Training Course

Amazon Web Service

Course Schedule

Day	Presentations	Lab
Day 1	System Operations on AWS	
Day 2	Computing on AWS	X
Day 3	Networking on AWS	X
Day 4	Storage and Archiving in the Cloud	X
Day 5	Monitoring in the Cloud	X
Day 6	Managing Resource Consumption in the Cloud	X

Module 5:

Elastic Load Balancing
Auto Scaling Group

Module 5: ELB & ASG

- **Goal**: Understanding Elastic Load Balancing & Auto Scaling Group
  - ✓ Understand what is ELB & ASG
  - ✓ How to load balancing in AWS
  - ✓ Auto Scaling Group
  - ✓ How to connect with others?

Lab: Create and configuring ELB & ASG

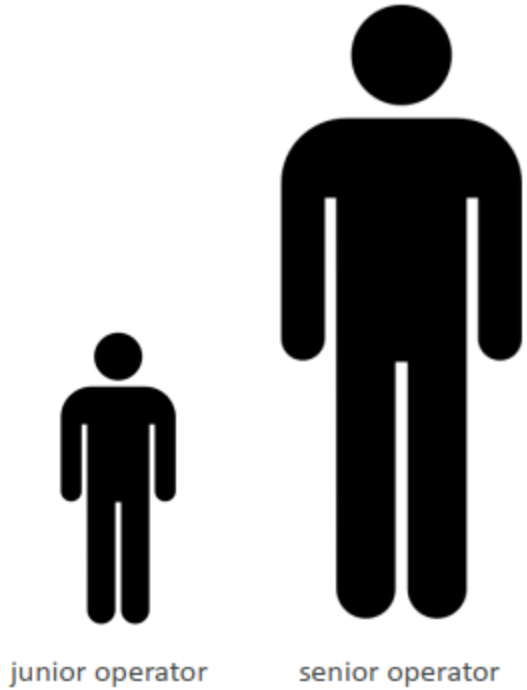
Elastic Load Balancing

Scalability & High Availibity

- Scalability means that an application/system can handle greater loads by adapting
- There are two kinds of scalability:
  - Vertical Scalability
  - Horizontal Scalability (=elasticity)
- **Scalability is linked but different to High Availability**
- Let's deep dive into the distinction, using a call center as an example

Vertical Scalability

- Vertially scalability means increasing the size of the instance
- For example, your application runs on a t2.micro
- Scaling that application vertically means running it on a t2.large
- Vertical scalability is very common for non distributed systems, such as a database
- RDS, ElastiCache are services that can scale vertically
- There's usually a limit to how much you can vertically scale (hardware limit)

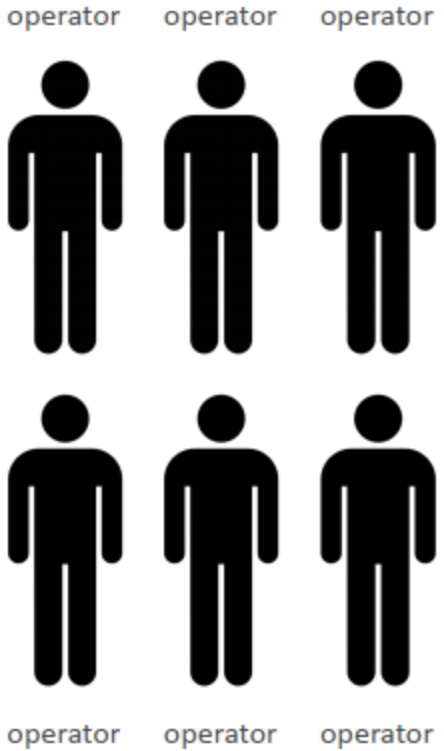


Horizontal Scalability

- Horizontal Scalability means increasing the number of instances/system for your application

- Horizontal scaling implies distributed system
- This is very common for web applications/modern applications

