

## Auto Scaling Group (ASG) Web Application Deployment Documentation

 **Author:** EC2 GROUP

 **Date:** April 2025

### **Objective**

To document the step-by-step process of setting up an EC2 instance, configuring a launch template, Auto Scaling Group (ASG), and ensuring every instance launched by the ASG runs the same web application.

## Initial Setup

### VPC and Network Setup

#### 1. VPC Creation:

A Virtual Private Cloud (VPC) was created to logically isolate the network for the application infrastructure.

#### 2. Subnets:

a. **Public Subnets:** Intended use, Load Balancer and ASG.

3. **Private Subnets:** Intended for use with EC2 and database.

#### 4. Route Tables:

a. Public subnets are associated with a route table that includes a route to the Internet Gateway for outbound internet access.

b. (Optional) Private subnets would typically use a NAT Gateway for outbound access, though not used here.

#### 5. Internet Gateway (IGW):

Attached to the VPC to allow public subnet traffic to access the internet.

#### 6. Security Groups:

Configured to allow inbound HTTP (port 80), HTTPS (443), and SSH (22) access.

#### 7. Network Access Control Lists (NACLs):

Set up to permit the same ports as security groups for both inbound and outbound traffic on public subnets.

You have successfully updated outbound rules for acl-0a983c2bad127ff46

### acl-0a983c2bad127ff46

Details		Associated with		Default		VPC ID																			
Network ACL ID	acl-0a983c2bad127ff46	4 Subnets		Yes		vpc-0e1c5dbbc23b3a62b / EC2-vpc																			
Owner	168705872550																								
<a href="#">Inbound rules</a>   <a href="#">Outbound rules</a>   <a href="#">Subnet associations</a>   <a href="#">Tags</a>																									
<a href="#">Edit inbound rules</a>																									
<b>Inbound rules (2)</b> <table border="1"> <thead> <tr> <th>Rule number</th> <th>Type</th> <th>Protocol</th> <th>Port range</th> <th>Source</th> <th>Allow/Deny</th> </tr> </thead> <tbody> <tr> <td>100</td> <td>All traffic</td> <td>All</td> <td>All</td> <td>0.0.0.0/0</td> <td><span style="color: green;">Allow</span></td> </tr> <tr> <td>*</td> <td>All traffic</td> <td>All</td> <td>All</td> <td>0.0.0.0/0</td> <td><span style="color: red;">Deny</span></td> </tr> </tbody> </table>								Rule number	Type	Protocol	Port range	Source	Allow/Deny	100	All traffic	All	All	0.0.0.0/0	<span style="color: green;">Allow</span>	*	All traffic	All	All	0.0.0.0/0	<span style="color: red;">Deny</span>
Rule number	Type	Protocol	Port range	Source	Allow/Deny																				
100	All traffic	All	All	0.0.0.0/0	<span style="color: green;">Allow</span>																				
*	All traffic	All	All	0.0.0.0/0	<span style="color: red;">Deny</span>																				

United States (N. Virginia) - Dragon-king @ 1687-0587-2550

### acl-02e31740882e7cf75

Details		Associated with		Default		VPC ID																									
Network ACL ID	acl-02e31740882e7cf75	4 Subnets		Yes		vpc-0d765784b4b26895e / EC2group-vpc																									
Owner	168705872550																														
<a href="#">Inbound rules</a>   <a href="#">Outbound rules</a>   <a href="#">Subnet associations</a>   <a href="#">Tags</a>																															
<a href="#">Edit inbound rules</a>																															
<b>Inbound rules (3)</b> <table border="1"> <thead> <tr> <th>Rule number</th> <th>Type</th> <th>Protocol</th> <th>Port range</th> <th>Source</th> <th>Allow/Deny</th> </tr> </thead> <tbody> <tr> <td>100</td> <td>All traffic</td> <td>All</td> <td>All</td> <td>0.0.0.0/0</td> <td><span style="color: green;">Allow</span></td> </tr> <tr> <td>120</td> <td>HTTP (80)</td> <td>TCP (6)</td> <td>80</td> <td>0.0.0.0/0</td> <td><span style="color: green;">Allow</span></td> </tr> <tr> <td>*</td> <td>All traffic</td> <td>All</td> <td>All</td> <td>0.0.0.0/0</td> <td><span style="color: red;">Deny</span></td> </tr> </tbody> </table>								Rule number	Type	Protocol	Port range	Source	Allow/Deny	100	All traffic	All	All	0.0.0.0/0	<span style="color: green;">Allow</span>	120	HTTP (80)	TCP (6)	80	0.0.0.0/0	<span style="color: green;">Allow</span>	*	All traffic	All	All	0.0.0.0/0	<span style="color: red;">Deny</span>
Rule number	Type	Protocol	Port range	Source	Allow/Deny																										
100	All traffic	All	All	0.0.0.0/0	<span style="color: green;">Allow</span>																										
120	HTTP (80)	TCP (6)	80	0.0.0.0/0	<span style="color: green;">Allow</span>																										
*	All traffic	All	All	0.0.0.0/0	<span style="color: red;">Deny</span>																										

