**Terraform Workspace**

**Assignment:**

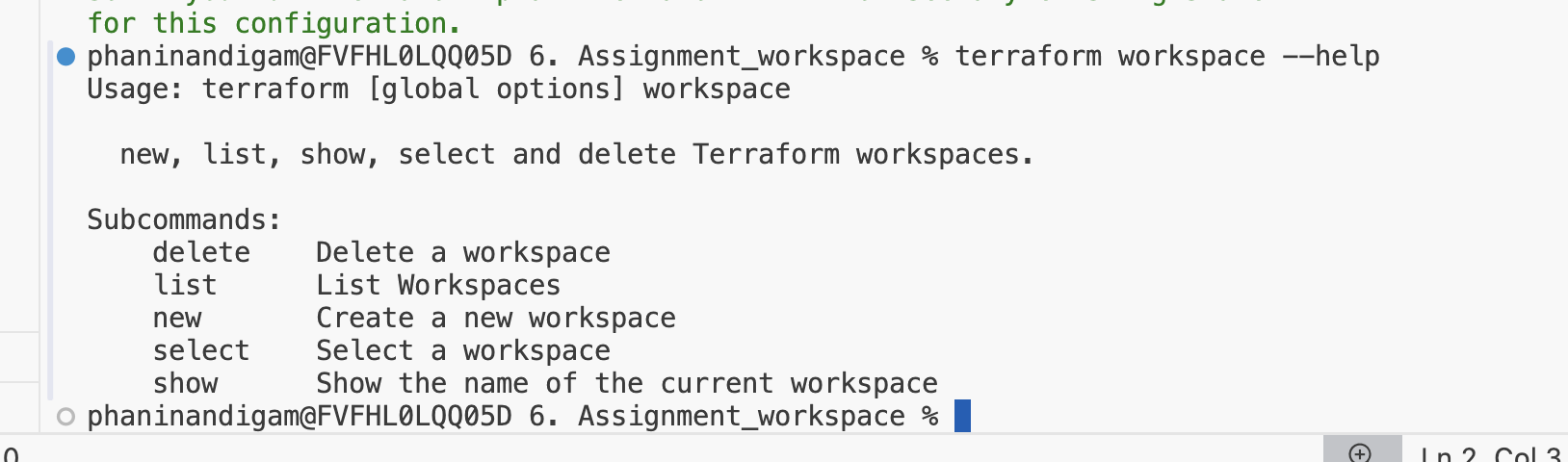
You are managing infrastructure for multiple environments (dev, staging, and prod) using Terraform workspaces. Your goal is to create an S3 bucket and an EC2 instance, where:

The S3 bucket name should be prefixed with the \*workspace name.\*

The EC2 \*instance type\* should vary based on the \*workspace\* (t2.micro for dev, t3.small for staging, and t3.medium for prod).

Store Terraform state in an \*S3 backend\* with \*state locking\* enabled using a DynamoDB table.

Use \*variables\* for flexibility, locals for calculated values, and functions to dynamically construct names.

Note: Terraform workspace commands  


Steps:

* Create 3 workspaces
  + terraform workspace new dev
  + terraform workspace new staging
  + terraform workspace new prod

A screenshot of a computer

AI-generated content may be incorrect.

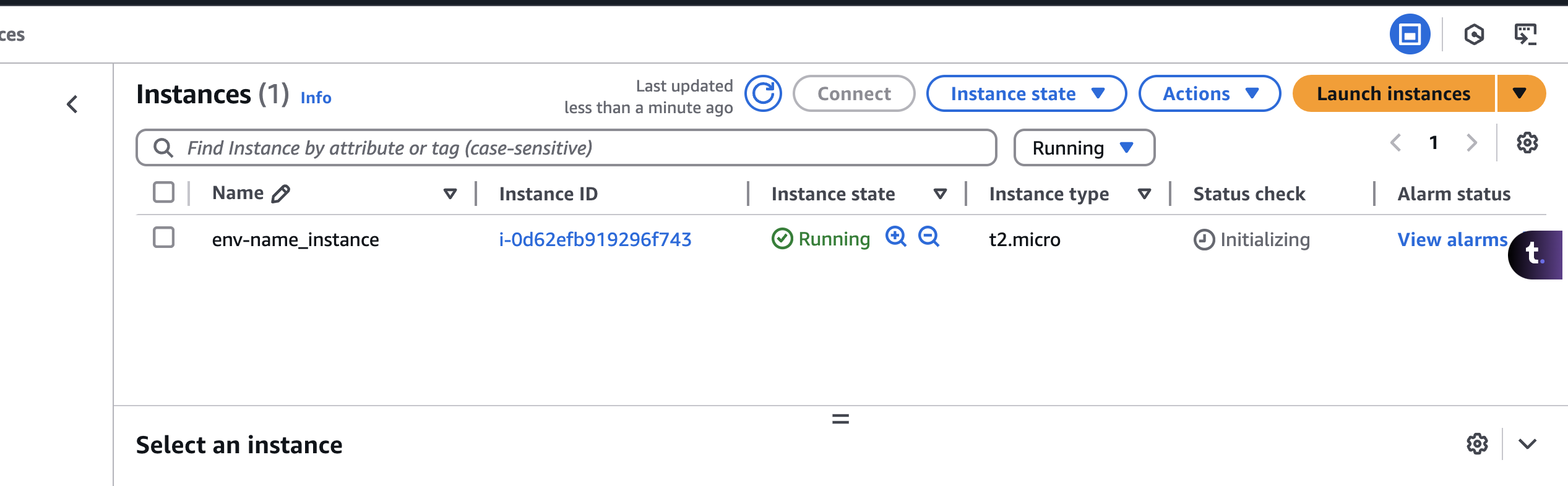
* Now, let’s write the code to launch the instance

