**CHAPTER 2**

Storage on AWS

In this chapter, you will

•   Learn the storage offerings of AWS

•   Learn to use Amazon S3

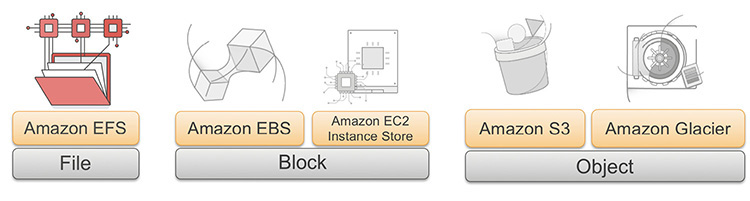
•   Learn to use Amazon S3 Glacier

•   Learn to use Amazon Elastic Block Store

•   Learn to use Amazon Elastic File System

•   Learn how to move a large amount of data to AWS

The storage offerings of AWS can be divided into three major categories, as shown in [Figure 2-1](https://learning.oreilly.com/library/view/aws-certified-solutions/9781260470192/ch2.xhtml#fig2_1).



**Figure 2-1** AWS storage platform

•   **Object storage** An object is a piece of data, like a document, image, or video, that is stored with some metadata in a flat structure. It provides that data to applications via APIs over the Internet. It is simple to build anything on top of an object store. For example, you can easily develop a web application on top of Amazon S3 that delivers content to users by making API calls over the Internet.

•   **Block storage** In block storage, data is presented to your instance as a disk volume. It provides low, single-digit-latency access to single Amazon EC2 instances. Elastic Block Store is popular, for example, for boot volumes and databases.

•   **File storage** In file storage, data is presented via a file system interface and with file system semantics to instances. When attached to an instance, it acts just like a local file system. Amazon Elastic File System (EFS) provides shared access to data via multiple Amazon EC2 instances, with low latencies.

AMAZON SIMPLE STORAGE SERVICE (S3)

Amazon launched S3 in 2006. Amazon S3 is an object store and is the backbone for many other services used at Amazon. It has a nice web interface to store and retrieve any amount of data from anywhere around the world. The capacity of S3 is unlimited, which means there is no limit to the amount of data you can store in S3. It is highly durable and has 99.99999999999 percent durability. According to Amazon, this durability level corresponds to an average annual expected loss of 0.000000001 percent of objects. For example, if you store 10,000 objects with Amazon S3, you can on average expect to incur a loss of a single object once every 10,000,000 years. In addition, Amazon S3 is designed to sustain the concurrent loss of data in two facilities.

It is fundamentally different from other file repositories because it does not have a file system. All objects are stored in a flat namespace organized by buckets. It is a regional service; that is, content is automatically replicated within a region for durability. It is one of the most popular object stores available on the Internet today. In this chapter, you’ll first evaluate some of the advantages of Amazon S3, which makes it uniquely popular among customers.

ADVANTAGES OF AMAZON S3

The following are the advantages of using Amazon S3:

•   **Simple** Amazon S3 is really easy to use. It has an intuitive graphical web-based console in which the data can be uploaded, downloaded, and managed. S3 also has a mobile app in which it can be managed. For easy integration with third parties, S3 provides REST APIs and SDKs. You can also easily access S3 via the AWS CLI, which offers a familiar Linux-like interface.

•   **Scalable** Amazon S3 is infinity scalable. You can store unlimited data in it without worrying about storage needs. You don’t have to do any kind of capacity planning to store data in S3. If your business needs petabytes of data, you should be able to store that in S3 easily and quickly. You can scale up or scale down anytime as per your business requirements. S3 is designed to scale infinitely.

•   **Durable** Amazon S3 is the only service that provides 99.999999999 percent durability of the objects stored in it. The underlying infrastructure is designed in such a way that this durability is achieved. The data is stored across multiple data centers and in multiple devices in a redundant manner. Amazon S3 is designed to sustain concurrent data loss in two facilities.

•   **Secured** Amazon S3 supports encryption, and the data can be automatically encrypted once it is uploaded. S3 also supports data transfer over SSL. Using AWS Identity and Access Management (IAM), you should be able to manage granular permissions and access to an S3 bucket.

•   **High performance** Amazon S3 supports multipart uploads to help maximize network throughput and resiliency and lets you choose the AWS region to store your data close to the end user and minimize network latency. Also, Amazon S3 is integrated with Amazon CloudFront, a content delivery web service that distributes content to end users with low latency, high data transfer speeds, and no minimum usage commitments.

•   **Available** Amazon S3 is designed to provide 99.99 percent availability of the objects annually. The SLA level of 99.99 percent uptime/availability gives the following periods of potential downtime/unavailability:

Daily: 8.6 seconds

Weekly: 1 minute and 0.5 seconds

Monthly: 4 minutes and 23.0 seconds

Yearly: 52 minutes and 35.7 seconds

•   **Low cost** Amazon S3 is very cost-effective and allows you to store a large amount of data at a low cost. There is no minimum cost associated with S3, and you pay only for what you need. Also, there are no up-front costs associated with S3. With the volume discount, the more data you store, the cheaper it becomes. You can further lower the cost by storing the data in a different class of S3 such as infrequent access or reduced redundancy or by creating a lifecycle policy in which you can archive old files to Amazon S3 Glacier to further reduce the cost.

•   **Easy to manage** The Amazon S3 storage management feature allows you to take a data-driven approach to storage optimization data security and management efficiency. As a result, you have better intel about your data and can manage the data based on personalized metadata.

•   **Easy integration** Amazon S3 can be easily integrated with third-party tools. As a result, it is easy to build an application on top of S3. S3 is also integrated with other AWS services. As a result, S3 can be used in conjunction with lots of AWS products.

USAGE OF AMAZON S3 IN REAL LIFE

You can use S3 in the following ways:

•   **Backup** Amazon S3 is popular for storing backup files among enterprises. Since the durability of S3 is 99.999999999 percent, losing data is rare. For S3 Standard, S3 Standard-Infrequent Access, and S3 Glacier storage classes, data is distributed in three copies for each file between multiple availability zones (AZs) within an AWS region. As a result, the data cannot be destroyed by a disaster in one AZ. S3 also provides versioning capacity; as a result, you can further protect your data against human error.

•   **Tape replacement** Another popular use case of S3 these days is magnetic tape replacement. Many organizations have started replacing their tape drives or tape infrastructures with S3.

•   **Static web site hosting** If you need to host a static web site, you get everything just by using Amazon S3. There is no need to procure web servers or worry about storage. Since S3 is scalable, it can handle any amount of traffic, and you can store unlimited data. For example, if you put a \*.html file in S3, you can access it from a web browser without the need for a web server. This is a great option for getting read-only information.

•   **Application hosting** Since Amazon S3 provides highly available storage, many customers use it for hosting mobile and Internet-based apps. Since S3 is accessible from anywhere in the world, you can access and deploy your applications from anywhere.

•   **Disaster recovery** Amazon S3 is also used as a disaster recovery solution. Using cross-region replication, you can automatically replicate each S3 object to a different bucket in a different region.

•   **Content distribution** Amazon S3 is often used to distribute the content over the Internet. It allows you to offload your entire storage infrastructure into the cloud, where you can take advantage of Amazon S3’s scalability and pay-as-you-go pricing to handle your growing storage needs. The content can be anything such as files, or you can host media such as photos, videos, and so on. You can also use it as a software delivery platform where customers can download your software. These contents can be distributed either directly from S3 or via Amazon CloudFront.

•   **Data lake** Amazon S3 is becoming extremely popular as a *data lake* solution. A data lake is a central place for storing massive amounts of data that can be processed, analyzed, and consumed by different business units in an organization. Any raw, semiprocessed, or processed data can be stored in S3. It is extremely popular in the world of big data as a big data store to keep all kinds of data. Whether you’re storing pharmaceutical or financial data or multimedia files such as photos and videos, Amazon S3 can be used as your big data object store. Amazon Web Services offers a comprehensive portfolio of services to help you manage big data by reducing costs, scaling to meet demand, and increasing the speed of innovation. S3 is often used with EMR, Redshift, Redshift Spectrum, Athena, Glue, and QuickSight for running big data analytics.

•   **Private repository** Using Amazon S3 you can create your own private repository like with Git, Yum, or Maven.

AMAZON S3 BASIC CONCEPTS

This section covers some Amazon S3 basic terminology and concepts that you will learn about throughout this chapter.

Just like you store water in a bucket, in the cloud you store the objects of the object store in a bucket. So, a *bucket* is actually a container for storing objects in Amazon S3. You can compare a bucket to a folder on a computer where you store various files. You can create multiple folders inside a folder, and in an S3 bucket you can create multiple folders. The name of the bucket must be unique, which means you cannot have two buckets with the same name even across multiple regions. Any object can be uniquely accessible from the bucket using a URL. For example, say an object name is ringtone.mp3, and it has been stored in the bucket newringtones. The file will be accessible using the URL <http://newringtones.s3.amazonaws.com/ringtone.mp3>. The bucket serves the following purposes:

•   Organizes the Amazon S3 namespace at the highest level

•   Identifies the account responsible for charges

•   Plays a role in access control

•   Serves as the unit of aggregation for usage reporting

You can create buckets in any region you want. By default, the data of a bucket is not replicated to any other region unless you do it manually or by using cross-region replication. S3 buckets allow versioning. If you use versioning, whenever an object is added to a bucket, a unique version ID is assigned to the object.

Objects are the fundamental entries stored in Amazon S3. Put simply, anything you store in an S3 bucket is called an *object,* and an object consists of data and metadata. The data portion stores the actual data in Amazon S3. Metadata is a set of name-value pairs describing the object. The metadata also includes additional information such as last-modified date, file type, and so on. An object is uniquely identified within a bucket by a name or key and by a version ID.

Using a key, you can uniquely identify an object in a bucket, which means that every object in a bucket has only one key. You can identify any object in an S3 bucket with a unique combination of bucket, key, and version ID. For example, to understand how you can use a key to identify an object, say the URL of the object of S3 bucket is <http://s3.amazonaws.com/2017-02/pictures/photo1.gif>. In this case, the name of the key is 2017-02/pictures/photo1.gif. When you combine the key with the version of the file (photo1.gif), you can uniquely define the particular object.

You can create an S3 bucket in any region where the service is available. You may want to create an object in a *region* that is near to you to optimize the latency and get a better user experience. You can also choose a region for data compliance purposes or to minimize the cost. Note that the object stored in the region never leaves the region unless you explicitly transfer it to a different region. For example, if you store a file in the region US East, the file will never leave the region US East unless you move the file manually or use cross-region replication and move the file to another region, say, US West. As of this writing, Amazon S3 supports the following regions:

•   The US East (N. Virginia) region uses Amazon S3 servers in Northern Virginia.

•   The US East (Ohio) region uses Amazon S3 servers in Columbus, Ohio.

•   The US West (N. California) region uses Amazon S3 servers in Northern California.

•   The US West (Oregon) region uses Amazon S3 servers in Oregon.

•   The Canada (Central) region uses Amazon S3 servers in Canada.

•   The Asia Pacific (Mumbai) region uses Amazon S3 servers in Mumbai.

•   The Asia Pacific (Seoul) region uses Amazon S3 servers in Seoul.

•   The Asia Pacific (Singapore) region uses Amazon S3 servers in Singapore.

•   The Asia Pacific (Sydney) region uses Amazon S3 servers in Sydney.

•   The Asia Pacific (Tokyo) region uses Amazon S3 servers in Tokyo.

•   The EU (Frankfurt) region uses Amazon S3 servers in Frankfurt.

•   The EU (Ireland) region uses Amazon S3 servers in Ireland.

•   The EU (London) region uses Amazon S3 servers in London.

•   The EU (Paris) region uses Amazon S3 servers in Paris.

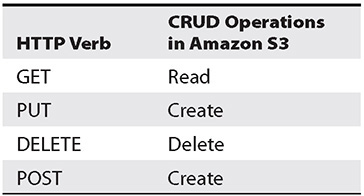
•   The EU (Stockholm) region uses Amazon S3 servers in Stockholm.

•   The South America (São Paulo) region uses Amazon S3 servers in São Paulo.

S3 is accessible from an application programming interface (API), which allows developers to write applications on top of S3. The fundamental interface for S3 is a Representational State Transfer (REST) API. Although S3 supports the Simple Object Access Protocol (SOAP) in HTTPS mode only, SOAP support over Hypertext Transfer Protocol (HTTP) is deprecated. It is recommended that you use REST over SOAP since new Amazon S3 features will not be supported for SOAP.

REST APIs rely on a stateless, client-server, cacheable communications protocol, and in virtually all cases, the HTTP protocol is used. Using a REST API, you should be able to perform all kinds of operations in an S3 bucket, including create, read, update, delete, and list. The REST API allows you to perform all operations in an S3 bucket with standard HTTP/HTTPS requests. Since HTTPS is more secure than HTTP, whenever using an API request with S3, always prefer HTTPS over HTTP to make sure that your request and data are secure.

The primary or most commonly used HTTP verbs (or *methods,* as they are properly called) are POST, GET, PUT, PATCH, and DELETE. These correspond to create, read, update, and delete (CRUD) operations, respectively. You can use these HTTP verbs with the corresponding actions in S3, as shown in [Table 2-1](https://learning.oreilly.com/library/view/aws-certified-solutions/9781260470192/ch2.xhtml#tab2_1).



**Table 2-1** HTTP Verbs and Their Corresponding Actions in Amazon S3 Using the REST API

In addition to APIs, lots of SDKs are available in various platforms, including in browsers, on mobile devices (Android and iOS), and in multiple programming languages (Java, .NET, Node.js, PHP, Python, Ruby, Go, C++). A mobile SDK and IoT device SDK are also available. You can use these SDKs with the combination APIs as per your programming language to simplify using AWS services in your applications.

