Terraform Session 7

Summary 19/05/2024

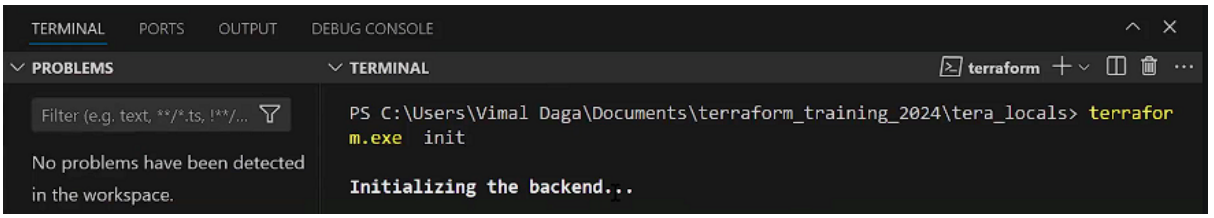
- Go to the folder that you created in your last class and for learning create a new folder in that. And for coding VS code is used.

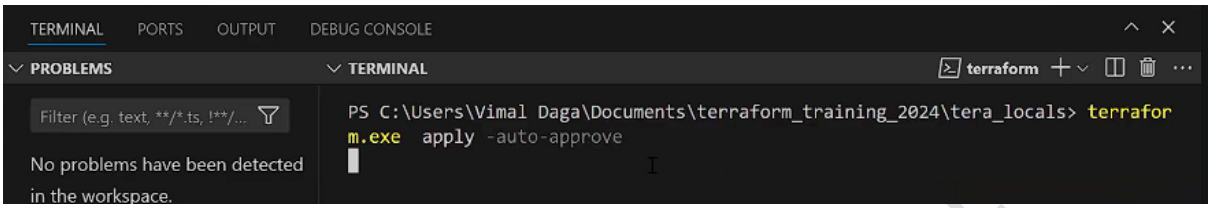
```
MINGW64:/c/Users/Vimal Daga/Documents/terraform_training_2024/tera_locals
Vimal Daga@DESKTOP-3E1AGGT MINGW64 ~
$ cd Documents/

Vimal Daga@DESKTOP-3E1AGGT MINGW64 ~/Documents
$ cd terraform
TerraForm_Training/      terraform-training-2022/ terraform-ws/
terraform-course/        terraform-training-ws/   terraform_training_2024/

Vimal Daga@DESKTOP-3E1AGGT MINGW64 ~/Documents
$ cd terraform_training_2024/

Vimal Daga@DESKTOP-3E1AGGT MINGW64 ~/Documents/terraform_training_2024
$ mkdir tera_locals

Vimal Daga@DESKTOP-3E1AGGT MINGW64 ~/Documents/terraform_training_2024
$ cd tera_locals/

Vimal Daga@DESKTOP-3E1AGGT MINGW64 ~/Documents/terraform_training_2024/tera_loca
ls
$ ls

Vimal Daga@DESKTOP-3E1AGGT MINGW64 ~/Documents/terraform_training_2024/tera_loca
ls
$ code .
```

