# INTERDISCIPLINARY PROJECT REPORT

**at**

# Sathyabama Institute of Science and Technology (Deemed to be University)

Submitted in partial fulfillment of the requirements for the award of Bachelor of Engineering Degree in Computer Science and Engineering

By

**KETHIREDDY YOGESH**

**(Reg.no:40111510)**



# DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING SCHOOL OF COMPUTING

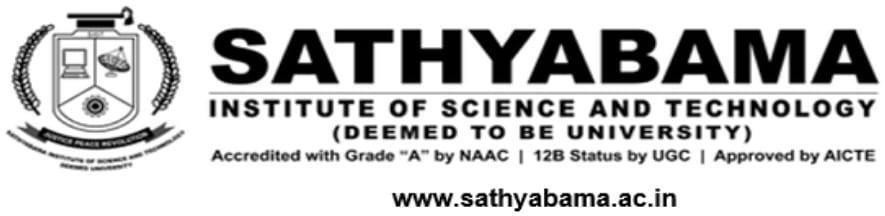
SATHYABAMA

## INSTITUTE OF SCIENCE AND TECHNOLOGY (DEEMED TO BE UNIVERSITY)

**Accredited with Grade “A” by NAAC | 12 B Status by UGC | Approved by AICTE**

## JEPPIAR NAGAR, RAJIV GANDHISALAI, CHENNAI – 600119

**APRIL 2023**



## DEPARTEMENT OF COMPUTER SCIENCE AND ENGINEERING BONAFIDE CERTIFICATE

This is to certify that this Project Report is the bonafide work of **KETHIREDDY YOGESH (40111510)** who carried out the project entitled “**APPLICATION LOAD BALANCER IN AWS”** under my supervision from January 2023 to April 2023.

## INTERNAL GUIDE Ms.DHARANI

**HEAD OF THE DEPARTMENT Dr.L.LAKSHMANAN M.E.,Ph.D.,**

**Submitted for Viva Voice Examination held on**

**Internal Examiner External Examiner**

# DECLARATION

**I KETHIREEDY YOGESH** hereby declare that the Project Report entitled “**APPLICATION LOAD BALANCER IN AWS”** done by me under the guidance of **Ms.DHARANI(**internal guide**)** at Sathyabama Institute of Science and Technology is submitted in partial fulfillment of the requirements for the award of Bachelor of Engineering degree in Computer Science and Engineering.

## DATE:

**PLACE:** CHENNAI **SIGNATURE OF THE CANDIDATE**

# ACKNOWLEDGEMENT

I am pleased to acknowledge my sincere thanks to **Board of Management** of **SATHYABAMA** for their kind encouragement in doing this project and for completing it successfully. I am grateful to them.

I convey my thanks to **Dr. T. Sasikala M.E., Ph.D.**, **Dean**, School of Computing **Dr.L.Lakshmanan M.E., Ph.D.,** Head of the Department of Computer Science and Engineering for providing me necessary support and details at the right time during the progressive reviews.

I would like to express my sincere and deep sense of gratitude to my Project Guide **Ms.DHARANI** for her valuable guidance, suggestions and constant encouragement paved way for the successful completion of my project work.

I wish to express my thanks to all Teaching and Non-teaching staff members of the **Department of Computer Science and engineering** who were helpful in many ways for the completion of the project.

# TRAINING CERTIFICATE



**ABSTRACT**

Load balancing is the method of distributing network traffic equally across a pool of resources that support an application. Modern applications must process millions of users simultaneously and return the correct text, videos, images, and other data to each user in a fast and reliable manner. To handle such high volumes of traffic, most applications have many resource servers with duplicate data between them. A load balancer is a device that sits between the user and the server group and acts as an invisible facilitator, ensuring that all resource servers are used equally.Companies usually have their application running on multiple servers. Such a server arrangement is called a server farm. User requests to the application first go to the load balancer. The load balancer then routes each request to a single server in the server farm best suited to handle the request. Load balancing is like the work done by a manager in a restaurant. Consider a restaurant with five waiters. If customers were allowed to choose their waiters, one or two waiters could be overloaded with work while the others are idle. To avoid this scenario, the restaurant manager assigns customers to the specific waiters who are best suited to serve them.