The AWS command-line interface (CLI) is a unified tool to manage all your AWS services. Using the AWS CLI, you can control multiple AWS services from the command line and automate them through scripts. The AWS CLI is often used by Amazon S3 in conjunction with REST APIs and SDKs. The CLI gives you the ability to perform all the S3 operations from a command line. The AWS CLI can be invoked by using aws from the command line. (Of course, you have to install and configure the AWS CLI before you can start using it.) S3 operations via the AWS CLI can be performed by using the aws s3 command. For example, if you want to create a bucket, the command is aws s3 mb. Similarly, to remove a bucket, you can use aws s3 rb.

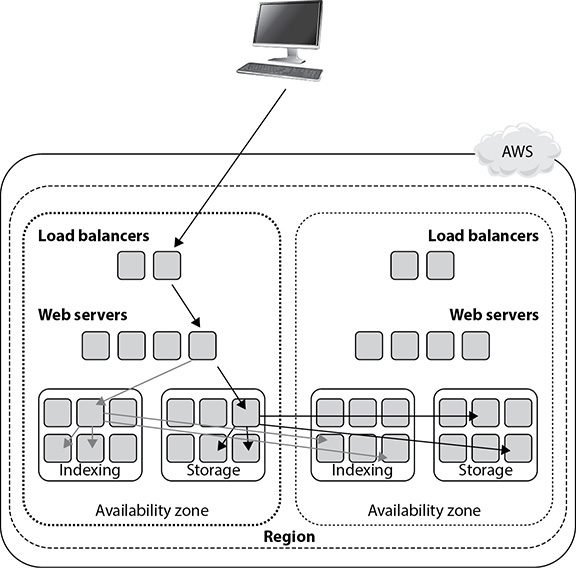
Image

The AWS CLI can be installed on Linux, Windows, Docker platform, and macOS. You should download and install the latest version of the AWS CLI.

AMAZON S3 DATA CONSISTENCY MODEL

It must be noted that Amazon S3 is a web store and not a file system. The S3 service is intended to be a “write once, read many” use case. Therefore, the architecture is a little bit different from a traditional file system or storage area network (SAN) architecture.

The S3 infrastructure consists of multiple load balancers, web servers, and storage across multiple availability zones. The entire architecture is redundant, and the data is stored in multiple storage locations across multiple availability zones (AZs) to provide durability. [Figure 2-2](https://learning.oreilly.com/library/view/aws-certified-solutions/9781260470192/ch2.xhtml#fig2_2) shows the S3 infrastructure.



**Figure 2-2** Amazon S3 infrastructure

Figure 2-2 shows how a file is written in S3. This example shows only two AZs, whereas in real life there could be more than that. For example Amazon S3 Standard uses a minimum of three AZ’s to store the data. Similarly, there could be multiple load balancers and storage. Now when you write an object, you first connect to one of the load balancers. From there you connect to one of the API endpoints on the web server, and then the data is stored in a redundant fashion in multiple AZs across multiple storages, which makes sure your data is protected. The exception to this is Amazon S3-One Zone Infrequent Access, where the data is stored in a single AZ. Once that is done, indexing will happen, and the indexes are also stored in multiple storage locations across multiple AZs. If for any reason a load balancer goes down or if a web server goes down, the S3 request will choose a different load balancer or web server to process the request. Similarly, if a storage unit goes down or the storage containing the index goes down, the data or the index will be served from a redundant storage unit. If the whole AZ goes down, failover will take place, and therefore the content will be served from a different AZ since the entire system is replicated across multiple AZs. This is the “write once, read many” architecture at work.

Let’s explore the consistency model of S3. Whenever you write a new object, the data will be synchronously stored across multiple facilities before returning success. This provides read-after-write consistency.

For all other objects (apart from new ones), S3 is an eventually consistent system. In an eventually consistent system, the data is automatically replicated and propagated across multiple systems and across multiple AZs within a region, so sometimes you will have a situation where you won’t be able to see the updates or changes instantly, or if you try to read the data immediately after update, you may not be able to see all the changes. If a PUT request is successful, your data is safely stored, and therefore there is no need to worry about the data since after a while you will be able to see it. Here are some examples of this:

•   A process writes a new object to Amazon S3 and immediately attempts to read it. Until the change is fully propagated, Amazon S3 might report “key does not exist.”

•   A process writes a new object to Amazon S3 and immediately lists keys within its bucket. Until the change is fully propagated, the object might not appear in the list.

•   A process replaces an existing object and immediately attempts to read it. Until the change is fully propagated, Amazon S3 might return the prior data.

•   A process deletes an existing object and immediately attempts to read it. Until the deletion is fully propagated, Amazon S3 might return the deleted data.

•   A process deletes an existing object and immediately lists keys within its bucket. Until the deletion is fully propagated, Amazon S3 might list the deleted object.

In the case of an update, updates to a single key are atomic. For example, if you PUT to an existing key, a subsequent read might return the old data or the updated data, but it will never write corrupted or partial data. Also, it should be noted that Amazon S3 does not support object locking, which means if there are requests to update the same file concurrently (PUT request), the request with the latest time stamp wins. Though this functionality is not available in S3, you can achieve this by building an object-locking mechanism into your application.

Data lag may comprise a few seconds, not minutes or hours. So it is recommended that you have your applications retry or use other strategies such as leveraging DynamoDB as an application-level metadata store to ensure that your application is accessing all the current information from S3. For example, if you’re building a photo app and you need to display all the photos uploaded by a user, you’ll want to maintain the list of photos in DynamoDB and store the photos themselves in S3. That way, when you need to display the user’s photos, your app will first get the names of all files from DynamoDB and retrieve them from S3.

AMAZON S3 PERFORMANCE CONSIDERATIONS

It is important to understand the best practice for partitioning if the workload you are planning to run on an Amazon S3 bucket is going to exceed 100 PUT/LIST/DELETE requests per second or 300 GET requests per second. In this case, you need to make sure you follow the partitioning guidelines so you don’t end up with any performance bottleneck. You want to able to provide users with a better experience and be able to scale as you grow. Amazon S3 scales to support very high request rates. To do so, internally S3 automatically partitions all your buckets.

As you studied earlier, the name of an S3 bucket is unique, and by combining the bucket name and object name (key), every object can be identified uniquely across the globe. Moreover, the object key is unique within a bucket, and the object keys are stored in UTF-8 binary, with a maximum size of 1,024 bytes. Let’s say you have a bucket named awsbook and you have the image image2.1.jpg in it in the folder chapter2/image. In this case, this is how it will look:

Image

Please note that the object key is chapter2/image/image2.1.jpg and not just the name of the file, which is image2.1.jpg. S3 automatically partitions based on the key prefix.

Say you have 20 objects in the bucket awsbook, as shown next. In this scenario, S3 is going to do the partitioning based on the key prefix. In this case, it is c, the first word of the key chapter, which is shown in bold in the following example:



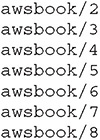


In this case, everything falls under the same partition, here awsbook/c, since the partitioning key is c. Now imagine a scenario where you have millions of objects. The performance of the objects in this bucket is definitely going to take a hit. In this scenario, a better way would be to partition the objects with a different key so that you get better performance.

To address this problem, you can change the chapter name to start with a number such as 2chapter, 3chapter, 4chapter, and so on, as shown next. What you have done here is to simply change one character, which changes the partitioning strategy.



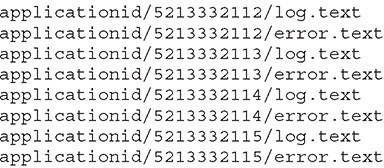
By doing this, you have distributed your objects into the following partitions instead of just one partition:



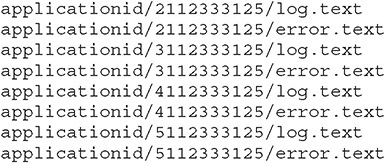
A few other ways of implementing the partitioning for performance would be to reverse the key name string and to add a hex hash prefix to the key name. Let’s see an example of each approach.

REVERSE THE KEY NAME STRING

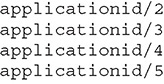
Say you are doing massive uploads from your application, and with every set of uploads, the sequence of the application IDs increases by 1. This is a common scenario at many organizations.



In this case, since the application ID starts with 5, everything will fall under the same partition, which is applicationid/5. Now you are simply going to reverse the key to solve the partitioning issue.

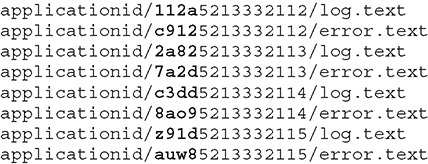


By simply reversing the keys, S3 will create multiple partitions in this case, thereby improving the performance.



ADDING A HEX HASH PREFIX TO A KEY NAME

In many scenarios, you will notice that by reversing the key or by changing a few characters, you will not get an optimal partitioning strategy. In that case, you can add a hash string as a prefix to the key name to introduce some randomness. You can compute an MD5 hash of the character sequence, pick a specific number of characters, and assign them as the prefix to the key name. For example, instead of reversing the keys this time, you will use a hex hash prefix. The hash prefix is shown in bold.



If you are planning to use a hash key, you may want to do it carefully because of randomness in the algorithm. If you have too many objects, you will end up with too many partition keys. For example, if you use a 4-character hex hash, there are 65,536 possible character combinations, so you will be sending 65,536 list bucket requests with each specific prefix, which is a combination of a four-digit hash and the date. In fact, you don’t need more than two or three prefix characters in your hash. Say you are targeting 100 operations per second and have 25 million objects stored per partition; a 4-character hex has a partition that could support millions of operations per second. This is a pretty big number, and in most cases you probably do not need to support this big of a number.

ENCRYPTION IN AMAZON S3

Let’s talk about the encryption features of Amazon S3. There are two main ways of securing the data: encryption of data in transit and encryption of data at rest. Encryption of data in transit means securing the data when it is moving from one point to other, and encryption of data at rest means securing the data or the objects in S3 buckets when the objects remain idle in the S3 bucket and there is no activity going on with them.

If you upload the data using HTTPS and use SSL-encrypted endpoints, the data is automatically secured for all the uploads and downloads, and the data remains encrypted during transit.

If the data is encrypted before being uploaded to an S3 bucket using an S3-encrypted client, the data will already be encrypted in transit. An Amazon S3 encryption client is used to perform client-side encryption for storing data securely in S3. Data encryption is done using a one-time randomly generated content encryption key (CEK) per S3 object. The encryption materials specified in the constructor will be used to protect the CEK, which is then stored alongside the S3 object. You can obtain the Amazon S3 encryption client from <http://docs.aws.amazon.com/AWSJavaSDK/latest/javadoc/com/amazonaws/services/s3/AmazonS3EncryptionClient.html>.

Let’s talk about securing the data at rest.

With Amazon S3 Server Side Encryption (SSE), Amazon S3 will automatically encrypt your data on write and decrypt your data on retrieval. This uses Advanced Encryption Standard (AES) 256-bit symmetric keys, and there are three different ways to manage those keys.

•   **SSE with Amazon S3 Key Management (SSE-SE)** In this case, Amazon S3 will encrypt your data at rest and manage the encryption keys for you. Each object is encrypted using a per-object key. The per-object key is encrypted using a master key, and the master key is managed using S3 key management. The master key is rotated on a monthly basis. You can turn on that option from the S3 console or from the command line or via the SDK, and you don’t have to do anything other than that for key management. Everything else is taken care of by Amazon.

•   **SSE with customer-provided keys (SSE-C)** With SSE-C, Amazon S3 will encrypt your data at rest using the custom encryption keys that you provide. To use SSE-C, simply include your custom encryption key in your upload request, and Amazon S3 encrypts the object using that key and securely stores the encrypted data at rest. Similarly, to retrieve an encrypted object, provide your custom encryption key, and Amazon S3 decrypts the object as part of the retrieval. Amazon S3 doesn’t store your encryption key anywhere; the key is immediately discarded after Amazon S3 completes your requests.

•   **SSE with AWS Key Management Service KMS (SSE-KMS)** With SSE-KMS, Amazon S3 will encrypt your data at rest using keys that you manage in AWS KMS. Using AWS KMS for key management provides several benefits. With AWS KMS, there are separate permissions for the use of the master key, providing an additional layer of control as well as protection against unauthorized access to your object stored in Amazon S3. AWS KMS provides an audit trail so you can see who used your key to access which object and when, as well as view failed attempts to access data from users without permission to decrypt the data. Additionally, AWS KMS provides additional security controls to support customer efforts to comply with PCI-DSS, HIPAA/HITECH, and FedRAMP industry requirements.

AMAZON S3 ACCESS CONTROL

Access control defines who is going to have what access in an S3 bucket. There are several ways to provide access to an S3 bucket, and access control is all about managing the access. You can define various rules specifying who can access which aspects of your S3 service, and by doing that you have total control over S3. In addition, by using access control, you have very fine manual control over the objects stored in S3. You will see a few examples in this chapter, but first let’s discuss all the ways of managing the access control in an S3 bucket. There are three main ways of using access control.

Access Policies

By creating an Identity and Access Management policy, you can provide fine-grained control over objects in S3 because an IAM policy helps you control who can access your data stored in S3. You can create an IAM policy and assign that policy to either a user, a group, or a role. You will explore more about users, groups, and roles in [Chapter 5](https://learning.oreilly.com/library/view/aws-certified-solutions/9781260470192/ch5.xhtml). Say you create an S3 full access policy and assign it to a particular group that has ten members. Now all ten users of that particular group will have full access to the S3 bucket. There are a lot of other things that can be done using IAM and S3; for example, you can choose which S3 bucket can be shared with which IAM user, you can allow a particular user to access a particular bucket, you can allow a specific user or all the users to read objects from a specific bucket or from a few buckets, and you can have a policy to allow your customers or partners to drop objects in a particular bucket.

