AWS Auto Scaling & ELB

* Auto Scaling dynamically adds and removes EC2 instances, while Elastic Load Balancing manages incoming requests by optimally routing traffic so that no one instance is overwhelmed
* Auto Scaling helps to automatically increase the number of EC2 instances when the user demand goes up, and decrease the number of EC2 instances when demand goes down
* ELB service helps to distribute the incoming web traffic (called the load) automatically among all the running EC2 instances
* ELB uses load balancers to monitor traffic and handle requests that come through the Internet.
* Using ELB & Auto Scaling
  + makes it easy to route traffic across a dynamically changing fleet of EC2 instances
  + load balancer acts as a single point of contact for all incoming traffic to the instances in an Auto Scaling group.

### **AWS Auto Scaling & ELB**

## **Attaching/Detaching ELB with Auto Scaling Group**

* Auto Scaling integrates with Elastic Load Balancing and enables to attach one or more load balancers to an existing Auto Scaling group.
* ELB registers the EC2 instance using its IP address and routes requests to the primary IP address of the primary interface (eth0) of the instance.
* After the ELB is attached, it automatically registers the instances in the group and distributes incoming traffic across the instances
* When ELB is detached, it enters the Removing state while deregistering the instances in the group.
* If connection draining is enabled, ELB waits for in-flight requests to complete before deregistering the instances.
* Instances remain running after they are deregistered from the ELB
* Auto Scaling adds instances to the ELB as they are launched, but this can be suspended. Instances launched during the suspension period are not added to load balancer, after resumption, and must be registered manually.

## **High Availability & Redundancy**

* Auto Scaling can span across multiple AZs, within the same region
* When one AZ becomes unhealthy or unavailable, Auto Scaling launches new instances in an unaffected AZ
* When the unhealthy AZs recovers, Auto Scaling redistributes the traffic across all the healthy AZs
* Elastic Load balancer can be setup to distribute incoming requests across EC2 instances in a single AZ or multiple AZs within a region
* It is recommended to take advantage of the safety and reliability of geographic redundancy by using Auto Scaling & ELB by **spanning Auto Scaling groups across multiple AZs within a region and then setting up ELB to distribute incoming traffic across those AZs.**
* Incoming traffic is load balanced equally across all the AZs enabled for ELB

## **Health Checks**

* Auto Scaling group determines the health state of each instance by periodically checking the results of EC2 instance status checks
* Auto Scaling marks the instance as unhealthy and replaces the instance, if the instance fails the EC2 instance status check
* ELB also performs health checks on the EC2 instances that are registered with the it for e.g. application is available by pinging and health check page
* Auto Scaling, by default, does not replace the instance, if the ELB health check fails
* **ELB health check with the instances should be used to ensure that traffic is routed only to the healthy instances**
* After a load balancer is registered with an Auto Scaling group, it can be configured to use the results of the ELB health check in addition to the EC2 instance status checks to determine the health of the EC2 instances in the Auto Scaling group.

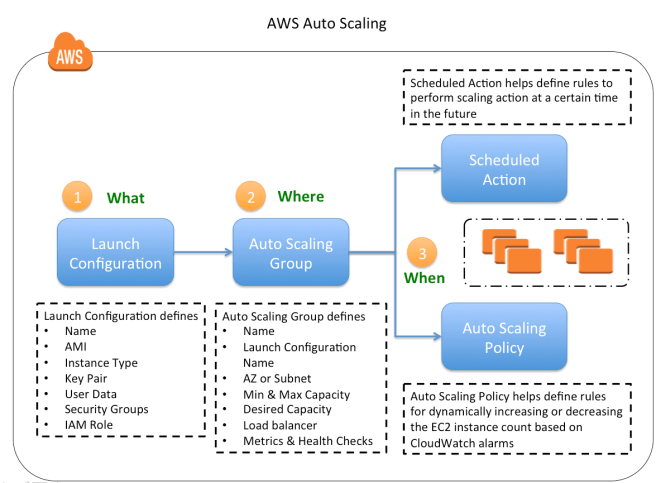
## **Monitoring**

* Elastic Load Balancing sends data about the load balancers and EC2 instances to Amazon CloudWatch. CloudWatch collects data about the performance of your resources and presents it as metrics.
* After registering one or more load balancers with the Auto Scaling group, Auto Scaling group can be configured to use ELB metrics (such as request latency or request count) to scale the application automatically