**TABLE OF CONTENTS**

|  |  |  |
| --- | --- | --- |
| **CHAPTER NO** | **TITLE** | **PAGE NO** |
|  | **ABSTRACT** | **VI** |
| **1** | **INTRODUCTION** | **1** |
| 1.1 Introduction of an Application Load Balancer | **1** |
| 1.2 Introduction of L4,L7 | **3** |
| 1.3 Introduction to Availability Zones | **6** |
| 1.4 Introduction to Local Zones | **6** |
| 1.5 Introduction of Outposts | **7** |
| 1.6 Introduction of Security Groups | **7** |
| 1.7 Introduction of Ip Address Types | **8** |
| **2** | **AIM AND SCOPE OF THE PRESENT INVESTIGATION** |  |
| 2.1 Aim | **10** |
| 2.2 Scope | **10** |
|  | 2.3 Create an Application Load Balancer | **10** |
|  | 2.3.1 Working of Load Balancer | **13** |
|  | 2.4 Types Of Load Balancer | **14** |
|  | 2.5 Load Balancer Algorithms | **14** |
|  | **2.6 Load Balancer Attributes** | **15** |
| **3** | **EXPERIMENTAL OR MATERIALS AND METHODS; ALGORITHMS USED** |  |
| 3.1 Specifications and Hardware Requirements | **12** |
| 3.2 Software requirements | **12** |
| **4** | **RESULTS** |  |
| 4.1 Results | **13** |
| **5** | **Conclusion and Summary** |  |
| 5.1 Conclusion | **15** |

|  |  |  |
| --- | --- | --- |
|  | 5.2 Further Enhancements | **15** |
|  | **Appendix** |  |
| A. Screenshots | **17** |
| B. Source Code | **24** |
|  | **References** | **16** |

# List of Figures

|  |  |  |
| --- | --- | --- |
| **Figure No.** | **Figure Name** | **Page No.** |
| **Fig 1.1** | **APPLICATION LOAD BALANCER** | 2 |
| **Fig 1.5.1** | **LOAD BALANCER** | 5 |
| **Fig 2.3.1** | **WORKING OF LOAD BALANCER** | 13 |

**CHAPTER 1 INTRODUCTION**

# INTRODUCTION OF AN APPLICATION LOAD BALANCER:

Elastic Load Balancing automatically distributes your incoming traffic across multiple targets, such as EC2 instances, containers, and IP addresses, in one or more Availability Zones. It monitors the health of its registered targets, and routes traffic only to the healthy targets. Elastic Load Balancing scales your load balancer as your incoming traffic changes over time. It can automatically scale to the vast majority of workloads.

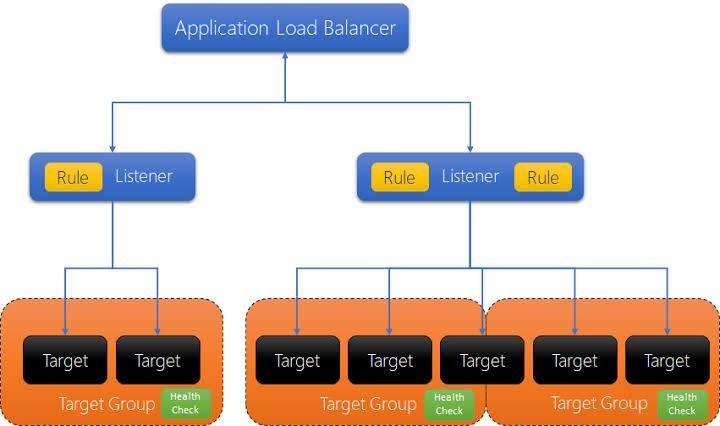
Elastic Load Balancing supports the following load balancers: Application Load Balancers, Network Load Balancers, Gateway Load Balancers, and Classic Load Balancers. You can select the type of load balancer that best suits your needs. This guide discusses Application Load Balancers. For more information about the other load balancers, see the User Guide for Network Load Balancers, the User Guide for Gateway Load Balancers, and the User Guide for Classic Load Balancers.

# APPLICATION LOAD BALANCER COMPONENTS:

A load balancer serves as the single point of contact for clients. The load balancer distributes incoming application traffic across multiple targets, such as EC2 instances, in multiple Availability Zones. This increases the availability of your application. You add one or more listeners to your load balancer.