- In your main file, we have to launch some instances

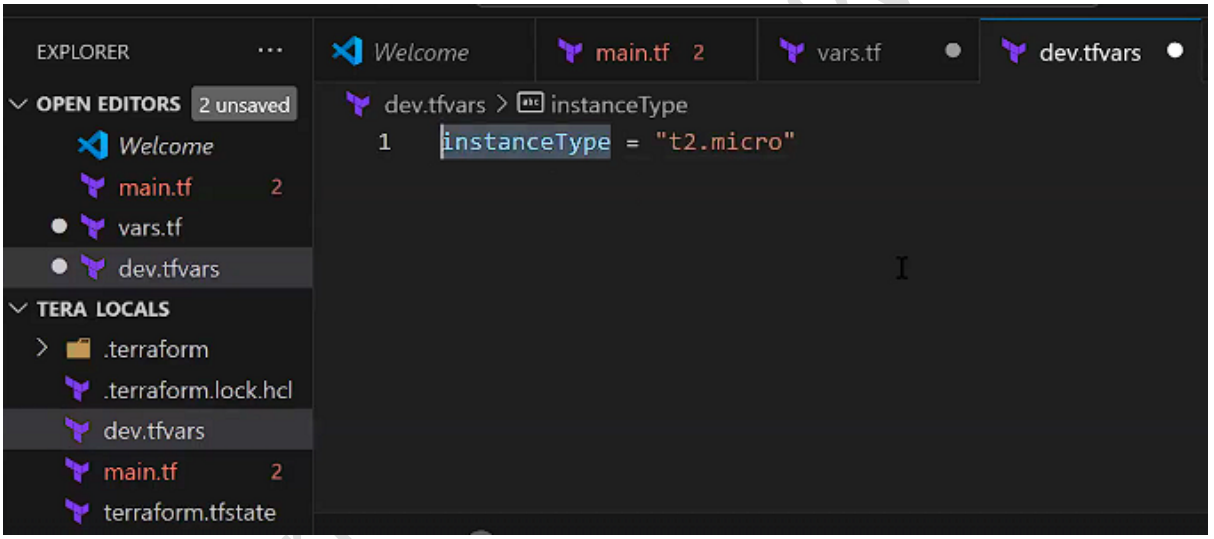
```
main.tf > resource "aws_instance" "name" > tags > Team
1   provider "aws" {
2     region = "us-east-1"
3   }
4
5   resource "aws_instance" "name" {
6     instance_type = "t2.micro"
7     ami = "ami-0bb84b8ffd87024d8"
8
9     tags = {
10      Name = "dev-media-os"
11      Team = "Dev"
```

- Now quickly initialize and launch this code.

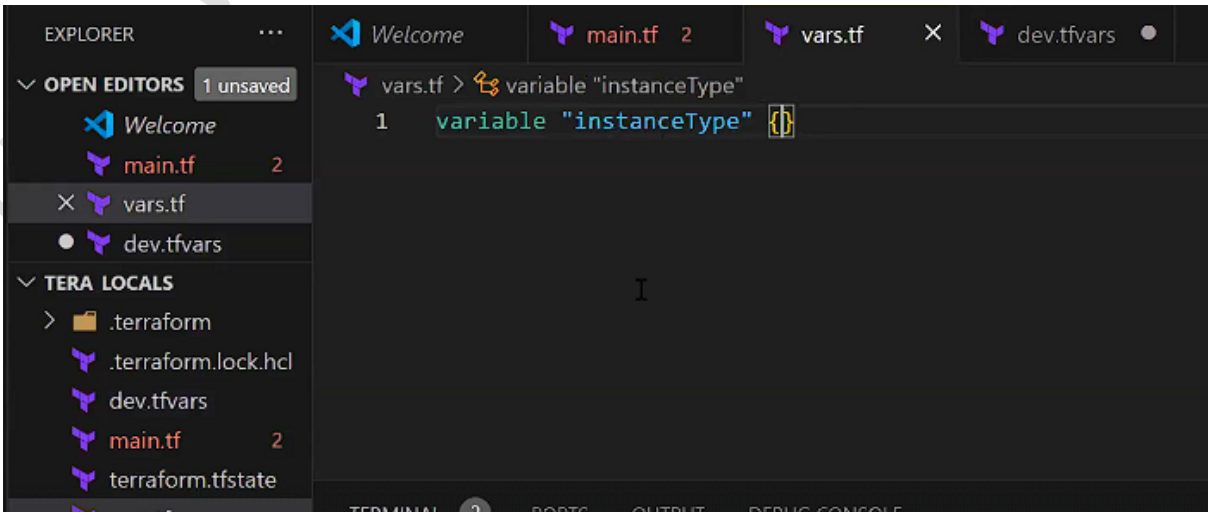




- Create a new tfvars file.



- Suppose you want to declare any variable, and you first have to declare it in some new file. Let's create a file in which we will declare this variable.



- Second, you use the variable somewhere.

```
provider "aws" {
  region = "us-east-1"
}

resource "aws_instance" "name" {
  instance_type = var.instanceType
  ami = "ami-0bb84b8ffd87024d8"

  tags = {
    Name = "dev-media-os"
    Team = "Dev"
```

- And third, you have to input the value either from the Command Prompt or tfvars file.

