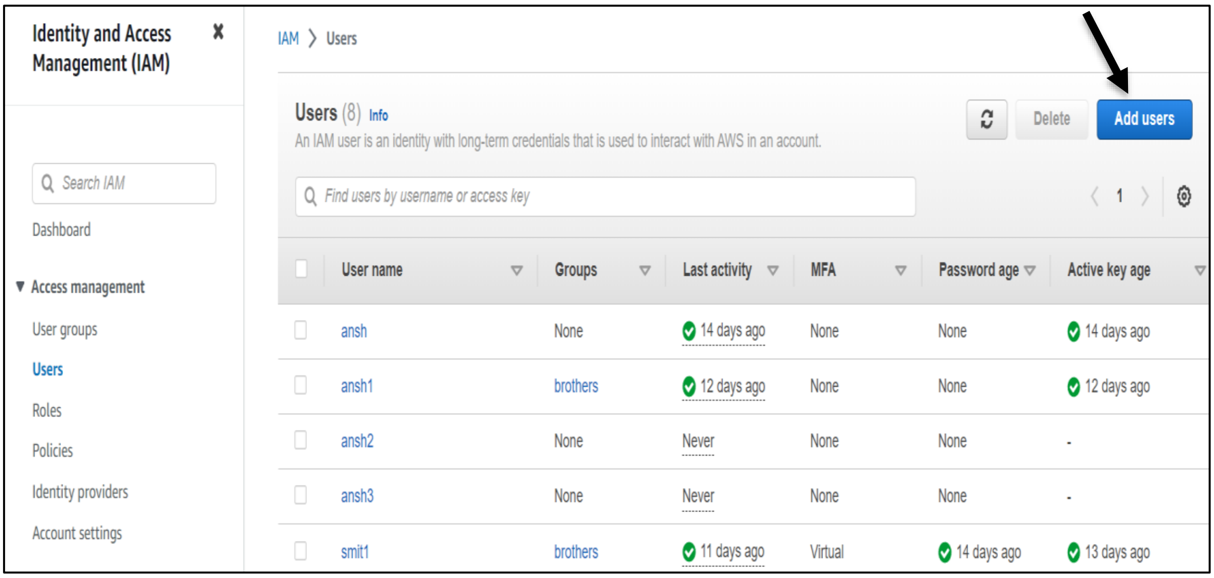
<u>Add User by Programmatic Access</u> (Access Key and Secret key)

- Login to your IAM console
- In the left navigation panel choose **Users** in **Access Management** section.
- Click on **Add Users**

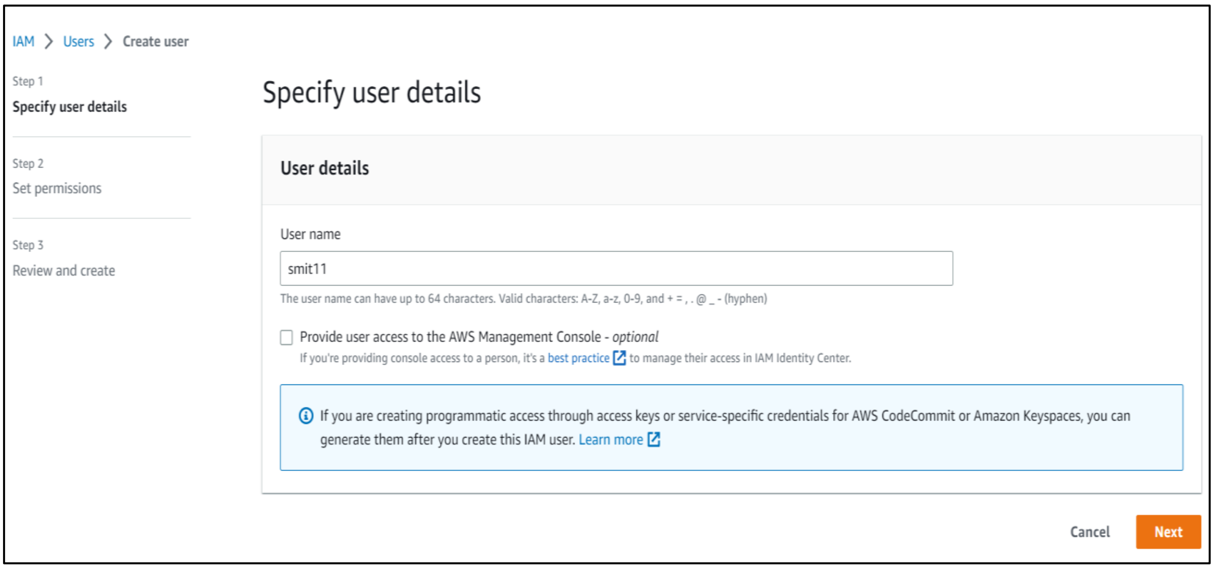


Users (8) Info

An IAM user is an identity with long-term credentials that is used to interact with AWS in an account.

