179-[JAWS]-Activity - Migrate to Amazon RDS

Migrating to Amazon RDS

Lab overview

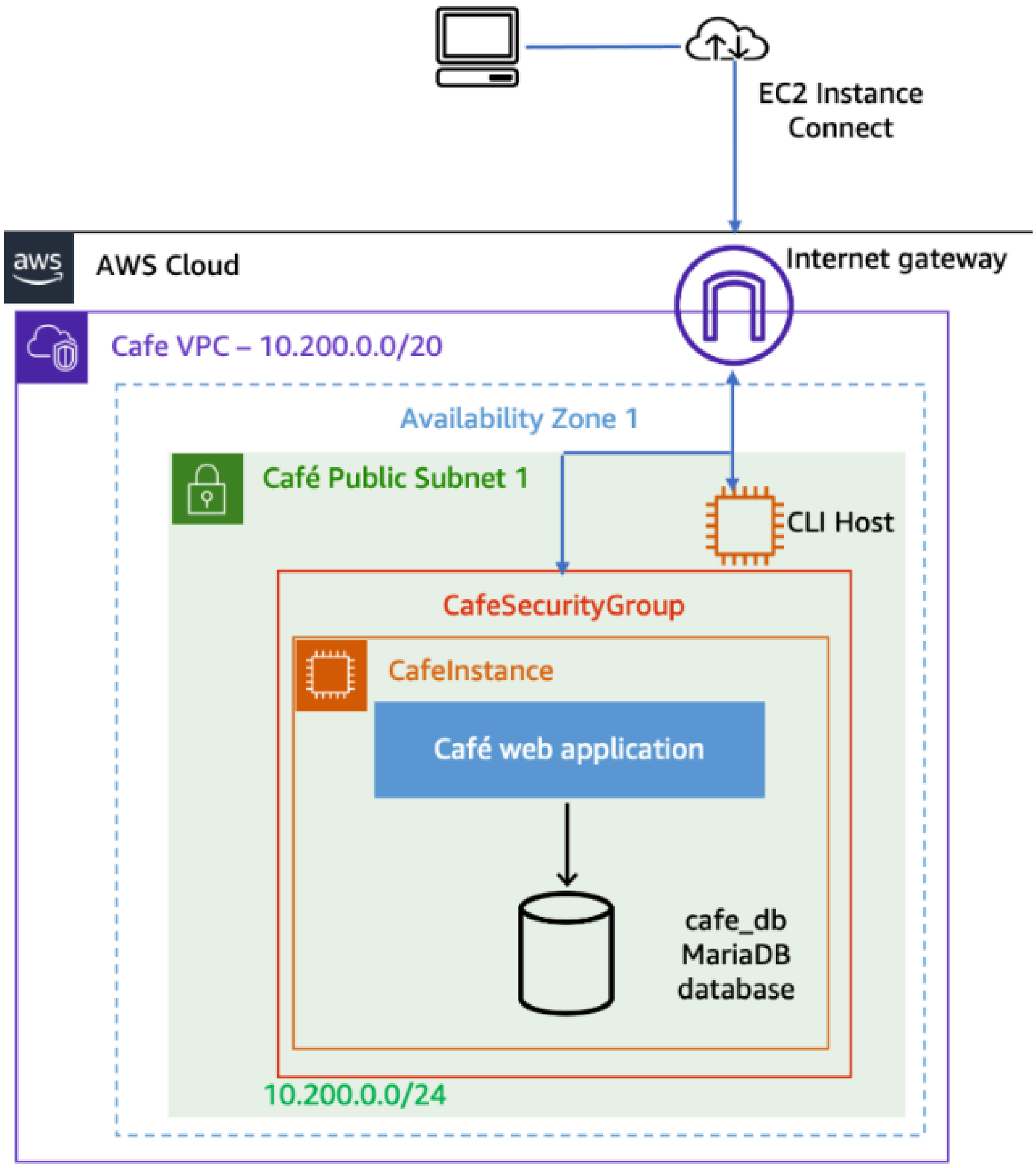
In this lab, you migrate the café web application to use a fully managed Amazon Relational Database Service (Amazon RDS) database (DB) instance instead of a local database instance.

