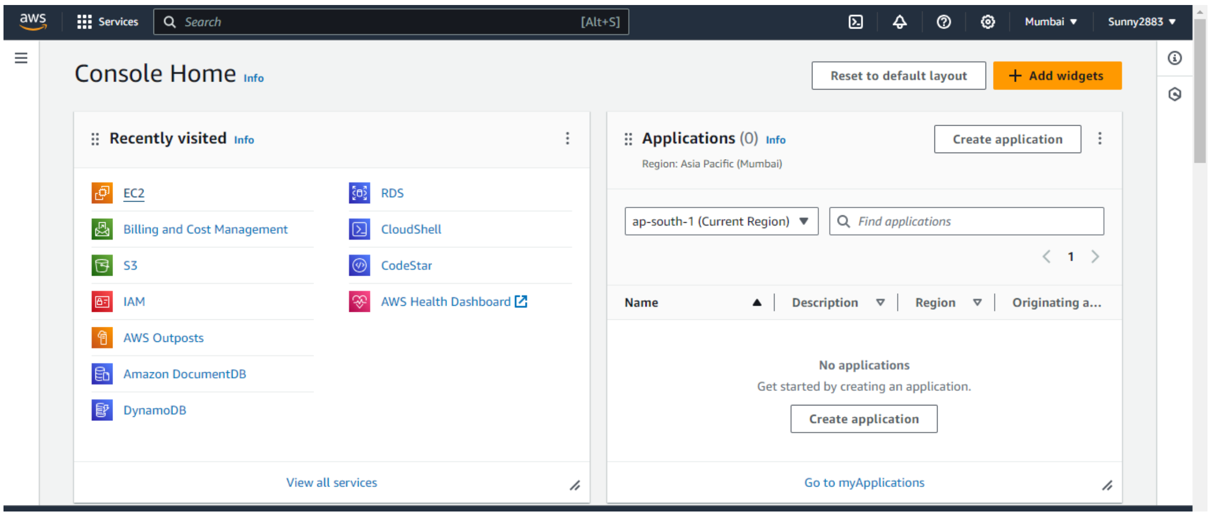
TASK 4: Have to create Linux backup with the use of S3 with specific requirements.

Requirements: Create a backup file of the database and store it in an S3 bucket. Automate the backup process to occur every hour, and apply a retention rule to retain backups for one day.

Step1: Go to the AWS Management Console and sign in using your AWS credentials.


Console Home

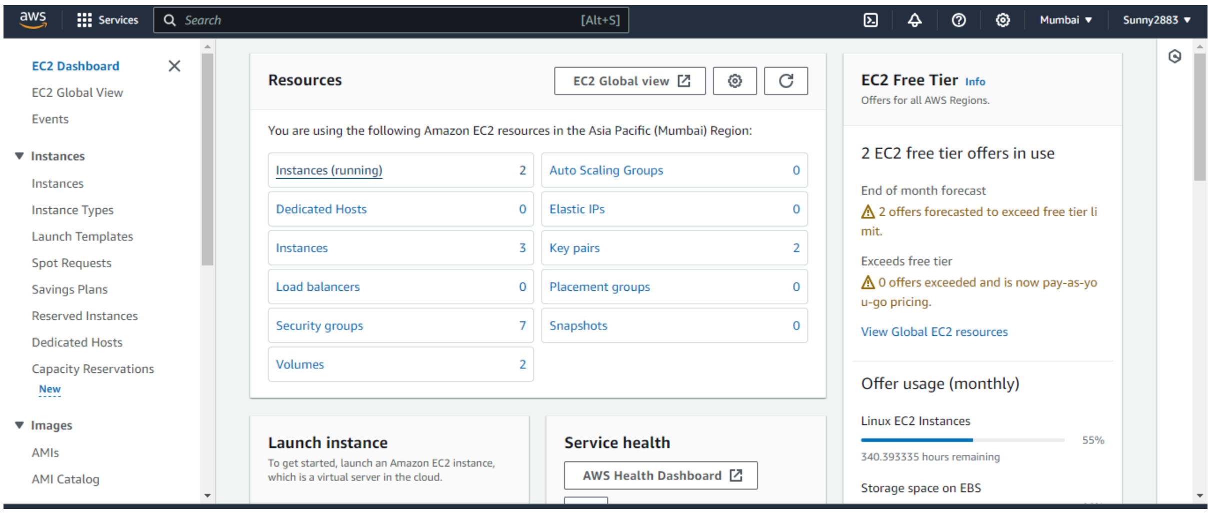
Recently visited: EC2, RDS, Billing and Cost Management, CloudShell, S3, CodeStar, IAM, AWS Health Dashboard, AWS Outposts, Amazon DocumentDB, DynamoDB

Applications (0)
Region: Asia Pacific (Mumbai)
ap-south-1 (Current Region)

Name	Description	Region	Originating a...

No applications
Get started by creating an application.

Step2: From the AWS Management Console, navigate to the EC2 service.

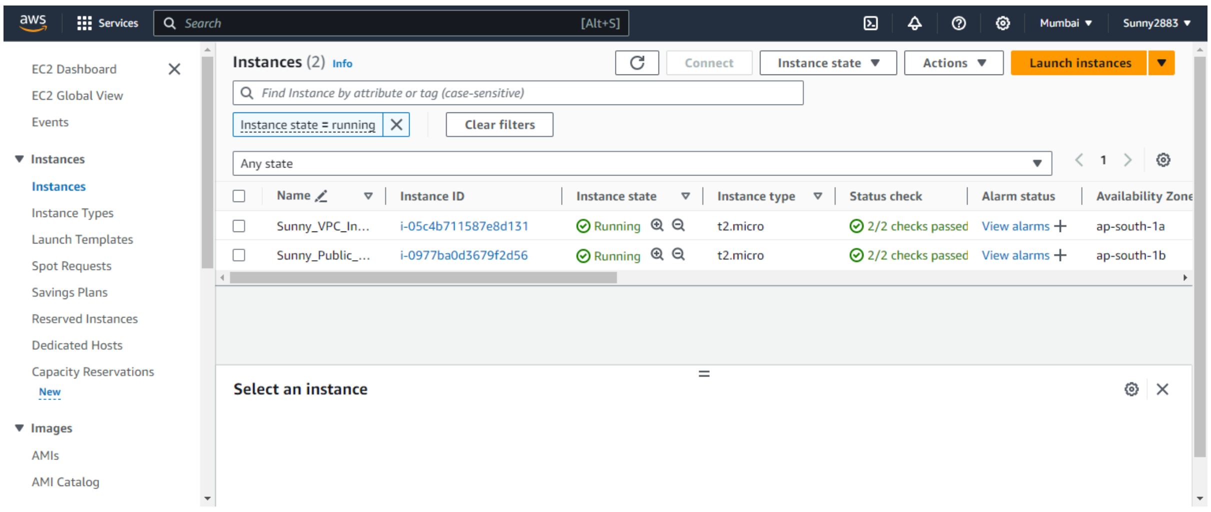

Resources

You are using the following Amazon EC2 resources in the Asia Pacific (Mumbai) Region:

Instances (running)	2	Auto Scaling Groups	0
Dedicated Hosts	0	Elastic IPs	0
Instances	3	Key pairs	2
Load balancers	0	Placement groups	0
Security groups	7	Snapshots	0
Volumes	2		

EC2 Free Tier
Offers for all AWS Regions.
2 EC2 free tier offers in use
End of month forecast
⚠ 2 offers forecasted to exceed free tier limit.
Exceeds free tier
⚠ 0 offers exceeded and is now pay-as-you-go pricing.
Offer usage (monthly)
Linux EC2 Instances 55%
340.393335 hours remaining

Step3: Click on the "Launch Instance" button to begin the process of creating a new EC2 instance.


