[*version_1.0.0]

Exercise: Backing Up a MySQL Database to Amazon S3, and migrating the App using an EBS Snapshot

After completing this exercise, you should be able to use a conventional MySQL dump to backup and restore a Database and restore using Amazon Simple Storage Service (Amazon S3).

Learning Objectives

- Create a `mysqldump`.
- Use AWS CloudFormation to launch a MySQL database and a Ghost application in the target Region.
- Restore the target database from the `mysqldump`.
- Create a snapshot of the Ghost application's Amazon Elatic Block Store (EBS) drive.
- Mount a new EBS drive that you create from the snapshot.

Story

From the previous exercise, you are familiar with the configuration of your Ghost application and how it communicates with your MySQL database. You have decided that there is no need to change the hardware configurations based on the information you got from the discovery agent.

You decide to use the *tried and tested "conventional" approach* to migrate the MySQL database. You will create a `mysqldump` of the database and copy it to an S3 bucket.

After that, you will use the `mysqldump` to recreate the database on a target machine that your colleague has prepared for you in `us-east-1`.

📓 Your database needs to be restarted to set the replication options, and it also needs to be in read-only mode while the backup copy is created, so you need to schedule a maintenance window.

Strategy

The first step in the process of migrating a large amount of data to AWS with minimal downtime is to create a copy of the source data.

You will do the following:

1. Shell into your source database Amazon Elastic Compute Cloud (Amazon EC2) instance and create an S3 bucket.
2. Create a `mysqldump` and use a CLI command to copy it to your new S3 bucket.
3. Run an AWS CloudFormation script that launches a MySQL database and a Ghost application in the target Region.
4. Restore (or seed) the target database from the `mysqldump` that you saved in Amazon S3.
5. Take a snapshot of the Ghost application's Amazon Elatic Block Store (EBS) drive in the source (on-premises) Region, and copy it to the new Region.
6. Un-mount the target Region's Ghost application's EBS drive and mount a new one that you create from the snapshot. This step will migrate all the static (non-database) data for your Ghost application.

Prepare the exercise

%Access Console%

Step 1: Backup the Database

To shell into your source database EC2 instance, do the following:

1. From the AWS Management Console, go 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 that you downloaded earlier, choose **File** and then choose **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 annd run the following command to ensure that you are in the correct `environment` directory:

```
cd ~/environment
```

5. To change the 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)"
```

   Next, you will create an S3 bucket.

```
Your bucket name must be <u>globally unique</u>. We suggest you use your initials and the date as a prefix for your bucket name.
```

   Your bucket name must be globally unique. We suggest you use your initials and the date as a prefix for your bucket name.

   Here is an **example** bucket name:

```
2019-08-23-rh-mysql-backup
```

   NOTE: Your bucket name will be different.

7. To create an S3 bucket, run the following AWS CLI command, making sure to replace the (or) with your unique bucket name:

```
aws s3 mb s3://<FMI> --region us-east-1
```

   After you run that command, you should see something similar to this example (but showing *your* bucket name):

```
make_bucket: 2019-08-23-rh-mysql-backup
```

   MySQL uses a configuration file that is in the /etc folder.

8. To look at this file, run the following command:

```
cat /etc/mysql/mysql.conf.d/mysqld.cnf
```

Here is the full file that you will see:

```
#
# The MySQL database server configuration file.
#
# You can copy this to one of:
# - "/etc/mysql/my.cnf" to set global options,
# - "~/.my.cnf" to set user-specific options.
#
# One can use all long options that the program supports.
# Run program with --help to get a list of available options and with
# --print-defaults to see which it would actually understand and use.
#
# For explanations see
# http://dev.mysql.com/doc/mysql/en/server-system-variables.html

# This will be passed to all mysql clients
# It has been reported that passwords should be enclosed with ticks/quotes
# escpecially if they contain "#" chars...
# Remember to edit /etc/mysql/debian.cnf when changing the socket location.

# Here is entries for some specific programs
# The following values assume you have at least 32M ram

[mysqld_safe]
socket          = /var/run/mysqld/mysqld.sock
nice            = 0

[mysqld]
#
# * Basic Settings
#
user            = mysql
pid-file        = /var/run/mysqld/mysqld.pid
socket          = /var/run/mysqld/mysqld.sock
port            = 3306
basedir         = /usr
datadir         = /ghost-db/mysql
tmpdir          = /tmp
lc-messages-dir = /usr/share/mysql
skip-external-locking
#
# Instead of skip-networking the default is now to listen only on
# localhost which is more compatible and is not less secure.
bind-address            = 0.0.0.0
#
# * Fine Tuning
#
key_buffer_size         = 16M
max_allowed_packet      = 16M
thread_stack            = 192K
thread_cache_size       = 8
# This replaces the startup script and checks MyISAM tables if needed
# the first time they are touched
myisam-recover-options  = BACKUP
#max_connections        = 100
#table_open_cache       = 64
#thread_concurrency     = 10
#
# * Query Cache Configuration
#
query_cache_limit       = 1M
query_cache_size        = 16M
#
# * Logging and Replication
#
# Both location gets rotated by the cronjob.
# Be aware that this log type is a performance killer.
# As of 5.1 you can enable the log at runtime!
#general_log_file        = /var/log/mysql/mysql.log
```

