Amazon Simple Storage Service (S3): S3 Security

Andru Estes
Principal Author
andru-estes

Controlling S3 Access with Bucket Policies

Reviewing Policy Types

Identity-based

Attached to IAM principals to control permissions and access to AWS resources and services

Resource-based

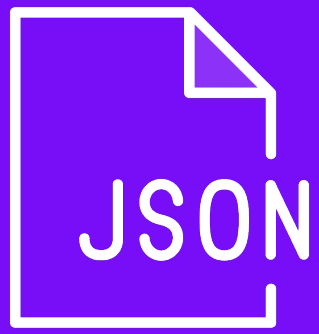
Attached to actual AWS resources themselves to help control permission and access

Reviewing Policy Types

Let's discuss Bucket Policies, which are classified as a resource-based policy!

Resource-based

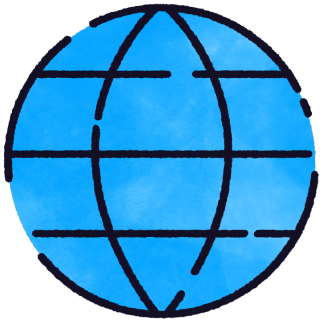
Attached to actual AWS resources themselves to help control permission and access



S3 Bucket Policy

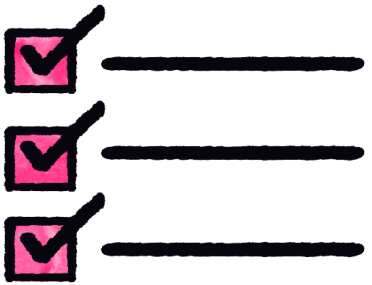
JSON-formatted AWS IAM resource-based policy that is attached to a bucket to control access to the bucket and its objects

S3 Bucket Policy Use Cases



Allowing public access to your Amazon S3 bucket and objects

Requiring objects are encrypted when being uploaded and that TLS is used for transport



Allowing another AWS account to access the bucket and its objects

Remember These Facts

Amazon S3 buckets and objects are private by default

You must explicitly allow access to the buckets and objects using a policy

Want to ensure that no objects are made public?

AWS has a “Block Public Access” easy setting that you can turn on!

Block Public Access

Useful for easily denying all public access to your account's buckets

Can be used at both the bucket level and the account level

More than likely you will want to turn this on immediately!

Breaking down an S3 Bucket Policy

You must know how to interpret a bucket policy!

Example Bucket Policy

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

Statement: List/array of individual statements

Example Bucket Policy

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

Effect: Allowing or denying the actions

Example Bucket Policy

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

Principal: Which IAM principal?

This example actually opens the bucket up to the world!

No IAM authentication is required with this policy. Anonymous users can access this.

Be careful!

Example Bucket Policy

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

Action: The API actions allowed or denied.

These must be specific to Amazon S3.

Example Bucket Policy

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

Resource: The S3 resources we are granting permissions for.

You need to know the difference between these two resource formats listed!

Example Bucket Policy

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

Reviewing Bucket and Object Resources

These look the same, but you must know how they differ!

arn:aws:s3:::pluralsight-bucket ← You grant bucket level permissions for this resource.

arn:aws:s3:::pluralsight-bucket/* ← You grant object level permissions for this resource.

Reviewing Bucket and Object Resources

These look the same, but you must know how they differ!

arn:aws:s3:::pluralsight-bucket ← You grant bucket level permissions for this resource.

arn:aws:s3:::pluralsight-bucket/* ← You grant object level permissions for this resource.

Bucket or Object Resource Action Examples

Bucket	vs.	Object
s3:ListBucket		s3:GetObject
s3:ListBucketVersions		s3:PutObject
s3:CreateBucket		s3:DeleteObject
s3:DeleteBucket		s3:RestoreObject
s3:GetBucketTagging		s3:PutObjectTagging

Tricky Scenario

Based on the policy combination below, could this IAM user access the S3 bucket objects?

Yes!

