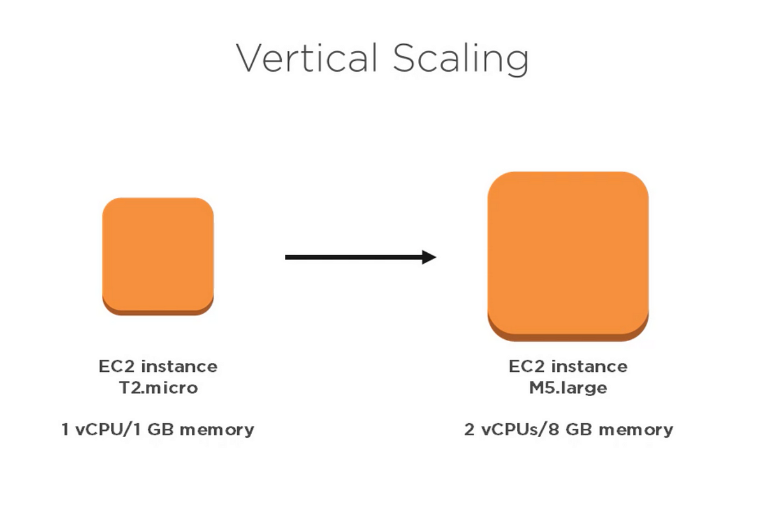
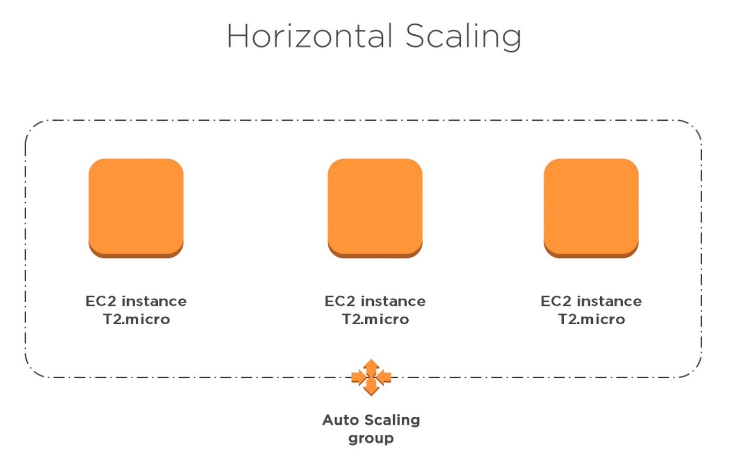
Scalability is the ability of a system to handle increased load. There are two types of

**Scalability**:

* **Vertical scalability**: Increasing the size of a single instance, such as upgrading from a t2.micro to a t2.large EC2 instance.



* **Horizontal scalability**: Increasing the number of instances, such as adding more EC2 instances to an Auto Scaling Group.



**High availability** is the ability of a system to remain available even if one or more components fail. This is achieved by running the system in multiple Availability Zones (AZs) and using load balancers to distribute traffic across the instances.

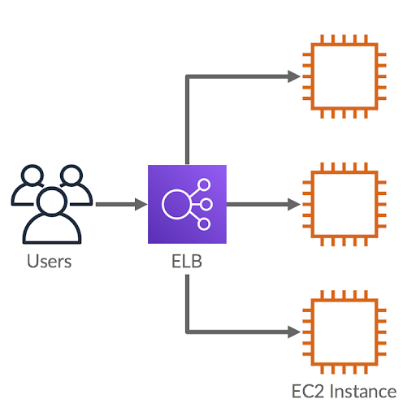
Here is an example of the difference between scalability and high availability:

Imagine that you have a call centre with a single server that can handle 100 calls per hour. This server is vertically scalable, meaning that you can increase its capacity to handle more calls by upgrading the hardware. However, if the server fails, all of the calls will be dropped.

To make the call centre more highly available, you could add a second server in a different AZ. The load balancer would then distribute traffic across the two servers. If one server fails, the other server can continue to handle calls.

**What is load balancing?**

A load balancer is a device that distributes traffic across multiple servers, making your application faster, more reliable, and easier to scale.



