



CloudX Associate: AWS for Testers

S3

---

**Legal Notice:**

This document contains privileged and/or confidential information and may not be disclosed, distributed or reproduced without the prior written permission of EPAM®.

---

**Error! Unknown document property name.**

## TASK 5 – S3

### 1. DEPLOY THE IMAGE APPLICATION

Deploy the **cloudximage** CDK stack: [deployment instructions](#).

### 2. DEPLOYMENT VALIDATION

Create a manual/automated deployment validation test suite that covers the following requirements:

CXQA-S3-01: Instance requirement:

- The application is deployed in the public subnet and should be accessible by HTTP from the internet via an Internet gateway by public IP address and FQDN.
- The application instance should be accessible by SSH protocol.
- The application should have access to the S3 bucket via an IAM role.

CXQA-S3-02: S3 bucket requirements:

- Name: cloudximage-imagestorebucket{unique id}
- Tags: cloudx: qa
- Encryption type: SSE-S3
- Versioning: disabled
- Public access: no.

#### 2.1 Testing Tools:

- AWS Console
- AWS CLI
- AWS SDK (for automated tests).
- Application Swagger OpenAPI documentation
- Postman / CURL
- SSH client

### 3. APPLICATION FUNCTIONAL VALIDATION

Create a manual/automated test suite that covers the following application API functions:

- CXQA-S3-03: Upload images to the S3 bucket
- CXQA-S3-04: Download images from the S3 bucket
- CXQA-S3-05: View a list of uploaded images
- CXQA-S3-06: Delete an image from the S3 bucket

#### 3.1 Testing Tools:

- Application Swagger OpenAPI documentation
- Postman / CURL
- SSH client

Execute test cases and verify that requirements are met.

**Legal Notice:** This document contains privileged and/or confidential information and may not be disclosed, distributed or reproduced without the prior written permission of EPAM®.

#### 4. REGRESSION TESTING

- Deploy **version 2** of **cloudximage** application [deployment instructions](#).
- Execute deployment and functional validation test suites against new application version deployment.
- Create bug/root cause report for the found regression issues.

#### 5. ENVIRONMENT CLEAN-UP

Delete the **cloudxiam** application stack and clean up the environment: [clean-up instructions](#).

#### 6. SUBMIT RESULTS

Upload the home task artifacts (screenshots, test cases/links to automated tests code in the git repository) to Learn Portal and change the task status to „Needs Review“.

#### IMPORTANT THINGS TO KEEP IN MIND

- Once you create AWS Account, setup Multi-factor Authentication!
- Do NOT share your account!
- Do NOT commit your account Credentials to the Git repository!
- Terminate/Remove (destroy) all created resources/services once you finish the module (or the learning for the day)!
- Please Do NOT forget to delete NAT Gateway if you used it!
- Do NOT keep the instances running if you don't use it!
- Carefully keep track of billing and working instances so you don't exceed limits!

## AWS S3 THEORY

### What are buckets in S3?

In Amazon Simple Storage Service (Amazon S3), buckets are the fundamental containers that store data. They function as top-level folders and are used to organize and manage objects (files) within Amazon S3. Here are some key points about buckets in S3:

- **Unique Names:** Bucket names must be globally unique across all of AWS. This uniqueness is enforced to ensure that each bucket's URI is unique and accessible on the internet.
- **Region-specific:** Buckets are region-specific, meaning they are created in a specific AWS region. Objects stored in a bucket are stored in that region, and data transfer costs may apply if accessed from a different region.
- **Access Control:** Access to buckets and objects within them is controlled through a combination of policies, access control lists (ACLs), and bucket policies. You can grant permissions to other AWS accounts or to anonymous users, or restrict access entirely.
- **Storage Classes:** You can specify the storage class for objects within a bucket, such as Standard, Intelligent-Tiering, Standard-IA (Infrequent Access), One Zone-IA, Glacier, and others. Each storage class has different pricing and performance characteristics.
- **Lifecycle Policies:** You can configure lifecycle policies on buckets to automatically transition objects to different storage classes or delete them after a specified period. This helps optimize storage costs and compliance requirements.
- **Logging and Monitoring:** Amazon S3 provides logging and monitoring features that allow you to track access to your buckets and objects, monitor storage usage, and set up alerts for certain events.
- **Static Website Hosting:** You can configure a bucket to host a static website by enabling static website hosting. This allows you to serve static content (HTML, CSS, JavaScript, etc.) directly from your bucket using a custom domain or the default Amazon S3 domain.

