**TERRAFORM AND DOCKER WITH KUBERNETES**

**DAY-1**

# **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

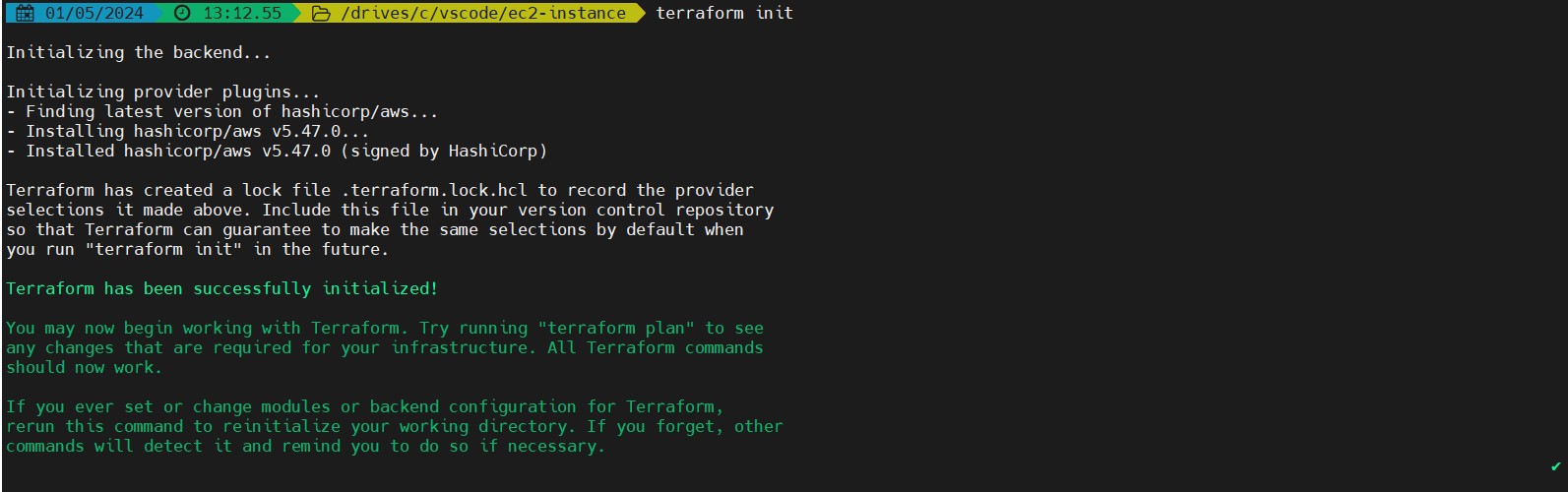
* Terraform will take care of dependency resolution while creating and deleting the resources

