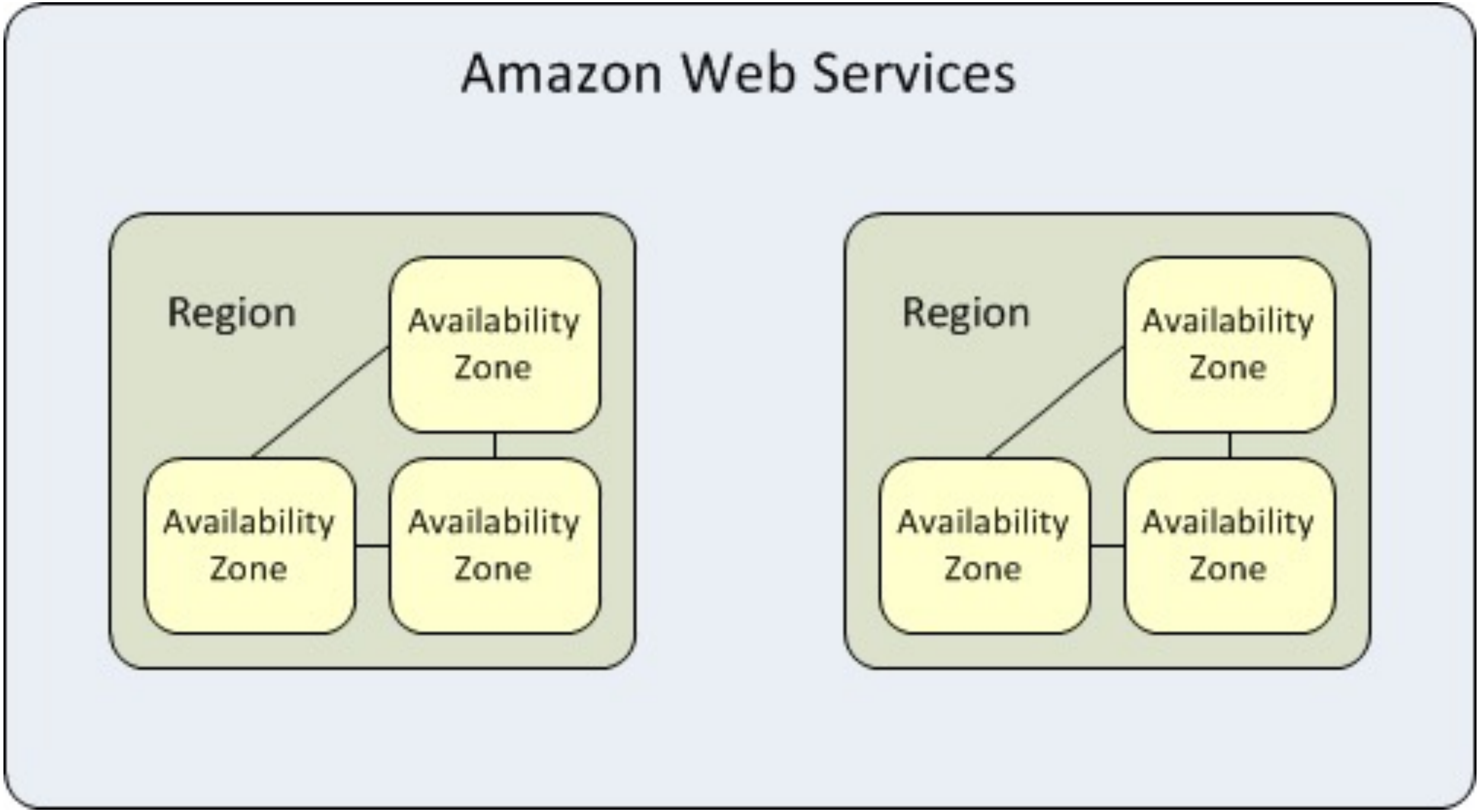
Load Balancing, Auto Scaling & Availability Zones

Tejas Parikh (t.parikh@northeastern.edu)
CSYE 6225
Northeastern University

Regions & Availability Zone


Amazon Web Services
Region
Availability Zone
Availability Zone
Availability Zone
Region
Availability Zone
Availability Zone
Availability Zone

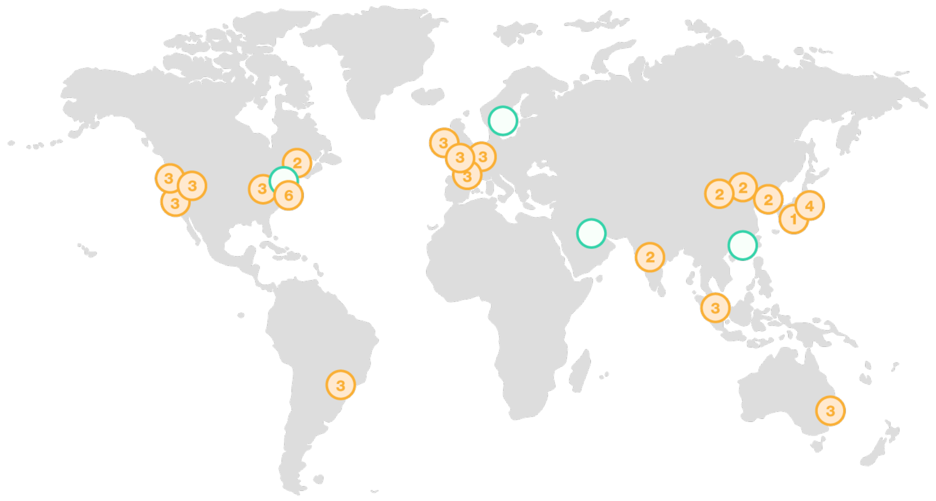
Regions

- Each Amazon EC2 region is designed to be completely isolated from the other Amazon EC2 regions. This achieves the greatest possible fault tolerance and stability.
- When you view your resources, you'll only see the resources tied to the region you've specified. This is because regions are isolated from each other, and AWS does not automatically replicate resources across regions.

