

Create IAM user

The screenshot shows the 'Create user' page in the AWS IAM console. A green success message box at the top states: 'User created successfully. You can view and download the user's password and email instructions for signing in to the AWS Management Console.' Below this, a 'View user' button is visible. To the left, a vertical navigation bar shows steps: Step 1 (Specify user details), Step 2 (Set permissions), Step 3 (Review and create), and Step 4 (Retrieve password), with Step 4 currently selected. The main content area is titled 'Retrieve password' and contains 'Console sign-in details'. It includes a 'Console sign-in URL' (https://512692426603.signin.aws.amazon.com/console), a 'User name' (shravani-admin), and a 'Console password' (represented by a series of asterisks). An 'Email sign-in instructions' link is also present. At the bottom right are 'Cancel', 'Download .csv file', and 'Return to users list' buttons.

Create VPC

The screenshot shows the 'Your VPCs' page in the AWS VPC console. A green success message box at the top states: 'You successfully created vpc-0e96a0d7b539dde35 / shravani-vpc'. Below this, the VPC details are listed in a table:

VPC ID	State	Block Public Access	DNS hostnames
vpc-0e96a0d7b539dde35	Available	Off	Disabled
DNS resolution	Tenancy	DHCP option set	Main route table
Enabled	default	dopt-0ba5e0a40b5476f28	rtb-06297d232b879469c
Main network ACL	Default VPC	IPv4 CIDR	IPv6 pool
acl-0669c5022c5dc0aa	No	10.0.0.0/16	-
IPv6 CIDR (Network border group)	Network Address Usage metrics	Route 53 Resolver DNS Firewall rule groups	Owner ID
-	Disabled	-	512692426603
Encryption control ID	Encryption control mode	-	-

Create subnet

The screenshot shows the 'Subnets' page in the AWS VPC console. A green success message box at the top states: 'Subnets (2) Info'. Below this, the subnet details are listed in a table:

Name	Subnet ID	State	VPC	Block Public...	IPv4 CIDR
shravani-private-subnet	subnet-0ed0ddeb19fa45c7	Available	vpc-0e96a0d7b539dde35 shr...	Off	10.0.2.0/24
shravani-public-subnet	subnet-0e8343503374b0b27	Available	vpc-0e96a0d7b539dde35 shr...	Off	10.0.1.0/24

Enabling auto-assign IP for Public subnet

The screenshot shows the 'Edit subnet settings' page for a public subnet. Under 'Auto-assign IP settings', the 'Enable auto-assign public IPv4 address' checkbox is checked, while the 'Enable auto-assign customer-owned IPv4 address' checkbox is unchecked.

Create Internet Gateway

The screenshot shows the 'Internet gateways' page. A success message indicates a new Internet Gateway 'igw-0d51d77e1c98c7ef1' has been created. The gateway details show it is in a 'Detached' state and associated with the VPC 'shravani-vpc'. It has one tag named 'shravani-IGW'.

Attach Internet Gateway to VPC

The screenshot shows the 'Internet gateways' page again, but this time the gateway 'igw-0d51d77e1c98c7ef1' is in an 'Attached' state, indicating it is now connected to a VPC.

Create route table -> Edit route -> edit subnet association.

The screenshot shows the 'Route tables' page. A route table 'shravani-public-RT' is selected, showing its association with the public subnet 'subnet-0e8343503374b0b27'. Other route tables listed include 'rtb-06297d232b879469c' and 'rtb-084f295b647d60ae3'.

Create an IAM role for EC2 with full S3 access

The screenshot shows the AWS IAM console with the following details:

- Role Name:** shra-ec2-role
- Description:** Allows EC2 instances to call AWS services on your behalf. IAM role for EC2 to access S3 without storing credentials.
- Summary:**
 - Creation date:** February 14, 2026, 17:45 (UTC+05:30)
 - Last activity:** -
 - ARN:** arn:aws:iam::512692426603:role/shra-ec2-role
 - Maximum session duration:** 1 hour
- Permissions:** One managed policy attached.
- Trust relationships:** Not specified.
- Tags:** None
- Last Accessed:** -
- Revoke sessions:** -

