Elastic Load Balancing

General ELB Concepts

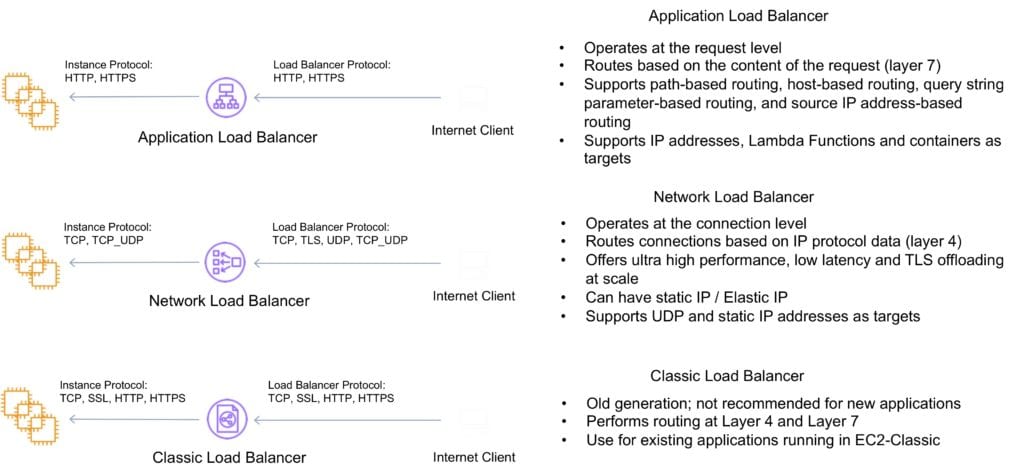
Elastic Load Balancing automatically distributes incoming application traffic across multiple targets, such as Amazon EC2 instances, containers, and IP addresses.

There are three types of Elastic Load Balancer (ELB) on AWS:

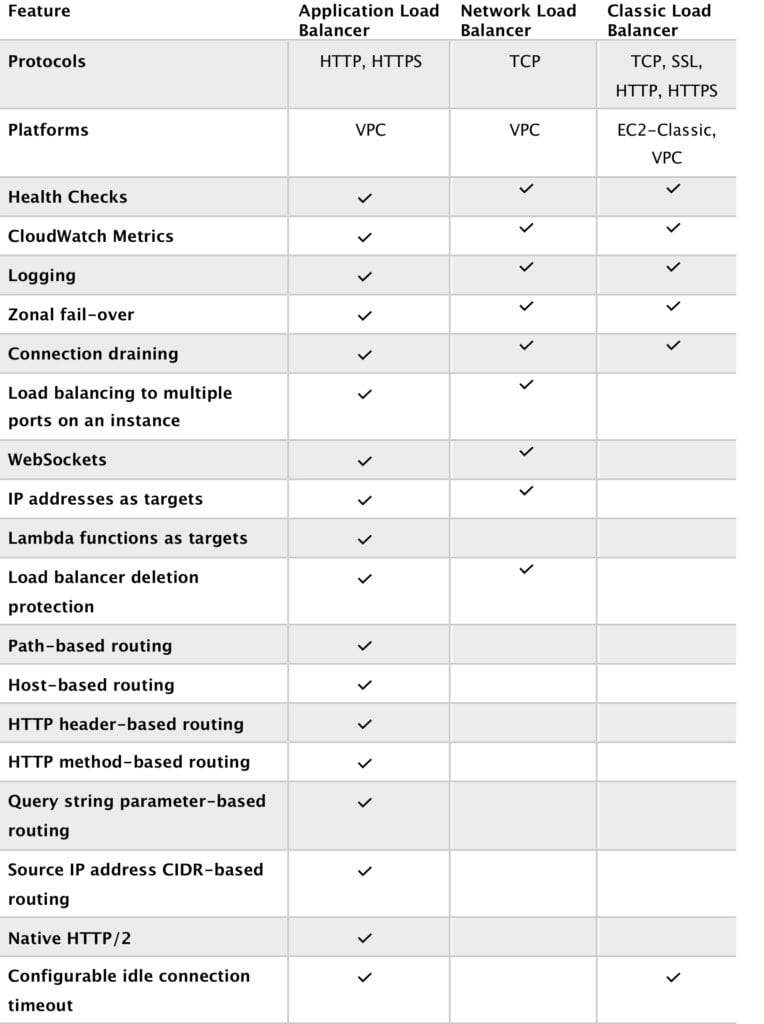
* Classic Load Balancer (CLB) – this is the oldest of the three and provides basic load balancing at both layer 4 and layer 7.
* Application Load Balancer (ALB) – layer 7 load balancer that routes connections based on the content of the request.
* Network Load Balancer (NLB) – layer 4 load balancer that routes connections based on IP protocol data.

Note: The Classic Load Balancer may be phased out over time and Amazon are promoting the ALB and NLB for most use cases within VPC.