Let’s see a few examples of what these policies look like. It is assumed you have a working knowledge of JavaScript Object Notation (JSON). Before you can start writing a policy, you need to know what an Amazon resource name (ARN) is. ARNs uniquely identify AWS resources. An ARN is needed to specify a resource unambiguously across all of AWS, such as in IAM policies, API calls, and so on. This is what an ARN looks like:

Image

partition is the partition that the resource is in. For standard AWS regions, the partition is aws. If you have resources in other partitions, the partition is aws-partitionname. For example, the partition for resources in the China (Beijing) region is aws-cn.

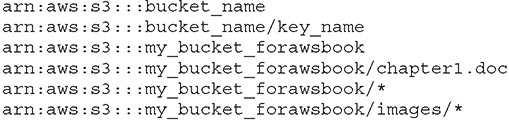
service is the service namespace that identifies the AWS product (for example, Amazon S3, IAM, or Amazon RDS). For Amazon S3, the service will be s3.

region is the region the resource resides in. This is optional; some ARNs do not require a region, so you can omit it in that case.

account is the ID of the AWS account that owns the resource, without the hyphens. For example, it can be 331983991. This is optional; some ARNs do not require an account.

resource, resourcetype:resource, or resourcetype/resource is the content; this part of the ARN varies by service. It often includes an indicator of the type of resource—for example, an IAM user or Amazon RDS database—followed by a slash (/) or a colon (:), followed by the resource name itself.

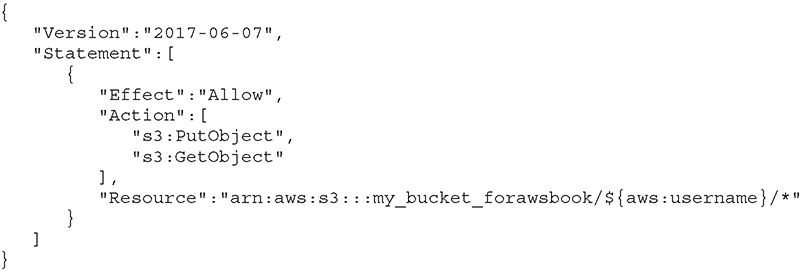
Let’s see a few examples of how an S3 ARN looks. Please note that S3 does not require an account number or region in ARNs.





**TIP**    You can also use a policy variable to make things simpler. For example, instead of manually putting in the username, you can use the variable ${aws:username}, so when the policy is executed, the username is replaced with the actual username. This is helpful when you want to assign a policy to a group.

Now let’s see an example of how a policy looks. Say you want to provide put and get permission to the S3 bucket my\_bucket\_forawsbook. The policy looks something similar to this:

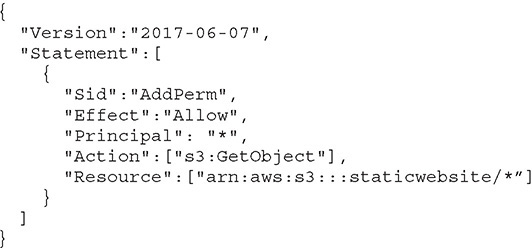


In this example, Effect is the effect of the policy, Allow or Deny. Action lists the actions allowed or denied by this policy. Resource is the AWS resource that this policy applies to.

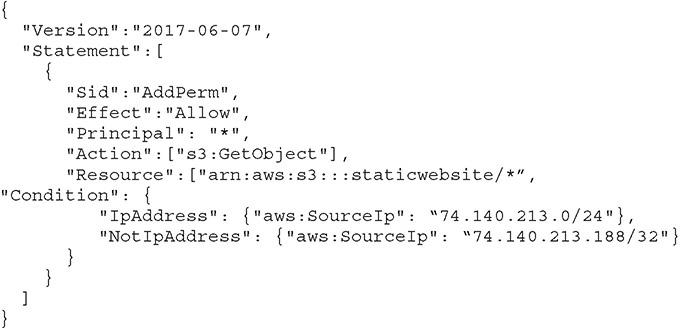
In Lab 2-3, you will generate a bucket policy using the AWS Policy Generator (<https://awspolicygen.s3.amazonaws.com/policygen.html>).

Bucket Policies

You can also create policies at the bucket level, which is called a *bucket policy*. Bucket policies also allow fine-grained control over S3 buckets. Using a bucket policy, you can incorporate user restrictions without using IAM. You can even grant other AWS accounts or IAM user permissions for buckets or any folders or objects inside it. When you create a bucket policy, any object permissions apply only to the objects that the bucket owner creates. A very common use case of a bucket policy would be that you can grant read-only permission to an anonymous user to access any object from a particular bucket. This is useful if you are planning to host a static web site in S3 and want everyone to be able to access the web site; you simply grant GetObject access. In the following example, you are giving access to all users for the S3 bucket staticwebsite. You will learn more about static web site hosting in S3 later in this chapter.



In the previous example, by specifying the principal with a wildcard (\*), the policy grants anonymous access. You can even do some fine-grained control using a bucket policy. For example, if for your S3 bucket you want to allow access from only from a particular region and want to deny access from another region, you can do this by specifying a condition to include and not include an IP address range.



A few other real-life uses of bucket policies would be granting permissions to multiple accounts with more conditions, making MFA authentication mandatory by using a policy, granting permission only when the origin is Amazon CloudFront, restricting access to a specific HTTP referrer just like you restricted access for a particular IP access in the previous example, and granting cross-account permissions to upload objects while ensuring the bucket owner has full control.

Access Control List

The third option is to use an access control list (ACL). Each bucket and object inside the bucket has an ACL associated with it. ACLs apply access control rules at the bucket or object level in S3. Unlike an IAM policy or bucket policy, an ACL does not allow fine-grained control. Rather, it allows coarse-grained control. An ACL is a list of grants identifying the grantee and permission granted. For example, you can grant permissions such as basic read-write permissions only to other AWS accounts, not users in your account.

Please note that bucket policies and access control lists are called *resource-based policies* since you attach them to your Amazon S3 resources.

S3 Security Best Practices

Because Amazon S3 contains all sorts of data, it is important that the data is secure. Following are some of the practices that will ensure data security:

•   *Ensure that S3 buckets are not publicly accessible.* If a bucket is publicly accessible, anyone can access the file. Use the Amazon S3 Block Public Access feature to limit the public access to S3 buckets.

•   *Use least privilege access.* Ensure that only those who need permission to access S3 buckets are given access. Those who don’t need to access S3 buckets should be denied access. By default, the permission should be denied unless someone needs it.

•   *Enable multifactor authentication (MFA).* You can set up MFA Delete to ensure that anyone who is not authorized to delete a file cannot delete it.

•   *Audit the S3 bucket.* Make sure you audit the S3 buckets regularly. This will help in identifying the security gaps. You can also look at using AWS Trusted Advisor to audit bucket permissions (see [Chapter 9](https://learning.oreilly.com/library/view/aws-certified-solutions/9781260470192/ch9.xhtml)).

•   *Use IAM roles.* Use IAM roles to manage credentials. Roles are covered in detail in [Chapter 5](https://learning.oreilly.com/library/view/aws-certified-solutions/9781260470192/ch5.xhtml).

AMAZON S3 STORAGE CLASS

Amazon S3 offers a variety of storage class designs to cater to different use cases. Depending on your use case and needs, you can choose the appropriate storage class in an S3 bucket to store the data. You can also move the files from one storage class to another storage class. You can also configure lifecycle policies and rules to automatically move files from one storage class to other. Once you create a policy and enable it, the files will be moved from one storage class to the other automatically.

These are storage classes offered by Amazon S3.

•   **Amazon S3 Standard** Amazon S3 Standard is the default storage. It offers high durability, availability, and performance and is used for frequently accessed data. This is the most common use case of S3 and is used for a variety of purposes, including web sites, content storage, big data analytics, mobile applications, and so on. The S3 Standard is designed for durability of 99.999999999 percent (11 nines) of objects and 99.99 percent availability over a given year and comes with the S3 SLA for availability. It supports SSL encryption of data in transit and at rest, data lifecycle policies, cross-region replication, and event notifications. The files in Amazon S3 Standard are synchronously copied across three facilities and designed to sustain the loss of data in two facilities.

•   **Amazon S3 Standard Infrequent Access (IA)** IA is an Amazon S3 storage class that is often used for storing data that is accessed less frequently. It provides the same durability over a given year (99.999999999 percent) and inherits all the S3 features, including concurrent facility fault tolerance, SSL encryption of data in transit and at rest, data lifecycle policies, cross-region replication, and event notifications. This storage class provides 99.9 percent availability over a given year. The price of this storage class is much cheaper than Amazon S3 Standard, which makes it economical for long-term storage, backups, and disaster recovery use cases. Using lifecycle policies, you can move the files from Amazon S3 Standard to IA.



**TIP**    The storage class is simply one of several attributes associated with each Amazon S3 object. The objects stay in the same S3 bucket (Standard versus IA) and are accessed from the same URLs when they transition from Standard to IA, and vice versa. There is no need to change the application code or point to a different URL when you move the files from one storage class to the other.

•   **Amazon S3 One Zone-Infrequent Access (S3 One Zone-IA)** Amazon S3 One Zone-IA is a new storage class for storing data that is accessed less frequently but requires rapid access when needed. Amazon S3 One Zone-IA stores data in a single AZ and not like S3 Standard where the data is stored in a minimum of three availability zones (AZs). S3 One Zone-IA offers the same high durability, high throughput, and low latency of Amazon S3 Standard and S3 Standard-IA but it costs 20 percent less than storing it in S3 Standard-IA. The S3 One Zone-IA storage class is set at the object level and can exist in the same bucket as S3 Standard and S3 Standard-IA, allowing you to use S3 Lifecycle Policies to automatically transition objects between storage classes without any application changes.



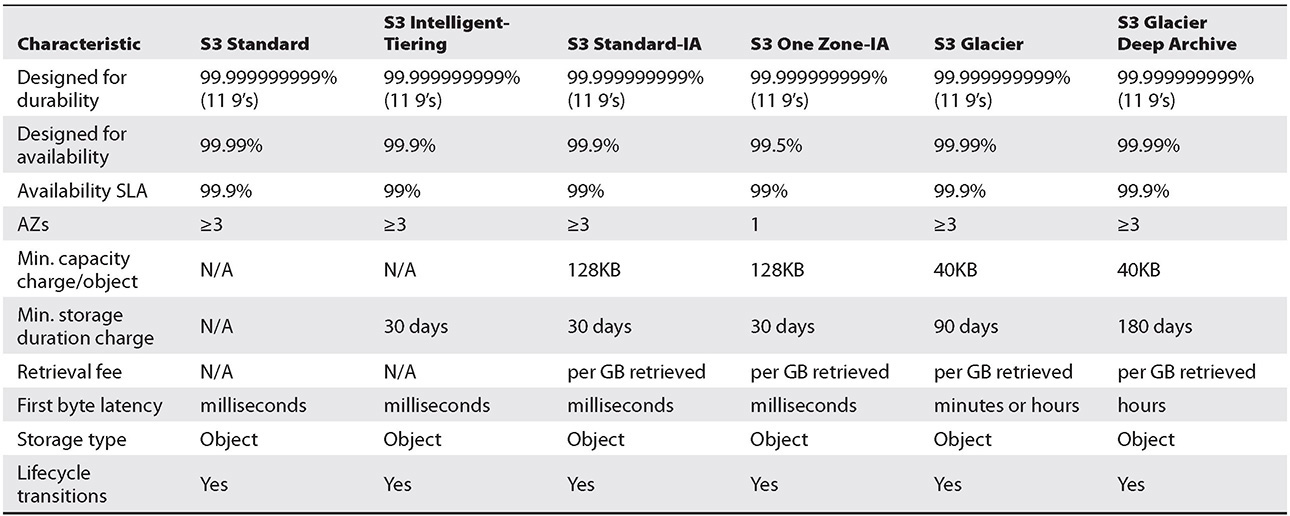
**NOTE**    Amazon keeps reducing the prices of its services. With the latest price cut, the cost of Amazon S3 RRS is now almost similar to Amazon S3 Standard. Therefore, instead of storing files in Amazon S3 RRS, you may want to store them in Amazon IA, which is much cheaper than Amazon S3 RRS.

•   **Amazon S3 Intelligent-Tiering** Amazon S3 Intelligent-Tiering is designed to optimize costs by automatically moving data from one storage class to another based on the access pattern. With this feature, Amazon S3 monitors the access pattern and moves the objects from one storage class to another automatically. Let’s say, for example, that an object is not accessed for more than 30 days. S3 Intelligent-Tiering automatically moves the file to a lower tier. Once the file is accessed again, it is moved back to the higher tier.

•   **Amazon S3 Glacier** Amazon S3 Glacier is the storage class mainly used for data archiving. This also provides 99.999999999 percent durability of objects and supports SSL encryption of data in transit and at rest, and so on. Since it is mainly used for archiving the data, there are three main options for retrieving data with varying access times and cost. These are expedited, standard, and bulk retrievals. The expedited retrievals allow for very quick retrieval of data in the range of one to five minutes. Standard takes about three to five hours to retrieve the data, and bulk enables you to retrieve large amounts of data such as petabytes of data in a day (five to twelve hours). Amazon S3 Glacier is much cheaper than all the other storage classes. You will study Amazon S3 Glacier in more detail later in this chapter.