Launch EC2 Instance

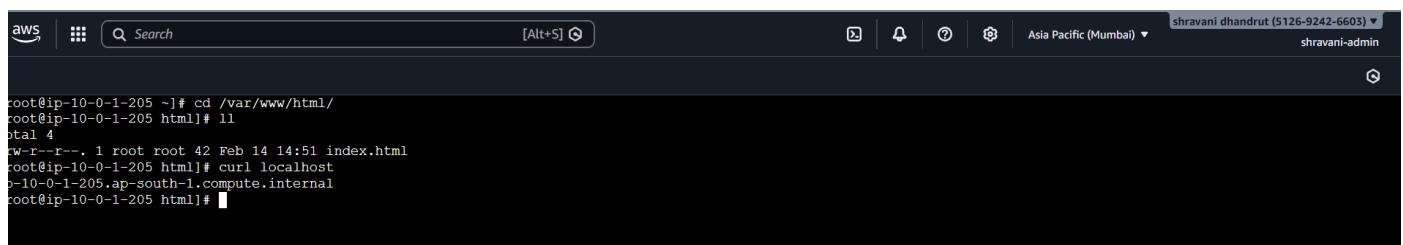
The screenshot shows the AWS EC2 Instances page with the following details:

- Instance ID:** i-080f0df9b983c0331
- Instance Type:** t3.micro
- Image:** shra-ec2-role
- Public IP Address:** 43.205.127.64
- Private IP Address:** 10.0.1.205
- Subnet ID:** subnet-0e8343503374b0b27
- Instance ARN:** arn:aws:ec2:ap-south-1:512692426603:instance/i-080f0df9b983c0331
- State:** Running
- Region:** Asia Pacific (Mumbai)

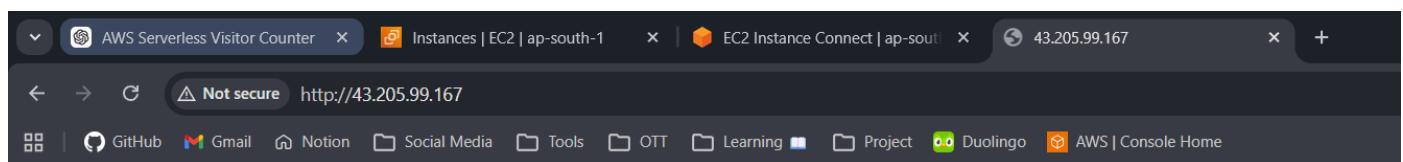
Configure Apache server over EC2

Commands:

```
yum install httpd* -y  
systemctl status httpd  
systemctl enable httpd  
systemctl start httpd  
cd /var/www/html/  
ls  
echo $(hostname -f) > index.html
```

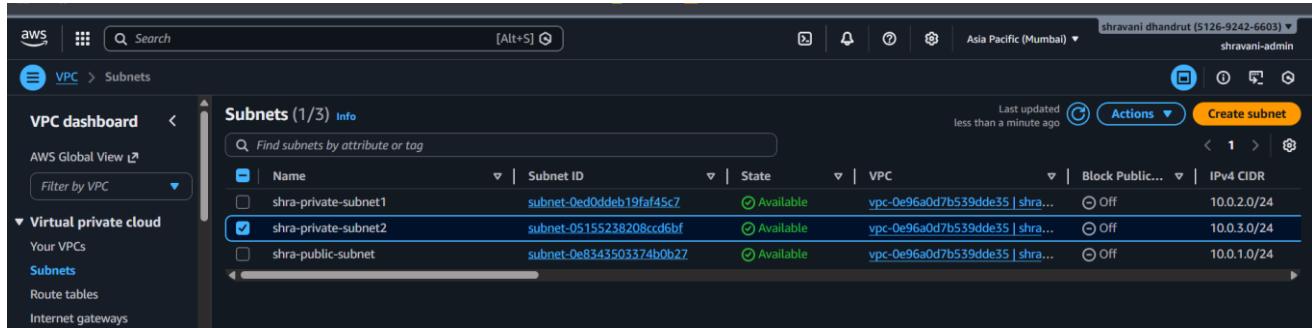


```
aws | Search [Alt+S] Asia Pacific (Mumbai) shravani-admin  
shravani dhandrut (5126-9242-6603)  
  
root@ip-10-0-1-205 ~# cd /var/www/html/  
root@ip-10-0-1-205 html]# ll  
total 4  
rw-r--r--. 1 root root 42 Feb 14 14:51 index.html  
root@ip-10-0-1-205 html]# curl localhost  
ip-10-0-1-205.ap-south-1.compute.internal  
root@ip-10-0-1-205 html]#
```