### What does S3 replication do? What is it for?

Amazon S3 replication is a feature that automatically replicates objects from one S3 bucket to another within the same or different AWS region. This replication process helps in achieving data resilience, compliance, and disaster recovery objectives. Here's what S3 replication does and what it's for:

- **Data Resilience and Durability:** S3 replication enhances the durability of data by creating redundant copies of objects across multiple buckets. In the event of hardware failures, data corruption, or other unforeseen issues affecting one bucket, the replicated objects in another bucket ensure data availability and integrity.
- **Geo-Redundancy:** Replicating objects across multiple AWS regions allows you to achieve geo-redundancy. By storing copies of data in different geographic locations, you can mitigate the risk of data loss due to regional disasters, such as earthquakes, floods, or power outages.
- **Reduced Latency:** By replicating objects closer to end-users or application servers in different regions, you can reduce data access latency and improve performance. This is particularly beneficial for global applications or services that serve users across different geographical regions.
- **Cross-Region Disaster Recovery:** S3 replication facilitates cross-region disaster recovery strategies. In the event of a regional outage or disaster affecting one AWS region, you can failover to replicated data in another region to maintain business continuity and minimize downtime.
- **Versioning and Object Locking:** S3 replication supports versioning and object locking features, allowing you to replicate both current versions and previous versions of objects. This helps protect against accidental deletions, unauthorized modifications, or ransomware attacks.

### What steps are required to gain AWS CLI access to an S3 bucket?

To gain AWS CLI access to an S3 bucket, the following steps can be executed:

**Legal Notice:** This document contains privileged and/or confidential information and may not be disclosed, distributed or reproduced without the prior written permission of EPAM®.

1. Install AWS CLI
2. Configure AWS CLI

Once installed, you need to configure the AWS CLI with your AWS credentials.

```
$ aws configure
```

```
AWS Access Key ID [None]: your-access-key-id
```

```
AWS Secret Access Key [None]: your-secret-access-key
```

```
Default region name [None]: your-default-region
```

```
Default output format [None]: json
```

3. Verify Configuration: After configuring the AWS CLI, you can verify your configuration by running the `aws s3 ls` command, which lists the buckets in your AWS account. If the configuration is correct, you should see a list of buckets.

```
$ aws s3 ls
```

4. Access S3 Bucket

Once the AWS CLI is configured, you can access the S3 bucket using various commands such as `aws s3 ls`, `aws s3 cp`, `aws s3 sync`, etc. For example, to list the contents of a bucket, you can use the `aws s3 ls s3://bucket-name` command.

```
$ aws s3 ls s3://your-bucket-name
```

### How is it possible to control access to S3 resources?

Access to Amazon S3 resources can be controlled through various mechanisms provided by AWS Identity and Access Management (IAM) and S3 itself. Here are the main methods for controlling access to S3 resources:

- **IAM Policies:** IAM policies define permissions for IAM identities (users, groups, roles) to access S3 resources. You can attach IAM policies to IAM identities to grant or deny permissions for specific S3 actions (e.g., `s3:GetObject`, `s3:PutObject`) on specific S3 buckets or objects.
- **Bucket Policies:** S3 bucket policies are JSON documents attached directly to S3 buckets. They allow you to control access to the bucket at the bucket level. Bucket policies can be used to grant cross-account access, enforce SSL connections, restrict access based on IP addresses, and more.
- **Access Control Lists (ACLs):** S3 ACLs are legacy access control mechanisms that can be used to grant basic read/write permissions to other AWS accounts or to the public. ACLs are less flexible than IAM policies and bucket policies, so it's generally recommended to use IAM policies and bucket policies instead.
- **Cross-Origin Resource Sharing (CORS):** CORS allows you to control which web applications can access your S3 resources from different domains. You can configure CORS rules on your S3 buckets to specify which origins are allowed to access your resources.
- **Object Locking:** S3 Object Lock allows you to prevent objects from being deleted or modified for a specified retention period. This helps to enforce compliance requirements and prevent accidental deletion or modification of critical data.
- **Bucket Versioning:** Enabling versioning on an S3 bucket allows you to preserve, retrieve, and restore every version of every object stored in the bucket. This can help protect against accidental deletions or overwrites by providing a history of changes to objects.
- **Pre-signed URLs:** Pre-signed URLs are URLs that grant temporary access to specific S3 objects. They are generated using your AWS credentials and can be used to provide temporary access to objects without requiring the requester to have AWS credentials.

