Mini Project for MGL Project

Setup Automated Daily Backup of EC2 Instance Data to Amazon S3 Using Cron Jobs.

Service used in these project – S3 ,EC2 Instance.

Cronjob- A cron job is a scheduled task in Unix-based operating systems that is automatically executed at specified intervals using the cron daemon (a background service). It is commonly used to automate repetitive tasks like backups, updates, or system maintenance.

```
* * * * * command-to-be-executed
- - - - -
| | | | |
| | | | +----- Day of the week (0 - 7) (Sunday is 0 or 7)
| | | +------- Month (1 - 12)
| | +--------- Day of the month (1 - 31)
| +----------- Hour (0 - 23)
+------------- Minute (0 - 59)
```

Step 1-

Create EC2 Instance and create folder and upload some data.

Step 2-

Create S3 Bucket in which your data will stored as backup.

Step 3-

Set Up IAM Role (If EC2 needs permission to access S3).

Step -4

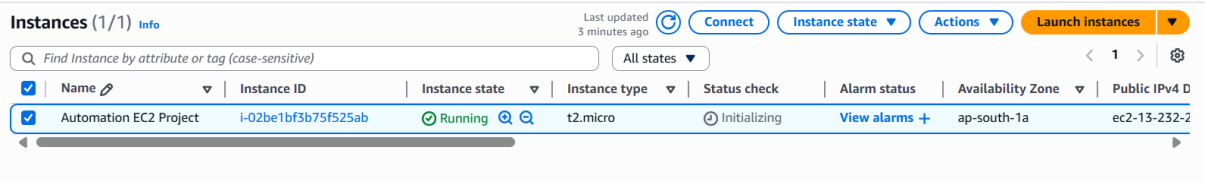
Prepare Backup Script on EC2.

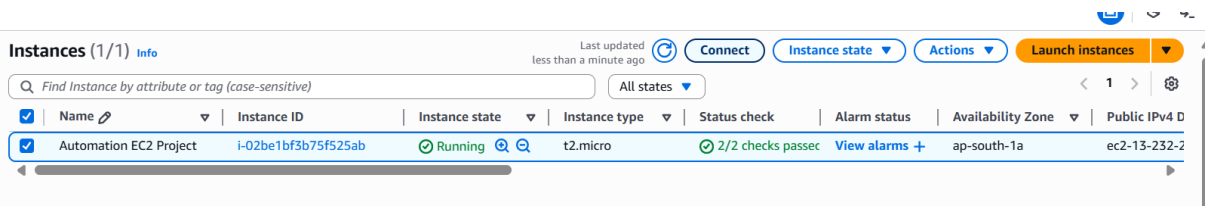
(Make sure AWS CLI is installed and configured on the EC2 instance, or the instance has an IAM role with S3 permissions.)

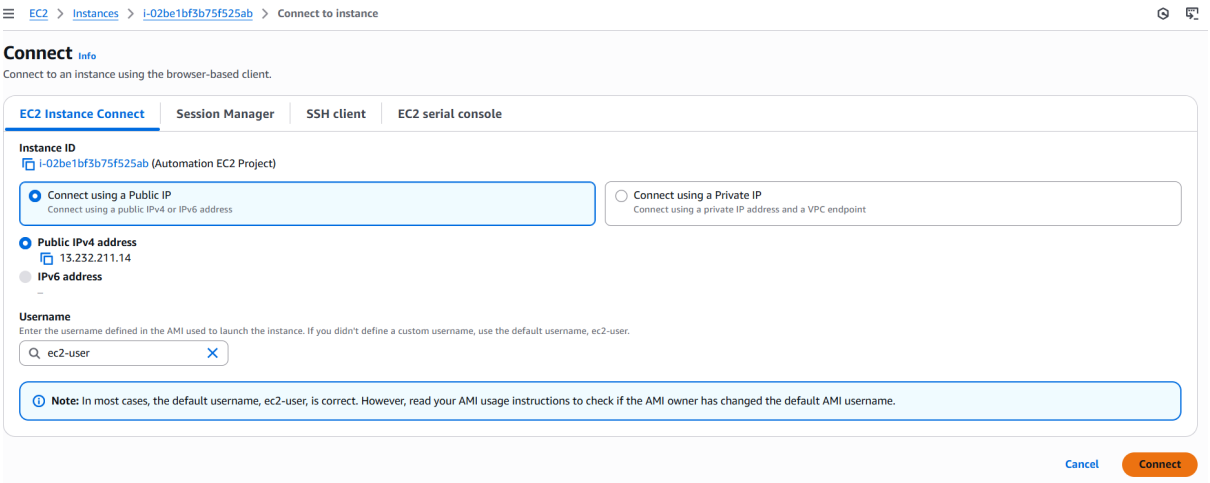
Step 5- Schedule Cron Job.

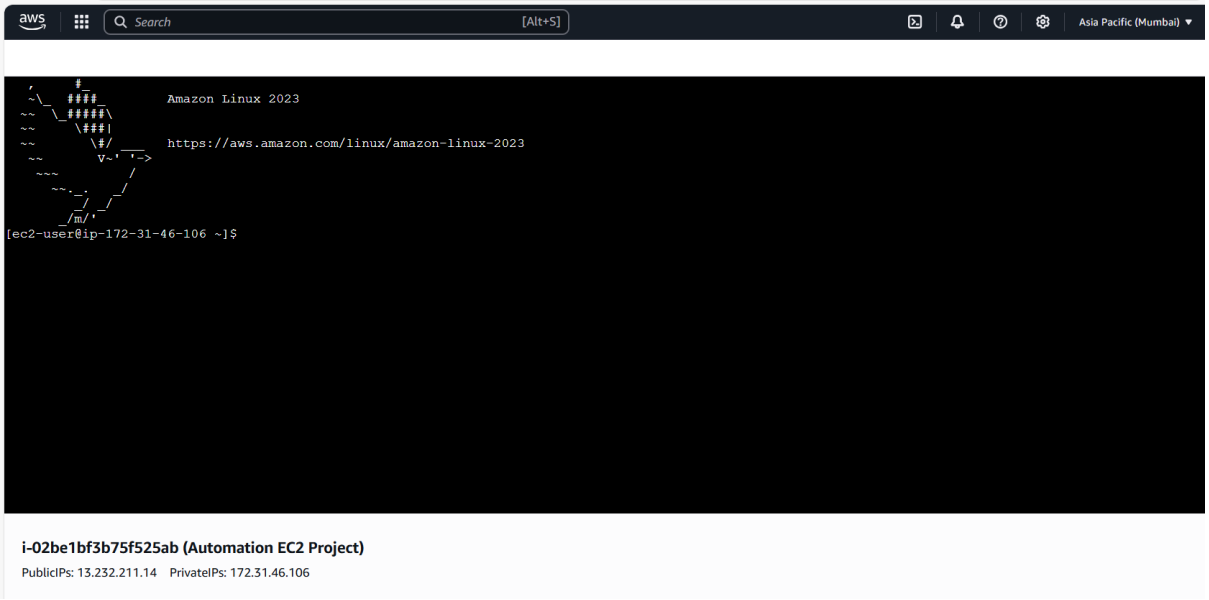
Step 1-

Create ec2 for your project and connect to instance.