•   **Amazon S3 Glacier Deep Archive** Amazon S3 Glacier Deep Archive is used for long-term archival storage. Suppose, for example, that you have a compliance requirement to retain all data for seven to ten years. You may be accessing this data only once or twice a year. Glacier Deep Archive is the ideal solution, a replacement for magnetic tape drives. The S3 Glacier Deep Archive also provides 99.999999999 percent durability, because the data is stored across three different AZs. With S3 Glacier Deep Archive, data can be restored within 12 hours.

[Table 2-2](https://learning.oreilly.com/library/view/aws-certified-solutions/9781260470192/ch2.xhtml#tab2_2) shows the performance characteristics across all the different storage classes of Amazon S3.



**Table 2-2** Performance Characteristics Across S3 Storage Class

You can move the files from one storage class to another in a couple of ways:

•   By creating a lifecycle policy

•   By running the S3 copy (aws s3 cp) command from the AWS CLI

•   From the Amazon S3 console

•   From an SDK

Versioning of Objects in Amazon S3

Versioning allows you to maintain multiple versions of the same file. Say you have enabled versioning and then upload ten different versions of the same file. All those files will be stored in an S3 bucket, and every file will be assigned a unique version number. When you look at the S3 bucket, you will see only one file name and won’t see all ten versions since all ten different versions of the file have the same name. But behind the scenes, S3 stores all the different versions, and with the version option, you can actually look at and download each version of the file. This is really helpful if you accidentally delete a file or if you accidentally update a file and want to go back to a previous version. Versioning is like an insurance policy; you know that regardless of what happens, your files are safe. With versioning, not only can you preserve the files and retrieve them, but you also can restore every version. Amazon S3 starts preserving the existing files in a bucket anytime you perform a PUT, POST, COPY, or DELETE operation on them with versioning enabled.

By default, GET requests will retrieve the most recently written version, or if you download a file directly from the console by default, it will show you only the latest version of the file. If you want to retrieve the older version of a file, you need to specify the version of the file in the GET request.

You can use versioning in conjunction with lifecycle rules to move the old versions of the files to a different class of storage. For example, you can write a rule to move all the old version files to Glacier after 30 days and then have them deleted from Glacier after an additional 30 days. Now you have a total of 60 days to roll back any change. Once you enable versioning, you can’t disable it. However, you can suspend versioning to stop the versioning of objects.



**TIP**    Versioning is always done at the bucket level. Say you have thousands of objects in an Amazon S3 bucket and you want to enable versioning for only a few files. In that case, create a separate bucket with only the files for which you want to enable versioning. This way you will avoid versioning thousands of files from your previous S3 bucket.

Amazon S3 Object Lifecycle Management

Using lifecycle management, you can get the best out of S3 in terms of performance as well as cost. You can create various policies according to your needs and can move the files that are stored in Amazon S3 from one storage class to another. You can perform two main kinds of actions with a lifecycle policy.

•   **Transition action** This means you can define when the objects can be transitioned to another storage class. For example, you may want to copy all older log files after seven days to S3-IA.

•   **Expiration action** In this case, you define what is going to happen when the objects expire. For example, if you delete a file from S3, what are you going to do with that file?

There are several reasons why you need to have a lifecycle policy in place. For example, you may have a compliance requirement to store all the data that is older than seven years in an archival storage, you may have a rule to archive all the financial and healthcare records, or you may want to create an archive with all the database backups that must be retained for a compliance reason for *n* number of years. You can also create policies to move the files from one storage class to another. For example, after a week, you could move all the log files from Amazon S3 to Amazon S3-IA; then after a month, move all the log files from Amazon S3-IA to Amazon S3 Glacier; and finally after a year, move all the log files from Amazon S3 Glacier to Amazon S3 Glacier Deep Archive.



**TIP**    Lifecycle rules are attached to a bucket. If you want to apply the rules to a few files, prefix the files with a unique prefix and then enable the lifecycle rule on the prefix.

Amazon S3 Replication

S3 replication enables you to copy all the files from your S3 bucket to another bucket automatically. Then, whenever a new file is added to the bucket, it is automatically replicated to the other bucket. Replication can be done from one account to another account, from one storage class to another, and from one region to another or within the same region. S3 replication happens behind the scenes, and the data is copied in an asynchronous fashion. To set up S3 replication, you need the details of the source bucket and the target bucket and you should have the role permissions to perform the replication.

AWS offers two types of S3 replication:

•   **Cross-region replication (CRR)** This is used to copy the objects across different regions.

•   **Same-region replication (SRR)** This is used to copy the files within the same region.

CROSS-REGION REPLICATION (CRR) AND SAME-REGION REPLICATION (SRR)

Many organizations need to keep copies of critical data in locations that are hundreds of miles apart. This is often because of a mandatory compliance requirement and disaster recovery considerations. To do this, you can make copies of your S3 bucket in a different AWS region. Using CRR, you can automatically copy all the files from one region to another. Every object uploaded to a particular S3 bucket can be automatically replicated to the destination bucket, which is located in a different AWS region. CRR allows asynchronous copying of objects. This helps to minimize latency if you have users from different parts of the world. You can set up S3 CRR to copy the data to a region near to the users to provide a better user experience.

By using SRR, you can automatically replicate the objects residing in an S3 bucket to a different bucket in the same region. This feature is very helpful if you need a copy of the data, if want to set up a replication from production to test or development environments, if you want to store the files under multiple accounts, or if you want to consolidate objects from multiple buckets into one bucket. SRR is a very useful feature, especially when you’re building a data lake, because you want to keep a golden copy of your log files and always use a copy of the log files on which to run the analytics.

As an example, say you have a bucket called jbawsbook and you want to enable CRR on it. Before you can set up CRR, you need to ensure a few things:

•   The versioning must be enabled in the source and target buckets. If you don’t enable versioning, you won’t be able to do the CRR.

•   In CRR, the source bucket owner must have the target region enabled where the replication needs to be set up. This doesn’t apply if the replication is being set up within the same region (SRR).

•   Amazon S3 must have the permission to replicate the files.

•   In some cases, the bucket owner may not own the objects. In those cases, the object owner must grant READ and READ\_ACP permission to the objects using an object access control list (ACL).

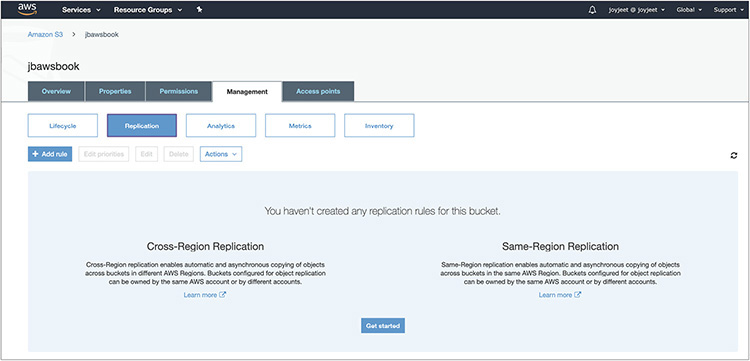
•   If the source bucket has Object Lock enabled, the destination bucket should also have Object Lock enabled. With S3 Object Lock, you can store objects using a write once read many (WORM) model. You can use this to prevent an object from being deleted or overwritten for a fixed amount of time, or indefinitely.

•   If your source and target buckets are owned by different accounts, the target owner must grant permission to the source bucket to replicate the objects.

Let’s see how to set up replication from the console.

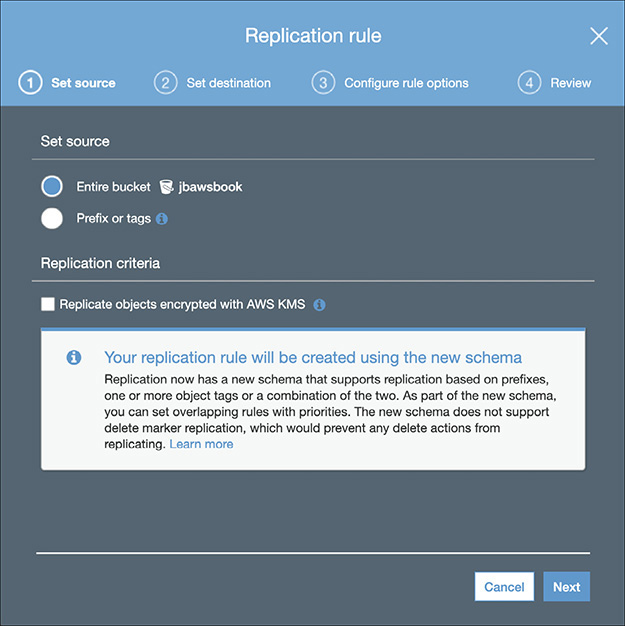
**1.**   Go to the console and select the S3 bucket.

**2.**   Then open the Management tab and click Replication. You will notice that you can configure CRR and SRR from the same screen. Click the Get Started button, as shown in [Figure 2-3](https://learning.oreilly.com/library/view/aws-certified-solutions/9781260470192/ch2.xhtml#fig2_3).



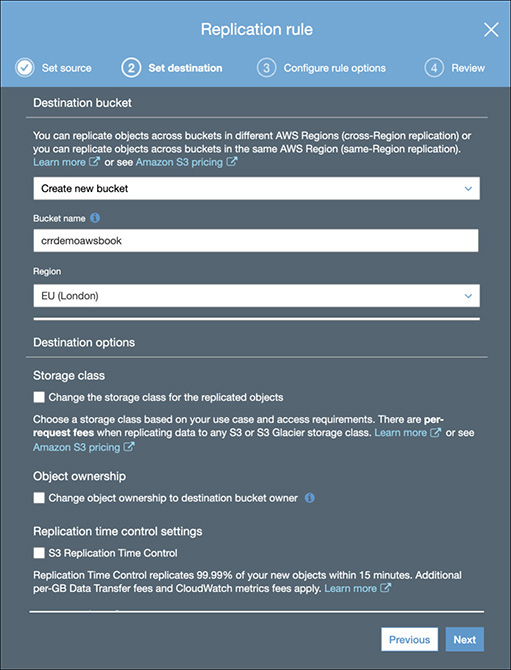
**Figure 2-3** Getting started

**3.**   Then in the next screen, as shown in [Figure 2-4](https://learning.oreilly.com/library/view/aws-certified-solutions/9781260470192/ch2.xhtml#fig2_4), the Set Source screen, you’ll see two options: Whole Bucket, which means that all the objects of the bucket will be replicated, and Prefix Or Tags, which means you can restrict replication to a subset of the objects in the bucket using a prefix. If you want to replicate the objects that are encrypted with AWS KMS, select Replicate Objects Encrypted With AWS KMS.



**Figure 2-4** Setting the source in the Replication Rule screen

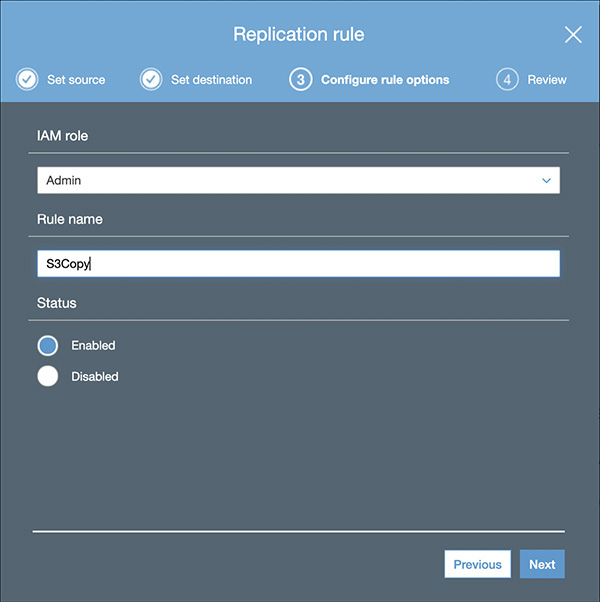
**4.**   Create or choose the destination bucket where you want to replicate the files, as shown in [Figure 2-5](https://learning.oreilly.com/library/view/aws-certified-solutions/9781260470192/ch2.xhtml#fig2_5). You can choose the destination bucket in same region or in a different region. You also have the option of selecting a different class of storage for replicating the files. By default, the files are replicated to the same storage class, but from the console you can choose one of the following storage classes: Standard, Intelligent-Tiering, Standard-IA, One Zone-IA, or Glacier and Glacier Deep Archive. The bucket can be either in this account or in another account. Select Object Ownership to change the object ownership. To do this, you must provide the account ID and bucket name.



**Figure 2-5** Choosing a destination bucket and setting time control

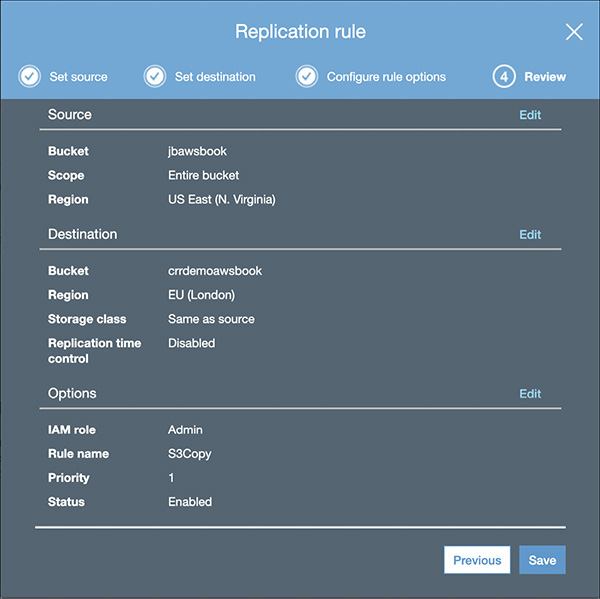
**5.**   The last option is S3 Replication Time Control. Using this feature, you can replicate your data in a predictable time frame. S3 Replication Time Control replicates 99.99 percent of new objects stored in Amazon S3 within 15 minutes. Select this option if you want to enable S3 Replication Time Control. Then click Next.

