



# Day 37



# BLOB Data Type

- BLOB (Binary Large Object) is a special type of column in MySQL for storing binary data
  - Eg. images, files, MP3, etc.
- Unlike other columns, the contents of BLOB are not searchable, sortable or comparable
  - Need to have additional columns to hold the BLOB's metadata
  - Eg. media type, size, original file name, etc.
- BLOB comes in 4 different sizes
  - TINYBLOB <  $2^8$  bytes
  - BLOB <  $2^{16}$  bytes / 16Kb
  - MEDIUMBLOB <  $2^{24}$  bytes / 16Mb
  - LONGBLOB <  $2^{32}$  bytes / 4Gb



# Blob Data Type

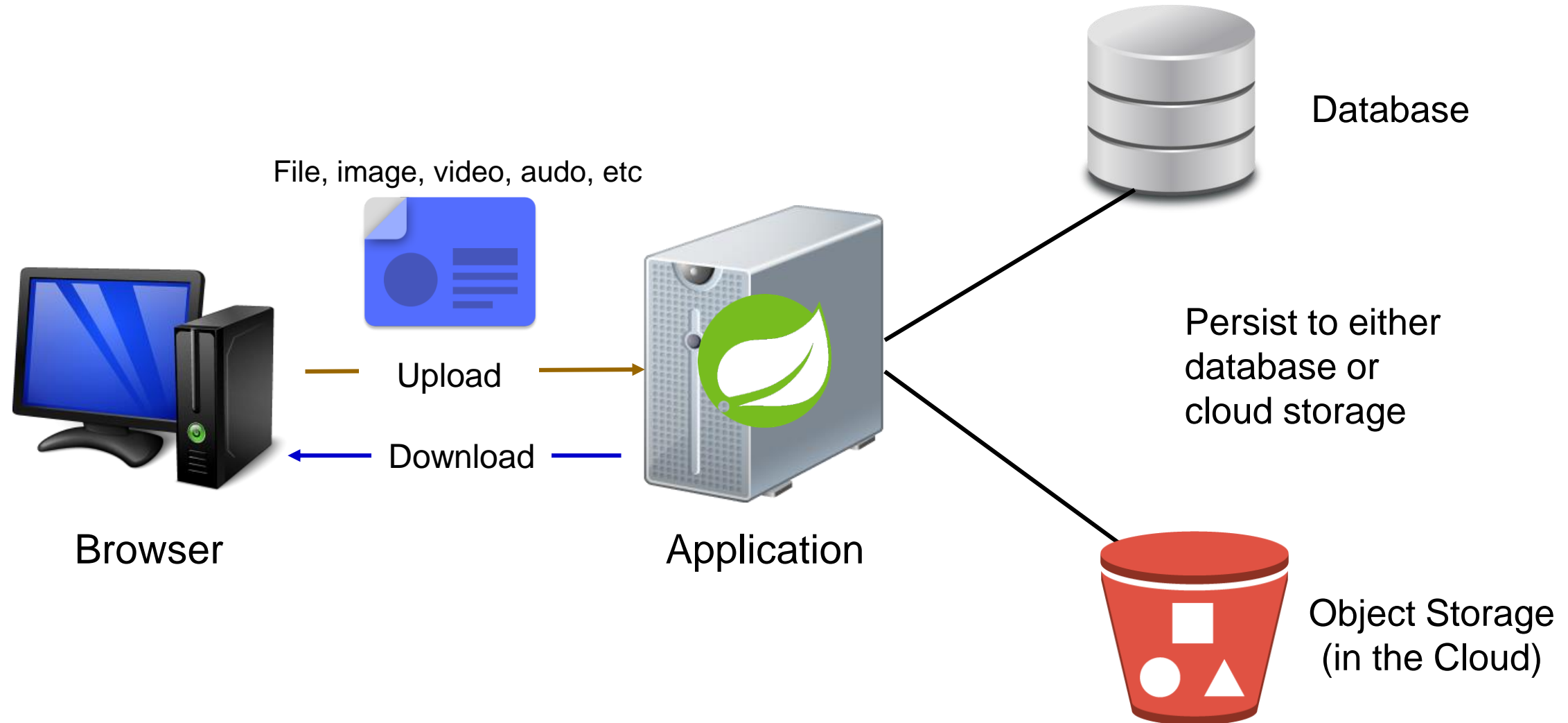
```
create table files (  
  id          int auto_increment,  
  filename    varchar(64) not null,  
  media_type  varchar(128) not null,  
  content     blob not null,  
  primary key (prog_id)  
);
```

Blob data type

A black arrow pointing from the text "Blob data type" to the "blob not null" part of the "content" column definition in the SQL code.