### What are versions in S3? What does it mean to delete an object in S3 when versioning is enabled?

In Amazon S3, versioning is a feature that allows you to keep multiple versions of an object in the same bucket. Each time you overwrite an existing object or delete it, a new version of the object is created. Enabling versioning provides an additional layer of protection and allows you to recover previous versions of objects in case of accidental deletions or modifications.

**Legal Notice:** This document contains privileged and/or confidential information and may not be disclosed, distributed or reproduced without the prior written permission of EPAM®.

Here's how versions work in S3:

- **Object Versioning:** When you upload an object to an S3 bucket with versioning enabled, the initial version of the object is created. Subsequent uploads or overwrites of the same object result in new versions being created. Each version is assigned a unique version ID.
- **Object Metadata:** Each version of an object in S3 has its own metadata, including metadata such as storage class, encryption settings, and custom user metadata. You can retrieve metadata for a specific version of an object using its version ID.
- **Deleting Objects:** When you delete an object in an S3 bucket with versioning enabled, the object is not immediately removed. Instead, a delete marker is added to indicate that the object has been logically deleted. The object and its versions remain in the bucket, but they are hidden from normal listing operations.
- **Recovery:** If you need to recover a deleted object or a previous version of an object, you can do so by either deleting the delete marker or by specifying the version ID of the object you want to restore. Once the delete marker is removed or a specific version is requested, the object or version becomes accessible again.
- **List Operations:** When you list objects in an S3 bucket with versioning enabled, both current objects and their previous versions are included in the results. You can use version-specific parameters in list operations to filter and retrieve specific versions of objects.
- **Storage Costs:** Storing multiple versions of objects in an S3 bucket incurs additional storage costs. Each version of an object consumes storage space, so enabling versioning may increase your storage costs over time.

### How nested file hierarchies are represented in S3?

In Amazon S3, nested file hierarchies are represented using object keys, which are essentially the paths of the objects within the bucket. S3 is an object storage service, not a traditional file system, so it doesn't have directories or folders in the same sense as a file system. Instead, the concept of a nested file hierarchy is achieved through the naming convention of object keys.

- **Object Keys:** Each object stored in an S3 bucket is identified by a unique key. The key serves as the object's identifier and is used to retrieve, store, and organize objects within the bucket. Object keys resemble file paths or URLs and can include slashes (/) to represent hierarchical relationships.
- **Prefixes:** In S3, prefixes are used to mimic the structure of directories or folders within a bucket. A prefix is essentially a string of characters that appears before the object key. When you create an object with a key that includes slashes (/), S3 interprets the text before each slash as a prefix.
- **Delimiter:** The delimiter is a character used to group objects with similar prefixes. By default, the delimiter is set to /, which means that objects with the same prefix up to the delimiter are grouped together. When listing objects in a bucket, the delimiter is used to organize and display objects hierarchically.
- **Listing Objects:** When you list objects in an S3 bucket, the objects are organized hierarchically based on their prefixes. Objects with the same prefix are grouped together and displayed as if they were in the same directory or folder.
- **Virtual Directories:** Although S3 doesn't have physical directories, it supports the concept of virtual directories through prefixes. You can create a nested file hierarchy by organizing objects with similar prefixes, effectively simulating the structure of directories and subdirectories within the bucket.

### What does S3 store inside its objects?

In Amazon S3, each object represents a piece of data that you store in a bucket. An S3 object consists of the following components:

- **Data:** The main component of an S3 object is the actual data that you upload to the bucket. This can be any type of data, such as text files, images, videos, documents, or binary files. S3 supports objects of virtually any size, from a few bytes to multiple terabytes.

**Legal Notice:** This document contains privileged and/or confidential information and may not be disclosed, distributed or reproduced without the prior written permission of EPAM®.

