

## Step 1: Create an S3 Bucket

1. Go to **AWS Management Console** → **S3** → **Create bucket**
2. Give a unique name like:
3. sakshi-ec2-backup-bucket
4. Choose your region (same as EC2 region).
5. Keep other settings default → **Create bucket.**

The screenshot shows the 'Create bucket' configuration page in the AWS Management Console. The 'General configuration' section is active. It includes fields for 'Bucket type' (set to 'General purpose'), 'Bucket name' (set to 'sakshi-ec2-backup-bucket'), and 'Object Ownership' (set to 'ACLs disabled (recommended)'). Other sections like 'Advanced settings' and 'Tags' are visible but not filled.

## Step 2: Create an IAM Role for EC2

1. Go to **IAM** → **Roles** → **Create Role**
2. Choose **AWS Service** → **EC2**
3. Click **Next: Permissions**
4. Attach the following policy:
  - **AmazonS3FullAccess** (for simplicity in learning)
  - In real projects, use a **custom policy** to limit access to only your backup bucket.
5. Name the role:
6. EC2S3BackupRole

## 7. Click Create Role

**Create bucket Info**  
Buckets are containers for data stored in S3.

**General configuration**

**AWS Region**  
US East (N. Virginia) us-east-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**  
sakshi-ec2-backup-bucket

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.

**Object Ownership**

ACLs disabled (recommended)

ACLs enabled

**Add permissions Info**

**Permissions policies (1/1081) Info**  
Choose one or more policies to attach to your new role.

**Filter by Type**

Policy name	Type	Description
<input checked="" type="checkbox"/> AmazonS3FullAccess	AWS managed	Provides full access to all buckets via the ...

**Set permissions boundary - optional**

**Cancel** **Previous** **Next**

**Name, review, and create**

**Role details**

**Role name**  
Enter a meaningful name to identify this role.  
**EC2S3BackupRole**

**Description**  
Add a short explanation for this role.  
Allows EC2 instances to call AWS services on your behalf.

**Step 1: Select trusted entities**

**Trust policy**

```

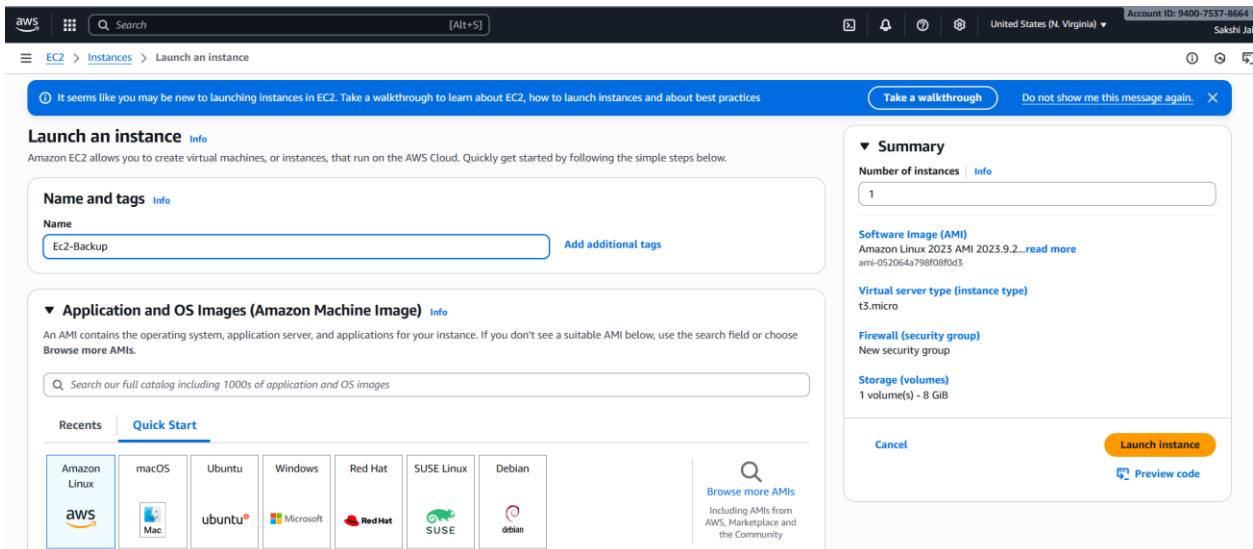
1 ~ [{ Version": "2012-10-17",
2 ~   "Statement": [
3 ~     {
4 ~       "Effect": "Allow",
5 ~       "Action": [
6 ~         "sts:AssumeRole"
7 ~       ],
8 ~       "Principal": [
9 ~

