Configure and Connect a MySQL Database Instance with a Web Server

Description:

You are working as a database administrator for an IT firm. You have been asked to create a new database instance on AWS cloud and connect it with the employee management portal hosted on a web server.

Problem statement:

Organization wants to deploy a new multi-tier application. The application will take live inputs from the employees and it will be hosted on a web server running on the AWS cloud.

The development team has asked you to set up the web server and configure it to scale automatically in cases of a traffic surge, to make the application highly available. They have also asked you to take the inputs from the employees and store them securely in the database.

Must use the following:

- Create a Database Instance with the following specifications:
    - Database creation method: Standard Create
    - Engine: MySQL
    - Database Instance size: db.t2.micro

- Create an EC2 Instance with the following specifications:
    - AMI: Amazon Linux
    - Region: Use only US East (N Virginia), us-east-1, and us-east-2
    - Instance types: t2.micro and t3.micro
    - Allowed EBS types: GP2 and Standard

Following requirements should be met:

- Follow the above-mentioned specifications
- Make sure that the Availability Zone is similar throughout the instances and volumes
- Ensure that the server scales automatically and the traffic is optimally routed among the scaled servers
- Document the step-by-step process involved in completing this task

Solution Overview and Configuration Steps

This multi-tier architecture consists of front end webserver and backend database. The frontend webserver will provide the application access to the users and take live inputs like Employee name and address from the user and insert the data into the backend data base.

For deploying the application, used AWS cloud services like Virtual private cloud, Internet Gateway, Security groups, subnets, EC2 Instances, RDS MySQL, Load balancer, Auto scaling group. This document will explain about how to create and configure all the required AWS services to design, deploy and test the application. Let's start with AWS services creation and configurations to achieve the final goal of this project.
The below diagram is high level architecture of the multi-tier application and we will follow this diagram for to creating services. I will explain more about each service information and configuration below

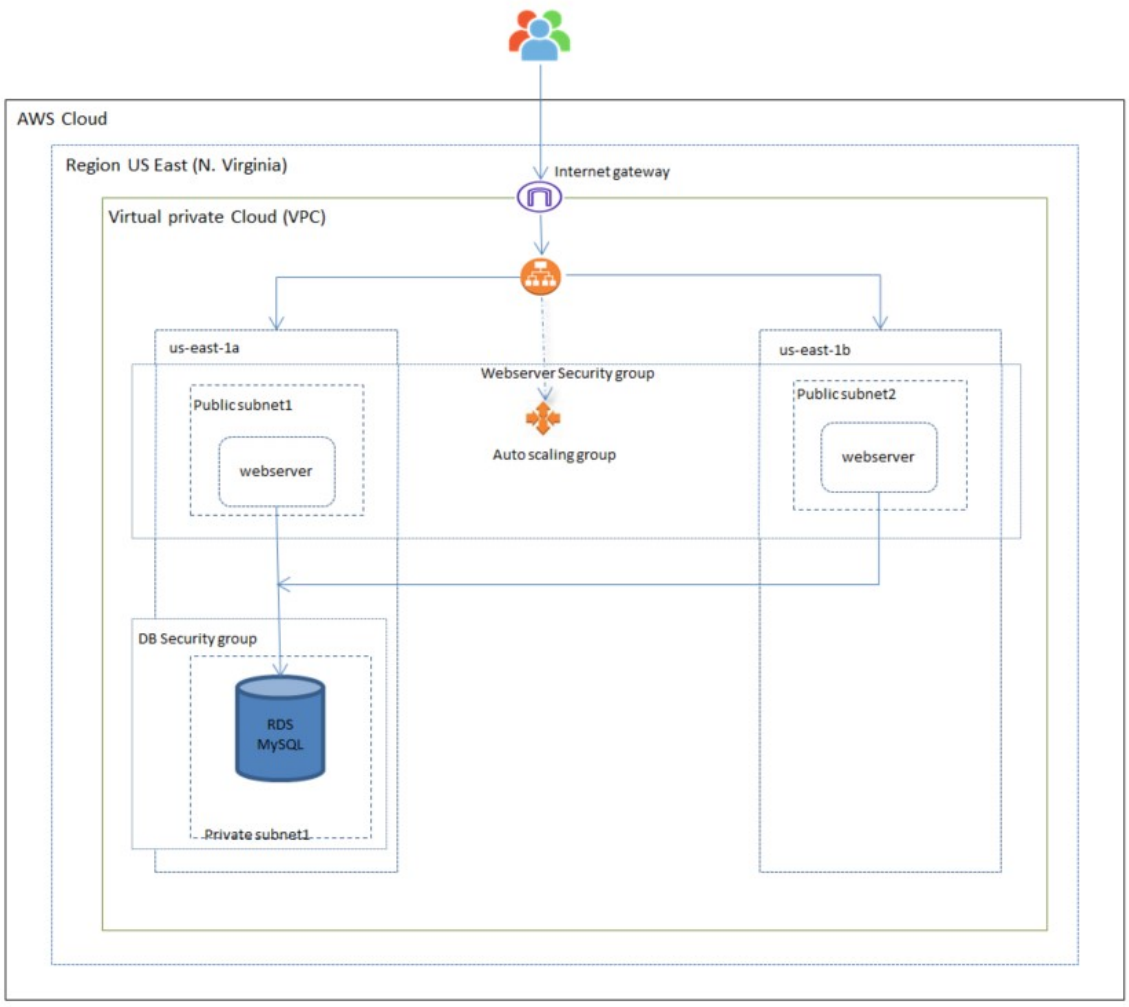


Diagram1: Application architecture

Create a Virtual private Cloud and all required components

- Looking at above architecture diagram, need to create VPC with below specified steps

- Logging into the AWS console, navigate to the VPC resource. Click “Create VPC” and you will see two options: “VPC only” or “VPC and more.” Select VPC and more because, we are creating a VPC with multiple subnets, availability zones, and more. So it’s will create internet Gateway, subnets, Route tables and attach internet Gateway to VPC.

  a) Logging into the AWS console

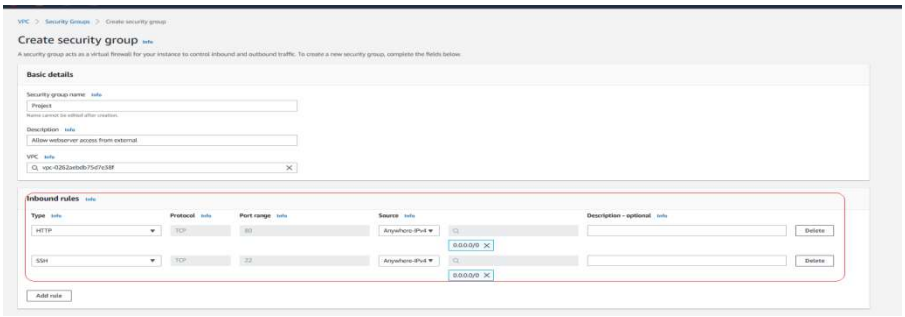
  b) Navigate to the VPC

  c) Click “Create VPC” and select “VPC and more” , during VPC creation selected the following inputs

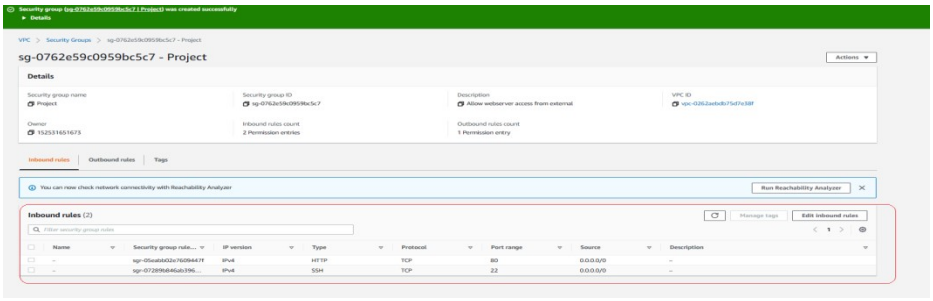
     IPv4 CIDR block: **10.0.0.0/16**
     Tenancy: **Default**
     Availability zones: **2 (us-east-1a & us-east-1b)**
     Number of public subnets: **2**
     Number of private subnets: **2**
     NAT gateways: **None**
     VPC endpoints: **None**
     Enable DNS hostnames: **check**
     Enable DNS resolutions: **check**

d) After all input parameters filled click create VPC and will create the VPC with below workflow

- Create Security groups for EC2 instance
    a) Navigated to Security group and click create security Group
    b) Provided Security group name as “project” and update description
    c) Select the VPC which was created as above step
    d) Add inbound rules with SSH and HTTP for webserver

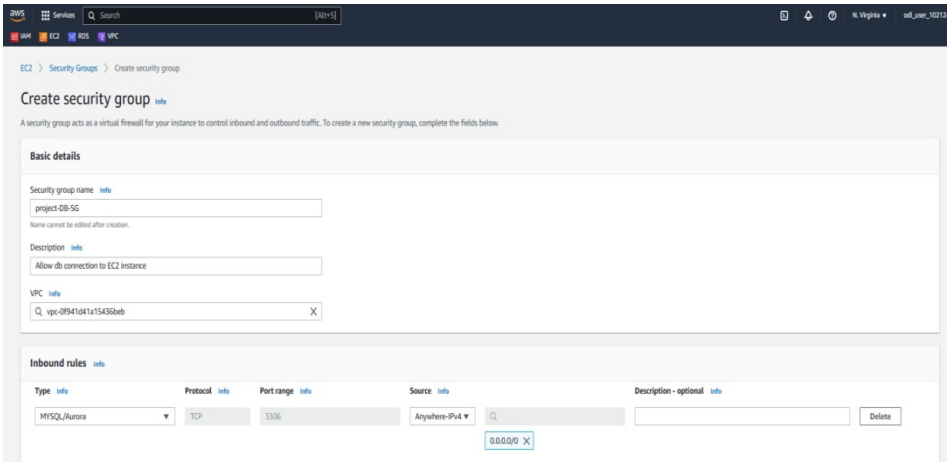


e) Click create Security Group

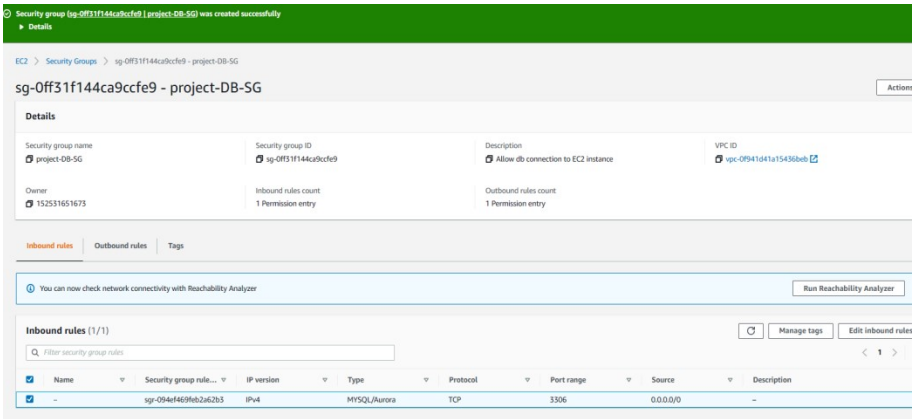


- Create Security group for RDS Database instance.

    a) Navigated to Security group and click create security Group
    b) Provided Security group name as “project-DB-SG” and update description
    c) Select the VPC which was created as above step
    d) Add inbound rules with SSH and HTTP for webserver



    e) Click create Security Group



Create a webserver Tier

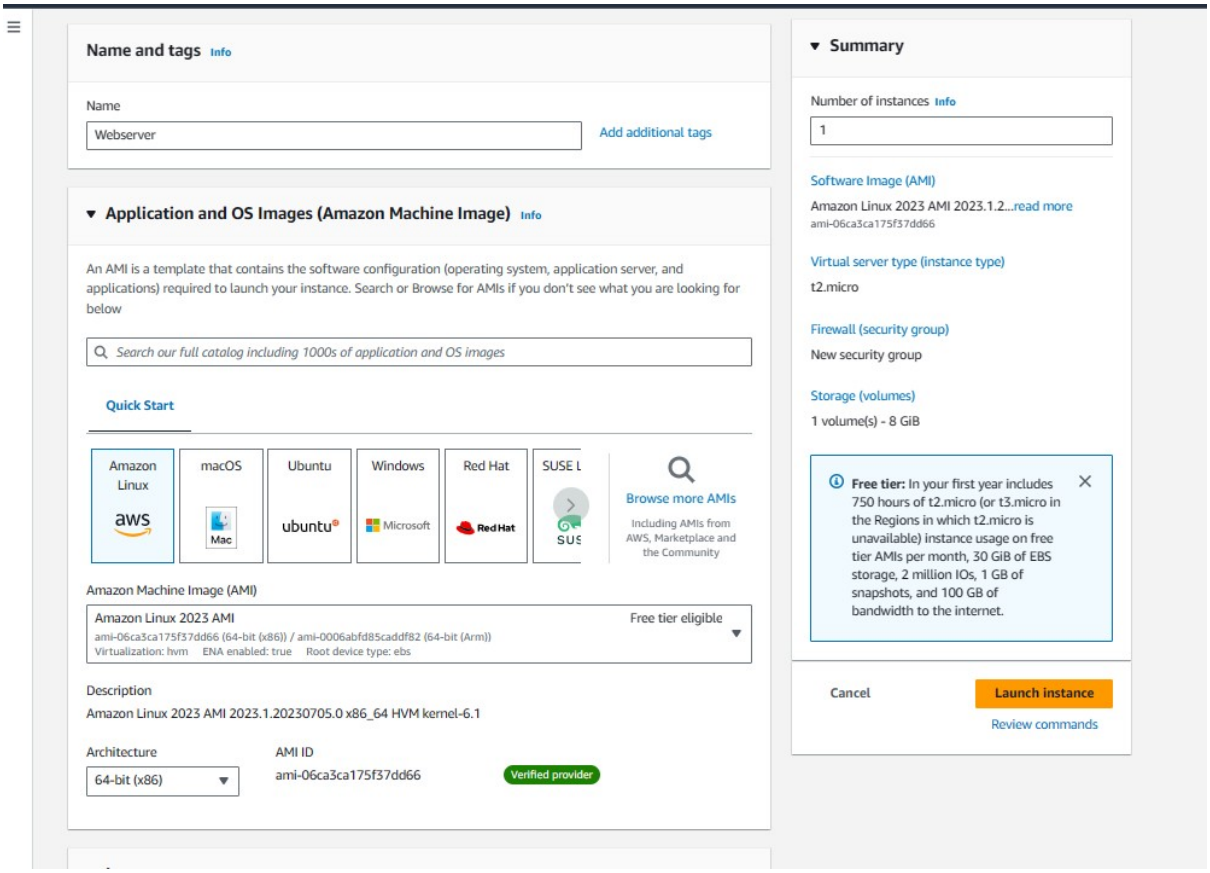
In this section created first tier EC2 instance and it represents the webserver front end application for user inputs.

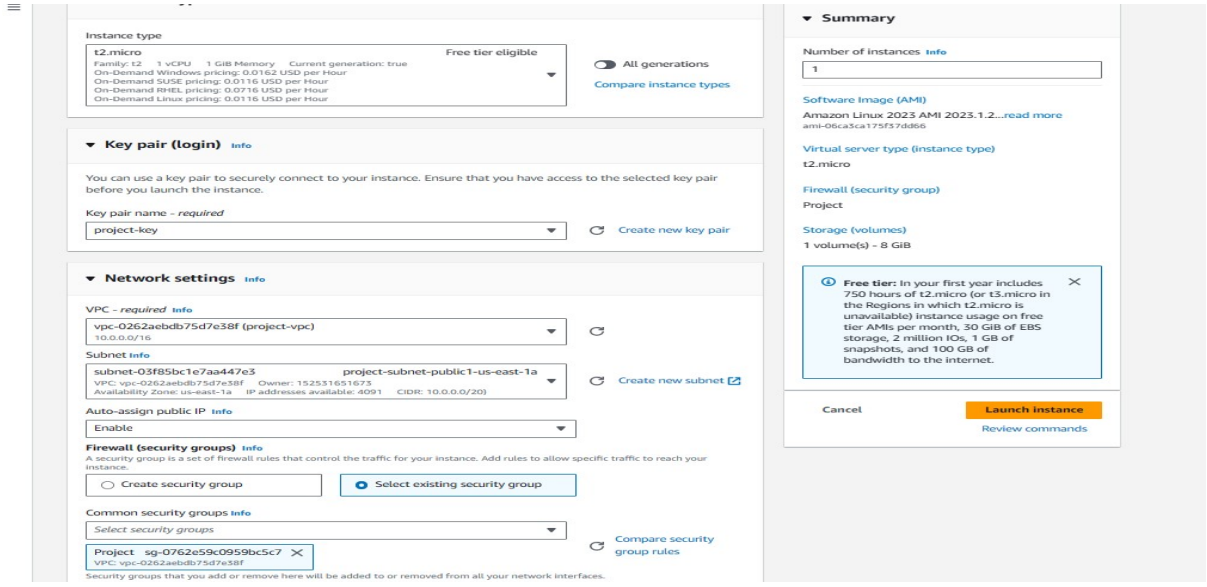
Launch an EC2 instance

Navigate to the EC2 Console and Click launch instance

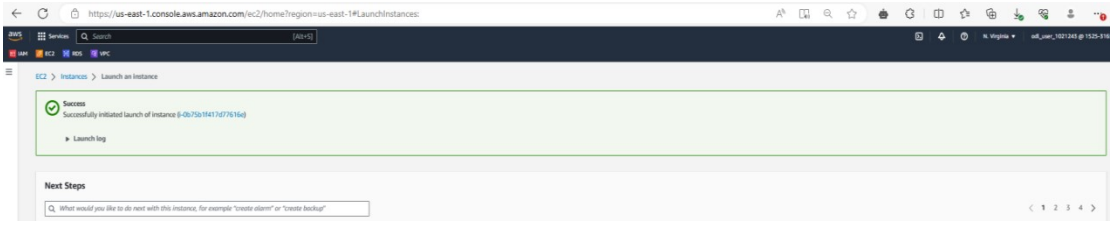
During EC2 instance selected the following inputs

Updated name as “webserver”
Selected AMI “Amazon Linux 2023 AMI”
Instance type t2.micro
Created key pair with name project-key
Selected VPC “Project-vpc” which was created above
Selected subnet ua-east-1a public subnet
Selected auto assign public IP enable
Selected existing service group which was created above for EC2 instance called “project”
Create EC2 instance



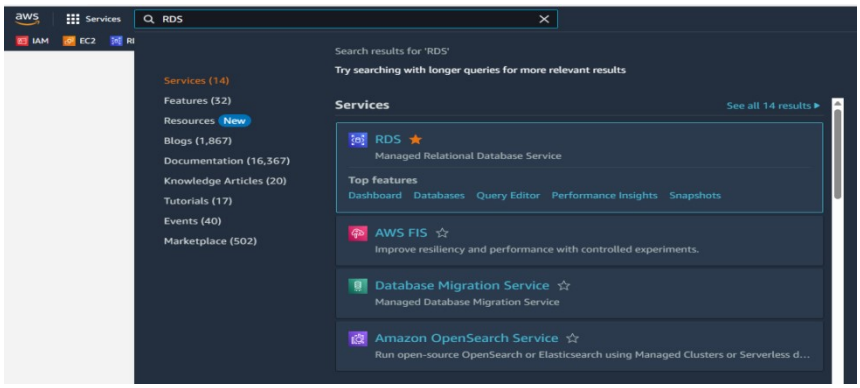


Now the EC2 Instance created successfully.

