

# S3 Image Handling

30/01/2026

I have HTML code in. Backend I bind that In frontend I have image in srv/ path I. Am able...  
Got it 👍 Earlier your frontend was reading images from local server path like /srv/... No...  
 This...  
Yes — this URL works without authentication ONLY if the S3 object (or bucket) is public ...  
If S3 is not public I mean to say I have Spring Boot project role based access so outer pa...  
Yes 👍 — if S3 is private, your Spring Boot role-based access system will handle it. You...  
Other way get image bytes and  
Yes 👍 — there are other ways to get image bytes from private S3 and send them to...



I have HTML code in. Backend I bind that In frontend I have image in `srv/` path I. Am able to read image file from system now image store in S3 bucket now what to do



Got it 👍

Earlier your frontend was reading images from local server path like ``/srv/...``

Now images are in **S3**, so you must **serve the image via URL**, not filesystem path.

Here are the correct approaches (based on real production practice):

### ✅ Option 1: Public S3 Object (Simplest)

If your images are **public** in S3:

#### Example S3 URL:

php-template

Copy code

```
https://<bucket-name>.s3.<region>.amazonaws.com/images/user1.png
```

#### Frontend HTML:

html

Copy code

```

```

#### Backend:

Store only the S3 path in DB:

text

Copy code

```
images/user1.png
```

And build URL dynamically:

java

Copy code

```
String url = "https://my-bucket.s3.ap-south-1.amazonaws.com/" + imagePath;
```

Bind to frontend.

## ✅ Option 2: Private S3 + Pre-Signed URL (Recommended for security)

### Backend (Spring Boot example):

java

Copy code

```
GeneratePresignedUrlRequest request =  
    new GeneratePresignedUrlRequest("my-bucket", "images/user1.png")  
        .withMethod(HttpMethod.GET)  
        .withExpiration(new Date(System.currentTimeMillis() + 3600 * 1000)); // 1 hour  
  
URL url = s3Client.generatePresignedUrl(request);
```

Send this URL to frontend.

### Frontend:

html

Copy code

```

```

## ✅ Option 3: Backend as Proxy API

### Backend API:

http

Copy code

```
GET /api/image/{id}
```

### Backend:

java

Copy code

```
S3Object obj = s3Client.getObject("my-bucket", key);  
return ResponseEntity.ok()  
    .contentType(MediaType.IMAGE_JPEG)  
    .body(obj.getObjectContent().readAllBytes());
```

### Frontend:

html

Copy code

```

```

## ❌ What you cannot do anymore

html

Copy code

```

```

✗ This only works for local filesystem, NOT S3

## ✓ Architecture Flow (Correct)

lua

Copy code

Frontend ---> Backend ---> S3

Frontend <--- URL/API <--- S3

## Best Practice Recommendation

Use case	Solution
Public images	Direct S3 URL
Secure images	Pre-signed URL
Controlled access	Backend proxy API

If you want, I can give you **full working Spring Boot code** for:

- ✓ Upload to S3
- ✓ Save path in DB
- ✓ Generate URL
- ✓ Bind image in frontend
- ✓ Download/View
- ✓ Access control
- ✓ Single service class for S3

Just tell me:

- 👉 Java version
- 👉 Spring Boot version
- 👉 Is bucket public or private
- 👉 Angular / React / JSP / Thymeleaf frontend?



```

```

This image path use without needing auth



Yes — **this URL works without authentication ONLY if the S3 object (or bucket) is public** ✓

html

Copy code

```

```

## 🔒 When it works without auth

If **either** of these is true:

1. Bucket is public
2. Object (`images/user1.png`) is public

Then anyone can access it directly via browser, `` tag, API, etc.

