**DevOps Tasks 1**

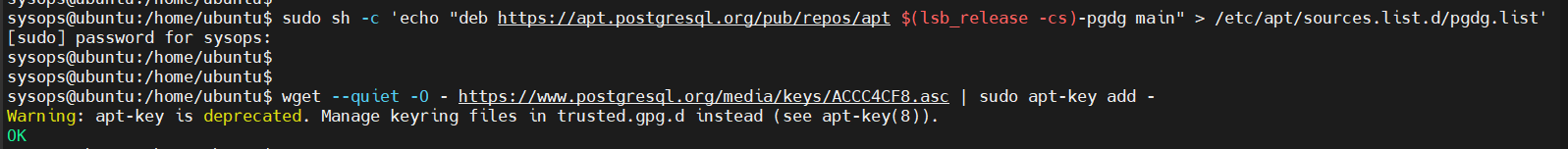
**Write bash script to loop on all databases into a postgresql server then backup and compress each database into a different directory then upload it to a S3 bucket**

**STEP1: Install and Create Postgresql database.**

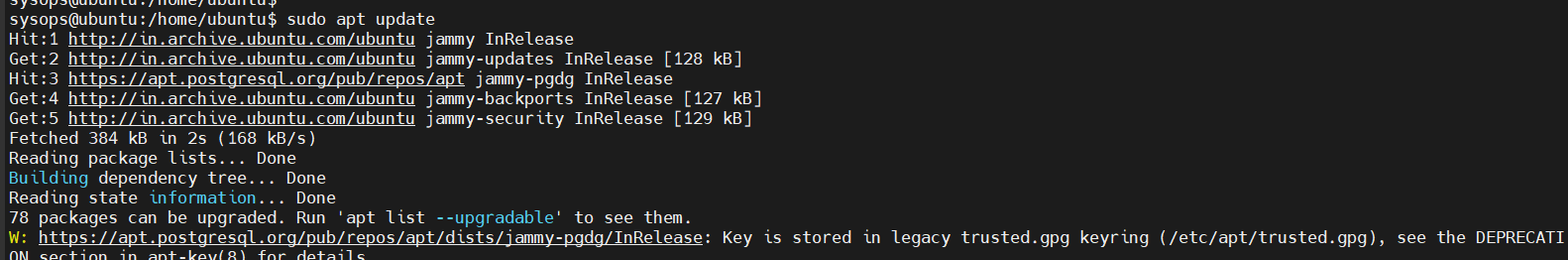
$ sudo sh -c 'echo "deb https://apt.postgresql.org/pub/repos/apt $(lsb\_release -cs)-pgdg main" > /etc/apt/sources.list.d/pgdg.list'

$ wget --quiet -O - https://www.postgresql.org/media/keys/ACCC4CF8.asc | sudo apt-key add -

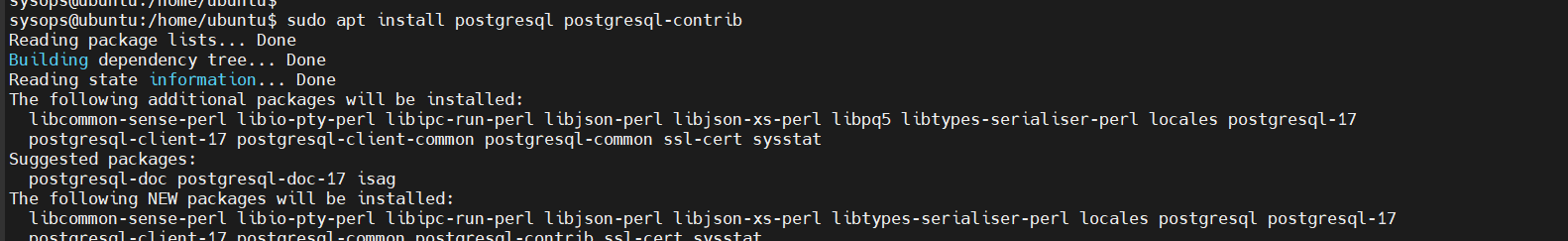
Warning: apt-key is deprecated. Manage keyring files in trusted.gpg.d instead (see apt-key(8)).



