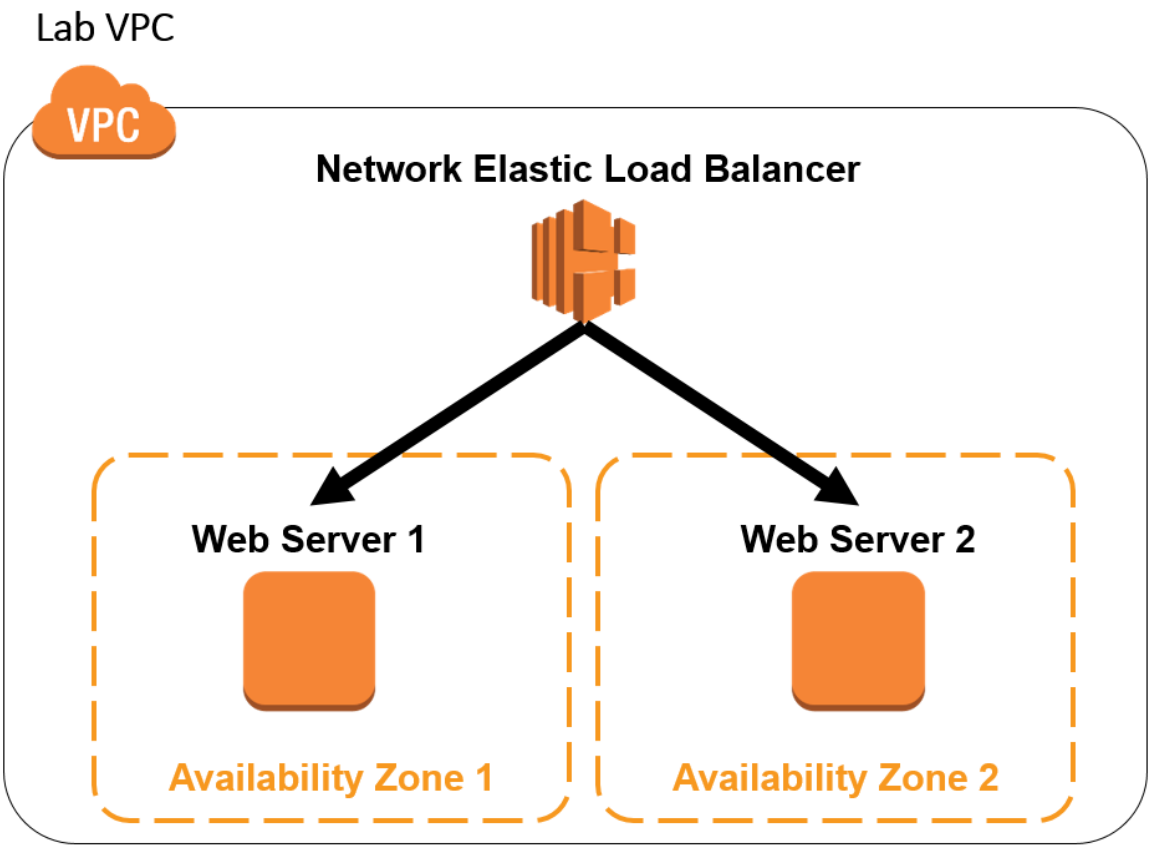
Introduction to Elastic Load Balancing

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 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**

7. In the row for the **Amazon Linux 2 AMI**, click **Select**

The **t2.micro** instance type, which is the lowest-cost option, should be automatically selected.

7. Click **Next: Configure Instance Details** then configure:

- **Number of instances:** 1
- **Network:** *DefaultVPC*
- **Subnet:** Select a subnet in AZ1a

8. Scroll to the bottom of the screen, then expand **Advanced Details**.

9. Enter the following into **User data:**

NLBServerBootstrap

This will allow you to bootstrap your instance. It will install Apache and PHP and sample code (PHP scripts) needed for this lab when the instance is created and launched. User data provides a mechanism to pass data or a script to the Amazon metadata service, which instances can access at launch time.

Tip If you type this text instead of copying it, press SHIFT+ENTER to create new lines in the text box.

10. Click **Next: Add Storage**

You will use the default storage device configuration.

11. Click **Next: Add Tags**

12. Click **Add Tag** then configure:

- **Key:** Nam
- **Value:** MyLBInstance

This name, more correctly known as a tag, will appear in the console when the instance launches. It makes it easy to keep track of running machines in a complex environment. Use a name that you can easily recognize and remember.

13. Click **Next: Configure Security Group** then configure:

- **Security group name:** MyLBInstance
- **Description:** MyLBInstance

A *security group* acts as a firewall that controls the traffic allowed into a group of instances. When you launch an Amazon EC2 instance, you can assign it to one or more security groups. For each security group, you add rules that govern the allowed inbound traffic to instances in the group. All other inbound traffic is discarded. You can modify rules for a security group at any time. The new rules are automatically enforced for all existing and future instances in the group.

By default, AWS creates a rule that allows Secure Shell (SSH) access from any IP address. It is *highly recommended* that you restrict terminal access to the ranges of IP addresses (e.g., IPs assigned to machines within your company) that have a legitimate business need to administer your Amazon EC2 instance.

14. Click to remove the SSH rule.

15. Click **Add Rule** then configure:

- **Type:** *HTTP*
- **Source:** *Anywhere*
- Click **Review and Launch**

This will add a default handler for HTTP that will allow requests from anywhere on the Internet. Since you want this web server to be accessible to the general public.

16. Review your choices, and then click **Launch**

You may see a warning on this screen that "Your security group … is open to the world." This is a result of not restricting SSH access to your machine, as described earlier. For the purposes of this lab only, you may ignore this warning.

17. In the **Select an existing key pair or create a new key pair** window:

- Select **Proceed without a key pair**.
- Click **I acknowledge that...**
- Click **Launch Instances**

Please do not launch your instance with a key pair.

A status page notifies you that your instances are launching.

18. Click **View Instances**

19. Wait for your instances to display:

20. Launch the second instance **AZ 1b**

- **Instance State:** *running*
- **Status Checks:** *2/2 checks passed*

This indicates that your instance is now fully available.

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:** myNL

- **VPC:** *DefaultVPC*
- **Network Mappings:** Select both Availability Zones 1a and 1b

10. 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 **IP addresses** section, select both EC2 instances.

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. In the **Description** 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.

Test Cross-Zone Load Balancing Functionality

28. In the **AWS Management Console**, in the left navigation pane, click **Load Balancers**.

29. In the **Description** tab, scroll down to the **Attributes** section.

30. Click **Edit attributes** then configure:

- Select **Cross-Zone Load Balancing** *Enable*
- Click **Save**

31. Wait a minute or two.

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

33. Refresh the page a few times.

You should see that your network load balancer now directs you to both of your EC2 instances.

Disable Cross-Zone Load Balancing Functionality

34. In the **AWS Management Console**, in the left navigation pane, click **Load Balancers**.

35. In the **Description** tab, scroll down to the **Attributes** section.

36. Click **Edit attributes** then configure:

- De-select **Cross-Zone Load Balancing** *Enable*
- Click **Save**

37. Wait a minute or two.

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

39. Refresh the page a few times.

You should see that your network load balancer is now only serving pages from one of your instances.

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
- Enabled Cross-Zone load balancing and tested how your load balancer behaved
- 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