# ASSIGNMENT 2

## Developer Operations

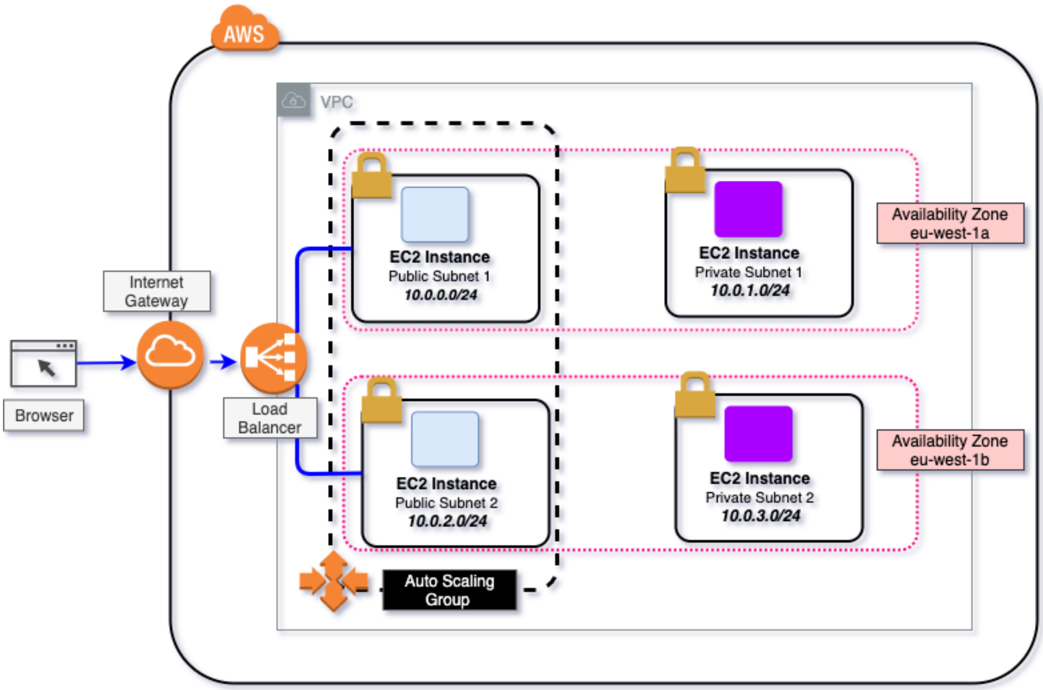
Loti Ibrahimi
20015453
Internet of Things

## Table of Contents

# Introduction/Overview

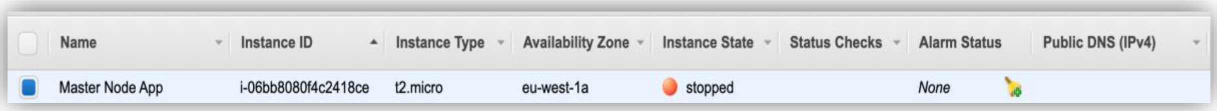
This report contains a breakdown of the deployment and automated management of a load-balanced auto-scaling web application.

## Architecture Diagram



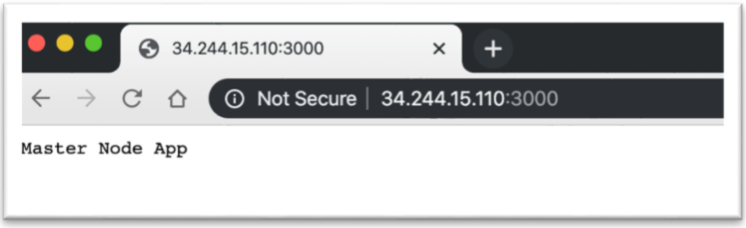
# Core assignment specification

## Step 1 - ‘Master’ Instance Configuration:



| □ | Name | Instance ID | Instance Type | Availability Zone | Instance State | Status Checks | Alarm Status | Public DNS (IPv4) |
|---|---|---|---|---|---|---|---|---|
| ■ | Master Node App | i-06bb8080f4c2418ce | t2.micro | eu-west-1a | ● stopped | | *None* | |