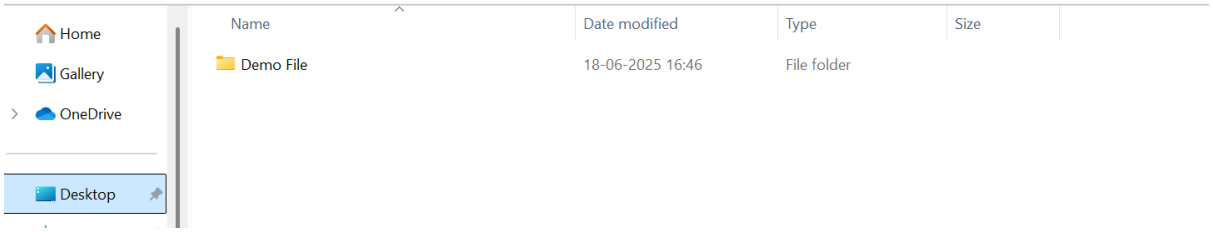


Now Creating a Directory name as Demo-Automation and upload data on it from your local machine folder.

```
[ec2-user@ip-172-31-46-106 /]$ ls
bin  boot  dev  etc  home  lib  lib64  local  media  mnt  opt  proc  root  run  sbin  srv  sys  tmp  usr  var
[ec2-user@ip-172-31-46-106 /]$
```

```
[ec2-user@ip-172-31-46-106 /]$ ls
Demo-Automation  bin  boot  dev  etc  home  lib  lib64  local  media  mnt  opt  pr
```

Suppose I want to transfer the content of these folder into my ec2 instance.



First I will copy these folder path and use SCP Cmd from my local machine.

Folder path- "C:\Users\ACC USER\Desktop\Demo File"

CMD-

scp -i "C:/Users/ACC USER/Downloads/LVM Demo.pem" -r "C:/Users/ACC USER/Desktop/Demo File" ec2-user@13.232.211.14: /Demo-Automation

What these cmd mean:-

Explanation of Each Part:

Part	Meaning
scp	Secure Copy – a command-line tool used to securely transfer files and directories between computers over SSH.
-i "C:\Users\ACC USER\Downloads\LVM Demo.pem"	Specifies the identity file (PEM key) used to authenticate with the EC2 instance. Required for SSH-based access.
-r	Recursive copy – tells scp to copy all files and subfolders inside the specified directory.
"C:\Users\ACC USER\Desktop\Demo File"	The local folder on your Windows machine that you want to upload. Quoted because the folder name has a space.
ec2-user@13.232.211.14	The username and public IP of your EC2 instance. For linux machine servers, the username is usually ec2-user.
/home/ec2-user/	The destination path on the EC2 instance where the folder will be uploaded. In this case, it goes into the ec2 user's home directory.

```
C:\Users\ACC USER>scp -i "C:/Users/ACC USER/Downloads/LVM Demo.pem" -r "C:/Users/ACC USER/Desktop/Demo File" ec2-user@13
.232.211.14:/home/ec2-user/
IA AVT Weekly Report 23-05-2025.docx                                    100%  280KB   2.5MB/s   00:00
IA UM - interactiveavenuesum Weekly Report 23-05-2025.docx              100%  101KB 715.0KB/s   00:00
IA-Voltas Weekly Report  23-05-2025.docx                                100%  294KB 883.6KB/s   00:00
IA-weekly-Utilization-17-05-2025 to 23-05-2025.xlsx                     100%   28KB   1.3MB/s   00:00
IBDIC Utilization Report  04-06-2025.xlsx                               100%   25KB   1.5MB/s   00:00
TASK incomplete.docx                                                    100%  394KB   1.5MB/s   00:00
Tata Cost Daily Report-  04-05-2025.xlsx                                100%  180KB 844.1KB/s   00:00
Zuno Daily Cost Report 30-April-2025.xlsx                               100%   16KB 622.9KB/s   00:00

C:\Users\ACC USER>
```

these is done from my local machine cmd

Cross verify it, check the content in your ec2 machine.

now here is issue that the content is copies in other folder we need to copy the content in our Demo-Automation FOLDER :-

To do these simply used cp command :

Cmd – cp -r “source path” “ destination path”

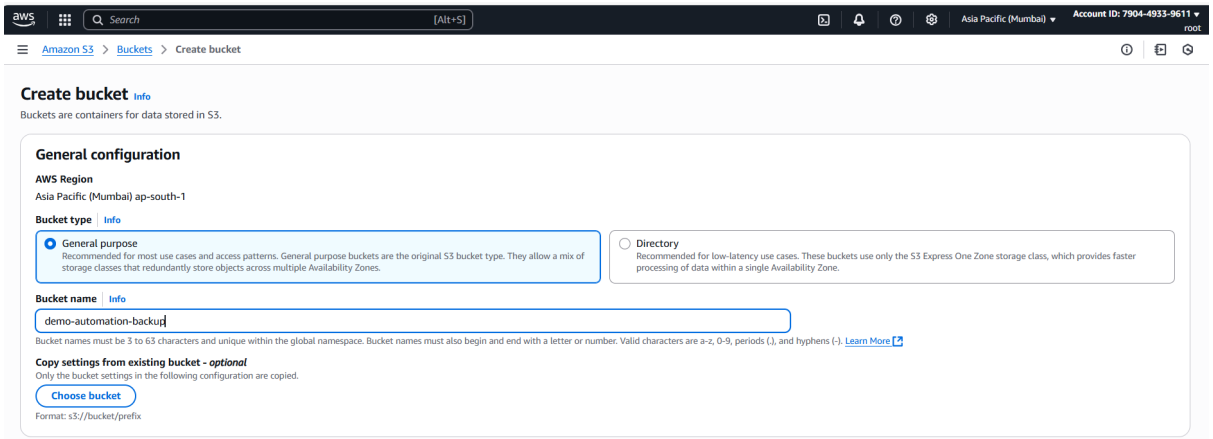
sudo cp -r ‘/home/ec2-user/Demo File’ /Demo-Automation