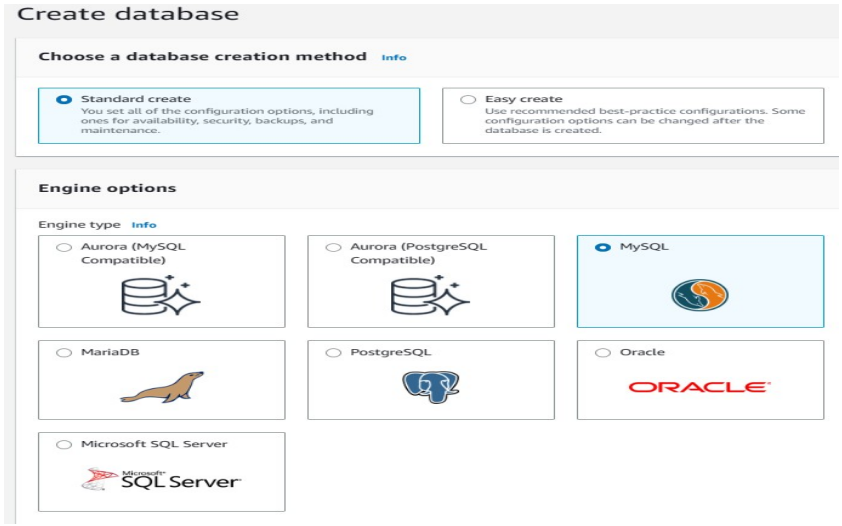


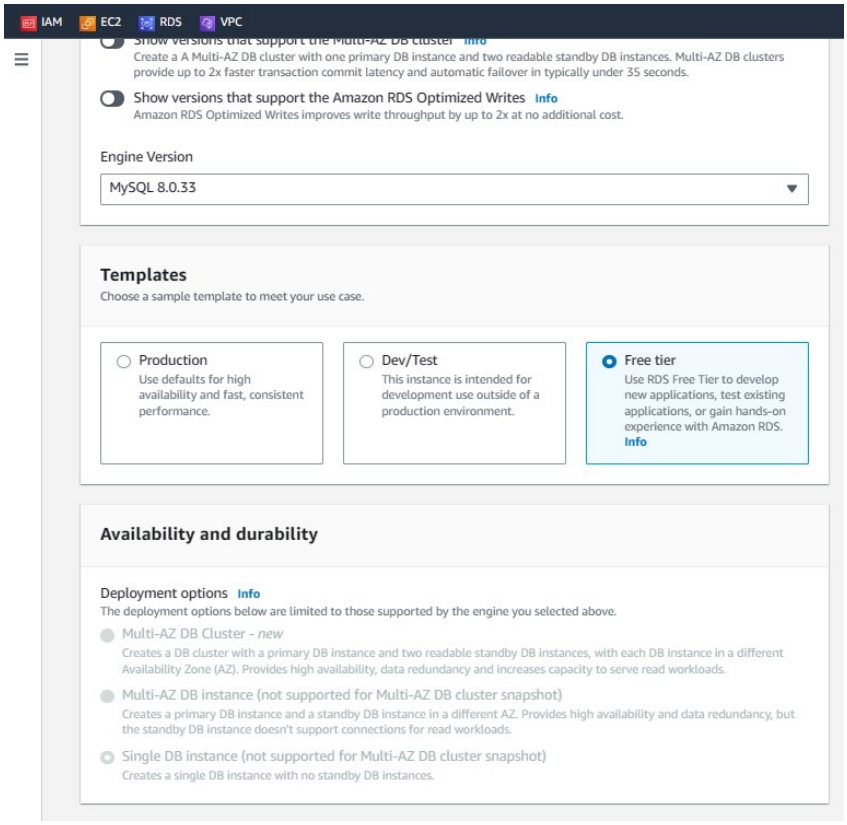
Create RDS MySQL Instance

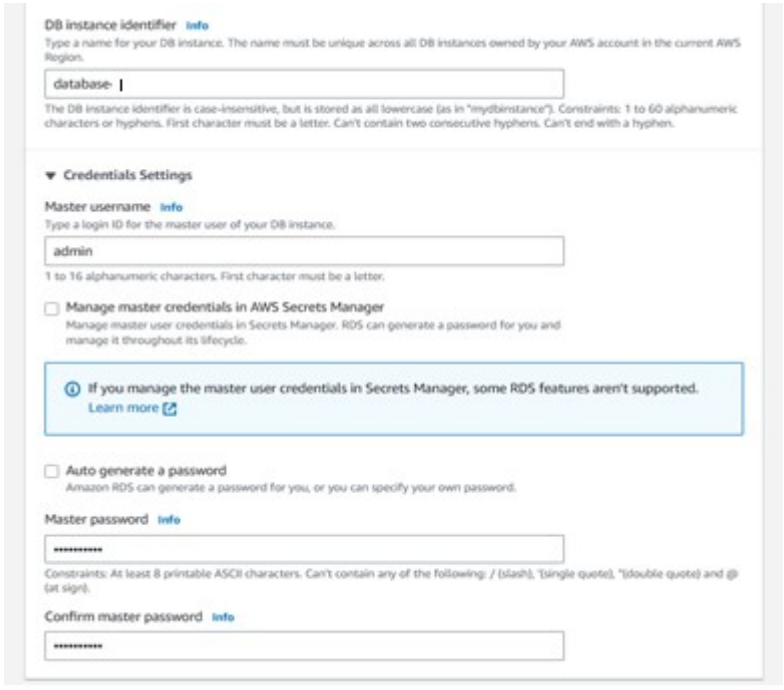
Navigate to RDS and click RDS



Selected “Standard Create”
Selected Engine options MySQL
Selected Free tier and DB instance name database-1
Kept the username as admin and provided Master password for connect database
Selected instance class t2.micro
Selected max storage threshold value 100
Selected connectivity to an EC2 compute resources and selected the EC2 instance name which was created above.
Selected Security group “project-db-sg” which was created for DB server instance
Click create Database




IAM
EC2
RDS
VPC
Create a A Multi-AZ DB cluster with one primary DB instance and two readable standby DB instances. Multi-AZ DB clusters provide up to 2x faster transaction commit latency and automatic failover in typically under 35 seconds.
Show versions that support the Amazon RDS Optimized Writes Info
Amazon RDS Optimized Writes improves write throughput by up to 2x at no additional cost.
Engine Version
MySQL 8.0.33
Templates
Choose a sample template to meet your use case.
Production
Use defaults for high availability and fast, consistent performance.
Dev/Test
This instance is intended for development use outside of a production environment.
Free tier
Use RDS Free Tier to develop new applications, test existing applications, or gain hands-on experience with Amazon RDS. Info
Availability and durability
Deployment options Info
The deployment options below are limited to those supported by the engine you selected above.
Multi-AZ DB Cluster - new
Creates a DB cluster with a primary DB instance and two readable standby DB instances, with each DB instance in a different Availability Zone (AZ). Provides high availability, data redundancy and increases capacity to serve read workloads.
Multi-AZ DB instance (not supported for Multi-AZ DB cluster snapshot)
Creates a primary DB instance and a standby DB instance in a different AZ. Provides high availability and data redundancy, but the standby DB instance doesn't support connections for read workloads.
Single DB instance (not supported for Multi-AZ DB cluster snapshot)
Creates a single DB instance with no standby DB instances.


DB instance identifier Info
Type a name for your DB instance. The name must be unique across all DB instances owned by your AWS account in the current AWS Region.
database-
The DB instance identifier is case-insensitive, but is stored as all lowercase (as in "mydbinstance"). Constraints: 1 to 60 alphanumeric characters or hyphens. First character must be a letter. Can't contain two consecutive hyphens. Can't end with a hyphen.
Credentials Settings
Master username Info
Type a login ID for the master user of your DB instance.
admin
1 to 16 alphanumeric characters. First character must be a letter.
Manage master credentials in AWS Secrets Manager
Manage master user credentials in Secrets Manager. RDS can generate a password for you and manage it throughout its lifecycle.
If you manage the master user credentials in Secrets Manager, some RDS features aren't supported.
Learn more
Auto generate a password
Amazon RDS can generate a password for you, or you can specify your own password.
Master password Info
Constraints: At least 8 printable ASCII characters. Can't contain any of the following: / (slash), '(single quote), "(double quote) and @ (at sign).
Confirm master password Info

Instance configuration

The DB instance configuration options below are limited to those supported by the engine that you selected above.

Amazon RDS Optimized Writes - *new* Info

Show instance classes that support Amazon RDS Optimized Writes

DB instance class Info

- Standard classes (includes m classes)
- Memory optimized classes (includes r and x classes)
- Burstable classes (includes t classes)

db.t2.micro
1 vCPUs 1 GiB RAM Not EBS Optimized

Include previous generation classes

Storage

Storage type Info

General Purpose SSD (gp2)
Baseline performance determined by volume size

Maximum storage threshold Info

Charges will apply when your database autoscales to the specified threshold

100 GiB

The minimum value is 22 GiB and the maximum value is 6,144 GiB

Connectivity Info

Compute resource

Choose whether to set up a connection to a compute resource for this database. Setting up a connection will automatically change connectivity settings so that the compute resource can connect to this database.

- Don't connect to an EC2 compute resource
  Don't set up a connection to a compute resource for this database. You can manually set up a connection to a compute resource later.
- Connect to an EC2 compute resource
  Set up a connection to an EC2 compute resource for this database.

EC2 instance Info

Choose the EC2 instance to add as the compute resource for this database. A VPC security group is added to this EC2 instance. A VPC security group is also added to the database with an inbound rule that allows the EC2 instance to access the database.

i-0b75b1f417d77616e
Webserver

Some VPC settings can't be changed when a compute resource is added

Adding an EC2 compute resource automatically selects the VPC, DB subnet group, and public access settings for this database. To allow the EC2 instance to access the database, a VPC security group rds-ec2-X is added to the database and another called ec2-rds-X to the EC2 instance. You can remove the new security group for the database only by removing the compute resource.

Virtual private cloud (VPC) Info

Choose the VPC. The VPC defines the virtual networking environment for this DB instance.

project-vpc (vpc-0262aebdb75d7e38f)
4 Subnets, 2 Availability Zones

Only VPCs with a corresponding DB subnet group are listed.

Choose the VPC. The VPC defines the virtual networking environment for this DB instance.

project-vpc (vpc-0262aebdb75d7e38f)
4 Subnets, 2 Availability Zones

Only VPCs with a corresponding DB subnet group are listed.

After a database is created, you can't change its VPC.

DB subnet group Info

Choose the DB subnet group. The DB subnet group defines which subnets and IP ranges the DB instance can use in the VPC that you selected.

- Choose existing
  Choose existing DB subnet group
- Automatic setup
  RDS creates a new subnet group for you or reuses an existing subnet group

DB subnet group name

rds-ec2-db-subnet-group-1

New DB subnet group created.

Public access Info

- Yes
  RDS assigns a public IP address to the database. Amazon EC2 instances and other resources outside of the VPC can connect to your database. Resources inside the VPC can also connect to the database. Choose one or more VPC security groups that specify which resources can connect to the database.
- No
  RDS doesn't assign a public IP address to the database. Only Amazon EC2 instances and other resources inside the VPC can connect to your database. Choose one or more VPC security groups that specify which resources can connect to the database.

VPC security group (firewall) Info

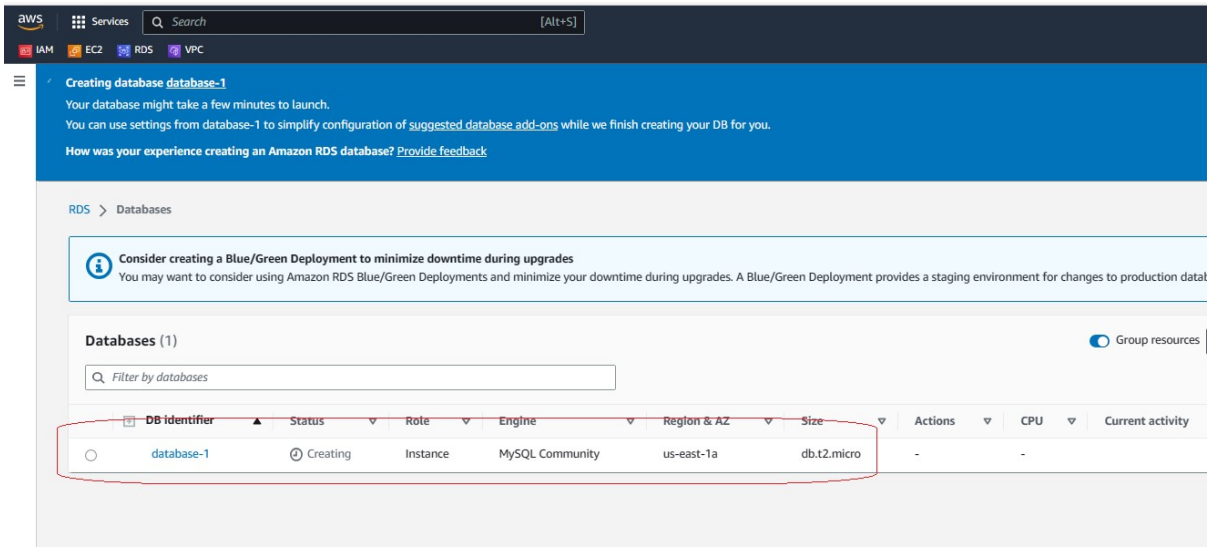
Choose one or more VPC security groups to allow access to your database. Make sure that the security group rules allow the appropriate incoming traffic.

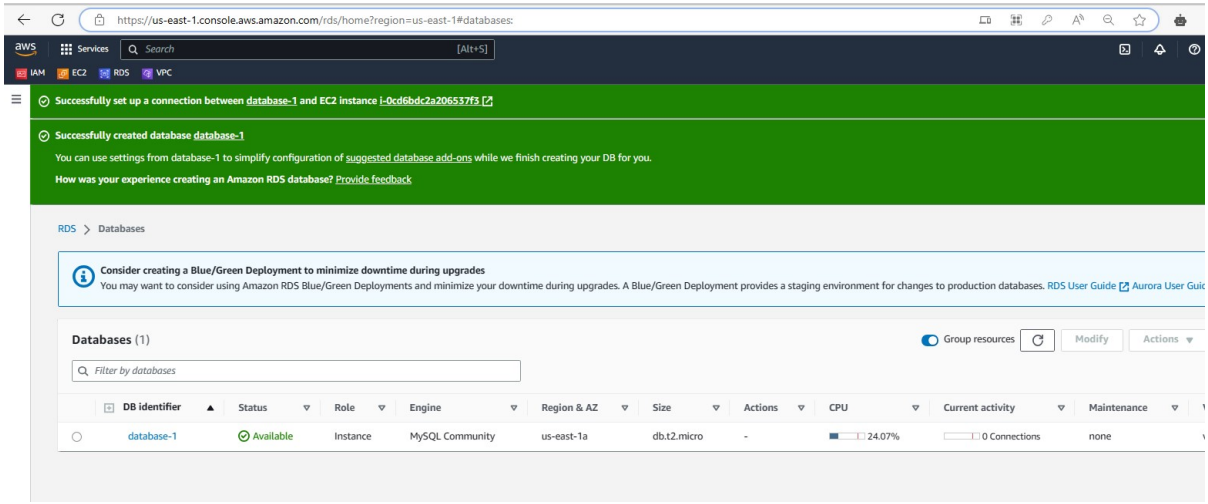
- Choose existing
  Choose existing VPC security groups
- Create new
  Create new VPC security group

