[*version_1.1.0]

Errors or corrections? Email us at aws-course-feedback@amazon.com

Other questions? Contact us at https://aws.amazon.com/contact-us/aws-training

Exercise: Migration Options for MySQL Using Percona Xtra Backup and Re-platforming to Amazon Aurora using the AWS Database Migration Service

After completing this exercise, you should be able to use a third-party tool called Percona XtraBackup to successfully migrate a MySQL database. You should also be able to use AWS Database Migration Service to migrate and replatform to Amazon Aurora.

Story

In the last exercise, you created a `myslqdump` of the database (DB) data and copied it to an Amazon Simple Storage Service (Amazon S3) bucket. You used that to restore (or seed) a newly prepared MySQL database in the target Region. You also copied over the Ghost application using an Amazon Elastic Block Store (Amazon EBS) snapshot.

In this exercise, you will try two methods to migrate the database. In the first part, you will use third-party software to create the `backup files` so that you can do a hot backup of the data. You will use that to create and seed a target MySQL database.

In the last part of this exercise, you will use AWS Database Migration Service (AWS DMS) to migrate the on-premises `us-west-2` MySQL database **directly** to Amazon Aurora in `us-east-1`. Using this more automated approach, you are essentially migrating and re-platforming.

Strategy

The first part of this exercise will be very *similar* to the previous exercise. You will do the following:

- From the AWS Cloud9 IDE, you will use a third-party tool (**Percona XtraBackup**) on the on-premises (`us-west-2`) database instance to carry out a backup of the database data to Amazon S3.
- You will use **Percona XtraBackup** to seed a prepared blank MySQL database in the target Region .

In the second part of this exercise, you will **do the migration all over again**. However, this time you will follow the instructions in the **AWS DMS** console to do a full database migration from Amazon Elastic Compute Cloud (Amazon EC2)-MySQL (`us-west-2`) to Amazon Aurora (`us-east-1`) to experience the ease of automating a **migration** and a **re-platform** using Amazon DMS.

Prepare the exercise

Accessing the AWS Management Console

1. At the top of these instructions, click Start Lab to launch your lab.
2. A Start Lab panel opens displaying the lab status.
3. Wait until you see the message "**Lab status: ready**", then click the **X** to close the Start Lab panel.
4. Choose Details and Show and then Download PEM
5. Close the credentials window.
6. At the top of these instructions, click AWS

This will open the AWS Management Console in a new browser tab. The system will automatically log you in.

> **TIP:** If a new browser tab does not open, there will typically be a banner or icon at the top of your browser indicating that your browser is preventing the site from opening pop-up windows. Click on the banner or icon and choose "Allow pop ups."

Arrange the AWS Management Console tab so that it displays along side these instructions. Ideally, you will be able to see both browser tabs at the same time, to make it easier to follow the lab steps.

Part 1 - Database Migration Using Percona XtraBackup

In Steps 1-4, you will use the **Percona XtraBackup** process to perform a database migration.

Step 1: Use AWS Cloud9 to Install Percona XtraBackup 2.4 on the On-premises (us-west-2) Database EC2 Instance.

You will start by heading over to the on-premises Region (`us-west-2`) and using AWS Cloud9, where you will shell into your existing database instance, install Percona on it, and initiate a *Percona style*backup.

> 📓 *Amazon Aurora MySQL can restore a DB cluster from files that were created using Percona XtraBackup. Percona XtraBackup is an open-source, free MySQL hot backup software that performs non-blocking backup. With the introduction of Percona XtraBackup 8.0, Percona XtraBackup 2.4 will continue to support MySQL and Percona Server 5.6 and 5.7 databases. Due to the new MySQL redo log and data dictionary formats, the Percona XtraBackup 8.0.x versions will only be compatible with MySQL 8.0.x and the upcoming Percona Server for MySQL 8.0.x.* For MySQL 5.7 migration, you must use Percona XtraBackup 2.4. For earlier MySQL versions, use Percona XtraBackup 2.3 or 2.4.

□□ As you are using MySQL 5.7 on your on-premises EC2 database instance, you are going to use `Percona XtraBackup 2.4` for this.

