## *Auto Scaling Group (ASG)*

An **Auto Scaling Group (ASG)** in AWS is a service that helps maintain the availability of your applications by automatically adjusting the number of Amazon EC2 instances in your group according to the conditions you define. This ensures that you have the right number of instances available to handle the load for your application.

**Why Use Auto Scaling Groups (ASGs)?**

1. **Availability**: ASGs ensure that your application has enough running instances to handle incoming traffic, increasing the overall availability and fault tolerance of your application.
2. **Cost Efficiency**: By automatically scaling out (adding instances) and scaling in (removing instances) based on demand, ASGs help optimize cost by using just enough resources to handle the current load.
3. **Load Balancing Integration**: ASGs can be integrated with Elastic Load Balancing (ELB) to distribute traffic evenly among instances.

AWS Auto Scaling groups (ASGs) are specifically designed to manage the scaling of Amazon EC2 instances. They automatically adjust the number of EC2 instances in your group based on demand, using metrics like CPU utilization or custom metrics.

## *Auto Scaling Group (ASG)*

Here are the key terms and definitions you need to understand when working with AWS Auto Scaling Groups:

1. **Auto Scaling Group (ASG):**
   - An **Auto Scaling Group** is a collection of Amazon EC2 instances managed as a single unit for automatic scaling and management. It is the core entity that handles launching, terminating, and maintaining a specified number of EC2 instances according to scaling policies.
2. **Launch Configuration:**
   - A **Launch Configuration** is an older mechanism that specifies the configuration for the instances within an ASG, such as the Amazon Machine Image (AMI), instance type, key pair, security groups, and other instance-specific details. Launch Configurations are immutable; once created, they cannot be modified.
3. **Launch Template:**
   - A **Launch Template** is a newer, more flexible way to define instance configurations. It includes parameters such as AMI ID, instance type, key pair, security groups, and network settings. Unlike Launch Configurations, Launch Templates can be versioned, allowing for more complex and customizable deployment setups.
4. **Desired Capacity:**
   - The **Desired Capacity** is the number of EC2 instances that you want the ASG to maintain. This is the target number of instances running at any given time, based on your application's needs.
5. **Minimum Size:**
   - The **Minimum Size** is the smallest number of EC2 instances that an ASG should have at all times. This setting ensures that there are always enough instances to handle a minimal level of traffic.
6. **Maximum Size:**
   - The **Maximum Size** is the largest number of EC2 instances that an ASG can scale up to. This limit is used to prevent the ASG from launching too many instances, which can help control costs and prevent over-provisioning.

7. **Scaling Policies:**

**Scaling Policies** define the rules for when and how the ASG should scale the number of EC2 instances. There are several types of scaling policies:

- **Dynamic Scaling:** Adjusts the number of instances in response to changing demand, based on CloudWatch metrics (like CPU utilization or network traffic).
- **Target Tracking Scaling:** Adjusts the capacity to maintain a specific metric at a target value, such as keeping average CPU utilization at 50%.
- **Step Scaling:** Increases or decreases the number of instances in response to specific metric thresholds. This allows for more precise scaling by defining steps and scaling actions based on metric values.
- **Scheduled Scaling:** Changes the number of instances based on a schedule, such as increasing capacity during expected traffic spikes at certain times of the day.

8. **Health Check Grace Period:**
   - The **Health Check Grace Period** is the amount of time that an instance is allowed to boot and start running its application before the ASG starts checking its health status. This helps avoid terminating instances that are still initializing.

9. **Instance Termination Policy:**
   - The **Instance Termination Policy** specifies how the ASG should decide which instances to terminate first when scaling in (reducing the number of instances). You can choose policies such as:

- **Default:** Terminates the oldest instance in the group that is not protected from scale-in.
- **Newest:** Terminates the newest instance first.
- **Oldest Launch Configuration:** Terminates instances with the oldest launch configuration first.

10. **Cool Down Period:**
    - The **Cool Down Period** is a time delay after a scaling activity during which no further scaling actions are taken, allowing instances time to stabilize. This helps prevent rapid scaling up and down in response to temporary spikes in demand.

11. **Scaling Activities:**
    - **Scaling Activities** are the actual operations performed by an ASG, such as launching or terminating instances. You can view the history of scaling activities to understand why and when the ASG scaled.

12. **Lifecycle Hooks:**
    - **Lifecycle Hooks** allow you to perform custom actions when instances are launched or terminated by an ASG. For example, you can use lifecycle hooks to complete software installations or run scripts before the instance is fully registered with the ASG or before it is terminated.
13. **Elastic Load Balancer (ELB) Integration:**
    - **ELB Integration** involves attaching an Elastic Load Balancer (Application Load Balancer, Network Load Balancer, or Classic Load Balancer) to the ASG. This ensures that incoming traffic is distributed evenly among the healthy instances in the group.
14. **Instance Warm-Up:**
    - **Instance Warm-Up** is a feature that allows new instances to warm up and be ready to handle traffic before they are included in scaling activities. This period ensures that instances are fully initialized and applications are running before they start serving requests.

