

Cloud Data Storage: Amazon S3

INF 551

Wensheng Wu

Roadmap

- Overview
- Eventual consistency model
- Amazon S3

Amazon S3

- Simple Storage Service
- A type of cloud data storage
 - Data are stored in the cloud (Amazon's data center)

Data replication

- Multiple copies of the same data are stored
 - Among different compute nodes of the same center
 - And over different data centers
- Read
 - Can be served from any data center
- Write
 - Need to be propagated to all data centers & all nodes

Features

- Simple API, e.g., RESTful web service
 - Store and retrieve data over HTTP
- High availability
 - Low read & write latency
- Eventual consistency model

Roadmap

- Overview
- Eventual consistency
- Amazon S3

Strong consistency

- Traditionally, a database transaction needs to satisfy ACID properties
 - 'C' in ACID for strong consistency
- Consider a balance-transfer transaction
 - \$500 from account A to account B
 - After transfer, the total balance remains the same
 - & users do not get to see the inconsistent state (e.g., debit \$500 from A, not yet credit B)

ACID

- Atomicity: Either all or none of operations in the transaction should be executed
- **Consistency**: After transaction completes, the database is in a consistent state
- Isolation: allow concurrent execution of multiple transactions that do not interfere with each other
- Durability: can recover from failure

Eventual consistency

- If no new updates are made to the object, **eventually** all accesses to the object will return the last updated value.
- A form of **weak consistency**
 - Allow users to see the inconsistency state
 - Needed to achieve high availability (HA)