**VPC dashboard**

**Outbound rules (2)**

Rule number	Type	Protocol	Port range	Destination	Allow/Deny
100	All traffic	All	All	0.0.0.0/0	Allow
*	All traffic	All	All	0.0.0.0/0	Deny

**sg-0a1c6997c89d44855 - EC2server**

**Inbound rules (2)**

Name	Security group rule ID	IP version	Type	Protocol	Port range	Source	Description
-	sgr-070f276df7f41f42	IPv4	HTTP	TCP	80	0.0.0.0/0	-
-	sgr-050c9ecd89ac2d6a	IPv4	SSH	TCP	22	0.0.0.0/0	-

**VPC dashboard**

**Virtual private cloud**

- Your VPCs
- Subnets
- Route tables
- Internet gateways
- Egress-only Internet gateways
- Carrier gateways
- DHCP option sets
- Elastic IPs
- Managed prefix lists
- NAT gateways
- Peering connections
- Route servers [New](#)

**Security**

- Network ACLs
- Security groups

**PrivateLink and Lattice**

- Getting started [Updated](#)
- Endpoints [Updated](#)
- Endpoint services
- Service networks [Updated](#)
- Lattice services

**CloudShell Feedback**

**VPC** > **Your VPCs** > **vpc-0e1c5dbbc23b3a62b**

**Resource map**

**Details**

VPC ID	vpc-0e1c5dbbc23b3a62b	State	Available	Block Public Access	Off
DNS resolution	Disabled	Tenancy	default	DHCP option set	dopt-026c858b2a43a386b
Main network ACL	acl-0a983c2bad127ff46	Default VPC	No	IPv4 CIDR	10.0.0.0/16
IPv6 CIDR (Network border group)	-	Network Address Usage metrics	Disabled	Route 53 Resolver DNS Firewall rule groups	-

**Resource map**

**Subnets (4)**

**Route tables (4)**

**Network connections (2)**

**VPC** > **Your VPCs** > **vpc-0e1c5dbbc23b3a62b**

**Actions**

**VPC dashboard**

**Virtual private cloud**

- Your VPCs
- Subnets
- Route tables
- Internet gateways
- Egress-only Internet gateways
- Carrier gateways
- DHCP option sets
- Elastic IPs
- Managed prefix lists
- NAT gateways
- Peering connections
- Route servers [New](#)

**Security**

- Network ACLs
- Security groups

**PrivateLink and Lattice**

- Getting started [Updated](#)
- Endpoints [Updated](#)
- Endpoint services
- Service networks [Updated](#)
- Lattice services

**CloudShell Feedback**

**VPC** > **Your VPCs** > **vpc-0e1c5dbbc23b3a62b / EC2-vpc**

**Actions**

**Details**

VPC ID	vpc-0e1c5dbbc23b3a62b	State	Available	Block Public Access	Off
DNS resolution	Disabled	Tenancy	default	DHCP option set	dopt-026c858b2a43a386b
Main network ACL	acl-0a983c2bad127ff46	Default VPC	No	IPv4 CIDR	10.0.0.0/16
IPv6 CIDR (Network border group)	-	Network Address Usage metrics	Disabled	Route 53 Resolver DNS Firewall rule groups	-

**Resource map**

**CIDRs**

**IPv4 CIDRs**

Address family	CIDR	Status
IPv4	10.0.0.0/16	Associated

**CloudShell Feedback**

## Auto Scaling Group Setup in the private subnets.

### Launch Template Creation

- A launch template was created based on the original EC2 instance.
- Includes instance type, AMI, security groups, and optionally User Data.

## ✓ ASG Linked to Launch Template

- **Screenshot:** EC2ASG.png, EC2ASG-2.png
- **Description:** Shows the Auto Scaling Group with its configuration attached to the launch template.

