Assignment 29: Multi-Cloud + Multi-Region Deployment with Terraform (Azure + AWS)

Objective :

Build confidence using the Terraform Registry to author configurations from scratch. You will deploy a minimal, low-cost, multi-cloud, multi-region footprint using only resources you can find and understand from the registry docs.

You'll provision storage foundations in two clouds and two regions each:

- AWS: S3 buckets in two regions
- Azure: Storage accounts (plus resource groups) in two regions

---

Real-World Scenario

Your company needs a resilient, cloud-agnostic "asset landing zone." Product teams will later push build artifacts and static assets to whichever region/cloud is closest. Today's goal is to establish the foundations, not the data flows.

Requirements

- AWS: Create one S3 bucket in us-east-1 and one in eu-central-1.
- Azure: Create one Storage Account in the East US and one in West Europe, each inside its own Resource Group.
- Apply a clear naming convention that embeds the environment and region (e.g., company-dev-assets-use1, company-devassetsweu).
- Tag all resources with at least: project=multicloud-foundation, owner=<your-name>, env=dev.
- Enable basic durability settings you can locate in the registry (e.g., bucket versioning on S3, standard redundancy on Azure).

Note : Keep everything in the free/lowest-cost tier wherever possible.

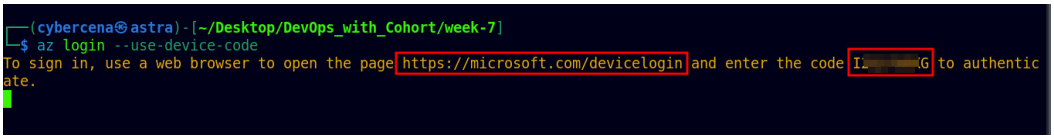
Solutions :

1. Setup Azure CLI with service principle authentication on local machine
2. Setup AWS CLI with Proper permission
3. Create 4 different folders with main.tf files to store terraform script.
4. Write a terraform scripts in main.tf file created for 4 different resources.
5. Deploy the IaaC using terraform .
6. Test if the resource group, S3 Bucket and Storage Account is created or not as per script.

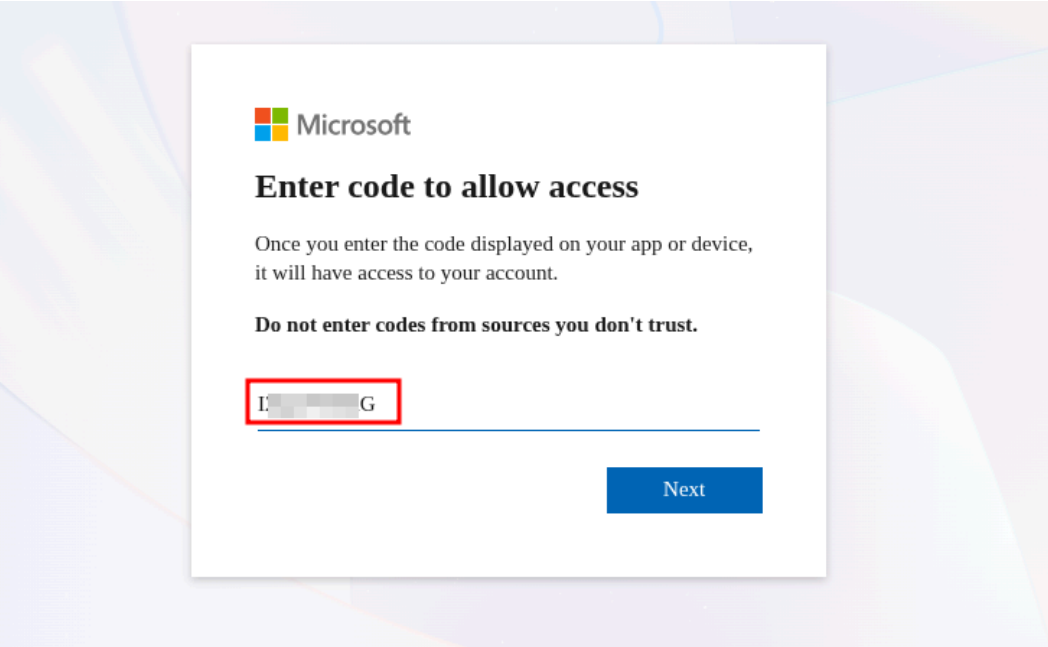
Task 1 : Setup Azure CLI with Proper permission on your local machine .

Step 1 : Setup Azure CLI with Service Principal.

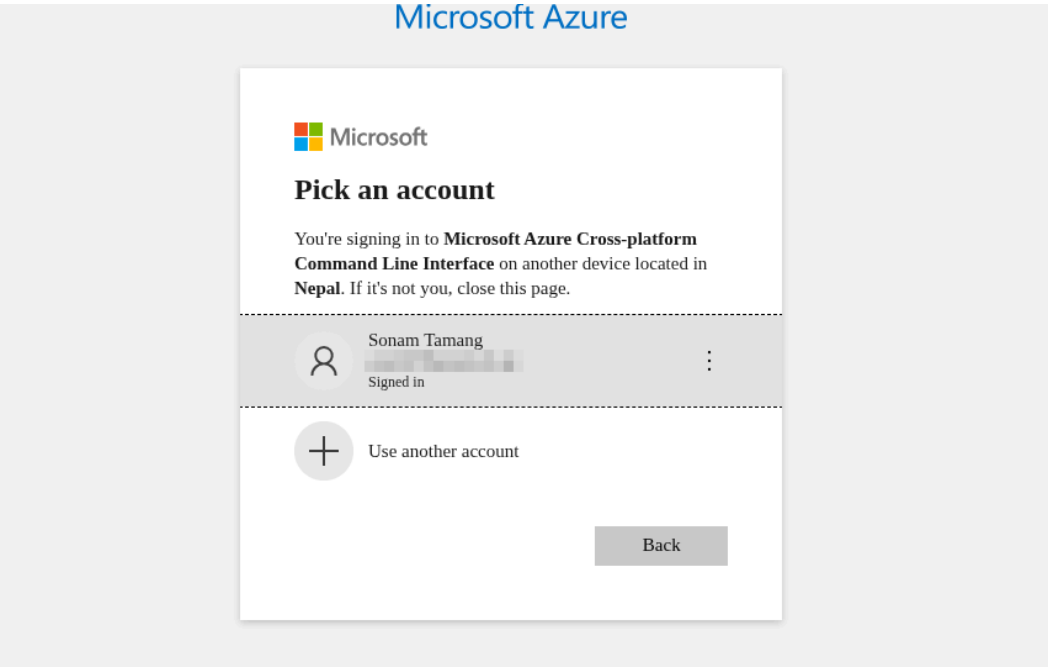
- Open Terminal in MacOS/Linux or Powershell in Windows and enter the command : az login --use-device-code .

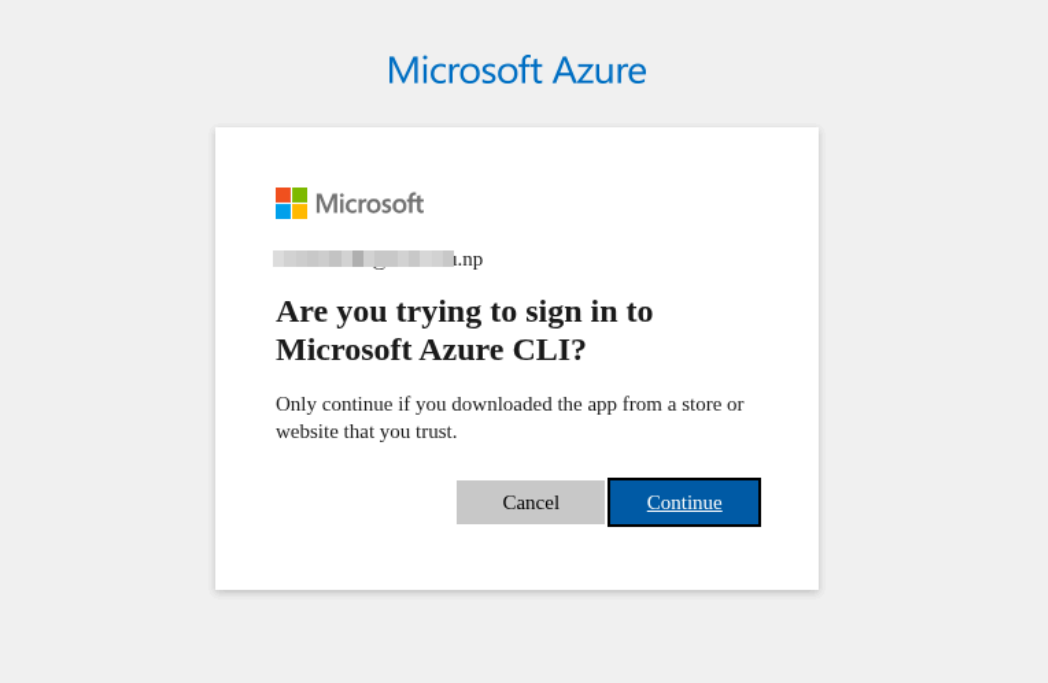


- Copy the Link and Visit your Favourite Browser and Enter the code provided in the terminal.

