



## 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_locals
$ ls
$ ls

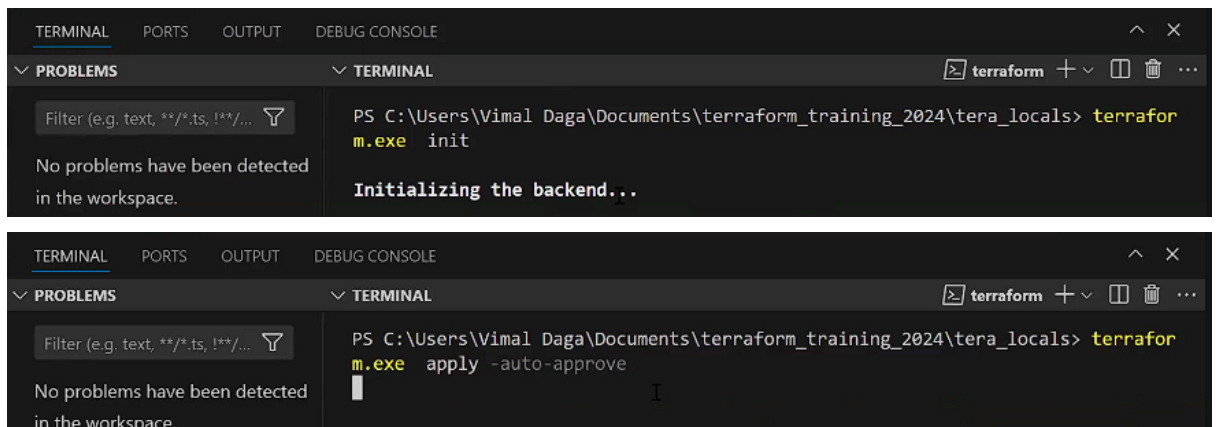
Vimal Daga@DESKTOP-3E1AGGT MINGW64 ~/Documents/terraform_training_2024/tera_locals
$ code .
```

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

```
File Edit Selection ... < > tera_locals
Welcome main.tf x
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"
```

[Terraform]

- Now quickly initialize and launch this code.



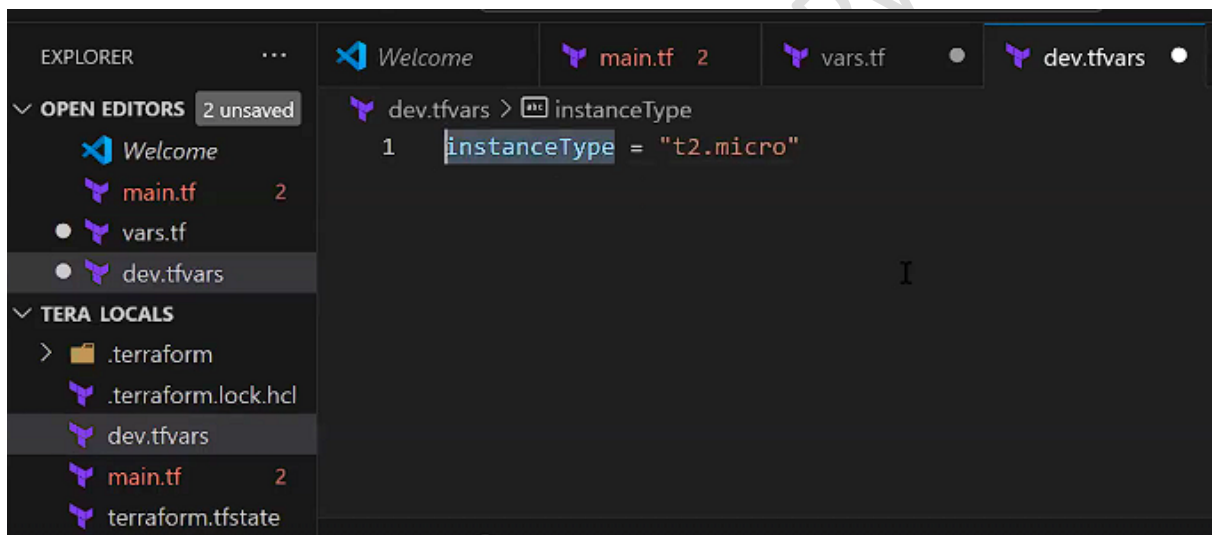
The first screenshot shows a terminal window with the command `terraform init` being executed. The output is `Initializing the backend...`. The second screenshot shows the command `terraform apply -auto-approve` being executed.

```
PS C:\Users\Vimal Daga\Documents\terraform_training_2024\tera_locals> terraform
m.exe init

Initializing the backend...