The screenshot shows the AWS Auto Scaling Groups console for the group 'EC2groupASG'. The 'Capacity overview' section displays a desired capacity of 1 instance. The 'Launch template' section shows details like AMI ID (ami-0d4407f5ba954b0d), Instance type (t2.micro), and Security group IDs (sg-0aa38b3b127e17ff). The 'Network' section lists Availability Zones (us-east-1a, us-east-1b) and Subnet IDs (subnet-01bb89ccfcbe4645, subnet-01a0a19a48e1ac6bf). The 'Instance type requirements' section indicates a balanced best effort distribution. Other sections shown include Health checks, Instance maintenance policy, Capacity Reservation preference, and Advanced configurations.

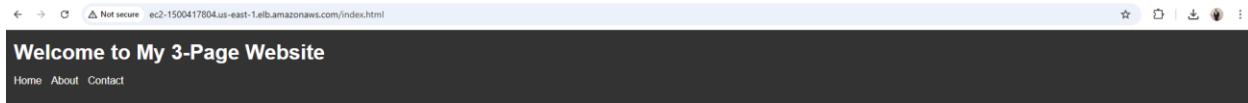
EC2 launched or spanned from ASG

## Load Balancer Setup

- Screenshot:** EC2GROUP-ELB-SETUP.png
- Description:** Displays the Load Balancer forwarding requests to instances within the ASG.

## Web Application Accessible via Elastic Load Balancer DNS

The app was uploaded through user date during Launch template setup or configuration.



#### Home Page Content

This is the home page of my simple website.



## CPU Stress Testing

### ✓ Test CPU Load

- **Screenshot:** CPU-STRESSED-TO-90%.png, ASG-CPU-STRESSED-72%.png
- **Description:** Load generated to test auto-scaling response.

```
ast metadata expiration check: 2:16:27 ago on Tue Apr  8 12:27:10 2025.
No match for argument: epel-release
Error: failed to find a match: epel-release
ast metadata expiration check: 2:16:28 ago on Tue Apr  8 12:27:10 2025.
Dependencies resolved.

Package          Architecture Version      Repository  Size
installing:
stress           x86_64       1.0.7-2.amzn2023.0.1
Transaction Summary
install 1 Package
total download size: 34 kB
installed size: 68 kB
downloading Packages:
stress-1.0.7-2.amzn2023.0.1.x86_64.rpm
total
running transaction check
transaction check succeeded.
transaction test succeeded.
running transaction
Preparing:
Installing   : stress-1.0.7-2.amzn2023.0.1.x86_64
Running scriptlet: stress-1.0.7-2.amzn2023.0.1.x86_64
Verifying    : stress-1.0.7-2.amzn2023.0.1.x86_64
Installed:
stress-1.0.7-2.amzn2023.0.1.x86_64

Complete!
root@ip-10-0-5-146 ~]# stress --cpu 2 --timeout 300
stress: info: [4355] dispatching hogs: 2 cpu(s) 0 io, 0 vm, 0 hdd
stress: info: [4355] stressful run completed: 0000000000000000
root@ip-10-0-5-146 ~]# stress-ng --cpu=8 --cpu-method all --verify --timeout 10m
stress-ng: info: [4545] setting to a 600 second (10 mins, 0.00 secs) run per stressor
stress-ng: info: [4545] dispatching hogs: 8 cpu
```

### ✓ Instances Scaled

- **Screenshot:** ASG-SPANNED-NEW-INSTANCE-TO COPE-WITH-THE-STRESS.png, 3RD-INSTANCE-SPANNED-TO-ACCOMODATE-THE-LOAD.png
- **Description:** New instances launched in response to high CPU usage.

**Instances (3) Info**

Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public IPv4 DNS	Public IPv4 ...	Elastic IP
EC2	i-0f3a883c90f0acab1	Running	t2.micro	Initializing	<a href="#">View alarms</a> +	us-east-1b	ec2-54-162-126-102.co...	54.162.126.102	-
EC2	i-06dbf71e3b347efaf7	Running	t2.micro	2/2 checks passed	<a href="#">View alarms</a> +	us-east-1a	ec2-98-80-138-235.co...	98.80.138.235	-
EC2	i-0871989bdac580d6e	Running	t2.micro	2/2 checks passed	<a href="#">View alarms</a> +	us-east-1b	ec2-54-165-194-111.co...	54.165.194.111	-

**Instances (2) Info**

Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public IPv4 DNS	Public IPv4 ...	Elastic IP
EC2	i-06dbf71e3b347efaf7	Running	t2.micro	2/2 checks passed	<a href="#">View alarms</a> +	us-east-1a	ec2-98-80-138-235.co...	98.80.138.235	-
EC2	i-0871989bdac580d6e	Running	t2.micro	2/2 checks passed	<a href="#">View alarms</a> +	us-east-1b	ec2-54-165-194-111.co...	54.165.194.111	-

