Scalibility and High Availibility

Contents

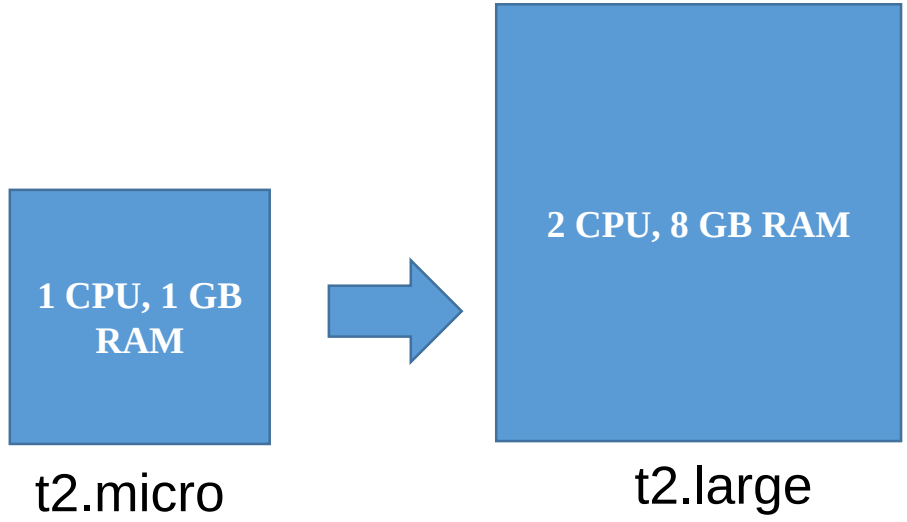
- High Availibility and Scalibility
- Load Balancer
- Auto Scaling Group
- Design HA Architecture

Scalibility

- Scalibility means the system can be adapted with greater loads, traffic
- There are 2 kinds of scalability:
  - Vertical Scalibility
  - Horizontal Scalibility

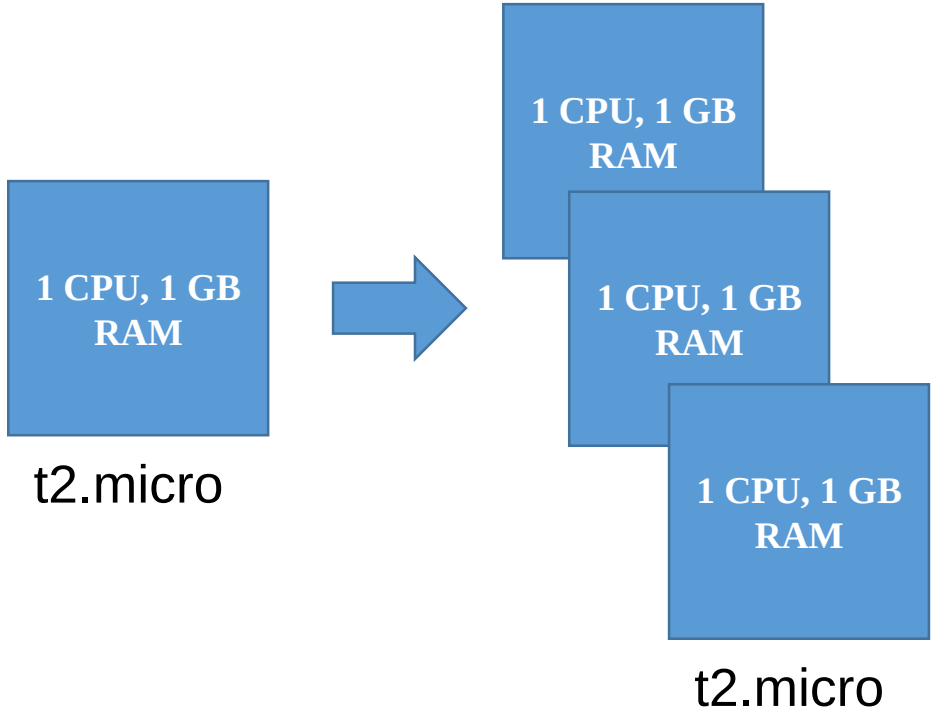
Scalibility – Vertical Scalibility

- Vertical Scalibility means increasing the **capacity** of the instance (increase RAM, CPU)
- Example:
  - Change from t2.micro instance type (1CPU, 1GB RAM) to t2.large (2CPU, 8GB RAM)
- Use cases:
  - For non-distributed system (Database)
  - Strictly coupling system



Scalibility – Horizontal Scalibility

- Horizontal Scalibility means increasing the **number** of the instances
- Example:
  - Increase number of instances from one to three instances
- Use cases:
  - Web, modern application
  - Strictly coupling system



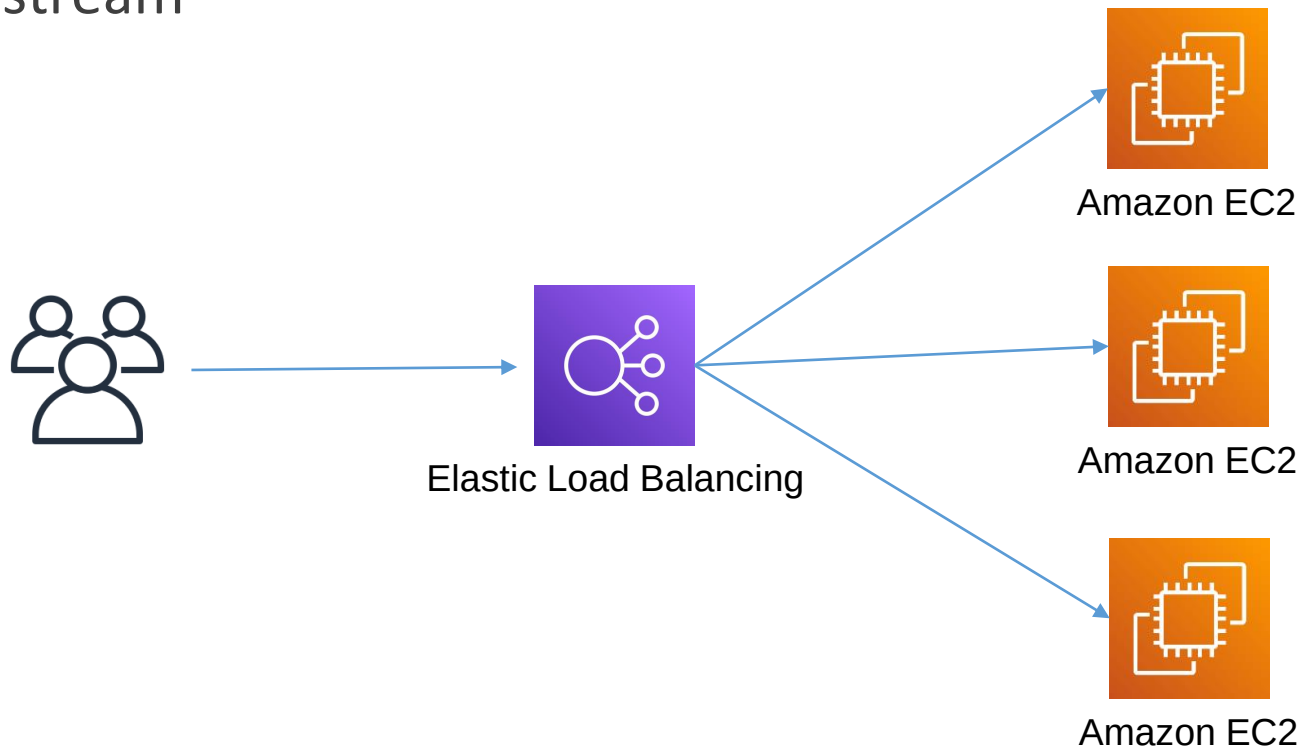
High Availibility (HA)

