Lecture 4: Cloud Computing Services and Setup

Contents

4.1. Announcements

- If you still want to know more about UNIX, please watch this video.

4.2. Learning objectives

- Identify the features and functions of commonly used AWS services.
- Compare the major services offered by AWS.
- Access and navigate to commonly used AWS services.
- Apply the concepts learned in the last class to a practical scenario.
- Configure an EC2 instance for use within your team.
- Create an S3 bucket and gain proficiency in interacting with it.

4.3. Refresher questions

- What is cloud computing, why is it so important, and how does it work?
- Can you give me a list of various cloud service providers?
- Have you heard of a Hypervisor? What is the function of a hypervisor in cloud computing ?
- If you were to start a business in the UK that uses cloud services, what are some of the factors that you would consider?
- How would not having VPCs and subnets impact the system?
- What steps are necessary to set up high availability systems?
- What is the purpose of private subnets, and how do you establish them?
- What are some different ways to interact with AWS? Do you know how to configure AWS CLI?

Answers:

- What is cloud computing, why is it so important, and how does it work?

Cloud Computing is when we get a virtual pool of resources in the cloud for our compute, storage, databases, and network services that are provided to users via the internet. It is important because it provides on-demand access to these resources, reducing the need for physical infrastructure and lowering costs. Cloud computing works by the principle of virtualization.

- Can you give me a list of various cloud service providers?

Amazon Web Services (AWS), Microsoft Azure, Google Cloud Platform, etc...

- Have you heard of a Hypervisor? What is the function of a hypervisor in cloud computing?

Yes, a hypervisor is a software layer that enables multiple virtual machines to run on a single physical machine. As an AWS user, it is invisible to us.

- If you were to start a business in the UK that uses cloud services, what are some of the factors that you would consider?

Security and compliance requirements of the business. Do they have a requirement for storing all the data and servers within the country? We set up the servers and storage in the same region to decrease the network delay.

- How would not having VPCs and subnets impact the system?

VPCs provide a private network within the cloud infrastructure, allowing resources to be isolated and secured. Subnets provide a way to partition the VPC into smaller networks, enabling finer control. VPCs are like building a house, and subnets are like rooms in the house. If we don't have VPCs and subnets, we will not be able to isolate the resources, and it will be difficult to manage and scale. Check the example from the last lecture.

- What steps are necessary to set up high availability systems?

From what we learned, we can use multiple availability zones. We can replicate data and servers across multiple availability zones. So if something happens with one AZ, there is other as backup. There are other things also we can consider, like using load balancers to distribute traffic and using automated failover and recovery.

- What is the purpose of private subnets, and how do you establish them?

Private subnets are used to provide an additional layer of security by isolating resources within the VPC by not having access to the public internet. To establish a private subnet, you would need to create a subnet with no internet gateway which provides access to the internet.

- What are some different ways to interact with AWS? Do you know how to configure AWS CLI?

You can interact with AWS using the AWS Management Console, AWS Command Line Interface (CLI), and AWS Software Development Kits (SDKs). We saw how to configure AWS CLI in the last lecture.

4.4. Video Zone

Here is the video helper for week2. The reason why this is here because some of you will be having lab sessions before the lectures. So, you can use this as a reference. You will see all below topics bookmarked in the video.

- AWS lab setup (lecture 3)
- Setting up AWS CLI (lecture 3)
- Setting up an EC2 instance (lecture 4)
- Logging into EC2 instance (lecture 4)
- Setting up a common space in EC2 (lecture 4)
- Setting up Jupyter notebook on EC2 (lecture 4)
- Installing packages in jupyter notebook(lecture 4)
- Setting up S3 bucket (lecture 4)

OPTIONAL: Following is a short video you can check for a quick refresher in UNIX commands.

4.5. Category of services in AWS

Here are the main category of services available in AWS;

- Compute
- Storage
- Database
- Networking and Content Delivery (we already discussed about VPC, subnets etc in last class)
- Security, Identity, and Compliance (we already discussed about IAM in last class)

Having an idea of different cloud services is essential in deciding the most suitable solution for your business. While it's impossible to cover every category and service within AWS, we will be focusing on the fundamental and commonly used services that serve as the building blocks for most AWS services. Every year they add around 10 new services, so again the most important skill is to adapt to new technologies and tools.

For a comprehensive list of services and their details, please refer to this link. Alternatively, for a simplified version of the list with brief descriptions, you may visit this link. Please note that the latter article is from 2020.

4.6. Compute (EC2)

A web service that provides secure, resizable compute capacity in the cloud. It is designed to make ***web-scale computing*** easier for developers.

4.6.1. The littlest JupyterHub (TLJH)

We are all using JupyterLab on our personal computers, but when we move to the industry, things change, and you will mostly use JupyterHub, which is a hub of Jupyter where employees log in with their username and password. We have a

similar setup here at UBC known as https://ubc.syzygy.ca/. In our example, we will set up a JupyterHub so that your team members can use it to log in and create a collaborative environment. This entire JupyterHub setup is made easy by TLJH, which we will be using. TLJH...

- Greatly simplifies the deployment process.
- Enables a much faster setup/teardown.
- Provides the same ability to customize authentication, etc.
- Defines the user environment by installing packages as admin.

4.6.2. Setting up an EC2 instance

This topic is also included in the Week 2 video. Look at the beginning of this lecture for the video zone.

You can use this doc as a reference for setup. For minimal setup instructions CLICK THE TOGGLE below.

1. (Name and tags) Give your instance a name.
2. (Application and OS Images (Amazon Machine Image))Choose an Amazon Machine Image (AMI). 2.1 Make sure you select `Ubuntu Server 22.04 LTS (HVM), SSD Volume Type - ami-XXXXXXXXXXXXXXXXX`, leaving 64-bit (x86) toggled.
3. (Instance type) Choose an Instance Type.
4. (Key pair (login)) select “Create a new key pair” if setting up the cluster for the first time, and give it a name. Download the private key and keep it secure. Next time, select “Choose an existing keypair” and choose the key you already created.
5. (Network Settings) Make sure to select the default Virtual Private Cloud (VPC) and subnet.
6. (Network Settings) Select or create a security group. When you first setup create a security group 6.1. Make sure you check following options.
    - Allow SSH traffic from Anywhere
    - Allow HTTPS traffic from the internet
    - Allow HTTP traffic from the internet
7. (Configure storage) Use Root as the storage with the desired size (30 GB size is perfect). Make sure to select “General purpose SSD (gp2)” as the Volume Type to save costs.
8. (Under Advanced details) Get the configuration code from step 7 in the above link and replace “admin-user-name” (remove < > as well) with your jupyter username. Example this is how my `EC2 user data` looks like (I replaced to “gittu”. Once my hub is setup I am going to use this username to login to my jupyterhub.

```
#!/bin/bash
curl -L https://tljh.jupyter.org/bootstrap.py \
  | sudo python3 - \
    --admin gittu
```

9. Search for your EC2 instance under instances to see if it’s running. Give it 15 - 20 minutes as it takes time to set up JupyterHub even if it shows running. You can also check system logs during this time.
10. Check the ***“Connect”*** button to determine how to connect to the instance. Select the `SSH Client` tab and follow the instructions there.

You are now given the DOOR access to the server mentioned in the first class. :)

While you set it up check if you can answer following questions;

Thoughts/Discussion?

- What are Amazon Machine Images (AMIs), and which one did we select for our exercise?
- What are the differences between community AMIs and marketplace AMIs?
- What are instances, which one did we select, and what are the CPU cores and memory specifications for the instance?
- What is EC2 user data, why is it used, and why did we use it? For more details on this topic, please check here.
- We have not discussed the details of storage, but which storage did we use?
- Which rules did we add to our security group, and what is the name of the security group we created?
- What is the name of your key pair?

4.6.3. Interacting with the EC2 instance

This topic is also included in the Week 2 video. Look at the beginning of this lecture for the video zone.

Now we have our EC2 instance ready let's see some things that we can do;

- Stop
- Terminate
- Start
- System logs
- Create an image of your instance.
- Create a template of your instance.
- Scale up your instance.

Assuming someone from your team has set up this EC2 instance, there are several questions we need to consider. In the following three sections, you will find answers to these questions:

- How can you and your team members log in to the EC2 instance?- Logging into EC2
- How can you make directories accessible to your team members?- Setting up a common space in EC2
- How can team members log in to the JupyterHub? -Setup your JupyterHub

While I won't be covering these topics now, there are instructions and video clips available for assistance. You can click the toggle below each section for more information.

4.6.3.1. Logging into EC2

This topic is also included in the Week 2 video. Look at the beginning of this lecture for the video zone.

Log in to your ec2 instance OR ssh into ec2 instance OR tunnel into ec2 instance - You can hear a variety of ways people say this.

Click the toggle below for more information.

The person who sets up the EC2 instance (gittu in this case) will only have access to the server, as that person will have the private key. Actually, Gittu has the private key of the root user `ubuntu` (the one I created when setting up EC2 instance and downloaded at last stage) and is logging into the EC2 instance as `ubuntu`. If you have not yet logged into the EC2 instance, please check out the last step in the EC2 creation process (Setting up an EC2 instance). The following is how you logged into the EC2 instance:

```
# Syntax
ssh -i <path_to_the_privatekey> <username>@<ec2_hostname>
# example
ssh -i ~/Downloads/george.pem ubuntu@ec2-54-555-555-555.compute-1.amazonaws.com
```

As you can see, I logged into the instance as ubuntu.

If others need to access the instance, a new user account must be created, and a keypair generated for them to use.

For instance, imagine a team with 3 members: Gittu, Maria, and Nicole. Gittu creates an EC2 instance and therefore has exclusive access to it, using the private key. But how can Maria and Nicole access it? It is not advisable for Gittu to share his private key with others. Instead, Gittu needs to create individual user accounts for Maria and Nicole and register their public keys into the system.

Here is the conversation that happens;

Gittu: Hey Maria - I have created an EC2 instance for us to use. I want to give you access to it. Can you provide me your public key? Maria: Sure, I don't have anything at the moment but I will create a keypair and provide you with the public key. Gittu: Great, One I have it, I will add your public key to the server so you can access it.

To set up this access, you will look at official documentation from AWS instead of me providing curated instructions. You will be in this situation and may have found an article through a Google search, and it is advisable to prioritize AWS documents from their website, https://aws.amazon.com/, if you can find one.

I found this document in AWS to set up user accounts for Maria and Nicole. Make sure to follow the instructions carefully. Suppose you encounter any issues or confusing steps. In that case, you can refer to the video that I created (although it is not necessary to watch it to complete the setup process successfully).

After the succesfull completion of the process, nicole and maria will be able to login like below

```
# example nicole
ssh -i <path_to_nicole_private_key> nicole@ec2-54-555-555-555.compute-1.amazonaws.com
# example maria
ssh -i <path_to_maria_private_key> maria@ec2-54-555-555-555.compute-1.amazonaws.com
```

Note

- Make sure you follow the above mentioned article carefully when performing the process. For example, we are dealing with the ubuntu server, so you need to include the `--disabled-password`
- You learned about key pairs in DSCI 521. Plese check there to refresh your memory.

4.6.3.2. Setting up a common space in EC2

This topic is also included in the Week 2 video. Look at the beginning of this lecture for the video zone.

Click the toggle below for more information.

After creating accounts for your team members, you'll notice folders for each user in the "/home" directory. However, the person who set up the EC2 instance (in my case gittu) doesn't have a user account. Instead, Gittu is using the ubuntu account (and has access to the ubuntu private key). Also, here ubuntu is the `root user`.

```
maria@ip-172-31-21-101:~$ cd /home
maria@ip-172-31-21-101:/home$ ls -ltrh
total 12K
drwxr-xr-x 2 nicole nicole 4.0K Mar 21 07:32 nicole
drwxr-xr-x 5 ubuntu ubuntu 4.0K Mar 21 07:39 ubuntu
drwxr-xr-x 5 maria  maria  4.0K Mar 21 07:39 maria
```

Setup a common data folder to download data, and this folder should be accessible by all users in the JupyterHub. Following commands, make a folder and make it accessible to everyone.

```
sudo mkdir -p /srv/data/my_shared_data_folder
sudo chmod 777 /srv/data/my_shared_data_folder/
```

> (OPTIONAL) If you want a sharing notebook environment, check out this. If you plan to do this, make sure you install the "members" package in your server; run "sudo apt-get install members".

4.6.3.3. Setup your JupyterHub

This topic is also included in the Week 2 video. Look at the beginning of this lecture for the video zone.

Note

The interface has been changed to jupyterlab, so you will see a different interface. Select the terminal from the homepage of jupyterlab.

We installed our TLJH while setting up our EC2 instance. Currently, we only have one admin user - the person who set up the EC2 instance - and the admin username is what you provided in your `EC2 user data` (in my case gittu). We will use that username to log in to JupyterHub. Click the toggle below to access the JupyterHub interface and log in.

1. Under description, check for "IPv4 Public IP" and paste the IP address in your browser for your JupyterHub. (make sure it is not automatically prepended with `https://`, we want to access it over `http://`)

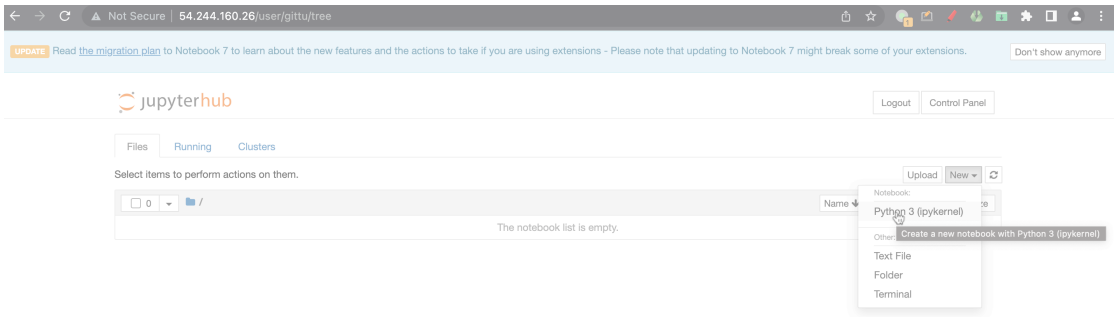
2. Enter your username (what you provided in your `EC2 user data`) and use a strong password & note it down somewhere, as what you enter here will be the admin password.