A listener checks for connection requests from clients, using the protocol and port that you configure. The rules that you define for a listener determine how the load balancer routes requests to its registered targets. Each rule consists of a priority, one or more actions, and one or more conditions. When the conditions for a rule are met, then its actions are performed. You must define a default rule for each listener, and you can optionally define additional rules. Each target group routes requests to one or more

registered targets, such as EC2 instances, using the protocol and port number that you specify. You can register a target with multiple target groups. You can configure health checks on a per target group basis. Health checks are performed on all targets registered to a target group that is specified in a listener rule for your load balancer. The following diagram illustrates the basic components. Notice that each listener contains a default rule, and one listener contains another rule that routes requests to a different target group. One target is registered with two target groups.



## Fig 1.1 Application Load Balancer

* + 1. **BENEFITS OF MIGRATING FROM A CLASSIC LOAD BALANCER:** Using an Application Load Balancer instead of a Classic Load Balancer has the following benefits:
* Support for Path conditions (p.38). You can configure rules for your listener that forward requests based on the URL in the request. This enables you to structure your

application as smaller services, and route requests to the correct service based on the content of the URL.

* Support for Host conditions (p. 37). You can configure rules for your listener that forward requests based on the host field in the HTTP header. This enables you to route requests to multiple domains using a single load balancer.
* Support for routing based on fields in the request, such as HTTP header conditions (p. 37) and methods, query parameters, and source IP addresses.
* Support for routing requests to multiple applications on a single EC2 instance. You can register an instance or IP address with multiple target groups, each on a different port.
* Support for redirecting requests from one URL to another.
* Support for returning a custom HTTP response.
* Support for registering targets by IP address, including targets outside the VPC for the load balancer.
* Support for containerized applications. Amazon Elastic Container Service (Amazon ECS) can select an unused port when scheduling a task and register the task with a target group using this port. This enables you to make efficient use of your clusters.
* Support for monitoring the health of each service independently, as health checks are defined at the target group level and many CloudWatch metrics are reported at the target group level. Attaching a target group to an Auto Scaling group enables you to scale each service dynamically based on demand.
* Access logs contain additional information and are stored in compressed format.

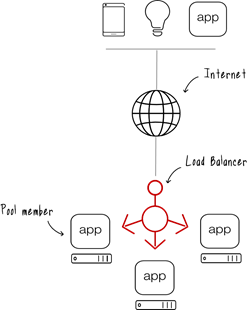
# INTRODUCTION OF L4,L7 AND GSLB LOAD BALANCERS:

Digital workspaces are heavily application-driven. As concurrent demand for [software-](https://www.citrix.com/solutions/digital-workspace/what-is-software-as-a-service.html) [as-a-service (SaaS)](https://www.citrix.com/solutions/digital-workspace/what-is-software-as-a-service.html) applications in particular continues to ramp up, reliably delivering them to end users can become a challenge if proper load balancing isn’t in place. Employees who already struggle to navigate multiple systems, interfaces, and security requirements will bear the additional burden of performance slowdowns and outages.

To promote greater consistency and keep up with ever-evolving user demand, server resources must be readily available and load balanced at Layers 4 and/or 7 of the Open Systems Interconnection (OSI) model:

* + - Layer 4(L4) load balancers work at the transport level. That means they can make routing decisions based on the TCP or UDP ports that packets use along with their source and destination IP addresses. L4 load balancers perform network address translation (NAT) but do not inspect the actual contents of each packet.
    - Layer 7 (L7) load balancers act at the application level, the highest in the OSI model. They can evaluate a wider range of data than L4 counterparts, including HTTP headers and SSL session IDs, when deciding how to distribute requests across the server farm.

Load balancers ensure reliability and availability by monitoring the "health" of applications and only sending requests to servers and applications that can respond in a timely manner.



**Fig 1.5.1 LOAD BALANCER**

# INTRODUCTION OF REGISTER TARGETS:

You can register EC2 instances, IP addresses, or Lambda functions as targets in a target group. This is an optional step to create a load balancer. However, you must register your targets to ensure that your load balancer routes traffic to them.

