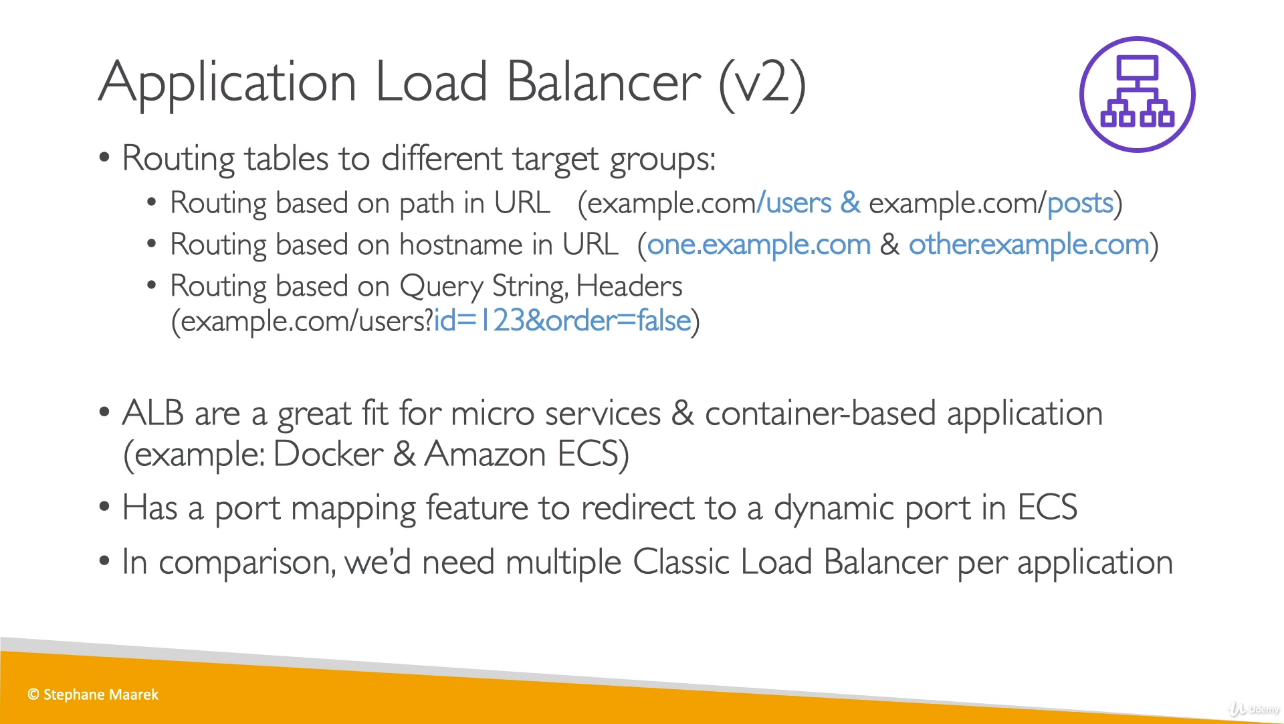
**Elastic Load Balancer (ELB) :**

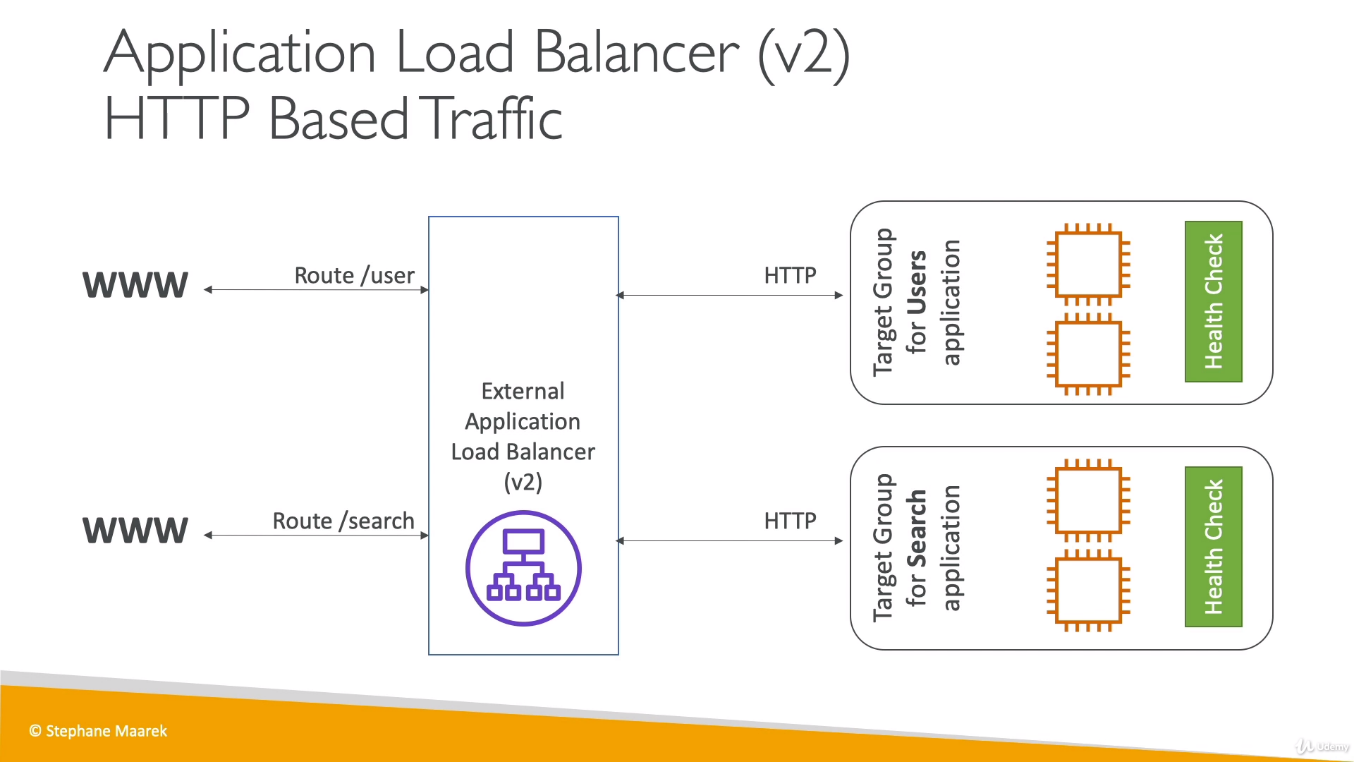
A load balancer distributes incoming application traffic across multiple EC2 instances in multiple Availability Zones. Elastic Load Balancing detects unhealthy instances and routes traffic only to healthy instances. Your load balancer serves as a single point of