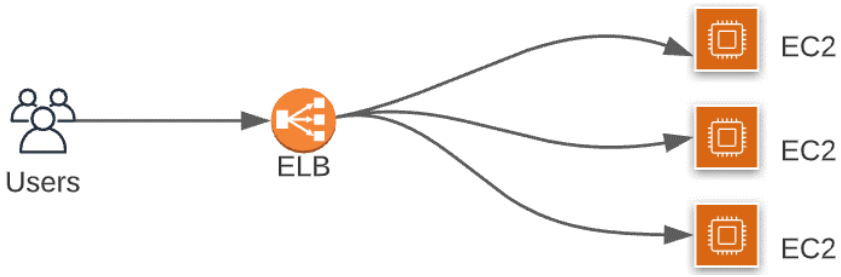
AWS Elastic Load Balancers (ELB)

What is load balancer :

- Servers/services that receives users traffic (public/private) and forward to multiple servers behind the load balancers.



Why load balancer is needed

- Distribute the load across multiple downstream EC2 instances
- Provides a single point of access to the application
- Handle single point of failure by performing health check of downstream instances
- Provides SSL termination/offloading for the application
- Provides high availability and fault tolerance for the application
- Allow users to enforce stickiness with cookies
- Provides extra layer of security by separating public and private traffic

AWS managed LB Vs own managed LB :

- AWS managed :
    - Comes with highly availability
    - Maintenance and upgrades is taken care by amazon
    - Comes with easy setup and configuration
    - Well integrated with many other aws services
    - Comes with basic monitoring based on CloudWatch metrics
- Own managed :
    - Required more effort on setup and configuration
    - Everything has to be managed by you

Types of AWS load balancers :

- Classic Load Balancer
- Application Load balancer
- Network Load Balancer
- Gateway Load Balancer

Use cases of ELBs :

- External (Public)
- Internal (Private)

How ELB works ? (Health check) :

- Load balancer works based on the health checks
- Always perform a health check of all its downstream instances based on the configuration parameters to check if they are healthy and ready to accept requests
- If the instance is healthy then it sends the traffic if not healthy then stop sending requests until it becomes healthy
- Health check can be configured on a port/protocol along with a path/route
- If the health check response is 200 then healthy else the instance will be treated by ELB as unhealthy

Classic Load Balancer : (Previous Generation)

- Layer-4/7 (Transport/Application) load balancer
- Supported ports are HTTP/HTTPS, TCP and SSL/TLS
- Target Type : Instance
- Supports SSL termination/offloading
- Support for sticky sessions
- Support for EC2-Classic
- **Used for :** single application architecture

Application Load Balancer : (Recommended)

- Layer-7 (Application) load balancer
- Supported protocols are HTTP/HTTPS and WebSocket
- Target Type : IP, Instance and lambda
- Support SSL termination/offloading
- Stickiness can be enabled at the target group level
    - Same request always goes to the same instance
    - Stickiness is generated by the ALB
- Traffic can be distributed to multiple application on the same instance (Containers)
- Traffic can be distributed across instances with multiple target groups
- Support path based routing (example.com/images)
- Support host based routing (example1.com/example2.com)
- It also has port mapping feature to redirect to a dynamic port
- **Used for :** Web App, Mobile App

Network Load balancer : (High performance and low latency)

- Layer-4 (Transport) load balancer
- Supported protocols are TCP, UDP and TLS
- Target Type : IP Instance
- It can handle millions of requests per second
- It supported with static IP (Elastic IP)
- Works with high performance and low latency (100ms) as compare to ALB (400ms)
- **Used for :** Latency sensitive application

Gateway Load Balancer : (3rd party appliances)

- Supported protocol is IP
- Layer-3/4 (Network/Transport) load balancer
- Target type : IP, Instance
- **Used for :** 3rd party appliances, such as Firewall, intrusion detection etc.

Important Options :

- Listener (Port/Protocol)
  - Frontend
  - Backend
- Target Group (Grouping instances to a target group)
- Target (an individual instance)
  - Instance
  - IP (Private IP)
  - Lambda
- Cross-zone load balancing
  - Distributes requests evenly across multiple AZs
  - Eliminates imbalance of EC2 instance utilization

Feature comparison between load balancers in more detail can be found on the below link

- https://aws.amazon.com/elasticloadbalancing/features/

LAB Test :

```
#!/bin/bash

sudo su
yum install httpd -y
systemctl enable httpd
mkdir /var/www/html/students/
echo "<h1> this page is from student app </h1>" > /var/www/html/students/index.html
systemctl start httpd
```

```
#!/bin/bash
sudo su
yum install httpd -y
systemctl enable httpd
mkdir /var/www/html/teachers/
echo "<h1> this page is from teachers app </h1>" > /var/www/html/teachers/index.html/
systemctl start httpd
```

```
#!/bin/bash
sudo su
yum update -y
yum install -y httpd
systemctl start httpd.service
systemctl enable httpd.service
echo "<h1> I am serving from $(hostname -f) </h1>" > /var/www/html/index.html
```