

NET402

AWS re:INVENT

Elastic Load Balancing Deep Dive and Best Practices

David Pessis

November 30, 2017

AWS
re:Invent

© 2017, Amazon Web Services, Inc. or its Affiliates. All rights reserved.



Elastic Load Balancing automatically distributes incoming application traffic across multiple Amazon EC2 instances for fault tolerance and load distribution. In this session, we go into detail about Elastic Load Balancing configuration and day-to-day management, and also its use with Auto Scaling. We explain how to make decisions about the service and share best practices and useful tips for success.

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

You can now add IP addresses addressable from your VPC as targets behind your LB,

Layer 4 (network)

Supports TCP and SSL

Incoming client connection bound to server connection

No header modification

Proxy Protocol prepends source and destination IP and ports to request

Layer 7 (application)

Supports HTTP and HTTPS

Connection terminated at the load balancer and pooled to the server

Headers may be modified

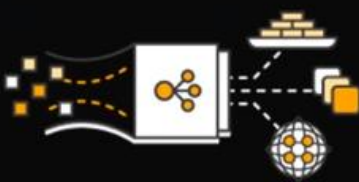
X-Forwarded-For header contains client IP address

There are 2 main types of load balancing, **Layer 4** and **Layer 7**, where headers are modified by AWS

The Elastic Load Balancing family

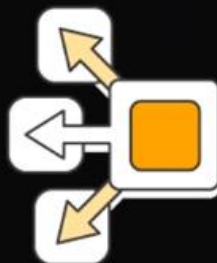
Application Load Balancer

HTTP & HTTPS (VPC)



Network Load Balancer

TCP Workloads (VPC)



Classic Load Balancer

Previous Generation
for HTTP, HTTPS, TCP
(Classic Network)

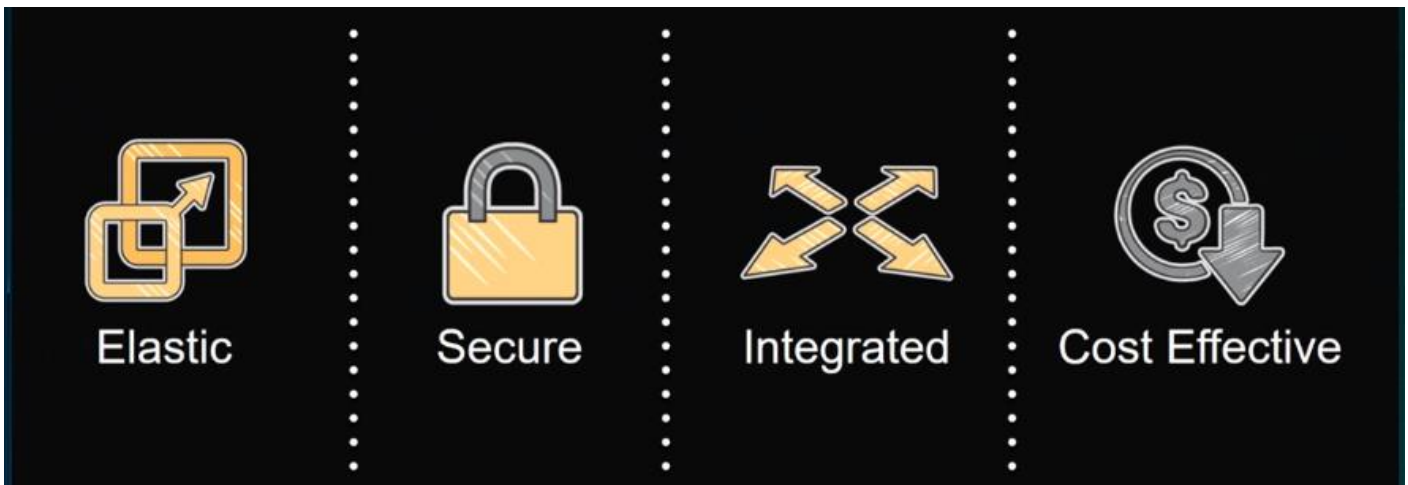


AWS
re:Invent

© 2017, Amazon Web Services, Inc. or its Affiliates. All rights reserved.

aws

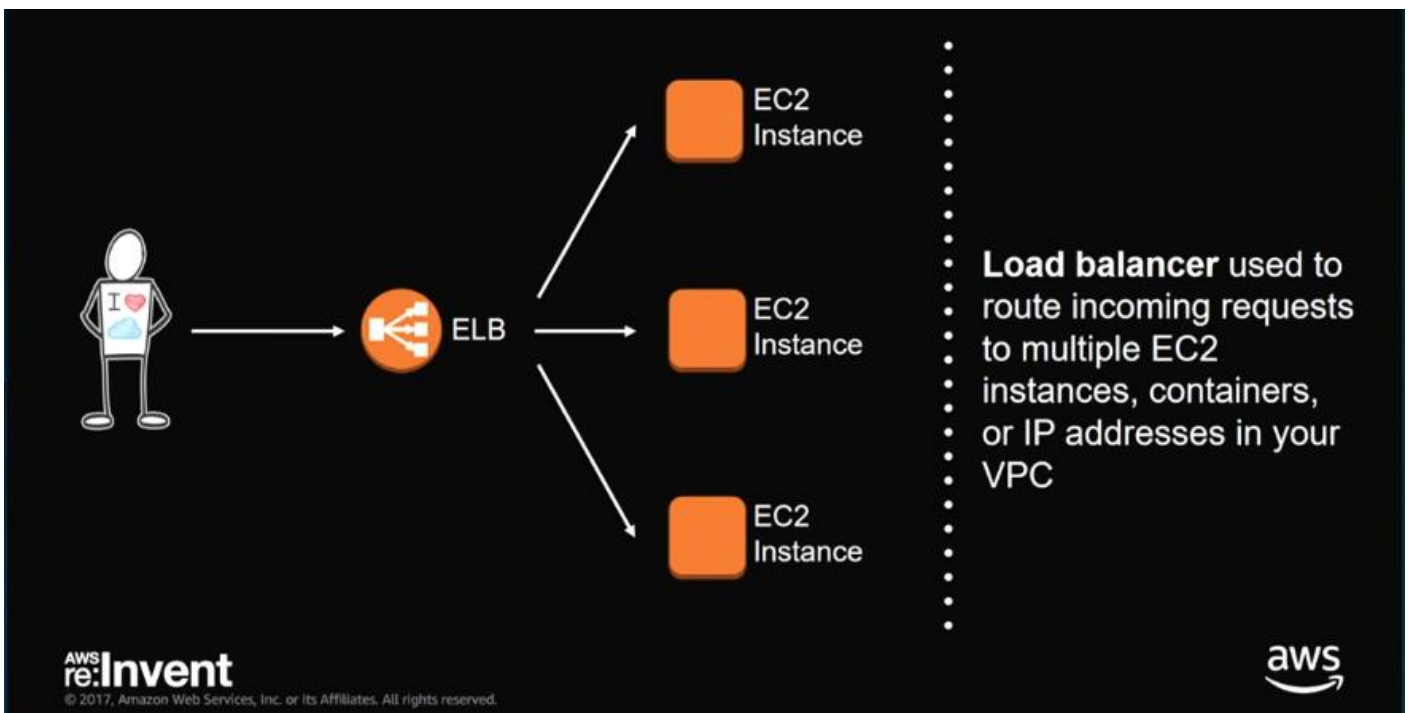
The **Application Load Balancer** ALB is a very feature rich type of **Layer 7 LB** and today's talk will focus on the ALB. The **Network Load Balancer** NLB is the Layer 4 LB that is very high performance, low latency for your TCP traffic, the **Classic Load Balancer** is very HA and reliable LB.



Elastic means that the LBs are dynamically scaled for you to handle increasing traffic, Security is maintained using SNI, managed security policies, and other advanced security features. ELB is Integrated with several AWS services.

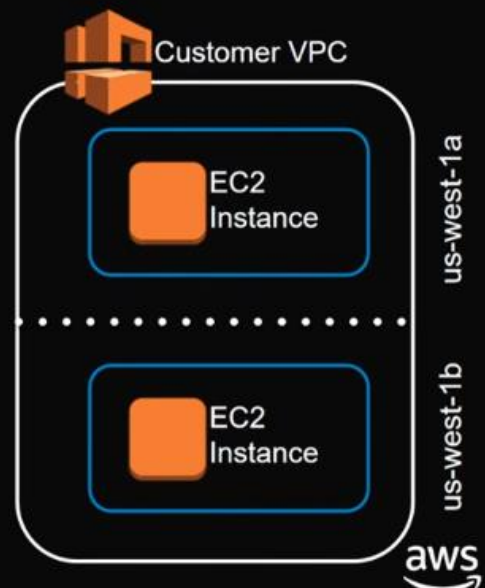


Running your applications and databases on a single EC2 instance is not safe or recommended



This is what we want your architecture to look like, you have an ELB and you put multiple application servers behind that ELB. If one target fails, the LB automatically fails away from using that EC2 instance and stops routing traffic to that instance.

Architecture

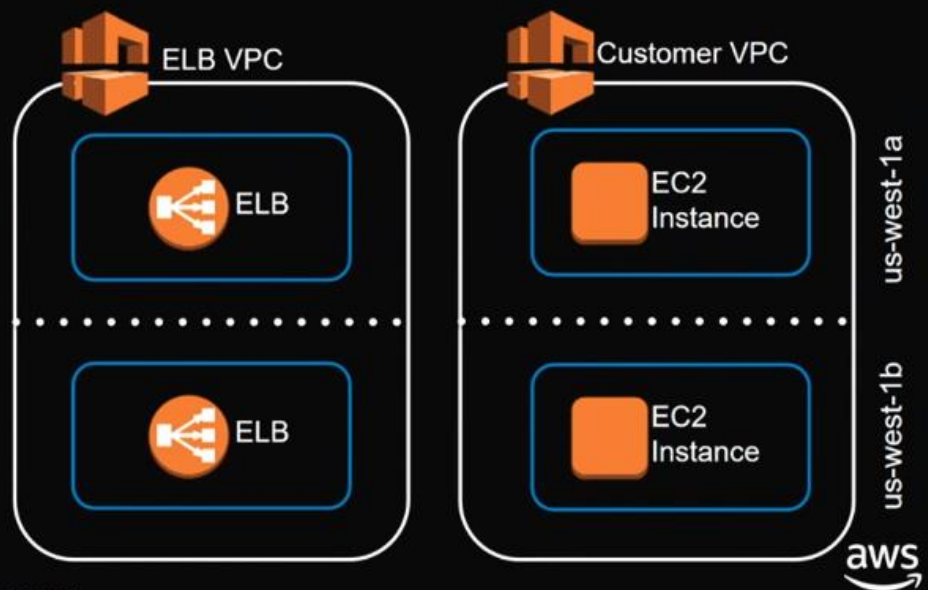


AWS re:Invent

© 2017, Amazon Web Services, Inc. or its Affiliates. All rights reserved.

This is the approach we recommend users take. When you create a VPC, it is recommended that you give 2 different subnets in 2 different AZs as above. You can then create your EC2 instances in the 2 different subnets to use as targets for the ELB, this helps to create HA.

Architecture

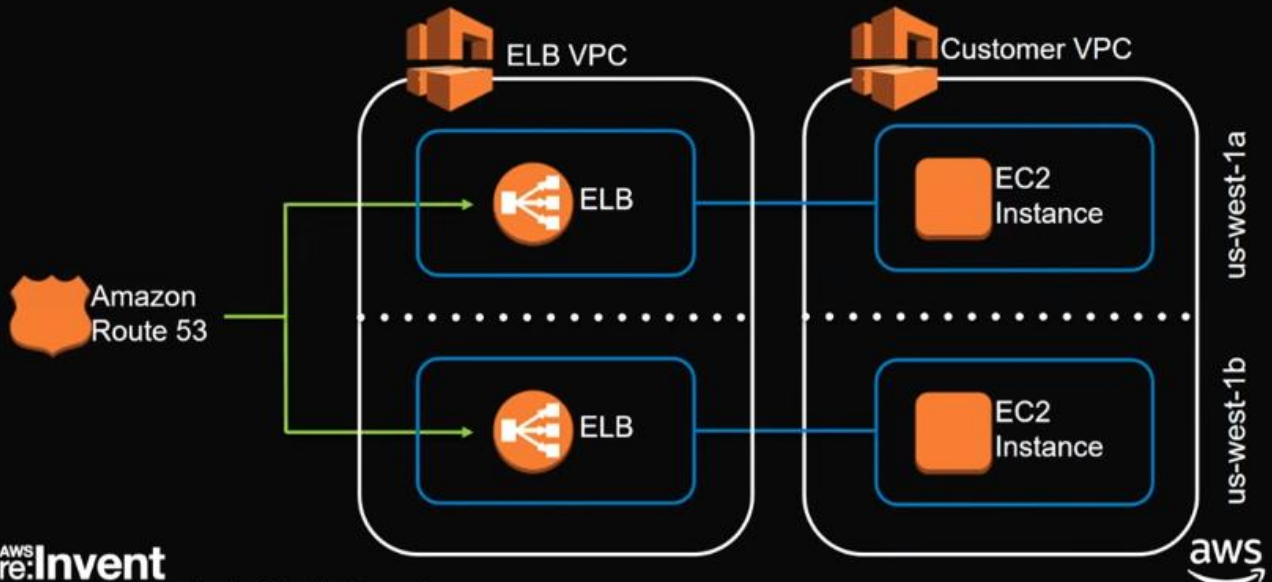


AWS re:Invent

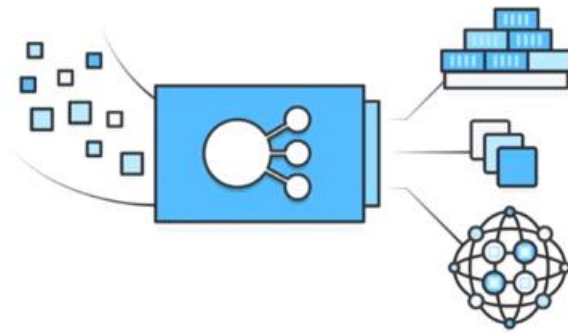
© 2017, Amazon Web Services, Inc. or its Affiliates. All rights reserved.

