

**Project in AWS
Practice Lab**

Using EC2 Roles and Instance Profiles in AWS

Andra-Diana Popescu

2025

ABOUT THIS LAB

AWS Identity and Access Management (IAM) roles for Amazon Elastic Compute Cloud (EC2) provide the ability to grant instances temporary credentials. These temporary credentials can then be used by hosted applications to access permissions configured within the role. IAM roles eliminate the need for managing credentials, help mitigate long-term security risks and simplify permissions management. Prerequisites for this lab include understanding how to log in to and use the AWS Management Console, EC2 basics (including how to launch an instance), IAM basics (including users, policies, and roles), and how to use the AWS CLI. Once inside the AWS account, make sure you are using us-east-1 (N. Virginia) as the selected region.

LEARNING OBJECTIVES

- Create a Trust Policy and Role Using the AWS CLI
- Create Instance Profile and Attach Role to an EC2 Instance
- Test S3 Permissions via the AWS CLI
- Create an IAM Policy and Role Using the AWS Management Console
- Attach IAM Role to an EC2 Instance Using the AWS Management Console

AWS Documentation about EC2 and IAM:

<https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/iam-roles-for-amazon-ec2.html>

<https://aws.amazon.com/iam/faqs/>

https://docs.aws.amazon.com/IAM/latest/UserGuide/id_roles_use_switch-role-ec2_instance-profiles.html

<https://aws.amazon.com/developer/tools/>

<https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/ec2-instance-metadata.html>

<https://docs.aws.amazon.com/cli/latest/userguide/cli-chap-configure.html#config-settings-and-precedence>

<https://boto3.amazonaws.com/v1/documentation/api/latest/guide/configuration.html>

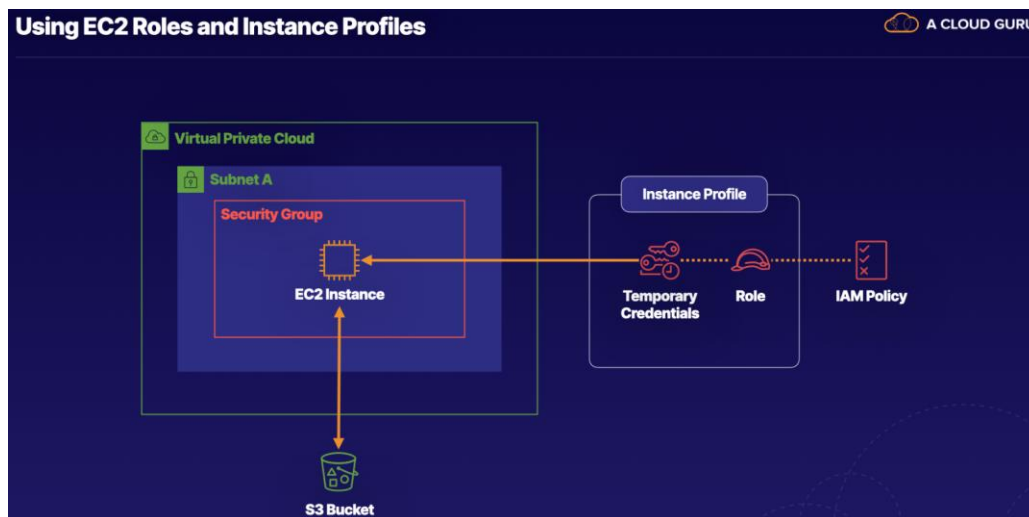
<https://docs.aws.amazon.com/cli/latest/userguide/cli-configure-envvars.html>

Source: <https://learn.acloud.guru/course/certified-solutions-architect-associate/>

Table of Contents

Lab Diagram	4
Log in to your AWS account	5
1. Create a Trust Policy and Role Using the AWS CLI.....	5
1.1. Obtain the labreferences.txt File	5
1.2. Connect to Bastion Host EC2 Instance and Set the AWS CLI Region and Output Type.....	6
1.3. Create IAM Trust Policy for an EC2 Role.....	7
1.4. Create the DEV_ROLE IAM Role	8
1.5. Create an IAM Policy Defining Read-Only Access Permissions to an S3 Bucket.....	9
2. Create Instance Profile and Attach Role to an EC2 Instance.....	11
2.1. Attach Managed Policy to Role	11
2.2. Create the Instance Profile and Add the DEV_ROLE via the AWS CLI.....	11
2.3. Attach the DEV_PROFILE Role to an Instance	12
3. Test S3 Permissions via the AWS CLI.....	13
3.1. Test S3 Permissions	13
4. Create an IAM Policy and Role Using the AWS Management Console.....	15
4.1. Create Policy.....	15
4.2. Create Role.....	18
5. Attach IAM Role to an EC2 Instance Using the AWS Management Console	19
5.1. Attach the Role	19
5.2. Test the Configuration	20

Lab Diagram



Let's look at this scenario. You are responsible for ensuring your applications hosted in Amazon EC2 are able to securely access other AWS services. Credentials need to be rotated regularly to minimize the adverse impact of a security breach. You want to minimize the time it takes to manage these credentials. AWS IAM roles provide the ability to automatically grant instances temporary credentials without the need for manual management. IAM instance profiles provide the mechanism to attach IAM roles to EC2 instances.

So, applications that run on an EC2 instance must include AWS credentials in their AWS API requests. You could have your developers store AWS credentials directly within the EC2 instance and allow applications in that instance to use those credentials, but developers would then have to manage their credentials and ensure they securely pass the credentials to each instance and update each instance when it's time to rotate the credentials. That's a lot of additional work, and there's definitely a better way, which is the focus on this lab.

The lab has pre-provisioned some resources for you in the environment at the start of the lab: a VPC, a subnet, EC2 instances, and S3 buckets. During the activities of this lab, we're going to create some roles and associated permissions with those roles to access the S3 buckets. The roles are entities that define a set of permissions for making AWS service requests. Think of an IAM role for EC2 as what you can do. But you can associate a role directly with an EC2 instance and you need an instance profile to do so.

An instance profile is an entity or a container that's used for connecting an IAM role to an EC2 instance. So, instance profile is like *whoami*. Instance profiles provide temporary credentials, which are rotated automatically. So, if a hacker gets into your server, they get the credentials, but those credentials live for a short period of time. When you create and attach the role to an EC2 instance in the AWS Management Console, the creation and use of the instance profile is actually handled behind the scenes.

