

Name:Aarya Gandhi

Roll no 15

Class D15B

Advance Devops Practical 6

Aim:To Build, change, and destroy AWS /GCP /Microsoft Azure/ DigitalOcean infrastructure Using Terraform.(S3 bucket or Docker) fdp

Theory:

Terraform is a powerful tool designed to automate the management of cloud infrastructure across providers like AWS, GCP, Microsoft Azure, and DigitalOcean. The practical aim of using Terraform is to efficiently build, modify, and destroy cloud resources—such as AWS S3 buckets or Docker containers—using code instead of manual operations. This is made possible through **Infrastructure as Code (IaC)**, where infrastructure is defined in configuration files that are both human-readable and version-controlled. This approach ensures consistency, reduces manual errors, and makes the process scalable.

Terraform operates by interacting with cloud providers through plugins, enabling you to manage specific resources for each platform. The workflow includes initialization to set up providers, followed by generating a plan that outlines the changes required to align the infrastructure with the desired state. When the configuration is applied, Terraform automatically provisions, modifies, or destroys resources based on the plan, maintaining efficiency and minimizing potential downtime.

A key aspect of Terraform is its state management, which tracks the infrastructure in a state file, ensuring future changes are applied incrementally. This automation not only saves time but also ensures that infrastructure changes are predictable and repeatable, making it an ideal solution for cloud infrastructure management across multiple platforms.

Name:Aarya Gandhi

Roll no 15

Class D15B

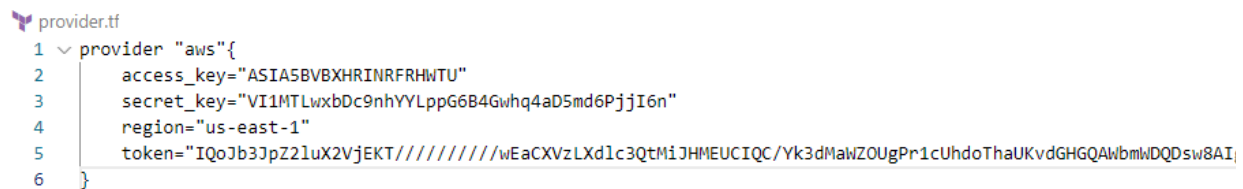
Steps:

Step 1: Write a Terraform Script in Atom for creating S3 Bucket on Amazon AWS

The screenshot shows the Atom text editor with two tabs: 's3.tf' and 'provider.tf'. The 's3.tf' tab is active, displaying a Terraform configuration for an AWS S3 bucket. The code is as follows:

```
1 resource "aws_s3_bucket" "Aarya" {  
2     bucket="my_terraform-bucket-test"  
3     acl="public-read"  
4  
5     tags={  
6         Name="Aarya Bucket"  
7         Environment="Dev"  
8     }  
9 }
```

Create a new provider.tf file and write the following contents into it.

The screenshot shows the Atom text editor with the 'provider.tf' tab active. It contains the AWS provider configuration with the following details:

```
1 provider "aws" {  
2     access_key="ASIA5BV8XHRINRFRHWTU"  
3     secret_key="VI1MTLwxbDc9nhYYLppG6B4Gwhq4aD5md6PjjI6n"  
4     region="us-east-1"  
5     token="IQoJb3JpZ21uX2VjEKT////////wEaCXVzLXd1c3QtMiJHMEUCIQC/Yk3dMawZOUgPr1cUhdoThaUKvdGHGQAWbmWDQDsw8AI"  
6 }
```

Save both the files in same directory Terraform_Scripts/S3

Step 2: Open Command Prompt and go to Terraform_Script\S3 directory where our .tf files are stored

Name:Aarya Gandhi

Roll no 15

Class D15B

```
version - An alias for the version subcommand.
PS C:\Users\admin> cd C:\Terraform_Scripts\S3
PS C:\Terraform_Scripts\S3> dir

Directory: C:\Terraform_Scripts\S3

Mode                LastWriteTime         Length Name
----                -
-a----            8/13/2024   9:32 AM             141 provider.tf
-a----            8/13/2024   9:29 AM             176 s3.tf

PS C:\Terraform_Scripts\S3>
```

Step 3: Execute Terraform Init command to initialize the resources

```
PS C:\Terraform_Scripts\S3> terraform init
Initializing the backend...
Initializing provider plugins...
- Finding latest version of hashicorp/aws...
- Installing hashicorp/aws v5.62.0...
- Installed hashicorp/aws v5.62.0 (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.
```