* + 1. In the Register targets page, add one or more targets as follows:
       - If the target type is Instances, select one or more instances, enter one or more ports, and then choose Include as pending below.
       - If the target type is IP addresses, do the following:

1. Select a network VPC from the list, or choose Other private IP addresses.
2. Enter the IP address manually, or find the IP address using instance details. You can enter up to five IP addresses at a time.
3. Enter the ports for routing traffic to the specified IP addresses.
4. Choose Include as pending below.
   * + - If the target type is Lambda, select a Lambda function, or enter a Lambda function ARN, and then choose Include as pending below.
     1. Choose Create target group

# INTRODUCTION OF AVAILABILITY ZONES:

You must select at least two Availability Zone subnets. The following restrictions apply

* Each subnet must be from a different Availability Zone.
* To ensure that your load balancer can scale properly, verify that each Availability Zone subnet for your load balancer has a CIDR block with at least a /27 bitmask (for example, 10.0.0.0/27) and at least eight free IP addresses per subnet. These eight IP addresses are required to allow the load balancer to scale out if needed. Your load balancer uses these IP addresses to establish connections with the targets. Without them your Application Load Balancer could experience difficulties with node replacement attempts, causing it to enter a failed state.

Note: If an Application Load Balancers subnet runs out of usable IP addresses while attempting to scale, the Application Load Balancer will run with insufficient capacity. During this time old nodes will continue to serve traffic, but the stalled scaling attempt may cause 5xx errors or timeouts when attempting to establish a connection.

# INTRODUCTION OF LOCAL ZONES:

You can specify one or more Local Zone subnets.The following restrictions apply:

* You cannot use AWS WAF with the load balancer.
* You cannot use a Lambda function as a target.

# INTRODUCTION OF OUTPOSTS:

You can specify a single Outpost subnet. The following restrictions apply:

* You must have installed and configured an Outpost in your on-premises data center. You must have a reliable network connection between your Outpost and its AWS Region. For more information, see the AWS Outposts User Guide.
* The load balancer requires two large instances on the Outpost for the load balancer nodes. The supported instance types are shown in the following table. The load balancer scales as needed, resizing the nodes one size at a time (from large to xlarge, then xlarge to 2xlarge, and then 2xlarge to 4xlarge). After scaling the nodes to the largest instance size, if you need additional capacity, the load balancer adds 4xlarge instances as load balancer nodes. If you do not have sufficient instance capacity or available IP addresses to scale the load balancer, the load balancer reports an event to the AWS Health Dashboard and the load balancer state is active Impaired.
* You can register targets by instance ID or IP address. If you register targets in the AWS Region for the Outpost, they are not used.
* The following features are not available: Lambda functions as targets, AWS WAF integration, sticky sessions, authentication support, and integration with AWS Global Accelerator.

## INTRODUCTION OF SECURITY GROUPS:

A security group acts as a firewall that controls the traffic allowed to and from your load balancer. You can choose the ports and protocols to allow for both inbound and outbound traffic.

The rules for the security groups that are associated with your load balancer must allow traffic in both directions on both the listener and the health check ports. Whenever you add a listener to a load balancer or update the health check port for a target group, you must review your security group rules to ensure that they allow traffic on the new port in both directions.

## INTRODUCTION OF IP ADDRESS TYPES:

You can set the types of IP addresses that clients can use to access your internet- facing and internal load balancers.

The following are the IP address types:

ipv4 - Clients must connect to the load balancer using IPv4 addresses (for example, 192.0.2.1)

dual stack - Clients can connect to the load balancer using both IPv4 addresses and IPv6 addresses.

Dual stack load balancer considerations

* The load balancer communicates with targets based on the IP address type of the target group
* When you enable dualstack mode for the load balancer, Elastic Load Balancing provides an AAAA DNS record for the load balancer. Clients that communicate with the load balancer using IPv4 addresses resolve the A DNS record. Clients that communicate with the load balancer using IPv6 addresses resolve the AAAA DNS record.
* Access to your internal dualstack load balancers through the internet gateway is blocked to prevent unintended internet access. However, this does not prevent non- IWG internet access (such as, through peering, Transit Gateway, AWS Direct Connect, or AWS VPN).

Connection idle timeout:

For each request that a client makes through a load balancer, the load balancer maintains two connections. The front-end connection is between a client and the load balancer. The backend connection is between the load balancer and a target. The load balancer has a configured idle timeout period that applies to its connections. If no data has been sent or received by the time that the idle timeout period elapses, the load balancer closes the connection. To ensure that lengthy operations such as file uploads have time to complete, send at least 1 byte of data before each idle timeout period elapses, and increase the length of the idle timeout period as needed.

