**AWS Fundamentals ELB+ASG**

**SCalability and high availability**

Adapt greater loads

Vertical scalability

* Increasing size of instance
* egs:Make better hardware
* Common for non distributed system like database
* RDS,ElastiCache can scale vertically
* There is a limit
* T2.nano to u-12tbl.metal

Horizontal scaling

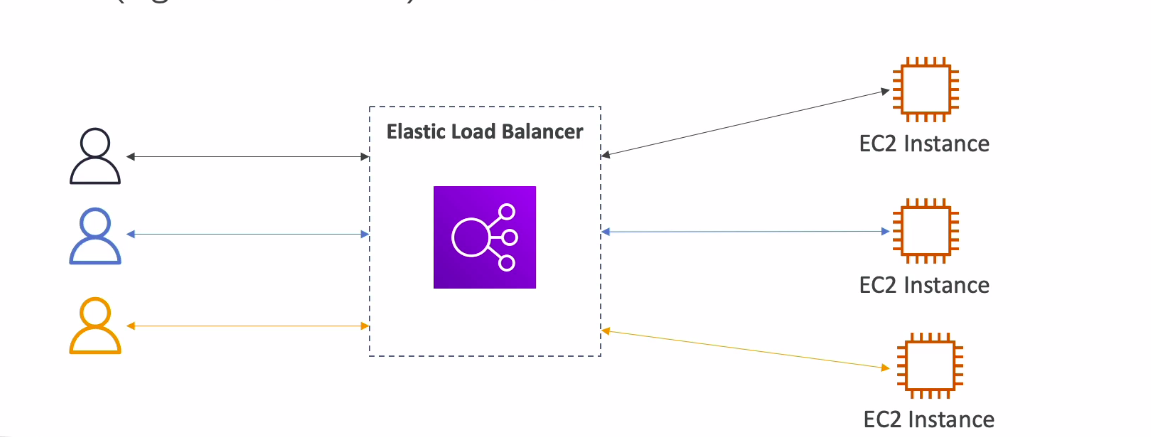
* Increasing the number of instances
* Implies distributed systems
* Aautoscaling group and load balancer

High availability

* Running application in atleast 2 data centers(AZ)
* Goal is ro survive a data center loss
* Can be both active and passive
* Autoscaling multi azs

**Load Balancing**

* THey are Servers that forward traffic to multiple servers
* Load is balanced across multiple ec2 instances



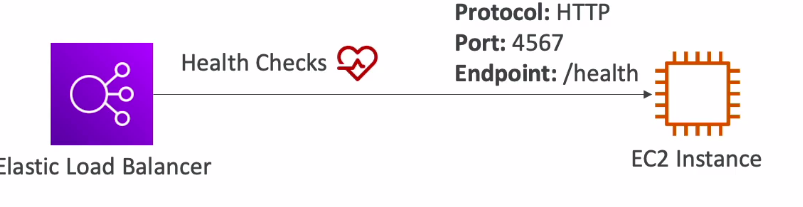
Why load balancer?

* Spread load
* Expose a single point of access
* Handle failures of downstream instances
* Regular health checks
* SSL termination for your webstes
* Enforce stickiness
* High availability across zones
* Separate public traffic from private traffic

**Elastic load balancer**

* AWS managed load balancer
* AWS provides only a few configuration options
* Costs less to setup you own load balancer but a lot more effort (scalability issues also)
* Integrates with other aws offerings

**Health CHecks**

* Enables the load balancer to know if a particular instance is working properly or not.
* Health check is done on a port and a route(/health is a common endpoint)
* If the response from the instance is not 200 then the instance is unhealthy and the load balancer will not send requests to that instance.

**Types of Load balancer**

1)Classic LB

2)Application LB(New gen)-Http,Https,websocket

3)Network LB(New gen)-TCP,TLS,UDP

4)Gateway load balancer- operates at the network layer- IP

Load balancer can be setup as internal or external

**Load balancer security group**

* Users can access load balancer from anywhere- Load balancer has its own security group
* The EC2 instance attached to the load blancer will have its own security group which will allow traffic only from that load balancer.
* Enhances security mechanism

