# AWS CloudFormation Task Documentation

This document provides a comprehensive walkthrough for the task of creating a public VPC, Subnet, EC2 instance, Elastic IP, Internet Gateway, and Security Group using AWS CloudFormation. The outputs and configurations are documented below as per the provided requirements.

## 1. CloudFormation Stack Creation

The CloudFormation stack was created using the following command:

$ aws cloudformation create-stack \  
> --stack-name MyCloudFormationStack \  
> --template-body file://cloudformation-template.yaml \  
> --region us-east-2

The stack creation was completed successfully with the following stack ID:

StackId: arn:aws:cloudformation:us-east-2:908027419603:stack/MyCloudFormationStack/d3171b50-abb2-11ef-969f-02250bef4841

## 2. Resources Created and Outputs

### 2.1 VPC Details

The VPC was created with the following details:

|  |  |  |
| --- | --- | --- |
| CidrBlock | IsDefault | VPCId |
| 10.0.0.0/16 | False | vpc-0287ad2c549dc6947 |

### 2.2 Subnet Details

The Subnet created under the VPC has the following details:

|  |  |  |  |
| --- | --- | --- | --- |
| CidrBlock | MapPublicIpOnLaunch | SubnetId | VpcId |
| 10.0.1.0/24 | True | subnet-03e46ce9420b1944c | vpc-0287ad2c549dc6947 |

### 2.3 Internet Gateway

The Internet Gateway was attached to the VPC with the following details:

|  |  |
| --- | --- |
| InternetGatewayId | VpcId |
| igw-016697cdf897a6949 | vpc-0287ad2c549dc6947 |

### 2.4 Security Group Details

A security group was created to allow SSH access with the following details:

|  |  |  |  |
| --- | --- | --- | --- |
| GroupName | Description | GroupId | VpcId |
| MySecurityGroup | Allow SSH access | sg-007947a9114c0a711 | vpc-0287ad2c549dc6947 |

Inbound Rule: TCP 22 from 0.0.0.0/0

Note: Allowing SSH access from 0.0.0.0/0 is not recommended for production environments. It is better to restrict access to specific IP ranges.

### 2.5 EC2 Instance Details

The EC2 instance was successfully launched with the following details:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| InstanceId | PublicIpAddress | SubnetId | SecurityGroups | State |
| i-0469ff1e1cd670ef6 | 3.16.48.236 | subnet-03e46ce9420b1944c | MyCloudFormationStack-MySecurityGroup-gWh2hSI7uspq | running |

## 3. Verification and Connectivity

Connectivity to the EC2 instance was verified using SSH and ping commands.

$ ssh -i "MyKeyPair.pem" ec2-user@3.16.48.236  
$ ping google.com  
64 bytes from ord38s28-in-f14.1e100.net (142.250.191.110): icmp\_seq=1 ttl=55 time=8.34 ms

## 4. Route Table Configuration

The route table configured for the VPC includes the following routes:

|  |  |  |
| --- | --- | --- |
| DestinationCidrBlock | GatewayId | State |
| 10.0.0.0/16 | local | active |
| 0.0.0.0/0 | igw-016697cdf897a6949 | active |

## 5. Terraform Comparison

**Pros:**

1. **Multi-cloud Support**: Terraform works across various cloud providers, not just AWS.
2. **State Management**: Tracks infrastructure changes with a state file, allowing better control and drift detection.
3. **Reusability**: Supports reusable modules, simplifying complex setups.
4. **Plan Feature**: Allows previewing changes before applying them.

**Cons:**

1. **State Management Overhead**: Requires careful handling of the state file (e.g., storing in S3 with locking).
2. **Learning Curve**: More effort required for teams familiar only with CloudFormation.
3. **Delayed AWS Features**: AWS features are available on CloudFormation before Terraform.
4. **Operational Complexity**: External dependencies like backends and state locking add complexity