

Build Your DB Server and Interact With Your DB Using an App

Here's what happened:

I'm learning about Amazon RDS, a service that helps me set up and manage a relational database in the cloud.

Here's what I'll be doing:

1. Launching a DB instance: I'll create a database instance using Amazon RDS, which will be highly available (meaning it's always accessible).
2. Configuring connections: I'll set up the database to allow connections from my web server.
3. Testing the database: I'll open a web application and interact with my database to see it in action.

What is Amazon RDS?

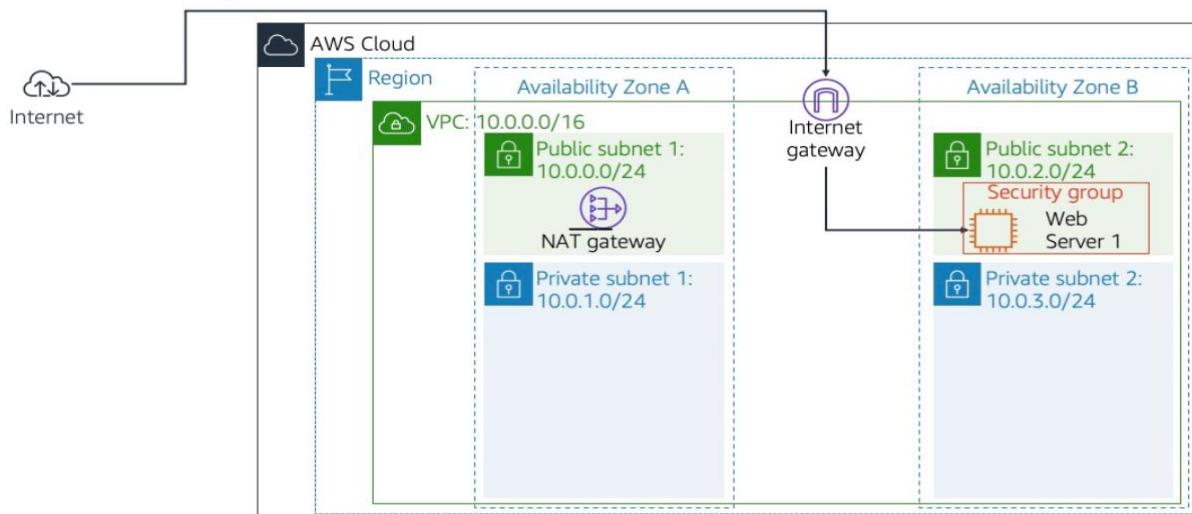
Amazon RDS is a service that makes it easy to set up and manage a relational database in the cloud. It takes care of time-consuming tasks, so I can focus on my applications and business.

What can I do with Amazon RDS?

I can choose from six popular database engines, including MySQL, PostgreSQL, and Oracle, and scale my database as my needs grow.

