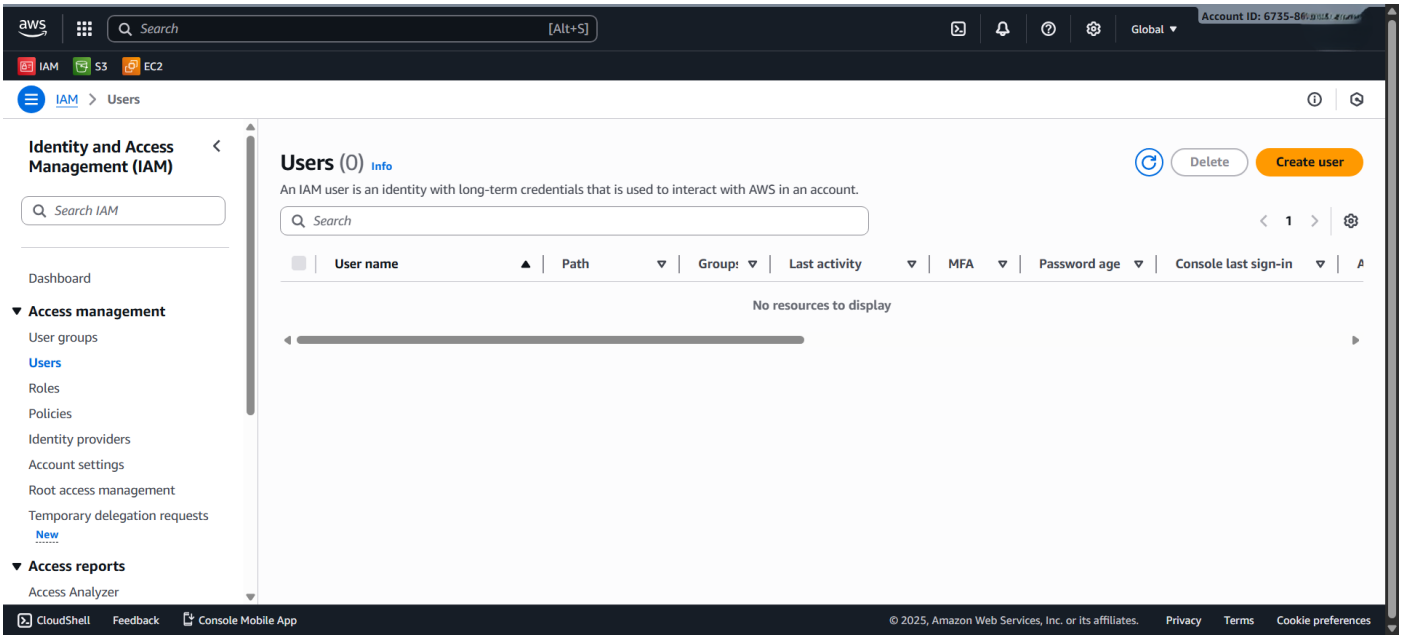
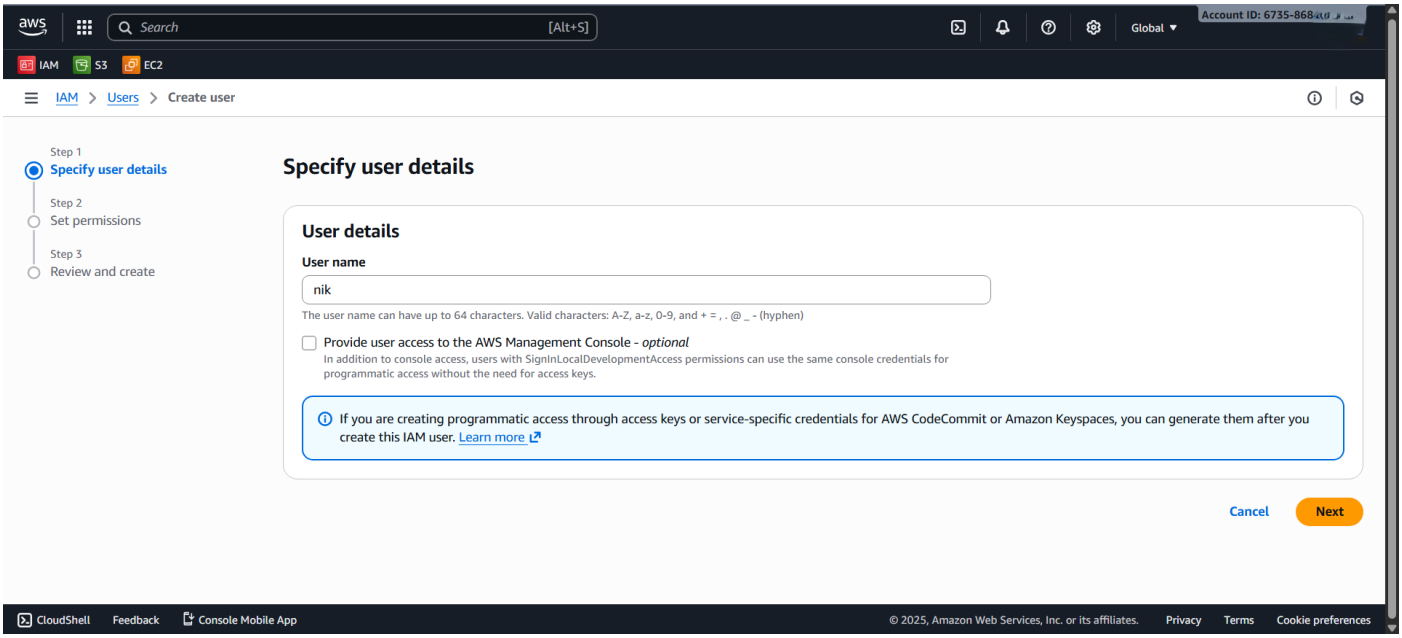
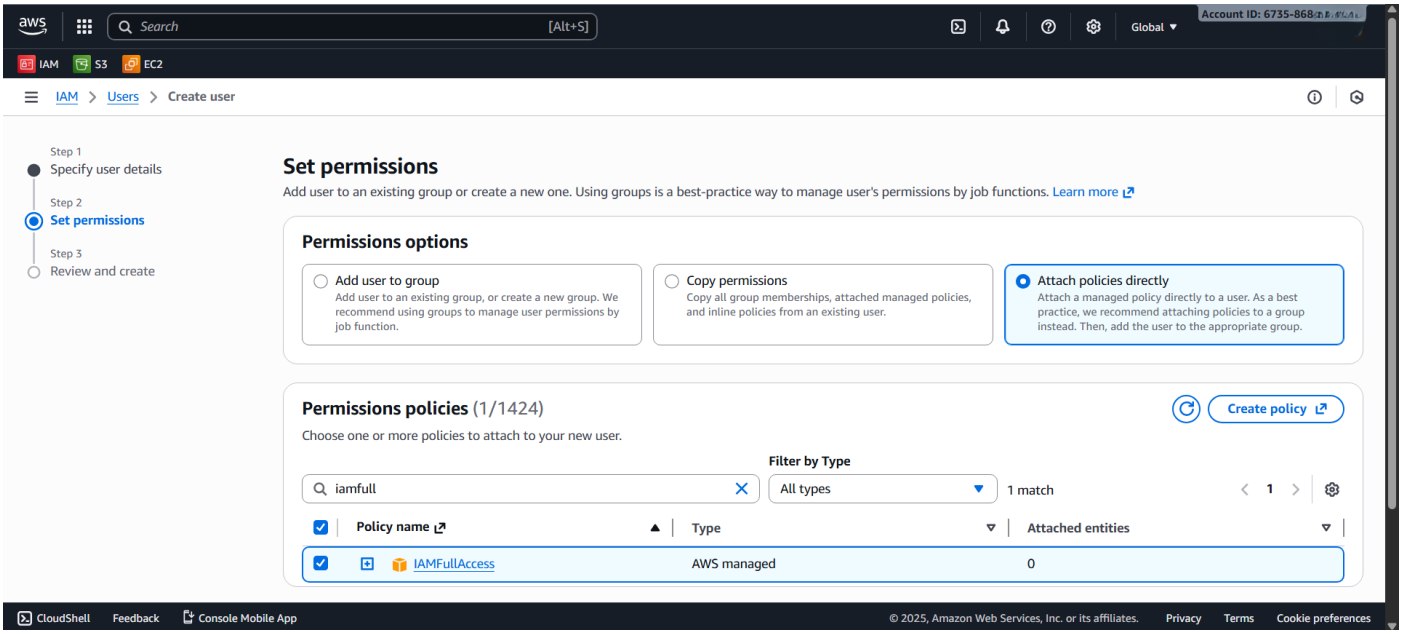
Use CLI to Login a user which have full access IAM Services and create a