**Why use a load balancer?**

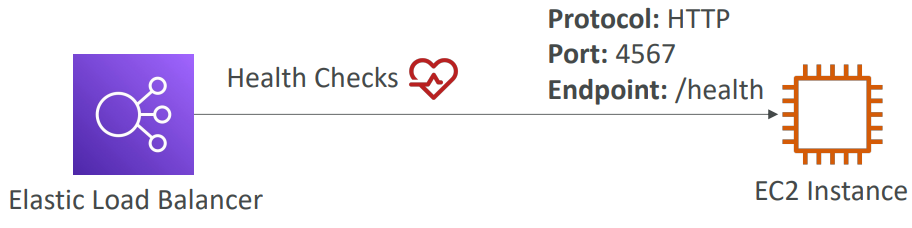
* **Spread Load:** Distribute traffic across multiple servers to prevent overload.
* **Single Access Point:** Provide a single DNS entry for your application.
* **Failover Handling:** Seamlessly manage failures of downstream instances.
* **Health Checks:** Regularly check the health of your instances.
* **SSL Termination:** Handle HTTPS for your websites.
* **Cookie Stickiness:** Enforce session stickiness with cookies.
* **High Availability:** Ensure high availability across zones.
* **Traffic Separation:** Separate public and private traffic.

**Why use an Elastic Load Balancer (ELB)?**

* ELB is a managed load balancer, so you don't have to worry about managing it yourself.
* AWS guarantees that ELB will be working.
* ELB is integrated with many AWS offerings and services.

**Health Checks**

* Test availability and responsiveness of backend servers.
* Used by load balancers to determine which servers can handle traffic.
* Send a request to a specific port and route on the backend server.
* Server is considered healthy if it responds with a 200 (OK) status code.
* Load balancers will only forward traffic to healthy servers.



**Type of load balancer on AWS**

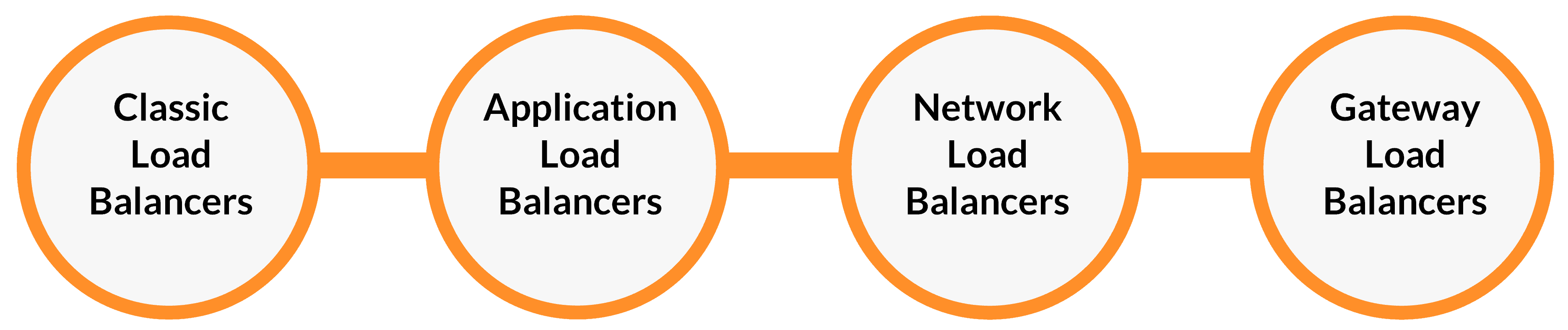
**Classic Load Balancer (CLB):** Older generation load balancer that supports HTTP, HTTPS, TCP, and SSL (secure TCP) traffic.

**Application Load Balancer (ALB):** Newer generation load balancer that supports HTTP, HTTPS, and WebSocket traffic.

**Network Load Balancer (NLB):** Newer generation load balancer that supports TCP, TLS (secure TCP), and UDP traffic.

**Gateway Load Balancer (GWLB):** Newer generation load balancer that operates at layer 3 (Network layer) and supports IP Protocol traffic.