We want to add other team members as admin users to the jupyterhub. Here are the instructions that you can use. I strongly recommend you check it out, but I will list down the main points.

> In your Jupyterlab, go to "File" –> "Hub Control Panel" –> "Control Panel" –> "admin." Here add other members of your group, use their names and make them admins.

3. Check if other members can log in to the JupyterHub from their machines by giving them the URL to connect. Step 2 is applicable here for other members.

To create a notebook;



Here this chapter you can use as a reference. The interface changed a bit when you wanted to "add other team members as admin users." Here are the instructions that you can use.

4.6.3.3.1. How to install packages in TLJH

This topic is also included in the Week 2 video. Look at the beginning of this lecture for the video zone.

Note

The interface has been changed to jupyterlab, so you will see a different interface. Select the terminal from the homepage of jupyterlab.

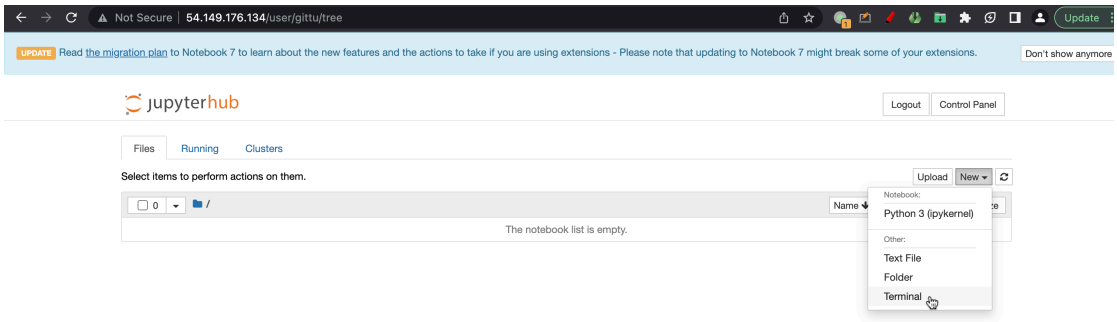
You have to install the packages that are needed. You ***should*** refer this TLJH document.Refer `pip` section..

Tip

Don't forget to add option -E. This way, all packages that you install will be available to other users in your JupyterHub.

For example;

Open up the terminal from the jupyter; Like in the screenshot below;



After that install packages that you need;

```
sudo -E pip install pandas
sudo -E pip install s3fs
sudo -E pip install pyarrow
```

Note

You surely want to install the packages mentioned above. Also, install any other packages that are required for your data wrangling. Please note that the setup and server we have are too small for a fully-fledged collaborative space. Therefore, please only install what is necessary, or it may lead to a system crash.

You can watch the this youtube chapter. The interface has been changed to jupyterlab, so you will see a different interface. Select the terminal from the homepage of jupyterlab.

4.6.4. How to create an image ?

Well, at this point, you have set up your EC2 instance and TLJH. You have also installed all the necessary packages that you need for doing your analysis in jupyter. You might have also installed some other packages in your Ubuntu OS. Whatever it is, if you want to obtain a copy of your computer with everything that it has, then you can make use of EC2 images. You can create the image by ***selecting the EC2 instance –> Click Actions –> Image and Templates –> Create Image***.

4.6.5. Other Services to mention

- AWS Lambda - This is a very useful piece for deployments. If time permits, I will visit this towards the last week.
- ECS – EC2 Container Service
- Elastic Beanstalk

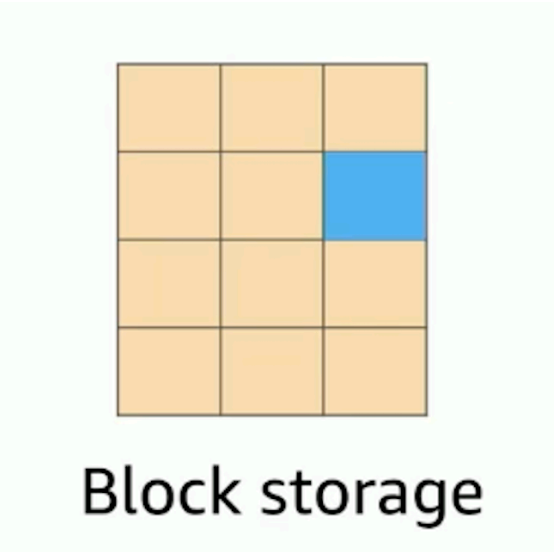
> Check out entire compute services here. Check out the cost here for On-Demand and Spot instances. For our student account we can't use spot instances.

4.7. Storage

- EBS - Elastic Block Store
- EFS - Elastic File System
- S3 - Simple Storage Service

4.7.1. EBS

Elastic Block Storage(EBS) is block storage, which means the disk utilization happens in blocks or data stored in blocks.



Like in the above diagram, a ***file is stored in blocks***, so if we want to change one character in a one GB file, we just want to change the block that contains that one bit. This is how data is stored, and it functions very similar to your laptop hard disk, flash drive, or any external disk. We use this as storage for our EC2 instance.

So we can use EBS as a boot volume or attach it to an existing EC2 instance, just like how we connect an external hard disk to our laptop.

> **Note**
>
> EC2 instances need an Amazon EBS volume to boot.

Here are some of the properties of EBS

- Persistent storage (non-volatile storage)
- Automatically replicated within the availability zones
- High availability and durability
- More durability by backing up (taking snapshots) data to S3.
- Low latency
- Scale storage up or down
- Data encryption

4.7.2. EFS

Provides storage for EC2 instances; unlike EBS, EFS storage can be accessed by multiple EC2 instances simultaneously. EFS offers all the advantages that we mentioned with EBS but also provides the following on top of it

- Auto-scaling
- Replication on multiple availability zones
- Sharing with other EC2 instance

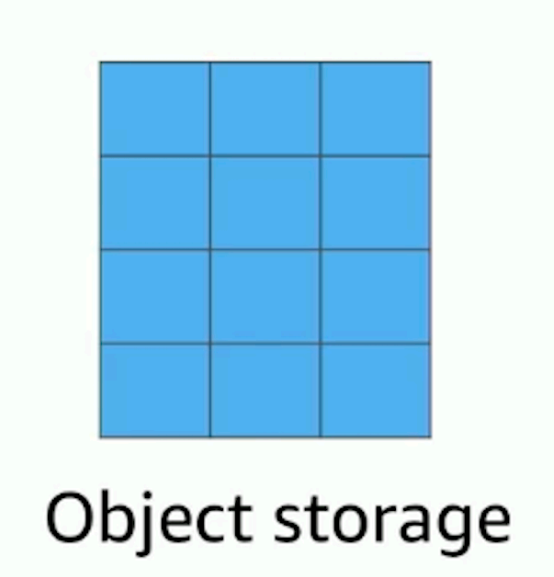
This is mainly used in industry to provide home directories for workers.

> **Note**
>
> EFS won't be replacing EBS; both have their use cases. For example, we need EBS storage for an EC2 instance as a boot drive. But, EFS, you can think more in a corporate environment performing centralized shared storage—uses like media processing or shared code repositories.

Here is an article that shows the difference between EBS and EFS.

4.7.3. S3

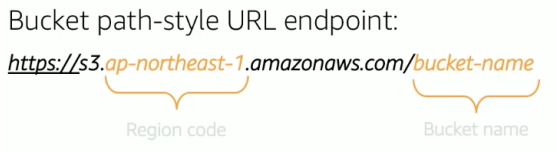
S3 stands for Simple Storage Service. This is one of the first storage services provided by AWS in 2006. It is an object-level storage



Like in the above diagram, a ***file is stored as an entire object***, so if we want to change one character in a one GB file, we want to update the file, and then we have to replace that entire file. By the object storage, you can think of it as the ***Key-Value store***, where the key is the filename, and the Value is the contents of the object or the file itself. So you store an object with a Key and retrieve the object with the key.

No concept of folders in S3. Even though the path you see looks like a path to a file, the path is a key representing that object.

Here is how the path to S3 means



Summarizing some properties for S3

- Data is stored as objects in buckets
- Virtually unlimited storage
  - Single object is limited to 5 TB
- Designed for 99.999999999% of durability
- Granular access to buckets and objects

> **Note**
>
> You don't need to specify the availability zones, as AWS will take care of all these in replicating your data across different availability zones.

4.8. Setting up S3 bucket

> This topic is also included in the Week 2 video. Look at the beginning of this lecture for the video zone.

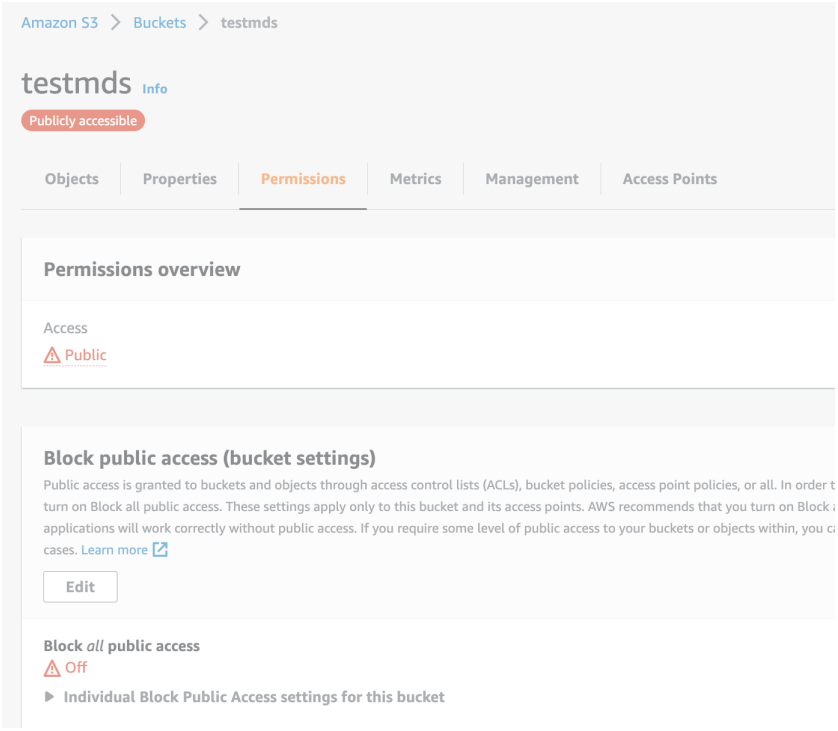
Before attempting to read or upload to your S3 bucket, you should ensure that the bucket exist and have access to it. Due to the limitations of the student account, we cannot dig into the permissions and policies required for this. However, in our case, we can simply make the bucket public so that anyone, including your team members, can access it.

Click the toggle below for instructions to create and setup your bucket and make it public:

- Go to the S3 Service in AWS console.
- Create a bucket here. (e.g.:I gave an example name `mds-s3-14-gittu`).
- Make sure AWS region is `us-west-2`.
- When creating the root bucket, make it public. Make sure you unchecked "Block all public access".
- (Default encryption) Bucket key make sure you check radio button for `disable`
- All other options leave as it is. And click on `Create bucket`.
- After your bucket is created make sure to change the policies of the bucket.
  - Click on your buckets
  - Click on Permissions tab
  - Edit bucket policy and paste the following policy and then Save changes. Make sure you replace the bucket name with your bucket name, in my case it is `mds-s3-14-gittu`.

```
{
    "Version": "2012-10-17",
    "Id": "Policy1649284381437",
    "Statement": [
        {
            "Sid": "Stmt1649284379371",
            "Effect": "Allow",
            "Principal": {
                "AWS": "*"
            },
            "Action": "s3:*",
            "Resource": [
                "arn:aws:s3:::mds-s3-14-gittu",
                "arn:aws:s3:::mds-s3-14-gittu/*"
            ]
        }
    ]
}
```

If everything has been done correctly, you will see a red `public` symbol in your bucket.



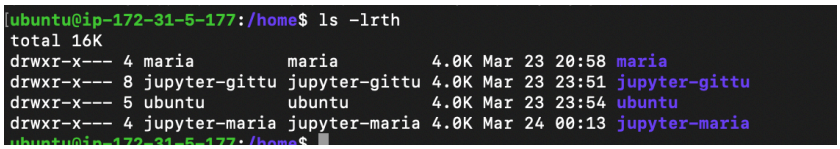
Here this video you can use as a reference.

4.8.1. More about credentials

You can create a folder in your S3 bucket and store any files in it. Since your bucket is now public, you and your team members can access it. Before moving on to the next session, please make sure you have correctly installed AWS CLI and updated your AWS credentials (as they change every time your lab restarts/starts) to `~/.aws/credentials`. You can find more details about this in our previous lecture here.

If you have already updated your credentials in `~/.aws/credentials`, you do not need to pass them explicitly anywhere. However, if you still want to pass them explicitly, I have included instructions in an optional section here.

Here is the confusing part: I said you do not want to pass credentials explicitly if you have already updated them in `~/.aws/credentials`. This is because, when you run the AWS CLI or any API, it looks for this credentials file in your home directory, finds it, and takes the credentials from there to execute. This process is straightforward when you are working on your laptop. However, when you work on a shared environment (like the EC2 instance we have), each user has their own home directory, and they must update it with their credentials in their home directory `~/.aws/credentials`. When you access TLJH notebook, it does not execute the notebook as the same user in your Linux (maria is the user in unix), but as a different user ("jupyter-maria" is user in TLJH), such as "jupyter-yourjupyterusername" (in my case, "jupyter-gittu") or "jupyter-maria". Therefore, if you execute something from the Jupyter notebook in EC2, you want to make sure that your credentials are in the respective home directories. I have included a picture of all home directories to make this situation more understandable.



