

Cloud Computing Lab

Lab 13 tasks

Lab 13 – Terraform IAM Management with AWS Balancer

Submitted To: Muhammad Shoaib

Submitted From: Bushra Ashraf Bhatti

Registration Number: 2023-BSE-015

```

inflating: aws/dist/prompt_toolkit-3.0.51.dist-info/licenses/AUTHORS.rst
inflating: aws/dist/wheel-0.45.1.dist-info/entry_points.txt
inflating: aws/dist/wheel-0.45.1.dist-info/INSTALLER
inflating: aws/dist/wheel-0.45.1.dist-info/REQUESTED
inflating: aws/dist/wheel-0.45.1.dist-info/METADATA
inflating: aws/dist/wheel-0.45.1.dist-info/LICENSE.txt
inflating: aws/dist/wheel-0.45.1.dist-info/direct_url.json
inflating: aws/dist/wheel-0.45.1.dist-info/WHEEL
inflating: aws/dist/wheel-0.45.1.dist-info/RECORD
● @bushraashraf05 → /workspaces/13lab (main) $ sudo ./aws/install
You can now run: /usr/local/bin/aws --version
● @bushraashraf05 → /workspaces/13lab (main) $ aws configure
AWS Access Key ID [None]: AKIA4ZT4HDFZEYXX50Q
AWS Secret Access Key [None]: iRWkcmSRngbHEN8rJp1Y9JczBrNtSV/+6dHqSNII
Default region name [None]: me-central-1
Default output format [None]: json
● @bushraashraf05 → /workspaces/13lab (main) $ aws sts get-caller-identity
{
  "UserId": "AIDA4ZT4HDFZLL2WQERQ4",
  "Account": "879655065970",
  "Arn": "arn:aws:iam::879655065970:user/terraform-admin"
}
○ @bushraashraf05 → /workspaces/13lab (main) $

```

Main.tf:

```

GNU nano 7.2 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
  }
}

```

```

● @bushraashraf05 → /workspaces/13lab (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 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!

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.
○ @bushraashraf05 → /workspaces/13lab (main) $

```

```
@bushraashraf05 →/workspaces/13lab (main) $ terraform validate
Success! The configuration is valid.
```

```
@bushraashraf05 →/workspaces/13lab (main) $
```

```
}
```

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

```
Changes to Outputs:
```

```
+ group_details = {
  + group_arn  = (known after apply)
  + group_name = "developers"
  + unique_id  = (known after apply)
}
```

```
aws_iam_group.developers: Creating...
```

```
aws_iam_group.developers: Creation complete after 1s [id=developers]
```

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

```
Outputs:
```

```
group_details = {
  "group_arn" = "arn:aws:iam::879655065970:group/groups/developers"
  "group_name" = "developers"
  "unique_id"  = "AGPA4ZT4HDFZPMZRMNPII"
}
```

```
@bushraashraf05 →/workspaces/13lab (main) $
```

Task 2 — Create IAM User with Group Membership:

Update 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
    }
}
```

Terraform-apply-auto approve:

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

@bushraashraf05 → /workspaces/13lab (main) $ terraform apply -auto-approve
+ user_arn = (known after apply)
+ user_name = "loadbalancer"
}
aws_iam_user.lb: Creating...
aws_iam_user.lb: Creation complete after 1s [id=loadbalancer]
aws_iam_user_group_membership.lb_membership: Creating...
aws_iam_user_group_membership.lb_membership: Creation complete after 0s [id=terraform-2026010619553618980000]

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

Outputs:

group_details = {
  "group_arn" = "arn:aws:iam::879655065970:group/groups/developers"
  "group_name" = "developers"
  "unique_id" = "AGPA4ZT4HDFZPMZRMNPPII"
}
user_details = {
  "unique_id" = "AIDA4ZT4HDFZC3NMB76V3"
  "user_arn" = "arn:aws:iam::879655065970:user/users/loadbalancer"
  "user_name" = "loadbalancer"
}
@bushraashraf05 → /workspaces/13lab (main) $
```

```
}
• @bushraashraf05 → /workspaces/13lab (main) $ terraform output
group_details = {
  "group_arn" = "arn:aws:iam::879655065970:group/groups/developers"
  "group_name" = "developers"
  "unique_id" = "AGPA4ZT4HDFZPMZRMNPPII"
}
user_details = {
  "unique_id" = "AIDA4ZT4HDFZC3NMB76V3"
  "user_arn" = "arn:aws:iam::879655065970:user/users/loadbalancer"
  "user_name" = "loadbalancer"
}
@bushraashraf05 → /workspaces/13lab (main) $
```

Confirm on aws:

Users deleted.