Instances (2) Info

Find Instance by attribute or tag (case-sensitive)

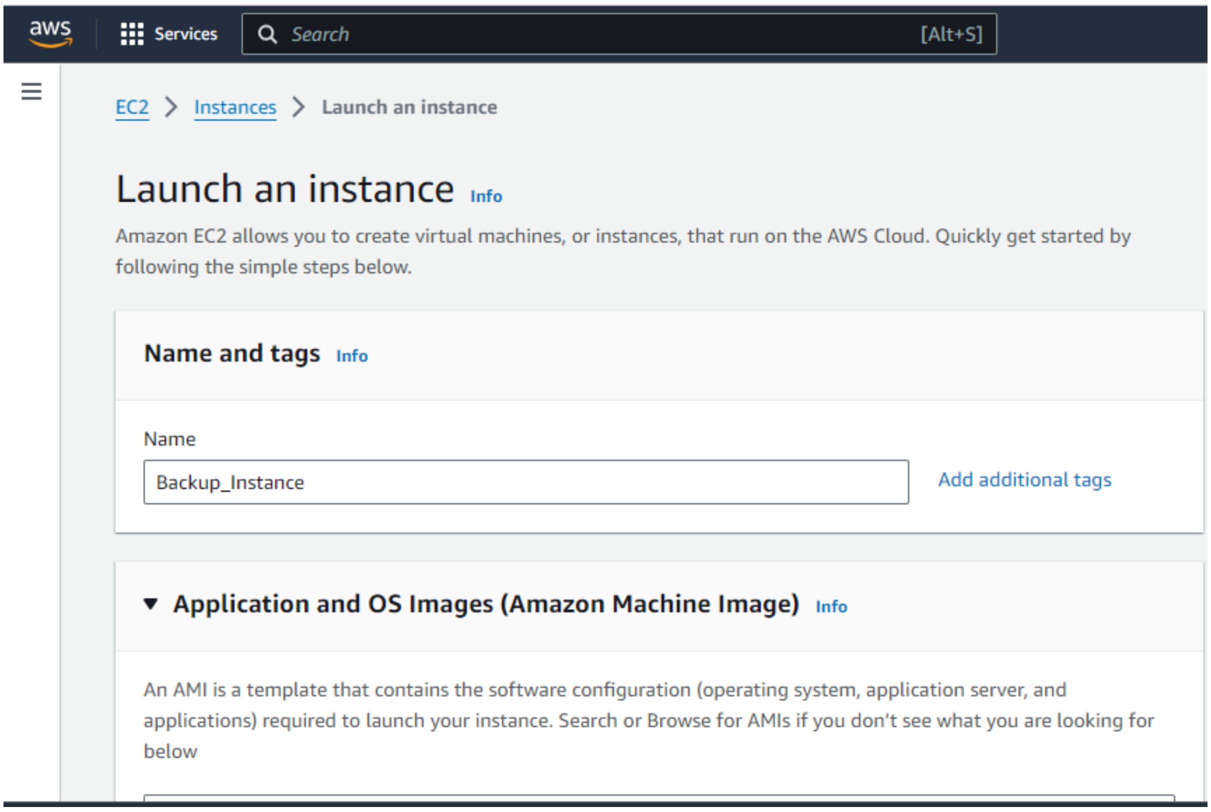
Instance state = running | Clear filters

Any state

Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone
Sunny_VPC_In...	i-05c4b711587e8d131	Running	t2.micro	2/2 checks passed	View alarms +	ap-south-1a
Sunny_Public_...	i-0977ba0d3679f2d56	Running	t2.micro	2/2 checks passed	View alarms +	ap-south-1b

Select an instance

Step4: Give name and additional tag(optional) for your instance.


EC2 > Instances > Launch an instance

Launch an instance Info

Amazon EC2 allows you to create virtual machines, or instances, that run on the AWS Cloud. Quickly get started by following the simple steps below.

Name and tags Info

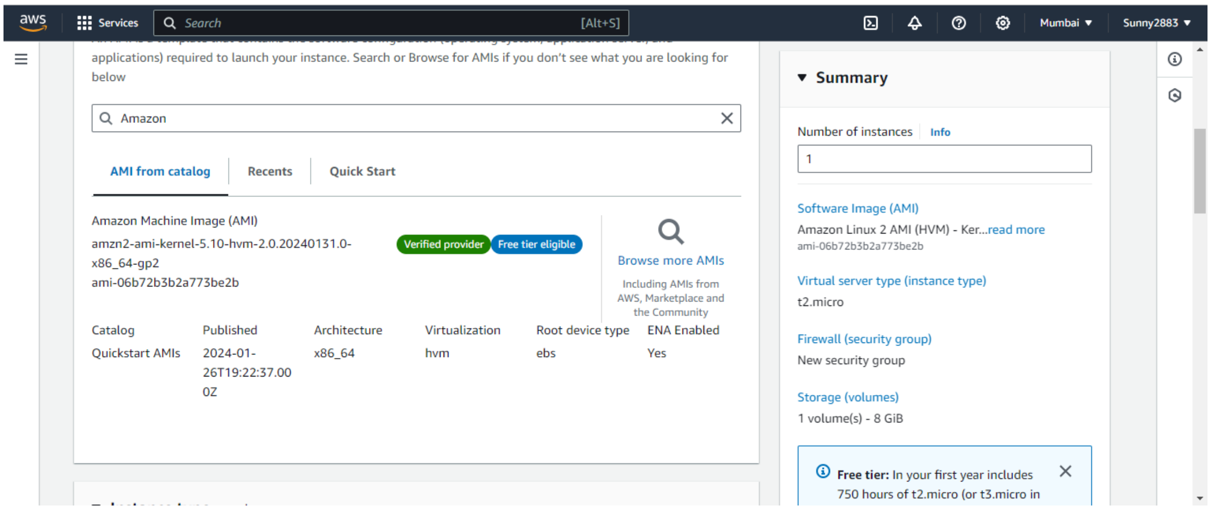
Name: Backup_Instance

Add additional tags

Application and OS Images (Amazon Machine Image) Info

An AMI is a template that contains the software configuration (operating system, application server, and applications) required to launch your instance. Search or Browse for AMIs if you don't see what you are looking for below

Step5: In the "Choose an Amazon Machine Image (AMI)" step, select "Amazon Linux 2 AMI". You can search for it in the search bar or navigate through the options.



applications) required to launch your instance. Search or Browse for AMIs if you don't see what you are looking for below

Amazon

AMI from catalog | Recents | Quick Start

Amazon Machine Image (AMI)
amzn2-ami-kernel-5.10-hvm-2.0.20240131.0-x86_64-gp2
ami-06b72b3b2a773be2b

Verified provider | Free tier eligible

Browse more AMIs
Including AMIs from AWS, Marketplace and the Community

Catalog	Published	Architecture	Virtualization	Root device type	ENA Enabled
Quickstart AMIs	2024-01-26T19:22:37.000Z	x86_64	hvm	ebs	Yes

Summary

Number of instances Info
1

Software Image (AMI)
Amazon Linux 2 AMI (HVM) - Ker...read more
ami-06b72b3b2a773be2b