	User name	Groups	Last activity	MFA	Password age	Active key age
□	ansh	None	✓ 14 days ago	None	None	✓ 14 days ago
□	ansh1	brothers	✓ 12 days ago	None	None	✓ 12 days ago
□	ansh2	None	Never	None	None	-
□	ansh3	None	Never	None	None	-
□	smit1	brothers	✓ 11 days ago	Virtual	✓ 14 days ago	✓ 13 days ago

<u>Step 1</u> :- Specify user details



Specify user details

User details

User name

smit11

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

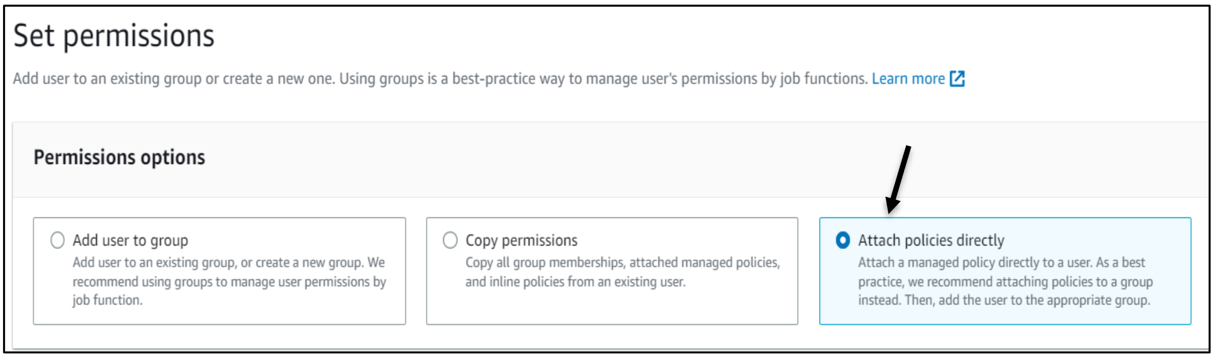
□ Provide user access to the AWS Management Console - *optional*

If you're providing console access to a person, it's a best practice to manage their access in IAM Identity Center.

ⓘ 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

- Enter **User name**
- Do not enable **Provide user access to the AWS Management Console**
- Click on **Next**

Step 2 :- Set Permissions



Set permissions

Add user to an existing group or create a new one. Using groups is a best-practice way to manage user's permissions by job functions. Learn more

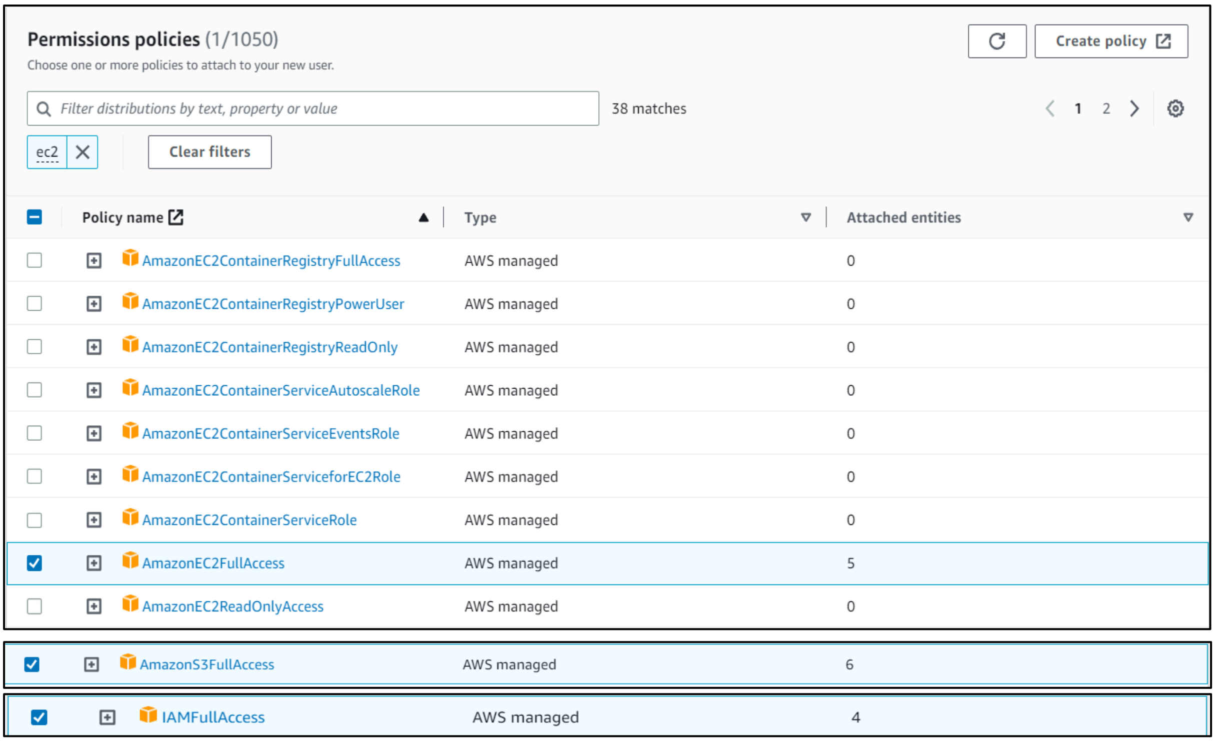
Permissions options

○ Add user to group
Add user to an existing group, or create a new group. We recommend using groups to manage user permissions by job function.

