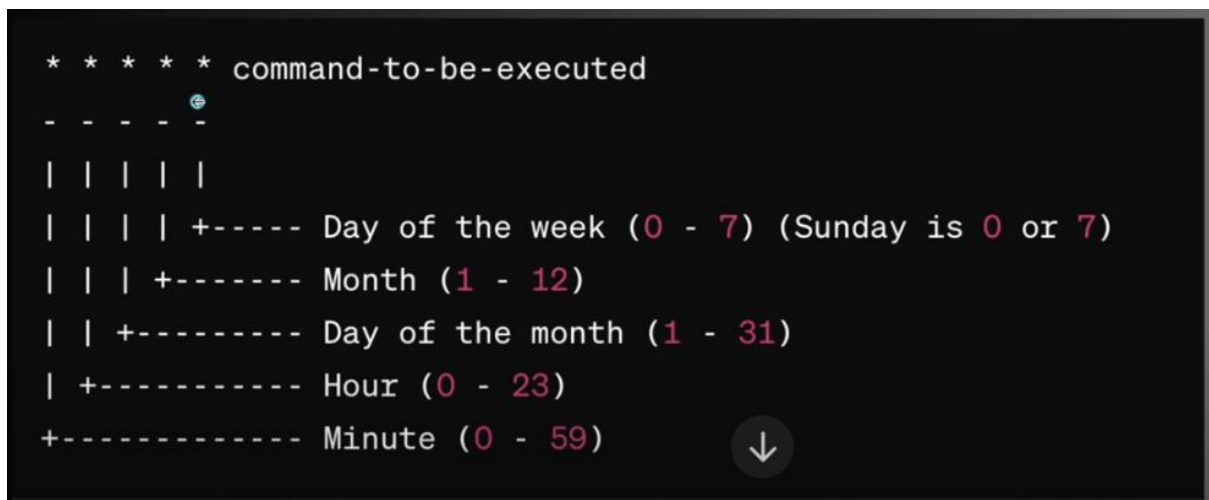


Mini Project for MGL Project

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.



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.

The screenshot displays the AWS Management Console interface for managing EC2 instances. The top section shows a list of instances with columns for Name, Instance ID, Instance state, Instance type, Status check, Alarm status, Availability Zone, and Public IPv4 D. The instance 'Automation EC2 Project' (ID: i-02be1bf3b75f525ab) is shown in a 'Running' state. Below the list, the 'Connect' page is visible, offering options to connect to the instance using a Public IP or a Private IP. The 'Public IPv4 address' is selected, showing the address 13.232.211.14. The 'Username' field is set to 'ec2-user'. A note at the bottom states: 'Note: In most cases, the default username, ec2-user, is correct. However, read your AMI usage instructions to check if the AMI owner has changed the default AMI username.'

Instances (1/1) Info Last updated 3 minutes ago [Connect](#) [Instance state](#) [Actions](#) [Launch instances](#)

Find Instance by attribute or tag (case-sensitive) All states

Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public IPv4 D
Automation EC2 Project	i-02be1bf3b75f525ab	Running	t2.micro	Initializing	View alarms +	ap-south-1a	ec2-13-232-2

Connect info Connect to an instance using the browser-based client.

EC2 Instance Connect Session Manager SSH client EC2 serial console

Instance ID i-02be1bf3b75f525ab (Automation EC2 Project)

☒ Connect using a Public IP Connect using a public IPv4 or IPv6 address

☐ Connect using a Private IP Connect using a private IP address and a VPC endpoint

