

AWS

S U M M I T

Elastic Load Balancing

Deep Dive & Best Practices

Mariano Vecchioli, Sr. Technical Account Manager – AWS

Michaela Kurkiewicz, Principal Service Manager – Co-op

Tina Howell, Platform Lead - Co-op

June 28th, 2017



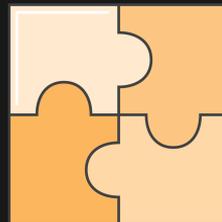
Elastic Load Balancing automatically distributes incoming application traffic across multiple applications, microservices and containers hosted on Amazon EC2 instances.



Elastic



Secure



Integrated



Cost Effective



**EC2
Instance**



ELB



EC2 Instance



EC2 Instance

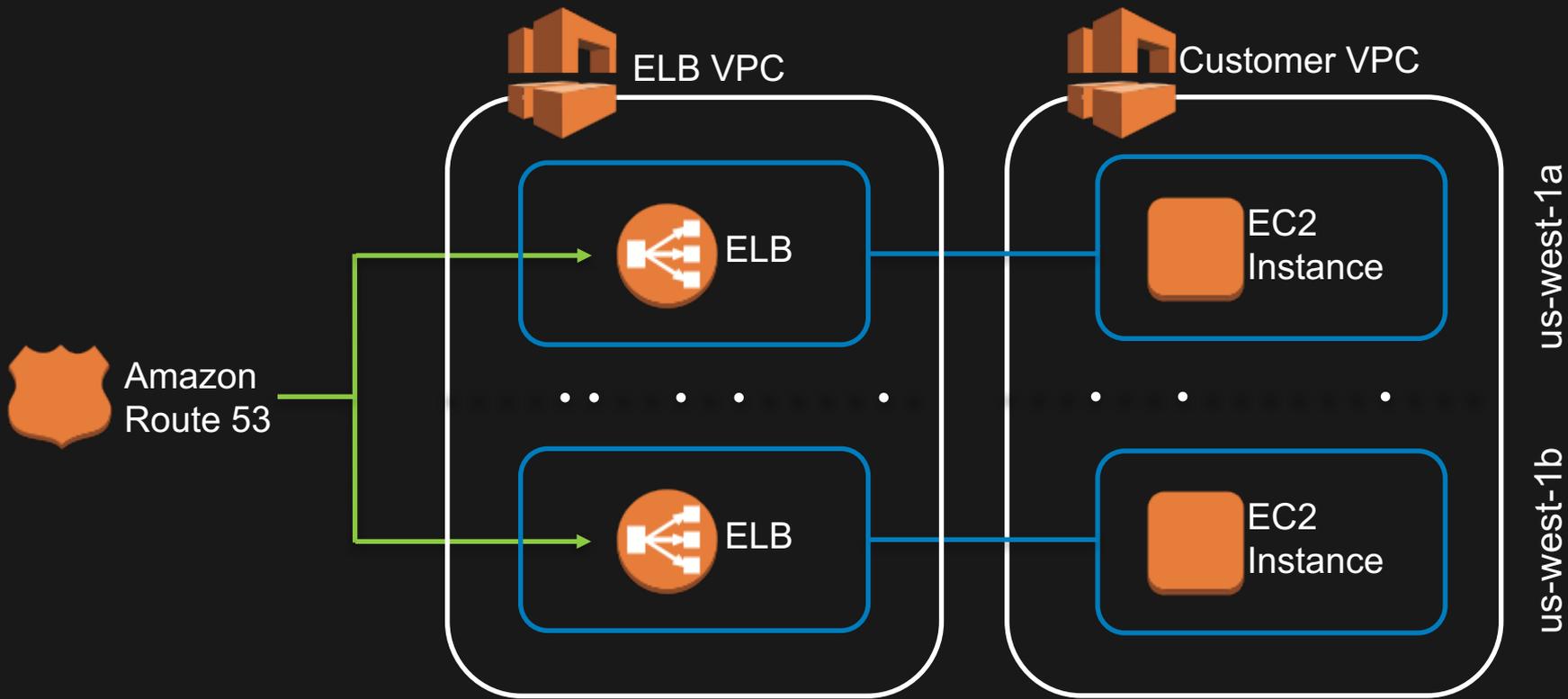


EC2 Instance

-
-
-
-
-
-

Load Balancer used to route incoming requests to multiple EC2 instances.

Elastic Load Balancing provides
high availability by utilizing **multiple**
Availability Zones



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



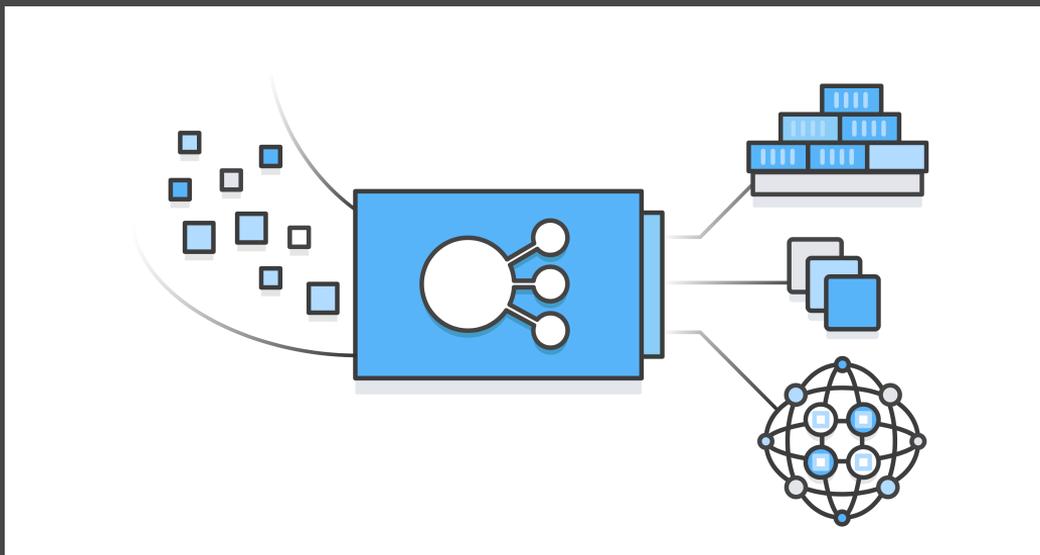
Layer 7 (application)

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

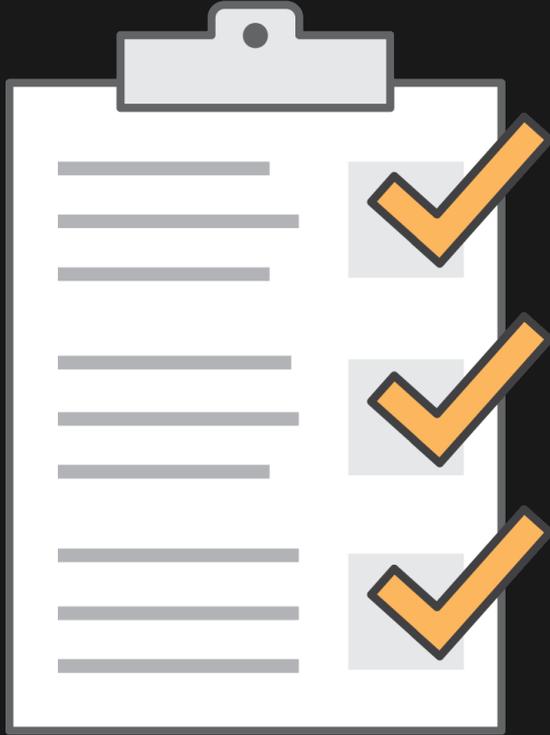


Application Load Balancer

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

	Classic	Application
Protocol	TCP, SSL, HTTP, HTTPS	HTTP, HTTPS
Platforms	EC2-Classic, EC2-VPC	EC2-VPC
Health checks	✓	Improved
CloudWatch metrics	✓	Improved
Path-based routing		✓
Container support		✓
WebSockets & HTTP/2		✓

Application Load Balancer



New, feature rich, **layer 7 load balancing platform**

Fully-managed, scalable and highly available load balancing platform

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

**Application Load Balancer allows for
multiple applications to be hosted
behind a single load balancer**



ELB



EC2 Instance



EC2 Instance



EC2 Instance

-
-
-
-
-
-
-
-
-
-

EC2 instances registered behind a **Classic Load Balancer**

orders.example.com



ELB



EC2 Instance



EC2 Instance



EC2 Instance



ELB



EC2 Instance



EC2 Instance



EC2 Instance

images.example.com



Running two separate applications with **Classic Load Balancer** requires multiple load balancers



example.com



ELB

/orders

/images



Application Load Balancer allows for multiple applications to be hosted behind a single load balancer



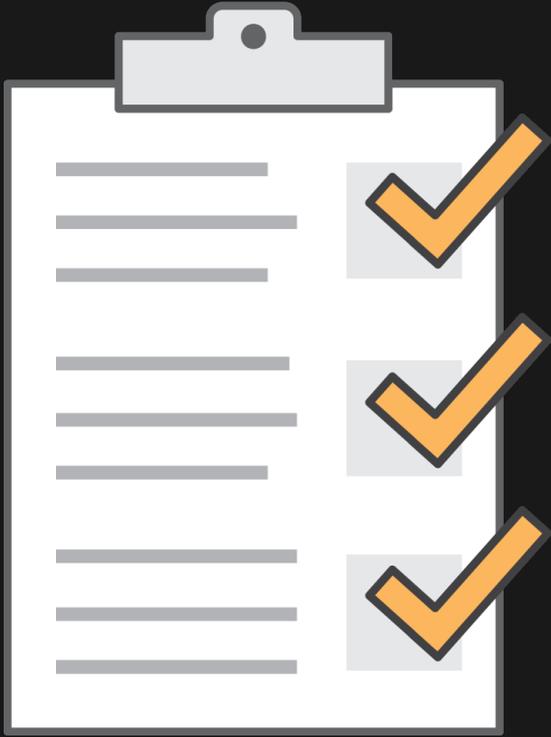
Multiple applications behind a single load balancer provides a significant cost saving



Consider **blast radius and isolation** when
grouping applications behind a
single load balancer

**Application Load Balancer provides
native support for **microservice and
container-based architectures****

Application Load Balancer



Instances can be registered with **multiple ports**, allowing for requests to be routed to multiple containers on a single instance

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

Can also be used with other container technologies



example.com



ELB

/orders

/images



Application Load Balancer allows for multiple applications to be hosted behind a single load balancer



example.com



ELB

/orders

/images



Application Load Balancer allows containers to be registered with the load balancer



Microservice and container-based architectures provide further cost savings by improving resource utilization

Who are the Co-op?