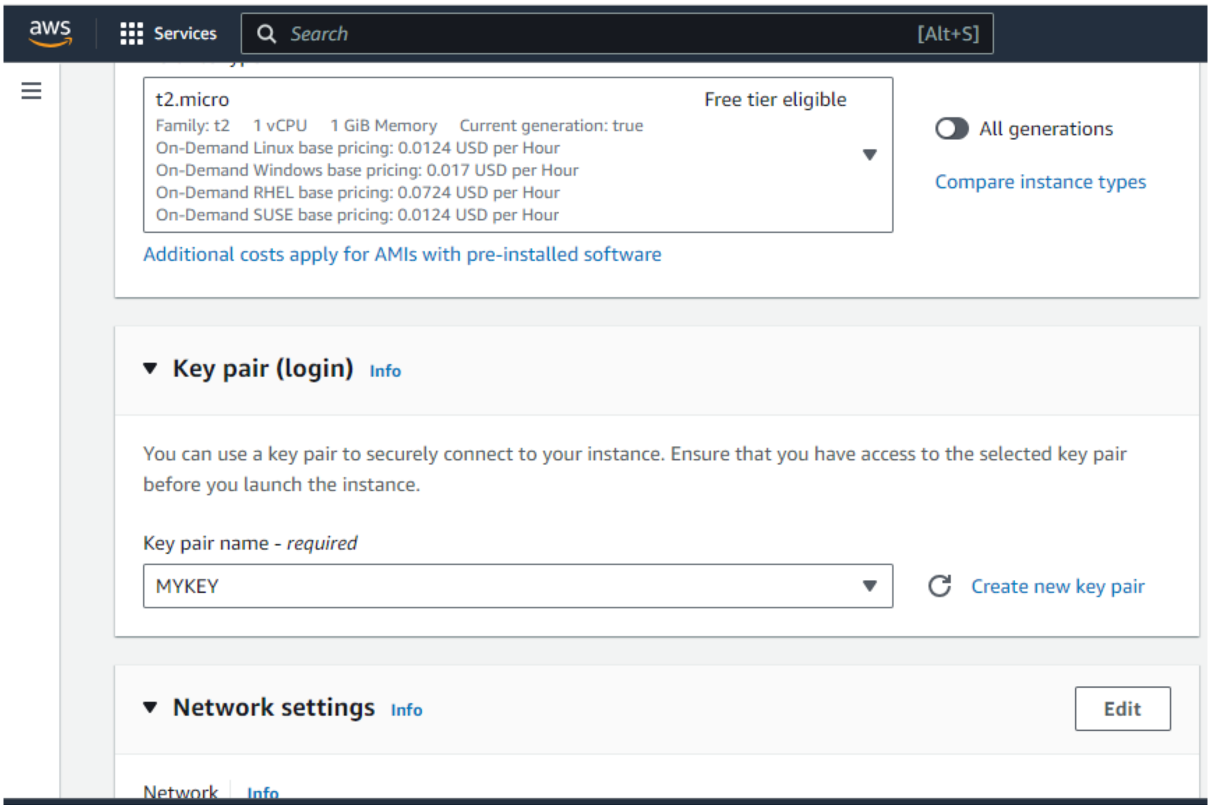
Virtual server type (instance type)
t2.micro

Firewall (security group)
New security group

Storage (volumes)
1 volume(s) - 8 GiB

Free tier: In your first year includes 750 hours of t2.micro (or t3.micro in

Step6: Select "t2.micro" as the instance type. This is a free tier eligible instance type.



t2.micro Free tier eligible
Family: t2 1 vCPU 1 GiB Memory Current generation: true
On-Demand Linux base pricing: 0.0124 USD per Hour
On-Demand Windows base pricing: 0.017 USD per Hour
On-Demand RHEL base pricing: 0.0724 USD per Hour
On-Demand SUSE base pricing: 0.0124 USD per Hour

All generations

Compare instance types

Additional costs apply for AMIs with pre-installed software

Key pair (login) Info

You can use a key pair to securely connect to your instance. Ensure that you have access to the selected key pair before you launch the instance.

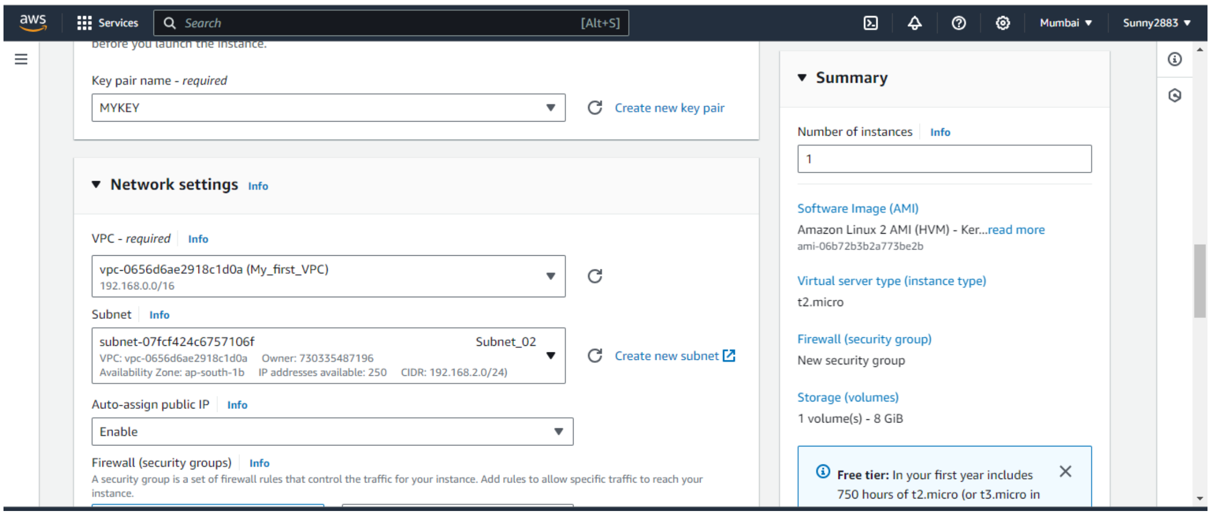
Key pair name - *required*
MYKEY

Create new key pair

Network settings Info
Edit

Network Info

Step7: Select Key pair If you don't have an existing key pair, create a new one. This key pair will be used to securely connect to your EC2 instance via SSH.


Key pair name - *required*
MYKEY
Create new key pair

Network settings Info

VPC - *required* Info
vpc-0656d6ae2918c1d0a (My_first_VPC)
192.168.0.0/16

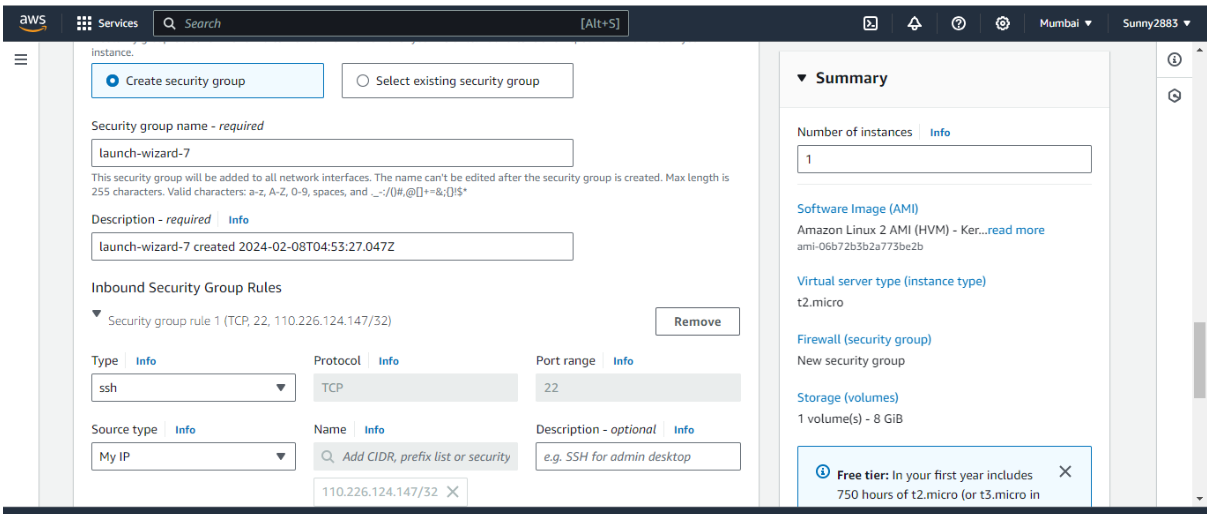
Subnet Info
subnet-07fcf424c6757106f Subnet_02
VPC: vpc-0656d6ae2918c1d0a Owner: 730335487196
Availability Zone: ap-south-1b IP addresses available: 250 CIDR: 192.168.2.0/24)
Create new subnet

Auto-assign public IP Info
Enable

Firewall (security groups) Info
A security group is a set of firewall rules that control the traffic for your instance. Add rules to allow specific traffic to reach your instance.

Summary
Number of instances Info
1
Software Image (AMI)
Amazon Linux 2 AMI (HVM) - Ker...read more
ami-06b72b3b2a773be2b
Virtual server type (instance type)
t2.micro
Firewall (security group)
New security group
Storage (volumes)
1 volume(s) - 8 GiB
Free tier: In your first year includes 750 hours of t2.micro (or t3.micro in

Step:8 Create a new security group or select an existing one. This group defines the firewall rules that control the traffic to your instance.