- The goal of HA is the service still be online if a data center loss
- HA means running the service in at least 2 Availibility Zone
- HA usually goes with Horizontal Scalibility
- HA can be passive (for RDS multi-AZ) or active (Horizontal Scaling - ASG)

Load Balancer

Load Balancer

- Load Balancers are servers that work as a proxy and forward traffic to multiple servers downstream



Load Balancer features

- Even/balance workload among downstream servers
- Provide a single point of access (DNS) to clients
- Handle failures of downstream instances
- Health check to downstream instances
- Provide SSL termination for your application
- HA across zones
- Separate Internet facing and internal facing traffic

Elastic Load Balancer

- Elastic Load Balancer is managed Load Balancer
  - Managed by AWS
  - High Availibility, Scalibility
- More costly than your own managed LB
- ELB can be easy intergrated with other AWS services (EC2, ASG, Route53…)

Elastic Load Balancer

Type of Elastic Load Balancer

- AWS provides 4 kinds of managed Load Balancers (LB)
- It highly recommends to use the modern, newer generation LB for more enhanced feature

LB type	Classic LB	Application LB	Network LB	Gateway LB
Release	2009	2016	2017	2020
Supported Protocols	HTTP/HTTPS, TCP, SSL (secure TCP)	HTTP, HTTPS, WebSocket	TCP, SSL (secure TCP), UDP	Working at layer 3 (Network Layer) – IP protocol

Classic Load Balancer (CLB)

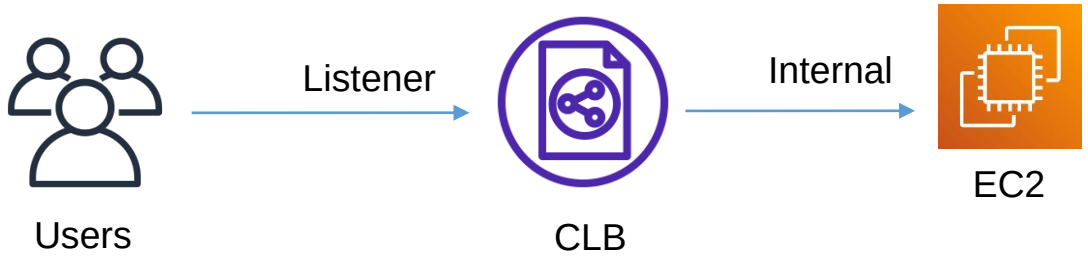
- Support TCP (layer 4), HTTP and HTTPS

  (Layer 7)

- Health checks are TCP or HTTP based

- DNS name (fixed)

  XXX.region.elb.amazonaws.com



Application Load Balancer (ALB)

- ALB work in layer 7 (HTTP)
- Load balancing traffic among multiple target groups (Group of machines)
- Support HTTP/2 and WebSocket
- Support redirects (from HTTP to HTTPs)

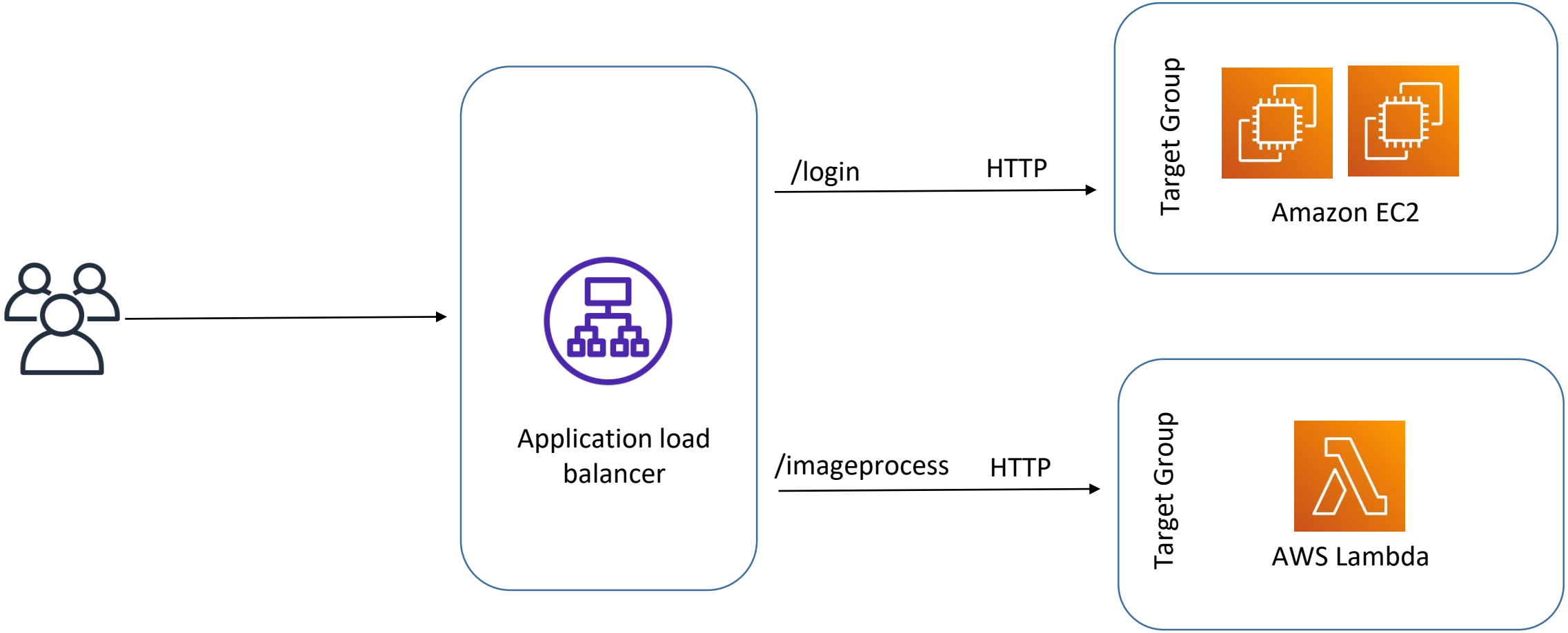
Application Load Balancer (ALB)

- Routing policy based on:
  - Path in URL (example.com/users & example.com/posts)
  - Hostname in URL (one.example.com & other.example.com)
  - Query String, Headers (example.com/users?id=123&order=false)
- ALB are a great fit for micro services & container-based application (example: Docker & Amazon EKS)
- In comparison, we'd need multiple Classic Load Balancer per application