**—------------------------------------------------------------------------------------------**

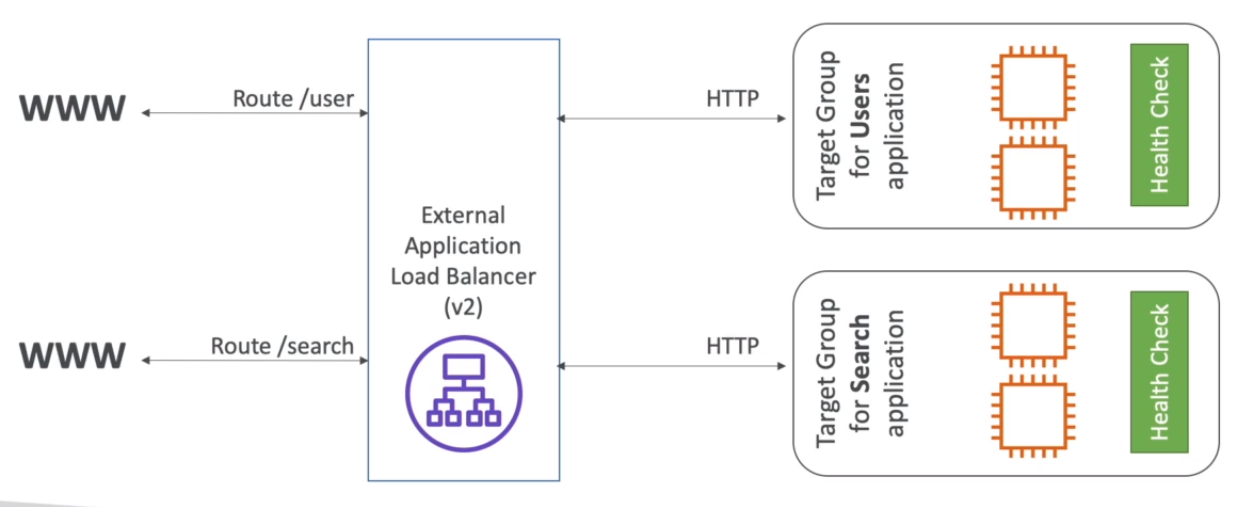
**Classic load balancers**

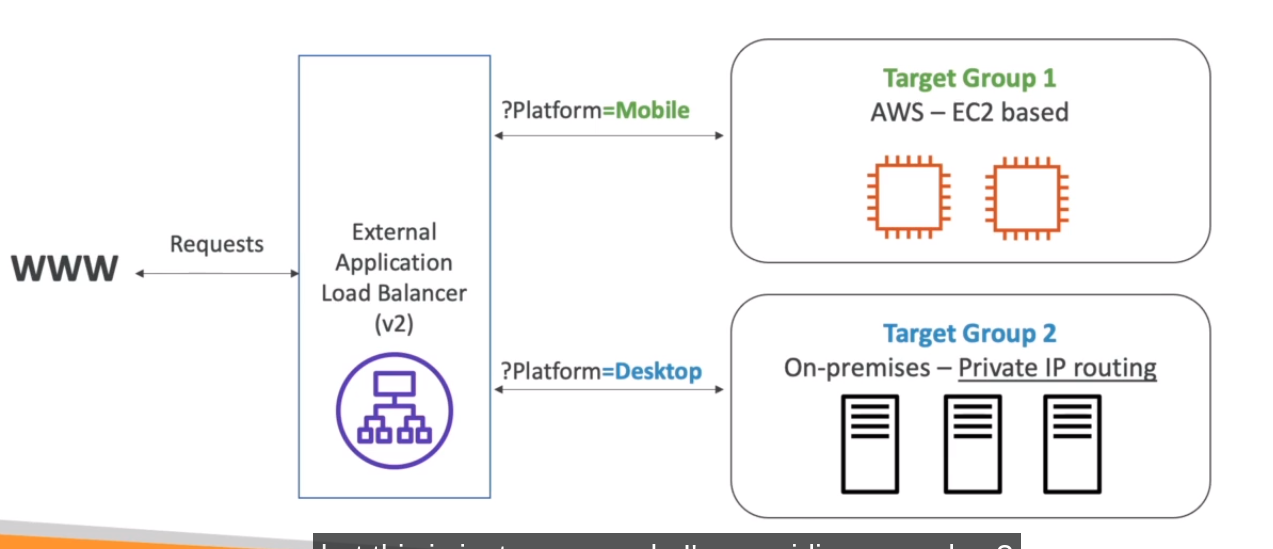
* Supports TCP(Layer 4), HTTP and HTTPS(layer7)
* Health checks are tcp or http based
* We get a fixed host name

Hands on:-

—---------------------------------------------------------------------------------------------

Application Load Balancer(v2)

* Layer 7(HTTP)
* Load balancing to multiple applications on the same machine
* Supports HTTP/2 abd websocket
* Supports redirects (from HTTP to HTTPS) automatically
* Routing tables to different target groups:
  + Routing based on path in URL (examples.com/users & example.com/posts)
  + Routing based on the hostname in URL(one.example.com& other.example.com)
  + Based on query string, headers(example.com/users?id=123&order=false)
* **ALB are great fit for microservices and container based applications**
* **Would need multiple classic load balancers per application**
* Health checks are done at target group level
* ALB can route multiple target groups

