Procedures and API examples

StorageGRID solutions and resources

NetApp
November 21, 2025

This PDF was generated from https://docs.netapp.com/us-en/storagegrid-enable/examples/test-demonstrate-S3-encryption.html on November 21, 2025. Always check docs.netapp.com for the latest.

Table of Contents

Procedures and API examples

Test and demonstrate S3 encryption options on StorageGRID

By Aron Klein

StorageGRID and the S3 API offer a number of different ways to encrypt your data at rest. To learn more, see Review StorageGRID encryption methods.

This guide will demonstrate the S3 API encryption methods.

Server Side Encryption (SSE)

SSE allows the client to store an object and encrypt it with a unique key that is managed by StorageGRID. When the object is requested, the object is decrypted by the key stored in storageGRID.

SSE Example

- PUT an object with SSE

```
aws s3api put-object --bucket <bucket>  --key <file> --body "<file>"
--server-side-encryption AES256 --endpoint-url https://s3.example.com
```

- HEAD the object to verify encryption

```
aws s3api head-object --bucket <bucket>  --key <file>   --endpoint-url
https://s3.example.com
```

```
{
    "AcceptRanges": "bytes",
    "LastModified": "2022-05-02T19:03:03+00:00",
    "ContentLength": 47,
    "ETag": "\"82e8bfb872e778a4687a26e6c0b36bc1\"",
    "ContentType": "text/plain",
    "ServerSideEncryption": "AES256",
    "Metadata": {}
}
```

- GET the object

```
aws s3api get-object --bucket <bucket>  --key <file> <file> --endpoint
-url https://s3.example.com
```

Server Side Encryption with Customer provided keys (SSE-C)

SSE allows the client to store an object and encrypt it with a unique key that is provided by the client with the object. When the object is requested, the same key must be provided in order to decrypt and return the object.

SSE-C Example

- For testing or demonstration purposes you can create an encryption key
  - Create an encryption key

```
openssl enc -aes-128-cbc -pass pass:secret -P`
```

```
salt=E9DBB6603C7B3D2A
key=23832BAC16516152E560F933F261BF03
iv =71E87C0F6EC3C45921C2754BA131A315
```

- Put an object with the generated key

```
aws s3api put-object --bucket <bucket> --key <file> --body "file" --sse
-customer-algorithm AES256 --sse-customer-key
23832BAC16516152E560F933F261BF03 --endpoint-url https://s3.example.com
```

- Head the object

```
aws s3api head-object --bucket <bucket> --key <file> --sse-customer
-algorithm AES256 --sse-customer-key 23832BAC16516152E560F933F261BF03
--endpoint-url https://s3.example.com
```

```
{
    "AcceptRanges": "bytes",
    "LastModified": "2022-05-02T19:20:02+00:00",
    "ContentLength": 47,
    "ETag": "\"f92ef20ab87e0e13951d9bee862e9f9a\"",
    "ContentType": "binary/octet-stream",
    "Metadata": {},
    "SSECustomerAlgorithm": "AES256",
    "SSECustomerKeyMD5": "rjGuMdjLpPV1eRuotNaPMQ=="
}
```

If you do not provide the encryption key, you will receive an error "An error occurred (404) when calling the HeadObject operation: Not Found"

- Get the object

```
aws s3api get-object --bucket <bucket> --key <file> <file> --sse
-customer-algorithm AES256 --sse-customer-key
23832BAC16516152E560F933F261BF03 --endpoint-url https://s3.example.com
```

If you do not provide the encryption key, you will receive an error "An error occurred (InvalidRequest) when calling the GetObject operation: The object was stored using a form of Server Side Encryption. The correct parameters must be provided to retrieve the object."

Bucket Server Side Encryption (SSE-S3)

SSE-S3 allows the client to define a default encryption behavior for all objects stored in a bucket. The objects are encrypted with a unique key that is managed by StorageGRID. When the object is requested, the object is decrypted by they key stored in storageGRID.

Bucket SSE-S3 Example

- Create a new bucket and set a default encryption policy
  - Create new bucket

```
aws s3api create-bucket --bucket <bucket> --region us-east-1
--endpoint-url https://s3.example.com
```

  - Put bucket encryption

```
aws s3api put-bucket-encryption --bucket <bucket> --server-side
-encryption-configuration '{"Rules":
[{"ApplyServerSideEncryptionByDefault": {"SSEAlgorithm":
"AES256"}}]}' --endpoint-url https://s3.example.com
```

- Put an object in the bucket

```
aws s3api put-object --bucket <bucket> --key <file> --body "file"
--endpoint-url https://s3.example.com
```

- Head the object

```
aws s3api head-object --bucket <bucket> --key <file> --endpoint-url
https://s3.example.com
```

```
{
"AcceptRanges": "bytes",
"LastModified": "2022-05-02T20:16:23+00:00",
"ContentLength": 47,
"ETag": "\"82e8bfb872e778a4687a26e6c0b36bc1\"",
"ContentType": "binary/octet-stream",
"ServerSideEncryption": "AES256",
"Metadata": {}
}
```

- GET the object

```
aws s3api get-object --bucket <bucket>  --key <file> <file> --endpoint
-url https://s3.example.com
```

Test and demonstrate S3 object lock on StorageGRID

By Aron Klein

Object Lock provides a WORM model to prevent objects from being deleted or overwritten. StorageGRID implementation of object lock is Cohasset assessed to help meet regulatory requirements, supporting legal hold and compliance mode for object retention, and default bucket retention policies.

This guide will demonstrate the S3 Object Lock API.

Legal hold

- Object Lock legal hold is a simple on/off status applied to an object.

```
aws s3api put-object-legal-hold --bucket <bucket> --key <file> --legal
-hold Status=ON --endpoint-url https://s3.company.com
```

- Verify it with a GET operation.

```
aws s3api get-object-legal-hold --bucket <bucket> --key <file>
--endpoint-url https://s3.company.com
```

```
{
    "LegalHold": {
        "Status": "ON"
    }
}
```

- Turn legal hold off

```
aws s3api put-object-legal-hold --bucket <bucket> --key <file> --legal
-hold Status=OFF --endpoint-url https://s3.company.com
```

- Verify it with a GET operation.

```
aws s3api get-object-legal-hold --bucket <bucket> --key <file>
--endpoint-url https://s3.company.com
```

```
{
    "LegalHold": {
        "Status": "OFF"
    }
}
```

Compliance mode

- The object retention is done with a retain until timestamp.

```
aws s3api put-object-retention --bucket <bucket> --key <file>
--retention '{"Mode":"COMPLIANCE", "RetainUntilDate": "2025-06-
10T16:00:00"}' --endpoint-url https://s3.company.com
```

- Verify the retention status

```
aws s3api get-object-retention --bucket <bucket> --key <file> --endpoint
-url https://s3.company.com
+
```

```
{
    "Retention": {
        "Mode": "COMPLIANCE",
        "RetainUntilDate": "2025-06-10T16:00:00+00:00"
    }
}
```

