Tutorial:
Configure S3 access with an instance profile

https://docs.databricks.com/en/connect/storage/tutorial-s3-instance-profile.html

This tutorial walks you through how to create an instance profile with read, write, update, and delete permissions on a single S3 bucket. You can grant privileges for multiple buckets using a single IAM role and instance profile. It is also possible to use instance profiles to grant only read and list permissions on S3.

Administrators configure IAM roles in AWS, link them to a Databricks workspace, and grant access to privileged users to associate instance profiles with compute. All users that have access to compute resources with an instance profile attached to it gain the privileges granted by the instance profile.

Before you begin

This tutorial is designed for workspace administrators. You must have sufficient privileges in the AWS account containing your Databricks workspace, and be a Databricks workspace administrator.

This tutorial assumes the following existing permissions and assets:

- Privileges to edit the IAM role used to deploy the Databricks workspace.
- Privileges to create new IAM roles in AWS.
- Privileges to edit permissions on an S3 bucket.

Step 1: Create an instance profile using the AWS console

1. In the AWS console, go to the **IAM** service.
2. Click the **Roles** tab in the sidebar.
3. Click **Create role**.

a. Under **Trusted entity type**, select **AWS service**.

b. Under **Use case**, select **EC2**.

c. Click **Next**.

d. At the bottom of the page, click **Next**.

e. In the **Role name** field, type a role name.

f. Click **Create role**.

4. In the role list, click the role.

5. Add an inline policy to the role. This policy grants access to the S3 bucket.

a. In the Permissions tab, click **Add permissions > Create inline policy**.

b. Click the **JSON** tab.

c. Copy this policy and set `<s3-bucket-name>` to the name of your bucket.

Copy

JSON

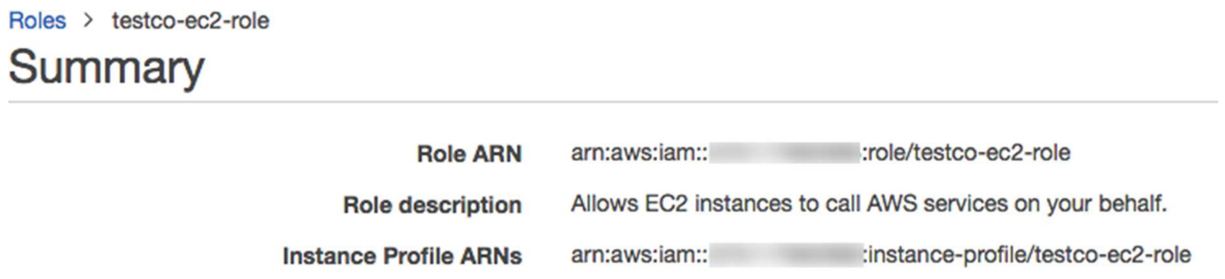
```json
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "s3:ListBucket"
      ],
     "Resource": [
        "arn:aws:s3:::<s3-bucket-name>"
      ]
    },
    {
      "Effect": "Allow",
      "Action": [
        "s3:PutObject",
        "s3:GetObject",
        "s3:DeleteObject",
        "s3:PutObjectAcl"
      ],
      "Resource": [
         "arn:aws:s3:::<s3-bucket-name>/*"
      ]
    }
  ]
}
```

d. Click **Review policy**.

e. In the **Name** field, type a policy name.

f. Click **Create policy**.

6. In the role summary, copy the **Role ARN**.



Roles > testco-ec2-role

Summary

Role ARN	arn:aws:iam::[redacted]:role/testco-ec2-role
Role description	Allows EC2 instances to call AWS services on your behalf.
Instance Profile ARNs	arn:aws:iam::[redacted]:instance-profile/testco-ec2-role

If you intend to enable encryption for the S3 bucket, you must add the IAM role as a **Key User** for the KMS key provided in the configuration. See Configure encryption for S3 with KMS.

Step 2: Enable the policy to work with serverless resources

This step ensures that your instance profile also works for configuring SQL warehouses with instance profiles. See Enable data access configuration.

1. In the role list, click your instance profile.

2. Select the **Trust Relationships** tab.

3. Click **Edit Trust Policy**.

4. Within the existing `Statement` array, append the following JSON block to the end of the existing trust policy. Ensure that you don't overwrite the existing policy.

Copy

JSON

```json
{
"Effect": "Allow",
"Principal": {
  "AWS": [
    "arn:aws:iam::790110701330:role/serverless-customer-resource-role"
  ]
},
"Action": "sts:AssumeRole",
"Condition": {
  "StringEquals": {
      "sts:ExternalId": [
        "databricks-serverless-<YOUR-WORKSPACE-ID1>",
        "databricks-serverless-<YOUR-WORKSPACE-ID2>"
      ]
      }
    }
  }
```

The only thing you need to change in the statement is the workspace IDs. Replace the `YOUR_WORKSPACE-ID`s with one or more Databricks workspace IDs for the workspaces that will use this role.

To get your workspace ID, check the URL when you're using your workspace. For example, in `https://<databricks-instance>/?o=6280049833385130`, the number after `o=` is the workspace ID.

Do **not** edit the principal of the policy. The `Principal.AWS` field must continue to have the value `arn:aws:iam::790110701330:role/serverless-customer-resource-role`. This references a serverless compute role managed by Databricks.

5. Click **Review policy**.
6. Click **Save changes**.

Step 3: Create the bucket policy

At a minimum, the S3 policy must include the `ListBucket` and `GetObject` actions, which provide read-only access to a bucket. Delta Lake uses `DeleteObject` and `PutObject` permissions during regular operations. The permissions in the example policy below are the recommended defaults for clusters that read and write data.