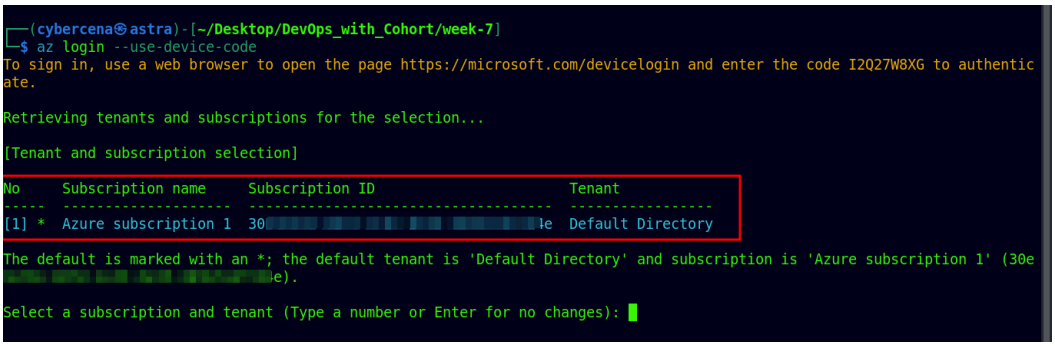


- Select the Azure account you want to use for Project and **confirm** to use Azure CLI.

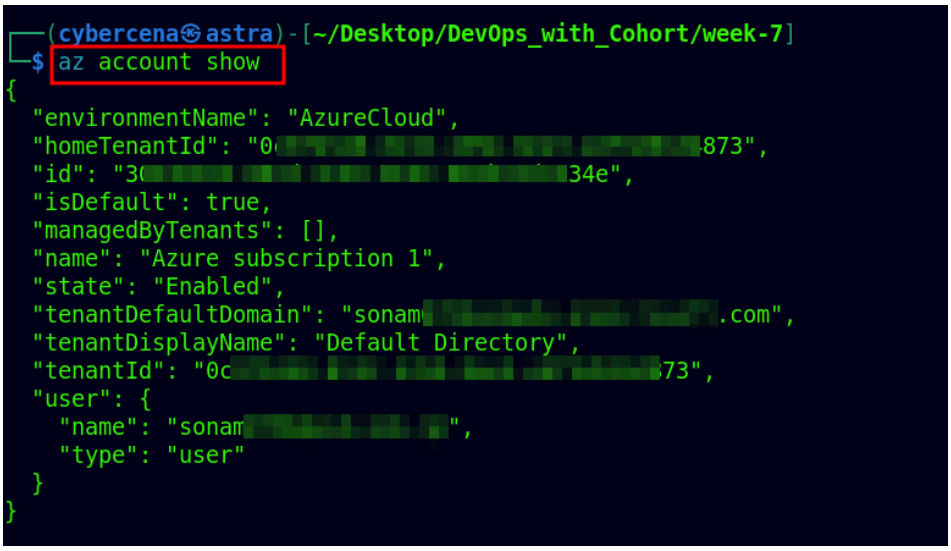




- Check the Terminal, you will see the Subscription details if you are logged in.



- Select the subscription or press ‘Enter’ to select by default.
- Enter the command **az account show** in the terminal to check the account info and copy the **subscription id** .



Step 2 : Create a Service Principal with RBAC.

- Copy the Subscription id and prepare a command to create a service principal with RBAC (Role Based Access Control).

```
SUBSCRIPTION_ID="<your-subscription-id>"

az ad sp create-for-rbac \
  --name "sp-terraform-epicbook" \
  --role "Contributor" \
  --scopes "/subscriptions/$SUBSCRIPTION_ID" \
  --years 1 \
  --query "{appId:appId,password:password,tenant:tenant}" -o json
```

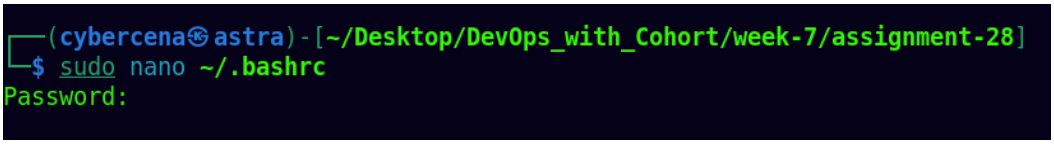
- Using the above command will generate the appId, tenant Id and Password.

Step 3 : Save the credentials as an environment variable.

- If you are using Linux, add the below content with real credentials to **~/.bashrc** file.

  Command : nano ~/.bashrc

```
export ARM_CLIENT_ID="<appId>"
export ARM_CLIENT_SECRET="<password>"
export ARM_TENANT_ID="<tenant>"
export ARM_SUBSCRIPTION_ID="$SUBSCRIPTION_ID"
```



- **Run the command : source ~/.bashrc**

Step 4 : Log out Azure CLI (to prove Terraform uses SP)

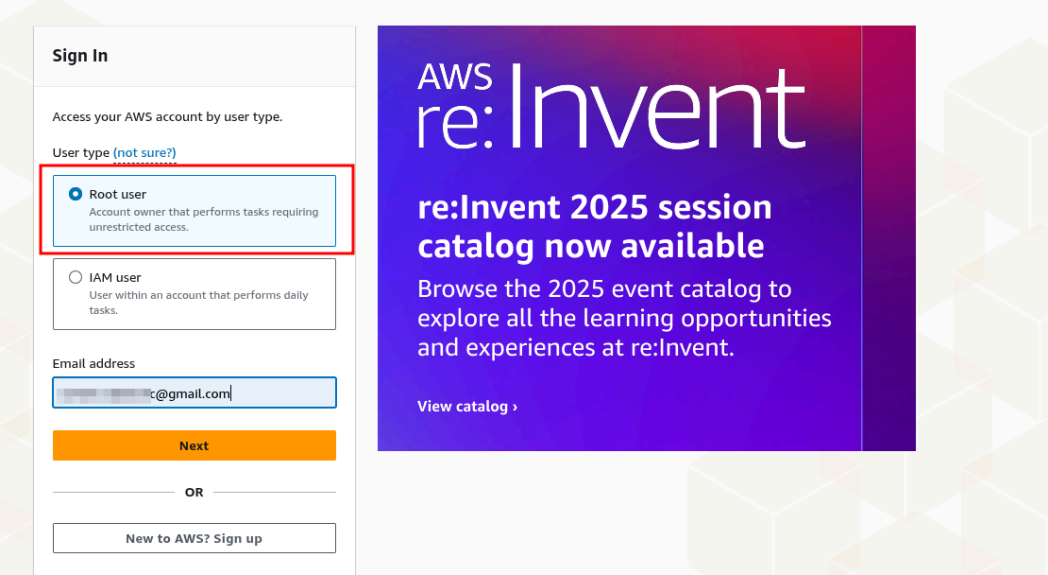
- **Command : az logout**

```
┌──(cybercena㉿astra)-[~/Desktop/DevOps_with_Cohort/week-7/assignment-28]
└─$ az logout
```

Task 2 : Setup AWS CLI with proper IAM Principles.

