

AWS Academy Cloud Developing

Module 3: Developing Storage Solutions

Module 3: Developing Storage Solutions

Section 1: Introduction

Module objectives



At the end of this module, you should be able to do the following:

- Describe how Amazon Simple Storage Service (Amazon S3) can be used as a storage solution
- Identify features and components of Amazon S3
- Describe how to protect data in Amazon S3
- Describe the function of S3 object operations
- Explain how to manage access to Amazon S3 resources
- Develop with Amazon S3 by using the AWS software development kits (SDKs)

Module overview



Sections

1. Introduction
2. Introducing Amazon S3
3. Creating S3 buckets
4. Working with S3 objects
5. Protecting data and managing access to Amazon S3 resources

Lab

- Working with Amazon S3



Knowledge check

Café business requirement

Sofía has decided on a development environment, and she is ready to start building. She wants to create a proof-of-concept website for the café without exposing the site to the outside world yet.

```
mqtt-client > mqtt-client-bees.py
  `timestamp` timestamp NOT NULL DEFAULT CURRENT_TIMESTAMP,
  PRIMARY KEY (`id`) ENGINE=InnoDB DEFAULT CHARSET=utf8 COLLATE=utf8_general_ci
cursor.close()

sql = "INSERT INTO hive (dev_id, humidity, temperature) VALUES (%s, %s, %s)"

def on_connect(client, userdata, flags, rc):
    print("Connected with result code " + str(rc))

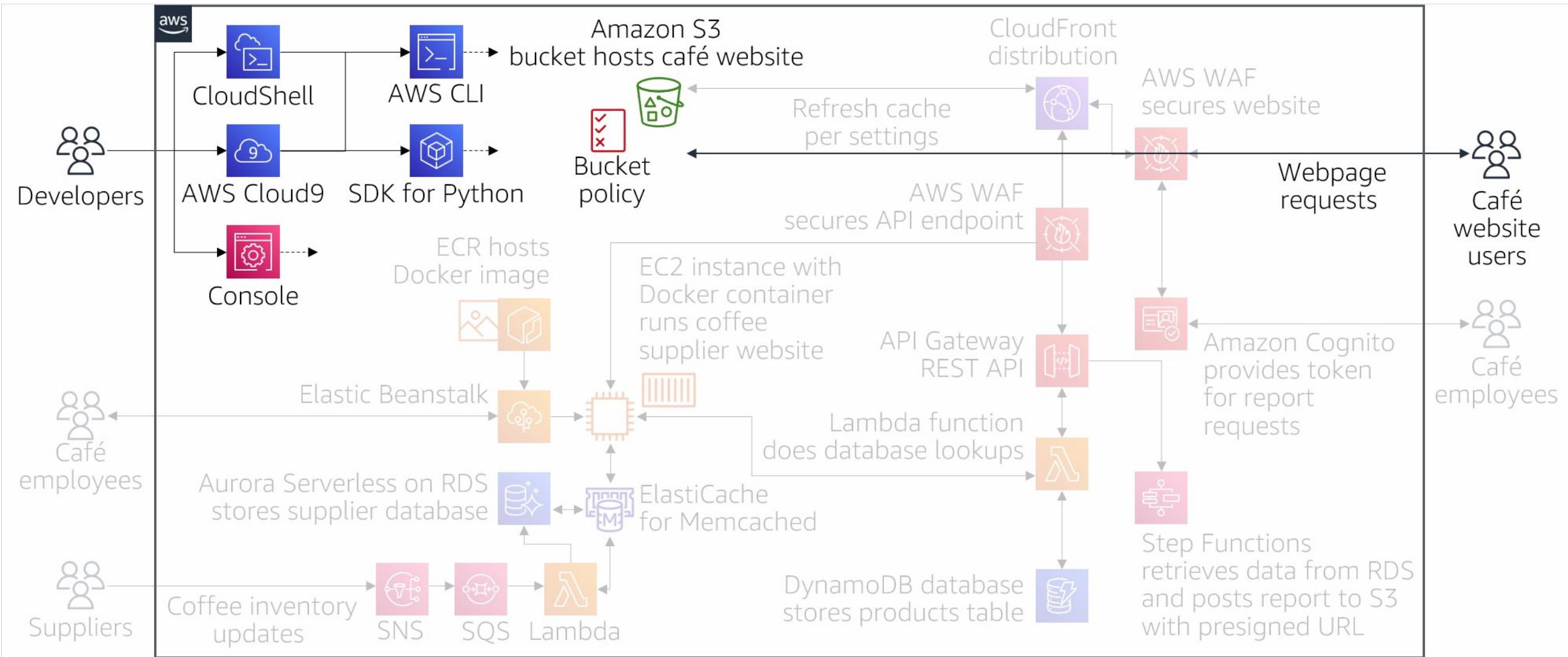
    client.subscribe("+/devices/+/up")

def on_message(client, userdata, msg):
    print(msg.topic + " " + str(msg.payload)+"\n")
    data = json.loads(msg.payload.decode('utf8').replace("'", ""))
    val = (data["dev_id"], data["payload_fields"]["humidity"], data["payload_fields"]["temperature"])
    connection = mysql.connector.connect(
        host="bees-mariaDB",
        user="bees",
        passwd=mysql_password,
        database="bees"
    )
    cursor = connection.cursor()
    cursor.execute(sql, val)
    connection.commit()
    print(cursor.rowcount, "record inserted.")
    cursor.close()
    connection.close()

client = mqtt.Client()
client.username_pw_set('hum_temp_app', password='ttn_password')
client.on_connect = on_connect
client.on_message = on_message
```



Amazon S3 as part of developing a cloud application



Module 3: Developing Storage Solutions

Section 2: Introducing Amazon S3



Amazon Simple
Storage Service
(Amazon S3)

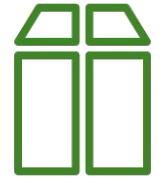
Object storage service that offers **scalability**,
data **availability**, **security**, and **performance**

- Designed for 99.999999999 percent (11 9s) of durability
- Provides easy-to-use management features
- Can respond to event triggers

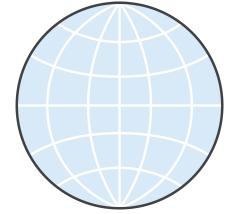
Amazon S3 use cases



Content storage
and distribution



Backup and
restore, and
archive



Data lakes and
big data
analytics



Disaster
recovery (DR)



Static website
hosting

Amazon S3 components



Bucket

`https://s3-<aws-region>.amazonaws.com/<bucket-name>/`

Each bucket is Regional and has a Region-specific endpoint in this format



Object

`https://s3-<aws-region>.amazonaws.com/<bucket-name>/<object-key>`

Object key example:
preview.mp4

Section 2 key takeaways



- Amazon S3 is an **object storage** service.
- Some uses for Amazon S3 include **content storage, backups, data lakes, DR, and static websites**.
- Objects in an S3 bucket can be referred to by their **URL**.
- The **key value** identifies the object in the bucket.