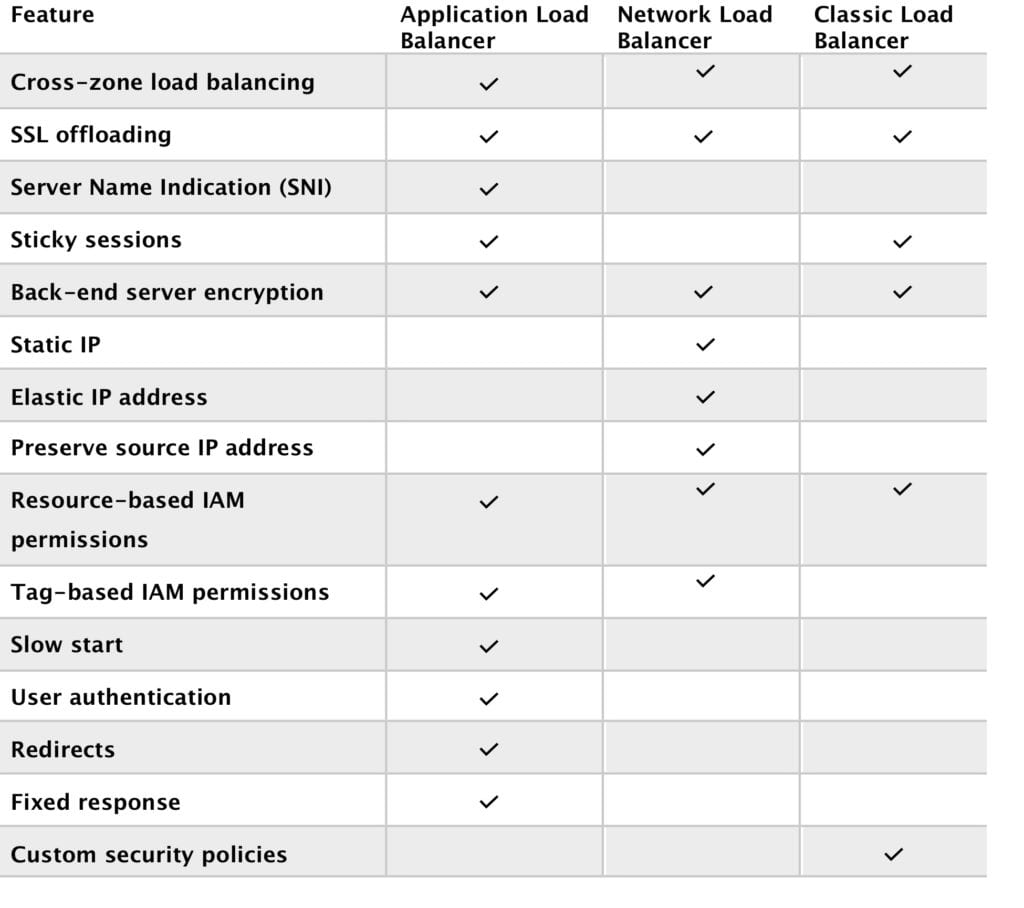
The following image provides an overview of some of the key differences between the three types of ELB:



The following table provides a more detailed feature comparison:



And continued….



Elastic Load Balancing provides fault tolerance for applications by automatically balancing traffic across targets – Amazon EC2 instances, containers and IP addresses – and Availability Zones while ensuring only healthy targets receive traffic.

An ELB can distribute incoming traffic across your Amazon EC2 instances in a single Availability Zone or multiple Availability Zones.

Only 1 subnet per AZ can be enabled for each ELB.

Route 53 can be used for region load balancing with ELB instances configured in each region.

ELBs can be **Internet** facing or **internal-only.**

Internet facing ELB:

* ELB nodes have public IPs.
* Routes traffic to the private IP addresses of the EC2 instances.
* Need one public subnet in each AZ where the ELB is defined.
* ELB DNS name format: <name>-<id-number>.<region>.elb.amazonaws.com.

Internal only ELB:

* ELB nodes have private IPs.
* Routes traffic to the private IP addresses of the EC2 instances.
* ELB DNS name format: **internal**-<name>-<id-number>.<region>.elb.amazonaws.com.

Internal-only load balancers do not need an Internet gateway.

EC2 instances and containers can be registered against an ELB.

ELB nodes use IP addresses within your subnets, ensure at least a /27 subnet and make sure there are at least 8 IP addresses available in order for the ELB to scale.

An ELB forwards traffic to eth0 (primary IP address).

An ELB listener is the process that checks for connection requests:

* Listeners for CLB provide options for TCP and HTTP/HTTPS.
* Listeners for ALB only provide options for HTTP and HTTPS.
* Listeners for NLB only provide TCP as an option.

Deleting an ELB does not affect the instances registered against it (they won’t be deleted, they just won’t receive any more requests).

For ALB at least 2 subnets must be specified.

For NLB only one subnet must be specified (recommended to add at least 2).

For CLB you don’t need to specify any subnets unless you have “Enable advanced VPC configuration” enabled in which case you must specify two.

ELB uses a DNS record TTL of 60 seconds to ensure new ELB node IP addresses are used to service clients.

By default the ELB has an idle connection timeout of 60 seconds, set the idle timeout for applications to at least 60 seconds.

Perfect Forward Secrecy (PFS) provides additional safeguards against the eavesdropping of encrypted data, through the use of a unique random session key.

Server Order Preference lets you configure the load balancer to enforce cipher ordering, providing more control over the level of security used by clients to connect with your load balancer.

ELB does not support client certificate authentication (API Gateway does support this).

ELB Security Groups

Security groups control the ports and protocols that can reach the front end listener.

In non-default VPCs you can choose which security group to assign.

