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

Contents

Lab Objectives

This lab presents a method 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

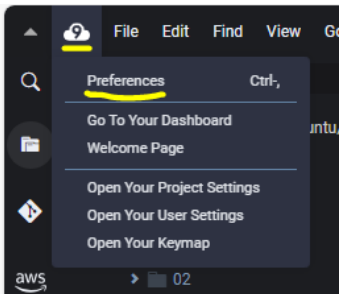
1. Ensure you have completed Lab0 before attempting this lab.
2. In the IDE terminal pane, enter the following command...

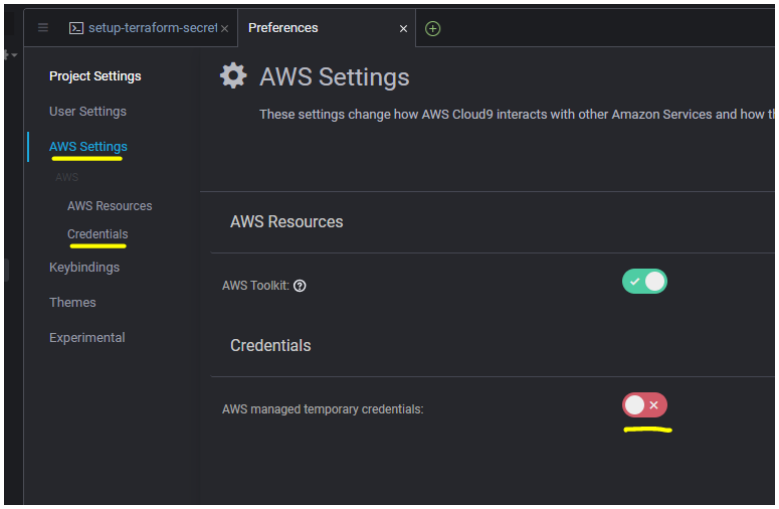
   cd ~/environment/aws-tf-int/labs/03
3. This shifts your current working directory to labs/03. Ensure all commands are executed in this directory
4. Close any open files and use the Explorer pane to navigate to and open the labs/03 folder.

Step 1: Administrators Set Up AWS Secrets Manager and IAM User

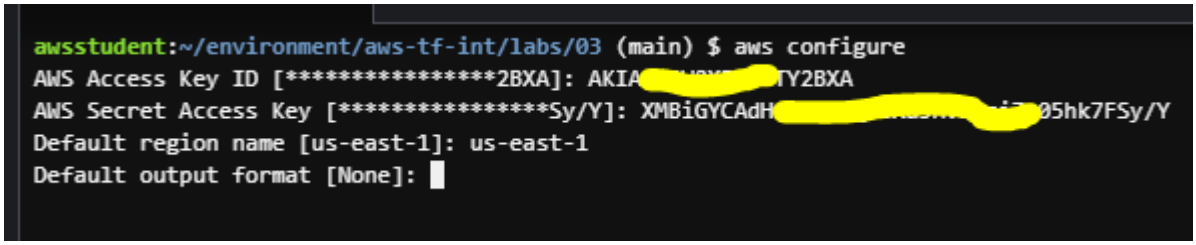
Stage 1: Disable Cloud9 temporary credentials and assign explicit credentials

1. 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...





2. 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)....



3. 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

Stage 2: Defining Variables and Creating an IAM User

Administrators will create an IAM User and store credentials securely in AWS Secrets Manager:

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

   chmod +x setup-terraform-secrets.sh

   chmod +x retrieve-aws-credentials.sh

3. Execute the script: **./setup-terraform-secrets.sh**
4. Run **aws secretsmanager get-secret-value --secret-id terraform-aws-credentials**
5. Switch to the Console and ensure the IAM user and secret are created successfully.

Step 2: Developers Retrieve IAM Credentials from AWS Secrets Manager

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

Stage 1: Define Parameters and Secure Retrieval Function

1. Examine the script **retrieve-aws-credentials.sh** in **aws-tf-int/labs/03**
2. Execute the script: **./retrieve-aws-credentials.sh**
3. Run **aws secretsmanager get-secret-value --secret-id terraform-aws-credentials**
4. Run: **echo $AWS_ACCESS_KEY_ID**
5. Check stored credentials: **cat aws_credentials.env**
6. Reload credentials in a new session: **source aws_credentials.env**

Stage 2: Verify Authentication & Run Terraform

Developers now use Terraform with the retrieved AWS credentials.

1. Update line 6 of **main.tf** with a unique S3 bucket name of your choice and save the change.
2. Ensure AWS credentials are loaded: **source aws_credentials.env**
3. Initialize Terraform: **terraform init**
4. Test infrastructure deployment: **terraform plan**
5. Deploy the infrastructure: **terraform apply**
6. Verify the AWS resources in the AWS Console.

Lab clean-up

1. Run **terraform destroy** and confirm with **yes**
2. Switch to the portal and manually delete resource group RG1

Congratulations, you have completed this lab

Alternative: Using IAM Roles Instead of IAM Users

If developers prefer to use IAM Roles instead of IAM Users, follow these steps:

1. Create an IAM Role:

   aws iam create-role --role-name TerraformRole --assume-role-policy-document file://trust-policy.json

2. The trust-policy.json file should contain:

```
{
 "Version": "2012-10-17",
 "Statement": [
  {
   "Effect": "Allow",
   "Principal": {
    "AWS": "arn:aws:iam::<ACCOUNT_ID>:user/TerraformUser"
   },
   "Action": "sts:AssumeRole"
  }
 ]
}
```

3. Attach Secrets Manager Read Policy:

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

4. Assume the Role Before Running Terraform:

   CREDENTIALS=$(aws sts assume-role --role-arn arn:aws:iam::<ACCOUNT_ID>:role/TerraformRole --role-session-name TerraformSession)

```
export AWS_ACCESS_KEY_ID=$(echo $CREDENTIALS | jq -r
'.Credentials.AccessKeyId')

export AWS_SECRET_ACCESS_KEY=$(echo $CREDENTIALS | jq -r
'.Credentials.SecretAccessKey')

export AWS_SESSION_TOKEN=$(echo $CREDENTIALS | jq -r
'.Credentials.SessionToken')
```

5. Using IAM Roles provides **temporary credentials**, which enhances security but requires assuming the role before each Terraform run.