# Auto Scaling Overview

* Auto Scaling provides the ability to ensure a correct number of EC2 instances are always running to handle the load of the application
* Auto Scaling helps to achieve better fault tolerance, better availability and cost management
* Auto Scaling also helps specify Scaling policies which can the be used to launch and terminate EC2 instances to handle any increase or decrease in demand on the application.
* Auto Scaling attempts to distribute instances evenly between the AZs that are enabled for the Auto Scaling group.
* Auto Scaling does this by attempting to launch new instances in the AZ with the fewest instances. If the attempt fails, Auto Scaling attempts to launch the instances in another Availability Zone until it succeeds

## **Auto Scaling Configuration**



### **Launch Configuration**

* Launch configuration is a template that an Auto Scaling group uses to launch EC2 instances.
* Launch configuration is similar to EC2 configuration and involves selection of the Amazon Machine Image (AMI), the instance type, a key pair, one or more security groups, and a block device mapping.
* Launch configuration can be associated multiple Auto Scaling groups
* Launch configuration can’t be modified after creation and needs to be created new if any modification required
* Basic or Detailed monitoring for the instances in the Auto Scaling group can be enabled when a launch configuration is created.
* By default, basic monitoring is enabled when you create the launch configuration using the AWS Management Console and detailed monitoring is enabled when you create the launch configuration using the AWS CLI or an API

### **Auto Scaling Group**

* Auto Scaling groups is the core of Auto Scaling and contains a collection of EC2 instances that share similar characteristics and are treated as a logical grouping for the purposes of instance scaling and management.
* Auto Scaling group requires
  + **Launch configuration** to determine the EC2 template to use for launching the instance
  + **Minimum & Maximum capacity** to determine the number of instances when an autoscaling policy is applied. The number of instances cannot grow beyond this boundaries
  + **Desired capacity**
    - to determine the number of instances the ASG must maintain at all times. If missing, it equals to the minimum size.
    - Desired capacity is different from minimum capacity.
    - An Auto Scaling group’s desired capacity is the default number of instances that should be running. A group’s minimum capacity is the fewest number of instances the group can have running
  + **Availability Zones or Subnets** in which the instances will be launched.
  + **Metrics & Health Checks** – metrics to determine when it should launch or terminate instances and health checks to determine if the instance is healthy or not
* Auto Scaling group starts by launching a desired capacity of instances and maintains this number by performing periodic health checks.
* If an instance becomes unhealthy, it terminates and launches a new instance
* Auto Scaling group can also use scaling policies to increase or decrease the number of instances automatically to meet changing demands
* An Auto Scaling group can contain EC2 instances in one or more AZs within the same region.
* Auto Scaling groups **cannot** span multiple regions.
* To merge separate single-zone Auto Scaling groups into a single Auto Scaling group spanning multiple AZs, rezone one of the single-zone groups into a multi-zone group, and then delete the other groups. This process works for groups with or without a load balancer, as long as the new multi-zone group is in one of the same AZs as the original single-zone groups.
* Auto Scaling group can be associated with a single launch configuration.
* As the Launch Configuration can’t be modified once created, only way to update the Launch Configuration for an Auto Scaling group is to create a new one and associate it with the Auto Scaling group.
* When the launch configuration for the Auto Scaling group is changed, any new instances launched use the new configuration parameters, but the existing instances are not affected.
* Auto Scaling group can be deleted from CLI, if it has no running instances else need to set the minimum and desired capacity to 0. This is handled automatically when deleting an ASG from AWS management console.