**Note:** Some load balancers can be set up as internal (private) or external (public).



**Classic Load Balancers (v1)**

* Supports Layer 4 (TCP) and Layer 7 (HTTP and HTTPS)
* Health checks are TCP or HTTP-based
* Fixed hostname: XXX.region.elb.amazonaws.com

**Application Load Balancer (v2)**

* Layer 7 (HTTP) load balancer
* Distributes traffic to multiple HTTP applications across machines (target groups) or containers on the same machine
* Supports HTTP/2, WebSocket, and redirects
* Routing tables allow for routing based on path in URL, hostname in URL, query string, and headers
* Well-suited for microservices and container-based applications
* Has a port mapping feature to redirect to a dynamic port in ECS

**Application Load Balancer (v2) Target Groups**

* ALB target groups can be used to route traffic to EC2 instances, ECS tasks, Lambda functions, or IP addresses.
* ALB target groups can be used to distribute traffic across multiple servers.
* Health checks are performed at the target group level.

**Good to Know**

* ALB has a fixed hostname (XXX.region.elb.amazonaws.com).
* Application servers do not see the client's IP address directly.
* The client's true IP address is inserted in the X-Forwarded-For header.
* We can also get the port (X-Forwarded-Port) and protocol (X-Forwarded-Proto).

**Network Load Balancer (NLB)**

* Forwards TCP and UDP traffic to instances
* Handles millions of requests per second
* Low latency (~100ms)
* Has one static IP per AZ and supports Elastic IP assignment
* Used for extreme performance, TCP or UDP traffic
* Not included in AWS free tier

**Target Groups**

* EC2 instances
* IP addresses (must be private IPs)
* Application Load Balancer

Health Checks

* Supports TCP, HTTP, and HTTPS protocols

**Gateway Load Balancer**

* **Gateway Load Balancer (GWL)** is a load balancer that provides transparent load balancing for third-party network virtual appliances. It works at Layer 3 (Network Layer) and uses the GENEVE protocol on port 6081 to encapsulate and forward traffic to the virtual appliances.
* GWL can be used to deploy, scale, and manage a fleet of virtual appliances such as firewalls, intrusion detection and prevention systems, and deep packet inspection systems.
* **GWL Target Groups** can contain EC2 instances or private IP addresses.

**Sticky Sessions (Session Affinity)**

**Sticky sessions (session affinity)** is a feature of load balancers that allows you to keep a user's session on the same server for the duration of the session. This is useful for applications that maintain state information on the server, such as shopping carts or login information.

To implement sticky sessions, you can use a load balancer such as Classic Load Balancer, Application Load Balancer, or Network Load Balancer. The load balancer will use a cookie to track the user's session and ensure that they are always forwarded to the same server.

**Use case:** Make sure the user doesn't lose their session data.

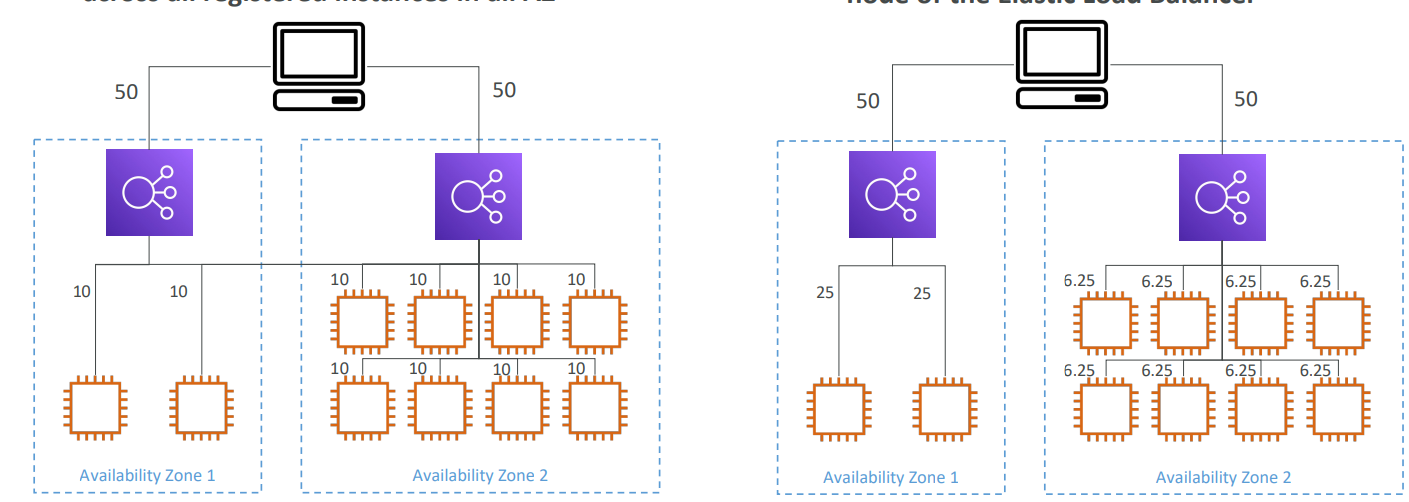
**Potential downside:** Enabling stickiness may bring imbalance to the load over the backend EC2 instances.

