🚀 AWS Auto Scaling - Complete Guide

From Single Instance to Highly Available Scalable Infrastructure

📊 The Problem: Single Instance Limitation

❌ Before Auto Scaling

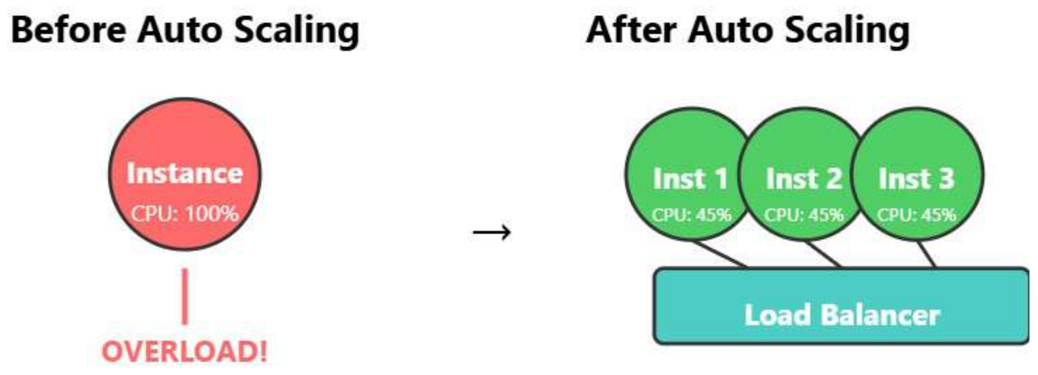
Current Setup:

- 1 EC2 Instance running
- Users increase → CPU 100%
- Application slows down
- Users get timeouts
- Manual scaling needed
- No redundancy