PS C:\Users\Vimal Daga\Documents\terraform_training_2024\tera_locals> terrafor
m.exe apply -auto-approve
```

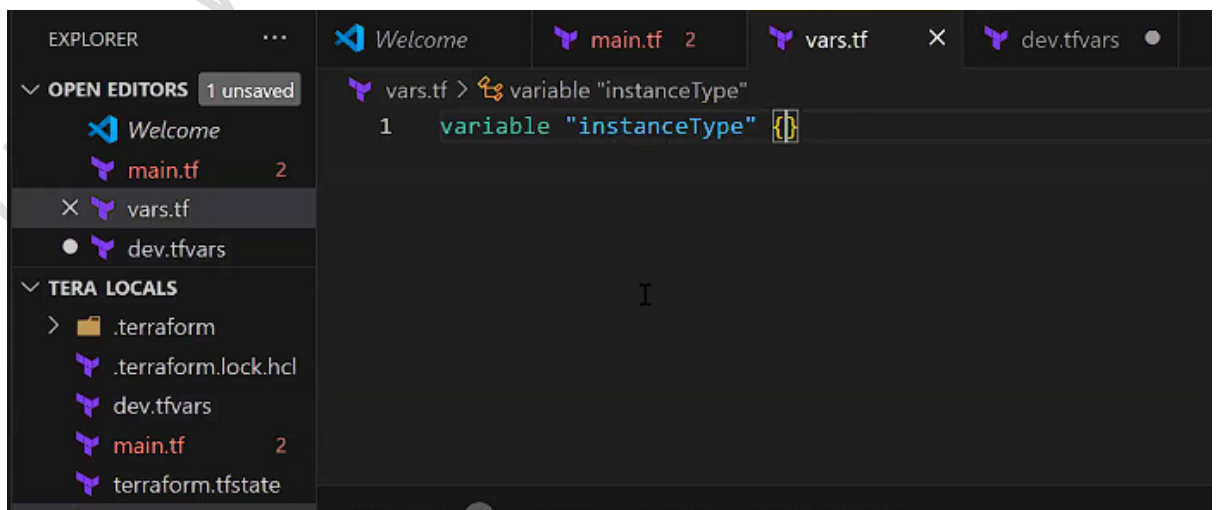
- Create a new tfvars file.



The screenshot shows the VS Code interface with the `dev.tfvars` file open. The file contains the following content:

```
dev.tfvars > instanceType
1 instanceType = "t2.micro"
```

- 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.

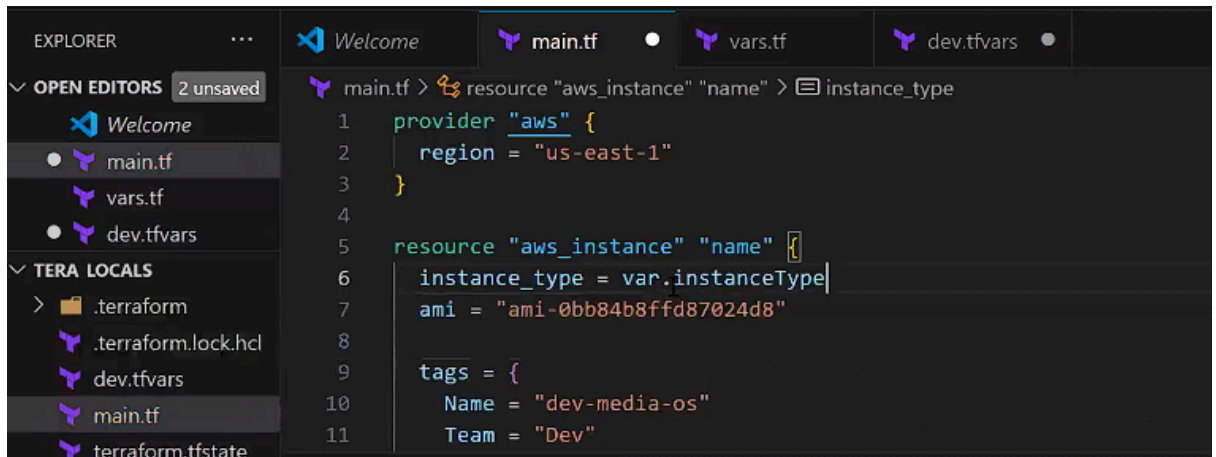


The screenshot shows the VS Code interface with the `vars.tf` file open. The file contains the following content:

```
vars.tf > variable "instanceType"
1 variable "instanceType"
```

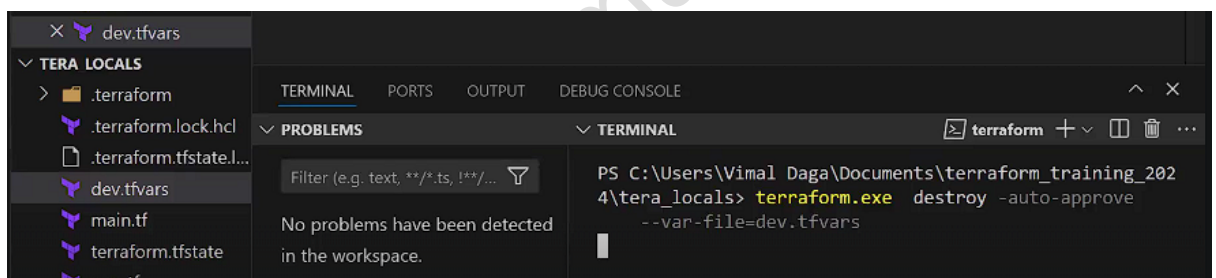
- Second, you use the variable somewhere.

[Terraform]



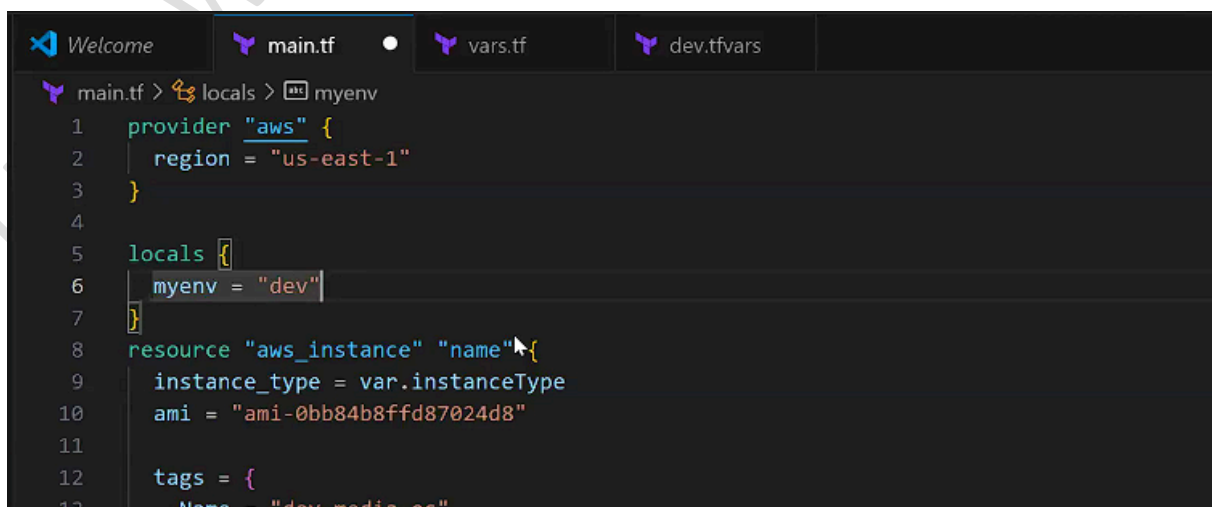
```
main.tf > resource "aws_instance" "name" > instance_type
1  provider "aws" {
2      region = "us-east-1"
3  }
4
5  resource "aws_instance" "name" {
6      instance_type = var.instanceType
7      ami = "ami-0bb84b8ffd87024d8"
8
9      tags = {
10         Name = "dev-media-os"
11         Team = "Dev"
12     }
```

- 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.tfvars**.



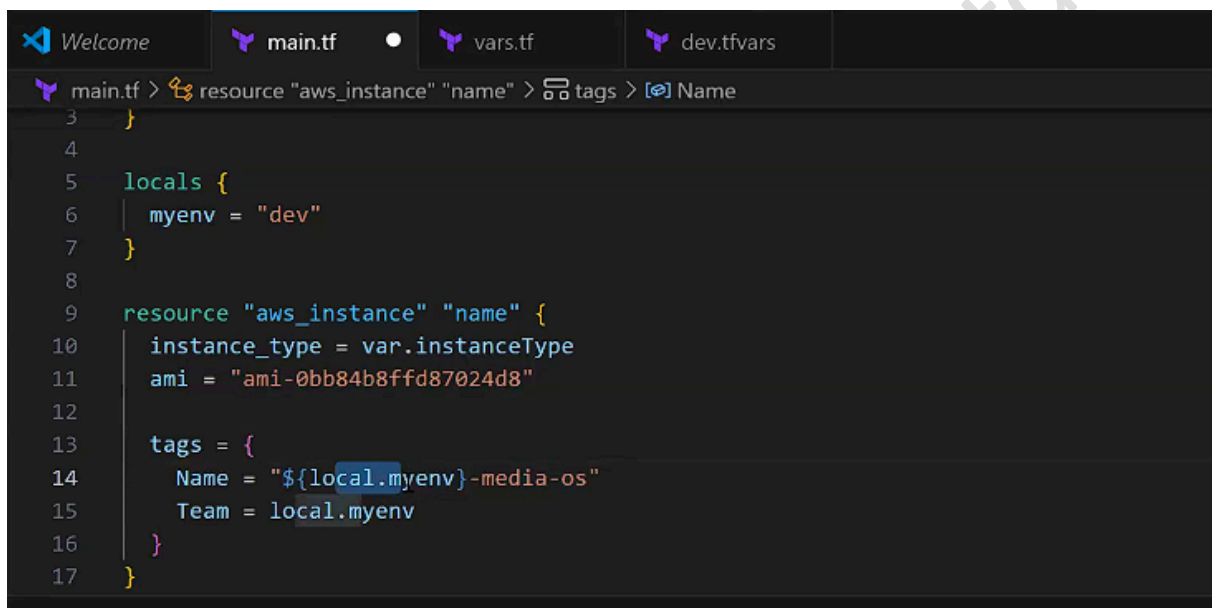
```
PS C:\Users\Vimal Daga\Documents\terraform_training_2024\tera_locals> terraform.exe destroy -auto-approve --var-file=dev.tfvars
```

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



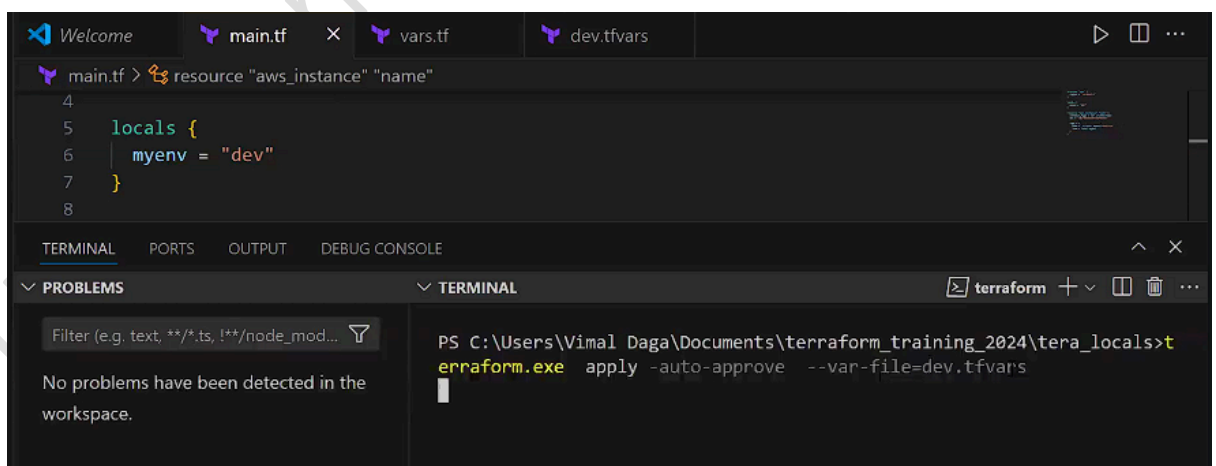
```
main.tf > locals > myenv
1  provider "aws" {
2      region = "us-east-1"
3  }
4
5  locals {
6      myenv = "dev"
7  }
8  resource "aws_instance" "name" {
9      instance_type = var.instanceType
10     ami = "ami-0bb84b8ffd87024d8"
11
12     tags = {
13         Name = "dev-media-os"
```

- 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 }`**”



```
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.