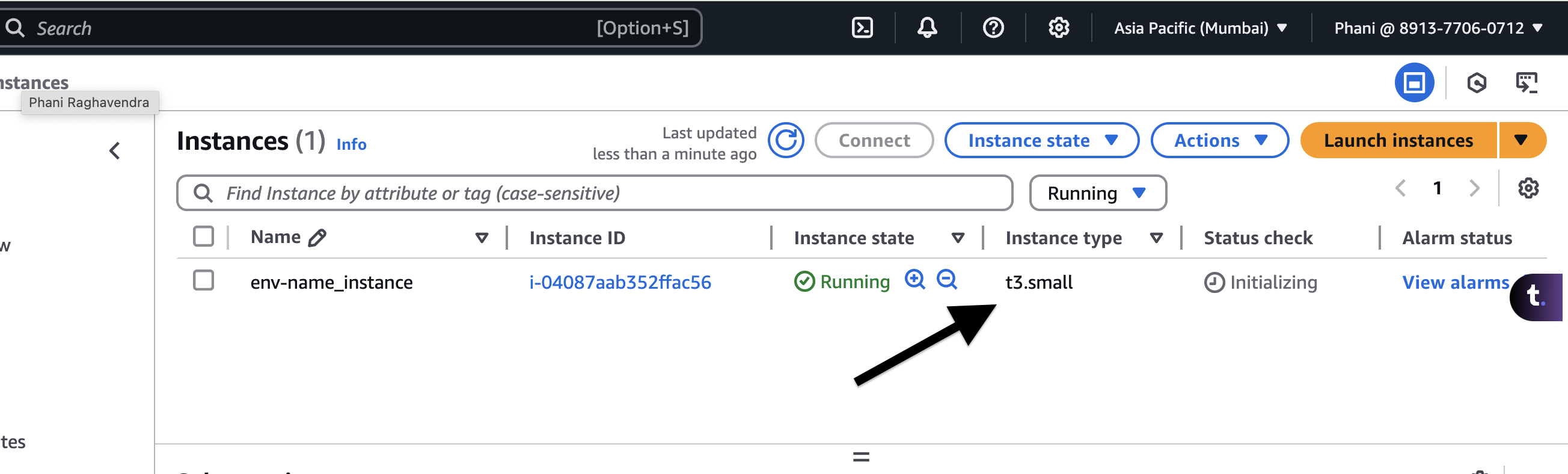
|  |
| --- |
| provider "aws" {  region = "ap-south-1"  }  resource "aws\_instance" "instance\_block" {  ami = "ami-00bb6a80f01f03502"  instance\_type = "t2.micro"  tags = {  Name ="env-name\_instance"  }    } |

* Let’s try creating this instance in one of the workspace
  + switch to the workspace and run the terraform apply command
  + terraform workspace select dev
  + terraform apply
  + Now this will create an tf state file only for the dev workspace.
* Now let’s try to set up the instance type based on the workspace using locals  
  “The EC2 \*instance type\* should vary based on the \*workspace\* (t2.micro for dev, t3.small for staging, and t3.medium for prod).”

Note: Here terraform.workspace is called as workspace interpolation, it will return the current workspace name.