Module 3: Developing Storage Solutions

Section 3: Creating S3 buckets

S3 bucket names

- Bucket names must be **globally unique**.
- Additional rules to follow when you choose bucket names –
 - Use 3–63 characters.
 - Use only lowercase letters, numbers, and hyphens (-).
 - Do not use uppercase characters or underscores (_).



Amazon S3 bucket Regions

Deciding factors:

- Latency
- Cost
- Regulatory requirements



Region
s



New Region (coming
soon)



*As of February 8,
2021

Virtual-host-style URL

- Bucket name is part of the domain name in the URL.
- Structure: `http://<bucket-name>.s3-<aws-region>.amazonaws.com/<object-key>`
- Example: `http://DOC-EXAMPLE-BUCKET.s3.eu-west-1.amazonaws.com/cat.jpg`
- Useful for hosting a static website (*must be enabled*)
- Structure: `http://<bucket-name>.s3-website-<aws-region>.amazonaws.com`
- Example: `http://DOC-EXAMPLE-BUCKET.s3-website-eu-west-1.amazonaws.com`

Creating folder structure in buckets: Using prefixes



Bucket name:
DOC-EXAMPLE-BUCKET

Bucket objects:

- 2021/DOC-EXAMPLE-BUCKET
/english/john.txt
- 2021/DOC-EXAMPLE-BUCKET
/english/maria.txt
- 2021/DOC-EXAMPLE-BUCKET
/math/john.txt
- 2021/DOC-EXAMPLE-BUCKET
/math/maria.txt
- 2021/DOC-EXAMPLE-BUCKET
/summary.txt

Use **prefixes** to imply a folder structure
in an S3 bucket

GET
object

Specify prefix:
2021/DOC-EXAMPLE-BUCKET/math

Returns the following object keys:

- 2021/DOC-EXAMPLE-BUCKET/math/john.txt
- 2021/DOC-EXAMPLE-BUCKET/math/maria.txt

Section 3 key takeaways



- Amazon S3 bucket names are **globally unique**.
- Buckets are located in **Regions**, which affects **performance** and is subject to **regulatory requirements**.
- Objects in buckets are referenced through **virtual-host-style URLs**.
- Prefixes imply a **folder structure** in an S3 bucket.

Module 3: Developing Storage Solutions

Section 4: Working with S3 objects

Set of **key-value pairs** that provides additional information about the object

System-defined

- Information that Amazon S3 controls:
 - Object-creation date
 - Object size
 - Object version
- Information that you can modify:
 - Storage-class configuration
 - Server-side encryption

User-defined

- Information that you assign to the object
- **x-amz-meta** key followed by a custom name
- For example:
x-amz-meta-alt-name

Use the PUT object to upload entire objects to a bucket

- Should use **single upload** for objects **up to 5 GB** in a single PUT operation

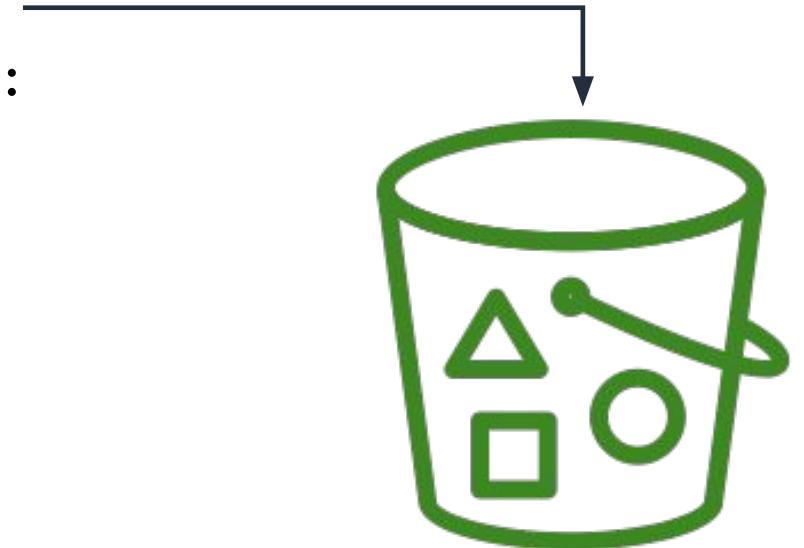


- Should use **multipart upload** for objects **over 100mb**
- **Must use multipart upload** for objects **over 5 GB**
 - Allows parallel uploading to improve throughput
 - Can resume uploads where it left off
 - Allows a maximum object size of 5 TB

