Title of the Project:

Auto Scaling with Load Balancer

Introduction In the age of digital transformation, cloud computing has revolutionized how modern applications are built, deployed, and maintained. One of the primary challenges in cloud infrastructure is managing dynamic workloads and ensuring that applications can scale efficiently based on user demand. Traditional server infrastructures are static and often struggle with traffic surges, resulting in slow performance, application downtime, or resource wastage. The project titled "Auto Scaling with Load Balancer" focuses on creating a cloudnative solution that adapts to changing traffic and resource demands automatically, thereby improving performance, availability, and cost-efficiency of web applications.

Problem Statement and Overview

Conventional IT systems are typically built with fixed computing resources. These static setups are prone to two critical inefficiencies: Over-Provisioning: Allocating more resources than necessary results in increased costs. Under-Provisioning: Limited resources cannot handle traffic spikes, leading to crashes and poor user experience.

Additionally, in the absence of efficient traffic distribution mechanisms, some servers become overloaded while others are underutilized. These issues are critical in cloud-hosted applications where traffic can vary by time, region, or events. Without proper auto scaling and load balancing, businesses risk system failures, unhappy users, and high operational expenses.

Objective

The objective of this project is to implement a scalable cloud infrastructure that: Dynamically adjusts the number of server instances based on real-time metrics. - Distributes incoming traffic evenly to avoid overloading any single server. - Ensures application high availability, performance consistency, and cost-effectiveness.

Tools and Technologies Used

This project is developed using cloud-native services, primarily from Amazon Web Services (AWS), along with scripting and monitoring tools. The following technologies are used: - Cloud Platform: Amazon Web Services (AWS) - Compute Resource: Amazon EC2 (Elastic Compute Cloud) - Scalability Component: Auto Scaling Group (ASG) - Traffic Distribution: Application Load Balancer (ALB) - Monitoring and Metrics: AWS CloudWatch - Security and Access Control: IAM (Identity and Access Management), Security

Groups - Scripting: Bash scripts and YAML configuration files - Optional: Python for automation tasks

Description of Submodules

1. EC2 Instance Configuration: A base Amazon EC2 instance is created with necessary configurations (such as a web application setup). This instance acts as a template for auto scaling.
2. Launch Template/Configuration: A launch template is defined with AMI, instance type,security groups, and startup scripts that are used by the Auto Scaling Group to spawn new instances.
3. Auto Scaling Group (ASG): The ASG is configured to automatically increase or decreasethe number of EC2 instances based on CloudWatch metrics (e.g., average CPU usage > 70%).
4. Application Load Balancer (ALB): ALB sits in front of the EC2 instances, distributing incoming HTTP/HTTPS traffic evenly to only healthy instances. Health checks are configured to detect and isolate failed instances.
5. Monitoring and Alerts: CloudWatch is used to continuously monitor resource usage andtrigger scaling events. Alarms can notify administrators of unusual behaviors or failures.
6. Security Configuration: IAM roles are used to control permissions, and security groupsdefine the traffic allowed to and from instances.

Design and Flow of the Project

The flow of the project is as follows:

1. A user sends a request via the web.
2. The Application Load Balancer receives the request and routes it to one of the healthy EC2 instances.
3. CloudWatch monitors instance metrics like CPU utilization and network traffic.
4. If traffic increases beyond a defined threshold, Auto Scaling Group launches new EC2 instances.
5. If traffic drops, ASG scales in by terminating excess instances.
6. All traffic continues to be load-balanced across the current healthy set of instances. This dynamic flow ensures a self-adjusting environment based on real-time needs.

Expected Outcome and Conclusion

The successful implementation of this project will result in: - High Availability: Continuous service with minimal downtime. - Elastic Scalability: Automatic addition/removal of resources based on demand. - Cost Efficiency: Reduced costs during low usage periods. - Optimized Performance: Balanced workload across all active servers.

This project demonstrates a practical and modern approach to managing cloud infrastructure using automation. The core idea is to eliminate manual intervention in resource allocation and enable web applications to be highly resilient, responsive, and efficient under varying traffic conditions. The project not only helps in understanding cloud architecture but also builds handson experience with real-world deployment strategies on AWS.