Scaling Policies in AWS Auto Scaling Groups (ASG)

Scaling policies in AWS Auto Scaling Groups (ASGs) allow automatic adjustments to the number of EC2 instances based on application needs and resource usage. These policies help ensure optimal performance, availability, and cost-efficiency by dynamically adding or removing instances to match demand.

There are three main types of scaling policies: **Target Tracking**, **Step Scaling**, and **Simple Scaling**. Additionally, **Scheduled Scaling** and **Predictive Scaling** are used for specific use cases like time-based scaling or anticipating future demand.

1. Types of Scaling Policies

1.1 Target Tracking Scaling Policy

A **Target Tracking Scaling Policy** automatically adjusts the number of instances to keep a specific CloudWatch metric at a predefined target value, similar to how a thermostat maintains a temperature.

Key Features:

- **Automatic Adjustment**: Automatically increases or decreases the number of EC2 instances to maintain the desired value for a selected metric (e.g., CPU utilization, request count per target).
- **Simplified Management**: Users don't need to define complex scaling rules. AWS manages the scaling to maintain the specified target metric.
- **Cooldown Period**: This policy has a built-in cooldown period, ensuring that the Auto Scaling Group doesn't scale too frequently and destabilize.

Example:

- **Metric**: CPU Utilization
- **Target**: 50%
    - If the average CPU utilization across instances exceeds 50%, the policy will add more instances.
    - If CPU utilization falls below 50%, the policy will reduce the number of instances.

Steps to Configure Target Tracking:

1. **Metric Selection**: Choose a CloudWatch metric (e.g., CPU utilization or network traffic).
2. **Set Target Value**: Define the value you want to maintain for the metric (e.g., keep CPU utilization at 50%).

3. **Instance Adjustment**: AWS will automatically adjust the number of instances in the ASG to achieve the target.

Best Use Case:

When you need to maintain a stable level of performance based on a specific metric without manually managing thresholds or steps.

1.2 Step Scaling Policy

A **Step Scaling Policy** adjusts the number of instances based on the magnitude of changes in a CloudWatch metric. This policy gives more granular control by allowing different scaling actions based on varying levels of metric thresholds.

Key Features:

- **Multiple Steps**: Allows different actions based on how much the metric deviates from the threshold.
- **Granular Control**: You can specify different responses based on how severe the metric deviation is. For example, add 1 instance if CPU usage is above 60%, add 2 instances if it's above 80%.
- **Proportional Scaling**: Adjust the scaling action in proportion to the level of demand.

Example:

Metric: CPU Utilization

- If CPU utilization is between 60% and 70%, add 1 instance.
- If CPU utilization is between 70% and 80%, add 2 instances.
- If CPU utilization exceeds 80%, add 3 instances.

Steps to Configure Step Scaling:

1. **Define Alarms**: Set CloudWatch alarms for different thresholds of the chosen metric (e.g., CPU utilization > 60%, > 80%, etc.).
2. **Define Scaling Actions**: For each threshold, define the scaling action (e.g., add or remove a specific number of instances).
3. **Cooldown Period**: Set cooldown periods between scaling actions to prevent rapid oscillation in scaling activities.

Best Use Case:

Applications with unpredictable load variations where scaling needs to respond differently based on the degree of demand.

1.3 Simple Scaling Policy

A **Simple Scaling Policy** scales the number of instances based on a single CloudWatch alarm and a single scaling action. It is less flexible compared to Step Scaling and Target Tracking but is useful for simpler use cases.

Key Features:

- **Single Action**: One scaling action for one alarm condition. For example, when CPU usage exceeds 70%, add 1 instance.
- **Cooldown Period**: A mandatory cooldown period after each scaling activity, during which no new scaling actions can occur.
- **Simplicity**: Easy to configure but lacks the fine-tuned control offered by step scaling.

Example:

Metric: CPU Utilization

- If CPU utilization exceeds 70%, add 1 instance.
- If CPU utilization falls below 30%, remove 1 instance.

Steps to Configure Simple Scaling:

1. **Create Alarm**: Set a CloudWatch alarm to trigger the scaling action (e.g., CPU utilization > 70%).
2. **Define Scaling Action**: Choose the action to take when the alarm is triggered (e.g., add or remove a specific number of instances).
3. **Set Cooldown**: Define the cooldown period to ensure the system stabilizes before triggering another action.