## Database

### Prerequisites

- Your **EC2** instance and **RDS** are in the **same VPC**
- Your EC2 has **network access** (via route tables and security groups) to the RDS
- You have the **RDS endpoint** and port
- Your EC2 instance has the necessary **DB client tools** installed (e.g., mysql, psql)

## Step 1: Verify RDS Details

Make sure you have:

- **RDS endpoint:**  (database-1.cu5emkwymw55.us-east-1.rds.amazonaws.com)
- **Port:** e.g., 3306 (MySQL) or 5432 (PostgreSQL)
- **DB username & password**
- **DB engine** (MySQL, PostgreSQL, etc.)
- **VPC and subnet setup for RDS**

## Step 2: Security Group Settings

To allow EC2 to reach RDS:

*Go to: VPC > Security Groups*

- Find the **security group attached to the RDS**
- Add an **Inbound Rule:**
  - Type: MySQL/Aurora or PostgreSQL (depending on your DB engine)
  - Port: 3306 or 5432
  - Source: Select **Custom** and type the **EC2's security group ID**

## Step 3: Network Routing

- Make sure **EC2** and **RDS** are in the **same VPC**
- Their **subnets can route** to each other
- DNS resolution is enabled for EC2:
  - Go to **VPC > Your VPC > Edit DNS resolution** → Ensure both **DNS hostnames** and **DNS resolution** are enabled

This ensures only your EC2 instance can access the DB

## SSH INTO EC2 USING ENDPOINTS

### : Install MySQL Client on EC2

If you don't have internet access yet, set up a NAT Gateway or use a bastion EC2 in public subnet to proxy installs.

To install:

Copysudo yum install mysql -y

## Connect to RDS from EC2

Once the security group rule is set and MySQL client is installed:

```
mysql -h <your-db-endpoint> -u <username> -p
```

```
mysql -h database-1.cu5emkwymw55.us-east-1.rds.amazonaws.com -u admin -p
```

