# Introduction to Elastic Load Balancing - Network Load Balancing

Overview

This lab provides you with an introduction to Elastic Load Balancer. It covers creating and testing a Network Elastic Load Balancer. When the lab is launched you are provided with two web servers in two different Availability Zones. In the lab you will create a Network Elastic Load Balancer and use the two Web Servers as targets. You will then test the functionality of the load balancer in different scenarios.

Lab VPC
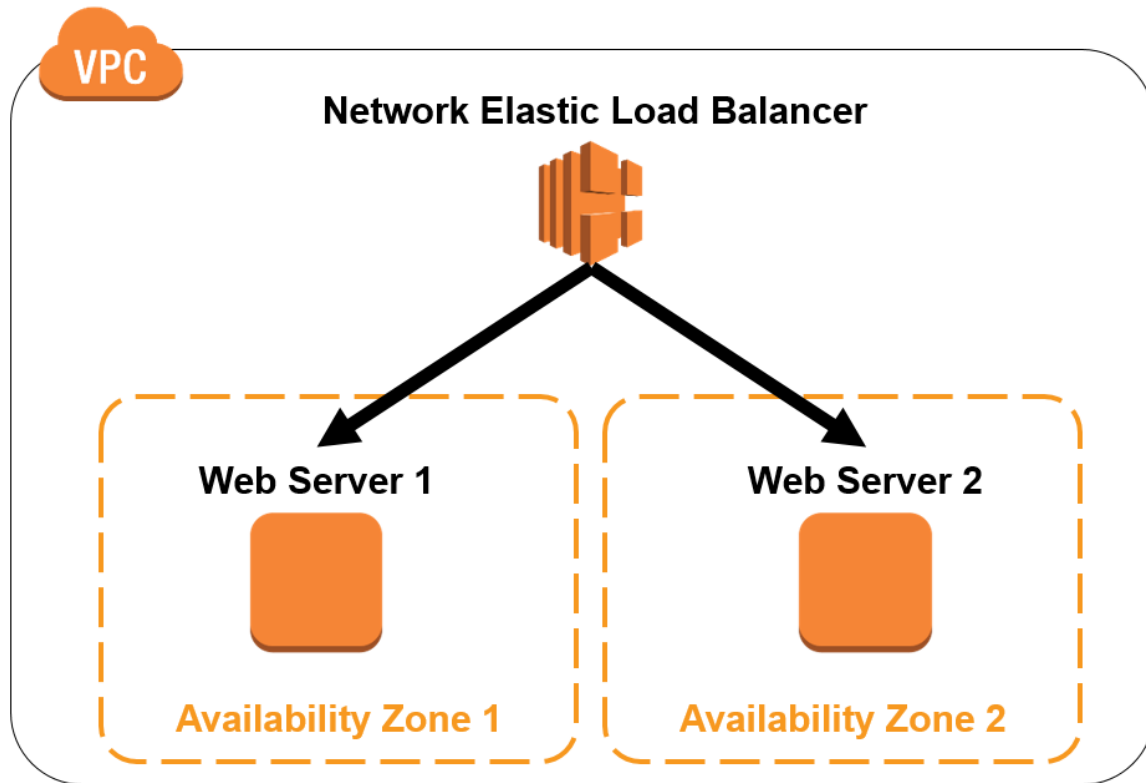
Lab Description

In this lab you will:

- Test connectivity to two web servers that reside in two different Availability Zones
- Create a Network Load Balancer and use the two web servers as Elastic Load Balancer targets
- Test the default functionality of your load balancer
- Enabled Cross-Zone load balancing and test how your load balancer behaves

- Test the behavior of your load balancer during a failure of one of your web servers
- Test the behavior of your load balancer after your web server has recovered from the failure

Amazon Elastic Load Balancer

An Amazon Elastic Load Balancer (Amazon ELB) is a service that automatically distributes incoming application traffic across multiple Amazon EC2 instances. It enables you to achieve even greater fault tolerance in your applications, seamlessly providing the amount of load balancing capacity needed in response to incoming application traffic. Elastic Load Balancing detects unhealthy instances within a pool and automatically reroutes traffic to healthy instances until the unhealthy instances have been restored.