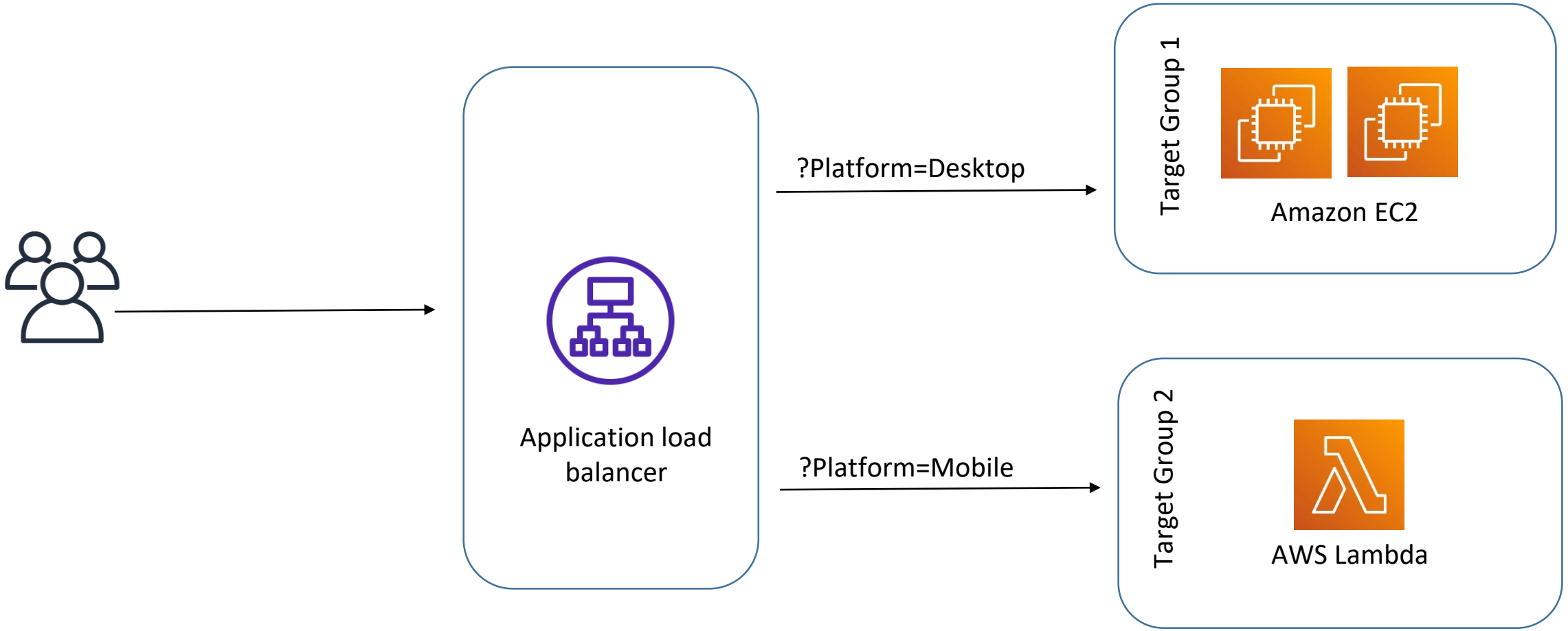
ALB – Target Groups

- Target Groups define what will process the requests from clients at downstream
- Target Groups can be:
  - EC2 instances (can be managed by an Auto Scaling Group) – HTTP
  - ECS tasks
  - Lambda function
  - IP addresses
- ALB can route to multiple target groups

ALB – Path Routing

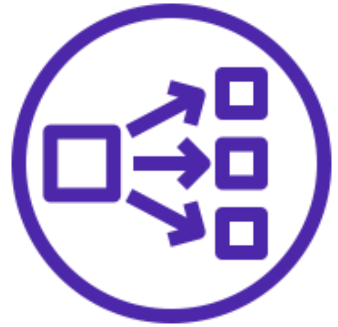


ALB – Query Strings/Parameters Routing



Network Load Balancer (NLB)

- NLB works in layer 4 (TCP + UDP)
  - Forward UDP/TCP traffics to instances
  - Low latency and handle millions of request per seconds
- NLB is supported to handle TCP/UDP traffic with high performance
- Elastic IP can be assign for NLB (good fit for whitelisting purposes)
- Not include in AWS free tier

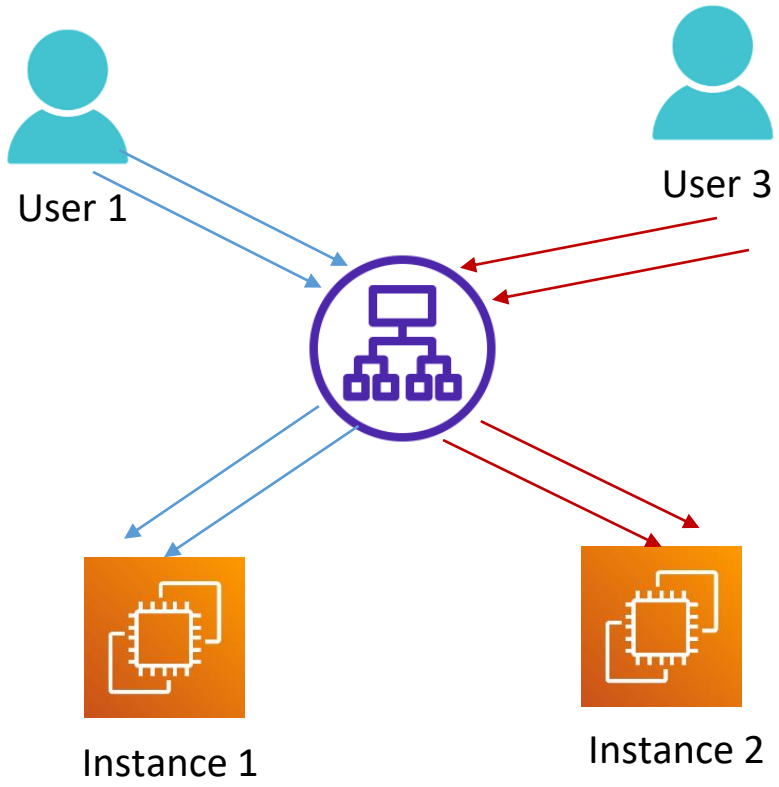


NLB – Target Groups

- NLB Target Groups can be:
  - EC2 instances (can be managed by an Auto Scaling Group)
  - IP addresses (Must be private IP)
  - Application Load Balancer

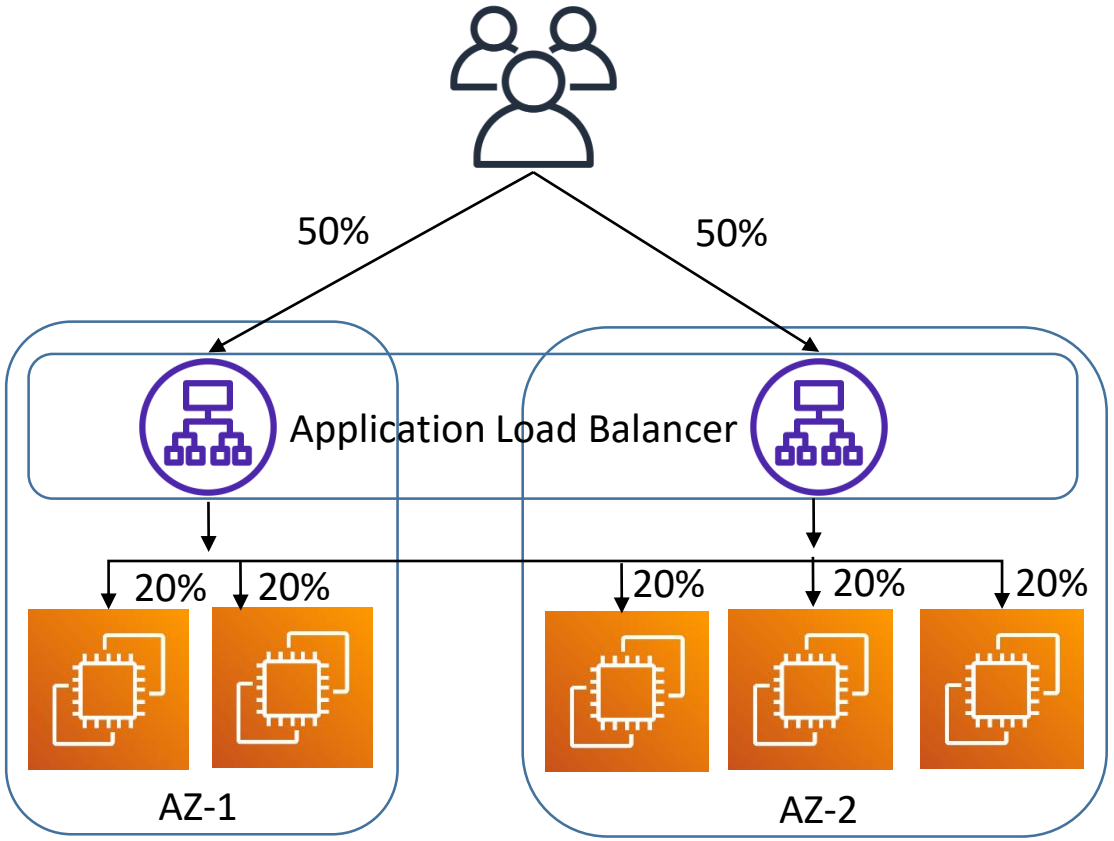
Sticky Session (Session Affinity)

