WHIZLABS 0 K

Create an Application Load Balancer to distribute the incoming traffic between two EC2 Instances using Terraform

Level: **Fundamental**

Amazon EC2 Amazon Web Services Elastic Load Balancing Terraform

Required Points	10
Lab Duration	01:00:00
Average Start time	Less than a minute

Start Lab →

Need help?

How to use Hands on Lab

Troubleshooting Lab

FAQs

Submit Feedback | Share

Lab Overview

Cloud Architect

Compute, Networking, Infrastructure

Lab Details

Privacy - Terms

1. This lab walks you through AWS Elastic Load Balancing. Elastic Load Balancing automatically distributes incoming application traffic across multiple Amazon EC2 instances in the cloud. In this lab, we will demonstrate elastic load balancing with 2 EC2 Instances using Terraform.
2. Duration: **1 hour**
3. AWS Region: **US East (N. Virginia) us-east-1.**

Introduction

What is Elastic Load Balancing?

- ELB is a service that automatically distributes incoming application traffic and scales resources to meet traffic demands.
- ELB helps in adjusting capacity according to incoming application and network traffic.
- ELB can be enabled within a single availability zone or across multiple availability zones to maintain consistent application performance.
- ELB offers features like:
- Detection of unhealthy EC2 instances.
- Spreading EC2 instances across healthy channels only.
- Centralized management of SSL certificates.
- Optional public key authentication.
- Support for both IPv4 and IPv6.
- ELB accepts incoming traffic from clients and routes requests to its registered targets.
- When an unhealthy target or instance is detected, ELB stops routing traffic to it and resumes only when the instance is healthy again.
- ELB monitors the health of its registered targets and ensures that the traffic is routed only to healthy instances.
- ELB's are configured to accept incoming traffic by specifying one or more **listeners**. A listener is a process that checks for connection requests.
- Listeners are configured with a protocol and port number from the client to the ELB and vice-versa i.e., back from ELB to the client.
- ELB supports 3 types of load balancers:
  - Application Load Balancers
  - Network Load Balancers

    - Classic Load Balancers
- Each load balancer is configured differently.
- For Application and Network Load Balancers, you register targets in target groups and route traffic to target groups.
- For Classic Load Balancers, you register instances with the load balancer.
- AWS recommends users to work with the Application Load Balancer to use multiple Availability Zones because if one availability zone fails, the load balancer can continue to route traffic to the next available one.
- We can have our load balancer be either internal or internet-facing.
- The nodes of an internet-facing load balancer have Public IP addresses, and the DNS name is publicly resolvable to the Public IP addresses of the nodes.
- Due to the point above, internet-facing load balancers can route requests from clients over the Internet.
- The nodes of an internal load balancer have only Private IP addresses, and the DNS name is publicly resolvable to the Private IP addresses of the nodes.
- Due to the point above, internal load balancers can only route requests from clients with access to the VPC for the load balancer.
- Both internet-facing and internal load balancers route requests to your targets using Private IP addresses.
- Your targets do not need Public IP addresses to receive requests from an internal or an internet-facing load balancer.

What is Terraform?

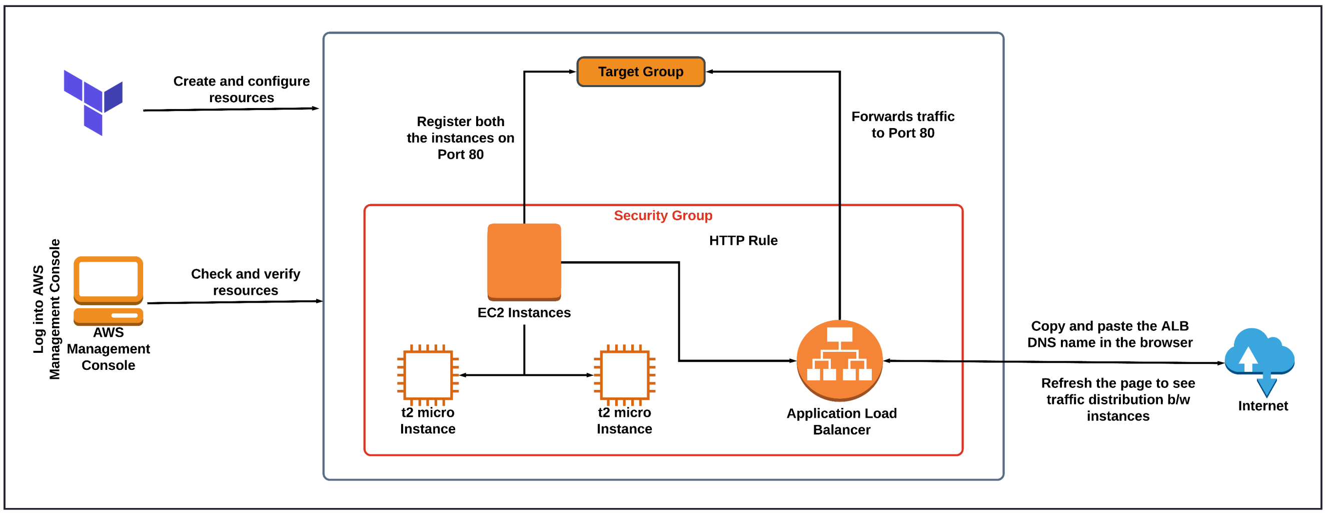
- It is an open-source IaaC (Infrastructure as a Code) software tool where you define and create resources using providers in the declarative configuration language example JSON.
- With Terraform, You can package and reuse the code in the form of modules.
- It supports a number of cloud infrastructure providers such as AWS, Azure, GCP, IBM Cloud, OCI, etc.
- Terraform has four major commands:
    - terraform init
    - terraform plan

- terraform apply
- terraform destroy

Prerequisite

- Install Terraform in your local machine using this official guide by Hashicorp.
  - To install Terraform using CLI, use this guide https://learn.hashicorp.com/tutorials/terraform/install-cli
  - To install Terraform by downloading, use this guide https://www.terraform.io/downloads.html
- Download and Install Visual Studio Code editor using this guide https://code.visualstudio.com/download

Architecture Diagram


Create and configure resources
Log into AWS Management Console
AWS Management Console
Check and verify resources
Target Group
Register both the instances on Port 80
Forwards traffic to Port 80
Security Group
HTTP Rule
EC2 Instances
t2 micro Instance
t2 micro Instance
Application Load Balancer
Copy and paste the ALB DNS name in the browser
Refresh the page to see traffic distribution b/w instances
Internet

Task Details

1. Sign in to AWS Management Console
2. Setup Visual Studio Code.
3. Create a Variables file.
4. Create EC2, ELB and its components in main.tf file
5. Create an Output file.
6. Confirm the installation of Terraform by checking the version.
7. Apply Terraform configurations.

8. Check the HTML page and traffic distribution.
9. Check the resources in AWS Console.
10. Validation of the lab.
11. Deleting AWS Resources.

Launching Lab Environment

1. To launch the lab environment, click on the **Start Lab** button.
2. Please wait until the cloud environment is provisioned. It will take less than a minute to provision.
3. Once the Lab is started, you will be provided with **IAM username**, **Password**, **Access Key**, and **Secret Access Key**.

Note : You can only start one lab at any given time

About Us Subscription Instructions and Guidelines FAQ's Contact Us