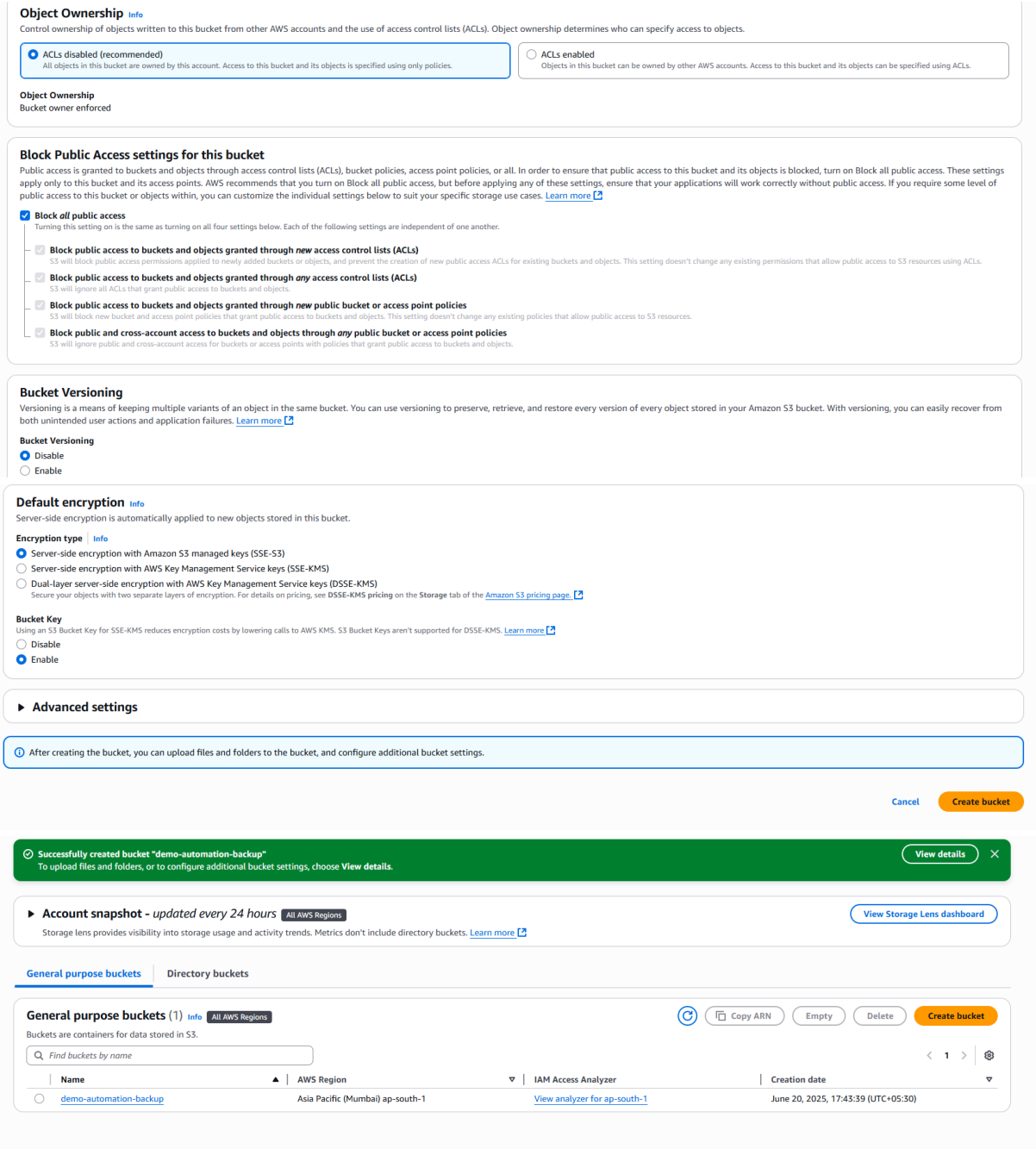
```
[ec2-user@ip-172-31-46-106 /]$ sudo cp -r '/home/ec2-user/Demo File' /Demo-Automation
[ec2-user@ip-172-31-46-106 /]$
```

till now we successfully copy and upload the content in our main ec2 instance server

Step 2-

Create S3 Bucket in which your data will stored as backup.



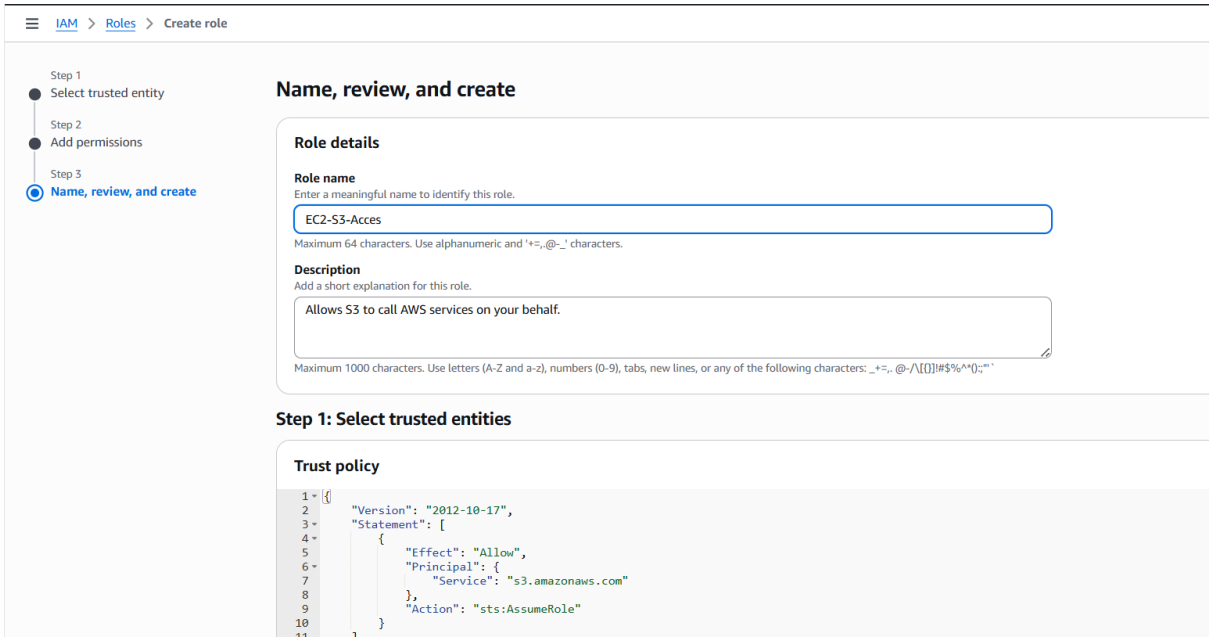


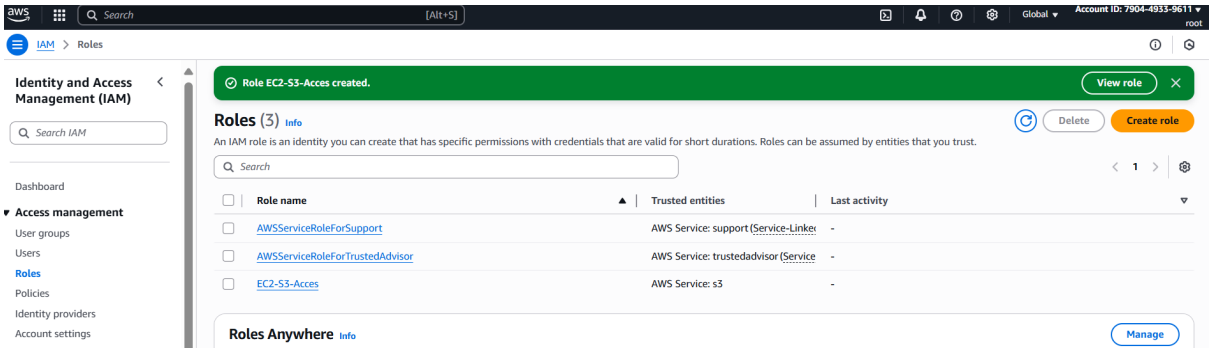
Step 3-

Set Up IAM Role (If EC2 needs permission to access S3).