So, we'll be creating EC2 roles and instance profiles via the AWS CLI (command line interface). Then, we'll create a role and instance profile and permissions using the AWS Management Console.

Log in to your AWS account



Sign in as IAM user

Account ID (12 digits) or account alias

Type Account ID

IAM user name

Type IAM user name

Password

☐ Remember this account

Sign in

Sign in using root user email

[Forgot password?](#)

AWS Skill Builder

Your new learning center to access 500+ free digital courses

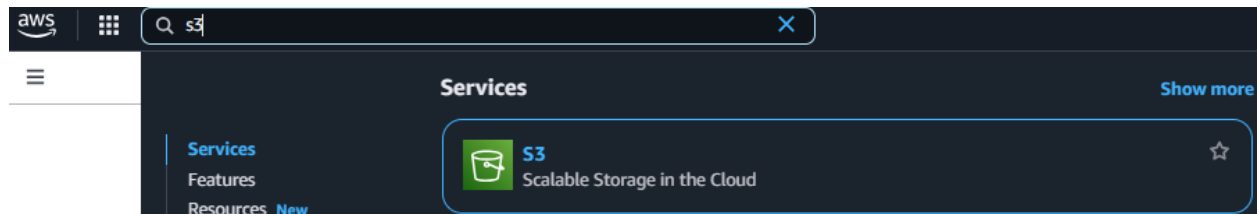
GET STARTED



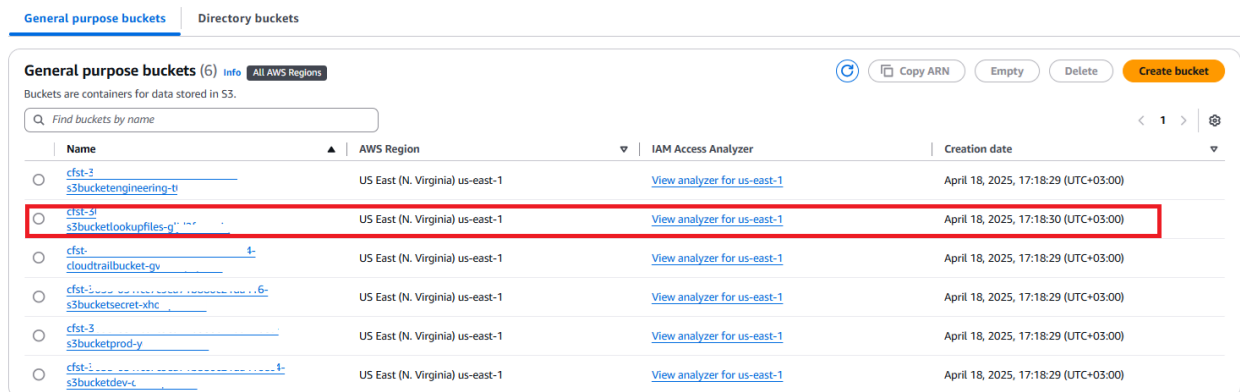
1. Create a Trust Policy and Role Using the AWS CLI

1.1. Obtain the **labreferences.txt** File

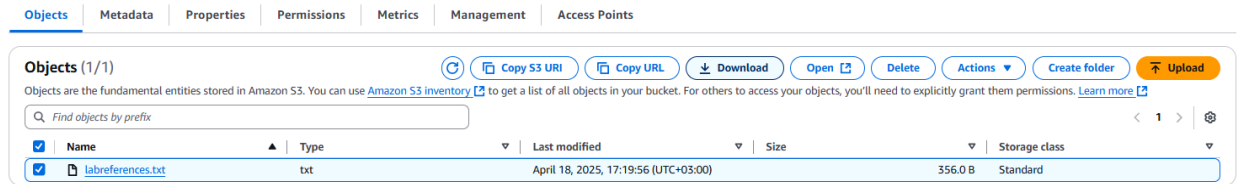
1. Once you are logged in to the AWS Management Console, navigate to S3.



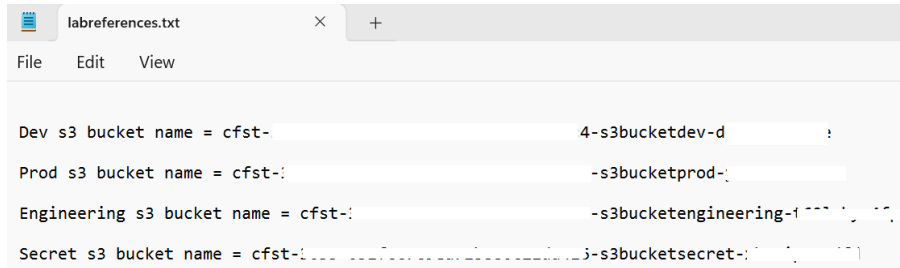
2. From the list of buckets, open the one that contains the text **s3bucketlookupfiles** in the middle of its name.



3. Select the **labreferences.txt** file and **Download**.

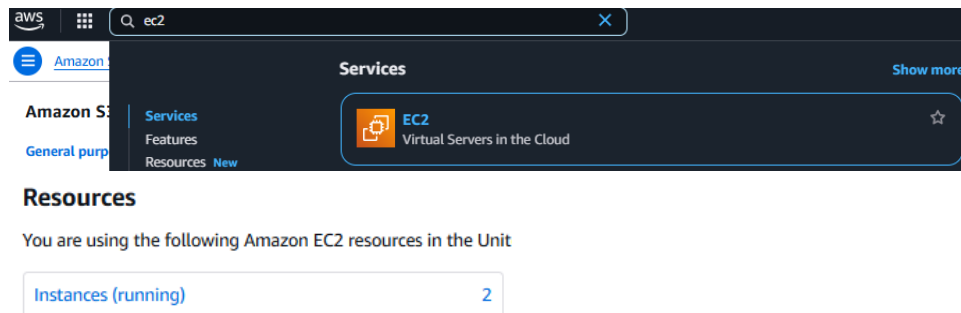


4. Open the *labreferences.txt* file, as we will need to reference it throughout the lab.

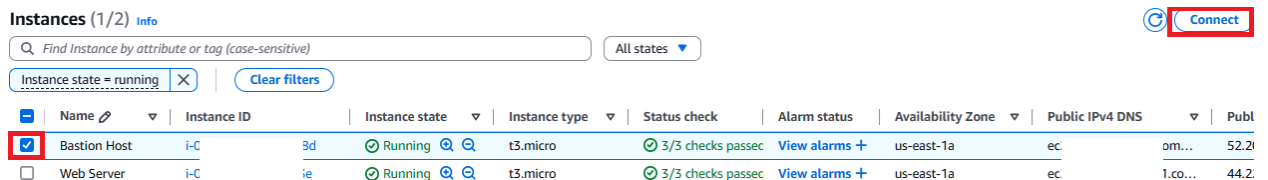


1.2. Connect to Bastion Host EC2 Instance and Set the AWS CLI Region and Output Type

1. Navigate to **EC2** → **Instances**.



2. Select *Bastion Host* and click **Connect**.



3. Use *EC2 Instance Connect* → **Connect**.

Connect to instance [info](#)