There are two types of sticky sessions:

* **Application-based sticky sessions**: Use a custom cookie generated by the target application. The cookie name must be specified individually for each target group.
* **Duration-based sticky sessions**: Use a cookie generated by the load balancer. The cookie name is AWSALB for Application Load Balancing (ALB) and AWSELB for Classic Load Balancing (CLB).
* Do not use the cookie names AWSALB, AWSALBAPP, or AWSALBTG, as these are reserved for use by the load balancer.
* Application-based sticky sessions are not supported with weighted target groups.
* Duration-based sticky sessions are not supported with cross-zone load balancing.

**Cross-Zone Load Balancing**

* Cross-zone load balancing distributes traffic across instances in all Availability Zones (AZs), while non-cross-zone load balancing only distributes traffic across instances in the same AZ.



|  |  |  |
| --- | --- | --- |
| **Load Balancer** | **Cross-zone by default** | **Cost for inter-AZ data** |
| Application Load Balancer | Enabled | No |
| Network Load Balancer | Disabled | Yes |
| Gateway Load Balancer | Disabled | Yes |
| Classic Load Balancer | Disabled | No |

**SSL/TLS Basics**

* SSL refers to Secure Sockets Layer, used to encrypt connections
* TLS refers to Transport Layer Security, which is a newer version
* SSL/TLS certificates encrypt traffic between clients and servers.
* Public SSL/TLS certificates are issued by Certificate Authorities (CAs).
* SSL/TLS certificates expire after a certain period of time and must be renewed.

**Load Balancer SSL Certificates**

* Load balancers use X.509 certificates to terminate SSL/TLS connections.
* You can manage certificates using AWS Certificate Manager (ACM) or upload your own.
* HTTPS listeners require a default certificate and can have an optional list of additional certificates for multiple domains.
* Clients can use Server Name Indication (SNI) to specify the hostname they want to reach.
* You can specify a security policy to support older versions of SSL/TLS.

**SSL Server Name Indication (SNI)**

* Solves the problem of loading multiple SSL certificates onto one web server.
* Requires the client to indicate the hostname of the target server in the initial SSL handshake.
* The server will then find the correct certificate, or return the default one.
* Only works for ALB, NLB, and CloudFront (newer generation). Does not work for CLB (older gen).

**Elastic Load Balancers – SSL Certificates**

Application Load Balancer (ALB) and Network Load Balancer (NLB) support multiple SSL certificates with Server Name Indication (SNI) to host multiple secure applications on a single load balancer listener. Classic Load Balancer (CLB) only supports one SSL certificate, requiring multiple CLBs for multiple hostnames with multiple SSL certificates.

**Connection Draining**

Connection draining is a feature that allows load balancers to gracefully remove instances from service. It does this by completing all in-flight requests to the instance before it is removed from service.