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

Bucket Policy

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "AllowAllEC2Actions",
            "Effect": "Allow",
            "Action": "ec2:*",
            "Resource": "*"
        }
    ]
}
```

IAM User Permissions Policy

Tricky Scenario

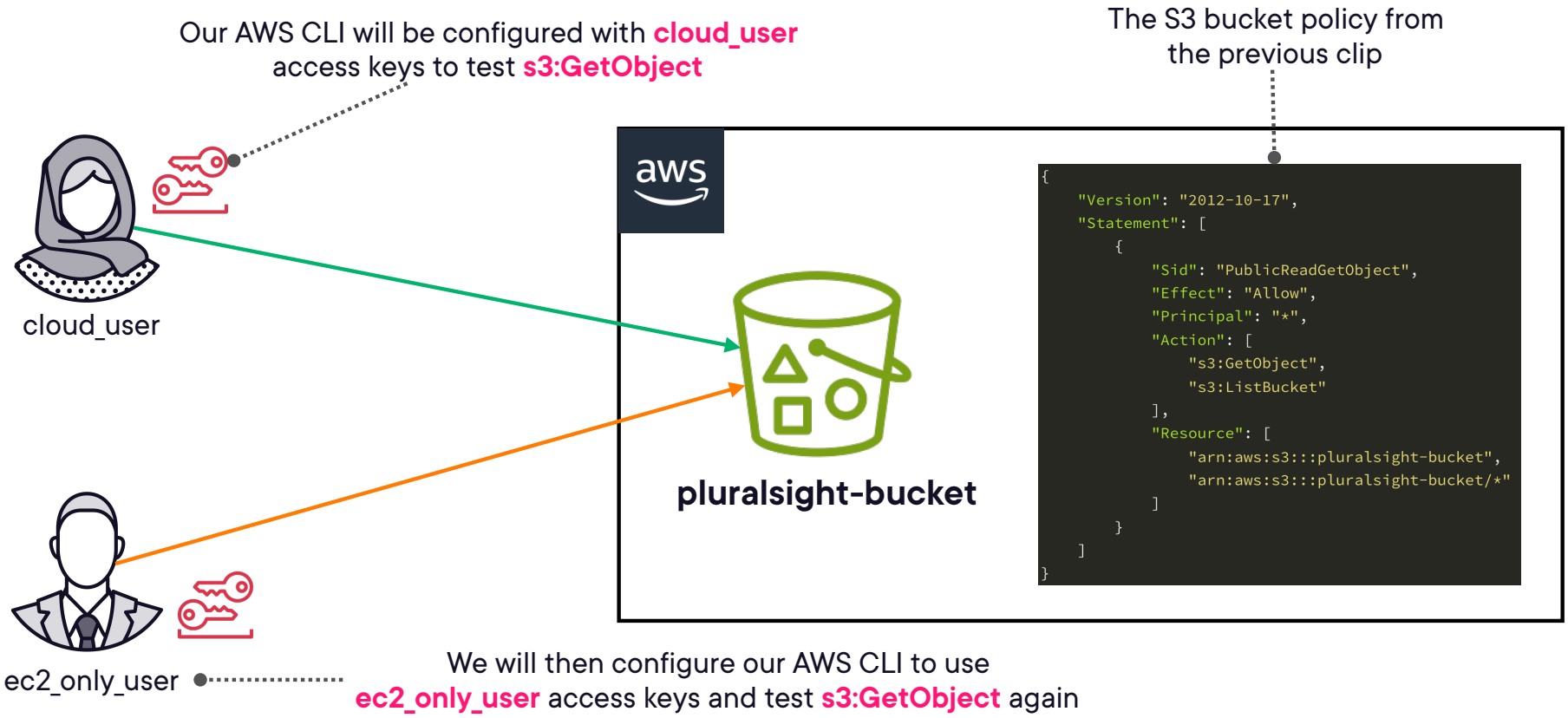
Let's break it down:

- We have an IAM user
- There is no explicit allow statement for S3 in the IAM user's permissions policy
- The bucket policy was set to open for all principals (*)
- There is no explicit deny in either the bucket policy or the IAM policy

This results in the IAM user being allowed to access the bucket objects because of how the S3 service operates!

The policy grants public access, which means IAM authentication is not needed!

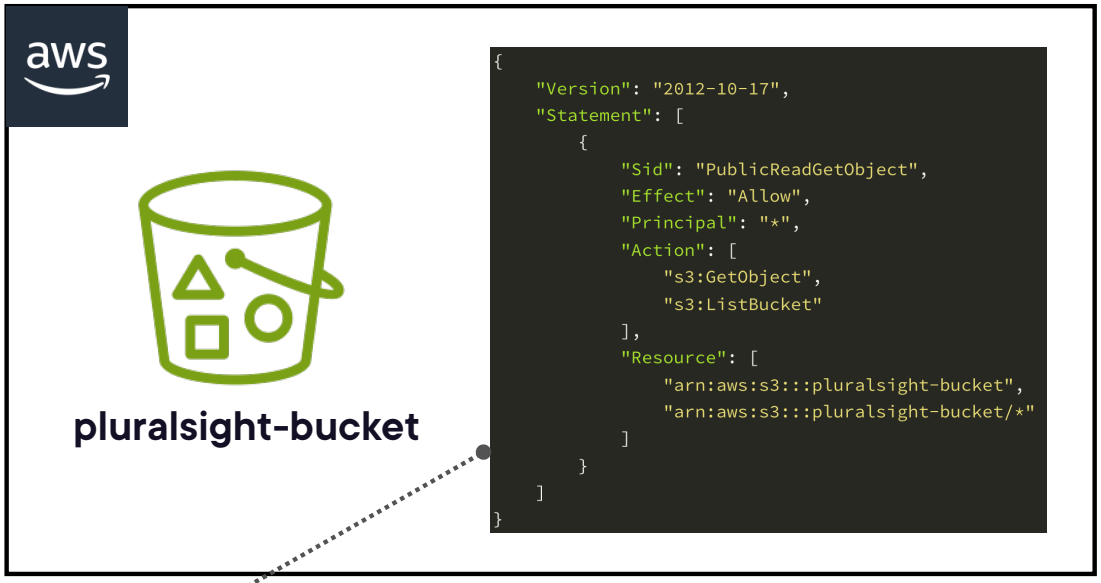
Demo: Implement a S3 Bucket Policy



Demo: Implement a S3 Bucket Policy

cloud_user

ec2_only_user



We will lock down the bucket policy to only work for our cloud_user IAM user, and then we will test again

Bucket and Object Access Control Lists in S3

Access Control List (ACL)

An ACL is a list of grants identifying the grantee and the permission granted. ACLs grant basic read or write permissions to other AWS accounts.

Citation: https://docs.aws.amazon.com/AmazonS3/latest/userguide/access-management.html

S3 ACLs Overview

These use an Amazon S3–specific XML schema

Similar to bucket policies, these are also a type of AWS IAM policy

Bucket ACLs are used to manage access to the bucket

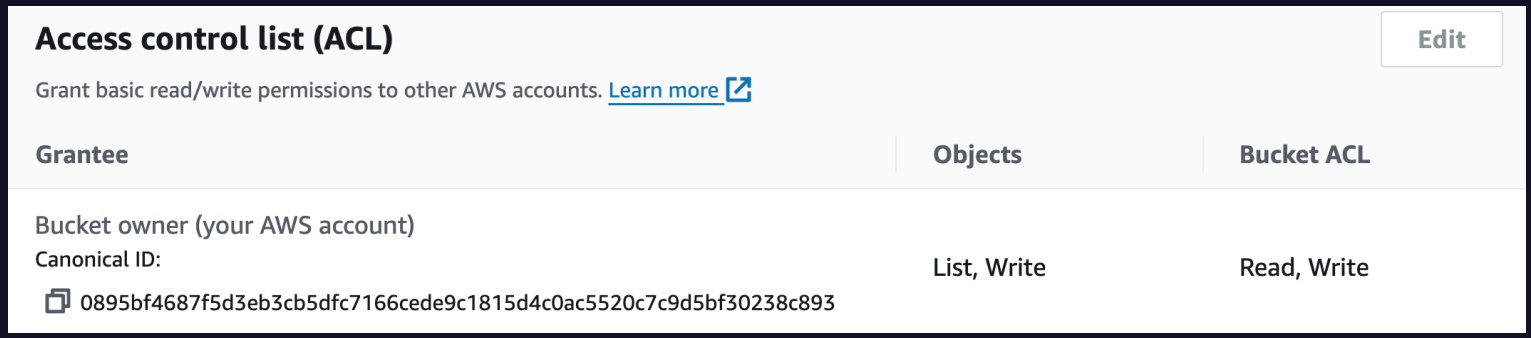
Object ACLs are used to manage access to the objects

One bucket ACL for the entire bucket, but one object ACL per object

AWS Account Canonical User ID

A unique, long-string identifier assigned to each AWS account. It is an obfuscated form of the AWS account ID that is frequently used within S3 ACLs!

Amazon S3 Canonical ID Example



Screenshot from one of our AWS sandbox environments

Amazon S3 Canned ACLs

Amazon S3 supports a set of predefined grants, known as canned ACLs.

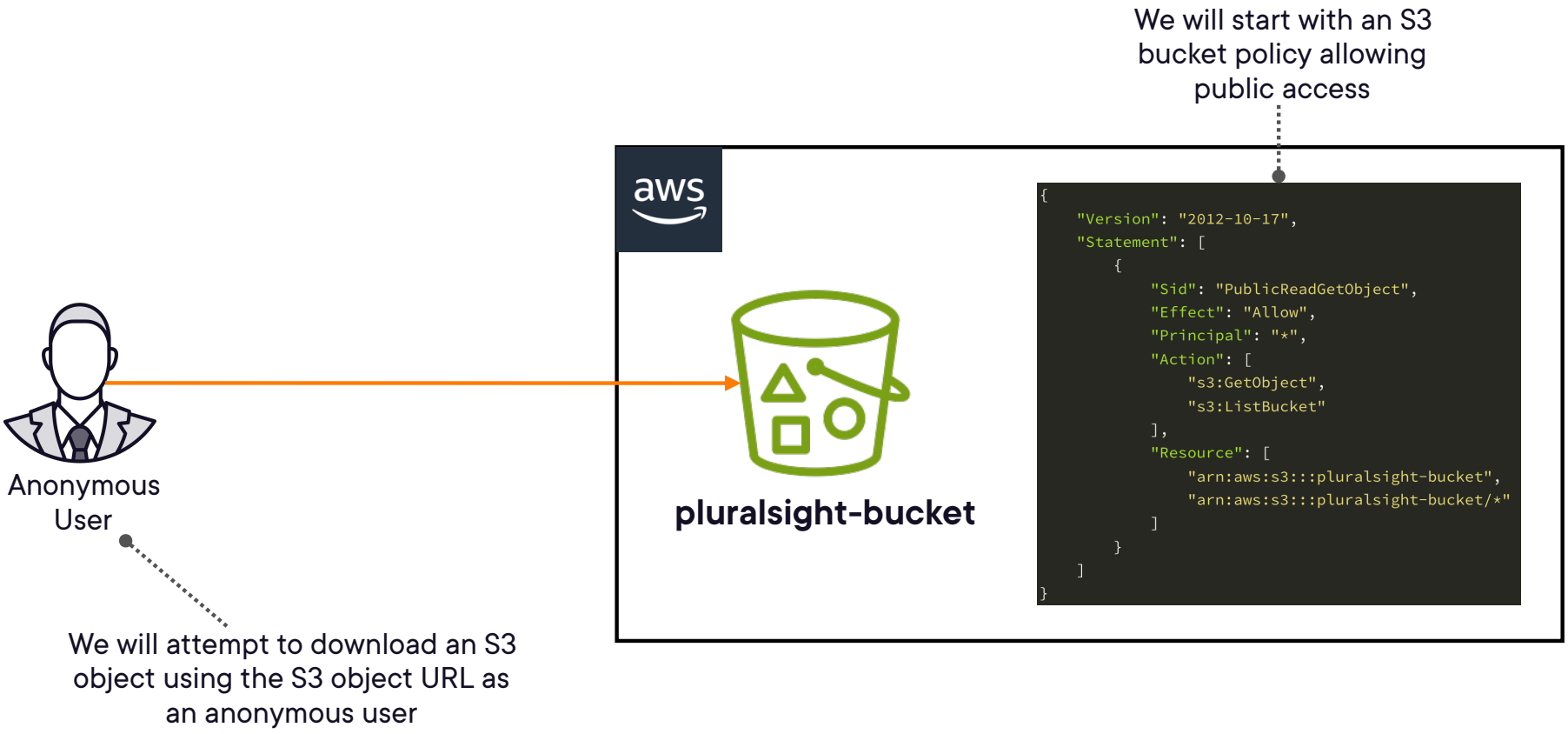
Amazon S3 Canned ACL Examples

Canned ACL Name	Applies To	Permissions Added to ACL
private	Bucket and object	Owner gets FULL_CONTROL. No one else has access rights (default).