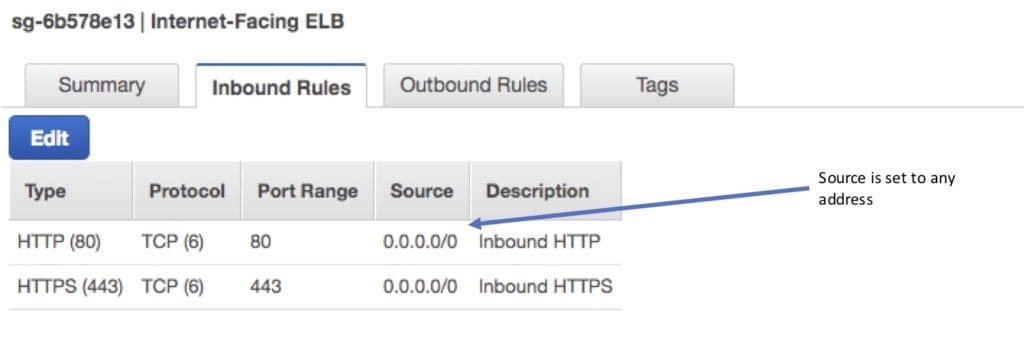
You must assign a security group for the ports and protocols on the front-end listener.

You need to also allow the ports and protocols for the health check ports and back-end listeners.

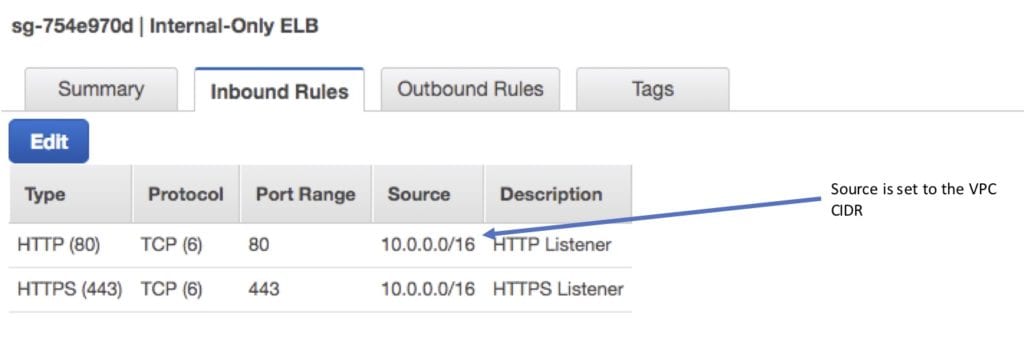
Security group configuration for ELB:

Inbound to ELB (allow).

* Internet-facing ELB:
  + Source: 0.0.0.0/0.
  + Protocol: TCP.
  + Port: ELB listener ports.

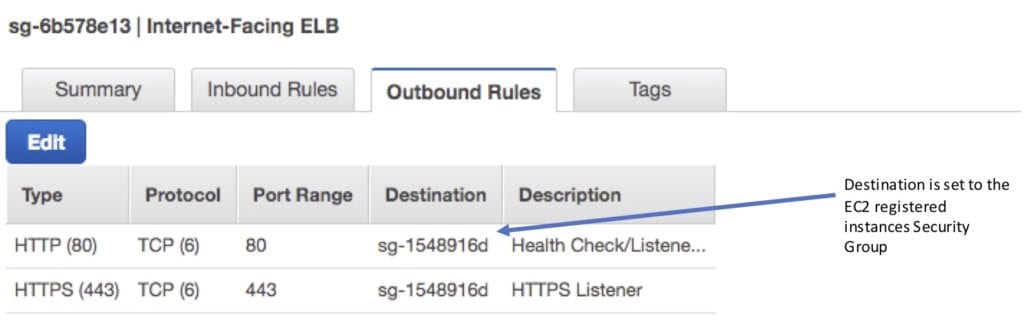


* Internal-only ELB:
  + Source: VPC CIDR.
  + Protocol: TCP.
  + Port: ELB Listener ports.



Outbound (allow, either type of ELB):

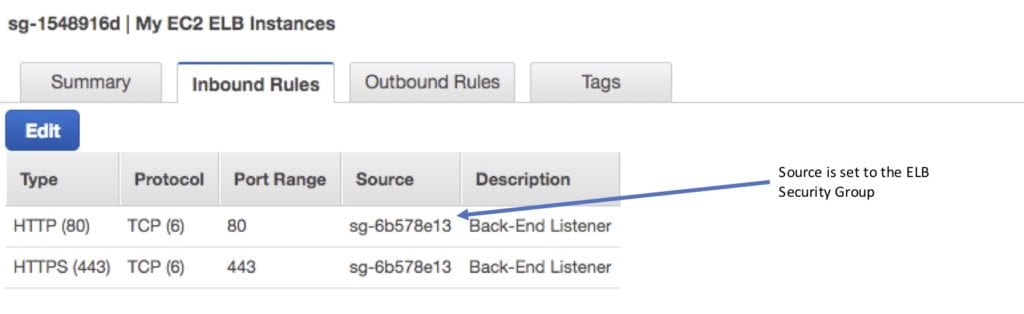
* Destination: EC2 registered instances security group.
* Protocol: TCP.
* Port: Health Check/Listener.