Availability Zones

- When you launch an instance, you can select an Availability Zone or let AWS choose one for you.
  - If you distribute your instances across multiple Availability Zones and one instance fails, you can design your application so that an instance in another Availability Zone can handle requests.
- You can also use **Elastic IP** addresses to mask the failure of an instance in one Availability Zone by rapidly remapping the address to an instance in another Availability Zone.
- An Availability Zone is represented by a region code followed by a letter identifier; for example, us-east-1a.
  - To ensure that resources are distributed across the Availability Zones for a region, AWS independently map Availability Zones to identifiers for each account.
  - For example, your Availability Zone us-east-1a might not be the same location as us-east-1a for another account.

AWS Regions



● Region & Number of Availability Zones

US East
N. Virginia (6),
Ohio (3)

US West
N. California (3),
Oregon (3)

Asia Pacific
Mumbai (2),
Seoul (2),
Singapore (3),
Sydney (3),
Tokyo (4),
Osaka-Local (1)[1]

Canada
Central (2)

China
Beijing (2),
Ningxia (2)

Europe
Frankfurt (3),
Ireland (3),
London (3),
Paris (3)

South America
São Paulo (3)

AWS GovCloud (US-West) (3)

○ New Region (coming soon)

Bahrain

Hong Kong SAR, China

Sweden

AWS GovCloud (US-East)

Code	Name
us-east-1	US East (N. Virginia)
us-east-2	US East (Ohio)
us-west-1	US West (N. California)
us-west-2	US West (Oregon)
ca-central-1	Canada (Central)
eu-central-1	EU (Frankfurt)
eu-west-1	EU (Ireland)
eu-west-2	EU (London)
eu-west-3	EU (Paris)
ap-northeast-1	Asia Pacific (Tokyo)
ap-northeast-2	Asia Pacific (Seoul)
ap-northeast-3	Asia Pacific (Osaka-Local)
ap-southeast-1	Asia Pacific (Singapore)
ap-southeast-2	Asia Pacific (Sydney)
ap-south-1	Asia Pacific (Mumbai)
sa-east-1	South America (São Paulo)

Load Balancing

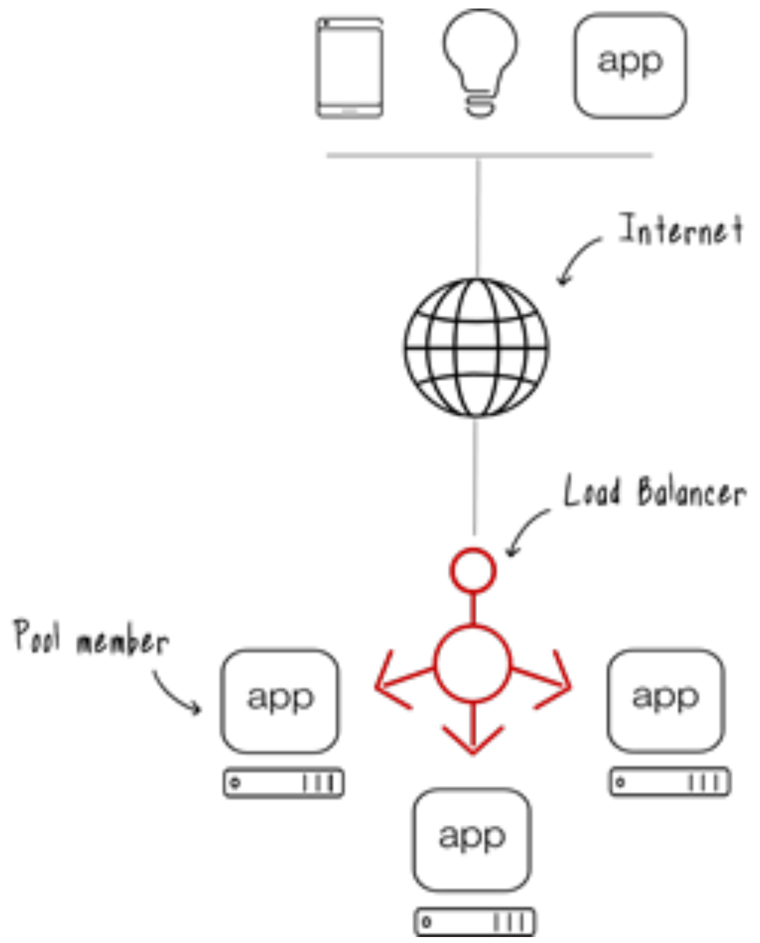
Can you check the load balancer?

