Live Migration Guide: Amazon Aurora to HeatWave MySQL on Oracle Cloud Infrastructure (OCI)

Before you start:

- You must have an account on Oracle Cloud Infrastructure (OCI) and Amazon Web Services (AWS).
- Some OCI knowledge is preferred.
- This live migration guide only covers how to migrate your database from Amazon Aurora MySQL to HeatWave MySQL on OCI. Before performing the migration, you should have considered downtime (even though this is a live migration, some/minimal downtime will be required to make sure your database application points to the new HeatWave MySQL database once migrated), application compatibility, current database metrics (CPU, storage size, RAM, max number of concurrent users, backups, binary logs expiration, number of replicas if any, etc.), desired database metrics, networking, security, user testing, etc.
- The live migration method shown in this guide works for Amazon Aurora MySQL v5.7 and above.
- When following the guide, you should always execute the commands/steps shown as an admin/root user wherever applicable.
    - On OCI and AWS you must have the ability to create and manage resources.
    - For your Amazon Aurora MySQL instance, use an admin/root user.
- This live migration method requires binary logs to be present on the Aurora instance. To enable Aurora binary logs – you must modify the parameter group used by Aurora and set the `binlog_format` variable to `ROW`. Any other values besides `ROW` will not work as HeatWave MySQL on OCI only uses row-based binary logging. For more information on how to set the Aurora binary logging, see [Configuring Aurora MySQL binary logging].
- This live migration can be performed using two replication methods - using binary log position or GTIDs. As HeatWave MySQL only supports GTIDs on OCI, once you migrate your Aurora instance to HeatWave MySQL - you cannot go back to using the binary log position for replication.
- If you have Aurora replication configured in your current AWS environment, you can perform the migration steps shown in this guide from either your writer or reader instance. Although it is recommended to use the reader instance for the migration when applicable. This is because if you have a high concurrency for your Aurora instance - performing the migration using the writer instance could negatively impact the database application performance.
- The Overview section of this live migration guide contains all the steps that are needed to finish the database migration from Amazon Aurora MySQL to HeatWave MySQL on OCI.
- In the Walkthrough section of this live migration guide, we will apply the information provided in the Overview section and give you a simple step-by-step guide. In this step-by-step guide, we will have an Amazon Aurora MySQL instance with some sample data pre-loaded and will migrate it over to HeatWave MySQL on OCI. This will help you follow and better visualize the process/information provided in the Overview section.
- You can use the Walkthrough section's step-by-step guide as a reference for your migration from Amazon Aurora MySQL to HeatWave MySQL. When following the guide, make changes along the way to your AWS and OCI environments accordingly or as required. Since each user following the step-by-step guide will have their environments configured differently, we cannot provide an ideal example that works for everyone.

Overview:

Following are the required steps to migrate data from Amazon Aurora MySQL to HeatWave MySQL on OCI using live migration (with zero or minimal downtime):

I) Have an Oracle Cloud Infrastructure (OCI) account and Amazon Web Services (AWS) account.

OCI Sign in/Sign up page: https://cloud.oracle.com
AWS Sign in/Sign up page: https://aws.amazon.com/

II) Set up a VPN connection from OCI to AWS.

[A VPN connection will allow you to bridge your AWS network with the OCI VCN. The VPN connection will allow your Amazon Aurora MySQL to connect to HeatWave MySQL on OCI and it also ensures that your data in transit is encrypted while it is being migrated.]
VPN Connection to AWS: https://docs.oracle.com/en-us/iaas/Content/Network/Tasks/vpn_to_aws.htm

III) On OCI, create a standalone HeatWave MySQL instance.

[If you require High Availability for your HeatWave MySQL instance, you must enable it after completing section **VIII)** of this guide.]
Provision HeatWave MySQL on OCI: https://docs.oracle.com/en-us/iaas/mysql-database/doc/creating-db-system1.html

IV) Install MySQL Shell 8.2 or above on an EC2 instance that can connect to Amazon Aurora MySQL.

[MySQL Shell on the EC2 instance will be used to copy DDL and data from Amazon Aurora MySQL to HeatWave MySQL on OCI. You must download MySQL Shell 8.2 or above.]
Download MySQL Shell: https://dev.mysql.com/downloads/shell/
Install MySQL Shell: https://dev.mysql.com/doc/mysql-shell/8.2/en/mysql-shell-install.html

V) For your Amazon Aurora MySQL, ensure `log_bin` is set to `1`, ensure `binlog_format` is set to `ROW`, and execute the `mysql.rds_set_configuration` stored procedure to retain binary logs.

[The Aurora binary logs are needed to set up replication from Aurora to HeatWave MySQL for data synchronization. The Aurora binary logs need to be retained until replication is set up from Aurora to HeatWave MySQL and all the pending transactions from Aurora have been replicated to HeatWave MySQL.]
Aurora Binary Logs Stored Procedure:
https://docs.aws.amazon.com/AmazonRDS/latest/AuroraUserGuide/USER_LogAccess.MySQL.Binarylog.html

VI) Connect to Amazon Aurora MySQL using MySQL Shell and create a replication user. Afterwards, execute the MySQL Shell `util.copyInstance()` utility to export all schemas (including users, indexes, routines, triggers) from Amazon Aurora MySQL to HeatWave MySQL on OCI. After the `util.copyInstance()` utility finishes, save the MySQL Shell `Dump_metadata` values.

[The dump created by MySQL Shell's instance copy utility comprises DDL files specifying the schema structure, and tab-separated `.tsv` files containing the data. MySQL Shell's `Dump_metadata` values will let the HeatWave MySQL instance on OCI know where to start the replication from for data synchronization.]
MySQL Shell Copy Utilities: https://dev.mysql.com/doc/mysql-shell/8.2/en/mysql-shell-utils-copy.html

VII) On OCI, create a replication channel to set up replication from Amazon Aurora MySQL to HeatWave MySQL on OCI. During the channel creation process, if the Aurora instance is using binary log positioning - under the replication positioning section, select Source cannot use GTID auto-positioning and provide the binlogFile and binlogPosition values. If the Aurora instance is using GTIDs - select Source can use GTID auto-positioning (recommended). Create the replication channel afterwards.

[Setting up this replication channel will propagate all the pending data changes to HeatWave MySQL that had occurred on the Aurora instance after the execution of MySQL Shell `util.copyInstance()` utility.]
Create OCI Replication Channel: https://docs.oracle.com/en-us/iaas/mysql-database/doc/creating-replication-channel.html#GUID-521ECA6C-4528-4DE9-8928-D9620893872A

VIII) After the replication channel is up, connect to HeatWave MySQL and execute the `SHOW REPLICA STATUS\G` command. From the query output, look for the `seconds_behind_source` and `Replica_SQL_Running_State` fields. If the `seconds_behind_source` field displays a value of `0` and the `Replica_SQL_Running_State` field displays a message of `Replica has read all relay log; waiting for more updates` - this indicates that the HeatWave MySQL instance has fully caught up with the Amazon Aurora MySQL changes and the replication channel can now be disabled.

[During this step, it is recommended to stop the database application for ~5 minutes to ensure that no writes are happening to the Aurora instance before the replication channel between HeatWave MySQL and Aurora is disabled. After the replication channel has been disabled, you may turn on High Availability for your HeatWave MySQL instance.]
MySQL Replica Replication Status: https://dev.mysql.com/doc/refman/8.0/en/show-replica-status.html
Disabling OCI Replication Channel: https://docs.oracle.com/en-us/iaas/mysql-database/doc/managing-replication-channel.html#GUID-4CD38EFA-7463-4175-8838-0EE40C0FABC9

IX) At this point, the live migration process for the database is complete. The database applications can now point to HeatWave MySQL on OCI.

X) (Optional) On OCI, if the HeatWave option was enabled during HeatWave MySQL DB creation, add the HW Cluster and load data from the MySQL InnoDB storage into the HW Cluster using automation.

[Attaching the HeatWave in-memory Cluster combines transactions, analytics, and machine learning services into one MySQL Database.]
Add a HeatWave Cluster: https://docs.oracle.com/en-us/iaas/mysql-database/doc/adding-heatwave-cluster.html#GUID-2335AC1F-FB01-4701-9EFD-810A3489A850
Load Data into HeatWave: https://dev.mysql.com/doc/heatwave/en/mys-hw-auto-parallel-load.html

Walkthrough:

I) Have an Oracle Cloud Infrastructure (OCI) account and Amazon Web Services (AWS) account.

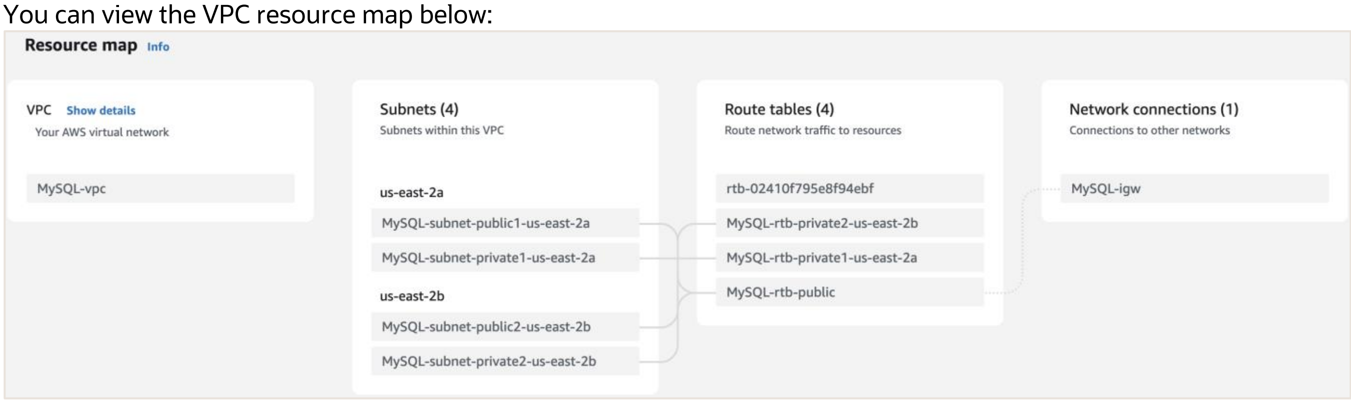
OCI Sign in/Sign up page: https://cloud.oracle.com
AWS Sign in/Sign up page: https://aws.amazon.com/

II) Set up a VPN connection from OCI to AWS.

1. Below is the Amazon Aurora MySQL instance version and [the sample database ("world")](#) that will be migrated for this guide. The sample world database consists of 3 tables. The Amazon Aurora MySQL instance used for this does not have public access.

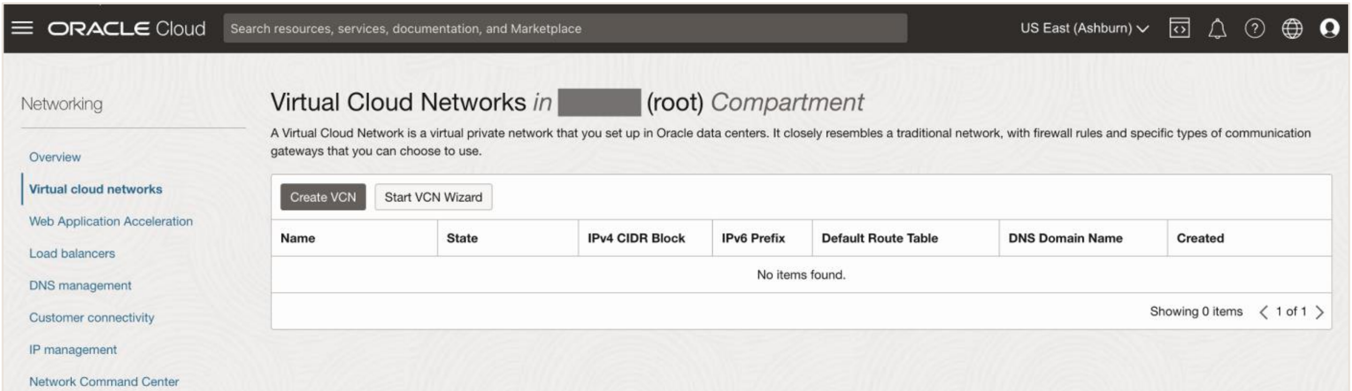
```
 MySQL  database-1-instance-1.[redacted].us-east-2.rds  SQL > SELECT @@VERSION;
+------------+
| @@VERSION  |
+------------+
| 5.7.12-log |
+------------+
1 row in set (0.0006 sec)
 MySQL  database-1-instance-1.[redacted].us-east-2.rds  SQL > SHOW SCHEMAS;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| mysql              |
| performance_schema |
| sys                |
| world              |
+--------------------+
5 rows in set (0.0014 sec)
 MySQL  database-1-instance-1.[redacted].us-east-2.rds  SQL > SHOW TABLES IN world;
+-----------------+
| Tables_in_world |
+-----------------+
| city            |
| country         |
| countrylanguage |
+-----------------+
3 rows in set (0.0007 sec)
```

2. The AWS VPC associated with the above Amazon Aurora MySQL instance uses an IPv4 CIDR: 10.1.0.0/16. You can view the VPC resource map below:

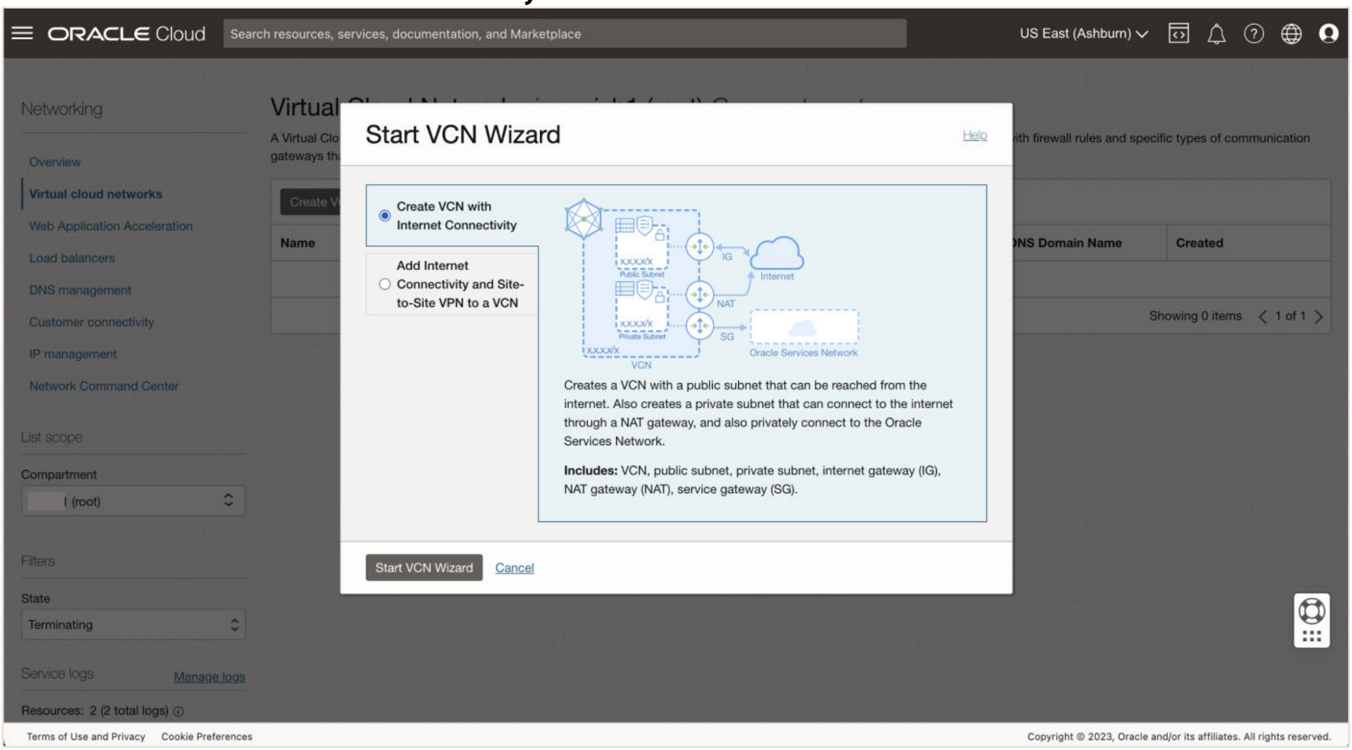


3. Log in to [OCI](#) and create a VCN. Open the OCI navigation menu, click **Networking**, and click **Virtual cloud networks**.

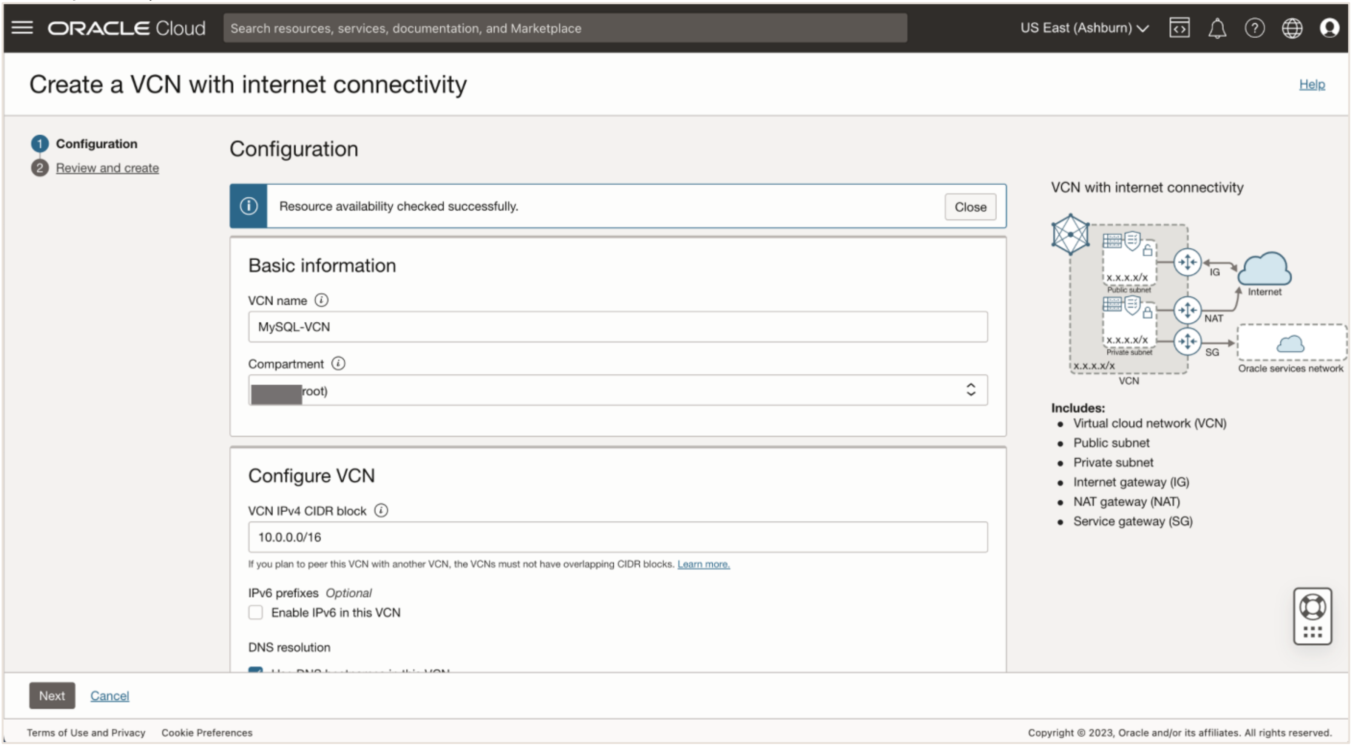
4. Ensure you are in your desired compartment - we have chosen the `root` compartment. Click **Start VCN Wizard**.



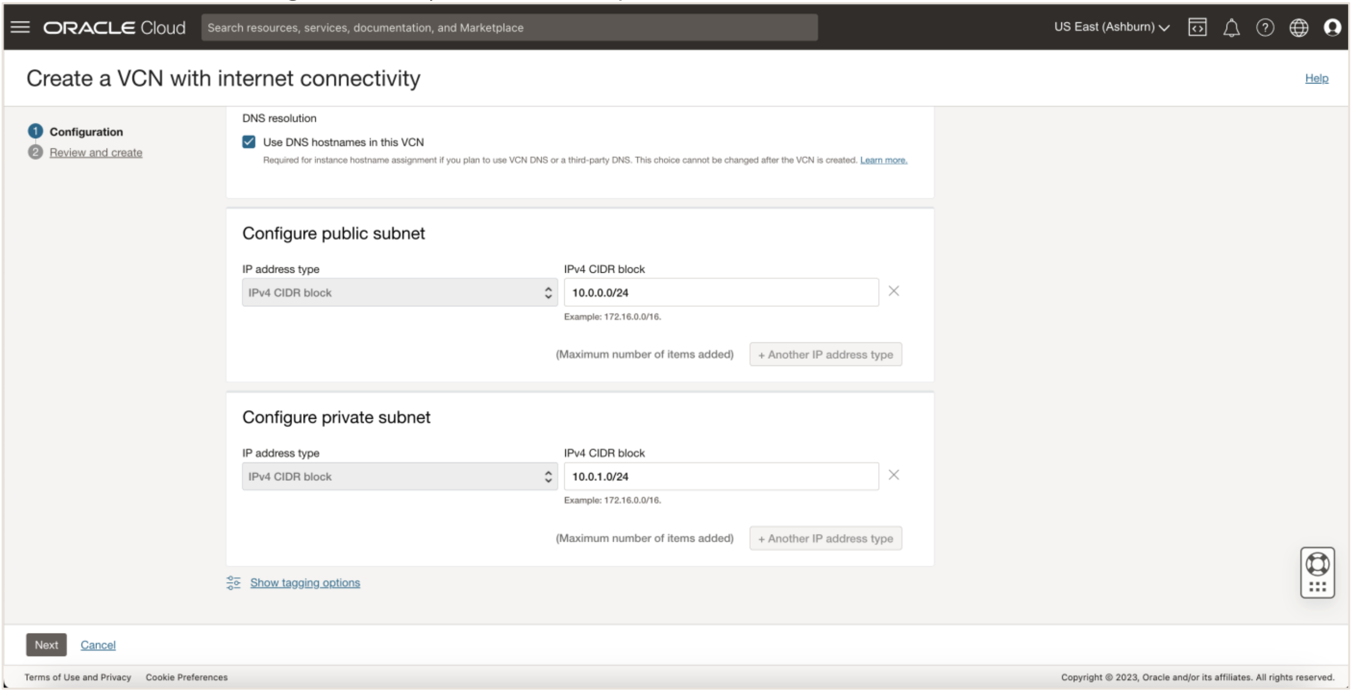
5. Select **Create VCN with Internet Connectivity** and click **Start VCN Wizard**.



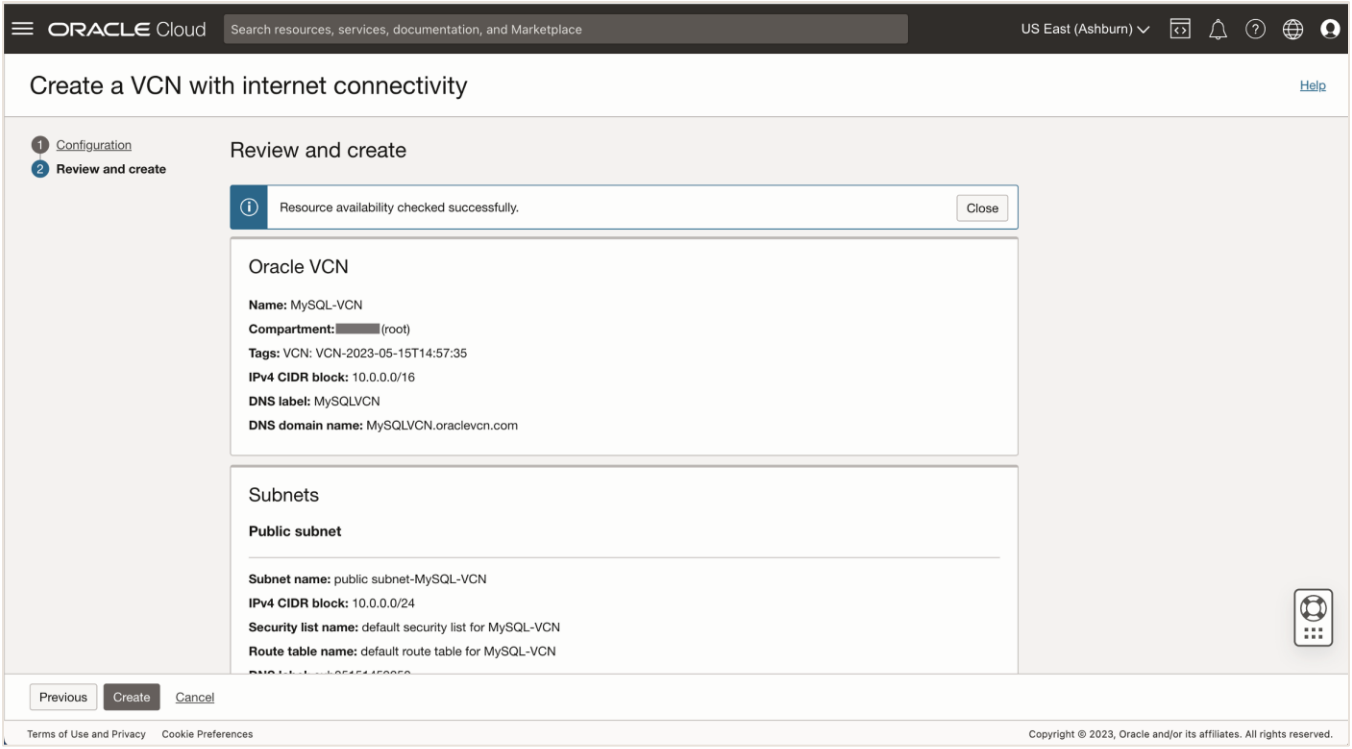
6. Enter a **VCN name** and **configure your VCN's IPv4 CIDR block** - **including the public and the private subnet**. The guide uses the default values for all. Make sure that the OCI VCN IPv4 CIDR block does not overlap with your AWS VPC IPv4 CIDR.



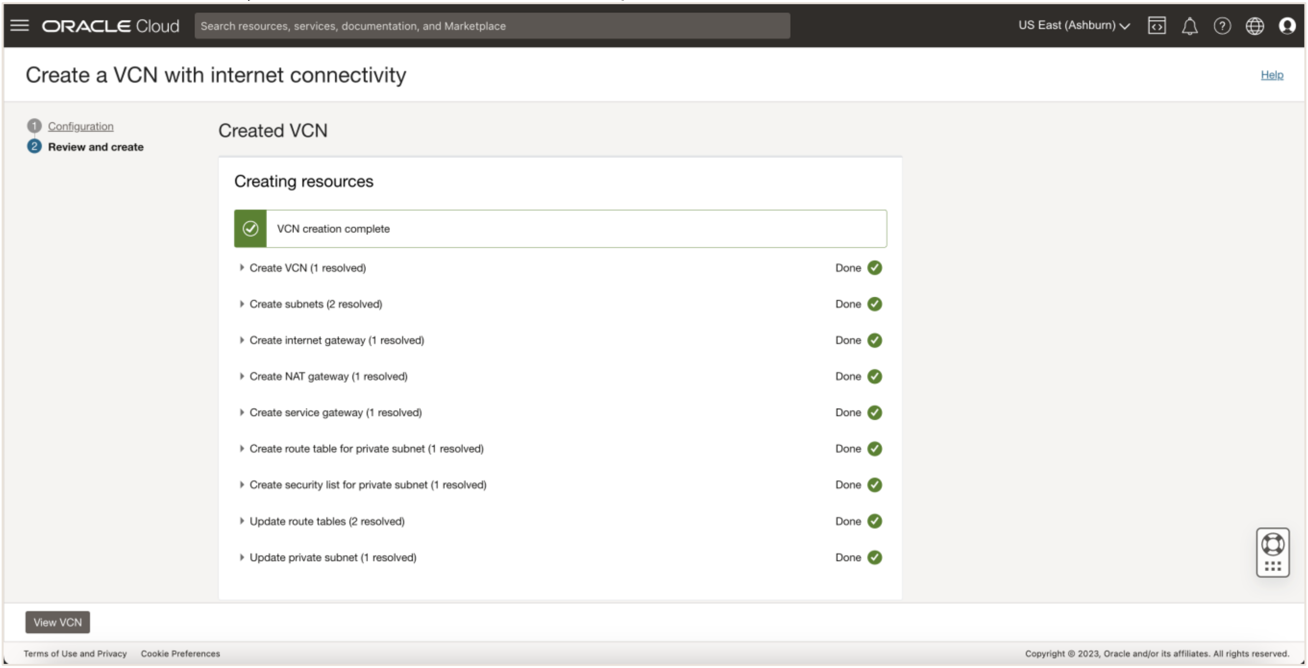
7. Click **Next** after the configuration for your VCN is completed.



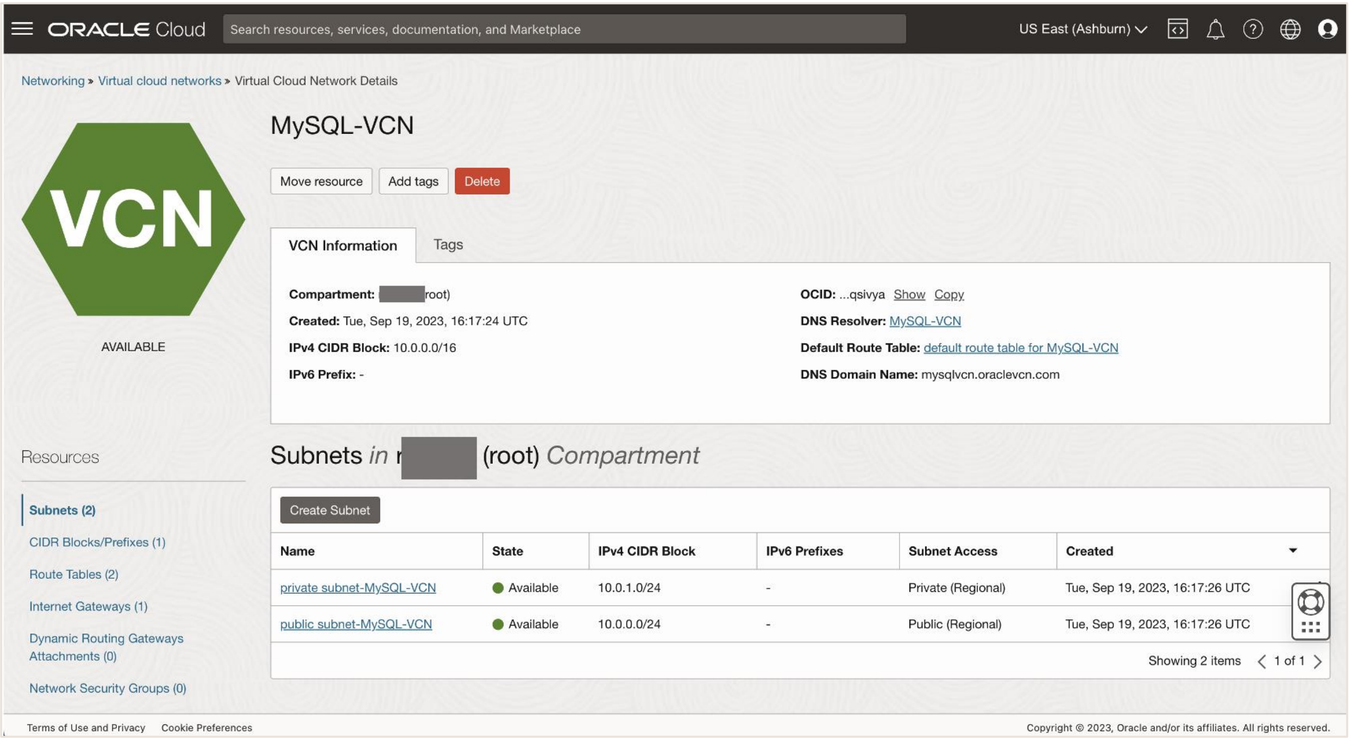
8. On the Review and create page, validate the information for your VCN and click **Create**.



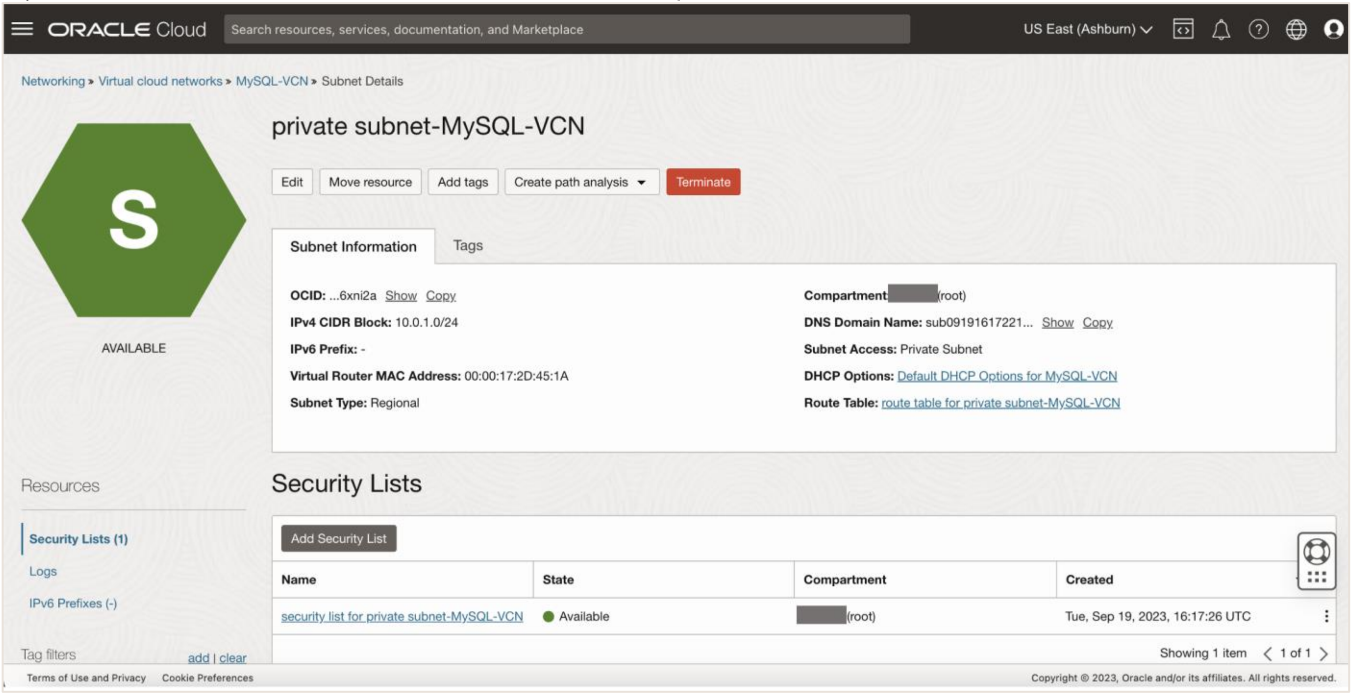
9. Click **View VCN** after your VCN creation has been completed.



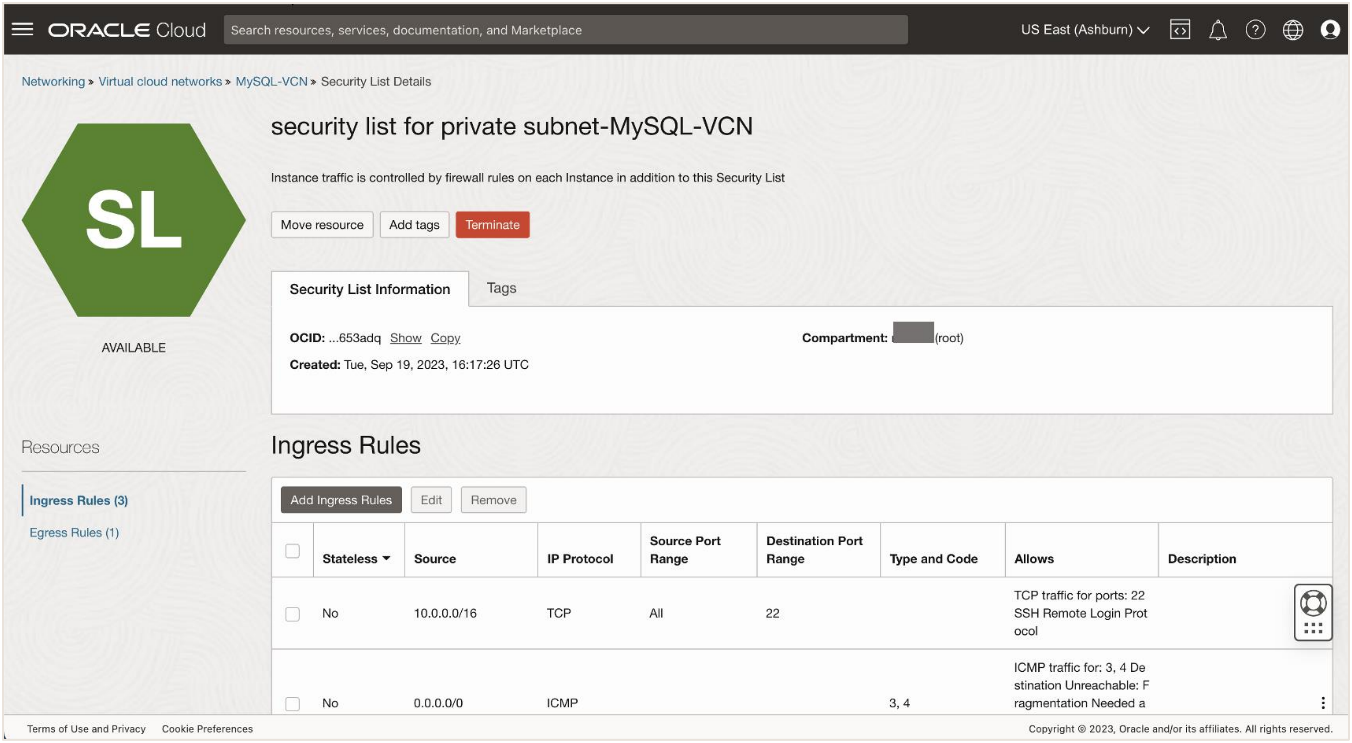
10. On the Virtual Cloud Network Details page under Resources, click **Subnets** section. Click on **private subnet-<vcn-name>**.



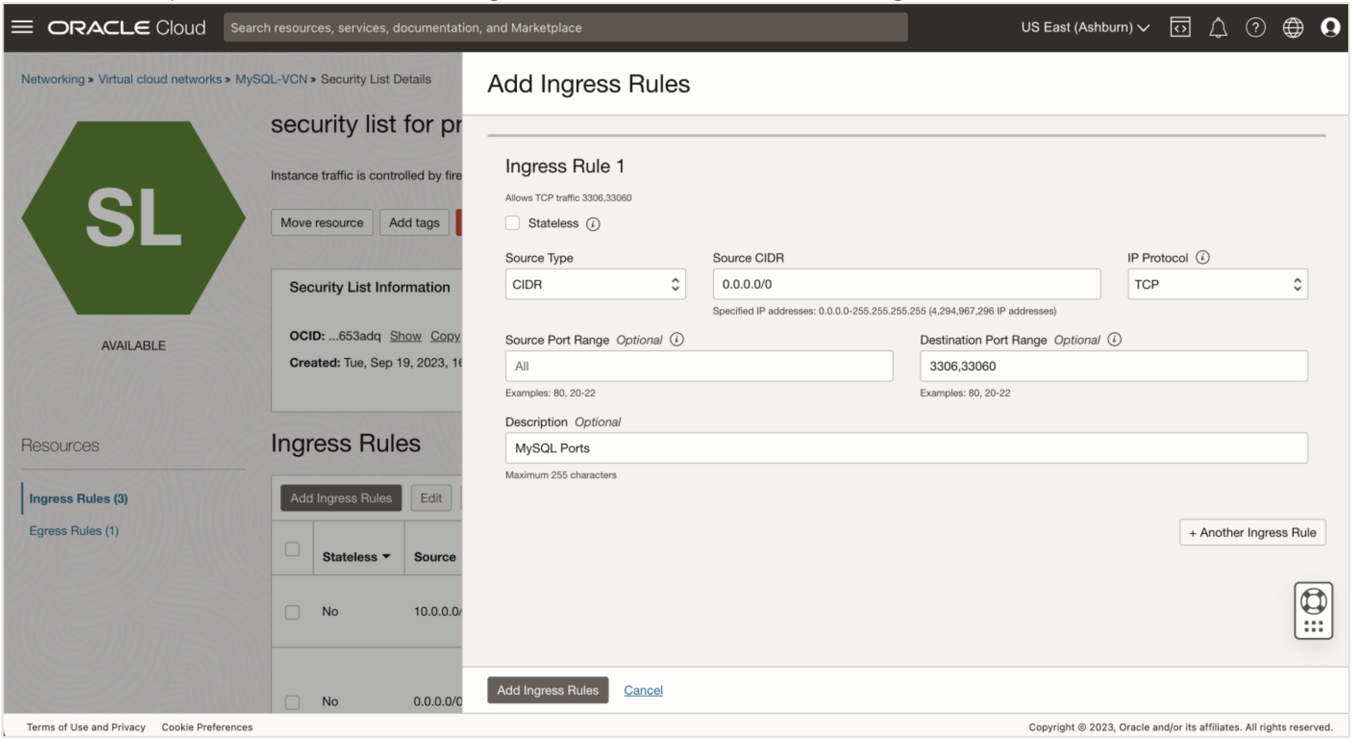
11. Click on **security list for private subnet-<vcn-name>** to add an Ingress Rule which will allow HeatWave MySQL to access the Aurora instance on AWS and the Compute instance on OCI.



