Amazon Aurora Connectivity

This document describes how to configure access to Amazon Aurora instances from on-premises equipment and Amazon EC2-Classic instances. It also describes the recommended configuration which hides the Aurora instances from the Internet.

Getting Started

Establishing connectivity to Aurora instances is mostly about configuring an Amazon Virtual Private Cloud (VPC) to suit your requirements. All Aurora database instances reside in a VPC associated with your AWS account. If you already have a VPC and want to create Aurora instances in it, these instructions will provide examples of how to configure your VPC to allow access to your Aurora instances. If you don't have a VPC or want to create a new VPC for your Aurora instances, the Amazon RDS launch wizard will create and configure a VPC for you.

Choosing Public vs. Private Accessibility

One of the choices you'll make when creating an Aurora instance is whether or not the instance will be publicly accessible. If you will be accessing your Aurora instances exclusively from EC2 instances or devices in the same VPC as the Aurora instances, answer **No** in the **Publicly Accessible** field in the Amazon RDS launch wizard (see the example following). When the Aurora instance is created, it will have a private IP address, but no public (Internet routable) IP address.

If you plan to access your Aurora instances from outside the VPC, such as from your on-premises equipment or from Amazon EC2 instances in other AWS regions, answering **Yes** in the **Publicly Accessible** field in the Amazon RDS launch wizard will provide the Aurora instance with a public (Internet routable) IP address and a private (non–Internet routable) one. Note that there might be additional steps required to configure your VPC to allow access to the Aurora instance from outside the VPC, such as configuring VPC route tables, network access control lists (ACLs), and security groups. Examples are provided in the sections following.

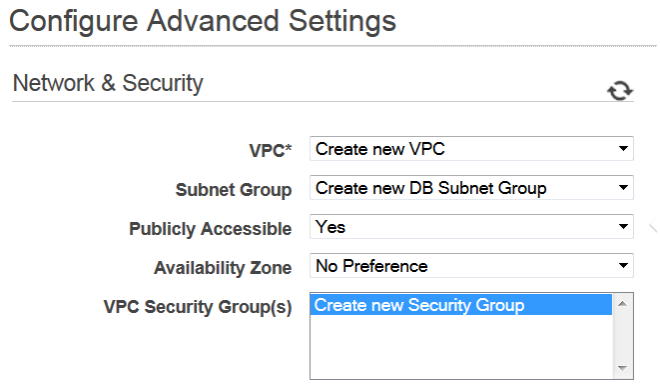
Using ClassicLink

If you plan to access your Aurora instances from EC2 instances residing in the same region, but not in a VPC (an approach commonly known as EC2-Classic), you can enable ClassicLink on the VPC where your Aurora instances reside. Enabling ClassicLink allows your EC2-Classic instances to communicate with each of your Aurora instances using its private IP address. Doing so allows you to take advantage of the higher throughput and lower latency connectivity available for interinstance communication within AWS and avoid network bandwidth charges associated with communicating over the Internet. This approach also can improve security.

Starting from Scratch (No Existing VPC or Aurora Instances)

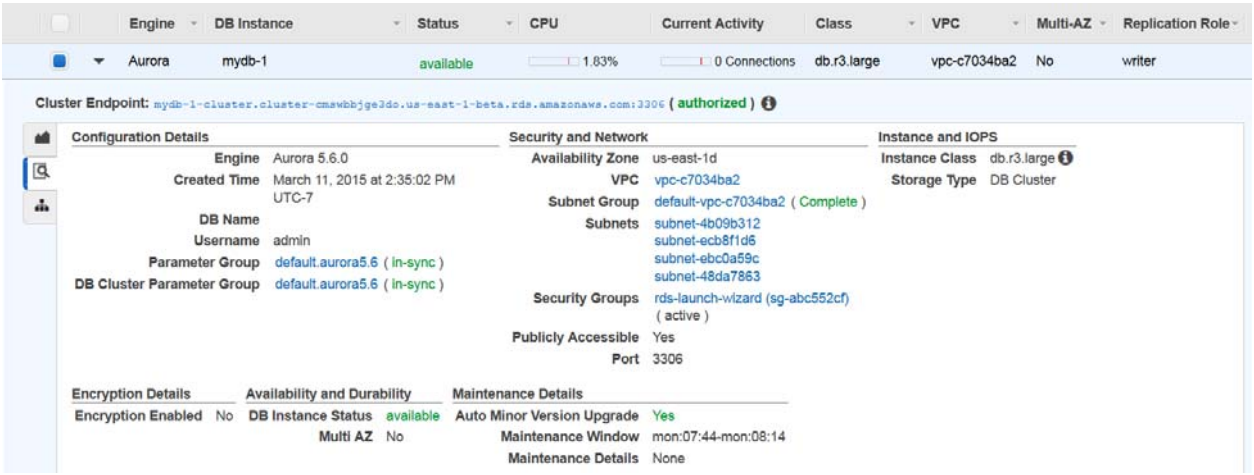
The simplest way to create a VPC for your Aurora instance is to let the Amazon RDS launch wizard do it for you. It will create and configure the VPC and create a new Aurora instance in it. The figure following shows the Amazon RDS launch wizard page where you can create a new VPC and make an Aurora instance publicly accessible.

amazon web services



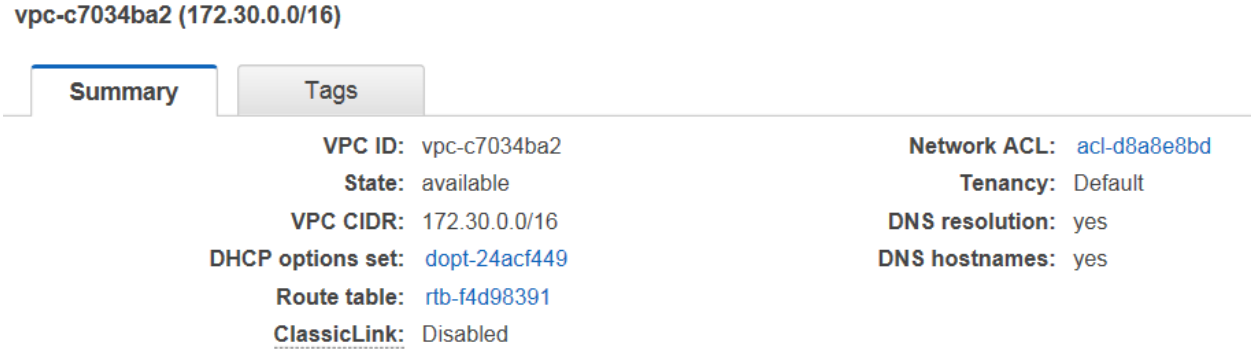
Aurora Instance

Here's the Aurora instance that the Amazon RDS launch wizard created for us. Notice that it is publicly accessible and that it has been assigned to a security group and four subnet groups—one for each Availability Zone in the current AWS region.



VPC

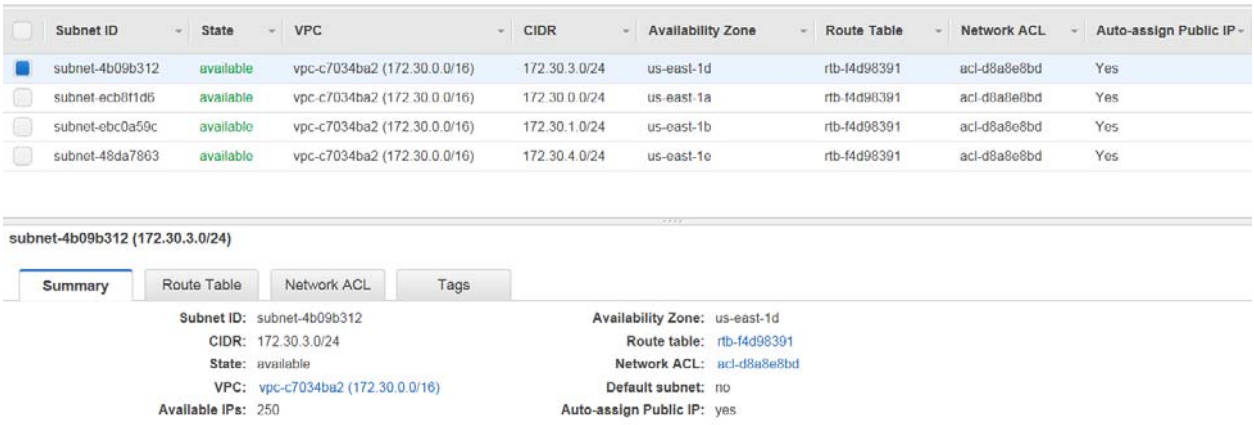
Let's take a look at the VPC that the Amazon RDS launch wizard created for us. Notice that the **DNS resolution** and **DNS hostnames** VPC attributes have been enabled because we specified that the Aurora instance will be publicly accessible.



