Name: Zuha Irfan

Reg no: 2023-BSE-073

Course: Cloud Computing Lab

Section: V-B

LAB 13

Task 0 Lab Setup (Codespace & GH CLI)

All actions below should be executed inside the Codespace shell.

Create Codespace & connect:

Task 1 — Create IAM Group and Output Details

In this task, you will create an IAM group named "developers" and output its details.

Create the initial project structure:

mkdir -p ~/Lab13

cd ~/Lab13

Create the main Terraform file:

touch main.tf

Create main.tf with AWS provider configuration:

```
@Zuha-Irfan ➜ /workspaces/CC-ZuhaIrfan-073/Lab13 (main) $vim main.tf
@Zuha-Irfan ➜ /workspaces/CC-ZuhaIrfan-073/Lab13 (main) $cat main.tf
provider "aws" {
  shared_config_files      = ["~/.aws/config"]
  shared_credentials_files = ["~/.aws/credentials"]
}

resource "aws_iam_group" "developers" {
  name = "developers"
  path = "/groups/"
}

output "group_details" {
  value = {
    group_name = aws_iam_group.developers.name
    group_arn  = aws_iam_group.developers.arn
    unique_id  = aws_iam_group.developers.unique_id
  }
}
```

Initialize Terraform:

terraform init

```
@Zuha-Irfan ➜ /workspaces/CC-ZuhaIrfan-073/Lab13 (main) $ terraform init
Initializing the backend...
Initializing provider plugins...
- Finding latest version of hashicorp/aws...
- Installing hashicorp/aws v6.27.0...
- Installed hashicorp/aws v6.27.0 (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 cont
rol repository
so that Terraform can guarantee to make the same selections by d
efault when
you run "terraform init" in the future.

Terraform has been successfully initialized!

You may now begin working with Terraform. Try running "terraform
 plan" to see
any changes that are required for your infrastructure. All Terra
form commands
should now work.

If you ever set or change modules or backend configuration for T
erraform,
rerun this command to reinitialize your working directory. If yo
u forget, other
commands will detect it and remind you to do so if necessary.
@Zuha-Irfan ➜ /workspaces/CC-ZuhaIrfan-073/Lab13 (main) $
```

Apply the configuration:

terraform apply -auto-approve

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

Outputs:

group_details = {
  "group_arn" = "arn:aws:iam::538079272540:group/groups/developers"
  "group_name" = "developers"
  "unique_id" = "AGPAX2SAGDZODXXWVG57E"
}
```

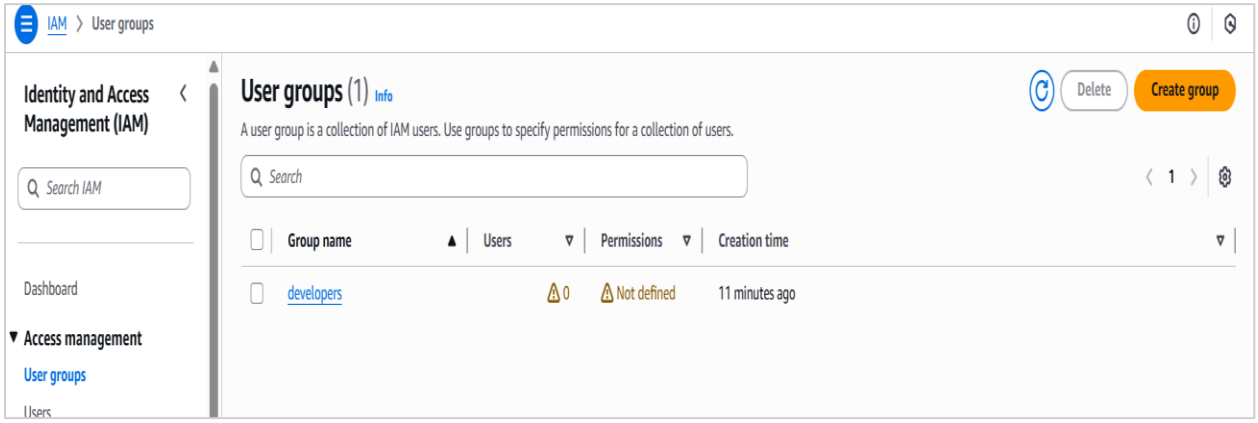
Display the output:

terraform output

```
@Zuha-Irfan →/workspaces/CC-ZuhaIrfan-073/Lab13 (main) $ terraform output
group_details = {
  "group_arn" = "arn:aws:iam::538079272540:group/groups/developers"
  "group_name" = "developers"
  "unique_id" = "AGPAX2SAGDZODXXWVG57E"
}
```

Verify the group in AWS Console:

Navigate to IAM → Groups in AWS Console



Task 2 — Create IAM User with Group Membership

In this task, you will create an IAM user named "loadbalancer" and add it to the developers group.

Update main.tf to add the IAM user resource:

```
@Zuha-Irfan → /workspaces/CC-ZuhaIrfan-073/Lab13 (main) $ vim main.tf
@Zuha-Irfan → /workspaces/CC-ZuhaIrfan-073/Lab13 (main) $ cat main.tf
provider "aws" {
  shared_config_files      = ["~/.aws/config"]
  shared_credentials_files = ["~/.aws/credentials"]
}

resource "aws_iam_group" "developers" {
  name = "developers"
  path = "/groups/"
}

output "group_details" {
  value = {
    group_name = aws_iam_group.developers. name
    group_arn  = aws_iam_group. developers.arn
    unique_id  = aws_iam_group.developers.unique_id
  }
}

resource "aws_iam_user" "lb" {
  name = "loadbalancer"
  path = "/users/"
  force_destroy = true
  tags = {
    DisplayName = "Load Balancer"
  }
}

resource "aws_iam_user_group_membership" "lb_membership" {
  user = aws_iam_user.lb.name
  groups = [
    aws_iam_group. developers.name
  ]
}

output "user_details" {
  value = {
    user_name = aws_iam_user.lb.name
    user_arn  = aws_iam_user.lb.arn
    unique_id = aws_iam_user.lb.unique_id
  }
}
```

Apply the configuration:

terraform apply -auto-approve

```
Apply complete! Resources: 2 added, 0 changed, 0 destroyed.