☒ Public IPv4 address 13.232.211.14

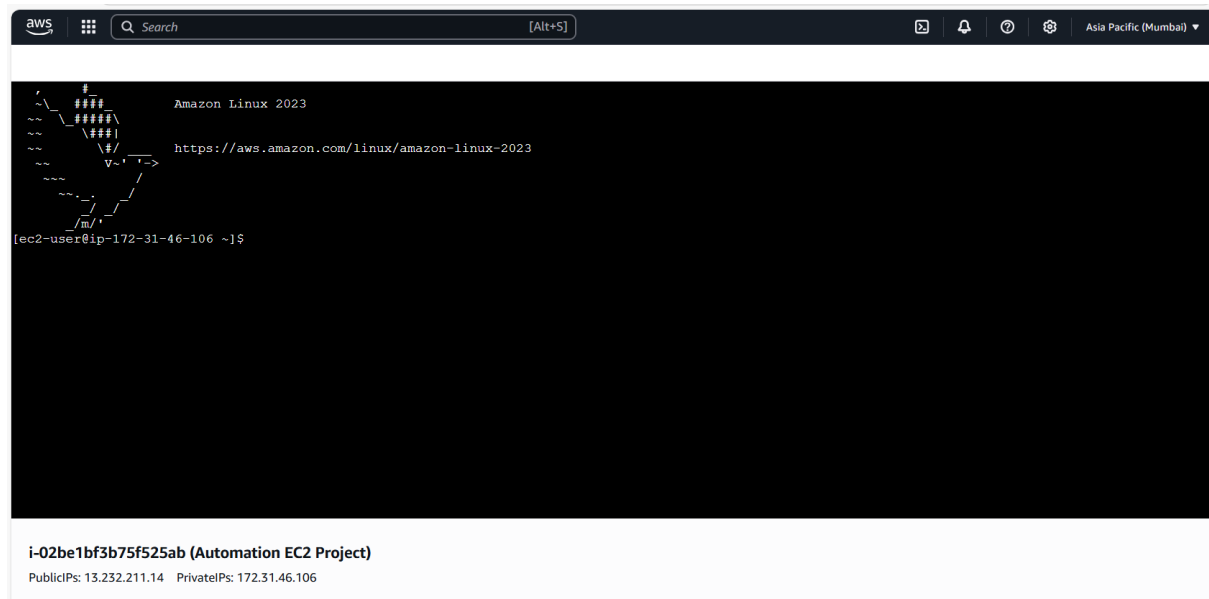
☐ IPv6 address -

Username Enter the username defined in the AMI used to launch the instance. If you didn't define a custom username, use the default username, ec2-user.

ec2-user

Note: In most cases, the default username, ec2-user, is correct. However, read your AMI usage instructions to check if the AMI owner has changed the default AMI username.

[Cancel](#) [Connect](#)

