**TERRAFORM AND DOCKER WITH KUBERNETES**

# **TERRAFORM**

It is an Infrastructure as a code, we can define your desired infrastructure configuration in code

# **Advantages of Terraform**

1. Automated infra-CRUD [CREATE, READ, UPDATE, DELETE]

* Within minutes we can create an infrastructure on AWS server

1. Version Control

* Since it is code, we can keep it in git and track it properly, it is easy to collaborate also with team members, we can maintain the versions of our infra

1. Consistent infra

* Using terraform code, we can create similar infra in all environments without duplicating code

1. Inventory management

* By seeing the terraform code we can understand the resources we are using, we no need to go to console and fetch the report

1. Cost optimization

* It can destroy the infra within minutes so that we can reduce the cost and other problems forgetting resources to delete

1. Automatic dependency resolution

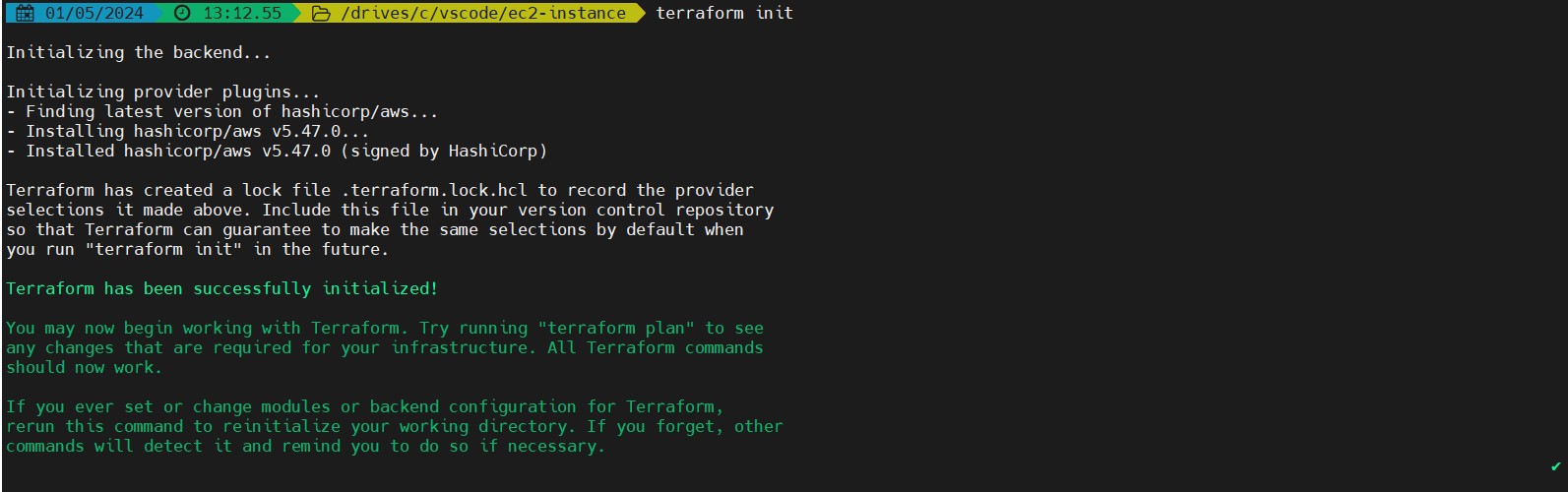
* Terraform will take care of dependency resolution while creating and deleting the resources