Default retention

- Set the retention period in days and years verses a retain until date defined with the per object api.

```
aws s3api put-object-lock-configuration --bucket <bucket> --object-lock
-configuration '{"ObjectLockEnabled": "Enabled", "Rule": {
"DefaultRetention": { "Mode": "COMPLIANCE", "Days": 10 }}}' --endpoint
-url https://s3.company.com
```

- Verify the retention status

```
aws s3api get-object-lock-configuration --bucket <bucket> --endpoint-url
https://s3.company.com
```

```
{
    "ObjectLockConfiguration": {
        "ObjectLockEnabled": "Enabled",
        "Rule": {
            "DefaultRetention": {
                "Mode": "COMPLIANCE",
                "Days": 10
            }
        }
    }
}
```

- Put an object in the bucket

```
aws s3api put-object --bucket <bucket> --key <file> --body "file"
--endpoint-url https://s3.example.com
```

- The retention duration set on the bucket is converted to a retention timestamp on the object.

```
aws s3api get-object-retention --bucket <bucket> --key <file> --endpoint
-url https://s3.company.com
```

```
{
    "Retention": {
        "Mode": "COMPLIANCE",
        "RetainUntilDate": "2022-03-02T15:22:47.202000+00:00"
    }
}
```

Test deleting an object with a defined retention

Object Lock is built on top of versioning. The retention is defined on a version of the object. If an attempt is made to delete an object with a retention defined, and no version is specified, a delete marker is created as the current version of the object.

- Delete the object with retention defined

```
aws s3api delete-object --bucket <bucket> --key <file> --endpoint-url
https://s3.example.com
```

- List the objects in the bucket

```
aws s3api list-objects --bucket <bucket> --endpoint-url
https://s3.example.com
```

  - Notice the object is not listed.

- List versions to see the delete marker, and the original locked version

```
aws s3api list-object-versions --bucket <bucket> --prefix <file>
--endpoint-url https://s3.example.com
```

```
{
    "Versions": [
        {
            "ETag": "\"82e8bfb872e778a4687a26e6c0b36bc1\"",
            "Size": 47,
            "StorageClass": "STANDARD",
            "Key": "file.txt",
            "VersionId":
"RDVDMjYwMTQtQkNDQS0xMUVDLThGOEUtNjQ3NTAwQzAxQTk1",
            "IsLatest": false,
            "LastModified": "2022-04-15T14:46:29.734000+00:00",
            "Owner": {
                "DisplayName": "Tenant01",
                "ID": "566223993089512949 26"
            }
        }
    ],
    "DeleteMarkers": [
        {
            "Owner": {
                "DisplayName": "Tenant01",
                "ID": "566223993089512949 26"
            },
            "Key": "file01.txt",
            "VersionId":
"QjVDQzgzOTAtQ0FGNi0xMUVDLThFMzgtQ0RGMjAwQjk0MjM1",
            "IsLatest": true,
            "LastModified": "2022-05-03T15:35:50.248000+00:00"
        }
    ]
}
```

- Delete the locked version of the object

```
aws s3api delete-object  --bucket <bucket> --key <file> --version-id
"<VersionId>" --endpoint-url https://s3.example.com
```

```
An error occurred (AccessDenied) when calling the DeleteObject
operation: Access Denied
```

Policies and permissions in StorageGRID

Here are example Policies and permissions in StorageGRID S3.

The structure of a policy

In StorageGRID, group policies are the same as AWS user (IAM) S3 service policies.

Group policies are required in StorageGRID. A user with S3 access keys but not assigned to a user group, or assigned to a group without a policy granting some permissions, will not be able to access any data.

Bucket and group policies share most of the same elements. Policies are built in json format and can be generated using the AWS policy generator

All policies will define the effect, action(s), and resource(s). Bucket policies will also define a principal.

The **Effect** will be either to Allow or Deny the request.

The **Principal**

- Only applies for bucket policies.
- The principal is the account(s)/user(s) being granted or denied the permissions.
- Can be defined as:
  - A wildcard "*"

```
"Principal":"*"
```

```
"Principal":{"AWS":"*"}
```

  - A tenant ID for all users in a tenant (equivalent to AWS account)

```
"Principal": { "AWS": "27233906934684427525" }
```

  - A user (local or federated from within the tenant the bucket resides, or another tenant in the grid)

```
"Principal": { "AWS":
"arn:aws:iam::76233906934699427431:user/tenant1user1" }
```

```
"Principal": { "AWS": "arn:aws:iam::27233906934684427525:federated-
user/tenant2user1" }
```

  - A group (local or federated from within the tenant the bucket resides, or another tenant in the grid).

```
"Principal": { "AWS":
"arn:aws:iam::76233906934699427431:group/DevOps"  }
```

```
"Principal": { "AWS": "arn:aws:iam::27233906934684427525:federated-
group/Managers"  }
```

The **Action** is the set of S3 operations being granted or denied to the user(s).

> For Group policies, the s3:ListBucket action Allowed is required for users to perform any S3 actions.

The **Resource** is the bucket or buckets the principals being granted or denied the ability to perform the actions on.
Optionally there can be a **Condition** for when the policy action is valid.

