AWS Auto Scaling & Load Balancing

Auto Scaling Groups, Launch Templates & Elastic Load Balancers

Complete Guide to AWS Scalability and High Availability

Prepared By: Rashi Rana

AWS Corporate Trainer

Table of Contents

🚀 Launch Templates

- Introduction & Benefits
- Configuration Options
- Best Practices

📈 Auto Scaling Groups

- Core Concepts
- Scaling Policies
- Health Checks

⚖️ Elastic Load Balancers

- Application Load Balancer
- Network Load Balancer
- Gateway Load Balancer

🔧 Integration & Best Practices

- ASG + ELB Integration
- Monitoring & Troubleshooting
- Cost Optimization

Prepared By: Rashi Rana
AWS Corporate Trainer

AWS Launch Templates

Launch Templates are a newer and more advanced way to specify instance configuration for Auto Scaling Groups, providing versioning and more advanced features compared to Launch Configurations.

Key Benefits of Launch Templates

- **Versioning:** Create multiple versions and track changes
- **Parameter Overrides:** Override specific parameters at launch
- **Mixed Instance Types:** Support for Spot and On-Demand instances
- **Advanced Features:** T2/T3 unlimited, placement groups, dedicated hosts
- **Source Templates:** Create templates from existing instances

Feature	Launch Configuration	Launch Template
Versioning	❌ No	✅ Yes
Mixed Instance Types	❌ No	✅ Yes
Spot Instances	❌ Limited	✅ Full Support
T2/T3 Unlimited	❌ No	✅ Yes
Newer Instance Types	❌ Limited	✅ All Types

Prepared By: Rashi Rana
AWS Corporate Trainer

Launch Template Configuration

Essential Configuration Parameters

🖥️ Instance Details

- AMI ID
- Instance Type
- Key Pair
- Security Groups

💾 Storage

- EBS Volume Configuration
- Volume Type & Size
- Encryption Settings
- Delete on Termination

🌐 Network Settings

- VPC & Subnet
- Public IP Assignment
- Security Group IDs
- Network Interfaces

⚙️ Advanced Options

- User Data Scripts
- IAM Instance Profile
- Monitoring (Detailed)
- Placement Groups

```
# AWS CLI Example - Create Launch Template
aws ec2 create-launch-template \
    --launch-template-name MyWebServerTemplate \
    --launch-template-data '{
        "ImageId": "ami-0abcdef1234567890",
        "InstanceType": "t3.micro",
        "KeyName": "my-key-pair",
        "SecurityGroupIds": ["sg-12345678"],
        "UserData": "IyEvYmluL2Jhc2gK...",
        "IamInstanceProfile": {
            "Name": "EC2-SSM-Role"
        },
        "Monitoring": {
            "Enabled": true
        }
    }'
```

Prepared By: Rashi Rana
AWS Corporate Trainer

Auto Scaling Groups (ASG)

Auto Scaling Groups automatically adjust the number of EC2 instances in response to demand, ensuring optimal performance and cost efficiency.

Core ASG Concepts

- **Desired Capacity:** Target number of instances to maintain
- **Minimum Size:** Minimum number of instances (never go below)
- **Maximum Size:** Maximum number of instances (never exceed)
- **Health Checks:** EC2 and/or ELB health check types
- **Availability Zones:** Distribute instances across multiple AZs

ASG Lifecycle

1️⃣ Launch

ASG launches instances using Launch Template configuration

2️⃣ Health Check

Monitors instance health using EC2 or ELB health checks

3️⃣ Scale Out/In

4️⃣ Terminate

Adds or removes instances based on scaling policies

Terminates unhealthy or excess instances gracefully

⚠️ Important Notes

- ASG is free - you only pay for the EC2 instances launched
- Always distribute instances across multiple Availability Zones
- Use health check grace period to allow instances to initialize

Prepared By: Rashi Rana
AWS Corporate Trainer

Auto Scaling Policies

Types of Scaling Policies

📊 Target Tracking

Recommended approach

- Maintain specific metric target
- CPU Utilization (e.g., 70%)
- Request Count per Target
- Network In/Out