1. Modular Infra

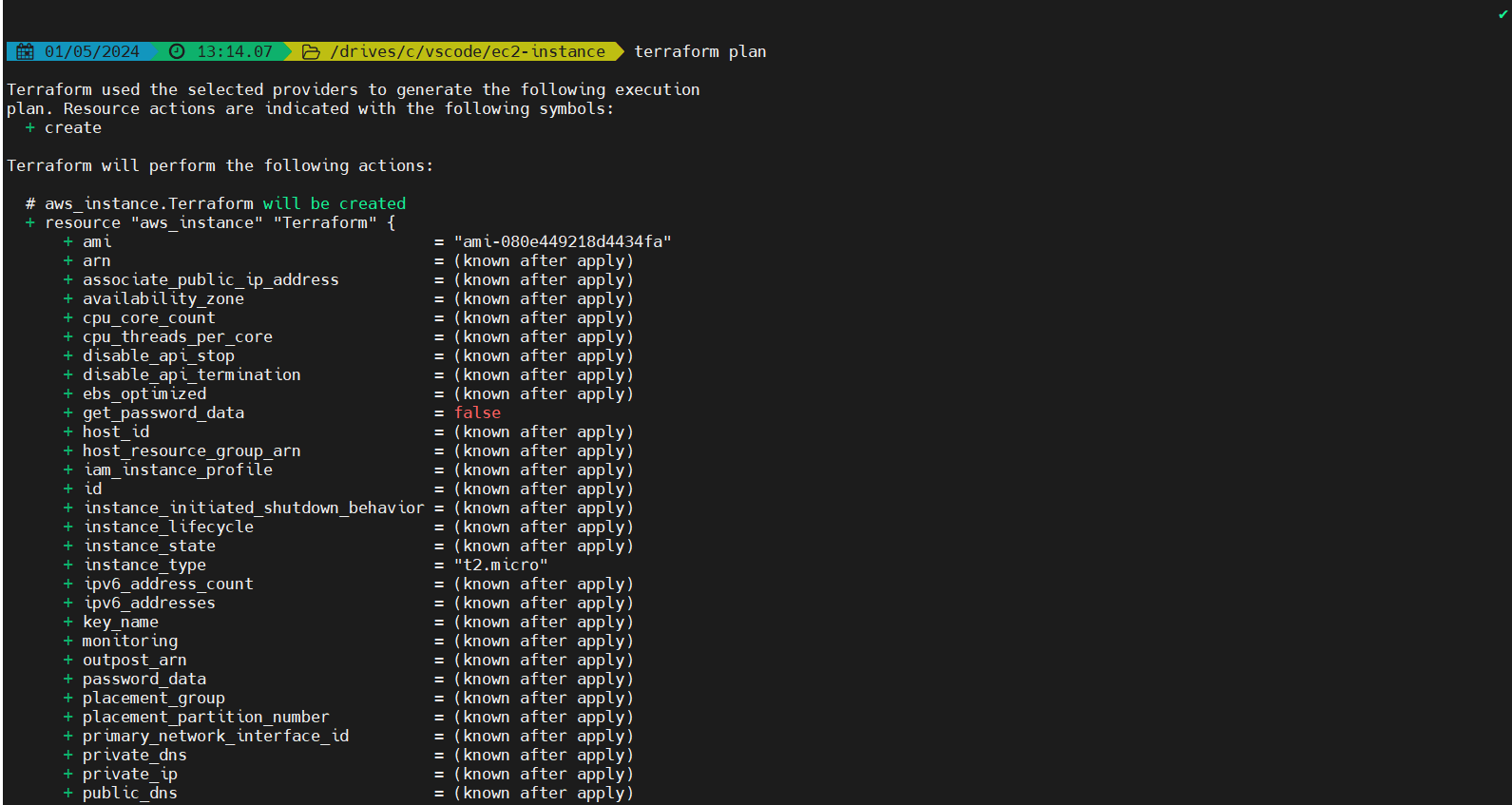
* Code reuse, we can have terraform modules created, any number of projects can call this module to create infra instead of writing from the scratch again