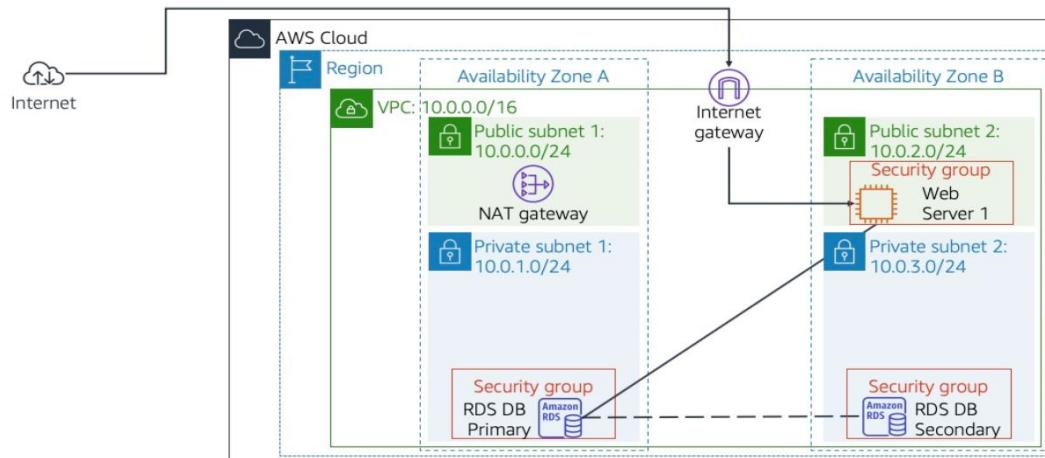
Scenario

You start with the following infrastructure:



At the end of the lab, this is the infrastructure:

MOKGADI SELEPE



1: Create a Security Group for the RDS DB Instance

Screenshot 1: AWS VPC Dashboard

This screenshot shows the AWS VPC dashboard. The left sidebar includes sections for Virtual private cloud, Security, and PrivateLink and Lattice. The main area displays resources by region, including VPCs, Subnets, Route Tables, Internet Gateways, Egress-only Internet Gateways, Carrier Gateways, DHCP Option Sets, Elastic IPs, Managed Prefix Lists, NAT Gateways, Peering Connections, and Route Servers. A prominent 'Create VPC' button is at the top.

Screenshot 2: AWS Security Groups

This screenshot shows the AWS Security Groups page. The left sidebar lists Virtual private cloud, Security (Network ACLs, Security groups), and PrivateLink and Lattice. The main area displays a table of existing security groups:

Name	Security group ID	Security group name	VPC ID	Description
sg-061db649c268060bd	default	vpc-02f6017a8cbf77f0d	default VPC securit	
sg-0acfd3608b90eb26e	default	vpc-03261aeaf566b8000a	default VPC securit	
Web Security Group	sg-0686e6c93f29387ae	vpc-03261aeaf566b8000a	Enable HTTP acces	

A 'Create security group' button is located at the top right of the table.

MOKGADI SELEPE

The screenshots show the 'Create security group' wizard in the AWS VPC console. The first two steps are identical, while the third step shows the successful creation of the security group.

Step 1: Basic details

- Security group name:** DB Security Group
- Description:** Permit access from Web Security Group
- VPC:** vpc-03261aef566b8000a (Lab VPC)

Step 2: Inbound rules

This security group has no inbound rules.

Step 3: Outbound rules

Type	Protocol	Port range	Source	Description - optional
MySQL/Aurora	TCP	3306	Custom	sg-0686e6c93f29387ae
				sg-0686e6c93f29387ae

Success Message: Security group (sg-0b7eeb3f82b9d10e7 | DB Security Group) was created successfully.

Here's what happened:

MOKGADI SELEPE

I created a security group to allow my web server to access my database.

Here's what I did:

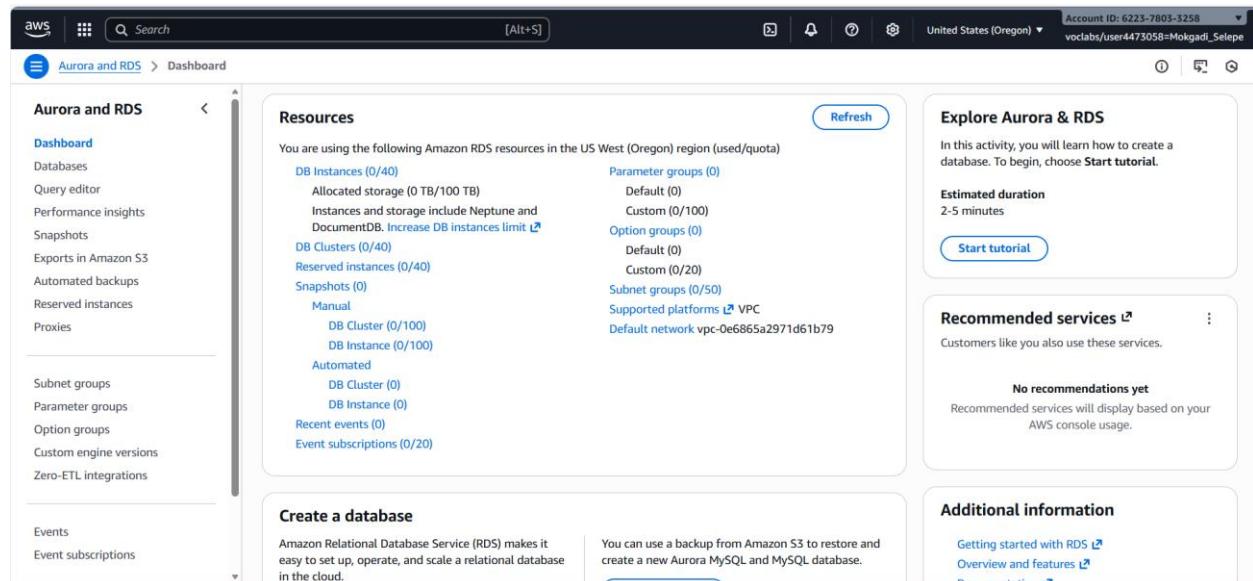
1. Created a security group: I made a new security group called "DB Security Group" to control access to my database.
2. Added a rule: I added a rule to allow incoming database requests from my web server's security group (Web Security Group).
3. Configured the rule: I set the rule to allow MySQL/Aurora traffic on port 3306 from the Web Security Group.

What does this mean?

This security group will allow my web server to connect to my database, but block connections from other sources. It's like setting up a firewall to protect my database.

I'll use this security group when I create my database instance.

2: Create a DB Subnet Group



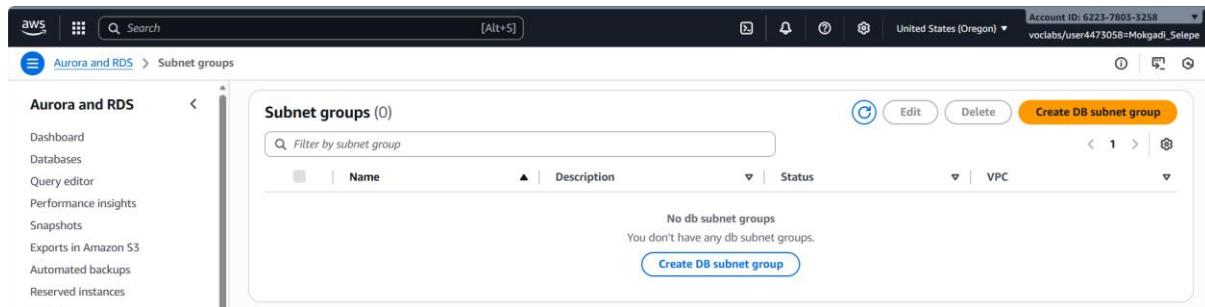
The screenshot shows the AWS Aurora and RDS Dashboard. On the left sidebar, under the 'Aurora and RDS' section, the 'Dashboard' option is selected. Other options include Databases, Query editor, Performance insights, Snapshots, Exports in Amazon S3, Automated backups, Reserved instances, Proxies, Subnet groups, Parameter groups, Option groups, Custom engine versions, Zero-ETL integrations, Events, and Event subscriptions. The main content area is titled 'Resources' and displays the following information:

- You are using the following Amazon RDS resources in the US West (Oregon) region (used/quota):
- DB Instances (0/40)**: Allocated storage (0 TB/100 TB). Instances and storage include Neptune and DocumentDB. [Increase DB instances limit](#).
- DB Clusters (0/40)**: Reserved instances (0/40).
- Snapshots (0)**: Manual (DB Cluster (0/100), DB Instance (0/100)), Automated (DB Cluster (0), DB Instance (0)). Recent events (0). Event subscriptions (0/20).
- Parameter groups (0)**: Default (0), Custom (0/100).
- Option groups (0)**: Default (0), Custom (0/20).
- Subnet groups (0/50)**: Supported platforms VPC. Default network vpc-0e6865a2971d61b79.