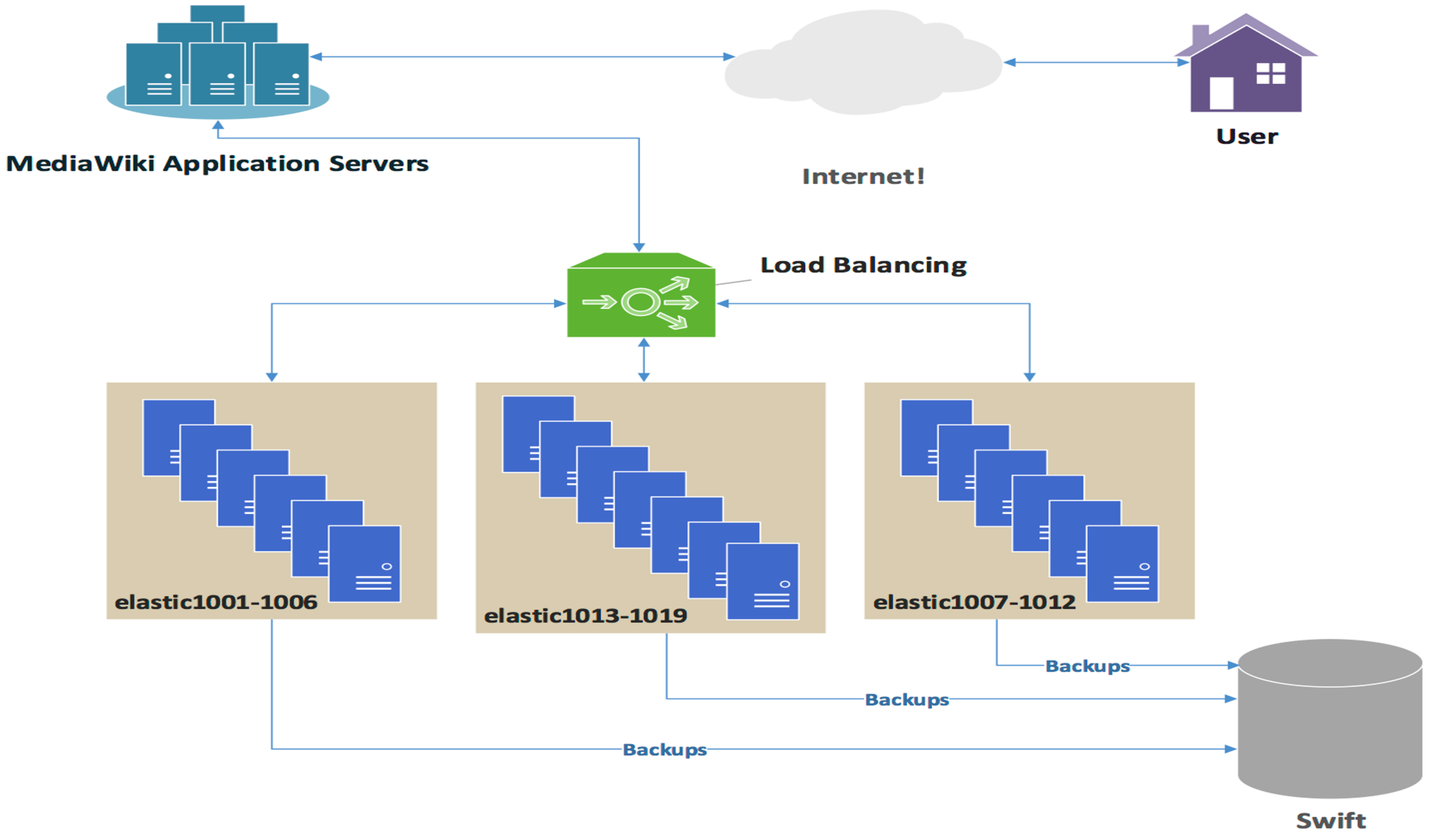
What is Load Balancer?

A load balancer is a device that acts as a reverse proxy and distributes network or application traffic across a number of servers.

Why Use Load Balancers?

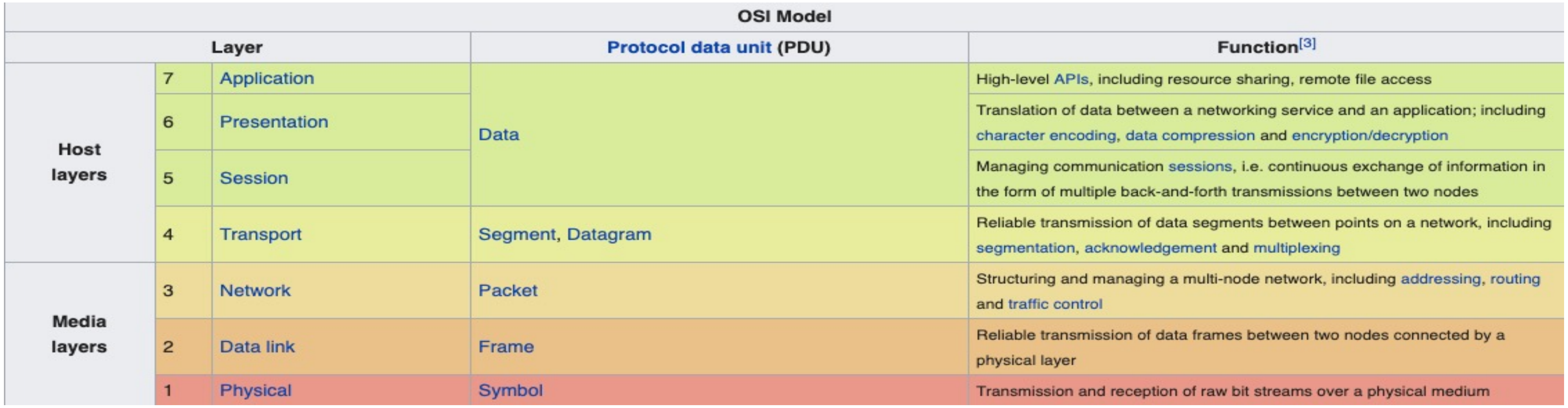
- Load balancers are used to increase capacity (concurrent users) and reliability of applications.
- They improve the overall performance of applications by decreasing the burden on servers associated with managing and maintaining application and network sessions, as well as by performing application-specific tasks.
- Load balancers ensure reliability and availability by monitoring the "health" of applications and only sending requests to servers and applications that can respond in a timely manner.