- **Object Key:** An object key is a unique identifier for the object within the bucket. It serves as the address for accessing the object and is used to retrieve, store, and organize objects within the bucket. The object key is specified when you upload the object to S3 and can include slashes (/) to represent hierarchical relationships.
- **Metadata:** S3 allows you to attach metadata to objects, which provides additional information about the object. Metadata includes key-value pairs such as content type, content encoding, cache control, and custom user metadata. Metadata can be specified when uploading an object or modified later using the AWS Management Console or the AWS CLI.
- **Object Version ID:** If versioning is enabled for the bucket, each object has a unique version ID that distinguishes different versions of the same object. Version IDs are assigned when objects are uploaded or overwritten and are used to retrieve specific versions of objects.
- **Object URL:** Every object in S3 is accessible via a unique URL that consists of the bucket name, the object key, and, optionally, the version ID. The object URL allows you to directly access the object over HTTP or HTTPS protocols from anywhere on the internet.
- **Storage Class:** Objects in S3 are stored in different storage classes, each designed for different access patterns and cost requirements. The storage class determines the durability, availability, and cost of storing the object. Common storage classes include Standard, Standard-IA (Infrequent Access), One Zone-IA, Glacier, and Intelligent-Tiering.
- **Access Control Settings:** S3 allows you to control access to objects using access control lists (ACLs), bucket policies, and IAM policies. You can grant permissions to specific AWS accounts, IAM users, or anonymous users, and specify whether they have read, write, or delete access to the object.

### Why S3 is better than a physically maintained file server?

Amazon S3 (Simple Storage Service) offers several advantages over a physically maintained file server:

- **Scalability:** S3 is highly scalable and can accommodate virtually unlimited amounts of data without the need for capacity planning or hardware provisioning. It automatically scales to meet your storage requirements, making it suitable for storing large amounts of data, including petabytes or even exabytes.
- **Durability and Availability:** S3 provides high durability and availability for stored objects. It replicates data across multiple data centers within an AWS region, ensuring redundancy and resilience against hardware failures, natural disasters, and other unforeseen events. S3 offers a durability of 99.999999999% (11 nines), which is significantly higher than what most organizations can achieve with on-premises file servers.
- **Cost-Effectiveness:** S3 offers a pay-as-you-go pricing model, where you only pay for the storage you use and any data transfer costs incurred. There are no upfront costs or long-term commitments, making it cost-effective for organizations of all sizes. Additionally, S3 offers various storage classes with different pricing tiers, allowing you to optimize costs based on your access patterns and retention requirements.
- **Global Accessibility:** S3 is accessible over the internet from anywhere in the world, allowing you to store and retrieve data globally with low latency. This makes it suitable for applications and services with global user bases or distributed teams. Additionally, S3 integrates seamlessly with other AWS services, enabling you to build scalable and resilient cloud-based architectures.
- **Security and Compliance:** S3 offers robust security features to protect your data, including encryption at rest and in transit, access control policies, and integration with AWS Identity and Access Management (IAM). It also supports compliance certifications such as SOC, PCI DSS, HIPAA, and GDPR, making it suitable for storing sensitive or regulated data.
- **Reliability and Performance:** S3 is designed for high reliability and performance, with low-latency access to stored objects and high throughput for data transfers. It is backed by the AWS global infrastructure, which includes multiple redundant network connections, advanced monitoring and alerting capabilities, and 24/7 support..

## TASK IMPLEMENTATION

### 1. DEPLOYMENT VALIDATION

#### 1.1 TAF

Link to repo with task implemented: <https://github.com/Chubaka2612/CloudX/pull/3/files>

The TAF solution was extended:

- CloudX.Auto.Tests – the following test suite was added:
  - S3 – covers CXQA-S3-01, CXQA-S3-02, CXQA-S3-03, CXQA-S3-04, CXQA-S3-05, CXQA-S3-06 scenarios;
- S3Service class – wrapper on AmazonS3Client for interaction with s3 resources;
- S3BucketSteps class – static class for interaction with s3 objects – extension for S3Service.

