**1.TERRAFORM LIFECYCLE:**

* **CREATE BEFORE DESTROY:**
  + Firstly create instance & then destroy.(e.g) created an instance(server 1) with t2.micro, suppose if I want to change that instance(server 1) with t2.medium ,
  + create a new instance(server 2) with t3.medium & wait for new instance to run healthy then terraform will destroy old instance (server 1).
* **PREVENT DESTROY:**
* Used to prevent critical resources & accidental deletion
* Once prevent\_destroy is set to “true”, if someone tries to destroy or delete terraform will raise an error. So thst instance will be protected.
* **IGNORE CHANGES:**

**IMPORTANCE OF STATE FILE:**

* Create an instance with tag, suppose if we want to change tag with different name, terraform wont change instead it will create a new instance with new tag.

(e.g) creating an instance with tag(ARUN) & with VPC, that info will be stored in state file, if we change the tag name with (KUMAR), that will be stored as (KUMAR) in state file but in terraform(main.tf) with IGNORE CHANGES set up it will prevent from creating the same instance with new tag(KUMAR), instead it will create separate instance with tag KUMAR .

**DEPENDS\_ON BLOCK:**

* Used to specify dependencies between resources.

(e.g) before creating an instance we will create VPC, security groups and then we will create instance with given vpc, security groups etc.,

**VPC 🡪 EC2 🡪 S3 🡪 IAM ROLES**

**TERRAFORM DATA SOURCE BLOCK:**

* If an instance is running for long period its AMI ID will be updated automatically by aws.

In this terraform data source block by using ‘**aws\_ami’ it will** fetch and update ami id automatically.

**TERRAFORM IMPORT:**

* An ordinary instance without terraform will be changed to terraform state by using this terraform import.

**LOCK\_ID:**

* Terraform uses a state file to keep track of the managed resources. When multiple users are working in a configuration, in order to avoid conflicts & corruption this LOCK\_ID is used.

This will generate a lock id which will not allow other users while running in a configuration.

**WORKSPACE:**

* In terraform, this terraform will allow to manage multiple instance with different configuration.

(e.g) creating 3 instances (development, test, production) with one terraform script but with we need 3 different states, in order to create separate state file terraform workspace will help.

Here are basic commands of terraform workspace

1. **CREATE NEW WORKSPACE:**

# terraform workspace new <workspace\_name>

1. **SELECT A WORKSPACE -** switches to the specified workspace

# terraform workspace select <workspace\_name>

1. **LIST WORKSPACE -** lists all the available workspaces

# terraform workspace list

1. **SHOW CURRENT WORKSPACE** - shows the name of the currently selected workspace.

# terraform workspace show