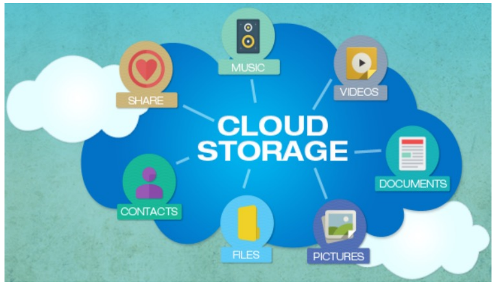
Week 4 AWS S3 and DynamoDB

Dr Zhi Zhang

Overview

- Amazon S3 (Simple Storage Service)
- Amazon DynamoDB
- Mid-semester Test

Cloud storage

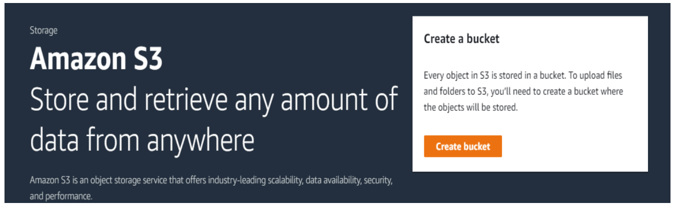


- Cloud storage, provided by a third-party cloud provider allows individuals and organizations to store and access data over the internet through remote servers.
- Examples: Dropbox, Google drive, iCloud, Amazon S3

Amazon S3 (Simple Storage Service)

- It is a popular and widely used cloud storage service provided by AWS
    - It allows users to store and retrieve any amount of data at any time over the internet
- Bucket
- Object

Create bucket



Storage

Amazon S3

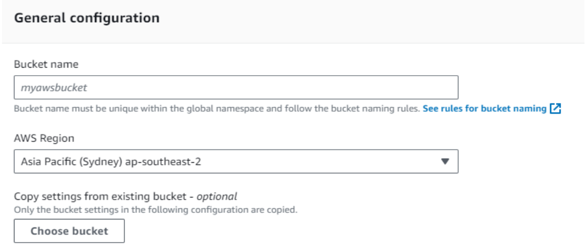
Store and retrieve any amount of data from anywhere

Amazon S3 is an object storage service that offers industry-leading scalability, data availability, security, and performance.

Create a bucket

Every object in S3 is stored in a bucket. To upload files and folders to S3, you'll need to create a bucket where the objects will be stored.

Create Bucket



General configuration

Bucket name

myawsbucket

Bucket name must be unique within the global namespace and follow the bucket naming rules. See rules for bucket naming

AWS Region

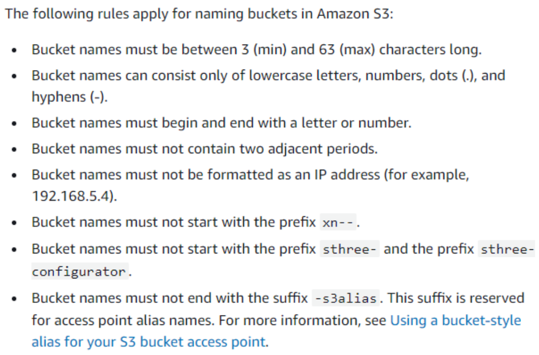
Asia Pacific (Sydney) ap-southeast-2

Copy settings from existing bucket - *optional*

Only the bucket settings in the following configuration are copied.

- Note:
  - Bucket name must be unique within the global namespace.
  - It must follow the bucket naming rules.

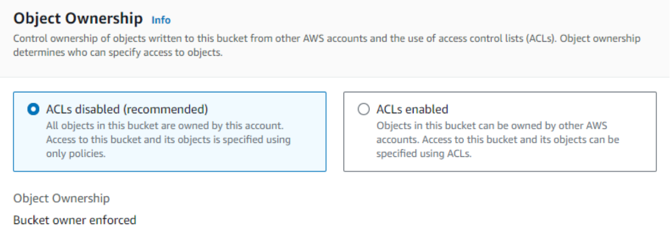
Rules for bucket naming



The following rules apply for naming buckets in Amazon S3:

- Bucket names must be between 3 (min) and 63 (max) characters long.
- Bucket names can consist only of lowercase letters, numbers, dots (.), and hyphens (-).
- Bucket names must begin and end with a letter or number.
- Bucket names must not contain two adjacent periods.
- Bucket names must not be formatted as an IP address (for example, 192.168.5.4).
- Bucket names must not start with the prefix `xn--`.
- Bucket names must not start with the prefix `sthree-` and the prefix `sthree-configurator`.
- Bucket names must not end with the suffix `-s3alias`. This suffix is reserved for access point alias names. For more information, see Using a bucket-style alias for your S3 bucket access point.

Object ownership



Object Ownership Info

Control ownership of objects written to this bucket from other AWS accounts and the use of access control lists (ACLs). Object ownership determines who can specify access to objects.

● ACLs disabled (recommended)
All objects in this bucket are owned by this account. Access to this bucket and its objects is specified using only policies.

○ ACLs enabled
Objects in this bucket can be owned by other AWS accounts. Access to this bucket and its objects can be specified using ACLs.

Object Ownership

Bucket owner enforced

- What is an object?

Object

- It is an individual unit of data stored in a bucket
- Can be a file of any type:
  - documents, images, videos, etc
- It contains both data and metadata:
  - Data refers to file contents. Metadata include file attributes.
  - e.g., a file called sunset.jpg is uploaded into a bucket.

