



AWS
SIMPLE STORAGE SERVICE

OR S3 FOR NORMAL PEOPLE

BY ALEXANDER ZOTOV

WHAT IS S3?

- Object storage
- Almost unlimited amount of data, accessible from anywhere
- 99.999999999% durability (that's eleven nines!)
- Cheapest way to store data on AWS
- Can even host static websites
- Supports BitTorrent, too
- Integrates with many AWS services

COMMON USE CASES

- Backup and recovery
- Data archiving
- Data lakes
- Hybrid cloud storage
- Cloud-native application data

BUCKET – CONTAINER RESOURCE

- Logical resource, similar to directory
- Region-specific, but has globally unique name
- Has its own set of access policies and ACLs
- Has multiple bucket-wide options:
 - Versioning
 - Lifecycle management
 - Logging
 - Notifications
 - Cross-region replication
 - And many more

OBJECT – KEY-VALUE RESOURCE

- **Object** is a key-value pair: key is file name, value is the content
- Can be **versioned**
- Metadata is a set of key-value pairs that store information about an **object**
- Has subresources, such as torrent and **ACL**
- Each **object** has a **storage class** associated with it

STORAGE CLASSES

- Standard (STANDARD & RRS) – default storage class
 - STANDARD – millisecond access times, full durability/availability
 - RRS – reduced redundancy storage – is meaningless now, don't use it
- Infrequent access (STANDARD_IA & ONEZONE_IA) – for infrequently accessed files
 - STANDARD_IA: millisecond access times, cheaper storage, expensive requests
 - ONEZONE_IA: like standard, but less available/resilient, so its somewhat cheaper
 - Suitable for files over 128KB that you plan to store for at least 30 days
- Glacier – for archiving data
 - Not available in real time! You need to restore objects first
 - Very cheap storage, very expensive requests

STORAGE CLASSES – IN NUMBERS

| Storage class | Durability | Availability | Comments |
|---------------|----------------|--------------|-----------------------|
| STANDARD | 99.9999999999% | 99.99% | |
| RRS | 99.99% | 99.99% | Useless! |
| STANDARD_IA | 99.9999999999% | 99.9% | |
| ONEZONE_IA | 99.9999999999% | 99.5% | Single AZ |
| GLACIER | 99.9999999999% | 99.99% | Need to restore first |

VERSIONING

- Off by default
- Useful to prevent unintended deletions or overwrites
- Once versioning is enabled, you cannot disable it (you can still suspend it)
- Each object version is stored separately (takes more space)
- GET request returns the latest version by default – you can specify version id to get specific version
- DELETE request does not delete all versions, it just puts a delete marker as a current version. You can still permanently delete specific versions of an object

ACL – ACCESS CONTROL LISTS

- A resource-based access policy
- Applies both to **buckets** and **objects**, each has an **ACL** attached as a **subresource**
- Works on account / group level
- Can be used to grant read/write permissions to other accounts
- Limitations:
 - Cannot be used to grant permissions to **IAM** users
 - No conditional permissions
 - No deny rules

ACL - GRANTEE

- A **Grantee** is an entity that receives permissions
- A **Grantee** could be:
 - An AWS account (identified by a Canonical User Id)
 - A predefined group (represented by a URL):
 - Authenticated Users (<http://acs.amazonaws.com/groups/global/AuthenticatedUsers>)
 - All Users (<http://acs.amazonaws.com/groups/global/AllUsers>)
 - Log Delivery (<http://acs.amazonaws.com/groups/s3/LogDelivery>)

ACL - PERMISSION

- **Permissions** describe which actions a **Grantee** is allowed to perform on a **resource**
- You can grant following permissions:

| Permission | When granted on a bucket | When granted on an object |
|--------------|--|---|
| READ | Allows to list objects in the bucket | Allows to read object data and metadata |
| WRITE | Allows to create, overwrite, delete any object in the bucket | Not applicable |
| READ_ACP | Allows to read bucket ACL | Allows to read object ACL |
| WRITE_ACP | Allows to write ACL for the bucket | Allows to write ACL for the object |
| FULL_CONTROL | Same as all of the above | Same as all of the above |