### **Auto Scaling Plans**

#### **Maintain steady count of Instances**

* Auto Scaling ensures a steady minimum (or desired if specified) count of Instances will always be running.
* If an instance is found unhealthy, Auto Scaling will terminate the Instance and launch a new one
* Auto Scaling group determines the health state of each instance by periodically checking the results of EC2 instance status checks
* Auto Scaling group can be associated with a load balancer enabled to use the Elastic Load Balancing health check, Auto Scaling determines the health status of the instances by checking the results of both EC2 instance status and Elastic Load Balancing instance health.
* Auto Scaling marks an instance unhealthy and launches a replacement if
  + the instance is in a state other than *running*,
  + the system status is *impaired*, or
  + Elastic Load Balancing reports the instance state as *OutOfService*.
* After an instance has been marked unhealthy as a result of an EC2 or ELB health check, it is almost immediately scheduled for replacement. It never automatically recovers its health.
* For an unhealthy instance, the instance’s health check can be changed back to healthy manually but you will get an error if the instance is already terminating. Because the interval between marking an instance unhealthy and its actual termination is so small, attempting to set an instance’s health status back to healthy is probably useful only for a suspended group
* When your instance is terminated, any associated Elastic IP addresses are disassociated and are not automatically associated with the new instance.
* Elastic IP addresses must be associated with the new instance manually.
* Similarly, when the instance is terminated, its attached EBS volumes are detached and must be attached to the new instance manually

#### **Manual scaling**

* Manual scaling can be performed by
  + Changing the desired capacity limit of the Auto Scaling group
  + Attaching/Detaching instances to the Auto Scaling group
* Attaching/Detaching of an EC2 instance can be done only if
  + Instance is in the running state.
  + AMI used to launch the instance must still exist.
  + Instance is not a member of another Auto Scaling group.
  + Instance is in the same Availability Zone as the Auto Scaling group.
  + If the Auto Scaling group is associated with a load balancer, the instance and the load balancer must both be in the same VPC.
* Auto Scaling increases the desired capacity of the group by the number of instances being attached. But if the number of instances being attached plus the desired capacity exceeds the maximum size of the group, the request fails.
* When Detaching instances, you have the option of decrementing the desired capacity for the Auto Scaling group by the number of instances being detached. If you choose not to decrement the capacity, Auto Scaling launches new instances to replace the ones that you detached.
* If you detach an instance from an Auto Scaling group that is also registered with a load balancer, the instance is deregistered from the load balancer. If connection draining is enabled for your load balancer, Auto Scaling waits for the in-flight requests to complete.

#### **Scheduled scaling**

* Scaling based on a schedule allows you to scale your application in response to predictable load changes for e.g. last day of the month, last day of an financial year
* Scheduled scaling requires configuration of Scheduled actions, which tells Auto Scaling to perform a scaling action at certain time in future, with the start time at which the scaling action should take effect, and the new minimum, maximum, and desired size the group should have
* Auto Scaling guarantees the order of execution for scheduled actions within the same group, but not for scheduled actions across groups
* Multiple Scheduled Actions can be specified but should have **unique** time value and they **cannot** have overlapping time scheduled which will lead to its rejection

#### **Dynamic scaling**

* Allows you to scale automatically in response to the changing demand for e.g. scale out in case CPU utilization of the instance goes above 70% and scale in when the CPU utilization goes below 30%
* Auto Scaling group uses a combination of **alarms & policies** to determine when the conditions for scaling are met.
  + An alarm is an object that watches over a single metric over a specified time period. When the value of the metric breaches the defined threshold, for the number of specified time periods the alarm performs one or more actions (such as sending messages to Auto Scaling).
  + A policy is a set of instructions that tells Auto Scaling how to respond to alarm messages.
* Dynamic scaling process works as below
  + Amazon CloudWatch monitors the specified metrics for all the instances in the Auto Scaling group
  + Changes are reflected in the metrics as the demand grows or shrinks
  + When the change in the metrics breaches the threshold of the CloudWatch alarm, the CloudWatch alarm performs an action. Depending on the breach, the action is a message sent to either the scale-in policy or the scale-out policy
  + After the Auto Scaling policy receives the message, Auto Scaling performs the scaling activity for the Auto Scaling group.
  + This process continues until you delete either the scaling policies or the Auto Scaling group.