You begin by generating some data on the existing database. This data is migrated to the new Amazon RDS instance.

During the migration process, you build the required components, including two private subnets in different Availability Zones, a security group for the database instance, and the RDS DB instance itself. After the database has been migrated, you reconfigure the café application to use the Amazon RDS instance instead of a local database.

Starting architecture

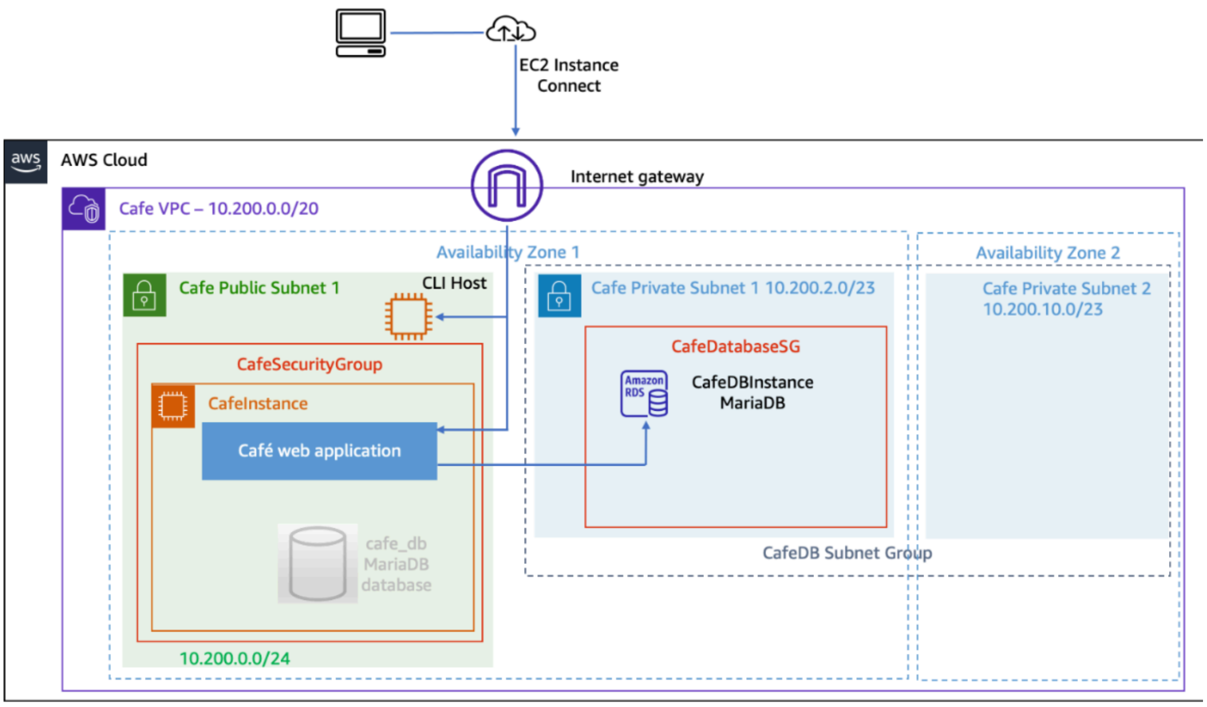
The following diagram illustrates the topology of the café web application runtime environment before the migration. The application database runs in an Amazon Elastic Compute Cloud (Amazon EC2) Linux, Apache, MySQL, and PHP (LAMP) instance along with the application code. The instance has a T3 small instance type and runs in a public subnet so that internet clients can access the website. A CLI Host instance resides in the same subnet to facilitate the administration of the instance by using the AWS Command Line Interface (AWS CLI).



Final architecture

The following diagram illustrates the topology of the café web application runtime environment after the migration.

You migrate the local café database to an Amazon RDS database that resides outside the instance. The Amazon RDS database is deployed in the same virtual private cloud (VPC) as the instance.



