Terraform III

If state file is present, can we create the resource again? No

Terraform project setup with modules:
1. Create a project directory
    a) mkdir 05-terraform-modules-project
2. Create a provider.tf file

```
provider "aws" {
        region = "ca-central-1"
}
```

3. Create a 'modules' directory inside main project directory
4. Inside 'modules' directory we need to create ec2, s3 directories
    a) mkdir 05-terraform-modules-project/modules/ec2
    b) mkdir 05-terraform-modules-project/modules/s3
5. Need to add access key, secret key, to the project
        export AWS_ACCESS_KEY_ID = ""
        export AWS_SECRET_ACCESS_KEY=""

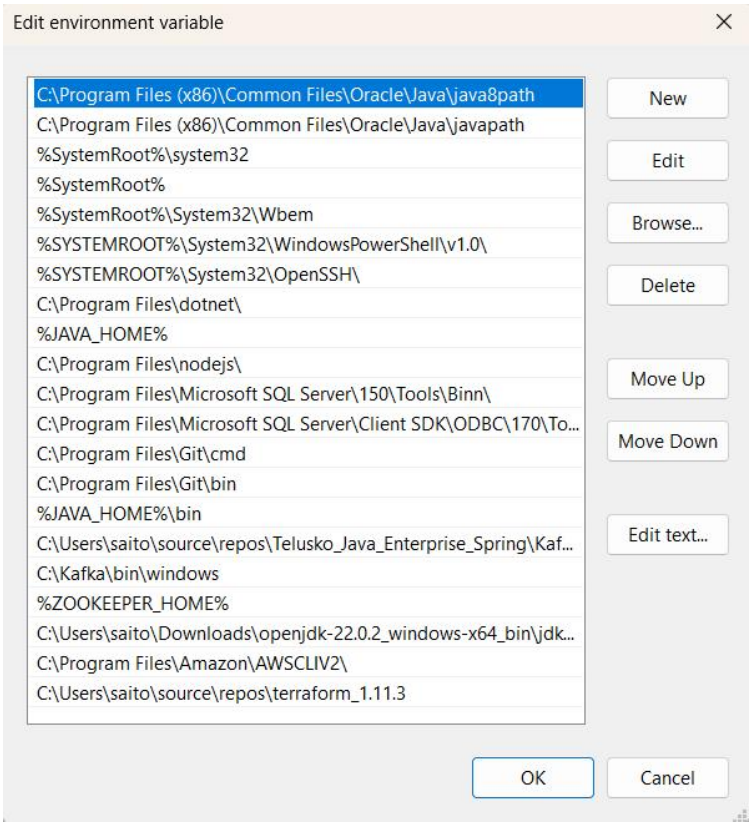
6. Create files in ec2 directory with names: input.tf, main.tf, output.tf
    a) touch input.tf
    b) main.tf
    c) output.tf

7. Create files in s3 directory with names: input.tf, main.tf, output.tf
    a) touch input.tf
    b) main.tf
    c) output.tf
8. Create main.tf, output.tf under main directory
9. Edit ec2 dir ---> main.tf, input.tf, output.tf
10. Edit s3 dir ---> main.tf

Working with Terraform in Windows machine, Lock file, Statefile, Taint and Untaint in Terraform

Working with Terraform in Windows machine

1. Download Terraform for Windows and extract the ZIP file
    a. After extracting we can see terraform.exe file
2. Set path for terraform in System variables --> Environmental variables



3. Configure AWS credentials in system environmental variables for local setup
4. Download and install VSCode

In VScode I add the code in main.tf and write terraform init

```
provider "aws" {
  region = "ca-central-1" # or your preferred AWS region like us-east-1

  # Optional if you have AWS credentials configured via CLI or environme
  # access_key = "YOUR_ACCESS_KEY"
  # secret_key = "YOUR_SECRET_KEY"
}

resource "aws_instance" "linux_vm" {
```

```
ROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS  powershell - 01-terraform-script

o that Terraform can guarantee to make the same selections by default when
ou run "terraform init" in the future.

erraform has been successfully initialized!

ou may now begin working with Terraform. Try running "terraform plan" to see
ny changes that are required for your infrastructure. All Terraform commands
hould now work.

f you ever set or change modules or backend configuration for Terraform,
erun this command to reinitialize your working directory. If you forget, other
ommands will detect it and remind you to do so if necessary.
```