# File Upload





# File Upload

HTTP POST method

Use the multipart encoding type when submitting the form

```
<form method="POST" action="/upload"
      enctype="multipart/form-data">
```

Set the input type to file

...

```
<input type="file" name="img-file" accept="image/*">
```

```
<textarea name="notes" cols="30" rows="10">
</textarea>
```

```
<button type="submit">Upload</button>
```

```
</form>
```

Optionally set the type of file to upload





# File Upload with Angular

- Media type of file upload is `multipart/form-data`
- Use `FormData` object type to hold the fields
  - See <https://developer.mozilla.org/en-US/docs/Web/API/FormData>
- For the list of files to be uploaded, need to get it from the `input` element
  - `files` attribute; see [https://developer.mozilla.org/en-US/docs/Web/API/File/Using\\_files\\_from\\_web\\_applications](https://developer.mozilla.org/en-US/docs/Web/API/File/Using_files_from_web_applications)
  - Define a template reference on the input as `TemplateRef`
  - Access the `files` `templateRef.nativeElement.files`



# Angular File Upload

```
<form [formGroup]="form" (ngSubmit)="upload()">  
  ...
```

Image:

```
<input type="file" accept="image/*"  
  [formControlName]="file' #file">
```

...

```
<button type="submit">POST</button>  
</form>
```

Define a template reference  
on the input element



# Angular File Upload

```
export class AppComponent implements OnInit {  
  @ViewChild('file') imageFile: ElementRef;  
  form1: FormGroup;  
  constructor(private http: HttpClient, private fb: FormBuilder) { }
```

Get a reference to the input  
element using its name

```
  ngOnInit() {  
    this.formGroup = this.fb.group({  
      'image-file': this.fb.control('')  
    })  
  }
```

Create a instance of FormData  
to hold the parameters to be  
send to the server

```
  upload() {  
    const formData = new FormData();  
    formData.set('name', this.form.get('image-file').value);  
    ...  
    formData.set('file', this.imageFile.nativeElement.files[0]);  
    firstVaulueFrom(  
      this.http.post('http://localhost:8080/upload', formData)  
    ).then(() => { ... })  
      .catch((error) => { ... })  
  }
```

Access the DOM attribute with  
nativeElement attribute

POST the FormData. Angular will use the  
correct media type for making the request

Populate the  
FormData instance





# multipart/form-data Media Type

```
POST /upload HTTP/1.1
Host: localhost:3000
Connection: keep-alive
Content-Length: 499207
Content-Type: multipart/form-data; boundary=-----0YsU72sGdwPe5B
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
```

Field separator

```
-----0YsU72sGdwPe5B
Content-Disposition: form-data; name="file"; filename="directions.png"
Content-Type: image/png

-----0YsU72sGdwPe5B
Content-Disposition: form-data; name="name"

Directions to Bukit Timah Nature Reserve
-----0YsU72sGdwPe5B--
```

image-file form field

name form field



# Enable Multipart Form Data

- Multipart form data processing is not enabled by default in SpringBoot
  - Add the following configuration to `application.properties`

Enable multipart uploads

Maximum size of a single file

Maximum request size for multiple file uploads

```
spring.servlet.multipart.enabled=true  
spring.servlet.multipart.max-file-size=200MB  
spring.servlet.multipart.max-request-size=200MB  
spring.servlet.multipart.file-size-threshold=1MB
```

Files exceeding this size will be written to disk temporarily instead of residing memory during processing



# Processing multipart/form-data

```
@Controller
```

```
@RequestMapping(path="/upload")
```

```
public class UploadController {
```

```
    @Autowired JdbcTemplate template;
```

```
    @PostMapping(consumes=Mediatype.MULTIPART_FORM_DATA)
```

```
    public ResponseEntity<String> postUpload(@RequestPart MultipartFile file,  
        @RequestPart String name, @RequestPart String email) {
```

```
        String name = file.getName();
```

```
        String originalName = file.getOriginalFileName();
```

```
        String mediaType = file.getContentType();
```

```
        InputStream is = file.getInputStream();
```

```
        template.update("insert into files(..., content) values (... , ?)"  
            , ..., is);
```

```
        ...
```

```
    }
```

```
}
```