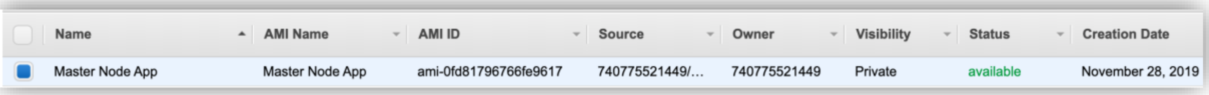
Master Node App (Master instance):



```
Master Node App
```

## Step 2 - Creation of custom AMI (for auto-scaling):

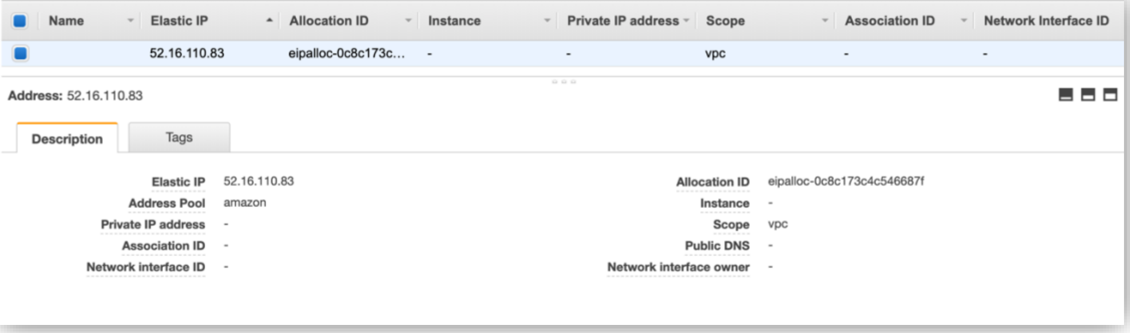
Created AMI of the ‘Master Node App’ from Step 1.



| □ | Name | AMI Name | AMI ID | Source | Owner | Visibility | Status | Creation Date |
|---|---|---|---|---|---|---|---|---|
| ■ | Master Node App | Master Node App | ami-0fd81796766fe9617 | 740775521449/... | 740775521449 | Private | available | November 28, 2019 |

## Step 3 - Creation of a VPC:

Firstly, an elastic IP Address allocated:

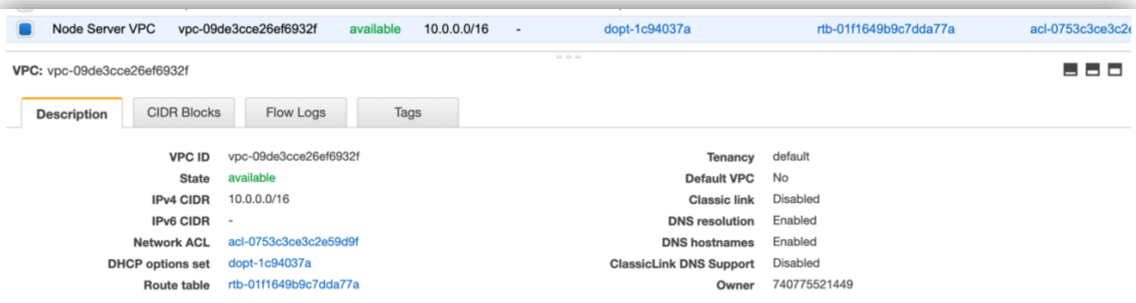


| Name | Elastic IP | Allocation ID | Instance | Private IP address | Scope | Association ID | Network Interface ID |
|---|---|---|---|---|---|---|---|
| | 52.16.110.83 | eipalloc-0c8c173c... | - | - | vpc | - | - |