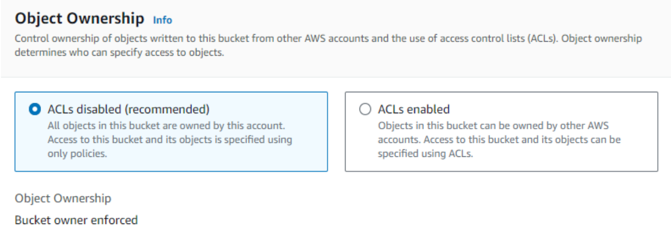
```
# Bucket and object information
bucket_name = 'my-bucket'
object_key = 'sunset.jpg'

# Metadata as key-value pairs
metadata = {
        'Content-Type': 'image/jpeg',
        'Author': 'John Doe',
        'Description': 'A beautiful sunset',
        'Location': 'New York',
        'Year': '2023'
}
```

Object

- It is an individual unit of data stored in a bucket
- Can be a file of any type:
  - documents, images, videos, etc
- It contains both data and metadata:
  - Data refers to file contents. Metadata include file attributes.
  - e.g., a file called sunset.jpg is uploaded into a bucket.
- **How to identify an object?** object key + version ID (if enabled)
  - Object key is a string specifying the object's location and name, e.g., *0123456-my-first-bucket/subdir/sunset.jpg*
  - Version ID: denotes a specific version of an object, e.g., Version ID: v1AbCdEfGhIjKlMnOpQrStUvWxYz1234567890
  - A combination of an object key and version ID uniquely identifies a specific version of an object in a bucket.

Object ownership



Object Ownership Info

Control ownership of objects written to this bucket from other AWS accounts and the use of access control lists (ACLs). Object ownership determines who can specify access to objects.

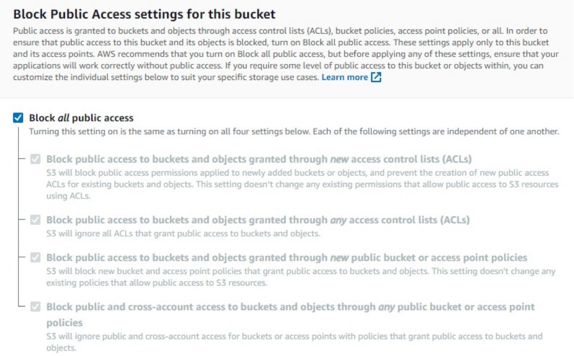
● ACLs disabled (recommended)
All objects in this bucket are owned by this account. Access to this bucket and its objects is specified using only policies.

○ ACLs enabled
Objects in this bucket can be owned by other AWS accounts. Access to this bucket and its objects can be specified using ACLs.

Object Ownership
Bucket owner enforced

- Bucket ACLs are old access-control mechanism for buckets.

Block Public Access settings for this bucket



Block Public Access settings for this bucket

Public access is granted to buckets and objects through access control lists (ACLs), bucket policies, access point policies, or all. In order to ensure that public access to this bucket and its objects is blocked, turn on Block all public access. These settings apply only to this bucket and its access points. AWS recommends that you turn on Block all public access, but before applying any of these settings, ensure that your applications will work correctly without public access. If you require some level of public access to this bucket or objects within, you can customize the individual settings below to suit your specific storage use cases. Learn more

☑ **Block *all* public access**
Turning this setting on is the same as turning on all four settings below. Each of the following settings are independent of one another.

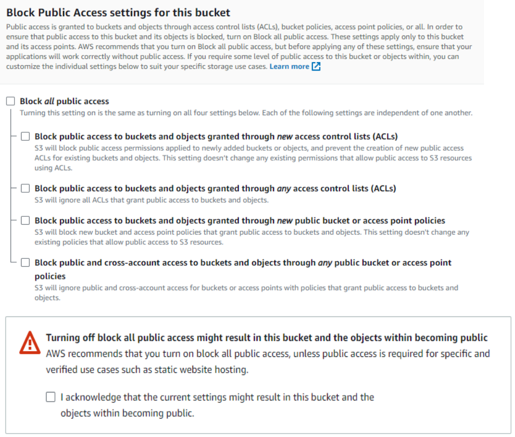
☑ **Block public access to buckets and objects granted through *new* access control lists (ACLs)**
S3 will block public access permissions applied to newly added buckets or objects, and prevent the creation of new public access ACLs for existing buckets and objects. This setting doesn't change any existing permissions that allow public access to S3 resources using ACLs.

☑ **Block public access to buckets and objects granted through *any* access control lists (ACLs)**
S3 will ignore all ACLs that grant public access to buckets and objects.

☑ **Block public access to buckets and objects granted through *new* public bucket or access point policies**
S3 will block new bucket and access point policies that grant public access to buckets and objects. This setting doesn't change any existing policies that allow public access to S3 resources.

☑ **Block public and cross-account access to buckets and objects through *any* public bucket or access point policies**
S3 will ignore public and cross-account access for buckets or access points with policies that grant public access to buckets and objects.

Block Public Access settings for this bucket



Block Public Access settings for this bucket

Public access is granted to buckets and objects through access control lists (ACLs), bucket policies, access point policies, or all. In order to ensure that public access to this bucket and its objects is blocked, turn on Block all public access. These settings apply only to this bucket and its access points. AWS recommends that you turn on Block all public access, but before applying any of these settings, ensure that your applications will work correctly without public access. If you require some level of public access to this bucket or objects within, you can customize the individual settings below to suit your specific storage use cases. Learn more

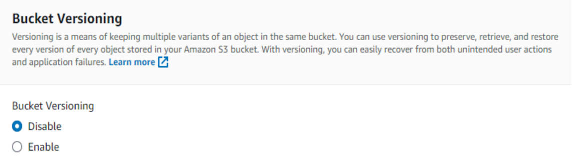
□ **Block *all* public access**
Turning this setting on is the same as turning on all four settings below. Each of the following settings are independent of one another.

- □ **Block public access to buckets and objects granted through *new* access control lists (ACLs)**
  S3 will block public access permissions applied to newly added buckets or objects, and prevent the creation of new public access ACLs for existing buckets and objects. This setting doesn't change any existing permissions that allow public access to S3 resources using ACLs.
- □ **Block public access to buckets and objects granted through *any* access control lists (ACLs)**
  S3 will ignore all ACLs that grant public access to buckets and objects.
- □ **Block public access to buckets and objects granted through *new* public bucket or access point policies**
  S3 will block new bucket and access point policies that grant public access to buckets and objects. This setting doesn't change any existing policies that allow public access to S3 resources.
- □ **Block public and cross-account access to buckets and objects through *any* public bucket or access point policies**
  S3 will ignore public and cross-account access for buckets or access points with policies that grant public access to buckets and objects.