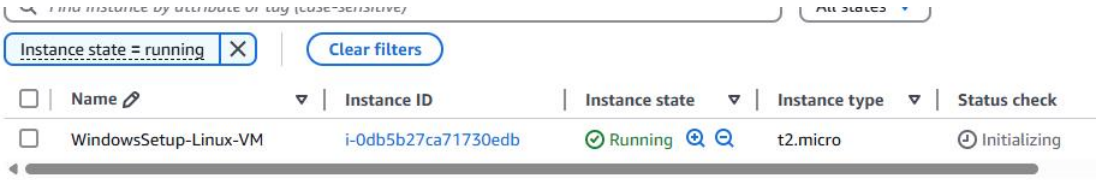
It has created EC2 from VS code:

```
      + private_dns_name_options (known after apply)

      + root_block_device (known after apply)
    }

Plan: 1 to add, 0 to change, 0 to destroy.
aws_instance.linux_vm: Creating...
aws_instance.linux_vm: Still creating... [10s elapsed]
aws_instance.linux_vm: Creation complete after 13s [id=i-0db5b27ca71730edb]

Apply complete! Resources: 1 added, 0 changed, 0 destroyed.
PS C:\Users\saito\source\repos\DevOpsWithAWS_Course\Terraform\Windows_setup\01-ter
```



Destroyed from VSCode

```
14  variable "instance_type" {
15    description = "Represents the type of instance"
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS powershell - 01-terraform-script

```
  Terraform will destroy all your managed infrastructure, as shown above.
  There is no undo. Only 'yes' will be accepted to confirm.

  Enter a value: yes

aws_instance.linux_vm: Destroying... [id=i-0db5b27ca71730edb]
aws_instance.linux_vm: Still destroying... [id=i-0db5b27ca71730edb, 10s elapsed]
aws_instance.linux_vm: Still destroying... [id=i-0db5b27ca71730edb, 20s elapsed]
aws_instance.linux_vm: Still destroying... [id=i-0db5b27ca71730edb, 30s elapsed]
aws_instance.linux_vm: Destruction complete after 31s

Destroy complete! Resources: 1 destroyed.
```

Without applying taint

```
Apply complete! Resources: 1 added, 0 changed, 0 destroyed.
PS C:\Users\saito\source\repos\DevOpsWithAWS_Course\Terraform\Windows_setu
rm apply --auto-approve
aws_instance.linux_vm: Refreshing state... [id=i-0e7bc097223ae0201]

No changes. Your infrastructure matches the configuration.

Terraform has compared your real infrastructure against your configuratior
no changes are needed.

Apply complete! Resources: 0 added, 0 changed, 0 destroyed.
```

If any of the resources is tainted or added as a tainted resource, it will destroy the existing resource and add the new resource, instead of just refreshing state
terraform taint aws_instance.linux_vm

```
Apply complete! Resources: 0 added, 0 changed, 0 destroyed.
PS C:\Users\saito\source\repos\DevOpsWithAWS_Course\Terraform\Windows_setup\01-terraform-script> terrafo
rm taint aws_instance.linux_vm
Resource instance aws_instance.linux_vm has been marked as tainted.
PS C:\Users\saito\source\repos\DevOpsWithAWS_Course\Terraform\Windows_setup\01-terraform-script>
```

After taint when we apply again
It will destroy then create the resource

```
Plan: 1 to add, 0 to change, 1 to destroy.
aws_instance.linux_vm: Destroying... [id=i-0e7bc097223ae0201]
aws_instance.linux_vm: Still destroying... [id=i-0e7bc097223ae0201, 10s elapsed]
aws_instance.linux_vm: Still destroying... [id=i-0e7bc097223ae0201, 20s elapsed]
aws_instance.linux_vm: Still destroying... [id=i-0e7bc097223ae0201, 30s elapsed]
aws_instance.linux_vm: Destruction complete after 31s
aws_instance.linux_vm: Creating...
```