Objectives

After completing this lab, you will be able to do the following:

- Create an Amazon RDS MariaDB instance by using the AWS CLI.
- Migrate data from a MariaDB database on an EC2 instance to an Amazon RDS MariaDB instance.
- Monitor the Amazon RDS instance by using Amazon CloudWatch metrics.

Duration

This lab requires approximately **60 minutes** to complete.

Accessing the AWS Management Console

1. At the top of these instructions, choose Start Lab to launch your lab.

   A **Start Lab** panel opens displaying the lab status.

2. Wait until the message "Lab status: ready" appears, and then choose **X** to close the **Start Lab** panel.
3. At the top of these instructions, choose AWS to open the AWS Management Console on a new browser tab. The system automatically signs you in.

   Tip If a new browser tab does not open, a banner or icon at the top of your browser will indicate that your browser is preventing the site from opening pop-up windows. Choose the banner or icon, and choose **Allow pop-ups**.

4. Arrange the AWS Management Console so that it appears alongside these instructions. Ideally, you should be able to see both browser tabs at the same time to follow the lab steps.

   Leave this browser tab open. You return to it later in this lab.

Task 1: Generating order data on the café website

In this task, you browse the café website and place a few orders that are stored in the existing database. Placing orders creates data for the application before the application is migrated to new Amazon RDS instance.

Now, you open the café web application, and place some orders.

5. To place orders, go to the top of these instructions, choose Details, and then choose Show.
6. Copy the **CafeInstanceURL** value, and paste it into a new browser window.

   Note: The **CafeinstanceURL** value looks similar to 34.55.102.33/cafe.

7. Copy the other values from the table, and paste them into a text editor to use throughout the lab.
8. On the cafe website, choose **Menu**, add at least one of each item to your order, and then choose **Submit Order**.
9. Go to the **Order History** page, and record the number of orders that you placed. Later in this lab, you can compare this number with the number of orders in the migrated database.

Task 2: Creating an Amazon RDS instance by using the AWS CLI

In this task, you create an Amazon RDS instance by using the AWS CLI. To begin, you use EC2 Instance Connect to securely connect to the CLI Host instance already provisioned for you. This instance has the AWS CLI installed on it as part of provisioning. You then run AWS CLI commands to do the following:

- Configure the AWS CLI.
- Create the following prerequisite components required to build the Amazon RDS instance:
  - A security group firewall for the Amazon RDS instance
  - Two private subnets and a database subnet group
- Create the Amazon RDS MariaDB instance.

Task 2.1: Connecting to the CLI Host instance

In this task, you use EC2 Instance Connect to connect to the CLI Host EC2 instance. You use this instance to run AWS CLI commands.

10. On the **AWS Management Console**, in the **Search** bar, enter and choose `EC2` to open the **EC2 Management Console**.
11. In the navigation pane, choose **Instances**.
12. From the list of instances, select the ☑ **CLI Host** instance.
13. Choose **Connect**.
14. On the **EC2 Instance Connect** tab, choose **Connect**.

    Note: If you prefer to use an SSH client to connect to the EC2 instance, see the guidance to Connect to Your Linux Instance.

    Now that you are connected to the CLI Host instance, you can configure and use the AWS CLI to call AWS services.

Task 2.2: Configuring the AWS CLI

In this task, you configure the AWS CLI by providing the configuration parameters that were made available to you when the lab was provisioned. After configuration, you run AWS CLI commands to interact with AWS services.

15. To set up the AWS CLI profile with credentials, in the EC2 Instance Connect terminal, run the following command:

    ```
    aws configure
    ```

    Tip: In EC2 Instance Connect, you might need to use the context (right-click) menu to paste values into the terminal.
16. When prompted, enter the following information:

    Note: Use the values that you copied into the text editor earlier in the lab, and paste them into the terminal window prompt.

    - **AWS Access Key ID:** Enter the value for **AccessKey**.
    - **AWS Secret Access Key:** Enter the value for **SecretKey**.
    - **Default region name:** Enter the value for **LabRegion**.
    - **Default output format:** Enter `json`.