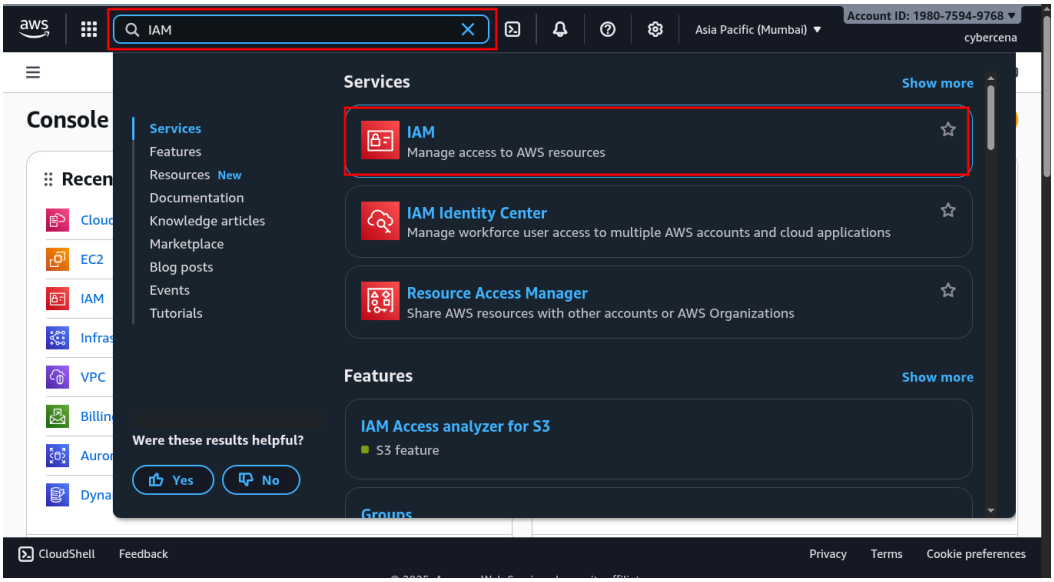
Step 1 : log in to your AWS console using root email

- Visit https://aws.amazon.com/console/ **and log in with your credentials.**

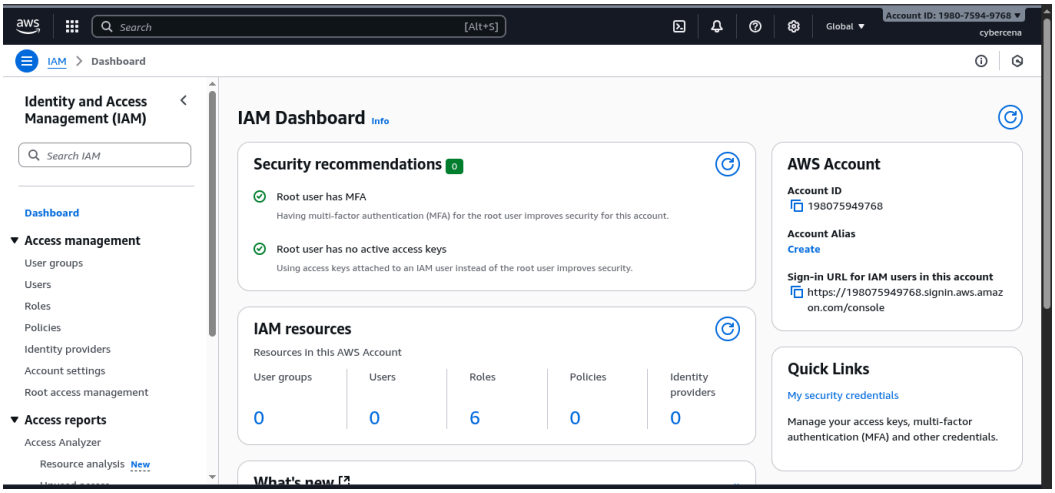


Step 2 : Create an IAM Principle with specific permission.

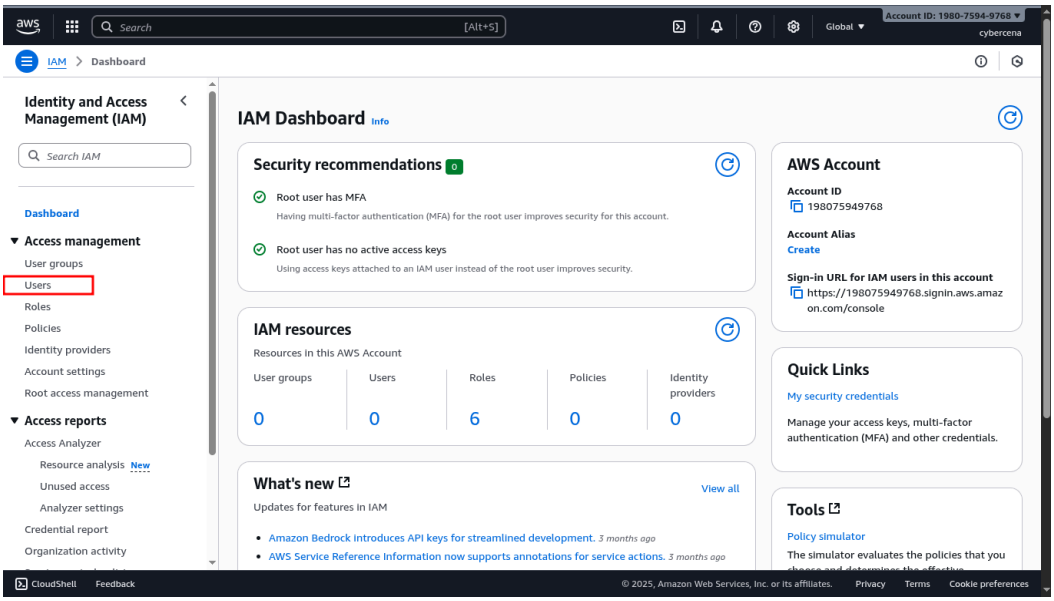
- Navigate to the Search bar of AWS console, and search for the keyword “IAM”.



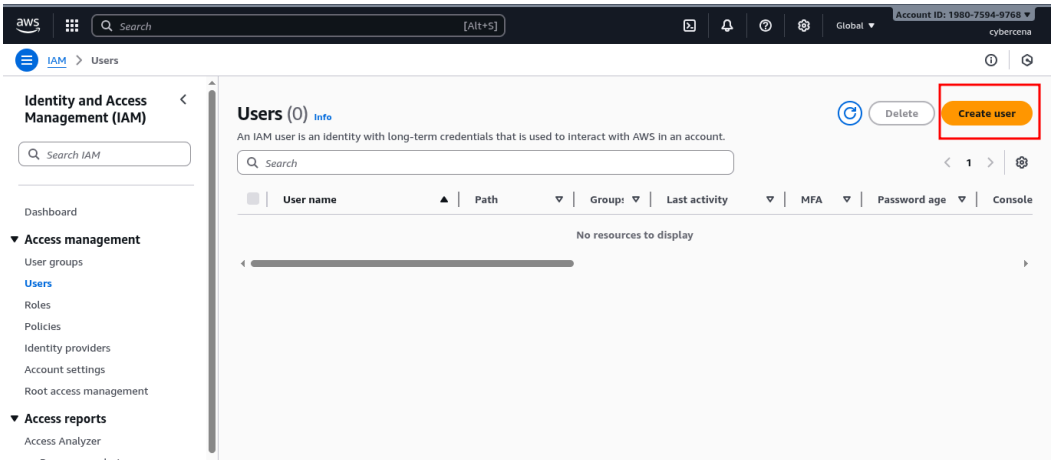
- Select **IAM** and go to IAM Dashboard.



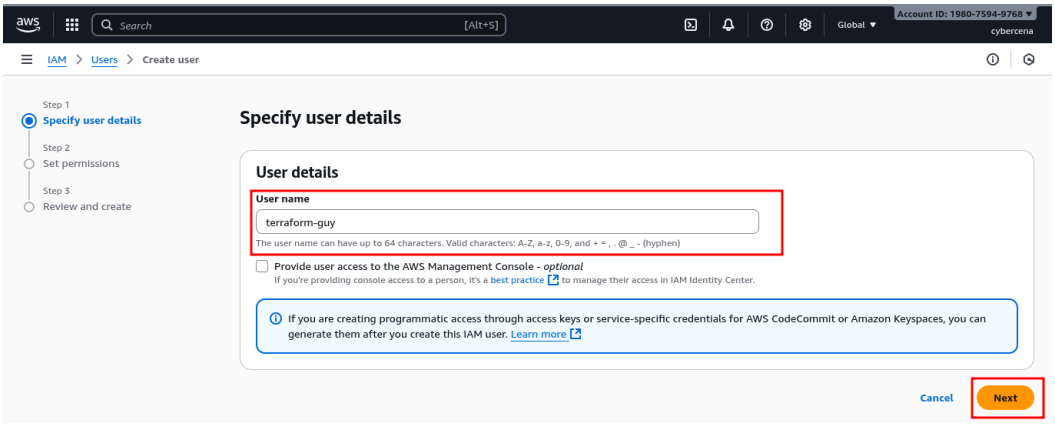
- Click on **users** and Navigate to the User Management Dashboard.