Then you create an ELB VPC as above, create ELB nodes in the AZs that correspond to where your targets are running as above. This provides HA as well.

Architecture



Finally, use Route53. This is because when you create a LB, AWS will give you an A-Record for that LB that you can then create a CNAME and DNS record like david.com to use for routing to the LB. Route53 will actually round robin across the 2 ELB nodes, this provides you redundancy and extra HA with your nodes and instances running in the 2 different VPCs and AZs.



Application Load Balancer

Advanced request routing with support for microservices and container-based applications

aws
re:Invent

© 2017, Amazon Web Services, Inc. or its Affiliates. All rights reserved.

aws

Application Load Balancer



New, feature-rich, Layer 7 load-balanced platform

Content-based routing allows requests to be routed to different applications behind a single load balancer

Support for **microservices and container-based applications**, including deep integration with **Elastic Container Service**

AWS re:Invent

© 2017, Amazon Web Services, Inc. or its Affiliates. All rights reserved.



Application Load Balancer

Support for **WebSockets and HTTP/2**

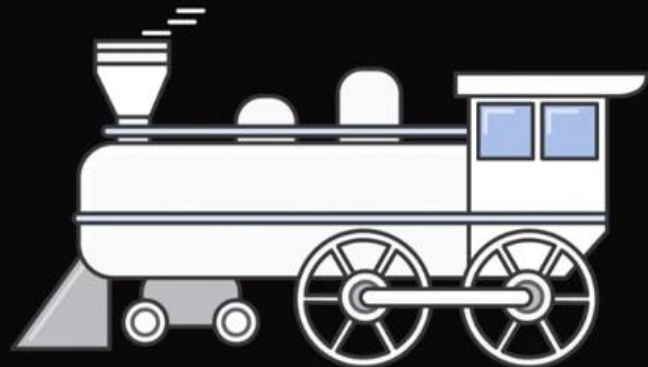
Path and host-based routing

Improved **health checks** and additional **Amazon CloudWatch metrics**

Improved performance for real-time and streaming applications

Improved Elastic Load Balancing API

Load balancer API deletion protection



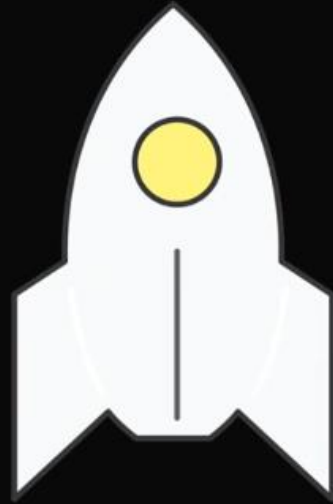
AWS re:Invent

© 2017, Amazon Web Services, Inc. or its Affiliates. All rights reserved.



Feature launches in the last year...

- Host-based routing
- Server Name Identification (SNI)
- CloudWatch percentiles support
- Request tracing
- Native IPv6
- AWS WAF support
- New predefined security policies
- IP as a target

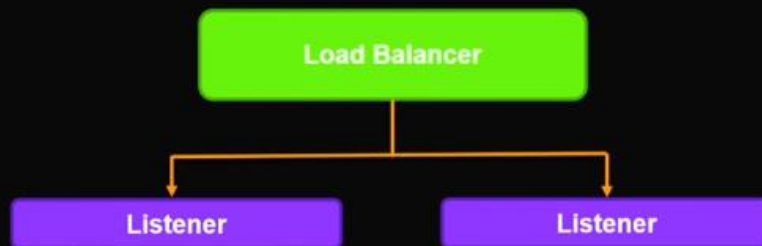


AWS
re:Invent

© 2017, Amazon Web Services, Inc. or its Affiliates. All rights reserved.



SNI allows you to put multiple certificates on your LB,



AWS
re:Invent

© 2017, Amazon Web Services, Inc. or its Affiliates. All rights reserved.



We are now going to **build an ALB from the ground up**. You start with a LB and **every LB needs a Listener**

Listeners



Define the **port and protocol** which the load balancer must listen on

Each Application Load Balancer needs **at least one listener** to accept traffic

Each Application Load Balancer can have **up to 10 listeners**

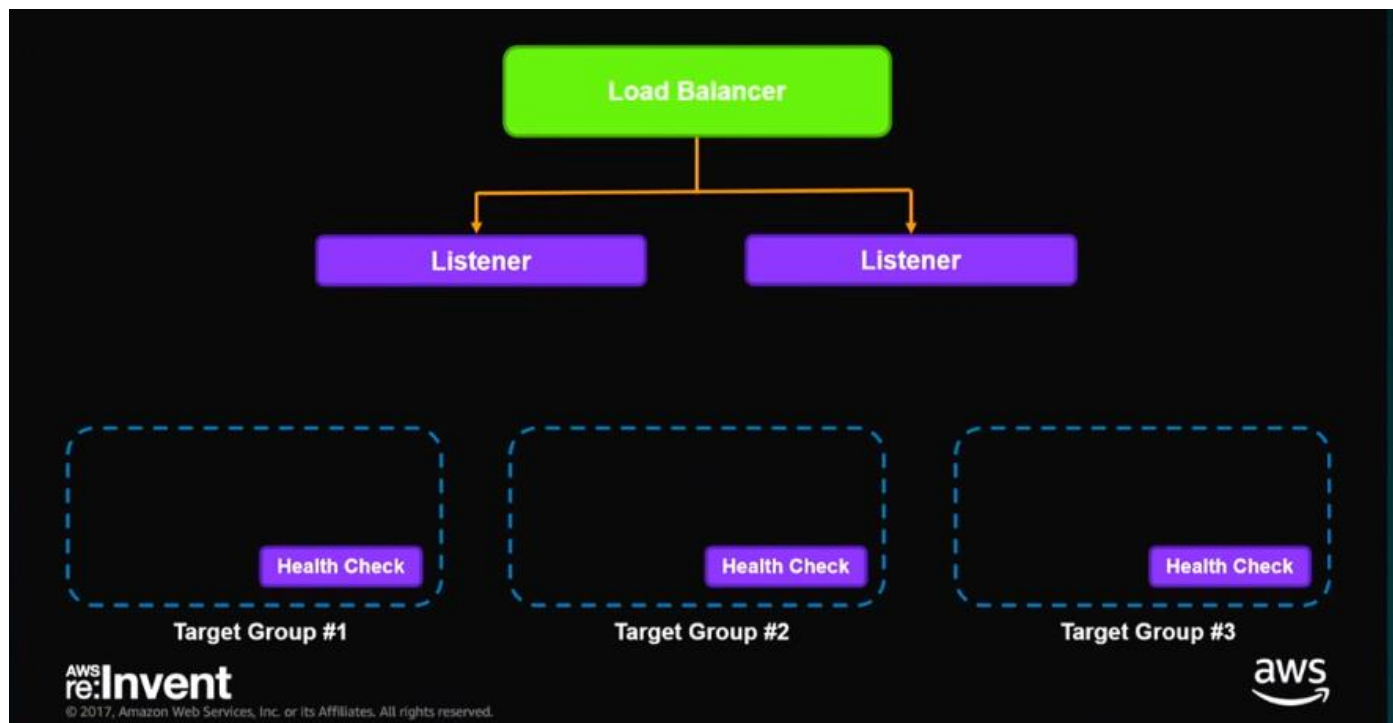
Routing rules are defined on listeners

AWS
re:Invent

© 2017, Amazon Web Services, Inc. or its Affiliates. All rights reserved.



For an **ALB**, the most **common listeners** are for **HTTP with port 80** or **HTTPS with port 443**. **Routing rules defined on listeners** allow you to do **content-based routing** or **host-based routing**, these are defined at the listener level.



AWS
re:Invent

© 2017, Amazon Web Services, Inc. or its Affiliates. All rights reserved.



We then need to add **Target Groups**, target groups are logical collections of AWS resources, they can be containers, IP addresses, or EC2 instances. Every target group also has its own customized **Health Check** that you can configure.

Target groups

Logical grouping of targets behind the load balancer

Target groups can exist independently from the load balancer

Regional construct that can be associated with an Auto Scaling group

Target groups can contain up to 1,000 targets



AWS
re:Invent

© 2017, Amazon Web Services, Inc. or its Affiliates. All rights reserved.



You can define and start creating a target group with its health check, and when it is ready you can simply attach it to a LB. each target group can be associated with an ASG. Since each target group can be different application, each target group can be scaled independently of the other target groups to get very granular and efficient resource scaling