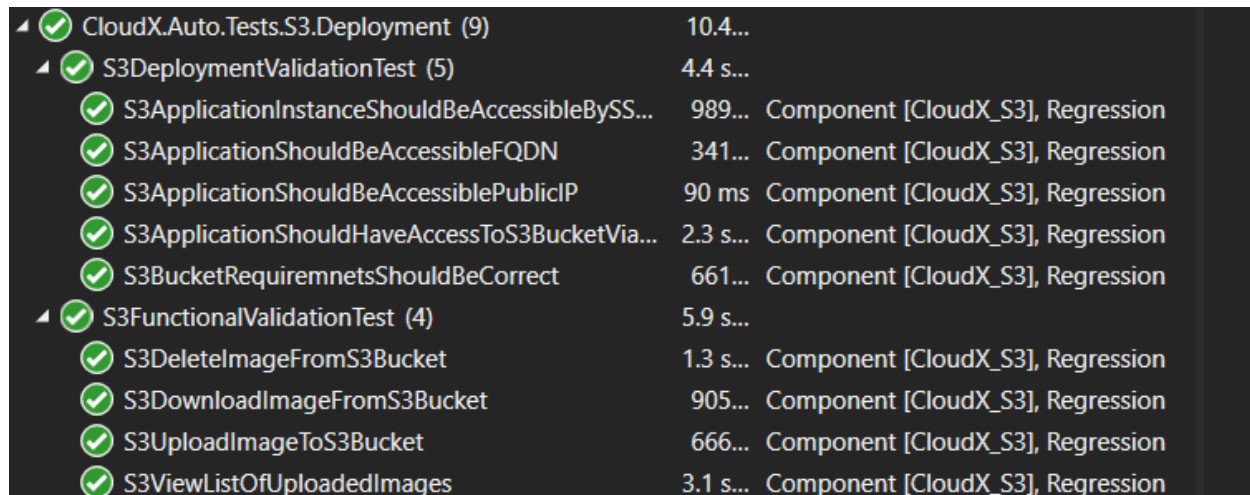
In total includes 9 new tests.

Used tools:

1. NUnit – test runner
2. AWS SDK for .NET – API for AWS resource interaction
3. Renci.SshNet – Library for ssh checks

#### 1.2 Deployment and Functional test execution

For cloudimage v=1 there are no issues found. Results are below:



▲ ✓ CloudX.Auto.Tests.S3.Deployment (9)	10.4...	
▲ ✓ S3DeploymentValidationTest (5)	4.4 s...	
✓ S3ApplicationInstanceShouldBeAccessibleBySS...	989...	Component [CloudX_S3], Regression
✓ S3ApplicationShouldBeAccessibleFQDN	341...	Component [CloudX_S3], Regression
✓ S3ApplicationShouldBeAccessiblePublicIP	90 ms	Component [CloudX_S3], Regression
✓ S3ApplicationShouldHaveAccessToS3BucketVia...	2.3 s...	Component [CloudX_S3], Regression
✓ S3BucketRequiremntsShouldBeCorrect	661...	Component [CloudX_S3], Regression
▲ ✓ S3FunctionalValidationTest (4)	5.9 s...	
✓ S3DeleteImageFromS3Bucket	1.3 s...	Component [CloudX_S3], Regression
✓ S3DownloadImageFromS3Bucket	905...	Component [CloudX_S3], Regression
✓ S3UploadImageToS3Bucket	666...	Component [CloudX_S3], Regression
✓ S3ViewListOfUploadedImages	3.1 s...	Component [CloudX_S3], Regression

Pic 1 – test results execution for cloudimage v = 1

Link to test execution: [demo](#)