```
#general_log             = 1
#
# Error log - should be very few entries.
#
log_error = /var/log/mysql/error.log
#
# Here you can see queries with especially long duration
#slow_query_log          = 1
#slow_query_log_file     = /var/log/mysql/mysql-slow.log
#long_query_time = 2
#log-queries-not-using-indexes
#
# The following can be used as easy to replay backup logs or for replication.
# note: if you are setting up a replication slave, see README.Debian about
#       other settings you may need to change.
#server-id               = 1
#log_bin                         = /var/log/mysql/mysql-bin.log
expire_logs_days        = 10
max_binlog_size   = 100M
#binlog_do_db            = include_database_name
#binlog_ignore_db        = include_database_name
#
# * InnoDB
#
# InnoDB is enabled by default with a 10MB datafile in /ghost-db/mysql/.
# Read the manual for more InnoDB related options. There are many!
#
# * Security Features
#
# Read the manual, too, if you want chroot!
# chroot = /ghost-db/mysql/
#
# For generating SSL certificates I recommend the OpenSSL GUI "tinyca".
#
# ssl-ca=/etc/mysql/cacert.pem
# ssl-cert=/etc/mysql/server-cert.pem
# ssl-key=/etc/mysql/server-key.pem
```

```

```

Notice that `/ghost-db/mysql` is the `datadir`. This is actually the EBS volume that is attached to the EC2 instance.

9. To confirm this, issue the following command:

```
df -h
```

You should see it is mounted here as `/dev/xvdb`.

```
Filesystem      Size  Used Avail Use% Mounted on
udev            481M     0  481M   0% /dev
tmpfs            99M  756K   98M   1% /run
/dev/xvda1      7.7G  1.6G  6.2G  21% /
tmpfs           492M     0  492M   0% /dev/shm
tmpfs           5.0M     0  5.0M   0% /run/lock
tmpfs           492M     0  492M   0% /sys/fs/cgroup
/dev/loop0       89M   89M     0 100% /snap/core/7169
/dev/loop1       18M   18M     0 100% /snap/amazon-ssm-agent/1335
/dev/xvdb       7.9G  238M  7.2G   4% /ghost-db
tmpfs            99M     0   99M   0% /run/user/1000
```

Also notice the following two lines in the config file:

`#server-id = 1`

and:

`#log_bin = /var/log/mysql/mysql-bin.log`

As you can see, they are both commented out in this file.

You need to change the config file to enable server logging. If you do not do change the config file, then when you do a `mysqldump` it will complain that Binlogging is not active on the server and error out.

10. To change the config file to enable server logging, run the following command:

```
sudo sed -i '/server-id/s/^#//g' /etc/mysql/mysql.conf.d/mysqld.cnf && sudo sed -i '/log_bin/s/^#//g'
/etc/mysql/mysql.conf.d/mysqld.cnf
```

11. To see the changes, run the following command:

```
cat /etc/mysql/mysql.conf.d/mysqld.cnf | grep -A1 server-id
```

Notice that the following change was made:

```
Before
#server-id              = 1
#log_bin                = /var/log/mysql/mysql-bin.log

After
server-id              = 1
log_bin                = /var/log/mysql/mysql-bin.log
```

12. To restart the MySQL service, run the following command:

```
sudo service mysql restart
```

13. To change directory to the home directory, run the following command:

```
cd ~
```

14. To create a backup of your database, run the following `mysqldump` command:

```
sudo mysqldump --databases ghost_prod --master-data=2 --single-transaction --order-by-primary -r backup.sql -u ghost -p
```

The password is `oranges`. You got this from Lab 1 when you inspected the Ghost configuration file.

15. You should now have a `backup.sql` file. To confirm, run the following command:

```
ls
```

You should see a file called `backup.sql`.

16. To copy the `backup.sql` file to your S3 bucket, run the following command, making sure to replace `<FMI>` with your unique bucket name:

```
aws s3 cp backup.sql s3://<FMI>
```

You should see the following output:

```
upload: ./backup.sql to s3://2019-08-23-rh-mysql-backup/backup.sql
```

17. To exit the DB server terminal and return to your AWS Cloud9 environment, run the exit command:

```
exit
#logout
#Connection to 10.16.11.80 closed.
```

Congratulations! You have created an S3 bucket, made a backup of your data, and stored it in the S3 bucket.

Time to create a new database (and the ghost app for later) in your target region and seed it with this `mysqldump`. This process is called a restore.

Step 2A: Prepare Your Target Environment in `us-east-1`

In this step, you will restore your **backup.sql** dump in your database instance in `us-east-1`. First, you need to replicate the network settings from `us-west-2` to `us-east-1`.

Luckily, your coworker has done most of the work already. *What a great guy!* He has provided you with a CloudFormation template that you can run in the target Region. He has uploaded it to an S3 bucket so it is easy for you to access.

He tells you that it will build out a standard MySQL database and a standard Ghost application that are already set up talk to each other.

You just need to seed the database with the `mysqldump`. Later, you will also seed the Ghost application's static data.

To begin, set up the target environment by creating a CloudFormation stack in the `us-east-1` Region as follows:

1. From the AWS Cloud9 IDE, choose **AWS Cloud9** at the top left and choose **Go To Your Dashboard**.
2. In the AWS Management Console, choose **Services**, and then choose **EC2**.
3. At the top right, switch to the **US East (N. Virginia)** Region. ⚠️ In this step, you are **changing Regions** from the `us-west-2` (Oregon) Region to the target `us-east-1` (N.Virgina) Region.
4. Under **Network & Security** in the left-hand navigation, choose **Key Pairs**.
5. Choose **Create Key Pair**.
6. Name the key pair `labsusereast`.
7. Choose **Create**.

   The key pair should automatically download to your local machine.
8. Choose **Services** and search for **CloudFormation** (still in `us-east-1`).
9. Choose **Create stack**.
10. Under **Specify template**, leave **Amazon S3 URL** highlighted.
11. Under **Amazon S3 URL**, enter the following URL:

```
https://aws-tc-largeobjects.s3-us-west-2.amazonaws.com/DEV-AWS-MO-Migration/lab-2-conventional/us-east-1.template
```

12. Choose **Next**.
13. Name the stack `clone-stack`.
14. Choose the `labsusereast` for **KeyName**.
15. Leave the other default settings and choose **Next**.
16. Again, leave the default settings, and choose **Next**.
17. Choose **I acknowledge that AWS CloudFormation might create IAM resources** at the bottom of the page.
18. Choose **Create stack**.

📓 This may take **5-10 minutes** for the stack to be created. This will replicate the network settings that are in `us-west-2`, build a standard Ghost application and MySQL database, and connect them for you.

⚠️ Before continuing to the next step, make sure that both stacks show **Create_Complete**.

Step 2B: Restore Your Database

Now that you have your target network set up with a MySQL database, you can restore (seed) it from your `backup.sql` dump.

⚠️ Only after the previous CloudFormation step has completed can you go to the AWS Cloud9 IDE in the **new** `us-east-1` Region. *Do some cleanup: close the tab for the AWS Cloud9 IDE in the* `us-west-2` *Region, but leave open the tab for the AWS Cloud9 IDE in* `us-east-1`.

After the stack finishes, do the following:

1. Choose **Services** and choose **Cloud9**. ⚠️ *Again, make sure you are in the* `us-east-1` *(N. Virginia) Region and not* `us-west-2`.
2. Choose **Open IDE**.
3. Choose **File** and **Upload Local Files**. Either drag and drop the `labsusereast.pem` file or choose **Select files** and browse to where you downloaded the `labsusereast.pem`.
4. To SSH into the database instance in `us-east-1` at `10.16.11.80` and copy the `backup.sql` from the S3 bucket, run the following commands:

```
cd ~/environment
```

```
chmod 400 labsusereast.pem
```

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

Next, you will copy the `backup.sql` from your S3 bucket to the database instance. You will need your unique bucket name.

TIP: *If you are unsure of your bucket name issue this command:*

```
aws s3 ls
#2019-08-23-rh-mysql-backup-2
```

5. To copy the `backup.sql` from your S3 bucket to the database instance, run the following command, making sure to replace with your unique bucket name:

```
aws s3 cp s3://<FMI>/backup.sql /home/ubuntu/backup.sql
```

You should see something similar to this example (but with your bucket name):

```
download: s3://2019-08-23-rh-mysql-backup-2/backup.sql to ./backup.sql
```

6. Run the following command:

```
mysql -u ghost -p
```

When prompted for the password, use `oranges`.

NOTE: *The ghost_prod databases may not appear immediately because the installation may still be running. Just wait a few moments and try again.*

7. At the MySQL command prompt, enter the following:

```
show databases;
```

8. Next, enter the following:

```
use ghost_prod;
```

You should see:

```
Database changed
```

9. To check for tables (which should be empty), run the following command:

```
show tables;
```

You should see:

```
Empty set (0.00 sec)
```

10. To run the restore, run the following command:

```
source backup.sql;
```

You should then see multiple lines of this:

```
...
...
Query OK, 0 rows affected (0.00 sec)
```

11. To see if you have all the tables that are needed for the Ghost application, run the following command:

```
show tables;
```

You should now see the following:

```
mysql> 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               |
+------------------------+
31 rows in set (0.00 sec)
```

Awesome! You have fully restored your database!

12. Now, exit out of the MySQL prompt:

```
exit
```

13. To get back to the `us-east-1` AWS Cloud9 environment, enter exit again:

```
exit
#logout
#Connection to 10.16.11.80 closed.
#voclabs:~/environment $
```

Step 3: Backup and Restore the Ghost Application

Now that you have your database backed up, you can migrate the Ghost application's data.

As you saw in the first exercise, the data for your Ghost application is stored on an EBS drive.

Thanks to the CloudFormation script that you ran, the target Region now has a Ghost application with a mounted EBS drive. However, that drive does not have useful data in it. You want the data on the EBS drive in the source Region.

You need to go through an EBS snapshot process to seed this target Ghost application.

Take this step by step.

First create a snapshot of the EBS drive attached to your Ghost application instance that is running in `us-west-2`.

From the first exercise, you know that the `/ghost-app/` EBS volume runs on `/dev/sdb`. However, you don't currently know the `InstanceId` of the Ghost application EC2 instance in `us-west-2`, so you need to get that first.

1. From the AWS Management Console, choose **AWS Cloud9** and then choose **Go To Your Dashboard**.
2. At the top right, change the Region to **US West (Oregon)**.
3. Choose **Services** and choose **EC2**.
4. In the left-hand navigation, choose **Instances**.
5. Choose **ApplicationInstance**, and under **Block devices** (at the bottom in the far-right) choose **/dev/sdb** .
6. In the pop up, open the **EBS ID** hyperlink to retrieve information on the volume.
7. Choose **Actions** and **Create Snapshot**.
8. Give it a description `ApplicationSnapshot`.
9. Choose **Create Snapshot**.
10. Choose **Close**.
11. On the left under **ELASTIC BLOCK STORE**, choose **Snapshots**. *Wait for the **Status** to change to **completed***. ☐☐ This can take a few minutes.
12. Choose **Actions** and **Copy**.
13. For the **Destination Region**, choose **US East (N. Virginia)**.
14. For the description, use `ApplicationSnapshot`.
15. Leave it unencrypted, and choose **Copy**.
16. Choose **Close**.

17. At the top right, switch back to **US East (N. Virginia)** to see if the snapshot is there. Choose **Snapshots** in the left-hand navigation under **ELASTIC BLOCK STORE**. Wait for the copy to complete. ☐☐ This can take a few minutes.

    After the copy is completed, you can create an EBS drive from that snapshot.

18. Choose **Actions** and **Create Volume**. Leave the default settings.
19. Choose **Create Volume**, and then **Close**.
20. On the left under **ELASTIC BLOCK STORE**, choose **Volumes**. It should be the volume that is **available**. ☐☐ You may need to scroll to the right to see this in your console. Make a note of the **Volume ID** because you will need to attach it to your target Ghost application instance.
21. Return to the **Cloud9** tab that is still open in **US East (N. Virginia)**.

    Although you now have a fully seeded volume ready to add to your application instance, you should first unmount the old one.

22. Log back into the Ghost application instance.

    To do this, you will need its IP address, which you can get by running this command:

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

    Now replace the with the IP address and run the following command:

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

    ```
    cd ~
    ```

23. Now change from root to `ghost-user`: ☐☐ Password is `pears`

    ```
    su ghost-user
    ```

24. To navigate to the Ghost application folder and stop the Ghost application, run the following commands:

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

    and:

    ```
    ghost stop
    ```

    You should see:

    ```
    + sudo systemctl is-active ghost_ghostblog
    + sudo systemctl stop ghost_ghostblog
    ✔ Stopping Ghost
    ```

25. Now navigate to the home directory:

    ```
    cd ~
    ```

26. To see the drive that you need to unmount first: `/ghost-app` A.K.A `/dev/xvdb`