MediaWiki Application Servers
Internet!
User
Load Balancing
elastic1001-1006
elastic1013-1019
elastic1007-1012
Backups
Backups
Backups
Swift

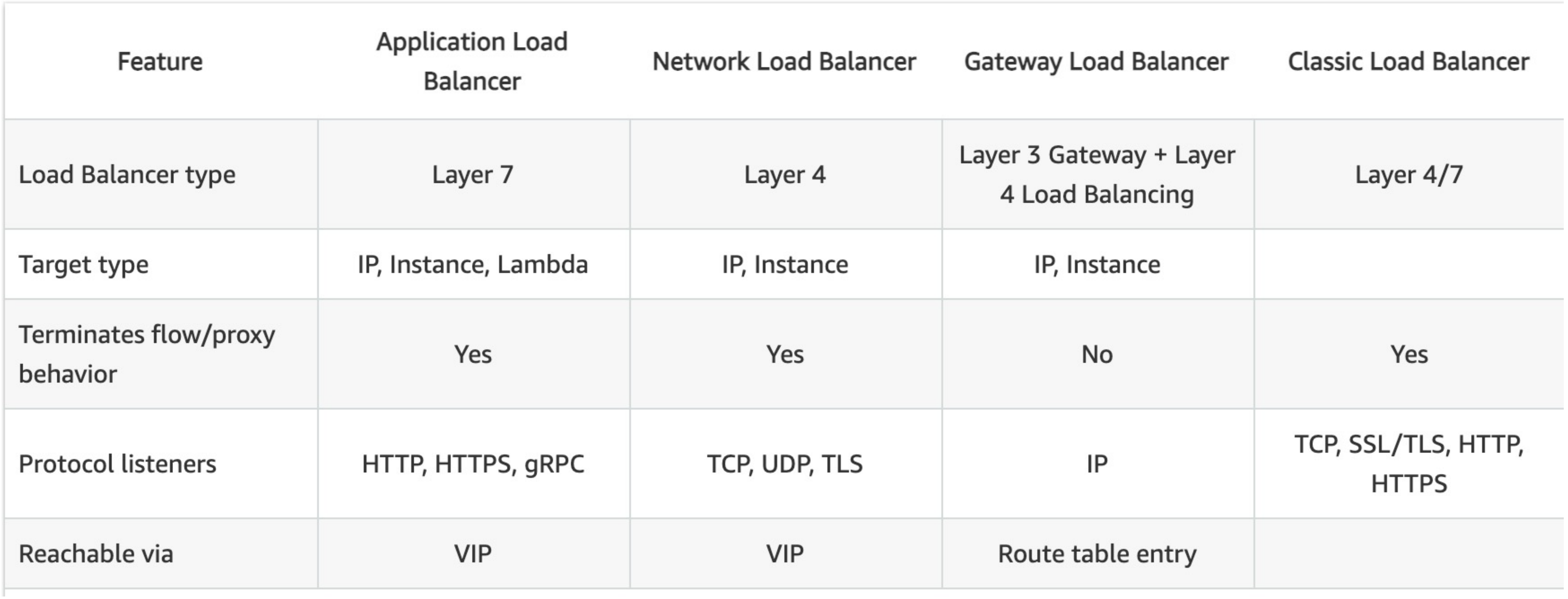
Types of Load Balancers

- **Layer 4 Load Balancers** - Layer 4 load balancers act upon data found in network and transport layer protocols (IP, TCP, FTP, UDP).
- **Layer 7 Load Balancers** - Layer 7 load balancers distribute requests based upon data found in application layer protocols such as HTTP.



OSI Model				
Layer			Protocol data unit (PDU)	Function[3]
Host layers	7	Application	Data	High-level APIs, including resource sharing, remote file access
	6	Presentation		Translation of data between a networking service and an application; including character encoding, data compression and encryption/decryption
	5	Session		Managing communication sessions, i.e. continuous exchange of information in the form of multiple back-and-forth transmissions between two nodes
	4	Transport	Segment, Datagram	Reliable transmission of data segments between points on a network, including segmentation, acknowledgement and multiplexing
Media layers	3	Network	Packet	Structuring and managing a multi-node network, including addressing, routing and traffic control
	2	Data link	Frame	Reliable transmission of data frames between two nodes connected by a physical layer
	1	Physical	Symbol	Transmission and reception of raw bit streams over a physical medium

AWS Load Balancer Comparision



Feature	Application Load Balancer	Network Load Balancer	Gateway Load Balancer	Classic Load Balancer
Load Balancer type	Layer 7	Layer 4	Layer 3 Gateway + Layer 4 Load Balancing	Layer 4/7
Target type	IP, Instance, Lambda	IP, Instance	IP, Instance	
Terminates flow/proxy behavior	Yes	Yes	No	Yes
Protocol listeners	HTTP, HTTPS, gRPC	TCP, UDP, TLS	IP	TCP, SSL/TLS, HTTP, HTTPS
Reachable via	VIP	VIP	Route table entry	

Request Distribution

- Requests are received by both types of load balancers and they are distributed to a particular server based on a configured algorithm. Some industry standard algorithms are:
  - Round robin
  - Weighted round robin
  - Least connections
  - Least response time
- Layer 7 load balancers can further distribute requests based on application specific data such as HTTP headers, cookies, or data within the application message itself, such as the value of a specific parameter.