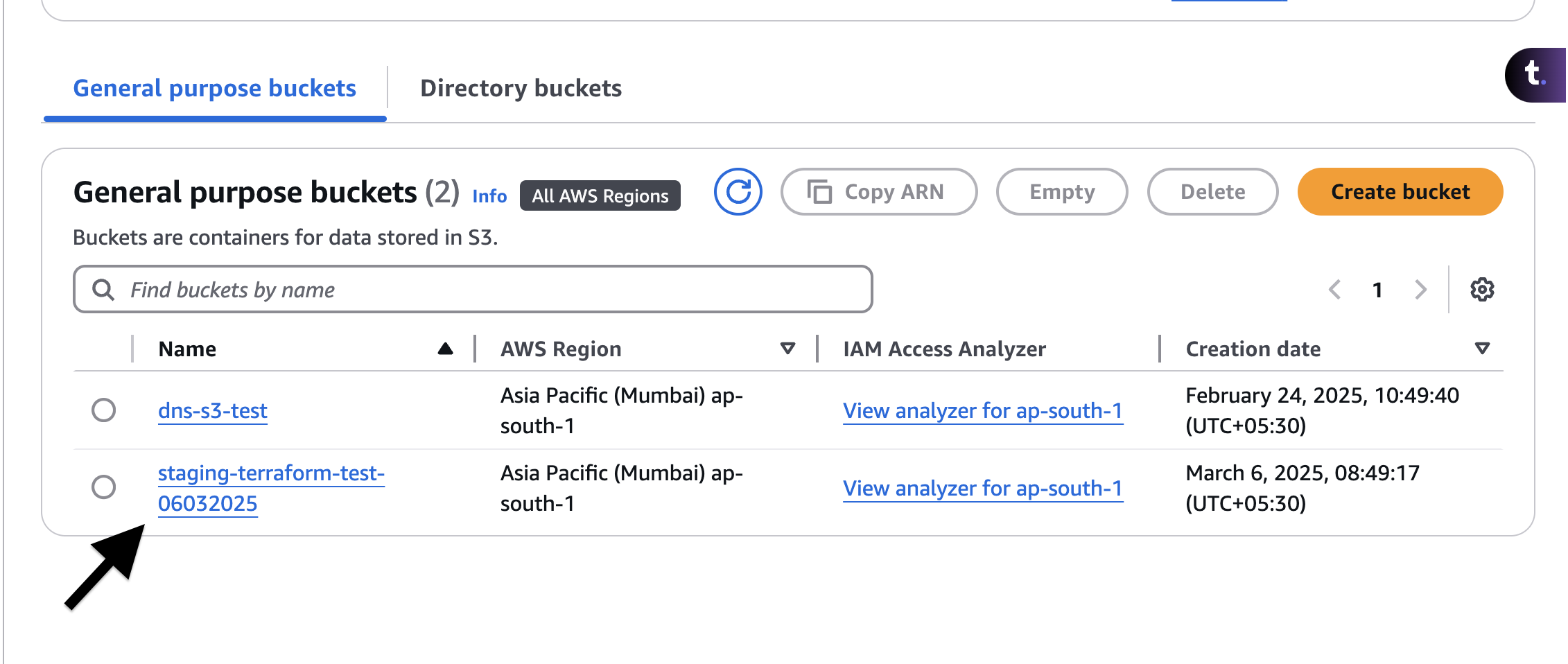
|  |
| --- |
| provider "aws" {  region = "ap-south-1"  }  resource "aws\_instance" "instance\_block" {  ami = "ami-00bb6a80f01f03502"  instance\_type = "${local.instance[terraform.workspace]}"  tags = {  Name ="env-name\_instance"  }    }  locals {  instance = {  "default" = "t2.micro"  "dev" = "t2.micro"  "staging" = "t3.small"  "prod" = "t3.medium"  }  } |

Dev:  


Staging:  
terraform workspace list  
terraform workspace select staging  
terraform apply  


* Now, let’s write a config file to create a s3 bucket  
  “The S3 bucket name should be prefixed with the \*workspace name.\*”

