BUCKET POLICIES

New in version Luminous.

The Ceph Object Gateway supports a subset of the Amazon S3 policy language applied to buckets.

CREATION AND REMOVAL

Bucket policies are managed through standard S3 operations rather than radosgw-admin.

For example, one may use s3cmd to set or delete a policy thus:

```
$ cat > examplepol
{
  "Version": "2012-10-17",
  "Statement": [{
    "Effect": "Allow",
    "Principal": {"AWS": ["arn:aws:iam::usfolks:user/fred"]},
    "Action": "s3:PutObjectAcl",
    "Resource": [
      "arn:aws:s3:::happybucket/*"
    ]
  }]
}

$ s3cmd setpolicy examplepol s3://happybucket
$ s3cmd delpolicy s3://happybucket
```

LIMITATIONS

Currently, we support only the following actions:

- s3:AbortMultipartUpload
- s3:CreateBucket
- s3:DeleteBucketPolicy
- s3:DeleteBucket
- s3:DeleteBucketWebsite
- s3:DeleteObject
- s3:DeleteObjectVersion
- s3:DeleteReplicationConfiguration
- s3:GetAccelerateConfiguration
- s3:GetBucketAcl
- s3:GetBucketCORS
- s3:GetBucketLocation
- s3:GetBucketLogging
- s3:GetBucketNotification
- s3:GetBucketPolicy
- s3:GetBucketRequestPayment
- s3:GetBucketTagging
- s3:GetBucketVersioning
- s3:GetBucketWebsite
- s3:GetLifecycleConfiguration
- s3:GetObjectAcl
- s3:GetObject
- s3:GetObjectTorrent
- s3:GetObjectVersionAcl
- s3:GetObjectVersion
- s3:GetObjectVersionTorrent
- s3:GetReplicationConfiguration
- s3:ListAllMyBuckets
- s3:ListBucketMultiPartUploads
- s3:ListBucket

- s3:ListBucketVersions
- s3:ListMultipartUploadParts
- s3:PutAccelerateConfiguration
- s3:PutBucketAcl
- s3:PutBucketCORS
- s3:PutBucketLogging
- s3:PutBucketNotification
- s3:PutBucketPolicy
- s3:PutBucketRequestPayment
- s3:PutBucketTagging
- s3:PutBucketVersioning
- s3:PutBucketWebsite
- s3:PutLifecycleConfiguration
- s3:PutObjectAcl
- s3:PutObject
- s3:PutObjectVersionAcl
- s3:PutReplicationConfiguration
- s3:RestoreObject

We do not yet support setting policies on users, groups, or roles.

We use the RGW 'tenant' identifier in place of the Amazon twelve-digit account ID. In the future we may allow you to assign an account ID to a tenant, but for now if you want to use policies between AWS S3 and RGW S3 you will have to use the Amazon account ID as the tenant ID when creating users.

Under AWS, all tenants share a single namespace. RGW gives every tenant its own namespace of buckets. There may be an option to enable an AWS-like 'flat' bucket namespace in future versions. At present, to access a bucket belonging to another tenant, address it as "tenant:bucket" in the S3 request.

In AWS, a bucket policy can grant access to another account, and that account owner can then grant access to individual users with user permissions. Since we do not yet support user, role, and group permissions, account owners will currently need to grant access directly to individual users, and granting an entire account access to a bucket grants access to all users in that account.

Bucket policies do not yet support string interpolation.

For all requests, condition keys we support are: - aws:CurrentTime - aws:EpochTime - aws:PrincipalType - aws:Referer - aws:SecureTransport - aws:SourceIp - aws:UserAgent - aws:username

We support certain s3 condition keys for bucket and object requests.

BUCKET RELATED OPERATIONS

Permission	Condition Keys	Comments
s3:createBucket	s3:x-amz-acl s3:x-amz-grant-<perm> where perm is one of read/write/read-acp write-acp/ full-control	
s3:ListBucket &	s3:prefix	
s3:ListBucketVersions	s3:delimiter	
	s3:max-keys	
s3:PutBucketAcl	s3:x-amz-acl s3:x-amz-grant-<perm>	

OBJECT RELATED OPERATIONS

Permission	Condition Keys	Comments
s3:PutObject	s3:x-amz-acl & s3:x-amz-grant-<perm>	
	s3:x-amz-copy-source	
	s3:x-amz-server-side-encryption	
	s3:x-amz-server-side-encryption-aws-kms-key-id	
	s3:x-amz-metadata-directive	PUT & COPY to overwrite/preserve metadata in COPY requests
	s3:RequestObjectTag/<tag-key>	

s3:PutObjectAcl s3:PutObjectVersionAcl	s3:x-amz-acl & s3-amz-grant-<perm>
	s3:ExistingObjectTag/<tag-key>
s3:PutObjectTagging & s3:PutObjectVersionTagging	s3:RequestObjectTag/<tag-key>
	s3:ExistingObjectTag/<tag-key>
s3:GetObject & s3:GetObjectVersion	s3:ExistingObjectTag/<tag-key>
s3:GetObjectAcl & s3:GetObjectVersionAcl	s3:ExistingObjectTag/<tag-key>
s3:GetObjectTagging & s3:GetObjectVersionTagging	s3:ExistingObjectTag/<tag-key>
s3:DeleteObjectTagging & s3:DeleteOBjectVersionTagging	s3:ExistingOBjectTag/<tag-key>

More may be supported soon as we integrate with the recently rewritten Authentication/Authorization subsystem.

SWIFT

There is no way to set bucket policies under Swift, but bucket policies that have been set govern Swift as well as S3 operations.

Swift credentials are matched against Principals specified in a policy in a way specific to whatever backend is being used.