#### 1.3 Regression Deployment and Functional test execution

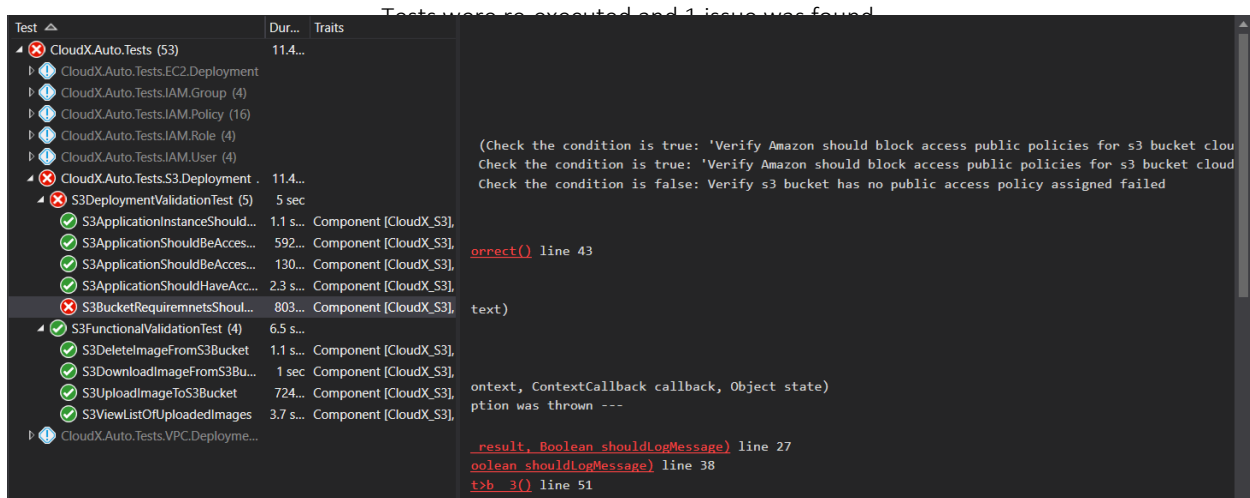
The cloudimage v=2 is deployed. Two tests failed.

One deployment test detects 1 issue.

#### Legal Notice:

This document contains privileged and/or confidential information and may not be disclosed, distributed or reproduced without the prior written permission of EPAM®.

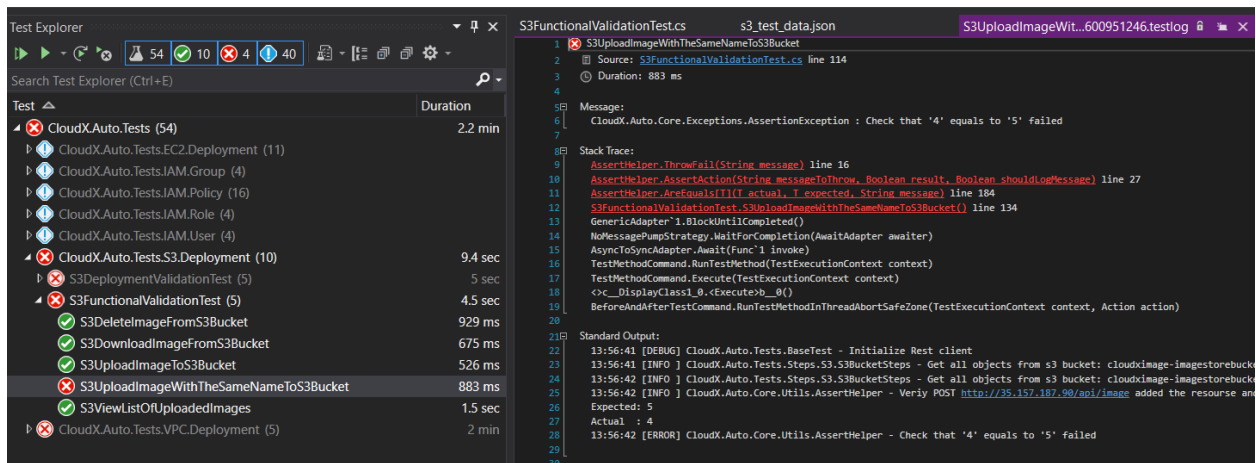




Pic 2 – deployment tests are executed on cloudximage v=2 and 1 test failed

Link to regression test execution: [demo](#)

One functional test detects 1 issue.



Pic 3 – functional tests are executed on cloudximage v=2 and 1 test failed

## 1.4 Bug report

**Summary:** [Deployment] The S3 bucket should not have public access

Type: Bug

Priority: Critical

Affects Version: 2

Component: S3

Labels: Regression

**Pre-conditions:**

**Legal Notice:**

This document contains privileged and/or confidential information and may not be disclosed, distributed or reproduced without the prior written permission of EPAM®.