Subnets

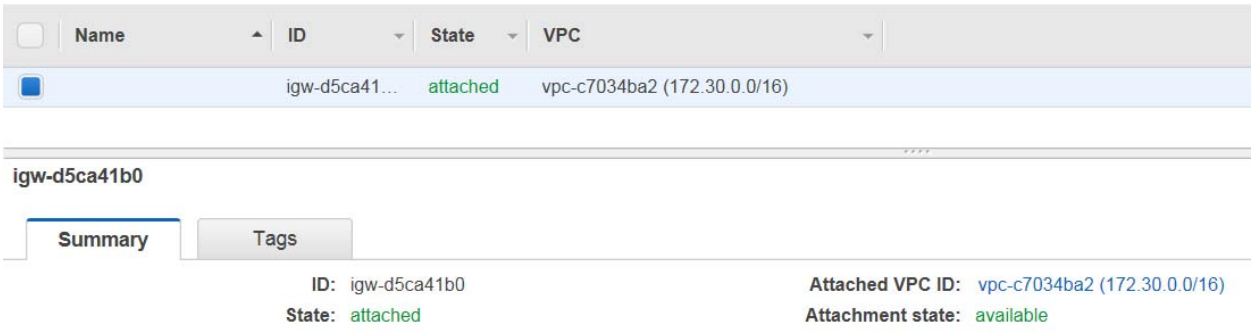
The Amazon RDS launch wizard has created a subnet in our VPC for each Availability Zone (AZ) in the current AWS region. Although an Aurora instance resides in a single AZ at any given time, it's necessary to have a subnet in at least three AZs to achieve high availability. If the AZ containing your Aurora instance becomes unavailable, RDS will automatically provision a new instance in an available AZ that has a VPC subnet. If there is no VPC subnet for an AZ, RDS won't provision Aurora instances in it.

An Aurora database can have more than one instance. The set of instances that belong to the same Aurora database is called an Aurora cluster. An Aurora cluster can have one writer node and multiple reader nodes. Another reason for defining subnets for multiple AZs is to allow Aurora instances in the same Aurora cluster to reside in different AZs. Each of those instances can reside in any AZ in the same region, as long as VPC subnets are defined for the AZ. This functionality provides you with options for load balancing database access across all available AZs in the region and for limiting the impact to your business if one of the AZs becomes temporarily unavailable. The following figure shows subnets in the launch wizard.



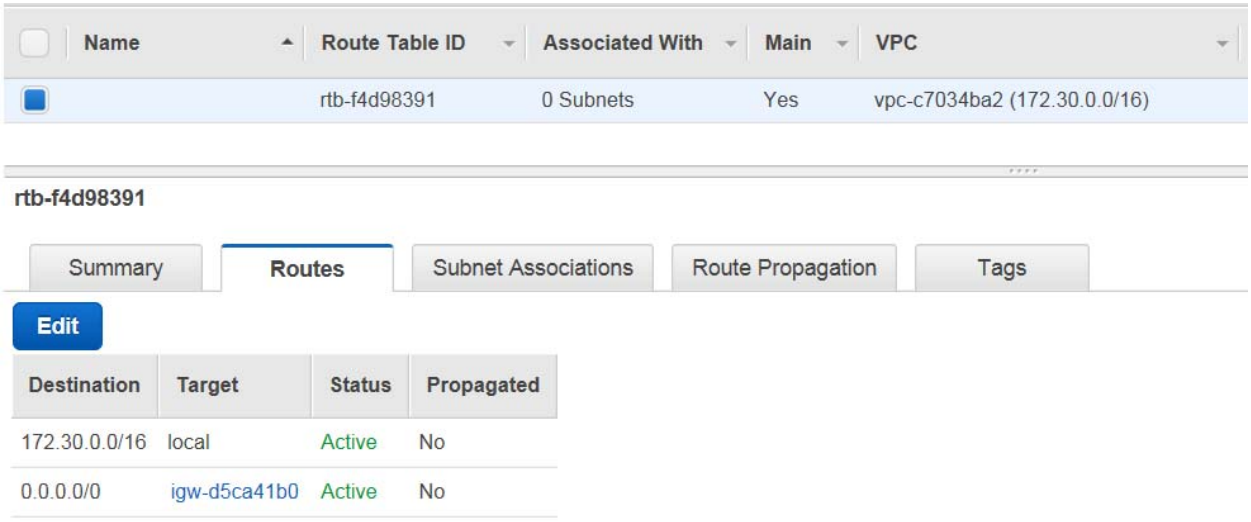
Internet Gateway

Because we specified that we wanted the Aurora instance to be publicly accessible, the Amazon RDS launch wizard provisioned an Internet gateway for our VPC, as shown following. An Amazon VPC Internet gateway is horizontally scaled, redundant, and highly available and imposes no bandwidth constraints.



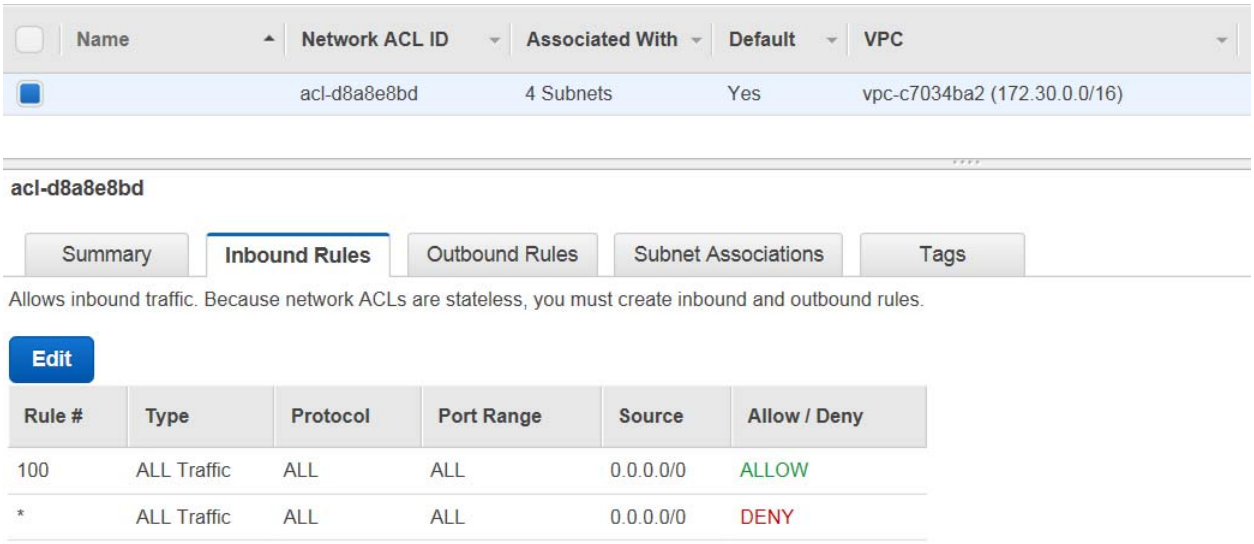
Route Table

The Amazon RDS launch wizard also created a route table in our VPC and configured it to route nonlocal network traffic to the Internet gateway, as shown following.



Network ACL

An Amazon VPC network ACL allows you to specify what traffic is allowed to enter and exit a subnet and disallowed from entering and exiting a subnet. Here, the Amazon RDS launch wizard has created a network ACL and associated it with all of the subnets in the VPC. The default is to allow all inbound and outbound traffic, but you can modify the rules to suit your requirements.



Name	Network ACL ID	Associated With	Default	VPC
	acl-d8a8e8bd	4 Subnets	Yes	vpc-c7034ba2 (172.30.0.0/16)

acl-d8a8e8bd

Summary | Inbound Rules | Outbound Rules | Subnet Associations | Tags

Allows inbound traffic. Because network ACLs are stateless, you must create inbound and outbound rules.