⚠ **Turning off block all public access might result in this bucket and the objects within becoming public**
AWS recommends that you turn on block all public access, unless public access is required for specific and verified use cases such as static website hosting.

□ I acknowledge that the current settings might result in this bucket and the objects within becoming public.

Bucket versioning

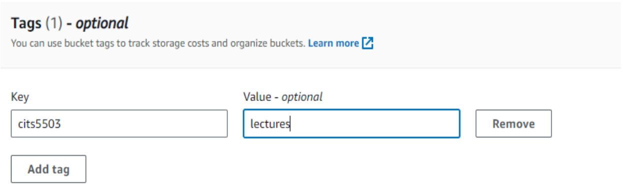


Bucket Versioning

Versioning is a means of keeping multiple variants of an object in the same bucket. You can use versioning to preserve, retrieve, and restore every version of every object stored in your Amazon S3 bucket. With versioning, you can easily recover from both unintended user actions and application failures. Learn more

Bucket Versioning
● Disable
○ Enable

Tags



Tags (1) - ***optional***

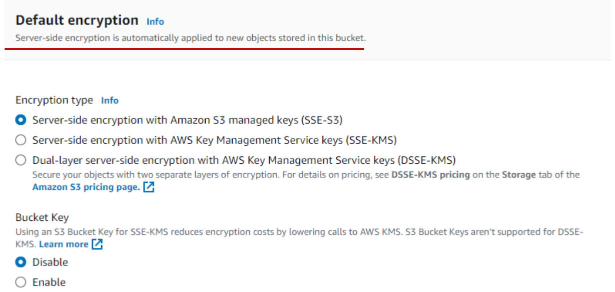
You can use bucket tags to track storage costs and organize buckets. Learn more

Key	Value - *optional*	
cits5503	lectures	Remove

Add tag

- Tags are key-value pairs that provide customized label to help identify and manage our buckets, e.g.,
  - For a bucket of cits5503/lectures, where ‘cits5503’ is a tag key and ‘lectures’ is a tag value.
  - For a bucket of cits5503/labs, where ‘cits5503’ is a tag key and ‘labs’ is a tag value.

Default encryption



Default encryption Info

Server-side encryption is automatically applied to new objects stored in this bucket.

Encryption type Info
● Server-side encryption with Amazon S3 managed keys (SSE-S3)
○ Server-side encryption with AWS Key Management Service keys (SSE-KMS)
○ Dual-layer server-side encryption with AWS Key Management Service keys (DSSE-KMS)
Secure your objects with two separate layers of encryption. For details on pricing, see **DSSE-KMS pricing** on the **Storage** tab of the **Amazon S3 pricing page.**

Bucket Key
Using an S3 Bucket Key for SSE-KMS reduces encryption costs by lowering calls to AWS KMS. S3 Bucket Keys aren't supported for DSSE-KMS. Learn more
● Disable
○ Enable

Default encryption

- **Server-side encryption with S3 managed keys (SSE-S3):** S3 manages the encryption keys (AES-256 encryption) used to encrypt and decrypt objects.
- **Server-side encryption with AWS Key Management Service keys (SSE-KMS):** It uses KMS to manage the encryption keys for each object stored in the bucket.

Security, Identity & Compliance

AWS Key Management Service

Easily create keys and control encryption across AWS and beyond

AWS Key Management Service (KMS) is a managed service that makes it easy for you to create and manage keys and control the use of encryption across a wide range of AWS services. KMS is a secure and resilient service that uses FIPS 140-2 validated hardware security modules to isolate and protect your keys.

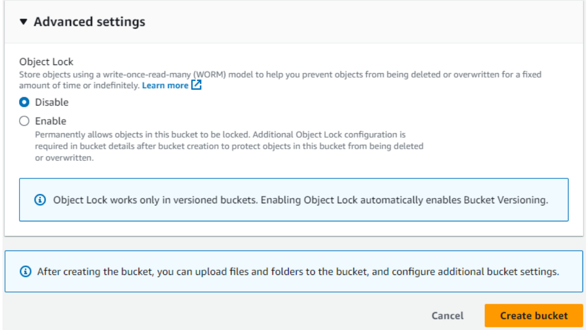
Get started now

You can create a key by clicking the button below.

Default encryption

- Server-side encryption with S3 managed keys (SSE-S3): S3 manages the encryption keys (AES-256 encryption) used to encrypt and decrypt objects.
- Server-side encryption with AWS Key Management Service keys (SSE-KMS): It uses KMS to manage the encryption keys for each object stored in the bucket.
- **Dual-layer server-side encryption with AWS KMS (DSSE-KMS):** It is a combination of SSE-S3 and SSE-KMS.

Object lock



▼ **Advanced settings**

Object Lock

Store objects using a write-once-read-many (WORM) model to help you prevent objects from being deleted or overwritten for a fixed amount of time or indefinitely. Learn more

● Disable

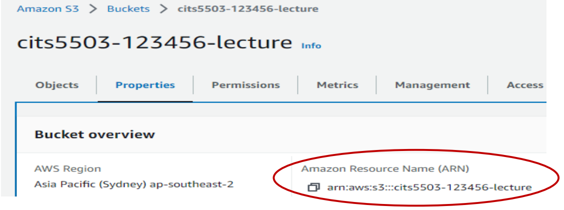
○ Enable

Permanently allows objects in this bucket to be locked. Additional Object Lock configuration is required in bucket details after bucket creation to protect objects in this bucket from being deleted or overwritten.

ⓘ Object Lock works only in versioned buckets. Enabling Object Lock automatically enables Bucket Versioning.

ⓘ After creating the bucket, you can upload files and folders to the bucket, and configure additional bucket settings.