12. Click **Add Ingress Rules**.

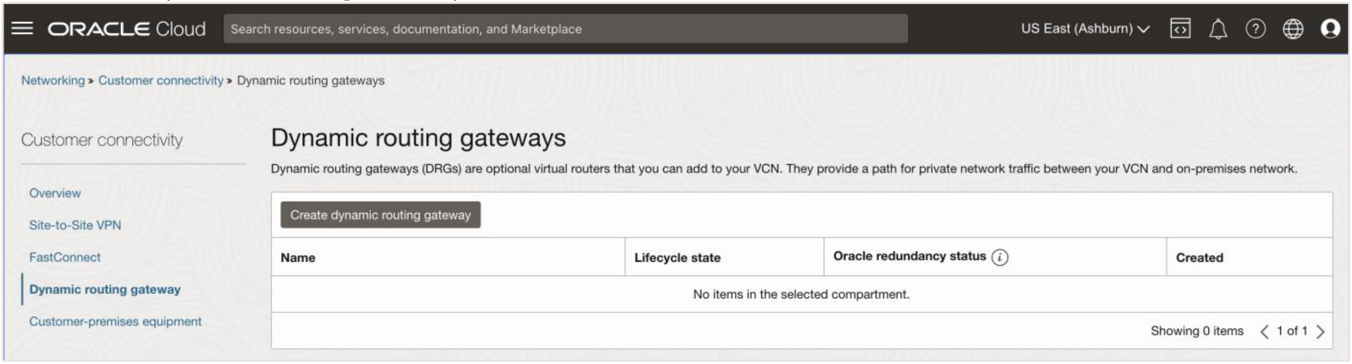


13. For **Source CIDR** type **0.0.0.0/0** (you can be more restrictive here and enter only the AWS and OCI VPC and VCN IPv4 CIDR). For **Destination Port Range**, enter **3306,33060**. Click **Add Ingress Rules.**

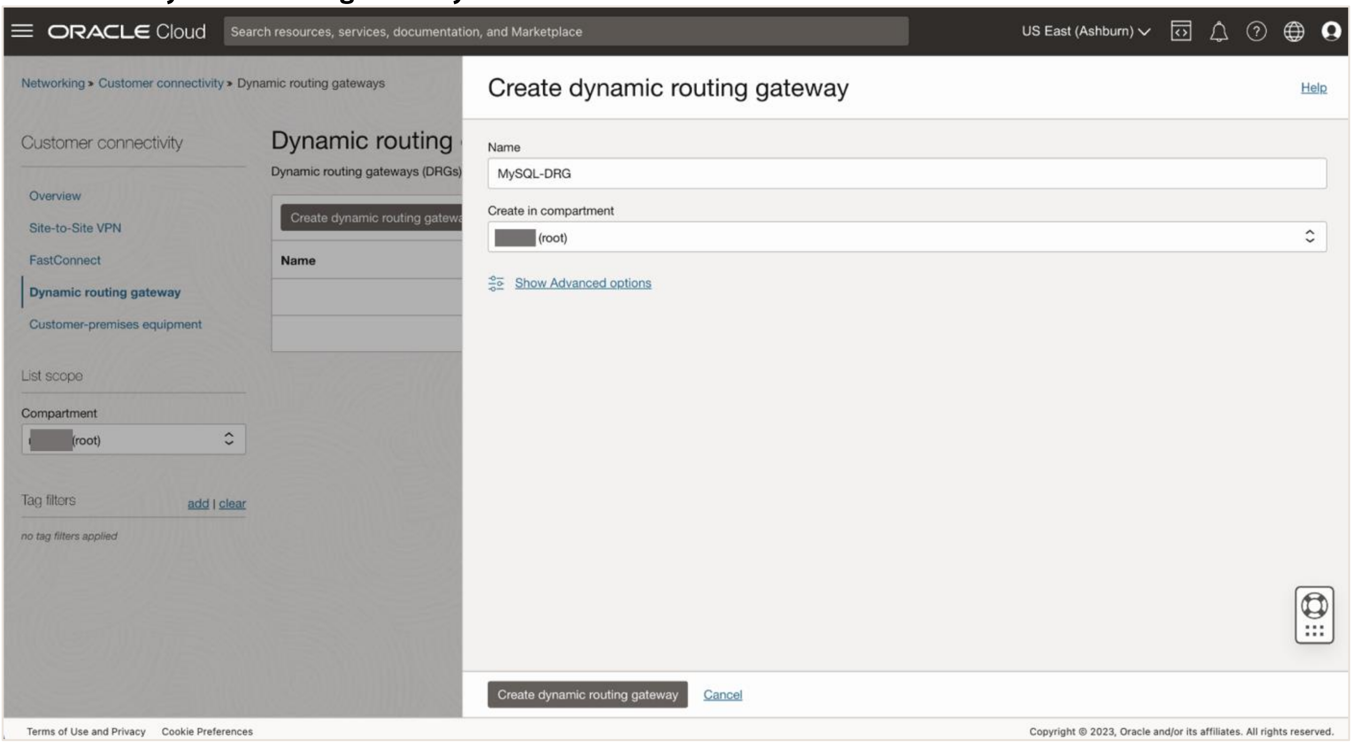


14. Open the OCI navigation menu, click **Networking** and click **Dynamic routing gateway** under Customer Connectivity.

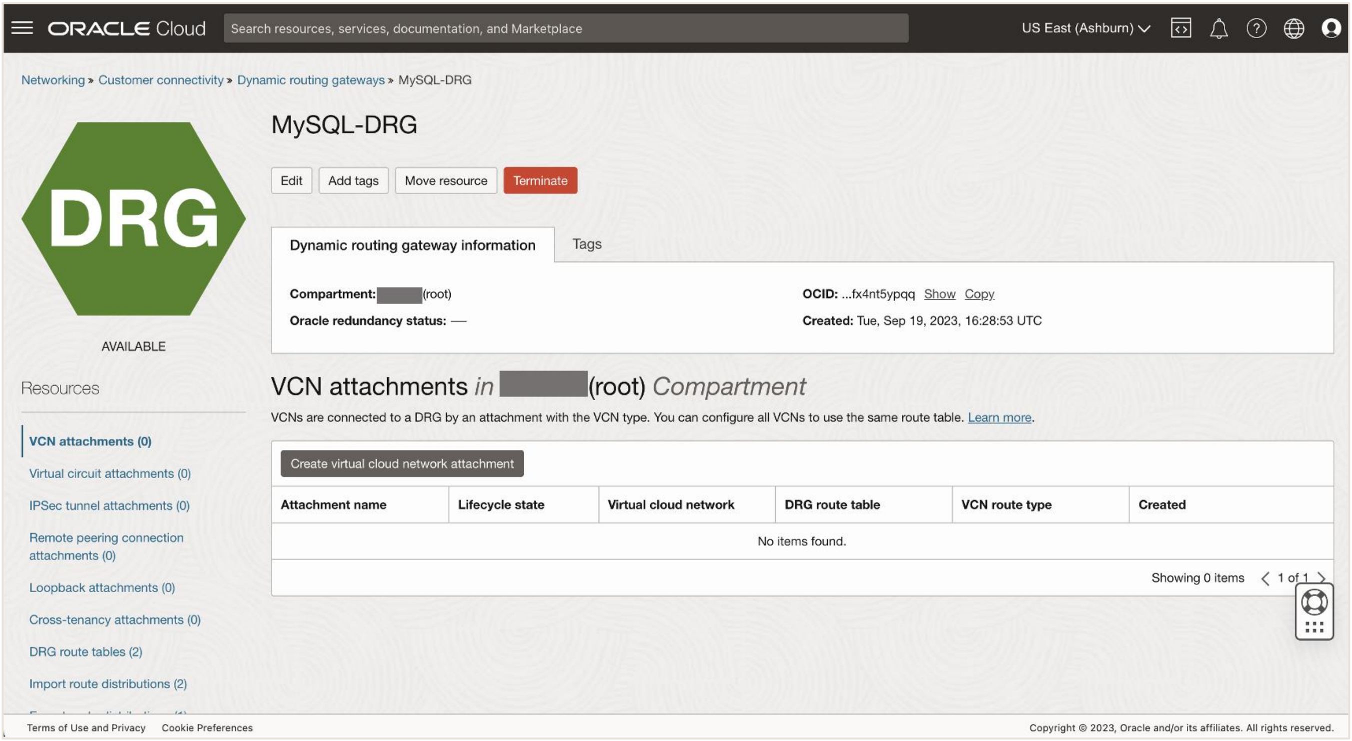
15. Click **Create Dynamic Routing Gateway**.



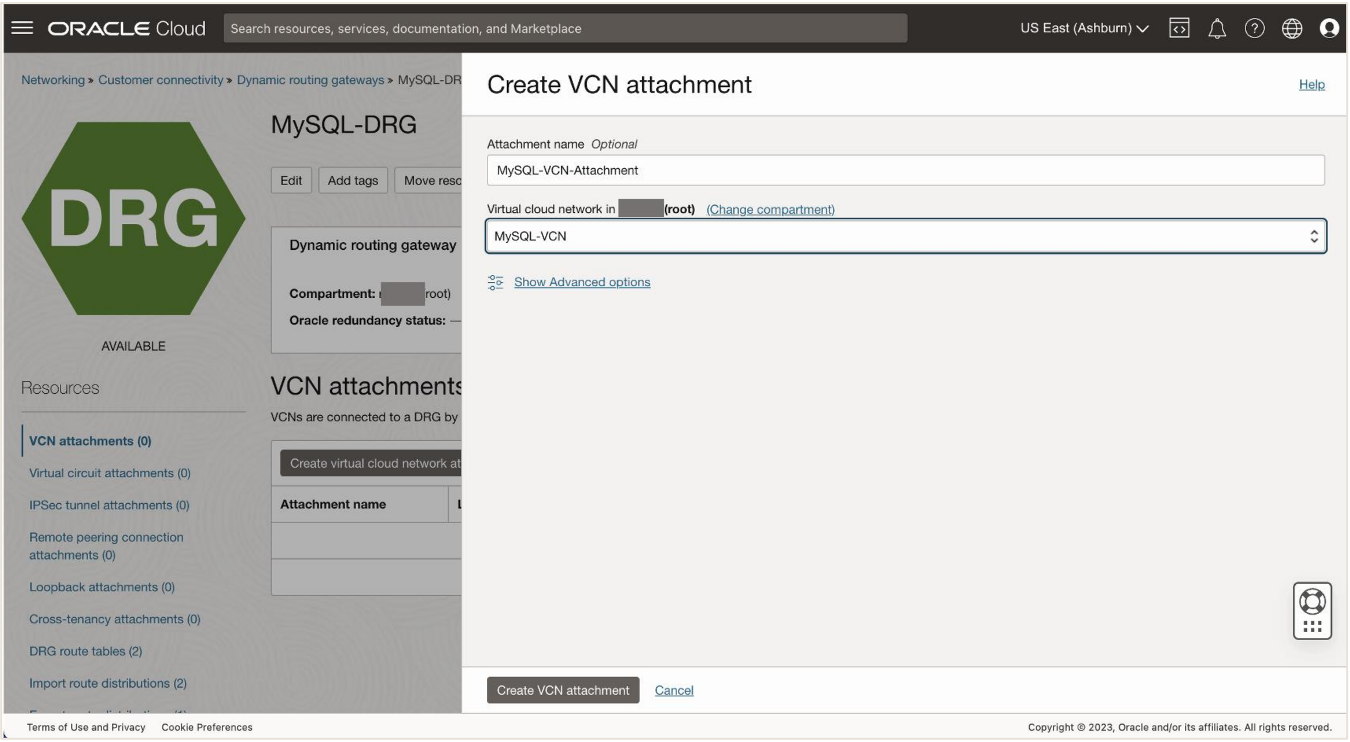
16. Enter a **DRG name**. Under **Create in compartment** - **choose the compartment where your VCN resides**. Click **Create Dynamic Routing Gateway**.



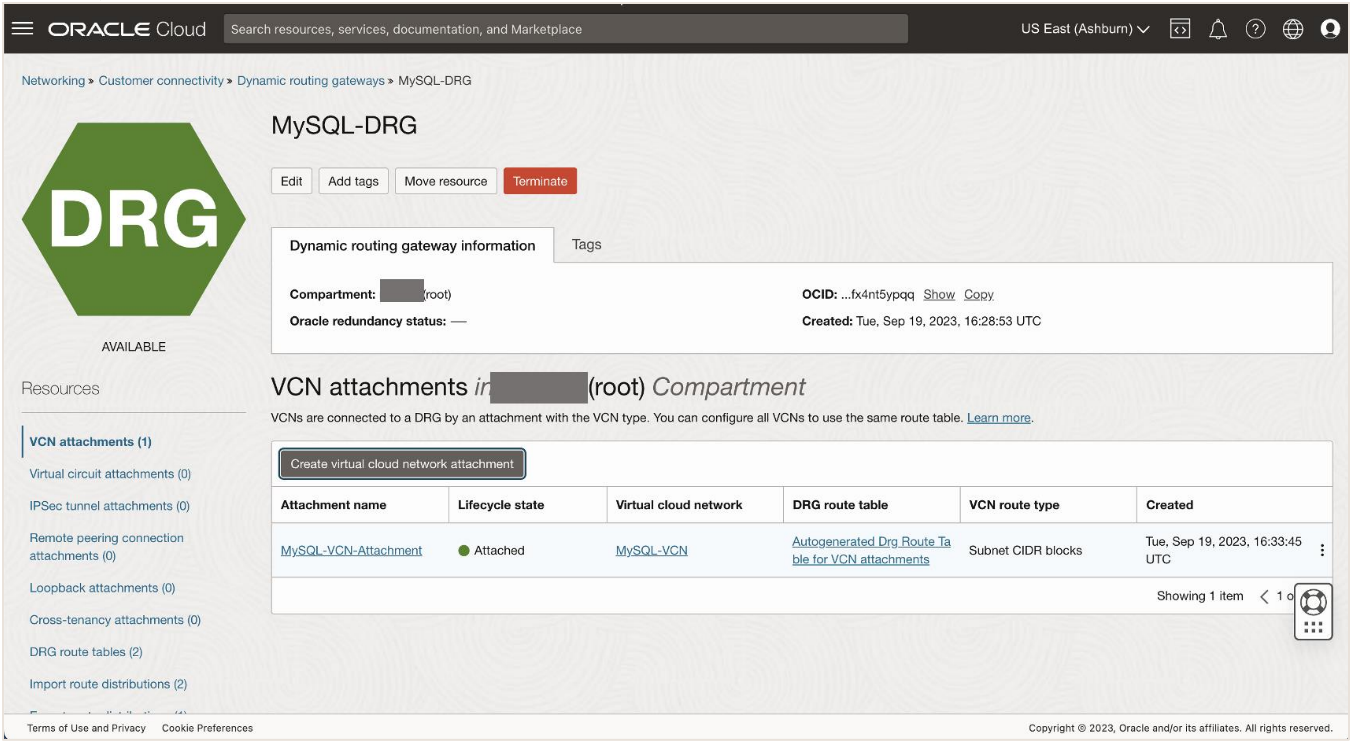
17. You will be taken to the DRG Details page. Once your DRG changes its state from Provisioning to **Available**, under Resources, click **Virtual Cloud Network Attachment**. Click **Create Virtual Cloud Network Attachment**.



18. Enter a **Virtual Cloud Network Attachment name** and **select the appropriate VCN** from the drop-down list. Click **Create Virtual Cloud Network Attachment**.

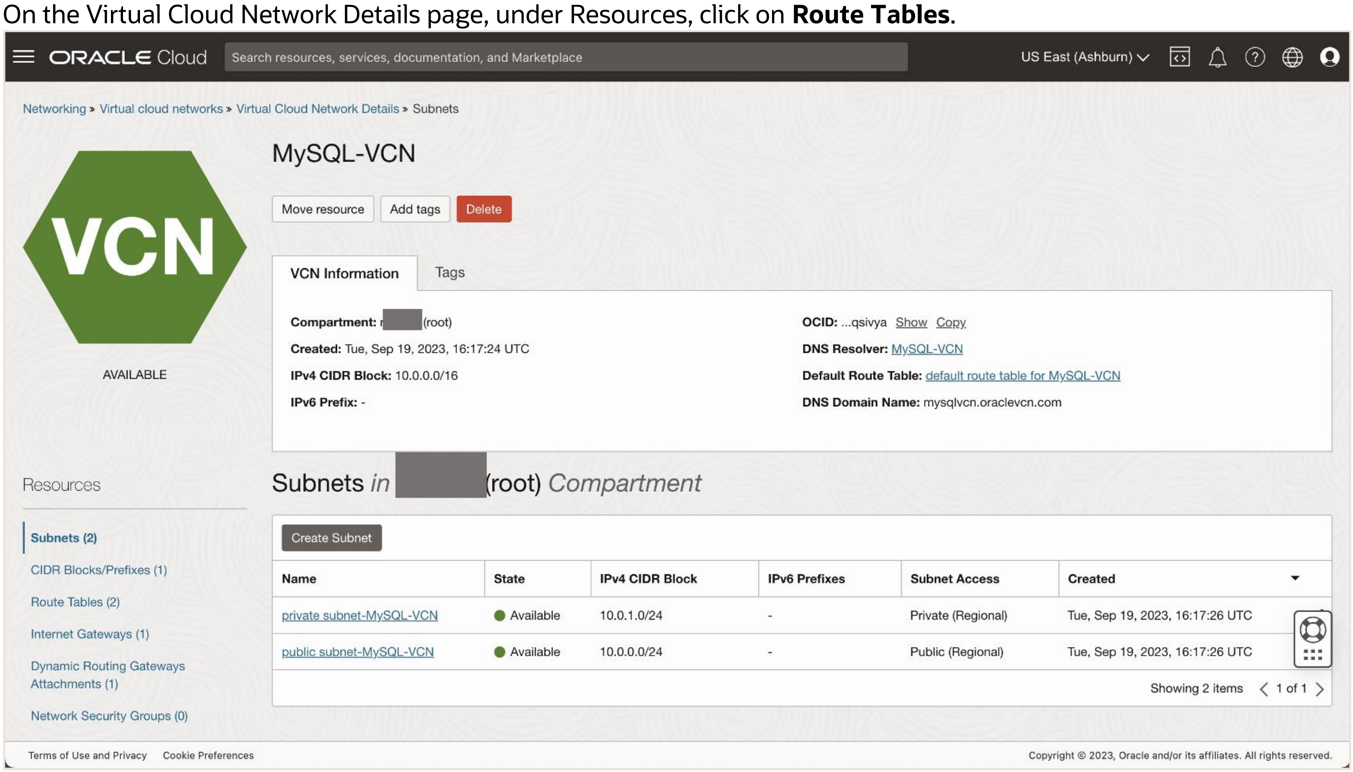


19. Wait for your VCN Attachment to be in an **Attached** state.



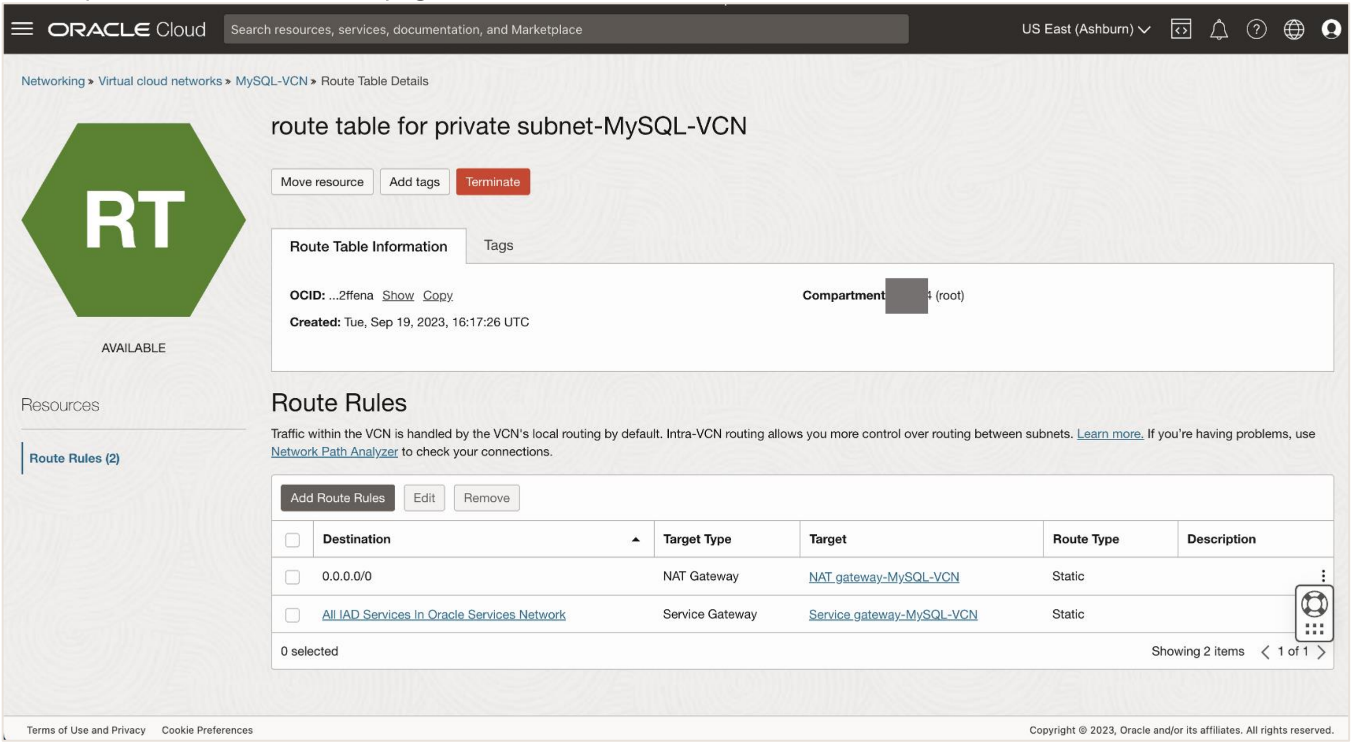
20. Open the OCI navigation menu, click **Networking** and click on **Virtual cloud networks**. After landing on the Virtual Cloud Networks page, click on **the name of your VCN**.

21. On the Virtual Cloud Network Details page, under Resources, click on **Route Tables**.

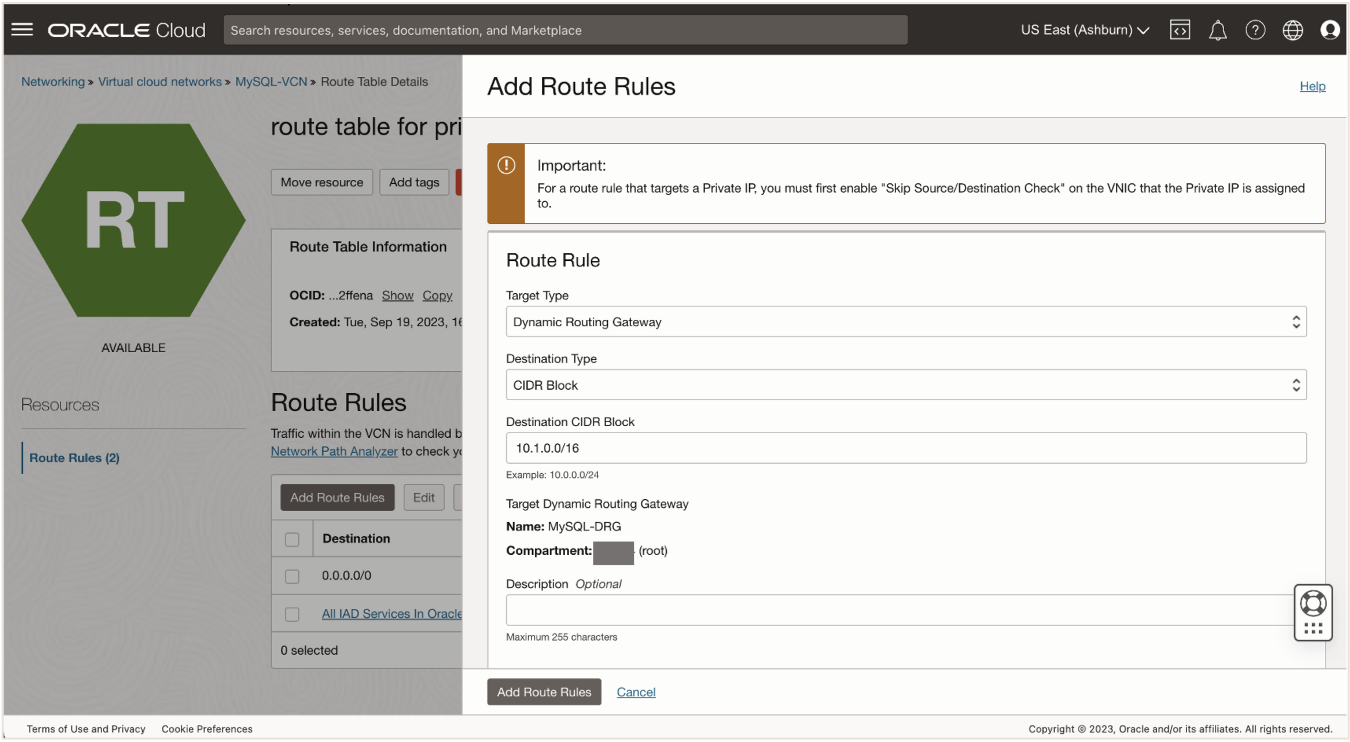


22. You should see two Route Tables, one for your private subnet and the other for your public subnet. Click on **route table for private subnet-<vcn-name>**.

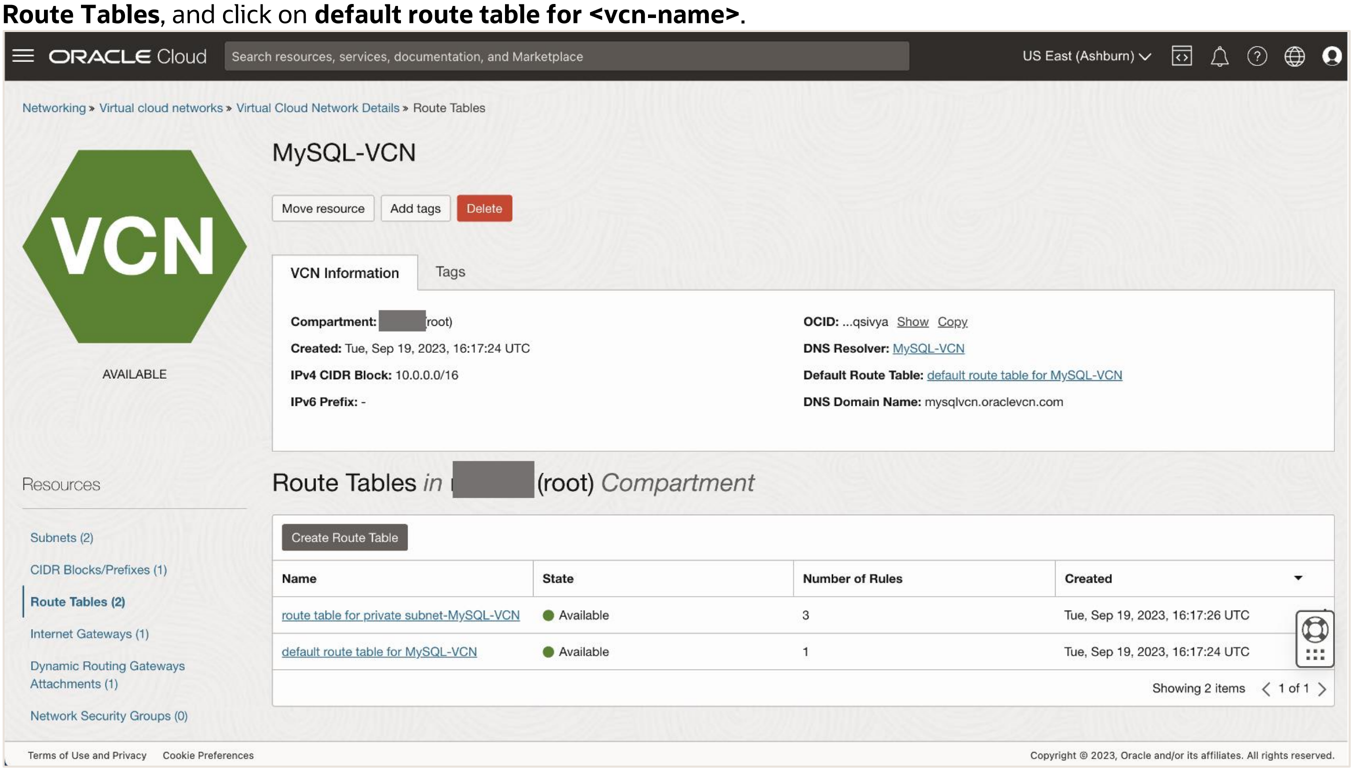
23. On the private subnet route table page, click **Add Route Rules**.



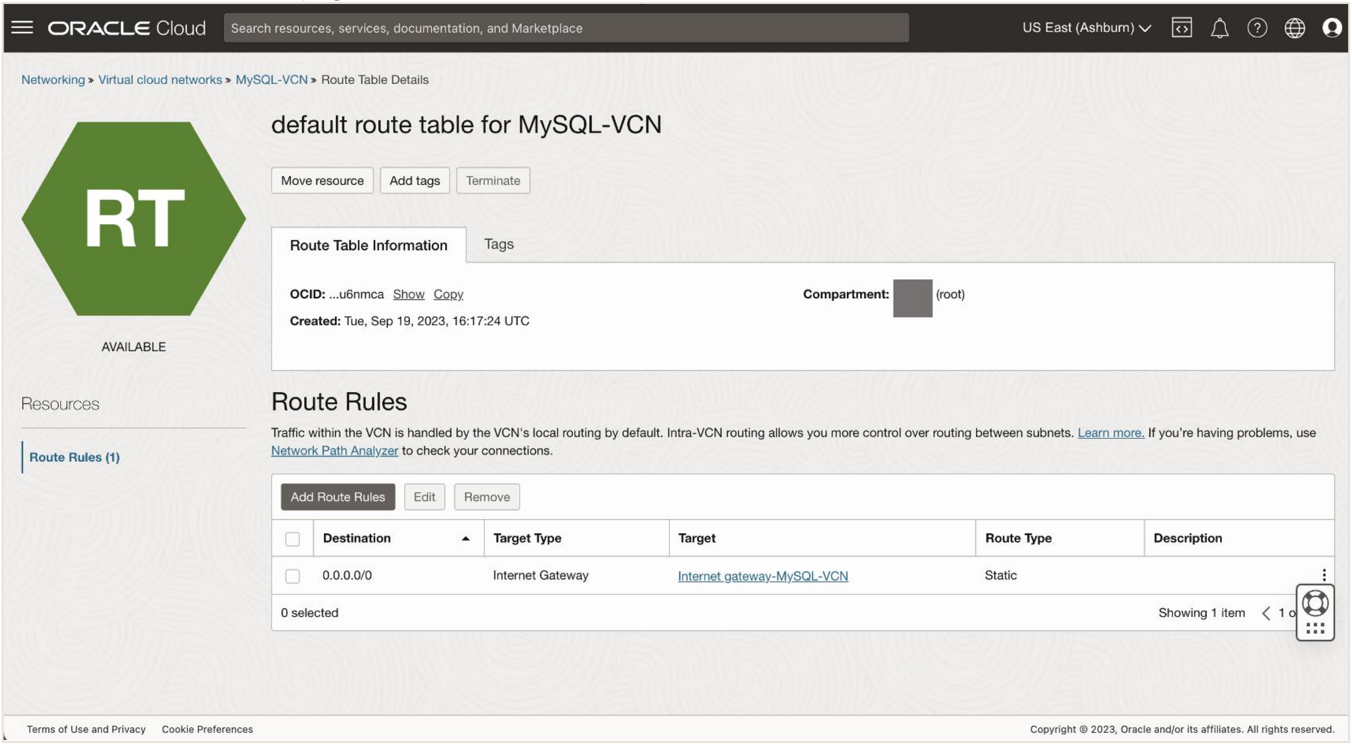
24. Under **Target Type**, select **Dynamic Routing Gateway** from the drop-down list. For **Destination Type**, select **CIDR Block** and for **Destination CIDR Block** - enter your **AWS VPC IPv4 CIDR block** that you will be using to connect to OCI. The AWS VPC CIDR block that will be used for this guide is **10.1.0.0/16**. Click **Add Route Rules** afterwards.



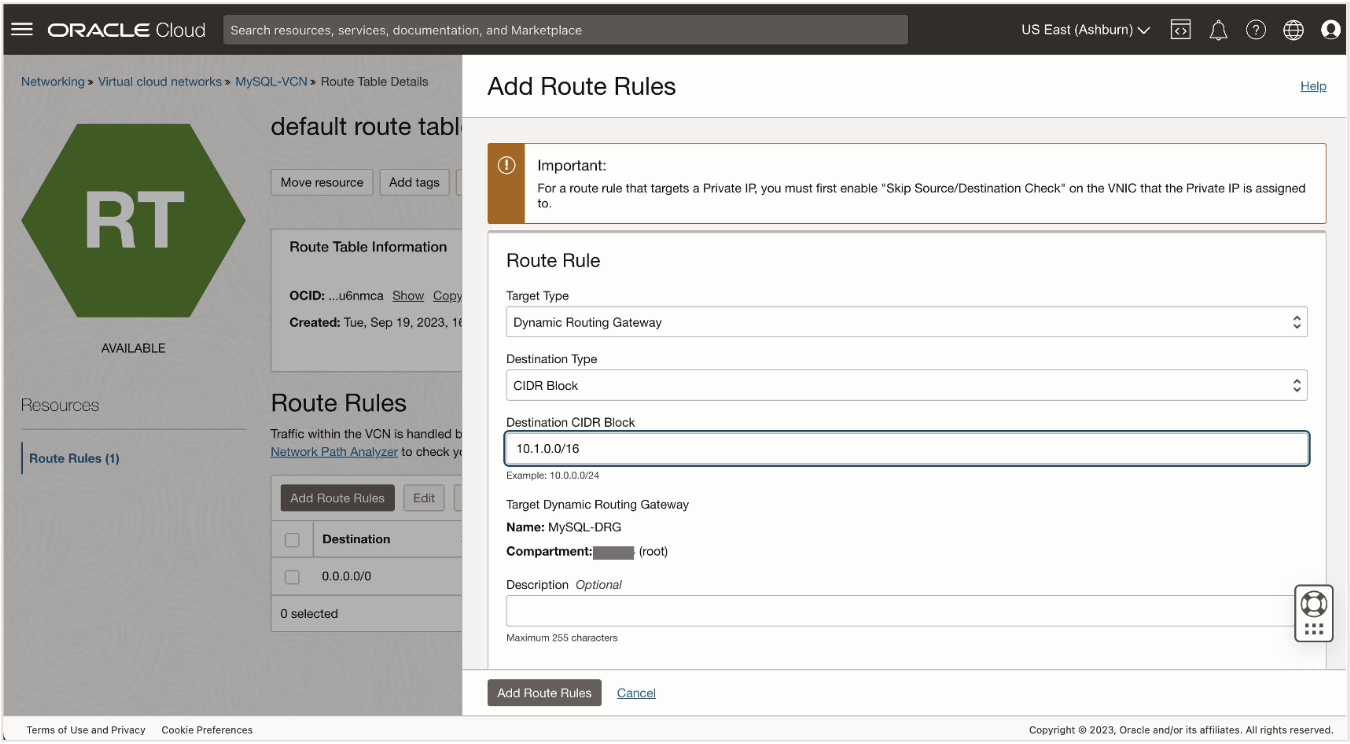
25. Now, repeat the same process for the other route table. Go back to Virtual Cloud Network Details page, click **Route Tables**, and click on **default route table for <vcn-name>**.



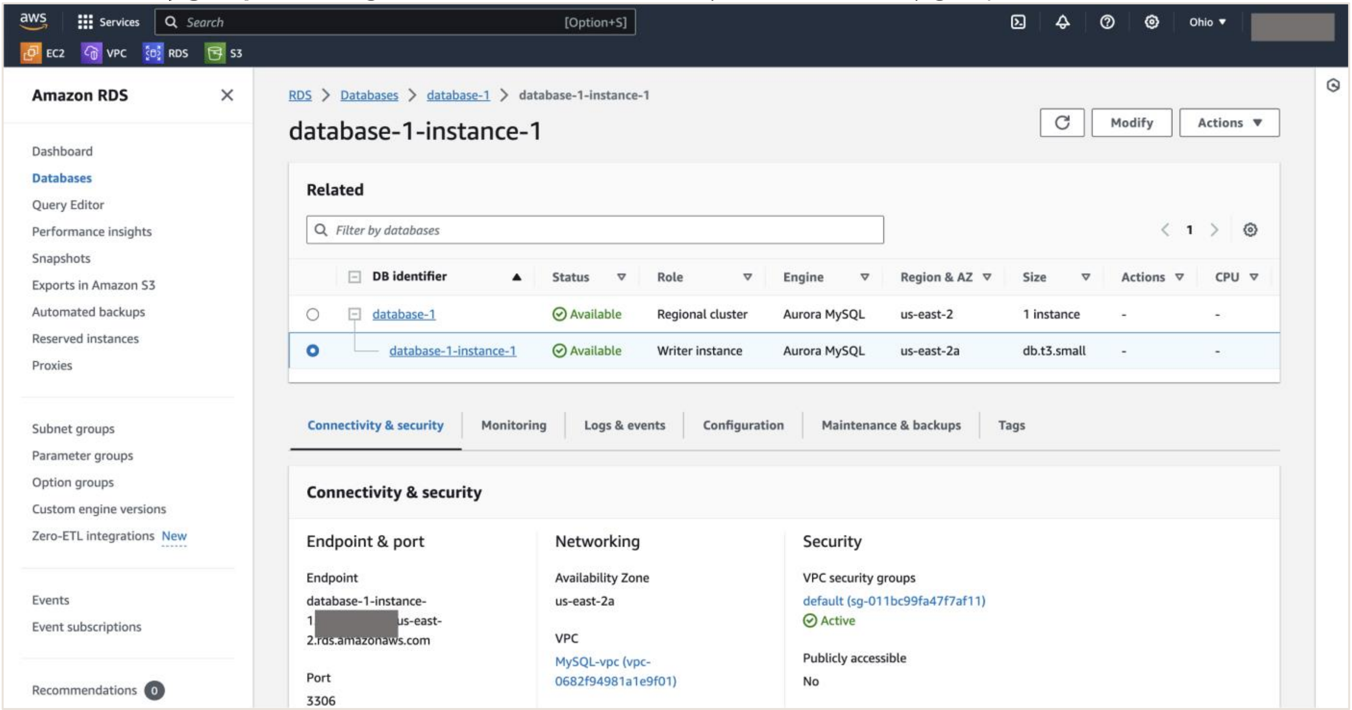
26. On the default route table page, click **Add Route Rules**.



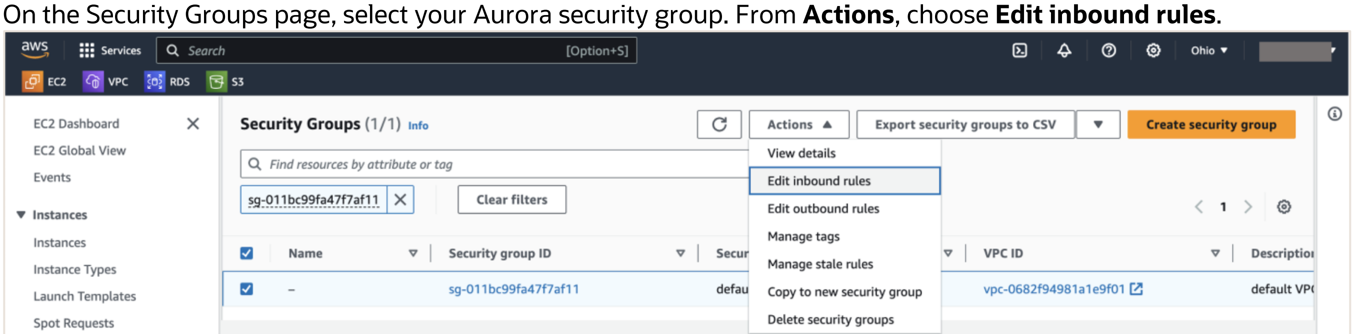
27. Under **Target Type**, **select Dynamic Routing Gateway** from the drop-down list. For **Destination Type**, select **CIDR Block** and for **Destination CIDR Block** - enter your **AWS VPC IPv4 CIDR block** that you will be using to connect to OCI. The AWS VPC CIDR block that will be used for this guide is **10.1.0.0/16**. Click **Add Route Rules** afterwards.



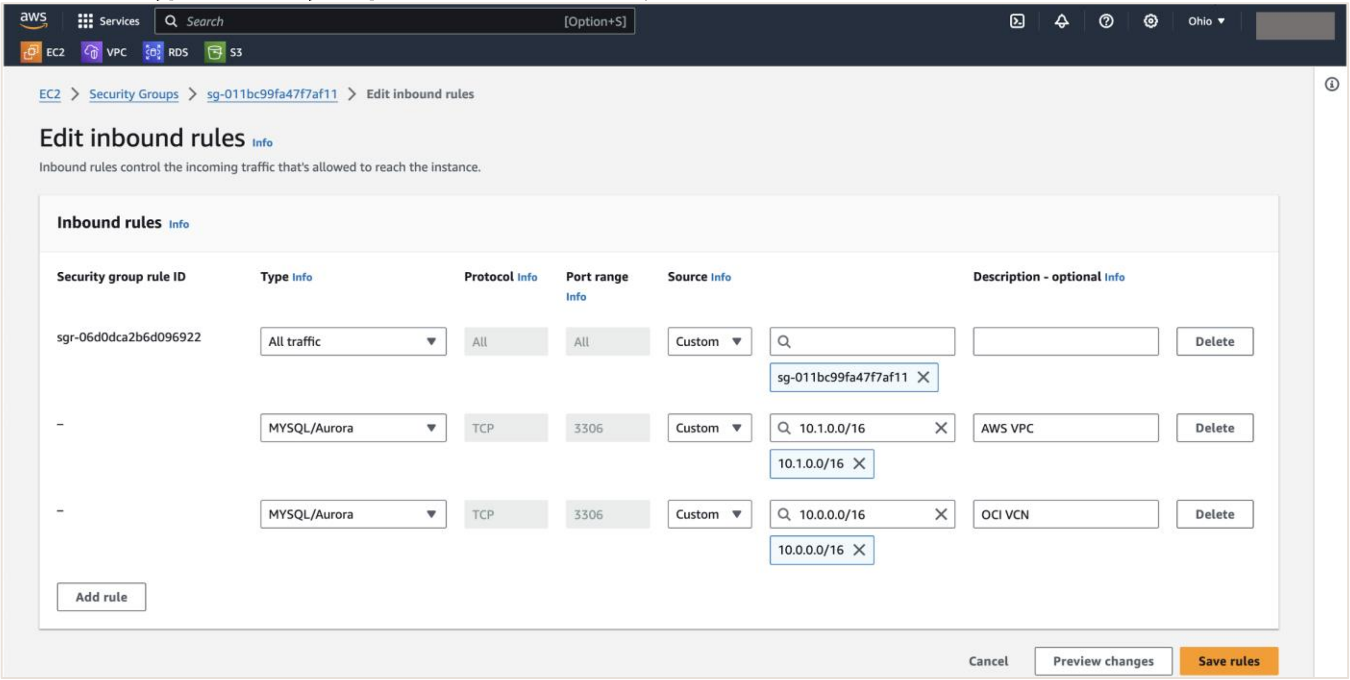
28. Login to [AWS](AWS) to modify the VPC security groups for the Aurora MySQL instance which will allow Aurora to access the HeatWave MySQL instance on OCI and the EC2 instance on AWS. From the main AWS portal, expand the Services menu at the top left of the screen, click **Databases**, click **RDS,** and **select your Aurora instance**. Click **Connectivity & security**, under the **Security** section, look for **VPC security groups** and click on **the security group**. For this guide, our Aurora instance only uses one security group - **default**.