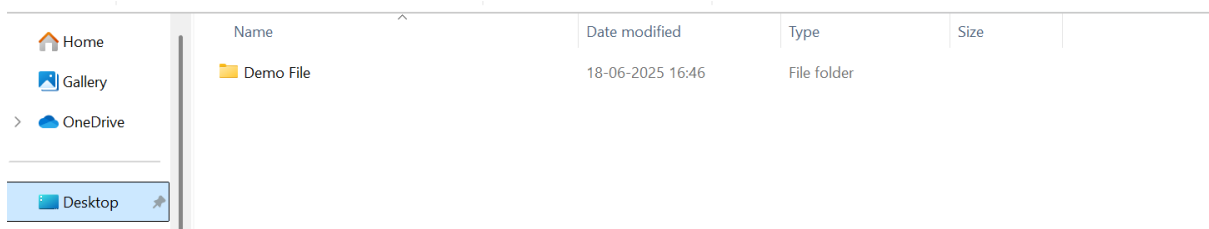


Now Creating a Directory name as Demo-Automation and upload data on it from your local machine folder.

```
[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.

Create bucket [Info](#)

Buckets are containers for data stored in S3.

General configuration

AWS Region
Asia Pacific (Mumbai) ap-south-1

Bucket type [Info](#)

☒ **General purpose**
Recommended for most use cases and access patterns. General purpose buckets are the original S3 bucket type. They allow a mix of storage classes that redundantly store objects across multiple Availability Zones.

☐ **Directory**
Recommended for low-latency use cases. These buckets use only the S3 Express One Zone storage class, which provides faster processing of data within a single Availability Zone.

Bucket name [Info](#)

demo-automation-backup

Bucket names must be 3 to 63 characters and unique within the global namespace. Bucket names must also begin and end with a letter or number. Valid characters are a-z, 0-9, periods (.), and hyphens (-). [Learn More](#)

Copy settings from existing bucket - optional
Only the bucket settings in the following configuration are copied.

[Choose bucket](#)

Format: s3://bucket/prefix

Object Ownership info

Control ownership of objects written to this bucket from other AWS accounts and the use of access control lists (ACLs). Object ownership determines who can specify access to objects.

☒ ACLs disabled (recommended)
 All objects in this bucket are owned by this account. Access to this bucket and its objects is specified using only policies.

☐ ACLs enabled
 Objects in this bucket can be owned by other AWS accounts. Access to this bucket and its objects can be specified using ACLs.

Object Ownership

Bucket owner enforced

Block Public Access settings for this bucket

Public access is granted to buckets and objects through access control lists (ACLs), bucket policies, access point policies, or all. In order to ensure that public access to this bucket and its objects is blocked, turn on Block all public access. These settings apply only to this bucket and its access points. AWS recommends that you turn on Block all public access, but before applying any of these settings, ensure that your applications will work correctly without public access. If you require some level of public access to this bucket or objects within, you can customize the individual settings below to suit your specific storage use cases. [Learn more](#)

☒ Block all public access
 Turning this setting on is the same as turning on all four settings below. Each of the following settings are independent of one another.

☒ Block public access to buckets and objects granted through new access control lists (ACLs)
 S3 will block public access permissions applied to newly added buckets or objects, and prevent the creation of new public access ACLs for existing buckets and objects. This setting doesn't change any existing permissions that allow public access to S3 resources using ACLs.

☒ Block public access to buckets and objects granted through any access control lists (ACLs)
 S3 will ignore all ACLs that grant public access to buckets and objects.

☒ Block public access to buckets and objects granted through new public bucket or access point policies
 S3 will block new bucket and access point policies that grant public access to buckets and objects. This setting doesn't change any existing policies that allow public access to S3 resources.

☒ Block public and cross-account access to buckets and objects through any public bucket or access point policies
 S3 will ignore public and cross-account access for buckets or access points with policies that grant public access to buckets and objects.

Bucket Versioning

Versioning is a means of keeping multiple variants of an object in the same bucket. You can use versioning to preserve, retrieve, and restore every version of every object stored in your Amazon S3 bucket. With versioning, you can easily recover from both unintended user actions and application failures. [Learn more](#)

Bucket Versioning

☒ Disable
☐ Enable

Default encryption info

Server-side encryption is automatically applied to new objects stored in this bucket.

Encryption type info

☒ Server-side encryption with Amazon S3 managed keys (SSE-S3)
☐ Server-side encryption with AWS Key Management Service keys (SSE-KMS)
☐ Dual-layer server-side encryption with AWS Key Management Service keys (DSSE-KMS)
 Secure your objects with two separate layers of encryption. For details on pricing, see DSSE-KMS pricing on the Storage tab of the [Amazon S3 pricing page](#).

Bucket Key

Using an S3 Bucket Key for SSE-KMS reduces encryption costs by lowering calls to AWS KMS. S3 Bucket Keys aren't supported for DSSE-KMS. [Learn more](#)

☐ Disable
☒ Enable

Advanced settings

After creating the bucket, you can upload files and folders to the bucket, and configure additional bucket settings.

Cancel

Create bucket

Successfully created bucket "demo-automation-backup"

View details

To upload files and folders, or to configure additional bucket settings, choose [View details](#).

Account snapshot - updated every 24 hours

All AWS Regions

View Storage Lens dashboard

Storage lens provides visibility into storage usage and activity trends. Metrics don't include directory buckets. [Learn more](#)

General purpose buckets

Directory buckets

General purpose buckets (1) info

All AWS Regions

Buckets are containers for data stored in S3.

Find buckets by name

Name

▲

▼

AWS Region

▼

▲

IAM Access Analyzer

▼

▲

Creation date

▼

▲

☐ demo-automation-backup
 Asia Pacific (Mumbai) ap-south-1
 [View analyzer for ap-south-1](#)
June 20, 2025, 17:43:39 (UTC+05:30)

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.

Step 1

Step 2

Step 3

Select trusted entity

Add permissions

Name, review, and create

Name, review, and create

Role details

Role name
Enter a meaningful name to identify this role.

Maximum 64 characters. Use alphanumeric and '+=, @-.' characters.

Description
Add a short explanation for this role.

Maximum 1000 characters. Use letters (A-Z and a-z), numbers (0-9), tabs, new lines, or any of the following characters: _+=, @-./[]!#\$%^&*~''

Step 1: Select trusted entities

Trust policy

