[version_1.0.1]

# Exercise: Installing the AWS Application Discovery Agent

AWS Application Discovery Agent is a discovery tool that you can access through AWS Migration Hub. The Discovery Agent is AWS software that you install on on-premises servers and virtual machines (VMs) targeted for discovery and migration. Agents capture system configuration, system performance, running processes, and details of the network connections between systems.

After completing this exercise, you should be able to use the Discovery Agent to discover details about applications that you want to migrate to AWS.

## Learning Objective

- Install the Discovery Agent.

## Story

You have been tasked with migrating an application and its data from the `us-west-2` Region (consider this to be your on-premises environment) to the `us-east-1` Region of the AWS Cloud.

Your coworker (Dave) told you that the application you are to migrate was created using the Ghost open source publishing platform. This Ghost application runs on an Amazon Elastic Compute Cloud (Amazon EC2) instance and communicates with a separate database server.

This information is all you know about the current on-premises set up.

## Strategy

You decide that your first step will be to install a Discovery Agent on the Ghost application server and use it to perform a detailed discovery and collect data about your application server.

You will use the AWS Cloud9 terminal to shell into the application EC2 instance and issue commands to install the agent correctly.

**NOTE**: Remember that for this exercise, the `us-west-2` Region is your "on-premises" network, so you will operate AWS Cloud9 from there.

You also decide that you will install a Discovery Agent on the database server, which means that you will perform the installation process twice (once for each machine).

After you have connected to each machine, you will explore the environment to find out what the application and database do and how they are set up to talk to each other, noting any important configuration settings that you might need for the migration.

Finally, you will send a command to each agent to start collecting data. Collection includes system specifications, time series utilization or performance data, network connections, and process data.

After you collect all this information, you can use it to create a plan for migrating your on-premises application and data to AWS.

At that point, you could continue to the next exercise.

# Prepare the exercise

%Access Console%

Before you start, ensure that you are in `us-west-2` Region (your "on-premises" network).

## Step 1: Create an AWS Identity and Access Management (IAM) User

To install a Discovery Agent on an EC2 instance, you must provide security credentials, specifically an `access key ID` and a `secret acess key`.

You can generate these security credentials by creating an IAM `user` with one of AWS's predefined policies called `AWSApplicationDiscoveryAgentAccess`.

Let's do that first.

1. From the AWS Management Console, go to the **Services** menu and search for **IAM**.
2. From the left-hand navigation, choose **Users**.
3. Choose **Add user**, and for **User name** enter `migration-user`.
4. Under **Select AWS access type**, choose **Programmatic access**.
5. Choose **Next: Permissions**.
6. Choose **Attach existing policies directly**.
7. Next to **Filter policies**, enter `AWSApplicationDiscoveryAgentAccess`.
8. Select the check box next to the policy name and choose **Next: Tags**.
9. Choose **Next: Review** and **Create user**.
10. Copy the **Access key ID**.
11. Choose **Show** to see the **Secret access key** and copy it also.

**NOTE**: Make sure to copy the access key ID and secret access key because this is the **only** time you will be able to see them. You will need these security credentials to install a Discovery Agent on both the Ghost application server and on the database server.

## Step 2A: Connect to the Ghost Application EC2 Instance and Install the AWS Application Discovery Agent.

In this step, you will connect to your Ghost application EC2 instance from the AWS Cloud9 terminal and install a Discovery Agent.

1. Choose the **Services** menu and choose **Cloud9**.
2. Ensure that you are in the `us-west-2` (Orgeon) Region.
3. Choose **Open IDE**.
4. Upload the PEM file you downloaded earlier by choosing **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`.
5. Go to the AWS Cloud9 **bash terminal** (at the bottom of the page) and drag the area up so you have more room to work.
6. In the AWS Cloud9 terminal, ensure you are in the correct `environment` directory:

   ```
   cd ~/environment
   ```

   ```
   chmod 400 labsuser.pem
   ```

7. To get the private IP address of your application instance, run the following command:

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

   You should see something like this example:

```
"PrivateIpAddress": "10.16.10.62",
```

   Your IP will vary. Make a note of it, and use it in the next steps where you see the `<FMI>` (Fill Me In).

8. To SSH into your application EC2 instance, run the following command, making sure to replace the `<FMI>` with your private IP:

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

   *Example*: `ssh -i labsuser.pem ubunutu@10.16.10.2`.

9. To download what you need for this exercise and check that the application agent is not corrupted, run the following commands on your application instance.