1. Modular Infra

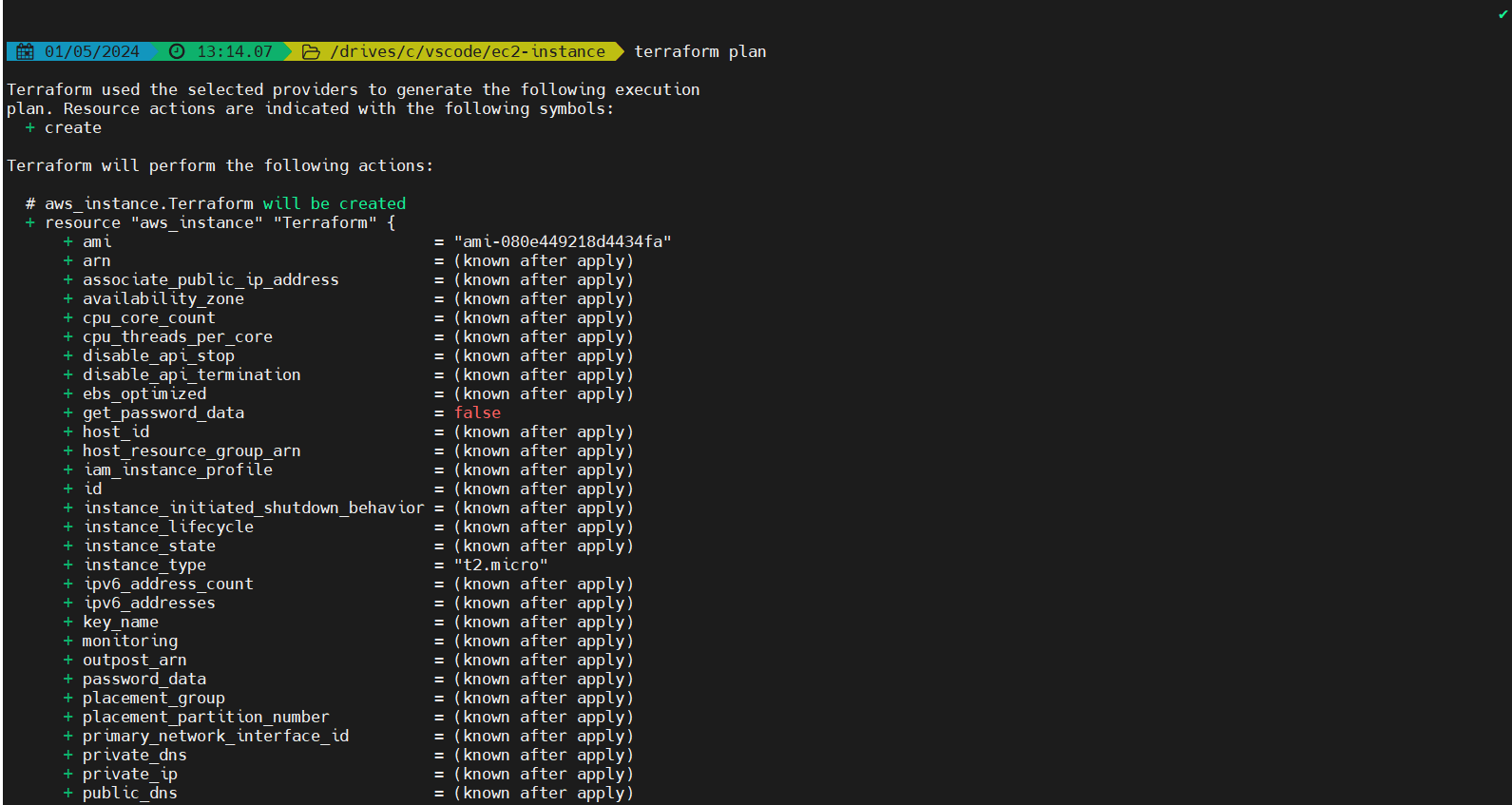
* Code reuse, we can have terraform modules created, any number of projects can call this module to create infra instead of writing from the scratch again

# **TERRAFORM COMMANDS**

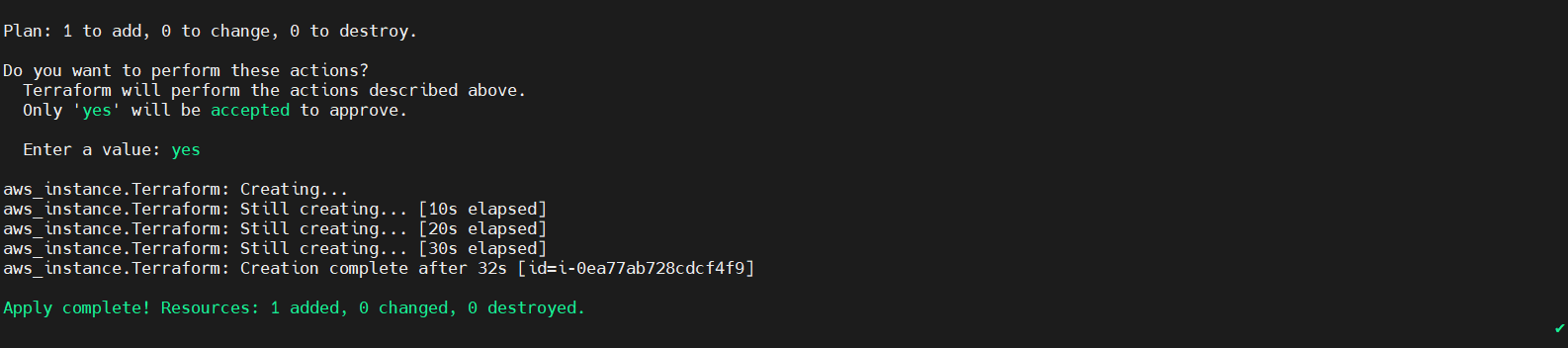
These terraform commands used for creating an ec2 instance in aws using terraform commands and vscode.

