

Project Overview

Goal:

Enable secure, automated access to AWS S3 buckets from Linux systems using IAM roles/policies and local permission controls.

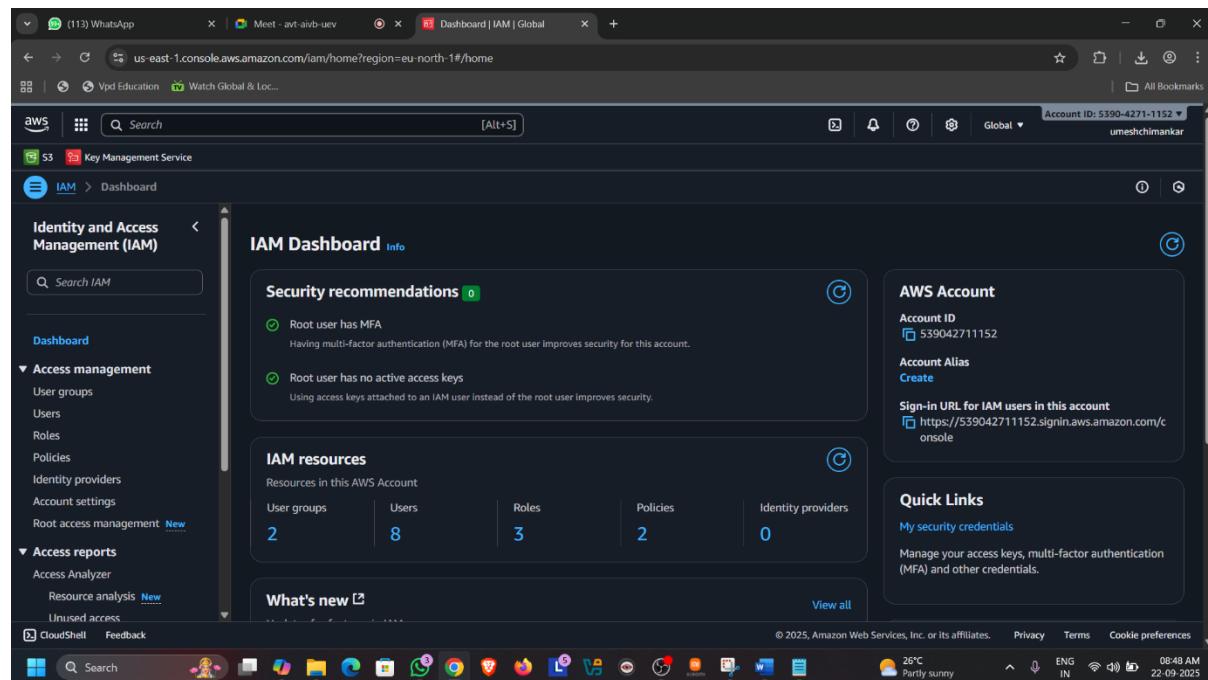
Use Case Examples:

- Centralized logging to S3 from Linux VMs
- Backup scripts pushing data to S3
- Controlled access for specific users/groups on Linux

Key Components

Step 1. IAM Policy Design and S3 Bucket Creation

Goto IAM Service



The screenshot shows the AWS IAM Policies page. The left sidebar includes sections for Identity and Access Management (IAM), Access management, Access reports, and CloudShell. The main content area displays a table titled "Policies (1391) Info" with columns for Policy name, Type, Used as, and Description. The table lists various AWS managed policies such as AccessAnalyzerServiceRole, AdministratorAccess, and AIOpsConsoleAdminPolicy. A search bar and filter options are at the top of the table.

The screenshot shows the "Create policy" wizard, Step 1: Specify permissions. It includes a "Policy editor" section with a "Select a service" dropdown and a "Service" dropdown. Below these are buttons for "Add more permissions" and "Next". The sidebar on the left shows "Step 1: Specify permissions" is selected.

Create a fine-grained IAM policy to control access to specific S3 buckets.

Screenshot of the AWS IAM Policy Editor showing Step 1: Specify permissions.