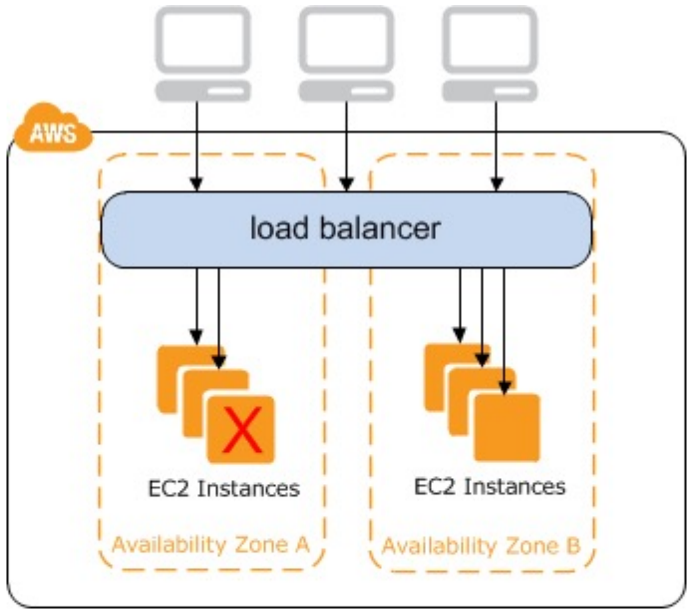
Classic Load Balancer

AWS Classic Load Balancer

The Classic Load Balancer routes traffic based on application or network level information and is ideal for simple load balancing of traffic across multiple EC2 instances where high availability, automatic scaling, and robust security are required.

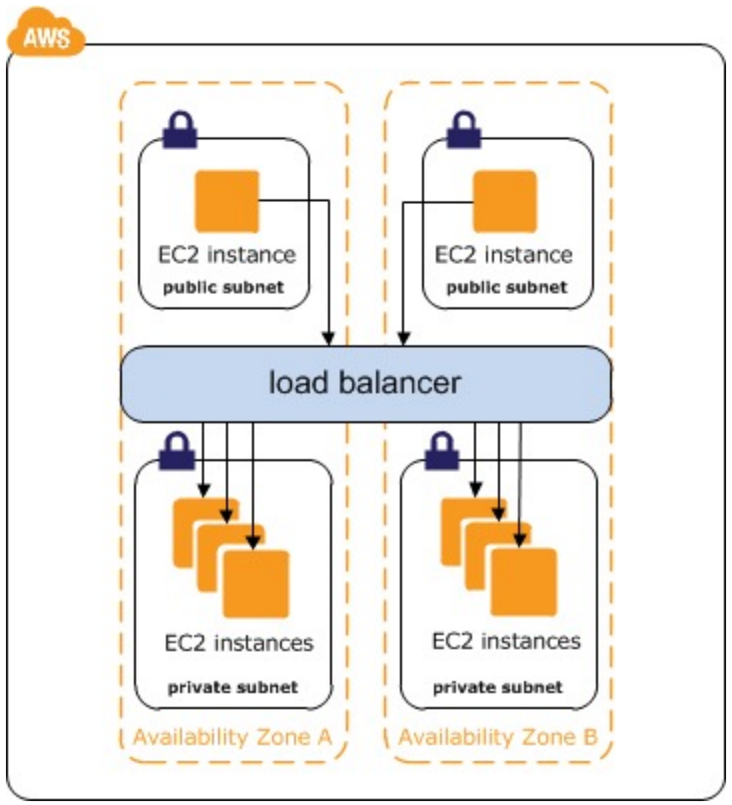
Classic Load Balancer overview

- A **load balancer** distributes incoming application traffic across multiple EC2 instances in multiple Availability Zones.
  - This increases the fault tolerance of your applications.
  - Elastic Load Balancing detects unhealthy instances and routes traffic only to healthy instances.
- **Load balancer** serves as a single point of contact for clients.
  - This increases the availability of your application.
  - You can add and remove instances from your load balancer as your needs change, without disrupting the overall flow of requests to your application.
  - Elastic Load Balancing scales your load balancer as traffic to your application changes over time.
  - Elastic Load Balancing can scale to the vast majority of workloads automatically.
- A **listener** checks for connection requests from clients, using the **protocol** and **port** that you configure, and **forwards requests** to one or more **registered instances** using the protocol and port number that you configure.
  - You add one or more listeners to your load balancer.
- You can configure **health checks**, which are used to monitor the health of the registered instances so that the load balancer **only sends requests to the healthy instances**.