```
1 {
2   "Version": "2012-10-17",
3   "Statement": [
4     {
5       "Effect": "Allow",
6       "Principal": {
7         "Service": "s3.amazonaws.com"
8       },
9       "Action": "sts:AssumeRole"
10    }
11  ]
12 }
```

aws

Search

[Alt+S]

Global

Account ID: 7904-4933-9611

root

Identity and Access Management (IAM)

Search IAM

Dashboard

Access management

User groups

Users

Roles

Policies

Identity providers

Account settings

Role EC2-S3-Acces created.

View role

Delete

Create role

Roles (3)

An IAM role is an identity you can create that has specific permissions with credentials that are valid for short durations. Roles can be assumed by entities that you trust.

Search

< 1 >

<input type="checkbox"/>	Role name	Trusted entities	Last activity
<input type="checkbox"/>	AWSServiceRoleForSupport	AWS Service: support (Service-Linker)	-
<input type="checkbox"/>	AWSServiceRoleForTrustedAdvisor	AWS Service: trustedadvisor (Service)	-
<input type="checkbox"/>	EC2-S3-Acces	AWS Service: s3	-

Roles Anywhere

Manage

Now attach these role to ec2 instance

Instances (1)

Info

Last updated less than a minute ago

Connect

Instance state

Actions

Launch instances

Find Instance by attribute or tag (case-sensitive)

All states

<input type="checkbox"/>	Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public IPv4 DNS	Public IPv4 ...	Elastic I
<input type="checkbox"/>	Automation EC2 Project	i-02be1bf3b75f525ab	Running	t2.micro	2/2 checks passed	View alarms	ap-south-1a	ec2-13-232-211-14.ap-...	13.232.211.14	-

Instances (1/1)

Info

Last updated less than a minute ago

Connect

Instance state

Actions

Launch instances

Find Instance by attribute or tag (case-sensitive)

All states

<input checked="" type="checkbox"/>	Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public IPv4 DNS	Public IPv4 ...	Elastic I
<input checked="" type="checkbox"/>	Automation EC2 Project	i-02be1bf3b75f525ab	Running	t2.micro	2/2 checks passed	View alarms	ap-south-1a	ec2-13-232-211-14.ap-...	13.232.211.14	-

Instance diagnostics

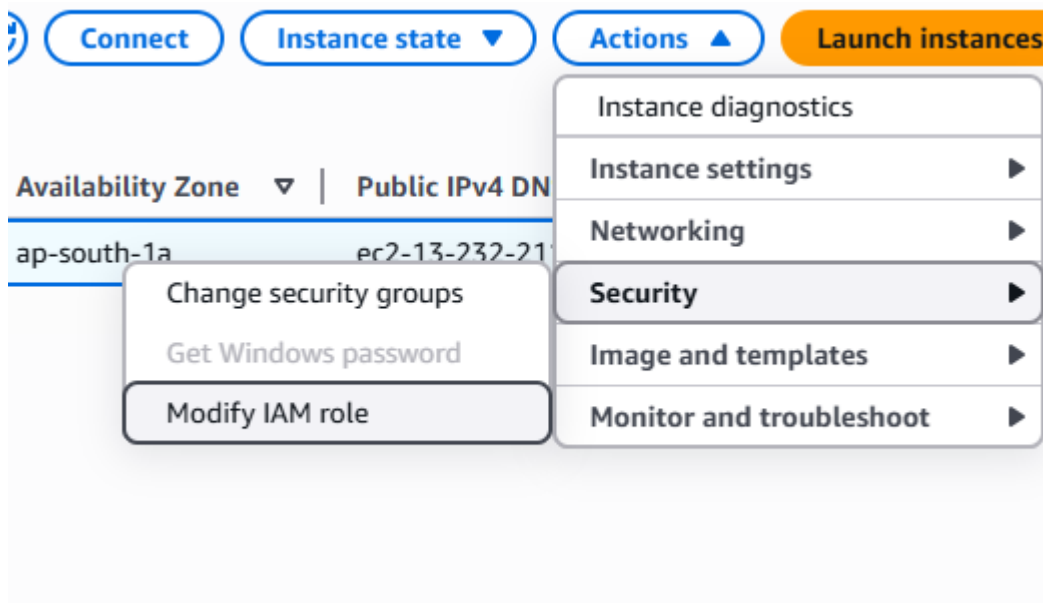
Instance settings

Networking

Security

Image and templates

Monitor and troubleshoot



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

☰ IAM > Security credentials > Create access key

🔔 Access key created

This is the only time that the secret access key can be viewed or downloaded. You cannot recover it later. However, you can create a new access key any time.

Step 1

● Alternatives to root user access keys

Step 2

● **Retrieve access key**

Retrieve access key [Info](#)

Access key

If you lose or forget your secret access key, you cannot retrieve it. Instead, create a new access key and make the old key inactive.

Access key	Secret access key
AKIA3QCTTXTNTUMBTHXP	***** Show

Access key best practices

- Never store your access key in plain text, in a code repository, or in code.
- Disable or delete access key when no longer needed.
- Enable least-privilege permissions.
- Rotate access keys regularly.

For more details about managing access keys, see the [best practices for managing AWS access keys](#).

[Download .csv file](#) [Done](#)

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

The screenshot shows the Amazon S3 console interface. The breadcrumb navigation at the top indicates the path: Amazon S3 > Buckets > demo-automation-backup. The left-hand navigation pane is expanded to show 'demo-automation-backup' under the 'Buckets' section. The main content area displays the 'demo-automation-backup' bucket details, with the 'Objects' tab selected. It shows a list of objects with one entry: 'backup_2025-06-20_13-23-25.tar.gz' of type 'gz', last modified on June 20, 2025, at 18:53:27 (UTC+05:30), with a size of 1.2 MB and stored in the 'Standard' storage class. The interface includes various action buttons like 'Copy S3 URI', 'Download', 'Delete', 'Actions', 'Create folder', and 'Upload'.