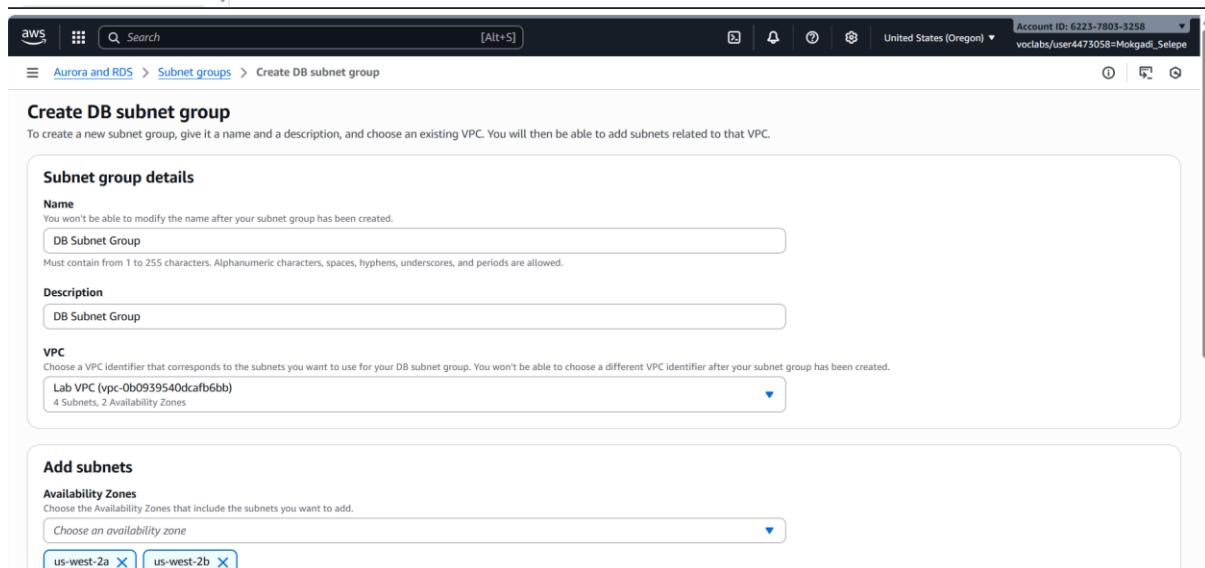
On the right side of the dashboard, there are three boxes:

- Explore Aurora & RDS**: In this activity, you will learn how to create a database. To begin, choose [Start tutorial](#). Estimated duration: 2-5 minutes. [Start tutorial](#).
- Recommended services**: Customers like you also use these services. No recommendations yet. Recommended services will display based on your AWS console usage.
- Additional information**: Getting started with RDS, Overview and features, Documentation.