   **NOTE**: You can also find these instructions here: https://docs.aws.amazon.com/application-discovery/latest/userguide/discovery-agent.html.

```
cd /home/ubuntu
```

```
curl -o ./aws-discovery-agent.tar.gz https://s3-us-west-2.amazonaws.com/aws-discovery-agent.us-west-2/linux/latest/aws-discovery-agent.tar.gz
```

```
curl -o ./agent.sig https://s3-us-west-2.amazonaws.com/aws-discovery-agent.us-west-2/linux/latest/aws-discovery-agent.tar.gz.sig
```

```
curl -o ./discovery.gpg https://s3-us-west-2.amazonaws.com/aws-discovery-agent.us-west-2/linux/latest/discovery.gpg
```

10. Now that you have all the files, run this command to make sure the fingerprint matches and that you have a good signature from the AWS Application Discovery Service:

```
gpg --no-default-keyring --keyring ./discovery.gpg --verify agent.sig aws-discovery-agent.tar.gz
```

   You should see something like this example (ignore the warning about the trusted signature):

```
gpg: Signature made Thu Apr 18 20:11:58 2019 UTC
gpg:                using RSA key 5133255E4DF42DA2
gpg: /home/ubuntu/.gnupg/trustdb.gpg: trustdb created
gpg: Good signature from "AWS Application Discovery Service <aws-application-discovery-service@amazon.com>" [unknown]
Primary key fingerprint: 7638 F24C 6717 F97C 4F1B  3BC0 5133 255E 4DF4 2DA2
```

11. To unzip and extract the tar file, run the following command:

```
tar -xzf aws-discovery-agent.tar.gz
```

12. To install the agent, run the following command, making sure to replace the first `<FMI>` with the `Access key ID` and the second `<FMI>` with the `Secret Access key` that you saved earlier:

```
sudo bash install -r us-west-2 -k <FMI> -s <FMI> -p true -c true -b true
```

*For example:*

```
sudo bash install -r us-west-2 -k AKIAXY5GUWFN7S5G26PI -s FUE13SE+pmjVScORfhUJsdj/Bafmibl87waDqVi8 -p true -c true
-b true
```

You should see something like this example:

```
**************************************************
FINISH PHASE
**************************************************

Notice:
By installing the Amazon Discovery Agent, you agree that your use is subject to the terms of your existing AWS Customer
Agreement or other agreement with Amazon Web Services, Inc. or its affiliates governing your use of AWS services. You may not
install and use the Amazon Discovery Agent unless you have an account in good standing with AWS.
* * *
Current running agent reports version as: 2.0.1252.0
This install script was created to install agent version: 2.0.1252.0
In most cases, these version numbers should be the same.
```

## Step 2B: Note Important Configuration Settings in the Ghost Application EC2 Instance.

First, you want to confirm that the Ghost application is actually running on the EC2 instance.

1. To confirm that the application is running, check for any processes that are in play with word "ghost" by running the following command:

```
ps aux | grep ghost
```

You should see something similar to the following example, which confirms that the Ghost application is running:

```
ghost    8303  0.0  4.4 584948 44472 ?      Ssl  17:47   0:00 ghost run
ghost    8331  0.5  9.4 1141484 95656 ?      Sl   17:47   0:04 /usr/bin/node current/index.js
ubuntu    9257  0.0  0.1  14856  1072 pts/0    S+   18:01   0:00 grep --color=auto ghost
```

Notice that you have a `ghost` user.

Now, continue exploring to see what else is configured:

2. To navigate up to the home directory, run:

```
cd ..
```

3. To list the information that is in the home directory, run:

   ```
   ls -la #list info on what is in this home directory
   ```

   You should see something like this:

   ```
   total 16
   drwxr-xr-x  4 root       root       4096 Aug  8 17:46 .
   drwxr-xr-x 24 root       root       4096 Aug  8 17:45 ..
   drwxr-xr-x  8 ghost-user ghost-user 4096 Aug  8 17:47 ghost-user
   drwxr-xr-x  5 ubuntu     ubuntu     4096 Aug  8 17:59 ubuntu
   ```

   Notice that you have a `ghost-user` in addition to a `ghost` user that you noted in a previous step.

   ☐☐ You will be provided with the configuration settings in later exercises so you don't need to write them down. Just be familiar with them.

4. Now look in the root folder by running the following command:

   ```
   cd / && ls
   ```

   You should see the following:

   ```
   bin   dev  ghost-app  initrd.img      lib    lost+found  mnt  proc  run   snap  sys  usr  vmlinuz
   boot  etc  home       initrd.img.old  lib64  media       opt  root  sbin  srv   tmp  var  vmlinuz.old
   ```

   Notice that you have a folder called `ghost-app`. To see if this folder is on the root volume or on an Amazon Elastic Block Store (EBS) volume, run the following command:

   ```
   df -h
   ```

   You should see the following:

   ```
   Filesystem      Size  Used Avail Use% Mounted on
   udev            481M     0  481M   0% /dev
   tmpfs            99M  760K   98M   1% /run
   /dev/xvda1      7.7G  2.2G  5.6G  28% /
   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  274M  7.2G   4% /ghost-app
   tmpfs            99M     0   99M   0% /run/user/1000
   ```

   You can see that the `ghost-app` is mounted on a 7.9G EBS drive. Make a mental note of this information.

5. Continue to explore the `ghost-app` folder by running the following command:

```
cd ghost-app && ls -la
```

You should see the following:

```
total 28
drwxr-xr-x  4 root       root       4096 Aug  8 17:46 .
drwxr-xr-x 24 root       root       4096 Aug  8 17:45 ..
drwxrwxr-x  5 ghost-user ghost-user  4096 Aug  8 17:47 ghost
drwx------  2 root       root      16384 Aug  8 17:45 lost+found
```

You can see that you have a `ghost` folder managed by `ghost-user`.