Handle POST request with  
form-data payload

Retrieve the form-data  
content. For file upload the  
type is MultipartFile

Get other form fields, if any

Insert the contents of the file into a blob  
column with the file's InputStream

Get information about  
the uploaded file



# File Upload

 `<form method="POST" action="/upload" enctype="multipart/form-data">  
 <input type="file" name="file">  
 ...`

↓ as multipart/data-form



**POST** /upload HTTP/1.1

Content-Type: multipart/form-data; boundary=----0YsU72sGdwPe5B



@Controller

@RequestMapping(path="/upload")

public class UploadController {

@PostMapping(consumes=MediaType.MULTIPART\_FORM\_DATA)

public ResponseEntity<String> postUpload(  
 @Requestpart MultipartFile file, ...)





# Retrieving Images

```
GET /api/tv_shows/1
```

```
{  
  "prog_id": 1,  
  "name": "...",  
  "lang": "...",  
  ...,  
  "image": "/image/1"  
}
```

Images/media have to be retrieved  
separately from the content



# Retrieving BLOB with ResultSetExtractor

```
@GetMapping("/{id}")
public ResponseEntity<byte[]> getImage(@PathVariable Integer id) {

    Optional(FileData) opt = template.query("select * from files where id = ?",
        params, (rs: ResultSet) -> {
        if (!rs.next())
            return Optional.empty();
        FileData file = new FileData();
        file.setName(rs.getString("name"));
        file.setContentType(rs.getString("media_type"));
        file.setContent(rs.getBytes("content"));
        ...
        return Optional.of(file);
    }, id
    )
}
```

Use next() to determine if the query produces any result

Get a byte array representing the blob column

Create an array to hold one or more parameters for the query



# Multiple Rows with ResultSetExtractor

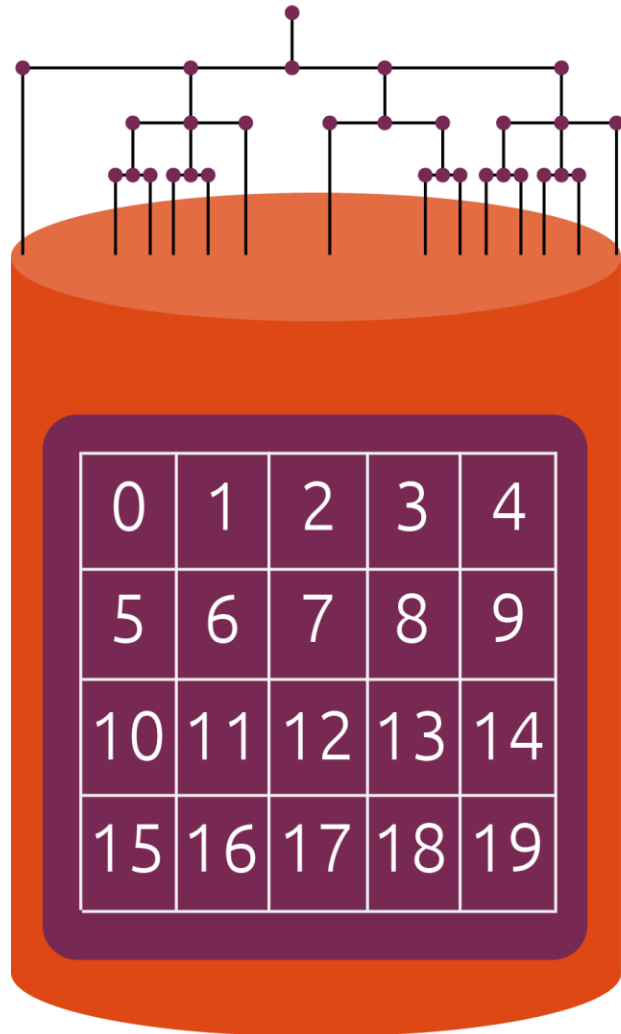
```
List<FileData> opt = template.query("select * from files where name like ?",  
    params, (rs: ResultSet) -> {  
        List<FileData> list = new LinkedList<>();  
        while (rs.next()) {  
            FileData file = new FileData();  
            file.setName(rs.getString("name"));  
            file.setContentType(rs.getString("media_type"));  
            file.setContent(rs.getBytes("content"));  
            list.add(file);  
        }  
        return list;  
    }, "%dog%")
```