Internal Classic Load Balancers

- When you create a load balancer in a VPC, you must choose whether to make it an **internal load balancer** or an **internet-facing load balancer**.
- The nodes of an **internet-facing load balancer have public IP addresses**.
  - The DNS name of an internet-facing load balancer is publicly resolvable to the public IP addresses of the nodes.
  - Internet-facing load balancers can route requests from clients over the internet.
- The nodes of an **internal load balancer have only private IP addresses**.
  - The DNS name of an internal load balancer is publicly resolvable to the private IP addresses of the nodes.
  - Internal load balancers can only route requests from clients with access to the VPC for the load balancer.



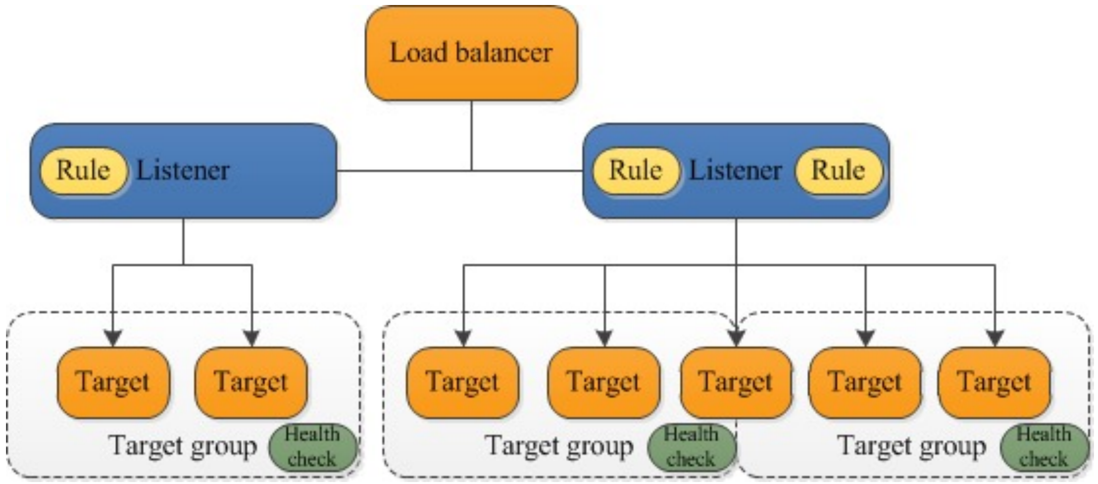
AWS Classic Load Balancer Features

- High Availability
- Health Checks
- SSL offloading
- Sticky Sessions
- Layer 4 or 7 Load Balancing
- Access Logs
- Operational Monitoring

Application Load Balancer

Application Load Balancer

- A **load balancer** serves as the single point of contact for clients.
  - The load balancer distributes incoming application traffic across multiple targets, such as EC2 instances, in multiple Availability Zones. This increases the availability of your application.
  - You add one or more listeners to your load balancer.
- A **listener** checks for connection requests from clients, using the **protocol** and **port** that you configure.
  - The rules that you define for a listener determine how the load balancer routes requests to its registered targets.
  - Each rule consists of a priority, one or more actions, and one or more conditions.
  - When the conditions for a rule are met, then its actions are performed.
- Each **target group** routes requests to one or more registered targets, such as EC2 instances, using the protocol and port number that you specify.
  - You can register a target with multiple target groups.
  - You can configure health checks on a per target group basis. Health checks are performed on all targets registered to a target group that is specified in a listener rule for your load balancer.



AWS Application Load Balancer

An Application Load Balancer is a load balancing option for the Elastic Load Balancing service that operates at the application layer and allows you to define routing rules based on content across multiple services or containers running on one or more Amazon Elastic Compute Cloud (Amazon EC2) instances.

AWS Application Load Balancer Features

- Host-based routing
- Path-based routing
- HTTP/2 Support
- WebSockets Support
- Health Checks
- Layer 7 Load Balancing
- SSL Offloading
- Request Tracking
- Web Application Firewall
- Containerized Application Support

Load Balancer Features

- Asymmetric Load
- Distributed Denial of Service (DDoS) attack protection - load balancers can provide features such as SYN cookies and delayed-binding (the back-end servers don't see the client until it finishes its TCP handshake) to mitigate SYN flood attacks and generally offload work from the servers to a more efficient platform.
- HTTP Compression
- See https://en.wikipedia.org/wiki/Load_balancing_(computing)#Load_balancer_features for more features.

Network Load Balancer

Network Load Balancer

- Network Load Balancer operates at the connection level (Layer 4), routing connections to targets - Amazon EC2 instances, containers and IP addresses based on IP protocol data.
- Ideal for load balancing of TCP traffic, Network Load Balancer is capable of handling millions of requests per second while maintaining ultra-low latencies.
- Network Load Balancer is optimized to handle sudden and volatile traffic patterns while using a single static IP address per Availability Zone.
- It is integrated with other popular AWS services such as Auto Scaling, Amazon EC2 Container Service (ECS), and Amazon CloudFormation.

Network Load Balancer Key Features

- Connection-based Load Balancing - You can load balance TCP traffic, routing connections to targets - Amazon EC2 instances, microservices and containers, and IP addresses.
- Preserve source IP address - Network Load Balancer preserves the client side source IP allowing the back-end to see the IP address of the client.
- Static IP support - Network Load Balancer automatically provides a static IP per Availability Zone (subnet) that can be used by applications as the front-end IP of the load balancer.
- Long-lived TCP Connections - Network Load Balancer supports long-lived TCP connections that are ideal for WebSocket type of applications.
- Central API Support - Network Load Balancer uses the same API as Application Load Balancer. This will enable you to work with target groups, health checks, and load balance across multiple ports on the same Amazon EC2 instance to support containerized applications.

