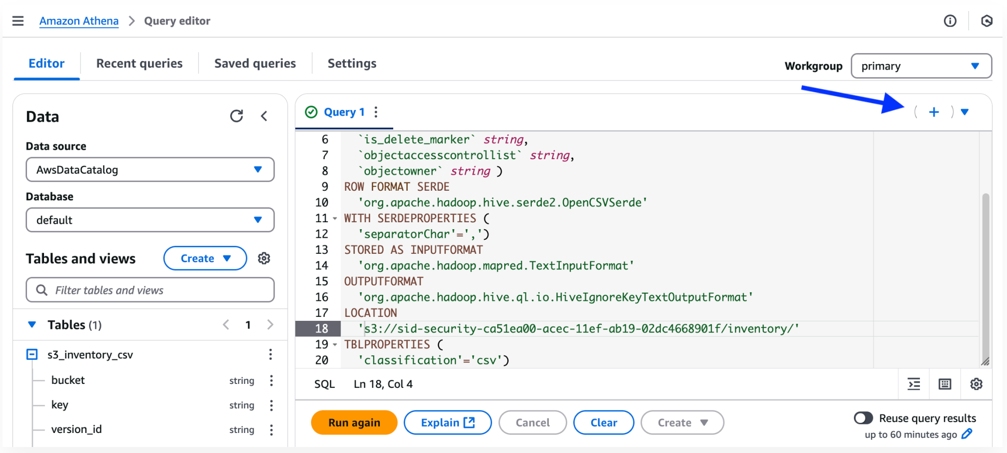
# Use Amazon Athena to query Amazon S3 Inventory and identify objects with ACL elements

You should still be on the Athena page in the AWS console, click the blue plus sign on the top right to add a new tab to the query window, as shown here:

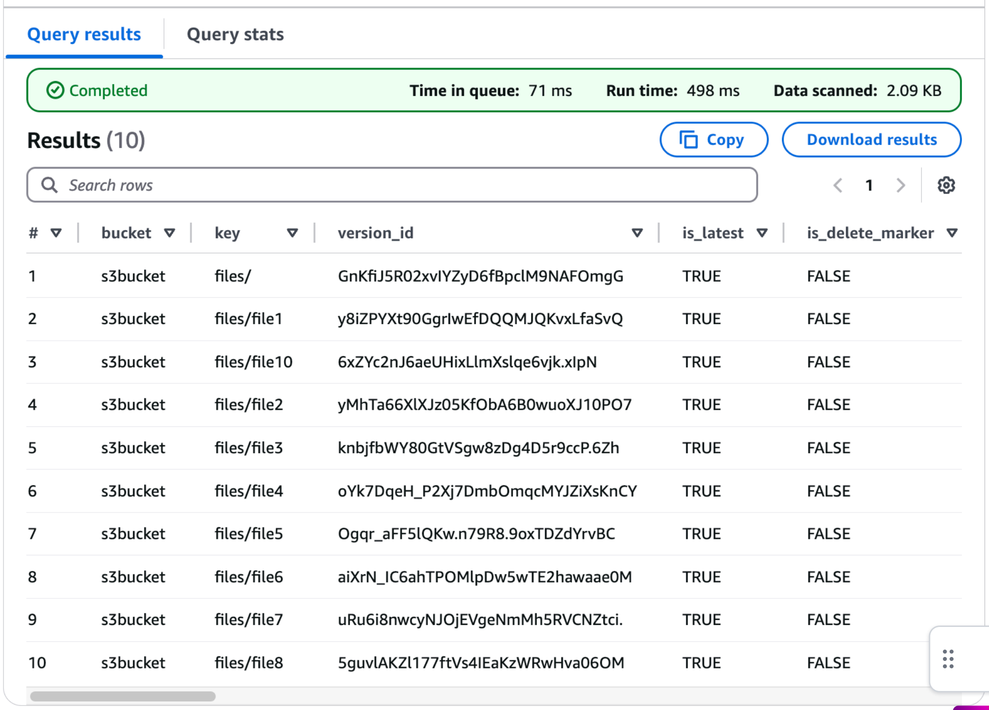


```
  `is_delete_marker` string,
  `objectaccesscontrollist` string,
  `objectowner` string )
ROW FORMAT SERDE
  'org.apache.hadoop.hive.serde2.OpenCSVSerde'
WITH SERDEPROPERTIES (
  'separatorChar'=',')
STORED AS INPUTFORMAT
  'org.apache.hadoop.mapred.TextInputFormat'
OUTPUTFORMAT
  'org.apache.hadoop.hive.ql.io.HiveIgnoreKeyTextOutputFormat'
LOCATION
  's3://sid-security-ca51ea00-acec-11ef-ab19-02dc4668901f/inventory/'
TBLPROPERTIES (
  'classification'='csv')
```

Lets first start with looking at all the data in the Inventory report, this will give context to the data and the schema format it is in. Copy the following in the new tab that we just created:

```
SELECT * FROM "default"."s3_inventory_csv" limit 10;
```

This is a simple select query that will only return 10 records. Click `Run` at the bottom and view the results. Take a moment to scroll up and down and side to side to see all the different elements in the report.



