

AWS Infrastructure Automation

Terraform and CloudFormation Project

PROG 8870 - Final Project

Spring 2025

Ray Chen

Project Overview

Multi-Service AWS Infrastructure Deployment

Project Scope

- **Infrastructure as Code (IaC)** using Terraform and CloudFormation
- **Multi-service environment** with S3, EC2, and RDS
- **Best practices** for modularity and reusability
- **Live demo** showcasing resource provisioning

Technology Stack

Terraform

- HashiCorp's Infrastructure as Code tool
- Declarative configuration
- State management and versioning

CloudFormation




- AWS native IaC service
- YAML-based templates
- Integrated with AWS services

Architecture Overview




- **VPC:** Custom network with CIDR 10.0.0.0/16
- **Subnets:** Multi-AZ deployment (us-east-1a, us-east-1b)
- **Services:** S3, EC2, RDS across availability zones

Task 1: S3 Bucket Setup

Terraform Requirements





-  4 Private S3 Buckets
-  No public access
-  Versioning enabled (Bonus)

CloudFormation Requirements




-  3 Private S3 Buckets
-  PublicAccessBlockConfiguration
-  Versioning enabled (Bonus)

Task 2: VPC and EC2 Instance

Terraform Implementation




-  Custom VPC with dynamic variables
-  EC2 instance with public IP
-  SSH access on port 22
-  Dynamic AMI selection