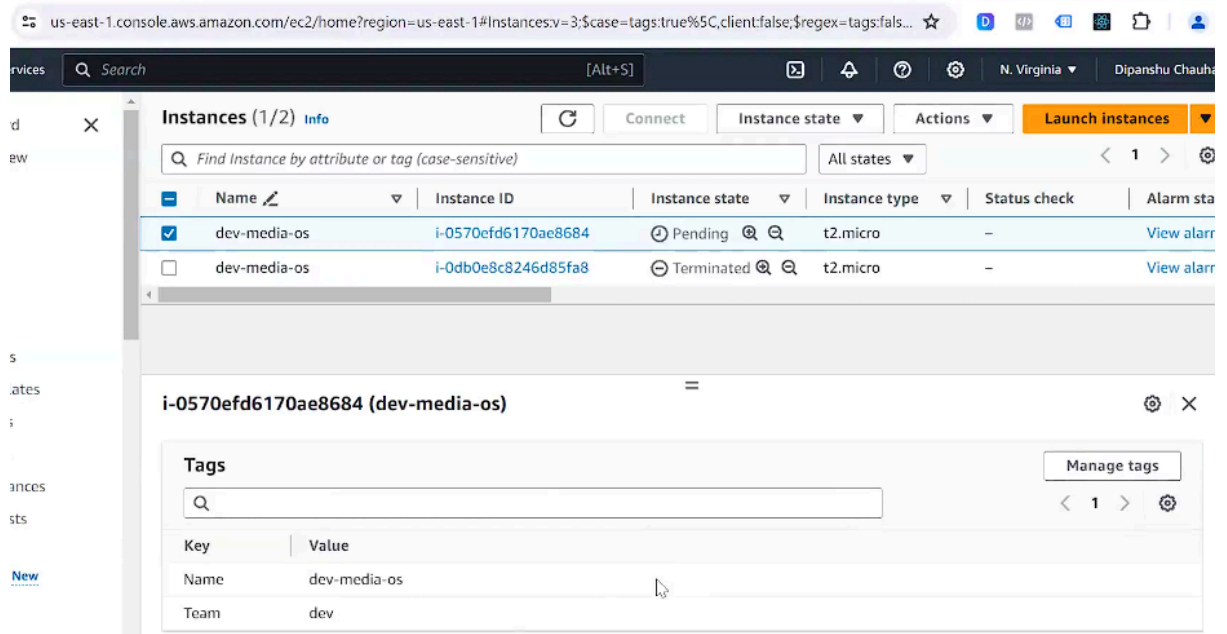


```
main.tf > resource "aws_instance" "name"
4
5   locals {
6     myenv = "dev"
7   }
8
9   resource "aws_instance" "name" {
10    instance_type = var.instanceType
11    ami = "ami-0bb84b8ffd87024d8"
12
13    tags = {
14      Name = "${local.myenv}-media-os"
15      Team = local.myenv
16    }
17  }
```

TERMINAL

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.



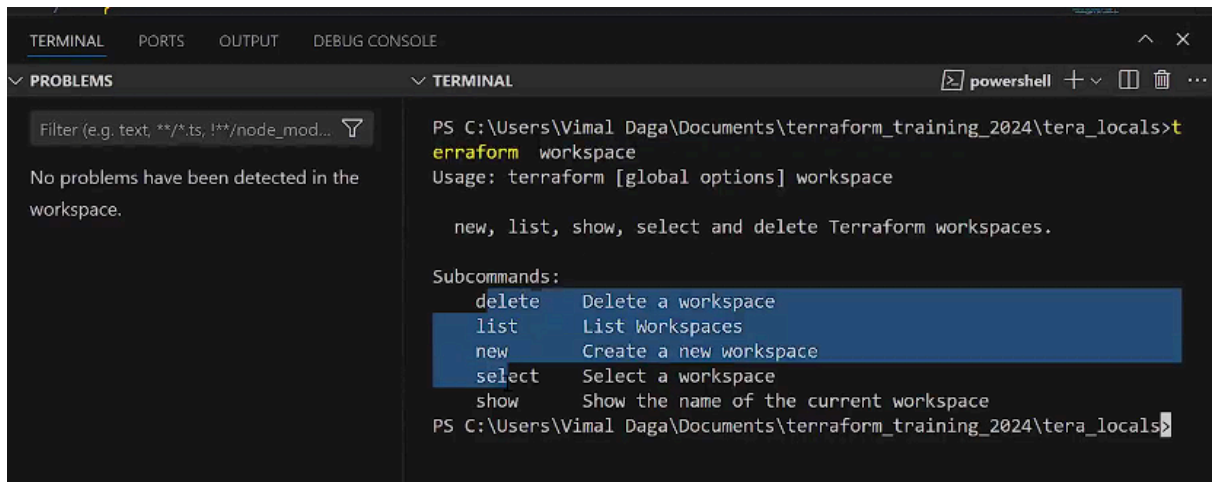
- **Workspace**

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.

[Terraform]

- 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.