Time in queue: 71 ms   Run time: 498 ms   Data scanned: 2.09 KB

Results (10)

| # | bucket | key | version_id | is_latest | is_delete_marker |
|---|---|---|---|---|---|
| 1 | s3bucket | files/ | GnKfiJ5R02xvIYZyD6fBpclM9NAFOmgG | TRUE | FALSE |
| 2 | s3bucket | files/file1 | y8iZPYXt90GgrIwEfDQQMJQKvxLfaSvQ | TRUE | FALSE |
| 3 | s3bucket | files/file10 | 6xZYc2nJ6aeUHixLlmXslqe6vjk.xIpN | TRUE | FALSE |
| 4 | s3bucket | files/file2 | yMhTa66XlXJz05KfObA6B0wuoXJ10PO7 | TRUE | FALSE |
| 5 | s3bucket | files/file3 | knbjfbWY80GtVSgw8zDg4D5r9ccP.6Zh | TRUE | FALSE |
| 6 | s3bucket | files/file4 | oYk7DqeH_P2Xj7DmbOmqcMYJZiXsKnCY | TRUE | FALSE |
| 7 | s3bucket | files/file5 | Ogqr_aFF5lQKw.n79R8.9oxTDZdYrvBC | TRUE | FALSE |
| 8 | s3bucket | files/file6 | aiXrN_IC6ahTPOMlpDw5wTE2hawaae0M | TRUE | FALSE |
| 9 | s3bucket | files/file7 | uRu6i8nwcyNJOjEVgeNmMh5RVCNZtci. | TRUE | FALSE |
| 10 | s3bucket | files/file8 | 5guvlAKZl177ftVs4IEaKzWRwHva06OM | TRUE | FALSE |

We will now create a query to find all keys (objects) in a bucket that have ACLs and how many have more than one grant. A similar query could be used to find and then take action on the objects in your environment, where ACLs are being used.

As we did before, click on the top right blue plus sign to create a new empty tab, then copy and paste the following query in the new tab that we opened.

Something to take note of below is the objectaccesscontrollist field is a base64 encoded json formatted string, so in order to view it we will need to un-encode via the query below.

```
SELECT key,
    from_utf8(from_base64(objectaccesscontrollist)) as grants,
    json_array_length(
        json_extract(
            from_utf8(from_base64(objectaccesscontrollist)),
            '$.grants'
        )
    ) AS grants_count
FROM "default"."s3_inventory_csv";
```

After you run the query scroll to the right in the "Results" window and you should notice a column called `grants_count`. Are one of these values in this column different than the others? There should be a corresponding row with the value 2, this is because there are two different grants in the ACL block, scroll to the left to look at the ACL.

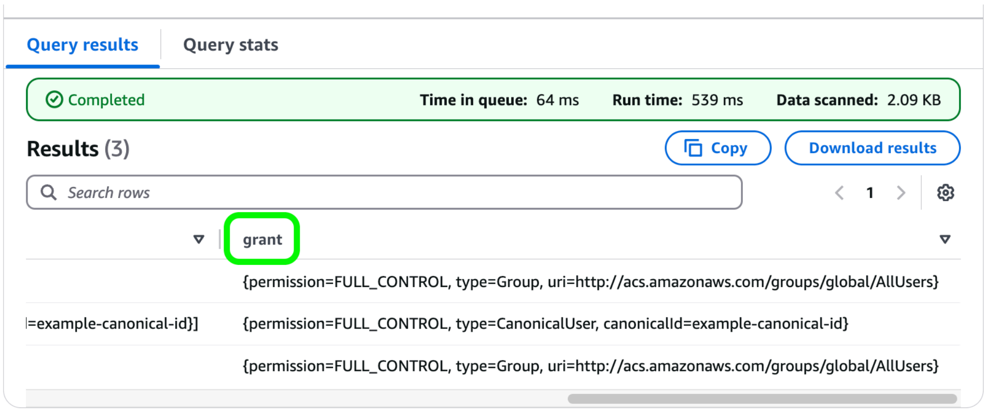
---

In this next scenario we will create a query that shows only objects that have ACLs and a permission called `FULL_CONTROL`. This allows the grantee read and write permissions to an object, typically this can lead to unknown access or misconfigured access to an object.

Once again click the plus sign for a new query window and copy and paste the following:

```
WITH grants AS (
    SELECT key,
        CAST(
            json_extract(from_utf8(from_base64(objectaccesscontrollist)), '$.grants') AS ARRAY
        ) AS grants_array
    FROM s3_inventory_csv
)
SELECT key,
       grants_array,
       grant
FROM grants, UNNEST(grants_array) AS t(grant)
WHERE element_at(grant, 'permission') = 'FULL_CONTROL';
```

Click `Run` and you should see the results, scroll to the right on the results window and notice there is a column named `grants` which have the ACL grant ACL displayed



Completed Time in queue: 64 ms Run time: 539 ms Data scanned: 2.09 KB

Results (3)

