CREATION OF AWS TERRAFORM AUTOSCALING GROUP WITH ALB DEPLOYMENT

Mrs.P.Madhavi [2].K.V.Supriya

[1]CSE, NEC, Gudur

2Associate Professor,CSE Department, NEC, Gudur

Abstract: *In today's cloud computing landscape, the adoption of Infrastructure as Code (IaC) tools such as Terraform has revolutionized the way organizations manage and scale applications on platforms like Amazon Web Services (AWS). This essay explores the process of creating an AWS Auto Scaling Group (ASG) integrated with an Application Load Balancer (ALB) using Terraform, highlighting its significance, steps . AWS Auto Scaling Groups (ASGs) and Application Load Balancers (ALBs) are indispensable components in modern cloud architectures. An ASG automates the scaling of compute resources based on predefined conditions, ensuring applications can handle varying workloads efficiently while optimizing costs. ALBs distribute incoming traffic across multiple targets within AWS, enhancing application availability and fault tolerance by routing traffic only to healthy instances. Together, ASGs and ALBs form*
Terraform, developed by HashiCorp, is a powerful IaC tool that simplifies the provisioning and management of cloud infrastructure. It operates using declarative configuration files, enabling users to define and automate the deployment of infrastructure components such as compute instances, networking resources, and services across different environments. This approach ensures consistency and reproducibility in infrastructure management

1. ***Keywords:*** *AWS, EC2 Instance, Load balancer, Auto Scaling group, Deployment automation*

I. INTRODUCTION

In today's dynamic cloud computing landscape, the ability to deploy and manage scalable, resilient applications efficiently is paramount. Amazon Web Services (AWS) provides a robust platform for hosting applications, while Terraform offers powerful tools for automating infrastructure management through Infrastructure as Code (IaC). This combination enables teams to provision, manage, and scale resources with precision and consistency.

This document explores the implementation of an Auto Scaling Group (ASG) integrated with an Application Load Balancer (ALB) using Terraform on AWS. We will delve into the essential concepts of IaC, demonstrate how to define and deploy infrastructure using Terraform configuration files, and highlight the benefits of leveraging ASGs and ALBs for achieving high availability and scalability.

By the end of this exploration, you will have a clear understanding of how to orchestrate auto-scaling applications on AWS using Terraform, enhancing reliability and operational efficiency in cloud environments.

Load balancing plays a very important role in the networking technology, Load balancer comes into play when the user tries to connect to the server. Any requests made to the internet will reach the server in various paths. There may be N number of servers which are serving the purpose of the request but the request will go to only one depending upon the traffic. The traffic here refers to how busy server is, in responding the requests made by servers. Requests will be directed towards servers by one of the networks called Load balancer which balances the load of server. There are different types of load balancer which handles the traffic in different ways. Major types of Load balancer are Static and Dynamic Load balancer. Amazon EC2 instances, containers, and IP addresses can be automatically distributed by ELB to handle incoming application traffic. ELB can handle a single Availability Zone or a number of

Availability Zones. ELB has features such as automatic scaling, robust security, and high availability. It is possible to use a Load Balancer to perform basic load balancing on both layers 4 and 7 at the same time.

When messages are delivered via Layer 4, load balancing takes place at the intermediate transport layer. Transmission Control Protocol (TCP) is an Internet Layer 4 protocol for Hypertext Transfer Protocol HTTP/TCP. Network packets are simply forwarded to and from the upstream server by Layer 4 load balancers, which do not examine their contents. At the application level, where the data in each message is handled, Layer 7 load balancing occurs. HTTP is the most common Layer 7 protocol for website traffic on the Internet. While Layer 4 load balancers are capable of routing HTTP traffic, Layer 7 load balancers are far more capable of doing so, making them a better choice for TCP-based traffic. The message is read by a Layer 7 load balancer after the network traffic has been terminated. Based on the message's content, it can decide on load balancing.