Enter password when prompted.

```
[ec2-user@ip-10-0-9-18 ~]$ sudo yum install mysql-community-client -y
last metadata expiration check: 0:00:54 ago on Thu Apr 10 11:11:29 2025.
No match for argument: mysql-community-client
Error: Nothing to do
[ec2-user@ip-10-0-9-18 ~]$ sudo dnf config-manager --enable mysql8.0
Error: No matching repo to modify: mysql8.0
[ec2-user@ip-10-0-9-18 ~]$ sudo yum install mysql-community-client -y
last metadata expiration check: 0:01:26 ago on Thu Apr 10 11:13:29 2025.
No match for argument: mysql-community-client
Error: Unable to find a match: mysql-community-client
[ec2-user@ip-10-0-9-18 ~]$ sudo dnf install mariadb105 -y
last metadata expiration check: 0:02:03 ago on Thu Apr 10 11:13:29 2025.
Dependencies resolved.
=====
| Package           | Architecture | Version      | Repository | Size |
|:-----|:-----|:-----|:-----|:-----|
| mariadb105       | x86_64       | 3:10.5.25-1.amzn2023.0.1 | amazonlinux | 1.6 M |
| mariadb dependencies:
|   mariadb-connector-c | x86_64 | 3.3.10-1.amzn2023.0.1 | amazonlinux | 211 k |
|   mariadb-connector-c-config | noarch | 3.3.10-1.amzn2023.0.1 | amazonlinux | 9.9 k |
|   mariadb105-common | x86_64 | 3:10.5.25-1.amzn2023.0.1 | amazonlinux | 29 k |
|   perl-Sys-Hostname | x86_64 | 1.23-477.amzn2023.0.6 | amazonlinux | 18 k |
=====
Transaction Summary
=====
Install 5 Packages

Total download size: 1.9 M
Installed size: 19 M
Downloading Packages:
(1/5): mariadb-connector-c-3.3.10-1.amzn2023.0.1.x86_64.rpm          3.4 MB/s | 211 kB     00:00
(2/5): mariadb-connector-c-config-3.3.10-1.amzn2023.0.1.noarch.rpm    145 kB/s | 9.9 kB     00:00
(3/5): mariadb105-common-10.5.25-1.amzn2023.0.1.x86_64.rpm        1.1 MB/s | 29 kB     00:00
(4/5): maril-Sys-Hostname-1.23-477.amzn2023.0.6.x86_64.rpm         535 kB/s | 18 kB     00:00
(5/5): mariadb105-10.5.25-1.amzn2023.0.1.x86_64.rpm            12 MB/s | 1.6 MB     00:00
=====
Total                                         11 MB/s | 1.9 MB     00:00
Running transaction check
transaction check succeeded.
Running transaction test
transaction test succeeded.
```

```

aws | Search [Alt+S] United States (N. Virginia) ▾ Dragon-king @ 1687-0587-2550 ▾
VPC EC2 S3 AWS Amplify Aurora and RDS CloudFormation CloudWatch

Upgrading mariadb105:
(1/5) mariadb-connector-c-3.3.10-1.amzn2023.0.1.x86_64.rpm
(2/5) mariadb-connector-c-config-3.3.10-1.amzn2023.0.1.noarch.rpm
(3/5) mariadb105-common-10.5.25-1.amzn2023.0.1.x86_64.rpm
(4/5) perl-Sys-Hostname-1.23-477.amzn2023.0.6.x86_64.rpm
(5/5) mariadb105-10.5.25-1.amzn2023.0.1.x86_64.rpm
Total 3.4 MB/s | 211 kB 00:00
145 kB/s | 9.9 kB 00:00
1.5 MB/s | 29 kB 00:00
535 kB/s | 18 kB 00:00
12 MB/s | 1.6 MB 00:00
11 MB/s | 1.9 MB 00:00

Running transaction check
Transaction check succeeded.
Running transaction test
Test transaction succeeded.

Preparing :
Installing : mariadb-connector-c-config-3.3.10-1.amzn2023.0.1.noarch 1/1
              : mariadb-connector-c-3.3.10-1.amzn2023.0.1.x86_64 1/5
Installing : mariadb105-common-10.5.25-1.amzn2023.0.1.x86_64 2/5
Installing : perl-Sys-Hostname-1.23-477.amzn2023.0.6.x86_64 3/5
Installing : mariadb105-10.5.25-1.amzn2023.0.1.x86_64 4/5
Running scriptlet: mariadb105-10.5.25-1.amzn2023.0.1.x86_64 5/5
Verifying : mariadb-connector-c-config-3.3.10-1.amzn2023.0.1.noarch 5/5
Verifying : mariadb105-common-10.5.25-1.amzn2023.0.1.x86_64 1/5
Verifying : mariadb105-10.5.25-1.amzn2023.0.1.x86_64 2/5
Verifying : mariadb105-common-10.5.25-1.amzn2023.0.1.x86_64 3/5
Verifying : perl-Sys-Hostname-1.23-477.amzn2023.0.6.x86_64 4/5
Verifying : mariadb105-10.5.25-1.amzn2023.0.1.x86_64 5/5

Installed:
  mariadb-connector-c-3.3.10-1.amzn2023.0.1.x86_64  mariadb-connector-c-config-3.3.10-1.amzn2023.0.1.noarch  mariadb105-10.5.25-1.amzn2023.0.1.x86_64  mariadb105-common-10.5.25-1.amzn2023.0.1.x86_64

Complete!
[ec2-user@ip-10-0-9-18 ~]$ mysql -h ec2group.cs5emkwym55.us-east-1.rds.amazonaws.com -u admin -p
Enter password:
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MySQL connection id is 28
Server version: 8.0.40 Source distribution

Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

MySQL [(none)]> 
```

```

aws | Search [Alt+S] United States (N. Virginia) ▾ Dragon-king @ 1687-0587-2550 ▾
VPC EC2 S3 AWS Amplify Aurora and RDS CloudFormation CloudWatch

--> name VARCHAR(100),
--> email VARCHAR(100),
--> created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
--> );
query OK, 0 rows affected (0.046 sec)

MySQL [sampleddb]> CREATE TABLE users (
-->   id INT AUTO_INCREMENT PRIMARY KEY,
-->   name VARCHAR(100),
-->   email VARCHAR(100),
-->   created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
--> );
ERROR 1050 (42S01): Table 'users' already exists
MySQL [sampleddb]> SELECT * FROM users;
empty set (0.000 sec)

MySQL [sampleddb]> INSERT INTO users (name, email) VALUES ('Emmanuel', 'emmanuel@example.com');
query OK, 1 row affected (0.004 sec)

MySQL [sampleddb]> SELECT * FROM users;
+----+----+----+
| id | name | email |
+----+----+----+
| 1 | Emmanuel | emmanuel@example.com |
+----+----+----+
row in set (0.001 sec)

MySQL [sampleddb]> INSERT INTO users (name, email) VALUES ('Mary Mensah', 'mary@example.com'),
--> ('Jeffery', 'jeffery@example.com'),
--> ('Priscilla', 'priscilla@example.com'),
--> ('Richard', 'richard@example.com');
query OK, 4 rows affected (0.005 sec)
records: 4 Duplicates: 0 Warnings: 0

MySQL [sampleddb]> SELECT * FROM users;
+----+----+----+
| id | name | email |
+----+----+----+
| 1 | Emmanuel | emmanuel@example.com |
| 2 | Mary Mensah | mary@example.com |
| 3 | Jeffery | jeffery@example.com |
| 4 | Priscilla | priscilla@example.com |
+----+----+----+
row in set (0.001 sec)

MySQL [sampleddb]> INSERT INTO users (name, email) VALUES ('Mary Mensah', 'mary@example.com'),
--> ('Jeffery', 'jeffery@example.com'),
--> ('Priscilla', 'priscilla@example.com'),
--> ('Richard', 'richard@example.com');
query OK, 4 rows affected (0.005 sec)
records: 4 Duplicates: 0 Warnings: 0

MySQL [sampleddb]> SELECT * FROM users;
+----+----+----+
| id | name | email |
+----+----+----+
| 1 | Emmanuel | emmanuel@example.com |
| 2 | Mary Mensah | mary@example.com |
| 3 | Jeffery | jeffery@example.com |
| 4 | Priscilla | priscilla@example.com |
| 5 | Richard | richard@example.com |
+----+----+----+
rows in set (0.001 sec)

MySQL [sampleddb]> 
```

```

aws | Search [Alt+S] United States (N. Virginia) ▾ Dragon-king @ 1687-0587-2550 ▾
VPC EC2 S3 AWS Amplify Aurora and RDS CloudFormation CloudWatch

MySQL [sampleddb]> UPDATE TABLE users$ (
-->   id INT AUTO_INCREMENT PRIMARY KEY,
-->   name VARCHAR(100),
-->   email VARCHAR(100),
-->   created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
--> );
query OK, 0 rows affected (0.000 sec)

MySQL [sampleddb]> SELECT * FROM users;
+----+----+----+
| id | name | email |
+----+----+----+
| 1 | Emmanuel | emmanuel@example.com |
+----+----+----+
row in set (0.001 sec)

MySQL [sampleddb]> INSERT INTO users (name, email) VALUES ('Emmanuel', 'emmanuel@example.com');
query OK, 1 row affected (0.004 sec)

MySQL [sampleddb]> SELECT * FROM users;
+----+----+----+
| id | name | email |
+----+----+----+
| 1 | Emmanuel | emmanuel@example.com |
+----+----+----+
row in set (0.001 sec)

MySQL [sampleddb]> INSERT INTO users (name, email) VALUES ('Mary Mensah', 'mary@example.com'),
--> ('Jeffery', 'jeffery@example.com'),
--> ('Priscilla', 'priscilla@example.com'),
--> ('Richard', 'richard@example.com');
query OK, 4 rows affected (0.005 sec)
records: 4 Duplicates: 0 Warnings: 0

MySQL [sampleddb]> SELECT * FROM users;
+----+----+----+
| id | name | email |
+----+----+----+
| 1 | Emmanuel | emmanuel@example.com |
| 2 | Mary Mensah | mary@example.com |
| 3 | Jeffery | jeffery@example.com |
| 4 | Priscilla | priscilla@example.com |
| 5 | Richard | richard@example.com |
+----+----+----+
rows in set (0.001 sec)

MySQL [sampleddb]> 
```