Outputs:

group_details = {
  "group_arn" = "arn:aws:iam::538079272540:group/groups/developers"
  "group_name" = "developers"
  "unique_id" = "AGPAX2SAGDZODXXWVG57E"
}
user_details = {
  "unique_id" = "AIDAX2SAGDZOH3YAHYCMA"
  "user_arn" = "arn:aws:iam::538079272540:user/users/loadbalancer"
  "user_name" = "loadbalancer"
}
@Zuha-Irfan → /workspaces/CC-ZuhaIrfan-073/Lab13 (main) $
```

Display the outputs:

terraform output

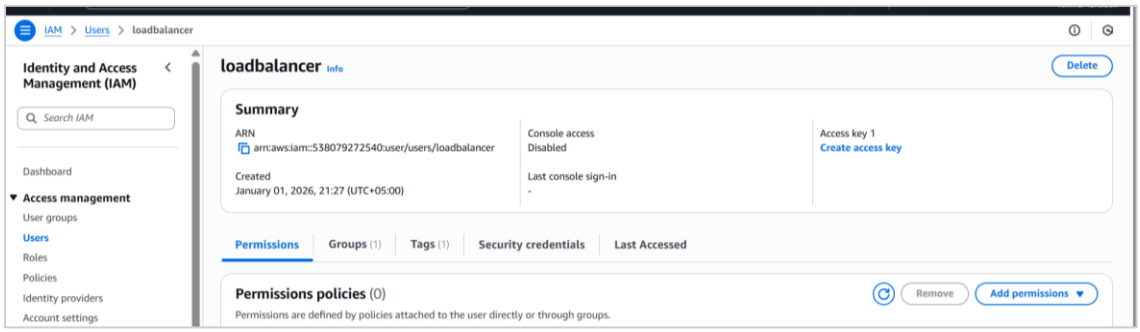
```
@Zuha-Irfan → /workspaces/CC-ZuhaIrfan-073/Lab13 (main) $ terraform output
group_details = {
  "group_arn" = "arn:aws:iam::538079272540:group/groups/developers"
  "group_name" = "developers"
  "unique_id" = "AGPAX2SAGDZODXXWVG57E"
}
user_details = {
  "unique_id" = "AIDAX2SAGDZOH3YAHYCMA"
  "user_arn" = "arn:aws:iam::538079272540:user/users/loadbalancer"
  "user_name" = "loadbalancer"
}
```

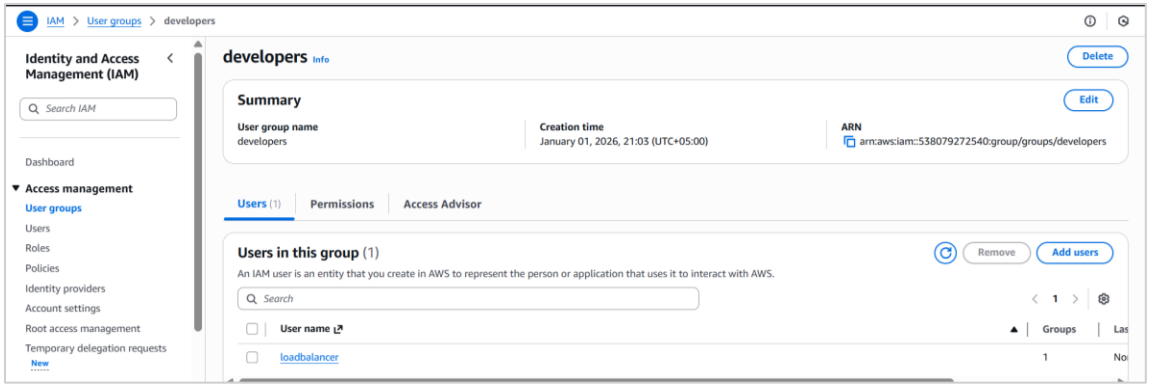
Verify the user in AWS Console:

Navigate to IAM → Users in AWS Console

Click on "loadbalancer" user

Check the "Groups" tab





Task 3 — Attach Policies to IAM Group

In this task, you will attach AWS managed policies (AmazonEC2FullAccess and IAMUserChangePassword) to the developers group.

Update main.tf to add policy attachments:

```
@Zuha-Irfan → /workspaces/CC-ZuhaIrfan-073/Lab13 (main) $ vim main.tf
@Zuha-Irfan → /workspaces/CC-ZuhaIrfan-073/Lab13 (main) $ cat main.tf
provider "aws" {
  shared_config_files      = ["~/.aws/config"]
  shared_credentials_files = ["~/.aws/credentials"]
}

resource "aws_iam_group" "developers" {
  name = "developers"
  path = "/groups/"
}

output "group_details" {
  value = {
    group_name = aws_iam_group.developers.name
    group_arn  = aws_iam_group.developers.arn
    unique_id  = aws_iam_group.developers.unique_id
  }
}

resource "aws_iam_user" "lb" {
  name = "loadbalancer"
  path = "/users/"
  force_destroy = true
  tags = {
    DisplayName = "Load Balancer"
  }
}

resource "aws_iam_user_group_membership" "lb_membership" {
  user = aws_iam_user.lb.name
  groups = [
    aws_iam_group.developers.name
  ]
}

output "user_details" {
  value = {
    user_name = aws_iam_user.lb.name
    user_arn  = aws_iam_user.lb.arn
    unique_id = aws_iam_user.lb.unique_id
  }
}

resource "aws_iam_group_policy_attachment" "developer_ec2_fullaccess" {
    group = aws_iam_group.developers.name
    policy_arn = "arn:aws:iam::aws:policy/AmazonEC2FullAccess"
}

resource "aws_iam_group_policy_attachment" "change_password" {
     group = aws_iam_group.developers.name
     policy_arn = "arn:aws:iam::aws:policy/IAMUserChangePassword"
```

Apply the configuration:

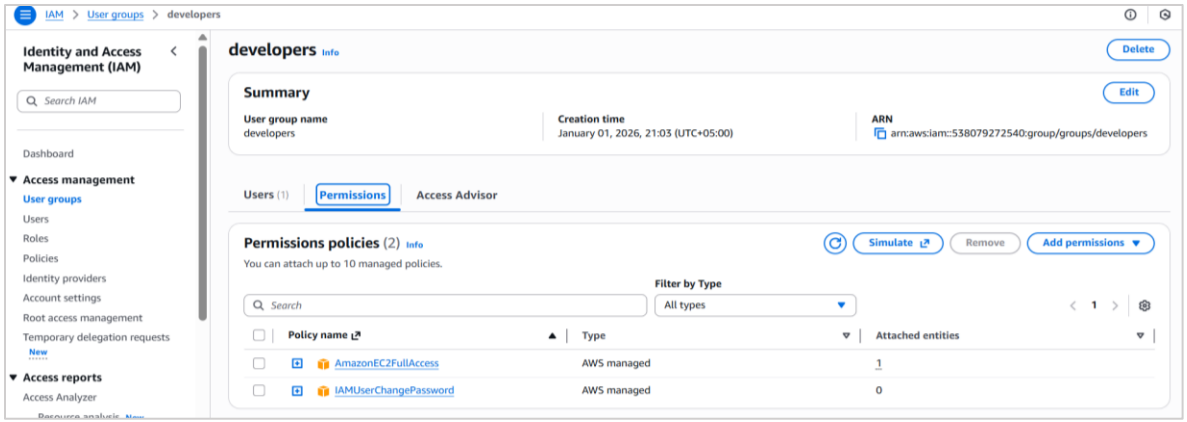
terraform apply -auto-approve

```
Apply complete! Resources: 2 added, 0 changed, 0 destroyed.

