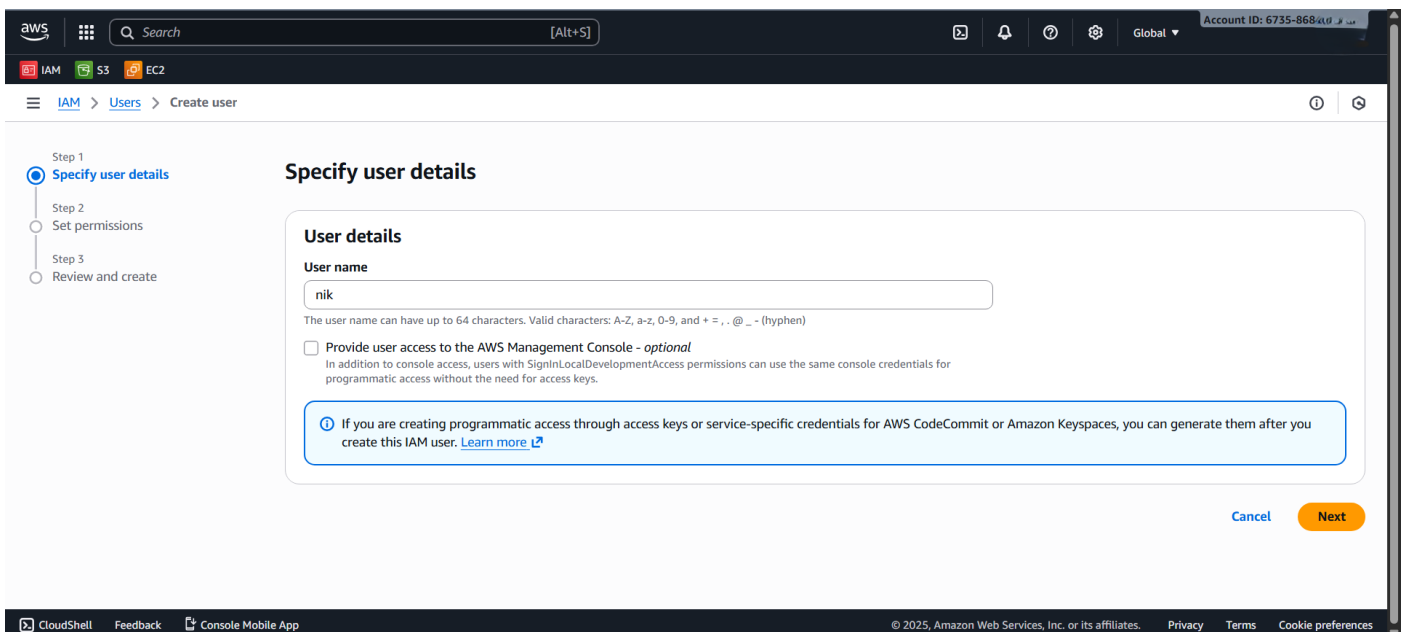
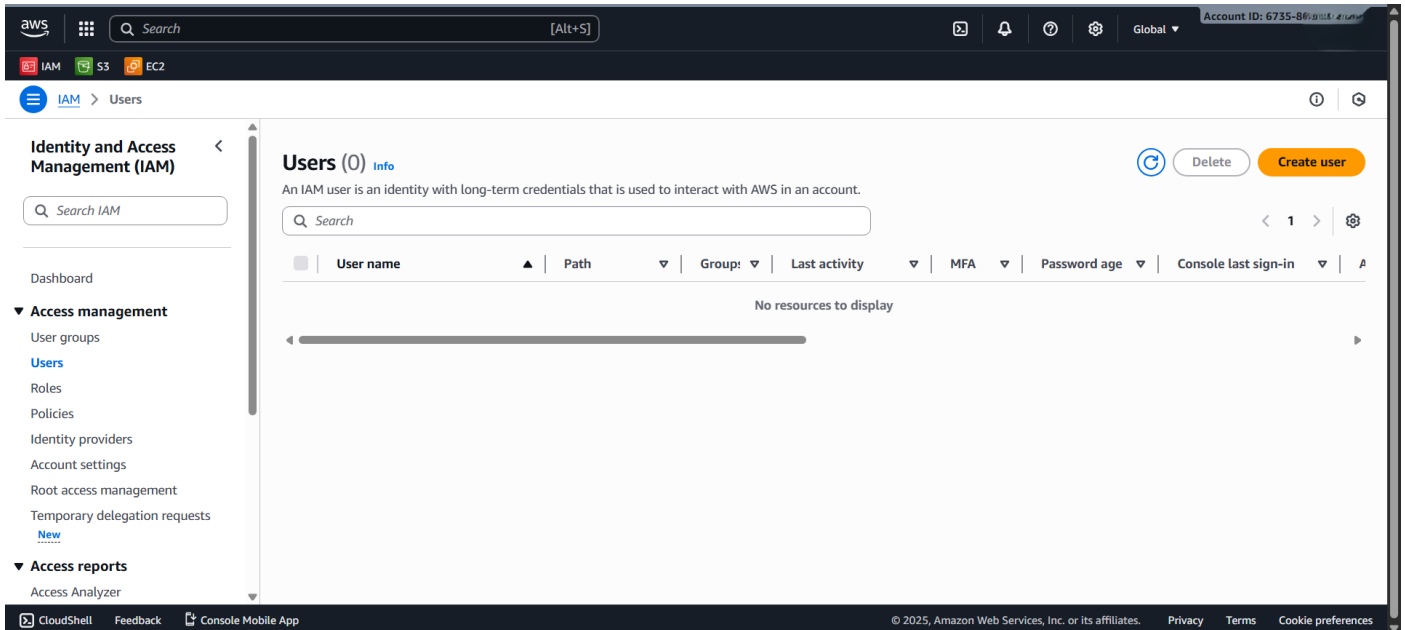


