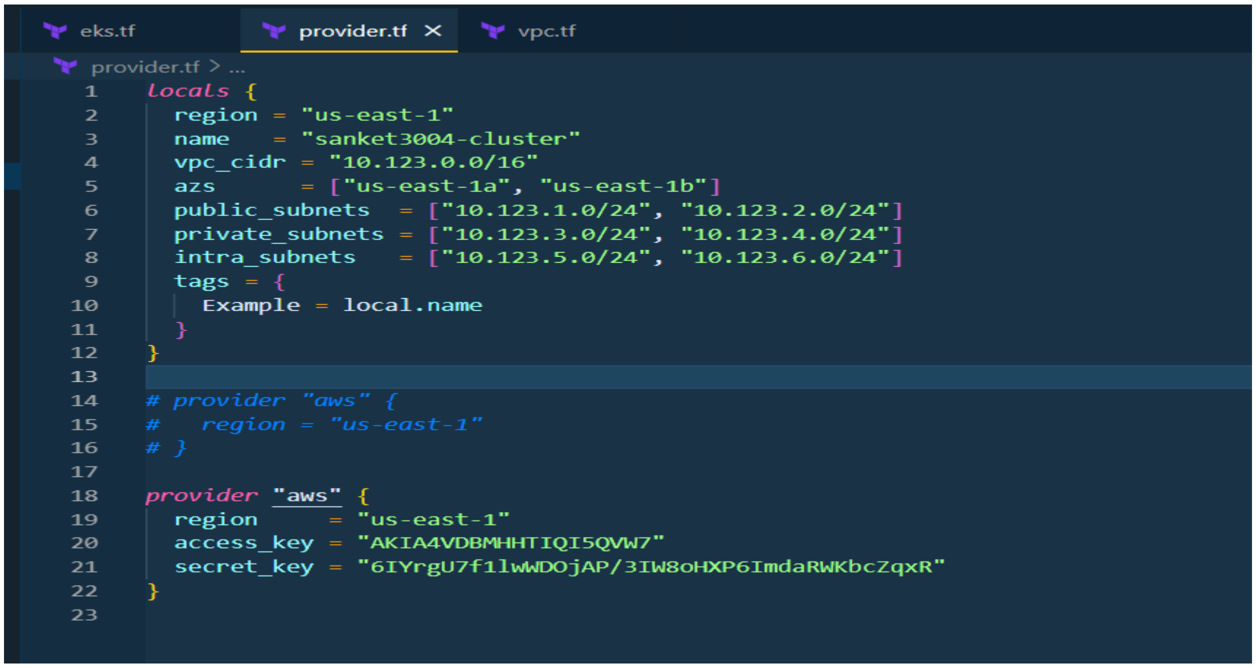


eks.tf × | provider.tf | vpc.tf

eks.tf > module "eks" > eks_managed_node_groups > amc-cluster-wg > tags > ExtraTag

```
module "eks" {
  source  = "terraform-aws-modules/eks/aws"
  version = "19.15.1"

  cluster_name                   = local.name
  cluster_endpoint_public_access = true

  cluster_addons = {
    coredns = {
      most_recent = true
    }
    kube-proxy = {
      most_recent = true
    }
    vpc-cni = {
      most_recent = true
    }
  }

  vpc_id                   = module.vpc.vpc_id
  subnet_ids               = module.vpc.private_subnets
  control_plane_subnet_ids = module.vpc.intra_subnets

  # EKS Managed Node Group(s)
  eks_managed_node_group_defaults = {
    ami_type       = "AL2_x86_64"
    instance_types = ["m5.large"]