MOKGADI SELEPE

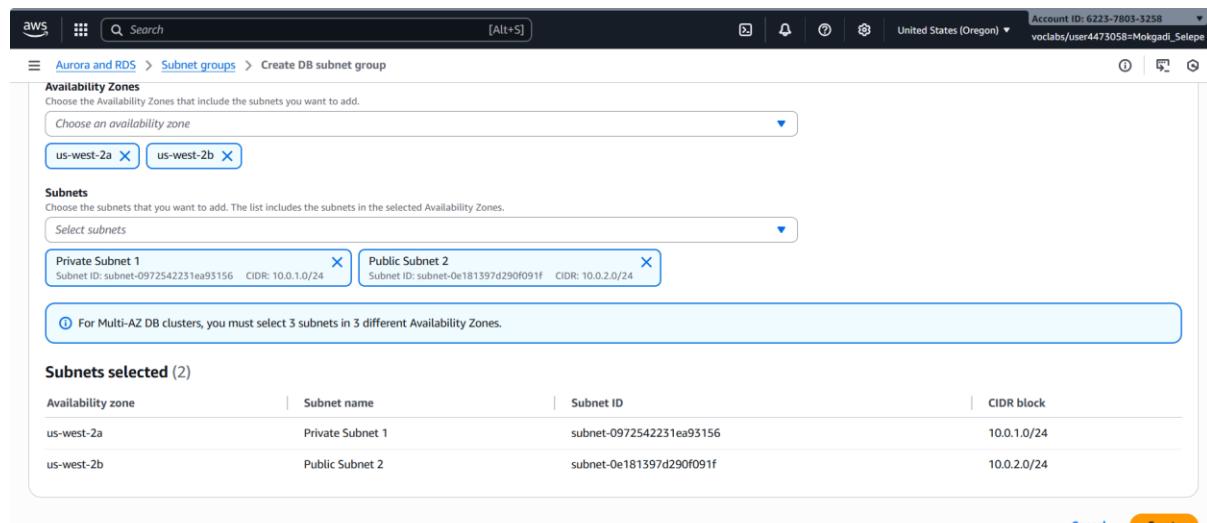


The screenshot shows the AWS Aurora and RDS Subnet groups page. The left sidebar includes links for Dashboard, Databases, Query editor, Performance insights, Snapshots, Exports in Amazon S3, Automated backups, Reserved instances, Proxies, Subnet groups (selected), Parameter groups, Option groups, Custom engine versions, Zero-ETL integrations, Events, and Event subscriptions. The main content area displays a table titled "Subnet groups (0)" with columns for Name, Description, Status, and VPC. A message states "No db subnet groups" and "You don't have any db subnet groups." A prominent orange "Create DB subnet group" button is at the bottom right.

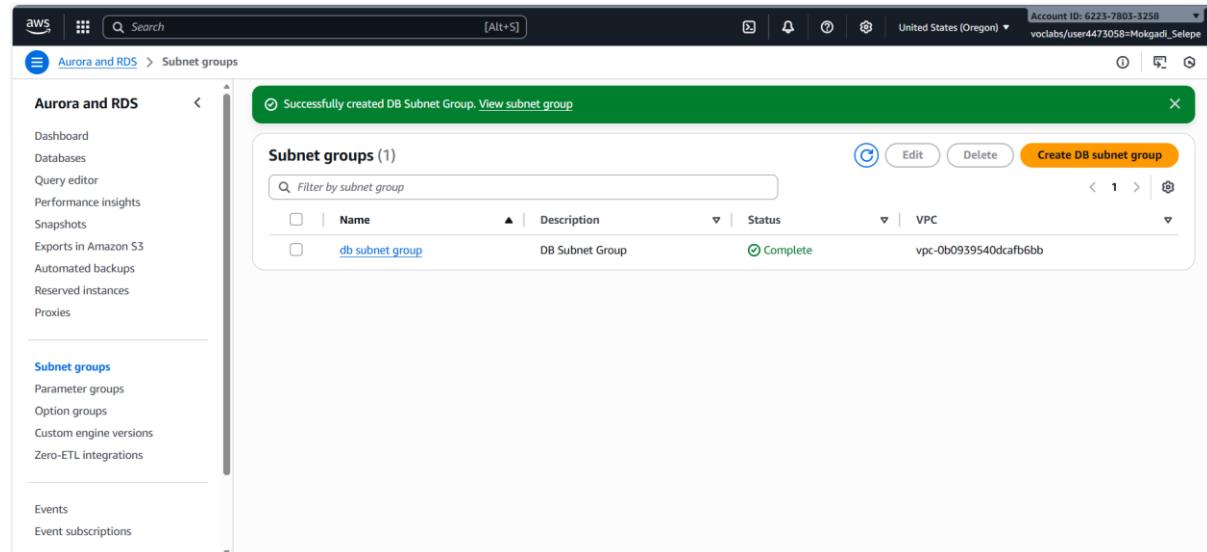


The screenshot shows the "Create DB subnet group" wizard, Step 1: Subnet group details. It includes fields for Name (DB Subnet Group), Description (DB Subnet Group), and VPC (Lab VPC (vpc-0b0939540dcafb6bb) selected). The "Add subnets" section shows Availability Zones (us-west-2a and us-west-2b selected).