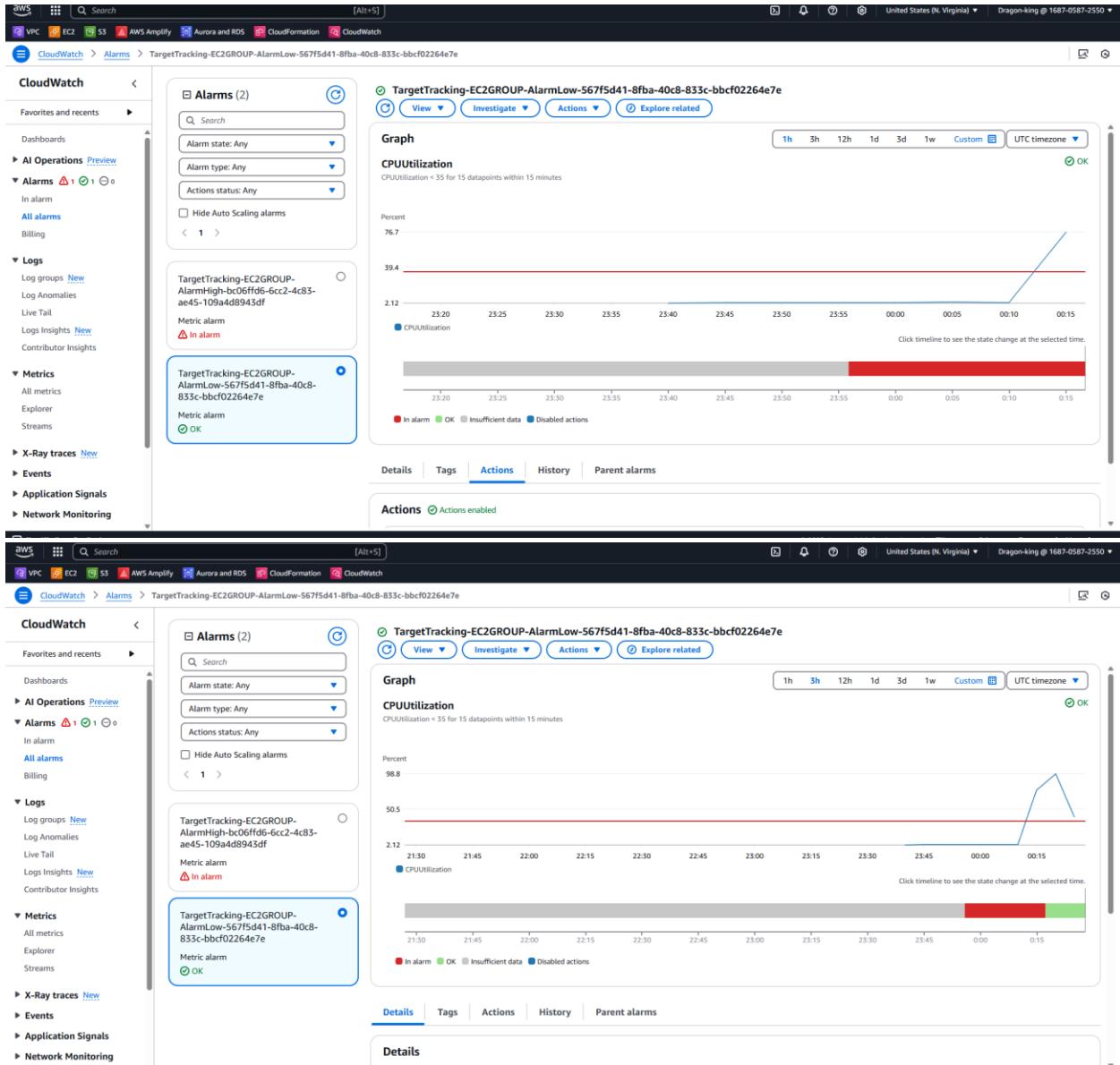
- ◆ **Monitoring & Logging**

## 6.1 Enable CloudWatch Monitoring

1. Go to **CloudWatch > Logs > Create Log Group**.
2. Name it: EC2group-Logs.
3. Go to **EC2 > CloudWatch Metrics** → Enable detailed monitoring.
4. Create **Alarms** for:
  - a. High CPU Usage (> 80%)
  - b. Low Available Memory

## 6.2 Enable VPC Flow Logs

1. Go to **VPC > Flow Logs > Create Flow Log**.
2. Choose:
  - a. **VPC:** EC2
  - b. **Destination:** CloudWatch Logs
  - c. **Filter:** Accept-only traffic
3. Click **Create**.



The screenshot shows the AWS CloudWatch Alarms interface. On the left, there's a navigation sidebar with links like Dashboards, AI Operations, Alarms, Logs, Metrics, X-Ray traces, Events, Application Signals, and Network Monitoring. The main area displays a table titled "Alarms (2)".

Name	State	Last state update (UTC)	Conditions	Actions
TargetTracking-EC2GROUP-AlarmHigh-bc06ffd6-6cc2-4c83-ae45-109a4d8943df	<span style="color: red;">⚠ In alarm</span>	2025-04-11 00:19:03	CPUUtilization > 50 for 3 datapoints within 3 minutes	<span style="color: green;">Actions enabled</span>
TargetTracking-EC2GROUP-AlarmLow-567f5d41-8fbab0c8-833c-bbcf02264e7e	<span style="color: green;">OK</span>	2025-04-11 00:17:21	CPUUtilization < 35 for 15 datapoints within 15 minutes	<span style="color: green;">Actions enabled</span>

## ✖ Issue Identified

### Problem

Instances launched by the Auto Scaling Group **did not run** the web application.

### Root Cause

The launch template did **not include** application deployment instructions.

