

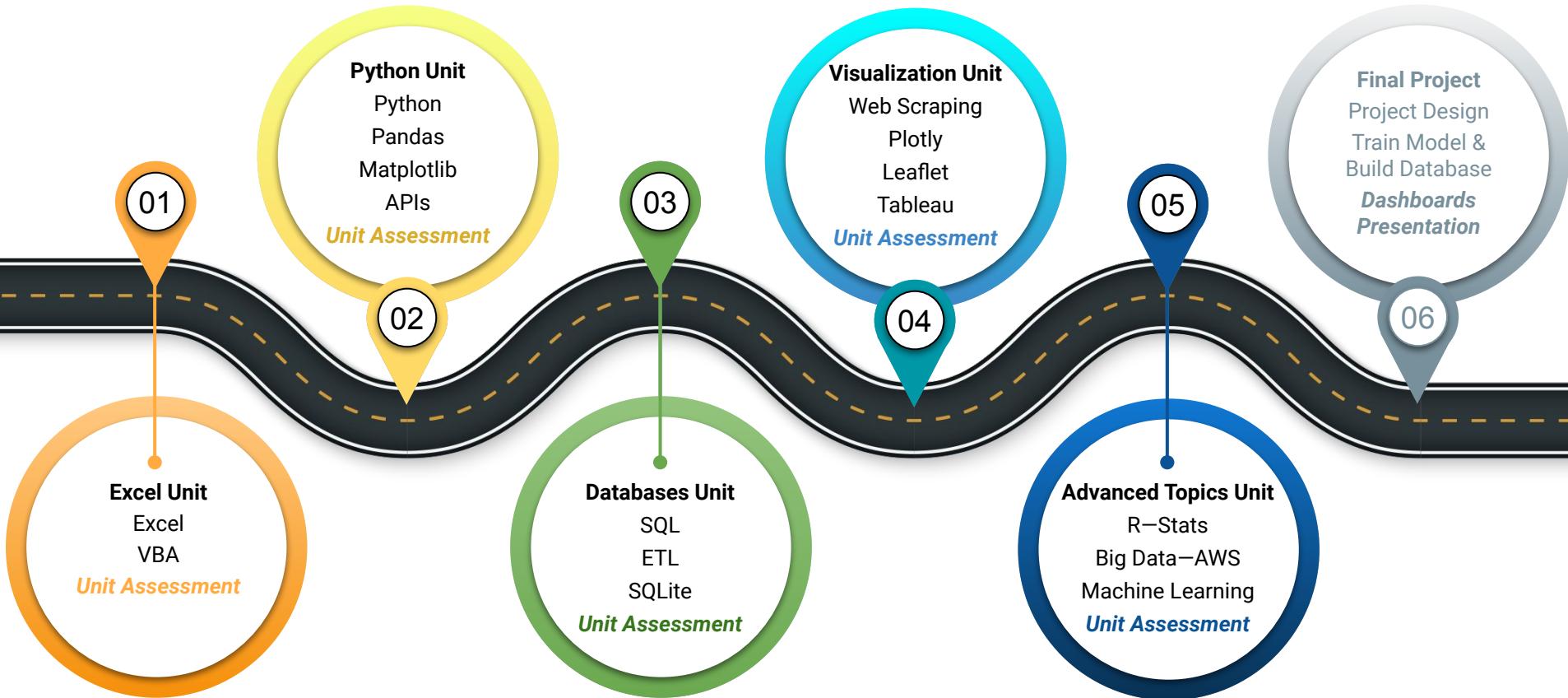


ETL with Amazon Web Services

Data Boot Camp
Lesson 16.2



The Big Picture



This Week: Big Data

By the end of this week, you'll know how to:



Define big data and describe the challenges associated with it



Define Hadoop and name the main elements of its ecosystem



Explain how MapReduce processes data



Define Spark and explain how it processes data



Describe how NLP collects and analyzes text data



Use AWS Simple Storage Service (S3) and relational databases for basic cloud storage



This Week's Challenge

Using the skills learned throughout the week, connect to an AWS RDS instance and perform ETL on an Amazon customer-review dataset to determine if Vine reviews show bias towards being more favorable.

Today's Agenda

By completing today's activities, you'll learn the following skills:

01

Using AWS

02

Creating a relational database (RDS) in AWS

03

Connecting an RDS to pgAdmin

04

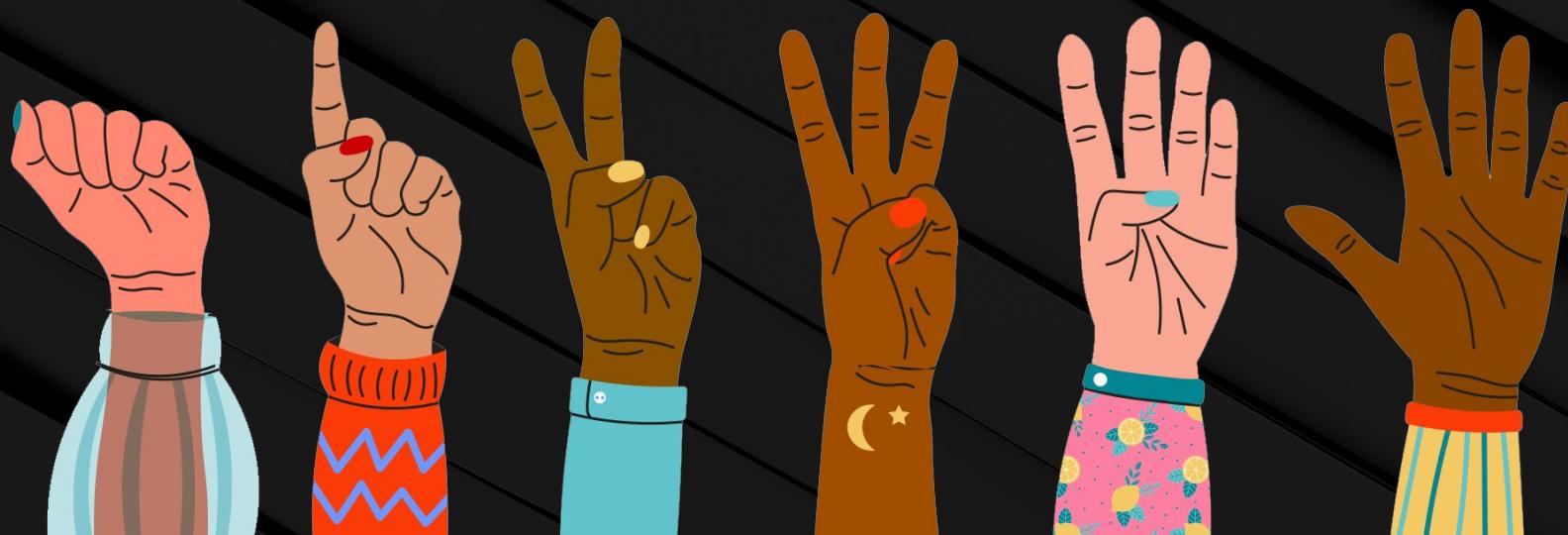
Storing data on AWS Simple Storage Service (Amazon S3)



Make sure you've downloaded
any relevant class files!

FIST TO FIVE:

How comfortable do you feel with this topic?