○ Copy permissions
Copy all group memberships, attached managed policies, and inline policies from an existing user.

● Attach policies directly
Attach a managed policy directly to a user. As a best practice, we recommend attaching policies to a group instead. Then, add the user to the appropriate group.

- In Permission Options select **Attach policies directly**

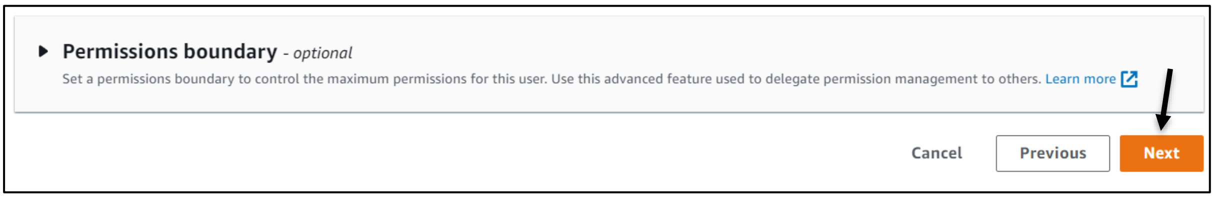


Permissions policies (1/1050)

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

		Policy name	Type	Attached entities
□	⊞	AmazonEC2ContainerRegistryFullAccess	AWS managed	0
□	⊞	AmazonEC2ContainerRegistryPowerUser	AWS managed	0
□	⊞	AmazonEC2ContainerRegistryReadOnly	AWS managed	0
□	⊞	AmazonEC2ContainerServiceAutoscaleRole	AWS managed	0
□	⊞	AmazonEC2ContainerServiceEventsRole	AWS managed	0
□	⊞	AmazonEC2ContainerServiceforEC2Role	AWS managed	0
□	⊞	AmazonEC2ContainerServiceRole	AWS managed	0
☑	⊞	AmazonEC2FullAccess	AWS managed	5
□	⊞	AmazonEC2ReadOnlyAccess	AWS managed	0
☑	⊞	AmazonS3FullAccess	AWS managed	6
☑	⊞	IAMFullAccess	AWS managed	4

- In Permissions policies select **AmazonEC2FullAccess**, **Amazon S3FullAccess** and **IAMFullAccess**

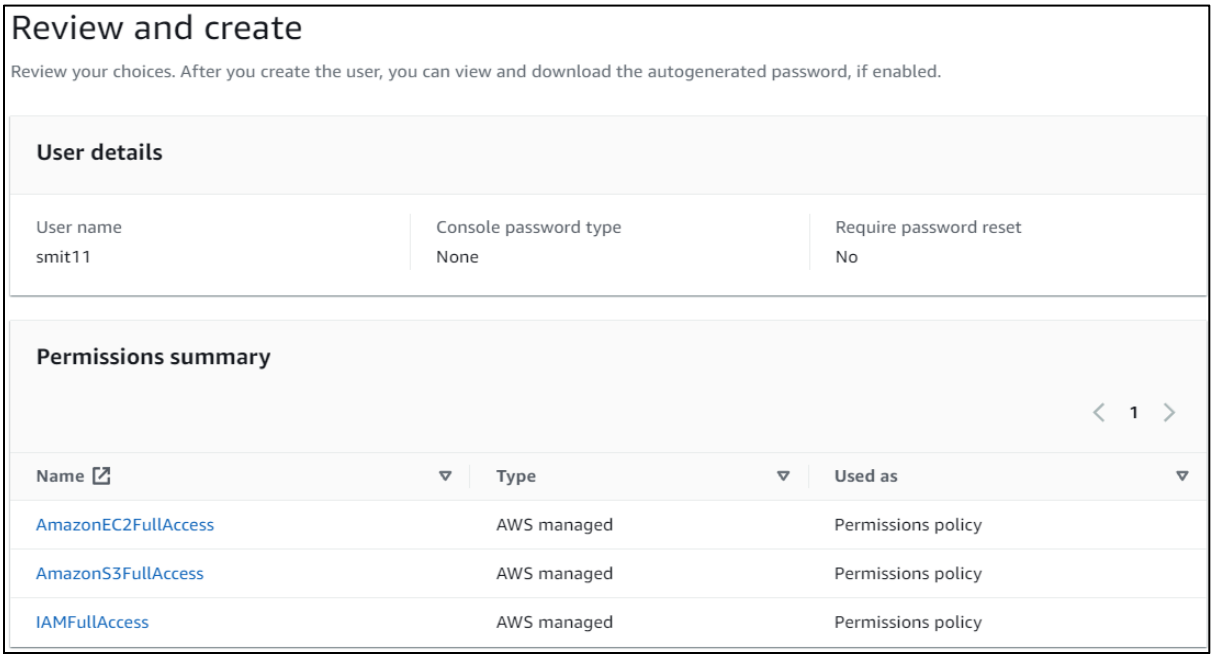


▸ **Permissions boundary** - *optional*

Set a permissions boundary to control the maximum permissions for this user. Use this advanced feature used to delegate permission management to others. Learn more

- We can set **Permissions boundary** to control the maximum permissions for this user. Use this advanced feature used to delegate permission management to others.
- Click on **Next**

Step 3 :- Review and create



Review and create

Review your choices. After you create the user, you can view and download the autogenerated password, if enabled.