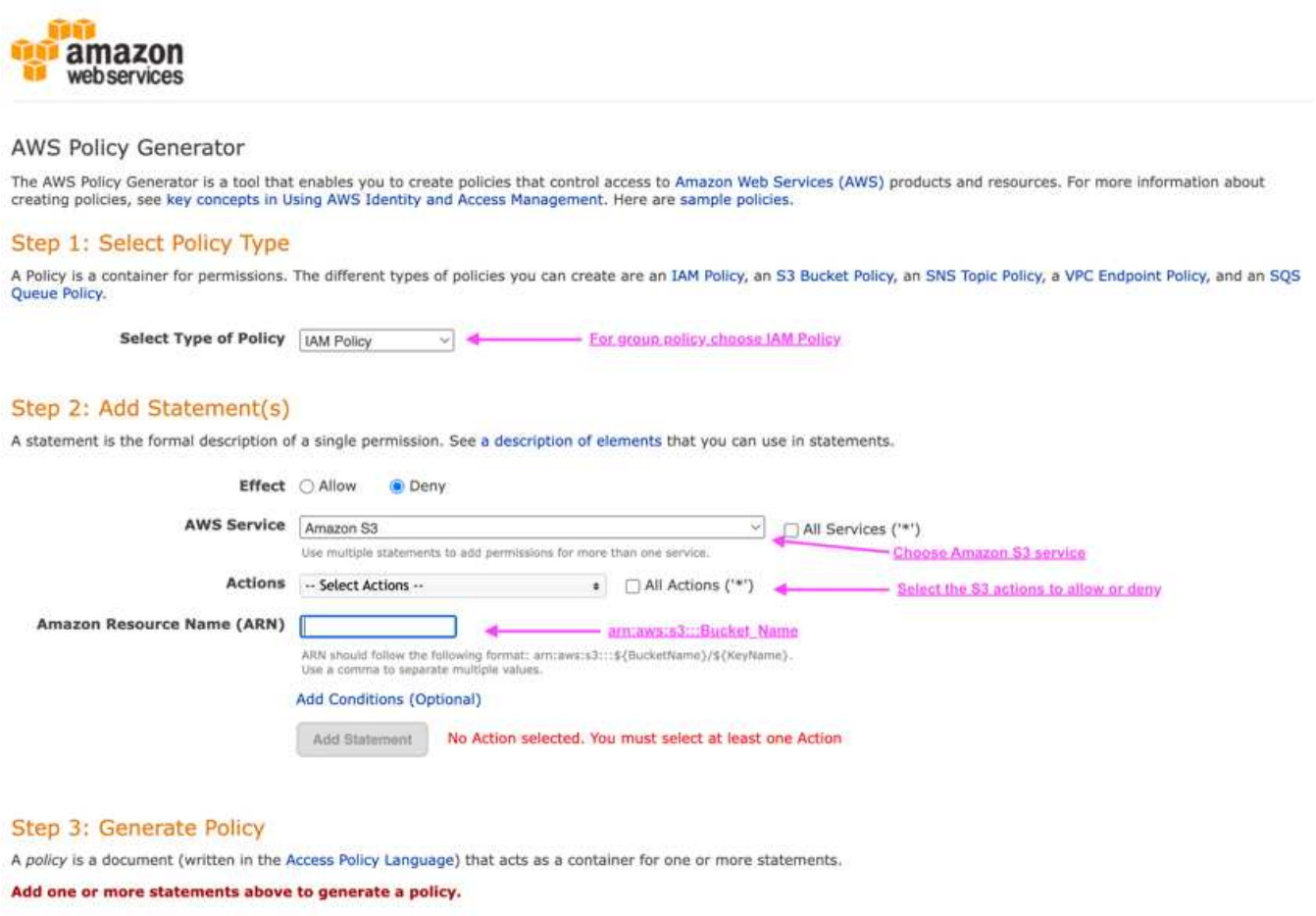
The format of the Json policy will look like this:

```
{
  "Statement": [
    {
      "Sid": "Custom name for this permission",
      "Effect": "Allow or Deny",
      "Principal": {
        "AWS": [
          "arn:aws:iam::tenant_ID::user/User_Name",
          "arn:aws:iam::tenant_ID::federated-user/User_Name",
          "arn:aws:iam::tenant_ID:group/Group_Name",
          "arn:aws:iam::tenant_ID:federated-group/Group_Name",
          "tenant_ID"
        ]
      },
      "Action": [
        "s3:ListBucket",
        "s3:Other_Action"
      ],
      "Resource": [
        "arn:aws:s3:::Example_Bucket",
        "arn:aws:s3:::Example_Bucket/*"
      ],
    }
  ]
}
```

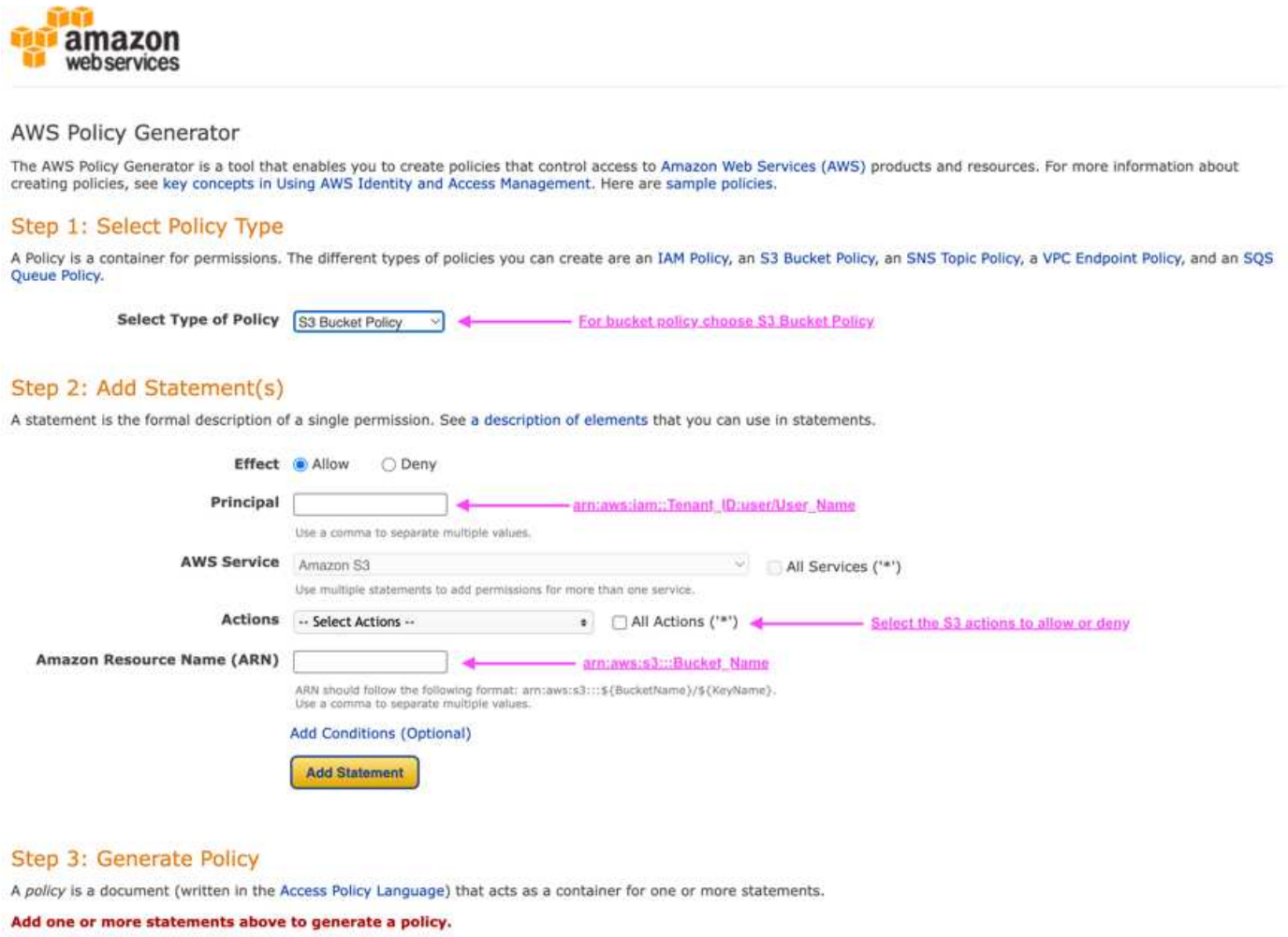
Using the AWS policy generator

The AWS policy generator is a great tool to assist with getting the json code with the correct format and information you are trying to implement.