- Using Stickness to route same client request to same backend instance
- This works for CLB & ALB
- Using Sticky session for stateful application
- Enable Stickness can cause imbalance load among EC2 backend instances

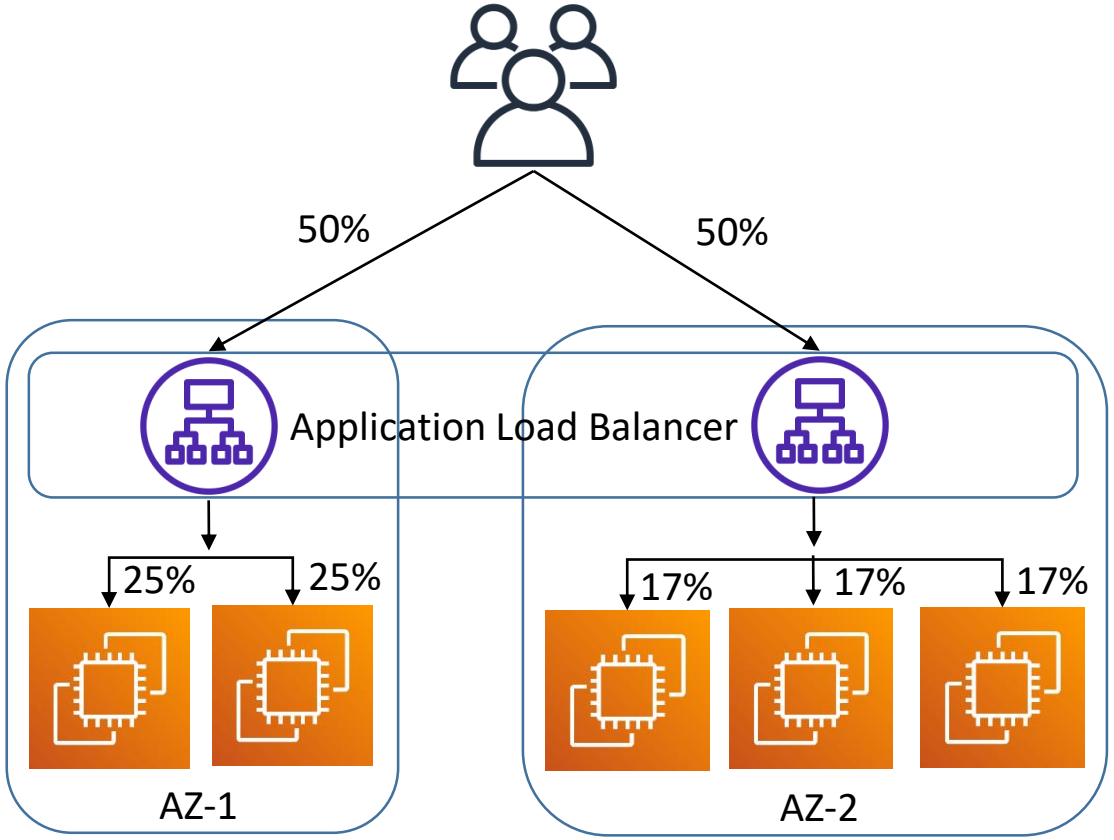


Cross-Zone Load Balancing

- With Cross-Zone Loab Balacing



- Without Cross-Zone Load Balancing



Cross-Zone Load Balancing (cont.)

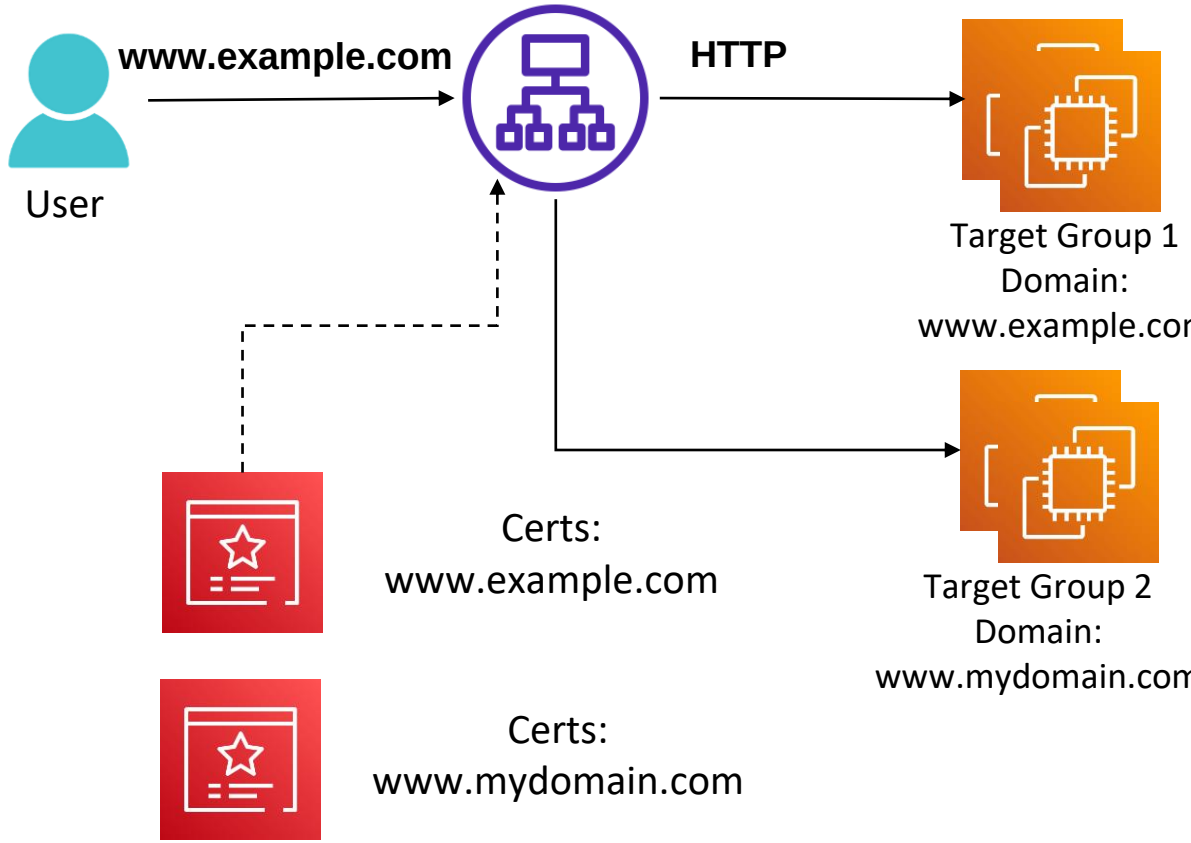
Characters	Application LB	Network LB	Classic LB
Enable by default	Yes (cannot disable)	No	No
Charge for Inter AZ (cost)	No	Yes	Yes

SSL Termination

- LB uses an X.509 certificates (SSL server certs)
- SSL certs are managed by ACM (Amazon Certificate Manager)
- SSL certs on ACM can be uploaded by our own certs