Health checks allow for
traffic to be shifted away
from **failed instances**

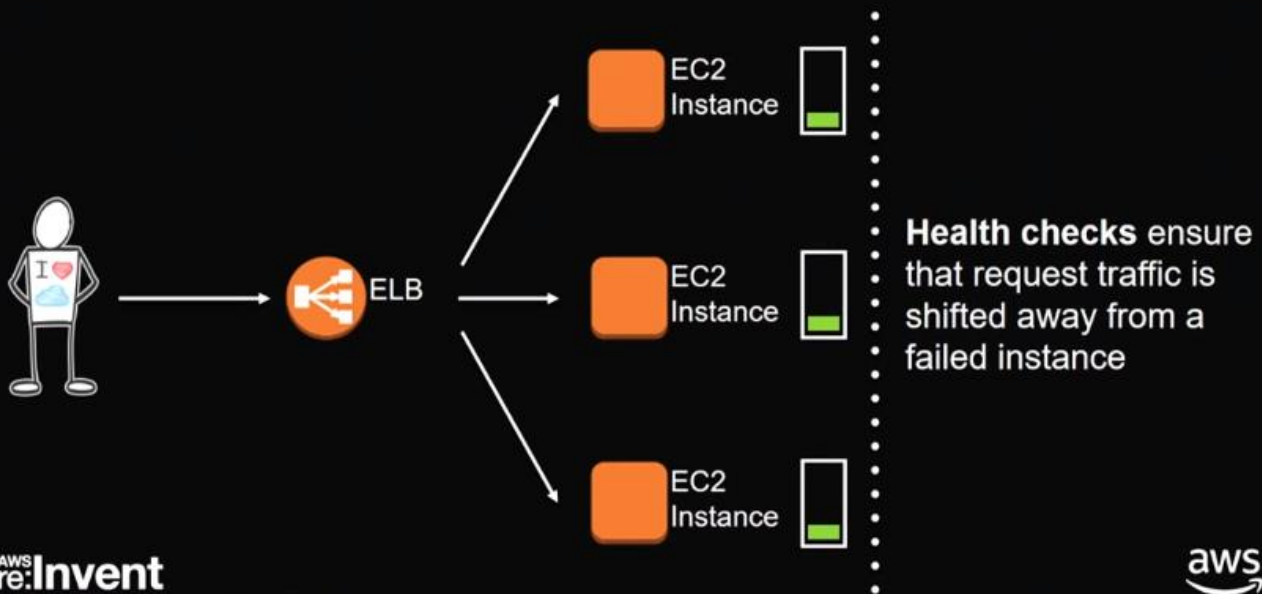


AWS
re:Invent

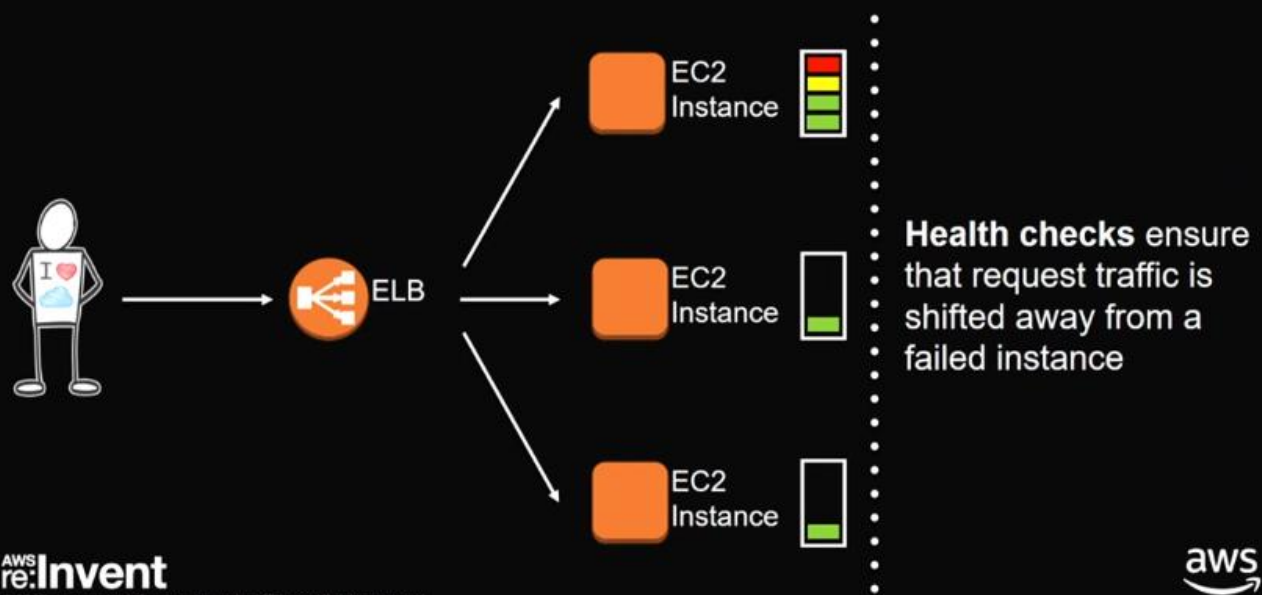
© 2017, Amazon Web Services, Inc. or its Affiliates. All rights reserved.



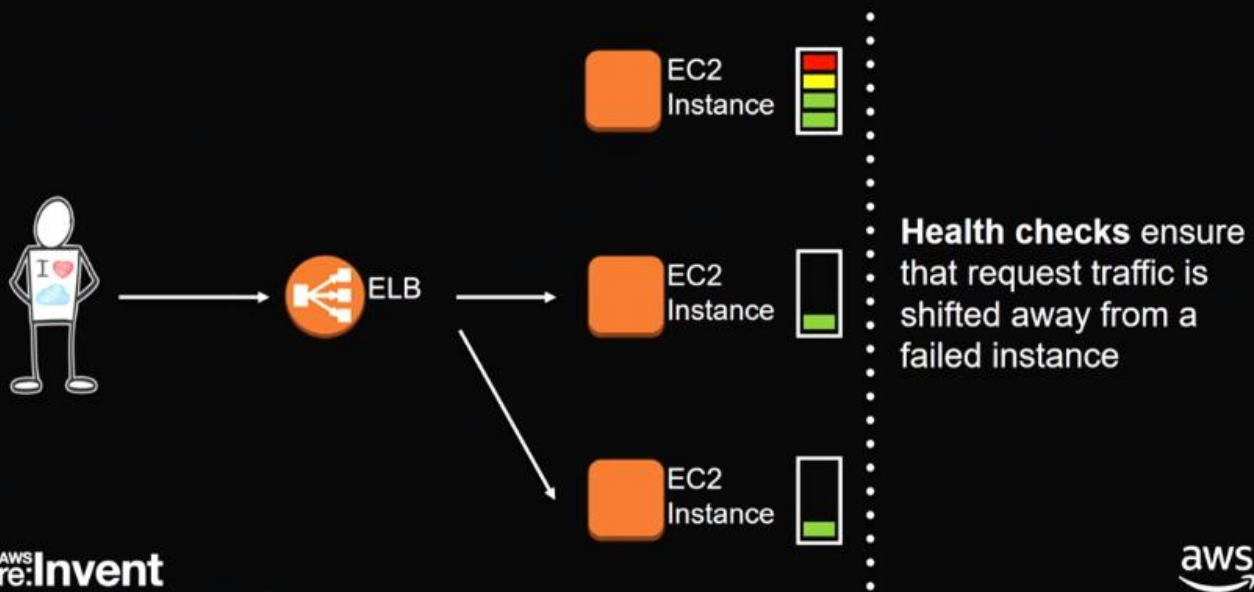
Health checks



Health checks

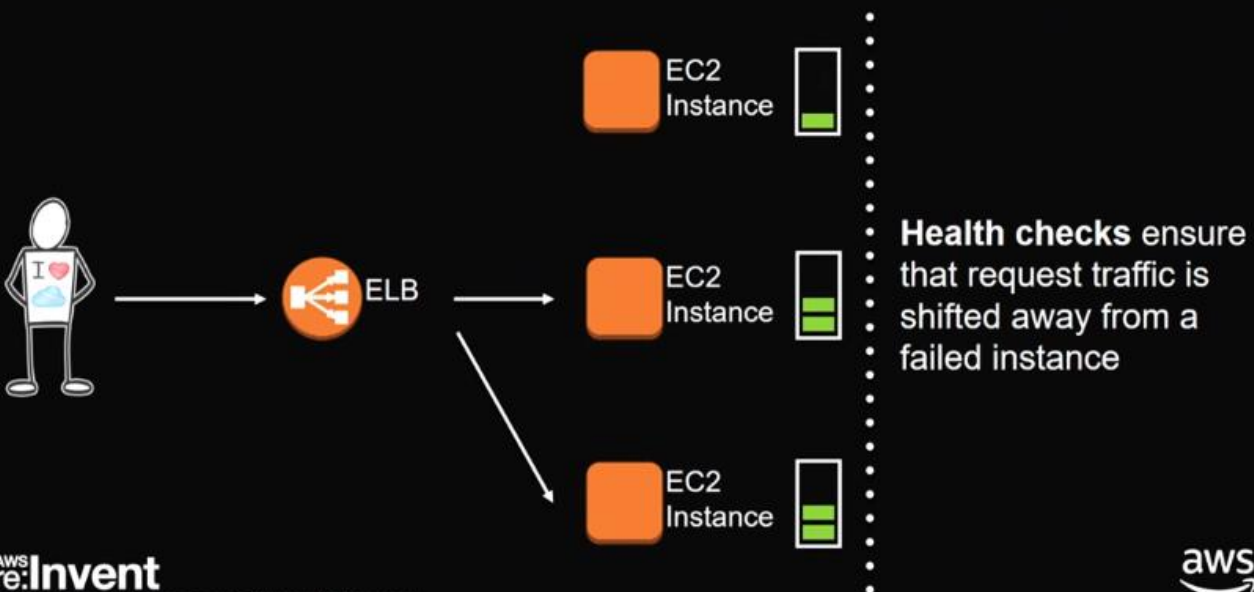


Health checks

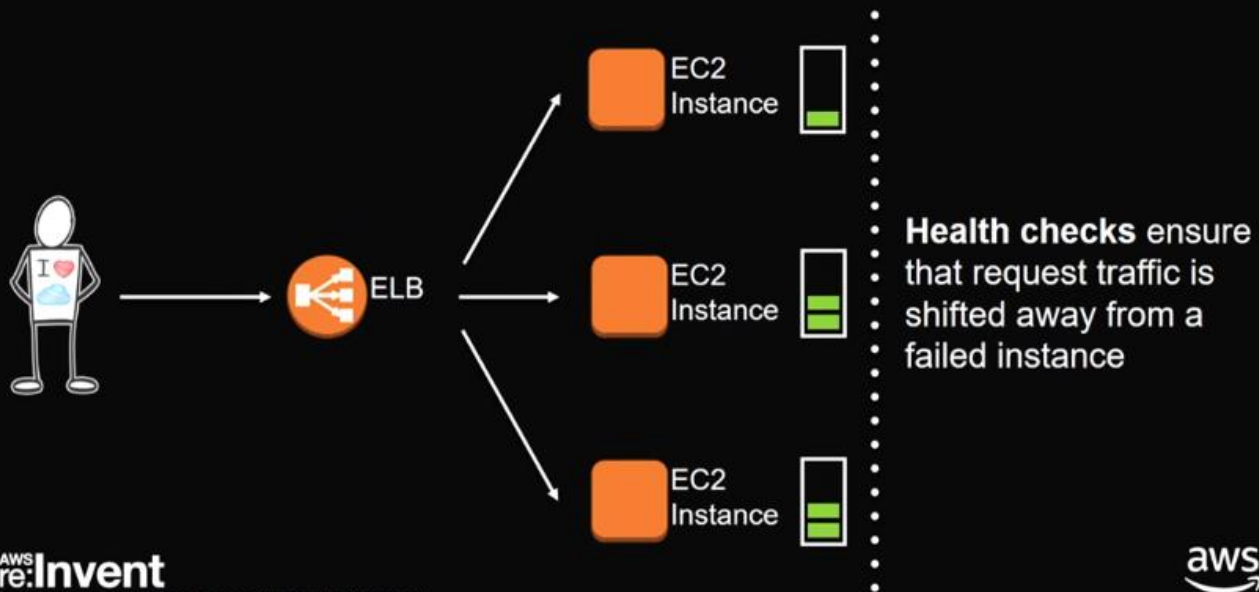


The ELB will automatically failover to the healthy EC2 instances within the target group, you will then get a notification that one of your hosts is not healthy. You can then fix it

Health checks



Health checks



AWS re:Invent

© 2017, Amazon Web Services, Inc. or its Affiliates. All rights reserved.

aws

Health checks

Support for **HTTP and HTTPS** health checks

Customize the frequency and failure thresholds

Consider the **depth and accuracy** of your health checks

Customize list of **successful response codes**, for example, 200–300

Details of **health check failures** are returned via the API and the AWS Management Console

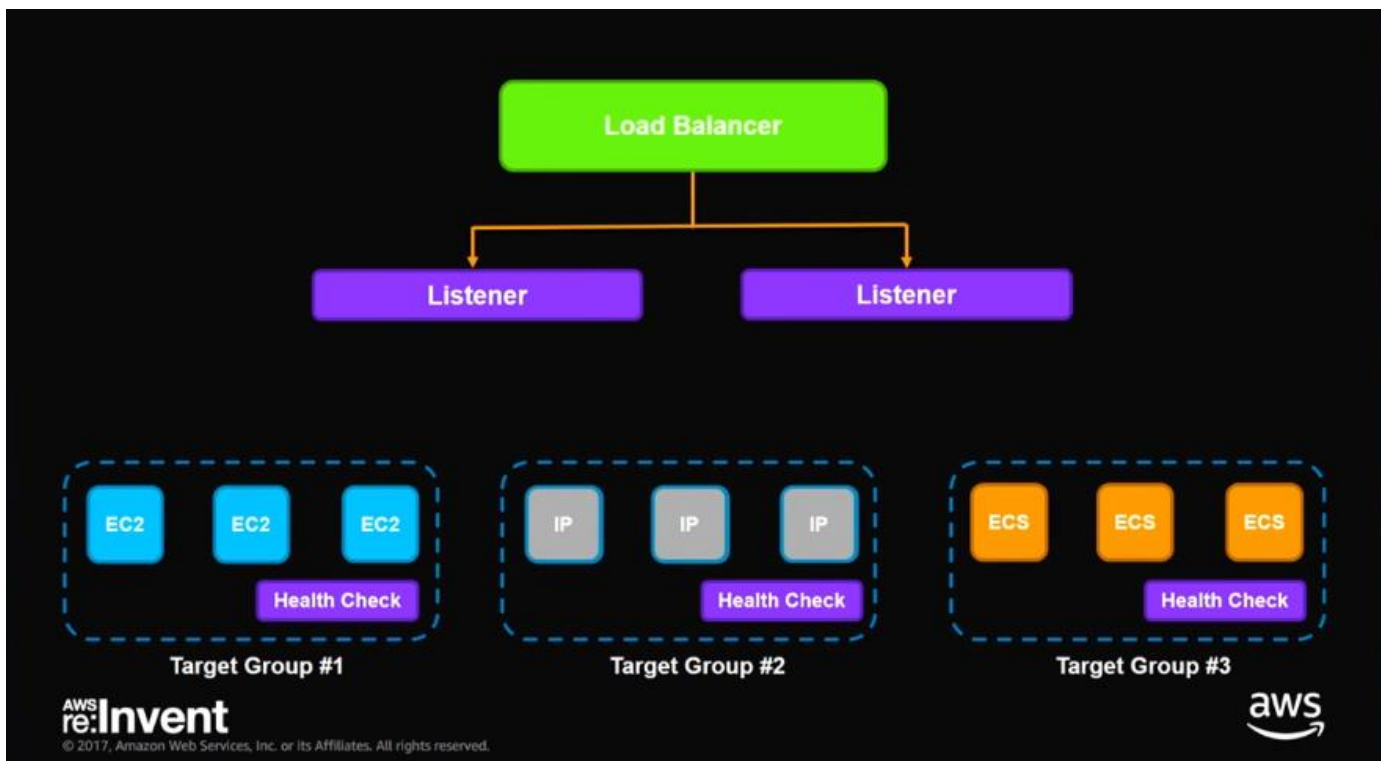


AWS re:Invent

© 2017, Amazon Web Services, Inc. or its Affiliates. All rights reserved.


aws

Don't make your health checks too deep, make them monitor just the health of your application.