Connect to your instance i-
d (Bastion Host) using any of these options

EC2 Instance Connect

Session Manager

SSH client

EC2 serial console

Instance ID
i- (Bastion Host)

Connection Type

☒ Connect using EC2 Instance Connect
Connect using the EC2 Instance Connect browser-based client, with a public IPv4 or IPv6 address.

☐ Connect using EC2 Instance Connect Endpoint
Connect using the EC2 Instance Connect browser-based client, with a private IPv4 address and a VPC endpoint.

☒ Public IPv4 address
i- s i-2

☐ IPv6 address
-

Username

Enter the username defined in the AMI used to launch the instance. If you didn't define a custom username, use the default username, root.

Note: In most cases, the default username, root, is correct. However, read your AMI usage instructions to check if the AMI owner has changed the default AMI username.

Cancel

Connect

4. Switch from *root* to *cloud_user*.

```
root@ip- [~]# sudo su - cloud_user
cloud_user@ip- [~]$
```

5. Run the following command: **aws configure**
6. Press **Enter** twice to leave the *AWS Access Key ID* and *AWS Secret Access Key* blank.
7. Enter **us-east-1** as the default region name.
8. Enter **json** as the default output format.

```
[cloud_user@ip- [~]$ aws configure
AWS Access Key ID [None]:
AWS Secret Access Key [None]:
Default region name [None]: us-east-1
Default output format [None]: json
[cloud_user@ip- [~]$
```

1.3. Create IAM Trust Policy for an EC2 Role

1. Create a file called *trust_policy_ec2.json*: **vi trust_policy_ec2.json**

```
[cloud_user@ip- [~]$ vi trust_policy_ec2.json
```

2. Click “i” for INSERT.
3. Paste in the following content:

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

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

4. Save and quit using “**esc**” and “**:wq**”.
5. You can use “**cat**” to assure that everything was saved.

```
[cloud_user@ip-10-0-1-10 ~]$ cat trust_policy_ec2.json
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {"Service": "ec2.amazonaws.com"},
      "Action": "sts:AssumeRole"
    }
  ]
}
[cloud_user@ip-10-0-1-10 ~]$
```

1.4. Create the **DEV_ROLE** IAM Role

1. Run the following AWS CLI command: **aws iam create-role --role-name DEV_ROLE --assume-role-policy-document file:///trust_policy_ec2.json**

```
[cloud_user@ip-10-0-1-10 ~]$ aws iam create-role --role-name DEV_ROLE --assume-role-policy-document file:///trust_policy_ec2.json
{
  "Role": {
    "Path": "/",
    "RoleName": "DEV_ROLE",
    "RoleId": "ARO1A5HJ49:role/DEV_ROLE",
    "Arn": "arn:aws:iam::2449:role/DEV_ROLE",
    "CreateDate": "2025-04-18T15:27:17+00:00",
    "AssumeRolePolicyDocument": {
      "Version": "2012-10-17",
      "Statement": [
        {
          "Effect": "Allow",
          "Principal": {
            "Service": "ec2.amazonaws.com"
          },
          "Action": "sts:AssumeRole"
        }
      ]
    }
  }
}
[cloud_user@ip-10-0-1-10 ~]$
```

Note: The policy has been created with the trust policy attached. Now we want to grant this role read-only access permissions to one of our S3 buckets. We need to create an IAM policy that defines those permissions.

1.5. Create an IAM Policy Defining Read-Only Access Permissions to an S3 Bucket

1. Create a file called *dev_s3_read_access.json*: **vi dev_s3_read_access.json**

```
[cloud user@ip- ~]$ vi dev_s3_read_access.json
```

2. Click “i” for INSERT.
3. Paste in the following content:

```
{  
  "Version": "2012-10-17",  
  "Statement": [  
    {  
      "Sid": "AllowUserToSeeBucketListInTheConsole",  
      "Action": ["s3:ListAllMyBuckets", "s3:GetBucketLocation"],  
      "Effect": "Allow",  
      "Resource": ["arn:aws:s3:::*"]  
    },  
    {  
      "Effect": "Allow",  
      "Action": [  
        "s3:Get*",  
        "s3:List*"  
      ],  
      "Resource": [  
        "arn:aws:s3:::<DEV_S3_BUCKET_NAME>/*",  
        "arn:aws:s3:::<DEV_S3_BUCKET_NAME>"  
      ]  
    }  
  ]  
}
```

4. Replace the placeholder `<DEV_S3_BUCKET_NAME>` with the bucket name provided in the *labreferences.txt* file:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "AllowUserToSeeBucketListInTheConsole",
      "Action": ["s3:ListAllMyBuckets", "s3:GetBucketLocation"],
      "Effect": "Allow",
      "Resource": ["arn:aws:s3:*"]
    },
    {
      "Effect": "Allow",
      "Action": [
        "s3:Get*",
        "s3:List*"
      ],
      "Resource": [
        "arn:aws:s3:::<DEV_S3_BUCKET_NAME>/*",
        "arn:aws:s3:::<DEV_S3_BUCKET_NAME>"
      ]
    }
  ]
}
```

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "AllowUserToSeeBucketListInTheConsole",
      "Action": ["s3:ListAllMyBuckets", "s3:GetBucketLocation"],
      "Effect": "Allow",
      "Resource": ["arn:aws:s3:*"]
    },
    {
      "Effect": "Allow",
      "Action": [
        "s3:Get*",
        "s3:List*"
      ],
      "Resource": [
        "arn:aws:s3:::cfst-4-s3bucketdev-d-ke/*",
        "arn:aws:s3:::cfst-4-s3bucketdev-d-ke"
      ]
    }
  ]
}
```

Note: You will see 2 resources: one has `/*` and one has none. This policy will allow us to perform GET and LIST operations on this S3 bucket. The `/*` at the end indicates that this applies to the objects within the bucket and without the `/*`, it indicates that it operates on just the bucket itself. So, make sure you have both of them.