Create security group
Select existing security group

Security group name - *required*
launch-wizard-7
This security group will be added to all network interfaces. The name can't be edited after the security group is created. Max length is 255 characters. Valid characters: a-z, A-Z, 0-9, spaces, and ._-:/()#,@[]+=&;{}!$*

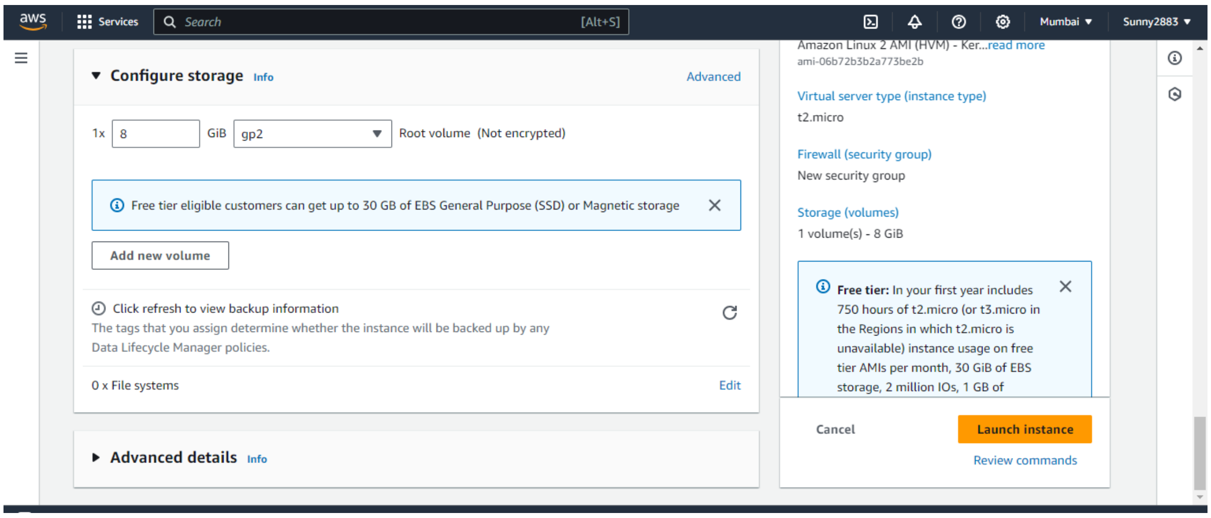
Description - *required* Info
launch-wizard-7 created 2024-02-08T04:53:27.047Z

Inbound Security Group Rules
Security group rule 1 (TCP, 22, 110.226.124.147/32)
Remove

Type Info: ssh
Protocol Info: TCP
Port range Info: 22
Source type Info: My IP
Name Info: Add CIDR, prefix list or security
Description - *optional* Info: e.g. SSH for admin desktop
110.226.124.147/32

Summary
Number of instances Info
1
Software Image (AMI)
Amazon Linux 2 AMI (HVM) - Ker...read more
ami-06b72b3b2a773be2b
Virtual server type (instance type)
t2.micro
Firewall (security group)
New security group
Storage (volumes)
1 volume(s) - 8 GiB
Free tier: In your first year includes 750 hours of t2.micro (or t3.micro in

Step9: Review the configuration of your instance to ensure everything is set up as desired. Click "Launch" when you're ready.



Configure storage Info Advanced

1x 8 GiB gp2 Root volume (Not encrypted)

Free tier eligible customers can get up to 30 GB of EBS General Purpose (SSD) or Magnetic storage

Add new volume

Click refresh to view backup information
The tags that you assign determine whether the instance will be backed up by any Data Lifecycle Manager policies.

0 x File systems Edit

Advanced details Info

Amazon Linux 2 AMI (HVM) - Ker...read more
ami-06b72b3b2a773be2b

Virtual server type (instance type)
t2.micro

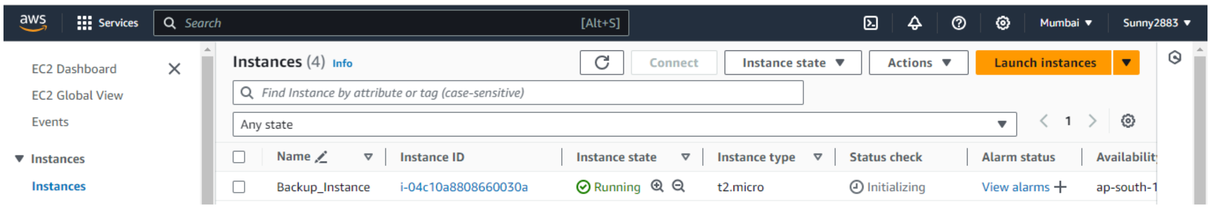
Firewall (security group)
New security group

Storage (volumes)
1 volume(s) - 8 GiB

Free tier: In your first year includes 750 hours of t2.micro (or t3.micro in the Regions in which t2.micro is unavailable) instance usage on free tier AMIs per month, 30 GiB of EBS storage, 2 million IOs, 1 GB of

Cancel Launch instance
Review commands

Step10: You'll be redirected to the Instances dashboard where you can view the status of your newly launched instance.

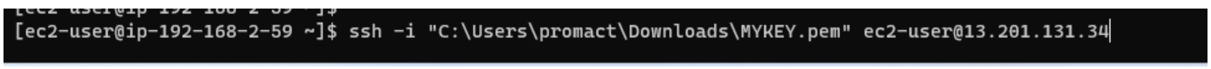


Instances (4) Info

	Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availabilit
☐	Backup_Instance	i-04c10a8808660030a	Running	t2.micro	Initializing	View alarms +	ap-south-1

Step11: Once the instance state transitions to "running", you can connect to it using SSH. You'll need the public IP address or public DNS name of the instance along with the private key associated with the key pair you selected.

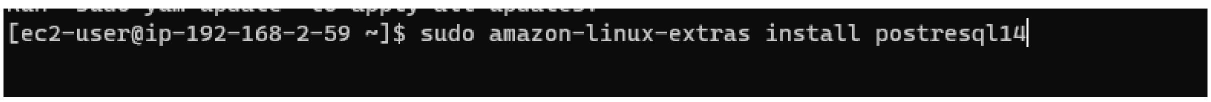
ssh -i C:\Users\promact\Downloads\MYKEY.pem ec2-user@13.201.131.34



```
[ec2-user@ip-192-168-2-59 ~]$ ssh -i "C:\Users\promact\Downloads\MYKEY.pem" ec2-user@13.201.131.34
```

Step12:Once the connection is established you can install postgresql. PostgreSQL packages for Amazon Linux 2 are available in the PostgreSQL yum repository. Use the following commands to add the repository and install PostgreSQL 14:

sudo amazon-linux-extras install postgresql14

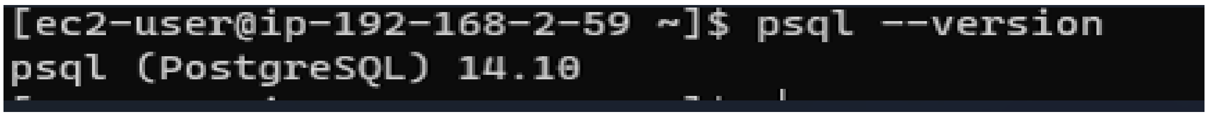


```
[ec2-user@ip-192-168-2-59 ~]$ sudo amazon-linux-extras install postresql14
```

Step13 :Check PostgreSQL Version

To check the version of PostgreSQL installed on your system, run the following command:

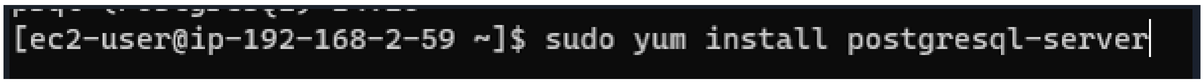
psql --version



```
[ec2-user@ip-192-168-2-59 ~]$ psql --version
psql (PostgreSQL) 14.10
```

Step14: Install PostgreSQL server using the following command:

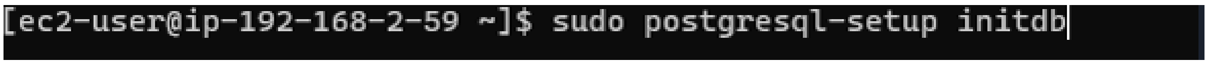
sudo yum install postgresql-server



```
[ec2-user@ip-192-168-2-59 ~]$ sudo yum install postgresql-server
```

