<https://explore.skillbuilder.aws/learn/course/161/play/36138/getting-started-with-amazon-simple-storage-service-amazon-s3;lp=82>

**Securing Data Access**

*Lesson 8 of 11*

As with all of the AWS services, security exists in all layers of Amazon S3. By default, all Amazon S3 resources are private and accessible by the resource owner or account administrator. By implementing a strict security stance, Amazon S3 allows you to work backwards, configure, and finely-tune your access policies. This helps to align your organizational, governance, security, and compliance requirements. One of AWS Security best practices is to design your environment based on the principle of least privilege.

**Principle of least privilege**

Least privilege is a security design strategy where granted permissions allow only the minimum necessary rights required to accomplish the task. With least privilege, you start with a strict set of minimum permissions and grant additional permissions only when necessary. Starting with tight restrictions and adding new ones when required is more secure. Starting with open permissions too lenient and then trying to tighten them later is less secure.

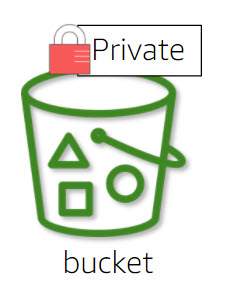
When working with Amazon S3, identify what each user, role, and application needs to accomplish within your buckets and then create policies that allow them to perform only those specific tasks. When granting permissions, you decide who gets permissions and into which Amazon S3 resources. You enable specific actions that you want to allow on those resources. Therefore, you should grant only the permissions that are required to perform a task. Implementing least privilege access is fundamental in reducing security vulnerabilities.

**Security mechanisms**

A newly created bucket can only be accessed by the user who created it or by the account owner. You must grant access to other users by using one or a combination of the following access management features.

**AWS Identity and Access Management**

IAM is used to create users and manage their respective access to resources, including buckets and objects.



**Bucket policies**

Bucket policies are used to configure permissions for all or a subset of objects using tags and prefixes.

**Pre-Signed URLs**

Pre-Signed URLs are used to grant time-limited access to others with temporary URLs.

**Access control lists**

Access Control List (ACLs) to make individual objects accessible to authorized users.

* Note: Amazon S3 ACLs are a legacy access control mechanism that predates IAM. AWS recommends using Amazon S3 bucket policies or IAM policies for access control.

In the next few sections, we discuss the recommended ways to secure your data and the differences between them.  Let's start by discussing Block Public Access settings.

**Block Public Access**

AWS introduced the S3 Block Public Access feature to help you avoid inadvertent data exposure. With Block Public Access, you can manage public access of your Amazon S3 resources at both the AWS account level and the bucket level, which helps ensure that your data is not publicly available. Any new bucket created has block all public access enabled by default.

What does this mean for you? If you want to grant public access on any resources managed by Block Public Access, you will have to adjust your Block Public Access configuration.



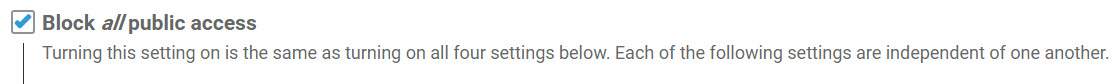
**Amazon S3 Block Public Access settings**

Review each of the available Block Public Access (BPA) options, which are either enabled at the account level or the bucket level. If enabled at the account level, the BPA settings override any settings on the individual buckets.

**Block all public access**

Sometimes, you want to make sure that a bucket will never allow public access. By enabling this one click option, you can prevent public access to your bucket. This overrides any configured ACLs and bucket polices that would normally grant public access. Choosing to enable this option equates to enabling all of the other options listed here.

Any new bucket you create will have this option enabled by default. You need to disable this option if you want to allow public access to your bucket or objects.

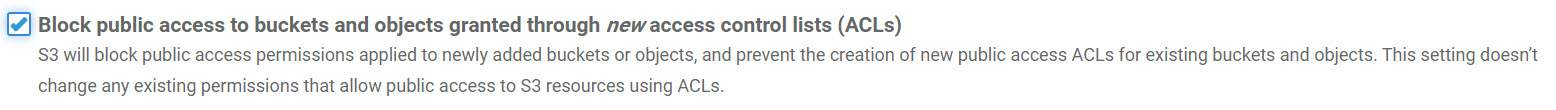


**Block public access granted through new ACLs**

This option prevents you from creating any **new** ACL, either for a bucket or object, which grants public access permissions. This option only affects the creation of new public ACLs; it does not alter any existing ACLs or policies. Any existing ACLs or policies granting public access will not affect permissions and public access to those resources will remain intact.

After you enable this option, you should then review your ACLs to evaluate any existing public access permissions and assess whether or not those permissions should stay the same.

Remember, with this option enabled, if you have any bucket policies or existing ACLs granting public access to buckets and objects, those resources will remain publicly accessible. If you wish to block all public access to buckets and objects, choose the block all public access option.



**Block public access granted through any ACLs**

This option only affects how you evaluate ACL public permissions. When you enable this option, it ignores any existing ACLs that grant public permission on buckets and objects. This does not alter the existing ACLs themselves, but any resources configured with existing public ACLs will no longer be publicly accessible.

It can be confusing because it does not prevent you from creating new ACLs that would normally grant public access. You can still create them, but those ACLs will not become effective, resulting in the bucket or object not being publicly accessible.

You should take the time to review your ACLs once enabled and remove any public ACLs to prevent any possible future mistakes. Any existing public ACLs, will no longer be ignored, if the block public access granted through any ACLs option is later disabled.

Remember, with this option enabled, if you have any bucket policies granting public access to buckets and objects, those buckets or objects will remain publicly accessible. If you wish to block all public access to buckets and objects, choose the block all public access option.

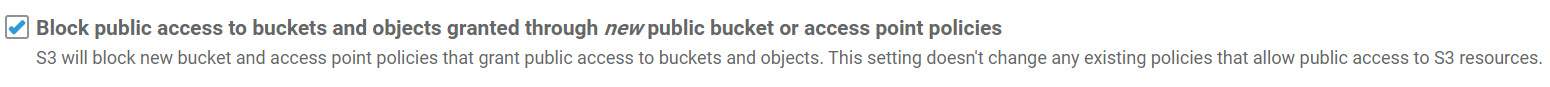


**Block public access granted through new public bucket policies**

This option only prevents the creation of new bucket policies that grant public access. Any existing bucket policies are not affected. If you currently have any bucket policies configured that grant public access, those buckets or objects will remain publicly accessible.

To use this setting effectively, you should apply it at the AWS account level. A bucket policy can allow users to alter a bucket's Block Public Access settings. Therefore, users who have permission to change a bucket policy could insert a policy that allows them to disable the Block Public Access settings for the bucket. If enabling this setting for the entire account, rather than for a specific bucket, Amazon S3 blocks public policies even if a user alters the bucket policy to disable this setting.

Remember, with this option enabled, if you have any existing bucket policies or ACLs granting public access to buckets and objects, those buckets or objects will remain publicly accessible. If you wish to block all public access to buckets and objects, choose the block all public access option.

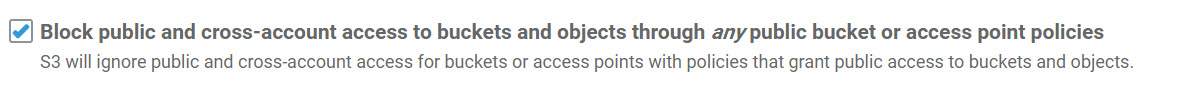


**Block public and cross-account access granted through any public bucket policies**

This option only affects how you evaluate bucket policy permissions. When you enable this option, it ignores any buckets or objects that have public permissions granted through bucket policies. This option, when enabled, restricts access to a bucket with a public policy to only AWS services and authorized users within the bucket owner's account. This setting blocks all cross-account access to the bucket with a public policy (except by AWS services), while still allowing users within the account to manage the bucket.

This does not alter existing bucket policies, but ignores any existing bucket policies that grant public access, blocking public access and any cross-account access configurations.

Remember, with this option enabled, if you have any ACLs granting public access to buckets and objects will remain publicly accessible. If you wish to block all public access to buckets and objects, choose the block all public access option



Unless you need to make your bucket or objects publicly available, we strongly recommend enabling the block all public access option.

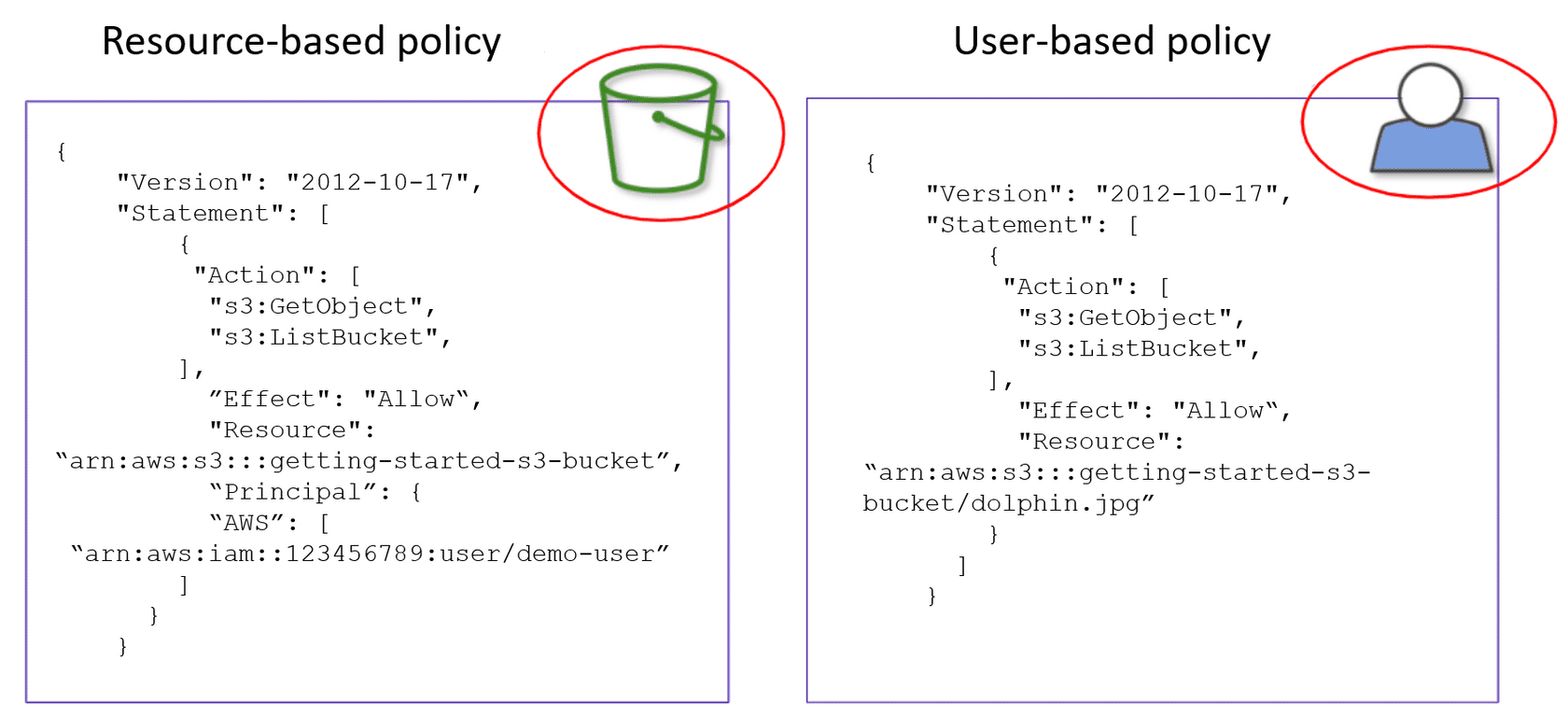
To find out more about what block public access considers public, select this link for [block public access documentation](https://docs.aws.amazon.com/AmazonS3/latest/dev/access-control-block-public-access.html#access-control-block-public-access-policy-status).

