

7. Laboratory Exercise

Step 1: First we will check that no instance is running on EC2.

The screenshot shows the AWS Management Console interface for the EC2 service. The left sidebar contains navigation links for EC2 Dashboard, Events, Instances, Instance Types, Launch Templates, Spot Requests, Savings Plans, Reserved Instances, Dedicated Hosts, Scheduled Instances, Capacity Reservations, Images, AMIs, AMI Catalog, Elastic Block Store, Volumes, Snapshots, and Lifecycle Manager. The main content area is titled 'Resources' and shows a table of EC2 resources in the US East (N. Virginia) Region. The table includes columns for Instances (running), Elastic IPs, Load balancers, Snapshots, Auto Scaling Groups, Instances, Placement groups, Volumes, Dedicated Hosts, Key pairs, and Security groups. A notification banner at the bottom of the Resources section states: 'Easily size, configure, and deploy Microsoft SQL Server Always On availability groups on AWS using the AWS Launch Wizard for SQL Server. Learn more'. The 'Launch instance' section is visible, showing a 'Launch instance' button and a 'Migrate a server' button. The 'Service health' section shows the status of the AWS service in the US East (N. Virginia) Region as 'This service is operating normally'. The 'Account attributes' section shows the Default VPC and Settings. The 'Explore AWS' section provides information about AWS Graviton2 and ways to reduce AWS costs.

Step 2: Create an IAM user with Programmatic Password, Administrator access and download access key and secret key from download.csv

The screenshot shows the 'Launch instance' wizard in the AWS Management Console. The 'Summary' section displays the configuration for the instance, including the number of instances (1), the virtual server type (t2.micro), the firewall (security group), and the storage (volumes). A 'Free tier' notification banner is visible, stating: 'Free tier: In your first year includes 750 hours of t2.micro (or t3.micro in the Regions in which t2.micro is unavailable) instance usage on free tier AMIs per month, 30 GiB of EBS storage, 2 million I/Os, 1 GB of snapshots, and 100 GB of bandwidth to the internet.' The 'Launch instance' button is visible at the bottom right.

The image displays three sequential screenshots of the AWS IAM console interface, showing the process of creating a new user.

Screenshot 1: User details

Step 2: Set permissions

User name: exp8

The user name can have up to 64 characters. Valid characters: A-Z, a-z, 0-9, and +, -, ., @, _ (hyphen)

☒ Provide user access to the AWS Management Console - optional

If you're providing console access to a person, it's a [best practice](#) to manage their access in IAM Identity Center.

Are you providing console access to a person?

User type

☐ Specify a user in Identity Center - Recommended

We recommend that you use Identity Center to provide console access to a person. With Identity Center, you can centrally manage user access to their AWS accounts and cloud applications.

☒ I want to create an IAM user

We recommend that you create IAM users only if you need to enable programmatic access through access keys, service-specific credentials for AWS CodeCommit or Amazon Keyspaces, or a backup credential for emergency account access.

Console password

☐ Autogenerated password

You can view the password after you create the user.

☒ Custom password

Enter a custom password for the user.

Must be at least 8 characters long

Must include at least three of the following mix of character types: uppercase letters (A-Z), lowercase letters (a-z), numbers (0-9), and symbols ! @ # \$ % ^ & * () _ + - (hyphen) ~ [] | ' " , . /

☐ Show password

© 2023, Amazon Web Services India Private Limited or its affiliates. Privacy Terms Cookie preferences

Screenshot 2: Permissions options

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](#)

☐ 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, and inline policies 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 (1/1124)

Choose one or more policies to attach to your new user.

Filter by Type: All types

Search

Policy name	Type	Attached entities
<input type="checkbox"/> AccessAnalyzerServiceRolePolicy	AWS managed	0
<input checked="" type="checkbox"/> AdministratorAccess	AWS managed - job function	0
<input type="checkbox"/> AdministratorAccess-Amplify	AWS managed	0
<input type="checkbox"/> AdministratorAccess-AWSElas...	AWS managed	0
<input type="checkbox"/> AlexaForBusinessDeviceSetup	AWS managed	0
<input type="checkbox"/> AlexaForBusinessFullAccess	AWS managed	0

© 2023, Amazon Web Services India Private Limited or its affiliates. Privacy Terms Cookie preferences

Screenshot 3: Retrieve password

User created successfully

You can view and download the user's password and email instructions for signing in to the AWS Management Console.

[View user](#)

[IAM](#) > [Users](#) > Create user

Step 1: Specify user details

Step 2: Set permissions

Step 3: Review and create

Step 4: Retrieve password

Retrieve password

You can view and download the user's password below or email users instructions for signing in to the AWS Management Console. This is the only time you can view and download this password.

Console sign-in details

Email sign-in instructions

Console sign-in URL

<https://928565427545.signin.aws.amazon.com/console>

User name

exp8

Console password

Show

Cancel Download .csv file Return to users list

© 2023, Amazon Web Services India Private Limited or its affiliates. Privacy Terms Cookie preferences

The screenshot shows the AWS IAM console for user 'exp8'. The left sidebar contains navigation links for Identity and Access Management (IAM), Access management, Access reports, and CloudShell. The main content area displays the user's summary, including ARN, console access status, and creation date. Below the summary, there are tabs for Permissions, Groups, Tags, Security credentials, and Access Advisor. The Permissions tab is active, showing a table of permissions policies attached to the user.