To generate the permissions for a StorageGRID group policy:
* choose the IAM policy for the type of policy.
* Select the button for the desired effect - Allow or Deny. It is a good practice to start your policies with the deny permissions and then add the allow permissions
* In the actions drop-down click the box next to as many of the S3 actions you want to include in this permission or the "All Actions" box.
* type in the bucket paths in the Amazon Resource Name (ARN) box. include "arn:aws:s3:::" before the bucket name. ex. "arn:aws:s3:::example_bucket"

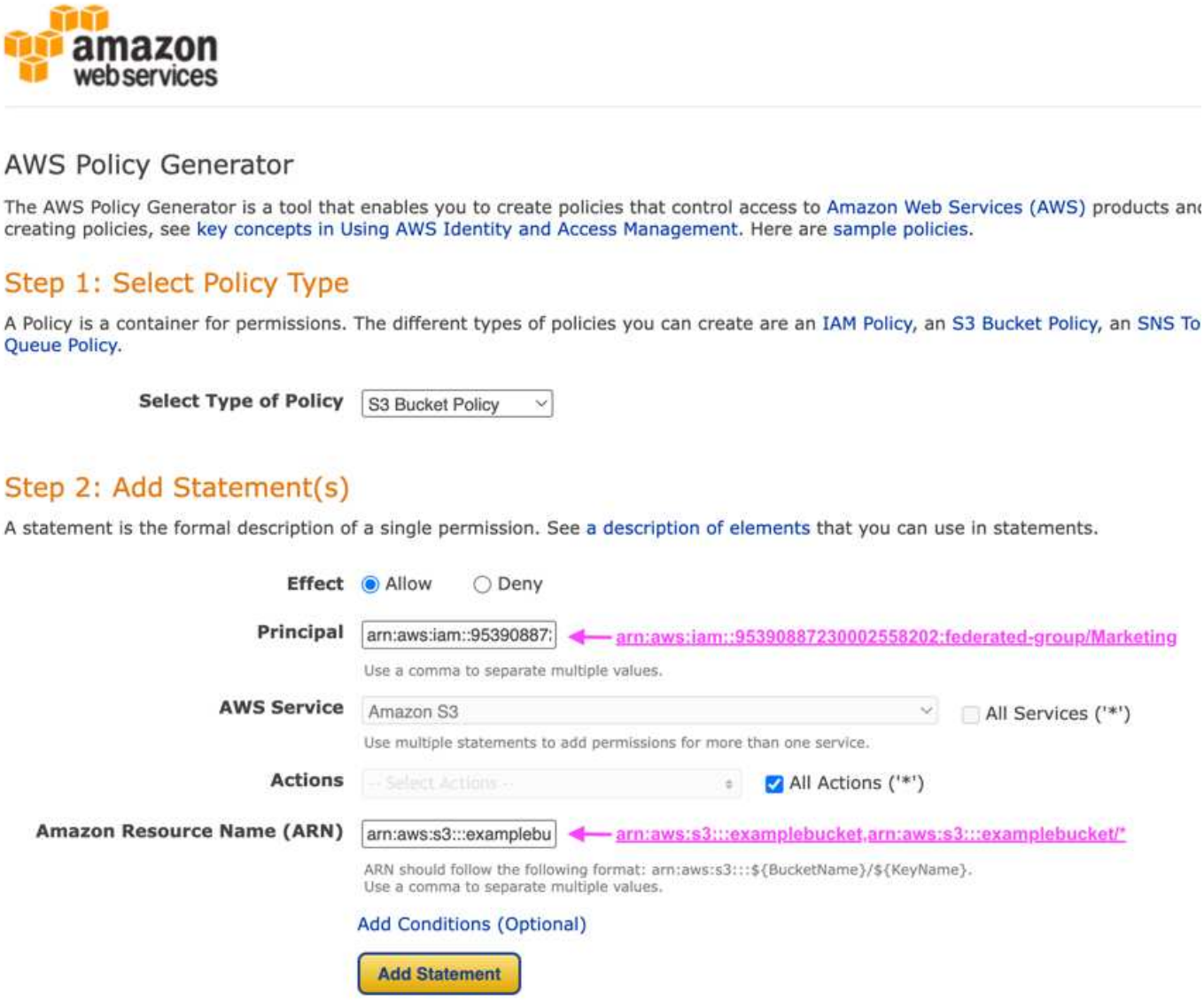


To generate the permissions for a bucket policy:
* choose the S3 Bucket Policy for the type of policy.
* Select the button for the desired effect - Allow or Deny. It is a good practice to start your policies with the deny permissions and then add the allow permissions
* Type in the user or group information for the Principal.
* In the actions drop-down click the box next to as many of the S3 actions you want to include in this permission or the "All Actions" box.
* type in the bucket paths in the Amazon Resource Name (ARN) box. include "arn:aws:s3:::" before the bucket name. ex. "arn:aws:s3:::example_bucket"



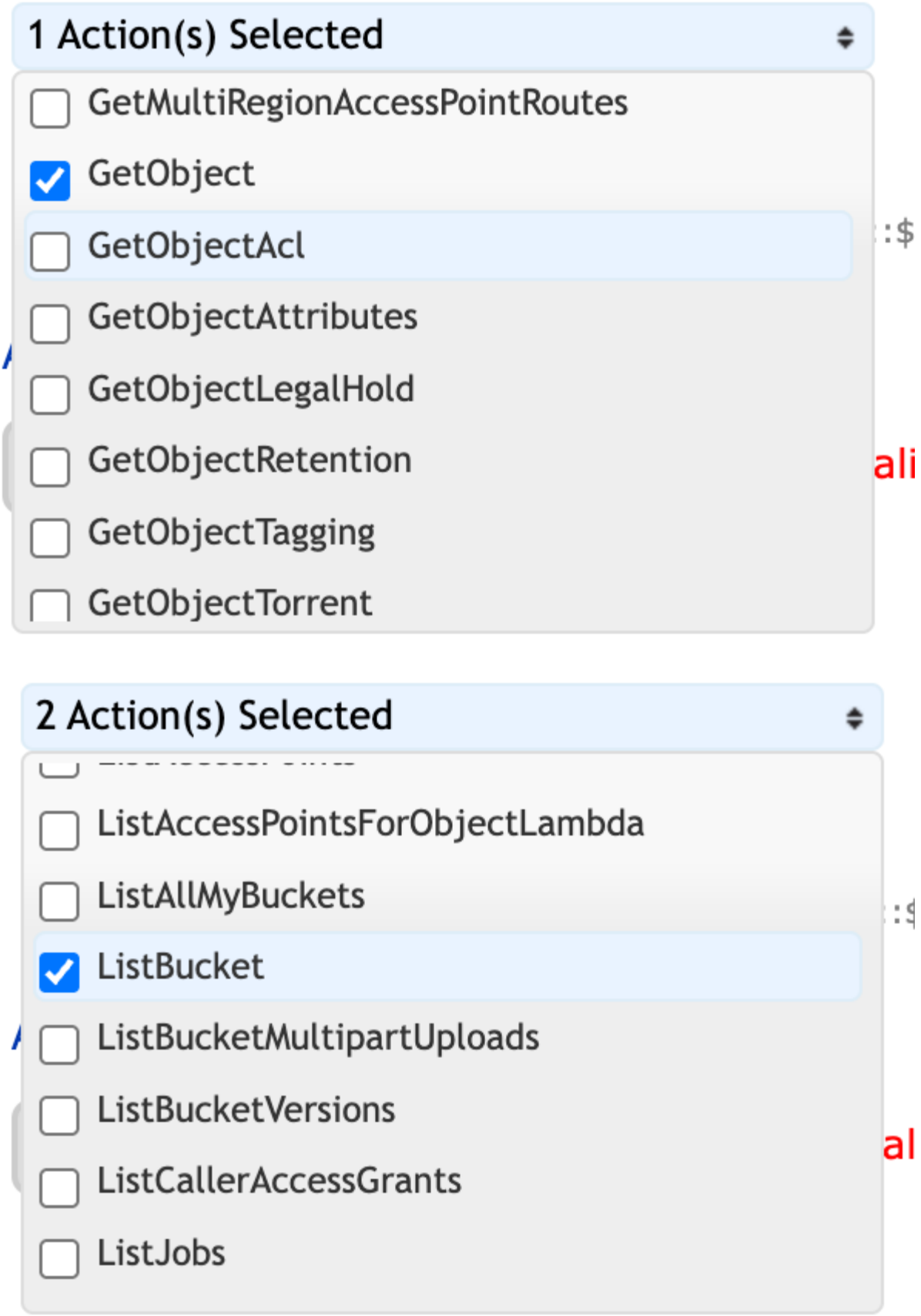
For an example, if you wanted to generate a bucket policy to allow all users to perform GetObject operations on all objects in the bucket, while only users belonging the group "Marketing" in the specified account are allowed full access.