    Now you are ready to run AWS CLI commands to interact with AWS services.

Task 2.3: Creating prerequisite components

In this task, you create the prerequisite infrastructure components for the Amazon RDS instance. Specifically, you create the following components that are shown in the final architecture diagram:

- CafeDatabaseSG (Security group for the Amazon RDS database)
- CafeDB Private Subnet 1
- CafeDB Private Subnet 2
- CafeDB Subnet Group (Database subnet group)

Now, you run AWS CLI commands in the EC2 Instance Connect terminal.

First, you create the CafeDatabaseSG security group. This security group is used to protect the Amazon RDS instance. It will have an inbound rule that allows only MySQL requests (using the default TCP protocol and port 3306) from instances that are associated with the CafeSecurityGroup. This rule ensures that only the CafeInstance is able to access the database.

17. To create the security group, run the following command. In the command, replace *<CafeInstance VPC ID>* with the **CafeVpcID** value that you recorded earlier:

```
aws ec2 create-security-group \
--group-name CafeDatabaseSG \
--description "Security group for Cafe database" \
--vpc-id <CafeInstance VPC ID>
```

18. After the command completes, copy and paste the **GroupId** from the output to a text editor to use later. You use this value as the group ID for the CafeDatabaseSG Group ID in this lab.

    Next, you create the inbound rule for the security group.

19. To create the inbound rule, run the following command. In the command, replace *<CafeDatabaseSG Group ID>* with the **GroupId** value that you recorded in the previous step, and replace *<CafeSecurityGroup Group ID>* with the **CafeSecurityGroupID** value that you recorded earlier in the lab:

```
aws ec2 authorize-security-group-ingress \
--group-id <CafeDatabaseSG Group ID> \
--protocol tcp --port 3306 \
--source-group <CafeSecurityGroup Group ID>
```

    Note: Be careful to make the correct substitutions.

20. To confirm that the inbound rule was applied appropriately, run the following command:

```
aws ec2 describe-security-groups \
--query "SecurityGroups[*].[GroupName,GroupId,IpPermissions]" \
--filters "Name=group-name,Values='CafeDatabaseSG'"
```

    The output of the command should show that the CafeDatabaseSG security group now has an inbound rule that allows connections from TCP port 3306 if the source of the connection is an instance that has the CafeSecurityGroup Group ID association.

    Next, you create two private subnets and a database subnet group. First, you create CafeDB Private Subnet 1. This subnet hosts the RDS DB instance. It is a private subnet that is defined in the same Availability Zone as the CafeInstance.

You must assign the subnet a Classless Inter-Domain Routing (CIDR) address block that is within the address range of the VPC but that does not overlap with the address range of any other subnet in the VPC. This reason is why you collected the information about the VPC and existing subnet CIDR blocks:

- Cafe VPC IPv4 CIDR block: 10.200.0.0/20
- Cafe Public Subnet 1 IPv4 CIDR block: 10.200.0.0/24

Consider these address ranges. Can you find a suitable CIDR block for the private subnet? One possible answer is to use the address range 10.200.2.0/23.

21. To create the subnet, run the following command. In the command, replace *<CafeInstance VPC ID>* and *<CafeInstance Availability Zone>* with the values of **CafeVpcID** and **CafeInstanceAZ**, respectively, that you recorded earlier.

```
aws ec2 create-subnet \
--vpc-id <CafeInstance VPC ID> \
--cidr-block 10.200.2.0/23 \
--availability-zone <CafeInstance Availability Zone>
```

22. From the output of the command, note the value for **SubnetId**. You use this information later for CafeDB Private Subnet 1.

Next, you create CafeDB Private Subnet 2. This is the extra subnet that is required to form the database subnet group. It is an empty private subnet that is defined in a different Availability Zone than the CafeInstance.