Best Use Case:

Basic applications or systems with straightforward, predictable scaling needs that don't require complex configurations.

2. Other Types of Scaling Policies

2.1 Scheduled Scaling

Scheduled scaling allows you to scale your ASG based on predefined time intervals. This is useful if your application experiences predictable traffic patterns (e.g., high traffic during business hours and low traffic during off-hours).

Key Features:

- **Time-Based Scaling**: Actions are triggered based on specific times and dates.
- **Preemptive Scaling**: You can scale up before anticipated traffic spikes and scale down during quieter periods.
- **Predictable Workloads**: Ideal for workloads that have known, recurring traffic patterns.

Example:

- Scale up to 10 instances at 9:00 AM (when traffic typically increases).
- Scale down to 2 instances at 6:00 PM (after peak hours).

Steps to Configure Scheduled Scaling:

1. **Set Time/Date**: Specify the start time and end time for the scaling action.
2. **Define Scaling Action**: Choose how many instances to scale up or down.
3. **Recurring Actions**: Optionally, set the scaling action to recur at regular intervals (e.g., daily, weekly).

Best Use Case:

Applications with predictable traffic patterns, such as business applications or batch processing jobs.

2.2 Predictive Scaling

Predictive scaling uses machine learning to predict future traffic based on historical data. It allows the ASG to scale ahead of demand, reducing response times to traffic spikes.

Key Features:

- **Machine Learning-Based Predictions**: AWS forecasts the future load based on historical patterns and automatically adjusts capacity.
- **Proactive Scaling**: Helps ensure that your application is scaled ahead of demand, reducing latency and improving performance.
- **Combination with Dynamic Scaling**: You can combine predictive scaling with other scaling policies to handle unexpected changes in demand.

Example:

Based on historical data, AWS predicts a traffic spike at 3:00 PM and automatically scales up the ASG to handle the increased load before the spike occurs.

Steps to Configure Predictive Scaling:

1. **Choose Metric**: Select a CloudWatch metric for which AWS will forecast future traffic patterns.
2. **Set Target**: Define how much ahead of the forecast AWS should scale (e.g., scale exactly to forecast or with a buffer).
3. **Combine with Other Policies**: Optionally, combine predictive scaling with dynamic scaling to handle unpredictable spikes.

Best Use Case:

- Applications with consistent traffic patterns, such as e-commerce sites during holiday sales or online services with daily peaks in user traffic.

3. Key Elements to Consider in Scaling Policies

3.1 Cooldown Periods

A **cooldown period** ensures that the system has time to stabilize after scaling activities. During this period, no additional scaling actions will occur. Cooldown periods help avoid scaling too frequently and can be adjusted based on your application's behavior.

3.2 CloudWatch Metrics

- **Predefined Metrics**: You can use AWS predefined metrics like CPU utilization, network traffic, and request count to trigger scaling actions.
- **Custom Metrics**: In addition to predefined metrics, you can create custom CloudWatch metrics specific to your application, such as request latency, queue length, or memory usage.

3.3 Health Checks

Auto Scaling Groups use **health checks** to determine the health of instances. Instances that fail health checks are automatically replaced. This ensures that only healthy instances are scaled.

4. Best Practices for Using Scaling Policies

1. **Use a Combination of Policies**: You can combine different types of scaling policies (e.g., predictive and dynamic scaling) to achieve the best balance between cost-efficiency and performance.
2. **Monitor and Adjust Scaling Policies**: Regularly monitor the performance of your ASG using CloudWatch and adjust scaling policies based on actual traffic patterns and application behavior.
3. **Set Appropriate Cooldown Periods**: Customize cooldown periods to prevent rapid scaling actions, which can result in increased costs and instability.
4. **Leverage Custom Metrics**: In addition to using basic metrics like CPU utilization, consider using application-specific or custom CloudWatch metrics to fine-tune your scaling policies.
5. **Test Your Scaling Policies**: Simulate traffic spikes and monitor how the ASG responds to scaling policies. This will help you identify any misconfigurations and fine-tune the scaling actions for optimal performance.