- Select S3 Bucket Policy as the policy type.
- Choose the Allow effect
- Enter the Marketing group information - arn:aws:iam::95390887230002558202:federated-group/Marketing
- Click the box for "All Actions"
- Enter the bucket information - arn:aws:s3:::example_bucket,arn:aws:s3:::example_bucket/*

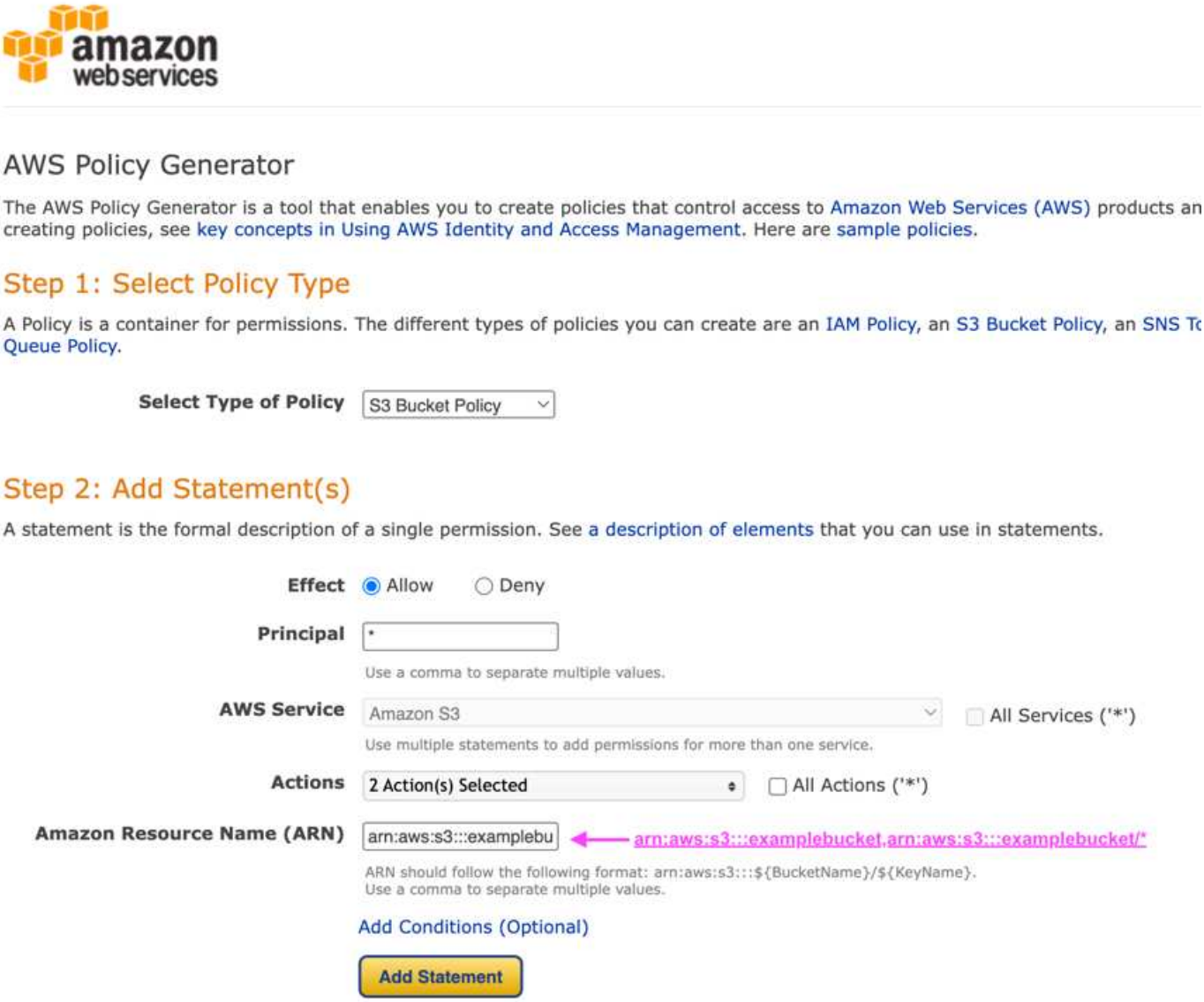


- click the "Add Statement" button

- Choose the Allow effect
- Enter the asterisk * for everyone
- Click the box next to GetObject and ListBucket actions"