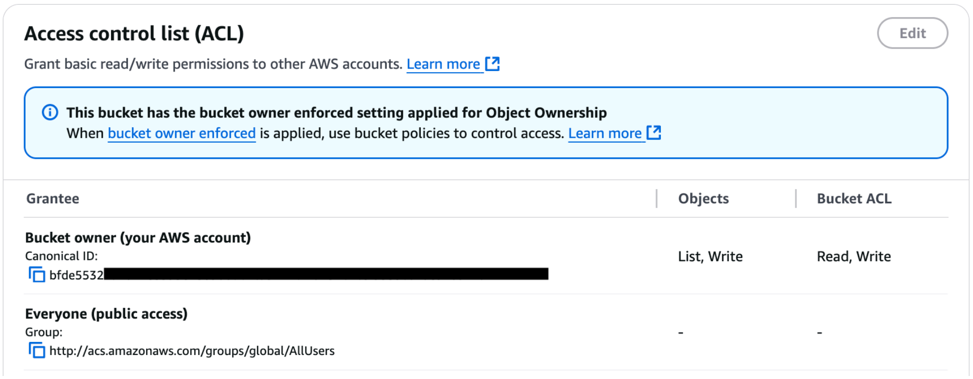
| | grant |
|---|---|
| | {permission=FULL_CONTROL, type=Group, uri=http://acs.amazonaws.com/groups/global/AllUsers} |
| l=example-canonical-id}] | {permission=FULL_CONTROL, type=CanonicalUser, canonicalId=example-canonical-id} |
| | {permission=FULL_CONTROL, type=Group, uri=http://acs.amazonaws.com/groups/global/AllUsers} |

If you were doing this in your own account you might want to take note or download the results a understand why these objects need a permission like FULL_CONTROL.

---

Lastly, we will create a query that looks for objects that have a canonical user ID as part of the grantee permission that belongs to another AWS account. The reason this is important is that this is an obfuscated form of the AWS account ID. You can use this ID to identify an AWS account when granting cross-account access to buckets and objects using Amazon S3. You would be able to give access to another account via the ACL to an object. Again this could lead to unknown or misconfigure access.

This might be the first time you have heard of a canonical ID, so let's find the canonical ID of the AWS account we are using today. To find the canonical user ID for your AWS account, browse to an S3 bucket in your account and select the Permission tab. Browse down to `Access control list (ACL)` to find the canonical user ID for your AWS account, as shown in the following figure.



**Access control list (ACL)**

Grant basic read/write permissions to other AWS accounts. Learn more

**This bucket has the bucket owner enforced setting applied for Object Ownership**
When bucket owner enforced is applied, use bucket policies to control access. Learn more

| Grantee | Objects | Bucket ACL |
|---|---|---|
| **Bucket owner (your AWS account)** Canonical ID: bfde5532[redacted] | List, Write | Read, Write |
| **Everyone (public access)** Group: http://acs.amazonaws.com/groups/global/AllUsers | - | - |

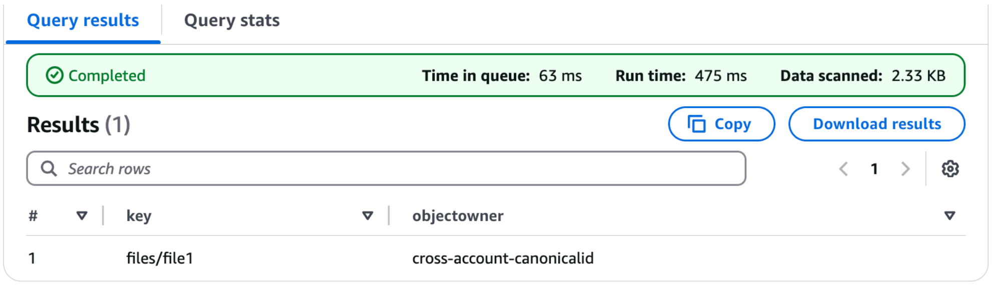
If we were doing this in your own/companies AWS account you could copy the canonical ID and use it in the following query, but in this instance we are going to use our imagination that we have a canonical ID that is called `my-account-canonicalid` and one that is from another AWS account that is called `cross-account-canonicalid`.

Now let's copy and paste the following query and click `Run`

```
SELECT key,objectowner FROM "default"."s3_inventory_csv" WHERE objectowner <> 'my-accou anc
```

Since a canonical user ID can be deemed as somewhat sensitive we are using the name `my-account-canonicalid` instead of a alpha-numeric identifier.

Once the query has been run, you should see a single result, since the query is using the `<>` operator you can think of this as similar to "not equals". So we are looking for a canonical ID that is not associated with our AWS account. As we mentioned previously, this can lead to mis-configured access to S3 objects or unintended access.



Completed **Time in queue:** 63 ms **Run time:** 475 ms **Data scanned:** 2.33 KB

**Results** (1)

| # | key | objectowner |
|---|---|---|
| 1 | files/file1 | cross-account-canonicalid |

We demonstrated how you can use the new object ACL fields in the Amazon S3 Inventory to help audit objects in Amazon S3 that have ACLs configured. This is a key component of migrating from ACLs to policy-based access controls such as bucket policies.

Previous Next