QATIP Intermediate
AWS Lab 03
Setting Up AWS Secrets Manager and IAM user for Terraform Authentication

Contents

Lab Objectives 1
Teaching Points 1
Before you begin 2
Administrators Set Up AWS Secrets Manager and IAM User 2
Lab set-up 2
Administration creates an IAM User and Secret 3
Developers Retrieve IAM Credentials from AWS Secrets Manager 5
Define Parameters and Secure Retrieval Function 5
Verify Authentication & Run Terraform 5
Using IAM Roles with Terraform 6
Revert to Lab IAM Identity (awsstudent) 6
Create the Role and Attach Policies 6
Assume the Role 7
Lab clean-up 7

Lab Objectives

This lab presents methods by which multiple developers can authenticate Terraform dynamically without needing to store or share sensitive credentials directly.

Teaching Points

Step 1: Administrator Tasks

- The security team sets up **AWS Secrets Manager** to securely store Terraform authentication credentials.
- An **IAM User** is created with restricted access.

Step 2: Developer Tasks

- Developers retrieve authentication credentials using **AWS Secrets Manager**.
- Developers can **retrieve** credentials but **cannot modify** them.

Before you begin

1. Ensure you have completed Lab0 before attempting this lab.
2. In the IDE terminal pane, enter the following command...

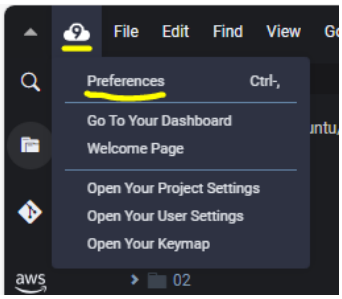
   cd ~/environment/aws-tf-int/labs/03

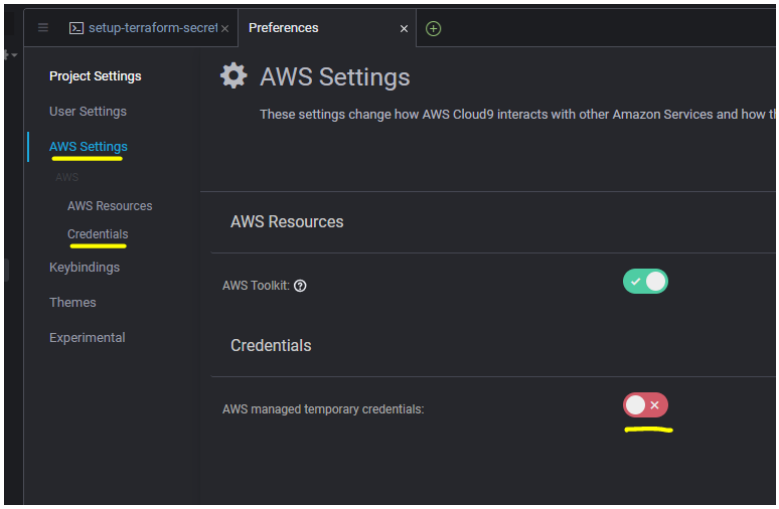
3. This shifts your current working directory to labs/03. Ensure all commands are executed in this directory
4. Close any open files and use the Explorer pane to navigate to and open the labs/03 folder.

Administrators Set Up AWS Secrets Manager and IAM User

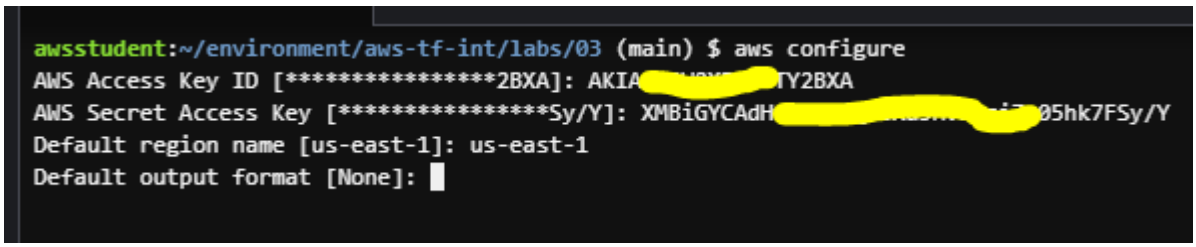
Lab set-up

5. Cloud9 uses temporary credentials by default which do not have sufficient authorization to configure Secrets Manager. Navigate to Preferences, AWS Settings, Credentials and disable temporary credentials...