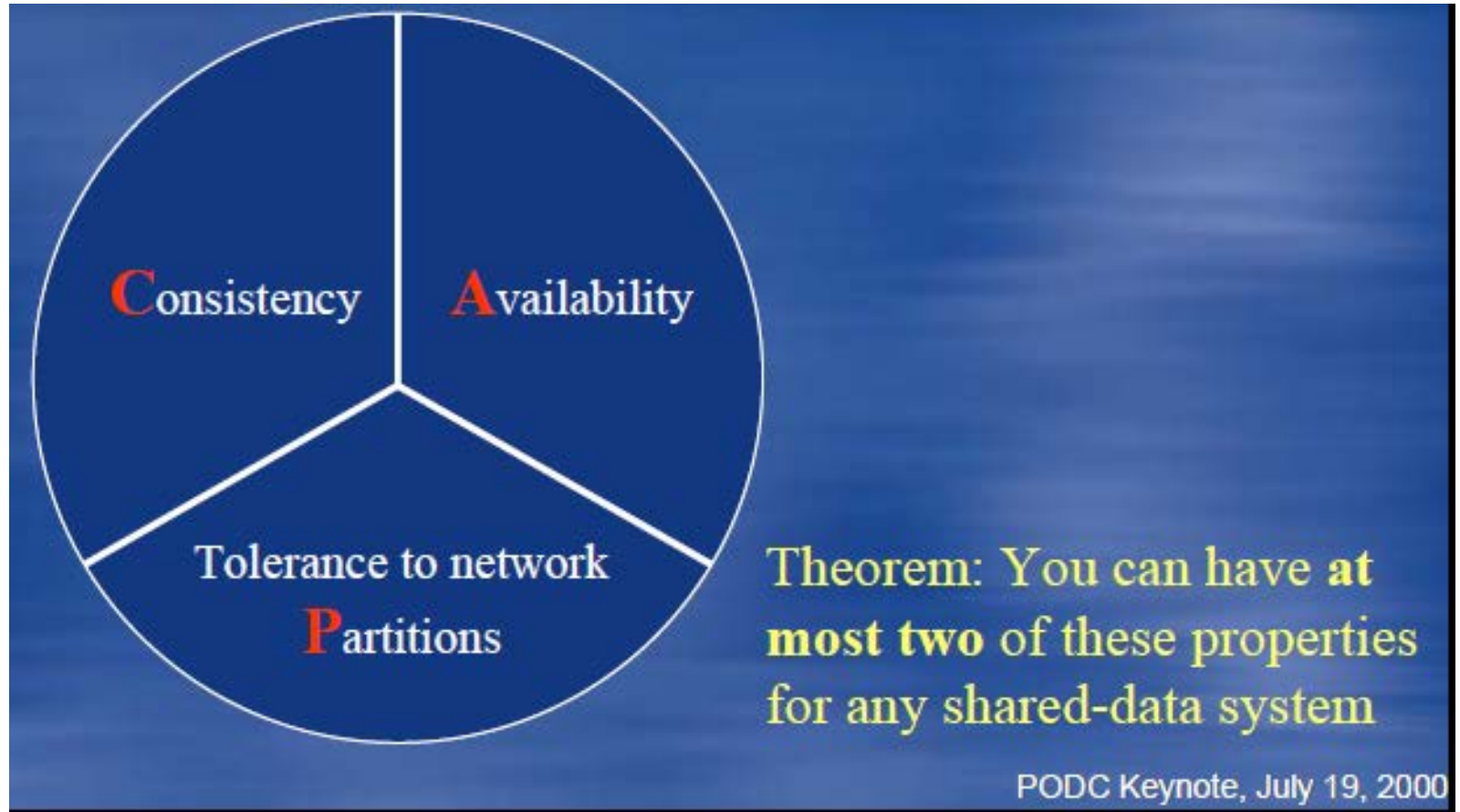
Inconsistency window

- Time between update acknowledged to user and eventual consistency achieved
 - i.e., updates propagated to all replicas
- Length of window determined by:
 - Communication delay
 - Load on the system
 - Number of replicas

Example

- DNS (domain name system) implements eventual consistency
 - DNS resolves www.usc.edu to 128.125.253.146
- Permissible for some DNS servers to have old data
 - As long as updates eventually propagated to them

CAP theorem



Consequence

- Consider a large distributed system
 - hence partitioning is a given (can not forfeit)
- Availability and (strong) consistency can not be achieved at the same time
 - => viability of eventual consistency model

Consequence

- Consider update made to an object O
- User A in LA may see the updated O right away
- But user B in NYC may see the old value of O
 - At least for a while

Eventual consistency model

- Acceptable to many applications
 - E.g., social media, cloud data storage, e-commerce
- Examples:
 - Amazon S3
 - Amazon DynamoDB (backbone of Amazon e-commerce and Web services)

Roadmap

- Overview
- Eventual consistency model
- Amazon S3

PRODUCTS & SERVICES

- [Amazon S3](#) >
- [Product Details](#) >
- [Storage Classes](#) >
- [Pricing](#) >
- [Getting Started](#) >
- [FAQs](#) >
- [Resources](#) >
- [Amazon S3 SLA](#) >

RELATED LINKS

- [AWS Management Console](#)
- [Documentation](#)
- [Release Notes](#)

Amazon S3

