**TERAFORM (IAC)**

**AWS CFT (CLOUD FORMATION TEMPLETE)**

1. **Why terraform?**

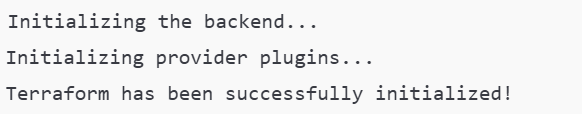
* **Multi-Cloud Support**
* **Large Ecosystem**

1. **What is terraform?**

Terraform is like a magic blueprint for infrastructure. It uses code to automate, manage, and scale resources across any cloud provider. It’s efficient, consistent, and perfect for DevOps engineers who value automation and reliability.

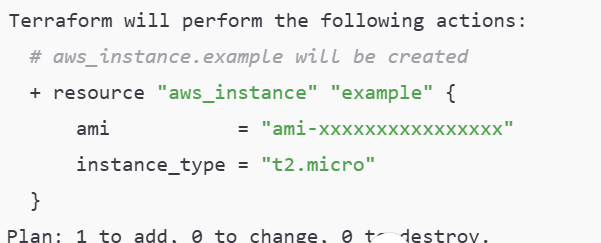
1. **Terraform init?**

* Terraform sets up the directory you’re working in.
* It downloads necessary provider plugins (e.g., AWS, Azure) to talk to your cloud.
* It creates a. terraform folder to store these files.



1. **Terraform plan ?**

* Terraform reads **your .tf configuration files**.
* It connects to your cloud provider to verify the resources you want to create or update.
* It generates an execution plan and shows you what will happen if you apply it.

****

1. **Terraform apply ?**

* Terraform takes the plan you reviewed and starts creating the resources in your cloud environment.
* For example, it provisions EC2 instances, S3 buckets, or VPCs as defined in your .tf files.

1. **Terraform State File ?**

The **Terraform state file** is a critical part of how Terraform operates. It acts as a record of the infrastructure resources Terraform manages.

1. **Terraform plugin?**

A **Terraform provider** is a plugin that allows Terraform to interact with APIs of cloud platforms, services, or other systems (e.g., AWS, Azure, Kubernetes).

1. **Alias**

In Terraform, an **alias** allows you to use the same provider with multiple configurations.

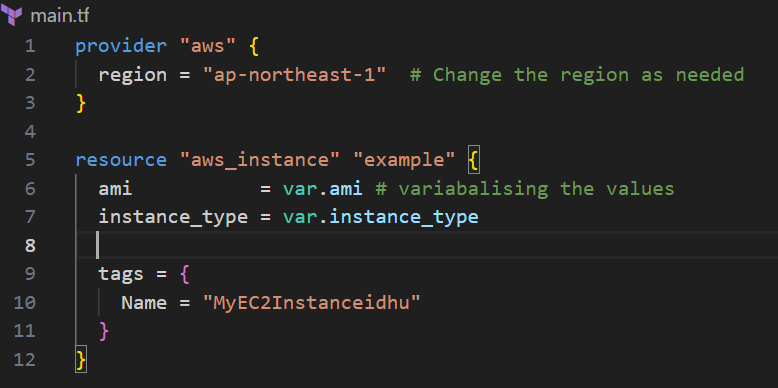
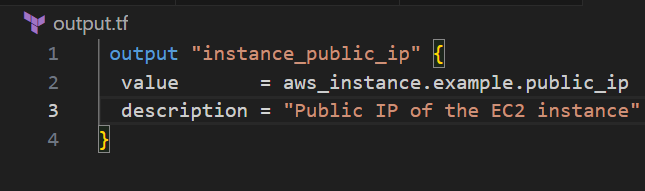


