AWS re:Invent

NET407-R

Get the most from Elastic Load Balancing for different workloads

Narayan Subramaniam
Senior Manager
Amazon Web Services

Jon Zobrist (zob)
System Dev Manager
Amazon Web Services

AWS re:Invent

© 2019, Amazon Web Services, Inc. or its affiliates. All rights reserved.

aws

Related breakout

NET413 – Elastic Load Balancing: Best practices for securing your applications

Monday, Dec 2, 6:15 PM - 7:15 PM – Aria, Level 3 West, Starvine 2

Wednesday, Dec 4, 2:30 PM - 3:30 PM – MGM, Level 3, North Concourse 306

Agenda

Different types of ELB (CLB, ALB, NLB)

For each ELB, we'll cover

- Client connectivity
- Request routing
- Target types

Health check configuration

Monitoring your ELB workload

Elastic

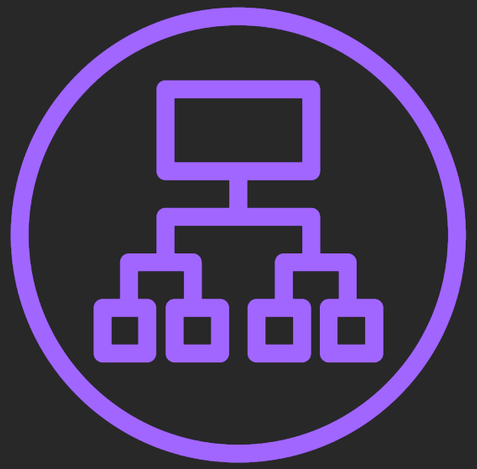
Secure

Integrated

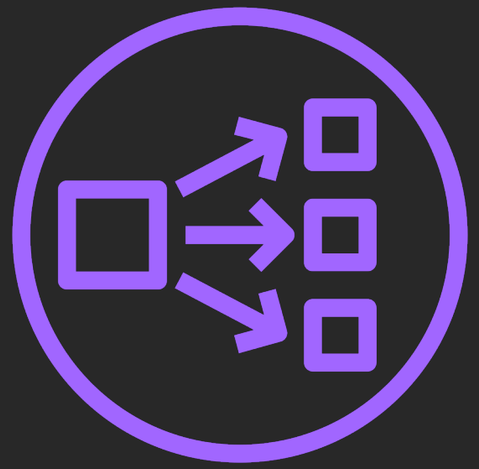
Cost effective

The Elastic Load Balancing family

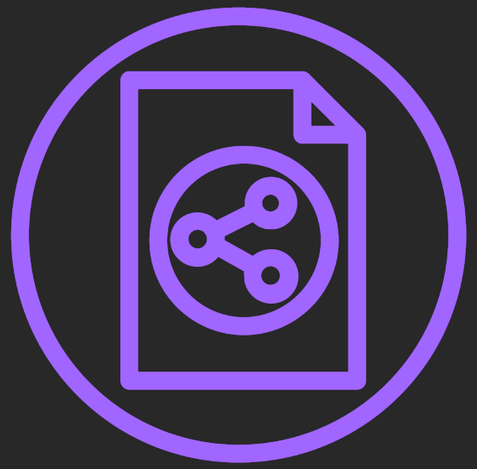
Application



Network



Classic



Listeners and connectivity

ELB comparison	ALB	NLB	CLB
Listener protocols	HTTP, HTTPS,HTTP/2	TCP, UDP, TLS	TCP, SSL, HTTP, HTTPS
TLS/SSL termination	✓	✓	✓
IAM Certificates	✓	✓	✓
ACM Certificates (with automated rotation)	✓	✓	✓
Websockets (HTTP upgrade)	✓		
Static IP (Use your own Elastic IP)		✓	
Preserve client IP		✓	
X-forwarded HTTP headers	✓		✓
Proxy protocol on TCP connections		✓	✓
VPC IPv6	✓	✓	
AWS PrivateLink		✓	

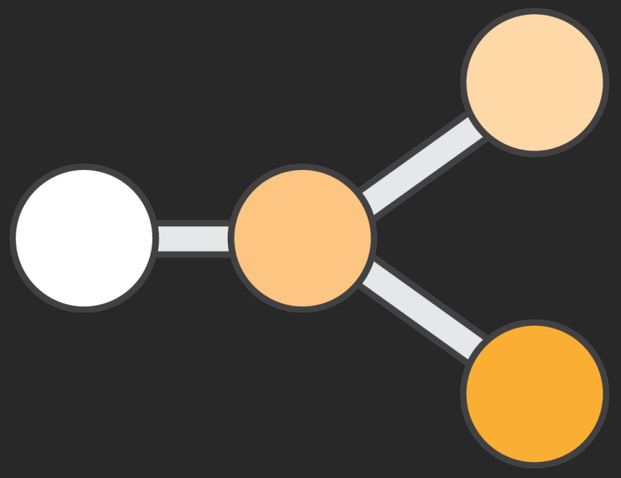
Request routing

ELB comparison	ALB	NLB	CLB
HTTP request load balancing	✓		✓
TCP flow/connection load balancing		✓	✓
UDP flow load balancing		✓	
HTTP host header	✓		
HTTP method	✓		
Source IP address or CIDR	✓		
Arbitrary HTTP header	✓		
HTTP query string parameter	✓		
Combined rules	✓		
AWS WAF	✓		
Weighted Target Group routing	✓		

Targets and target groups

ELB comparison	ALB	NLB	CLB
EC2 instances	✓	✓	✓
IP addresses	✓	✓	
Per-target ports	✓	✓	
Shared-target ports			✓
HTTP request logging	✓		✓
TCP connection logging		✓(TLS only)	✓
Containers (ECS, EKS, Kubernetes, Docker)	✓	✓	
Security groups	✓		✓
HTTP(S) health checks	✓	✓	✓
TCP health checks	✓	✓	✓

Connectivity



Health checks

Monitoring

Client connectivity

Client to Elastic Load Balancer connections

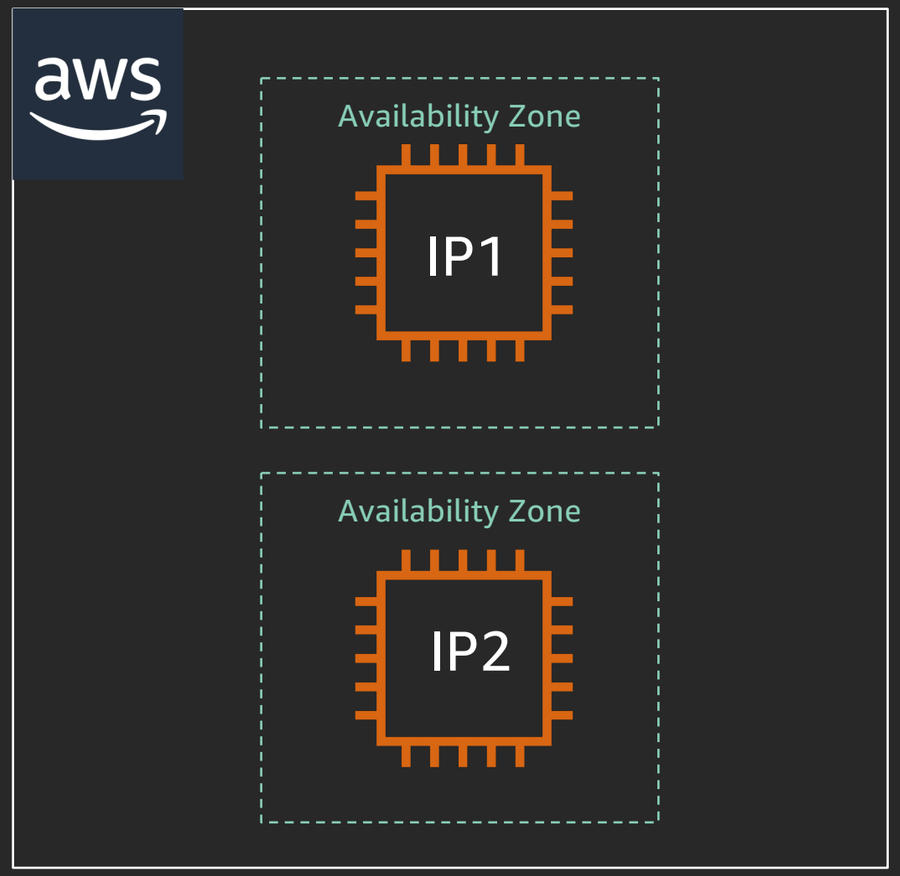
Client connectivity

Route 53 → ELB

Amazon Route 53

Users

*Route 53 aliases



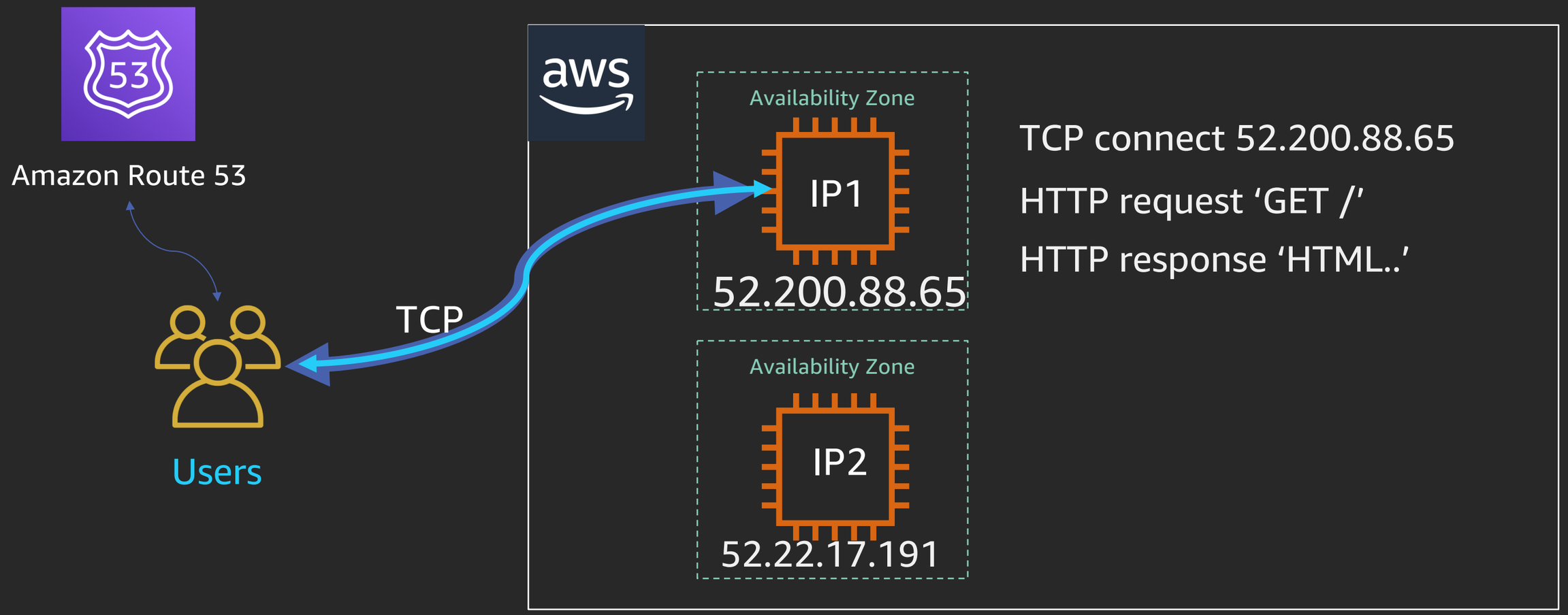
Client connectivity

Route 53 → ELB

*Route 53 aliases

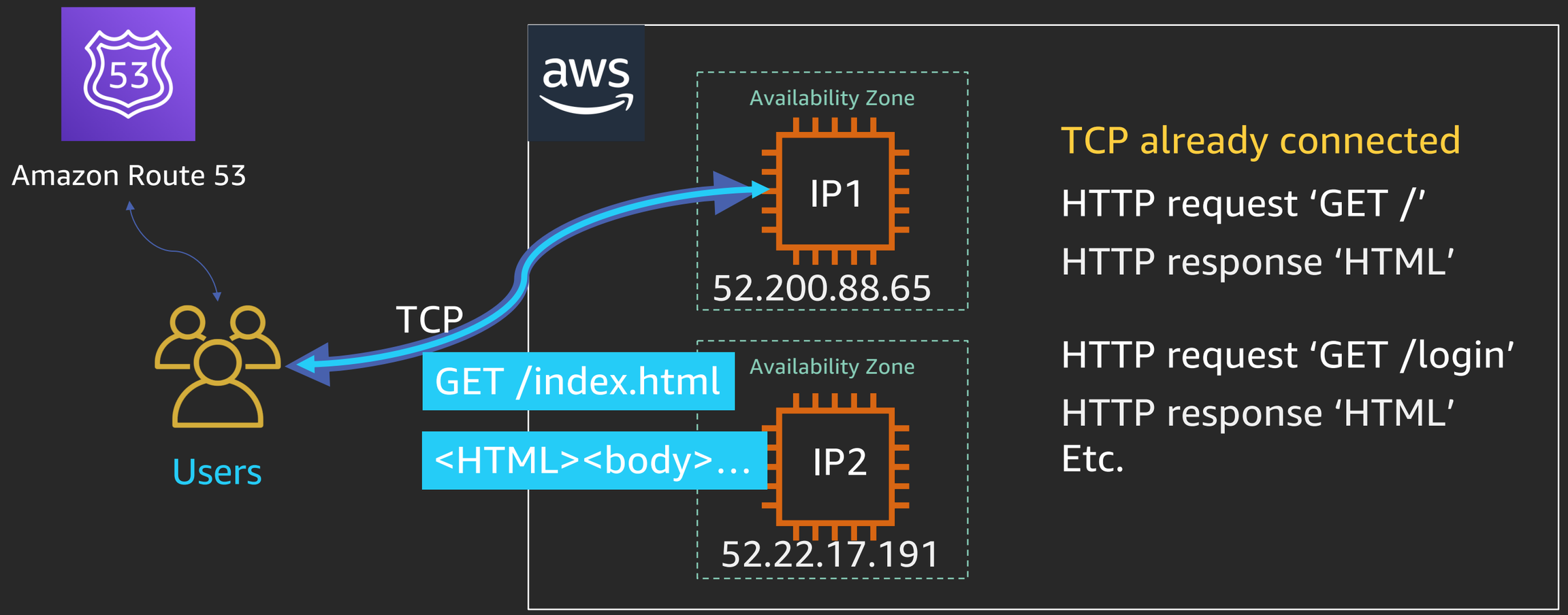
Client connectivity

Route 53 → ELB



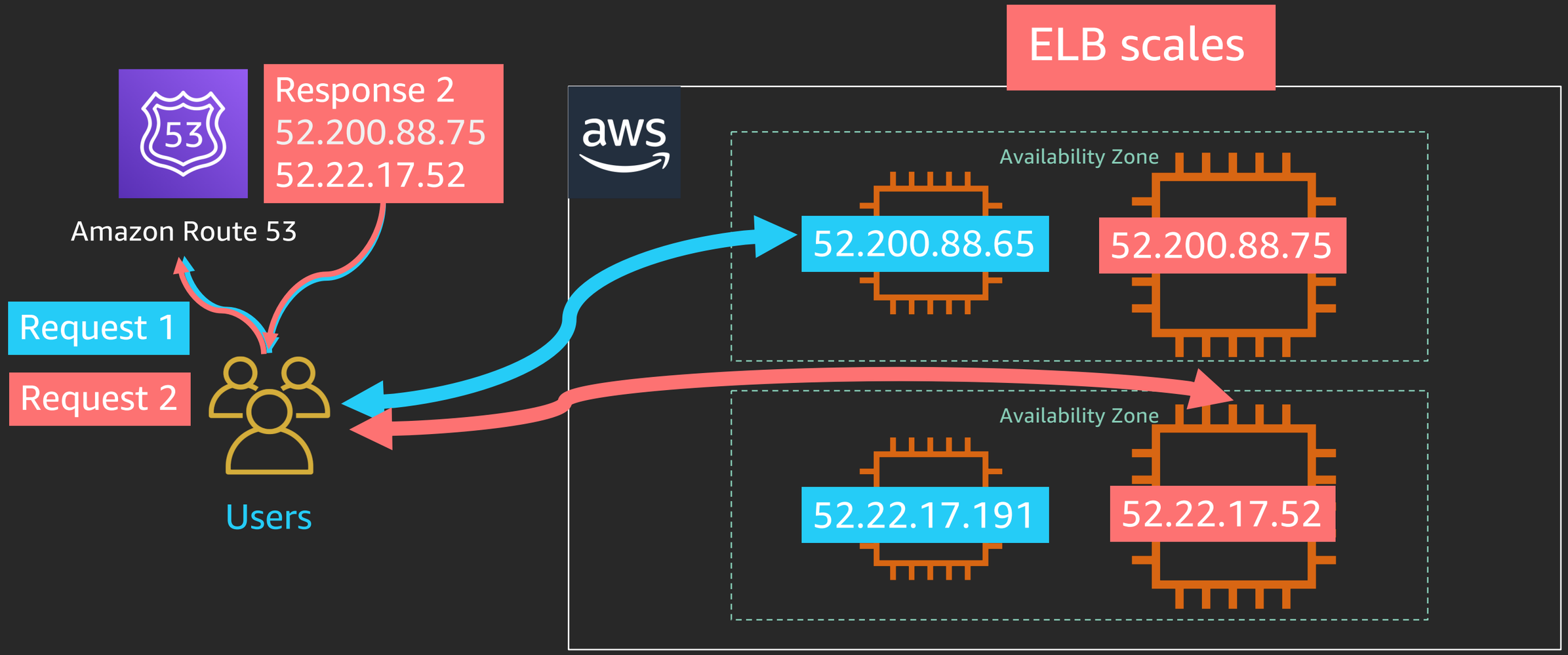
Client connectivity

Route 53 → ELB



Client connectivity

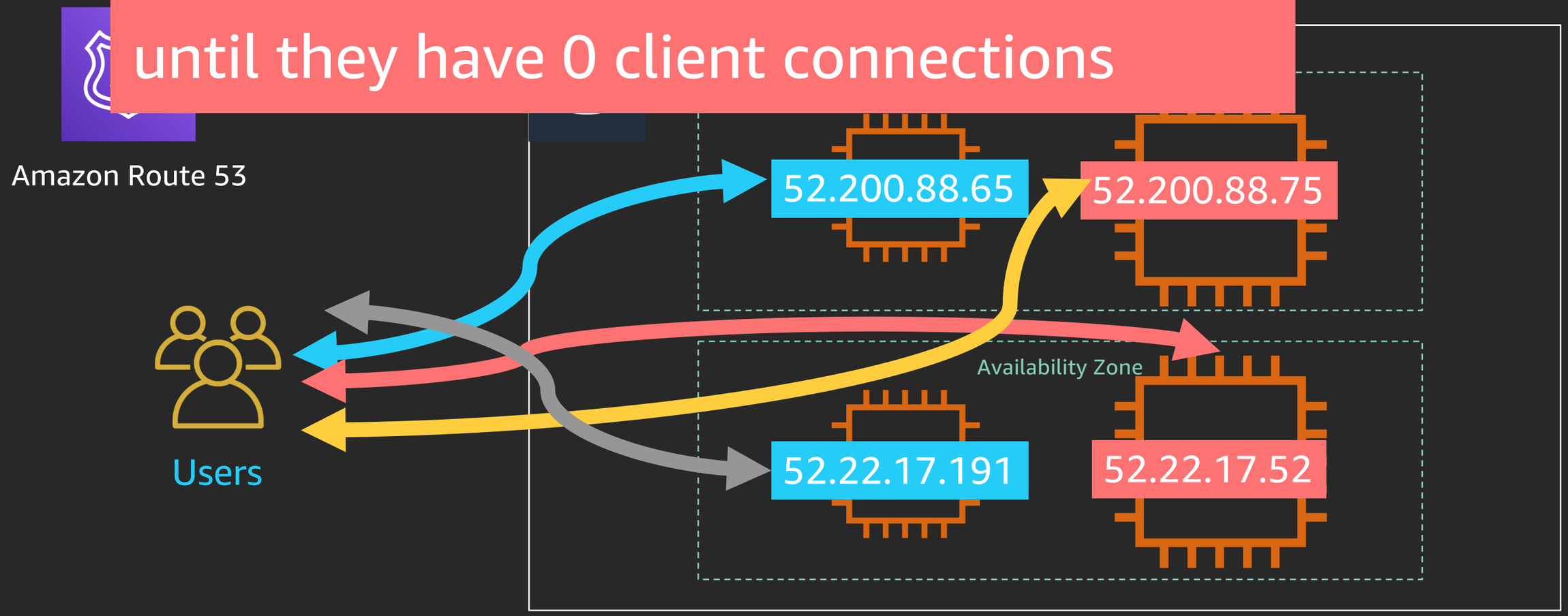
Route 53 → ELB



Client connectivity

Route 53 → ELB

Previous ELB IPs continue to be available until they have 0 client connections



Client connection best practices

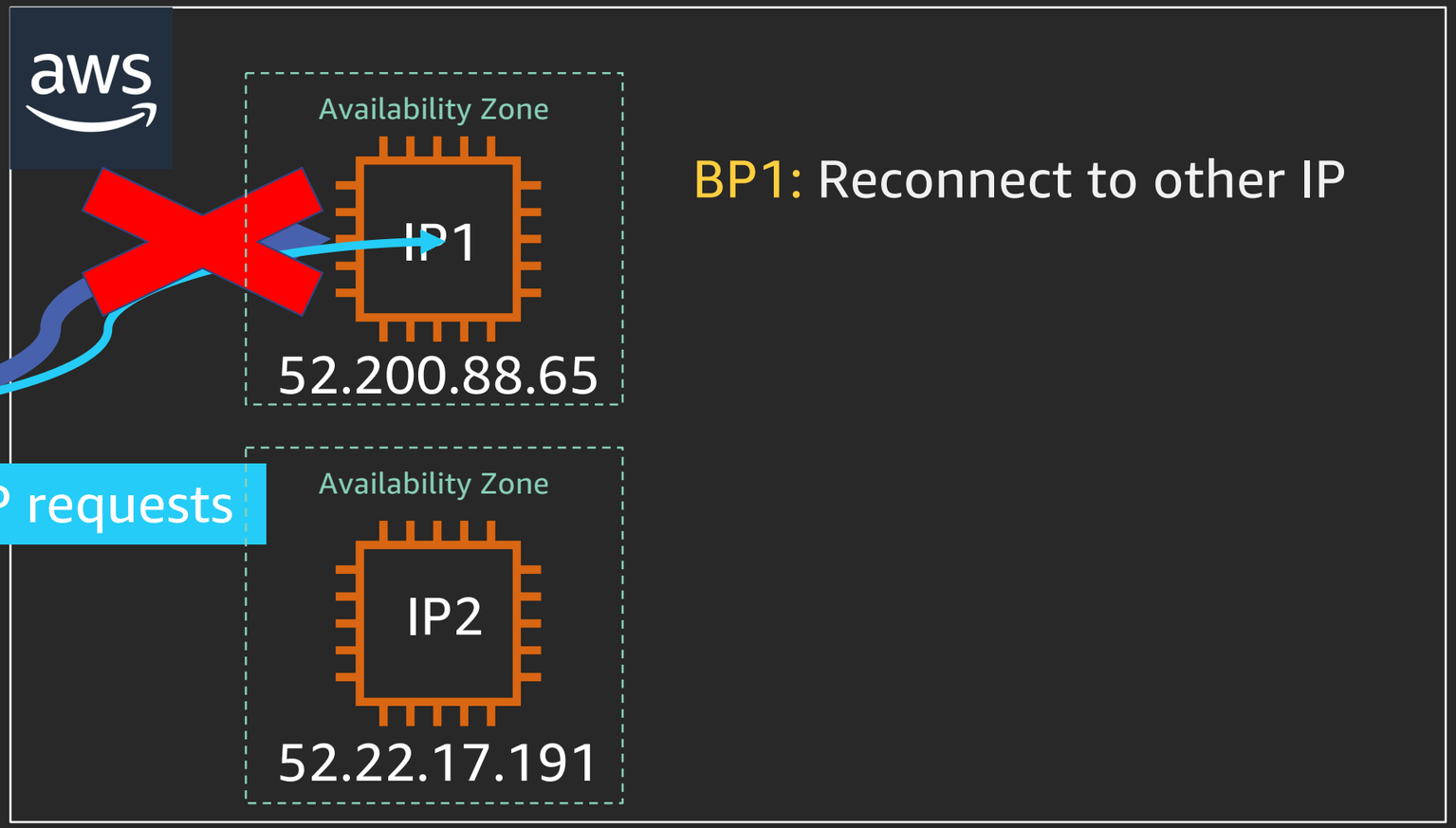
Route 53 → ELB

Amazon Route 53

1. Establish TCP connection
2. Send HTTP request
3. Send more

ers

HTTP requests



BP1: Reconnect to other IP

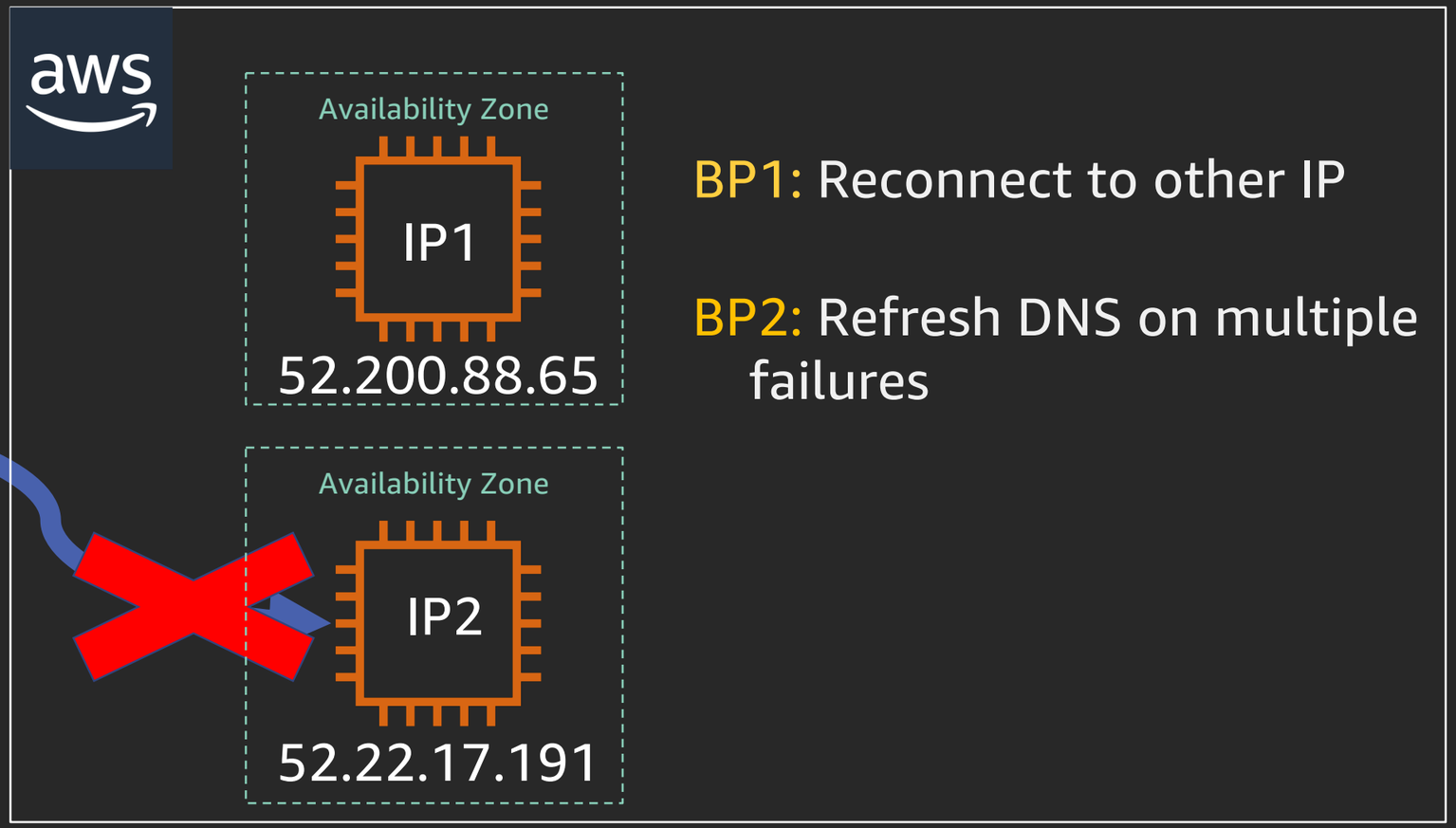
Client connection best practices

Route 53 → ELB

Amazon Route 53

1. Establish TCP connection
2. Send HTTP request

Users

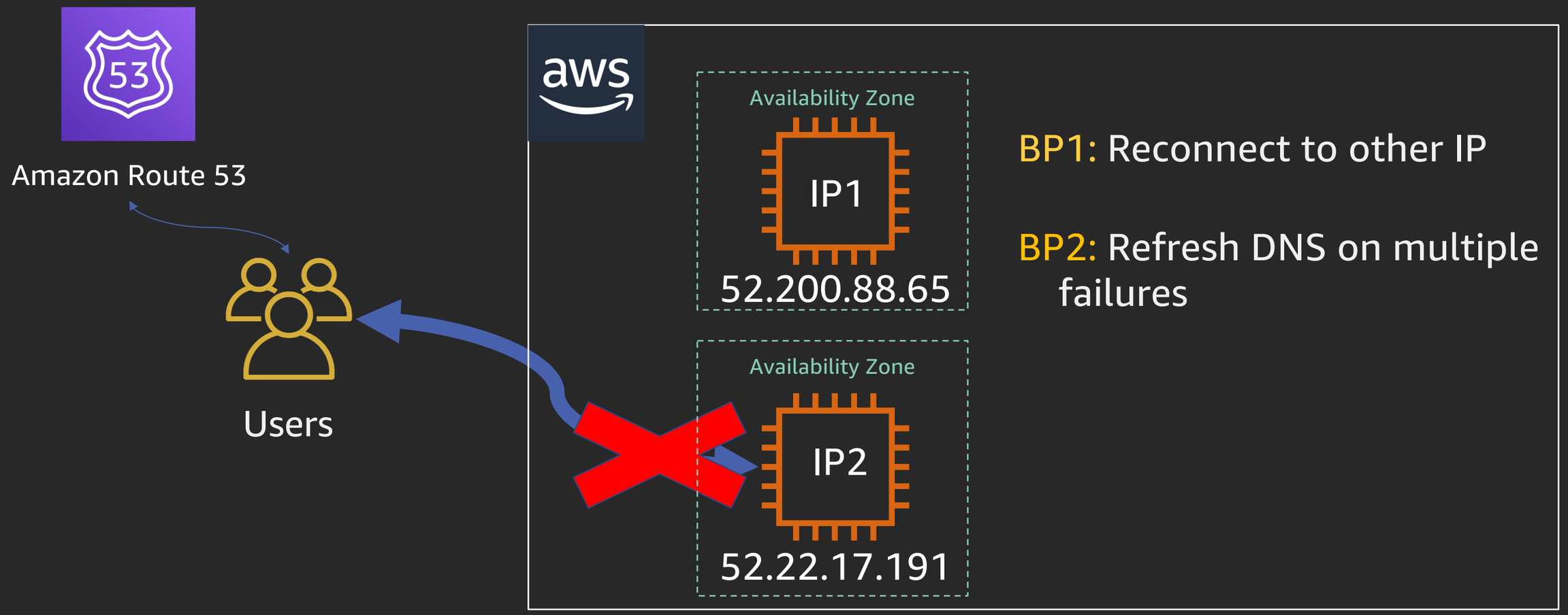


BP1: Reconnect to other IP

BP2: Refresh DNS on multiple failures

Client connection best practices

Route 53 → ELB

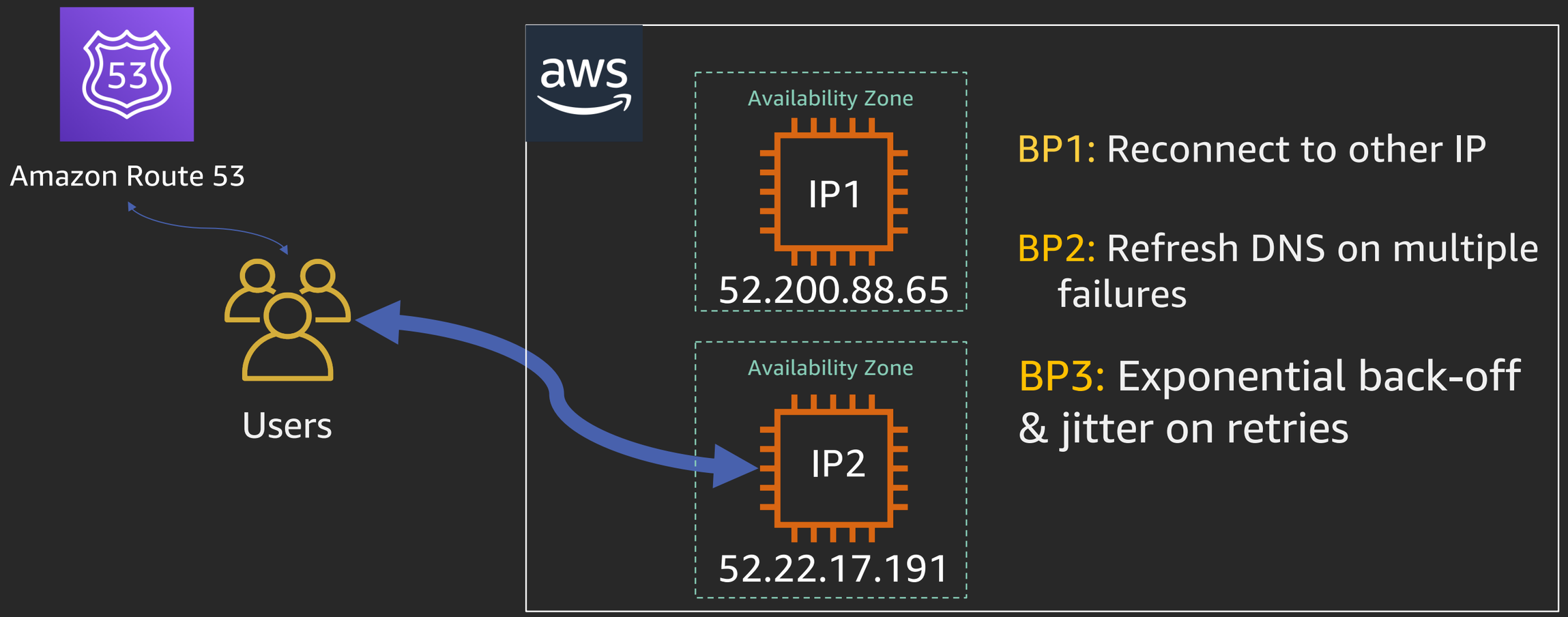


BP1: Reconnect to other IP

BP2: Refresh DNS on multiple failures

Client connection best practices

Route 53 → ELB



BP1: Reconnect to other IP

BP2: Refresh DNS on multiple failures

BP3: Exponential back-off & jitter on retries

sleep(2^retry_count * 100ms) + (rand(1, 10) * 100ms)

Client connection summary

Use DNS

DNS rocks!