Auto Scaling

What is Autoscaling

- Autoscaling is a method used in cloud computing, whereby the amount of computational resources in a server farm, typically measured in terms of the number of active servers, scales automatically based on the load on the farm.
- It is closely related to, and builds upon, the idea of load balancing.

How does auto-scaling work?

- Auto Scaling **detects** impaired compute instances and unhealthy applications and **replace** the instances without human intervention ensuring that your application is getting the compute capacity it needs.

AWS Autoscaling

Auto Scaling will perform three main functions to automate fleet management for EC2 instances:

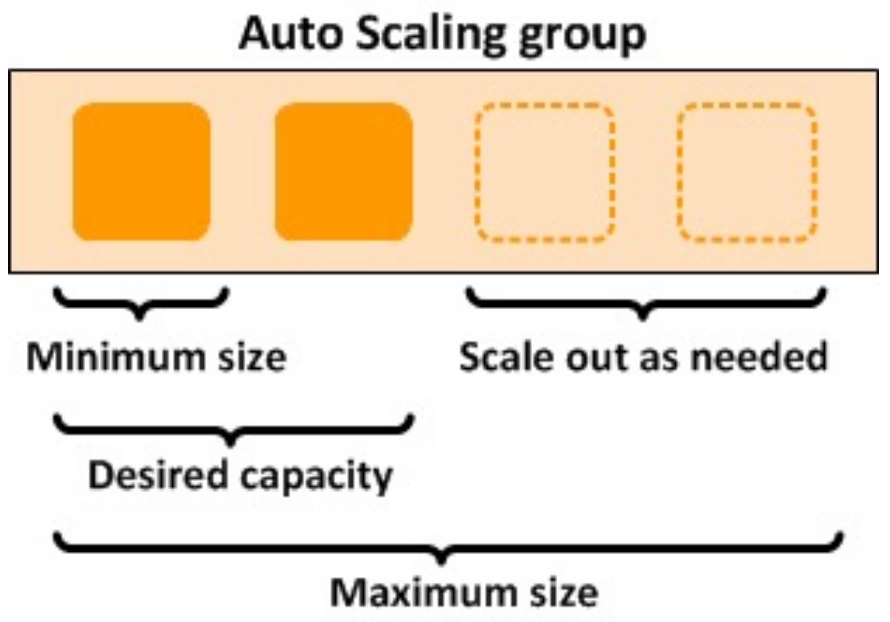
1. **Monitor the health of running instances** - Amazon EC2 Auto Scaling ensures that your application is able to receive traffic and that EC2 instances are working properly. Amazon EC2 Auto Scaling periodically performs health checks to identify any instances that are unhealthy.
2. **Replace impaired instances automatically** - When an impaired instance fails a health check, Amazon EC2 Auto Scaling automatically terminates it and replaces it with a new one. That means that you don't need to respond manually when an instance needs replacing.
3. **Balance capacity across Availability Zones** - Amazon EC2 Auto Scaling can automatically balance instances across zones, and always launches new instances so that they are balanced between zones as evenly as possible across your entire fleet.