Outputs:

group_details = {
  "group_arn" = "arn:aws:iam::538079272540:group/groups/developers"
  "group_name" = "developers"
  "unique_id" = "AGPAX2SAGDZODXXWVG57E"
}
user_details = {
  "unique_id" = "AIDAX2SAGDZOH3YAHYCMA"
  "user_arn" = "arn:aws:iam::538079272540:user/users/loadbalancer"
  "user_name" = "loadbalancer"
}
@Zuha-Irfan → /workspaces/CC-ZuhaIrfan-073/Lab13 (main) $
```

Verify policies in AWS Console:

Navigate to IAM → Groups → developers

Click on "Permissions" tab



Task 4 — Create Login Profile for IAM User

In this task, you will create a login profile for the loadbalancer user using a bash script and null_resource provisioner.

Create variables.tf file:

```
@Zuha-Irfan ➜ /workspaces/CC-ZuhaIrfan-073/Lab13 (main) $ vim variable.tf
@Zuha-Irfan ➜ /workspaces/CC-ZuhaIrfan-073/Lab13 (main) $ cat variable.tf
variable "iam_password" {
  description = "Temporary password for the IAM user"
  type        = string
  sensitive   = true
  default = "1dontKnow"
}
```

Create the bash script create-login-profile.sh:

```
@Zuha-Irfan ➜ /workspaces/CC-ZuhaIrfan-073/Lab13 (main) $ vim create-login-profile.sh
@Zuha-Irfan ➜ /workspaces/CC-ZuhaIrfan-073/Lab13 (main) $ cat create-login-profile.sh
#!/usr/bin/env bash
set -euo pipefail

USERNAME="$1"
PASSWORD="$2"

# Check if login profile already exists
if aws iam get-login-profile --user-name "$USERNAME" >/dev/null 2>&1; then
  echo "Login profile already exists for $USERNAME.  Skipping."
else
  echo "Creating login profile for $USERNAME"
  aws iam create-login-profile \
    --user-name "$USERNAME" \
    --password "$PASSWORD" \
    --password-reset-required
fi
```

Make the script executable:

chmod +x create-login-profile.sh

```
@Zuha-Irfan ➜ /workspaces/CC-ZuhaIrfan-073/Lab13 (main) $ chmod +x create-login-profile.sh
@Zuha-Irfan ➜ /workspaces/CC-ZuhaIrfan-073/Lab13 (main) $ ls -l create-login-profile.sh
-rwxrwxrwx 1 codespace codespace 423 Jan  1 16:37 create-login-profile.sh
```

Update main.tf to add the null_resource provisioner:

```
@Zuha-Irfan ➜ /workspaces/CC-ZuhaIrfan-073/Lab13 (main) $ vim main.tf
@Zuha-Irfan ➜ /workspaces/CC-ZuhaIrfan-073/Lab13 (main) $ cat main.tf
provider "aws" {
  shared_config_files      = ["~/.aws/config"]
  shared_credentials_files = ["~/.aws/credentials"]
}

resource "aws_iam_group" "developers" {
  name = "developers"
  path = "/groups/"
}

output "group_details" {
  value = {
    group_name = aws_iam_group.developers.name
    group_arn  = aws_iam_group.developers.arn
    unique_id  = aws_iam_group.developers.unique_id
  }
}

resource "aws_iam_user" "lb" {
  name = "loadbalancer"
  path = "/users/"
  force_destroy = true
  tags = {
    DisplayName = "Load Balancer"
  }
}

resource "aws_iam_user_group_membership" "lb_membership" {
  user = aws_iam_user.lb.name
  groups = [
    aws_iam_group.developers.name
  ]
}

output "user_details" {
  value = {
    user_name = aws_iam_user.lb.name
    user_arn  = aws_iam_user.lb.arn
    unique_id = aws_iam_user.lb.unique_id
  }
}
resource "null_resource" "create_login_profile" {
  triggers = {
    password_hash = sha256(var.iam_password)
    user          = aws_iam_user.lb.name
  }

  depends_on = [aws_iam_user.lb]

  provisioner "local-exec" {
    command = "${path.module}/create-login-profile.sh ${aws_iam_user.lb.name} '${var.iam_password}'"
  }
}

resource "aws_iam_group_policy_attachment" "developer_ec2_fullaccess" {
```

Apply the configuration with a custom password:

terraform apply -auto-approve -var="iam_password=MySecurePass123!"

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

Outputs:

group_details = {
  "group_arn" = "arn:aws:iam::538079272540:group/groups/developers"
  "group_name" = "developers"
  "unique_id" = "AGPAX2SAGDZODXXWVG57E"
}
user_details = {
  "unique_id" = "AIDAX2SAGDZOH3YAHYCMA"
  "user_arn" = "arn:aws:iam::538079272540:user/users/loadbalancer"
  "user_name" = "loadbalancer"
}
```

Verify login profile creation:

aws iam get-login-profile --user-name loadbalancer

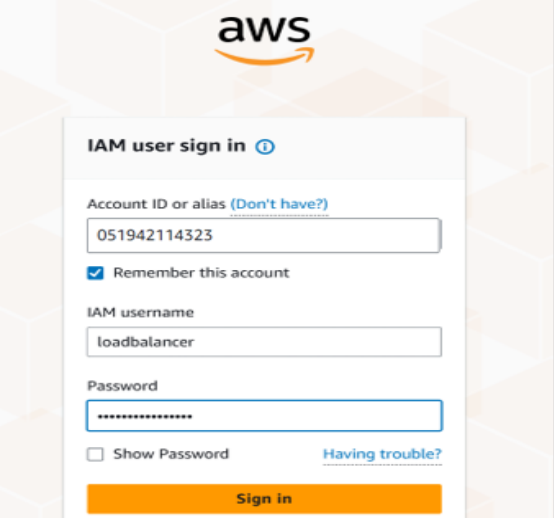
```
@Zuha-Irfan → /workspaces/CC-ZuhaIrfan-073/Lab13 (main) $ aws iam get-login-profile --user-name loadbalancer
{
    "LoginProfile": {
        "UserName": "loadbalancer",
        "CreateDate": "2026-01-01T16:44:27+00:00",
        "PasswordResetRequired": true
    }
}
```

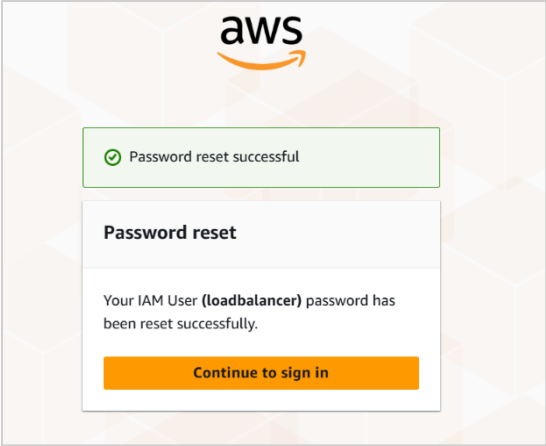
Test login in AWS Console:

Open AWS Console login page

Sign in as IAM user with username "loadbalancer" and the password you set

You should be prompted to change password





Task 5 — Generate Access Keys for IAM User

In this task, you will create access keys for the loadbalancer user and view them in terraform state.