**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

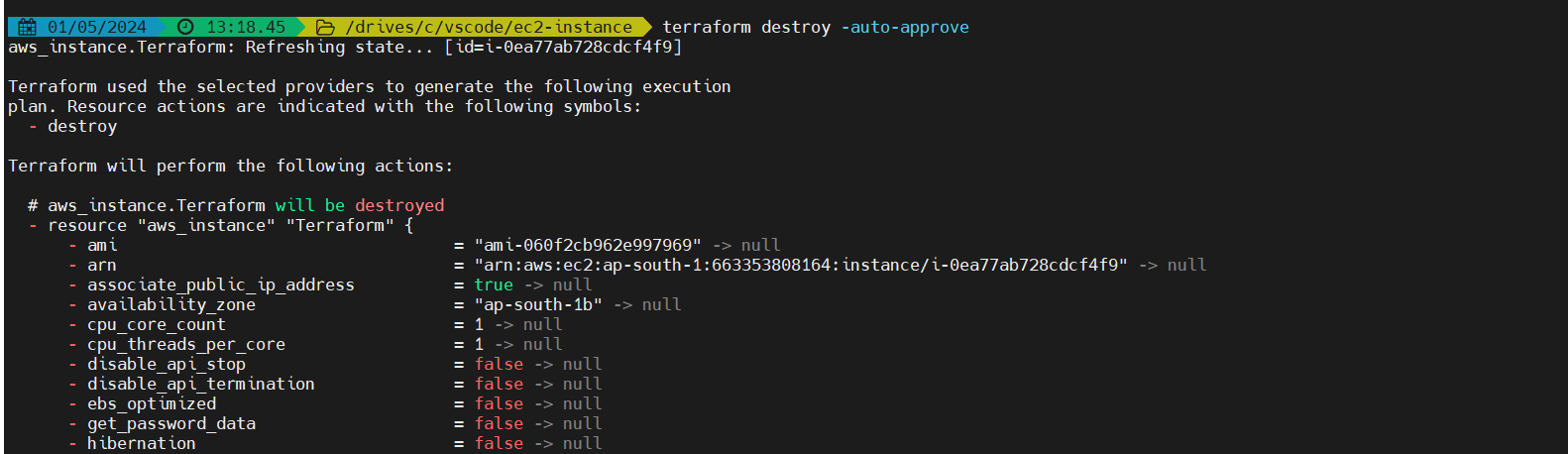


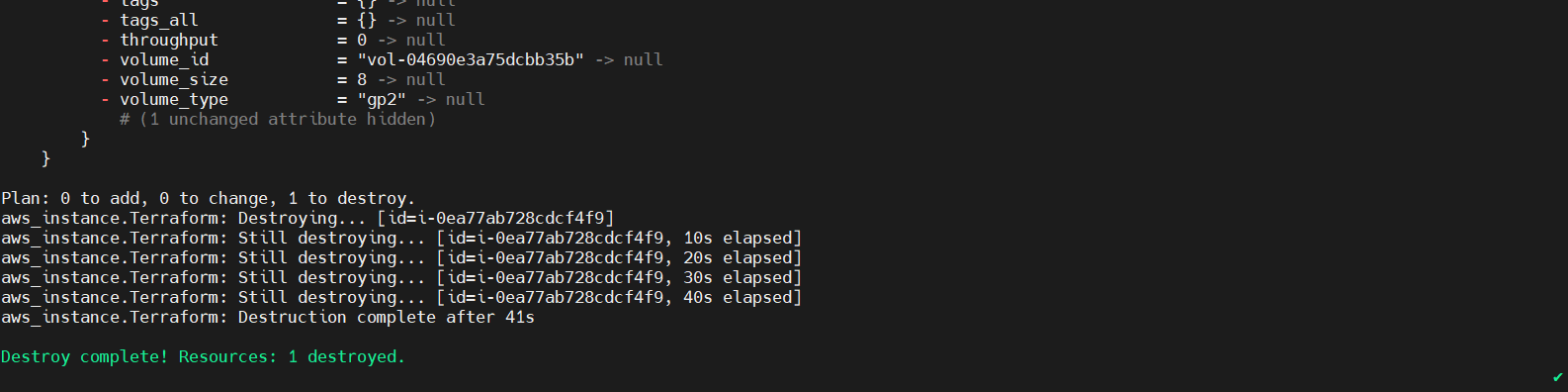
**terraform apply: -** It will create the resources what has displayed using terraform plan



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

(command: -terraform destroy -auto-approve)





**DAY-2**

.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

# **TERRAFORM CONCEPTS**

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

**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