```

provider.tf file

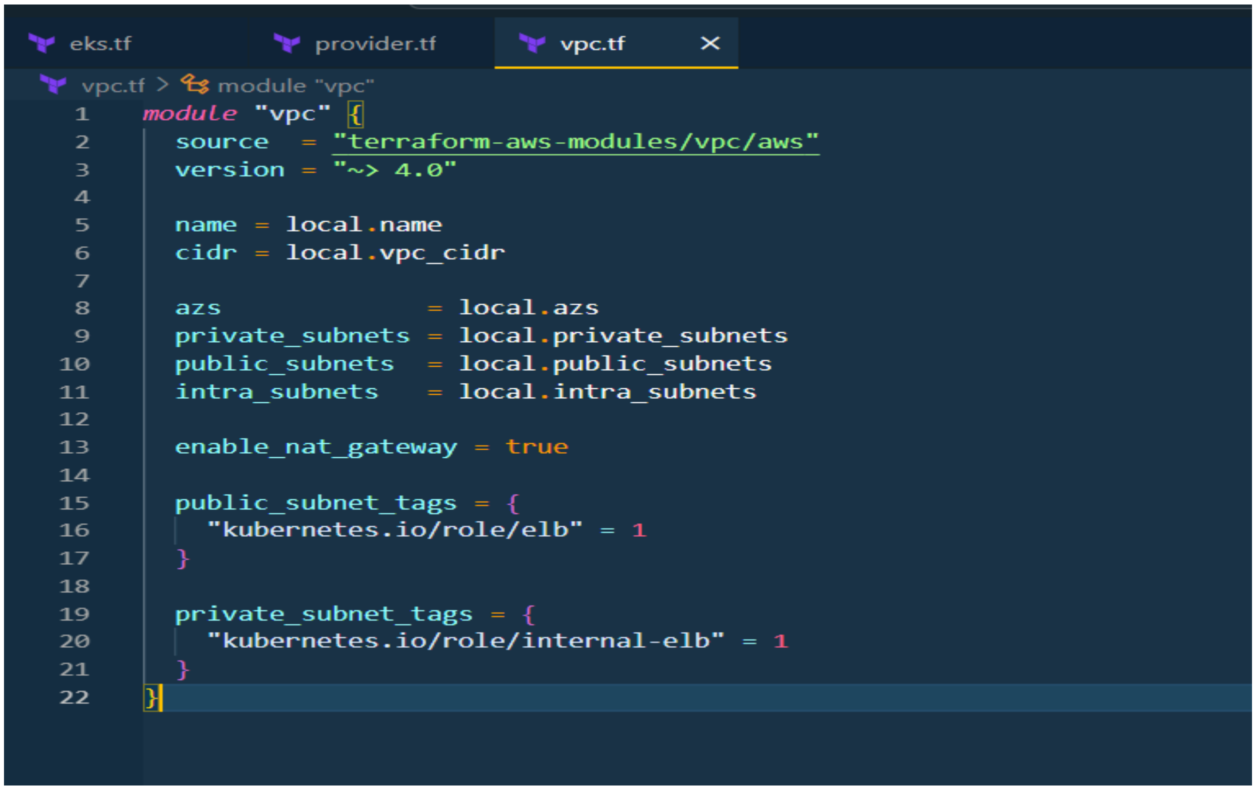


eks.tf | provider.tf × | vpc.tf

provider.tf > ...

```
locals {
  region = "us-east-1"
  name   = "sanket3004-cluster"
  vpc_cidr = "10.123.0.0/16"
  azs      = ["us-east-1a", "us-east-1b"]
  public_subnets  = ["10.123.1.0/24", "10.123.2.0/24"]
  private_subnets = ["10.123.3.0/24", "10.123.4.0/24"]
  intra_subnets   = ["10.123.5.0/24", "10.123.6.0/24"]
  tags = {
    Example = local.name
  }
}

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

provider "aws" {
  region     = "us-east-1"
  access_key = "AKIA4VDBMHHTIQI5QVW7"
  secret_key = "6IYrgU7f1lwWDOjAP/3IW8oHXP6ImdaRWKbcZqxR"
}