Update main.tf to add access key resource and outputs:

```
resource "aws_iam_group_policy_attachment" "developer_ec2_fullaccess" {
    group = aws_iam_group.developers.name
    policy_arn = "arn:aws:iam::aws:policy/AmazonEC2FullAccess"
}

resource "aws_iam_group_policy_attachment" "change_password" {
     group = aws_iam_group.developers.name
     policy_arn = "arn:aws:iam::aws:policy/IAMUserChangePassword"
}
resource "aws_iam_access_key" "lb_access_key" {
  user = aws_iam_user.lb.name
}

output "access_key_id" {
  value = aws_iam_access_key.lb_access_key.id
}

output "access_key_secret" {
  value     = aws_iam_access_key.lb_access_key.secret
  sensitive = true
}
```

Apply the configuration:

terraform apply -auto-approve -var="iam_password=MySecurePass123!"

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

Outputs:

access_key_id = "AKIAX2SAGDZOBAZSUHMU"
access_key_secret = <sensitive>
group_details = {
  "group_arn" = "arn:aws:iam::538079272540:group/groups/developers"
  "group_name" = "developers"
  "unique_id" = "AGPAX2SAGDZODXXWVG57E"
}
user_details = {
  "unique_id" = "AIDAX2SAGDZOH3YAHYCMA"
  "user_arn" = "arn:aws:iam::538079272540:user/users/loadbalancer"
  "user_name" = "loadbalancer"
}
```

Display outputs:

terraform output

```
@Zuha-Irfan → /workspaces/CC-ZuhaIrfan-073/Lab13 (main) $ terraform output
access_key_id = "AKIAX2SAGDZOBAZSUHMU"
access_key_secret = <sensitive>
group_details = {
  "group_arn" = "arn:aws:iam::538079272540:group/groups/developers"
  "group_name" = "developers"
  "unique_id" = "AGPAX2SAGDZODXXWVG57E"
}
user_details = {
  "unique_id" = "AIDAX2SAGDZOH3YAHYCMA"
  "user_arn" = "arn:aws:iam::538079272540:user/users/loadbalancer"
  "user_name" = "loadbalancer"
}
```

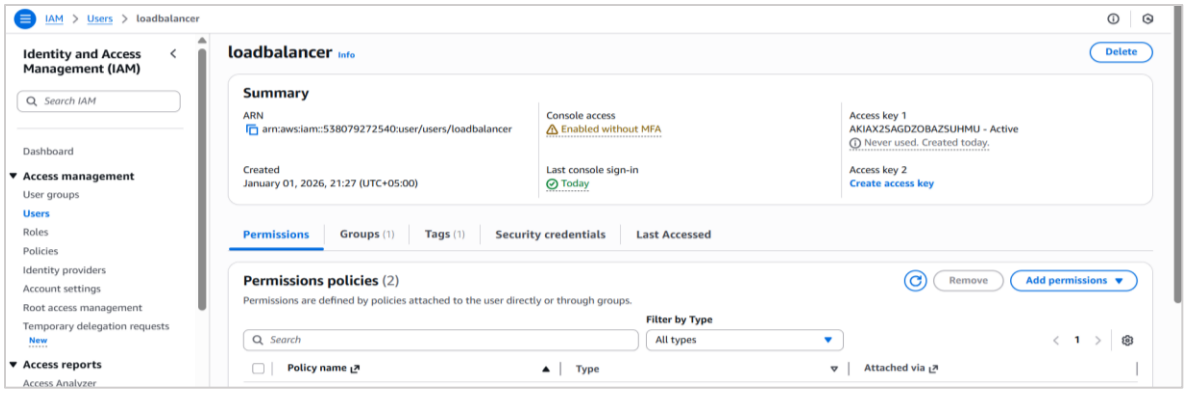
View the secret in terraform state:

cat terraform.tfstate | grep -A 10 "access_key_secret"

```
@Zuha-Irfan → /workspaces/CC-ZuhaIrfan-073/Lab13 (main) $ cat terraform.tfstate | grep -A 10 "access_key_secret"
    "access_key_secret": {
      "value": "757IpPCA4j4O5x9H/LFUJMvvr2lKTfKJQlhUGLb/",
      "type": "string",
      "sensitive": true
    },
    "group_details": {
      "value": {
        "group_arn": "arn:aws:iam::538079272540:group/groups/developers",
        "group_name": "developers",
        "unique_id": "AGPAX2SAGDZODXXWVG57E"
      },
```

Verify access key in AWS Console:

Navigate to IAM → Users → loadbalancer → Security credentials



Task 6 — Implement Terraform Remote State with S3

In this task, you will configure Terraform to use S3 backend for remote state storage.

Create S3 bucket in AWS Console:

Navigate to S3 in AWS Console

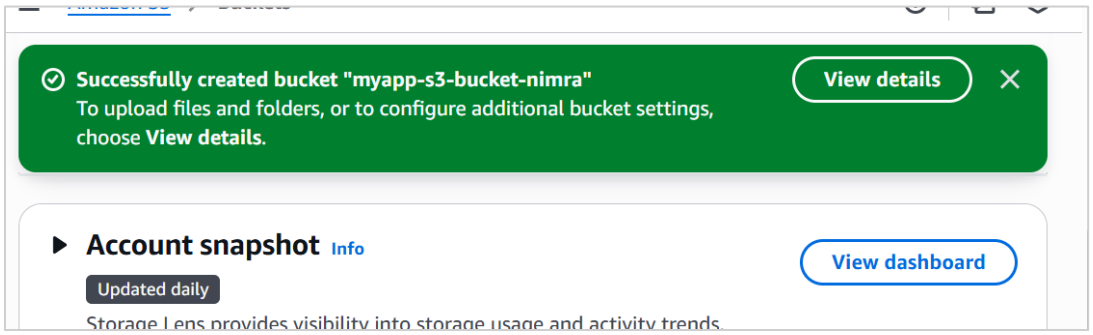
Click "Create bucket"

Bucket name: myapp-s3-bucket-demo (use a unique name if this is taken)

Enable versioning

Keep other settings as default

Click "Create bucket"



Update main.tf to add S3 backend configuration:

Add this at the beginning of main.tf (before the provider block):

```
@Zuha-Irfan → /workspaces/CC-ZuhaIrfan-073/Lab13 (main) $ vim main.tf
@Zuha-Irfan → /workspaces/CC-ZuhaIrfan-073/Lab13 (main) $ cat main.tf
provider "aws" {
  shared_config_files      = ["~/.aws/config"]
  shared_credentials_files = ["~/.aws/credentials"]
}
terraform {
  backend "s3" {
    bucket = "myapp-s3-bucket-nimra"
    key    = "myapp/terraform.tfstate"
    region = "me-central-1"
    encrypt = true
    use_lockfile = true
  }
}
resource "aws_iam_group" "developers" {
  name = "developers"
  path = "/groups/"
}
```

Reinitialize Terraform with the backend:

terraform init -migrate-state

Type yes when prompted to migrate state

```
@Zuha-Irfan → /workspaces/CC-ZuhaIrfan-073/Lab13 (main) $ terraform init -migrate-state
Initializing the backend...
Do you want to copy existing state to the new backend?
  Pre-existing state was found while migrating the previous "local" backend to the
  newly configured "s3" backend. No existing state was found in the newly
  configured "s3" backend. Do you want to copy this state to the new "s3"
  backend? Enter "yes" to copy and "no" to start with an empty state.