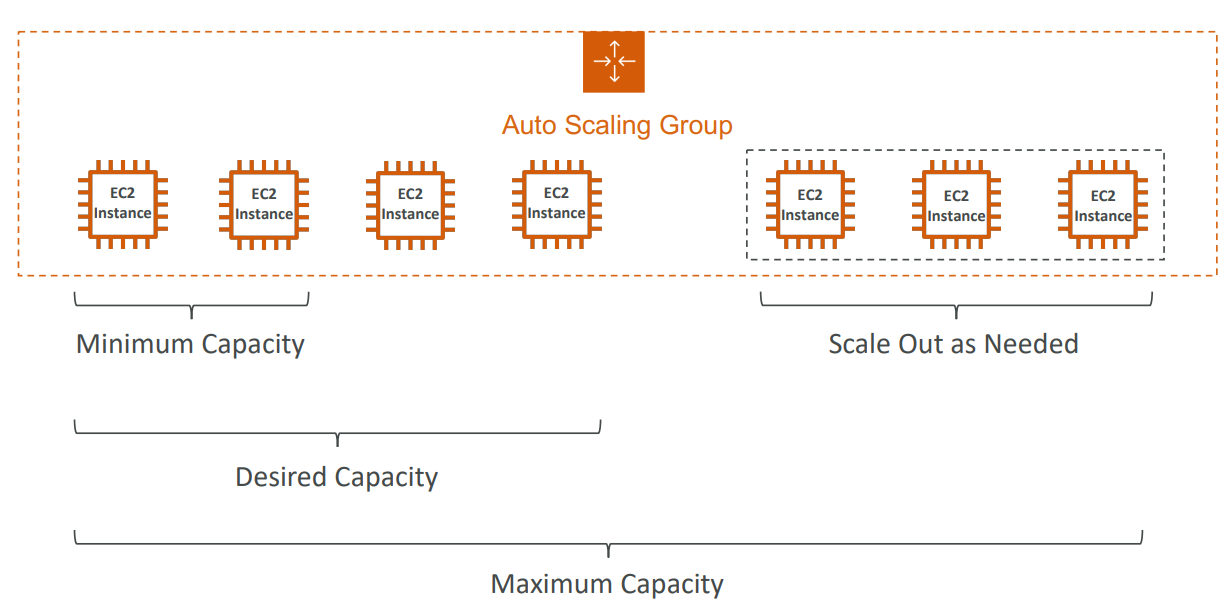
Connection draining can be enabled on Classic Load Balancers (CLBs), Application Load Balancers (ALBs), and Network Load Balancers (NLBs). On ALBs and NLBs, the feature is called deregistration delay.

The connection draining timeout can be set from 1 to 3600 seconds. The default is 300 seconds.

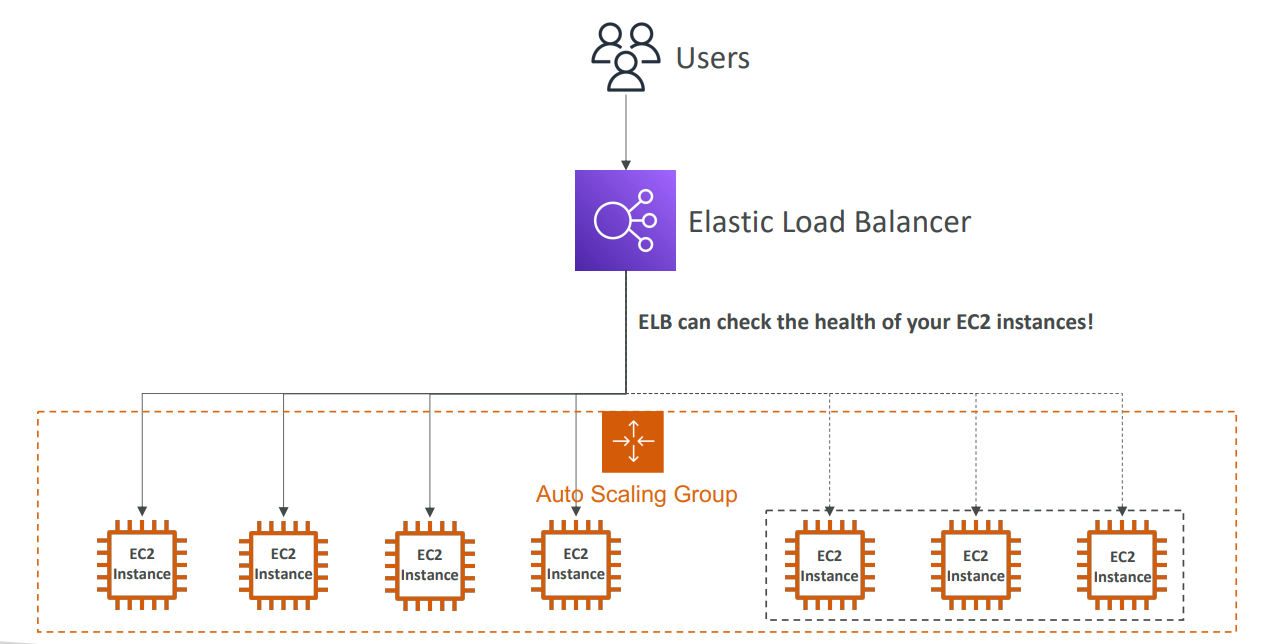
**Auto Scaling Group (ASG)**

* Scales out (add EC2 instances) to match an increased load
* Scales in (remove EC2 instances) to match a decreased load
* Ensures a minimum and a maximum number of EC2 instances running
* Registers new instances to a load balancer
* Recreates an EC2 instance in case a previous one is terminated
* Free (you only pay for the underlying EC2 instances)