Call `next()`. If `next()` returns true, read a record.  
If `next()` returns false, there are no more records  
Every call to `next()` advances the cursor

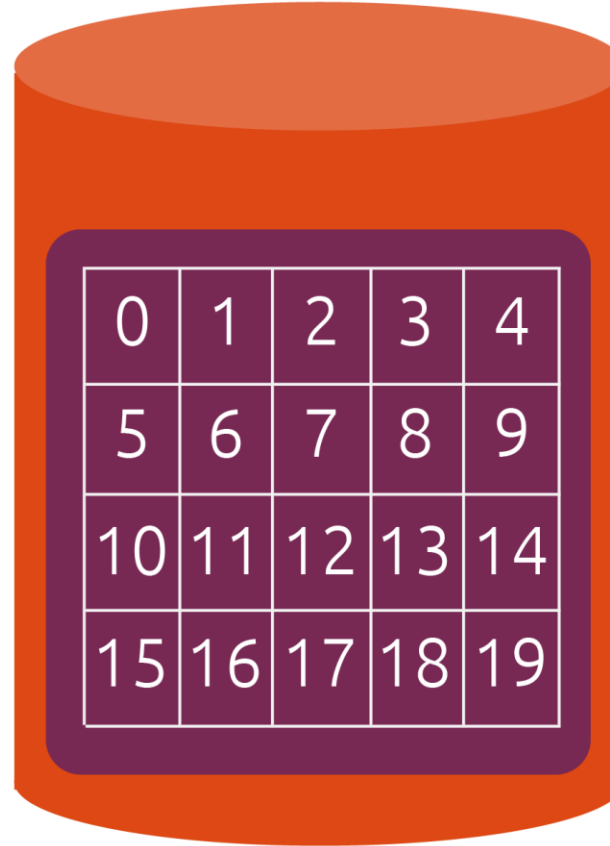


# What is Object Storage?

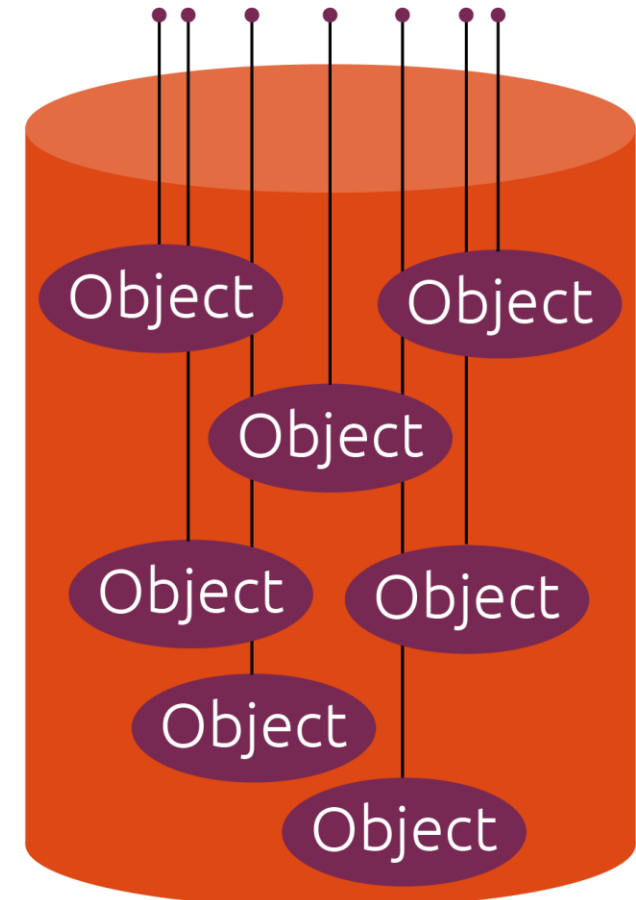
File Storage



Block Storage



Object Storage







# Object Storage

- Object storage stores data as an opaque 'blob'
  - Cannot search the contents of the blob unlike a file or collection or record
- Identified by a key
- Associated metadata with an object
  - Eg. MIME type, caching options, storage class, permissions
  - Also allow users to set custom metadata
- Examples of object storage
  - AWS - S3
  - GCP - Firebase Storage
  - Azure - Blob Storage
  - MySQL - Blob data type



# Creating a S3 Compatible Storage

PROJECTS

first-kube

newsletter

+ New Project

MANAGE

DISCOVER

ACCOUNT

Search by resource name or IP (Ctrl+B)

Create ^

USAGE

Spaces

Name	Size
[redacted]	95 KB 2 items

Object storage basics

[Spaces overview](#)

Discover features, tips, and tools that put Spaces to work for you and your data.