Users (1) Info

An IAM user is an identity with long-term credentials that is used to interact with AWS in an account.

Search

<input type="checkbox"/>	User name	Path	Groups	Last activity	MFA	Password age	Console last sign-in	Access key ID
<input type="checkbox"/>	loadbalancer	/users/	1	-	-	-	-	-

1

Access Management

- User groups
- Users
- Roles
- Policies
- Identity providers

Delete Create user

Task 3 — Attach Policies to IAM Group

Update 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"
}
```

```
@bushraashraf05 → /workspaces/13lab (main) $ nano main.tf
@bushraashraf05 → /workspaces/13lab (main) $ terraform init
Initializing the backend...
Initializing provider plugins...
- Reusing previous version of hashicorp/aws from the dependency lock file
- 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.

```
@bushraashraf05 → /workspaces/13lab (main) $
```

Commands will detect it and remind you to do so if necessary.

```
@bushraashraf05 → /workspaces/13lab (main) $ terraform validate
Success! The configuration is valid.
```

```
@bushraashraf05 → /workspaces/13lab (main) $
```

```
main* ↻ 0 0 0 0
```

```
@bushraashraf05 → /workspaces/13lab (main) $ terraform apply -auto-approve
aws_iam_user.lb: Refreshing state... [id=loadbalancer]
aws_iam_group.developers: Refreshing state... [id=developers]
```

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

Terraform planned the following actions, but then encountered a problem:

```
# aws_iam_user.loadbalancer will be created
+ resource "aws_iam_user" "loadbalancer" {
  + arn          = (known after apply)
  + force_destroy = true
  + id           = (known after apply)
  + name         = "loadbalancer"
  + path         = "/users/"
  + tags         = {
    + "DisplayName" = "Load Balancer"
  }
  + tags_all     = {
```

Task 4 — Create Login Profile for IAM User

```
GNU nano 7.2 variables.tf *
variable "iam_password" {
  description = "Temporary password for the IAM user"
  type        = string
  sensitive   = true
  default     = "IdontKnow"
}
```



```

GNU nano 7.2                                     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

```

Update main.tf:

```

GNU nano 7.2                                     main.tf *
value = {
  user_name = aws_iam_user.loadbalancer.name
  user_arn  = aws_iam_user.loadbalancer.arn
  unique_id = aws_iam_user.loadbalancer.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}'"
  }
}

```

```

• @bushraashraf05 → /workspaces/13lab (main) $ terraform init -upgrade
  Initializing the backend...
  Initializing provider plugins...
  - Finding latest version of hashicorp/aws...
  - Finding latest version of hashicorp/null...
  - Using previously-installed hashicorp/aws v6.27.0
  - Installing hashicorp/null v3.2.4...
  - Installed hashicorp/null v3.2.4 (signed by HashiCorp)
  Terraform has made some changes to the provider dependency selections recorded
  in the .terraform.lock.hcl file. Review those changes and commit them to your
  version control system if they represent changes you intended to make.

  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.
• @bushraashraf05 → /workspaces/13lab (main) $

```

main* 0 0 0 0

Add this resource after user creation:

```

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}'"

  }

}

```

```

51 }
52 resource "null_resource" "create_login_profile" {
53   triggers = {
54     password_hash = sha256(var.iam_password)
55     user          = aws_iam_user.lb.name
56   }
57
58   depends_on = [aws_iam_user.lb]
59
60   provisioner "local-exec" {
61     command = "${path.module}/create-login-profile.sh ${aws_iam_user.lb.name} '${var.iam_password}'"
62   }
63 }

```

terraform init -upgrade

1. Download the installer

curl "https://awscli.amazonaws.com/awscli-exe-linux-x86_64.zip" -o "awscliv2.zip"

2. Unzip it (Install unzip first if needed)

sudo apt-get install unzip -y

unzip -o awscliv2.zip

3. Run the installer

sudo ./aws/install --update

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

```

null_resource.create_login_profile (local-exec): (output suppressed due to sensitive value in config)
null_resource.create_login_profile: Creation complete after 4s [id=8755808408375960610]

```

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

Outputs:

```

group_details = {
  "group_arn" = "arn:aws:iam::279635434623:group/groups/developers"
  "group_name" = "developers"
  "unique_id" = "AGPAUCG4SDR7Z4TE35ZRX"
}
user_details = {
  "unique_id" = "AIDAUCG4SDR7ZJYJP2L40"
  "user_arn" = "arn:aws:iam::279635434623:user/users/loadbalancer"
  "user_name" = "loadbalancer"
}

```

IAM username

loadbalancer

Password

.....