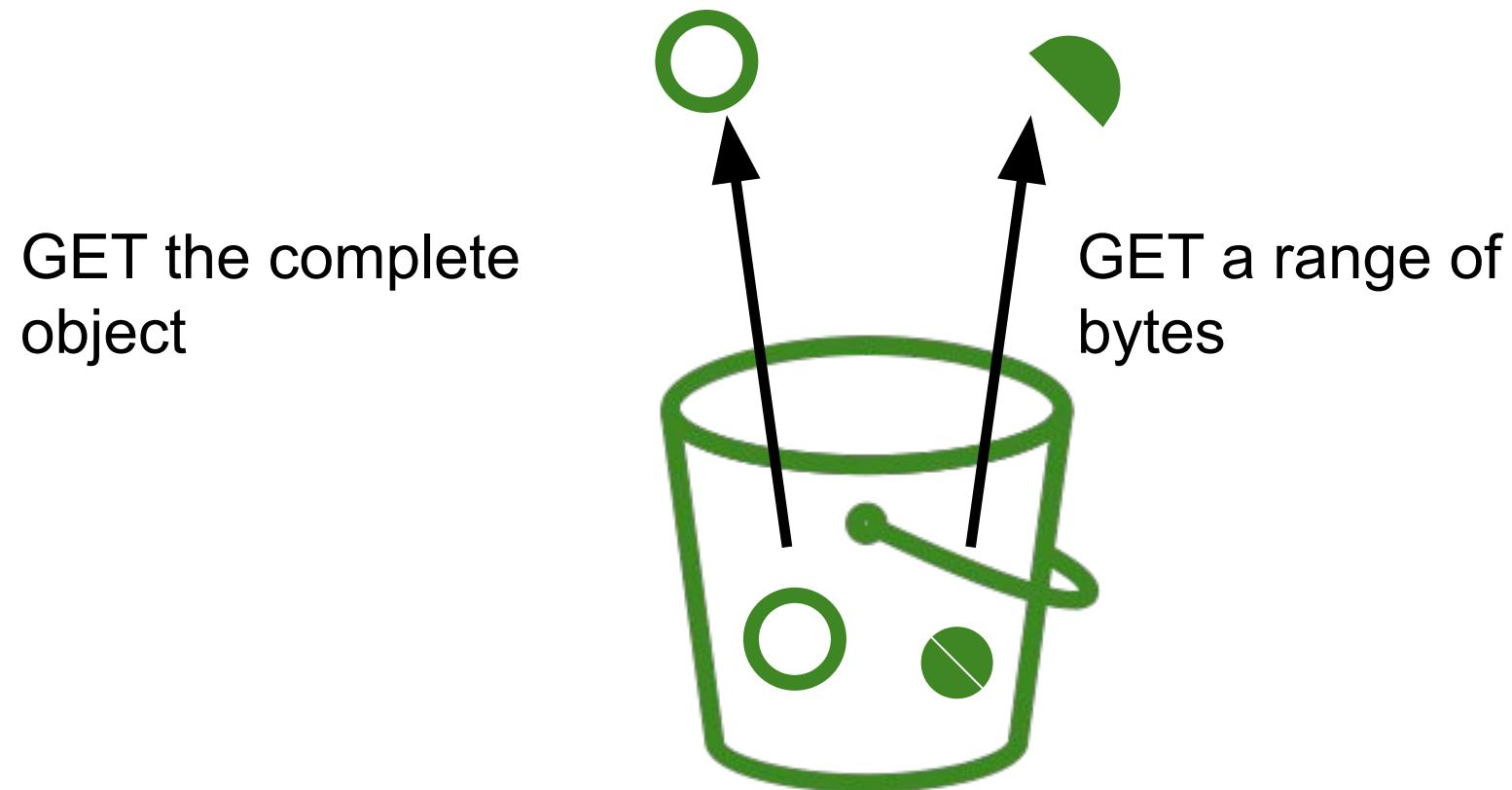
PUT object: Code

The following code PUTS core.css to the bucket

```
import boto3
S3API = boto3.client("s3", region_name="us-east-1")
bucket_name = "samplebucket"
filename = "/resources/website/core.css"
S3API.upload_file(filename, bucket_name, "core.css",
ExtraArgs={'ContentType': "text/css", "CacheControl": "max-age=0"})
```



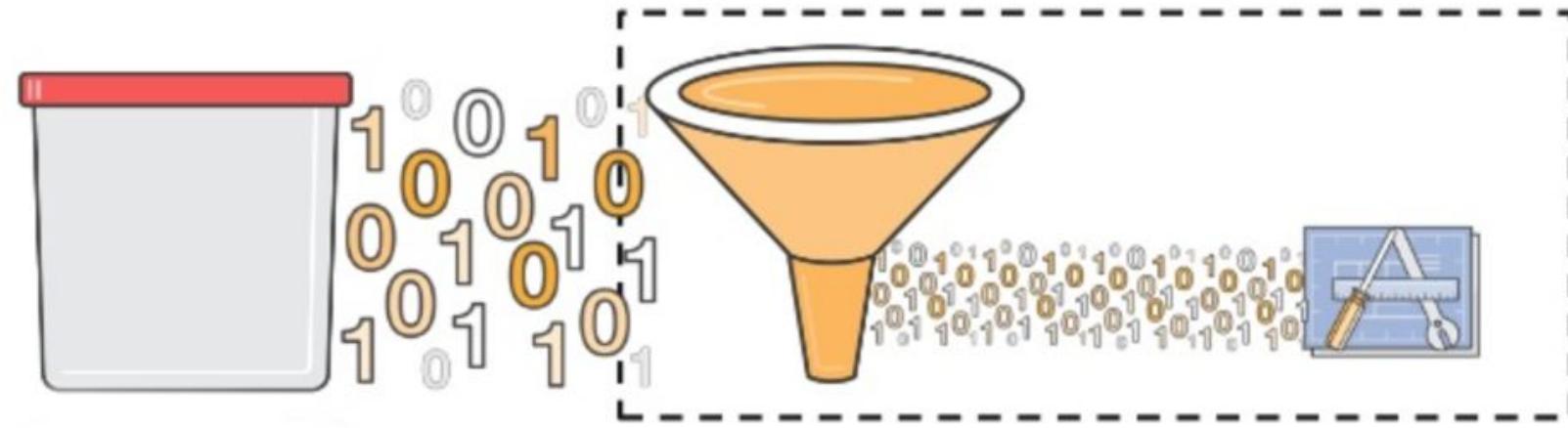
Use the GET object operation to retrieve objects from Amazon S3



SELECT object

Before

:



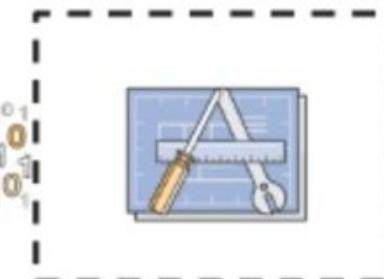
Amazon
S3

After:



Amazon
S3

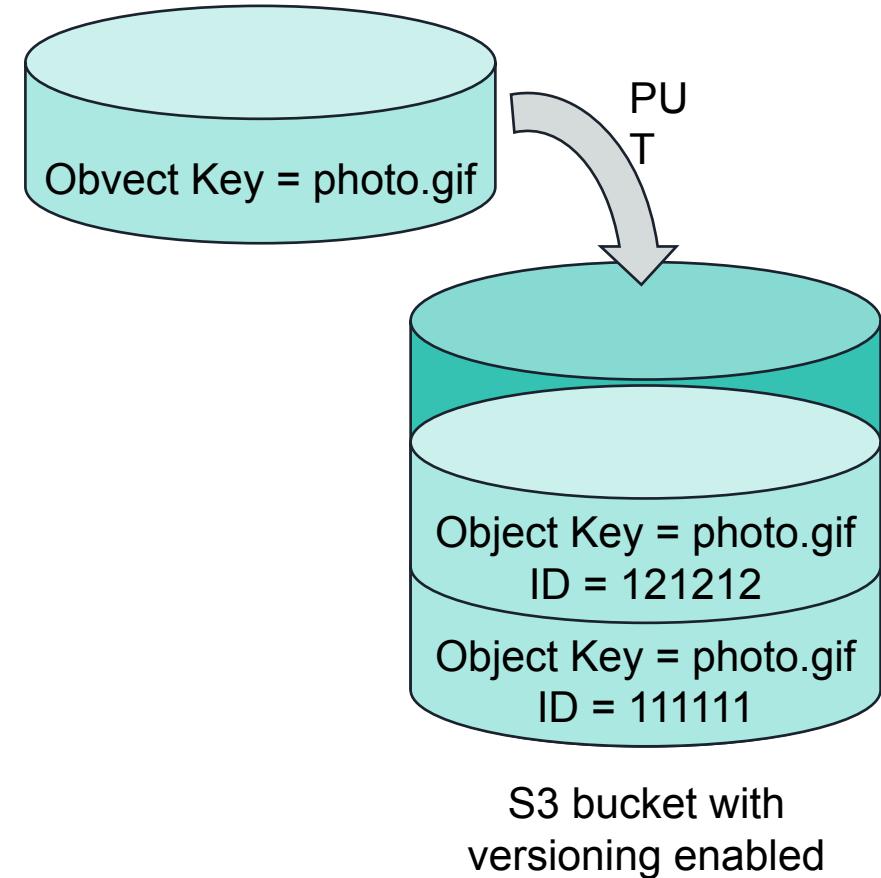
Up to 400 percent faster



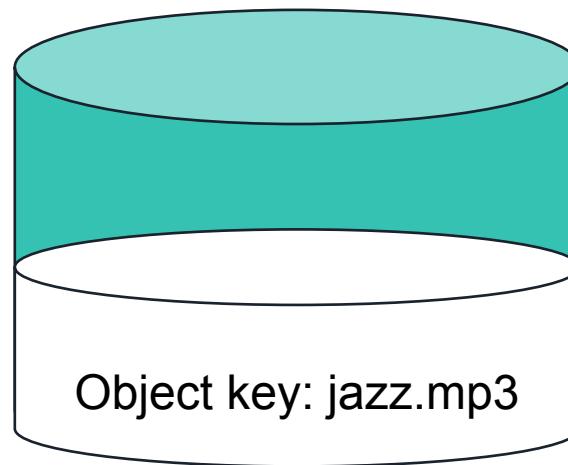
Up to 80 percent less
expensive

Versioning

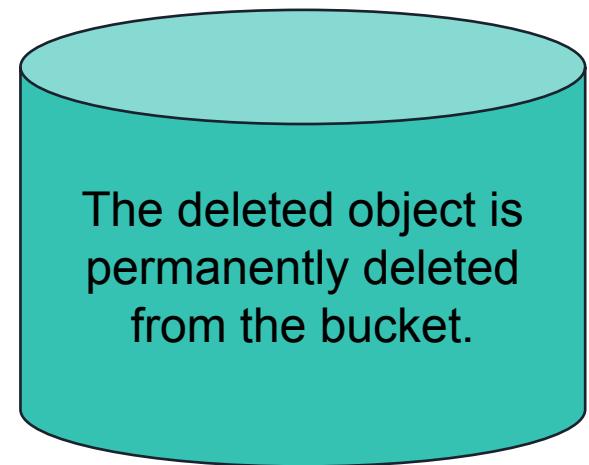
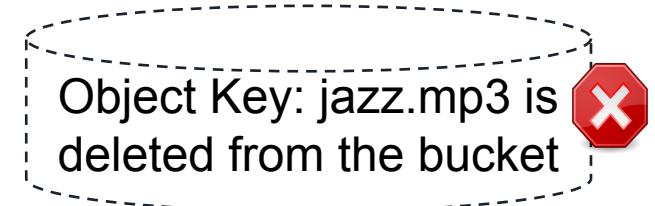
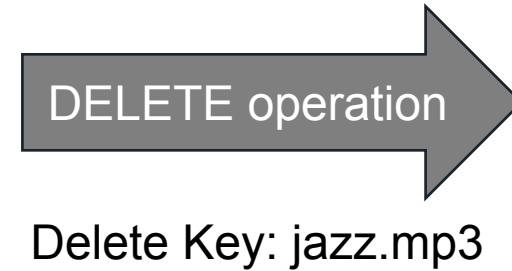
- It is a way to keep multiple variants of an object in the same bucket.
- It is a way to recover from unintended user actions and application failures.
- In versioning-enabled S3 buckets, each object has a version ID.
- After versioning is enabled, it can only be *suspended* (it can't be disabled).
- Versioned buckets support object locking.



DELETE object: Versioning disabled



S3 bucket with
versioning **disabled**



The deleted object is
permanently deleted
from the bucket.

DELETE Object: Versioning enabled



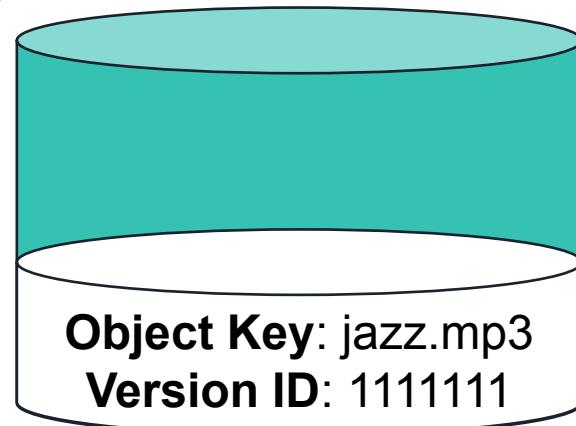
S3 bucket with
versioning **enabled**

Delete with **only the object key**



Object Key: jazz.mp3
Version ID: 2222222

Delete with **key and ID**



Section 4 key takeaways



- Objects in S3 buckets have **two types of metadata:**
 - System-defined metadata
 - User-defined metadata
- Amazon S3 has three operations:
 - **PUT**
 - **GET**
 - **DELETE**
- **S3 Select** is a powerful tool to query data in place.
- **S3 bucket versioning** protects objects.