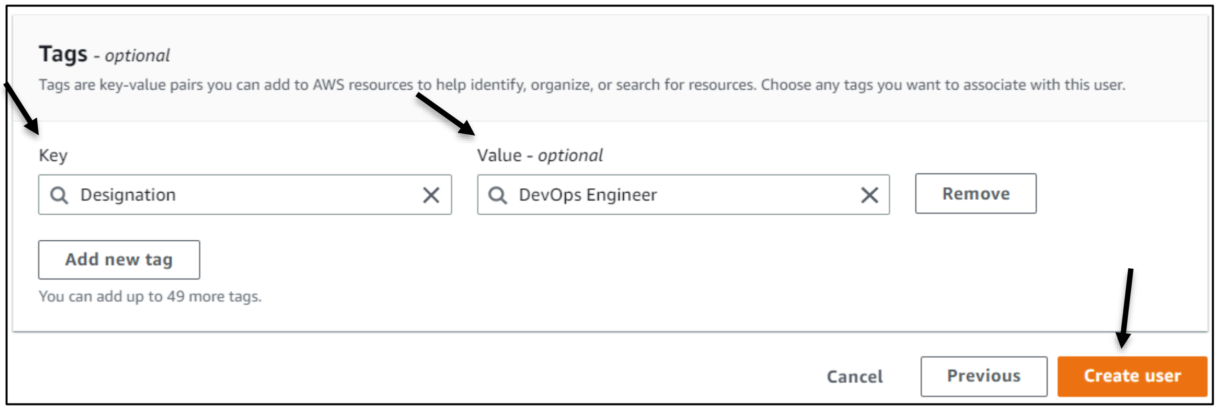
User details

User name	Console password type	Require password reset
smit11	None	No

Permissions summary

Name	Type	Used as
AmazonEC2FullAccess	AWS managed	Permissions policy
AmazonS3FullAccess	AWS managed	Permissions policy
IAMFullAccess	AWS managed	Permissions policy

- We can add **Tags** to AWS resources to help identify, organize, or search for resources.



Tags - *optional*

Tags are key-value pairs you can add to AWS resources to help identify, organize, or search for resources. Choose any tags you want to associate with this user.

Key: Designation

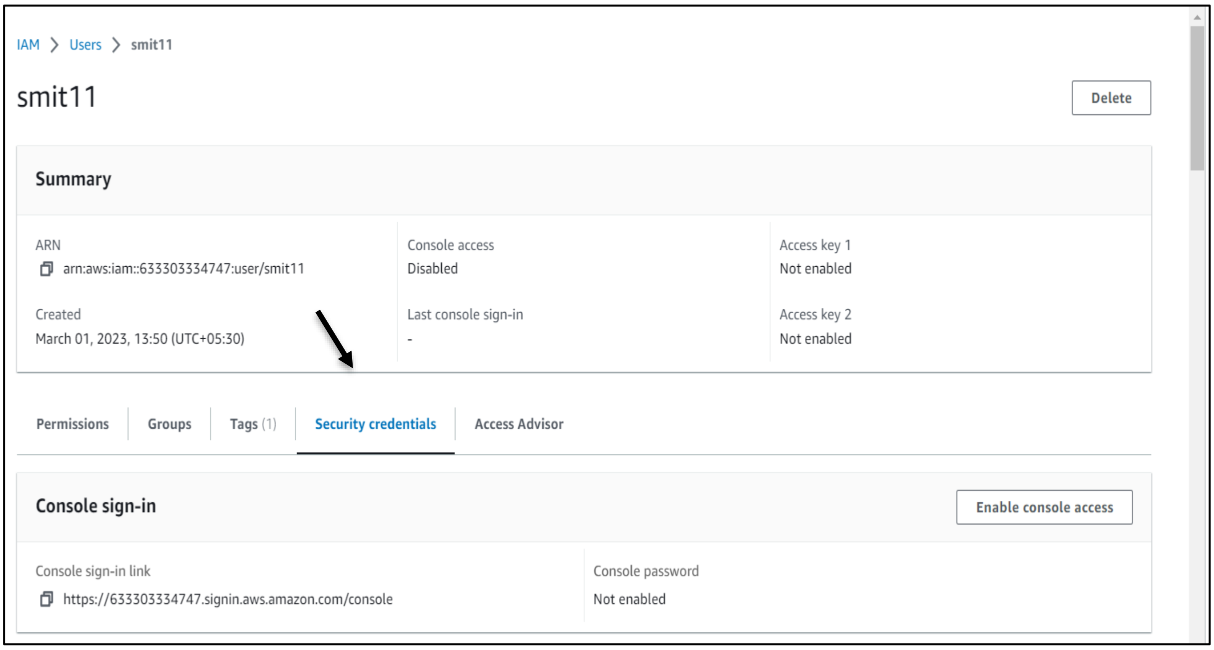
Value - *optional*: DevOps Engineer

You can add up to 49 more tags.

