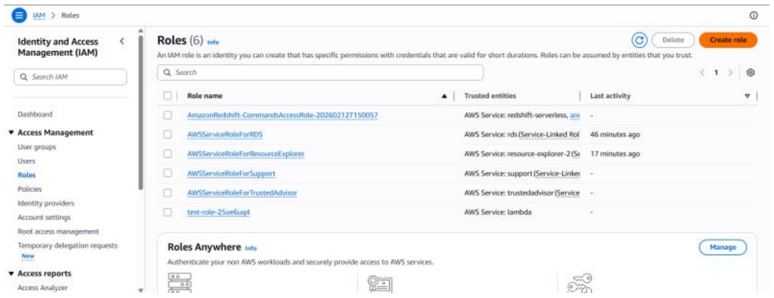
Secure S3 Access from EC2 using IAM Role

In this project, I configured secure access between EC2 and S3 using an IAM Role instead of hardcoding access keys inside the server

STEP 1 : create an iam role



Roles (6) Info

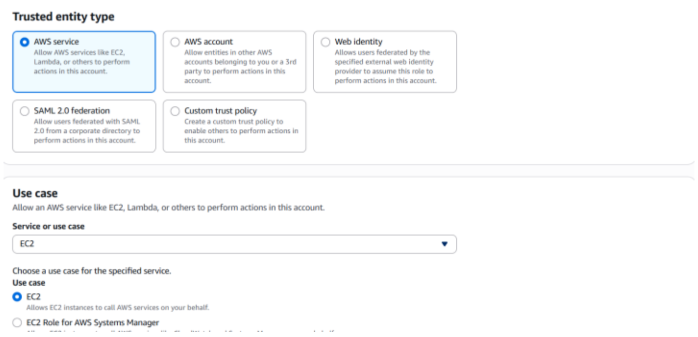
An IAM role is an identity you can create that has specific permissions with credentials that are valid for short durations. Roles can be assumed by entities that you trust.

Role name	Trusted entities	Last activity
AmazonRedshift-CommandsAccessRole-20260212T150057	AWS Service: redshift-serverless, an	-
AWSServiceRoleForRDS	AWS Service: rds (Service-Linked Rol	46 minutes ago
AWSServiceRoleForResourceExplorer	AWS Service: resource-explorer-2 (Se	17 minutes ago
AWSServiceRoleForSupport	AWS Service: support (Service-Linke	-
AWSServiceRoleForTrustedAdvisor	AWS Service: trustedadvisor (Service	-
test-role-25ue6uq4	AWS Service: lambda	-

Roles Anywhere Info

Authenticate your non AWS workloads and securely provide access to AWS services.

- On that click create role



Trusted entity type

[✓ AWS service] Allow AWS services like EC2, Lambda, or others to perform actions in this account.

AWS account: Allow entities in other AWS accounts belonging to you or a 3rd party to perform actions in this account.

Web identity: Allows users federated by the specified external web identity provider to assume this role to perform actions in this account.

SAML 2.0 federation: Allow users federated with SAML 2.0 from a corporate directory to perform actions in this account.

Custom trust policy: Create a custom trust policy to enable others to perform actions in this account.

Use case

Allow an AWS service like EC2, Lambda, or others to perform actions in this account.

Service or use case: EC2

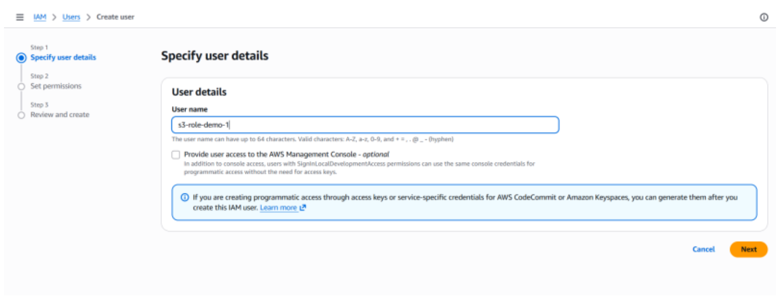
Choose a use case for the specified service.

Use case

[✓ EC2] Allows EC2 instances to call AWS services on your behalf.