# Use CLI to Login a user which have full access IAM Services and create a user, group, policy and a role using AWS CLI

By: Nikhilesh Sakhare

1. First, an IAM user named nik was created in the console and granted the IAMFullAccess managed policy so it can manage all IAM resources.



aws

Search

[Alt+S]

IAM

S3

EC2

CloudShell

Feedback

Console Mobile App

Account ID: 6735-868-...

IAM > Users > Create user

Step 1  
Specify user details

Step 2  
**Set permissions**

Step 3  
Review and create

### 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.

Permissions policies (1/1424)

Create policy

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

iamfull

Filter by Type

All types

1 match

< 1 >

Policy name	Type	Attached entities
<input checked="" type="checkbox"/> <a href="#">IAMFullAccess</a>	AWS managed	0

CloudShell

Feedback

Console Mobile App

© 2025, Amazon Web Services, Inc. or its affiliates. [Privacy](#) [Terms](#) [Cookie preferences](#)

aws

Search

[Alt+S]

IAM

S3

EC2

CloudShell

Feedback

Console Mobile App

Account ID: 6735-868-...

IAM > Users > Create user

Step 1  
Specify user details

Step 2  
Set permissions

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

nik

Console password type

None

Require password reset

No

Permissions summary

< 1 >

Name	Type	Used as
<a href="#">IAMFullAccess</a>	AWS managed	Permissions policy

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.

No tags associated with the resource.

[Add new tag](#)

CloudShell

Feedback

Console Mobile App

© 2025, Amazon Web Services, Inc. or its affiliates. [Privacy](#) [Terms](#) [Cookie preferences](#)

aws

Search

[Alt+S]

IAM

S3

EC2

CloudShell

Feedback

Console Mobile App

Account ID: 6735-868-...

IAM > Users > nik

Identity and Access Management (IAM)

Search IAM

Dashboard

▼ Access management

User groups

**Users**

Roles

Policies

Identity providers

Account settings

Root access management

Temporary delegation requests

New

▼ Access reports

Access Analyzer

nik

Info

Delete

Summary

ARN

[arn:aws:iam::673586849368:user:nik](#)

Console access

Disabled

Access key 1

[Create access key](#)

Created

November 23, 2025, 10:20 (UTC+05:30)

Last console sign-in

-

Permissions

Groups

Tags

Security credentials

Last Accessed

Permissions policies (1)

Remove

Add permissions

Permissions are defined by policies attached to the user directly or through groups.

Search

Filter by Type

All types

< 1 >

Policy name	Type	Attached via
<input type="checkbox"/> <a href="#">IAMFullAccess</a>	AWS managed	Directly

CloudShell

Feedback

Console Mobile App

© 2025, Amazon Web Services, Inc. or its affiliates. [Privacy](#) [Terms](#) [Cookie preferences](#)

2. An access key was then created for user nik by choosing the Command Line Interface (CLI) use case and confirming the access key creation steps.

aws

Search

[Alt+S]

Global

Account ID: 6735-8684

IAM

S3

EC2

IAM > Users > nik > Create access key

Step 1

Access key best practices & alternatives

Info

Step 2 - optional

Set description tag

Step 3

Retrieve access keys

Access key best practices & alternatives

Info

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

Use case

☒ 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 authenticate workloads running in your data center or other infrastructure outside of AWS that needs to access your AWS resources.

