Thanks & Regards,

Shantaram Vernekar

**From:** Shantaram Vernekar   
**Sent:** Monday, July 16, 2018 9:01 PM  
**To:** Shantaram Vernekar <[shantaram\_vernekar@persistent.com](mailto:shantaram_vernekar@persistent.com)>; 'shantaram.vernekar@citi.com' <[shantaram.vernekar@citi.com](mailto:shantaram.vernekar@citi.com)>  
**Subject:** AWS - LB and auto scaling

Load Balancers

The Elastic Load Balancing service allows you to distribute traffic across a group of Amazon EC2 instances in one or more *Availability Zones*, enabling you to achieve high availability in your applications.

Elastic Load Balancing supports routing and load balancing of Hypertext Transfer Protocol (HTTP), Hypertext Transfer Protocol Secure (HTTPS), *Transmission Control Protocol (TCP)*, and *Secure Sockets Layer (SSL)* traffic to Amazon EC2 instances.

provides a stable, single *Canonical Name record (CNAME)* entry point for *Domain Name System (DNS)* configuration

supports both Internet-facing and internal application-facing load balancers.

supports health checks for Amazon EC2 instances to ensure traffic is not routed to unhealthy or failing instances

Elastic Load Balancing can automatically scale based on collected metrics

Elastic Load Balancing is a managed service, it scales in and out automatically to meet the demands of increased application traffic and is highly available within a region itself as a service

Elastic Load Balancing also supports integrated certificate management and SSL termination.

Elastic Load Balancing seamlessly integrates with the Auto Scaling service to automatically scale the Amazon EC2 instances behind the load balancer

Elastic Load Balancing is a highly available service itself and can be used to help build highly available architectures

Types:

**Internet-Facing Load Balancers**

Public DNS name is given, no IP is shared – as it can change when it is auto scaled.

An AWS recommended best practice is always to reference a load balancer by its DNS name, instead of by the IP address of the load balancer, in order to provide a single, stable entry point

Because Elastic Load Balancing scales in and out to meet traffic demand, it is not recommended to bind an application to an IP address that may no longer be part of a load balancer’s pool of resources.

Elastic Load Balancing in Amazon VPC supports IPv4 addresses only. Elastic Load Balancing in EC2-Classic supports both IPv4 and IPv6 addresses.

**Internal Load Balancers**

In a multi-tier application, it is often useful to load balance between the tiers of the Application

You can use *internal load balancers* to route traffic to your Amazon EC2 instances in VPCs with privatesubnets.

**HTTPS Load Balancers**

Elastic Load Balancing does not support *Server Name Indication* (SNI) on your load balancer. This means that if you want to host multiple websites on a fleet of Amazon EC2 instances behind Elastic Load Balancing with a single SSL certificate, you will need to add a *Subject Alternative Name (SAN)* for each website to the certificate to avoid site users seeing a warning message when the site is accessed

**Listeners**

Every load balancer must have one or more *listeners* configured.

A listener is a process that checks for connection requests

The SSL protocol is primarily used to encrypt confidential data over insecure networks such as the Internet. The SSL protocol establishes a secure connection between a client and the back-end server and ensures that all the data passed between your client and your server is private

Elastic Load Balancing supports the following protocols:

HTTP

HTTPS

TCP

SSL

The SSL protocol is primarily used to encrypt confidential data over insecure networks such as the Internet.

**Configuring Elastic Load Balancing**

*idle connection timeout*, *cross-zone load balancing*, *connection draining*, *proxy protocol*, *sticky sessions,* and *health checks*

**Connection Draining**

You should enable *connection draining* to ensure that the load balancer stops sending requests to instances that are deregistering or unhealthy, while keeping the existing connections open.

**Proxy Protocol == x forwarder**

**Sticky Sessions**

By default, a load balancer routes each request independently to the registered instance with the smallest load.

However, you can use the *sticky session* feature (also known as *session affinity*), which enables the load balancer to bind a user’s session to a specific instance. This ensures that all requests from the user during the session are sent to the

same instance.