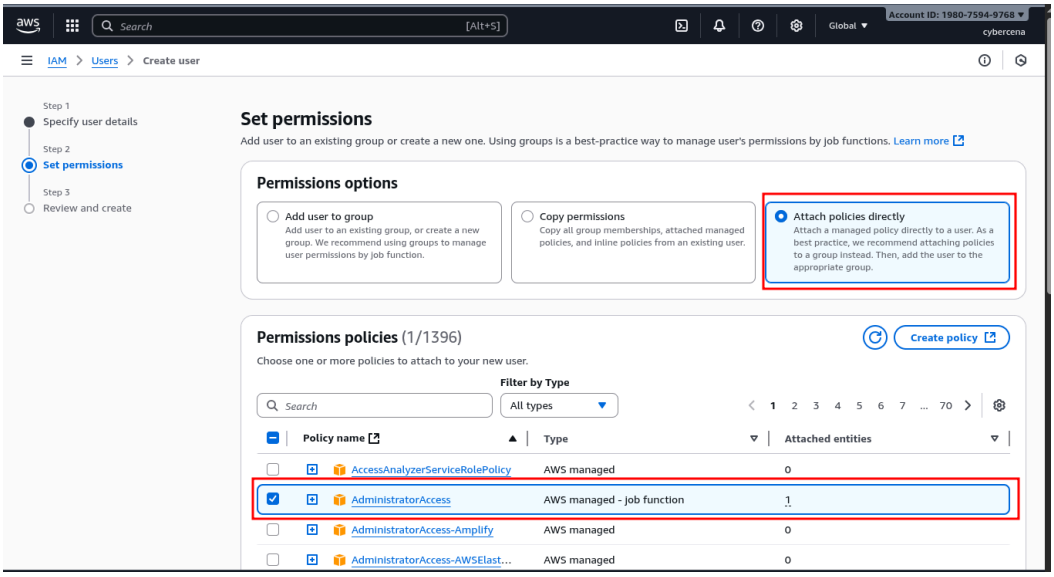
- Click on **Create user** to create a new user.



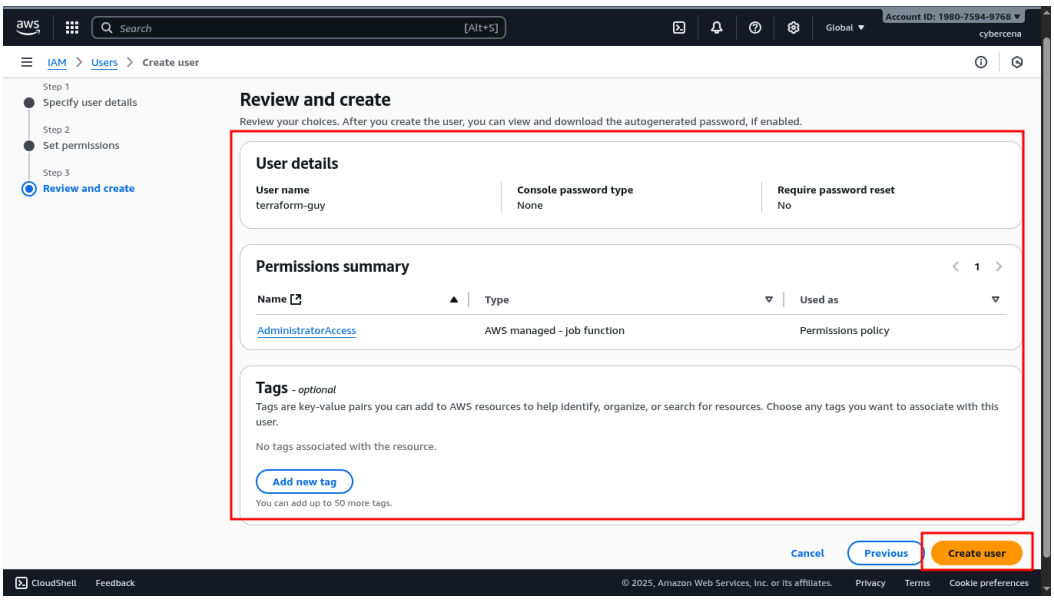
- Give a Suitable name for user :



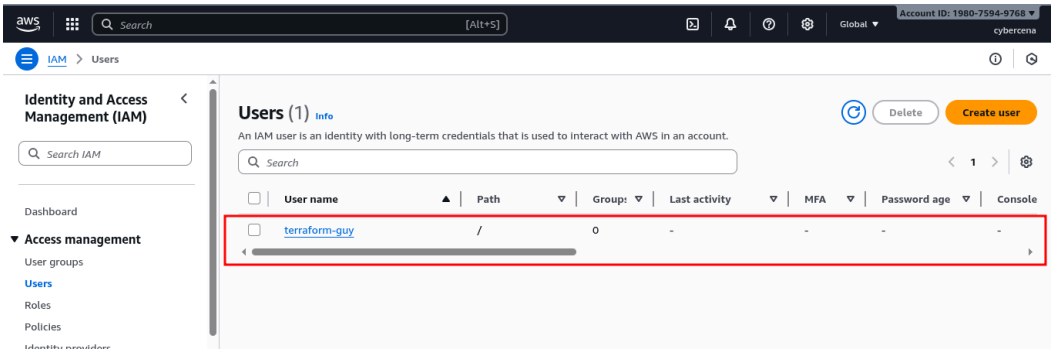
- Click on **Attach Policies directly** and attach suitable policy, i am giving Administration access (usually not recommended).

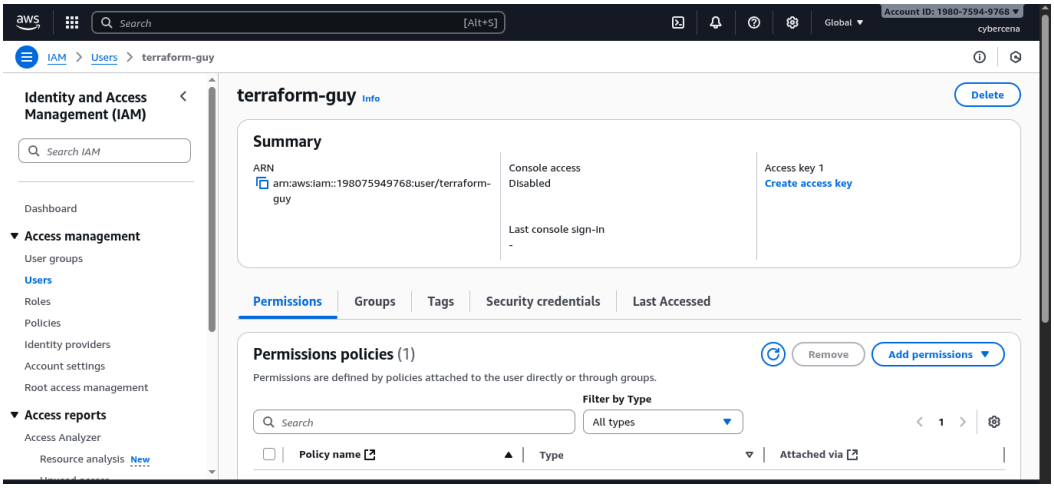


- Review the Specification and permission of IAM user and click on **create user.**

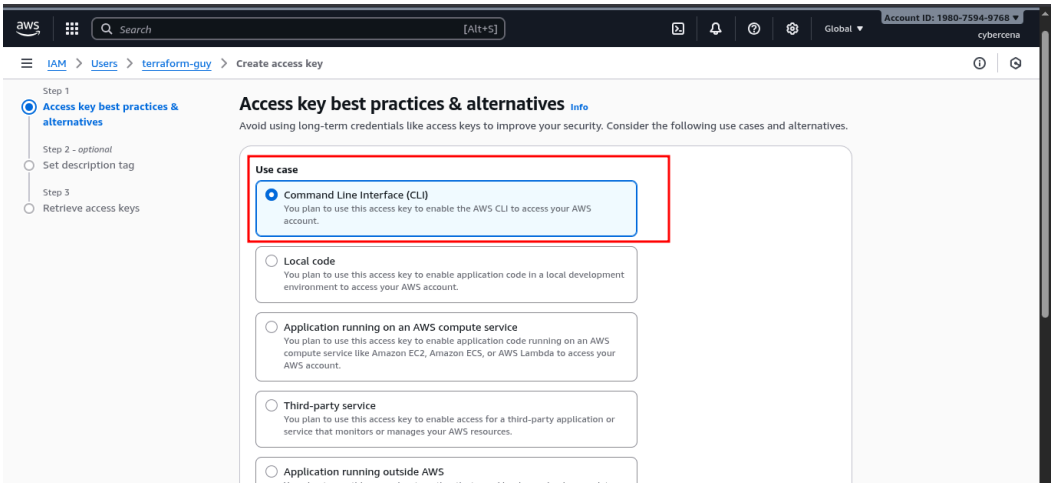


- Click on the **username** and see the user details.