**6.**   Choose an IAM role and Rule Name so that S3 can list and retrieve objects from the source bucket and replicate them to the destination bucket. Using roles, you can also replicate objects owned by separate AWS accounts. Select the IAM role and rule name, as shown in [Figure 2-6](https://learning.oreilly.com/library/view/aws-certified-solutions/9781260470192/ch2.xhtml#fig2_6). (We will explore IAM in detail in [Chapter 5](https://learning.oreilly.com/library/view/aws-certified-solutions/9781260470192/ch5.xhtml).) Then click Next.



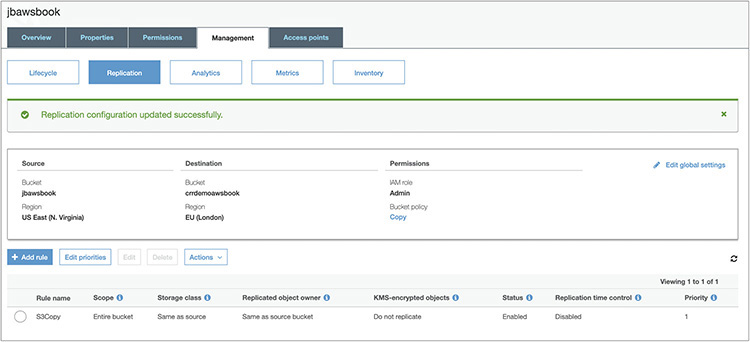
**Figure 2-6** Configuring IAM details

**7.**   In the next screen, shown in [Figure 2-7](https://learning.oreilly.com/library/view/aws-certified-solutions/9781260470192/ch2.xhtml#fig2_7), you can review all the options you have configured. If things look good, click Save. Or click Edit to make the changes.



**Figure 2-7** Review your configuration

**8.**   If you clicked Save, a confirmation screen, shown in [Figure 2-8](https://learning.oreilly.com/library/view/aws-certified-solutions/9781260470192/ch2.xhtml#fig2_8), informs you that your replication configuration updated successfully. The details of your replication are also displayed here.



**Figure 2-8** Confirmation screen



**NOTE**    The replication copies only the new objects. If you have preexisting files in the bucket, you must copy them manually from the source to the destination using an S3 copy or via the CLI or SDK.

If you want to monitor the replication status, you can do so by using the HEAD operation on a source object. The HEAD operation retrieves metadata from an object without returning the object itself. This operation is useful if you are interested only in an object’s metadata. To use HEAD, you must have read access to the object. Covering the HEAD operation in more detail is beyond the certification objective; however, if you want to learn more about it, refer to the Amazon S3 API guide.

STATIC WEB SITE HOSTING IN AMAZON S3

You can host a static web site on Amazon S3. A static web site is one where the content does not change and remains static. A static web site may contain some client-side scripts, but the content of the web site is almost stagnant all the time. By contrast, a dynamic web site is one where the content changes frequently, and a lot of server-side processing happens by running scripts built on PHP, JSP, and so on. Amazon S3 does not support server-side scripting.

It is easy to host a static web site on Amazon S3. Using the following steps, you should be able to host your static web site on Amazon S3 in a few minutes:

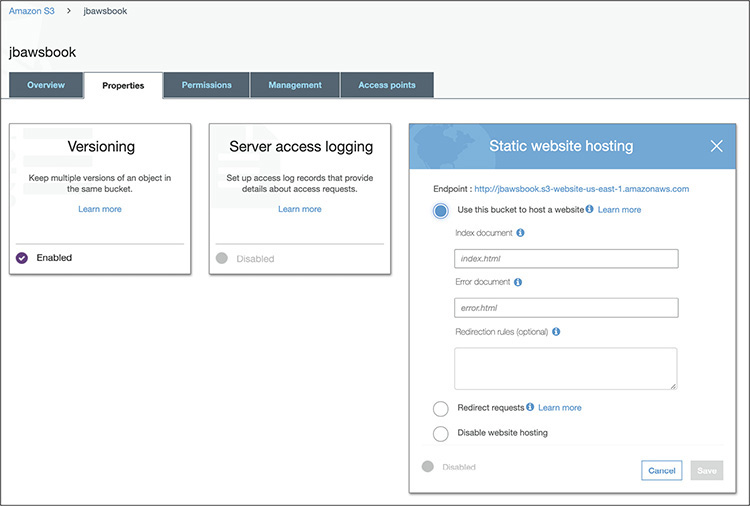
**1.**   Sign in to the AWS management console and open the Amazon S3 console at <https://console.aws.amazon.com/s3/>.

**2.**   In the Bucket Name list, choose the name of the bucket that you want to enable static web site hosting for.

**3.**   Choose Properties.

**4.**   Choose Static Web Site Hosting. After you enable your bucket for static web site hosting, web browsers can access all of your content through the Amazon S3 web site endpoint for your bucket.

**5.**   Choose Use This Bucket To Host A Website, as shown in [Figure 2-9](https://learning.oreilly.com/library/view/aws-certified-solutions/9781260470192/ch2.xhtml#fig2_9). For Index Document, type the name of the index document, which is typically index.html. When you configure a bucket for web site hosting, you must specify an index document. Amazon S3 returns this index document when requests are made to the root domain or any of the subfolders. When you’re done, click Save.



**Figure 2-9** Choosing static website hosting

**6.**   Add a bucket policy to the web site bucket that grants everyone access to the objects in the bucket. When you configure a bucket as a web site, you must make the objects that you want to serve publicly readable. To do so, you write a bucket policy that grants everyone s3:GetObject permission. You will create this bucket policy as part of Lab 2-3.

AMAZON S3 GLACIER

Amazon S3 Glacier is a low-cost cloud storage service that is mainly used for data archiving and long-term backup. Just like S3, Amazon S3 Glacier is extremely secure and durable and provides the same security and durability as S3. The best part about Amazon S3 Glacier is it is extremely low in cost. You can store data in Amazon S3 Glacier for as little as $1 per terabyte per month, which is another reason why it’s becoming popular these days. As discussed previously, Amazon S3 Glacier has two flavors: Amazon S3 Glacier and Amazon S3 Glacier Deep Archive.

Following are the most common use cases of Amazon S3 Glacier in the industry currently:

•   **Magnetic tape replacement** Managing tape archives is a pretty painful activity, and if you have a compliance requirement to archive all old data, you end up maintaining a library of magnetic tape drives. Creating a tape library needs a huge capital investment and of course specialized maintenance. Often customers ship the tape archives to a different location to create a redundant backup. The shipping costs also add to the cost of managing the tape libraries and are often overlooked. Sometimes when restoring the magnetic tape you realize the tape has gone bad and you can’t retrieve the data from it. Amazon S3 Glacier has zero up-front costs, so you don’t have to incur a huge capital expenditure to use it. In addition, there is no maintenance overhead like with magnetic tape, and you get the same durability as S3. Therefore, customers are replacing their table library footprint with Amazon S3 Glacier Deep Archive.

•   **Healthcare/life sciences/scientific data storage** With the advancement in life sciences such as genomic data, a single sequence of genomes can take up to a terabyte of data. Also, if more data is available for running the analytics, the better results scientists will get to run their research. The same applies to scientists in other areas. They need to generate, analyze, and archive all the data, and Glacier is the best place to store all of it. In addition, even hospitals have to meet compliance requirements; they need to keep all the patients’ records, which sometimes can be petabytes of data. Amazon S3 Glacier helps you to achieve this at a low cost.

•   **Media assets archiving/digital preservation** Media assets such as video of news coverage and game coverage can grow to several petabytes quickly. Amazon S3 Glacier is the best place to archive all these media assets. If you need to redistribute/rebroadcast them, you can quickly move them to Amazon S3. Similarly, you can use Amazon S3 Glacier to archive audio, e-books, and so on.

•   **Compliance archiving/long-term backup** Many organizations have a compliance requirement to archive all the data that is *x* years old. Similarly, many organizations have an internal policy to keep a long-term backup of several files, though it may not be a compliance requirement. Amazon S3 Glacier Vault Lock helps you set compliance controls to meet your compliance objectives. You will learn more about Amazon S3 Glacier Vault Lock in the next section.

Amazon S3 Glacier Key Terminology

This section will define some key terminology used in Amazon S3 Glacier.

When you upload data in Amazon S3 Glacier, you store it as an *archive*. Even if you store one file, that will still be an archive with a single file. It is recommended that you aggregate your files and then upload them to Amazon S3 Glacier. You can use popular tools such as TAR or ZIP to do this. By consolidating your data into a single file as an archive, your cost will be much lower than storing single files as separate archives. Like with Amazon S3, there is no limit to how many files you can store in Amazon S3 Glacier. A single Amazon S3 Glacier archive can be anything from 1 byte to 40TB, and there is no limit on how many of these you can upload. Please note that archives are write-once, which means once you create an archive, you won’t be able to modify the files in it. Therefore, the data stored in an archive is immutable, meaning that after an archive is created, it cannot be updated. If you want to update or edit the contents, you need to download them and modify and re-upload them in Amazon S3 Glacier. If you are planning to upload large archives (100MB or bigger), it is recommended that you use the multipart upload capability where the files are broken into smaller chunks and uploaded individually. Once all the smaller chunks are successfully uploaded, they are combined to make a single archive. The max archive size that can be uploaded in a single upload request is 4GB.

Every vault has a unique address. The format looks like the following:

Image

Archives are stored in a *vault*, which is like a safe deposit box or locker. You can think of a vault as being similar to a bucket in S3. You can group multiple archives and put them in a vault. You can say a vault is like a container for an archive. A vault gives you the ability to organize your data residing in Amazon S3 Glacier. You can set different access policies for each vault, thereby allowing different people to access different vaults. You can use IAM and create the vault-level access policies. You can create up to 1,000 vaults per account per region. For the ease of managing your vault, AWS provides you with the ability to tag your vault. If you want to delete a vault, it should not contain any archive. If it contains an archive, then you need to delete the archive first before deleting the vault.

*Amazon S3 Glacier Vault Lock* allows you to easily deploy and enforce compliance controls on individual Amazon S3 Glacier vaults via a lockable policy. You can specify controls such as Write Once Read Many (WORM) in a Vault Lock policy and lock the policy from future edits. Once locked, the policy becomes immutable, and Amazon S3 Glacier will enforce the prescribed controls to help achieve your compliance objectives.

Amazon S3 Glacier maintains a cold index of archives refreshed every 24 hours, which is known as an *inventory* or *vault inventory*.

Whenever you want to retrieve an archive and vault inventory, you need to submit an Amazon S3 Glacier *job,* which is going to run behind the scenes to deliver you the files requested. This operation is an asynchronous operation in Amazon S3 Glacier. For each job, Amazon S3 Glacier maintains all the information related to the job such as type of the job, creation date, status of the job, completion date, and so on. As soon as the job is finished, you should be able to download the job output, which is your files. Amazon S3 Glacier also supports notifications; therefore, when the job is finished, you can be notified. You can configure Amazon Simple Notification Service with Amazon S3 Glacier, which allows you to create notifications once a job is finished. When you want to retrieve one or more archives from your vault, you need to submit a job, which runs behind the scenes.

Accessing Amazon S3 Glacier

There are three ways to access Amazon S3 Glacier:

•   You can access it directly via the Amazon S3 Glacier API or SDK.

•   You can access it via Amazon S3 lifecycle integration. You can create various lifecycle policies to move the files to Amazon S3 Glacier. For example, you can write a policy that says to move any file residing in S3 that is more than one year old to Amazon S3 Glacier, move any file that is more than three months old to Amazon S3 Glacier, and so on.

•   You can access it via various third-party tools and gateways.

Uploading Files to AmazonS3 Glacier

Uploading files to Amazon S3 Glacier is pretty simple. You can use the Internet directly to upload the files. You can also upload the files to S3 Glacier from your corporate data center using AWS Direct Connect. If you have a large number of files or a huge data set, you can use AWS Snowball to ship your files and then upload them to S3 Glacier.

To upload a file in Amazon S3 Glacier first, you need to create a vault, which is the container where your data will be stored. You can visualize this vault as a safety deposit box that you often see in a bank. Vault names must be unique within an account and within the region in which the vault is being created. During the creation of a vault, you can also enable notifications.

Then you need to create an access policy that you can attach directly to your S3 Glacier vault (the resource) to specify who has access to the vault and what actions they can perform on it.

The next step is to create the archives and upload them to the vault. For large archives, S3 Glacier provides a multipart upload API that enables you to upload an archive in parts. Amazon also provides you with SDKs for Java and .NET to upload files to S3 Glacier. Whenever you upload an archive, Amazon internally creates an archive ID to uniquely identify the archive.

Retrieving Files from Amazon S3 Glacier

There are three ways to retrieve data from Amazon S3 Glacier:

•   **Standard** This is a low-cost option for retrieving data in just a few hours. Typically it takes about three to five hours to retrieve the data. The standard retrieval cost is $0.01 per gigabyte.

•   **Expedited** This is designed for occasional urgent access to a small number of archives. Using expedited retrieval, the data can be accessed almost instantly within one to five minutes. The expedited retrieval cost is $0.03 per gigabyte.

•   **Bulk** This is the lowest-cost option optimized for large retrievals, up to petabytes of data. It takes between five and twelve hours to retrieve the data. The bulk retrieval cost is $0.0025 per gigabyte.

Retrieving files from Amazon S3 Glacier is simple. All the jobs are done in a four-step process.