user, group, policy and a role using AWS CLI

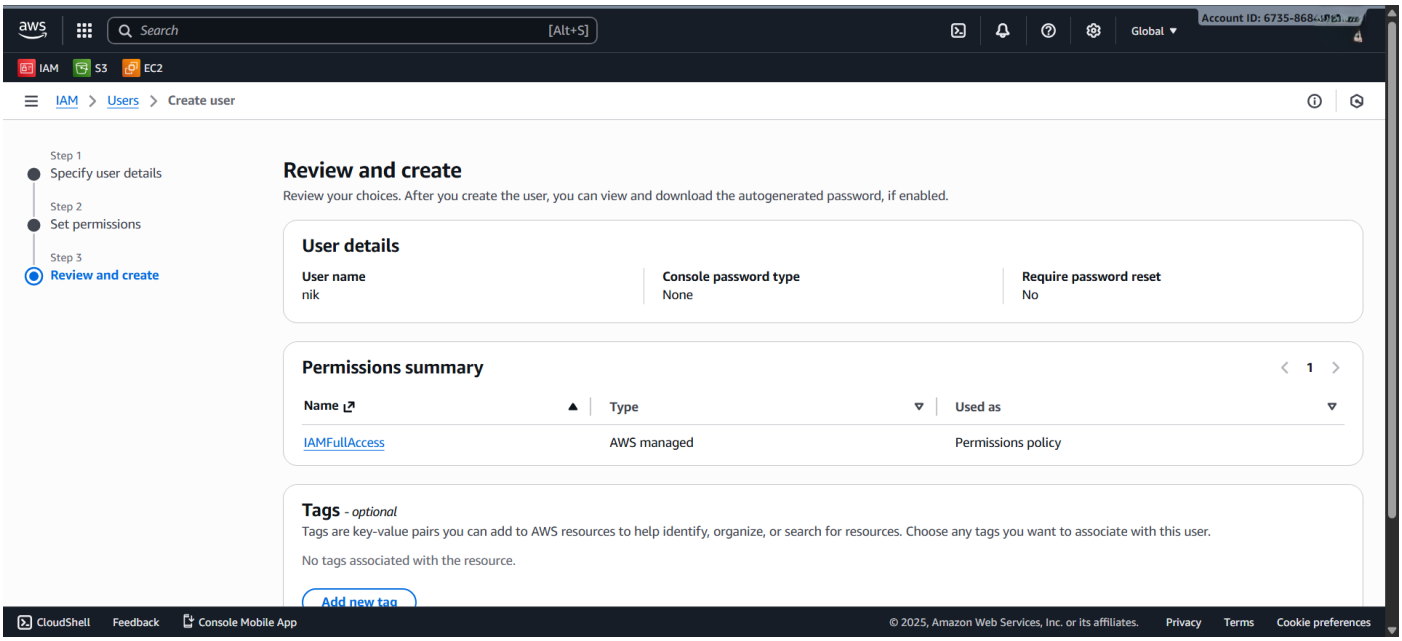
By: Nikhilesh Sakhare

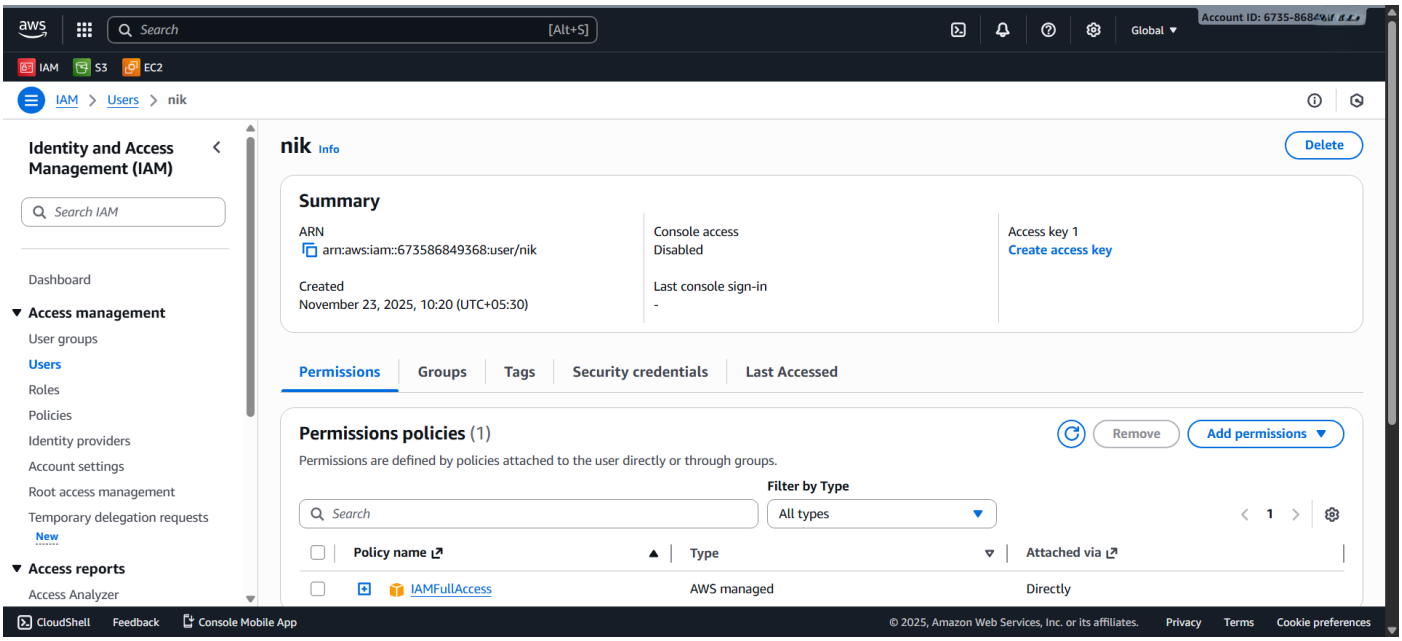
1. First, an IAM user named nik was created in the