[\*version\_1.0.0]

©2019 Amazon Web Services, Inc. and its affiliates. All rights reserved. This work may not be reproduced or redistributed, in whole or in part, without prior written permission from Amazon Web Services, Inc. Commercial copying, lending, or selling is prohibited.

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: 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:

Shell into your source database Amazon Elastic Compute Cloud (Amazon EC2) instance and create an S3 bucket.

Create a mysqldump and use a CLI command to copy it to your new S3 bucket.

Run an AWS CloudFormation script that launches a MySQL database and a Ghost application in the target Region.

Restore (or seed) the target database from the mysqldump that you saved in Amazon S3.

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.

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

Accessing the AWS Management Console

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

A Start Lab panel opens displaying the lab status.

Wait until you see the message "Lab status: ready", then click the X to close the Start Lab panel.

Choose Details and Show and then Download PEM

Close the credentials window.

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.

Step 1: Backup the Database

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

From the AWS Management Console, go to the Services menu and search for Cloud9. Make sure you are in the US West (Oregon) Region.

Choose Open IDE. Close the other AWS Management Console tab.

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.

Go to the AWS Cloud9 terminal annd run the following command to ensure that you are in the correct environment directory:

cd ~/environment

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

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.

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.

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.

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.

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

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

To restart the MySQL service, run the following command:

sudo service mysql restart

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

cd ~

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.

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

ls

You should see a file called backup.sql.

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

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:

From the AWS Cloud9 IDE, choose AWS Cloud9 at the top left and choose Go To Your Dashboard.

In the AWS Management Console, choose Services, and then choose EC2.

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.

Under Network & Security in the left-hand navigation, choose Key Pairs.

Choose Create Key Pair.

Name the key pair labsusereast.

Choose Create.

The key pair should automatically download to your local machine.

Choose Services and search for CloudFormation (still in us-east-1).

Choose Create stack.

Under Specify template, leave Amazon S3 URL highlighted.

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

Choose Next.

Name the stack clone-stack.

Choose the labsusereast for KeyName.

Leave the other default settings and choose Next.

Again, leave the default settings, and choose Next.

Choose I acknowledge that AWS CloudFormation might create IAM resources at the bottom of the page.

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:

Choose Services and choose Cloud9. ⚠️ Again, make sure you are in the us-east-1 (N. Virginia) Region and not us-west-2.

Choose Open IDE.

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.

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

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

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.

At the MySQL command prompt, enter the following:

show databases;

Next, enter the following:

use ghost\_prod;

You should see:

Database changed

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

show tables;

You should see:

Empty set (0.00 sec)

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)

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!

Now, exit out of the MySQL prompt:

exit

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.

From the AWS Management Console, choose AWS Cloud9 and then choose Go To Your Dashboard.

At the top right, change the Region to US West (Oregon).

Choose Services and choose EC2.

In the left-hand navigation, choose Instances.

Choose ApplicationInstance, and under Block devices (at the bottom in the far-right) choose /dev/sdb .

In the pop up, open the EBS ID hyperlink to retrieve information on the volume.

Choose Actions and Create Snapshot.

Give it a description ApplicationSnapshot.

Choose Create Snapshot.

Choose Close.

On the left under ELASTIC BLOCK STORE, choose Snapshots. Wait for the Status to change to completed. 💁‍♂ This can take a few minutes.

Choose Actions and Copy.

For the Destination Region, choose US East (N. Virginia).

For the description, use ApplicationSnapshot.

Leave it unencrypted, and choose Copy.

Choose Close.

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.

Choose Actions and Create Volume. Leave the default settings.

Choose Create Volume, and then Close.

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.

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.

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 ~

Now change from root to ghost-user: 💁‍♂ Password is pears

su ghost-user

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

Now navigate to the home directory:

cd ~

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

To unmount the drive, run:

exit

#to become root, and then

sudo umount /ghost-app

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.

Switch back to the Volumes|EC2 tab that should still be open in us-east-1.

Choose the Volume ID that was available earlier, when you noted the Volume ID.

Choose Actions and Attach Volume.

In the Instance field, choose the ApplicationInstance.

Rename the Device to /dev/sdb.

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.

Go back to the Cloud9 tab that should still be open.

To mount the new drive, run the following command:

sudo mount /dev/xvdb /ghost-app/

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

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

su ghost-user

#pears

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:

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%