CS 240
Computer Systems

#19: Data Stores and Cloud Architectures

Oct 26, 2023 · G Carl Evans

Cloud Object Storage

Instead of using local file storage, large data storage in the cloud-based systems are commonly stored as **"objects"**. These objects (files) are organized into _____:

Public Cloud Providers	Private Cloud Solutions

Example: AWS

Amazon AWS S3 CreateBucket REST API
https://docs.aws.amazon.com/AmazonS3/latest/API/API_CreateBucket.html

```
PUT / HTTP/1.1
Host: Bucket.s3.amazonaws.com
x-amz-acl: ACL
x-amz-grant-read: GrantRead : UserList
x-amz-grant-write: GrantWrite : UserList
x-amz-grant-full-control: GrantFullControl : UserList
x-amz-grant-read-acp: GrantReadACP : UserList
x-amz-grant-write-acp: GrantWriteACP : UserList
[...]
```

Bucket:	Name of the bucket. *[Required]*
ACL:	The canned Access Control to apply to the bucket. `private` | `public-read` | `public-read-write` | `authenticated-read`
UserList:	You specify each grantee (user) as a type=value pair, where the type is one of the following: **id** – if the value specified is the canonical user ID of an AWS account **uri** – if you are granting permissions to a predefined group **emailAddress** – if the value specified is the email address of an AWS account Ex: `x-amz-grant-read: id="11112222333",id="444455556666"`
ACP:	**x-amz-grant-read** grants permission for the file itself; **x-amz-grant-read-acp** grants permissions for the access control policies.

+ Lots of Language-level Libraries

Private Cloud Solutions:

MinIO: https://docs.min.io/docs/python-client-api-reference.html#make_bucket
OpenStack/Swift:
https://docs.openstack.org/api-ref/object-store/index.html?expanded=create-container-detail#create-container
Adding files to storage are also HTTP endpoints:

Amazon AWS S3 PutObject REST API
https://docs.aws.amazon.com/AmazonS3/latest/API/API_PutObject.html

```
PUT /Key HTTP/1.1
Host: Bucket.s3.amazonaws.com
x-amz-tagging: Tagging
x-amz-acl: ACL
x-amz-grant-full-control: GrantFullControl : UserList
x-amz-grant-read: GrantRead : UserList
x-amz-grant-read-acp: GrantReadACP : UserList
x-amz-grant-write-acp: GrantWriteACP : UserList
[...]
Content-Length: ContentLength

Body
```

Q: Is there a directory structure similar to traditional file systems?

Cloud Object Storage in Python

Instead of using file storage on disk, object storage in the cloud provides us access to a file-system-like interface without the need for all programs to be running on the same computer!

Reading a file in Python:

18/local.py

```
1 f = open("settings.json", "r")
2 print(f.read())
```

Initializing an S3 connection:

18/s3.py

```
1 import boto3
2 s3 = boto3.client('s3', [...])
```

Reading Data from S3:

18/s3.py

```
4 # Reading data from S3:
5 obj = s3.get_object(Bucket="cs240", Key="session_data")
6 f = obj["Body"]
```

- The **f** variable in local.py and s3.py are both _____!
- **Key Idea:**

18/s3.py

```
 8 print("== S3 Response ==")
 9 print(obj)
10 print()
11
12 print("== Contents ==")
13 print(f.read().decode("utf-8"))
14 print()
```

Writing Data to S3 is one line just like files to disk:

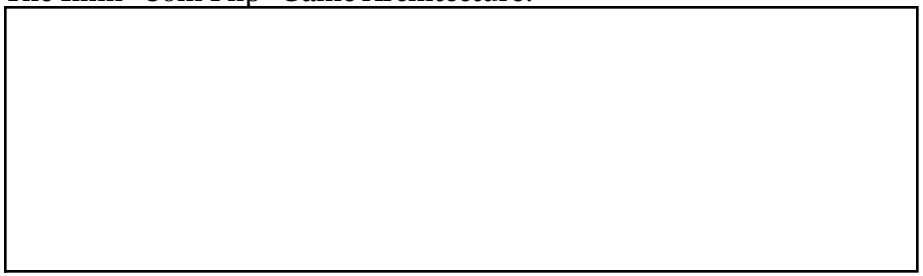
18/s3-put.py

```
14 # Add an object as a string:
15 s3.put_object(Bucket="cs240", Key="session_data",
                  Body=json.dumps({"hello": "world"}))
16
17 # Upload a file:
18 s3.upload_file("cs240.png", Bucket="cs240",
                             Key="profile-picture.png")
```

Common Data Storage Options:

Scope	Data Storage	Interface	Technology
Local	**Variables**	Lang. Feature	All Programs!
	Files	File or File-like	open / fopen
Remote (Cloud)	**Object Storage**		S3 / MinIO
	Key-Value DB	Dictionary API set / get	redis
	Document DB	JSON find / update / insert	mongodb
	Relational DB	SQL	mysql / postgres
	Special Purpose	Objects	neo4j

The Illini "Coin Flip" Game Architecture:



Using Local Variable Storage for the Wallet Service:

19/wallet-service-local/app.py

```
 5 d = {}
 …
 8 def createUser(sessionID):
 9   d[sessionID] = 100
 …
30   d[sessionID] = userData["amount"]
```

Using a Key-Value Store (redits) for the Wallet Service:

19/wallet-service-kvstore/app.py

```
 5 kvStore = redis.Redis()
 …
 9 def createUser(sessionID):
10   kvStore.set(sessionID, 100)
 …
32   kvStore.set(sessionID, userData["amount"])
```

Using a Document Database (mongo) for the Wallet Service:

19/wallet-service-documentdb/app.py

```
 5 mongo = MongoClient(port=27017)
 6 db = mongo["IlliniCoin"]["users"]
 …
10 def createUser(sessionID):
11   userData = {"amount": 100, "sessionID": sessionID}
12   r = db.insert_one(userData)
 …
32   db.update_one({"sessionID": sessionID}, { "$set":
   {"amount": userData["amount"]}})
```