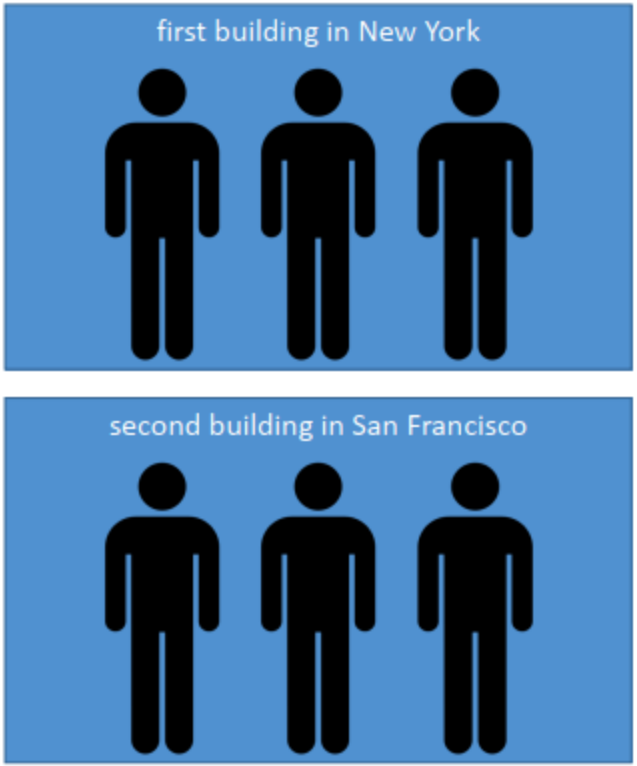
- It's easy to horizontally scale thanks the cloud offerings such as Amazon EC2



High Availability

- High Availability usually goes hand in hand with honrizontal scaling
- High Availability means running your applications/system in at least 2 data center (==Availability Zones)
- The goal of high availability is to survive a data center loss

- The high availability can be passive (for RDS Multi AZ for example)
- The high availability can be active (for horizontal scaling)

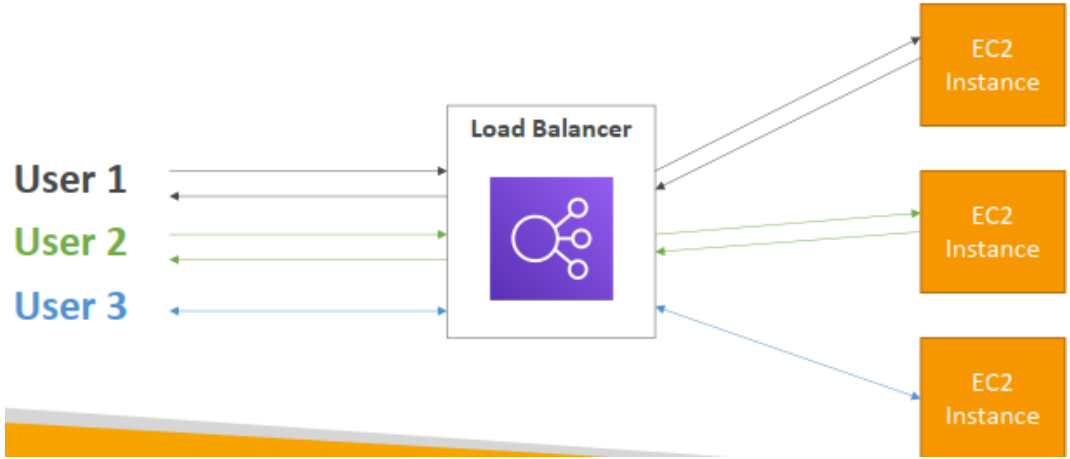


High Availability & Scalability For EC2

- Vertical Scaling: Increase instance size (= scale up/down)
  - From: t2.nano – 0.5G of RAM, 1 vCPU
  - To: u-1 2tb 1.metal – 12.3 TB of RAM, 449 vCPUs
- Horizontal Scaling: Increase number of instances (= scale out/in)
  - Auto Scaling Group
  - Load Balancer
- High Availability: Run instances for the same application across multi AZ
  - Auto Scaling Group multi AZ
  - Load Balancer multi AZ

What is load balancing?

- Load balancers are servers that forward internet traffic to multiple servers (EC2 Instances) downstream



Why use a load balancer?