console and granted the IAMFullAccess managed policy so it can manage all IAM resources.



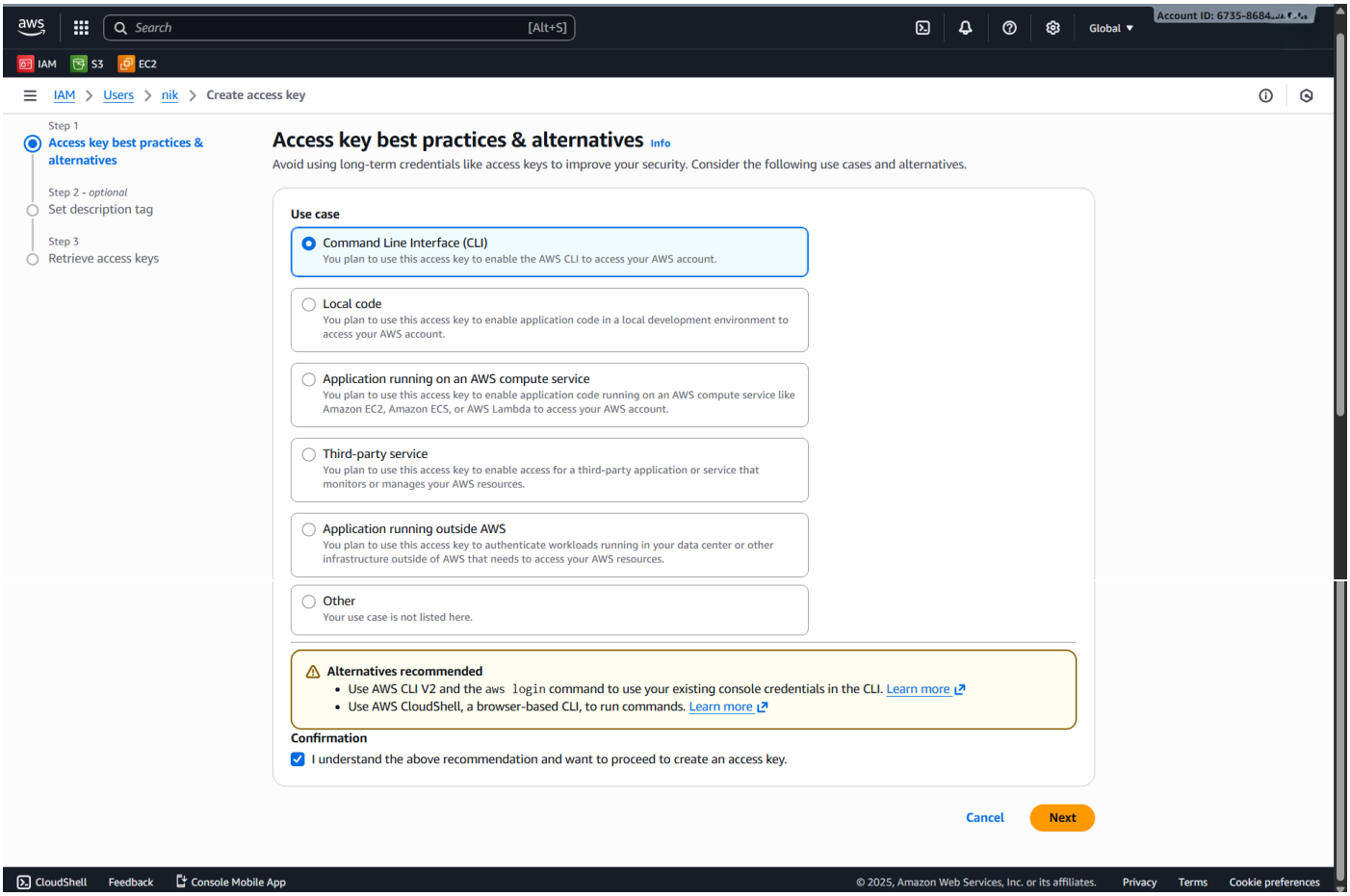


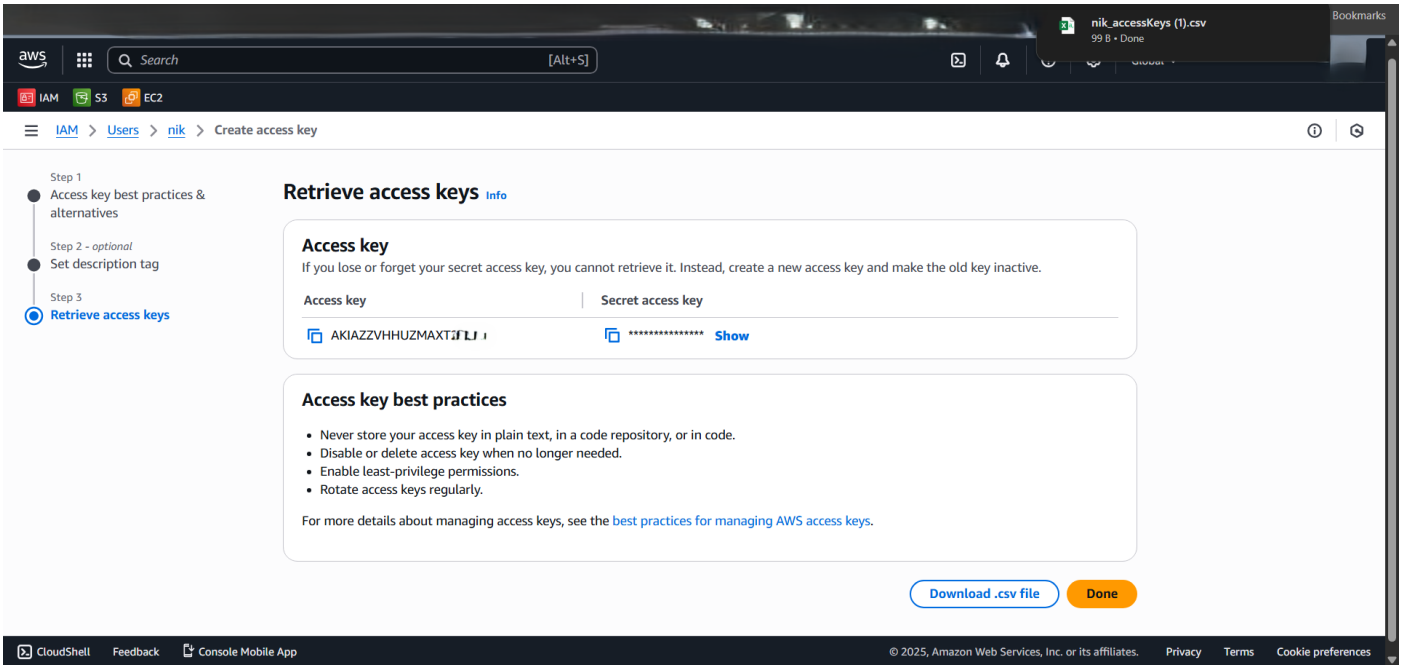

Search [Alt+S]
Global
IAM S3 EC2
IAM > Users > Create user
Step 1
Specify user details
Step 2
Set permissions
Step 3
Review and create
Set permissions
Add user to an existing group or create a new one. Using groups is a best-practice way to manage user's permissions by job functions. Learn more
Permissions 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)
Choose one or more policies to attach to your new user.