  Enter a value: yes


Successfully configured the backend "s3"! Terraform will automatically
use this backend unless the backend configuration changes.
Initializing provider plugins...
- Reusing previous version of hashicorp/aws from the dependency lock file
- Reusing previous version of hashicorp/null from the dependency lock file
- Using previously-installed hashicorp/aws v6.27.0
- Using previously-installed hashicorp/null v3.2.4

Terraform has been successfully initialized!
```

Apply the configuration:

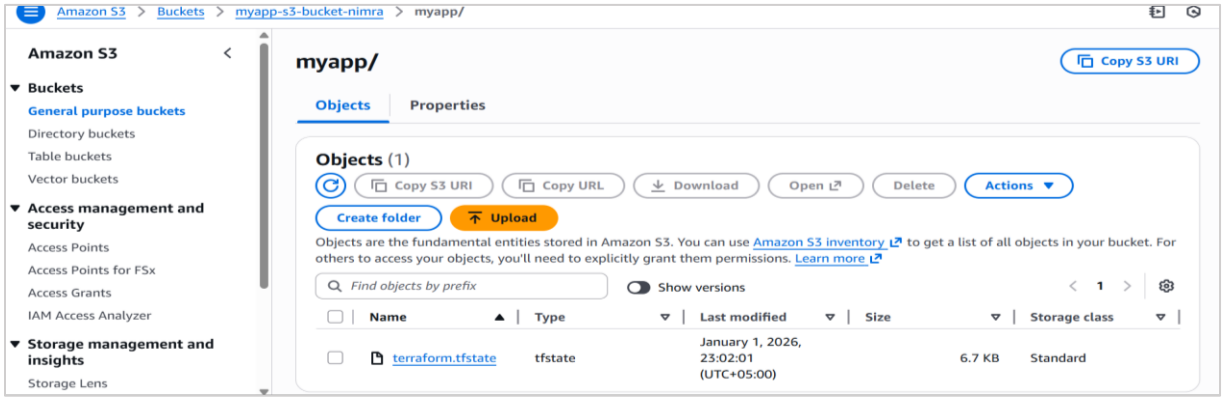
terraform apply -auto-approve -var="iam_password=MySecurePass123!"

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

Outputs:

access_key_id = "AKIAX2SAGDZOBAZSUHMU"
access_key_secret = <sensitive>
group_details = {
  "group_arn" = "arn:aws:iam::538079272540:group/groups/developers"
  "group_name" = "developers"
  "unique_id" = "AGPAX2SAGDZODXXWVG57E"
}
user_details = {
  "unique_id" = "AIDAX2SAGDZOH3YAHYCMA"
  "user_arn" = "arn:aws:iam::538079272540:user/users/loadbalancer"
  "user_name" = "loadbalancer"
}
```

Verify state file in S3:

Navigate to S3 → myapp-s3-bucket-demo → myapp/

You should see terraform.tfstate file



Check local state file:

ls -la terraform.tfstate*

```
@Zuha-Irfan ➜ /workspaces/CC-ZuhaIrfan-073/Lab13 (main) $ ls -la terraform.tfstate*
-rw-rw-rw- 1 codespace codespace    0 Jan  1 18:02 terraform.tfstate
-rw-rw-rw- 1 codespace codespace 6882 Jan  1 18:02 terraform.tfstate.backup
```

Destroy resources and verify state change:

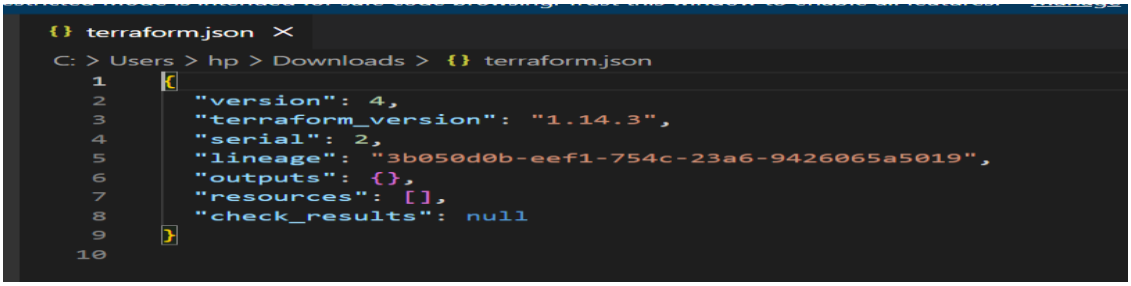
terraform destroy -auto-approve

```
aws_iam_group_policy_attachment.developer_ec2_fullaccess: Destroying... [id=developers-20260101163333406800000001]
aws_iam_user_group_membership.lb_membership: Destroying... [id=terraform-20260101162718388700000001]
aws_iam_access_key.lb_access_key: Destroying... [id=AKIAX2SAGDZOBAZSUHMU]
aws_iam_group_policy_attachment.developer_ec2_fullaccess: Destruction complete after 1s
aws_iam_user_group_membership.lb_membership: Destruction complete after 1s
aws_iam_access_key.lb_access_key: Destruction complete after 1s
aws_iam_user.lb: Destroying... [id=loadbalancer]
aws_iam_group_policy_attachment.change_password: Destruction complete after 1s
aws_iam_group.developers: Destroying... [id=developers]
aws_iam_group.developers: Destruction complete after 0s
aws_iam_user.lb: Destruction complete after 2s

Destroy complete! Resources: 7 destroyed.
```

Verify updated state in S3:

Refresh S3 bucket view

Check the terraform.tfvstate file (it should show empty resources)



Task 7 — Create Multiple Users from CSV File

In this task, you will create multiple IAM users dynamically from a CSV file.