1. **Variables**

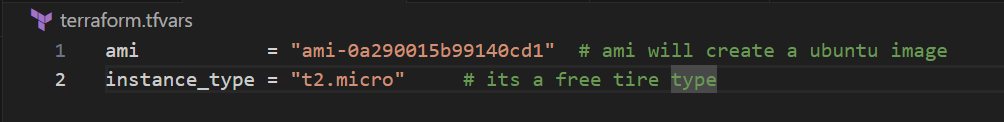
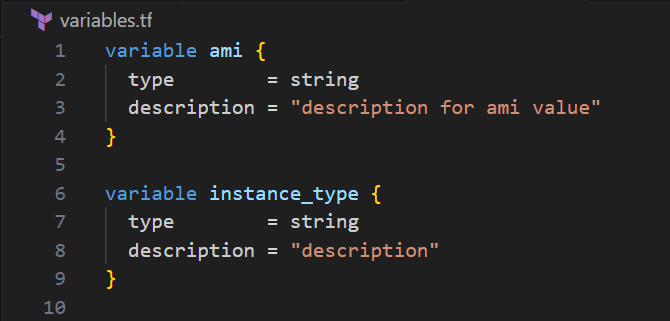
Terraform **variables** are placeholders used to make your configuration dynamic, reusable, and easy to manage. Instead of hardcoding values directly into your Terraform files, you define variables and assign values to them.

**Snapshots examples**

**Main.tf Outputs.tf**

** **

**Terraform.tfvars Variables.tf**

** **

1. **Tf.vars**

**Providing values in .tfvars to variabelize from main.tf**

1. **Conditional expression**

Terraform supports conditional expressions using the condition? true value : false value syntax.



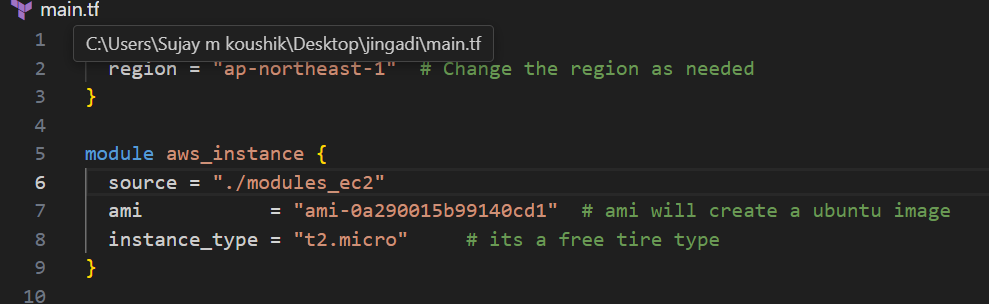
1. **Built-in function**

Terraform provides built-in functions for string manipulation, numeric operations, collections, etc. Some common functions include:

1. **What is terraform Modules?**

In the terraform Micro services called as modules

**For modules we just need to write main .tf and give src file location and values to create**



1. **State file (terraform show) display all state file info**

* The Terraform state file is **heart of Terraform**.
* It keeps track of the real-world infrastructure created or managed by Terraform and maps it to the configuration defined in your .tf files.
* It store the information in state file which its tracked, because **it will updated the available infra**

**Draw backs**

It will record everything and store in state file.

**include password and sensitive information so comes remote backend**

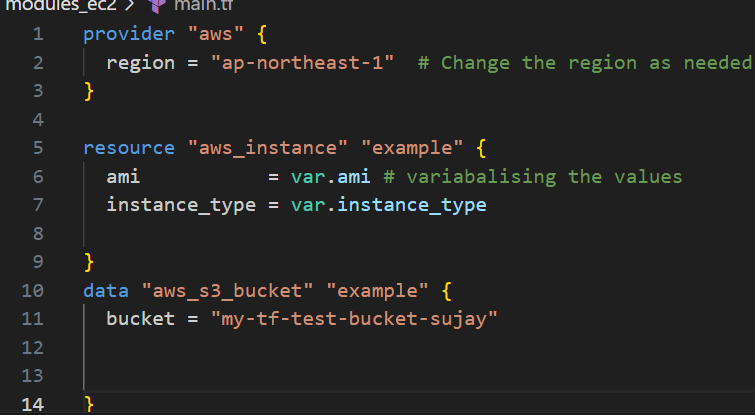
1. **Remote backend**

It is used to host the statefile example S3 Buckect

We have GitHub repository ,to host terraform code instead of storing state file in GitHub we are use remote backend logic and use S3 Bucket