Create policy
Filter by Type
iamfull
All types
1 match
Policy name
Type
Attached entities
IAMFullAccess
AWS managed
0
CloudShell
Feedback
Console Mobile App
© 2025, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences


Search [Alt+S]
Global
IAM S3 EC2
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
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

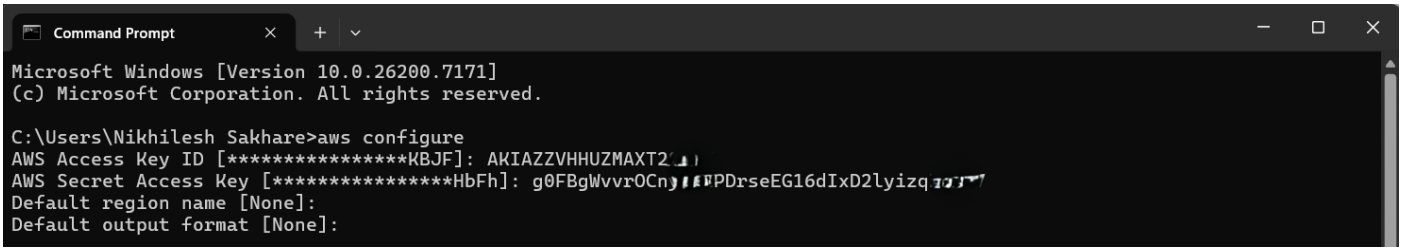

Search [Alt+S]
Global
IAM S3 EC2
IAM > Users > nik
Identity and Access Management (IAM)
Search IAM
Dashboard
Access management
User groups
Users
Roles
Policies
Identity providers
Account settings
Root access management
Temporary delegation requests
New
Access reports
Access Analyzer
nik Info
Delete
Summary
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)
Permissions are defined by policies attached to the user directly or through groups.
Remove
Add permissions
Filter by Type
Search
All types
Policy name
Type
Attached via
IAMFullAccess
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.





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



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::673586849[illegible]: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": "AIDAZZVHHUZMM222PA64O",
