

## How to set up 1 master, 1 read replica using AWS RDS



### Requirements:

- 4 subnets (1 for frontend and bastion-host, 1 for backend and 2 for DB instances contains 1 primary and 1 read replica) – 4 different AZs
- Backend: write data record into DB instance per minute (using bash script and crontab)
- Frontend: allow HTTP from fixed IP addresses from customers, only read data in DB Read replica (frontend using PHP)
- Subnet, security group only allow least privileges

### References:

- [Install PHP and config MySQL](#)

### Step by step:

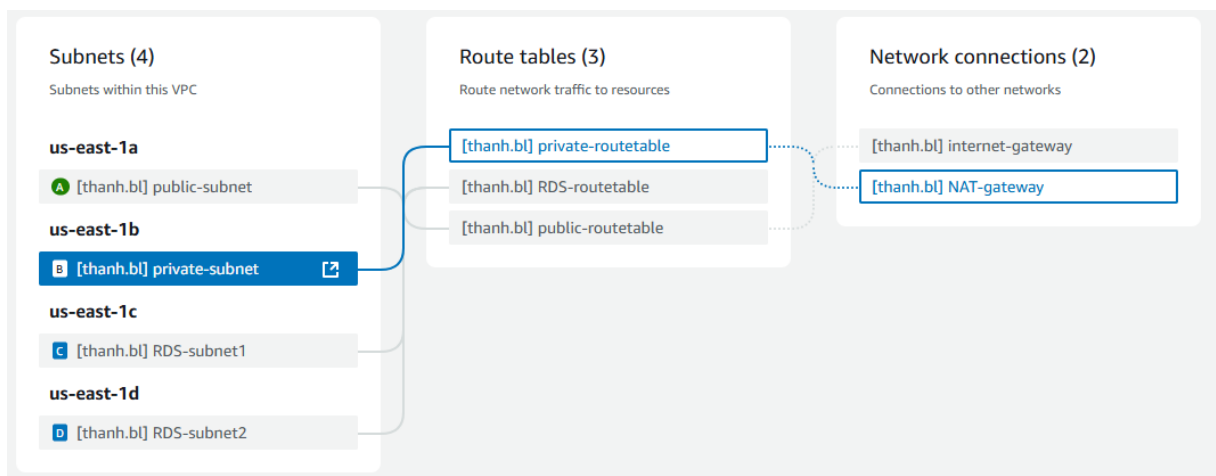
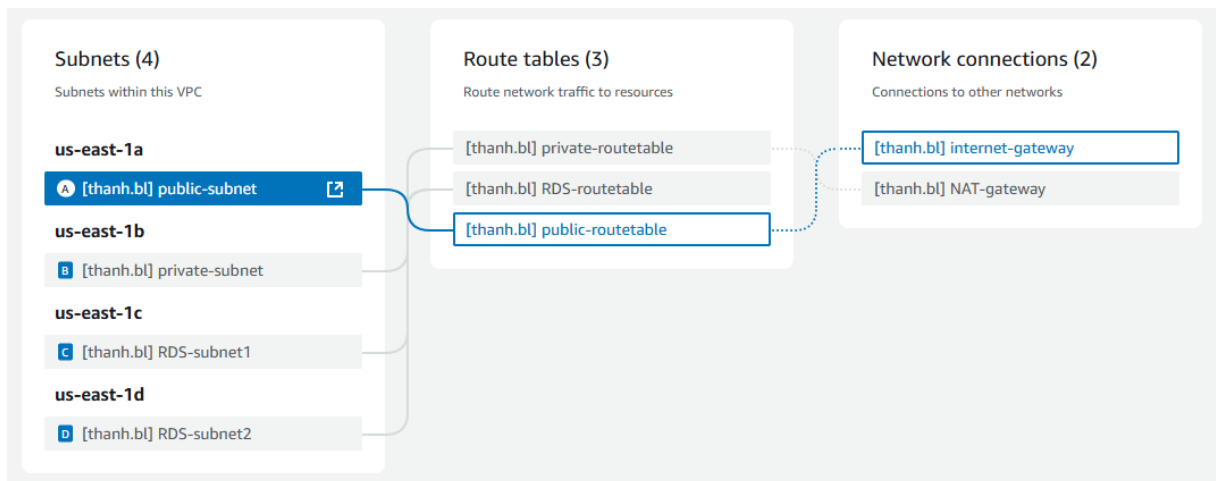
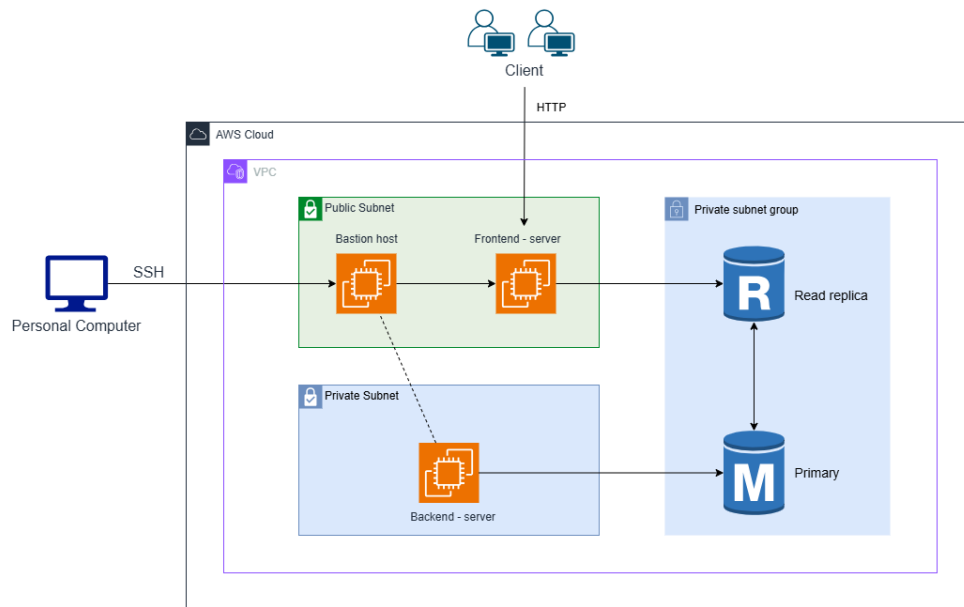
- **Step 1: Infrastructure config**