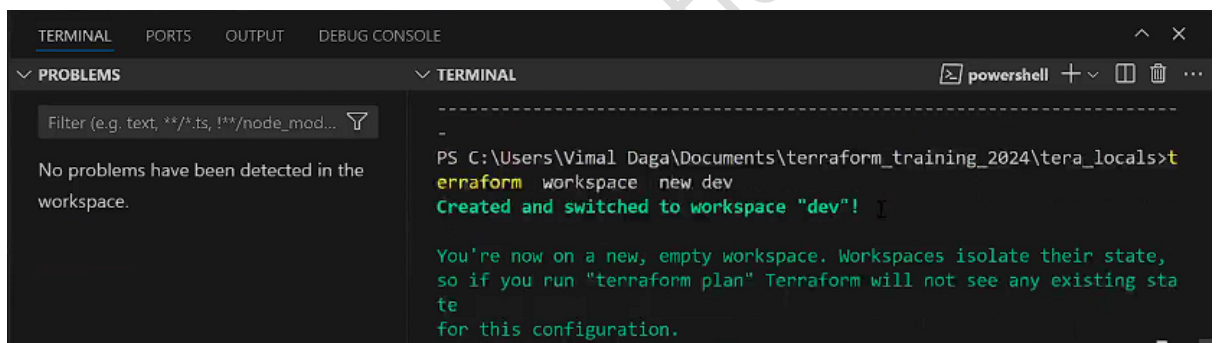


```
PS C:\Users\Vimal Daga\Documents\terraform_training_2024\tera_locals> terraform workspace
Usage: terraform [global options] workspace

    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>
```

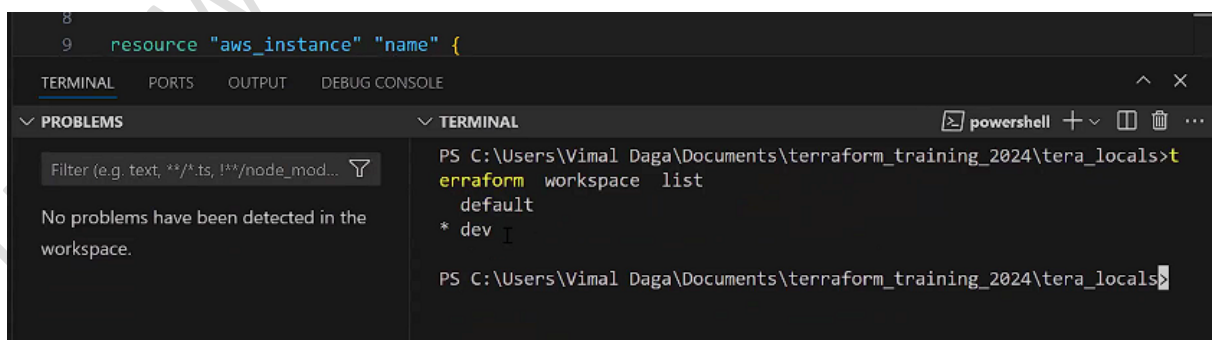
- Let's say I want to create a new workspace:



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

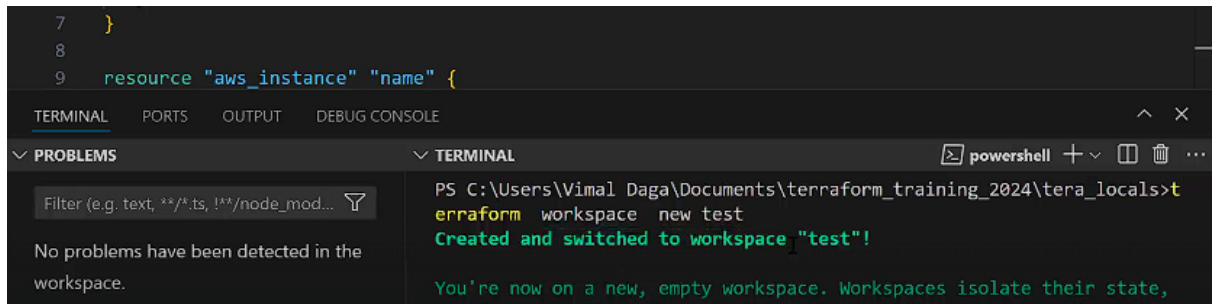


```
PS C:\Users\Vimal Daga\Documents\terraform_training_2024\tera_locals> terraform workspace list
default
* dev
```

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



[Terraform]

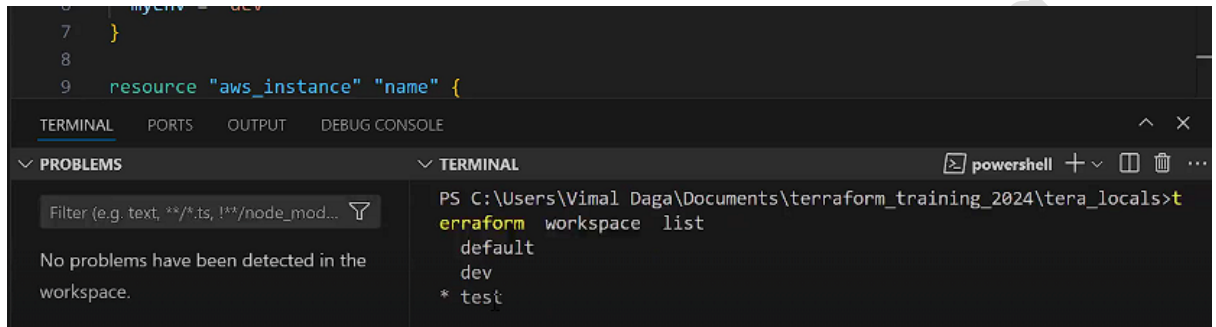


```
7 }
8
9 resource "aws_instance" "name" {
  ...
}
```

TERMINAL

PS C:\Users\Vimal Daga\Documents\terraform\_training\_2024\tera\_locals> terraform 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.