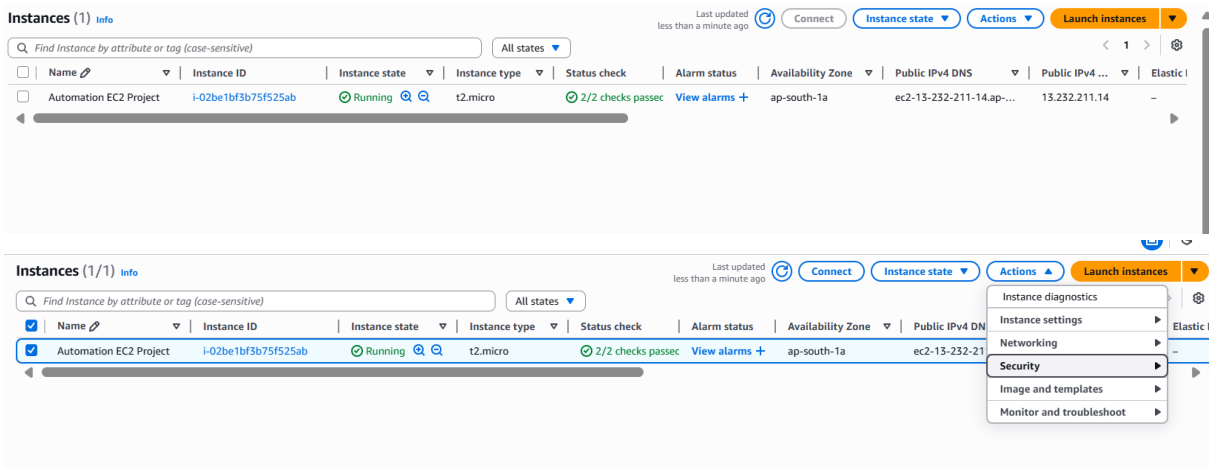
Select service – ec2

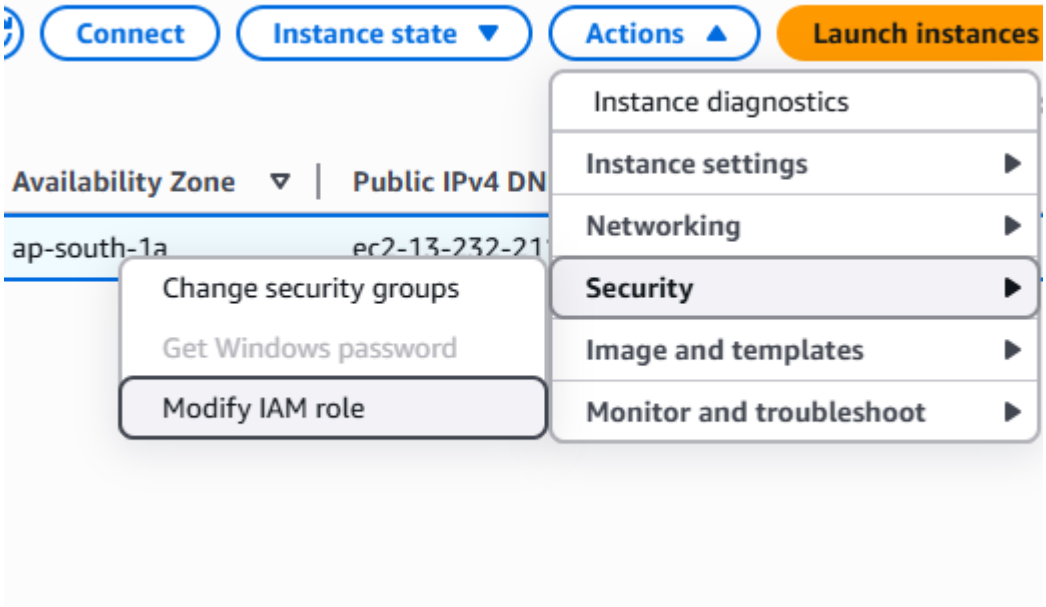
and assign full admin policy to these role and further attach to ec2 instance.





Now attach these role to ec2 instance





Step -4

1Prepare Backup Script on EC2.
(Make sure AWS CLI is installed and configured on the EC2 instance, or the instance has an IAM role with S3 permissions.)

1-Now for these we need to install aws cli,
cmd – sudo yum install awscli

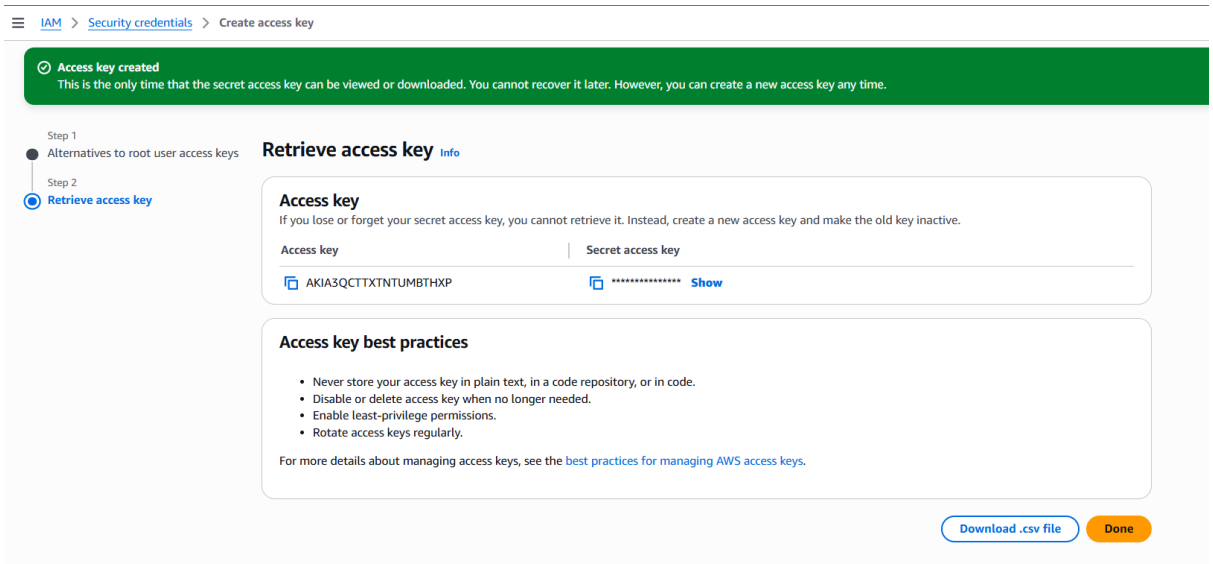
```
[ec2-user@ip-172-31-46-106 /]$ sudo yum install awscli
Amazon Linux 2023 Kernel Livepatch repository
Last metadata expiration check: 0:00:01 ago on Fri Jun 20 12:29:43 2025.
Package awscli-2-2.23.11-1.amzn2023.0.1.noarch is already installed.
Dependencies resolved.
Nothing to do.
Complete!
[ec2-user@ip-172-31-46-106 /]$
```