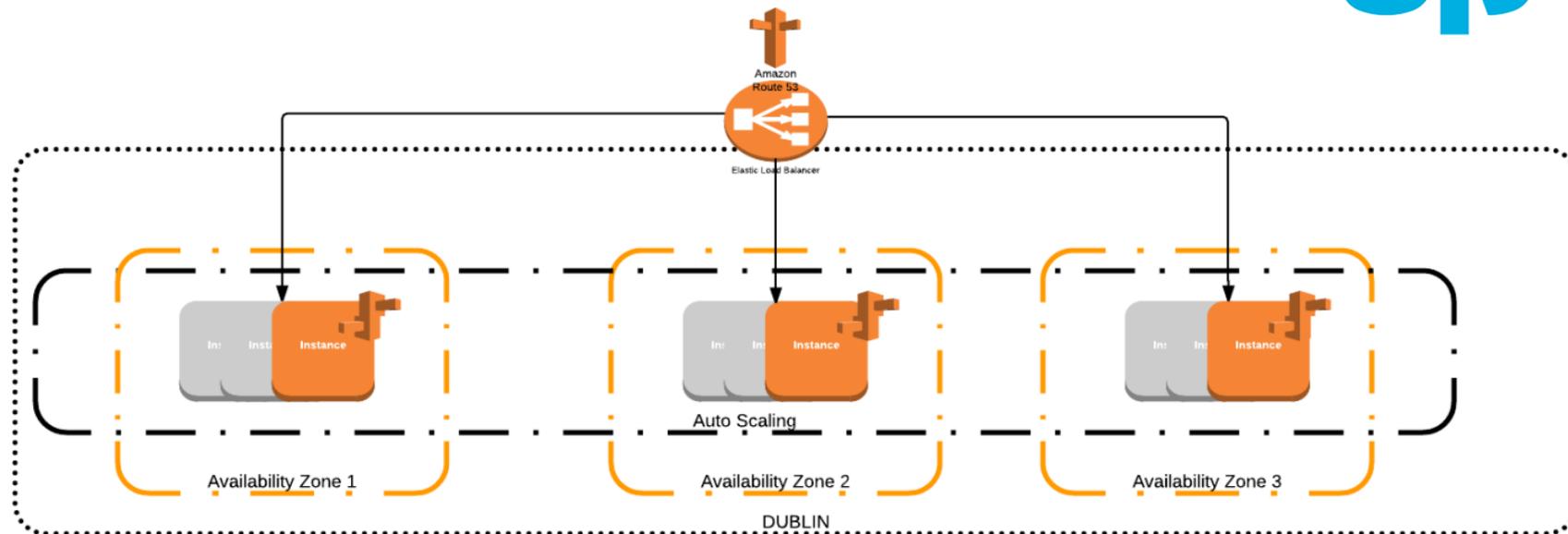
- Food, General Insurance, Legal Services, Electrical, Funeralcare
- Digital team formed in 2016
- Membership
 - 4.5 million active members
 - 5% for you, 1% for your community
 - 12 months from vision to beta

Our challenges

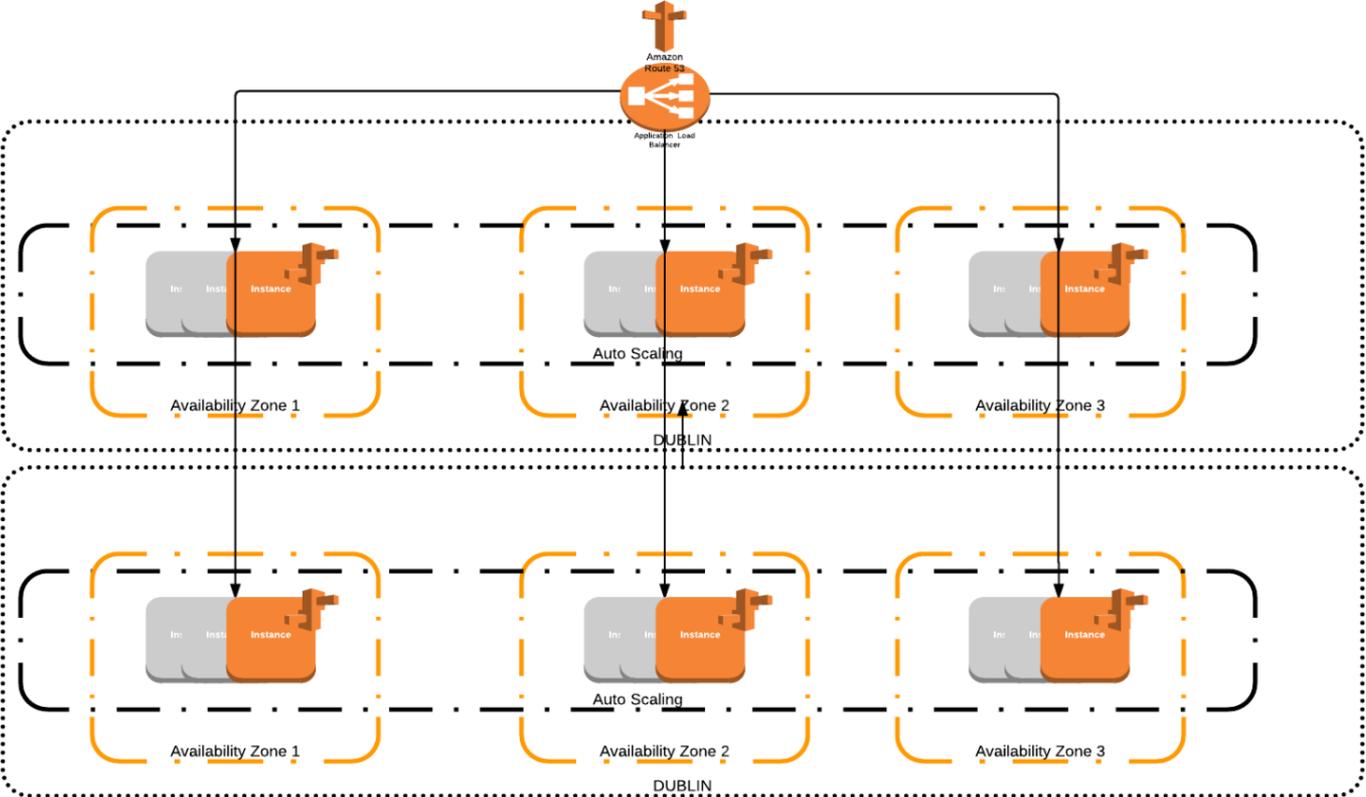


- Speed of making changes
(6 weeks to do a port change)
- Infrastructure monitoring controlled by another team - only found out something was wrong when it was no longer available
- High availability, scalable, low cost per change