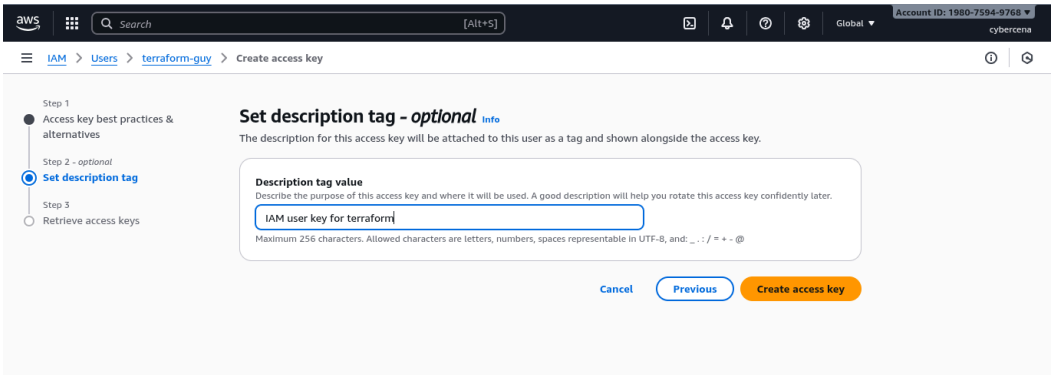




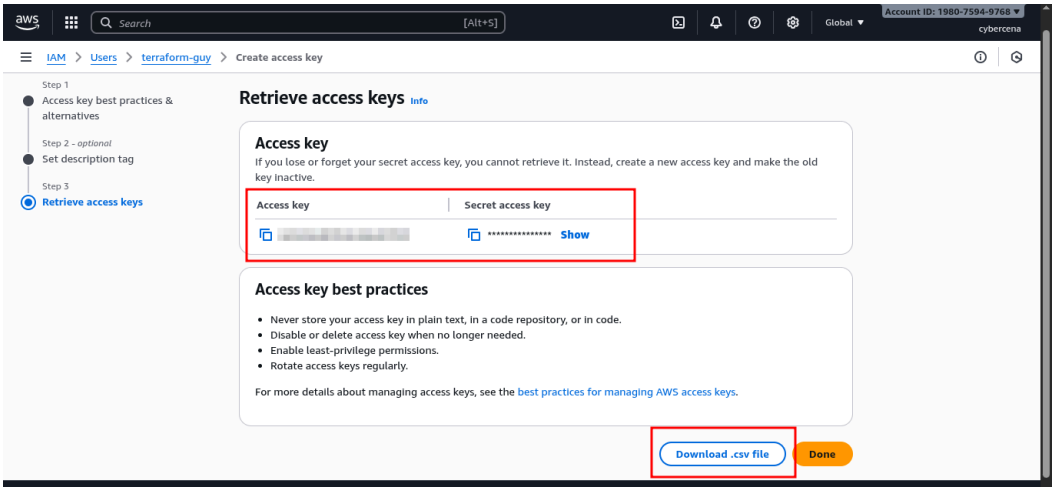
- Click on **create access key** and select **CLI** as a use case.



- Click on **next,** and give a descriptive tag.

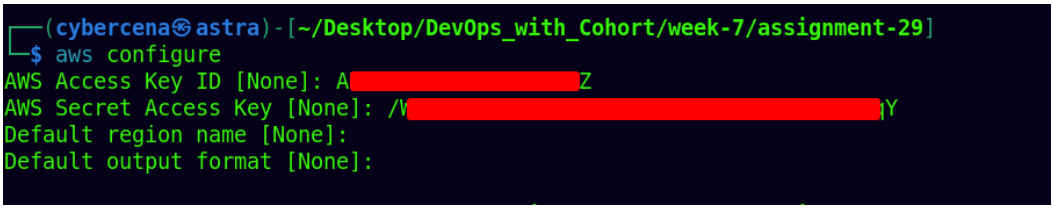


- Copy the access key and secret key or download the **.csv** file (keep the keys safe and secure).



Step 3 : Configure AWS CLI with access key (IAM user) in local machine.

- Go to the terminal and enter the command : aws configure .
- Enter the access key and secret key .



- Check if it is configured successfully or not by using command: aws configure list

```
┌──(cybercena㉿astra)-[~/Desktop/DevOps_with_Cohort/week-7/assignment-29]
└─$ aws configure list

      Name                    Value             Type    Location
      ----                    -----             ----    --------
   profile                <not set>             None    None
access_key     ****************XYMZ shared-credentials-file
secret_key     ****************GFqY shared-credentials-file
    region                <not set>             None    None
```

Task 3 : Create 4 different folders with main.tf files to store terraform script.

Step 1 : Create a plan to choose the descriptive name for folders that store the terraform scripts.

```
13 Name for folders to store terraform scripts
14 aws-s3-us-east-1  ⇒ for S3 bucket in us-east-1 region
15 aws-s3-eu-central-1 ⇒ for S3 bucket in eu-central-1
16 azure-sa-eastus ⇒ for Storage account in east us
17 azure-sa-westeu ⇒ for storage acccount in west europe
18
19
```

Step 2 : Create a Folders with name as per plan :

- Use a single command to create all the folders :

  Command : mkdir aws-s3-us-east-1 aws-s3-eu-central-1 azure-sa-eastus azure-sa-westeu

```
┌──(cybercena㉿astra)-[~/…/DevOps_with_Cohort/week-7/assignment-29/terra-scripts]
└─$ mkdir aws-s3-us-east-1 aws-s3-eu-central-1 azure-sa-eastus azure-sa-westeu
```

- Check if the folders were created or not using command: **ls**

```
┌──(cybercena㉿astra)-[~/…/DevOps_with_Cohort/week-7/assignment-29/terra-scripts]
└─$ ls
aws-s3-eu-central-1  aws-s3-us-east-1  azure-sa-eastus  azure-sa-westeu
```

Task 4 : Write a Script in the main.tf file inside each folder.

- **Script 1 : aws-s3-us-east-1 (AWS S3 bucket in us east region)**

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

```
}

#configure the resource
resource "aws_s3_bucket" "s3-us-east-1" {
 bucket = "sonam-tamang-s3-us-east-1"

 tags = {
   project= "multicloud-foundation"
   Name  = "My bucket"
   Owner = "Sonam Tamang"
   Environment = "dev"
 }
}
```

- **Script 2 : aws-s3-eu-central-1 (AWS S3 bucket in europe central region)**

```
# Configure the AWS Provider
provider "aws" {
 region = "eu-central-1"
}

#configure the resource
resource "aws_s3_bucket" "s3-eu-central-1" {
 bucket = "sonam-tamang-s3-eu-central-1"

 tags = {
   project= "multicloud-foundation"
   Name  = "My bucket"
   Owner = "Sonam Tamang"
   Environment = "dev"
 }
}
```

- **Script 3 : azure-sa-eastus (Storage account for east us region)**

```
#configuring provider
provider "azurerm" {
```

```
 features {}
}

#creating resources
resource "azurerm_resource_group" "rg" {
 name     = "company-dev-assets-use1"
 location = "East US"
}

#defining length for random number
resource "random_id" "suffix" {
 byte_length = 2
}

#creating a storage account
resource "azurerm_storage_account" "example" {
 name                     =
"azsaeastus${random_id.suffix.hex}"
 resource_group_name      = azurerm_resource_group.rg.name
 location                 = azurerm_resource_group.rg.location
 account_tier             = "Standard"
 account_replication_type = "LRS"

 tags = {
   project     = "multicloud-foundation"
   owner       = "Sonam Tamang"
   environment = "dev"
 }
}
```

- **Script 4 : azure-sa-westeu (Storage account for west europe)**

```
#configuring provider
provider "azurerm" {
 features {}
}

#creating resources
resource "azurerm_resource_group" "rg" {
```

```
 name     = "company-devassetsweu"
 location = "West Europe"
}

#defining length for random number
resource "random_id" "suffix" {
 byte_length = 2
}

#creating a storage account
resource "azurerm_storage_account" "example" {
 name                     =
"azsaeastus${random_id.suffix.hex}"
 resource_group_name      = azurerm_resource_group.rg.name
 location                 = azurerm_resource_group.rg.location
 account_tier             = "Standard"
 account_replication_type = "LRS"

 tags = {
   project     = "multicloud-foundation"
   owner       = "Sonam Tamang"
   environment = "dev"
 }
}
```

Task 5 : Deploy the Infrastructure as a Code by using terraform .