We can then add resources to our 3 target groups like above.

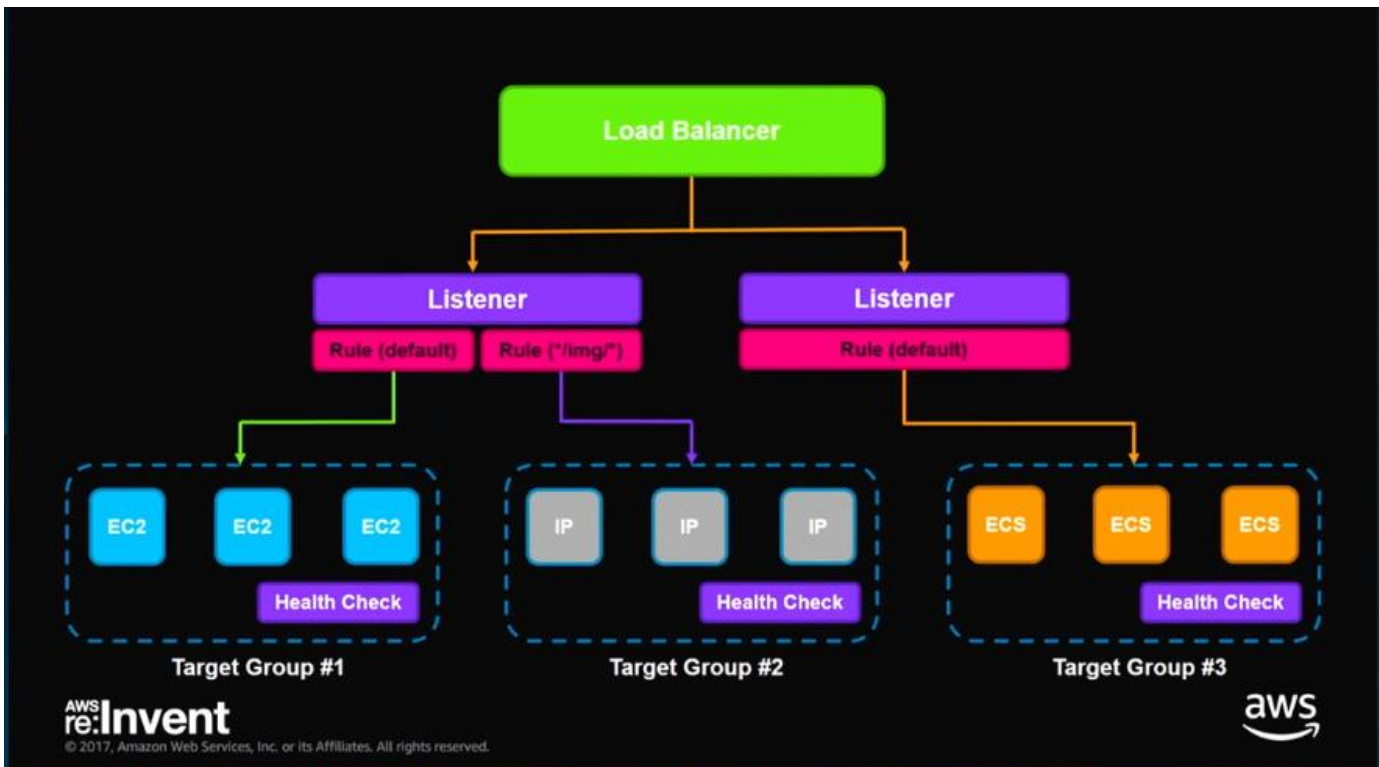
Targets



- Support for **EC2 instances** and Amazon ECS containers, and **IP Addresses**
- EC2 instances can be **registered** with the same target group using **multiple ports**
- A single target can be registered with **multiple target groups**
- IP addresses** both accessible within your VPC or via DX and VPN

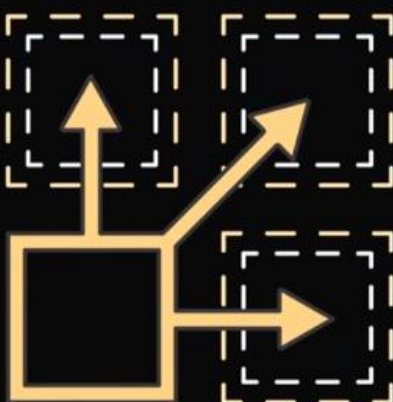
AWS re:Invent © 2017, Amazon Web Services, Inc. or its Affiliates. All rights reserved. 

You can register EC2 instances with the same target group using multiple ports, this allows you to put multiple containers on the same EC2 instance and then register that EC2 instance 20 times with a different port for each container. A single EC2 instance can also be registered with multiple target groups. You can also route to IP addresses that are accessible from your VPC or IP addresses that are on-prem using VPC Connect



We now add some rules to the listeners to define what routes traffic they should allow and route to the different target groups attached to them. These are how you can get really granular about what traffic you want to direct to applications behind you LB.

Rules



- Each **listener** can have one or more rules for routing requests to target groups
- Rules consist of **conditions and actions**
- When a request meets the condition of the rule, the action is taken
- Today, rules can forward requests to a specified target group

The diagram shows a solid yellow square box on the left. Three yellow arrows originate from the top-right corner of this box and point towards three separate dashed yellow rectangular boxes arranged in a triangular pattern to the right, illustrating the concept of routing traffic based on rules.

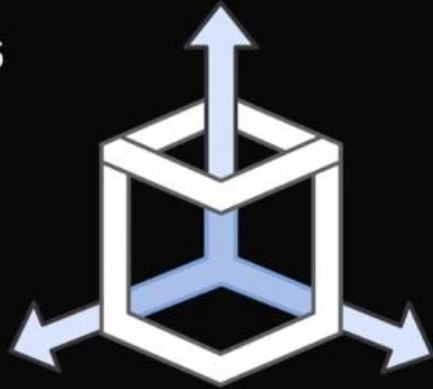
AWS re:Invent © 2017, Amazon Web Services, Inc. or Its Affiliates. All rights reserved. aws

Rules (continued)

Conditions can be specified in path pattern format

A path pattern is case-sensitive, can be up to 255 characters in length, and can contain any of the following characters:

- A-Z, a-z, 0-9
- _ - . \$ / ~ " ' @ : +
- & (using &#38;)
- * (matches 0 or more characters)
- ? (matches exactly 1 character)



AWS
re:Invent

© 2017, Amazon Web Services, Inc. or its Affiliates. All rights reserved.



Host-based routing

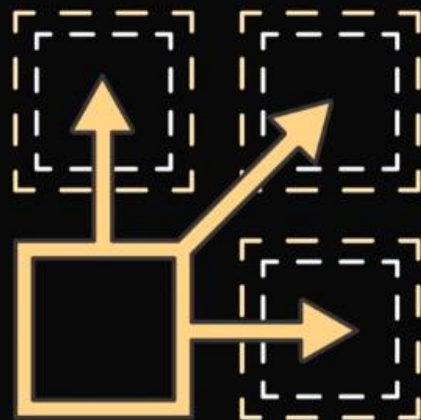
Route based on host field in the HTTP header

Support multiple domains using a single load balancer

Route each host name to a different target group

Combine host-based routing and path-based routing

- 128-character limit
- A-Z, a-z, 0-9, -, .
- * (matches 0 or more characters)
- ? (matches exactly 1 character)