EC2 Role for AWS Systems Manager

- Choose that what you want to work with



Specify user details

User details

User name: s3-role-demo-1

The user name can have up to 64 characters. Valid characters: A-Z, a-z, 0-9, and + = , . @ _ - (hyphen)

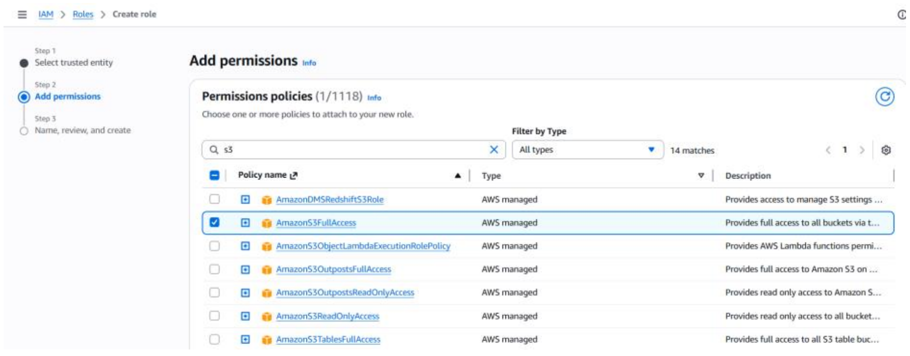
☐ Provide user access to the AWS Management Console - *optional*

In addition to console access, users with SignInLocalDevelopmentAccess permissions can use the same console credentials for programmatic access without the need for access keys.

If you are creating programmatic access through access keys or service-specific credentials for AWS CodeCommit or Amazon Keyspaces, you can generate them after you create this IAM user. Learn more

- Give an specfic name for that role

- After that give the permissions to that



Add permissions

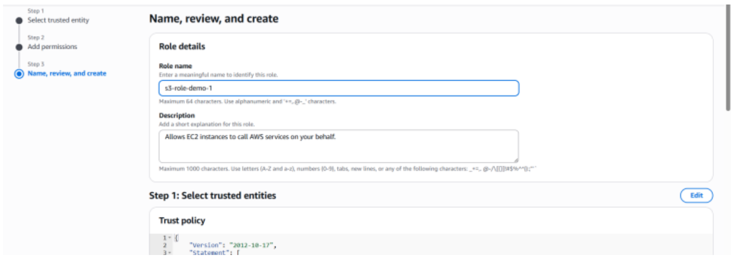
Permissions policies (1/1118)

Choose one or more policies to attach to your new role.

Search: s3 — Filter by Type: All types — 14 matches — 1

	Policy name	Type	Description
☐	AmazonDMSRedshiftS3Role	AWS managed	Provides access to manage S3 settings ...
✓	AmazonS3FullAccess	AWS managed	Provides full access to all buckets via t...
☐	AmazonS3ObjectLambdaExecutionRolePolicy	AWS managed	Provides AWS Lambda functions permi...
☐	AmazonS3OutpostsFullAccess	AWS managed	Provides full access to Amazon S3 on ...
☐	AmazonS3OutpostsReadOnlyAccess	AWS managed	Provides read only access to Amazon S...
☐	AmazonS3ReadOnlyAccess	AWS managed	Provides read only access to all bucket...
☐	AmazonS3TablesFullAccess	AWS managed	Provides full access to all S3 table buc...

- On that we are going to use s3 so choose s3 full acess
- After that give an name for that role



Name, review, and create

Role details

Role name

Enter a meaningful name to identify this role.

s3-role-demo-1

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

Description

Add a short explanation for this role.

Allows EC2 instances to call AWS services on your behalf.

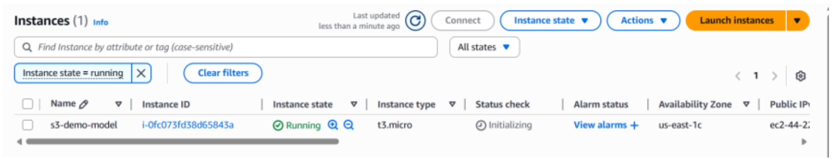
Maximum 1000 characters. Use letters (A-Z and a-z), numbers (0-9), tabs, new lines, or any of the following characters: _+=,.@-/\[{()}]!#$%^*();:"'