ACL – CANNED ACL

| Canned ACL | Applies to | Permissions added to ACL |
|---------------------------|-------------------|--|
| private | Bucket and object | Owner: FULL_CONTROL (default) |
| public-read | Bucket and object | Owner: FULL_CONTROL, AllUsers: READ |
| public-read-write | Bucket and object | Owner: FULL_CONTROL, AllUsers: READ, WRITE |
| aws-exec-read | Bucket and object | Owner: FULL_CONTROL, EC2: READ access to GET an AMI from S3 |
| authenticated-read | Bucket and object | Owner: FULL_CONTROL, AuthenticatedUsers: READ |
| bucket-owner-read | Object | Object owner: FULL_CONTROL, bucket owner: READ Ignored during bucket creation |
| bucket-owner-full-control | Object | Object owner, bucket owner: FULL_CONTROL Ignored during bucket creation |
| log-delivery-write | Bucket | LogDelivery: WRITE, READ_ACP |

POLICIES – POLICY LANGUAGE

- JSON-based documents
- User policies ([IAM](#)) and Bucket policies ([S3](#))
- Policies consist of following sections:
 - Resources: [buckets](#) and [objects](#) in S3, identified by [ARN](#)
 - Actions: for each resource you can define a set of operations that will be allowed or denied
 - Effect: allow or deny
 - Principal: account, user, service, or other entity affected by the policy
 - Condition (optional): lets you specify conditions for when your policy is in effect

POLICIES – AN EXAMPLE

```
{
  "Version": "2012-10-17",
  "Id": "ExamplePolicy01",
  "Statement": [{
    "Sid": "ExampleStatement01",
    "Effect": "Allow",
    "Principal": {
      "AWS": "arn:aws:iam::Account-ID:user/Dave"
    },
    "Action": [
      "s3:GetObject",
      "s3:GetBucketLocation",
      "s3:ListBucket"
    ],
    "Resource": [
      "arn:aws:s3:::examplebucket/*",
      "arn:aws:s3:::examplebucket"
    ]
  }]
}
```

POLICIES – SPECIFYING RESOURCES

- Resources are specified by **ARN**
- Arn format: **arn:partition:service:region:namespace:relative-id**
 - Partition : commonly just “aws”, “aws-cn” in China
 - Service: “s3” in our case
 - Region: not needed for s3
 - Namespace: not needed for s3
 - Relative-id: either bucket name or bucket-name/object-key.
- You can use wildcards (* and ?), but they cannot span segments
- You can also use policy variables, such as `${aws:username}` (requires version 2012-10-17)

POLICIES – SPECIFYING PRINCIPALS

- Can be an account, user, service, or other entity
- To grant permissions to an account:
 - "Principal":{"AWS":"arn:aws:iam::[accountid](#):root"}
 - "Principal":{"CanonicalUser":"[canonical_user_id](#)"}
- To grant permissions to an user:
 - "Principal":{"AWS":"arn:aws:iam::[accountid](#):user/[username](#)"}
- To grant permissions to everyone:
 - "Principal":"*"
 - "Principal":{"AWS":"*"}

POLICIES – SPECIFYING PERMISSION

- Permissions are keywords that map to S3 operations (GET, PUT, DELETE, etc).
- Format: s3:<Action><Resource><Property>
- Common Actions are: Get, Put/Create, Delete, Abort, Restore, List
- Common Resources are: Object, Bucket, MultipartUpload,
- Common Properties: Acl, Version, Tagging, Parts
- Wildcards are allowed
- Examples:
 - s3:ListBucket
 - s3:List*
 - s3:GetBucketAcl
 - s3>DeleteObjectVersion

POLICIES – SPECIFYING CONDITIONS

- Access policies allow you to specify conditions when policy takes effect
- Use Boolean operators and special expressions to match your condition against values in the request
- <https://docs.aws.amazon.com/AmazonS3/latest/dev/amazon-s3-policy-keys.html>

POLICIES – USER POLICIES

- You can use IAM user policies to control access to S3 resources
- ACLs, bucket policies, and user policies are all affect S3 resources
- Will be covered in [IAM](#) section