$ sudo apt update



$ sudo apt install postgresql postgresql-contrib

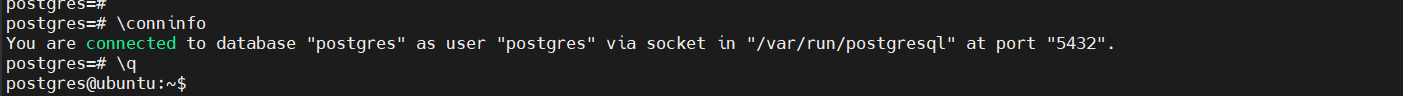


$ sudo su – postgres

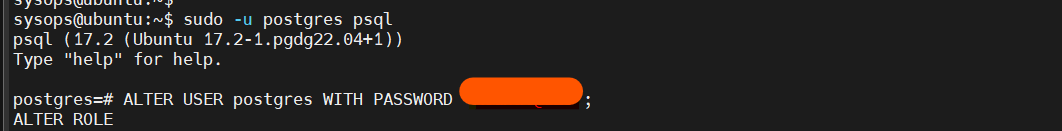
postgres=# CREATE DATABASE TestDB;



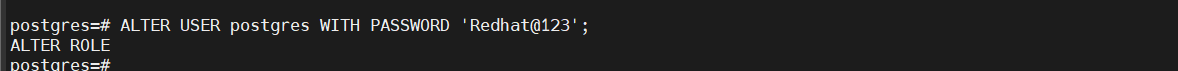
postgres=# \conninfo



$ sudo -u postgres psql



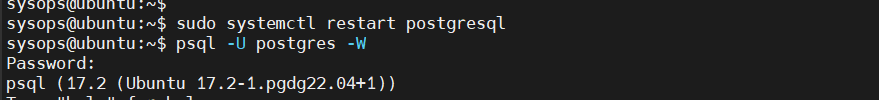
postgres=# ALTER USER postgres WITH PASSWORD 'Redhat@123';



$ sudo systemctl restart postgresql

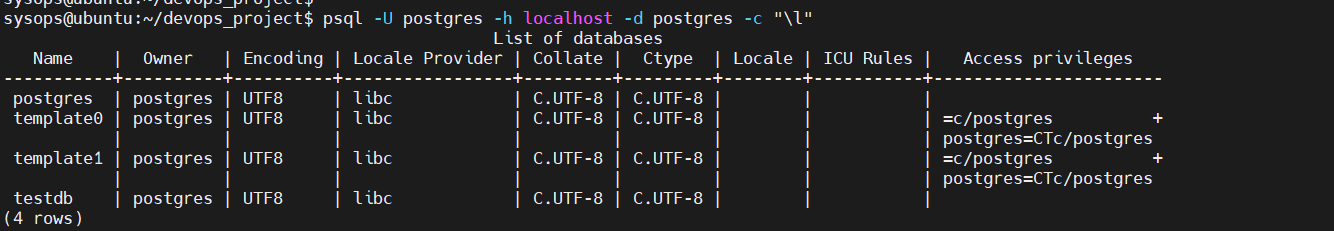


$ psql -U postgres -W



postgres=# \q

$ psql -U postgres -h localhost -d postgres -c "\l"



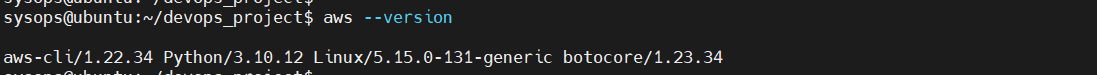
**STEP 2: Install and Configure AWS cli**