**Access policies**

Access policy describes who has access to what resources. They attach to your resources, such as buckets and objects, and are also called resource policies. For example, bucket policies and access control lists are resource based policies because you attach them directly to buckets and objects.

User policies or IAM policies are access policies attached to users in your account. You may choose to use one type of policy or a combination of both, to manage permissions with your Amazon S3 resources.

Examine the example of access policies in the image below. Both the bucket and user policies are written in JSON format. Just by looking at the code, it may not be immediately apparent which policy is for a user and which is for a bucket. However, by looking at what the policy is attached to, you can quickly determine which type of policy it is. If you are not familiar with JSON, AWS has a policy generator tool to assist you.



Example of a bucket policy (resource-based policy) and an IAM policy (user-based policy).



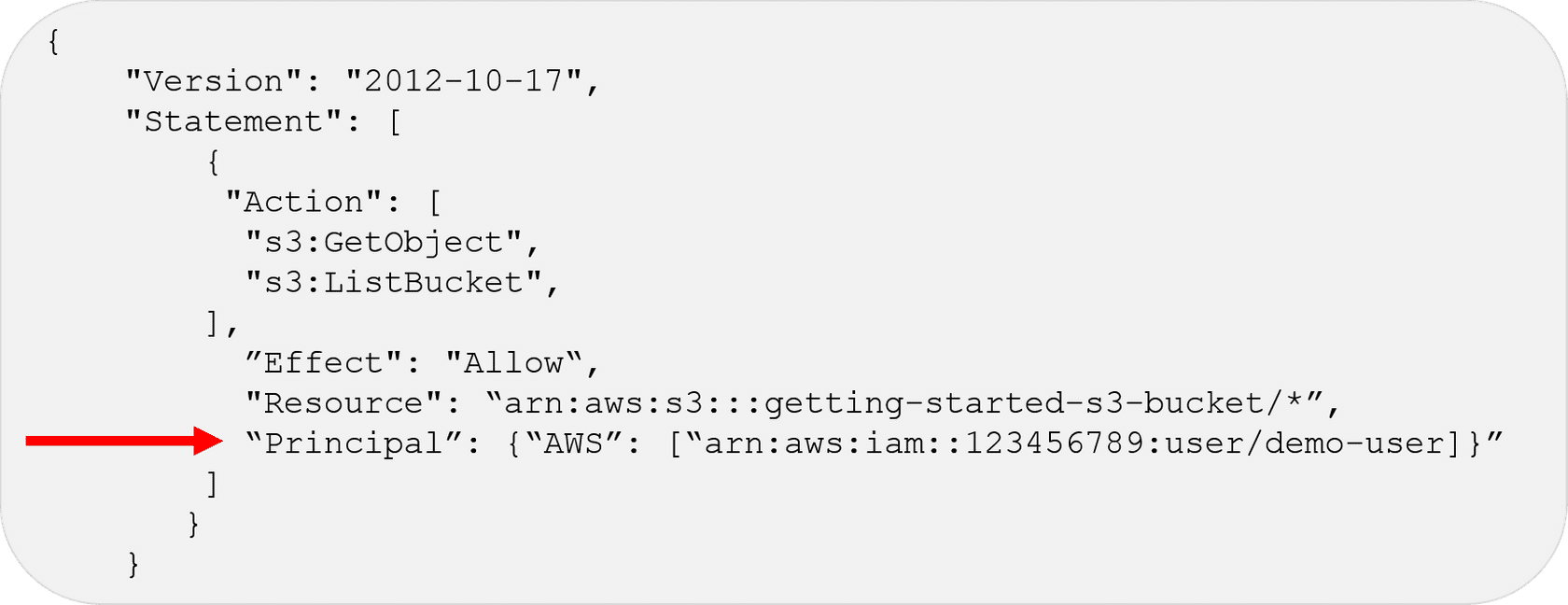
The AWS Policy Generator can assist you with JSON policy creation.

**Bucket policies**

In order to grant other AWS accounts or IAM users access to the bucket and the objects in it, you need to attach a bucket policy. Because you are granting access to a user or account, a bucket policy must define a PRINCIPAL (which is an account, user, role, or service) entity within the policy. You will notice that the "Principal" statement is listed in the policy. Consult the image below as an example.

When using bucket policies, Amazon S3 is managing the security. Bucket policies supplement, and in many cases, replace legacy ACL-based access policies. Amazon S3 supports a bucket policy size limit of up 20 kb.

Because bucket policies grant access to another AWS account or IAM user, you must specify the principal, or the user to whom you are granting access, as a "Principal" in the bucket policy.



The red arrow points to the Principal called demo-user who is being granted access to the getting-started-s3-bucket.

**When to use a bucket policy**

Use a bucket policy if:

* You need to grant cross-account permissions to other AWS accounts or users in another account, without using IAM roles.
* Your IAM policies reach the size limits for users, groups, roles.
* You prefer to keep access control policies in the Amazon S3 environment.
* Although both bucket and user policies support granting permission for all Amazon S3 operations, the user policies are for managing permissions for users in your account.

**IAM policies**

You can use IAM to manage access to your Amazon S3 resources. You can create IAM users, groups, and roles in your account and **attach access policies to them granting them access to AWS resources, including Amazon S3.**

There are maximum size limitations for IAM policies for users, groups, and roles. IAM policies have

* a 2 kb size limit for users,
* 5 kb for groups,
* and a 10 kb for roles.

Also note that there is no principal stanza listed in the User policy as the principal is whichever user the policy is applied to.

Afbeelding met tekst

Automatisch gegenereerde beschrijving

In this example, you grant your IAM user access to your buckets, ocean-life-bucket, and allow the user to Put\*, Get\*, Delete\* objects. The policy also grants the s3:ListAllMyBuckets, s3:GetBucketLocation,  s3:ListBucket, s3:PutObjectAcl, and s3:GetObjectAcl permissions which are additional permissions required by the AWS console. This example also shows each permission applied to different resources. In the top permission, the resource is all buckets (\*). In the middle the permission is applied to one specific bucket, called ocean-life-bucket. The bottom permission applies to only objects within the ocean-life-bucket.

**When to use IAM user policies**

Use IAM policies if:

* You need to control access to AWS services other than Amazon S3. IAM policies allow for easier centralized management all of your permissions.
* You have numerous Amazon S3 buckets each with different permissions requirements. IAM policies will be easier to manage than having to define a large number of Amazon S3 bucket policies. This way you can focus on having fewer, more detailed IAM policies.
* You prefer to keep access control policies in the IAM environment.

**Query string authentication**

You can use a query string to express a request entirely in a URL. To do this, you use query parameters to provide request information, including the authentication information. Because the request signature is part of the URL, this referred to as a **presigned URL**.

**Presigned URLs**

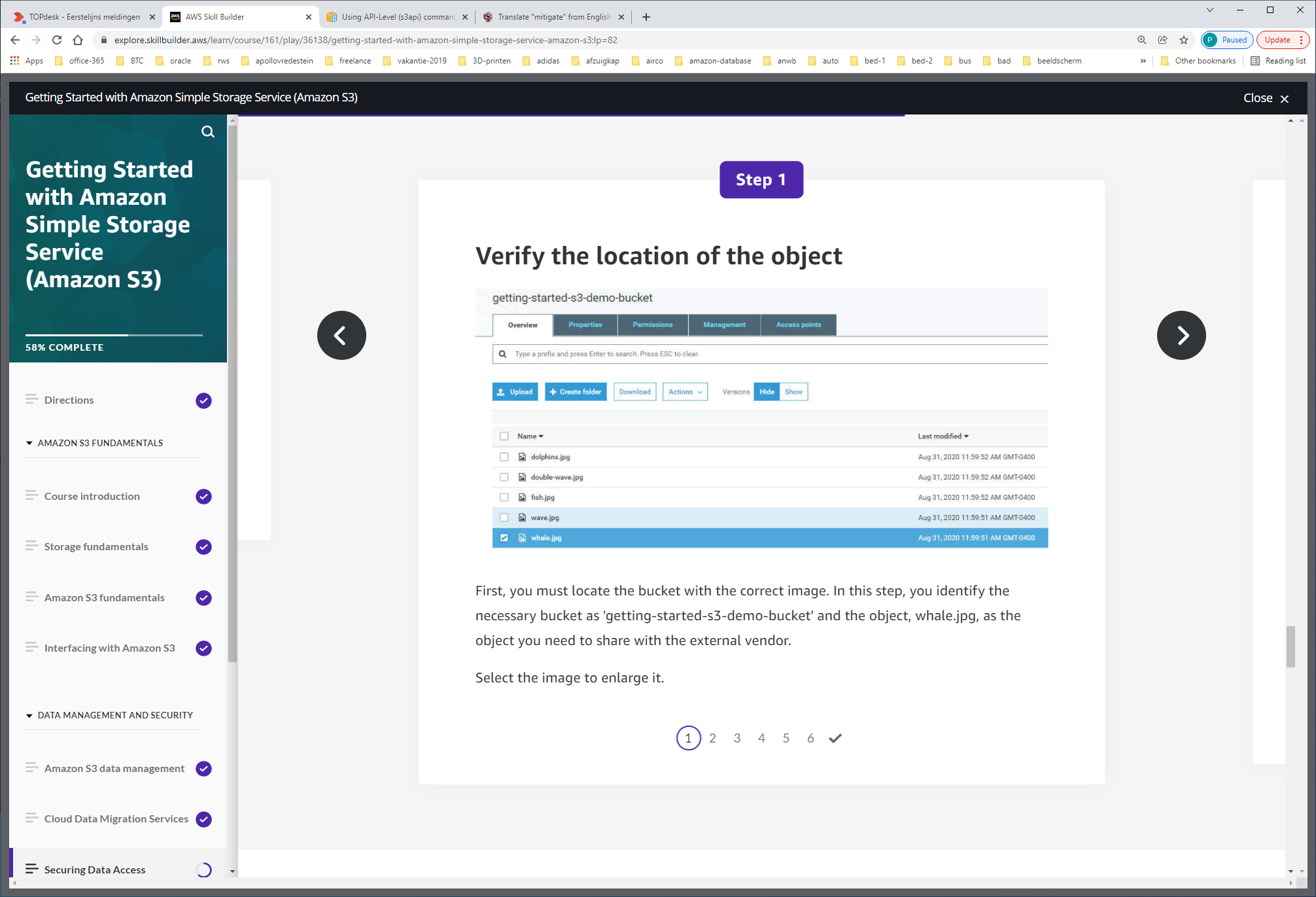
All objects and buckets by default are private and only the object owner has permission to access these objects. However, the object owner can share their objects with others who do not have AWS credentials. They can create a presigned URL to grant time-limited permission to download the objects[ .