6. To see what's inside the `ghost` folder, run the following command:

```
cd ghost && ls -la
```

You should see the following:

```
drwxrwxr-x 5 ghost-user ghost-user 4096 Aug  8 17:47 .
drwxr-xr-x 4 root       root       4096 Aug  8 17:46 ..
-rw-rw-r-- 1 ghost-user ghost-user  115 Aug  8 17:48 .ghost-cli
-rw-rw-r-- 1 ghost-user ghost-user  537 Aug  8 17:47 config.production.json
drwxrwxr-x 8 ghost      ghost      4096 Aug  8 17:47 content
lrwxrwxrwx 1 ghost-user ghost-user   32 Aug  8 17:47 current -> /ghost-app/ghost/versions/2.27.0
drwxrwxr-x 3 ghost-user ghost-user 4096 Aug  8 17:47 system
drwxrwxr-x 3 ghost-user ghost-user 4096 Aug  8 17:47 versions
```

Note that there are two users here: `ghost` and `ghost-user`.

The `ghost` user is the user that will access the database from the Ghost application. The `ghost-user` is the system user that runs the application as a non-root user.

Now you want to check the configuration settings for the Ghost application, where you would expect to see some reference to the `ghost` user and how it is used.

7. Run the following command:

```
cat config.production.json
```

You should see all of the application settings:

```
{
  "url": "http://34.220.181.121:2368",
  "server": {
    "port": 2368,
    "host": "0.0.0.0"
  },
  "database": {
    "connection": {
      "host": "10.16.11.80",
      "user": "ghost",
      "password": "oranges",
      "database": "ghost_prod"
    }
  },
  "mail": {
    "transport": "Direct"
  },
  "logging": {
    "transports": [
      "file",
      "stdout"
    ]
  },
  "process": "systemd",
  "paths": {
    "contentPath": "/ghost-app/ghost/content"
  },
  "bootstrap-socket": {
    "port": 8000,
    "host": "localhost"
  }
}
```

Note the following in the application settings:

`10.16.11.80` is the database server's private IP address.

`ghost` is indeed the user that will access the database.

You can also see the credentials there. `contentPath` is where all the other "static" content is stored inside the mounted EBS.

You can also see a URL for the webapp on port 2368. However, that is an internal URL. If you want to see the public website, you must find the public IP address of the EC2 instance. Because your coworker Dave set up the network correctly, you should be able to access the EC2 instance from outside the network.

8. To get the public IP address for the Ghost application EC2 instance, run the following command:

```
curl http://169.254.169.254/latest/meta-data/public-ipv4
```

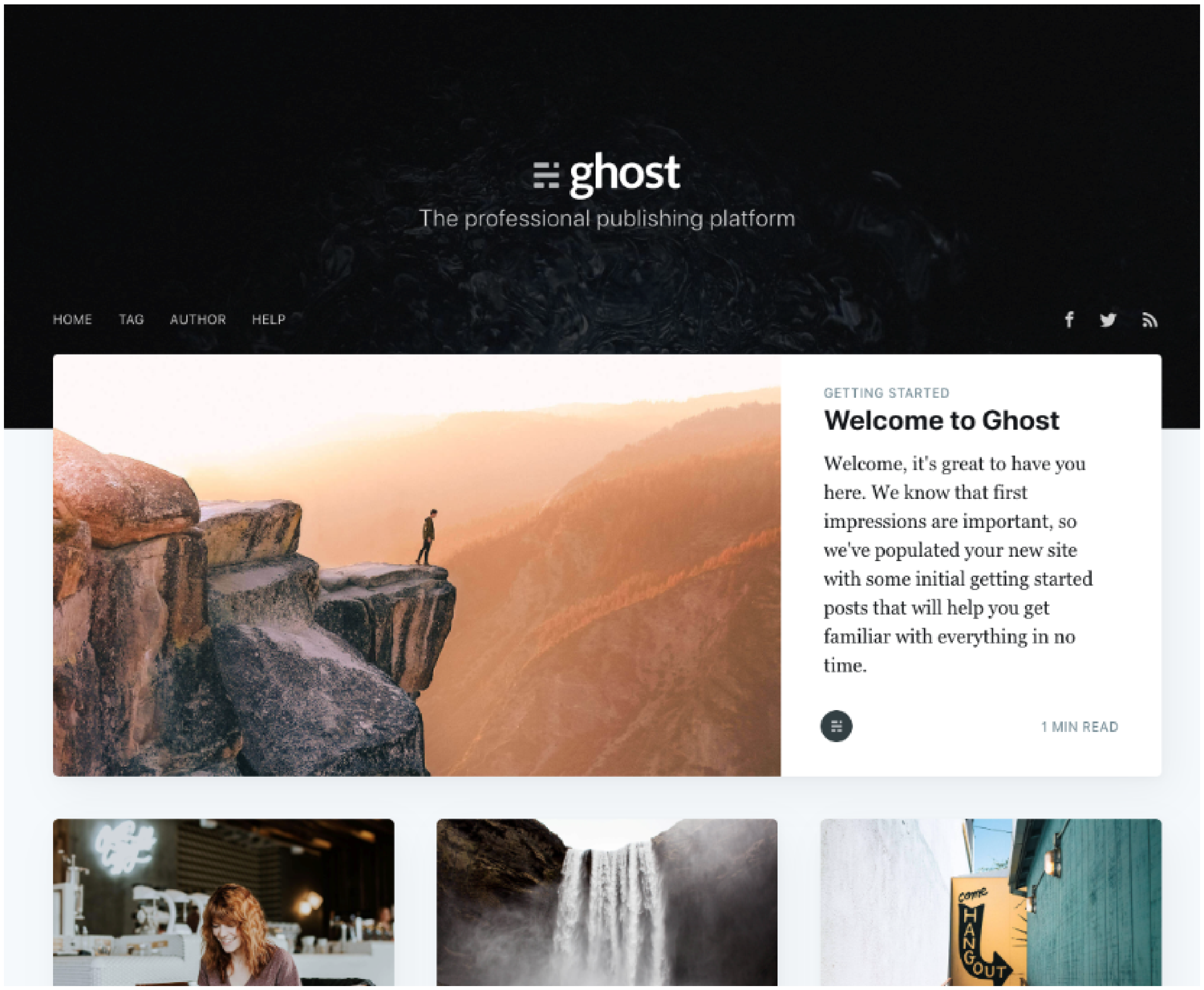
You should see something similar to this example:

```
34.220.181.121
```