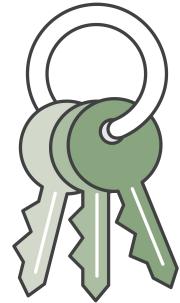
Module 3: Developing Storage Solutions

Section 5: Protecting data and managing access to Amazon S3 resources



Securing data in transit

- SSL/TLS-encrypted endpoints with HTTPS
- Client-side encryption of data before transmission



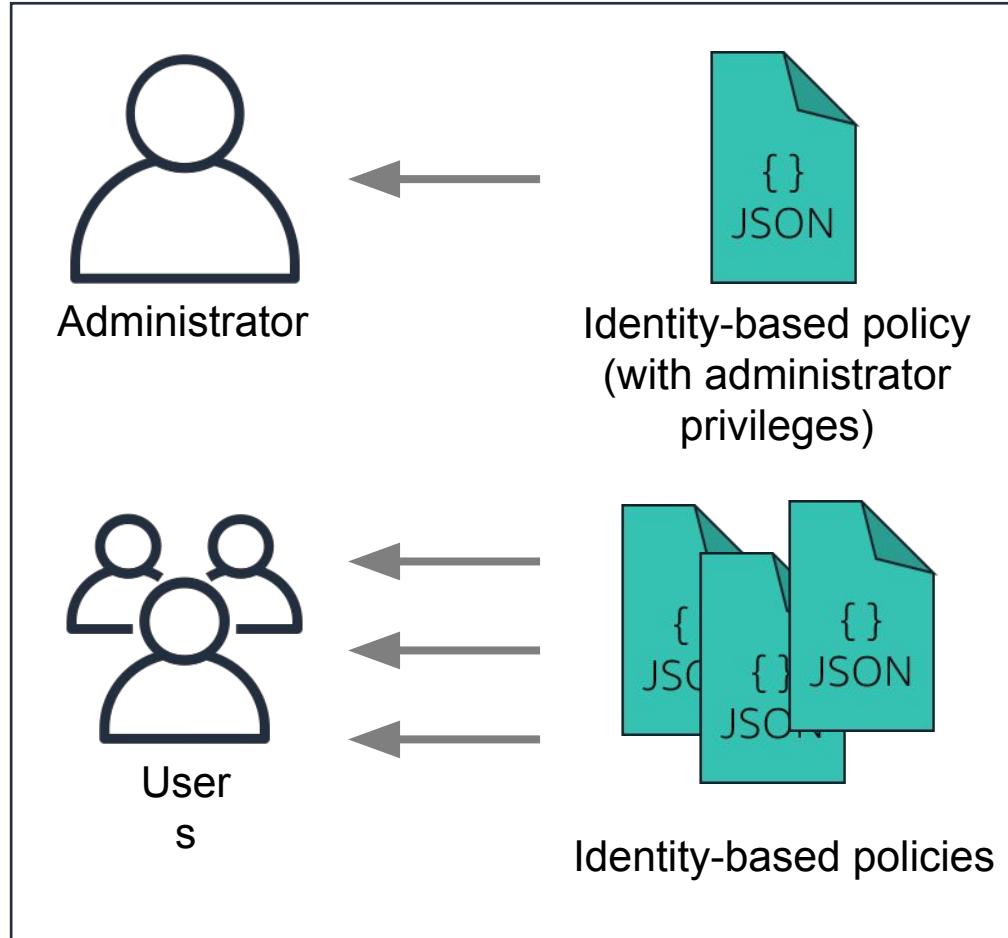
Securing data at rest

- Client-side encryption
- Server-side encryption
 - Amazon S3-managed keys (SSE-S3)
 - AWS KMS-managed keys (SSE-KMS)
 - Customer-provided keys (SSE-C)

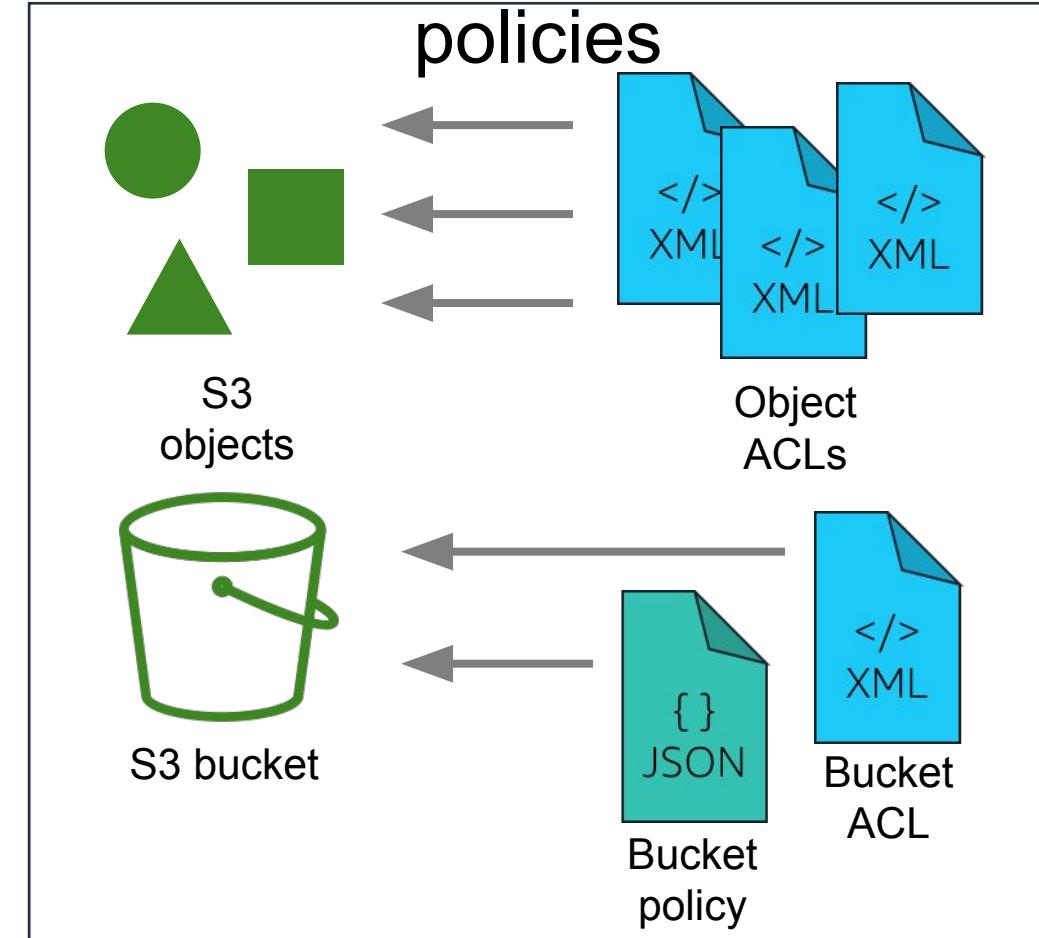
Identity-based policies and resource-based policies