Policy name	Type	Attached via
AdministratorAccess	AWS managed - job function	Directly

The screenshot shows the 'Create access key' wizard in the AWS IAM console, specifically Step 1: 'Access key best practices & alternatives'. The wizard provides guidance on avoiding long-term credentials and offers several use cases for the access key. The 'Command Line Interface (CLI)' option is selected.

Use case

- ☒ Command Line Interface (CLI)
You plan to use this access key to enable the AWS CLI to access your AWS account.
- ☐ Local code
You plan to use this access key to enable application code in a local development environment to access your AWS account.
- ☐ Application running on an AWS compute service
You plan to use this access key to enable application code running on an AWS compute service like Amazon EC2, Amazon ECS, or AWS Lambda to access your AWS account.
- ☐ Third-party service
You plan to use this access key to enable access for a third-party application or service that monitors or manages your AWS resources.
- ☐ Application running outside AWS
You plan to use this access key to enable an application running on an on-premises host, or to use a local AWS client or third-party AWS plugin.
- ☐ Other

The screenshot shows the 'Retrieve access keys' step (Step 3) of the 'Create access key' wizard. It displays the generated access key and secret access key. A green banner at the top states: 'Access key created. This is the only time that the secret access key can be viewed or downloaded. You cannot recover it later. However, you can create a new access key any time.' Below the keys, there are 'Access key best practices' listed.

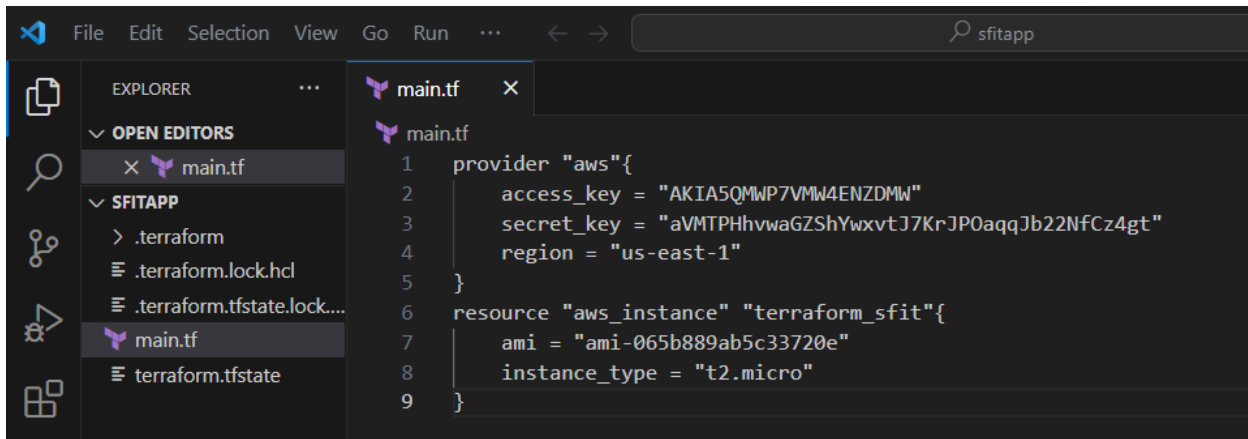
Access key best practices

- Never store your access key in plain text, in a code repository, or in code.
- Disable or delete access key when no longer needed.
- Enable least-privilege permissions.
- Rotate access keys regularly.

For more details about managing access keys, see the [best practices for managing AWS access keys](#).

Download .csv file Done

Step 3: Now write a Terraform program in vs code, create new file with .tf extension



Step 4: Now initialize the terraform ...type c:\SfitApp> terraform init

```
PS C:\sfitapp> terraform init

Initializing the backend...

Initializing provider plugins...
- Finding latest version of hashicorp/aws...
- Installing hashicorp/aws v5.13.1...
- Installed hashicorp/aws v5.13.1 (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.
```

Step 5: c:\sfitApp>terraform plan

```
PS C:\sfitapp> 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_instance.terraform_sfit will be created
+ resource "aws_instance" "terraform_sfit" {
+   ami                        = "ami-065b889ab5c33720e"
+   arn                       = (known after apply)
+   associate_public_ip_address = (known after apply)
+   availability_zone          = (known after apply)
+   cpu_core_count             = (known after apply)
+   cpu_threads_per_core       = (known after apply)
+   disable_api_stop           = (known after apply)
+   disable_api_termination    = (known after apply)
+   ebs_optimized              = (known after apply)
+   get_password_data          = false
+   host_id                   = (known after apply)
+   host_resource_group_arn    = (known after apply)
+   iam_instance_profile       = (known after apply)
+   id                        = (known after apply)
+   instance_initiated_shutdown_behavior = (known after apply)
+   instance_lifecycle         = (known after apply)
+   instance_state             = (known after apply)
+   instance_type              = "t2.micro"
}
```

```
+ subnet_id = (known after apply)
+ tags_all = (known after apply)
+ tenancy = (known after apply)
+ user_data = (known after apply)
+ user_data_base64 = (known after apply)
+ user_data_replace_on_change = false
+ vpc_security_group_ids = (known after apply)
}
```