```

---

## Step 3: Launch an EC2 Instance

1. Go to **EC2 → Launch Instance**
2. Select:
  - Amazon Linux 2 or Ubuntu (Recommended)
  - t2.micro (Free Tier)
3. Under **IAM Role**, choose:
4. EC2S3BackupRole
5. Add key pair → Launch instance



**Instance type** t2.micro  
 Family: t2  
 1 vCPU - 1 GiB Memory Current generation: true On-Demand Windows base pricing: 0.0162 USD per Hour  
 On-Demand Ubuntu Pro base pricing: 0.0134 USD per Hour On-Demand SUSE base pricing: 0.0116 USD per Hour  
 On-Demand RHEL base pricing: 0.026 USD per Hour On-Demand Linux base pricing: 0.0116 USD per Hour

**Key pair (login)** download

**Network settings** Auto-assign public IP

**Summary**  
 Number of instances: 1  
 Software Image (AMI): Amazon Linux 2023 AMI 2023.9.2...  
 Virtual server type (instance type): t2.micro  
 Firewall (security group): New security group  
 Storage (volumes): 1 volume(s) - 8 GiB

## Step 4: Connect to EC2 via SSH

Use your terminal (Linux/Mac) or PowerShell (Windows):

**Connect** i-0bf285aa1d81d67eb

**Connection type**  
 Connect using a Public IP  
 Connect using a Private IP

**Public IPv4 address**  
 54.85.15.226

**Username**  
 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

## Step 5: Install AWS CLI

Check if AWS CLI is installed:

```
aws --version
```

If not installed (Ubuntu example):

```
sudo apt update
```

```
sudo apt install awscli -y
```

```
Amazon Linux 2023
https://aws.amazon.com/linux/amazon-linux-2023

[ec2-user@ip-172-31-16-78 ~]$ sudo -i
[root@ip-172-31-16-78 ~]# aws --version
aws-cli/2.30.4 Python/3.9.23 Linux/6.1.153-175.280.amzn2023.x86_64 source/x86_64 amzn.2023
[root@ip-172-31-16-78 ~]# aws s3 ls

Unable to locate credentials. You can configure credentials by running "aws configure".
[root@ip-172-31-16-78 ~]# aws configure
AWS Access Key ID [None]: AKIA5VG347UJLUBSKOV
AWS Secret Access Key [None]: xvylsW7QizD9+nduTOHswvO0dwCxF6+l+e3W3huD
Default region name [None]:
Default output format [None]:
[root@ip-172-31-16-78 ~]# aws s3 ls
2025-09-29 17:21:03 mentore-solution001
2025-10-15 05:08:30 sakshi-ec2-backup-bucket
[root@ip-172-31-16-78 ~]#
```

## Step 6: Verify IAM Role Access

Run:

```
aws s3 ls
```

You should see your S3 buckets — no need for manual key configuration because the EC2 instance has the IAM role attached.

```
[root@ip-172-31-16-78 ~]# aws s3 ls
2025-09-29 17:21:03 mentore-solution001
2025-10-15 05:08:30 sakshi-ec2-backup-bucket
[root@ip-172-31-16-78 ~]# mkdir -p /home/backup/data
[root@ip-172-31-16-78 ~]# echo "Sample backup file" > /home/backup/data
-bash: /home/backup/data: Is a directory
[root@ip-172-31-16-78 ~]# echo "Sample backup file" > /home/backup/data/sample.txt
[root@ip-172-31-16-78 ~]#
```

## 📁 Step 7: Create Backup Directory

Example: You want to back up /home/ubuntu/data

```
mkdir -p /home/ubuntu/data
```

```
echo "Sample backup file" > /home/ubuntu/data/sample.txt
```

```
[root@ip-172-31-16-78 ~]# aws s3 ls
2025-09-29 17:21:03 mentore-solution001
2025-10-15 05:08:30 sakshi-ec2-backup-bucket
[root@ip-172-31-16-78 ~]# mkdir -p /home/ubuntu/data
[root@ip-172-31-16-78 ~]# echo "Sample backup file" > /home/ubuntu/data/sample.txt
[root@ip-172-31-16-78 ~]#
```