[API docs](#)

Use the Spaces API to create and manage Spaces programmatically.

[Join the](#)

Join the and peer

Droplets

Create cloud servers

Clusters

Create Kubernetes clusters

Databases

Create database clusters

Volumes

Add storage to Droplets

Domains/DNS

Route your existing domains

Cloud Firewalls

Increase Droplet security

Floating IPs

Reserve IP addresses for Droplets

Load Balancers

Distribute traffic to Droplets

Alert Policies

Monitor your Droplets

**Spaces**

Store and serve static assets



# Select Region

PROJECTS ^

first-kube

newsletter

+ New Project

MANAGE v

DISCOVER v

ACCOUNT v

Search by resource name or IP (Ctrl+B)

Create v

USAGE

Create a Space

Choose a datacenter region

San Francisco

2

New York

3

Amsterdam

3

Singapore

1

Frankfurt

1

CDN (Content Delivery Network)

Deliver web assets up to 70% faster with global edge caching technology. Use a custom subdomain you own as an additional endpoint.

Enable CDN

Allow file listing?

Restrict File Listing

Enable File Listing



# Specify Bucket Name

PROJECTS

first-kube

newsletter

+ New Project

MANAGE

DISCOVER

ACCOUNT

Search by resource name or IP (Ctrl+B)

Create

USAGE

☒ **Restrict File Listing**  
Only users who connect to this Space using access keys can list the contents.

☐ **Enable File Listing**  
Anyone can list the contents of this Space.

**Important:** This setting has no effect on whether individual files are visible. It only determines if anonymous users can list the name, size, and other metadata for the files in this Space.

## Finalize and Create

Choose a unique name ?

Choose a unique subdomain name for your Space

acme

✓

Your Space's origin URL: <https://acme.sgp1.digitaloceanspaces.com>

## Select project

Select an existing project for this Space to belong to.

first-kube

▼

Create a Space

Bucket name



# Provisioned Bucket

PROJECTS

first-kube

newsletter

+ New Project

MANAGE

DISCOVER

ACCOUNT

Search by resource name or IP (Ctrl+B)

Create

USAGE

acme

in first-kube

https://acme.sgp1.digitaloceanspaces.com

Files

Settings

Start typing to filter the list of files and folders

Actions

New Folder

Upload Files

acme

0 items

Name	Size	Last Modified
<div></div>		



# Generate Access Key

+ New Project

MANAGE

Droplets

Kubernetes

Volumes

Databases

Spaces

Images

Networking

Monitoring

API

DISCOVER

ACCOUNT

Search by resource name or IP (Ctrl+B)



Create

USAGE

Spaces

Go to to API key

Manage Keys

Name	Size	Created	
 <b>acme</b> https://acme.sgp1.digitaloceanspaces.com	— 0 items	23 minutes ago	<a href="#">More</a>
 [REDACTED]	95 KB 2 items	13 days ago	<a href="#">More</a>

Object storage basics

[Spaces overview](#)

Discover features, tips, and tools that put Spaces to work for you and your data.

[API docs](#)

Use the Spaces API to create and manage Spaces programmatically.

[Join the Discussion](#)

Join the Spaces community and engage with experts and peers.



# Generate Access Key

+ New Project

MANAGE

Droplets

Kubernetes

Volumes

Databases

Spaces

Images

Networking

Monitoring

API

DISCOVER

ACCOUNT

Search by resource name or IP (Ctrl+B)

Create ▾

USAGE

github\_actions

READWRITE

5 months ago

More ▾

doctl

READWRITE

7 months ago

More ▾

Personal access tokens function like a combined name and password for API authentication.

Create keys →

Generate New Key

Spaces access keys

Keys you have generated to connect with third party clients or to access the Spaces API.

Name	Key	Created	
		13 days ago	More ▾







# Generate Access Key

## Spaces access keys

Generate New Key

Keys you have generated to connect with third party clients or to access the [Spaces API](#).