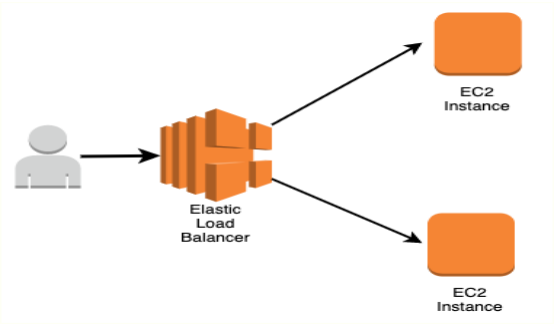


Fig-1 Load balancer

TYPES OF ELB: Elastic Load Balancing supports three types of load balancers:

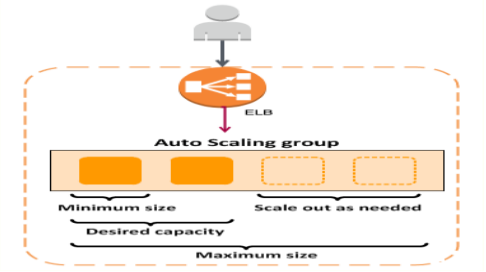
- Application Load Balancers
- Network Load Balancers
- Classic Load Balancers

Application Load Balancers (ALBs):

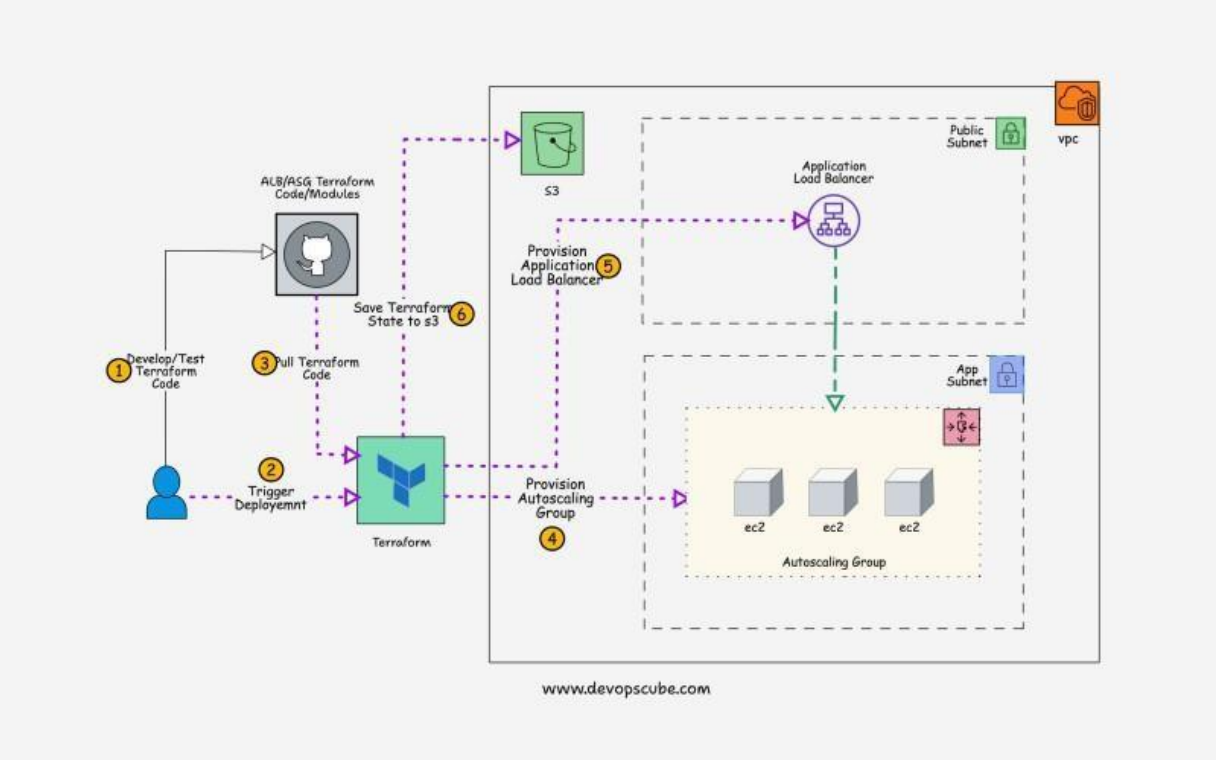
Application Load Balancers (ALBs) are advanced Layer 7 (application layer) load balancers that operate at the application level, making routing decisions based on content within the HTTP and HTTPS headers. ALBs are highly versatile and offer a wide range of features for modern application architectures, including support for path-based routing, host-based routing, and WebSocket protocols. They are well-suited for distributing traffic to microservices-based architectures and containerized workloads.