Security group configuration for registered instances:

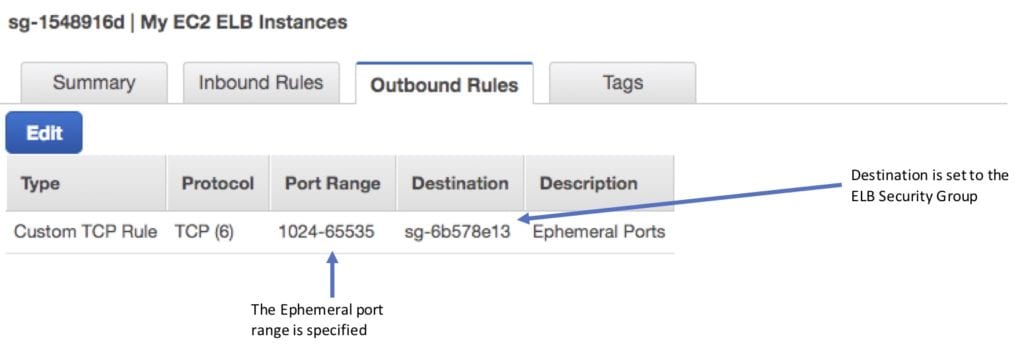
Inbound to registered instances (Allow, either type of ELB).

* Source: ELB Security Group.
* Protocol: TCP.
* Port: Health Check/Listener.



Outbound (Allow, for both types of ELB).

* Destination: ELB Security Group.
* Protocol: TCP.
* Port: Ephemeral.



It is also important to ensure NACL settings are set correctly.

Distributed Denial of Service (DDoS) protection:

* ELB automatically distributes incoming application traffic across multiple targets, such as Amazon Elastic Compute Cloud (Amazon EC2) instances, containers, and IP addresses, and multiple Availability Zones, which minimizes the risk of overloading a single resource.
* ELB, like CloudFront, only supports valid TCP requests, so DDoS attacks such as UDP and SYN floods are not able to reach EC2 instances.
* ELB also offers a single point of management and can serve as a line of defense between the internet and your backend, private EC2 instances.

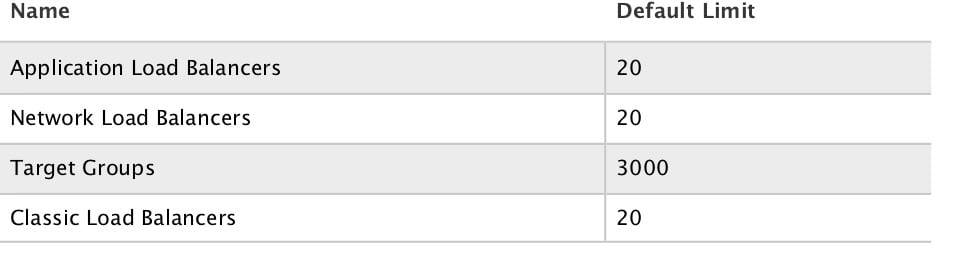
ELB Monitoring

Monitoring takes place using:

* CloudWatch – every 1 minute.
  + ELB service only sends information when requests are active.
  + Can be used to trigger SNS notifications.
* Access Logs.
  + Disabled by default.
  + Includes information about the clients (not included in CloudWatch metrics).
  + Can identify requester, IP, request type etc.
  + Can be optionally stored and retained in S3.
* CloudTrail.
  + Can be used to capture API calls to the ELB.
  + Can be stored in an S3 bucket.

Limits

The following table details the default limits for your account on a per-region basis:



Classic Load Balancer (CLB)

The Classic Load Balancer provides basic load balancing across multiple Amazon EC2 instances and operates at both the request level and connection level.

Operates at layer 4 and layer 7.

Supported protocols: TCP, SSL, HTTP, HTTPS.

CLB does not support HTTP/2.

Load balancers can listen on the following ports:

* [EC2-VPC] 1-65535.
* [EC2-Classic] 25, 80, 443, 465, 587, 1024-65535.

CLB’s do not have pre-defined IPv4 addresses but are resolved using a DNS name.

Does not support Elastic IPs.

Supports IPv4 and IPv6.

Within a VPC only IPv4 is supported.

Provides SSL termination and processing.

Sticky Sessions:

* Cookie-based sticky sessions are supported.
* Session stickiness uses cookies and ensures a client is bound to an individual back-end instance for the duration of the cookie lifetime.
* Cookies can be inserted by the application or by the load balancer when configured.
* After cookies expire new requests will be routed by the load balancer normally and a new cookie will be inserted and bind subsequent sessions to the same back-end instance.
* With application-inserted cookies if the back-end instance becomes unhealthy, new requests will be routed by the load balancer normally and a new cookie will be inserted and bind subsequent sessions to the same back-end instance.
* With CLB-inserted cookies if the back-end instance becomes unhealthy, new requests will be routed by the load balancer normally BUT the session will no longer be sticky.

Must have multiple CLBs for multiple SSL certs.

Integrates with Auto Scaling, CloudWatch, CloudTrail and Route 53.

Instances monitored by CLB are reported as InService or OutofService.

Supports domain zone apex records, e.g. example.com.

Wildcard certificates are supported.

Health checks:

* Can be configured for HTTP, TCP, HTTPS, SSL.
* Ping port specifies the port for the health check.
* Ping path specifies the path to check, e.g. /index.html.
* Can define timeout, interval, unhealthy threshold, healthy threshold.

For fault tolerance it is recommended to distribute registered instances across multiple AZs (ideally evenly).