- Spread load across multiple downstream instances
- Expose a single point of access (DNS) to your application
- Seamlessly handle failures of downstream instances
- Do regular health checks to your instances
- Provide SSL termination (HTTPS) for your websites
- Enforce stickiness with cookies
- High availability across zones
- Separate public traffic from private traffic

Why use an EC2 Load Balancer?

- An ELB (EC2 Load Balancer) is a managed load balancer
  - AWS guarantees that it will be working
  - AWS takes care of upgrades, maintenance, high availability
  - AWS provides only a few configuration knobs
- It cost less to setup your own load balancer but it will be a lot more effort on your end
- It is integrated with many AWS offerings/services

Health Checks

- Health Checks are crucial for Load Balancers
- Thye enable the load balancer to know if instances it forwards traffic to available to reply to requests
- The health check is done on a port and a route (/health is common)
- If the response is not 200 (OK), then the instance is unhealthy

Types of load balancer on AWS

- AWS has **3 kinds of managed Load Balancers**
- Classic Load Balancer (v1 – old generation) – 2009
  - HTTP, HTTPS, TCP
- Application Load Balancer (v2 – new generation) – 2016
  - HTTP, HTTPS, WebSocket
- Network Load Blancer (v2 – new generation) – 2017
  - TCP, TLS (secure TCP) & UDP
- Overall, it is recommended to use the newer/v2 generation load balancers as they provide more features
- You can setup **internal** (private) or **external** (public) ELBs

Load Balancer Security Groups

Load Balancer Security Group:

Type	Protocol	Port Range	Source	Description
HTTP	TCP	80	0.0.0.0/0	Allow HTTP from an...
HTTPS	TCP	443	0.0.0.0/0	Allow HTTPS from a...

Application Security Group: Allow traffic only from Load Balancer

Type	Protocol	Port Range	Source	Description
HTTP	TCP	80	sg-054b5ff5ea02f2b6e (load-b	Allow Traffic only...

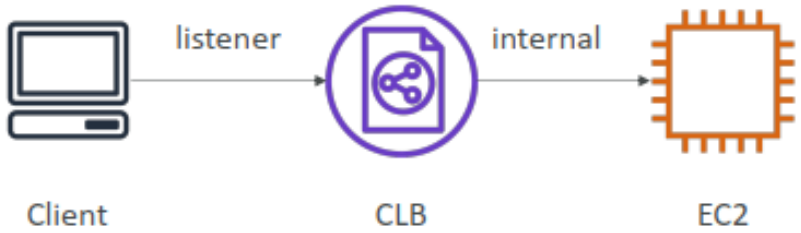
Load Balancer Good to Know

- LBs can scale but not instantaneously – contact AWS for a “warm-up”
- Troubleshooting
  - 4xx errors are client induced errors
  - 5xx errors are application induced errors
  - Load Balancer Errors 503 means at capacity or no registered target
  - If the LB can’t connect to your application, check your security groups!
- Monitoring
  - ELB access logs will log all access request (so you can debug per request)
  - CloudWatch Metrics will give you aggregate statistics (ex: connections count)

Classic Load Balancer (v1)

- Supports TCP (Layer 4), HTTP & HTTPS (Layer 7)
- Health checks are TCP or HTTP based
- Fixed hostname

  XXX. Regionx.elb.amazonaws.com



Application Load Balancer (v2)