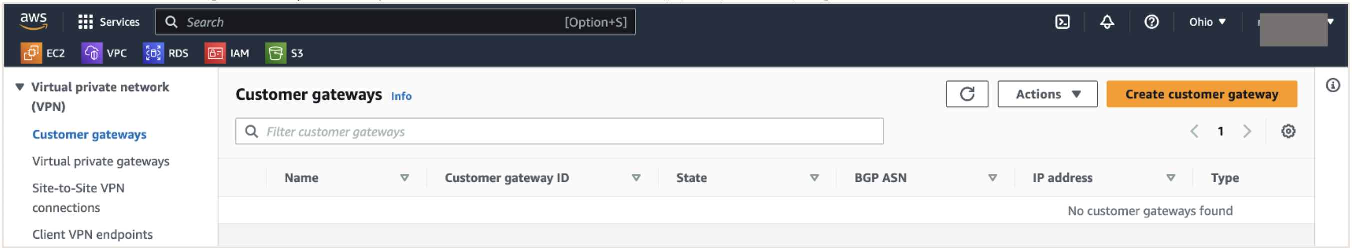
29. On the Security Groups page, select your Aurora security group. From **Actions**, choose **Edit inbound rules**.



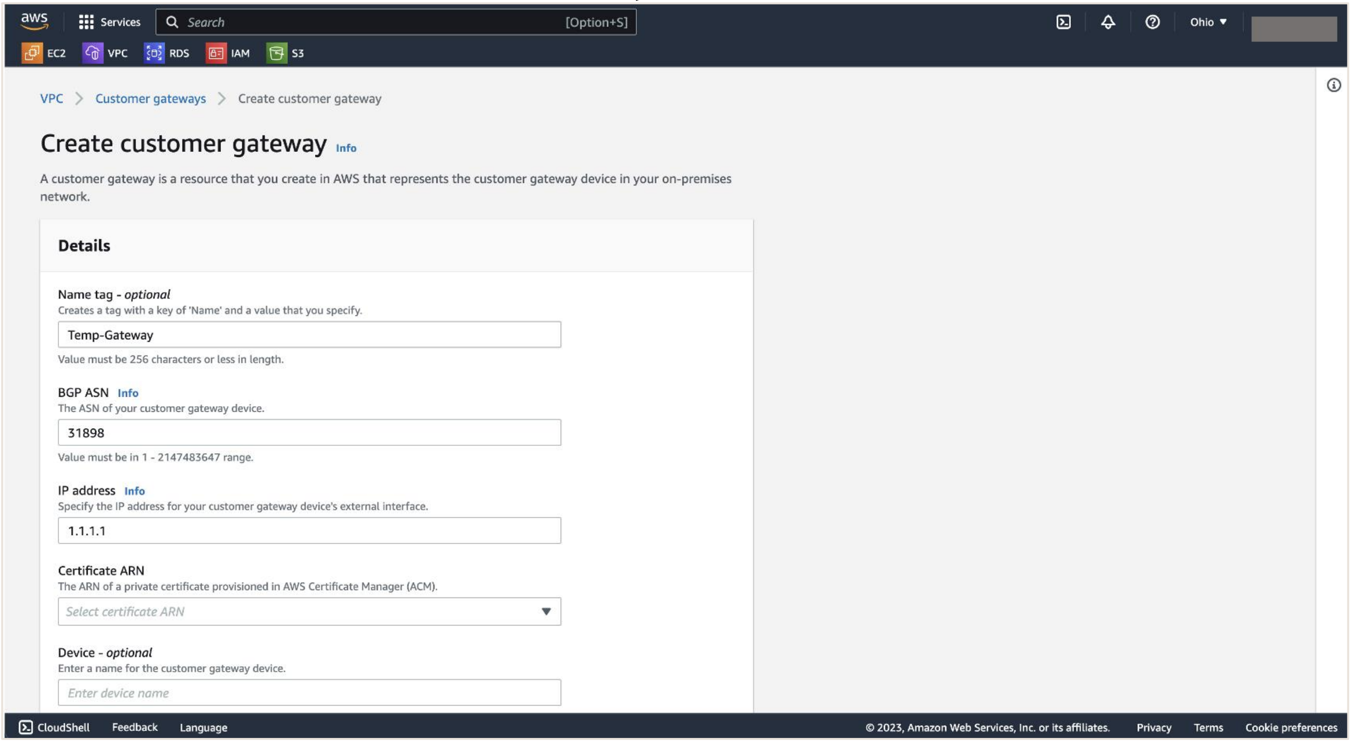
30. Click **Add rule**. Under **Type**, select **MySQL/Aurora**. For **Source**, input the **AWS VPC IPv4 CIDR**. Click **Add rule**. Under **Type**, select **MySQL/Aurora.** For **Source**, input the **OCI VCN IPv4 CIDR block**. Click **Save rules**.



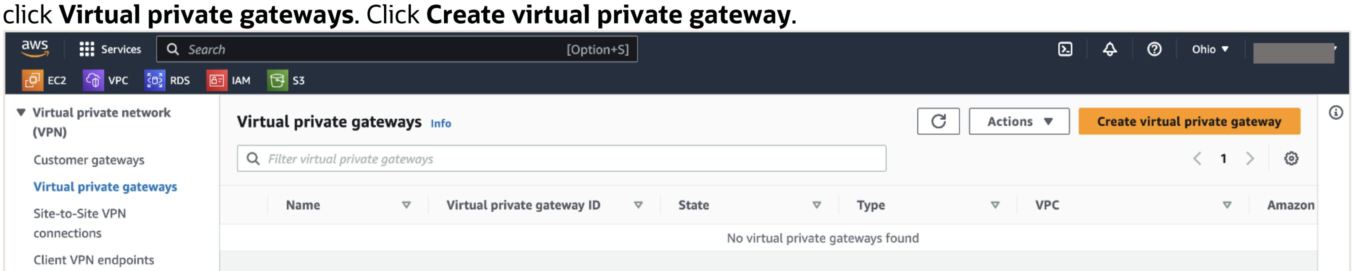
31. From the main AWS Services menu, navigate to **Networking & Content Delivery** and click **VPC**. From the left-hand AWS menu, scroll down and click **Customer Gateways** under Virtual private network (VPN). Click **Create customer gateway** once you have landed on the appropriate page.



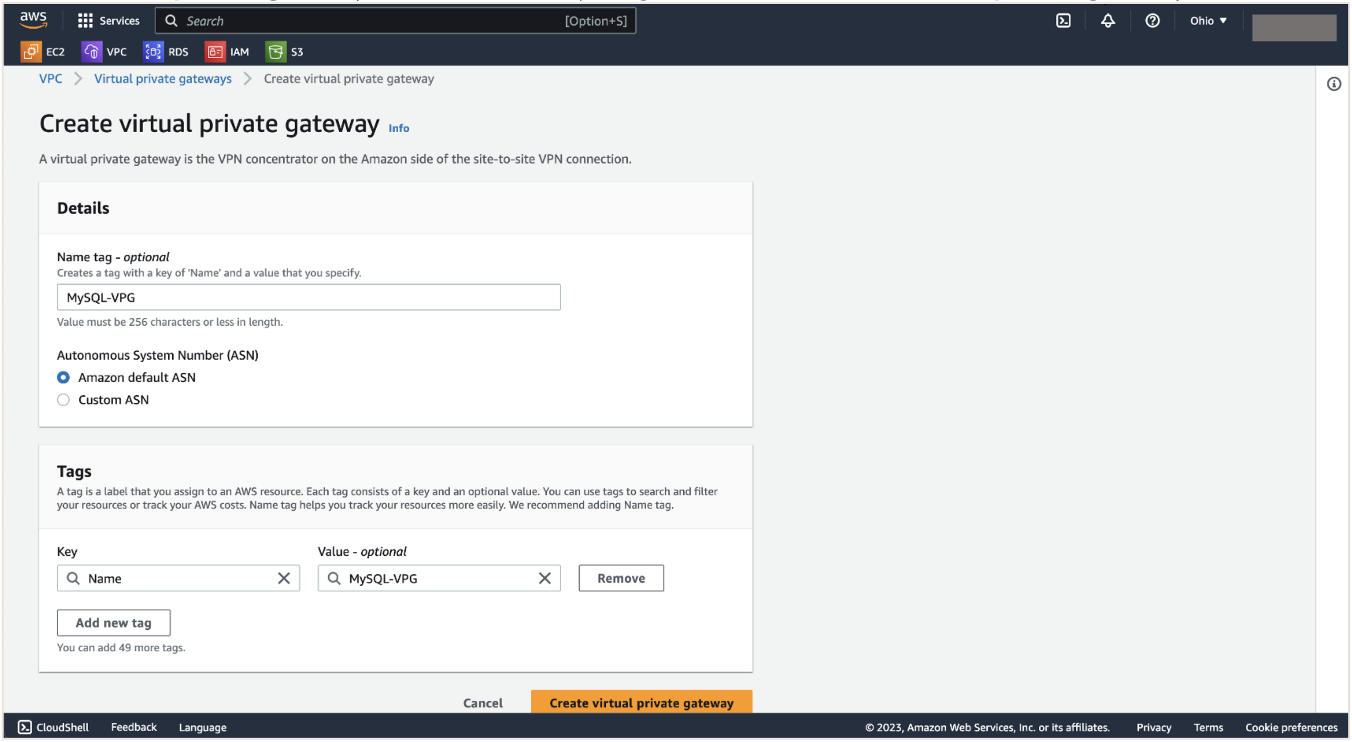
32. Enter a **temporary customer gateway name**. For **BGP ASN** input **31898** and for **IP address** enter **1.1.1.1**. Leave the rest as-is and click **Create Customer Gateway**.



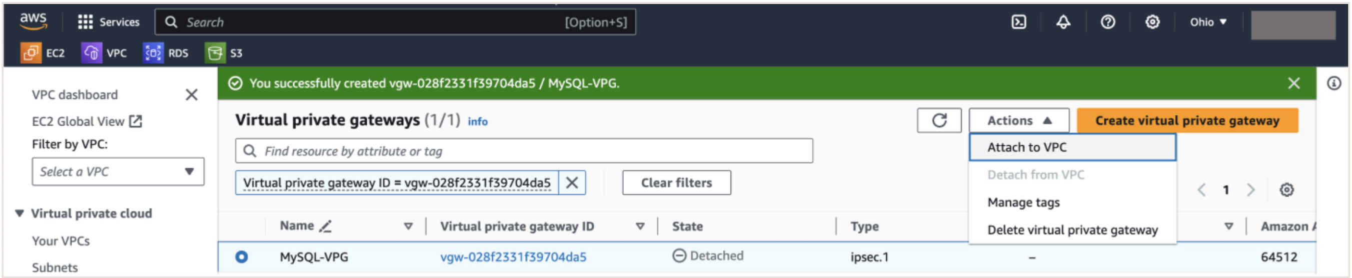
33. From the Customer gateways page, scroll down on the left-hand AWS menu. Under Virtual private network click **Virtual private gateways**. Click **Create virtual private gateway**.



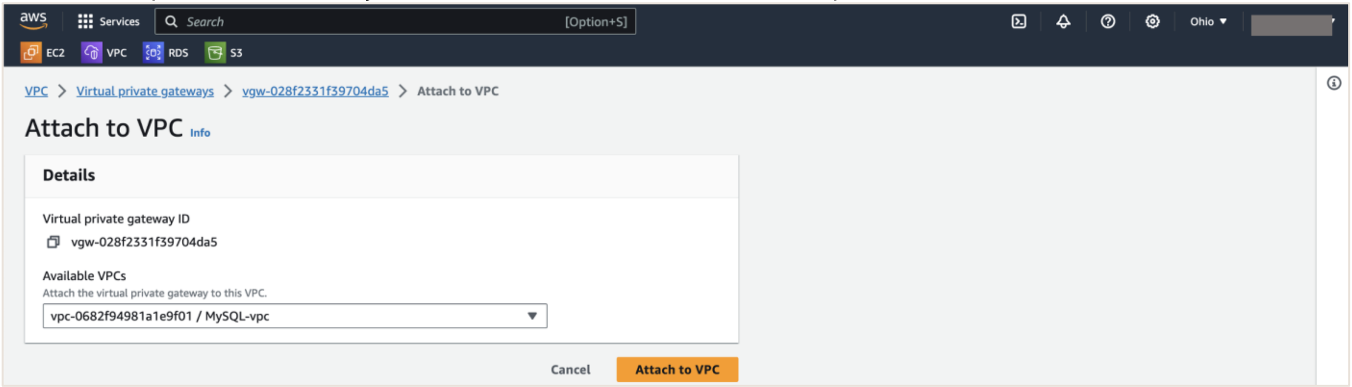
34. Enter a **virtual private gateway name**. Leave everything as-is and click **Create virtual private gateway**.



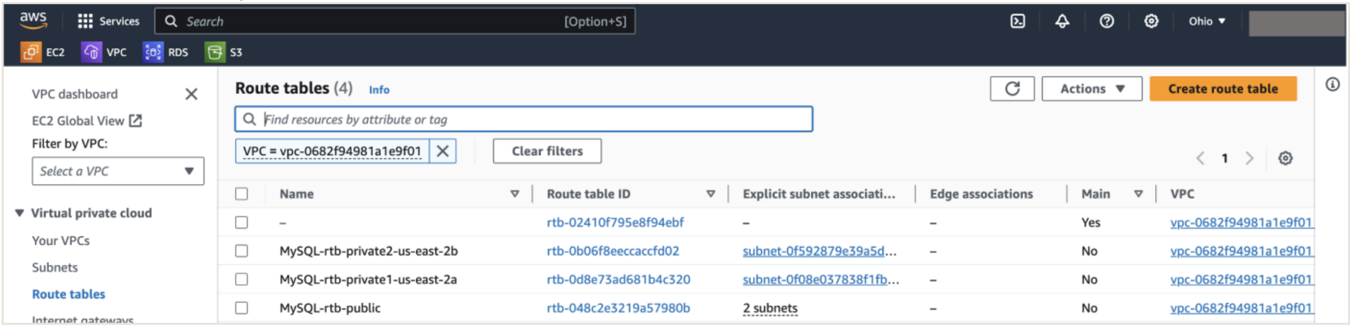
35. While still on the Virtual Private Gateway page, select the **virtual private gateway** that we just created. Click on the **Actions** menu and select **Attach to VPC**.



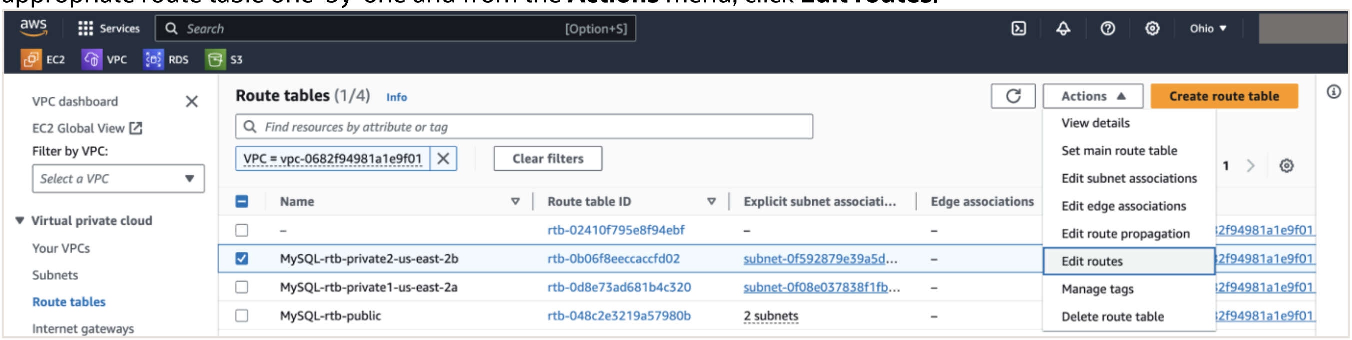
36. From the drop-down list, select **your VPC**. Click **Attach to VPC** once completed.



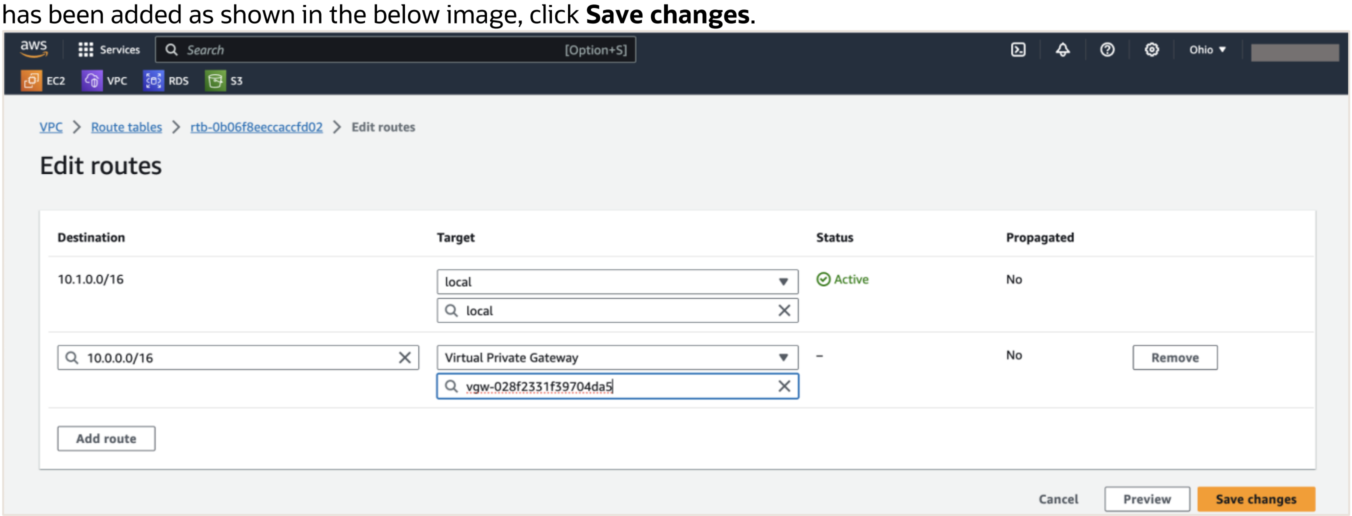
37. Wait until your Virtual private gateway changes its state to **Attached**. It is now time to update the AWS route tables - similar to what we did on OCI. From the Virtual private gateways page, scroll up on the left-hand AWS menu. Under Virtual private cloud, select **Route tables**.



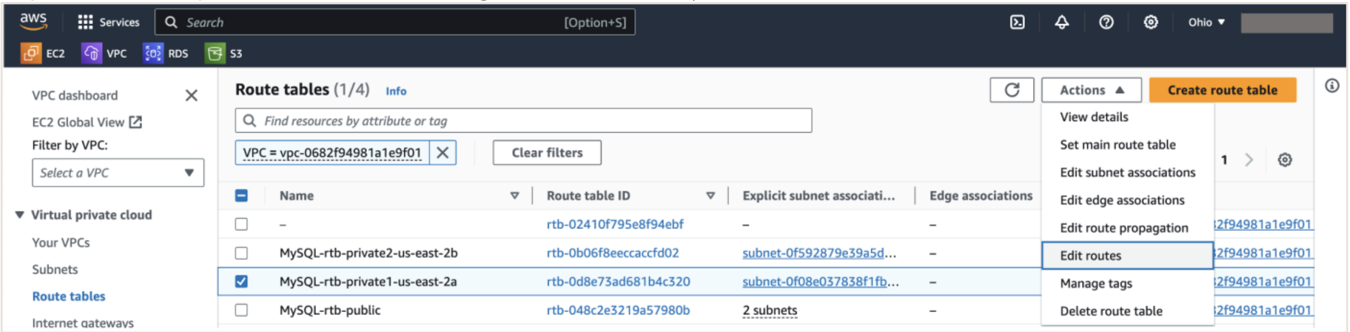
38. For this guide, the main route table (rtb-02410f795e8f94ebf - the one with no name) is not being used, although we will use the public route table (to deploy an EC2 later) and both private route tables (for Aurora). For each of the route tables that you wish to use, you will need to add an additional route rule. Select the appropriate route table one-by-one and from the **Actions** menu, click **Edit routes**.

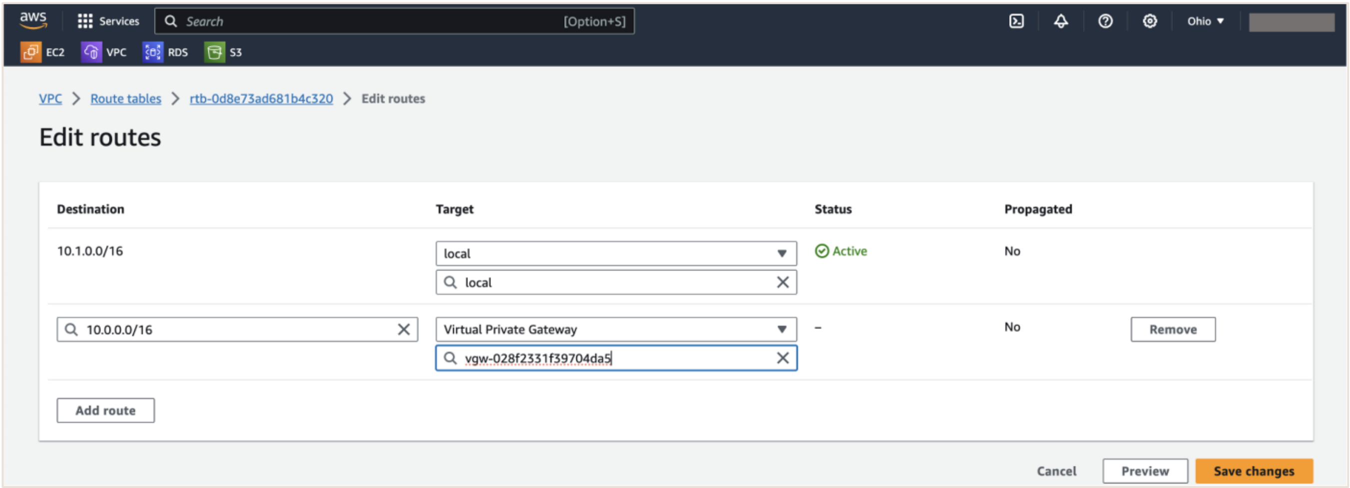


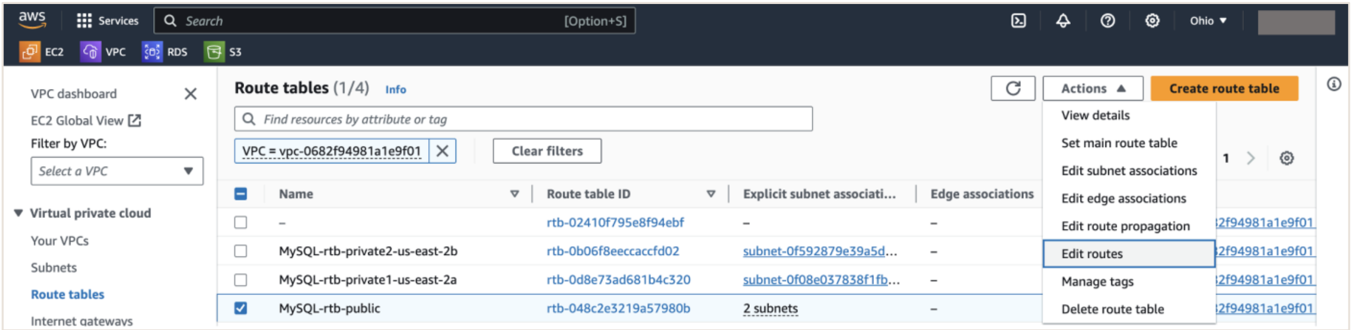
39. Click **Add route** and under the **Destination**, input your **OCI VCN CIDR block** that you are using when you created your OCI VCN (the guide uses OCI VCN CIDR block of **10.0.0.0/16**). Afterwards, for **Target**, click **Virtual Private Gateway** from the drop-down list and **select your Virtual Private Gateway**. Once your route has been added as shown in the below image, click **Save changes**.

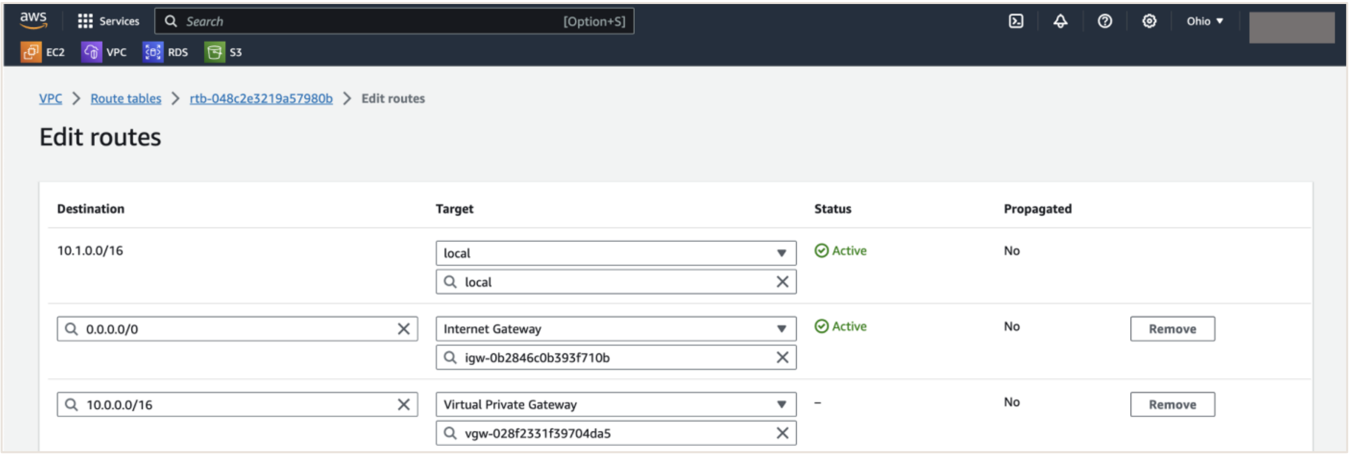


40. Repeat the same process for the remaining route tables that you will use.

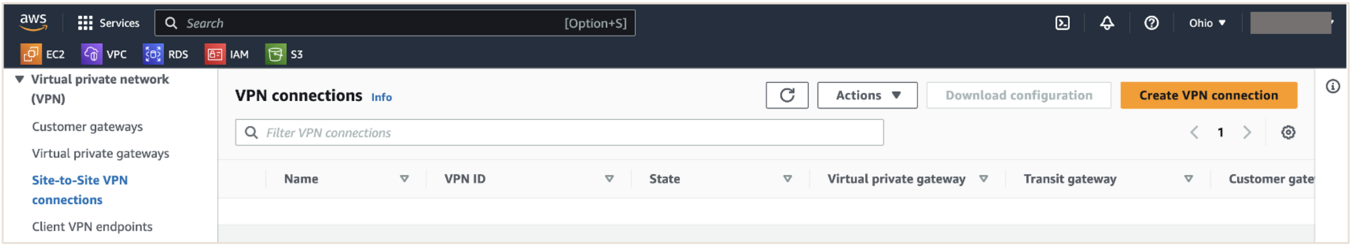




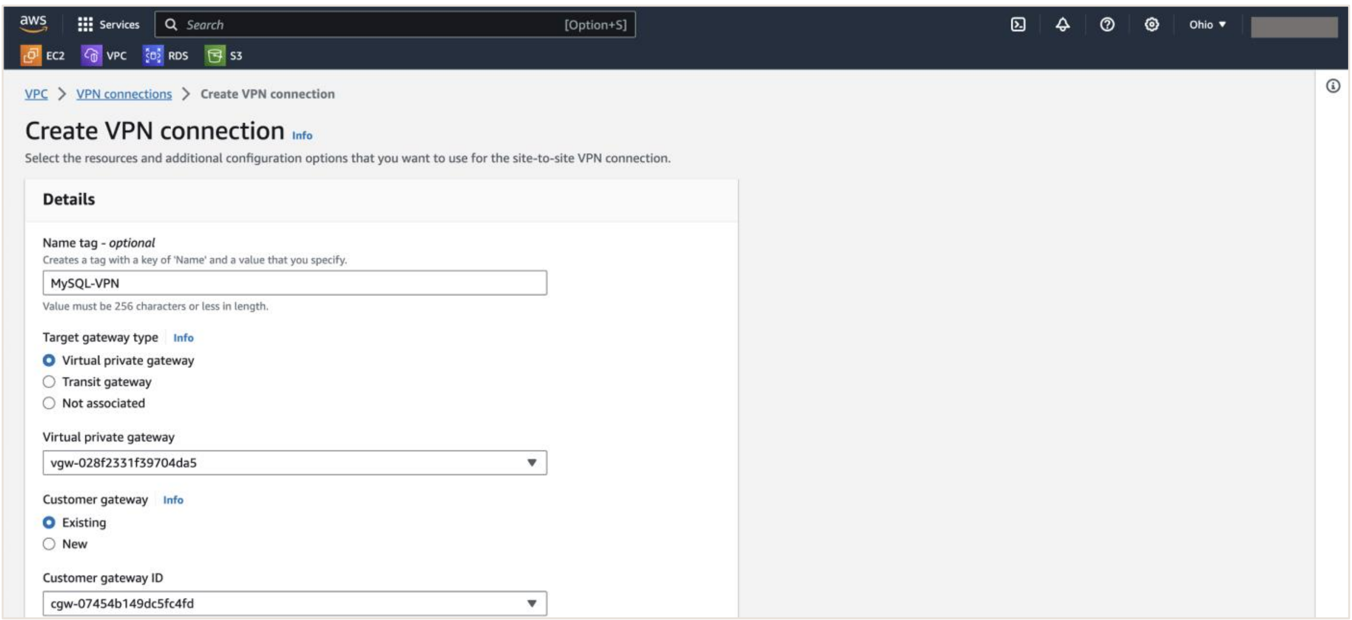


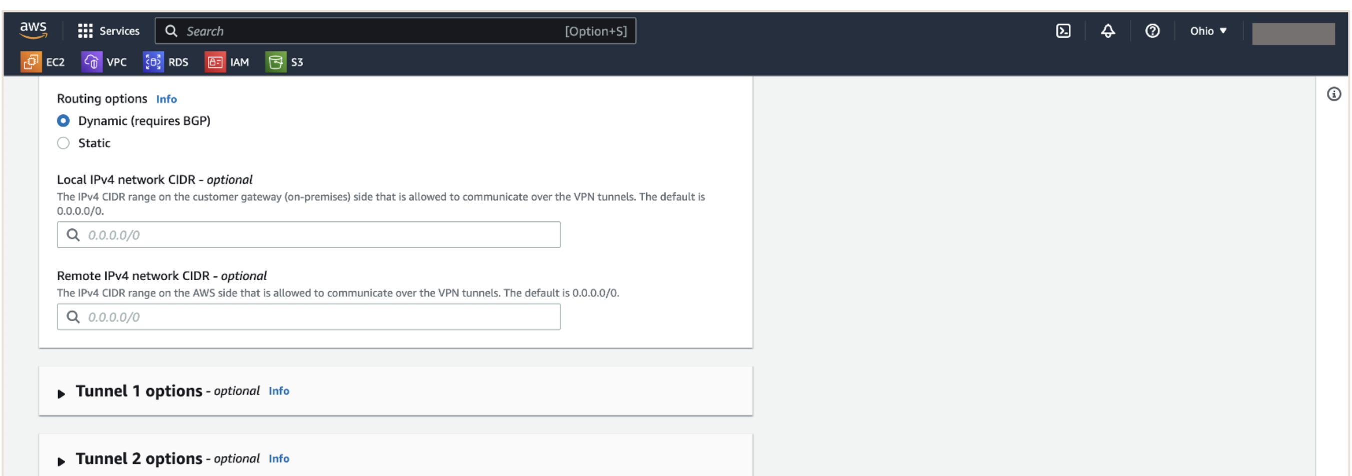


41. After you have updated all your route tables on AWS, from the left-hand menu, scroll down and click **Site-to-Site VPN Connections** under Virtual Private Network (VPN). Once on the appropriate page, click **Create VPN Connection**.



42. Give a **VPN connection name**, for **Target gateway type** select **Virtual private gateway**. Under **Virtual private gateway** drop-down - select **the VPG that we had created earlier**. For **Customer gateway** select **Existing** and under the **Customer gateway ID** drop-down - select **the temporary Customer Gateway that we had created earlier**. Under **Routing options** select **Dynamic (requires BGP)**. Leave the **Local and Remote IPv4 network CIDR** fields blank.

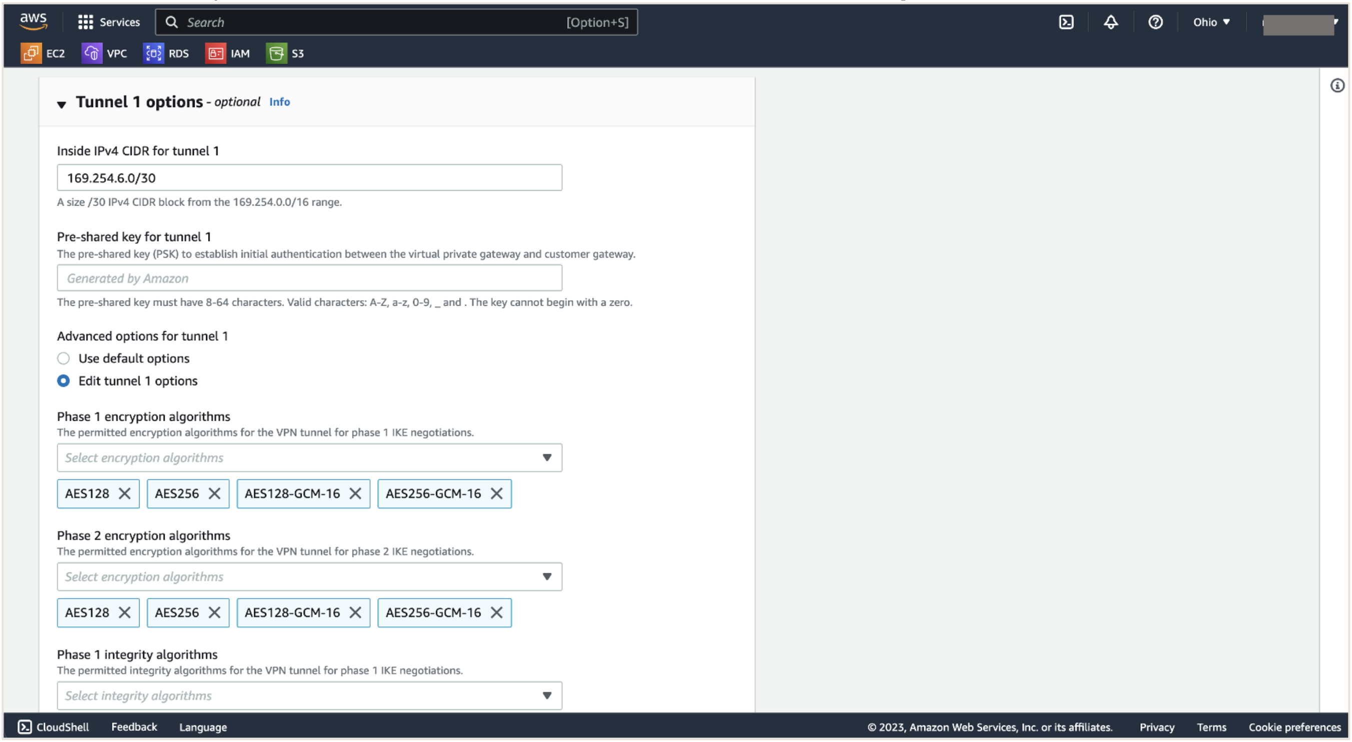




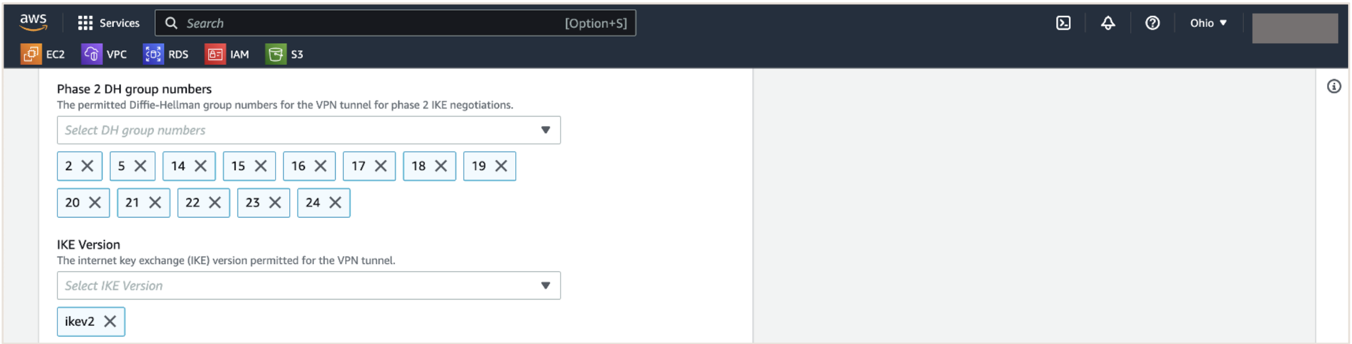
43. While still on the Create VPN Connection page, expand the **Tunnel 1 options**. **Choose a /30 CIDR from within the link local 169.254.0.0/16 range**. Input the full CIDR in the **Inside IPv4 CIDR for Tunnel 1** field. The guide uses the CIDR block of **169.254.6.0/30**. Ensure that OCI supports the chosen /30 address for the inside tunnel IPs. OCI does not allow you to use the following IP ranges for inside tunnel IPs:

    - 169.254.10.0-169.254.19.255
    - 169.254.100.0-169.254.109.255
    - 169.254.192.0-169.254.201.255

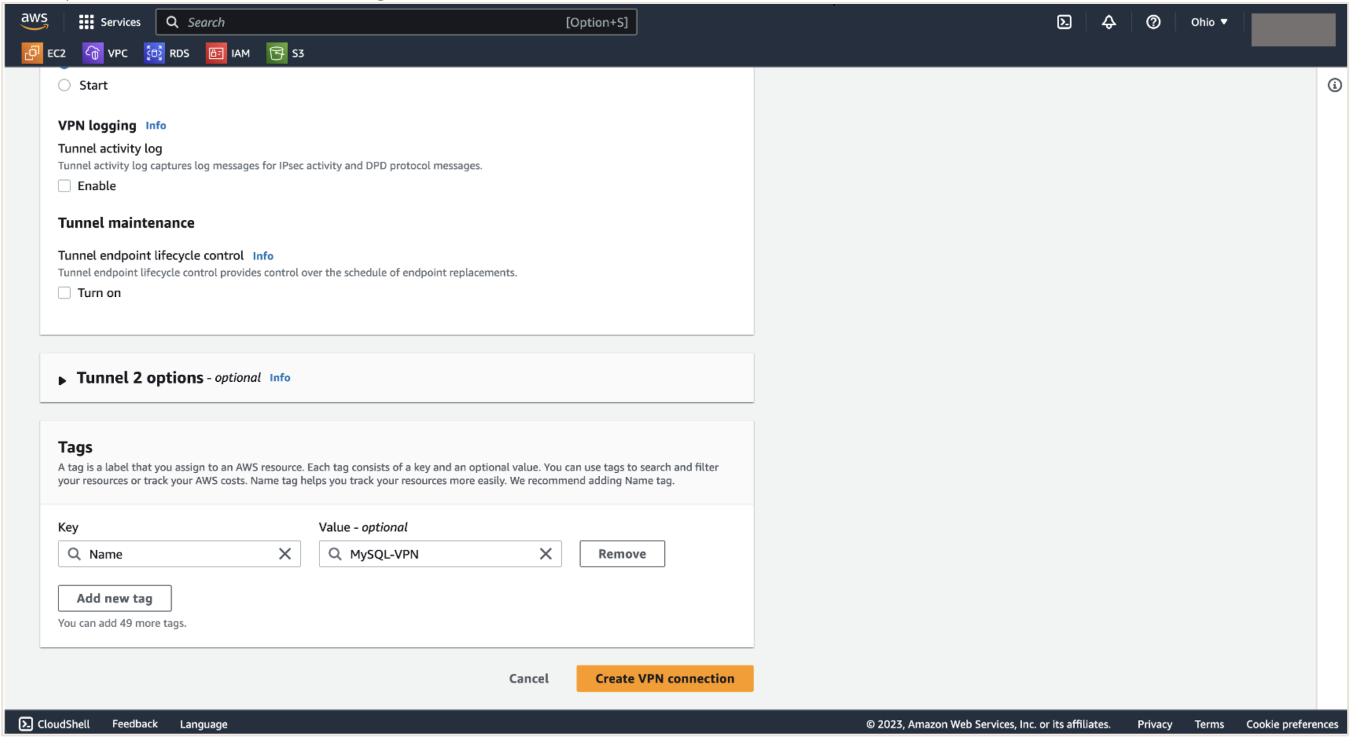
    Under Advanced options for tunnel 1, click the radio button for **Edit tunnel 1 options**.



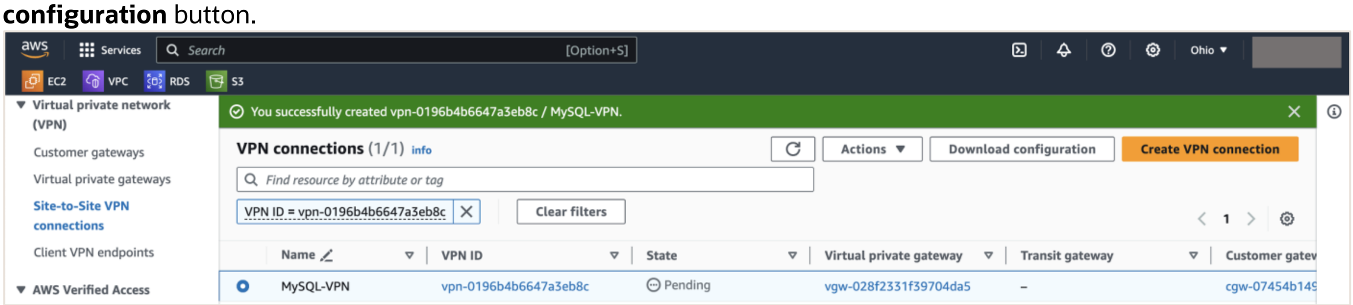
44. Once the tunnel 1 options expand, scroll down and look for **IKE Version**. Click the **X** and remove the **ikev1** field.