Workaround - For the simplicity you can do the following

Simply move credentials (here all users will be sharing the root user credentials) to a shared folder, and let aws know that this is now where the credentials are:

1. Create a shared folder in the `srv` folder

```
sudo mkdir -p /srv/keys
```

2. Copy credentials to that folder and make it readable by other (chmod)

```
sudo cp ~/.aws/credentials /srv/keys/
sudo chmod -R 777 /srv/keys
```

3. In your Jupyter Hub notebook file, add this cell:

```
os.environ["AWS_SHARED_CREDENTIALS_FILE"] = "/srv/keys/credentials"
```

4.9. How to transfer data to S3

There are many ways you can put data into S3. You can do it using...

4.9.1. Web interface

You can upload tiny files using UI and bigger files using CLI. I will show this in class.

4.9.2. SDK

Using pandas and AWS wrangler.

4.9.3. Using CLI

You can use AWS CLI what you installed previously to upload data.

> If you want to upload a folder (or a parquet file) best is to use AWS CLI.

eg: `aws s3 cp aws/ s3://mds-s3-gittu/output/ --recursive`

Here in the above eg I am moving a folder named "aws/" to the S3 path "s3://mds-s3-gittu/output/" ; adding –recursive as it's a folder. This is similar to the `cp - R` in linux.

> **See also**
>
> You can also use the AWS CLI for moving local data to AWS. Refer to this document.

4.10. How to read data from S3

Popular data processing libraries like pandas and arrow offer built-in features for retrieving data from Amazon S3 cloud storage. You can access the data just as you would access it from your local computer by simply adding "s3://<path_to_file_in_s3>" at the beginning of the file path.

To enable this functionality, most of these libraries use either the `boto3` or `s3fs` library in the background. For example, pandas use s3fs, while awswrangler uses boto3. To make it pretty straightforward you can use pandas to read data from S3 and I will be showing you that below. But it is bit buggy, and if you get into any `forbidden error` despite of making sure that you made bucket public and updated credentials correctly in `~/.aws/credentials` (remember your credential gets updated everytime it restarts), then this could be with some issues with pandas reader, if that is the case please switch to awswrangler. I am giving instruction on how to use awswrangler at the end as optional.

To start, let's install the necessary packages using pip. Then, we can easily read files from S3 and start analyzing the data. If you want to see how to install please check above section on How to install packages in TLJH.

```
sudo -E pip install pandas
sudo -E pip install s3fs
sudo -E pip install pyarrow
```

```
import json
import urllib.parse
import pandas as pd
```

```
---------------------------------------------------------------------------
ImportError                               Traceback (most recent call last)
Cell In[1], line 3
      1 import json
      2 import urllib.parse
----> 3 import pandas as pd

File ~/miniforge3/envs/525_dev_2025_jupyter/lib/python3.11/site-packages/pandas/__init__.py:19
     16         _missing_dependencies.append(f"{_dependency}: {_e}")
     18 if _missing_dependencies:  # pragma: no cover
---> 19     raise ImportError(
     20         "Unable to import required dependencies:\n" + "\n".join(_missing_dependencies)
     21     )
     22 del _hard_dependencies, _dependency, _missing_dependencies
     24 try:
     25     # numpy compat

ImportError: Unable to import required dependencies:
numpy: Error importing numpy: you should not try to import numpy from
        its source directory; please exit the numpy source tree, and relaunch
        your python interpreter from there.
```

```
%%time
df = pd.read_parquet("s3://testmds2024/combined_data.parquet",
        filters=[('year','=', 2004)],
                columns=['year', 'UniqueCarrier', 'ArrDelay'])
print(df[(df.year== 2004) & (df.ArrDelay > 10)]["UniqueCarrier"].value_counts())
```

```
UniqueCarrier
WN    225167
DL    189001
AA    176385
MQ    130511
UA    126692
NW    125030
US     95326
XE     94467
OO     87328
OH     83799
CO     74989
EV     68299
DH     66330
HP     55272
FL     42911
AS     42187
B6     19421
TZ     17751
HA      3642
Name: count, dtype: int64
CPU times: user 2.08 s, sys: 372 ms, total: 2.45 s
Wall time: 10.2 s
```

```
%load_ext rpy2.ipython
```

```
import os
os.environ['R_HOME'] = '/Users/ggeorg02/opt/miniconda3/envs/525_dev_2024/lib/R'
```

```
%%R
suppressMessages(library(arrow, warn.conflicts = FALSE))
suppressMessages(library(dplyr, warn.conflicts = FALSE))
```

```
%%time
%%R
open_dataset("s3://testmds2024/combined_data.parquet") %>%
    filter(year==2004,ArrDelay >10) %>%
    count(UniqueCarrier) %>%
    collect()
```

```
# A tibble: 19 × 2
   UniqueCarrier      n
   <chr>          <int>
 1 UA            126692
 2 FL             42911
 3 HA              3642
 4 HP             55272
 5 MQ            130511
 6 DH             66330
 7 DL            189001
 8 US             95326
 9 WN            225167
10 AA            176385
11 NW            125030
12 TZ             17751
13 XE             94467
14 AS             42187
15 B6             19421
16 CO             74989
17 OH             83799
18 OO             87328
19 EV             68299
CPU times: user 1.96 s, sys: 228 ms, total: 2.19 s
Wall time: 7.95 s
```

```
## just getting 100 rows to save time
df = pd.read_csv("s3://testmds2024/combined_data.csv",nrows = 1000)
print(df[(df.year== 1996) & (df.ArrDelay > 10)]["UniqueCarrier"].value_counts())
```

```
UniqueCarrier
HP    169
DL    103
WN     65
US     41
CO     14
Name: count, dtype: int64
```

Thoughts/Discussion ?

- What is the difference between block and object-level storage?
- Which storage will be appropriate when building a relational database?
- List down situations where it's important/preferred to use S3 over EBS?
- Uploading data using a web interface seems to be easy. But why is it not recommended to upload data using that way?

4.11. What we learned today?

- The main category of services offered by AWS.
- Understanding the main compute services.

- How to set up an EC2 instance and understand the various options at different stages of setup.
- Understanding a variety of storage services and the ability to choose a storage solution based on the use case.
- Understanding how working with EC2 can be considered a scale-UP solution.

4.12. Outside of class

- Setting up an EC2 instance for collaboration
- Configuring JupyterHub and AWS CLI
- Accessing data from S3

4.13. Appendix

Some useful unix commands you used in lab and tutorial:

```
# Add your aws credentials to the file
vi ~/.aws/credentials

# to make sure that your cli is set correctly
aws sts get-caller-identity

# to move folder to s3
aws s3 cp lectures/ s3://test-gittu/lectures --recursive

# command to generate keypairs
ssh-keygen -m PEM -f maria
# Extract the public key out of private key
ssh-keygen -y -f Maria.pem

# Add a new user to ubuntu machine
sudo adduser new_user --disabled-password
# become the new user, using sudo
sudo su - new_user
# creating directory
mkdir .ssh
# changing permissions
chmod 700 .ssh
#Creating a file
touch .ssh/authorized_keys
```

4.14. S3 Glacier (optional)

Amazon S3 glacier is an extremely low-cost cloud service for long-term backup (data archiving service). The drawback is that it takes several hours to retrieve the stored data and should only be used for archiving. The retrieval time depends on the retrieval option that the user goes for. There are mainly 3 retrieval options...

- Bulk (5 - 12 hours)
- Standard (3- 5 hours)
- Expedited (1 - 5 minutes)

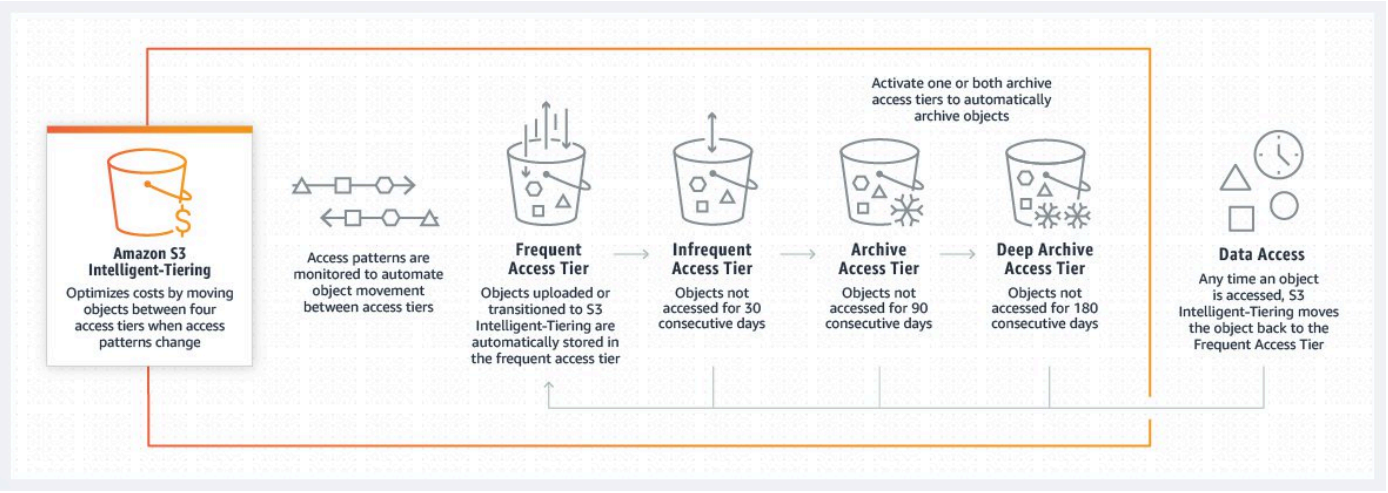
Cost increases as the speed of retrieval increases.

There are 3 main elements of glacier...

- Archive All files (Any object such as a photo, video file, or document) should be zipped before uploading to the glacier. This is considered the base unit of storage. All have a unique ID and also have a description
- Vault It is a container for storing archives, and you can specify the region where you want to locate your vault.
- Access policies This is how you control access to the vault. You can give permissions to individuals and what kind of operation they can do.

Can you think of use-cases for storing data in a glacier?

In industries, data usually follows a lifecycle, and we can achieve this life cycle using S3 and glacier.



Here you can read some best practices with S3.

4.15. How to read data from S3 using awswrangler (optional)

First install it;

```
!conda install -c conda-forge -y awswrangler
```

If you are planning to install in EC2 instance then;

```
sudo -E conda install -y awswrangler
```

```
import awswrangler as wr
```

```
## just getting 100 rows to save time
df = wr.s3.read_csv("s3://testmds2024/combined_data.csv",nrows=100)
print(df)
```

```
   UniqueCarrier TailNum  ArrDelay  DepDelay Origin Dest  Distance  year
0             DL  N673DL      66.0      69.0    ATL  PHX    1587.0  1996
1             DL  N686DA       3.0       1.0    ATL  PHX    1587.0  1996
2             DL  N685DA      52.0      26.0    ATL  PHX    1587.0  1996
3             DL  N522DA      84.0     100.0    ATL  PHX    1587.0  1996
4             DL  N2824W      56.0      84.0    ATL  PHX    1587.0  1996
..           ...     ...       ...       ...    ...  ...       ...   ...
95            DL  N603DL      55.0      31.0    ATL  PHX    1587.0  1996
96            DL  N677DL      19.0       5.0    ATL  PHX    1587.0  1996
97            DL  N663DN      59.0      22.0    ATL  PHX    1587.0  1996
98            DL  N675DL       8.0       9.0    ATL  PHX    1587.0  1996
99            DL  N618DL      85.0      63.0    ATL  PHX    1587.0  1996

[100 rows x 8 columns]
```

```
df = wr.s3.read_parquet(path="s3://testmds2024/combined_data.parquet", dataset=True)
print(df[(df.year== "2004") & (df.ArrDelay > 10)]["UniqueCarrier"].value_counts())
```

```
UniqueCarrier
WN    225167
DL    189001
AA    176385
MQ    130511
UA    126692
NW    125030
US     95326
XE     94467
OO     87328
OH     83799
CO     74989
EV     68299
DH     66330
HP     55272
FL     42911
AS     42187
B6     19421
TZ     17751
HA      3642
Name: count, dtype: Int64
```

```
/Users/ggeorg02/opt/miniconda3/envs/525_dev_2024/lib/python3.11/site-packages/awswrangler/_distributed.py
  return cls.dispatch_func(func)(*args, **kw)
```

```
df = wr.s3.read_parquet(path="s3://testmds2024/combined_data.parquet",
      dataset=True,
      columns=['year', 'UniqueCarrier', 'ArrDelay'],
      # We have to give it this way to filter on partitions
      partition_filter=lambda x: x["year"] == "2004"
      # I don't see this filters getting pushdown internally maybe bug https://github.com/aws/aws-sdk-pan
      # pyarrow_additional_kwargs={"filters":[('ArrDelay', '>', 10)]}
    )
print(df[(df.year== "2004") & (df.ArrDelay > 10)]["UniqueCarrier"].value_counts())
```

```
UniqueCarrier
WN    225167
DL    189001
AA    176385
MQ    130511
UA    126692
NW    125030
US     95326
XE     94467
OO     87328
OH     83799
CO     74989
EV     68299
DH     66330
HP     55272
FL     42911
AS     42187
B6     19421
TZ     17751
HA      3642
Name: count, dtype: Int64
```

If you want upload a file using awswrangler;

```
# install first
# sudo -E conda install -y s3fs
# sudo -E conda install -y awswrangler
import awswrangler as wr
wr.s3.upload(local_file='img/task.jpeg', path='s3://testmdsprivate/task.jpeg')
```

Note

The above usage is to upload a file. If you want to upload a folder (or a parquet file) best is to use AWS CLI.

