**AUTO SCALLING GROUP**

In real-life, the load on your websites and application can change

In the cloud, you can crate and get rid of servers very quickly

Scale out and scale in EC2 instance based on the load

Scale out 🡪 adding instance based on the requirement

Scale in 🡪 Deleting instance based on the requirement

Ensure we have a minimum and a maximum number of machines running

Automatically Register new instances to a load balancer

Whenever there is a dement on traffic Autoscaling will be ***scale Out and Scale In***

**MIN 🡪** The min number of EC2 instances that the ASG should have Ex: 2

**MAX 🡪** The max number of EC2 instances that the ASG should have Ex: 6

**DESIRED🡪** The number of EC2 instance that you wish / desired to launch initially Ex: 4

We have 3 types of scaling options

**Manual Scaling**

If you are manually modifying Min and Max/Desired capacity

**Scheduled Scaling**

Based on time period of day

**Dynamic Scaling**

Based on the load (CPU, no of requests, network)

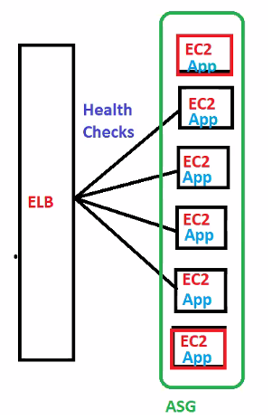
**Launch Configuration**

[AMI application+ Volume +SG +AZ + Tags+ Instance Type etc]

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

ELB does the health check for the application

CloudWatch will monitor EC2 instance



ASG = ELB +EC2 Instance + Launch Configs + Cloud Watch +SNS

**Why we need Elastic Load Balancer**

An ELB (EC2 Load Balancer) is a managed load balancer

AWS guarantees that it will be working

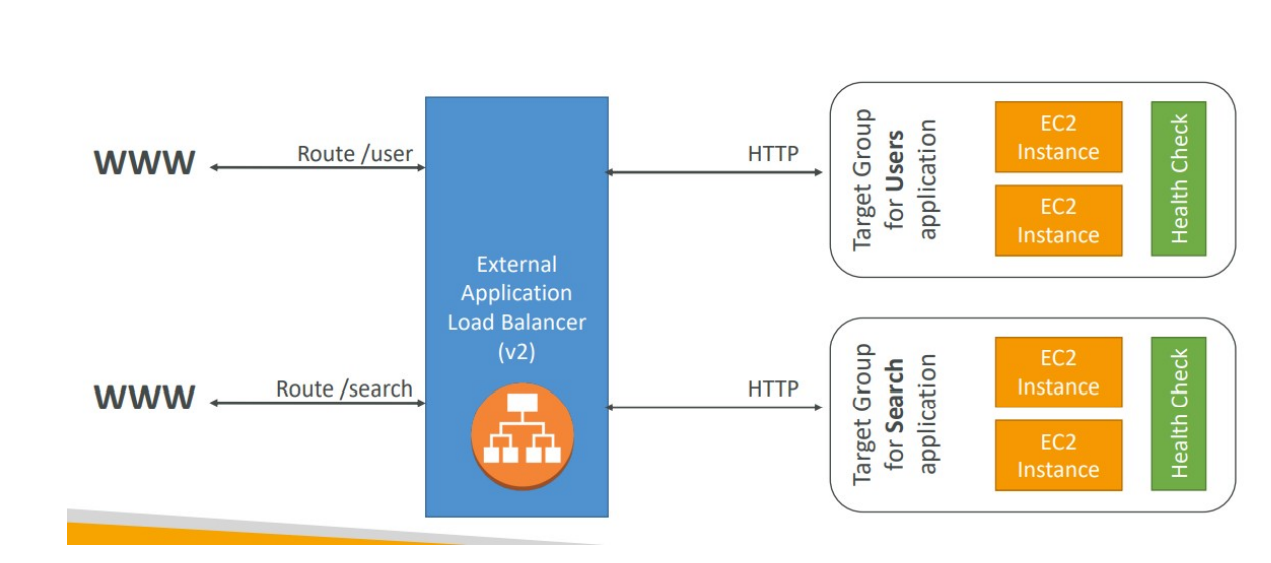
AWS takes care of upgrades, maintenance, high availability