**Address:** 52.16.110.83

| | | | |
|---|---|---|---|
| Elastic IP | 52.16.110.83 | Allocation ID | eipalloc-0c8c173c4c546687f |
| Address Pool | amazon | Instance | - |
| Private IP address | - | Scope | vpc |
| Association ID | - | Public DNS | - |
| Network interface ID | - | Network interface owner | - |

Then VPC created with Public/Private subnets, specifying NAT gateway using the elastic IP ID from the one allocated above:

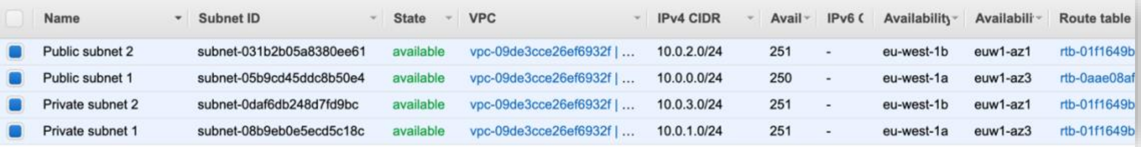


| | | | | | | | |
|---|---|---|---|---|---|---|---|
| Node Server VPC | vpc-09de3cce26ef6932f | available | 10.0.0.0/16 | - | dopt-1c94037a | rtb-01f1649b9c7dda77a | acl-0753c3ce3c2 |

**VPC:** vpc-09de3cce26ef6932f

| | | | |
|---|---|---|---|
| VPC ID | vpc-09de3cce26ef6932f | Tenancy | default |
| State | available | Default VPC | No |
| IPv4 CIDR | 10.0.0.0/16 | Classic link | Disabled |
| IPv6 CIDR | - | DNS resolution | Enabled |
| Network ACL | acl-0753c3ce3c2e59d9f | DNS hostnames | Enabled |
| DHCP options set | dopt-1c94037a | ClassicLink DNS Support | Disabled |
| Route table | rtb-01f1649b9c7dda77a | Owner | 740775521449 |

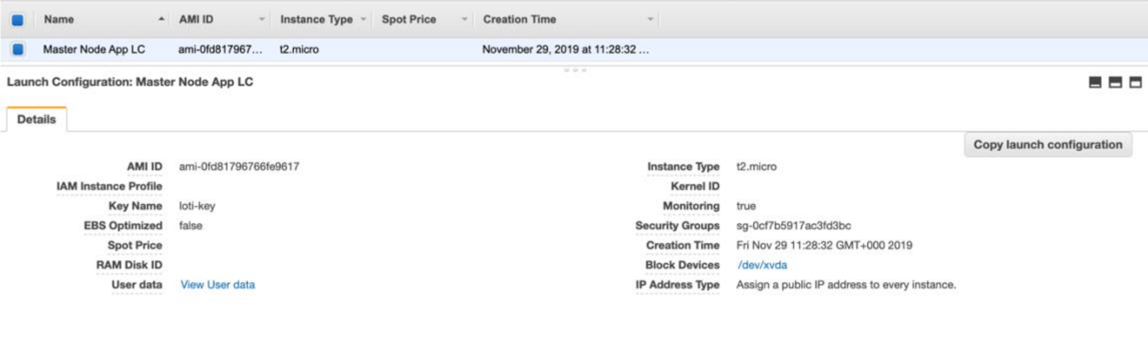
Public/Private subnets on different availability zones:



| Name | Subnet ID | State | VPC | IPv4 CIDR | Avail | IPv6 ( | Availability | Availabili | Route table |
|---|---|---|---|---|---|---|---|---|---|
| Public subnet 2 | subnet-031b2b05a8380ee61 | available | vpc-09de3cce26ef6932f \| ... | 10.0.2.0/24 | 251 | - | eu-west-1b | euw1-az1 | rtb-01f1649b |
| Public subnet 1 | subnet-05b9cd45ddc8b50e4 | available | vpc-09de3cce26ef6932f \| ... | 10.0.0.0/24 | 250 | - | eu-west-1a | euw1-az3 | rtb-0aae08af |
| Private subnet 2 | subnet-0daf6db248d7fd9bc | available | vpc-09de3cce26ef6932f \| ... | 10.0.3.0/24 | 251 | - | eu-west-1b | euw1-az1 | rtb-01f1649b |
| Private subnet 1 | subnet-08b9eb0e5ecd5c18c | available | vpc-09de3cce26ef6932f \| ... | 10.0.1.0/24 | 251 | - | eu-west-1a | euw1-az3 | rtb-01f1649b |

## Step 4 - Launch Configuration based on custom AMI:

Launch Config. based on custom AMI (Master Node App AMI):



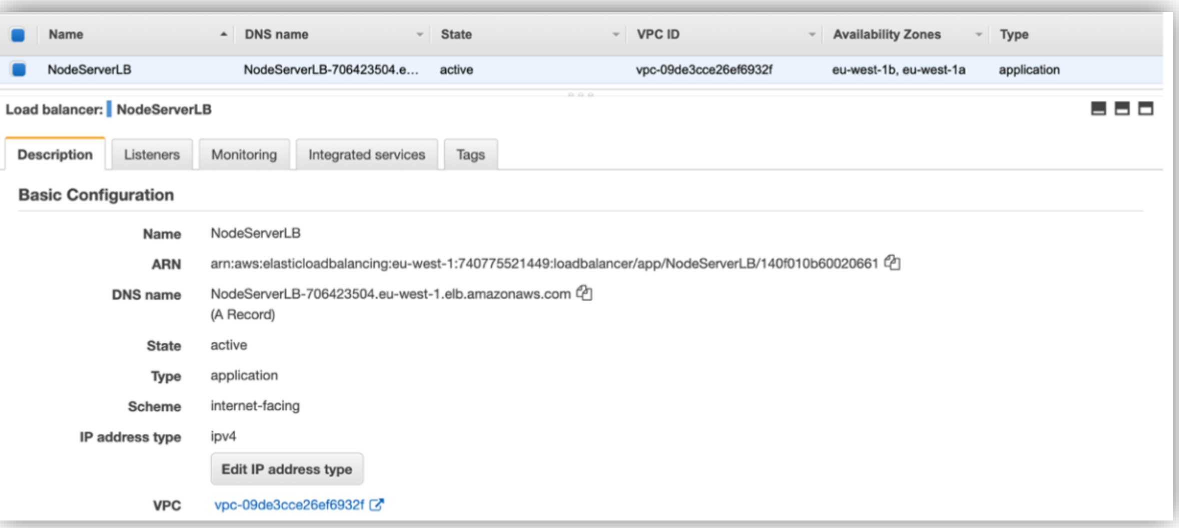
| Name | AMI ID | Instance Type | Spot Price | Creation Time |
|---|---|---|---|---|
| Master Node App LC | ami-0fd817967... | t2.micro | | November 29, 2019 at 11:28:32 ... |

**Launch Configuration: Master Node App LC**

| | | | |
|---|---|---|---|
| AMI ID | ami-0fd81796766fe9617 | Instance Type | t2.micro |
| IAM Instance Profile | | Kernel ID | |
| Key Name | loti-key | Monitoring | true |
| EBS Optimized | false | Security Groups | sg-0cf7b5917ac3fd3bc |
| Spot Price | | Creation Time | Fri Nov 29 11:28:32 GMT+000 2019 |
| RAM Disk ID | | Block Devices | /dev/xvda |
| User data | View User data | IP Address Type | Assign a public IP address to every instance. |

## Step 5 - Creation of elastic load balancer:

Elastic Load Balancer:



| Name | DNS name | State | VPC ID | Availability Zones | Type |
|---|---|---|---|---|---|
| NodeServerLB | NodeServerLB-706423504.e... | active | vpc-09de3cce26ef6932f | eu-west-1b, eu-west-1a | application |