Create a MySQL Database using RDS

1. Create one more private subnet in another AZ



The screenshot shows the AWS VPC Subnets page. On the left, there's a sidebar with "VPC dashboard" and "Virtual private cloud" sections. The main area is titled "Subnets (1/3) Info" and shows a table with three rows:

Name	Subnet ID	State	VPC	Block Public...	IPv4 CIDR
shra-private-subnet1	subnet-0ed0deb19faf45c7	Available	vpc-0e96a0d7b539dde35 shra...	Off	10.0.2.0/24
shra-private-subnet2	subnet-05155238208cc6bf	Available	vpc-0e96a0d7b539dde35 shra...	Off	10.0.3.0/24
shra-public-subnet	subnet-0e8343503374b0b27	Available	vpc-0e96a0d7b539dde35 shra...	Off	10.0.1.0/24

2. Create db subnet group

The screenshot shows the AWS RDS Subnet Groups page. A green success message at the top states: "Successfully created shra-db-subG. View subnet group". Below this, a table lists two subnet groups:

Name	Description	Status	VPC
default-vpc-0e756ee694d5251ca	Created from the RDS Management Console	Complete	vpc-0e756ee694d5251ca
shra-db-subg	subnet group for shra-db	Complete	vpc-0e96a0d7b539dde35

Create Database

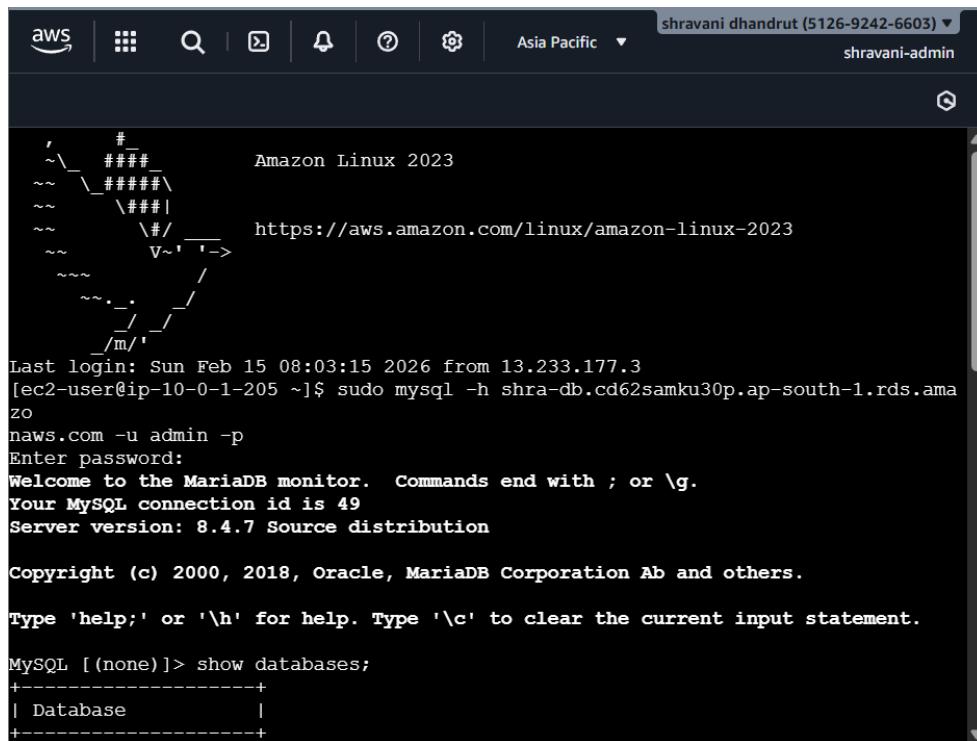
The screenshot shows the AWS RDS Database Details page for "shra-db". A green success message at the top states: "Successfully created database shra-db". Below this, a summary table provides details about the database:

DB identifier	Status	Role	Engine	Recommendations
shra-db	Backing-up	Instance	MySQL Community	
CPU	29.43%	Class	db.t4g.micro	
		Current activity	0 Connections	

The "Connectivity & security" tab is selected. It includes sections for "Connect using" (Code snippets, CloudShell, Endpoints) and "IAM Authentication". The bottom of the screen shows the AWS navigation bar and system status.

Install and open mysql in EC2

- Yum install mysql -y OR yum install mariadb
- mysql -h shra-db.cd62samku30p.ap-south-1.rds.amazonaws.com -u admin -p



A screenshot of an AWS Lambda function's terminal window. The title bar shows "shravani dhandrut (5126-9242-6603)" and "shravani-admin". The terminal output shows the connection to an Amazon Linux 2023 RDS instance:

```
'~\_\_###_#_          Amazon Linux 2023
~~ \####\_
~~ \|#
~~ \|/_--> https://aws.amazon.com/linux/amazon-linux-2023
~~ V~,'-->
~~ . /'
~~ . /' /'
~/m/'Last login: Sun Feb 15 08:03:15 2026 from 13.233.177.3
[ec2-user@ip-10-0-1-205 ~]$ sudo mysql -h shra-db.cd62samku30p.ap-south-1.rds.amazonaws.com -u admin -p
Enter password:
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MySQL connection id is 49
Server version: 8.4.7 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)]> show databases;
+-----+
| Database      |
+-----+
5 rows in set (0.006 sec)

MySQL [(none)]> use appdb;
Database changed
MySQL [appdb]> CREATE TABLE visitors(
    -> id INT AUTO_INCREMENT PRIMARY KEY,
    -> message VARCHAR(255),
    -> created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
    -> );
Query OK, 0 rows affected (0.091 sec)

MySQL [appdb]> show tables;
+-----+
| Tables_in_appdb |
+-----+
| visitors        |
+-----+
1 row in set (0.009 sec)

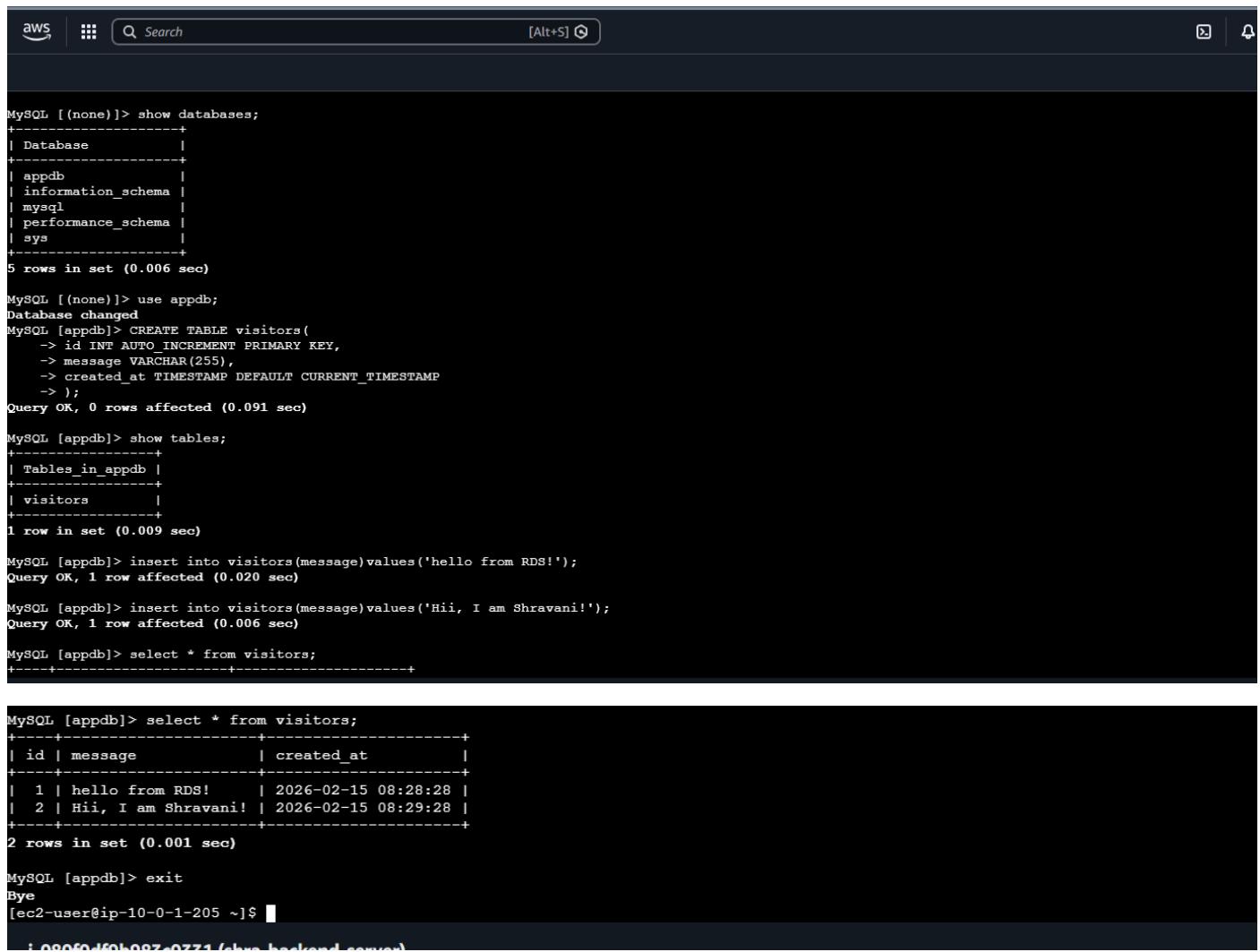
MySQL [appdb]> insert into visitors(message)values('hello from RDS!');
Query OK, 1 row affected (0.020 sec)

MySQL [appdb]> insert into visitors(message)values('Hii, I am Shravani!');
Query OK, 1 row affected (0.006 sec)

MySQL [appdb]> select * from visitors;
+----+-----+-----+
| id | message           | created_at       |
+----+-----+-----+
| 1  | hello from RDS!   | 2026-02-15 08:28:28 |
| 2  | Hii, I am Shravani! | 2026-02-15 08:29:28 |
+----+-----+-----+
2 rows in set (0.001 sec)

MySQL [appdb]> select * from visitors;
+----+-----+-----+
| id | message           | created_at       |
+----+-----+-----+
| 1  | hello from RDS!   | 2026-02-15 08:28:28 |
| 2  | Hii, I am Shravani! | 2026-02-15 08:29:28 |
+----+-----+-----+
2 rows in set (0.001 sec)

MySQL [appdb]> exit
Bye
[ec2-user@ip-10-0-1-205 ~]$
```