Additional VPC security group

Choose one or more options

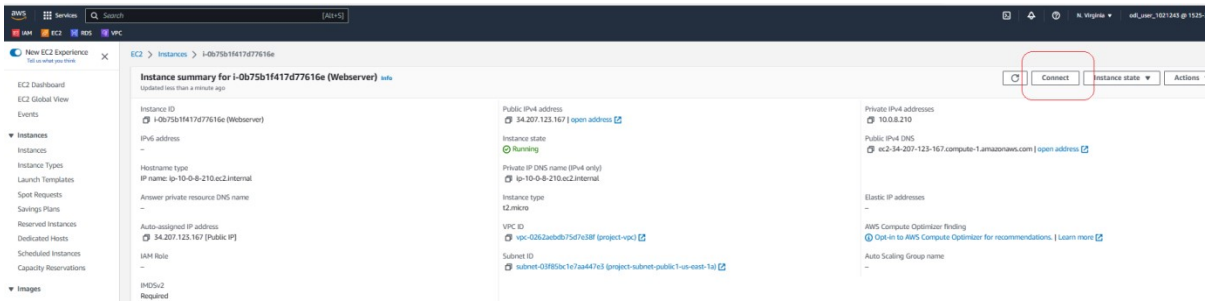
project-DB-SG ×

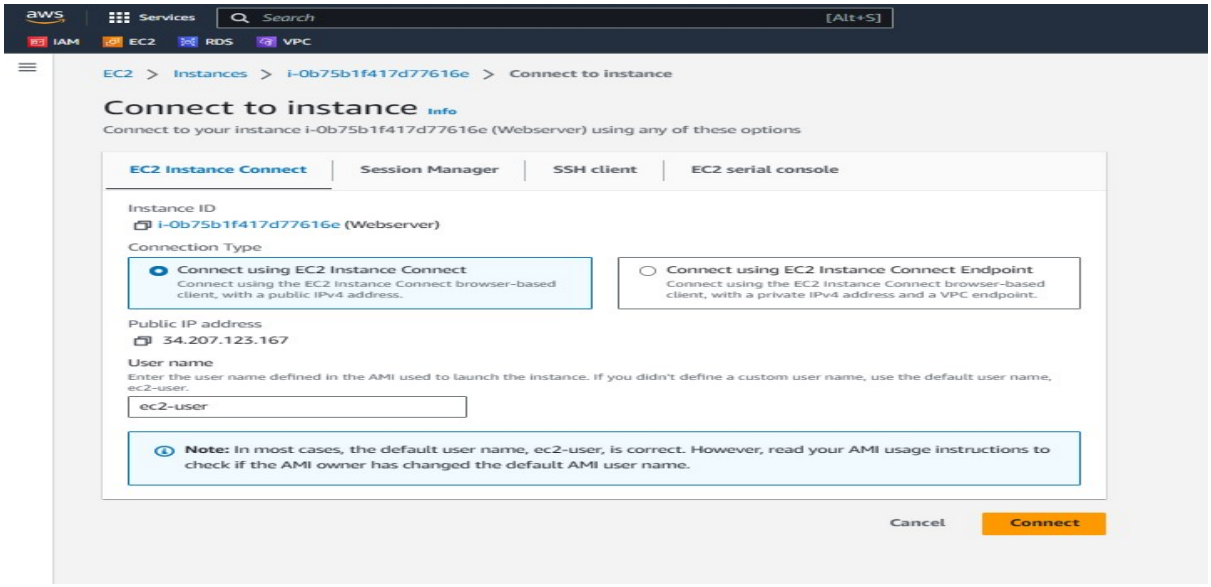




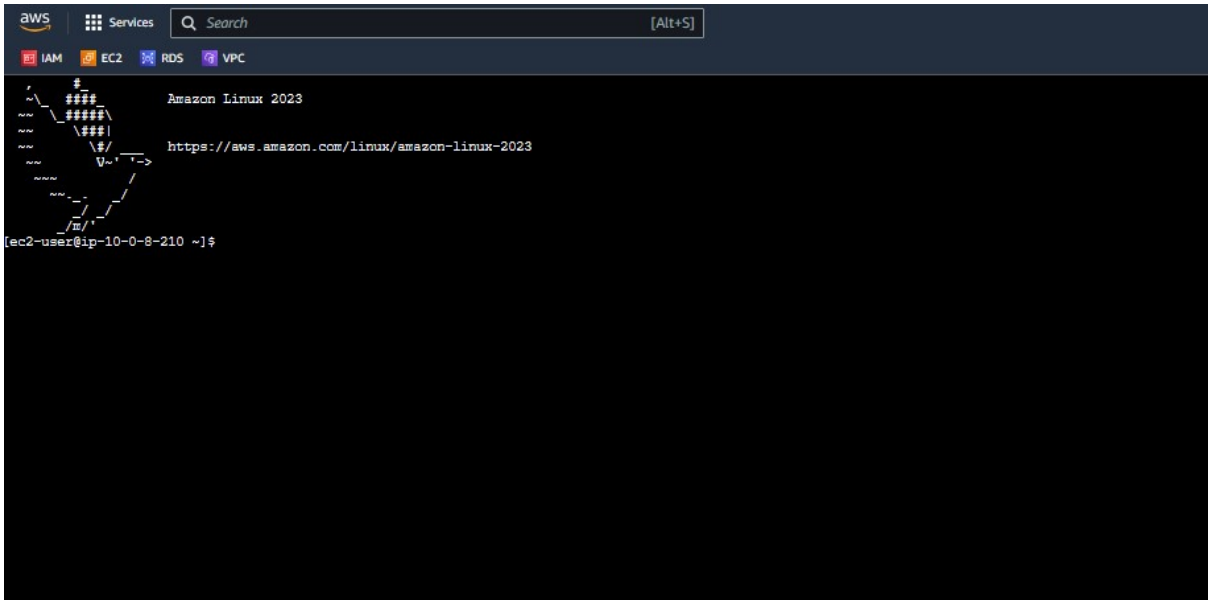
Install an Apache webserver with PHP and MariaDB

Connect to Webserver EC2 instance which was created above.
Navigate to EC2 instance and select the webserver EC2 instance and click connect
Connect to EC2 instance connect and click connect



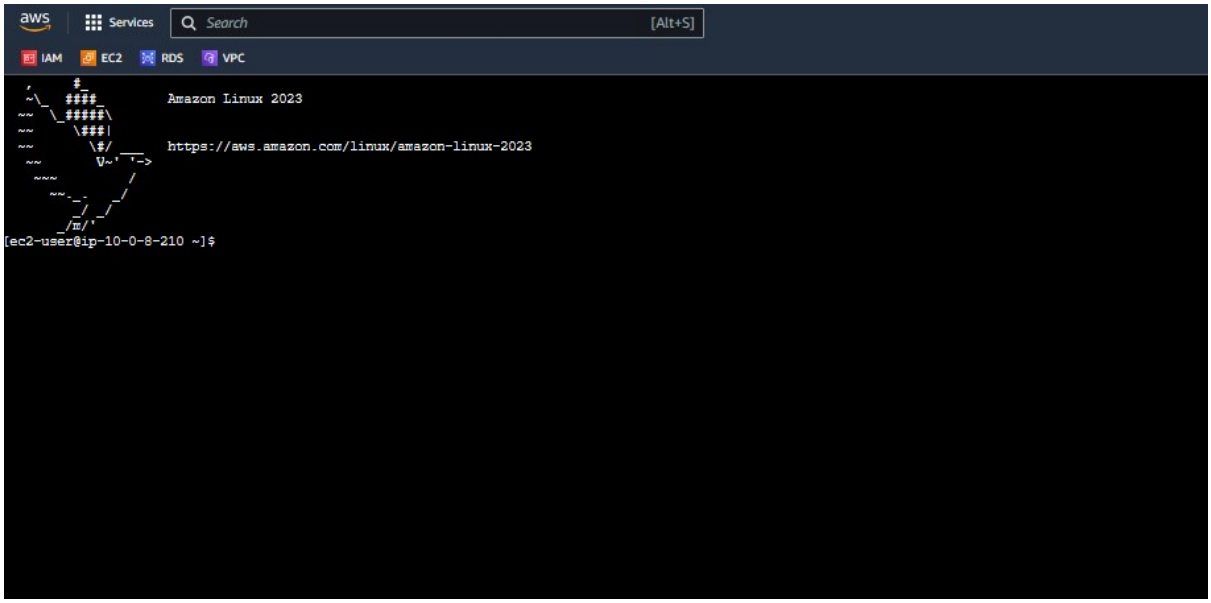


Connected EC2 instance console

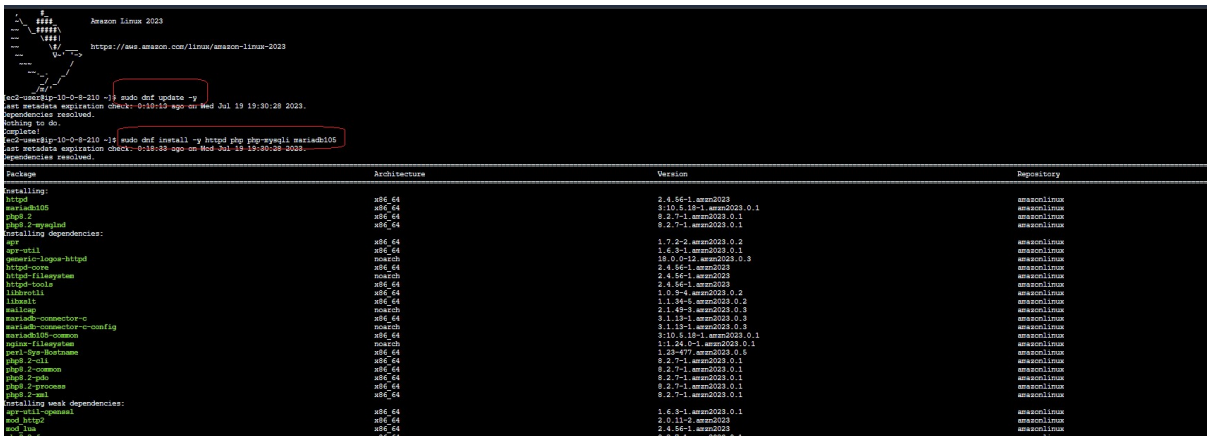


Install an Apache web server with PHP and MariaDB

Connect EC2 instance using



Get the latest bug fixes and security updates by updating the software on your EC2 instance
sudo dnf update -y
install the Apache web server, PHP, and MariaDB software using the following command.
sudo dnf install -y httpd php php-mysqli mariadb105



Start the web server with the command shown following.
Configure the web server to start with each system boot using the `systemctl` command.

```
$sudo systemctl start httpd
```

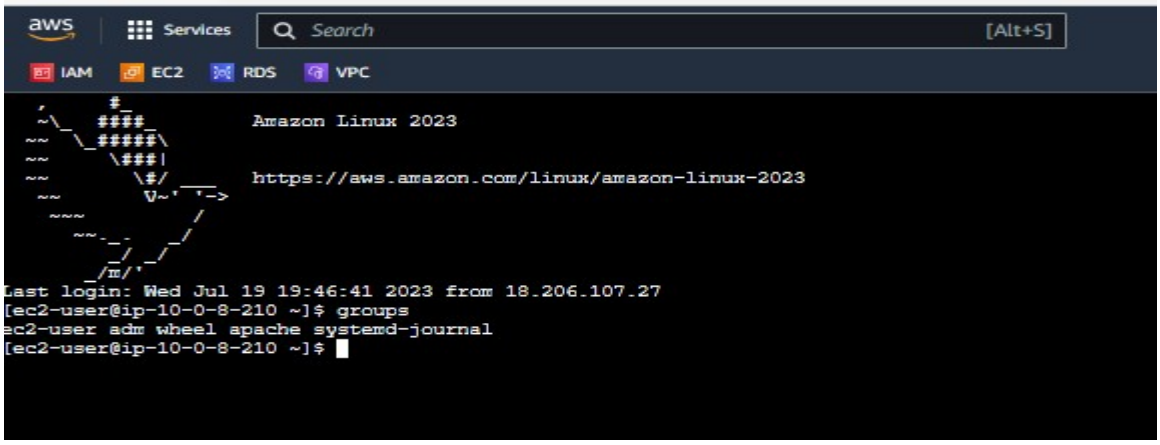
$sudo systemctl enable httpd

```
[ec2-user@ip-10-0-8-210 ~]$ sudo systemctl start httpd
[ec2-user@ip-10-0-8-210 ~]$ sudo systemctl enable httpd
Created symlink /etc/systemd/system/multi-user.target.wants/httpd.service → /usr/lib/systemd/system/httpd.service.
[ec2-user@ip-10-0-8-210 ~]$
```

To set file permissions for the Apache web server
Add the `ec2-user` user to the `apache` group.
$sudo usermod -a -G apache ec2-user

```
[ec2-user@ip-10-0-8-210 ~]$ sudo usermod -a -G apache ec2-user
[ec2-user@ip-10-0-8-210 ~]$
```

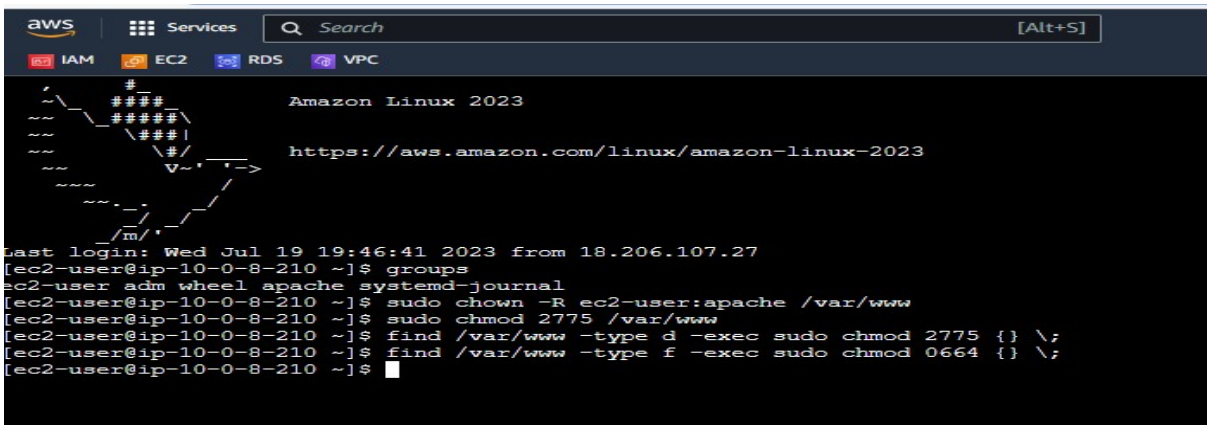
Exit and Log back in again and verify that the `apache` group exists with the `groups` command.



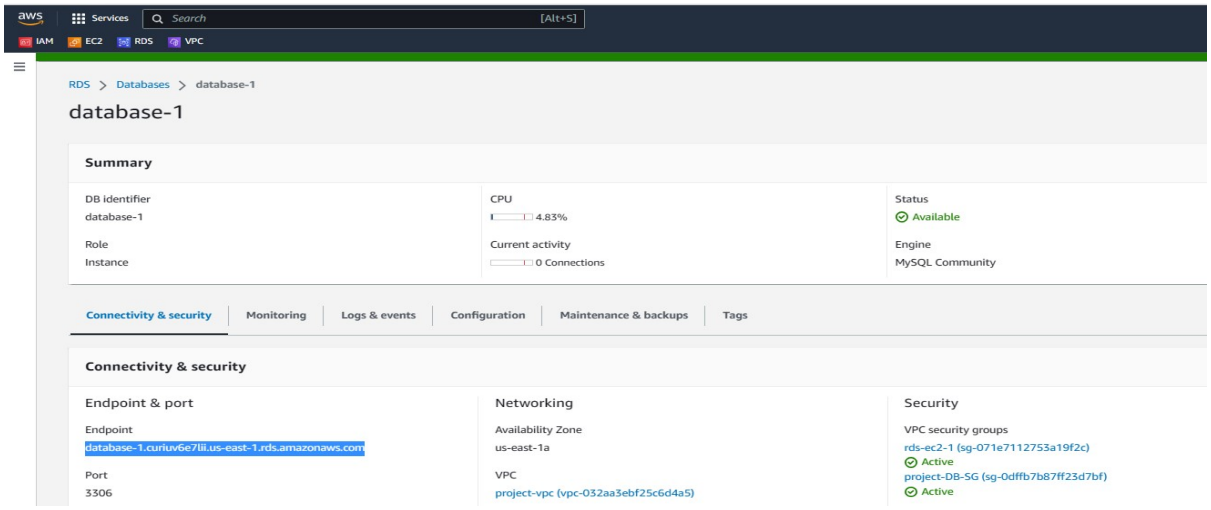
Change the group ownership of the `/var/www` directory and its contents to the `apache` group.
$sudo chown -R ec2-user:apache /var/www

Change the directory permissions of `/var/www` and its subdirectories to add group write permissions and set the group ID on subdirectories created in the future.
$sudo chmod 2775 /var/www
$find /var/www -type d -exec sudo chmod 2775 {} \;

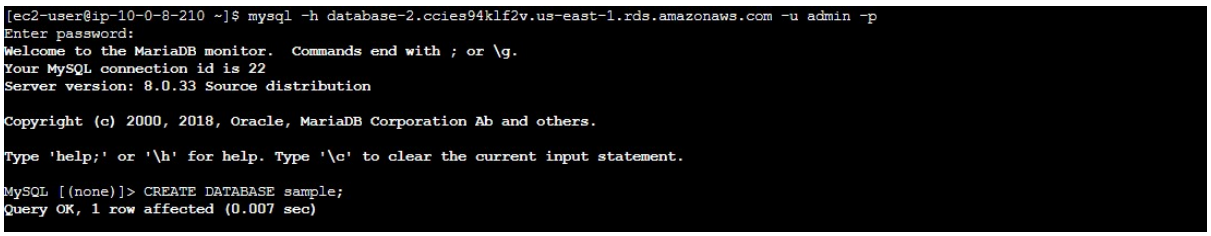
Recursively change the permissions for files in the `/var/www` directory and its subdirectories to add group write permissions.
$find /var/www -type f -exec sudo chmod 0664 {} \;



Connected Apache web server to DB instance
Check the DB endpoint from database instance



Create DATABASE in MYSQL DB



```
MySQL [(none)]> SHOW DATABASES;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| mysql              |
| performance_schema |
| sample             |
| sys                |
+--------------------+
5 rows in set (0.006 sec)

MySQL [(none)]>
```

To add content to the Apache web server that connects to your DB instance

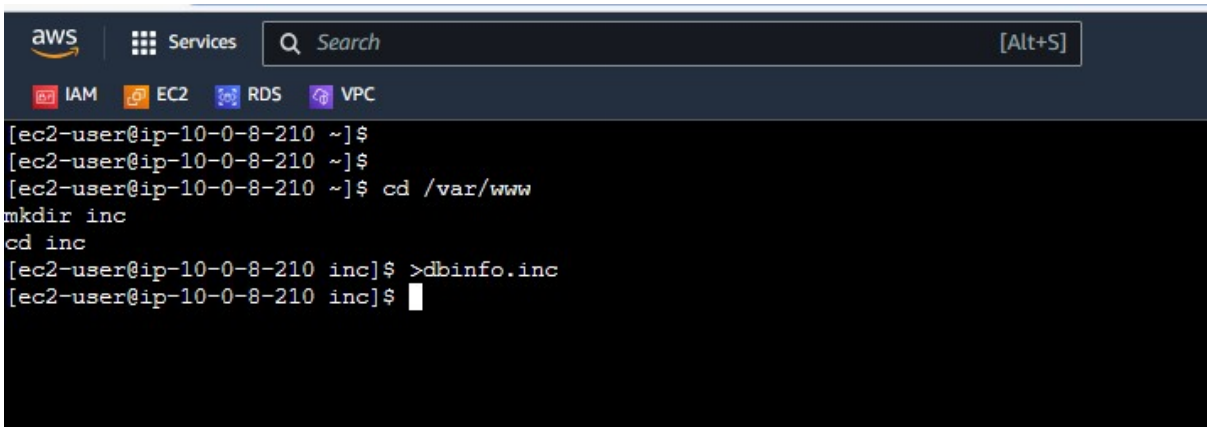
$cd /var/www

$mkdir inc

$cd inc

Created a new file in the `inc` directory named `dbinfo.inc`, and then edit the file by calling nano (or the editor of your choice).

```
>dbinfo.inc
```