1. From the AWS Management Console, got to the **Services** menu and search for **Cloud9**. *Make sure you are in the US West (Oregon) Region.*
2. Choose **Open IDE**. *Close the other AWS Management Console tab.*
3. To upload the PEM file, choose **File** and **Upload Local Files**. Either drag and drop the `labsuser.pem` file or choose **Select files** and browse to the location where you downloaded the `labsuser.pem` file.
4. Go to the AWS Cloud9 terminal and run the following command to ensure that you are in the correct environment directory:

```
cd ~/environment
```

5. To change permissions of the .pem file so that only the root user can read it, run the following command:

```
chmod 400 labsuser.pem
```

6. To ssh into your database EC2 instance, run the following command:

```
ssh -i labsuser.pem ubuntu@10.16.11.80
#type yes when it asks, "Are you sure you want to continue connecting (yes/no)"
```

7. To confirm that your database is operational, run the following command:

```
mysql -u ghost -p
#Enter password: oranges
```

8. At the mysql prompt, run the following command:

```
show databases;
```

You should see:

```
+--------------------+
| Database           |
+--------------------+
| information_schema |
| ghost_prod         |
| mysql              |
| performance_schema |
| sys                |
+--------------------+
5 rows in set (0.00 sec)
```

> ⚠ If you do not see ghost_prod in there, it means it has not populated yet. Grab a cup of coffee and run the `show databases;` command again UNTIL it is there. You can also check the state of the population process by going to the **Services** menu and seraching for `CloudFormation`. From the AWS CloudFormation console, make sure it says `creation complete`

As you can see, **mysql** is running here with your `ghost_prod` database already in there.

9. Type `exit` to exit the **mysql** prompt.

You should see this:

```
mysql> exit
Bye
ubuntu@ip-10-16-11-80:~$
```

10. Now, as you are on the on-premises database EC2 instance (which is running **Ubuntu 16.04.2 LTS**), run the following commands to install **Percona XtraBackup 2.4** and prepare to do your backup:

```
wget https://repo.percona.com/apt/percona-release_latest.$(lsb_release -sc)_all.deb
```

```
sudo dpkg -i percona-release_latest.$(lsb_release -sc)_all.deb
```

```
sudo apt-get update
```

```
sudo apt-get -y install percona-xtrabackup-24
```

Now you will create an S3 bucket. The naming convention for your bucket should be the date + your initials + perforce-backup, for example: `2019-07-07-rh-perforce-backup`

11. To create your S3 bucket in the `us-east-1` Region, run the following command, making sure to replace the example bucket name with your unique bucket name:

```
aws s3api create-bucket --bucket 2019-07-07-rh-perforce-backup --region us-east-1
```

You should see something like this example:

```
{
    "Location": "/2019-07-07-rh-perforce-backup"
}
```

□□ You can disregard the `/` in terms of your bucket name. You will need this bucket name later in this lab.

12. You know from the previous two exercises that the EBS volume and your database are mounted to a folder called `ghost-db`. To navigate to it and inspect what's in there, run the following commands:

```
cd /ghost-db
```

```
ls
```

You should see:

```
lost+found  mysql
```

This output is what you would expect to see.

13. To exit from the folder, enter:

```
cd ~
```

14. Now it is time to create the backup. To create the backup, run the following Perforce command:

```
sudo xtrabackup --backup --stream=tar --user=ghost --password=oranges | gzip -c > /home/ubuntu/xtrabackup.tar.gz
```

You should see something similar to this example:

```
...

191001 16:54:52 Executing FLUSH NO_WRITE_TO_BINLOG ENGINE LOGS...
xtrabackup: The latest check point (for incremental): '4135171'
xtrabackup: Stopping log copying thread.
.191001 16:54:52 >> log scanned up to (4135180)

191001 16:54:52 Executing UNLOCK TABLES
191001 16:54:52 All tables unlocked
191001 16:54:52 [00] Streaming ib_buffer_pool to <STDOUT>
191001 16:54:52 [00]        ...done
191001 16:54:52 Backup created in directory '/home/ubuntu/xtrabackup_backupfiles/'
191001 16:54:52 [00] Streaming <STDOUT>
```

```
191001 16:54:52 [00]       ...done
191001 16:54:52 [00] Streaming <STDOUT>
191001 16:54:52 [00]       ...done
xtrabackup: Transaction log of lsn (4135171) to (4135180) was copied.
191001 16:54:52 completed OK!
```

