Pre-Signed URL API Creation Guide (Spring Boot + AWS S3)

Purpose

This document explains how we created the API endpoint **/api/products/images/presigned-url** using Spring Boot to generate a **Pre-Signed URL** for private S3 objects. The generated URL allows temporary access to view/download an image stored in AWS S3 without making the object public.

Why Pre-Signed URL is Needed

When you upload an image to S3, by default it is **private**. So if you open the S3 URL directly in browser, you get **AccessDenied**. A Pre-Signed URL is a secure temporary URL generated by your backend using AWS credentials.

Step 1: Add AWS SDK Dependency (pom.xml)

```
<dependency>
    <groupId>software.amazon.awssdk</groupId>
    <artifactId>s3</artifactId>
    <version>2.25.62</version>
</dependency>
```

Step 2: Add AWS Properties in application.yml

```
aws:
  s3:
    bucket-name: advann-product-images-2026
    region: ap-south-1
    access-key: ${AWS_ACCESS_KEY}
    secret-key: ${AWS_SECRET_KEY}
```

Step 3: Create S3Config (S3Client + S3Presigner)

```
@Configuration
public class S3Config {

    @Value("${aws.s3.access-key}")
    private String accessKey;

    @Value("${aws.s3.secret-key}")
    private String secretKey;

    @Value("${aws.s3.region}")
    private String region;

    @Bean
    public S3Client s3Client() {
        AwsBasicCredentials credentials = AwsBasicCredentials.create(accessKey, secretKey);

        return S3Client.builder()
                .region(Region.of(region))
                .credentialsProvider(StaticCredentialsProvider.create(credentials))
                .build();
    }
```

```
    @Bean
    public S3Presigner s3Presigner() {
        AwsBasicCredentials credentials = AwsBasicCredentials.create(accessKey, secretKey);

        return S3Presigner.builder()
                .region(Region.of(region))
                .credentialsProvider(StaticCredentialsProvider.create(credentials))
                .build();
    }
}
```

Step 4: Create Service Method generatePresignedUrl()

```
@Override
public String generatePresignedUrl(String fileUrl) {

    String baseUrl = "https://" + bucketName + ".s3." + region + ".amazonaws.com/";

    if (!fileUrl.startsWith(baseUrl)) {
        throw new InvalidFileException("Invalid S3 file URL: " + fileUrl);
    }

    String key = fileUrl.substring(baseUrl.length());

    GetObjectRequest getObjectRequest = GetObjectRequest.builder()
            .bucket(bucketName)
            .key(key)
            .build();

    GetObjectPresignRequest presignRequest = GetObjectPresignRequest.builder()
            .signatureDuration(Duration.ofMinutes(10))
            .getObjectRequest(getObjectRequest)
            .build();

    return s3Presigner.presignGetObject(presignRequest).url().toString();
}
```

Step 5: Create Controller API Endpoint

```
@GetMapping("/images/presigned-url")
public ResponseEntity<ApiResponse<String>> generatePresignedUrl(
        @RequestParam String fileUrl
) {
    String presignedUrl = s3Service.generatePresignedUrl(fileUrl);

    return ResponseEntity.ok(
            new ApiResponse<>(true, "Presigned URL generated successfully", presignedUrl)
    );
}
```

How to Call the API (Example)

```
GET http://localhost:8081/api/products/images/presigned-url
?fileUrl=https://advann-product-images-2026.s3.ap-south-1.amazonaws.com/products/full/abc.jpg
```

Response Example

```
{
```

```
  "success": true,
  "message": "Presigned URL generated successfully",
  "data": "https://advann-product-images-2026.s3.ap-south-1.amazonaws.com/products/full/abc.jpg?X-Amz-
}
```


Important Notes

1. The generated Pre-Signed URL is valid only for the configured time (example: 10 minutes).
2. If your original S3 URL contains spaces, encode them as **%20**.
3. Best practice is to store only the S3 object key in DB (products/full/abc.jpg) instead of full URL.