## ✓ How to make it public (correct way)

### Option A: Public bucket policy (recommended for static images)

json

Copy code

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "PublicReadGetObject",
      "Effect": "Allow",
      "Principal": "*",
      "Action": "s3:GetObject",
      "Resource": "arn:aws:s3:::my-bucket/*"
    }
  ]
}
```

### Option B: Public object ACL (single file)

bash

Copy code

```
aws s3api put-object-acl \
  --bucket my-bucket \
  --key images/user1.png \
  --acl public-read
```



## 🔒 When it will NOT work

If bucket/object is **private**, then ❌ this will fail:

html

Copy code

```

```

You will get:

❌ 403 Forbidden

❌ AccessDenied

Then you must use:

### ✅ Pre-signed URL

text

Copy code

```
https://my-bucket.s3.ap-south-1.amazonaws.com/images/user1.png?X-Amz-Signature=...
```

or

### ✅ Backend proxy API

html

Copy code

```

```

## Real Production Architecture

Type	Method
Public profile pics, banners, static assets	Direct S3 URL
Private docs, invoices, KYC	Pre-signed URL
Controlled access system	Backend proxy

## Best Practice

**Do not make full bucket public** if it has sensitive data.

Use **folder-based policy**:

json

Copy code

```
"Resource": "arn:aws:s3:::my-bucket/public/*"
```

Store images in:

swift

Copy code

```
public/images/user1.png
private/docs/aadhar.pdf
```

## ✅ Final Answer

Yes —

```

```

works **without auth only if the file is public**.

Otherwise → you must use **pre-signed URL** or **backend API**.

If you want, I'll give you a **secure production structure**:

- ✅ Public + private folders
- ✅ Bucket policy
- ✅ Spring Boot S3 config
- ✅ Upload service
- ✅ Auto URL generator
- ✅ Pre-signed URL service
- ✅ DB schema design

Just say: **"Give me full S3 production setup"** 🚀



If S3 is not public mean to say i have spring boot project role based access so outer page That like work



Yes 👍 — **if S3 is private**, your **Spring Boot role-based access** system will handle it.

You **must not expose direct S3 URLs** to frontend.

Instead, use **backend-controlled access**. This is exactly how secure systems work.

## ✅ Correct Architecture (Private S3 + Role-Based Access)

```
SCSS
```

Copy code

```
Frontend → Spring Boot (Auth + Roles) → S3 (Private)
```

S3 is **never accessed directly** by frontend.

## 🔒 Two Secure Solutions

### 🏆 BEST PRACTICE: Pre-Signed URL with Role Check

#### Flow:

1. User logs in
2. Role validated (`ADMIN`, `USER`, etc)
3. Backend generates **temporary S3 URL**
4. Frontend loads image using that URL

#### Spring Boot Example:

java

Copy code

```
@PreAuthorize("hasRole('USER') or hasRole('ADMIN')")
@GetMapping("/api/image-url/{key}")
public String getImageUrl(@PathVariable String key) {

    GeneratePresignedUrlRequest request =
        new GeneratePresignedUrlRequest("my-bucket", key)
            .withMethod(HttpMethod.GET)
            .withExpiration(new Date(System.currentTimeMillis() + 10 * 60 * 1000)); // 10 min

    URL url = s3Client.generatePresignedUrl(request);
    return url.toString();
}
```

#### Frontend:

html

Copy code

```

```

### 🏆 Backend Proxy API (Full Control)

#### Flow:

arduino

Copy code

Frontend → /api/image/123 → Spring Boot → S3 → Spring Boot → Frontend

#### Spring Boot:

java

Copy code



```
@PreAuthorize("hasRole('USER')")
@GetMapping("/api/image/{key}")
public ResponseEntity<byte[]> getImage(@PathVariable String key) throws Exception {

    S3Object obj = s3Client.getObject("my-bucket", key);
    byte[] bytes = obj.getObjectContent().readAllBytes();

    return ResponseEntity.ok()
        .contentType(MediaType.IMAGE_JPEG)
        .body(bytes);
}
```

## Frontend:

html

Copy code

```

```

## Comparison

Method	Security	Performance	Scalability
Pre-signed URL	★★★★★	★★★★★	★★★★★
Backend Proxy	★★★★	★★	★★
Public S3	✗	★★★★★	★★★★★

👉 **Pre-signed URL = industry standard**

## Role-Based Control Example

java

Copy code

```
@PreAuthorize("hasRole('ADMIN')")
@GetMapping("/api/admin/image-url/{key}")
```

java

Copy code

```
@PreAuthorize("hasRole('USER')")
@GetMapping("/api/user/image-url/{key}")
```

## Real Production Setup

cpp

Copy code

```
S3 (private)
├─ public/          (optional public assets)
├─ users/
│   └─ user123/profile.png
├─ docs/
│   └─ invoices/inv1.pdf
```

DB:

```
bash
```

Copy code

```
id | s3_key
1 | users/user123/profile.png
```

## ✗ Never Do This

```
html
```

Copy code

```

```

✗ breaks security model

## ✅ Final Answer

If S3 is **private** and you have **Spring Boot role-based access**:

✓ You must use:

- **Pre-signed URL** (recommended)
- OR
- **Backend proxy API**

This keeps:

- ✓ S3 private
- ✓ Access controlled by Spring Security
- ✓ Roles enforced
- ✓ No public exposure
- ✓ Secure production architecture

If you want, I'll give you a **complete production-grade Spring Boot setup**:

- ✓ IAM role policy
- ✓ S3 config
- ✓ Upload service
- ✓ Role-based APIs

- ✅ Pre-signed generator
- ✅ Security config
- ✅ JWT integration
- ✅ Folder strategy
- ✅ DB schema
- ✅ Frontend binding example

Just say: **“Give me full private S3 + Spring Security setup”** 🗝️🚀



Other way get image bytes and



Yes 👍 — there **are other ways to get image bytes** from private S3 and send them to frontend without exposing S3 at all.

Here are the main **secure alternatives** 📌

## ✅ Method 1: Send Image as Bytes (Direct Streaming API)

### Spring Boot:

java

Copy code

```
@PreAuthorize("hasRole('USER')")
@GetMapping("/api/image-bytes/{key}")
public ResponseEntity<byte[]> getImageBytes(@PathVariable String key) throws Exception {

    S3Object s3Object = s3Client.getObject("my-bucket", key);
    byte[] bytes = s3Object.getObjectContent().readAllBytes();

    return ResponseEntity.ok()
        .contentType(MediaType.IMAGE_PNG)    // or IMAGE_JPEG
        .body(bytes);
}
```

### Frontend:

html

Copy code

```

```

- ✅ Simple
- ✅ Secure

✗ Heavy load on backend for large traffic

## ✓ Method 2: Base64 Image (JSON API)

### Backend:

java

Copy code

```
@PreAuthorize("hasRole('USER')")
@GetMapping("/api/image-base64/{key}")
public Map<String, String> getImageBase64(@PathVariable String key) throws Exception {

    S3Object s3Object = s3Client.getObject("my-bucket", key);
    byte[] bytes = s3Object.getObjectContent().readAllBytes();

    String base64 = Base64.getEncoder().encodeToString(bytes);

    return Map.of("image", base64);
}
```

### Frontend:

html

Copy code

```

```

✓ Works in any frontend

✗ Bigger payload (base64 = ~33% size increase)

✗ Not good for large images

## ✓ Method 3: Byte Stream (Best backend-controlled way)

java

Copy code

```
@PreAuthorize("hasRole('USER')")
@GetMapping("/api/image-stream/{key}")
public void streamImage(@PathVariable String key, HttpServletResponse response) throws
Exception {

    S3Object s3Object = s3Client.getObject("my-bucket", key);
    InputStream in = s3Object.getObjectContent();

    response.setContentType("image/png"); // or jpeg
}
```

```
IOUtils.copy(in, response.getOutputStream());
}
```

## Frontend:

html

Copy code

```

```

- ✓ Streaming (memory efficient)
- ✓ Secure
- ✓ Production safe

## 🔥 Which one should YOU use?

Use Case	Best Method
High traffic system	Pre-signed URL
Strict security	Stream API
Small images only	Base64
Internal apps	Byte API

## 🧠 Real Architecture Choice

### Enterprise apps use:

- Auth in Spring Boot
- Role validation
- Pre-signed URL for files
- Stream API for sensitive docs
- Private S3 bucket

## ✓ Final Answer

Other secure ways to get image bytes from private S3:

- ✓ ``byte[] API``
- ✓ ``stream API``
- ✓ ``base64 API``
- ✓ ``pre-signed URL``

All work with **role-based Spring Security**.

If you want, I can give you **one clean service class** that supports:

- ✅ upload
- ✅ download bytes
- ✅ stream
- ✅ base64
- ✅ pre-signed
- ✅ delete
- ✅ role validation

Say: **"Give me S3 ImageService class"** 🚀