A screenshot of an AWS Lambda function's terminal window, continuing from the previous one. It shows the creation of a table, insertion of data, and retrieval of data from the "visitors" table.

```
MySQL [(none)]> show databases;
+-----+
| Database      |
+-----+
| appdb         |
| information_schema |
| mysql          |
| performance_schema |
| sys            |
+-----+
5 rows in set (0.006 sec)

MySQL [(none)]> use appdb;
Database changed
MySQL [appdb]> CREATE TABLE visitors(
    -> id INT AUTO_INCREMENT PRIMARY KEY,
    -> message VARCHAR(255),
    -> created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
    -> );
Query OK, 0 rows affected (0.091 sec)

MySQL [appdb]> show tables;
+-----+
| Tables_in_appdb |
+-----+
| visitors        |
+-----+
1 row in set (0.009 sec)

MySQL [appdb]> insert into visitors(message)values('hello from RDS!');
Query OK, 1 row affected (0.020 sec)

MySQL [appdb]> insert into visitors(message)values('Hii, I am Shravani!');
Query OK, 1 row affected (0.006 sec)

MySQL [appdb]> select * from visitors;
+----+-----+-----+
| id | message           | created_at       |
+----+-----+-----+
| 1  | hello from RDS!   | 2026-02-15 08:28:28 |
| 2  | Hii, I am Shravani! | 2026-02-15 08:29:28 |
+----+-----+-----+
2 rows in set (0.001 sec)

MySQL [appdb]> select * from visitors;
+----+-----+-----+
| id | message           | created_at       |
+----+-----+-----+
| 1  | hello from RDS!   | 2026-02-15 08:28:28 |
| 2  | Hii, I am Shravani! | 2026-02-15 08:29:28 |
+----+-----+-----+
2 rows in set (0.001 sec)

MySQL [appdb]> exit
Bye
[ec2-user@ip-10-0-1-205 ~]$
```