Amazon Simple Storage Service (Amazon S3), provides developers and IT teams with secure, durable, highly-scalable object storage. Amazon S3 is easy to use, with a simple web service interface to store and retrieve any amount of data from anywhere on the web. With Amazon S3, you pay only for the storage you actually use. There is no minimum fee and no setup cost.

Amazon S3 offers a range of storage classes designed for different use cases including Amazon S3 Standard for general-purpose storage of frequently accessed data, Amazon S3 Standard - Infrequent

Get Started with AWS Today

[Try Amazon S3 for Free](#)

AWS Free Tier includes 5GB storage, 20,000 Get Requests, and 2,000 Put Requests with Amazon S3.

[View AWS Free Tier Details »](#)

In Recent News

New: Amazon VPC

Data organization in S3

- Bucket
 - Container for objects
 - Has a unique name (across S3)
- Objects
 - Text files, images, etc.
- Folders
 - Grouping objects together
 - Can be nested

Create a bucket

Create bucket

1

Name and region

2

Set properties

3

Set permissions

4

Review

Name and region

Bucket name ⓘ

inf551-lab

Region

US West (N. California) ▼

Copy settings from an existing bucket

Select bucket (optional)

3 Buckets ▼

Create

Cancel

Next

Create bucket

✓ Name and region

✓ Set properties

3 Set permissions

4 Review

▶ Manage users

▼ Manage public permissions

Group ⓘ	Objects ⓘ	Object permissions ⓘ
Any authenticated AWS user	<input type="checkbox"/> Read <input type="checkbox"/> Write	<input type="checkbox"/> Read <input type="checkbox"/> Write
Everyone	<input checked="" type="checkbox"/> Read <input type="checkbox"/> Write	<input type="checkbox"/> Read <input type="checkbox"/> Write