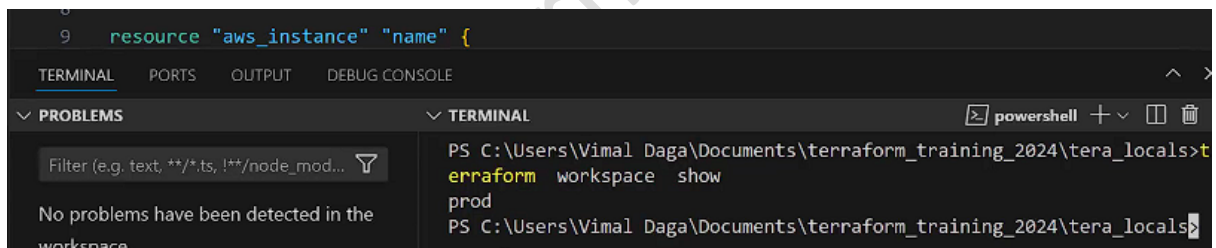


```
8 myenv = dev
7 }
8
9 resource "aws_instance" "name" {
  ...
}
```

TERMINAL

PS C:\Users\Vimal Daga\Documents\terraform\_training\_2024\tera\_locals> terraform 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**.

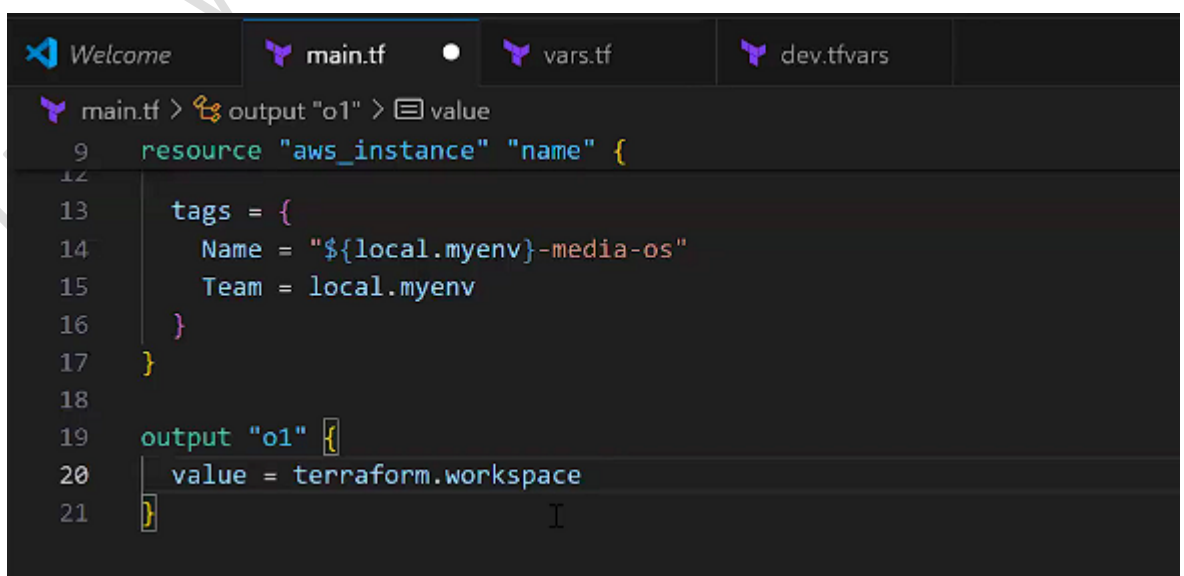


```
9 resource "aws_instance" "name" {
  ...
}
```

TERMINAL

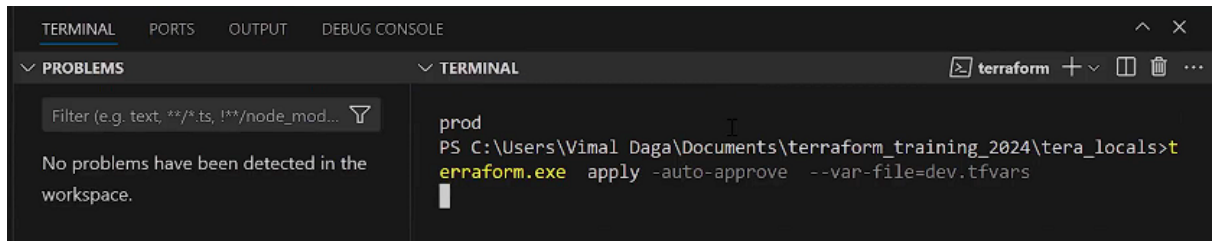
PS C:\Users\Vimal Daga\Documents\terraform\_training\_2024\tera\_locals> terraform workspace show  
prod  
PS C:\Users\Vimal Daga\Documents\terraform\_training\_2024\tera\_locals>

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

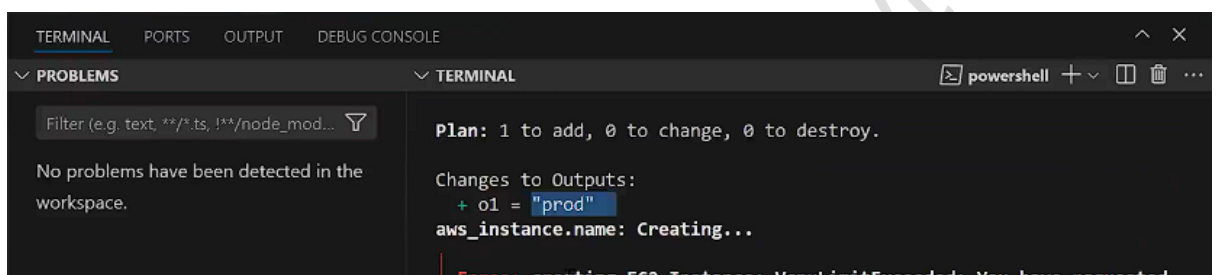


```
Welcome | main.tf | vars.tf | dev.tfvars
main.tf > output "o1" > value
9 resource "aws_instance" "name" {
12
13   tags = {
14     Name = "${local.myenv}-media-os"
15     Team = local.myenv
16   }
17 }
18
19 output "o1" {
20   value = terraform.workspace
21 }
```

- 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 be able to retrieve your current workspace.