Step15: Initialize the PostgreSQL database using the following command:

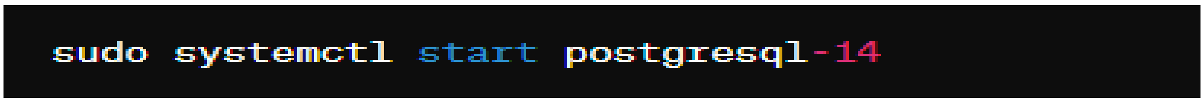
sudo postgresql-setup initdb



```
[ec2-user@ip-192-168-2-59 ~]$ sudo postgresql-setup initdb
```

Step16: Start PostgreSQL Service:

sudo systemctl start postgresql

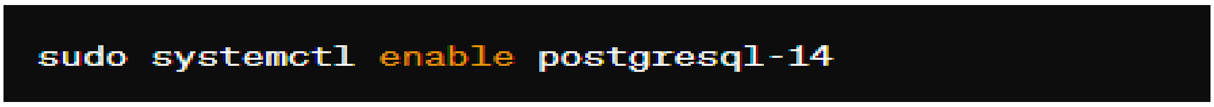


```
sudo systemctl start postgresql-14
```

Step17: Enable PostgreSQL Service to Start on Boot

Ensure that the PostgreSQL service starts automatically on system boot by running the following command:

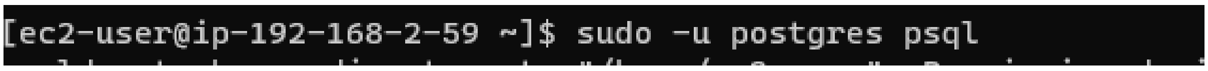
sudo systemctl enable postgresql



```
sudo systemctl enable postgresql-14
```

Step18: First, access the PostgreSQL prompt by running

Sudo -u postgres psql



```
[ec2-user@ip-192-168-2-59 ~]$ sudo -u postgres psql
```

Step19: Identify the Database:

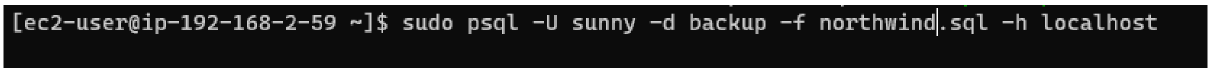
Identify the specific database you want to download. This could be a database available online or provided by a source.

Step20: Download the Database File:

Use a tool like wget or curl to download the database file from the internet

scp -i C:\Users\promact\Downloads\MYKEY.pem "C:\Users\promact\Downloads\northwind.sql" ec2-user@13.201.131.34:/home/ec2-user

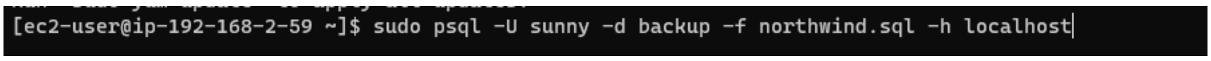
northwind.sql



```
[ec2-user@ip-192-168-2-59 ~]$ sudo psql -U sunny -d backup -f northwind.sql -h localhost
```

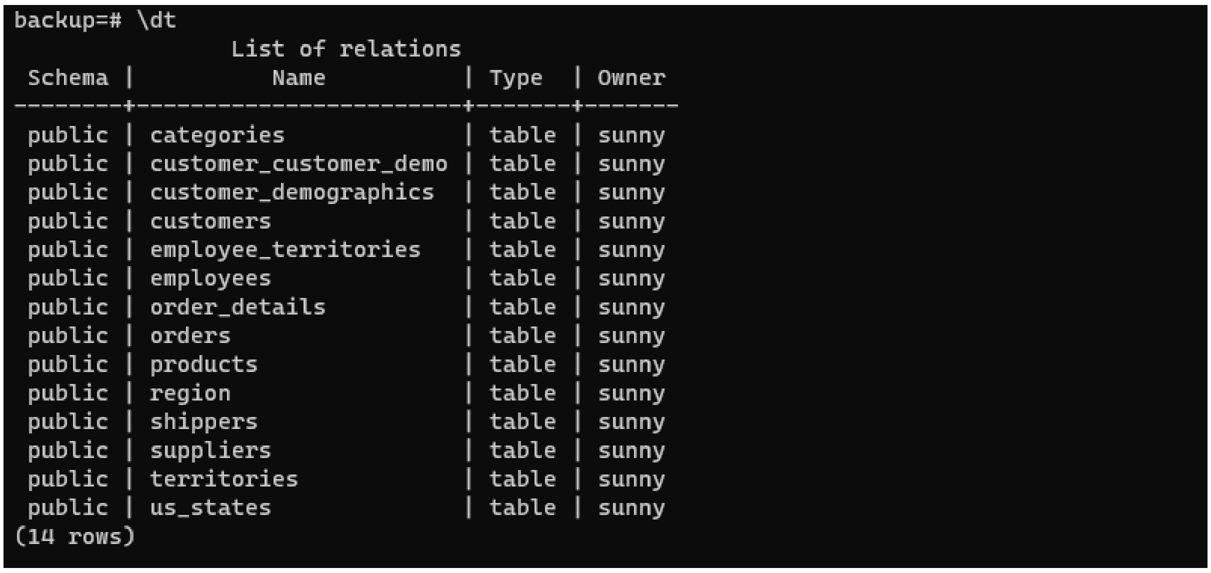
Step21: Restore the extracted file in to the database which is created using command:

sudo psql -U sunny -d backup -f northwind.sql -h localhost



```
[ec2-user@ip-192-168-2-59 ~]$ sudo psql -U sunny -d backup -f northwind.sql -h localhost
```

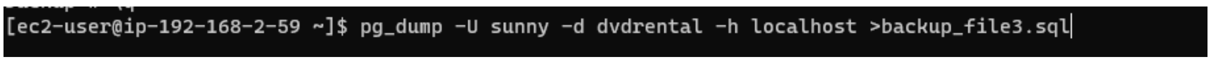
You are now connected to the database backup as user sunny:



```
backup=# \dt
                List of relations
 Schema |          Name           | Type  | Owner
--------+-------------------------+-------+-------
 public | categories              | table | sunny
 public | customer_customer_demo  | table | sunny
 public | customer_demographics   | table | sunny
 public | customers               | table | sunny
 public | employee_territories    | table | sunny
 public | employees               | table | sunny
 public | order_details           | table | sunny
 public | orders                  | table | sunny
 public | products                | table | sunny
 public | region                  | table | sunny
 public | shippers                | table | sunny
 public | suppliers               | table | sunny
 public | territories             | table | sunny
 public | us_states               | table | sunny
(14 rows)
```

Step22: Use the pg_dump command to create a backup of your database. The basic syntax for pg_dump is:

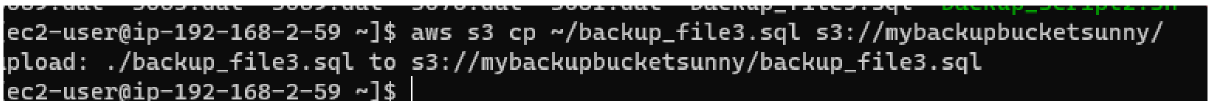
pg_dump -U sunny -d dvdrental -h localhost >backup_file3.sql



```
[ec2-user@ip-192-168-2-59 ~]$ pg_dump -U sunny -d dvdrental -h localhost >backup_file3.sql
```

Step23: If your S3 bucket is already created, you can upload your PostgreSQL database backup file to the bucket using the AWS s3 cp command:

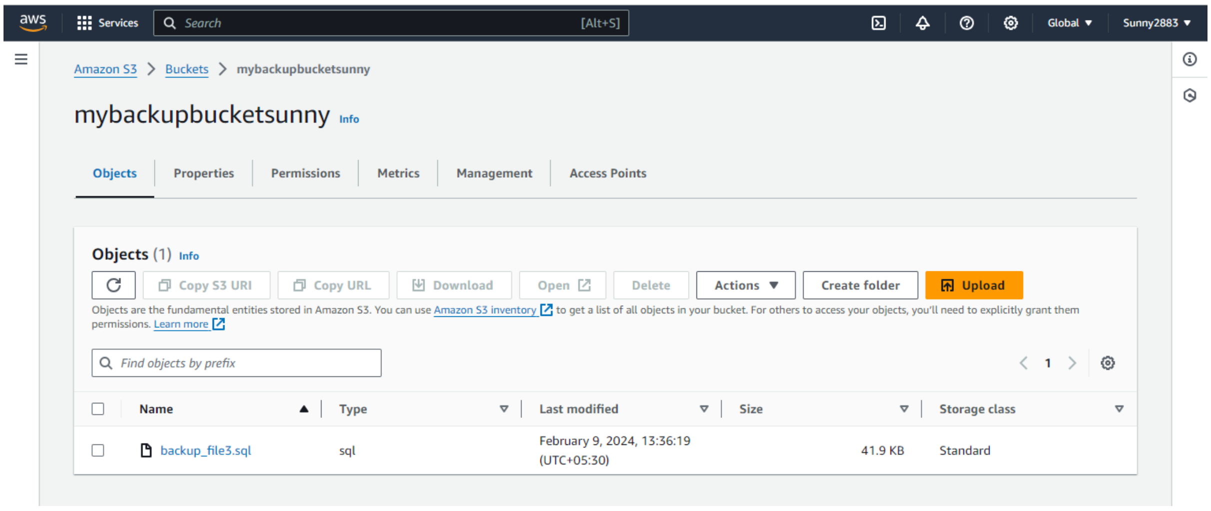
aws s3 cp ~/backup_file3.sql s3://mybackupbucketsunny/



```
ec2-user@ip-192-168-2-59 ~]$ aws s3 cp ~/backup_file3.sql s3://mybackupbucketsunny/
pload: ./backup_file3.sql to s3://mybackupbucketsunny/backup_file3.sql
ec2-user@ip-192-168-2-59 ~]$
```

Step24: Backup File Stored to S3 Bucket Successfully

Confirm that the backup file has been successfully stored in the S3 bucket.



mybackupbucketsunny Info

Objects (1) Info

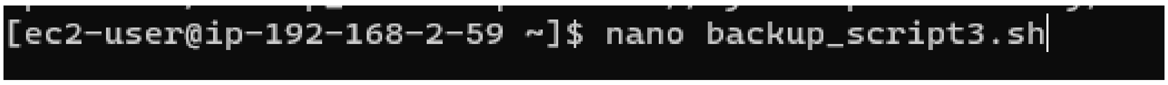
Objects are the fundamental entities stored in Amazon S3. You can use Amazon S3 inventory to get a list of all objects in your bucket. For others to access your objects, you'll need to explicitly grant them permissions. Learn more

Name	Type	Last modified	Size	Storage class
backup_file3.sql	sql	February 9, 2024, 13:36:19 (UTC+05:30)	41.9 KB	Standard

Step25: Create a Backup_script3.sh File

Create a new shell script named backup_script3.sh using your preferred text editor. You can use the nano command to create an empty file and then edit it:

nano backup_script3.sh

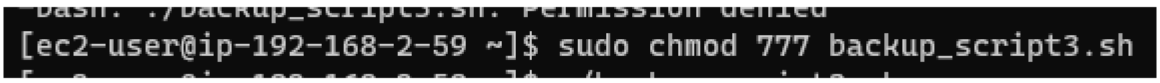


```
[ec2-user@ip-192-168-2-59 ~]$ nano backup_script3.sh
```

Step26: Change Permissions of Backup_script2.sh File

Change the permissions of the backup_script2.sh file to make it executable. Use the chmod command to modify the file permissions:

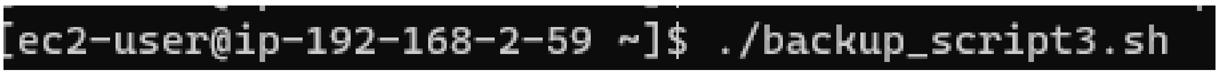
sudo chmod 777 backup_script3.sh



```
[ec2-user@ip-192-168-2-59 ~]$ sudo chmod 777 backup_script3.sh
```

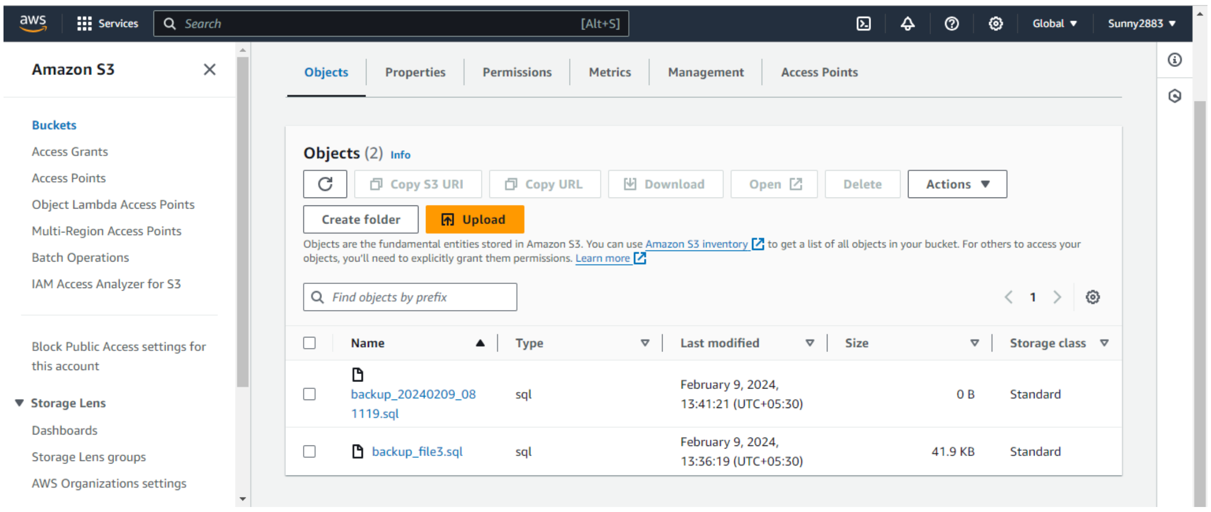
Step27: Execute backup_script2.sh file.

./backup_script3.sh



```
[ec2-user@ip-192-168-2-59 ~]$ ./backup_script3.sh
```

Backup file stored to s3 successfully.



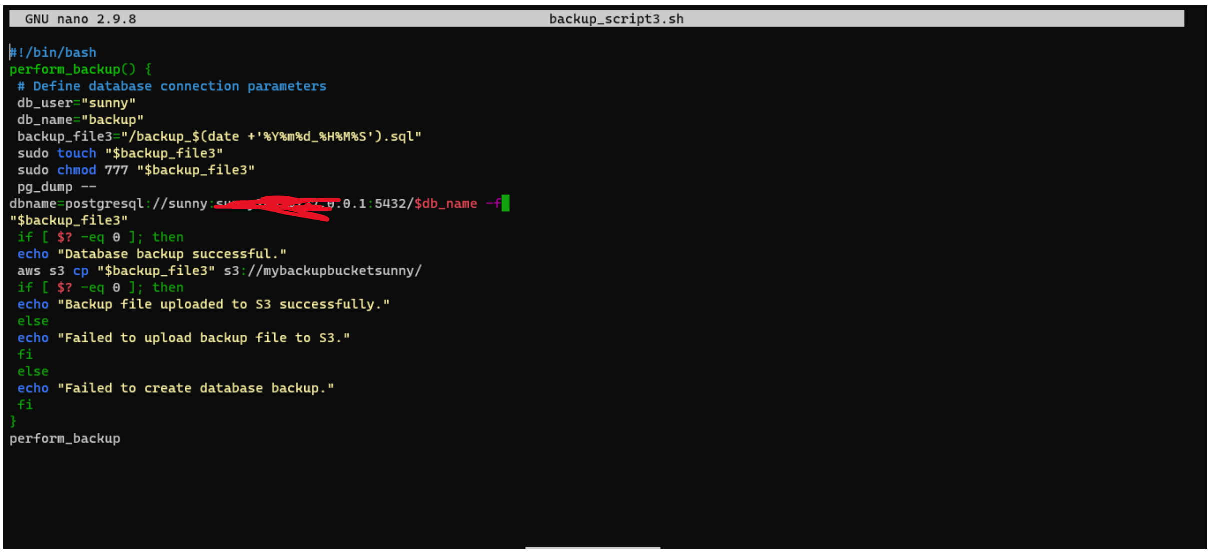
Objects are the fundamental entities stored in Amazon S3. You can use Amazon S3 inventory to get a list of all objects in your bucket. For others to access your objects, you'll need to explicitly grant them permissions. Learn more