Re-resolve DNS before reconnecting after failures

Respect Time To Live (TTL), 60 seconds for ELBs

Use exponential back-off on reconnect

sleep(2^retry_count * 100ms) + (rand(1, 10) * 100ms)

Add jitter to reconnect

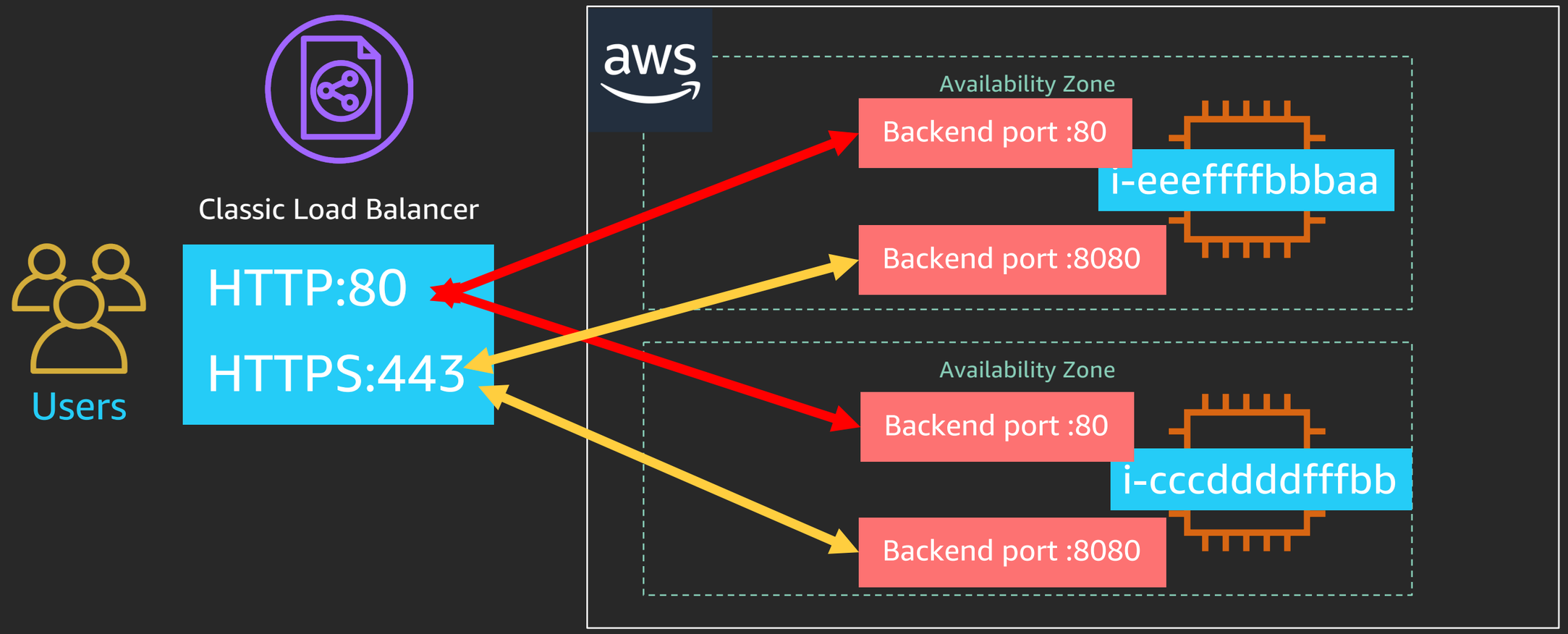
Related blog post – Error Exponential Backoff and Jitter, AWS Architecture Blog
https://aws.amazon.com/blogs/architecture/exponential-backoff-and-jitter/

Client connectivity and request routing

Classic Load Balancer request routing

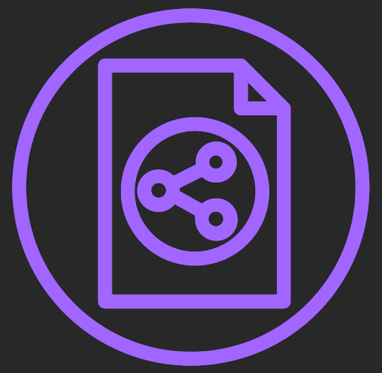
Client connectivity and request routing

Route 53 → ELB (listeners) → Backend target



Client connectivity and request routing

Route 53 → ELB (listeners) → Backend target



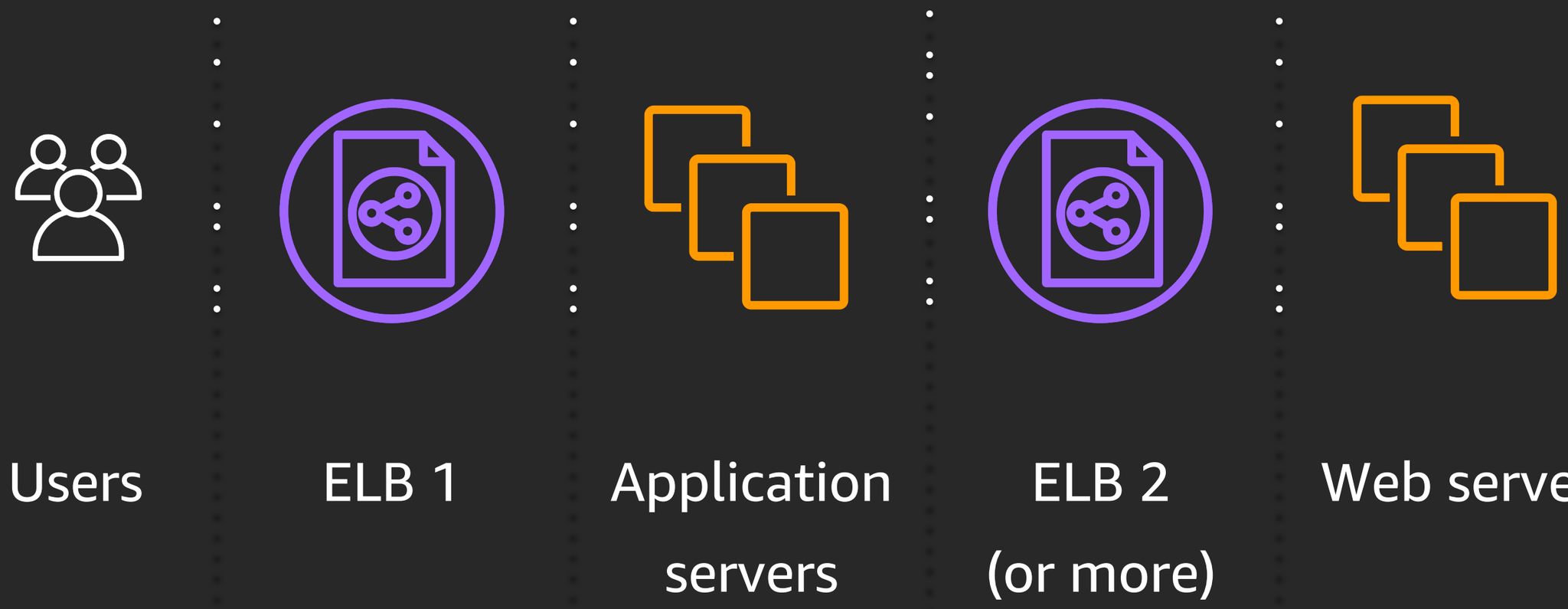
Classic Load Balancer

HTTP/HTTPS	Routes per HTTP request Least outstanding request routing
TCP/TLS (SSL)	Routes per TCP connect Round robin connection routing

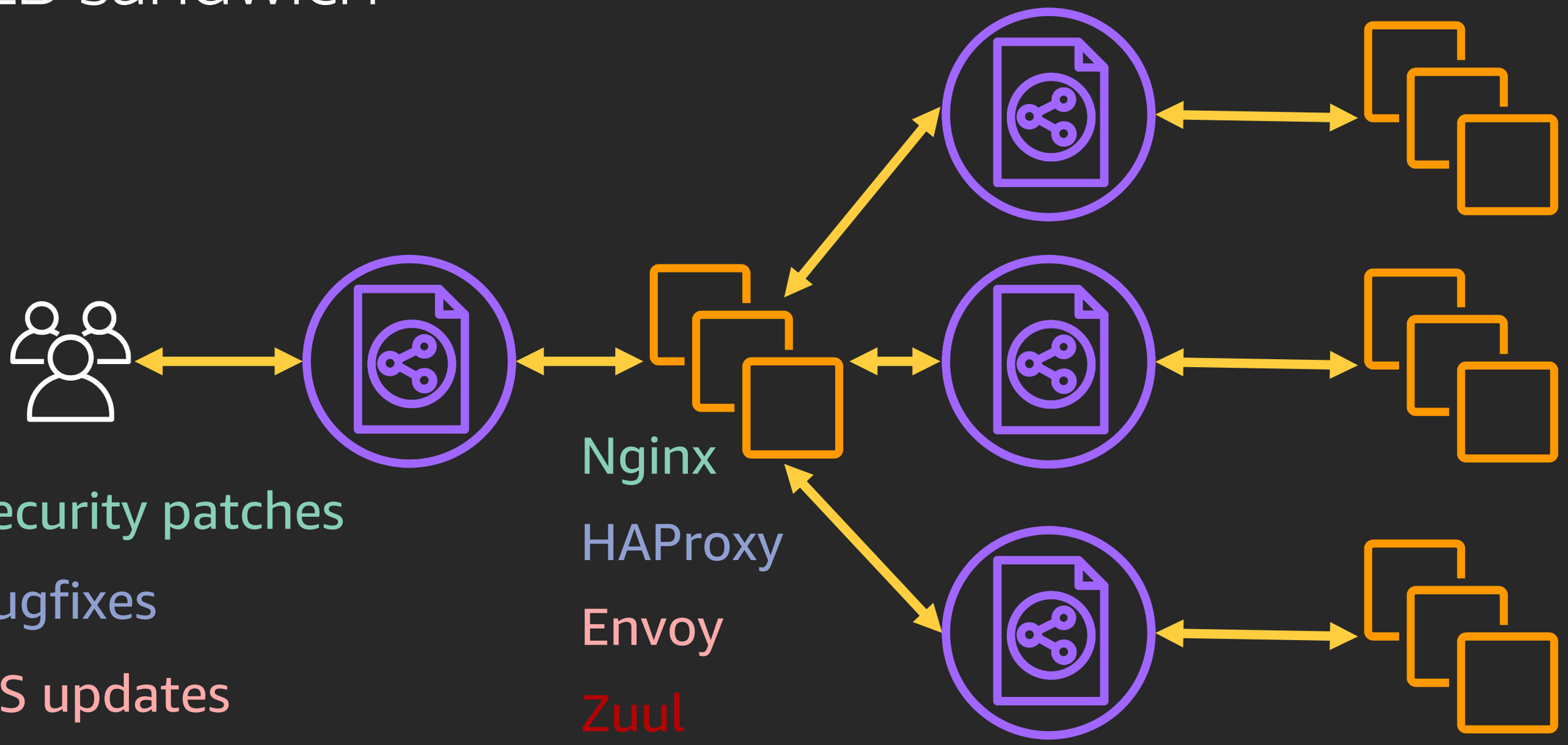
Client connectivity and request routing

Application Load Balancer request routing, target groups & rules

ELB sandwich



ELB sandwich



Security patches

Bugfixes

OS updates

Configuration changes

Scaling

Nginx

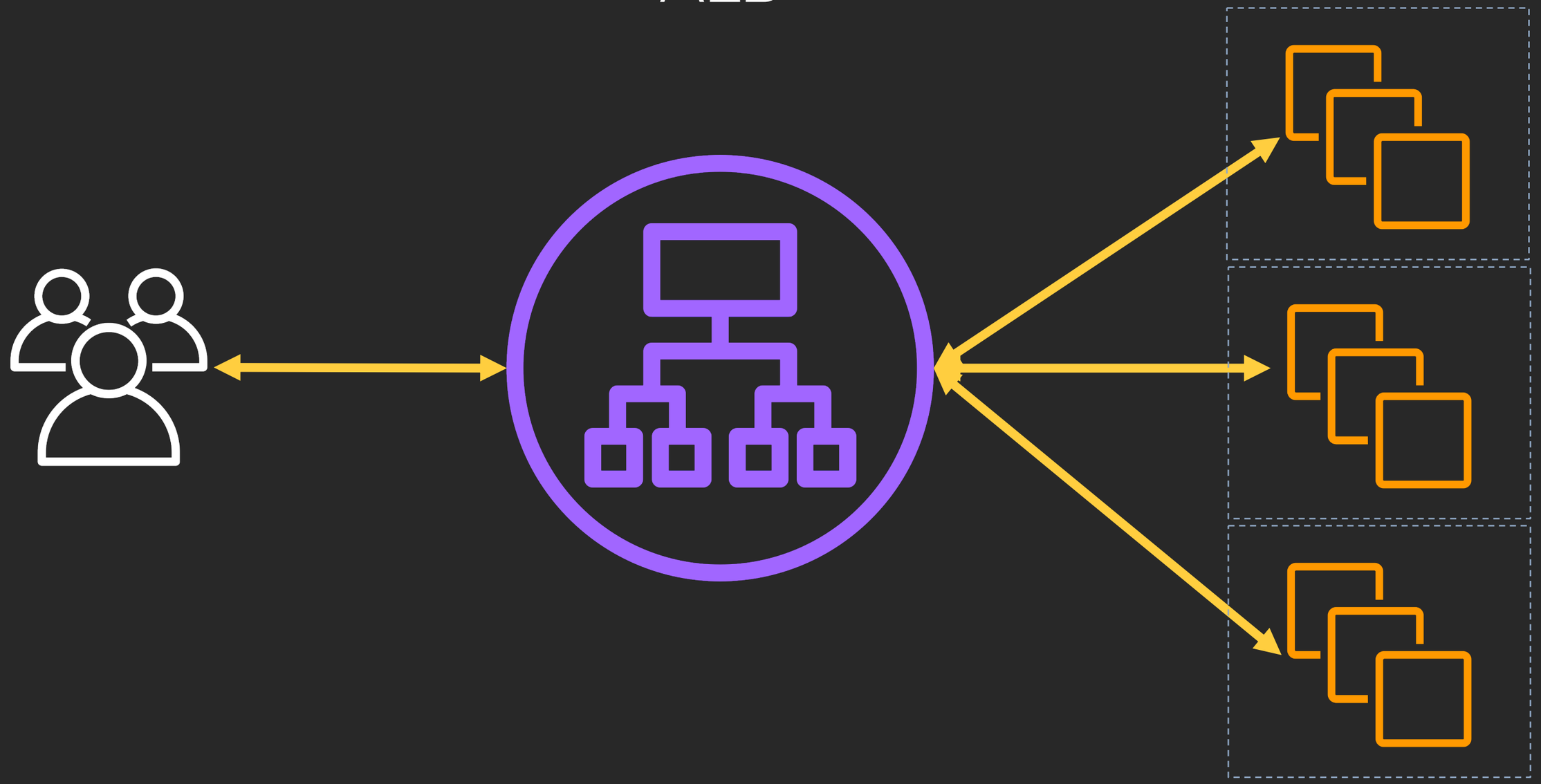
HAProxy

Envoy

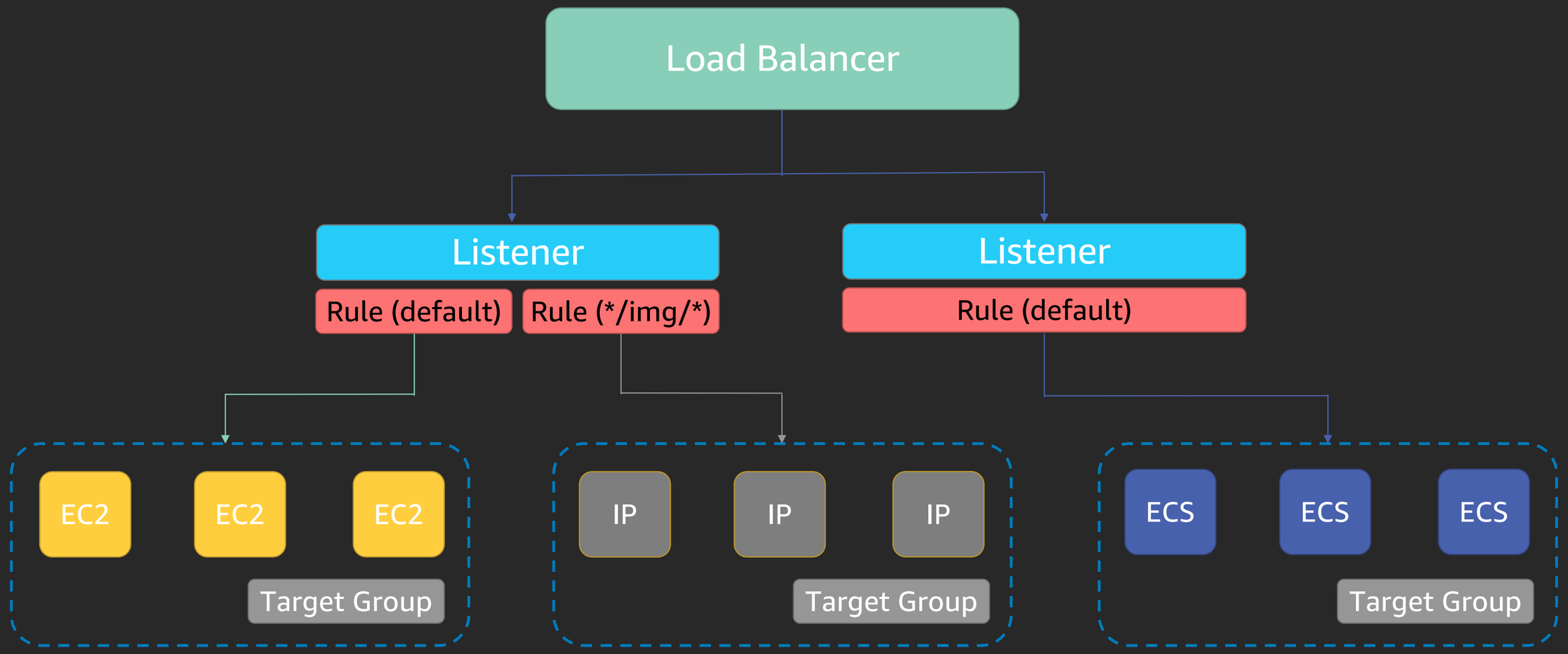
Zuul

Apache

ALB

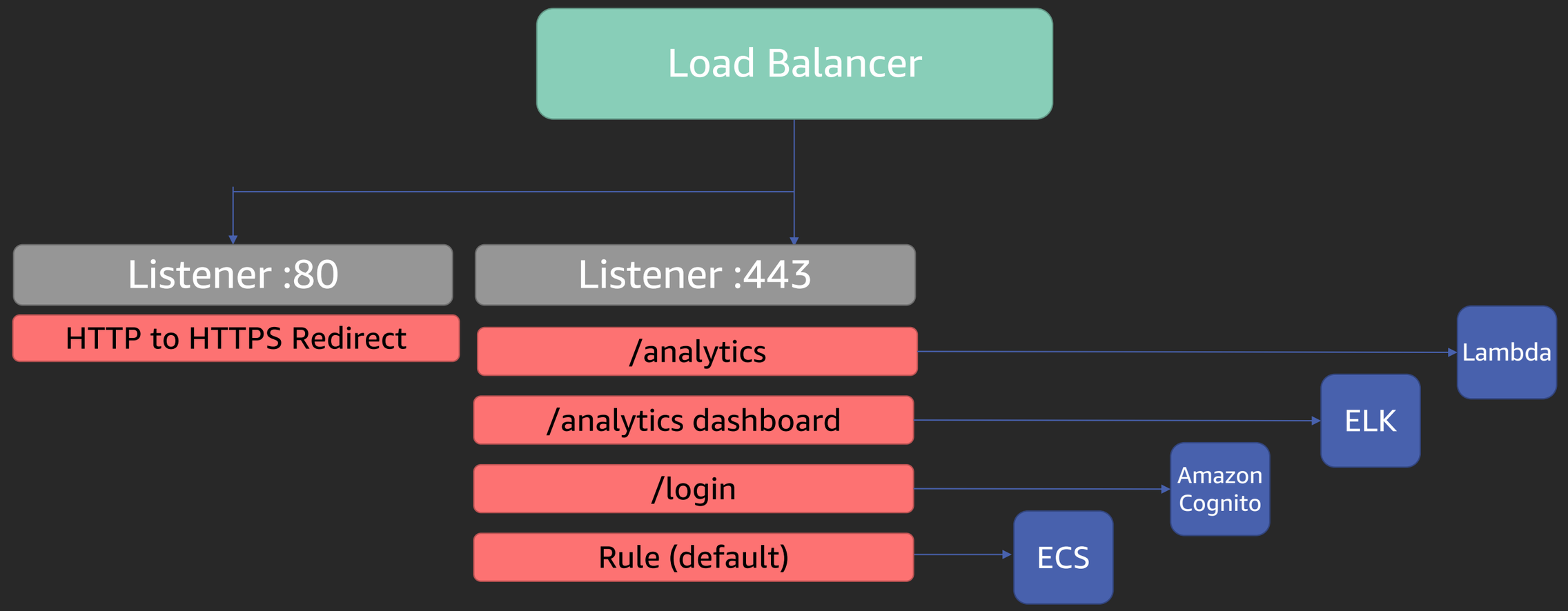


Logical overview of an ALB



Client connectivity and request routing

Route 53 → ELB → Target Group (rules) → Target

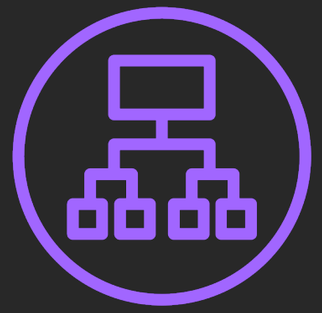


Client connectivity and request routing

Route 53 → ELB → Target Group (rules) → Target

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

Up to 100 rules



Application Load Balancer

Host header

Host: www.example.com

Path

/webapp

HTTP headers

X-MyApp-Source: 'Cloudfront'

HTTP methods

GET, POST

Query parameters and query string

?Stage=Production

Source IP addresses

99.87.0.0/17

Conditions and actions in a rule

5 matches (evaluations) per rule

5 wildcards per rule

100 rules per load balancer

You can combine existing rules

Condition (if)

Host header

URI path

Arbitrary header

URI query string

HTTP method

Match 1 | OR | Match 2 | OR | Match 3

AND

Next condition

Action (then)

Forward

Redirect

Fixed response

Authenticate

Client connectivity and request routing

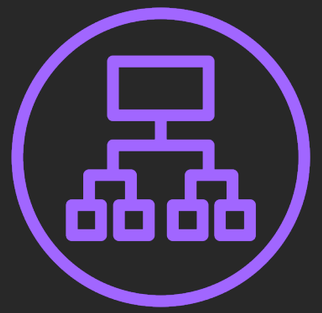
Route 53 → ELB → Target Group (rules) → Target

Method Path Query String/Query parameters

```
GET /hello.htm?version=A HTTP/1.1
Host: www.exampleloadbalancer.com
User-Agent: Mozilla/4.0 (compatible; MSIE5.01; Windows NT)
Accept-Language: en-us
Accept-Encoding: gzip, deflate
Connection: Keep-Alive
```

Host Header

Headers

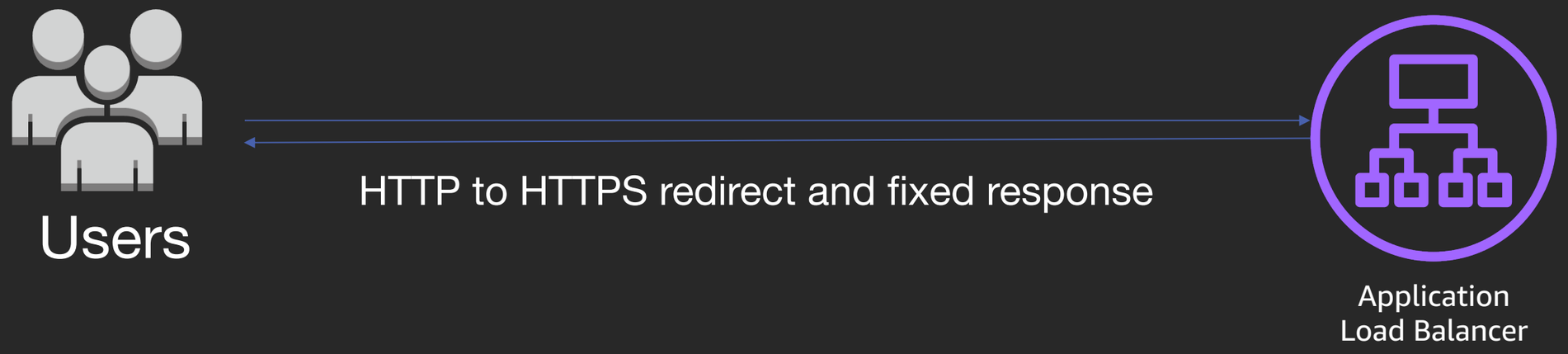


Application Load Balancer

Routing based on any field in an HTTP request header

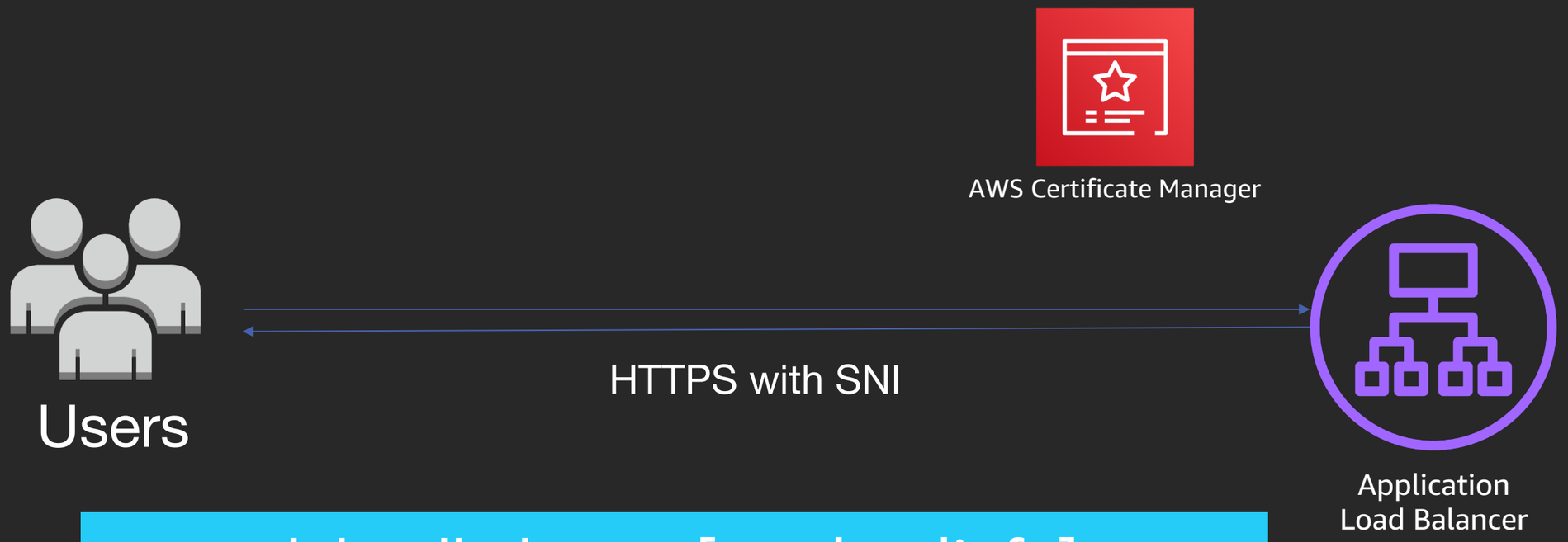
Client connectivity and request routing

Route 53 → ELB → Target Group (rules) → Target



Client connectivity and request routing

Route 53 → ELB → Target Group (rules) → Target



exampleloadbalancer.[com|net|info]

www.exampleloadbalancer.[com|net|info]

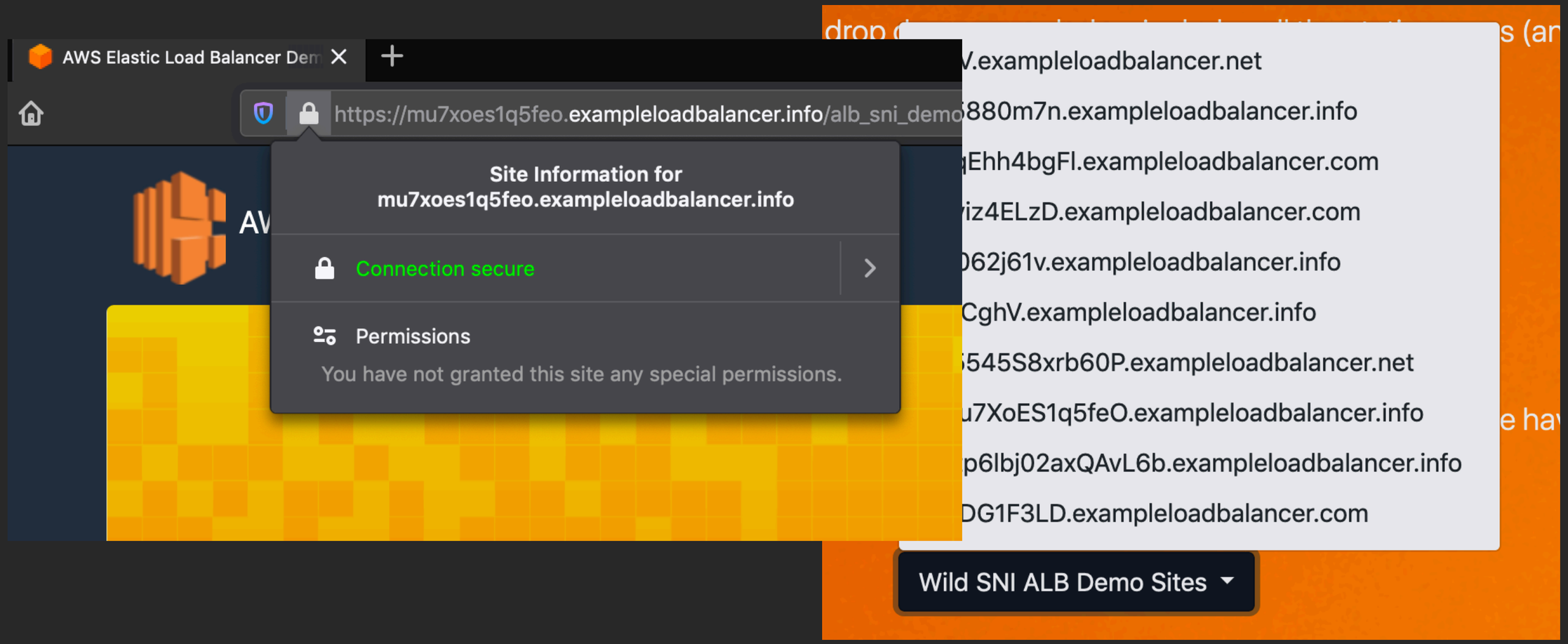
*.exampleloadbalancer.[com|net|info]

*9 certs total for this use case

* Total limit—25 certs

Client connectivity and request routing

Route 53 → ELB → Target Group (rules) → Target



Client connectivity and request routing

Route 53 → ELB → Target Group (rules) → Target

SSL Report: randomstring.exampleloadbalancer.com

Assessed on: Thu, 24 Oct 2019 13:42:12 UTC | Hide | Clear cache

Scan Another >>

	Server	Test time	Grade
1	52.206.66.47 ec2-52-206-66-47.compute-1.amazonaws.com Ready	Thu, 24 Oct 2019 13:36:15 UTC **Duration:** 117.806 sec	A
2	52.22.17.191 ec2-52-22-17-191.compute-1.amazonaws.com Ready	Thu, 24 Oct 2019 13:38:12 UTC **Duration:** 118.682 sec	A
3	52.200.88.65 ec2-52-200-88-65.compute-1.amazonaws.com Ready	Thu, 24 Oct 2019 13:40:11 UTC **Duration:** 121.394 sec	A

SSL Report v1.36.2

SSL Report: www.exampleloadbalan

Assessed on: Thu, 24 Oct 2019 13:39:29 UTC | Hide | Clear cache

Scan Another >

	Server	Test time	Grade
1	52.206.66.47 ec2-52-206-66-47.compute-1.amazonaws.com Ready	Thu, 24 Oct 2019 13:33:32 UTC **Duration:** 118.834 sec	A
2	52.200.88.65 ec2-52-200-88-65.compute-1.amazonaws.com Ready	Thu, 24 Oct 2019 13:35:31 UTC **Duration:** 119.320 sec	A
3	52.22.17.191 ec2-52-22-17-191.compute-1.amazonaws.com Ready	Thu, 24 Oct 2019 13:37:31 UTC	A

SSL Report v1.36.2

SSL Report: www.exampleloadbala

Assessed on: Thu, 24 Oct 2019 13:42:19 UTC | Hide | Clear cache

Scan Another >>

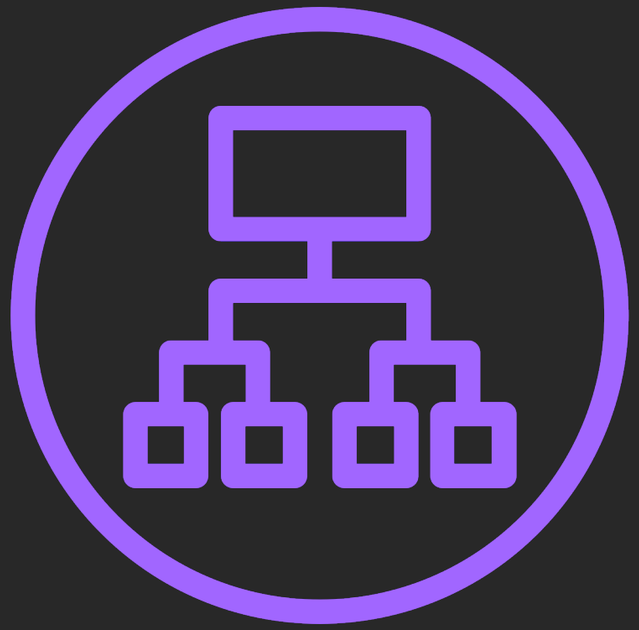
	Server	Test time	Grade
1	52.206.66.47 ec2-52-206-66-47.compute-1.amazonaws.com Ready	Thu, 24 Oct 2019 13:36:22 UTC **Duration:** 118.140 sec	A
2	52.22.17.191 ec2-52-22-17-191.compute-1.amazonaws.com Ready	Thu, 24 Oct 2019 13:38:21 UTC **Duration:** 119.140 sec	A
3	52.200.88.65 ec2-52-200-88-65.compute-1.amazonaws.com Ready	Thu, 24 Oct 2019 13:40:20 UTC **Duration:** 119.377 sec	A

A

- SSL/TLS server assessment service provided in partnership with Qualys SSL Labs (www.ssllabs.com).

New: Weighted Target Groups

New!



ALB

Weighted Target Groups: Use cases

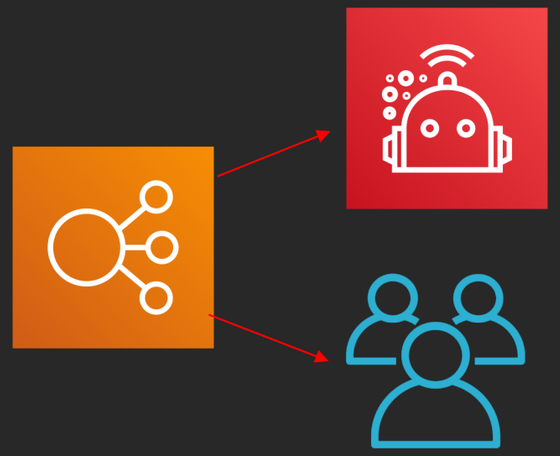
Separate bots/crawlers from human users

Variety in user experience

A/B testing

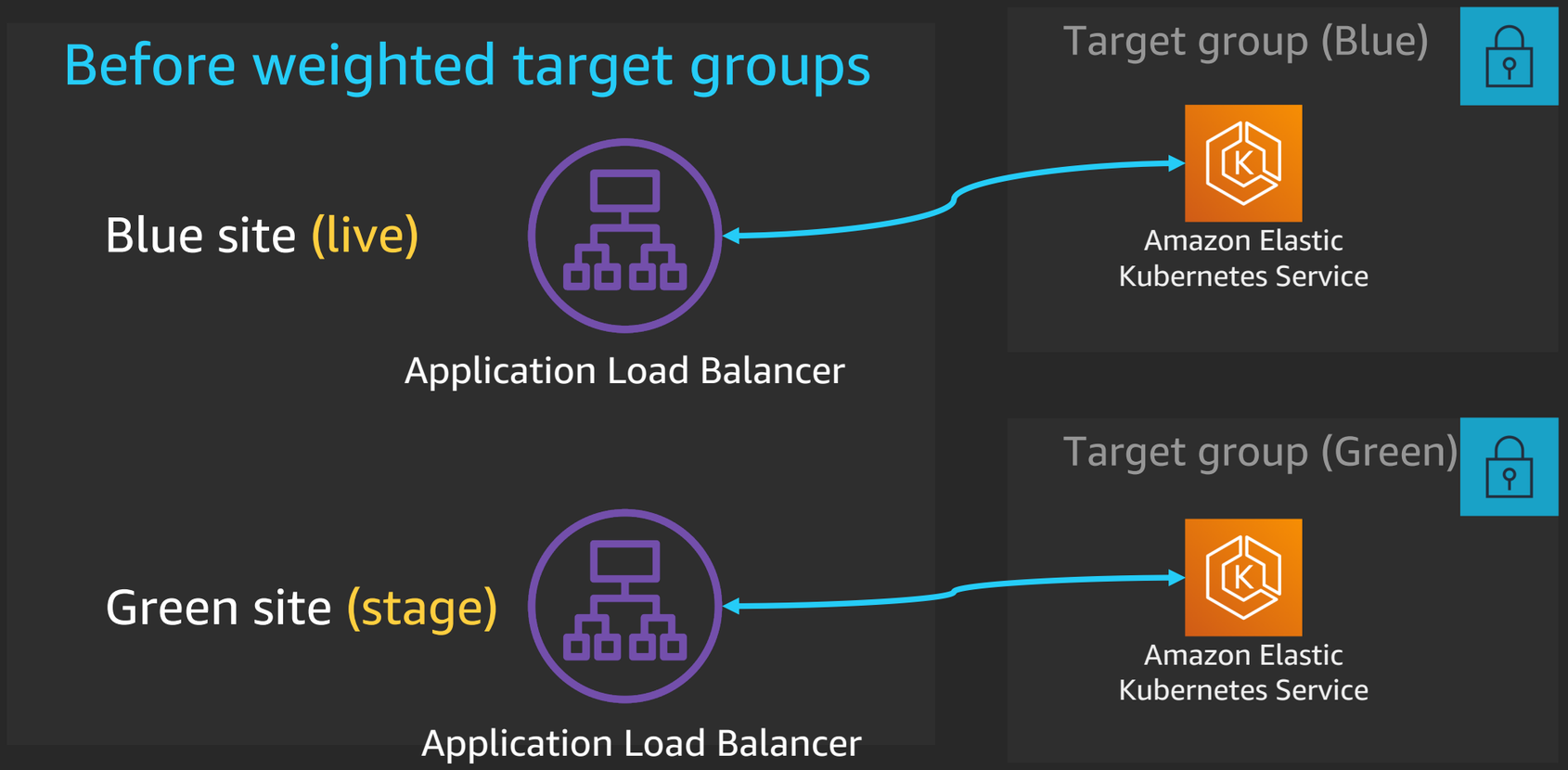
Blue/Green and canary deployments

Access restrictions based on IP addresses/CDNs



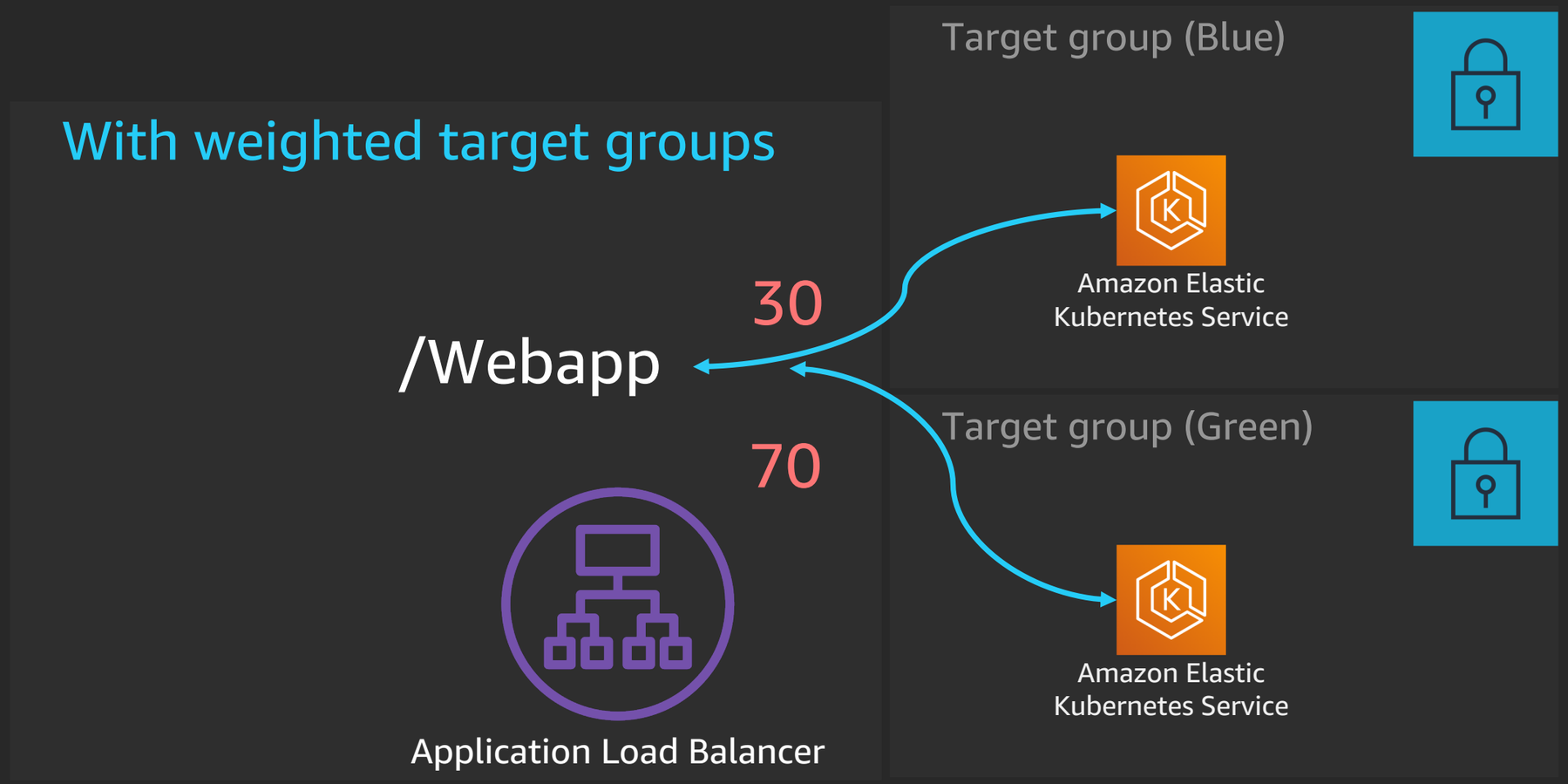
Client connectivity and request routing

Route 53 → ELB → Target Group (rules) → Target



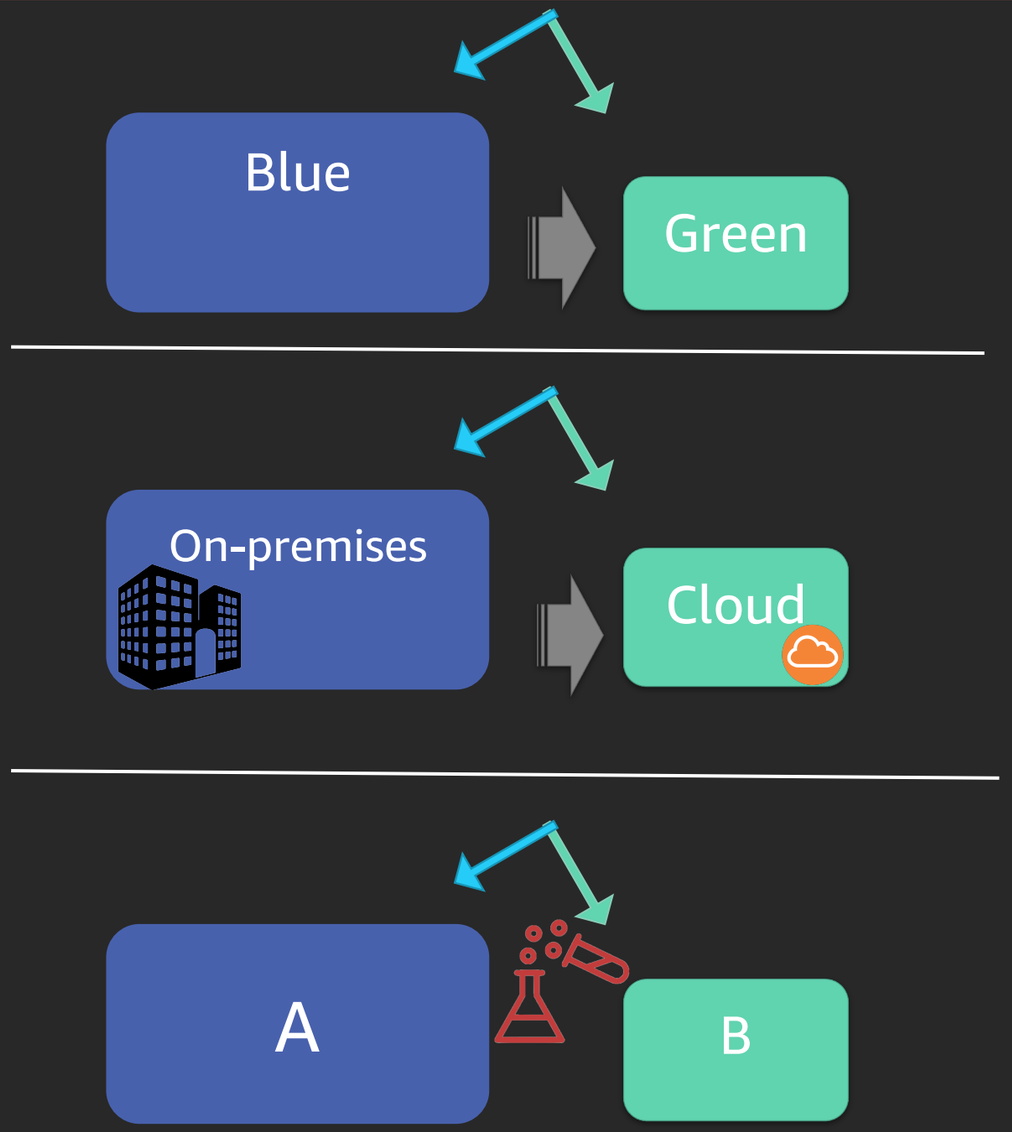
Weighted Target Groups for ALB

Route 53 → ELB → Target Group (rules) → Target



Weighted Target Groups for ALB

- Blue/green deployments
  - In-place deployments, same ELB
    - No DNS caching—TTL issues, no multiple load balancers, no load shifting
- Hybrid deployments, enable cloud migration
  - Targets on-premises & cloud
    - Adjust weights to move traffic from on-premises to cloud or different types of compute
- A/B testing
  - Run experiments with any desired distribution between application versions
  - Update dynamically