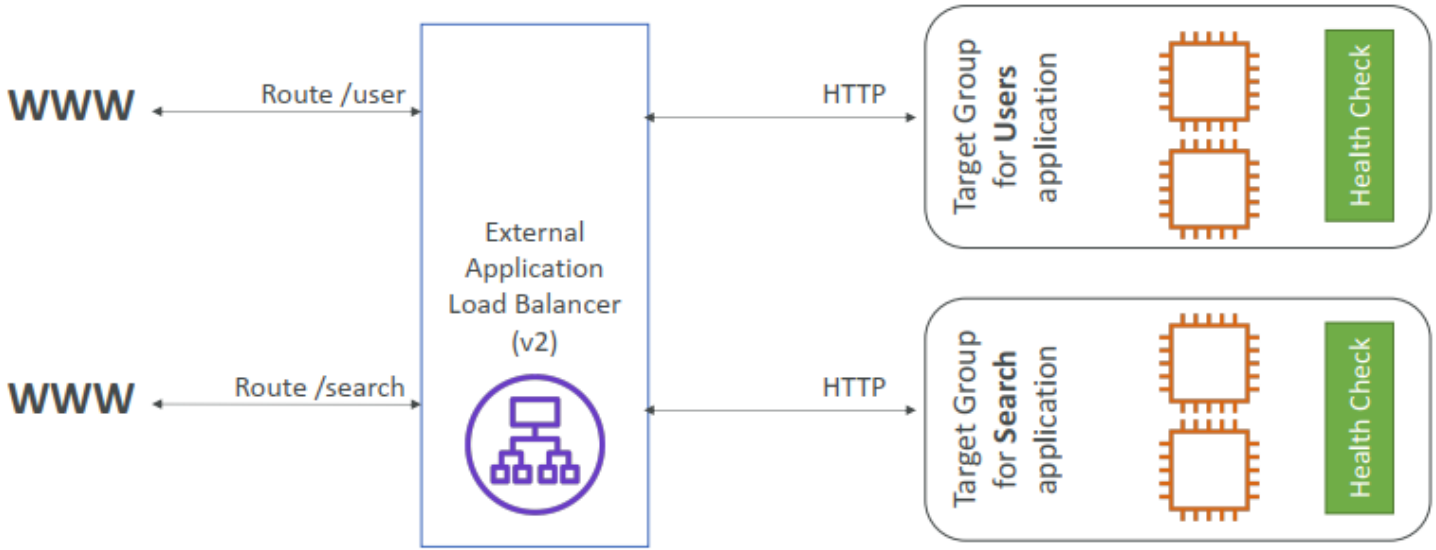
- Application load balancers is Layer 7 (HTTP)

- Load balancing to multiple HTTP applications across machines (Target group)
- Load balancing to multiple applications on the same machine (ex: containers)
- Support for HTTP/2 and WebSocket
- Suppot redirects (from HTTP to HTTPS for example)

Application Load Balancer (v2)

- Routing tables to different target groupsusers
  - Routing based on path in URL (example.com/ & example.com/posts)
  - Routing based on hostname in URL (one.example.com & other.example.com)
  - Routing based on Query String, Headers (example.com/users?id=123&order=false)
- ALB are a great fit for micro services & container-based application (example: Docker & Amazon ECS)
- Has a port mapping feature to redirect to a dynamic port in ECS
- In comparison, we'd need multiple Classic Load Balancer per application

Application Load Balancer (v2)
HTTP Based Traffic



Application Load Balancer (v2)

