

Terraform AWS IAM Labs - Student Guide

Overview

This guide covers two essential labs from KodeKloud's Terraform AWS course:

1. **Lab 1:** AWS CLI and IAM - Understanding AWS CLI basics and IAM management
2. **Lab 2:** IAM with Terraform - Deploying AWS IAM resources using Infrastructure as Code

Both labs use **LocalStack** to simulate AWS services locally, providing a safe learning environment without AWS costs.

Prerequisites

- Basic understanding of command line interface
 - Familiarity with AWS IAM concepts (users, groups, policies)
 - Text editor for creating configuration files
-

Lab 1: AWS CLI and IAM

Learning Objectives

- Configure and use AWS CLI
- Understand LocalStack for AWS simulation
- Create and manage IAM users, groups, and policies
- Apply best practices for IAM permissions

Lab Environment Setup

LocalStack Configuration

- **Service Endpoint:** `http://aws:4566`
- **Region:** `us-east-1`
- **Access Key:** `foo`
- **Secret Key:** `bar`

Important Note

All AWS CLI commands in this lab must include the LocalStack endpoint:

```
bash

--endpoint http://aws:4566
```

Section 1: AWS CLI Basics

1.1 Check AWS CLI Version

```
bash

aws --version
```

Expected Output: `aws-cli/2.18.12 Python/3.12.6 Linux/5.15.0-1083-gcp exe/x86_64.ubuntu.22`

1.2 View AWS Configuration

```
bash

# Check configured region
aws configure get region

# Check access key
aws configure get aws_access_key_id

# Check secret key
aws configure get aws_secret_access_key
```

Section 2: IAM Commands Reference

2.1 Essential IAM Commands

```
bash
```

List all IAM users

```
aws iam list-users
```

Create a new IAM user

```
aws iam create-user --user-name <username>
```

Create an IAM group

```
aws iam create-group --group-name <groupname>
```

Add user to group

```
aws iam add-user-to-group --user-name <username> --group-name <groupname>
```

Attach policy to user

```
aws iam attach-user-policy --user-name <username> --policy-arn <policy-arn>
```

Attach policy to group

```
aws iam attach-group-policy --group-name <groupname> --policy-arn <policy-arn>
```

Section 3: Hands-on Exercises

Exercise 3.1: Initial Setup and User Creation

1. Test LocalStack connectivity:

```
bash
```

This will fail without LocalStack endpoint

```
aws iam list-users
```

This will work with LocalStack endpoint

```
aws --endpoint http://aws:4566 iam list-users
```

2. Create a new user named 'mary':

```
bash
```

```
aws iam create-user --user-name mary --endpoint http://aws:4566
```

Exercise 3.2: User Permission Management

1. Grant mary administrative access:

```
bash
```

```
aws iam attach-user-policy \  
--user-name mary \  
--policy-arn arn:aws:iam::aws:policy/AdministratorAccess \  
--endpoint http://aws:4566
```

Exercise 3.3: Group Management

1. Create a developer group:

```
bash  
  
aws iam create-group \  
--group-name project-sapphire-developers \  
--endpoint http://aws:4566
```

2. Add users to the group:

```
bash  
  
aws iam add-user-to-group \  
--user-name jack \  
--group-name project-sapphire-developers \  
--endpoint http://aws:4566  
  
aws iam add-user-to-group \  
--user-name jill \  
--group-name project-sapphire-developers \  
--endpoint http://aws:4566
```

3. Attach EC2 permissions to the group:

```
bash  
  
aws iam attach-group-policy \  
--group-name project-sapphire-developers \  
--policy-arn arn:aws:iam::aws:policy/AmazonEC2FullAccess \  
--endpoint http://aws:4566
```

Key Takeaways from Lab 1

- LocalStack requires endpoint specification for all AWS CLI commands
- IAM permissions can be granted directly to users or through groups
- Group-based permissions provide better scalability and management
- AWS managed policies provide pre-configured permission sets

Lab 2: IAM with Terraform

Learning Objectives

- Configure Terraform AWS provider for LocalStack
- Create IAM resources using Terraform configuration
- Use variables and meta-arguments for scalable infrastructure
- Apply Terraform best practices for IAM management

Section 1: Terraform Configuration Setup

1.1 Project Structure

```
/root/terraform-projects/IAM/  
├── provider.tf  
├── variables.tf  
├── iam-user.tf  
└── .terraform.lock.hcl
```

1.2 Provider Configuration

File: `provider.tf`

```
hcl  
  
provider "aws" {  
    region          = "us-east-1"  
    skip_credentials_validation = true  
    skip_metadata_api_check   = true  
    skip_requesting_account_id = true  
    access_key         = "foo"  
    secret_key         = "bar"  
    s3_use_path_style    = true  
  
    endpoints {  
        iam = "http://aws:4566"  
    }  
}
```

Key Configuration Notes:

- `skip_credentials_validation = true` - Required for LocalStack
- `endpoints` block - Directs IAM calls to LocalStack
- Additional skip options prevent AWS API calls during planning

1.3 Variables Configuration

File: `variables.tf`

```
hcl

variable "project-sapphire-users" {
  type = list(string)
  default = ["mary", "jack", "jill", "mack", "buzz", "mater"]
}
```