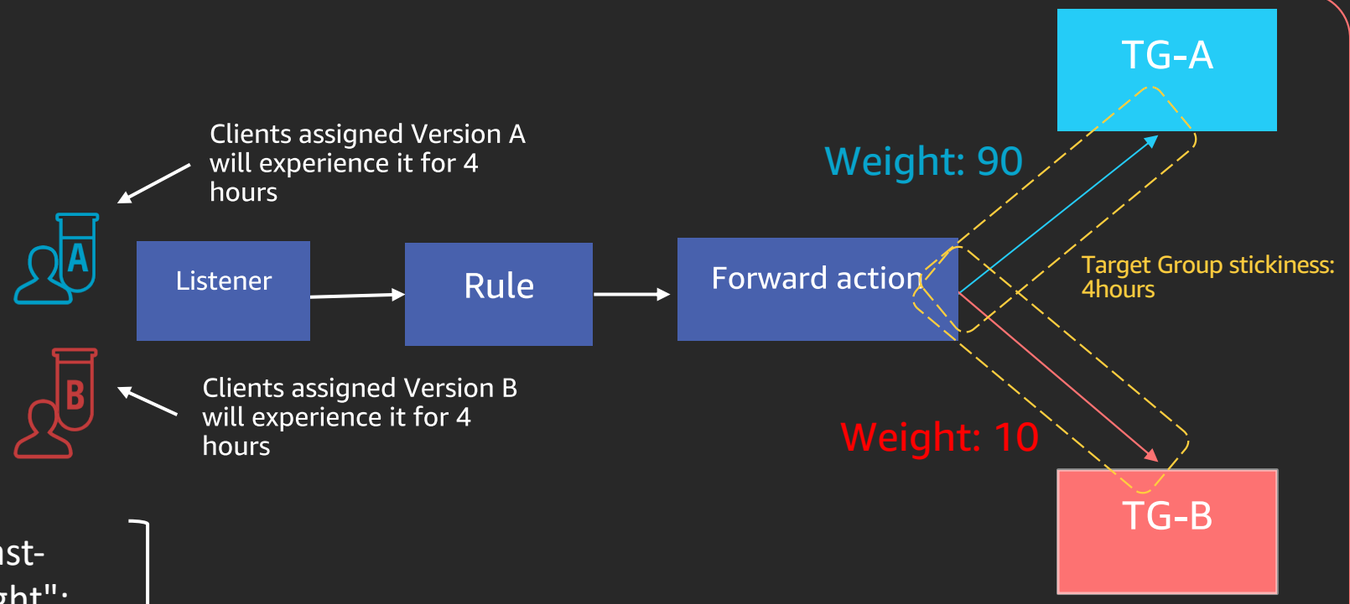
Weighted Target Groups for ALB

```
aws elbv2 modify-listener --listener-arn
arn:aws:elasticloadbalancing:us-east-
1:843251314501:listener/app/WTG-
test/7404d9cf0d0b6e0e/8c54228f6c66762d \
--default-actions \
'[{
   "Type": "forward",
   "Order": 1,
   "ForwardConfig": {
      "TargetGroups": [
         {"TargetGroupArn": "arn:aws:elasticloadbalancing:us-east-
1:843251314501:targetgroup/TG-A/b2e5fa50ebf2b9ef", "Weight":
90},
         {"TargetGroupArn": "arn:aws:elasticloadbalancing:us-east-
1:843251314501:targetgroup/TG-B/1b230eccd0c2bcea", "Weight":
10}
      ],
      "TargetGroupStickinessConfig": {
         "Enabled": true,
         "DurationSeconds": 14400
      }
   }
}]'
```

1. Pick forward action

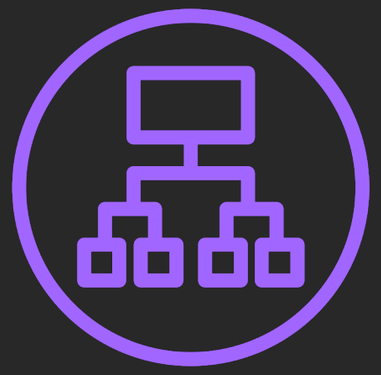
2. Define the target groups to forward to and the desired weights

3. Configure the requests from a client to be served from the same target group that served their first request for X duration



Client connectivity and request routing

Route 53 → ELB (listeners) → Backend target



Application Load Balancer

HTTP/HTTPS

New!

Routes per HTTP request

Least outstanding request routing

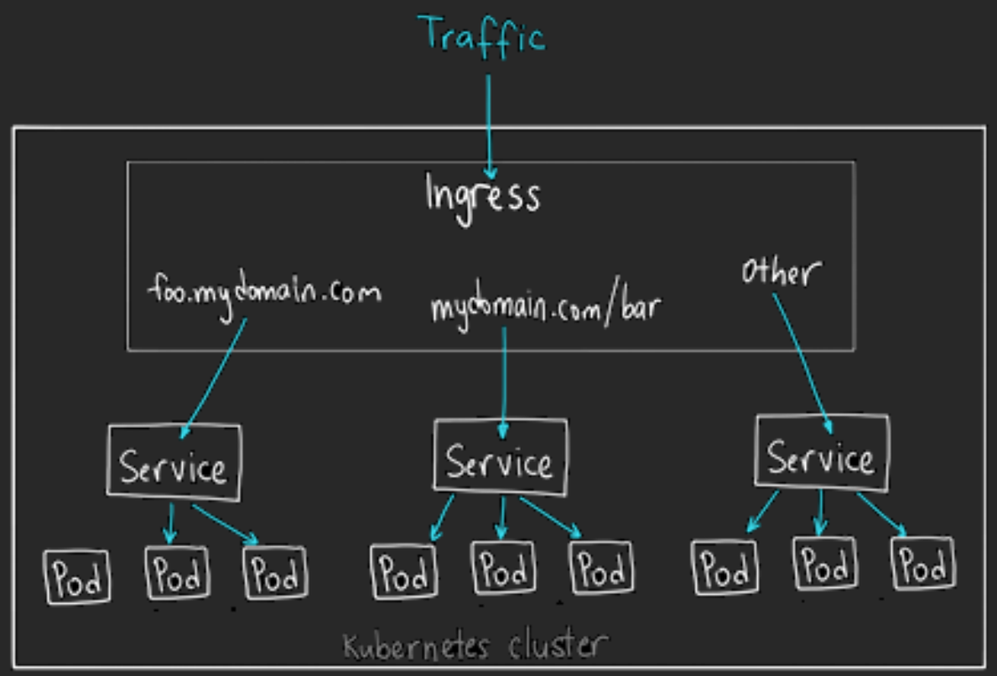
Round robin request routing

ALB Ingress Controller for Kubernetes

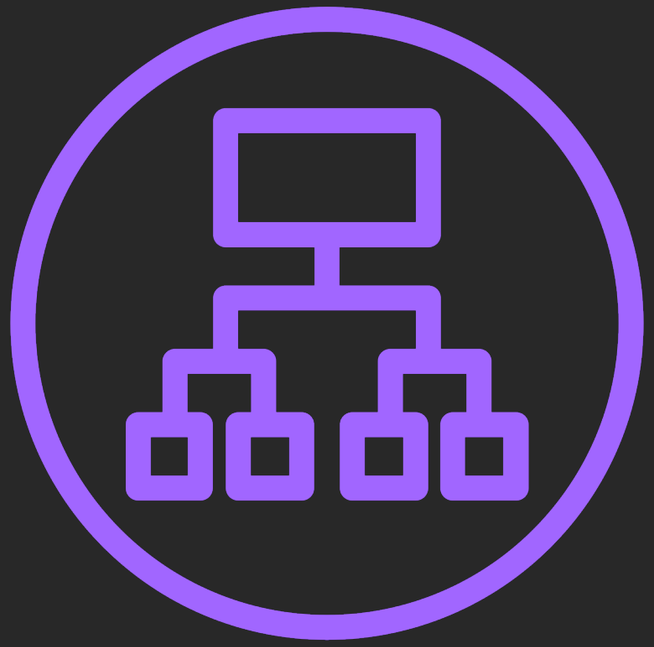
- ALB sits in front of multiple services and act as a “smart router” or entry point into the Kubernetes cluster
- Enables routing requests to different services in the cluster based on host, path, or any advanced request routing features supported by ALB
- Actively maintained by AWS—fast follow-up on any new ALB features
- Ready to use for production workloads!

Learn more:

https://github.com/kubernetes-sigs/aws-alb-ingress-controller



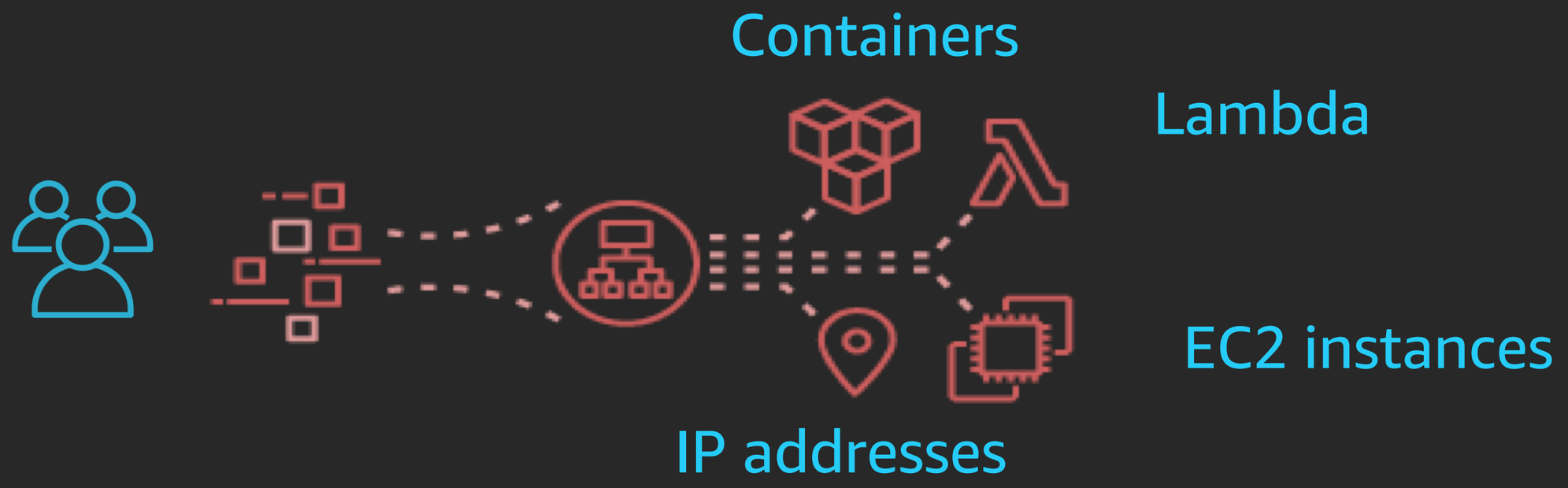
Application Load Balancer



ALB

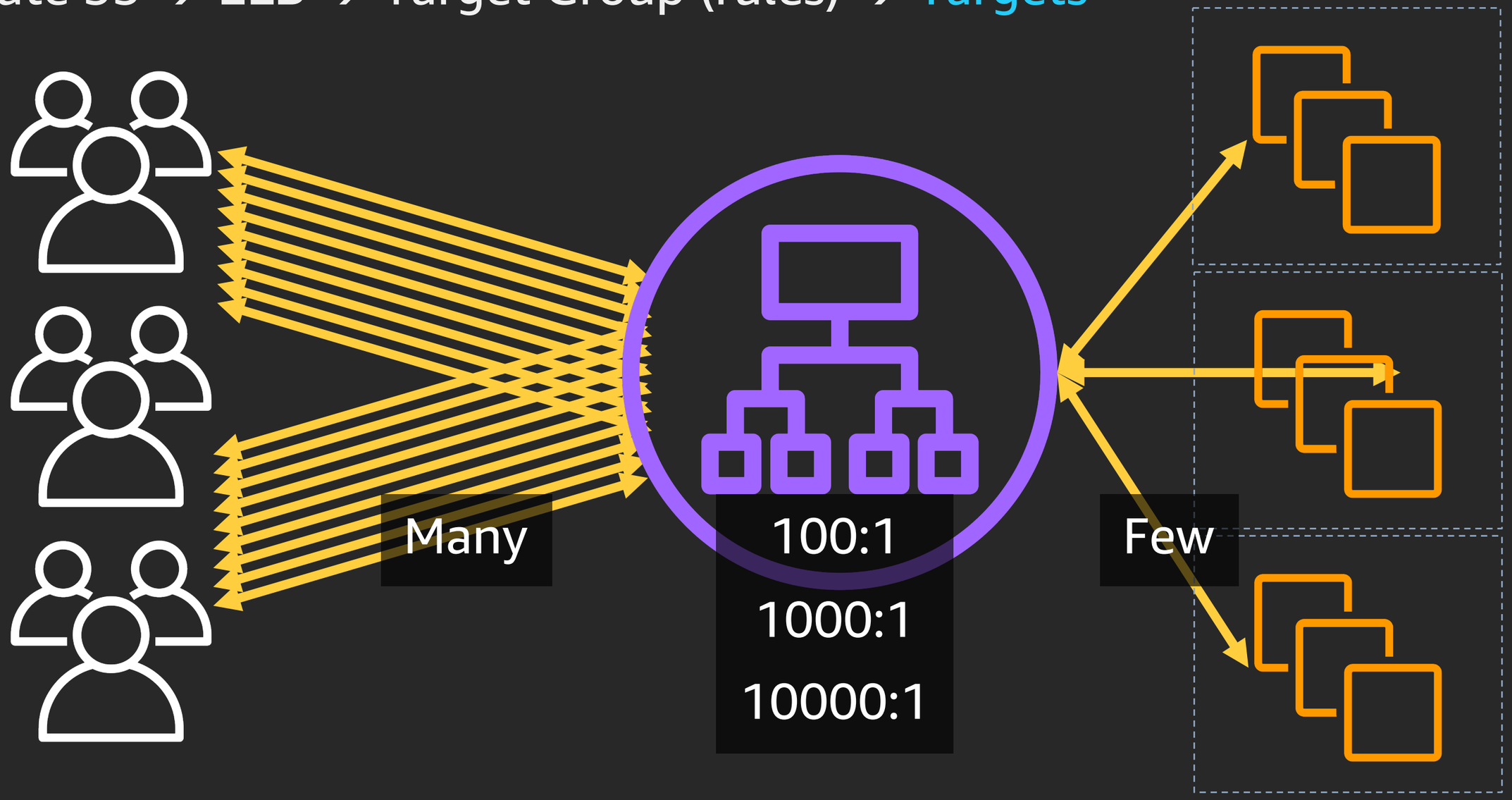
Target configuration

Route 53 → **ELB** → Target Group (rules) → Targets



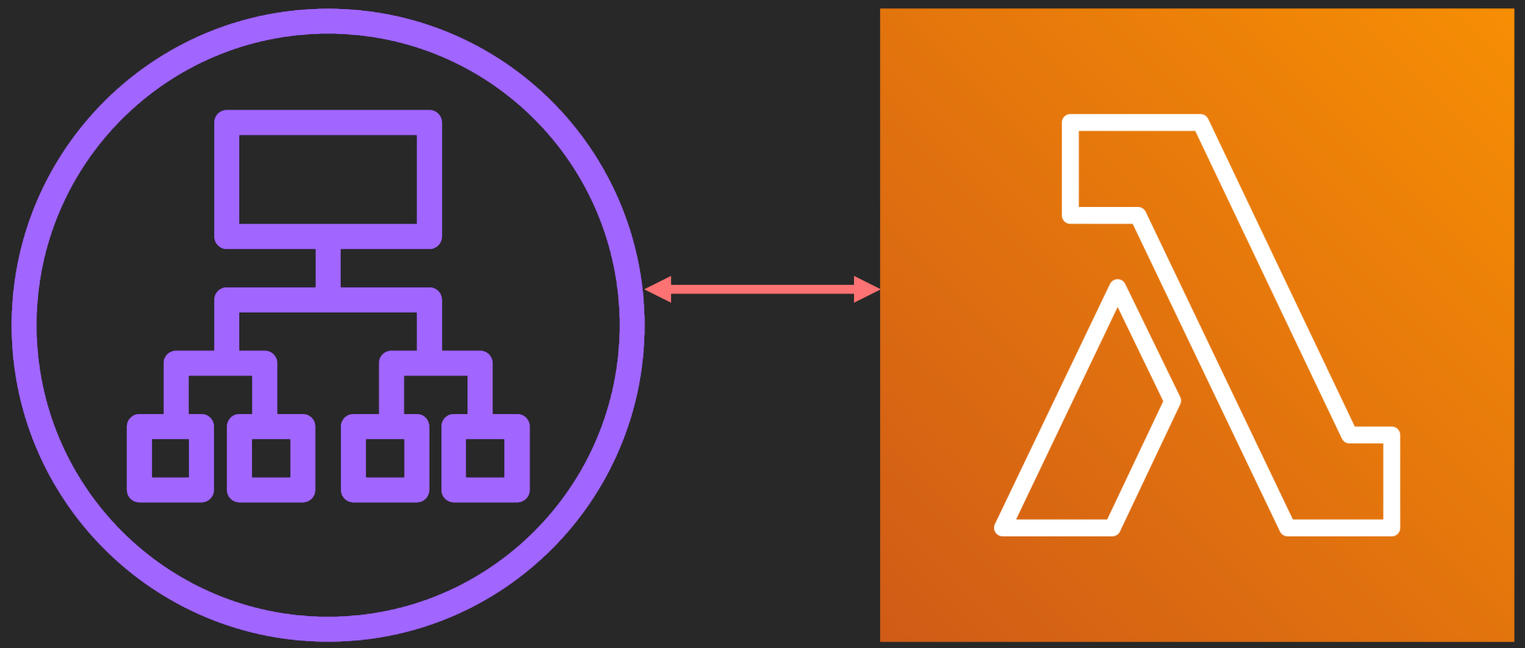
HTTP(S) targets with ALB/CLB

Route 53 → **ELB** → Target Group (rules) → Targets



Client connectivity and request routing

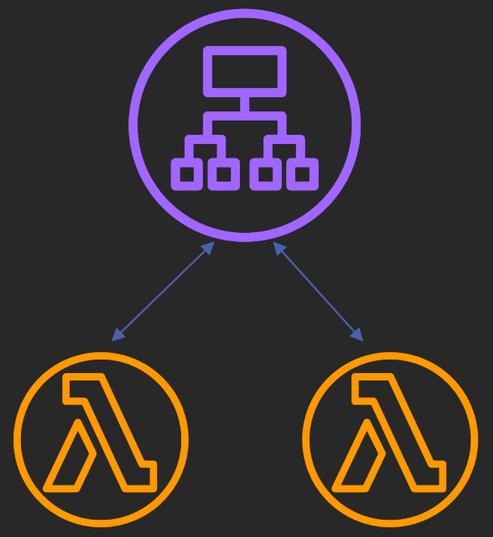
Route53 → ELB → Target Group (rules) → Target Group



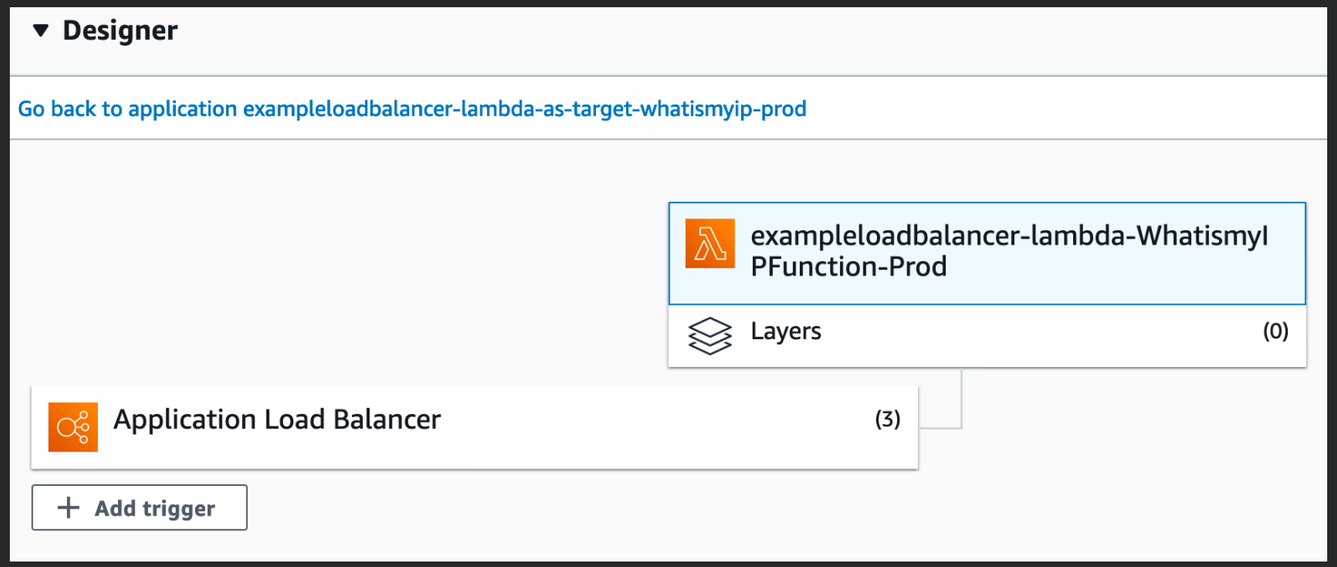
AWS Lambda

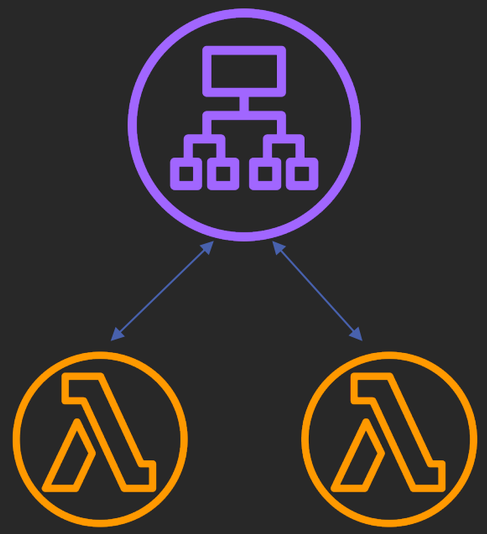
Benefits of Lambda functions as targets

- Build websites as serverless code
- AWS Lambda to manage your functions
- Hybrid applications with mixed compute
- Use any Lambda supported programming languages
- ALB as an HTTP/S frontend to Lambda functions
- Browsers & web clients can invoke Lambda functions
- Create custom targets for ALB like object upload/download to S3
- Reduce cost of infrequently run request paths



Setting up ALB as a trigger in the Lambda console





Author in Lambda console

Create Target Group

Configure ALB rules to route to it

Invoke Lambda events from ALB


```
{
    "requestContext": {
        "elb": {
            "targetGroupArn": "arn:aws:elasticloadbalancing:region:123456789012:targetgroup/my-target-group/6d0ecf831eec9f09"
        }
    },
    "httpMethod": "GET",
    "path": "/",
    "queryStringParameters": {parameters},
    "headers": {
        "accept": "text/html,application/xhtml+xml",
        "accept-language": "en-US,en;q=0.8",
        "content-type": "text/plain",
        "cookie": "cookies",
        "host": "lambda-846800462-us-east-2.elb.amazonaws.com",
        "user-agent": "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_11_6)",
        "x-amzn-trace-id": "Root=1-5bdb40ca-556d8b0c50dc66f0511bf520",
        "x-forwarded-for": "72.21.198.66",
        "x-forwarded-port": "443",
        "x-forwarded-proto” "https"
    },
    "isBase64Encoded": false,
    "body": "request_body"
}
```

Expected response from Lambda function

```
{
    "isBase64Encoded": false,
    "statusCode": 200,
    "statusDescription": "200 OK",
    "headers": {
        "Set-cookie": "cookies",
        "Content-Type": "application/json"
    },
    "body": "Hello from Lambda (body is optional)"
}
```

ALB Lambda limits

Request body (ALB→Lambda function) = 1 MB

Response JSON (Lambda function → ALB) = 1 MB

The Lambda function and ALB must be in the same AWS account

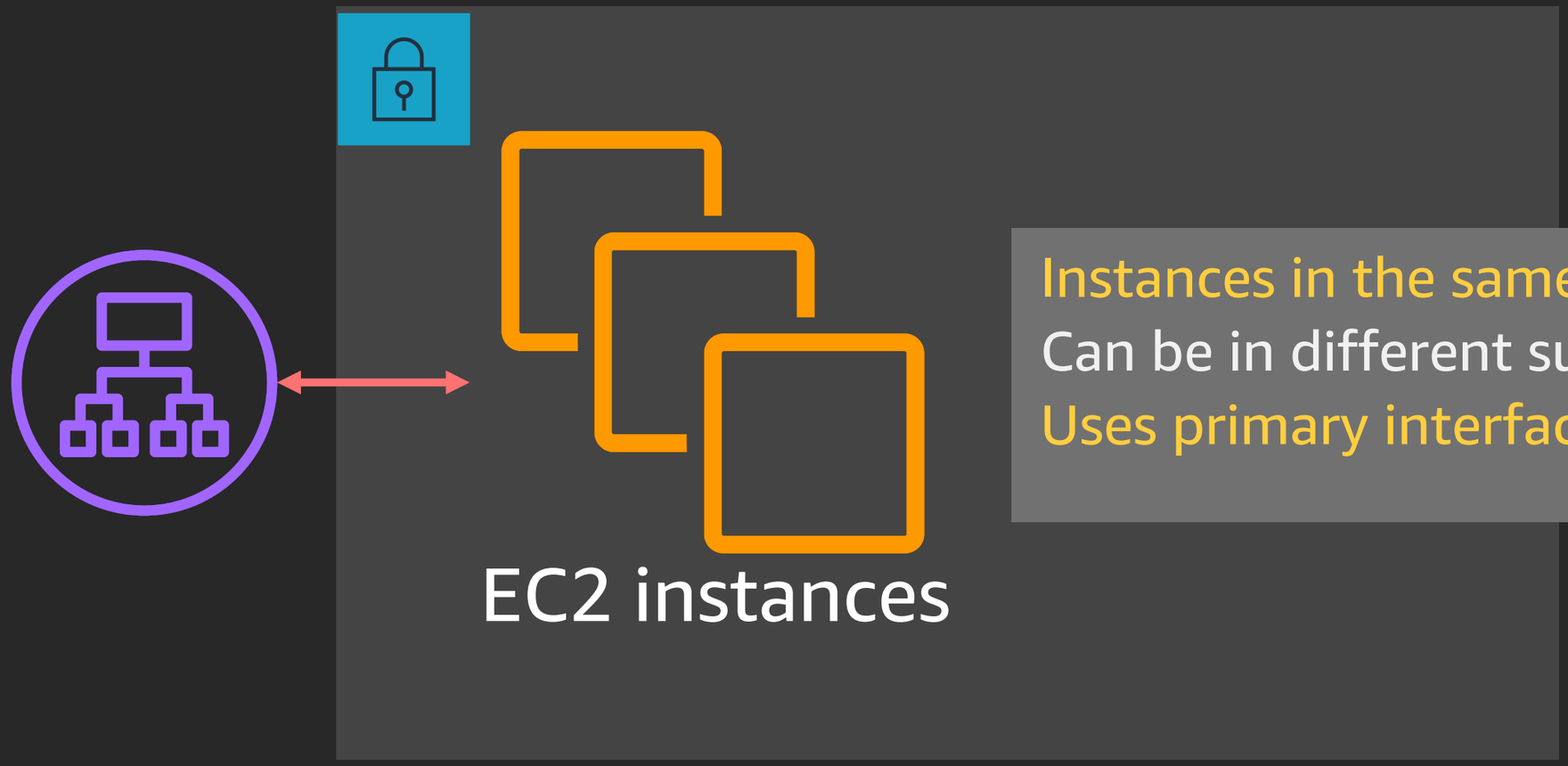
Binary responses need to be BASE64 encoded

Enable multi-value headers to use arrays instead of strings

AWS Lambda

Client connectivity and request routing

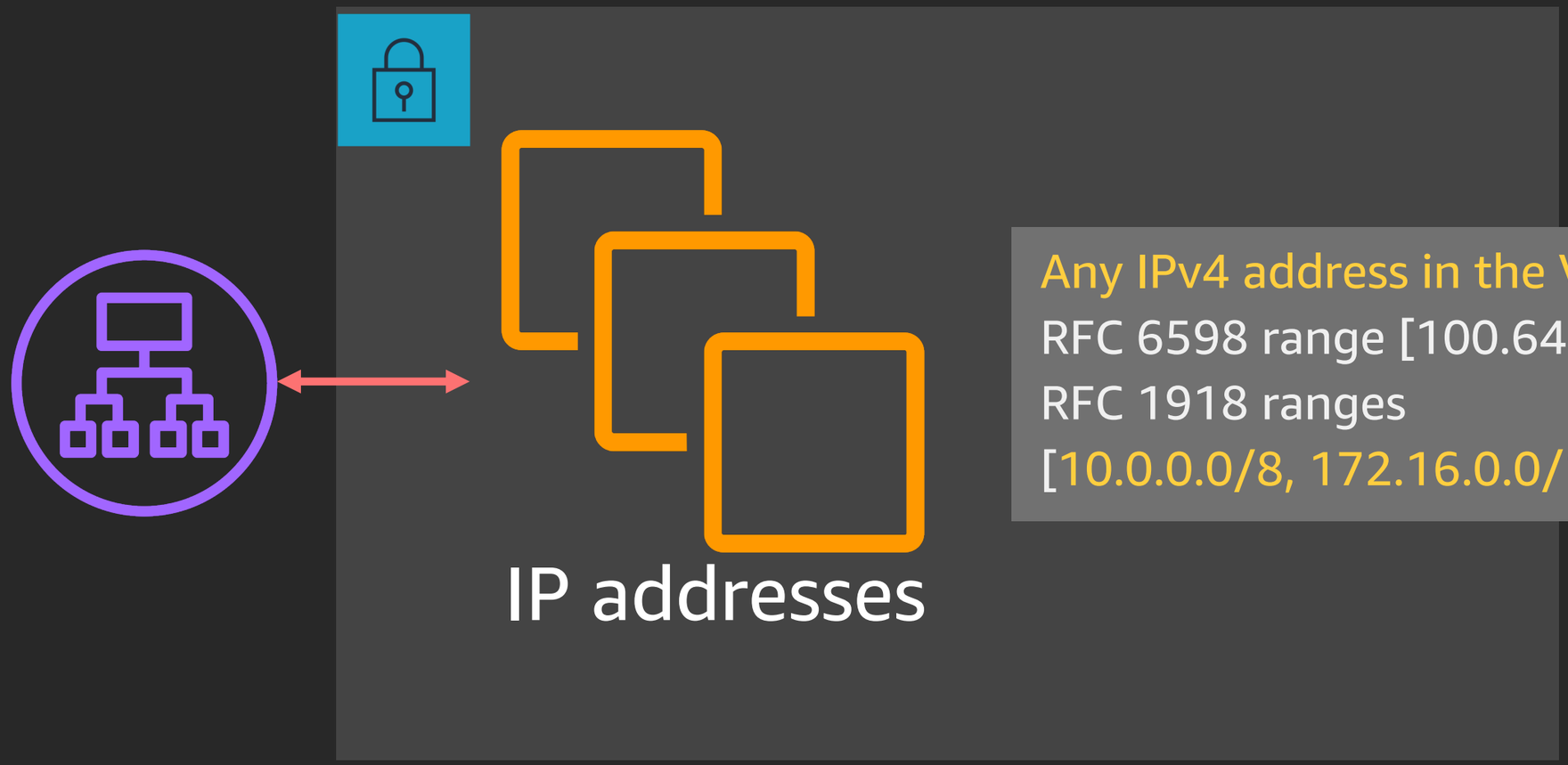
Route53 → ELB (rules) → Target Group



Instances in the same VPC
Can be in different subnets
Uses primary interface (eth0)