Added the following contents to the `dbinfo.inc` file. Here, *`db_instance_endpoint`* is your DB instance endpoint, without the port, for your DB instance.

```
<?php

define('DB_SERVER', 'db_instance_endpoint');
define('DB_USERNAME', 'tutorial_user');
define('DB_PASSWORD', 'master password');
define('DB_DATABASE', 'sample');

?>
```

```
[ec2-user@ip-10-0-10-143 inc]$ cat dbinfo.inc
<?php

define('DB_SERVER', 'database-1.curiuv6e7lii.us-east-1.rds.amazonaws.com');
define('DB_USERNAME', 'admin');
define('DB_PASSWORD', '[redacted]');
define('DB_DATABASE', 'sample');

?>
```

Save and close the `dbinfo.inc` file.
Change the directory to `/var/www/html`.
Add the following code in SamplePage.php

```
<?php include "../inc/dbinfo.inc"; ?>
<html>
<body>
<h1>Employees Data</h1>
<?php

 /* Connect to MySQL and select the database. */
 $connection = mysqli_connect(DB_SERVER, DB_USERNAME, DB_PASSWORD);

 if (mysqli_connect_errno()) echo "Failed to connect to MySQL: " . mysqli_connect_error();

 $database = mysqli_select_db($connection, DB_DATABASE);

 /* Ensure that the EMPLOYEES table exists. */
 VerifyEmployeesTable($connection, DB_DATABASE);

 /* If input fields are populated, add a row to the EMPLOYEES table. */
 $employee_name = htmlentities($_POST['NAME']);
 $employee_address = htmlentities($_POST['ADDRESS']);

 if (strlen($employee_name) || strlen($employee_address)) {
   AddEmployee($connection, $employee_name, $employee_address);
 }
?>

<!-- Input form -->
<form action="<?PHP echo $_SERVER['SCRIPT_NAME'] ?>" method="POST">
 <table border="0">
   <tr>
     <td>NAME</td>
     <td>ADDRESS</td>
   </tr>
   <tr>
```

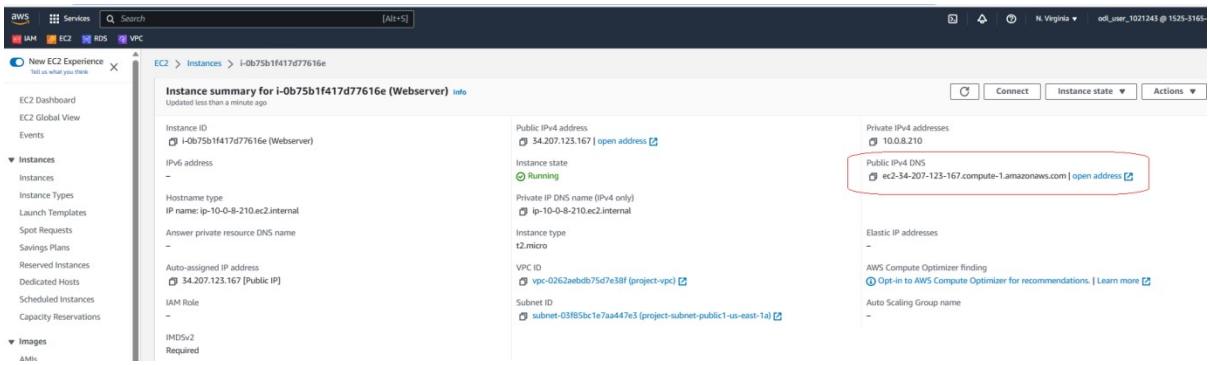
```
    <td>
     <input type="text" name="NAME" maxlength="45" size="30" />
    </td>
    <td>
     <input type="text" name="ADDRESS" maxlength="90" size="60" />
    </td>
    <td>
     <input type="submit" value="Add Data" />
    </td>
   </tr>
  </table>
</form>

<!-- Display table data. -->
<table border="1" cellpadding="2" cellspacing="2">
  <tr>
   <td>ID</td>
   <td>NAME</td>
   <td>ADDRESS</td>
  </tr>

<?php

$result = mysqli_query($connection, "SELECT * FROM EMPLOYEES");

while($query_data = mysqli_fetch_row($result)) {
  echo "<tr>";
  echo "<td>",$query_data[0], "</td>",
       "<td>",$query_data[1], "</td>",
       "<td>",$query_data[2], "</td>";
  echo "</tr>";
}
?>

</table>

<!-- Clean up. -->
<?php

  mysqli_free_result($result);
  mysqli_close($connection);

?>

</body>
</html>


<?php

/* Add an employee to the table. */
```

```
function AddEmployee($connection, $name, $address) {
  $n = mysqli_real_escape_string($connection, $name);
  $a = mysqli_real_escape_string($connection, $address);

  $query = "INSERT INTO EMPLOYEES (NAME, ADDRESS) VALUES ('$n', '$a');";

  if(!mysqli_query($connection, $query)) echo("<p>Error adding employee data.</p>");
}

/* Check whether the table exists and, if not, create it. */
function VerifyEmployeesTable($connection, $dbName) {
 if(!TableExists("EMPLOYEES", $connection, $dbName))
 {
   $query = "CREATE TABLE EMPLOYEES (
      ID int(11) UNSIGNED AUTO_INCREMENT PRIMARY KEY,
      NAME VARCHAR(45),
      ADDRESS VARCHAR(90)
    )";

   if(!mysqli_query($connection, $query)) echo("<p>Error creating table.</p>");
 }
}

/* Check for the existence of a table. */
function TableExists($tableName, $connection, $dbName) {
 $t = mysqli_real_escape_string($connection, $tableName);
 $d = mysqli_real_escape_string($connection, $dbName);

 $checktable = mysqli_query($connection,
    "SELECT TABLE_NAME FROM information_schema.TABLES WHERE TABLE_NAME = '$t' AND
TABLE_SCHEMA = '$d'");

 if(mysqli_num_rows($checktable) > 0) return true;

 return false;
}
?>
```

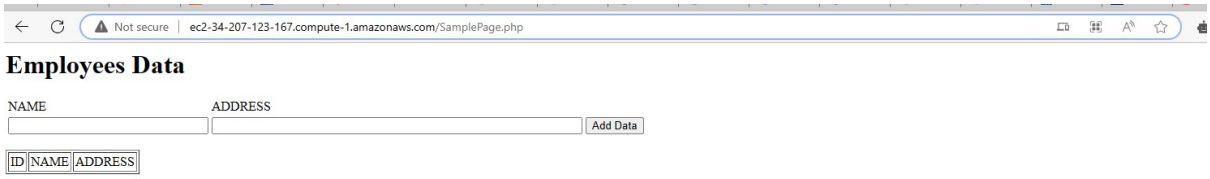
```
[ec2-user@ip-10-0-8-210 inc]$ cd /var/www/html
[ec2-user@ip-10-0-8-210 html]$ >SamplePage.php
[ec2-user@ip-10-0-8-210 html]$ vi SamplePage.php
[ec2-user@ip-10-0-8-210 html]$
```

Verify that your web server successfully connects to your DB instance by opening a web browser and browsing

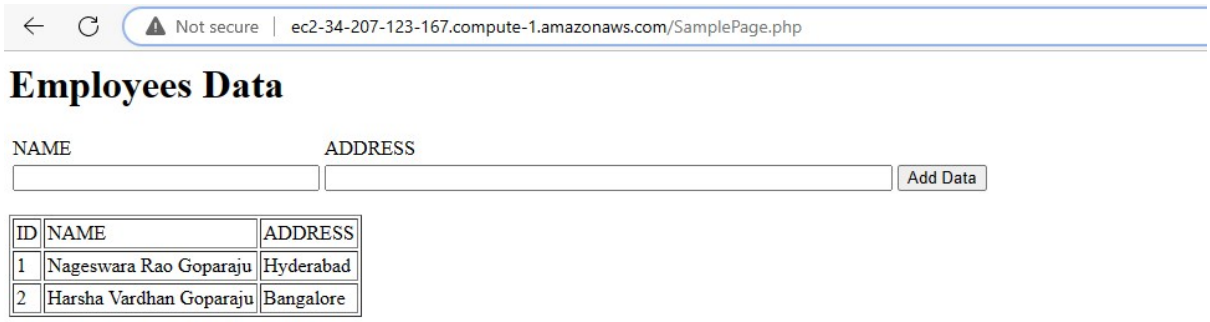
Get the public IP4 DNS name from the webserver EC2 instance.



Copy the webserver Instance public IP4 DNS name and try to open the URL from web browser. Web browser automatically redirected to the secure https site but changed the URL to use http. Success! I am able to see the webpage with input variables and table

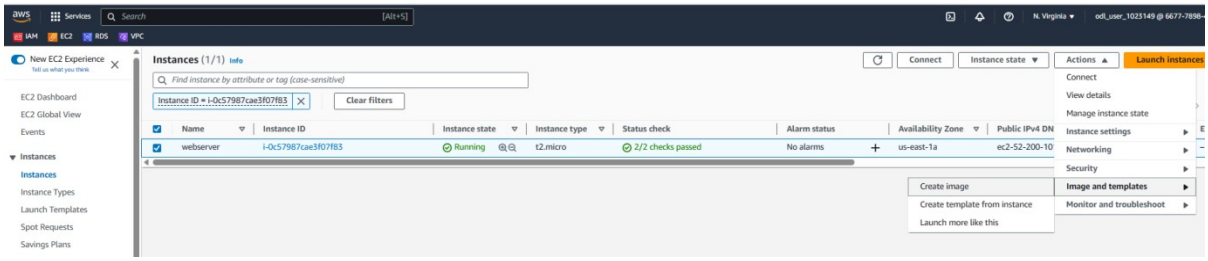


Now adding data into the database using input values from the webserver. Successfully able to add data into the database and table updated with the updated information

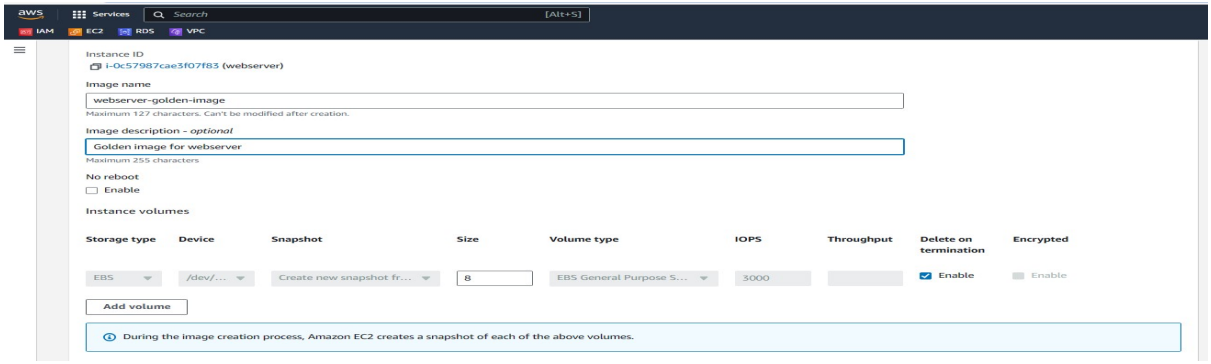


Creating Golden Image AMI and Instance template

Now created a Golden image of the webserver and will be used for Instance template.
Go to EC2 instances and select the webserver and click Actions→ image and templates→Create Image

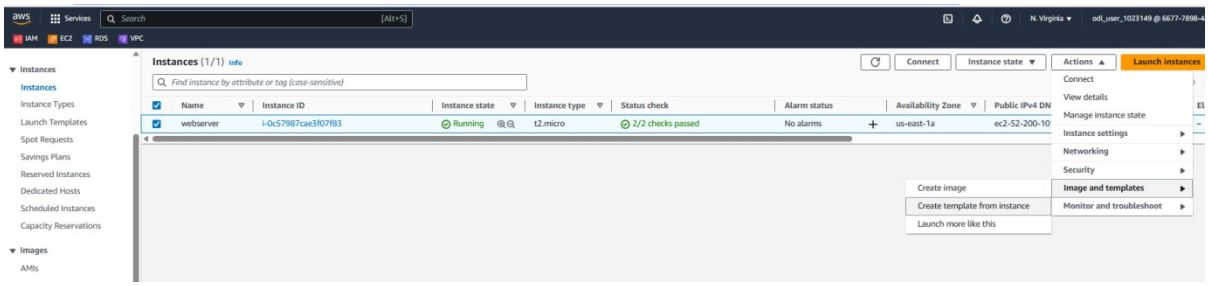


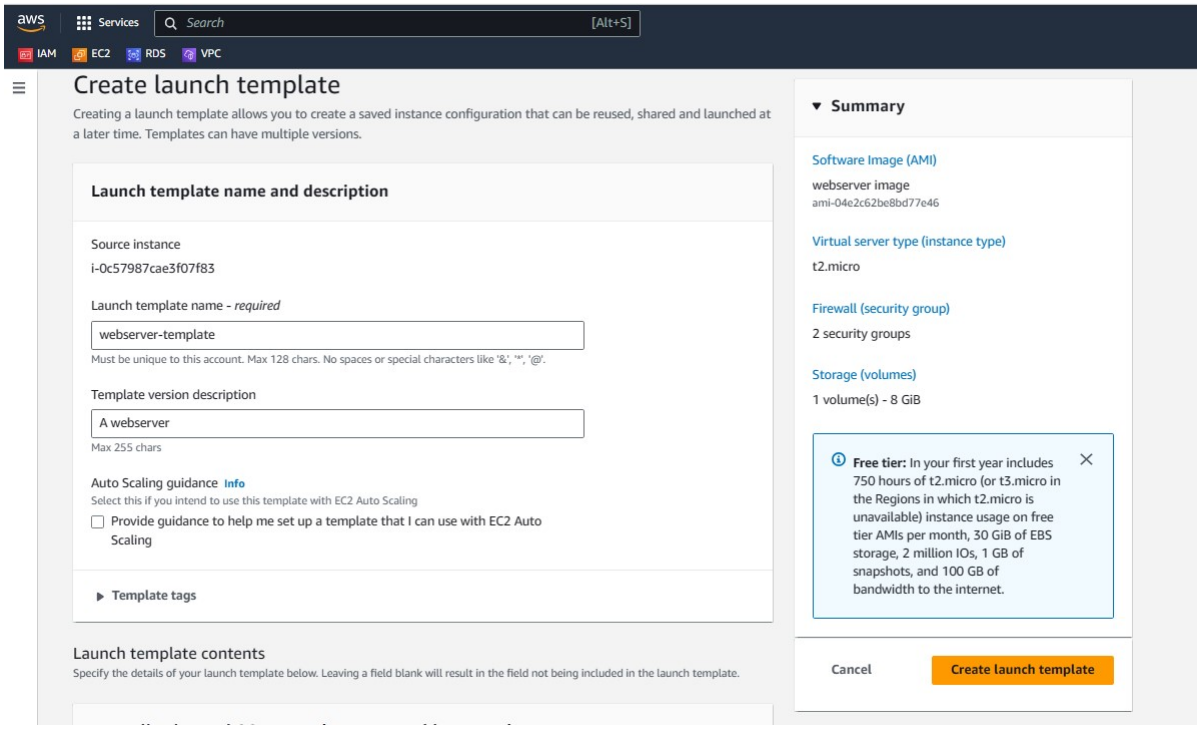
Provided Image name as webserver-golden-image and click image

