AWS Lab 05: Checks and Validations – Walkthrough

Overview

This lab demonstrates how to enforce resource constraints and naming policies in AWS using Terraform.
You will use input validation, preconditions, postconditions, and IAM tagging best practices to validate compliance.

Lab Objectives

1. Apply input validation rules for variables (S3 bucket name, EC2 instance type).
2. Use precondition and postcondition checks in Terraform.
3. Enforce tag-based governance using IAM.
4. Perform compliance checks manually using AWS CLI.

Before You Begin

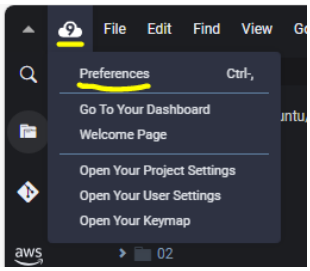
Ensure you have completed Lab 0 and configured the AWS CLI.
Navigate to the lab directory:

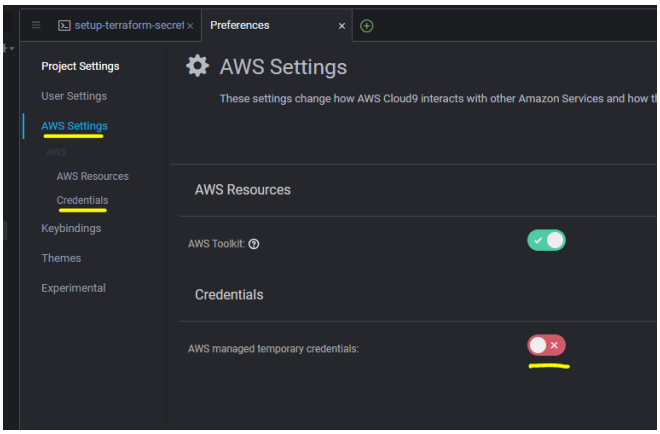
```
cd /aws-tf-int/labs/05
```

Disable Cloud9 temporary credentials.

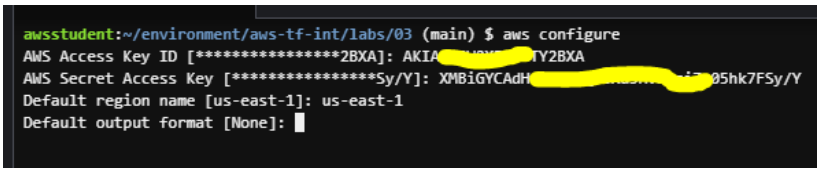
Note: You only need to complete this section if you have not completed Lab3 today.

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





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)...



Terraform Files

- main.tf: Deploys an S3 bucket, IAM role/profile, and EC2 instance.
- variables.tf: Declares variables with validation conditions.
- terraform.tfvars: Supplies input values.
- outputs.tf: Displays deployment metadata after apply.

Step 1: Validate IAM Policies

Use the following AWS CLI commands to inspect IAM policy attachments:

aws iam list-attached-user-policies --user-name <your-username>
aws iam list-attached-role-policies --role-name <your-role>

Step 2: Initialize and Deploy

Update line 1 in **terraform.tfvars** with a lowercase, unique S3 bucket name and save the changes.

Run the following commands:

terraform init
terraform plan
terraform apply
terraform output

Step 3: Test Validation Failures

Edit terraform.tfvars and try:

1. Invalid bucket name:
s3_bucket_name = "Invalid_Bucket_Name!"

Expected error: "S3 bucket name must be lowercase, alphanumeric or hyphens, 3-63 chars."

2. Invalid instance type:
instance_type = "m5.large"

Expected error: "Only approved instance types (t2.micro, t3.micro, t3.small) are allowed."

Step 4: Postcondition Check

In main.tf, a postcondition is defined on the IAM role resource to verify that the name matches:

```
lifecycle {
  postcondition {
    condition     = self.name == var.iam_role_name
    error_message = "IAM Role name does not match the expected value."
  }
}
```

Step 5: Clean Up

Run **terraform destroy** to remove all deployed resources.

References

- Terraform Input Validation: https://developer.hashicorp.com/terraform/language/values/variables

- Terraform Preconditions: https://developer.hashicorp.com/terraform/language/expressions/preconditions-and-postconditions

- AWS IAM Policies: https://docs.aws.amazon.com/IAM/latest/UserGuide/access_policies.html

- AWS Tag Policies: https://docs.aws.amazon.com/organizations/latest/userguide/orgs_manage_policies_tag-policies.html

Appendix: Breakdown of main.tf

Provider Block

```
provider "aws" {
 region = "us-east-1"
}
```

Sets the AWS region for Terraform operations.

S3 Bucket Resource

```
resource "aws_s3_bucket" "lab_bucket" {
 bucket = var.s3_bucket_name

tags = {
   Name       = var.s3_bucket_name
   Environment = "Lab05"
 }
}
```

Creates a tagged S3 bucket. Bucket name must meet validation rules.

IAM Role Definition

```
resource "aws_iam_role" "lab_role" {
 name = var.iam_role_name

assume_role_policy = jsonencode({
   Version = "2012-10-17",
   Statement = [{
    Action = "sts:AssumeRole",
    Effect = "Allow",
    Principal = {
     Service = "ec2.amazonaws.com"
    }
   }]
 })

lifecycle {
   postcondition {
```

```
    condition     = self.name == var.iam_role_name
    error_message = "IAM Role name does not match the expected
value."
   }
 }
}
```

Creates an IAM role trusted by EC2. Postcondition ensures correct role name.

IAM Instance Profile

```
resource "aws_iam_instance_profile" "lab_profile" {
 name = "Lab05InstanceProfile"
 role = aws_iam_role.lab_role.name
}
```

Wraps the IAM role into a profile for EC2 use.

EC2 Instance

```
resource "aws_instance" "lab_instance" {
 ami         = var.ami_id
 instance_type = var.instance_type
 iam_instance_profile = aws_iam_instance_profile.lab_profile.name

tags = {
   Name = "Lab05-Instance"
 }
}
```

Deploys an EC2 instance with validated AMI, instance type, and IAM role.