Network Load Balancers (NLBs):

Network Load Balancers (NLBs) are Layer 4 (transport layer) load balancers that operate at the network level, making routing decisions based on IP address and port information. NLBs are designed for high-performance, low-latency scenarios and are capable of handling millions of requests per second with minimal overhead.



System architecture:



II. LOAD BALANCING TOOLS

- Cloud Analyst
- Cloud Project
- Amazon Web Service

Cloud Analyst:

Cloud Analyst is a specialized role within organizations responsible for analyzing and optimizing cloud resources, costs, and performance. Cloud Analysts leverage various tools and methodologies to monitor, analyze, and optimize cloud infrastructure, ensuring efficient resource allocation and cost management. Key Responsibilities of Cloud Analyst:

Cloud Resource Optimization: Cloud Analysts analyze cloud resource utilization and performance metrics to identify inefficiencies and optimize resource allocation for improved performance and cost-effectiveness.

Cost Management: Cloud Analysts monitor cloud spending, identify cost-saving opportunities, and implement cost optimization strategies to reduce overall cloud expenses.

Amazon Web Services (AWS):

Amazon Web Services (AWS) is a leading cloud computing platform that offers a broad set of services, including compute, storage, networking, databases, analytics, machine learning, and more. AWS provides organizations with scalable, flexible, and secure cloud infrastructure to build and deploy a wide range of applications and services.

Key Services and Features of AWS:

Compute Services: AWS offers a variety of compute services, including Amazon EC2 (Elastic Compute Cloud) for virtual servers, AWS Lambda for serverless computing, and Amazon ECS (Elastic Container Service) for containerized applications.

Storage Services: AWS provides scalable storage solutions such as Amazon S3 (Simple Storage Service) for object storage, Amazon EBS (Elastic Block Store) for block storage, and Amazon Glacier for archival storage.

Auto Scaling Group :

An Auto Scaling Group in AWS is a powerful tool for automating the management of EC2 instances and ensuring the availability and scalability of your applications. By following the steps outlined above, you can set up an ASG to dynamically adjust the number of instances in response to changes in demand, improving the reliability and efficiency of your infrastructure.

Scaling policies:

Scaling Policies are attached with the ASG, it tells the ASG when to launch a new server and when to terminate the idle server based on the scaling policies given. Types of scaling policies

1. Target Tracking Scaling
2. Step Scaling
3. Simple Scaling

Create AMI:

Creating an Amazon Machine Image (AMI) in Amazon Web Services (AWS) allows you to capture the configuration of an EC2 instance, including the operating system, installed software, and any customizations you've made, so that you can launch identical instances in the future. To create an AMI, start by logging in to the AWS Management Console and navigating to the EC2 service. From there, select the EC2 instance that you want to use as the basis for your AMI. Once you've chosen the instance, right-click on it and select "Image and templates" from the dropdown menu, then click "Create image". This will initiate the process of creating a new AMI based on the selected instance.

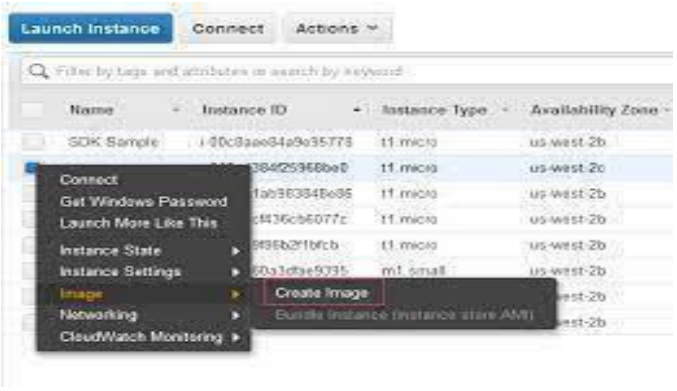


Fig-8 Create AMI

Configure Launch template:

Configuring a Launch Template in Amazon Web Services (AWS) enables you to define the configuration settings for EC2 instances that you intend to launch repeatedly with consistency. To create a Launch Template, start by logging in to the AWS Management Console and navigating to the EC2 service. Once there, locate the "Launch Templates" section in the navigation pane and click on "Create launch template". Begin by specifying a name and version for your Launch Template, ensuring it reflects the purpose or configuration of the template. Then, proceed to configure the template settings. This includes specifying the Amazon Machine Image (AMI) to use for the instances, the instance type, key pair for SSH access, security groups, network settings, and other parameters.