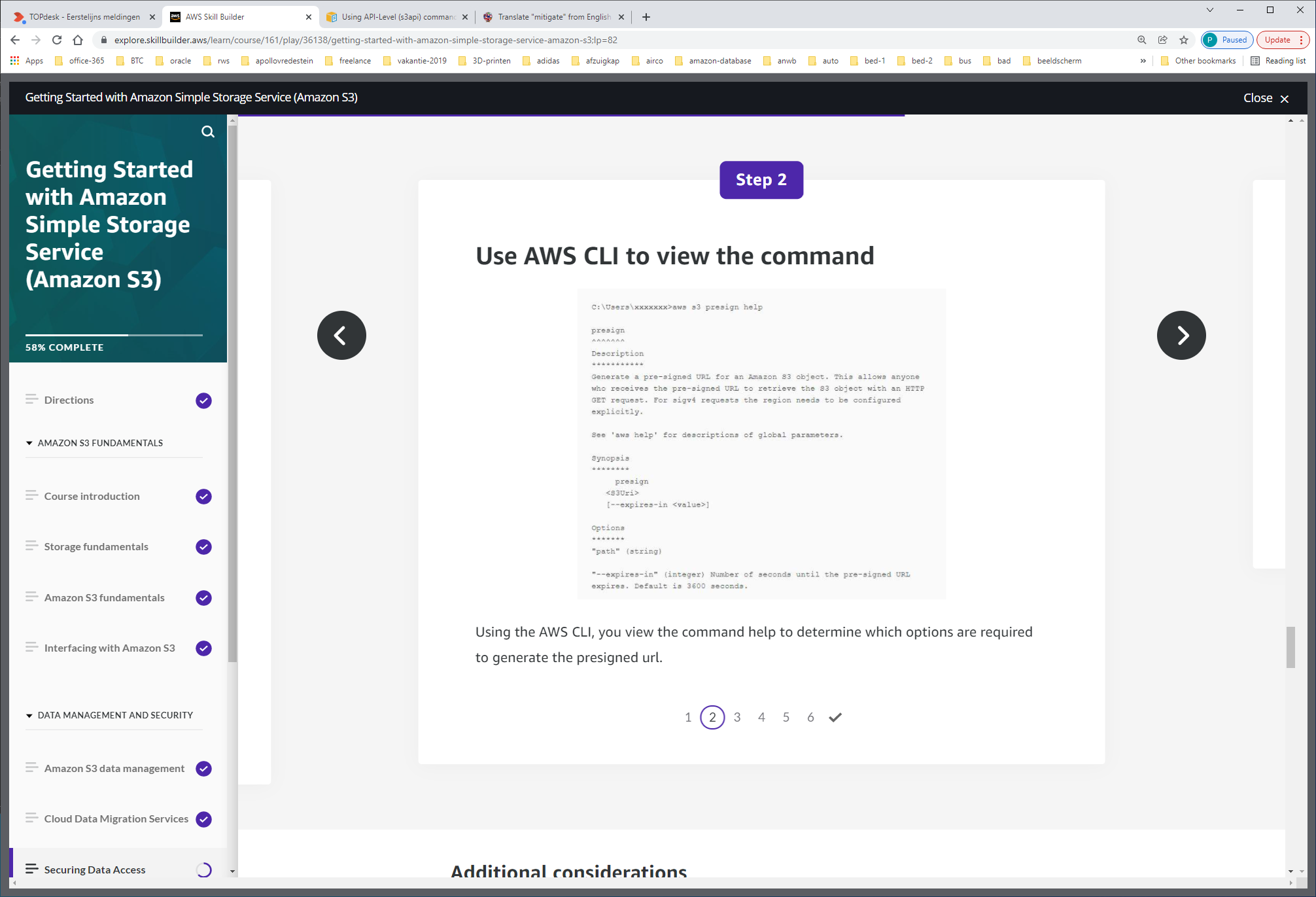
The main purpose of a pre-signed URL is to grant temporary access to the required object. When you create a presigned URL, you must provide your security credentials and then specify   
-a bucket name,   
-an object key,   
-an HTTP method (PUT for uploading objects),   
-and an expiration date and time.

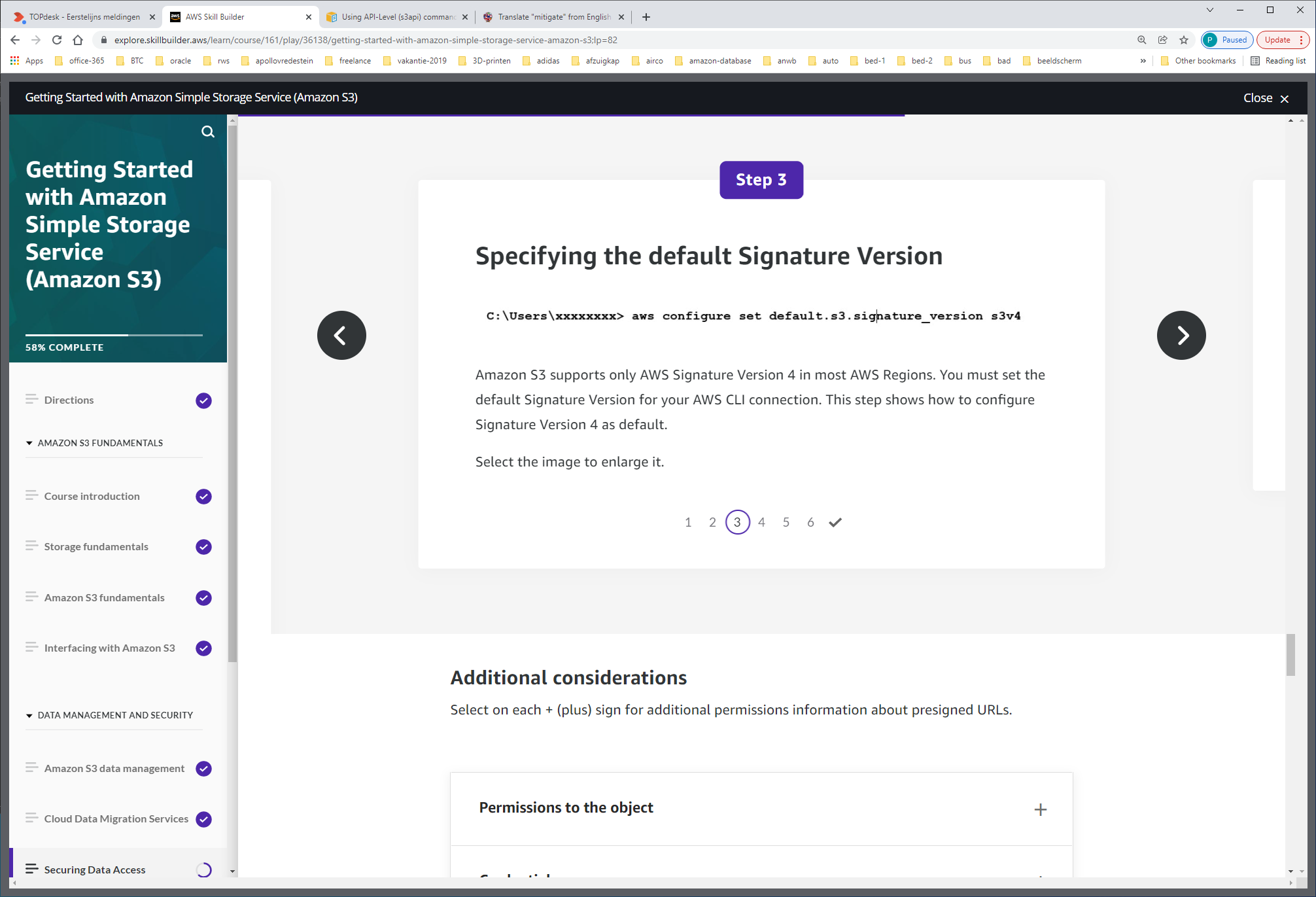
Anyone who receives the presigned URL can then access the object. For example, if you have a video in your bucket and both the bucket and the object are private, you can share the video with others by generating a presigned URL. A presigned URL remains valid for a limited period of time, which is specified when the URL generates. You can use presigned URLs to embed clickable links, which can be valid for up to seven days, in HTML.

A use case scenario for presigned URLs is that you can grant temporary access to your Amazon S3 resources. For example, you can embed a presigned URL on your website or alternatively use it in command line client (such as Curl) to download objects. You could also programmatically generate a presigned URL to allow a user to an upload an object to a bucket.