For backend connections, we recommend that you enable the HTTP keep-alive option for your EC2 instances. You can enable HTTP keep-alive in the web server settings for your EC2 instances. If you enable HTTP keep-alive, the load balancer can reuse backend connections until the keep-alive timeout expires. We also recommend that you configure the idle timeout of your application to be larger than the idle timeout configured for the load balancer. Otherwise, if the application closes the TCP connection to the load balancer ungracefully, the load balancer might send a request to the application before it receives the packet indicating that the connection is closed. If this is the case, then the load balancer sends an HTTP 502 Bad Gateway error to the client.

By default, Elastic Load Balancing sets the idle timeout value for your load balancer to 60 seconds. Use the following procedure to set a different idle timeout value

## To update the idle timeout value using the console

* + 1. Open the Amazon EC2 console at <https://console.aws.amazon.com/ec2/>.
    2. On the navigation pane, choose Load Balancers.
    3. Select the load balancer.
    4. On the Attributes tab, choose Edit.
    5. Under Traffic configuration, enter a value for Idle timeout, in seconds. The valid range is from 1 through 4000.
    6. Choose Save changes. To update the idle timeout value using the AWS CLI Use the modify-load-balancer-attributes command with the idle\_timeout.timeout\_seconds attribute.

# CHAPTER 2

**AIM AND SCOPE OF THE PRESENT INVESTIGATION**

* 1. **AIM:**

To create a Application Load Balancer in Aws

## SCOPE:

If your organization runs high-traffic websites and applications or databases that receive a lot of queries, load balancing delivers multiple benefits by optimizing resource use, data delivery, and response time. In high-traffic environments, load balancing is what makes user requests go smoothly and accurately.They spare users the frustration of wrangling with unresponsive applications and resources.

Load balancing also plays a key role in preventing downtime and simplifying security, reducing the likelihood of lost productivity and lost profits for your organization.

## Create an Application Load Balancer:-

A load balancer takes requests from clients and distributes them across targets in a target group.

Before you begin, ensure that you have a virtual private cloud (VPC) with at least one public subnet in each of the Availability Zones used by your targets.

To create a load balancer using the AWS Management Console, complete the following tasks:-

## Step 1: Configure a target group :-

* + 1. Open the Amazon EC2 console at https://console.aws.amazon.com/ec2/.
    2. In the navigation pane, choose Target Groups.
    3. Choose Create target group.
    4. In the Basic configuration section, set the following parameters:
       1. For Choose a target type, select Instances to specify targets by instance ID or IP addresses to specify targets by IP address. If the target type is a Lambda function, you can enable health checks by selecting Enable in the Health checks section.
       2. For Target group name, enter a name for the target group.
       3. Modify the Port and Protocol as needed.
       4. If the target type is IP addresses, choose IPv4 or IPv6 as the IP address type, otherwise skip to the next step. Note that only targets that have the selected IP address type can be included in this target group. The IP address type cannot be changed after the target group is created.
       5. For VPC, select a virtual private cloud (VPC) with the targets that you want to include in your target group.
       6. For Protocol version, select HTTP1 when the request protocol is HTTP/1.1 or HTTP/2; select HTTP2, when the request protocol is HTTP/2 or gRPC; and select gRPC, when the request protocol is gRPC.

## Step 2: Register targets:-

You can register EC2 instances, IP addresses, or Lambda functions as targets in a target group. This is an optional step to create a load balancer. However, you must register your targets to ensure that your load balancer routes traffic to them.

1. In the Register targets page, add one or more targets as follows:
   * If the target type is Instances, select one or more instances, enter one or more ports, and then choose Include as pending below.
   * If the target type is IP addresses, do the following:
2. Select a network VPC from the list, or choose Other private IP addresses.
3. Enter the IP address manually, or find the IP address using instance details. You can enter up to five IP addresses at a time.
4. Enter the ports for routing traffic to the specified IP addresses.
5. Choose Include as pending below.
6. Choose Create target group

## Step 3: Configure a load balancer and a listener:-

To create an Application Load Balancer, you must first provide basic configuration information for your load balancer, such as a name, scheme, and IP address type. Then, you provide information about your network, and one or more listeners. A listener is a process that checks for connection requests. It is configured with a protocol and a port for connections from clients to the load balancer.