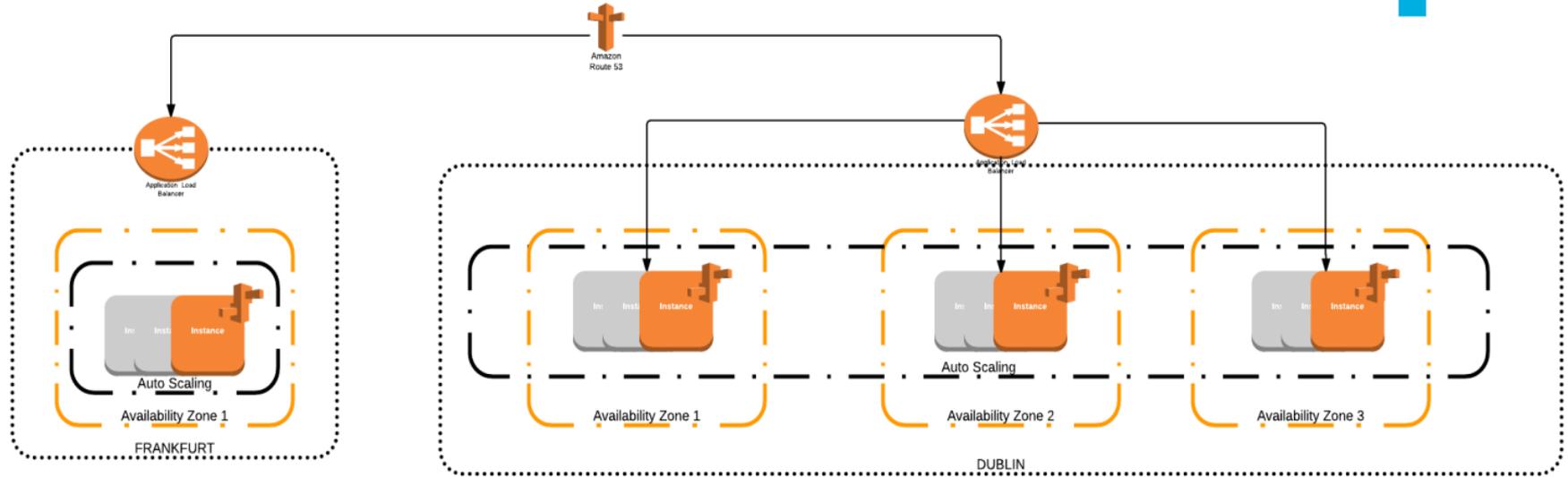
Templated Environment



Using one ALB for multiple products



What's next: Moving to latency-based DNS for multi-region applications



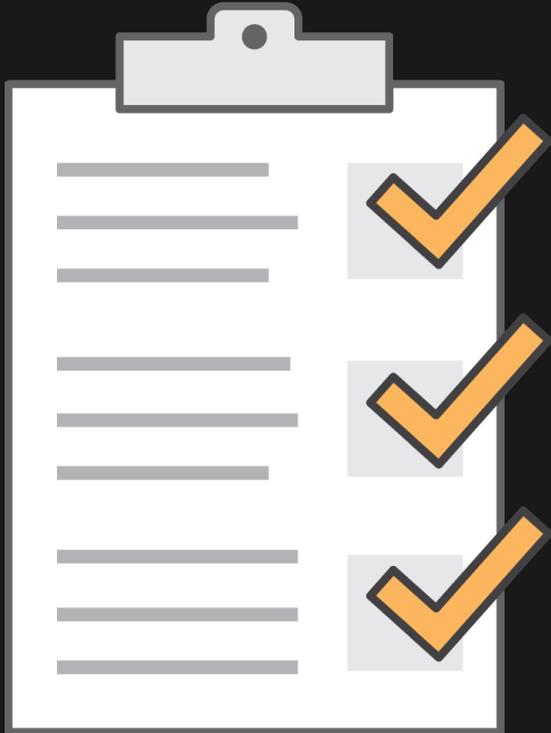
What this has given us

- Resiliency
- High availability
- Security
- Enabled faster rate of change

What's next



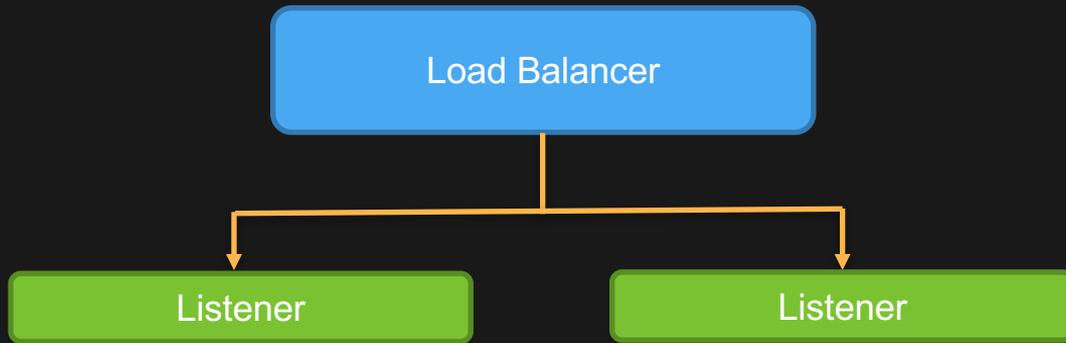
Application Load Balancer



New API version provided for creating, configuring and managing Application Load Balancers

Follows latest **AWS best practices** for resource identifiers and API design

Provides several new resource types, including **target groups**, **targets** and **rules**



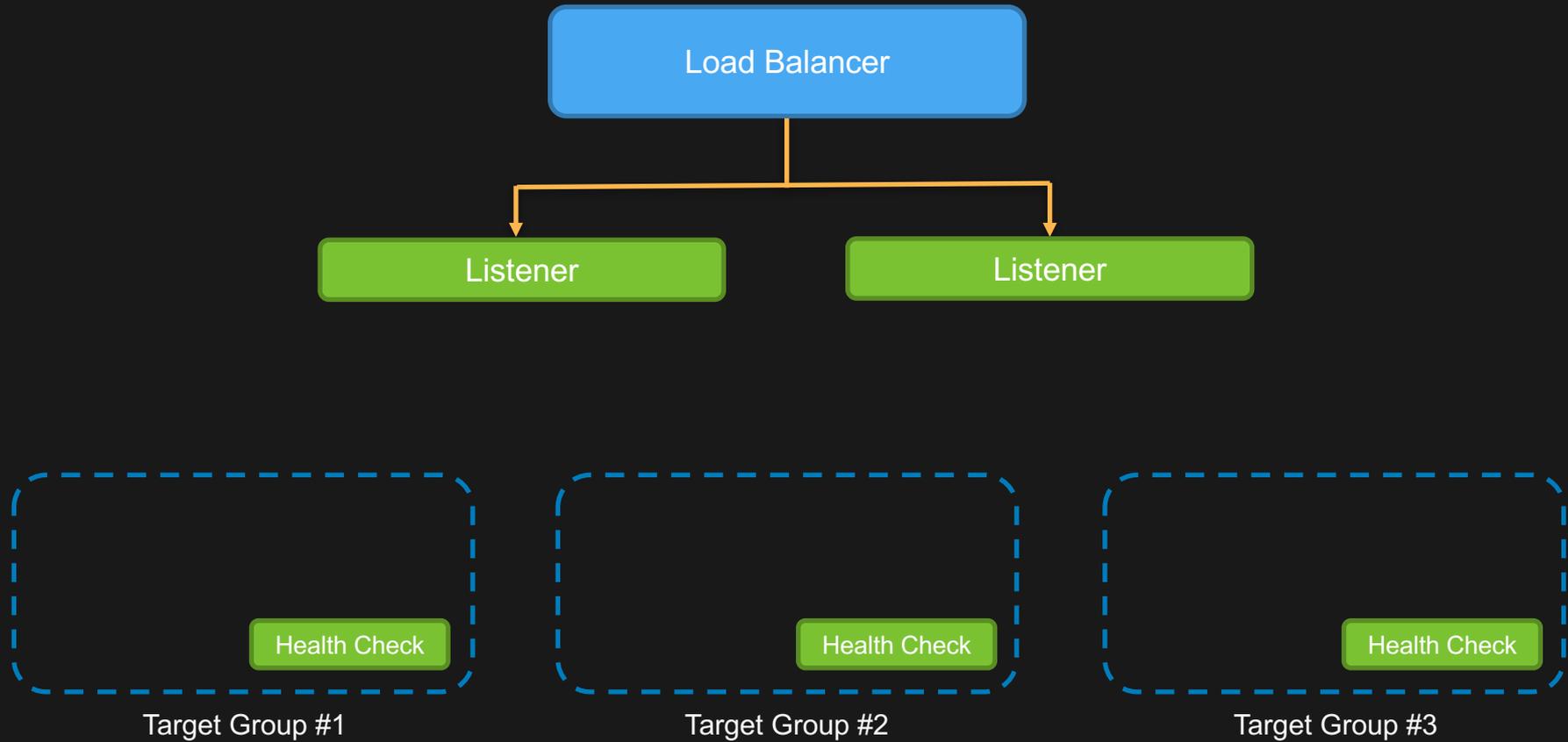
Listeners



Define the **protocol and port** on which the load balancer listens for incoming connections

Each load balancer needs **at least one listener** to accept incoming traffic, and can support **up to 10** listeners

Routing rules are defined on listeners



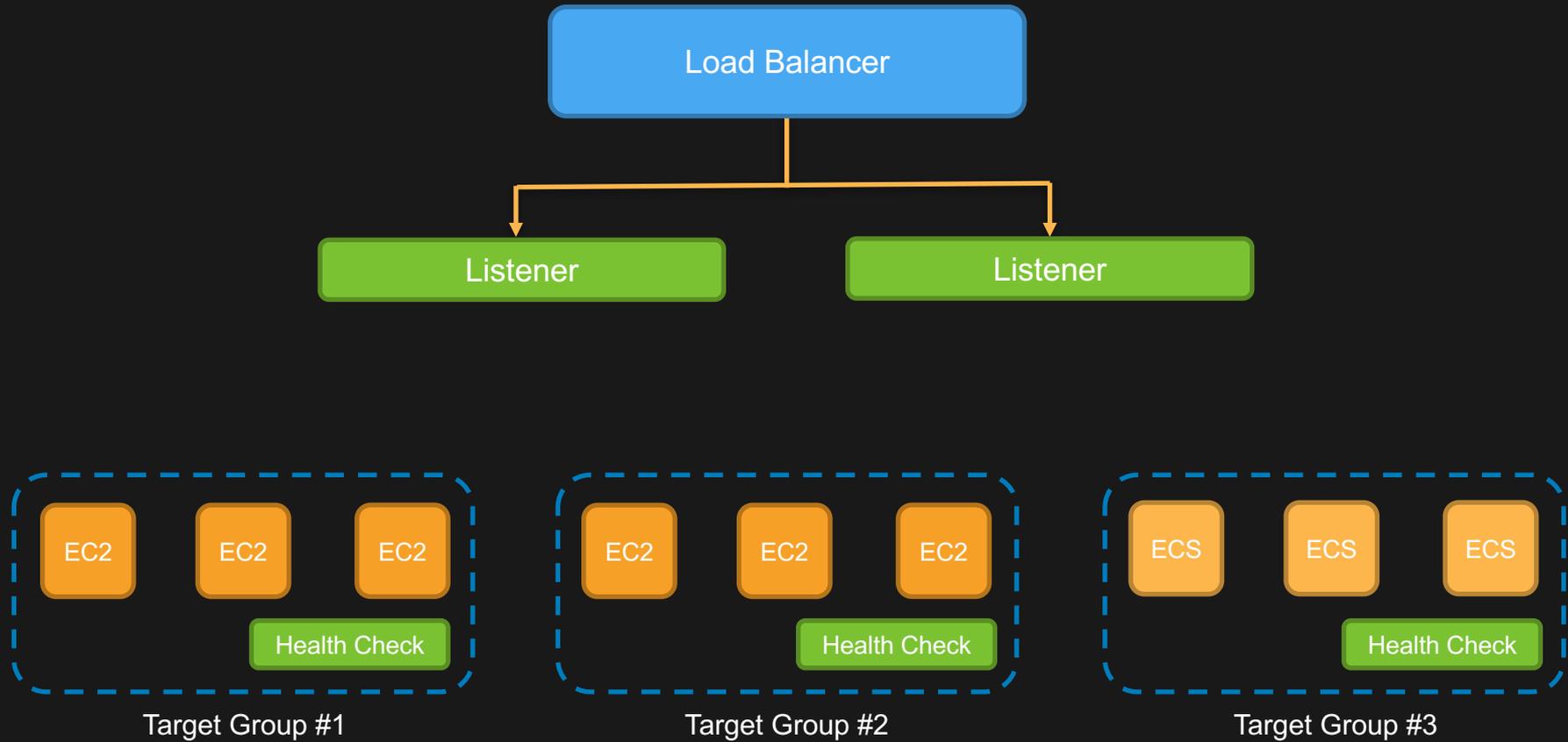
Target Groups

Logical grouping of targets
behind a load balancer

Target groups can **exist independently**
from the load balancer, and be associated
with a load balancer when needed