☐ Other

Your use case is not listed here.

Alternatives recommended

Use AWS CLI V2 and the `aws login` command to use your existing console credentials in the CLI. [Learn more](#)

Use AWS CloudShell, a browser-based CLI, to run commands. [Learn more](#)

Confirmation

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

Cancel

Next

IAM

S3

EC2

Search [Alt+S]

nik\_accessKeys (1).csv  
99 B • Done

Bookmarks

< >

IAM > Users > nik > Create access key

(?) ?

Step 1  
Access key best practices &  
alternatives

Step 2 - optional  
Set description tag

Step 3  
**Retrieve access keys**

## Retrieve access keys 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

AKIAZZVHHUZXMTJFLJ

\*\*\*\*\* 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

3. On the local machine, the AWS CLI was configured using these credentials with the following command:



```
Microsoft Windows [Version 10.0.26200.7171]
(c) Microsoft Corporation. All rights reserved.

C:\Users\Nikhilesh Sakhare>aws configure
AWS Access Key ID [*****KBJF]: AKIAZZVHHUZMAXT2
AWS Secret Access Key [*****HbFh]: g0FBgWvvr0CnjlPDrseEG16dIxD2lyizq
Default region name [None]:
Default output format [None]:
```

4. With the CLI configured for user nik, a new IAM user named ram was created using:

```
C:\Users\Nikhilesh Sakhare>aws iam create-user --user-name ram
{
  "User": {
    "Path": "/",
    "UserName": "ram",
    "UserId": "AIDAZZVHHUZZMNVJJPYRARE",
    "Arn": "arn:aws:iam::673586841413:user/ram",
    "CreateDate": "2025-11-23T04:56:28+00:00"
  }
}
```

5. To verify, the following command was run to list all IAM users:

```
C:\Users\Nikhilesh Sakhare>aws iam list-users
{
  "Users": [
    {
      "Path": "/",
      "UserName": "nik",
      "UserId": "AIDAZZVHHUZZMM222PA640",
      "Arn": "arn:aws:iam::673586841413:user/nik",
      "CreateDate": "2025-11-23T04:50:33+00:00"
    },
    {
      "Path": "/",
      "UserName": "ram",
      "UserId": "AIDAZZVHHUZZMNVJJPYRARE",
      "Arn": "arn:aws:iam::673586841413:user/ram",
      "CreateDate": "2025-11-23T04:56:28+00:00"
    }
  ]
}
```

6. Next, an IAM group named devops was created using the CLI command

```
C:\Users\Nikhilesh Sakhare>aws iam create-group --group-name devops
{
  "Group": {
    "Path": "/",
    "GroupName": "devops",
    "GroupId": "AGPAZZVHHUZZMHIKJKDNTX",
    "Arn": "arn:aws:iam::673586841413:group/devops",
    "CreateDate": "2025-11-23T04:57:54+00:00"
  }
}
```

7. The groups in the account were then listed with:

```
C:\Users\Nikhilesh Sakhare>aws iam list-groups
{
  "Groups": [
    {
      "Path": "/",
      "GroupName": "devops",
      "GroupId": "AGPAZZVHHUZZMHIKJKDNTX",
      "Arn": "arn:aws:iam::673586841413:group/devops",
      "CreateDate": "2025-11-23T04:57:54+00:00"
    }
  ]
}
```

8. In the browser, the AWS Policy Generator was used to compose an IAM policy that allows all S3 actions on all resources. The following options were selected:

The screenshot shows the AWS Policy Generator web interface. At the top, it says "AWS Policy Generator" and provides a brief description. Below this, "Step 1: Select policy type" shows "IAM Policy" selected in a dropdown menu. "Step 2: Add statement(s)" shows "Effect" set to "Allow", "AWS" set to "Amazon S3", "Actions" set to "All Actions (\*\*\*)", and "Amazon Resource Name (ARN)" set to "All Resources (\*\*\*)". There is a button labeled "Add Statement". "Step 3: Generate policy" has a button labeled "Generate Policy". At the bottom, there is a disclaimer and a copyright notice: "©2025, Amazon Web Services or its affiliates. All rights reserved."

