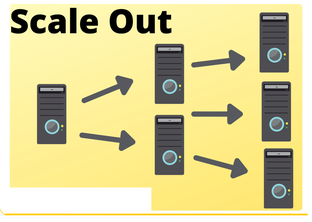
Auto Scaling

1 Intro

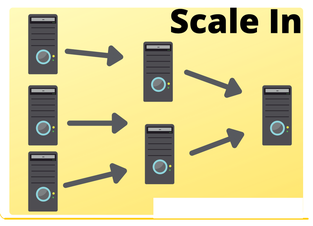
1. Amazon EC2 Auto Scaling helps you ensure that you have the correct number of Amazon EC2 instances available to handle the load for your application.
2. You can specify the minimum number of instances and Amazon EC2 Auto Scaling ensures that your group never goes below this size.
3. You can specify the maximum number of instances and Amazon EC2 Auto Scaling ensures that your group never goes above this size.
4. If you specify the desired capacity Amazon EC2 Auto Scaling ensures that your group has this many instances.

2 Benefits

1. Better fault tolerance
   1. Auto Scaling can detect when an instance is unhealthy, terminate it, and launch an instance to replace it.
   2. You can also configure Auto Scaling to use multiple Availability Zones. If one Availability Zone becomes unavailable, Auto Scaling can launch instances in another one to compensate.
2. Better availability — Auto Scaling can help you ensure that your application always has the right amount of capacity to handle the current traffic demands.
3. Scalability — Scaling Terminology
   1. Vertical Scaling
   2. Horizontal Scaling
      1. Scale Out — Add more nodes to a system, such as adding a new computer to a distributed software application.



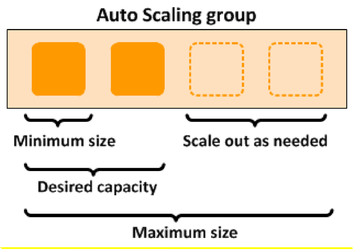
      2. Scale In — Remove nodes from a system



   3. Auto Scaling Provides Horizontal Scaling

3 Auto Scaling components

1. EC2 Autoscaling Groups
   1. Your EC2 instances are organized into groups so that they can be treated as a logical unit for the purposes of scaling and management. In Auto scaling Group You Can Specify



2. Launch Templates
   1. Launch configurations OR Launch templates
      1. A launch configuration is an instance configuration template that an Auto Scaling group uses to launch EC2 instances. When you create a launch configuration, you specify information for the instances. Include the ID of the Amazon Machine Image (AMI), the instance type, a key pair, one or more security groups, and a block device mapping.
   2. Launch Templates Advantage Over Launch Configuration
      1. allows you to have multiple versions of a template With versioning,
      2. launch templates to ensure that you're accessing the latest features and improvements Like
         1. current generation of EBS volume types (gp3 and io2)
         2. EBS volume tagging
      3. you cannot create an Auto Scaling group that launches both Spot and On-Demand Instances or that specifies multiple instance types or multiple launch templates. You must use a launch template to configure these features.

4 Scaling options

1. Manual Scaling
   1. Manual scaling is the most basic way to scale your resources.
   2. You Need to specify the change in the maximum, minimum, or desired capacity of your Auto Scaling group.
   3. Use Case
      1. To increase resources for an infrequent event, such as the release of a new game version that will be available for download and require a user registration.
2. Scheduled Scaling
   1. Means that scaling actions are performed automatically as a function of time and date
   2. This is useful when you know exactly when to increase or decrease the number of instances in your group,
   3. Use Case
      1. periodic events such as end-of-month, end-of-quarter, or end-of-year processing, and also other predictable, recurring events.
3. Dynamic Scaling
   1. Dynamic scaling lets you define parameters that control the Auto Scaling process in a scaling policy.
   2. 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.
   3. Use Case — It is used to Handle Uncertain Load