5. Save and quit using `esc` and `:wq`.
6. You can use `cat` to assure that everything was saved.

```
[cloud_user@ip- ~]$ cat dev_s3_read_access.json
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "AllowUserToSeeBucketListInTheConsole",
      "Action": ["s3:ListAllMyBuckets", "s3:GetBucketLocation"],
      "Effect": "Allow",
      "Resource": ["arn:aws:s3:*"]
    },
    {
      "Effect": "Allow",
      "Action": [
        "s3:Get*",
        "s3:List*"
      ],
      "Resource": [
        "arn:aws:s3:::cfst-4-s3bucketdev-d-ke/*",
        "arn:aws:s3:::cfst-4-s3bucketdev-d-ke"
      ]
    }
  ]
}
```

7. Create the managed policy called *DevS3ReadAccess*: **aws iam create-policy --policy-name DevS3ReadAccess --policy-document file:///dev_s3_read_access.json**
8. Copy the policy ARN (Amazon Resource Name) in the output and paste it into the *labreferences.txt* file — we'll need it in a minute.

```
[cloud_user@ip-10-0-1-10 ~]$ aws iam create-policy --policy-name DevS3ReadAccess --policy-document file:///dev_s3_read_access.json
{
  "Policy": {
    "PolicyName": "DevS3ReadAccess",
    "PolicyId": "AN1[REDACTED]ZHY",
    "Arn": "arn:aws:iam::2[REDACTED]:policy/DevS3ReadAccess",
    "Path": "/",
    "DefaultVersionId": "v1",
    "AttachmentCount": 0,
    "PermissionsBoundaryUsageCount": 0,
    "IsAttachable": true,
    "CreateDate": "2025-04-18T16:00:09+00:00",
    "UpdateDate": "2025-04-18T16:00:09+00:00"
  }
}
```

2. Create Instance Profile and Attach Role to an EC2 Instance

2.1. Attach Managed Policy to Role

1. Attach the managed policy to the role, replacing `<DevS3ReadAccess_POLICY_ARN>` with the ARN you just copied: **aws iam attach-role-policy --role-name DEV_ROLE --policy-arn "<DevS3ReadAccess_POLICY_ARN>"**
2. Verify the managed policy was attached: **aws iam list-attached-role-policies --role-name DEV_ROLE**

```
[cloud_user@ip-10-0-1-10 ~]$ aws iam attach-role-policy --role-name DEV_ROLE --policy-arn "arn:aws:iam::5[REDACTED]:policy/DevS3ReadAccess"
[cloud_user@ip-10-0-1-10 ~]$ aws iam list-attached-role-policies --role-name DEV_ROLE
{
  "AttachedPolicies": [
    {
      "PolicyName": "DevS3ReadAccess",
      "PolicyArn": "arn:aws:iam::5[REDACTED]:policy/DevS3ReadAccess"
    }
  ]
}
```

2.2. Create the Instance Profile and Add the DEV_ROLE via the AWS CLI

1. Create instance profile named *DEV_PROFILE*: **aws iam create-instance-profile --instance-profile-name DEV_PROFILE**

```
[cloud_user@ip-10-0-1-10 ~]$ aws iam create-instance-profile --instance-profile-name DEV_PROFILE
{
  "InstanceProfile": {
    "Path": "/",
    "InstanceProfileName": "DEV_PROFILE",
    "InstanceProfileId": "AI[REDACTED]NS",
    "Arn": "arn:aws:iam::5[REDACTED]:instance-profile/DEV_PROFILE",
    "CreateDate": "2025-04-18T17:07:15+00:00",
    "Roles": []
  }
}
```

2. Add role to the **DEV_PROFILE** called **DEV_ROLE**: **aws iam add-role-to-instance-profile --instance-profile-name DEV_PROFILE --role-name DEV_ROLE**
3. Verify the configuration: **aws iam get-instance-profile --instance-profile-name DEV_PROFILE**

```
[cloud_user@ip-10.0.0.1 ~]$ aws iam add-role-to-instance-profile --instance-profile-name DEV_PROFILE --role-name DEV_ROLE
[cloud_user@ip-10.0.0.1 ~]$ aws iam get-instance-profile --instance-profile-name DEV_PROFILE
{
  "InstanceProfile": {
    "Path": "/",
    "InstanceProfileName": "DEV_PROFILE",
    "InstanceProfileId": "AIP[REDACTED]NS",
    "Arn": "arn:aws:iam::5[REDACTED]:instance-profile/DEV_PROFILE",
    "CreateDate": "2025-04-18T17:07:15+00:00",
    "Roles": [
      {
        "Path": "/",
        "RoleName": "DEV_ROLE",
        "RoleId": "A[REDACTED]OI",
        "Arn": "arn:aws:iam::5[REDACTED]:role/DEV_ROLE",
        "CreateDate": "2025-04-18T16:47:15+00:00",
        "AssumeRolePolicyDocument": {
          "Version": "2012-10-17",
          "Statement": [
            {
              "Effect": "Allow",
              "Principal": {
                "Service": "ec2.amazonaws.com"
              },
              "Action": "sts:AssumeRole"
            }
          ]
        }
      }
    ]
  },
  "Tags": []
}
```

2.3. Attach the **DEV_PROFILE** Role to an Instance

1. In the AWS console, navigate to **EC2 → Instances**.
2. Copy the **Instance ID** of the instance named **Web Server**.