You have all the information you need now about your application environment. Before you log out of this server, go to the URL and look at the application.

9. Open a new browser tab and paste in your Ghost application URL, being sure to replace the `<FMI>` with your public IP address:

```
http://<FMI>:2368
```


ghost
The professional publishing platform
HOME
TAG
AUTHOR
HELP
GETTING STARTED
Welcome to Ghost
Welcome, it's great to have you here. We know that first impressions are important, so we've populated your new site with some initial getting started posts that will help you get familiar with everything in no time.
1 MIN READ

**Congratulations!** You have installed a Discovery Agent on your Ghost application EC2 instance, and you know the basics of how your Ghost site is configured!

If you are curious, you can go to the admin section of the site and poke around:

```
http://<FMI>:2368/admin
```

Try changing some content on the Ghost website, for example edit the welcome post etc. You should see that any changes you make will be stored in the content folder (EBS) or the database.

For example, if you add a new post with an image file, the file will be placed on the application instance at /ghost-app/ghost/content/images. The further sub-directories will be by year and month.

☐ In the next exercise, you will be provided with a new environment that is very similar to this one. Any changes you make here in your Ghost application will not affect future exercises.

10. To log out of the Ghost application EC2 instance, run the following command in the AWS Cloud9 terminal:

```
exit
```

You should see something like this example:

```
logout
connection to 10.16.10.x closed
```

11. To confirm that you are in the root directory, run the following command:

```
pwd
```

You should see `/home/ec2-user/environment`. If you do not see this, run this command:

```
cd /home/ec2-user/environment
```

In the next step, you will connect to the database server and run a very similar process to install a Discovery Agent.

## Step 3A: Connect to the Database EC2 Instance and Install the AWS Application Discovery Agent.

In this step, you will connect to the database server from the AWS Cloud9 terminal and run the same commands to install the AWS Application Discovery Agent.

1. In the AWS Cloud9 terminal, ensure you are in the correct directory:

```
cd ~/environment
```

□□ Remember when you explored the application server? The database instance IP address 10.16.11.80 should match the private IP of the database that you noted in the Ghost configuration file.

2. To SSH into your database 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)"
```

3. To download what you need and check that the agent is valid and not corrupted, run the following commands on your database instance from the AWS Cloud9 terminal:

**NOTE**: You can also find these instructions here: https://docs.aws.amazon.com/application-discovery/latest/userguide/discovery-agent.html.

```
cd /home/ubuntu
```

```
curl -o ./aws-discovery-agent.tar.gz https://s3-us-west-2.amazonaws.com/aws-discovery-agent.us-west-2/linux/latest/aws-discovery-agent.tar.gz
```

```
curl -o ./agent.sig https://s3-us-west-2.amazonaws.com/aws-discovery-agent.us-west-2/linux/latest/aws-discovery-agent.tar.gz.sig
```

```
curl -o ./discovery.gpg https://s3-us-west-2.amazonaws.com/aws-discovery-agent.us-west-2/linux/latest/discovery.gpg
```

4. Now that you have all the files, run this command to make sure the fingerprint matches and that you have a good signature from AWS Application Discovery Service:

```
gpg --no-default-keyring --keyring ./discovery.gpg --verify agent.sig aws-discovery-agent.tar.gz
```

You should see something like this example:

```
gpg: Signature made Thu Aug  8 18:31:03 2019 UTC
gpg:                using RSA key 5133255E4DF42DA2
gpg: /home/ubuntu/.gnupg/trustdb.gpg: trustdb created
gpg: Good signature from "AWS Application Discovery Service <aws-application-discovery-service@amazon.com>" [unknown]
Primary key fingerprint: 7638 F24C 6717 F97C 4F1B  3BC0 5133 255E 4DF4 2DA2
```

5. To unzip and extract the tar file, run the following command:

```
tar -xzf aws-discovery-agent.tar.gz
```

6. Run the following command, making sure to replace the first `<FMI>` with the `Access key ID` and the second `<FMI>` with the `Secret Access key` that you copied earlier: (We are using the same IAM user as before).