Client connectivity and request routing

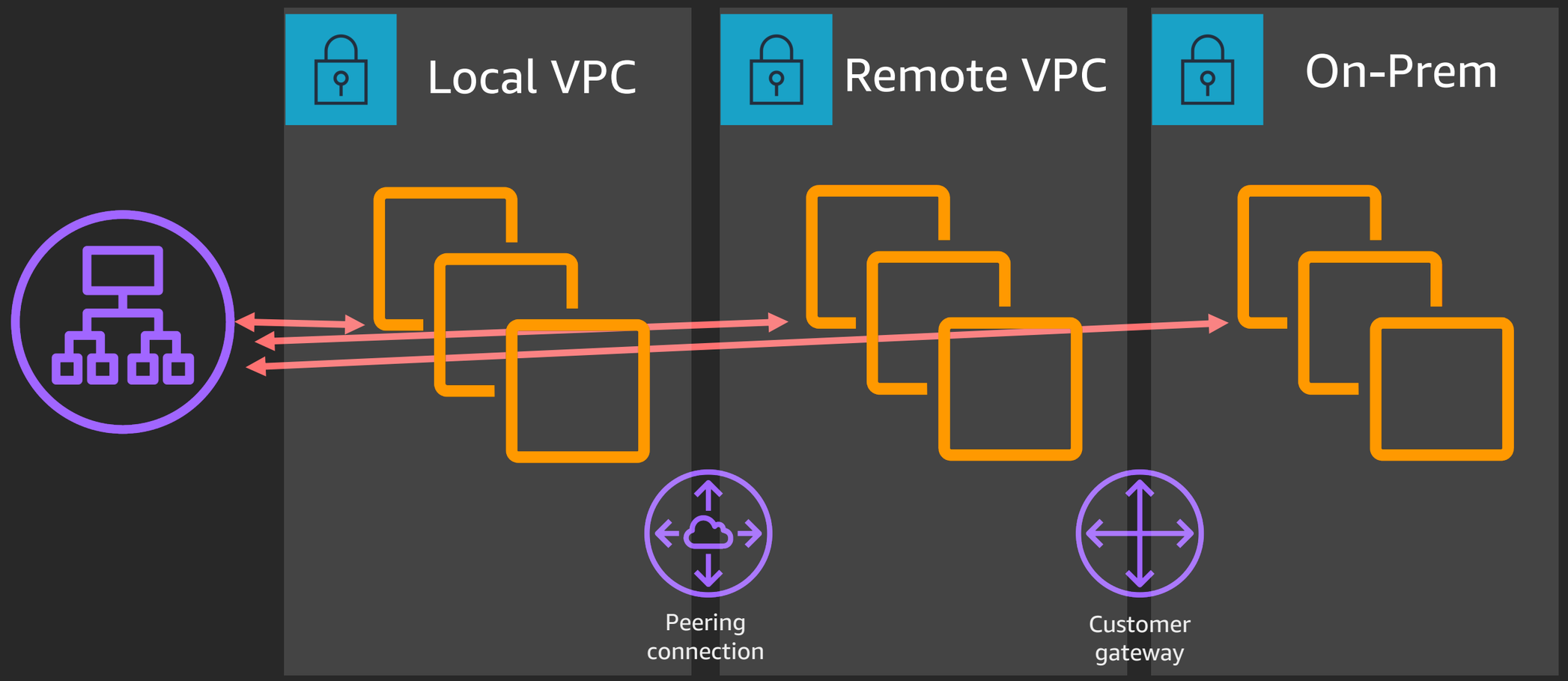
Route53 → ELB (rules) → Target Group



Any IPv4 address in the VPC
RFC 6598 range [100.64.0.0/10]
RFC 1918 ranges
[10.0.0.0/8, 172.16.0.0/12, and 192.168.0.0/16]

Client connectivity and request routing

Route53 → ELB (rules) → Target Group

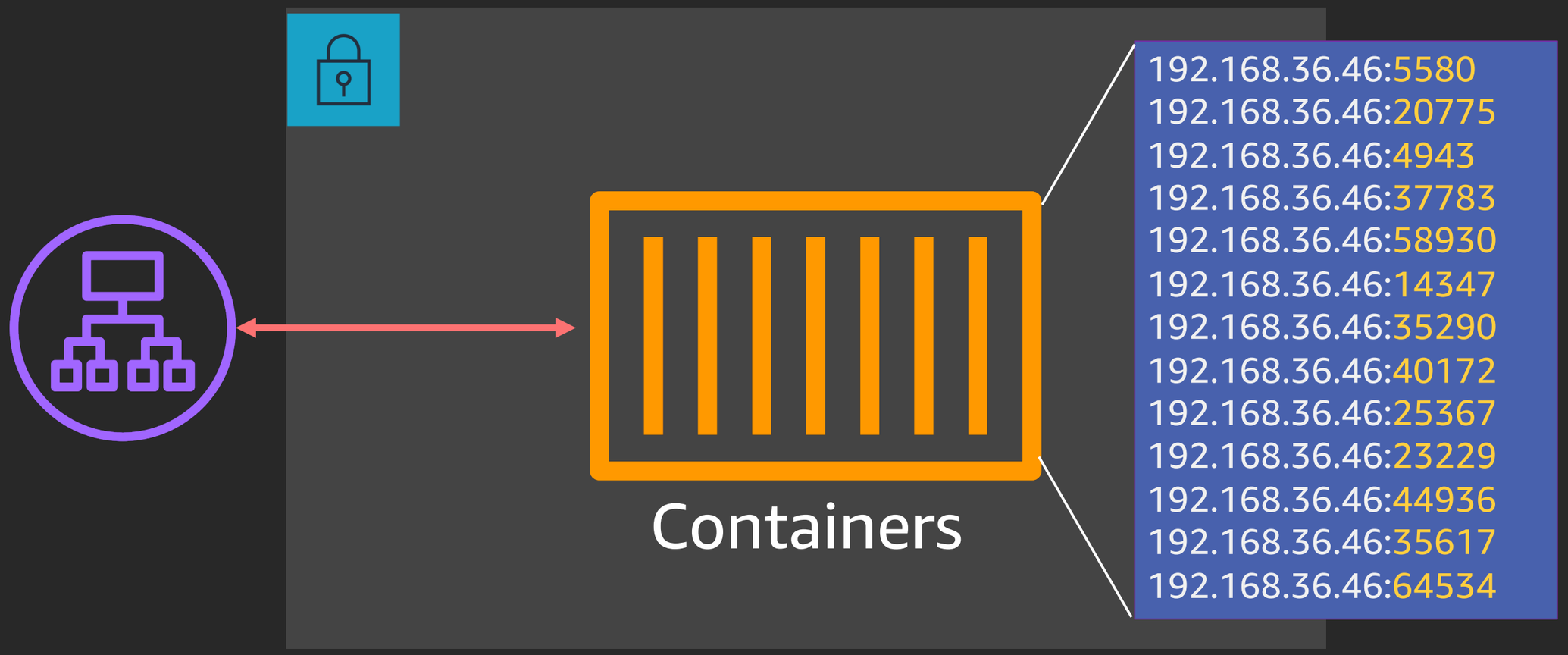


{Instance}

{IP targets}

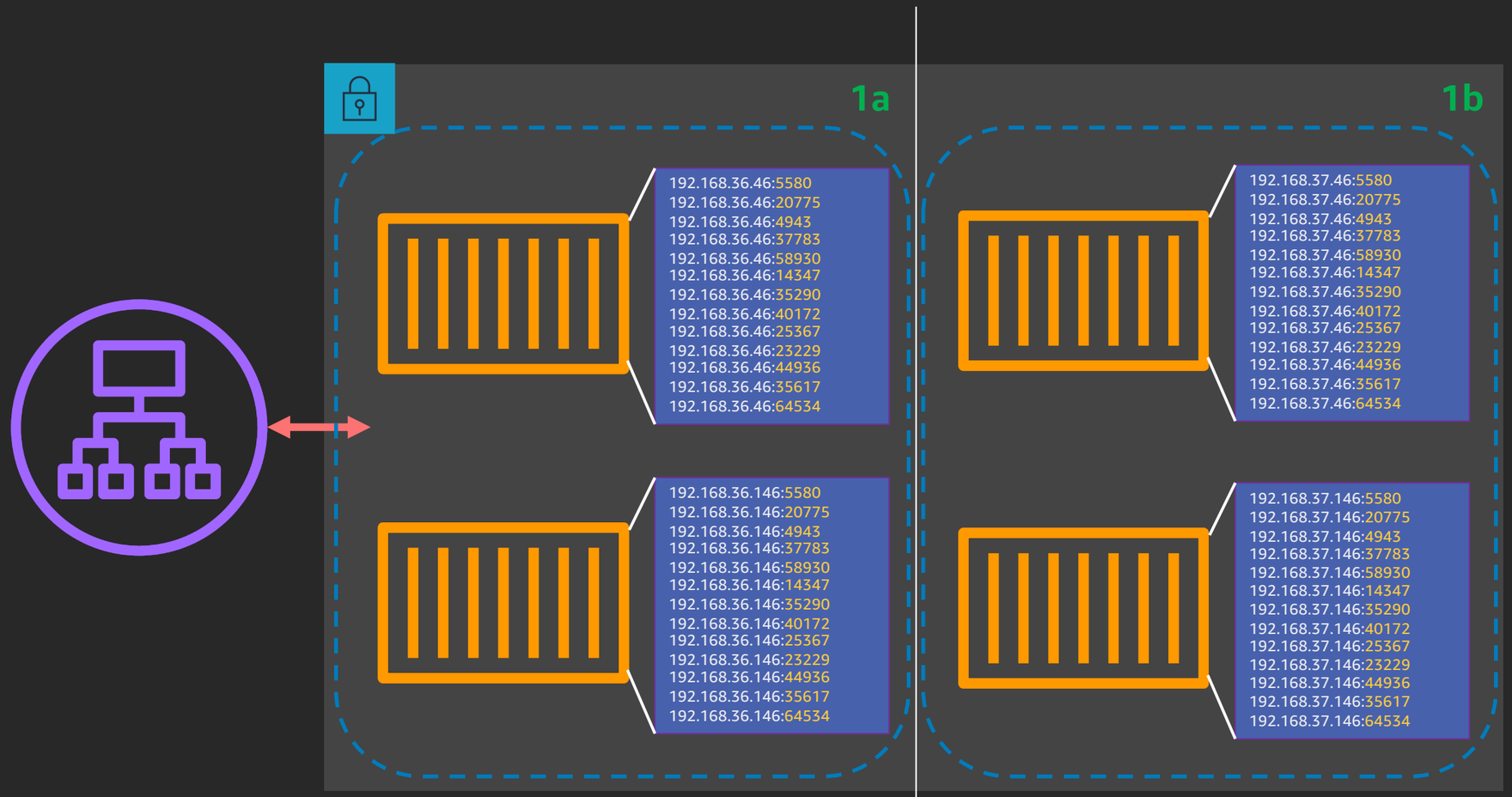
Client connectivity and request routing

Route53 → ELB (rules) → Target Group



Client connectivity and request routing

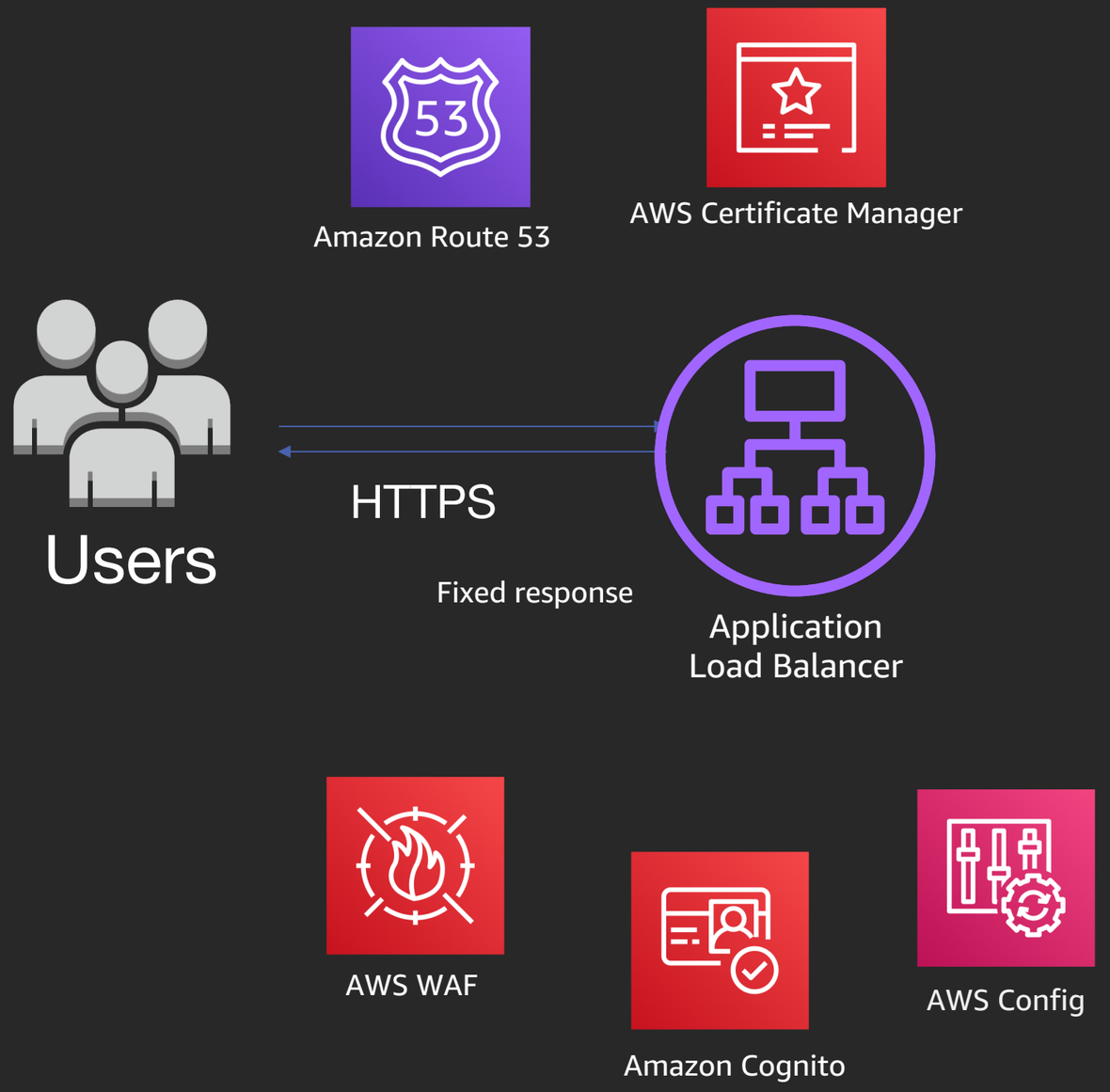
Route53 → ELB (rules) → Target Group



Client connectivity and request routing

Security

Target best practices: Security



- Managed certificate rotation
- Offload authentication to ALB
- Calculated health checks
- AWS WAF
- AWS Config - monitor state

Target best practices: Security

- Offload work to load balancer
- Encrypt traffic with TLS!
- Automatic certificate renewal management with ACM
- Use the most secure TLS policy you can

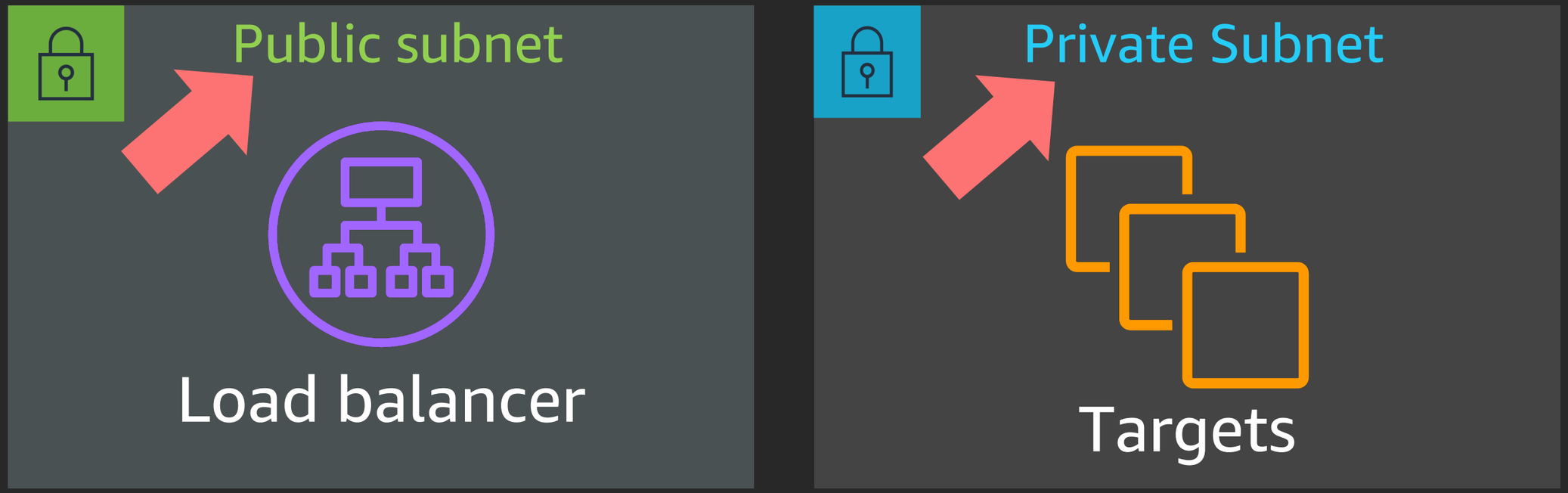
ELB SecurityPolicy -2016-08 (default)
ELB SecurityPolicy-TLS- 1-0-2015-04
ELB SecurityPolicy-TLS- 1-1-2017-01
ELB SecurityPolicy-TLS- 1-2-2017-01
ELB SecurityPolicy-TLS- 1-2-Ext-2018-06
ELB SecurityPolicy-FS-2018-06
ELB SecurityPolicy-FS-1-1-2019-08
ELB SecurityPolicy-FS-1-2-2019-08
ELB SecurityPolicy-FS-1-1-Res-2019-08

ELB SecurityPolicy -2016-08 (default)

ELB SecurityPolicy-FS-1-2-2019-08

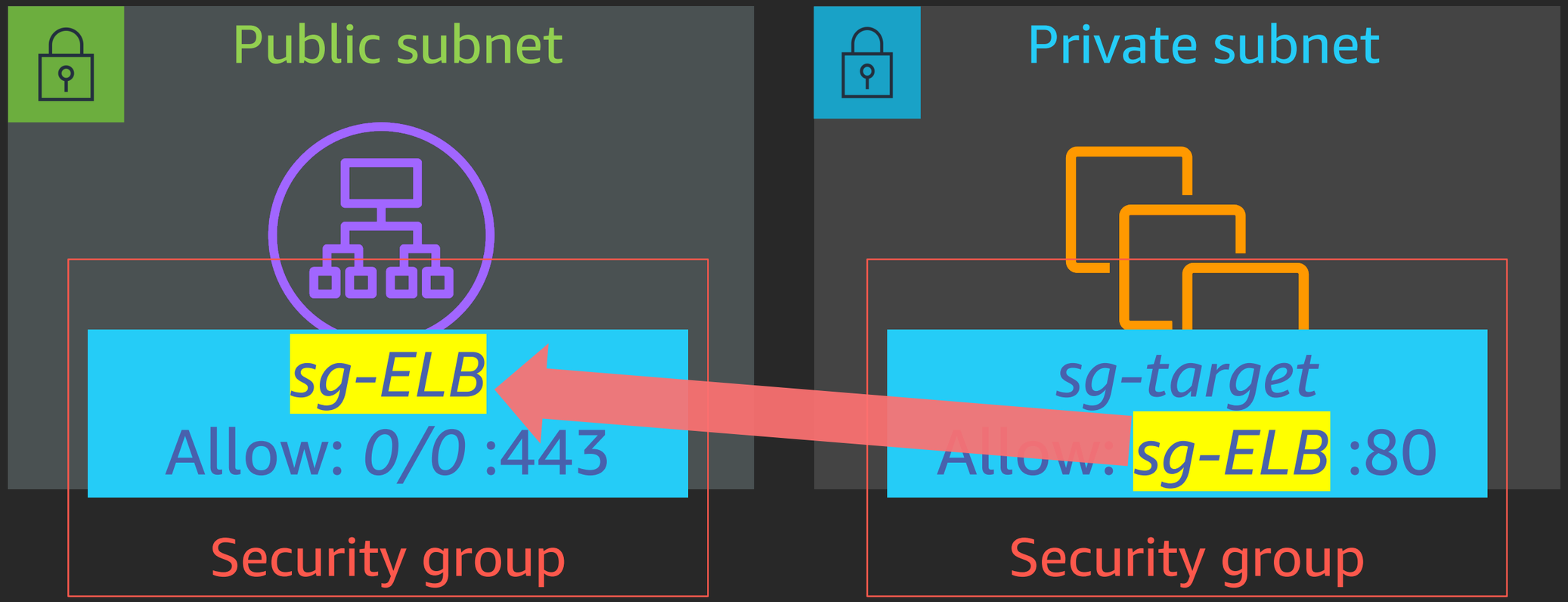
Target best practices: Security

Best practice: Internet-facing load balancer in public subnet
With targets in a private subnet



Target best practices: Security

Best practice: Target security groups only allow inbound traffic from load balancer by security group ID



ELB best practices: Security

Best practice:
Use AWS WAF for protecting your HTTP/S traffic to your target

Scalable custom rules engine
Write hundreds of rules
Update rule in seconds
Inspection done with minimal latency impact
Supports regex pattern matching, SQLi/XSS detection, and more

Threat intelligence and managed rules
AWS Managed Rules
- Core rule set based on OWASP top 10 and leverages Amazon threat intelligence

AWS Marketplace sellers
- Comprehensive OWASP top 10 protection, CVEs, IP reputation list, and more

Advanced automation through API
AWS WAF Security Automation (AWS CloudFormation available)

ELB best practices: Security

Best practice:
Offload authentication to the load balancer

Simplify login with Application Load Balancer built-in authentication

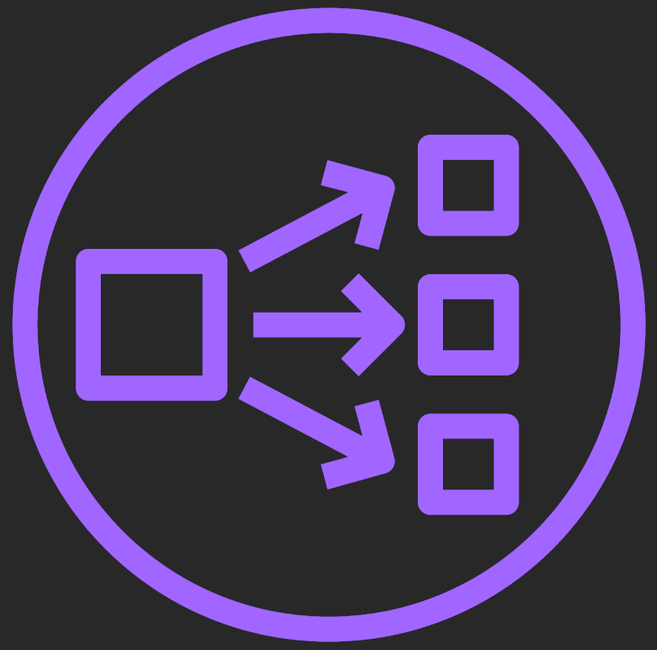
- Native integration with any OIDC supported IdP
  - No additional AWS charges incurred

- Seamless integration with Amazon Cognito

  Authenticate with corporate identities using SAML, LDAP, Microsoft AD, or OIDC

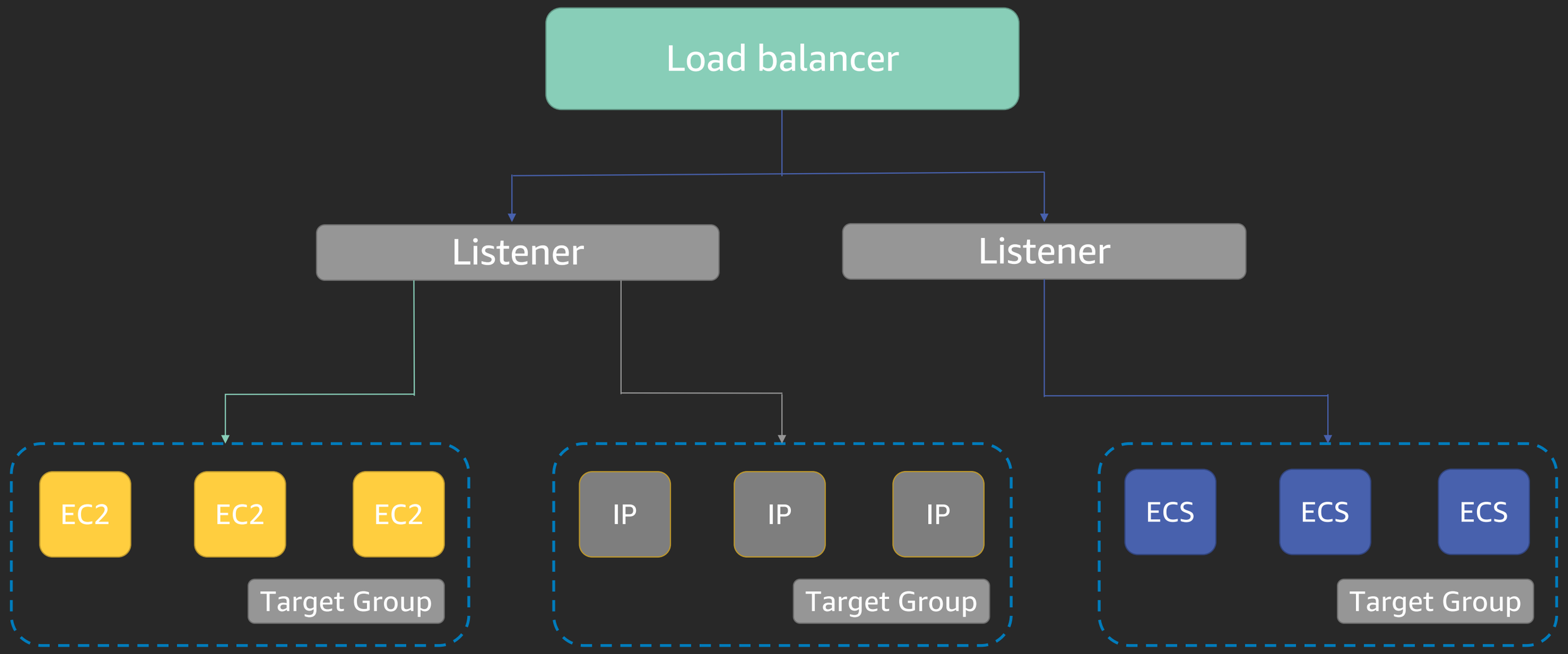
  Authenticate with federated identities based on social networks (Facebook, Google, Amazon)

Network Load Balancer

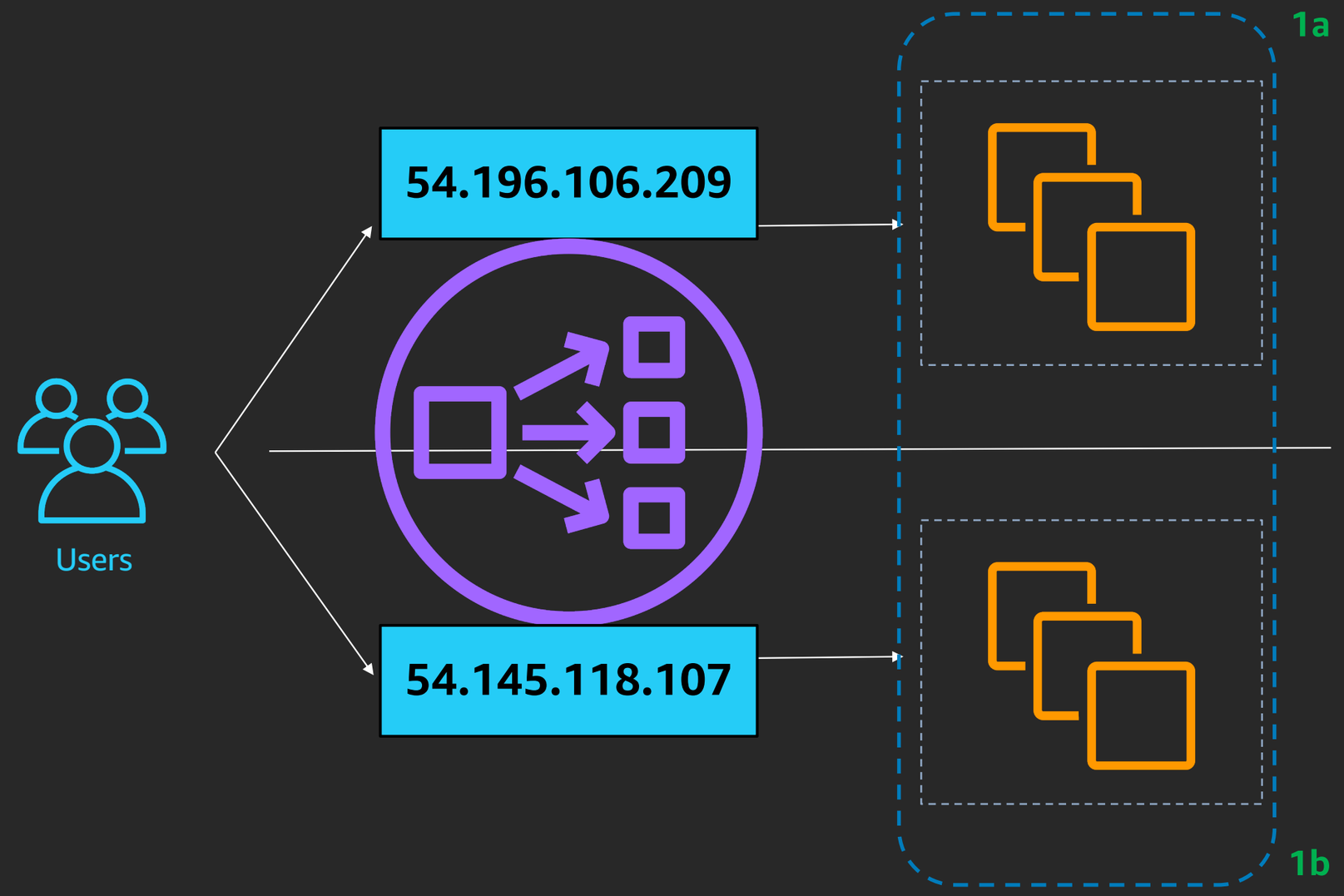


NLB

Logical overview of NLB

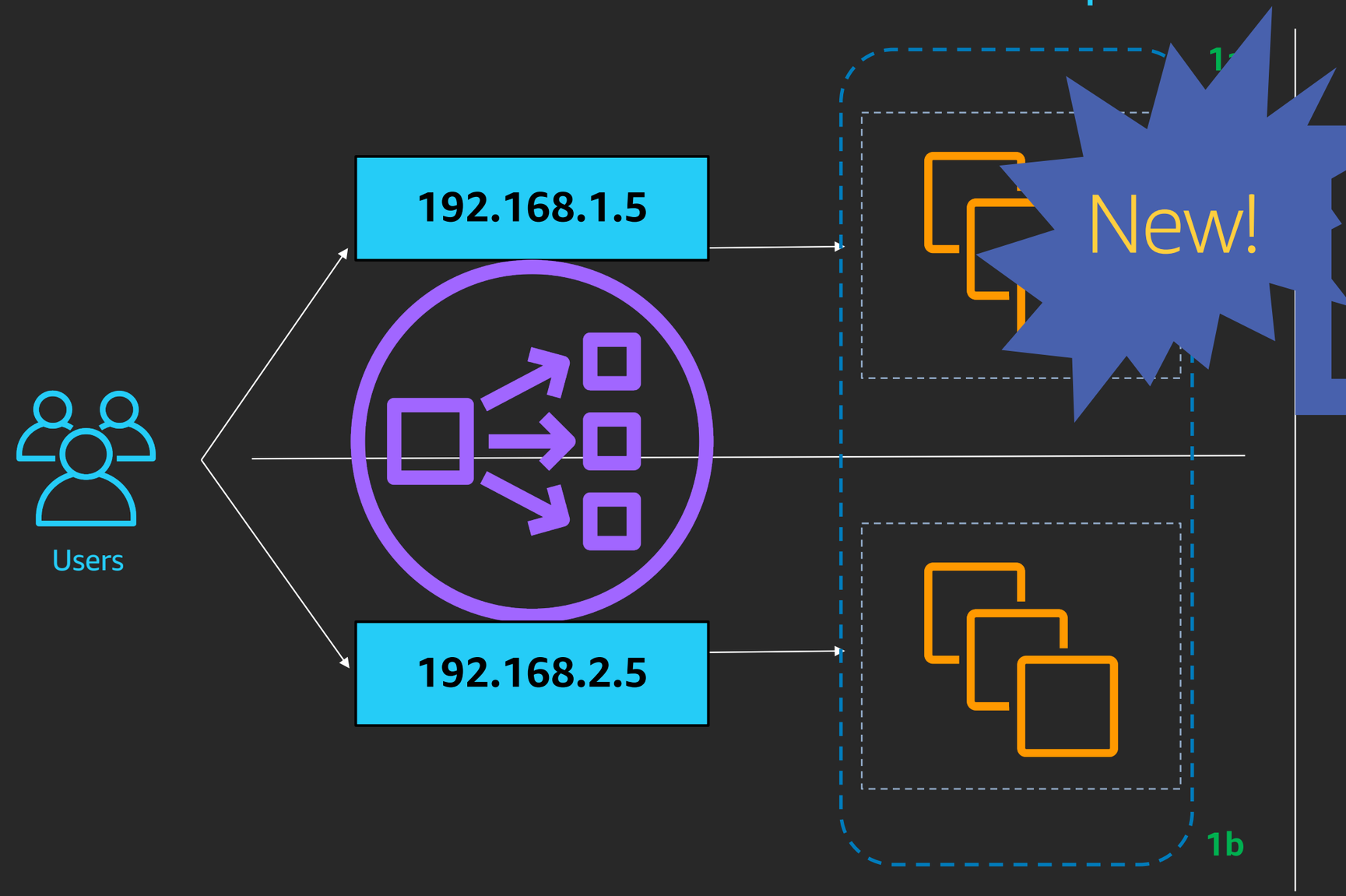


NLB Elastic IP addresses



Assigning Elastic IP provides a single IP address per Availability Zone per load balancer that will not change

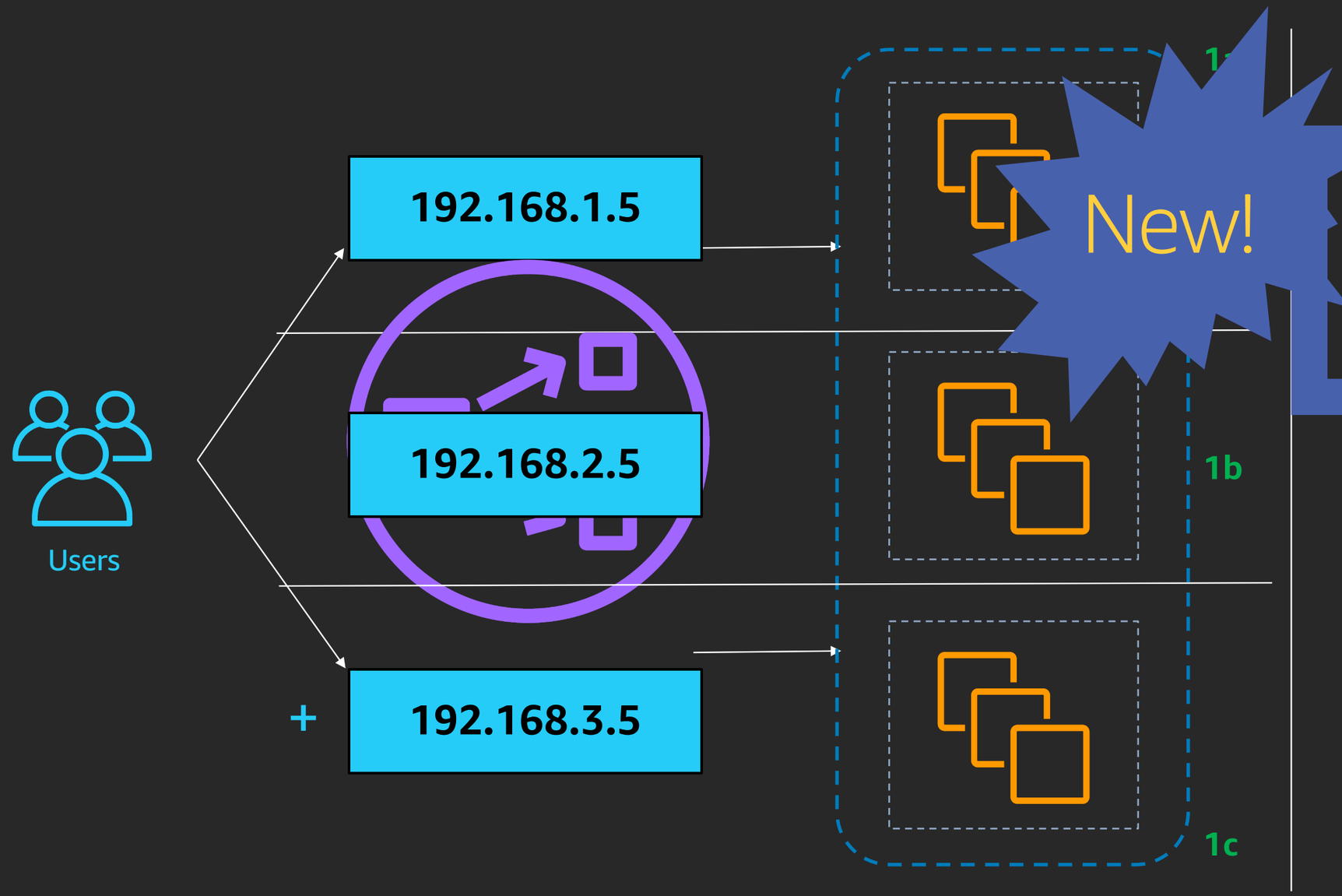
NLB Elastic IP addresses: Select private IP



New!

You can now select the private IP address of subnets in your NLB

NLB Elastic IP addresses: Add subnets

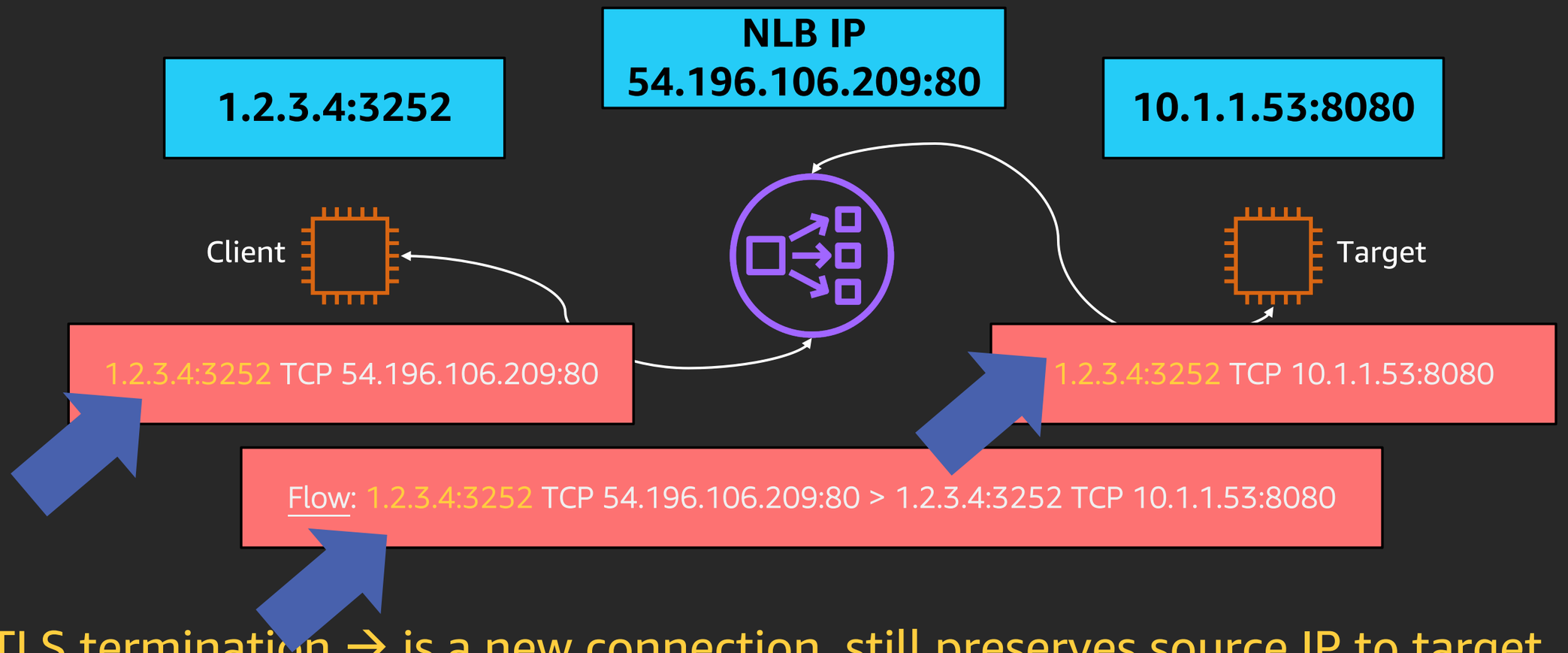


New!