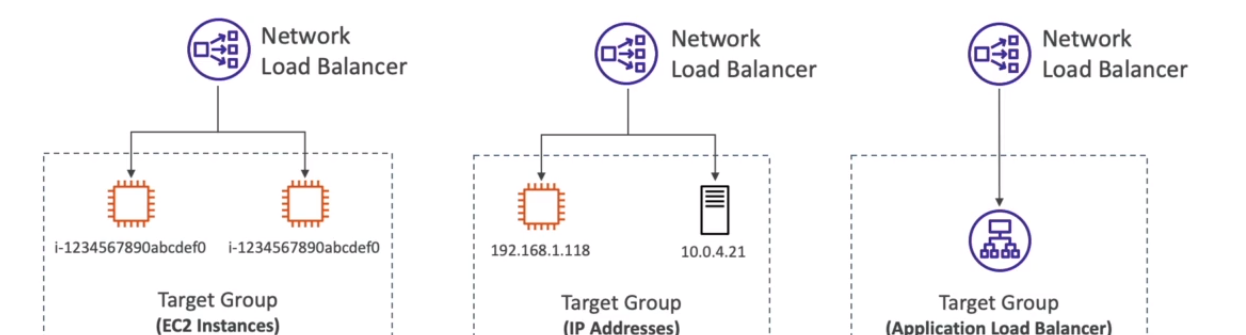


* Get fixed hostname
* App server dont see the IP of the client directly (xport)

Hands on-

**—----------------------------------------------------------------------------------------**

**Network Load Balancer (NLB)**

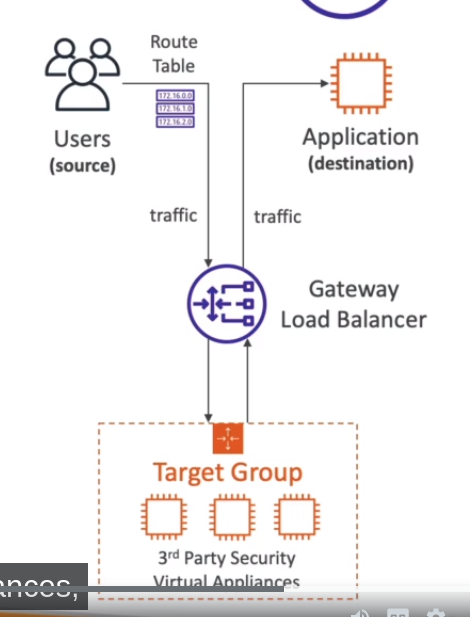
* Layer 4
* Handle millions of request per seconds
* Less latency
* HAs one static Ip per AZ (unlike fixed host name in ALB and CLB)
* Supports elastic IP
* **Used for extreme performance(TCP or UDP traffic)**
* TCP+rules traffic 

**Hands on-**

**—--------------------------------------------------------------------------------**

**Gateway load Balancer**

* Deploy,scale and manage a fleet of 3rd Party network virtual appliances in AWS like firewalls,intrusion detection and prevention systems.
* Operates on level 3
* Uses **GENEVE Protocol** on PORT 6081



**—-----------------------------------------------------------------------**

**Sticky sessions**

* Same client redirected to the same instance
* CLB and ALB
* Cookies used to implement stickiness.
* Makes sure user doesn't lose session data
* May bring imbalance to the load over the backend ec2 instances