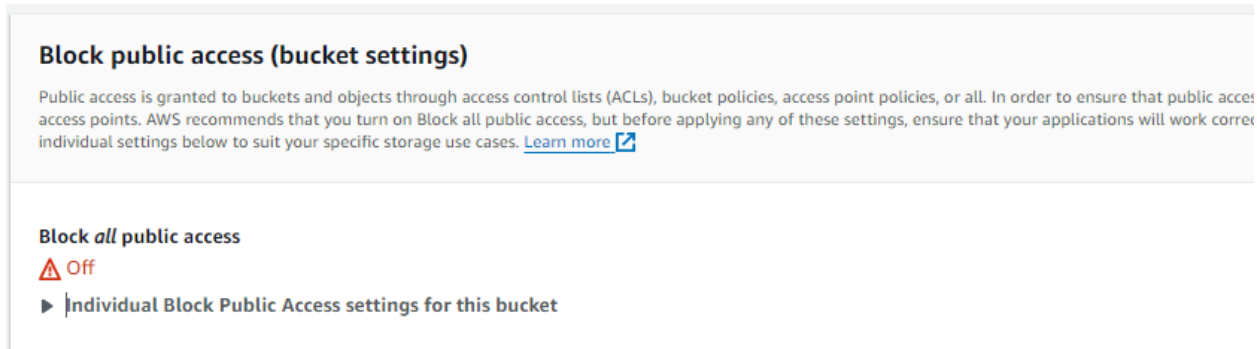
Error! Unknown document property name.

1. cloudximage is deployed on the EC2 instance
2. The s3 bucket is configured

**Steps to reproduce:**

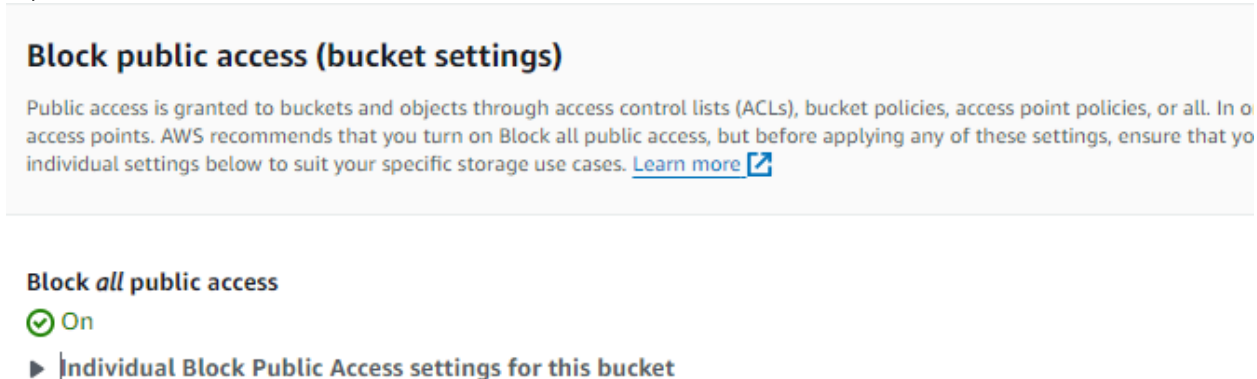
1. Get S3 bucket's **PublicAccessBlockConfiguration** (use any convenient way: CLI, S3 Dashboard, .NET SDK)
2. Observe **BlockPublicPolicy** value.

**Actual results:** BlockPublicPolicy has value 'false' ('Off' on UI).



Pic 4 – Actual 'Block public access' is 'Off'

**Expected results:** BlockPublicPolicy should have value 'true' ('On' on UI).



Pic 5 – Expected 'Block public access' is 'On'

**Summary: [Functional]** User is not able to upload the image with the same name to s3 bucket

Type: Bug  
Priority: Major  
Affects Version: 2  
Component: S3  
Labels: Regression

**Pre-conditions:**

1. cloudximage is deployed on the EC2 instance
2. The s3 bucket is configured

**Legal Notice:** This document contains privileged and/or confidential information and may not be disclosed, distributed or reproduced without the prior written permission of EPAM®.

**Error! Unknown document property name.**

**Steps to reproduce:**

1. Call POST <http://{publicip}/api/image> with image named "test\_photo.jpg" (use swagger, postman or another appropriate option)
2. Call one more time POST <http://{publicip}/api/image> with image named the same "test\_photo.jpg"
3. Observe results in s3 bucket.

**Actual results:**

1. POST method returns 200 (OK) and has id of s3 resource created in response.
2. In S3 bucket there is only one image with "test\_photo.jpg". The image name doesn't have UUID as a prefix.

**Expected results:**

1. POST method returns 200 (OK) and has id of s3 resource created in response.
2. In S3 bucket there are 2 images with "test\_photo.jpg". The images name has UUID as a prefix.