Creating an AWS Relational Database



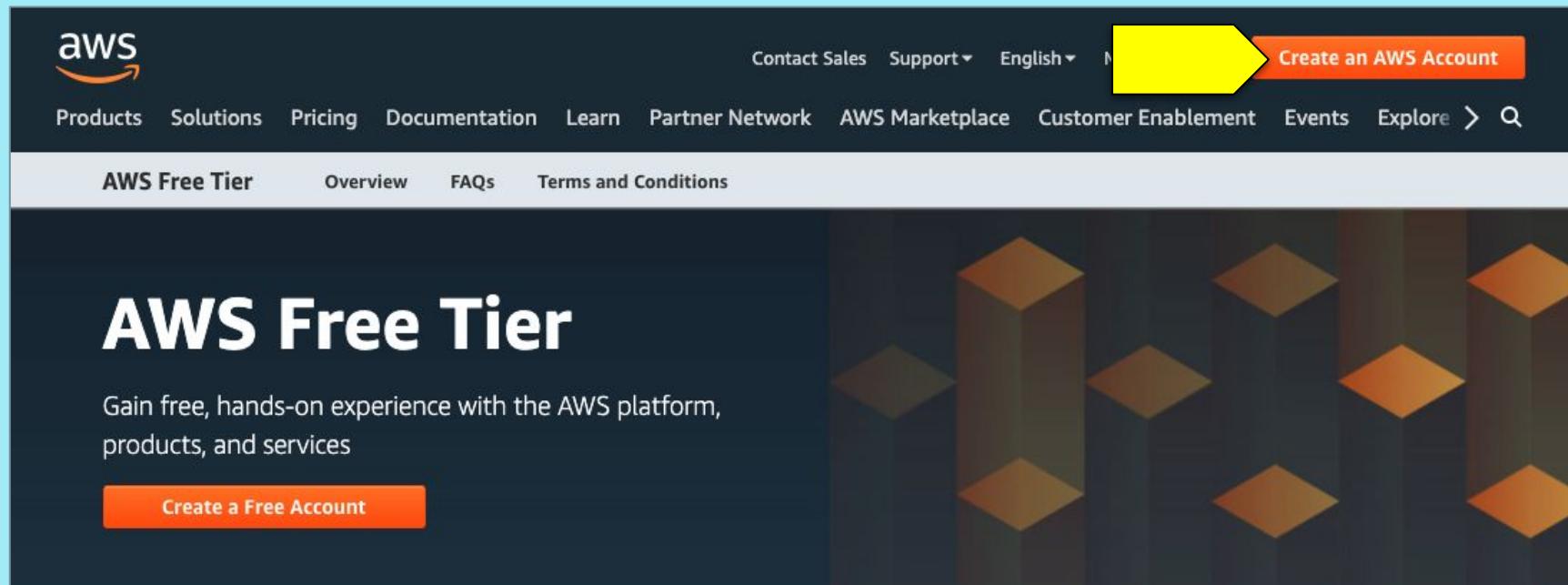
Create a PostgreSQL Database in RDS

Suggested Time:

15 minutes

Create a PostgreSQL Database in RDS

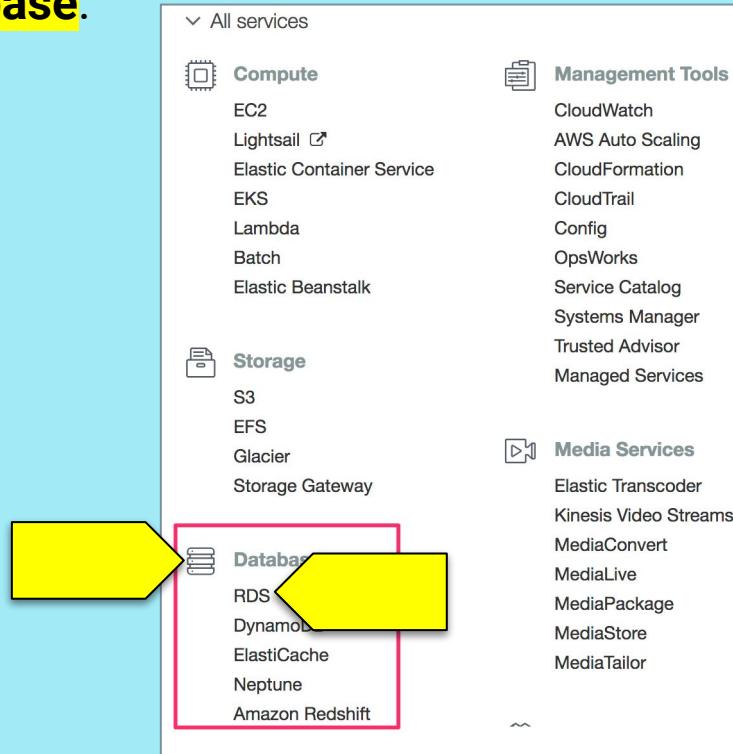
Create a Free Tier account.



The screenshot shows the AWS Free Tier landing page. At the top, there's a navigation bar with links for Contact Sales, Support, English, and a yellow button labeled "Create an AWS Account". Below the navigation bar, there are links for Products, Solutions, Pricing, Documentation, Learn, Partner Network, AWS Marketplace, Customer Enablement, Events, Explore, and a search icon. A yellow arrow points to the "Create an AWS Account" button. The main content area has a dark background with orange geometric shapes. It features the "AWS Free Tier" logo and a call-to-action button labeled "Create a Free Account". Below the button, text reads: "Gain free, hands-on experience with the AWS platform, products, and services".

PostgreSQL on RDS

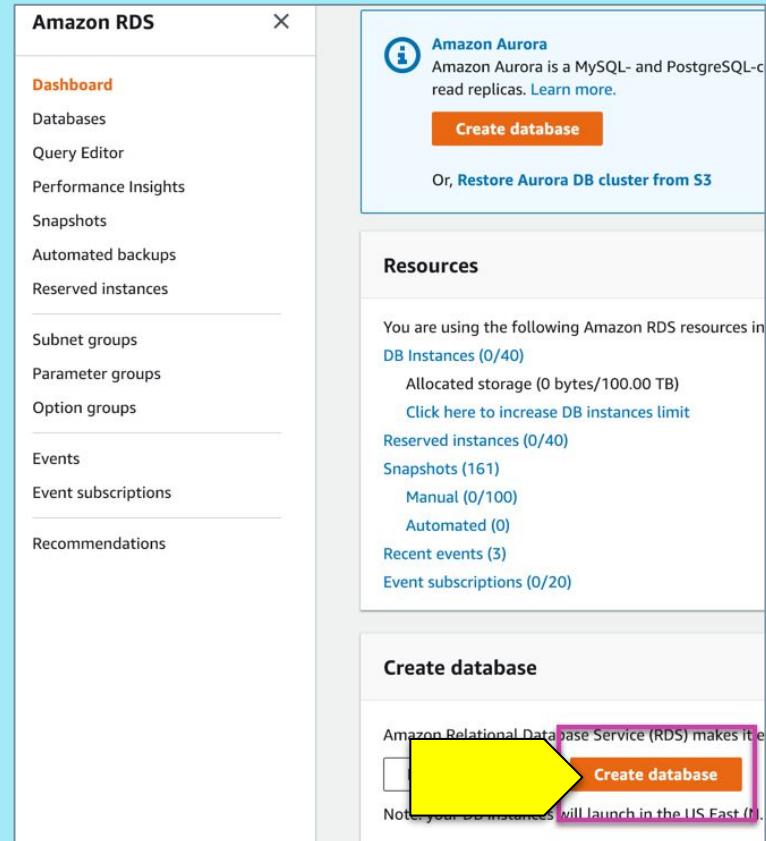
Start by logging in to the AWS Management Console, then navigate to the **RDS** section under **Database**.



PostgreSQL on RDS

Click **Create database** from the **Create database** section to the right.

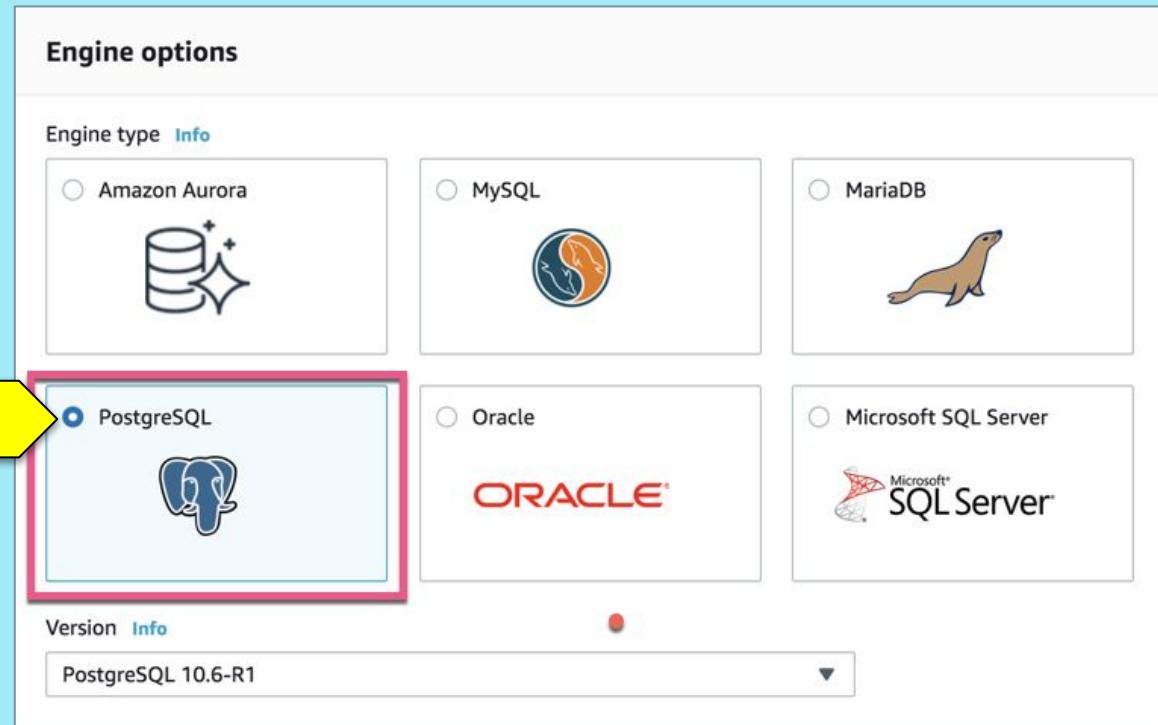
This button will take you to the **Engine options** page, which brings up a menu of different relational databases.