- **Run the below commands in each folder :**

```
terraform init
terraform plan
terraform apply -auto-approve
```

- **Screenshots of aws-s3-us-east-1**

```
┌──(cybercena㉿astra)-[~/…/week-7/assignment-29/terra-scripts/aws-s3-us-east-1]
└─$ terraform init

Initializing the backend...

Initializing provider plugins...
- Finding latest version of hashicorp/aws...
- Installing hashicorp/aws v6.16.0...
- Installed hashicorp/aws v6.16.0 (signed by HashiCorp)

Terraform has created a lock file .terraform.lock.hcl to record the provider
selections it made above. Include this file in your version control repository
so that Terraform can guarantee to make the same selections by default when
you run "terraform init" in the future.

Terraform has been successfully initialized!
```

```
┌──(cybercena㉿astra)-[~/…/week-7/assignment-29/terra-scripts/aws-s3-us-east-1]
└─$ terraform plan

Terraform used the selected providers to generate the following execution plan. Resource actions are indicated with the
following symbols:
  + create

Terraform will perform the following actions:

  # aws_s3_bucket.s3-us-east-1 will be created
  + resource "aws_s3_bucket" "s3-us-east-1" {
      + acceleration_status         = (known after apply)
      + acl                         = (known after apply)
      + arn                         = (known after apply)
      + bucket                      = "sonam-tamang-s3-us-east-1"
      + bucket_domain_name          = (known after apply)
      + bucket_prefix               = (known after apply)
      + bucket_region               = (known after apply)
      + bucket_regional_domain_name = (known after apply)
      + force_destroy               = false
      + hosted_zone_id              = (known after apply)
      + id                          = (known after apply)
      + object_lock_enabled         = (known after apply)
      + policy                      = (known after apply)
      + region                      = "us-east-1"
      + request_payer               = (known after apply)
      + tags                        = {
          + "Environment" = "dev"
          + "Name"        = "My bucket"
```

```
      + tags                        = {
          + "Environment" = "dev"
          + "Name"        = "My bucket"
          + "Owner"       = "Sonam Tamang"
          + "project"     = "multicloud-foundation"
        }
      + tags_all                    = {
          + "Environment" = "dev"
          + "Name"        = "My bucket"
          + "Owner"       = "Sonam Tamang"
          + "project"     = "multicloud-foundation"
        }
      + website_domain              = (known after apply)
      + website_endpoint            = (known after apply)
    }

Plan: 1 to add, 0 to change, 0 to destroy.
aws_s3_bucket.s3-us-east-1: Creating...
aws_s3_bucket.s3-us-east-1: Creation complete after 9s [id=sonam-tamang-s3-us-east-1]

Apply complete! Resources: 1 added, 0 changed, 0 destroyed.
```

- **Screenshots of aws-s3-eu-central-1**

```
┌──(cybercena㉿astra)-[~/…/week-7/assignment-29/terra-scripts/aws-s3-eu-central-1]
└─$ ls
main.tf

┌──(cybercena㉿astra)-[~/…/week-7/assignment-29/terra-scripts/aws-s3-eu-central-1]
└─$ terraform init

Initializing the backend...

Initializing provider plugins...
- Finding latest version of hashicorp/aws...
- Installing hashicorp/aws v6.16.0...
- Installed hashicorp/aws v6.16.0 (signed by HashiCorp)

Terraform has created a lock file .terraform.lock.hcl to record the provider
selections it made above. Include this file in your version control repository
so that Terraform can guarantee to make the same selections by default when
you run "terraform init" in the future.

Terraform has been successfully initialized!
```

```
┌──(cybercena㉿astra)-[~/…/week-7/assignment-29/terra-scripts/aws-s3-eu-central-1]
└─$ terraform plan

Terraform used the selected providers to generate the following execution plan. Resource actions are indicated with the
following symbols:
  + create

Terraform will perform the following actions:

  # aws_s3_bucket.s3-eu-central-1 will be created
  + resource "aws_s3_bucket" "s3-eu-central-1" {
      + acceleration_status         = (known after apply)
      + acl                         = (known after apply)
      + arn                         = (known after apply)
      + bucket                      = "sonam-tamang-s3-eu-central-1"
      + bucket_domain_name          = (known after apply)
      + bucket_prefix               = (known after apply)
      + bucket_region               = (known after apply)
      + bucket_regional_domain_name = (known after apply)
      + force_destroy               = false
      + hosted_zone_id              = (known after apply)
      + id                          = (known after apply)
      + object_lock_enabled         = (known after apply)
      + policy                      = (known after apply)
      + region                      = "eu-central-1"
      + request_payer               = (known after apply)
      + tags                        = {
          + "Environment" = "dev"
          + "Name"        = "My bucket"
```

```
      + request_payer               = (known after apply)
      + tags                        = {
          + "Environment" = "dev"
          + "Name"        = "My bucket"
          + "Owner"       = "Sonam Tamang"
          + "project"     = "multicloud-foundation"
        }
      + tags_all                    = {
          + "Environment" = "dev"
          + "Name"        = "My bucket"
          + "Owner"       = "Sonam Tamang"
          + "project"     = "multicloud-foundation"
        }
      + website_domain              = (known after apply)
      + website_endpoint            = (known after apply)
    }

Plan: 1 to add, 0 to change, 0 to destroy.
aws_s3_bucket.s3-eu-central-1: Creating...
aws_s3_bucket.s3-eu-central-1: Creation complete after 8s [id=sonam-tamang-s3-eu-central-1]

Apply complete! Resources: 1 added, 0 changed, 0 destroyed.
```

- **Screenshots of azure-sa-eastus**

```
┌──(cybercena㉿astra)-[~/…/week-7/assignment-29/terra-scripts/azure-sa-eastus]
└─$ ls
main.tf

┌──(cybercena㉿astra)-[~/…/week-7/assignment-29/terra-scripts/azure-sa-eastus]
└─$ terraform init

Initializing the backend...

Initializing provider plugins...
- Finding latest version of hashicorp/azurerm...
- Installing hashicorp/azurerm v4.48.0...
- Installed hashicorp/azurerm v4.48.0 (signed by HashiCorp)

Terraform has created a lock file .terraform.lock.hcl to record the provider
selections it made above. Include this file in your version control repository
so that Terraform can guarantee to make the same selections by default when
you run "terraform init" in the future.

Terraform has been successfully initialized!
```

```
┌──(cybercena㉿astra)-[~/…/week-7/assignment-29/terra-scripts/azure-sa-eastus]
└─$ terraform plan

Terraform used the selected providers to generate the following execution plan. Resource actions are indicated with the
following symbols:
  + create

Terraform will perform the following actions:

  # azurerm_resource_group.rg will be created
  + resource "azurerm_resource_group" "rg" {
      + id       = (known after apply)
      + location = "eastus"
      + name     = "company-dev-assets-use1"
    }

  # azurerm_storage_account.example will be created
  + resource "azurerm_storage_account" "example" {
      + access_tier                     = (known after apply)
      + account_kind                    = "StorageV2"
      + account_replication_type        = "LRS"
      + account_tier                    = "Standard"
      + allow_nested_items_to_be_public = true
      + cross_tenant_replication_enabled = false
      + default_to_oauth_authentication = false
      + dns_endpoint_type               = "Standard"
      + https_traffic_only_enabled      = true
```

```
┌──(cybercena㉿astra)-[~/…/week-7/assignment-29/terra-scripts/azure-sa-eastus]
└─$ terraform apply -auto-approve

Terraform used the selected providers to generate the following execution plan. Resource actions are indicated with the
following symbols:
  + create

Terraform will perform the following actions:

  # azurerm_resource_group.rg will be created
  + resource "azurerm_resource_group" "rg" {
      + id       = (known after apply)
      + location = "eastus"
      + name     = "company-dev-assets-use1"
    }

  # azurerm_storage_account.example will be created
  + resource "azurerm_storage_account" "example" {
      + access_tier                     = (known after apply)
      + account_kind                    = "StorageV2"
      + account_replication_type        = "LRS"
      + account_tier                    = "Standard"
      + allow_nested_items_to_be_public = true
      + cross_tenant_replication_enabled = false
      + default_to_oauth_authentication = false
      + dns_endpoint_type               = "Standard"
      + https_traffic_only_enabled      = true
```

```
Plan: 3 to add, 0 to change, 0 to destroy.
random_id.suffix: Creating...
random_id.suffix: Creation complete after 0s [id=Edc]
azurerm_resource_group.rg: Creating...
azurerm_resource_group.rg: Still creating... [10s elapsed]
azurerm_resource_group.rg: Creation complete after 18s [id=/subscriptions/[redacted]/resourceGro
ups/company-dev-assets-use1]
azurerm_storage_account.example: Creating...
azurerm_storage_account.example: Still creating... [10s elapsed]
azurerm_storage_account.example: Still creating... [20s elapsed]
azurerm_storage_account.example: Still creating... [30s elapsed]
azurerm_storage_account.example: Still creating... [40s elapsed]
azurerm_storage_account.example: Still creating... [50s elapsed]
azurerm_storage_account.example: Still creating... [1m0s elapsed]
azurerm_storage_account.example: Still creating... [1m10s elapsed]
azurerm_storage_account.example: Still creating... [1m20s elapsed]
azurerm_storage_account.example: Creation complete after 1m22s [id=/subscriptions/[redacted]res
ourceGroups/company-dev-assets-use1/providers/Microsoft.Storage/storageAccounts/azsaeastus11d7]

Apply complete! Resources: 3 added, 0 changed, 0 destroyed.
```