15. Now run:

```
ls ~
```

You should see:

```
percona-release_latest.xenial_all.deb  xtrabackup_backupfiles  xtrabackup.tar.gz
```

Congratulations! You now have your backup, which is the Perforce equivalent of `mysqldump`. Next, you will move the backup from the `us-west-2` Region to the S3 bucket you created in the target `us-east-1` Region. ☐☐ *It is a Percona restore requirement to have the backup in the same Region as the target machine.*

16. To move your backup files to your S3 bucket, run the following command, making sure to replace the `<FMI>` (or `<Fill Me In>`) with your unique bucket name, e.g., `2019-07-07-rh-perforce-backup`.

```
aws s3 cp /home/ubuntu/xtrabackup.tar.gz s3://<FMI> --region us-east-1
```

You should see the following:

```
upload: ./xtrabackup.tar.gz to s3://<FMI>/xtrabackup.tar.gz
```

☐☐ Your colleague already set up the AWS Identity and Access Management (AWS IAM) role for the database EC2 instance to allow write access to these backup files directly to Amazon S3 in the on-premises Region. This is why that S3 `copy` command worked when you ran it.

📓 **Percona** does allow you to use replication to set up *FULL synchronizing* of the MySQL DB cluster with the MySQL database so that ongoing changes can be synchronized. However, this requires additional network configuration changes and is outside the scope of this exercise.

In this step, you have used **Percona** to create a more efficient backup than the standard `mysqldump` that you used in the previous exercise. You can now use this backup to efficiently create and seed a MySQL database.

Step 2: Seed the Target MySQL Database From Your New Percona Backup.

You could create (and seed) your target MySQL database from the Amazon Relational Database Service (Amazon RDS) console. However, it is sometimes faster to just use a CLI command.

In this step, you will use the `restore` command to create an RDS-MySQL database from your **Percona** backup that is stored in your `us-east-1` S3 bucket.

📓 There are a few things to note about your RDS-MySQL database:

- You will use a `db.t2.large`.

- You will be using the MySQL password for the `admin` user, and not for the `ghost` user.

  ⚠ You will still create the `ghost` user later on because it will not have been copied over during the **Percona** backup. This is because a **Percona** backup does not copy the **users** database or any stored procedures.

- You will need to provide an **IAM role** in the `restore` command, which will be different for you.

  Your IAM role will look similar to the following example:

```
arn:aws:iam::xxxxxxxxxxxxx:role/xxxxxxxxxxxxxxxxxxx-MigrationRole-xxxxxxxxxxxx
```

- To retrieve your IAM role, do the following:

1. From the AWS Cloud9 IDE, choose **AWS Cloud9** and then choose **Go To Your Dashboard**.
2. Choose the **Services** menu and search for **IAM**.
3. Choose **Roles**. Search for `MigrationRole` and note the **role ARN** not the *instance profile ARN*.

   □□ If you see multiple roles, click on the one beginning with the letter `c`, and copy down its Role ARN.

   The role ARN will look similar to this example:

```
arn:aws:iam::544487673545:role/c6060a57348u294892t1w544487673545-MigrationRole-HK5Z5O7TF1EO
```

   📓 You can only back up from **Percona** files that are in the same S3 Region as your target RDS Region (`us-east-1`), which is where your backup is stored.

- You will use `ghost-db-mysql` as the database identifier.