📈 Step Scaling

More granular control

- Scale based on CloudWatch alarms
- Different scaling amounts for different thresholds
- Faster response than simple scaling

📅 Scheduled Scaling

Predictable patterns

- Scale based on time/date
- Recurring schedules
- One-time events

🎯 Predictive Scaling

ML-powered scaling

- Uses machine learning
- Analyzes historical data
- Proactive scaling

```
# Target Tracking Scaling Policy Example
aws autoscaling put-scaling-policy \
    --auto-scaling-group-name my-asg \
    --policy-name cpu-target-tracking \
    --policy-type TargetTrackingScaling \
    --target-tracking-configuration '{
        "TargetValue": 70.0,
        "PredefinedMetricSpecification": {
            "PredefinedMetricType":
"ASGAverageCPUUtilization"
        }
    }'
```

✅ Best Practices for Scaling Policies

- Start with Target Tracking for CPU utilization
- Set appropriate cooldown periods to prevent flapping
- Use multiple metrics for comprehensive scaling decisions
- Test scaling policies in non-production environments first

Prepared By: Rashi Rana
AWS Corporate Trainer

Elastic Load Balancers (ELB)

Elastic Load Balancers automatically distribute incoming application traffic across multiple targets, such as EC2 instances, containers, and IP addresses, in multiple Availability Zones.

Key Benefits of ELB

- **High Availability:** Distributes traffic across multiple AZs
- **Health Checks:** Routes traffic only to healthy targets
- **Security:** Integrates with AWS security services
- **Elasticity:** Automatically scales to handle traffic
- **Monitoring:** Detailed metrics and logging

Types of Load Balancers

Load Balancer Type	Layer	Protocol Support	Use Cases
Application Load Balancer (ALB)	Layer 7 (Application)	HTTP, HTTPS, gRPC	Web applications, microservices
Network Load Balancer (NLB)	Layer 4 (Transport)	TCP, UDP, TLS	High performance, low latency

Load Balancer Type	Layer	Protocol Support	Use Cases
Gateway Load Balancer (GWLB)	Layer 3 (Network)	GENEVE	Third-party appliances
Classic Load Balancer (CLB)	Layer 4 & 7	HTTP, HTTPS, TCP, SSL	Legacy applications (deprecated)

Prepared By: Rashi Rana
AWS Corporate Trainer

Target Groups

Target Groups are used to route requests to one or more registered targets (EC2 instances, IP addresses, Lambda functions) using the protocol and port number you specify.

Key Concepts

- **Targets:** The destinations that receive traffic from the load balancer
- **Health Checks:** Determine if targets are healthy and available
- **Routing Rules:** Define how requests are distributed to targets
- **Sticky Sessions:** Route requests from the same client to the same target

Target Types

Target Type	Description	Use Cases	Load Balancer Support
Instance	EC2 instances by instance ID	Traditional EC2 deployments	ALB, NLB

Target Type	Description	Use Cases	Load Balancer Support
IP	Targets by IP address	On-premises, containers, multiple NICs	ALB, NLB
Lambda	Lambda functions	Serverless applications	ALB only
ALB	Application Load Balancer	Multi-tier architectures	NLB only

🏥 Health Check Settings

- Protocol (HTTP, HTTPS, TCP)
- Path (/health, /status)
- Port (80, 443, custom)
- Interval (15-300 seconds)
- Timeout (2-120 seconds)
- Healthy/Unhealthy thresholds

⚙️ Advanced Features

- Deregistration delay
- Load balancing algorithm
- Stickiness configuration
- Slow start mode
- Cross-zone load balancing
- Preserve client IP

```
# Create Target Group
aws elbv2 create-target-group \
    --name my-web-servers \
    --protocol HTTP \
    --port 80 \
    --vpc-id vpc-12345678 \
    --health-check-path /health \
    --health-check-interval-seconds 30 \
    --healthy-threshold-count 2 \
    --unhealthy-threshold-count 3
```

Prepared By: Rashi Rana
AWS Corporate Trainer

Application Load Balancer (ALB)

Application Load Balancer operates at Layer 7 (Application layer) and is ideal for HTTP and HTTPS traffic with advanced routing capabilities.

