



Hands-on tutorials

# Connect to Private Amazon RDS PostgreSQL Database Using AWS CloudShell



# Connect to Private Amazon RDS PostgreSQL Database Using AWS CloudShell: Hands-on tutorials

Copyright © 2026 Amazon Web Services, Inc. and/or its affiliates. All rights reserved.

Amazon's trademarks and trade dress may not be used in connection with any product or service that is not Amazon's, in any manner that is likely to cause confusion among customers, or in any manner that disparages or discredits Amazon. All other trademarks not owned by Amazon are the property of their respective owners, who may or may not be affiliated with, connected to, or sponsored by Amazon.

---

# Table of Contents

<b>Connect to Private Amazon RDS PostgreSQL Database Using AWS CloudShell .....</b>	<b>1</b>
Introduction .....	1
Prerequisites .....	1
Tasks .....	2
Implementation .....	2
Conclusion .....	12

# Connect to Private Amazon RDS PostgreSQL Database Using AWS CloudShell

<b>AWS experience</b>	Beginner
<b>Time to complete</b>	30 minutes
<b>Cost to complete</b>	Less than \$1 when completed in 1 hour
<b>Services used</b>	<a href="#">AWS CloudShell</a> and <a href="#">Amazon RDS for PostgreSQL</a>
<b>Last updated</b>	February 23, 2026

## Introduction

Following AWS best practices, databases should be hosted in private subnets within an [Amazon Virtual Private Cloud \(Amazon VPC\)](#) for enhanced security. When an Amazon RDS PostgreSQL database is hosted in private subnets without public access, you must create another instance in a public subnet and then connect to the database instance. Alternatively, you can establish a connection by creating an [AWS Client VPN](#); however, both options incur additional costs.

A simpler and more cost-effective alternative is to use [AWS CloudShell](#). AWS CloudShell is a browser-based, pre-authenticated shell that you can launch directly from the AWS Management Console. The AWS CloudShell VPC feature allows you to create a CloudShell environment within your VPC. For each VPC environment, you can specify a VPC, add a subnet, and associate up to five security groups. CloudShell inherits the network configuration of the VPC, enabling you to use CloudShell securely within the same subnet as other resources in the VPC and connect to them.

There is no additional charge for AWS CloudShell. You only pay for other AWS resources you use with CloudShell to create and run your applications.

## Prerequisites

Before starting this tutorial, you will need:

- An AWS account: If you don't already have one, follow the [Setting Up Your AWS Environment](#) getting started guide for a quick overview.

## Tasks

This tutorial is divided into the following short tasks. You must complete each task before moving on to the next one.

1. Create a custom Amazon Virtual Private Cloud with public and private subnets (5 Minutes)
2. Create an Amazon RDS PostgreSQL database hosted in private subnets within an Amazon VPC. (10 Minutes)
3. Set up an AWS CloudShell Virtual Private Cloud environment and test connectivity (10 Minutes)
4. Clean up resources (5 Minutes)

## Implementation

### Task 1: Create a custom Amazon Virtual Private Cloud with public and private subnets

In this task, you will use an AWS CloudFormation template to create a custom Amazon VPC with public and private subnets.

1. Open [AWS CloudShell](#)
2. Copy and paste the following commands into CloudShell:

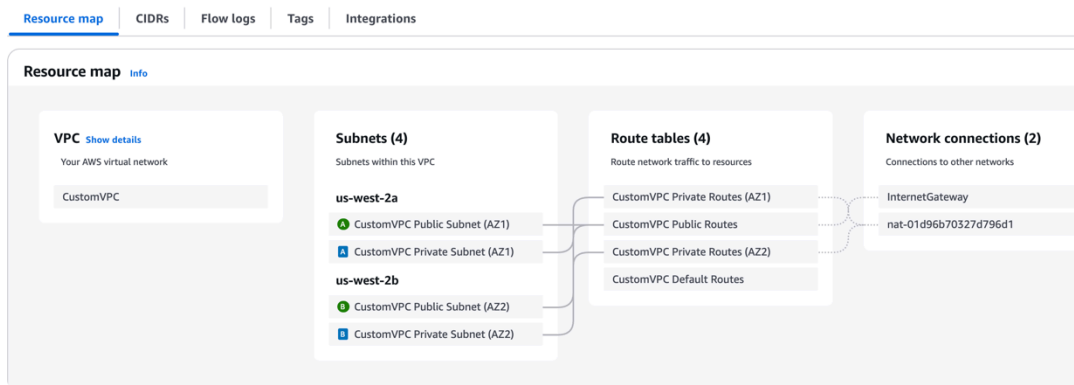
```
git clone https://github.com/aws-samples/sample-Amazon-Q-Developer-Cookbook.git
cd sample-Amazon-Q-Developer-Cookbook/dev-vpc-with-private-subnet/example-result/
custom-vpc
chmod 700 deploy.sh
./deploy.sh
```