CREATE A PRE-SIGNED URL

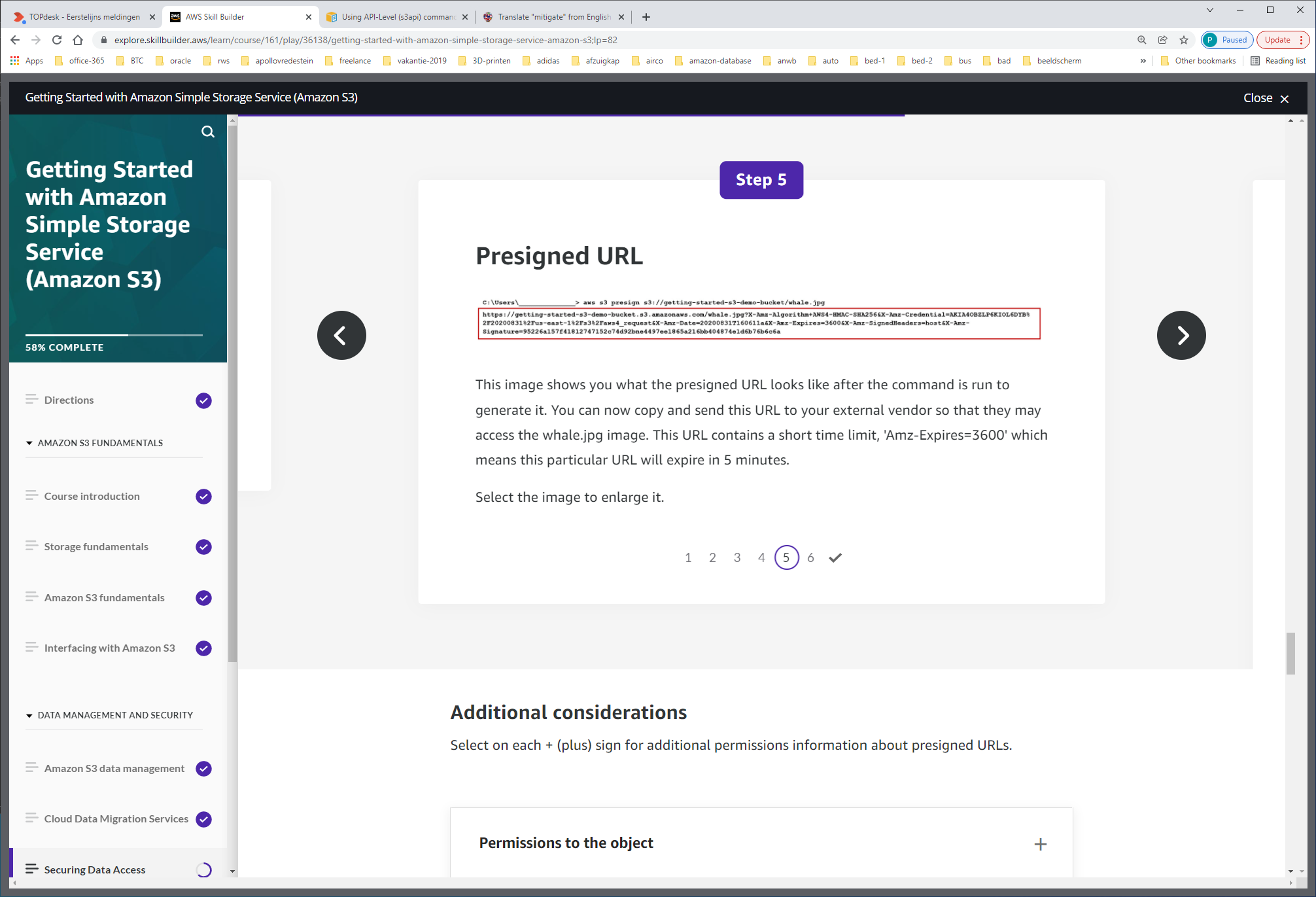


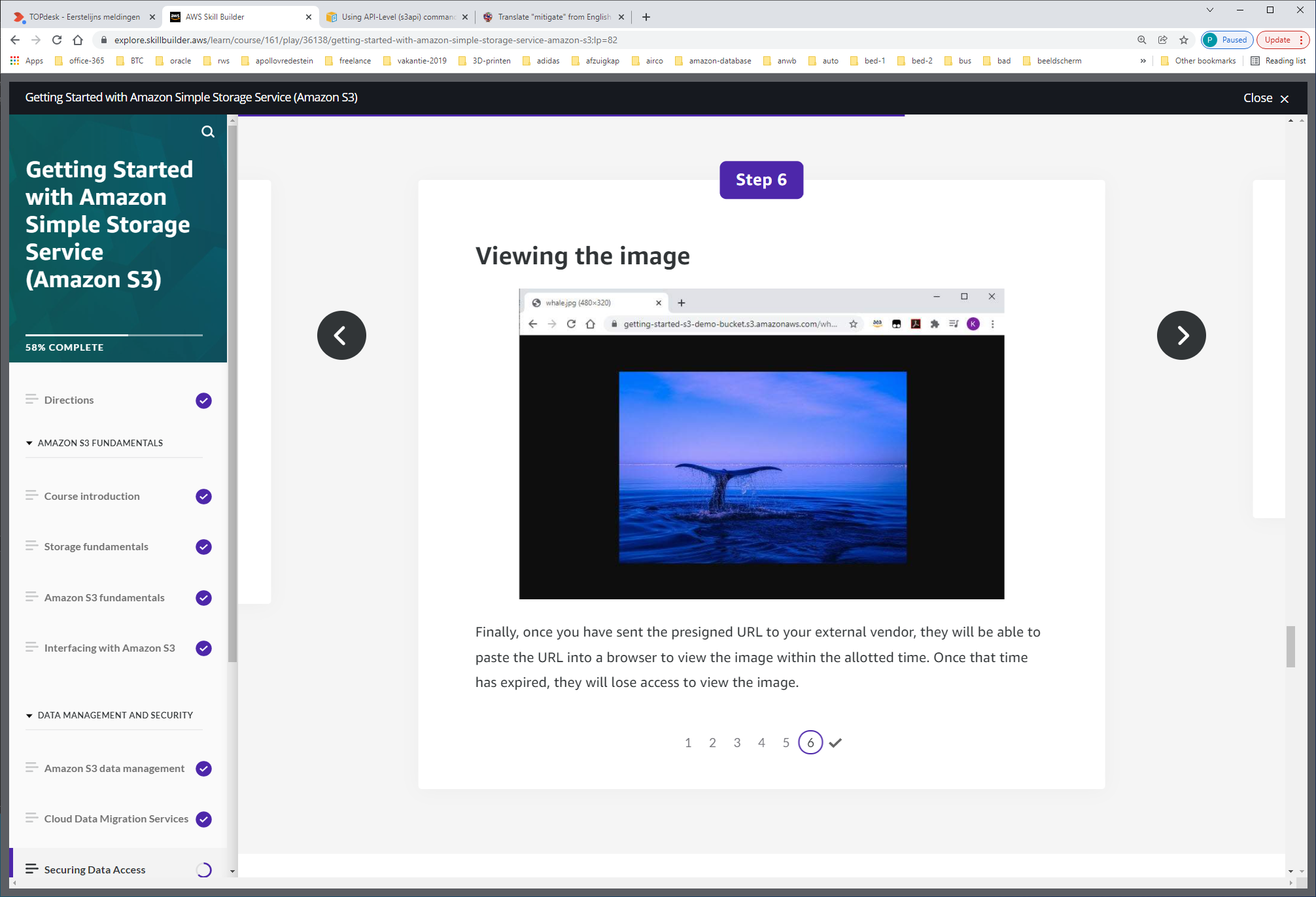




Afbeelding met tekst, schermafbeelding, computer, computer

Automatisch gegenereerde beschrijving





**Permissions to the object**

Anyone with valid security credentials can create a presigned URL. However, to successfully access an object, someone who has permission to perform the operation must create the presigned URL.

**Credentials**

The credentials that you can use to create a presigned URL include:

* IAM instance profile: Valid up to 6 hours
* AWS Security Token Service: Valid up to 36 hours when signed with permanent credentials, such as the credentials of the AWS account root user or an IAM user
* IAM user: Valid up to 7 days when using AWS Signature Version 4

To create a presigned URL that's valid for up to 7 days, first designate IAM user credentials (the access key and secret access key) to the SDK that you're using. Then, generate a presigned URL using AWS Signature Version 4.

**Token expiration**

If you created a presigned URL using a temporary token, then the URL expires when the token expires, even if you created the URL with a later expiration time.

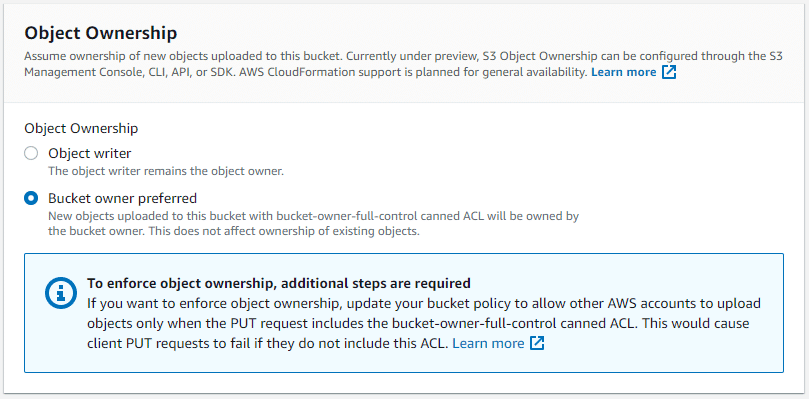
**Amazon S3 Object Ownership**

Prior to the addition of Amazon S3 Object Ownership, an S3 object was owned by the AWS account that uploaded the object. If a bucket owner uploads an object, the bucket owner remains the owner of that object. If another AWS account uploads objects to your bucket, the objects remain owned by the other AWS account that uploaded the object.

AWS has released a new feature that allows all objects written to a bucket to be owned by the bucket owner.  With Amazon S3 Object Ownership, the bucket owner, now has full control of the objects, and may own any new objects written by other accounts automatically.

Amazon S3 Object Ownership has two modes: (see image below)

1. Object writer – The account that is writing the object owns the object.
2. Bucket owner preferred – The bucket owner will own the object if uploaded with the **bucket-owner-full-control** canned ACL. Without this setting and canned ACL, the object is uploaded to the bucket but remains owned by the uploading account.



Edit Object Ownership options.

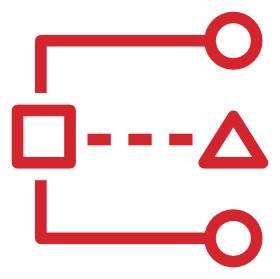
**Enforcing S3 Object Ownership**

After setting S3 Object Ownership to bucket owner preferred, you can add a bucket policy to require all Amazon S3 PUT operations to include the bucket-owner-full-control canned ACL. This ACL grants the bucket owner full control of new objects, and with the S3 Object Ownership setting it transfers object ownership to the bucket owner. If the uploader fails to meet the ACL requirement in their upload, the request will fail. This enables bucket owners to enforce uniform object ownership across all newly uploaded objects in their buckets.

For additional information on Amazon S3 Object Ownership choose this [link](https://docs.aws.amazon.com/AmazonS3/latest/dev/about-object-ownership.html).

**Access Analyzer**

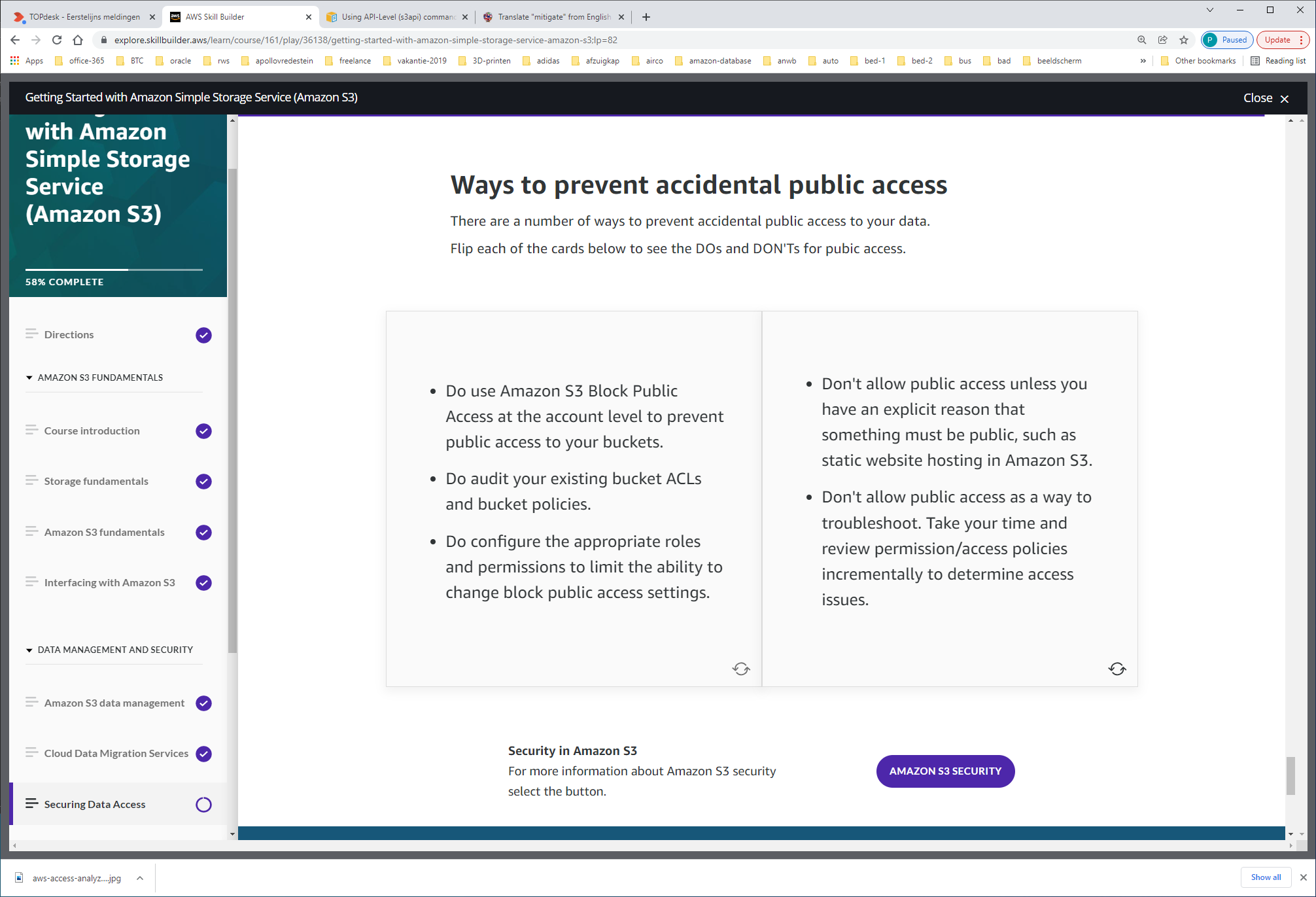
On the Amazon S3 console, you can use Access Analyzer for Amazon S3 to review all buckets that have bucket access control lists (ACLs), bucket policies, or access point policies that grant public or shared access