Configure a bucket



cits5503-123456-lecture Info

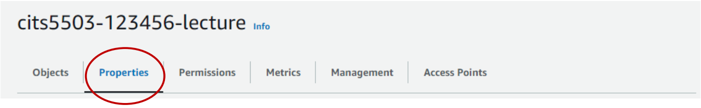
Bucket overview

AWS Region: Asia Pacific (Sydney) ap-southeast-2

Amazon Resource Name (ARN): arn:aws:s3:::cits5503-123456-lecture

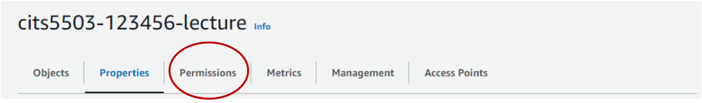
- ARN: a unique ID for any AWS resources, such as S3 buckets, EC2 instances and IAM users.
- ARN is needed when configuring a bucket policy

Configure a bucket



- Properties
  - Bucket versioning
  - Tags
  - Default encryption
  - Object Lock
  - **Server access logging**
    - Provides records for the requests that are made to a bucket.

Configure a bucket



- Permissions
  - Bucket policy
  - CORS (Cross-origin resource sharing) policy

Bucket policy

- Secure access to objects in one or more buckets
  - For unauthenticated users, access is denied
  - For authenticated users, access is dependent on their permissions

Version: indicates the language version of the policy language.

Statement: represents a permission rule.

Effect: what the effect will be when a user requests the specific action—this can be either **'Allow**' or **'Deny**'.

Principal: refers to a set of users/applications who have access to the actions and resources in the statement.

Action: defines a set of resource operations a user/application is allowed (or denied) to perform.

Resource: specifies AWS resources for which a user is allowed or denied to take actions. ARN identifies the bucket.

```
{
        "Version": "2012-10-17",
        "Statement": [{
                "Effect": "Allow",
                "Principal": "*",
                "Action": "s3:GetObject",
                "Resource": "arn:aws:s3::: cits5503-123456-lecture /*"
        }]
}
```

Bucket policy

Id: An optional identifier for the policy, denoting a unique name for the policy.

Sid: An optional identifier for the statement, denoting a unique name for the statement.

s3:GetObject: Allows users to read objects in the bucket.

s3:GetBucketLocation: Allow users to retrieve the bucket's region.

s3:ListBucket: Allows users to list the objects in the bucket.

What is the difference between the two ARNs in the Resource field?

```
{
        "Version": "2012-10-17",
        "Id": "CITS5503Policy",
        "Statement": [{
                "Sid": "CITS5503Statement",
                "Effect": "Allow",
                "Principal": {
                        "AWS": "arn:aws:iam::489389878001:user/Dave"
                },
                "Action": [
                        "s3:GetObject",
                        "s3:GetBucketLocation",
                        "s3:ListBucket"
                ],
                "Resource": [
                        "arn:aws:s3:::cits5503-123456-lecture/*",
                        "arn:aws:s3::: cits5503-123456-lecture "
                ]
        }]
}
```

Bucket policy: practice

- What does this code snippet do?

```
{
	"Version": "2012-10-17",
	"Id": "S3PolicyIP",
	"Statement": [{
		"Sid": "IPAllow",
		"Effect": "Allow",
		"Principal": "*",
		"Action": [
			"s3:GetObject",
			"s3:ListBucket",
		],
		"Resource": [
			"arn:aws:s3:::cits5503-123456-lecture/",
			"arn:aws:s3::: cits5503-123456-lecture/*"
		],
		"Condition": {
			"IpAddress": {
				"aws:SourceIp": "192.0.2.0/24"
			}

		}

	}]

}
```

Bucket policy: practice

- This policy allows two operations to the bucket named "cits5503-123456-lecture") and its contents from IP addresses within the "192.0.2.0/24" range. This range covers all IP addresses from 192.0.2.0 to 192.0.2.255, inclusive.

- What does 192.0.2.0/16 mean?

```
{
	"Version": "2012-10-17",
	"Id": "S3PolicyIP",
	"Statement": [{
		"Sid": "IPAllow",
		"Effect": "Allow",
		"Principal": "*",
		"Action": [
			"s3:GetObject",
			"s3:ListBucket",
		],
		"Resource": [
			"arn:aws:s3:::cits5503-123456-lecture/",
			"arn:aws:s3::: cits5503-123456-lecture/*"
		],
		"Condition": {
			"IpAddress": {
				"aws:SourceIp": "192.0.2.0/24"
			}

		}

	}]

}
```

Common S3 actions

s3:GetObject: Allow users to read objects from the bucket.

s3:PutObject: Allow users to upload new objects to the bucket.

s3:DeleteObject: Allow users to delete objects from the bucket.

s3:ListBucket: Allow users to list the objects in the bucket.

s3:GetBucketLocation: Allow users to retrieve the bucket's region.

A complete list of actions (https://docs.aws.amazon.com/AmazonS3/latest/API/API_Operations.html)

CORS (Cross-origin Resource Sharing)

- CORS allows specific origins to access a bucket and specifies the allowed HTTP methods and headers for each origin.
- An example:

- Rule 1:
  - AllowedHeaders: All headers are allowed.
  - AllowedMethods: PUT, POST, and DELETE HTTP methods are allowed.
  - AllowedOrigins: HTTP requests originating from "http://www.example1.com" are allowed to access the bucket.
  - ExposeHeaders: No additional headers are exposed.
- Rule 2: The same as Rule 1 except the origin.
- Rule 3:
  - AllowedHeaders: No specific headers are allowed.
  - AllowedMethods: Only the GET HTTP method is allowed.
  - AllowedOrigins: All origins are allowed.
  - ExposeHeaders: No additional headers are exposed.

```
[
	{
		"AllowedHeaders": ["*"],
		"AllowedMethods": ["PUT", "POST", "DELETE"],
		"AllowedOrigins": ["http://www.example1.com"],
		"ExposeHeaders": []
	},
	{
		"AllowedHeaders": ["*"],
		"AllowedMethods": ["PUT", "POST", "DELETE"],
		"AllowedOrigins": ["http://www.example2.com"],
		"ExposeHeaders": []
	},
	{
		"AllowedHeaders": [],
		"AllowedMethods": ["GET"],
		"AllowedOrigins": ["*"],
		"ExposeHeaders": []
	}
]
```