If any DevOps team want to update any logic they simply clone repo make and verify raise pull request back it will update in s3 bucket

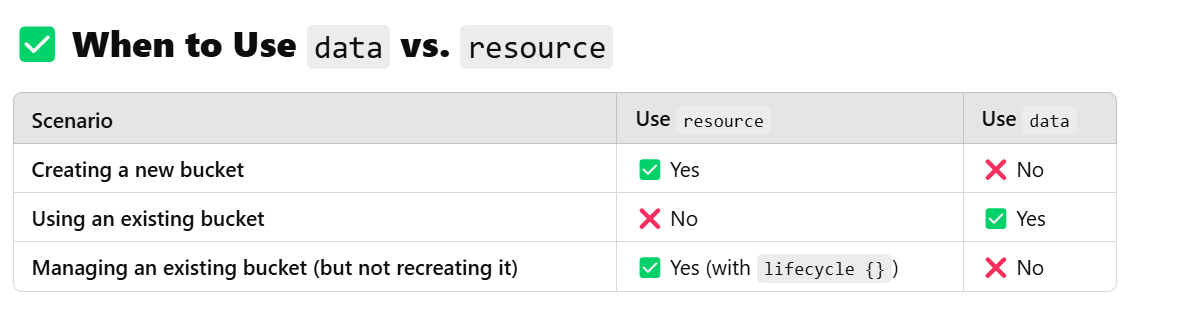
**Main.tf**



**Backend.tf**



**When you have an existing AWS resource (like an S3 bucket), you should use the data block instead of the resource block.**

****

**it is not possible to create an S3 bucket and use it as a Terraform backend in the same run.**

1. **Locking mechanism of terraform ?**

Sujay and Vijay, two DevOps engineers, were working on different Terraform changes. Sujay initiated terraform apply, which locked the state in DynamoDB. Meanwhile, Vijay unknowingly tried to apply his changes but was blocked due to the lock. Suddenly, Sujay’s system crashed, leaving the lock stuck. After confirming Sujay’s process was no longer running, Vijay used terraform force-unlock <LOCK\_ID> to release the lock and proceed. This taught them the importance of checking locks, coordinating deployments, and leveraging Terraform Cloud for smoother collaboration.



1. **Terraform Provisioner?**

**By using provisioner we can deploy the application**

**Inbound is ingress**

**Outbound is egress**

**Types of provisioner**

**Remote exec provisioner**

**During the creation of infrastructure, it will run all the commands**

**Example**

**apt install pthon 3**

**apt update**

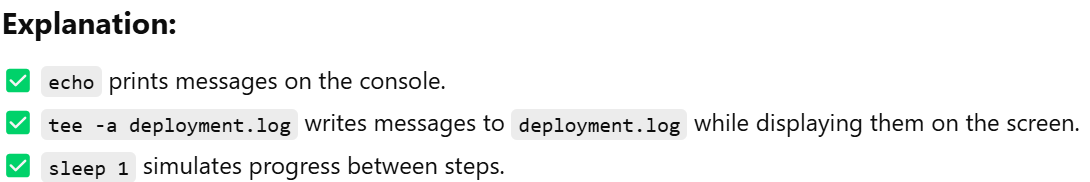
**Local exec**

**To print anything on console during creation using echo**

**Copy all output to the particular file printing that**

**1/8**

**2/8**

****

**8/8 sucessfully created all resources**

**File provisioner**

**Using this we can copy files from local to remote**

**Example copy app.py from local to ec2 instance that created**

1. **Terraform Workspace?**

A Terraform workspace is an isolated environment that allows you to manage multiple instances of the same Terraform configuration without duplicating code.

* terraform workspace new dev
* terraform workspace list
* terraform workspace select dev