**Cross-zone load balancing:**

* Cross-zone load balancing is enabled by default for CLB and ALB but not for NLB (when created through the console).
* Cross-zone load balancing is NOT enabled by default if the CLB is created from the CLI or API.
* You can enable or disable cross-zone load balancing on the CLB and NLB at any time.
* For the ALB, cross-zone load balancing is always on and cannot be disabled.
* When cross-zone load balancing is enabled, each load balancer node distributes traffic across the registered targets in all enabled Availability Zones.
* When cross-zone load balancing is disabled, each load balancer node distributes traffic across the registered targets in its Availability Zone only.

Connection draining is enabled by default and provides a period of time for existing connections to close cleanly.

When connection draining is in action a CLB will be in the status “InService: Instance deregistration currently in progress”.

CLB can take 1 to 7 minutes to detect an increase in load and scale.

If you’re anticipating a fast increase in load you can contact AWS and instruct them to pre-warm (provision) additional CLB nodes.

**Listeners:**

* A CLB listener is the process that checks for connection requests.
* You can configure the protocol/port on which your CLB listener listens.
* Front-end listeners check for traffic from clients to the CLB.
* Back-end listeners are configured with the protocol/port to check for traffic from the CLB to the EC2 instances.
* Front-end and back-end listeners can listen on ports 1-65535.
* Front-end and back-end listeners must be at the same layer (e.g. layer 4 or layer 7).
* There is a 1:1 mapping between front-end and back-end listeners.
* Up to 100 listeners can be configured.
* Supports L4 (TCP, SSL) and L7 (HTTP, HTTPS) listeners.

With packet interception the source IP/port will be from the ELB.

Proxy protocol for TCP/SSL carries the source (client) IP/port information.

The Proxy Protocol header helps you identify the IP address of a client when you have a load balancer that uses TCP for back-end connections.

Ensure the client doesn’t go through a proxy or there will be multiple proxy headers.

Also need to ensure the EC2 instance’s TCP stack can process the extra information.

X-forwarded-for for HTTP/HTTPS carries the source IP/port information.

To use an HTTPS listener the CLB must have an X.509 SSL/TLS server certificate – this will allow the CLB to terminate the secure session from the client to the CLB.

The session between the CLB and the EC2 instance can be re-encrypted.

You can use a certificate generated by AWS Certificate Manager (ACM) or your own certificate.

If you don’t want interception/offloading you can use TCP listeners with certificates on the EC2 instances (traffic is secured end-to-end).

Proxy protocol only applies to L4.

X-forwarded-for only applies to L7.

To filter by source IP use NACLs for proxy protocol (L4) / X-forwarded-for (L7) headers with the EC2  instance’s application performing the filtering.

Security

CLB supports a single X.509 certificate.

Two way authentication with client certificates is not supported on the CLB – you would need to pass through the session using the proxy protocol and have an application that supports client-side certificates.

When using end-to-end encryption use TCP not SSL/HTTPS on the CLB (does not support Session Stickiness).

AWS ACM certificates include an RSA public key – ensure you include a set of ciphers that support RSA in the security policy.

The latest predefined security policy does not include support for SSLv3.

When choosing a custom security policy you can select the ciphers and protocols (only for CLB).

SSL Security Policy includes:

* Protocol Versions (SSL/TLS).
  + Supports TLS 1.0, 1.1, 1.2, SSL 3.0.
* SSL Ciphers.
  + Encryption algorithms.
  + SSL can use different ciphers to encrypt data.
* Server Order Preference.
  + When enabled the first match in the cipher list with the Client list is used.

If disabled (default) the first match in the client cipher list with the CLB is used.

Application Load Balancer (ALB)

The Application Load Balancer operates at the request level (layer 7), routing traffic to targets – EC2 instances, containers and IP addresses based on the content of the request.

You can load balance HTTP/HTTPS applications and use layer 7-specific features, such as X-Forwarded-For headers.

Supports HTTPS termination between the clients and the load balancer.

Supports management of SSL certificates through AWS IAM and AWS Certificate Manager for pre-defined security policies.

Server Name Indication (SNI) supports multiple secure websites using a single secure listener.

With Server Name Indication a client indicates the hostname to connect to.

IP addresses as targets allows load balancing any application hosted in AWS or on-premises using IP addresses of the application back-ends as targets.

Need at least 2 availability zones and you can distribute incoming traffic across your targets in multiple Availability Zones.

Automatically scales its request handling capacity in response to incoming application traffic.

Can configure an Application Load Balancer to be Internet facing or create a load balancer without public IP addresses to serve as an internal (non-Internet-facing) load balancer.

Native IPv6 support.

Internal only ALB only supports IPv4.

Content-Based Routing allows the routing of requests to a service based on the content of the request:

* Host-based routing – route client requests based on the Host field of the HTTP header allowing you to route to multiple domains from the same load balancer.
* Path-based routing – route a client request based on the URL path of the HTTP header (e.g. /images or /orders).

Provides support for micro-services and containers with load balancing across multiple ports on a single EC2 instance.

Better performance for real-time streaming.

Deletion protection can be enabled.

Request tracing (allows you to track a request by its unique ID).

Better health checks and CloudWatch metrics.

Integration with Amazon Cognito for user authentication.

Uses a round-robin load balancing algorithm.

Slow start mode allows targets to “warm up” with a ramp-up period.