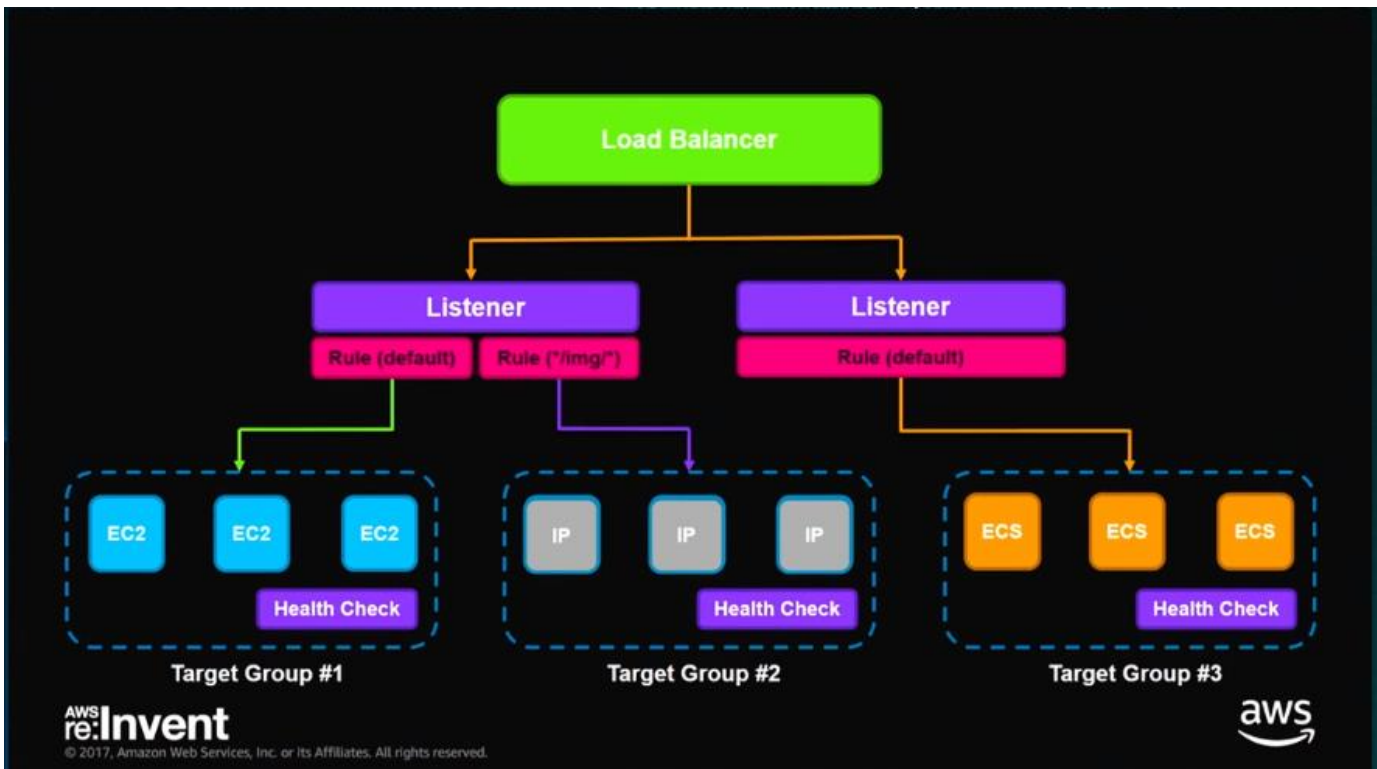


AWS
re:Invent

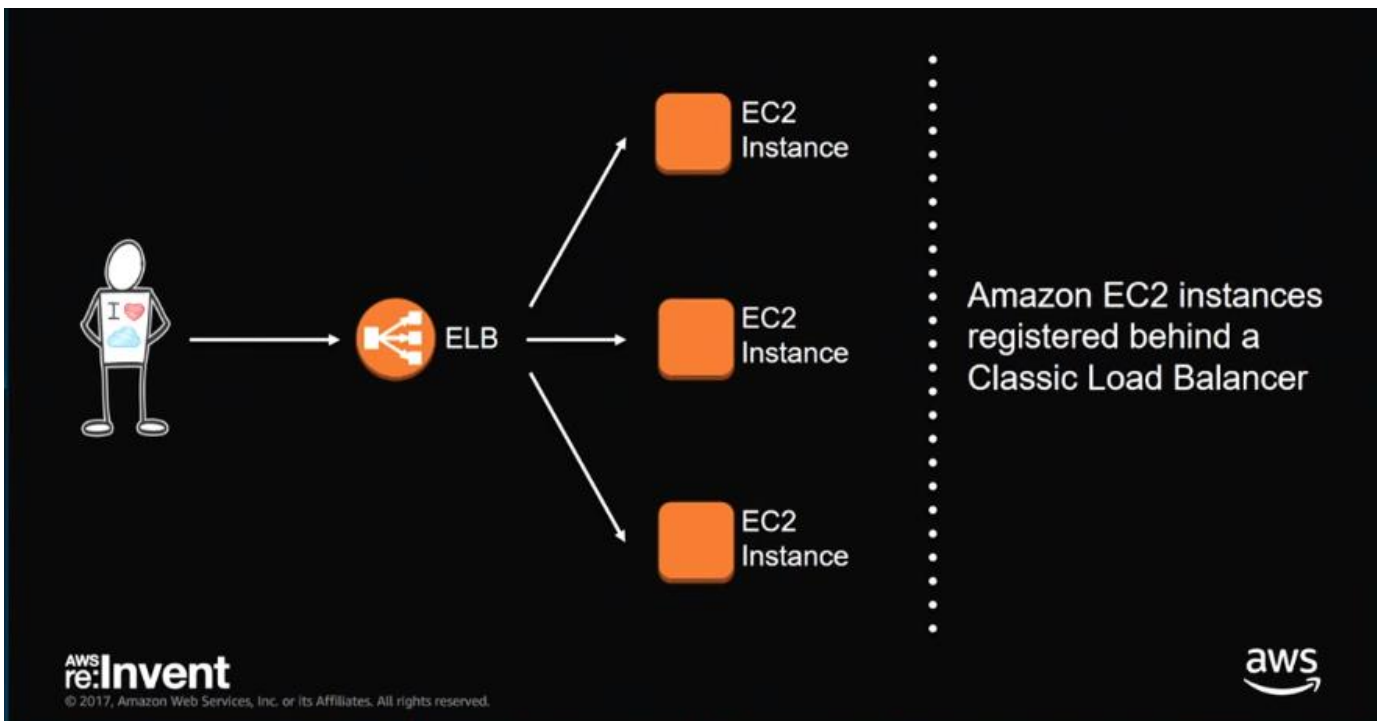
© 2017, Amazon Web Services, Inc. or its Affiliates. All rights reserved.



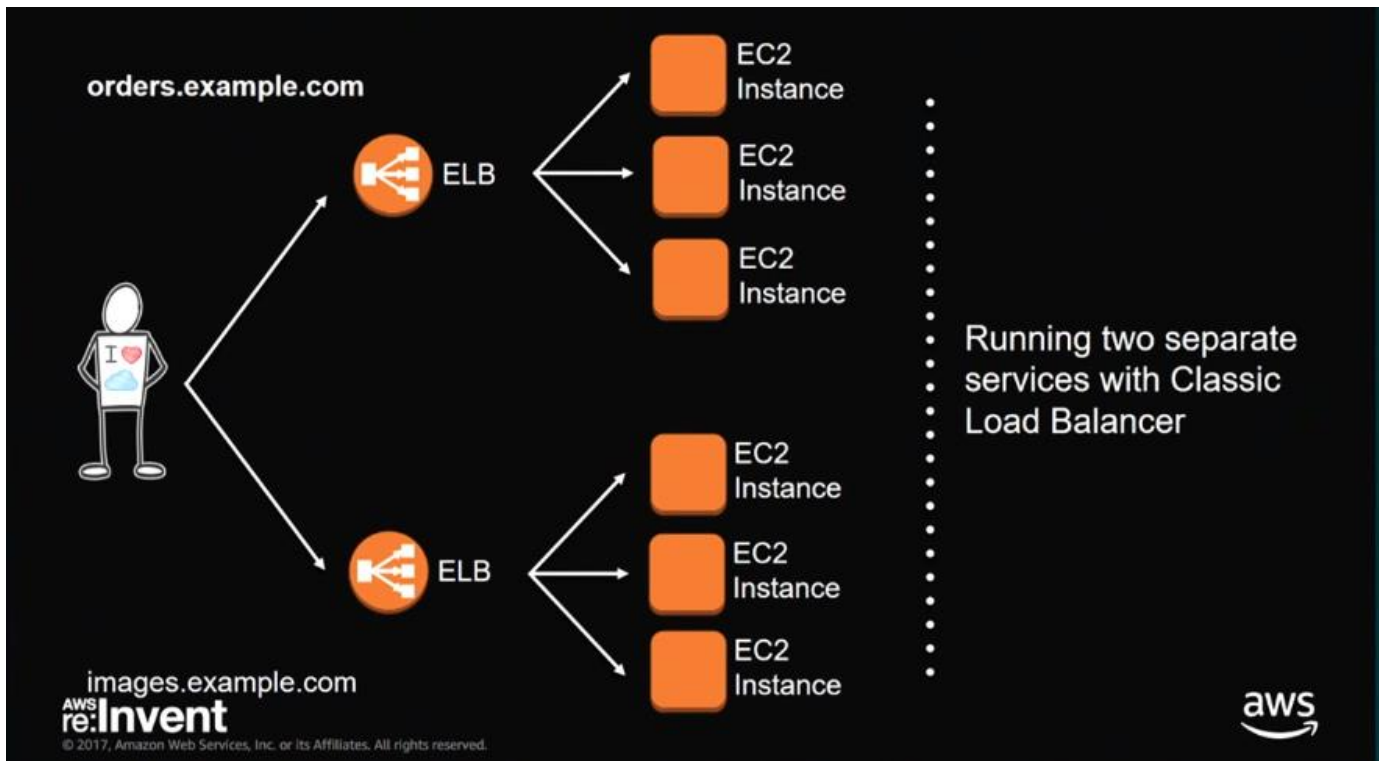
We can also do **host-based routing**, this allows **david.com** to go to a **specific target group** and **andrew.com** can go to **another target group**.



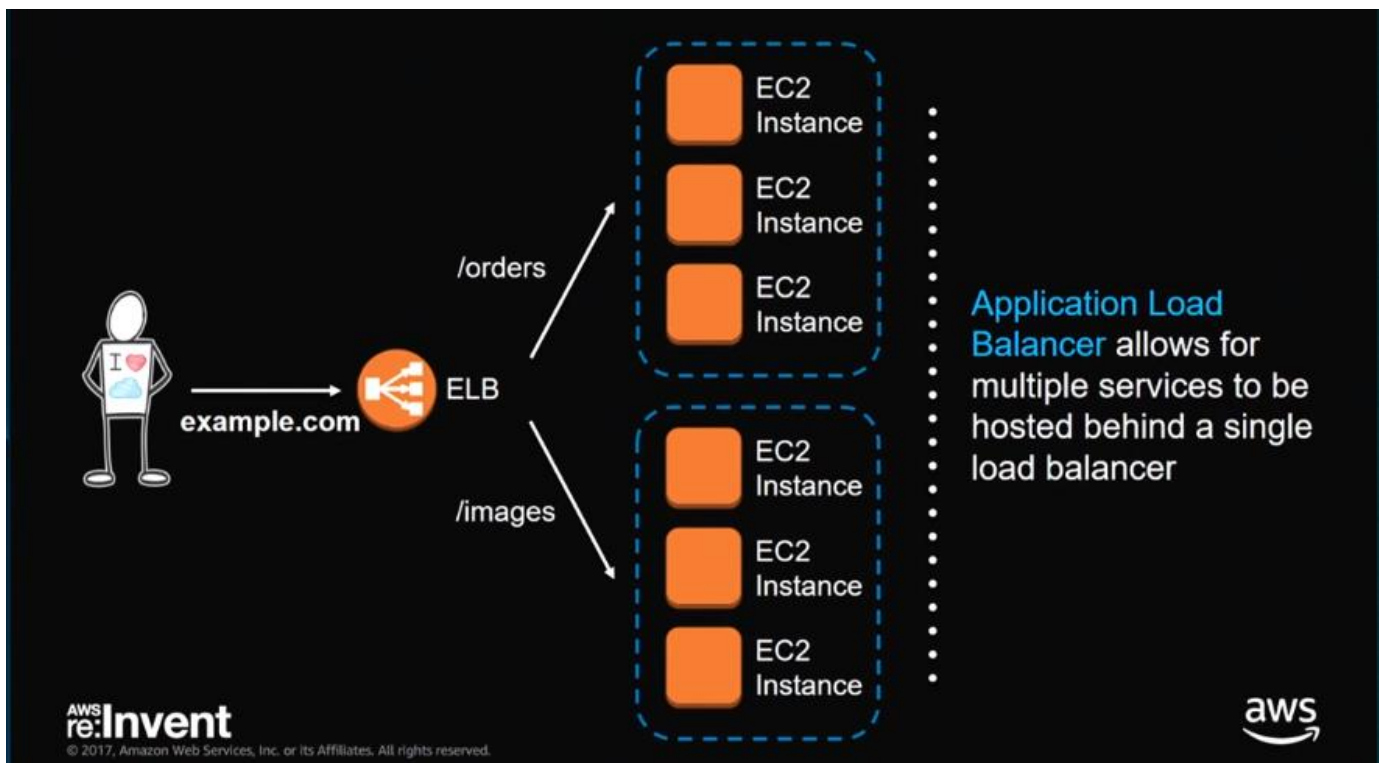
We now have the rules, targets, and the health checks configured for our LB



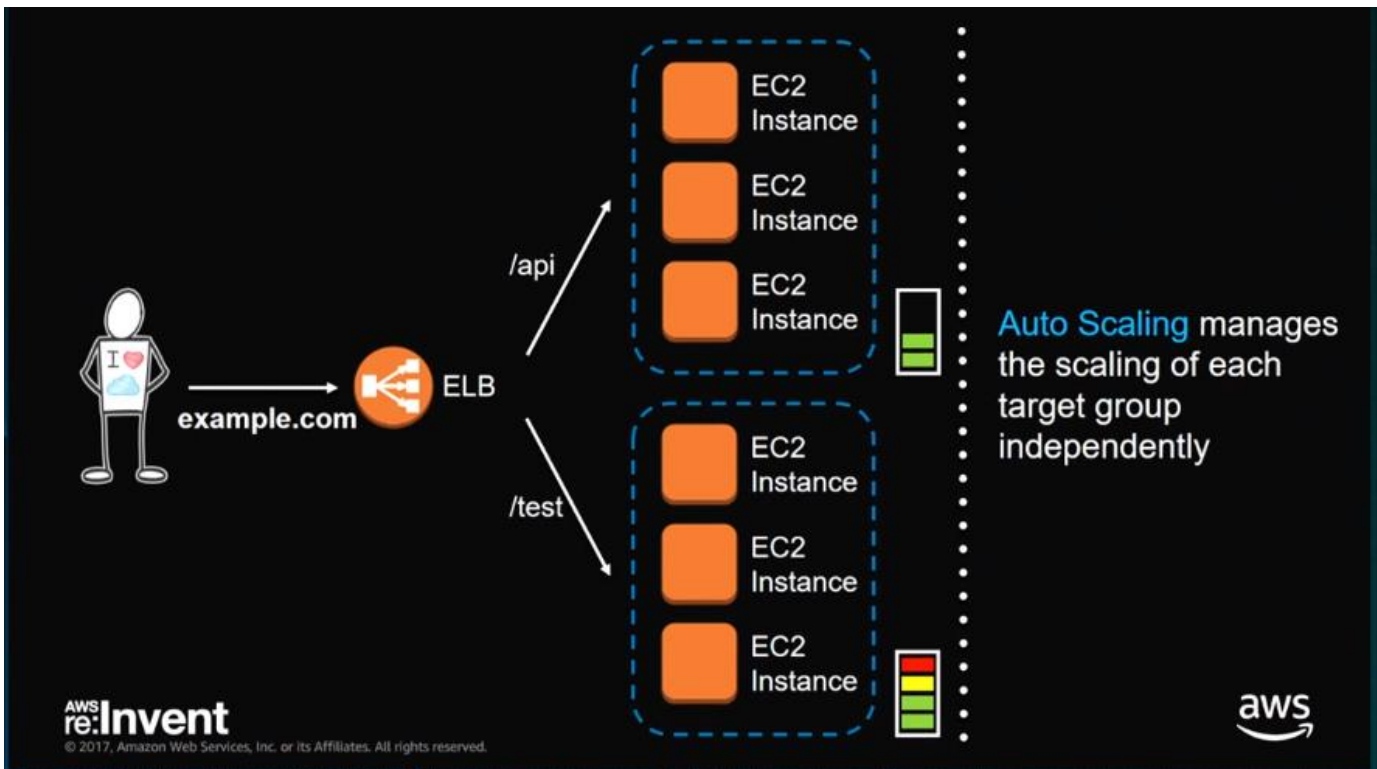
Here is the classic LB architecture where we have applications behind our LB for orders.example.com, if we want to add another application behind the LB for images.example.com