Specify permissions

Add permissions by selecting services, actions, resources, and conditions. Build permission statements using the JSON editor.

Policy editor

```
1 var {
2   "Version": "2012-10-17",
3   "Statement": [
4     {
5       "Sid": "AllowListAndGet",
6       "Effect": "Allow",
7       "Action": [
8         "s3:ListBucket",
9         "s3:GetObject",
10        "s3:PutObject"
11      ],
12      "Resource": [
13        "arn:aws:s3:::projectno2",
14        "arn:aws:s3:::projectno2/*"
15      ]
16    }
17  ]
18 }
```

Actions

Choose a service

Included

- S3

Available

- AI Operations
- AMP
- API Gateway

CloudShell Feedback © 2025, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences 26°C Partly sunny ENG IN 08:49 AM 22-09-2025

Screenshot of the AWS IAM Policy Editor showing Step 2: Review and create.

Permissions defined in this policy

Permissions defined in this policy document specify which actions are allowed or denied. To define permissions for an IAM identity (user, user group, or role), attach a policy to it.

Service	Access level	Resource	Request condition
S3	Limited: List, Read, Write	Multiple	None

Add tags - optional

Tags are key-value pairs that you can add to AWS resources to help identify, organize, or search for resources.

No tags associated with the resource.

Create policy

Cancel Previous Create policy

CloudShell Feedback © 2025, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences 26°C Partly sunny ENG IN 08:49 AM 22-09-2025

The screenshot shows the AWS IAM Policies page. A green notification bar at the top says "Policy s3accesscontrol created." Below it, a table lists three policies:

Policy name	Type	Used as	Description
s3-list	Customer managed	Permissions policy (1)	-
s3-read-list	Customer managed	None	-
s3accesscontrol	Customer managed	None	-

Now create S3 Bucket

The screenshot shows the AWS Create bucket configuration page. It includes sections for General configuration, Object Ownership, and a summary step.

General configuration

- AWS Region:** Europe (Stockholm) eu-north-1
- Bucket type:** General purpose (selected)
- Bucket name:** project-4-bucket
- Copy settings from existing bucket - optional:** Choose bucket
- Format:** Format: s3://bucket/prefix

Object Ownership

Control ownership of objects written to this bucket from other AWS accounts and the use of access control lists (ACLs). Object ownership determines who can specify access to objects.

Step 2. IAM Role or User Setup

- Attach the above policy to an IAM role (for EC2) or IAM user (for CLI access).
- If using EC2, assign the role to the instance via instance profile.

Create user "testuser" and download their access key

The screenshot shows two side-by-side views of the AWS IAM User Details page for a user named 'testuser'.

Top View (Permissions Tab):

- Summary:** ARN: arn:aws:iam::539042711152:user/testuser, Console access: Disabled, Created: September 22, 2025, 08:54 (UTC+05:30), Last console sign-in: -.
- Permissions:** One policy attached: s3Accesscontrol (Customer managed, Directly).
- Actions:** Delete, Create access key.

Bottom View (Security Credentials Tab):

- Message:** No MFA devices. Assign an MFA device to improve the security of your AWS environment. [Assign MFA device](#).
- 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](#). [Create access key](#).
- API keys for Amazon Bedrock:** 0. Use API keys for Amazon Bedrock to integrate into your library of choice and make API requests programmatically. You can have a maximum of two long-term API keys (active, inactive, or expired) at a time. [Learn more](#). [Generate API Key](#).

Both views include a sidebar with navigation links like Dashboard, Access management, Access reports, and CloudShell.

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 interact with AWS services.

Create access key

Access keys (1)

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](#)

API key name	Created	Expires	Status
AKIAJX3ALTAZYIDTS7NRQ	Now	N/A	Active

Actions **Generate API Key**

Step 3. Linux Configuration

Install AWS CLI & Configure Credentials

Open CloudShell and login through Access key.