- Let us destroy it again because we have to relaunch it one more time. And while running the command you have to tell that my input **var** file would be **dev.tfars.**

```
PS C:\Users\Vimal Daga\Documents\terraform_training_2024\tera_locals> terraform.exe destroy -auto-approve
    --var-file=dev.tfvars
```

- Let's define locals in the main.tf file. Here you can give all the variable names whatever name you want to use.

```
provider "aws" {
  region = "us-east-1"
}

locals {
  myenv = "dev"
}
resource "aws_instance" "name" {
  instance_type = var.instanceType
  ami = "ami-0bb84b8ffd87024d8"

  tags = {
    Name = "dev-media-os"
```

- So, if you want to use this variable in other variable files you won't be able to use it. You can use this variable in this file.

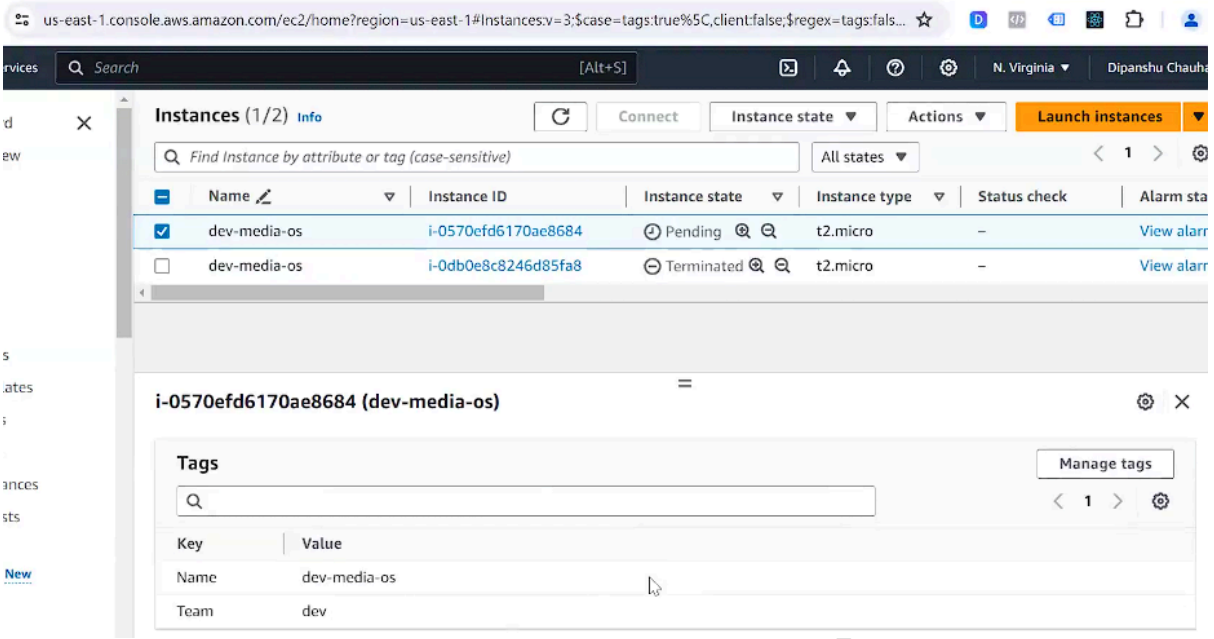
- For example, we can use this variable in the tag.
  - It means this variable is defined somewhere at the top and you have to get the value from there.
  - within the string, if you want to use somewhere the variable then the syntax we use is "**${ variable name }**"

```
Welcome   main.tf   vars.tf   dev.tfvars
main.tf > resource "aws_instance" "name" > tags > Name
3   }
4
5   locals {
6     myenv = "dev"
7   }
8
9   resource "aws_instance" "name" {
10    instance_type = var.instanceType
11    ami = "ami-0bb84b8ffd87024d8"
12
13    tags = {
14      Name = "${local.myenv}-media-os"
15      Team = local.myenv
16    }
17  }
```

- Let's run or apply this code again.

```
Welcome   main.tf   vars.tf   dev.tfvars
main.tf > resource "aws_instance" "name"
4
5   locals {
6     myenv = "dev"
7   }
8
TERMINAL   PORTS   OUTPUT   DEBUG CONSOLE
PROBLEMS                    TERMINAL                    terraform
Filter (e.g. text, **/*.ts, !**/node_mod...
No problems have been detected in the workspace.
PS C:\Users\Vimal Daga\Documents\terraform_training_2024\tera_locals>terraform.exe  apply -auto-approve  --var-file=dev.tfvars
```

- And you will see that new instances have been launched in the AWS cloud.