Dynamic Scaling

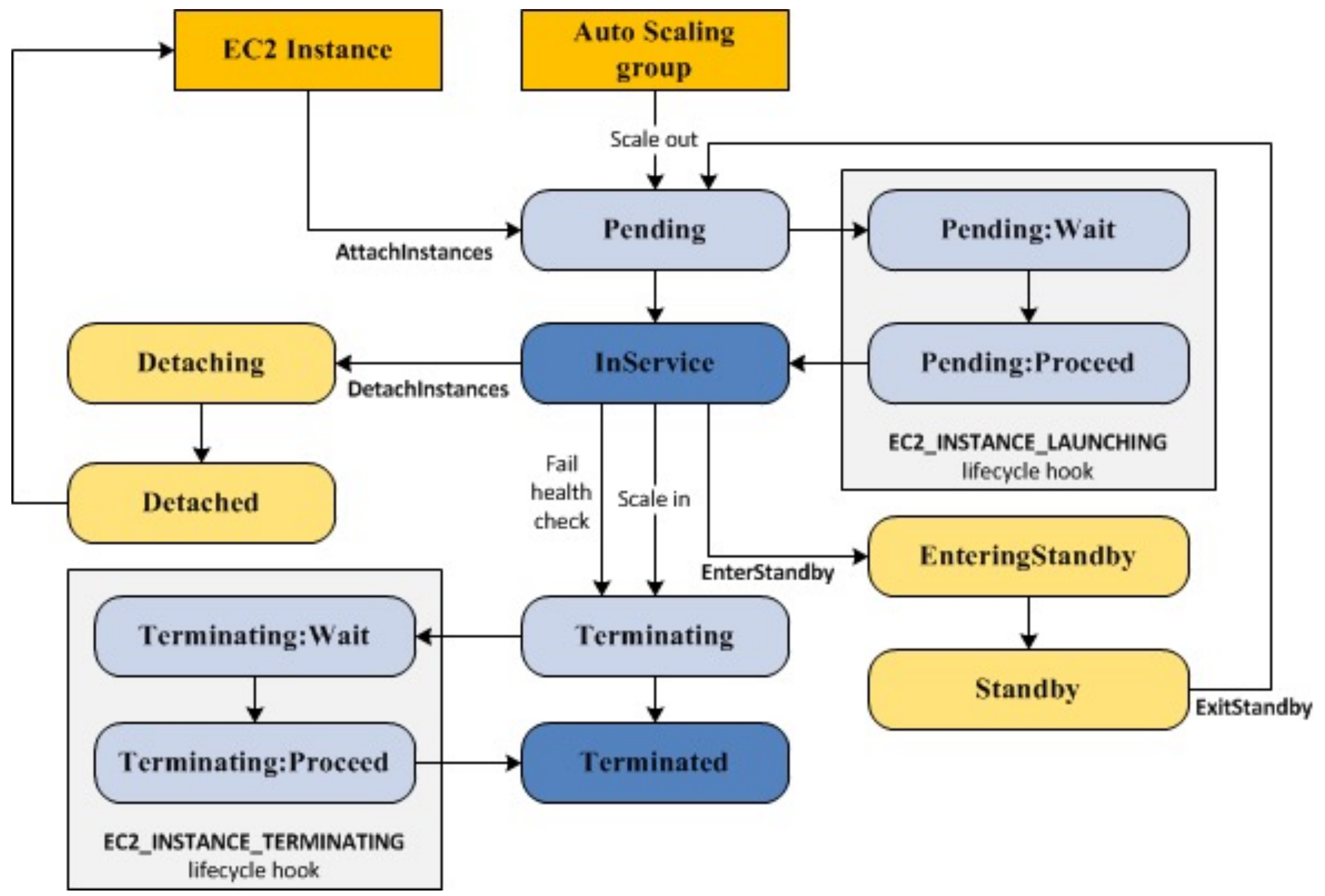
- Amazon EC2 Auto Scaling enables you to follow the demand curve for your applications closely, reducing the need to manually provision Amazon EC2 capacity in advance. For example, you can use target tracking scaling policies to select a load metric for your application, such as CPU utilization. Or, you could set a target value using the new “Request Count Per Target” metric from Application Load Balancer, a load balancing option for the Elastic Load Balancing service. Amazon EC2 Auto Scaling will then automatically adjust the number of EC2 instances as needed to maintain your target.
- You can also use simple scaling policies to set a condition to add new Amazon EC2 instances in increments when the average utilization of your Amazon EC2 fleet is high, and similarly, you can set a condition to remove instances in the same increments when CPU utilization is low. If you have predictable load changes, you can also set a schedule through Amazon EC2 Auto Scaling to plan your scaling activities.
- Amazon EC2 Auto Scaling can also be used with Amazon CloudWatch, which can send alarms to trigger scaling activities, and Elastic Load Balancing to help distribute traffic to your instances within EC2 Auto Scaling groups.
- You can also use Amazon EC2 Auto Scaling in combination with AWS Auto Scaling to scale multiple services.

Auto Scaling Groups

- You create collections of EC2 instances, called Auto Scaling groups.
- You can specify the **minimum** number of instances in each Auto Scaling group, and 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 Auto Scaling ensures that your group never goes above this size.



Auto Scaling Lifecycle



Scaling Policies

- **Manual scaling** - Manual scaling is the most basic way to scale your resources. Specify only the change in the maximum, minimum, or desired capacity of your Auto Scaling group. Amazon EC2 Auto Scaling manages the process of creating or terminating instances to maintain the updated capacity.
- **Scheduled Scaling** - Sometimes you know exactly when you will need to increase or decrease the number of instances in your group, simply because that need arises on a predictable schedule. Scaling by schedule means that scaling actions are performed automatically as a function of time and date.
- **Dynamic Scaling** - A more advanced way to scale your resources, scaling by policy, lets you define parameters that control the scaling process. For example, you can create a policy that calls for enlarging your fleet of EC2 instances whenever the average CPU utilization rate stays above ninety percent for fifteen minutes. This is useful when you can define how you want to scale in response to changing conditions, but you don't know when those conditions will change.

Scale Out

- The following scale out events direct the Auto Scaling group to launch EC2 instances and attach them to the group:
  - You manually increase the size of the group.
  - You create a scaling policy to automatically increase the size of the group based on a specified increase in demand.
  - You set up scaling by schedule to increase the size of the group at a specific time.
- When a scale out event occurs, the Auto Scaling group launches the required number of EC2 instances, using its assigned launch configuration.
- When each instance is fully configured and passes the Amazon EC2 health checks, it is attached to the Auto Scaling group and it enters the InService state. The instance is counted against the desired capacity of the Auto Scaling group.

Scale In

- The following scale in events direct the Auto Scaling group to detach EC2 instances from the group and terminate them:
    - You manually decrease the size of the group.
    - You create a scaling policy to automatically decrease the size of the group based on a specified decrease in demand.
    - You set up scaling by schedule to decrease the size of the group at a specific time.
- When a scale in event occurs, the Auto Scaling group detaches one or more instances. The Auto Scaling group uses its termination policy to determine which instances to terminate. Instances that are in the process of detaching from the Auto Scaling group and shutting down enter the Terminating state, and can't be put back into service. If you add a lifecycle hook to your Auto Scaling group, you can perform a custom action here. Finally, the instances are completely terminated and enter the Terminated state.

Scaling Cooldowns

- The cooldown period is a configurable setting for your Auto Scaling group that helps to ensure that it doesn't launch or terminate additional instances before the previous scaling activity takes effect.
- After the Auto Scaling group dynamically scales using a simple scaling policy, it waits for the cooldown period to complete before resuming scaling activities.
- Cooldown periods are <u>not</u> supported for scheduled scaling.

Additional Resources

See Lecture Page