

# Deploying VPC, EC2, ALB, and Auto Scaling Group using Terraform

## Overview

This document provides a step-by-step guide to deploying a cloud infrastructure setup using Terraform. The setup includes:

- A Virtual Private Cloud (VPC)
- An Elastic Load Balancer (ALB)
- An Auto Scaling Group (ASG) with EC2 instances

Terraform is used to automate the infrastructure provisioning process.

## Prerequisites

Before proceeding, ensure the following requirements are met:

- Terraform installed on your local machine
- AWS CLI configured with appropriate credentials
- A working AWS account with permissions to create VPCs, EC2 instances, ALBs, and ASGs

## Architecture Overview

The infrastructure consists of:

1. A VPC with public and private subnets
2. An Internet Gateway to allow external access
3. A Load Balancer for distributing traffic across EC2 instances
4. An Auto Scaling Group (ASG) to manage instance scaling
5. EC2 instances running Apache with a simple HTML page

## Terraform File Structure

The project consists of the following Terraform files:

- main.tf - Defines the primary infrastructure resources
- variables.tf - Contains input variables for customization
- output.tf - Specifies output values for easy reference
- alb.tf - Configures the Application Load Balancer
- ec2.tf - Defines EC2 instance configurations

- userdata.sh - A script to install and configure Apache on EC2 instances

## Deployment Steps

### Step 1: Initialize Terraform

Navigate to the directory containing Terraform files and initialize Terraform:

terraform init

```
PS I:\Devops Projects\Project-x> terraform init
Initializing the backend...
Initializing modules...
Initializing provider plugins...
- Reusing previous version of hashicorp/aws from the dependency lock file
- Using previously-installed hashicorp/aws v5.91.0

Terraform has been successfully initialized!

You may now begin working with Terraform. Try running "terraform plan" to see
any changes that are required for your infrastructure. All Terraform commands
should now work.

If you ever set or change modules or backend configuration for Terraform,
rerun this command to reinitialize your working directory. If you forget, other
commands will detect it and remind you to do so if necessary.
PS I:\Devops Projects\Project-x> terraform plan
```

### Step 2: Plan the Infrastructure

Review the resources that will be created:

terraform plan

### Step 3: Apply the Configuration

Deploy the infrastructure:

terraform apply -auto-approve

```
+ tags_all = {
+   "Environment" = "production"
+   "Name"        = "stylish-website-vpc"
+   "Project"     = "stylish-website"
+   "Terraform"  = "true"
+ }
}

Plan: 38 to add, 0 to change, 0 to destroy.

Changes to Outputs:
+ alb_dns_name = (known after apply)

Note: You didn't use the -out option to save this plan, so Terraform can't guarantee to take exactly these actions if you run "terraform apply" now.
PS I:\Devops Projects\Project-x> terraform apply -auto-approve
```

### Step 4: Retrieve Outputs

After deployment, retrieve essential output values such as Load Balancer DNS:

terraform output

#### Outputs:

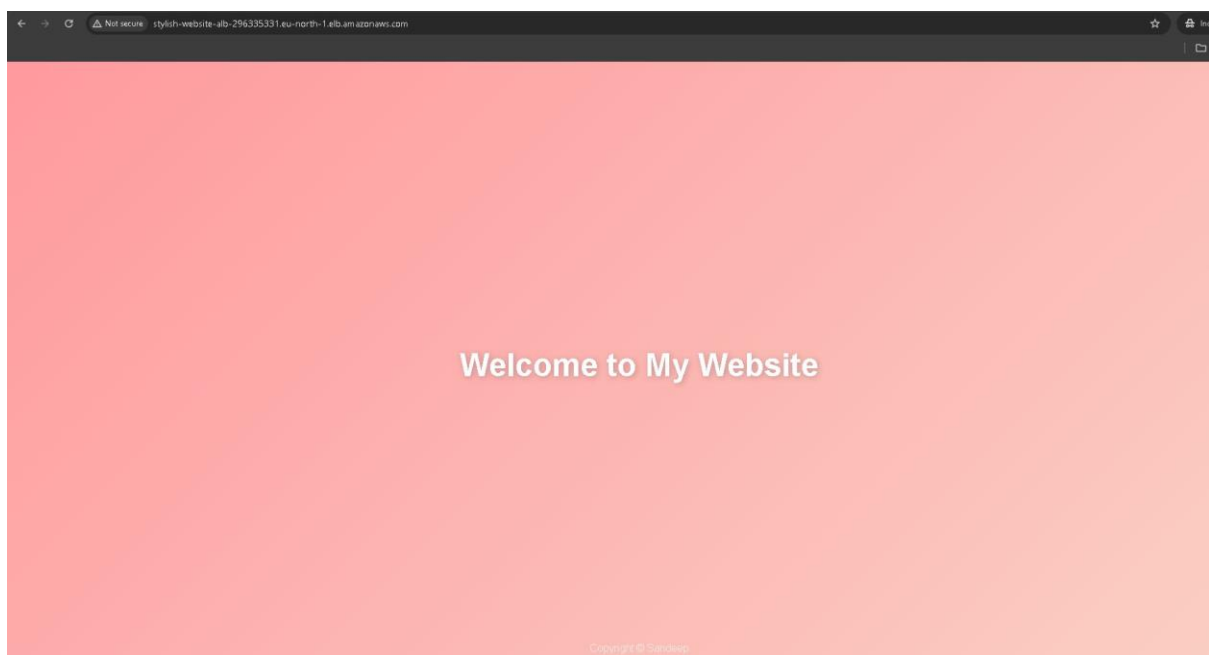
```
alb_dns_name = "stylish-website-alb-296335331.eu-north-1.elb.amazonaws.com"
PS I:\Devops Projects\Project-x> 
```