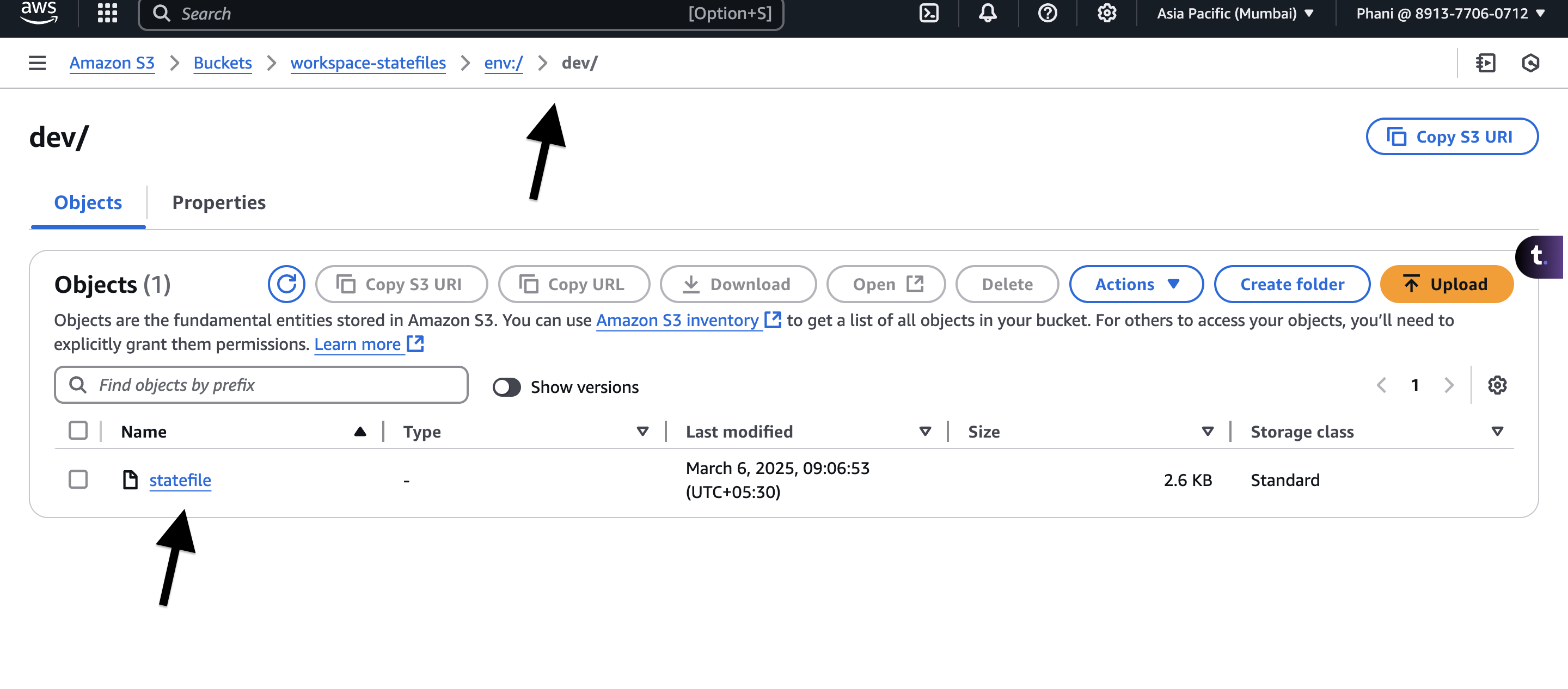
|  |
| --- |
| provider "aws" {  region = "ap-south-1"  }  # resource "aws\_instance" "instance\_block" {  # ami = "ami-00bb6a80f01f03502"  # instance\_type = "${local.instance[terraform.workspace]}"  # tags = {  # Name ="env-name\_instance"  # }    # }  locals {  instance = {  "default" = "t2.micro"  "dev" = "t2.micro"  "staging" = "t3.small"  "prod" = "t3.medium"  }  }  resource "aws\_s3\_bucket" "terraform\_bucket" {  bucket = "${terraform.workspace}-terraform-test-06032025"    } |



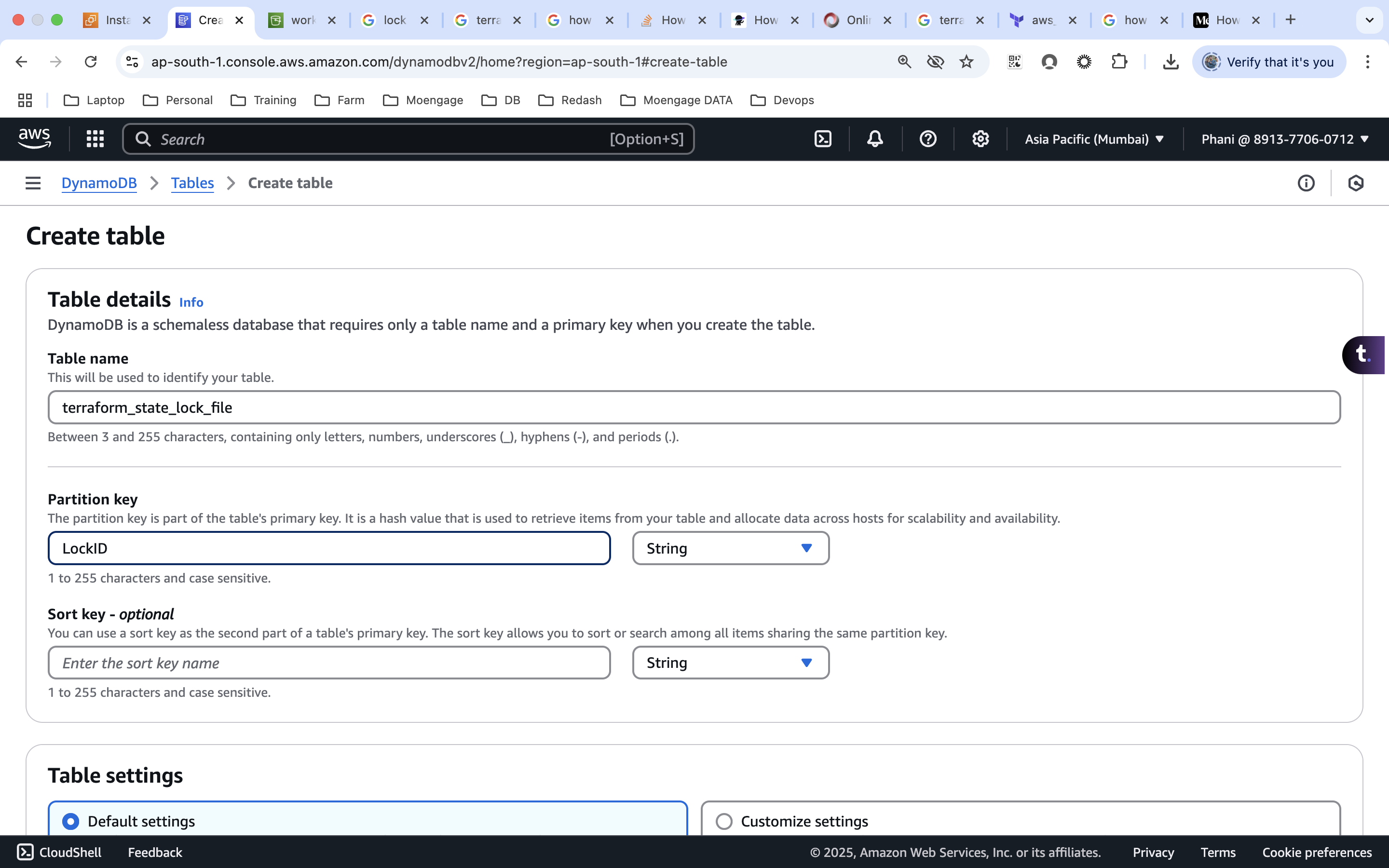
* Now, let’s work on the backend state file.
* Usually, the state file will be stored locally, but in this case, we will be storing it in the s3 bucket.  
  <https://developer.hashicorp.com/terraform/language/backend>
* Create an s3 bucket
* once the s3 bucket has been created, we need to create the workspaces. the reason is we are no longer managing the workspaces locally.
* Also, every time we make any changes to the backend block, we need to initialize it again
* Note: Once we setup the backend, don’t make any changes to it. If we do it we need to setup the workspace again