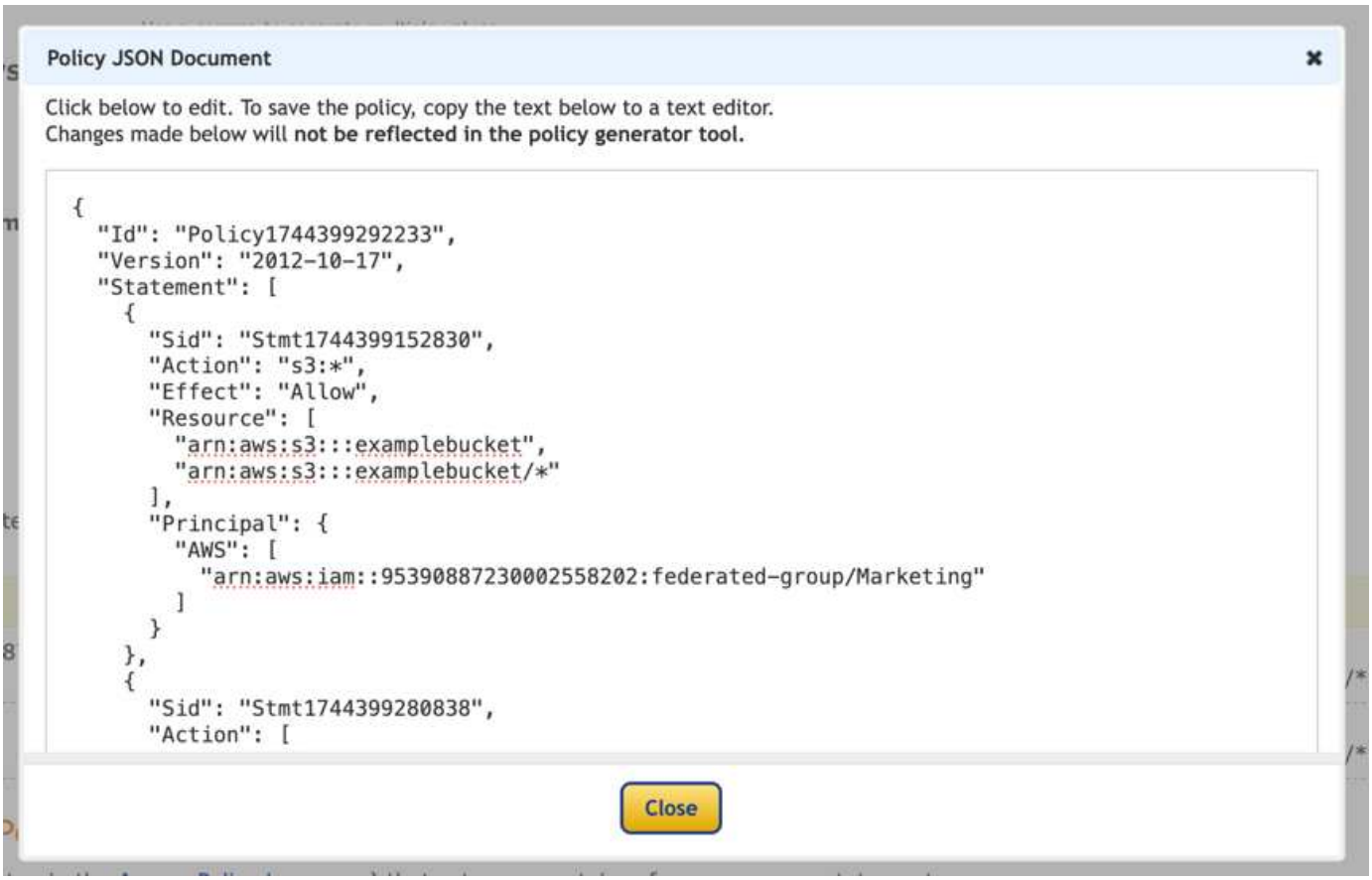


- Enter the bucket information - arn:aws:s3:::example_bucket,arn:aws:s3:::example_bucket/*



- click the "Add Statement" button

- Click on the button "Generate Policy" and a pop-up window will appear with your generated policy.



- Copy out the complete json text that should look like this:

```
{
  "Id": "Policy1744399292233",
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "Stmt1744399152830",
      "Action": "s3:*",
      "Effect": "Allow",
      "Resource": [
        "arn:aws:s3:::example_bucket",
        "arn:aws:s3:::example_bucket/*"
      ],
      "Principal": {
        "AWS": [
          "arn:aws:iam::95390887230002558202:federated-group/Marketing"
        ]
      }
    },
    {
      "Sid": "Stmt1744399280838",
      "Action": [
        "s3:GetObject",
        "s3:ListBucket"
      ],
      "Effect": "Allow",
      "Resource": [
        "arn:aws:s3:::example_bucket",
        "arn:aws:s3:::example_bucket/*"
      ],
      "Principal": "*"
    }
  ]
}
```

This json can be used as is, or you can remove the ID and Version lines above the "Statement" line and you can customize the Sid for each permission with a more meaningful title for each permission or these can be removed as well.

For example:

```
{
  "Statement": [
    {
      "Sid": "MarketingAllowFull",
      "Action": "s3:*",
      "Effect": "Allow",
      "Resource": [
        "arn:aws:s3:::example_bucket",
        "arn:aws:s3:::example_bucket/*"
      ],
      "Principal": {
        "AWS": [
          "arn:aws:iam::95390887230002558202:federated-group/Marketing"
        ]
      }
    },
    {
      "Sid": "EveryoneReadOnly",
      "Action": [
        "s3:GetObject",
        "s3:ListBucket"
      ],
      "Effect": "Allow",
      "Resource": [
        "arn:aws:s3:::example_bucket",
        "arn:aws:s3:::example_bucket/*"
      ],
      "Principal": "*"
    }
  ]
}
```

Group Policies (IAM)

Home Directory style bucket access

This group policy will only allow users to access objects in the bucket named the users username.

```
{
"Statement": [
    {
      "Sid": "AllowListBucketOfASpecificUserPrefix",
      "Effect": "Allow",
      "Action": "s3:ListBucket",
      "Resource": "arn:aws:s3:::home",
      "Condition": {
        "StringLike": {
          "s3:prefix": "${aws:username}/*"
        }
      }
    },
    {
      "Sid": "AllowUserSpecificActionsOnlyInTheSpecificUserPrefix",
      "Effect": "Allow",
      "Action": "s3:*Object",
      "Resource": "arn:aws:s3:::home/?/?/${aws:username}/*"
    }

  ]
}
```

Deny object lock bucket creation

This group policy will restrict users from creating a bucket with object lock enabled on the bucket.

> This policy is not enforced in the StorageGRID UI, it is only enforced by S3 API.

```
{
    "Statement": [
        {
            "Action": "s3:*",
            "Effect": "Allow",
            "Resource": "arn:aws:s3:::*"
        },
        {
            "Action": [
                "s3:PutBucketObjectLockConfiguration",
                "s3:PutBucketVersioning"
            ],
            "Effect": "Deny",
            "Resource": "arn:aws:s3:::*"
        }
    ]
}
```

Object lock retention limit

This Bucket policy will restrict Object-Lock retention duration to 10 days or less

```
{
 "Version":"2012-10-17",
 "Id":"CustSetRetentionLimits",
 "Statement": [
   {
    "Sid":"CustSetRetentionPeriod",
    "Effect":"Deny",
    "Principal":"*",
    "Action": [
      "s3:PutObjectRetention"
    ],
    "Resource":"arn:aws:s3:::testlock-01/*",
    "Condition": {
      "NumericGreaterThan": {
        "s3:object-lock-remaining-retention-days":"10"
      }
    }
   }
  ]
}
```

Restrict users from deleting objects by versionID

This group policy will restrict users from deleting versioned objects by versionID

```
{
    "Statement": [
        {
            "Action": [
                "s3:DeleteObjectVersion"
            ],
            "Effect": "Deny",
            "Resource": "arn:aws:s3:::*"
        },
        {
            "Action": "s3:*",
            "Effect": "Allow",
            "Resource": "arn:aws:s3:::*"
        }
    ]
}
```