4.16. Passing credentials explicitly (optional)

4.16.1. In pandas

```
import pandas as pd
aws_credentials ={"key": "paste_key","secret": "paste_secret","token":"paste_token"}
## dont include you secret and key when submitting the notebook
df = pd.read_parquet("s3://testmds/combined_data.parquet",
        filters=[('year','=', 2004)],
                columns=['year', 'UniqueCarrier', 'ArrDelay'], storage_options=aws_credentials)
```

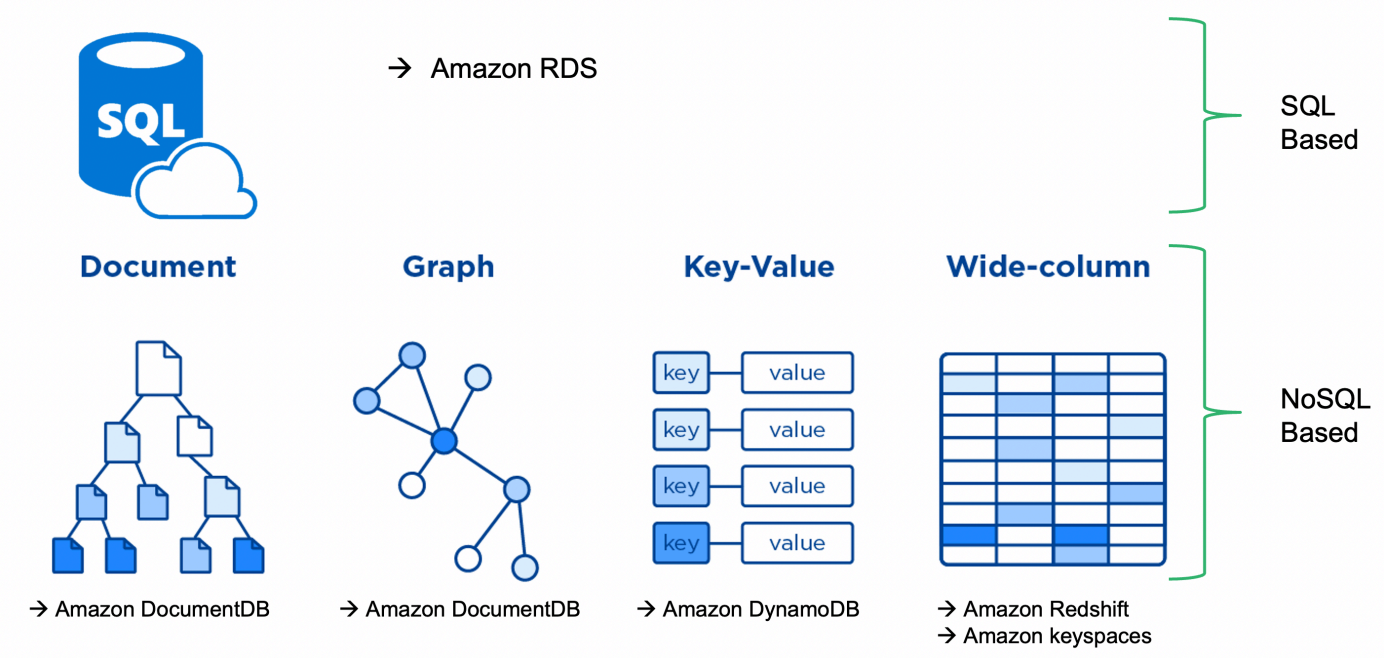
```
df = pd.read_csv("s3://testmds/combined_data.csv",storage_options=aws_credentials)
print(df[(df.year== 2004) & (df.ArrDelay > 10)]["UniqueCarrier"].value_counts())
```

4.16.2. In awswrangler

```
## install boto if you plan to do pass it explicitly
import boto3
import awswrangler as wr
session = boto3.Session(
    aws_access_key_id="xxx",
    aws_secret_access_key="xxx",
  aws_session_token="xxxx"
)
df = wr.s3.read_parquet(path="s3://testmds/combined_data.parquet", dataset=True, boto3_session= session)
```

```
df = wr.s3.read_csv("s3://testmds/combined_data.csv", boto3_session= session)
```

4.17. Database (optional)



Source: dbtest

4.17.1. Database (Amazon RDS)

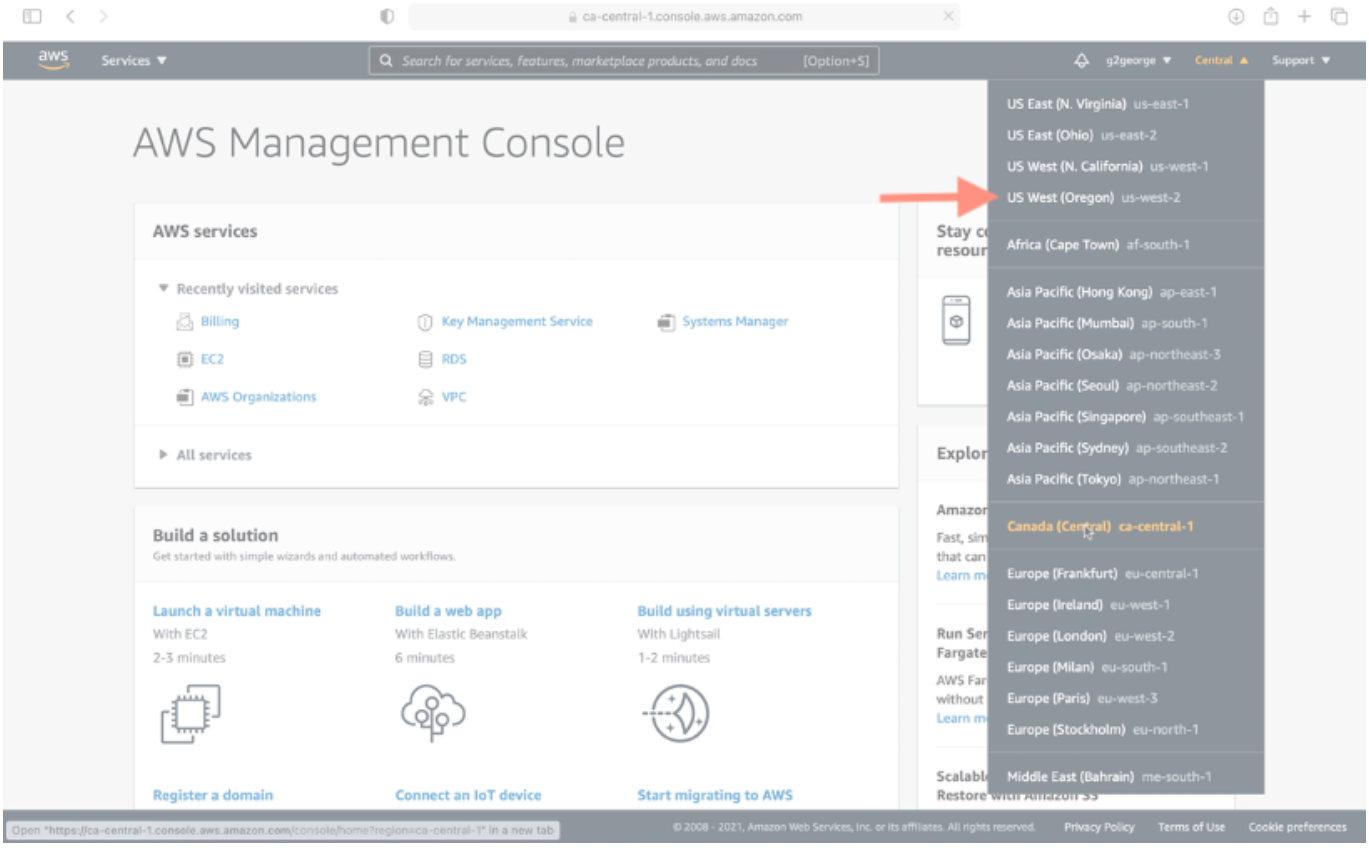
- Relational database service (RDS) provides a simple way to provision, create, and scale a relational database within AWS.
- Managed service – AWS takes responsibility for the administrative tasks
- Following database engines are supported
  - Amazon Aurora
  - PostgreSQL
  - MySQL
  - MariaDB
  - Oracle Database
  - SQL Server

Click below toggle to see the steps to create an RDS instance.

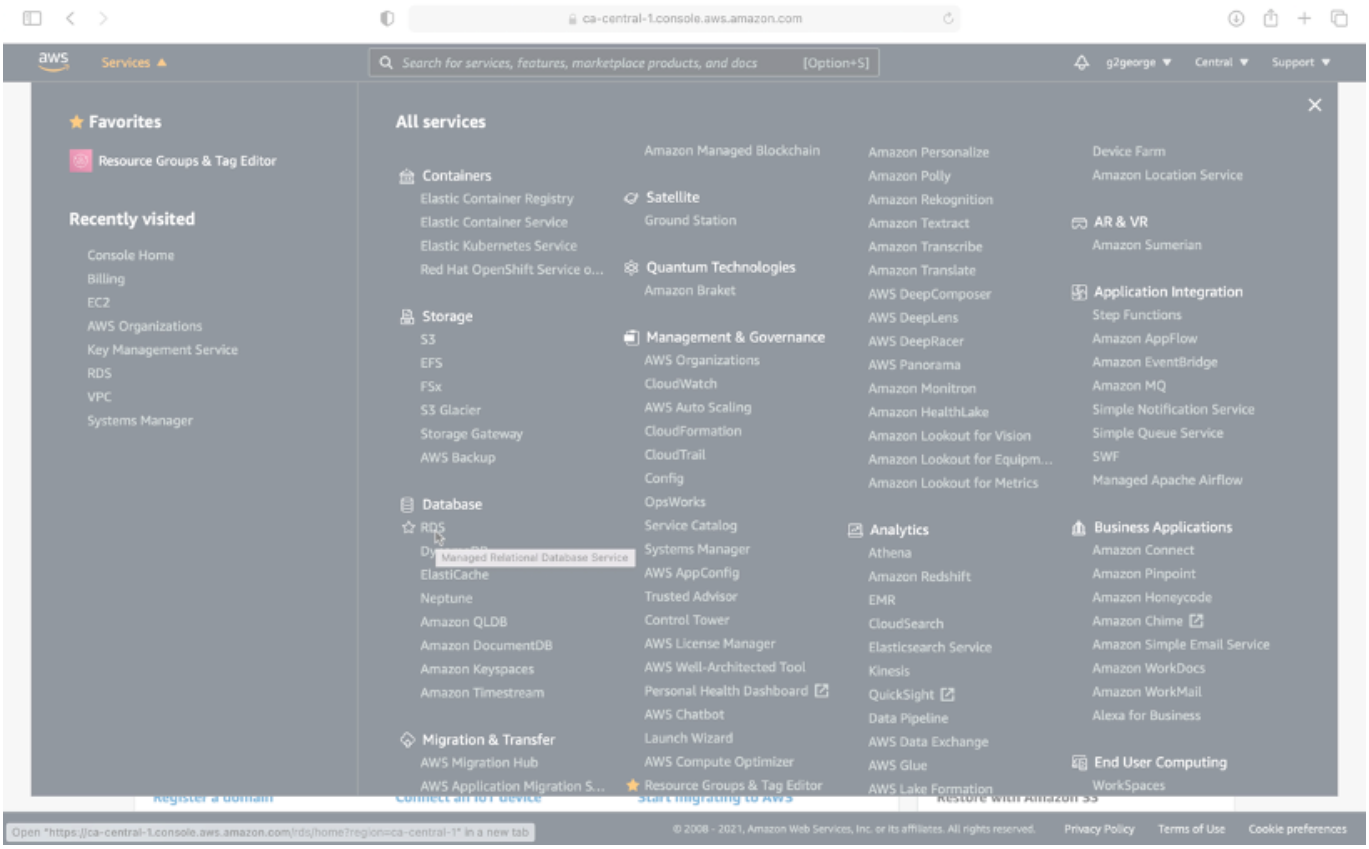
Here we will go through the steps to create your first database in RDS.

- Go to your AWS management console (refer to AWS installation instructions).