Similar to what you did in the previous steps, you must assign a CIDR address block to the subnet that is within the address range of the VPC but does not overlap with the address range of any other subnet in the VPC. So far, you have used the

- Cafe VPC IPv4 CIDR block: 10.200.0.0/20
- Cafe Public Subnet 1 IPv4 CIDR block: 10.200.0.0/24
- Cafe Private Subnet 1 IPv4 CIDR block: 10.200.2.0/23

You use the 10.200.10.0/23 address range for this second private subnet.

For the Availability Zone for the second subnet, you can choose any other Availability Zone (but not the one ending in *a*).

23. To create the second subnet, run the following command. In the command, replace *<CafeInstance VPC ID>* with the value of **CafeVpcID** that you recorded earlier, and replace *<availability-zone>* with an Availability Zone that is different than the one that you used for the first subnet (for example, us-west-2b).

```
aws ec2 create-subnet \
--vpc-id <CafeInstance VPC ID> \
--cidr-block 10.200.10.0/23 \
--availability-zone <availability-zone>
```

24. From the output of the command, note the value for **SubnetId**. You use this information later for CafeDB Private Subnet 2.

Next, you create CafeDB Subnet Group.

For the Amazon RDS instance for the café, the DB subnet group consists of the two private subnets that you created in the previous steps: CafeDB Private Subnet 1 and CafeDB Private Subnet 2.

25. In the terminal window, run the following command. In the command, replace *<Cafe Private Subnet 1 ID>* and *<Cafe Private Subnet 2 ID>* with the subnet ID values that you recorded earlier for each private subnet. Note that there is a space between the subnet IDs in the command.

```
aws rds create-db-subnet-group \
--db-subnet-group-name "CafeDB Subnet Group" \
--db-subnet-group-description "DB subnet group for Cafe" \
--subnet-ids <Cafe Private Subnet 1 ID> <Cafe Private Subnet 2 ID> \
--tags "Key=Name,Value= CafeDatabaseSubnetGroup"
```

After the command completes, it returns the attributes of the DB subnet group.

Task 2.4: Creating the Amazon RDS MariaDB instance

You now create the CafeDBInstance that is shown in the final architecture. Using the AWS CLI, you create an Amazon RDS MariaDB instance with the following configuration settings:

- DB instance identifier: CafeDBInstance
- Engine option: MariaDB
- DB engine version: 10.5.13
- DB instance class: db.t3.micro
- Allocated storage: 20 GB
- Availability Zone: CafeInstanceAZ
- DB Subnet group: CafeDB Subnet Group
- VPC security groups: CafeDatabaseSG
- Public accessibility: No
- Username: root
- Password: Re:Start!9

These options specify the creation of a MariaDB database instance that is deployed in the same Availability Zone as the café instance. The MariaDB database instance also uses the DB subnet group that you built in the previous step.

26. In the terminal window, run the following command. In the command, replace <*CafeInstance Availability Zone*> with the **CafeInstanceAZ** value that you recorded at the beginning of the lab, and replace <*CafeDatabaseSG Group ID*> with the value that you recorded in a previous step.

    ```
    aws rds create-db-instance \
    --db-instance-identifier CafeDBInstance \
    --engine mariadb \
    --engine-version 10.5.13 \
    --db-instance-class db.t3.micro \
    --allocated-storage 20 \
    --availability-zone <CafeInstance Availability Zone> \
    --db-subnet-group-name "CafeDB Subnet Group" \
    --vpc-security-group-ids <CafeDatabaseSG Group ID> \
    --no-publicly-accessible \
    --master-username root --master-user-password 'Re:Start!9'
    ```

    The command immediately returns some information about the database, but the database instance might take up to 10 minutes to become available.

Next, you monitor the status of the database instance until it shows a status of *available*.

27. To check the status of the database, run the following command:

```
aws rds describe-db-instances \
--db-instance-identifier CafeDBInstance \
--query "DBInstances[*].
[Endpoint.Address,AvailabilityZone,PreferredBackupWindow,BackupRetentionPeriod,DBInstanceStatus]"
```

This command shows the following information for the database, including the status of the database as the last returned value:

- Endpoint address
- Availability Zone
- Preferred backup window
- Backup retention period
- Status of the database

Automated backups occur daily during the preferred backup window, and they are retained for the duration that is specified by the backup retention period. Note the value of **1** for the backup retention period, which indicates that, by default, daily backups are retained for only 1 day. Also, the preferred backup window is set to a 30 minute time interval by default. You can modify these settings to match your desired backup policy.