9. The resulting statement was added, and Generate Policy was clicked to produce the JSON.

This screenshot shows the "Statements added (1)" section of the AWS Policy Generator. It displays a table with one statement: Effect: Allow, Action: s3:\*, Resource(s): \*, Condition(s): None, and a Remove button. Below the table, "Step 3: Generate policy" shows the "Generate Policy" button. The same disclaimer and copyright notice are at the bottom.

Effect	Action	Resource(s)	Condition(s)	Remove
Allow	s3:*	*	None	Remove

10. The generated document was saved locally as s3full.json and contains content similar to:

The screenshot shows a code editor with a file named "s3full.json". The JSON content is as follows:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "Statement1",
      "Effect": "Allow",
      "Action": "s3:*",
      "Resource": "*"
    }
  ]
}
```

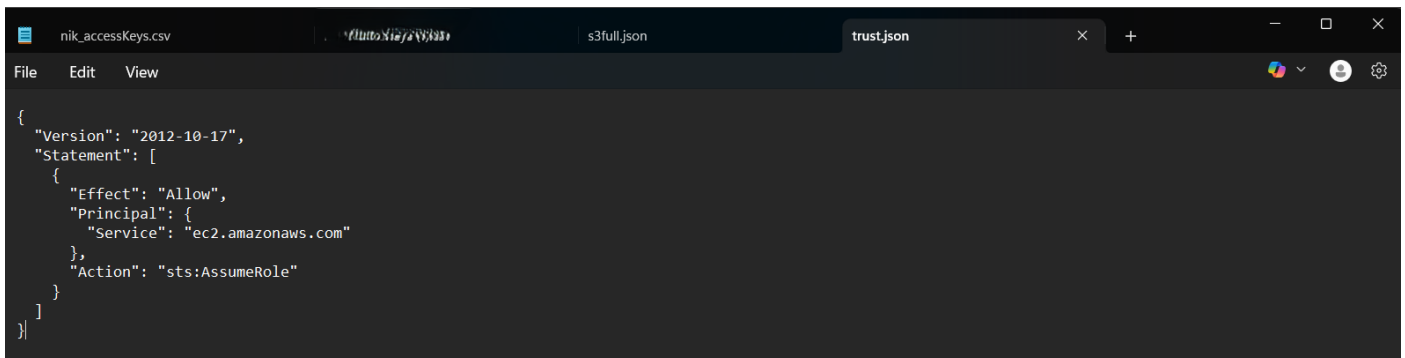
11. Using the saved JSON file, a new customer-managed policy named MyS3Full was created via CLI:

```
C:\Users\Nikhilesh Sakhare>aws iam create-policy --policy-name MyS3Full --policy-document file://Downloads/s3full.json
{
  "Policy": {
    "PolicyName": "MyS3Full",
    "PolicyId": "ANPAZZVHHUZNIGKZFG6U",
    "Arn": "arn:aws:iam::673586841613:policy/MyS3Full",
    "Path": "/",
    "DefaultVersionId": "v1",
    "AttachmentCount": 0,
    "PermissionsBoundaryUsageCount": 0,
    "IsAttachable": true,
    "CreateDate": "2025-11-23T05:26:00+00:00",
    "UpdateDate": "2025-11-23T05:26:00+00:00"
  }
}
```

12. Listing customer policies shows MyS3Full along with other custom policies such as TempS3FullAccess.

**Command:** `aws iam list-policies --scope Local`

13. A separate trust policy JSON file, trust.json, was created locally to allow EC2 to assume a role.



```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "ec2.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

The content grants the ec2.amazonaws.com service permission to assume the role, This trust policy file is shown opened in the editor as trust.json.

14. Using the trust policy, an IAM role named MyEC2Role was created from the CLI with:

```
C:\Users\Nikhilesh Sakhare>aws iam create-role --role-name MyEC2Role --assume-role-policy-document file://Downloads/trust.json
{
  "Role": {
    "Path": "/",
    "RoleName": "MyEC2Role",
    "RoleId": "AR0AZZVHHUZMP67X4MKQX",
    "Arn": "arn:aws:iam::673586841613:role/MyEC2Role",
    "CreateDate": "2025-11-23T05:34:19+00:00",
    "AssumeRolePolicyDocument": {
      "Version": "2012-10-17",
      "Statement": [
        {
          "Effect": "Allow",
          "Principal": {
            "Service": "ec2.amazonaws.com"
          },
          "Action": "sts:AssumeRole"
        }
      ]
    }
  }
}
```

The command output displays the new role, including its ARN and embedded assume-role policy document for the EC2 service.

15. Subsequently, all roles were listed: **[Command: `aws iam list-roles`]** The output will confirm the presence of MyEC2Role along with other service roles in the account.