Instances (1/2) Info

Q Find Instance by attribute or tag (case-sensitive)								All states ▼
Instance state = running X Clear filters								
<input type="checkbox"/>	Name	Instance ID		Instance state	Instance type	Status check	Alarm status	Availability Zone
<input type="checkbox"/>	Bastion Host	i-0[REDACTED]	19	Running	t3.micro	3/3 checks passed	View alarms +	us-east-1a
<input checked="" type="checkbox"/>	Web Server	i-0[REDACTED]	3	Running	t3.micro	3/3 checks passed	View alarms +	us-east-1a

i-0[REDACTED] 3 (Web Server)

Details	Status and alarms	Monitoring	Security	Networking	Storage	Tags
<div>▼ Instance summary Info</div> <div> <div>Instance ID</div> <div> <input type="checkbox"/> i-0[REDACTED] 3 </div> </div> <div> <div>IPv6 address</div> <div>-</div> </div> <div> <div>Public IPv4 address</div> <div> <input type="checkbox"/> [REDACTED] open address </div> </div> <div> <div>Instance state</div> <div>Running</div> </div>						

- In the terminal, attach the *DEV_PROFILE* to an EC2 instance, replacing `<LAB_WEB_SERVER_INSTANCE_ID>` with the *Web Server* instance ID you just copied:
aws ec2 associate-iam-instance-profile --instance-id
`<LAB_WEB_SERVER_INSTANCE_ID>` --iam-instance-profile Name="DEV_PROFILE"

```
[cloud_user@ip-10.0.0.1 ~]$ aws ec2 associate-iam-instance-profile --instance-id i-09f3b3b3 --iam-instance-profile Name="DEV_PROFILE"
{
  "IamInstanceProfileAssociation": {
    "AssociationId": "iip-assoc-04b3b3b3",
    "InstanceId": "i-09f3b3b3",
    "IamInstanceProfile": {
      "Arn": "arn:aws:iam::56464646:instance-profile/DEV_PROFILE",
      "Id": "AIF3b3b3"
    },
    "State": "associating"
  }
}
```

- Verify the configuration (be sure to replace `<LAB_WEB_SERVER_INSTANCE_ID>` with the *Web Server* instance ID again): **aws ec2 describe-instances --instance-ids**
`<LAB_WEB_SERVER_INSTANCE_ID>`

```
cloud_user@ip-10.0.0.1 ~]$ aws ec2 describe-instances --instance-ids i-09f3b3b3
```

- This command's output should show this instance is using *DEV_PROFILE* as an *IamInstanceProfile*. Verify this by locating the *IamInstanceProfile* section in the output and look below to make sure the "ARN" ends in `/DEV_PROFILE`.

```
"IamInstanceProfile": {
  "Arn": "arn:aws:iam::56464646:instance-profile/DEV_PROFILE",
  "Id": "AIF3b3b3"
},
"NetworkInterfaces": [
```

3. Test S3 Permissions via the AWS CLI

3.1. Test S3 Permissions

- Return to the AWS Management Console and navigate to **EC2**.
- On the EC2 dashboard, select *Web Server* and click **Connect**.

The screenshot shows the AWS Management Console 'Instances' page. At the top right, there is a 'Connect' button highlighted with a red box. Below the header, there is a table of instances. The 'Web Server' instance is selected, and its details are shown below the table.

Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Platform
Bastion Host	i-09f3b3b3	Running	t3.micro	3/3 checks passed	View alarms +	us-east-1a	ec2
Web Server	i-09f3b3b3	Running	t3.micro	3/3 checks passed	View alarms +	us-east-1a	ec2

- Use *EC2 Instance Connect* → **Connect**.

Connect to instance [Info](#)

Connect to your instance `i-0f1111111111111111` (Web Server) using any of these options

EC2 Instance Connect

Session Manager

SSH client

EC2 serial console

Instance ID

📄

i-0

🌐

(Web Server)

Connection Type

☒ Connect using EC2 Instance Connect

Connect using the EC2 Instance Connect browser-based client, with a public IPv4 or IPv6 address.

☐ Connect using EC2 Instance Connect Endpoint

Connect using the EC2 Instance Connect browser-based client, with a private IPv4 address and a VPC endpoint.

☒ Public IPv4 address

📄

5

|

☐ IPv6 address

-

Username

Enter the username defined in the AMI used to launch the instance. If you didn't define a custom username, use the default username, root.

🔍 root

✕

📌

Note: In most cases, the default username, root, is correct. However, read your AMI usage instructions to check if the AMI owner has changed the default AMI username.

Cancel

Connect

4. Switch from *root* to *cloud_user*.

```
[root@ip-10-10-10-10 ~]# sudo su - cloud_user
[cloud_user@ip-10-10-10-10 ~]$
```

5. Verify the instance is assuming the DEV_ROLE role: **aws sts get-caller-identity**

```
[cloud_user@ip-██████ ~]$ aws sts get-caller-identity
```

```
"UserId": "AR:aws:iam::█████:role/IAM_ROLE_I09_█████_3",  
"Account": "5█████6",  
"Arn": "arn:aws:sts::5█████6:assumed-role/DEV_ROLE/i-09_█████_3"
```

```
[cloud_user@ip-██████ ~]$
```

Note: We should see DEV_ROLE in the ARN.