PostgreSQL on RDS

Check the box next to **Only enable options eligible for RDS Free Usage Tier** at the bottom of the menu.

Select PostgreSQL.



PostgreSQL on RDS

Under **Templates**, select **Free tier**.

Fill out the fields under **Settings**. Use **myPostgresDB** as the database instance identifier and **root** as the master username.

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.



Note: Although the database instance identifier and master username can take any name, we recommend sticking to these settings in this case for consistency.

PostgreSQL on RDS

Uncheck the box next to **Auto generate a password.**

Enter a password and be sure to record it somewhere.

The other settings will be accessible in the future, but the password will not.

Settings

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

myPostgresDB 1.

The DB instance identifier is case-insensitive, but is stored as all lowercase (as in "mydbinstance"). Constraints: 1 to 60 alphanumeric characters or hyphens (1 to 15 for SQL Server). 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.

root 2.

1 to 16 alphanumeric characters. First character must be a letter

Auto generate a password 3.
Amazon RDS can generate a password for you, or you can specify your own password

Master password [Info](#)

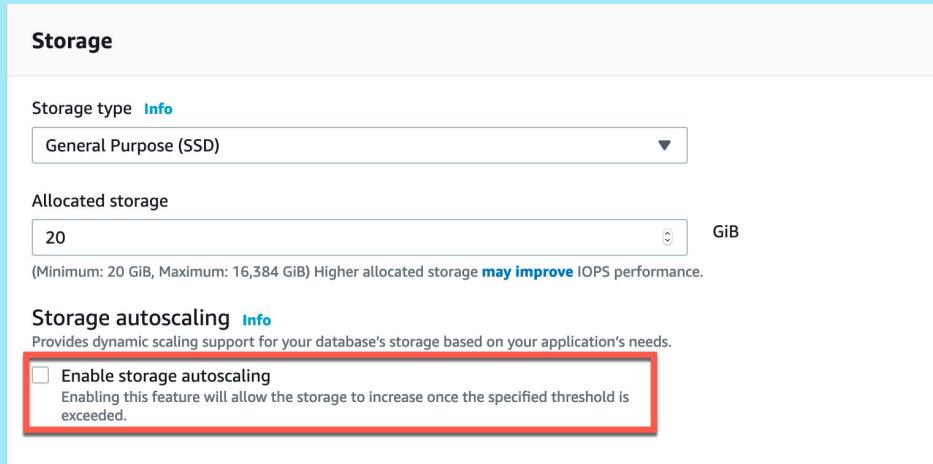
Constraints: At least 8 printable ASCII characters. Can't contain any of the following: / (slash), "(double quote)" and @ (at sign).

Confirm password [Info](#)
***** 4.

1.
2.
3.
4.

PostgreSQL on RDS - NEW

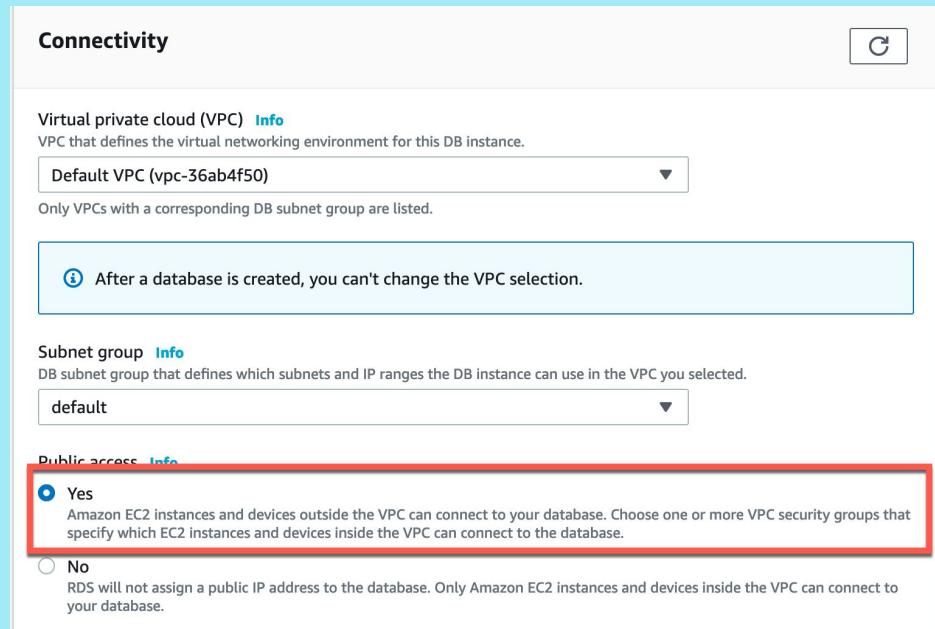
Under **Storage**, uncheck the box for **Enable storage autoscaling**.



PostgreSQL on RDS - New

Under **Connectivity**, select **Yes** under the **Public access** option.

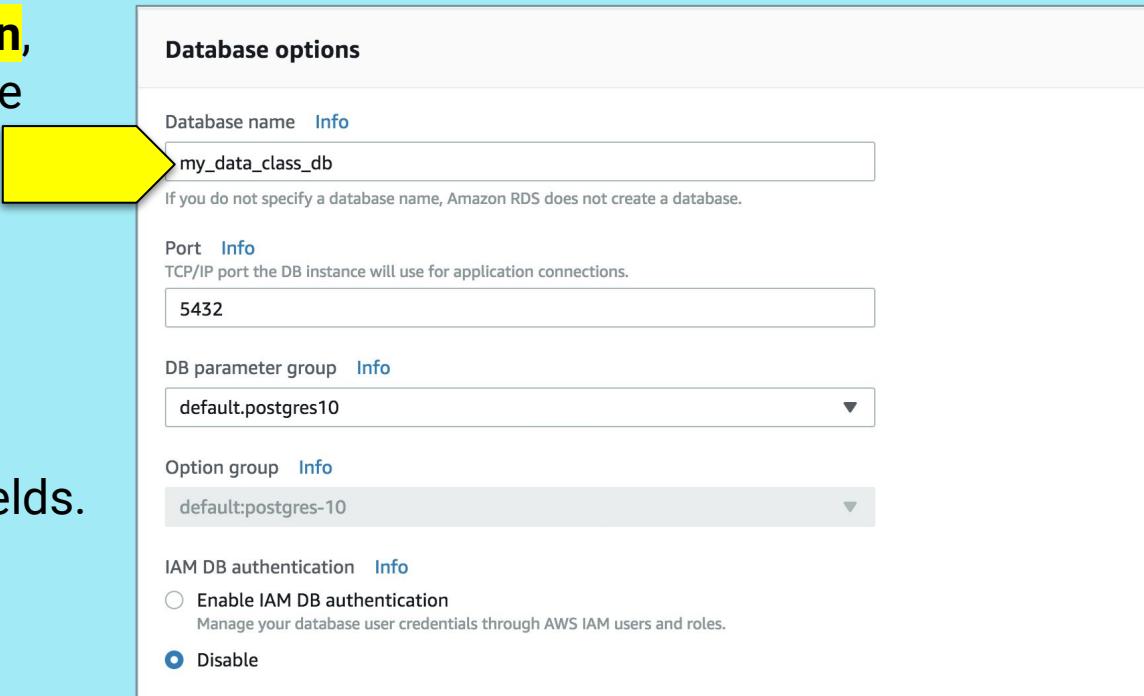
This does not mean anyone can access the database, as a password is still required, but it allows connections from outside sources like pgAdmin.



PostgreSQL on RDS

Under **Additional configuration**, click the down arrow and make the database name **my_data_class_db**.

Use this name for the sake of consistency. In the future, any name can be used. Keep the default settings in the other fields.



PostgreSQL on RDS

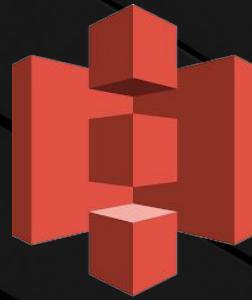
Uncheck the boxes for **Enable automatic backups**, **Enable performance insights**, and **Enable auto minor version upgrade**.