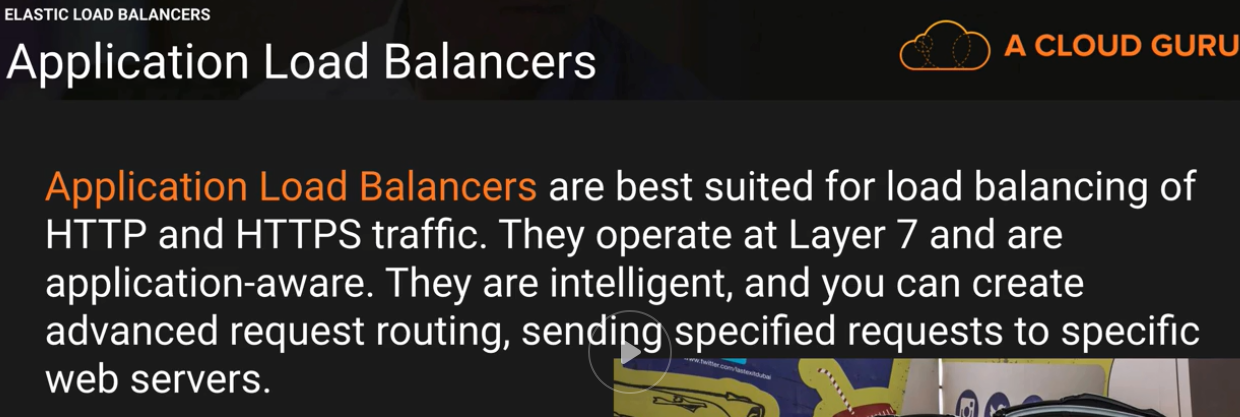
Application Load Balancers

Http/Https traffic

Operate at layer 7

Application aware

Intelligent

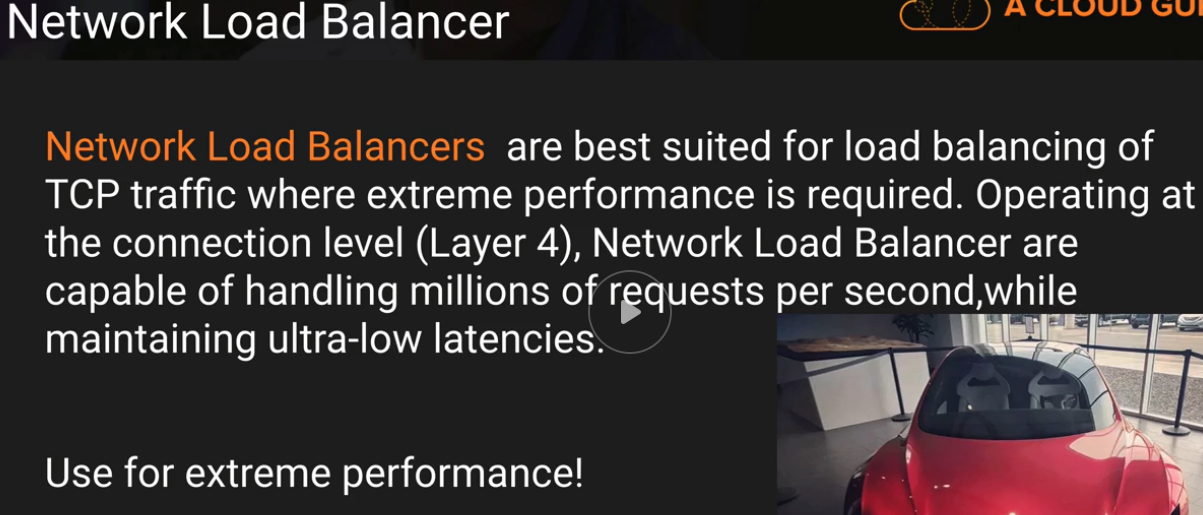


TCP traffic

Operate at layer 4

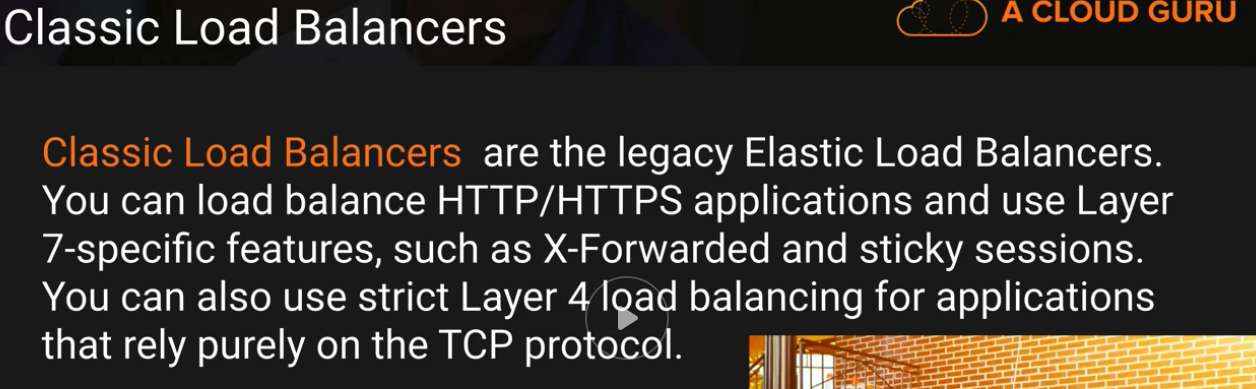
Capable of handling milling of requests per second