**Auto Scaling Group in AWS**

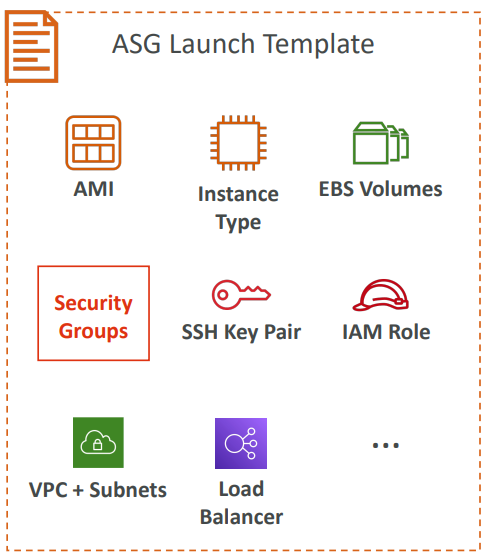


**Auto Scaling Group in AWS with Load Balancer**



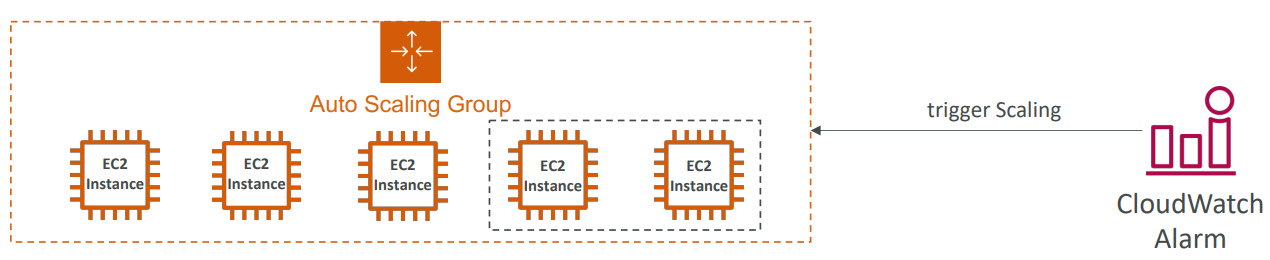
**ASG Attributes**

* Launch Template
* AMI + Instance Type
* EC2 User Data
* EBS Volumes
* Security Groups
* SSH Key Pair
* IAM Roles
* Network + Subnets Information
* Load Balancer Information
* Min Size / Max Size / Initial Capacity
* Scaling Policies



**Auto Scaling - CloudWatch Alarms & Scaling**

* Scale ASG based on CloudWatch alarms
* Monitor metric (Average CPU, custom metric)
* Create scale-out/in policies based on alarm



**Dynamic Scaling Policies**

* **Target Tracking Scaling**: Keep a metric such as CPU utilization at a target value.
* **Simple / Step Scaling**: Scale based on CloudWatch alarm thresholds.
* **Scheduled Actions**: Scale based on known usage patterns.

**Predictive Scaling**

continuously forecast load and schedule scaling ahead

**Good metrics to scale on**

* **CPUUtilization:** How busy your instances are, on average.
* **RequestCountPerTarget:**How many requests each instance is handling, on average.
* **Average Network In/Out:** How much data each instance is sending and receiving, on average.
* **Any custom metric:** Any other metric that you think is important for your application's performance or health.

**Scaling Cooldowns**

**Cooldown period:** After an Auto Scaling Group scales out or in, there is a period of time during which it will not scale again. This is to allow for metrics to stabilize.

**Reduce cooldown period:** Use a ready-to-use AMI to reduce the configuration time for new instances, allowing them to start serving requests faster.

**Auto Scaling - Instance Refresh:**

* Update the launch template and recreate all EC2 instances.
* Use the native Instance Refresh feature.
* Set the minimum healthy percentage.
* Specify the warm-up time (how long until the instance is ready to use).