🎯 Advanced Routing

- Path-based routing (/api, /images)
- Host-based routing (api.example.com)
- Query string routing
- HTTP header routing

🔒 Security Features

- SSL/TLS termination
- AWS WAF integration
- Security groups
- Authentication (OIDC, SAML)

📊 Target Types

- EC2 instances
- IP addresses
- Lambda functions
- ECS containers

⚡ Performance

- HTTP/2 support
- WebSocket support
- Sticky sessions
- Connection draining

ALB Routing Rules Example

- **Path Pattern:** /api/* → API Target Group
- **Host Header:** admin.example.com → Admin Target Group
- **Query String:** ?version=v2 → V2 Target Group

- **HTTP Method:** POST requests → Processing Target Group

```
# Create ALB with AWS CLI
aws elbv2 create-load-balancer \
    --name my-application-load-balancer \
    --subnets subnet-12345678 subnet-87654321 \
    --security-groups sg-12345678 \
    --scheme internet-facing \
    --type application \
    --ip-address-type ipv4
```

Prepared By: Rashi Rana
AWS Corporate Trainer

Network Load Balancer (NLB)

Network Load Balancer operates at Layer 4 (Transport layer) and is designed for high performance, low latency, and handling millions of requests per second.

⚡ Ultra Performance

- Millions of requests/second
- Ultra-low latency
- Static IP addresses
- Elastic IP support

🌐 Protocol Support

- TCP traffic
- UDP traffic
- TLS traffic
- TCP_UDP (mixed)

🎯 Target Types

- EC2 instances
- IP addresses
- Application Load Balancers
- On-premises servers

🔧 Features

- Source IP preservation
- Cross-zone load balancing
- Health checks
- Connection draining

Feature	Application LB	Network LB
Latency	~100ms	~100μs
Static IP	❌ No	✅ Yes

Feature	Application LB	Network LB
Source IP Preservation	❌ No (X-Forwarded-For)	✅ Yes
SSL Termination	✅ Yes	✅ Yes (TLS)
Content-based Routing	✅ Yes	❌ No

⚠️ NLB Considerations

- Security groups don't apply to NLB - configure target security groups
- Cross-zone load balancing is disabled by default (charges apply when enabled)
- Health checks are less sophisticated than ALB

Prepared By: Rashi Rana
AWS Corporate Trainer

Gateway Load Balancer (GWLB)

Gateway Load Balancer operates at Layer 3 (Network layer) and is designed for deploying, scaling, and managing third-party network virtual appliances.

Key Use Cases

- **Firewalls:** Deploy third-party firewall appliances
- **Intrusion Detection:** IDS/IPS systems
- **Deep Packet Inspection:** Advanced security analysis
- **Payload Manipulation:** Content filtering and modification

🔧 Architecture

- GENEVE protocol (port 6081)
- Transparent network gateway
- Single entry/exit point
- Flow hash algorithm

🎯 Target Types

- EC2 instances
- IP addresses
- Third-party appliances
- Auto Scaling Groups

⚖️ Load Balancing

- 5-tuple hash (src IP, dst IP, src port, dst port, protocol)

🔗 Integration

- VPC Endpoint Services
- Route Tables

- Flow stickiness
- Symmetric routing
- Health checks

- Transit Gateway
- AWS Marketplace

```
# GWLB Traffic Flow
1. Client sends traffic to application
2. Route table directs traffic to GWLB Endpoint
3. GWLB forwards traffic to security appliance
4. Appliance processes and returns traffic
5. GWLB forwards processed traffic to destination
6. Response follows same path in reverse
```

✅ GWLB Benefits

- Transparent insertion of security appliances
- Horizontal scaling of appliances
- High availability across multiple AZs
- Simplified network architecture

Prepared By: Rashi Rana
AWS Corporate Trainer

ASG + ELB Integration

Auto Scaling Groups and Elastic Load Balancers work together to provide a highly available, scalable, and resilient architecture.