5 Scaling Policy

1. SIMPLE SCALING
   1. SINGLE THRESHOLD
   2. THRESHOLD - add 1 instance If CPU Utilization is between 50% and 60%
2. STEP SCALING
   1. MULTIPLE thresholds
   2. Threshold A - add 1 instance when CPU Utilization is between 50% and 60%
      Threshold B - add 2 instances when CPU Utilization is between 60% and 70%
      Threshold C - add 3 instances when CPU Utilization is between 70% and 90%
3. TARGET TRACKING SCALING
   1. It's automatic
   2. Set the value and that's Done
   3. Self Optimization

6 Predictive Scaling

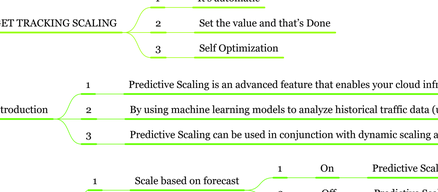
1. Introduction
   1. Predictive Scaling is an advanced feature that enables your cloud infrastructure, specifically EC2 instances, to automatically scale in or out based on predicted traffic
   2. By using machine learning models to analyze historical traffic data (usually three weeks of data is a minimum requirement), Predictive Scaling accurately forecasts future demand and proactively manages the compute capacity of your Auto Scaling group.
   3. Predictive Scaling can be used in conjunction with dynamic scaling and scheduled scaling. While dynamic scaling responds to changes in real-time, predictive scaling prepares for expected changes, offering a more proactive approach.
2. Configuration
   1. Scale based on forecast
      1. On — Predictive Scaling will automatically adjust the number of instances in your Auto Scaling group based on its predictions of future traffic.
      2. Off — Predictive Scaling will still make predictions about future traffic, but it will not take any automatic actions to scale your resources.
   2. Metrics and target utilization
      1. Metrics
         1. The chosen metric is what the system will analyze historically to predict future needs.
         2. These could be CPU utilization, network I/O, request count per target (for load balancers), or any other relevant metric that reflects the load or performance of your instances.
         3. Example — If you choose CPU utilization, the algorithm will look at past CPU usage trends to forecast future CPU requirements.
      2. Target utilization
         1. Target Utilization refers to the desired level of resource usage that you want to maintain.
         2. It's a threshold that guides the scaling process.
         3. Example — If the metric is CPU utilization, setting a target utilization of 50% means you're indicating that the ideal operational state is when each server is using around 50% of its available CPU power.
   3. Pre-launch instances
      1. This setting configures the amount of time by which the instance launch time can be advanced
      2. Example — The forecast says to add capacity at 10:00 AM, and you choose to pre-launch instances by 5 minutes. In that case, the instances will be launched at 9:55 AM.
   4. Buffer maximum capacity above the forecasted capacity
      1. This is an extra percentage or number of instances that you choose to add on top of the forecasted capacity.
      2. Example
         1. Suppose the Predictive Scaling algorithm forecasts that you will need 10 instances to meet the demand at a certain time.
         2. If you set a buffer maximum capacity of 20%, it will actually provision 12 instances (10 + 20% of 10), giving you 2 extra instances as a buffer.

7 Instance Maintenance Policy

1. The Instance Maintenance Policy in AWS Auto Scaling allows you to define how your Auto Scaling group replaces instances during various operations like
   1. Updating instances to newer versions of the AMI.
   2. Replacing unhealthy instances identified by health checks.
   3. Maintaining the desired capacity across Availability Zones.
2. Terminate and launch — Unhealthy instances are terminated immediately, followed by new launches
3. Launch before terminating — New instances are launched first before unhealthy ones are terminated, ensuring minimal downtime
4. Custom behavior — Fine-tune scaling behavior for unique needs with advanced control over replacements and health checks.