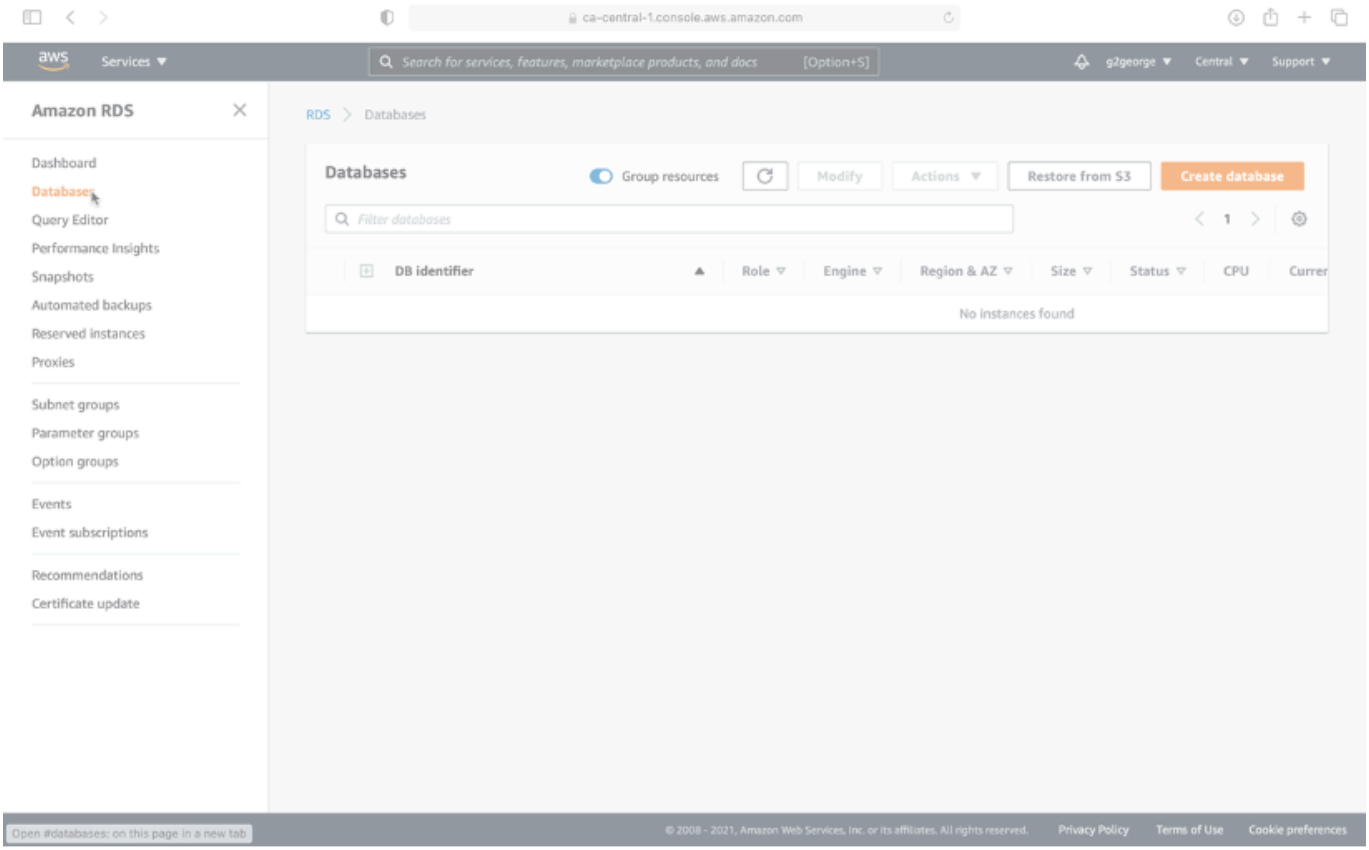
> **Attention**
>
> Make sure your region is in AWS US West (Oregon) region.



- Select the RDS from the Services list `SERVICES` –> `DATABASE` –> `RDS`

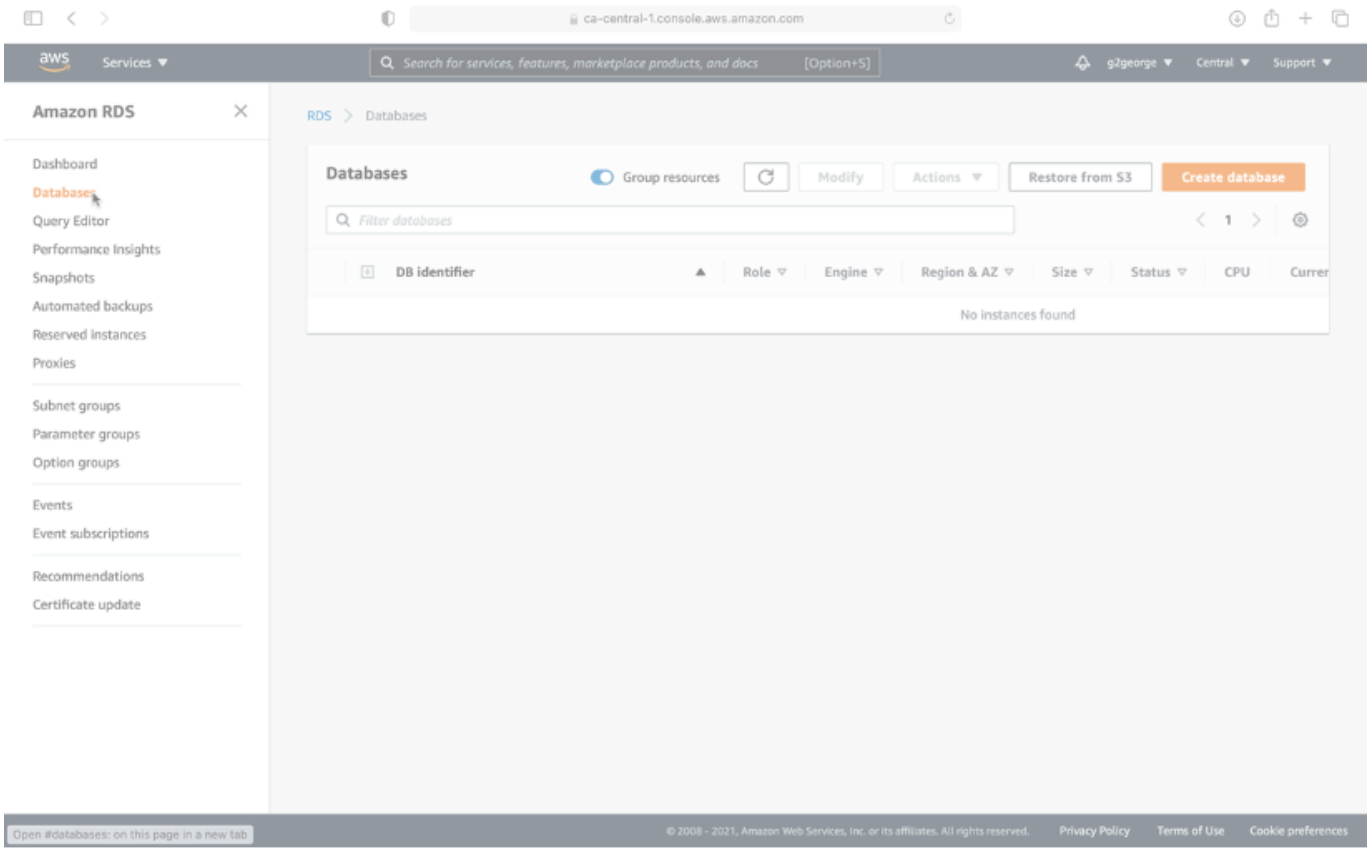


- Click on DATABASES



As you see here, you don't have any database running. Are you ready to create one? Stay with me...

- Click on `CREATE DATABASE.`

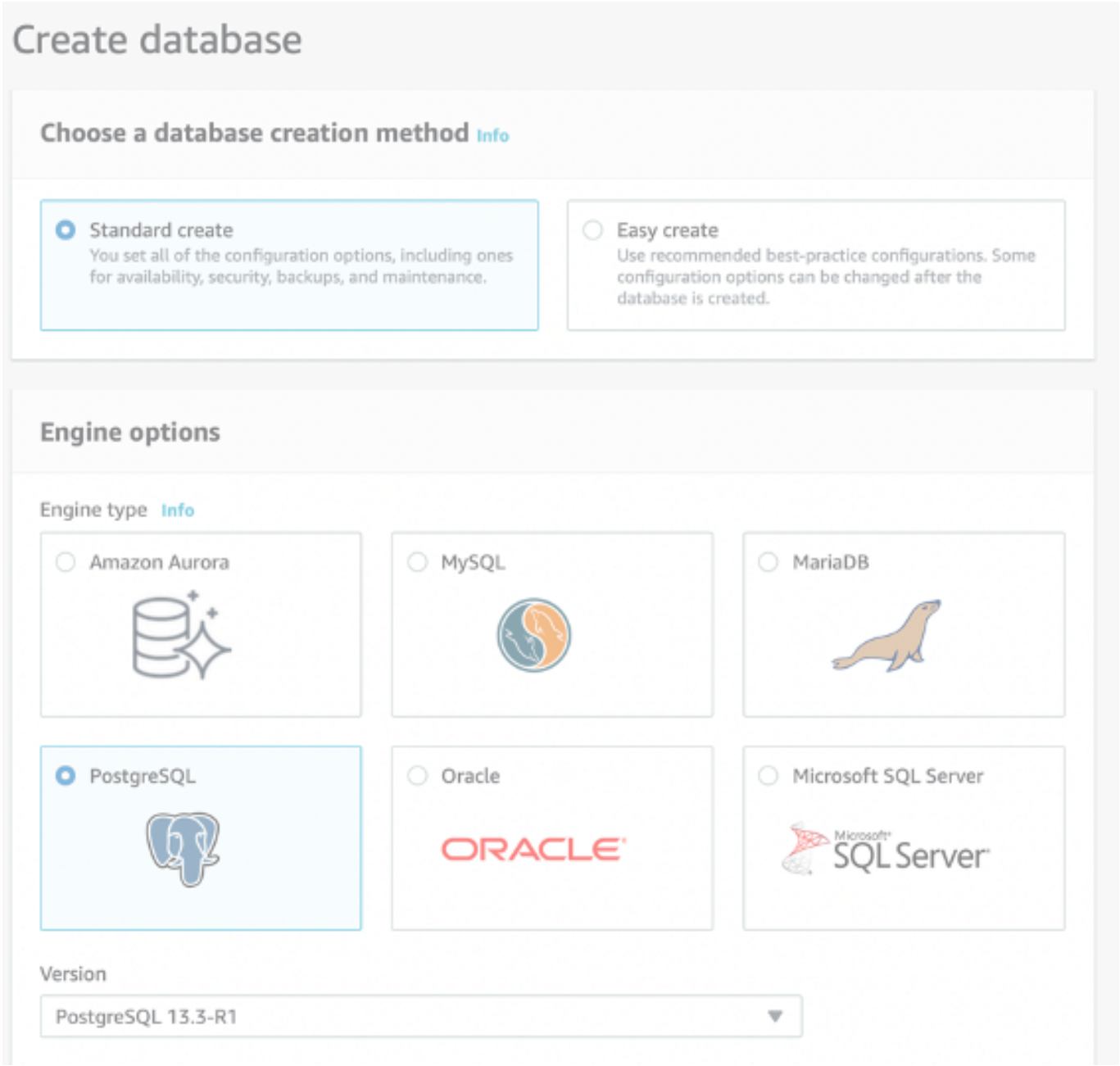


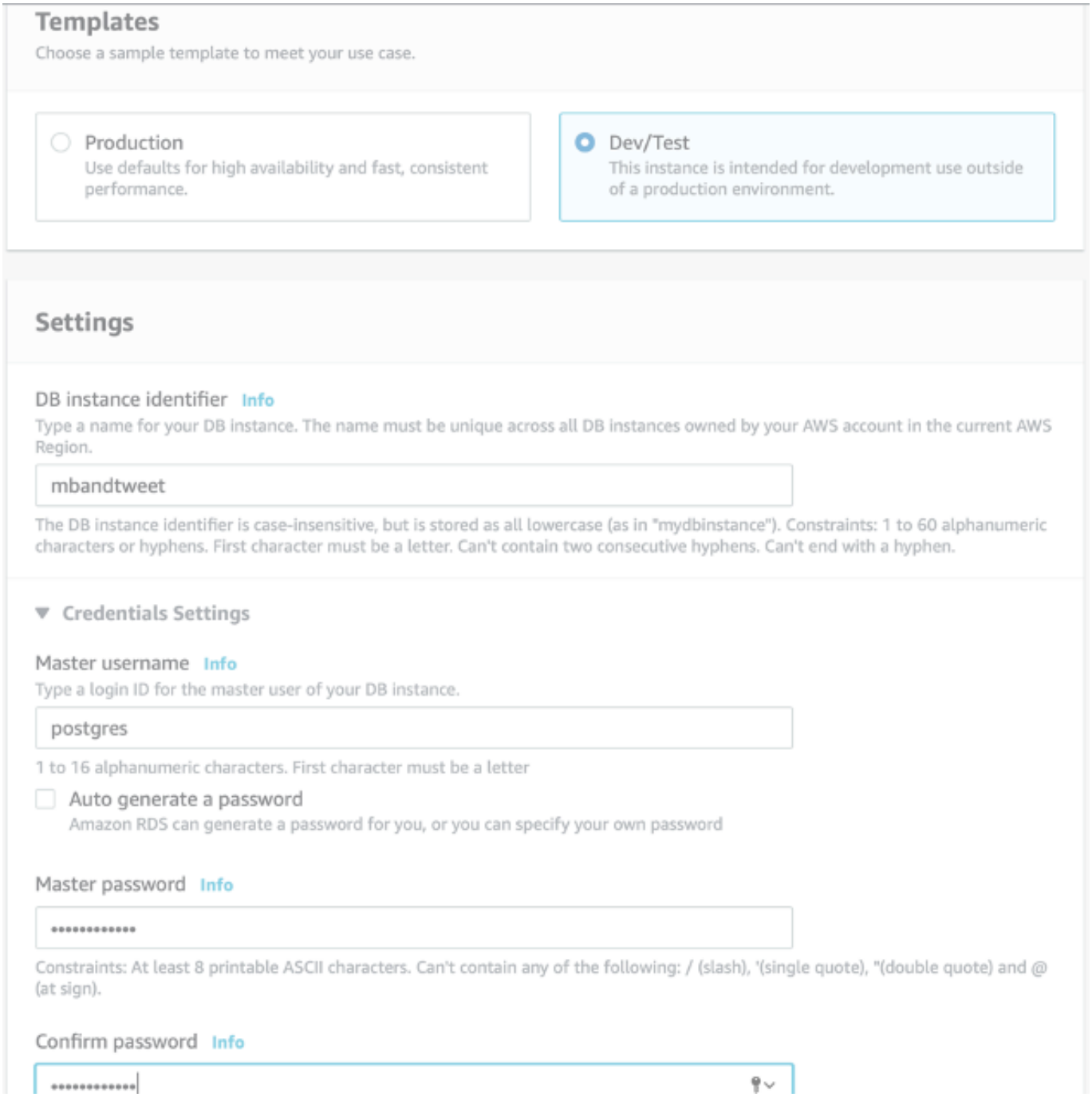
This will take you to the console, where you can see various options for creating your database, starting with the type of database you want to create.