Regional construct that can be associated
with **Auto Scaling group**





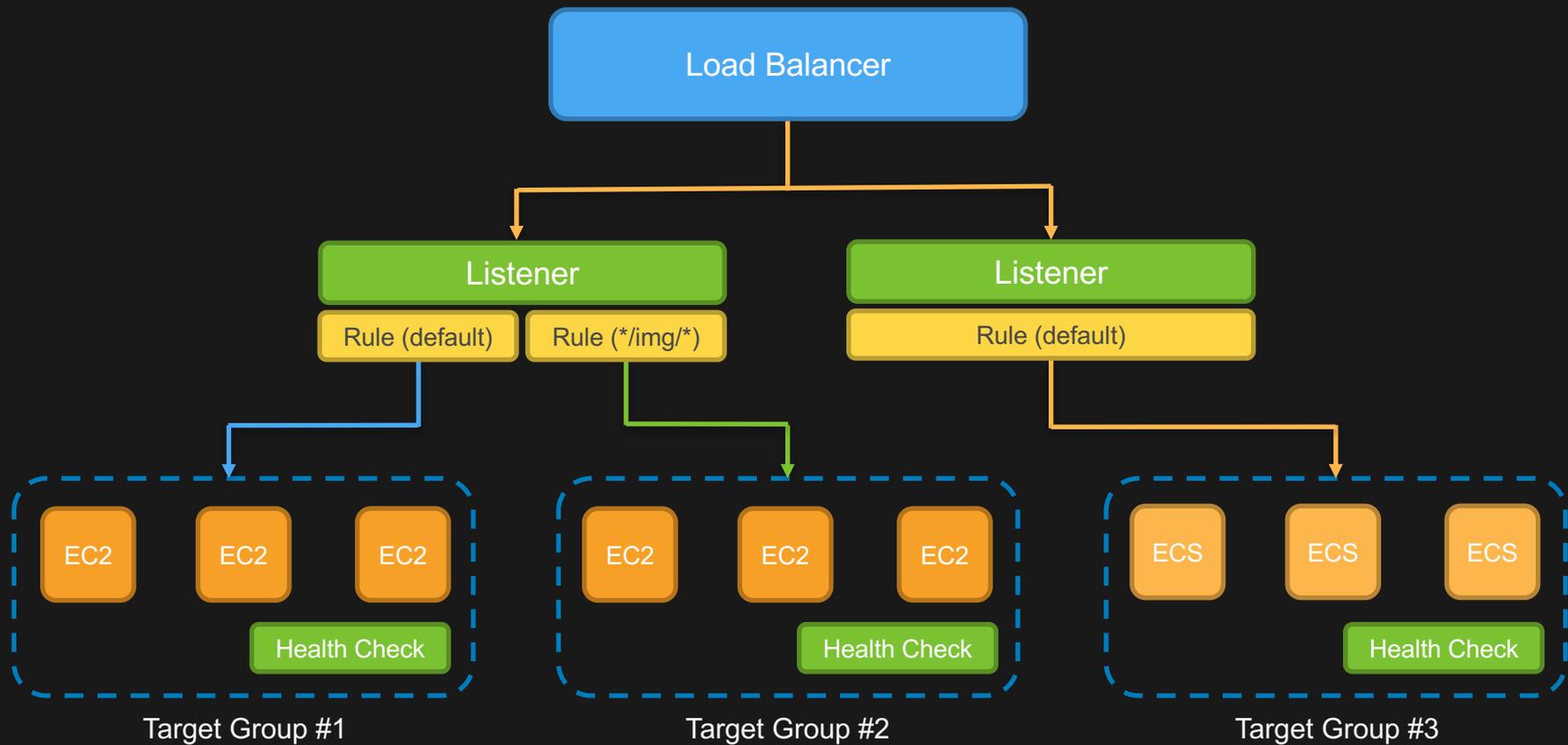
Targets



Logical load balancing target, which can be an **EC2 instance**, **microservice**, or **container-based application**

EC2 instances can be registered with the same target group using **multiple ports**

A single target can be registered with **multiple target groups**



Rules



Provide the **link between** listeners and target groups, and consist of **conditions and actions**

When a request **meets the condition** of the rule, the associated **action is taken**

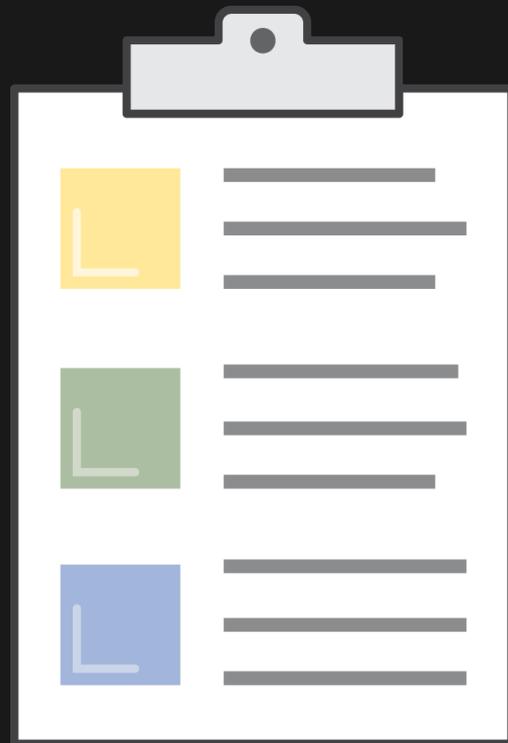
Today, rules can **forward** requests to a specified target group

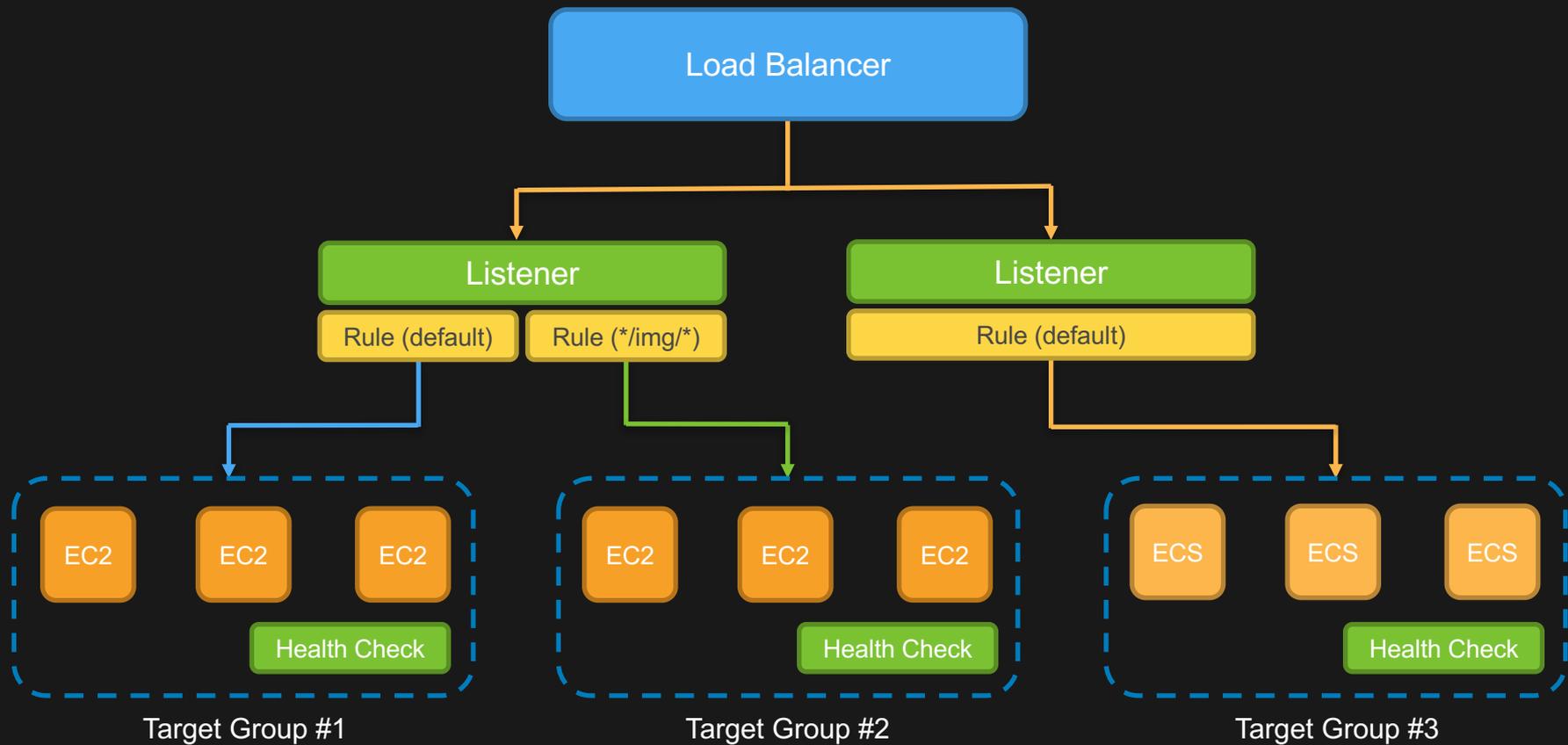
Rules (continued)

Conditions can be specified in **path pattern** format

A path pattern is **case sensitive**, can be up to **128 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)







Use API **deletion protection** to prevent a load balancer from being erroneously deleted

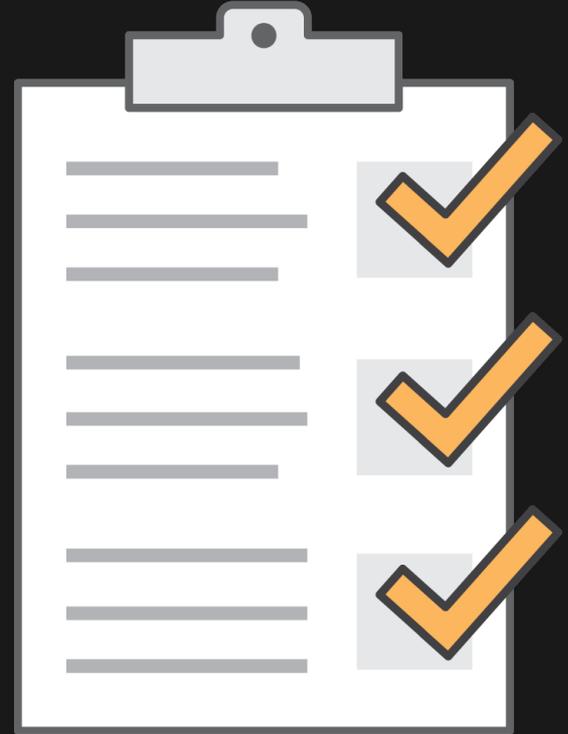
Application Load Balancer provides
improved performance for
Internet applications