Previous

Next

Allow public to
list the content
of bucket ←

Create bucket



Name and region



Set properties



Set permissions



Review

Name and region

Edit

Bucket name inf551-lab **Region** US West (N. California)

Properties

Edit

Versioning Disabled

Logging Disabled

Tagging 0 Tags

Permissions

Edit

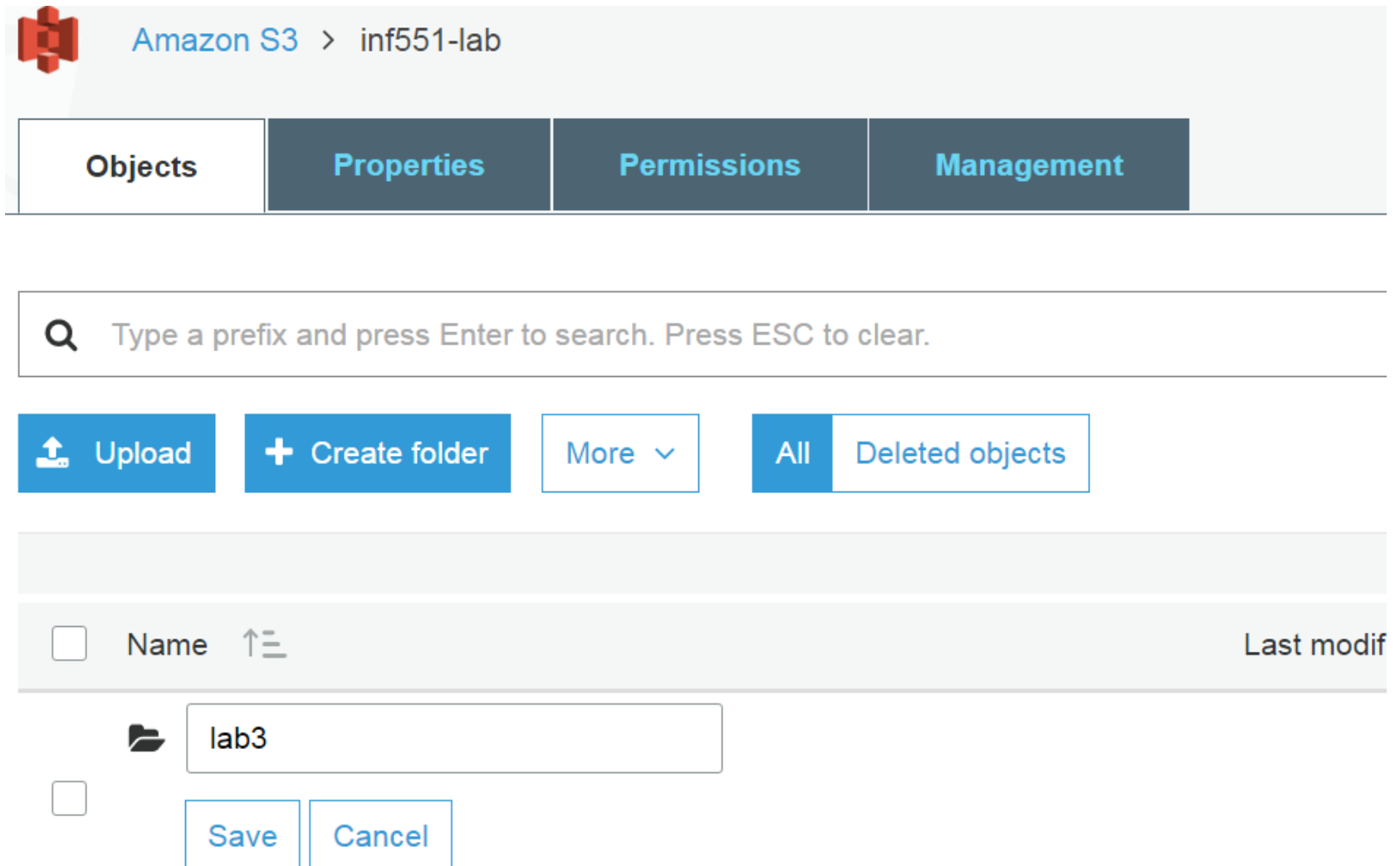
Users 1

Public permissions Enabled

Previous

Create bucket

Create a folder inside the bucket



The screenshot shows the Amazon S3 console interface. At the top, the breadcrumb navigation reads "Amazon S3 > inf551-lab". Below this is a tabbed interface with four tabs: "Objects", "Properties", "Permissions", and "Management". The "Objects" tab is currently selected. A search bar is located below the tabs, containing a magnifying glass icon and the text "Type a prefix and press Enter to search. Press ESC to clear." Below the search bar is a row of action buttons: "Upload" (with an upload icon), "Create folder" (with a plus icon), "More" (with a dropdown arrow), "All", and "Deleted objects". Below these buttons is a table header with columns "Name" (with a checkbox and a sort icon) and "Last modified". Below the table header, a "Create folder" dialog is open. It features a folder icon, a text input field containing the text "lab3", a checkbox, and two buttons: "Save" and "Cancel".

Amazon S3 > inf551-lab

Objects Properties Permissions Management

Q Type a prefix and press Enter to search. Press ESC to clear.

Upload + Create folder More ▾ All Deleted objects

☐ Name ↑ Last modified

☐ lab3

☐ Save Cancel

Upload a file to the folder

Upload

1 Select files

2 Set permissions

3 Set properties


4 Review

1 Files


Size: 606.3 KB

Target path: inf551-lab/lab3/

+ Add more files



Tulips.jpg
- 606.3 KB



Upload

Next

Upload

Select files

2

Set permissions

3

Set properties

4

Review

1 Files **Size:** 606.3 KB **Target path:** inf551-lab/lab3/

▸ Manage users

▼ Manage public permissions

Group	Objects	Object permissions
Any authenticated AWS user	<input type="checkbox"/> Read <input type="checkbox"/> Write	<input type="checkbox"/> Read <input type="checkbox"/> Write
Everyone	<input checked="" type="checkbox"/> Read <input type="checkbox"/> Write	<input type="checkbox"/> Read <input type="checkbox"/> Write