Name	Type	Last modified	Size	Storage class
backup_20240209_081119.sql	sql	February 9, 2024, 13:41:21 (UTC+05:30)	0 B	Standard
backup_file3.sql	sql	February 9, 2024, 13:36:19 (UTC+05:30)	41.9 KB	Standard

Step28: write a shell script:

nano backup_script.sh

```
#!/bin/bash
perform_backup() {
 # Define database connection parameters
 db_user="sunny"
 db_name="backup"
 backup_file3="/backup_$(date +'%Y%m%d_%H%M%S').sql"
 sudo touch "$backup_file3"
 sudo chmod 777 "$backup_file3"
 pg_dump --
dbname=postgresql://sunny:pswd@127.0.0.1:5432/$db_name -f
"$backup_file3"
 if [ $? -eq 0 ]; then
 echo "Database backup successful."
 aws s3 cp "$backup_file3" s3://mybackupbucketsunny/
 if [ $? -eq 0 ]; then
 echo "Backup file uploaded to S3 successfully."
 else
 echo "Failed to upload backup file to S3."
 fi
 else
 echo "Failed to create database backup."
 fi
}
perform_backup
```

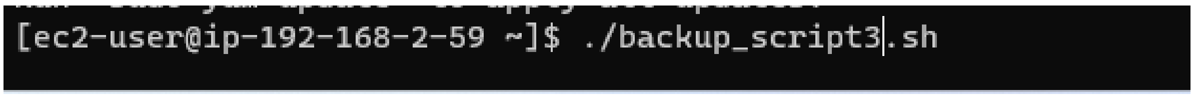


```
  GNU nano 2.9.8                                backup_script3.sh

#!/bin/bash
perform_backup() {
 # Define database connection parameters
 db_user="sunny"
 db_name="backup"
 backup_file3="/backup_$(date +'%Y%m%d_%H%M%S').sql"
 sudo touch "$backup_file3"
 sudo chmod 777 "$backup_file3"
 pg_dump --
dbname=postgresql://sunny:[redacted].0.1:5432/$db_name -f
"$backup_file3"
 if [ $? -eq 0 ]; then
 echo "Database backup successful."
 aws s3 cp "$backup_file3" s3://mybackupbucketsunny/
 if [ $? -eq 0 ]; then
 echo "Backup file uploaded to S3 successfully."
 else
 echo "Failed to upload backup file to S3."
 fi
 else
 echo "Failed to create database backup."
 fi
}
perform_backup
```

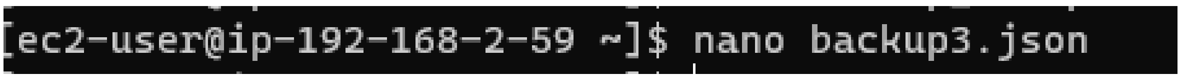
Step29: Execute the file:

./backup_script3.sh



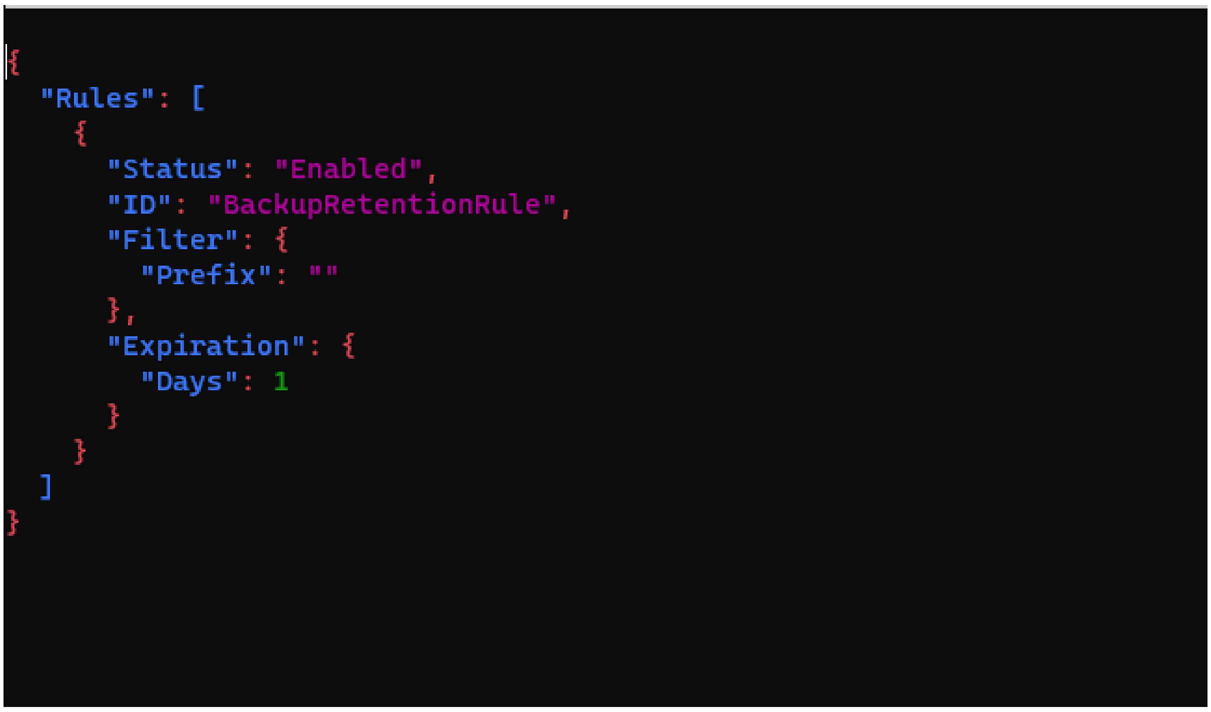
```
[ec2-user@ip-192-168-2-59 ~]$ ./backup_script3.sh
```

Step30: Create a backup.json file for retention of 1 day:



```
[ec2-user@ip-192-168-2-59 ~]$ nano backup3.json
```

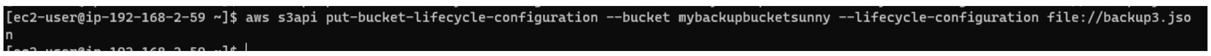
```
{
  "Rules": [
    {
      "Status": "Enabled",
      "ID": "BackupRetentionRule",
      "Filter": {
        "Prefix": ""
      },
      "Expiration": {
        "Days": 1
      }
    }
  ]
}
```



```
{
  "Rules": [
    {
      "Status": "Enabled",
      "ID": "BackupRetentionRule",
      "Filter": {
        "Prefix": ""
      },
      "Expiration": {
        "Days": 1
      }
    }
  ]
}
```