## Solutions

### Option 1: Create a Custom AMI

1. Launch & configure EC2.
2. Install the application.
3. Go to EC2 > Actions > Image > Create Image.
4. Use the AMI in your launch template.

### Option 2: Use User Data Script

Add the following script in the launch template:

```
#!/bin/bash
yum update -y
yum install -y nginx unzip
cd /usr/share/nginx/html
aws s3 cp s3://ec2/livestock-management-1.zip .
unzip livestock-management-1.zip
systemctl restart nginx

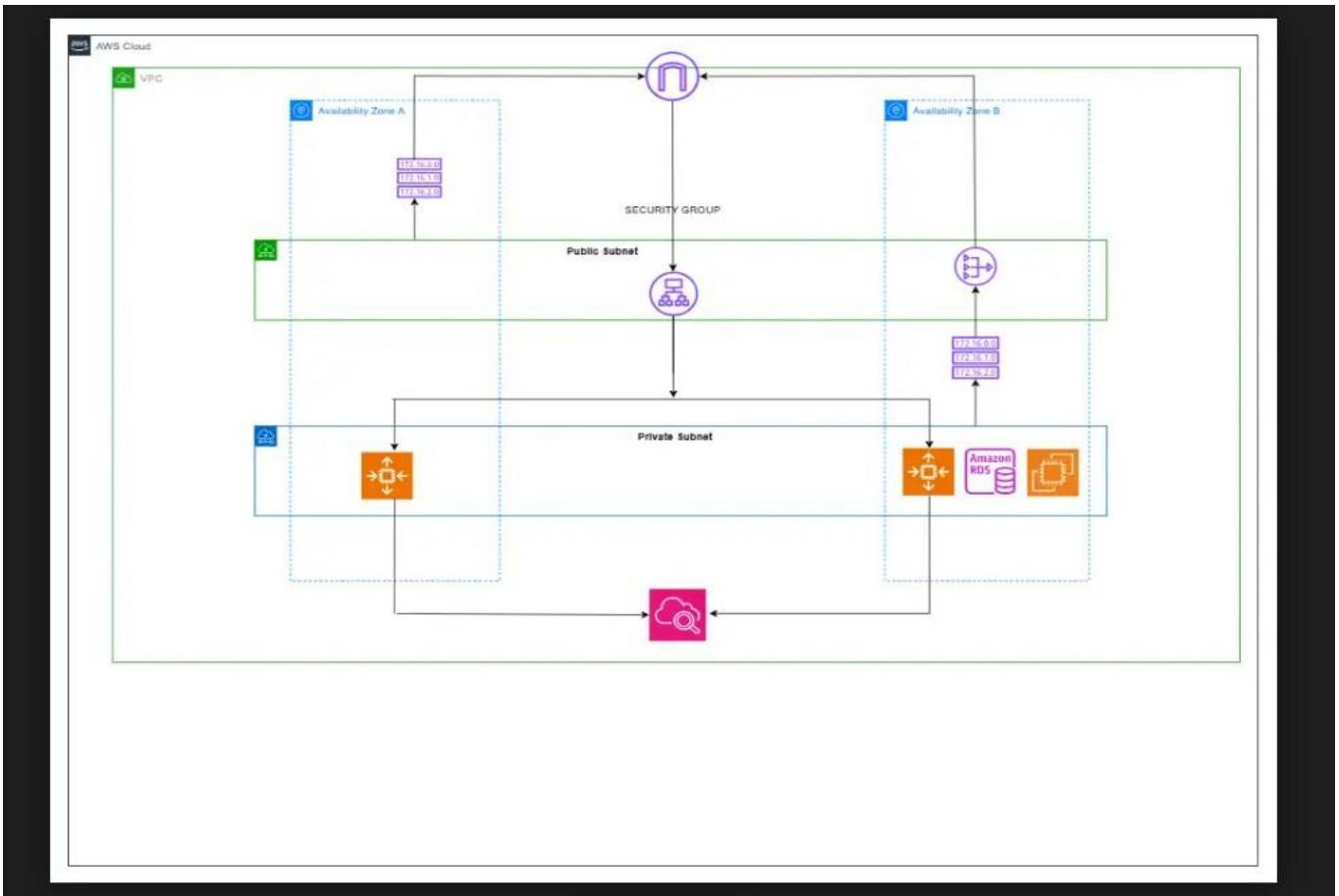
npm build

npm run
```

## Architecture Overview

## Flowchart

- **Screenshot:** flowchat.png
- **Description:** High-level visual representation of the AWS architecture.



## Summary

To ensure all ASG instances are provisioned with your app:

- Use a pre-baked AMI, or
- Use a proper user data script