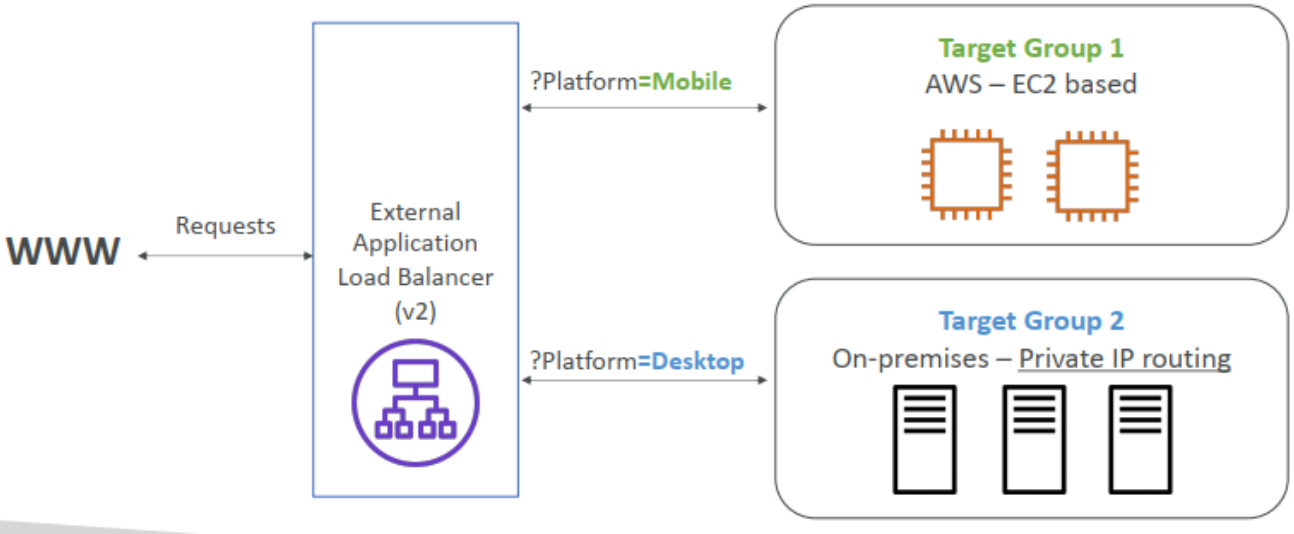
Target Groups

- EC2 instances (can be managed by an Auto Scaling Group) – HTTP
- ECS task (managed by ECS itself) – HTTP
- Lambda functions – HTTP request is translated into a JSON event
- IP Addresses – must be private Ips

- ALB can route to multiple target groups
- Health checks are at the target group level

Application Load Balancer (v2)

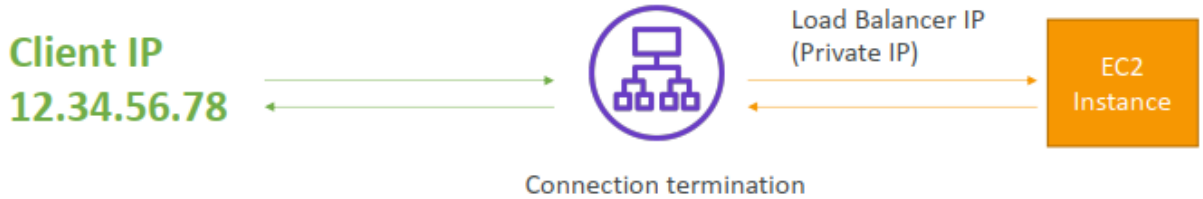
Query Strings/Parameters Routing



Application Load Balancer (v2)

Good to Know

- Fix hostname (XXX. Region.elb.amazonaws.com)
- The application servers don't see the IP of the client directly
  - The true IP of the client is inserted in the head X-Forwarded-For
  - We can also get Port (X-Forwarded-Port) and proto (X-Forwarded-Proto)