|  |
| --- |
| terraform {  backend "s3" {  bucket = "workspace-statefiles" #bucketname  key = "statefile" #path  region = "ap-south-1"    }  } |

* A screenshot of a computer

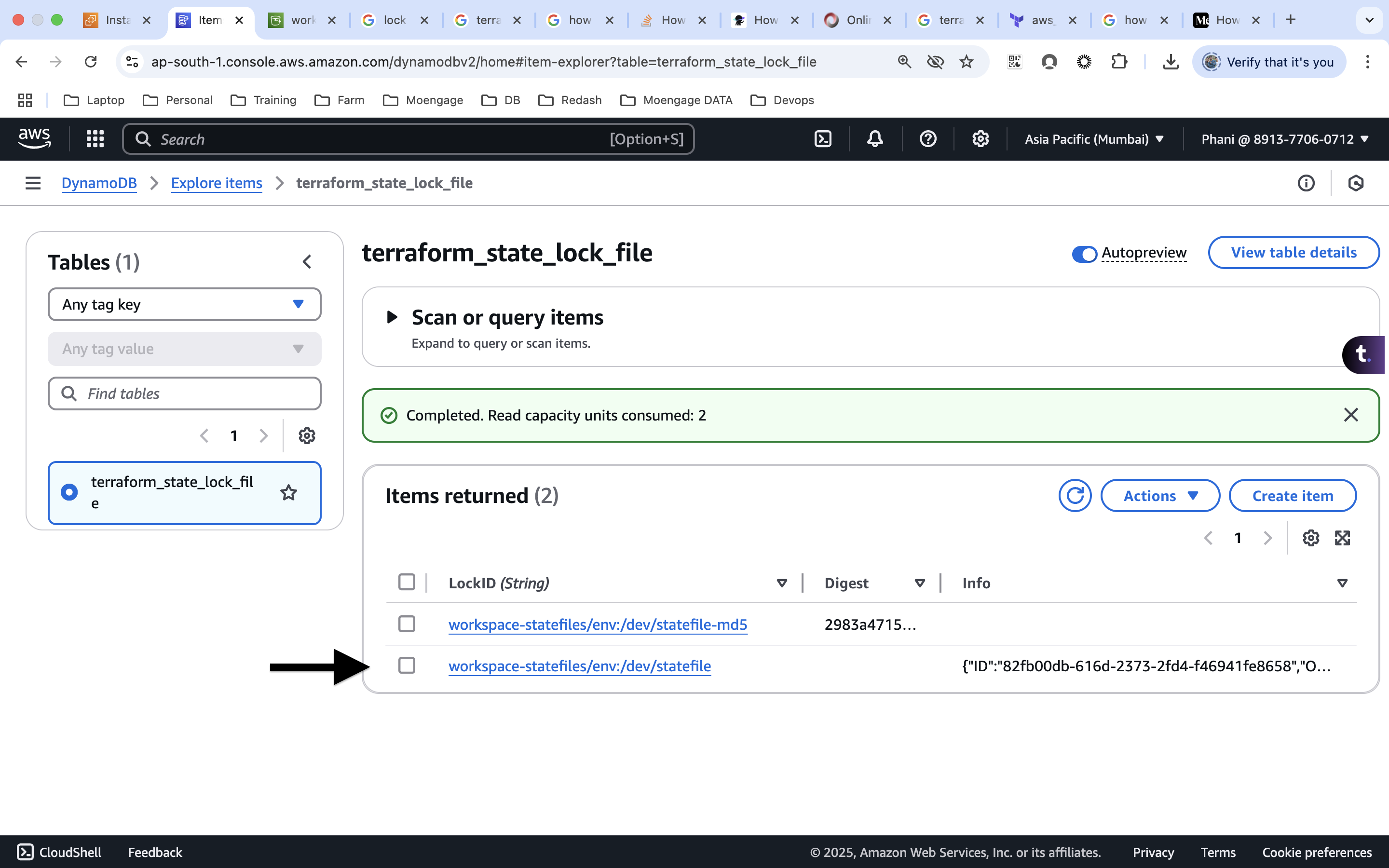
  AI-generated content may be incorrect.  
    
