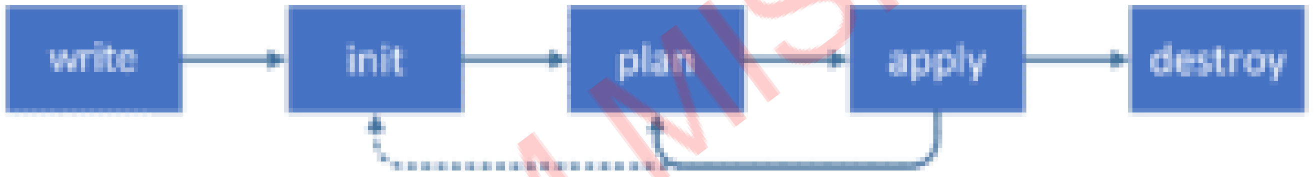
Terraform Basic Command

Terraform Workflow

It is extremely helpful in a team and can benefit you even if you work individually. A good workflow enables you to streamline a process, organize it, and make it less error-prone.



The workflows of Terraform are built on top of five key steps: Write, Init, Plan, Apply, and Destroy. Nevertheless, their details and actions vary between workflows.

Write – this is where you create changes to the code.

Init – command will initialize the working directory containing Terraform configuration files and install any required plugins.

Plan – this is where you review changes and choose whether to simply accept them.

Apply – this is where you accept changes and apply them against real infrastructure.

Destroy – this is where to destroy your entire created infrastructure.

Setting up cloud user credentials:

Deploying servers to your cloud ,account requires you to have your username and password safely stored in your environmental variables. Use the commands below to include an account name and password in your profile. Replace the username and password with you rcloud account username and password.

```
# echo 'export CLOUD_USERNAME=username' | tee -a ~/.bashrc
```

```
# echo 'export CLOUD_PASSWORD=password' | tee -a ~/.bashrc
```

Then reload the profile to apply the new additions.

```
# source ~/.bashrc
```

Terraform Plugins: These are executable binaries written in Go that communicate with Terraform Core over an RPC interface. Each plugin exposes an implementation for a specific service, such as the AWS provider or the cloud provider. Terraform currently supports one type of Plugin called providers.

Imp Keywords:

Before talking about structure, there are some key Terraform concepts we need to know.

- **Resources:** A resource is a cloud component that is created based on a specified configuration (e.g. virtual networks, compute instances, DNS records, ...).
- **Providers:** Plugins used by Terraform to interact with cloud providers. A configuration must define a provider, so Terraform can install and use them.
- **State File:** Mechanism used by Terraform to keep track of all the resources that are deployed in the cloud.

Creating Our First Project:

A Terraform project is just a set of files in a directory containing resource definitions.

Those files, which by convention end in .tf, use Terraform's configuration language to define the resources we want to create.

Each Terraform project is organised in their own directories. When invoking any command that loads the Terraform configuration, Terraform loads all configuration files within the working directory in alphabetical order. This is important to remember when configuring resources that might be dependent on one another.

Lab Session: Launch Ec2 Instance.

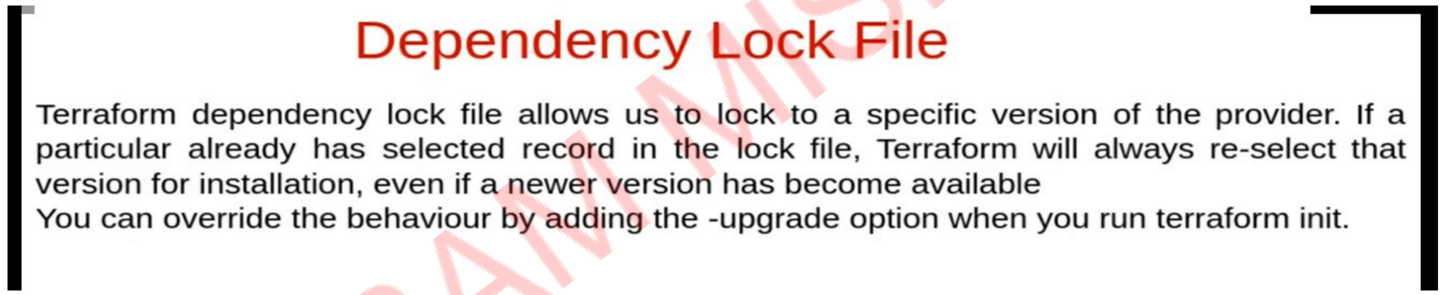
Let's understand File Structure for terraform dir….

These new files include the following:

- The **.terraform** *folder was created during the init process. The terraform directory is a local cache where Terraform retains some imp files it will need for subsequent operations against this configuration.*

- **.terraform.lock.hcl** What's a lock file?

  Terraform 0.14 added support for a lock file which gets created or updated every time you run terraform init. The file is typically generated into your working directory (i.e., the folder in which you ran terraform init) and is called .terraform.lock.hcl. It captures the versions of all the Terraform providers you're using. Normally, you want to check this file into version control so that when your team members run Terraform, they get the exact same provider versions.



Dependency Lock File

Terraform dependency lock file allows us to lock to a specific version of the provider. If a particular already has selected record in the lock file, Terraform will always re-select that version for installation, even if a newer version has become available
You can override the behaviour by adding the -upgrade option when you run terraform init.

- What is tfstate file> *This state file contains information about the provisioned infrastructure which terraform manage. This is created and updated each time an apply or destroy command is run.*

"IT BASICALLY ACT AS A DATABASE FOR PROVISINED RESOURCES".