Edit

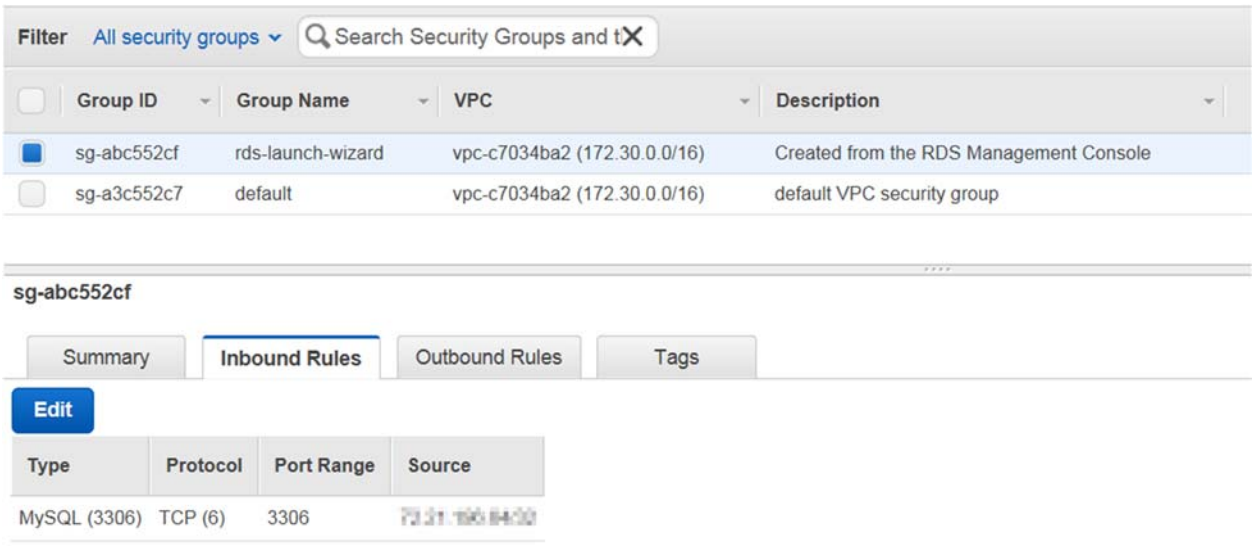
Rule #	Type	Protocol	Port Range	Source	Allow / Deny
100	ALL Traffic	ALL	ALL	0.0.0.0/0	ALLOW
*	ALL Traffic	ALL	ALL	0.0.0.0/0	DENY

Security Groups

Amazon VPC security groups specify which network traffic is allowed and disallowed at the instance level. The Amazon RDS launch wizard creates a security group that allows incoming traffic on the MySQL port (3306) for traffic originating from the system you accessed the AWS Management Console from, as shown following. If you specified a port other than 3306 for the Aurora instance, that port will be used instead. Note that VPC security groups are managed from the Amazon VPC console rather than the Amazon RDS console.

If you plan to access your Aurora instances from devices with different IP addresses, you will need to add rules to the security group to allow the inbound traffic.

If you're accessing Aurora from a corporate network, you might need to create inbound rules for each of the IP address ranges used by your corporate network for Internet traffic. Engage your corporate network support team to determine which IP address ranges you should use.