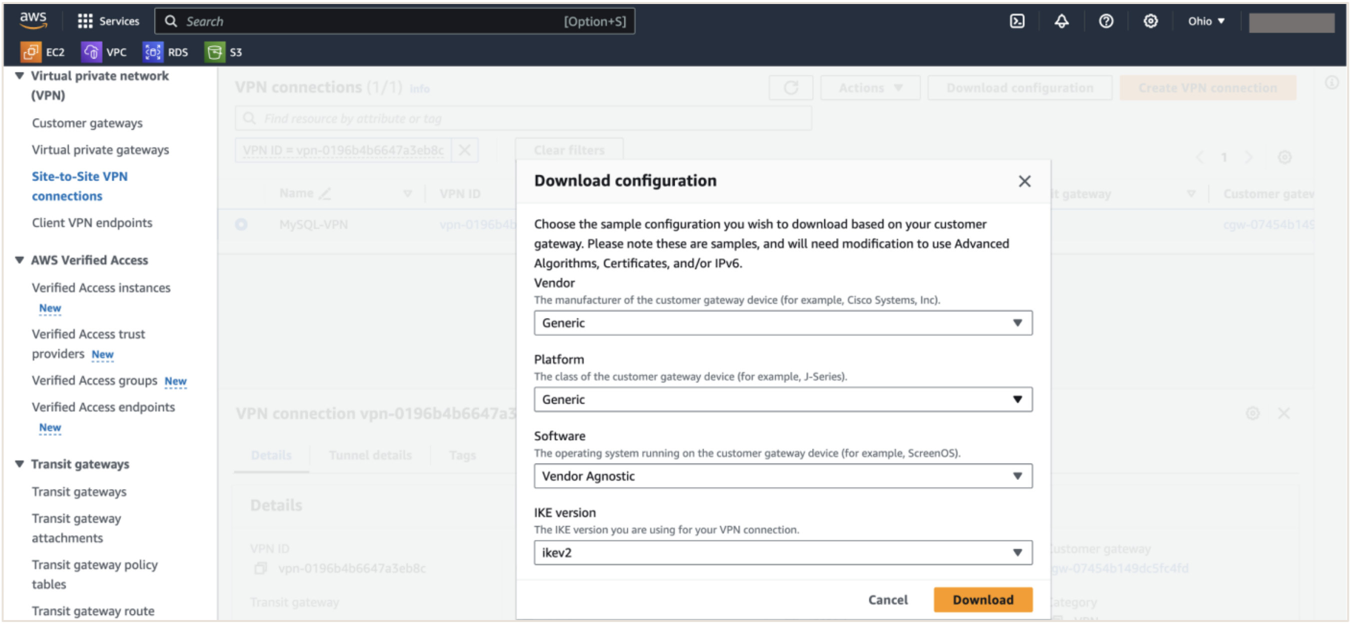
45. After you have finished the configuration, click **Create VPN connection**.



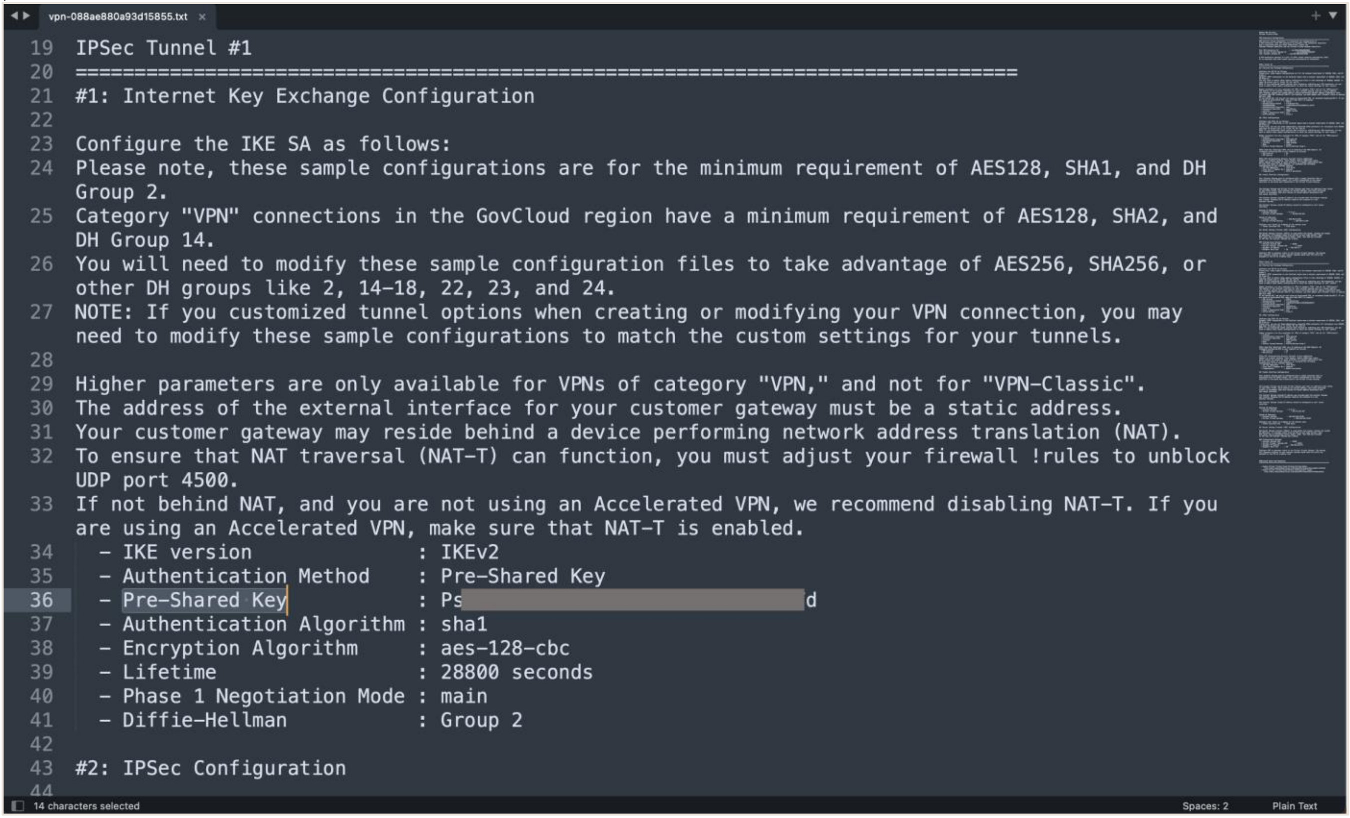
46. On the VPN Connections page, make sure that your VPN connection is selected and click the **Download configuration** button.



47. For **Vendor** and **Platform**, select **Generic**. For **IKE version**, select **ikev2**. Click **Download** afterwards.



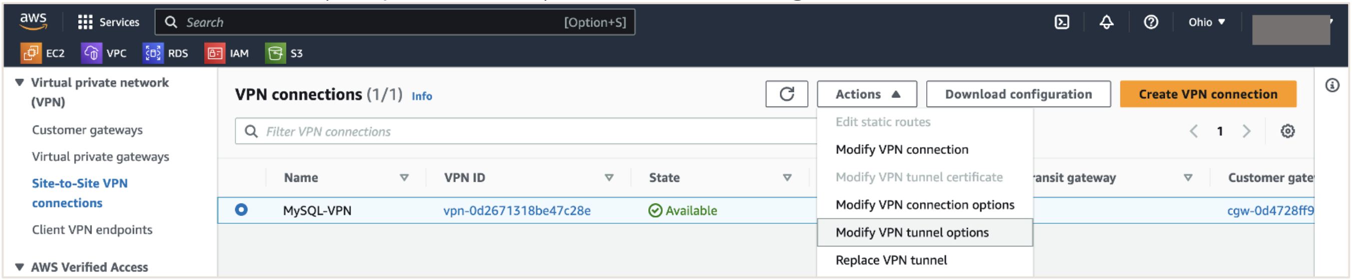
48. Open the downloaded configuration file in your text editor of choice. Look under **IPSec Tunnel #1**, section **#1 Internet Key Exchange Configuration**. Here you find your automatically generated **Pre-Shared Key** for your tunnel. Save this value.

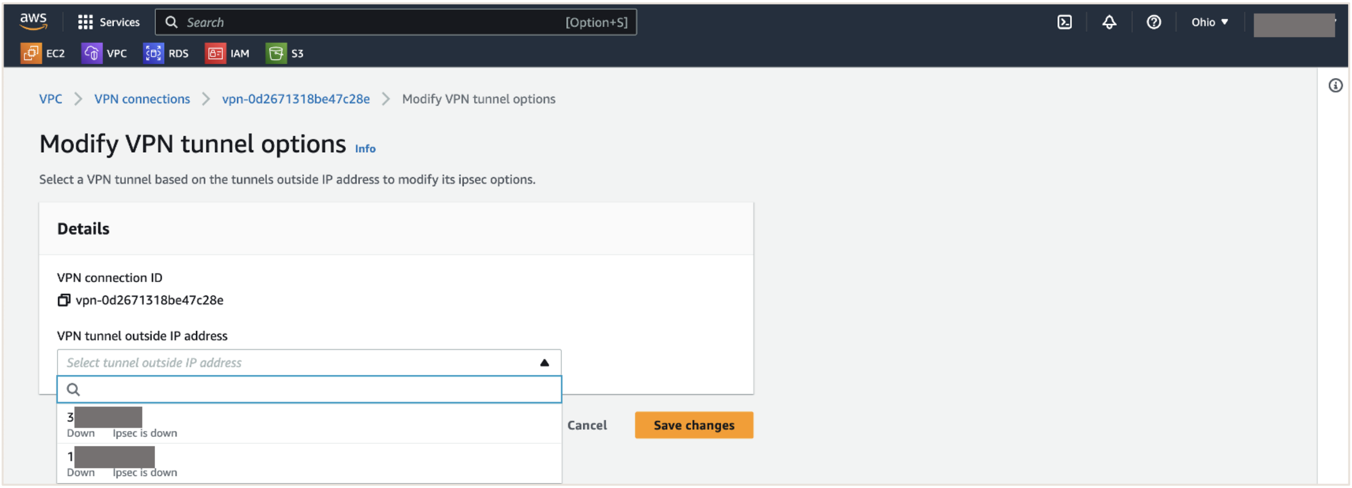


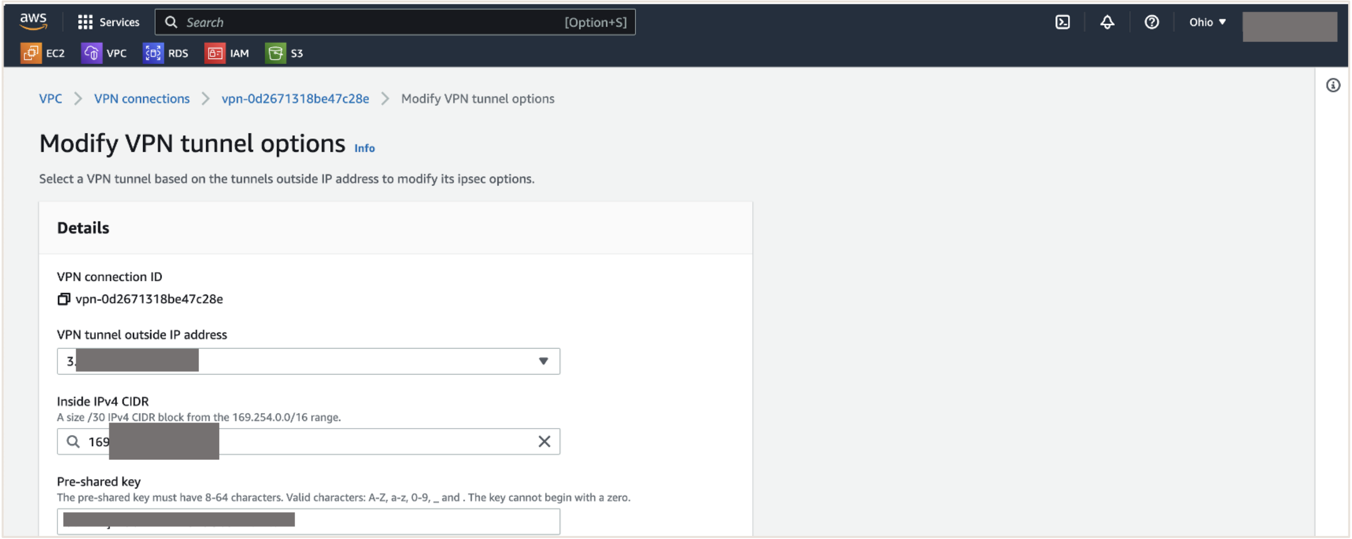
```
IPSec Tunnel #1
================================================================================
#1: Internet Key Exchange Configuration

Configure the IKE SA as follows:
Please note, these sample configurations are for the minimum requirement of AES128, SHA1, and DH Group 2.
Category "VPN" connections in the GovCloud region have a minimum requirement of AES128, SHA2, and DH Group 14.
You will need to modify these sample configuration files to take advantage of AES256, SHA256, or other DH groups like 2, 14-18, 22, 23, and 24.
NOTE: If you customized tunnel options when creating or modifying your VPN connection, you may need to modify these sample configurations to match the custom settings for your tunnels.

Higher parameters are only available for VPNs of category "VPN," and not for "VPN-Classic".
The address of the external interface for your customer gateway must be a static address.
Your customer gateway may reside behind a device performing network address translation (NAT).
To ensure that NAT traversal (NAT-T) can function, you must adjust your firewall !rules to unblock UDP port 4500.
If not behind NAT, and you are not using an Accelerated VPN, we recommend disabling NAT-T. If you are using an Accelerated VPN, make sure that NAT-T is enabled.
  - IKE version              : IKEv2
  - Authentication Method    : Pre-Shared Key
  - Pre-Shared Key           : Ps[redacted]d
  - Authentication Algorithm : sha1
  - Encryption Algorithm     : aes-128-cbc
  - Lifetime                 : 28800 seconds
  - Phase 1 Negotiation Mode : main
  - Diffie-Hellman           : Group 2

#2: IPSec Configuration
```

Note: AWS might generate a pre-shared key using the period or underscore characters (. or _). OCI does not support using those characters in a pre-shared key. A key that includes these values must be changed. To change your pre-shared key in AWS for a tunnel, select your VPN connection, click the **Actions** button, then

Modify VPN Tunnel Options. Select the **IPSec Tunnel #1 Virtual Private Gateway outisde IP address** from the drop-down (you can find this in the AWS downloaded configuration file). Remove the period or underscore characters from your pre-shared key and click **Save changes**.





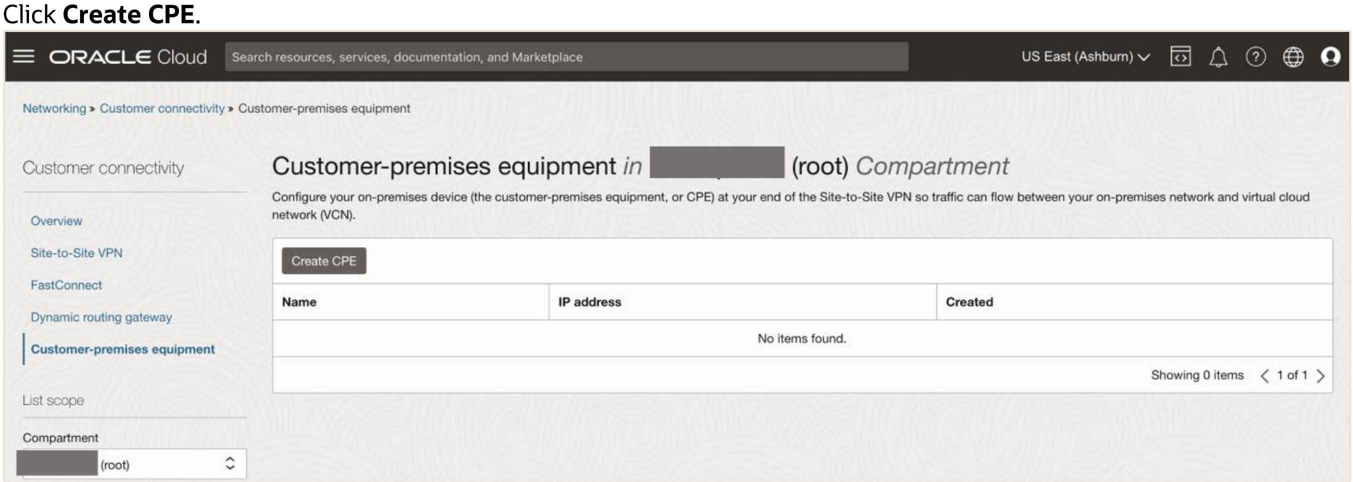


49. While still under Tunnel 1 in the downloaded configuration, scroll down to section **#3 Tunnel Interface Configuration**. Here, note down all the values for **Outside IP Addresses** and **Inside IP Addresses**.

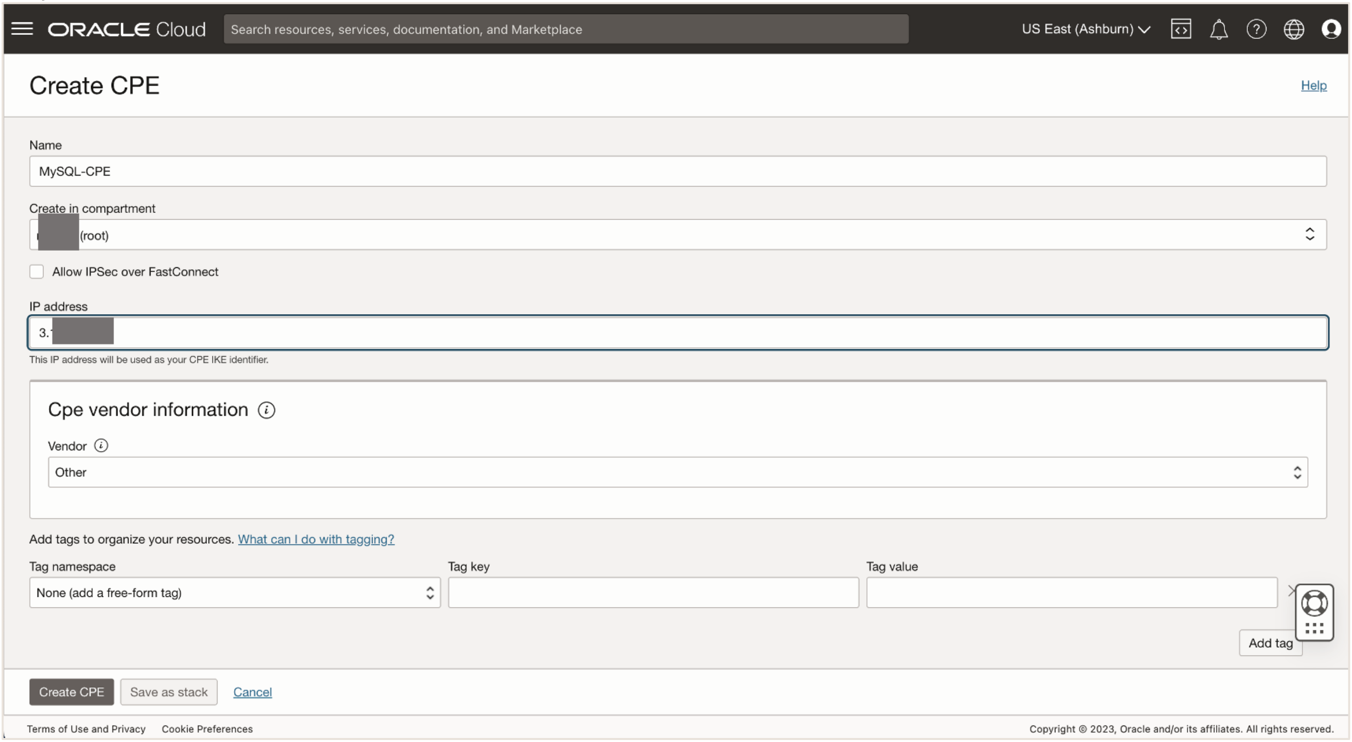
    Scroll down to section **#4: Border Gateway Protocol (BGP) Configuration** and note down the **Virtual Private Gateway ASN** value.

```
The Customer Gateway outside IP address was provided when the Customer Gateway
was created. Changing the IP address requires the creation of a new
Customer Gateway.

The Customer Gateway inside IP address should be configured on your tunnel
interface.

Outside IP Addresses:
  - Customer Gateway              : 1.1.1.1
  - Virtual Private Gateway         : 3.[redacted]

Inside IP Addresses
  - Customer Gateway              : 169.[redacted]
  - Virtual Private Gateway             : 169.[redacted]

Configure your tunnel to fragment at the optimal size:
  - Tunnel interface MTU     : 1436 bytes

#4: Border Gateway Protocol (BGP) Configuration:

The Border Gateway Protocol (BGPv4) is used within the tunnel, between the inside
IP addresses, to exchange routes from the VPC to your home network. Each
BGP router has an Autonomous System Number (ASN). Your ASN was provided
to AWS when the Customer Gateway was created.

BGP Configuration Options:
  - Customer Gateway ASN            : 31898
  - Virtual Private  Gateway ASN          : 64512
  - Neighbor IP Address           : 16[redacted].
  - Neighbor Hold Time       : 30
```

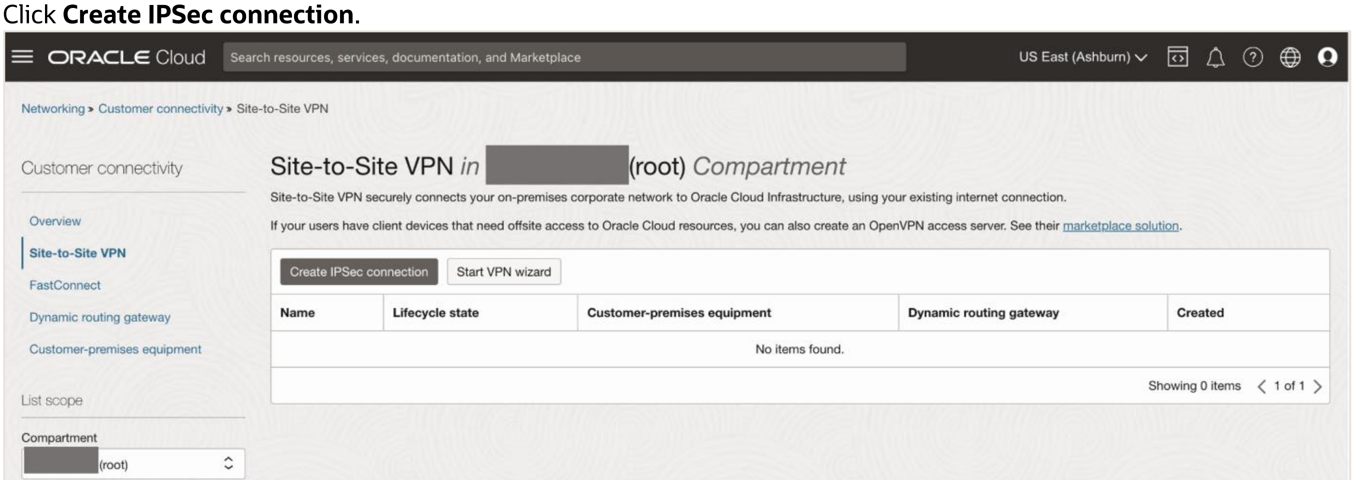
50. Log back in to OCI. From the OCI Navigation menu, navigate to **Networking**, click **Customer connectivity**, and click on **Customer-premises equipment**.
51. Click **Create CPE**.



52. Enter a **CPE name**. For the **Public IP address**, input the **Outside IP Address of the Virtual Private Gateway** - you can find this in the configuration file downloaded from AWS. For **CPE Vendor**, select **Other** from the dropdown. Click **Create CPE**.

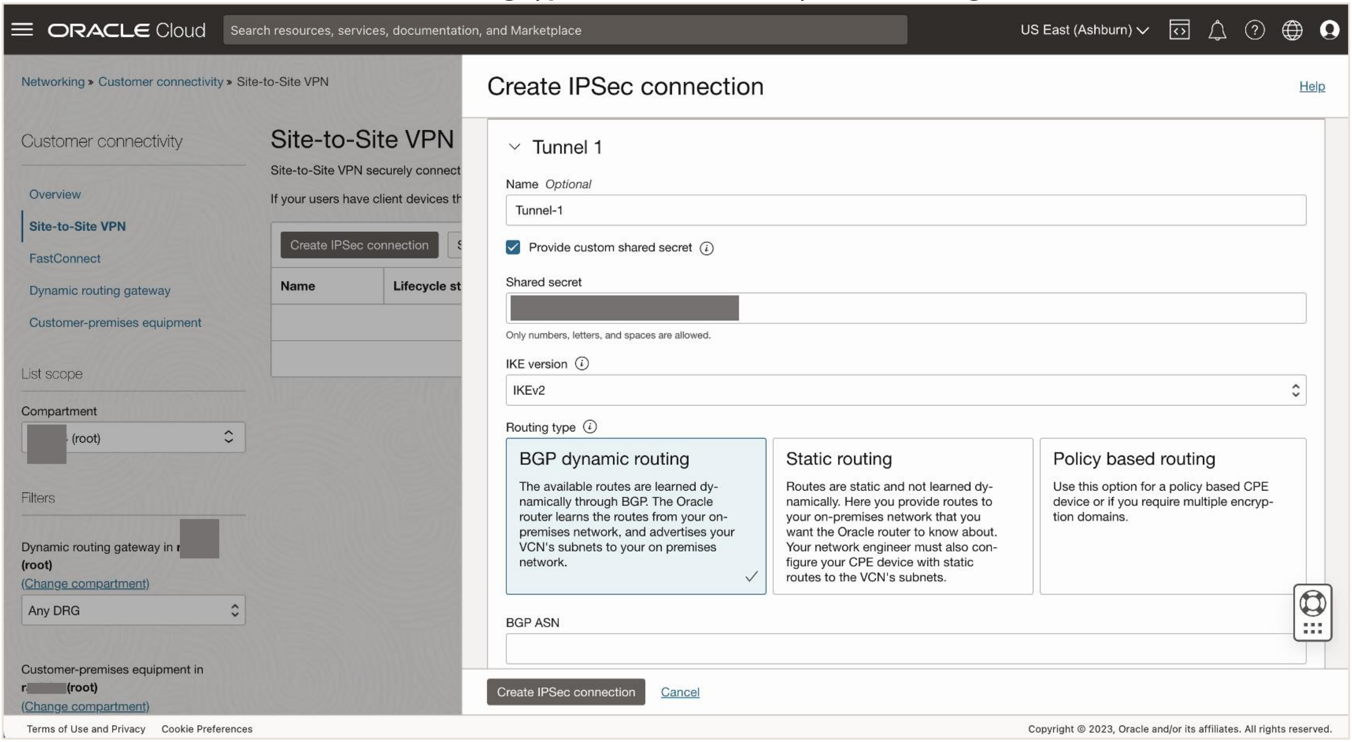


53. From the OCI Navigation menu, navigate to **Networking** and click on **Site-to-Site VPN**.
54. Click **Create IPSec connection**.

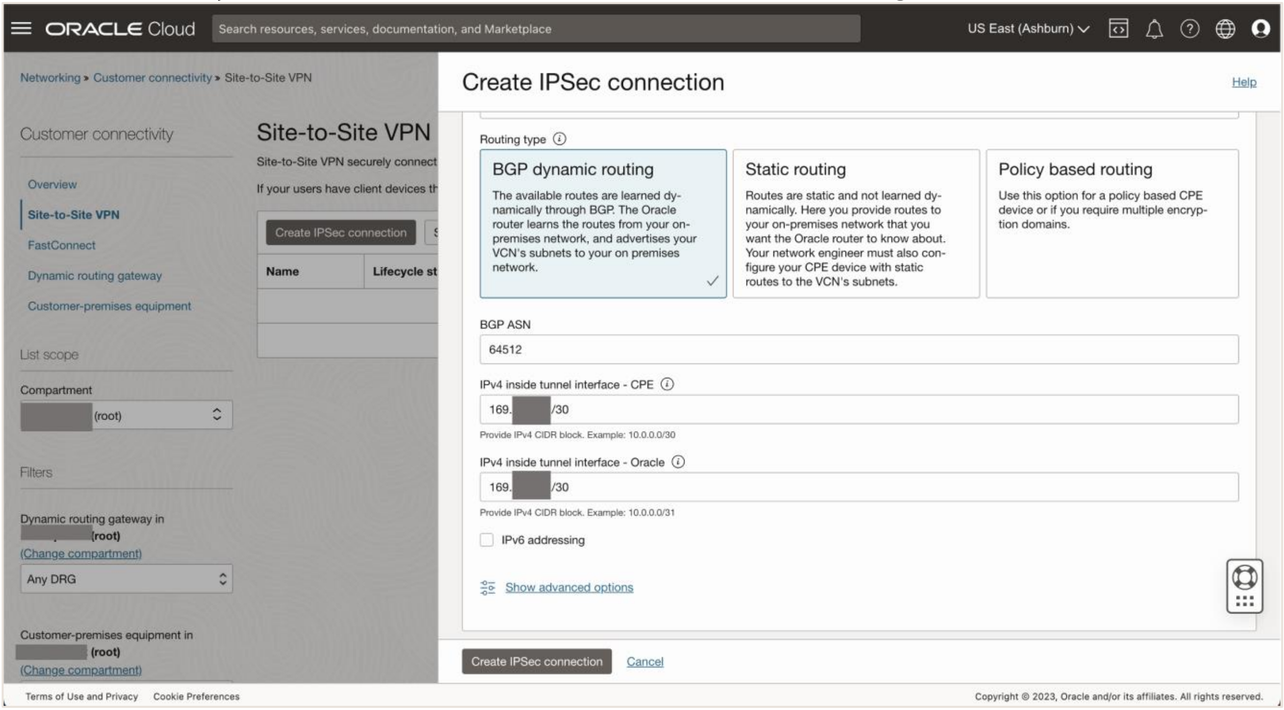


55. Enter a **IPSec connection name**. Under **Customer-premises equipment** dropdown, select the CPE we previously created. For **Dynamic routing gateway compartment** select the DRG we created. For **Routes to your on-premises network**, enter **0.0.0.0/0**.

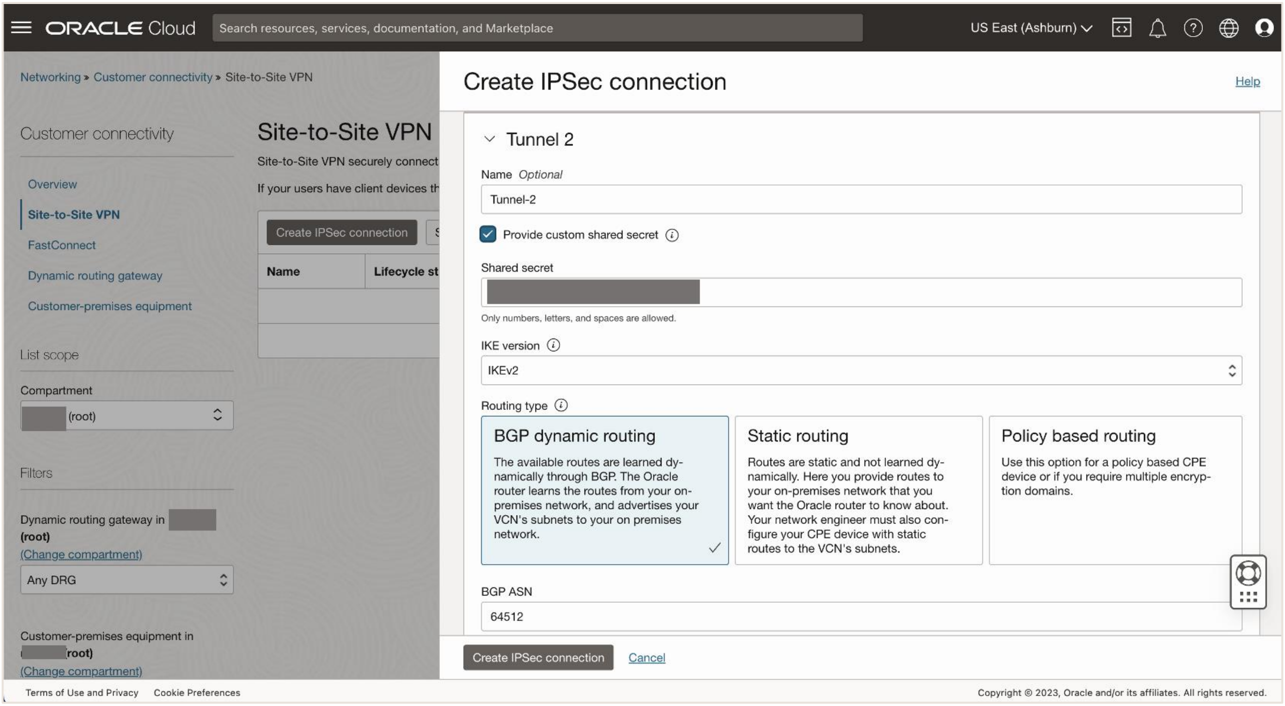
56. While on the Create IPSec connection page, configure your **Tunnel 1**. Enter a **tunnel name**, check the **Provide custom shared secret** box, and input the **Pre-Shared Key** from the AWS VPN configuration file. For **IKE version**, select **IKEv2** and under **Routing type** - make sure **BGP dynamic routing** is selected.



57. Under **BGP ASN**, input the **BGP Virtual Private Gateway ASN** from the AWS VPN configuration file. The default AWS BGP ASN is **64512**. For **IPv4 inside tunnel interface - CPE**, enter the **Inside IP Address of the Virtual Private Gateway**. For **IPv4 inside tunnel interface - Oracle**, enter the **Inside IP Address of the Customer Gateway**. You can find all this information from the AWS VPN configuration file.

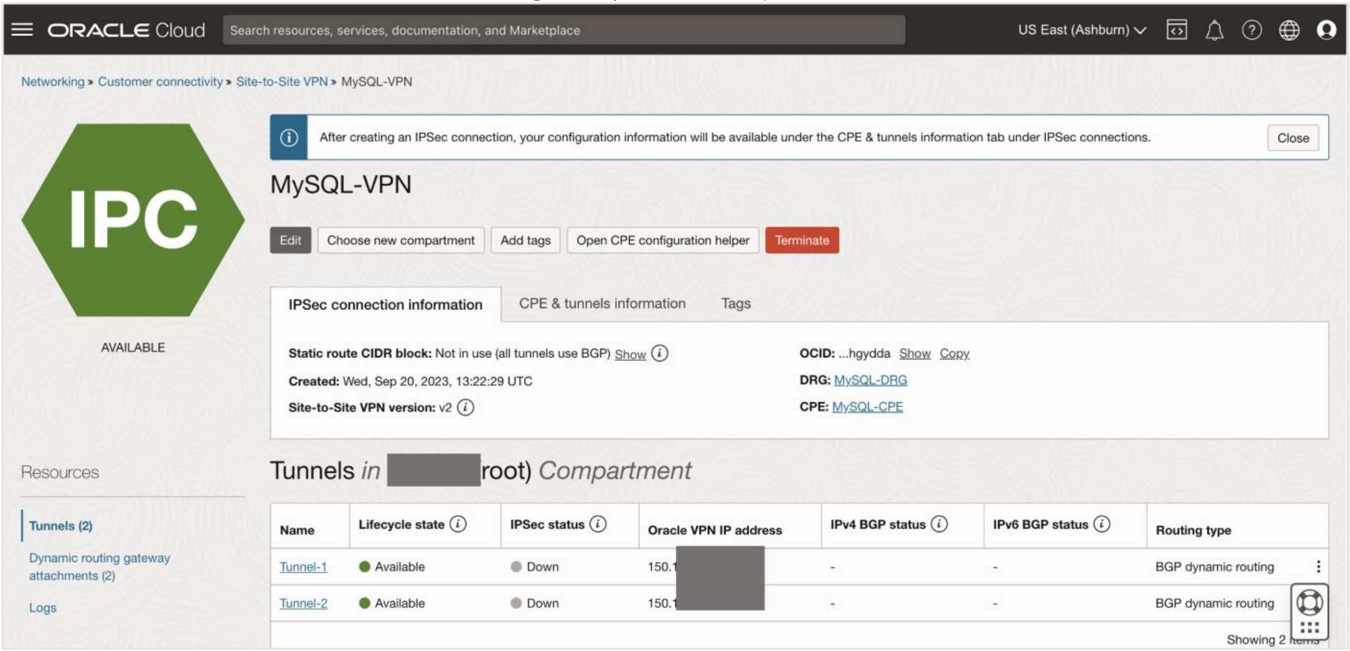


58. Configure your **Tunnel 2** by copying and pasting the same values from Tunnel 1 into Tunnel 2. Click **Create IPSec connection**.

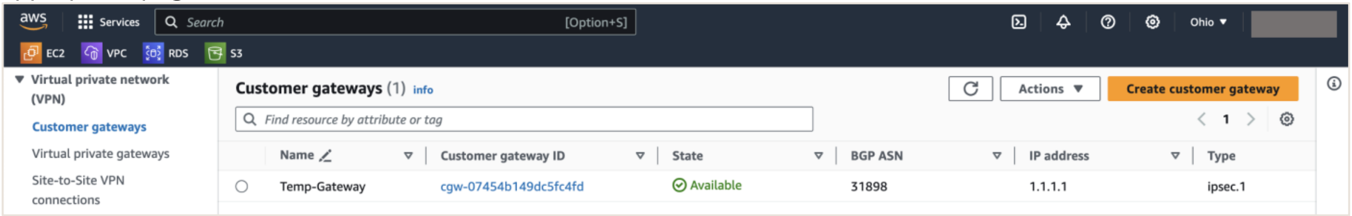


Note: only Tunnel 1 will be used for this VPN connection and migration. We need to configure Tunnel 2 otherwise we cannot click Create IPSec connection.

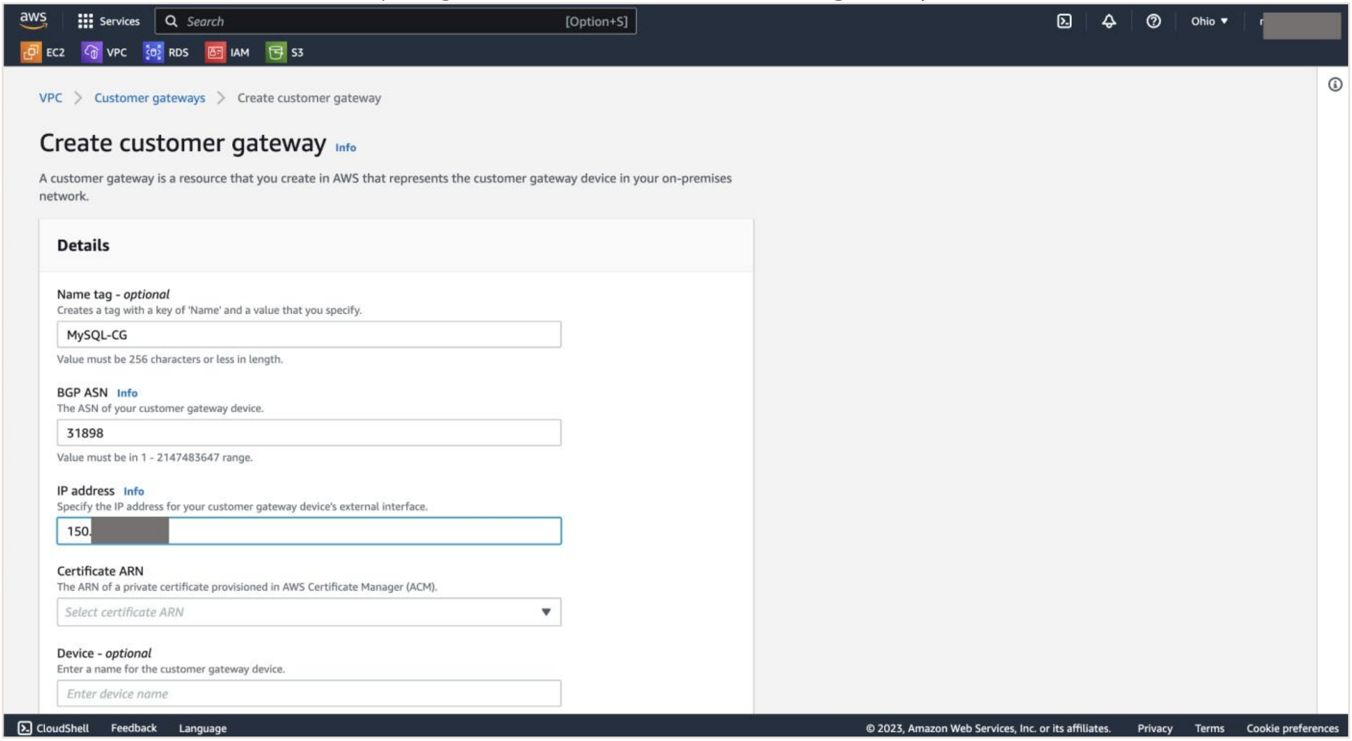
59. After your IPSec connection is provisioned, make note of the **Oracle VPN IP Address** of **Tunnel-1**. This address will be used to create a new customer gateway in the AWS portal.



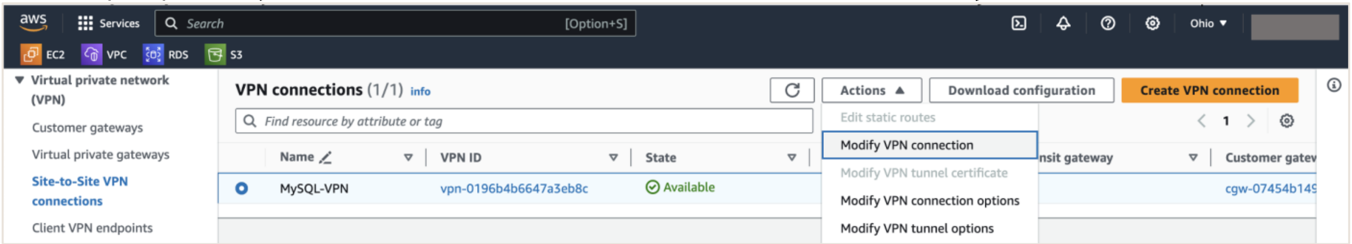
60. Log back in to AWS. Expand the Services menu at the top left of the screen. Navigate to **Networking & Content Delivery** and select **VPC**. From the left-hand menu, scroll down and click **Customer Gateways** under Virtual private network (VPN). Click **Create customer gateway** once you have landed on the appropriate page.