Application Load Balancer

Native support for **WebSockets**, supporting full-duplex communication channels over a single TCP connection

Support for **HTTP/2** provides improved page load times from most of today's browsers

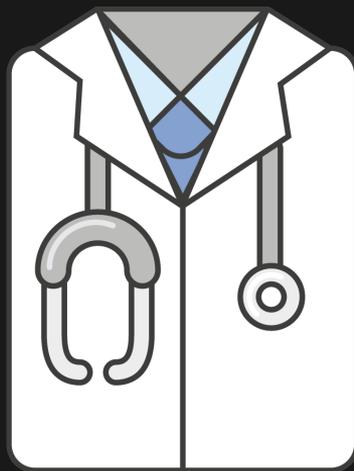
Improved performance for **real-time and streaming applications**



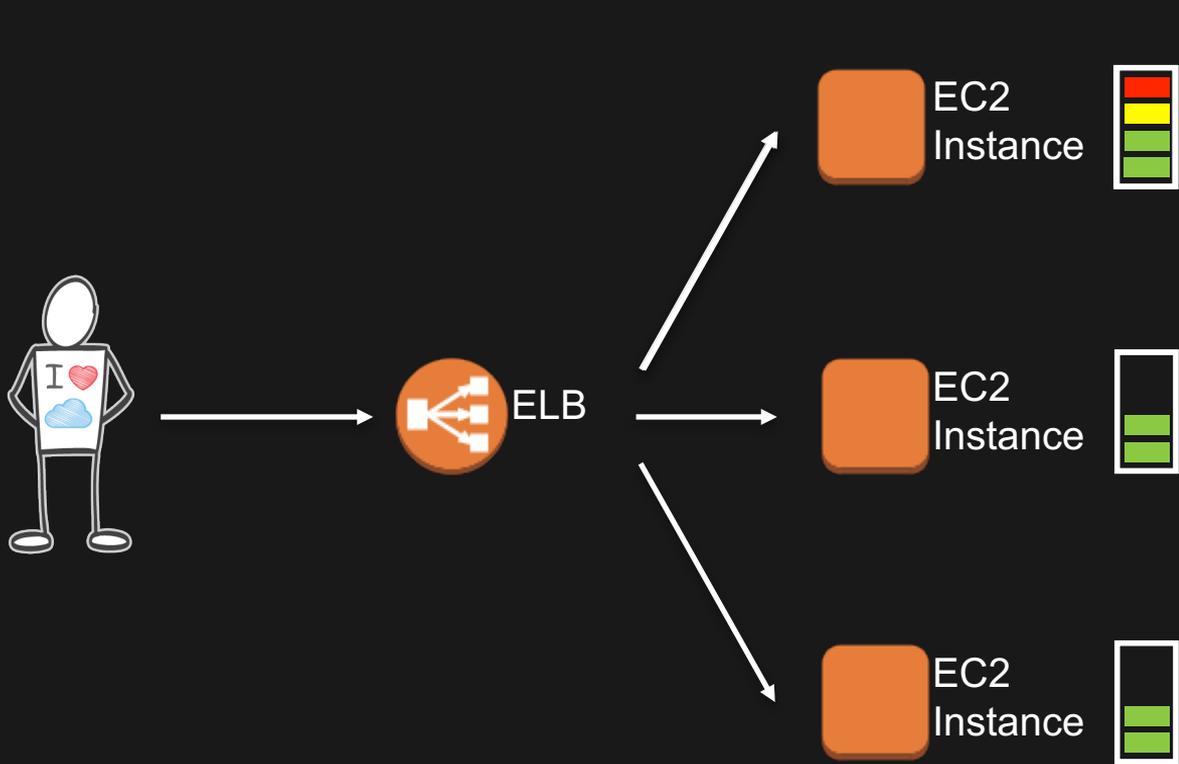
**Improvements to application
availability and scalability**



**EC2
Instance**



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

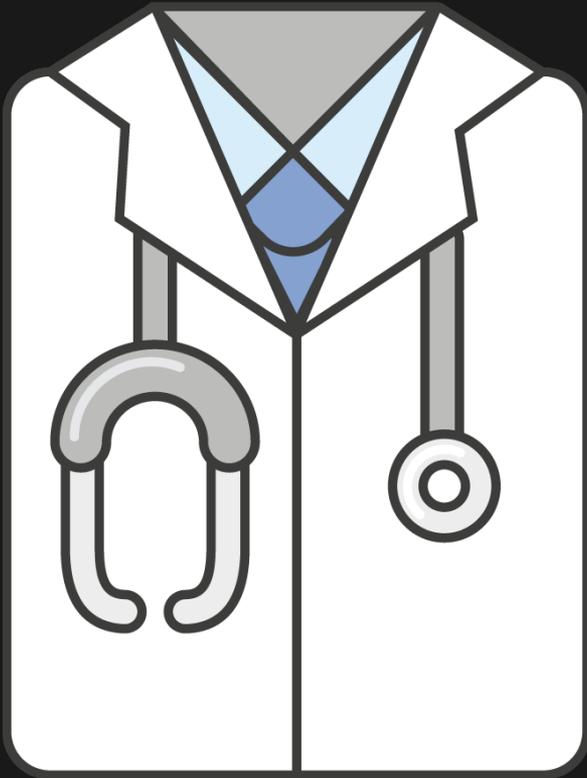


•
•
•

•
•
•
•

Health checks ensure that request traffic is **shifted away** from a failed instance.

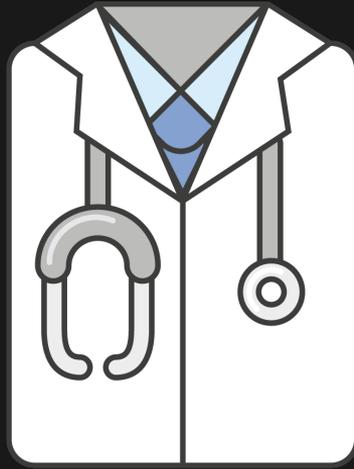
Health Checks



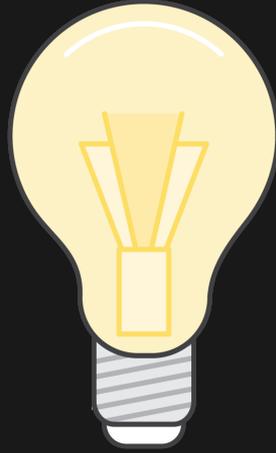
HTTP and HTTPS health checks

Customize the frequency, failure thresholds, and **list of successful response codes**

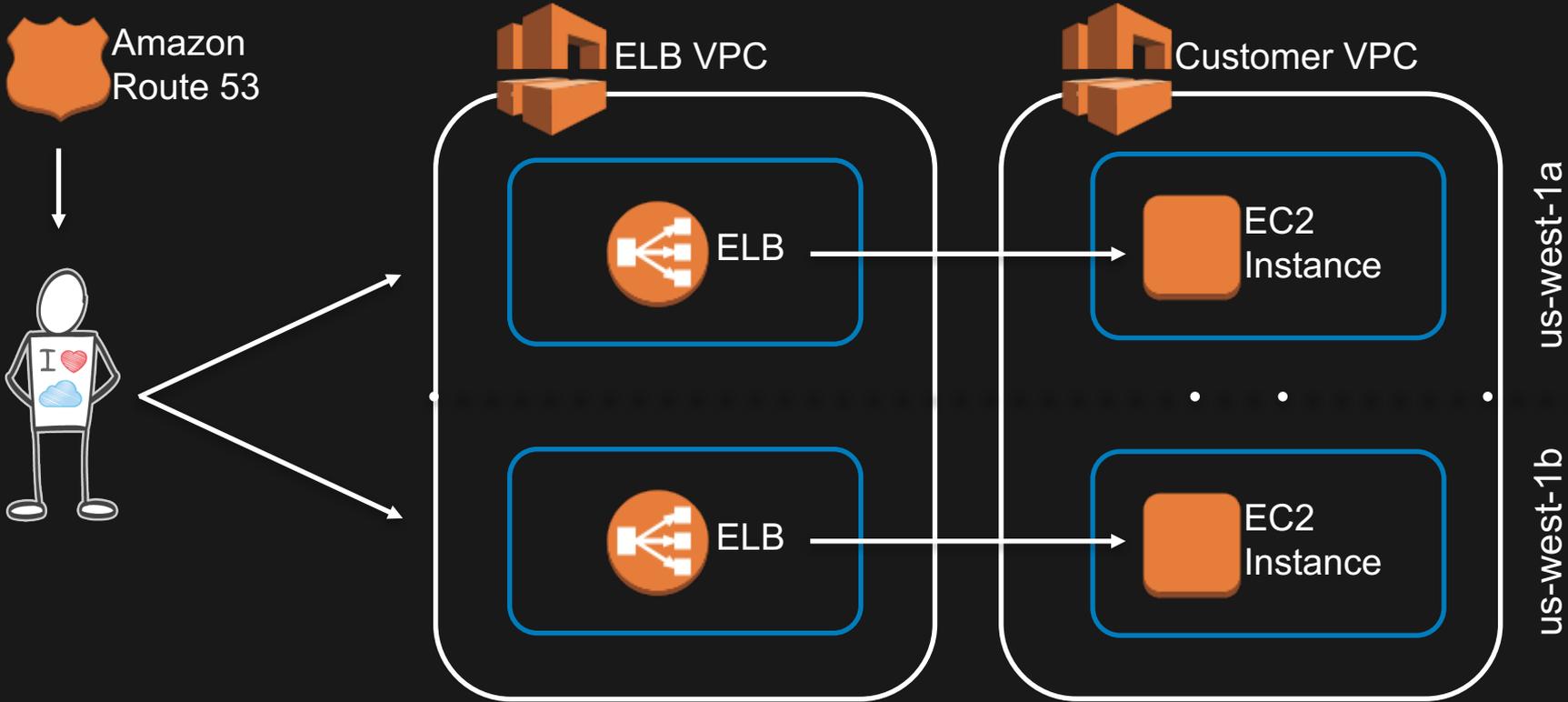
Detailed **reasons for health check failures are now returned via the API** and displayed in the AWS Management Console