☐ Show Password

[Having trouble?](#)

Sign in

Sign in using root user email

[Create a new AWS account](#)

Password reset ⓘ

Your account (**279635434623**) password has expired or requires a reset.

To continue, please verify your old and set a new password for **loadbalancer** ([not you?](#)).

Old Password

☐ Show Password

New Password

Confirm New Password

☐ Show Password

Confirm Password Change

Task 5 — Generate Access Keys for IAM User:

```
GNU nano 7.2                                main.tf *
# IAM User
#####
resource "aws_iam_user" "lb" {
  name       = "loadbalancer"
  path       = "/users/"
  force_destroy = true

  tags = {
    DisplayName = "Load Balancer"
  }
}

#####
# Login Profile (via script)
#####
resource "null_resource" "create_login_profile" {
  triggers = {
    password_hash = sha256(var.iam_password)
    user          = aws_iam_user.lb.name
  }
}
```

```
"group_arn" = "arn:aws:iam::279635434623:group/groups/developers"
"group_name" = "developers"
"unique_id" = "AGPAUCG4SDR7Z4TE35ZRX"
}
user_details = {
  "unique_id" = "AIDAUCG4SDR7ZJYJP2L40"
  "user_arn" = "arn:aws:iam::279635434623:user/users/loadbalancer"
  "user_name" = "loadbalancer"
}
```

Terraform output:

```
"group_arn" = "arn:aws:iam::279635434623:group/groups/developers"
"group_name" = "developers"
"unique_id" = "AGPAUCG4SDR7Z4TE35ZRX"
}
user_details = {
  "unique_id" = "AIDAUCG4SDR7ZJYJP2L40"
  "user_arn" = "arn:aws:iam::279635434623:user/users/loadbalancer"
  "user_name" = "loadbalancer"
}
```

Access keys (1)

[Create access key](#)

Use access keys to send programmatic calls to AWS from the AWS CLI, AWS Tools for PowerShell, AWS SDKs, or direct AWS API calls. You can have a maximum of two access keys (active or inactive) at a time. [Learn more](#)

AKIAUCG4SDR73CFIFEHN

Description

-

Last used

None

Last used region

N/A

Status

✓ Active

Created

14 minutes ago

Last used service

N/A

Actions ▾

Task 6 — Implement Terraform Remote State with S3

aws

Search

[Alt+S]

United States (N. Virginia)

bushraashraf05

[Amazon S3](#) > [Buckets](#) > Create bucket

Create bucket [Info](#)

Buckets are containers for data stored in S3.

General configuration

AWS Region

US East (N. Virginia) us-east-1

Bucket type [Info](#)

☒ General purpose

Recommended for most use cases and access patterns. General purpose buckets are the original S3 bucket type. They allow a mix of storage classes that redundantly store objects across multiple Availability Zones.

☐ Directory

Recommended for low-latency use cases. These buckets use only the S3 Express One Zone storage class, which provides faster processing of data within a single Availability Zone.

Bucket name [Info](#)

Bucket names must be 3 to 63 characters and unique within the global namespace. Bucket names must also begin and end with a letter or number. Valid characters are a-z, 0-9, periods (.), and hyphens (-). [Learn more](#)

Copy settings from existing bucket - optional

Only the bucket settings in the following configuration are copied.

[Choose bucket](#)

Format: s3://bucket/prefix

Bucket Versioning

Versioning is a means of keeping multiple variants of an object in the same bucket. You can use versioning to preserve, retrieve, and restore every version of every object stored in your Amazon S3 bucket. With versioning, you can easily recover from both unintended user actions and application failures. [Learn more](#)

Bucket Versioning

- ☐ Disable
☒ Enable

Tags - optional

You can use bucket tags to analyze, manage and specify permissions for a bucket. [Learn more](#)

① You can use s3:ListTagsForResource, s3:TagResource, and s3:UntagResource APIs to manage tags on S3 general purpose buckets for access control in addition to cost allocation and resource organization. To ensure a seamless transition, please provide permissions to s3:ListTagsForResource, s3:TagResource, and s3:UntagResource actions. [Learn more](#)



Search

[Alt+S]



Amazon S3 > Buckets

General purpose buckets

All AWS Regions

Directory buckets

General purpose buckets (1) Info



Copy ARN

Empty

Delete

Create bucket

Buckets are containers for data stored in S3.