contact for clients. Elastic Load Balancing supports three types of load balancers: Application Load Balancers, Network Load Balancers, and Classic Load Balancers.

**Application Load Balancer:**

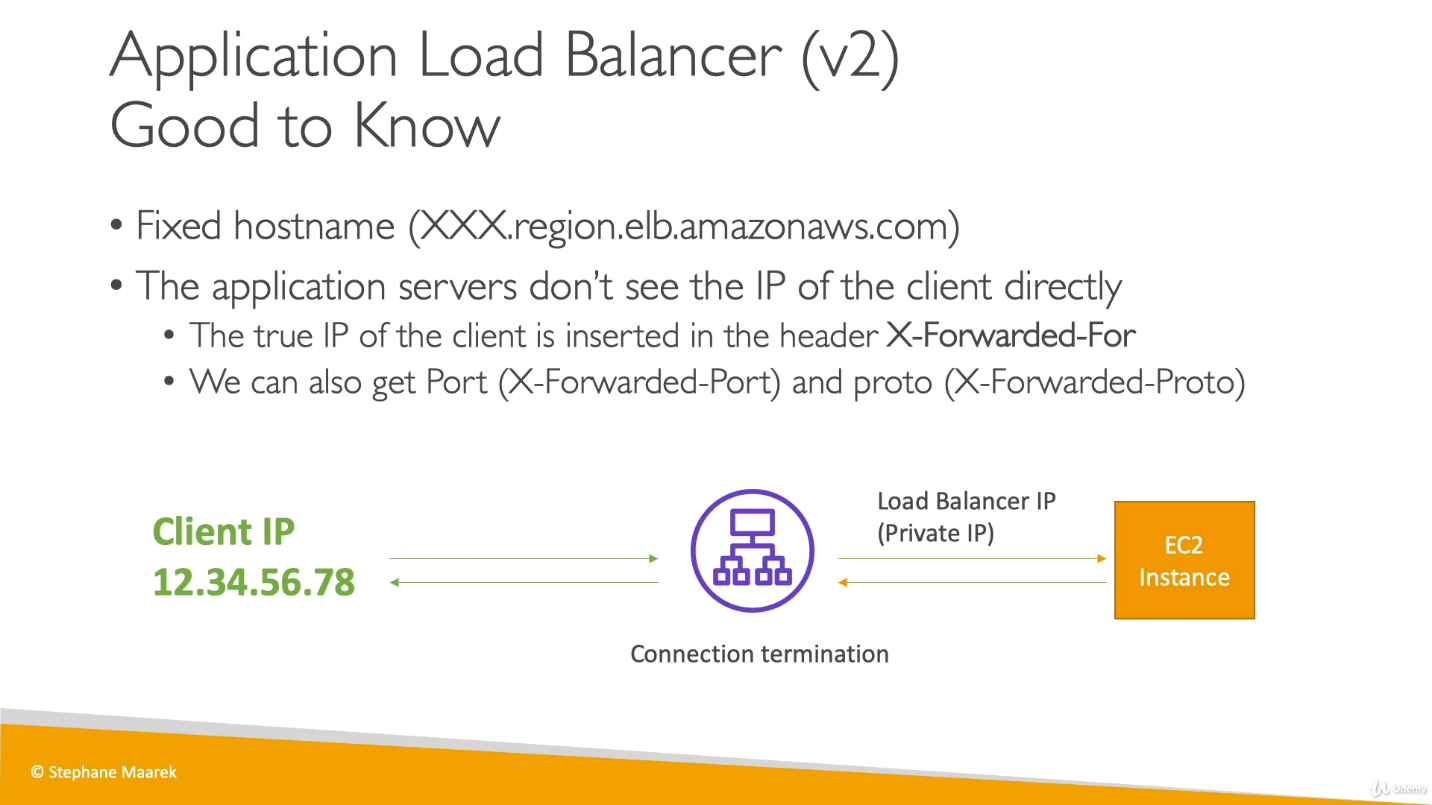
One ALB can have multiple target groups, each target group will filter the request based on path in URL, hostname in URL, parameter in URL.