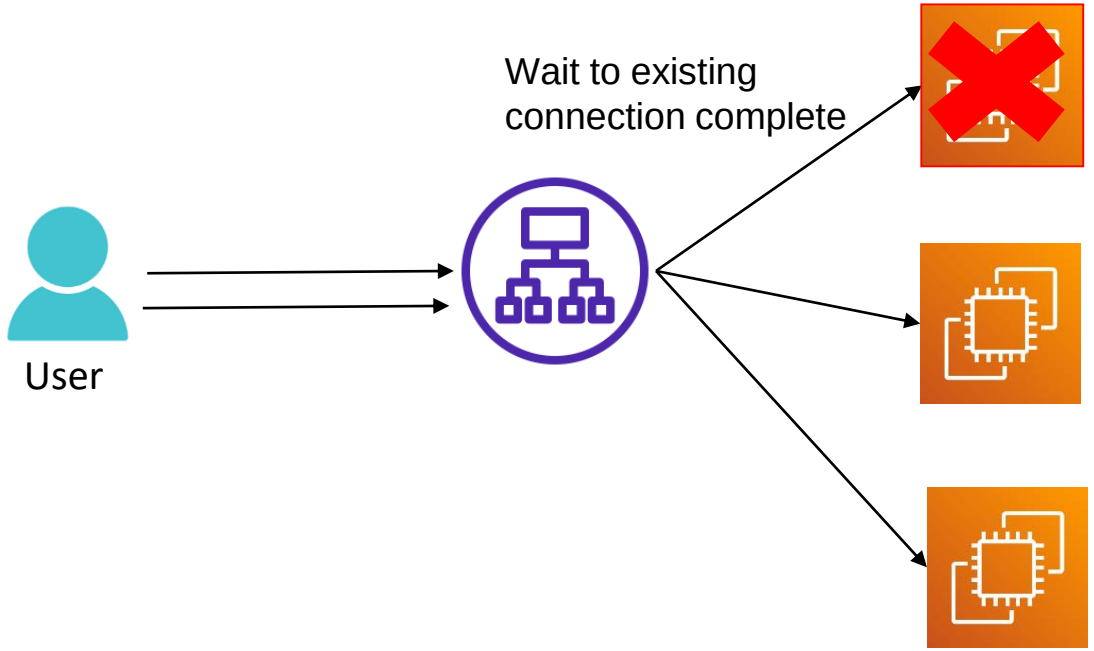
SSL – Server Name Indicator (SNI)

- SNI allows to support multiple web server with multiple of SSL certs
- Clients must indicate the hostname of target server for handshake
- Note:
  - Only work for new generation LB (ALB, NLB)
  - Not supported for CLB



Connection Draining

- Time to wait the in-processing requests when the instances unhealthy or de-registering
- Stop sending requests to those instances (unhealthy or de-registering)
- Draining period from 1 ~ 3600s (default 300s)
- Can be set to 0



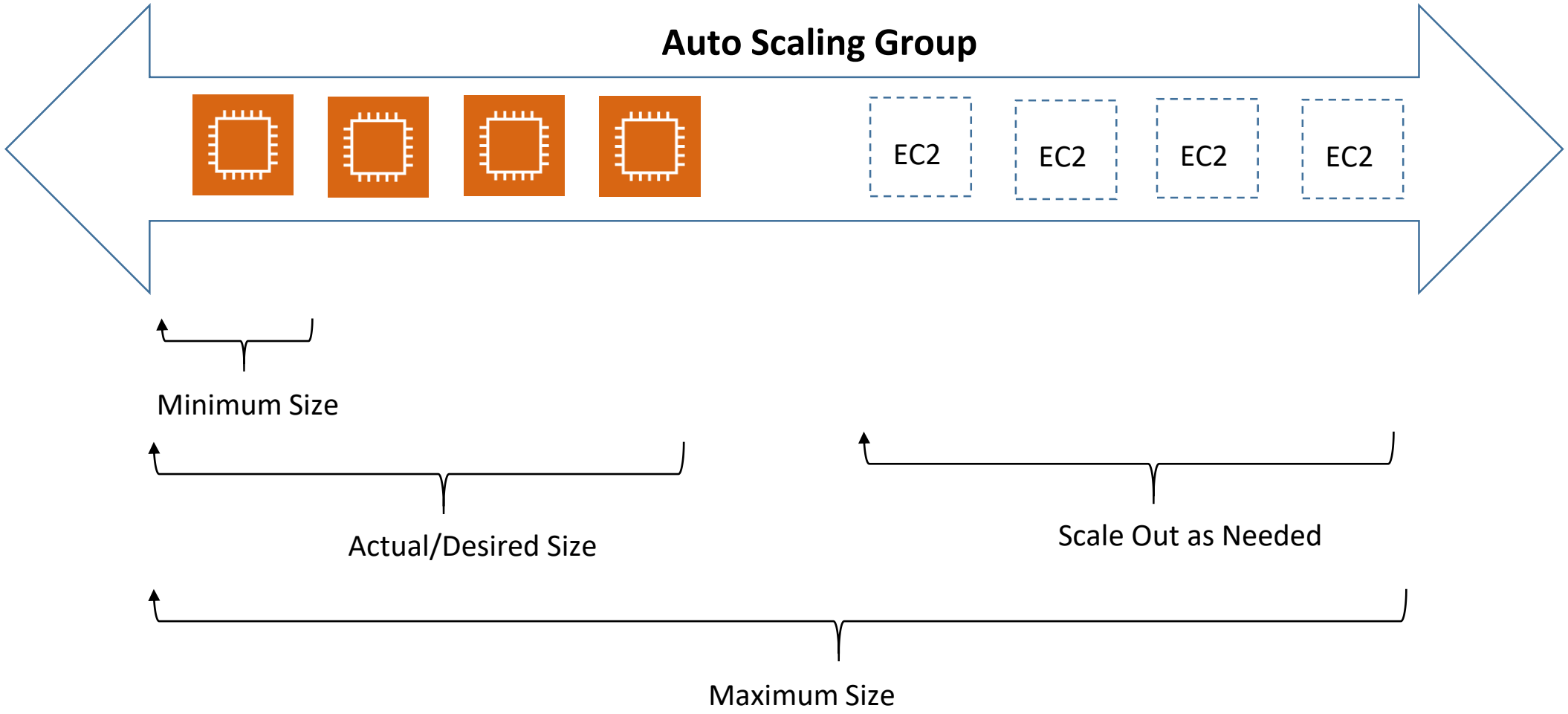
Auto Scaling Group

What is Auto Scaling Group (ASG)