S3 buckets have universally unique names and do not require an account ID for universal identification. If you choose to link an S3 bucket to an IAM role and Databricks workspace in a different AWS account, you must specify the account ID when configuring your S3 bucket policy.

1. Go to your S3 console. From the **Buckets** list, select the bucket for which you want to create a policy.
2. Click **Permissions**.

3. Under **Bucket policy**, click **Edit**.

4. Paste in a policy. A sample cross-account bucket IAM policy could be the following, replacing `<aws-account-id-databricks>` with the AWS account ID where the Databricks environment is deployed, `<iam-role-for-s3-access>` with the instance profile role, and `<s3-bucket-name>` with the bucket name.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "Example permissions",
      "Effect": "Allow",
      "Principal": {
        "AWS": "arn:aws:iam::<aws-account-id-databricks>:role/<iam-role-for-s3-
access>"
      },
      "Action": [
        "s3:GetBucketLocation",
        "s3:ListBucket"
      ],
      "Resource": "arn:aws:s3:::<s3-bucket-name>"
    },
    {
      "Effect": "Allow",
      "Principal": {
        "AWS": "arn:aws:iam::<aws-account-id-databricks>:role/<iam-role-for-s3-
access>"
      },
      "Action": [
        "s3:PutObject",
        "s3:GetObject",
        "s3:DeleteObject",
        "s3:PutObjectAcl"
      ],
      "Resource": "arn:aws:s3:::<s3-bucket-name>/*"
    }
  ]
}
```

5. Click **Save**.

Step 4: Locate the IAM role that created the Databricks deployment

If you don't know which IAM role created the Databricks deployment, do the following:

1. As an account admin, log in to the account console.

2. Go to **Workspaces** and click your workspace name.

3. In the **Credentials** box, note the role name at the end of the Role ARN.

For example, in the Role ARN `arn:aws:iam::123456789123:role/finance-prod`, the role name is `finance-prod`.

Step 5: Add the S3 IAM role to the EC2 policy

1. In the AWS console, go to the **IAM** service.
2. Click the **Roles** tab in the sidebar.
3. Click the role that created the Databricks deployment.
4. On the **Permissions** tab, click the policy.
5. Click **Edit Policy**.
6. Append the following block to the end of the `Statement` array. Ensure that you don't overwrite any of the existing policy. Replace `<iam-role-for-s3-access>` with the role you created in Tutorial: Configure S3 access with an instance profile:

```
{
  "Effect": "Allow",
  "Action": "iam:PassRole",
  "Resource": "arn:aws:iam::<aws-account-id-databricks>:role/<iam-role-for-s3-
access>"
}
```

7. Click **Review policy**.
8. Click **Save changes**.

Step 6: Add the instance profile to Databricks

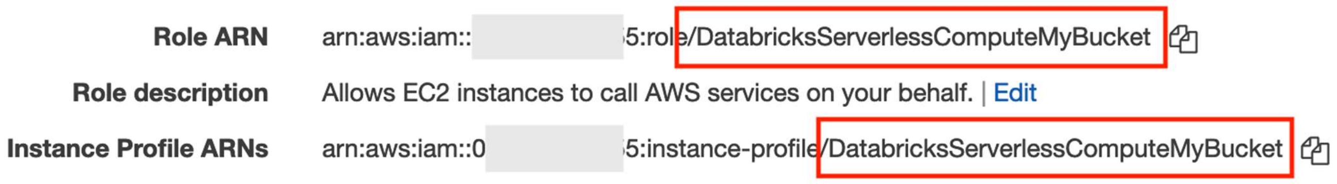
1. As a workspace admin, go to the settings page.
2. Click the **Security** tab.
3. Click **Manage** next to **Instance profiles**.
4. Click **Add Instance Profile**.
5. Paste your instance profile ARN into the **Instance profile ARN** field. If you don't have the ARN, see Tutorial: Configure S3 access with an instance profile.
6. For serverless SQL to work with your instance profile, you might need to explicitly specify the role ARN associated with your instance profile in the **IAM role ARN** field.

This is only a required step if your instance profile's associated role name (the text after the last slash in the role ARN) and the instance profile name (the text after the last slash in the instance profile ARN) do not match. To confirm whether this applies to you:

a. In the AWS console, go to the IAM service's Roles tab. It lists the IAM roles in your account.

b. Click the role with the name that matches the instance profile name in the Databricks SQL admin settings in the **Data Security** section for the **Instance Profile** field that you found earlier in this section.

c. In the summary area, find the **Role ARN** and **Instance Profile ARNs** fields and see if they match.



Role ARN	arn:aws:iam::[redacted]5:role/DatabricksServerlessComputeMyBucket
Role description	Allows EC2 instances to call AWS services on your behalf. | Edit
Instance Profile ARNs	arn:aws:iam::0[redacted]5:instance-profile/DatabricksServerlessComputeMyBucket

d. If they do not match, paste the role ARN into the **IAM role ARN** field. If the names match, you do not need to set the **IAM role ARN** field.

e. Only if you are setting up IAM credential passthrough, select the **Meta Instance Profile** property.

7. Databricks validates that the instance profile ARN is both syntactically and semantically correct. To validate semantic correctness, Databricks does a dry run by launching a cluster with this instance profile. Any failure in this dry run produces a validation error in the UI. Validation of the instance profile can fail if the instance profile contains the `tag-enforcement` policy, preventing you from adding a legitimate instance profile. If the validation fails and you still want to add the instance profile, select the **Skip Validation** checkbox.

8. Click **Add**.

Manage instance profiles

Workspace admins can manage manage access to instance profiles and update them. See Manage instance profiles in Databricks.