Health Checks:

* Can have custom response codes in health checks (200-399).
* There are more details provided in the API and management console for health check failures.
* Reason codes are returned with failed health checks.
* Health checks do not support WebSockets.
* Fail open means if no AZ contains a healthy target, the load balancer nodes route requests to all targets.

Detailed access log information is provided and saved to an S3 bucket every 5 or 6 minutes.

ALB does not support back-end server authentication (CLB does).

ALB does not support EC2-Classic (CLB does).

Deletion protection is possible.

Deregistration delay is similar to connection draining.

**Sticky Sessions:**

* Session stickiness uses cookies and ensures a client is bound to an individual back-end instance for the duration of the cookie lifetime.
* ALB supports load balancer-generated cookies only.
* The name of the cookie is AWSALB.
* The contents of these cookies are encrypted using a rotating key.
* You cannot decrypt or modify load balancer-generated cookies.
* Sticky sessions are enabled at the target group level.
* You can also set the duration for the stickiness of the load balancer-generated cookie, in seconds.
* WebSockets connections are inherently sticky (following the upgrade process).

Monitoring

CloudTrail can be used to capture API calls. Only pay for the S3 storage charges.

CloudTrail records information on API calls only.

To monitor other actions such as time the request was received, the client’s IP address, request paths etc. use access logs.

Access logging is optional and disabled by default.

You are only charged for the S3 storage.

ALB logs requests sent to the load balancer including requests that never made it to targets.

ALB does not log health check requests.

Logging of requests is best effort so shouldn’t be relied on for auditing.

Target groups

Target groups are a logical grouping of targets (EC2 instances or ECS).

Targets are the endpoints and can be EC2 instances, ECS containers, or IP addresses.

Target groups can exist independently from the ALB.

Target groups can have up to 1000 targets.

A single target can be in multiple target groups.

Only one protocol and one port can be defined per target group.

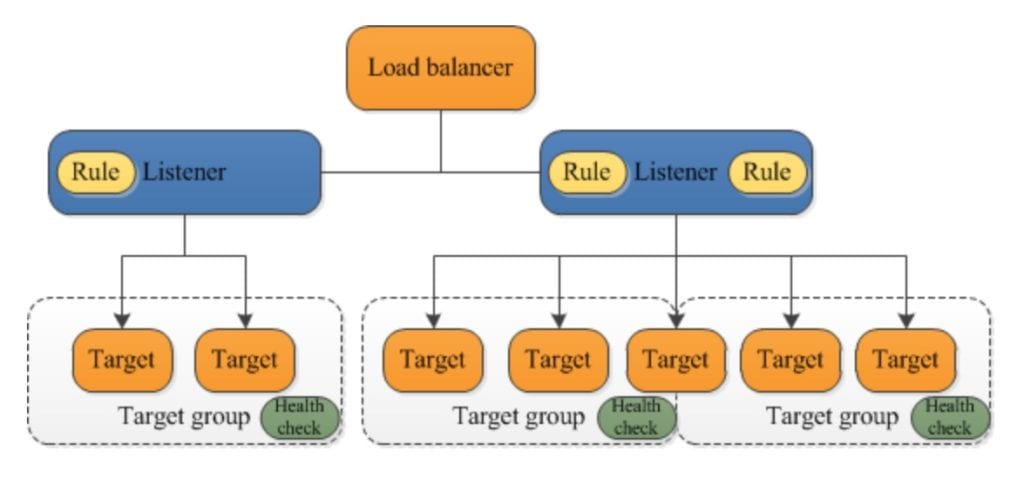
The target type in a target group can be an EC2 instance ID, IP address (must be a valid private IP from an existing subnet) or AWS Lambda Function (ALB only).

You cannot use public IP addresses as targets.

You cannot use instance IDs and IP address targets within the same target group.

A target group can only be associated with one load balancer.

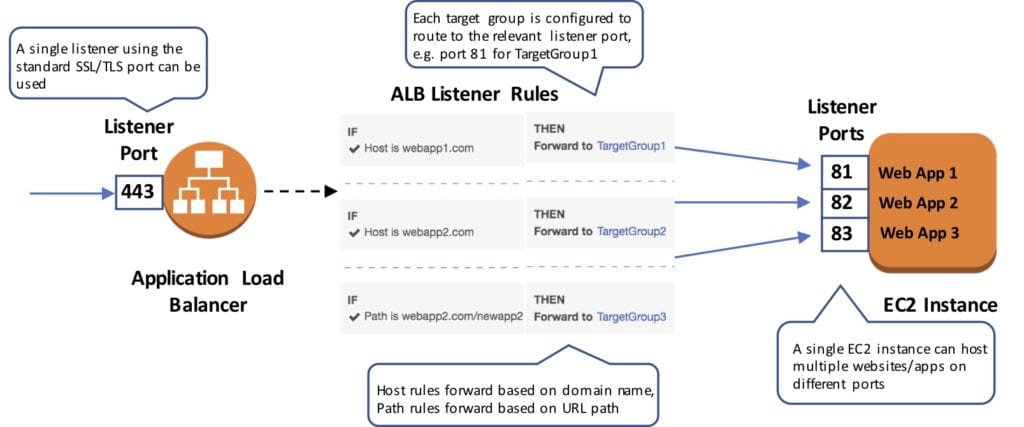
The following diagram illustrates the basic components. Notice that each listener contains a default rule, and one listener contains another rule that routes requests to a different target group. One target is registered with two target groups.