3. Choose **Paste**.

The commands performed the following actions:

- Deployed an AWS CloudFormation template in a VPC with a pair of public and private subnets spread across two Availability Zones.

- Deployed an internet gateway with a default route on the public subnets.
  - Deployed a NAT gateway and default routes for the NAT gateway in the private subnets.
4. Open [AWS CloudFormation](#) and wait for the Status column of the **custom-vpc** stack to show **CREATE\_COMPLETE**.
  5. Open [Amazon VPC](#).
  6. Select **Your VPCs** from the left menu.
  7. Select **CustomVPC**, and then select the **Resource map tab** to review the layout of the subnets and route tables.



## Task 2: Create an Amazon RDS PostgreSQL database hosted in private subnets within an Amazon VPC

In this task, you will create an Amazon RDS PostgreSQL database hosted in private subnets within an Amazon VPC you've created in the previous task.

1. Open the [Amazon RDS](#) console, and select **Create a database**.

**Create a database**

Amazon Relational Database Service (RDS) makes it easy to set up, operate, and scale a relational database in the cloud.

[Create a database](#)

Note: your DB instances will launch in the **US East (N. Virginia)** region

You can use a backup from Amazon S3 to restore and create a new Aurora MySQL and MySQL database.

[Restore from S3](#)

2. For Engine options, select **PostgreSQL** Engine type.


**Create database** Info


**Choose a database creation method**


- Full configuration**  
You set all of the configuration options, including ones for availability, security, backups, and maintenance.
- Easy create**  
Use recommended best-practice configurations. Some configuration options can be changed after the database is created.


**Engine options**


Engine type Info


Aurora (MySQL Compatible)  



Aurora (PostgreSQL Compatible)  



MySQL  


PostgreSQL  


MariaDB  


Oracle  


Microsoft SQL Server  


IBM Db2  


3. For Engine version, select **PostgreSQL 16.8-R2**.

**Engine version** Info

View the engine versions that support the following database features.

▼ Hide filters

Show only versions that support the Multi-AZ DB cluster Info

Create a Multi-AZ DB cluster with one primary DB instance and two readable standby DB instances. Multi-AZ DB clusters provide up to 2x faster transaction commit latency and automatic failover in typically under 35 seconds.

**Engine version**

PostgreSQL 16.8-R2
▼

Enable RDS Extended Support Info

Amazon RDS Extended Support is a [paid offering](#). By selecting this option, you consent to being charged for this offering if you are running your database major version past the RDS end of standard support date for that version. Check the end of standard support date for your major version in the [RDS for PostgreSQL documentation](#).

4. Select the **Dev/Test** template with the **Single-AZ DB instance deployment** option.

**Templates**

Choose a sample template to meet your use case.

- Production**  
Use defaults for high availability and fast, consistent performance.
- Dev/Test**  
This instance is intended for development use outside of a production environment.
- Free tier**  
Use RDS Free Tier to develop new applications, test existing applications, or gain hands-on experience with Amazon RDS. Info

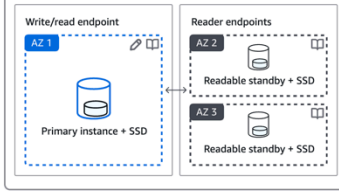
**Availability and durability**

**Deployment options** Info

Choose the deployment option that provides the availability and durability needed for your use case. AWS is committed to a certain level of uptime depending on the deployment option you choose. Learn more in the [Amazon RDS service level agreement \(SLA\)](#).