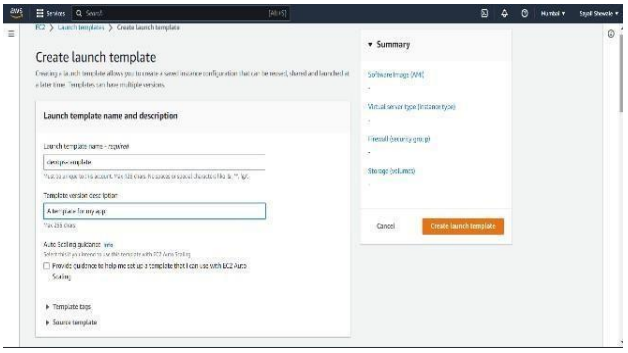
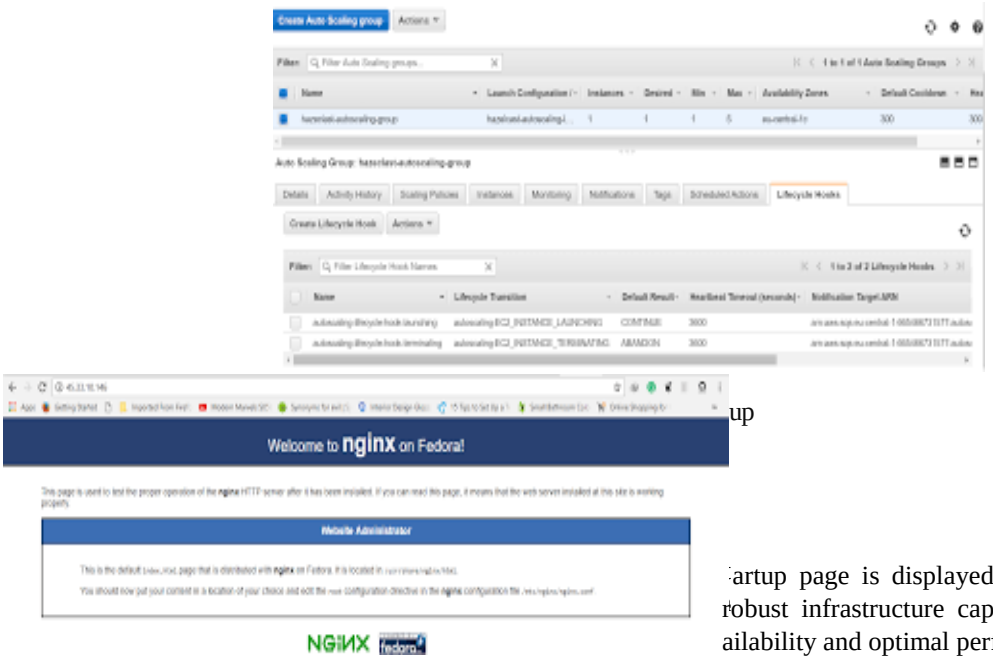


Fig-9 Launch Template

Implement Auto Scaling:

Creating an Auto Scaling Group (ASG) in Amazon Web Services (AWS) is a fundamental step in ensuring the availability and scalability of your EC2 instances to meet varying levels of demand. To create an ASG, begin by logging in to the AWS Management Console and navigating to the EC2 service. In the EC2 dashboard, locate the "Auto Scaling Groups" section in the navigation pane and click on "Create Auto Scaling group".

