

# Creating a Scalable and Highly Available Environment for the Café

## TASK 1 — Update Network for Multi-AZ High Availability

### Task 1.1 — Create NAT Gateway in Public Subnet 2

#### Steps

1. Open **VPC Console**.
2. Left menu → **NAT Gateways**.
3. Click **Create NAT Gateway**.
4. Configure:
  - **Subnet:** Public Subnet 2
  - **Elastic IP:** Allocate Elastic IP
  - **Name:** NAT-GW-AZ2
5. Click **Create NAT Gateway**.

**Create NAT gateway** Info

A highly available, managed Network Address Translation (NAT) service that instances in private subnets can use to connect to services in other VPCs, on-premises networks, or the internet.

**NAT gateway settings**

**Name - optional**  
Create a tag with a key of 'Name' and a value that you specify.

The name can be up to 256 characters long.

**Subnet**  
Select a subnet in which to create the NAT gateway.

**Connectivity type**  
Select a connectivity type for the NAT gateway.  
 Public  
 Private

**Elastic IP allocation ID** Info  
Assign an Elastic IP address to the NAT gateway.

### Update Private Subnet 2 Route Table

1. VPC console → **Route Tables**.
2. Find the route table for **Private Subnet 2**
3. Select it → **Routes** tab → **Edit routes**.
4. Add route:
  - **Destination:** 0.0.0.0/0
  - **Target:** NAT Gateway → NAT-GW-AZ2

5. Click **Save changes**.

Destination	Target	Status	Propagated	Route Origin
10.0.0.0/16	local	Active	No	CreateRouteTable
	Q_ local	X		
	NAT Gateway	Active	No	CreateRoute
	Q_ nat-00018f6648f2b7446	X		

Add route      Cancel      Preview      Save changes

## TASK 2 — Create a Launch Template

### 1. Open Launch Templates

EC2 Console → left menu → **Launch Templates** → **Create launch template**

### 2. Template Information

- Name: CafeWebServerTemplate

### 3. AMI

- Application & OS Images → **My AMIs**
- Select: **Cafe WebServer Image**

### 4. Instance Type

- Choose: **t2.micro**

### 5. Key Pair

- vokey

### 6. Network Settings

- Security Group: **CafeSG**

### 7. Tags

Add tag:

- Key:** Name
- Value:** webserver
- Resource:** Instances

### 8. IAM Role

- Under Advanced details:
  - IAM Instance Profile:** CafeRole

### 9. Create

Click **Create launch template**.

The screenshot shows the AWS EC2 Launch Templates page. On the left, there's a navigation sidebar with links like Dashboard, EC2 Global View, Instances, Launch Templates (which is selected), Images, and Elastic Block Store. The main content area has a header 'Launch Templates (1/1)'. Below it is a table with one row, showing the following details:

Launch Template ID	Launch Template Name	Default Version	Latest Version	Create Time	Created By
lt-0fd12de1a72db7cc9	CafeWebServerTemplate	1	1	2025-11-18T07:10:38.000Z	arnawssts:6374232...

Below the table, there's a section titled 'CafeWebServerTemplate (lt-0fd12de1a72db7cc9)' with tabs for 'Launch template details', 'Versions', and 'Template tags'. The 'Launch template details' tab shows the launch template ID, name, default version, and owner information. The owner is listed as 'Owner arnawssts:6374232...'. There are also 'Actions' and 'Delete template' buttons.

## TASK 3 — Create Application Load Balancer

### 1. Open Load Balancer Console

EC2 → Load Balancers → **Create Load Balancer** → Application Load Balancer

### 2. ALB Configuration

- Name: CafeALB
- Scheme: **Internet-facing**
- IP type: IPv4
- VPC: *lab VPC*
- Subnets:
  - Public Subnet 1
  - Public Subnet 2
- Select security group

### 3. Create Target Group

- Target type: **Instances**
- Name: CafeTargetGroup
- Protocol: **HTTP**
- Health check path: /cafe
- Click **Create target group**

### 4.Create

#### Create Load Balancer

The screenshot shows the AWS EC2 Load Balancers console. On the left, there's a navigation sidebar with options like Elastic Block Store, Network & Security, Load Balancing, and Auto Scaling. The main area displays a table titled 'Load balancers (1/1)'. The table has columns for Name, State, Type, Scheme, IP address type, VPC ID, and Availability Zones. One row is selected, showing 'CafeALB' as the name, 'Active' as the state, 'application' as the type, 'Internet-facing' as the scheme, 'IPv4' as the IP address type, 'vpc-091f9a3f317680236' as the VPC ID, and '2 Availability Zones'. Below the table, a detailed view for 'Load balancer: CafeALB' is shown with tabs for Details, Listeners and rules, Network mapping, Resource map, Security, Monitoring, Integrations, Attributes, and Capacity. The 'Details' tab is selected, displaying information such as Load balancer type (Application), Status (Active), VPC (vpc-091f9a3f317680236), and Load balancer IP address type (IPv4).