You can also add additional subnets to an existing NLB

NLB connection routing

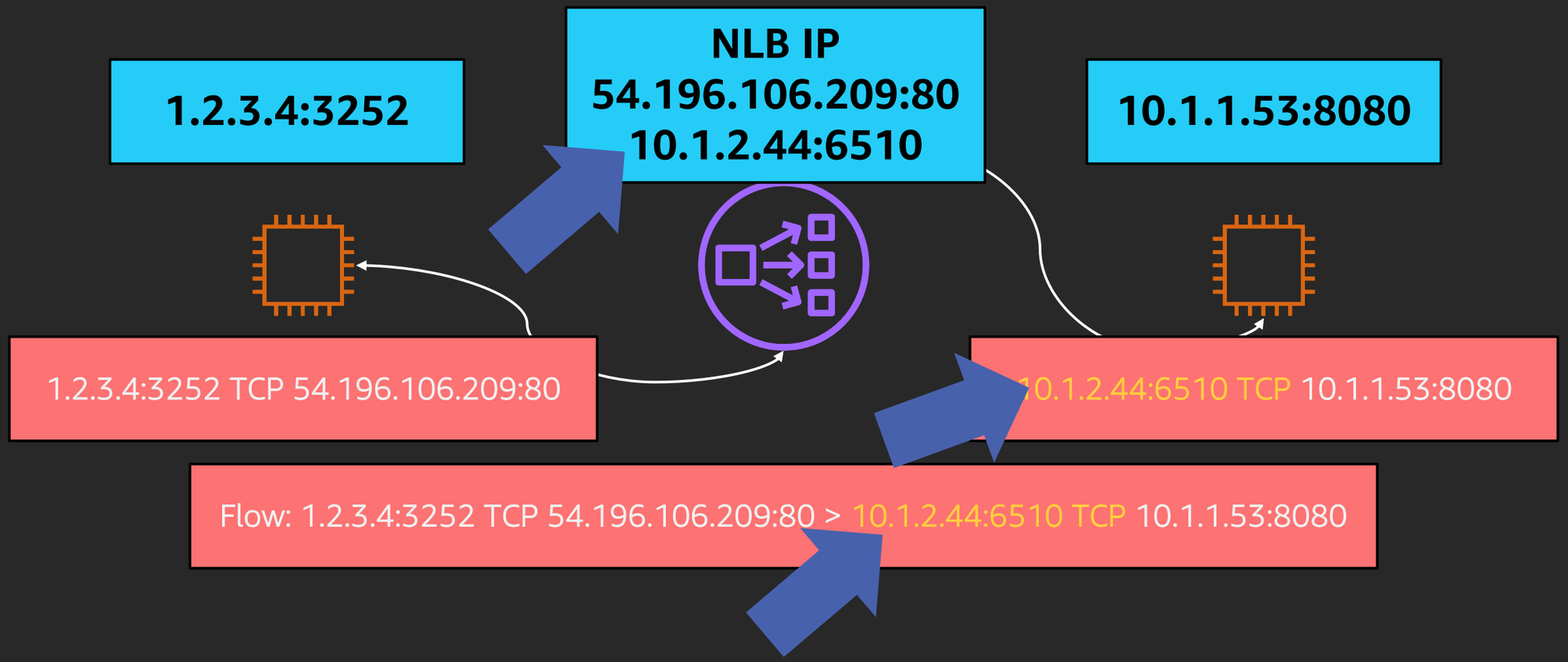
Client → NLB → Targets (instance targets)



TLS termination → is a new connection, still preserves source IP to target

NLB connection routing

Client → NLB → Targets (IP targets)



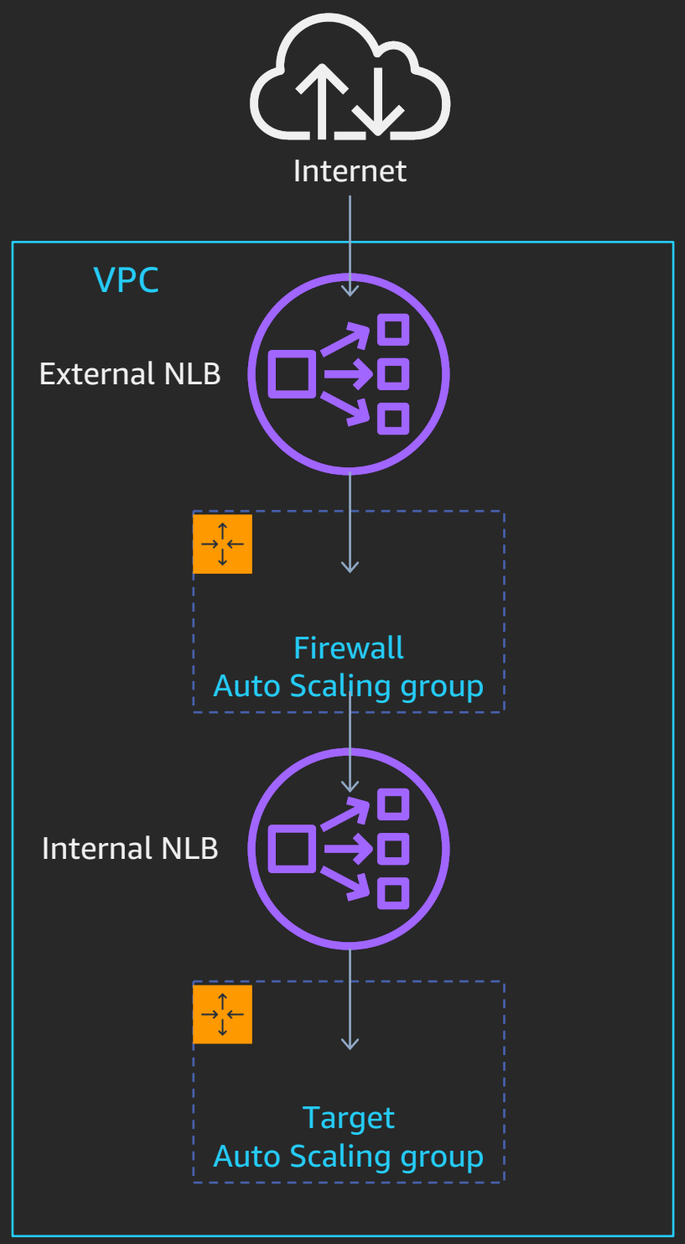
Firewall example using NLB

External connectivity to fewer IP addresses

1 IP per Availability Zone

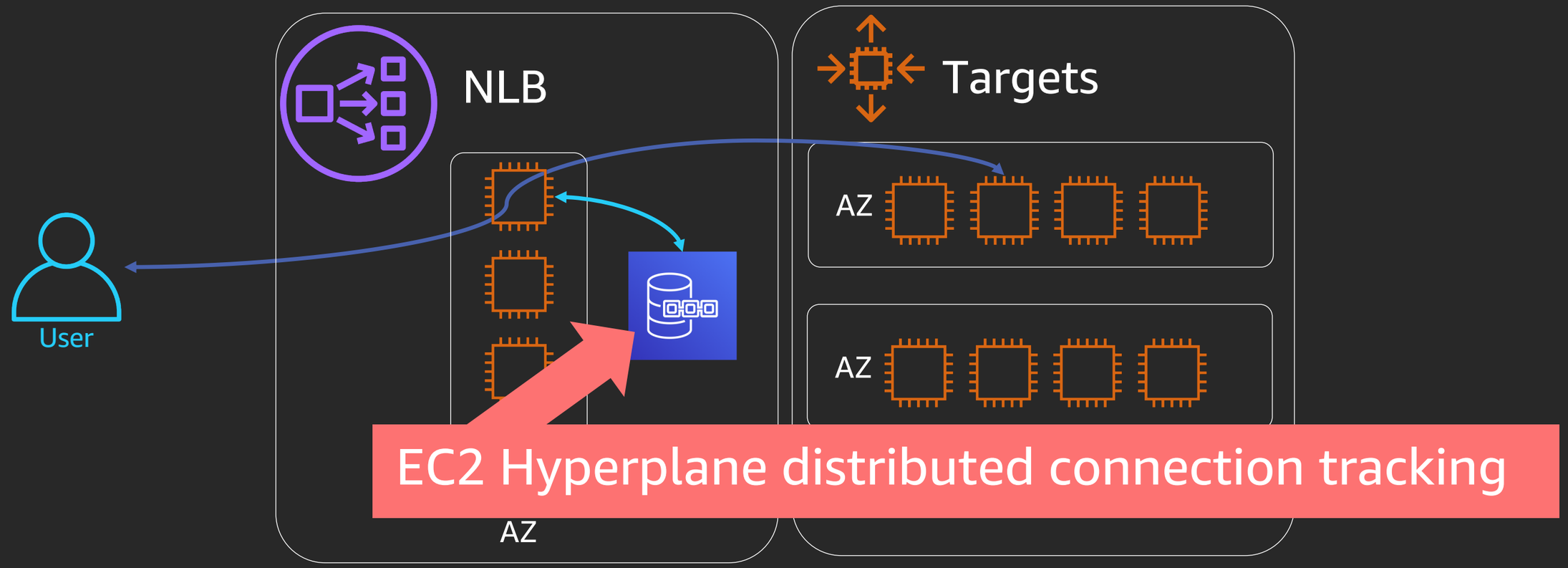
Preserves source IP to your Firewall

Firewall can target the static IPs of internal NLB



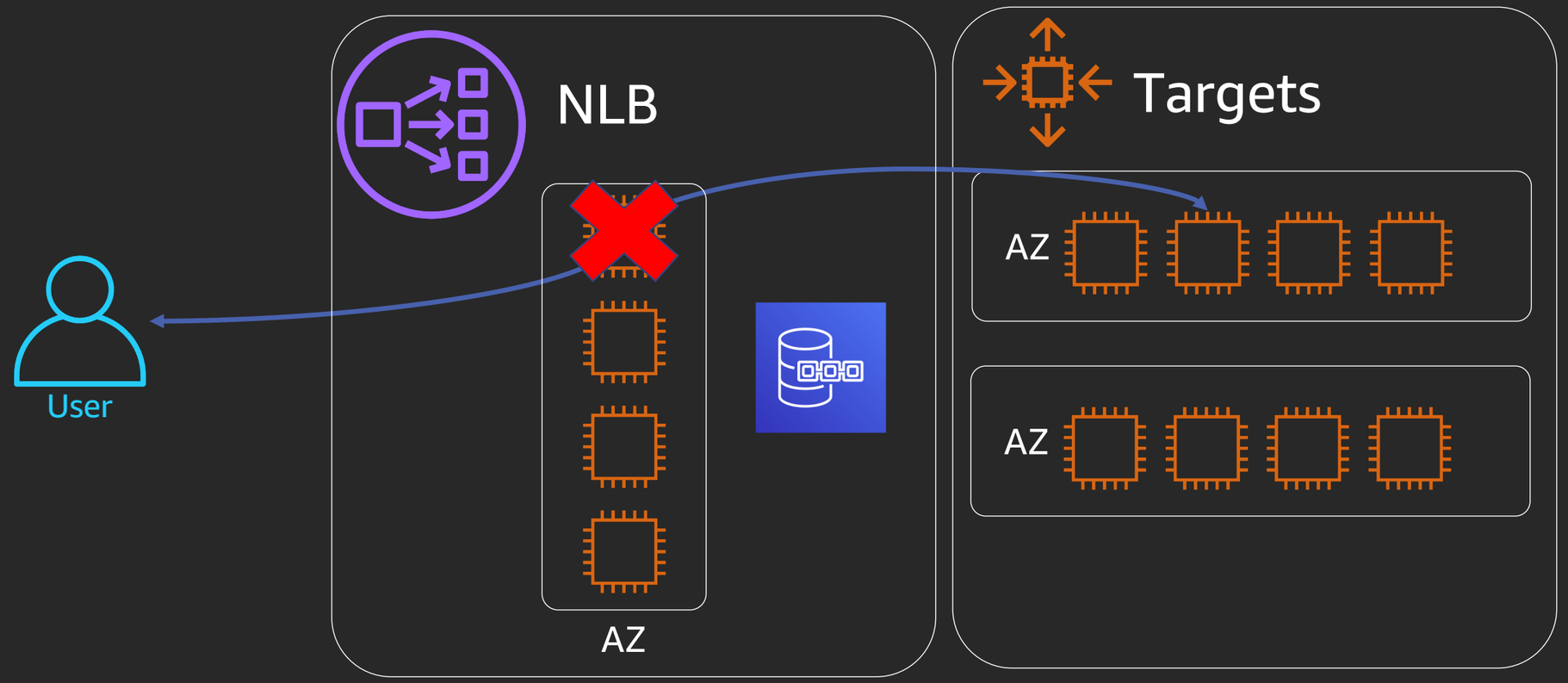
Long-lived connections on NLB

Client → NLB → Targets



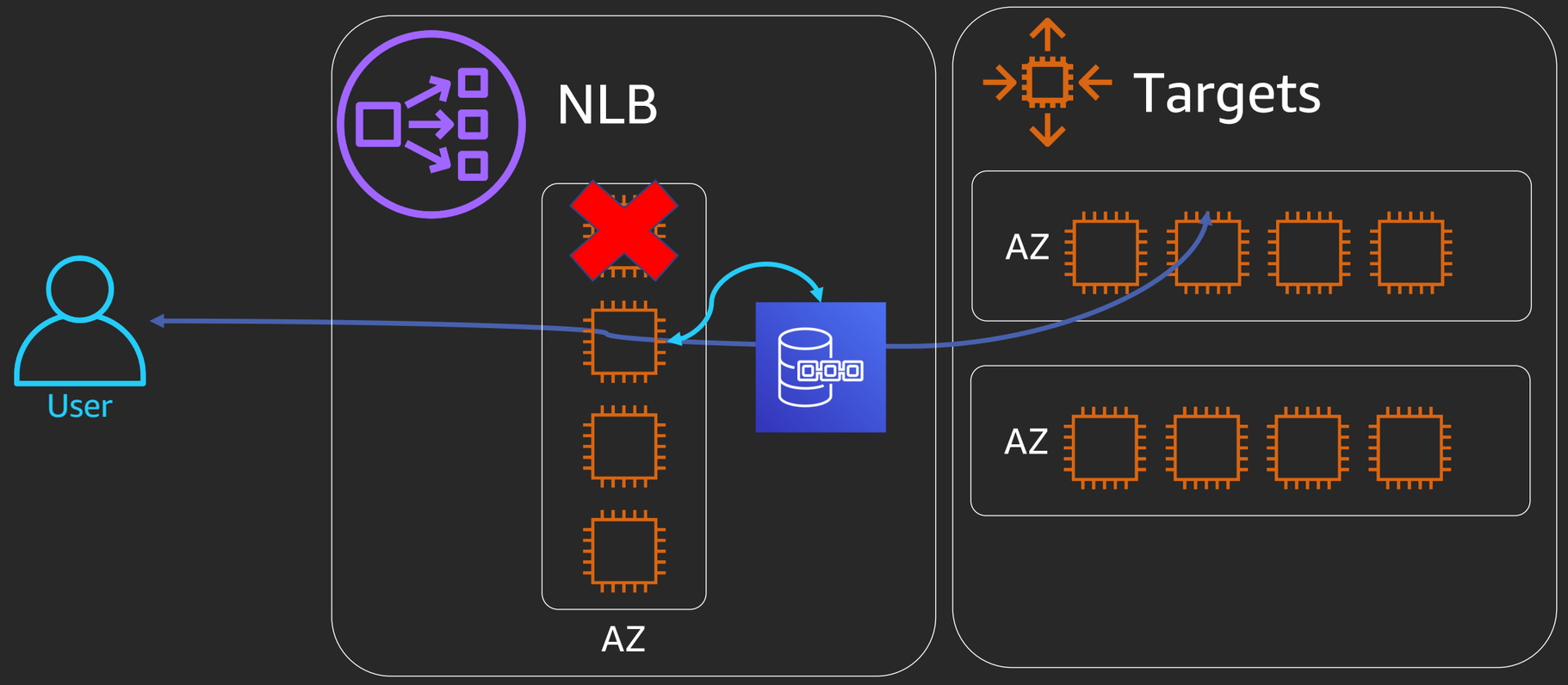
Long-lived connections on NLB

Client → NLB → Targets

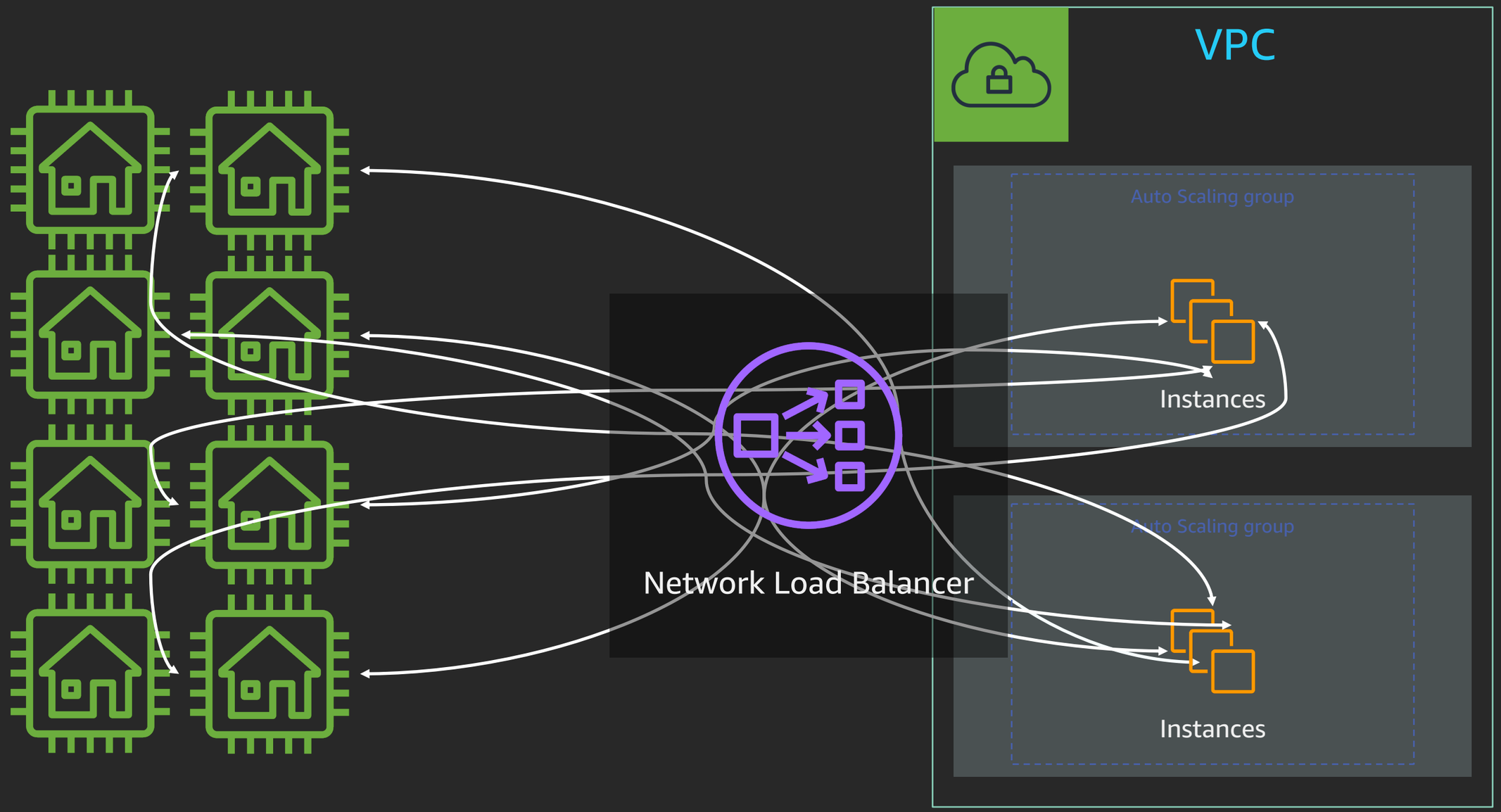


Long-lived connections on NLB

Client → NLB → Targets



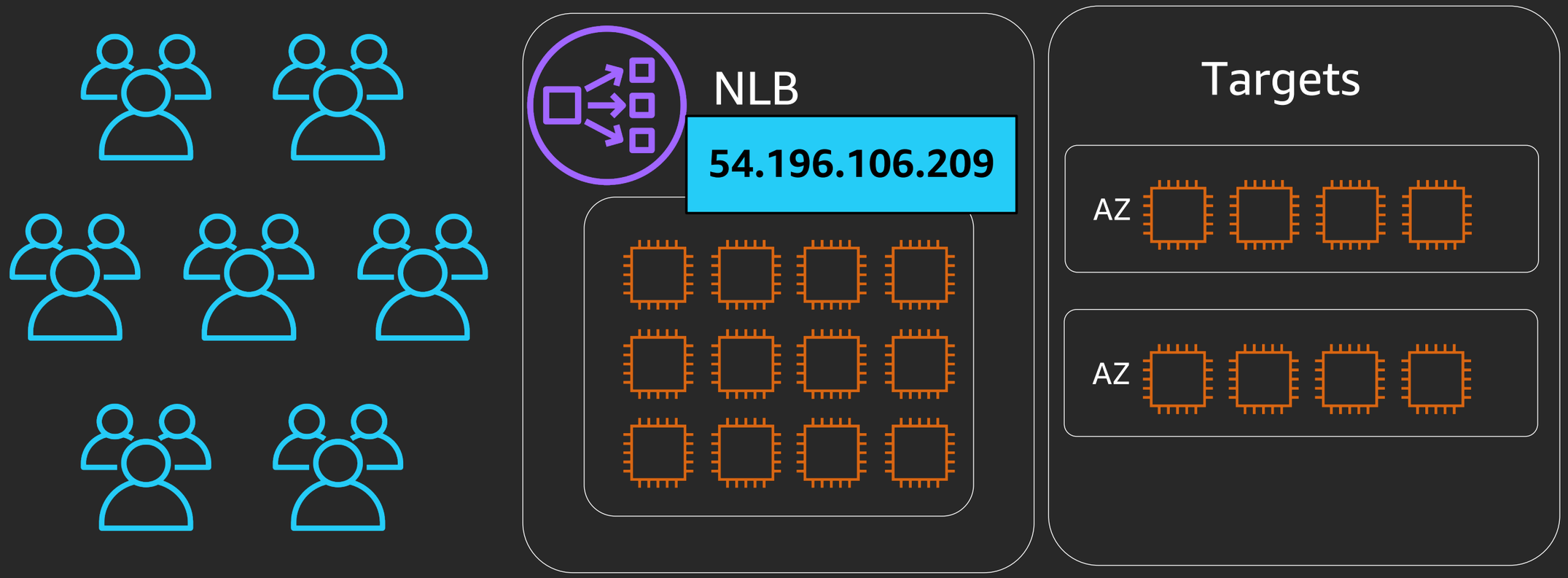
NLB: Device communications / IoT



NLB high performance with transparent scaling

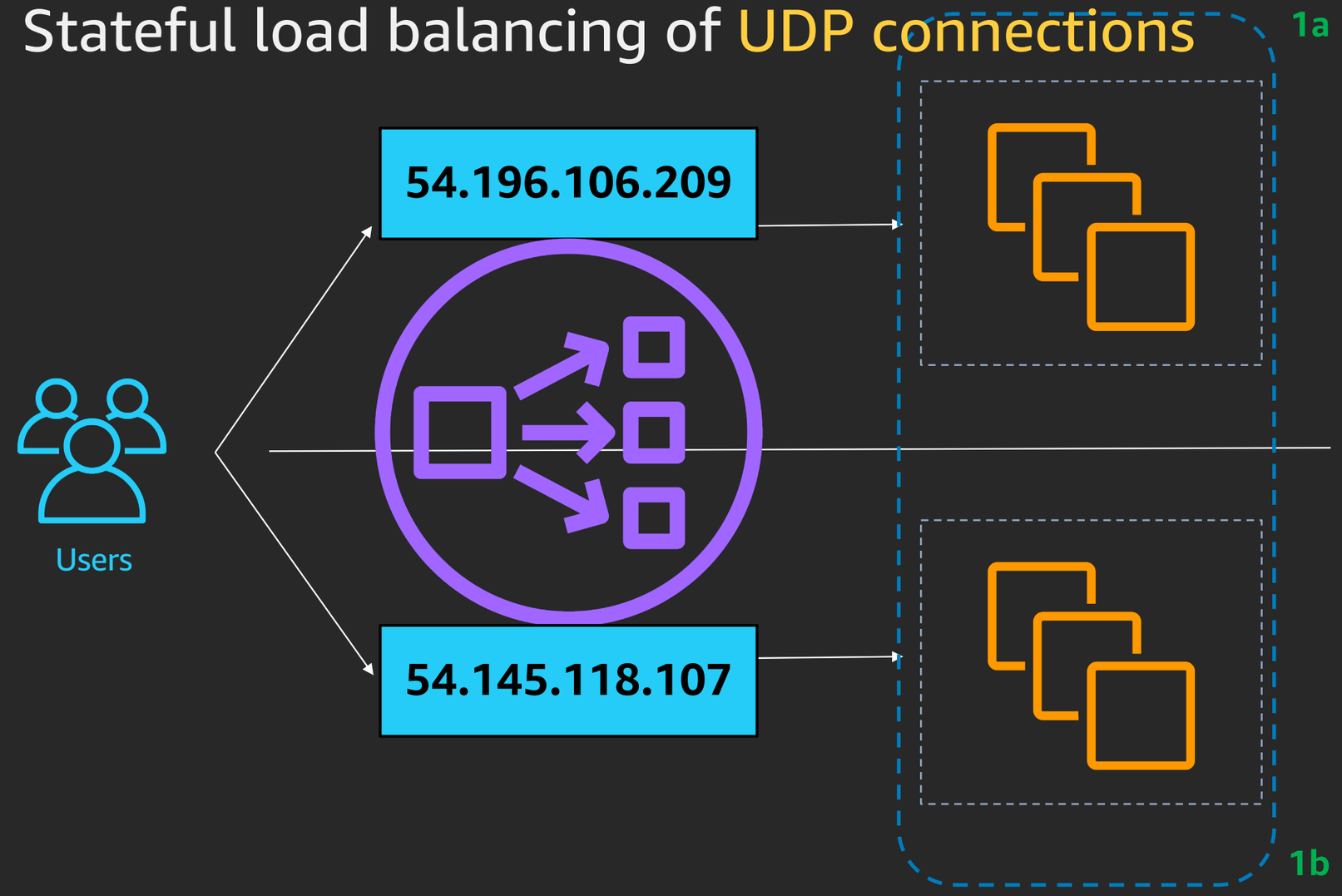
No DNS changes

Established connections uninterrupted



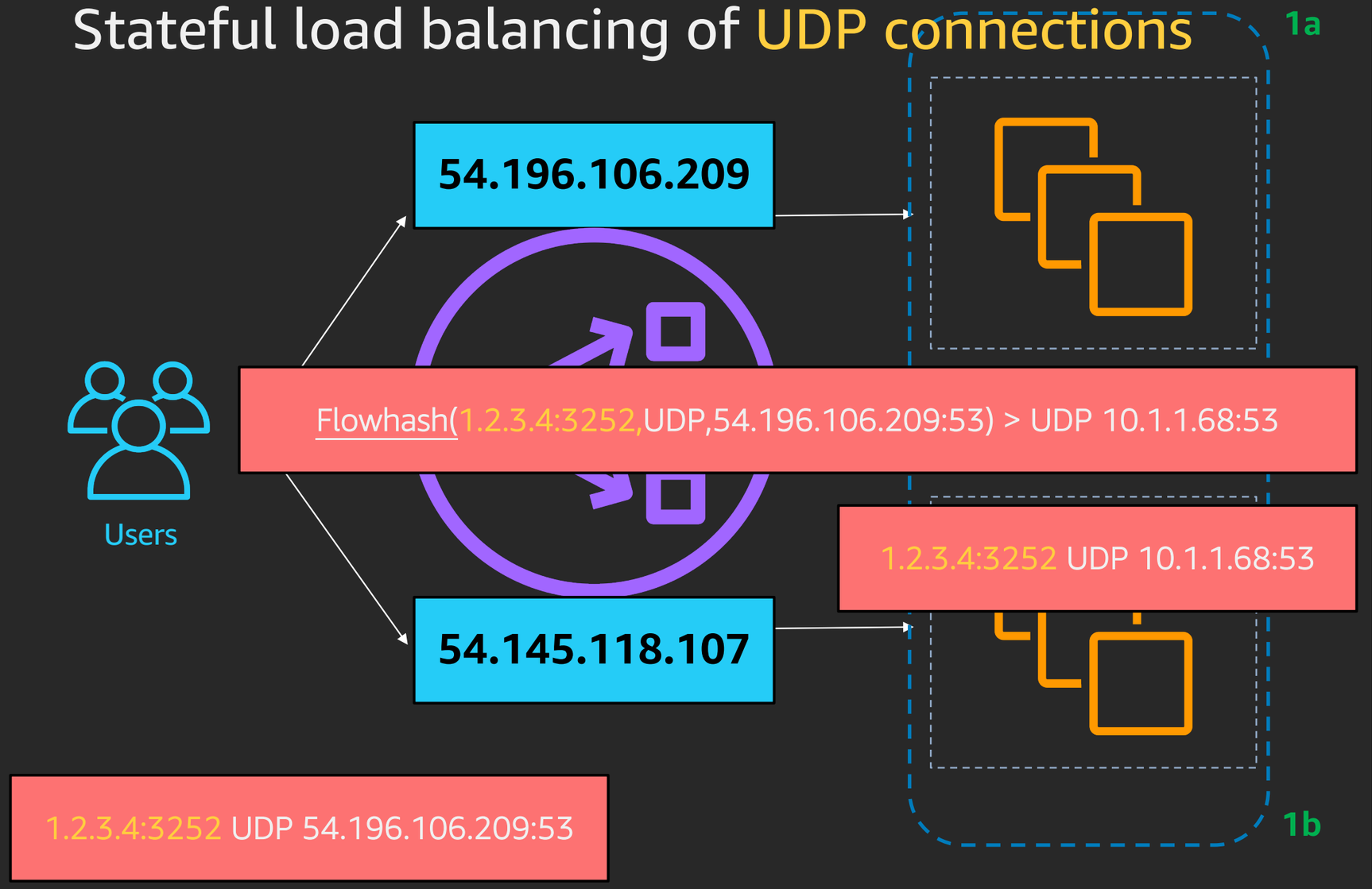
UDP listeners on Network Load Balancer

Stateful load balancing of UDP connections

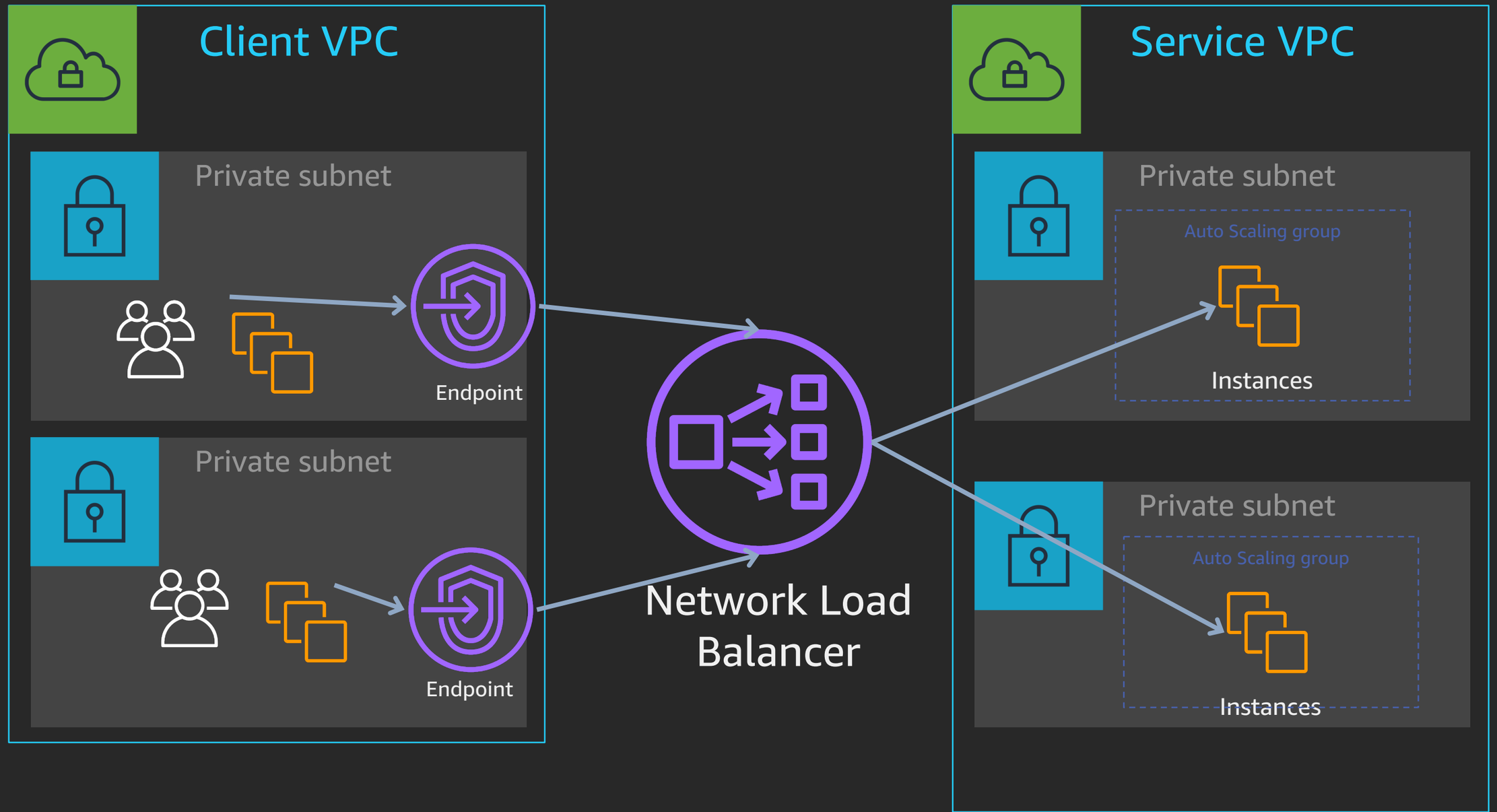


UDP listeners on Network Load Balancer

Stateful load balancing of UDP connections



NLB: Log ingestion via AWS PrivateLink endpoints

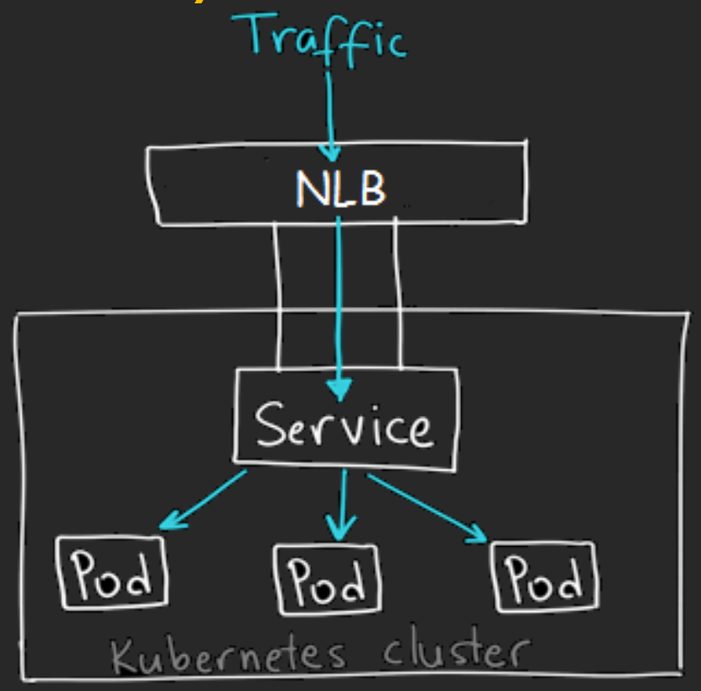


NLB as a load balancer in Kubernetes

- Use Network Load Balancer to expose Kubernetes services outside the cluster
- Each service gets its own IP address
- NLB support currently in beta (as of Kubernetes v1.15)
- UDP support in the works

To use a Network Load Balancer, use the annotation service.beta.kubernetes.io/aws-load-balancer-type with the value set to nlb.

```
metadata:
  name: my-service
  annotations:
    service.beta.kubernetes.io/aws-load-balancer-type: "nlb"
```



https://kubernetes.io/docs/concepts/services-networking/service/#aws-nlb-support

NLB recap

Client → NLB → Targets

- Logically, NLB is like ALB without the HTTP layer-7 rules
- NLBs have static IPs, which you can use your own EIPs for
- NLBs preserve source IP to your targets
- NLBs route via flow-hashing
- NLBs use AWS Hyperplane, built for long-lived connections
- NLB scales instantly to support virtually any workload
- NLB supports TCP, UDP & TCP_UDP
- NLB can expose your Kubernetes services

ELB health checks

© 2019, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Elastic Load Balancing: Health checks

"Elastic Load Balancing puts IP addresses in DNS that are adequately scaled, and healthy."

Implicitly detect problems from within the data plane

Acts as an explicit confirmation that the target can accept new traffic

Monitor in CloudWatch metrics at load balancer and Target Group

Elastic Load Balancing: Health checks

External (Route 53)

Checks each ELB IP from all over the world

Target health checks

This checks your target's health from the ELB

What does passing an ELB health check mean?

"Send me new requests!"