Install php in ec2

- dnf install php php-mysqlnd -y
- cd /var/www/html
- rm -f index.html
- sudo nano index.php
- write a php code inside index.php

The screenshot shows the AWS Cloud9 IDE interface. At the top, there are navigation icons (aws, grid, search, refresh, help, gear), account information (shravani dhandrut (5126-9242-6603) Asia Pacific, shravani-admin), and a status bar. The main area contains a block of PHP code. The code connects to an RDS MySQL database named 'appdb' using credentials from environment variables. It then queries the 'visitors' table and outputs the results as an HTML page. The code is as follows:

```
<?php
$host = "shra-db.cd62samku30p.ap-south-1.rds.amazonaws.com";
$user = "admin";
$password = "shra-db123";
$db = "appdb";

$conn = new mysqli($host, $user, $password, $db);

if ($conn->connect_error) {
    die("Connection failed: " . $conn->connect_error);
}

$result = $conn->query("SELECT * FROM visitors");

echo "<h1>Visitor Messages</h1>";

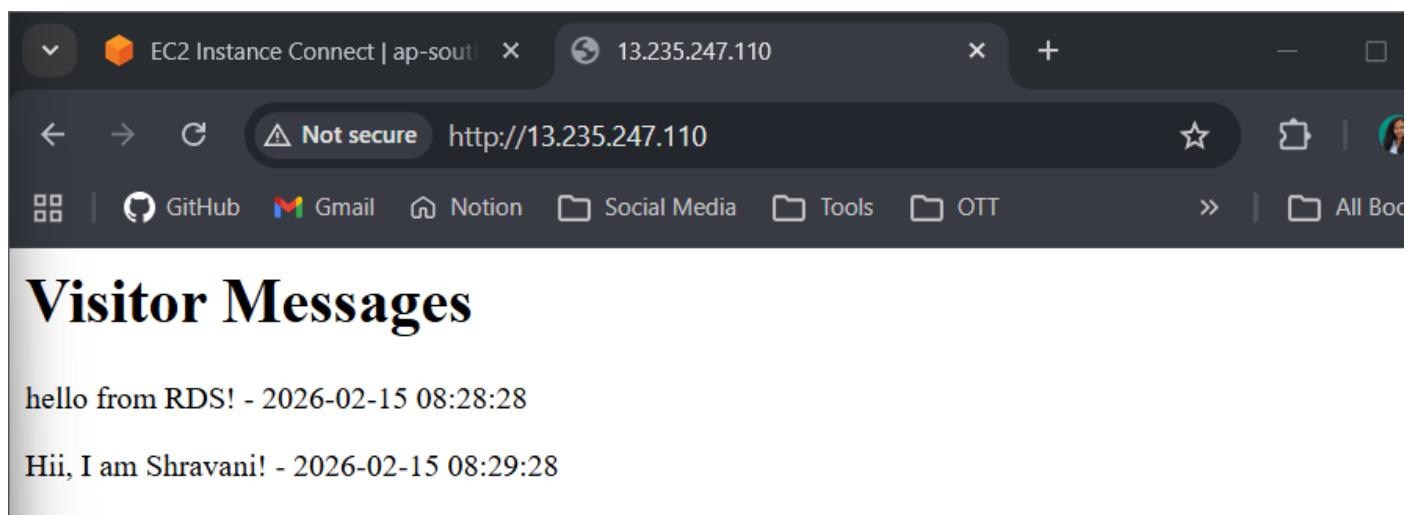
while($row = $result->fetch_assoc()) {
    echo "<p>" . $row["message"] . " - " . $row["created_at"] . "</p>";
}

$conn->close();
?>
~
~
~
~

index.php" [readonly] 22L, 482B
```

At the bottom right of the code editor, there are status indicators: 1,1 and All.

