AWS EC2 Auto Scaling – Full In-Depth Notes

🔹 What is EC2 Auto Scaling?

EC2 Auto Scaling automatically launches or terminates Amazon EC2 instances in response to changing demand. It helps ensure:

- High availability
- Fault tolerance
- Cost efficiency by using the right number of instances

Part of the **Amazon EC2 Auto Scaling service** (distinct from the broader AWS Auto Scaling service which includes ECS, DynamoDB, etc.).

🔹 Key Concepts

Concept	**Description**
Auto Scaling Group (ASG)	A logical group of EC2 instances managed together for scaling and health checks.
Launch Template or **Launch Configuration**	Specifies instance type, AMI, security groups, key pair, etc.
Scaling Policy	Defines how scaling actions (scale-in or scale-out) are triggered.
Health Checks	Monitors instance health and replaces failed instances.
Lifecycle Hooks	Add custom actions at launch or termination (e.g., run scripts).

🔹 Auto Scaling Group (ASG)

Key Settings:

- **Min Size** – Minimum number of instances at any time
- **Max Size** – Upper limit of instances allowed
- **Desired Capacity** – Ideal number of instances (adjusted by scaling policies)
- **Availability Zones** – For distributing instances
- **Health Checks** – EC2 or ELB health integration

ASG Placement:

- Across **multiple Availability Zones** (highly recommended)
- Optionally associated with **Elastic Load Balancer (ALB/NLB/CLB)**

🔹 Launch Templates vs Launch Configurations

Feature	Launch Template (Recommended)	Launch Configuration
Reusable across services	✅ Yes	🚫 No
Multiple versions	✅ Yes	🚫 No
Spot + On-Demand Mix	✅ Yes	🚫 No
T2/T3 Unlimited Mode	✅ Yes	🚫 No

🔹 Scaling Types

1. Dynamic Scaling (Policy-based)

- Responds to CloudWatch alarms
- Types:
  - **Target Tracking Scaling**: E.g., keep CPU utilization at 50%
  - **Step Scaling**: Add/remove X instances based on thresholds
  - **Simple Scaling**: Basic scale out/in on alarm

2. Scheduled Scaling

- Scale at specific times (e.g., scale up at 8 AM, scale down at 6 PM)

3. Predictive Scaling

- Uses **machine learning** to predict future load and scale in advance (for consistent patterns)

🔹 Scaling Policies (Dynamic)

Type	Use Case
Target Tracking	Maintain a metric like CPU at a target level
Step Scaling	Gradual response to metric thresholds
Simple Scaling	Trigger a scaling action on a single alarm
Scheduled Scaling	Use for known time-based load patterns
Predictive Scaling	Based on machine learning forecasts

🔹 Health Checks in Auto Scaling

- **EC2 Status Checks**: Default health check (system + instance)
- **ELB Health Check**: Checks if the instance is healthy from the load balancer's view
- **Custom Health Check**: Use AWS CLI/API to manually mark instances as unhealthy

Unhealthy instances are **automatically terminated and replaced**.

🔹 Lifecycle Hooks

Lifecycle hooks allow you to **pause Auto Scaling operations** for custom workflows:

Hook Type	Purpose
Instance Launching	Run initialization (e.g., config scripts) before instance becomes InService
Instance Terminating	Perform cleanup before instance termination

Can be combined with Lambda, SNS, SQS, or custom scripts.

🔹 Instance Refresh

Feature: Gradually updates instances in an Auto Scaling group.

Use case: Apply **new AMI** or config changes without downtime.

Steps:

1. Enable **instance refresh** on ASG.
2. ASG replaces old instances with new ones in batches.
3. Health checks ensure success before continuing.

🔹 Mixed Instances Policy

Allows mixing:

- **Instance types** (e.g., m5.large, m5a.large)
- **Purchase options** (On-Demand + Spot)

Benefits:

- Cost optimization
- Instance diversification
- Higher availability with fallback

🔹 Monitoring & Metrics

Monitored via **Amazon CloudWatch**.

Common metrics:

- GroupDesiredCapacity
- GroupInServiceInstances
- GroupMinSize, GroupMaxSize
- GroupTotalInstances
- GroupTerminatingInstances
- GroupPendingInstances
- GroupStandbyInstances

Enable **detailed monitoring** for 1-minute granularity.

🔹 Notifications (Optional)

ASG can send notifications via **Amazon SNS** for:

- Instance launch
- Instance terminate
- Launch failure
- Terminate failure

Use this for **alerts or automation workflows**.

🔹 Pricing

- **Auto Scaling itself is free.**
- You pay for:
  - EC2 instances
  - CloudWatch alarms
  - Optional services like SNS, Lambda

🔹 Security

- Use **IAM roles** for EC2 instances via Launch Templates.
- Secure ASG access with **Instance Profile**.
- Set **Auto Scaling Group termination policies** (e.g., oldest launch template, AZ rebalancing).

🔹 Termination Policies

When scaling in, AWS chooses which instance to terminate using **termination policies**, such as:

1. **OldestInstance**
2. **NewestInstance**
3. **ClosestToNextInstanceHour**
4. **OldestLaunchConfiguration**
5. **AZRebalance** (even distribution)

🔹 Use Cases

Scenario	Benefit from EC2 Auto Scaling
E-commerce app with fluctuating load	Scale out during traffic spikes
Batch processing jobs	Use spot instances with scaling
SaaS product across regions	High availability with AZ spreading
Web app needing zero downtime updates	Use **instance refresh** with rolling deployments
Cost optimization	Combine spot + on-demand in ASG

🔹 Best Practices

- Always use **Launch Templates**, not Launch Configurations.
- Use **Target Tracking Policies** for simplicity.
- Combine with **ALB/NLB** for load balancing and health checks.
- Enable **detailed monitoring** for real-time metrics.
- Set **cooldown periods** to prevent thrashing.
- Tag resources for visibility and cost tracking.
- Use **lifecycle hooks** for app setup/teardown automation.