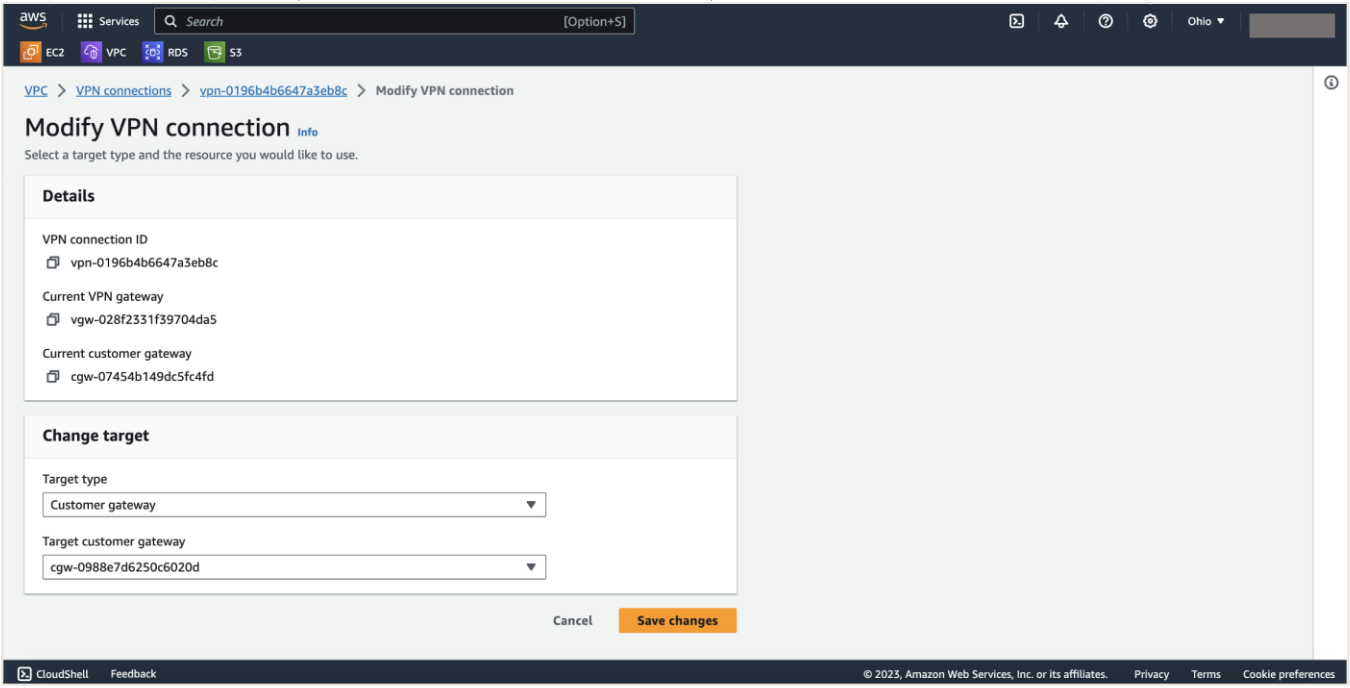
61. Enter a **customer gateway name**. For **BGP ASN**, enter **31898** and for **IP address**, enter the **Oracle VPN IP address for tunnel 1**. Leave everything as-is and click **Create customer gateway**.



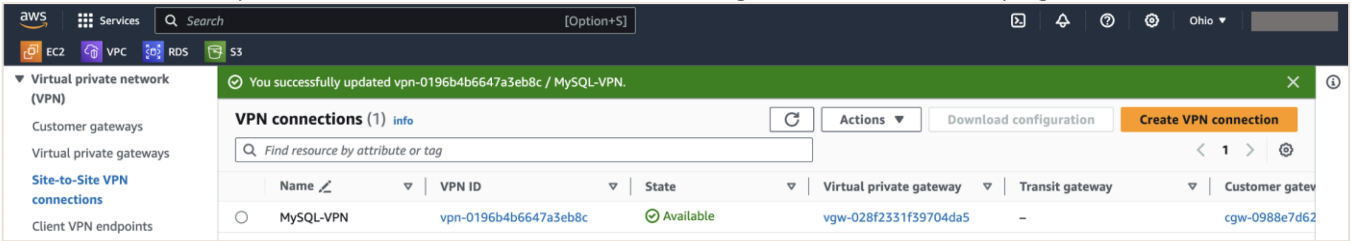
62. From the left-hand AWS menu, scroll down and click **Site-to-Site VPN Connections** under Virtual Private Network (VPN). Select your VPN connection and click the **Actions** button, then **Modify VPN connection**.



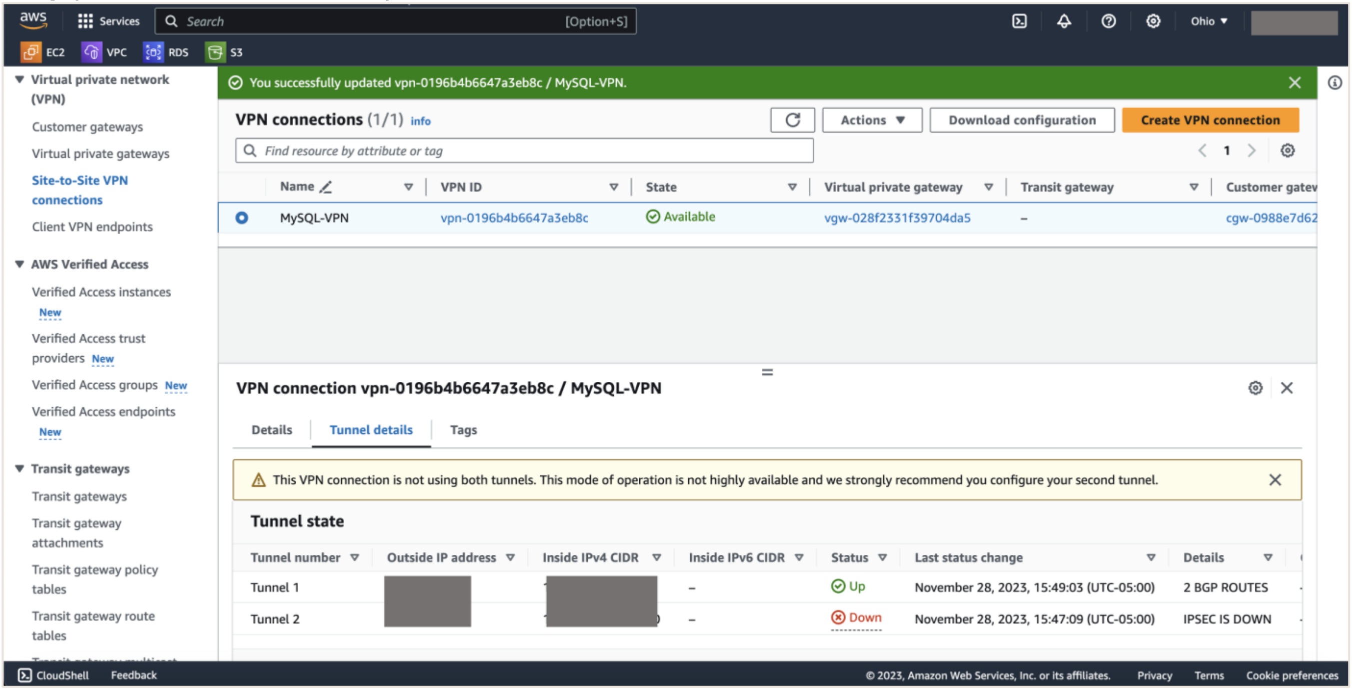
63. You will land on the Modify VPN connection page. Under **Target type**, select **Customer gateway** and for **Target customer gateway**, select the **new Customer Gateway** (not the Temp). Click **Save changes**.



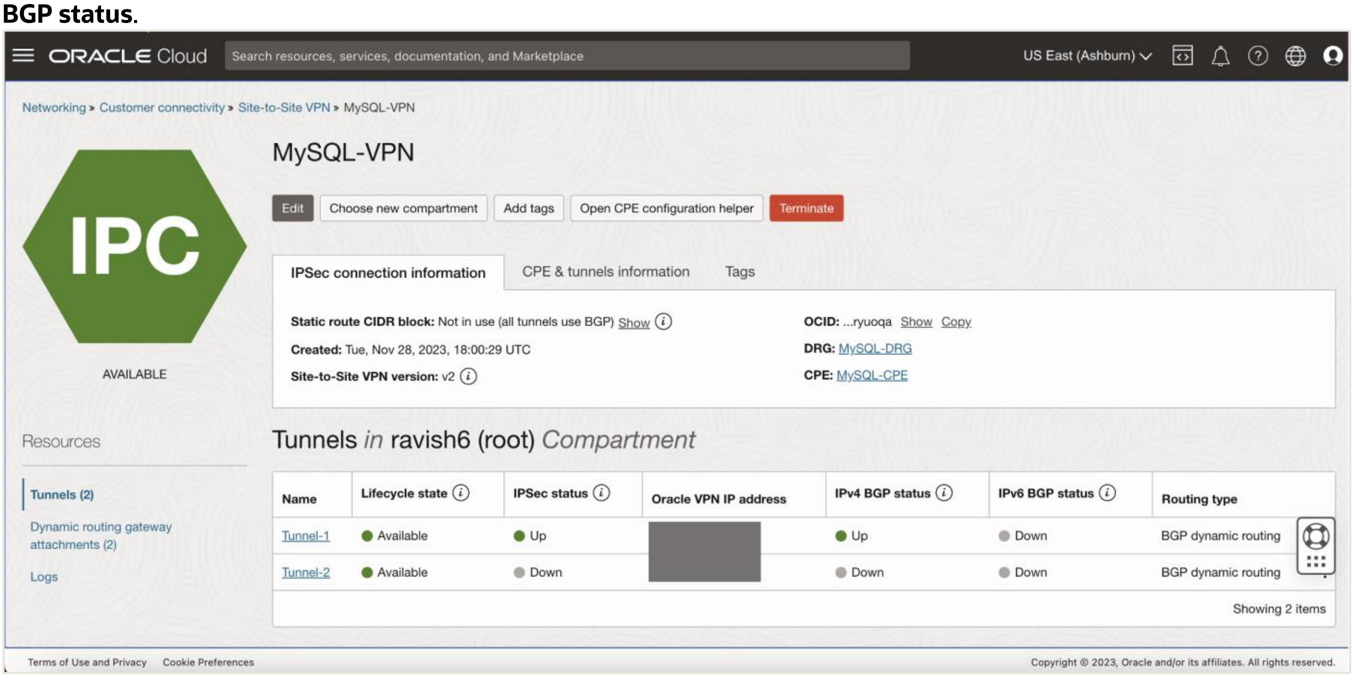
64. After a few minutes, your modified VPN connection should change its **State** from Modifying to **Available**.



65. The VPN connection from OCI to AWS is now setup. To verify if your VPN tunnel is up, select your VPN connection and go to the **Tunnel details** tab which can be found on the same page. You should see a **Status** of **Up** (this will take a few minutes).



66. You can verify the same on the OCI side. Select your Site-to-Site VPN and under the Resources, click **Tunnels** (the page where you got the Oracle VPN IP address). You should see an **Up** status for **IPSec status** and **IPv4 BGP status**.

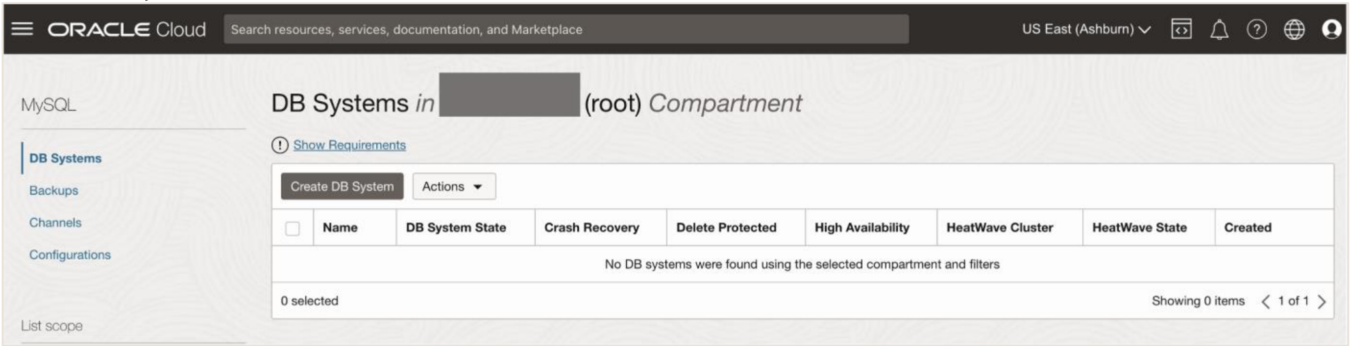


67. We are now ready to perform the migration.

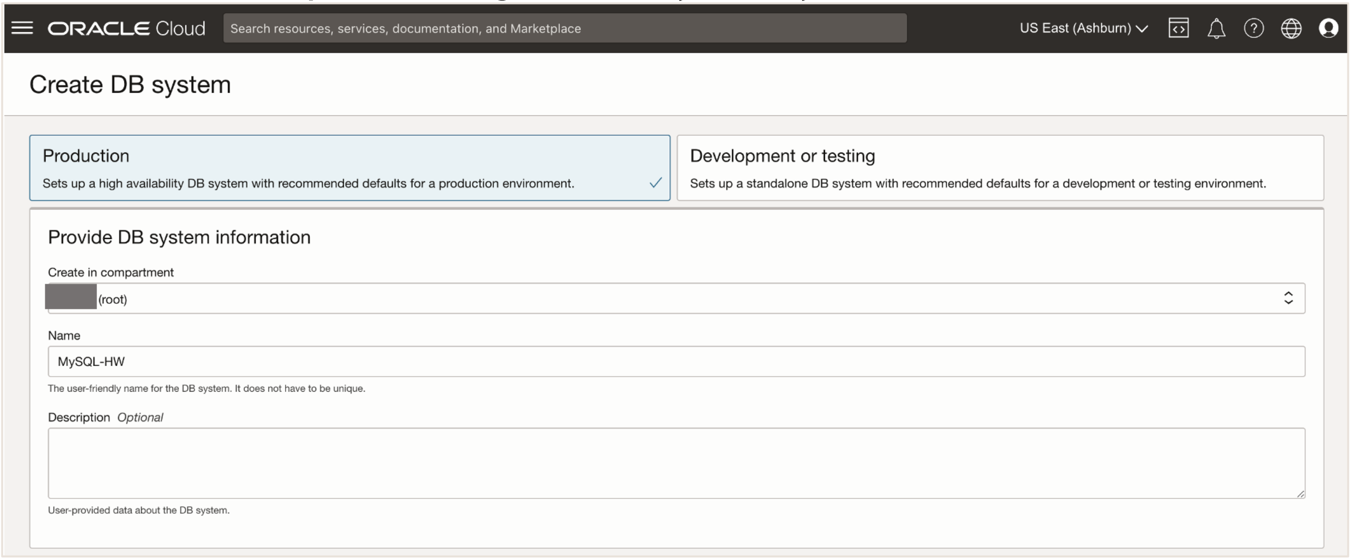
ORACLE

III) On OCI, create a standalone HeatWave MySQL instance.

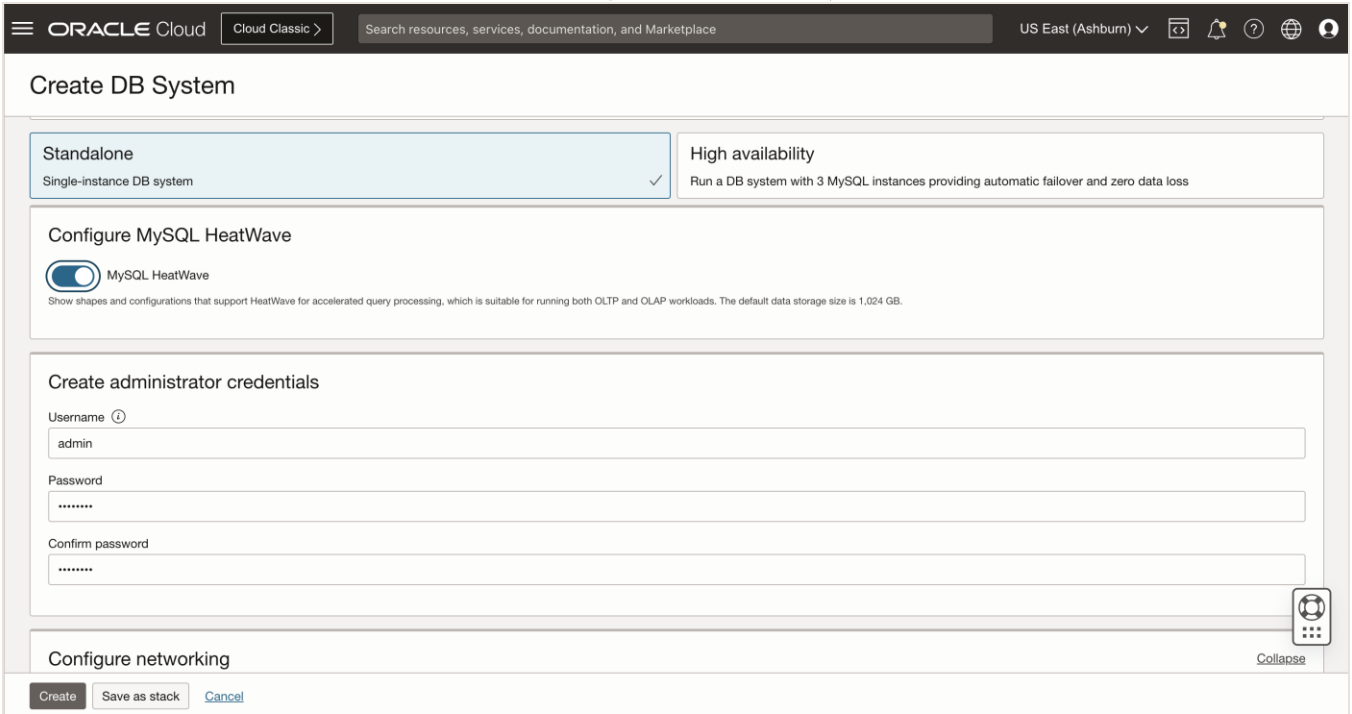
68. From the OCI Console, click on the navigation menu, click **Databases**, and click **HeatWave MySQL**. Click **Create DB System**.



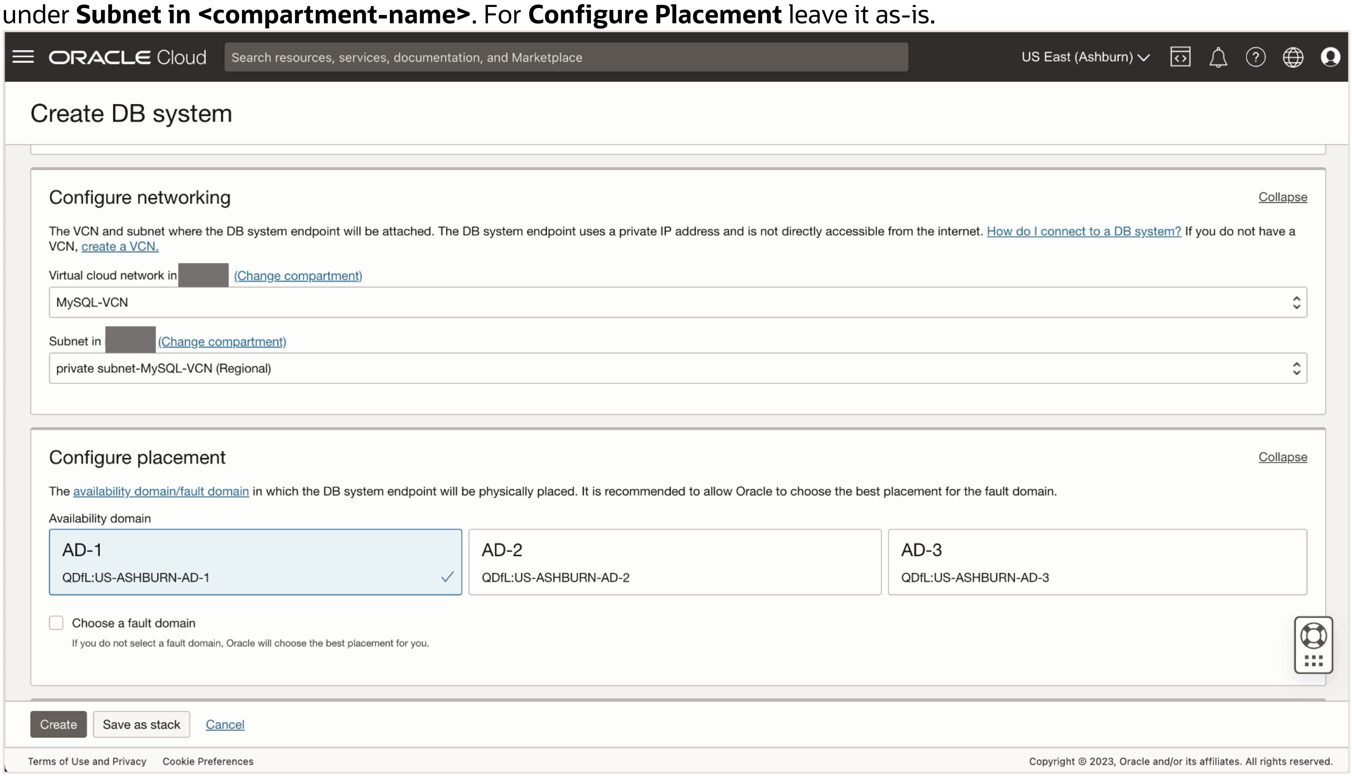
69. Pick **Production** or **Development or testing** and enter **a MySQL DB system name**.



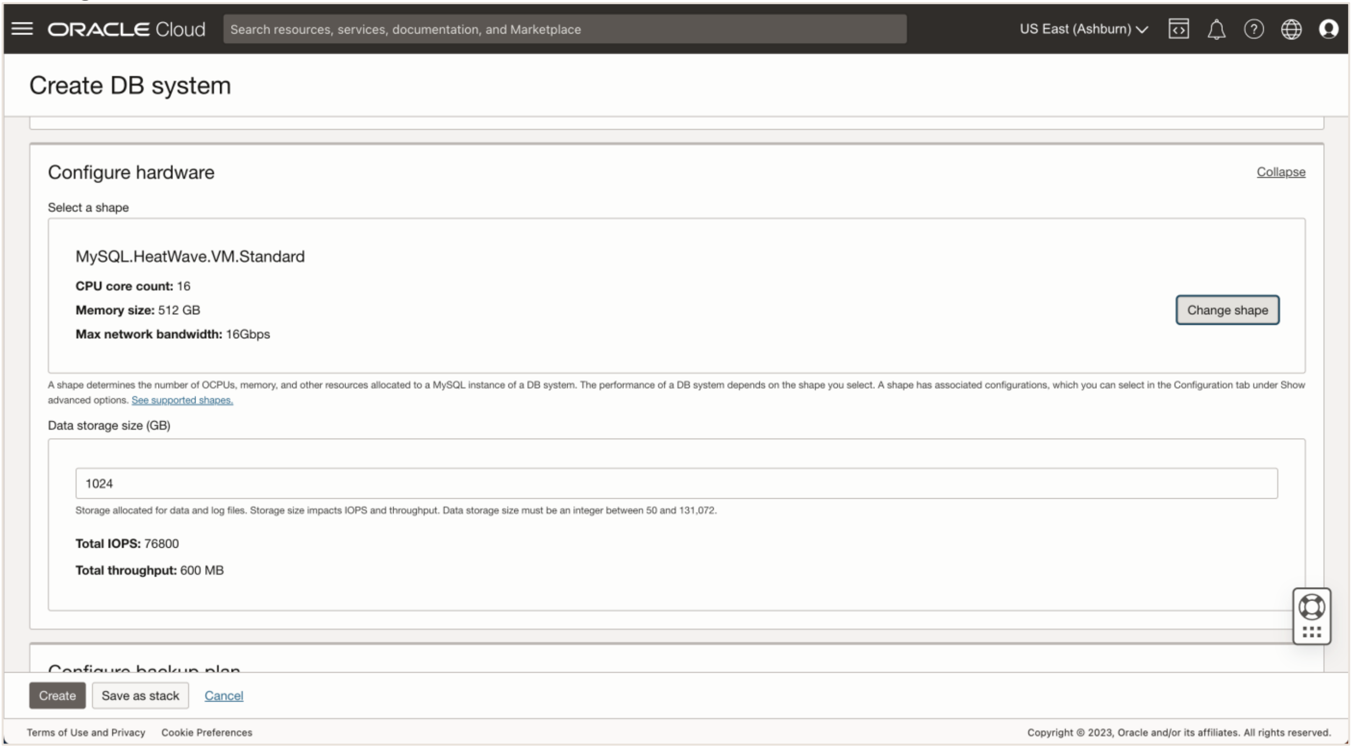
70. Select **Standalone**, do not choose High Availability (HA) here as replicating to a MySQL HA instance on OCI for this migration may create some complications. You may enable HA after you have completed section **VIII)** of this live migration guide. Information on how to enable HA later can be found [here](). Turn **ON** the button for HeatWave MySQL - if you want to run OLTP, OLAP, and ML workloads. Afterwards, create your **Administrator credentials** that will be used to manage the HeatWave MySQL database.



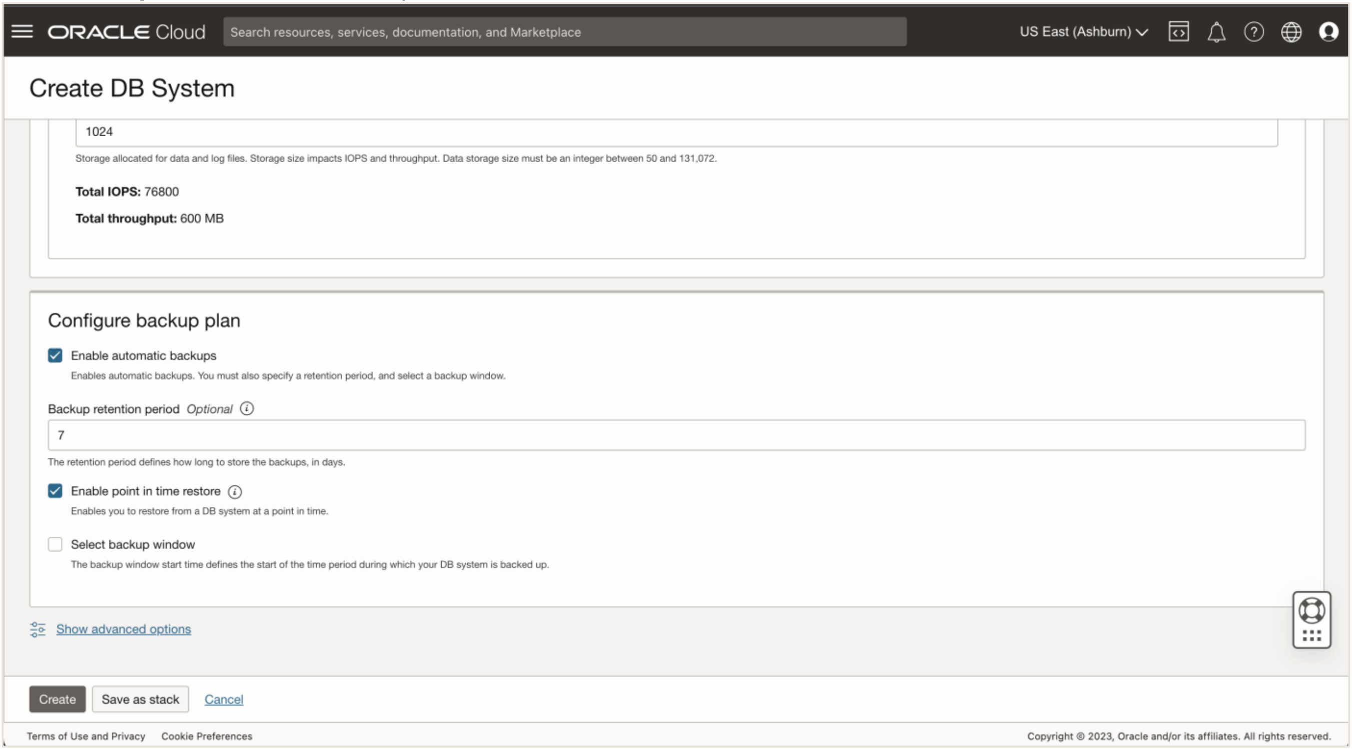
71. For **Configuring Networking** - choose the earlier created VCN and make sure the **Private Subnet** is selected under **Subnet in <compartment-name>**. For **Configure Placement** leave it as-is.

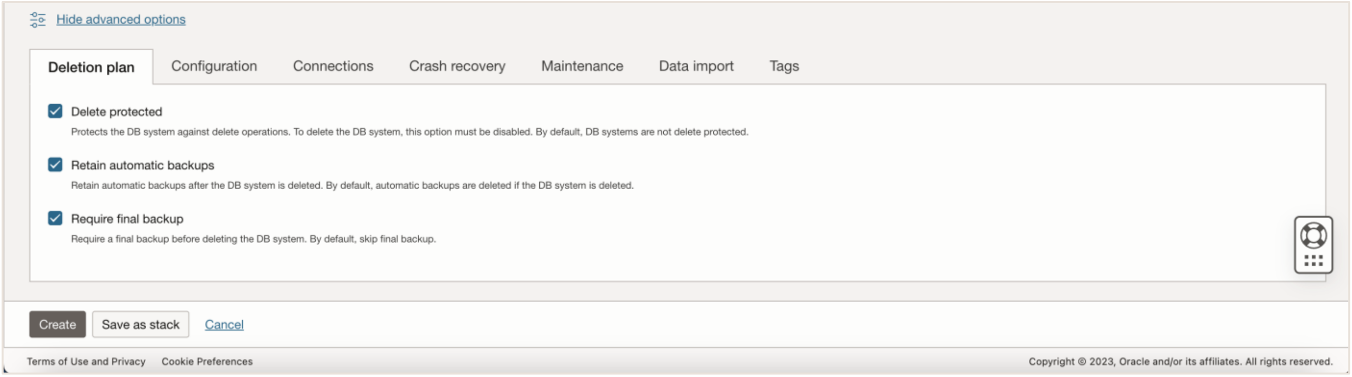


72. **Configure hardware** (OCPU and Memory) for MySQL by choosing an appropriate DB Shape. For this guide, we will use the default HeatWave shape. For the **Data Storage Size** be sure to make the size large enough for future growth.

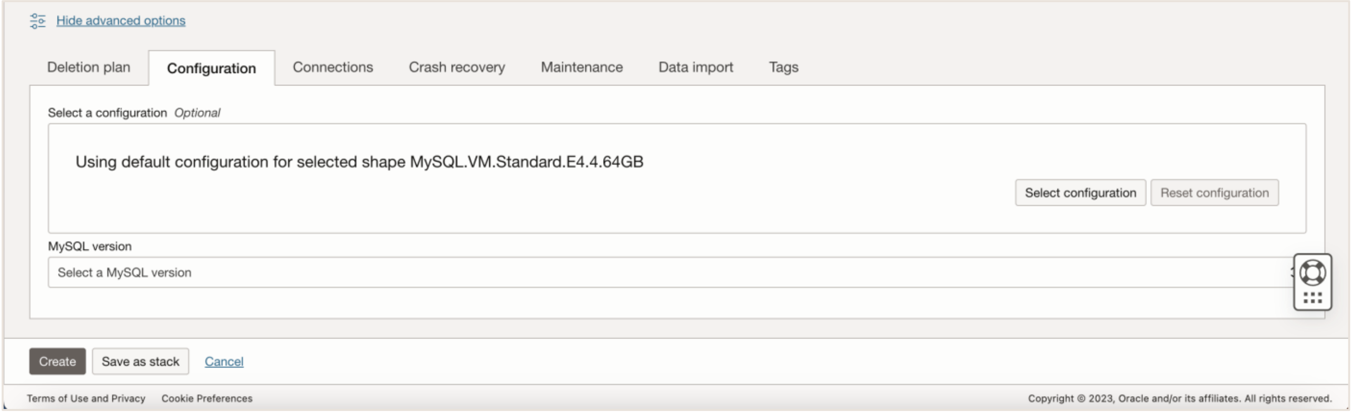


73. **Configure a backup plan** according to what suits your needs. Lastly, scroll down until you see **Show advanced options**. Click on it to expand.

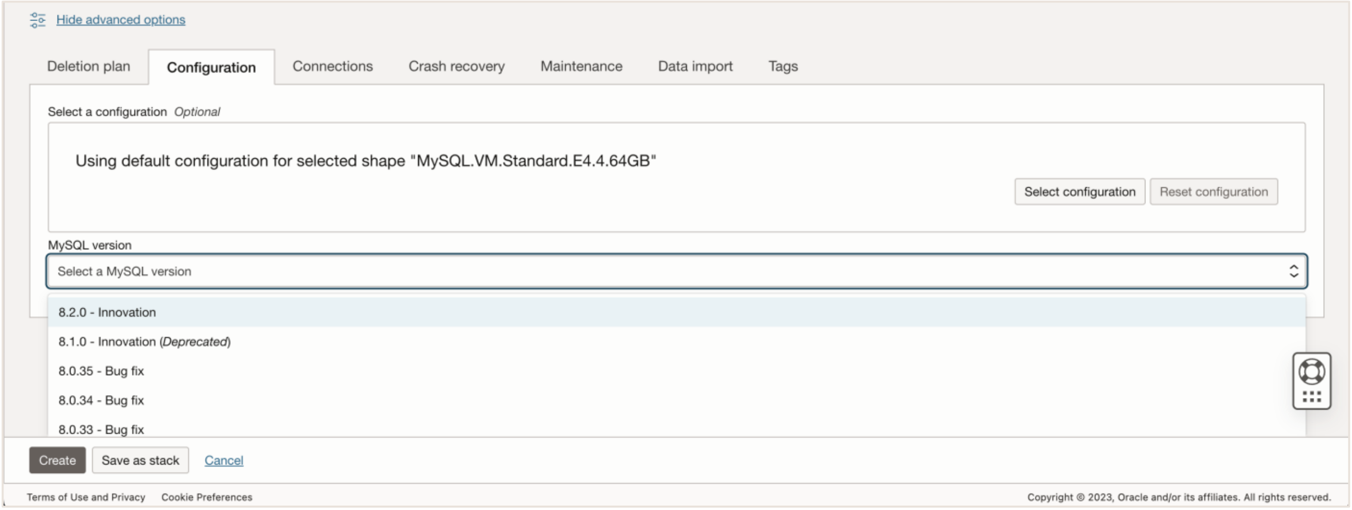




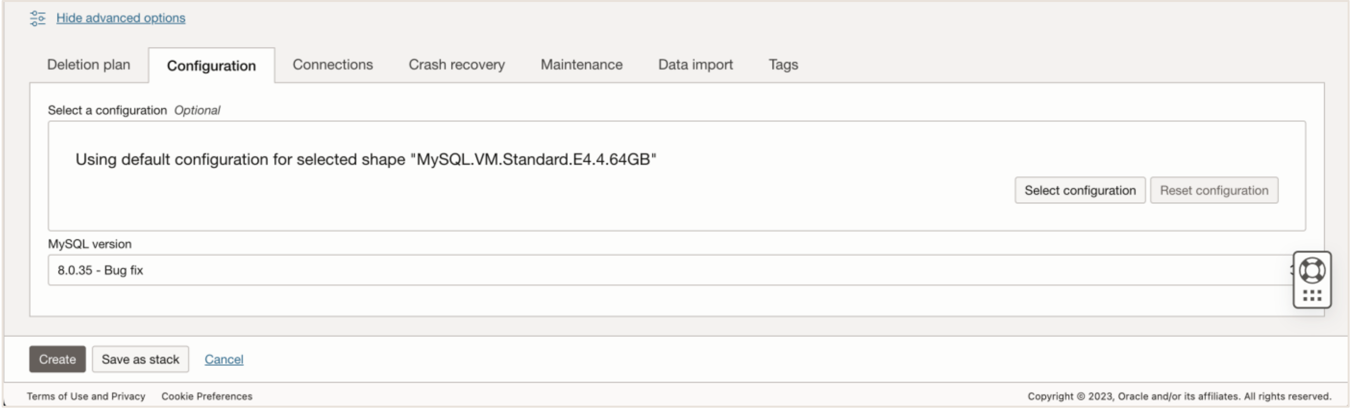
74. From the advanced options screen, go to the **Configuration** tab. If you have a custom configuration that you would like to apply to your HeatWave MySQL instance - you can do so by clicking **Select configuration**. Custom configurations allow you to tweak MySQL variables (i.e., max connections, binary log expire seconds, etc.) rather than using the default values. You must create a custom configuration in advance before applying. For more information regarding custom configurations, see [Configuration of a DB System](). For this guide, we have chosen the default configuration.



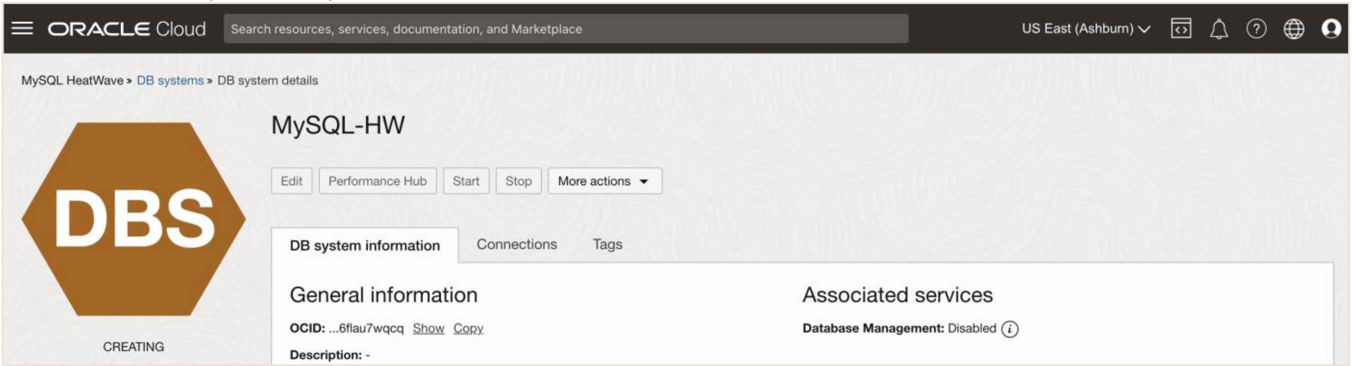
75. For **MySQL version**, choose either **Innovation** or **Bug fix**. With the new MySQL versioning model, you have the flexibility to select an innovation or a bug fix release. Both releases are production-grade quality. MySQL innovation releases allow you to access the latest features and improvements. Innovation releases are ideal for fast-paced development environments with high levels of automated tests and modern continuous integration techniques for faster upgrade cycles. MySQL bug fix releases (aka long-term support releases) allow you to reduce the risks associated with changes in the database software behavior, as these releases only contain necessary fixes (bugfix and security patches). For more information regarding MySQL innovation and bug fix releases, see Introducing MySQL Innovation and Bug fix versions. For this guide, we have chosen **8.0.35 - Bug fix**.



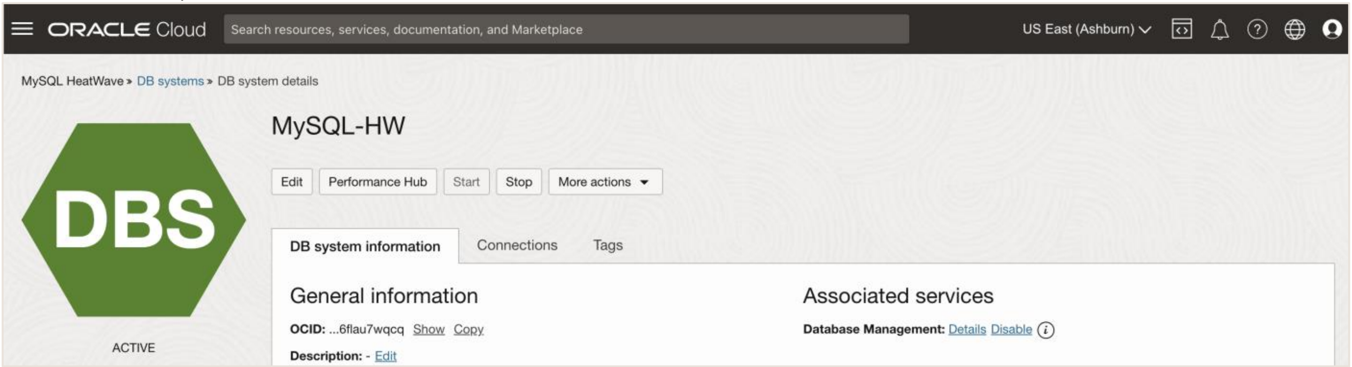
76. Click **Create** to finish the HeatWave MySQL DB system creation process.



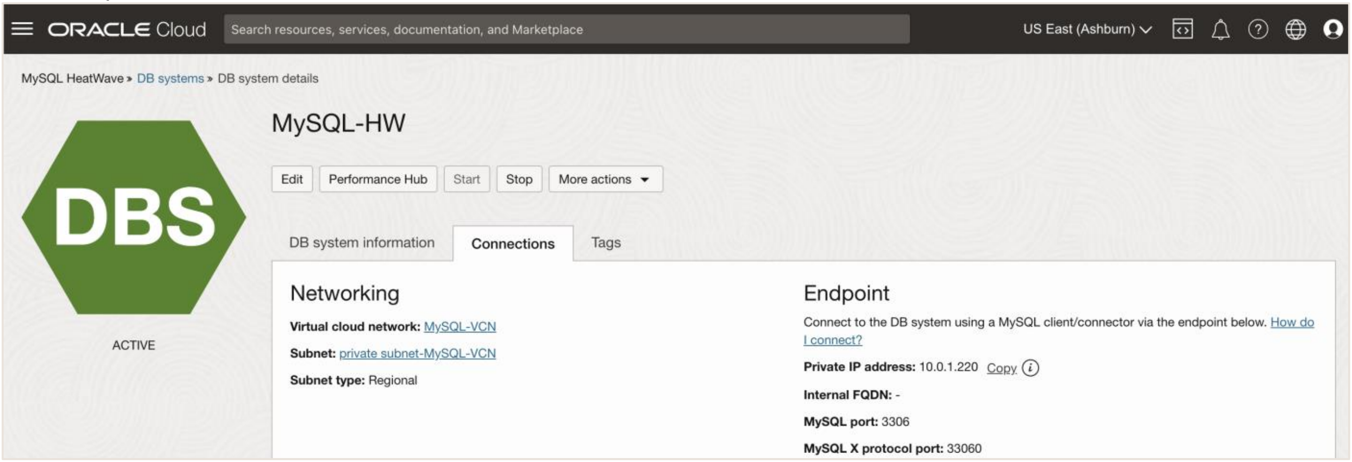
77. Your HeatWave MySQL DB system will start **CREATING**.



78. Within a few minutes, HeatWave MySQL DB system will change its state from CREATING to **ACTIVE** once the instance is ready.



79. On the same DB system details page, click **Connections** to grab the **private IP address** for HeatWave MySQL. Save the private IP Address for later use.

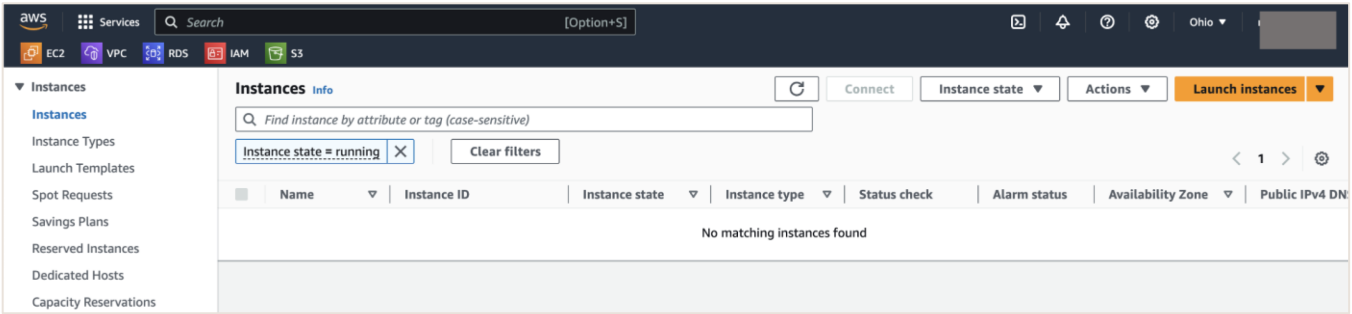


Note: you can navigate to the **DB System Details** page by going to the Navigation menu in OCI. Click **Databases** and click **HeatWave MySQL**. Click on the name of your MySQL DB System to open the **DB System Details** page.