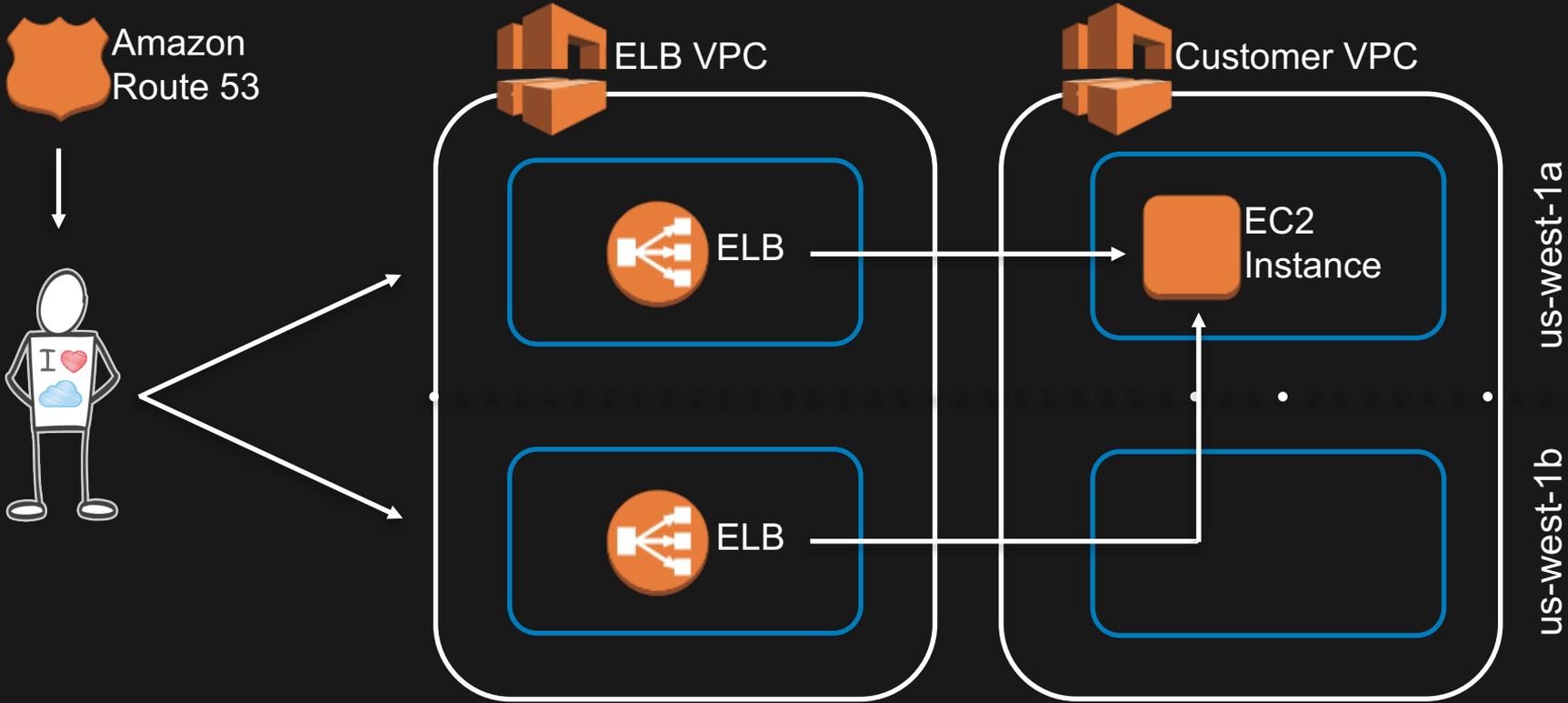


**Application Load Balancer will fail open
should all back-ends fail the health check**



Always use
multiple Availability Zones





1 Available Zone



Risks Availability

2 Available Zones

⋮



100% Extra Capacity

3 Available Zones

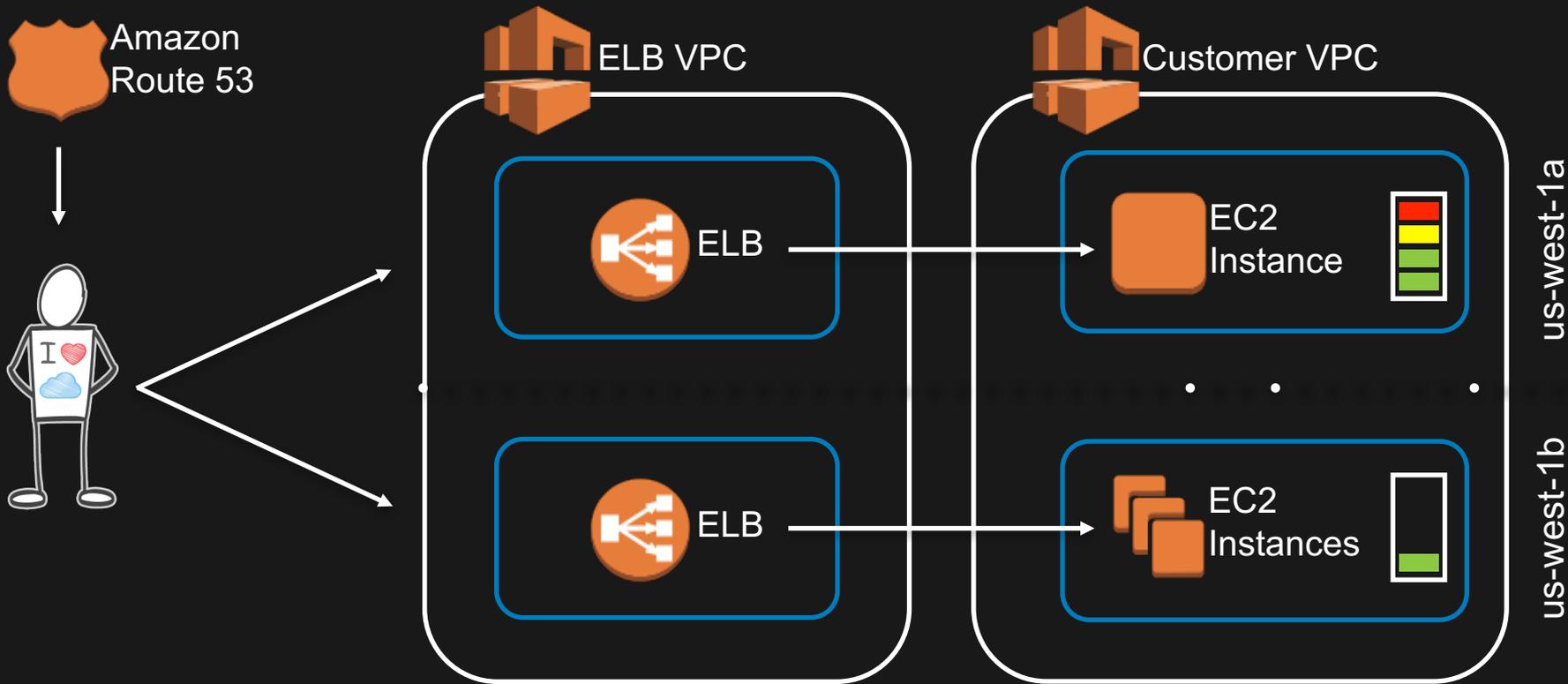
⋮



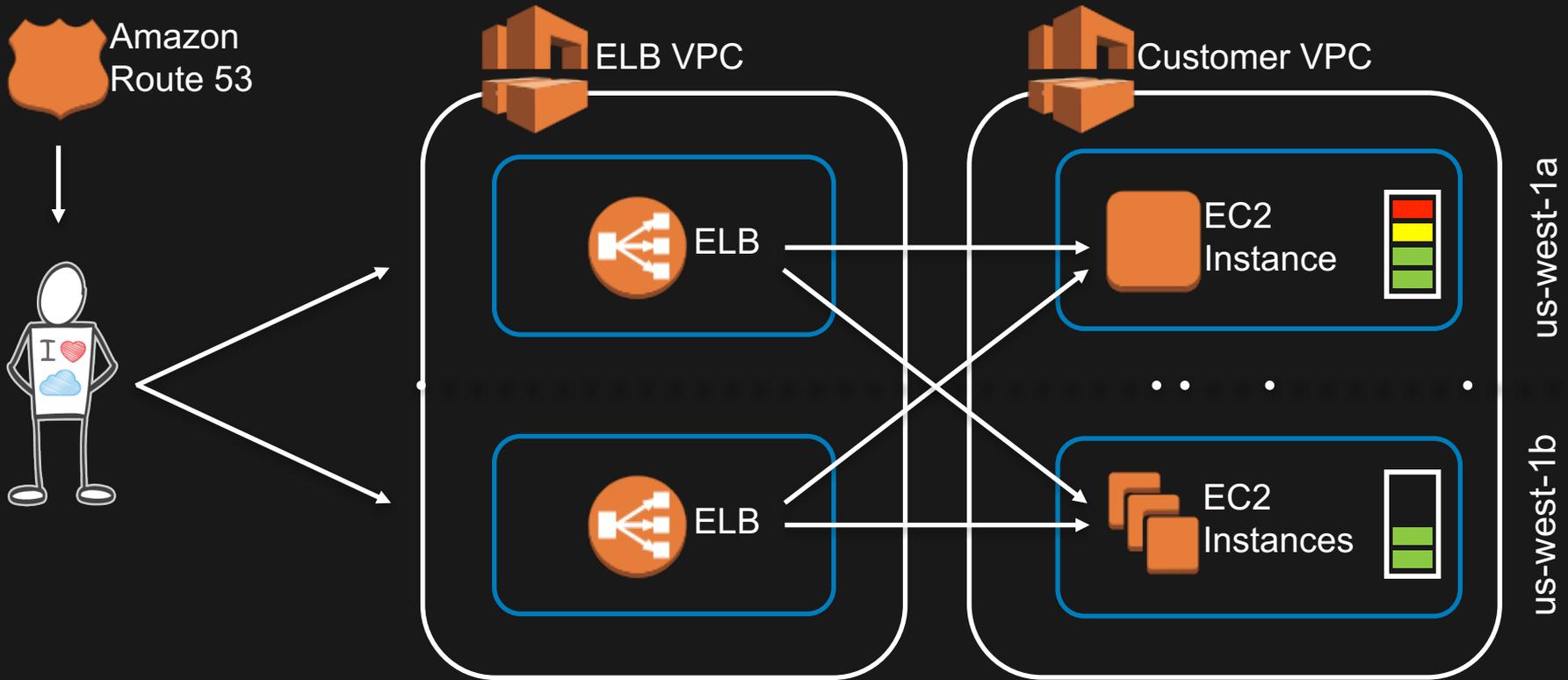
50% Extra Capacity

Using **multiple** Availability Zones
can bring a few **challenges** ...