MOKGADI SELEPE



Availability zone	Subnet name	Subnet ID	CIDR block
us-west-2a	Private Subnet 1	subnet-0972542231ea93156	10.0.1.0/24
us-west-2b	Public Subnet 2	subnet-0e181397d290f091f	10.0.2.0/24



I've created a special group called a "DB Subnet Group" for my database.

This group tells Amazon's database service (RDS) which parts of my network (called "subnets") it can use to store my database.

Here's what I did:

1. I chose two private areas of my network (called "subnets") in different locations (called "Availability Zones").
2. I put these subnets into a group called "DB Subnet Group".
3. I told RDS that it's okay to use these subnets for my database.

Why is this important?

- It helps keep my database safe by putting it in a secure part of my network.
- It makes sure my database is available in multiple locations, so if one location has a problem, my database is still available.

MOKGADI SELEPE

Think of it like choosing two safe and separate locations for my valuable data!

3: Create an Amazon RDS DB Instance

The screenshots show the process of creating a new database instance:

- Screenshot 1: Choose a database creation method**

Shows two options:
 - Standard create**: You set all of the configuration options, including ones for availability, security, backups, and maintenance.
 - Easy create**: Use recommended best-practice configurations. Some configuration options can be changed after the database is created.
- Screenshot 2: Engine options**

Shows engine type selection:
 - Aurora (MySQL Compatible)
 - Aurora (PostgreSQL Compatible)**
 - MySQL
 - PostgreSQL
 - MariaDB
 - Oracle
 - Microsoft SQL Server
 - IBM Db2
- Screenshot 3: Suggested add-ons for lab-db**

Shows suggested add-ons:
 - Create an ElastiCache cluster from RDS using your DB settings
 - Use RDS Proxy
- Screenshot 4: Creating database lab-db**

Shows the database creation progress table:

DB identifier	Status	Role	Engine	Region ...	Size
lab-db	Creating	Instance	MySQL Co...	us-west-2b	db.t3.medium

MOKGADI SELEPE

The top screenshot shows the 'Databases' page with a success message: 'Successfully created database lab-db'. The database 'lab-db' is listed with details: DB identifier 'lab-db', Status 'Available', Role 'Instance', Engine 'MySQL Community', Region & AZ 'us-west-2b', and Size 'db.t3.medium'. The bottom screenshot shows the 'lab-db' instance details page under the 'Connectivity & security' tab. It displays the endpoint URL 'lab-db.clsifl1q98y6.us-west-2.rds.amazonaws.com', port '3306', availability zone 'us-west-2b', VPC 'Lab VPC (vpc-0b0939540dcfb6bb)', subnet group 'db subnet group', and security group 'DB Security Group (sg-0bdd2369596b91f0) Active'. The status is 'Available'.

I've created a MySQL database instance on Amazon RDS that's designed for high availability and durability. Here's what happened:

I configured a Multi-AZ deployment, which means Amazon RDS creates a primary database instance and replicates it to a standby instance in a different Availability Zone (AZ).

I chose the latest MySQL version and a Dev/Test template, and set up my database instance with a specific identifier, username, and password. I selected a db.t3.medium instance class and General Purpose SSD storage, and configured my Virtual Private Cloud (VPC) and security group settings.

I created the database, and it's currently deploying in two different Availability Zones (AZs). Once it's available, I'll get an endpoint URL that I can use to connect to my database.