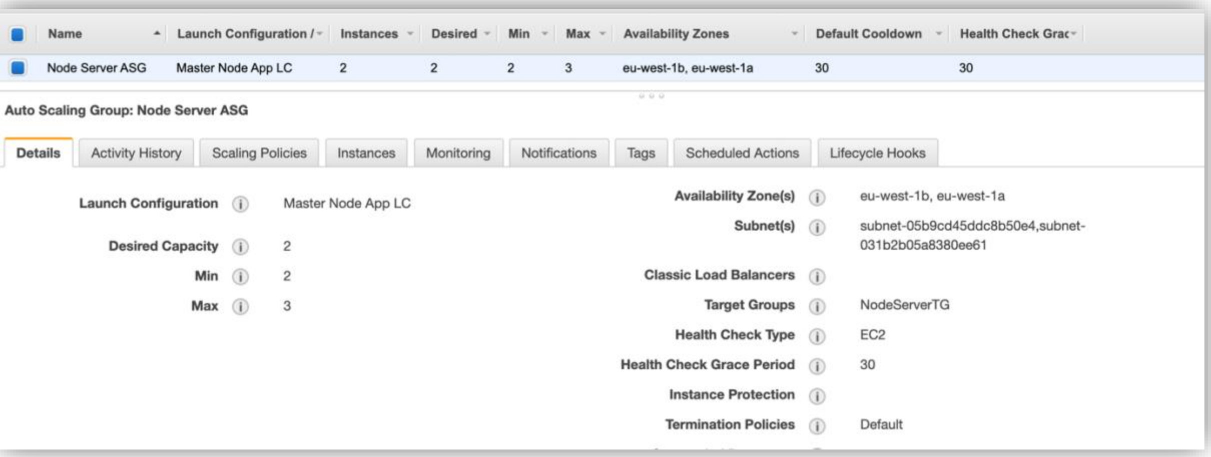
Load balancer: NodeServerLB

Basic Configuration

| | |
|---|---|
| Name | NodeServerLB |
| ARN | arn:aws:elasticloadbalancing:eu-west-1:740775521449:loadbalancer/app/NodeServerLB/140f010b60020661 |
| DNS name | NodeServerLB-706423504.eu-west-1.elb.amazonaws.com<br>(A Record) |
| State | active |
| Type | application |
| Scheme | internet-facing |
| IP address type | ipv4 |
| | Edit IP address type |
| VPC | vpc-09de3cce26ef6932f |

## Step 6 - Creation of auto-scaling group:

Auto Scaling group created based on the *Launch Configuration* in Step 4;
Only public subnets selected - means scaled instances will launch behind a public subnet & will be accessible/visible by the load balancer.



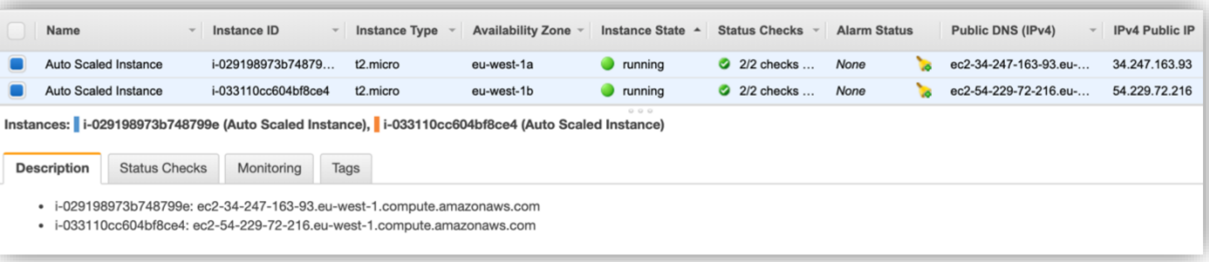
| Name | Launch Configuration / | Instances | Desired | Min | Max | Availability Zones | Default Cooldown | Health Check Grac |
|---|---|---|---|---|---|---|---|---|
| Node Server ASG | Master Node App LC | 2 | 2 | 2 | 3 | eu-west-1b, eu-west-1a | 30 | 30 |