## TASK 4 — Create Auto Scaling Group

### 1. Open ASG Console

EC2 → Auto Scaling Groups → Create Auto Scaling group

### 2. Basic Details

- ASG name: CafeWebServerASG
- Launch template: CafeWebServerTemplate

### 3. Network

- VPC: *your lab VPC*
- Subnets (select both):
  - Private Subnet 1
  - Private Subnet 2

### 4. Group Size

- Desired:** 2
- Min:** 2
- Max:** 6

### 5. Scaling Policy

- Select **Target tracking scaling policy**
  - Metric: **Average CPU Utilization**
  - Target: **25%**
  - Instance warmup: **60 sec**

### 6. Create ASG

Click **Create Auto Scaling group**.

Auto Scaling groups (1/1) [Info](#)

Last updated less than a minute ago [Launch configurations](#) [Launch templates](#) [Actions](#) [Create Auto Scaling group](#)

Search your Auto Scaling groups

Name	Launch template/configuration	Instances	Status	Desired capacity	Min	Max	Availability Zones
<input checked="" type="checkbox"/> CafeWebServerASG	CafeWebServerTemplate   Version Default	2	-	2	2	6	2 Availability Zones

Auto Scaling group: CafeWebServerASG

Details Integrations Automatic scaling Instance management Instance refresh Activity Monitoring Tags - moved

**CafeWebServerASG Capacity overview**

arn:aws:autoscaling:us-east-1:637423272929:autoScalingGroup:cc5fc03b-b996-40b9-8630-139cd5d30f21:autoScalingGroupName/CafeWebServerASG

Desired capacity	Scaling limits (Min - Max)	Desired capacity type	Status
2	2 - 6	Units (number of instances)	-

## Task 5 — Testing and Validation

### 5.1 Application Reachability Test

The ALB DNS name was used to verify that the café application was accessible:

<http://cafealb-641114072.us-east-1.elb.amazonaws.com/cafe/>

This confirmed:

- Instances were healthy
- Target group was functioning
- ALB routing was operational

AWS Academy Cloud ... Challenge (Café) lat: ... Systems Manager | u ... Instances | EC2 | us- ... Auto Scaling group: ... Café! RouteTables | VPC Co ... All Bookmarks

Not secure cafealb-641114072.us-east-1.elb.amazonaws.com/cafe/ MATCH (MyAnatom... AWS Academy MSLCM LMS

**Café**

Home About Us Contact Us Menu Order History

Frank bakes a rich variety of cookies. Try them all!

Tea, Coffee, Lattes.

Our tarts are always a customer favorite!

## 5.2 Auto Scaling Stress Test

To validate autoscaling behavior, a stress test was performed using **AWS Systems Manager Session Manager**.

### Commands to Run:

```
sudo amazon-linux-extras install epel
```

```
sudo yum install stress -y
```

```
stress --cpu 1 --timeout 600
```

This generated artificial CPU load.

The ASG detected CPU >25% and initiated **scale-out**, launching additional EC2 instances.

Monitoring the ASG dashboard confirmed:

- Scaling policies were applied correctly
- New instances launched in both private subnets
- Load gradually balanced across instances

```
Installing:
 stress                               x86_64                         1.0.4-16.el7                    epel                           39 k
Transaction Summary
Install 1 Package
Total download size: 39 k
Installed size: 94 k
Downloading packages:
warning: /var/cache/yum/x86_64/2/epel/packages/stress-1.0.4-16.el7.x86_64.rpm: Header V3 RSA/SHA256 Signature, key ID 352c64e5: NOKEY
Public key for stress-1.0.4-16.el7.x86_64.rpm is not installed
stress-1.0.4-16.el7.x86_64.rpm
Retrieving key from file:///etc/pki/rpm-gpg/RPM-GPG-KEY-EPEL-7
Importing GPG key 0x352C64E5:
 Userid   : "Fedora EPEL (7) <epel@fedoraproject.org>"
 Fingerprint: 91e9 7d7c 4a5e 96f1 7f3e 888f 6a2f aea2 352c 64e5
 Package   : epel-release-7-11.noarch (@amzn2extra-epel)
 From     : /etc/pki/rpm-gpg/RPM-GPG-KEY-EPEL-7
Running transaction check
Running transaction test
transaction test succeeded
Running transaction
  Installing : stress-1.0.4-16.el7.x86_64
  Verifying   : stress-1.0.4-16.el7.x86_64
                                                               1/1
                                                               1/1
Installed:
  stress.x86_64 0:1.0.4-16.el7
Complete!
rh 4.25 stress --cpu 1 --timeout 600
stress: info: [3367] dispatching hogs: 1 cpu, 0 io, 0 vm, 0 hdd
```

The screenshot shows the AWS CloudWatch Metrics Insights interface. A query is being run against CloudWatch Metrics data. The query is as follows:

```
CloudWatchMetricsQuery {
    MetricName: "CPUUtilization", 
    Namespace: "AWS/CloudWatchMetricsInsightsTest", 
    StartTime: "2018-06-01T00:00:00Z", 
    EndTime: "2018-06-01T01:00:00Z", 
    Statistics: "Average", 
    Dimensions: [
        { Name: "AutoScalingGroupName", Value: "CafeWebServerASG" }
    ]
}
```

The results of the query are displayed in a table. There is one row of data:

AutoScalingGroupName	CloudWatchMetricsInsightsTest	CPUUtilization	Timestamp
CafeWebServerASG	CloudWatchMetricsInsightsTest	100.0	2018-06-01T00:00:00Z

**Instances (4) Info**

Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public IPv4
webserver	i-043869f27f8393563	Running	t2.micro	2/2 checks passed	<a href="#">View alarms +</a>	us-east-1a	-
webserver	i-0cdc809a4a99aa173	Shutting-down	t2.micro	-	<a href="#">View alarms +</a>	us-east-1a	-
webserver	i-0f94678aeba34169a	Running	t2.micro	2/2 checks passed	<a href="#">View alarms +</a>	us-east-1b	-
CafeWebAppS...	i-002df20ccf01c176d	Running	t2.micro	2/2 checks passed	<a href="#">View alarms +</a>	us-east-1a	ec2-54-196-11-145

**Auto Scaling groups (1) Info**

Name	Launch template/configuration	Instances	Status	Desired capacity	Min	Max
CafeWebServerASG	CafeWebServerTemplate   Version Default	6	Updating capacity...	3	2	6

**Instances (9) Info**

Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public IPv4
webserver	i-043869f27f8393563	Running	t2.micro	2/2 checks passed	<a href="#">View alarms +</a>	us-east-1a	-
webserver	i-0cdc809a4a99aa173	Terminated	t2.micro	-	<a href="#">View alarms +</a>	us-east-1a	-
webserver	i-0f94678aeba34169a	Terminated	t2.micro	-	<a href="#">View alarms +</a>	us-east-1b	-
webserver	i-0763a15c42aedd3f4	Running	t2.micro	2/2 checks passed	<a href="#">View alarms +</a>	us-east-1b	-
webserver	i-0b36ce2b70e23b2d2	Running	t2.micro	Initializing	<a href="#">View alarms +</a>	us-east-1b	-
webserver	i-0ac8d48636c050975	Running	t2.micro	Initializing	<a href="#">View alarms +</a>	us-east-1b	-
CafeWebAppS...	i-002df20ccf01c176d	Running	t2.micro	2/2 checks passed	<a href="#">View alarms +</a>	us-east-1a	ec2-54-196-11-145
webserver	i-0f6bb9080fb56fa	Running	t2.micro	Initializing	<a href="#">View alarms +</a>	us-east-1a	-
webserver	i-078dd64e1b4e6ae0	Running	t2.micro	Initializing	<a href="#">View alarms +</a>	us-east-1a	-

## Conclusion

This lab successfully demonstrated the deployment of a highly available, fault tolerant, and scalable web application architecture using core AWS services. A multi-AZ design combined with NAT Gateways, private subnets, an ALB, and an Auto Scaling Group ensures robust performance and resiliency.