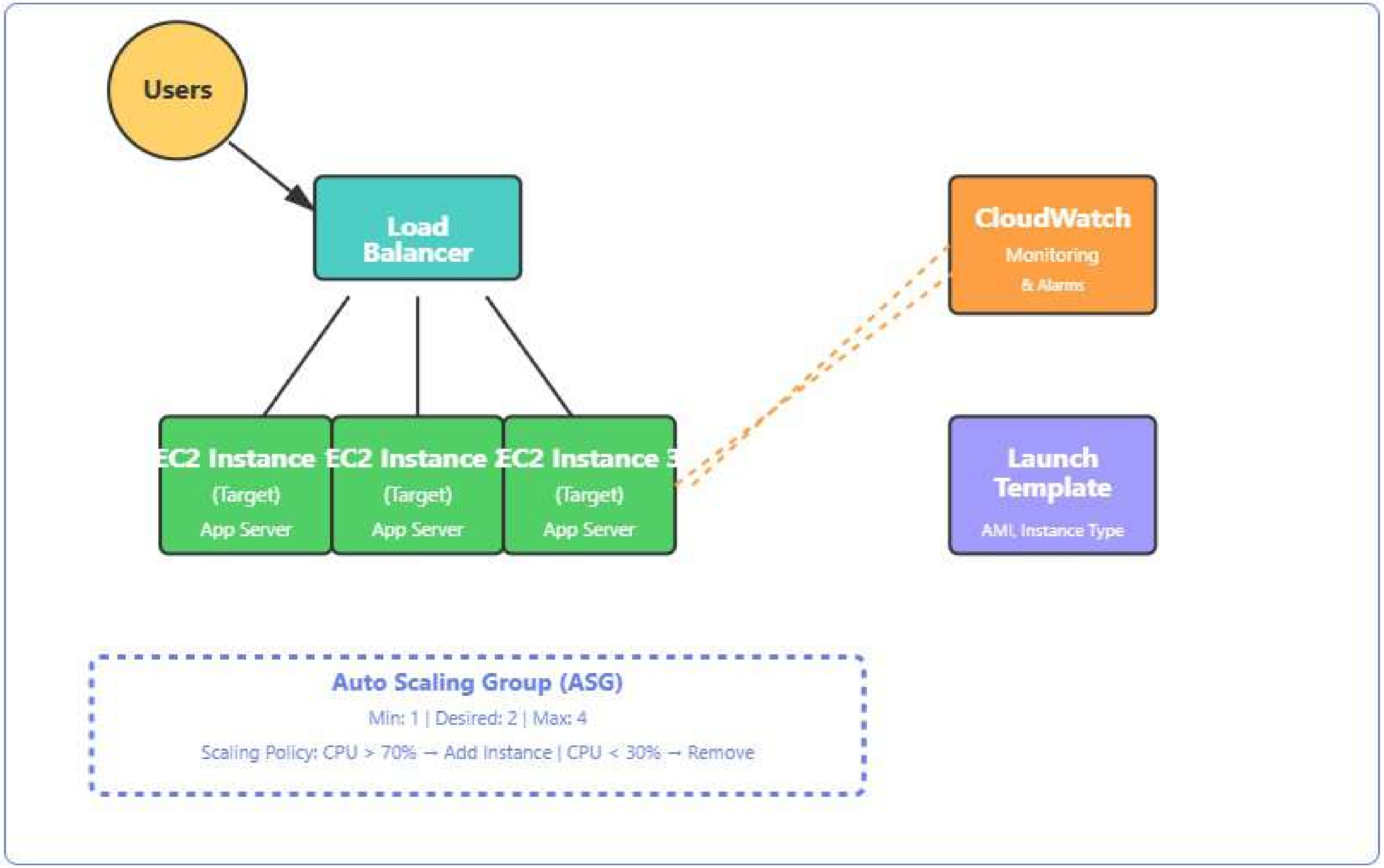
✅ After Auto Scaling

Desired Setup:

- Multiple EC2 Instances
- Auto scales based on load
- Always optimal performance
- No user interruption
- Automatic scaling policy
- High availability



Auto Scaling Architecture



🔧 Complete Step-by-Step Setup Guide

Step 1: Create a Launch Template

1 **Navigate to Launch Templates in EC2 Console**

```
AWS Console → EC2 → Launch Templates → Create Launch Template
```

Configuration:

- Name: `my-app-template`
- AMI: Select your base image (Ubuntu 22.04 LTS recommended)
- Instance Type: `t3.micro` (or t3.small for production)
- Key Pair: Select your existing key pair
- Security Group: Create/select one allowing HTTP (80), HTTPS (443), SSH (22)

Step 2: Create Auto Scaling Group (ASG)

2 **Create Auto Scaling Group**

```
AWS Console → EC2 → Auto Scaling → Auto Scaling Groups → Create
```

Configuration:

- Name: `my-app-asg`
- Launch Template: Select `my-app-template`
- VPC: Select your VPC
- Subnets: Select 2-3 subnets (different AZs for HA)

🎯 **Key ASG Settings:**

- **Min Capacity:** 1 (minimum instances always running)
- **Desired Capacity:** 2 (initial number of instances)
- **Max Capacity:** 4 (maximum during peak load)

Step 3: Configure Load Balancer

3 **Create Application Load Balancer (ALB)**

```
AWS Console → EC2 → Load Balancing → Load Balancers → Create
```

Configuration:

- Type: **Application Load Balancer**
- Name: `my-app-alb`
- Scheme: **Internet-facing**
- Subnets: Select same subnets as ASG
- Listener Port: **80 (HTTP)**

Create Target Group:

- Name: `my-app-targets`
- Target Type: **Instances**
- Protocol: **HTTP**
- Port: **80**
- Health Check Path: /
- Healthy Threshold: 2
- Unhealthy Threshold: 3
- Interval: 30 seconds

Step 4: Attach Load Balancer to ASG

4 **Link ALB to Auto Scaling Group**

```
Edit ASG → Load Balancing → Select Target Group
```

This connects:

- Incoming traffic through ALB
- Distributed to ASG instances
- Health checks through target group

Step 5: Create Scaling Policies

5 **Configure Scaling Policies (Target Tracking)**

```
Edit ASG → Automatic Scaling → Target Tracking Scaling Policies
```

Policy 1: Scale Out (Add Instances)

Metric Type: **Average CPU Utilization**
Target Value: **70%**
When CPU > 70%, ASG adds instances

Policy 2: Scale In (Remove Instances)

Metric Type: **Average CPU Utilization**
Target Value: **30%**
When CPU < 30%, ASG removes instances

⏱️ **Important Cooldown Settings:**

Scale-out Cooldown: 60 seconds (wait before adding more)
Scale-in Cooldown: 300 seconds (wait before removing)