## *Auto Scaling Group (ASG)*

**Scaling Policies**

- **Scheduled Scaling**
  - Scale based on a schedule
  - Used when the load pattern is predictable
- **Simple Scaling**
  - Scale to certain size on a CloudWatch alarm
  - Ex. when CPU > 90%, then scale to 10 instances
- **Step Scaling**
  - Scale incrementally in steps using CloudWatch alarms
  - Ex. when CPU > 70%, then add 2 units and when CPU < 30%, then remove 1 unit
  - Specify the **instance warmup time** to scale faster
- **Target Tracking Scaling**
  - ASG maintains a CloudWatch metric and scale accordingly (automatically creates CW alarms)
  - Ex. maintain CPU usage at 40%
- **Predictive Scaling**
  - Historical data is used to predict the load pattern using ML and scale automatically

## *Auto Scaling Group (ASG)*

**Launch Configuration & Launch Template**

- Defines the following info for ASG
    - AMI (Instance Type)
    - EC2 User Data
    - EBS Volumes
    - Security Groups
    - SSH Key Pair
    - Min / Max / Desired Capacity
    - Subnets (where the instances will be created)
    - Load Balancer (specify which ELB to attach instances)
    - Scaling Policy
- **Launch Configuration** (legacy)
    - Cannot be updated (must be re-created)
    - **Does not support Spot Instances**
- **Launch Template** (newer)
    - Versioned
    - Can be updated
    - **Supports both On-Demand and Spot Instances**
    - Recommended by AWS

## *Auto Scaling Group (ASG)*

### Scaling Cooldown

- After a scaling activity happens, the ASG goes into cooldown period (**default 300 seconds**) during which it does not launch or terminate additional instances (ignores scaling requests) to allow the metrics to stabilize.
- Use ready-to-use AMI to launch instances faster to be able to reduce the cooldown period

### Health Checks

- By default, ASG uses the EC2 status check (not the ELB health check). This could explain why some instances that are labelled as unhealthy by an ELB are still not terminated by the ASG.
- To prevent ASG from replacing unhealthy instances, suspend the **ReplaceUnhealthy** process type.

### Termination Policy

- Select the AZ with the most number of instances
- Delete the instance with the oldest launch configuration
- Delete the instance which is closest to the next billing hour

### Rebalancing AZs

ASG ensures that the group never goes below the minimum scale. Actions such as changing the AZ for the group or explicitly terminating or detaching instances can lead to the ASG becoming unbalanced between AZs. In such cases, ASG compensates by **rebalancing** the AZs by **launching new instances before terminating the old ones**, so that rebalancing does not compromise the performance or availability of the application.

## Lifecycle Hooks

- Used to perform extra steps before launching or terminating an instance. Example:
  - Install some extra software or do some checks (during pending state) before declaring the instance as "in service"
  - Before the instance is terminated (terminating state), extract the log files
- **Without lifecycle hooks, pending and terminating states are avoided**

## Instance Refresh

- StartInstanceRefresh API call to update the instances to the latest launch template (doesn’t happen automatically if we just update the launch template)
- Can specify **minimum healthy percentage** (for rolling update)
- Can specify **warm-up time** (how long to wait for the new instances to become ready to use)

## *Auto Scaling Group (ASG)*

Working Process

**Define the Launch Configuration/Template**:

- You start by defining a launch configuration or a launch template that specifies the settings for the Amazon EC2 instances. This includes the instance type, Amazon Machine Image (AMI), security groups, and other configurations.

**Create the Auto Scaling Group**:

- An ASG is created with a defined minimum, maximum, and desired number of instances. For example, you might set a minimum of 2 instances, a maximum of 10, and a desired capacity of 5 instances.

**Monitor and Scale Based on Policies**:

- ASG monitors your application and automatically adjusts the number of running instances based on scaling policies and metrics (like CPU utilization or network traffic). If demand increases, ASG can add more instances (scale-out); if demand decreases, it can terminate instances (scale-in).

**Health Checks and Replacement**:

- ASG regularly performs health checks on the instances. If an instance fails a health check, ASG automatically terminates the unhealthy instance and launches a new one to replace it, ensuring that the desired capacity is maintained.

**Distribute Instances Across Availability Zones**:

- ASG can distribute instances across multiple Availability Zones to improve the fault tolerance and availability of your application. If one zone fails, instances in other zones can handle the load.

**Scale Based on Scheduled Actions**:

- You can also set up scheduled actions to scale the number of instances at specific times. For example, you might scale up during peak hours and scale down during off-peak hours.

**Terminate Instances Gracefully**:

- When scaling in (reducing instances), ASG ensures that instances are terminated gracefully, respecting the termination policies you've set (like keeping instances in specific Availability Zones).

**Maintain Desired State**:

- ASG continuously ensures that the number of instances running matches the desired capacity, automatically adding or removing instances as needed to maintain the application’s performance and reliability.

## *Auto Scaling Group (ASG)*

Scenario based Questions

https://lisireddy.medium.com/aws-asg-scenario-based-questions-32b793526581

**Wish you the best …! Happy Learning ..!**

**Yours' Love (@lisireddy across all the platforms)**