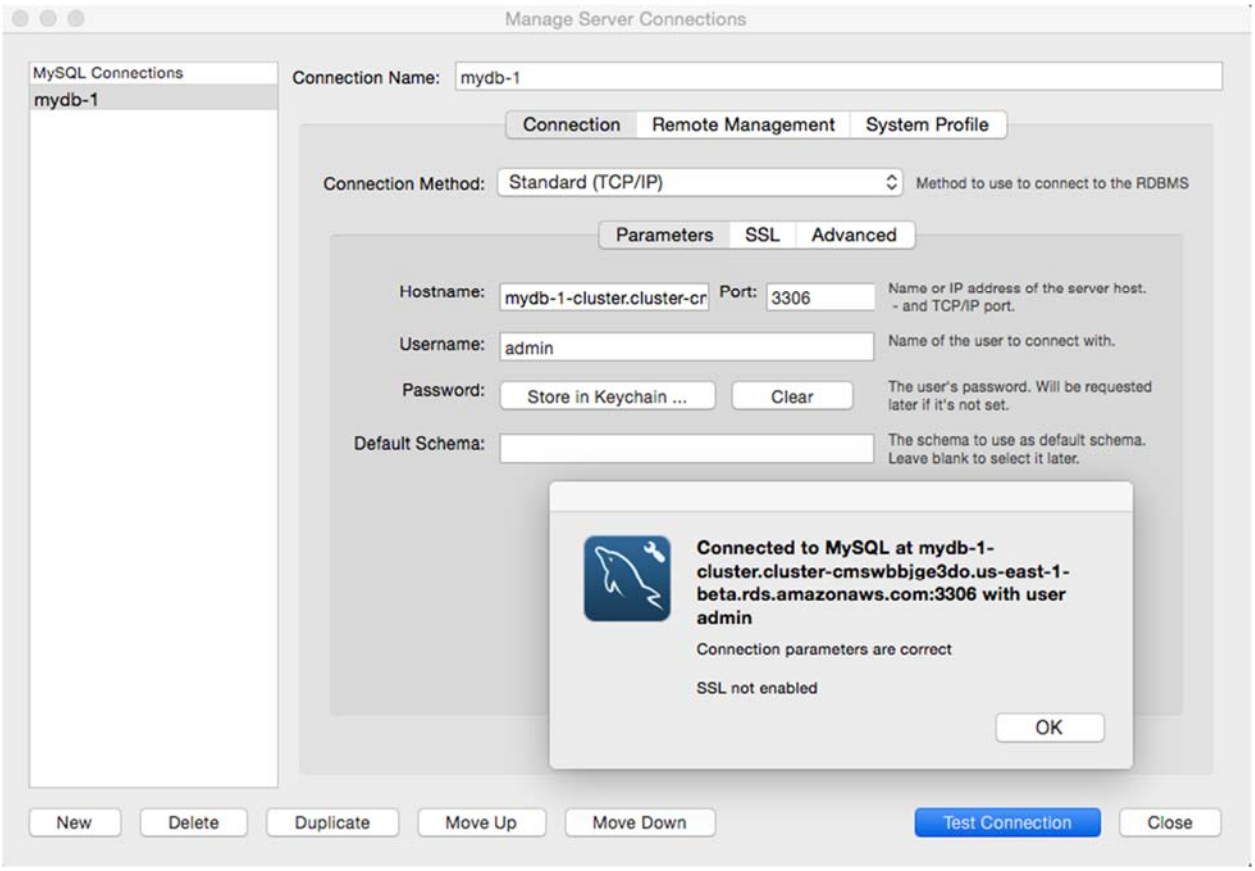


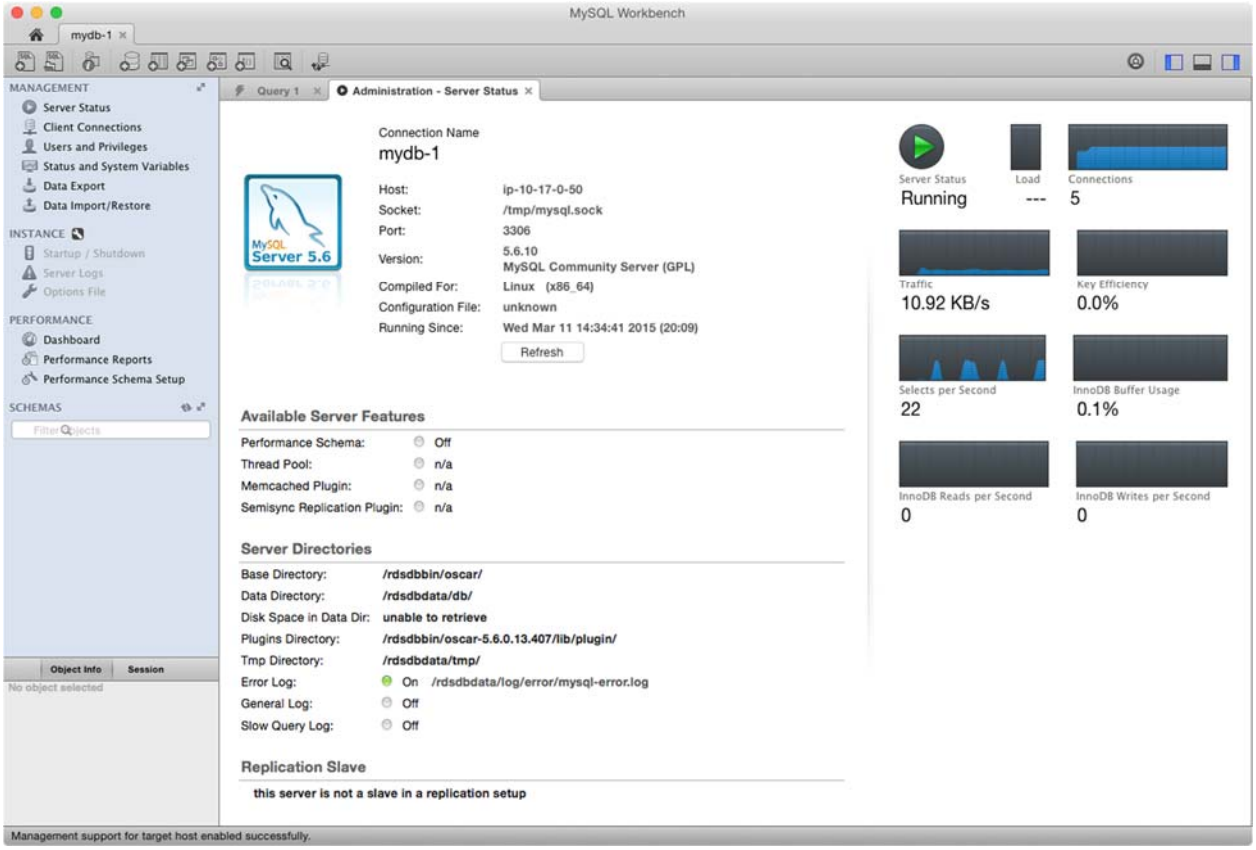
Getting Connected

Connecting Over the Internet

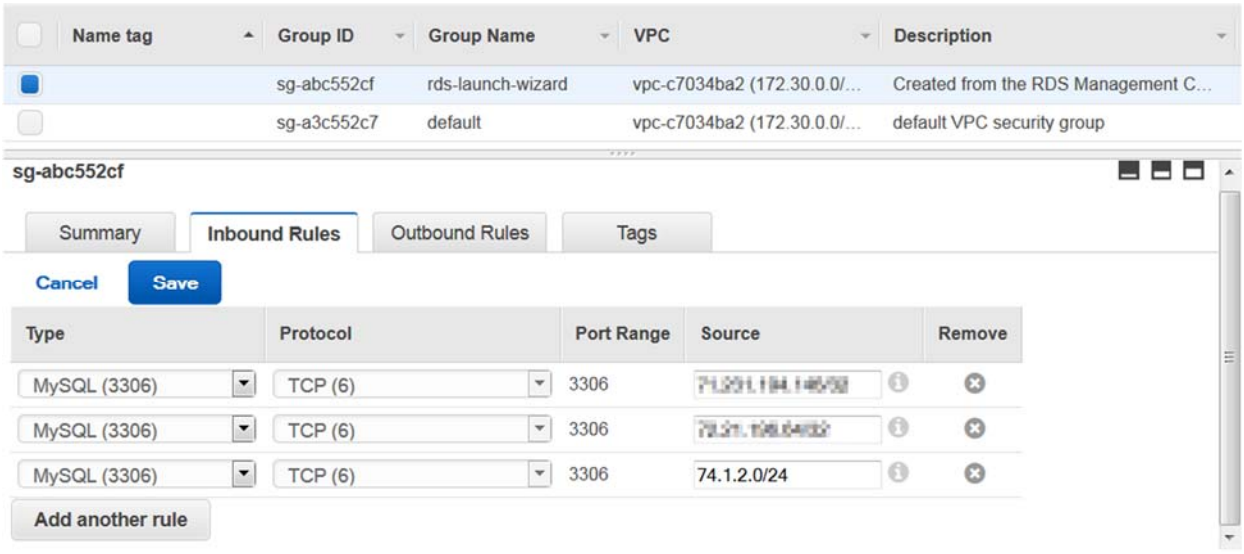
The previous sections describe a simple VPC configuration that enables connectivity to Aurora instances over the Internet. If you were following along and used the Amazon RDS launch wizard to create the VPC and the Aurora instance, the VPC will already be configured to accept incoming connections from the IP address used to run the Amazon RDS launch wizard. The endpoint (DNS name and port) that you will use to connect to the instance can be found in the **Instances** section of the Amazon RDS console as shown in the figure following.

The following figures show an example of connecting to an Aurora instance using the MySQL Workbench utility.





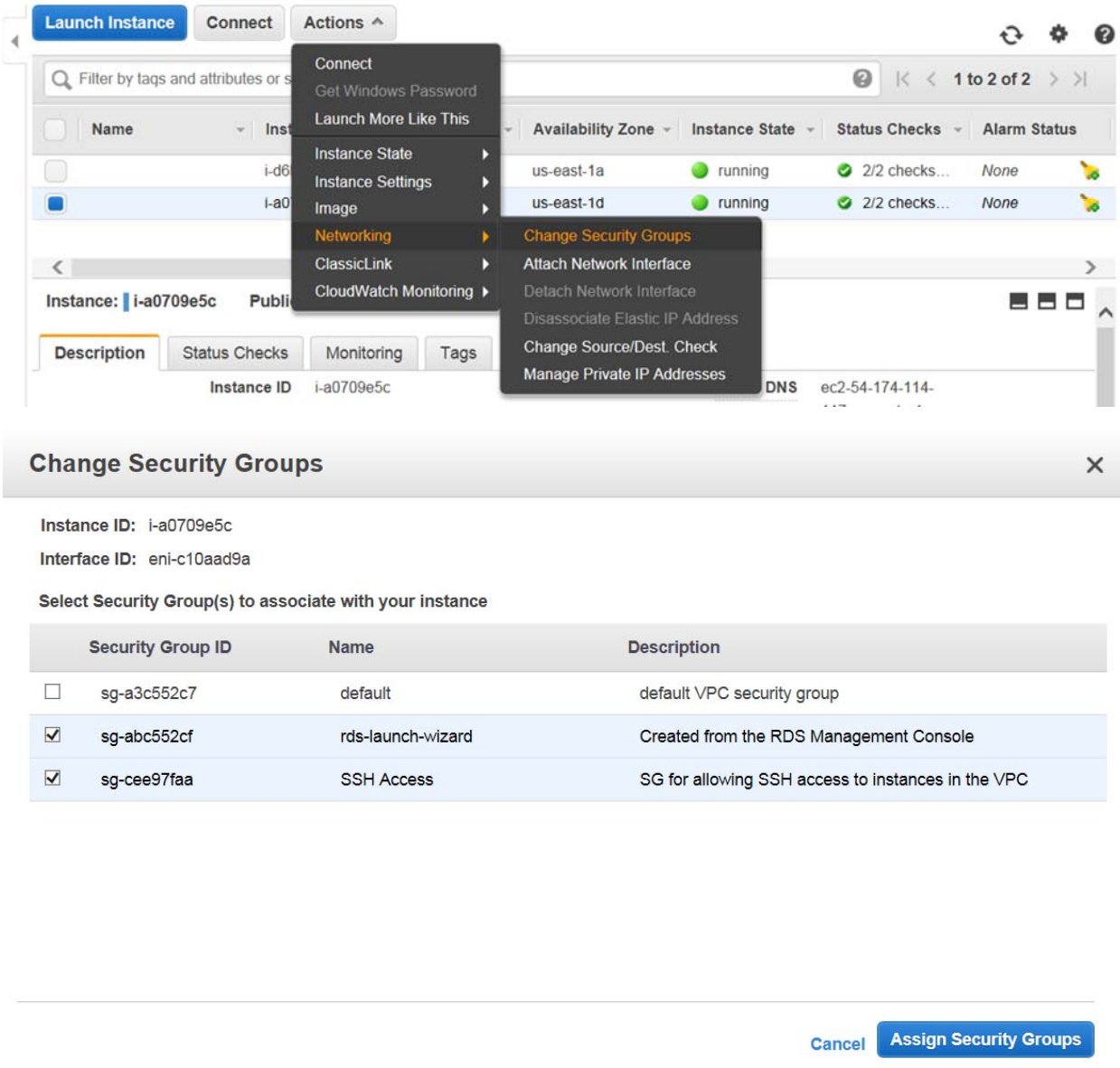
If you need to connect to Aurora from other devices, you'll need to add their IP addresses or IP address ranges to the VPC security group, as shown following.



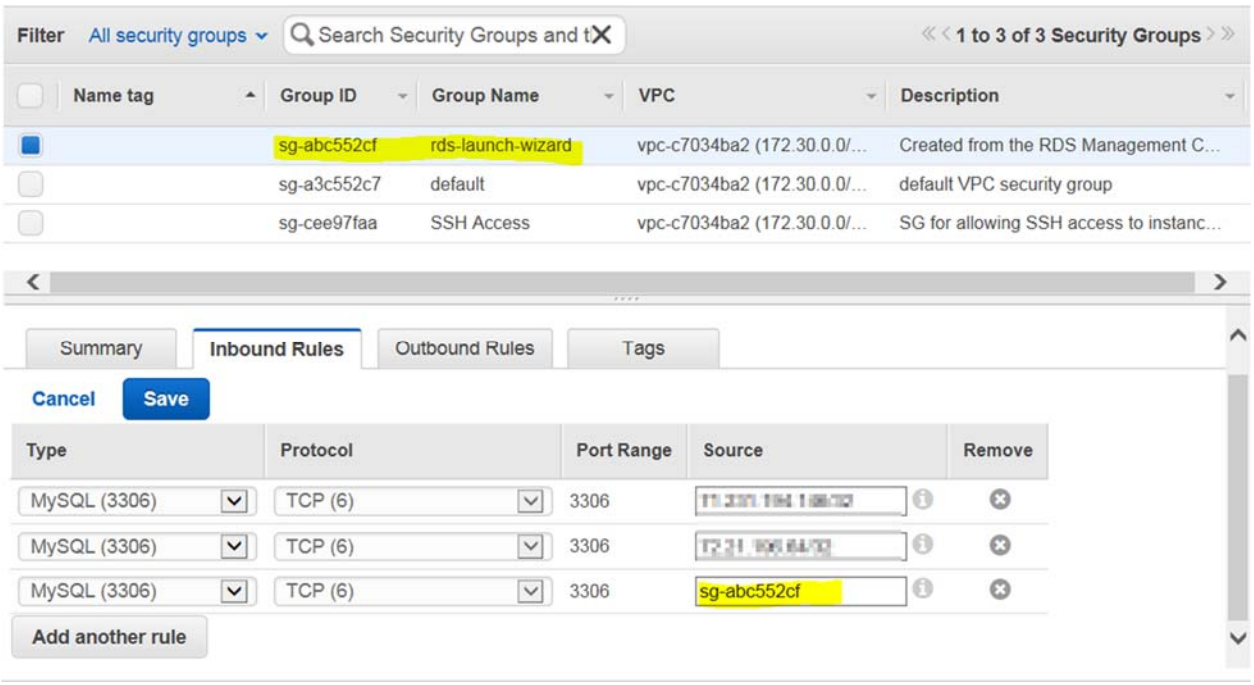
If you used a port other than the default (3306), use **Custom TCP Rule** for **Type** and specify the appropriate port in the **Port Range** field.