public-read	Bucket and object	Owner gets FULL_CONTROL. The AllUsers group (see Who is a grantee?) gets READ access.
public-read-write	Bucket and object	Owner gets FULL_CONTROL. The AllUsers group gets READ and WRITE access. Not recommended.
aws-exec-read	Bucket and object	Owner gets FULL_CONTROL. Amazon EC2 gets READ access to GET an Amazon Machine Image (AMI) bundle from Amazon S3.
authenticated-read	Bucket and object	Owner gets FULL_CONTROL. The AuthenticatedUsers group gets READ access.
bucket-owner-read	Object	Object owner gets FULL_CONTROL. Bucket owner gets READ access. If you specify this canned ACL when creating a bucket, Amazon S3 ignores it.
bucket-owner-full-control	Object	Both the object owner and the bucket owner get FULL_CONTROL over the object. If you specify this canned ACL when creating a bucket, Amazon S3 ignores it.
log-delivery-write	Object	The LogDelivery group gets WRITE and READ_ACP permissions on the bucket.

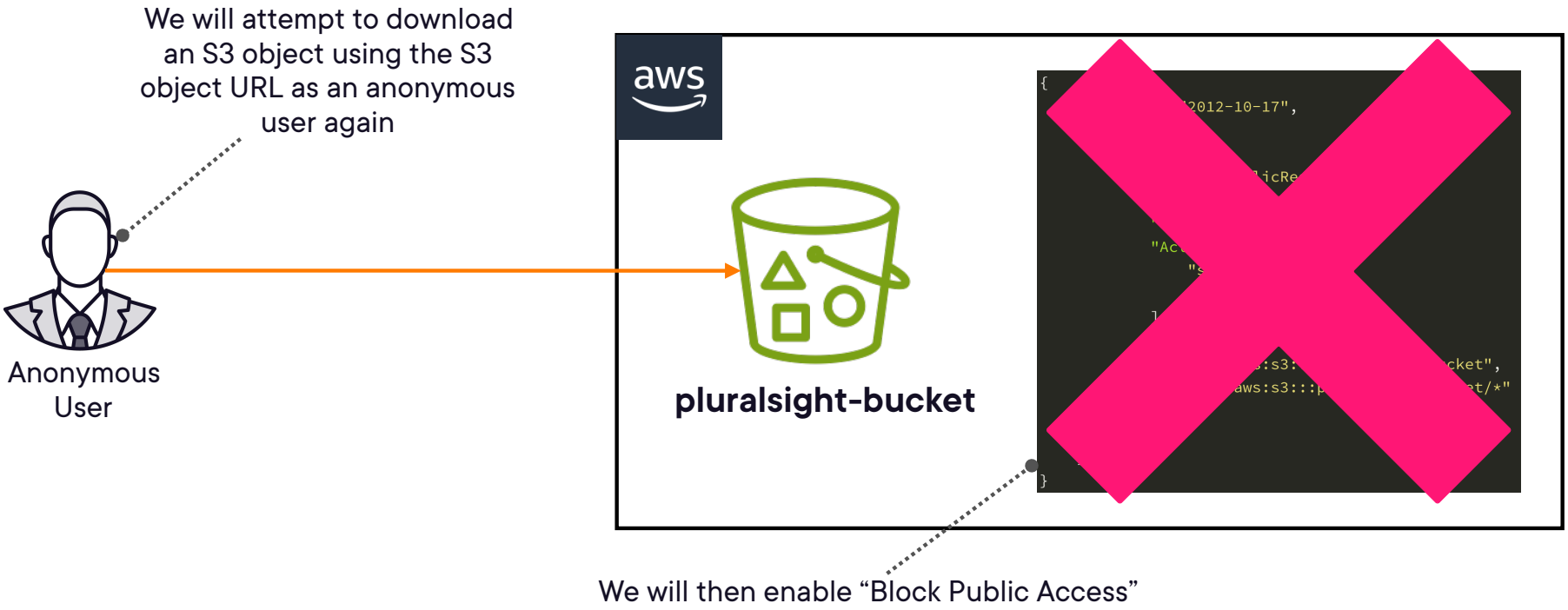
Pro Tip: Generally, bucket policies are favored over using ACLs due to their ease of use to implement controls.

AWS even recommends you leave ACLs turned off unless absolutely necessary!

Demo: Blocking Public Access to your Amazon S3 Bucket



Demo: Blocking Public Access to your Amazon S3 Bucket



Encrypting Data at Rest in Amazon S3: Overview

Encryption at Rest in Amazon S3 Overview

All new buckets now have encryption at rest configured by default!

You can optionally use one of the following methods:

- SSE-S3: S3-managed keys, using AES 256-bit encryption. Default method!
- SSE-KMS: AWS Key Management Service-managed keys
- SSE-C: Customer-provided keys
- Client-Side Encryption

We will explore these encryption types coming up soon!

Enforcing Server-side Encryption

S3 makes it possible to actually enforce that server-side encryption be used in the upload calls for objects before they are stored

This is done via the bucket policy and a condition statement

The request headers needs to include the **x-amz-server-side-encryption** header

Header value options:

- AES256 *(S3-managed keys)*
- aws:kms *(KMS-managed keys)*

Image Source: https://unsplash.com/

Enforcing Server-side Encryption for S3 Uploads

HTTP request example