**Types of cookies:**

* Application based cookies
* Duration based cookie

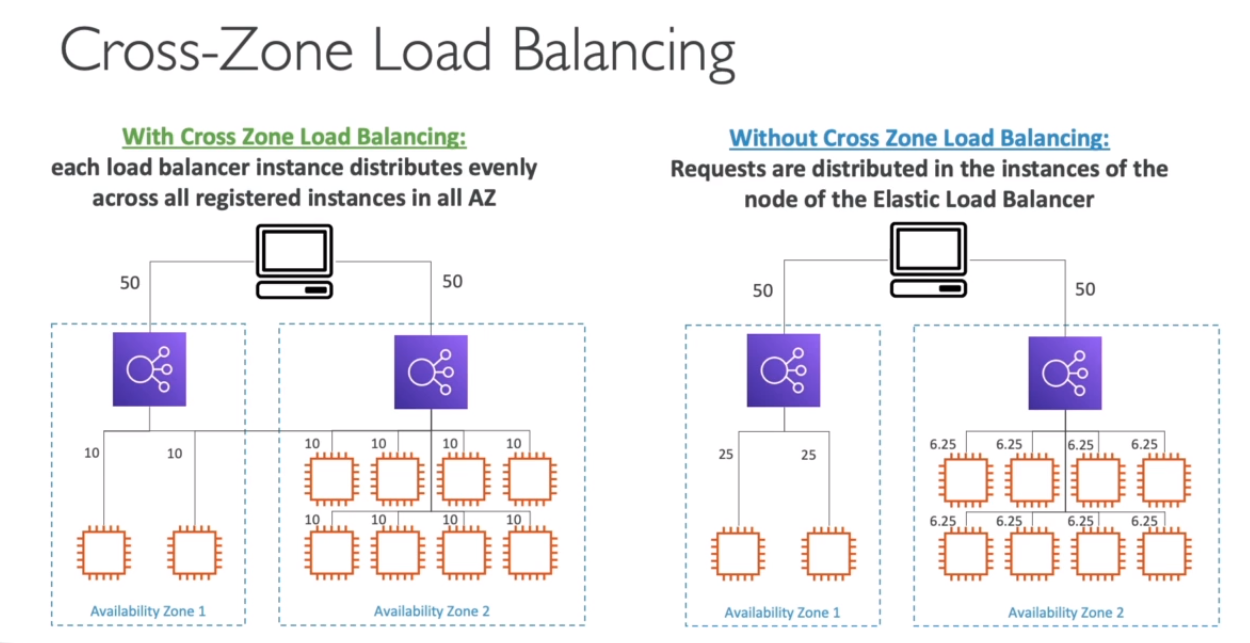
**Application based Cookies**

* Custom cookie
* Application cookie

**—-------------------------------------------------------------------------------**

**Cross-zone Load Balancing**

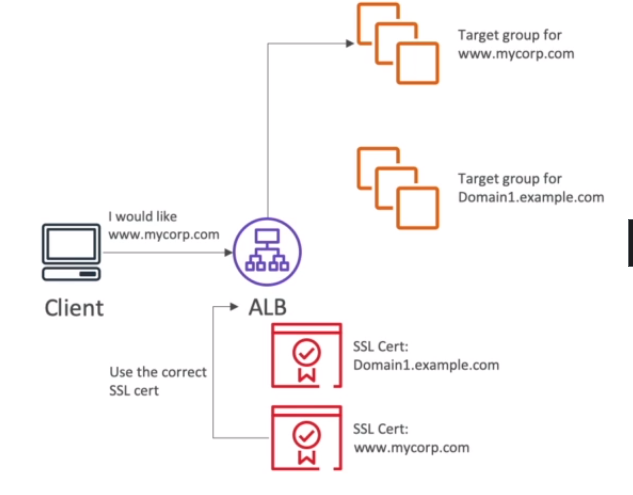
* Each load balancer distributes evenly across all registered instances in all AZ’s



* **ALB**
  + Always on(Cant be disabled)
  + No charges for inter AZ data
* **NLB**
  + Disabled by Data
  + Pay charges for inter AZ transfer
* **CLB**
  + Disabled by default
  + No charges

**—---------------------------------------------------------------------**

**SSL certificates**

* ALlows data to be encrypted in transit (user-load balancer).
* SSL- secure socket layer
* TLS - transport layer security
* Certificates are issued by certification Authority
* LB uses x.509 certificates
* Manage certificates using ACM (AWS certification manager)
* **SNI - Server Name indication**
  + Solves the problem of loading multiple SSL certificates onto one web server
  + Requires client to indicate host name of target server
  + Server finds the correct certificate, or returns the default one.
* CLB
  + Supports only one SSL certificate
  + Need to use multiple clb for multiple SSL certificates
* ALB
  + Supports Multiple listeners with multiple SSL certificates
  + Uses SNI
* NLB - same as ALB

**—------------------------------------------------------------------**

**Connection draining**

* An instance which is draining is given some time to handle the request already present and new users are not sent to that instance
* ALso called deregistration delay

**—-----------------------------------------------------------------------**

**Auto scaling group**

* Load on your website can change. On cloud you can add or get rid of servers very easily
* Goal of an ASG is :
  + Scale out (add ec2 instance)
  + Scale in (remove ec2 instance)
  + Ensure we have a min and max number of ec2 instances running
  + Automatically add new instances to the load balancer
  + Recreate ec2 instance in case it was terminated
* It is free( You only pay for the underlying ec2 instances)
* Possible to scale an asg based on cloudwatch alarm
* **Dynammic scaling policies**
  + Target tracking scaling
    - I want cpu to stay around 40%
  + simple/step scaling
    - Cloudwatch alarm triggered if(CPU>70%) add 2instance , if (CPU<30%) remove 1 instance.
  + Scheduled Actions
    - Anticipate scaling based on usage pattern
* Predictive scaling- based on forecast using ML
* **Metrics to scale on**
  + CPU Utilization
  + Request count per second
  + Average network in/out
* Scaling cooldown
  + After scaling , there is a cooldown period. ASG will not launch or terminate additional instances