Section 2: Resource Configuration

2.1 Basic IAM User Resource

File: `iam-user.tf` (Initial Version)

```
hcl

resource "aws_iam_user" "users" {
  name = "mary"
}
```

2.2 Scalable IAM User Resource with Count

File: `iam-user.tf` (Final Version)

```
hcl

resource "aws_iam_user" "users" {
  count = length(var.project-sapphire-users)
  name = var.project-sapphire-users[count.index]
}
```

Section 3: Terraform Workflow

3.1 Initialization

```
bash
```

```
terraform init
```

Purpose: Downloads required providers and sets up working directory

3.2 Planning

```
bash  
  
terraform plan
```

Purpose: Shows what resources will be created/modified/destroyed

3.3 Application

```
bash  
  
terraform apply
```

Purpose: Executes the planned changes

Section 4: Hands-on Exercises

Exercise 4.1: Basic Setup

1. Create initial user resource:

- Create `(iam-user.tf)` with single user configuration
- Run `(terraform init)`
- Observe initialization process

2. Troubleshoot provider configuration:

- Run `(terraform plan)` without provider configuration
- Understand error messages
- Create `(provider.tf)` with proper configuration

Exercise 4.2: Deploy Single User

1. Plan and apply single user:

```
bash  
  
terraform plan  
terraform apply
```

2. Verify user creation:

```
bash  
  
aws --endpoint http://aws:4566 iam list-users
```

Exercise 4.3: Scale with Variables and Count

1. Review variables file:

- Examine `variables.tf` structure
- Understand list data type
- Note default values

2. Update resource configuration:

- Modify `iam-user.tf` to use count meta-argument
- Reference variable with proper syntax

3. Apply scaled configuration:

```
bash  
  
terraform apply
```

- Observe Terraform's handling of existing resources
- Note the creation of additional users

Common Terraform Patterns

Using Count Meta-Argument

```
hcl  
  
resource "aws_iam_user" "users" {  
  count = length(var.user_list)  
  name = var.user_list[count.index]  
}
```

Using Variables

```
hcl
```



```
variable "user_list" {  
  type    = list(string)  
  description = "List of IAM users to create"  
  default  = ["user1", "user2", "user3"]  
}
```

Using Functions

- `length()` - Returns the length of a list
- `var.variable_name` - References a variable
- `count.index` - Current iteration index in count loop

Section 5: Best Practices

5.1 Configuration Organization

- Use separate files for different resource types
- Keep provider configuration in `provider.tf`
- Define variables in `variables.tf`
- Use descriptive resource and variable names

5.2 LocalStack Specific Considerations

- Always include endpoint configuration for mock services
- Use skip options to prevent unnecessary AWS API calls
- Test configurations thoroughly in mock environment

5.3 IAM Security Best Practices

- Use groups for permission management
- Apply principle of least privilege
- Regularly review and audit permissions
- Use AWS managed policies when possible

Key Takeaways from Lab 2

- Terraform provides Infrastructure as Code capabilities for AWS resources
- Provider configuration is essential for connecting to AWS (or LocalStack)
- Variables and meta-arguments enable scalable, maintainable configurations

- The terraform workflow (init, plan, apply) ensures controlled infrastructure changes
 - LocalStack provides excellent testing environment for AWS configurations
-

Troubleshooting Guide

Common Issues and Solutions

1. "Invalid provider configuration" error

- **Cause:** Missing or incomplete provider block
- **Solution:** Ensure provider.tf contains all required arguments

2. "Invalid AWS Region" error

- **Cause:** Region not specified in provider configuration
- **Solution:** Add region argument to provider block

3. "InvalidClientTokenId" error

- **Cause:** Missing LocalStack endpoint in CLI commands
- **Solution:** Add `--endpoint http://aws:4566` to all AWS CLI commands

4. Terraform plan fails with network errors

- **Cause:** LocalStack service not running or incorrect endpoint
- **Solution:** Verify LocalStack is accessible at <http://aws:4566>

Verification Commands

```
bash

# Verify LocalStack connectivity
curl http://aws:4566/health

# List all created users
aws --endpoint http://aws:4566 iam list-users

# Check Terraform state
terraform show
terraform state list
```

Additional Resources

- [AWS CLI Command Reference](#)

- [Terraform AWS Provider Documentation](#)
 - [LocalStack Documentation](#)
 - [AWS IAM Best Practices](#)
-

Lab Completion Checklist

Lab 1: AWS CLI and IAM

- ☐ Successfully executed AWS CLI commands with LocalStack
- ☐ Created individual IAM users
- ☐ Created and managed IAM groups
- ☐ Attached policies to users and groups
- ☐ Verified permissions and group memberships

Lab 2: IAM with Terraform

- ☐ Configured Terraform AWS provider for LocalStack
- ☐ Created basic IAM user resource
- ☐ Implemented variables for configuration management
- ☐ Used count meta-argument for resource scaling
- ☐ Successfully applied Terraform configuration
- ☐ Verified resource creation in AWS/LocalStack

Congratulations! You have successfully completed both labs and gained practical experience with AWS IAM management using both CLI and Infrastructure as Code approaches.