- Click on **Create user**
- So now user **"smit11"** has been created in users list by using Programmatic Access.

❖ To Create Access Keys

- Click on username **"smit11"** from **users list.**



IAM > Users > smit11

smit11

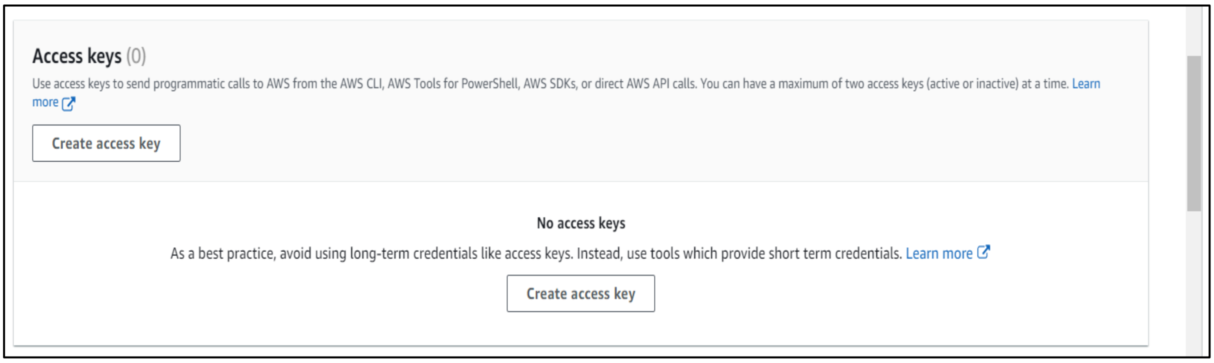
Summary

ARN	Console access	Access key 1
arn:aws:iam::633303334747:user/smit11	Disabled	Not enabled
Created	Last console sign-in	Access key 2
March 01, 2023, 13:50 (UTC+05:30)	-	Not enabled

Console sign-in

Console sign-in link	Console password
https://633303334747.signin.aws.amazon.com/console	Not enabled

- Go to option **Security credentials** and scroll down to **Access keys** option.



Access keys (0)

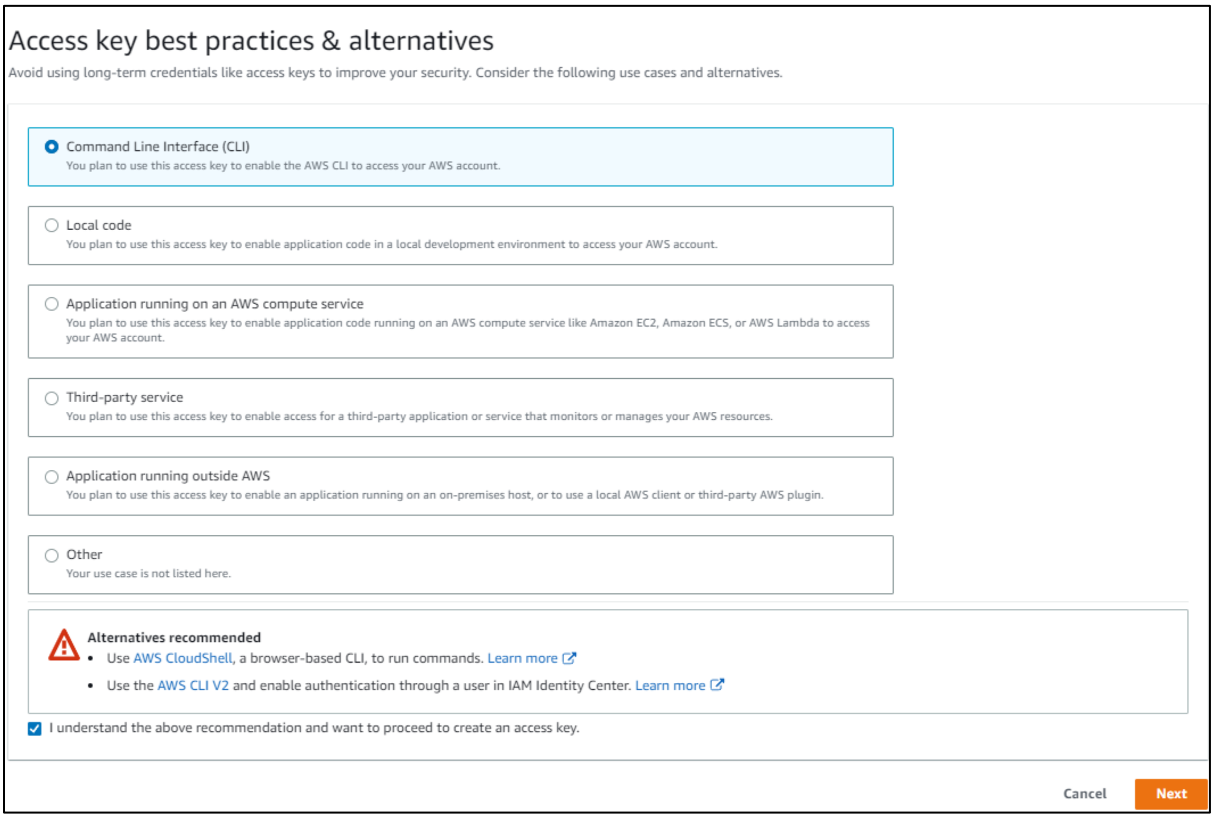
Use access keys to send programmatic calls to AWS from the AWS CLI, AWS Tools for PowerShell, AWS SDKs, or direct AWS API calls. You can have a maximum of two access keys (active or inactive) at a time. Learn more