            "Arn": "arn:aws:iam::673586841[illegible]:user/nik",
            "CreateDate": "2025-11-23T04:50:33+00:00"
        },
        {
            "Path": "/",
            "UserName": "ram",
            "UserId": "AIDAZZVHHUZMNVJPYRARE",
            "Arn": "arn:aws:iam::673586849[illegible]:user/ram",
            "CreateDate": "2025-11-23T04:56:28+00:00"
        }
    ]
}
```

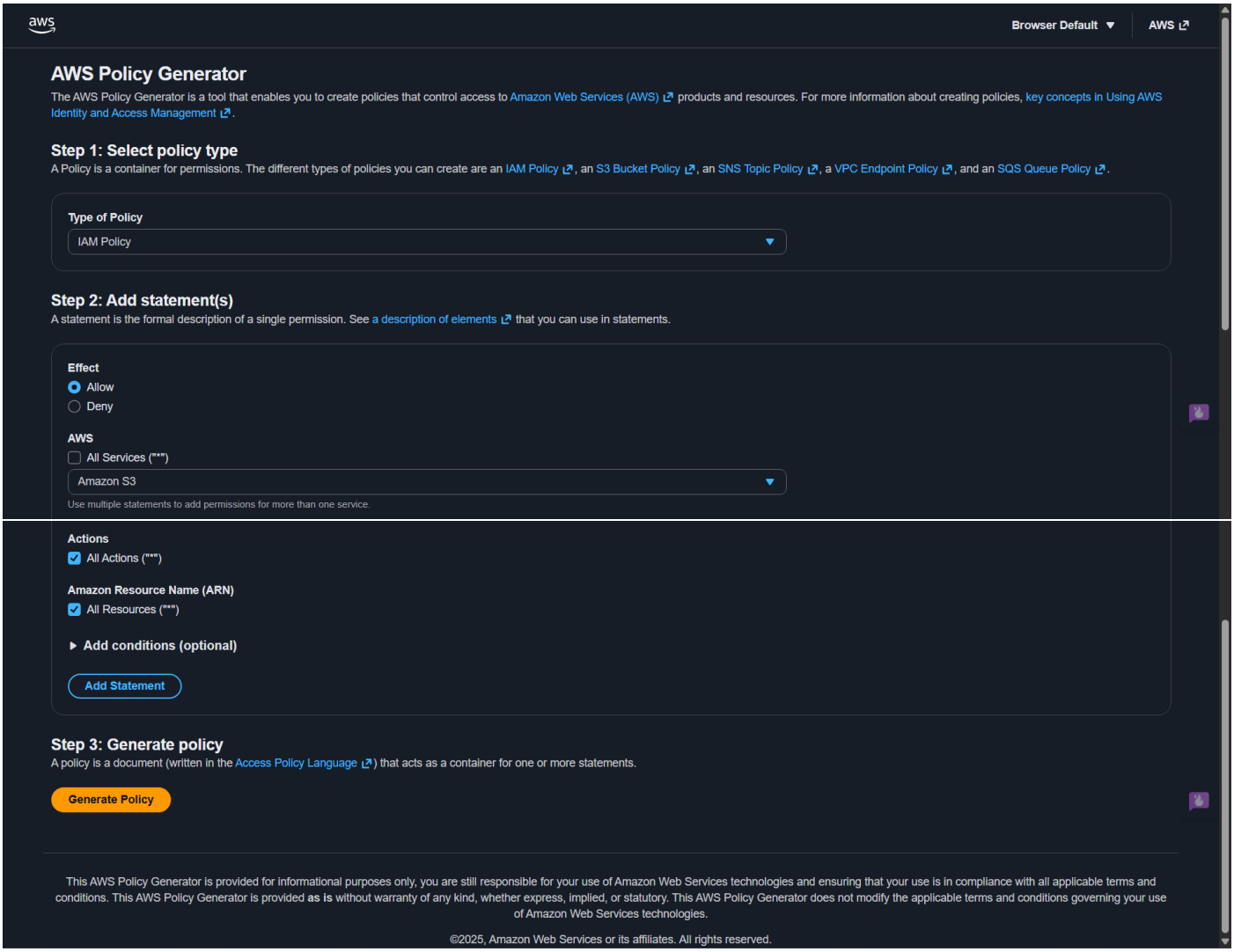
6. Next, an IAM group named devops was created using the CLI command

```
C:\Users\Nikhilesh Sakhare>aws iam create-group --group-name devops
{
    "Group": {
        "Path": "/",
        "GroupName": "devops",
        "GroupId": "AGPAZZVHHUZMHIKJKDNTX",
        "Arn": "arn:aws:iam::673586849[illegible]:group/devops",
        "CreateDate": "2025-11-23T04:57:54+00:00"
    }
}
```

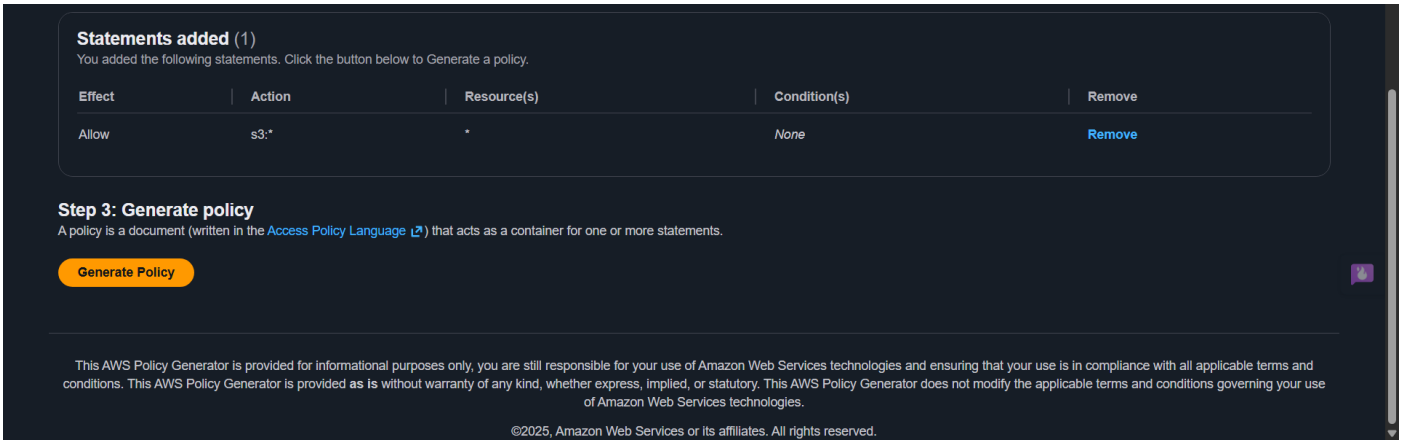
7. The groups in the account were then listed with:

```
C:\Users\Nikhilesh Sakhare>aws iam list-groups
{
    "Groups": [
        {
            "Path": "/",
            "GroupName": "devops",
            "GroupId": "AGPAZZVHHUZMHIKJKDNTX",
            "Arn": "arn:aws:iam::673586849[illegible]:group/devops",