```
sudo bash install -r us-west-2 -k <FMI> -s <FMI> -p true -c true -b true
```

*For example:*

```
sudo bash install -r us-west-2 -k AKIAXY5GUWFN7S5G26PI -s FUE13SE+pmjVScORfhUJsdj/Bafmibl87waDqVi8 -p true -c true
-b true
```

After you run this line, the agent should be fully installed on the database server.

In the next step, you will explore the database server and note any configuration settings.

## Step 3B: Note Important Configuration Settings in the Database EC2 Instance.

Recall the following important information you noted earlier about the Ghost configuration:

```
"database": {
  "connection": {
    "host": "10.16.11.80",
    "user": "ghost",
    "password": "oranges",
    "database": "ghost_prod"
  }
},
```

This information told you that there is a `ghost` user that accesses the database server. You will check that this user exists on this machine.

7. First, to check if the data is mounted like the Ghost application or if it is part of the root file system, run the following command:

```
df -h
```

You should see the following:

```
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
```

This shows that you have an EBS mount for this database EC2 instance as well: `ghost-db`

8. To find some more information about the database on this instance, run the following command.

```
ps aux
```

This will give you the following information on the processes:

```
USER       PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
root         1  0.1  0.9 159824  9080 ?        Ss   17:14   0:02 /sbin/init
root         2  0.0  0.0      0     0 ?        S    17:14   0:00 [kthreadd]
root         3  0.0  0.0      0     0 ?        I    17:14   0:00 [kworker/0:0]
root         4  0.0  0.0      0     0 ?        I<   17:14   0:00 [kworker/0:0H]
root         6  0.0  0.0      0     0 ?        I<   17:14   0:00 [mm_percpu_wq]
root         7  0.0  0.0      0     0 ?        S    17:14   0:00 [ksoftirqd/0]
root         8  0.0  0.0      0     0 ?        I    17:14   0:00 [rcu_sched]
root         9  0.0  0.0      0     0 ?        I    17:14   0:00 [rcu_bh]
root        10  0.0  0.0      0     0 ?        S    17:14   0:00 [migration/0]
root        11  0.0  0.0      0     0 ?        S    17:14   0:00 [watchdog/0]
root        12  0.0  0.0      0     0 ?        S    17:14   0:00 [cpuhp/0]
root        13  0.0  0.0      0     0 ?        S    17:14   0:00 [kdevtmpfs]
root        14  0.0  0.0      0     0 ?        I<   17:14   0:00 [netns]
root        15  0.0  0.0      0     0 ?        S    17:14   0:00 [rcu_tasks_kthre]
root        16  0.0  0.0      0     0 ?        S    17:14   0:00 [kauditd]
root        17  0.0  0.0      0     0 ?        S    17:14   0:00 [xenbus]
root        18  0.0  0.0      0     0 ?        S    17:14   0:00 [xenwatch]
root        20  0.0  0.0      0     0 ?        S    17:14   0:00 [khungtaskd]
root        21  0.0  0.0      0     0 ?        S    17:14   0:00 [oom_reaper]
root        22  0.0  0.0      0     0 ?        I<   17:14   0:00 [writeback]
root        23  0.0  0.0      0     0 ?        S    17:14   0:00 [kcompactd0]
root        24  0.0  0.0      0     0 ?        SN   17:14   0:00 [ksmd]
root        25  0.0  0.0      0     0 ?        SN   17:14   0:00 [khugepaged]
root        26  0.0  0.0      0     0 ?        I<   17:14   0:00 [crypto]
root        27  0.0  0.0      0     0 ?        I<   17:14   0:00 [kintegrityd]
root        28  0.0  0.0      0     0 ?        I<   17:14   0:00 [kblockd]
root        29  0.0  0.0      0     0 ?        I<   17:14   0:00 [ata_sff]
root        30  0.0  0.0      0     0 ?        I<   17:14   0:00 [md]
root        31  0.0  0.0      0     0 ?        I<   17:14   0:00 [edac-poller]
root        32  0.0  0.0      0     0 ?        I<   17:14   0:00 [devfreq_wq]
root        33  0.0  0.0      0     0 ?        I<   17:14   0:00 [watchdogd]
root        34  0.0  0.0      0     0 ?        I    17:14   0:00 [kworker/u30:1]
root        36  0.0  0.0      0     0 ?        S    17:14   0:00 [kswapd0]
root        37  0.0  0.0      0     0 ?        I<   17:14   0:00 [kworker/u31:0]
root        38  0.0  0.0      0     0 ?        S    17:14   0:00 [ecryptfs-kthrea]
root        80  0.0  0.0      0     0 ?        I<   17:14   0:00 [kthrotld]
root        81  0.0  0.0      0     0 ?        I<   17:14   0:00 [nvme-wq]
root        82  0.0  0.0      0     0 ?        S    17:14   0:00 [scsi_eh_0]
root        83  0.0  0.0      0     0 ?        I<   17:14   0:00 [scsi_tmf_0]
root        84  0.0  0.0      0     0 ?        S    17:14   0:00 [scsi_eh_1]