- **Workspace**

  A workspace is a feature that allows you to manage multiple instances of the same set of infrastructure resources within a single configuration. Each workspace maintains its own state file, allowing you to have different configurations or versions of your infrastructure without interfering with each other.

  - For instance, you might have a development workspace, a staging workspace, and a production workspace, each representing different environments with their own configurations and resources. Workspaces make it easier to manage these environments separately and perform operations like testing changes in isolation before applying them to production.

  - Terraform workspaces are especially useful in scenarios where you want to maintain separate environments with similar configurations, as it helps streamline the management of infrastructure across different stages of development or deployment.

- For example in the terraform, there is a command "**terraform workspace**". And if you hit enter they have their internal options like new, list, show, select, and delete workspaces.

```
TERMINAL   PORTS   OUTPUT   DEBUG CONSOLE
PROBLEMS                    TERMINAL                    powershell
Filter (e.g. text, **/*.ts, !**/node_mod...
No problems have been detected in the workspace.

PS C:\Users\Vimal Daga\Documents\terraform_training_2024\tera_locals>t
erraform  workspace
Usage: terraform [global options] workspace

  new, list, show, select and delete Terraform workspaces.

Subcommands:
    delete    Delete a workspace
    list      List Workspaces
    new       Create a new workspace
    select    Select a workspace
    show      Show the name of the current workspace
PS C:\Users\Vimal Daga\Documents\terraform_training_2024\tera_locals>
```

- Let's say I want to create a new workspace:

```
TERMINAL   PORTS   OUTPUT   DEBUG CONSOLE
PROBLEMS                    TERMINAL                    powershell
Filter (e.g. text, **/*.ts, !**/node_mod...
No problems have been detected in the workspace.

PS C:\Users\Vimal Daga\Documents\terraform_training_2024\tera_locals>t
erraform  workspace  new dev
Created and switched to workspace "dev"!

You're now on a new, empty workspace. Workspaces isolate their state,
so if you run "terraform plan" Terraform will not see any existing sta
te
for this configuration.
```

- Now if you list the workspace, it will give you 2 workspaces. First is default and second is that you just created.

```
9   resource "aws_instance" "name" {
TERMINAL   PORTS   OUTPUT   DEBUG CONSOLE
PROBLEMS                    TERMINAL                    powershell
Filter (e.g. text, **/*.ts, !**/node_mod...
No problems have been detected in the workspace.

PS C:\Users\Vimal Daga\Documents\terraform_training_2024\tera_locals>t
erraform  workspace  list
  default
* dev

PS C:\Users\Vimal Daga\Documents\terraform_training_2024\tera_locals>
```

- Whenever you create any workspace with the new command they create a workspace for you and automatically switch you to that workspace.

```
7   }
8
9   resource "aws_instance" "name" {
```

TERMINAL PORTS OUTPUT DEBUG CONSOLE

PROBLEMS

Filter (e.g. text, **/*.ts, !**/node_mod...

No problems have been detected in the workspace.

TERMINAL powershell

```
PS C:\Users\Vimal Daga\Documents\terraform_training_2024\tera_locals>t
erraform  workspace  new test
Created and switched to workspace "test"!

You're now on a new, empty workspace. Workspaces isolate their state,
```

- So, now we have 3 workspaces.

```
7   }
8
9   resource "aws_instance" "name" {
```

TERMINAL PORTS OUTPUT DEBUG CONSOLE

PROBLEMS

Filter (e.g. text, **/*.ts, !**/node_mod...

No problems have been detected in the workspace.

TERMINAL powershell

```
PS C:\Users\Vimal Daga\Documents\terraform_training_2024\tera_locals>t
erraform  workspace  list
  default
  dev
* test
```

- Let's create one more workspace named **prod .** And the command to check which workspace you are in is **terraform workspace show.**

```
9   resource "aws_instance" "name" {
```

TERMINAL PORTS OUTPUT DEBUG CONSOLE

PROBLEMS

Filter (e.g. text, **/*.ts, !**/node_mod...