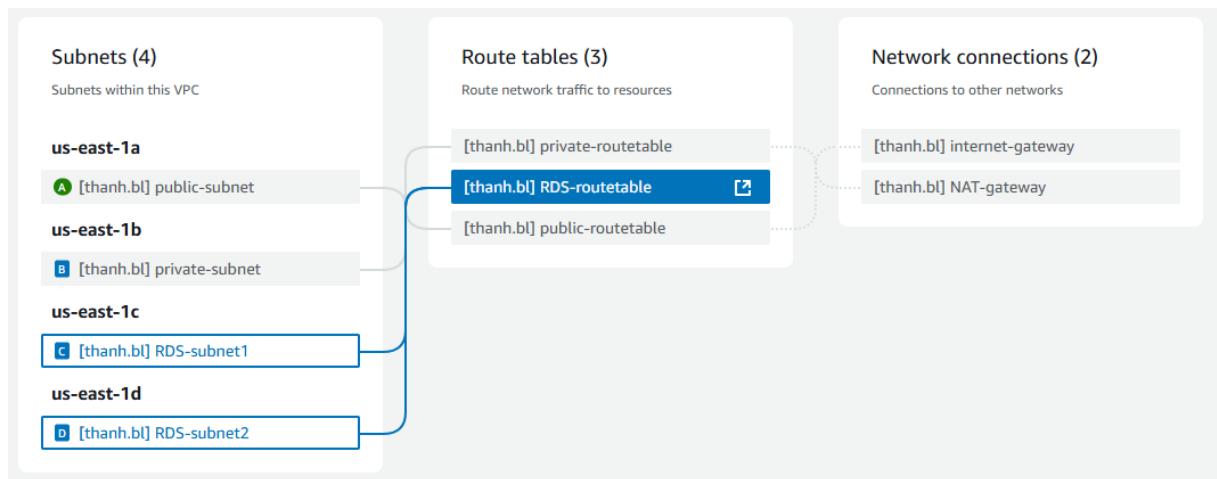
This picture below describe overview of our infrastructure:

VPC overview:

- 1 internet gateway
- 1 NAT gateway
- 4 subnets (3 route tables) in 4 different Azs (1 for bastion and frontend, 1 for backend and 2 for subnet group for RDS instances)

- 4 security groups





Set up least privileges for security group:

- **Red:** delete after use
- **Green:** keep config

SG permission (allow)	Inbound	Outbound
Bastion	Allow SSH (local)	FE, BE
Frontend – server	Allow SSH (bastion), HTTP (IP from customers)	RDS (9306)
Backend – server	Allow SSH (bastion)	RDS (9306)
RDS instances	9306 (BE, FE)	None

Subnets route table:

Network Access Control List:

- **Step 2: Set up resources**

This step, we will install necessary software:

- **Bastion host:** None
  - **Frontend – server:** PHP (show data in frontend) and MySQL – client (for access data in read replica)
  - **Backend – server:** MySQL – client (access and write data to primary)
- **Step 3: Config**









- Change default SSH port at all instances: 2222

Use userdata to change SSH port when first create instance.

- Create a subnet group for primary RDS database instance
- Create database instance (primary)

Choose database engine

Engine type [Info](#)

<input type="radio"/> Aurora (MySQL Compatible) 	<input type="radio"/> Aurora (PostgreSQL Compatible) 
<input checked="" type="radio"/> MySQL 	<input type="radio"/> MariaDB 
<input type="radio"/> PostgreSQL 	<input type="radio"/> Oracle 
<input type="radio"/> Microsoft SQL Server 	<input type="radio"/> IBM Db2 

Create a admin user:

#### ▼ Credentials Settings

Master username [Info](#)

Type a login ID for the master user of your DB cluster.

1 to 16 alphanumeric characters. The first character must be a letter.

#### Credentials management

You can use AWS Secrets Manager or manage your master user credentials.

☐ **Managed in AWS Secrets Manager - most secure**  
 RDS generates a password for you and manages it throughout its lifecycle using AWS Secrets Manager.

☒ **Self managed**  
 Create your own password or have RDS create a password that you manage.

☐ **Auto generate password**

Amazon RDS can generate a password for you, or you can specify your own password.

Master password [Info](#)

#### Password strength

Minimum constraints: At least 8 printable ASCII characters. Can't contain any of the following symbols: / ' " @

Confirm master password [Info](#)

Change default MySQL port: 9306 (in /Connectivity/Additional configuration):

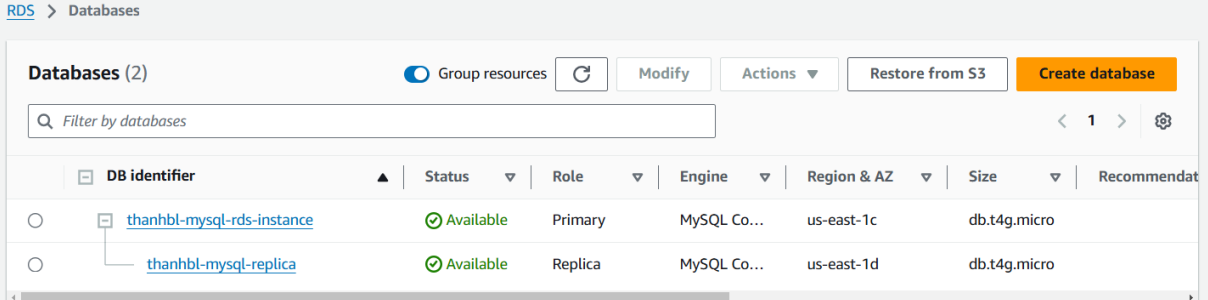
#### ▼ Additional configuration

Database port [Info](#)

TCP/IP port that the database will use for application connections.