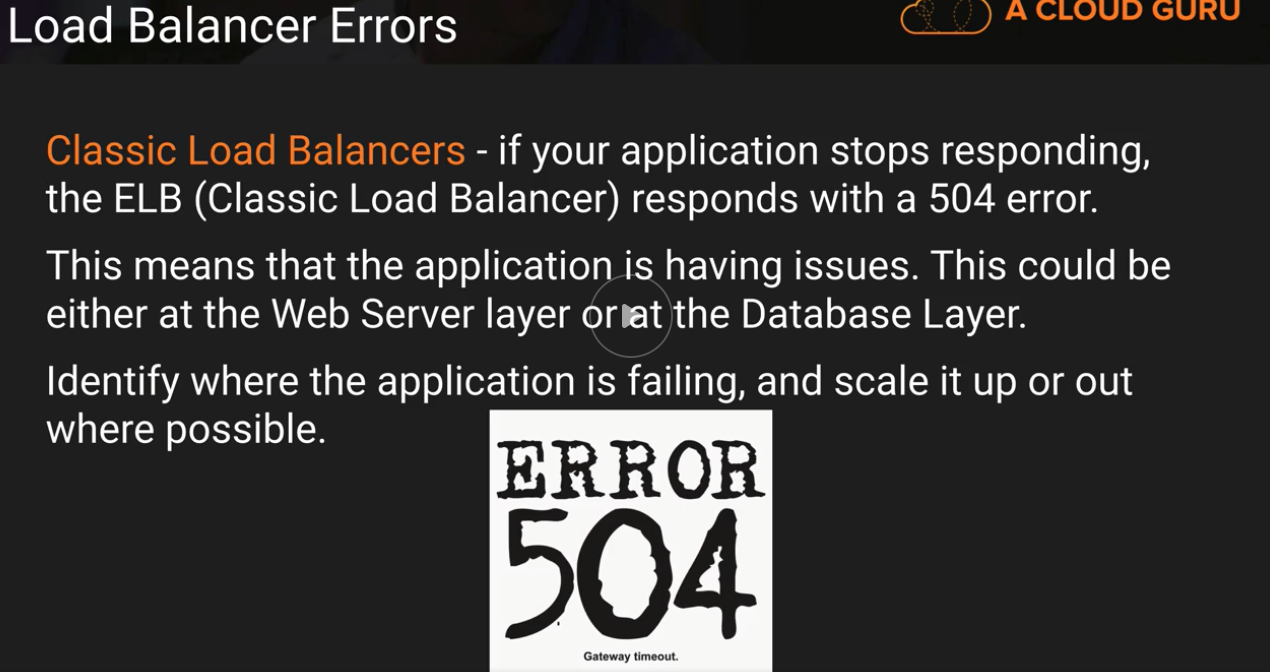
Ultra low latency

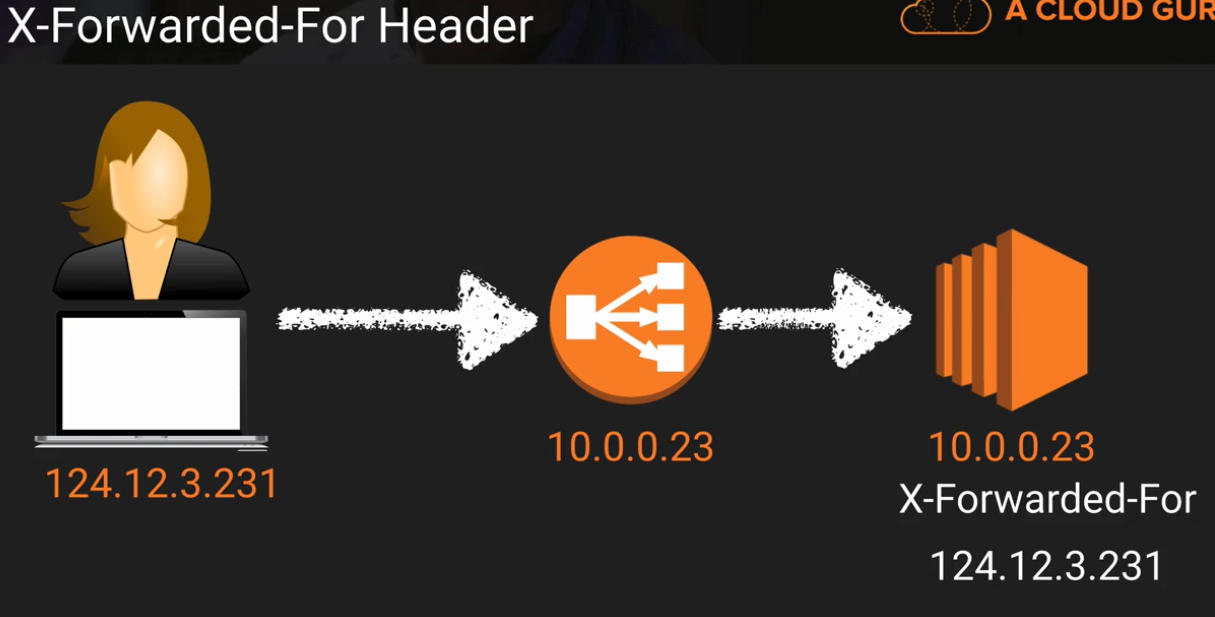


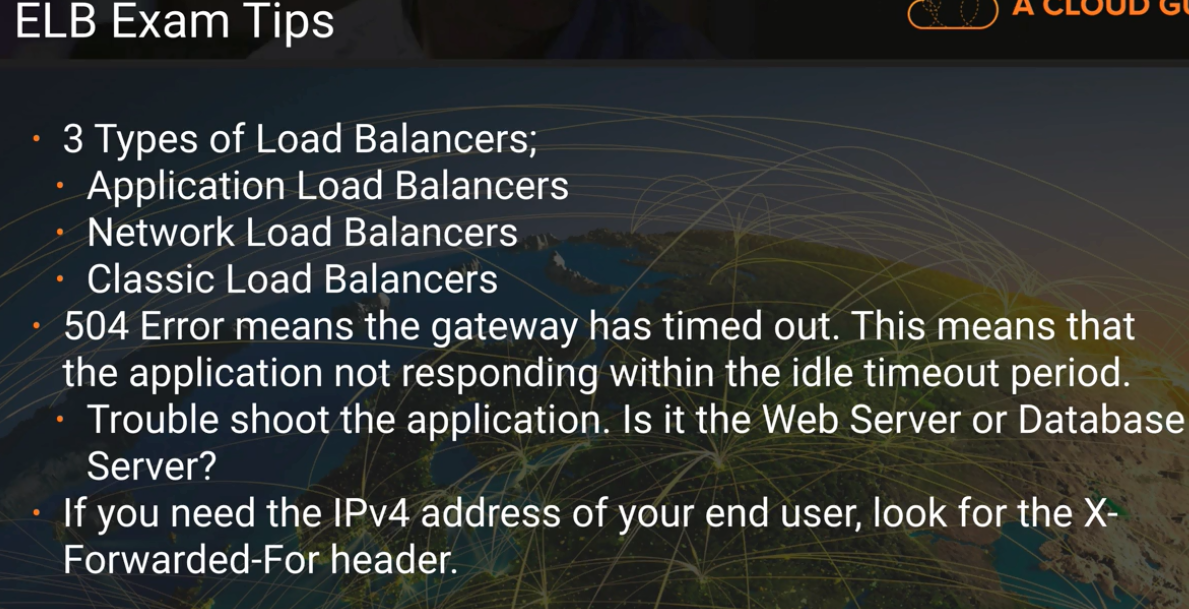
Does both

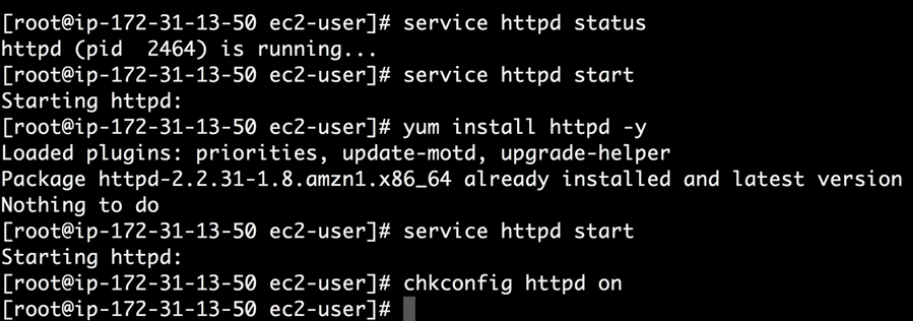
Old Elastic load balancers

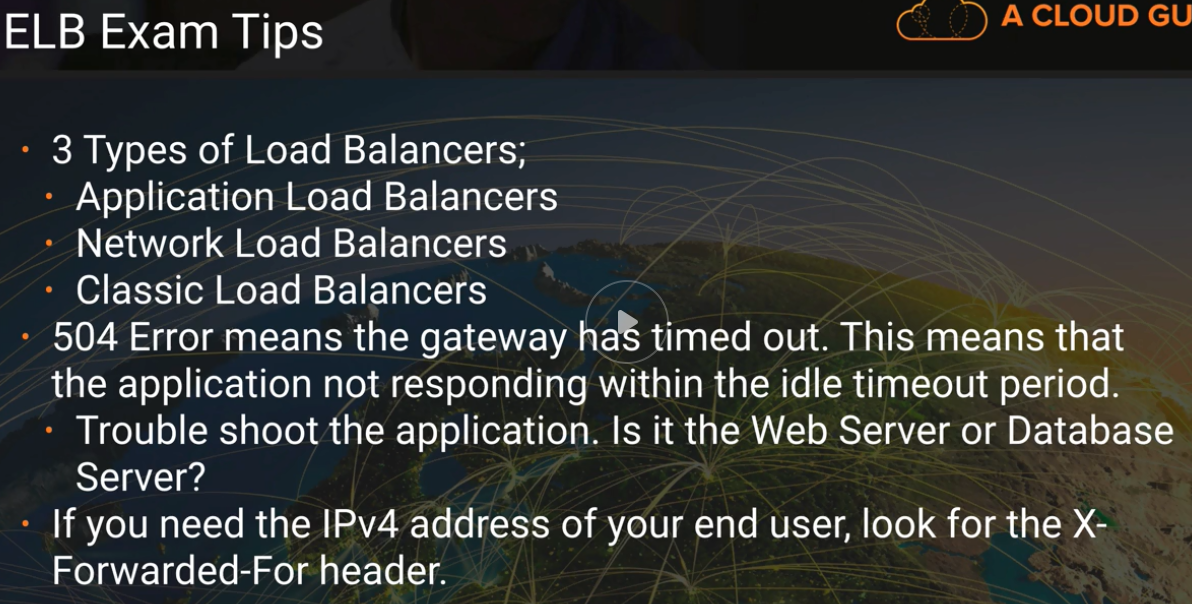


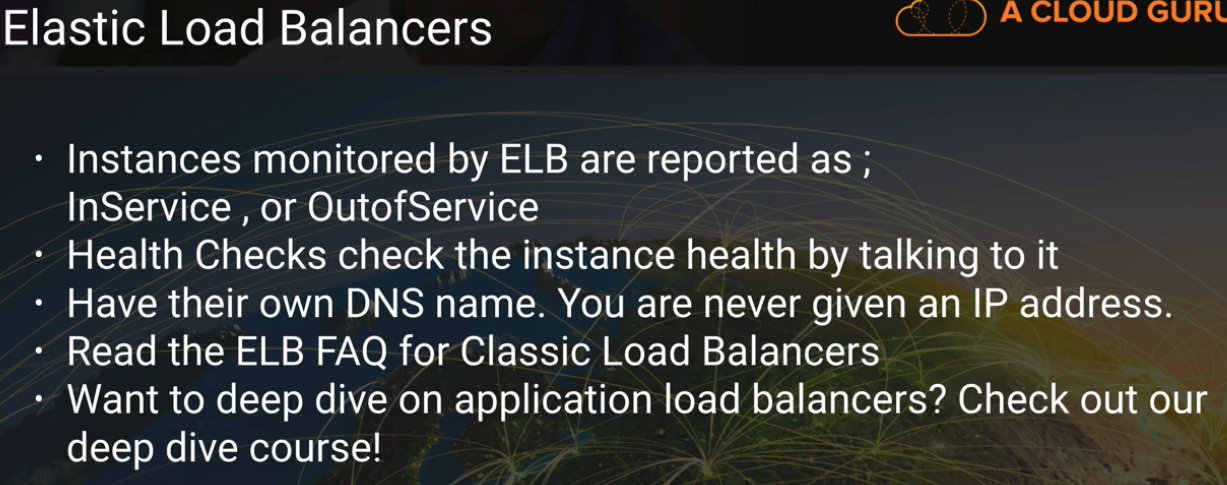












Cloud Watch

For monitoring

Cloud trail – for auditing – create role, etc.

**Amazon CloudWatch**

*Amazon CloudWatch* is a service that you can use to monitor your AWS resources and your applications in real time. With Amazon CloudWatch, you can collect and track metrics, create alarms that send notifications, and make changes to the resources being monitored based on rules you define.

basic or detailed monitoring

*Basic monitoring* sends data points to Amazon CloudWatch every five minutes for a limited number of preselected metrics at no charge. Default.

*Detailed monitoring* sends data points to Amazon CloudWatch every minute and allows data aggregation for an additional charge.    Aggregate metrics

metrics can be retrieved by performing a GET request.

Amazon CloudWatch does not aggregate data across regions but can aggregate across Availability Zones within a region

CloudWatch supports an Application Programming Interface (API) that allows programs and scripts to PUT metrics into Amazon CloudWatch as name-value pairs that can then be used to create events and trigger alarms in the same manner as the default Amazon CloudWatch metrics

Amazon CloudWatch Logs – to monitor, store, and access log files from Amazon EC2 instances, AWS CloudTrail, and other sources. To store logs logs in S3 and glacier

Cloud Watch agent  install on EC2 instance  Agent send data to cloud watch

Each AWS account is limited to 5,000 alarms per AWS account

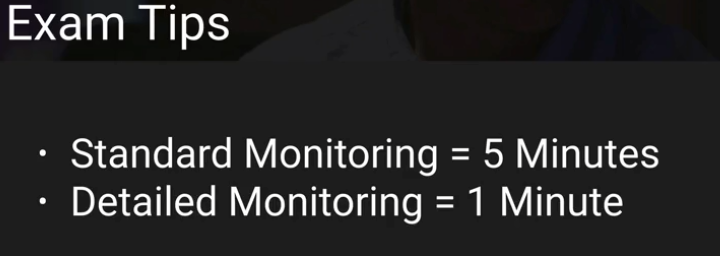