---

## Step 8: Create Backup Script

Create a script file:

```
sudo vim /home/backup/backup_to_s3.sh
```

Add the following code:

```
#!/bin/bash
```

```
# Variables
```

```
SOURCE_DIR="/home/backup/data"
```

```
S3_BUCKET="s3://sakshi-ec2-backup-bucket"
```

```
DATE=$(date +'%Y-%m-%d-%H-%M')
```

```
BACKUP_FILE="/tmp/backup-$DATE.tar.gz"
```

```
# Create backup archive
```

```
tar -czf $BACKUP_FILE $SOURCE_DIR
```

```
# Upload to S3
```

```
aws s3 cp $BACKUP_FILE $S3_BUCKET/
```

```
# Remove local temp backup file
```

```
rm $BACKUP_FILE
```

```
echo "Backup completed successfully at $(date)" >> /home/backup/backup_log.txt
```

```
[root@ip-172-31-16-78 ~]# vim /home/backup/backup_to_s3.sh
[root@ip-172-31-16-78 ~]# chmod +x /home/backup/backup_to_s3.sh
[root@ip-172-31-16-78 ~]# ls -ltr /home/backup/ backup_to_s3.sh
ls: invalid option -- '/'
Try 'ls --help' for more information.
[root@ip-172-31-16-78 ~]# ls -ltr /home/backup/ backup_to_s3.sh
ls: invalid option -- '/'
Try 'ls --help' for more information.
[root@ip-172-31-16-78 ~]# ls -ltr /home/backup/backup_to_s3.sh
-rwxr-xr-x. 1 root root 408 Oct 15 05:25 /home/backup/backup_to_s3.sh
[root@ip-172-31-16-78 ~]#
```

---

## Make Script Executable

```
chmod +x /home/backup/backup_to_s3.sh
```

```
[root@ip-172-31-16-78 ~]# vim /home/backup/backup_to_s3.sh
[root@ip-172-31-16-78 ~]# chmod +x /home/backup/backup_to_s3.sh
[root@ip-172-31-16-78 ~]# ls -ltr/home/backup/ backup_to_s3.sh
ls: invalid option -- '/'
Try 'ls --help' for more information.
[root@ip-172-31-16-78 ~]# ls -ltr /home/backup/ backup_to_s3.sh
ls: invalid option -- '/'
Try 'ls --help' for more information.
[root@ip-172-31-16-78 ~]# ls -ltr /home/backup/backup_to_s3.sh
-rwxr-xr-x. 1 root root 408 Oct 15 05:25 /home/backup/backup_to_s3.sh
[root@ip-172-31-16-78 ~]#
```

---

## Test It Manually

```
/home/backup/backup_to_s3.sh
```

Check S3:

```
aws s3 ls s3://sakshi-ec2-backup-bucket
```

You should see your backup file uploaded 

```
[root@ip-172-31-16-78 ~]# /home/backup/backup_to_s3.sh
tar: Removing leading '/' from member names
upload: ./tmp/backup-2025-10-15-05-33.tar.gz to s3://sakshi-ec2-backup-bucket/backup-2025-10-15-05-33.tar.gz
[root@ip-172-31-16-78 ~]# aws s3 ls s3://sakshi-ec2-backup-bucket
2025-10-15 05:33:03          178 backup-2025-10-15-05-33.tar.gz
```

---

## Step 9: Automate with Cron Job

To schedule the backup **daily at midnight**:

Open cron:

```
crontab -e
```

Add the line:

```
20 11 * * * /home/backup/backup_to_s3.sh
```

 This runs your backup script every day at **12:00 AM**.

```
[root@ip-172-31-16-78 ~]# yum install cronie -y
Last metadata expiration check: 0:02:29 ago on Wed Oct 15 05:35:19 2025.
Dependencies resolved.

=====
| Package           | Architecture | Version      | Repository |
| ---              | ---          | ---          | ---         |
=====
Installing:
| cronie           | x86_64       | 1.5.7-1.amzn2023.0.2 | amazonlinux |
| cronie-anacron   | x86_64       | 1.5.7-1.amzn2023.0.2 | amazonlinux |

Transaction Summary
=====
Install 2 Packages

Total download size: 147 k
Installed size: 341 k
Downloading Packages:
=====
(1/2): cronie-1.5.7-1.amzn2023.0.2.x86_64.rpm           [=====
(1/2): cronie-anacron-1.5.7-1.amzn2023.0.2.x86_64.rpm    0% [=====
(2/2): cronie-1.5.7-1.amzn2023.0.2.x86_64.rpm           33% [======
(2/2): cronie-1.5.7-1.amzn2023.0.2.x86_64.rpm           100% [=====

Total                                         2.0 MB/s | 147 kB 00:00

Running transaction check
```

```
[root@ip-172-31-16-78 ~]# systemctl enable crond
[root@ip-172-31-16-78 ~]# systemctl start crond
[root@ip-172-31-16-78 ~]# systemctl status crond
● crond.service - Command Scheduler
   Loaded: loaded (/usr/lib/systemd/system/crond.service; enabled; preset: enabled)
     Active: active (running) since Wed 2025-10-15 05:38:24 UTC; 10s ago
       Main PID: 26977 (crond)
          Tasks: 1 (limit: 1106)
        Memory: 992.0K
         CPU: 5ms
        CGroup: /system.slice/crond.service
                  └─26977 /usr/sbin/crond -n

Oct 15 05:38:24 ip-172-31-16-78.ec2.internal crond[26977]: (CRON) STARTUP (1.5.7)
Oct 15 05:38:24 ip-172-31-16-78.ec2.internal systemd[1]: Started crond.service - Command Scheduler.
Oct 15 05:38:24 ip-172-31-16-78.ec2.internal crond[26977]: (CRON) INFO (Syslog will be used instead of sendmail.)
Oct 15 05:38:24 ip-172-31-16-78.ec2.internal crond[26977]: (CRON) INFO (RANDOM_DELAY will be scaled with factor 13% if used.)
Oct 15 05:38:24 ip-172-31-16-78.ec2.internal crond[26977]: (CRON) INFO (running with inotify support)
[root@ip-172-31-16-78 ~]#
```

```
20 11 * * * /home/backup/backup_to_s3.sh

-- INSERT --
-- INSERT --
1          0,0-1      A1
          0,1      A1
          0,7      A1
          1,7      A1
13
```

## Step 10: Create Restore Script

Create another script file:

```
sudo vim /home/backup/restore_from_s3.sh
```

Add:

```
#!/bin/bash
```

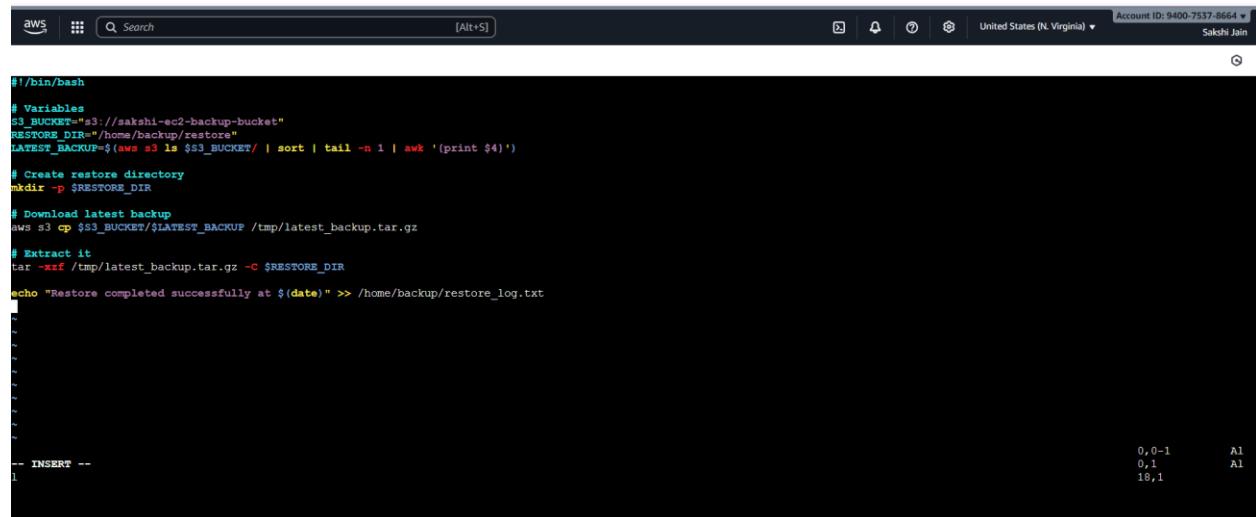
```
# Variables
S3_BUCKET="s3://sakshi-ec2-backup-bucket"
RESTORE_DIR="/home/backup/restore"
LATEST_BACKUP=$(aws s3 ls $S3_BUCKET/ | sort | tail -n 1 | awk '{print $4}')
```

```
# Create restore directory
mkdir -p $RESTORE_DIR
```

```
# Download latest backup
aws s3 cp $S3_BUCKET/$LATEST_BACKUP /tmp/latest_backup.tar.gz
```

```
# Extract it
tar -xzf /tmp/latest_backup.tar.gz -C $RESTORE_DIR
```

```
echo "Restore completed successfully at $(date)" >> /home/backup/restore_log.txt
```