```
Filesystem      Size  Used Avail Use% Mounted on
udev            481M     0  481M   0% /dev
tmpfs            99M  752K   98M   1% /run
/dev/xvda1      7.7G  2.0G  5.7G  26% /
tmpfs           492M     0  492M   0% /dev/shm
tmpfs           5.0M     0  5.0M   0% /run/lock
tmpfs           492M     0  492M   0% /sys/fs/cgroup
/dev/loop0       89M   89M     0 100% /snap/core/7169
/dev/loop1       18M   18M     0 100% /snap/amazon-ssm-agent/1335
/dev/xvdb       7.9G  292M  7.2G   4% /ghost-app
tmpfs            99M     0   99M   0% /run/user/1000
```

27. To unmount the drive, run:

```
exit
#to become root, and then
sudo umount /ghost-app
```

28. To verify that the drive is gone, run:

```
df -h
```

You should see:

```
Filesystem      Size  Used Avail Use% Mounted on
udev            481M     0  481M   0% /dev
tmpfs            99M  752K   98M   1% /run
/dev/xvda1      7.7G  2.0G  5.8G  25% /
tmpfs           492M     0  492M   0% /dev/shm
tmpfs           5.0M     0  5.0M   0% /run/lock
tmpfs           492M     0  492M   0% /sys/fs/cgroup
/dev/loop0       89M   89M     0 100% /snap/core/7169
/dev/loop1       18M   18M     0 100% /snap/amazon-ssm-agent/1335
tmpfs            99M     0   99M   0% /run/user/1000
```

From the previous steps, your volume is ready to use in `us-east-1` (your target Region). You need to mount it to the application EC2 instance that you are currently in and restart the application.

□□ In the real world, you no longer need older EBS drives, so you should also delete them from your console.

Now, attach the **new** volume that you created before from your snapshot.

29. Switch back to the **Volumes|EC2** tab that should still be open in `us-east-1`.
30. Choose the **Volume ID** that was available earlier, when you noted the Volume ID.
31. Choose **Actions** and **Attach Volume**.
32. In the **Instance** field, choose the **ApplicationInstance**.
33. Rename the **Device** to `/dev/sdb`.
34. Choose **Attach**.

    □□ You can ignore this warning: *Note: Newer Linux kernels may rename your devices to /dev/xvdf through /dev/xvdp internally, even when the device name entered here (and shown in the details) is /dev/sdf through /dev/sdp.*

35. Go back to the **Cloud9** tab that should still be open.
36. To mount the new drive, run the following command:

```
sudo mount /dev/xvdb /ghost-app/
```

37. To confirm the **new** volume is mounted, run the following command:

```
df -h
```

You should see that `/ghost-app` is there again:

```
udev          481M     0  481M   0% /dev
tmpfs          99M  756K   98M   1% /run
/dev/xvda1     7.7G  2.0G  5.7G  26% /
tmpfs         492M     0  492M   0% /dev/shm
tmpfs         5.0M     0  5.0M   0% /run/lock
tmpfs         492M     0  492M   0% /sys/fs/cgroup
/dev/loop0      89M   89M     0 100% /snap/core/7169
/dev/loop1      18M   18M     0 100% /snap/amazon-ssm-agent/1335
tmpfs          99M     0   99M   0% /run/user/1000
/dev/xvdb      7.9G  292M  7.2G   4% /ghost-app
```

38. Become the `ghost-user` again so that you can restart the Ghost application:

```
su ghost-user
#pears
```

39. Restart the Ghost application:

```
ghost restart
```

You should see the Ghost application interface using the new PublicIP:

```
+ sudo systemctl is-active ghost_ghostblog
✔ Ensuring user is not logged in as ghost user
✔ Checking if logged in user is directory owner
✔ Checking current folder permissions
+ sudo systemctl is-active ghost_ghostblog
✔ Validating config
✔ Checking folder permissions
✔ Checking file permissions
✔ Checking content folder ownership
✔ Checking memory availability
+ sudo systemctl start ghost_ghostblog
✔ Starting Ghost
+ sudo systemctl is-enabled ghost_ghostblog

------------------------------------------------------------------------------

Your admin interface is located at:
```

40. Browse to your EC2 instances's IP address, which is at port `2368`. You should see that your blog has been moved successfully to the target Region. ☐☐ The IP address will be in the above message when you restarted the Ghost application.

You now have a clone of your application and database instances running in `us-west-2`, with all the data copied over to your target Region!

Congratulations! You have completed the exercise.

Next, you will use a slightly different approach than `mysqldump` to migrate only the database. At the same time you will re-platform to Amazon Aurora.

%lab_complete%