Now to verify used cmd – aws –version

```
[ec2-user@ip-172-31-46-106 /]$ aws --version
aws-cli/2.23.11 Python/3.9.22 Linux/6.1.140-154.222.amzn2023.x86_64 source/x86_64.amzn.2023
[ec2-user@ip-172-31-46-106 /]$
```

Do login with you access key and secret access key.

Create access and secret access key in aws



then used cmd – aws configure

```
[ec2-user@ip-172-31-46-106 /]$ aws configure
AWS Access Key ID [None]: AKIA3QCTTXTNTUMBTHXP
AWS Secret Access Key [None]:
```

```
AWS Access Key ID [None]: AKIA3QCTTXTNTUMBTHXP
AWS Secret Access Key [None]: 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
Default region name [None]:
Default output format [None]:
[ec2-user@ip-172-31-46-106 /]$
```

Step 2- Verify AWS CLI Access to S3 (via IAM Role)

cmd- aws s3 ls

```
[ec2-user@ip-172-31-46-106 /]$ aws s3 ls
2025-06-20 12:13:40 demo-automation-backup
[ec2-user@ip-172-31-46-106 /]$
```

3. Now create script file in ec2 home directory

cmd-nano /home/ec2-user/backup-to-s3.sh

```
[root@ip-172-31-46-106 /]# nano /home/ec2-user/backup-to-s3.sh
[root@ip-172-31-46-106 /]#
```

Source directory- “Demo-Automation/Demo File”
bucket name- "s3://demo-automation-backup"

Script -

```
#!/bin/bash

# Set variables

SOURCE_DIR="/Demo-Automation/Demo File"

S3_BUCKET="s3://demo-automation-backup"

TIMESTAMP=$(date +%F_%H-%M-%S)

BACKUP_NAME="backup_$TIMESTAMP.tar.gz"


# Compress the folder into /tmp directory

tar -czf /tmp/$BACKUP_NAME -C "$(dirname "$SOURCE_DIR")" "$(basename "$SOURCE_DIR")"


# Upload the archive to S3

aws s3 cp /tmp/$BACKUP_NAME $S3_BUCKET/


# Optionally, delete the local archive after upload

rm /tmp/$BACKUP_NAME
```

```
  GNU nano 8.3                                                    /home/
#!/bin/bash

# Set variables
SOURCE_DIR="/Demo-Automation/Demo File"
S3_BUCKET="s3://demo-automation-backup"
TIMESTAMP=$(date +%F_%H-%M-%S)
BACKUP_NAME="backup_$TIMESTAMP.tar.gz"

# Compress the folder into /tmp directory
tar -czf /tmp/$BACKUP_NAME -C "$(dirname "$SOURCE_DIR")" "$(basename "$SOURCE_DIR")"

# Upload the archive to S3
aws s3 cp /tmp/$BACKUP_NAME $S3_BUCKET/

# Optionally, delete the local archive after upload
rm /tmp/$BACKUP_NAME
```

Now last step run it manually but before give necessary permission

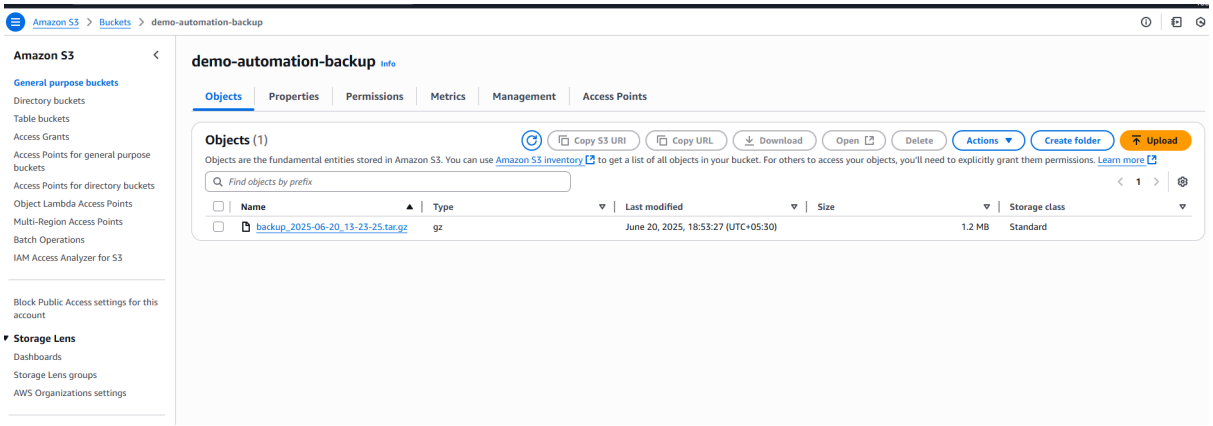
cmd- chmod +x /home/ec2-user/backup-to-s3.sh

last cmd- /home/ec2-user/backup-to-s3.sh

```
[root@ip-172-31-46-106 /]# chmod +x /home/ec2-user/backup-to-s3.sh
[root@ip-172-31-46-106 /]#
```

```
[root@ip-172-31-46-106 /]# /home/ec2-user/backup-to-s3.sh
upload: tmp/backup_2025-06-20_13-23-25.tar.gz to s3://demo-automation-backup/backup_2025-06-20_13-23-25.tar.gz
[root@ip-172-31-46-106 /]#
```

Finally our backup work successfully



NOW we need to set just cronjob here to schedule daily automation backup from our ec2 server to our S3 bucket.

