

# AWS S3

Ravindra Kudache

# Outline

- Concepts
- How to Access S3
- Scalability
- Availability
- Versioning
- Consistency

<http://doc.s3.amazonaws.com/2006-03-01/AmazonS3.wsdl>

## Concepts of S3 – Bucket

- Container for objects stored in S3
- Unlimited size
- Organize the namespace at the highest level
- Internet accessible storage via HTTP/HTTPS
- Global unique name



From: <http://techblogsearch.com/a/amazon-announced-new-s3-usability-enhancements.html>

<http://doc.s3.amazonaws.com/2006-03-01/AmazonS3.wsdl>

## Concepts of S3 – Object

- Similar to files.
- No hierarchy.
- Objects are immutable.
- Size up to 5 TB
- Uniquely identified within a bucket by a key(name) and a version ID



<http://doc.s3.amazonaws.com/2006-03-01/AmazonS3.wsdl>

## Concepts of S3 – Key

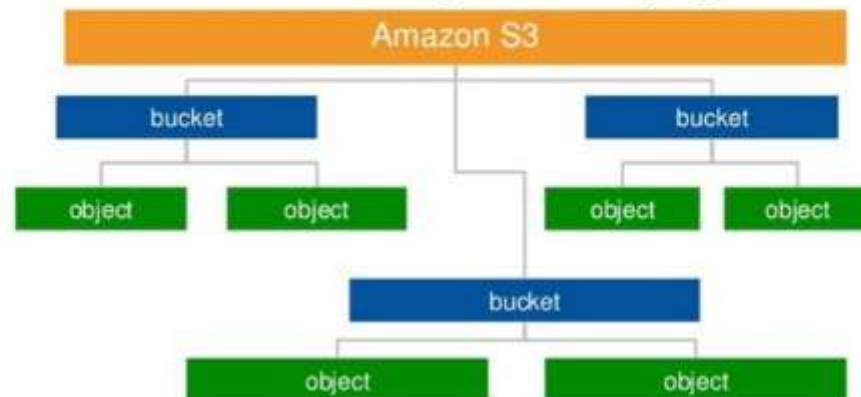
- Name of an object
- Unique identifier for an object within a bucket
- Use the object key to retrieve the object

# Concepts of S3 – Namespace

Globally Unique



Bucket Name + Object Name (key)



## How to Access S3 – CLI & REST

- **AWS Command**
  - ✦ **Put Object**  
aws s3api put-object bucket key
  - ✦ **Get Object**  
aws s3api get-object bucket key outfile
- **REST API**
  - ✦ **Put Object**
  - ✦ **Get Object**

```
GET /ObjectName HTTP/1.1
Host: BucketName.s3.amazonaws.com
Date: date
Authorization: authorization string
```

```
PUT /ObjectName HTTP/1.1
Host: BucketName.s3.amazonaws.com
Date: date
Authorization: authorization string
```

## How to Access S3 – AWS SDK

## SKD for Java

- **Get Object**

```
AmazonS3 s3Client = new AmazonS3Client(new
ProfileCredentialsProvider());
```

```
S3Object object = s3Client.getObject(new
GetObjectRequest(bucketName, key));
```