$ sudo apt update

$ sudo apt install awscli -y



$ aws –version



$ aws configure

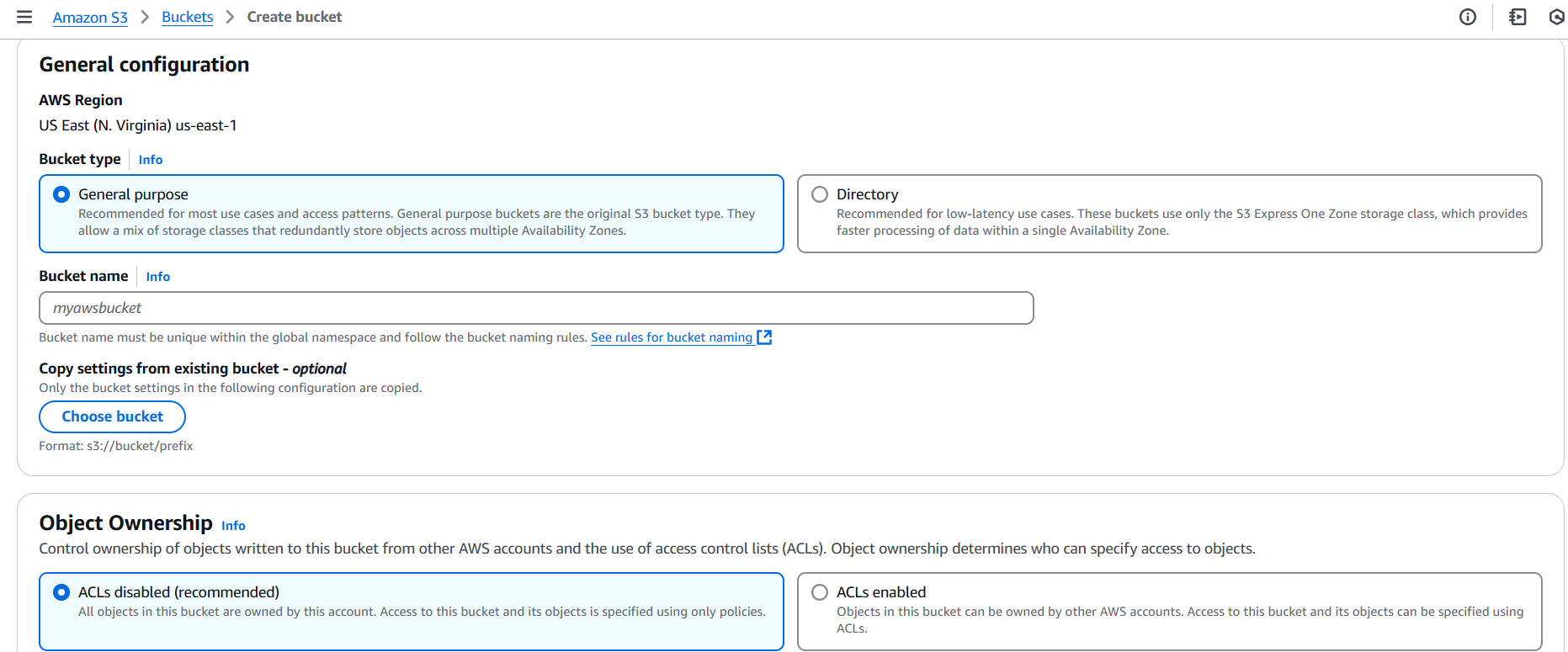


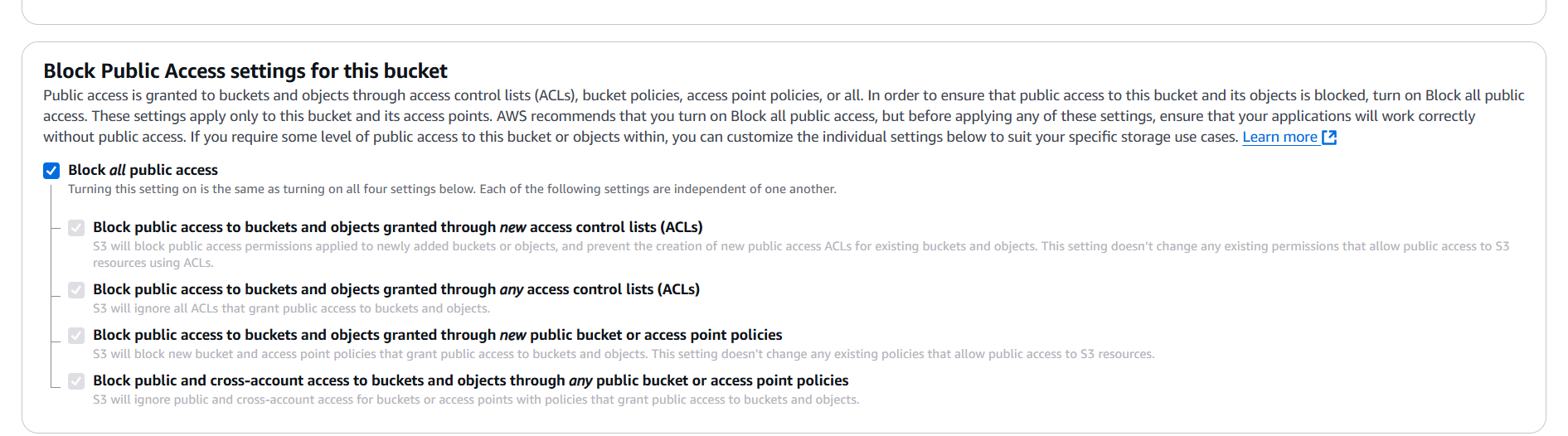
$ aws s3 ls

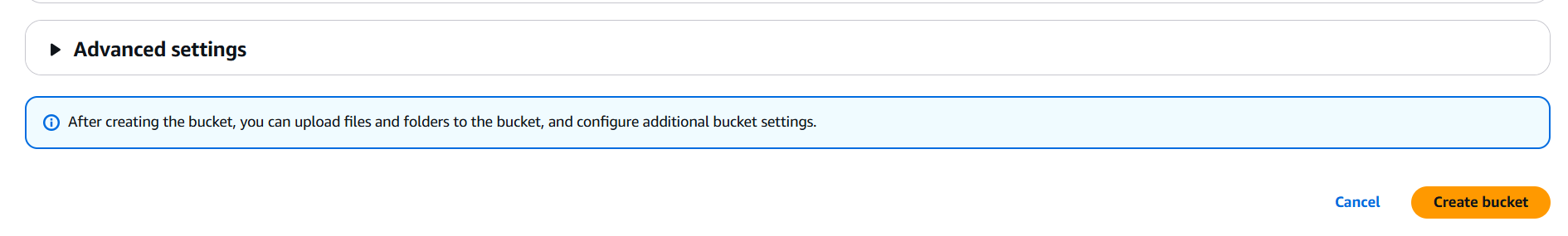


**STEP 3: Create S3 Bucket in AWS account as below,**

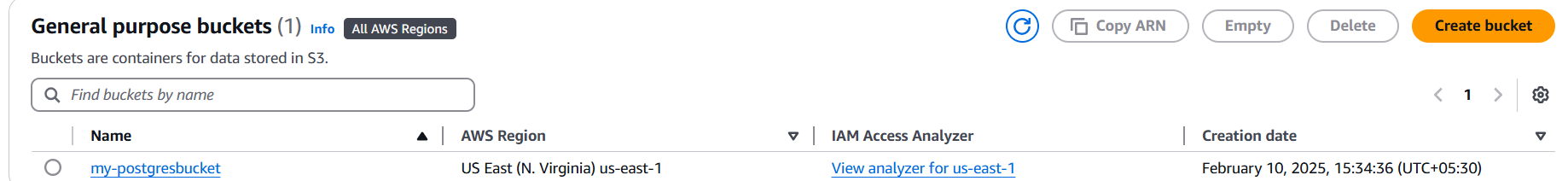
Bucket name must be unique. My bucket name is "**my-postgresbucket**." Additionally, I need to block public access by using the following checklist with the default settings.





Next, click on create bucket.  
  


Ckeck the available Bucket on AWS S3 service,



**STEP 4: Create a bash script as below**.

$ nano backup\_postgres.sh

#!/bin/bash

# Define AWS S3 bucket name

S3\_BUCKET="my-postgresbucket"

# Define PostgreSQL credentials

PG\_USER="postgres"

PG\_HOST="localhost"

BASE\_BACKUP\_DIR="/home/sysops/postgres\_backups"

# Export PostgreSQL password to avoid interactive prompt

export PGPASSWORD=$PG\_PASSWORD

# Create the main backup directory if it does not exist

mkdir -p $BASE\_BACKUP\_DIR

# Get the list of all databases

DB\_LIST=("postgres" "testdb")

# Loop through all databases and back them up

for DB in "${DB\_LIST[@]}"; do

TIMESTAMP=$(date +%Y-%m-%d\_%H-%M-%S)

DB\_BACKUP\_DIR="$BASE\_BACKUP\_DIR/$DB"

BACKUP\_FILE="$DB\_BACKUP\_DIR/${DB}\_${TIMESTAMP}.sql.gz"

# Create a separate directory for each database

mkdir -p $DB\_BACKUP\_DIR

echo "Backing up database $DB"

pg\_dump -U $PG\_USER -h $PG\_HOST -F c $DB | gzip > $BACKUP\_FILE

echo "Uploading $BACKUP\_FILE to S3"

aws s3 cp $BACKUP\_FILE s3://$S3\_BUCKET/$DB/

# Remove the local backup after upload

rm -f $BACKUP\_FILE

done

echo "All databases have been backed up and uploaded to S3 successfully"

Next, change the file permissions by granting execute access using the following command.