Here's what I did in simple terms:

1. I created a database instance with a primary and standby instance in different AZs.

MOKGADI SELEPE

2. I configured the database settings, like instance type and storage.
3. I created the database, and it's deploying now.
4. I'll get an endpoint URL to connect to my database once it's available.

Think of it like setting up a highly available database that's safe and secure!

4: Interact with Your Database

The screenshot shows the AWS Cloud9 IDE interface. At the top, there are navigation buttons: AWS, Start Lab, End Lab, AWS Details, Details, Reset, Submit, Submission Report, and Grades. The AWS Details button is highlighted. The status bar indicates "Used \$0.3 of \$10" and "02:57".

Cloud Access panel:

- AWS CLI: Show
- Cloud Labs: Remaining session time: 02:56:19(177 minutes)
Session started at: 2025-11-09T04:54:35-0800
Session to end at: 2025-11-09T10:18:01-0800
- Accumulated lab time: 05:27:00 (327 minutes)
- ips -- public:35.93.203.246, private:10.0.2.61
- SSH key Show Download PEM Download PPK
- AWS SSO Download URL
- awsstudentAccessSecretKey Access Key: AKIAZB2E7EBVNPWWY
Access Key: OfyWC/TurjqqOrJwzkSZG+haploV
- WebServer 35.93.203.246

Task 4: Interact with Your Database

In this task, you will open a web application running on your web server and configure it to use the database.

33. Wait until the **Status** changes to **Modifying** or **Available**.
34. Scroll down to the **Connectivity & Security** section and copy the **Endpoint** field. It will look similar to: `lab-db.cggq8lnxvnv.us-west-2.rds.amazonaws.com`
35. Paste the Endpoint value into a text editor. You will use it later in the lab.

Browsing the Web Application:

- Tab 1 (Top):** Shows the AWS logo, Load Test, and RDS buttons. A red arrow points to the RDS button.
- Tab 2 (Middle):** Shows the AWS logo, Load Test, and RDS buttons. Below is a table:

Meta-Data	Value
InstanceId	i-07064a6fa36f93e61
Availability Zone	us-west-2b

Current CPU Load: 0%
- Tab 3 (Bottom):** Shows the AWS logo, Load Test, and RDS buttons. A message says "Under High CPU Load! (auto refresh in 5 seconds)". Below is a message: "Current CPU Load: 75%".

MOKGADI SELEPE

The image consists of three vertically stacked screenshots of a web browser. Each screenshot shows a form for configuring a database connection.

Screenshot 1: A blank form with four input fields: Endpoint, Database, Username, and Password, followed by a Submit button.

Screenshot 2: The same form, but the Endpoint field contains the value `http://lab-db.clsif1q98y6.us-west-2.rds.amazonaws.com`. The other fields (Database, Username, Password) are also populated, and the Password field has a red asterisk icon indicating it's required. The Submit button is visible at the bottom.

Screenshot 3: A different page titled "Address Book". It displays a table of contacts with columns: Last name, First name, Phone, Email, and Admin. Two rows of data are shown: one for Doe (Jane, 010-110-1101, janed@someotheraddress.org) and one for Johnson (Roberto, 123-456-7890, robertoj@someaddress.com). Each row has "Edit" and "Remove" links under the Admin column.

I've interacted with my database by opening a web application on my web server and configuring it to use the database.

Here's what I did:

- I copied the WebServer IP address and opened the web application in a new browser tab.
- I clicked the RDS link and configured the application with my database's endpoint, username, and password.
- The application connected to the database and started storing data in it.
- I tested the application by adding, editing, and removing contacts, and the data was saved to the database.
- The database is replicating the data to a second Availability Zone, ensuring it's safe and available.

MOKGADI SELEPE

In simple terms, I've set up a web application that uses my RDS database to store and retrieve data, and it's working as expected!