The status attribute initially shows a value of *creating* and then changes to *modifying*.

28. Wait a few moments, and repeat the command again. The status progressively changes to *backing-up* and finally to *available*.
29. Keep repeating the command until the status shows *available*. Then record the value that is returned for the endpoint address by using the following format:

```
RDS Instance Database Endpoint Address: cafedbinstance.xxxxxxx.us-west-2.rds.amazonaws.com
```

Task 3: Migrating application data to the Amazon RDS instance

In this task, you migrate the data from the existing local database to the newly created Amazon RDS database. Specifically, you do the following:

- Connect to the CafeInstance by using EC2 Instance Connect.
- Use the mysqldump utility to create a backup of the local database.
- Restore the backup to the Amazon RDS database.
- Test the data migration.

Note that you perform these steps from the command line after connecting to the CafeInstance. This instance can communicate with the Amazon RDS instance by using the MySQL protocol because you associated the CafeDatabaseSG security group with the Amazon RDS instance.

30. Connect to the CafeInstance by using EC2 Instance Connect. Follow the instructions that you used earlier to connect to the CLI Host instance.

    You use the mysqldump utility to create a backup of the local cafe_db database. This utility program is part of the MySQL database product, and it is available for you to use.

31. In the terminal window, run the following command:

```
mysqldump --user=root --password='Re:Start!9' \
--databases cafe_db --add-drop-database > cafedb-backup.sql
```

This command generates SQL statements in a file named cafedb-backup.sql, which can be run to reproduce the schema and data of the original cafe_db database.

32. To review the contents of the backup file, open the cafedb-backup.sql file in your preferred text editor. Alternatively, if you want to view it using the Linux less command, enter the following command in the terminal window:

    ```
    less cafedb-backup.sql
    ```

    Tip: When you use the less command, use the up and down arrow keys to move one line up or one line down, respectively. You can also use the Page up and Page down keys to navigate one page up or one page down, respectively. To quit the command, enter `q`.

    Notice the various SQL commands that create the database, create its tables and indexes, and populate the database with its original data.

    Next, you restore the backup to the Amazon RDS database by using the mysql command. You must specify the endpoint address of the Amazon RDS instance in the command.

33. In the terminal window, enter the following command. In the command, replace *<RDS Instance Database Endpoint Address>* with the value that you recorded earlier.

    ```
    mysql --user=root --password='Re:Start!9' \
    --host=<RDS Instance Database Endpoint Address> \
    < cafedb-backup.sql
    ```

    This command creates a MySQL connection to the Amazon RDS instance and runs the SQL statements in the cafedb-backup.sql file.

    Finally, you verify that the cafe_db was successfully created and populated in the Amazon RDS instance. You open an interactive MySQL session to the instance and retrieve the data in the product table of the cafe_db database.

34. In the SSH window, enter the following command. In the command, replace *<RDS Instance Database Endpoint Address>* with the value that you recorded earlier.

    ```
    mysql --user=root --password='Re:Start!9' \
    --host=<RDS Instance Database Endpoint Address> \
    cafe_db
    ```

35. Next, enter the SQL statement to retrieve the data in the product table:

    ```
    select * from product;
    ```

    The query should return rows from the table. Ensure that the information matches the number of items that you ordered at the beginning of the lab.

    You can now exit the interactive SQL session.

36. In the terminal window, enter the following command:

    ```
    exit
    ```

Task 4: Configuring the website to use the Amazon RDS instance

You are now ready to configure the café website to use the Amazon RDS instance. This step is straightforward because the designer of the application followed best practices and externalized the database connection information as parameters in Parameter Store, a capability of AWS Systems Manager. In this task, you change the database URL parameter of the café application to point to the endpoint address of the RDS instance.