```

vpc.tf file

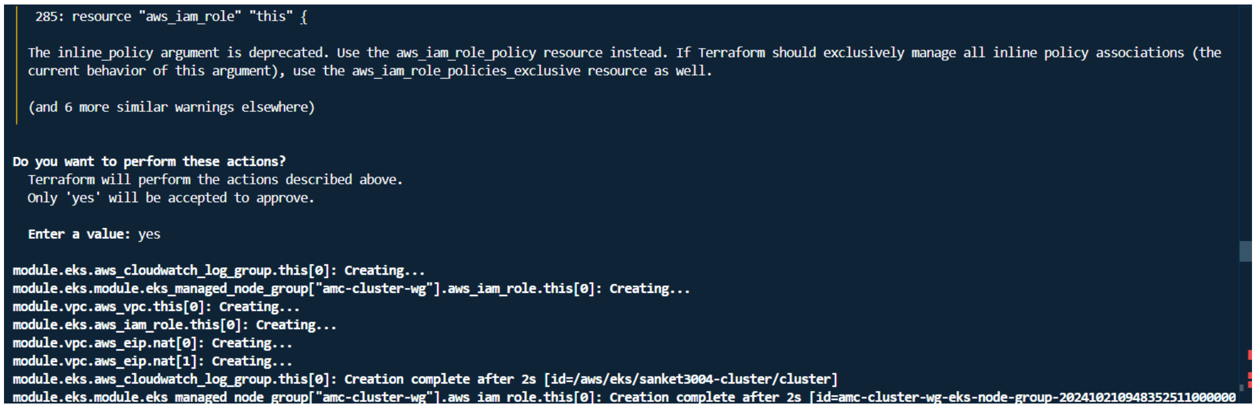


eks.tf   provider.tf   vpc.tf

vpc.tf > module "vpc"

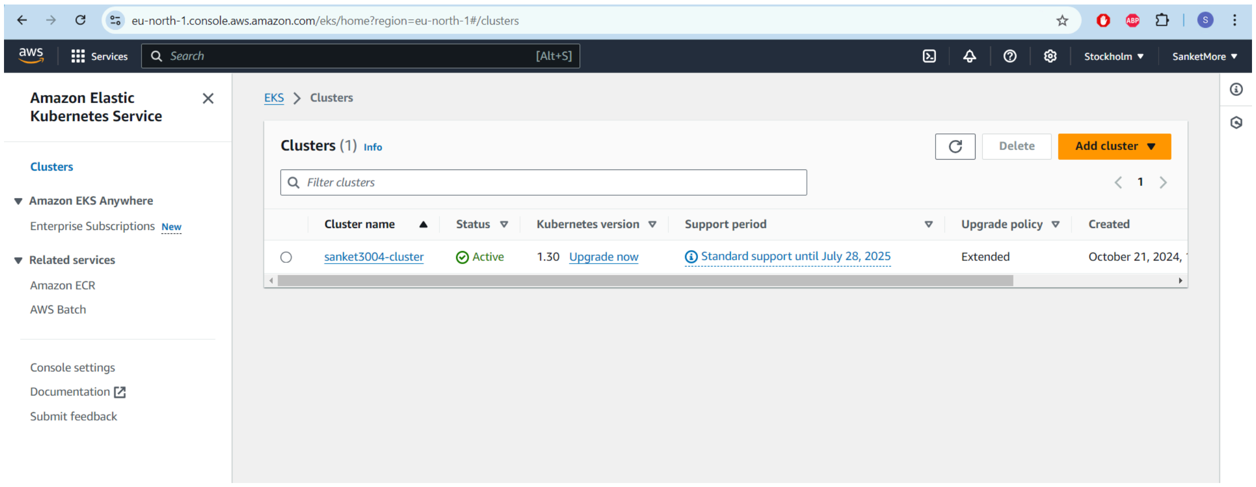
```
module "vpc" {
  source  = "terraform-aws-modules/vpc/aws"
  version = "~> 4.0"

  name = local.name
  cidr = local.vpc_cidr

  azs             = local.azs
  private_subnets = local.private_subnets
  public_subnets  = local.public_subnets
  intra_subnets   = local.intra_subnets

  enable_nat_gateway = true

  public_subnet_tags = {
    "kubernetes.io/role/elb" = 1
  }

  private_subnet_tags = {
    "kubernetes.io/role/internal-elb" = 1
  }
}
```