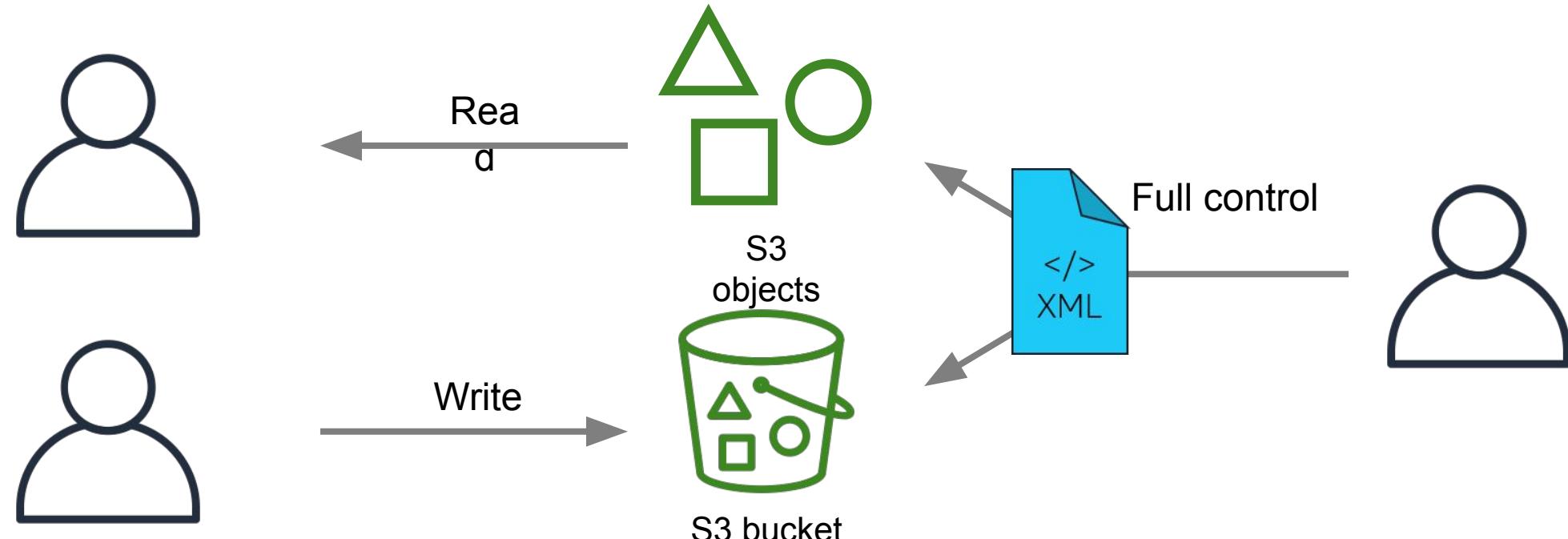
Identity-based policies



Resource-based policies



Access control lists (ACLs)



- Resource-based access policy to manage access at the object level or bucket level
- Use to grant basic read/write permissions to other AWS accounts

Bucket policies

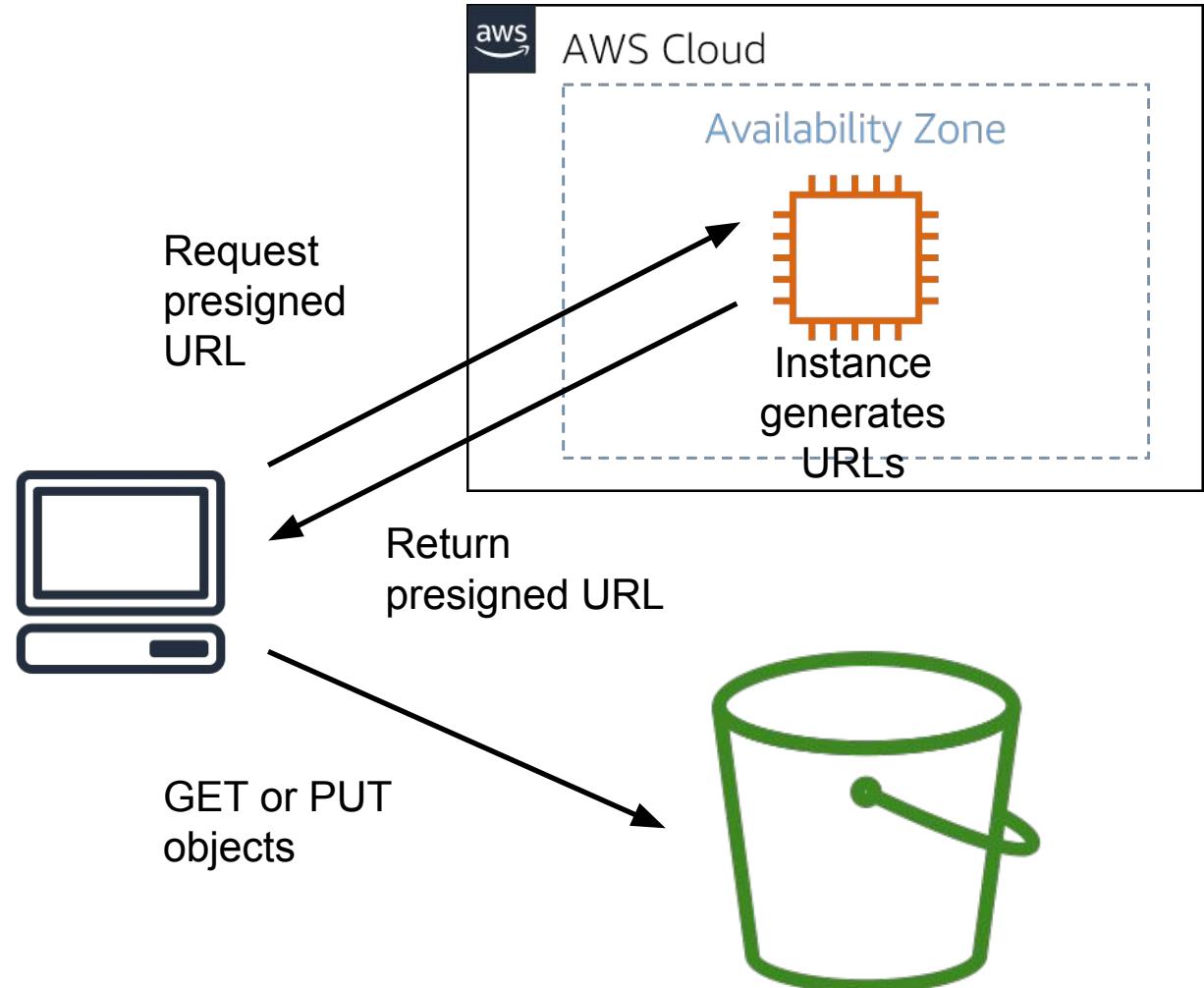


An IAM policy language option that grants granular permissions to Amazon S3 resources

```
{  
  "Version": "2012-10-17",  
  "Statement": [  
    {  
      "Effect": "Allow",  
      "Principal": "*",  
      "Action": ["s3:GetObject"],  
      "Resource": ["arn:aws:s3:::DOC-EXAMPLE-BUCKET/*"]  
    }  
  ]  
}
```