Name:Aarya Gandhi

Roll no 15

Class D15B

Step 4: Execute Terraform plan to see the available resources

```
PS C:\Terraform_Scripts\S3> 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.Aarya will be created
+ resource "aws_s3_bucket" "Aarya" {
+   acceleration_status      = (known after apply)
+   acl                     = "public-read"
+   arn                    = (known after apply)
+   bucket                 = "my terraform-bucket-test"
+   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                   = {
+     "Environment" = "Dev"
+     "Name"        = "Aarya Bucket"
+   }
+   tags_all              = {
+     "Environment" = "Dev"
+     "Name"        = "Aarya Bucket"
+   }
+   website_domain        = (known after apply)
+   website_endpoint      = (known after apply)

+ cors_rule (known after apply)

+ grant (known after apply)

+ lifecycle_rule (known after apply)

+ logging (known after apply)

+ object_lock_configuration (known after apply)

+ replication_configuration (known after apply)

+ server_side_encryption_configuration (known after apply)

+ versioning (known after apply)
```

Step 5: Execute Terraform apply to apply the configuration, which will automatically create an S3 bucket based on our configuration.

Name:Aarya Gandhi

Roll no 15

Class D15B

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

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

aws_s3_bucket.aarya: Creating...
aws_s3_bucket.aarya: Creation complete after 5s [id=crossgobucket]

Apply complete! Resources: 1 added, 0 changed, 0 destroyed.
PS C:\Terraform_Scripts\S3>
```

AWS S3 Bucket dashboard, After Executing Apply step:

The screenshot shows the AWS S3 console interface. At the top, there's a navigation bar with the AWS logo, a search bar, and various icons. Below the navigation bar, there's a section for 'Account snapshot' and a 'View Storage Lens dashboard' button. The main content area is titled 'General purpose buckets (2)' and includes a 'Create bucket' button. Below this, there's a search bar and a table listing the buckets. The table has columns for Name, AWS Region, IAM Access Analyzer, and Creation date. Two buckets are listed: 'crossgobucket' and 'elasticbeanstalk-us-east-1-896913587280'.

Name	AWS Region	IAM Access Analyzer	Creation date
crossgobucket	US East (N. Virginia) us-east-1	View analyzer for us-east-1	August 13, 2024, 10:20:44 (UTC+05:30)
elasticbeanstalk-us-east-1-896913587280	US East (N. Virginia) us-east-1	View analyzer for us-east-1	July 30, 2024, 09:24:35 (UTC+05:30)

Step 6: Execute Terraform destroy to delete the configuration, which will automatically delete an EC2 instance

Name:Aarya Gandhi

Roll no 15

Class D15B

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

Do you really want to destroy all resources?
Terraform will destroy all your managed infrastructure, as shown above.
There is no undo. Only 'yes' will be accepted to confirm.

Enter a value: yes

aws_s3_bucket.aarya: Destroying... [id=crossgobucket]
aws_s3_bucket.aarya: Destruction complete after 1s

Destroy complete! Resources: 1 destroyed.
```

AWS EC2 dashboard, After Executing Destroy step:

Amazon S3

► **Account snapshot** - updated every 24 hours All AWS Regions View Storage Lens dashboard

Storage lens provides visibility into storage usage and activity trends. [Learn more](#)

General purpose buckets | Directory buckets

General purpose buckets (1) Info All AWS Regions Refresh Copy ARN Empty Delete Create bucket

Buckets are containers for data stored in S3.

	Name	AWS Region	IAM Access Analyzer	Creation date
<input type="radio"/>	elasticbeanstalk-us-east-1-896913587280	US East (N. Virginia) us-east-1	View analyzer for us-east-1	July 30, 2024, 09:24:35 (UTC+05:30)

Conclusion:

Name:Aarya Gandhi

Roll no 15

Class D15B

Using Terraform to build, change, and destroy infrastructure across AWS, GCP, Microsoft Azure, and DigitalOcean showcases the power and flexibility of Infrastructure as Code (IaC). Terraform's ability to automate infrastructure provisioning, manage state, and support multiple cloud platforms streamlines operations, reduces human error, and enhances scalability. By employing Terraform, organizations can efficiently manage their cloud environments, enforce best practices, and achieve consistent infrastructure management across diverse cloud providers, ultimately accelerating their cloud adoption and operational agility.