Target groups are used for registering instances against an ALB or NLB.

Target groups are a regional construct.

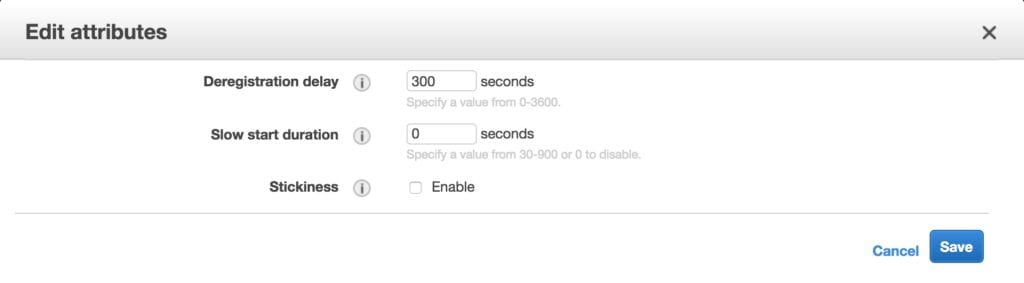
The following diagram shows how target groups can be used with host-based and target-based routing to route traffic to multiple websites, running on multiple ports, on a single EC2 instance:



The following attributes can be defined:

* Deregistration delay – the amount of time for Elastic Load Balancing to wait before deregistering a target.
* Slow start duration – the time period, in seconds, during which the load balancer sends a newly registered target a linearly increasing share of the traffic to the target group.
* Stickiness – indicates whether sticky sessions are enabled.

The default settings for attributes are shown below:



Auto Scaling groups can scale each target group individually.

You can only use Auto Scaling with the load balancer if using instance IDs in your target group.

Health checks are defined per target group.

ALB can route to multiple target groups.

You can register the same EC2 instance or IP address with the same target group multiple times using different ports (used for routing requests to micro-services).

If you register by instance ID the traffic is routed using the primary private IP address of the primary network interface.

If you register by IP address you can route traffic to an instance using any private address from one or more network interfaces.

You cannot mix different types within a target group (EC2, ECS, IP).

An EC2 instance can be registered with the same target group multiple times using multiple ports.

IP addresses can be used to register:

* Instances in a peered VPC.
* AWS resources that are addressable by IP address and port.
* On-premises resources linked to AWS through Direct Connect or a VPN connection.

Listeners and Rules

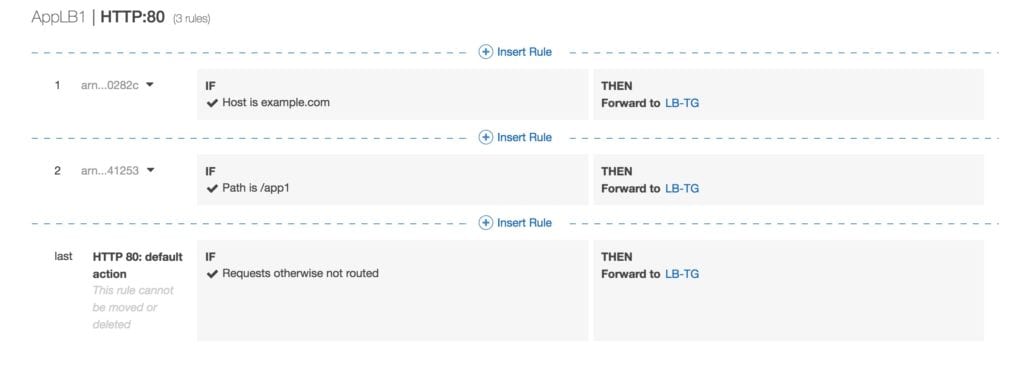
Listeners:

* Each ALB needs at least one listener and can have up to 10.
* Listeners define the port and protocol to listen on.
* Can add one or more listeners.
* Cannot have the same port in multiple listeners.

**Listener rules:**

* Rules determine how the load balancer routes requests to the targets in one or more target groups.
* Each rule consists of a priority, one or more actions, an optional host condition, and an optional path condition.
* Only one action can be configured per rule.
* One or more rules are required.
* Each listener has a default rule and you can optionally define additional rules.
* Up to 100 rules per ALB.
* Rules determine what action is taken when the rule matches the client request.
* Rules are defined on listeners.
* You can add rules that specify different target groups based on the content of the request (content-based routing).
* If no rules are found the default rule will be followed which directs traffic to the default target groups.

The image below shows a ruleset with a host-based and path-based entry and a default rule at the end:



**Default rules:**

* When you create a listener you define an action for the default rule.
* Default rules cannot have conditions.
* You can delete the non-default rules for a listener at any time.
* You cannot delete the default rule for a listener.
* When you delete a listener all of its rules are deleted.
* If no conditions for any of a listener’s rules are met, the action for the default rule is taken.

**Rule priority:**

* Each rule has a priority and they are evaluated in order of lowest to highest.
* The default rule is evaluated last.
* You can change the value of a non-default rule at any time.
* You cannot change the value of the default rule.

**Rule action:**

* Only one target group per action.
* Each rule has a type and a target group.
* The only supported action type is forward, which forwards requests to the target group.
* You can change the target group for a rule at any time.