Presigned URLs

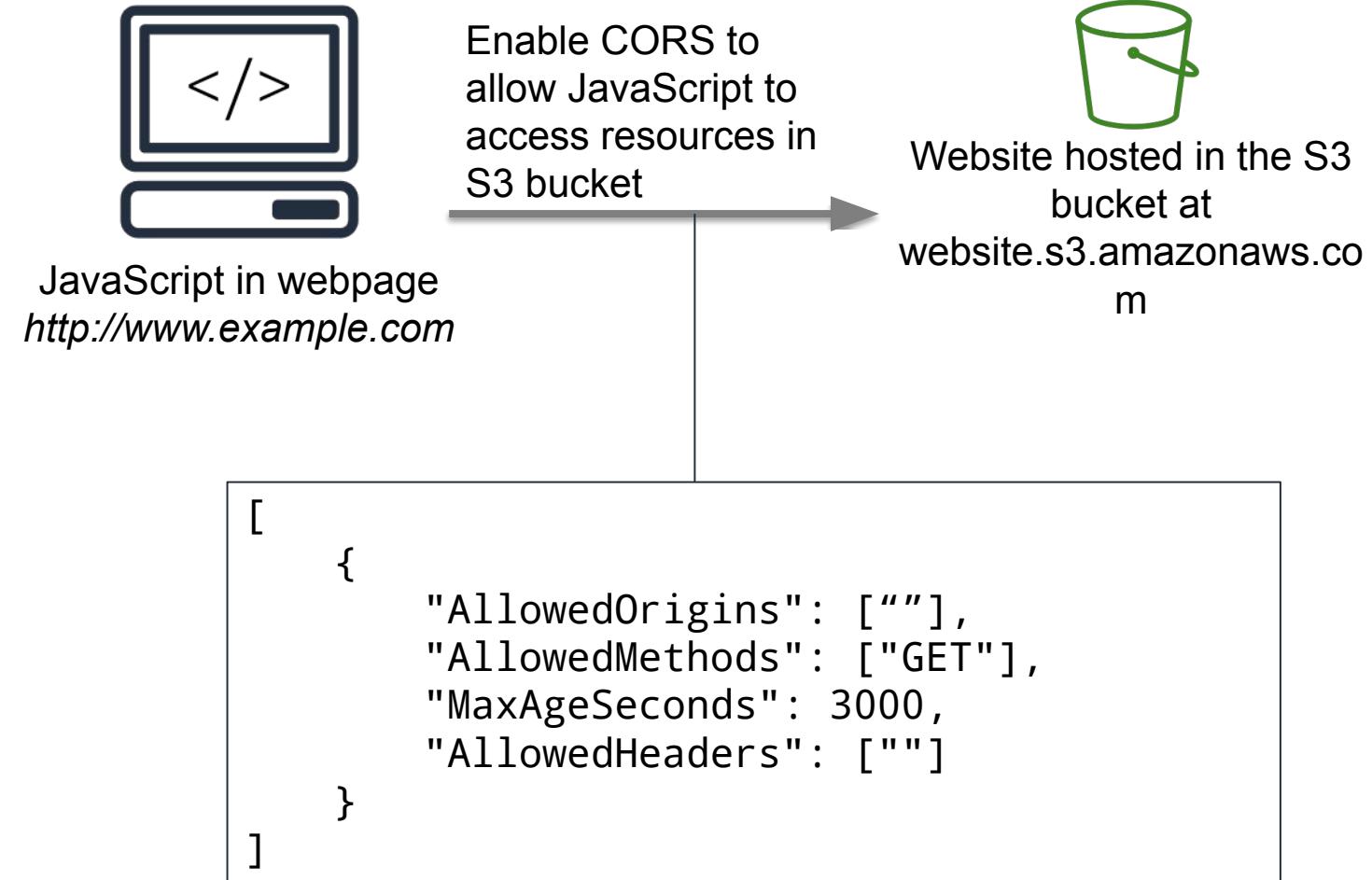
- Provide access to PUT or GET objects without granting permissions to perform any other actions
- Use the permissions of the user who creates the URL
- Provide security credentials, a bucket name, an object key, an HTTP method, and an expiration date and time
- Are only valid until the expiration time (maximum of 1 week)



Cross-origin resource sharing (CORS)



Cross-origin resource sharing (CORS) defines a way for client web applications that are loaded in one domain to interact with resources that are in a different domain.