AWS has 3 kinds of Load Balancers

* Classic Load Balancer (v1 - old generation) - 2009
* Application Load Balancer (v2 - new generation) – 2016
* Network Load Balancer (v2 - new generation) – 2017

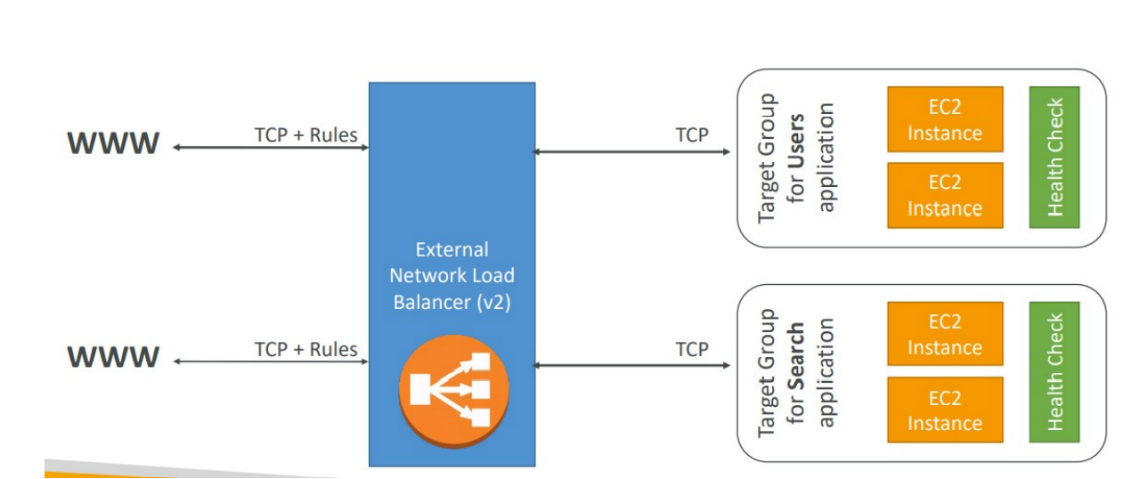
**Application Loan Balancer:**

* Load balancing to multiple HTTP applications across machines (target groups)
* Load balancing to multiple applications on the same machine (ex: containers)
* Load balancing has path based and host based in URL
* Support routing based on hostname (users.example.com & payments.example.com)
* Support routing based on path (example.com/users & example.com/payments)
* Basically, they’re awesome for micro services & container-based application (example: Docker & Amazon ECS)
* In comparison, we would need to create one Classic Load Balancer per application before. That was very expensive and inefficient!
* Application Load Balancer works on HTTP and HTTPS

****

**Network Load Balancer**

* Network load balancers (Layer 4) allow to do:
* Forward TCP traffic to your instances
* Handle millions of requests per seconds
* Support for static IP or elastic IP
* Less latency ~100 ms (vs 400 ms for ALB)
* Network Load Balancers are mostly used for extreme performance and should not be the default load balancer you choose



**Classic Load Balancer**

* Classic Load Balancer works on HTTP , HTTP and TCP but no routes(path and host based)

**Why we need Elastic Load Balancer**

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

**Load Balancer – Health Checks**

* Health Checks are crucial for Load Balancers
* They enable the load balancer to know if instances it forwards traffic
* to are 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

**Load Balancers**

* Classic Load Balancers are Deprecated
* Application Load Balancers for HTTP / HTTPs
* Network Load Balancer for TCP
* CLB, ALB & NLB support SSL certificates and provide SSL termination
* All Load Balancers have health check capability
* ALB can route on based on hostname / path
* ALB is a great fit with ECS (Docker)
* Any Load Balancer (CLB, ALB, NLB) has a static host name. Do not resolve and use underlying IP

**Load Balancers Stickiness**

* It is possible to implement stickiness so that the same

client is always redirected to the same instance behind a load balancer

This works for Classic Load Balancers & Application Load Balancers