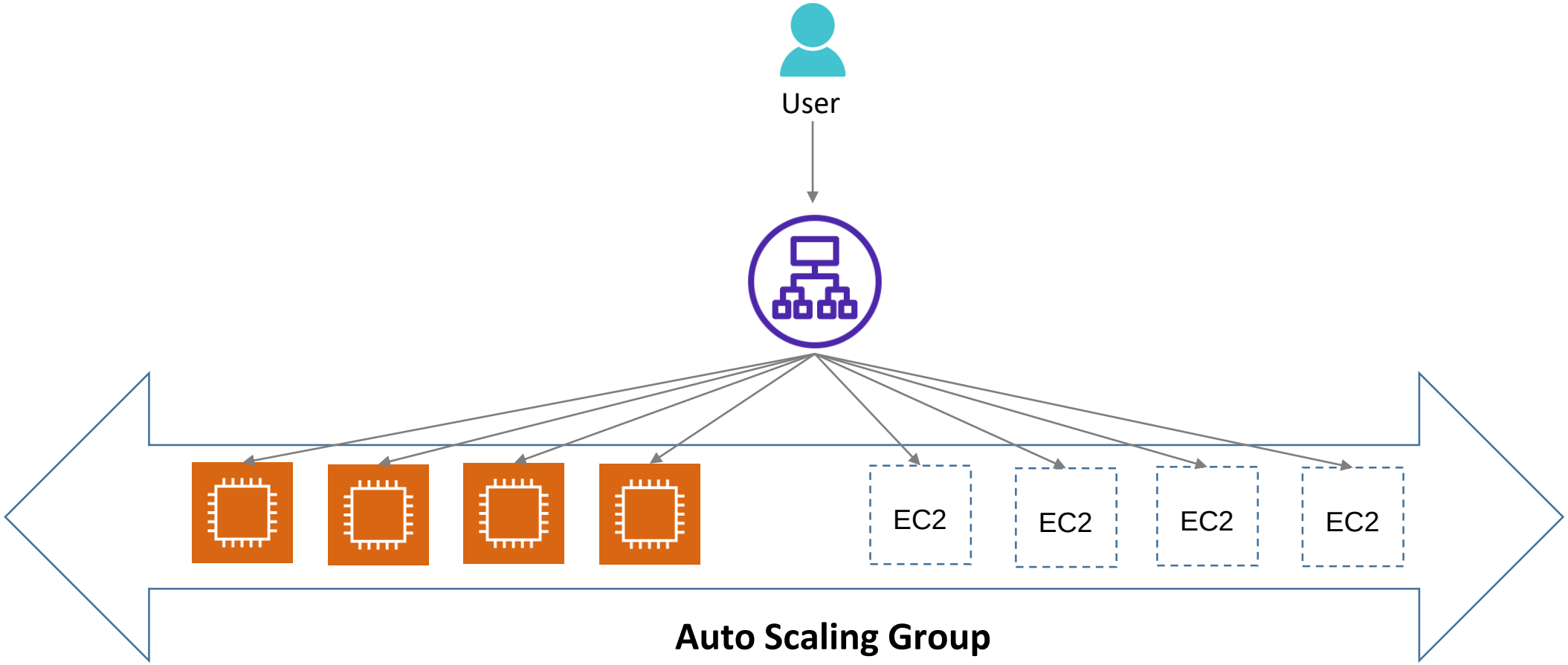
- ASG allows to Scale Out and Scale In the number of instances (EC2) to adapt with the load
- Take advantage of elastiscity of Cloud
- ASG provide features:
  - Scale Out (Increase number of EC2 instances)
  - Scale In (Remove EC2 instances)
  - Guarantee Maximum/minimum instances running
  - Automatically add instances to a Load Balancer

Auto Scaling Group

Auto Scaling Group in AWS



Auto Scaling Group with Load Balancer



ASG Component

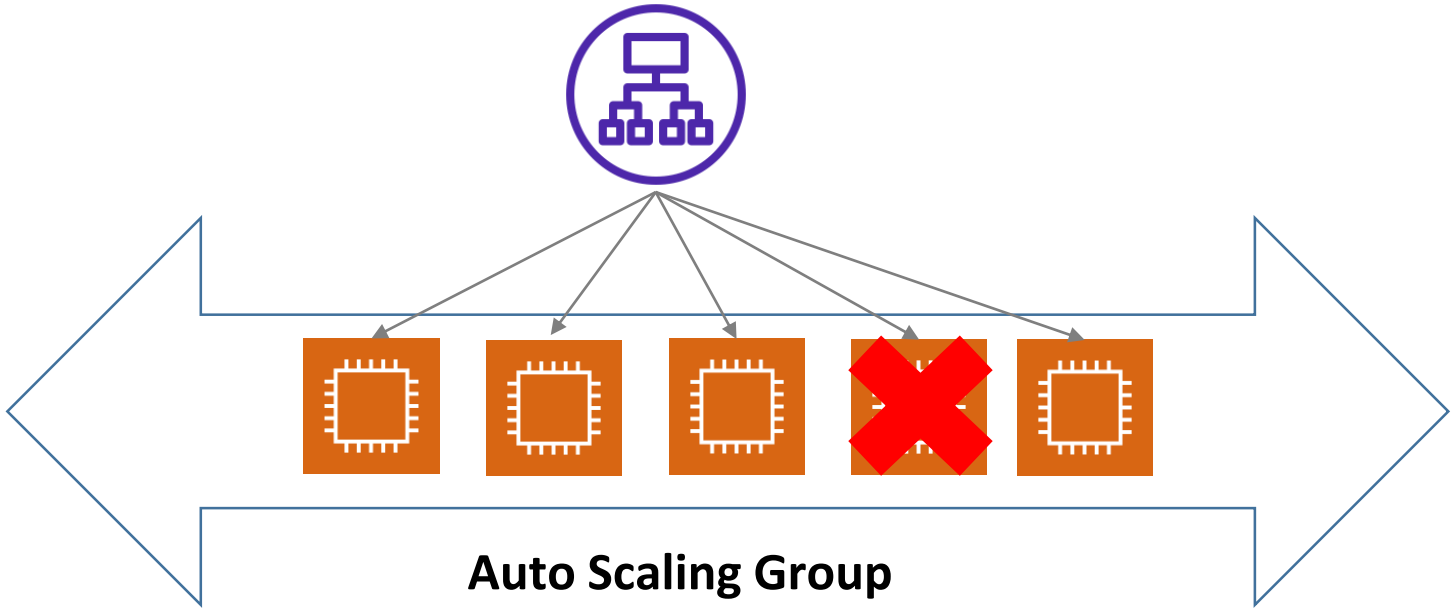
- Groups: Logical Group (Web, App or DB)
- Configuration Template
  - Groups uses a Launch Template or Launch Configuration (LC) template for its EC2 instances
  - LC includes specific information as AMI ID, Instance type, key pair, SGs, EBS…
- Scaling Options
  - Provides several ways to scale your ASGs
  - Example: Scale action based on specific condition (CPU, Custom metric…s)

Scaling Options

- Maintain number of instances at all times
- Scale Manually
- Scale based on Scheduler
- Scale based on demand
- Use predictive scaling

Maintain number of instances at all times

- ASG performs a periodic health check on running instances within Auto Scaling Group
- When ASG finds unhealthy instance, it terminates that instance and create new one



Scale Manually

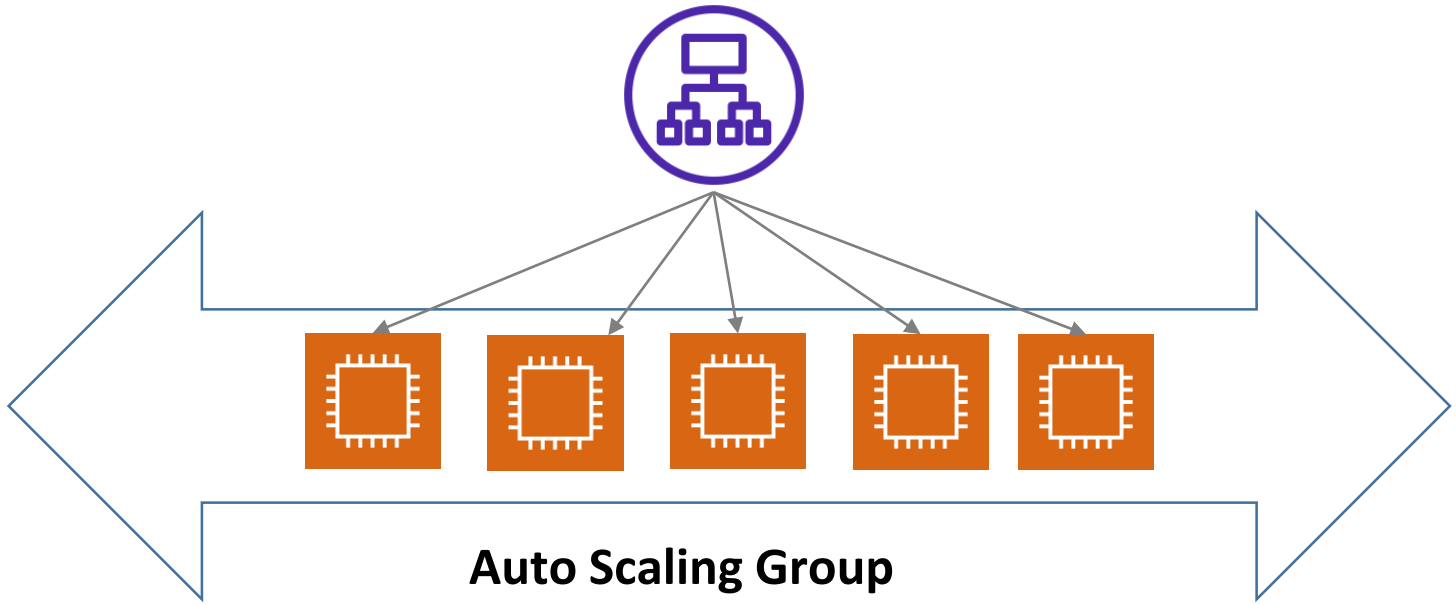
- You specific only change in the maximum, minimum or desired capacity of your ASG