```
PUT /myFile HTTP/1.1
Host: myBucket.s3.amazonaws.com
Date: Wed, 25 Nov 2020 09:50:00 GMT
Authorization: authorization string
Content-Type: text/plain
Content-Length: 27364
x-amz-meta-author: Ryan
Expect: 100-continue
x-amz-server-side-encryption: AES256
[27364 bytes of object data]
```

Enforcing Server-side Encryption for S3 Uploads

HTTP request example

```
PUT /myFile HTTP/1.1
Host: myBucket.s3.amazonaws.com
Date: Wed, 25 Nov 2020 09:50:00 GMT
Authorization: authorization string
Content-Type: text/plain
Content-Length: 27364
x-amz-meta-author: Ryan
Expect: 100-continue
x-amz-server-side-encryption: AES256
[27364 bytes of object data]
```

Encrypting Data at Rest in Amazon S3: SSE-S3

What is SSE-S3?

Server-side encryption at rest using AWS-managed keys

Leverages AES-256 encryption type

Currently the default setting for all new S3 buckets that get created

Header will be set to “x-amz-server-side-encryption: AES256”

There is no additional cost or performance impact with this type of encryption at rest!

Encrypting Data at Rest in Amazon S3: SSE-KMS

What is SSE-KMS?

Server-side encryption at rest using AWS KMS-managed keys

KMS is a secure service for cryptographic key management

This method offers greater control and customization for the encryption

Header will be set to “x-amz-server-side-encryption: aws:kms”

SEE-KMS Concepts

Useful for encryption-related compliance requirements

The AWS KMS keys must be in the same Region as the bucket

Pro Tip: There are additional charges for using AWS KMS keys!

If you use SSE-KMS, you must have permissions to decrypt using the KMS key before you can read an object!

Encrypting Data at Rest in Amazon S3: SSE-C

What is SSE-C?

Server-side encryption where you fully manage and use your own keys

This means that Amazon S3 has no access to the key used

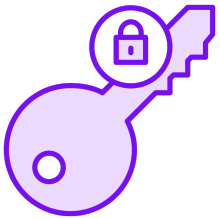
You are required to leverage HTTPS for interactions

You must include the key in every single HTTP request made to S3

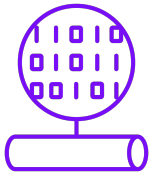
There are three very important headers to be aware of...

SSE-C HTTP Headers

x-amz-server-side-encryption-customer-algorithm: Specify the encryption algorithm, which must be AES256



x-amz-server-side-encryption-customer-key: The 256-bit, base64-encoded encryption key used for S3 encryption and decryption



x-amz-server-side-encryption-customer-key-MD5: The base64-encoded 128-bit MD5 digest of the encryption key used for integrity checks

There are no additional charges for using SSE-C!

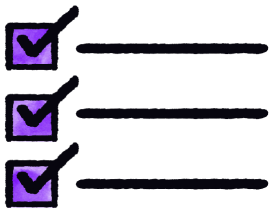
Understand the tradeoffs for using this encryption method.

Encrypting Data at Rest in Amazon S3: Client-side

This method means you encrypt your data locally before transporting it to Amazon S3.

Client-side Encryption Overview

You fully manage the keys and the encryption cycles/rotations



You have to encrypt before sending to S3, and then decrypt after getting from S3

Leverage the Amazon S3 Encryption Client to help simplify this process

When using this method, S3 has no idea that your object is encrypted.

The service only sees it as a normal object!

Optimizing S3 Encryption Using Bucket Keys

Remember SSE-KMS incurs additional charges for calls made to the KMS service (*encrypt*, *decrypt*).

This is where a “bucket key” is useful!

Amazon S3 Bucket Keys

A bucket-level key for SSE-KMS encryption keys

Feature is capable of reducing costs by up to 99%

AWS generates a short-lived, bucket-level key from AWS KMS

The short-lived key is stored in S3 for a finite period of time

Data keys are generated via this key and used for encryption

Requests are now not needed for every cryptographic call

TL;DR: Bucket Keys allow you to save on SSE-KMS costs by decreasing the number of requests made to the AWS KMS service.

Amazon S3 Encryption in Transit

Amazon S3 Encryption in Transit

You can and should leverage TLS during HTTP requests made to Amazon S3

S3 offers HTTP and HTTPS endpoints, but usually HTTPS is the default one used

Use the aws:SecureTransport condition to require the use of TLS

aws:SecureTransport Conditional Requirement