            "CreateDate": "2025-11-23T04:57:54+00:00"
        }
    ]
}
```

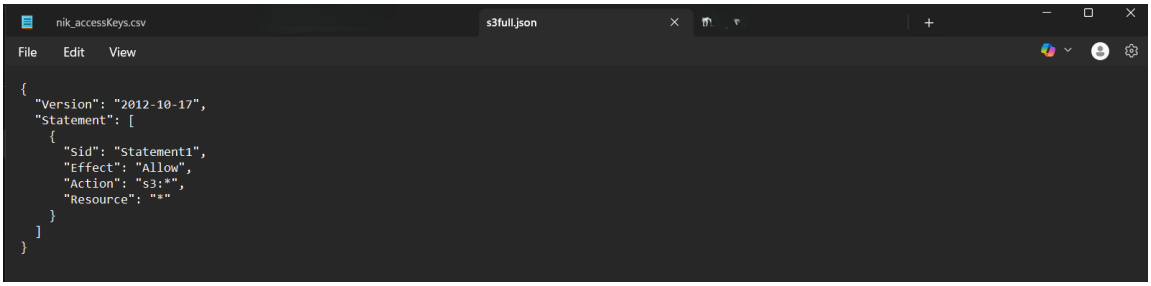
8. In the browser, the AWS Policy Generator was used to compose an IAM policy that allows all S3 actions on all resources. The following options were selected:



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



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

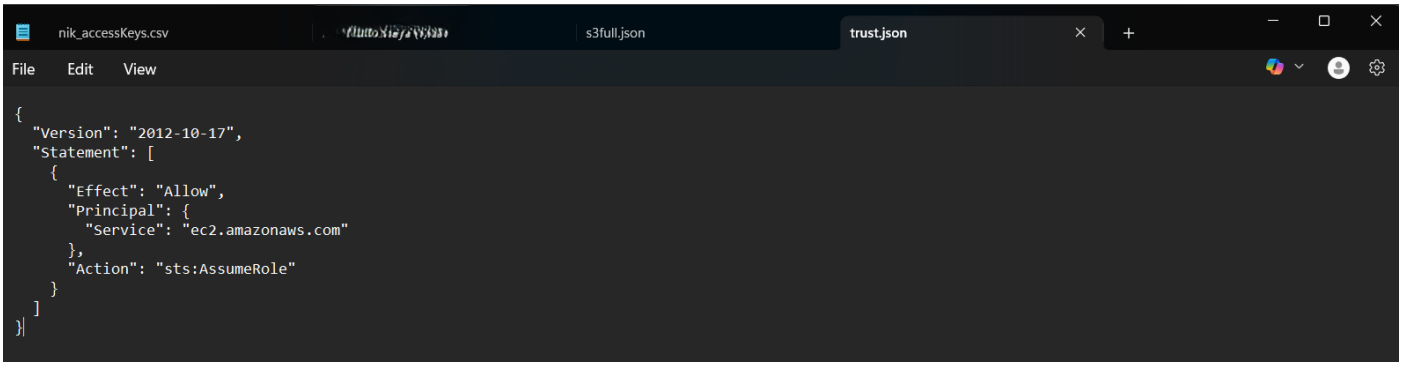


11. Using the saved JSON file, a new customer-managed policy named MyS3Full was created via CLI:

```
C:\Users\Nikhilesh Sakhare>aws iam create-policy --policy-name MyS3Full --policy-document file://Downloads/s3full.json
{
    "Policy": {
        "PolicyName": "MyS3Full",
        "PolicyId": "ANPAZZVHHUZMNIGKZFG6U",
        "Arn": "arn:aws:iam::673586849[illegible]3: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.



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/trus
t.json
{
    "Role": {
        "Path": "/",
        "RoleName": "MyEC2Role",
        "RoleId": "AROAZZVHHUZMP67X4MKQX",
        "Arn": "arn:aws:iam::673586849[illegible]3: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.