Desired=5

Desired=4

Desired=3

Desired=2

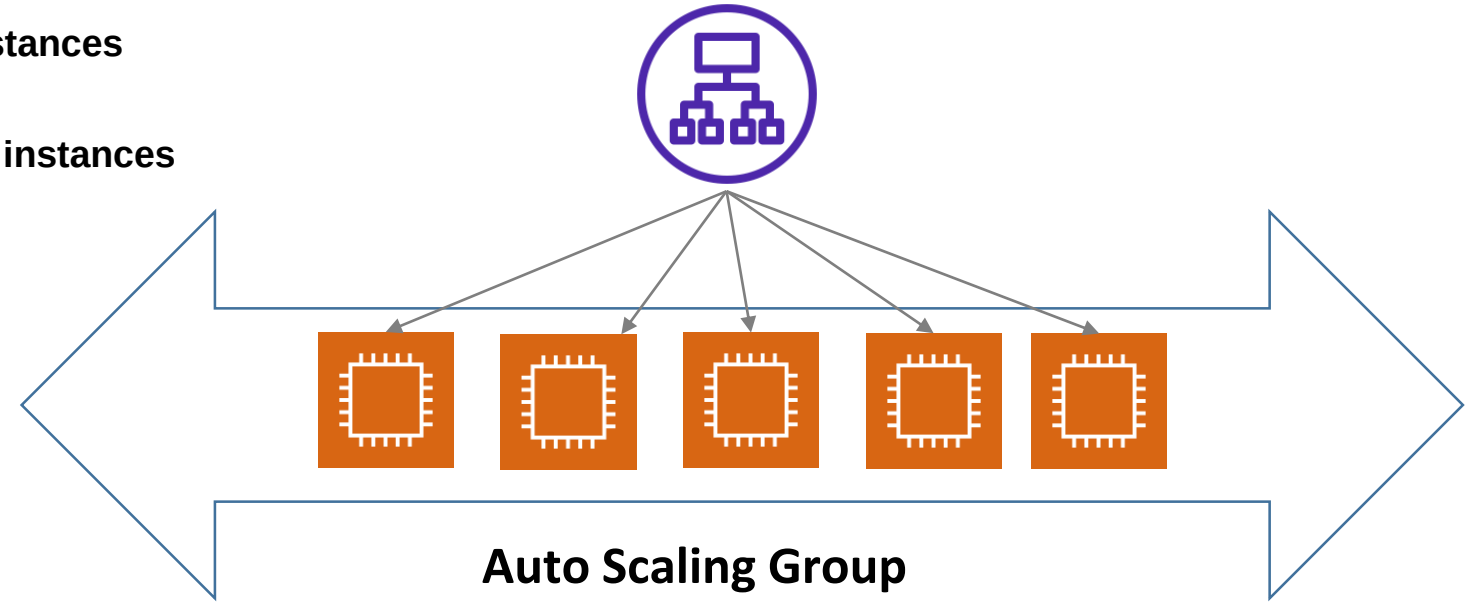


Scale Based On Schedule

- Scaling Action is performed automatically at time and date

19:00 Scale Up 2 Instances

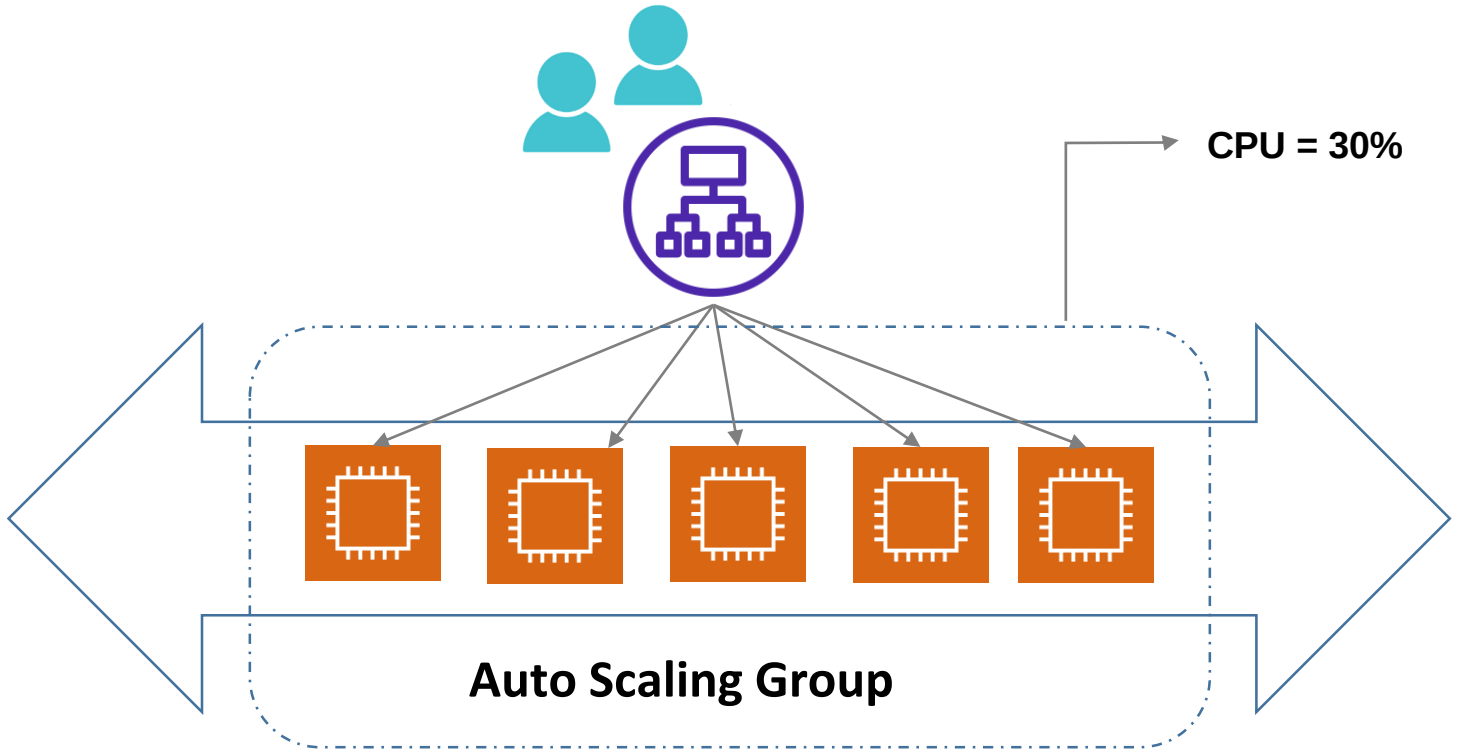
08:00 Scale Down 2 instances



Scale Based On Demand

- This method is helpful for scaling in response to changing conditions

Policy
- CPU > 80% Scale Up 2 Instances
- CPU < 45% Scale Down 1 instances

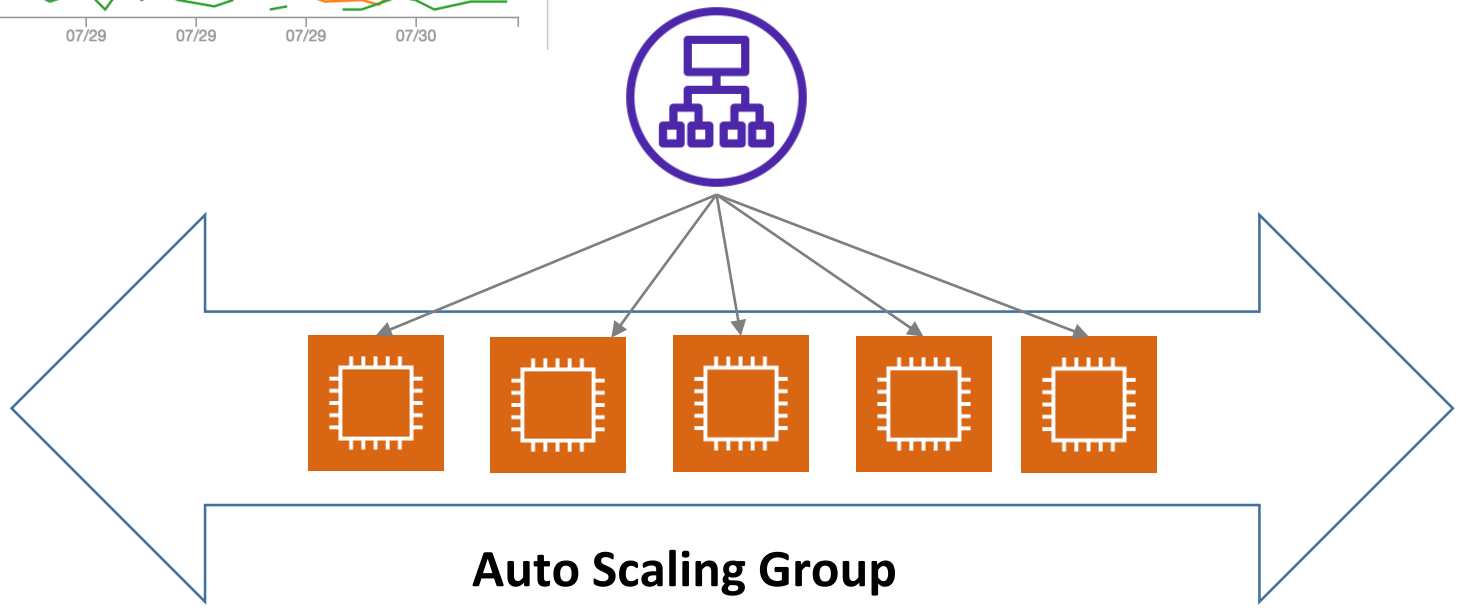


Predictive Scaling

- Using machine learning to predict capacity requirements
- Data is collected from Cloud Watch for ML model
- Scaling action is triggered before the change of load