Customers can enable Elastic Load Balancing within a single Availability Zone or across multiple zones for even more consistent application performance. Elastic Load Balancing can also be used in an Amazon Virtual Private Cloud (VPC) to distribute traffic between application tiers.

Task 1: Launch Web Servers

In this task, you will launch two Amazon Linux EC2 instances, with an Apache PHP web server and basic application installed on initialization. You will also demonstrate a simple example of bootstrapping instances using the Amazon EC2 metadata service.

Amazon Machine Images (AMIs) and instances

Amazon EC2 provides templates known as *Amazon Machine Images (AMIs)* that contain a software configuration (for example, an operating system, an application server, and applications). You use these templates to launch an *instance*, which is a copy of the AMI running as a virtual server in the cloud.

You can launch different types of instances from a single AMI. An *instance type* essentially determines the hardware capabilities of the virtual host computer

for your instance. Each instance type offers different compute and memory capabilities. Select an instance type based on the amount of memory and computing power that you need for the application or software that you plan to run on the instance. You can launch multiple instances from an AMI.

Your instance keeps running until you stop or terminate it, or until it fails. If an instance fails, you can launch a new one from the AMI.

When you create an instance, you will be asked to select an instance type. The instance type you choose determines how much throughput and processing cycles are available to your instance.

3. Launch 2 instances in availability zones **1a** and **1b**

4. On the Services menu, click **EC2**.

5. In the left navigation pane, click **Instances**.

6. Click Launch Instance

Step 1: Name and tags

- Provide a tag value for your instance example ➔ **webserver1**

Name and tags Info

Name

e.g. My Web Server   Add additional tags

Tags enable you to categorize your AWS resources in different ways, for example, by purpose, owner, or environment. This is useful when you have many resources of the same type — you can quickly identify a specific resource based on the tags you have assigned to it. Each tag consists of a Key and a Value, both of which you define.

Step 2: Application and OS Images (Amazon Machine Image)

An **Amazon Machine Image (AMI)** provides the information required to launch an instance, which is a virtual server in the cloud.

An AMI includes:

1. A template for the root volume for the instance (for example, an operating system or an application server with applications)
2. Launch permissions that control which AWS accounts can use the AMI to launch instances
3. A block device mapping that specifies the volumes to attach to the instance when it is launched

The **Quick Start** list contains the most commonly-used AMIs. You can also create your own AMI or select an AMI from the AWS Marketplace, an online store where you can sell or buy software that runs on AWS.

- Click **Quick Start**
- Select **Amazon Linux AWS**
- Under **Amazon Machine Image (AMI)** select **Amazon Linux 2 AMI** (HVM) (make sure it says **Free tier eligible**)

Step 3: Instance Type

Amazon EC2 provides a wide selection of *instance types* optimized to fit different use cases. Instance types comprise varying combinations of CPU, memory, storage, and networking capacity and give you the flexibility to choose the appropriate mix of resources for your applications. Each instance type includes one or more *instance sizes*, allowing you to scale your resources to the requirements of your target workload.

- Click on the drop down and select **t2.micro**.
  - A **t2.micro** instance type has 1 virtual CPUs and 1 GiB of memory.

Step 4: Key pair (login)

- Select an existing key pair or create a new key pair.
  - Please refer to previous lab and use the Keypair that was created.

  Do not create multiple keypair per region.

Amazon EC2 uses public–key cryptography to encrypt and decrypt login information. To log in to your instance, you must create a key pair, specify the name of the key pair when you launch the instance, and provide the private key when you connect to the instance.

  - Click the **key pair name** drop-down and select an existing keypair

Step 5: Network settings

This includes networking settings.

The **Network** indicates which Virtual Private Cloud (VPC) you wish to launch the instance into. You can have multiple networks, such as different ones for development, testing and production.

