

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

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

The screenshot shows the AWS IAM service in the AWS Management Console. The left sidebar is titled 'Identity and Access Management (IAM)' and includes sections for 'Access management' (User groups, Roles, Policies, Identity providers, Account settings, Root access management, Temporary delegation requests) and 'Access reports' (Access Analyzer). The main content area is titled 'Users (0) Info' and contains a search bar and a table header with columns: User name, Path, Group, Last activity, MFA, Password age, and Console last sign-in. A message at the top states: 'An IAM user is an identity with long-term credentials that is used to interact with AWS in an account.' A large message at the bottom center says 'No resources to display'. The top right corner shows the account ID: 6735-8610-0000-0000.

The screenshot shows the 'Create user' wizard in the AWS IAM service. The left sidebar shows the steps: Step 1 (selected), Step 2 (Set permissions), and Step 3 (Review and create). The main content area is titled 'Specify user details' and has a section for 'User details' with a 'User name' field containing 'nik'. Below the field is a note: 'The user name can have up to 64 characters. Valid characters: A-Z, a-z, 0-9, and + = , . @ _ - (hyphen)'. There are two optional checkboxes: 'Provide user access to the AWS Management Console - optional' (unchecked) and '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](#)' (unchecked). At the bottom right are 'Cancel' and 'Next' buttons. The top right corner shows the account ID: 6735-8684-0000-0000.

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)

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

Filter by Type	
<input type="text" value="iamfull"/>	All types
<input checked="" type="checkbox"/> Policy name ↗	Type
<input checked="" type="checkbox"/> IAMFullAccess	AWS managed
0	

[Create policy](#)

CloudShell Feedback Console Mobile App © 2025, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences

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	Console password type	Require password reset
nik	None	No

Permissions summary

Name ↗	Type	Used as
IAMFullAccess	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] Account ID: 6735-8684-9368 Global

IAM Users nik

nik Info [Delete](#)

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

Summary

ARN	arn:aws:iam::673586849368:user/nik	Console access	Disabled
Created	November 23, 2025, 10:20 (UTC+05:30)	Last console sign-in	-
		Access key 1	Create access key

Permissions Groups Tags Security credentials Last Accessed

Permissions policies (1)

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

Filter by Type	
<input type="text" value="Search"/>	All types
<input checked="" type="checkbox"/> Policy name ↗	Type
<input checked="" type="checkbox"/> IAMFullAccess	AWS managed
Attached via ↗	
Directly	

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.

Access key best practices & alternatives

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.

Next

Retrieve access keys

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
<input type="text"/> AKIAZZVHUUZMAXT2JFLJ	<input type="text"/> ***** 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]: AKIAZZVHUUZMAXT2JFLJ
AWS Secret Access Key [*****HbFh]: g0FBgWvvr0CnjiPDrseEG16dIx2lyizq
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": "AIDAZZVHHUZMNVJPYRARE",
    "Arn": "arn:aws:iam::673586849113: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": "AIDAZZVHHUZMM222PA640",
      "Arn": "arn:aws:iam::673586849113:user/nik",
      "CreateDate": "2025-11-23T04:50:33+00:00"
    },
    {
      "Path": "/",
      "UserName": "ram",
      "UserId": "AIDAZZVHHUZMNVJPYRARE",
      "Arn": "arn:aws:iam::673586849113: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": "AGPAZZVHHUZMHIKJKDNTX",
    "Arn": "arn:aws:iam::673586849113: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": "AGPAZZVHHUZMHIKJKDNTX",
      "Arn": "arn:aws:iam::673586849113: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:

AWS Policy Generator

The AWS Policy Generator is a tool that enables you to create policies that control access to Amazon Web Services (AWS) products and resources. For more information about creating policies, see [key concepts in Using AWS Identity and Access Management](#).

Step 1: Select policy type

A Policy is a container for permissions. The different types of policies you can create are an [IAM Policy](#), an [S3 Bucket Policy](#), an [SNS Topic Policy](#), a [VPC Endpoint Policy](#), and an [SQS Queue Policy](#).