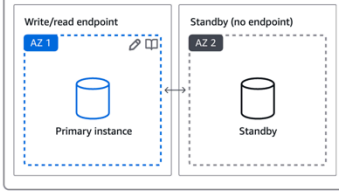
**Multi-AZ DB cluster deployment (3 instances)**  
Creates a primary DB instance with two readable standbys in separate Availability Zones. This setup provides:

- 99.95% uptime
- Redundancy across Availability Zones
- Increased read capacity
- Reduced write latency



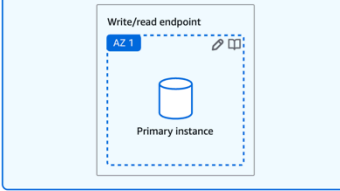
**Multi-AZ DB instance deployment (2 instances)**  
Creates a primary DB instance with a non-readable standby instance in a separate Availability Zone. This setup provides:

- 99.95% uptime
- Redundancy across Availability Zones



**Single-AZ DB instance deployment (1 instance)**  
Creates a single DB instance without standby instances. This setup provides:

- 99.5% uptime
- No data redundancy



5. Name your DB instance identifier.

- a. For example, **postgresql-demo**

**Settings**

**DB instance identifier** [Info](#)  
Type a name for your DB instance. The name must be unique across all DB instances owned by your AWS account in the current AWS Region.

postgresql-demo

The DB instance identifier is case-insensitive, but is stored as all lowercase (as in "mydbinstance"). Constraints: 1 to 63 alphanumeric characters or hyphens. First character must be a letter. Can't contain two consecutive hyphens. Can't end with a hyphen.

▼ **Credentials Settings**

**Master username** [Info](#)  
Type a login ID for the master user of your DB instance.

postgres

1 to 16 alphanumeric characters. The first character must be a letter.

**Credentials management**  
You can use AWS Secrets Manager or manage your master user credentials.

**Managed in AWS Secrets Manager - most secure**  
RDS generates a password for you and manages it throughout its lifecycle using AWS Secrets Manager.

**Self managed**  
Create your own password or have RDS create a password that you manage.

If you manage the master user credentials in AWS Secrets Manager, additional charges apply. See [AWS Secrets Manager pricing](#). Additionally, some RDS features aren't supported. See [limitations here](#).

**Select the encryption key** [Info](#)  
You can encrypt using the KMS key that Secrets Manager creates or a customer managed KMS key that you create.

aws/secretsmanager (default)

[Add new key](#)

6. Under Instance Configuration, select **Burstable classes**.

7. Select **db.t3.medium** for DB instance class, and set Allocated Storage to **20GB**.

**Instance configuration**

The DB instance configuration options below are limited to those supported by the engine that you selected above.

**DB instance class** [Info](#)

▼ **Hide filters**

Include previous generation classes

Standard classes (includes m classes)

Memory optimized classes (includes r and x classes)

**Burstable classes (includes t classes)**

db.t3.medium  
2 vCPUs 4 GiB RAM Network: Up to 2,085 Mbps

**Storage**

**Storage type** [Info](#)  
Provisioned IOPS SSD (io2) storage volumes are now available.

General Purpose SSD (gp3)  
Performance scales independently from storage

**Allocated storage** [Info](#)  
20 GIB  
Minimum: 20 GiB. Maximum: 32,768 GiB

**Provisioned IOPS** [Info](#)  
3000 IOPS  
Baseline IOPS of 3,000 IOPS is included for allocated storage less than 400 GiB.

8. Under **Connectivity**:

- Select the **CustomVPC** you created in previous task.
- Confirm that the **Public access** setting is set to **No**.
- Select the **default security group**.

**Virtual private cloud (VPC)** Info  
Choose the VPC. The VPC defines the virtual networking environment for this DB instance.

CustomVPC (vpc-0e715a73793ea2ad6)  
4 Subnets, 2 Availability Zones

Only VPCs with a corresponding DB subnet group are listed.