- To the right of **Network settings**, click on **edit**

▼ Network settings Get guidance   Edit

Network Info

vpc-
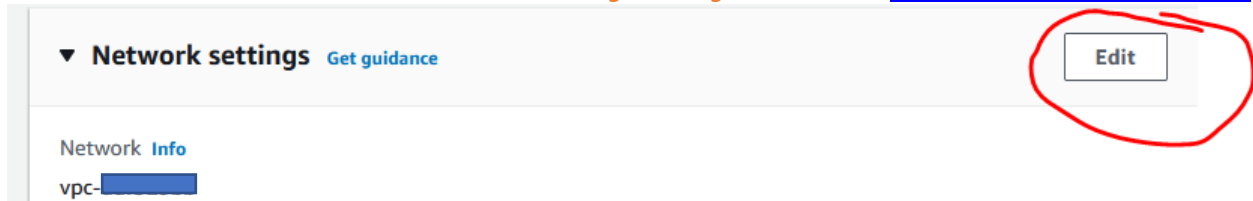
- Under **VPC**, select **Default VPC**.
- Under **Subnet**, select available zone 1a
- Under **Auto-assign public IP**, select Enable.

Firewall (security Groups)

A *security group* acts as a virtual firewall that controls the traffic for one or more instances. When you launch an instance, you associate one or more security groups with the instance. You add *rules* to each security group that allow traffic to or from its associated instances. You can modify the rules for a security group at any time; the new rules are automatically applied to all instances that are associated with the security group.

- Select Create Security Group
- Security group name **webserver-sg**
- Description **webserver-sg**
- Add 2 security groups rules (SSH and HTTP)

Step 6: Configure storage.

Amazon EC2 stores data on a network-attached virtual disk called *Elastic Block Store*.

You will launch the Amazon EC2 instance using a default 8 GiB disk volume. This will be your root volume (also known as a 'boot' volume).

- Leave this section as default.

Step 7: Advance details

- Scroll down, then expand **Advanced Details**.

At the bottom of the page in the **User data** section, copy and paste the attached user data content.

When you launch an instance, you can pass *user data* to the instance that can be used to perform common automated configuration tasks and even run scripts after the instance starts.

Your instance is running Amazon Linux, so you will provide a *shell script* that will run when the instance starts.

- Click on the link below, copy the data and paste into the **User data** field:

User data Script

The script will:

- Install an Apache web server (httpd)
- Configure the web server to automatically start on boot
- Activate the Web server.
- Create a simple web page.

- Click on **Launch instance.**

- Launch the second instance in **AZ 1b. webser2**

- Make sure to use same security group as server1 webserver-sg

This may take a few minutes. You can refresh the status of your instances by clicking the refresh icon.

Task 2: Create an Elastic Load Balancer

5. In the **AWS Management Console**, on the **Services** menu, click **EC2.**

6. In the left navigation pane, click **Load Balancers**.

7. Click Create Load Balancer

A load balancer serves as the single point of contact for clients. The load balancer distributes incoming traffic across multiple targets, such as Amazon EC2 instances. This increases the availability of your application.

8. On the **Select load balancer type**, below **Network Load Balancer**, click Create

A Network Load Balancer functions at the fourth layer of the Open Systems Interconnection (OSI) model. It can handle millions of requests per second. After the load balancer receives a connection request, it selects a target from the target group for the default rule. It attempts to open a TCP connection to the selected target on the port specified in the listener configuration.

9. On the **Basic configuration** section, configure:

- **Load balancer name:** myNLB
- **Scheme: Internet facing**
- **VPC:** *DefaultVPC*
- **Availability Zones**
- **Mappings:** Select both Availability Zones 1a and 1b

10. Remove the **default** security group and don't select security group.

11. In the **Listeners and routing** section, choose **Create target group**.

You register targets for your Network Load Balancer with a target group. By default, the load balancer sends requests to registered targets using the port and protocol that you specified for the target group.