Demo (python)

- Create a bucket, called cits5503-lecture-bucket
- Enable bucket versioning
- PUT an object (called uwa_campus.jpg) into the bucket
- GET the uploaded object from the bucket
- Update uwa_campus.jpg in the cits5503-lecture-bucket
  - When we PUT an object, it gets a new version
- GET the latest version of uwa_campus.jpg
  - When we get an object:
    - 1) an unversioned request likely receives the last version, but this is not guaranteed;
    - 2) a request for object key + version ID uniquely maps to a single object of a specified version.

Practice Questions

- [5 marks] Q1: When a Bucket is created, AWS allows the specification of a number of features that can be managed. What are the key properties and features?

- [1 mark] **Bucket Name**: A unique name that identifies a bucket. Bucket names must be globally unique across all existing bucket names in S3.
- [1 mark] **Region**: a bucket is associated with an AWS region, which determines the physical location where a bucket is stored.
- [1 mark] **Bucket versioning**: keeps multiple versions of an object in the same bucket.
- [1 mark] **Default Encryption**: All objects uploaded to the bucket will be automatically encrypted using one server-side encryption.
- [1 mark] **Object Lock**: it prevent objects in a bucket from being deleted or overwritten for a fixed amount of time or indefinitely.

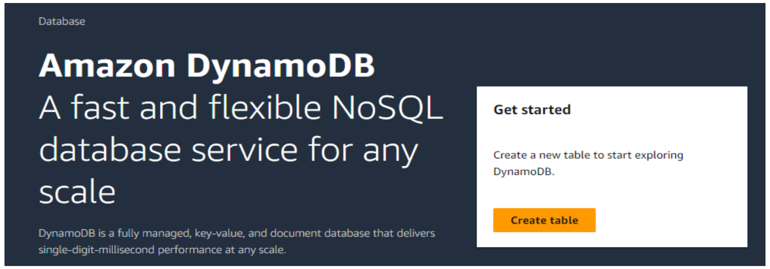
Practice Questions

- [5 marks] Q2: Describe how S3 handles consistency of objects and how this approach affects the state of objects when they are read using a GET.

- [4 marks] S3 delivers strong **read-after-write** and **list** (i.e., GET, PUT and LIST operations) consistency automatically. Specifically, **what a user write is what they will read**, and **the results of a LIST will be an accurate reflection of what's in the bucket**.
- [1 mark] When GET is used to read an object, the read request immediately receives the latest version of the object.

Amazon DynamoDB

- A fast and flexible non-relational database service
  - List a relational database management system: MySQL, PostgreSQL, etc.



Amazon DynamoDB

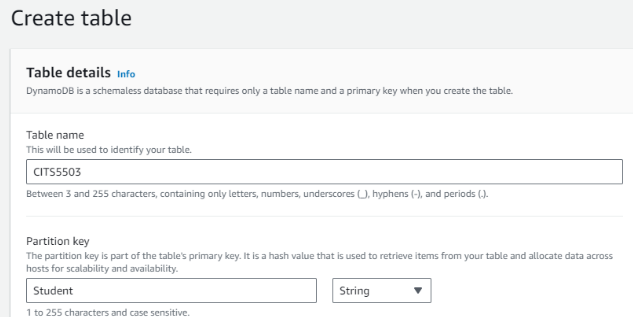
A fast and flexible NoSQL database service for any scale

DynamoDB is a fully managed, key-value, and document database that delivers single-digit-millisecond performance at any scale.

Get started

Create a new table to start exploring DynamoDB.

Amazon DynamoDB



Create table

Table details Info

DynamoDB is a schemaless database that requires only a table name and a primary key when you create the table.

Table name

This will be used to identify your table.

CITS5503

Between 3 and 255 characters, containing only letters, numbers, underscores (_), hyphens (-), and periods (.).

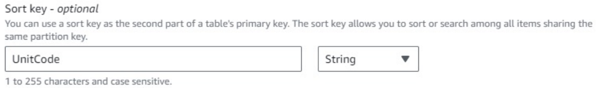
Partition key

The partition key is part of the table's primary key. It is a hash value that is used to retrieve items from your table and allocate data across hosts for scalability and availability.

Student | String ▼

1 to 255 characters and case sensitive.

Amazon DynamoDB

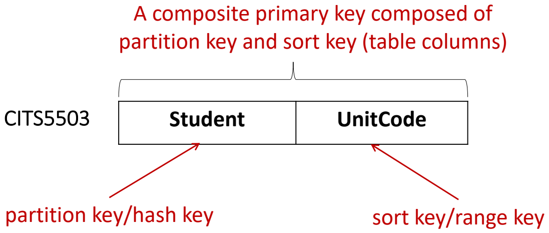


Sort key - *optional*

You can use a sort key as the second part of a table's primary key. The sort key allows you to sort or search among all items sharing the same partition key.

UnitCode | String ▼

1 to 255 characters and case sensitive.



A composite primary key composed of partition key and sort key (table columns)

CITS5503

Student	UnitCode

partition key/hash key

sort key/range key

Amazon DynamoDB

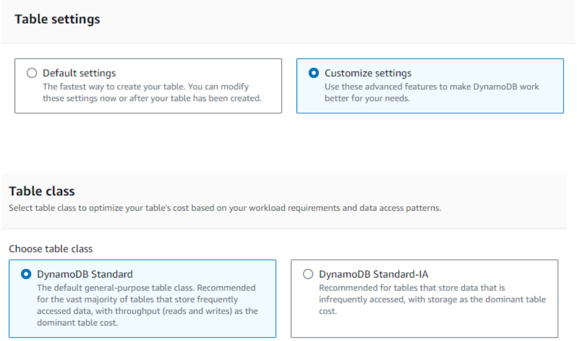


Table settings

○ Default settings
The fastest way to create your table. You can modify these settings now or after your table has been created.