# **TERRAFORM COMMANDS**

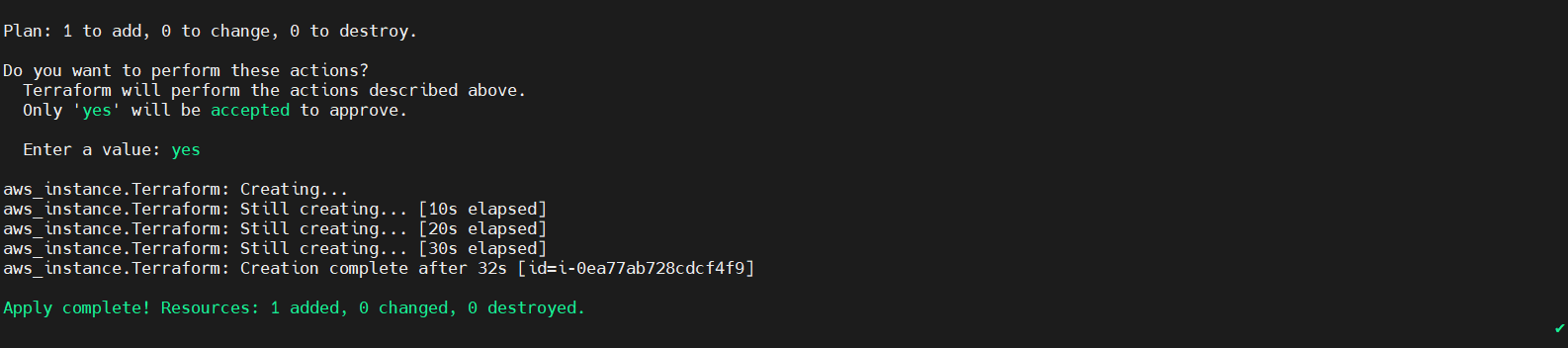
These terraform commands used for creating an ec2 instance in aws using terraform commands and vscode.

**terraform init**: - it will connect to the cloud provider and download the clod provide along with other configurations