  

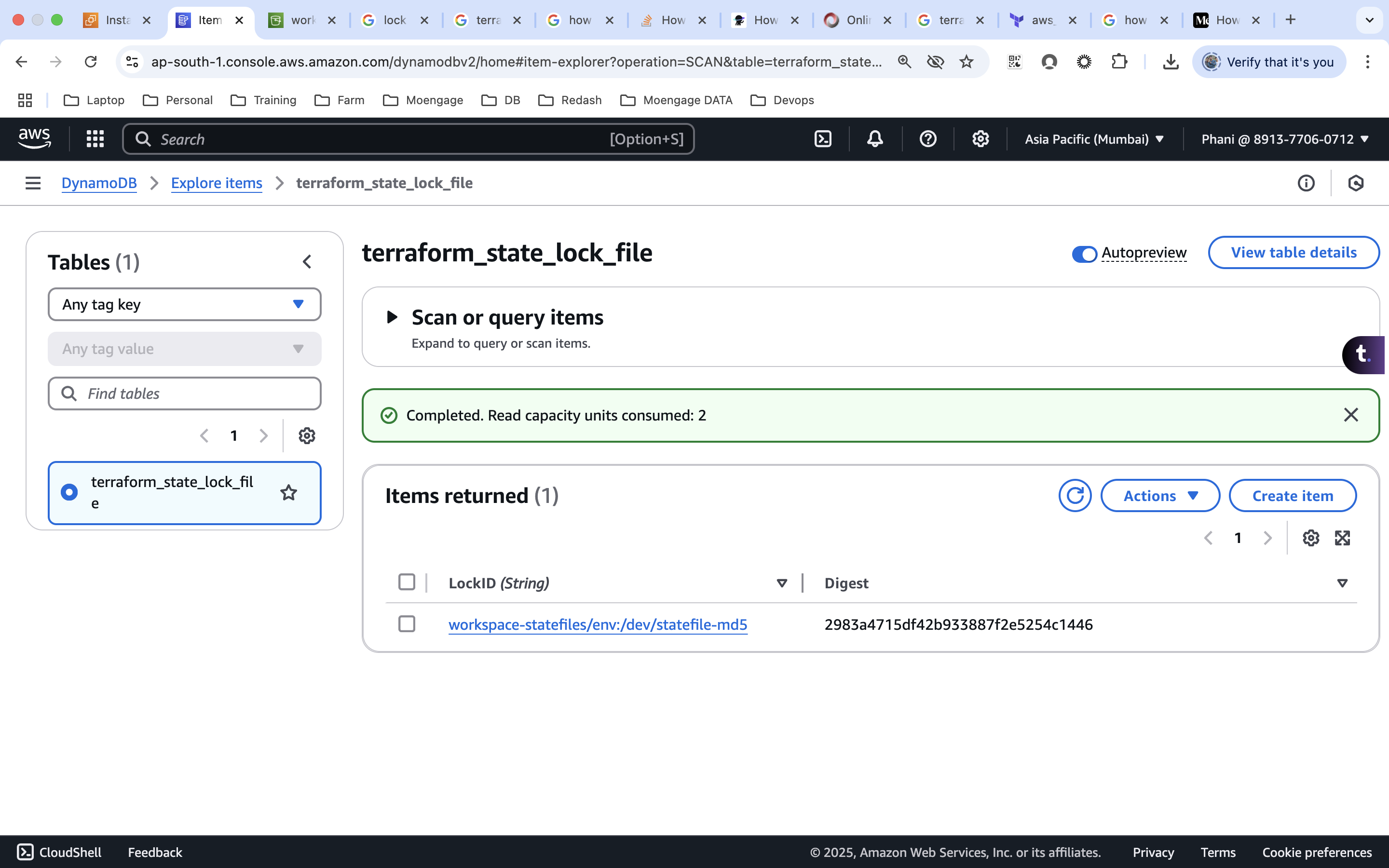


* We need to setup the lock file, so that 2 users cannot perfrom the apply or destroy action at the same time.  
  “Store Terraform state in an \*S3 backend\* with \*state locking\* enabled using a DynamoDB table.”
* we have created the s3 bucket and storing the state files
* Let’s create a dynamo db to store the lock file. Create atable with any name an user LockID as the partition key  
    