6. Use "**aws configure**" to supply explicit credentials, providing the Access Key and Secret Access Key generated for your student account, as documented earlier. (Navigate to QA.QWIKLABS if you have not noted these down). Configure us-east-1 as your default region and leave default output format empty....



7. Update the Cloud9 environment to install **jq**, a lightweight command-line tool for parsing, filtering, and transforming JSON data...

   sudo apt-get update && sudo apt-get install -y jq

Administration creates an IAM User and Secret

8. Examine the script **setup-terraform-secrets.sh** in **aws-tf-int/labs/03**
9. Ensure lab shell scripts are executable...

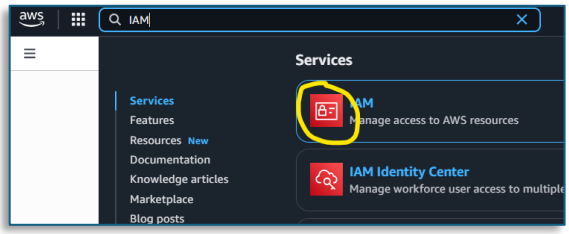
   chmod +x setup-terraform-secrets.sh

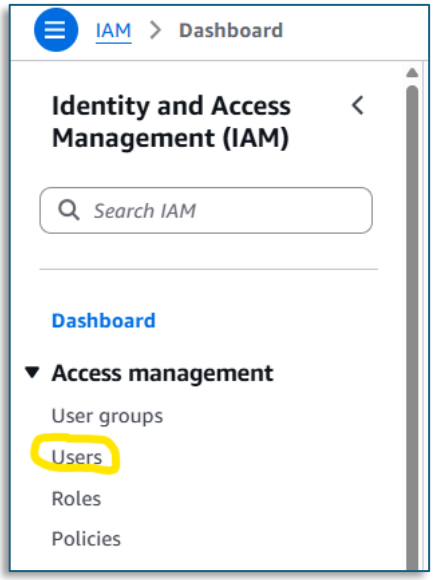
   chmod +x retrieve-aws-credentials.sh

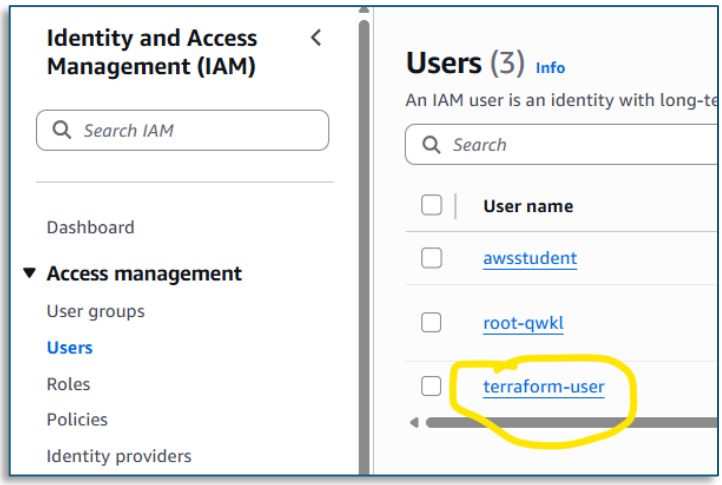
10. Execute the script: **./setup-terraform-secrets.sh**
11. To verify the secret is created you can run