No access keys

As a best practice, avoid using long-term credentials like access keys. Instead, use tools which provide short term credentials. Learn more

- Click on **Create access key**

Step 1 :- Access key best practices & alternatives



Access key best practices & alternatives

Avoid using long-term credentials like access keys to improve your security. Consider the following use cases and alternatives.

● Command Line Interface (CLI)
You plan to use this access key to enable the AWS CLI to access your AWS account.

○ Local code
You plan to use this access key to enable application code in a local development environment to access your AWS account.

○ Application running on an AWS compute service
You plan to use this access key to enable application code running on an AWS compute service like Amazon EC2, Amazon ECS, or AWS Lambda to access your AWS account.

○ Third-party service
You plan to use this access key to enable access for a third-party application or service that monitors or manages your AWS resources.

○ Application running outside AWS
You plan to use this access key to enable an application running on an on-premises host, or to use a local AWS client or third-party AWS plugin.

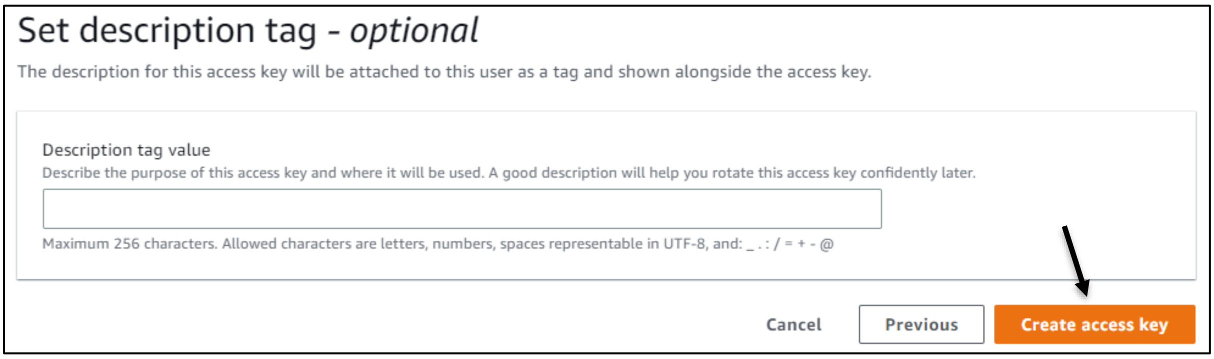
○ Other
Your use case is not listed here.

Alternatives recommended
- Use AWS CloudShell, a browser-based CLI, to run commands. Learn more
- Use the AWS CLI V2 and enable authentication through a user in IAM Identity Center. Learn more

☑ I understand the above recommendation and want to proceed to create an access key.

- Select Command Line Interface (CLI)
- Select box [ ] **I understand the above recommendation and want to proceed to create an access key.**
- Click on **Next**

Step 2 :- Set description tag



Set description tag - *optional*

The description for this access key will be attached to this user as a tag and shown alongside the access key.