6. List the buckets in the account: **aws s3 ls**
7. Copy the entire name (starting with **cfst**) of the bucket with **s3bucketdev** in its name.
8. Attempt to view the files in the **s3bucketdev**- bucket, replacing **<s3bucketdev-123>** with the bucket name you just copied: **aws s3 ls s3://<s3bucketdev-123>**
9. We should see a list of files.

```
[cloud_user@ip-10.0.0.1 ~]$ aws s3 ls
2025-04-18 14:48:08 cfst-3035-fb-1 s3bucketengineering-1
2025-04-18 14:48:08 cfst-3035-fb-2 s3bucketlookupfiles-2
2025-04-18 14:48:08 cfst-3035-fb-3 af1-cloudtrailbucket-v
2025-04-18 14:48:08 cfst-3035-fb-4 af123-s3bucketsecret-1
2025-04-18 14:48:08 cfst-3035-fb-5 af123dc-s3bucketprod-2
2025-04-18 14:48:08 cfst-3035-fb-6 af123dcd-s3bucketdev-n

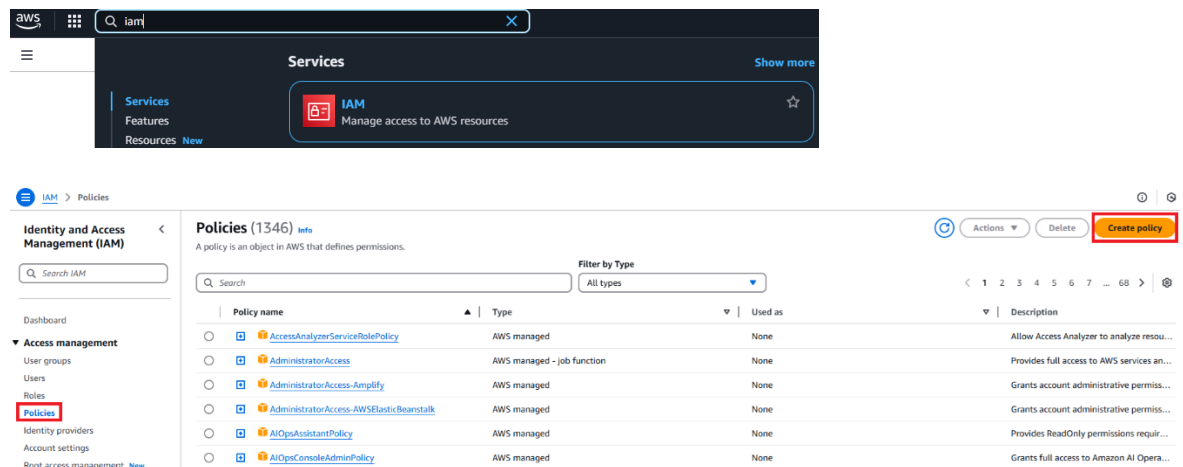
[cloud_user@ip-10.0.0.1 ~]$ aws s3 ls s3://cfst-3035-fb-6/s3bucketdev-n
2025-04-18 14:49:34 41 file1-cfst-3035-fb-6 s3bucketdev-n
2025-04-18 14:49:34 41 file2-cfst-3035-fb-6 s3bucketdev-n
2025-04-18 14:49:34 41 file3-cfst-3035-fb-6 s3bucketdev-n
2025-04-18 14:49:34 41 file4-cfst-3035-fb-6 s3bucketdev-n
2025-04-18 14:49:34 41 file5-cfst-3035-fb-6 s3bucketdev-n
```

4. Create an IAM Policy and Role Using the AWS Management Console

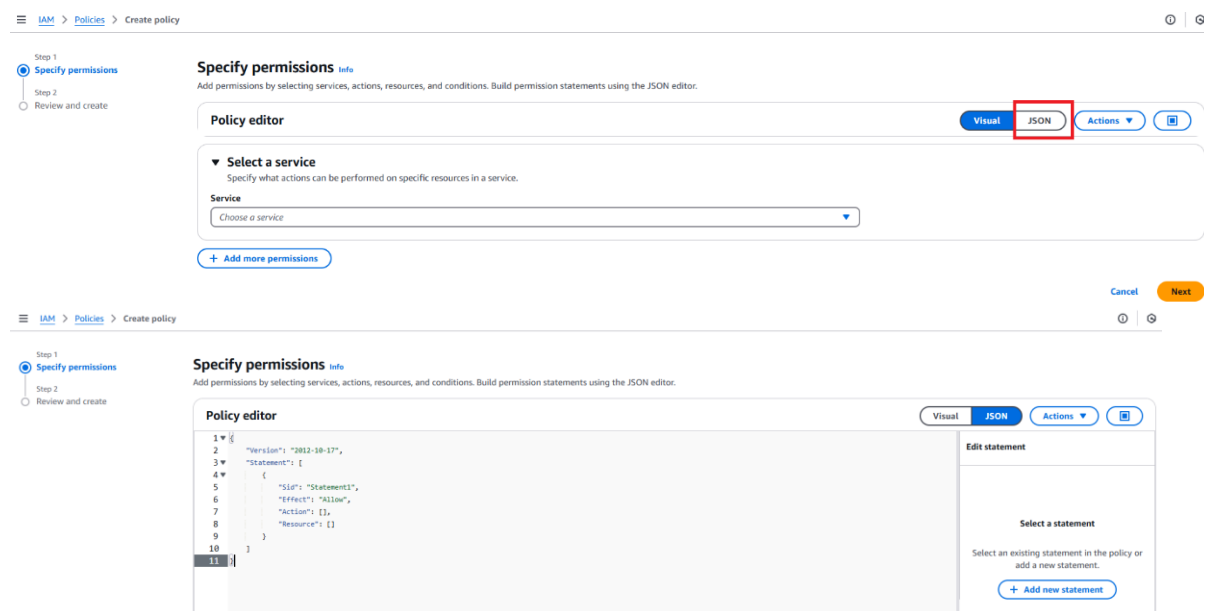
Here we'll cover the creation of a production role, a permission policy, and an instance profile with the AWS Management Console. When you create and attach a role to an EC2 instance using the AWS Management Console, the creation and use of the instance profile is actually handled behind the scenes. From a UI perspective, it looks as we're only dealing with IAM roles. However, AWS automatically runs commands behind the scenes to replicate the steps we did in the CLI. Now, let's do this through AWS Management Console.