4. Run the following command to upgrade the awscli so that you have access to the `restore-db-instance`:

```
pip3 install awscli --upgrade --user
```

   You should see:

```
Successfully built PyYAML
Installing collected packages: docutils, jmespath, six, python-dateutil, urllib3, botocore, s3transfer, PyYAML, pyasn1, rsa, colorama, awscli
Successfully installed PyYAML-3.11 awscli-1.11.13 botocore-1.4.70 colorama-0.3.7 docutils-0.12 jmespath-0.9.0 pyasn1-0.1.9 python-dateutil-2.4.2 rsa-3.2.3 s3transfer-0.1.9 six-1.10.0 urllib3-1.13.1
You are using pip version 8.1.1, however version 19.2.3 is available.
You should consider upgrading via the 'pip install --upgrade pip' command.
```

□□ * For the following step, do not exit to your AWS Cloud9 environment home. You can run these commands while in the DB instance, which you should currently be in.*

5. Run the following `restore` command from your current AWS Cloud9 terminal in `us-west-2`, making sure to replace the first `<FMI>` with **your** S3 bucket name, and the second `<FMI>` in quotes with **your** Role ARN:

```
aws rds restore-db-instance-from-s3  --db-instance-identifier ghost-db-mysql --db-instance-class db.t2.large  --engine mysql  --source-engine mysql  --source-engine-version 5.7.22  --s3-bucket-name <FMI>  --s3-ingestion-role-arn "<FMI>"  --allocated-storage 100 --master-username admin  --master-user-password foobarfoobar --region us-east-1
```

   For example:

```
aws rds restore-db-instance-from-s3  --db-instance-identifier ghost-db-mysql --db-instance-class db.t2.large  --engine mysql  --source-engine mysql  --source-engine-version 5.7.22  --s3-bucket-name 2019-07-07-rh-perforce-backup  --s3-ingestion-role-arn "arn:aws:iam::544487673545:role/c6060a57348u294892t1w544487673545-MigrationRole-6NJ6743A3YS2"  --allocated-storage 100 --master-username admin  --master-user-password foobarfoobar --region us-east-1
```

You should see something *similar* to this example:


```
{
    "DBInstance": {
        "PerformanceInsightsEnabled": false,
        "MasterUsername": "admin",
        "MonitoringInterval": 0,
        "CopyTagsToSnapshot": false,
        "IAMDatabaseAuthenticationEnabled": false,
        "EngineVersion": "5.7.22",
        "DBSecurityGroups": [],
        "DomainMemberships": [],
        "DBInstanceStatus": "creating",
        "AutoMinorVersionUpgrade": true,
        "DBParameterGroups": [
            {
                "ParameterApplyStatus": "in-sync",
                "DBParameterGroupName": "default.mysql5.7"
            }
        ],
        "CACertificateIdentifier": "rds-ca-2015",
        "Engine": "mysql",
        "DeletionProtection": false,
        "StorageType": "gp2",
        "DBSubnetGroup": {
            "DBSubnetGroupName": "default",
            "VpcId": "vpc-0795557937fcf1eef",
            "SubnetGroupStatus": "Complete",
            "DBSubnetGroupDescription": "default",
            "Subnets": [
                {
                    "SubnetStatus": "Active",
                    "SubnetIdentifier": "subnet-0594a4eaa3679ace9",
                    "SubnetAvailabilityZone": {
                        "Name": "us-east-1d"
                    }
                },
                {
                    "SubnetStatus": "Active",
                    "SubnetIdentifier": "subnet-07bb60abcefd09dfb",
                    "SubnetAvailabilityZone": {
                        "Name": "us-east-1f"
                    }
                },
                {
                    "SubnetStatus": "Active",
                    "SubnetIdentifier": "subnet-04b20b2049f0daa0e",
                    "SubnetAvailabilityZone": {
                        "Name": "us-east-1c"
                    }
                },
                {
                    "SubnetStatus": "Active",
                    "SubnetIdentifier": "subnet-08534b0a24f04c7d0",
                    "SubnetAvailabilityZone": {
                        "Name": "us-east-1e"
                    }
                },
                {
                    "SubnetStatus": "Active",
                    "SubnetIdentifier": "subnet-08ab30724872ac16f",
                    "SubnetAvailabilityZone": {
```

```
                "Name": "us-east-1a"
            }
        },
        {
            "SubnetStatus": "Active",
            "SubnetIdentifier": "subnet-075af8d9e2733236a",
            "SubnetAvailabilityZone": {
                "Name": "us-east-1b"
            }
        }
    ]
},
"MultiAZ": false,
"PendingModifiedValues": {
    "MasterUserPassword": "****"
},
"BackupRetentionPeriod": 1,
"OptionGroupMemberships": [
    {
        "OptionGroupName": "default:mysql-5-7",
        "Status": "in-sync"
    }
],
"AllocatedStorage": 100,
"PreferredBackupWindow": "04:10-04:40",
"DBInstanceClass": "db.t2.large",
"PreferredMaintenanceWindow": "wed:09:44-wed:10:14",
"DBInstanceArn": "arn:aws:rds:us-east-1:544487673545:db:ghost-db-mysql",
"DBInstanceIdentifier": "ghost-db-mysql",
"StorageEncrypted": false,
"AssociatedRoles": [],
"PubliclyAccessible": true,
"DbInstancePort": 0,
"DbiResourceId": "db-4VIVFE3VVKDXACEYTE6BRTT7KE",
"ReadReplicaDBInstanceIdentifiers": [],
"LicenseModel": "general-public-license",
"VpcSecurityGroups": [
    {
        "Status": "active",
        "VpcSecurityGroupId": "sg-0938fee7bf28599e8"
    }
]
}
}
```