```
{
   "Version": "2012-10-17",
   "Statement": [
      {
         "Sid": "RequireSecureTransport",
         "Effect": "Deny",
         "Principal": "*",
         "Action": "s3:PutObject",
         "Resource": “arn:aws:s3:::pluralsight-bucket/*”,
         "Condition": {
            "Bool": {
               "aws:SecureTransport": "false"
            }
         }
      }
   ]
}
```

aws:SecureTransport Conditional Requirement

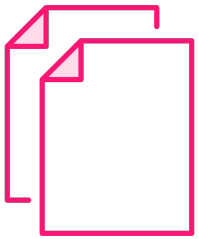
```
{
   "Version": "2012-10-17",
   "Statement": [
      {
         "Sid": "RequireSecureTransport",
         "Effect": "Deny",
         "Principal": "*",
         "Action": "s3:PutObject",
         "Resource": “arn:aws:s3:::pluralsight-bucket/*”,
         "Condition": {
            "Bool": {
               "aws:SecureTransport": "false"
            }
         }
      }
   ]
}
```


Condition block saying that you **MUST use TLS** when putting an object into the bucket. If false, then the action is denied.

Preventing Accidental Deletions with MFA Delete

MFA Delete for S3

Enforces that MFA is required to perform different important actions pertaining to Amazon S3



You must enable versioning for the bucket to successfully enable the MFA Delete feature

The bucket owner is the only one who can actually enable and disable this feature for an S3 bucket

Actions Where MFA Is Required

Required

Permanent deletion of objects, and suspending versioning for the bucket

Not Required

Enabling versioning for the bucket, and listing deleted versions of objects

MFA options can be virtual *(e.g. Google Authenticator)* **or hardware** *(e.g. YubiKey)*.

Logging Interactions Using S3 Access Logs

Amazon S3 Access Logs provide detailed records about requests that are made to a bucket.

S3 Server Access Logging Concepts

Very useful for security and auditing of your Amazon S3 bucket access

Helpful for analyzing your bucket and object usage, as well as your bills

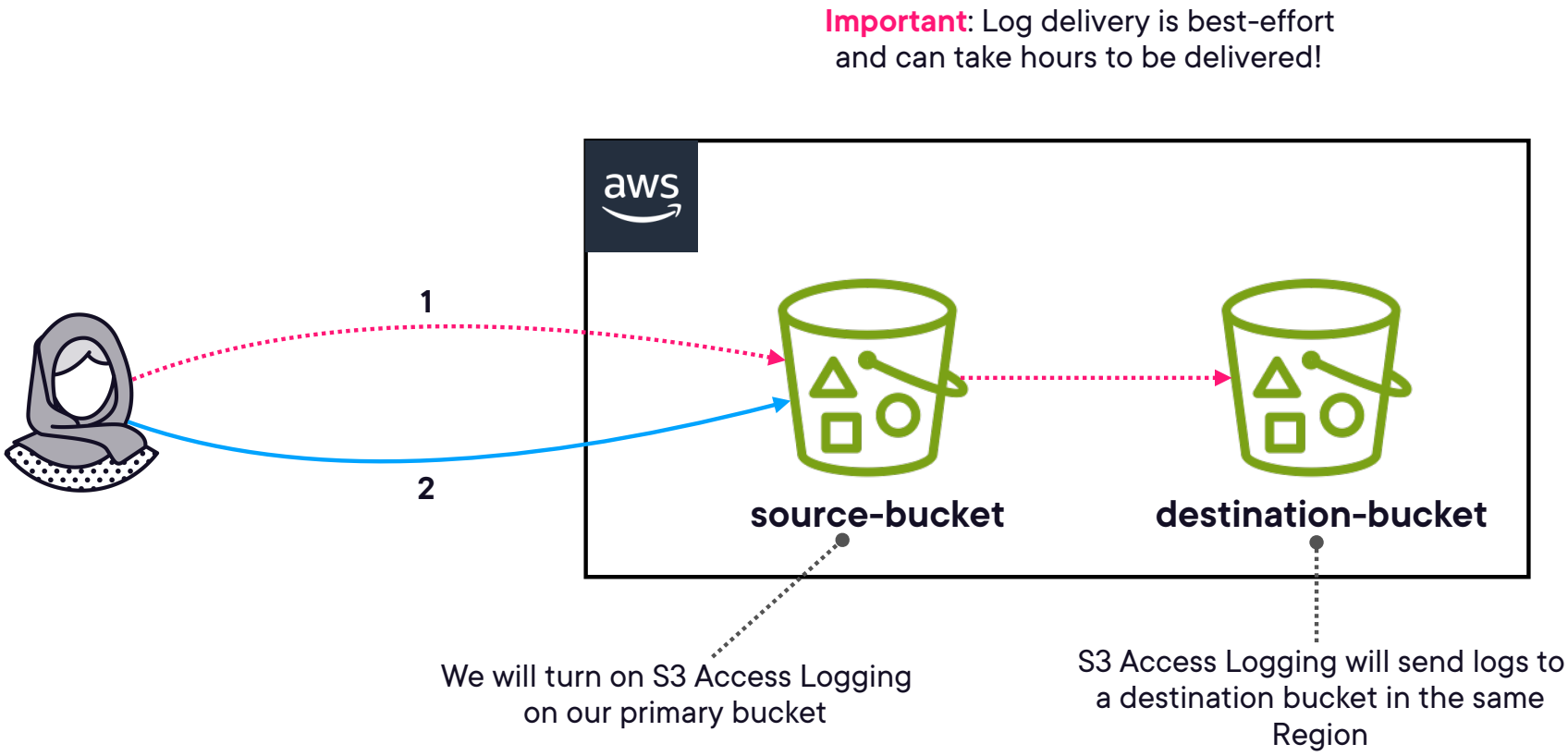
When enabling this feature, you must specify a separate target bucket

Target buckets must reside in the same Region as the source buckets

Avoid infinite loops by NOT using the source bucket as the destination

Exam Pro Tip: Turn this feature on if you need to see any information about requests made to your buckets!

Demo: Turning on Access Logging in S3



Granting Access to Objects with S3 Presigned URLs

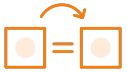
Presigned URLs can be used to grant time-limited access to objects in S3 without updating having to your bucket policy!

Presigned URLs Concepts

You generate a URL via console, AWS CLI, and even supported AWS SDKs

The URL can be entered into a browser/app to download or upload objects

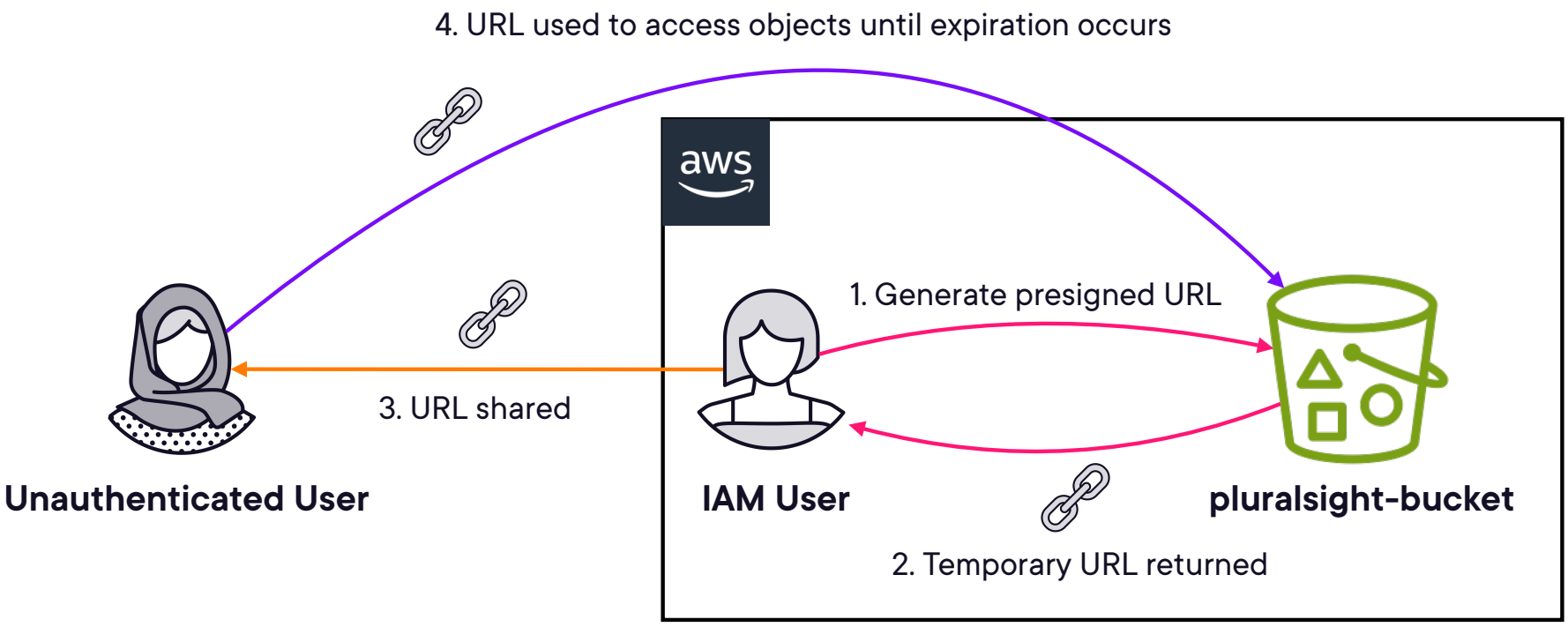
Provides the same credentials as the AWS user who generated the URL



Can be used multiple times, up to the specified expiry date and time

Specify the bucket, object key, HTTP method, and expiration time interval

Presigned URL Architecture Diagram



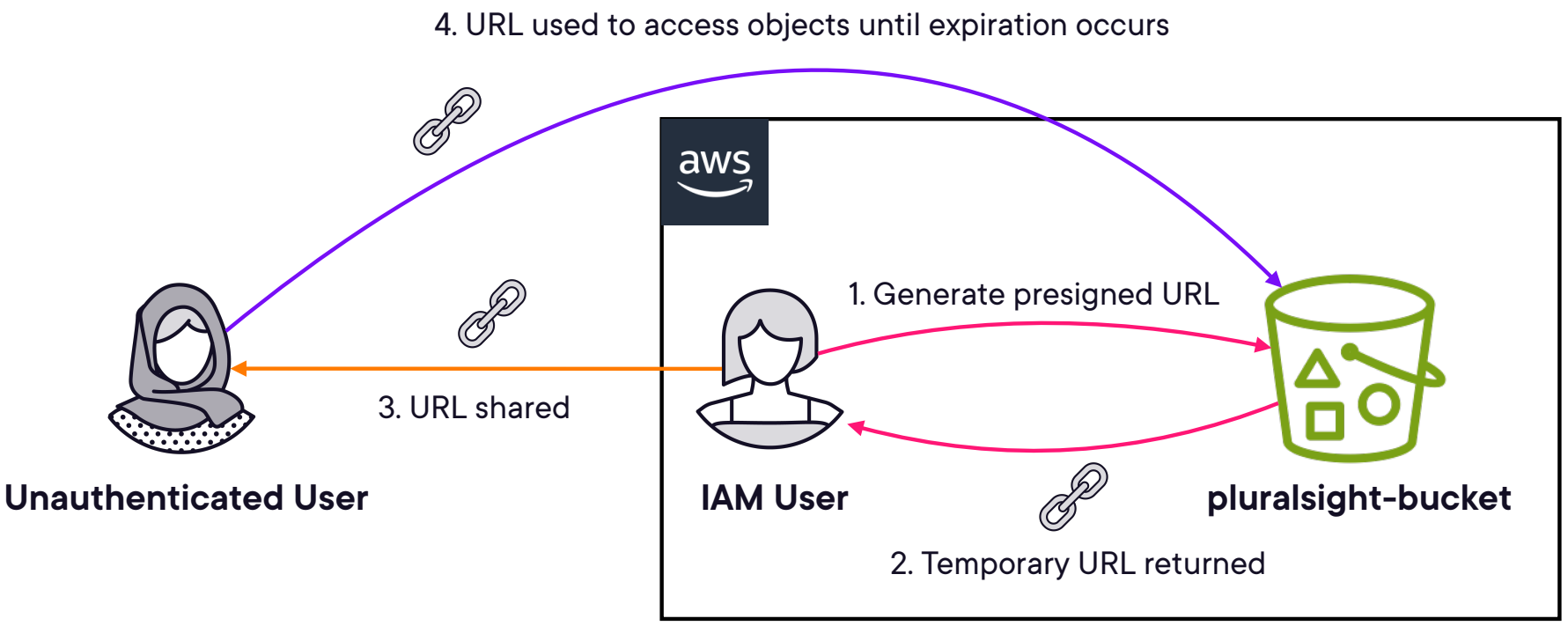