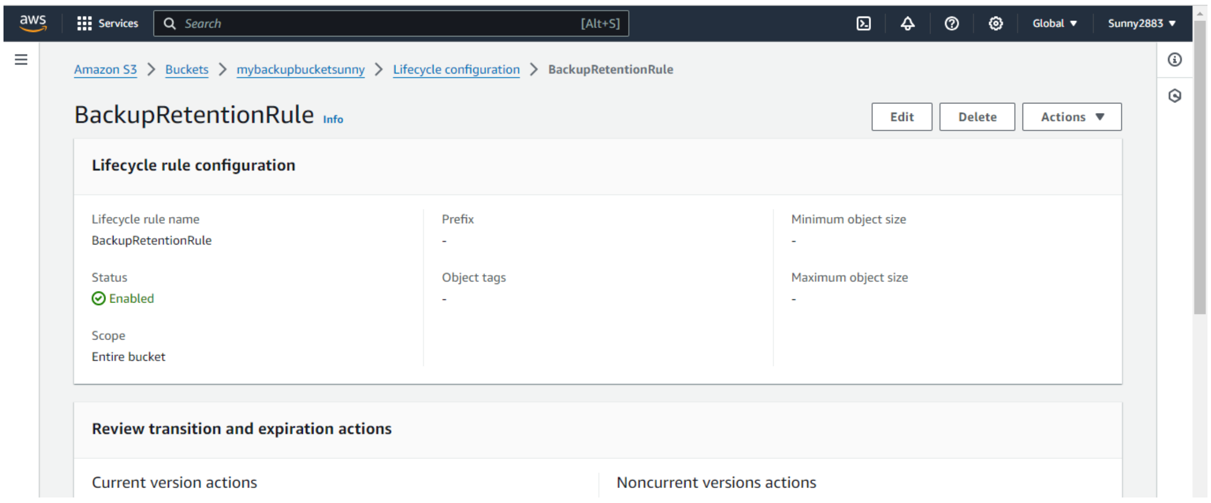
Step31: Run the command for the retention rule;

aws s3api put-bucket-lifecycle-configuration --bucket mybackupbucketsunny --lifecycle-configuration file://backup3.json



```
[ec2-user@ip-192-168-2-59 ~]$ aws s3api put-bucket-lifecycle-configuration --bucket mybackupbucketsunny --lifecycle-configuration file://backup3.jso
n
```

Retention rule for 1 day applied:



Amazon S3 > Buckets > mybackupbucketsunny > Lifecycle configuration > BackupRetentionRule

BackupRetentionRule Info

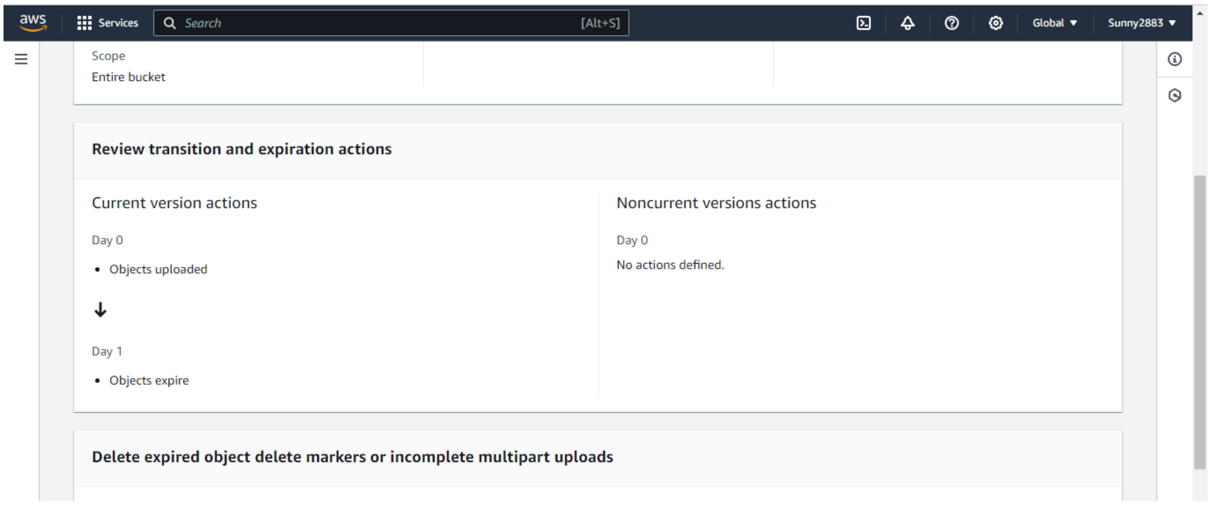
Edit Delete Actions

Lifecycle rule configuration

Lifecycle rule name BackupRetentionRule	Prefix -	Minimum object size -
Status Enabled	Object tags -	Maximum object size -
Scope Entire bucket		

Review transition and expiration actions

Current version actions | Noncurrent versions actions



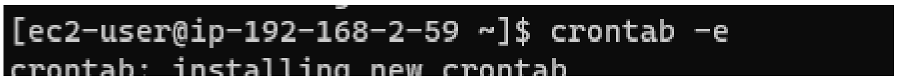
Scope
Entire bucket

Review transition and expiration actions

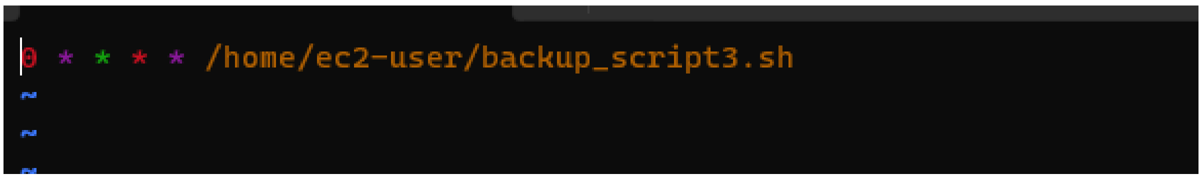
Current version actions	Noncurrent versions actions
Day 0 • Objects uploaded ↓ Day 1 • Objects expire	Day 0 No actions defined.

Delete expired object delete markers or incomplete multipart uploads

Step32: Open the crontab file for editing Add a cron job for every hour :

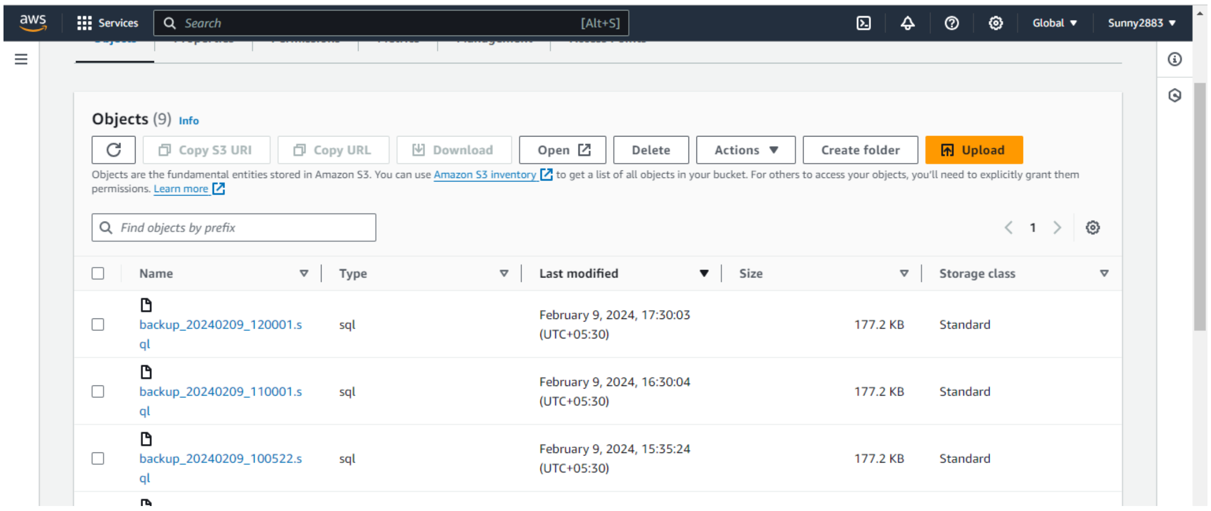


```
[ec2-user@ip-192-168-2-59 ~]$ crontab -e
crontab: installing new crontab
```



```
0 * * * * /home/ec2-user/backup_script3.sh
~
~
~
```

The backup process is set to occur automatically every hour, ensuring that backups are performed consistently and at regular intervals.



Objects (9) Info

Objects are the fundamental entities stored in Amazon S3. You can use Amazon S3 inventory to get a list of all objects in your bucket. For others to access your objects, you'll need to explicitly grant them permissions. Learn more

Name	Type	Last modified	Size	Storage class
backup_20240209_120001.sql	sql	February 9, 2024, 17:30:03 (UTC+05:30)	177.2 KB	Standard
backup_20240209_110001.sql	sql	February 9, 2024, 16:30:04 (UTC+05:30)	177.2 KB	Standard
backup_20240209_100522.sql	sql	February 9, 2024, 15:35:24 (UTC+05:30)	177.2 KB	Standard