FINAL STEP – Setup cronjob

Install cron (cronie)

Enable and start the cron service

Confirm it's running

To install cron system
use cmd- yum install cronie -y

```
[ec2-user@ip-172-31-46-106 ~]$ sudo yum install cronie
Amazon Linux 2023 repository
Dependencies resolved.
================================================================================================================================
 Package                                          Architecture                              Version
================================================================================================================================
Installing:
 cronie                                           x86_64                                    1.5.7-1.amzn2023.0.2
Installing dependencies:
 cronie-anacron                                   x86_64                                    1.5.7-1.amzn2023.0.2

Transaction Summary
================================================================================================================================
Install  2 Packages

Total download size: 147 k
Installed size: 341 k
Is this ok [y/N]:
```

```
[ec2-user@ip-172-31-46-106 ~]$ systemctl enable crond
Failed to enable unit: Access denied
[ec2-user@ip-172-31-46-106 ~]$ sudo systemctl enable crond
[ec2-user@ip-172-31-46-106 ~]$ sudo systemctl start crond
[ec2-user@ip-172-31-46-106 ~]$ sudo systemctl status crond
● crond.service - Command Scheduler
     Loaded: loaded (/usr/lib/systemd/system/crond.service; enabled; preset: enabled)
     Active: active (running) since Sat 2025-06-21 17:19:13 UTC; 17s ago
   Main PID: 129946 (crond)
      Tasks: 1 (limit: 1111)
     Memory: 988.0K
        CPU: 5ms
     CGroup: /system.slice/crond.service
             └─129946 /usr/sbin/crond -n

Jun 21 17:19:13 ip-172-31-46-106.ap-south-1.compute.internal crond[129946]: (CRON) STARTUP (1.5.7)
Jun 21 17:19:13 ip-172-31-46-106.ap-south-1.compute.internal systemd[1]: Started crond.service - Command Scheduler.
Jun 21 17:19:13 ip-172-31-46-106.ap-south-1.compute.internal crond[129946]: (CRON) INFO (Syslog will be used instead of sendmail.)
Jun 21 17:19:13 ip-172-31-46-106.ap-south-1.compute.internal crond[129946]: (CRON) INFO (RANDOM_DELAY will be scaled with factor 31% if used.)
Jun 21 17:19:13 ip-172-31-46-106.ap-south-1.compute.internal crond[129946]: (CRON) INFO (running with inotify support)
[ec2-user@ip-172-31-46-106 ~]$
```

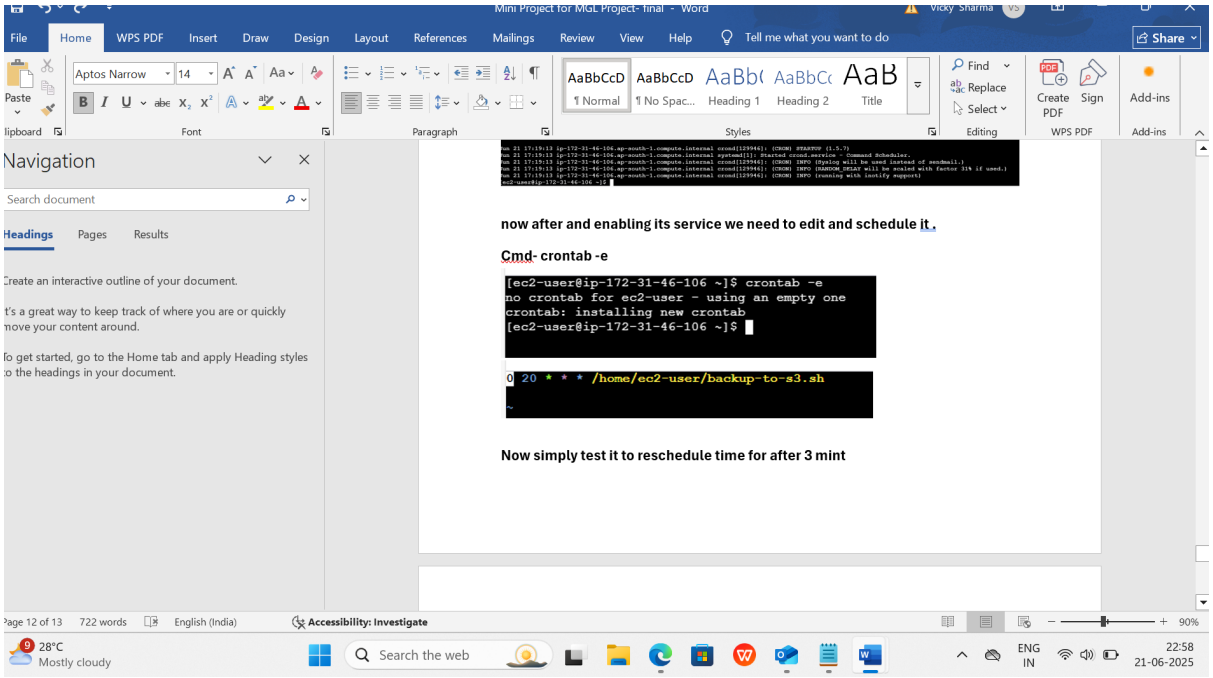
now after and enabling its service we need to edit and schedule it .

Cmd- crontab -e

```
[ec2-user@ip-172-31-46-106 ~]$ crontab -e
no crontab for ec2-user - using an empty one
crontab: installing new crontab
[ec2-user@ip-172-31-46-106 ~]$
```

```
0 20 * * * /home/ec2-user/backup-to-s3.sh

~
```

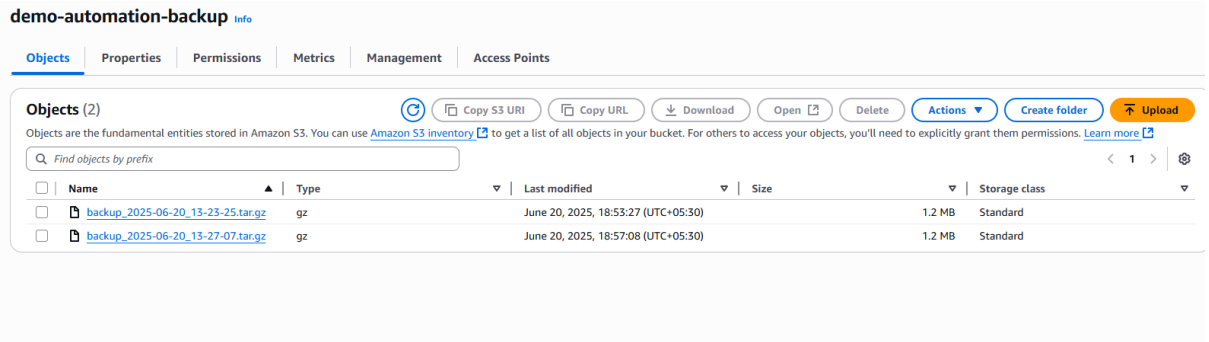
Now simply test it to reschedule time for after 8 mint



Now it is 22:58 schedule cronjob for test check at 11:05 pm

TROUBLE SHOOT PART -

Still there is no backup which indicate cronjob didn't work so we need to update script file in order to work cronjob properly



Previous script
#!/bin/bash

Set variables

SOURCE_DIR="/Demo-Automation/Demo File"

S3_BUCKET="s3://demo-automation-backup"

TIMESTAMP=$(date +%F_%H-%M-%S)

BACKUP_NAME="backup_$TIMESTAMP.tar.gz"

Compress the folder into /tmp directory

tar -czf /tmp/$BACKUP_NAME -C "$(dirname "$SOURCE_DIR")" "$(basename "$SOURCE_DIR")"

Upload the archive to S3

aws s3 cp /tmp/$BACKUP_NAME $S3_BUCKET/

Optionally, delete the local archive after upload

rm /tmp/$BACKUP_NAME This script creates a timestamped compressed .tar.gz backup of the folder and uploads it to S3.