**terraform plan: -** It will not create but display what we are going to create in cloud provider

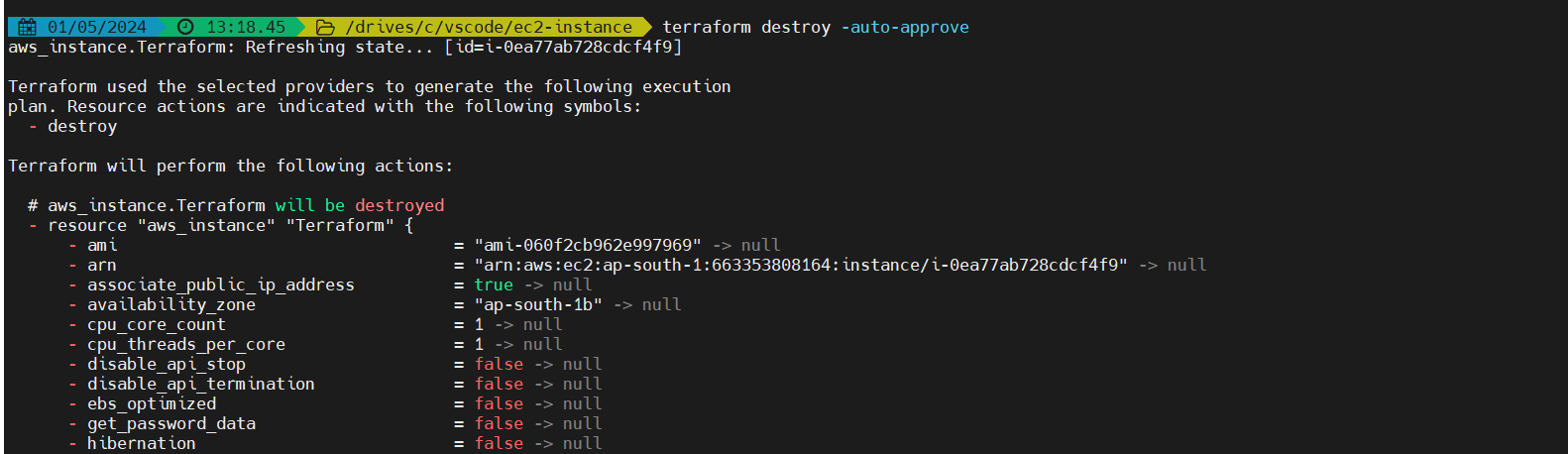


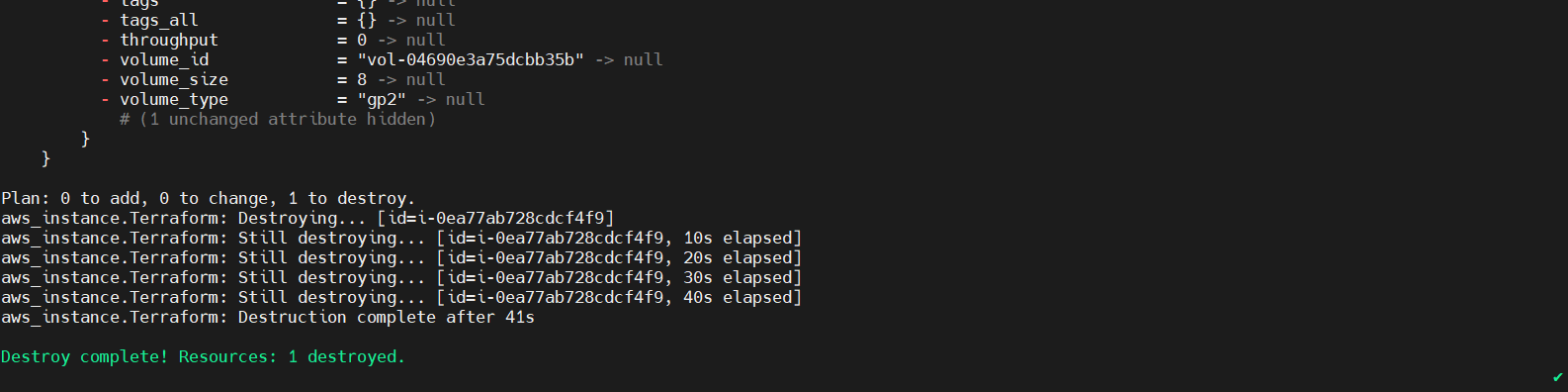
**terraform apply: -** It will create the resources what has displayed using terraform plan



**terraform destroy: -** It will delete the resources what we have created

(command: -terraform destroy -auto-approve)





# **TERRAFORM CONCEPTS**

* Variables
* Count and Count Index
* Locals
* Data Sources
* Functions
* Provisions
* Modules

.terraform folder should not be pushed into the GitHub repository, so use (.gitignore) folder in vscode and write the git commands to ignore the. terraform folder

Terraform files are the declared configuration which is the terraform responsibility to create the resources what we have written in .tf file

Actual configuration is in tfstate. this is where terraform track the resources it was created

**For setting up the network**

1. Creating a VPC
2. Create Internet Gateway
3. Attach Internet Gateway to VPC
4. Create Public subnet
5. Create Public route table
6. Create Private subnet
7. Create Private route table
8. Create Database subnet
9. Create Database route table
10. Associations of all subnets with the route tables
11. NAT Gateway

* used to allow internet to the private subnet
* For this nat gateway we need to get elastic ip first then we need to get nat gateway
* For this allocation id is elastic ip and subnet id is public subnet

1. Attach NAT Gateway

* For this we need to get the aws\_route from the terraform documentation
* In that route table id is private and database alternatively and the destination cidr block is (0.0.0.0/0) internet
* And then add nat gateway id as the name we have given for nat gateway
* All these we can get from the terraform documentation
* Inbound 🡪 Traffic coming in
* Outbound 🡪 Traffic going out, it is completely in our control

# **Variables**

The syntax we are using for variables in terraform is

**For String, Number and Bool: -**

variable ”name-of-variable”{

type = variable-type

default = “value that we are passing”

description = “ ”

}

**For Map: -**

variable “tags”{

type = map(string)

default = {

“Name” = “name-of-variable”

}

}

**For List: -**

Variable “list\_name”{

type = list

default = [“value1”,”value2”,”value3”]

}

In terraform the variables we have are

1. Number
2. String
3. Bool
4. List
5. Map

**Count: -** It is useful for creating the multiple resources by adding **(“count = number”).** but it will create with the same name

**Count Index: -** It is usefulfor creating multiple resources with the specific names by declaring in the variables as list . (count.index)