It can take *10-20 minutes* for the Aurora database to spin up.

6. To confirm that your Aurora database is ready, run the following command:

```
aws rds describe-db-instances --db-instance-identifier ghost-db-mysql --region us-east-1 --query
DBInstances[0].DBInstanceStatus
```

Keep running that command until it says `"available"`. It may remain in the `"creating"` state for a while.

⚠ Do not move on to the next step until it is `"available"`.

Congratulations! You have restored Aurora with your Percona backup.

7. You can now `exit` out of the database instance and return to the AWS Cloud9 environment:

```
exit
#ubuntu@ip-10-16-11-80:~$ exit
#logout
#Connection to 10.16.11.80 closed.
```

Step 3: Confirm that the ghost-prod Database is in the RDS-MySQL Database.

You will need to switch to the target Region `us-east-1` for this step.

1. Go to the IAM console tab, which should still be open.
2. Choose **Services** and choose **EC2**.
3. At the top right, switch to **US East (N. Virginia).**
4. From the left-hand navigation, choose **Instances** and choose the **ApplicationInstance**. *Note the VPC and Subnet ID as you will need them later. They should be called `Cloud9 VPC` and `Cloud9 subnet`.*
5. Choose **Services** and **Cloud9**.
6. Choose **Create environment**.
7. Name the environment `C9-us-east`.
8. Choose **Next step**.
9. Scroll down to **Network settings (advanced)**. For **Network (VPC)** choose the VPC you noted in step 4. For **Subnet** choose the subnet you noted.
10. Choose **Next step** and then choose **Create environment**.

    You now need to open the network settings and use Security Groups to allow Amazon Aurora to talk to the Ghost application instance that is also in `us-east-1`.
11. Using your new **Cloud9** instance in `us-east-1`, you first need to get the Public IP address of the application instance:

```
aws ec2 describe-instances --region us-east-1 --filters "Name=tag:Name,Values=ApplicationInstance" | grep -i -m 1 "PublicIpAddress"
```

You should see output similar to the following example:

```
"PublicIpAddress": "18.209.245.108,
```

12. Choose **AWS Cloud9** and **Go To Your Dashboard**.
13. Choose **Services** and search for **RDS**.
14. At the top right, confirm that you are in the **US East (N. Virginia)** Region.
15. From the left-hand navigation, choose **Databases**.
16. Select the **ghost-db-mysql** hyperlink.
17. Choose the **Connectivity & security** tab and scroll down to **Security group rules**. Note the `sg-xxxxxxxxx` under **Security group**. *You will need this in a moment*.
18. Choose **Services** and **VPC**.
19. On the left choose **Security Groups** and choose the *sg* you made a note of earlier.
20. Choose the **Inbound Rules** tab.
21. Choose **Edit rules** and choose **Add Rule**.
22. For the **Type**, choose **MySQL/Aurora**. This will populate **Protocol** and **Port Range**.
23. Leave the **Source** as **Custom** and enter the application instance IP address that you noted above. □□ Use `/32` at the end like this: `18.209.245.108/32`. Choose **Save rules** and **Close**.