CloudShell

eu-north-1

```
~ $ aws configure
AWS Access Key ID [None]: AKIAJ3ALTAZYIDTS7NRQ
AWS Secret Access Key [None]: 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
Default region name [None]:
AWS Secret Access Key [None]: 6Y6dJM+8gSPh35knRy5rHAtQUkRTA6fAhtEo4lGm
Default region name [None]:
Default output format [None]:
```

Step 4. Create Local User & Group

```
~ $ aws configure
AWS Access Key ID [None]: AKIAJ3ALTAZYIDTS7NRQ
AWS Secret Access Key [None]: 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
Default region name [None]:
AWS Secret Access Key [None]: 6Y6dJM+8gSPh35knRy5rHAtQUkRTA6fAhtEo4lGm
Default region name [None]:
Default output format [None]:
~ $ sudo useradd projectno2
~ $ sudo groupadd s3access
~ $ sudo usermod -aG s3access projectno2
~ $ cd /usr/local/bin
bin $ ls
aws      chardetect    cqlsh      dateparser-download flask      isympy    jsonpointer  markdown-it  q          rdfpipe      slugify
aws_completer   chevron    cqish-expansion  dsq1-connect   geomet    jp.py     jsonschema  normalizer  qterm      rdfs2dot  watchmedo
begin-ssm-session cookiecutter  cqish-expansion.init eb       git-remote-codecommit jsondiff  kubectl  __pycache__ rdf2dot    sam
cfn-lint   copilot    csv2rdf      ecs-cli      graphsh    jsonpatch lambda-builders pygmentize rdfgraphisomorphism session-manager-plugin
bin $ sudo touch s3_upload.sh
bin $ ls
aws      chardetect    cqlsh      dateparser-download flask      isympy    jsonpointer  markdown-it  q          rdfpipe      session-manager-plugin
!/bin/bash
aws_completer   chevron    cqish-expansion  dsq1-connect   geomet    jp.py     jsonschema  normalizer  qterm      rdfs2dot  slugify
begin-ssm-session cookiecutter  cqish-expansion.init eb       git-remote-codecommit jsondiff  kubectl  __pycache__ rdf2dot  s3_upload.sh  watchmedo
cfn-lint   copilot    csv2rdf      ecs-cli      graphsh    jsonpatch lambda-builders pygmentize rdfgraphisomorphism sam
wheel
```

```
bin $ sudo vim s3_upload.sh
```

create **s3_upload.sh** file and write below script

```
#!/bin/bash
#/usr/local/bin/s3_upload.sh
BUCKET="project-4.bucket"
FILE="$1"
if [[ ! -f "$FILE" ]]; then echo "File not found!"
exit 1
fi
aws s3 cp "$FILE" "s3://$BUCKET/"
```

Step 5. Set File Permissions

Ensure only authorized users can run the S3 access script.

```
bin $ cd
~ $ sudo chown root:s3access /usr/local/bin/s3_upload.sh
~ $ sudo chmod 750 /usr/local/bin/s3_upload.sh
~ $ sudo s3 ls
sudo: s3: command not found
~ $ sudo aws s3 ls
2025-09-22 03:12:10 projectno2
```

Testing & Validation

- Test with a user in group.
- Validate IAM permissions.
- Monitor CloudTrail for access logs.

```
bin $ cd /usr/local/bin
bin $ ls
aws      chardetect  cqlsh      dateparser-download  flask      isympy    jsonpointer    markdown-it  q      rdfpipe    session-manager-plugin
aws_completer   chevron     cqlsh-expansion  dsql-connect  geopolit  jp-py     jsonschema    normalizer  qterm    rdf2dot    slugify
begin-ssm-session cookiecutter  cqlsh-expansion.init  eb      git-remote-codecommit jsondiff  kubectl    __pycache__  rdf2dot    s3_upload.sh  watchdog
cfn-lint      copilot     csv2rdf      ecs-cli      graphsh  jsonpatch  lambda-builders  pygmentize  rdfgraphisomorphism  sam
bin $ ls -ltr s3_upload.sh
-rwxr-x---. 1 root s3access 180 Sep 22 03:59 s3_upload.sh
bin $
```

Contribution:

Sakshi Jain

Jay Soni

Nikita Binnar

Ankita Shinde

Umesh Chimankar

Kaveri Kanawade