```
Plan: 1 to add, 0 to change, 1 to destroy.
aws_instance.linux_vm: Destroying... [id=i-0e7bc097223ae0201]
aws_instance.linux_vm: Still destroying... [id=i-0e7bc097223ae0201, 10s elapsed]
aws_instance.linux_vm: Still destroying... [id=i-0e7bc097223ae0201, 20s elapsed]
aws_instance.linux_vm: Still destroying... [id=i-0e7bc097223ae0201, 30s elapsed]
aws_instance.linux_vm: Destruction complete after 31s
aws_instance.linux_vm: Creating...
aws_instance.linux_vm: Still creating... [10s elapsed]
aws_instance.linux_vm: Creation complete after 12s [id=i-0e551082f0a51ef23]

Apply complete! Resources: 1 added, 0 changed, 1 destroyed.
PS C:\Users\saito\source\repos\DevOpsWithAWS_Course\Terraform\Windows_setup\01-terrafo
```

The default behavior is once State file is created, we cannot re-apply
terraform taint and terraform untaint are commands used to **force Terraform to recreate or preserve specific resources** during the next apply. They're useful when you want to manually intervene in Terraform's lifecycle management of infrastructure.
Once we taint the resource, it is possible to re-create the new resource

You can "taint" a resource using terraform taint to mark it as needing to be re-created during the next terraform apply. It's like telling Terraform that a resource is "bad" or needs re-deployment
terraform taint aws_instance.linux_vm

Using terraform untaint, you can remove the taint from a resource , so it will not be re-created and will stay as is during the next apply

To get back default behavior, we run untaint

terraform untaint aws_instance.linux_vm

Before untaint, taint once again then untaint
PS C:\Users\saito\source\repos\DevOpsWithAWS_Course\Terraform\Windows_setup\01-terraform-script> terraform taint aws_instance.linux_vm
Resource instance aws_instance.linux_vm has been marked as tainted.
PS C:\Users\saito\source\repos\DevOpsWithAWS_Course\Terraform\Windows_setup\01-terraform-script> terraform untaint aws_instance.linux_vm

Resource instance aws_instance.linux_vm has been successfully untainted.
PS C:\Users\saito\source\repos\DevOpsWithAWS_Course\Terraform\Windows_setup\01-terraform-script>

```
PS C:\Users\saito\source\repos\DevOpsWithAWS_Course\Terraform\Windows_setup\01-terraform-script> terrafo
rm taint aws_instance.linux_vm
Resource instance aws_instance.linux_vm has been marked as tainted.
PS C:\Users\saito\source\repos\DevOpsWithAWS_Course\Terraform\Windows_setup\01-terraform-script> terrafo
rm untaint aws_instance.linux_vm
Resource instance aws_instance.linux_vm has been successfully untainted.
PS C:\Users\saito\source\repos\DevOpsWithAWS_Course\Terraform\Windows_setup\01-terraform-script>
```

What's a State file in Terraform?

The state file (terraform.tfstate) is where Terrform stores the current state of your infrastructure. It keeps track of all the resources that Terraform manages, so it knows what to create, update, or delete. This file is essential for Terraform to understand what is already deployed.

What's Lock file in Terraform?
Lock file is a set of files. For example, here version="~>5.0" is locked
If different team members within the same team are using different versions that might create a problem. We lock the version and it will maintain the consistency. All pipelines will have the same versions. Lock the versions of providers to maintain consistency across all environments

Lock file in Terraform:
A lock file (.terraform.lock.hcl) is used to lock the versions of provider plugins that Terraform uses. This ensures that your team or CI/CD pipelines are using the same versions of providers across all environments. It's automatically generated by Terraform to avoid unexpected changes due to provider updates.

```
terraform {
 required_providers {
   aws = {
     source  = "hashicorp/aws"
     version = "~> 5.0"
   }
 }
}

# Configure the AWS Provider
provider "aws" {
 region = "us-east-1"
}

# Create a VPC
resource "aws_vpc" "example" {
 cidr_block = "10.0.0.0/16"
}
```

We don't have to create the lockfile automatically in Terraform. The moment we initialize terraform, lockfile will automatically be created.