Create locals.tf file:

```
@Zuha-Irfan → /workspaces/CC-ZuhaIrfan-073/Lab13 (main) $ vim locals.tf
@Zuha-Irfan → /workspaces/CC-ZuhaIrfan-073/Lab13 (main) $ cat locals.tf
locals {
  users = csvdecode(file("users.csv"))
}
```

Create users.csv file:

```
@Zuha-Irfan → /workspaces/CC-ZuhaIrfan-073/Lab13 (main) $ vim users.csv
@Zuha-Irfan → /workspaces/CC-ZuhaIrfan-073/Lab13 (main) $ cat users.csv
user_name
Michael
Dwight
Jim
Pam
Ryan
Andy
Robert
Stanley
Kevin
Angela
Oscar
Phyllis
Toby
Kelly
Darryl
Creed
Meredith
Erin
Gabe
Jan
David
Holly
Charles
Jo
Clark
Peter
```

Update main.tf to create multiple users:

```
# Create login profiles for all users
resource "null_resource" "create_login_profiles" {
  for_each = aws_iam_user.users

  triggers = {
    password_hash = sha256(var.iam_password)
    user          = each.value.name
  }

  depends_on = [aws_iam_user.users]

  provisioner "local-exec" {
    command = "${path.module}/create-login-profile.sh ${each.value.name} '${var.iam_password}'"
  }
}

# Create access keys for all users
resource "aws_iam_access_key" "users_access_keys" {
  for_each = aws_iam_user.users

  user = each.value.name
}

# Output all user details
output "all_users_details" {
  value = {
    for user_name, user in aws_iam_user.users : user_name => {
      user_arn       = user.arn
      user_unique_id = user.unique_id
      access_key_id  = aws_iam_access_key.users_access_keys[user_name].id
    }
  }
}

# Output all access key secrets (sensitive)
output "all_access_key_secrets" {
  value = {
    for user_name, key in aws_iam_access_key.users_access_keys : user_name => key.secret
  }
  sensitive = true
}
```

Replace the single user resources with:

Reinitialize Terraform (since we changed the configuration significantly):

terraform init

```
@Zuha-Irfan → /workspaces/CC-ZuhaIrfan-873/Lab13 (main) $ terraform init
Initializing the backend...
Initializing provider plugins...
- Reusing previous version of hashicorp/null from the dependency lock file
- Reusing previous version of hashicorp/aws from the dependency lock file
- Using previously-installed hashicorp/null v3.2.4
- Using previously-installed hashicorp/aws v6.27.0

Terraform has been successfully initialized!

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

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

Apply the configuration to create all users:

terraform apply -auto-approve -var="iam_password=MySecurePass123!"

```
Apply complete! Resources: 107 added, 0 changed, 0 destroyed.

Outputs:

all_access_key_secrets = <sensitive>
all_users_details = {
  "Andy" = {
    "access_key_id" = "AKIAX2SAGDZOEWI5VT62"
    "user_arn" = "arn:aws:iam::538079272540:user/users/Andy"
    "user_unique_id" = "AIDAX2SAGDZOPDE3FPKE2"
  }
  "Angela" = {
    "access_key_id" = "AKIAX2SAGDZOIVAU6GV5"
    "user_arn" = "arn:aws:iam::538079272540:user/users/Angela"
    "user_unique_id" = "AIDAX2SAGDZOJPGRAK4MV"
  }
  "Charles" = {
    "access_key_id" = "AKIAX2SAGDZOEF6W5RN6"
    "user_arn" = "arn:aws:iam::538079272540:user/users/Charles"
    "user_unique_id" = "AIDAX2SAGDZOOYCFQXMCK"
  }
  "Clark" = {
    "access_key_id" = "AKIAX2SAGDZOLSABSMHS"
    "user_arn" = "arn:aws:iam::538079272540:user/users/Clark"
    "user_unique_id" = "AIDAX2SAGDZOKERCTXF3J"
  }
  "Creed" = {
    "access_key_id" = "AKIAX2SAGDZOH45OJ3XG"
    "user_arn" = "arn:aws:iam::538079272540:user/users/Creed"
```

Display the outputs:

terraform output

```
@Zuha-Irfan → /workspaces/CC-ZuhaIrfan-873/Lab13 (main) $ terraform output
all_access_key_secrets = <sensitive>
all_users_details = {
  "Andy" = {
    "access_key_id" = "AKIAX2SAGDZOEWI5VT62"
    "user_arn" = "arn:aws:iam::538079272540:user/users/Andy"
    "user_unique_id" = "AIDAX2SAGDZOPDE3FPKE2"
  }
  "Angela" = {
    "access_key_id" = "AKIAX2SAGDZOIVAU6GV5"
    "user_arn" = "arn:aws:iam::538079272540:user/users/Angela"
    "user_unique_id" = "AIDAX2SAGDZOJPGRAK4MV"
  }
  "Charles" = {
    "access_key_id" = "AKIAX2SAGDZOEF6W5RN6"
    "user_arn" = "arn:aws:iam::538079272540:user/users/Charles"
    "user_unique_id" = "AIDAX2SAGDZOOYCFQXMCK"
  }
  "Clark" = {
    "access_key_id" = "AKIAX2SAGDZOLSABSMHS"
    "user_arn" = "arn:aws:iam::538079272540:user/users/Clark"
    "user_unique_id" = "AIDAX2SAGDZOKERCTXF3J"
  }
  "Creed" = {
    "access_key_id" = "AKIAX2SAGDZOH45OJ3XG"
    "user_arn" = "arn:aws:iam::538079272540:user/users/Creed"
    "user_unique_id" = "AIDAX2SAGDZOFFEMSNBUO"
  }
  "Darryl" = {
    "access_key_id" = "AKIAX2SAGDZOHMRADRVS"
    "user_arn" = "arn:aws:iam::538079272540:user/users/Darryl"
    "user_unique_id" = "AIDAX2SAGDZOH3PMSD564"
  }
```

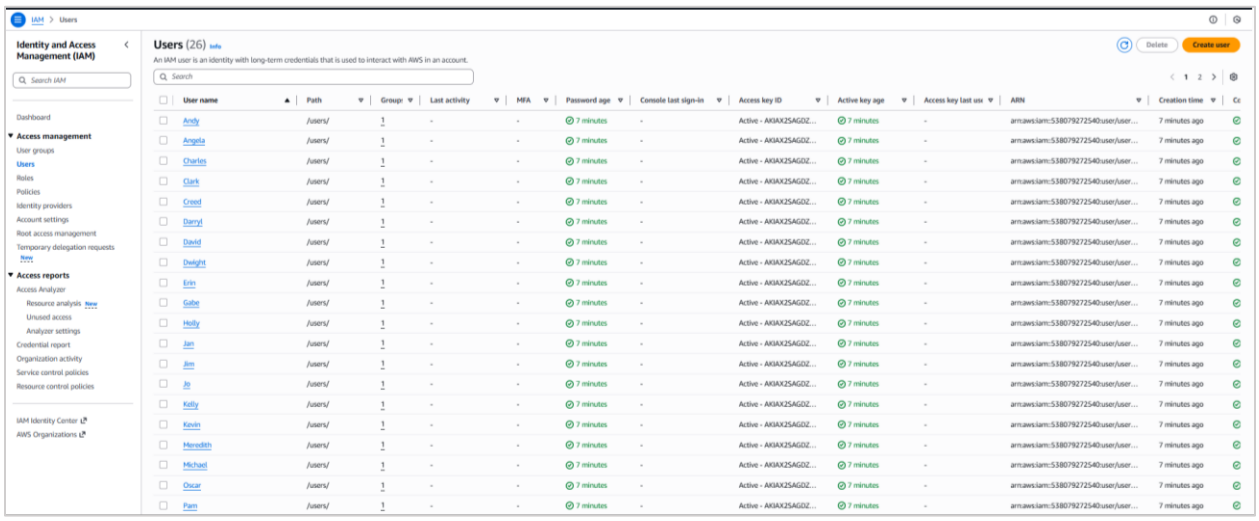
View secrets in terraform. tfstate:

cat terraform.tfstate | grep -A 5 "all_access_key_secrets"