Connecting from Within the Same VPC

In order to connect to your Aurora instances from EC2 instances in the same VPC, you'll need to associate the EC2 instances with a VPC security group that allows access to the Aurora instances. To do so, select the instances in the Amazon EC2 console. For **Actions**, choose **Networking**, **Change Security Groups**, and then choose an appropriate VPC security group, as shown in the following example.



You might also need to add a rule to the VPC security group allowing traffic from instances associated with that group, as shown in the following example from the Amazon VPC console.



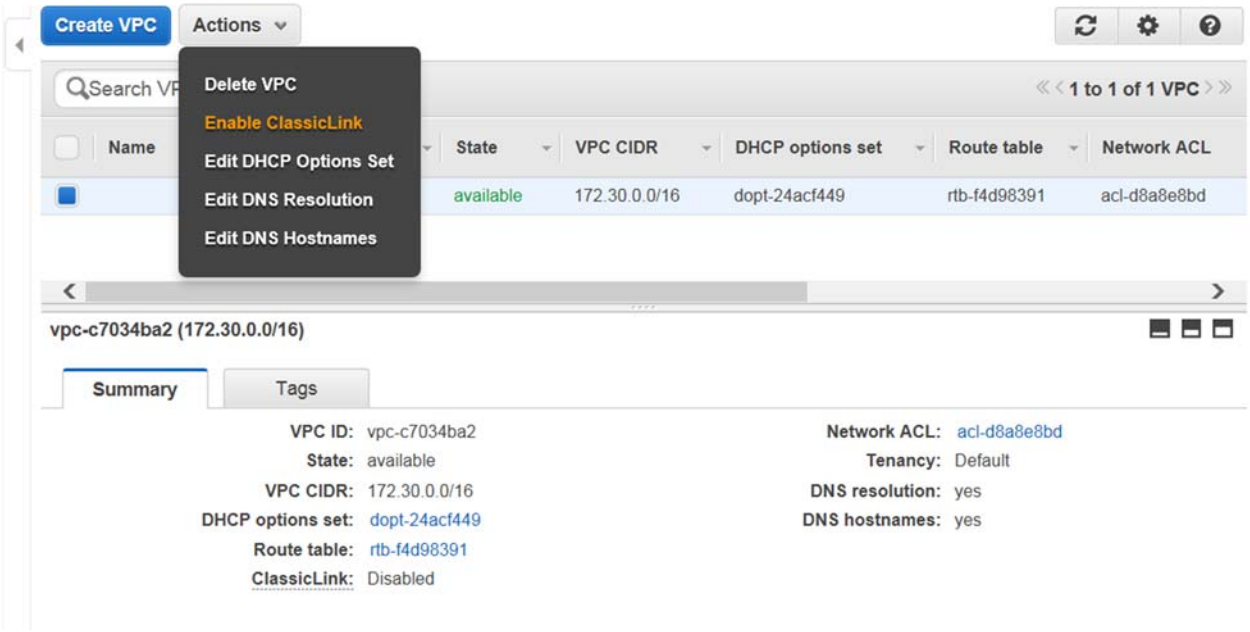
Once these changes are made, you can connect to the Aurora instance—for example, using the MySQL command line tool:

```
$ mysql --user=admin -p --host=mydb-1-cluster.cluster-cmswbbjge3do.us-east-1-
beta.rds.amazonaws.com
```

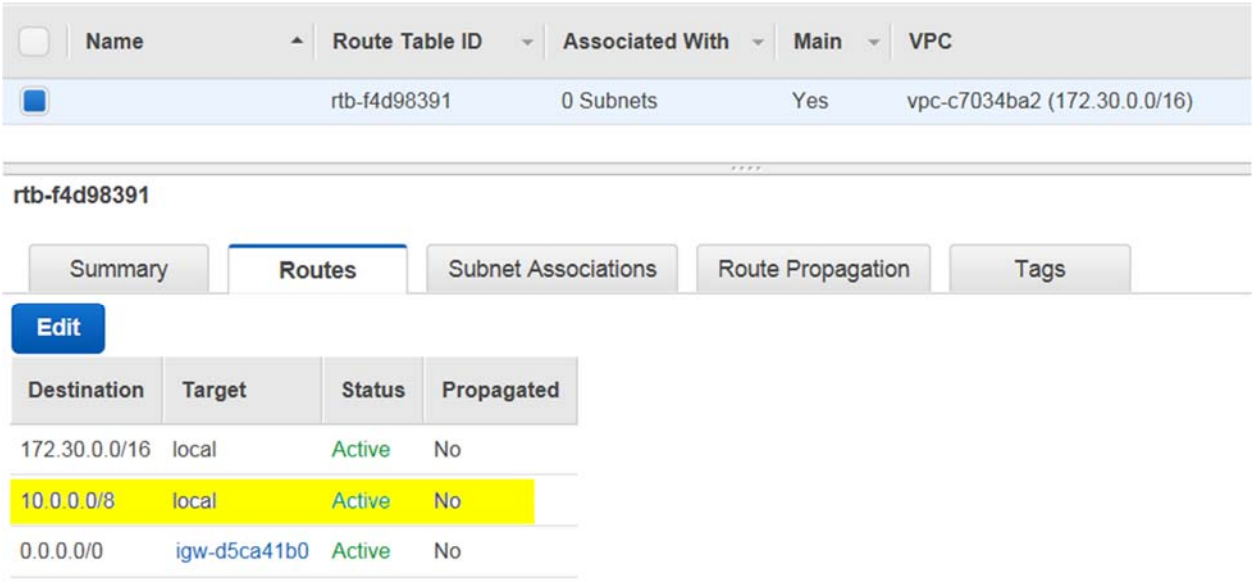
Note that the DNS hostname for the Aurora instance resolves to its internal IP address when used within EC2 instances in the same VPC. This approach allows communication over the AWS interinstance network, providing high bandwidth and low latency without incurring network bandwidth charges associated with communicating over the Internet.

Connecting from EC2-Classic

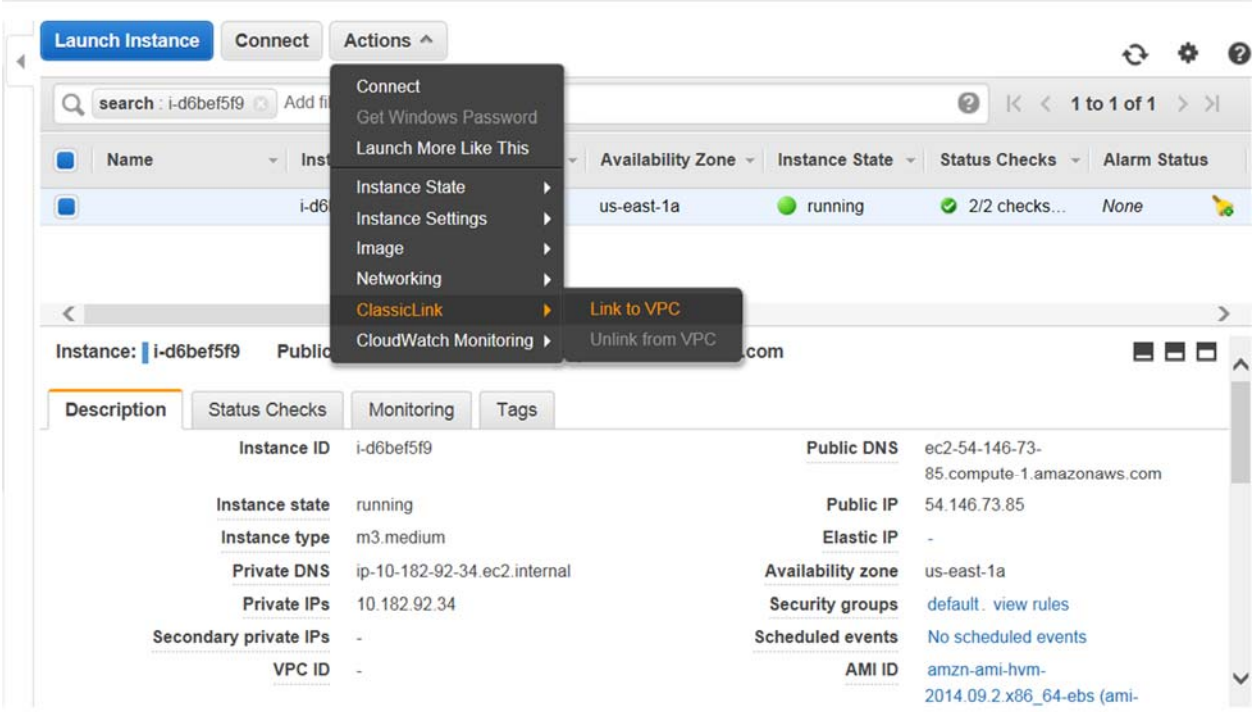
If you need to connect to Aurora from EC2-Classic instances (that is, instances that are not in a VPC) in the same region as the Aurora instance, you can enable ClassicLink in the VPC and manage access using VPC security groups, as shown in the following example from the Amazon VPC console.