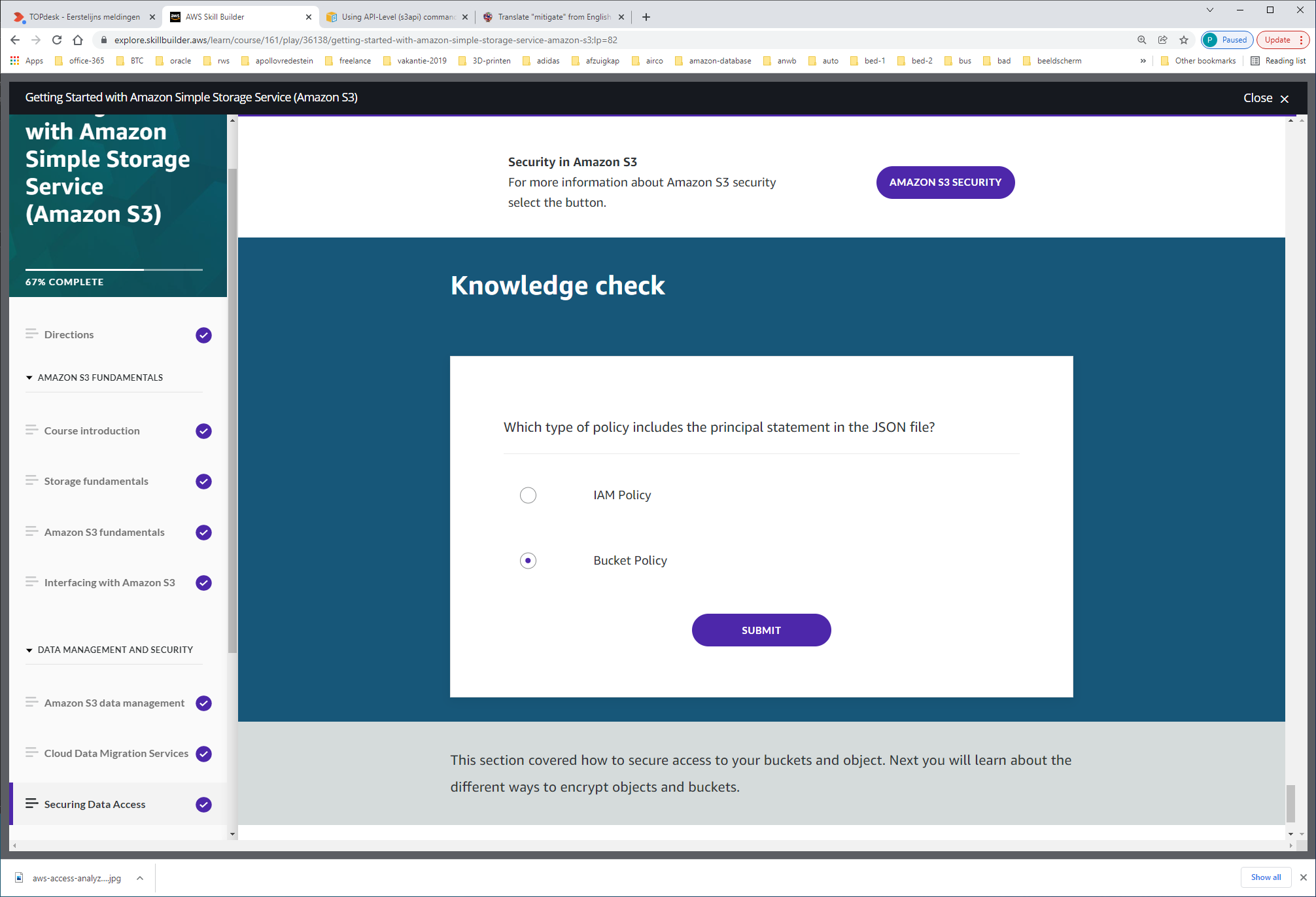


AWS Access Analyzer for Amazon S3

Access Analyzer for Amazon S3 alerts you to buckets configured to allow access to anyone on the internet or other AWS accounts, including AWS accounts outside of your organization. For each public or shared bucket, you receive findings that report the source and level of public or shared access.

DO's & DON'Ts





Because bucket policies grant access to another AWS account or IAM user, you must specify the principal, or the user to whom you are granting access, as a "Principal" in the bucket policy.

# Encrypting Data

**Data in-transit and data at-rest**

Data protection refers to protecting data while in-transit (data traveling to and from Amazon S3) and while at rest (while stored on disks in Amazon S3 data centers).

You can protect data in transit using Secure Socket Layer/Transport Layer Security (SSL/TLS) or client-side encryption. By providing the appropriate level of protection for your data in transit, you protect the confidentiality and integrity of your workload’s data from any bad actor who may intercept the data while in transit. Because the AWS API is a REST service that supports SSL/TLS connections, all official AWS SDKs and CLI tools connect to the AWS API using SSL/TLS by default.

For data at rest (data already stored on disk), Amazon S3 provides you with two options:

-Server-side encryption,

-and client-side encryption.

**Server-side encryption**

When you use server-side encryption, Amazon S3 encrypts an object before saving it to disk and decrypts it when you download it. As long as you authenticate and have access permissions, there is no difference in how you access encrypted or unencrypted objects. For example, if you share your objects using a presigned URL, that URL works the same way for both encrypted and unencrypted objects. Additionally, when you list objects in your bucket, the list API returns a list of all objects, regardless of their encryption.

**Three server-side encryption options**

You have three mutually exclusive options, depending on how you choose to manage the encryption keys.

Server-Side Encryption with Amazon S3-Managed Keys (SSE-S3)

When you use Server-Side Encryption with Amazon S3-Managed Keys (SSE-S3), each object encrypts with a unique key. As an additional safeguard, it encrypts the key itself with a master key that it regularly rotates. Amazon S3 server-side encryption uses one of the strongest block ciphers available, 256-bit Advanced Encryption Standard (AES-256), to encrypt your data.

For additional information: <https://docs.aws.amazon.com/AmazonS3/latest/dev/UsingServerSideEncryption.html>

Server-Side Encryption with Customer Master Keys (CMKs) Stored in AWS Key Management Service (SSE-KMS)

Server-Side Encryption with Customer Master Keys (CMKs) Stored in AWS Key Management Service (SSE-KMS) is similar to SSE-S3, but with some additional benefits and charges for using this service. There are separate permissions for the use of a CMK that provides added protection against unauthorized access of your objects in Amazon S3. SSE-KMS also provides you with an audit trail showing when and who used the CMK. Additionally, you can choose to create and manage customer managed CMKs, or use AWS managed CMKs that are unique to you, your service, and your Region.

For more information:

link: <https://docs.aws.amazon.com/AmazonS3/latest/dev/UsingKMSEncryption.html>

Server-Side Encryption with Customer-Provided Keys (SSE-C)

With Server-Side Encryption with Customer-Provided Keys (SSE-C), you manage the encryption keys and Amazon S3 manages the encryption, as it writes to disks, and decryption, when you access your objects. With this option, the customer is responsible for managing and rotating the keys, and without access to these keys the Amazon S3 data can not be decrypted.

For more Information: <https://docs.aws.amazon.com/AmazonS3/latest/dev/ServerSideEncryptionCustomerKeys.html>

You can't apply different types of server-side encryption to the same object simultaneously.

**Client-side encryption**

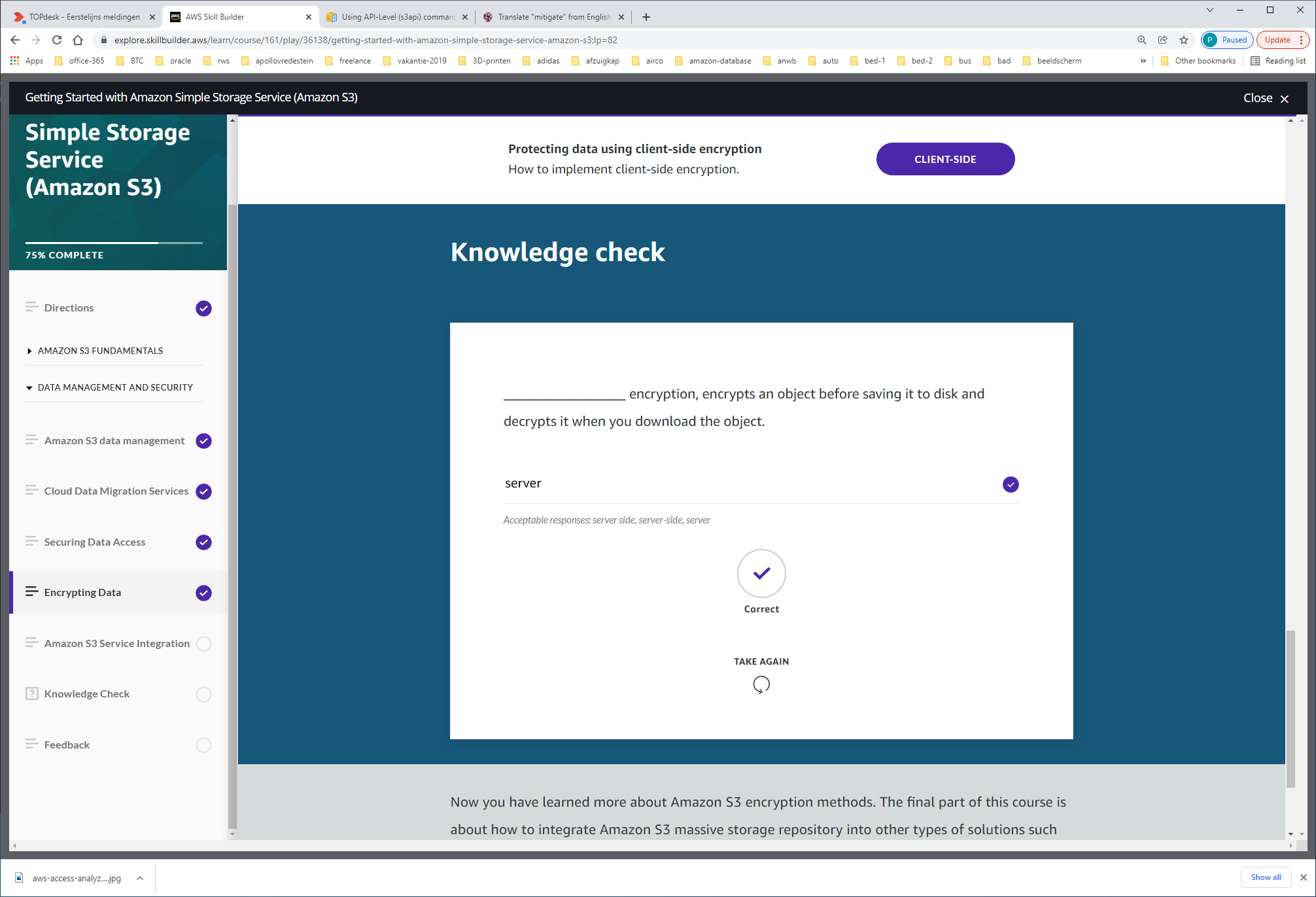
Client-side encryption is the act of encrypting sensitive data before sending it to Amazon S3. When using client-side encryption, the encryption performs locally and your data never leaves the run environment unencrypted. You maintain possession of your master encryption keys, and they are never sent to AWS therefore, it is important that you safely store them (i.e., as a file or using a separate key management system) and load them when uploading or downloading objects. This ensures that no one outside of your environment has access to your master keys and without access to the master keys; your data cannot be decrypted. If your master encryption keys are lost, you will not be able to decrypt your own data, therefore it is essential that if you use client-side encryption, that you store your keys safely.