11. In the **Basic configuration** section, configure:

- **Target group name:** MyTG
- **Choose a target type:** Instances
- **Protocol:** TCP
- **Port:** 80
- **VPC:** *DefaultVPC*
12. Expand **Advanced health check settings**, then configure:

- **Healthy threshold:** 2
- **Interval:** *10 seconds*

The load balancer sends a health check request to each registered target every HealthCheckIntervalSeconds seconds, using the specified port, protocol, and ping path. It waits for the target to respond within the response timeout period. If the health checks exceed the threshold for consecutive failed responses, the load balancer takes the target out of service. When the health checks exceed the threshold for consecutive successful responses, the load balancer puts the target back in service.

Since you've set the **Healthy threshold** to *2* and the interval to *10 seconds*, it will take at least *20 seconds* for your instance to report a *healthy* status. By default, each load balancer node routes requests only to the healthy targets in its Availability Zone.

13. At the bottom of the screen, click Next

14. In the **Available Instances** Section

15. Select both instances

16. Click Include as pending below

Make sure you click Include as pending below

16. Click Create target group

17. Navigate to the tab that you were using to create your load balancer.

18. In the **Listeners and routing** section, click the refresh button.

19. Select the target group that you created earlier.

- Review your load balancer configuration
- Click Create load balancer
20. Click View load balancers

21. Select your NLB and In the **Details** tab of your load balancer, copy the DNS name to your text editor.

Your DNS name will look similar to *myELB-4e009e86b4f704cc.elb.us-west-2.amazonaws.com*

22. In the left navigation pane, click **Target Groups**.

23. Click **MyTG**.

24. Click the **Targets** tab.

25. Wait for your EC2 instances to have a status of *healthy*.

You may have to refresh the screen to see the changes. Here are the possible values for the health status of a registered target:

- **initial:** The load balancer is in the process of registering the

target or performing the initial health checks on the target
- **healthy:** The target is healthy
- **unhealthy:** The target did not respond to a health check or failed the health check.
- **unused:** The target is not registered with a target group, the target group is not used in a listener rule for the load balancer, or the target is in an Availability Zone that is not enabled for the load balancer.
- **draining:** The target is deregistering and connection draining is in process

Task 3: Test Your Load Balancer

Test the Default Functionality

26. In a new browser tab, paste the DNS name and then press **Enter**.

You will see the HTML page for one of your EC2 instances.

27. Refresh the page a few times.

Notice that the same EC2 instance page is displayed.

With Network Load Balancers, cross-zone load balancing is disabled by default. After you create a Network Load Balancer, you can enable or disable cross-zone load balancing at any time.

Cross-zone load balancing distributes traffic evenly across all targets in the Availability Zones enabled for the load balancer.

Task 4: Test Your Load Balancer During a Failure

Test Load Balancer During a Failure

40. In the **AWS Management Console**, in the left navigation pane, click **Instances**.

41. Select the EC2 instance that is currently serving you the web page.

42. In the **Instance state** menu, click **Stop instance**.

43. Click Stop

44. Wait a minute or two.

45. Return to the browser tab that you used to access your load balancer.

46. Refresh the page a few times.

You will see that that your load balancer now displays the web page for your other instance.

Test Load Balancer After Recovering From a Failure

47. In the **AWS Management Console**, in the left navigation pane, click **Instances**.

48. Select the instance that you stopped earlier.

49. In the **Instance state** menu, click **Start instance**.

50. Wait for a minute or two for your instance to fully start.

You can click the **refresh** button to update the status.

51. In the browser tab for your Load Balancer, refresh the page.

You will see that that your load balancer now displays the web page of the instance that it originally used.

Conclusion

Congratulations! You now have successfully:

- Tested connectivity to two web servers that resided in two different Availability Zones
- Created a Network Load Balancer

- Tested the default functionality of your load balancer
- Tested the behavior of your load balancer during a failure of one of your web servers
- Tested the behavior of your load balancer after your web server recovered

Additional Resources

- [How Elastic Load Balancing Works](#)
- [Amazon ELB](#)
- [What Is a Network Load Balancer](#)