The way we used to have to do that is to create another classic LB because orders.example.com and images.example.com could not be routed through the same LB



Using the new ALB, you can now consolidate all your applications behind a single LB as above. Note that this increases your blast radius too and you need to be careful how much to consolidate



ECS integration



Application Load Balancer (ALB) is fully integrated with Amazon EC2 Container Service (Amazon ECS), managing target groups, paths, and targets

ECS will automatically register tasks with the load balancer using a dynamic port mapping

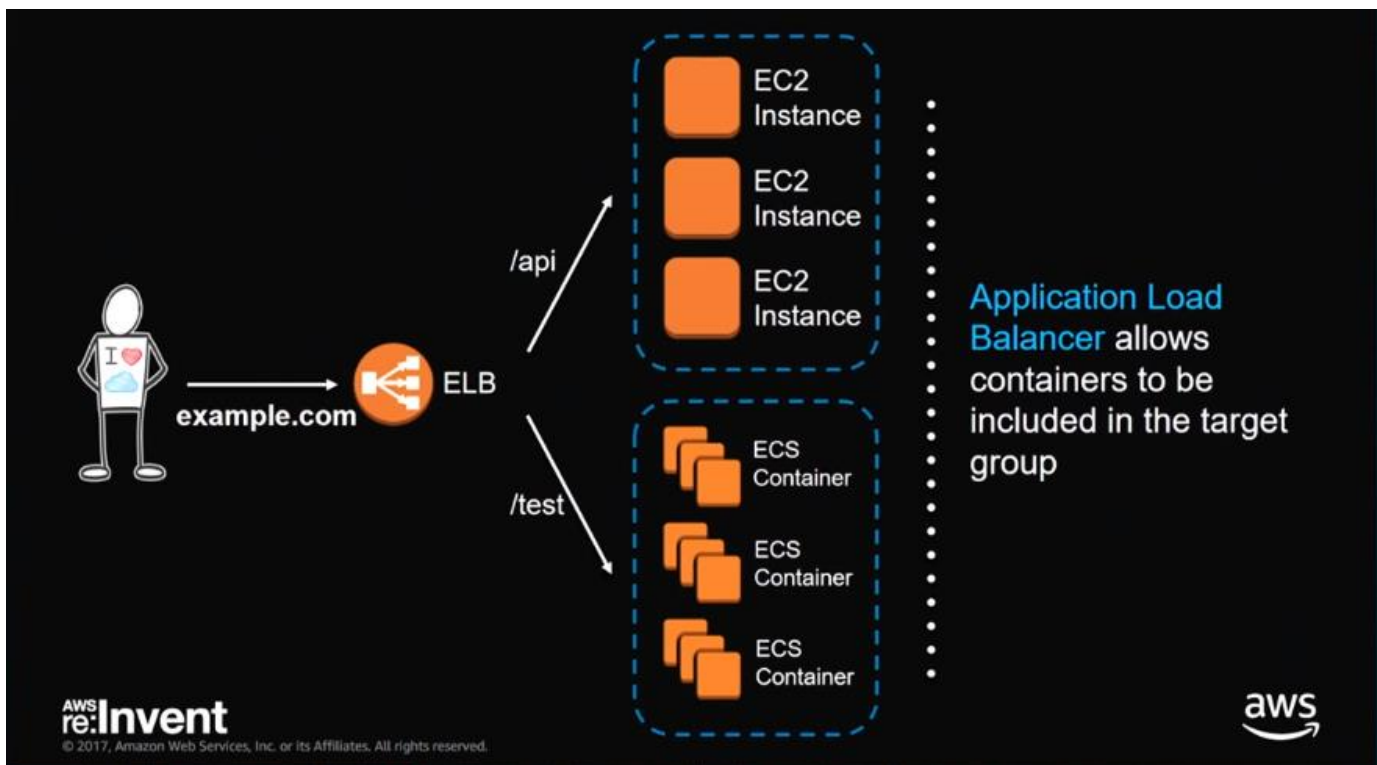
Can also be used with other container technologies

aws re:Invent

© 2017, Amazon Web Services, Inc. or its Affiliates. All rights reserved.

aws

Using ECS, EKS and Fargate now makes it far easier for you to deploy containers behind your LBs.



Predefined security policies

ELBSecurityPolicy-TLS-1-1-2017-01—Supports TLS 1.1 and above

ELBSecurityPolicy-TLS-1-2-2017-01—Strictly supports TLS1.2

ELBSecurityPolicy-2016-08—New default policy, same as Classic Load Balancer default policy

Windows XP Security Policy

Windows XP supported policy – Coming soon



aws re:Invent

© 2017, Amazon Web Services, Inc. or its Affiliates. All rights reserved.

aws

ALB has the above pre-defined security policies that are available in a dropdown to select from to apply to your LB.

Server Name Indication (SNI)

Host multiple TLS secured applications, each with its own TLS certificate

Bind multiple certificates to the same secure listener on your load balancer

ALB will automatically choose the optimal TLS certificate for each client

Support for both the classic RSA algorithm and the newer, faster elliptic-curve-based ECDSA algorithm

Integration with ACM



AWS re:Invent

© 2017, Amazon Web Services, Inc. or its Affiliates. All rights reserved.



Server Name Indication **SNI** allows you to have multiple domains on the same LB with each with its own TLS certificate. This allows you to put more applications behind your LB and be more efficient. It is integrated with ACM where you can get free SSL certificates that are updated automatically.

Native IPv6 support



AWS re:Invent

© 2017, Amazon Web Services, Inc. or its Affiliates. All rights reserved.



When you create your LB, there is a dropdown menu where you can choose to create a 'dual stack' and you will get an IPv4 and IPv6 address for your LB.

Application Load Balancer with WAF

Monitor web requests and protect web applications from malicious requests at the load balancer

Block or allow requests based on conditions such as IP addresses

Preconfigured protection to block common attacks, such as SQL injection or cross-site scripting

Set up web ACLs and rules from WAF console and apply them to the load balancer



AWS re:Invent

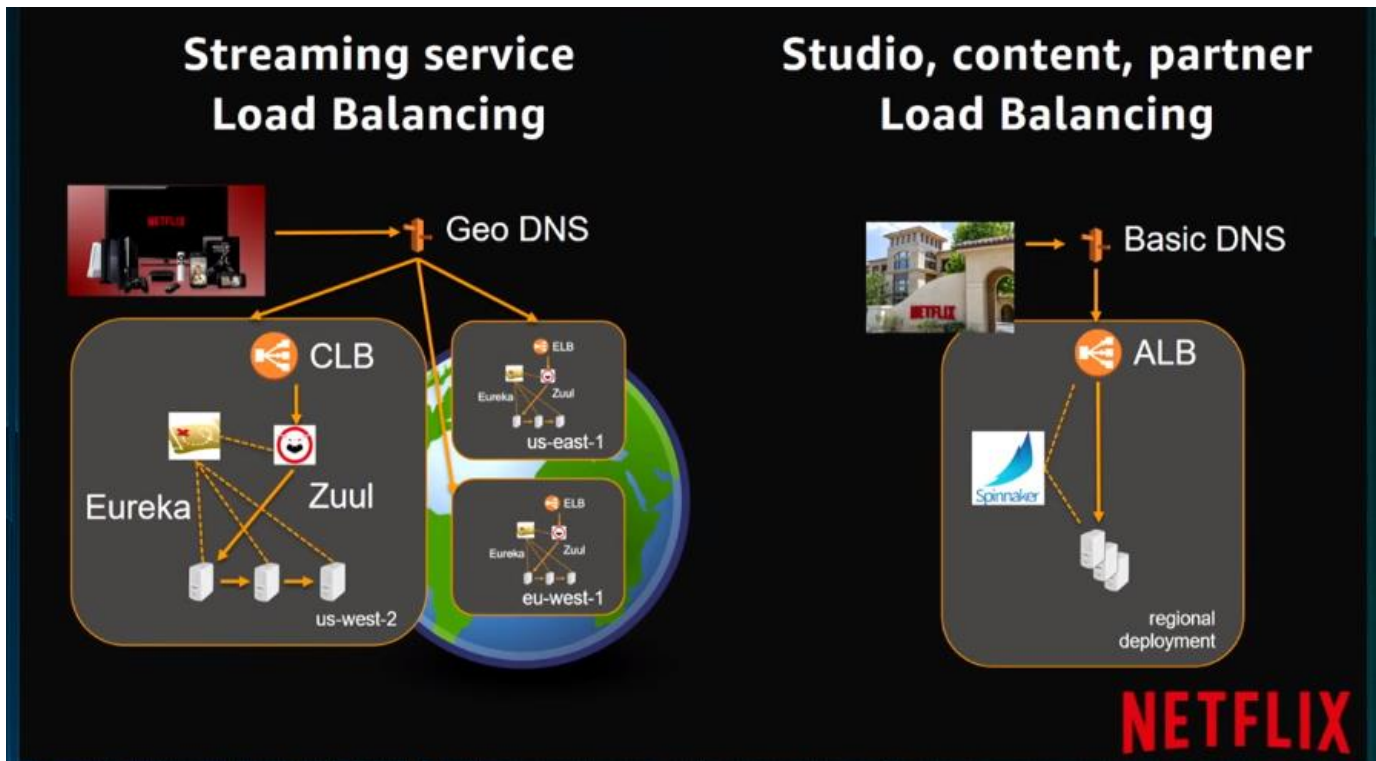
© 2017, Amazon Web Services, Inc. or its Affiliates. All rights reserved.



Web Application Firewall **WAF** allows every request that comes through your LB gets inspected to be allowed or rejected if you turn WAF turned on on your LB, we can look for SQL injection and XSS attacks in here using a WAF on our LB. There are now **managed policies on WAF** that you can get from **Trend Micro** and **AlertLogic** that you can add to your WAF, this automatically gets updated to keep your security up to date.

NETFLIX

IP as target demonstration



Netflix has advanced LB technology being used in their streaming services. As you use Netflix, you are going through Route53 and getting **Geo-located across us-east-1, eu-west-1 and us-west-2** which ever one is closest to you. Below that level, we then start using a CLB to start talking to the Netflix OSS **cloud gateway** technology **Zuul**, then in the **microservices layer** we use the OSS technology called **Eureka**. All these layers of LB help provide **fault-tolerance** such that if something goes wrong in an AWS region we can start to shift traffic away from that region at the Zuul layer across regions to change DNS and start moving traffic around the world. It takes about 6 minutes to take traffic out of a region like us-east-1 and split it across us-west-1 and eu-west-1 using Zuul without the customer having any downtime. Netflix also uses ALB for studio, content and partner apps where they use Spinnaker to configure their ALB.