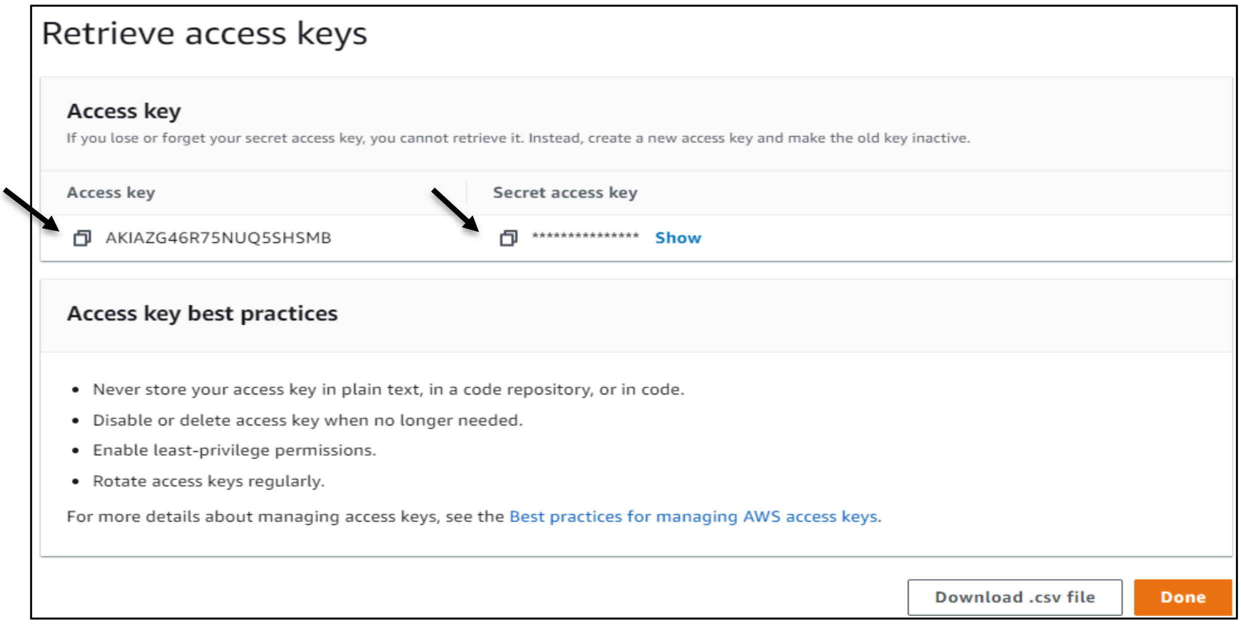
Description tag value

Describe the purpose of this access key and where it will be used. A good description will help you rotate this access key confidently later.

Maximum 256 characters. Allowed characters are letters, numbers, spaces representable in UTF-8, and: _ . : / = + - @

- Click on **Create access key**

Step 3 :- Retrieve access keys



Retrieve access keys

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
AKIAZG46R75NUQ5SHSMB	*************** Show

Access key best practices

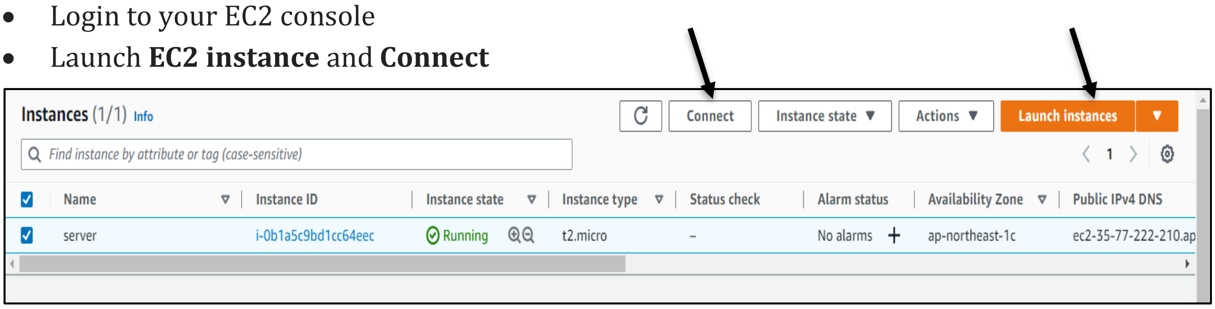
- Never store your access key in plain text, in a code repository, or in code.
- Disable or delete access key when no longer needed.
- Enable least-privilege permissions.
- Rotate access keys regularly.

For more details about managing access keys, see the Best practices for managing AWS access keys.

- So now we can copy **Access key** and **Secret Access** key in notepad or **Download .csv file**.

- **Now take access of user machine in instance by using Programmatic Access (Access and Secret Access Keys)**

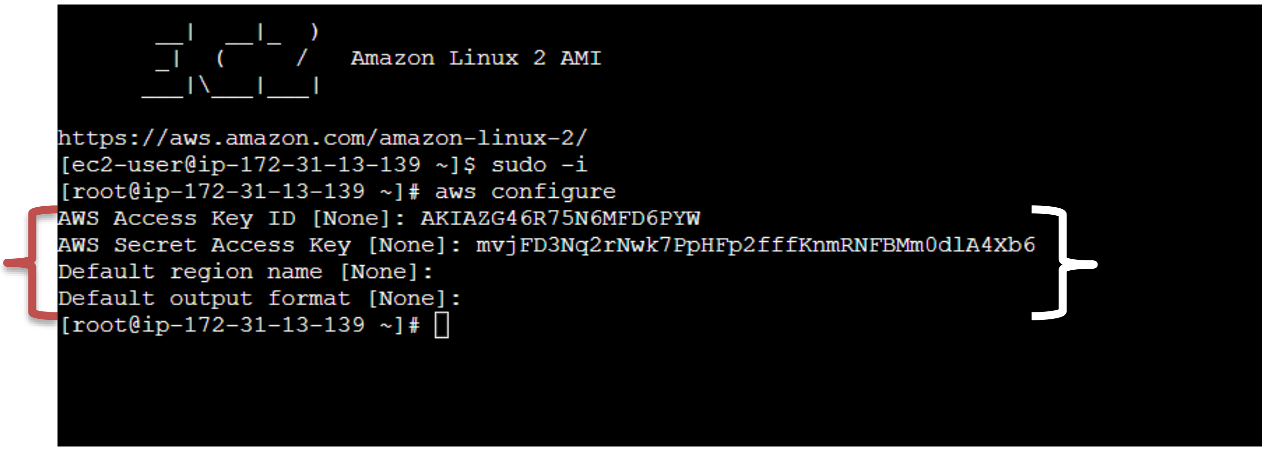
  - Login to your EC2 console
  - Launch **EC2 instance** and **Connect**