Elastic Load Balancing: Route 53 health checks

Route 53 is an external view of ELB health

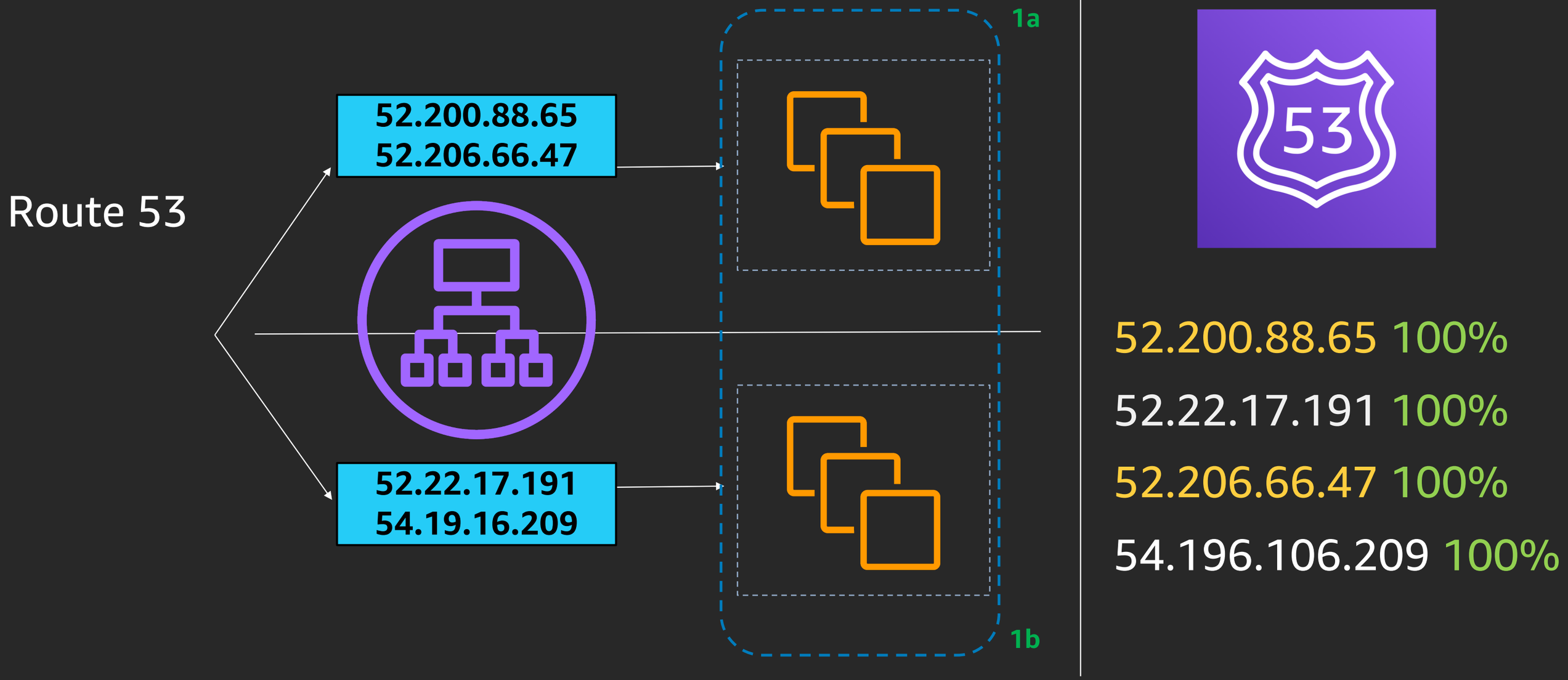
Includes cases where ELB IP is healthy but targets are not

- All targets in at least 1 Target Group are unhealthy (ALB & NLB)
- All backends unhealthy (CLB)
- All Target Groups are empty

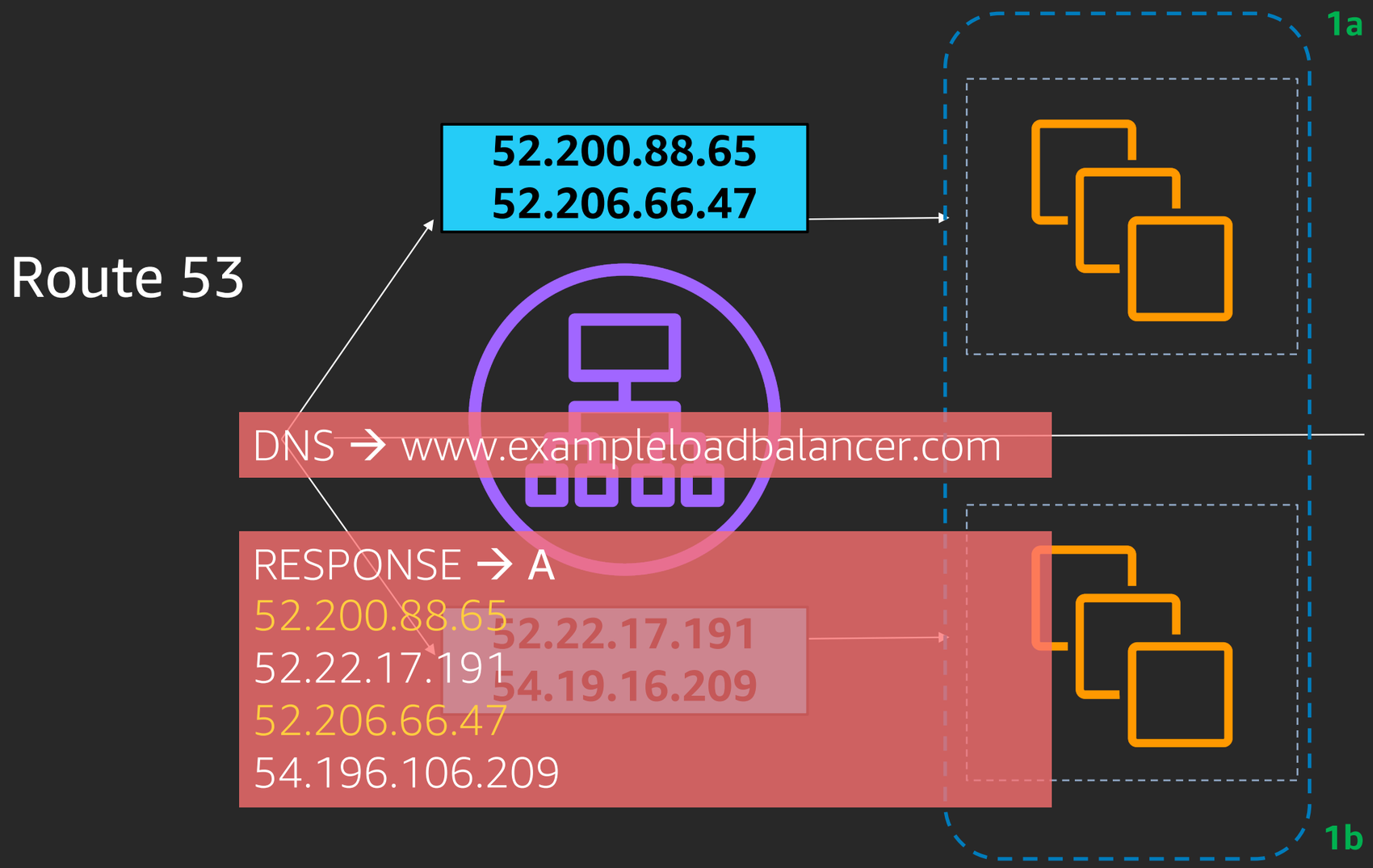
IPs failing Route 53 health checks are removed from DNS responses

Using cross-zone increases availability

ELB's Route 53 health check: External to ELB



ELB's Route 53 health check: External to ELB



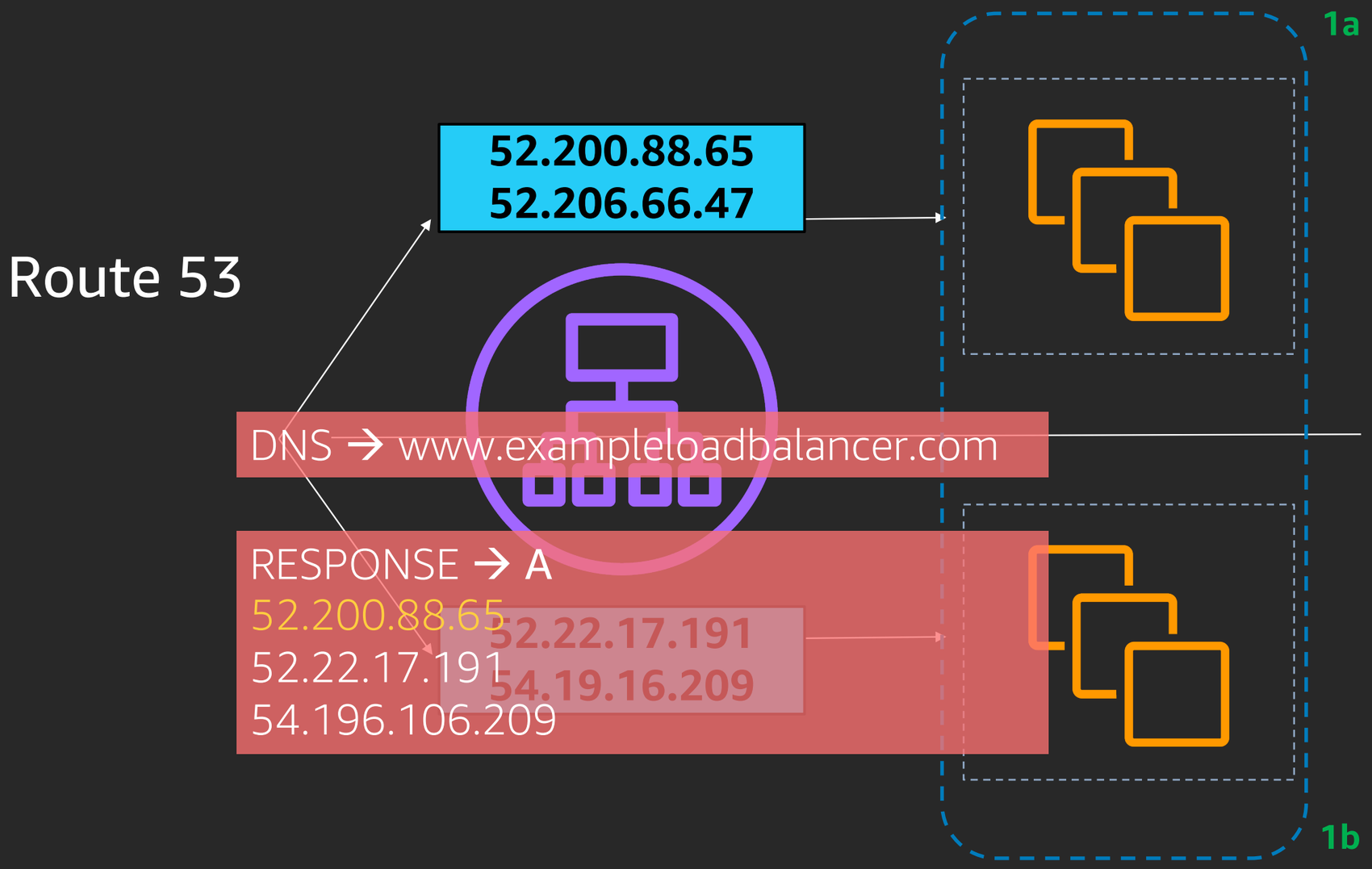
52.200.88.65 100%

52.22.17.191 100%

52.206.66.47 100%

54.196.106.209 100%

ELB's Route 53 health check: External to ELB



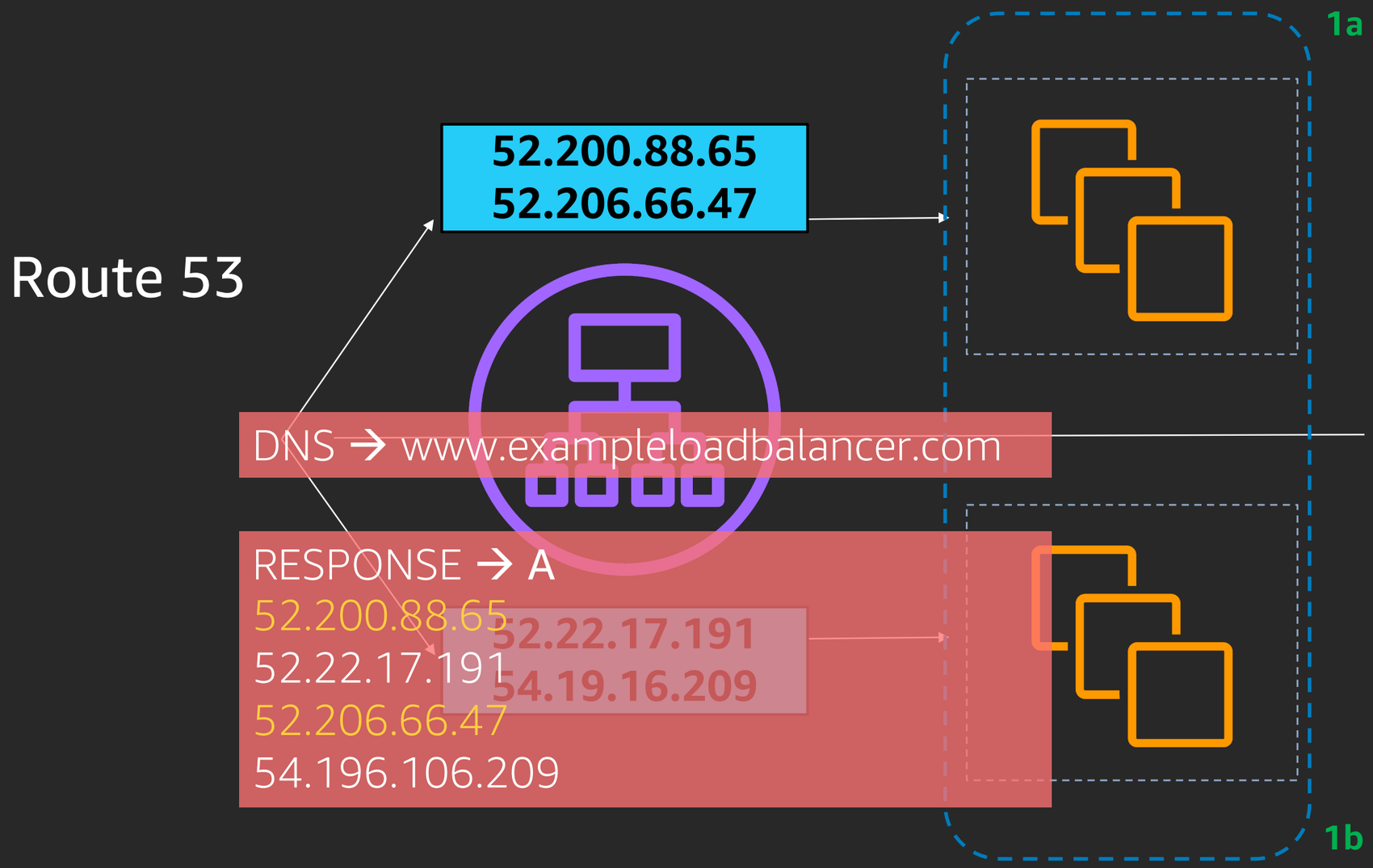
52.200.88.65 100%

52.22.17.191 100%

52.206.66.47 0%

54.196.106.209 100%

ELB's Route 53 health check: External to ELB



52.200.88.65 0%

52.22.17.191 0%

52.206.66.47 0%

54.196.106.209 0%

Elastic Load Balancing

Internal (load balancer target) health checks

ELB: Types of target health checks

TCP

- Simple health check, completes a TCP handshake
- Validates operating system is up and is listening on the configured target port

TLS/SSL

- Simple health check, completes a TLS negotiation
- Validates something can complete handshake

HTTP/HTTPS

- Can be deep or light, depending on your application
- Validates HTTP application server is up

ELB: Target health check configuration options

Interval: How often between health checks

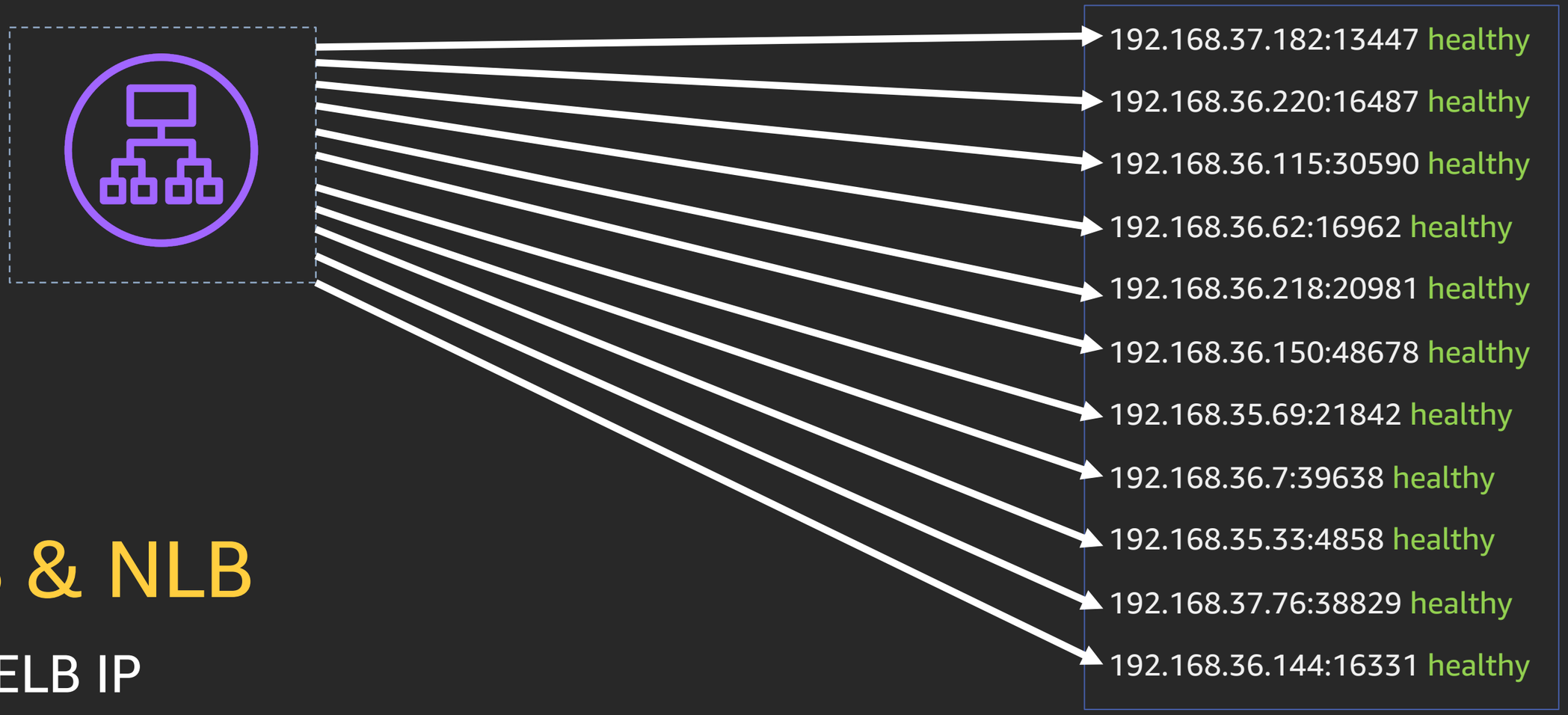
Timeout: How long we'll wait after sending the check for an answer

Healthy threshold: How many times in a row to count it as healthy

Unhealthy threshold: How many times in a row to count it as unhealthy

Type: HTTP(S)/TCP/TLS

Elastic Load Balancer: Target health checks

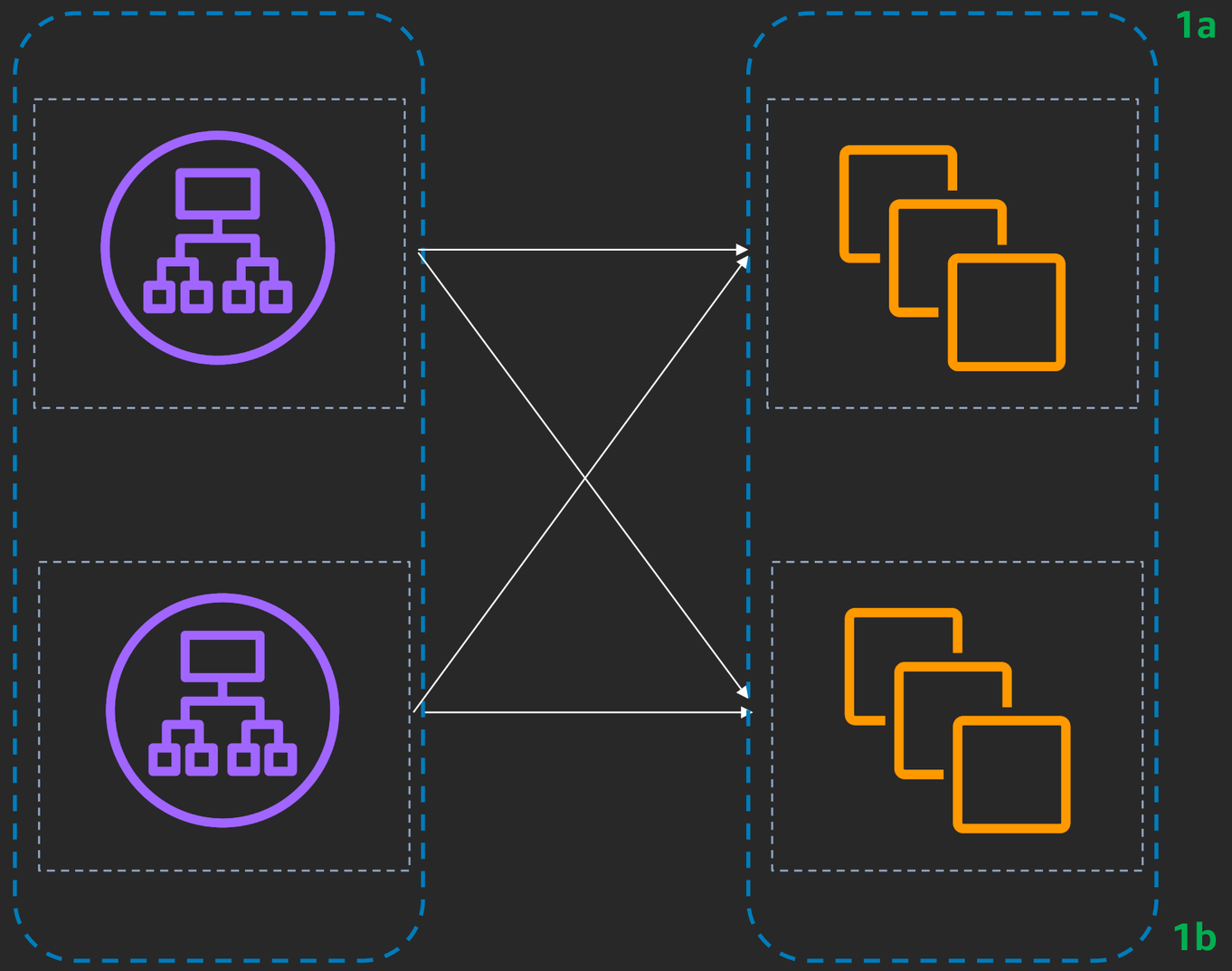


ALB & NLB

Each ELB IP

Independently health checks each target

Elastic Load Balancer: Target health checks



192.168.37.182:13447 healthy

192.168.36.220:16487 healthy

192.168.36.115:30590 healthy

192.168.36.62:16962 healthy

192.168.36.218:20981 healthy

192.168.36.150:48678 healthy

192.168.35.69:21842 healthy

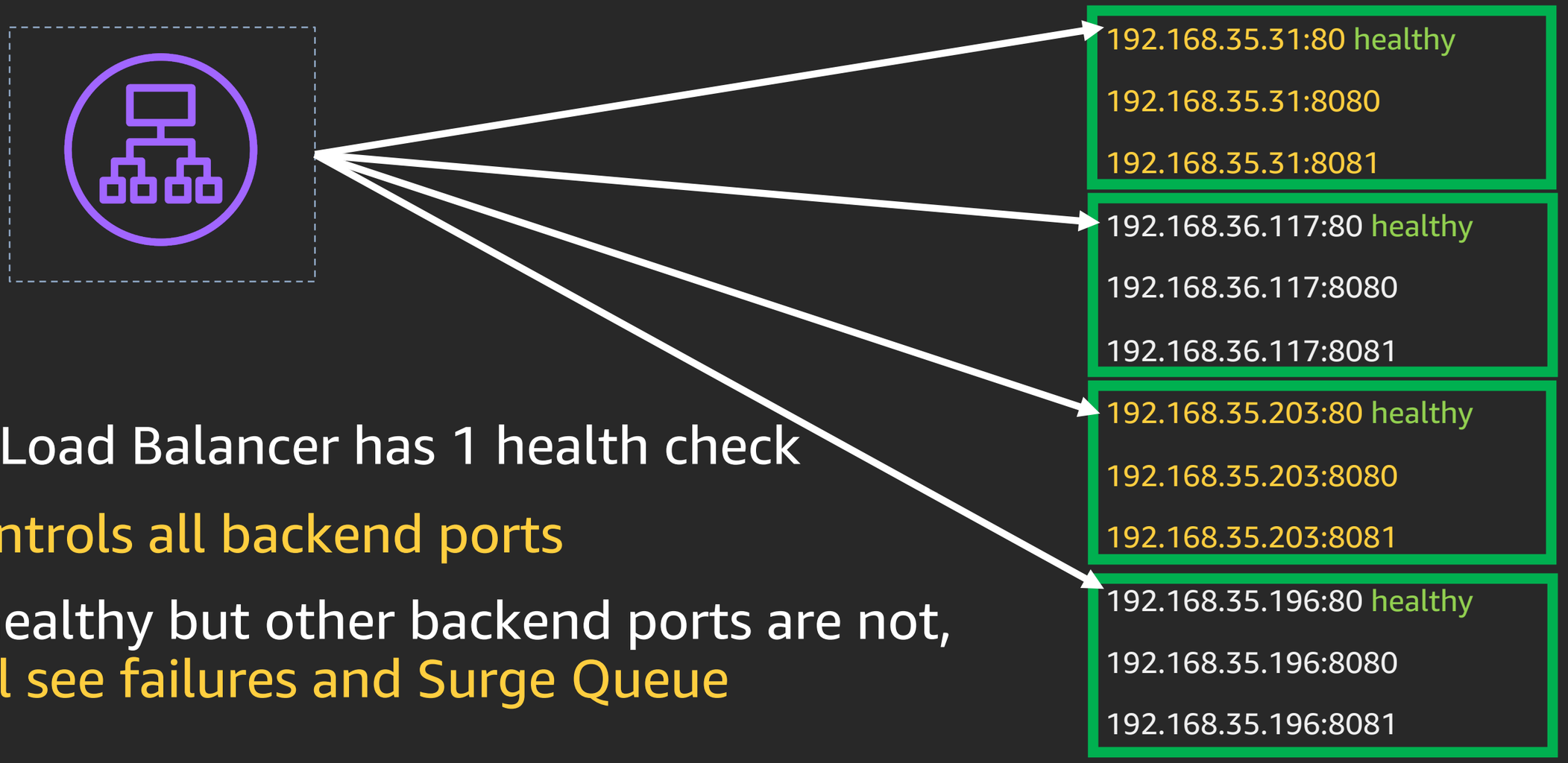
192.168.36.7:39638 healthy

192.168.35.33:4858 healthy

192.168.37.76:38829 healthy

192.168.36.144:16331 healthy

Classic Load Balancer: Target health checks

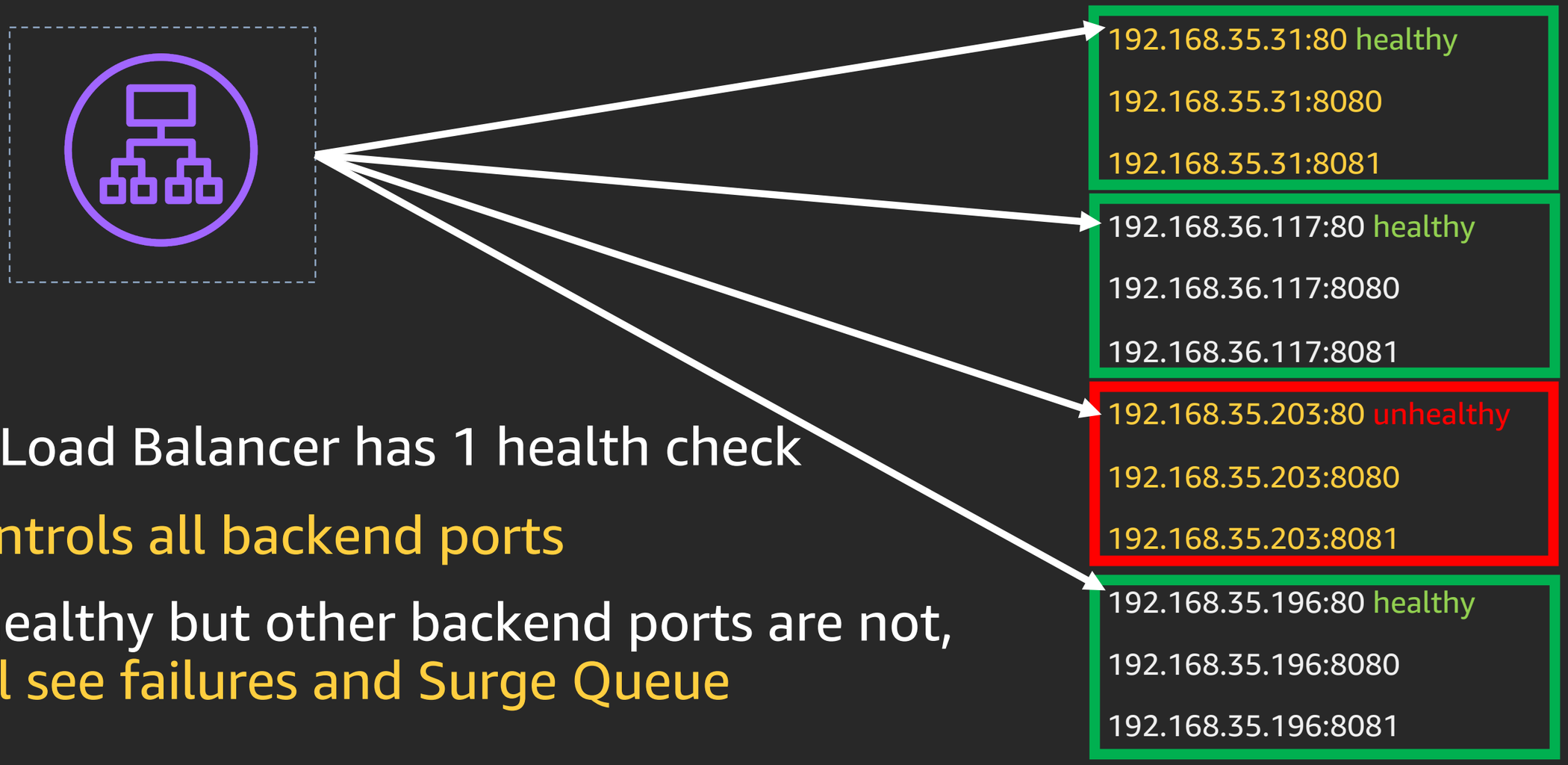


Classic Load Balancer has 1 health check

This controls all backend ports

If it is healthy but other backend ports are not,
you will see failures and Surge Queue

Classic Load Balancer: Target health checks

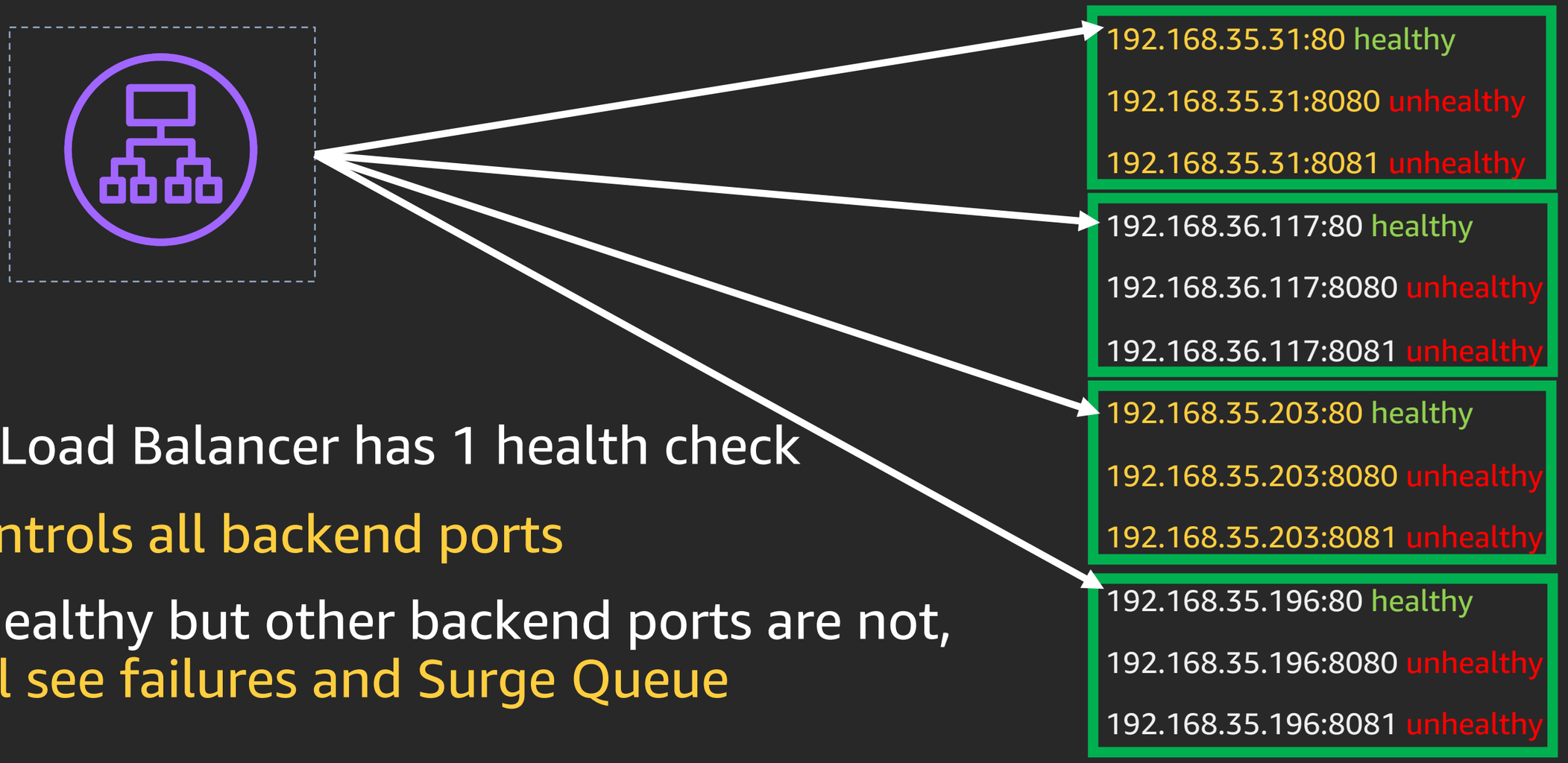


Classic Load Balancer has 1 health check

This controls all backend ports

If it is healthy but other backend ports are not,
you will see failures and Surge Queue

Classic Load Balancer: Target health checks



Classic Load Balancer has 1 health check

This controls all backend ports

If it is healthy but other backend ports are not, you will see failures and Surge Queue

Elastic Load Balancer: Target health check summary

Independent health checks from each ELB IP to each target

- Includes cross-zone (when enabled)

Health check state emits to metrics

- CLB only 1 health check per load balancer

NLB & ALB directly check the target's port by default

- You can have a single port for the whole Target Group
- *Configurable per Target Group and per target

Elastic Load Balancing

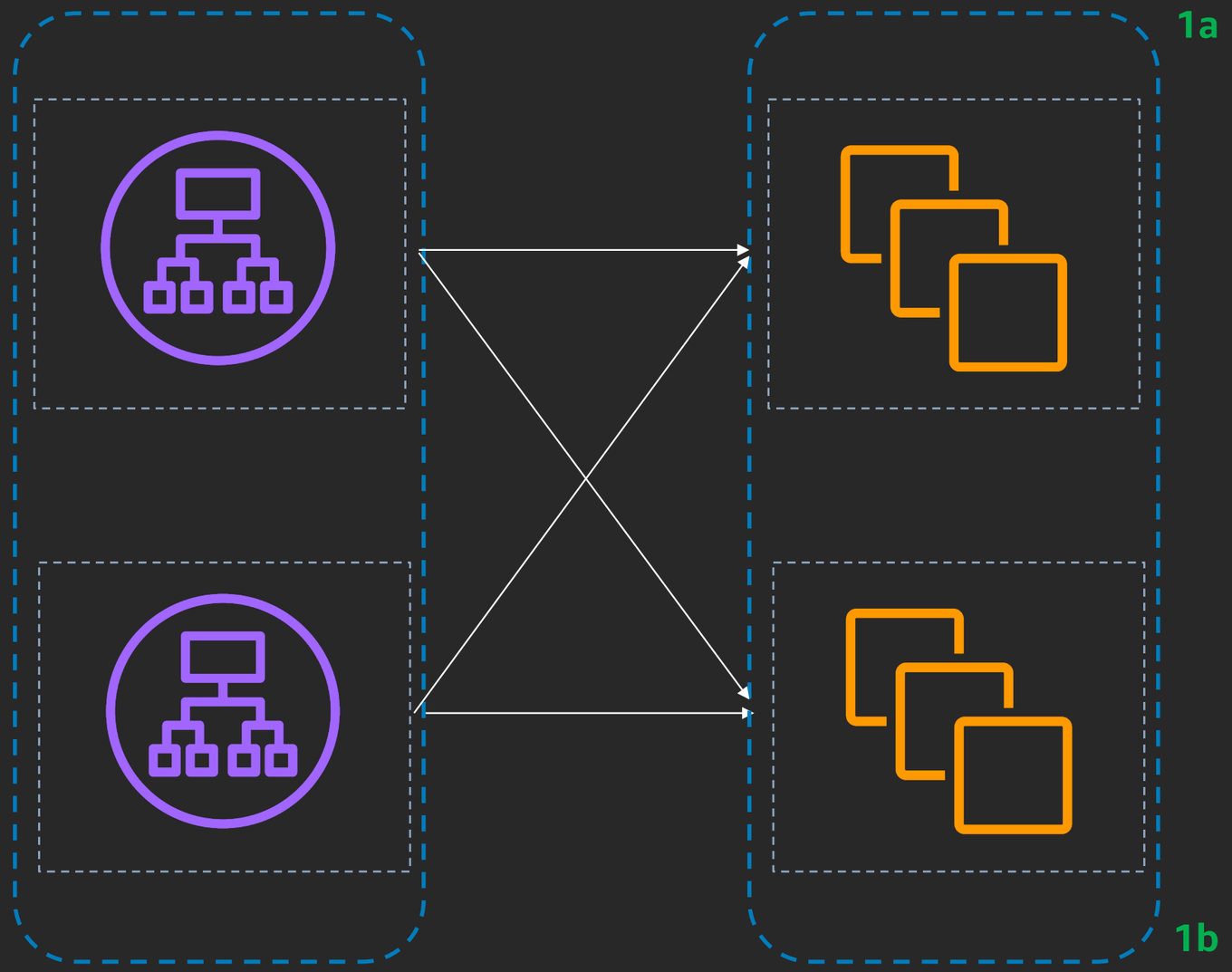
Handling failures & configuring ELB health checks

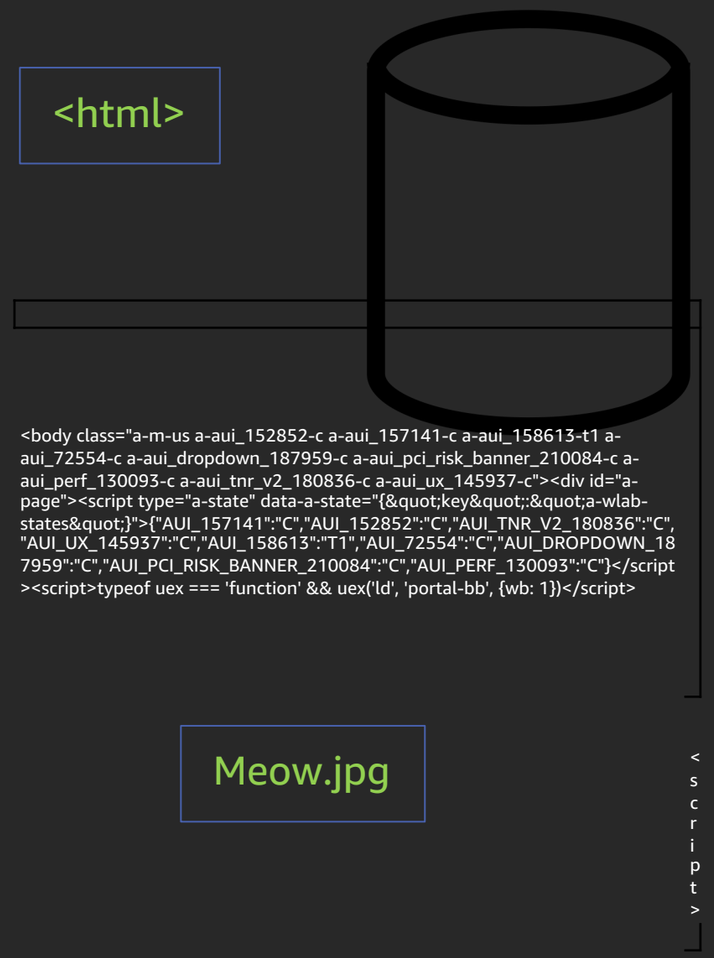
Handling failures

"We all want to build high-availability systems that are up 100% of the time, but the next failure will happen!"