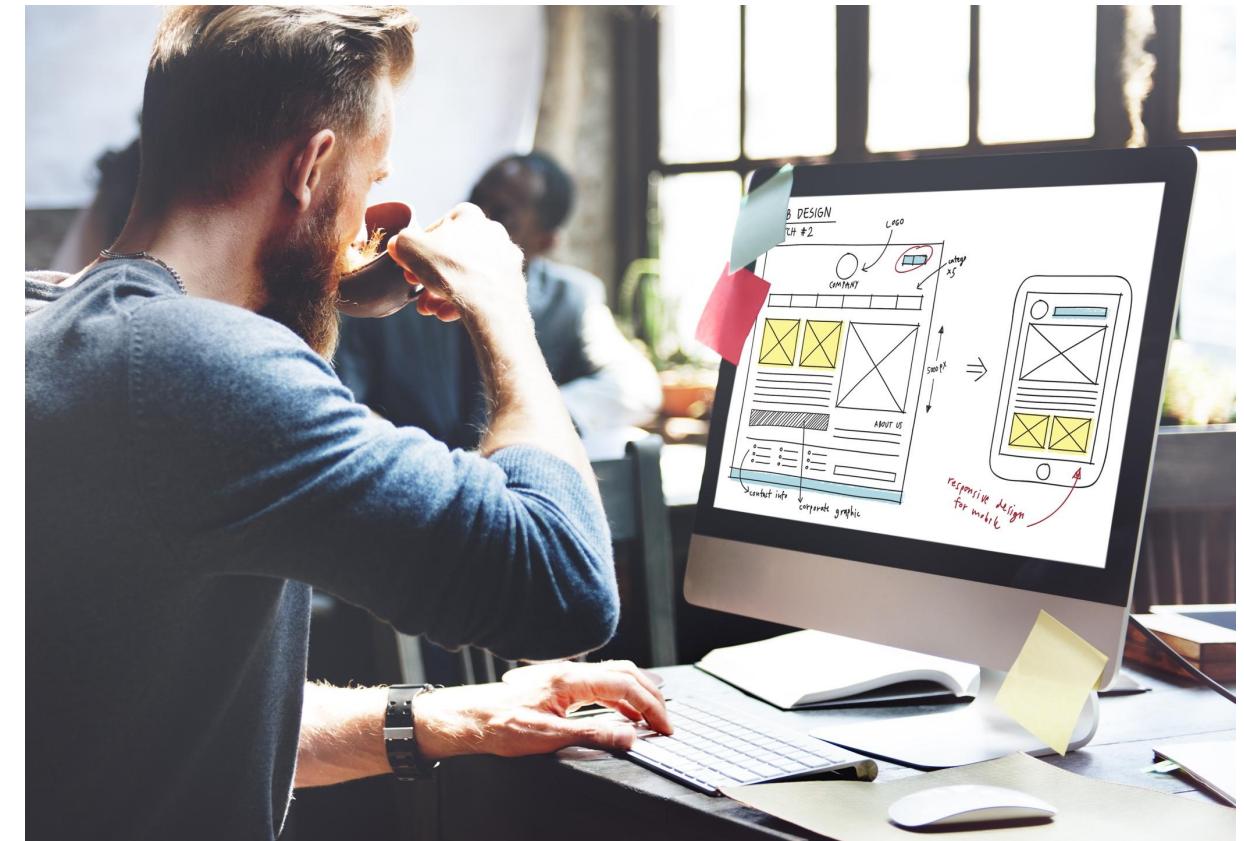


Section 5 key takeaways



- S3 buckets can be encrypted.
- Amazon S3 has two types of policies for bucket access:
 - Identity-based policies
 - Resource-based policies

Lab 3.1: Working with Amazon S3



Lab: Tasks



1. Connecting to the AWS Cloud9 IDE and configuring the environment
2. Creating an S3 bucket by using the AWS CLI
3. Setting a bucket policy on the bucket by using the SDK for Python
4. Uploading objects to the bucket to create the website
5. Testing access to the website
6. Analyzing the website code



~ 60 minutes



Begin Lab 3.1: Working with Amazon S3

Lab debrief: Key takeaways



Module 3: Developing Storage Solutions

Module wrap-up

In summary, in this module, you learned how to do the following:

- Describe how Amazon S3 can be used as a storage solution
- Identify features and components of Amazon S3
- Describe two ways to protect data in Amazon S3
- Describe the function of S3 object operations
- Explain how to manage access to Amazon S3 resources
- Develop with Amazon S3 by using the AWS SDKs

Complete the knowledge check



Sample exam question

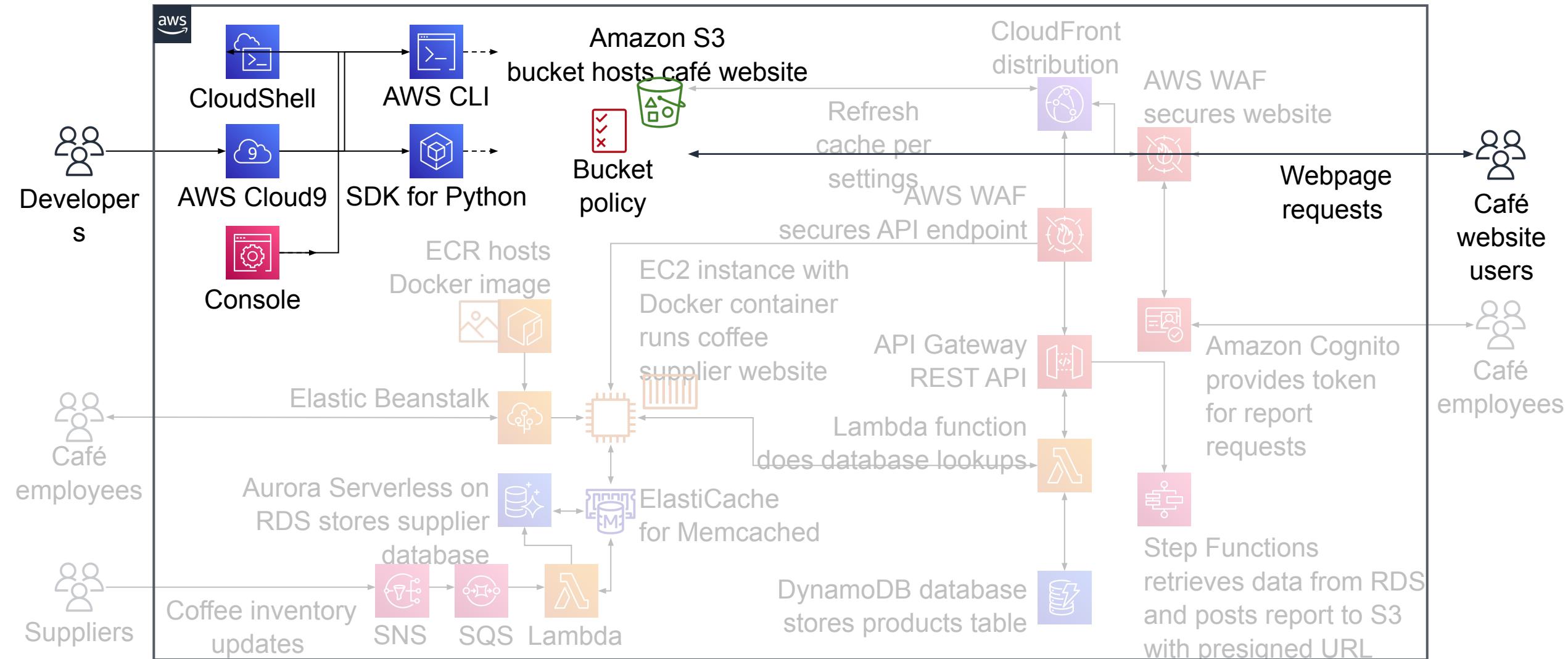
Company salespeople upload their sales figures daily. A solutions architect needs a durable storage solution for these documents that also protects against users accidentally deleting important documents.

Which action will protect against unintended user actions?

- A. Store data in an EBS volume and create snapshots once a week.
- B. Store data in an S3 bucket and enable versioning.
- C. Store data in two S3 buckets in different AWS Regions.
- D. Store data on EC2 instance storage.

Thank you

Amazon S3 as part of developing a cloud application



Presigned URLs

- Provide access to PUT or GET objects without granting permissions to perform any other actions
- Use the permissions of the user who creates the URL
- Provide security credentials, a bucket name, an object key, an HTTP method, and an expiration date and time
- Are only valid until the expiration time (maximum of 1 week)