Upload

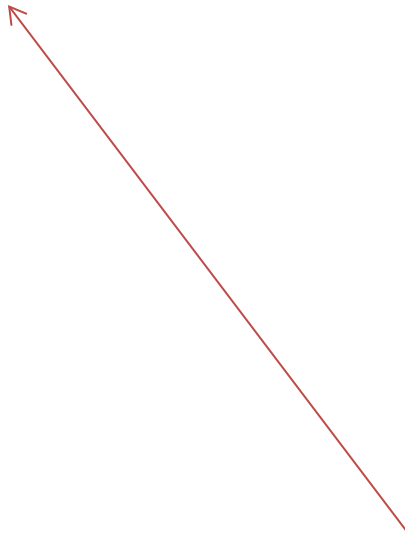
Previous

Next

Make it
accessible to
the public

<https://s3-us-west-1.amazonaws.com/inf551-lab/lab3/Tulips.jpg>

URL for the file



Tulips.jpg

Latest version ▼

Overview

Properties

Permissions

Open

Download

Download as

Make public

Owner
wuwens

Last activity
Mar 22, 2017 9:51:43 AM

Etag
fafa5efeaf3cbe3b23b2748d13e629a1

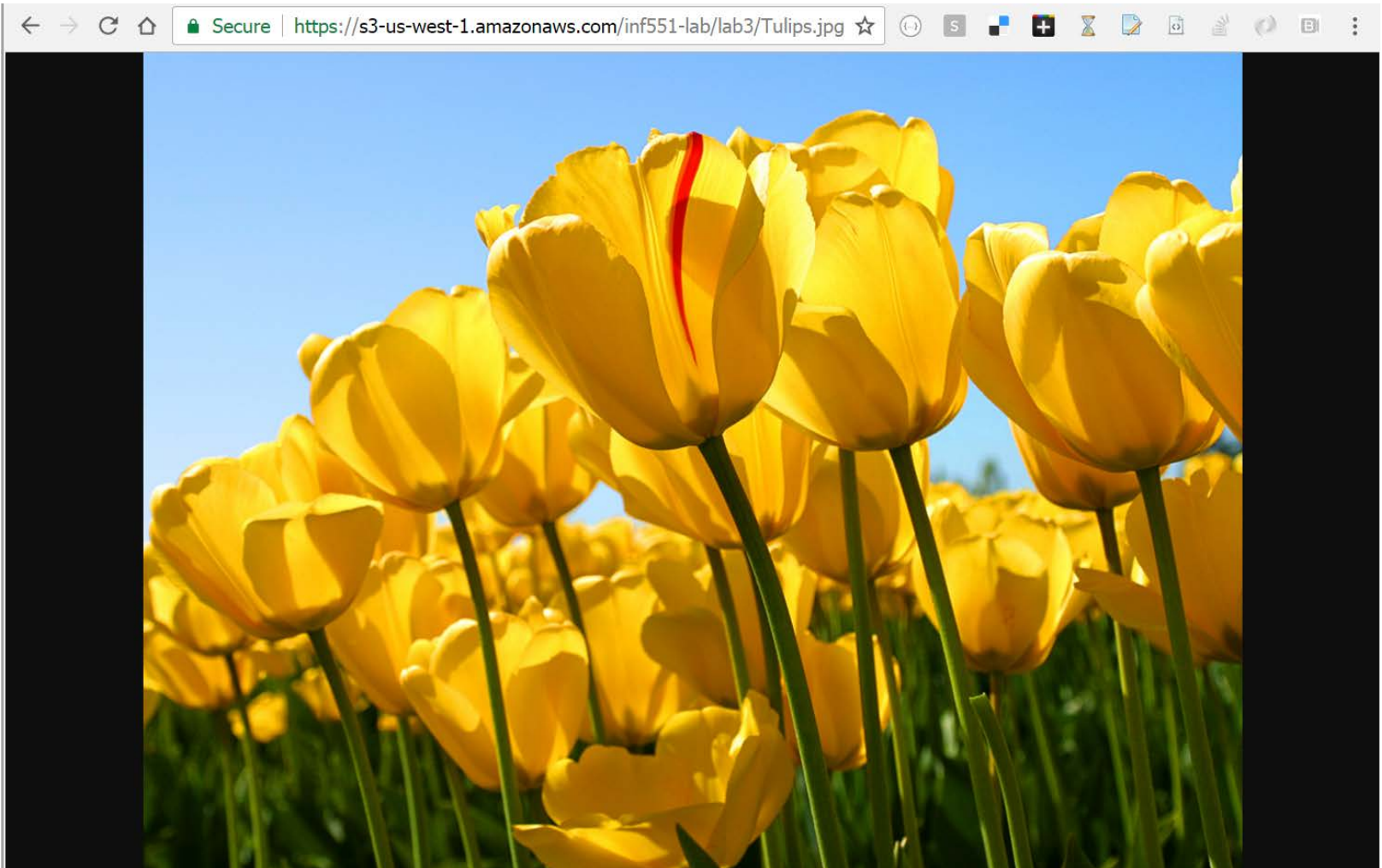
Storage class
Standard

Server side encryption
None

Size
606.3 KB

Link
<https://s3-us-west-1.amazonaws.com/inf551-lab/lab3/Tulips.jpg>

Open it in a browser



Resources

- Amazon S3:
<https://aws.amazon.com/documentation/s3/>
 - [Read this:](http://docs.aws.amazon.com/AmazonS3/latest/gsg/s3-gsg.pdf)
<http://docs.aws.amazon.com/AmazonS3/latest/gsg/s3-gsg.pdf>
- Amazon S3 Restful API
 - <http://docs.aws.amazon.com/AmazonS3/latest/API/APIRest.html>

References

- All things distributed by Werner Vogels (Amazon CTO)
 - <http://www.allthingsdistributed.com/2008/12/EventuallyConsistent.html>
- Towards robust distributed systems by Brewer
 - <https://people.eecs.berkeley.edu/~brewer/cs262b-2004/PODC-keynote.pdf>
 - CAP theorem proposed in this keynote