    Now that your network allows your Ghost application instance to talk to Amazon Aurora, you can log in to the application instance and test the database contents using a `MySQL` client already installed on that machine. *Close the Security Groups tab that is open.*
24. Retrieve the RDS endpoint, as you will need this in a bit to tell the Ghost application where to look for its database. From the AWS Cloud9 terminal in `us-east-1`, run the following command:

```
aws rds describe-db-instances --db-instance-identifier ghost-db-mysql --region us-east-1 | grep Address
```

It will give you an Amazon Aurora host name similar to this example:

```
"Address": "ghost-db-aurora.czvran13tqn0.us-east-1.rds.amazonaws.com"
```

25. Run the following commands to log into the application instance in `us-east-1` and get the Private IP address:

```
aws ec2 describe-instances --region us-east-1 --filters "Name=tag:Name,Values=ApplicationInstance" | grep -i -m 1 "PrivateIpAddress"
```

```
ssh ubuntu@<PrivateIpAddress>
#type yes when it asks, "Are you sure you want to continue connecting (yes/no)"
```

The password is:

```
M1gration01L@B;
```

26. Using the MySQL client on that machine, try hitting the RDS Aurora endpoint to check that your security group modifications worked and that you have access to your DB cluster. Replace the `<FMI>` below with the host name you got earlier for your Aurora cluster.

```
mysql -u ghost -p -h <FMI>
#Enter password: oranges
```

For example:

```
mysql -u ghost -p -h ghost-db-aurora.ciphfxqewljg.us-east-1.rds.amazonaws.com
```

27. To verify that the `ghost_prod` database shows up, type the following at the MySQL prompt:

```
show databases;
```

28. You should see the following:

```
+--------------------+
| Database           |
+--------------------+
| information_schema |
| ghost_prod         |
| innodb             |
| mysql              |
| performance_schema |
| sys                |
+--------------------+
6 rows in set (0.08 sec)
```

29. Choose the `ghost_prod` database:

```
use ghost_prod;
```

30. Verify that the tables exist:

```
+------------------------+
| Tables_in_ghost_prod   |
+------------------------+
| accesstokens           |
| actions                |
| api_keys               |
| app_fields             |
| app_settings           |
| apps                   |
| brute                  |
| client_trusted_domains |
| clients                |
| integrations           |
| invites                |
| members                |
| migrations             |
| migrations_lock        |
| mobiledoc_revisions    |
| permissions            |
| permissions_apps       |
| permissions_roles      |
| permissions_users      |
| posts                  |
| posts_authors          |
```

```
| posts_tags            |
| refreshtokens         |
| roles                 |
| roles_users           |
| sessions              |
| settings              |
| subscribers           |
| tags                  |
| users                 |
| webhooks              |
+-----------------------+
31 rows in set (0.08 sec)
```

31. To exit out of the MySQL client but remain in the application instance, type `exit` once:

```
exit
```

Step 4: Configure and Test the Ghost Application.

1. As you are already in the target application instance, you can change to the `ghost-user`:

```
su ghost-user
#Password: pears
```

2. Change the directory to where your Ghost application is stored:

```
cd /ghost-app/ghost
```

3. To edit the file `config.production.json`, run the following sed command, making sure to replace `<FMI>` with your RDS endpoint:

```
sed -i 's/10.16.11.80/FMI'/ /ghost-app/ghost/config.production.json
```

For example:

```
sed -i 's/10.16.11.80/ghost-db-mysql.clahfmyp8re3.us-east-1.rds.amazonaws.com'/ /ghost-app/ghost/config.production.json
```

4. To verify the changes, run the following command:

```
cat config.production.json | grep rds
```

You should see that host has been updated with your RDS endpoint:

```
"host": "ghost-db-mysql.clahfmyp8re3.us-east-1.rds.amazonaws.com",
```

5. Because you are logged in as `ghost-user`, you can restart Ghost:

```
ghost restart
```

You should see something similar to the following example:

```
+ sudo systemctl is-active ghost_ghostblog
+ sudo systemctl restart ghost_ghostblog
✔ Restarting Ghost
```

6. You can run a `ghost status` command and access your Ghost site using the URL that is now using your RDS endpoint:

```
ghost status
```

7. Exit out of `ghost-user` and out of the application instance:

```
exit
exit
```

Congratulations! You have migrated your MySQL database to RDS-MySQL using **Percona**.