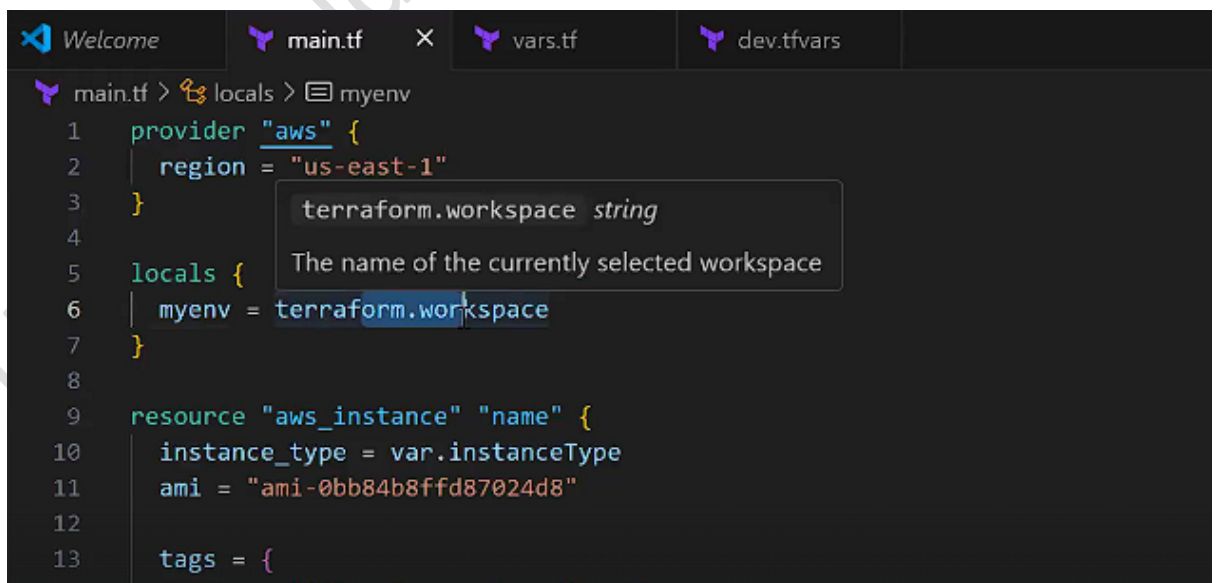


The screenshot shows a VS Code terminal window with the 'terminal' tab selected. The command prompt is 'PS C:\Users\Vimal Daga\Documents\terraform\_training\_2024\tera\_locals> terraform.exe apply -auto-approve --var-file=dev.tfvars'. The output shows the plan: 'Plan: 1 to add, 0 to change, 0 to destroy.' and the changes to outputs: '+ o1 = "prod"'. The aws\_instance.name is 'Creating...'. The error message at the bottom is 'Error: creating EC2 Instance: VcpuLimitExceeded: You have requested'.



The screenshot shows a VS Code terminal window with the 'terminal' tab selected. The command prompt is 'PS C:\Users\Vimal Daga\Documents\terraform\_training\_2024\tera\_locals> terraform.exe apply -auto-approve --var-file=dev.tfvars'. The output shows the plan: 'Plan: 1 to add, 0 to change, 0 to destroy.' and the changes to outputs: '+ o1 = "prod"'. The aws\_instance.name is 'Creating...'. The error message at the bottom is 'Error: creating EC2 Instance: VcpuLimitExceeded: You have requested'.

- 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**.



The screenshot shows the VS Code editor with the 'main.tf' file open. The code is as follows:

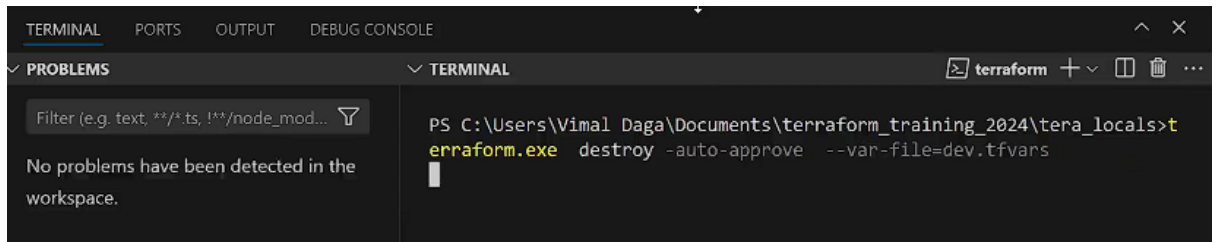
```
1 provider "aws" {
2   region = "us-east-1"
3 }
4
5 locals {
6   myenv = terraform.workspace
7 }
8
9 resource "aws_instance" "name" {
10   instance_type = var.instanceType
11   ami = "ami-0bb84b8ffd87024d8"
12
13   tags = {
```

A tooltip is visible over the `terraform.workspace` variable, showing the text: 'terraform.workspace string' and '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.