Click **Create Database** followed by **View DB Instance details** to navigate to the instance console page. The database creation on AWS's end will take anywhere from 10 to 15 minutes.

The screenshot shows the 'Configure' step of a PostgreSQL database creation wizard. Several checkboxes are unselected:

- Backup**: Enable automatic backups (Creates a point-in-time snapshot of your database)
- Performance Insights**: Enable Performance Insights
- Monitoring**: Enable Enhanced monitoring (Enabling Enhanced monitoring metrics are useful when you want to see how different processes or threads use the CPU)
- Log exports**:
 - Postgresql log
 - Upgrade log
- IAM role**: The following service-linked role is used for publishing logs to CloudWatch Logs.
RDS service-linked role
- Maintenance**:
 - Enable auto minor version upgrade (Enabling auto minor version upgrade will automatically upgrade to new minor versions as they are released. The automatic upgrades occur during the maintenance window for the database.)

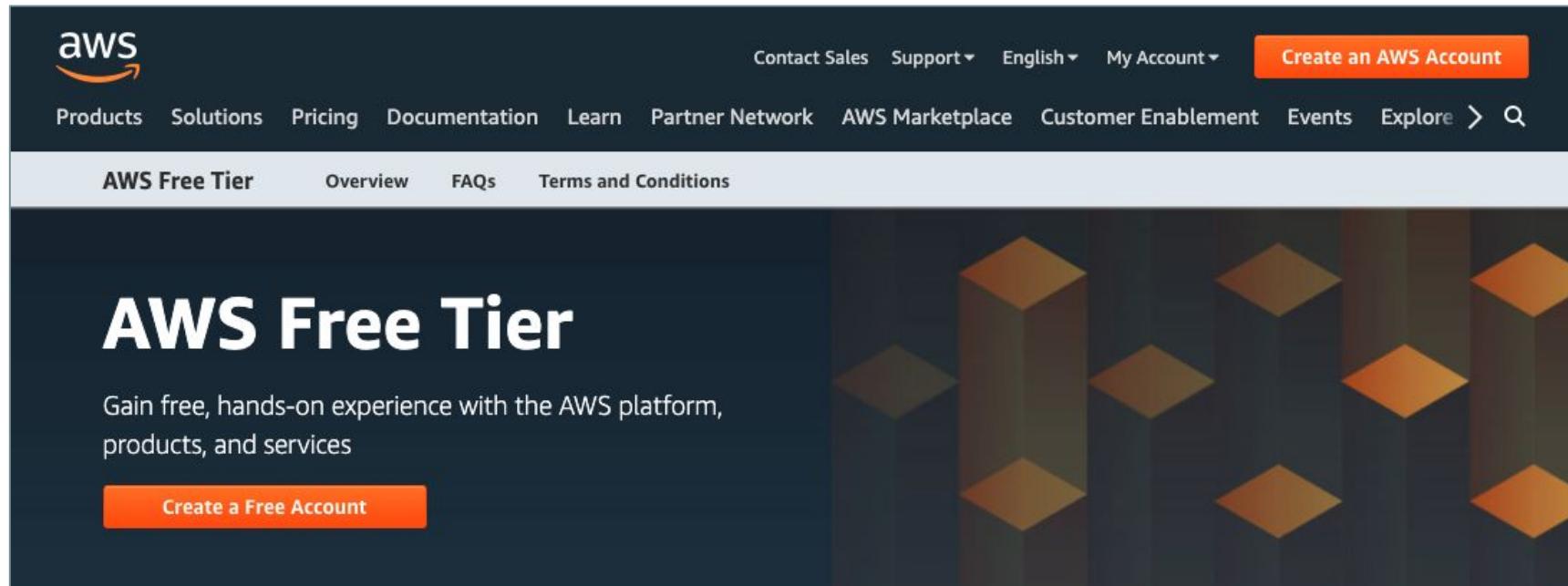
A callout box highlights the 'more' link under the maintenance section.



Storage with AWS S3

Storage with AWS S3

Simple Storage Service, or S3, is Amazon's cloud file-storage service that uses key-value pairs. Files are stored on multiple servers and have a high rate of availability.

A screenshot of the AWS Free Tier landing page. The top navigation bar includes the AWS logo, Contact Sales, Support, English, My Account, a Create an AWS Account button, and links for Products, Solutions, Pricing, Documentation, Learn, Partner Network, AWS Marketplace, Customer Enablement, Events, Explore, and a search icon. Below the navigation is a secondary navigation bar with links for AWS Free Tier (which is bolded), Overview, FAQs, and Terms and Conditions. The main content area features a large title "AWS Free Tier" in white. Below the title is a subtext: "Gain free, hands-on experience with the AWS platform, products, and services". At the bottom left is a red "Create a Free Account" button. The background of the main content area has a dark gradient with orange diamond shapes.

AWS Free Tier

Gain free, hands-on experience with the AWS platform, products, and services

Create a Free Account

Storage with AWS S3

S3 uses **buckets** to store files, which are similar to computer folders or directories. Buckets can contain additional folders and files. Each bucket must have a unique name.

S3 has fine-grained control over files, such as read and write permissions. Buckets can assign individual access or total public access.



The S3 bucket structure is somewhat similar to a GitHub repository, which also holds files and folders.



Storage with AWS S3

An S3 bucket can contain files, but it cannot contain another bucket.





**Amazon guarantees an uptime,
or availability, of over 99.99%
for S3 files**

**Each S3 bucket must have a URL
that is unique across AWS**

Questions?





Cloud Storage with S3

Suggested Time:

15 minutes

PostgreSQL on RDS Introduction

AWS is a cloud-computing platform provided by Amazon, offering over 175 different services via the cloud





Because the infrastructure is already in place, companies can easily scale up as needed without a huge up-front investment, and because they offer flexible usage options there is a cost-appropriate option for everyone.

RDS

RDS stands for Relational Database Service.

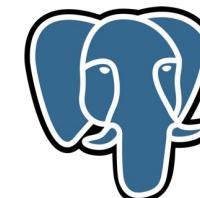
This is what Amazon uses to host a variety of relational databases in the cloud.

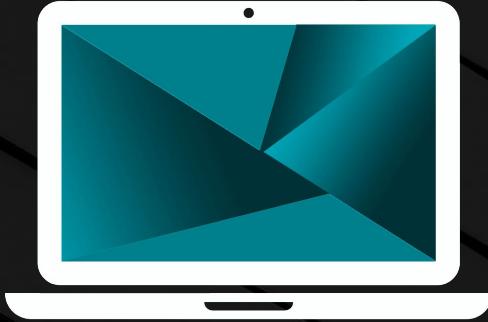




These databases can have different dialects, such as MySQL, PostgreSQL, and Amazon's own Aurora database.

PostgreSQL, usually referred to as "Postgres," is an object-relational database system that uses the SQL language.





Instructor Demonstration

PostgreSQL on RDS

PostgreSQL on RDS

The **Summary** section shows what kind of database the instance is and whether it is available.

Summary			
Engine PostgreSQL 10.4	DB instance class Info db.t2.micro	DB instance status creating	Pending maintenance none

PostgreSQL on RDS

The **Connectivity** tab lists the endpoint, port, and security groups associated with the instance. The endpoint will be used to connect to the database.

Connectivity

Monitoring

Logs & events

Connectivity

Endpoint & port

Endpoint

mypostgresdb.cae1r8ifpdhe.us-east-1.rds.amazonaws.com

Port

5432



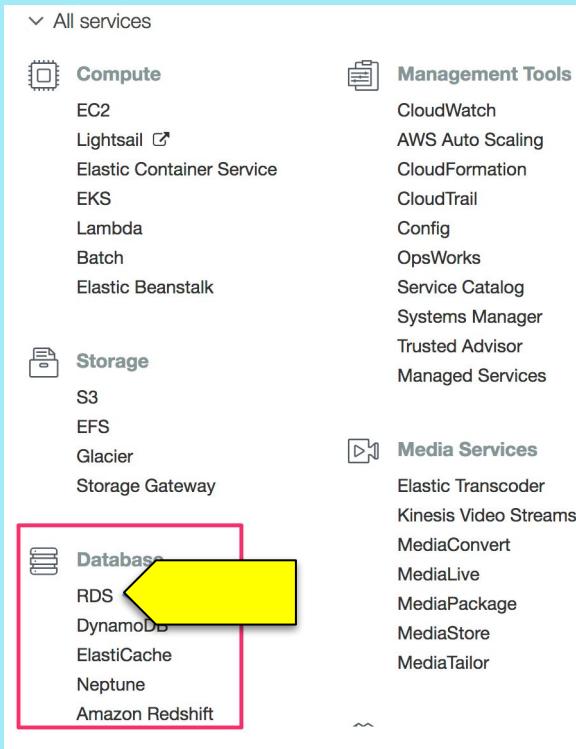
RDS PostgreSQL and pgAdmin with CRUD

Suggested Time:

20 minutes

RDS PostgreSQL and pgAdmin with CRUD

Log in to the AWS console, then navigate to **RDS** under **Database**.