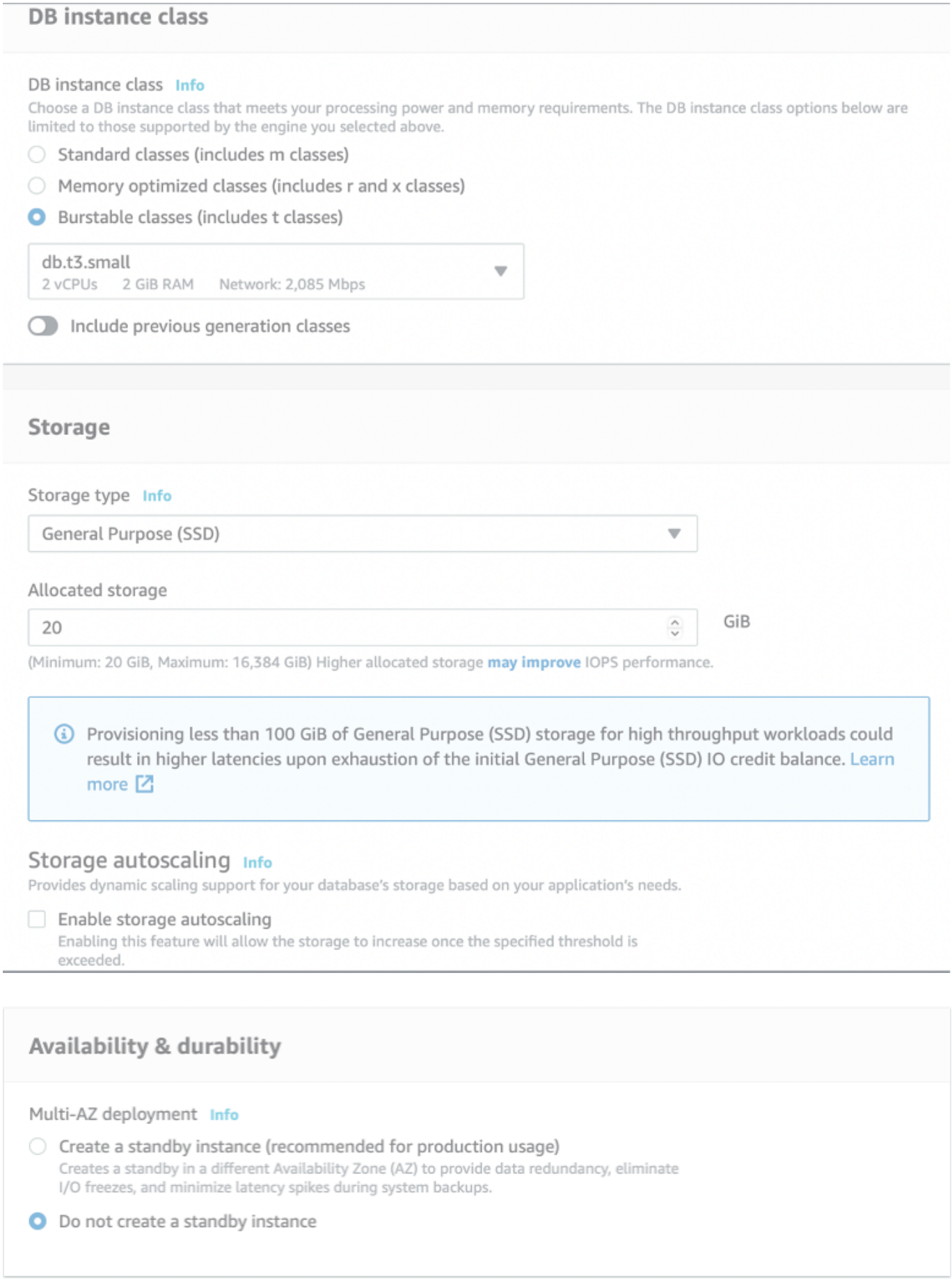
Let's go through each option one by one in the following screenshots.

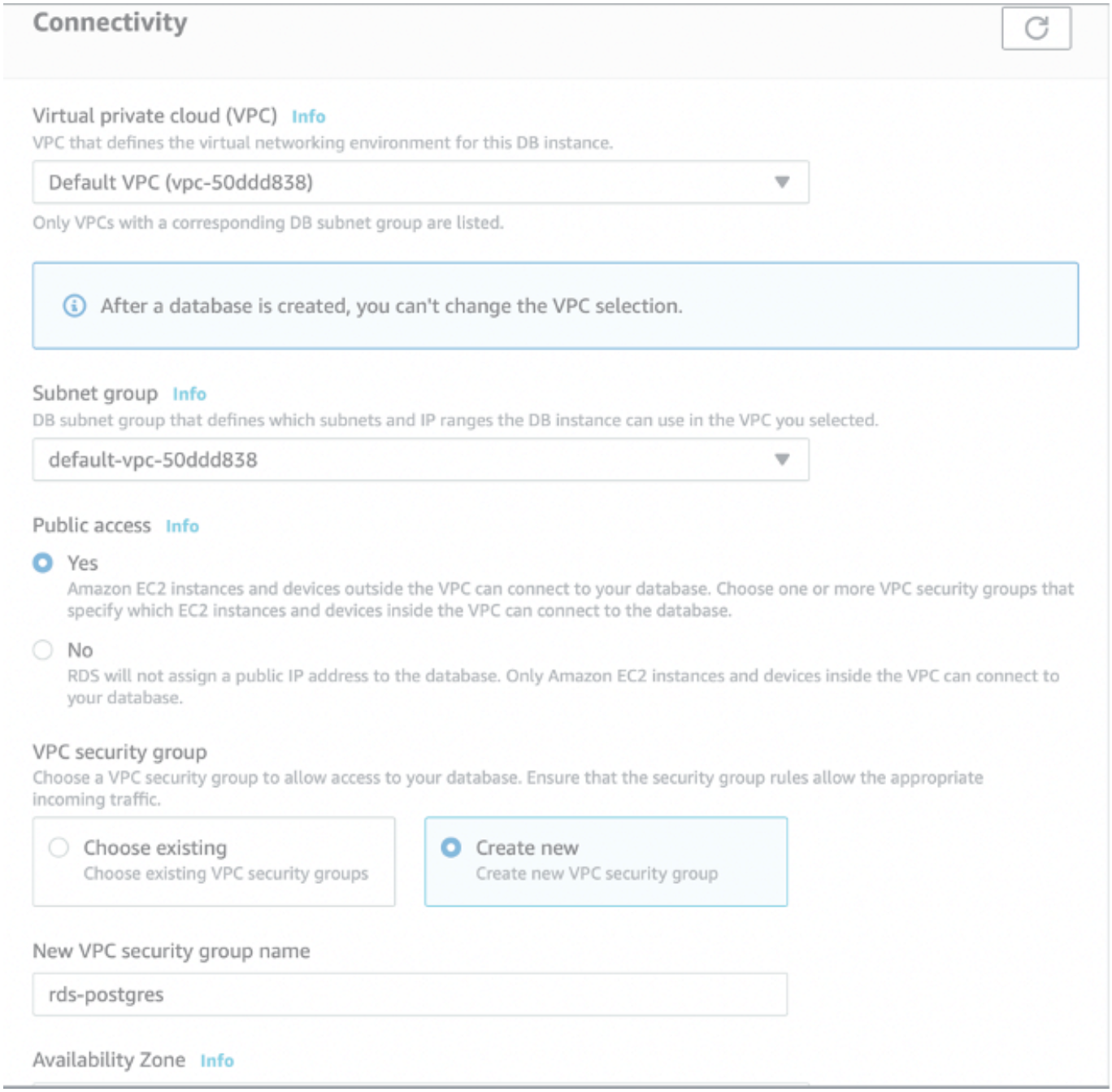
- Here, we are selecting the database engine that we want. As you see, there are 6 of them available, and we will be going with Postgres. You will be hearing more about it in your Lecture 2. Here we also select which version of Postgres we want.

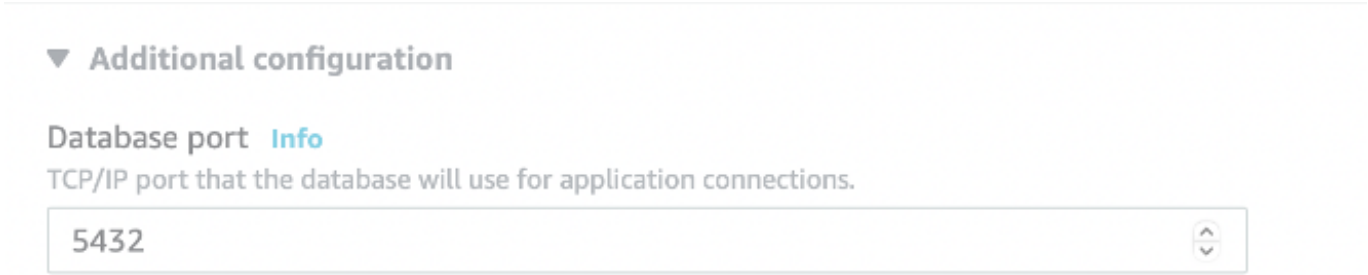
Make sure you select the latest version of postgres. It might not be what I selected here, ie `PostgreSQL 13.3-R1`.

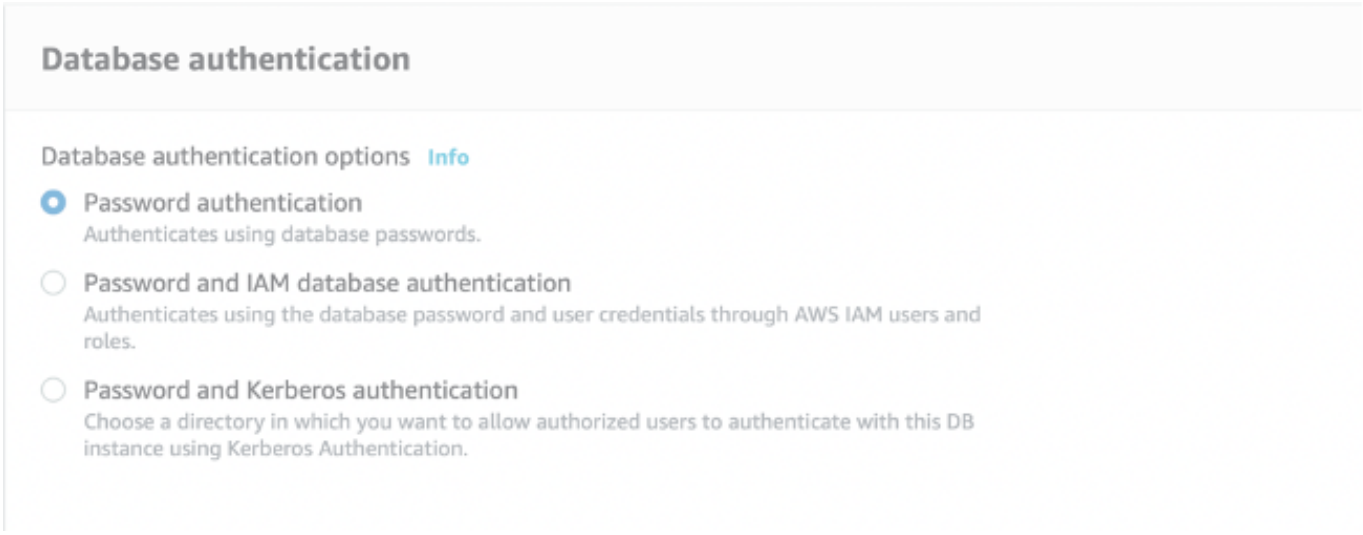

Create database
Choose a database creation method Info
Standard create
You set all of the configuration options, including ones for availability, security, backups, and maintenance.
Easy create
Use recommended best-practice configurations. Some configuration options can be changed after the database is created.
Engine options
Engine type Info
Amazon Aurora
MySQL
MariaDB
PostgreSQL
Oracle
ORACLE
Microsoft SQL Server
Microsoft SQL Server
Version
PostgreSQL 13.3-R1


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.
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.
mbandtweet
The DB instance identifier is case-insensitive, but is stored as all lowercase (as in "mydbinstance"). Constraints: 1 to 60 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. First character must be a letter
Auto generate a password
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), '(single quote), "(double quote) and @ (at sign).
Confirm password Info