To enable client-side encryption, you have the following options:

* Use a customer master key (CMK) stored in AWS Key Management Service (AWS KMS). With this option, you use an AWS KMS CMK for client-side encryption when uploading or downloading data in Amazon S3.
* Use a master key that you store within your application. With this option, you provide a client-side master key to the Amazon S3 encryption client. The client uses the master key only to encrypt the data encryption key that it generates randomly.

Important

Your client-side master keys and your unencrypted data not sent to AWS. It's important that you safely manage your encryption keys. If you lose them, you can't decrypt your data.



# Amazon S3 Service Integration

As organizations are collecting and analyzing increasing amounts of data, traditional on-premises solutions for data storage, data management, and analytics can no longer keep pace. Data siloes that aren’t built to work well together make it difficult to consolidate storage so that you can perform comprehensive and efficient analytics. This limits an organization’s agility, and ability to derive more insights and value from its data. It also limits the capability to adopt more sophisticated analytics tools and processes as its needs evolve.

Amazon S3 has a broad integration with a variety of services and use cases. In this section we use data lakes and high performance computing as two such examples.

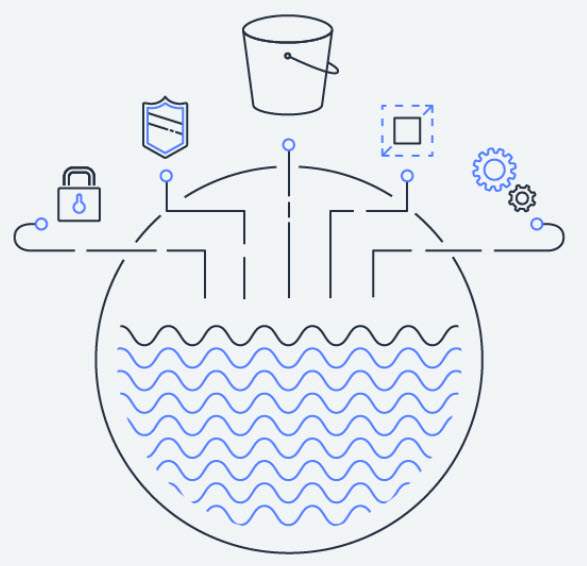


Image of a data lake.

**A data lake**

A data lake is a centralized repository that allows you to migrate, store, and manage all structured and unstructured data at an unlimited scale. Once the data is centralized, you can extract value and gain insights from your data through analytics and machine learning.

A data lake makes the data and the analytics tools available to more of your users, across more lines of business enabling them to get the business insights they need, whenever they need them.

**Amazon S3 as a data lake**

The Amazon S3-based data lake solution uses Amazon S3 as its primary storage platform. Amazon S3 provides an optimal foundation for a data lake because of its virtually unlimited scalability. You can nondisruptively increase storage from gigabytes to petabytes of content.

Amazon S3 is designed for 99.999999999% (11 9s) of durability. It has scalable performance, ease-of-use features, native encryption, and access control capabilities. It also integrates with a broad portfolio of AWS and third party ISV data processing tools.

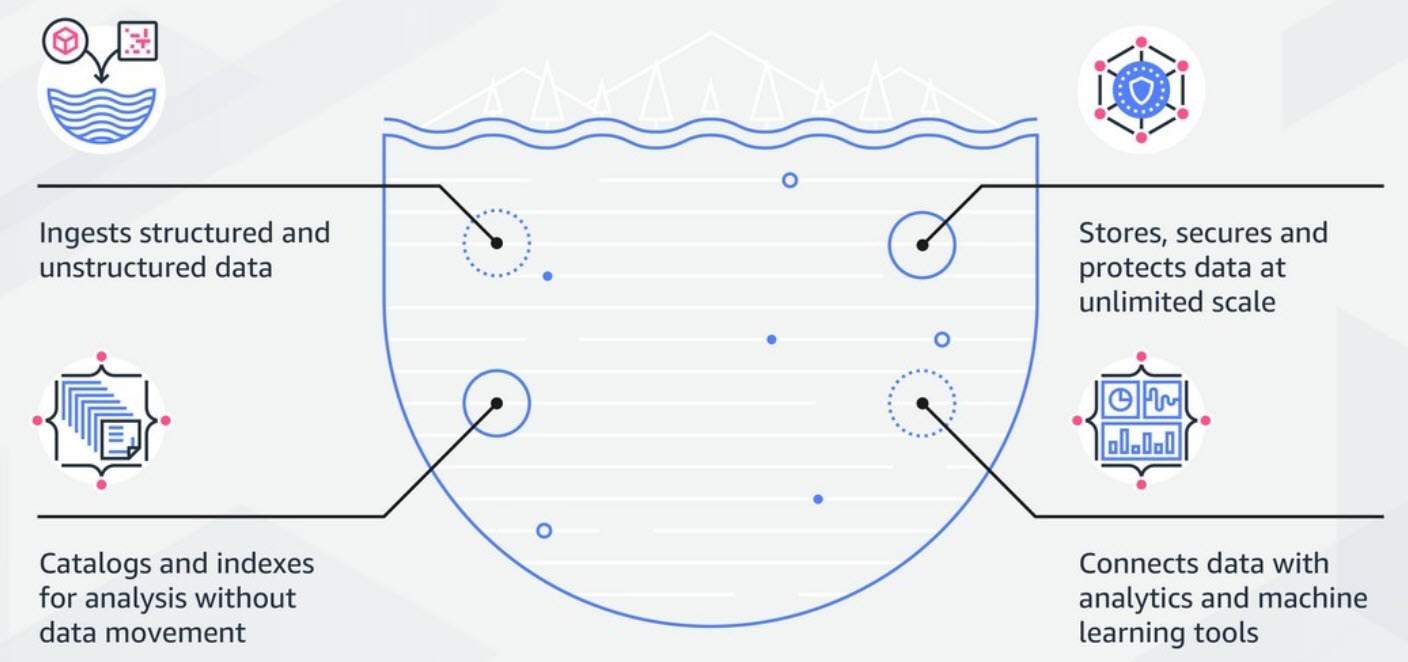


Image of how a data lake functions.

Amazon Web Services (AWS) has developed a data lake architecture that allows you to build data lake solutions cost-effectively using Amazon Simple Storage Service and other services.

**Decoupling storage from compute**

In traditional Hadoop and data warehouse solutions, storage and compute remain tightly coupled, making it difficult to optimize costs and data processing workflows. With Amazon S3, you can cost-effectively store all data types in their native formats. Then, you can launch as many or as few virtual servers as you need using Amazon Elastic Compute Cloud (EC2), and use AWS analytics tools to process your data. You can also optimize your EC2 instances to provide the right ratios of CPU, memory, and bandwidth for best performance.

**Centralized data architecture**

Amazon S3 makes it easy to build a multi-tenant environment, where many users can bring their own data analytics tools to a common set of data. This improves both cost and data governance over that of traditional solutions, which require multiple copies of data to be distributed across multiple processing platforms.

**Integration with clusterless and serverless AWS services**

You can use Amazon S3 with Amazon Athena, Amazon Redshift Spectrum, Amazon Rekognition, and AWS Glue to query and process data. Amazon S3 also integrates with AWS Lambda serverless computing to run code without provisioning or managing servers. With all of these capabilities, you only pay for the actual amounts of data you process or for the compute time consumed.

**Standardized APIs**

Amazon S3 REST APIs are easy to use, and supported by most major third-party independent software vendors (ISVs), including leading Apache Hadoop and analytics tool vendors. This allows customers to bring the tools they are comfortable and knowledgeable about to help them perform analytics on data in Amazon S3.

**Data cataloging**

The earliest challenges that inhibited building a data lake were keeping track of all of the raw assets, as they were loaded into the data lake. Then, keeping track of all of the new data assets and versions created by data transformation, data processing, and analytics.

Thus, an essential component of an Amazon S3-based data lake is the data catalog. The data catalog provides a query-able interface of all assets stored in the data lake’s S3 buckets. The design of the data catalog is to provide a single source of truth about the contents of the data lake.

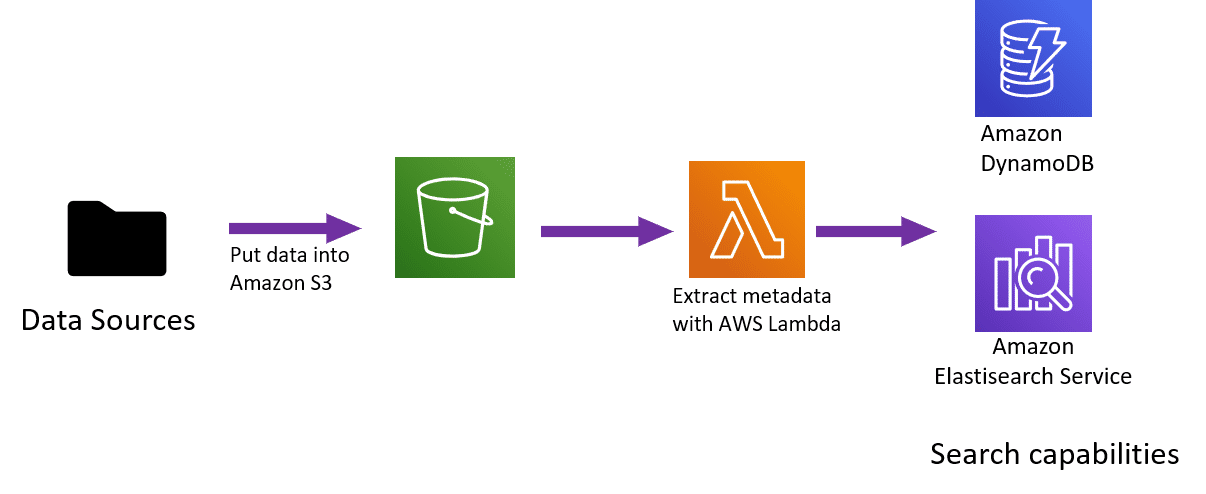


Figure: Comprehensive data catalog using AWS Lambda, Amazon DynamoDB, and Amazon Elasticsearch Service

You can create the comprehensive data catalog by using standard AWS services like AWS Lambda, Amazon DynamoDB, and Amazon Elasticsearch Service (Amazon ES). At a high level, you can use Lambda triggers to populate DynamoDB tables with object names and metadata. When those objects are put into Amazon S3 then Amazon ES is used to search for specific assets, related metadata, and data classifications.

