**Task 3 Documentation: Advanced S3 Storage for Product Images**

**Project Title:** Advanced S3 Storage for Product Images with CloudFront, Object Lambda & Security Automation

**Objective:** To design and partially implement a secure and efficient S3-based storage architecture for product images in the EverShop application. This includes private bucket access, encryption, lifecycle management, malware scanning, and integration with CloudFront for optimized content delivery.

**Step 1: Create Secure Private S3 Bucket** 1. Go to the **AWS Console > S3 > Create Bucket**. 2. Bucket name: evershop-product-images. 3. Region: us-east-1 (default). 4. Disable **Block all public access** → *Keep it enabled* for security. 5. Enable **Bucket versioning** → For data recovery. 6. Enable **Default encryption** (choose SSE-S3 or SSE-KMS). 7. Under **Bucket Ownership**, select *Bucket owner enforced*. 8. Click **Create Bucket**.

**Step 2: Upload Sample Product Images** 1. Go to **Objects > Upload > Add Files**. 2. Upload a few .jpg or .png images. 3. Confirm uploads appear successfully.

**Step 3: Generate Pre-Signed URLs for Access** Run the following AWS CLI command:

aws s3 presign s3://evershop-product-images/sample.jpg --expires-in 3600

This URL allows temporary access to the image for 1 hour.

**Step 4: Integrate with CloudFront** 1. Go to **AWS Console > CloudFront > Create Distribution**. 2. Origin domain: select the S3 bucket. 3. Enable **Origin Access Control (OAC)**. 4. Set **Viewer protocol policy** → Redirect HTTP to HTTPS. 5. Enable **Caching** and **Compress objects automatically**. 6. Create distribution.

**Step 5: Enable S3 Object Lambda (for Image Transformations)** 1. Go to **S3 Object Lambda Access Points** → Create. 2. Choose the source bucket evershop-product-images. 3. Create an access point. 4. Associate a Lambda function that creates thumbnails or watermarks.

Example Lambda function:

import boto3  
import io  
from PIL import Image  
  
def lambda\_handler(event, context):  
 s3 = boto3.client('s3')  
 object\_context = event["getObjectContext"]  
 request\_route = object\_context["outputRoute"]  
 request\_token = object\_context["outputToken"]  
 s3\_url = object\_context["inputS3Url"]  
  
 # Download the image  
 image = Image.open(requests.get(s3\_url, stream=True).raw)  
 image.thumbnail((200, 200))  
 buffer = io.BytesIO()  
 image.save(buffer, 'JPEG')  
  
 s3.write\_get\_object\_response(  
 Body=buffer.getvalue(),  
 RequestRoute=request\_route,  
 RequestToken=request\_token  
 )  
 return {"status\_code": 200}

**Step 6: Configure Lifecycle Policies** 1. Open **Management > Lifecycle Rules > Create rule**. 2. Rule name: tiering-and-archival. 3. Transition to Intelligent-Tiering after 30 days. 4. Transition to Glacier after 90 days. 5. Expire old versions after 365 days.

**Step 7: Enable S3 Inventory Reports** 1. Go to **Management > Inventory > Create Inventory Configuration**. 2. Destination bucket: same or another bucket. 3. Schedule: daily. 4. Enable encryption and CSV format.

**Step 8: Malware Scanning with Lambda + ClamAV (Conceptual)** 1. Create a new **Lambda function** triggered by S3 ObjectCreated event. 2. In the function code, use ClamAV to scan uploaded files. 3. If a file is infected → move it to quarantine/ folder.

Sample code snippet:

import os  
import boto3  
import subprocess  
  
def lambda\_handler(event, context):  
 s3 = boto3.client('s3')  
 bucket = event['Records'][0]['s3']['bucket']['name']  
 key = event['Records'][0]['s3']['object']['key']  
   
 s3.download\_file(bucket, key, '/tmp/file')  
 result = subprocess.run(['clamscan', '/tmp/file'], stdout=subprocess.PIPE)  
  
 if b'Infected files: 0' not in result.stdout:  
 s3.copy\_object(Bucket=bucket, CopySource={'Bucket': bucket, 'Key': key}, Key='quarantine/'+key)  
 s3.delete\_object(Bucket=bucket, Key=key)

**Step 9: Testing and Validation** - Upload a file → confirm it triggers the Lambda. - Access an image using the presigned URL. - Verify CloudFront delivers images securely. - Check inventory reports and lifecycle transitions after a few days.