**1.**   You submit a retrieval job. While submitting the retrieval job, you can specify whether it is standard, expedited, or bulk retrieval. The moment you submit the retrieval job, you get a unique job ID that you can use to track the retrieval job.

**2.**   Depending on the type of job you have submitted, it can take from a few minutes to a few hours to complete the job.

**3.**   Once the job is completed, you get a notification that the job has been completed.

**4.**   You can now download the output.

If you have enabled the lifecycle policy of S3, you can also restore the data using lifecycle management.

AMAZON ELASTIC BLOCK STORE

Amazon Elastic Block Store (EBS) offers persistent storage for Amazon EC2 instances. A persistent storage means the storage is independent outside the life span of an EC2 instance. EBS volumes provide durable block-level storage for use with Amazon EC2 instances. Amazon EBS volumes are network-attached and continue independently after the life of an instance. Amazon EBS volumes are highly available, highly reliable volumes that can be leveraged as an Amazon EC2 instance’s boot partition or attached to a running Amazon EC2 instance as a standard block device. Once you attach an EBS volume to an EC2 instance, the EBS volume can be used as a physical hard drive of a server (of course, you may have to format the volume before using it). You can attach multiple EBS volumes to an EC2 instance. This gives you a great advantage since you can separate the boot volumes from the data volume. You can have one EBS volume for the data and another EBS volume for the boot volume and attach both of them to the same EC2 instance. An EBS volume can be attached to only one EC2 server at a time. You cannot attach or mount the same EBS volume to multiple EC2 instances. The only exception to this is using EBS volume with Multi-Attach. Amazon EBS Multi-Attach enables you to attach a single Provisioned IOPS SSD (io1) volume to up to 16 Nitro-based instances that are in the same AZ. But at any point in time you can detach an EBS volume from an EC2 instance and can mount it to a different EC2 instance. Since EBS volumes are part of a particular AZ, you can detach and reattach an EBS volume between instances within the same AZ. You can’t detach and reattach EBS volumes across different AZs.

Once you attach the EBS volume to the EC2 instance, you can create a file system on top of that. After creating the file system, you can run any kind of workload you want to run on those servers. You can run any type of workload on EBS volumes such as databases, applications, big data workloads, NoSQL databases, web sites, and so on.

EBS volumes can also be used as a boot partition. When EBS volumes are used as a boot partition, then Amazon EC2 instances can be stopped and subsequently restarted, enabling you to pay only for the storage resources used while maintaining your instance’s state. Also, since the EBS volumes persist after the restart, all the data that you store in the EBS volume stays as is. Amazon EBS volumes offer greatly improved durability over local Amazon EC2 instance stores.

Amazon EBS provides the ability to create point-in-time consistent snapshots of your volumes that are then stored in Amazon S3 and automatically replicated across multiple availability zones. These snapshots can be used as the starting point for new Amazon EBS volumes and can protect your data for long-term durability. You can also easily share these snapshots with co-workers and other AWS developers or to another account. The snapshot can also be copied across multiple regions. Therefore, if you are planning for disaster recovery, data center migration, or geographical expansion or you want to leverage multiple AWS regions, you can use EBS snapshots to get your data quickly and provision the infrastructure.

Features of Amazon EBS

The following are the main features of Amazon EBS:

•   **Persistent storage** As discussed previously, the volume’s lifetime is independent of any particular Amazon EC2 instance.

•   **General purpose** Amazon EBS volumes are raw, unformatted block devices that can be used from any operating system.

•   **High availability and reliability** Amazon EBS volumes provide 99.999 percent availability and automatically replicate within their availability zones to protect your applications from component failure. It is important to note that EBS volumes are not replicated across multiple AZs; rather, they are replicated within different facilities within the same AZ.

•   **Encryption** Amazon EBS encryption provides support for the encryption of data at rest and data in transit between EC2 instances and EBS volumes.

•   **Variable size** Volume sizes range from 1GB to 16TB and are allocated in 1GB increments.

•   **Easy to use** Amazon EBS volumes can be easily created, attached, backed up, restored, and deleted.

•   **Designed for resiliency** The annual failure rate (AFR) of Amazon EBS is between 0.1 percent and 0.2 percent.

AWS Block Storage Offerings

There are three types of block storage offerings that AWS provides:

•   Amazon EC2 instance store

•   Amazon EBS SSD-backed volume

•   Amazon EBS HDD-backed volume

AMAZON EC2 INSTANCE STORE

An Amazon EC2 instance store is the local storage of an EC2 instance. It is local to the instance, and unlike EBS volumes, it can’t be mounted into different servers. The instance store is ephemeral, which means all the data stored in the instance store is gone the moment the EC2 instance is shut down. The data neither persists nor is replicated in the instance store. Also, there is no snapshot support for the instance store, which means you won’t be able to take a snapshot of the instance store. The instance store is available in a solid-state drive (SSD) or hybrid hard drive (HDD).

AMAZON EBS VOLUMES

Amazon EBS provides multiple options that allow you to optimize storage performance and cost for any workload you would like to run. These options are divided into two major categories: SSD-backed storage, which is mainly used for transactional workloads such as databases and boot volumes, and HDD-backed storage, which is for throughput-intensive workloads such as log processing and MapReduce.

**Amazon EBS-Backed Volume** Elastic volumes are a feature of Amazon EBS that allow you to dynamically increase capacity, tune performance, and change the type of live volumes with no downtime or performance impact. You can simply use a metric from CloudWatch and write a Lambda function to automate it. (You will learn about CloudWatch and Lambda later in this book.) This allows you to easily right-size your deployment and adapt to performance changes.

**Amazon EBS SSD-Backed Volume** The SSD-backed volumes are of two types, General-Purpose SSD (gp2) and Provisioned IOPS SSD (io1). SSD-backed volumes include the highest performance io1 for latency-sensitive transactional workloads and gp2, which balances price and performance for a wide variety of transactional data.

Before going deep into the types of volume, it is important to understand the concept of IOPS. The performance of a block storage device is commonly measured and quoted in a unit called IOPS, short for input/output operations per second. A drive spinning at 7,200 RPM can perform at 75 to 100 IOPS, whereas a drive spinning at 15,000 RPM will deliver 175 to 210. The exact number will depend on a number of factors, including the access pattern (random or sequential) and the amount of data transferred per read or write operation.

***General-Purpose SSD*** General-Purpose SSD delivers single-digit-millisecond latencies, which is actually a good use case for the majority of workloads. gp2s can deliver between 100 and 16,000 IOPS. gp2 provides great performance for a broad set of workloads, all at a low cost. They reliably deliver three sustained IOPS for every gigabyte of configured storage. For example, a 100GB volume will reliably deliver 300 IOPS. The volume size for gp2 can be anything from 1GB to 16TB. It provides the maximum throughput of 160Mb per volume. You can run any kind of workload in gp2; however, some use cases that are a good fit for gp2 are system boot volumes, applications requiring low latency, virtual desktops, development and test environments, and so on.

The General-Purpose SSDs under 1TB have the ability to burst the IO. This means if you are not using the IOPS, they will get accumulated as IO credit and will be used as IO during peak loads. Let’s see an example to understand this. Say you have a 100GB EBS volume; at 3 IOPS per gigabyte you can expect a total of 300 IOPS. Now whenever you are not using 300 IOPS, the system accumulates the unused IO and keeps it as IO credits. When you are running a peak workload or when there is a heavy activity, it will use the IO credits that it has accumulated, and you will see a lot more IO. The IO can burst up to 3,000 IOPS. When all the IO credits are over, it is going to revert to 300 IOPS. Since the max limit for burst is 3,000 IOPS, volumes more than 1TB in size won’t get the benefit of it. By the way, each volume receives an initial IO credit balance of 5.4 million IO credits, which is good enough to sustain the maximum burst performance of 3,000 IOPS for 30 minutes. This initial credit balance provides a fast initial boot cycle for boot volumes and provides a better bootstrapping experience.

***Provisioned IOPS SSD*** If you have an IO-intense workload such as databases, then you need predictable and consistence IO performance. Provisioned IOPS is designed to cater to that requirement. When you create an EBS volume with Provisioned IOPS, you can specify the IOPS rate, and EBS volumes deliver within 10 percent of the Provisioned IOPS performance 99.9 percent of the time in a given year. You can create an io1 volume between 4GB and 16TB and can specify anything between 100 and 20,000 IOPS per volume. The ratio for volume versus Provisioned IOPS is 1:50. For example, if you have a volume of 100GB, then the IOPS you can provision with that will be 100\*50, which is 5,000 IOPS. Since there is a limit of 20,000 IOPS per volume, even if the size of the volume is 1TB, the maximum IOPS you can provision with that volume would be 20,000 only. If you need more Provisioned IOPS, then you should distribute your workload on multiple EBS volumes. You can also use technologies such as RAID on top of multiple EBS volumes to stripe and mirror the data across multiple volumes. (Please note that RAID is irrespective of the type of EBS volume.) The maximum throughput per volume you will get is 320MB per second, and the maximum IOPS an instance can have is 75,000. Provisioned costs more than the general-purpose version, and it is based on the volume size as well as the amount of IOPS reserved. It can be used for database workloads such as Oracle, PostgreSQL, MySQL, Microsoft SQL Server, Mongo DB, and Cassandra; running mission-critical applications; or running production databases, which need sustained IO performance and so on.

**Amazon EBS HDD-Backed Volume** HDD-backed volumes include Throughput Optimized HDD (st1), which can be used for frequently accessed, throughput-intensive workloads; and HDD (sc1), which is for less frequently accessed data that has the lowest cost.

***Throughput-Optimized HDD*** Throughput-Optimized HDD (st1) is good when the workload you are going to run defines the performance metrics in terms of throughput instead of IOPS. The hard drives are based on magnetic drives. There are lots of workloads that can leverage this EBS volume such as data warehouses, ETL, log processing, MapReduce jobs, and so on. This volume is ideal for any workload that involves sequential IO. Any workload that has a requirement for random IO should be run either on general-purpose or on Provisioned IOPS depending on the price/performance need.

Like the General-Purpose SSDs, Throughput-Optimized HDD also uses a burst-bucket model for performance. In this case, it is the volume size that determines the baseline for the throughput of the volume. It bursts 250MB per second per terabyte up to 500Mb per second. The capacity of the volume ranges from 500GB to 16TB.

***Cold HDD*** Just like st1, Cold HDD (sc1) also defines performance in terms of throughput instead of IOPS. The throughput capacity is less compared to sc1; therefore, the prices are also very cheap for sc1. This is a great use case for noncritical, cold data workloads and is designed to support infrequently accessed data. Similar to st1, sc1 uses a burst-bucket model, but in this case the burst capacity is less since overall throughput is less.

EBS offers snapshot capabilities that can be used to back up EBS volumes. You can take a snapshot at any point in time when a volume is in use without any outages. The snapshot will back up the data that resides on the EBS volume, so if you have any data cached in the application, the snapshot won’t be able to take the backup. To ensure a consistent snapshot, it is recommended that you detach the EBS volume from the EC2 instance, issue the snapshot command, and then reattach the EBS volume to the instance. You can take a snapshot of the root volume as well. In that case, it is recommended that you shut down the machine first and then take it. You can store the snapshots in Amazon S3. There is no limit to the number of snapshots you can take. Each snapshot is uniquely identified by name.

AMAZON ELASTIC FILE SYSTEM

As the name suggests, Amazon Elastic File System (EFS) provides a file system interface and file system semantics to Amazon EC2 instances. When EFS is attached to an EC2 instance, it acts just like a local file system. EFS is also a shared file system, which means you can mount the same file system across multiple EC2 instances, and EFS shares access to the data between multiple EC2 instances, with low latencies.

The following are the attributes of EFS:

•   **Fully managed** It is a fully managed file system, and you don’t have to maintain any hardware or software. There is no overhead of managing the file system since it is a managed service.

•   **File system access semantics** You get what you would expect from a regular file system, including read-after-write consistency, locking, the ability to have a hierarchical directory structure, file operations like appends, atomic renames, the ability to write to a particular block in the middle of a file, and so on.

•   **File system interface** It exposes a file system interface that works with standard operating system APIs. EFS appears like any other file system to your operating system. Applications that leverage standard OS APIs to work with files will work with EFS.

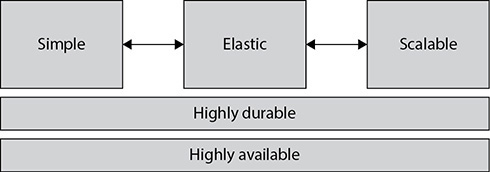
•   **Shared storage** It is a shared file system. It can be shared across thousands of instances. When an EFS is shared across multiple EC2 instances, all the EC2 instances have access to the same data set.

•   **Elastic and scalable** EFS elastically grows to petabyte scale. You don’t have to specify a provisioned size up front. You just create a file system, and it grows and shrinks automatically as you add and remove data.

•   **Performance** It is built for performance across a wide variety of workloads. It provides consistent, low latencies, high throughput, and high IOPS.

•   **Highly available and durable** The data in EFS is automatically replicated across AZs within a region. As a result, your files are highly available, accessible from multiple AZs, and also well protected from data loss.

[Figure 2-10](https://learning.oreilly.com/library/view/aws-certified-solutions/9781260470192/ch2.xhtml#fig2_10) shows the foundation on which EFS is built. As you can see, it aims to be highly durable and highly available. Since your data is automatically available in multiple AZs, EFS is designed to sustain AZ offline conditions. It is superior to traditional NAS availability models since the data is mirrored across multiple AZs. Therefore, it is appropriate for running mission-critical production workloads.