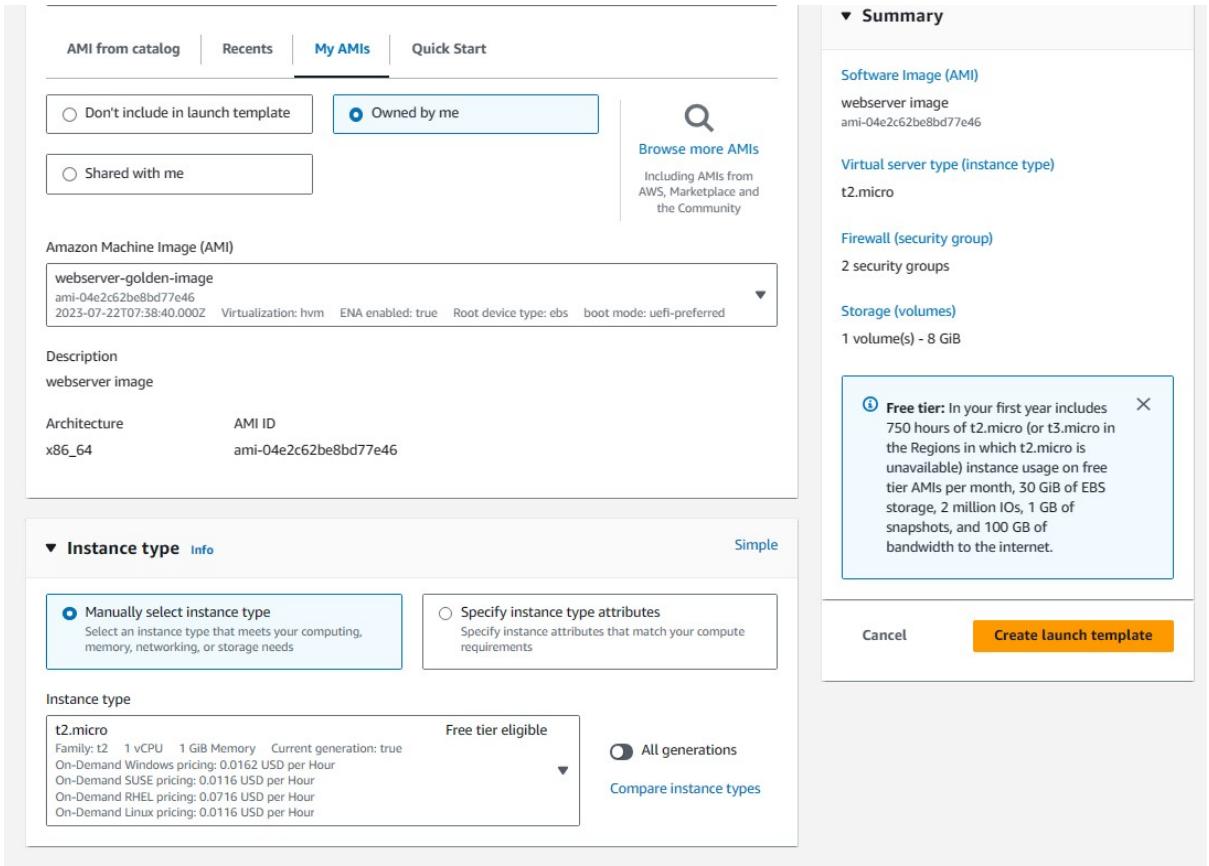


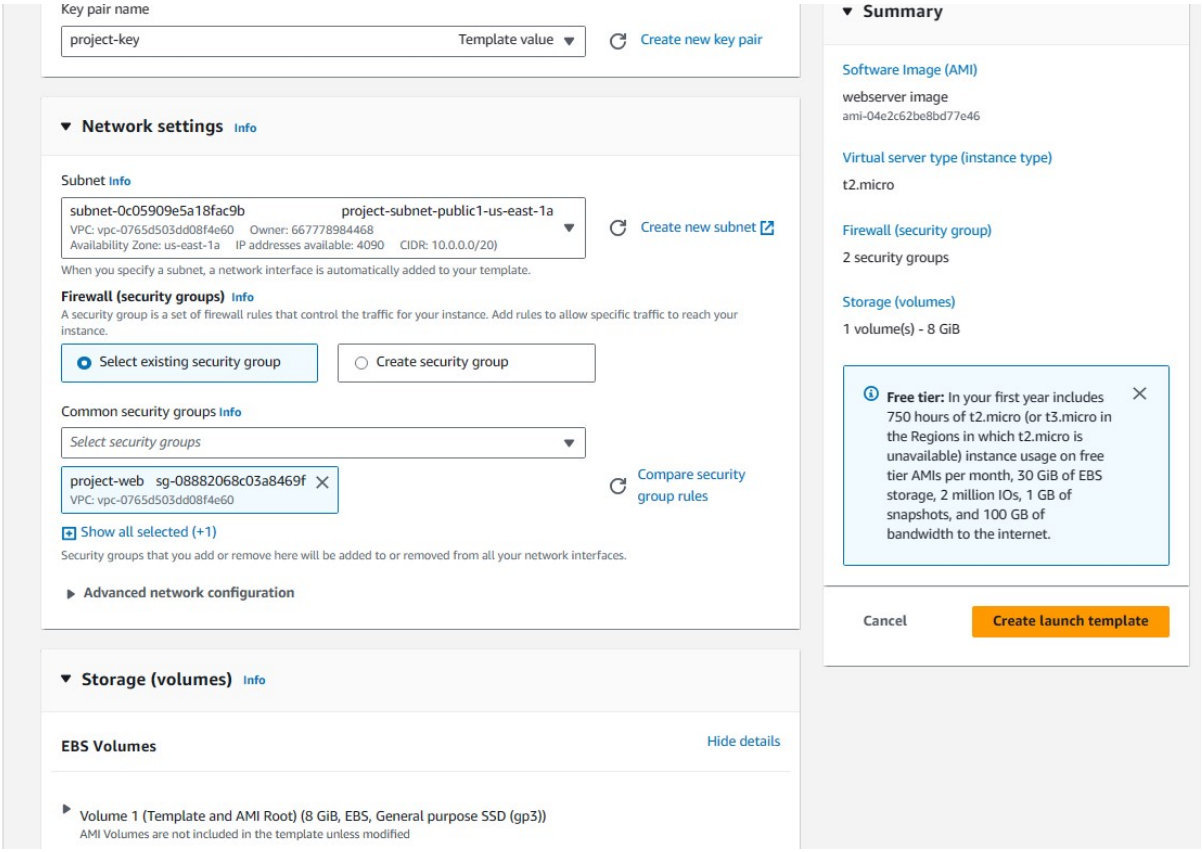
Create EC2 instance Template.

Go to EC2 instances and select the webserver and click Actions→ image and templates→Create template from instance
Provided Launch template name “webserver-template”
Selected owned by me “webserver-golden-image”
Selected instance type t2.micro
Selected the existing key pair name project-key
Selected the subnet from us-east-1a public
Selected the existing security group which was created for webserver
Click create launch instance

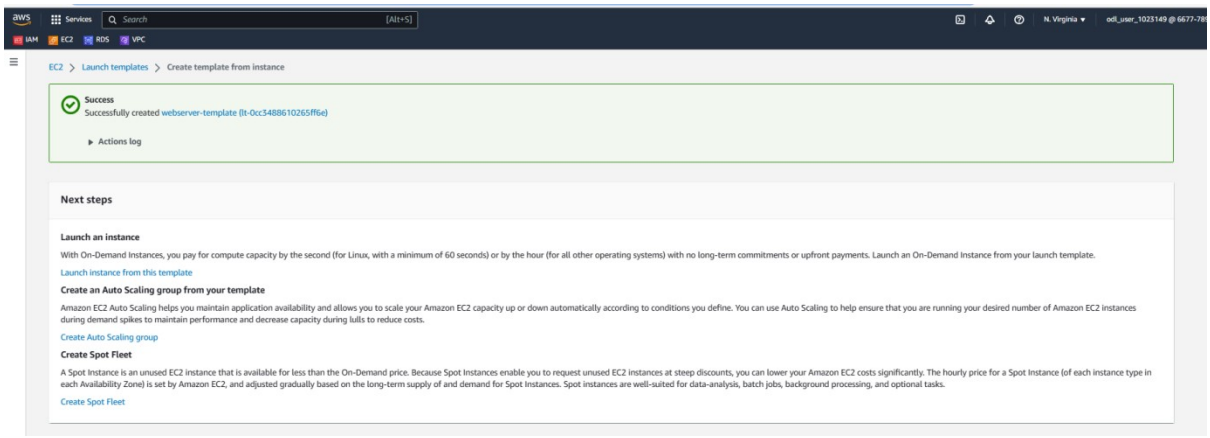



aws
Services
Search
[Alt+S]
IAM
EC2
RDS
VPC
Create launch template
Creating a launch template allows you to create a saved instance configuration that can be reused, shared and launched at a later time. Templates can have multiple versions.
Launch template name and description
Source instance
i-0c57987cae3f07f83
Launch template name - required
webserver-template
Must be unique to this account. Max 128 chars. No spaces or special characters like '&', '*', '@'.
Template version description
A webserver
Max 255 chars
Auto Scaling guidance Info
Select this if you intend to use this template with EC2 Auto Scaling
Provide guidance to help me set up a template that I can use with EC2 Auto Scaling
Template tags
Launch template contents
Specify the details of your launch template below. Leaving a field blank will result in the field not being included in the launch template.
Summary
Software Image (AMI)
webserver image
ami-04e2c62be8bd77e46
Virtual server type (instance type)
t2.micro
Firewall (security group)
2 security groups
Storage (volumes)
1 volume(s) - 8 GiB
Free tier: In your first year includes 750 hours of t2.micro (or t3.micro in the Regions in which t2.micro is unavailable) instance usage on free tier AMIs per month, 30 GiB of EBS storage, 2 million IOs, 1 GB of snapshots, and 100 GB of bandwidth to the internet.
Cancel
Create launch template


AMI from catalog
Recents
My AMIs
Quick Start
Don't include in launch template
Owned by me
Shared with me
Browse more AMIs
Including AMIs from AWS, Marketplace and the Community
Amazon Machine Image (AMI)
webserver-golden-image
ami-04e2c62be8bd77e46
2023-07-22T07:38:40.000Z Virtualization: hvm ENA enabled: true Root device type: ebs boot mode: uefi-preferred
Description
webserver image
Architecture
x86_64
AMI ID
ami-04e2c62be8bd77e46
Instance type Info
Simple
Manually select instance type
Select an instance type that meets your computing, memory, networking, or storage needs
Specify instance type attributes
Specify instance attributes that match your compute requirements
Instance type
t2.micro
Free tier eligible
Family: t2 1 vCPU 1 GiB Memory Current generation: true
On-Demand Windows pricing: 0.0162 USD per Hour
On-Demand SUSE pricing: 0.0116 USD per Hour
On-Demand RHEL pricing: 0.0716 USD per Hour
On-Demand Linux pricing: 0.0116 USD per Hour
All generations
Compare instance types
Summary
Software Image (AMI)
webserver image
ami-04e2c62be8bd77e46
Virtual server type (instance type)
t2.micro
Firewall (security group)
2 security groups
Storage (volumes)
1 volume(s) - 8 GiB
Free tier: In your first year includes 750 hours of t2.micro (or t3.micro in the Regions in which t2.micro is unavailable) instance usage on free tier AMIs per month, 30 GiB of EBS storage, 2 million IOs, 1 GB of snapshots, and 100 GB of bandwidth to the internet.
Cancel
Create launch template



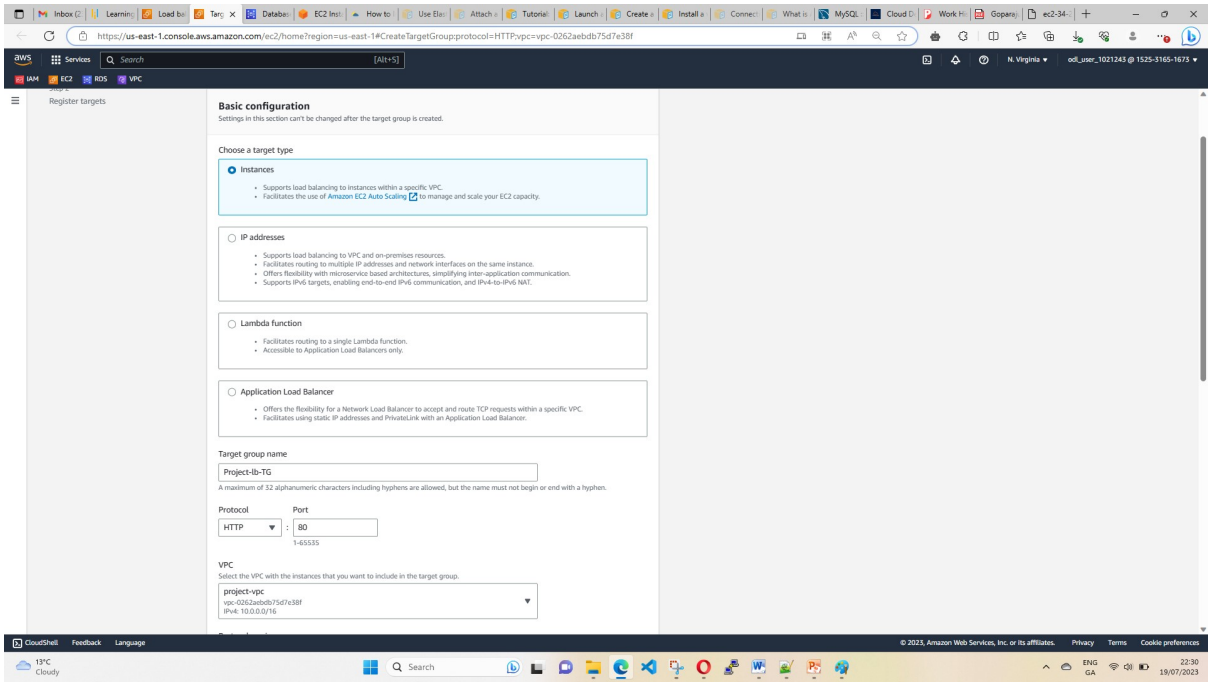
Successfully Instance template created and this will be used for Auto scaling group

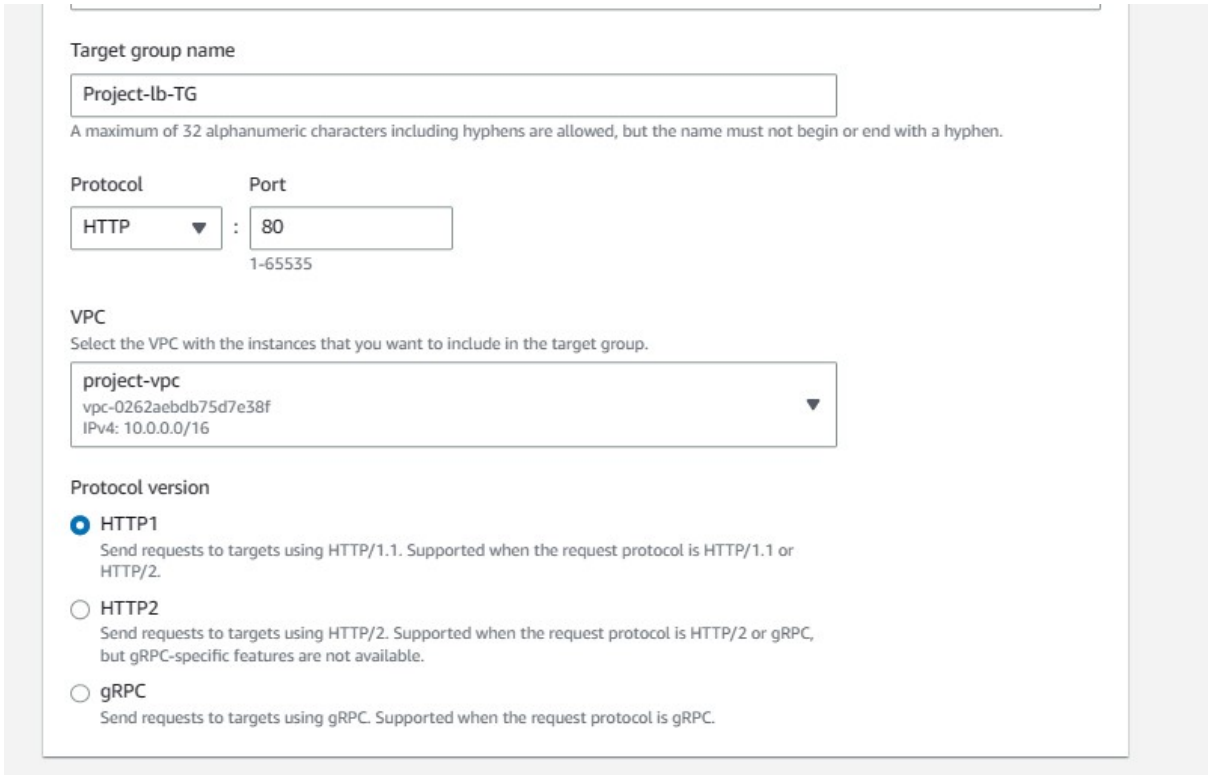


Creating Auto scaling group

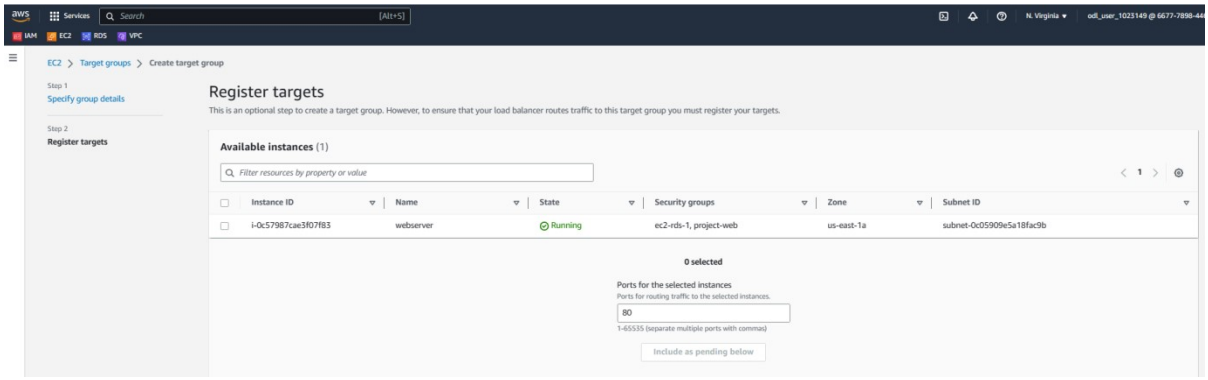
Create Target Group with following inputs:

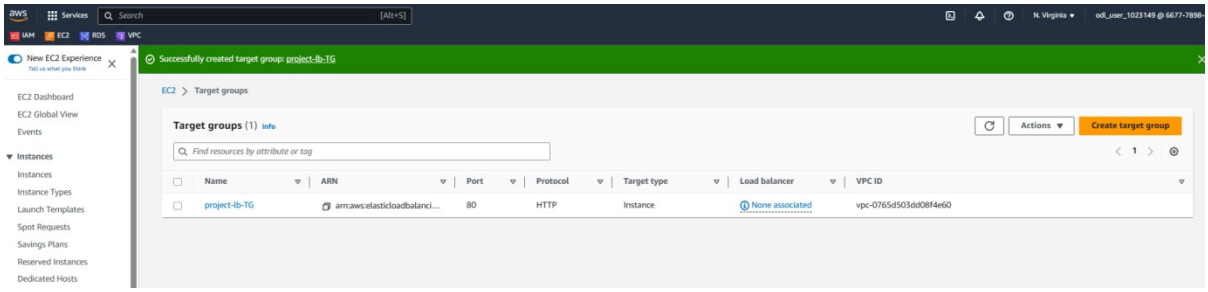
Target type: Instances
Target group name: Project-lb-TG
Protocol: HTTP port 80
VPC: Select the created VPC “project-vpc”
Create Target Group