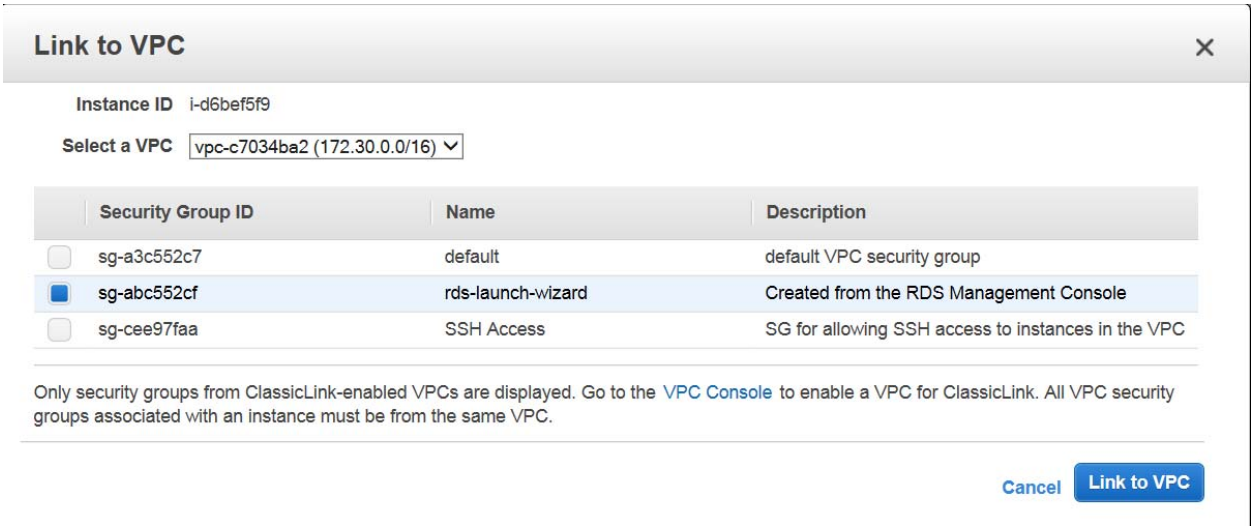


Enabling ClassicLink adds a new entry to the VPC route table to allow network traffic to the AWS interinstance network, as shown following.



After enabling ClassicLink on the VPC, we can now add EC2-Classic instances to the VPC security group that provides access to your Aurora instances, as shown in the following example from the Amazon EC2 console.





We can now connect to the Aurora instance by using its private IP address, as shown following. Note that we cannot use the DNS name because that resolves to the public IP address from EC2-Classic instances and we haven't defined rules to allow communication with the public IP address from EC2-Classic. We can add rules to allow communication by using the public IP address, but then we wouldn't be taking advantage of the benefits ClassicLink provides.

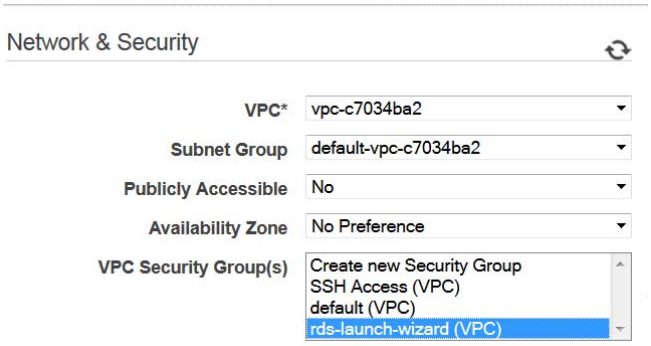
```
$ mysql --user=admin -p --host=172.30.3.168
```

Hiding Aurora Instances from the Internet

For many use cases, allowing direct access to databases from the Internet is undesirable. In this section, we'll describe how to configure the VPC so that your Aurora instances are not visible to the Internet and

can be accessed only by EC2 instances or devices in the same VPC as the Aurora instances. A common use case is a public-facing web application and an Aurora instance that is not publicly accessible.

The simplest way to hide Aurora instances from the Internet is to specify **No** in the **Publicly Accessible** field when creating an instance. The instance will be created with a private IP address, but no public IP address. In this case, the only way to communicate with the instance is from within the VPC or EC2-Classic instances that have a ClassicLink connection to the VPC. In the example following, the Aurora instance is being created in the same VPC we've been using previously and uses the VPC security group that we configured for public access. However, because the Aurora instance has no public IP address, it cannot be reached from the Internet even though the VPC security group allows incoming traffic from the Internet.



Notice that we can use the DNS name for this Aurora instance (within AWS in the same region) because it resolves only to the private IP address:

```
$ mysql --user=admin -p --host=my-private-db-2-cluster.cluster-cmswbbjge3do.us-east-1-
beta.rds.amazonaws.com
```

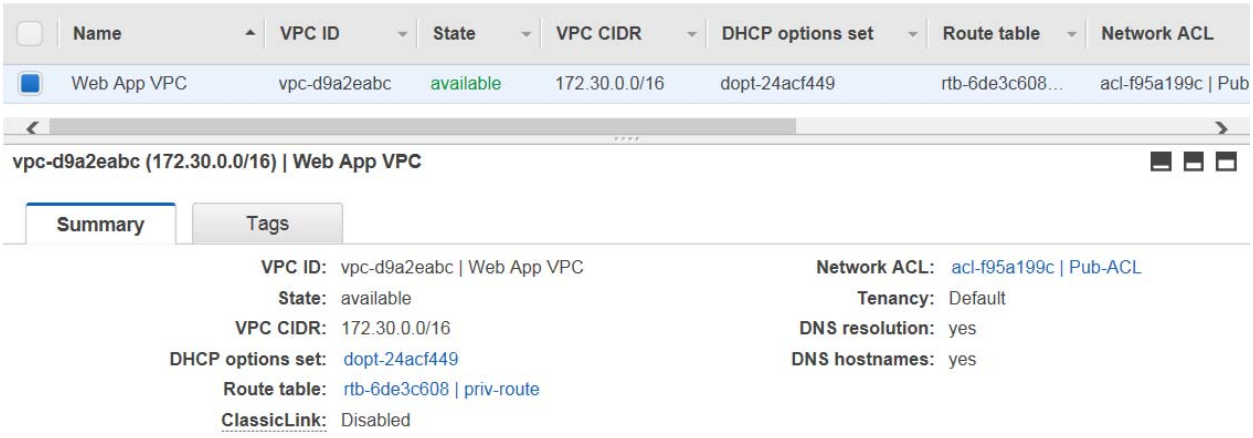
Using an Existing VPC

If you already have an Amazon VPC, you can provision Aurora instances in it as well. Aurora requires a minimum of three AWS Availability Zones for high availability, so you'll need at least three VPC subnets—one for each AZ. You'll also need to create an RDS DB subnet group so that Amazon RDS knows which subnets to use for your Aurora instances.

We will host a web-facing app that accesses an Aurora cluster with one writer and two reader instances. The database instances should be accessible only by the web app.

Configuring Your Amazon VPC

In order to prevent access to the Aurora instances from outside the Amazon VPC, we will use both VPC subnet groups and AWS network ACLs to limit access to the databases, as shown following.



The VPC is in a region with four Availability Zones, so it has been configured with four private and four public subnets—one private and one public subnet for each of the four Availability Zones. The databases will reside in the private subnets, and the web app resides in the public subnets.