metrics data is retained for two weeks by default

Create Dashboard

Alarms – sent notification or executing an Auto Scaling policy.

Events – take some action based on some cation – call lamda function

Metrics





**Auto Scaling**

**Auto Scaling Plans**

**Maintain Current Instance Levels**

You can configure your Auto Scaling group to maintain a minimum or specified number of running instances at all times

To maintain the current instance levels, Auto Scaling performs a periodic health check on running instances within an *Auto Scaling group*. When Auto Scaling finds an unhealthy instance, it terminates that instance and launches

a new one.

Steady state workloads that need a consistent number of Amazon EC2 instances at all times can use Auto Scaling to monitor and keep that specific number of Amazon EC2 instances running

**Manual Scaling**

Manual scaling is the most basic way to scale your resources. You only need to specify the change in the maximum, minimum, or desired capacity of your Auto Scaling group. Auto Scaling manages the process of creating or terminating instances to maintain the updated capacity.

useful to increase resources for an infrequent event, such as the release of a new game version that will be available for download and require a user registration. For extremely large-scale events, even the Elastic

Load Balancing load balancers can be pre-warmed by working with your local solutions architect or AWS Support.

**Scheduled Scaling**

Recurring events such as end-of-month, quarter, or year processing, or scheduled and recurring automated load and performance testing, can be anticipated and Auto Scaling can be ramped up appropriately at the time of the scheduled event