**AWS Glue**

AWS Glue is a fully managed ETL (extract, transform, and load) service that makes it simple and cost-effective to categorize your data. You can use AWS Glue to organize, cleanse, validate, and format data for storage in a data warehouse or data lake. The AWS Glue Data Catalog is an index to the location, schema, and runtime metrics of your data. In order to create your data warehouse or data lake, you must catalog this data.

**In-Place data querying**

One of the most important capabilities of a data lake built into AWS is the ability to do in-place transformation and querying of data assets, without having to provision and manage clusters. This allows you to run sophisticated analytic queries directly on your data stored in Amazon S3. You don’t have to copy and load data into separate analytics platforms or data warehouses.

This makes the ability to analyze vast amounts of unstructured data accessible to any data lake user who can use SQL. In addition, it makes it more cost effective than the traditional method of performing an ETL process, creating a Hadoop cluster or data warehouse, loading the transformed data into these environments, and then running query jobs.

Amazon Athena and Amazon Redshift Spectrum provide the in-place querying capabilities of an Amazon S3 data lake.

**Amazon Athena**

Amazon Athena is an interactive query service that makes it easy for you to analyze data directly in Amazon S3, using standard SQL. You can get results in a matter of seconds. Athena is serverless, so there is no infrastructure to set up or manage. You only pay for the volume of data assets scanned during the queries you run.

You can use Athena to process unstructured, semi-structured, and structured data sets. It integrates with Amazon QuickSight for easy visualization. It can also be used with third-party reporting and business intelligence tools by connecting these tools to Athena with a JDBC driver.



Amazon Athena provides in-place SQL querying for an Amazon S3 data lake.

**Amazon Redshift Spectrum**

A second way to perform in-place querying of data assets in an Amazon S3-based data lake is to use Amazon Redshift Spectrum. Amazon **Redshift** is a large-scale, managed data warehouse service used with data assets in Amazon S3. However, data assets must be loaded into Amazon Redshift before queries run.



Amazon Redshift Spectrum enables you to run Amazon Redshift SQL queries directly against  data stored in an Amazon S3-based data lake.

**Amazon Redshift Spectrum enables you to run Amazon Redshift SQL queries directly against data stored in an Amazon S3-based data lake**.

Amazon Redshift Spectrum applies sophisticated query optimization, scaling processing across thousands of nodes so results are fast—even with large data sets and complex queries. You can directly query a wide variety of data assets stored in the data lake, including CSV, TSV, Parquet, Sequence, and RCFile.

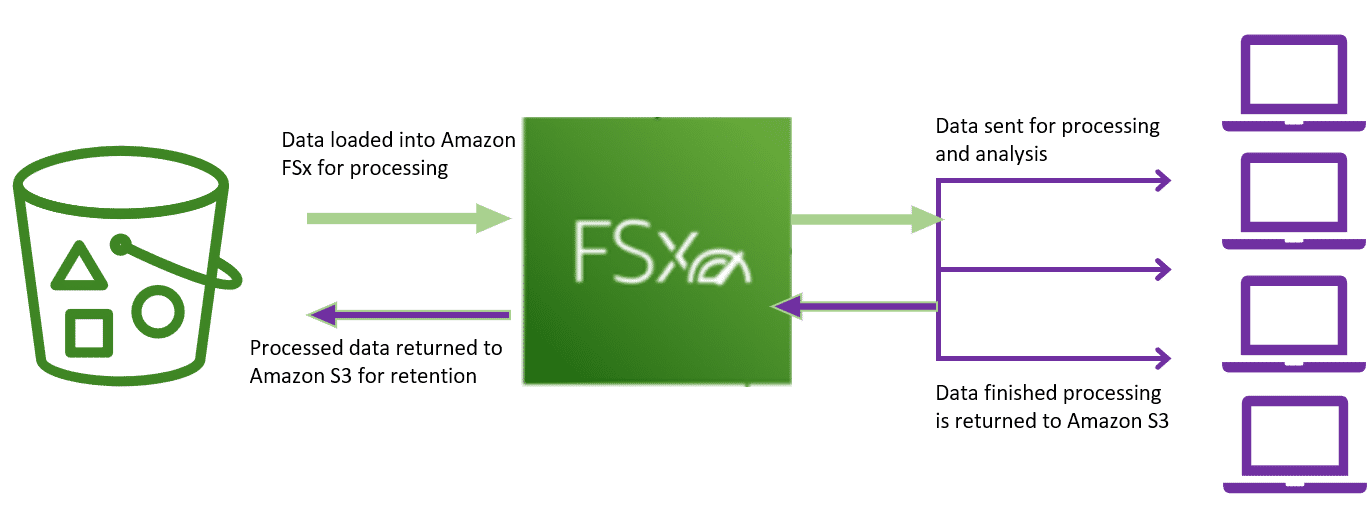
Because Amazon Athena and Amazon Redshift share a **common data catalog** and data formats, you can use both Athena and Redshift Spectrum against the same data assets.

You would typically use Athena for ad hoc data discovery and SQL querying, and then use Redshift Spectrum for more complex queries and scenarios where a large number of data lake users want to run concurrent BI and reporting workloads.

**Amazon FSx for Lustre and Amazon S3 data lakes**

Amazon FSx for Lustre, is a fully managed file system that is optimized for compute-intensive workloads, such as high performance computing, machine learning, and media data processing workflows.

With Amazon FSx for Lustre, you can launch and run a Lustre file system that can process massive data sets at up to hundreds of gigabytes per second of throughput, millions of IOPS, and sub-millisecond latencies.  Amazon FSx for Lustre file systems can link to Amazon S3 buckets, allowing you to access and process data concurrently from a high-performance file system.



Amazon FSx for Lustre can use Amazon S3 as a raw data repository as well as a repository for processed data.

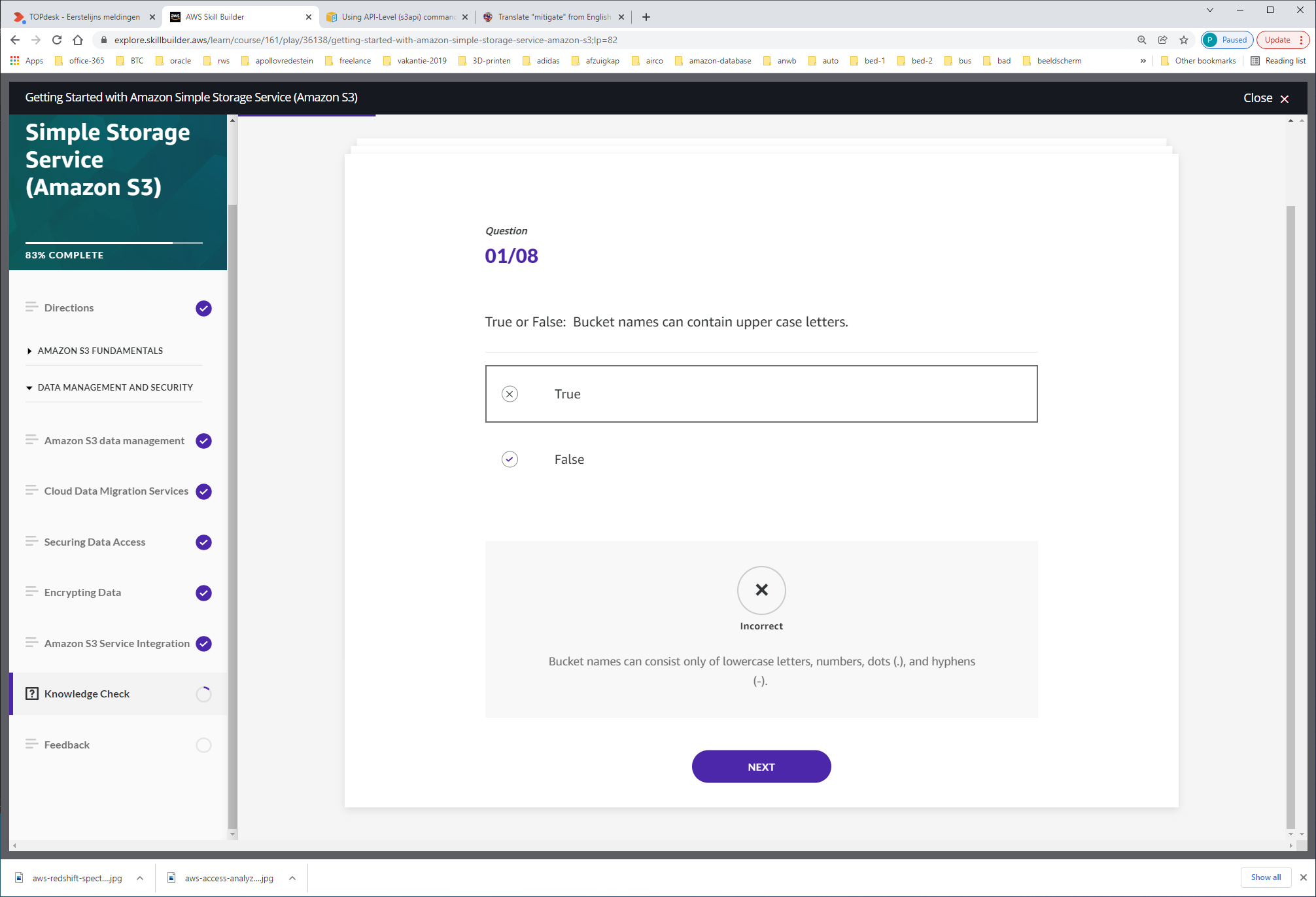
Amazon FSx for Lustre can use Amazon S3 as a raw data repository as well as a repository for processed data. It makes it easy to process your cloud datasets in Amazon S3.