Presigned URL Exam Scenarios

Scenario #1: Expired URL resulting in failed downloads when attempting to get objects.

Scenario #2: IAM user who generated the URL does not actually have the correct intended permissions, resulting in an *access denied*. Yes, this is possible!

Scenario #3: Generating URLs to allows users to access premium content behind a paywall.

Demo: Sharing an S3 Presigned URL



Fine-grained Access Control with S3 Access Points

Amazon S3 Access Points

Amazon S3 access points simplify data access for any AWS service or customer application that stores data in S3.

Citation: https://docs.aws.amazon.com/AmazonS3/latest/userguide/access-points.html

Amazon S3 Access Points Concepts

Appear as named network endpoints (DNS name) that are attached to your buckets

Used for performing S3 object operations like GetObject and PutObject

Each one has its own permissions policy and network controls assigned to it

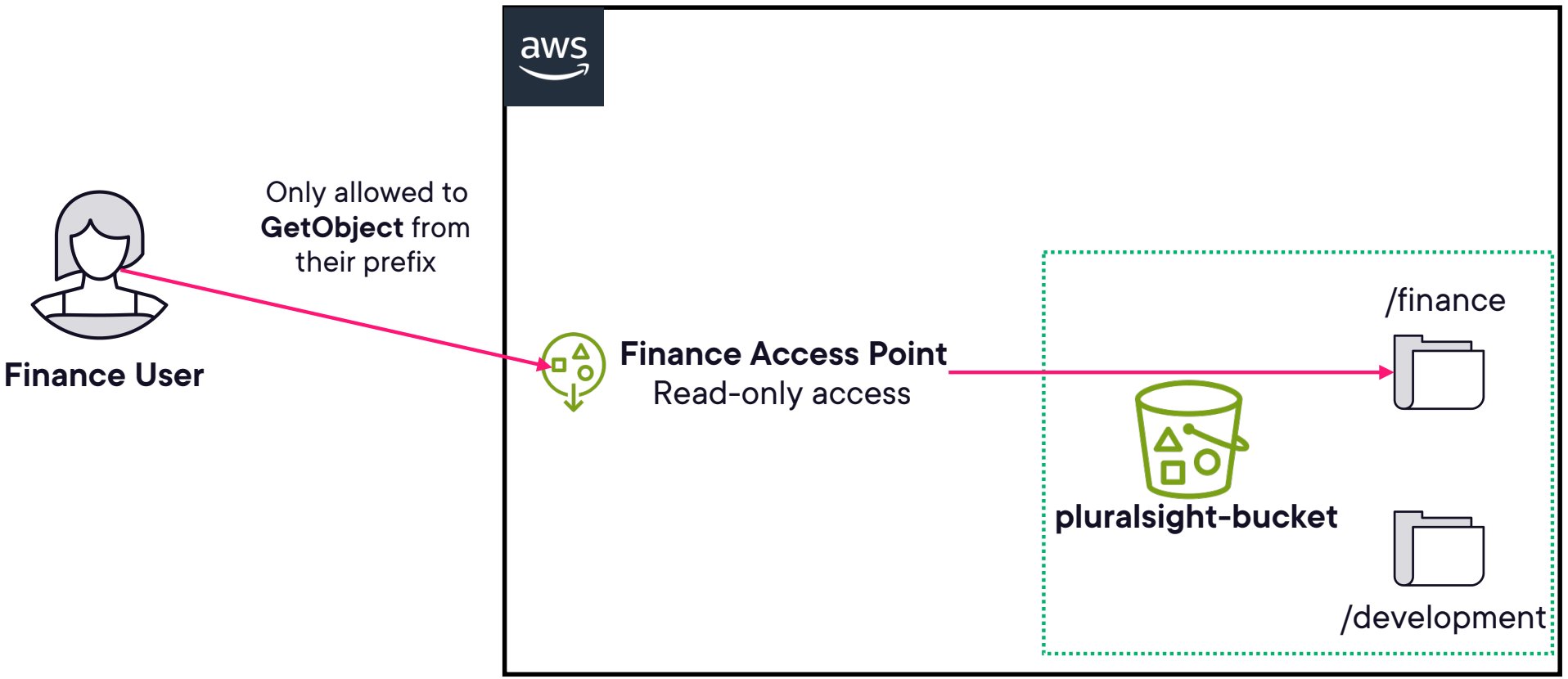
Customized access point policy works in conjunction with the bucket policy

Configuring Access Scenarios

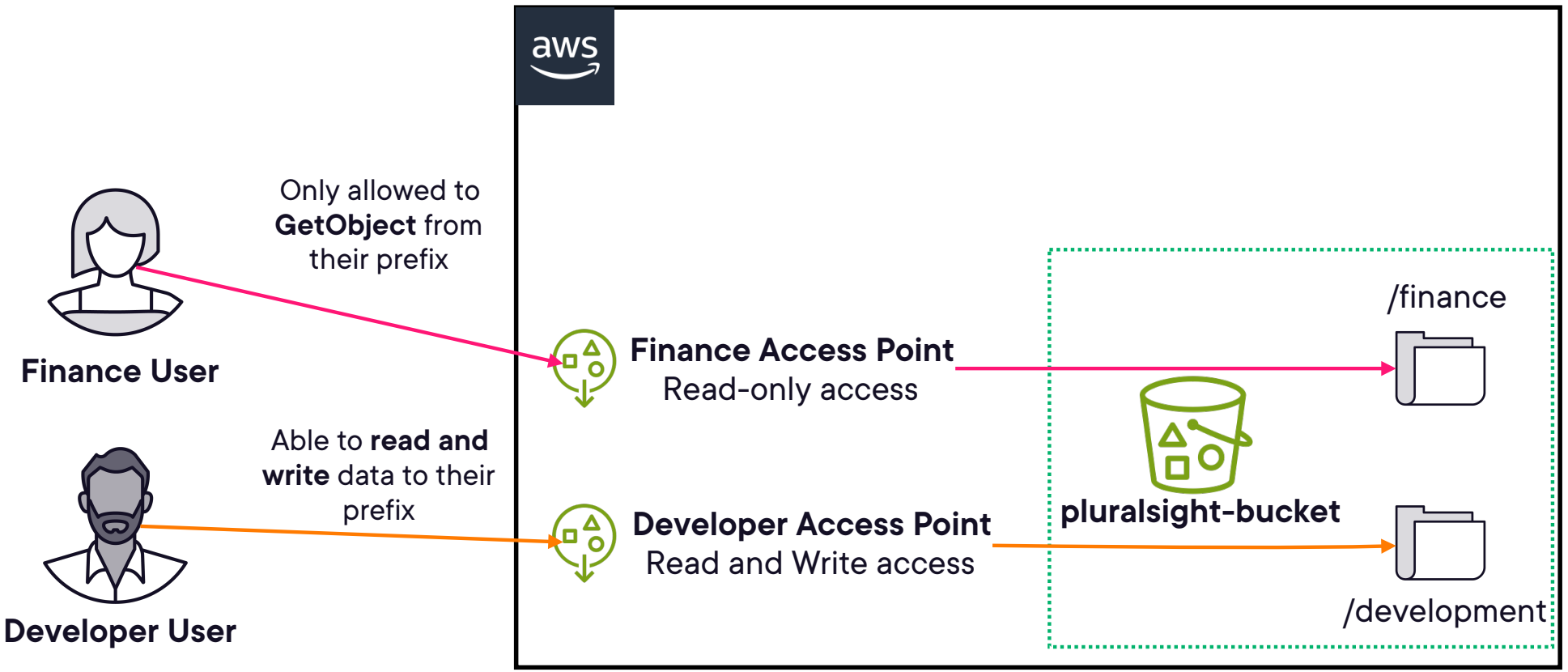
Configure the access point to only allow traffic from a VPC to access the bucket objects (requires a VPC endpoint)

Configure a custom block public access setting for individual endpoints instead of the entire bucket

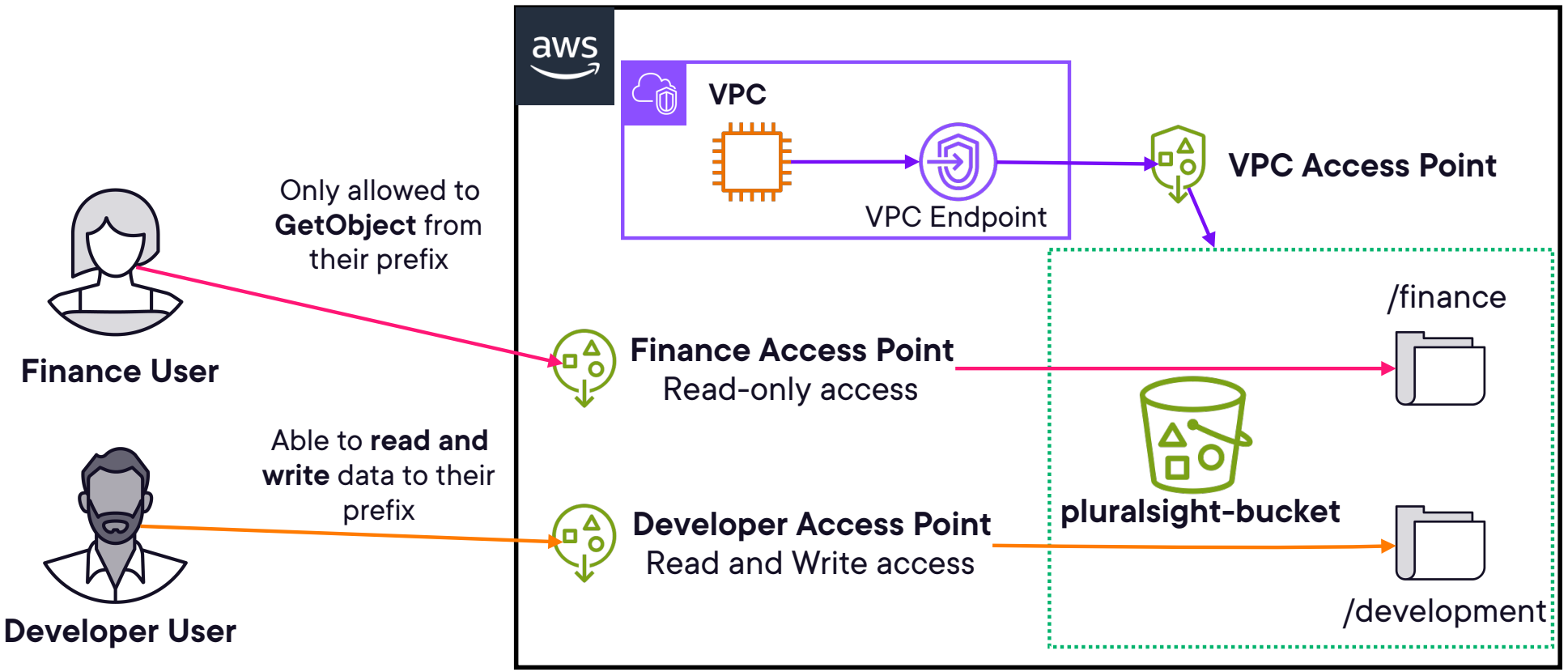
S3 Access Point Architecture Diagram



S3 Access Point Architecture Diagram

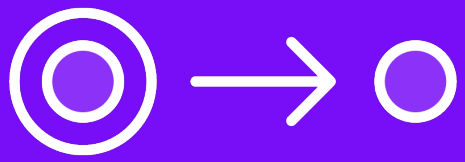


S3 Access Point Architecture Diagram



Remember that S3 Access Points allow you to simplify and customize access to your S3 objects and data.

Transforming S3 Objects with Object Lambdas



Amazon S3 Object Lambda

Easily add your own code to S3 GET, HEAD, and LIST requests to modify and process data as it is returned to an application by using Lambda functions

Using S3 Object Lambdas

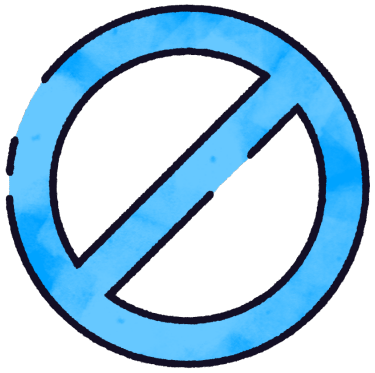
Lambda functions are a serverless offering allowing you to run simple code functions to execute a workload

You must create an S3 Object Lambda Access Point

You must create the AWS Lambda function

Then, you must create an S3 Access Point as well

S3 Object Lambda Use Cases

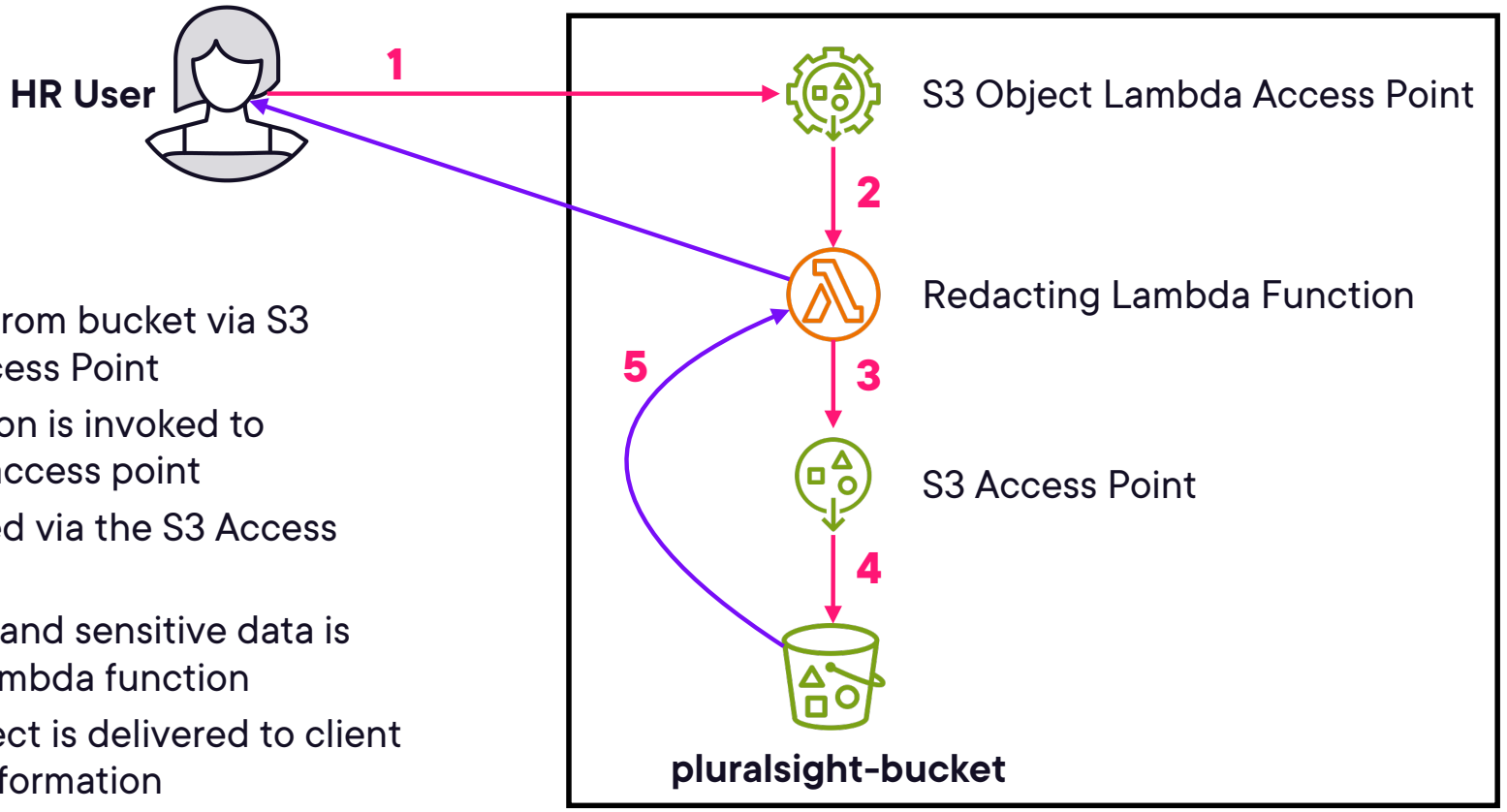


Useful for removing sensitive information from data before it is returned to a client

Create a customized view of objects in S3 that are returned from a LIST request

Convert data between formats like CSV and TSV while being returned

S3 Object Lambda Architecture Diagram



1. Requests object from bucket via S3 Object Lambda Access Point

2. Redacting function is invoked to request object via access point

3. Object is retrieved via the S3 Access Point in use

4. Object retrieved and sensitive data is redacted via the Lambda function

5. Transformed object is delivered to client without sensitive information

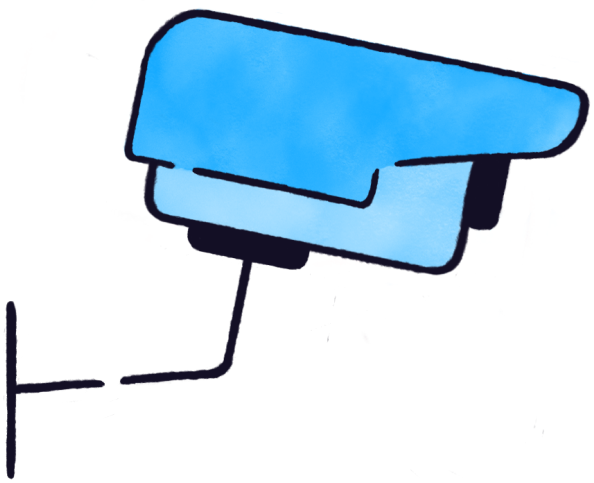
If you need to transform S3 objects before returning them to a client, think S3 Object Lambdas!