4.1. Create Policy

1. In the AWS console, navigate to **IAM** → **Policies**.
2. Click **Create policy**.



3. Click the **JSON** tab.



4. Paste the following text as the policy, replacing **<PROD_S3_BUCKET_NAME>** with the bucket name provided in the *labreferences.txt* file:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "AllowUserToSeeBucketListInTheConsole",
      "Action": ["s3:ListAllMyBuckets", "s3:GetBucketLocation"],
      "Effect": "Allow",
      "Resource": ["arn:aws:s3::*"]
    },
    {
      "Effect": "Allow",
      "Action": [
        "s3:Get*",
        "s3:List*"
      ],
      "Resource": [
        "arn:aws:s3:::<PROD_S3_BUCKET_NAME>/*",
        "arn:aws:s3:::<PROD_S3_BUCKET_NAME>"
      ]
    }
  ]
}
```


- Step 1
Specify permissions
- Step 2
Review and create

Specify permissions [info](#)

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

VisualJSONActions

1

{

2

"Version": "2012-10-17",

3

"Statement": [

4

{

5

"Sid": "AllowUserToSeeBucketListInTheConsole",

6

"Action": ["s3:ListAllMyBuckets", "s3:GetBucketLocation"],

7

"Effect": "Allow",

8

"Resource": ["arn:aws:s3:*"]

9

},

10

{

11

"Effect": "Allow",

12

"Action": [

13

"s3:Get*",

14

"s3:List*"

15

],

16

"Resource": [

17

"arn:aws:s3:::PROD-S3_BUCKET_NAME/*",

18

"arn:aws:s3:::PROD-S3_BUCKET_NAME"

19

]

20

}]

21

}

22

}

Edit statement

Select a statement

Select an existing statement in the policy or add a new statement.

+ Add new statement

Specify permissions [info](#)

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

Policy editor

1

{

2

"Version": "2012-10-17",

3

"Statement": [

4

{

5

"Sid": "AllowUserToSeeBucketListInTheConsole",

6

"Action": ["s3:ListAllMyBuckets", "s3:GetBucketLocation"],

7

"Effect": "Allow",

8

"Resource": ["arn:aws:s3:*"]

9

},

10

{

11

"Effect": "Allow",

12

"Action": [

13

"s3:Get*",

14

"s3:List*"

15

],

16

"Resource": [

17

"arn:aws:s3:::cfst-:s3bucketprod-:h/*",

18

"arn:aws:s3:::cfst-:s3bucketprod-:ha"

19

]

20

}]

21

}

22

}

- Click **Next**.
- Enter **ProdS3ReadAccess** as the policy name.
- Click **Create policy**.

- Step 1
Specify permissions
- Step 2
Review and create

Review and create [info](#)

Review the permissions, specify details, and tags.

Policy details

Policy name

Enter a meaningful name to identify this policy.

ProdS3ReadAccess

Maximum 128 characters. Use alphanumeric and *-._ characters.

Description - optional

Add a short explanation for this policy.

Maximum 1,000 characters. Use alphanumeric and *-._ characters.

This policy defines some actions, resources, or conditions that do not provide permissions. To grant access, policies must have an action that has an applicable resource or condition. For details, choose **Show remaining**. [Learn more](#)

Permissions defined in this policy [info](#)

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

Search

Allow (1 of 439 services)

Show remaining 438 services

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

Add tags - optional [info](#)

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.

Add new tag

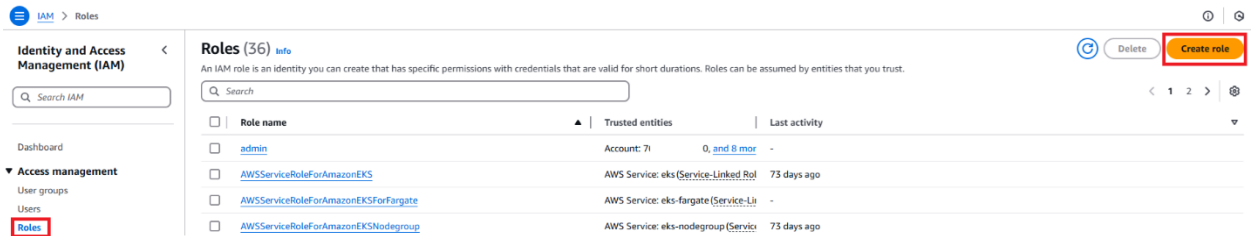
You can add up to 50 more tags.

Cancel Previous **Create policy**

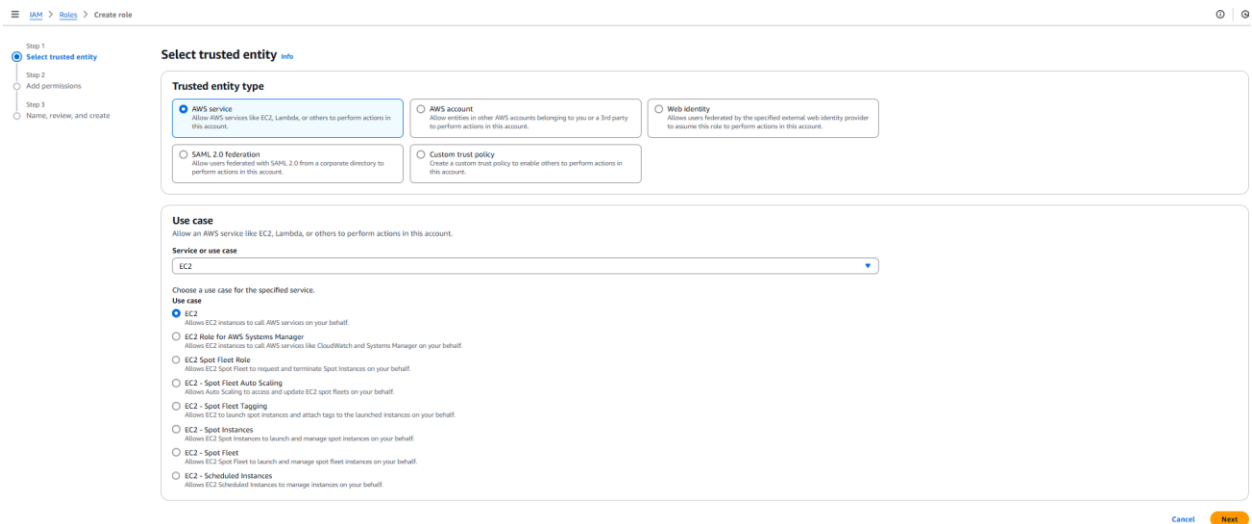
4.2. Create Role

We need to associate this policy to a role.

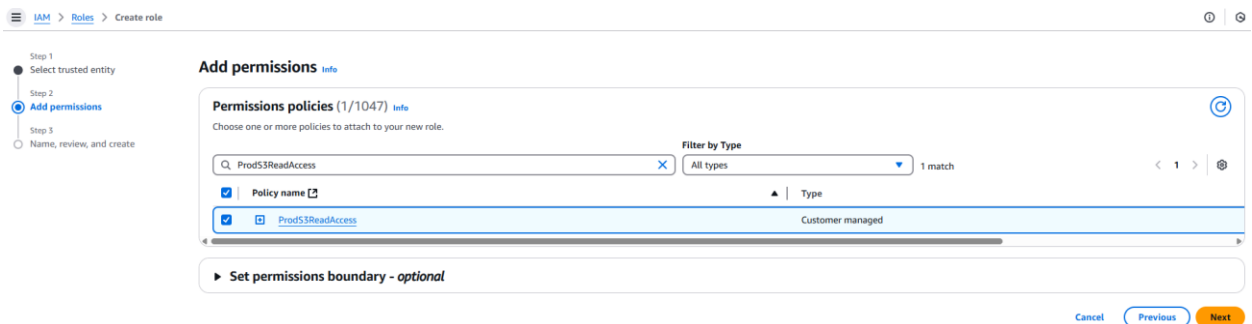
1. Let's navigate to **Roles** → **Create role**.



2. Under *Choose a use case*, select **EC2**.
3. Click **Next**.



4. In the *Filter policies* search box, enter **ProdS3ReadAccess**.
5. Click the checkbox to select **ProdS3ReadAccess**.
6. Click **Next**.



7. Give it a **Role name** of **PROD_ROLE**.
8. Click **Create role**.

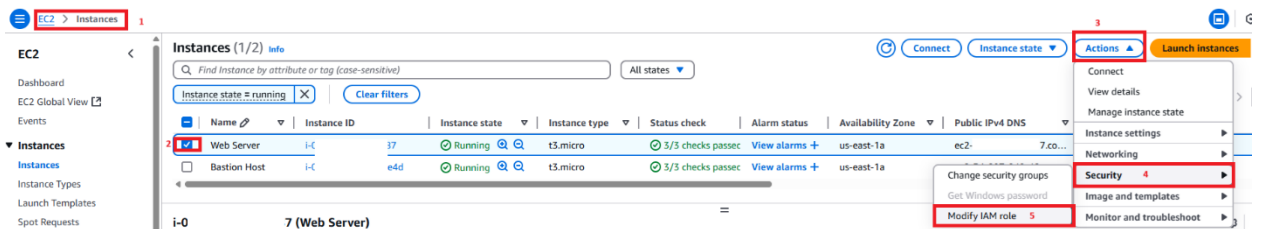
[illegible]

- This automatically allows the EC2 instance to assume this role and it created an instance profile for us.

5. Attach IAM Role to an EC2 Instance Using the AWS Management Console

5.1. Attach the Role

1. Navigate to **EC2 → Instances**.
2. Select the **Web Server** instance. The current IAM role is DEV_ROLE.
3. Click **Actions → Security → Modify IAM role**.



4. In the *IAM role* dropdown, select **PROD_ROLE**.

5. Click **Update IAM role**.

Modify IAM role [Info](#)
Attach an IAM role to your instance.

Instance ID
i-0... 7 (Web Server)

IAM role
Select an IAM role to attach to your instance or create a new role if you haven't created any. The role you select replaces any roles that are currently attached to your instance.

Choose IAM role

No IAM Role
Choose this option to detach an IAM role

cfst-
arn:aws:iam::...:instance-profile/cfst-... InstanceProfileBastionHost-... InstanceProfileBastionHost-...

PROD_ROLE
arn:aws:iam::...:instance-profile/PROD_ROLE

Cancel **Update IAM role**

5.2. Test the Configuration

1. Select **Web Server** → **Connect**.

Instances (1/2) [Info](#)

Last updated 3 minutes ago **Connect**

Find Instance by attribute or tag (case-sensitive) All states

	Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone
<input checked="" type="checkbox"/>	Web Server	i-0...	Running	t3.micro	3/3 checks passed	View alarms +	us-east-1a
<input type="checkbox"/>	Bastion Host	i-0...	Running	t3.micro	3/3 checks passed	View alarms +	us-east-1a

2. Use **EC2 Instance Connect** → **Connect**.

Connect to instance [Info](#)
Connect to your instance i-0... 7 (Web Server) using any of these options

EC2 Instance Connect Session Manager SSH client EC2 serial console

Instance ID
i-0... 7 (Web Server)

Connection Type

☒ Connect using EC2 Instance Connect
Connect using the EC2 Instance Connect browser-based client, with a public IPv4 or IPv6 address.

☐ Connect using EC2 Instance Connect Endpoint
Connect using the EC2 Instance Connect browser-based client, with a private IPv4 address and a VPC endpoint.

☒ Public IPv4 address
5 7

☐ IPv6 address

Username
Enter the username defined in the AMI used to launch the instance. If you didn't define a custom username, use the default username, root.

root

Note: In most cases, the default username, root, is correct. However, read your AMI usage instructions to check if the AMI owner has changed the default AMI username.

Cancel **Connect**

3. Determine the identity currently being used: **aws sts get-caller-identity**

4. This time, we should see **PROD_ROLE** in the ARN.