**Dynamic Scaling**

Dynamic scaling lets you define parameters that control the Auto Scaling process in a scaling policy. For example, you might create a policy that adds more Amazon EC2

instances to the web tier when the network bandwidth, measured by Amazon CloudWatch, reaches a certain threshold.

**Auto Scaling Components**

**Launch Configuration (limit – 100 per region )**

template that Auto Scaling uses to create new instances, and it is composed of the configuration name, *Amazon Machine Image (AMI)*, Amazon EC2 instance type, security group, and instance key pair. Each Auto Scaling group can

have only one launch configuration at a time



The CLI command that follows will create a launch configuration with the following attributes:

Name: myLC

AMI: ami-0535d66c

Instance type: m3.medium

Security groups: sg-f57cde9d

Instance key pair: myKeyPair

> aws autoscaling create-launch-configuration -–launch-configuration-name

myLC --image-id ami-0535d66c --instance-type m3.medium --security-groups sgf57cde9d

--key-name myKeyPair

The default limit for launch configurations is 100 per region

Update using below command

default number of Amazon EC2 instances you can currently launch within a region, which is 20

aws autoscaling describe-account-limits

**Auto Scaling Group**

An Auto Scaling group is a collection of Amazon EC2 instances managed by the Auto Scaling service

contains configuration options that control when Auto Scaling should launch new instances and terminate existing instances

An Auto Scaling group must contain a name and a minimum and maximum number of instances that can be in the group. You can optionally specify desired capacity, which is the number

of instances that the group must have at all times. If you don’t specify a desired capacity, the default desired capacity is the minimum number of instances that you specify.