- **Screenshots of azure-sa-westeu**

```
┌──(cybercena㉿astra)-[~/…/week-7/assignment-29/terra-scripts/azure-sa-westeu]
└─$ ls
main.tf

┌──(cybercena㉿astra)-[~/…/week-7/assignment-29/terra-scripts/azure-sa-westeu]
└─$ terraform init

Initializing the backend...

Initializing provider plugins...
- Finding latest version of hashicorp/azurerm...
- Finding latest version of hashicorp/random...
- Installing hashicorp/azurerm v4.48.0...
- Installed hashicorp/azurerm v4.48.0 (signed by HashiCorp)
- Installing hashicorp/random v3.7.2...
- Installed hashicorp/random v3.7.2 (signed by HashiCorp)

Terraform has created a lock file .terraform.lock.hcl to record the provider
selections it made above. Include this file in your version control repository
so that Terraform can guarantee to make the same selections by default when
you run "terraform init" in the future.

Terraform has been successfully initialized!
```

```
┌──(cybercena㉿astra)-[~/…/week-7/assignment-29/terra-scripts/azure-sa-westeu]
└─$ terraform plan

Terraform used the selected providers to generate the following execution plan. Resource actions are indicated with the
following symbols:
  + create

Terraform will perform the following actions:

  # azurerm_resource_group.rg will be created
  + resource "azurerm_resource_group" "rg" {
      + id       = (known after apply)
      + location = "westeurope"
      + name     = "company-devassetsweu"
    }

  # azurerm_storage_account.example will be created
  + resource "azurerm_storage_account" "example" {
      + access_tier                     = (known after apply)
      + account_kind                    = "StorageV2"
      + account_replication_type        = "LRS"
      + account_tier                    = "Standard"
      + allow_nested_items_to_be_public = true
      + cross_tenant_replication_enabled = false
      + default_to_oauth_authentication = false
      + dns_endpoint_type               = "Standard"
      + https_traffic_only_enabled      = true
```

```
┌──(cybercena㉿astra)-[~/…/week-7/assignment-29/terra-scripts/azure-sa-westeu]
└─$ terraform apply -auto-approve

Terraform used the selected providers to generate the following execution plan. Resource actions are indicated with the
following symbols:
  + create

Terraform will perform the following actions:

  # azurerm_resource_group.rg will be created
  + resource "azurerm_resource_group" "rg" {
      + id       = (known after apply)
      + location = "westeurope"
      + name     = "company-devassetsweu"
    }
```

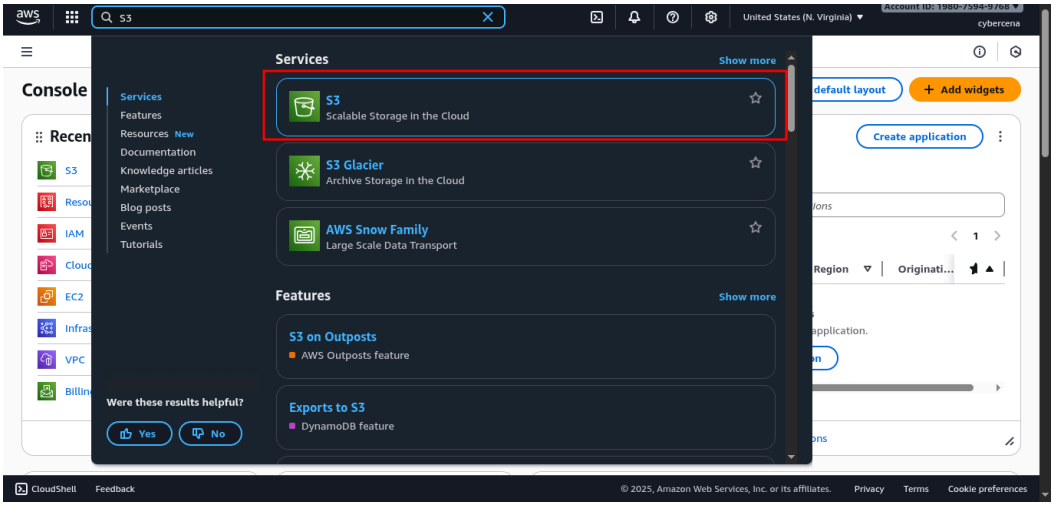
```
Plan: 3 to add, 0 to change, 0 to destroy.
random_id.suffix: Creating...
random_id.suffix: Creation complete after 0s [id=h24]
azurerm_resource_group.rg: Creating...
azurerm_resource_group.rg: Still creating... [10s elapsed]
azurerm_resource_group.rg: Creation complete after 19s [id=/subscriptions/[redacted]/resourceGro
ups/company-devassetsweu]
azurerm_storage_account.example: Creating...
azurerm_storage_account.example: Still creating... [10s elapsed]
azurerm_storage_account.example: Still creating... [20s elapsed]
azurerm_storage_account.example: Still creating... [30s elapsed]
azurerm_storage_account.example: Still creating... [40s elapsed]
azurerm_storage_account.example: Still creating... [50s elapsed]
azurerm_storage_account.example: Still creating... [1m0s elapsed]
azurerm_storage_account.example: Still creating... [1m10s elapsed]
azurerm_storage_account.example: Still creating... [1m20s elapsed]
azurerm_storage_account.example: Creation complete after 1m28s [id=/subscriptions/[redacted]/res
ourceGroups/company-devassetsweu/providers/Microsoft.Storage/storageAccounts/azsaeastus8/be]

Apply complete! Resources: 3 added, 0 changed, 0 destroyed.
```

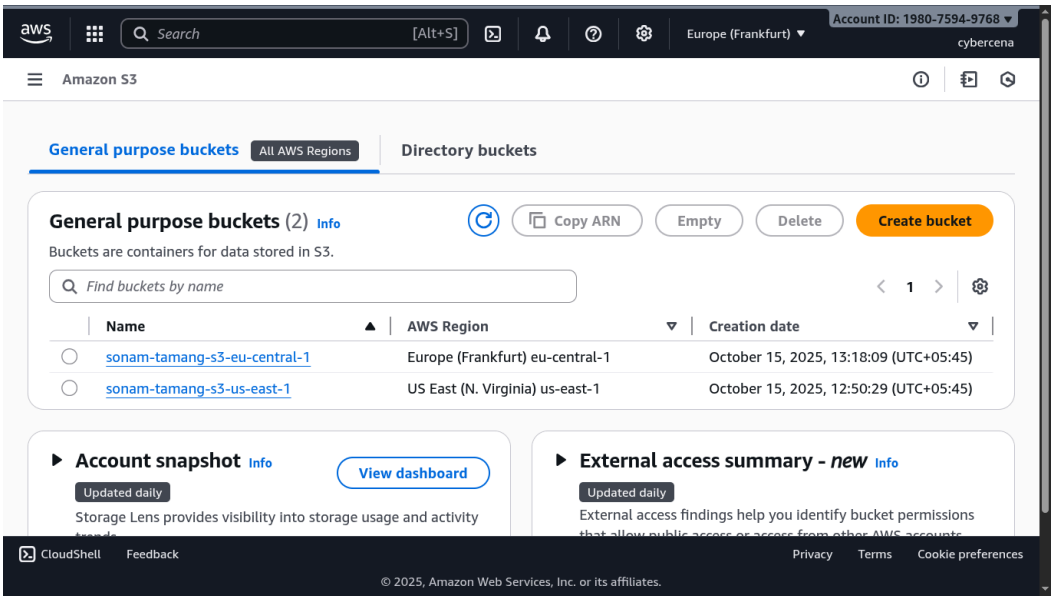
Task 6 : Check if the S3 buckets and storage accounts are created or not in AWS Management Console and Azure Portal respectively .