root        85  0.0  0.0      0     0 ?        I<   17:14   0:00 [scsi_tmf_1]
root        90  0.0  0.0      0     0 ?        I<   17:14   0:00 [ipv6_addrconf]
root       100  0.0  0.0      0     0 ?        I<   17:14   0:00 [kstrp]
root       167  0.0  0.0      0     0 ?        I<   17:14   0:00 [kworker/0:1H]
root       271  0.0  0.0      0     0 ?        I<   17:14   0:00 [raid5wq]
root       323  0.0  0.0      0     0 ?        S    17:14   0:00 [jbd2/xvda1-8]
root       324  0.0  0.0      0     0 ?        I<   17:14   0:00 [ext4-rsv-conver]
root       391  0.0  1.1  86660 12000 ?        S<s  17:14   0:02 /lib/systemd/systemd-journald
root       396  0.0  0.0      0     0 ?        I<   17:14   0:00 [iscsi_eh]
root       407  0.0  0.0      0     0 ?        I<   17:14   0:00 [ib-comp-wq]
root       408  0.0  0.0      0     0 ?        I<   17:14   0:00 [ib_mcast]
root       412  0.0  0.1  97708  1876 ?        Ss   17:14   0:00 /sbin/lvmetad -f
root       413  0.0  0.0      0     0 ?        I<   17:14   0:00 [ib_nl_sa_wq]
root       414  0.0  0.5  44220  5332 ?        Ss   17:14   0:00 /lib/systemd/systemd-udevd
root       419  0.0  0.0      0     0 ?        I<   17:14   0:00 [rdma_cm]
systemd+   526  0.0  0.3 141924  3044 ?        Ssl  17:14   0:00 /lib/systemd/systemd-timesyncd
systemd+   628  0.0  0.5  80040  5108 ?        Ss   17:14   0:00 /lib/systemd/systemd-networkd
root       652  0.0  0.0      0     0 ?        I    17:14   0:00 [kworker/0:2]
systemd+   653  0.0  0.6  70740  6064 ?        Ss   17:14   0:00 /lib/systemd/systemd-resolved
root       794  0.0  0.6  70576  6148 ?        Ss   17:14   0:00 /lib/systemd/systemd-logind
root       814  0.0  1.3 170872 13376 ?        Ssl  17:14   0:00 /usr/bin/python3 /usr/bin/networkd-dispatcher --run-startup-triggers
message+   817  0.0  0.4  50056  4124 ?        Ss   17:14   0:00 /usr/bin/dbus-daemon --system --address=systemd: --nofork --
nopidfile --
root       848  0.0  0.6 287992  6440 ?        Ssl  17:14   0:00 /usr/lib/accountsservice/accounts-daemon
root       849  0.0  0.1 169408  1588 ?        Ssl  17:14   0:00 /usr/bin/lxcfs /var/lib/lxcfs/
root       858  0.0  0.2  31848  2748 ?        Ss   17:14   0:00 /usr/sbin/cron -f
daemon     863  0.0  0.2  28332  2164 ?        Ss   17:14   0:00 /usr/sbin/atd -f
syslog     868  0.0  0.4 267268  4216 ?        Ssl  17:14   0:00 /usr/sbin/rsyslogd -n
root       893  0.0  1.6 187716 16380 ?        Ssl  17:14   0:00 /usr/bin/python3 /usr/share/unattended-upgrades/unattended-
upgrade-shutd
root       913  0.0  0.1  16412  1800 ttyS0    Ss+  17:14   0:00 /sbin/agetty -o -p -- \u --keep-baud 115200,38400,9600 ttyS0 vt220
root       946  0.0  0.1  14888  1580 tty1     Ss+  17:14   0:00 /sbin/agetty -o -p -- \u --noclear tty1 linux
root       969  0.0  0.6 288876  6064 ?        Ssl  17:14   0:00 /usr/lib/policykit-1/polkitd --no-debug
root      1110  0.0  0.3  72296  3456 ?        Ss   17:14   0:00 /usr/sbin/sshd -D
root      1231  0.0  0.0      0     0 ?        S<   17:14   0:00 [loop0]
root      1285  0.0  2.2 632676 22732 ?        Ssl  17:14   0:00 /usr/lib/snapd/snapd
root      1386  0.0  0.0      0     0 ?        S<   17:14   0:00 [loop1]
root      1439  0.0  1.6 553800 17008 ?        Ssl  17:14   0:00 /snap/amazon-ssm-agent/1335/amazon-ssm-agent
root      3034  0.0  0.0      0     0 ?        S    17:15   0:00 [jbd2/xvdb-8]
root      3035  0.0  0.0      0     0 ?        I<   17:15   0:00 [ext4-rsv-conver]
mysql     3168  0.1 19.8 1163844 199736 ?      Sl   17:15   0:05 /usr/sbin/mysqld --daemonize --pid-file=/run/mysqld/mysqld.pid
root      3253  0.0  0.0      0     0 ?        I    17:29   0:00 [kworker/u30:2]
root      3269  0.0  0.7 107984  7216 ?        Ss   17:57   0:00 sshd: ubuntu [priv]
ubuntu    3279  0.0  0.7  76508  7284 ?        Ss   17:57   0:00 /lib/systemd/systemd --user
ubuntu    3287  0.0  0.2 193808  2580 ?        S    17:57   0:00 (sd-pam)
ubuntu    3404  0.0  0.3 107984  3552 ?        S    17:57   0:00 sshd: ubuntu@pts/0
ubuntu    3405  0.0  0.4  23144  4852 pts/0    Ss   17:57   0:00 -bash
root      3435  0.0  0.0      0     0 ?        I    17:58   0:00 [kworker/u30:0]
root      3692  0.0  0.0      0     0 ?        I    17:58   0:00 [kworker/0:1]
root      3915  0.0  0.0      0     0 ?        I    17:58   0:00 [kworker/u30:3]
root      4027  0.0  1.3  80320 13340 ?        Sl   17:58   0:00 /var/opt/aws/discovery/aws-discoveryd
ubuntu    4050  0.0  0.3  40092  3536 pts/0    R+   18:01   0:00 ps aux
```