Step 6: Monitor with CloudWatch

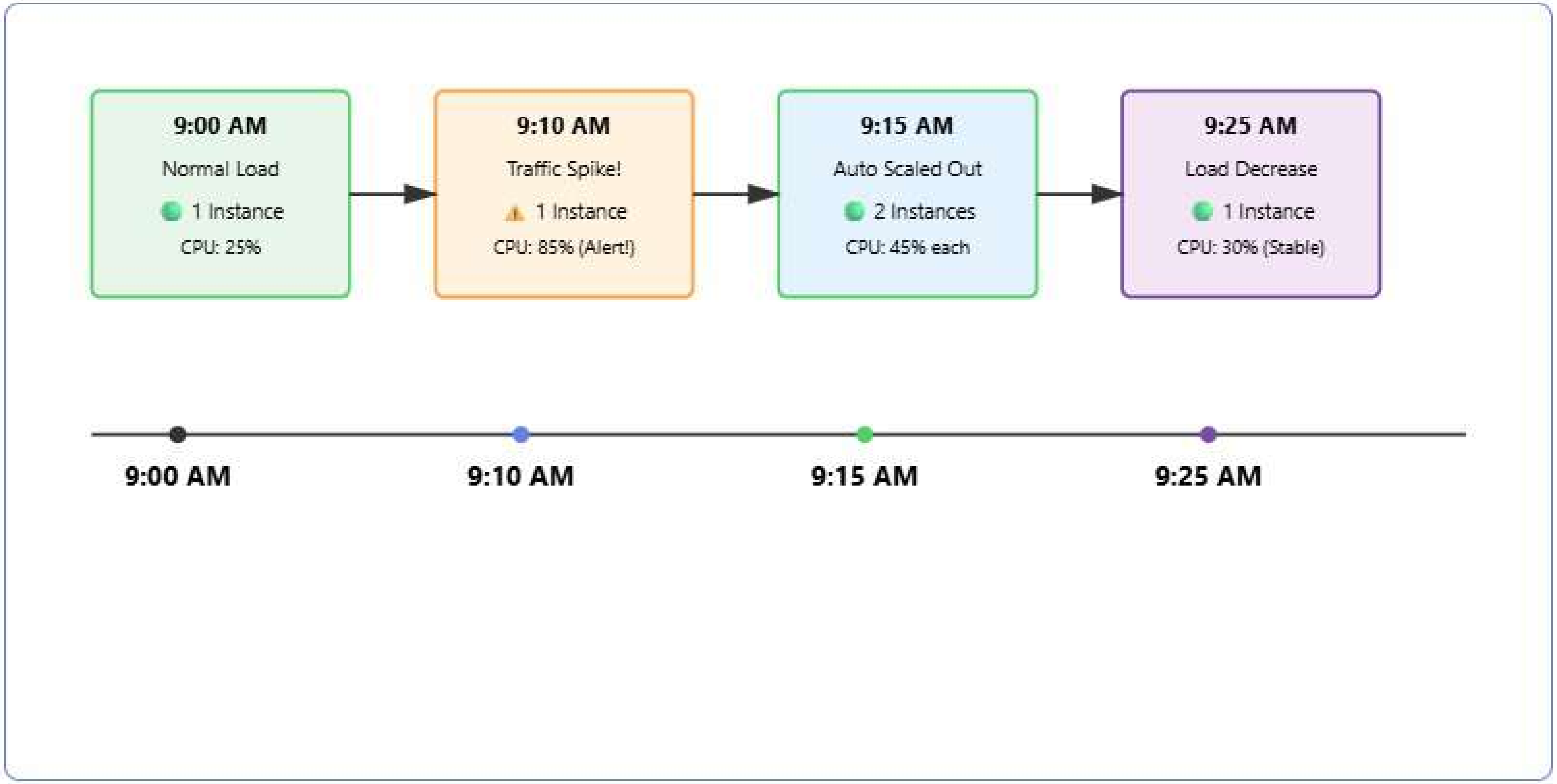
6 **Setup CloudWatch Monitoring**

```
AWS Console → CloudWatch → Dashboards → Create Custom Dashboard
```

Metrics to Monitor:

ASG GroupInServiceInstances
ASG GroupTerminatingInstances
Average CPU Utilization
ALB TargetResponseTime
ALB RequestCount

📈 How Auto Scaling Works - Timeline



💻 Practical Example - Setup from Scratch

Your Scenario: E-commerce Website

You have a website currently running on 1 t3.micro instance. During sales, traffic can spike 10x. You need auto scaling.

✅ **Solution Implementation:**

Component	Configuration	Reason
Instance Type	t3.micro (free tier) or t3.small	Good for web apps, cost-effective
Min Capacity	1	Always have 1 running (cost savings)
Desired Capacity	2	Start with 2 for redundancy
Max Capacity	5	Cap at 5 to control costs
Scale Out Threshold	CPU ≥ 70%	Add instance when busy
Scale In Threshold	CPU ≤ 30%	Remove instance when idle

Implementation Script (AWS CLI)

```
# 1. Create Launch Template aws ec2 create-launch-template \ --launch-template-
name my-ecommerce-template \ --version-description "Web app template" \ --
launch-template-data '{ "ImageId":"ami-0c55b159cbfafe1f0",
"InstanceType":"t3.micro", "KeyName":"my-key-pair", "SecurityGroupIds":["sg-
12345678"] }' # 2. Create Auto Scaling Group aws autoscaling create-auto-
scaling-group \ --auto-scaling-group-name my-ecommerce-asg \ --launch-template
LaunchTemplateName=my-ecommerce-template \ --min-size 1 \ --desired-capacity 2 \
--max-size 5 \ --availability-zones us-east-1a us-east-1b \ --target-
```