RDS PostgreSQL and pgAdmin with CRUD

Navigate to **DB Instances** in the **Resources** section to the right.

The screenshot shows the Amazon RDS console interface. On the left, there is a sidebar with the following menu items:

- Dashboard
- Databases
- Query Editor
- Performance Insights
- Snapshots
- Automated backups
- Reserved instances
- Subnet groups
- Parameter groups
- Option groups

On the right, there is a main content area titled "Resources". It displays the following information:

You are using the following Amazon RDS resources:

- DB Instances (1/40)** (highlighted with a yellow arrow)
- Allocated storage (20.00 GB/100.00 TB)
- [Click here to increase DB instances limit](#)
- Reserved instances (0/40)
- Snapshots (161)
 - Manual (0/100)
 - Automated (0)
- Recent events (4)
- Event subscriptions (0/20)

RDS PostgreSQL and pgAdmin with CRUD

Go to the database created earlier, mypostgresdb.

Navigate to the **Security group rules** section on the right:

Connect

Endpoint mydbinstance.cae1r8ifpdhe.us-east-1.rds.amazonaws.com	Port 5432	Publicly accessible Yes
-------------------------------------------------------------------	--------------	----------------------------

Security group rules (2) 

Security group	Type	Rule
rds-launch-wizard-6 (sg-017e797f489620ed7)	CIDR/IP - Inbound	24.38.223.97/32
rds-launch-wizard-6 (sg-017e797f489620ed7)	CIDR/IP - Outbound	0.0.0.0/0

RDS PostgreSQL and pgAdmin with CRUD

Click the security group for type **CIDR/IP - Inbound**.

Connect

Endpoint mydbinstance.cae1r8ifpdhe.us-east-1.rds.amazonaws.com	Port 5432	Publicly accessible Yes
-------------------------------------------------------------------	--------------	----------------------------

Security group rules (2)

Security group	Type	Rule
rds-launch-wizard-6 (sg-017e797f489620ed7)	CIDR/IP - Inbound	24.38.223.97/32
rds-launch-wizard-6 (sg-017e797f489620ed7)	CIDR/IP - Outbound	0.0.0.0/0

RDS PostgreSQL and pgAdmin with CRUD

From the management console, navigate to the **Inbound rules** tab on the bottom part of the screen, and then click **Edit inbound rules**. This will bring up a menu to set rules for the security group.

The screenshot shows the AWS RDS Security Groups page. At the top, there are tabs: Details, Inbound rules (which is highlighted in orange), Outbound rules, and Tags. Below the tabs, a red arrow points to the 'Inbound rules' tab with the text '1. Select Inbound rules tab'. The main area displays a table titled 'Inbound rules (1)'. The table has columns: Type, Protocol, Port range, Source, and Description - optional. One row is shown: Type is 'PostgreSQL', Protocol is 'TCP', Port range is '5432', Source is '71.104.22.245/32', and Description is '-'. To the right of the table is a button labeled 'Edit inbound rules', which is also highlighted with a red arrow and the text '2. Click Edit inbound rules'.

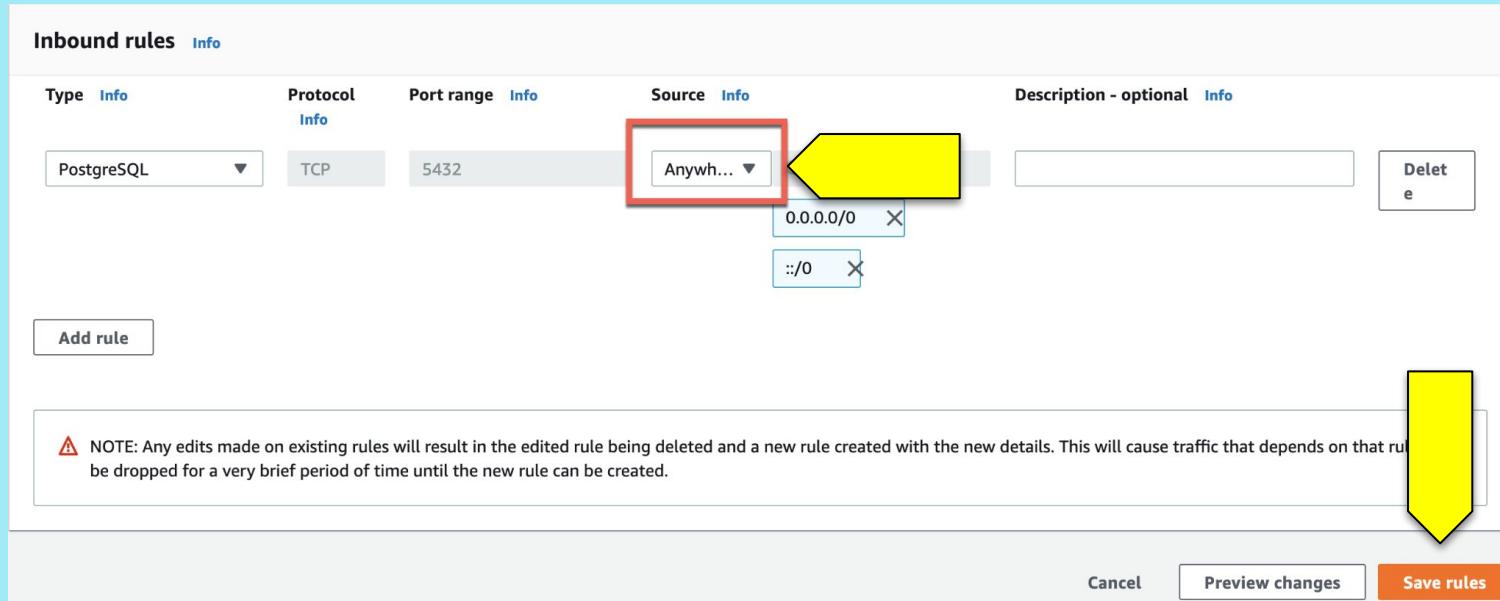
Type	Protocol	Port range	Source	Description - optional
PostgreSQL	TCP	5432	71.104.22.245/32	-

1. Select Inbound rules tab

2. Click Edit inbound rules

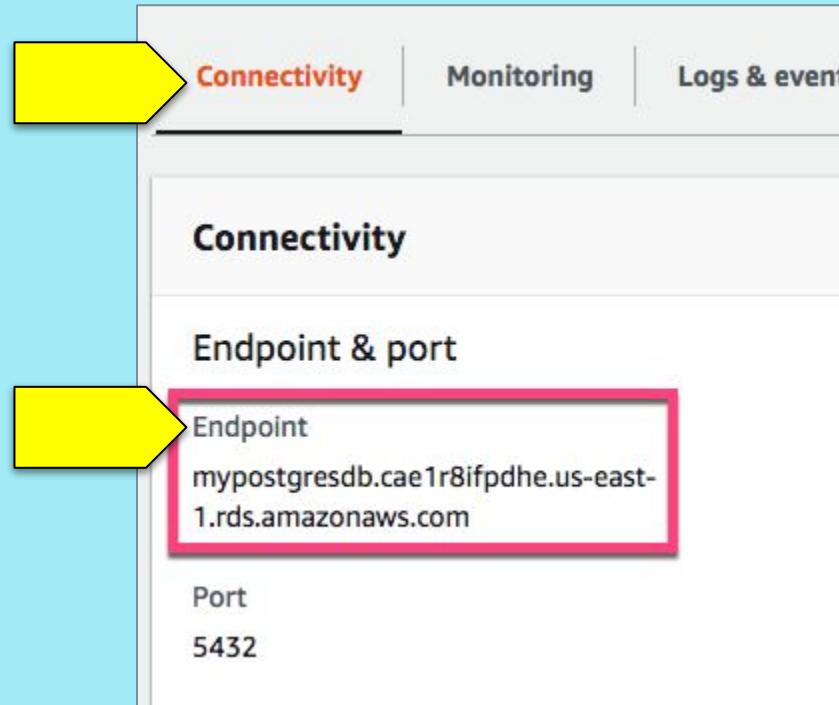
RDS PostgreSQL and pgAdmin with CRUD

Change the **Source** to **Anywhere**, then click **Save rules**. The RDS instance will now accept a connection from anywhere. This isn't completely open to the world because the endpoint, username, and password are still needed to connect.