● Customize settings
Use these advanced features to make DynamoDB work better for your needs.

Table class

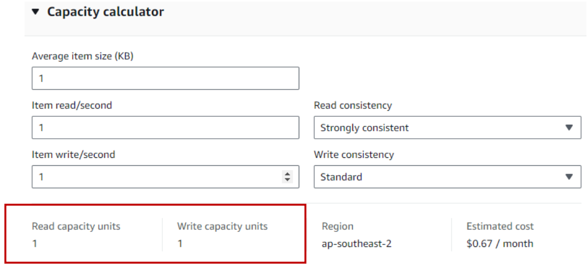
Select table class to optimize your table's cost based on your workload requirements and data access patterns.

Choose table class

● DynamoDB Standard
The default general-purpose table class. Recommended for the vast majority of tables that store frequently accessed data, with throughput (reads and writes) as the dominant table cost.

○ DynamoDB Standard-IA
Recommended for tables that store data that is infrequently accessed, with storage as the dominant table cost.

Amazon DynamoDB



▼ **Capacity calculator**

Average item size (KB)
1

Item read/second
1

Read consistency
Strongly consistent ▼

Item write/second
1

Write consistency
Standard ▼

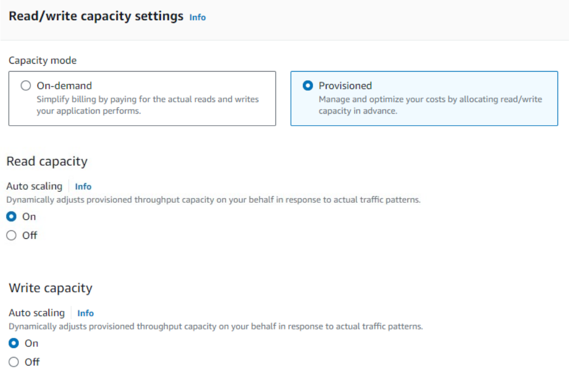
Read capacity units	Write capacity units	Region	Estimated cost
1	1	ap-southeast-2	$0.67 / month

- Throughput capacity
  - 1 read/write capacity unit (RCU/WCU) = read/write 1 item of 1 KB in 1 second

Read/Write consistency

- Read from DynamoDB can be:
  - Eventually Consistent
    - The response of a read may not reflect the result of a recent write operation
  - Strongly Consistent
    - The response of a read returns the most up-to-date data reflecting all updates from all previous write operations
  - Transactional
    - Group multiple read actions together and submit them in a single all-or-nothing operation
- Write from DynamoDB can be:
  - Standard
  - Transactional
    - Group multiple read actions together and submit them in a single all-or-nothing operation

Amazon DynamoDB



Read/write capacity settings Info

Capacity mode

○ On-demand
Simplify billing by paying for the actual reads and writes your application performs.

● Provisioned
Manage and optimize your costs by allocating read/write capacity in advance.

Read capacity

Auto scaling | Info
Dynamically adjusts provisioned throughput capacity on your behalf in response to actual traffic patterns.

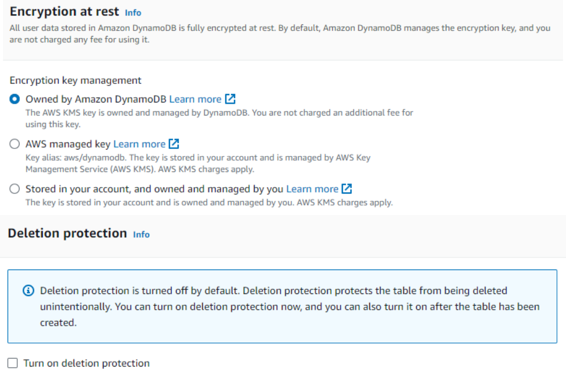
● On
○ Off

Write capacity

Auto scaling | Info
Dynamically adjusts provisioned throughput capacity on your behalf in response to actual traffic patterns.

● On
○ Off

Amazon DynamoDB



Encryption at rest Info
All user data stored in Amazon DynamoDB is fully encrypted at rest. By default, Amazon DynamoDB manages the encryption key, and you are not charged any fee for using it.

Encryption key management

● Owned by Amazon DynamoDB Learn more
The AWS KMS key is owned and managed by DynamoDB. You are not charged an additional fee for using this key.

○ AWS managed key Learn more
Key alias: aws/dynamodb. The key is stored in your account and is managed by AWS Key Management Service (AWS KMS). AWS KMS charges apply.

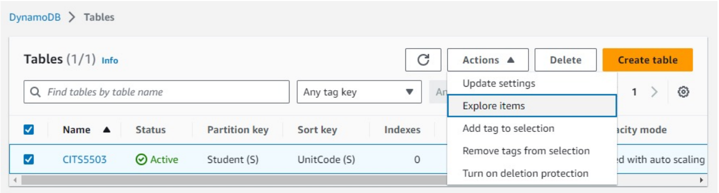
○ Stored in your account, and owned and managed by you Learn more
The key is stored in your account and is owned and managed by you. AWS KMS charges apply.

Deletion protection Info

Deletion protection is turned off by default. Deletion protection protects the table from being deleted unintentionally. You can turn on deletion protection now, and you can also turn it on after the table has been created.