Auto Scaling Group: Node Server ASG

| | |
|---|---|
| Launch Configuration | Master Node App LC |
| Desired Capacity | 2 |
| Min | 2 |
| Max | 3 |

| | |
|---|---|
| Availability Zone(s) | eu-west-1b, eu-west-1a |
| Subnet(s) | subnet-05b9cd45ddc8b50e4,subnet-031b2b05a8380ee61 |
| Classic Load Balancers | |
| Target Groups | NodeServerTG |
| Health Check Type | EC2 |
| Health Check Grace Period | 30 |
| Instance Protection | |
| Termination Policies | Default |

Auto Scaled Instances:



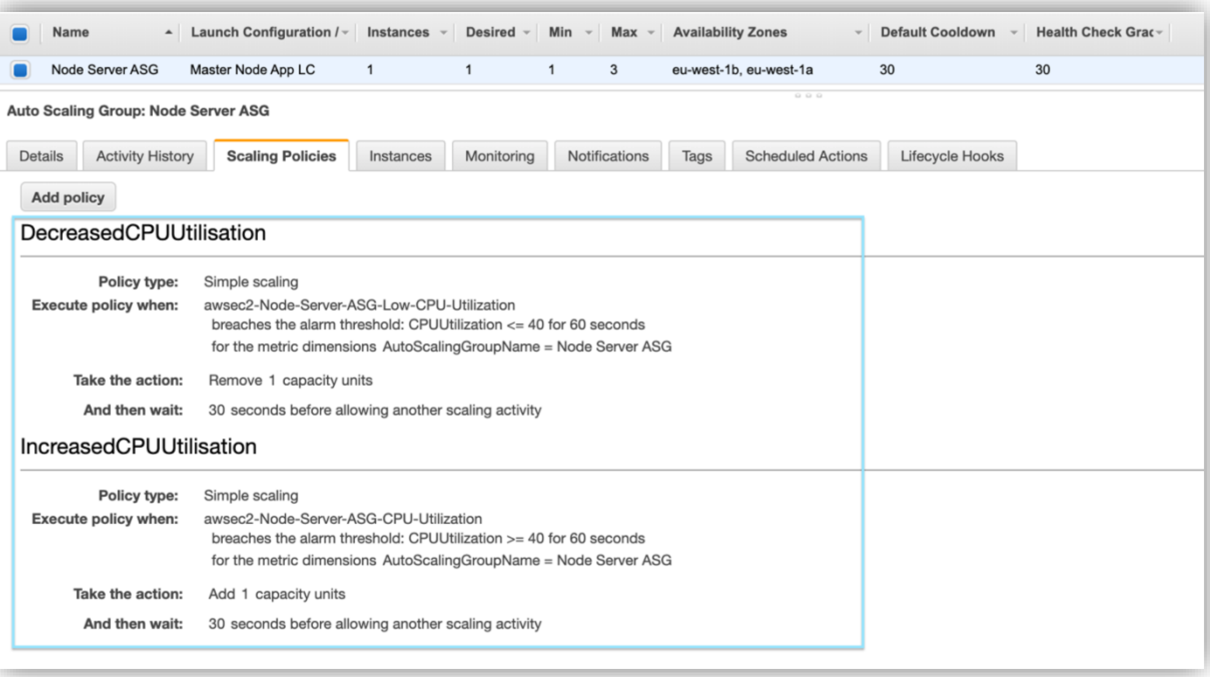
| Name | Instance ID | Instance Type | Availability Zone | Instance State | Status Checks | Alarm Status | Public DNS (IPv4) | IPv4 Public IP |
|---|---|---|---|---|---|---|---|---|
| Auto Scaled Instance | i-029198973b74879... | t2.micro | eu-west-1a | ● running | ✓ 2/2 checks ... | *None* | ec2-34-247-163-93.eu-... | 34.247.163.93 |
| Auto Scaled Instance | i-033110cc604bf8ce4 | t2.micro | eu-west-1b | ● running | ✓ 2/2 checks ... | *None* | ec2-54-229-72-216.eu-... | 54.229.72.216 |