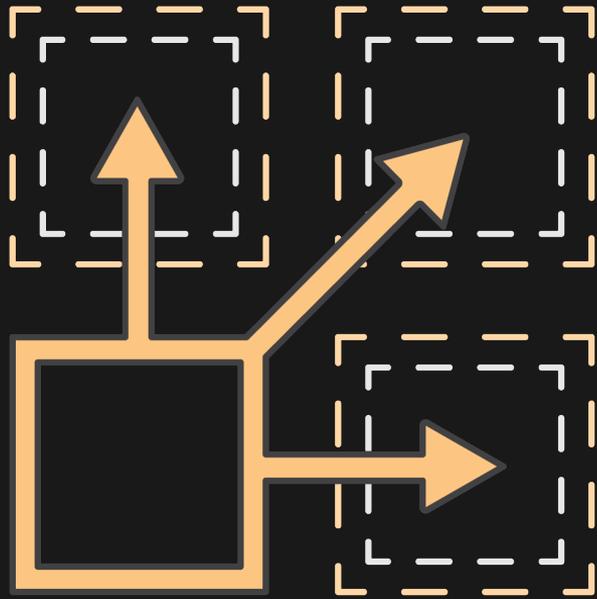
Imbalanced Instance Capacity



Cross-Zone Load Balancing



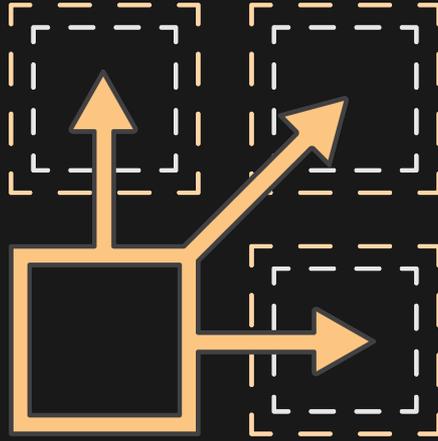
Cross-Zone Load Balancing



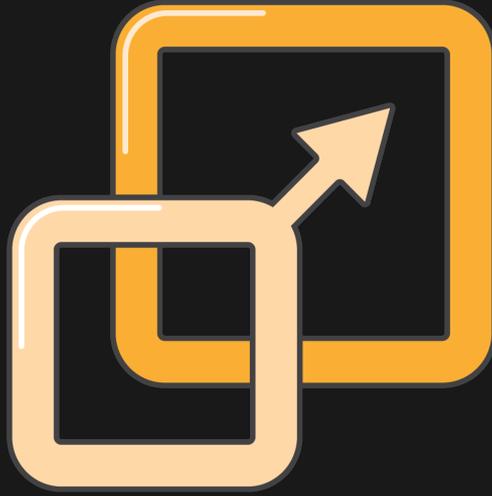
Distributes requests **evenly across multiple Availability Zones**

Absorbs impact of **DNS caching** and eliminates **imbalances in backend instance utilization**

No additional bandwidth charge for cross-zone traffic



Cross Zone Load Balancing **enabled by default on all Application Load Balancers**



Auto Scaling now supports the scaling of applications at the **target group level**



example.com



ELB

/orders

/images



Application Load Balancer integrates with **Auto Scaling** to manage the scaling of each target group **independently**



**When using Auto Scaling, keep in mind that
your application may be **under load**
during **quiet times****

**Continued support for advanced
application **security** features**

SSL Offloading



SSL Negotiation Policies provide selection of ciphers and protocols that adhere to the **latest industry best practices**

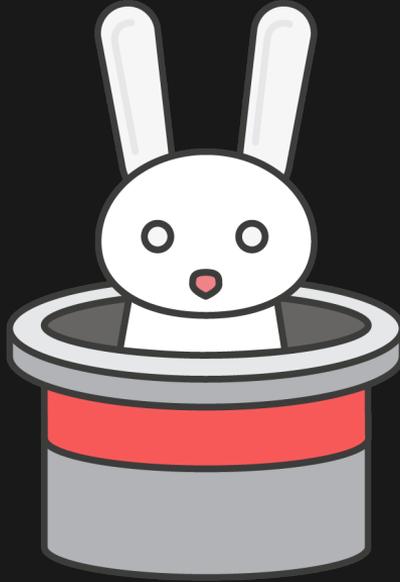
Optimized for balance between security and client connectivity, as tested with **Amazon.com** traffic

ACM Integration



Application Load Balancer supports **security groups** to limit access to specified ranges

New



Web Application Firewall now supports
Application Load Balancers

Website Application Firewall *New*

Monitors requests and **protects web applications from malicious activities** at the load balancer level

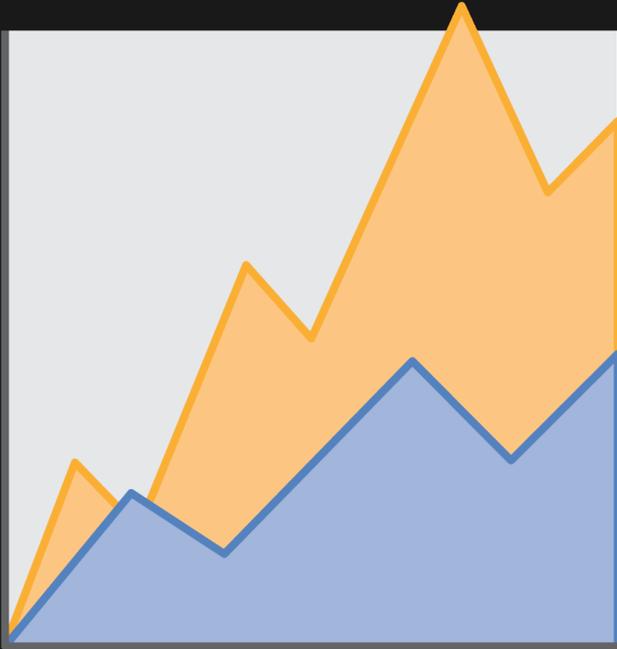
Block, allow, or count web requests based on WAF rules and conditions

Preconfigured rules available for common protections: SQL-injection, cross-site scripting, bad-actor IPs, bad bots, and HTTP flood attacks



**Improved load balancer and
application monitoring**

Amazon CloudWatch Metrics



CloudWatch metrics provided for each load balancer

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

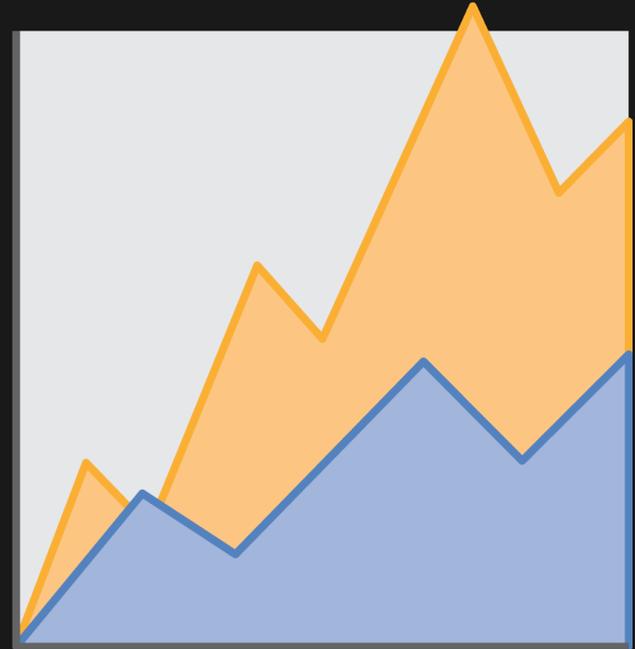
All metrics provided at 1-minute granularity

Amazon CloudWatch Metrics

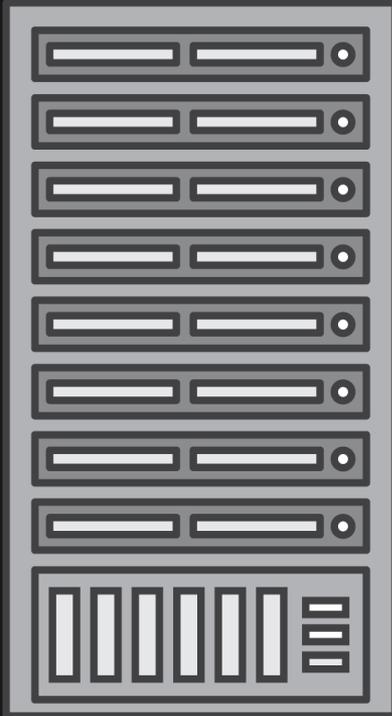
Metrics provided at both the **load balancer** and **target group level**

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

Auto Scaling can use these metrics for scaling of the back-end fleet



HealthyHostCount



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 backend instance from another EC2 instance

View at the zonal dimension

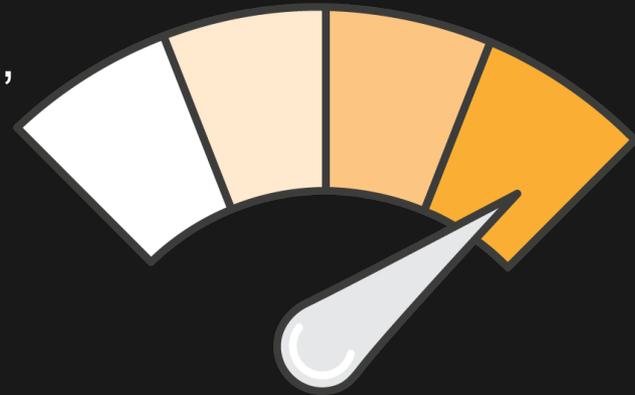
Latency

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 backend instance from another instance