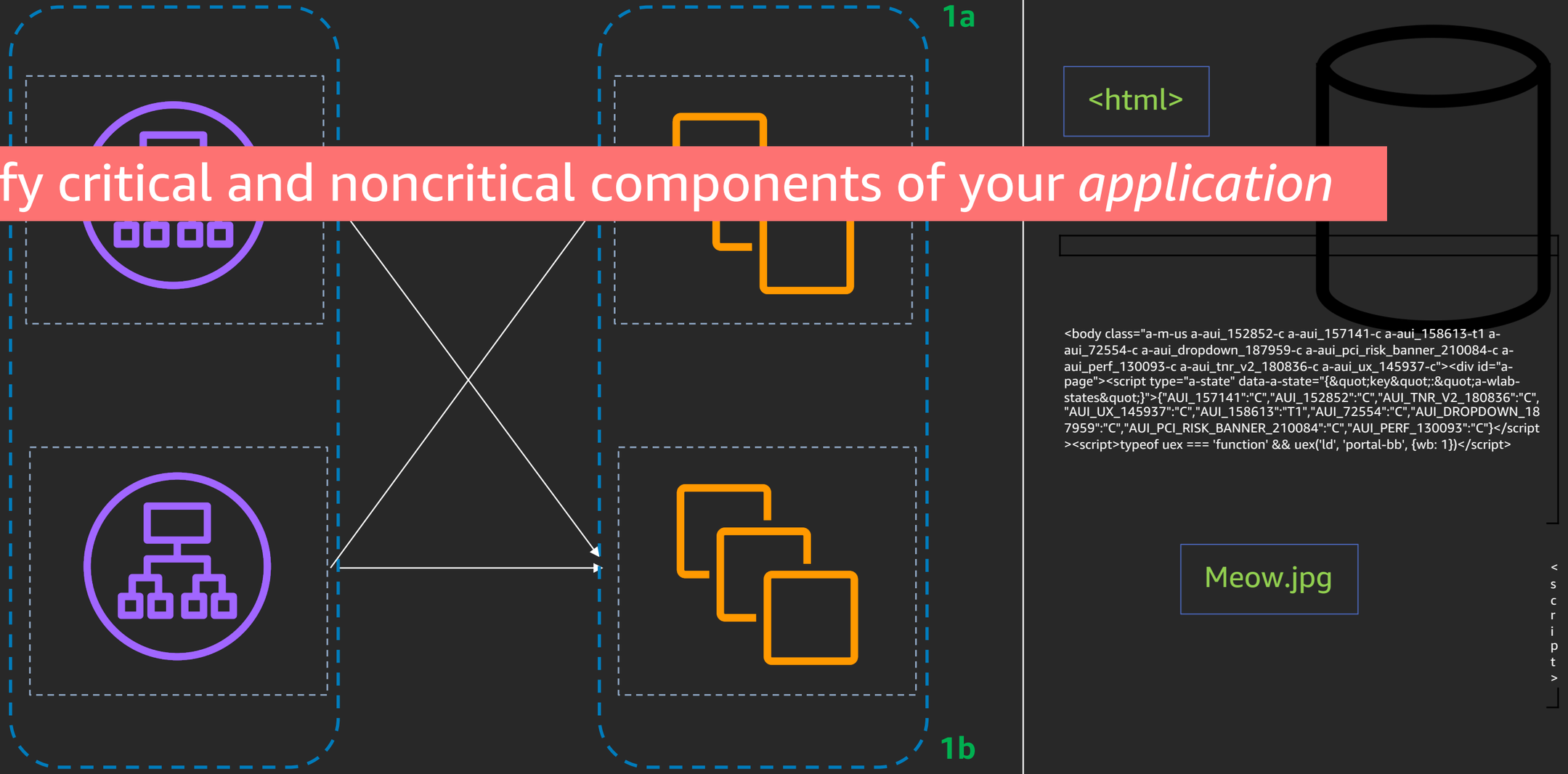
– Werner Vogels, AWS CTO

Handling failures by ensuring critical components



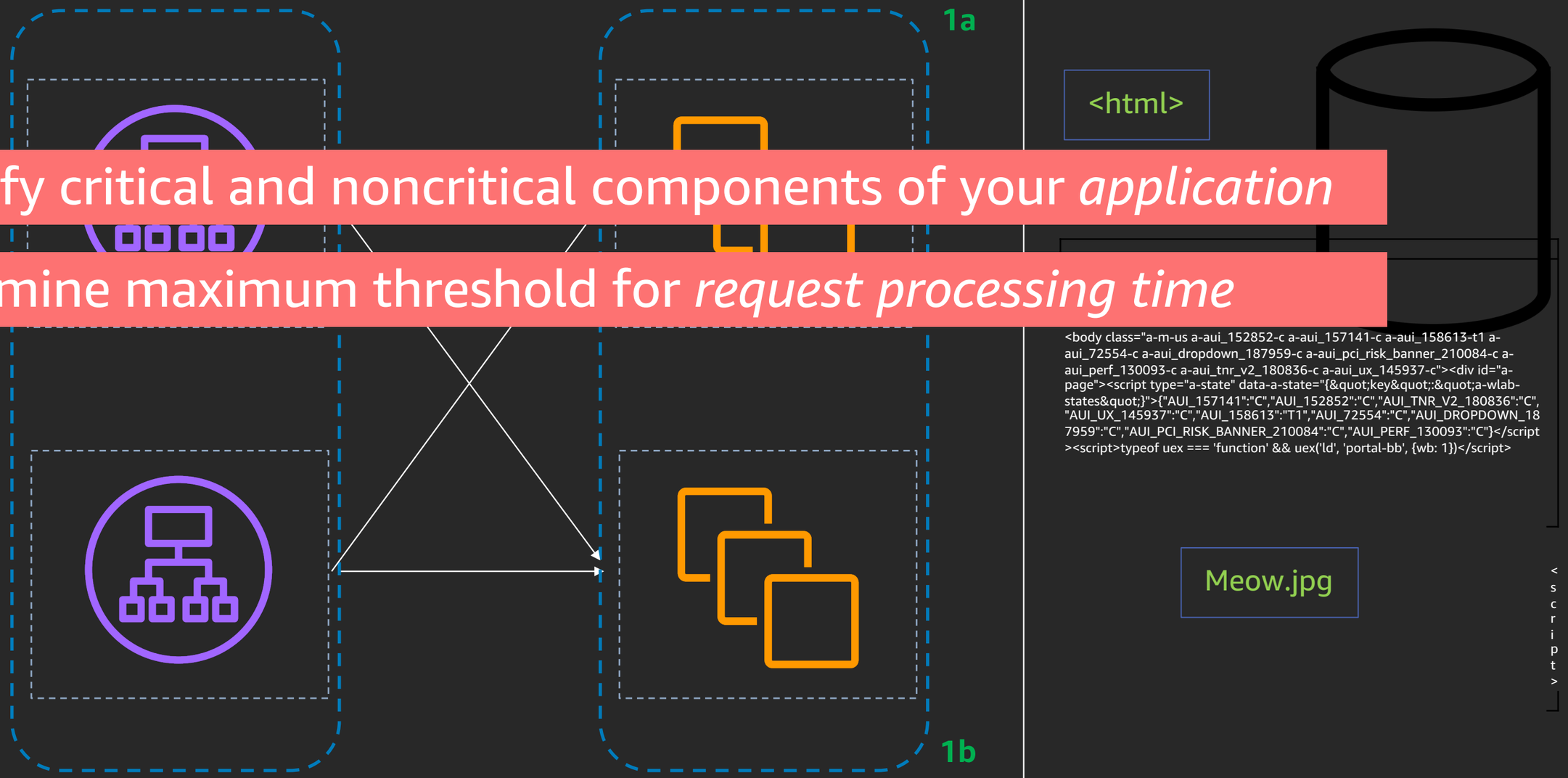


Handling failures by ensuring critical components



Identify critical and noncritical components of your *application*

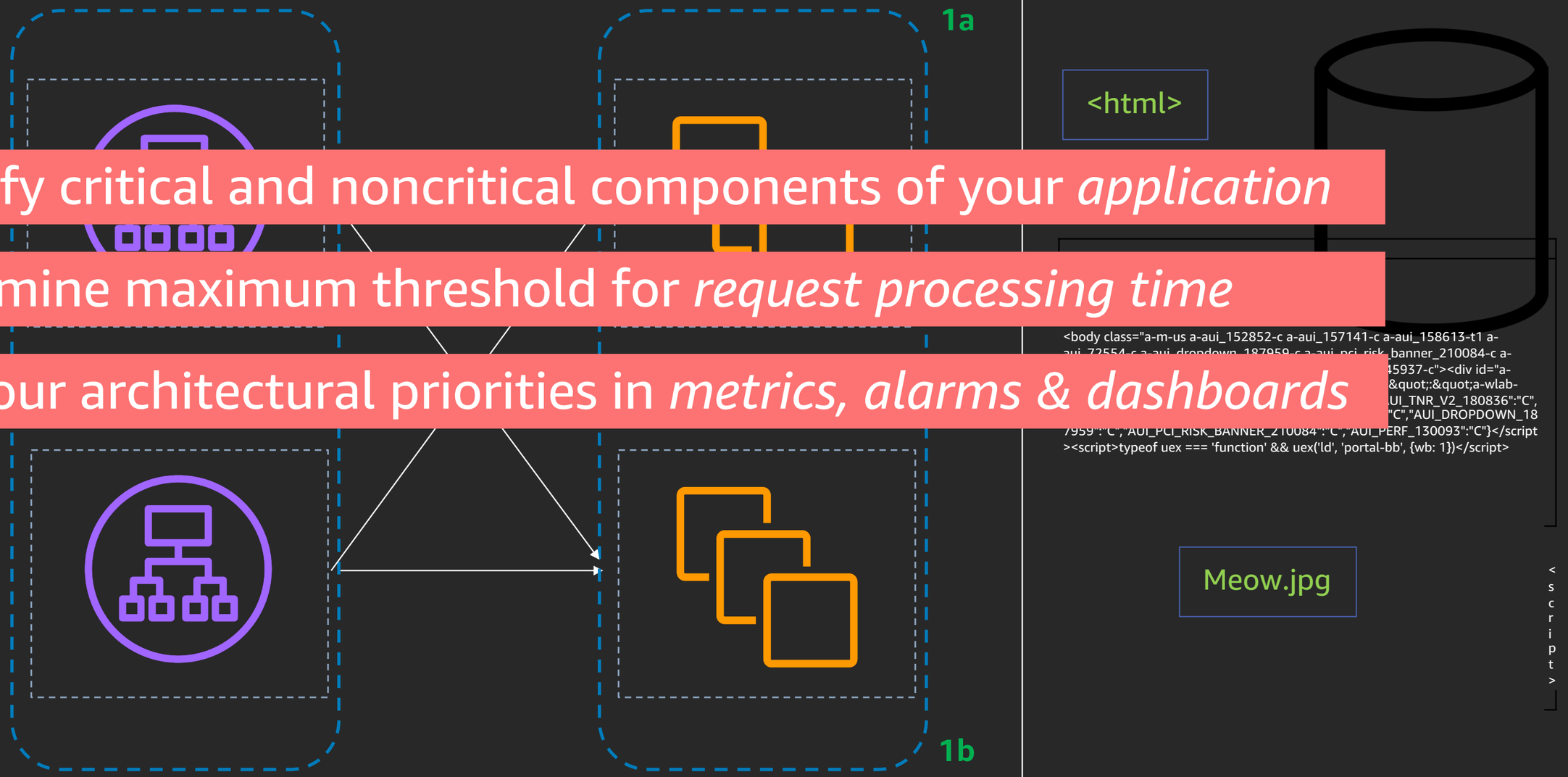
Handling failures by ensuring critical components



Identify critical and noncritical components of your *application*

Determine maximum threshold for *request processing time*

Handling failures by ensuring critical components

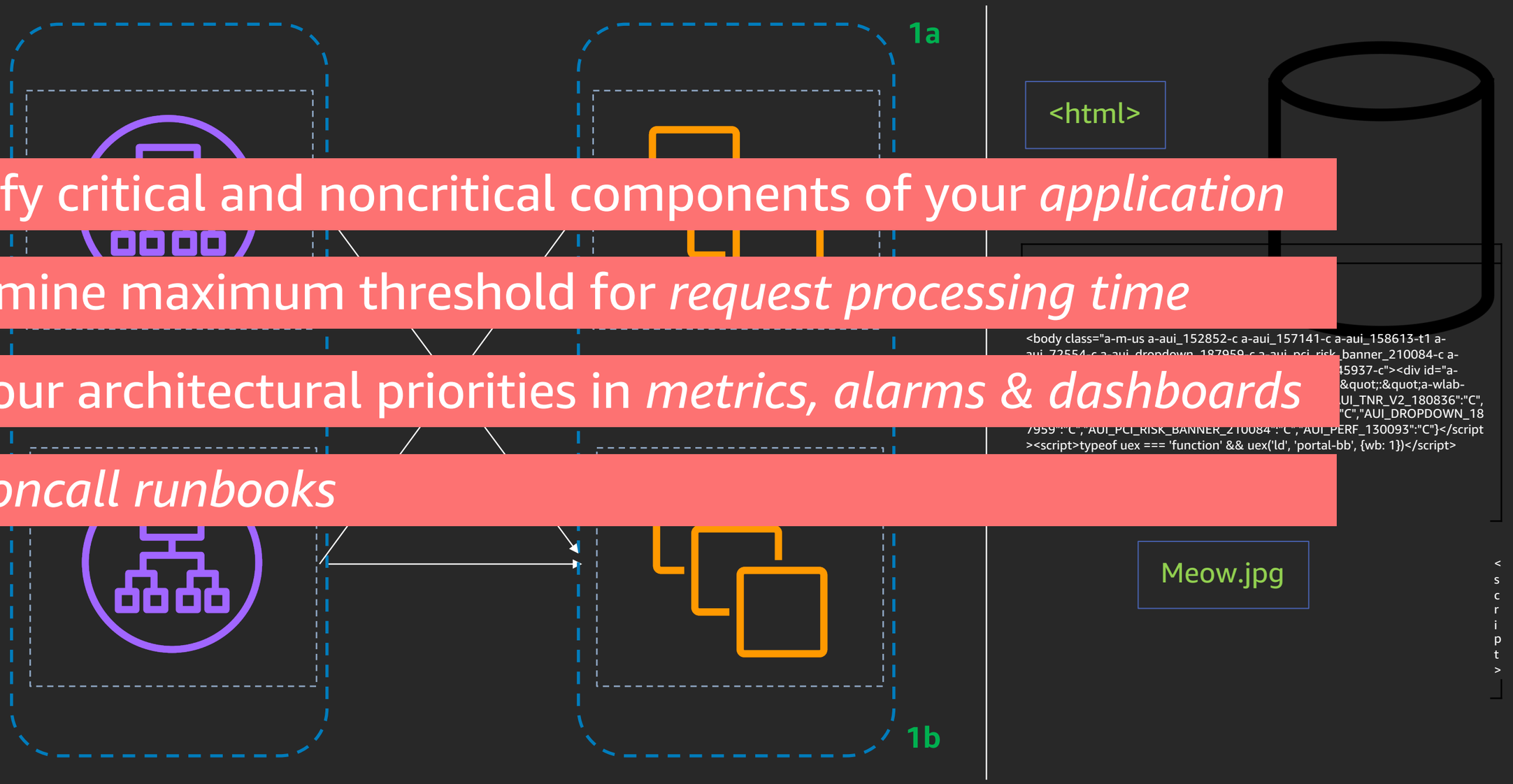


Identify critical and noncritical components of your *application*

Determine maximum threshold for *request processing time*

Put your architectural priorities in *metrics, alarms & dashboards*

Handling failures by ensuring critical components



Identify critical and noncritical components of your *application*

Determine maximum threshold for *request processing time*

Put your architectural priorities in *metrics, alarms & dashboards*

*and *oncall runbooks*

ELB health checks: Cascading failure prevention

Overloaded, impaired, or failing targets – what do you want to happen?

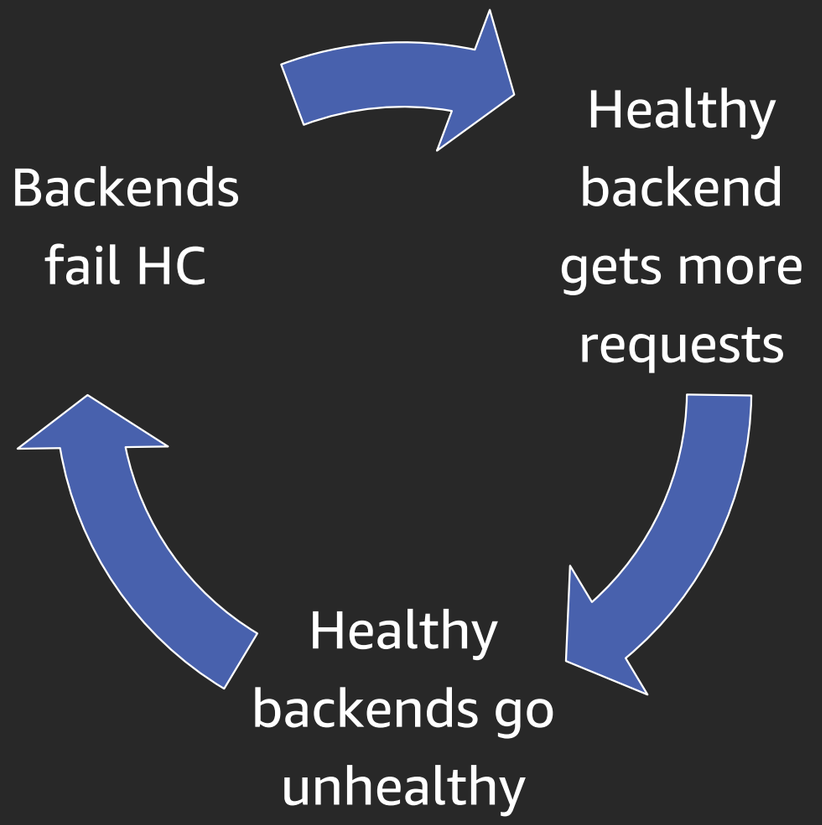
Brownout vs. blackout

Backend death spiral

Mitigations: Scale, throttle, relax health check

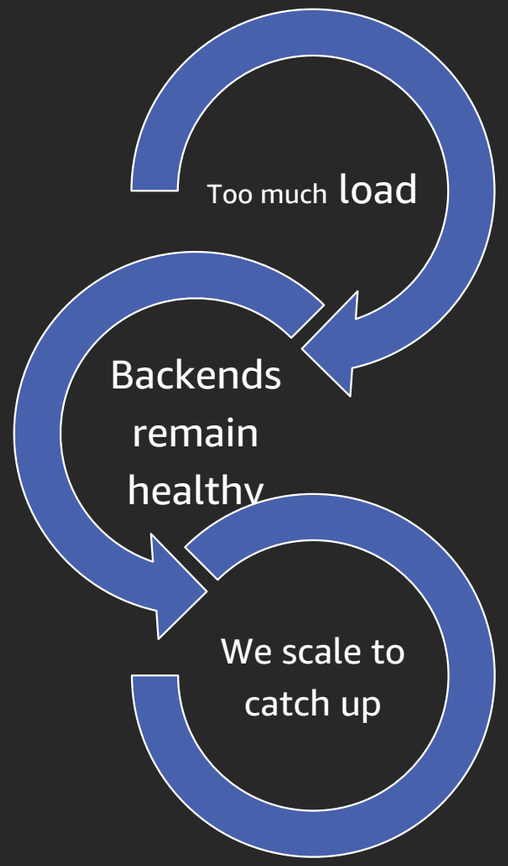
Brownout: Live with underperformance

Blackout: Fail away from that ELB



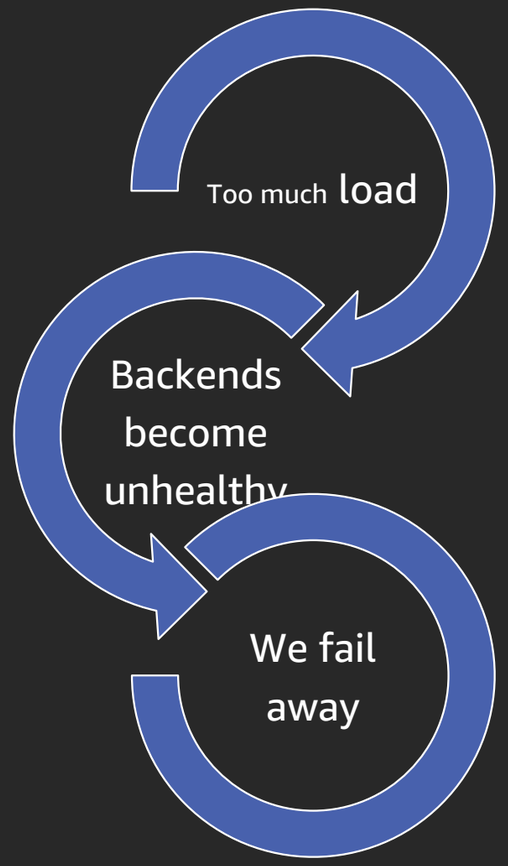
Brownout

I want my backends to try as best as they can. Recovery will be when traffic reduces or scaling targets catch up to load.