Instances: i-029198973b748799e (Auto Scaled Instance), i-033110cc604bf8ce4 (Auto Scaled Instance)

- i-029198973b748799e: ec2-34-247-163-93.eu-west-1.compute.amazonaws.com
- i-033110cc604bf8ce4: ec2-54-229-72-216.eu-west-1.compute.amazonaws.com

## Step 7 - Creation of auto-scaling policy:

Simple Scaling policies:



| Name | Launch Configuration / | Instances | Desired | Min | Max | Availability Zones | Default Cooldown | Health Check Grac |
|---|---|---|---|---|---|---|---|---|
| Node Server ASG | Master Node App LC | 1 | 1 | 1 | 3 | eu-west-1b, eu-west-1a | 30 | 30 |

Auto Scaling Group: Node Server ASG

DecreasedCPUUtilisation

Policy type: Simple scaling

Execute policy when: awsec2-Node-Server-ASG-Low-CPU-Utilization
breaches the alarm threshold: CPUUtilization <= 40 for 60 seconds
for the metric dimensions AutoScalingGroupName = Node Server ASG

Take the action: Remove 1 capacity units

And then wait: 30 seconds before allowing another scaling activity

IncreasedCPUUtilisation

Policy type: Simple scaling

Execute policy when: awsec2-Node-Server-ASG-CPU-Utilization
breaches the alarm threshold: CPUUtilization >= 40 for 60 seconds
for the metric dimensions AutoScalingGroupName = Node Server ASG

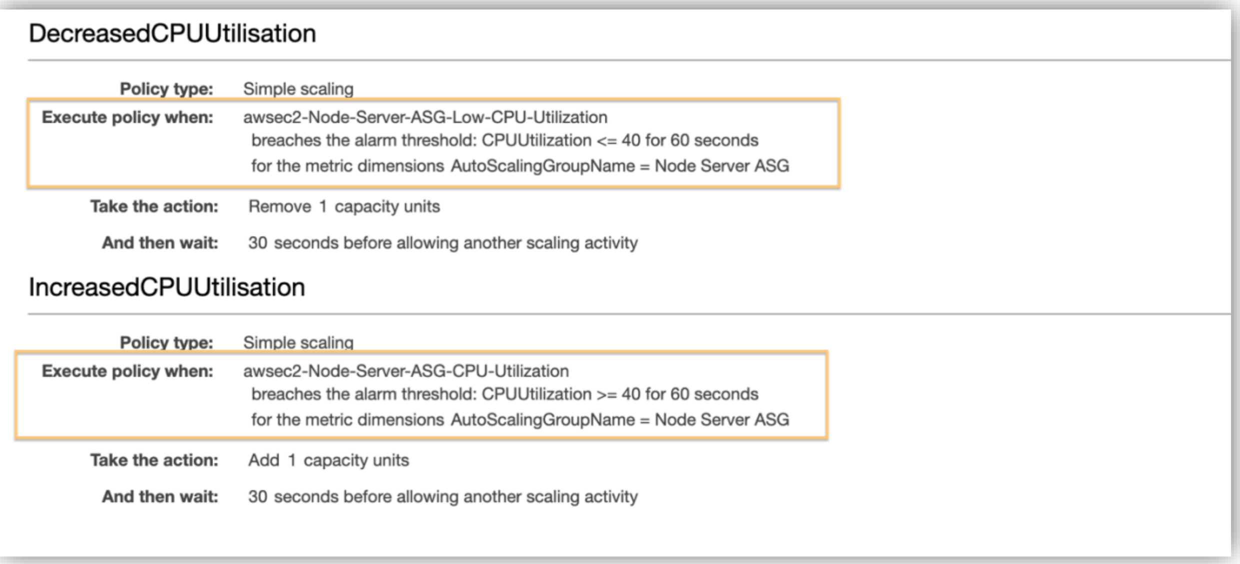
Take the action: Add 1 capacity units

And then wait: 30 seconds before allowing another scaling activity

## Step 8 - CloudWatch alarm to trigger increase in resources:

CPU Utilisation was the chosen metric because it's a key performance indicator for applications.