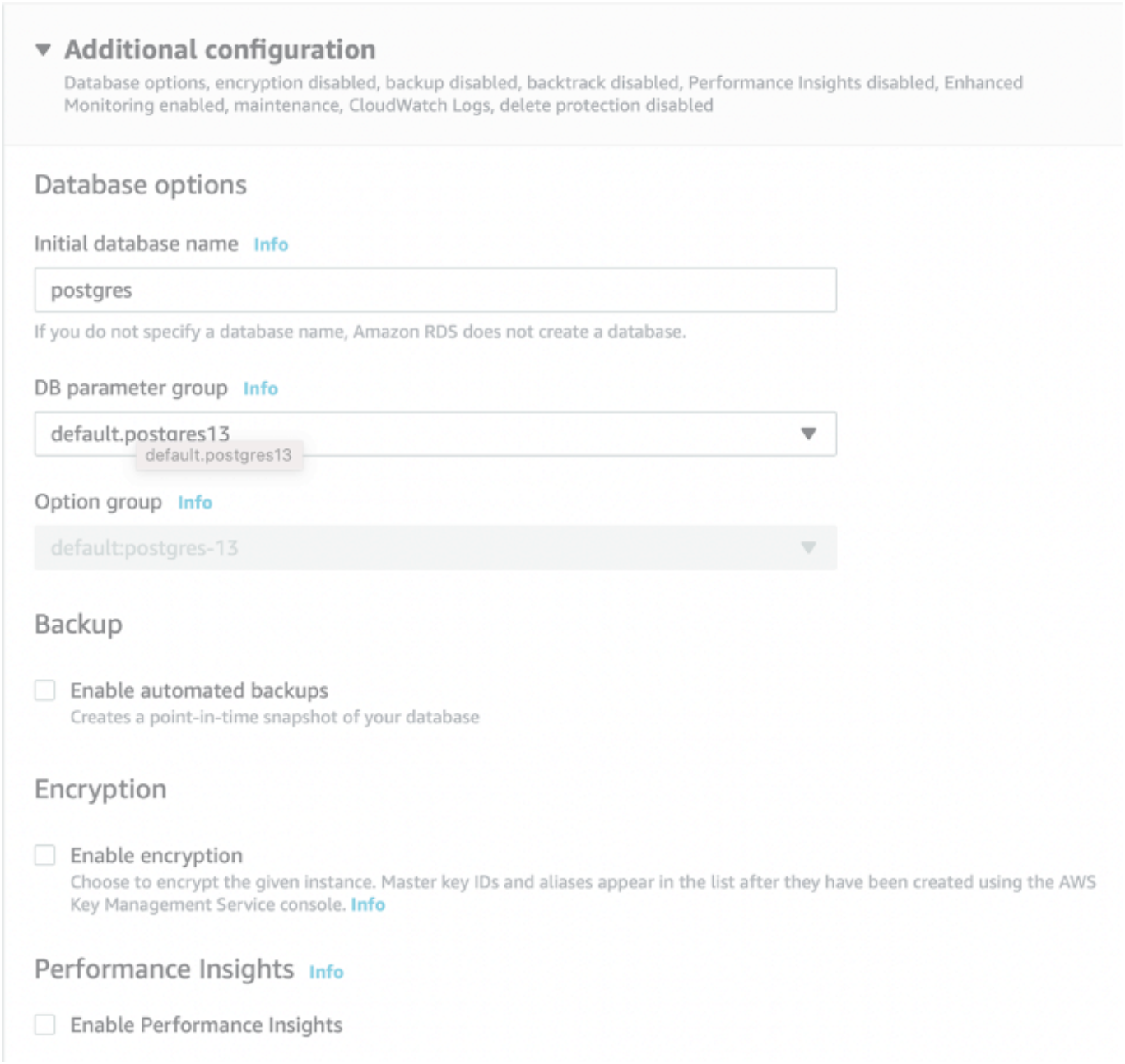

DB instance class
DB instance class Info
Choose a DB instance class that meets your processing power and memory requirements. The DB instance class options below are limited to those supported by the engine you selected above.
Standard classes (includes m classes)
Memory optimized classes (includes r and x classes)
Burstable classes (includes t classes)
db.t3.small
2 vCPUs 2 GiB RAM Network: 2,085 Mbps
Include previous generation classes
Storage
Storage type Info
General Purpose (SSD)
Allocated storage
20
GiB
(Minimum: 20 GiB, Maximum: 16,384 GiB) Higher allocated storage may improve IOPS performance.
Provisioning less than 100 GiB of General Purpose (SSD) storage for high throughput workloads could result in higher latencies upon exhaustion of the initial General Purpose (SSD) IO credit balance. Learn more
Storage autoscaling Info
Provides dynamic scaling support for your database's storage based on your application's needs.
Enable storage autoscaling
Enabling this feature will allow the storage to increase once the specified threshold is exceeded.
Availability & durability
Multi-AZ deployment Info
Create a standby instance (recommended for production usage)
Creates a standby in a different Availability Zone (AZ) to provide data redundancy, eliminate I/O freezes, and minimize latency spikes during system backups.
Do not create a standby instance


Connectivity
Virtual private cloud (VPC) Info
VPC that defines the virtual networking environment for this DB instance.
Default VPC (vpc-50ddd838)
Only VPCs with a corresponding DB subnet group are listed.
After a database is created, you can't change the VPC selection.
Subnet group Info
DB subnet group that defines which subnets and IP ranges the DB instance can use in the VPC you selected.
default-vpc-50ddd838
Public access Info
Yes
Amazon EC2 instances and devices outside the VPC can connect to your database. Choose one or more VPC security groups that specify which EC2 instances and devices inside the VPC can connect to the database.
No
RDS will not assign a public IP address to the database. Only Amazon EC2 instances and devices inside the VPC can connect to your database.
VPC security group
Choose a VPC security group to allow access to your database. Ensure that the security group rules allow the appropriate incoming traffic.
Choose existing
Choose existing VPC security groups
Create new
Create new VPC security group
New VPC security group name
rds-postgres
Availability Zone Info

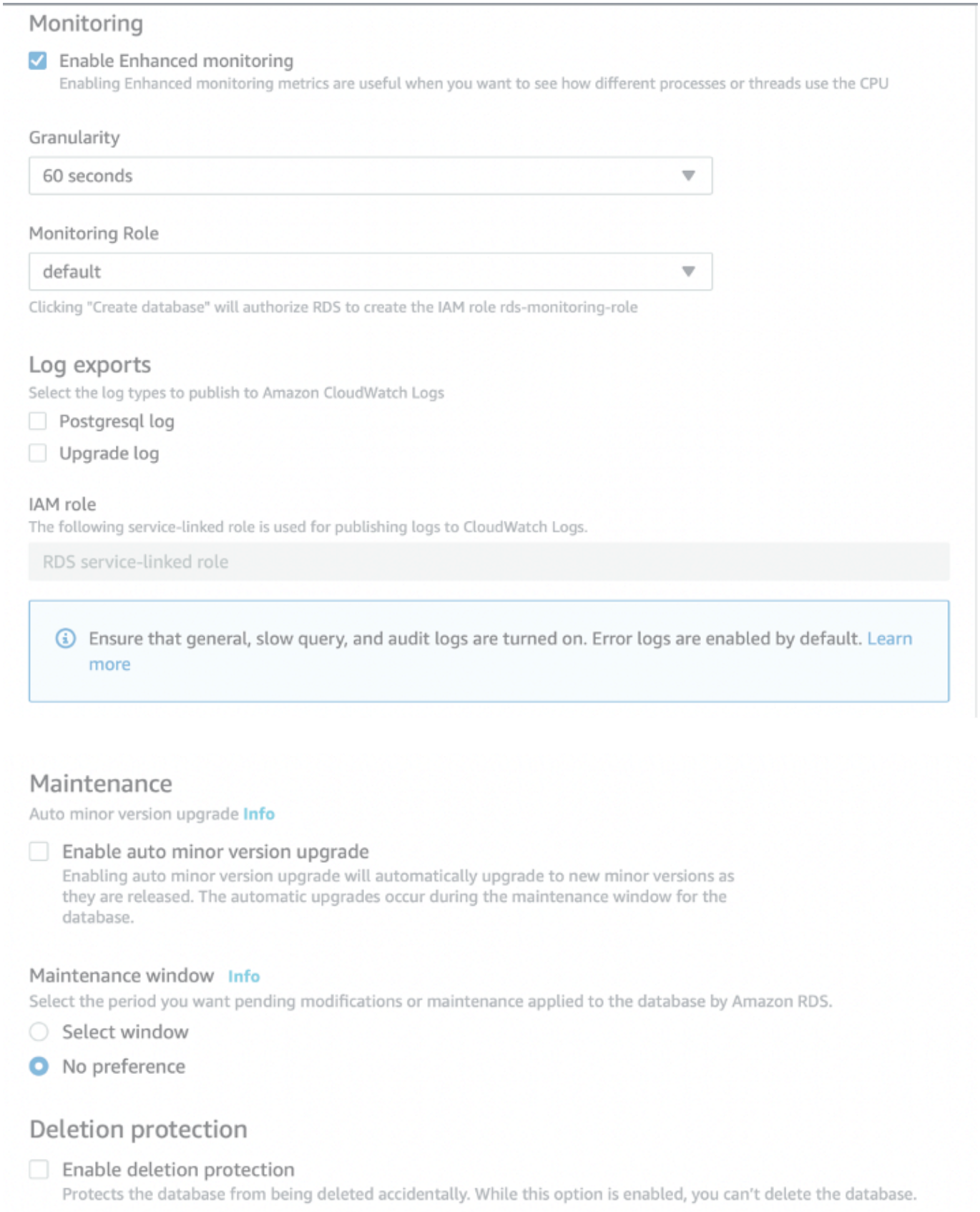

Additional configuration
Database port Info
TCP/IP port that the database will use for application connections.
5432

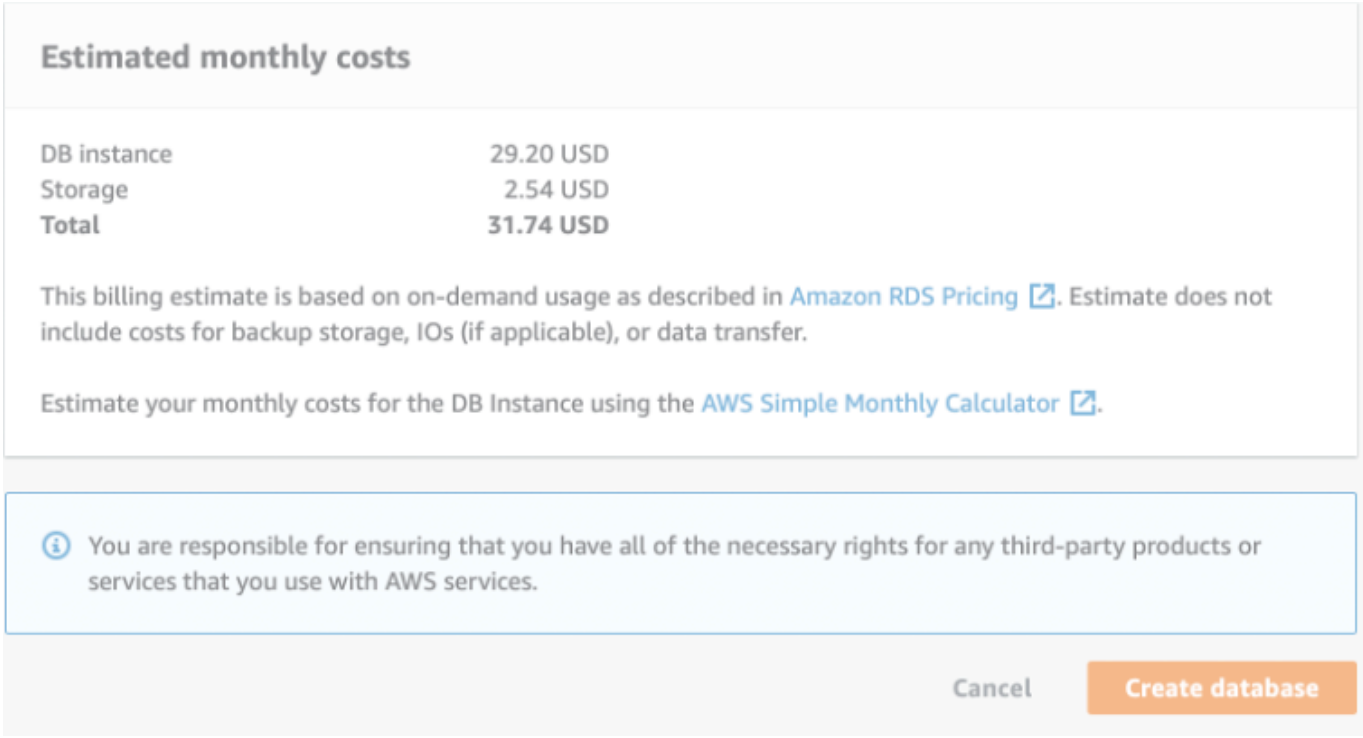


- Under additional configuration. Leave everything as default. Add the database name that you will be using to load the data.



Make sure you uncheck `Enable enhanced monitoring`, as you don't have permissions to do this.


Monitoring
Enable Enhanced monitoring
Enabling Enhanced monitoring metrics are useful when you want to see how different processes or threads use the CPU
Granularity
60 seconds
Monitoring Role
default
Clicking "Create database" will authorize RDS to create the IAM role rds-monitoring-role
Log exports
Select the log types to publish to Amazon CloudWatch Logs
Postgresql log
Upgrade log
IAM role
The following service-linked role is used for publishing logs to CloudWatch Logs.
RDS service-linked role
Ensure that general, slow query, and audit logs are turned on. Error logs are enabled by default. Learn more
Maintenance
Auto minor version upgrade Info
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.
Maintenance window Info
Select the period you want pending modifications or maintenance applied to the database by Amazon RDS.
Select window
No preference
Deletion protection
Enable deletion protection
Protects the database from being deleted accidentally. While this option is enabled, you can't delete the database.



- Click on CREATE DATABASE. Sit back and relax AWS is setting up your database in the cloud.