Now, you will stop your Ghost application and delete that RDS database. You will then use AWS DMS and do this over again just to see how much easier it is. 😜

Part 2 - Database Migration Using AWS Database Migration Service

Now in Steps 5-7 you are going to use AWS DMS, which makes it easy to migrate your on-premises database from the AWS DMS console.

Step 5: Create an Aurora Database for AWS DMS.

Ensure you are in `us-east-1`, which is your target region.

1. Choose **AWS Cloud9** and choose **Go To Your Dashboard**.
2. Choose **Services** and under **Database** choose **RDS**. In the upper right, make sure you are in the **US East (N. Virginia)** Region.*
3. Choose **Create database**.
4. For **Engine options** choose **Amazon Aurora**.
5. Choose **Next**.
6. Under **Capacity type**, leave **Provisioned** checked.
7. Under **Multi-AZ deployment**, choose **No**.
8. In the **Settings** section, name the cluster `ghost-aurora-dms`.
9. For the **Master username**, enter `ghost`.

10. For the **Master/Confirm password**, enter `foobarfoobar`.
11. Choose **Next**.
12. For **VPC**, choose the **Cloud9 VPC**.
13. For **Public accessibility**, check **Yes**.
14. Under **VPC security groups**, choose **Choose existing VPC security groups**. Select the sg with **DBGroup** in the name.
15. For **DB cluster identifier**, use `ghost-aurora-dms`.
16. Under **Encryption**, choose **Disable encryption**.
17. Under **Monitoring**, choose **Disable enhanced monitoring**.
18. Under **Performance Insights**, choose **Disable Performance Insights**.
19. Under **Deletion protection**, remove the check for **Enable deletion protection**.
20. Choose **Create database**.

    It may take several minutes before the cluster spins up. *Do not move onto the next step until the status of the cluster and all instances below it are available.*

Step 6: Set up the AWS Database Migration Service

1. Choose **Services** and under **Migration & Transfer** choose **Database Migration Service**.
2. Choose **Create replication instance**. Type `ghost-db-replication` under the **Name** field.
3. Under **Replication instance configuration**, choose **VPC** and select the **Cloud9 VPC**.
4. Under **Advanced security and networking configuration** and **VPC security group(s)** choose the *sg* with **DBGroup** in the name.
5. Choose **Create**. This may take several minutes. *Before moving on wait for the status to change to available.*
6. Under **Resource management**, choose **Endpoints**.
7. Choose **Create endpoint**, and then choose **Source endpoint**.
8. Name the endpoint `ghost-db-replication-source`.
9. Under **Source engine**, choose mysql.
10. Under **Server name**, enter **10.16.11.80**. For **Port**, enter **3306**.
11. For **Username**, use `ghost`. For **Password**, use `oranges`.
12. Under **Test endpoint connection**, make sure the **Cloud9 VPC** is selected as well as your replication instance `ghost-db-replication`.
13. Choose **Run test**. It should be successful.
14. Choose **Create endpoint**.
15. On the right, choose **Create endpoint**.
16. Choose **Target endpoint**.
17. Choose **Select RDS DB instance**, and from the drop-down list choose your **RDS Instance**. It should be `ghost-aurora-dms`.
18. It will populate everything except the password. Use the password `foobarfoobar`.
19. Under **Test endpoint connection**, under VPC, choose the **Cloud9 VPC**. Leave the **Replication instance** as `ghost-db-replication`.
20. Choose **Run test**.
21. After it succeeds, choose **Create endpoint**.
22. On the left under **Conversion & migration**, choose **Database migration tasks**.
23. Choose **Create task**.
24. Type `ghost-db-migration` for the **Task identifier**. Choose your **Replication instance** from the drop-down list. It should be `ghost-db-replication`.
25. Choose your **Source database endpoint**: `ghost-db-replication-srouce`. Then choose the **Target database endpoint**: `ghost-aurora-dms`.
26. Under **Table mappings**, choose **Add new selection rule**.
27. Under **Schema**, choose **Enter a schema**, and under **Schema name**, type `ghost_prod` .
28. Choose **Create task**. Wait for the status to change to **Load complete** before proceedign to the next step.