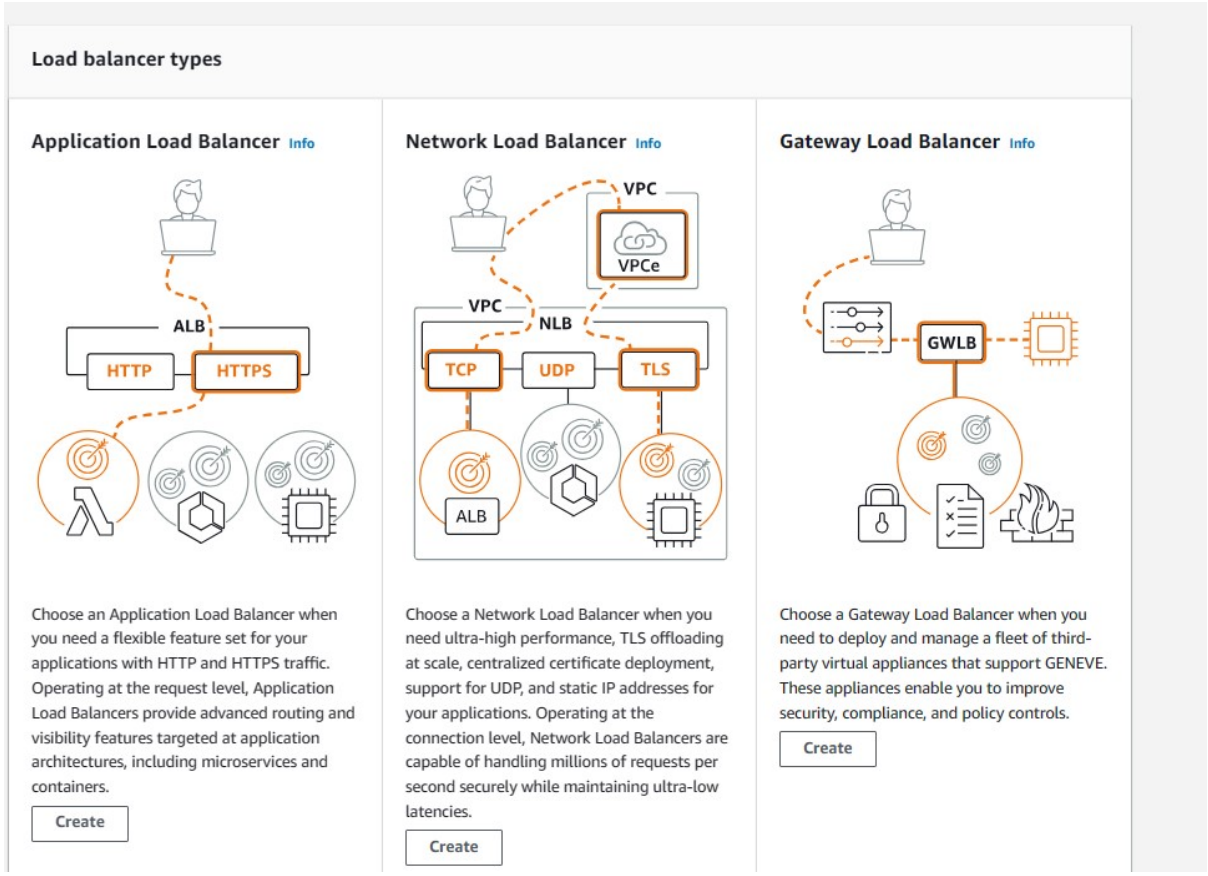


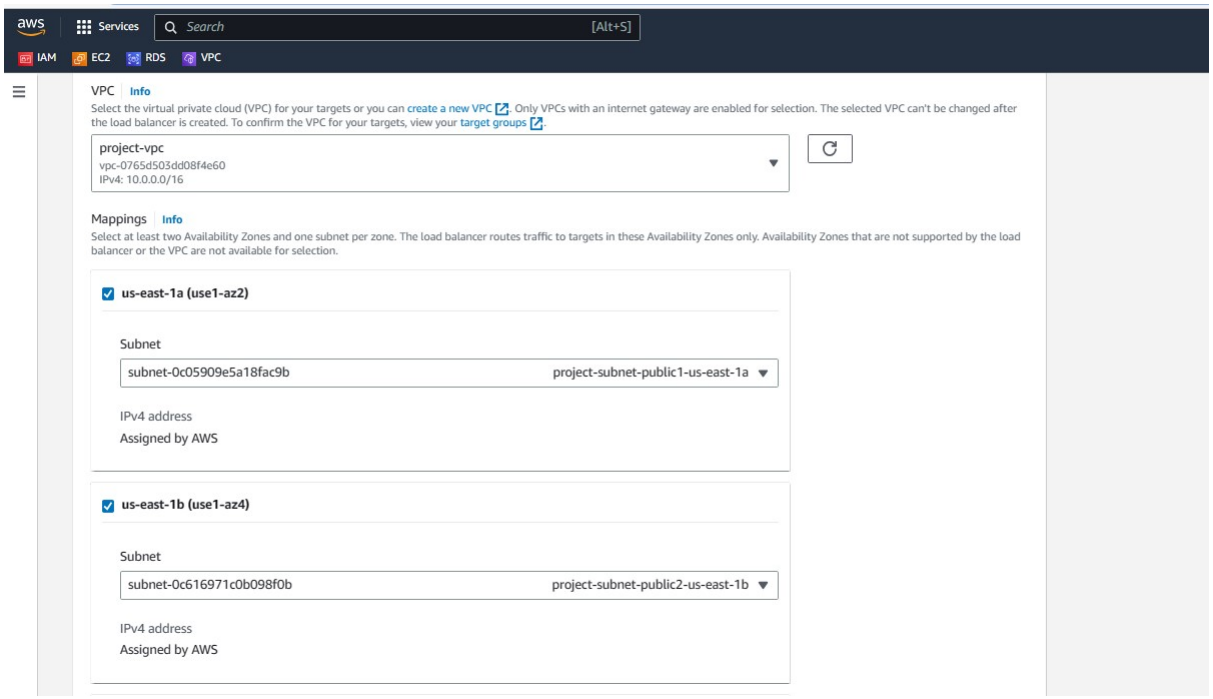
Successfully created Target Group

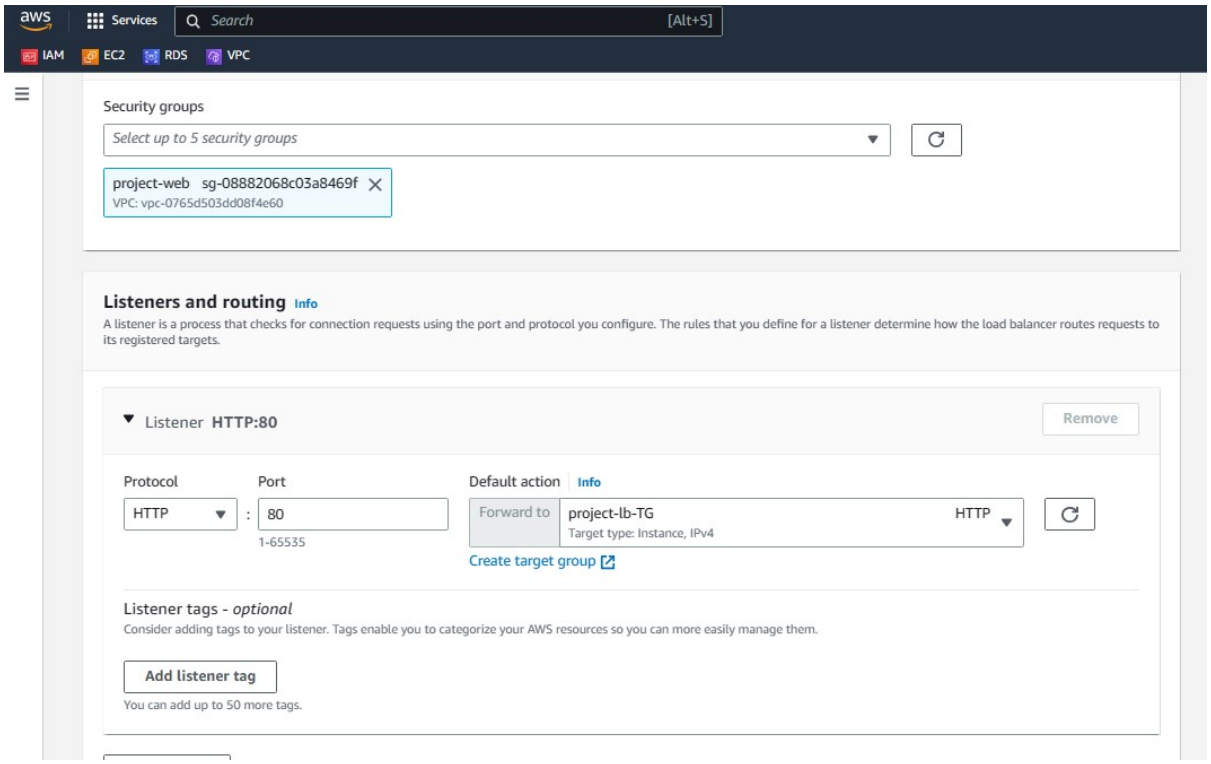


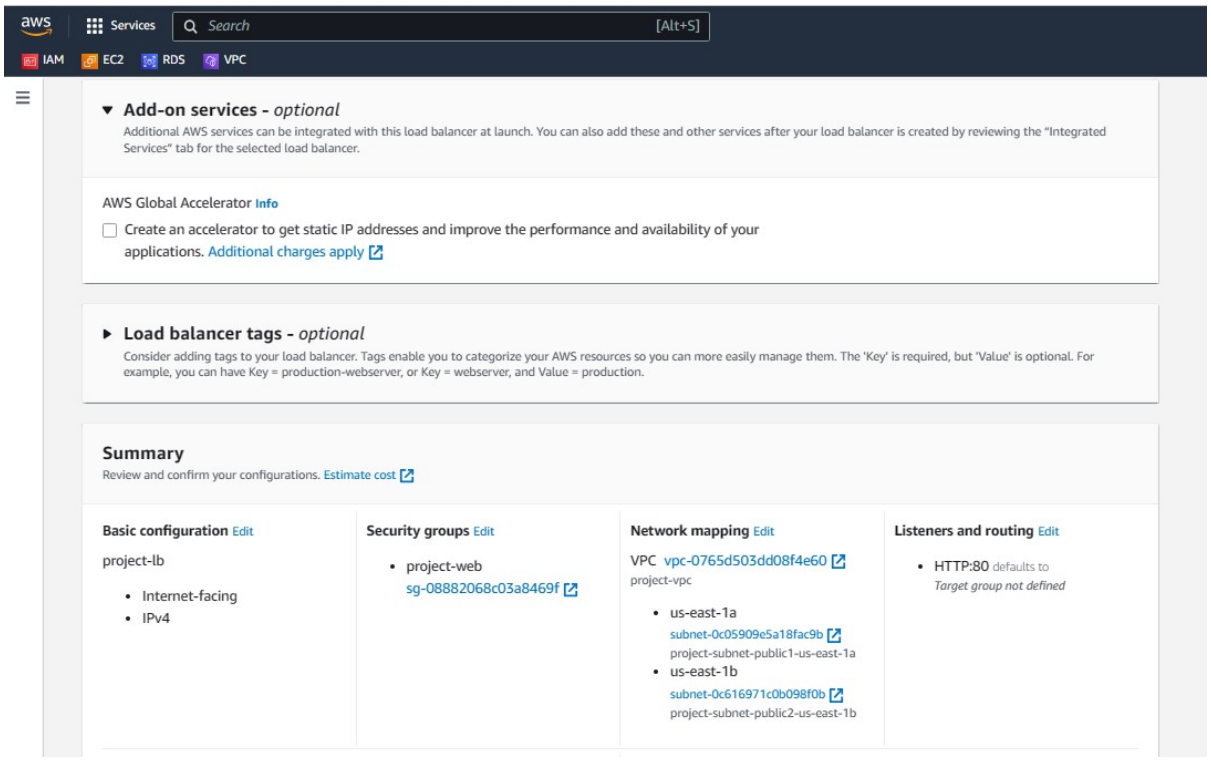


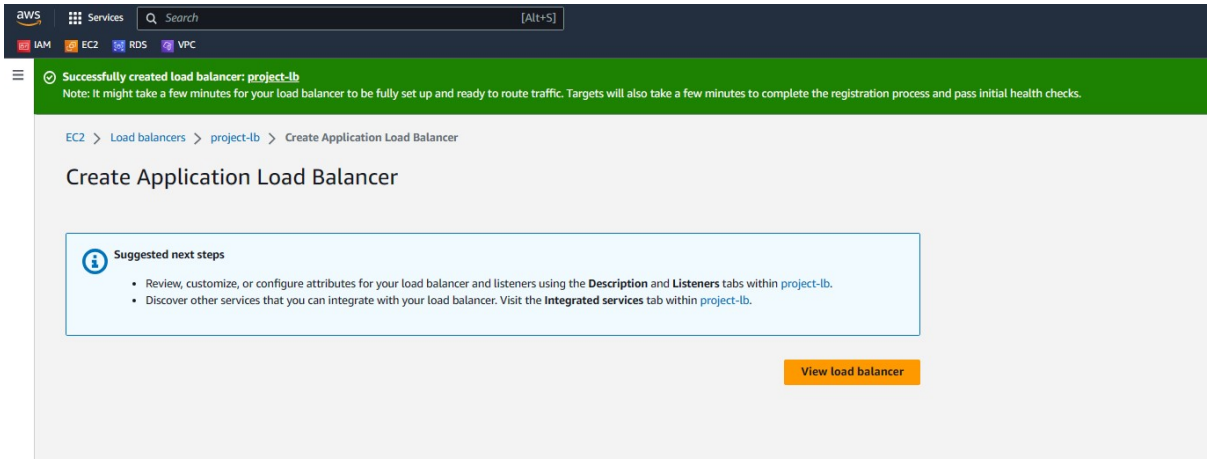
Created application load balancer



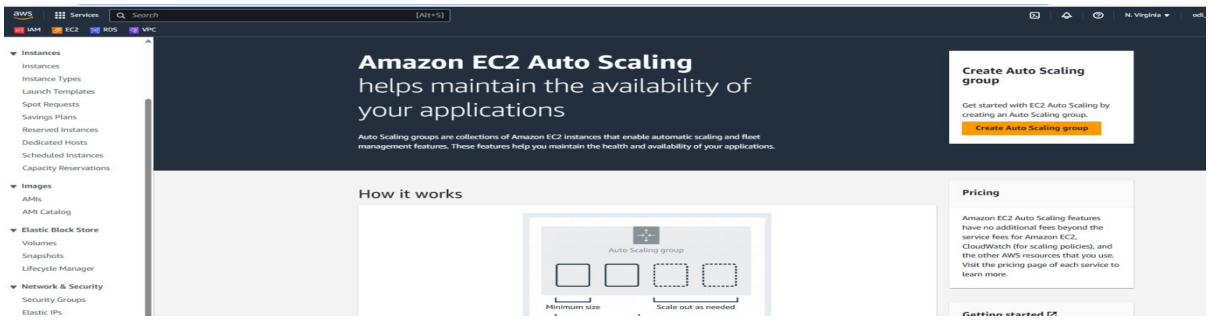

aws
Services
Search
[Alt+S]
IAM
EC2
RDS
VPC
VPC Info
Select the virtual private cloud (VPC) for your targets or you can create a new VPC. Only VPCs with an internet gateway are enabled for selection. The selected VPC can't be changed after the load balancer is created. To confirm the VPC for your targets, view your target groups.
project-vpc
vpc-0765d503dd08f4e60
IPv4: 10.0.0.0/16
Mappings Info
Select at least two Availability Zones and one subnet per zone. The load balancer routes traffic to targets in these Availability Zones only. Availability Zones that are not supported by the load balancer or the VPC are not available for selection.
us-east-1a (use1-az2)
Subnet
subnet-0c05909e5a18fac9b
project-subnet-public1-us-east-1a
IPv4 address
Assigned by AWS
us-east-1b (use1-az4)
Subnet
subnet-0c616971c0b098f0b
project-subnet-public2-us-east-1b
IPv4 address
Assigned by AWS

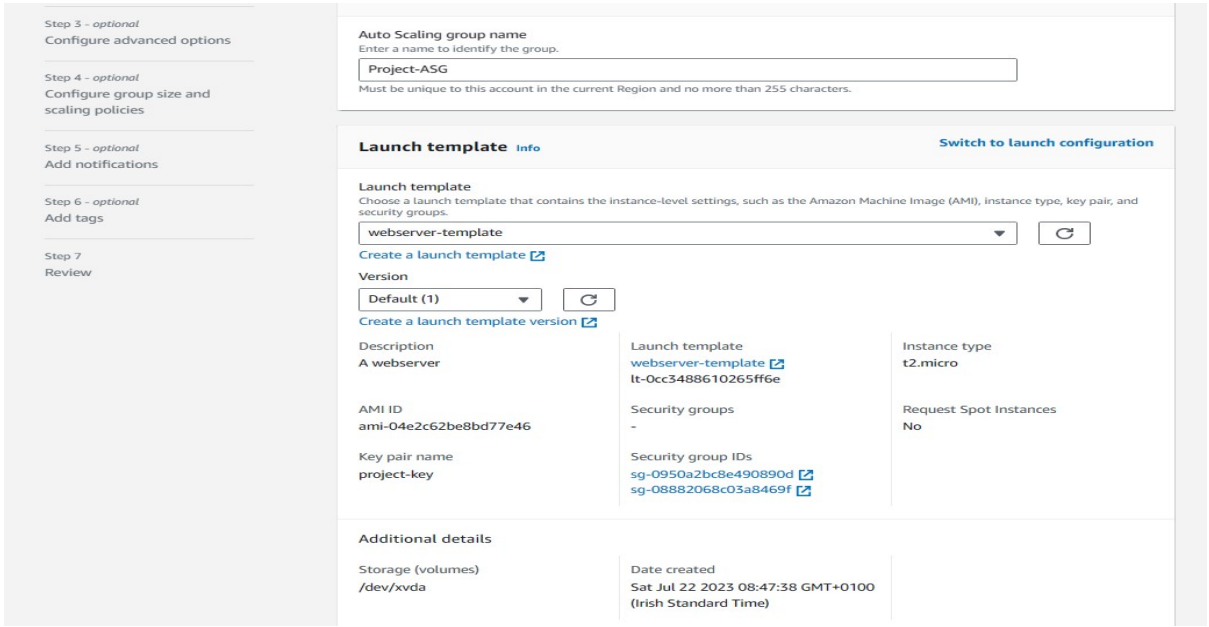

aws
Services
Search
[Alt+S]
IAM
EC2
RDS
VPC
Security groups
Select up to 5 security groups
project-web sg-08882068c03a8469f
VPC: vpc-0765d503dd08f4e60
Listeners and routing Info
A listener is a process that checks for connection requests using the port and protocol you configure. The rules that you define for a listener determine how the load balancer routes requests to its registered targets.
Listener HTTP:80
Remove
Protocol
Port
Default action Info
HTTP
80
1-65535
Forward to
project-lb-TG
HTTP
Target type: Instance, IPv4
Create target group
Listener tags - optional
Consider adding tags to your listener. Tags enable you to categorize your AWS resources so you can more easily manage them.
Add listener tag
You can add up to 50 more tags.