Name	Key	Type in key name	Created
acme_key	  		Just now <a href="#">More</a> 

## Spaces access keys

Generate New Key

Keys you have generated to connect with third party clients or to access the [Spaces API](#).

Name	Key	Created
acme_key	PRIIR4ZWOQJPXPJRXE76	Just now
Secret	RJzNMf2WL3R0mmxCZWwKaNd4E8HweB1YZxs20Wh65eg	

API and secret key. Make a copy of the secret key (longer). Will only be shown once





# Setup

- Add the following dependencies

```
<dependency>
  <groupId>com.amazonaws</groupId>
  <artifactId>aws-java-sdk-s3</artifactId>
  <version> latest version </version>
</dependency>
<dependency>
  <groupId>org.glassfish.jaxb</groupId>
  <artifactId>jaxb-runtime</artifactId>
  <version> latest version </version>
</dependency>
```



# Configure S3 Client

Create credentials with  
the access and secret  
key pair

Configure the  
S3 endpoint

```
@Bean
public AmazonS3 getS3Client() {
    BasicAWSCredentials cred = new BasicAWSCredentials(
        spacesAccess, spacesSecret);

    EndpointConfiguration epConfig = new EndpointConfiguration(
        "sgp1.digitaloceanspaces.com", "sgp1");

    return AmazonS3ClientBuilder.standard()
        .withEndpointConfiguration(epConfig)
        .withCredentials(new AWSStaticCredentialsProvider(cred))
        .build();
}
```

Build the S3 client with the  
credentials and endpoint



# PutObject into S3 Bucket

```
@PostMapping(consumes=MediaType.MULTIPART_FORM_DATA_VALUE)
public ResponseEntity<String> postUpload(@RequestPart Multipart file,
    @RequestPart String name, @RequestPart String email) {
```

```
    Map<String, String> userData = new HashMap<>();
    userData.put("name", name);
    userData.put("email", email);
    ...
```

} One or more file metadata to be associated with the object

```
    ObjectMetadata metadata = new ObjectMetadata();
    metadata.setContentType(file.getContentType());
    metadata.setContentLength(file.getSize());
    metadata.setUserMetadata(userData);
    ...
```

} Set the media type of the object

```
}
    ↑
    Associate the user data with the object
```



# PutObject into S3 Bucket

@Autowired

private AmazonS3 **s3**;

@PostMapping(consumes=MediaType.MULTIPART\_FORM\_DATA\_VALUE)

public ResponseEntity<String> postUpload(@RequestPart Multipart **file**,  
@RequestPart String name, @RequestPart String email) {

...

PutObjectRequest **putReq** = new PutObjectRequest("mybucket",  
"pet/%s".formatted(**file.getName()**), **file**.getInputStream(), **metadata**);  
**putReq** = **putReq.withCannedAcl**(CannedAccessControlList.**PublicRead**);

**s3.putObject**(**putReq**);

...

}

Create a put request with the bucket's name, the key name (eg `dog.png`), input stream and the metadata

Configure the object to be publically accessible

Upload the file to the S3 bucket



# PutObject into S3 Bucket

`https://mybucket.sgp1.digitaloceanspaces.com/pet/dog.png`



GET /pet/dog.png



200 OK

**Content-Length:** 123456

**Content-Type:** image/png

**X-Amz-Meta-name:** fred

**X-Amz-Meta-email:** fred@gmail.com

} From ObjectMetadata

} From user's metadata



# GetObject from S3 Bucket

Create a get object request with the bucket name and key

```
try {
    GetObjectRequest getReq = new GetObjectRequest("mybucket", "pet/dog.png");
    S3Object result = s3.getObject(getReq);
    ObjectMetadata metadata = result.getObjectMetadata();
    Map<String, String> userData = metadata.getUserMetadata();
    try (S3ObjectInputStream is = result.getObjectContent()) {
        byte[] buffer = is.getAllBytes();
        return ResponseEntity.status(HttpStatus.OK)
            .contentType(MediaType.parseMediaType(result.getContentType()))
            .header("X-name", userData.get("name"))
            .body(buffer);
    }
} catch (AmazonS3Exception ex) {
    // If key is not found
} catch (Exception ex) {
    // For S3ObjectInputStream
}
```

Get the object

Get the metadata and user data

Get the object's content from its InputStream

S3 client will throw an exception if the object is not found. Return a 404