9306

- Access primary endpoint using admin user and create a new DB user for frontend – server to read data in read replica
- Create a new replica for DB instance:



DB identifier	Status	Role	Engine	Region & AZ	Size	Recommendat
<a href="#">thanhbl-mysql-rds-instance</a>	Available	Primary	MySQL Co...	us-east-1c	db.t4g.micro	
<a href="#">thanhbl-mysql-replica</a>	Available	Replica	MySQL Co...	us-east-1d	db.t4g.micro	

- Write a script to write record into *primary RDS* instance and check if record inserted successfully

In this script, we will write data to primary instance so we will use *admin* user:

```
#!/bin/bash
DB_USER="admin"
DB_PASSWORD="admin212"
DB_NAME="todo_list"
TABLE_NAME="todo_list"
DB_HOST="thanhbl-mysql-rds-instance.ceahr1jdchqz.us-east-1.rds.amazonaws.com"
DB_PORT="9306"
CURRENT_TIMESTAMP=$(date +"%Y-%m-%d %H:%M:%S")
CONTENT="This database record insert in $CURRENT_TIMESTAMP"
mysql -h "$DB_HOST" -P "$DB_PORT" -u "$DB_USER" -p"$DB_PASSWORD" -D "$DB_NAME" -e
"INSERT INTO $TABLE_NAME (content) VALUES ('$CONTENT');"
```

Check if data inserted successfully:

```
mysql> SELECT * FROM todo_list;
```

item_id	content
1	This is first record ever!
2	This database record insert in 2024-11-06 13:00:49

```
2 rows in set (0.00 sec)
```

Data inserted successfully!

- Config PHP web to read data on *read replica* instance with *client* user:

```
<?php
$user = "client";
$password = "client212";
$database = "todo_list";
$table = "todo_list";
$host = "thanhbl-mysql-replica.ceahrljdchqz.us-east-1.rds.amazonaws.com";
try {
    $db = new PDO("mysql:host=$host;port=9306;dbname=$database", $user, $password);
    echo "<h2>MySQL record inserted!</h2><ol>";
    foreach($db->query("SELECT content FROM $table") as $row) {
        echo "<li>" . $row['content'] . "</li>";
    }
    echo "</ol>";
} catch (PDOException $e) {
    print "Error!: " . $e->getMessage() . "<br/>";
    die();
}
?>
```

#### - Step 4: Check

Access frontend server we will get result like:

We can see list of record by *client* user!

## MySQL record inserted!

1. This is first record ever!
2. This database record insert in 2024-11-06 13:00:49
3. This database record insert in 2024-11-06 13:03:01
4. This database record insert in 2024-11-06 13:04:01
5. This database record insert in 2024-11-06 13:05:01
6. This database record insert in 2024-11-06 13:06:01
7. This database record insert in 2024-11-06 13:07:01
8. This database record insert in 2024-11-06 13:08:01
9. This database record insert in 2024-11-06 13:09:01
10. This database record insert in 2024-11-06 13:10:01
11. This database record insert in 2024-11-06 13:11:01
12. This database record insert in 2024-11-06 13:12:01

### Check if data in RDS database instance encrypted:

```
PS C:\> aws rds describe-db-instances --db-instance-identifier thanhbl-mysql-rds-instance --query "DBInstances[*].StorageEncrypted"
[
  true
]

PS C:\> aws rds describe-db-instances --db-instance-identifier thanhbl-mysql-replica --query "DBInstances[*].StorageEncrypted"
[
  true
]
```