- Put Object

```
AmazonS3 s3client = new AmazonS3Client(new
ProfileCredentialsProvider());
```

```
s3client.putObject(new  
PutObjectRequest(bucketName, keyName, file));
```

```
import java.io.*;
import java.util.*;

import com.amazonaws.services.s3.AmazonS3;
import com.amazonaws.services.s3.model.PutObjectRequest;
import com.amazonaws.services.s3.model.PutObjectResult;

public class UploadFileToS3 {
    private static String bucketName = "my-bucket";
    private static String fileName = "my-file.txt";
    private static String s3Endpoint = "s3.amazonaws.com";

    public static void main(String[] args) throws Exception {
        AmazonS3 s3 = new AmazonS3Client(new DefaultAWSCredentialsProviderChain());

        try {
            System.out.println("Uploading a new object to S3 from a file.");
            File file = new File(fileName);
            FileInputStream fis = new FileInputStream(file);
            UploadFileToS3.uploadFile(s3, bucketName, fileName, fis);
        } catch (AmazonS3Exception e) {
            System.out.println("Caught an AmazonS3Exception: " + e.getMessage());
            System.out.println("Error Message: " + e.getMessage());
            System.out.println("Error Code: " + e.getErrorCode());
            System.out.println("Error Type: " + e.getErrorType());
            System.out.println("Request ID: " + e.getRequestId());
        }
    }

    private static void uploadFile(AmazonS3 s3, String bucketName, String
        fileName, FileInputStream fis) throws Exception {
        try {
            System.out.println("Uploading file " + fileName + " to bucket " + bucketName);
            PutObjectRequest request = new PutObjectRequest(bucketName, fileName, fis);
            PutObjectResult result = s3.putObject(request);
            System.out.println("File uploaded successfully. ETag: " + result.getETag());
        } catch (Exception e) {
            System.out.println("Caught an AmazonS3Exception: " + e.getMessage());
            System.out.println("Error Message: " + e.getMessage());
            System.out.println("Error Code: " + e.getErrorCode());
            System.out.println("Error Type: " + e.getErrorType());
            System.out.println("Request ID: " + e.getRequestId());
        }
    }
}
```

From: <http://docs.aws.amazon.com/AmazonS3/latest/dev/RetrievingObjectUsingJava.html>

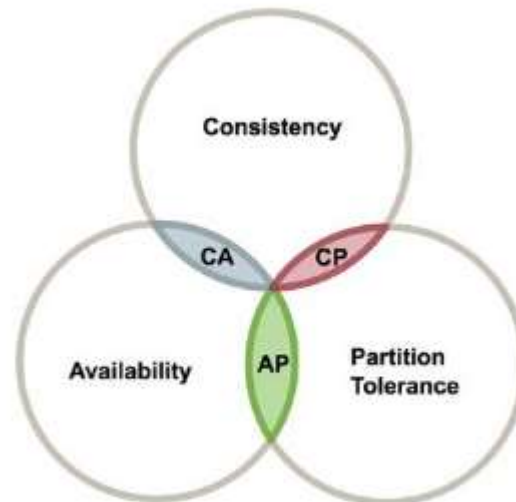


# CAP Theorem

In the presence of a network partition, strong consistency and high availability can not be achieved at the same time

## Amazon S3 (AP)

- High availability
- Eventual consistency
- Read-after-write consistency

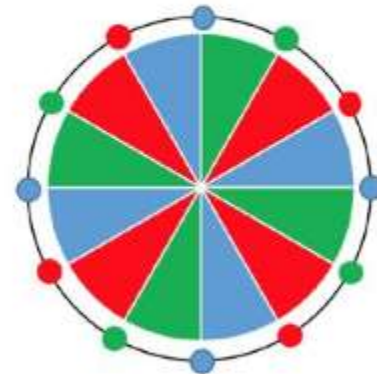


From: [http://berb.github.io/diploma-thesis/original/061\\_challenge.html](http://berb.github.io/diploma-thesis/original/061_challenge.html)

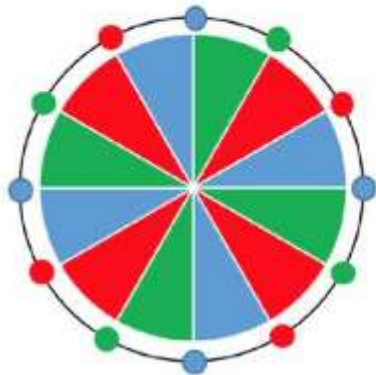
# Scalability – Consistent Hashing

## Avoid re-hashing everything

- Object key mapped to a point on the edge of ring
- Map machine to many random points on the edge of ring
- Machine responsible for arc before its nodes
- If a machine joins, it adds new nodes
- If a machine leaves, its range moves to after machine