### **Multiple Policies**

* An Auto Scaling group can have more than one scaling policy attached to it any given time.
* Each Auto Scaling group would have at least two policies: one to scale the architecture out and another to scale the architecture in.
* If an Auto Scaling group has multiple policies, there is always a chance that both policies can instruct the Auto Scaling to Scale Out or Scale In at the same time.
* When this situations occur, Auto Scaling chooses the policy that has the **greatest impact** on the Auto Scaling group for e.g. if two policies are triggered at the same time and Policy 1 instructs to scale out the instance by 1 while Policy 2 instructs to scale out the instances by 2, Auto Scaling will use the Policy 2 and scale out the instances by 2 as it has a greater impact

### **Auto Scaling Cooldown**

* Auto Scaling cooldown period is a configurable setting for your Auto Scaling group that helps to ensure that Auto Scaling doesn’t launch or terminate additional instances before the previous scaling activity takes effect and allows the newly launched instances to start handling traffic and reduce load.
* When Auto Scaling group dynamically scales using a simple scaling policy and launches an instance, Auto Scaling suspends the scaling activities for the cooldown period (default 300 seconds) to complete before resuming scaling activities.
* Example Use Case
  + You configure an scale out alarm to increase the capacity if the CPU utilization increases more then 80%
  + An CPU spikes occurs and cause the alarm to be triggered, Auto Scaling launches a new instance
  + However, it would take time for the newly launched instance to be configured, instantiated and started, lets say 5 mins
  + Without a cooldown period, if an other CPU spikes occurs Auto Scaling would launch a new instance again and this would continue for 5 mins till the previously launched instance is up and running and started handling traffic
  + With a cooldown period, Auto Scaling would suspend the activity for the specified time period enabling the newly launched instance to start handling traffic and reduce load
  + After the cooldown period, Auto scaling resumes to act on the alarms
* When manually scaling the Auto Scaling group, the default is not to wait for the cooldown period, but you can override the default and honor the cooldown period.
* Note that if an instance becomes unhealthy, Auto Scaling does not wait for the cooldown period to complete before replacing the unhealthy instance.
* Cooldown periods are automatically applied to dynamic scaling activities for simple scaling policies and is **not** supported for step scaling policies.

## **Termination Policy**

Termination policy helps the Auto Scaling to decide which instances it should terminate first when Auto Scaling automatically scales in.  Auto Scaling specifies a default termination policy and also allows you to create a customized one

### **Default Termination Policy**

Default termination policy is designed to help ensure that the network architecture spans Availability Zones evenly and instances are selected for termination as follows :-

1. Selection of Availability Zone
   * selects the AZ, in multiple AZs environment, with the **most instances**and at least one instance that is not protected from scale in.
   * selects the AZ with instances that use the **oldest launch configuration**, if there are more than one AZ with same number of instances
2. Selection of an Instance in the Availability Zone
   * terminates the **unprotected instance using the oldest launch configuration**, if one exists.
   * terminates unprotected instances **closest to the next billing hour,** If multiple instances with oldest launch configuration. This helps in maximizing the use of the EC2 instances while minimizing the number of hours billed for EC2 usage.
   * terminates instances at **random**, if more than one unprotected instance closest to the next billing hour

### **Customized Termination Policy**

1. Auto Scaling first assesses the AZs for any imbalance. If an AZ has more instances than the other AZs that are used by the group, then it applies the specified termination policy on the instances from the imbalanced AZ
2. If the Availability Zones used by the group are balanced, then Auto Scaling applies the termination policy that you specified.
3. Following Customized Termination policies are supported
   1. **OldestInstance –**terminates the oldest instance in the group and can be useful to upgrade to new instance types
   2. **NewestInstance –**terminates the newest instance in the group and can be useful when testing a new launch configuration
   3. **OldestLaunchConfiguration –**terminates instances that have the oldest launch configuration
   4. **ClosestToNextInstanceHour –** terminates instances that are closest to the next billing hour and helps to maximize the use of your instances and manage costs.
   5. **Default –**terminates as per the default termination policy