NEW UPDATE SCRIPT FOR CRONJOB

```
#!/bin/bash

# Log output
exec >> /home/ec2-user/cron-script.log 2>&1
echo "🟡Backup started at $(date)"

SRC="/home/ec2-user/Demo File"
BUCKET="s3://demo-automation-backup"
TS=$(date +%F_%H-%M-%S)
ARCHIVE="/tmp/backup_$TS.tar.gz"

/usr/bin/tar -czf "$ARCHIVE" -C "$(dirname "$SRC")" "$(basename "$SRC")"
/usr/bin/aws s3 cp "$ARCHIVE" "$BUCKET" && rm -f "$ARCHIVE"

echo "✅Backup completed at $(date)"
```

New update script

```
#!/bin/bash

# Log output
exec >> /home/ec2-user/cron-script.log 2>&1
echo "⏳ Backup started at $(date)"

SRC="/home/ec2-user/Demo File"
BUCKET="s3://demo-automation-backup"
TS=$(date +%F_%H-%M-%S)
ARCHIVE="/tmp/backup_$TS.tar.gz"

/usr/bin/tar -czf "$ARCHIVE" -C "$(dirname "$SRC")" "$(basename "$SRC")"
/usr/bin/aws s3 cp "$ARCHIVE" "$BUCKET" && rm -f "$ARCHIVE"

echo "✅ Backup completed at $(date)"
```

Why It Works Manually but Not in Cron

When you run the script manually, it inherits:

- Your full environment ($PATH, $HOME, etc.)
- Your user's permissions and session
- Your shell (with known variables, access to aws, etc.)

Changes we made in these scripts

Use Full Path for aws (Cron does not inherit your shell PATH)

Problem:

In cron, the environment is limited, so just writing aws might fail.

Add Logging to See What Cron Sees

At the top of your script, add this to capture the output and errors for debugging:

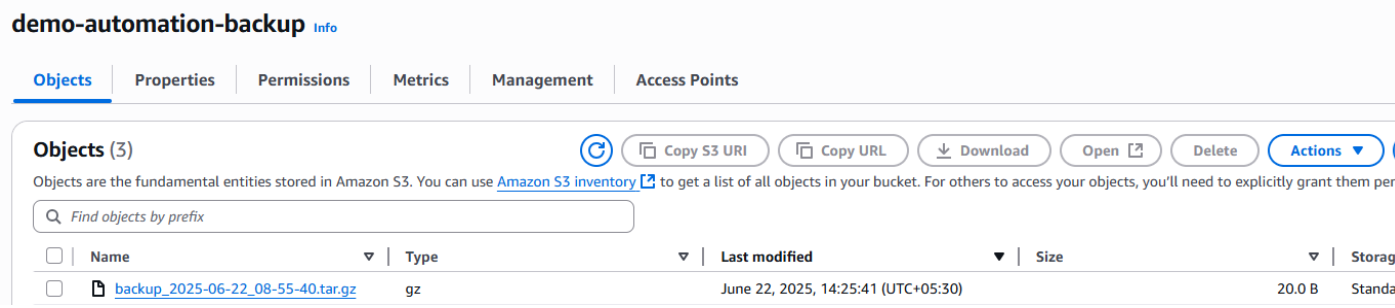
Now manually test the script

cmd-sudo chmod +x /home/ec2-user/backup-to-s3.sh

cmd-sudo /home/ec2-user/backup-to-s3.sh

```
[ec2-user@ip-172-31-46-106 ~]$ sudo  chmod +x /home/ec2-user/backup-to-s3.sh
[ec2-user@ip-172-31-46-106 ~]$ sudo  /home/ec2-user/backup-to-s3.sh
[ec2-user@ip-172-31-46-106 ~]$
```

Perfect! Script is running properly



Now scheduling cron once again

```
[ec2-user@ip-172-31-46-106 ~]$ crontab -l
5 23 * * * /home/ec2-user/backup-to-s3.sh


[ec2-user@ip-172-31-46-106 ~]$
```

now add one more cronjob which scheduling after 5 mint

```
[ec2-user@ip-172-31-46-106 ~]$ crontab -l
55 14 * * * /home/ec2-user/backup-to-s3.sh

[ec2-user@ip-172-31-46-106 ~]$
```

using tail command to see output

```
[ec2-user@ip-172-31-46-106 ~]$ crontab -e
crontab: installing new crontab
[ec2-user@ip-172-31-46-106 ~]$ crontab -l
55 14 * * * /home/ec2-user/backup-to-s3.sh

[ec2-user@ip-172-31-46-106 ~]$ tail -f /home/ec2-user/cron-script.log
upload: ../../tmp/backup_2025-06-22_08-55-40.tar.gz to s3://demo-automation-backup/backup_2025-06-22_08-55-40.tar.gz
Backup completed at Sun Jun 22 08:55:40 UTC 2025
Backup started at Sun Jun 22 09:09:18 UTC 2025
tar: /home/ec2-user/Demo-Automation: Cannot open: No such file or directory
tar: Error is not recoverable: exiting now
upload: ../../tmp/backup_2025-06-22_09-09-18.tar.gz to s3://demo-automation-backup/backup_2025-06-22_09-09-18.tar.gz
Backup completed at Sun Jun 22 09:09:18 UTC 2025
Backup started at Sun Jun 22 09:21:15 UTC 2025
upload: ../../tmp/backup_2025-06-22_09-21-15.tar.gz to s3://demo-automation-backup/backup_2025-06-22_09-21-15.tar.gz
✅Backup completed at Sun Jun 22 09:21:16 UTC 2025
```

our cronjob is working perfectly but there is small issue the time of our system is set in utc due to which it run in utc time simply we need to edit cronjob and set time according to utc.

Still cronjob doesn't work

when I try to troubleshoot these issue after an 30 mint I find major issue is in permission

Problem Identified:

```
[ec2-user@ip-172-31-46-106 ~]$ ls -l /home/ec2-user/backup-to-s3.sh
-rwxr-xr-x. 1 root root 670 Jun 22 16:00 /home/ec2-user/backup-to-s3.sh
[ec2-user@ip-172-31-46-106 ~]$
```

Your script is owned by root, but your cron job is scheduled under ec2-user.

-rwxr-xr-x. 1 root root 670 Jun 22 16:00 /home/ec2-user/backup-to-s3.sh

This means the ec2-user does not have permission to properly access or execute this script via cron.