**Figure 2-10** Foundation on which Amazon EFS is built

Simplicity is the foundation of EFS. You can create an EFS instance in seconds. There are no file layers or hardware to manage. EFS eliminates ongoing maintenance and the constant upgrade/refresh cycle. It can be seamlessly integrated with existing tools and apps. It works with the NFS protocol. Using a direct connect and VPC, you can also mount EFS on your on-premise servers via the NFS 4.1 protocol. It is a great use case if you want to transfer a large number of data from the servers running on your premise to the AWS cloud. Also, if you have a workload on your premise that needs additional compute, you can leverage EFS and EC2 servers to offload some of the application workload processing to the cloud. You can even use EFS to back up files to the cloud.

Since EFS is elastic, the file system grows and shrinks automatically as you add or remove files. You don’t have to specify any capacity up front or provision anything, and there is no limit to the number of files you can add in the file system. You just pay for the storage space you use. There are no minimum fees or minimum commitment.

Since EFS is scalable, the throughput and IOPS scale automatically as file systems grow. There is no need to reprovision, adjust performance settings, or do anything in order for your file system to grow to petabyte-scale. The performance aspect is taken care of automatically as you add files. EFS has been designed in such a way that with each gigabyte of stored files you get a particular amount of throughput and IOPS, and as a result, you get consistent performance and are able get low latencies. EFS supports thousands of concurrent NFS connections.

Using Amazon Elastic File System

The first step in using Amazon EFS is to create a file system. The file system is the primary resource in EFS where you store files and directories. You can create ten file systems per account. Of course, like any other AWS service, you can increase this limit by raising a support ticket. To access your file system from instances in a VPC, you create mount targets in the VPC. A mount target is an NFS v4 endpoint in your VPC. A mount target has an IP address and a DNS name you use in your mount command. You need to create a separate mount target from each AZ. Now you can run the mount command on the file system’s DNS name to mount the EFS on your EC2 instance. When the EFS is mounted, the file system appears like a local set of directories and files.

To summarize, these are the steps:

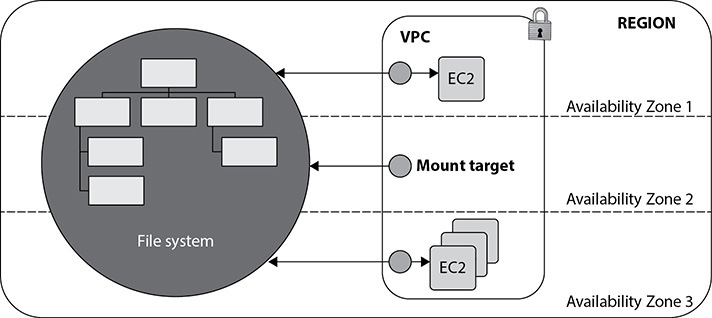
**1.**   Create a file system.

**2.**   Create a mount target in each AZ from which you want to access the file system.

**3.**   Run the mount command from the EC2 instance on the DNS name of the mount of EFS.

**4.**   Start using the EFS.

[Figure 2-11](https://learning.oreilly.com/library/view/aws-certified-solutions/9781260470192/ch2.xhtml#fig2_11) shows the process of creating an EFS instance.



**Figure 2-11** Amazon EFS creation

The EFS instance can be administered via the AWS management console or via the AWS command-line interface (CLI) or AWS software development kit (SDK), which provides EFS support for Java, Python, PHP, .NET, and others. AWS also provides APIs that allow you to administer your EFS file systems via the three options mentioned.

Since EFS provides predictive performance, it serves the vast majority of file workloads, covering a wide spectrum of performance needs from big data applications that are massively parallelized and require the highest possible throughput to single-threaded, latency-sensitive workloads.

There are a lot of places where it can be used. Some of the most common use cases of EFS are genomics, big data analytics, web serving, home directories, content management, media-intensive jobs, and so on.

Performance Mode of Amazon EFS

General-purpose mode is the default for Amazon EFS. It is optimized for latency-sensitive applications and general-purpose file-based workloads. This mode is the best option for the majority of use cases.

Max I/O mode is optimized for large-scale and data-heavy applications where tens, hundreds, or thousands of EC2 instances are accessing the file system. It scales to higher levels of aggregate throughput and operations per second with a trade-off of slightly higher latencies for file operation.

You can use Amazon CloudWatch metrics to get visibility into EFS’s performance. You can use CloudWatch to determine whether your application can benefit from the maximum I/O. If not, you’ll get the best performance in general-purpose mode.

ON-PREMISE STORAGE INTEGRATION WITH AWS

If you have to transfer a large amount of data from your data centers, uploading it to S3 might not be a good choice. When you are planning to transfer more than 10TB of data, you can integrate your on-premise storage with AWS. AWS offers a couple of options through which you can move your data to the cloud. From a certification point of view, this topic is not important, but as a solutions architect, you should know about these offerings.

AWS Storage Gateway

The AWS Storage Gateway (SGW) service is deployed as a virtual machine in your existing environment. This VM is called a *storage gateway*, and you connect your existing applications, storage systems, or devices to this storage gateway. The storage gateway provides standard storage protocol interfaces so apps can connect to it without changes. The gateway in turn connects to AWS so you can store data securely and durably in Amazon S3 Glacier.

The gateway optimizes data transfer from on-premises to AWS. It also provides low-latency access through a local cache so your apps can access frequently used data locally. The service is also integrated with other AWS services such as CloudWatch, CloudTrail, and IAM. Therefore, you can leverage these services within the storage gateway running on-premise. The storage gateway supports three storage interfaces, as described here:

•   The *file gateway* enables you to store and retrieve objects in Amazon S3 using industry-standard file protocols. Files are stored as objects in your S3 buckets and accessed through a Network File System (NFS) mount point. Ownership, permissions, and time stamps are durably stored in S3 in the user metadata of the object associated with the file. Once objects are transferred to S3, they can be managed as native S3 objects, and bucket policies such as versioning, lifecycle management, and cross-region replication apply directly to objects stored in your bucket. It also supports Server Message Block (SMB) and NFS file shares.

•   The *volume gateway* presents your applications with disk volumes using the iSCSI block protocol. Data written to these volumes can be asynchronously backed up as point-in-time snapshots of your volumes and stored in the cloud as Amazon EBS snapshots. You can set the schedule for when snapshots occur or create them via the AWS Management Console or service API. Snapshots are incremental backups that capture only changed blocks. All snapshot storage is also compressed to minimize your storage charges. When connecting with the block interface, you can run the gateway in two modes: cached and stored. In cached mode, you store your primary data in Amazon S3 and retain your frequently accessed data locally. With this mode, you can achieve substantial cost savings on the primary storage, minimizing the need to scale your storage on-premises, while retaining low-latency access to your frequently accessed data. You can configure up to 32 volumes of 32TB each, for a total 1PB storage per gateway. In stored mode, you store your entire data set locally while performing asynchronous backups of this data in Amazon S3. This mode provides durable and inexpensive off-site backups that you can recover locally or from Amazon EC2.

•   The *tape gateway* presents the storage gateway to your existing backup application as an industry-standard iSCSI-based virtual tape library (VTL), consisting of a virtual media changer and virtual tape drives. You can continue to use your existing backup applications and workflows while writing to a nearly limitless collection of virtual tapes. Each virtual tape is stored in Amazon S3. When you no longer require immediate or frequent access to data contained on a virtual tape, you can have your backup application archive it from the virtual tape library into Amazon S3 Glacier, further reducing storage costs.

AWS Snowball and AWS Snowball Edge

Snowball is an AWS import/export tool that provides a petabyte-scale data transfer service that uses Amazon-provided storage devices for transport. Previously, customers had to purchase their own portable storage devices and use these devices to ship their data. With Snowball, customers are now able to use highly secure, rugged, Amazon-owned network-attached storage (NAS) devices, called *Snowballs,* to ship their data. Once the Snowballs are received and set up, customers are able to copy up to 80TB data from their on-premises file system to the Snowball via the Snowball client software and a 10Gbps network interface. Snowballs come with two storage sizes: 50TB and 80TB. Prior to transfer to the Snowball, all the data is encrypted by 256-bit GSM encryption by the client. When customers finish transferring their data to the device, they simply ship the Snowball back to an AWS facility where the data is ingested at high speed into Amazon S3.

AWS Snowball Edge, like the original Snowball, is a petabyte-scale data transfer solution, but it transports more data, up to 100TB of data, and retains the same embedded cryptography and security as the original Snowball. Snowball Edge comes in two flavors: Snowball Edge Compute Optimized and Snowball Edge Storage Optimized. In addition, Snowball Edge hosts a file server and an S3-compatible endpoint that allows you to use the NFS protocol, S3 SDK, or S3 CLI to transfer data directly to the device without specialized client software. Multiple units may be clustered together, forming a temporary data collection storage tier in your data center so you can work as data is generated without managing copies. As your storage needs scale up and down, you can easily add or remove devices to/from the local cluster and return them to AWS.

AWS Snowmobile

AWS Snowmobile is a secure, exabyte-scale data transfer service used to transfer large amounts of data into and out of AWS. Each Snowmobile instance can transfer up to 100PB. When you order a Snowmobile, it comes to your site, and AWS personnel connect a removable, high-speed network switch from Snowmobile to your local network. This makes Snowmobile appear as a network-attached data store. Once it is connected, the secure, high-speed data transfer can begin. After your data is transferred to Snowmobile, it is driven back to AWS where the data is loaded into the AWS service you select, including S3, Amazon S3 Glacier, Redshift, and others.

CHAPTER REVIEW

In this chapter, you learned about the various storage offerings from AWS.

Amazon S3 is the object store that has 99.99999999999 percent durability. S3 can be used for a variety of purposes. The objects are stored in a bucket in S3, and the name of the bucket must be unique. S3 has different classes of storage. Amazon S3 Standard is the default storage. IA is an Amazon S3 storage class that is often used for storing data that is accessed less frequently. Amazon S3 Intelligent Tiering is a storage class which moves the data back and forth from one storage class to other based on the access patterns. S3 One Zone-IA stores data in a single AZ Amazon S3 Glacier is the storage class mainly used for data archiving. Amazon S3 Glacier Deep Archive is used for long-term retention and can be used as a replacement for magnetic tape drives.

Amazon Elastic Block Store (EBS) offers persistent storage for Amazon EC2 instances. It is the block storage that can be used in AWS. EBS can be based on SSD or HDD. The SSD-backed volumes are of two types: General-Purpose SSD (gp2) and Provisioned IOPS SSD (io1). HDD-backed volumes include Throughput Optimized HDD (st1), which can be used for frequently accessed, throughput-intensive workloads; and HDD (sc1), which is for less frequently accessed data that has the lowest cost. Some EC2 instances also have a local storage built in that is called the *instance store*. The instance store is ephemeral, and it is gone whenever you terminate the instance.

Amazon Elastic File System provides a shared file system across many EC2 instances. EFS shares access to the data between multiple EC2 instances, with low latencies. If you need a shared file system for your applications, you can use EFS.

Amazon Storage Gateway can be integrated with your applications running on-premise to the storage on AWS so you can transfer the data from your data center to AWS.

AWS Snowball and AWS Snowball Edge are devices that can be used to ship a large amount of data to AWS. AWS Snowball has a capacity of either 50TB or 80TB, whereas Snowball Edge has a capacity of up to 100TB. As mentioned, Snowball Edge can be Compute Optimized or Storage Optimized.

Amazon Snowmobile is an exabyte-scale data transfer service. Each Snowmobile can transfer up to 100PB. It is delivered to your site container.

**Lab 2-1: Creating, Moving, and Deleting Objects in Amazon S3**

This lab helps you navigate the AWS console and use the Amazon S3 service using the AWS management console. In this lab, you will do the following:

•   Create a bucket in Amazon S3

•   Add an object in Amazon S3

•   View an object in Amazon S3

•   Cut, copy, and paste the file from one bucket to another

•   Make an object public and access the object from a URL

•   Download the object from Amazon S3 to a local machine

•   Delete an object in Amazon S3

If you don’t have an account with AWS, you need to open one before you can start working on the labs. To create an account, go to <https://aws.amazon.com/console/> and in the top-right corner click Create An AWS Account. You can sign up using your e-mail address or phone number. You will require a credit card to sign up for an AWS account. AWS offers most of its service for free for one year under the Free Tier category. Please read the terms and conditions of the Free Tier to understand what is included for free and what is not. You should be up and running with an AWS account in a couple of minutes.

**1.**   Log in to your AWS account. Once you log in to your account, you will notice at the top-right side your default AWS region. Click the region; you will notice a drop-down list. Choose the region of choice where you want to host your bucket.

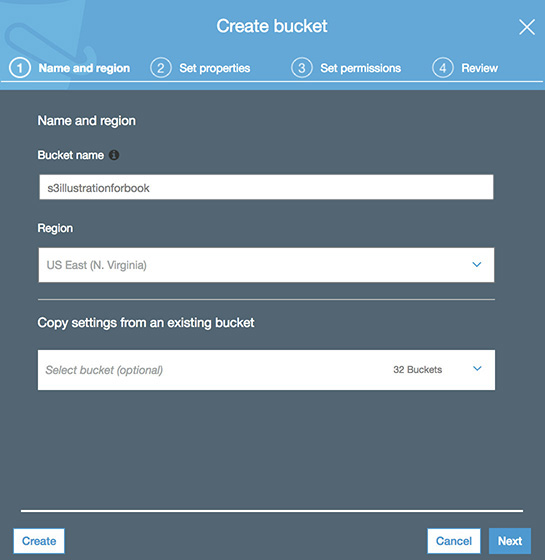
**2.**   From the console, choose S3 from AWS Service.