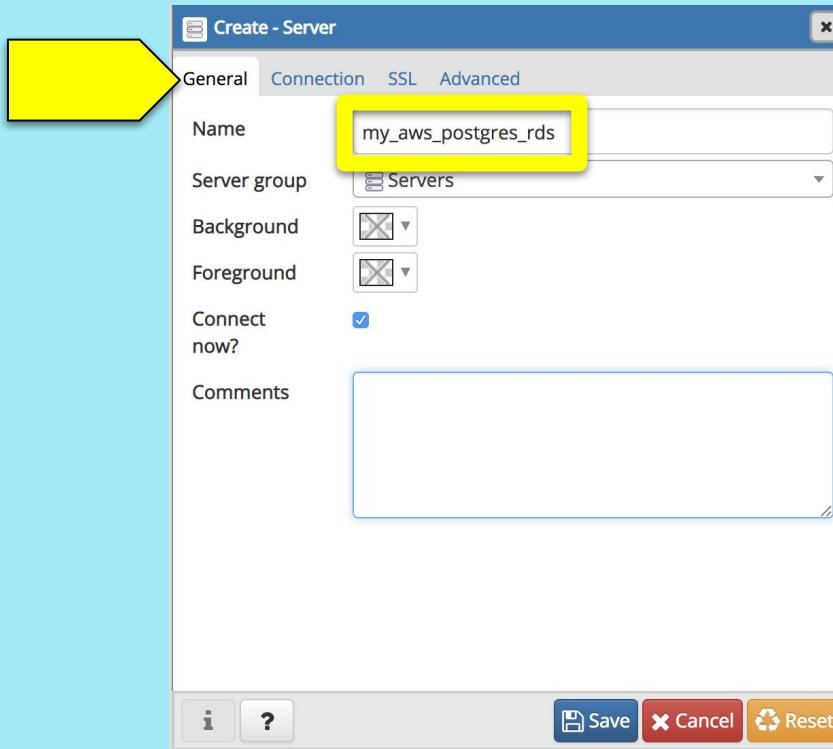
RDS PostgreSQL and pgAdmin with CRUD

Navigate back to the instance console and have the class find the endpoint, which is found in the **Connectivity** tab.



RDS PostgreSQL and pgAdmin with CRUD

Under the **General** tab, enter the server name as **my_aws_postgres_rds**.



RDS PostgreSQL and pgAdmin with CRUD

Under the **Connection** tab, do the following:



Enter the endpoint in the **Host name/address** field. This is unique to the instance.



Enter **postgres** in the **Maintenance database** field. This is the default for all Postgres RDS instances.



Enter the username **root** in the **Username** field.



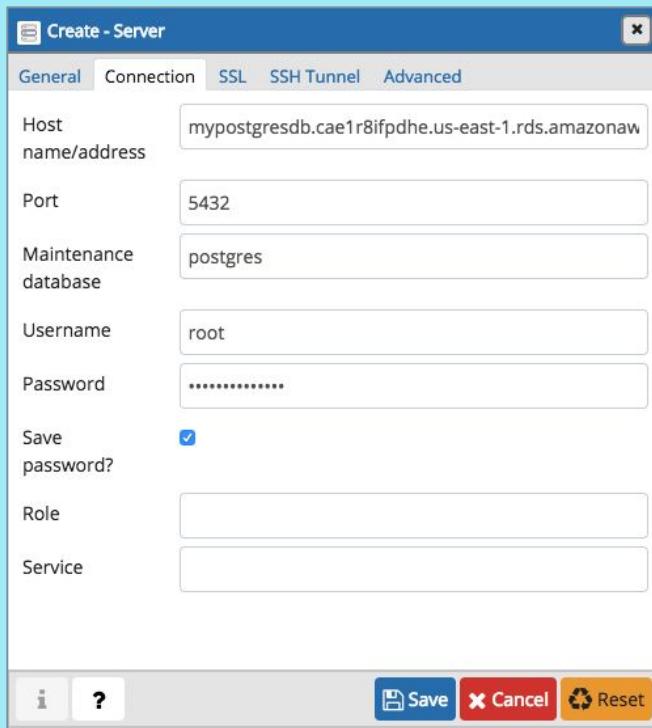
Enter the password that was created for your RDS instance.



Check the box next to **Save password**.

RDS PostgreSQL and pgAdmin with CRUD

Click **Save**. If all information is entered correctly, this will set up the connection and not return an error.



RDS PostgreSQL and pgAdmin with CRUD

The four basic functions of persistent data storage are create, read, update, and delete (CRUD).

Create	INSERT table info (column1, column2, column3)
Read	SELECT * FROM table
Update	UPDATE table SET column1 = VALUE WHERE id = 1
Delete	DELETE FROM table WHERE id = 4

RDS PostgreSQL and pgAdmin with CRUD

This schema consists of the first part of CRUD, create.

Create	INSERT table info (column1, column2, column3)
Read	SELECT * FROM table
Update	UPDATE table SET column1 = VALUE WHERE id = 1
Delete	DELETE FROM table WHERE id = 4

RDS PostgreSQL and pgAdmin with CRUD



The schema will create the tables. The insertion creates the data.



A foreign key is used in the `patients` table to reference the `doctor` table.



Create a new database named `medical`, open a query tool, and then run the schema. This creates two tables and uploads the data.

RDS PostgreSQL and pgAdmin with CRUD

The read functions of a database are run with **SELECT** statements.

```
-- Read tables
```

```
SELECT * FROM doctors;
```

```
SELECT * FROM patients;
```

RDS PostgreSQL and pgAdmin with CRUD

An error will occur after running the first **INSERT**.

This is because the `doctor_id` key 22 does not exist in the `doctor` table.

```
-- Inserting with invalid foreign key
INSERT INTO patients(id, doctor_id, health_status)
VALUES
(6, 22, 'sick');
```

RDS PostgreSQL and pgAdmin with CRUD

The second **INSERT** statement will run because the foreign key is located in the doctor table.

```
-- Inserting with invalid foreign key
INSERT INTO patients(id, doctor_id, health_status)
VALUES
(6, 22, 'sick');
```

RDS PostgreSQL and pgAdmin with CRUD

The update functions are run with UPDATE.

```
-- Update rows
UPDATE doctors
SET taking_patients = FALSE
WHERE id = 1;

UPDATE patients
SET health_status = 'healthy'
WHERE id = 1;
```

RDS PostgreSQL and pgAdmin with CRUD

The delete functions are run with **DELETE**.

```
-- Delete row  
DELETE FROM patients  
WHERE id = 1;
```

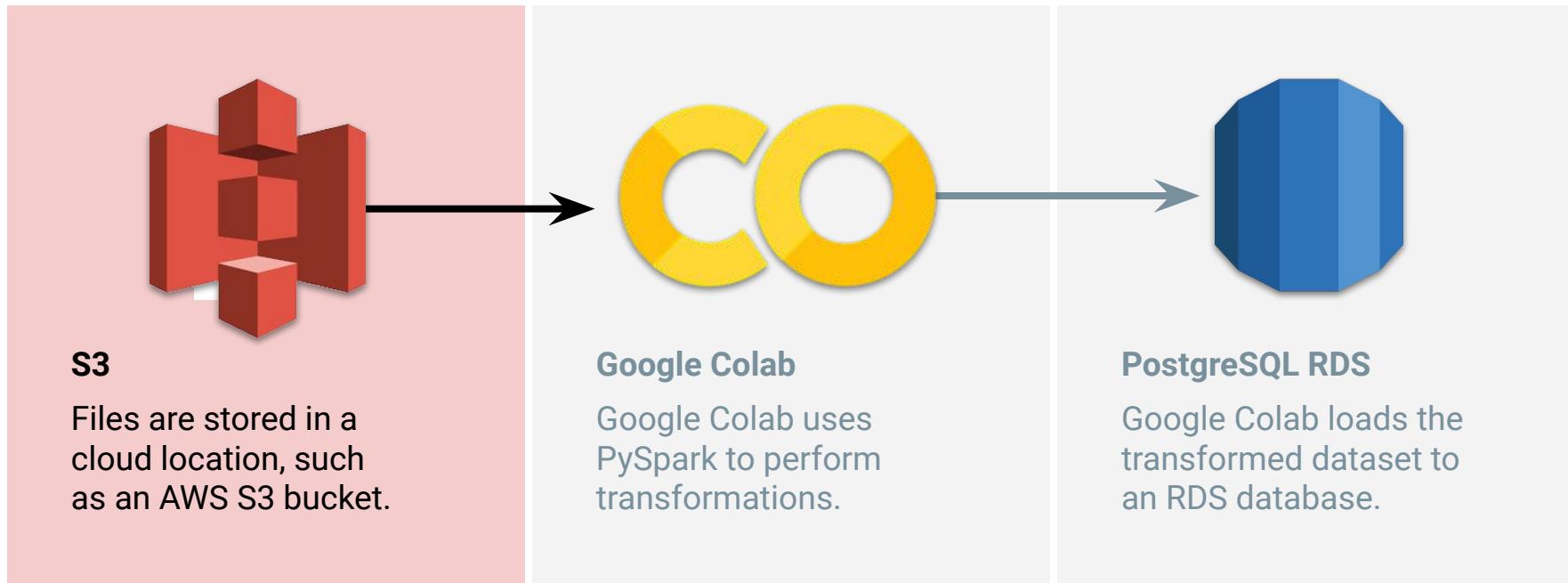
Questions?