Creating kubernetes cluster using terraform



```
  285: resource "aws_iam_role" "this" {

 The inline_policy argument is deprecated. Use the aws_iam_role_policy resource instead. If Terraform should exclusively manage all inline policy associations (the
 current behavior of this argument), use the aws_iam_role_policies_exclusive resource as well.

 (and 6 more similar warnings elsewhere)


Do you want to perform these actions?
  Terraform will perform the actions described above.
  Only 'yes' will be accepted to approve.

  Enter a value: yes

module.eks.aws_cloudwatch_log_group.this[0]: Creating...
module.eks.module.eks_managed_node_group["amc-cluster-wg"].aws_iam_role.this[0]: Creating...
module.vpc.aws_vpc.this[0]: Creating...
module.eks.aws_iam_role.this[0]: Creating...
module.vpc.aws_eip.nat[0]: Creating...
module.vpc.aws_eip.nat[1]: Creating...
module.eks.aws_cloudwatch_log_group.this[0]: Creation complete after 2s [id=/aws/eks/sanket3004-cluster/cluster]
module.eks.module.eks_managed_node_group["amc-cluster-wg"].aws_iam_role.this[0]: Creation complete after 2s [id=amc-cluster-wg-eks-node-group-20241021094835251100000
```

Cluster has been successfully created



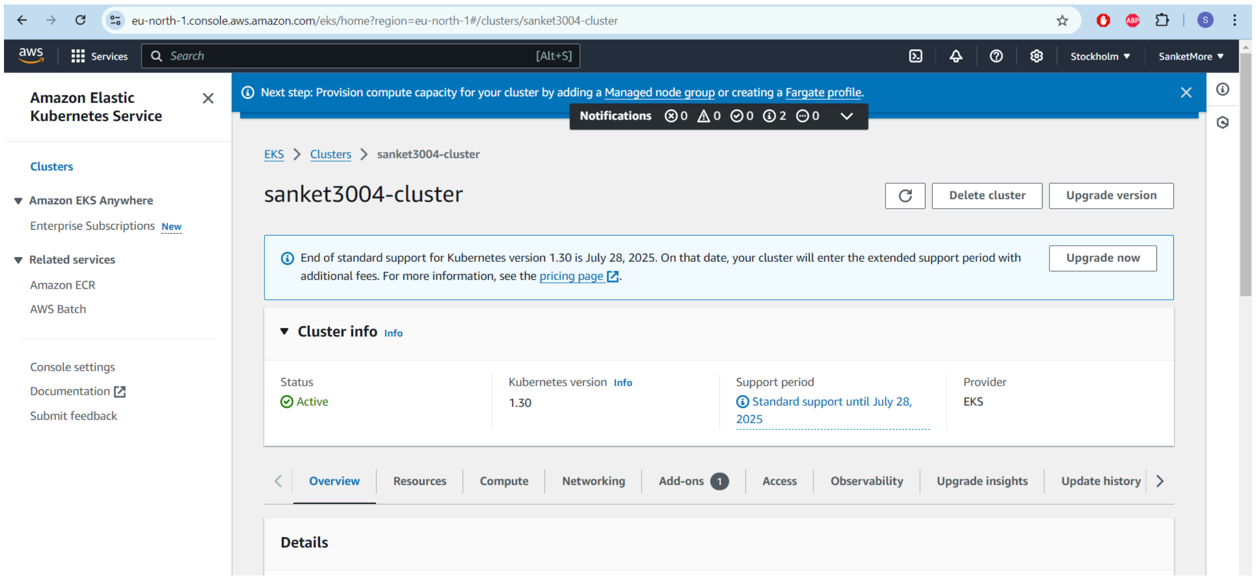
eu-north-1.console.aws.amazon.com/eks/home?region=eu-north-1#/clusters

Amazon Elastic Kubernetes Service

EKS > Clusters

Clusters (1) Info

Cluster name	Status	Kubernetes version	Support period	Upgrade policy	Created
sanket3004-cluster	Active	1.30 Upgrade now	Standard support until July 28, 2025	Extended	October 21, 2024,



eu-north-1.console.aws.amazon.com/eks/home?region=eu-north-1#/clusters/sanket3004-cluster

Next step: Provision compute capacity for your cluster by adding a Managed node group or creating a Fargate profile.