Name: myASG

Launch configuration: myLC

Availability Zones: us-east-1a and us-east-1c

Minimum size: 1

Desired capacity: 3

Maximum capacity: 10

Load balancers: myELB

> aws autoscaling create-auto-scaling-group --auto–scaling-group-name myASG

--launch-configuration-name myLC --availability-zones us-east-1a, us-east-1c

--min-size 1 --max-size 10 --desired-capacity 3 --load-balancer-names myELB

An Auto Scaling group can use either On-Demand or Spot Instances. Default is on-demand

Auto Scaling supports using cost-effective Spot Instances

This can be very useful when you are hosting sites where you want to provide additional compute capacity

but are price constrained. An example is a “freemium” site model where you may

offer some basic functionality to users for free and additional functionality for

premium users who pay for use. Spot Instances can be used for providing the basic

functionality when available by referencing a maximum bid price in the launch

configuration (—spot-price "0.15") associated with the Auto Scaling group.

**Scaling Policy**

You can associate Amazon CloudWatch alarms and *scaling policies* with an Auto Scaling group to adjust Auto Scaling dynamically

When a threshold is crossed, Amazon CloudWatch sends alarms to trigger changes (scaling in or out) to the number of Amazon EC2 instances currently receiving traffic behind a load balancer.

You can associate more than one scaling policy with an Auto Scaling group A recommended best practice is to scale out quickly and scale in slowly so you can

respond to bursts or spikes but avoid inadvertently terminating Amazon EC2 instances too quickly, only having to launch more Amazon EC2 instances if the burst is sustained.

Auto Scaling also supports a *cooldown period*, which is a configurable setting that determines when to suspend scaling activities for a short time for an Auto Scaling group.

**Rolling Out a Patch at Scale**

* In large deployments of Amazon EC2 instances, Auto Scaling can be used to make rolling out a patch to your instances easy.

The launch configuration associated with the Auto Scaling group may be modified to reference a new AMI and even a new Amazon EC2 instance if needed.

* Then you can deregister or terminate instances one at a time or in small groups, and the new Amazon EC2 instances will reference the new patched AMI.

**Benefits:**

Better fault tolerance - You can also configure Amazon EC2 Auto Scaling to use multiple Availability Zones

Better availability - Amazon EC2 Auto Scaling can help you ensure that your application always has the right amount of capacity to handle the current traffic demand.

Better cost management - Amazon EC2 Auto Scaling can dynamically increase and decrease capacity as needed.

**Dynamic scaling policies**

Target tracking scaling policy - You can also configure Amazon EC2 Auto Scaling to use multiple Availability Zones. Example av CPU – 50 %

Simple scaling – Simple scaling policies must wait for the cooldown period to expire after a scaling activity or health check replacement before they can respond to alarms that are breached. For more information

Step scaling - Scaling policies with steps continuously evaluate alarms as they are breached, even while a scaling activity or health check replacement is in progress

**Default Termination Policy**

The default termination policy is designed to help ensure that your network architecture spans Availability Zones evenly. With the default termination policy, the behavior of the Auto Scaling group is as follows:

1. If there are instances in multiple Availability Zones, select the Availability Zone with the most instances and at least one instance that is not protected from scale in. If there is more than one Availability Zone with this number of instances, select the Availability Zone with the instances that use the oldest launch configuration.
2. Determine which unprotected instances in the selected Availability Zone use the oldest launch configuration. If there is one such instance, terminate it.
3. If there are multiple instances that use the oldest launch configuration, determine which unprotected instances are closest to the next billing hour. (This helps you maximize the use of your EC2 instances and manage your Amazon EC2 usage costs.) If there is one such instance, terminate it.
4. If there is more than one unprotected instance closest to the next billing hour, select one of these instances at random.

## Instance Protection

To control whether an Auto Scaling group can terminate a particular instance when scaling in, use instance protection.

You can use AWS Auto Scaling to configure automatic scaling for the following scalable resources:

* Aurora DB clusters
* Auto Scaling groups
* DynamoDB GSIs
* DynamoDB tables
* ECS services
* Spot Fleet requests

AWS Auto Scaling searches for scalable resources as follows:

* If the application source is a CloudFormation stack, search for scalable resources defined in the stack.
* If the application source is a set of tags, search for scalable resources with all of the specified tags.

Thanks & Regards,

Shantaram Vernekar

**Elastic Load balancing**

<https://aws.amazon.com/documentation/elastic-load-balancing/>

**Classic load balancers**

If a load balancer is in a VPC with ClassicLink enabled, its instances can be linked EC2-Classic instances. If a load balancer is in EC2-Classic, its instances must be in EC2-Classic.

Load balancers in a VPC support IPv4 addresses only.

*name*-*1234567890*.*region*.elb.amazonaws.com

Load balancers in EC2-Classic support both IPv4 and IPv6 addresses.

