9. Monitoring, Auto Scaling & Observability

This section describes how the application tier is monitored and automatically scaled based on real-time metrics.
The design leverages native AWS services to ensure **consistent instance configuration, predictable performance under load, and cost efficiency during low-traffic periods**.

9.1 EC2 Launch Template Integration

All EC2 instances in the Auto Scaling Group are launched using a Launch Template, which defines:

- AMI and instance type
- Application bootstrap via `user_data`
- Security groups for controlled access
- IAM instance profile for CloudWatch and SSM access

Design Benefits

- Ensures consistent configuration across all instances
- Automatically enables metric publishing to CloudWatch
- New instances are fully configured and production-ready at launch
- Eliminates configuration drift within the application tier

9.2 Auto Scaling Group Metrics (Enabled)

The Auto Scaling Group explicitly enables CloudWatch metrics with **1-minute granularity** to provide near-real-time visibility into scaling behavior.

```
enabled_metrics = [
```

```
  "GroupMinSize",
  "GroupMaxSize",
  "GroupDesiredCapacity",
  "GroupInServiceInstances",
  "GroupTotalInstances"
]
```

Operational Visibility Provided

- Current and desired instance capacity
- Number of healthy, in-service instances
- Validation that scale-out and scale-in actions are executed successfully

These metrics are critical for troubleshooting scaling events and performing capacity planning.

9.3 CPU-Based Scaling Policy (Primary Signal)

CPU utilization is used as the primary scaling signal.
For a Java + Apache workload, CPU usage directly correlates with request processing and traffic volume, making it a reliable indicator for scaling decisions.

Scale-Out Policy (High Load)

Trigger Conditions

- Average CPU utilization ≥ 70%
- Sustained for 2 evaluation periods
- Metric sampling interval: 120 seconds

```
threshold           = 70
comparison_operator = "GreaterThanOrEqualToThreshold"
```

Scaling Action

- Increase ASG capacity by 1 instance

```
scaling_adjustment = 1
```

Cooldown

- 300 seconds
- Prevents rapid, repetitive scaling actions

Scale-In Policy (Low Load)

Trigger Conditions

- Average CPU utilization ≤ 20%
- Sustained for 2 evaluation periods

```
threshold               = 20
comparison_operator     = "LessThanOrEqualToThreshold"
```

Scaling Action

- Decrease ASG capacity by 1 instance

```
scaling_adjustment = -1
```

Cooldown

- 300 seconds
- Ensures system stability before further scale-down

9.4 End-to-End Scaling Flow

High-Traffic Scenario

1. Increased traffic is received by the Application Load Balancer
2. Requests are distributed across existing EC2 instances
3. CPU utilization exceeds the 70% threshold
4. CloudWatch alarm triggers the scale-out policy
5. The Auto Scaling Group launches a new EC2 instance
6. The new instance:
    - Bootstraps via user-data
    - Registers automatically with the ALB target group
7. Load is redistributed, reducing CPU pressure on existing instances

---

Low-Traffic Scenario

1. Traffic volume decreases over time
2. CPU utilization falls below the 20% threshold
3. CloudWatch alarm triggers the scale-in policy
4. The Auto Scaling Group gracefully terminates one EC2 instance
5. Remaining instances continue serving traffic efficiently

Outcome

- Performance is maintained
- Infrastructure costs are reduced automatically

9.5 ALB and Application Health Monitoring

- The Application Load Balancer continuously performs health checks on EC2 instances via the target group

- Unhealthy instances are:
    - Removed from traffic rotation
    - Replaced automatically by the Auto Scaling Group

Resulting Behavior

- Zero downtime during instance-level failures
- Automatic self-healing without operator intervention

9.6 Operational Benefits

This monitoring and scaling setup provides:

- No requirement for manual scaling
- Predictable application performance under load
- Automatic cost optimization during low usage
- Rapid recovery from instance-level failures

The design reflects a **production-grade monitoring and auto-scaling model** while remaining simple, observable, and maintainable.