Step 1: Select trusted entities

Trust policy

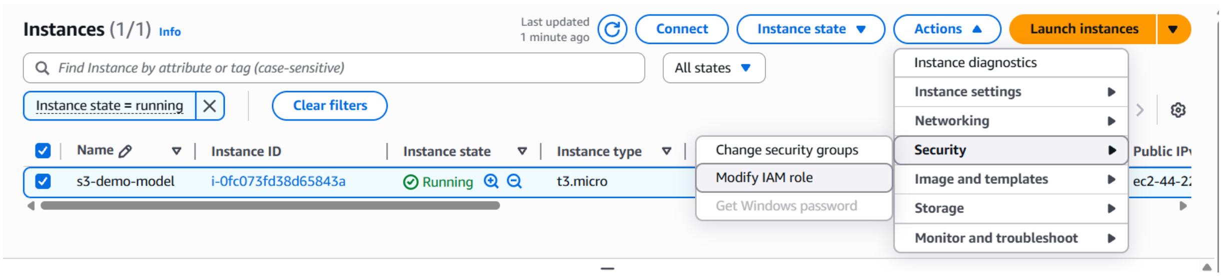
```
{
    "Version": "2012-10-17",
    "Statement": [
```

- A role has created
- After that create an ec2



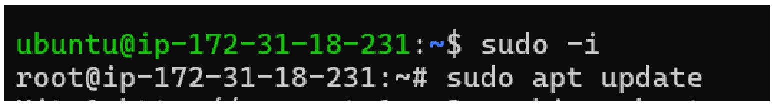
	Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public IP
☐	s3-demo-model	i-0fc073fd38d65843a	Running	t3.micro	Initializing	View alarms +	us-east-1c	ec2-44-2

- After that creating the instance we have already done an iam role policy attach the iam role to ec2 instance



	Name	Instance ID	Instance state	Instance type	Public IP
✓	s3-demo-model	i-0fc073fd38d65843a	Running	t3.micro	ec2-44-2

- After attaching the iam role connect to the server

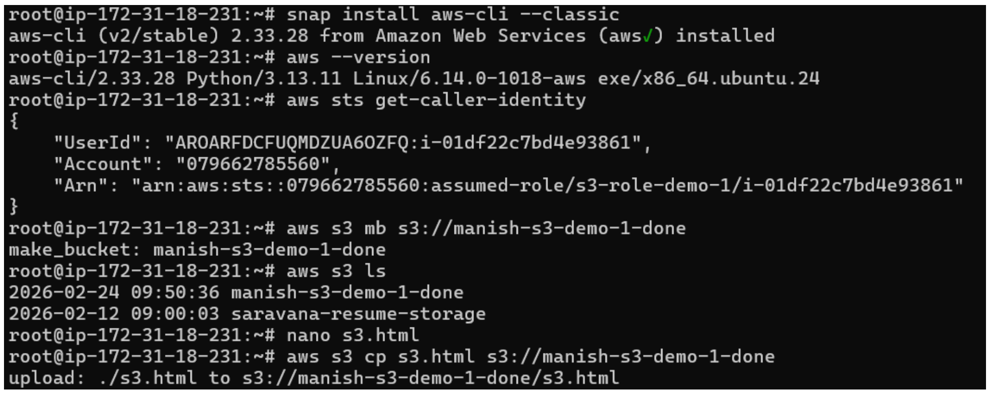


```
ubuntu@ip-172-31-18-231:~$ sudo -i
root@ip-172-31-18-231:~# sudo apt update
```

Sudo -i

Sudo apt update

- Change to the root user
- And then update

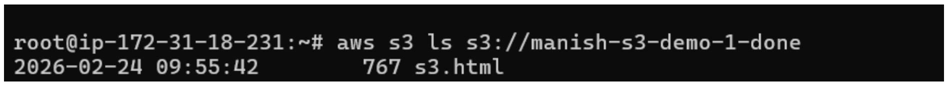


```
root@ip-172-31-18-231:~# snap install aws-cli --classic
aws-cli (v2/stable) 2.33.28 from Amazon Web Services (aws✓) installed
root@ip-172-31-18-231:~# aws --version
aws-cli/2.33.28 Python/3.13.11 Linux/6.14.0-1018-aws exe/x86_64.ubuntu.24
root@ip-172-31-18-231:~# aws sts get-caller-identity
{
    "UserId": "AROARFDCFUQMDZUA6OZFQ:i-01df22c7bd4e93861",
    "Account": "079662785560",
    "Arn": "arn:aws:sts::079662785560:assumed-role/s3-role-demo-1/i-01df22c7bd4e93861"
}
root@ip-172-31-18-231:~# aws s3 mb s3://manish-s3-demo-1-done
make_bucket: manish-s3-demo-1-done
root@ip-172-31-18-231:~# aws s3 ls
2026-02-24 09:50:36 manish-s3-demo-1-done
2026-02-12 09:00:03 saravana-resume-storage
root@ip-172-31-18-231:~# nano s3.html
root@ip-172-31-18-231:~# aws s3 cp s3.html s3://manish-s3-demo-1-done
upload: ./s3.html to s3://manish-s3-demo-1-done/s3.html
```