After a database is created, you can't change its VPC.

**DB subnet group** Info  
Choose the DB subnet group. The DB subnet group defines which subnets and IP ranges the DB instance can use in the VPC that you selected.

Create new DB Subnet Group

**Public access** Info

Yes  
RDS assigns a public IP address to the database. Amazon EC2 instances and other resources outside of the VPC can connect to your database. Resources inside the VPC can also connect to the database. Choose one or more VPC security groups that specify which resources can connect to the database.

No  
RDS doesn't assign a public IP address to the database. Only Amazon EC2 instances and other resources inside the VPC can connect to your database. Choose one or more VPC security groups that specify which resources can connect to the database.

**VPC security group (firewall)** Info  
Choose one or more VPC security groups to allow access to your database. Make sure that the security group rules allow the appropriate incoming traffic.

Choose existing  
Choose existing VPC security groups

Create new  
Create new VPC security group

**Existing VPC security groups**  
Choose one or more options

default

**Availability Zone** Info  
No preference

9. Leave all other options as their default settings, and choose **Create database**.

10 After the database instance successfully creates, select **View connection Details**.

Successfully created database **postgresql-demo** [View connection details](#)

You can use settings from postgresql-demo to simplify configuration of suggested database add-ons while we finish creating your DB for you.

**Databases (1)**  Group resources

Filter by databases

DB identifier	Status	Role	Engine	Region ...	Size	Recommendations
<a href="#">postgresql-demo</a>	Available	Instance	PostgreSQL	us-west-2a	db.t3.medium	

11 Copy the hostname of the instance, and select **Manage Credentials**.

**Connection details to your database postgresql-demo**

[Learn about connecting to your database](#)

**Master username**  
postgres

**Manage master credentials in AWS Secrets Manager**  
[Manage Credentials](#)

**Endpoint**  
postgresql-demo.-west-2.rds.amazonaws.com

12 Retrieve the password by selecting **Retrieve secret value**.

### Important

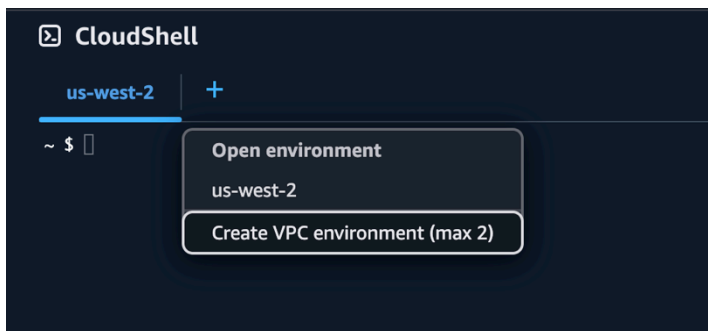
Take note of the **username**, **Endpoint**, and **password**. You will need these values for your VPC environment in the next task.



## Task 3: Set up an AWS CloudShell Virtual Private Cloud environment and test connectivity

In this task, you will set up an AWS CloudShell VPC environment and test connectivity.

1. Open [AWS CloudShell](#), and select the + button to bring up an option for **Create VPC environment**.



2. Name the VPC environment.
  - a. For example, **cloudshell-vpc-demo**.
3. Select **CustomVPC**, any **Private subnet**, and the **default security group**.
4. Choose **Create**.

## < CloudShell us-west-2

### Create a VPC environment

After creating a VPC environment, a new tab linked to this environment is added to CloudShell. You can access your VPC environment by selecting this tab.

#### Name

A unique VPC environment name used to identify it within AWS CloudShell.

Must contain up to 28 alphanumeric characters, hyphens, and no spaces. The first character must be a letter or a number.

#### Virtual private cloud (VPC)

#### Subnet

#### Security group

  
default - default VPC security group

Maximum of 5.

**i** After 30 minutes of inactivity, the shell session will terminate and the home directory of the VPC environment will be deleted.