Using S3 Locks to Meet Compliance Requirements

S3 Object Lock

You can use S3 Object Lock to store objects using a write once, read many (WORM) model. It can help prevent objects from being deleted or modified for a fixed amount of time or indefinitely.

S3 Object Lock Governance Mode



Protect objects against being deleted by most users

Regular users cannot overwrite or delete an object version, or alter the lock settings

Actions require specific admin privileges

You can grant some users permission to alter the retention settings or delete the object if necessary

This is a good mode for trial runs of object locks

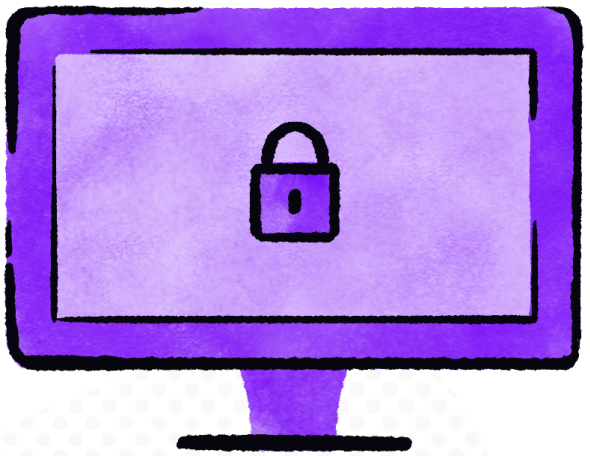
S3 Object Lock Compliance Mode

A protected object version can't be overwritten or deleted by any user including the root user in your AWS account

When an object is locked in compliance mode, its retention mode can't be changed and its retention period can't be shortened

Ensures an object version can't be overwritten or deleted for the duration of the retention period

The unbeatable, final boss for locking objects!



Versioning must be enabled for either of the S3 Object Lock modes to be used!

S3 Object Lock Retention Periods

A retention period protects an object version for a fixed amount of time

A timestamp is stored in the object version's metadata to indicate when the retention period expires

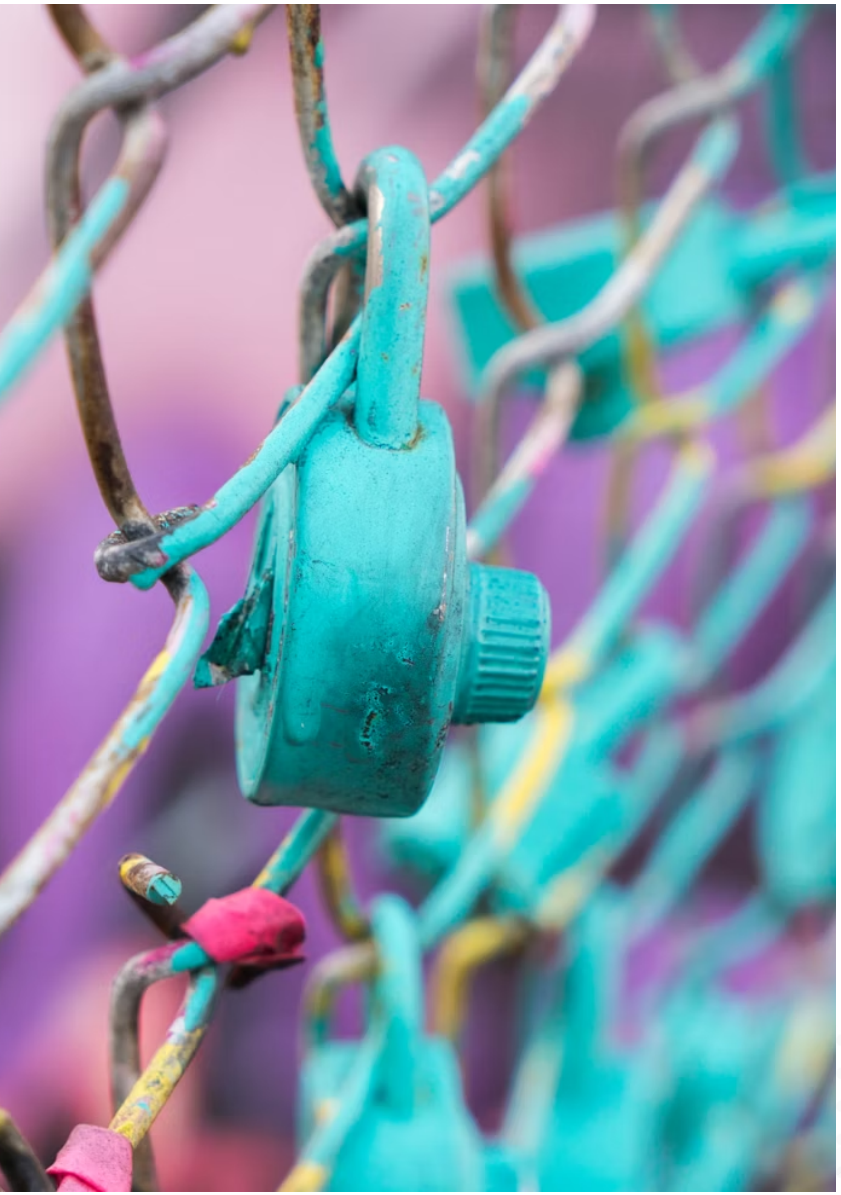
After the retention period expires, you can overwrite and delete the object unless there is a legal hold on the object version

S3 Object Lock Legal Holds

This also prevents an object version from being overwritten or deleted

These don't have an associated retention period, and remain until removed

Placed and removed by any user who has the s3:PutObjectLegalHold permission



S3 Glacier Vault Lock

S3 Glacier Vault Lock allows you to easily deploy and enforce compliance controls for individual S3 Glacier vaults with a vault lock policy

Specify controls, such as WORM, in a vault lock policy and lock the policy from future edits

Once locked, the policy can no longer be changed

Image Source: https://unsplash.com/

Scenarios revolving around S3 data retention and compliance against tampering likely involve one of these solutions!

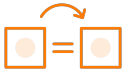
Module Summary and Exam Tips

Amazon S3 Exam Tips

S3 buckets are private by default (*including all objects within it*)

Object ACLs: You can make individual objects public using object ACLs.

Bucket ACLs: XML policy that is used to manage access to the bucket



Bucket policies: You can make entire buckets public using bucket policies.

Bucket policies are favored over using any of the ACL types if possible

Be able to identify what a bucket policy is doing.

Remember the difference between these two resource ARNs and how they operate!

Example Bucket Policy

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

Bucket Policy Conditions Exam Tips

x-amz-server-side-encryption

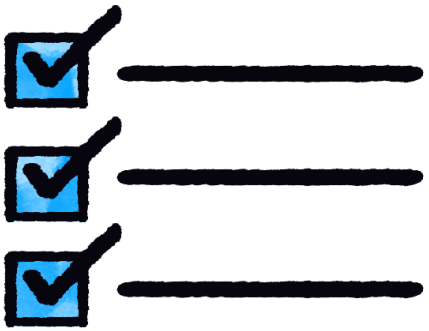