- check it works or not -> open browser -> enter EC2 public IP



Scaling

Create a second public subnet 10.0.4.0/24

The screenshot shows the AWS VPC Subnets console. On the left, there's a navigation sidebar with sections like Virtual private cloud, Security, and Subnets. The main area displays a table of subnets:

Name	Subnet ID	State	VPC	Block Public Access	IPv4 CIDR
shra-private-subnet1	subnet-0ed0ddeb19faf45c7	Available	vpc-0e96a0d7b539dde35 shra...	Off	10.0.2.0/24
shra-public-subnet2	subnet-028207067c880f92e	Available	vpc-0e96a0d7b539dde35 shra...	Off	10.0.4.0/24
shra-private-subnet2	subnet-05155238208cccd6bf	Available	vpc-0e96a0d7b539dde35 shra...	Off	10.0.3.0/24
shra-public-subnet1	subnet-0e8343503374b0b27	Available	vpc-0e96a0d7b539dde35 shra...	Off	10.0.1.0/24

Below the table, a specific subnet is selected: **shra-public-subnet2**. The details pane shows:

Details			
Subnet ID: subnet-028207067c880f92e	Subnet ARN: arn:aws:ec2:ap-south-1:512692426603:subnet/subnet-028207067c880f92e	State: Available	Block Public Access: Off
IPv4 CIDR: 10.0.4.0/24	Available IPv4 addresses: 251	IPv6 CIDR: -	IPv6 CIDR association ID: -

Enable Auto-assign IP

Route table -> Edit subnet association -> add shra-public-subnet2

The screenshot shows the AWS Route Tables console. The Subnet associations tab is selected. It lists two explicit subnet associations:

Name	Subnet ID	IPv4 CIDR	IPv6 CIDR
shra-public-subnet2	subnet-092ffa04ec46082fc	10.0.4.0/24	-
shra-public-subnet1	subnet-0e8343503374b0b27	10.0.1.0/24	-

Create ALB -> create SG for ALB ->

AWS Serverless Visitor Counter | Load balancer details | EC2 | ap-south-1 | Target group details | EC2 | ap-south-1 | Databases | Aurora and RDS | AWS | GitHub | Gmail | Notion | Social Media | Tools | OTT | Learning | Project | Duolingo | AWS | Console Home

Search [Alt+S]

AMI Catalog

Elastic Block Store

- Volumes
- Snapshots
- Lifecycle Manager

Network & Security

- Security Groups
- Elastic IPs
- Placement Groups
- Key Pairs
- Network Interfaces

Load Balancing

- Load Balancers
- Target Groups
- Trust Stores

Auto Scaling

- Auto Scaling Groups

shra-ALB

Successfully created load balancer: shra-ALB
It might take a few minutes for your load balancer to fully set up and route traffic. Targets will also take a few minutes to complete the registration process and pass initial health checks.

Details

Load balancer type Application	Status Provisioning	VPC vpc-0e96a0d7b539dde35	Load balancer IP address type IPv4
Scheme Internet-facing	Hosted zone ZP97RAFLXTNZK	Availability Zones subnet-092fa04ec46082fc ap-south-1b (aps1-az3) subnet-0e8343503374b0b27 ap-south-1a (aps1-az1)	Date created February 15, 2026, 15:29 (UTC+05:30)
Load balancer ARN arn:aws:elasticloadbalancing:ap-south-1:512692426603:loadbalancer/app/shra-ALB/d1b538270c8512690		DNS name info shra-ALB-877060537.ap-south-1.elb.amazonaws.com (A Record)	