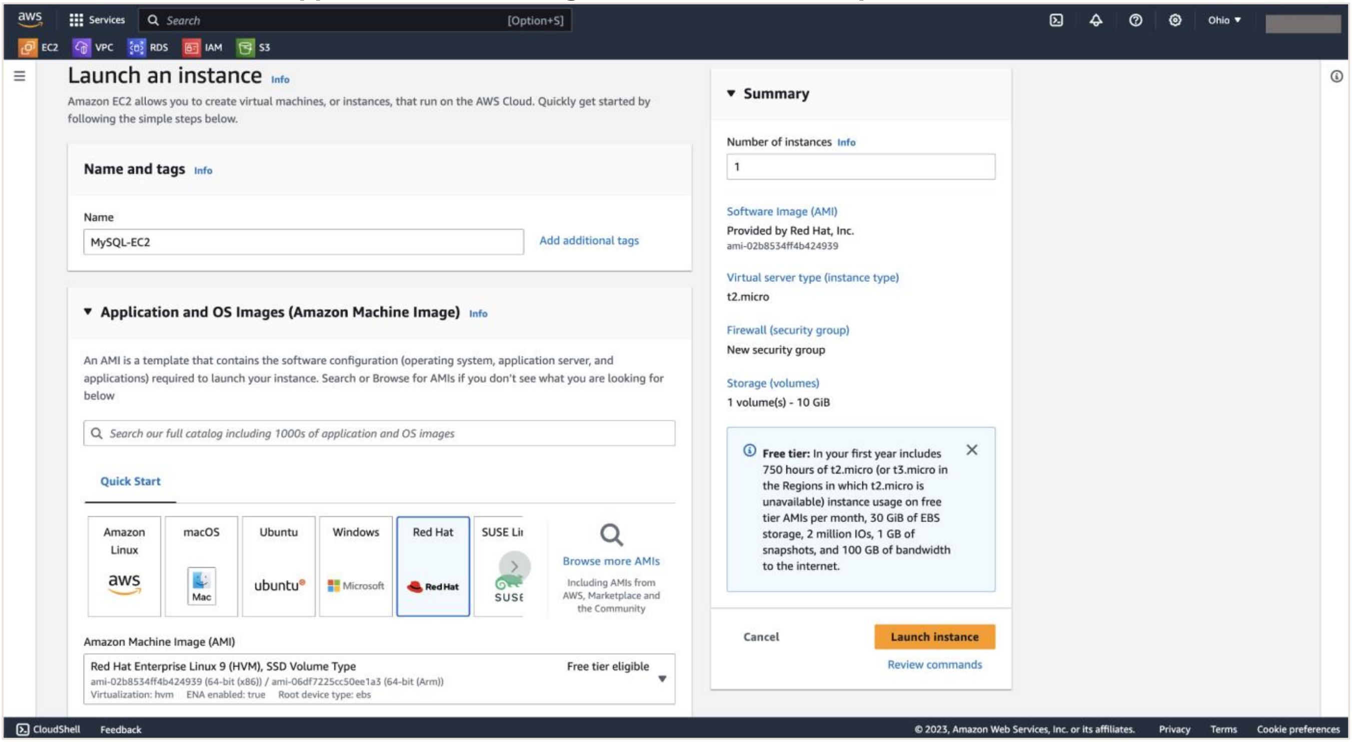
ORACLE

IV) Install MySQL Shell 8.2 or above on an EC2 instance that can connect to Amazon Aurora MySQL.

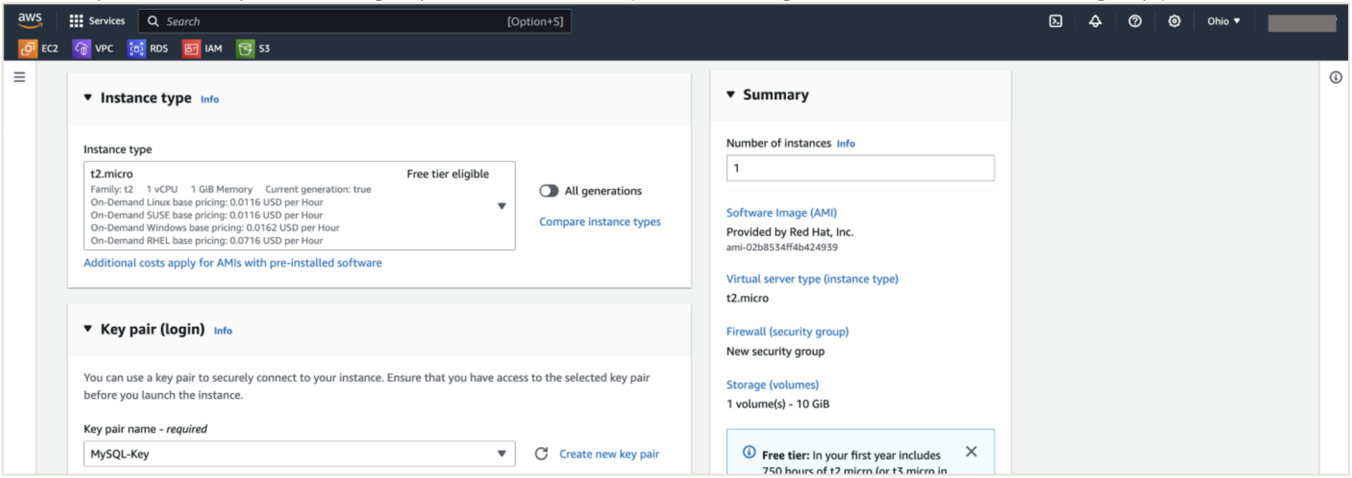
80. Login to [AWS](). From the Services menu, go to **Compute** and select **EC2**.
81. Click **Launch instance**.



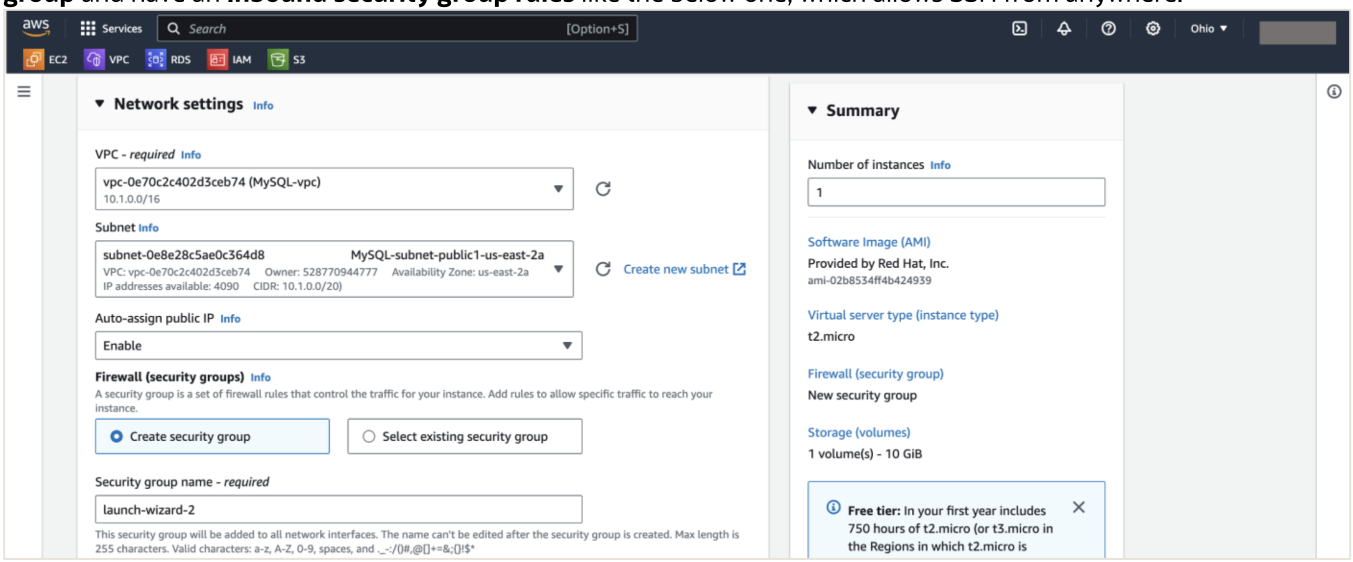
82. Enter an **EC2 name**. For **Application and OS Images**, select **Red Hat Enterprise Linux 9**.

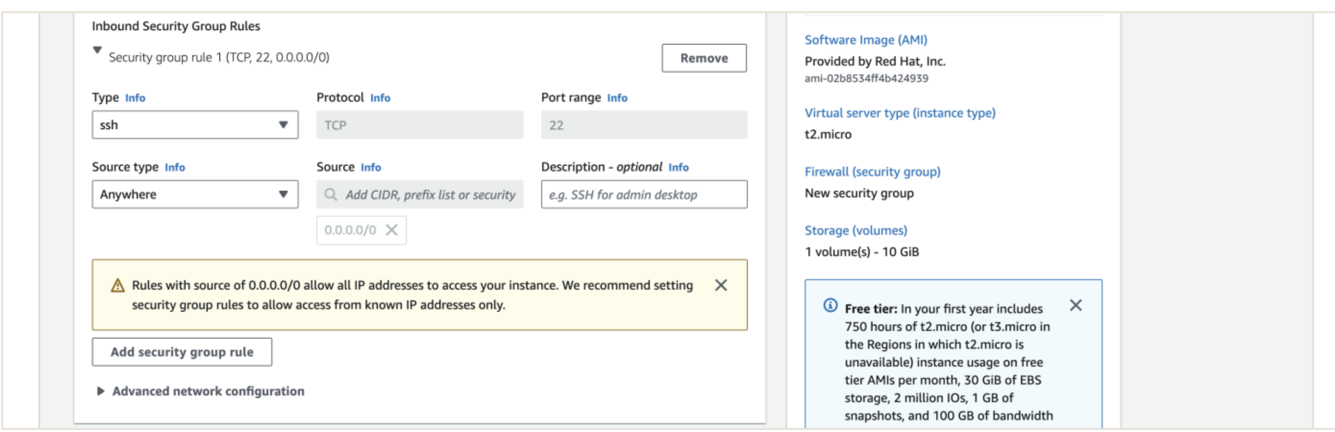


83. For **Instance type**, choose an instance type you think is appropriate. If you have large amounts of data - provisioning an EC2 with more vCPUs and Memory will speed up the migration process. For the **Key pair** section, you can use your existing keys or create a new pair. For this guide, we will use an existing key pair.

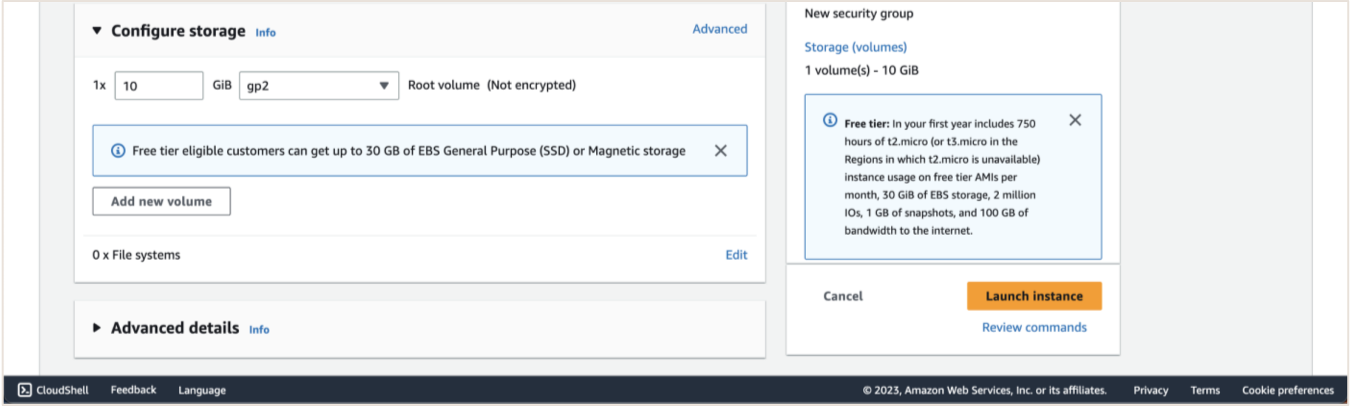


84. Under Network settings, ensure that the correct **VPC** (the VPC that is associated with your Aurora instance) and **Subnet** are selected. For this guide - we have decided to deploy the EC2 instance inside a public subnet. For **Auto-assign public IP** select **Enable**. Under the **Firewall (security groups)**, choose **Create security group** and have an **Inbound security group rules** like the below one, which allows SSH from anywhere.

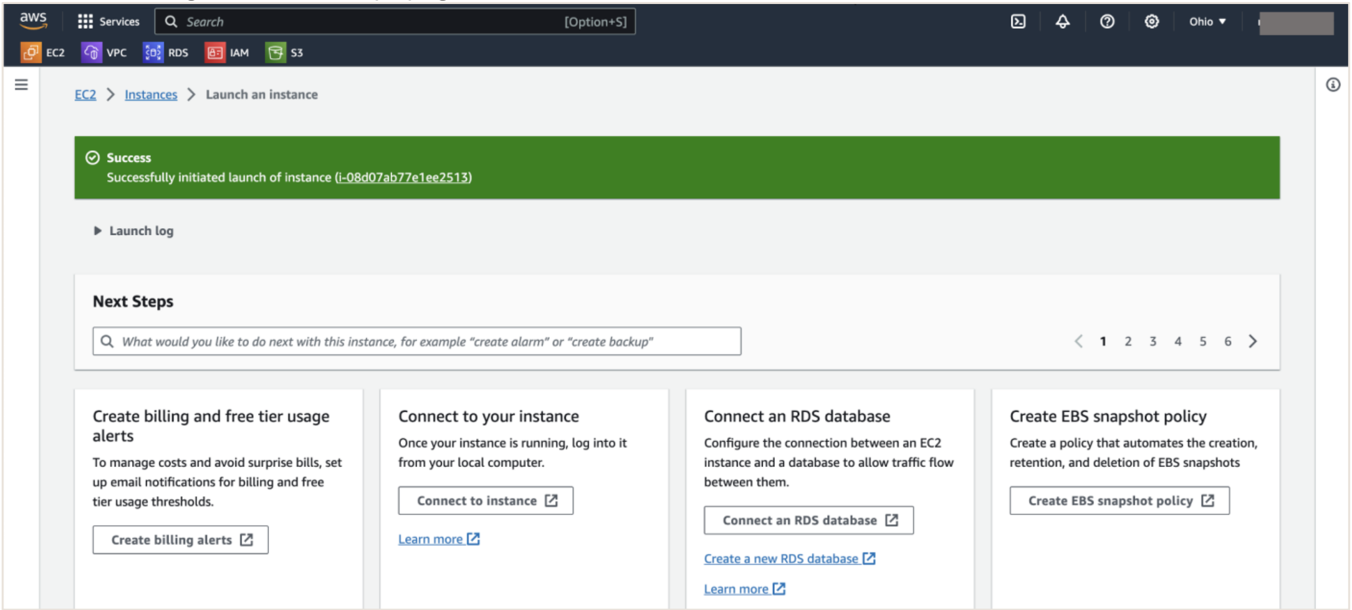




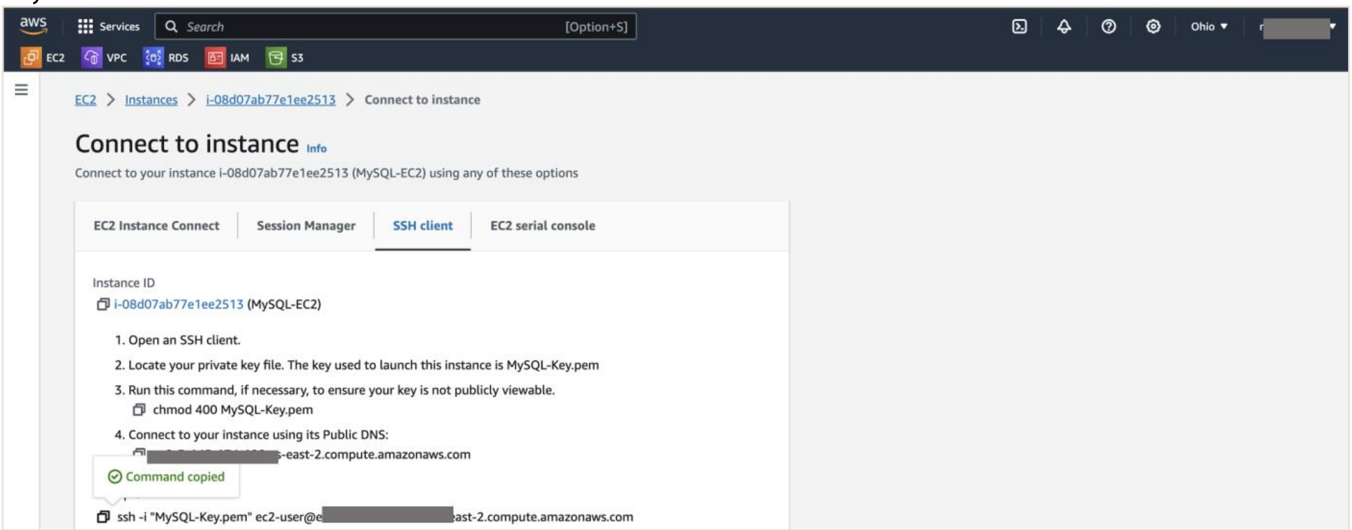
85. Leave everything as-is and click **Launch instance**.



86. You will be brought to a Next Steps page. Here, click **Connect to instance**.



87. If you are using the SSH client to connect to your EC2 instance, copy the **Example** SSH command and login to your EC2 instance.



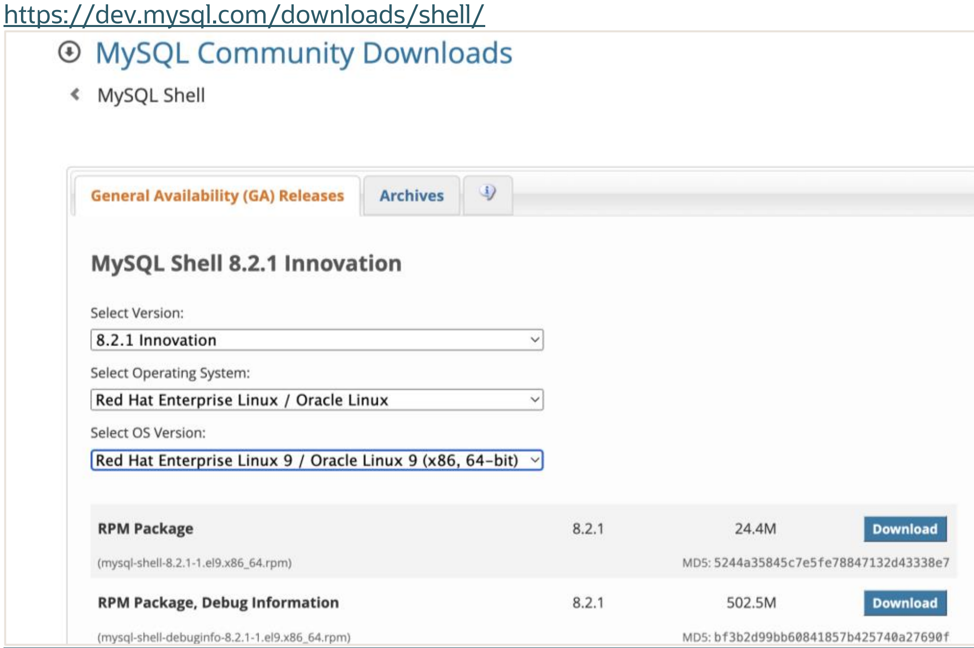
88. You can SSH into EC2 using the below command:

```
$ ssh -i </path/to/private-ssh-key> ec2-user@<ec2-Public-DNS>
```

Note: after running the above SSH command, if prompted **Are you sure you want to continue connecting (yes/no/[fingerprint])?**, type **yes**.

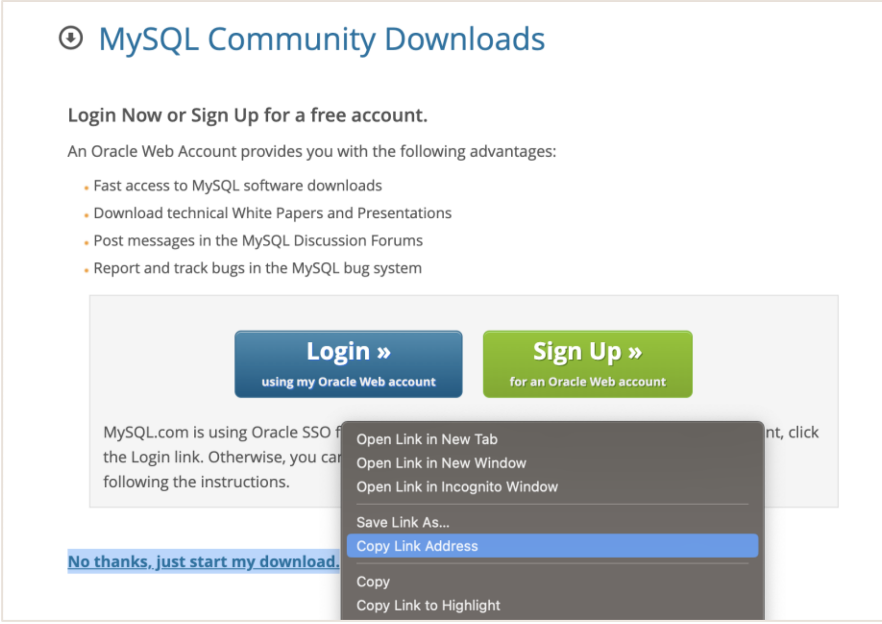
89. We are now successfully connected to the EC2 instance.

90. After making a connection to the EC2 instance, go to the below website and download MySQL Shell 8.2 on your EC2 instance. From the MySQL Shell download page, ensure **8.2.x Innovation or higher** is selected under **Select Version**. MySQL Shell 8.2 is fully compatible with MySQL 8.2, 8.1, 8.0, and 5.7. For **Operating System** and **OS Version** - pick the appropriate option depending on the OS and the OS Version that you are running. Click **Download**.
https://dev.mysql.com/downloads/shell/



Note: for this guide, we will show you how to install MySQL Shell on a Linux environment. For other environments, see Installing MySQL Shell on Windows, Installing MySQL Shell on Linux, and Installing MySQL Shell on macOS.

91. Right-click on **No thanks, just start my download** and click **Copy link address**.

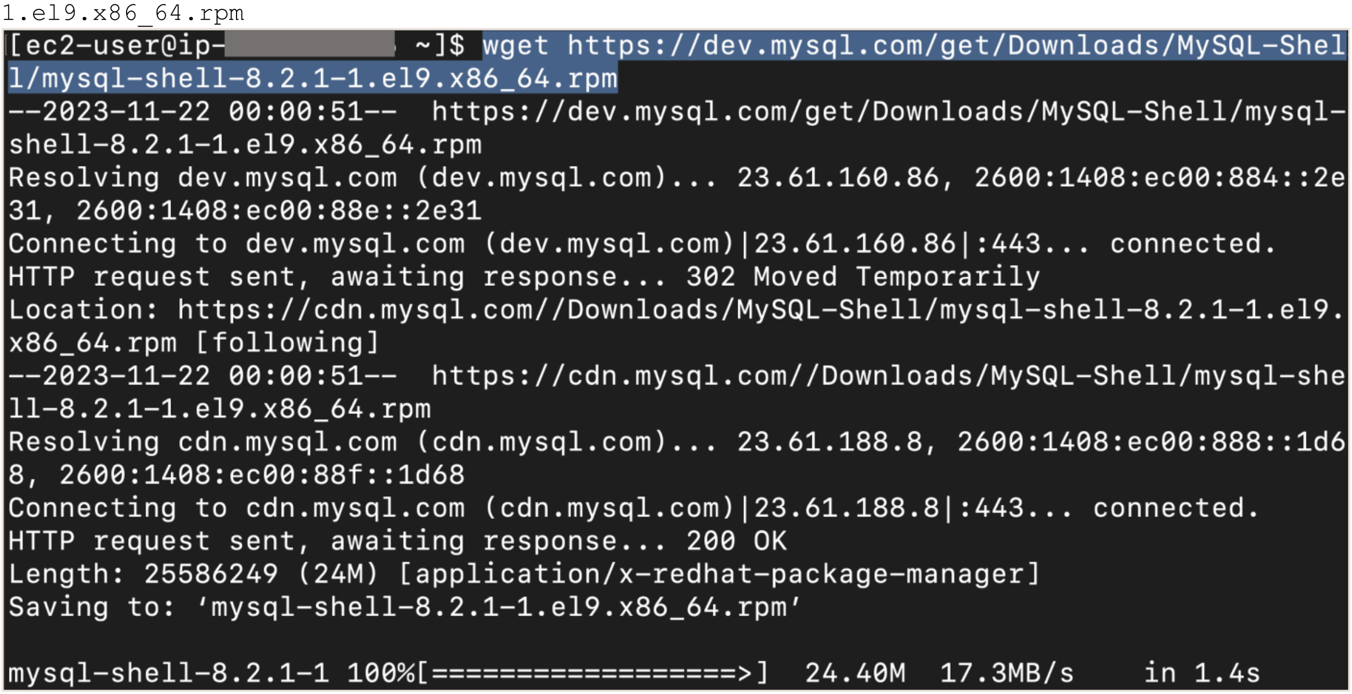


92. Go back to the EC2 instance that can connect to your Amazon Aurora MySQL and execute the below command to download MySQL Shell:

```
$ wget <MySQL-Shell-Download-Link>
```

Replace the link with what you have.

```
$ wget https://dev.mysql.com/get/Downloads/MySQL-Shell/mysql-shell-8.2.1-1.el9.x86_64.rpm
```



```
[ec2-user@ip-[redacted] ~]$ wget https://dev.mysql.com/get/Downloads/MySQL-Shell/mysql-shell-8.2.1-1.el9.x86_64.rpm
--2023-11-22 00:00:51--  https://dev.mysql.com/get/Downloads/MySQL-Shell/mysql-shell-8.2.1-1.el9.x86_64.rpm
Resolving dev.mysql.com (dev.mysql.com)... 23.61.160.86, 2600:1408:ec00:884::2e31, 2600:1408:ec00:88e::2e31
Connecting to dev.mysql.com (dev.mysql.com)|23.61.160.86|:443... connected.
HTTP request sent, awaiting response... 302 Moved Temporarily
Location: https://cdn.mysql.com//Downloads/MySQL-Shell/mysql-shell-8.2.1-1.el9.x86_64.rpm [following]
--2023-11-22 00:00:51--  https://cdn.mysql.com//Downloads/MySQL-Shell/mysql-shell-8.2.1-1.el9.x86_64.rpm
Resolving cdn.mysql.com (cdn.mysql.com)... 23.61.188.8, 2600:1408:ec00:888::1d68, 2600:1408:ec00:88f::1d68
Connecting to cdn.mysql.com (cdn.mysql.com)|23.61.188.8|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 25586249 (24M) [application/x-redhat-package-manager]
Saving to: ‘mysql-shell-8.2.1-1.el9.x86_64.rpm’

mysql-shell-8.2.1-1 100%[==================>]  24.40M  17.3MB/s    in 1.4s
```

Note: to install `wget` on the EC2, execute:

```
$ sudo yum install wget
```

93. After downloading the MySQL Shell rpm, install MySQL Shell:

```
$ sudo yum localinstall mysql-shell*
```

```
[ec2-user@ip-[redacted] ~]$ sudo yum localinstall mysql-shell-8.2.1-1.el9.x86_64.rpm
Updating Subscription Management repositories.
Unable to read consumer identity

This system is not registered with an entitlement server. You can use subscription-manager to register.

Last metadata expiration check: 0:00:57 ago on Wed 22 Nov 2023 12:00:04 AM UTC.
Dependencies resolved.
================================================================================
 Package             Architecture  Version            Repository          Size
================================================================================
Installing:
 mysql-shell         x86_64        8.2.1-1.el9        @commandline        24 M

Transaction Summary
================================================================================
Install  1 Package
```

94. You can now verify if MySQL Shell has successfully installed on your EC2 instance by executing the below command:

```
$ mysqlsh -version
```

```
[ec2-user@ip-[redacted] ~]$ mysqlsh --version
mysqlsh   Ver 8.2.1 for Linux on x86_64 - for MySQL 8.2.0 (MySQL Community Server (GPL))
[ec2-user@ip-[redacted] ~]$
```

95. To login to your Amazon Aurora MySQL using MySQL Shell, use the below commands:

```
$ mysqlsh <user>@<hostname>:<port-number>
```

-OR-

```
$ mysqlsh -u <user> -p -h <hostname> -P <port-number>
```

```
[ec2-user@ip-[redacted] ~]$ mysqlsh admin@database-1-instance-1.[redacted].us
-east-2.rds.amazonaws.com
Please provide the password for 'admin@database-1-instance-1.[redacted].us-ea
st-2.rds.amazonaws.com': ********
Save password for 'admin@database-1-instance-1.[redacted].us-east-2.rds.amazo
naws.com'? [Y]es/[N]o/Ne[v]er (default No): Y
MySQL Shell 8.2.1

Copyright (c) 2016, 2023, Oracle and/or its affiliates.
Oracle is a registered trademark of Oracle Corporation and/or its affiliates.
Other names may be trademarks of their respective owners.

Type '\help' or '\?' for help; '\quit' to exit.
Creating a session to 'admin@database-1-instance-1.[redacted].us-east-2.rds.a
mazonaws.com'
Fetching schema names for auto-completion... Press ^C to stop.
Your MySQL connection id is 5104
Server version: 5.7.12-log MySQL Community Server (GPL)
No default schema selected; type \use <schema> to set one.
 MySQL  database-1-instance-1.[redacted].us-east-2  JS >
```

Note: you can interact with MySQL Shell using JavaScript, Python, or SQL mode. The default is JavaScript. To switch between the different modes, execute `/js` for JavaScript, `/py` for Python, and `/sql` for SQL mode inside MySQL Shell. To exit out of MySQL Shell, execute `/q`.

V) For your Amazon Aurora MySQL, ensure `log_bin` is set to `1`, ensure `binlog_format` is set to `ROW`, and execute the `mysql.rds_set_configuration` stored procedure to retain binary logs.

96. Stay connected to your Aurora instance and execute the below commands to ensure your Aurora is configured correctly for the live migration.

```
MySQL JS> \sql
MySQL SQL> SELECT @@log_bin;
MySQL SQL> SELECT @@binlog_format;
```

```
 MySQL  database-1-instance-1.[redacted].us-east-2 JS > \sql
Switching to SQL mode... Commands end with ;
Fetching global names for auto-completion... Press ^C to stop.
 MySQL  database-1-instance-1.[redacted].us-east-2 SQL > SELECT @@log_bin;
+-----------+
| @@log_bin |
+-----------+
|         1 |
+-----------+
1 row in set (0.0005 sec)
 MySQL  database-1-instance-1.[redacted].us-east-2 SQL > SELECT @@binlog_format;
+-----------------+
| @@binlog_format |
+-----------------+
| ROW             |
+-----------------+
1 row in set (0.0008 sec)
 MySQL  database-1-instance-1.[redacted].us-east-2 SQL > SHOW BINARY LOGS;
+----------------------------+-----------+
| Log_name                   | File_size |
+----------------------------+-----------+
| mysql-bin-changelog.000003 |       154 |
| mysql-bin-changelog.000004 |    692708 |
+----------------------------+-----------+
2 rows in set (0.0006 sec)
 MySQL  database-1-instance-1.[redacted].us-east-2 SQL >
```

Note: you must have a value of `1` for `log_bin` and a value of `ROW` for `binlog_format`.

97. After confirming you have binary logs on Aurora, execute the below stored procedure to retain the binary logs - as [Amazon Aurora normally purges a binary log as soon as possible](). For us to perform the live database migration - we will need to retain the current binary log that is in use/will be used during the data export of Aurora and the binary logs that will be generated afterwards. The binary logs will be needed until the replication setup is completed on OCI. Since the sample database 'world' (the one that will be migrated to HeatWave MySQL on OCI for the purposes of this step-by-step guide) is fairly small, we will set the binary log retention hours to 24. Set the binlog retention hours required depending on the data that you are migrating, high volumes of data will require a longer retention period; monitor the usage of your Aurora system afterwards.

```
MySQL SQL> call mysql.rds_set_configuration('binlog retention hours', 24);
```

```
 MySQL  database-1-instance-1.[redacted].us-east-2.rds SQL > call mysql.rds_set_config
uration('binlog retention hours', 24);
Query OK, 0 rows affected (0.0114 sec)
 MySQL  database-1-instance-1.[redacted].us-east-2.rds SQL >
```

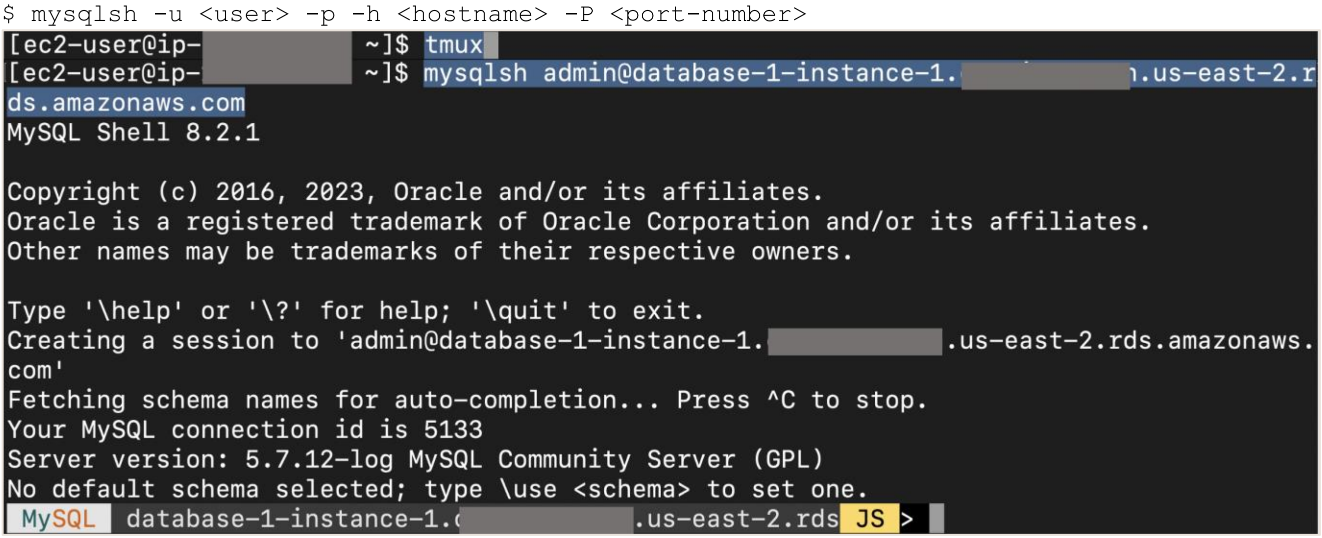
VI) Connect to Amazon Aurora MySQL using MySQL Shell and create a replication user. Afterwards, execute the MySQL Shell `util.copyInstance()` utility to export all schemas (including users, indexes, routines, triggers) from Amazon Aurora MySQL to HeatWave MySQL on OCI. After the `util.copyInstance()` utility finishes, save the MySQL Shell `Dump_metadata` values.

98. Before proceeding with the below steps, it is highly recommended that you use a command like **screen** or **tmux**. These commands will allow you to reconnect to a dropped session in case your connection drops in the middle of performing the MySQL Shell export using `util.copyInstance()`. For small databases, the screen or tmux may not be necessary. For this guide, we will use tmux. To learn more about tmux, see [A beginner's guide to tmux](). Below are the basics of using the tmux command:
    - Install tmux on Linux: `$ sudo yum install tmux`
    - Start a new tmux session, from your terminal execute: `$ tmux`
    - List all the active tmux sessions: `$ tmux ls`
    - Detach from a tmux session and leave it running in the background: `$ Ctrl+B d`
    - Attach a tmux session running in the background: `$ tmux attach`
    - End a tmux session: `$ Ctrl+B &`
99. Start a tmux session and connect to your Amazon Aurora MySQL using MySQL Shell on EC2.

```
$ tmux
$ mysqlsh <user>@<hostname>:<port-number>
```

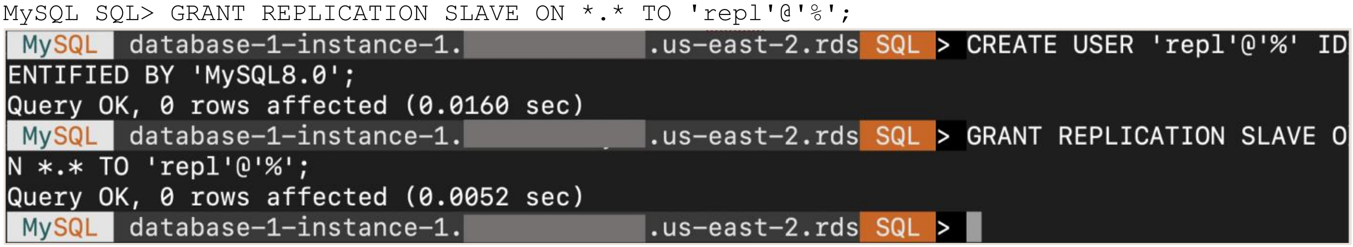
-OR-

```
$ mysqlsh -u <user> -p -h <hostname> -P <port-number>
```



100. Change to the SQL mode of MySQL Shell and [create a replication user](), we will use this user to establish a replication connection from Aurora MySQL to HeatWave MySQL on OCI.

```
MySQL SQL> CREATE USER 'repl'@'%' IDENTIFIED BY '<password>';
MySQL SQL> GRANT REPLICATION SLAVE ON *.* TO 'repl'@'%';
```



101. Change to the JavaScript mode of MySQL Shell and run the `util.copyInstance()` utility to export all Amazon Aurora MySQL data into HeatWave MySQL on OCI.

```
MySQL JS> \js
MySQL JS> util.copyInstance('mysql://admin@10.0.1.220', {"compatibility":
["force_innodb", "skip_invalid_accounts", "strip_definers",
"strip_restricted_grants", "strip_tablespaces", "ignore_wildcard_grants",
"strip_invalid_grants", "create_invisible_pks"], updateGtidSet: "append", users:
"true", threads: 4, dryRun:"true"})
```

Note: replace the username (`admin`) and IP address (`10.0.1.220`) with your HeatWave MySQL username and IP address (not the Amazon Aurora MySQL username and IP address).

```
 MySQL  database-1-instance-1.[redacted].us-east-2.rds  JS > util.copyInstance('mysql://admin@10.0.1.220', {"compatibility": ["force_innodb", "skip_invalid_accounts", "strip_definers", "strip_restricted_grants", "strip_tablespaces", "ignore_wildcard_grants", "strip_invalid_grants", "create_invisible_pks"], updateGtidSet: "append", users: "true", threads: 4, dryRun:"true"})
Please provide the password for 'admin@10.0.1.220': ********
Save password for 'admin@10.0.1.220'? [Y]es/[N]o/Ne[v]er (default No): Y
Copying DDL, Data and Users from in-memory FS, source: ip-[redacted]:3306, target: fw5erxp3afmvjpsu:3306.
SRC: dryRun enabled, no locks will be acquired and no files will be created.
NOTE: SRC: Backup lock is not supported in MySQL 5.7 and DDL changes will not be blocked. The dump may fail with an error if schema changes are made while dumping.
SRC: Acquiring global read lock
WARNING: SRC: The current user lacks privileges to acquire a global read lock using 'FLUSH TABLES WITH READ LOCK'. Falling back to LOCK TABLES...
SRC: Table locks acquired
Initializing - done
SRC: 1 out of 5 schemas will be dumped and within them 3 tables, 0 views.
SRC: 3 out of 4 users will be dumped.
Gathering information - done
SRC: All transactions have been started
SRC: Global read lock has been released
NOTE: SRC: When migrating to MySQL HeatWave Service, please always use the latest available version of MySQL Shell.
```

[… output truncated]

```
TGT: Starting data load
?% (0 bytes / ?), 0.00 B/s, 0 / 3 tables done
TGT: Executing common postamble SQL
Recreating indexes - done
TGT: Appending dumped gtid set to GTID_PURGED
TGT: No data loaded.
TGT: 0 accounts were loaded
TGT: 0 warnings were reported during the load.

---
Dump_metadata:
  Binlog_file: mysql-bin-changelog.000004
  Binlog_position: 693142
  Executed_GTID_set: [redacted]:1-13

 MySQL  database-1-instance-1.[redacted].us-east-2.rds  JS >
```

102. Running the above step 101 command may generate **Errors** regarding **table locks** (see image below).

```
WARNING: SRC: The current user lacks privileges to acquire a global read lock using 'FLUSH TAB
LES WITH READ LOCK'. Falling back to LOCK TABLES...
ERROR: SRC: The current user does not have required privileges to execute FLUSH TABLES WITH RE
AD LOCK.
    Backup lock is not supported in MySQL 5.7 and DDL changes cannot be blocked.
    The gtid_mode system variable is set to OFF or OFF_PERMISSIVE.
    The log_bin system variable is set to OFF or the current user does not have required privi
leges to execute SHOW MASTER STATUS.
The consistency of the dump cannot be guaranteed.
ERROR: SRC: Unable to acquire global read lock neither table read locks.
SRC: Global read lock has been released
Initializing - done
Util.copyInstance: While 'Initializing': Unable to lock tables: Consistency check has failed.
(MYSQLSH 52002)
```

103. If you do encounter the table lock problem (if and only if) run the same command as in step 101 but this time add an additional option: `consistent: "false"` and re-run the command.

```
MySQL JS> util.copyInstance('mysql://admin@10.0.1.220', {"compatibility":
["force_innodb", "skip_invalid_accounts", "strip_definers",
"strip_restricted_grants", "strip_tablespaces", "ignore_wildcard_grants",
"strip_invalid_grants", "create_invisible_pks"], updateGtidSet: "append", users:
"true", threads: 4, dryRun:"true", consistent: "false"})
```

Note:

- `util.copyInstance(connectionData[, options])`: MySQL instance copy utility enables copying of an entire instance to another server. By default, this utility includes all schemas, users, indexes, routines, and triggers. See Copy Utilities.
  - `connectionData`: Defines the connection details for the destination server you want to copy to.
- `compatibility`: Apply the specified requirements for compatibility with HeatWave MySQL for all tables in the dump output, altering the dump files as necessary.
  - `force_innodb`: Change `CREATE TABLE` statements to use the InnoDB storage engine for any tables that do not already use it.
  - `skip_invalid_accounts`: You cannot export a user that has no password defined. This option skips any such users.
  - `strip_definers`: Remove the `DEFINER` clause from views, routines, events, and triggers, so these objects are created with the default definer (the user invoking the schema), and change the `SQL SECURITY` clause for views and routines to specify `INVOKER` instead of `DEFINER`. HeatWave MySQL requires special privileges to create these objects with a definer other than the user loading the schema. If your security model requires that views and routines have more privileges than the account querying or calling them, you must manually modify the schema before loading it.
  - `strip_restricted_grants`: Certain privileges are restricted in HeatWave MySQL. Privileges such as `RELOAD`, `FILE`, `SUPER`, `BINLOG_ADMIN`, and `SET_USER_ID`. You cannot create users granting these privileges. This option strips these privileges from dumped `GRANT` statements.