```
@Zuha-Irfan → /workspaces/CC-ZuhaIrfan-073/Lab13 (main) $ cat terraform.tfstate | grep -A 5 "all_access_key_secrets"
```

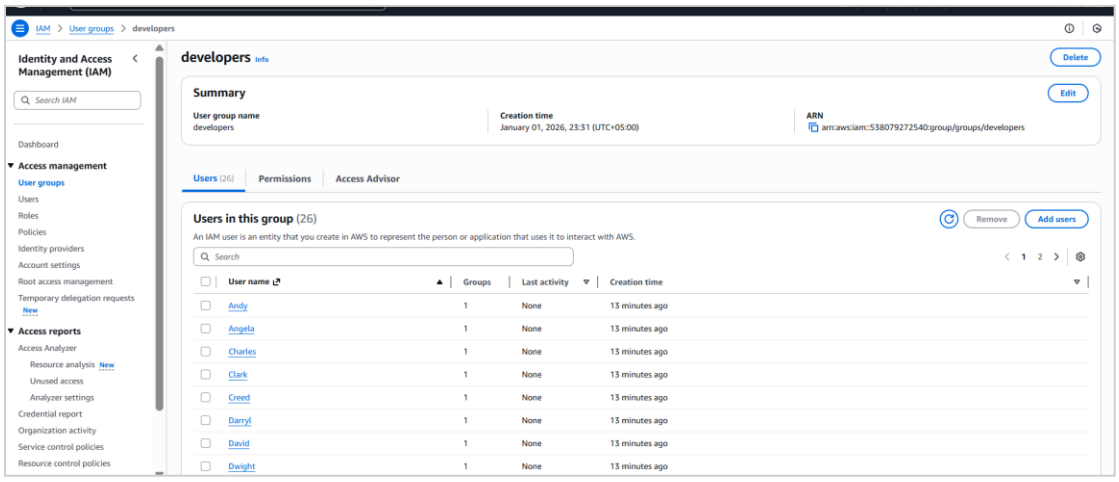
Verify all users in AWS Console:

Navigate to IAM → Users



Verify group membership:

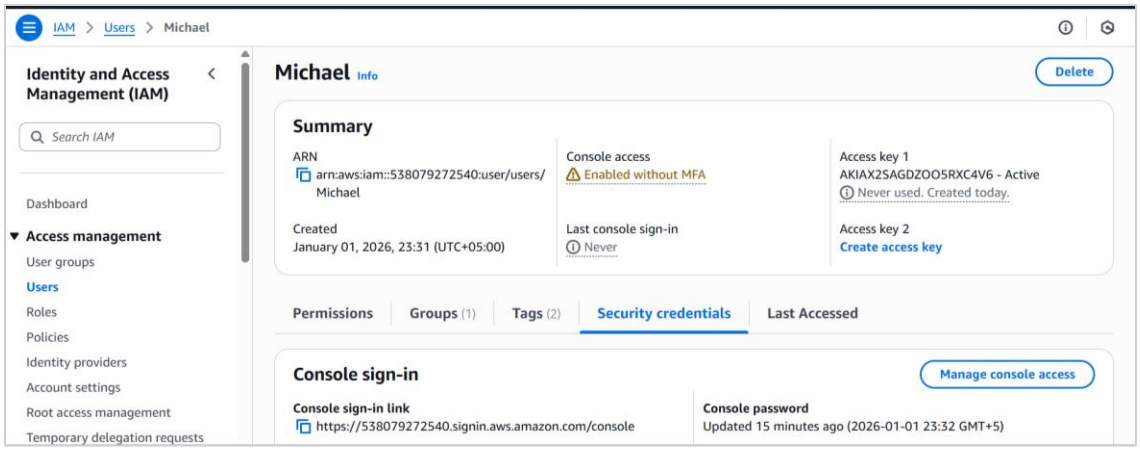
Navigate to IAM → Groups → developers → Users tab



Verify one user's access keys:

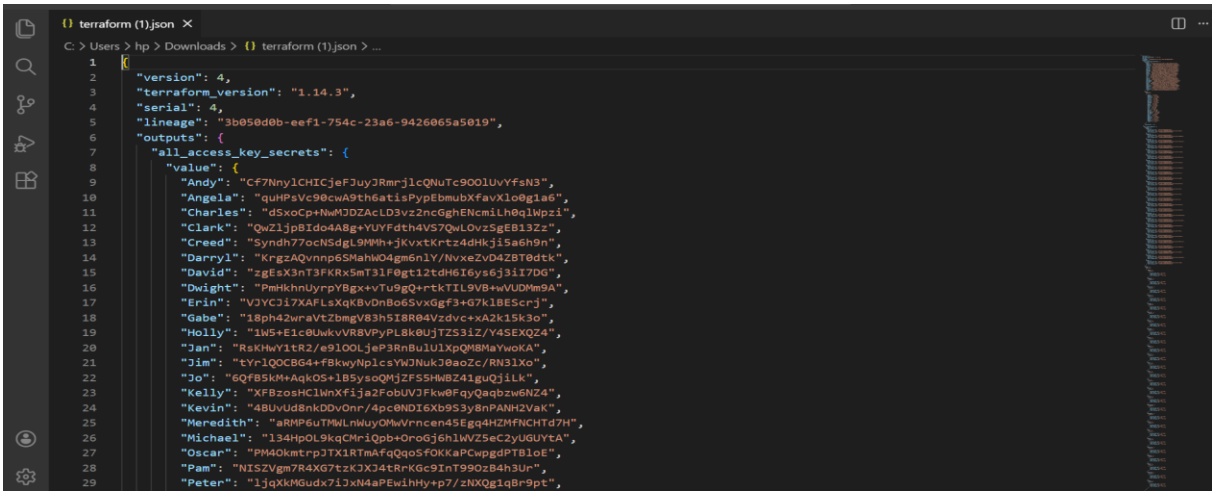
Click on any user (e.g., "Michael")

Go to Security credentials tab



Check terraform state in S3:

Navigate to S3 bucket and view the state file



Cleanup

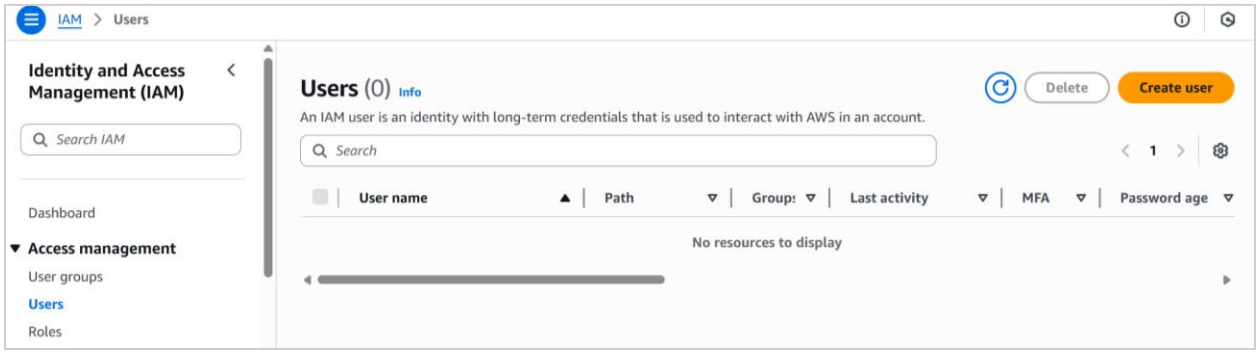
Destroy all resources:

```
aws_iam_user.users["Jim"]: Destroying... [id=Jim]
aws_iam_user.users["Meredith"]: Destruction complete after 3s
aws_iam_user.users["Clark"]: Destroying... [id=Clark]
aws_iam_user.users["David"]: Destruction complete after 2s
aws_iam_user.users["Oscar"]: Destruction complete after 2s
aws_iam_user.users["Charles"]: Destruction complete after 2s
aws_iam_user.users["Jim"]: Destruction complete after 2s
aws_iam_user.users["Michael"]: Destruction complete after 5s
aws_iam_user.users["Peter"]: Destruction complete after 4s
aws_iam_user.users["Jo"]: Destruction complete after 5s
aws_iam_user.users["Stanley"]: Destruction complete after 5s
aws_iam_user.users["Gabe"]: Destruction complete after 7s
aws_iam_user.users["Clark"]: Destruction complete after 7s

Destroy complete! Resources: 107 destroyed.
@Zuha-Irfan → /workspaces/CC-ZuhaIrfan-073/Lab13 (main) $
```

terraform destroy -auto-approve

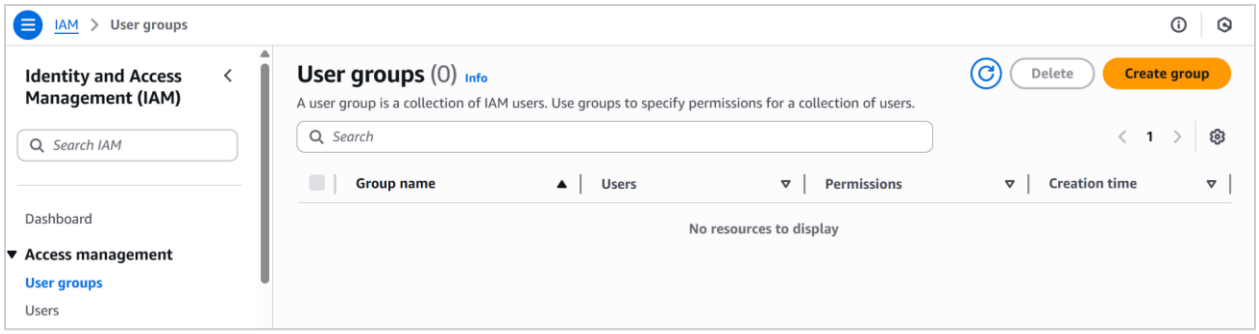
Verify users deleted in AWS Console:

Navigate to IAM → Users



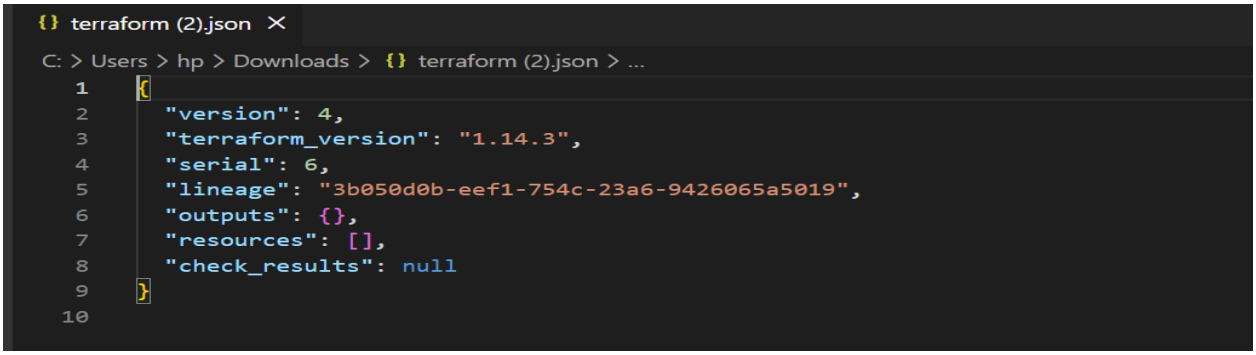
Verify group deleted in AWS Console:

Navigate to IAM → Groups



Check S3 state file:

Navigate to S3 bucket



List all project files:

ls -la

```
@Zuha-Irfan ➜ /workspaces/CC-ZuhaIrfan-073/Lab13 (main) $ ls -la
total 61776
drwxrwxrwx+ 5 codespace root          4096 Jan  1 18:30 .
drwxr-xrwx+ 5 codespace root          4096 Jan  1 15:21 ..
drwxrwxrwx+ 8 codespace root          4096 Jan  1 15:21 .git
drwxr-xr-x+ 3 codespace codespace     4096 Jan  1 18:02 .terraform
-rw-r--r--  1 codespace codespace     2422 Jan  1 16:44 .terraform.lock.hcl
-rw-rw-rw-  1 codespace root            10 Jan  1 15:21 README.md
drwxr-xr-x+ 3 codespace codespace     4096 Dec 30 19:13 aws
-rw-rw-rw-  1 codespace codespace 63198381 Jan  1 15:44 awscliv2.zip
-rwxrwxrwx  1 codespace codespace      423 Jan  1 16:37 create-login-profile.sh
-rw-rw-rw-  1 codespace codespace       50 Jan  1 18:17 locals.tf
-rw-rw-rw-  1 codespace codespace     2899 Jan  1 18:30 main.tf
-rw-rw-rw-  1 codespace codespace        0 Jan  1 18:02 terraform.tfstate
-rw-rw-rw-  1 codespace codespace     6882 Jan  1 18:02 terraform.tfstate.backup
-rw-rw-rw-  1 codespace codespace      167 Jan  1 18:18 users.csv
-rw-rw-rw-  1 codespace codespace      150 Jan  1 16:37 variable.tf
```

(Optional) Delete S3 bucket:

If you want to clean up completely, delete the S3 bucket from AWS Console