*name*-*123456789*.*region*.elb.amazonaws.com

ipv6.*name*-*123456789*.*region*.elb.amazonaws.com

dualstack.*name*-*123456789*.*region*.elb.amazonaws.com

The base public DNS name returns only IPv4 records. The public DNS name with the ipv6 prefix returns only IPv6 records. The public DNS name with the dualstack prefix returns both IPv4 and IPv6 records

We recommend that you enable IPv6 support by using the DNS name with the dualstack prefix to ensure that clients can access the load balancer using either IPv4 or IPv6.

Clients can connect to your load balancer in EC2-Classic using either IPv4 or IPv6. However, communication between the load balancer and its back-end instances uses only IPv4, regardless of how the client communicates with your load balancer.

When you create a load balancer in a VPC, you can make it an internal load balancer or an Internet-facing load balancer.

You create an Internet-facing load balancer in a public subnet. Load balancers in EC2-Classic are always Internet-facing load balancers.

A listener checks for connection requests from clients, using the protocol and port that you configure, and forwards requests to one or more registered instances using the protocol and port number that you configure. You add one or more listeners to your load balancer.

You can configure health checks, which are used to monitor the health of the registered instances so that the load balancer only sends requests to the healthy instances.

By default, the load balancer distributes traffic evenly across the Availability Zones that you enable for your load balancer.

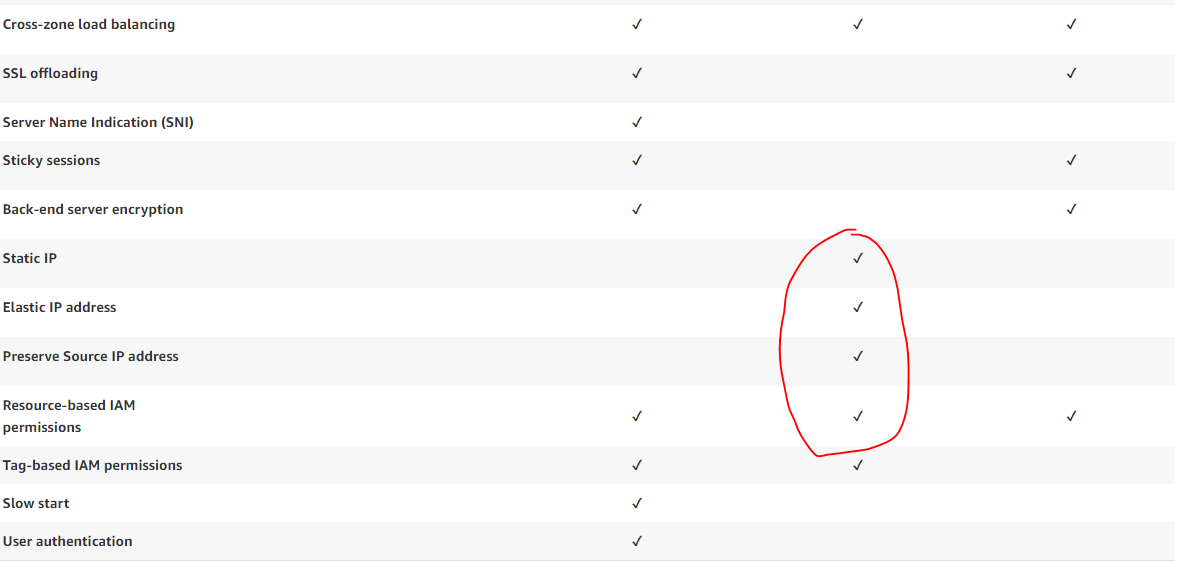
To distribute traffic evenly across all registered instances in all enabled Availability Zones, enable **cross-zone load balancing** on your load balancer. However, we still recommend that you maintain approximately equivalent numbers of instances in each Availability Zone for better fault tolerance.

Classic Load Balancer v/s Application Load Balancer has the following benefits:

* Support for EC2-Classic (Classic LB supports both VPC and EC2 classic)
* Support for TCP and SSL listeners
* Support for sticky sessions using application-generated cookies

<https://aws.amazon.com/elasticloadbalancing/details/#compare>

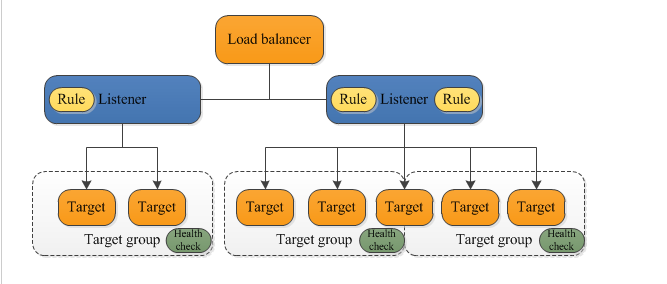




**Application load balancers**

A *listener* checks for connection requests from clients, using the protocol and port that you configure, and forwards requests to one or more target groups, based on the rules that you define. Each rule specifies a target group, condition, and priority. When the condition is met, the traffic is forwarded to the target group. You must define a default rule for each listener, and you can add rules that specify different target groups based on the content of the request (also known as *content-based routing*).

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.