    - `strip_tablespaces`: Tablespaces have some restrictions in HeatWave MySQL. If you need tables created in their default tablespaces, this option strips the `TABLESPACE=` option from `CREATE TABLE` statements.
    - `ignore_wildcard_grants`: If enabled, ignores errors from grants on schemas with wildcards, which are interpreted differently in systems where the partial_revokes system variable is enabled.
    - `strip_invalid_grants`: If enabled, strips grant statements which would fail when users are copied. Such as grants referring to a specific routine which does not exist.
    - `create_invisible_pks`: Primary keys are required by High Availability and HeatWave. If you intend to export data for use in a highly available DB system or a HeatWave DB system, add primary keys as they are not defined on the tables. This compatibility flag adds invisible primary keys to each table that requires them.
- `updateGtidSet: append`: If your Aurora MySQL is using GTIDs, for inbound replication, add the transaction IDs from the source `gtid_executed` GTID set to the replica `gtid_purged` GTID set. This lets you begin replication from the source without re-executing every past transaction from the source. Adding the GTIDs to `gtid_purged` tells the replica that those transactions have already been executed, although they are not present in the source binary log. This must be set to `append` during a live migration.
- `users`: Include (`true`) or exclude (`false`) users and their roles and grants in the dump.
- `threads`: (Optional) The number of parallel threads to use to copy chunks of data from the MySQL instance. Each thread has its own connection to the MySQL instance. The default is 4. The copy utilities require twice the number of threads, one thread to copy and one thread to write. If threads is set to N, 2N threads are used.
- `consistent`: Enable (`true`) or disable (`false`) consistent data dumps by locking the instance for backup during the dump.
- `dryRun`: Displays information about the copy with the specified set of options, and about the results of HeatWave MySQL Service compatibility checks, but does not proceed with the copy. Setting this option enables you to list out all the compatibility issues before starting the copy.

104. Once you have run the command in step 101/103 and did not see any errors in the output (warnings are okay), run the same step 101/103 command but this time change the `dryRun` option to `false`.

```
MySQL JS> util.copyInstance('mysql://admin@10.0.1.220', {"compatibility":
["force_innodb", "skip_invalid_accounts", "strip_definers",
"strip_restricted_grants", "strip_tablespaces", "ignore_wildcard_grants",
"strip_invalid_grants", "create_invisible_pks"], updateGtidSet: "append", users:
"true", threads: 4, dryRun:"false"})
```

Note: replace the username (`admin`) and IP address (`10.0.1.220`) with your HeatWave MySQL username and IP address (not the Amazon Aurora MySQL username and IP address). Add `consistent: "false"` to your step 104 command if you had encountered the table lock issue.

```
 MySQL  database-1-instance-1.[redacted].us-east-2.rds  JS > util.copyInstance('mysql://admin@10.0.1.220', {"compatibility": ["force_innodb", "skip_invalid_accounts", "strip_definers", "strip_restricted_grants", "strip_tablespaces", "ignore_wildcard_grants", "strip_invalid_grants", "create_invisible_pks"], updateGtidSet: "append", users: "true", threads: 4, dryRun:"false"})
Copying DDL, Data and Users from in-memory FS, source: ip-[redacted]3306, target: fw5erxp3afmvjpsu:3306.
NOTE: SRC: Backup lock is not supported in MySQL 5.7 and DDL changes will not be blocked. The dump may fail with an error if schema changes are made while dumping.
SRC: Acquiring global read lock
WARNING: SRC: The current user lacks privileges to acquire a global read lock using 'FLUSH TABLES WITH READ LOCK'. Falling back to LOCK TABLES...
SRC: Table locks acquired
Initializing - done
SRC: 1 out of 5 schemas will be dumped and within them 3 tables, 0 views.
SRC: 3 out of 4 users will be dumped.
```

[… output truncated]

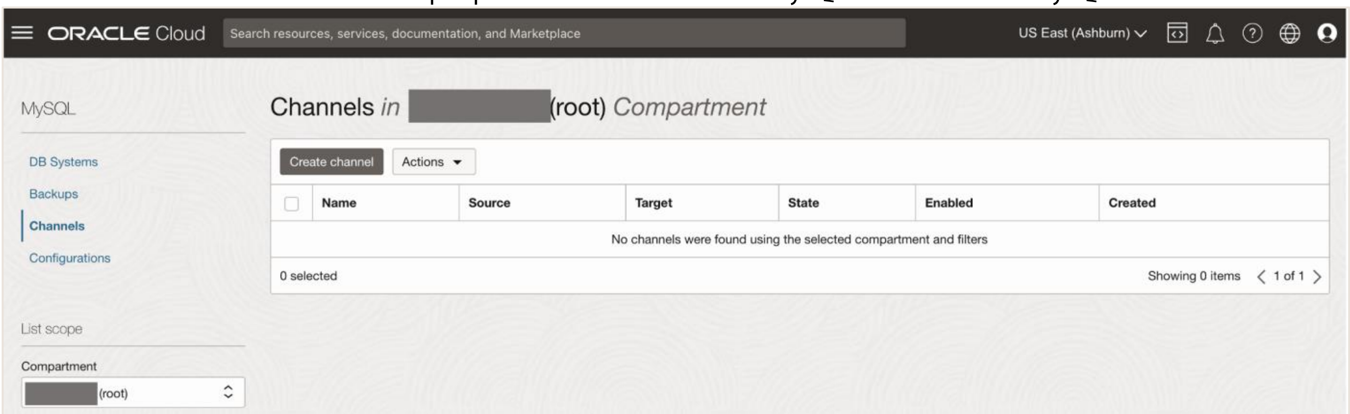
```
100% (194.61 KB / 194.61 KB), 0.00 B/s, 3 / 3 tables done
Recreating indexes - done
TGT: 3 chunks (5.30K rows, 194.61 KB) for 3 tables in 1 schemas were loaded in 1 sec (avg throughput 194.61 KB/s)
TGT: 2 accounts were loaded
TGT: 0 warnings were reported during the load.

---
Dump_metadata:
  Binlog_file: mysql-bin-changelog.000004
  Binlog_position: 693142
  Executed_GTID_set: [redacted]:1-13

 MySQL  database-1-instance-1.[redacted].us-east-2.rds  JS >
```

105. Once the copy utility finishes, if your Aurora MySQL uses binary log positioning - save the `Binlog_file` and `Binlog_position` values from the MySQL Shell latest `Dump_metadata` for later use. This will let the HeatWave MySQL instance on OCI know where to start the replication from for data synchronization. If your Aurora MySQL uses GTIDs, you don't need to save any of the MySQL Shell `Dump_metadata` values. The initial data transfer from Aurora MySQL to HeatWave MySQL on OCI is now complete, you can end your tmux session.

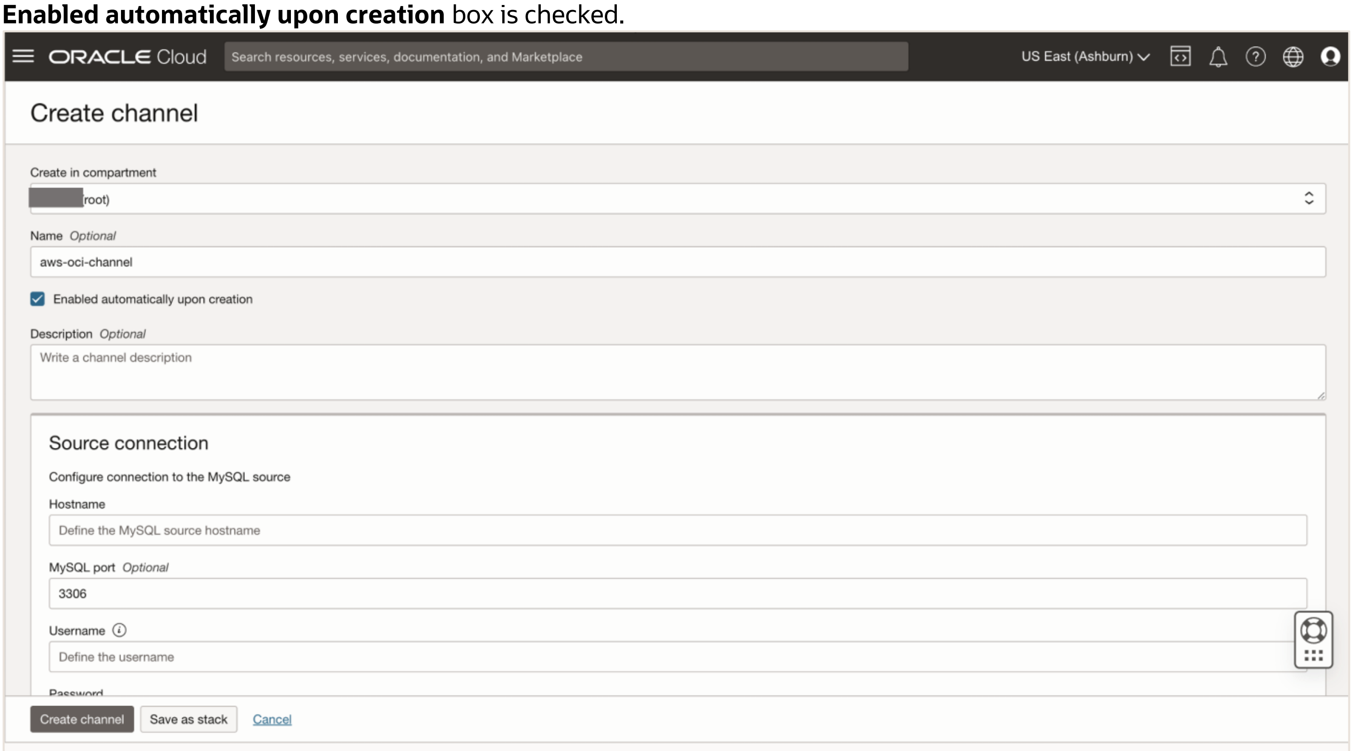
VII) On OCI, create a replication channel to set up replication from Amazon Aurora MySQL to HeatWave MySQL on OCI. During the channel creation process, if the Aurora instance is using binary log positioning - under the replication positioning section, select Source cannot use GTID auto-positioning and provide the binlogFile and binlogPosition values. If the Aurora instance is using GTIDs - select Source can use GTID auto-positioning (recommended). Create the replication channel afterwards.

106. After your data has successfully imported into HeatWave MySQL, from the OCI Console, click on the navigation menu again, go to **Databases**, and click **Channels**.

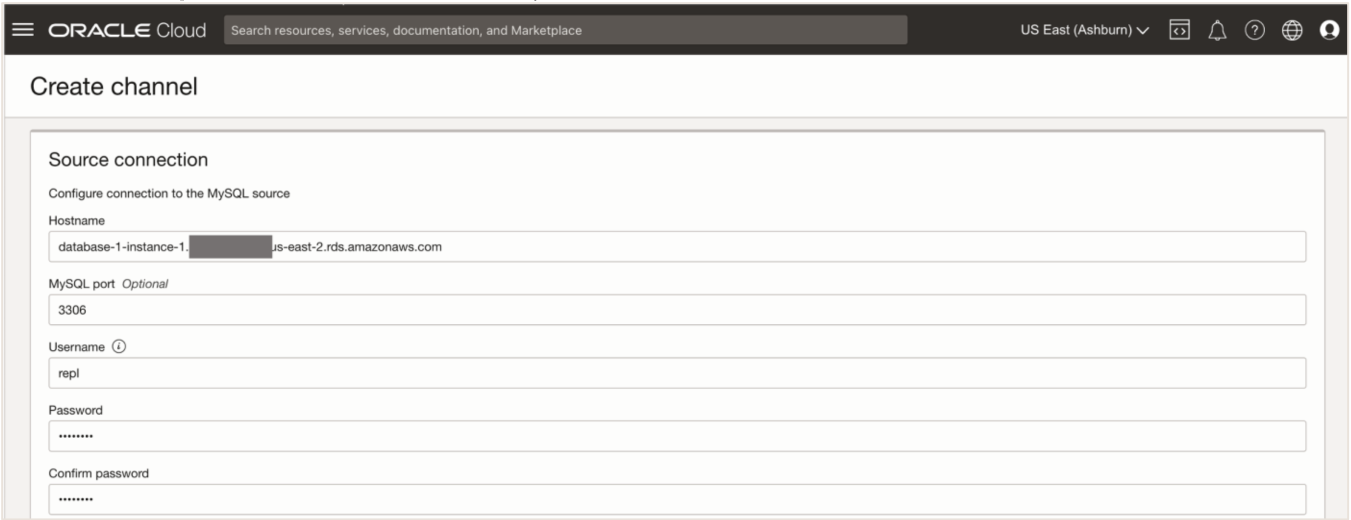
107. Click **Create channel** to set up replication between Aurora MySQL and HeatWave MySQL on OCI.



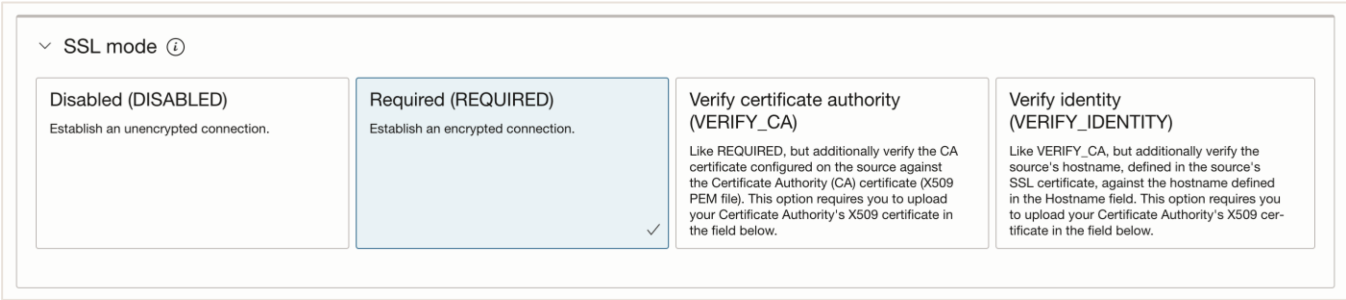
108. Ensure you are in the right compartment and enter a **replication channel name**. Ensure that the **Enabled automatically upon creation** box is checked.



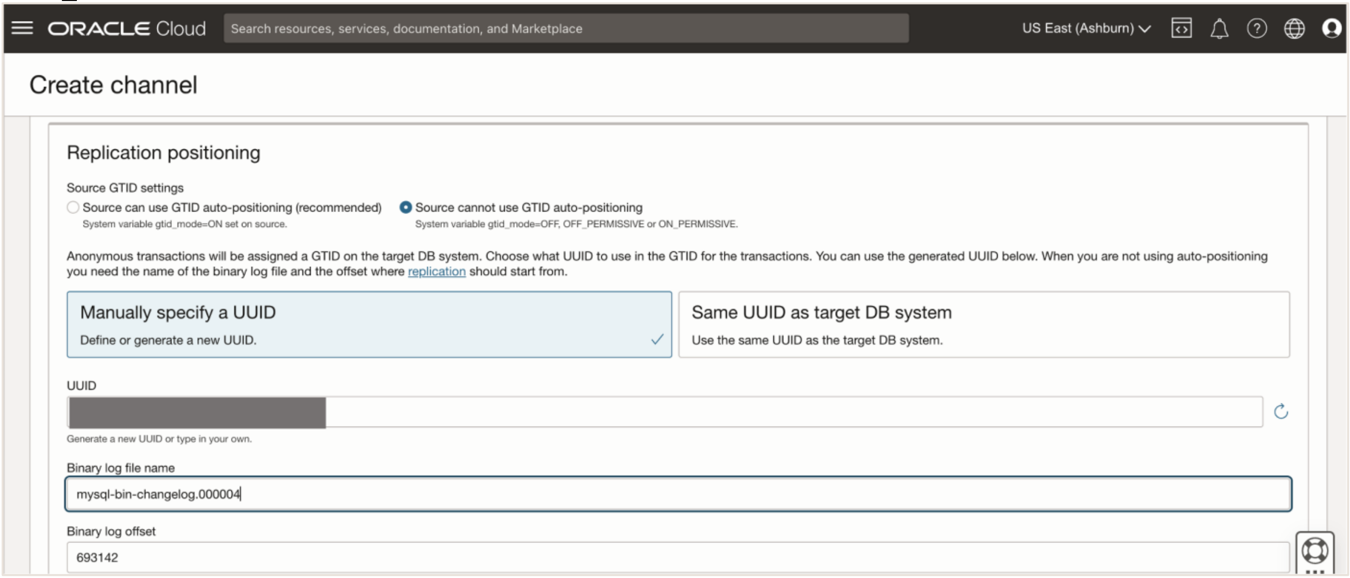
109. Under **Source connection**, for **Hostname** input your **Aurora Endpoint**. For **Port**, specify the port number the Aurora listens on - the default is **3306**. For **Username** and **Password** - specify the **replication username and password** for the account that you created on the Aurora instance.



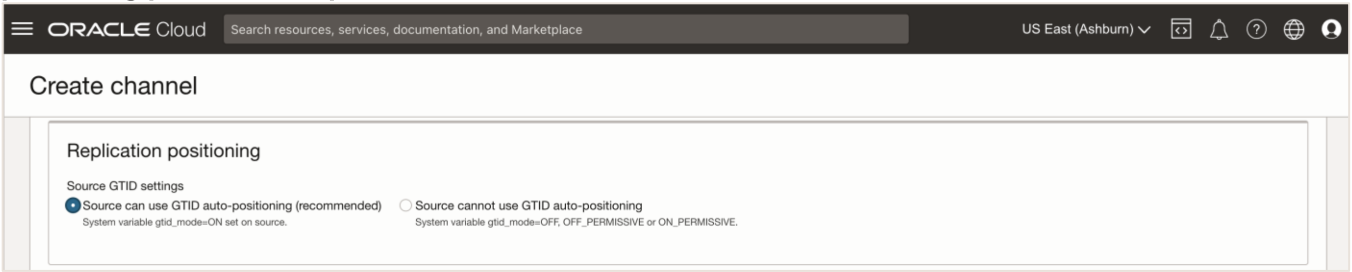
110. For **SSL mode** select the one that meets your need. For this guide, we have chosen **Required (REQUIRED)**.



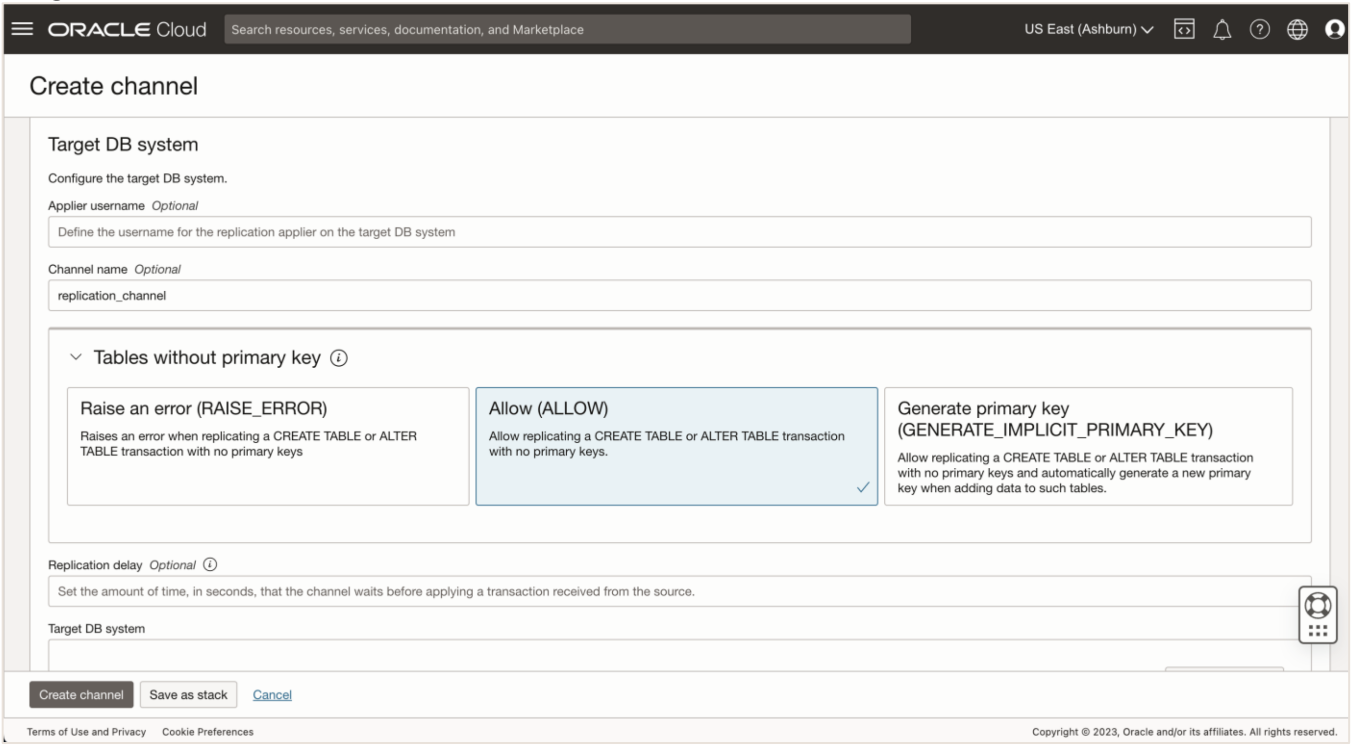
111. For **Replication positioning**, if your Aurora MySQL uses binary log positioning – select **Source cannot use GTID auto-positioning**. Keep the **UUID** field as-is, for **Binary log file name** and **Binary log offset**, input the `Binlog_file` and `Binlog_position` values respectively from the MySQL Shell's `Dump_metadata` that you had saved from step 104.



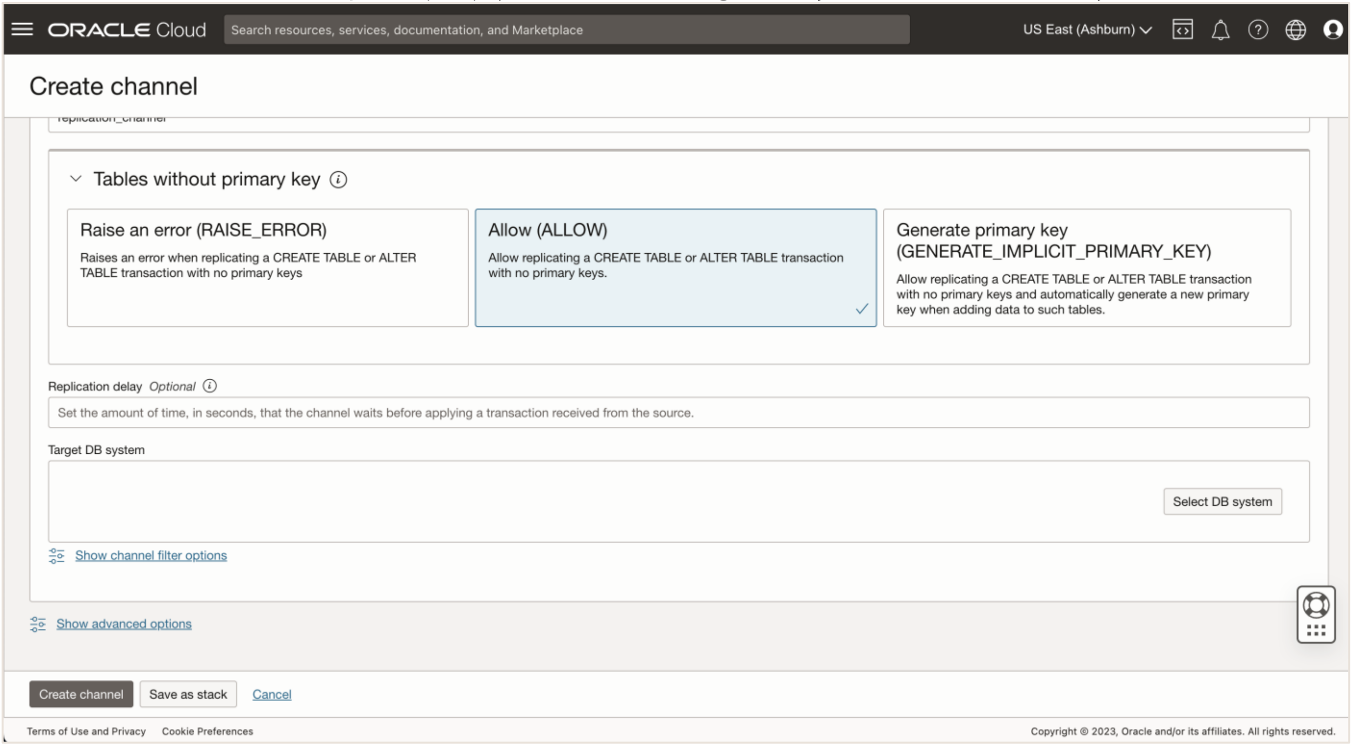
112. For **Replication positioning**, if your Aurora MySQL uses GTIDs – select **Source can use GTID auto-positioning (recommended)**.



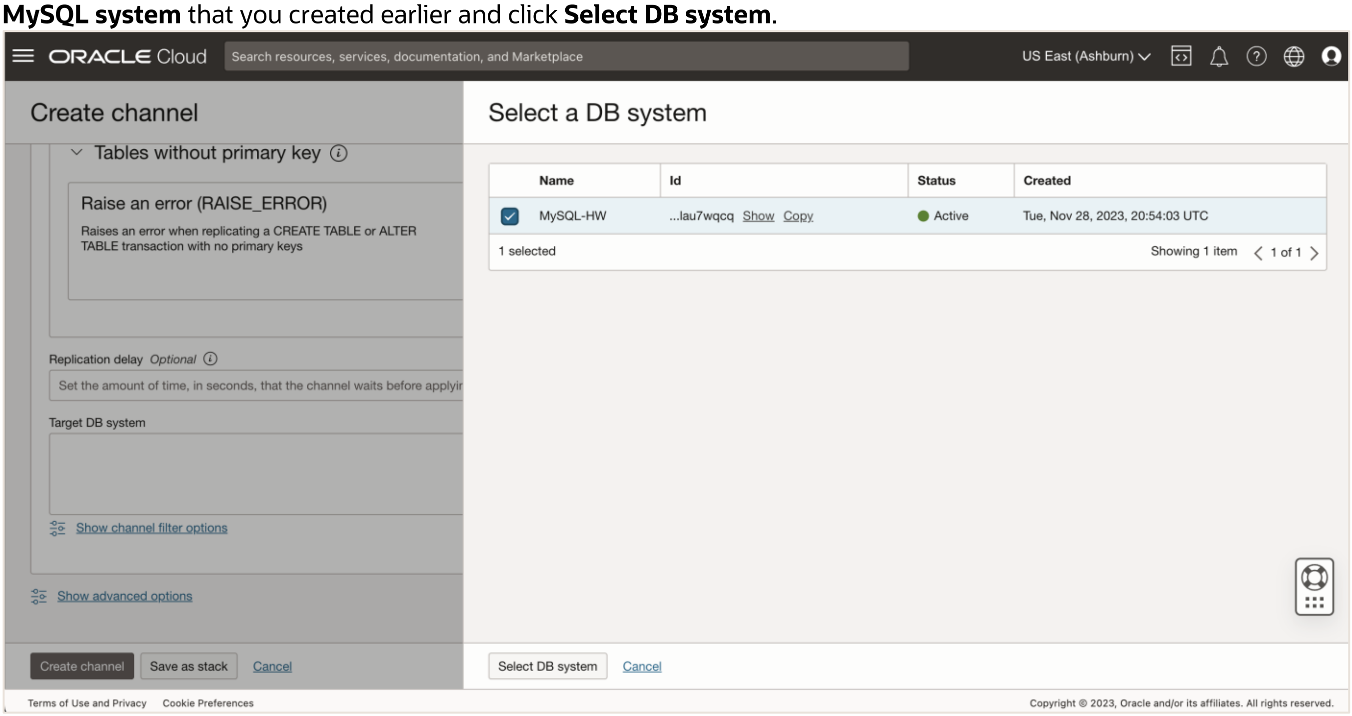
113. Scroll down until you see **Tables without primary key**. If you plan on using the High Availability or HeatWave option, select **Generate primary key** since these options require primary keys on every table. If you don't plan on using High Availability or HeatWave – you can either select **Raise an error** or **Allow**. For this guide, we have chosen **Allow**.



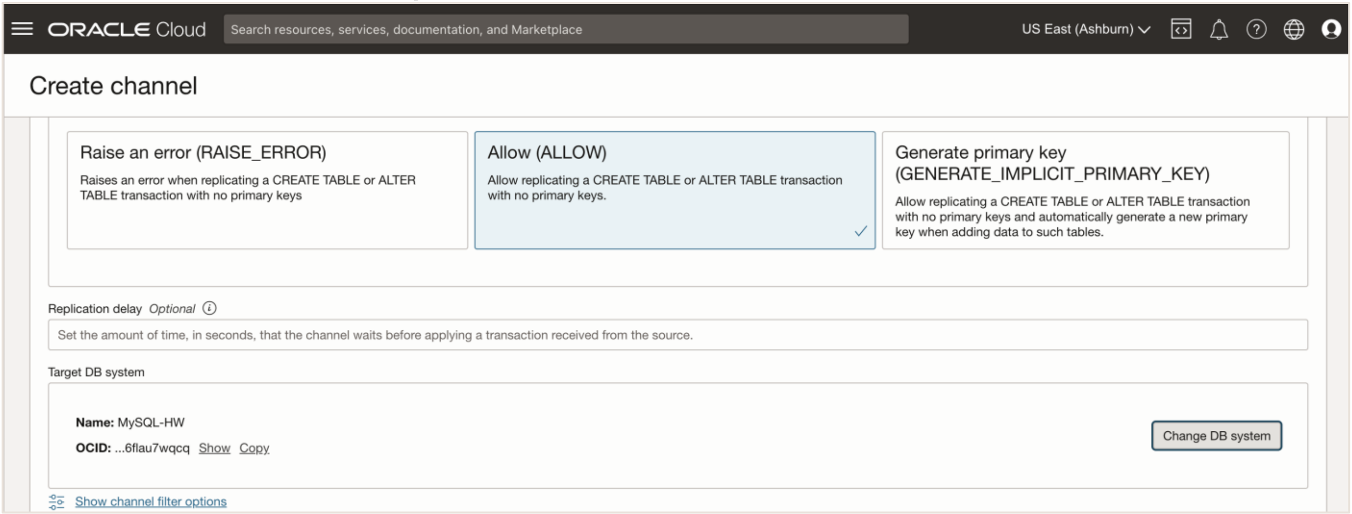
114. Under Tables without primary key, you should see **Target DB system**. Click **Select DB system**.



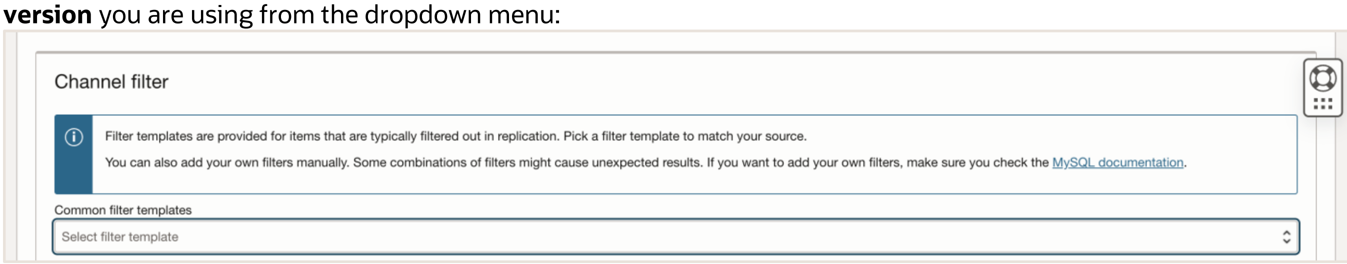
115. A list of your MySQL DB systems will open after completing the previous step. Select the **HeatWave MySQL system** that you created earlier and click **Select DB system**.



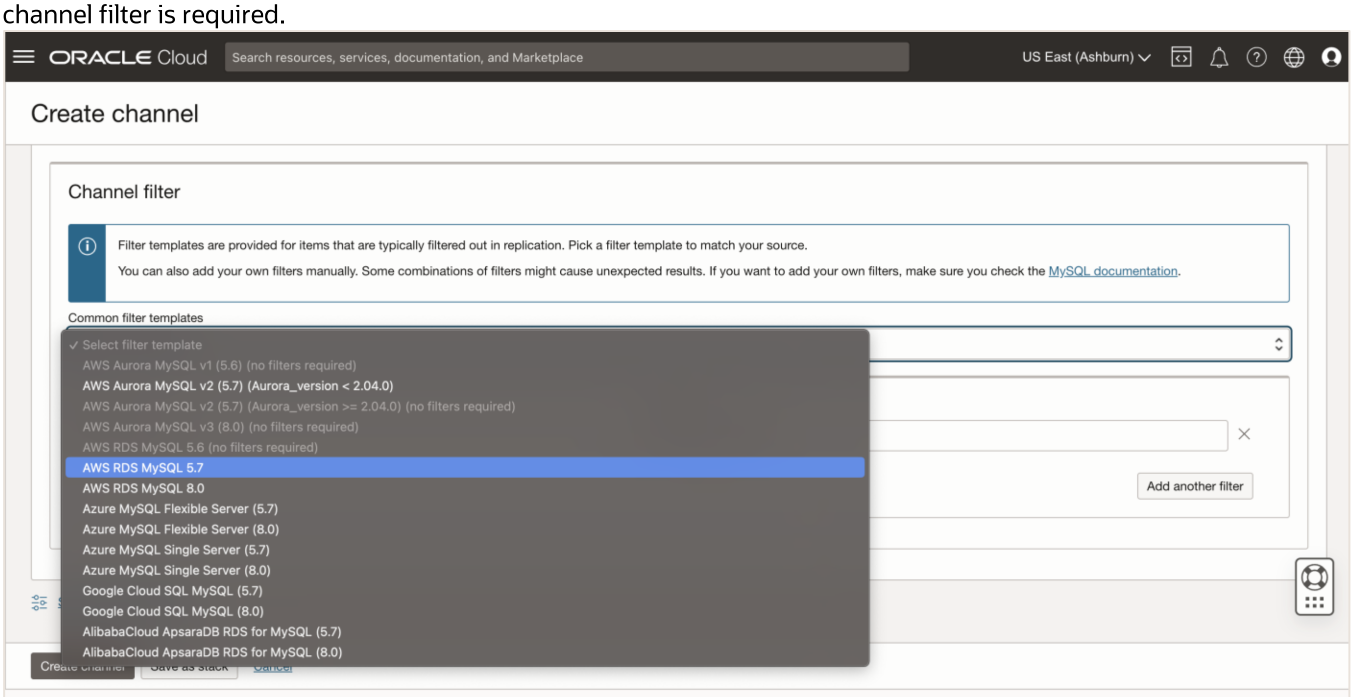
116. Click **Show channel filter options**.



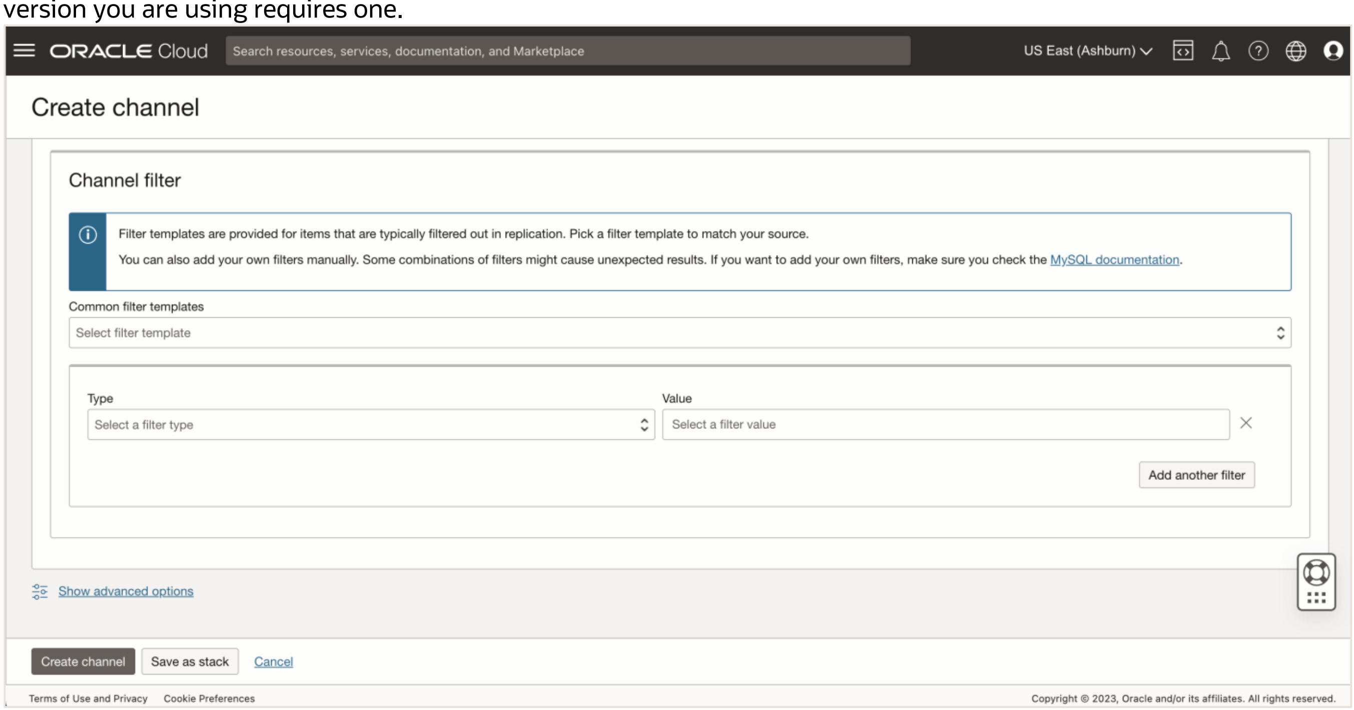
117. For **Channel filter**, under **Common filter templates** choose the appropriate **Aurora instance version** you are using from the dropdown menu:



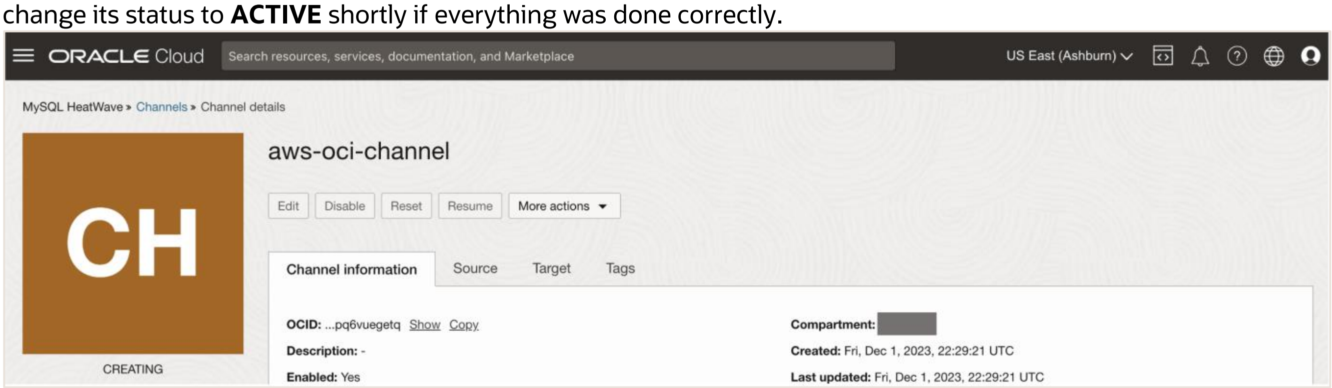
Note: for this step-by-step guide, we are using Aurora MySQL v5.7.12 (Aurora_version 2.11.2), thus no channel filter is required.

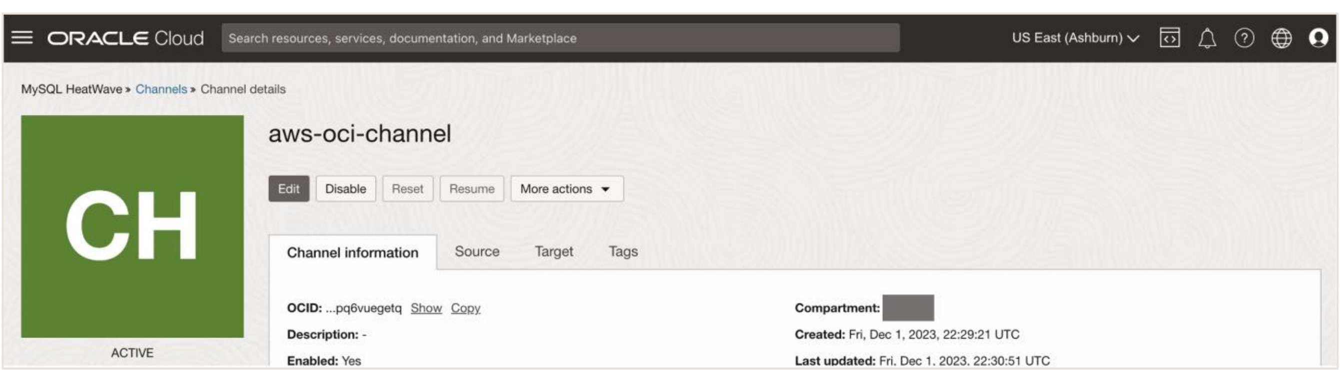


118. We need to provide the appropriate replication filter depending on the database and the database version that we are using. Since there are some tables in Aurora that will cause the replication to fail - hence we are filtering those tables out. Click **Create channel** after you have applied the channel filter – if the Aurora version you are using requires one.



119. The replication channel from your Aurora MySQL to HeatWave MySQL on OCI will now start CREATING so that we can propagate all the pending data changes to HeatWave MySQL that had occurred on the Aurora MySQL after the execution of MySQL Shell `util.copyInstance()` utility. Your channel should change its status to **ACTIVE** shortly if everything was done correctly.





VIII) After the replication channel is up, connect to HeatWave MySQL and execute the `SHOW REPLICA STATUS\G` command. From the query output, look for the `seconds_behind_source` and `Replica_SQL_Running_State` fields. If the `seconds_behind_source` field displays a value of `0` and the `Replica_SQL_Running_State` field displays a message of `Replica has read all relay log; waiting for more updates` - this indicates that the HeatWave MySQL instance has fully caught up with the Amazon Aurora MySQL changes and the replication channel can now be disabled.