**Conditions: -**

condition ? true-value : false-value

# **Functions**

<https://developer.hashicorp.com/terraform/language/functions>

file: - file function is used to read the content in a file

**ssh-keygen -t rsa -f name:** used to create rsa keygen

**Merge: -** it is used to change the particular tag from variables. Used as merge (var.tags, {Name = “a”}}

**Loops: -**

There are two types of loops they are

1. **Count based loop: -** It is mostly works with list
2. **For each loop: -** It is mostly used with maps

**Variables: -** we can use variables in 3 types

1. Default values through variables.tf
2. Values through terraform.vars
3. Through command line

We can always override existing default values in variables.tf

* If there are no variables in .tf and .vars files it will ask for variables through command line
* If there are variables in both .tf and .vars files then it will take from .vars file
* If there is variable in any of the files, then terraform will take from that particular file

# **State and Remote State files**

If we keep the state file in local, there is a chance of creating duplicate infra

1. In a collaboration env, to avoid duplicate infra and errors we will use remote state
2. Keeping state file in local may be deleted or changed, restoring it manually is difficult

**Terraform responsibility is to match the declared configuration of files with state file. Terraform keeps the actual configuration in state file. if the state file is in local it will be difficult to collaboration env and there may be a chance of duplicate and errors, so that is the reason we’ll keep the state file in remote location ex: s3 bucket and lock it with DynamoDB so that no duplicates and no errors and also no two persons cannot create infra at a time**

**Outputs: -** Outputs are useful to print the output values of resources

**Locals: -** Locals can run some functions or expressions and it stores the values

**Difference between locals and variables**

In variables file we can declare the variable and values so as in the local file but locals have some capability of running the functions or expressions and storing the values inside the variable whereas variables can only store the values but it will not run the functions and expressions

**Data sources: -** Ifwe want some information from cloud providers, we can query the providers and fetch the data

# **Module Development**

**VPC Module: -**

This module is going to create following resources. It can automatically fetch the first 2 AZ and create the resources.

* VPC
* Internet Gateway
* 2 public subnets
* 1 public route table
* 2 private subnets
* 1 private route table
* 2 database subnets
* 1 database route table
* 1 EIP – Elastic IP
* 1 NAT gateway in 1a az.
* Association between public subnets and public route table
* Association between private subnets and private route table
* Association between database subnets and database route table

**Name format: -**

**for subnets:** [project\_name]-public/private/database-1a/1b  
**for routetables:** [project\_name]-public/private/database

**Arguments: -**

**Project\_name** (Required) - User must provide his project name

**vpc\_cidr** (Optional) - Default value is 10.0.0.0/16

**vpc\_tags** (Optional) - User can provide tags, otherwise empty

**public\_subnet\_cidr** (Required) - User must provide 2 valid subnet CIDR

**public\_subnet\_tags** (Optional) - User can provide tags, otherwise empty

**private\_subnet\_cidr** (Required) - User must provide 2 valid subnet CIDR

**private\_subnet\_tags** (Optional) - User can provide tags, otherwise empty

**database\_subnet\_cidr** (Required) - User must provide 2 valid subnet CIDR

**database\_subnet\_tags** (Optional) - User can provide tags, otherwise empty

**private\_route\_table\_tags** (Optional) - User can provide tags, otherwise empty

**public\_route\_table\_tags** (Optional) - User can provide tags, otherwise empty

**database\_route\_table\_tags** (Optional) - User can provide tags, otherwise empty

Outputs

vpc\_id - This is the ID of VPC created

# **DOCKER**

Docker is one of the software to provide image and container capability

1. There will be no hypervisor, hardware, OS, container engine
2. Container or image is the combination of Base OS + required packages + app server + app code
3. It will not block storage, proper resource utilization and bootable time is very less
4. Cost is very less
5. High availability: - if one server goes down immediately app will be shifted to another server
6. Auto scaling: - based on traffic, app instances will be increased within seconds
7. Reliability: - Trust on applications availability
8. Consistent environment
9. Containers are immutable in nature
10. Portable in nature, Easy to shift between the environments