Integration Benefits

- **Automatic Registration:** New instances automatically register with load balancer
- **Health Checks:** ELB health checks can trigger ASG instance replacement
- **Traffic Distribution:** Load balancer distributes traffic across all healthy instances
- **Graceful Scaling:** Connection draining during scale-in events

Health Check Types

Health Check Type	What it Checks	When to Use
EC2 Health Check	Instance status, system status	Basic infrastructure health
ELB Health Check	Application response to health check requests	Application-level health verification

```
# Create ASG with ELB integration
```

```
aws autoscaling create-auto-scaling-group \
    --auto-scaling-group-name my-asg \
    --launch-template
LaunchTemplateName=MyTemplate,Version=1 \
    --min-size 2 \
    --max-size 10 \
    --desired-capacity 4 \
    --target-group-arns
arn:aws:elasticloadbalancing:region:account:targetgroup/my-
targets/1234567890123456 \
    --health-check-type ELB \
    --health-check-grace-period 300 \
    --vpc-zone-identifier "subnet-12345678,subnet-87654321"
```

Prepared By: Rashi Rana
AWS Corporate Trainer

Best Practices & Recommendations

🚀 Launch Templates

- Use Launch Templates over Launch Configurations
- Version your templates for change tracking
- Include detailed monitoring
- Use IAM instance profiles

📈 Auto Scaling

- Start with Target Tracking policies
- Distribute across multiple AZs
- Set appropriate health check grace periods
- Use multiple scaling metrics

⚖️ Load Balancers

- Choose the right LB type for your use case
- Enable access logs for troubleshooting
- Configure proper health checks
- Use SSL/TLS certificates

💰 Cost Optimization

- Use Spot Instances in ASG
- Right-size your instances
- Monitor and adjust scaling policies
- Use Reserved Instances for baseline capacity

✅ Architecture Checklist

- Multi-AZ deployment for high availability
- Proper security group configuration
- CloudWatch monitoring and alarms
- Regular testing of scaling policies
- Backup and disaster recovery planning

Prepared By: Rashi Rana
AWS Corporate Trainer

Monitoring & Troubleshooting

Key Metrics to Monitor

📊 ASG Metrics

- GroupDesiredCapacity
- GroupInServiceInstances
- GroupTotalInstances
- GroupMinSize/GroupMaxSize

⚖️ ELB Metrics

- RequestCount
- TargetResponseTime
- HTTPCode_Target_2XX_Count
- UnHealthyHostCount

🖥️ EC2 Metrics

- CPUUtilization
- NetworkIn/NetworkOut
- StatusCheckFailed
- DiskReadOps/DiskWriteOps

🚨 Alarms

- High CPU utilization
- Unhealthy targets
- Scaling activity failures
- Load balancer errors

⚠️ Common Issues & Solutions

- **Instances not launching:** Check IAM permissions, subnet capacity, security groups
- **Health check failures:** Verify application startup time, health check path
- **Scaling not working:** Review scaling policies, CloudWatch metrics, cooldown periods

- **Load balancer timeouts:** Check target health, security groups, NACLs

Prepared By: Rashi Rana
AWS Corporate Trainer

Summary

Key Takeaways

- **Launch Templates** provide versioning and advanced features over Launch Configurations
- **Auto Scaling Groups** ensure optimal capacity and high availability
- **Elastic Load Balancers** distribute traffic and improve application resilience
- **Integration** of ASG + ELB creates robust, scalable architectures

Next Steps

🧪 Hands-on Practice

- Create Launch Templates
- Set up Auto Scaling Groups
- Configure Load Balancers
- Test scaling scenarios

📚 Further Learning

- Advanced scaling strategies
- Blue/Green deployments
- Container orchestration
- Serverless architectures

Thank You!

Questions & Discussion

Prepared By: Rashi Rana
AWS Corporate Trainer