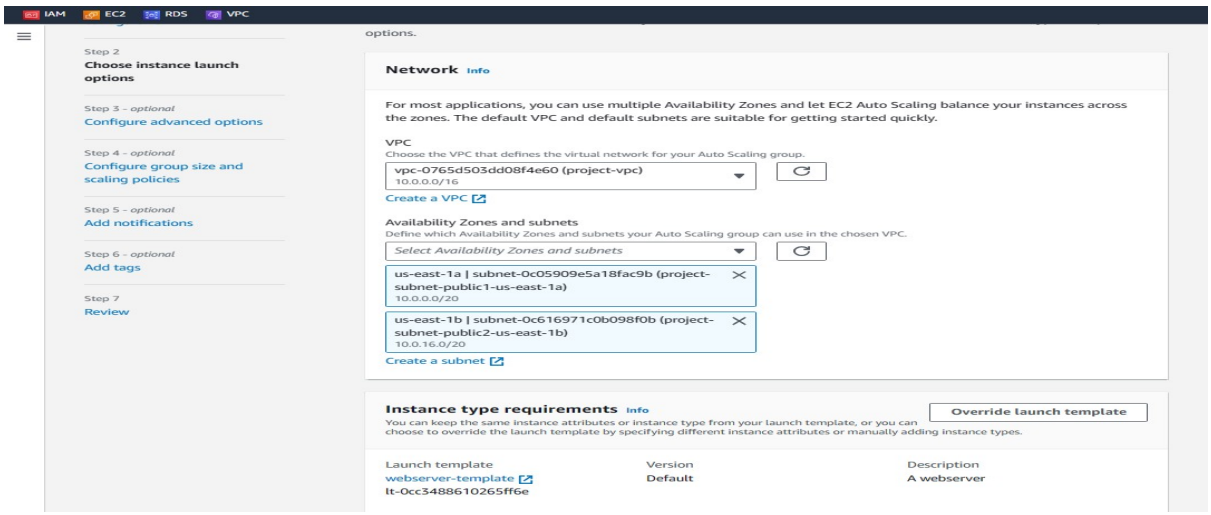

aws
Services
Search
[Alt+S]
IAM
EC2
RDS
VPC
Add-on services - optional
Additional AWS services can be integrated with this load balancer at launch. You can also add these and other services after your load balancer is created by reviewing the "Integrated Services" tab for the selected load balancer.
AWS Global Accelerator Info
Create an accelerator to get static IP addresses and improve the performance and availability of your applications. Additional charges apply
Load balancer tags - optional
Consider adding tags to your load balancer. Tags enable you to categorize your AWS resources so you can more easily manage them. The 'Key' is required, but 'Value' is optional. For example, you can have Key = production-webserver, or Key = webserver, and Value = production.
Summary
Review and confirm your configurations. Estimate cost
Basic configuration Edit
project-lb
Internet-facing
IPv4
Security groups Edit
project-web
sg-08882068c03a8469f
Network mapping Edit
VPC vpc-0765d503dd08f4e60
project-vpc
us-east-1a
subnet-0c05909e5a18fac9b
project-subnet-public1-us-east-1a
us-east-1b
subnet-0c616971c0b098f0b
project-subnet-public2-us-east-1b
Listeners and routing Edit
HTTP:80 defaults to
Target group not defined

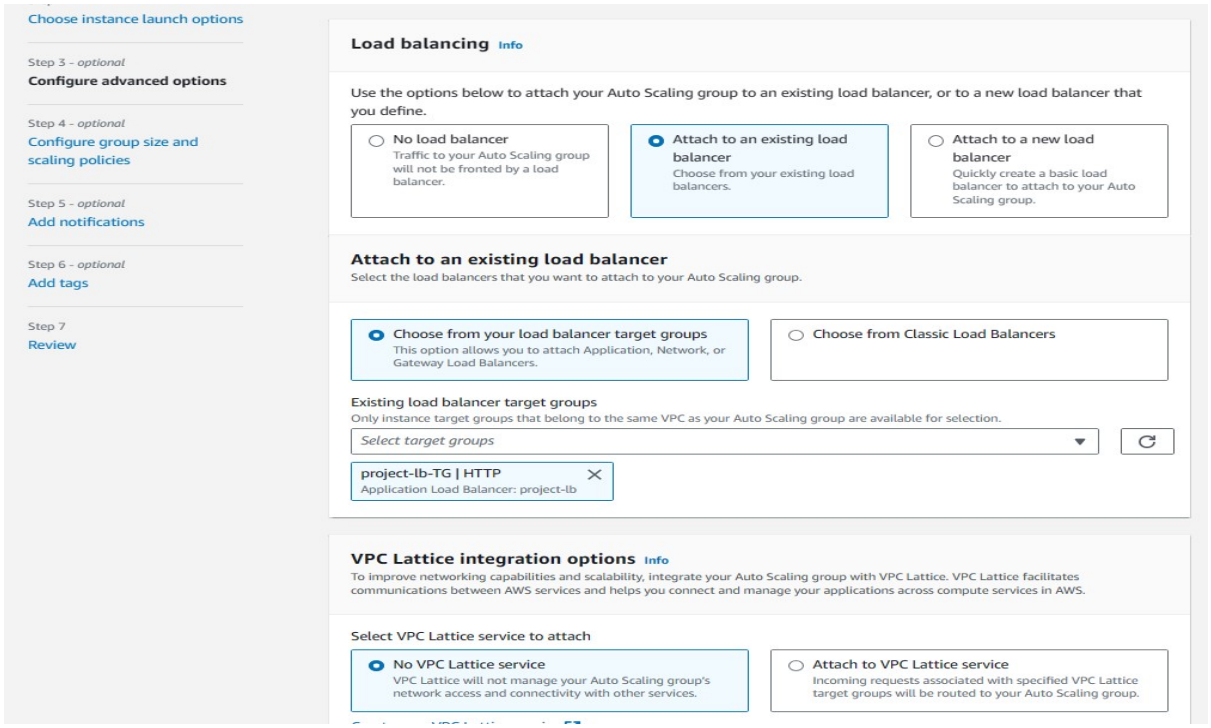

aws
Services
Search
[Alt+S]
IAM
EC2
RDS
VPC
Successfully created load balancer: project-lb
Note: It might take a few minutes for your load balancer to be fully set up and ready to route traffic. Targets will also take a few minutes to complete the registration process and pass initial health checks.
EC2 > Load balancers > project-lb > Create Application Load Balancer
Create Application Load Balancer
Suggested next steps
Review, customize, or configure attributes for your load balancer and listeners using the Description and Listeners tabs within project-lb.
Discover other services that you can integrate with your load balancer. Visit the Integrated services tab within project-lb.
View load balancer

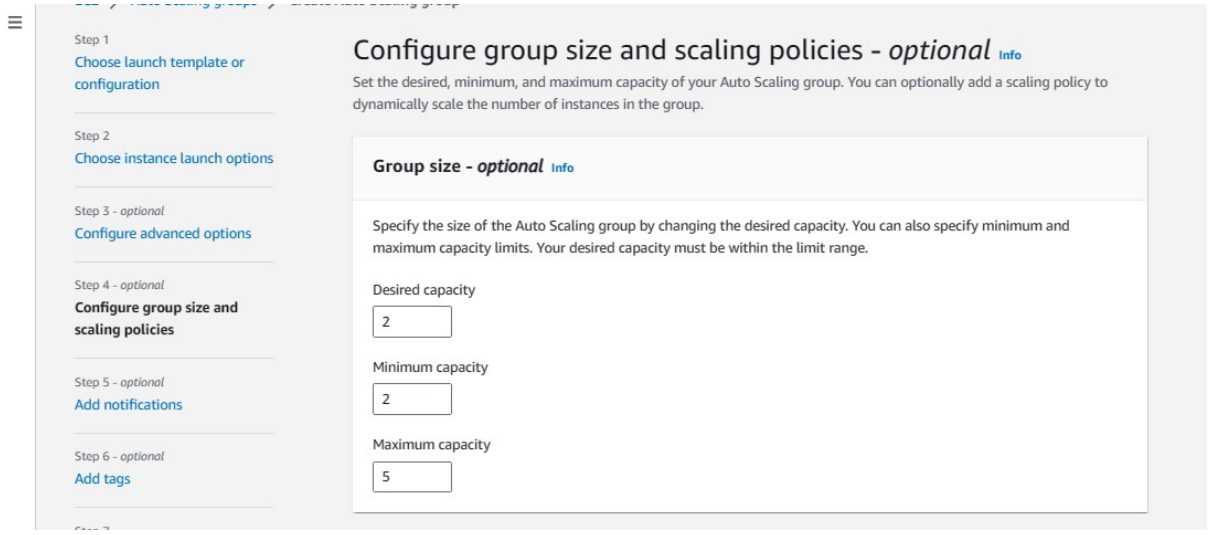
Created Auto scaling group:

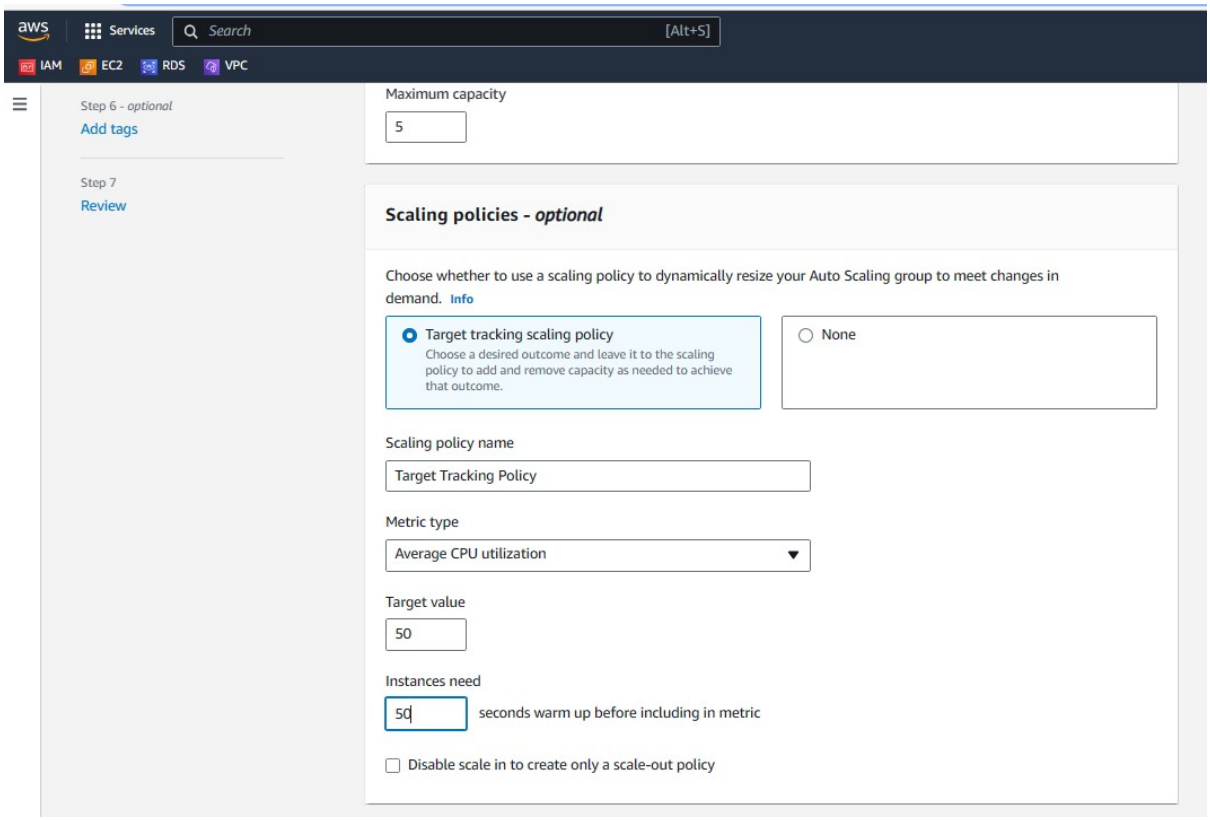




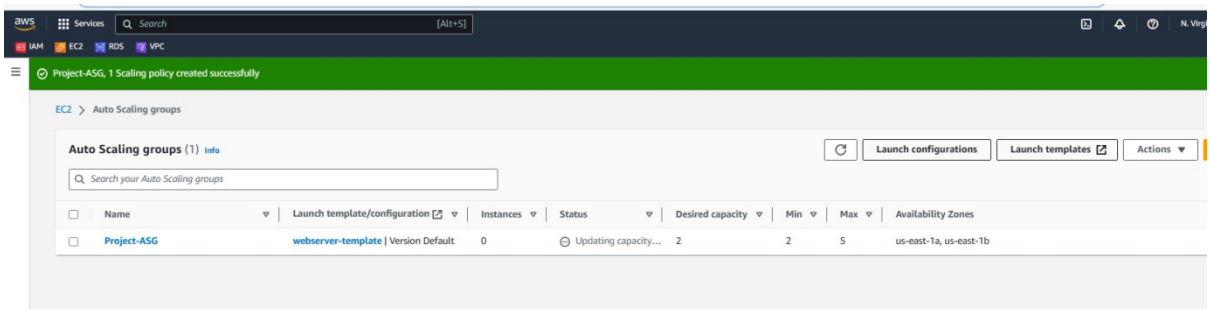



Choose instance launch options
Step 3 - optional
Configure advanced options
Step 4 - optional
Configure group size and scaling policies
Step 5 - optional
Add notifications
Step 6 - optional
Add tags
Step 7
Review
Load balancing Info
Use the options below to attach your Auto Scaling group to an existing load balancer, or to a new load balancer that you define.
No load balancer
Traffic to your Auto Scaling group will not be fronted by a load balancer.
Attach to an existing load balancer
Choose from your existing load balancers.
Attach to a new load balancer
Quickly create a basic load balancer to attach to your Auto Scaling group.
Attach to an existing load balancer
Select the load balancers that you want to attach to your Auto Scaling group.
Choose from your load balancer target groups
This option allows you to attach Application, Network, or Gateway Load Balancers.
Choose from Classic Load Balancers
Existing load balancer target groups
Only instance target groups that belong to the same VPC as your Auto Scaling group are available for selection.
Select target groups
project-lb-TG | HTTP
Application Load Balancer: project-lb
VPC Lattice integration options Info
To improve networking capabilities and scalability, integrate your Auto Scaling group with VPC Lattice. VPC Lattice facilitates communications between AWS services and helps you connect and manage your applications across compute services in AWS.
Select VPC Lattice service to attach
No VPC Lattice service
VPC Lattice will not manage your Auto Scaling group's network access and connectivity with other services.
Attach to VPC Lattice service
Incoming requests associated with specified VPC Lattice target groups will be routed to your Auto Scaling group.

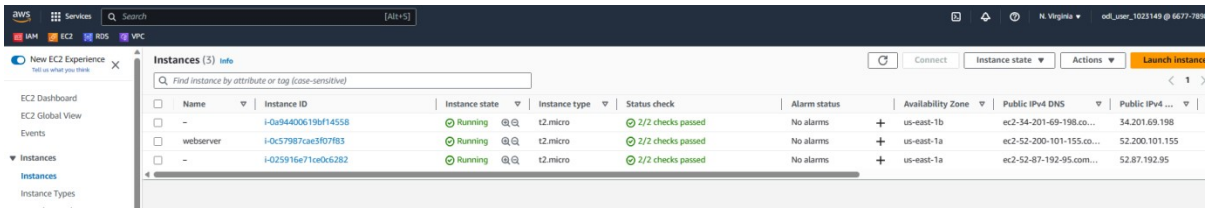

Step 1
Choose launch template or configuration
Step 2
Choose instance launch options
Step 3 - optional
Configure advanced options
Step 4 - optional
Configure group size and scaling policies
Step 5 - optional
Add notifications
Step 6 - optional
Add tags
Configure group size and scaling policies - optional Info
Set the desired, minimum, and maximum capacity of your Auto Scaling group. You can optionally add a scaling policy to dynamically scale the number of instances in the group.
Group size - optional Info
Specify the size of the Auto Scaling group by changing the desired capacity. You can also specify minimum and maximum capacity limits. Your desired capacity must be within the limit range.
Desired capacity
2
Minimum capacity
2
Maximum capacity
5



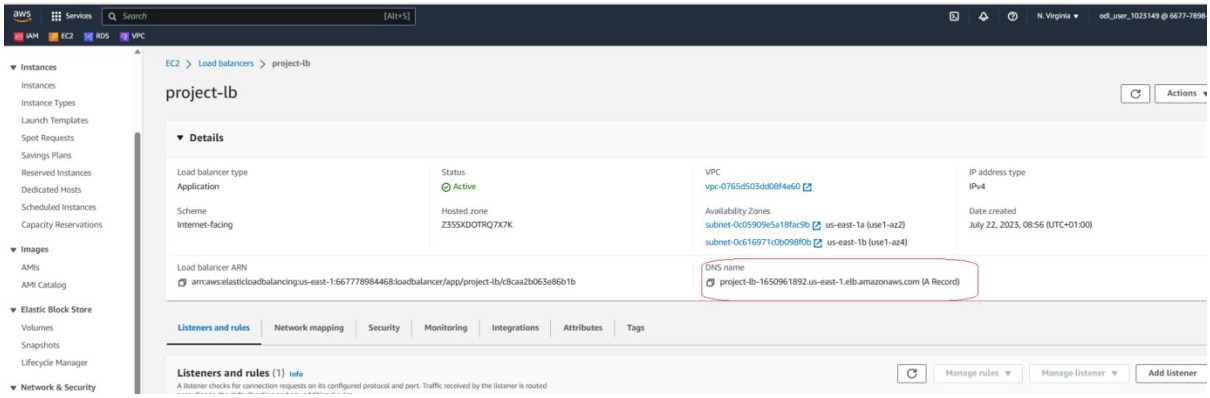
Successfully created auto scaling group



Desired 2 webserver EC2 instances created and running

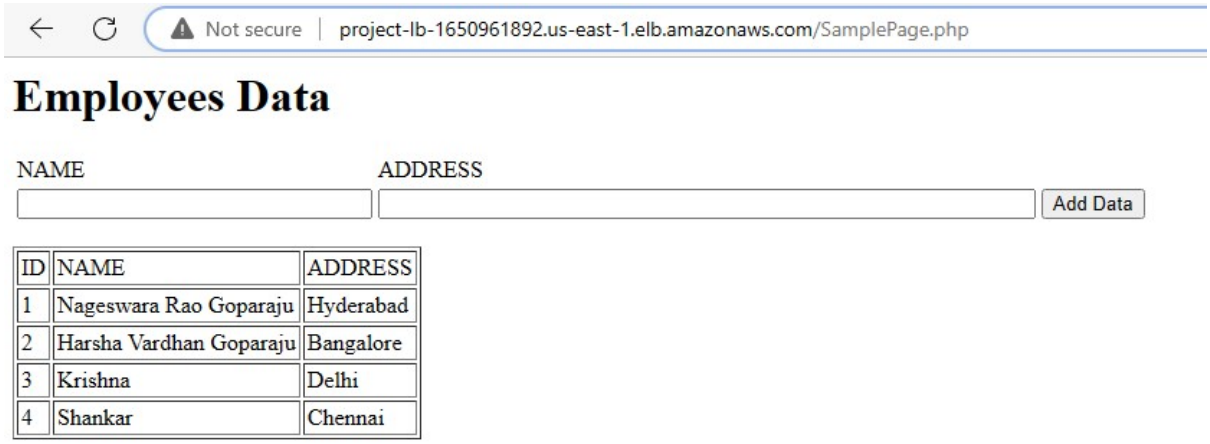


Check the LB DNS name and open the URL from the web browser.