  
* Now, we need to update the backend code.

|  |
| --- |
| terraform {  backend "s3" {  bucket = "workspace-statefiles" #bucketname  key = "statefile" #path  region = "ap-south-1"  dynamodb\_table = "terraform\_state\_lock\_file"  }  } |

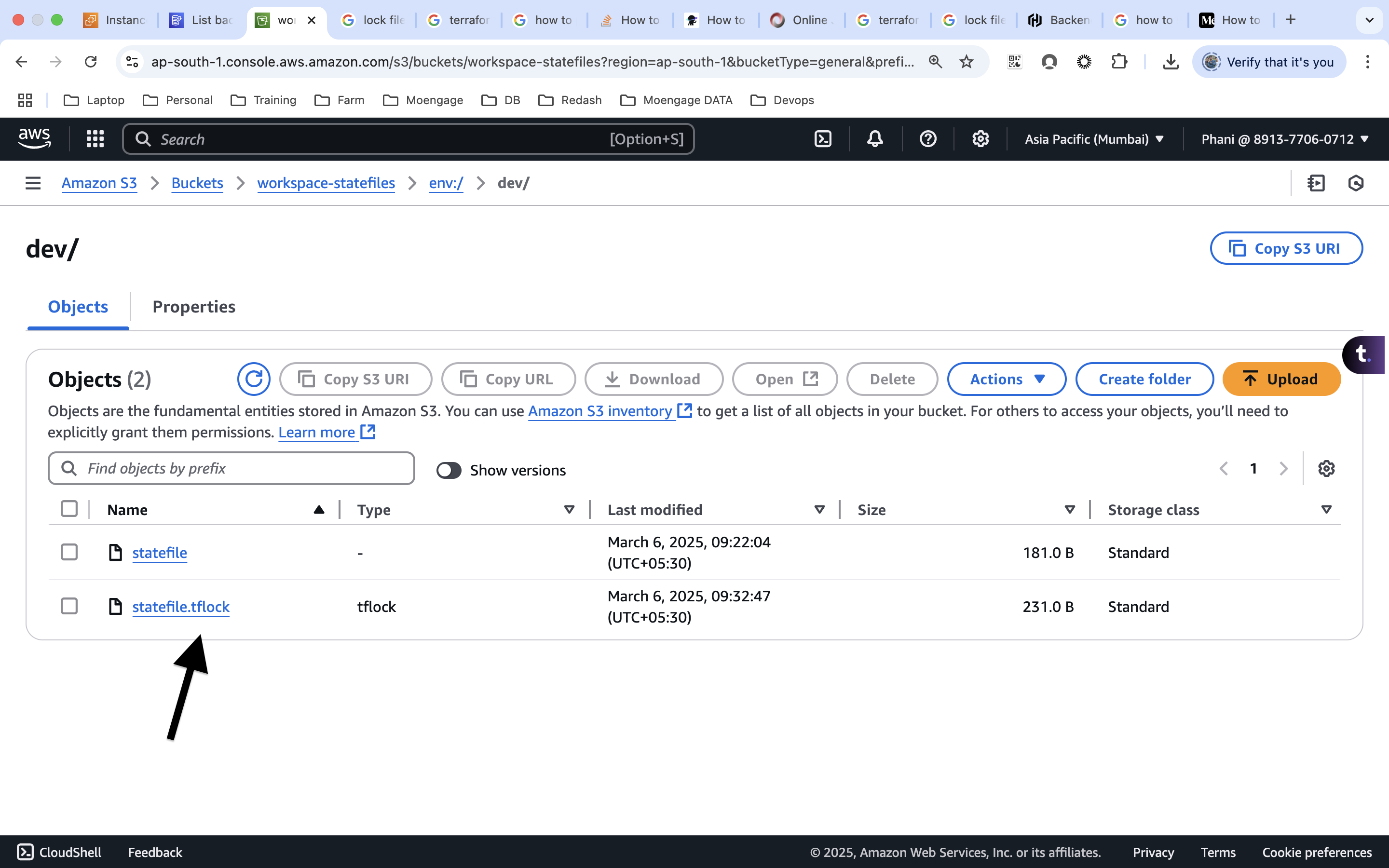
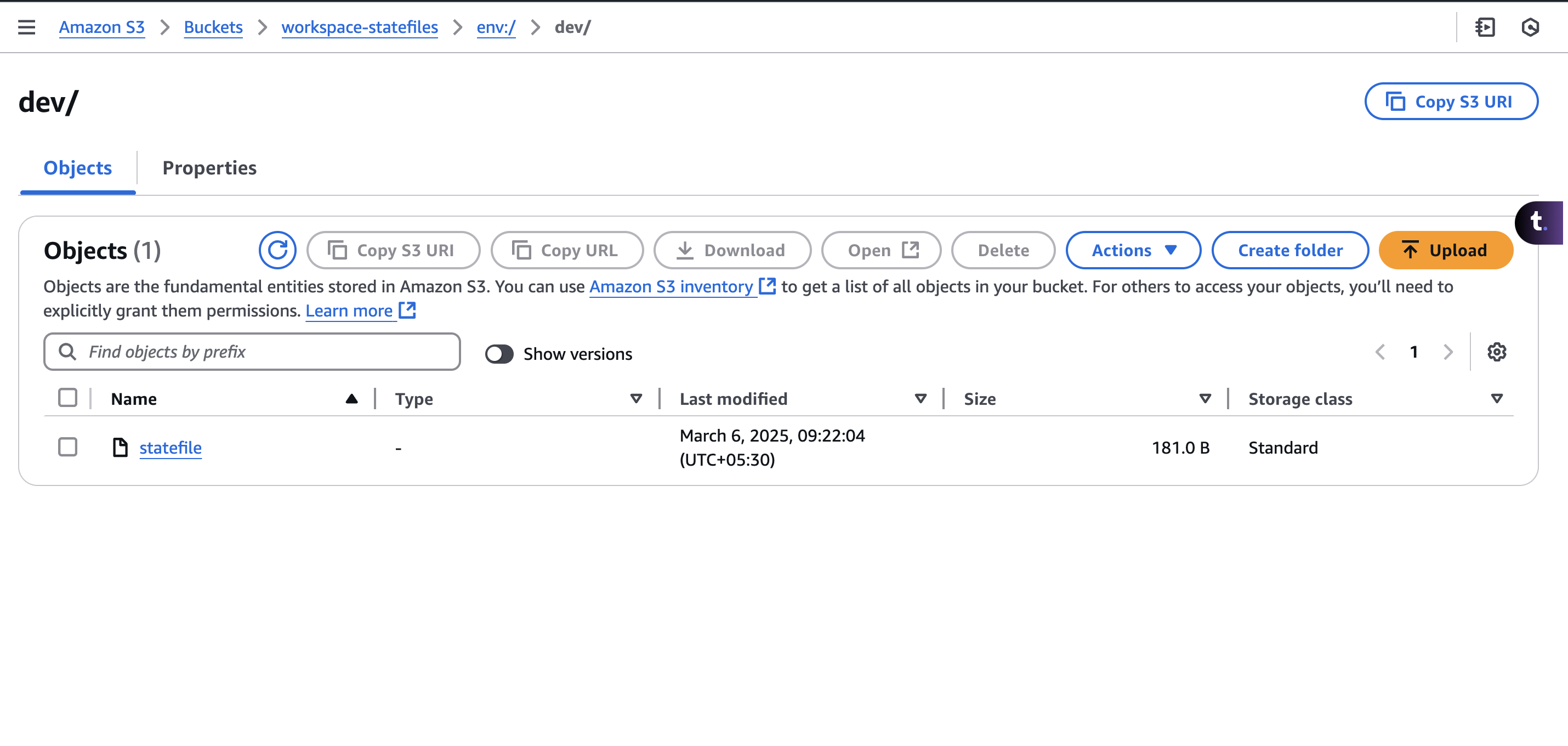
* If we try to run the apply or destroy commands, a new lockfile will be created and other won’t be able to perform until and unless this action is completed.