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]: y

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

~

[illegible]

TROUBLE SHOOT PART -

demo-automation-backup

Info

Objects

Properties

Permissions

Metrics

Management

Access Points

Objects (2)

Copy S3 URI

Copy URL

Download

Open

Delete



Actions

Create folder

Upload

Find objects by prefix

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](#)

<input type="checkbox"/>	Name	Type	Last modified	Size	Storage class
<input type="checkbox"/>	 backup_2025-06-20_13-23-25.tar.gz	gz	June 20, 2025, 18:53:27 (UTC+05:30)		1.2 MB Standard
<input type="checkbox"/>	 backup_2025-06-20_13-27-07.tar.gz	gz	June 20, 2025, 18:57:08 (UTC+05:30)		1.2 MB Standard

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

demo-automation-backup [Info](#)

demo-automation-backup Info						
Objects Properties Permissions Metrics Management Access Points						
Objects (3) Refresh Copy S3 URI Copy URL Download Open Delete Actions						
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.						
<input type="text" value="Find objects by prefix"/>						
<input type="checkbox"/>	Name	Type	Last modified	Size	Storage class	Tags
<input type="checkbox"/>	backup_2025-06-22_08-55-40.tar.gz	gz	June 22, 2025, 14:25:41 (UTC+05:30)	20.0 B	Standard	

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

Microsoft Bing

Search: 17:14 ist in utc time?

English Sign in 200 Mobile

ALL SEARCH IMAGES VIDEOS MAPS NEWS COPILOT MORE TOOLS

About 1,320,000 results

Convert time zones

India Standard Time (IST) Coordinated Universal Time (UTC)

5:14 PM 11:44 AM

Feedback

Worldtime Buddy
https://www.worldtimebuddy.com › ist-to-utc-converter
IST to UTC Converter - Convert India Time to Universal Time
Quickly convert India Standard Time (IST) to Universal Time (UTC) with this easy-to-use, modern time zone converter.

Utc to Ist Converter - Conver...
Converting UTC to IST. This time zone converter lets you visually and very ...

Ist to Copenhagen Time
Converting IST to Copenhagen Time. This time zone converter lets you visually and ...

Ist to Hong Kong Time
Converting IST to Hong Kong Time. This time zone converter lets you visually and ...

Ist to Melbourne Time
Converting IST to Melbourne Time. This time zone converter lets you visually and ...

Ist to Austin Time
Converting IST to Austin Time. This time zone converter lets you visually and very ...

GMT to Utc Converter
Converting GMT to UTC. This time zone converter lets you visually and very ...

Ist to Gmt Converter
Converting IST to GMT. This time zone converter lets you visually and very ...

Pst to Utc Converter
Converting PST to UTC. This time zone converter lets you visually and very ...

Coordinated Universal Time
Primary time standard

Coordinated Universal Time is the primary time standard globally used to regulate clocks and time. It establishes a reference for the current time, forming the basis for civil time and time zones. UTC...

Wikipedia

Timeline

1970 The International Bureau of Weights and Measures (BIPM) started publishing monthly bulletins of TAL.

1972 Coordinated Universal Time (UTC) was officially adopted as the uniform time scale for broadcasting and...

1999 The Global Positioning System (GPS) Time Scale was aligned to UTC(USNO) within 100 ns.

26°C Mostly cloudy

Search the web

ENG IN 17:11 22-06-2025