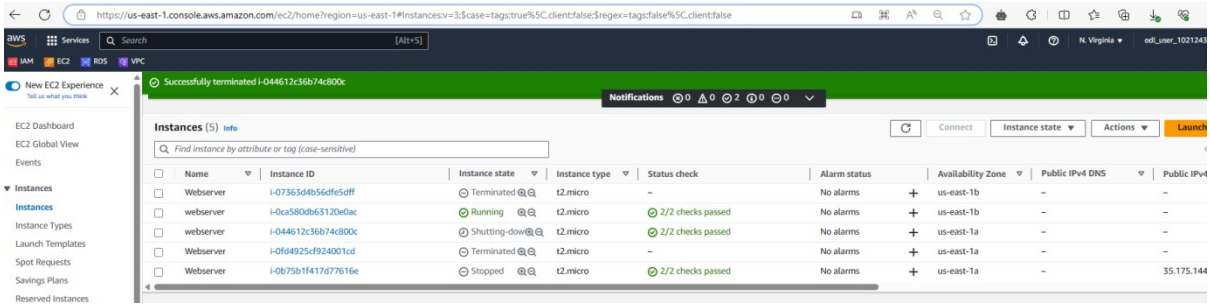
Successfully able to connect webserver page

Update more data to check the webserver and database are working as expected



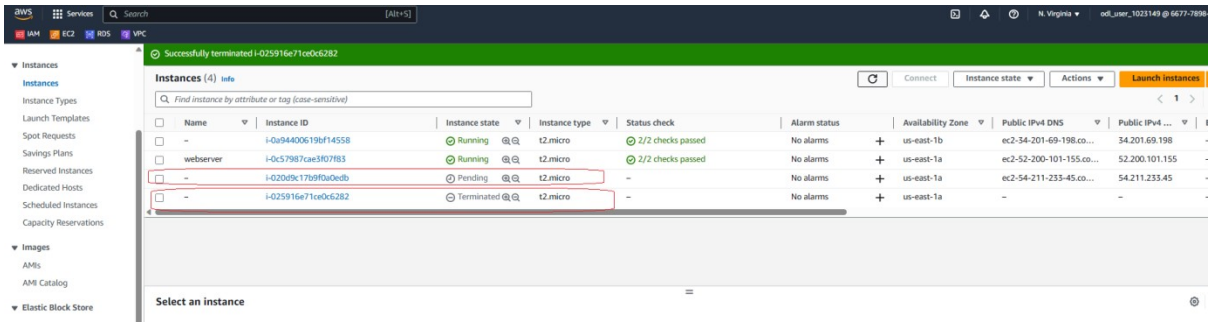
Checking the High Availability using the termination of the running webserver

Terminated one webserver and check the application availability from the second instance



Still the application up and running

Check the HA working to make it desired state 2
Within few mints second instances initiated and started running



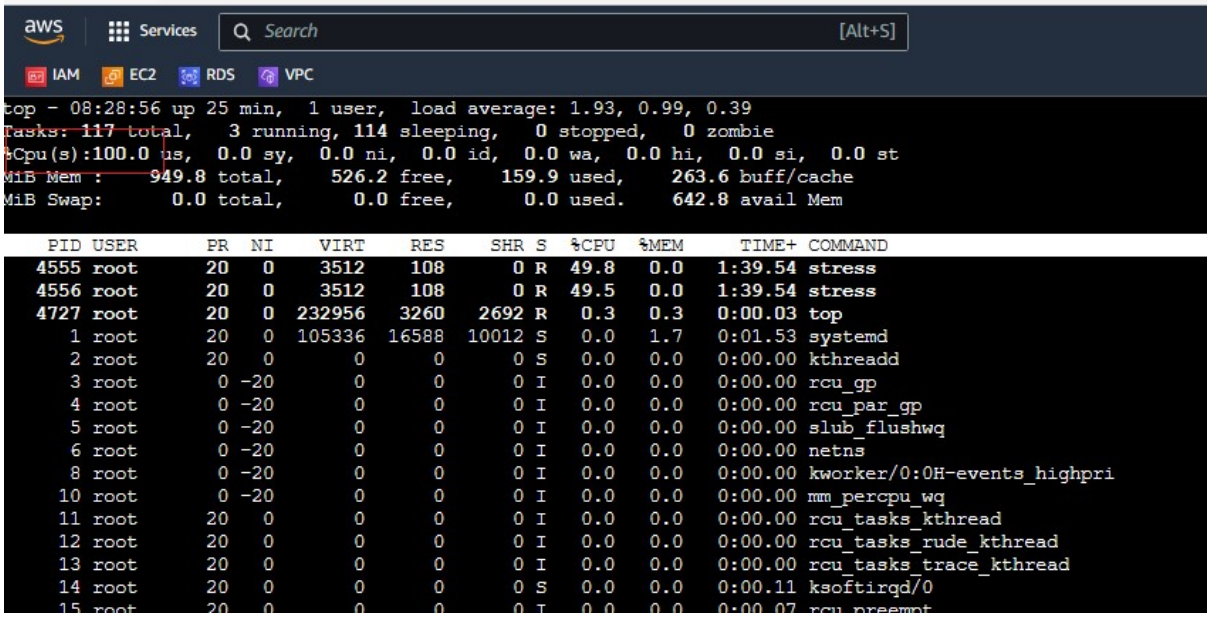
Check the Stress testing to make it increase webserver instances
Install stress on EC2 instances and run stress command

```
yum install stress -y
```

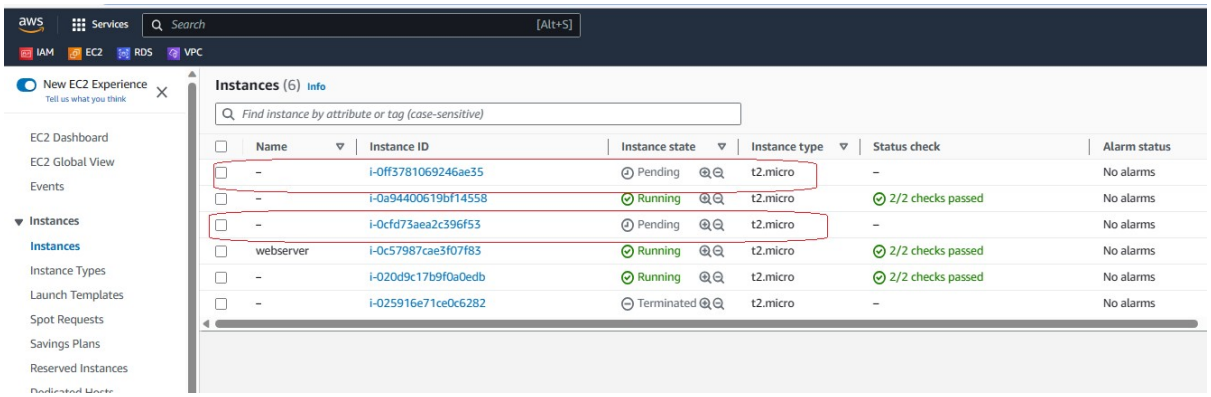
```
[root@ip-10-0-31-125 ~]# yum install stress -y
Last metadata expiration check: 1:02:43 ago on Sat Jul 22 07:20:33 2023.
Dependencies resolved.
================================================================================
 Package                    Architecture                    Version
================================================================================
Installing:
 stress                     x86_64                          1.0.4-28.amzn2023.0.2

Transaction Summary
================================================================================
Install  1 Package

Total download size: 37 k
Installed size: 78 k
Downloading Packages:
stress-1.0.4-28.amzn2023.0.2.x86_64.rpm
--------------------------------------------------------------------------------
Total
Running transaction check
Transaction check succeeded.
Running transaction test
Transaction test succeeded.
Running transaction
  Preparing        :
  Installing       : stress-1.0.4-28.amzn2023.0.2.x86_64
  Running scriptlet: stress-1.0.4-28.amzn2023.0.2.x86_64
  Verifying        : stress-1.0.4-28.amzn2023.0.2.x86_64

Installed:
  stress-1.0.4-28.amzn2023.0.2.x86_64

Complete!
```

Stress applied on the CPU

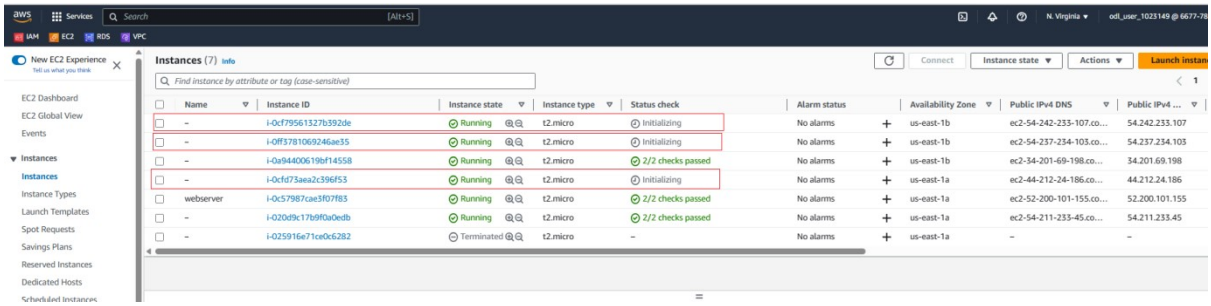
```
Complete!
[root@ip-10-0-0-17 ~]# stress --cpu 2 --timeout 1000 &
[1] 3252
[root@ip-10-0-0-17 ~]# stress: info: [3252] dispatching hogs: 2 cpu, 0 io, 0 vm, 0 hdd
```



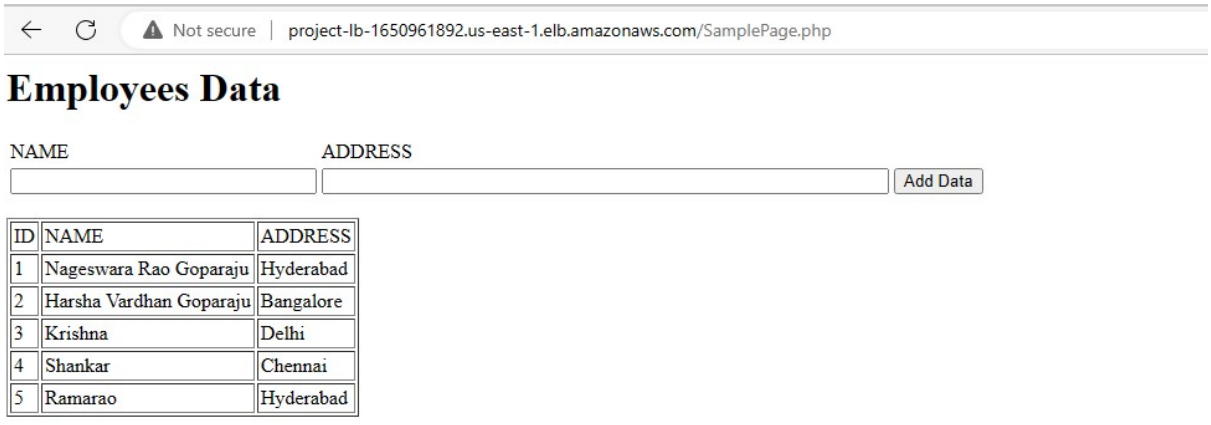
Started creating new Instances after its reaches 50% threshold



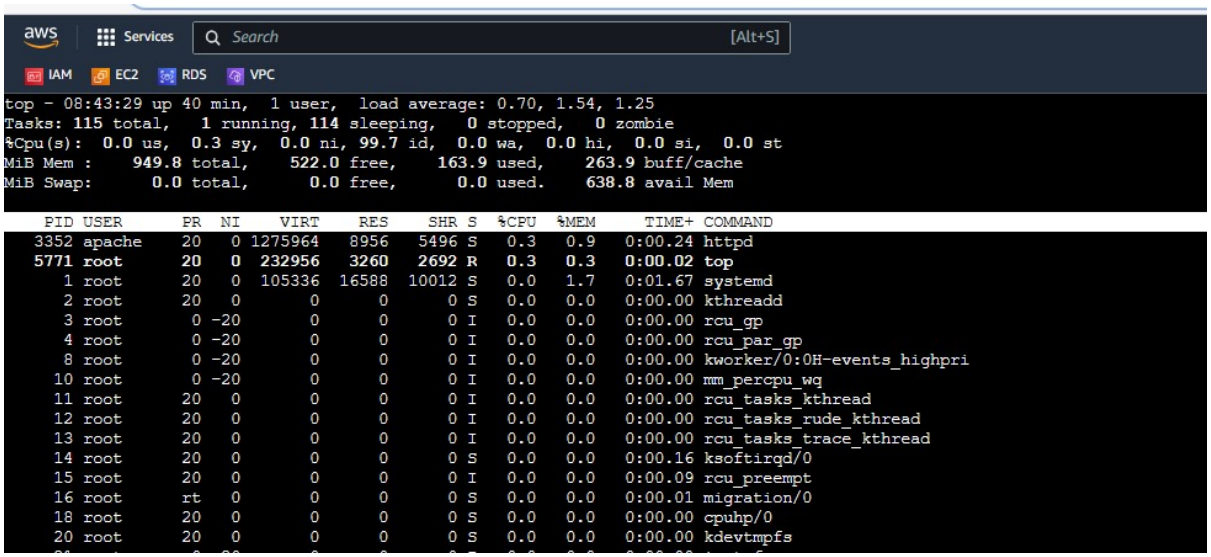
Reached max 5 instances count



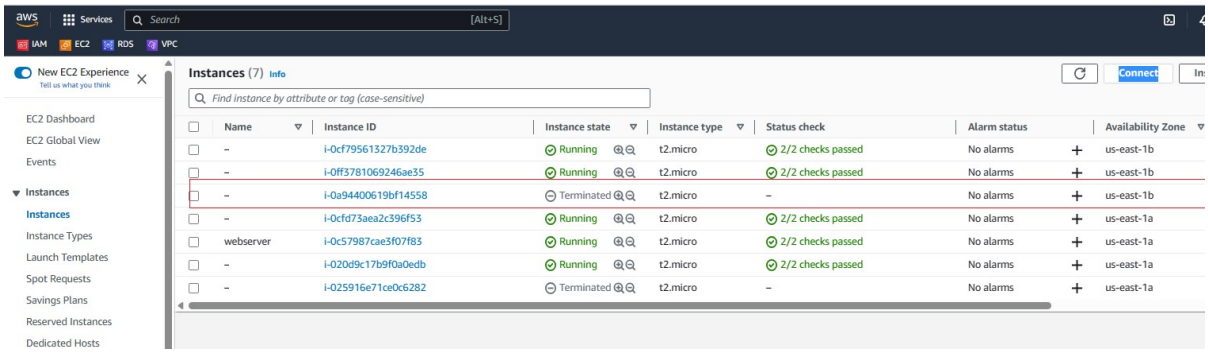
Able to insert data



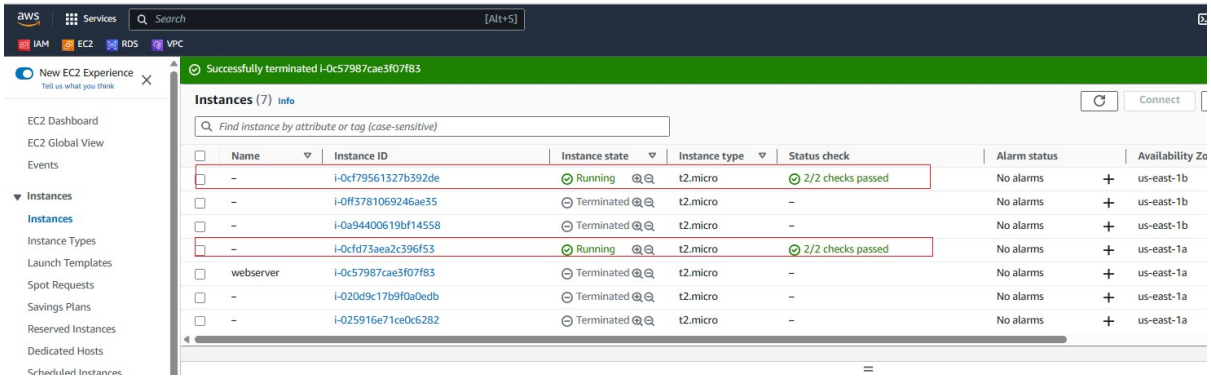
Now the CPU load reduced to below 50%



Started Scale in from Auto scaling



Now only running with desired instances



Result

Finally the webserver up and running with High availability using Auto scaling group as expected.

END of the Project