You can require this header to ensure that the object is encrypted by AWS before it is stored

aws:SecureTransport

This conditional key check ensures that requests are made over HTTPS, which implies the use of TLS

Remember that S3 has a "Block Public Access" easy setting that you can turn on!

Encryption at Rest in Amazon S3 Exam Tips



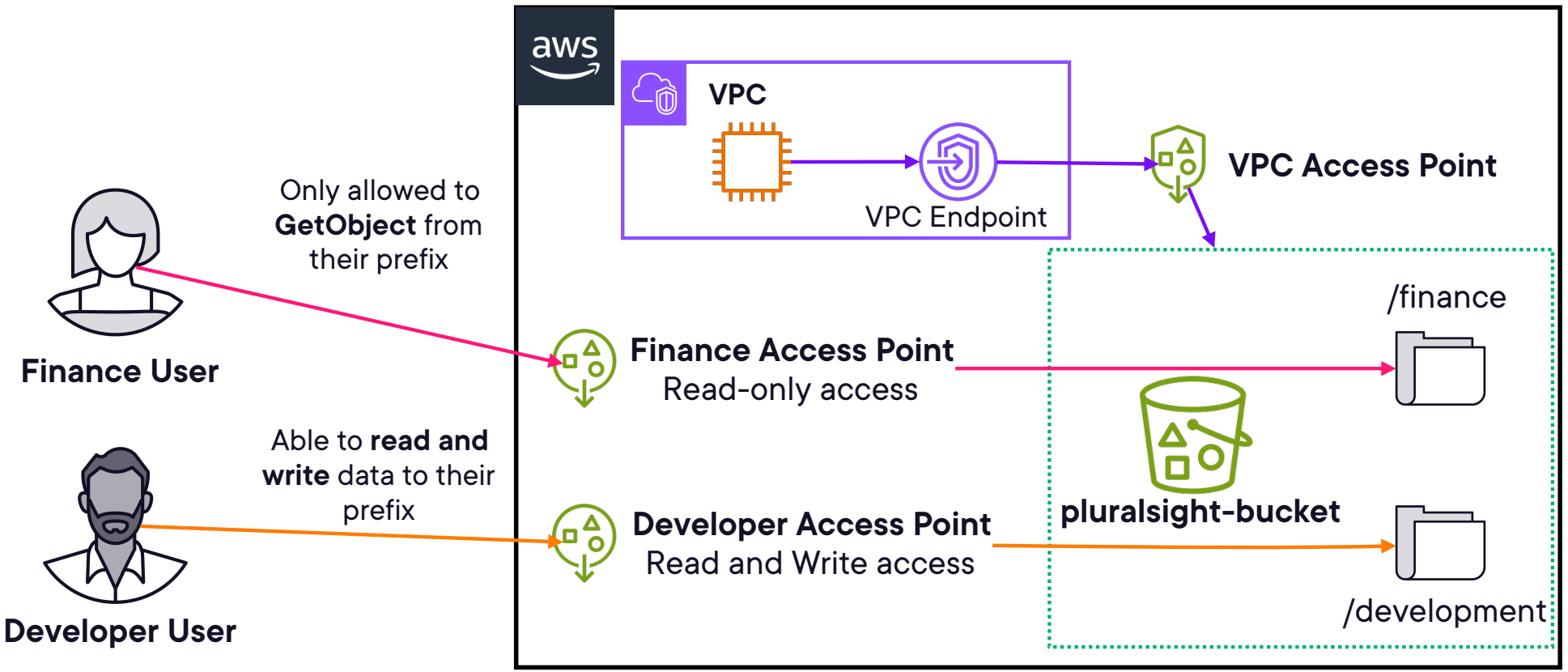
All new buckets now have encryption at rest configured by default!

Remember the following methods:

- SSE-S3: S3-managed keys, using AES 256-bit encryption. This is the default method!
- SSE-KMS: AWS Key Management Service-managed keys. Watch for costs and quotas!
- SSE-C: Customer-provided keys. You manage your own cryptographic keys.
- Client-Side Encryption: AWS has no idea that the objects are even encrypted!

Exam Pro Tip: Use Bucket Keys to save on KMS costs and avoid potential quota throttles!

S3 Access Point Architecture Diagram Review



Remember that S3 Access Points are meant to allow you to simplify and customize access to your S3 objects and data.

Need to view details about requests made to your Amazon S3 buckets?

Use S3 Access Logs!

Reviewing Presigned URL Exam Scenarios

Scenario #1: Expired URL resulting in failed downloads when attempting to get objects.

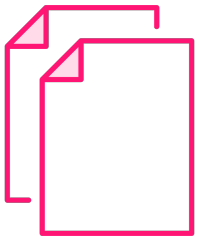
Scenario #2: IAM user who generated the URL does not actually have the correct intended permissions, resulting in an *access denied*. Yes, this is possible!

Scenario #3: Generating URLs to allows users to access premium content behind a paywall.

If you need to transform S3 objects before returning them to a client, think S3 Object Lambdas!

MFA Delete for S3 Exam Tips

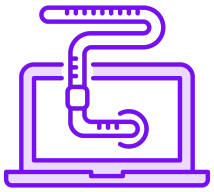
Enforces that MFA is required to perform different important actions pertaining to Amazon S3



You must enable versioning for the bucket to successfully enable the MFA Delete feature

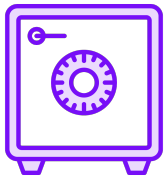
The bucket owner (root) is the only one who can actually enable and disable this feature for an S3 bucket

S3 Object Lock Exam Tips



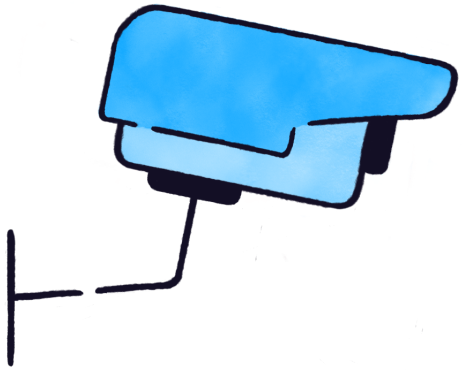
Use S3 Object Lock to store objects using a write once, read many (WORM) model

Object Lock can be on individual objects or applied across the bucket as a whole



Object Lock comes in two modes: Governance Mode and Compliance Mode

Object Lock Mode Exam Tips



Governance Mode

Users can't overwrite or delete an object version or alter its lock settings unless they have special permissions

Compliance Mode

Protected object versions can't be overwritten or deleted by any user, including the root user in your AWS account

S3 data retention and compliance scenario?

Leverage one of those solutions!

Image Source: http://unsplash.com

S3 Glacier Vault Lock Exam Tips

Allows you to easily deploy and enforce compliance controls for individual S3 Glacier vaults with a vault lock policy.

You can specify controls, such as WORM, in a vault lock policy and lock the policy from future edits.

Important: Once locked, the policy can no longer be changed!