Cancel

Create

**Note**

Public IP addresses are not allocated to CloudShell VPC environments by default. VPC environments created in public subnets with routing tables configured to route all traffic to Internet Gateway will not have access to public internet, but private subnets configured with Network Address Translation (NAT) have access to public internet. VPC environments created in such private subnets will have access to public internet.

- Once the environment is set up, install version 16 of PostgreSQL by copying and pasting these commands.
- Choose **Paste**.

**Note**

It is possible that your PostgreSQL version may be outdated compared to your Amazon RDS PostgreSQL database. These commands remove the older version and installs PostgreSQL version 16.

```
psql --version
sudo dnf remove postgresql15* -y
sudo dnf clean all
```

- After that completes, copy and paste the following command to install version 16 of PostgreSQL.
- Choose **Paste**.

```
sudo dnf install postgresql16 -y
```

- In your AWS CloudShell VPC environment, run the following PostgreSQL command:

**Note**

These are the values at the end of Task 2.

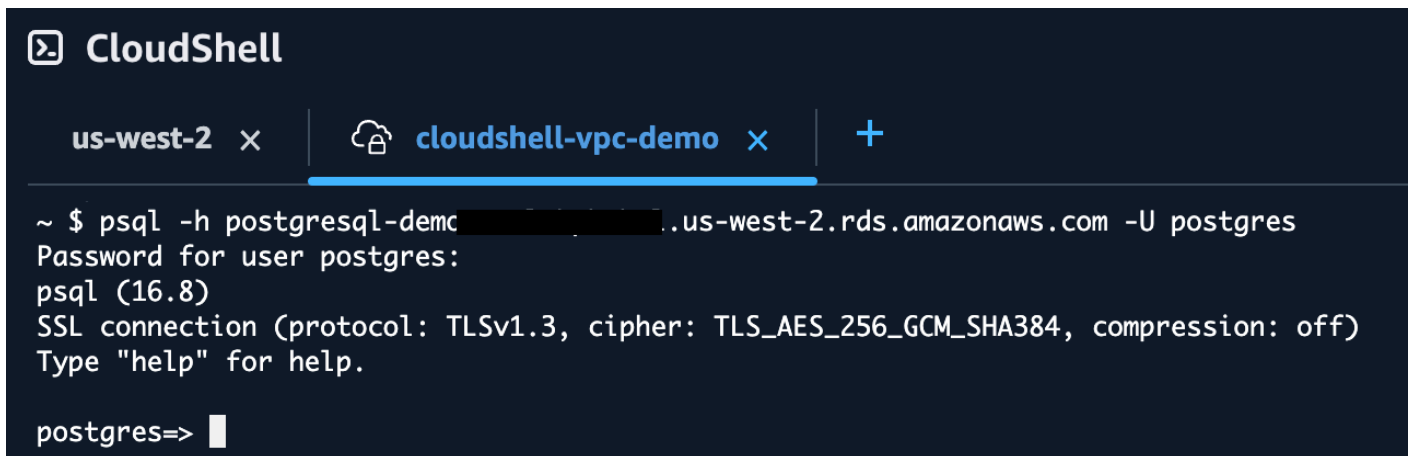
```
psql -h <HOSTNAME> -U <USERNAME>
```

