# AWS Web Hosting

# Objective

Deploy a basic web application using two EC2 instances running Apache2 behind an Application Load Balancer (ALB), all provisioned using Terraform.

---------------------------------------------------------------------------------------------------------------------------------**Steps Taken & Design Choices**

## 1. VPC & Networking

* Created a custom VPC (10.0.0.0/16) with 2 public subnets in different Availability Zones (AZs) for high availability.
* Configured internet gateway and route table for outbound access.

## 2. EC2 Setup

* Launched 2 EC2 instances (Amazon Linux 2) using t2.micro.
* Apache2 installed via user\_data.sh script.
* Website deployed from Tooplate Mini Finance template.
* EC2s registered with a target group.

## 3. Load Balancing

* ALB listens on port 80 and forwards to the target group.
* Health check on path / (HTTP).
* Security groups:
  + ALB: HTTP from 0.0.0.0/0
  + EC2: HTTP from ALB SG only.

## 4. Automation

* Entire infrastructure provisioned via Terraform.
* Apache and app setup automated via Bash script.

---------------------------------------------------------------------------------------------------------------------------------

**Public URL (ALB DNS)**

* <http://web-alb-1389464065.us-east-1.elb.amazonaws.com/>

---------------------------------------------------------------------------------------------------------------------------------

**Screenshots**

* EC2 instances running

A screenshot of a computer

AI-generated content may be incorrect.

* Apache2 running on each

A computer screen with white text

AI-generated content may be incorrect.

* ALB and Target Group configuration

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

* Your application in the browser (via ALB)

A screenshot of a computer

AI-generated content may be incorrect.

---------------------------------------------------------------------------------------------------------------------------------**Terraform code / Scripts Used**

## user\_data.sh

#!/bin/bash  
sudo yum install wget unzip httpd -y  
sudo systemctl start httpd  
sudo systemctl enable httpd  
mkdir -p /tmp/webfiles && cd /tmp/webfiles  
wget https://www.tooplate.com/zip-templates/2135\_mini\_finance.zip  
unzip 2135\_mini\_finance.zip  
sudo cp -r 2135\_mini\_finance/\* /var/www/html/  
sudo systemctl restart httpd  
rm -rf /tmp/webfiles

* **main.tf**

provider "aws" {

region = var.region

}

data "template\_file" "user\_data" {

template = file("${path.module}/user\_data.sh")

}

resource "aws\_vpc" "main" {

cidr\_block = var.vpc\_cidr

tags = { Name = "web-vpc" }

}

resource "aws\_internet\_gateway" "gw" {

vpc\_id = aws\_vpc.main.id

}

resource "aws\_subnet" "public" {

count = 2

vpc\_id = aws\_vpc.main.id

cidr\_block = var.public\_subnet\_cidrs[count.index]

availability\_zone = "${var.region}${element(["a", "b"], count.index)}"

map\_public\_ip\_on\_launch = true

tags = { Name = "public-${count.index}" }

}

resource "aws\_route\_table" "public\_rt" {

vpc\_id = aws\_vpc.main.id

}

resource "aws\_route" "internet" {

route\_table\_id = aws\_route\_table.public\_rt.id

destination\_cidr\_block = "0.0.0.0/0"

gateway\_id = aws\_internet\_gateway.gw.id

}

resource "aws\_route\_table\_association" "assoc" {

count = 2

subnet\_id = aws\_subnet.public[count.index].id

route\_table\_id = aws\_route\_table.public\_rt.id

}

resource "aws\_security\_group" "alb\_sg" {

name = "alb-sg"

vpc\_id = aws\_vpc.main.id

ingress {

from\_port = 80

to\_port = 80

protocol = "tcp"

cidr\_blocks = ["0.0.0.0/0"]

}

egress {

from\_port = 0

to\_port = 0

protocol = "-1"

cidr\_blocks = ["0.0.0.0/0"]

}

tags = { Name = "alb-sg" }

}

resource "aws\_security\_group" "web\_sg" {

name = "web-sg"

vpc\_id = aws\_vpc.main.id

ingress {

from\_port = 80

to\_port = 80

protocol = "tcp"

security\_groups = [aws\_security\_group.alb\_sg.id]

}

egress {

from\_port = 0

to\_port = 0

protocol = "-1"

cidr\_blocks = ["0.0.0.0/0"]

}

tags = { Name = "web-sg" }

}

resource "aws\_instance" "web" {

count = 2

ami = var.ami\_id

instance\_type = var.instance\_type

key\_name = var.key\_name

subnet\_id = aws\_subnet.public[count.index].id

vpc\_security\_group\_ids = [aws\_security\_group.web\_sg.id]

user\_data = data.template\_file.user\_data.rendered

tags = {

Name = "web-${count.index}"

}

}

resource "aws\_lb" "alb" {

name = "web-alb"

internal = false

load\_balancer\_type = "application"

security\_groups = [aws\_security\_group.alb\_sg.id]

subnets = aws\_subnet.public[\*].id

tags = {

Name = "web-alb"

}

}

resource "aws\_lb\_target\_group" "tg" {

name = "web-tg"

port = 80

protocol = "HTTP"

vpc\_id = aws\_vpc.main.id

health\_check {

path = "/"

protocol = "HTTP"

interval = 30

timeout = 5

healthy\_threshold = 2

unhealthy\_threshold = 2

}

tags = {

Name = "web-tg"

}

}

resource "aws\_lb\_target\_group\_attachment" "attach" {

count = 2

target\_group\_arn = aws\_lb\_target\_group.tg.arn

target\_id = aws\_instance.web[count.index].id

port = 80

}

resource "aws\_lb\_listener" "http" {

load\_balancer\_arn = aws\_lb.alb.arn

port = 80

protocol = "HTTP"

default\_action {

type = "forward"

target\_group\_arn = aws\_lb\_target\_group.tg.arn

}

}

* **variable.tf**

variable "region" {

description = "AWS region"

type = string

}

variable "ami\_id" {

description = "AMI ID for EC2"

type = string

}

variable "instance\_type" {

description = "EC2 instance type"

type = string

}

variable "key\_name" {

description = "Name of existing AWS key pair"

type = string

}

variable "vpc\_cidr" {

description = "CIDR for VPC"

type = string

}

variable "public\_subnet\_cidrs" {

description = "CIDRs for public subnets"

type = list(string)

}

* **terraform.tfvars**

region = "us-east-1"

ami\_id = "ami-0c7217cdde317cfec" # Ubuntu in us-east-1

instance\_type = "t2.micro"

key\_name = "ec2\_login"

vpc\_cidr = "10.0.0.0/16"

public\_subnet\_cidrs = ["10.0.1.0/24", "10.0.2.0/24"]

* **outputs.tf**

output "alb\_dns\_name" {

description = "DNS name of the ALB"

value = aws\_lb.alb.dns\_name

}