### **Instance Protection**

* Instance protection controls whether Auto Scaling can terminate a particular instance or not.
* Instance protection can be enabled on a Auto Scaling group or an individual instance as well, at any time
* Instances launched within an Auto Scaling group with Instance protection enabled would inherit the property.
* Instance protection starts as soon as the instance is InService and if the Instance is detached, it loses its Instance protection
* If all instances in an ASG are protected from termination during scale in and a scale-in event occurs, it can’t terminate any instance and will decrement the desired capacity.
* Instance protection does not protect for the below cases
  + Manual termination through the EC2 console, the terminate-instances command, or the TerminateInstances API.
  + Termination if it fails health checks and must be replaced
  + Spot instances in an Auto Scaling group from interruption

## **Standby State**

Auto Scaling allows you to put the InService instance in the Standby state during which the instance is still a part of the ASG but does not serve any requests. This can be used to either troubleshoot an instance or update an instance and return the instance back to service

* An instance can be put into Standby state and it will continue to remain in the Standby state unless exited
* Auto Scaling, by default, decrements the desired capacity for the group and prevents it from launching a new instance. If no decrement is selected, it would launch a new instance
* When the instance is in the standby state, instance can be updated or used for troubleshooting
* If a load balancer is associated with Auto Scaling, the instance is automatically deregistered when the instance is in Standby state and registered again when the instance exits the Standby state

## **Suspension**

* Auto Scaling processes can be suspended and then resumed. This can be very useful to investigate a configuration problem or debug an issue with the application, without triggering the Auto Scaling process.
* Auto Scaling also performs **Administrative Suspension**where it would suspend processes for ASGs, if the Auto Scaling groups that have been trying to launch instances for over 24 hours but have not succeeded in launching any instances.
* Auto Scaling processes include
  + **Launch** – Adds a new EC2 instance to the group, increasing its capacity.
  + **Terminate** – Removes an EC2 instance from the group, decreasing its capacity.
  + **HealthCheck** -Checks the health of the instances.
  + **ReplaceUnhealthy** – Terminates instances that are marked as unhealthy and subsequently creates new instances to replace them.
  + **AlarmNotification** – Accepts notifications from CloudWatch alarms that are associated with the group. If suspended,  Auto Scaling does not automatically execute policies that would be triggered by an alarm
  + **ScheduledActions** – Performs scheduled actions that you create.
  + **AddToLoadBalancer** – Adds instances to the load balancer when they are launched.
  + **AZRebalance** – Balances the number of EC2 instances in the group across the Availability Zones in the region.
    - If an AZ either is removed from the ASG or becomes unhealthy or unavailable, Auto Scaling launches new instances in an unaffected AZ before terminating the unhealthy or unavailable instances
    - When the unhealthy AZ returns to a healthy state, Auto Scaling automatically redistributes the instances evenly across the Availability Zones for the group.
    - Note that if you suspend AZRebalance and a scale out or scale in event occurs, Auto Scaling still tries to balance the Availability Zones for e.g. during scale out, it launches the instance in the Availability Zone with the fewest instances.
    - If you suspend Launch, AZRebalance neither launches new instances nor terminates existing instances. This is because AZRebalance terminates instances only after launching the replacement instances.
    - If you suspend Terminate, the ASG can grow up to 10% larger than its maximum size, because Auto Scaling allows this temporarily during rebalancing activities. If it cannot terminate instances, your ASG could remain above its maximum size until the Terminate process is resumed

**Exam Question Scenario :-**  
EC2 instances fail to launch with Autoscaling configuration

**Description :-**  
Autoscaling configuration requires the following :-

Autoscaling launch configuration which allows you to select an

* + AMI
  + Instance type
  + IAM role (optional)
  + Security group
  + Key pair file

Autoscaling group configuration allows you to select AZ to be used to launch the EC2 instances with the selected launch configuration

**Troubleshooting key points :-**

1. AMI id does not exist or is still pending and cannot be used to launch instances
2. Security group provided in the launch configuration does not exist
3. Key pair associated with the EC2 instance does not exist
4. Autoscaling group not found or is incorrectly configured
5. AZ configured with the Autoscaling group is no longer supported cause it might not be available
6. Invalid EBS block device mappings
7. Instance type is not supported in the AZ
8. Capacity limits reached either cause of the [restriction](https://aws.amazon.com/ec2/faqs/) on the number of instance type that can be launched in a region or cause AWS is not able to provision the specified instance type in the AZ (for e.g. no more spot instances or On-demand instances availability)