```
```

If you scan through that information, you can see mysql.

9. To see the mysql, do a new search by running the following command:

```
ps aux | grep mysql
```

You should the following:

```
mysql     3103  0.1 20.1 1161236 203320 ?      Sl   17:46   0:06 /usr/sbin/mysqld --daemonize --pid-file=/run/mysqld/mysqld.pid
ubuntu    3442  0.0  0.1  14856  1040 pts/0    S+   19:18   0:00 grep --color=auto mysql
```

10. To investigate the MySQL configuration file, run the following command using MySQL's default config location:

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

You should see the following:

```
#
# 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
```

What you will notice is that its using the default MYSQL port of 3306, and that the data directory is pointing to this machine's EBS mount, `/ghost-db/mysql`

The bind-address is set to `0.0.0.0` which tells MySQL to listen to every available network. This is fine since our Database instance is in a private subnet.

11. To check that you can access the database using the password you found before in the Ghost application configuration file (that is, `oranges`), run the following command:

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

12. Enter the password `oranges`.
13. At the mysql prompt, enter the following:

```
show databases;
```

You should see:

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

14. To check that `ghost_prod` looks like your database, enter the following at the mysql prompt:

```
use ghost_prod;
```

It should say "Database Changed".

15. Now, list your tables:

```
show tables;
```

You should see:

```
+------------------------+
| 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               |
+------------------------+
```

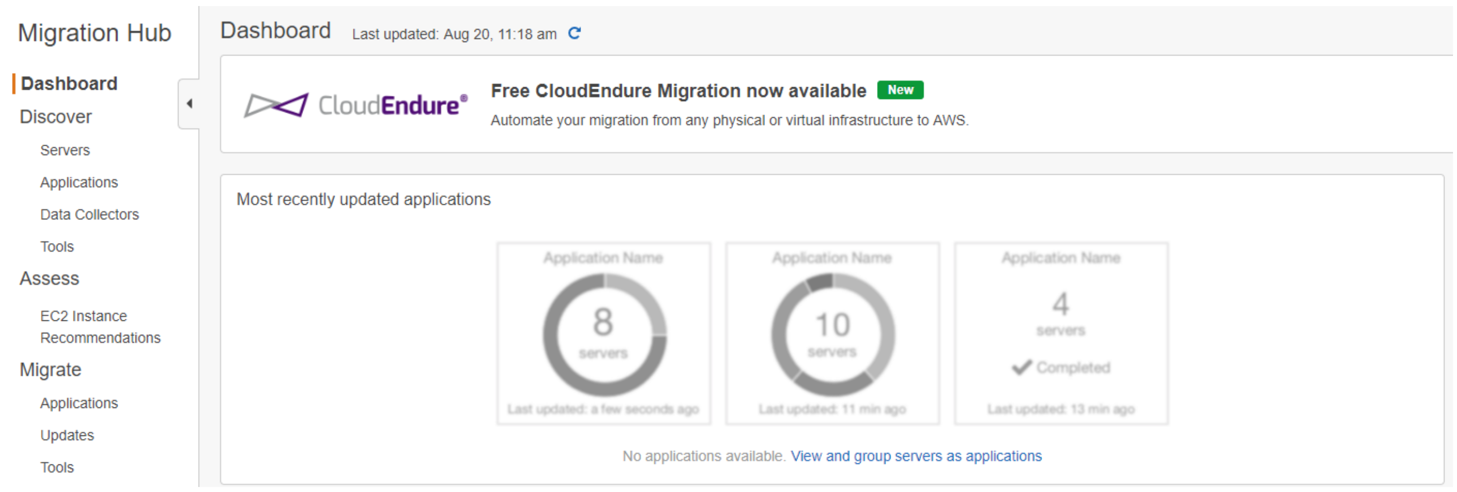
**Congratulations!** You have connected to the database server and successfully installed a Discovery Agent. You know a lot about the configuration and how the database is connected to the application.

The final step of this exercise is to make sure that the agent you installed is actually collecting data.

## Step 4: Start the Agent Data Collection Process.

Now that you have installed the agent on both instances, you will start the collection process.

1. Choose **AWS Cloud9** and select **Go To Your Dashboard**.



Dashboard Last updated: Aug 20, 11:18 am

Free CloudEndure Migration now available

Automate your migration from any physical or virtual infrastructure to AWS.

Most recently updated applications

Last updated: a few seconds ago

Last updated: 11 min ago

Last updated: 13 min ago

No applications available. View and group servers as applications

2. Choose **Services** and search for **AWS Migration Hub**.
3. From the left-hand navigation, under **Discover**, choose **Data Collectors** and select the **Agents**tab.

**NOTE**: You may have to wait `4-5` minutes for them to appear.

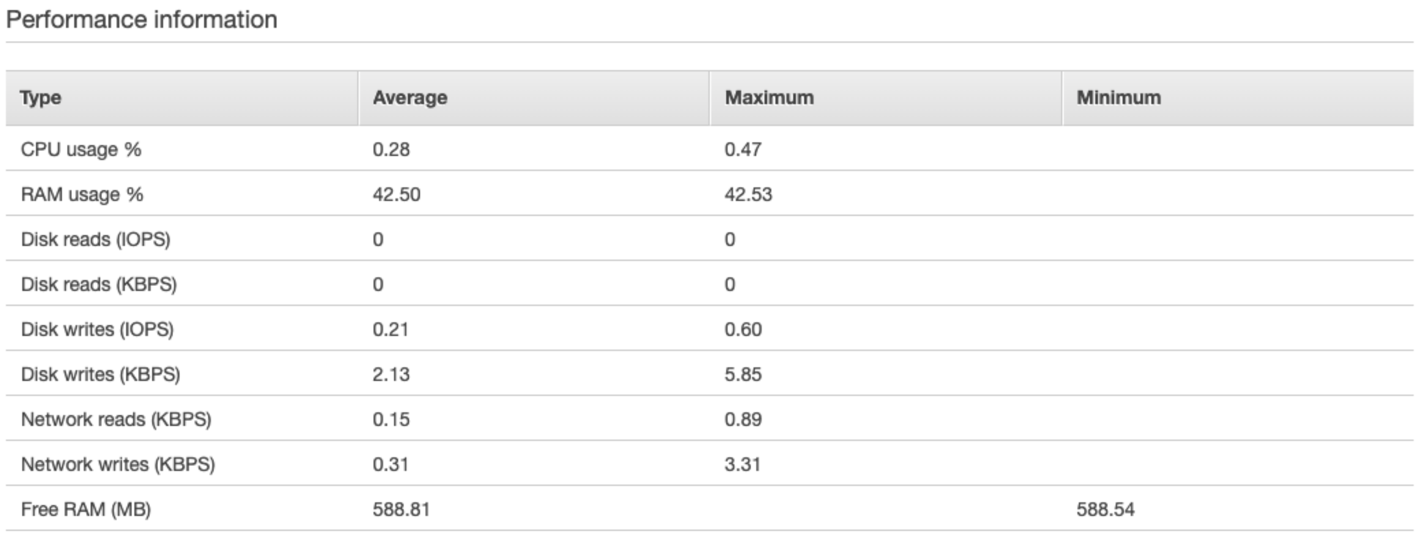
4. Select the **Agent** and choose **Start data collection**.
5. Uncheck **Enable data exploration in Amazon Athena for all agents.** and choose **Continue**.

The AWS Migration Hub provides a single place to discover your existing servers and track the status of each application migration. It's useful to get a current baseline of your environment before doing an AWS Cloud migration.

6. Under **Discover**, choose **Servers**. You will see both of your machines.
7. Select the IP address of each machine to see the data.

□□ It may take up to **10 minutes** before your data shows up in the console.

Here is an example of what you might see:



Performance information

| Type | Average | Maximum | Minimum |
|---|---|---|---|
| CPU usage % | 0.28 | 0.47 | |
| RAM usage % | 42.50 | 42.53 | |
| Disk reads (IOPS) | 0 | 0 | |
| Disk reads (KBPS) | 0 | 0 | |
| Disk writes (IOPS) | 0.21 | 0.60 | |
| Disk writes (KBPS) | 2.13 | 5.85 | |
| Network reads (KBPS) | 0.15 | 0.89 | |
| Network writes (KBPS) | 0.31 | 3.31 | |
| Free RAM (MB) | 588.81 | | 588.54 |

This will provide you with information relating to your machine, specifically you should look at the `performance information`.

You can use this performance information to determine the current load on each machine and make sure that the machines and drives that you are currently using are appropriate to use as the target machines.

It is very easy to make decisions about your target machines, based on the performance metrics of your source machines.

Some examples:

## Too much CPU usage on your Database Server?

Opt for a C2 machine as the target.

## Too much RAM usage on your Application Server?

Opt for an R2 machine as the target.

## Too many disc reads and writes?

Perhaps opt for a high IO provisioned IOPS EBS drive.

As you can imagine, this type of information is invaluable for a migration because it helps you decide which type of machines and hardware you need to use as your target.

You could also use Amazon CloudWatch metrics if you are moving from one AWS region to another AWS region. However, remember this agent **can run on your on-premises servers** and give you the data you need to make good instance-sizing decisions before migrating applications and data to the cloud.

After the performance metrics start appearing in the console, you will see that your machines are appropriately sized for the Ghost application.

There is no need to change to a more performant EBS drive or a new type of machine for either of your servers.

This will make the steps that you carry out in the next exercise a bit easier because now you only have to worry about migrating data to machines that are like for like! #win 😃

**Congratulations!** You have completed the first exercise.

%Lab Complete%