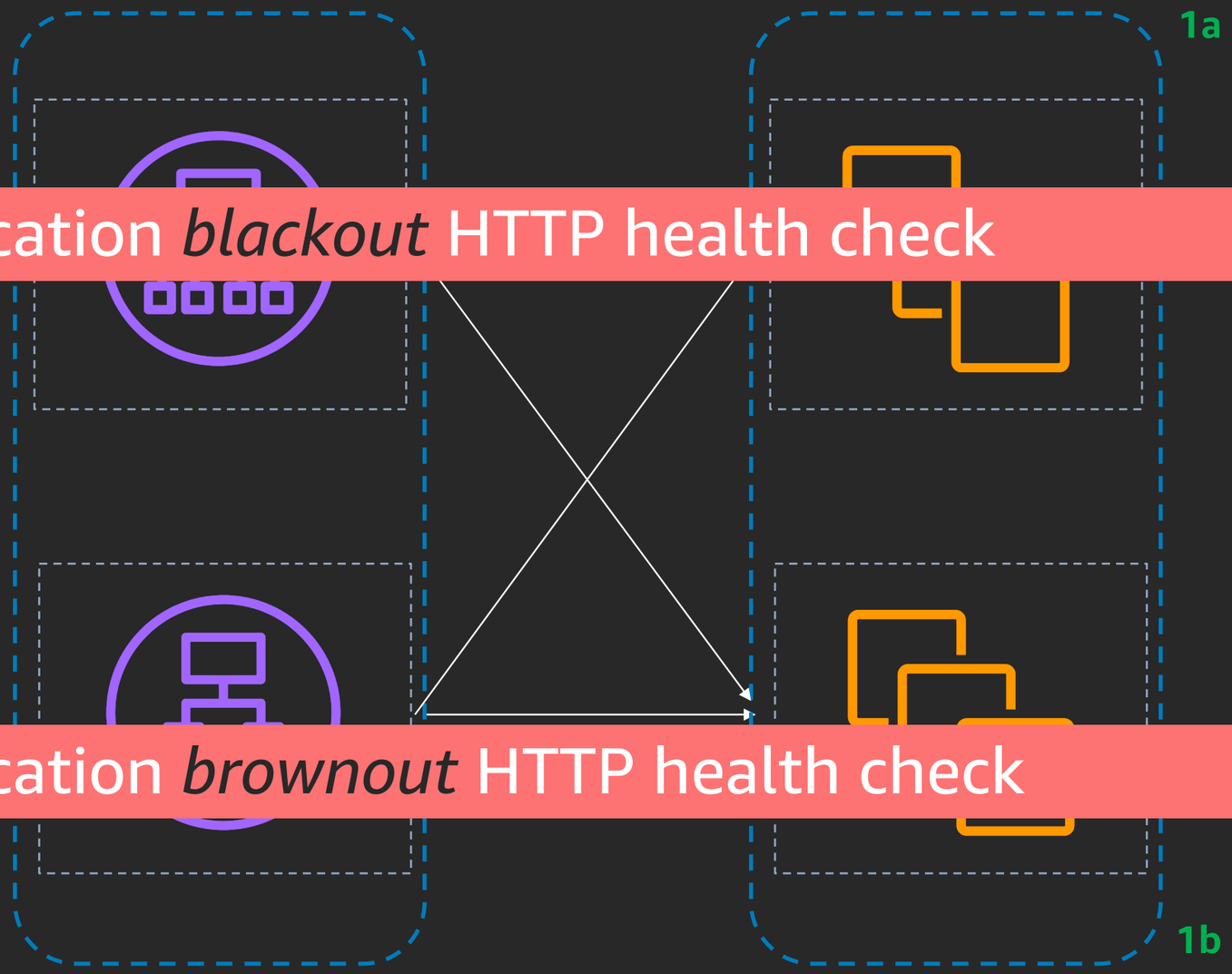


Blackout

I want to fail away…
or terminate any failing resources



Health check configuration: General guidance



Application *blackout* HTTP health check

Type: HTTP

Interval: 5

Timeout: 2

Healthy threshold: 6

Unhealthy threshold: 4

Application *brownout* HTTP health check

Type: HTTP

Interval: 30

Timeout: 9

Healthy threshold: 4

Unhealthy threshold: 3

Elastic Load Balancing

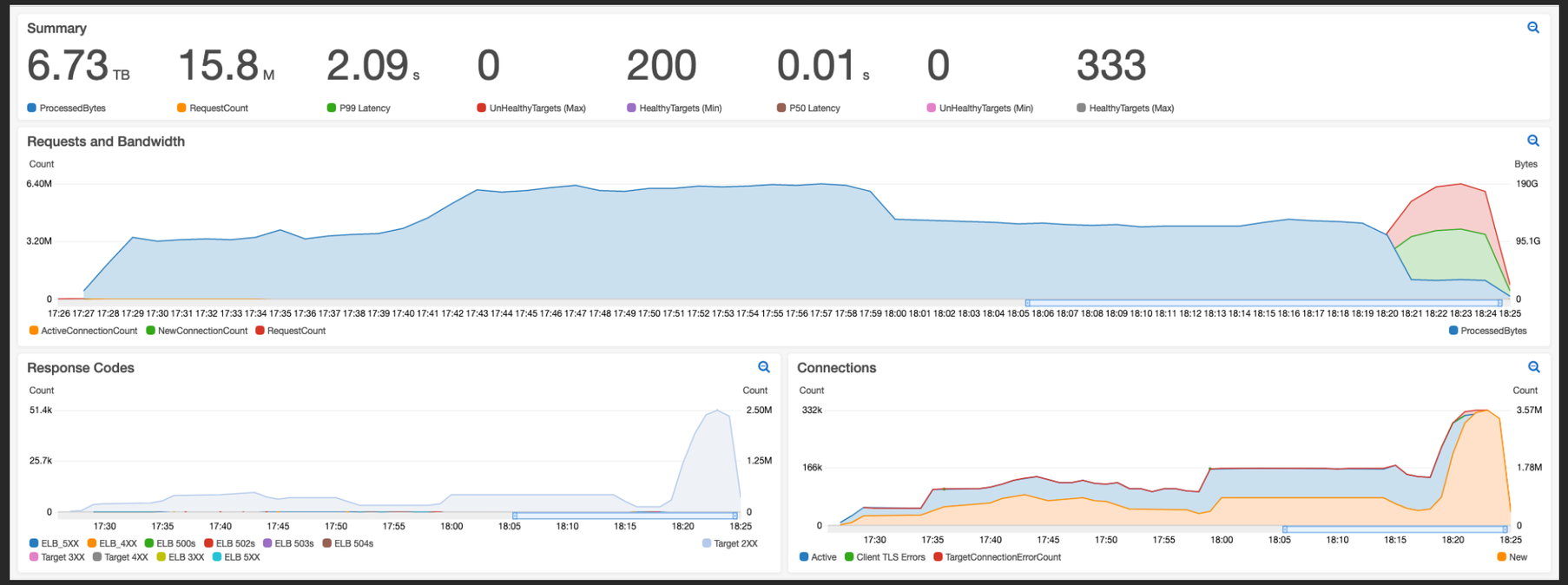
ELB CloudWatch metrics

ELB CloudWatch metrics: Overview

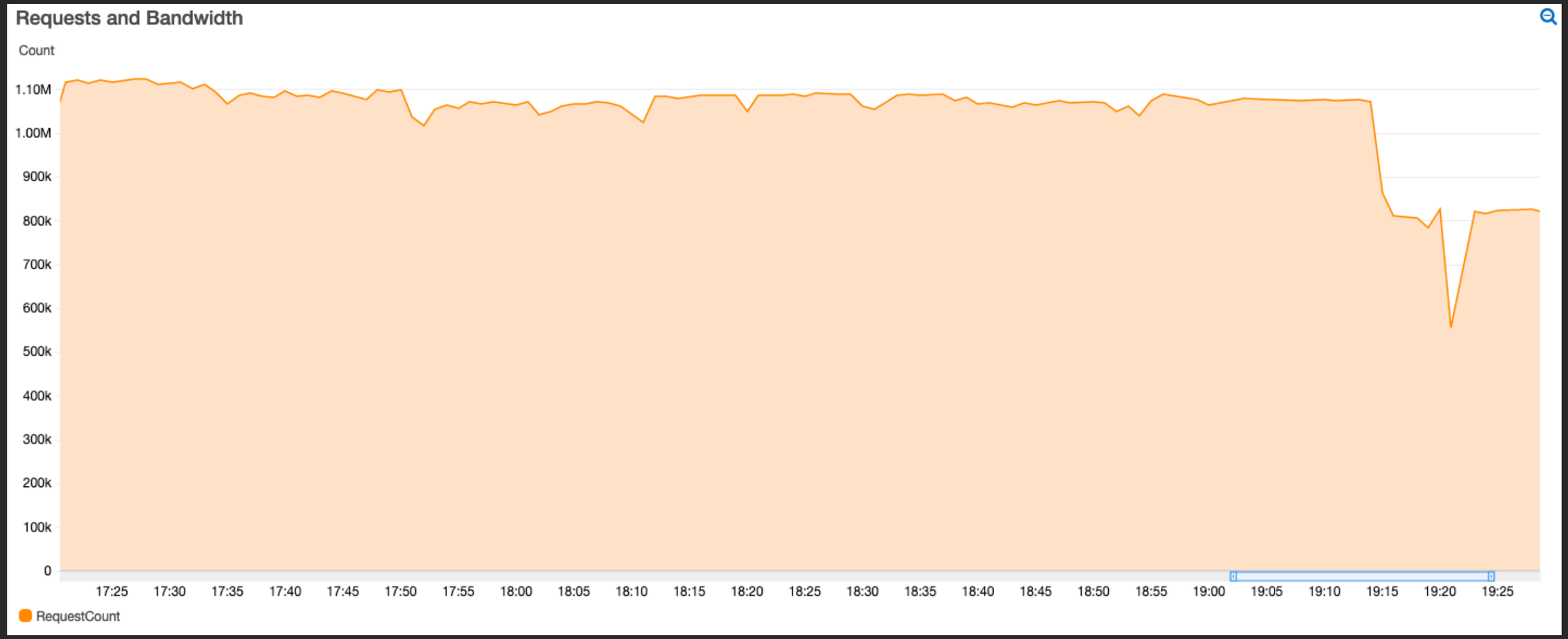
Workload metrics

ELB metrics

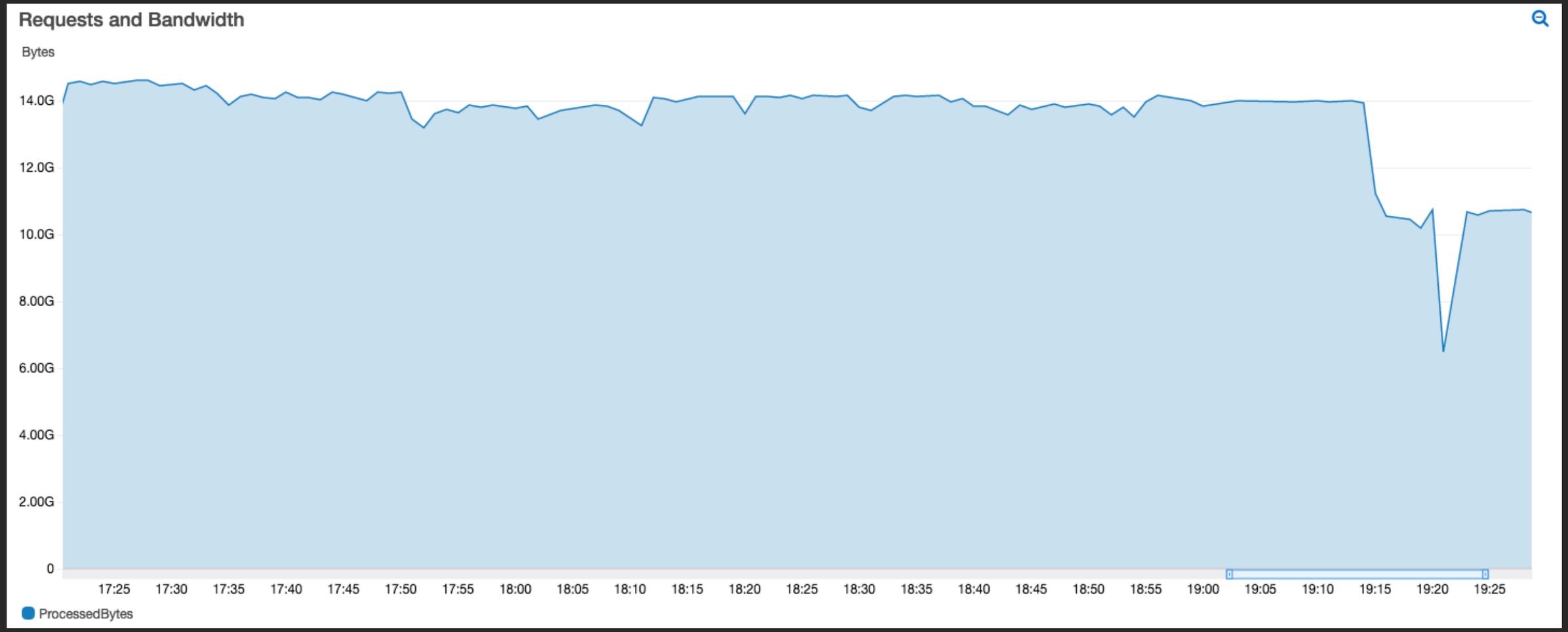
Target health metrics



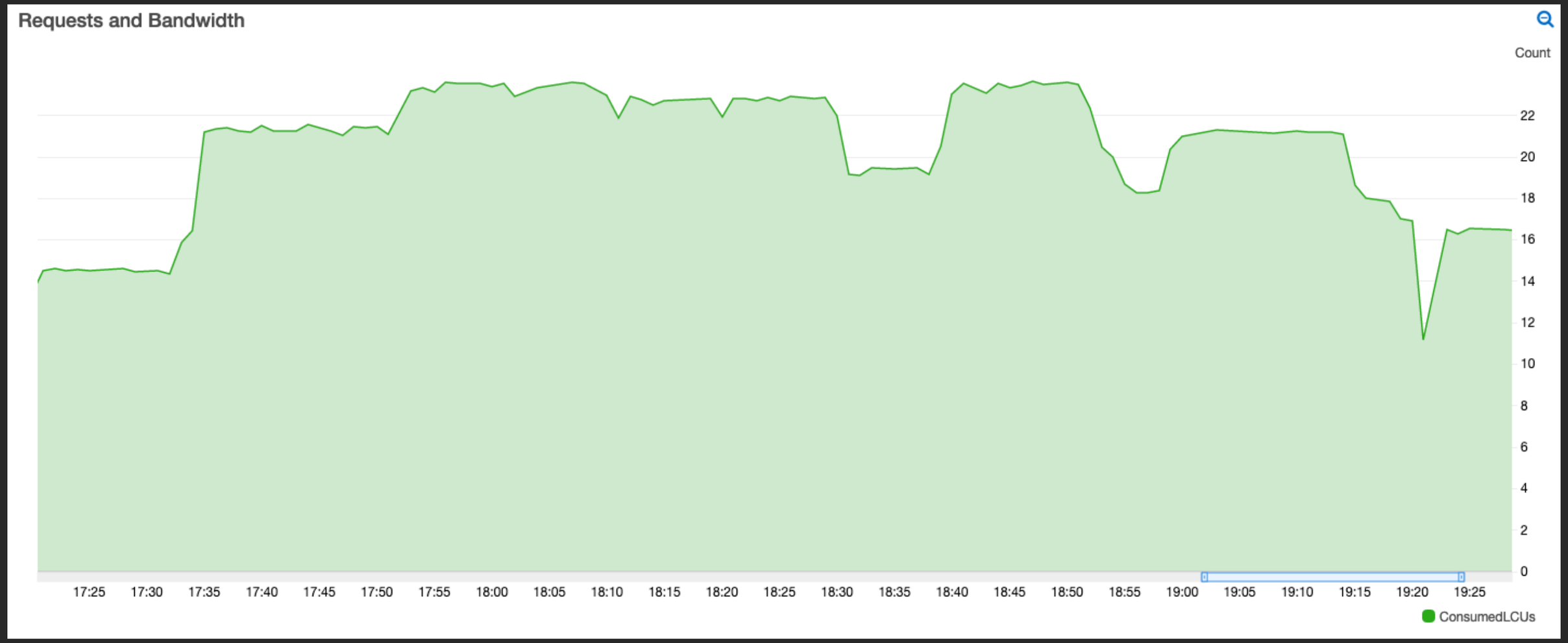
Workload metric: RequestCount



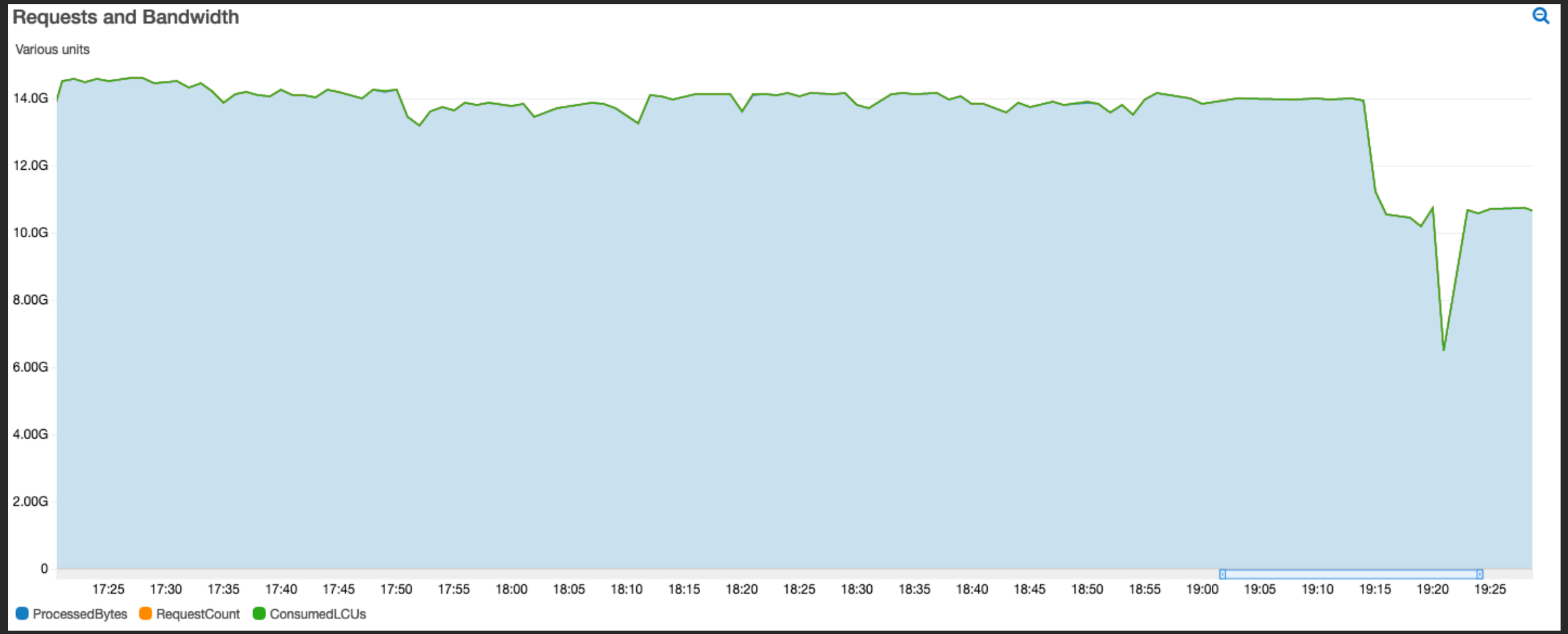
Workload metric: ProcessedBytes



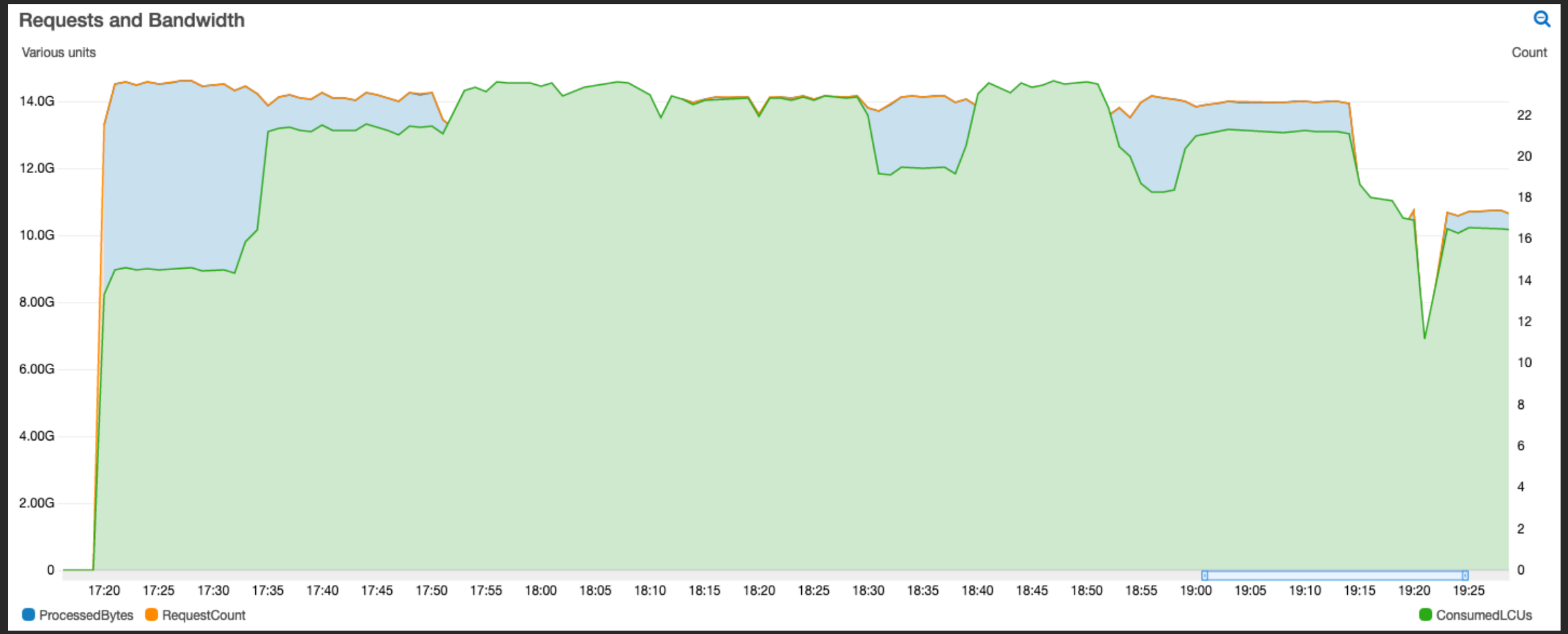
Workload metric: ConsumedLCU



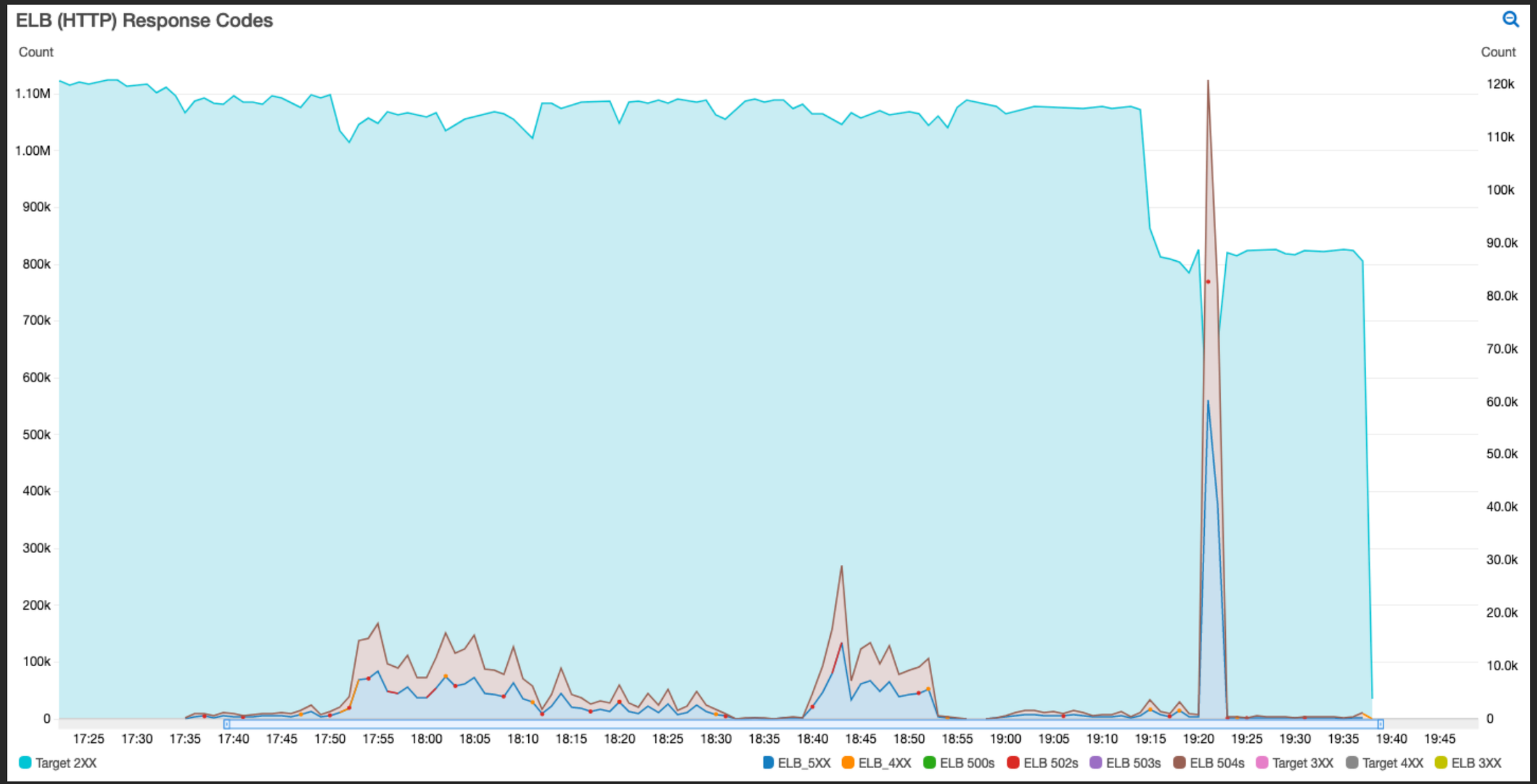
Workload metric: Merged graphs for dashboard



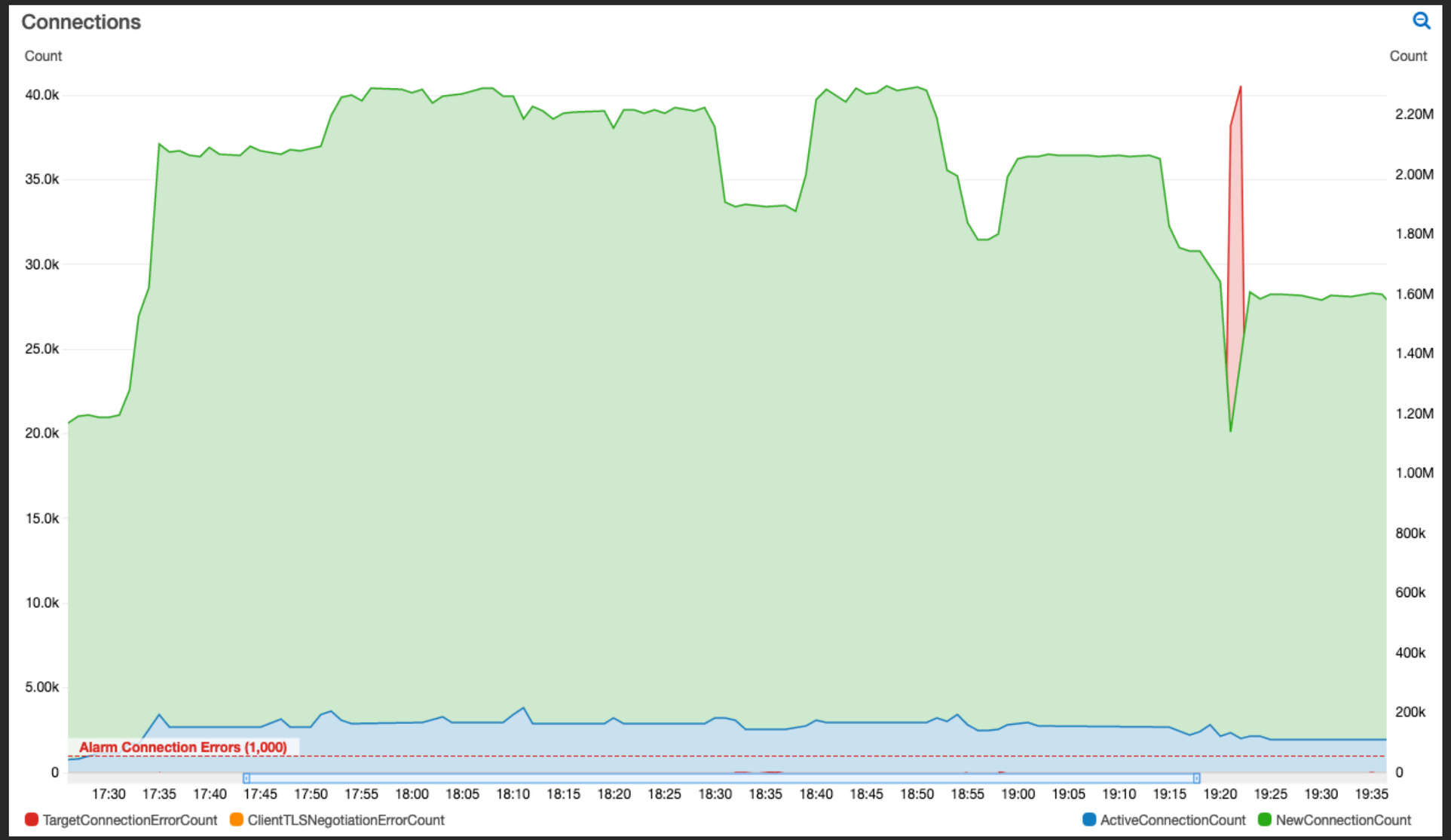
Workload metric: Merged graphs for dashboard



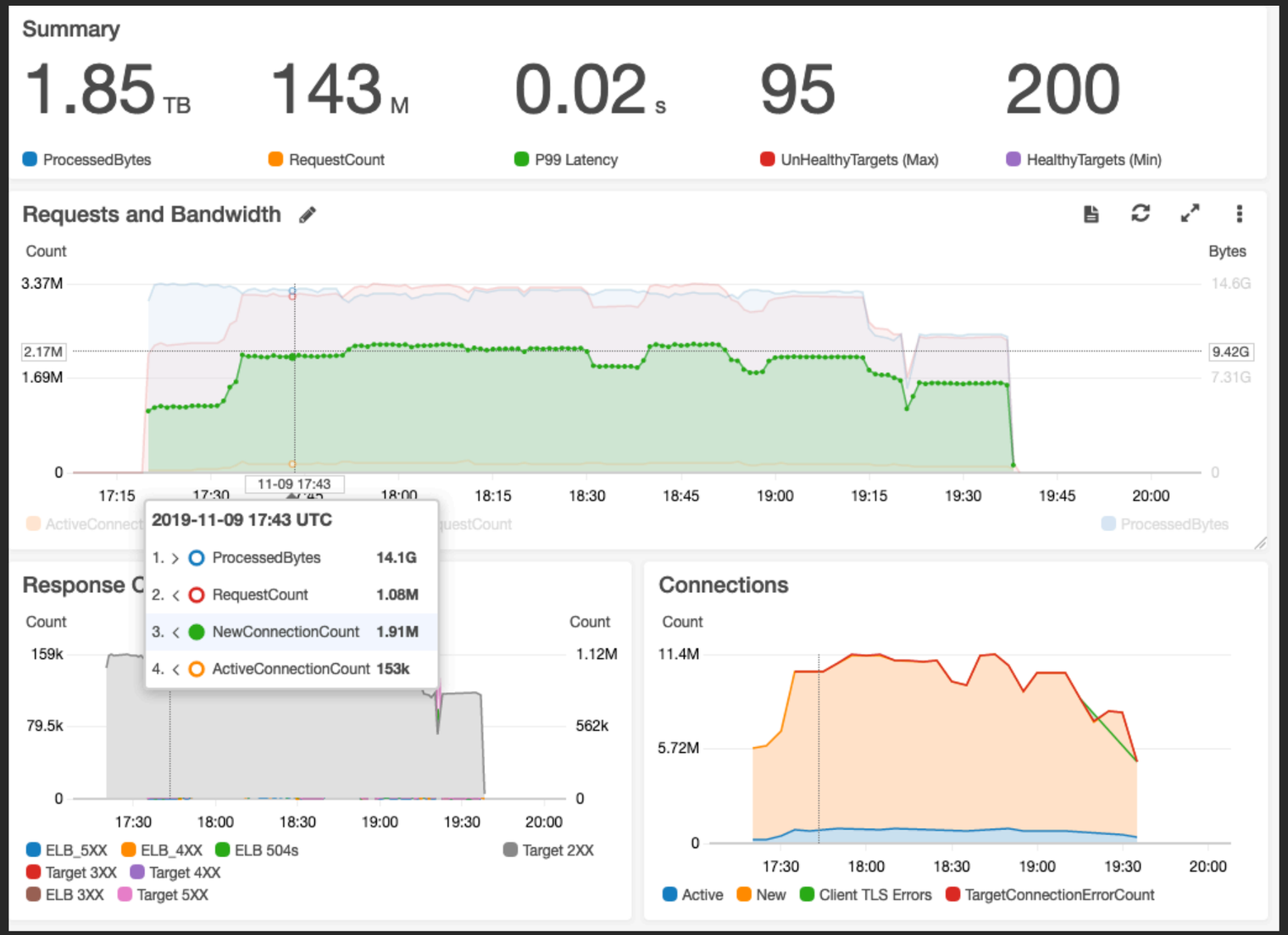
ELB metrics: Response codes



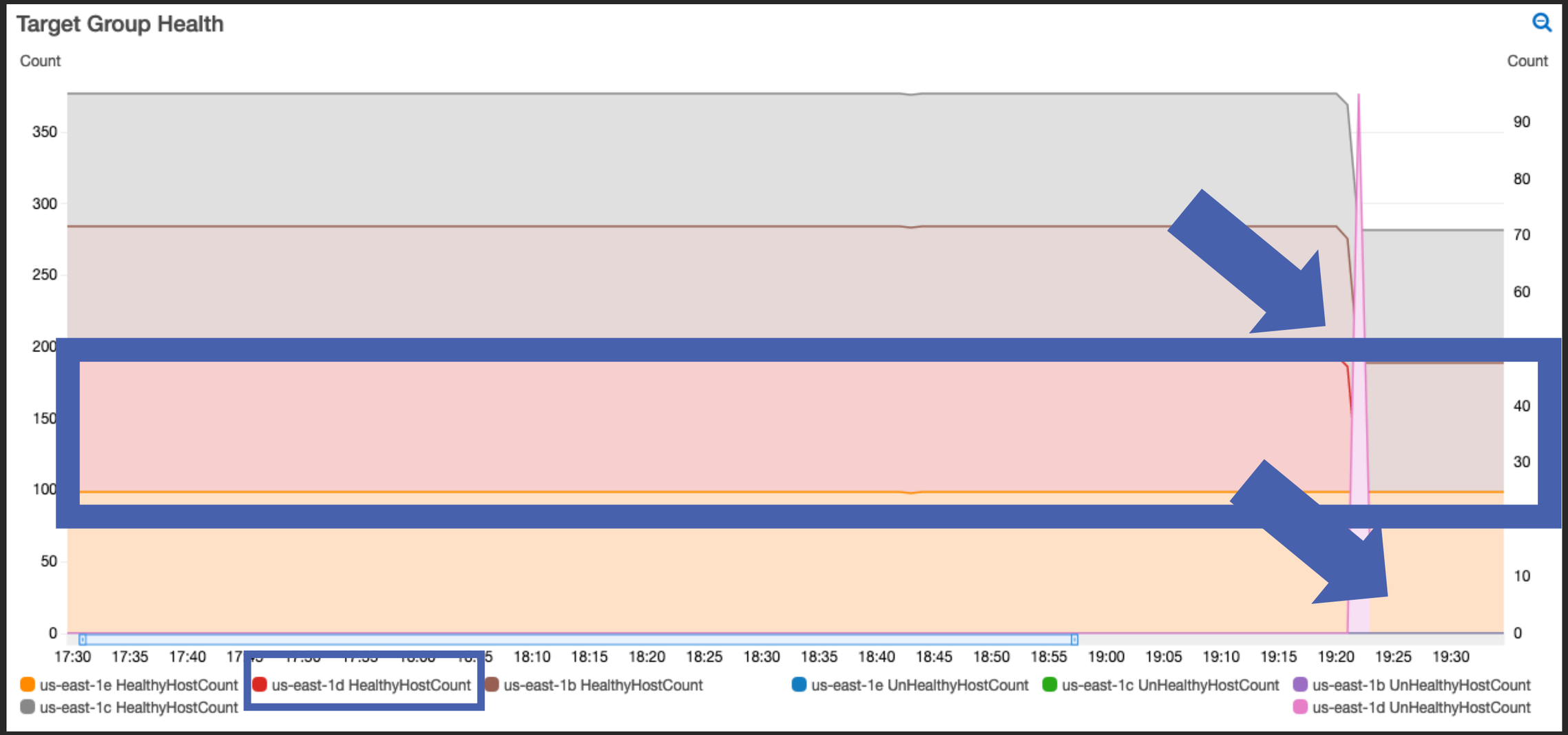
ELB metrics: Connections



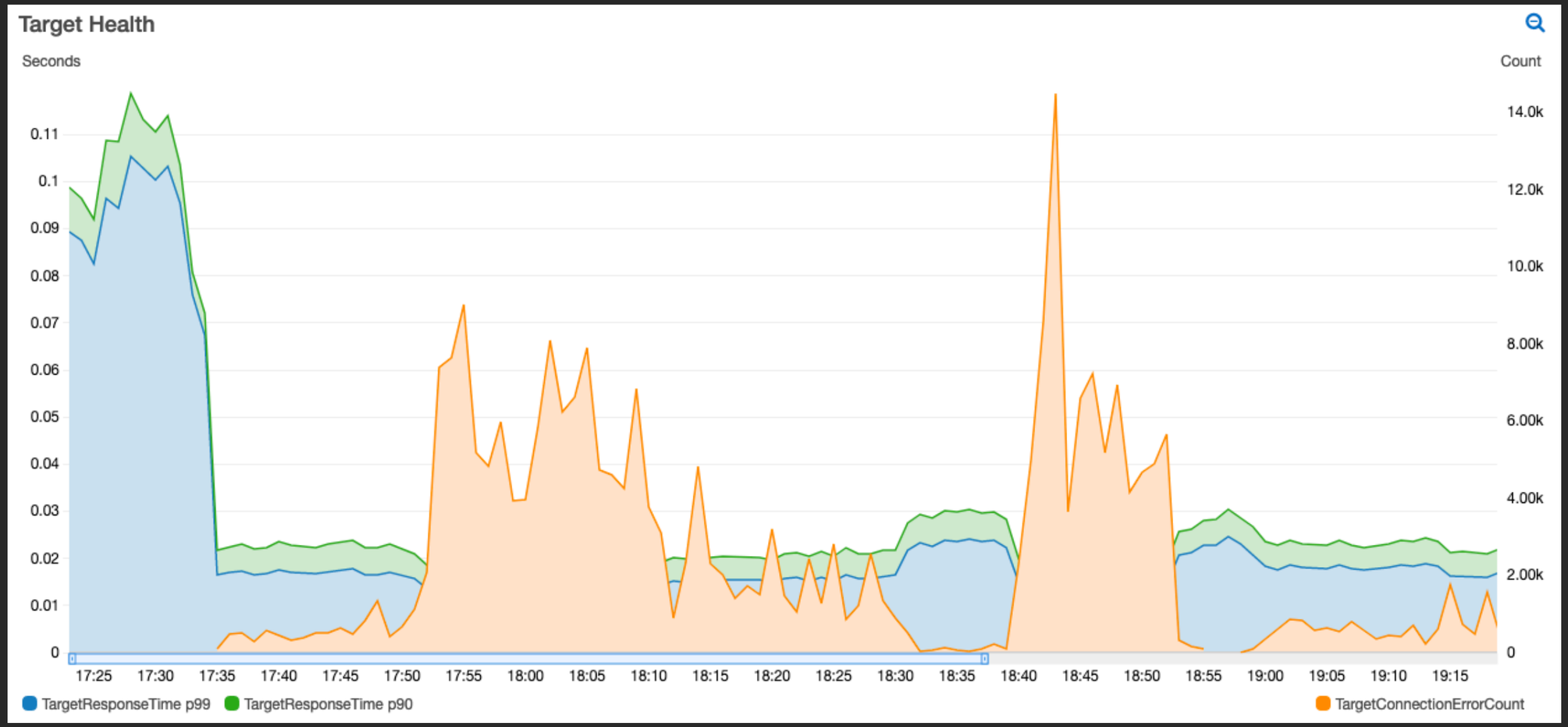
ELB metrics: Dashboard with graphs and summary



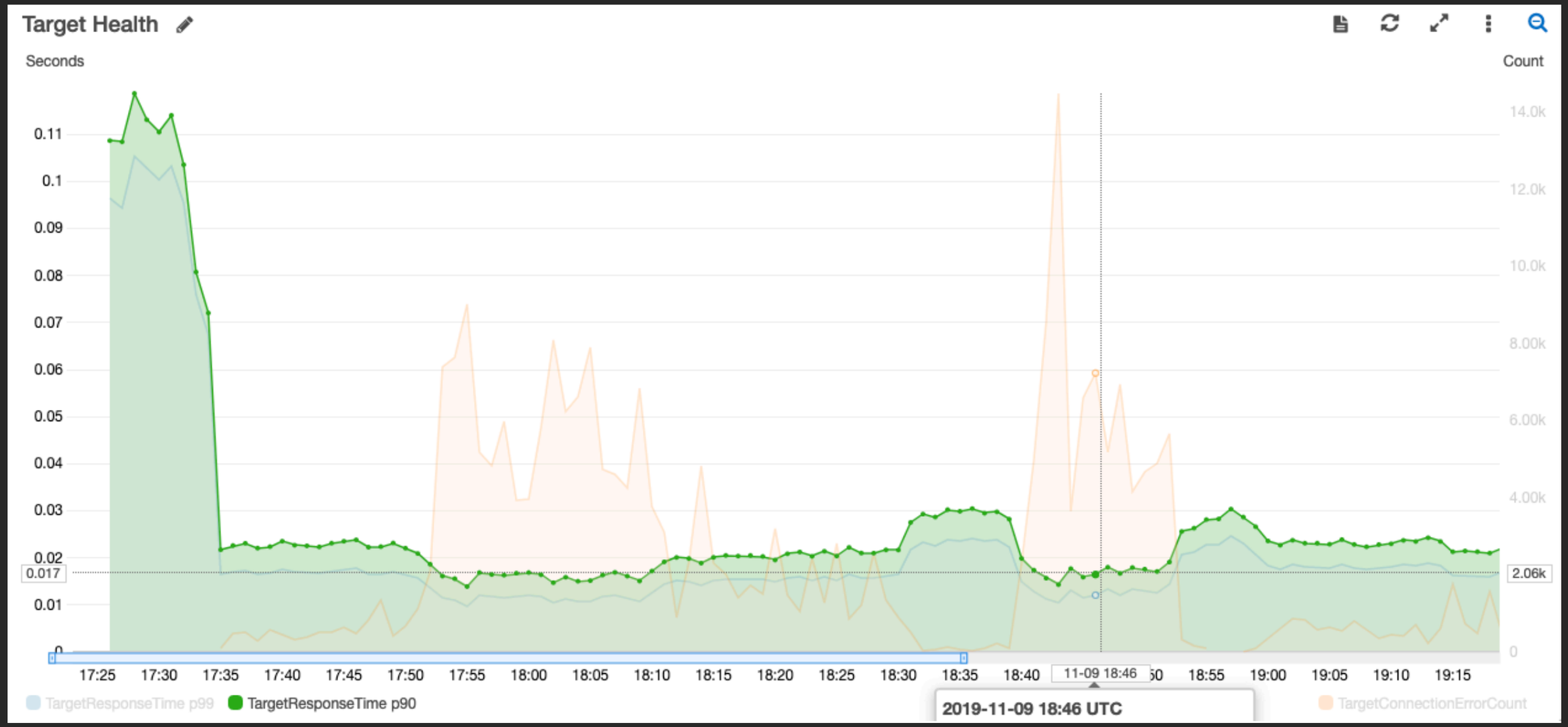
Target metrics: Healthy and unhealthy targets



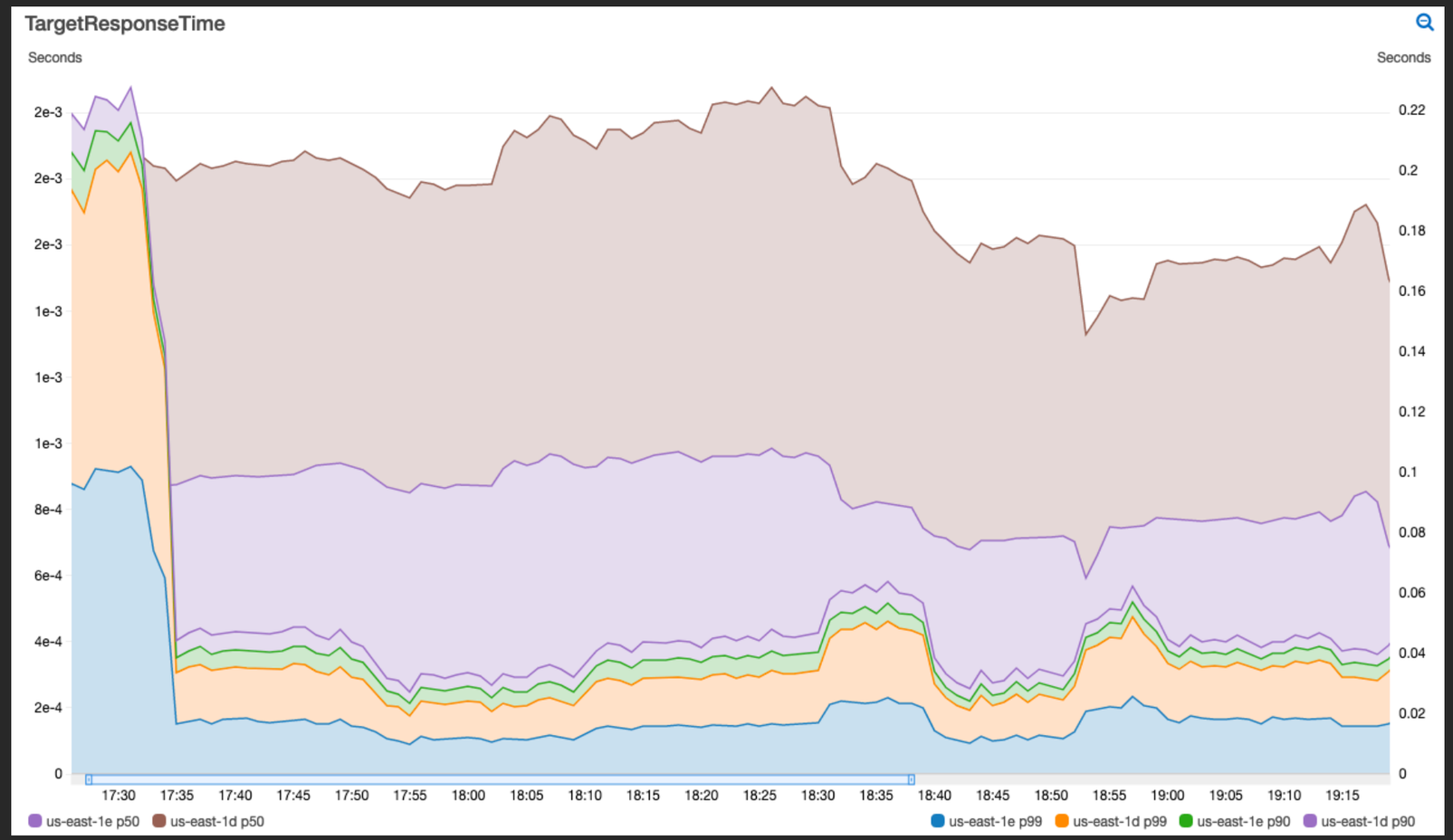
Target metrics: Response time and connection errors



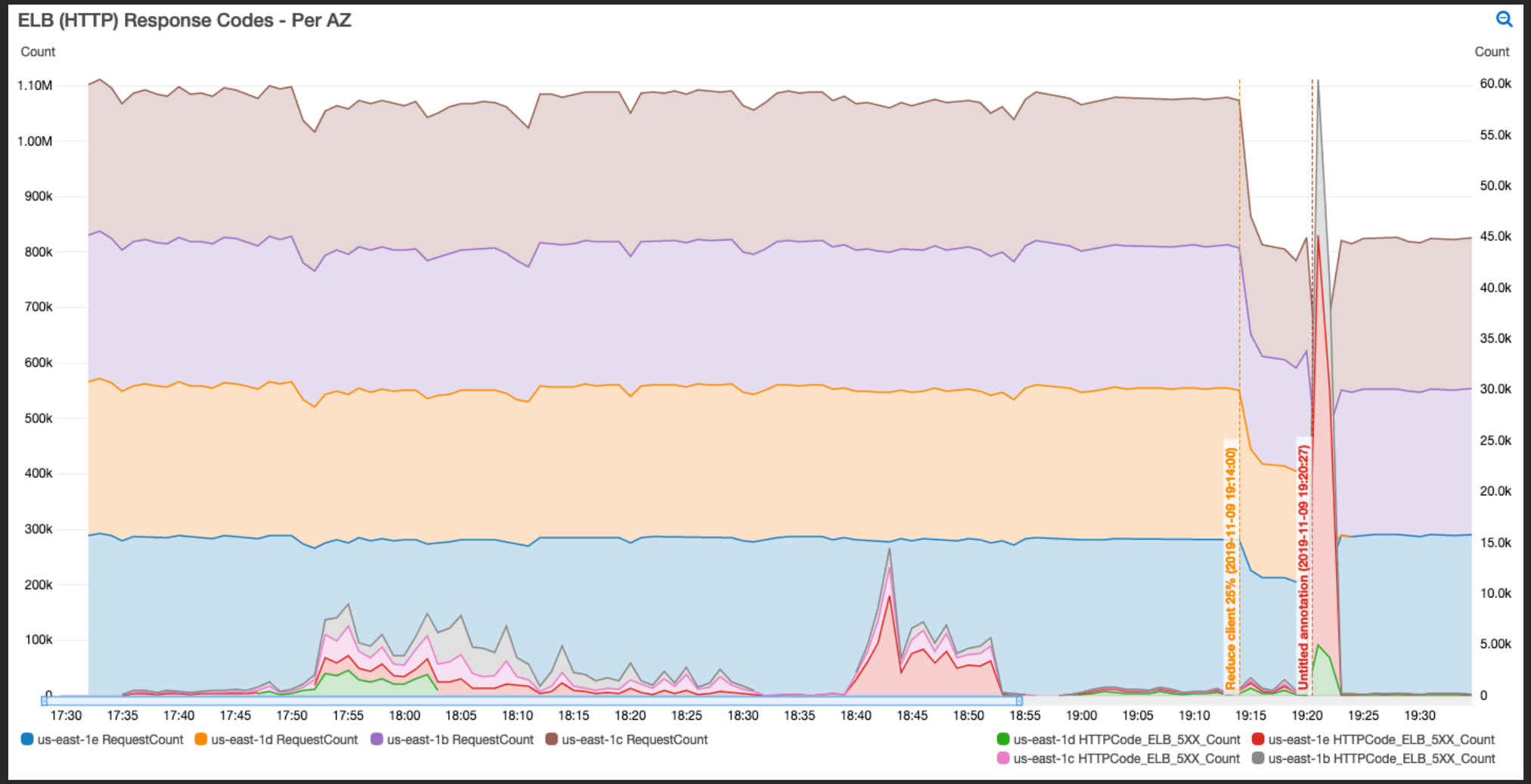
Target metrics: Response time and connection errors



Target metrics: Response time and connection errors



Troubleshooting - Is it me or the ELB?



Combined metrics: Dashboard

Learn networking with AWS Training and Certification

Resources created by the experts at AWS to help you build and validate networking skills

Free digital courses cover topics related to networking and content delivery, including Introduction to Amazon CloudFront and Introduction to Amazon VPC

Validate expertise with the
AWS Certified Advanced Networking - Specialty exam

Visit aws.amazon.com/training/paths-specialty

© 2019, Amazon Web Services, Inc. or its affiliates. All rights reserved.

aws training and certification

Thank you!

AWS re:Invent

© 2019, Amazon Web Services, Inc. or its affiliates. All rights reserved.

aws

Please complete the session survey in the mobile app.

AWS re:Invent

© 2019, Amazon Web Services, Inc. or its affiliates. All rights reserved.

aws