ALB support Dynamic Port Binding with ECS (container service) where we know only request protocol not the port numbers.





Whenever any request comes from Browser, it will not go directly to EC2 instance, first it will come ALB and then ALB will send request to instance. We can find browser request details in URL’s header with parameters like X-Forwarded-For etc. This browser’s IP details might important to capture logs.



**Listener:**

Listener is a process that checks for connection requests, using the protocol and port that you configured. When we create an ALB, we define Listener which have a protocol like HTTP/HTTPS and a port number where ALB will listen from external browsers.

**Target Group:**

A target group tells a load balancer where to direct traffic to : EC2 instances, fixed IP addresses or AWS Lambda functions. When creating a load balancer, we create one or more listeners and configure listener rules to direct the traffic to one target group. Now we register target for our target group,  If, we register a target in an enabled Availability Zone, the load balancer starts routing requests to the targets as soon as the registration process completes, and the target passes the initial health checks.

So, in simple language we attach EC2 instances (one or more) to our Target group created for ALB*. [We have to do all these configurations while create an ELB]*

**Auto Scaling Group:**

Amazon EC2 Auto Scaling helps you ensure that you have the correct number of Amazon EC2 instances available to handle the load for your application. You create collections of EC2 instances, called *Auto Scaling groups*. You can specify the minimum number of instances in each Auto Scaling group, and Amazon EC2 Auto Scaling ensures that your group never goes below this size. You can specify the maximum number of instances in each Auto Scaling group, and Amazon EC2 Auto Scaling ensures that your group never goes above this size. If you specify the desired capacity, either when you create the group or at any time thereafter, Amazon EC2 Auto Scaling ensures that your group has these many instances. If you specify scaling policies, then Amazon EC2 Auto Scaling can launch or terminate instances as demand on your application increases or decreases.

Once we start creating an ASG, we have to first create a template/launch configuration (which will have EC2 instances and configuration for them) and then scaling group.

*We can specify one launch configuration for an EC2 Auto Scaling group at a time, and you can't modify a launch configuration after you've created it.*

So now, we have one or more instances under this ASG,

For ALB: We have target group, according to which traffic will go to different EC2 instances from ALB.

For ASG: We have ASG templates/ASG launch configuration, (configuration for EC2) which will be required to create ASG. Once this is setup, we can use already created one or more ELB (here it will ask to add Target group, so it will automatically understand by LB is associated with this Target group), so the traffic that was coming to ALB will be served by instances that was configured in ASG templates.

We can’t create a new Launch Template until we have old one.

ELB => Listener + Target Group + ASG

Source 🡪 ELB [filter requests based on query parameter/ path in URL/ hostname in URL] 🡪 client [EC2/ Lambda/ fixed IP]