Type of Policy

IAM Policy

Step 2: Add statement(s)

A statement is the formal description of a single permission. See [a description of elements](#) that you can use in statements.

Effect

Allow

Deny

AWS

All Services (***)

Amazon S3

Use multiple statements to add permissions for more than one service.

Actions

All Actions (***)

Amazon Resource Name (ARN)

All Resources (***)

► Add conditions (optional)

Add Statement

Step 3: Generate policy

A policy is a document (written in the [Access Policy Language](#)) that acts as a container for one or more statements.

Generate Policy

This AWS Policy Generator is provided for informational purposes only, you are still responsible for your use of Amazon Web Services technologies and ensuring that your use is in compliance with all applicable terms and conditions. This AWS Policy Generator is provided **as is** without warranty of any kind, whether express, implied, or statutory. This AWS Policy Generator does not modify the applicable terms and conditions governing your use of Amazon Web Services technologies.

©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.

Statements added (1)				
You added the following statements. Click the button below to Generate a policy.				
Effect	Action	Resource(s)	Condition(s)	Remove
Allow	s3:*	*	None	Remove

Step 3: Generate policy

A policy is a document (written in the [Access Policy Language](#)) that acts as a container for one or more statements.

Generate Policy

This AWS Policy Generator is provided for informational purposes only, you are still responsible for your use of Amazon Web Services technologies and ensuring that your use is in compliance with all applicable terms and conditions. This AWS Policy Generator is provided **as is** without warranty of any kind, whether express, implied, or statutory. This AWS Policy Generator does not modify the applicable terms and conditions governing your use of Amazon Web Services technologies.

©2025, Amazon Web Services or its affiliates. All rights reserved.

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

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

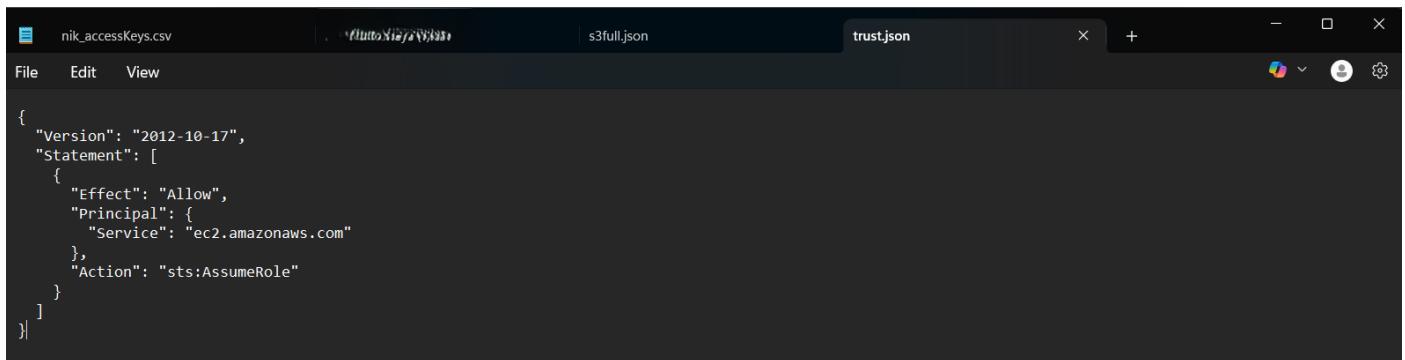
11. Using the saved JSON file, a new customer-managed policy named MyS3Full was created via CLI:

```
C:\Users\Nikhilesh Sakhare>aws iam create-policy --policy-name MyS3Full --policy-document file://Downloads/s3full.json
{
  "Policy": {
    "PolicyName": "MyS3Full",
    "PolicyId": "ANPAZZVHHUZMNLGKZFG6U",
    "Arn": "arn:aws:iam::673586841033: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": "AROAZZVHHUZMP67X4MKQX",
    "Arn": "arn:aws:iam::673586841033: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 confirms the presence of MyEC2Role along with other service roles in the account.