CloudFormation Implementation




-  YAML-based configuration
-  IGW and Route Tables
-  Public IP output

Task 3: RDS Instance Deployment

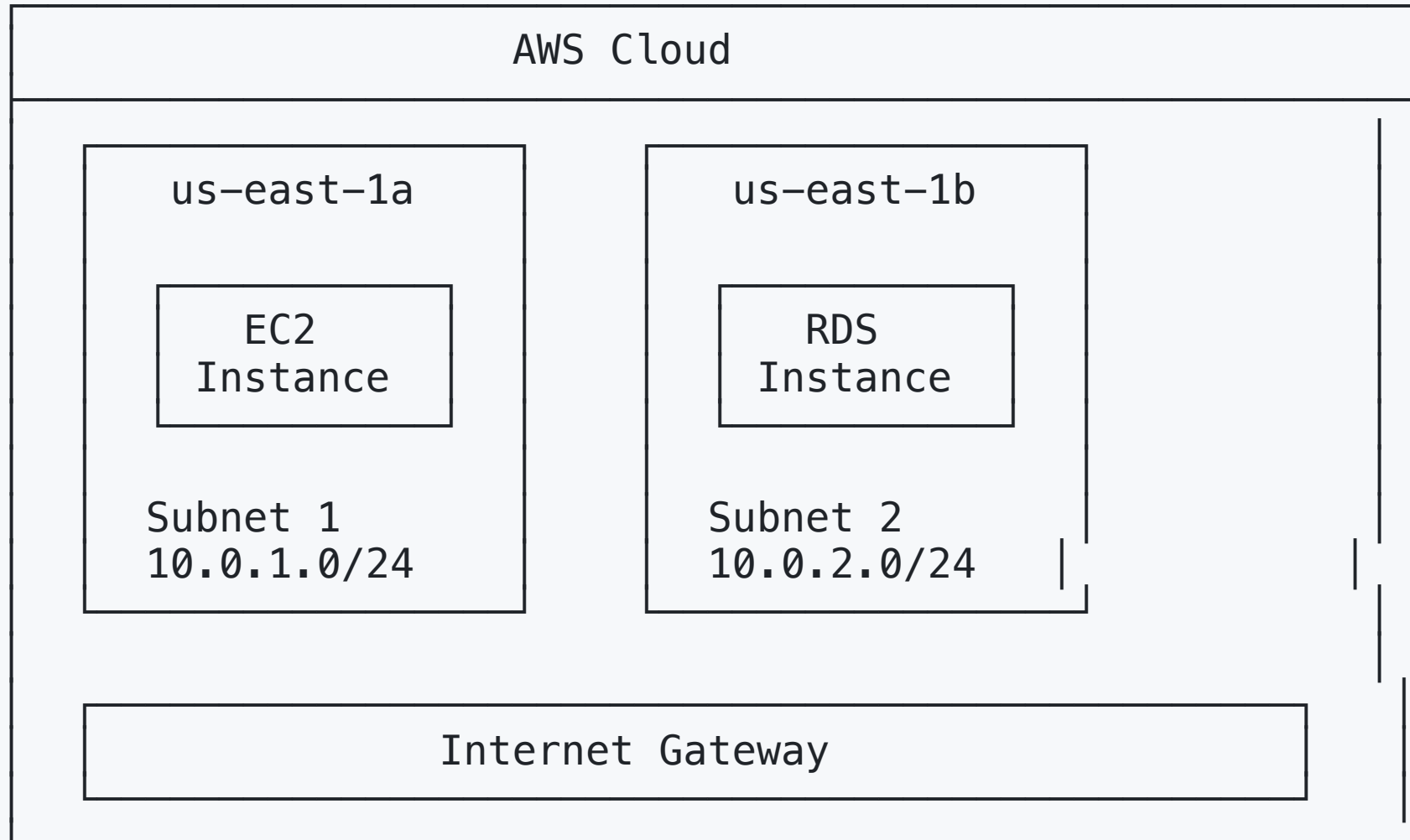
Terraform Features

-  MySQL RDS with db.t3.micro
-  Dynamic database credentials
-  Dedicated DB Subnet Group

CloudFormation Features

-  YAML template deployment
-  Public access enabled
-  Security groups for port 3306

Multi-AZ Design Architecture



Terraform Configuration Structure

```
@Terraform/  
├── main.tf          # Main infrastructure configuration  
├── variables.tf     # Variable definitions  
├── terraform.tfvars # Variable values  
├── provider.tf      # AWS provider configuration  
├── backend.tf       # State management  
└── terraform.tfstate # State file (local storage)
```

CloudFormation Template Structure

```
@CloudFormat/  
├── ec2-vpc.yaml      # EC2 and VPC resources  
├── rds-instance.yaml # RDS database configuration  
└── s3-buckets.yaml  # S3 bucket resources
```

Deployment Order - CloudFormation

 **Critical:** Stacks must be deployed in this specific order due to dependencies:

1. **EC2-VPC Stack** (creates VPC and subnets)
2. **RDS Stack** (depends on subnet outputs)
3. **S3 Stack** (independent, can deploy anytime)

Terraform Deployment Commands

```
cd @Terraform
terraform init      # Initialize backend
terraform plan      # Preview changes
terraform apply     # Deploy infrastructure
terraform destroy   # Cleanup resources
```

CloudFormation Deployment Commands

1. Deploy EC2-VPC Stack

```
aws cloudformation create-stack \  
  --stack-name ray-ec2-vpc-stack \  
  --template-body file:///@CloudFormat/ec2-vpc.yaml \  
  --parameters ParameterKey=RayProjectName,ParameterValue=ray-infrastructure
```

2. Deploy RDS Stack

```
aws cloudformation create-stack \  
  --stack-name ray-rds-stack \  
  --template-body file:///@CloudFormat/rds-instance.yaml
```

3. Deploy S3 Stack

```
aws cloudformation create-stack \  
  --stack-name ray-s3-stack \  
  --template-body file:///@CloudFormat/s3-buckets.yaml
```

Best Practices Implemented

✓ Dynamic Configuration

- Variables files (variables.tf, .tfvars)
- CloudFormation Parameters
- No hardcoded values

✓ Modularity

- Separate templates for each service
- Reusable configurations
- Clean code structure

Best Practices Implemented (Cont.)

Security

- Private S3 buckets
- Proper security groups
- VPC isolation

State Management

- Local Terraform state
- CloudFormation stack management
- Resource tracking

Key Features & Challenges

Features

- Multi-AZ architecture
- Automated deployment
- Comprehensive documentation

Challenges

- CloudFormation dependency management
- RDS multi-AZ requirements
- Proper subnet group configuration

Live Demo Agenda

1. Terraform Deployment

- `terraform init` → `terraform plan` → `terraform apply`
- Show resource creation in AWS Console

2. CloudFormation Deployment

- Deploy stacks in order
- Verify resource creation

3. Resource Verification

- S3 buckets with versioning
- EC2 instance with public IP
- RDS instance running

Resource Verification Checklist

S3 Buckets

- 4 Terraform buckets created
- 3 CloudFormation buckets created
- Versioning enabled on all
- No public access

EC2 & VPC

- Custom VPC created
- EC2 instance running
- Public IP accessible
- SSH connectivity

Cleanup Commands

```
# Terraform Cleanup  
cd @Terraform  
terraform destroy
```

```
# CloudFormation Cleanup  
aws cloudformation delete-stack --stack-name ray-s3-stack  
aws cloudformation delete-stack --stack-name ray-rds-stack  
aws cloudformation delete-stack --stack-name ray-ec2-vpc-stack
```

Thank You!

Questions & Live Demo

GitHub Repository: [Your Repository URL]

Contact: [Your Information]