ETL with AWS

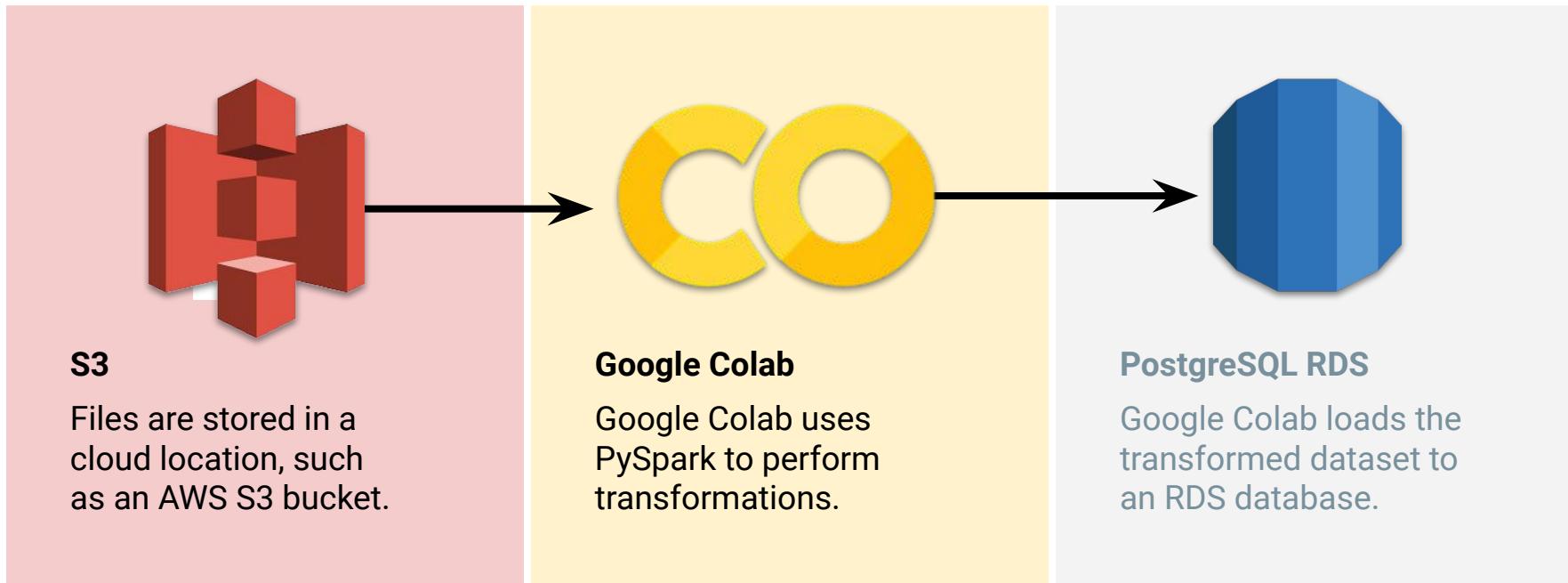
ETL with AWS

Files are stored in a cloud location such as an AWS S3 bucket. These files are extracted from S3 and read into PySpark DataFrames using Google Colab.



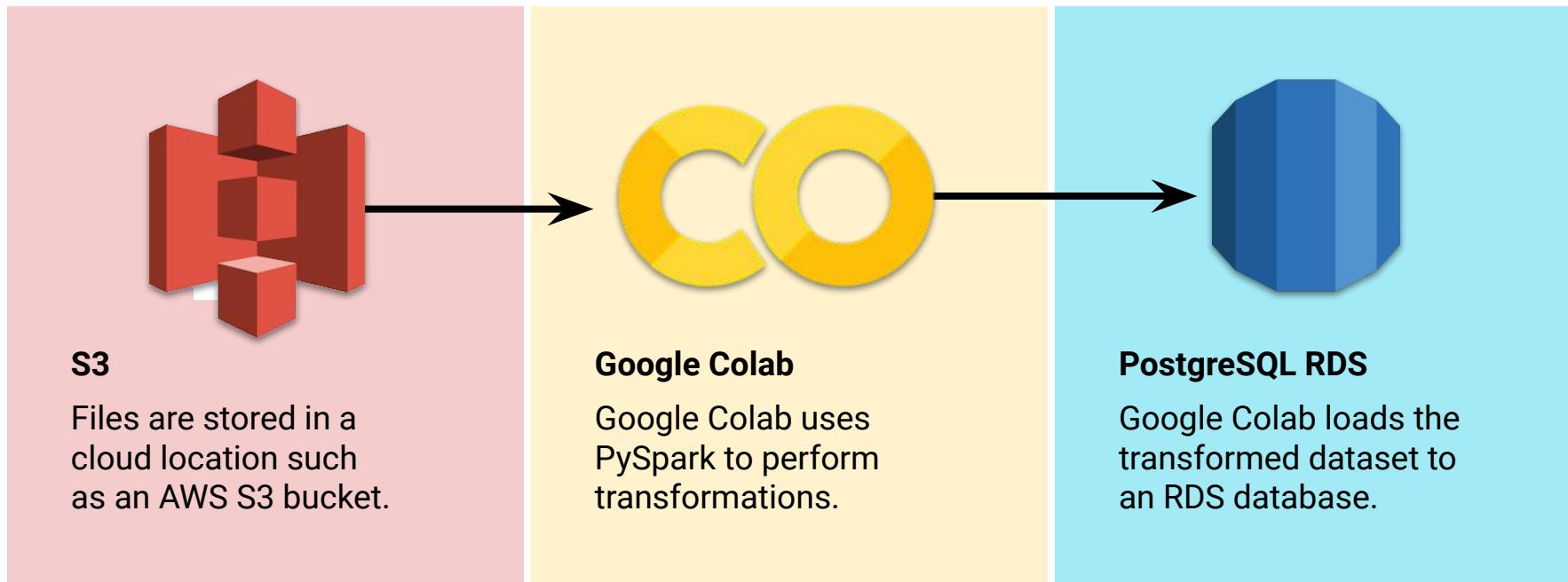
ETL with AWS

Once the files are extracted into Google Colab, transformations can take place. PySpark is used to transform the data.



ETL with AWS

After the data is transformed, Google Colab creates a connection to an RDS instance. Once connected, Google Colab loads the DataFrame into the RDS database.





Instructor Demonstration

ETL with S3, PySpark, and RDS



Activity: ETL with S3, PySpark, and RDS

As the sole data person at your new company, you have been tasked with cleaning the data from an Excel spreadsheet—which has been exported as a CSV—and creating a SQL database with this data. In other words, you will be performing ETL.

Suggested Time:
15 minutes



ETL with S3, PySpark, and RDS

Take a moment to review the SQL table schema, which reflects the requirements for the company's new database. Your company will be using S3 for file storage and RDS to host SQL databases.

Then, complete the following tasks:



Upload the CSV file to S3, and be sure to make the S3 bucket public.



Use Spark on Colab to clean and transform the data.



Use pgAdmin to create the table schema in RDS.



Load the data from Pandas DataFrames into RDS.



Let's Review

AWS Cleanup



Everything we have done today will fall under the AWS Free Tier. However, as a precaution, we will delete everything we created.



AWS Cleanup

Suggested Time:

15 minutes

AWS Cleanup

Log in to the AWS management console and navigate to the RDS dashboard.
Click **DB Instances**.