Listeners and rules | Network mapping | Resource map | Security | Monitoring | Integrations | Attributes | Capacity | Tags

https://512692426603-7du37dbm.ap-south-1.console.aws.amazon.com/vpc/home?region=ap-south-1#VpcDetails:VpcId=vpc-0e96a0d7b539dde35

© 2026, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences

ENG IN 3:31 PM 2/15/2026

Create Golden AMI from existing EC2 instance (ec2-backend-server)

AWS Serverless Visitor Counter | shra-db - Database Details | Auto Scaling group details | EC2 | Images | EC2 | ap-south-1 | AWS | GitHub | Gmail | Notion | Social Media | Tools | OTT | Learning | Project | Duolingo | AWS | Console Home

Search [Alt+S]

EC2

AMIs

Amazon Machine Images (AMIs) (1/1)

Owned by me		Find AMI by attribute or tag		Actions		Launch instance from AMI	
Name	AMI name	AMI ID	Source	Owner	Visibility		
<input checked="" type="checkbox"/> shra-golden-AMI	ami-0718ae63b1375331e	512692426603/shra-golden-AMI	512692426603	Private			

Instances | Instance Types | Launch Templates | Spot Requests | Savings Plans

Launch template using Golden AMI

AWS Serverless Visitor Counter | shra-golden-AMI | Launch Templates | EC2 | AWS Global View | Events | Instances | AWS | GitHub | Gmail | Notion | Social Media | Tools | OTT | Learning | Project | Duolingo | AWS | Console Home

Search [Alt+S]

EC2

Launch Templates (1/1)

Search		Actions		Create launch template	
Launch Template ID	Launch Template Name	Default Version	Latest Version	Create Time	Created By
<input checked="" type="checkbox"/> lt-0649cc24a8aa29867	shra-AMI-template	1	1	2026-02-15T17:34:35.000Z	arn:aws:iam::512692426603

Dashboard | AWS Global View | Events | Instances

Create auto-scaling group

AWS Serverless Visitor Counter | Auto Scaling groups | EC2 | AWS Global View | Events | Instances | Launch Templates | AWS | GitHub | Gmail | Notion | Social Media | Tools | OTT | Learning | Project | Duolingo | AWS | Console Home

Search [Alt+S]

EC2

Auto Scaling groups (1/1)

Search your Auto Scaling groups		Last updated less than a minute ago		Actions		Create Auto Scaling group	
Name	Launch template/configuration	Instances	Status	Desired capacity	Min	Max	
<input checked="" type="checkbox"/> shra-ASG	shra-AMI-template Version Default	2	-	2	2	2	

Dashboard | AWS Global View | Events | Instances | Instance Types | Launch Templates

Instances created automatically

The screenshot shows the AWS EC2 Instances page. The left sidebar is collapsed. The main area displays a table titled "Instances (2/3) Info" with the following columns: Name, Instance ID, Instance state, Instance type, Status check, Alarm status, and Availability Zone. There are three rows:

Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone
shra-backend-server	i-080f0df9b983c0331	Running	t3.micro	3/3 checks passed	View alarms +	ap-south-1a
auto-ec2-backend-server1	i-060e3871deddb13b8	Running	t3.micro	3/3 checks passed	View alarms +	ap-south-1a
auto-ec2-backend-server2	i-0199ed9f5ecd2658d	Running	t3.micro	3/3 checks passed	View alarms +	ap-south-1b

Go to RDS-> security group -> edit inbound rule -> add(custom=auto-scaling SG)

Check if it is working properly using ALB dns name

The screenshot shows a browser window with multiple tabs open. The active tab is "AWS Serverless Visitor Counter" at the URL <http://shra-alb-877060537.ap-south-1.elb.amazonaws.com>. The page displays a simple "Visitor Messages" section with two entries:

- hello from RDS! - 2026-02-15 08:28:28
- Hii, I am Shravani! - 2026-02-15 08:29:28

Visitor Messages

hello from RDS! - 2026-02-15 08:28:28

Hii, I am Shravani! - 2026-02-15 08:29:28