### Step 5: Verify the Deployment

- Navigate to the Load Balancer URL from the Terraform output.
- The webpage should display "Welcome to My Website."

Access the website using the ALB DNS

<http://stylish-website-alb-296335331.eu-north-1.elb.amazonaws.com>



### Step 6: Auto Scaling Group Validation

- Delete two instances from the Auto Scaling Group (ASG), and verify that it automatically spins up two new instances.

## website-nodes

website-nodes Capacity overview

Desired capacity

2

Scaling limits (Min - Max)

1 - 4

Desired capacity type

Units (number of instances)

Status

-

Date created

Thu Mar 20 2025 15:53:40 GMT+0530 (India Standard Time)

Details

Integrations - new

Automatic scaling

Instance management

Instance refresh

Activity

Monitoring

Instances (4)

Filter instances

Instance ID

Lifecycle

Instance type

Weighted capac...

Launch templat...

Availability Zone

Health status

Protected from

<input type="checkbox"/>	i-015e52634521722e4	Pending	t3.small	-	website-node202503201	eu-north-1c	Healthy	
<input type="checkbox"/>	i-03867f102da8a3892	Terminating	t3.small	-	website-node202503201	eu-north-1a	Unhealthy	
<input type="checkbox"/>	i-0512b9424f4e4c77c	Terminating	t3.small	-	website-node202503201	eu-north-1c	Unhealthy	
<input type="checkbox"/>	i-0abce22e9124671	InService	t3.small	-	website-node202503201	eu-north-1a	Healthy	

Ensure that the Target Groups update dynamically with the newly created instances.

Target groups

web-20250320101115650400000006

Details

Target type

Instance

Protocol : Port

HTTP: 80

Protocol version

HTTP1

VPC

vpc-0c0db6baa772ae8595

IP address type

IPv4

Load balancer

styl:sh-website-alb

2

2

0

0

0

0

Total targets

Healthy

Unhealthy

Unused

Initial

Draining

0 Anomalous

Distribution of targets by Availability Zone (AZ)

Select values in this table to see corresponding filters applied to the Registered targets table below

Targets

Monitoring

Health checks

Attributes

Tags

Registered targets (2)

Anomaly mitigation: Not applicable

Deregister

Register targets

Target groups route requests to individual registered targets using the protocol and port number specified. Health checks are performed on all registered targets according to the target group's health check settings. Anomaly detection is automatically applied to HTTP/HTTPS target groups with at least 3 healthy targets.

Filter targets

Instance ID

Name

Port

Zone

Health status

Health status details

Administrative o...

Override details

Launch...

Anomaly

<input type="checkbox"/>	i-015e52634521722e4	website-node	80	eu-north-1c (eu...	Healthy	-	No override	No override (s current...	March 20, ...	Normal
<input type="checkbox"/>	i-0abce22e9124671	website-node	80	eu-north-1a (e...	Healthy	-	No override	No override (s current...	March 20, ...	Normal

## Cleanup

To destroy the infrastructure when no longer needed:

terraform destroy -auto-approve

## Conclusion

This guide helps deploy a scalable, highly available setup using Terraform. It ensures efficient traffic distribution using ALB and dynamic resource allocation through ASG.