Netflix containers—Titus

EC2 applications should work in containers unchanged

- Provide IP per container
- Native VPC, security group, IAM role support

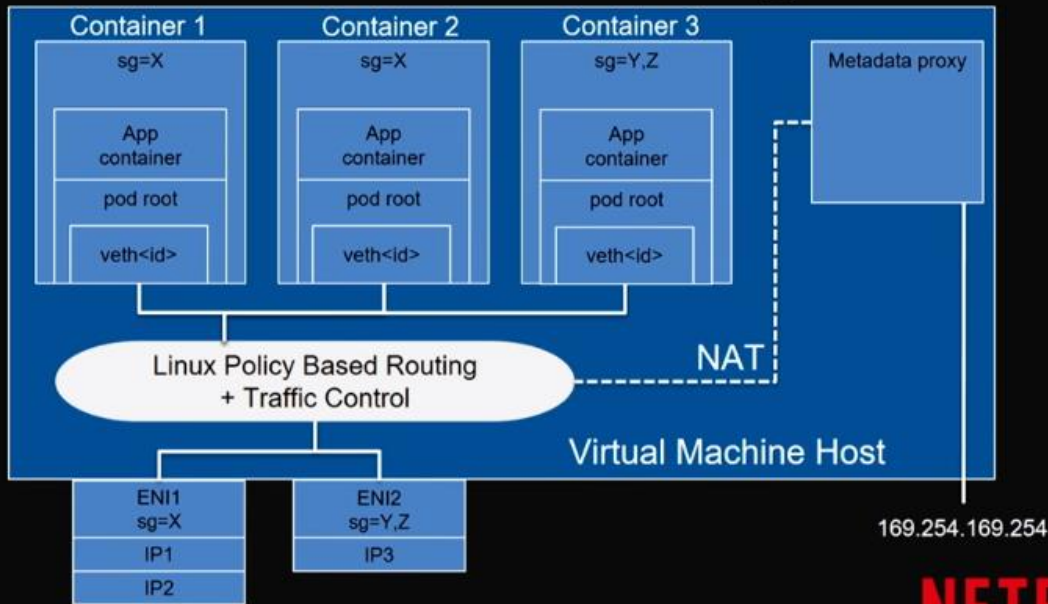
Problems for load balancing

- Multitenant container EC2 hosts presented problems



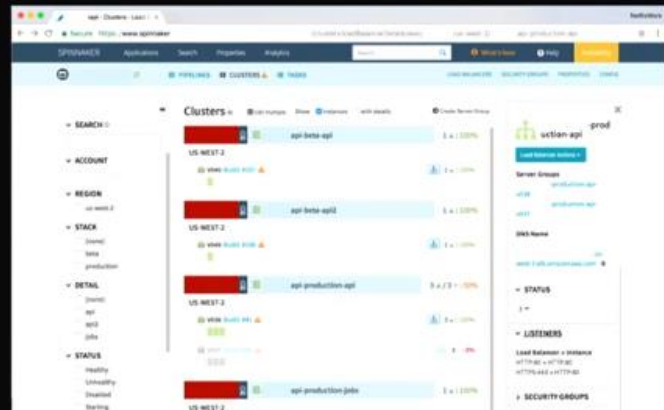
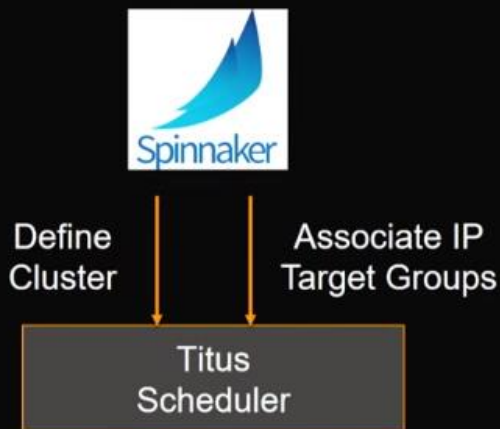
Titus is Netflix's container management platform that currently launches over 1 million containers per week. Titus is **natively AWS integrated** and can network full IP stacks for each of the containers so that we don't need to do port mapping or use an overlay network, we want simple mapping into VPC that gives us SGs and IAM roles.

Titus container networking



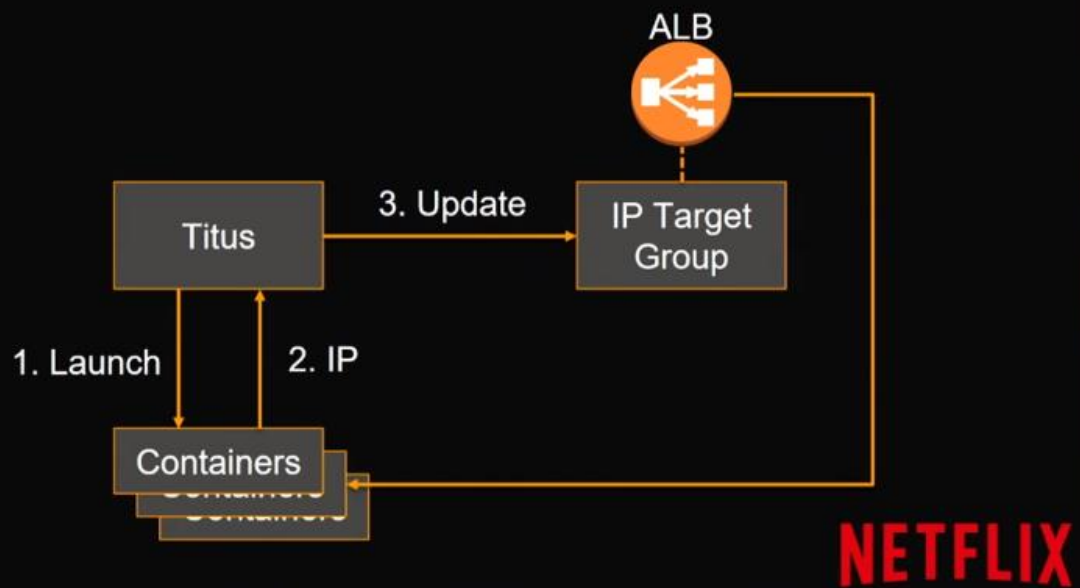
This is what the network stack looks like on a single EC2 instance that is running multiple containers in Titus. The networking magic maps the running containers within this EC2 instance to 2 different ENIs and 3 different IP addresses.

Configuring ALBs with Spinnaker



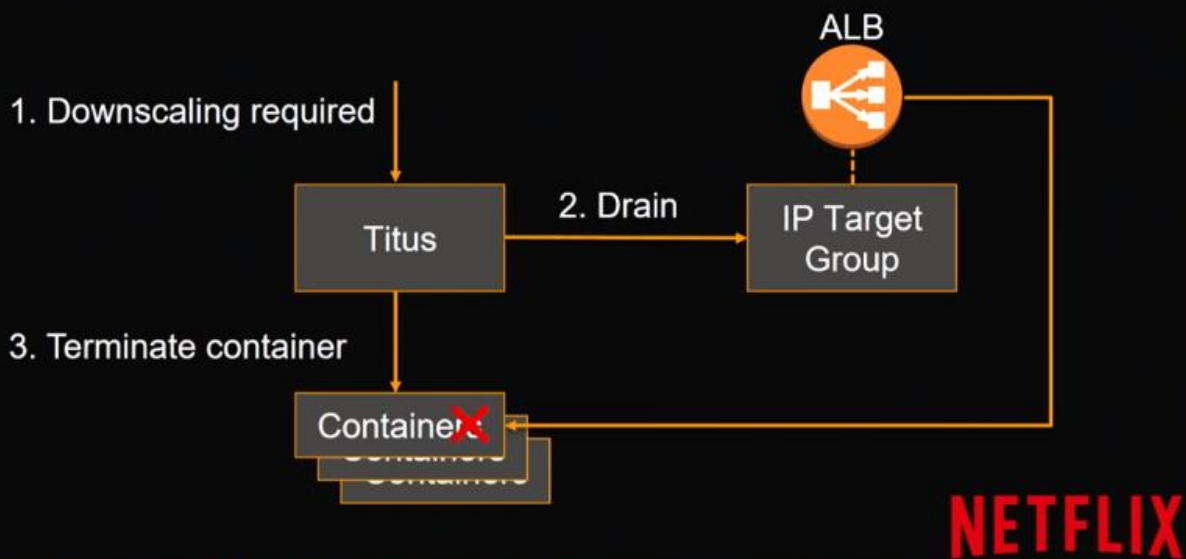
Spinnaker is the Netflix Continuous Deployment platform. Spinnaker allows managing ALBs and associate them with container clusters. Spinnaker takes your image, LBs, SGs, IAM Roles, all the things that go together to create a microservice, and it exposes it out into a single container for management. For ALB, we define a cluster and associate a target group to use.

ALB IP target group registration



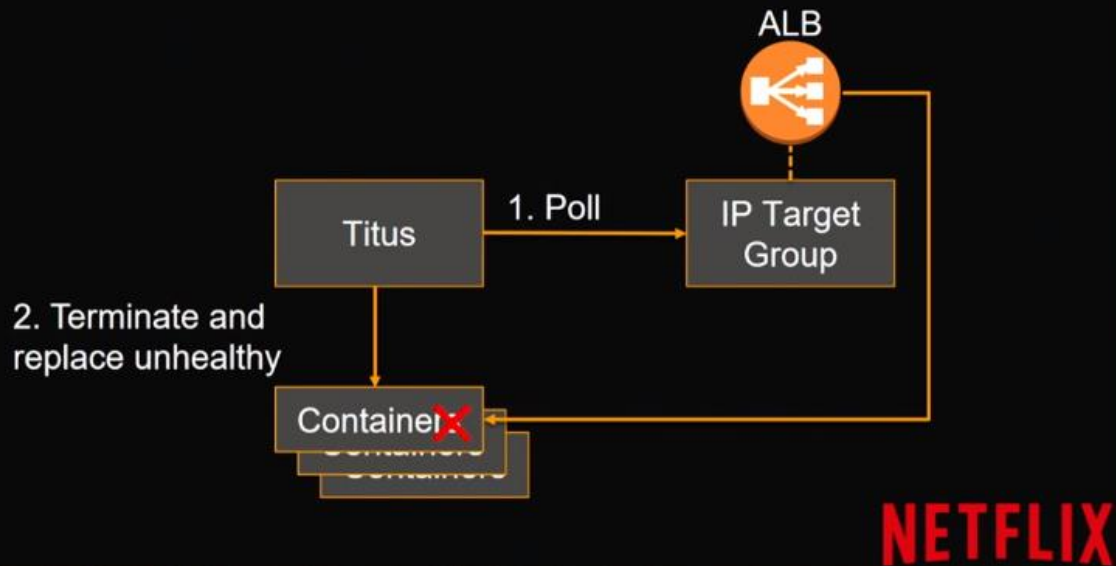
You associate the ALB to an initially empty target group, but as containers start up or containers are replaced, Titus launches a container and configures the VPC resources for that container, gets back an IP address for that instance, then updates the IP target group with the details for that container using the ALB API. At that point, the ALB can start flowing traffic down to the container.

Advanced features—downscaling



The ALB APIs also lets us do the above. If some container shuts down due to auto-scaling and we see a downscaling event come into Titus, we make a call to the ALB service and say 'can you start draining this IP address in this IP target group?', when it is done, then we can terminate the container.

Advanced features—healthchecks



The ALB is constantly doing health checks over containers, Titus is constantly polling IP target groups for the health status of all the instances, when Titus sees a failing health check for a number of time, Titus will then terminate the container instance and replace it with a new healthy instance.

IP as a target

Use any IPv4 address from the **load balancer's VPC CIDR** for targets within load balancer's VPC in RFC 1918 ranges (10.0.0.0/8, 172.16.0.0/12, and 192.168.0.0/16)

Use any IP address from the RFC 6598 range (100.64.0.0/10) for targets located outside the load balancer's VPC (this includes **Peered VPC, EC2-Classical, and on-premises targets reachable over Direct Connect or VPN**).



AWS re:Invent

© 2017, Amazon Web Services, Inc. or its Affiliates. All rights reserved.

aws

Cross-zone load balancing

Requests distributed evenly across multiple Availability Zones

Load balancer absorbs impact of DNS caching

Eliminates imbalances in backend instance utilization

No additional bandwidth charge for cross-zone traffic

Enabled on all ALBs

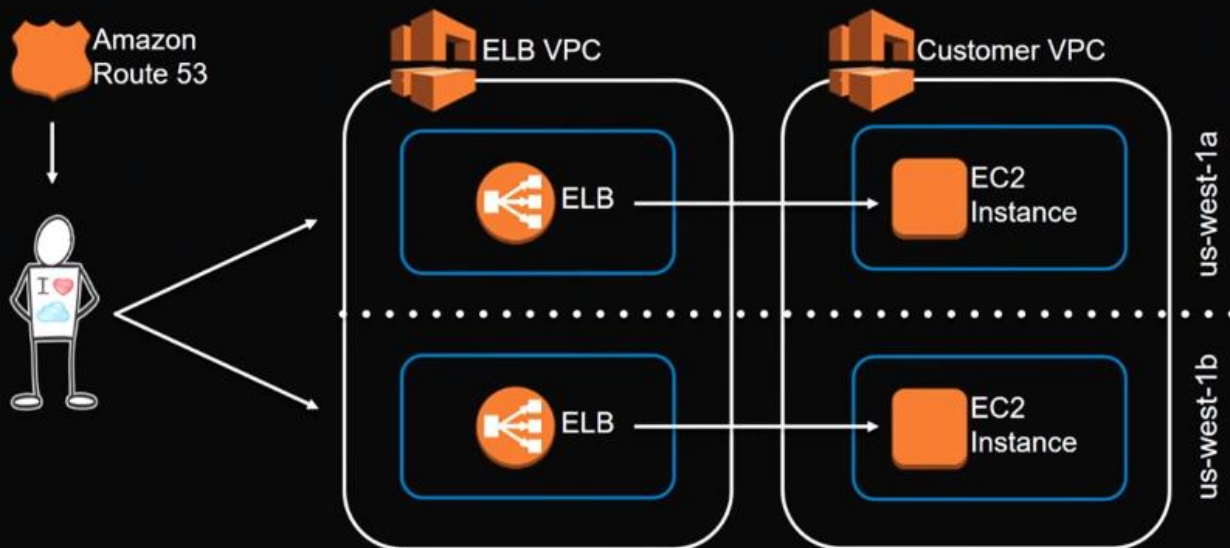


AWS
re:Invent

© 2017, Amazon Web Services, Inc. or its Affiliates. All rights reserved.