Instances (1/1) Info

☑	Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public IPv4 DNS
☑	server	i-0b1a5c9bd1cc64eec	Running	t2.micro	-	No alarms	ap-northeast-1c	ec2-35-77-222-210.ap

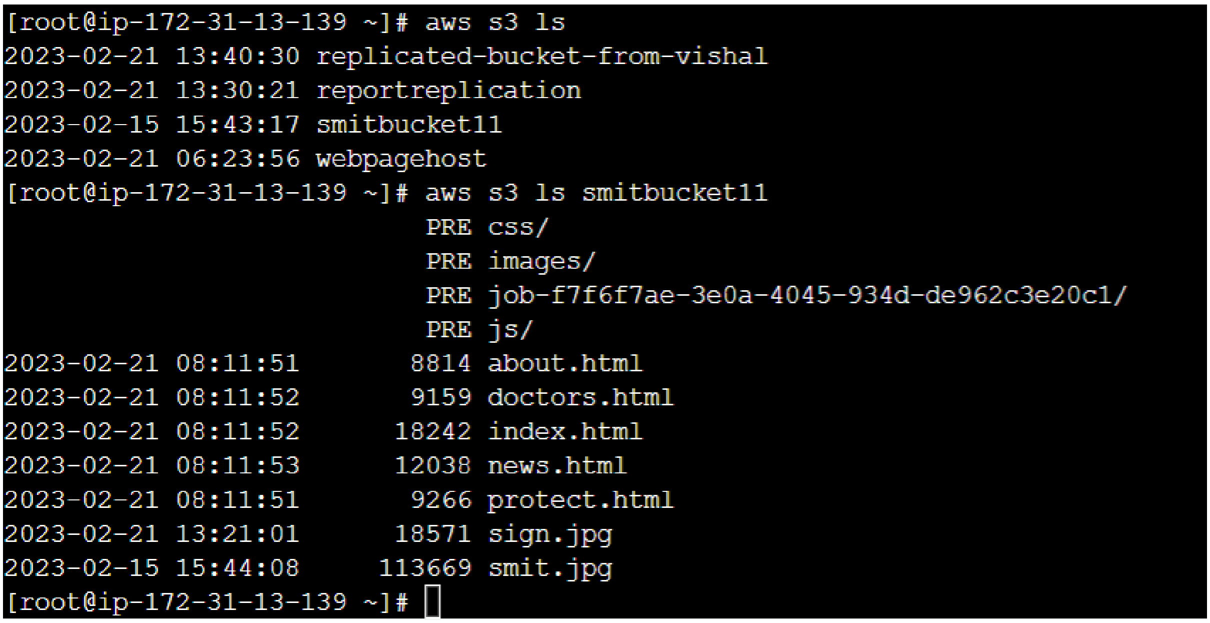
- Use command “**aws configure**” to take access in user machine and enter
  - AWS Access Key ID
  - AWS Secrets Access Key
  - Default region name
  - Default output format



```
       __|  __|_  )
       _|  (     /   Amazon Linux 2 AMI
      ___|\___|___|

https://aws.amazon.com/amazon-linux-2/
[ec2-user@ip-172-31-13-139 ~]$ sudo -i
[root@ip-172-31-13-139 ~]# aws configure
AWS Access Key ID [None]: AKIAZG46R75N6MFD6PYW
AWS Secret Access Key [None]: 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
Default region name [None]:
Default output format [None]:
[root@ip-172-31-13-139 ~]# 
```

- Now you have been accessed in user's machine by using Programmatic Access.
- And we can see the buckets and its contents because we have set permission (**attached policies of EC2, S3 and IAM services**).
  - By using "**aws s3 ls**" command , list buckets
  - By using "**aws s3 ls smitbucket11**" command, can see contents in bucket **'smitbucket11'**
- To add User by using terminal
  - **aws iam create-user –user-name username**



```
[root@ip-172-31-13-139 ~]# aws s3 ls
2023-02-21 13:40:30 replicated-bucket-from-vishal
2023-02-21 13:30:21 reportreplication
2023-02-15 15:43:17 smitbucket11
2023-02-21 06:23:56 webpagehost
[root@ip-172-31-13-139 ~]# aws s3 ls smitbucket11
                           PRE css/
                           PRE images/
                           PRE job-f7f6f7ae-3e0a-4045-934d-de962c3e20c1/
                           PRE js/
2023-02-21 08:11:51       8814 about.html
2023-02-21 08:11:52       9159 doctors.html
2023-02-21 08:11:52      18242 index.html
2023-02-21 08:11:53      12038 news.html
2023-02-21 08:11:51       9266 protect.html
2023-02-21 13:21:01      18571 sign.jpg
2023-02-15 15:44:08     113669 smit.jpg
[root@ip-172-31-13-139 ~]# 
```