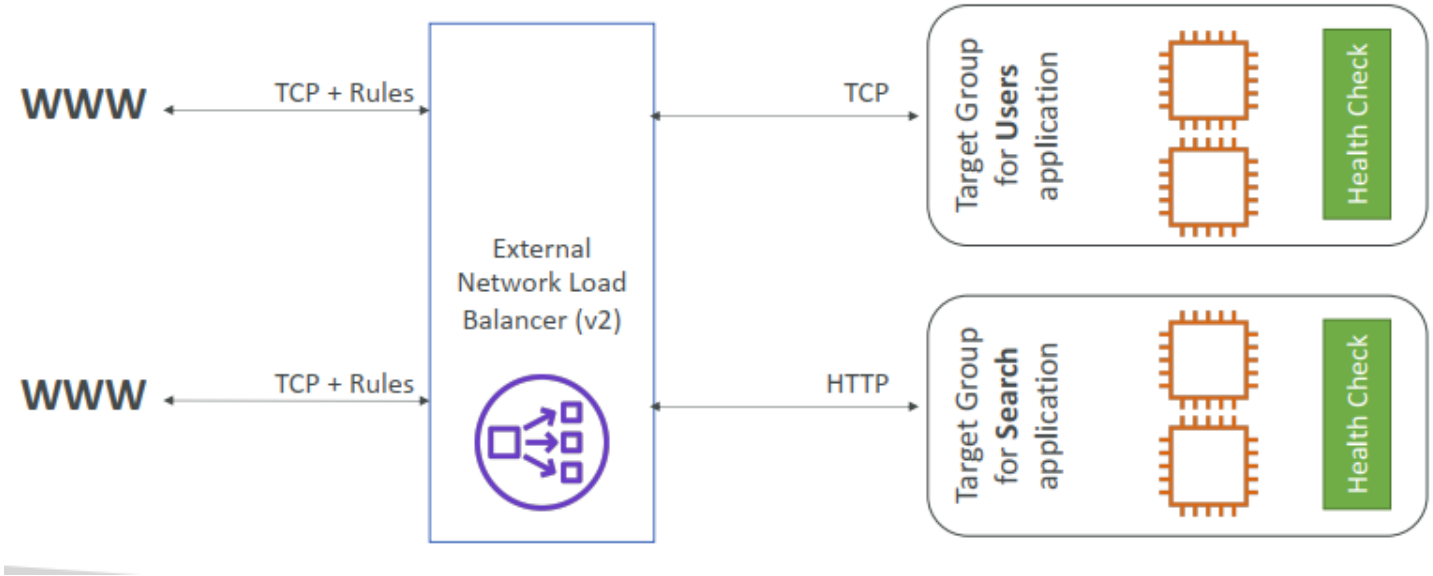


Network Load Balancer (v2)

- Network load balancers (Layer 4) allow to:
  - **Forward TCP & UDP traffic to your instances**
  - Handle millions of request per seconds
  - Less latency ~ 100 ms (vs 400 ms for ALB)

- **NLB has one static IP per AZ, and supports assigning Elastic IP** (helpful for whitelisting specific IP)
- NLB are used for extreme performance, TCP or UDP traffic
- Not included in the AWS free tier