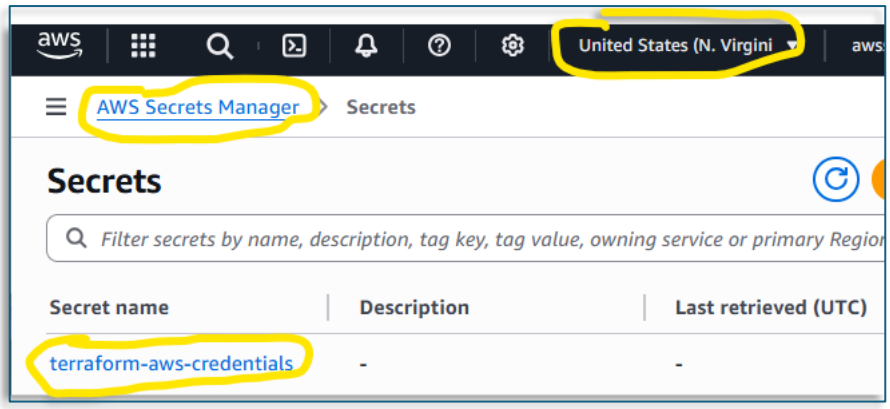
aws secretsmanager get-secret-value /
--secret-id terraform-aws-credentials

12. Switch to the Console and ensure the IAM user and secret are created successfully.







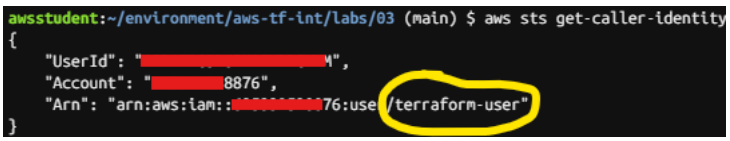


Developers Retrieve IAM Credentials from AWS Secrets Manager

13. Developers retrieve stored credentials using AWS CLI and export them as environment variables.

Define Parameters and Secure Retrieval Function

14. Examine the script **retrieve-aws-credentials.sh** in **aws-tf-int/labs/03**
15. Execute the script: **./retrieve-aws-credentials.sh**
16. Check stored credentials: **cat aws_credentials.env**
17. Reload credentials in a new session: **source aws_credentials.env**
18. Verify that you are now using the terraform-user credentials by running: **aws sts get-caller-identity**



Verify Authentication & Run Terraform

Developers now use Terraform with the retrieved AWS credentials.

19. Update line 18 of **main.tf** with a unique S3 bucket name of your choice and save the change.
20. Ensure AWS credentials are loaded: **source aws_credentials.env**
21. Initialize Terraform: **terraform init**

22. Test infrastructure deployment: **terraform plan**
23. Deploy the infrastructure: **terraform apply**
24. Verify the AWS resources in the AWS Console.

Using IAM Roles with Terraform

The preceding steps outlined the method of using IAM Users and Secrets Manager. As an alternative, we will now switch to using IAM Roles with temporary credentials, which improves security and aligns with best practices. In production environments, Terraform developers would perform steps in the 'Assuming a role' section below, all other tasks would be performed by cloud administrators. In the lab, you will perform all steps.

Revert to Lab IAM Identity (awsstudent)

25. Before assuming a role, ensure you are no longer using the 'terraform-user' credentials.
26. Run the following in your terminal:

```
unset AWS_ACCESS_KEY_ID
unset AWS_SECRET_ACCESS_KEY
unset AWS_SESSION_TOKEN
```

27. Verify your identity (should return 'awsstudent'):

```
aws sts get-caller-identity
```

Create the Role and Attach Policies

28. Update the provided file **trust-policy.json**, replacing **<AccountID>** with your actual AWS account ID. Save the changes.

29. Create the IAM Role and attach AdministratorAccess (This is very broad access, in production environments the level of access granted to this role would be more limited):

```
aws iam create-role   --role-name TerraformRole /
```

--assume-role-policy-document file://trust-policy.json

aws iam attach-role-policy --role-name TerraformRole /
--policy-arn arn:aws:iam::aws:policy/AdministratorAccess

Assume the Role

As a terraform developer you would authenticate to google cloud with normal credentials, typically with minimal permissions, but including the ability to assume a role. As you action your terraform files, this role is assumed, with sufficient permissions to manage AWS infrastructure as needed.

30. Uncomment all the currently commented-out lines in **main.tf**
31. Update line 13, replacing **<Account_ID>** with you lab account ID
32. Save the changes
33. Re-initialize Terraform: **terraform init -upgrade**
34. Test infrastructure deployment: **terraform plan**
35. Deploy the infrastructure: **terraform apply**
36. Note the output showing the Role being used during command execution...

Lab clean-up

37. Run **terraform destroy** and confirm with **yes**

Congratulations, you have completed this lab