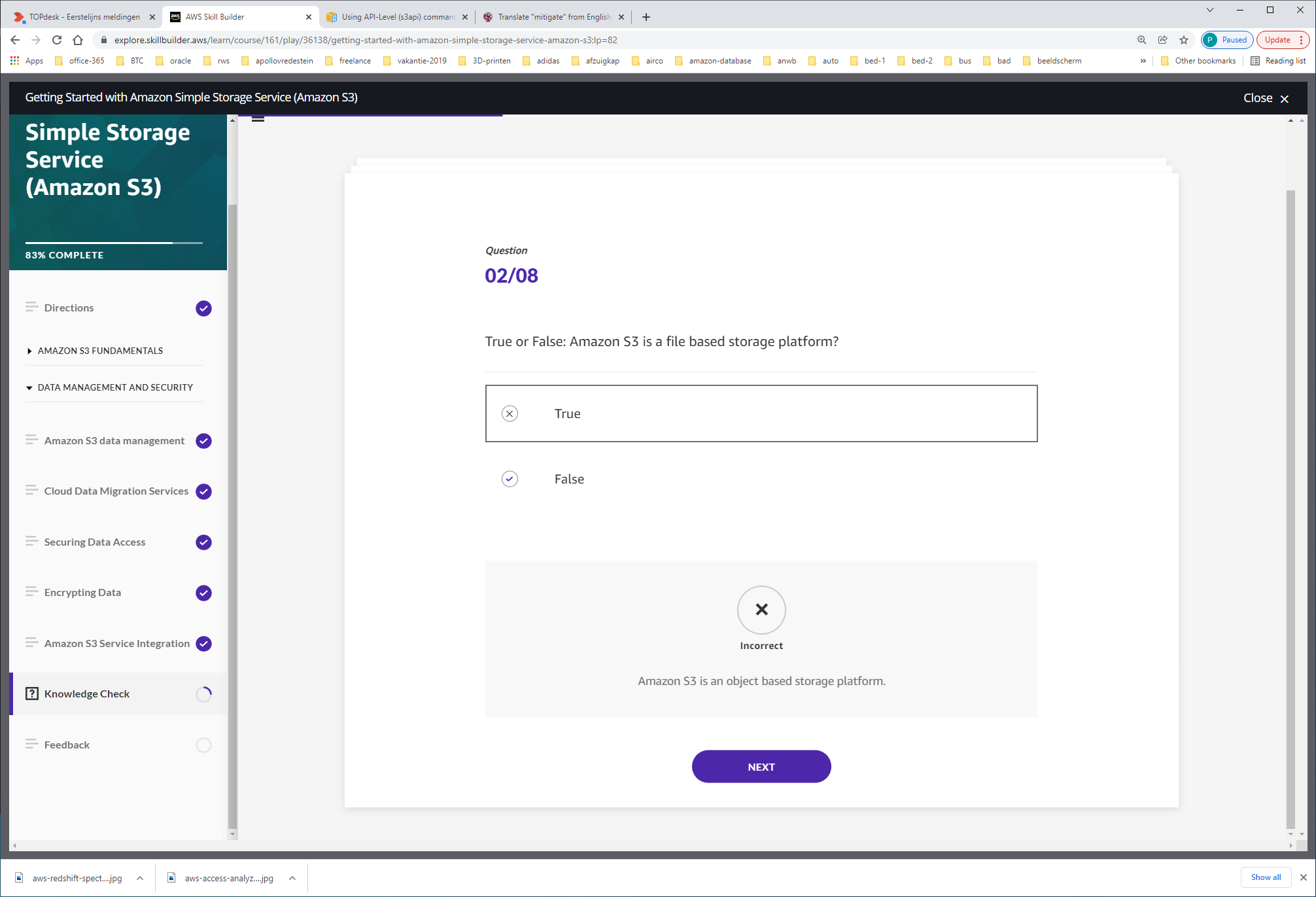
When linked to an S3 bucket, FSx for Lustre transparently presents objects as files, allowing you to run your workload without managing data transfer from S3. As the contents of your S3 bucket change, FSx for Lustre automatically updates your file system with the latest data available to run your workload.

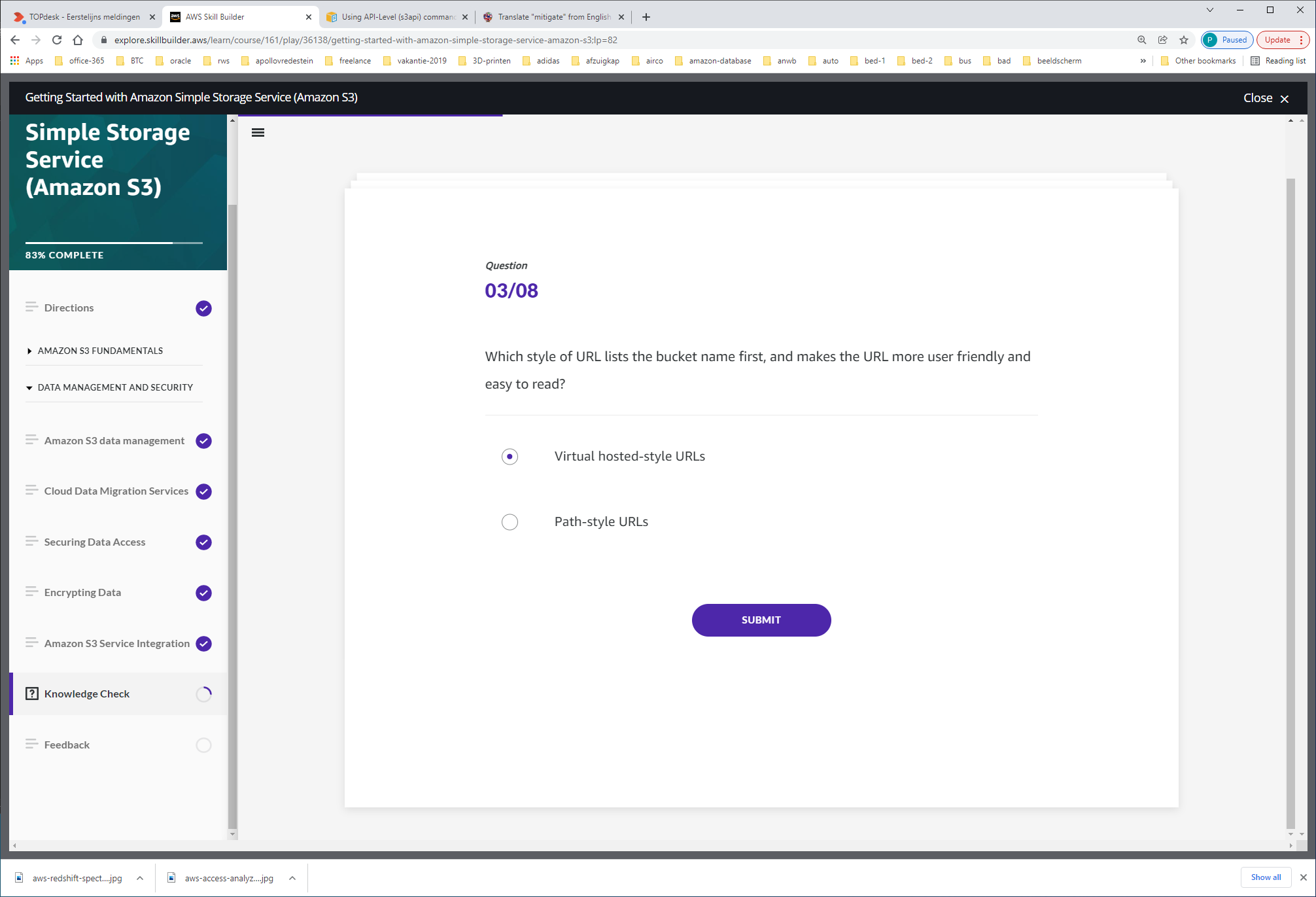
Afbeelding met tekst, schermafbeelding, monitor

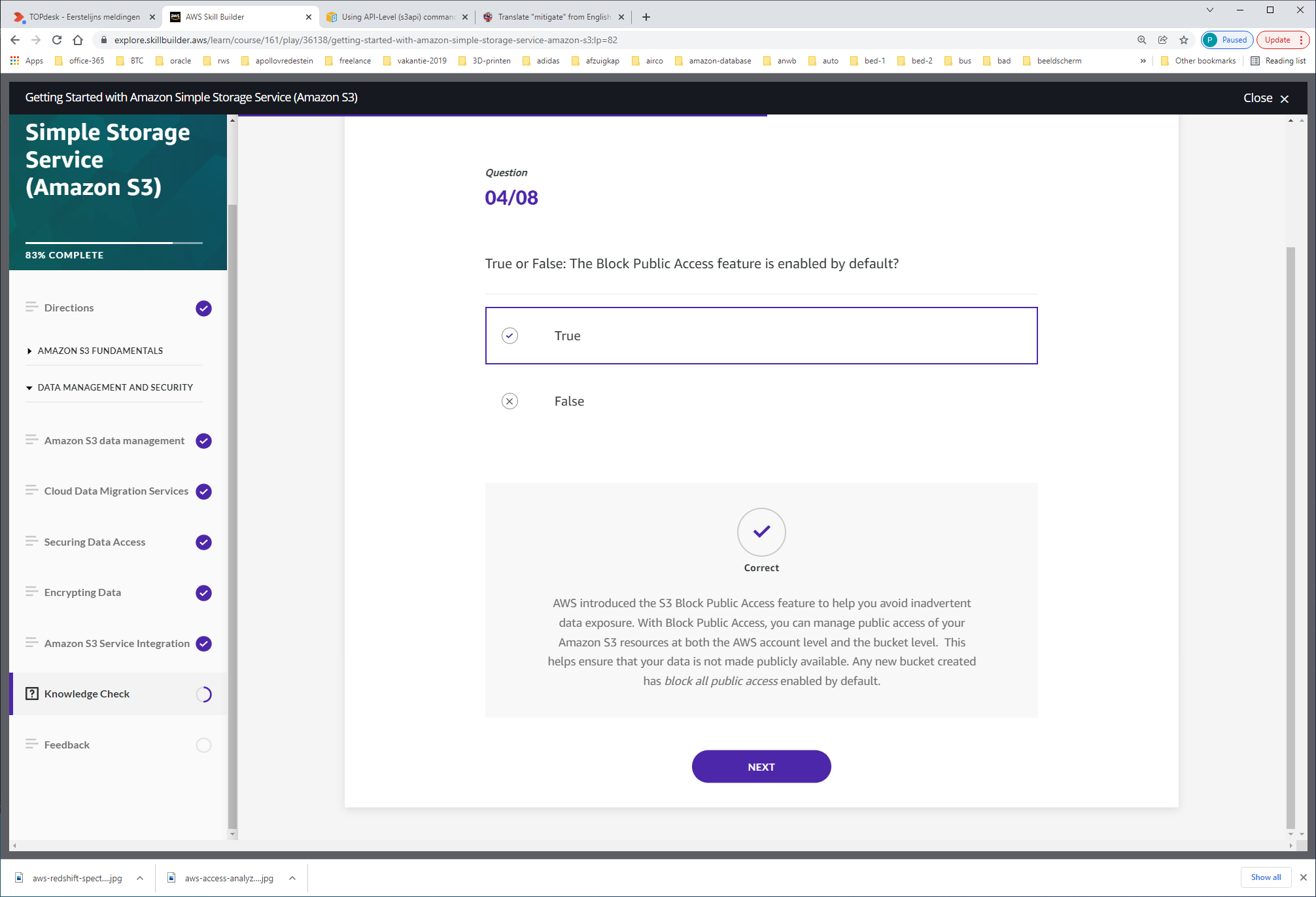
Automatisch gegenereerde beschrijving

KNOWLEDGE-CHECK









Afbeelding met tekst, schermafbeelding, monitor

Automatisch gegenereerde beschrijving