Network Load Balancer (v2) – TCP (Layer 4) Based Traffic

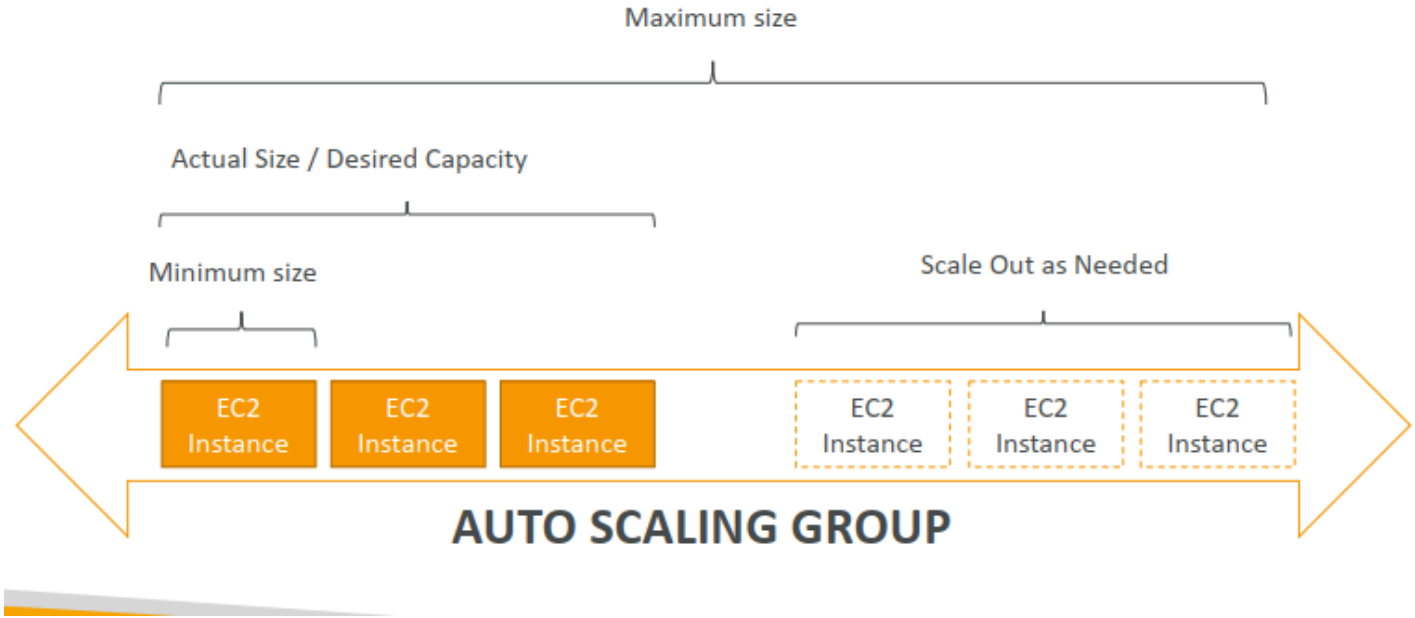


Auto Scaling Group

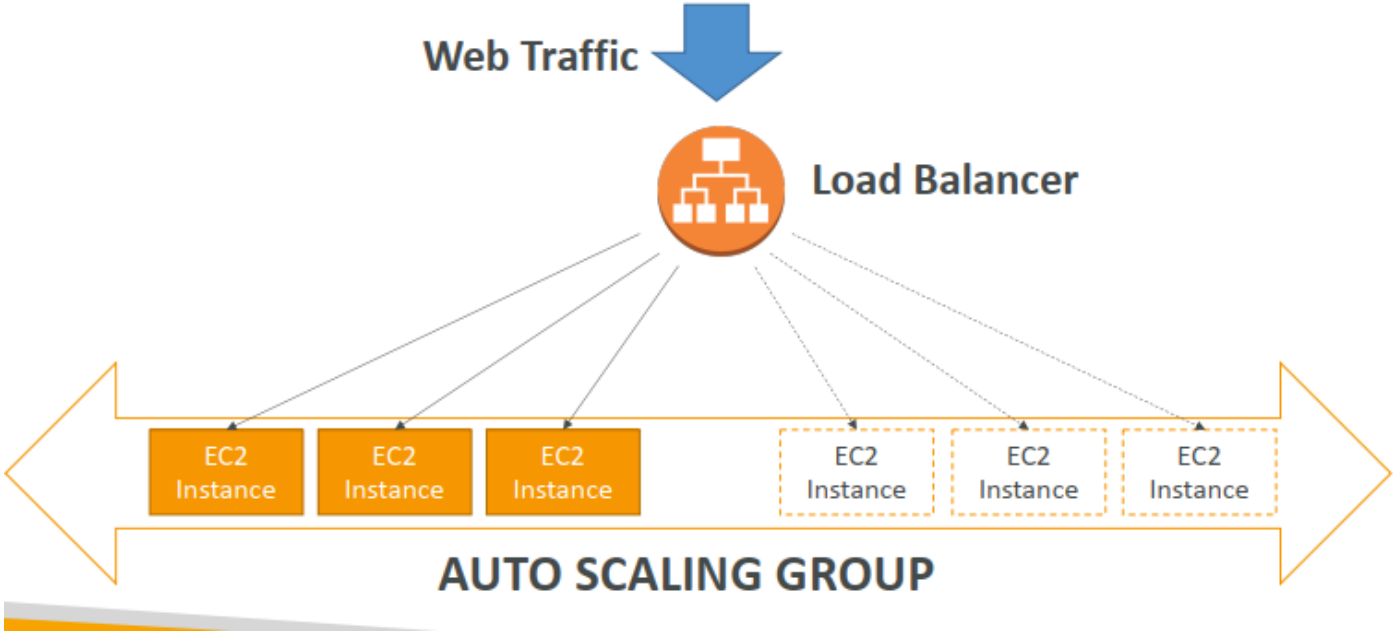
What' s an Auto Scaling Group?