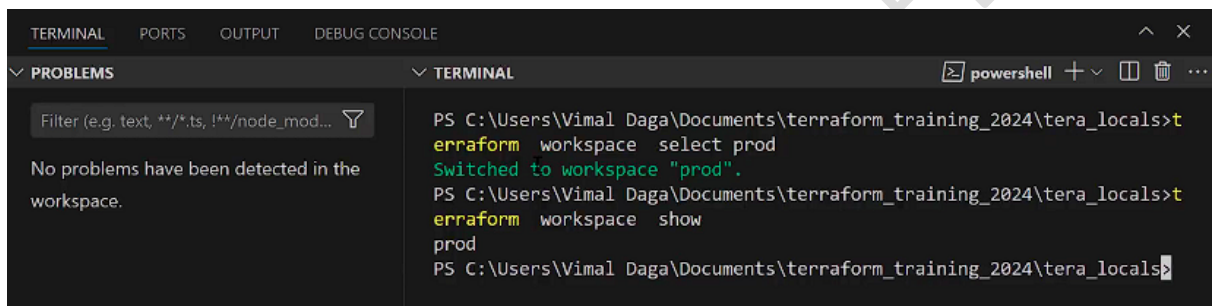


- Before running the command first destroy the previously made workspaces.



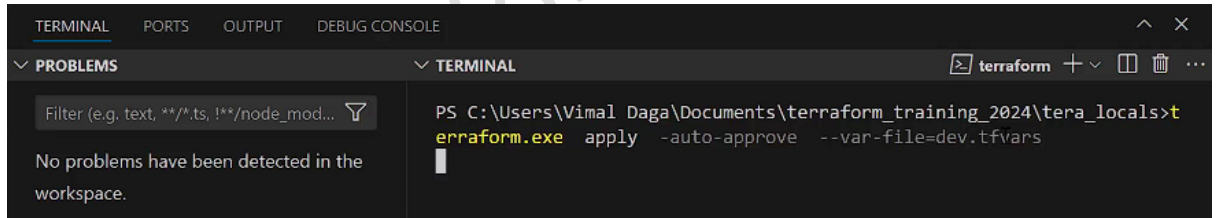
```
PS C:\Users\Vimal Daga\Documents\terraform_training_2024\tera_locals>terraform.exe destroy -auto-approve --var-file=dev.tfvars
```

- Now create a new workspace.



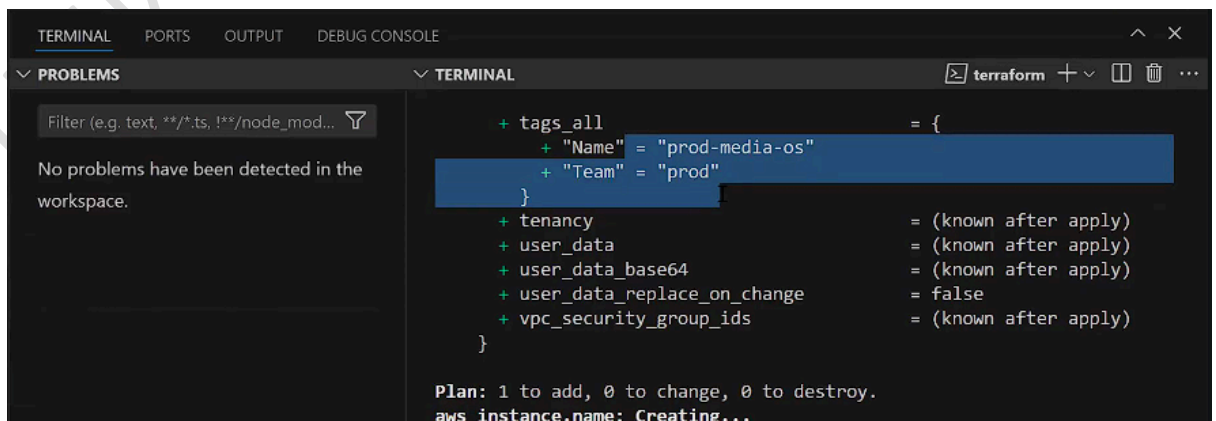
```
PS C:\Users\Vimal Daga\Documents\terraform_training_2024\tera_locals>terraform workspace select prod
Switched to workspace "prod".
PS C:\Users\Vimal Daga\Documents\terraform_training_2024\tera_locals>terraform workspace show
prod
PS C:\Users\Vimal Daga\Documents\terraform_training_2024\tera_locals>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

- Now apply.



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



```
+ 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]

us-east-1.console.aws.amazon.com/ec2/home?region=us-east-1#instances?v=3;\$case=tags:true%5C,client:false;\$regex=tags:fals...

Services Search [Alt+S] N. Virginia Dipanshu Chauhan

EC2 Dashboard  
EC2 Global View  
Events  
Console-to-Code [Preview](#)

▼ Instances  
Instances  
Instance Types  
Launch Templates  
Spot Requests  
Savings Plans  
Reserved Instances  
Dedicated Hosts  
Capacity  
Reservations [New](#)

Successfully terminated i-0570efd6170ae8684

Instances (1/3) Info [Refresh](#) [Connect](#) Instance state Instance type Actions [Launch instances](#)

Find Instance by attribute or tag (case-sensitive) All states < 1 >

	Name	Instance ID	Instance state	Instance type	Status check	Alarm status
<input type="checkbox"/>	dev-media-os	i-0570efd6170ae8684	Terminated	t2.micro	—	<a href="#">View alarms</a>
<input checked="" type="checkbox"/>	prod-media-os	i-070a8312d3a3b1bc1	Running	t2.micro	Initializing	<a href="#">View alarms</a>
<input type="checkbox"/>	dev-media-os	i-0db0e8c8246d85fa8	Terminated	t2.micro	—	<a href="#">View alarms</a>

i-070a8312d3a3b1bc1 (prod-media-os)

Tags [Manage tags](#)

Key Value

Name	prod-media-os
Team	prod