## Step 4: Test the load balancer :-

After creating your load balancer, you can verify that your EC2 instances pass the initial health check. You can then check that the load balancer is sending traffic to your EC2 instance.

1. After the load balancer is created, choose Close.
2. In the navigation pane, choose Target Groups.
3. Select the newly created target group.
4. Choose Targets and verify that your instances are ready. If the status of an instance is initial, it's typically because the instance is still in the process of being registered. This status can also indicate that the instance has not passed the minimum number of health checks to be considered healthy. After the status of at least one instance is healthy, you can test your load balancer. For more information, see Target health status (p. 82).
5. In the navigation pane, choose Load Balancers.
6. Select the newly created load balancer.
7. Choose Description and copy the DNS name of the internet facing or internal load balancer (for example, my-load-balancer-1234567890abcdef.elb.us-east- 2.amazonaws.com). • For internet facing load balancers, paste the DNS name into the address field of an internet connected web browser. • For internal load balancers, paste the DNS name into the address field of a web browser which has private connectivity to the VPC.If everything is configured correctly, the browser displays the default page of your server.
8. If the web page does not display, refer to the following documents for additional configuration help and troubleshooting steps.

## 2.3.1 Working Of Load Balancer:-

Load balancers handle incoming requests from users for information and other services. They sit between the servers that handle those requests and the internet. Once a request is received, the load balancer first determines which server in a pool is available and online and then routes the request to that server. During times of heavy loads, a load balancer acts promptly and can dynamically add servers in response to spikes in traffic. Conversely, load balancers can drop servers if demand is low.



## Fig 2.3.1 WORKING OF LOAD BALANCER

* 1. **Types Of Load Balancer:-**

Load balancing is a key component of highly available infrastructures. Depending on a network's needs, various types of load balancers can be deployed with different storage capabilities, functionalities and complexities.

A load balancer can be a physical appliance, a software instance or a combination of both.

The following are two types of load balancers:

1. Hardware load balancer:-

A hardware load balancer is a hardware device with specialized and proprietary built- in software that's designed to handle massive amounts of application traffic. These load balancers have a built-in virtualization capability and enable multiple instances of a virtual load balancer to be used on a single device.

Traditionally, vendors loaded proprietary software onto dedicated hardware and sold it to users as standalone appliances -- usually in pairs to provide failover if one system goes down. Growing networks require an organization to purchase additional or larger appliances.

1. Software load balancer:-

Software load balancing runs on virtual machines (VMs) or White Box Servers, most likely as an application delivery controller (ADC) function. ADCs typically offer additional features, including caching, compression and traffic shaping. Popular in cloud environments, virtual load balancing can offer a high degree of flexibility. For example, it enables users to automatically scale up or down to mirror traffic spikes or decreased network activity.

## Load Balancer Algorithms:-

Load-balancing algorithms determine which servers receive specific incoming client requests. There are two main types of load balancing algorithms: static and dynamic. 1.Static load-balancing algorithm:-

The IP hash-based approach calculates a given client's preferred server based on designated keys, such as HTTP headers or IP address information. This method

supports session persistence, or stickiness, which benefits applications that rely on user-specific stored state information, such as checkout carts on e-commerce

The round-robin method goes through all the available servers in sequential order and distributes traffic to a list of servers in rotation using the domain name system (DNS). An authoritative nameserver carries a list of different "A" records and provides one in response to each DNS query.

The weighted round-robin approach enables admins to assign varying weights to each server. This way, the servers that can handle more traffic receive slightly more traffic based on their weight. Weighting is configured within DNS records.

Dynamic load-balancing algorithm:-

The least-connections method favors servers with the fewest ongoing transactions and checks and sends traffic to those servers that have the fewest open connections. This algorithm assumes that all connections require almost equal processing power.

The weighted least connection method assumes that some servers can handle more traffic compared to others. Thus, it enables admins to assign different weights to each server.

The weighted response time approach uses the response time averages of each server and combines them with the number of connections each server has open to find the best destination for sending traffic. This algorithm ensures faster service, as it sends traffic to the servers with the quickest response time.

The resource-based algorithm distributes load based on the availability of resources on each server at the time. Before traffic distribution, it queries a specialized software called an agent that runs on each server to measure the availability of the central processing unit and memory.