No problems have been detected in the workspace

TERMINAL powershell

```
PS C:\Users\Vimal Daga\Documents\terraform_training_2024\tera_locals>t
erraform  workspace  show
prod
PS C:\Users\Vimal Daga\Documents\terraform_training_2024\tera_locals>
```

- But if you want to take the workspace name from the code,by the keyword **terraform.workspace**:

Welcome | main.tf | vars.tf | dev.tfvars

main.tf > output "o1" > value

```
9   resource "aws_instance" "name" {
13    tags = {
14      Name = "${local.myenv}-media-os"
15      Team = local.myenv
16    }
17  }
18
19  output "o1" {
20    value = terraform.workspace
21  }
```

- terraform code has the capability, with the help of the code they will pick your current workspace name.
- If you run the code, you will see that they will able to retrieve your current workspace.

```
TERMINAL   PORTS   OUTPUT   DEBUG CONSOLE
PROBLEMS                    TERMINAL                    terraform
Filter (e.g. text, **/*.ts, !**/node_mod...
No problems have been detected in the workspace.

prod
PS C:\Users\Vimal Daga\Documents\terraform_training_2024\tera_locals>terraform.exe  apply -auto-approve  --var-file=dev.tfvars
```

```
TERMINAL   PORTS   OUTPUT   DEBUG CONSOLE
PROBLEMS                    TERMINAL                    powershell
Filter (e.g. text, **/*.ts, !**/node_mod...
No problems have been detected in the workspace.

Plan: 1 to add, 0 to change, 0 to destroy.

Changes to Outputs:
  + o1 = "prod"
aws_instance.name: Creating...
```

- So, what we can do is, as soon as I switch the workspace to **'dev'** and run the code, my resources will be automatically tagged as **'dev'**.
- But in this case, we are taking the tag from the local variable, and local we have given this dev value hard coded. But instead of giving it hard-coded we can type **terraform.workspace**.

```
Welcome   main.tf   vars.tf   dev.tfvars
main.tf > locals > myenv
1   provider "aws" {
2     region = "us-east-1"
3   }
4
5   locals {
6     myenv = terraform.workspace
7   }
8
9   resource "aws_instance" "name" {
10    instance_type = var.instanceType
11    ami = "ami-0bb84b8ffd87024d8"
12
13    tags = {

terraform.workspace string
The name of the currently selected workspace
```

- As you run this code, it will automatically tag the instance, indicating that someone from the test department has launched it.

- Before running the command first destroy the previously made workspaces.

```
TERMINAL   PORTS   OUTPUT   DEBUG CONSOLE
PROBLEMS                    TERMINAL                    terraform
Filter (e.g. text, **/*.ts, !**/node_mod...
No problems have been detected in the workspace.

PS C:\Users\Vimal Daga\Documents\terraform_training_2024\tera_locals>t
erraform.exe  destroy -auto-approve  --var-file=dev.tfvars
```

- Now create a new workspace.

```
TERMINAL   PORTS   OUTPUT   DEBUG CONSOLE
PROBLEMS                    TERMINAL                    powershell
Filter (e.g. text, **/*.ts, !**/node_mod...
No problems have been detected in the workspace.

PS C:\Users\Vimal Daga\Documents\terraform_training_2024\tera_locals>t
erraform  workspace  select prod
Switched to workspace "prod".
PS C:\Users\Vimal Daga\Documents\terraform_training_2024\tera_locals>t
erraform  workspace  show
prod
PS C:\Users\Vimal Daga\Documents\terraform_training_2024\tera_locals>
```

- Now apply.

```
TERMINAL   PORTS   OUTPUT   DEBUG CONSOLE
PROBLEMS                    TERMINAL                    terraform
Filter (e.g. text, **/*.ts, !**/node_mod...
No problems have been detected in the workspace.

PS C:\Users\Vimal Daga\Documents\terraform_training_2024\tera_locals>t
erraform.exe  apply  -auto-approve  --var-file=dev.tfvars
```

- It will create your instance, but this time we are in the production environment. So, all our instance tags, wherever we write them, will get the name from the workspace.

```
TERMINAL   PORTS   OUTPUT   DEBUG CONSOLE
PROBLEMS                    TERMINAL                    terraform
Filter (e.g. text, **/*.ts, !**/node_mod...
No problems have been detected in the workspace.

      + tags_all                             = {
          + "Name" = "prod-media-os"
          + "Team" = "prod"
        }
      + tenancy                              = (known after apply)
      + user_data                            = (known after apply)
      + user_data_base64                     = (known after apply)
      + user_data_replace_on_change          = false
      + vpc_security_group_ids               = (known after apply)
    }

Plan: 1 to add, 0 to change, 0 to destroy.
aws_instance.name: Creating...
```

[Terraform]