It is important to know the amount of resources being used by the server. Based on certain thresholds, we can increase/decrease available instances to facilitate this demand.



DecreasedCPUUtilisation

Policy type: Simple scaling

Execute policy when: awsec2-Node-Server-ASG-Low-CPU-Utilization
breaches the alarm threshold: CPUUtilization <= 40 for 60 seconds
for the metric dimensions AutoScalingGroupName = Node Server ASG

Take the action: Remove 1 capacity units

And then wait: 30 seconds before allowing another scaling activity

IncreasedCPUUtilisation

Policy type: Simple scaling

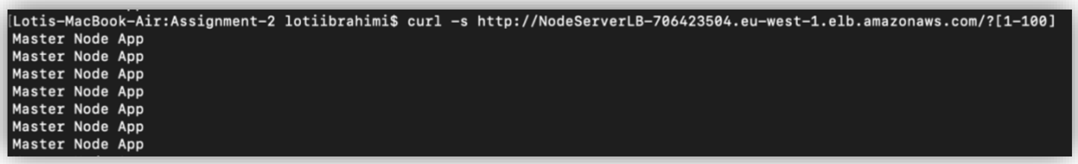
Execute policy when: awsec2-Node-Server-ASG-CPU-Utilization
breaches the alarm threshold: CPUUtilization >= 40 for 60 seconds
for the metric dimensions AutoScalingGroupName = Node Server ASG

Take the action: Add 1 capacity units

And then wait: 30 seconds before allowing another scaling activity

### Step 9 - Generation of test traffic to load balancer:

Generating traffic to the Load Balancer using following command:
**curl -s http://** [http://nodeserverlb-706423504.eu-west-1.elb.amazonaws.com/**?[1-100**](http://nodeserverlb-706423504.eu-west-1.elb.amazonaws.com/?[1-100)**]**



```
[Lotis-MacBook-Air:Assignment-2 lotiibrahimi$ curl -s http://NodeServerLB-706423504.eu-west-1.elb.amazonaws.com/?[1-100]
Master Node App
Master Node App
Master Node App
Master Node App
Master Node App
Master Node App
Master Node App
```

### Step 10 - Distributed load (logs or web server)

### Step 11 - Server Activity monitor script:

## Architecture Analysis (AWS Well-Architected Framework):

### Operational Excellence:

CloudWatch monitoring and scaling policies, allow for the load-balancing & auto scaling structure to deliver business value. By automating changes & responding to events, it ensures a successful management of daily operations to cope with demand.

### Reliability:

The usage of a load balancer & directing traffic to different subnets on the network ensures changes are in sync across different availability zones. A fault in one will not disrupt any customer demands. This highlights the ability to prevent, & quickly recover from failures.

### Performance Efficiency:

Auto scaling is a prime example of computing resources efficiently, with the aid of CloudWatch monitoring. This system determines the amount of resources required and provides the necessary amount. In this assignment, it was evident in the scaling of instances in the Node app, based on server CPU Utilisation. It's a structure which maintains efficiency as the platform evolves.

### Cost Optimization:

As highlighted in 'Performance Efficiency' above, the system in place also focuses on avoiding un-needed costs by monitoring & controlling the appropriate number of resources being spent. This ultimately limits/prevents unnecessary overspending.

### Security:

Through the use of configured security groups, systems and information are protected to some extent, establishing port interfaces for inbound/outbound requests.