Find buckets by name

< 1 > ⚙

Name	AWS Region	Creation date
<input type="radio"/> myapp-s3-bucket-demo-bushra123	Middle East (UAE) me-central-1	January 10, 2026, 11:19:47 (UTC+05:00)

Objects Metadata **Properties** Permissions Metrics Management Access Points

Bucket overview

AWS Region
Middle East (UAE) me-central-1

Amazon Resource Name (ARN)
 arn:aws:s3::myapp-s3-bucket-demo-bushra123

Creation date
January 10, 2026, 11:19:47 (UTC+05:00)

Bucket Versioning

Edit

Versioning is a means of keeping multiple variants of an object in the same bucket. You can use versioning to preserve, retrieve, and restore every version of every object stored in your Amazon S3 bucket. With versioning, you can easily recover from both unintended user actions and application failures. [Learn more](#)

Bucket Versioning
Enabled

Multi-factor authentication (MFA) delete

An additional layer of security that requires multi-factor authentication for changing Bucket Versioning settings and permanently deleting object versions. To modify MFA delete settings, use the AWS CLI, AWS SDK, or the Amazon S3 REST API. [Learn more](#)

Disabled

Update main.tf:

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
GNU nano 7.2 main.tf *
terraform {
  backend "s3" {
    bucket    = "myapp-s3-bucket-demo"
    key       = "myapp/terraform.tfstate"
    region    = "me-central-1"
    encrypt   = true
    use_lockfile = true
  }
}
#####
# Provider
#####
provider "aws" {
  shared_config_files = ["~/.aws/config"]
  shared_credentials_files = ["~/.aws/credentials"]
}

#####
# Variable
#####
```

```
• @bushraashraf05 →/workspaces/13lab (main) $ terraform validate
Success! The configuration is valid.
```

```
○ @bushraashraf05 →/workspaces/13lab (main) $
```

```
• @bushraashraf05 →/workspaces/13lab (main) $ aws sts get-caller-identity
{
  "UserId": "AIDA4ZT4HDFZC3NMB76V3",
  "Account": "879655065970",
  "Arn": "arn:aws:iam::879655065970:user/users/loadbalancer"
}
```

```
○ @bushraashraf05 →/workspaces/13lab (main) $
```

Add permissions

Add user to an existing group or create a new one. Using groups is a best-practice way to manage user's permissions by job functions. [Learn more](#)

Permissions options

☐ Add user to group

Add user to an existing group, or create a new group. We recommend using groups to manage user permissions by job function.

☐ Copy permissions

Copy all group memberships, attached managed policies, inline policies, and any existing permissions boundaries from an existing user.

☒ Attach policies directly

Attach a managed policy directly to a user. As a best practice, we recommend attaching policies to a group instead. Then, add the user to the appropriate group.

Permissions policies (1/1440)

Filter by Type			
Q AmazonS3FullAccess		All types	1 match
Policy name	Type	Attached entities	
<input checked="" type="checkbox"/> AmazonS3FullAccess	AWS managed	0	

Cancel

Next

```
@bushraashraf05 →/workspaces/13lab (main) $ rm -rf .terraform
@bushraashraf05 →/workspaces/13lab (main) $ rm -f .terraform.lock.hcl
@bushraashraf05 →/workspaces/13lab (main) $ terraform init -reconfigure -migrate-state
```

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

Destroy complete! Resources: 7 destroyed.

Task 7 — Create Multiple Users from CSV File

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

GNU nano 7.2

locals.tf *

```
locals {  
  users = csvdecode(file("users.csv"))  
}
```

Create users.csv file:

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

GNU nano 7.2

users.csv *

```
user_name  
Michael  
Dwight  
Jim  
Pam  
Ryan  
Andy  
Robert  
Stanley  
Kevin  
Angela  
Oscar  
Phyllis  
Toby  
Kelly  
Darryl  
Creed  
Meredith  
Erin  
Gabe
```

Help	Write Out	Where Is	Cut	Execute	Location
Exit	Read File	Replace	Paste	Justify	Go To Line

Update main.tf:

(After IAM group resource):

Add this code:

```
#####  
  
# Create multiple IAM users from CSV  
  
#####  
  
resource "aws_iam_user" "users" {  
  for_each = { for user in local.users : user.user_name => user }  
  
  name      = each.value.user_name
```



```
path      = "/users/"
force_destroy = true

tags = {
  DisplayName = each.value.user_name
  CreatedBy   = "Terraform"
}
}

#####
# Add all users to developers group
#####

resource "aws_iam_user_group_membership" "users_membership" {
  for_each = aws_iam_user.users

  user = each.value.name

  groups = [
    aws_iam_group.developers.name
  ]
}

#####
# 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  
}
```

Terraform init:

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:

```
"user_unique_id" = "AIDAUCG4SDR7T27MJUDY3"  
}  
"Stanley" = {  
  "access_key_id" = "AKIAUCG4SDR74XX5LVNR"  
  "user_arn" = "arn:aws:iam::279635434623:user/users/Stanley"  
  "user_unique_id" = "AIDAUCG4SDR7XYJNOBK12"  
}  
"Toby" = {  
  "access_key_id" = "AKIAUCG4SDR72DDAUOSN"  
  "user_arn" = "arn:aws:iam::279635434623:user/users/Toby"  
  "user_unique_id" = "AIDAUCG4SDR7UR6JT5D3I"  
}  
}
```

Users (28) [Info](#)

An IAM user is an identity with long-term credentials that is used to interact with AWS in an account.

<input type="checkbox"/>	User name	Path	Group:	Last activity
<input type="checkbox"/>	admin	/	0	✔ 23 days ago
<input type="checkbox"/>	Andy	/users/	1	-
<input type="checkbox"/>	Angela	/users/	1	-
<input type="checkbox"/>	Charles	/users/	1	-
<input type="checkbox"/>	Clark	/users/	1	-
<input type="checkbox"/>	Creed	/users/	1	-
<input type="checkbox"/>	Darryl	/users/	1	-

Users in this group (26) [Refresh](#) [Filter](#)

An IAM user is an entity that you create in AWS to represent the person or application that uses it to interact with AWS.

<input type="checkbox"/>	User name
<input type="checkbox"/>	Andy
<input type="checkbox"/>	Angela
<input type="checkbox"/>	Charles
<input type="checkbox"/>	Clark
<input type="checkbox"/>	Creed
<input type="checkbox"/>	Darryl
<input type="checkbox"/>	David
<input type="checkbox"/>	Dwight

Verify one user's access keys:

Access keys (1) [Create access key](#)

Use access keys to send programmatic calls to AWS from the AWS CLI, AWS Tools for PowerShell, AWS SDKs, or direct AWS API calls. You can have a maximum of two access keys (active or inactive) at a time. [Learn more](#)

AKIAUCG4SDR76A2KGBNB		Actions
Description	-	Status
		✔ Active
Last used	None	Created
		42 minutes ago
Last used region	N/A	Last used service
		N/A

Check terraform state in S3:

Access keys (1)

[Create access key](#)

Use access keys to send programmatic calls to AWS from the AWS CLI, AWS Tools for PowerShell, AWS SDKs, or direct AWS API calls. You can have a maximum of two access keys (active or inactive) at a time. [Learn more](#)

AKIAUCG4SDR76A2KGBNB

[Actions](#)

Description

-

Status

✓ Active

Last used

None

Created

42 minutes ago

Last used region

N/A

Last used service

N/A

Destroy all resources:

PROBLEMS

OUTPUT

DEBUG CONSOLE

TERMINAL

PORTS

```
● @bushraashraf05 → /workspaces/13lab (main) $ terraform init -reconfigure
```

```
Initializing the backend...
```

```
Initializing provider plugins...
```

- Finding latest version of hashicorp/aws...
- Finding latest version of hashicorp/null...
- Installing hashicorp/aws v6.28.0...
- Installed hashicorp/aws v6.28.0 (signed by HashiCorp)
- Installing hashicorp/null v3.2.4...
- Installed hashicorp/null v3.2.4 (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!
```

```
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.
```

```
○ @bushraashraf05 → /workspaces/13lab (main) $
```

main* 0 0 0 0

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.

```
@bushraashraf05 → /workspaces/13lab (main) $ terraform destroy -auto-approve
```

```
aws_iam_user.lb: Refreshing state... [id=loadbalancer]
```

```
aws_iam_user_group_membership.lb_membership: Refreshing state... [id=terraform-20260106195536189800000001]
```

```
aws_iam_group.developers: Refreshing state... [id=developers]
```