Solution-

Run this to change ownership:

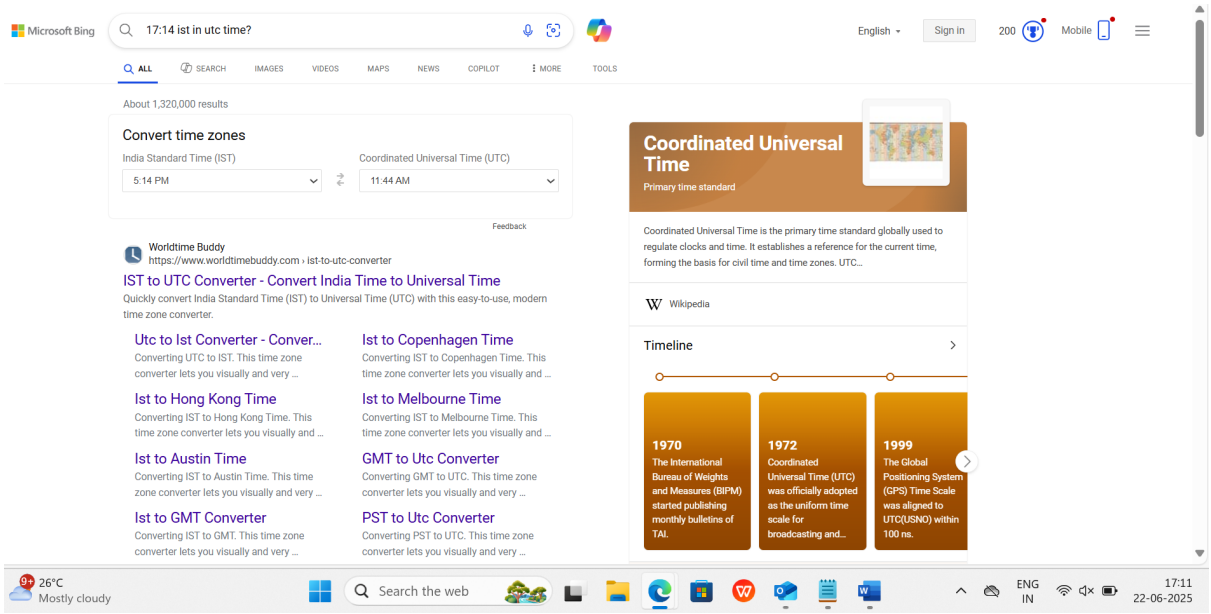
```
[ec2-user@ip-172-31-46-106 ~]$ sudo chown ec2-user:ec2-user /home/ec2-user/backup-to-s3.sh
[ec2-user@ip-172-31-46-106 ~]$ ls -l /home/ec2-user/backup-to-s3.sh
-rwxr-xr-x. 1 ec2-user ec2-user 670 Jun 22 16:00 /home/ec2-user/backup-to-s3.sh
[ec2-user@ip-172-31-46-106 ~]$
```

Now trying again to run cronjob

```
[ec2-user@ip-172-31-46-106 ~]$ crontab -l
MAILTO=""
SHELL=/bin/bash
PATH=/usr/local/bin:/usr/bin:/bin:/usr/sbin:/sbin

36 11 * * * /home/ec2-user/backup-to-s3.sh

[ec2-user@ip-172-31-46-106 ~]$
```

Adding shell and path at top of cronjob to work properly

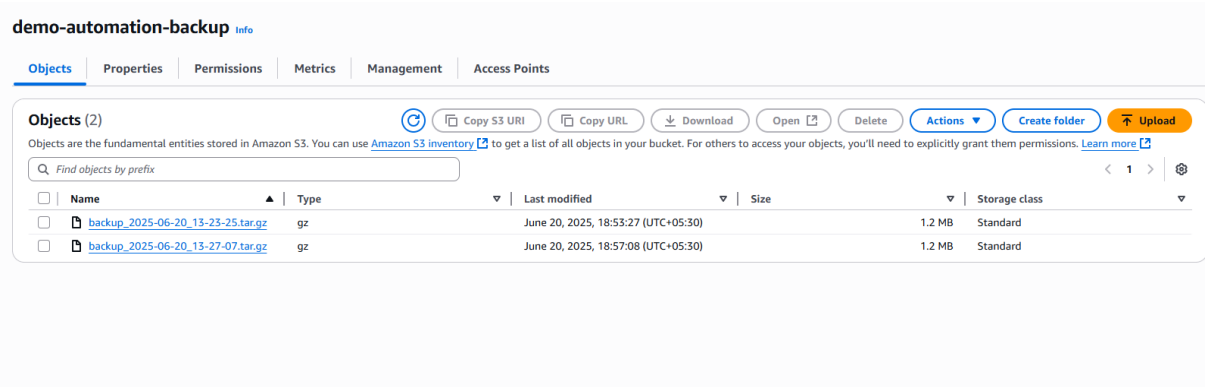
Setting time according to utc




```
SHELL=/bin/bash
PATH=/usr/local/bin:/usr/bin:/bin:/usr/sbin:/sbin

44 11 * * * /home/ec2-user/backup-to-s3.sh

~
~
~
```

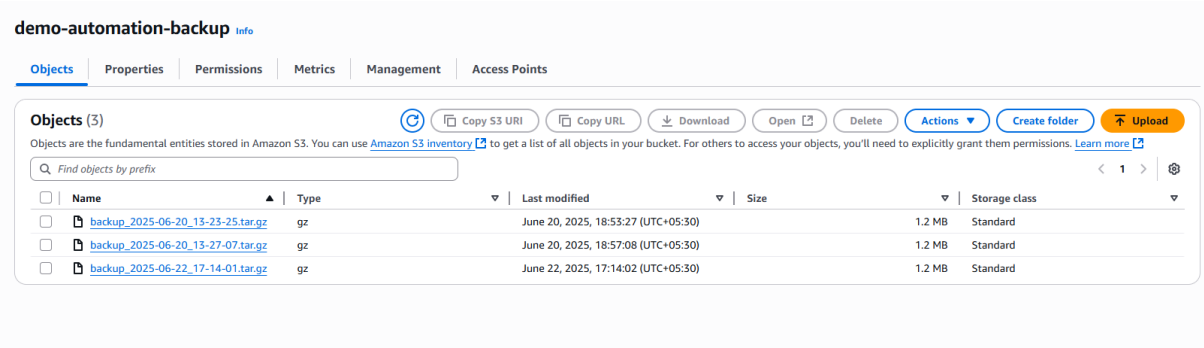
```
[ec2-user@ip-172-31-46-106 ~]$ crontab -l
SHELL=/bin/bash
PATH=/usr/local/bin:/usr/bin:/bin:/usr/sbin:/sbin

44 11 * * * /home/ec2-user/backup-to-s3.sh

[ec2-user@ip-172-31-46-106 ~]$
```

Before 17:14



After 17:14

Backup Successfully using cronjob



Finally Mini Project completed successfully after a lot of troubleshooting.