Predictive Scaling (cont.)

MACHINE LEARNING



Exam Tips

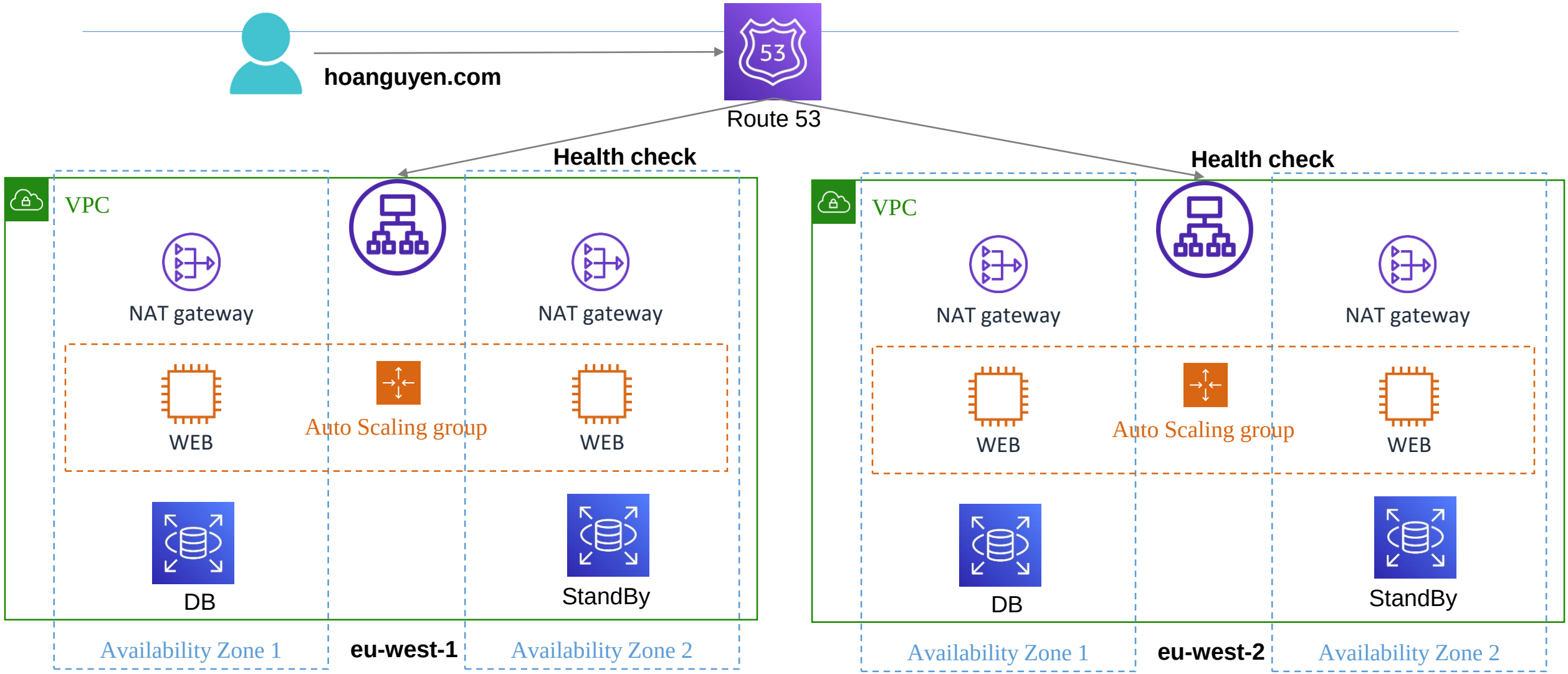
- Scaling Options
  - Maintain number of instances at all times
  - Scale Manually
  - Scale based on Scheduler
  - Scale based on demand
  - Use predictive scaling

Design HA Architecture

Plan for failure

- Werner Vogels (CTO AWS): “Everything fails all the time”
- You should always prepare for failure

HA Architecture



Exam Tips

- Always design for failure
- Use multiple AZ or regions where ever you can
- Know difference between Multi-AZ and Read Replicas for RDS
- Know difference between Scale Up and Scale Out
- Always think about the cost element