$ chmod +x backup\_postgres.sh

Before running the script, we need to store the PostgreSQL user password as a secret instead of placing it in the Bash script, as shown below.

$ nano ~/.pgpass

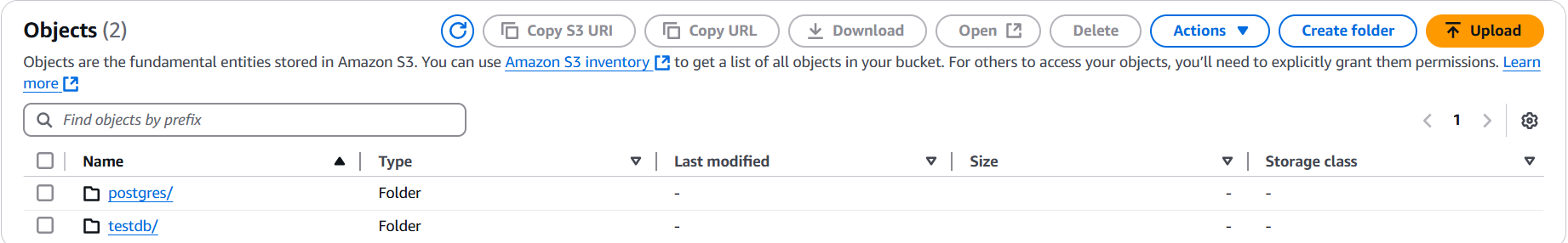
localhost:5432:\*:postgres:<postgres pass>

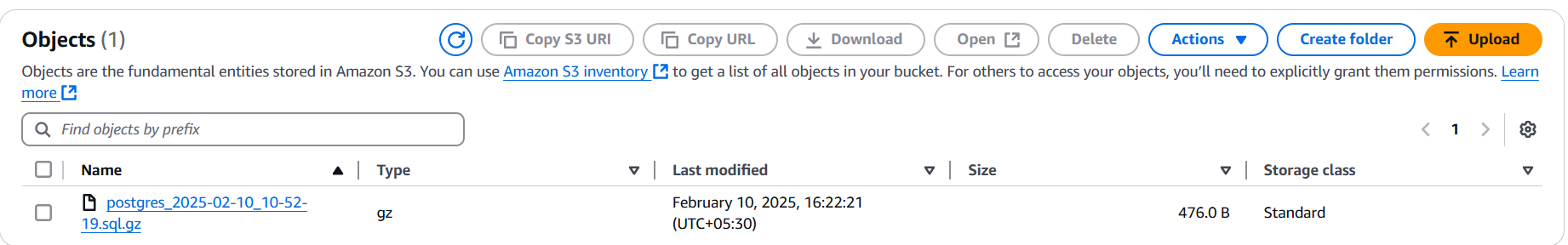
****

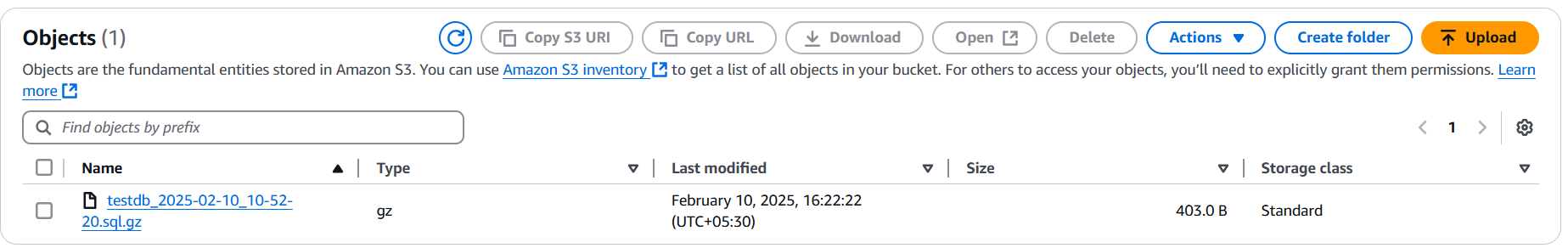
$ chmod 600 ~/.pgpass

$ bash backup\_postgres.sh

After successfully running the above script then backup will store on S3 Bucket in different directory as shown in below screenshot.







Also, check the backup directory on the server by using the following command.

$ cd /home/sysops/postgres\_backups

$ ls