- In real-life, the load on your websites and application can change
- In the cloud, you can create and get rid of servers very quickly
- The goal of an Auto Scaling Group (ASG) is to
  - Scale out (add EC2 instances) to match an increased load
  - Scale in (remove EC2 instances) to match a descread load
  - Ensure we have a minimum and a maximum number of machines running
  - Automatically Register new instances to load balancer

Auto Scaling Group in AWS



Auto Scaling Group in AWS with Load Balancer



ASGs have the following attributes

- A **launch configuration**
  - AMI + Instance Type
  - EC2 User Data
  - EBS Volumes
  - Security Groups/ SSH Key Pair
- Min Size/ Max Size / Initial Capacity
- Network + Subnets Information
- Load Balancer Information
- Scaling Policies

Auto Scaling Alarms

- It is possible to scale an ASG based on CloudWatch alarms
- An Alarm monitors a metric (such as Average CPU)
- Metrics are computed for the overall ASG instances
- Based on the alarm:
  - We can create scale-out policies (increase the number of instances)
  - We can create scale-in policies (decrease the number of instances)

Auto Scaling New Rules

- It is now possible to define “better” auto scaling rules that are directly managed by EC2
  - Target Average CPU Usage
  - Number of requests on the ELB per instance
  - Average Network In
  - Average Network Out
- These rules are easier to set up and can make more sense

Auto Scaling Custom Metric

- We can auto scale based on custom metric (ex: number of connected users)
- 1. Send custom metric from application on EC2 to CloudWatch (PutMetric API)
- 2. Create CloudWatch alarm to react to low/high values
- 3. Use the CloudWatch alarm as the scaling policy for ASG

ASG Brain Dump

- Scaling policies can be on CPU, Network… and can even be on custom metrics or based on a schedule (if you know your visitors patterns)
- ASGs use Launch configurations or **Launch Templates** (newer)
- To update an ASG, you must provide a new launch configuration/launch template
- IAM roles attached to an ASG will get assigned to EC2 instaces
- ASG are free. You pay for the underlying resources being launched
- Having instances under an ASG means that if they get terminated for whatever reason, the ASG will automatically create new ones as a replacement. Extra safety!
- ASG can terminate instancese marked as unhealthy by an LB

Auto Scaling Groups – Scaling Policies

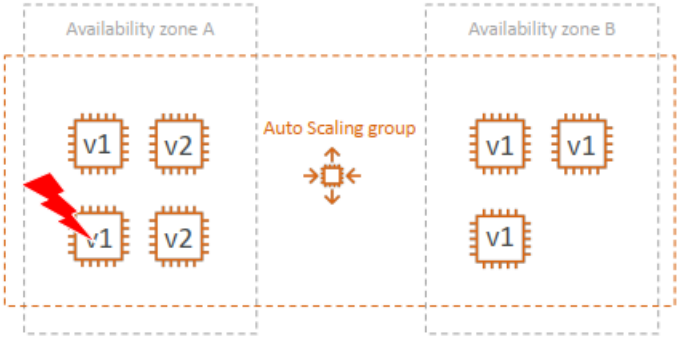
- Target Tracking Scaling
  - Most simple and easy to set-up
  - Example: I want the average ASG CPU to stay at around 40%
- Simple / Step Scaling
  - When a CloudWatch alarm is triggered (example CPU>70%), then add 2 units
  - When a CloudWatch alarm is triggered (example CPU<30%), then remove 1
- Scheduled Actions
  - Anticipate a scaling based on known usage patterns
  - Example: increase the min capacity to 10 at 5pm on Fridays

Auto Scaling Groups – Scaling Cooldowns

- The cooldown period helps to ensure that your Auto Scaling groups doesn't launch or terminate additional instances before the previos scaling activity takes effect
- If the default cooldown period of 300 seconds is too long – you can reduce cost by applying a scaling-specific cooldown period of 180 seconds to the scale-in policy

ASG for Solutions Architects

- ASG Default Termiantion Policy (simplified version)
  - Find the AZ which have the most number of instance
  - IF there are multiple instances in the AZ to choose from, delete the one with the oldest launch configuration
- ASG tries the balance the number of instances across AZ by default



ASG for Solutions Architects – Launch Template & Launch Configuration

- **Boths**
  - ID of the AMI, the instance type, a key pair, security groups, and the other parameters that you use to launch EC2 instance (tags, EC2 user-data, ..)
- **Launch Configuration (legancy)**
  - Must be re-created every time
- **Launch Template (newer)**
  - Can have multiple versions
  - Create parameters subsets
  - Provision using both On-Demand and Spot instances (or a mix)
  - Can use T2 unlimited burst feature
  - Recommend by AWS going forward

Thank you!!!