Note

If you get any error, Check

- If the region is AWS US West (Oregon)
- If you unchecked everything under Additional configuration

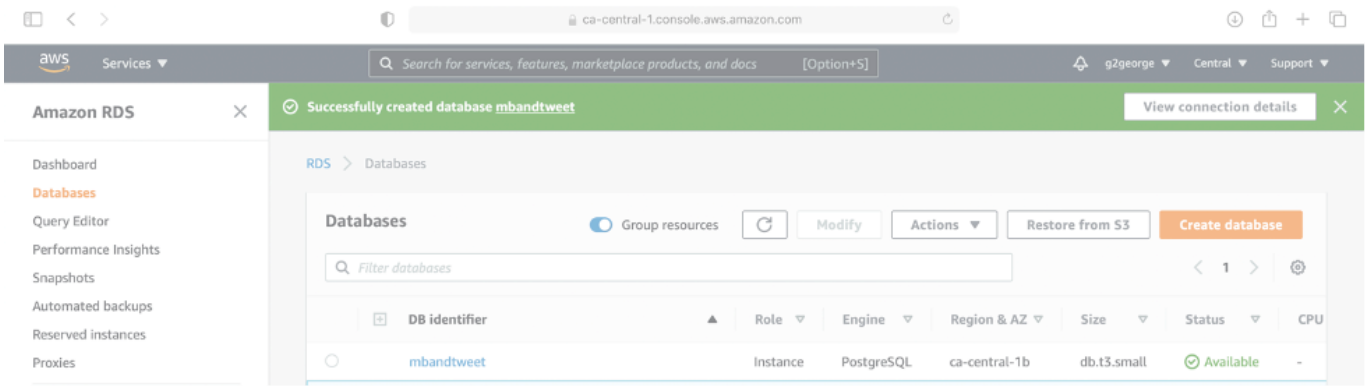
Attention

This process might take some time. For me, it took around 10 min. Better you grab some coffee.

Important

This database is available only in AWS US West (Oregon). So if you come here later to start/stop your database and don't find your database, then it's mostly the case that you are looking in a different region.

After your coffee, you will see your database available.



Once the database is up and running, we need the server details and credentials to access this database.

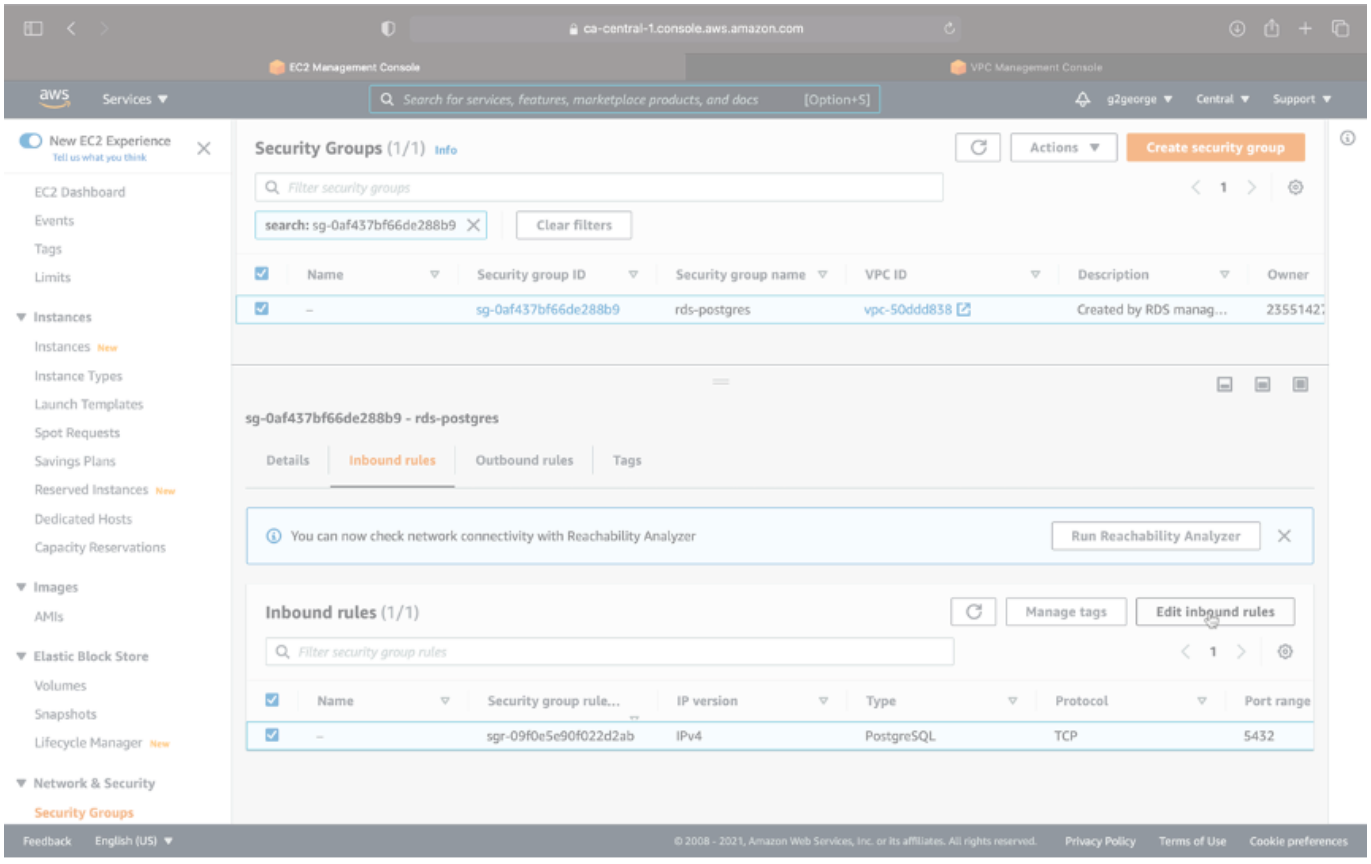
Click on the DB identifier (I marked red in the previous image). This contains details for connecting to the database that you created now.

Capture the endpoint and port; we need to connect to this database.

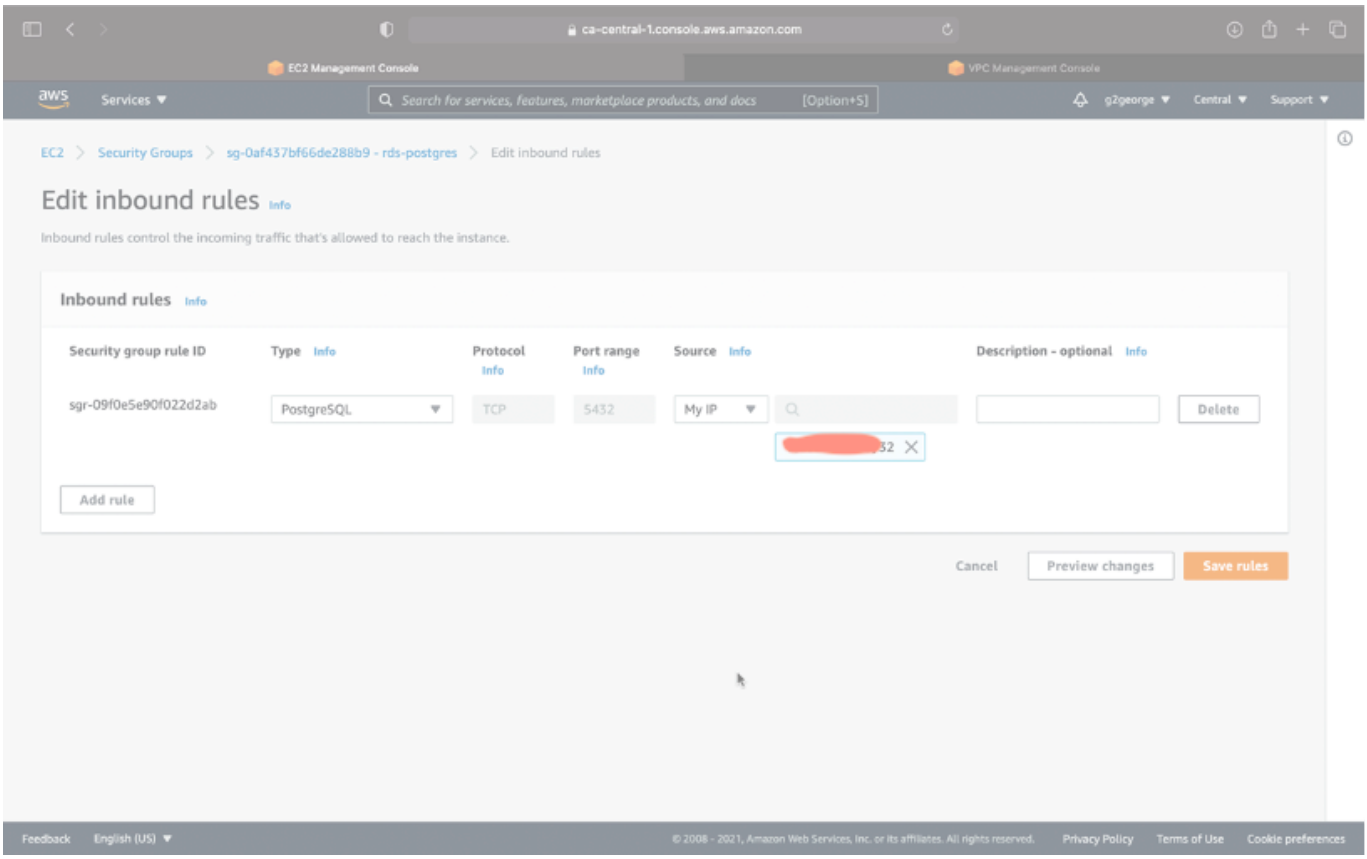
We also need to make some changes to the security group; here, it controls the range of IPs this database can accept connections. Read more about the security group here.

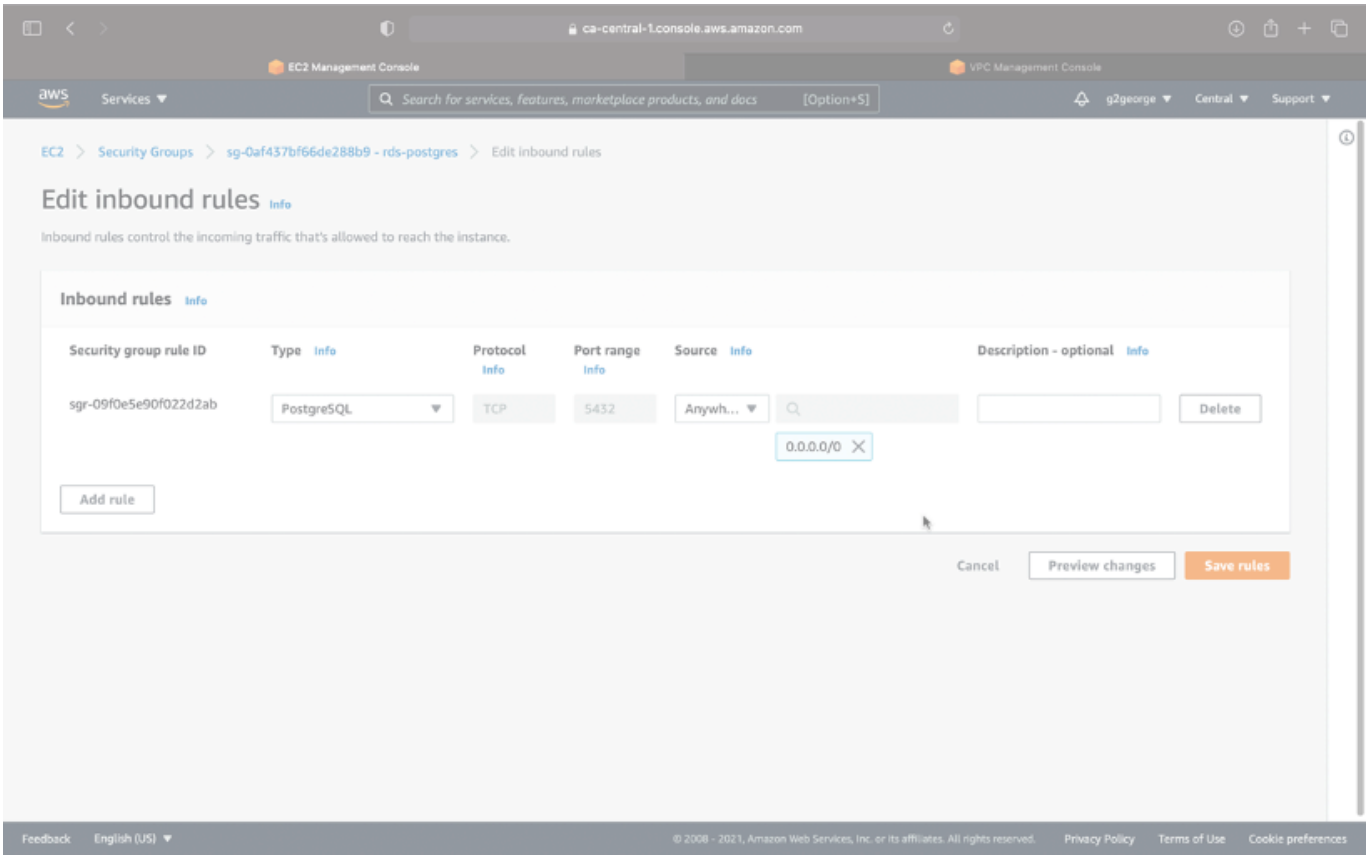
Click on the VPC security groups under the Security. (Check the previous screenshot). This will take you to the Security group settings (the screenshot that you see below); after this, follow the instructions given below.

- Click on edit inbound rules



Before we change this, let me tell you why we want to do this. By default, this database only takes connections from your IP (the laptop you are using now). However, your laptop IP is dynamic and can change based on the Wi-Fi that you are using. Since this is not a production database and nothing sensitive is in there, so it's okay to accept connections from anywhere; do as shown in the following 2 screenshots (not advised to do something like this when you start working in the industry, most of the cases these security group related things would be managed by a different team, so you have to less worry about it as data scientists/ analysts).





Save rules...This completes the task that you need to achieve from the AWS side. Remember you got the hostname and port. Next, we will be setting up the Postgres client in your local computer to access this database.

Here is another set of instructions available online to create and connect to the RDS instance. It's good to check out these if you want to know more about various options. We will be touching upon some of the options in our Lecture 2.

Congratulations, you spun up your database in the cloud !!

Here is a short video that you can use as a reference. (Passcode: `s=Mzw9Qy`)