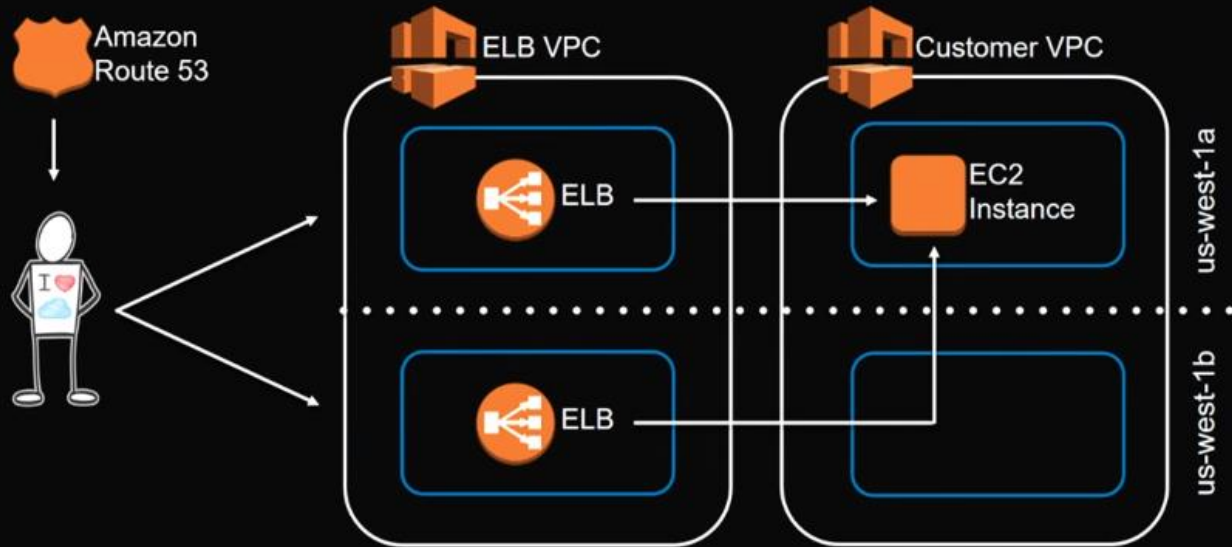
Multiple Availability Zones



© 2017, Amazon Web Services, Inc. or its Affiliates. All rights reserved.

Above is a functioning application that is using 2 AZs and target groups in each AZ as recommended

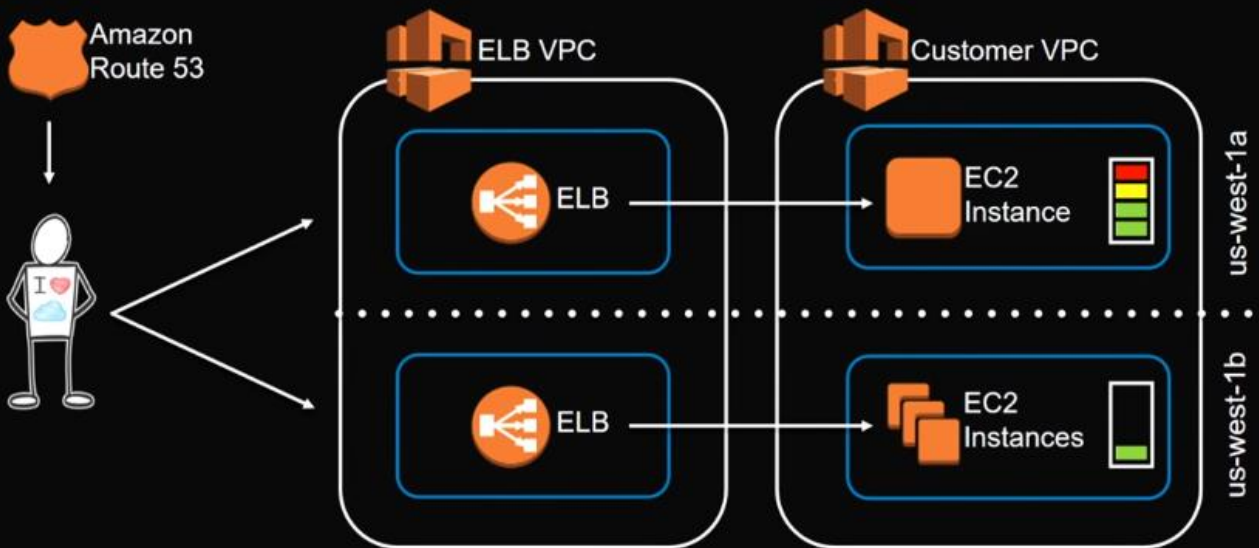
Multiple Availability Zones



© 2017, Amazon Web Services, Inc. or its Affiliates. All rights reserved.

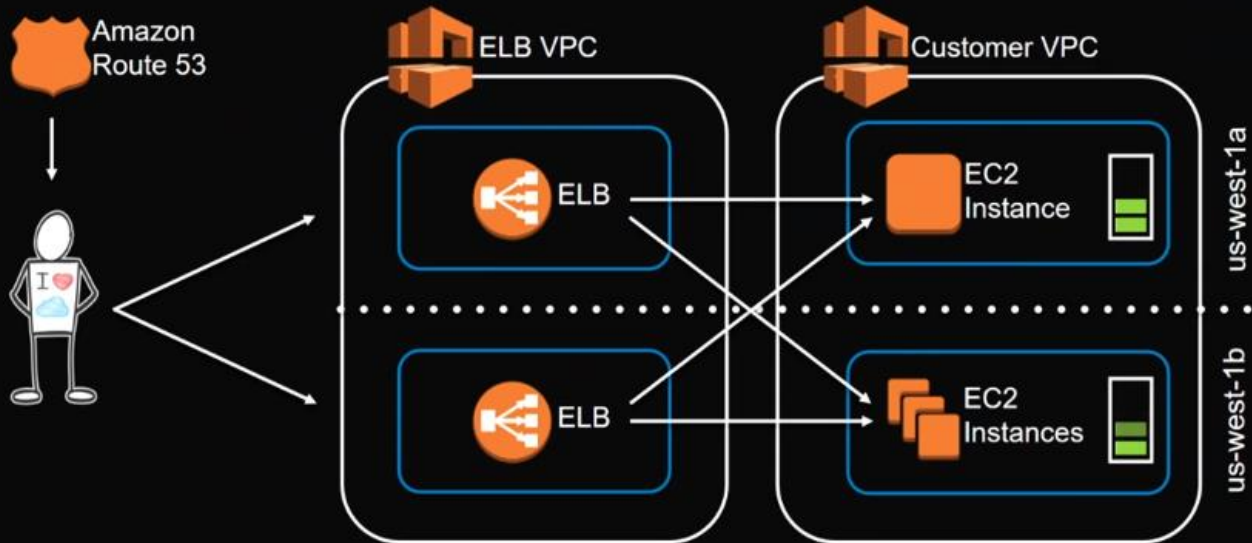
If targets in one of the AZ starts failing, the node in us-west-1b can still reach the targets in us-west-1a for HA. When possible, put resources in both AZs to enable this HA feature.

Cross-zone load balancing



© 2017, Amazon Web Services, Inc. or its Affiliates. All rights reserved.

Cross-zone load balancing



© 2017, Amazon Web Services, Inc. or its Affiliates. All rights reserved.

This helps to distribute traffic load evenly

Amazon CloudWatch metrics



CloudWatch metrics provided for each load balancer at 1-minute granularity

Request response times provided with percentile dimensions in the 90th, 95th, 99th or 99.9th percentile

Provide **detailed insight** into the health of the load balancer and application stack

CloudWatch alarms can be configured to notify or take action if any metric goes outside of the acceptable range

AWS re:Invent

© 2017, Amazon Web Services, Inc. or its Affiliates. All rights reserved.

aws

There is no charge for using this with the ALB. ALB also provides you with a lot of CloudWatch metrics at 1-minute granularity

Healthy host count



The count of the number of healthy instances in each Availability Zone

Most common cause of unhealthy hosts is health check exceeding the allocated timeout

Test by making repeated requests to the back-end instance from another EC2 instance

View at the zonal dimension

AWS re:Invent

© 2017, Amazon Web Services, Inc. or its Affiliates. All rights reserved.



Latency

Measures the elapsed time, in seconds, from when the request leaves the load balancer until the response is received

Test by sending requests to the back-end instance from another instance

Using min, average, and max CloudWatch stats, provide upper and lower bounds for latency

Debug individual requests using access logs



AWS re:Invent

© 2017, Amazon Web Services, Inc. or its Affiliates. All rights reserved.



You can actually use this as a metric to scale your ASG

Rejected connection count

The number of connections that were rejected because the load balancer could not establish a connection with a healthy target in order to route the request

This replaces the surge queue metrics that are used by the Classic Load Balancer

Surge queues often impact client applications, which fast request rejection improves

Normally a sign of an under-scaled application



AWS
re:Invent

© 2017, Amazon Web Services, Inc. or its Affiliates. All rights reserved.



This means that the ALB is getting requests and has no where to route them to because all the targets are busy, the ALB will then start dropping these requests. This is a sign of an under-scaled application

Target group metrics

The following metrics are now provided at the target group level, allowing for individual applications to be closely monitored:

- RequestCount
- HTTPCode_Target_2XX_Count
- HTTPCode_Target_3XX_Count
- HTTPCode_Target_4XX_Count
- HTTPCode_Target_5XX_Count
- TargetResponseTime (Latency)
- UnHealthyHostCount
- HealthyHostCount
- Request Count per Target



AWS
re:Invent

© 2017, Amazon Web Services, Inc. or its Affiliates. All rights reserved.



You can monitor performance of specific apps and target groups using these metrics

Access logs



Provide detailed information on each request processed by the load balancer

Includes request time, client IP address, latencies, request path, and server responses

Delivered to an Amazon S3 bucket every 5 or 60 minutes

AWS re:Invent

© 2017, Amazon Web Services, Inc. or its Affiliates. All rights reserved.



The access logs contain detailed information about the types of requests coming through the LB, you use this to deep dive into what is happening if you detect a problem and need to do some debugging

Application Load Balancer pricing

With the Application Load Balancer, you pay only for what you use. You are charged for each hour or partial hour your Application Load Balancer is running and the number of Load Balancer Capacity Units (LCUs) used per hour.

- **\$0.0225** per Application Load Balancer-hour (or partial hour)
- **\$0.008** per LCU-hour (or partial hour)

Hourly charge is 10% cheaper than Classic Load Balancer today, reducing the cost for the virtually all of our customers.



AWS re:Invent

© 2017, Amazon Web Services, Inc. or its Affiliates. All rights reserved.



Load balancer capacity units

An LCU measures the dimensions on which the Application Load Balancer processes your traffic (averaged over an hour).

The three dimensions measured are:

- 25 new connections per second
- 3,000 active connections per minute
- 2.22 Mbps (which translates to 1 GB per hour)
- 1,000 rule evaluations per second

You are charged only on the dimension with the highest usage over the hour



AWS
re:Invent

© 2017, Amazon Web Services, Inc. or its Affiliates. All rights reserved.



Migrating to Application Load Balancer

Publishing LCU Metrics for Classic Load Balancer, which allows customers to estimate pricing if they migrate from Classic to ALB

Migration is as simple as creating a new Application Load Balancer, registering targets, and updating DNS to point to the new CNAME



AWS
re:Invent

© 2017, Amazon Web Services, Inc. or its Affiliates. All rights reserved.



When you create a new ALB, we give you a new A-Record and you can start creating targets and putting them behind that ALB. Then using Route53, you can start weighting away from your classic load balancer CLB to the new ALB. It is very easy via DNS to create a LB and start testing things.

Migration Wizard

The migration wizard in the AWS console makes it simple to create an Application Load Balancer with a configuration that is equivalent to your Classic Load Balancer

Enables you to quickly test your application with a new type of load balancer

After migration, you can configure the advanced features offered by the new load balancer



AWS re:Invent

© 2017, Amazon Web Services, Inc. or its Affiliates. All rights reserved.



The wizard will create an ALB and replicate your CLB setup to your ALB

When should I use Application Load Balancer?

	Application Load Balancer	Network Load Balancer	Classic Load Balancer
Protocol	HTTP, HTTPS, HTTP/2	TCP	TCP, SSL, HTTP, HTTPS
SSL offloading	✓		✓
IP as target	✓	✓	
Path-based routing, Host-based routing	✓		
Static IP		✓	
WebSockets	✓	✓	
Container support	✓	✓	

© 2017, Amazon Web Services, Inc. or its Affiliates. All rights reserved.

Here is a feature breakdown for the different LBs

For **TCP**, use Network Load Balancer

For all other use cases, use Application Load Balancer

Learn more

<https://aws.amazon.com/elasticloadbalancing/>

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

Follow me on Twitter: @davidpessis

AWS
re:Invent

Thank you!

AWS
re:Invent

© 2017, Amazon Web Services, Inc. or its Affiliates. All rights reserved.

