Required AWS Account Permissions

Contents:

To access the following AWS resources, you must configure your AWS account or accounts with the listed permissions. These permissions can be applied through AWS access key/secret combinations or through IAM roles applied to the account.

S3

All access to S3 sources occurs through a single AWS account (system mode) or through an individual user's account (user mode). For either mode, the AWS access key and secret combination must provide access to the default bucket associated with the account.

> **NOTE:** These permissions should be set up by your AWS administrator.

Read-only access polices

> **NOTE:** To enable viewing and browsing of all folders within a bucket, the following permissions are required:
>
> - The system account or individual user accounts must have the `ListAllMyBuckets` access permission for the bucket.
> - All objects to be browsed within the bucket must have Get access enabled.

The policy statement to enable read-only access to your default S3 bucket should look similar to the following. Replace `3c-my-s3-bucket` with the name of your bucket:

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "VisualEditor0",
            "Effect": "Allow",
            "Action": [
                "s3:GetObject",
                "s3:ListBucket",
                "s3:GetBucketLocation"
            ],
            "Resource": [
                "arn:aws:s3:::3c-my-s3-bucket",
                "arn:aws:s3:::3c-my-s3-bucket/*",
            ]
        }
    ]
}
```

Write access polices

Write access is enabled by adding the `PutObject` and `DeleteObject` actions to the above. Replace `3c-my-s3-bucket` with the name of your bucket:

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "VisualEditor0",
            "Effect": "Allow",
            "Action": [
                "s3:GetObject",
                "s3:ListBucket",
                "s3:GetBucketLocation",
                "s3:PutObject",
                "s3:DeleteObject"
            ],
            "Resource": [
                "arn:aws:s3:::3c-my-s3-bucket",
                "arn:aws:s3:::3c-my-s3-bucket/*",
            ]
        }
    ]
}
```

Other AWS policies for S3

Policy for access to Trifacta public buckets

To access S3 assets that are created by Trifacta, you must apply the following policy definition to any IAM role that is used to access the Trifacta SaaS. These buckets contain demo assets:

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "VisualEditor1",
            "Effect": "Allow",
            "Action": [
                "s3:GetObject",
                "s3:ListBucket"
            ],
            "Resource": [
                "arn:aws:s3:::aws-saas-samples-prod",
                "arn:aws:s3:::aws-saas-samples-prod/*",
                "arn:aws:s3:::aws-saas-datasets",
                "arn:aws:s3:::aws-saas-datasets/*",
                "arn:aws:s3:::3fac-data-public",
                "arn:aws:s3:::3fac-data-public/*"
                "arn:aws:s3:::trifacta-public-datasets",
                "arn:aws:s3:::trifacta-public-datasets/*"
            ]
        }
    ]
}
```

For more information on creating policies, see *https://console.aws.amazon.com/iam/home#/policies*.

KMS policy

If any accessible bucket is encrypted with KMS-SSE, another policy must be deployed. For more information, see *https://docs.aws.amazon.com/kms/latest/developerguide/iam-policies.html*.

Redshift

Since Redshift requires S3 to be used, to enable read/write access to Redshift using an IAM role, the sole additional requirement to the above is to add the `GetClusterCredentials` permission to the IAM role used for S3. A policy statement similar to the following example needs to be included as part of any IAM role used by the T rifacta platform users to access AWS resources.

The following example policy adds the `GetClusterCredentials` permission for the specified AWS user (`aws:userid`). This user is permitted to get cluster credentials for three different resources:

- a personal Redshift cluster
- The `testdb` cluster
- The `common_group` cluster

```
{
"Version": "2012-10-17",
  "Statement": [
    {
     "Sid": "GetClusterCredsStatement",
      "Effect": "Allow",
      "Action": [
        "redshift:GetClusterCredentials"
      ],
      "Resource": [
        "arn:aws:redshift:us-west-2:123456789012:dbuser:examplecluster/${redshift:DbUser}",
        "arn:aws:redshift:us-west-2:123456789012:dbname:examplecluster/testdb",
        "arn:aws:redshift:us-west-2:123456789012:dbgroup:examplecluster/common_group"
      ],
        "Condition": {
            "StringEquals": {
           "aws:userid":"AIDIODR4TAW7CSEXAMPLE:${redshift:DbUser}@yourdomain.com"
                           }
                     }
    },
  }
}
```

For more information on getClusterCredentials, see *https://docs.aws.amazon.com/redshift/latest/APIReference/API_GetClusterCredentials.html*.

Snowflake

If you are creating a connection to your AWS-based Snowflake deployment, you must specify the following policies in the operative IAM role(s) for each S3 bucket:

Stage bucket

If you are creating your own Snowflake stage, it must point to the default S3 bucket in use by Trifacta SaaS. The policy that you created for read-write access to S3 should be applied to the Snowflake user.

> **NOTE:** If users in your deployment are using IAM roles in user mode for AWS access, then the Snowflake stage must have permissions to write to the user's S3 bucket.

Snowflake bucket

You must create a separate policy to permit access to the S3 bucket that backs your AWS-based Snowflake deployment. The following example permission provides the minimum set of permissions.

Notes:

- The `s3:GetBucketLocation` is required for access to the S3 bucket that Snowflake requires for itself.
- The additional `s3:PutObject` and `s3:DeleteObject` permissions are required only if you plan to unload files to the bucket or automatically purge the files after loading them into a table.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
              "s3:PutObject",
              "s3:GetObject",
              "s3:GetObjectVersion",
              "s3:DeleteObject",
              "s3:DeleteObjectVersion",
              "s3:GetBucketLocation"
            ],
            "Resource": "arn:aws:s3:::<snowflake_bucket_name>/<prefix>/*"
        },
        {
            "Effect": "Allow",
            "Action": "s3:ListBucket",
            "Resource": "arn:aws:s3:::<snowflake_bucket_name>",
            "Condition": {
                "StringLike": {
                    "s3:prefix": [
                        "<prefix>/*"
                    ]
                }
            }
        }
    ]
}
```

Where:

- `<snowflake_bucket_name>` = the name of the S3 bucket that is used by Snowflake
- `<prefix>` = the folder path prefix within the bucket. This value can be omitted if it is not required.
    - The above `StringLike` definition grants access to all prefixes on the bucket.

      > **NOTE:** If your bucket or prefixed path contains more than 1000 files, you may encounter the following error: `Access Denied (Status Code: 403; Error Code: AccessDenied)`.

    - To address the above error, specify the `StringLike` condition with the following change. This change allows access to all files while eliminating the condition that causes the above error:

      ```
      "Condition": {
            "StringLike": {
                "s3:prefix": [
                    "*"
                ]
            }
        }
      ```

For more information, see *https://docs.snowflake.com/en/user-guide/data-load-s3-config-aws-iam-user.html*.

EMR

No additional permissions are required. All jobs are executed on EMR clusters that are managed by the product, which also manages the credentials and policies to access them.