**Rule conditions:**

* There are two types of rule condition: host and path.
* When the conditions for a rule are met the action is taken.
* Each rule can have up to 2 conditions, 1 path condition and 1 host condition.
* Optional condition is the path pattern you want the ALB to evaluate in order for it to route requests.

**Request routing:**

* 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 using the round robin routing algorithm.
* Routing is performed independently for each target group even when a target is registered with multiple target groups.
* You can configure listener rules to route requests to different target groups based on the content of the application traffic.

**Content-based routing:**

* ALB can route requests based on the content of the request in the host field: host-based or path-based.
* Host-based is domain name based routing e.g. example.com or app1.example.com.
* The host field contains the domain name and optionally the port number.
* Path-based is URL based routing e.g. example.com/images, example.com/app1.
* You can also create rules that combine host-based and path-based routing.
* Anything that doesn’t match content routing rules will be sent to a default target group.

ALB and ECS

ECS service maintains the “desired count” of instances.

Optionally a load balancer can distribute traffic across tasks.

All containers in a single task definition are placed on a single EC2 container instance.

You can put multiple containers in the same task definition behind a CLB.

* Define multiple host ports in the service definition.
* Define these listener ports as listeners on the CLB.

ECS service can only use a single load balancer.

If your task definition requires multiple ports per container you must use a CLB with multiple listeners.

ALB cannot do multiple listeners on a single task definition.

AWS does not recommend connecting multiple services to the same CLB.

ALB allows containers to use dynamic host port mapping so that multiple tasks from the same service are allowed on the same container host.

ALB supports path-based routing and priority rules.

ALB integrates with EC2 container service using service load balancing.

If a service uses multiple ports then multiple task definitions will need to be created with multiple target groups.

Federated authentication:

* ALB now supports authentication from OIDC compliant identity providers such as Google, Facebook and Amazon.
* Implemented through an authentication action on a listener rule that integrates with Amazon Cognito to create user pools.
* AWS SAM can also be used with Amazon Cognito.

Network Load Balancer

Network Load Balancer operates at the connection level (Layer 4), routing connections to targets – Amazon EC2 instances, containers and IP addresses based on IP protocol data.

It is architected to handle millions of requests/sec, sudden volatile traffic patterns and provides extremely low latencies.

Network Load Balancer supports features including:

* WebSockets.
* TLS termination.
* Preserves the source IP of the clients.
* Provides stable IP support and Zonal isolation.
* Long-running connections that are very useful for WebSocket type applications.

High throughput – designed to handle traffic as it grows and can load balance millions of requests/second.

Extremely low latencies for latency-sensitive applications.

Want to see the Network Load Balancer in action? In the AWS Hands-On Labs video tutorial, below we’ll be covering the Network Load Balancer (NLB). In around 10 minutes, we’ll review the differences between the NLB and the Application Load Balancer (CLB) and Classic Load Balancer (CLB), setup an Internet-facing NLB with Target Groups and EC2 web server instances and demonstrate the load balancing behaviour of the NLB.

Uses static IP addresses – each NLB provides a single IP address for each AZ.

Can also assign an Elastic IP to the load balancer per AZ.

The IP-per-AZ feature reduces latency with improved performance, improves availability through isolation and fault tolerance and makes the use of NLBs transparent to your client applications.

Preserves the source IP of clients, and provides stable IP support and Zonal isolation.

Can load balance any application hosted in AWS or on-premises using IP addresses of the application back-ends as targets.

NLB supports connections from clients to IP-based targets in peered VPCs across different AWS Regions.

Supports both network and application target health checks.

Supports long-running/lived connections (ideal for WebSocket applications).

Supports failover between IP addresses within and across regions (uses Route 53 health checks).

Integration with Route 53 enables the removal of a failed load balancer IP address from service and subsequent redirection of traffic to an alternate Network Load Balancer in another region.

Supports cross-zone load balancing (not enabled by default when created through the console, unlike ALB and CLB).

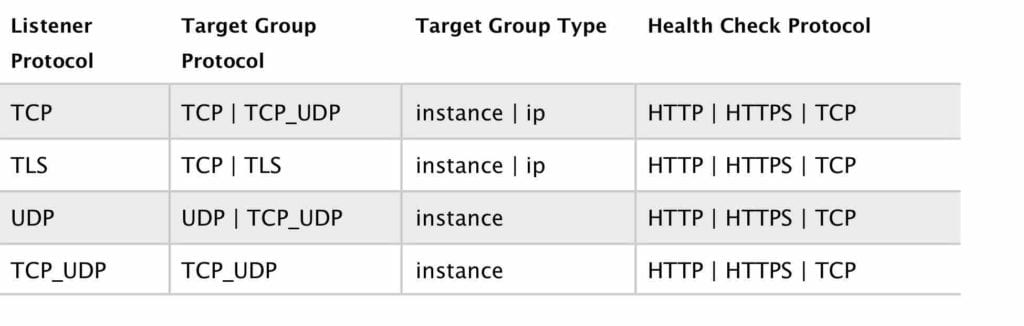
Uses the same API as the Application Load Balancer.

Also uses Target Groups (see section above).

Target groups for Network Load Balancers support the following protocols and ports:

* **Protocols:** TCP, TLS, UDP, TCP\_UDP.
* **Ports:** 1-65535.

The following table summarizes the supported combinations of listener protocol and target group settings:



CloudWatch reports Network Load Balancer metrics.

Enhanced logging – can use the Flow Logs feature to record all requests sent to your load balancer.