The screenshot shows a terminal window with the AWS logo in the top-left corner. The title bar includes 'Search [Alt+S]', a maximize button, a minimize button, a close button, and the text 'Account ID: 9400-7537-8664' and 'United States (N. Virginia)'. Below the title bar, the user 'Sakshi Jain' is listed. The main area of the terminal displays a bash script being run. The script starts with '#!/bin/bash' and defines variables for the S3 bucket and restore directory. It then finds the latest backup file using 'aws s3 ls', creates the restore directory, downloads the backup tar file, extracts it, and finally logs the completion message to a log file. The terminal shows the script's output in yellow and the command history in white. The bottom right corner of the terminal shows line numbers 0, 0-1, 0, 1, and 18, 1.

```
#!/bin/bash
# Variables
S3_BUCKET="s3://sakshi-ec2-backup-bucket"
RESTORE_DIR="/home/backup/restore"
LATEST_BACKUP=$(aws s3 ls $S3_BUCKET/ | sort | tail -n 1 | awk '{print $4}')

# Create restore directory
mkdir -p $RESTORE_DIR

# Download latest backup
aws s3 cp $S3_BUCKET/$LATEST_BACKUP /tmp/latest_backup.tar.gz

# Extract it
tar -xzf /tmp/latest_backup.tar.gz -C $RESTORE_DIR

echo "Restore completed successfully at $(date)" >> /home/backup/restore_log.txt
```

## Make Script Executable

```
chmod +x /home/backup/restore_from_s3.sh
```

```
[ec2-user@ip-172-31-16-78 ~]$ sudo vim /home/backup/restore_from_s3.sh
[ec2-user@ip-172-31-16-78 ~]$ chmod +x /home/backup/restore_from_s3.sh
chmod: changing permissions of '/home/backup/restore_from_s3.sh': Operation not permitted
[ec2-user@ip-172-31-16-78 ~]$ sudo chmod +x /home/backup/restore_from_s3.sh
[ec2-user@ip-172-31-16-78 ~]$ sudo /home/backup/restore_from_s3.sh
download: s3://sakshi-ec2-backup-bucket/backup-2025-10-15-05-33.tar.gz to .../tmp/latest_backup.tar.gz
[ec2-user@ip-172-31-16-78 ~]$
```

## Run Restore Script

/home/backup/restore\_from\_s3.sh

Your data will be restored under /home/backup/restore/.

The screenshot shows the AWS S3 console interface. The left sidebar lists 'General purpose buckets' including Directory buckets, Table buckets, Vector buckets, Access Grants, Access Points (General Purpose Buckets, FSx file systems), Access Points (Directory Buckets), Object Lambda Access Points, Multi-Region Access Points, Batch Operations, and IAM Access Analyzer for S3. It also includes sections for Block Public Access settings and Storage Lens. The main content area shows the 'sakshi-ec2-backup-bucket' with one object listed:

Name	Type	Last modified	Size	Storage class
backup-2025-10-15-05-33.tar.gz	gz	October 15, 2025, 11:05:03 (UTC+05:30)	178.0 B	Standard