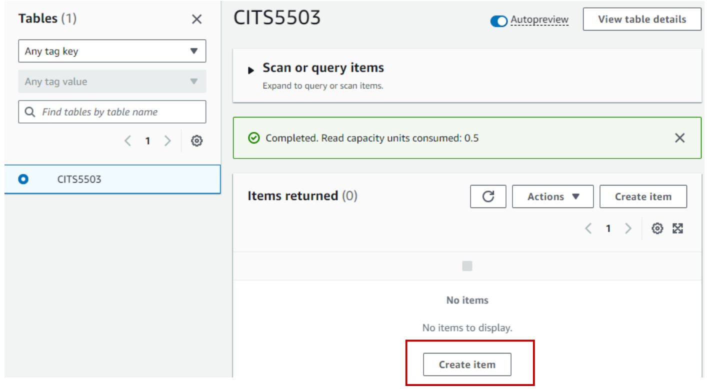
□ Turn on deletion protection

Amazon DynamoDB



☑	Name	Status	Partition key	Sort key	Indexes	acity mode
☑	CITS5503	✓ Active	Student (S)	UnitCode (S)	0	d with auto scaling

Amazon DynamoDB



CITS5503

▸ Scan or query items

Expand to query or scan items.

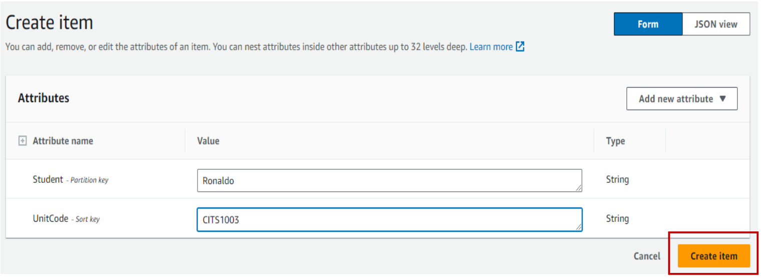
Completed. Read capacity units consumed: 0.5

Items returned (0)

No items

No items to display.

Amazon DynamoDB



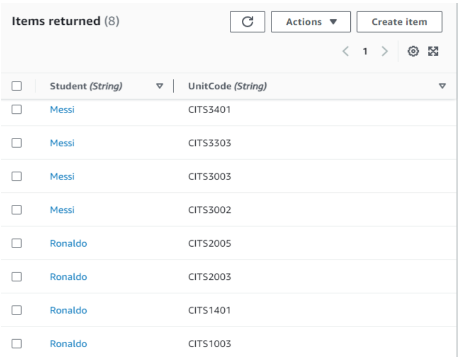
Create item

You can add, remove, or edit the attributes of an item. You can nest attributes inside other attributes up to 32 levels deep. Learn more

Attributes

Attribute name	Value	Type
Student - *Partition key*	Ronaldo	String
UnitCode - *Sort key*	CITS1003	String

Amazon DynamoDB

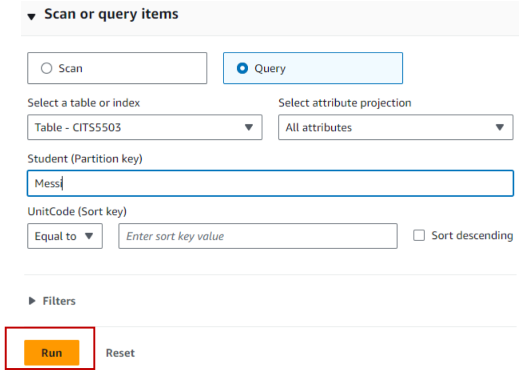


Items returned (8)

□	Student *(String)*	UnitCode *(String)*
□	Messi	CITS3401
□	Messi	CITS3303
□	Messi	CITS3003
□	Messi	CITS3002
□	Ronaldo	CITS2005
□	Ronaldo	CITS2003
□	Ronaldo	CITS1401
□	Ronaldo	CITS1003

The table looks similar to a spreadsheet

Amazon DynamoDB



▾ Scan or query items

○ Scan ● Query

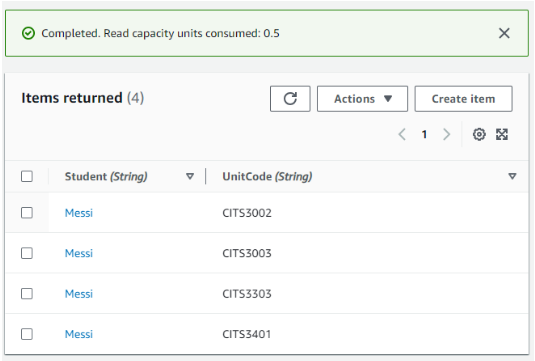
Select a table or index: Table - CITS5503

Select attribute projection: All attributes

Student (Partition key): Messi

UnitCode (Sort key): Equal to | *Enter sort key value* | □ Sort descending

Amazon DynamoDB

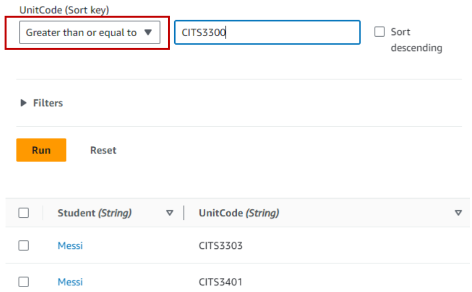


Completed. Read capacity units consumed: 0.5

Items returned (4)

□	Student *(String)*	UnitCode *(String)*
□	Messi	CITS3002
□	Messi	CITS3003
□	Messi	CITS3303
□	Messi	CITS3401

Amazon DynamoDB

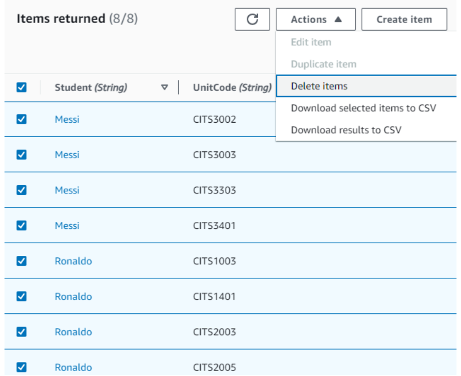


UnitCode (Sort key): Greater than or equal to ▼ CITS3300

□ Sort descending

□	Student *(String)*	UnitCode *(String)*
□	Messi	CITS3303
□	Messi	CITS3401

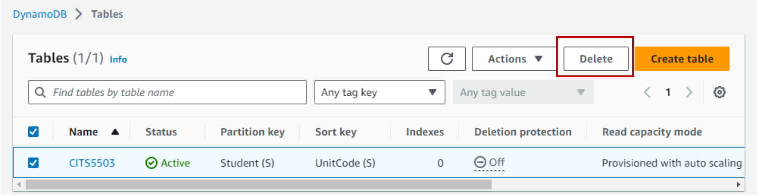
Amazon DynamoDB



Items returned (8/8)