Restrict a group to single subdirectory (prefix) with read-only access

This policy allows members of the group to have read-only access to a subdirectory (prefix) within a bucket. The bucket name is "study" and the subdirectory is "study01".

```
{
    "Statement": [
        {
            "Sid": "AllowUserToSeeBucketListInTheConsole",
            "Action": [
                "s3:ListAllMyBuckets"
            ],
            "Effect": "Allow",
            "Resource": [
                "arn:aws:s3:::*"
            ]
        },
        {
            "Sid": "AllowRootAndstudyListingOfBucket",
            "Action": [
                "s3:ListBucket"
            ],
            "Effect": "Allow",
            "Resource": [
                "arn:aws:s3::: study"
```

```
            ],
            "Condition": {
                "StringEquals": {
                    "s3:prefix": [
                        "",
                        "study01/"
                    ],
                    "s3:delimiter": [
                        "/"
                    ]
                }
            }
        },
        {
            "Sid": "AllowListingOfstudy01",
            "Action": [
                "s3:ListBucket"
            ],
            "Effect": "Allow",
            "Resource": [
                "arn:aws:s3:::study"
            ],
            "Condition": {
                "StringLike": {
                    "s3:prefix": [
                        "study01/*"
                    ]
                }
            }
        },
        {
            "Sid": "AllowAllS3ActionsInstudy01Folder",
            "Effect": "Allow",
            "Action": [
                "s3:Getobject"
            ],
            "Resource": [
                "arn:aws:s3:::study/study01/*"
            ]
        }
    ]
}
```

Bucket Policies

Restrict bucket to single user with read-only access

This policy allows a single user to have read-only access to a bucket and explicitly denys access to all other users. Grouping the Deny statements at the top of the policy is a good practice for faster evaluation.

```
{
    "Statement": [
        {
            "Sid": "Deny non user1",
            "Effect": "Deny",
            "NotPrincipal": {
                "AWS": "arn:aws:iam::349215141330028336 65:user/user1"
            },
            "Action": [
                "s3:*"
            ],
            "Resource": [
                "arn:aws:s3:::bucket1",
                "arn:aws:s3:::bucket1/*"
            ]
        },
        {
            "Sid": "Allow user1 read access to bucket bucket1",
            "Effect": "Allow",
            "Principal": {
                "AWS": "arn:aws:iam::349215141330028336 65:user/user1"
            },
            "Action": [
                "s3:GetObject",
                "s3:ListBucket"
            ],
            "Resource": [
                "arn:aws:s3:::bucket1",
                "arn:aws:s3:::bucket1/*"
            ]
        }
    ]
}
```

restrict a bucket to a few users with read-only access.

```
{
    "Statement": [
      {
        "Sid": "Deny all S3 actions to employees 002-005",
        "Effect": "deny",
        "Principal": {
          "AWS": [
            "arn:aws:iam::465215141330027038882:user/employee-002",
            "arn:aws:iam::465215141330027038882:user/employee-003",
            "arn:aws:iam::465215141330027038882:user/employee-004",
            "arn:aws:iam::465215141330027038882:user/employee-005"
          ]
        },
        "Action": "*",
        "Resource": [
          "arn:aws:s3:::databucket1",
          "arn:aws:s3:::databucket1/*"
        ]
      },
      {
        "Sid": "Allow read-only access for employees 002-005",
        "Effect": "Allow",
        "Principal": {
          "AWS": [
            "arn:aws:iam::465215141330027038882:user/employee-002",
            "arn:aws:iam::465215141330027038882:user/employee-003",
            "arn:aws:iam::465215141330027038882:user/employee-004",
            "arn:aws:iam::465215141330027038882:user/employee-005"
          ]
        },
        "Action": [
          "s3:GetObject",
          "s3:GetObjectTagging",
          "s3:GetObjectVersion"
        ],
        "Resource": [
          "arn:aws:s3:::databucket1",
          "arn:aws:s3:::databucket1/*"
        ]
      }
    ]
}
```

Restrict user deletes of versioned objects in a bucket

This bucket policy will restrict a user(identified by userID "56622399308951294926") from deleting versioned objects by versionID

```
{
  "Statement": [
    {
      "Action": [
        "s3:DeleteObjectVersion"
      ],
      "Effect": "Deny",
      "Resource": "arn:aws:s3:::verdeny/*",
      "Principal": {
        "AWS": [
          "56622399308951294926"
        ]
      }
    },
    {
      "Action": "s3:*",
      "Effect": "Allow",
      "Resource": "arn:aws:s3:::verdeny/*",
      "Principal": {
        "AWS": [
          "56622399308951294926"
        ]
      }
    }
  ]
}
```

Bucket lifecycle in StorageGRID

You can create an S3 lifecycle configuration to control when specific objects are deleted from the StorageGRID system.

What is a lifecycle configuration

A lifecycle configuration is a set of rules that are applied to the objects in specific S3 buckets. Each rule specifies which objects are affected and when those objects will expire (on a specific date or after some number of days).

Each object follows the retention settings of either an S3 bucket lifecycle or an ILM policy. When an S3 bucket lifecycle is configured, the lifecycle expiration actions override the ILM policy for objects matching the bucket lifecycle filter. Objects that do not match the bucket lifecycle filter use the retention settings of the ILM policy. If an object matches a bucket lifecycle filter and no expiration actions are explicitly specified, the retention

settings of the ILM policy are not used and it is implied that object versions are retained forever.

As a result, an object might be removed from the grid even though the placement instructions in an ILM rule still apply to the object. Or, an object might be retained on the grid even after any ILM placement instructions for the object have lapsed

StorageGRID supports up to 1,000 lifecycle rules in a lifecycle configuration. Each rule can include the following XML elements:

- Expiration: Delete an object when a specified date is reached or when a specified number of days is reached, starting from when the object was ingested.
- NoncurrentVersionExpiration: Delete an object when a specified number of days is reached, starting from when the object became noncurrent.
- Filter (Prefix, Tag)
- Status
  *ID

StorageGRID supports the use of the following bucket operations to manage lifecycle configurations:

- DeleteBucketLifecycle
- GetBucketLifecycleConfiguration
- PutBucketLifecycleConfiguration

Structure of a lifecycle policy

As the first step in creating a lifecycle configuration, you create a JSON file that includes one or more rules. For example, this JSON file includes three rules, as follows:

1. **Rule 1** applies only to objects that match the prefix category1/ and that have a key2 value of tag2. The Expiration parameter specifies that objects matching the filter will expire at midnight on 22 August 2020.
2. **Rule 2** applies only to objects that match the prefix category2/. The Expiration parameter specifies that objects matching the filter will expire 100 days after they are ingested.

   Rules that specify a number of days are relative to when the object was ingested. If the current date exceeds the ingest date plus the number of days, some objects might be removed from the bucket as soon as the lifecycle configuration is applied.

3. **Rule 3** applies only to objects that match the prefix category3/. The Expiration parameter specifies that any noncurrent versions of matching objects will expire 50 days after they become noncurrent.

```
{
    "Rules": [
        {
            "ID": "rule1",
            "Filter": {
                "And": {
                    "Prefix": "category1/",
                    "Tags": [
                        {
                            "Key": "key2",
                            "Value": "tag2"
                        }
                    ]
                }
            },
            "Expiration": {
                "Date": "2020-08-22T00:00:00Z"
            },
            "Status": "Enabled"
        },
        {
            "ID": "rule2",
            "Filter": {
                "Prefix": "category2/"
            },
            "Expiration": {
                "Days": 100
            },
            "Status": "Enabled"
        },
        {
            "ID": "rule3",
            "Filter": {
                "Prefix": "category3/"
            },
            "NoncurrentVersionExpiration": {
                "NoncurrentDays": 50
            },
            "Status": "Enabled"
        }
    ]
}
```

Apply lifecycle configuration to bucket

After you have created the lifecycle configuration file, you apply it to a bucket by issuing a PutBucketLifecycleConfiguration request.