## Load Balancer Attributes:-

The following are the load balancer attributes: access\_logs.s3.enabled

Indicates whether access logs stored in Amazon S3 are enabled. The default is false access\_logs.s3.bucket

The name of the Amazon S3 bucket for the access logs. This attribute is required if access logs are enabled. For more information, see Enable access logs (p. 133). access\_logs.s3.prefix

The prefix for the location in the Amazon S3 bucket deletion\_protection.enabled

Indicates whether deletion protection is enabled.The default is false. idle\_timeout.timeout\_seconds

The idle timeout value, in seconds. The default is 60 seconds. ipv6.deny\_all\_igw\_traffic

Blocks internet gateway (IGW) access to the load balancer, preventing unintended access to your internal load balancer through an internet gateway. It is set to false for internet-facing load balancers and true for internal load balancers. This attribute does not prevent non-IGW internet access (such as, through peering, Transit Gateway, AWS Direct Connect, or AWS VPN).

routing.http.drop\_invalid\_header\_fields.enabled

Indicates whether HTTP headers with header fields that are not valid are removed by the load balancer (true), or routed to targets (false). The default is false. Elastic Load Balancing requires that valid HTTP header names conform to the regular expression [-A-Za-z0-9]+, as described in the HTTP Field Name Registry. Each name consists of alphanumeric characters or hyphens. Select true if you want HTTP headers that do not conform to this pattern, to be removed from requests. routing.http.x\_amzn\_tls\_version\_and\_cipher\_suite.enabled

Indicates whether the two headers (x-amzn-tls-version and x-amzn-tls-cipher-suite), which contain information about the negotiated TLS version and cipher suite, are added to the client request before sending it to the target. The x-amzn-tls-version header has information about the TLS protocol version negotiated with the client, and the x-amzn-tls-cipher-suite header has information about the cipher suite negotiated with the client. Both headers are in OpenSSL format. The possible values for the attribute are true and false. The default is false.

routing.http.xff\_client\_port.enabled

Indicates whether the X-Forwarded-For header should preserve the source port that the client used to connect to the load balancer. The possible values are true and false. The default is false.

routing.http.xff\_header\_processing.mode

Enables you to modify, preserve, or remove the X-Forward-For header in the HTTP request before the Application Load Balancer sends the request to the target. The possible values are append, preserve, and remove. The default is append.

* If the value is append, the Application Load Balancer adds the client IP address (of the last hop) to the X-Forward-For header in the HTTP request before it sends it to targets.
* If the value is preserve, the Application Load Balancer preserves the X-Forward-For header in the HTTP request, and sends it to targets without any change.
* If the value is remove, the Application Load Balancer removes the X-Forward-For header in the HTTP request before it sends it to targets.

# CHAPTER 3

**EXPERIMENTAL OR REQUIREMENTS AND METHODS**

## Secification and Hardware Requirements:

* + 1. AWS Console Login
    2. Suported system
    3. System with OS (Windows or Linux)

**Display Mode:**

1. Color Quality – Highest[64bit]
2. Screen Resolution – 250 \* 400Pixels

# Chapter 4 RESULTS AND DISCUSSION

* 1. **RESULT:-**
  2. **Discussion:-**

The distribution of a collection of tasks over multiple computing units (or similar resources) to make the entire process easier to complete and much more efficient is known as load balancing. It increases the responsiveness and availability of applications or websites for users by ensuring that no single server bears too much strain and evenly disperses the load.

IT departments can use an application delivery controller (ADC) with load balancing features to assure service scalability and availability. Its extensive traffic management capabilities can assist a company in efficiently routing requests to the appropriate resources for each end-user. Many other functions *(such as encryption, authentication, and web application firewalling)* are available through an ADC, which can provide a single point of control for securing, managing, and monitoring multiple applications and services across multiple environments while ensuring the best end-user experience.

# Chapter 5

**REFERENCES**

https://elearning.starcertification.org https://[www.starcertification.org/Certifications/Certificate/cloud](http://www.starcertification.org/Certifications/Certificate/cloud) https://docs.aws.amazon.com/Route53/routing-policy-simple.html https://docs.aws.amazon.com/AWSEC2//LaunchingAndUsingInstances.html https://docs.aws.amazom.com/route53/

# Appendix

**A: Screenshot:**