## Availability – Replication



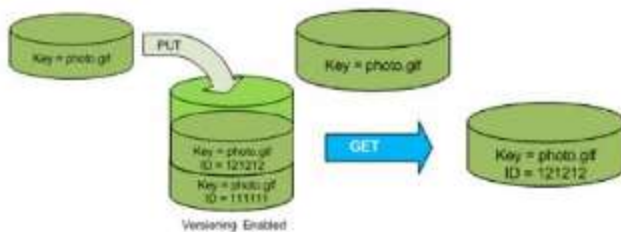
**Pick n next nodes (different machines!)**  
**n -- Replication factor**

**If one node fails, the other nodes still  
have the key information**

# Versioning – Basics

- keep multiple versions of an object in one bucket
- protect objects from unintended overwrites and deletions
- retrieve previous versions of them
- Write faster

- PUT doesn't overwrite: pushes version
- GET returns most recent version



- DELETE doesn't wipe
- GET will return not found



# Versioning

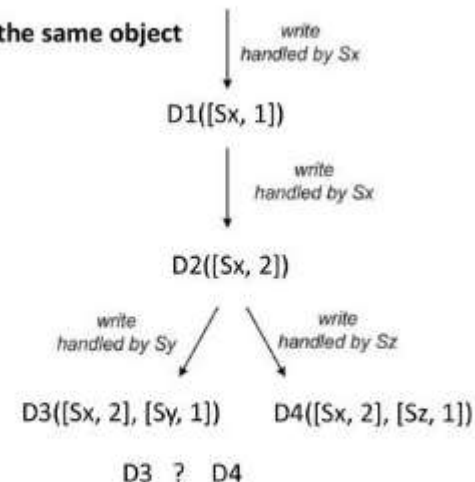
Using vector clock to capture causality between different versions of the same object

--D: object

--Sx,Sy,Sz: nodes

--[Sx, 1]: vector clock [node, count]

- One vector clock is associate with every version of every object
- Determine the versions of an object by examining their vector clocks
- If the counters on D1's clock is less than or equal to all of the nodes in the second clock, D1 is ancestor of the second, can be forgotten
- Otherwise, the two changes require reconciliation



Version evolution of an object over time

# Consistency

## Read-after-write consistency

- PUTS of new objects in S3 bucket
- Guarantee visibility of new data to all applications
- Allow you to retrieve objects immediately after creation



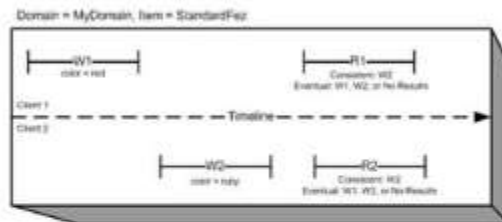
## Eventual consistency

- Overwrite PUTS and DELETES of objects in S3 bucket
- Weak consistency, to achieve high availability
- If no additional updates are made to a given data item, all reads to that item will eventually return the same value.



**Updates to a single key are atomic**

# Eventual Consistency



From: <http://docs.aws.amazon.com/AmazonS3/latest/dev/introduction.html>

**Both W1 (write 1) and W2 (write 2) complete before the start of R1 (read 1) and R2 (read 2)**

- **Consistent read**

**R1 and R2 both return color = ruby.**

- **Eventually consistent read**

**R1 and R2 might return color = red, color = ruby, or no results, depending on the amount of time that has elapsed.**

---

## Consistency Protocol

A consistency protocol similar to those used in quorum system is used to maintain consistency among its replicas.

- This protocol has two key configurable values:  $R$  and  $W$ .
  - $R$  is the minimum number of nodes that must participate in a successful read operation.
  - $W$  is the minimum number of nodes that must participate in a successful write operation.
- $N$  is the total number of nodes. Setting  $R$  and  $W$ .
- $R + W \leq N$



- END