```
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

demo-automation-backup

Info

Objects

Properties

Permissions

Metrics

Management

Access Points

Objects (2)

Copy S3 URI

Copy URL

Download

Open

Delete

Actions

Create folder

Upload

Find objects by prefix

	Name	Type	Last modified	Size	Storage class
<input type="checkbox"/>	backup_2025-06-20_13-23-25.tar.gz	gz	June 20, 2025, 18:53:27 (UTC+05:30)	1.2 MB	Standard
<input type="checkbox"/>	backup_2025-06-20_13-27-07.tar.gz	gz	June 20, 2025, 18:57:08 (UTC+05:30)	1.2 MB	Standard

After 17:14

Backup Successfully using cronjob

demo-automation-backup

Info

Objects

Properties

Permissions

Metrics

Management

Access Points

Objects (3)

Copy S3 URI

Copy URL

Download

Open

Delete

Actions

Create folder

Upload

Find objects by prefix

	Name	Type	Last modified	Size	Storage class
<input type="checkbox"/>	backup_2025-06-20_13-23-25.tar.gz	gz	June 20, 2025, 18:53:27 (UTC+05:30)	1.2 MB	Standard
<input type="checkbox"/>	backup_2025-06-20_13-27-07.tar.gz	gz	June 20, 2025, 18:57:08 (UTC+05:30)	1.2 MB	Standard
<input type="checkbox"/>	backup_2025-06-22_17-14-01.tar.gz	gz	June 22, 2025, 17:14:02 (UTC+05:30)	1.2 MB	Standard

Finally Mini Project completed successfully after a lot of troubleshooting.