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

Step 6: Check the instance on Ec2 before terraform apply

The screenshot shows the AWS Management Console with the EC2 Global view selected. The Resources section displays a table of EC2 resources in the US East (N. Virginia) Region. The table shows 0 Instances (running), 0 Auto Scaling Groups, 0 Dedicated Hosts, 0 Elastic IPs, 0 Instances, 0 Key pairs, 0 Load balancers, 0 Placement groups, 1 Security groups, 0 Snapshots, and 0 Volumes. A notification banner at the bottom of the Resources section states: "Easily size, configure, and deploy Microsoft SQL Server Always On availability groups on AWS using the AWS Launch Wizard for SQL Server. [Learn more](#)". The "Launch instance" section is visible, showing a "Launch instance" button and a "Migrate a server" button. The "Service health" section shows the status of the AWS Health Dashboard as "This service is operating normally".

Step 7: Terraform apply

```
Note: You didn't use the -out option to save this plan, so Terraform can't guarantee to take exactly these actions if you run "terraform apply" now.
PS C:\sfitapp> terraform apply

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_instance.terraform_sfit will be created
+ resource "aws_instance" "terraform_sfit" {
  + ami = "ami-065b889ab5c33720e"
  + arn = (known after apply)
  + associate_public_ip_address = (known after apply)
  + availability_zone = (known after apply)
  + cpu_core_count = (known after apply)
  + cpu_threads_per_core = (known after apply)
  + disable_api_stop = (known after apply)
  + disable_api_termination = (known after apply)
  + ebs_optimized = (known after apply)
  + get_password_data = false
  + host_id = (known after apply)
  + host_resource_group_arn = (known after apply)
  + iam_instance_profile = (known after apply)
  + id = (known after apply)
  + instance_initiated_shutdown_behavior = (known after apply)
  + instance_lifecycle = (known after apply)
  + instance_state = (known after apply)
  + instance_type = "t2.micro"
  + ipv6_address_count = (known after apply)
  + ipv6_addresses = (known after apply)
}
```

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

Do you want to perform these actions?
  Terraform will perform the actions described above.
  Only 'yes' will be accepted to approve.

  Enter a value: yes

aws_instance.terraform_sfit: Creating...
aws_instance.terraform_sfit: Still creating... [10s elapsed]
aws_instance.terraform_sfit: Still creating... [20s elapsed]
aws_instance.terraform_sfit: Still creating... [30s elapsed]
aws_instance.terraform_sfit: Creation complete after 36s [id=i-04f61b42162d119c1]

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

Step 8: Check terraform created instances on EC2...we have created 3 instances.

The screenshot shows the AWS Management Console for the US East (N. Virginia) region. The 'Resources' section displays a table of EC2 resources:

Resource Type	Count
Instances (running)	1
Elastic IPs	0
Load balancers	0
Snapshots	0
Auto Scaling Groups	0
Instances	1
Placement groups	0
Volumes	1
Dedicated Hosts	0
Key pairs	0
Security groups	1

The 'Launch instance' section is visible, showing a 'Launch instance' button and a 'Migrate a server' button. The 'Service health' section shows the AWS Health Dashboard with a status of 'This service is operating normally'.

Step 9: Now destroy the instance from command prompt....c:\SfitApp> terraform destroy

```
PS C:\sfitapp> terraform destroy
aws_instance.terraform_sfit: Refreshing state... [id=i-04f61b42162d119c1]

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

Terraform will perform the following actions:

# aws_instance.terraform_sfit will be destroyed
- resource "aws_instance" "terraform_sfit" {
  - ami              = "ami-065b889ab5c33720e" -> null
  - arn              = "arn:aws:ec2:us-east-1:928565427545:instance/i-04f61b42162d119c1" -> null
  - associate_public_ip_address = true -> null
  - availability_zone = "us-east-1d" -> null
  - cpu_core_count    = 1 -> null
  - cpu_threads_per_core = 1 -> null
}
```

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

Do you really want to destroy all resources?
  Terraform will destroy all your managed infrastructure, as shown above.
  There is no undo. Only 'yes' will be accepted to confirm.

  Enter a value: yes

aws_instance.terraform_sfit: Destroying... [id=i-04f61b42162d119c1]
aws_instance.terraform_sfit: Still destroying... [id=i-04f61b42162d119c1, 10s elapsed]
aws_instance.terraform_sfit: Still destroying... [id=i-04f61b42162d119c1, 20s elapsed]
aws_instance.terraform_sfit: Still destroying... [id=i-04f61b42162d119c1, 30s elapsed]
aws_instance.terraform_sfit: Destruction complete after 33s

Destroy complete! Resources: 1 destroyed.
```