- Install aws cli on that
- Snap is used for installing the softwre
- And aws cli is ude for we can make a work without manual way in the terimal
- Now check your version
- After that check that status of this you entered here
- Now we can create the bucket
- Mb means make bucket in that s3://choose your name here
- Now view your bucket
- Now create a nano file or vim file
- Now push the nano inside your bucket
- Cp means copy on that first we want to give your nano file after that your bucket file name

sudo apt install awscli -y - to install it to access the aws command

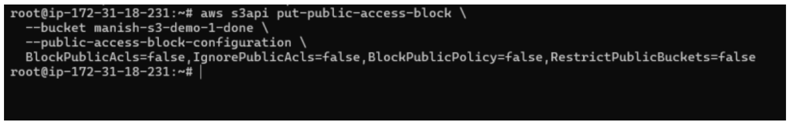
```
snap install aws-cli --classic

aws –version

aws sts get-caller-identity

aws s3 mb s3:// manish-s3-demo-1-done

nano s3.html

aws s3 cp s3.html s3://kanisma-ec2-project-001/
```



```
root@ip-172-31-18-231:~# aws s3 ls s3://manish-s3-demo-1-done
2026-02-24 09:55:42        767 s3.html
```

```
aws s3 ls s3://manish-s3-demo-1-done
```

- Now view this after that
- Now disable the public acess



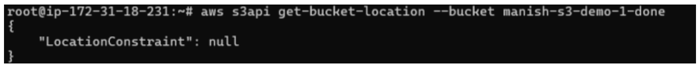
```
root@ip-172-31-18-231:~# aws s3api put-public-access-block \
  --bucket manish-s3-demo-1-done \
  --public-access-block-configuration \
  BlockPublicAcls=false,IgnorePublicAcls=false,BlockPublicPolicy=false,RestrictPublicBuckets=false
root@ip-172-31-18-231:~# |
```

```
aws s3api put-public-access-block \

-- manish-s3-demo-1-done \

--public-access-block-configuration
BlockPublicAcls=false,IgnorePublicAcls=false,BlockPublicPolicy=false,RestrictPublicBuckets=fa
lse
```

- After that add public read bucket policy
- nano policy.json - opens text editor,paste the json code
- {

- "Version": "2012-10-17",
- "Statement": [
- {aws
- "Sid": "PublicRead",
- "Effect": "Allow",
- "Principal": "*",
- "Action": "s3:GetObject",
- "Resource": "arn:aws:s3:::manish-ec2-project-2026/*"
- }
-]
- }

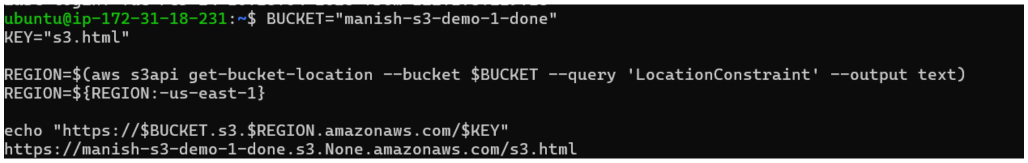
- After that get theget the bucket location



```
root@ip-172-31-18-231:~# aws s3api get-bucket-location --bucket manish-s3-demo-1-done
{
    "LocationConstraint": null
}
```

aws s3api get-bucket-location --bucket manish-s3-demo-1-done

- After that get the object url link



```
ubuntu@ip-172-31-18-231:~$ BUCKET="manish-s3-demo-1-done"
KEY="s3.html"

REGION=$(aws s3api get-bucket-location --bucket $BUCKET --query 'LocationConstraint' --output text)
REGION=${REGION:-us-east-1}

echo "https://$BUCKET.s3.$REGION.amazonaws.com/$KEY"
https://manish-s3-demo-1-done.s3.None.amazonaws.com/s3.html
```

BUCKET="manish-s3-demo-1-done"

KEY="s3.html"

REGION=$(aws s3api get-bucket-location --bucket $BUCKET --query 'LocationConstraint' --output text)

REGION=${REGION:-us-east-1}

echo https://$BUCKET.s3.$REGION.amazonaws.com/$KEY

- If you want to delete that s3 bucket

  aws s3 rb s3:// manish-s3-demo-1-done

- If you want to delete that file

  ```
  Aws s3 rm s3:// manish-s3-demo-1-done
  ```