Start by defining the launch configuration for your ASG, which specifies the AMI, instance type, key pair, security groups, and other settings for the EC2 instances launched by the ASG. Once the launch configuration is set up, configure the scaling policies for the ASG. These policies determine how the ASG will automatically adjust the number of instances based on metrics such as CPU utilization, network traffic, or custom CloudWatch metrics.



up

artup page is displayed in the browser.The obust infrastructure capable of dynamically ailability and optimal performance.

Fig-11 Server startup page

V. CONCLUSION

Deploying an AWS infrastructure using Terraform to set up an Auto Scaling Group (ASG) with an Application Load Balancer (ALB) represents a powerful approach to managing scalable and resilient applications in the cloud. Terraform's Infrastructure as Code (IaC) paradigm offers significant advantages by codifying infrastructure configurations, thereby ensuring consistency, reproducibility, and version control across deployments. This approach not only enhances operational efficiency but also facilitates seamless collaboration among teams by providing a clear and documented way to manage cloud resources.

The integration of an Auto Scaling Group allows applications to dynamically scale based on defined metrics such as CPU utilization or request rates, ensuring optimal performance during peak times and cost-efficiency during periods of lower demand. Coupled with an Application Load Balancer, which distributes incoming traffic across multiple EC2 instances within the ASG, this setup enhances availability and fault tolerance by intelligently routing requests and seamlessly managing traffic spikes.

VI. REFERENCES

[1] .Danielo Gonc¸alves Gomes and Jos´e Neuman de Souza, “An Autonomic Computing-based Architecture for Cloud Computing Elasticity ,” Virtual University Institute (UFC VIRTUAL) , Fortaleza - Brazil .IEEE,2015

[2] .Satish Narayana Srirama, Alireza Ostovar, “Optimal Resource Provisioning for Scaling Enterprise Ap plications on the Cloud,” Institute of Computer Science,University of Tartu, Estonia.IEEE,2015

[3] . Shridhar G.Domanal and G. Ram Mohana Reddy, “Load Balancing in Cloud Environment using a Novel Hybrid Scheduling Algorithm ,”National Institute of Technology Karnataka, India.IEEE,2015 Surathkal, Mangalore

[4] . Ying Liu, Navaneeth Rameshan, Enric Monte, Vladimir Vlassov and Leandro Navarro, “ProRenaTa: Proactive and Reactive Tuning to Scale a Distributed Storage System,” KTH Royal Institute of Technology, Sweden.IEEE,2015

[5] . Ali Yadavar Nikravesh, Samuel A. Ajila, Chung- Horng Lung, “Towards an Autonomic AutoScaling Prediction System for Cloud Resource Provisioning,” Department of Systems andComputer Engineering, Carleton University 1125 Colonel By Drive, Ottawa K1S 5B6, Ontario Canada.IEEE,2015

[6] . Sidra Aslam, Maunam Ali Shah, “Load Balancing in Cloud Computing: A Survey of Modern Techniques,”COMSATS Institute of information technology, Islamabad, Pakistan .IEEE,2015

[7] . Mayanka Katyal, Atul Mishra, “A Comparative Study of Load Balancing in Cloud Computing Environment,” YMCA University of Science and Technolohy ,Faridabad, Hariyana.India

[8] . Sandeep Sharma, Sarabjit Singh, and Meenakshi Sharma, “Performance Analysis of Load Balancing Algorithms,” World Academy of Science, Engineering and Technology.IEEE,2008

[9] A. Y. Nikravesh, S. A. Ajila, C.H. Lung, “Cloud resource autoscaling system based on Hidden Markov Model (HMM)”,Proc. of the 8th IEEE International Conference on June 2014. Semantic Computing

[10] T. Lorido-Botran, J. Miguel-Alonso, J. A. Lozano, “A review of auto-scaling techniques for elastic applications in cloud environments,” Journal of Grid Computing, vol. 12, no. 4, December 2014

[11] Resma K. S, Dr. Sharvani G. S, Edge Distributed Cloud Middleboxes International Journal Of Advance Research, Ideas And Innovations In Technology, ISSN: 2454-132X, Volume 5, Issue 3. Luo, J., Rao, L., & Liu, X. (2017). Spatio-Temporal Load Balancing For Energy Cost Optimization In Distributed Internet Data Centres. IEEE Transactions On Cloud Computing, 3(3), 387– 397.