This request applies the lifecycle configuration in the example file to objects in a bucket named `testbucket`.

```
aws s3api --endpoint-url <StorageGRID endpoint> put-bucket-lifecycle-
configuration
--bucket testbucket --lifecycle-configuration file://bktjson.json
```

To validate that a lifecycle configuration was successfully applied to the bucket, issue a GetBucketLifecycleConfiguration request. For example:

```
aws s3api --endpoint-url <StorageGRID endpoint> get-bucket-lifecycle-
configuration
 --bucket testbucket
```

Example lifecycle policies for standard (non-versioned) buckets

Delete objects after 90 days

Use case: This policy is ideal for managing data that is only relevant for a limited time, such as temporary files, logs, or intermediate processing data.
Benefit: Reduce storage costs and ensure the bucket is clutter-free.

```
{
    "Rules": [
      {
        "ID": "Delete after 90 day rule",
        "Filter": {},
        "Status": "Enabled",
          "Expiration": {
              "Days": 90
        }
      }
    ]
}
```

Example lifecycle policies for versioned buckets

Delete noncurrent versions after 10 days

Use case: This policy helps manage the storage of non-current version objects, which can accumulate over time and consume significant space.
Benefit: Optimize storage usage by keeping only the most recent version.

```
{
    "Rules": [
            {
        "ID": "NoncurrentVersionExpiration 10 day rule",
        "Filter": {},
        "Status": "Enabled",
          "NoncurrentVersionExpiration": {
              "NoncurrentDays": 10
        }
    }
    ]
}
```

Keep 5 noncurrent versions

Use case: Useful where you want to retain a limited number of previous versions for recovery or audit purposes
Benefit: Retain enough non-current versions to ensure sufficient history and recovery points.

```
{
    "Rules": [
      {
        "ID": "NewerNoncurrentVersions 5 version rule",
        "Filter": {},
        "Status": "Enabled",
        "NoncurrentVersionExpiration": {
            "NewerNoncurrentVersions": 5
        }
    }
    ]
}
```

Remove delete markers when no other versions exist

Use case: This policy helps manage the delete markers left over after all non-current versions are removed which can accumulate over time.
Benefit: Reduce unnecessary clutter.

```
{
    "Rules": [
    {
        "ID": "Delete marker cleanup rule",
        "Filter": {},
        "Status": "Enabled",
        "Expiration": {
        "ExpiredObjectDeleteMarker": true
        }
    }
    ]
}
```

Delete current versions after 30 days, delete non-current versions after 60 days, and remove the delete markers created by the current version delete once no other versions exist.

Use case: Provide a complete lifecycle for current and non-current versions including the delete markers. Benefit: Reduce storage costs and ensure the bucket is clutter-free while retaining sufficient recovery points and history.

```
{
  "Rules": [
    {
      "ID": "Delete current version",
      "Filter": {},
      "Status": "Enabled",
      "Expiration": {
        "Days": 30
      }
    },
    {
      "ID": "noncurrent version retention",
      "Filter": {},
      "Status": "Enabled",
      "NoncurrentVersionExpiration": {
        "NoncurrentDays": 60
      }
    },
    {
      "ID": "Markers",
      "Filter": {},
      "Status": "Enabled",
      "Expiration": {
        "ExpiredObjectDeleteMarker": true
      }
    }
  ]
}
```

remove delete markers that have no other versions, Retain 4 non-current versions and at least 30 days worth of history for objects with the "accounts_ prefix" and keep 2 versions and at least 10 days worth of history for all other object versions.

Use case: Provide unique rules for specific objects along side other objects to manage the complete lifecycle for current and non-current versions including the delete markers.
Benefit: Reduce storage costs and ensure the bucket is clutter-free while retaining sufficient recovery points and history to meet a mix of client requirements.

```
{
  "Rules": [
    {
      "ID": "Markers",
      "Filter": {},
      "Status": "Enabled",
      "Expiration": {
        "ExpiredObjectDeleteMarker": true
      }
    },
    {
      "ID": "accounts version retention",
      "Filter": {"Prefix":"account_"},
      "Status": "Enabled",
      "NoncurrentVersionExpiration": {
        "NewerNoncurrentVersions": 4,
        "NoncurrentDays": 30
      }
    },
    {
      "ID": "noncurrent version retention",
      "Filter": {},
      "Status": "Enabled",
      "NoncurrentVersionExpiration": {
        "NewerNoncurrentVersions": 2,
        "NoncurrentDays": 10
      }
    }
  ]
}
```

Conclusion

- Regularly review and update lifecycle policies and align them with ILM and data management goals.
- Test policies in a non-production environment or bucket before applying them broadly to ensure they work as intended
- Use Descriptive ID’s for rules to make it more intuitive, as the logic structure can get complex
- Monitor the impact of these bucket lifecycle policies on storage usage and performance to make necessary adjustments.

Copyright information

Copyright © 2025 NetApp, Inc. All Rights Reserved. Printed in the U.S. No part of this document covered by copyright may be reproduced in any form or by any means—graphic, electronic, or mechanical, including photocopying, recording, taping, or storage in an electronic retrieval system—without prior written permission of the copyright owner.

Software derived from copyrighted NetApp material is subject to the following license and disclaimer:

THIS SOFTWARE IS PROVIDED BY NETAPP “AS IS” AND WITHOUT ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE, WHICH ARE HEREBY DISCLAIMED. IN NO EVENT SHALL NETAPP BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

NetApp reserves the right to change any products described herein at any time, and without notice. NetApp assumes no responsibility or liability arising from the use of products described herein, except as expressly agreed to in writing by NetApp. The use or purchase of this product does not convey a license under any patent rights, trademark rights, or any other intellectual property rights of NetApp.

The product described in this manual may be protected by one or more U.S. patents, foreign patents, or pending applications.

LIMITED RIGHTS LEGEND: Use, duplication, or disclosure by the government is subject to restrictions as set forth in subparagraph (b)(3) of the Rights in Technical Data -Noncommercial Items at DFARS 252.227-7013 (FEB 2014) and FAR 52.227-19 (DEC 2007).

Data contained herein pertains to a commercial product and/or commercial service (as defined in FAR 2.101) and is proprietary to NetApp, Inc. All NetApp technical data and computer software provided under this Agreement is commercial in nature and developed solely at private expense. The U.S. Government has a non-exclusive, non-transferrable, nonsublicensable, worldwide, limited irrevocable license to use the Data only in connection with and in support of the U.S. Government contract under which the Data was delivered. Except as provided herein, the Data may not be used, disclosed, reproduced, modified, performed, or displayed without the prior written approval of NetApp, Inc. United States Government license rights for the Department of Defense are limited to those rights identified in DFARS clause 252.227-7015(b) (FEB 2014).

Trademark information

NETAPP, the NETAPP logo, and the marks listed at http://www.netapp.com/TM are trademarks of NetApp, Inc. Other company and product names may be trademarks of their respective owners.