Note: During this step, it is recommended to stop the database application for ~5 minutes to ensure that no writes are happening to the Aurora MySQL instance before the replication channel between HeatWave MySQL and Aurora MySQL is disabled.

120. Connect to your HeatWave MySQL on OCI instance using MySQL Shell which is installed on your EC2.

```
$ mysqlsh <user>@<hostname>:<port-number>
```

-OR-

```
$ mysqlsh -u <user> -p -h <hostname> -P <port-number>
```

```
[ec2-user@ip-[redacted] ~]$ mysqlsh admin@10.0.1.220
MySQL Shell 8.2.1

Copyright (c) 2016, 2023, Oracle and/or its affiliates.
Oracle is a registered trademark of Oracle Corporation and/or its affiliates.
Other names may be trademarks of their respective owners.

Type '\help' or '\?' for help; '\quit' to exit.
Creating a session to 'admin@10.0.1.220'
Fetching schema names for auto-completion... Press ^C to stop.
Your MySQL connection id is 5378 (X protocol)
Server version: 8.0.35-u1-cloud MySQL Enterprise - Cloud
No default schema selected; type \use <schema> to set one.
 MySQL  10.0.1.220:33060+ ssl  JS >
```

121. Switch to the SQL mode of MySQL Shell and run the below statement:

```
MySQL JS> \sql
MySQL SQL> SHOW REPLICA STATUS\G
```

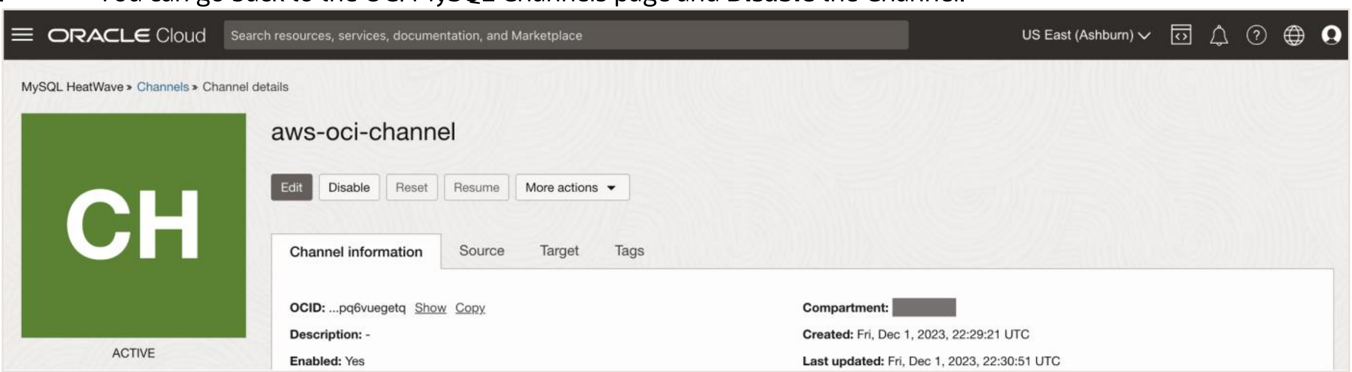
```
 MySQL  10.0.1.220:33060+ ssl  SQL > SHOW REPLICA STATUS\G
*************************** 1. row ***************************
             Replica_IO_State: Waiting for source to send event
                  Source_Host: database-1-instance-1.[redacted].us-east-2.rds.amazonaw
s.com
                  Source_User: repl
                  Source_Port: 3306
                Connect_Retry: 60
              Source_Log_File: mysql-bin-changelog.000004
          Read_Source_Log_Pos: 693142
               Relay_Log_File: relay-log-replication_channel.000002
                Relay_Log_Pos: 447
        Relay_Source_Log_File: mysql-bin-changelog.000004
           Replica_IO_Running: Yes
          Replica_SQL_Running: Yes
```

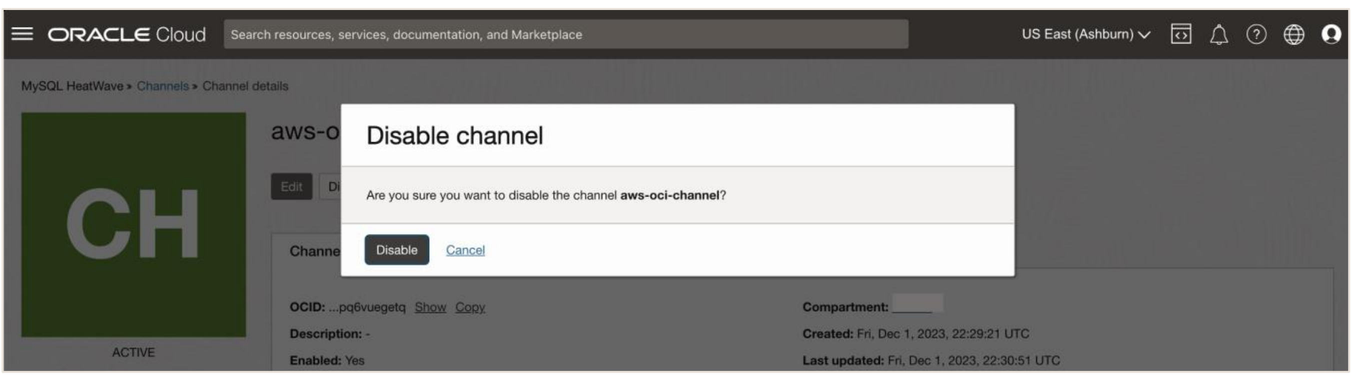
122. If the replication is successfully ongoing from Aurora MySQL to HeatWave MySQL, you should see the status of `Replica_IO_Running` and `Replica_SQL_Running` as `Yes`. If one or the other shows an output different than `Yes`, your replication has failed or encountered an error.

123. When executing the above SQL statement `SHOW REPLICA STATUS\G`, also look for `Seconds_Behind_Source` and `Replica_SQL_Running_State` fields. If you see a value of `0` for `Seconds_Behind_Source` and a value string of `Replica has read all relay log; waiting for more updates` for `Replica_SQL_Running_State` - this suggests that HeatWave MySQL instance has fully caught up with the Aurora MySQL instance and there are no pending transactions/changes on Aurora MySQL that needs to be replicated to HeatWave MySQL.

```
         Seconds_Behind_Source: 0
Source_SSL_Verify_Server_Cert: No
                Last_IO_Errno: 0
                Last_IO_Error:
               Last_SQL_Errno: 0
               Last_SQL_Error:
  Replicate_Ignore_Server_Ids:
             Source_Server_Id: 1
                  Source_UUID: [redacted]
             Source_Info_File: mysql.slave_master_info
                    SQL_Delay: 0
          SQL_Remaining_Delay: NULL
    Replica_SQL_Running_State: Replica has read all relay log; waiting for more updates
```

124. You can go back to the OCI MySQL Channels page and **Disable** the Channel.



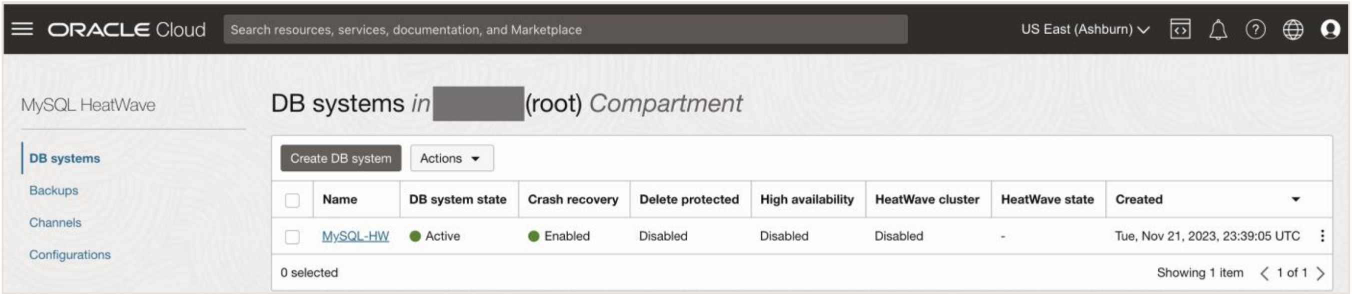


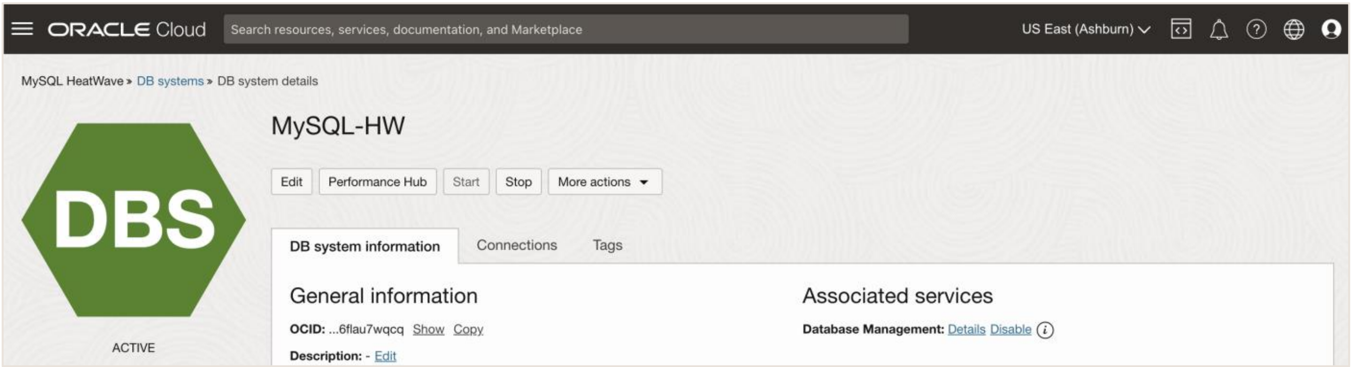
125. Once the channel is disabled, you may enable HA for your HeatWave MySQL instance.

IX) At this point, the live migration process for the database is complete. The database applications can now point to HeatWave MySQL on OCI.

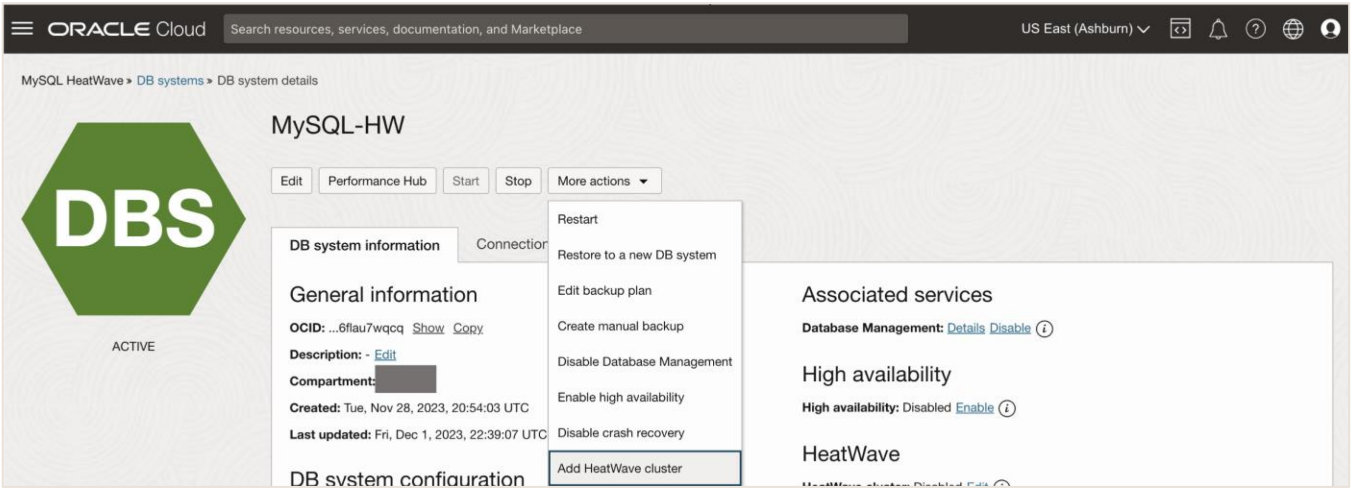
X) (Optional) On OCI, if the HeatWave option was enabled during HeatWave MySQL DB creation, add the HW Cluster and load data from MySQL InnoDB storage into the HW Cluster using automation.

126. Login to OCI. Click on the navigation menu, go to **Databases**, and click **HeatWave MySQL**.
127. Click on the name of your HeatWave MySQL instance to go to the **DB System Details** page.

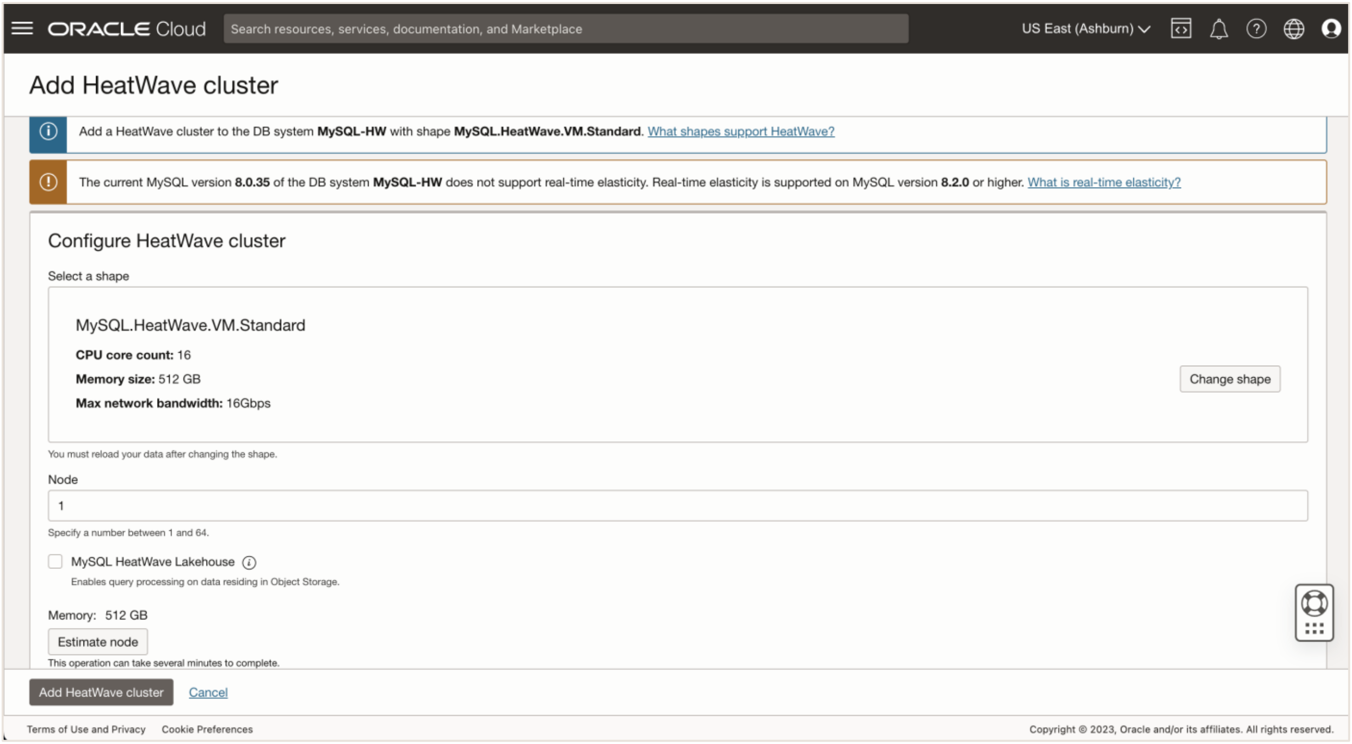




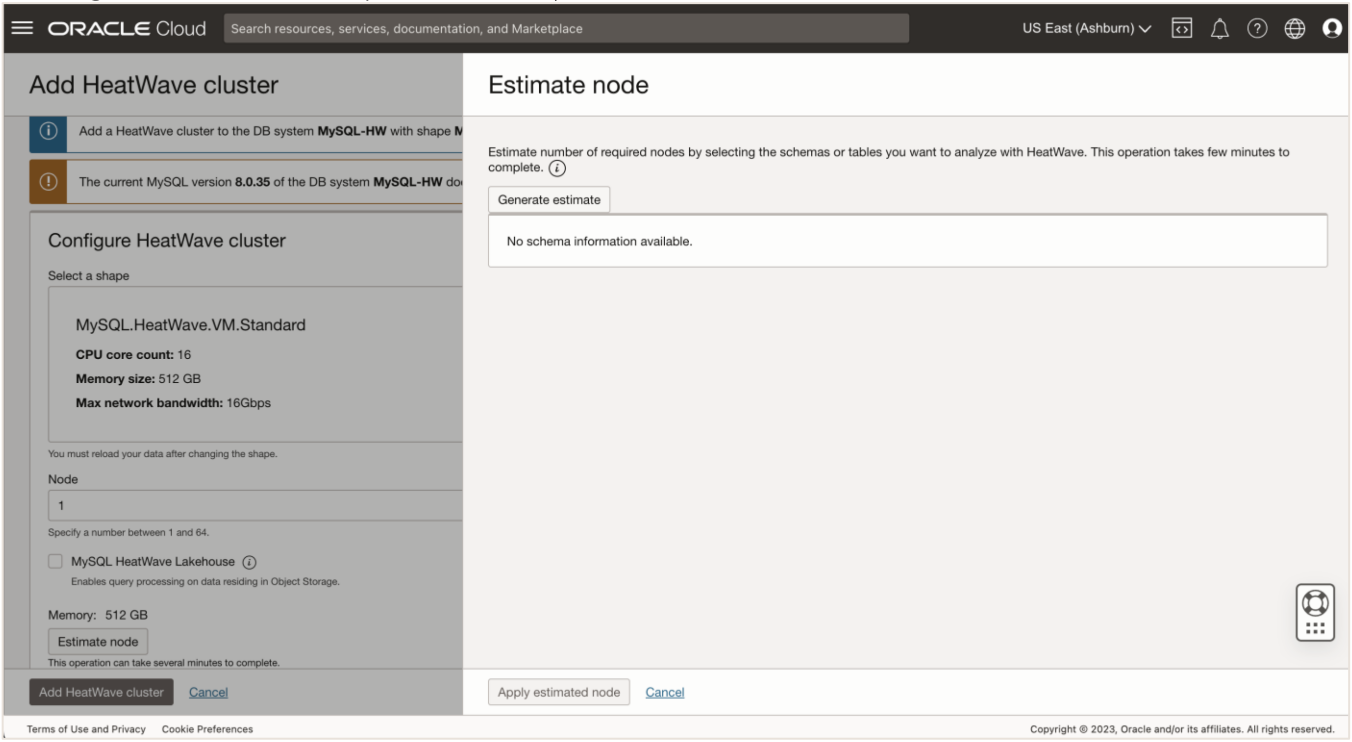
128. Click **More actions** and click **Add HeatWave cluster**.



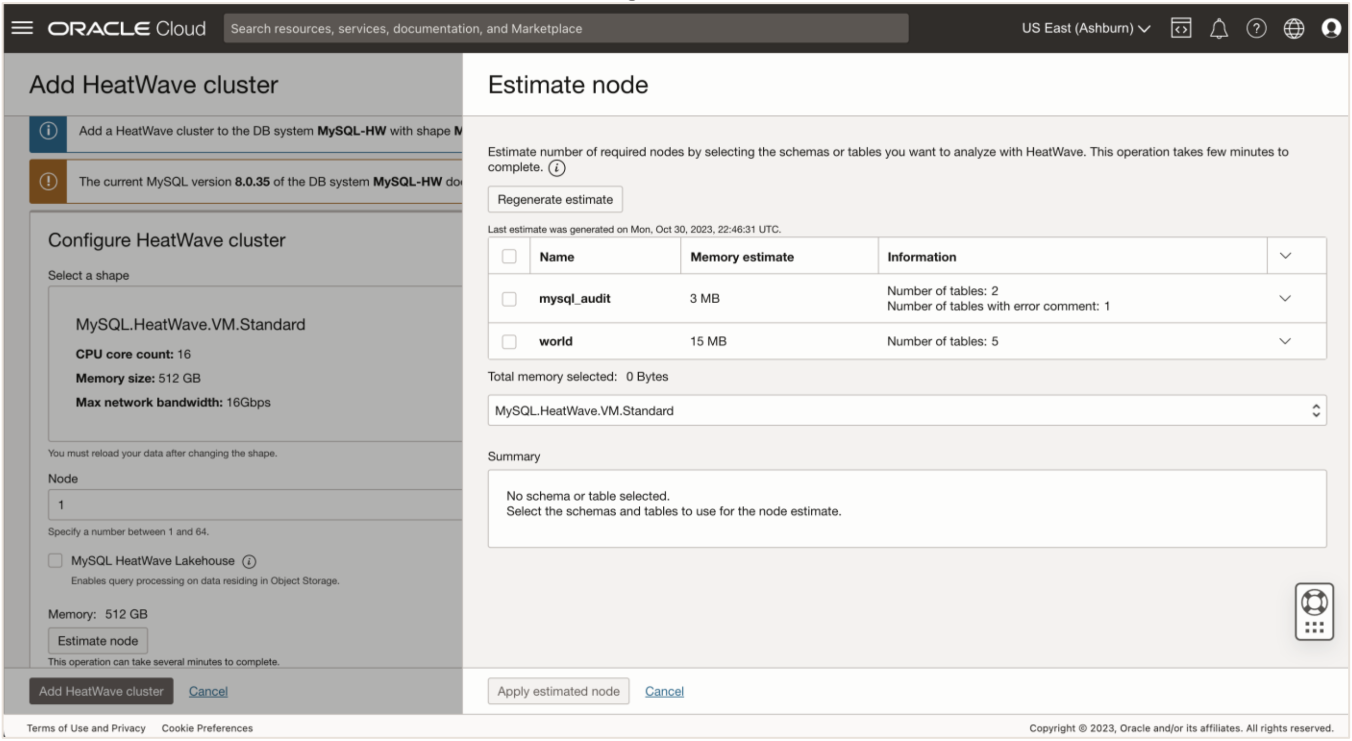
129. Click **Estimate node**.



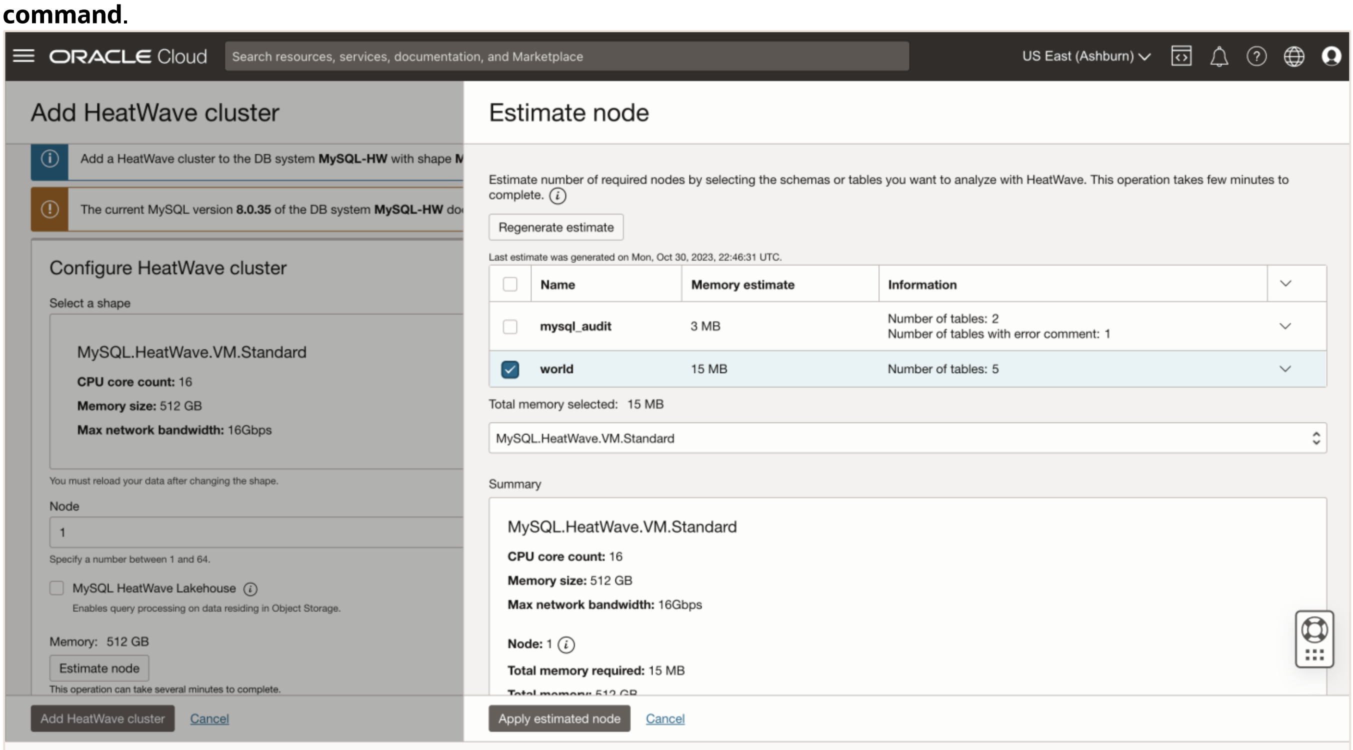
130. Click **Generate estimate**. This step will estimate the number of HeatWave nodes required by selecting the schemas or tables you want to analyze with HeatWave.



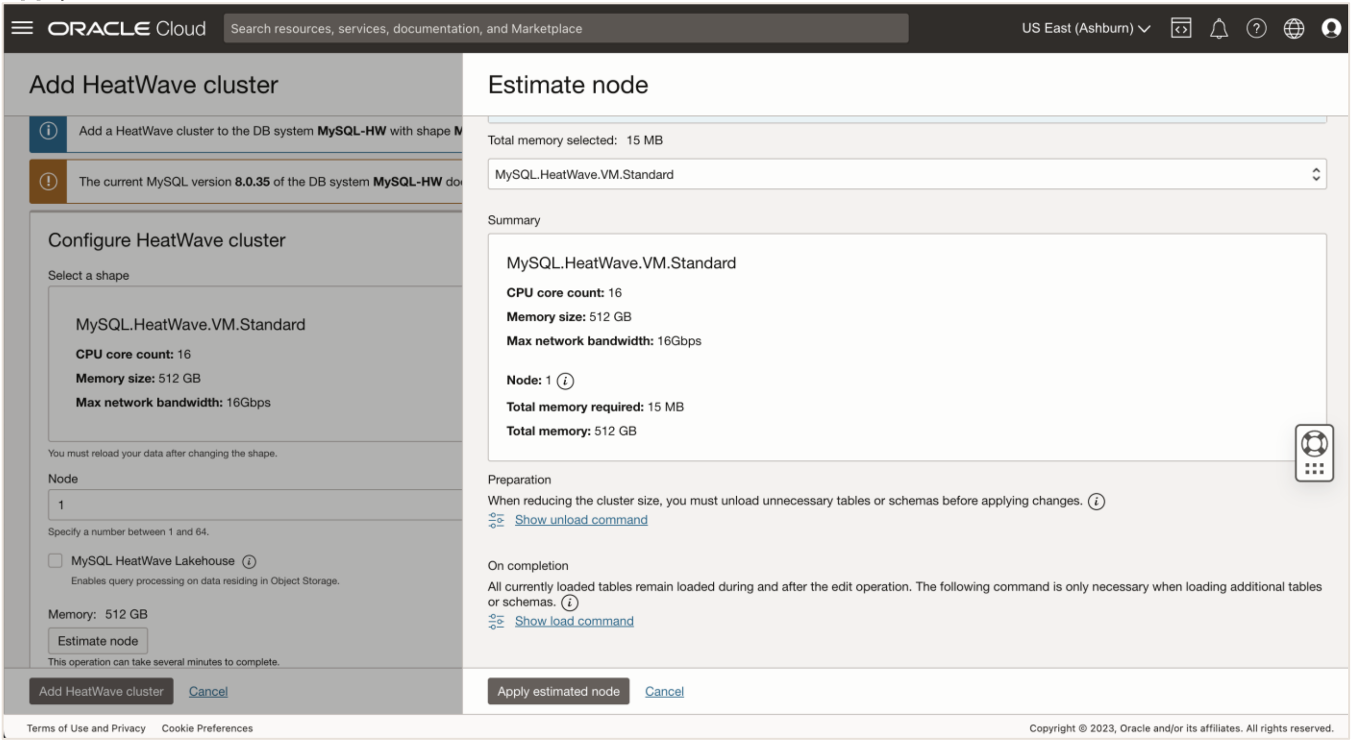
131. Within a few minutes, the list of your schemas that are in the MySQL InnoDB storage engine will be listed. **Check the box** next to the schema or table name that you wish to load in HeatWave for query acceleration and to run OLAP and ML workloads - alongside OLTP.

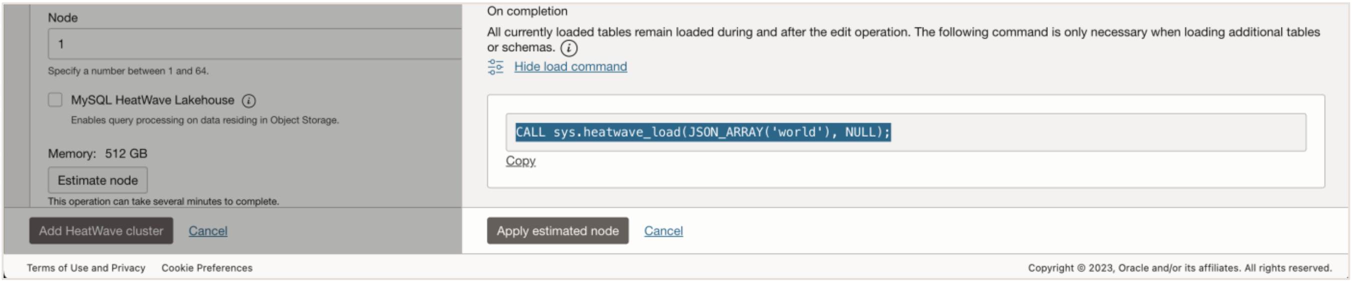


132. After selecting the schemas or tables, scroll down on that page until you see the **Show load command**.

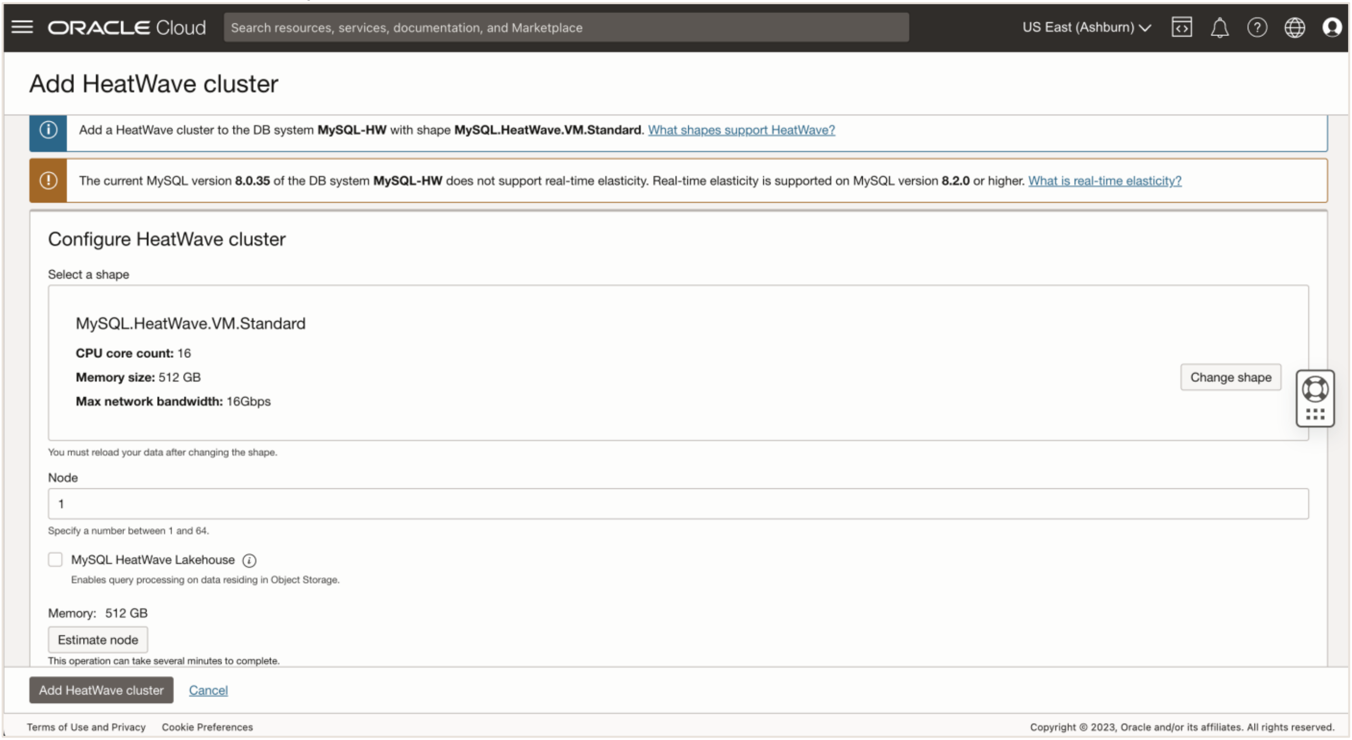


133. Click **Show load command**, copy the `CALL sys.heatwave_load` command, and save it. Click **Apply estimated node**.

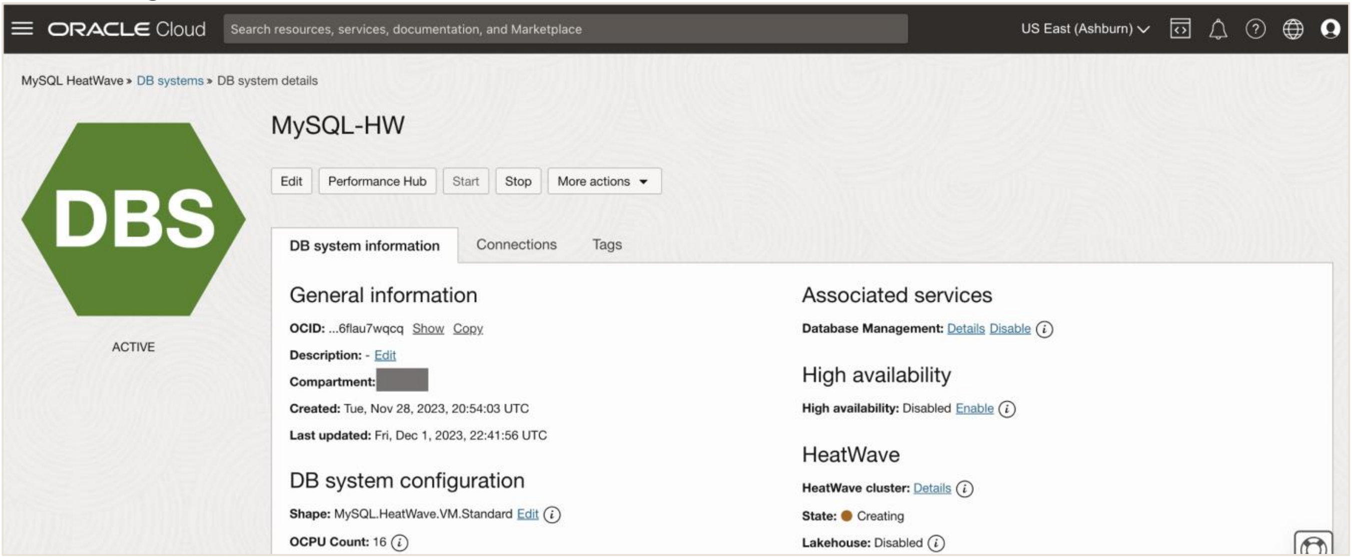


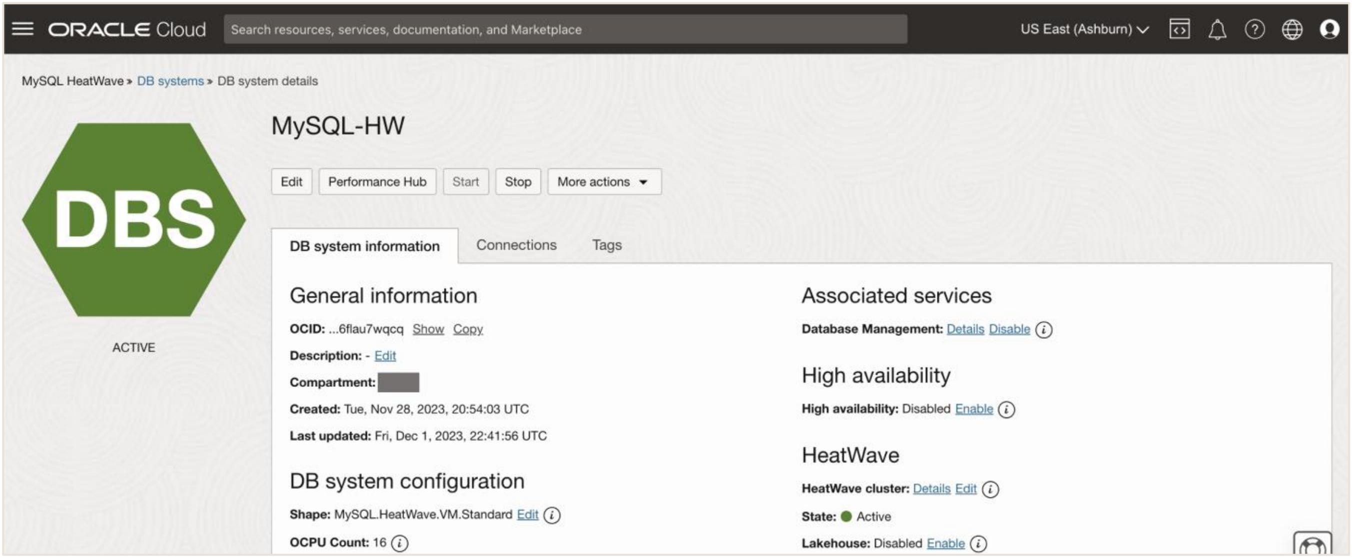


134. Executing the previous step will change the HeatWave node count depending on the data you have selected to load into the HeatWave in-memory engine. Click **Add HeatWave cluster** to finish adding the HeatWave cluster creation process.



135. The HeatWave cluster will be ready within a few minutes. You should see the HeatWave state change from Creating to **Active**.



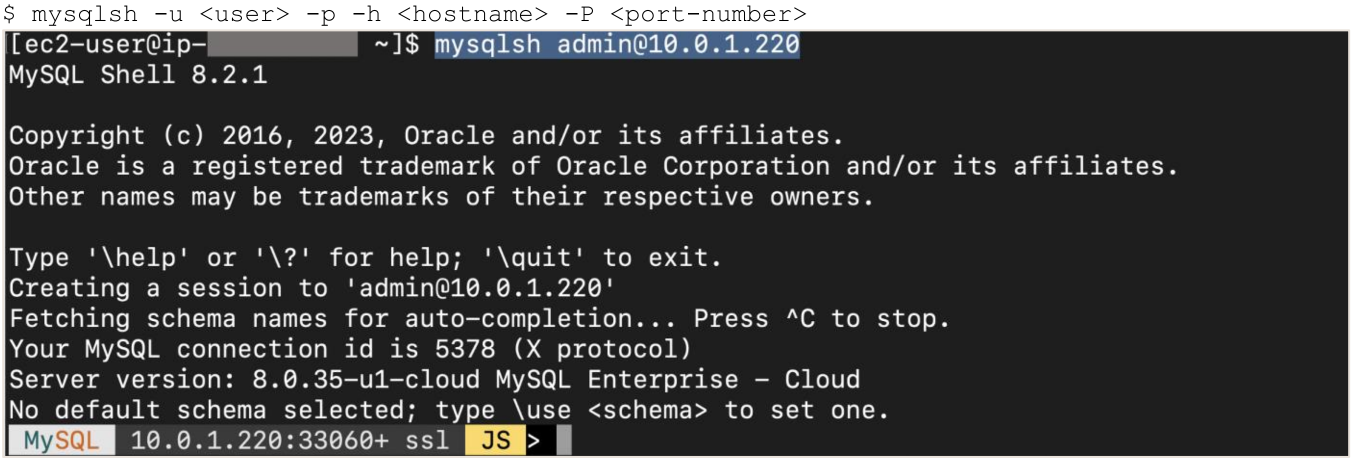


136. Connect to your HeatWave MySQL on OCI instance using MySQL Shell which is installed on your EC2 instance.

```
$ mysqlsh <user>@<hostname>:<port-number>
```

-OR-

```
$ mysqlsh -u <user> -p -h <hostname> -P <port-number>
```



137. Switch to the SQL mode of MySQL Shell and execute the Load command that we had copied earlier to load data into HeatWave from the MySQL InnoDB storage engine.

```
MySQL JS> \sql
MySQL SQL> CALL sys.heatwave_load(JSON_ARRAY('world'), NULL);
```

Note: replace the `sys.heatwave_load` command with what you have.

```
MySQL  10.0.1.220:33060+ ssl  SQL > CALL sys.heatwave_load(JSON_ARRAY('world'), NULL);
+------------------------------------------+
| INITIALIZING HEATWAVE AUTO PARALLEL LOAD |
+------------------------------------------+
| Version: 2.20                            |
|                                          |
| Load Mode: normal                        |
| Load Policy: disable_unsupported_columns |
| Output Mode: normal                      |
|                                          |
+------------------------------------------+
6 rows in set (1.2705 sec)

+--------------------------------------------------------------------------+
| OFFLOAD ANALYSIS                                                         |
+--------------------------------------------------------------------------+
| Verifying input schemas: 1                                               |
| User excluded items: 0                                                   |
|                                                                          |
| SCHEMA                      OFFLOADABLE    OFFLOADABLE     SUMMARY OF   |
| NAME                             TABLES        COLUMNS     ISSUES       |
| ------                      -----------    -----------     ----------   |
| `world`                               3             24                  |
```

[…output truncated]

```
+---------------------------------------------------------------------------------+
| LOAD SUMMARY                                                                    |
+---------------------------------------------------------------------------------+
|                                                                                 |
| SCHEMA                         TABLES       TABLES      COLUMNS         LOAD |
| NAME                           LOADED       FAILED       LOADED     DURATION |
| ------                         ------       ------       -------    -------- |
| `world`                             3            0           24       1.13 s |
|                                                                                 |
+---------------------------------------------------------------------------------+
6 rows in set (1.2705 sec)

Query OK, 0 rows affected (1.2705 sec)
MySQL  10.0.1.220:33060+ ssl  SQL >
```

138. You now have a complete HeatWave MySQL cluster.

To learn more about using HeatWave, please visit our documentation.

Connect with us

Call **+1.800.ORACLE1** or visit **oracle.com**. Outside North America, find your local office at: **oracle.com/contact**.

blogs.oracle.com   facebook.com/oracle   twitter.com/oracle