EKS > Clusters > sanket3004-cluster

sanket3004-cluster

End of standard support for Kubernetes version 1.30 is July 28, 2025. On that date, your cluster will enter the extended support period with additional fees. For more information, see the pricing page.

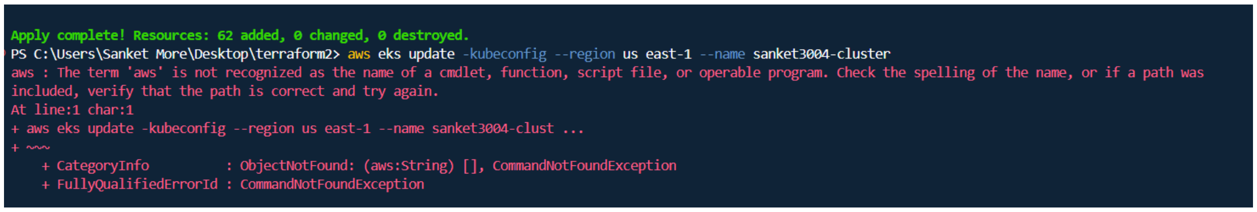
Cluster info Info

Status	Kubernetes version	Support period	Provider
Active	1.30	Standard support until July 28, 2025	EKS

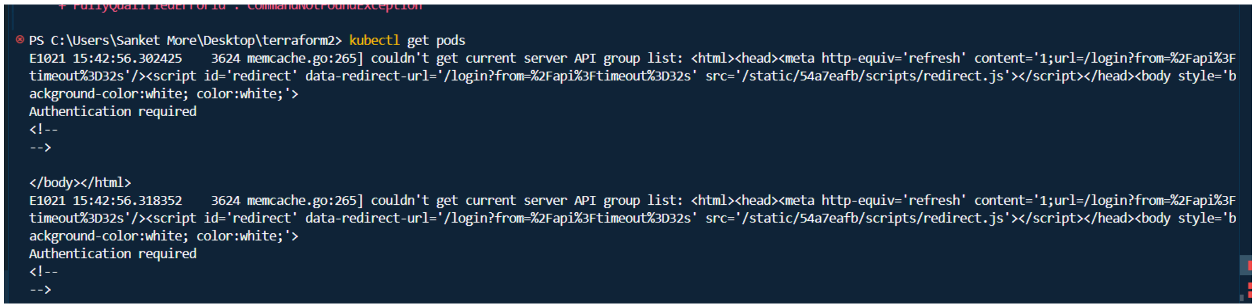
Overview | Resources | Compute | Networking | Add-ons 1 | Access | Observability | Upgrade insights | Update history