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

Debug individual requests using **access logs**



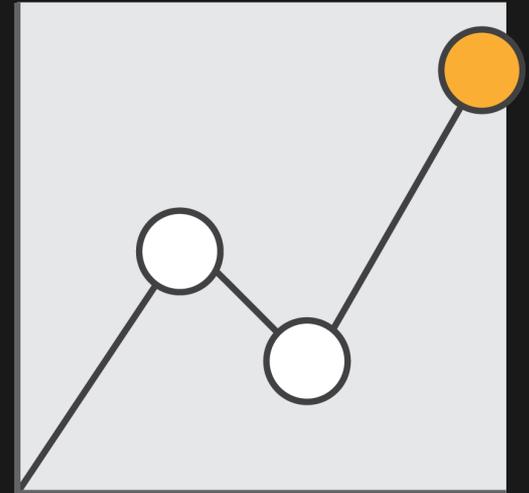
Rejected Connection Count

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 surge queue metrics which are used by the Classic Load Balancer

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

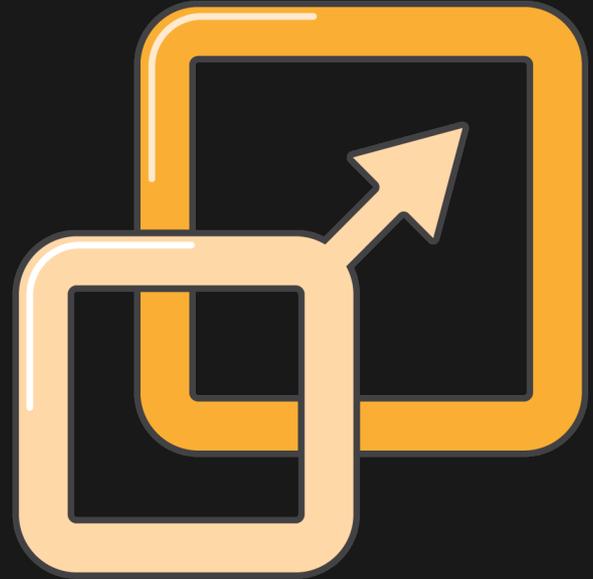
Normally a sign of an under-scaled application



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

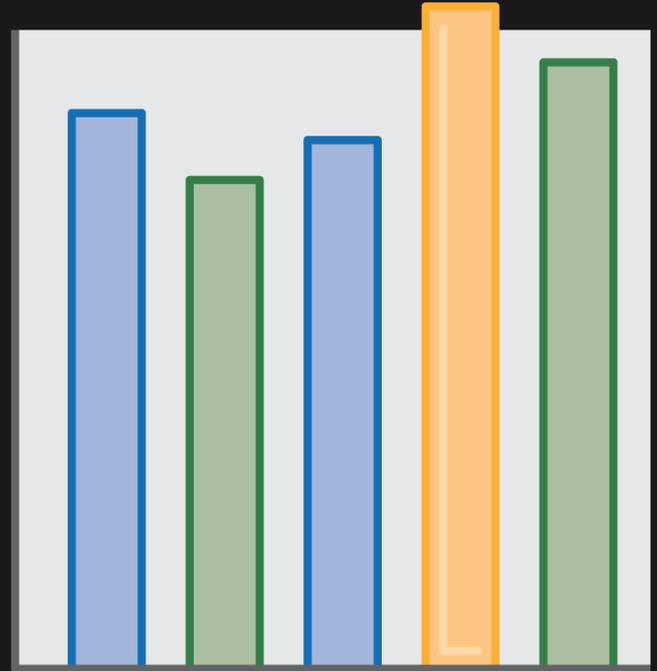


CloudWatch Percentiles

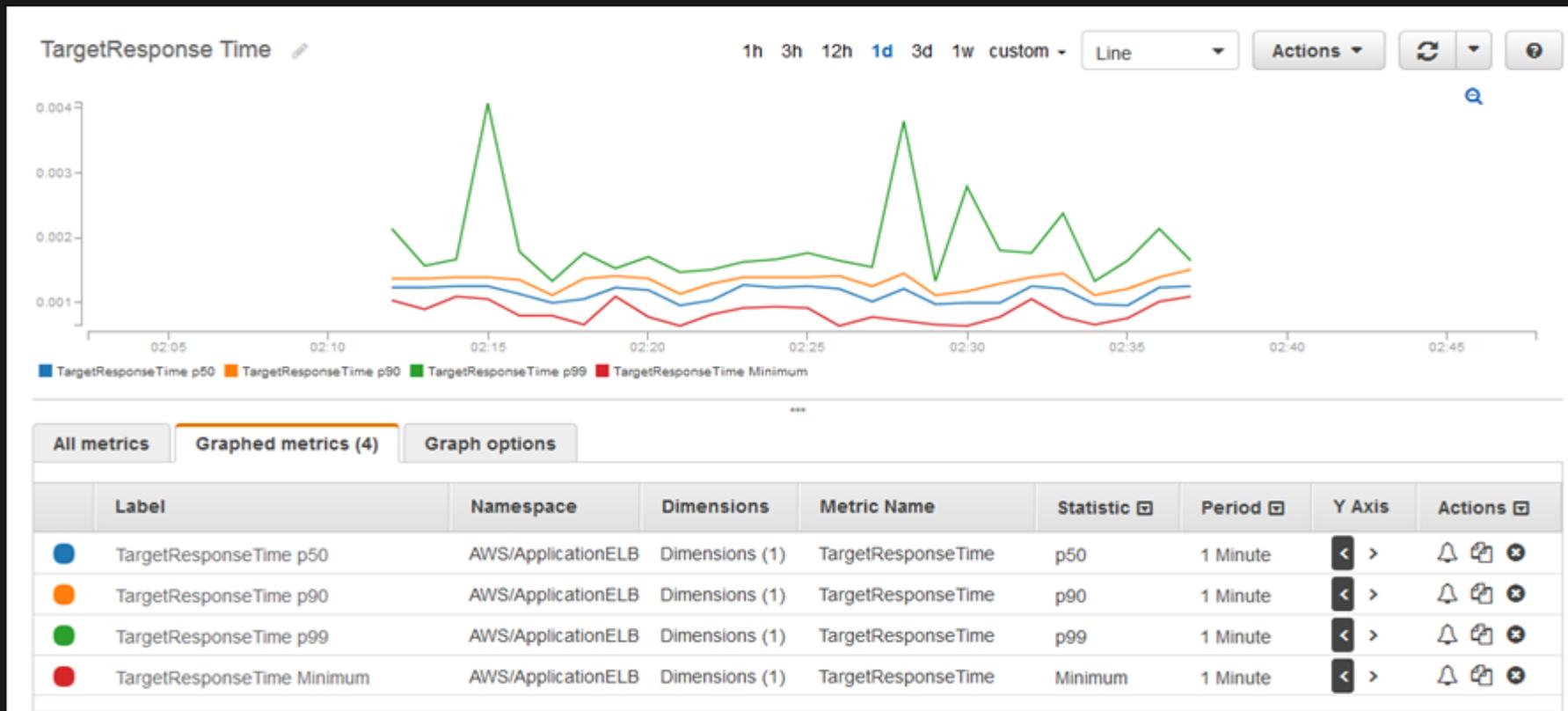
Load balancer **request response times** are now provided with percentile dimensions

Provides visibility into the **90th, 95th, 99th, or 99.9th percentile** of response times

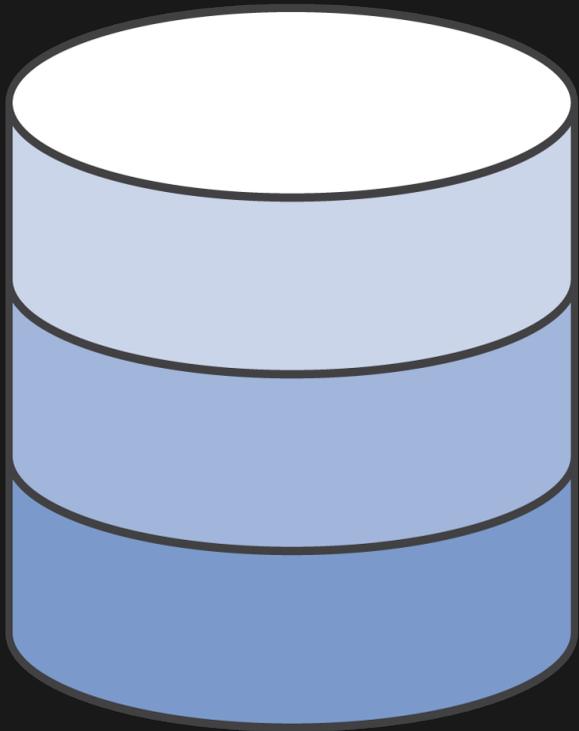
Allows for more **meaningful, and aggressive, performance targets** for applications



CloudWatch Percentiles



Access Logs



Provide detailed information on **each request processed by the load balancer**

Includes **request time, client IP address, latencies, request path, server responses, ciphers and protocols, and user-agents**

Delivered to an **Amazon S3 bucket every 5 or 60 minutes**

Request Tracing

Application Load Balancers insert a **unique trace identifier** into each request using a custom header: **X-Amzn-Trace-ID**

Trace identifiers are **preserved** through the request chain to allow for request tracing

Trace identifiers are included in **access logs** and can also be logged by applications themselves





**When should I use
Application Load Balancer?**

	Classic	Application
Protocol	TCP, SSL, HTTP, HTTPS	HTTP, HTTPS
Platforms	EC2-Classic, EC2-VPC	EC2-VPC
Health checks	✓	Improved
CloudWatch metrics	✓	Improved
Path-based routing		✓
Container support		✓
WebSockets & HTTP/2		✓

For **TCP/SSL** or **EC2-Classic**,
use **Classic Load Balancer**

For all other use-cases,
use **Application Load Balancer**



**Remember to complete
your evaluations!**

AWS

S U M M I T

Thank you!