☑	Student *(String)*	UnitCode *(String)*
☑	Messi	CITS3002
☑	Messi	CITS3003
☑	Messi	CITS3303
☑	Messi	CITS3401
☑	Ronaldo	CITS1003
☑	Ronaldo	CITS1401
☑	Ronaldo	CITS2003
☑	Ronaldo	CITS2005

Amazon DynamoDB



DynamoDB > Tables

Tables (1/1) Info

☑	Name ▲	Status	Partition key	Sort key	Indexes	Deletion protection	Read capacity mode
☑	CITS5503	Active	Student (S)	UnitCode (S)	0	Off	Provisioned with auto scaling

DynamoDB API

- Control Plane (manage a table)
  - CreateTable, DescribeTable, ListTables, UpdateTable, DeleteTable
- Data Plane (manage data in a table)
  - Create data: PutItem/BatchWriteItem
  - Read data: GetItem/BatchGetItem/Query/Scan
  - Update data: UpdateItem
  - Delete data: DeleteItem/BatchWriteItem
- A summary of API operations

https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/HowItWorks.API.html#HowItWorks.API.ControlPlane

Practice Questions

- Q3: You are asked to store data about music albums in a local DynamoDB table. For each album, you need to record the title of the album and the artist name. Describe the AWSCLI commands you would use to create a table to store such information and write entries to that table in DynamoDB.

Set up a local DynamoDB

- From a terminal:
  - mkdir dynamodb; cd dynamodb
  - install jre if not done (sudo apt-get install default-jre)
  - wget https://s3-ap-northeast-1.amazonaws.com/dynamodb-local-tokyo/dynamodb_local_latest.tar.gz
  - java -Djava.library.path=./DynamoDBLocal_lib -jar DynamoDBLocal.jar –sharedDb

Create Table

```
aws dynamodb create-table --table-name MusicAlbum
  --attribute-definitions \
    AttributeName=Artist,AttributeType=S \
    AttributeName=Song,AttributeType=S \
  --key-schema AttributeName=Artist,KeyType=HASH \
               AttributeName=Song,KeyType=RANGE  \
 --provisioned-throughput ReadCapacityUnits=1,WriteCapacityUnits=1 \
 --endpoint-url=http://localhost:8000
```

- Creates a table named "MusicAlbum" with two attributes ("Artist" and "Song"),
- Sets up "Artist" as the partition key and "Song" as the sort key,
- Assigns provisioned throughput capacity of 1 read capacity unit and 1 write capacity unit,
- Connects to a DynamoDB instance running locally on http://localhost:8000.

Create Entries

```
aws dynamodb put-item \
  --table-name MusicAlbum \
  --item \ '{"Artist": {"S": “Tom"}, "Song": {"S": "Call Me Today"}}’\
  --return-consumed-capacity TOTAL --endpoint-url=http://localhost:8000


aws dynamodb put-item \
  --table-name MusicAlbum \
  --item '{"Artist": {"S": “Jerry"}, "Song": {"S": "Happy Day"}}’ \
  --return-consumed-capacity TOTAL  --endpoint-url=http://localhost:8000
```

- Insert two items with values about of attributes,
- Request information about the consumed capacity for the operations,
- Specify a local DynamoDB for the connection.

- What if we want to add one more attribute to this table, e.g., AlbumTitle?

Create Entries

```
aws dynamodb put-item \
  --table-name MusicAlbum \
  --item \ '{"Artist": {"S": “Tom"}, "Song": {"S": "Call Me Today"},
               "AlbumTitle": {"S": "Somewhat Famous"}}’ \
  --return-consumed-capacity TOTAL --endpoint-url=http://localhost:8000


aws dynamodb put-item \
  --table-name MusicAlbum \
  --item '{"Artist": {"S": “Jerry"}, "Song": {"S": "Happy Day"}, \
            "AlbumTitle": {"S": "Songs About Life"} }’ \
--return-consumed-capacity TOTAL  --endpoint-url=http://localhost:8000
```

Query

```
aws dynamodb query \
  --table-name MusicAlbum \
  --key-condition-expression "Artist = :artist" \
  --expression-attribute-values '{":artist":{"S":"Tom"}}’ \
  --endpoint-url=http://localhost:8000
```

- Queries the "MusicAlbum" table for items where the "Artist" is "Tom".

Scan

```
aws dynamodb scan \
  --table-name MusicAlbum \
  --endpoint-url=http://localhost:8000
```

Data Types

- Attributes can be:
  - Scalar Types: represent exactly one value
    - Number, String, Binary, Boolean, null
  - Set Types: an array of the same scalar type
    - ["Black", "Green" ,"Red"]
    - [42.2, -19, 7.5, 3.14]
  - Document Types
- A detailed description of data types:

  https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/HowItWorks.NamingRulesDataTypes.html#HowItWorks.DataTypes

Document Types

- **List**: a collection of values, enclosed in square brackets: [...]
  - FavoriteThings: ["Cookies", "Coffee", 3.14159]
- **Map**: a hierarchical structure of attributes within a single attribute, enclosed in curly brackets: {...}

```
{
        Day: "Monday",
        UnreadEmails: 42,
        ItemsOnMyDesk: [
                "Coffee Cup",
                "Telephone",
                {
                        Pens: { Quantity : 3},
                        Pencils: { Quantity : 2}
                }
        ]
}
```