Details

Getting error trying to configure the cluster

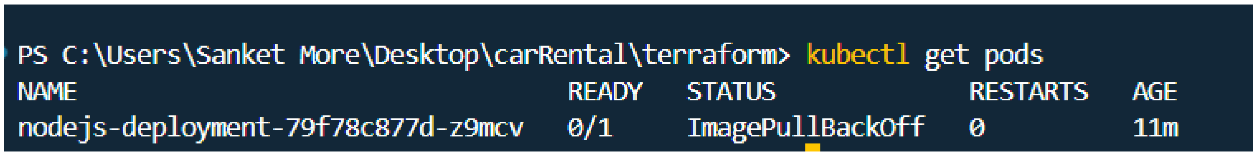


```
Apply complete! Resources: 62 added, 0 changed, 0 destroyed.
PS C:\Users\Sanket More\Desktop\terraform2> aws eks update -kubeconfig --region us east-1 --name sanket3004-cluster
aws : The term 'aws' is not recognized as the name of a cmdlet, function, script file, or operable program. Check the spelling of the name, or if a path was
included, verify that the path is correct and try again.
At line:1 char:1
+ aws eks update -kubeconfig --region us east-1 --name sanket3004-clust ...
+ ~~~
    + CategoryInfo          : ObjectNotFound: (aws:String) [], CommandNotFoundException
    + FullyQualifiedErrorId : CommandNotFoundException
```



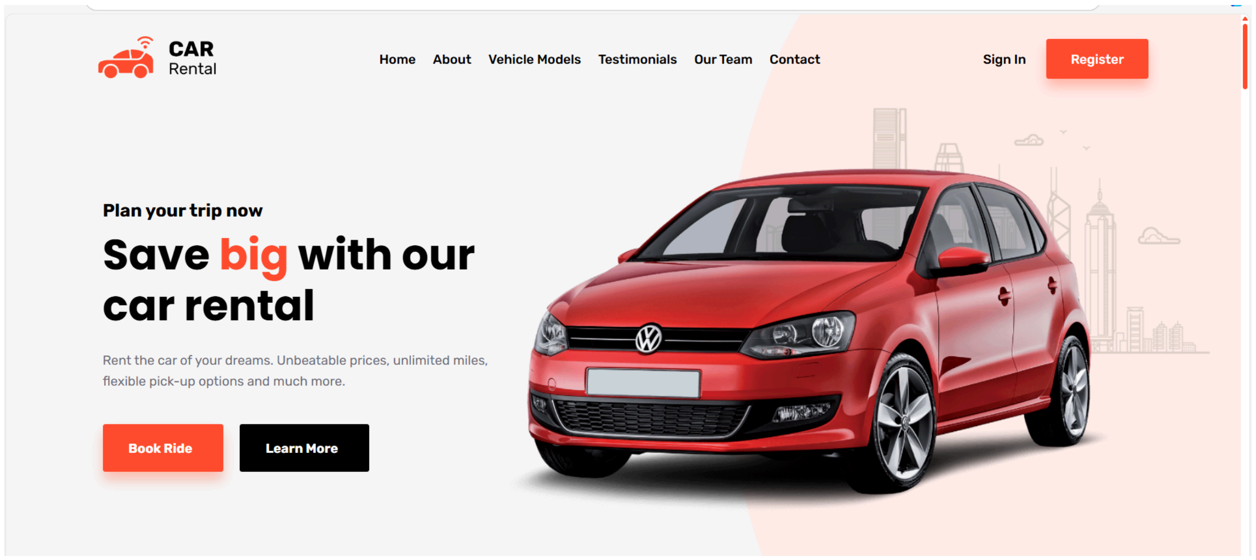
```
PS C:\Users\Sanket More\Desktop\terraform2> kubectl get pods
E1021 15:42:56.302425    3624 memcache.go:265] couldn't get current server API group list: <html><head><meta http-equiv='refresh' content='1;url=/login?from=%2Fapi%3F
timeout%3D32s'/><script id='redirect' data-redirect-url='/login?from=%2Fapi%3Ftimeout%3D32s' src='/static/54a7eafb/scripts/redirect.js'></script></head><body style='b
ackground-color:white; color:white;'>
Authentication required
<!--
-->

</body></html>
E1021 15:42:56.318352    3624 memcache.go:265] couldn't get current server API group list: <html><head><meta http-equiv='refresh' content='1;url=/login?from=%2Fapi%3F
timeout%3D32s'/><script id='redirect' data-redirect-url='/login?from=%2Fapi%3Ftimeout%3D32s' src='/static/54a7eafb/scripts/redirect.js'></script></head><body style='b
ackground-color:white; color:white;'>
Authentication required
<!--
-->
```

So,used minikube locally to host the cluster



```
PS C:\Users\Sanket More\Desktop\carRental\terraform> kubectl get pods
NAME                                 READY   STATUS             RESTARTS   AGE
nodejs-deployment-79f78c877d-z9mcv   0/1     ImagePullBackOff   0          11m
```

Running Successfully.



Plan your trip now

Save big with our car rental

Rent the car of your dreams. Unbeatable prices, unlimited miles, flexible pick-up options and much more.