For AWS :

- Search “S3” in the search bar and select S3 .

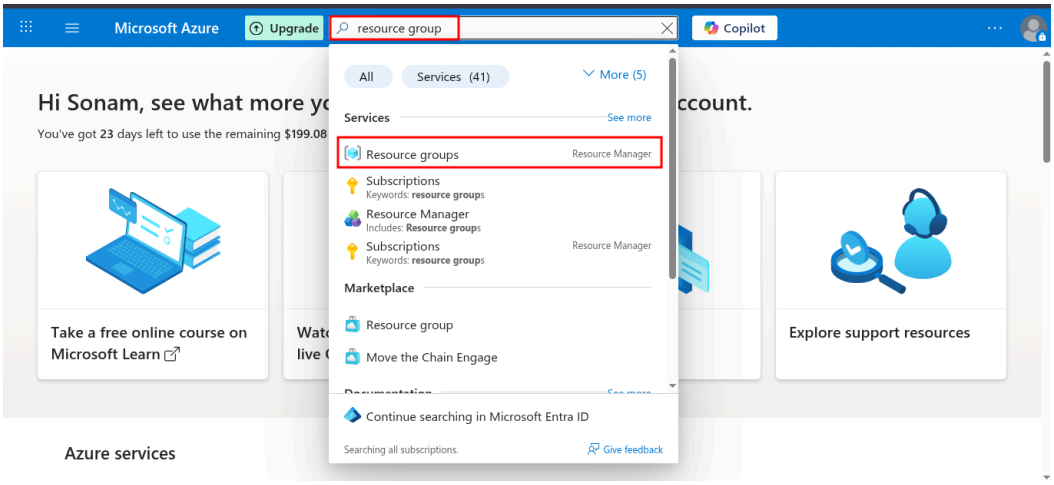


- Proof of AWS S3 Bucket created in two different regions.

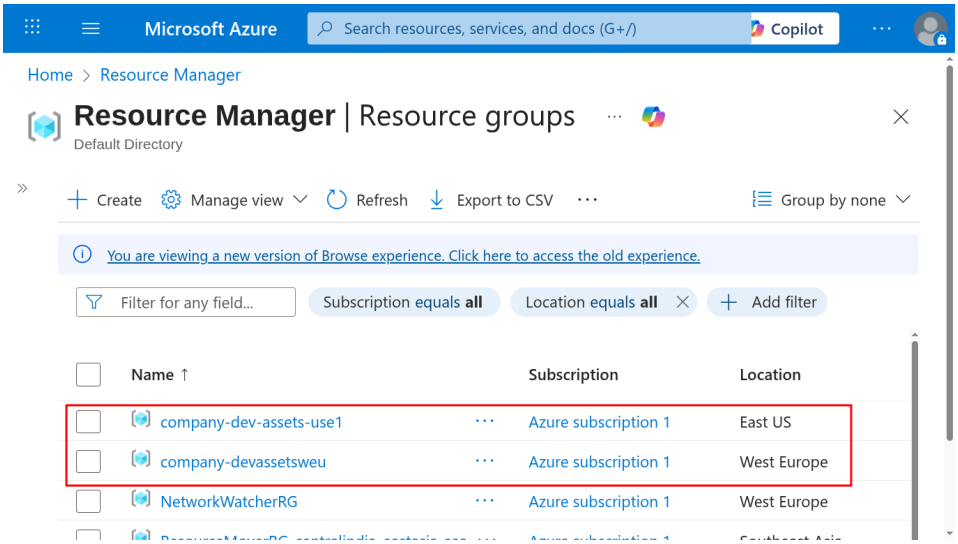


For Azure :

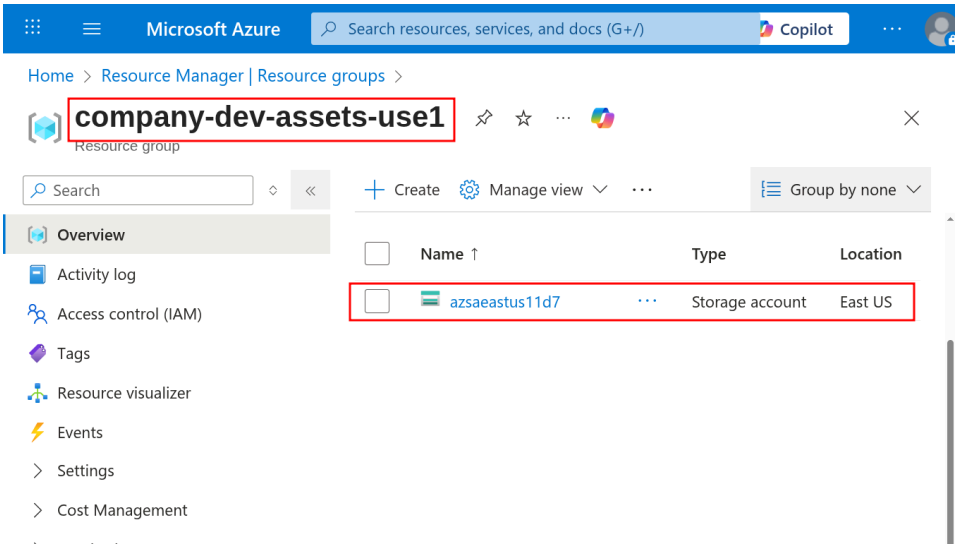
- **Log in to Azure Portal**
- **Search for "Resource Group", and select Resource Group**



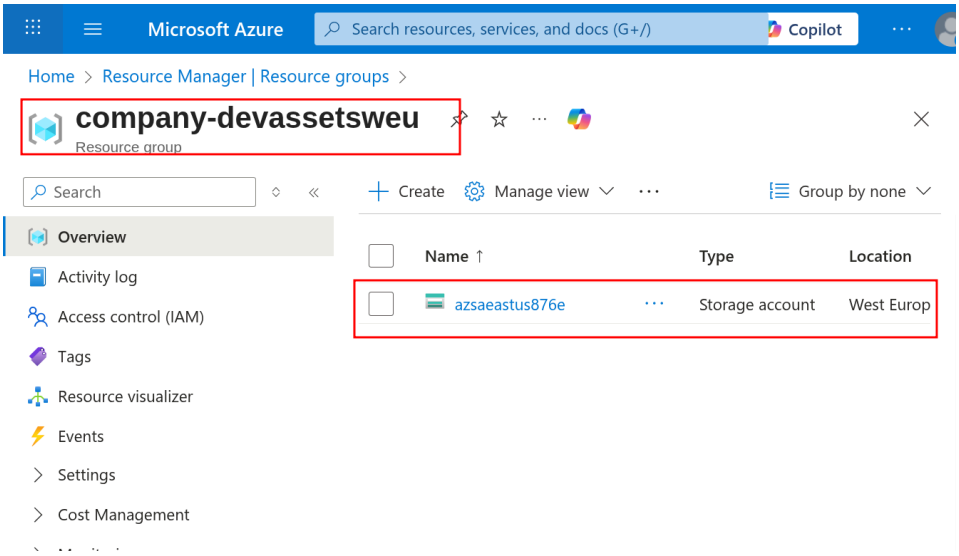
- **Check if the Resource Group is created or not, if created check for Storage account.**
    - Proof of Resource group creation



- Storage account created in the East US.



- Storage account created in West Europe.



Lesson Learned :

During this assignment, I learned how to use Terraform to deploy infrastructure across multiple clouds and regions. I gained hands-on experience with AWS S3 and Azure Storage Accounts, including setting tags, versioning, and replication for durability. I understood the importance of global uniqueness for S3 buckets and Azure storage account names, and how to use random_id to avoid conflicts. I also practiced defining providers per region, managing resource groups in Azure, and ensuring idempotency by running a plan and applying multiple times. Overall, this exercise improved my skills in multi-cloud provisioning, Terraform syntax, and best practices for reusable, low-cost infrastructure.

References :

1. https://registry.terraform.io/providers/hashicorp/aws/latest
2. https://registry.terraform.io/providers/hashicorp/azurerm/latest/docs

3. https://github.com/cybercena/DevOps-Micro-Internship-DMI-/tree/main/week-7/assignment-29