**Note**

<HOSTNAME> is your database endpoint

<USERNAME> is your database administrator username

10 Enter your **password** to finish establishing a connection to your database.



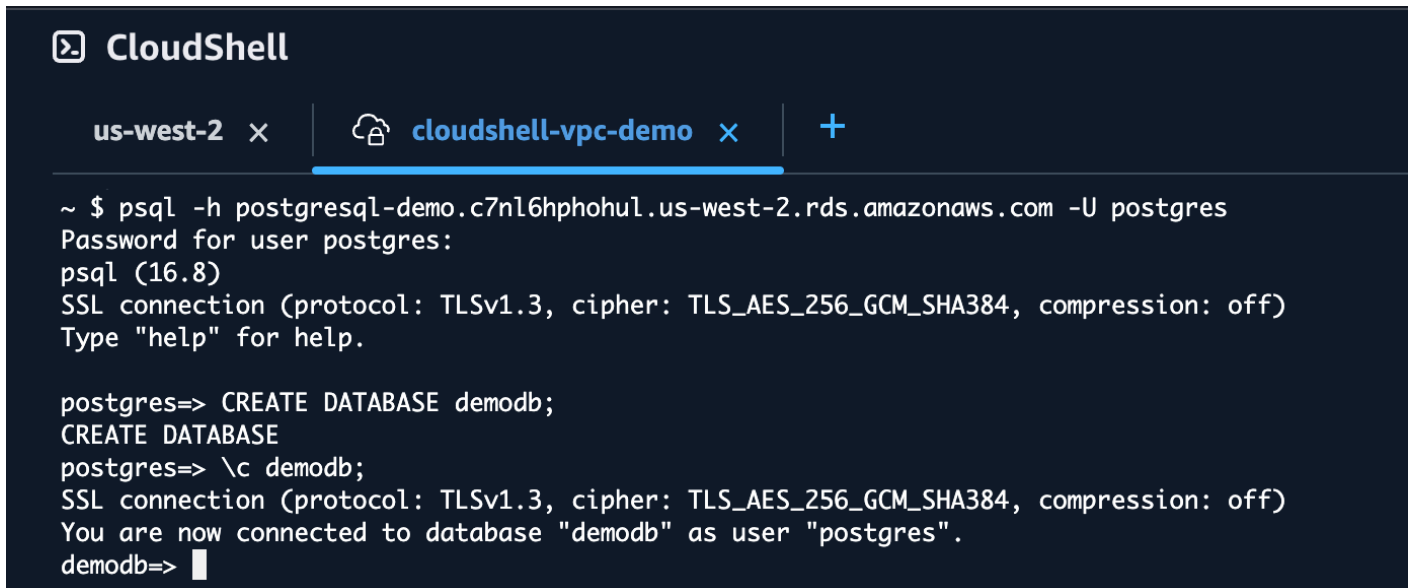
The screenshot shows the AWS CloudShell interface. At the top, there are two tabs: 'us-west-2' and 'cloudshell-vpc-demo'. The terminal window displays the following text:

```
~ $ psql -h postgresql-demo-xxxxxx.us-west-2.rds.amazonaws.com -U postgres
Password for user postgres:
psql (16.8)
SSL connection (protocol: TLSv1.3, cipher: TLS_AES_256_GCM_SHA384, compression: off)
Type "help" for help.

postgres=> |
```

11 Validate your setup by running this test command:

```
CREATE DATABASE demodb;
```



```
CloudShell
us-west-2 x | cloudshell-vpc-demo x | +
~ $ psql -h postgresql-demo.c7nl6hphohu1.us-west-2.rds.amazonaws.com -U postgres
Password for user postgres:
psql (16.8)
SSL connection (protocol: TLSv1.3, cipher: TLS_AES_256_GCM_SHA384, compression: off)
Type "help" for help.

postgres=> CREATE DATABASE demodb;
CREATE DATABASE
postgres=> \c demodb;
SSL connection (protocol: TLSv1.3, cipher: TLS_AES_256_GCM_SHA384, compression: off)
You are now connected to database "demodb" as user "postgres".
demodb=> |
```

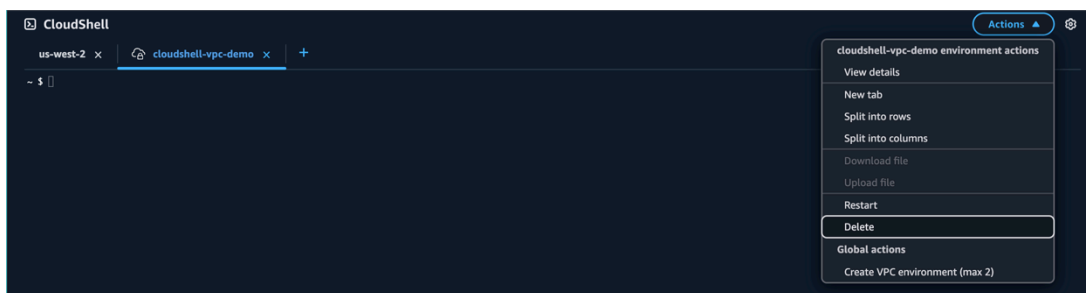
## Task 4: Clean up resources

To avoid unexpected charges, follow these clean-up steps:

1. Open [AWS CloudShell](#), and select **Delete**.

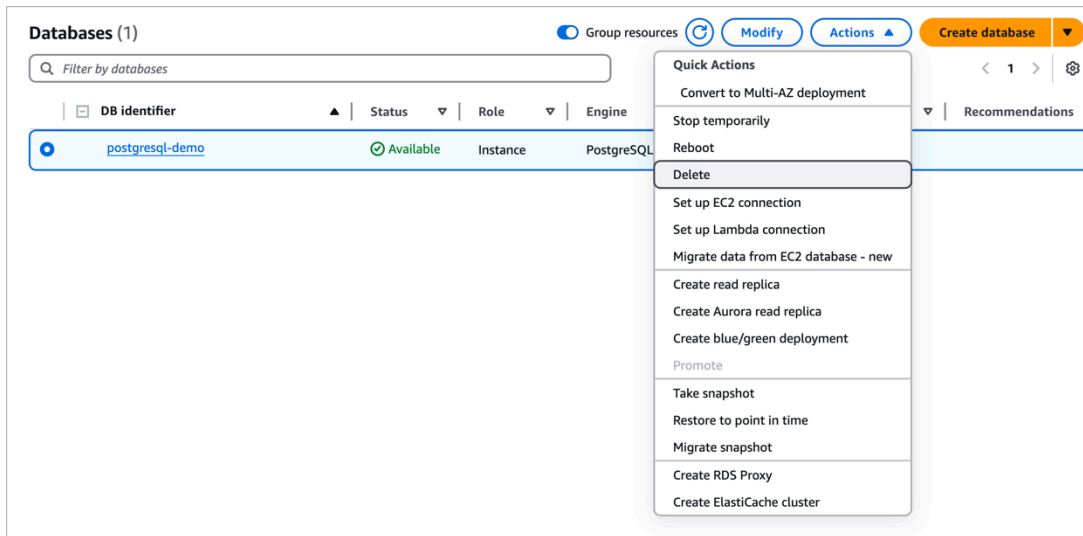
### Note

VPC environments do not have persistent storage. The \$HOME directory is deleted when your VPC environment times out (after 20-30 minutes of inactivity), or when you delete or restart your environment.



2. Enter **delete**, and choose **Delete** to confirm the deletion of the VPC environment.
3. Open [Amazon RDS](#), and select **Databases**.
4. Select **postgresql-demo**.

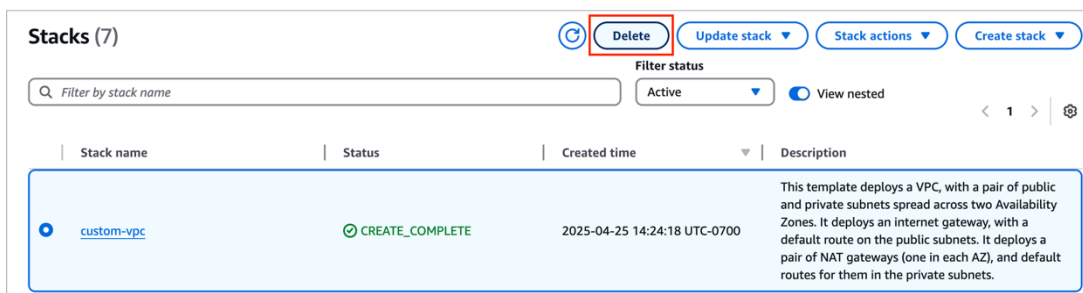
## 5. Select **Actions**, and select **Delete**.



6. Enter **delete me** to remove the PostgreSQL database instance.

7. Open [AWS CloudFormation](#), and select **custom-vcpc**.

8. Select **Delete**.



9. Choose **Delete** to remove the CloudFormation stack.

## Conclusion

You have learned how to connect to an Amazon RDS PostgreSQL instance in private subnets within Amazon VPC using AWS CloudShell.