37. On the **AWS Management Console**, in the **Search** bar, enter and choose `Systems Manager` to go to **AWS Systems Manager**.
38. In the left navigation pane, choose **Parameter Store**.
39. From the **My parameters** list, choose **/cafe/dbUrl**. The current value of the parameter is displayed, along with its description and other metadata information.
40. Choose **Edit**.
41. In the **Parameter details** page, replace the text in the **Value** box with the **RDS Instance Database Endpoint Address** value that you recorded earlier.
42. Choose **Save changes**.

    The dbUrl parameter now references the RDS DB instance instead of the local database.

    Next, you test the website to confirm that it is able to access the new database correctly.
43. In a new browser window, paste the URL for CafeInstanceURL that you copied to a text editor at the beginning of the lab.

    The website's home page should load correctly.
44. Choose the **Order History** tab. and observe the number of orders in the database. Compare this number with the number of

45. (Optional) Place some new orders, and verify the successful operation of the website. When you are finished, close the browser tab.

Task 5: Monitoring the Amazon RDS database

One of the benefits of using Amazon RDS is the ability to monitor the performance of a database instance. Amazon RDS automatically sends metrics to CloudWatch every minute for each active database. In this task, you identify some of these performance metrics and learn how to monitor a metric in the Amazon RDS console.

46. On the **AWS Management Console**, in the **Search** bar, enter and choose `RDS` to open the **RDS Management Console**.
47. In the left navigation pane, choose **Databases**.
48. From the **Databases** list, choose **cafedbinstance**. Detailed information about the database is displayed.
49. Choose the **Monitoring** tab. By default, this tab displays a number of key database instance metrics that are available from CloudWatch. Each metric includes a graph that shows the metric as it is monitored over a specific time span.

    The list of displayed metrics includes the following:

    - **CPUUtilization:** The percent of CPU utilization
    - **DatabaseConnections:** The number of database connections in use
    - **FreeStorageSpace:** The amount of available storage space
    - **FreeableMemory:** The amount of memory (RAM) available on the Amazon RDS instance
    - **WriteIOPS:** The average number of disk write I/O operations per second
    - **ReadIOPS:** The average number of disk read I/O operations per second

    Tip: Some of the metrics listed might appear on the second or third pages of charts. To see additional metrics, choose **2** or **3** to the right of the CloudWatch search box.

Next, you monitor the **DatabaseConnections** metric as you create a connection to the database from the CafeInstance.

50. To open an interactive SQL session to the RDS cafe_db instance, in the CafeInstance terminal window, enter the following command. In the command, replace *<RDS Instance Database Endpoint Address>* with the value that you recorded earlier.

```
mysql --user=root --password='Re:Start!9' \
--host=<RDS Instance Database Endpoint Address> \
cafe_db
```

51. To retrieve the data in the product table, enter the following SQL statement:

```
select * from product;
```

The query should return rows from the table.

52. In the Amazon RDS console, choose the **DatabaseConnections** graph to open it in a larger view. The graph shows a line that indicates that **1** connection is in use. This connection was established by the interactive SQL session from the CafeInstance.

    Tip: If the graph does not show any connections in use, wait 1 minute (which is the sampling interval), and then choose **Refresh**. It should now show one open connection.

53. To close the connection from the interactive SQL session, in the CafeInstance terminal window, enter the following command

```
exit
```

54. Wait 1 minute, and then in the **DatabaseConnections** graph in Amazon RDS console, choose **Refresh**. The graph now shows that the number of connections in use is **0**.

Conclusion

Congratulations! You now have successfully done the following:

- Created an Amazon RDS MariaDB instance by using the AWS CLI
- Migrated data from a MariaDB database on an EC2 instance to an Amazon RDS MariaDB instance
- Monitored the Amazon RDS instance by using CloudWatch metrics

Lab complete

Congratulations! You have completed the lab.

55. At the top of this page, choose End Lab and then choose **Yes** to confirm that you want to end the lab.

    A panel appears indicating that "You may close this message box now. Lab resources are terminating."

56. To close the **End Lab** panel, choose the **X** in the upper-right corner.