once the apply or destroy action is completed, the lock file will be removed.  
  


* We can achieve this in s3 as well, but this is supported only from 1.10 version of terraform
  + To enable S3 state locking, use the following optional argument:
  + use\_lockfile - (Optional) Whether to use a lockfile for locking the state file. Defaults to false.

|  |
| --- |
| terraform {  backend "s3" {  bucket = "workspace-statefiles" #bucketname  key = "statefile" #path  region = "ap-south-1"  # dynamodb\_table = "terraform\_state\_lock\_file"  use\_lockfile = true  }  } |

**Complete code:**

|  |
| --- |
| provider "aws" {  region = "ap-south-1"  }  resource "aws\_instance" "instance\_block" {  ami = "ami-00bb6a80f01f03502"  instance\_type = "${local.instance[terraform.workspace]}"  tags = {  Name ="env-name\_instance"  }    }  locals {  instance = {  "default" = "t2.micro"  "dev" = "t2.micro"  "staging" = "t3.small"  "prod" = "t3.medium"  }  }  resource "aws\_s3\_bucket" "terraform\_bucket" {  bucket = "${terraform.workspace}-terraform-test-06032025"  }  terraform {  backend "s3" {  bucket = "workspace-statefiles" #bucketname  key = "statefile" #path  region = "ap-south-1"  # dynamodb\_table = "terraform\_state\_lock\_file"  use\_lockfile = true  }  } |