* **Loadbalancers**
* **Listeners**
* **targetGroups**

After the load balancer receives a request, it evaluates the listener rules in priority order to determine which rule to apply, and then selects a target from the target group for the rule action. You can configure listener rules to route requests to different target groups based on the content of the application traffic. Routing is performed independently for each target group, even when a target is registered with multiple target groups

You can configure **health checks,** which are used to monitor the health of the registered targets so that the load balancer can send requests only to the healthy targets..

**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-based routing. 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-based routing. 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 requests to multiple applications on a single EC2 instance. You can register each instance or IP address with the same target group using multiple ports.
* 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.
* Improved load balancer performance.

**Network Load balancers**

* **Loadbalancers**
* **Listeners**
* **targetGroups**

## Benefits of Migrating from a Classic Load Balancer

Using a Network Load Balancer instead of a Classic Load Balancer has the following benefits:

* Ability to handle volatile workloads and scale to millions of requests per second.
* Support for static IP addresses for the load balancer. You can also assign one Elastic IP address per subnet enabled for the load balancer.
* Support for registering targets by IP address, including targets outside the VPC for the load balancer.
* Support for routing requests to multiple applications on a single EC2 instance. You can register each instance or IP address with the same target group using multiple ports.
* 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 Amazon 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.

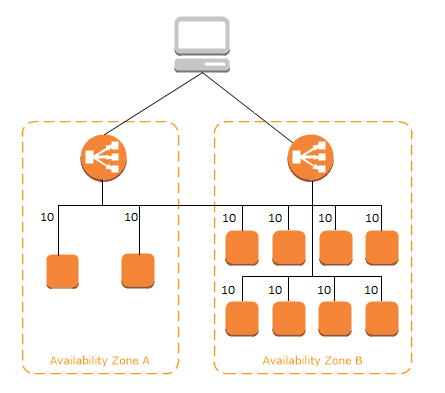
Availability zone

When you enable an Availability Zone for your load balancer, Elastic Load Balancing creates a load balancer node in the Availability Zone

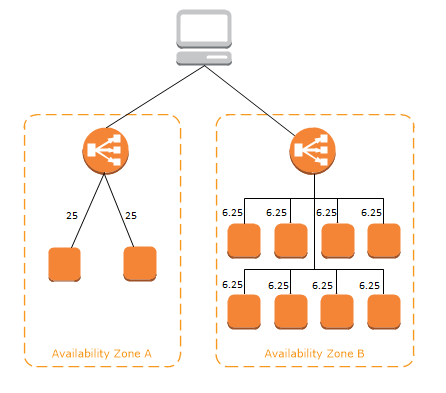
### **Cross-Zone Load Balancing**

* With Application Load Balancers, cross-zone load balancing is always enabled.
* 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
* Classic Load Balancer
* With the API or CLI, cross-zone load balancing is disabled by default
* With the AWS Management Console, the option to enable cross-zone load balancing is selected by default
* After you create a Classic Load Balancer, you can enable or disable cross-zone load balancing at any time

If cross-zone load balancing is enabled, each of the 10 targets receives 10% of the traffic. This is because each load balancer node can route its 50% of the client traffic to all 10 targets.



If cross-zone load balancing is disabled, each of the 2 targets in Availability Zone A receives 25% of the traffic and each of the 8 targets in Availability Zone B receives 6.25% of the traffic. This is because each load balancer node can route its 50% of the client traffic only to targets in its Availability Zone.



### Routing Algorithm

Application LB - using the round robin routing algorithm

Network -  flow hash algorithm

Elastic LB -

round robin routing algorithm for TCP listeners

least outstanding requests routing algorithm for HTTP and HTTPS listeners.

### HTTP Headers

Application Load Balancers and Classic Load Balancers support **X-Forwarded-For**, **X-Forwarded-Proto**, and **X-Forwarded-Port** headers.

### HTTP Connections

### Classic Load Balancers use pre-open connections but Application Load Balancers do not.

### Both Classic Load Balancers and Application Load Balancers use connection multiplexing.

Elastic Load Balancing works with the following services to improve the availability and scalability of your applications.

* **Amazon EC2**
* **ECS**
* **Auto Scaling**
* **Cloudwatch**
* **Route 53**

**Connection Draining**stops the load balancer sending traffic to faulty instances.

If connection draining is enabled, then LB do not direct traffic to unhealthy VMs

Allows the inflight request to complete

If inflight request do not complete within timeout (1 sec to 300 secs) then inflight request are forcibly closed

If connection draining is not enabled, then inflight requests are immediately closed

Slow Start Mod

Using slow start mode gives targets time to warm up before the load balancer sends them a full share of requests. After you enable slow start for a target group, targets enter slow start mode when they are registered with the target group and exit slow start mode when the configured slow start duration period elapses. The load balancer linearly increases the number of requests that it can send to a target in slow start mode. e

**Proxy Protocol**: To receive the clients IP and User Agent, it needs to be enabled. (only on Network Load Balancer and Classic Load Balancer)

**X-Forwarded-For – HTTP header for getting client IP (Application and classic load balancer..)**

# Lambda@Edge

Lambda@Edge lets you run Lambda functions to customize content that CloudFront delivers, executing the functions in AWS locations closer to the viewer. The functions run in response to CloudFront events, without provisioning or managing servers. You can use Lambda functions to change CloudFront requests and responses at the following points:

* Functions can access a **single** VPC not multiple.
* The **minimum memory is 128 MB** and **maximum 3008 MB**. Functions will be terminated if it uses more than 3008 MB.

Thanks & Regards,

Shantaram Vernekar