Resources

You are using the following Amazon RDS resources in the US East (N. Virginia) region (used/quota)

DB Instances (1/40)	Parameter groups (3)
Allocated storage (20.00 GB/100.00 TB)	Default (3)
Click here to increase DB instances limit	Custom (0/100)
Reserved instances (0/40)	Option groups (3)
Snapshots (146)	Default (3)
Manual (4/100)	Custom (0/20)
Automated (0)	Subnet groups (2/50)
Recent events (0)	Supported platforms VPC
Event subscriptions (0/20)	Default network vpc-36ab4f50

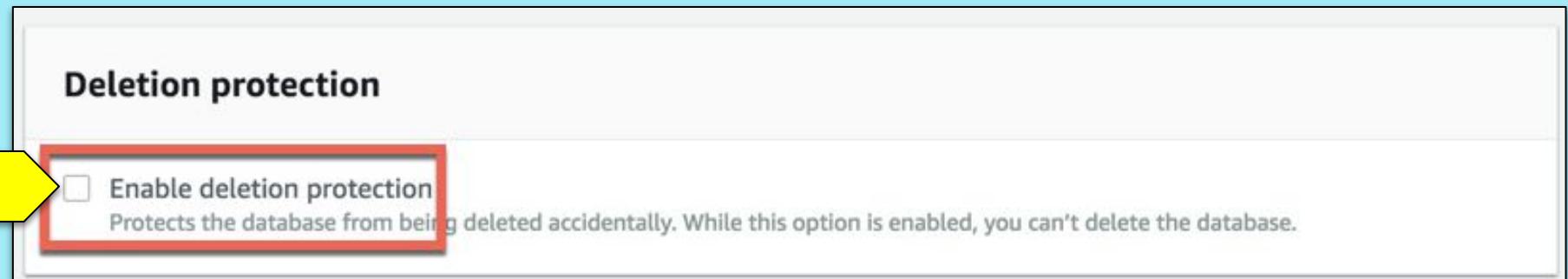
AWS Cleanup

Select our database under **DB Name**, then click **Modify**.

Databases			
<input type="checkbox"/> Group resources			
<input type="text"/> Filter databases			
<input type="checkbox"/>	DB Name	▲	Role ▼ Engine
<input checked="" type="radio"/>	mypostgresdb		Instance Postgre

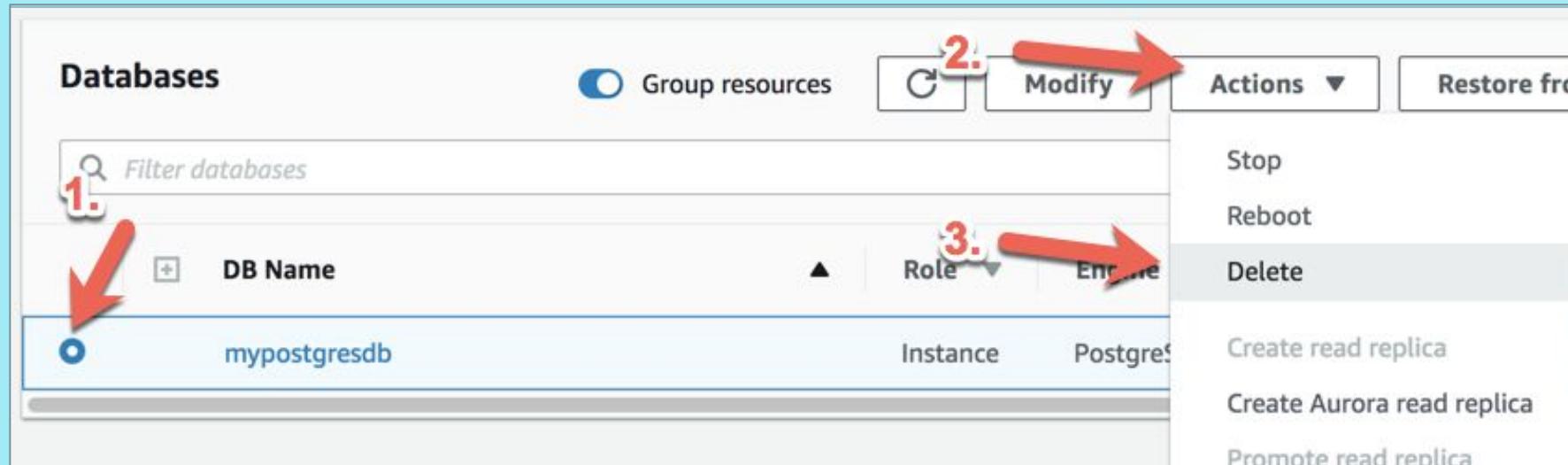
AWS Cleanup

Scroll down to **Deletion protection**. If the box next to **Enable deletion protection** is selected, uncheck the box. Then, click **Continue, Apply immediately**, and **Modify DB Instance**. If the box is already empty, return back to the instance page.



AWS Cleanup

Next, on the database dashboard, make sure the database is selected, and then click **Actions** followed by **Delete**.

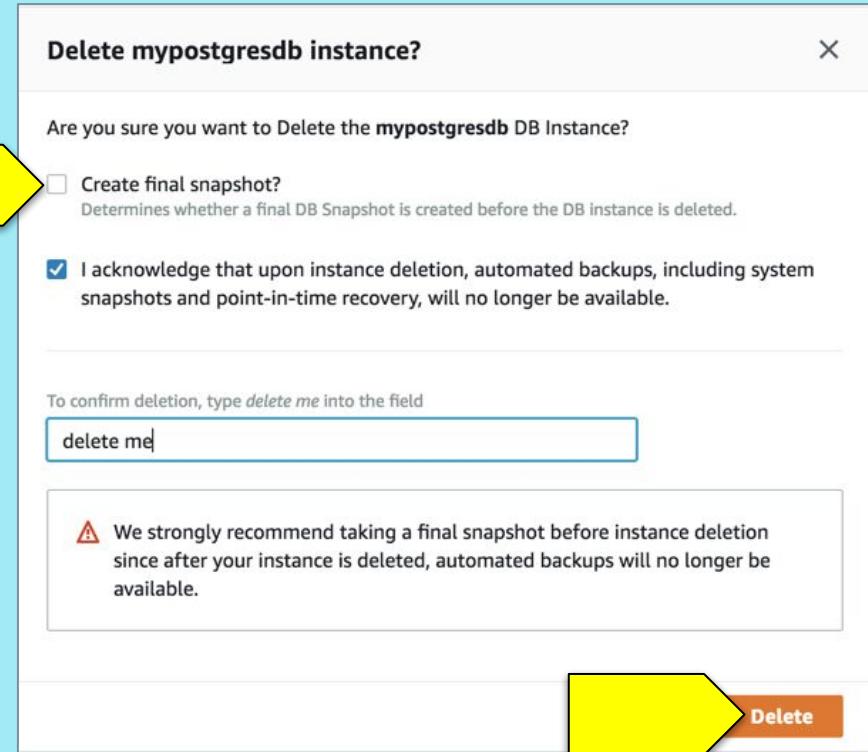


AWS Cleanup

Important: Uncheck **Create final snapshot?**, then check the acknowledgement box.

Type delete me, then click Delete.

If you do not uncheck this box, your databases will create a backup that could accrue additional costs, so be sure not to skip over this step.





This will take a few
minutes to fully delete.

AWS Cleanup

Check the box next to the bucket you want to remove files from, then click **Empty**.

The screenshot shows the AWS S3 Buckets list interface. On the left, there is a sidebar with the text "Buckets (9)". Below it, a descriptive text states: "Buckets are containers for data stored in S3. [Learn more](#)". To the right, there is a list of buckets. At the top of this list is a horizontal toolbar with several buttons: a "C" icon, a "Copy ARN" button, an "Empty" button (which is highlighted with a red box), a "Delete" button, and an "Create bucket" button.

AWS Cleanup

Type **permanently delete** in the text input field, then click **Empty**.

Permanently delete all objects in bucket "a-data-test-bucket"?

To confirm deletion, type *permanently delete* in the text input field.

permanently delete



AWS Cleanup

Select the circle next to the bucket you want to delete, then click **Delete**.

The screenshot shows the AWS S3 Buckets list interface. At the top, there are buttons for Refresh, Copy ARN, Empty, Delete, and Create bucket. Below that is a search bar labeled "Find buckets by name". The main table lists 9 buckets. The first bucket, "a-data-test-bucket", has a blue circular selection button to its left, which is highlighted with a red arrow labeled "1.". The "Delete" button in the top right corner is also highlighted with a red arrow labeled "2.". The table columns are Name, AWS Region, Access, and Creation date.

Name	AWS Region	Access	Creation date
a-data-test-bucket	US East (N. Virginia) us-east-1	Bucket and objects not public	March 22, 2021, 15:01:23 (UTC-04:00)

AWS Cleanup

Type the name of the bucket, then click **Delete bucket**.

Delete bucket "a-data-test-bucket"?

To confirm deletion, enter the name of the bucket in the text input field.

 **Delete bucket**

Questions?