Name	Subnet ID	State	CIDR	Availability Zone	Route Table	Auto-assign Public
priv-1a	subnet-93551da9	available	172.30.1.0/24	us-east-1a	rtb-6de3c608 | p...	No
priv-1b	subnet-6b5a3e1c	available	172.30.2.0/24	us-east-1b	rtb-6de3c608 | p...	No
priv-1d	subnet-1ab33643	available	172.30.3.0/24	us-east-1d	rtb-6de3c608 | p...	No
priv-1e	subnet-ab2b8980	available	172.30.4.0/24	us-east-1e	rtb-6de3c608 | p...	No
pub-1a	subnet-09561e33	available	172.30.5.0/24	us-east-1a	rtb-2fefca4a | pu...	Yes
pub-1b	subnet-0d5a3e7a	available	172.30.6.0/24	us-east-1b	rtb-2fefca4a | pu...	Yes
pub-1d	subnet-dfb33686	available	172.30.7.0/24	us-east-1d	rtb-2fefca4a | pu...	Yes
pub-1e	subnet-f22b89d9	available	172.30.8.0/24	us-east-1e	rtb-2fefca4a | pu...	Yes

There are two route tables, as shown following—one that has a route to an Amazon VPC Internet gateway for the public subnets and one that has no external routing for the private subnets. You can see the route table associations for each subnet in the figure preceding.

Name	Route Table ID	Associated With	Main	VPC
priv-route	rtb-6de3c608	0 Subnets	Yes	vpc-d9a2eabc (172.30.0.0/16) | We...
pub-route	rtb-2fefca4a	4 Subnets	No	vpc-d9a2eabc (172.30.0.0/16) | We...

rtb-2fefca4a | pub-route

Summary | **Routes** | Subnet Associations | Route Propagation | Tags

Edit

Destination	Target	Status	Propagated
172.30.0.0/16	local	Active	No
0.0.0.0/0	igw-c248c3a7	Active	No

The network ACLs for the private subnets are configured to allow incoming traffic only from the public subnets and only on the port used by the Aurora instances:

Name	Network ACL ID	Associated With	Default	VPC
Priv-ACL	acl-b45e1dd1	4 Subnets	No	vpc-d9a2eabc (172.30.0.0/16) | Web A...
Pub-ACL	acl-f95a199c	4 Subnets	Yes	vpc-d9a2eabc (172.30.0.0/16) | Web A...

acl-b45e1dd1 | Priv-ACL

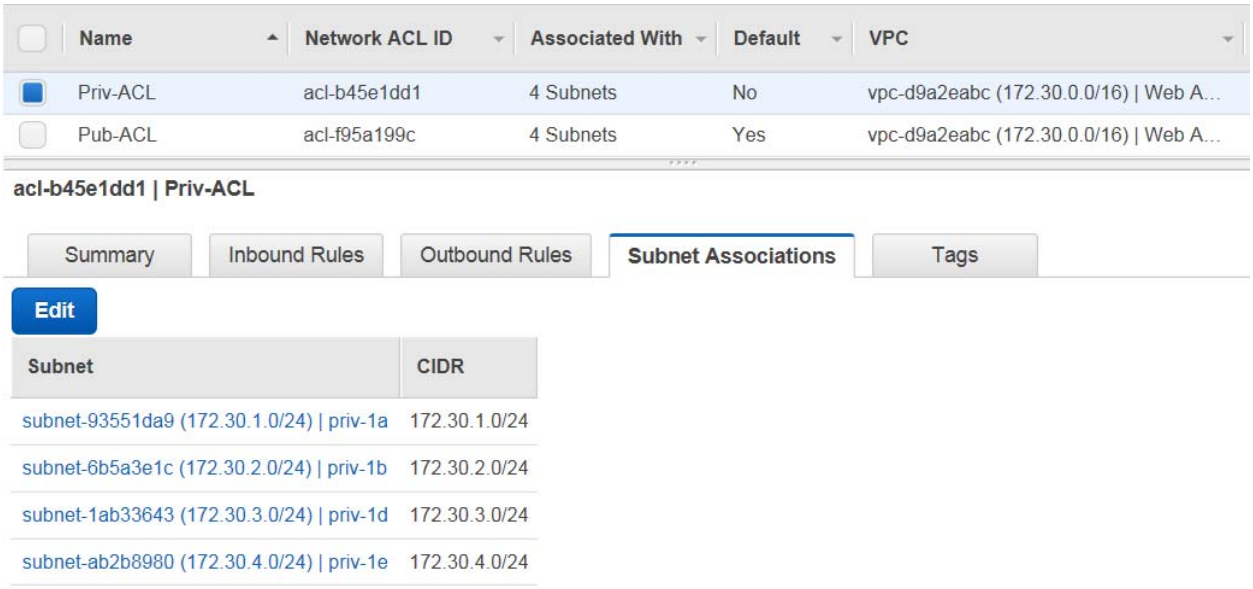
Summary | **Inbound Rules** | Outbound Rules | Subnet Associations | Tags

Allows inbound traffic. Because network ACLs are stateless, you must create inbound and outbound rules.

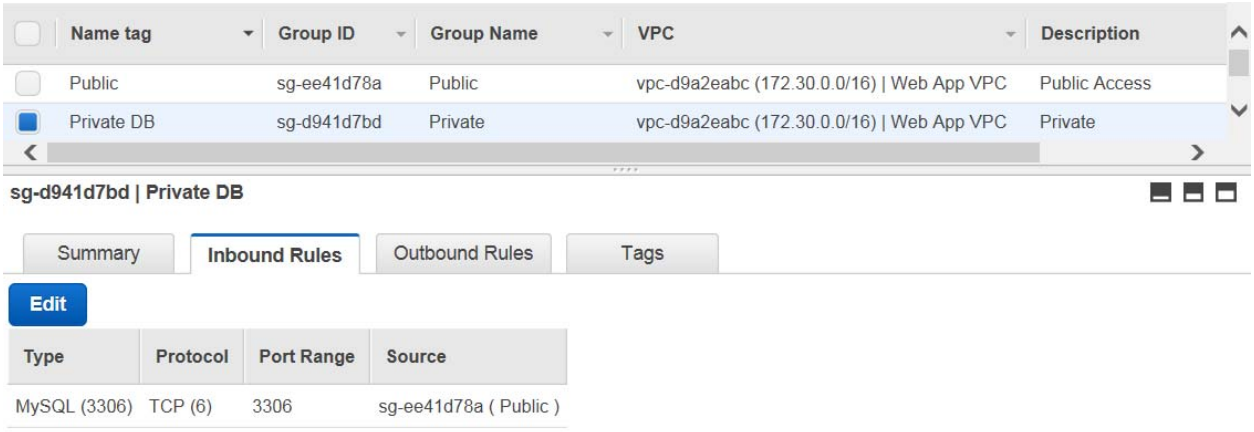
Edit

Rule #	Type	Protocol	Port Range	Source	Allow / Deny
100	MySQL (3306)	TCP (6)	3306	172.30.5.0/24	ALLOW
200	MySQL (3306)	TCP (6)	3306	172.30.6.0/24	ALLOW
300	MySQL (3306)	TCP (6)	3306	172.30.7.0/24	ALLOW
400	MySQL (3306)	TCP (6)	3306	172.30.8.0/24	ALLOW
*	ALL Traffic	ALL	ALL	0.0.0.0/0	DENY

Each of the four private subnets is associated with this network ACL:

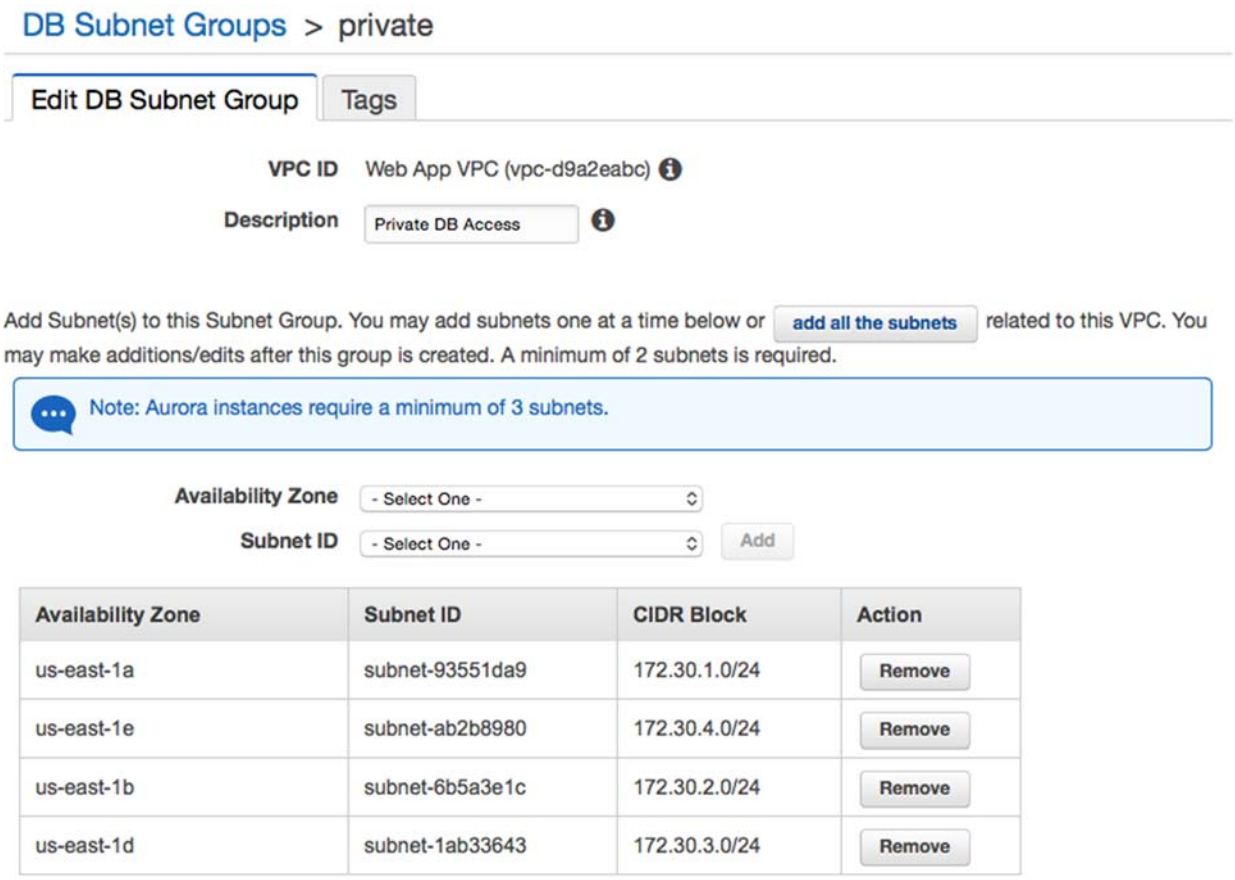


There are also two VPC security groups, as shown following—one for database use and one for web app use. The inbound rule for the private security group accepts traffic only on the database port and only from instances that are associated with the public VPC security group. The web app instances are associated with the public VPC security group.



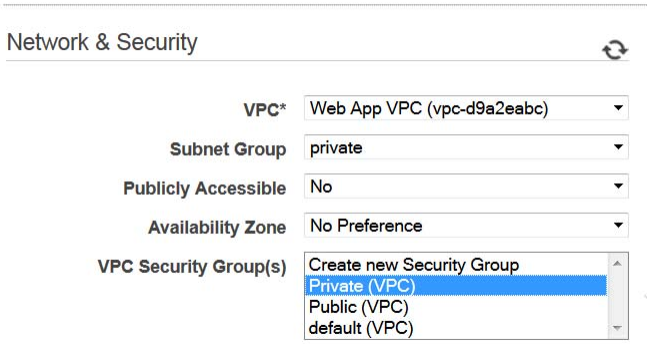
Configuring RDS DB Subnet Groups

Before we can create Aurora instances in this VPC, we need to tell Amazon RDS which VPC subnets to use. In the example illustrated following, an RDS DB subnet group named `private` was created that maps to each of the VPC private subnets where we want RDS to provision our Aurora instances.

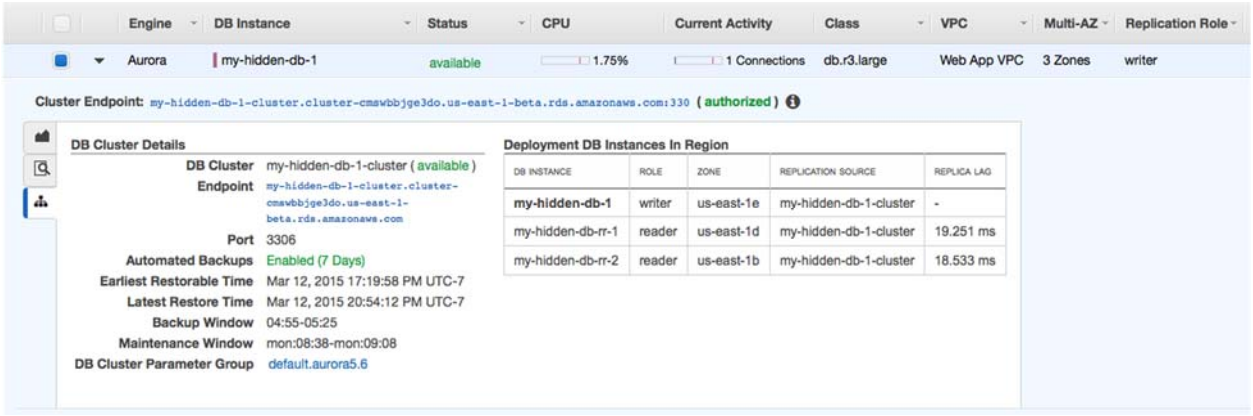


Aurora Cluster Creation

Now we can create Aurora instances in the VPC. The subnet group is set to use the `private` RDS DB subnet group, **Publicly Accessible** is set to **No** so that the Aurora instance will have a private IP address only, and the **VPC Security Group** field is set to the `Private` VPC security group.

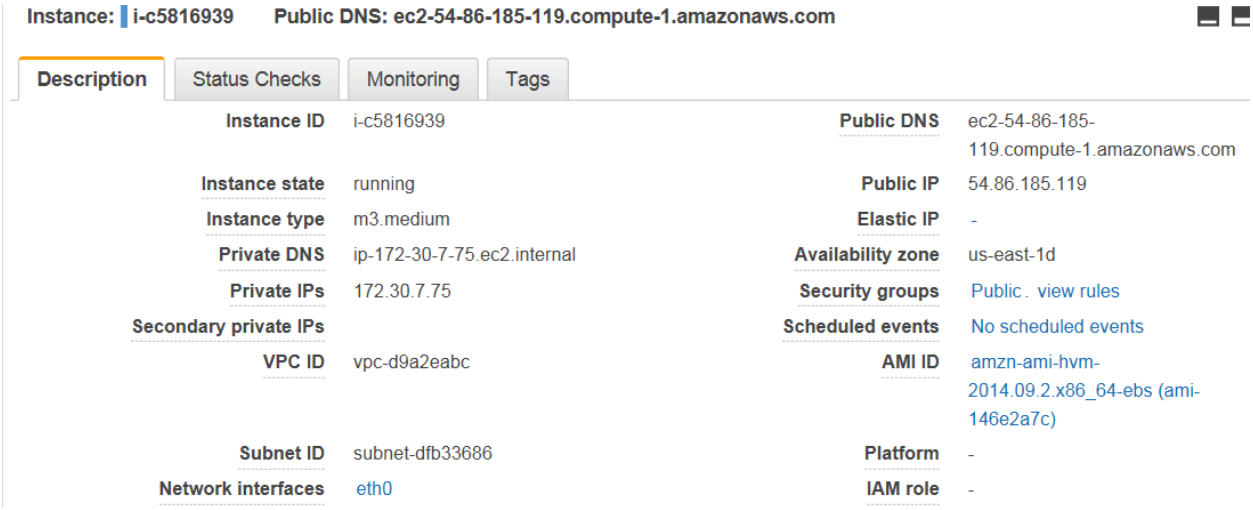


Here's the cluster after creating two Aurora Replicas. Notice that each instance is in a different AZ.



Accessing the Aurora Cluster Instances

Using an EC2 instance created in the same VPC in one of the public subnets, we can connect to each of the instances in the Aurora cluster, but they will not be accessible from anywhere outside the VPC.



First, we'll create a new database and table on the writer instance using the following commands.

```
[ec2-user@ip-172-30-7-75 ~]$ mysql --user=admin -p --host=my-hidden-db-1-cluster.cluster-
cmswbbjge3do.us-east-1-beta.rds.amazonaws.com

mysql> create database mydb;
Query OK, 1 row affected (0.02 sec)

mysql> create table mydb.myuser as select * from mysql.user;

Query OK, 3 rows affected (0.05 sec)
Records: 3  Duplicates: 0  Warnings: 0
```

Now we'll read the table from one of the reader instances using the following commands.

```
[ec2-user@ip-172-30-7-75 ~]$ mysql --user=admin -p --host=my-hidden-db-rr-1.cmswbbjge3do.us-east-
1-beta.rds.amazonaws.com

mysql> select count(*) from mydb.myuser;
+----------+
| count(*) |
+----------+
|        3 |
+----------+
1 row in set (0.01 sec)

mysql> exit
```

Now we'll read the table from the other reader instance.

```
[ec2-user@ip-172-30-7-75 ~]$ mysql --user=admin -p --host=my-hidden-db-rr-2.cmswbbjge3do.us-east-
mysql>
select count(*) from mydb.myuser;
+----------+
| count(*) |
+----------+
|        3 |
+----------+
1 row in set (0.01 sec)

mysql>
```

Now you've learned how to connect to RDS Aurora instances from on-premises equipment and EC2-Classic instances using AWS ClassicLink, and how to hide RDS Aurora instances from the Internet while allowing access from Web-facing apps. For further reading, see the following.

Further Reading

Amazon Aurora: http://aws.amazon.com/rds/aurora/
Amazon VPC: http://aws.amazon.com/vpc/