```
[cloud_user@ip-... ~]$ aws sts get-caller-identity
{
  "UserId": "AR...V:i-0e...",
  "Account": "...",
  "Arn": "arn:aws:sts::...:assumed-role/PROD_ROLE/i-0..."
}
```

5. List the buckets: **aws s3 ls**

6. Copy the entire name (starting with **cfst**) of the bucket with **s3bucketprod** in its name.

8. It should list the files.

```
[cloud_user@ip-10.0.0.1 ~]$ aws s3 ls s3://cfat-3035-[REDACTED]-s3bucketprod-i-[REDACTED].cha
2025-04-19 09:13:38 41 file1-cfat-3035-[REDACTED]-s3bucketprod-[REDACTED].cha
2025-04-19 09:13:38 41 file2-cfat-3035-[REDACTED]-s3bucketprod-[REDACTED].cha
2025-04-19 09:13:38 41 file3-cfat-3035-[REDACTED]-s3bucketprod-[REDACTED].cha
2025-04-19 09:13:38 41 file4-cfat-3035-[REDACTED]-s3bucketprod-[REDACTED].cha
2025-04-19 09:13:38 41 file5-cfat-3035-[REDACTED]-s3bucketprod-[REDACTED].cha
[cloud_user@ip-10.0.0.1 ~]$
```

11. This time, our access will be denied — which means our configuration is properly set up, because we haven't granted access to this bucket in our bucket policy.

```
[cloud user@ip-10-0-1-1 ~]$ aws s3 ls s3://cfst-1234567890-s3bucketsecret-1234567890
An error occurred (AccessDenied) when calling the ListObjectsV2 operation: User: arn:aws:sts::123456789012:assumed-role/PROD_ROLE/i-1234567890 is not authorized to perform: s3:ListBucket on resource: "arn:aws:s3:::cfst-1234567890-s3bucketsecret-1234567890" because no identity-based policy allows the s3:ListBucket action
[cloud user@ip-10-0-1-1 ~]$
```