8 Termination policy

1. Introduction
   1. The termination policy of an Auto Scaling group determines which instances the Auto Scaling group will terminate first when it needs to scale In.
   2. AWS provides several built-in termination policies, and you can also set up custom termination policies to suit your specific needs.
2. Default Termination Policy



   1. Step 1 — Balance Across Availability Zones
      1. The default termination policy aims to maintain an even distribution of instances across Availability Zones. In our scenario, Ap-South-1A has more instances than the others. So, the policy targets this AZ for scale-in to balance the distribution.
      2. Target for Consideration: Instances A1, A2, A3 in Ap-South-1A
   2. Step 2 — Instance Protection Status
      1. All three instances in Ap-South-1a are checked for scale-in protection
      2. If one of them is marked as protected from scale-in actions, it's excluded from termination.
      3. Assuming none of these instances have scale-in protection enabled:
      4. Still in Consideration — Instances A1 and A2 and A3
   3. Step 3 — Oldest Launch Configuration or Template
      1. Suppose Instance A1 and A2 were launched using the same launch configuration, and Instance A3 was launched with a newer version because you updated the launch template after A1 and A2 were already running.
      2. Target for Further Consideration — Instances A1 and A2, because they have the oldest launch configuration.
   4. Step 4 — Closest to the Next Billing Hour
      1. The default termination policy considers which instance is closest to the next billing hour to optimize cost savings.
      2. If Instance A1 is 50 minutes into its current hour, and Instance A2 is only 10 minutes in
      3. Terminating A1 would result in a smaller portion of unused time that you've already paid for.
      4. Instance Selected for Termination — Instance A1
3. Other Built-in Termination Policies
   1. Allocation strategies — Terminate instances in the Auto Scaling group to align the remaining instances to the allocation strategy for the type of instance that is terminating (either a Spot Instance or an On-Demand Instance).
   2. OldestLaunchTemplate Or OldestLaunchConfiguration
      1. Terminate instances that have the oldest launch template or OldestLaunchConfiguration
      2. This policy is useful when you're updating a group and phasing out the instances from a previous configuration.
   3. ClosestToNextInstanceHour
      1. Terminate instances that are closest to the next billing hour.
      2. This policy helps you maximize the use of your instances that have an hourly charge.
      3. Only instances that use Amazon Linux, Windows, or Ubuntu are billed in one-second increments.
   4. NewestInstance
      1. Terminate the newest instance in the group.
      2. This policy is useful when you're testing a new launch configuration but don't want to keep it in production.
   5. OldestInstance
      1. Terminate the oldest instance in the group.
      2. This option is useful when you're upgrading the instances in the Auto Scaling group to a new EC2 instance type.
      3. You can gradually replace instances of the old type with instances of the new type.
4. Custom Termination Policies
   1. A custom termination policy provides better control over which instances are terminated, and when.
   2. A Lambda function can be used in conjunction with custom termination policies to provide advanced, programmable logic for determining which instances to terminate during a scale-in event.
   3. 5 Solid Reasons to Use a Custom Termination Policy
      1. Better Control Over Which Instances to Terminate
         1. Standard Rules Might Not Fit All Needs — Sometimes, the usual ways AWS decides which EC2 Instance to shut down don't match exactly what you need.
         2. Custom Checks for Better Choices
            1. Picking specific servers based on special labels (tags) you've given them,
            2. Looking at very specific metric of how well your application is running on each instances to decide which ones to shut down.
      2. Graceful Shutdown of Applications
         1. Regular rules for terminating EC2 instances might stop them suddenly while they're still working.
         2. Your Lambda function can initiate a graceful shutdown sequence within the instance before termination
         3. This prevents data loss, service disruptions, and improves overall application resilience.
      3. Pre-Termination Actions and Data Backups
         1. You might need to perform specific actions before terminating an instance, such as
            1. Backing up critical data to other instances or external storage.
            2. Draining connections or gracefully terminating running processes
            3. Unregistering the instance from any service discovery mechanism.
         2. Your Lambda function can execute these pre-termination steps, ensuring data safety and a smooth transition even when scaling down.
      4. Integration with External Systems or Databases
         1. Certain termination decisions might require access to information beyond what Auto Scaling exposes directly.
         2. Your Lambda function can interact with external systems or databases (e.g., for license management, custom health checks) to make informed termination decisions based on broader context.
      5. Flexible and Dynamic Termination Logic
         1. As your app gets more complex or your needs change, you might need flexible rules for turning off EC2 instances.
         2. By customizing the logic within your Lambda function, you can easily update the termination criteria as your needs change, without code modifications to the Auto Scaling group itself.

9

11