Step 7: Test Your RDS Endpoint

1. Choose **Services** and under **Database** choose **RDS**.
2. Choose **Databases** and then choose the `ghost-aurora-dms` database. Copy down the **Endpoint name** with the Type **Writer**.

   For example: `ghost-aurora-dms.cluster-ctc3juarhalw.us-east-1.rds.amazonaws.com`
3. Go back to AWS Cloud9 in `us-east-1`. To log into the database instance and test the connectivity to the RDS endpoint, run the following command:

```
ssh ubuntu@10.16.11.80
#type yes when it asks, "Are you sure you want to continue connecting (yes/no)"
```

   Password is

```
M1gration01L@B;
```

4. To connect to the RDS endpoint database, run the following command:

```
mysql -u ghost -p -h <FMI>
Enter password:foobarfoobar

 For example:

mysql -u ghost -p -h ghost-aurora-dms.cluster-ciphfxqewljg.us-east-1.rds.amazonaws.com
Enter password:foobarfoobar
```

5. To make sure your database is there, enter the following at the MySQL prompt:

```
show databases;
```

```
+--------------------+
| Database           |
+--------------------+
| information_schema |
| awsdms_control     |
| ghost_prod         |
| mysql              |
| performance_schema |
+--------------------+
5 rows in set (0.01 sec)
```

6. You can also make sure your tables are intact:

```
use ghost_prod;
```

```
show tables;
```

```
+------------------------+
| Tables_in_ghost_prod   |
+------------------------+
| accesstokens           |
| actions                |
| api_keys               |
| app_fields             |
| app_settings           |
| apps                   |
| brute                  |
| client_trusted_domains |
| clients                |
| integrations           |
| invites                |
| members                |
| migrations             |
| migrations_lock        |
| mobiledoc_revisions    |
| permissions            |
| permissions_apps       |
| permissions_roles      |
| permissions_users      |
| posts                  |
| posts_authors          |
| posts_tags             |
| refreshtokens          |
| roles                  |
| roles_users            |
| sessions               |
| settings               |
| subscribers            |
| tags                   |
| users                  |
| webhooks               |
+------------------------+
```

7. Exit out of the MySQL client and out of the instance:

```
exit
exit
```

Optional Step

You can also edit the Ghost application to use your new RDS cluster as well as update the config file to use your new password.

1. To find the Private IP address of your application instance, run the following command:

```
aws ec2 describe-instances --region us-east-1 --filters "Name=tag:Name,Values=ApplicationInstance" | grep -i -m 1 "PrivateIpAddress"
```

2. Use the Private IP address to log back into the application instance:

```
ssh ubuntu@FMI
ubuntu@FMI's password: M1gration01L@B;
```

3. Switch to the `ghost-user`:

```
su ghost-user
Password: pears
```

4. Change to the directory where your Ghost application is stored:

```
cd /ghost-app/ghost
```

5. Use your favorite editor to edit the config.production.json file and change the following:

```
Example:

Before:
"host": "ghost-db-mysql.clahfmyp8re3.us-east-1.rds.amazonaws.com",
"password": "oranges",

After:
"ghost-aurora-dms.cluster-clahfmyp8re3.us-east-1.rds.amazonaws.com",
"password": "foobarfoobar",
```

6. Restart the Ghost application:

```
ghost restart
```

```
+ sudo systemctl is-active ghost_ghostblog
+ sudo systemctl restart ghost_ghostblog
✔ Restarting Ghost
```

You are now running your Ghost application using your new RDS cluster endpoint.

7. To see the status, run the following command:

```
ghost status
```

8. Browse to the new IP address, which is at port `2368`. You should see that the blog has been moved successfully and is now using your RDS cluster as the back-end database.

Congratulations! You have completed this exercise. You can now migrate a database from an EC2 instance to RDS-MySQL using Percona XtraBackup. You also know how to use Amazon DMS to migrate and re-platform to RDS-Aurora.