**3.**   Click Create Bucket.

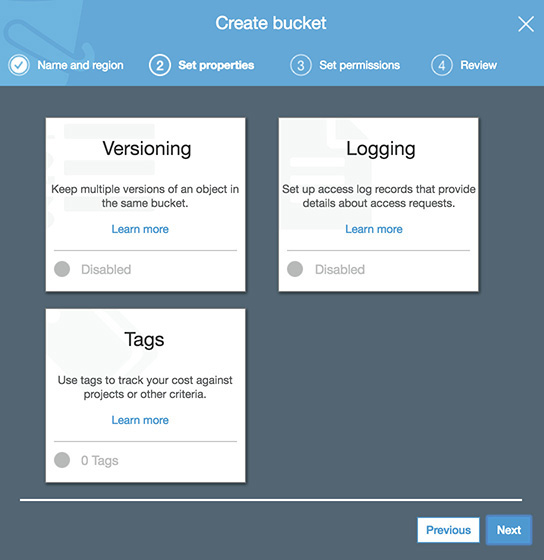
**4.**   In Bucket Name, put a unique name for your bucket. To create a bucket, you are not charged anything. It’s only when you upload stuff in your bucket that you are charged (you won’t be charged if you are within the Free Tier limit).

**5.**   In Region, choose US East (N. Virginia).

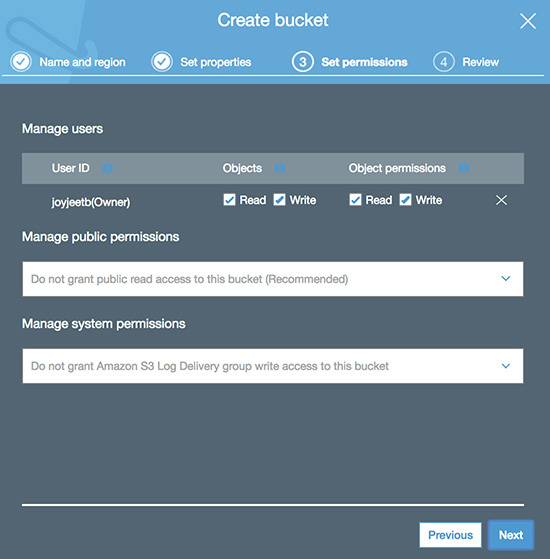
**6.**   In Copy Settings From An Existing Bucket, don’t type anything. Click Next.



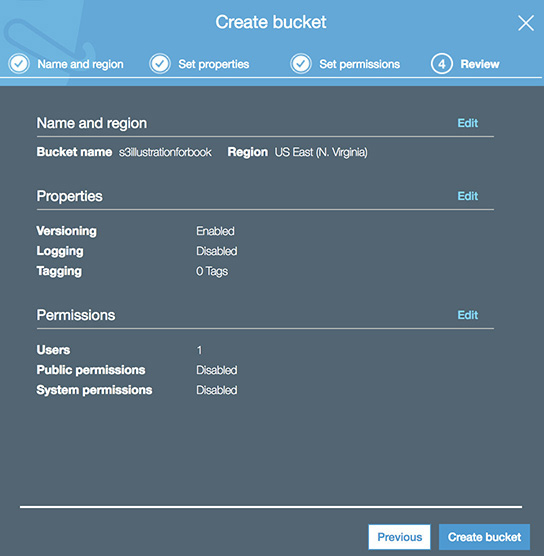
**7.**   Leave the defaults for Versioning, Logging, and Tags; then click Next.



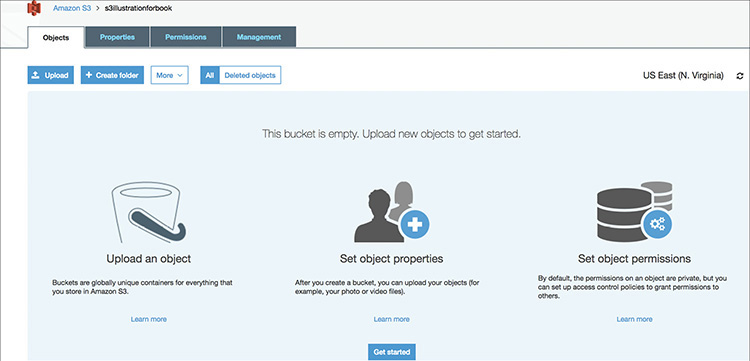
**8.**   On the next screen, called Manage Users, leave Manage Public Permissions and Manage System Permissions at the defaults and click Next.



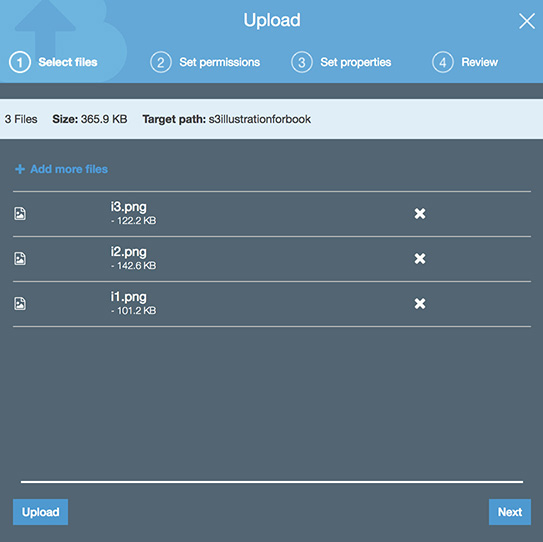
**9.**   Click Create Bucket. Now the bucket will be created for you. Next, you will add an object in the newly created bucket.



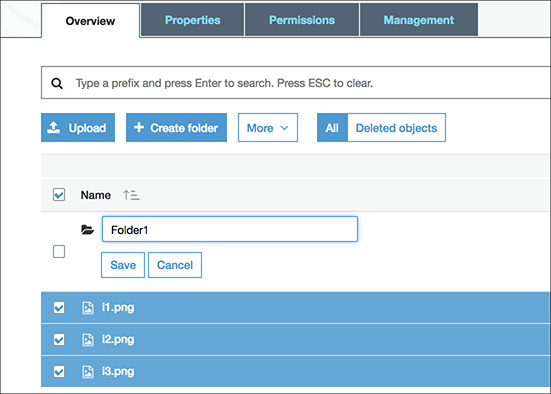
**10.**   You should be able to see the newly created bucket; click it. You will see a couple of options such as Upload, Create Folder, More, and so on.



**11.**   Click Upload and upload a few files. Keep everything at the defaults in the next three screens and upload the files.

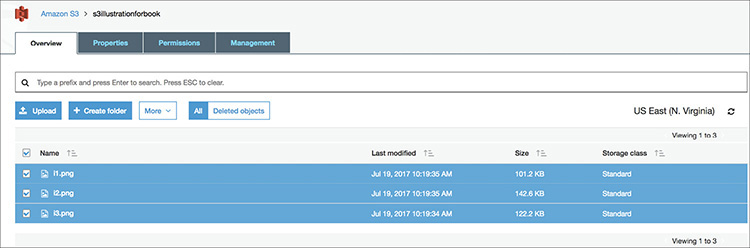


**12.**   Click Create Folder and create a folder with a name of your choice.



**13.**   Go to the folder and upload a few files.

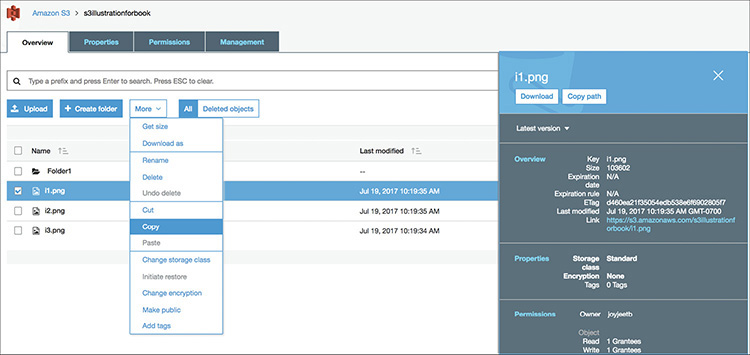
**14.**   Go to your S3 bucket and select a file that you have uploaded. Notice the pop-up on the right once the file is selected. It has all the information related to the object you have uploaded.

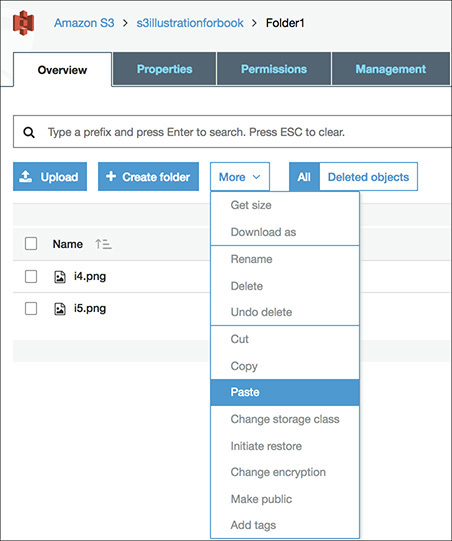


**15.**   With the file selected, click More on the top and browse all the options.

**16.**   Do the same for the files you have created under the folder.

**17.**   Now go to the original file you have created (the first file), select the file, and click More. Click the Copy option. Go to the new folder you have created, click More, and click Paste. You will notice the file is now copied to the new folder.



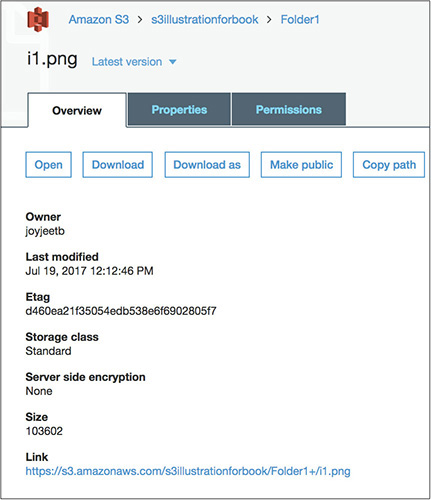


**18.**   In the same way, try Cut and Paste. By cutting and pasting, you can copy the files from one bucket to other.

**19.**   Go to the new folder, select the file, click More | Download, and save the file to your local machine. The file will be stored in your local machine.

**20.**   Again, from the new folder, select the file from the S3 bucket, and click More | Make Public.

**21.**   Reselect the file, and from the right side on the Object tab, select the URL for the file. Now you have made your file publicly accessible from anywhere in the world.



**22.**   Copy the URL and paste it in a different browser window. You will notice that now you can download the file without even logging into S3. Since the file is public, anyone who has this URL can download the file.

**23.**   Keep the file selected, and from S3 bucket click More | Delete.

The file will be deleted from the S3 bucket.

Congratulations! You have finished the first lab on S3.

**Lab 2-2: Using Version Control in Amazon S3**

In this lab, you will do the following:

•   Create an Amazon S3 bucket and enable versioning

•   Upload multiple versions of the same file

•   Delete a file

•   Restore the file after deletion

Log in to the AWS admin console, and from Services select Amazon S3.

**1.**   Repeat steps 1 to 6 from Lab 2-1.

**2.**   In the Versioning window, click the Versioning box. A window will appear asking you to select Enable Versioning or Suspend Versioning.

**3.**   Choose Enable Versioning and click Save; then click Next.



**4.**   Leave everything at the defaults on the next two screens.

**5.**   Click Create Bucket. Now you will have the newly created bucket with versioning on. By the way, you can enable versioning on existing buckets as well. Try that and see how it works.

**6.**   Now create a text file called s3versiontest.text and insert some text in it. For simplicity insert *California* in the text file and save it.

**7.**   Upload the file in the newly created bucket.

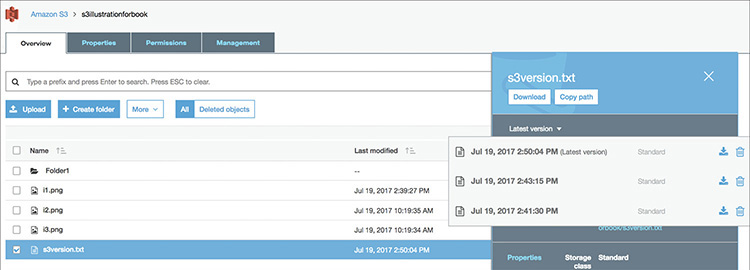
**8.**   From your local machine, replace the word *California* with *Texas* in the s3versiontest.text file.

**9.**   Upload the file again.

**10.**   Repeat steps 8 and 9 a couple of times, every time replacing the text with a new state name.

**11.**   Go to the S3 bucket; you will see there is only one file called s3versiontext.text.

**12.**   Select the file. On the right you will notice a button called Latest Version. Click it, and you will notice all the file versions.



**13.**   Download the files with different versions to your local machine. Download them with different names and don’t overwrite the files on your local machine since the file name will be the same. Compare the files; you will notice that file with the text *California*, *Texas*, and other states all are there.

**14.**   Select the file and click More | Delete.

**15.**   Go to the S3 bucket, and you will see a button called Deleted Objects.

**16.**   Click Deleted Objects, and you will notice your deleted file is there.

**17.**   Select the file and click More | Undo Delete. You have now restored the deleted file successfully.

**18.**   Create another S3 bucket without enabling versioning.

**19.**   Repeat steps 6 to 8.

**20.**   Download the file. You will notice that the file has been overwritten, and there is no option to download the previous version of the file.

**21.**   Repeat step 10, and download each file; you will notice each time the file gets overwritten.

**22.**   Try to repeat steps 14 to 17.

You will realize that the button Deleted Objects does not exist, which means you can no longer restore a deleted file if versioning is not enabled.

This completes the second lab.

**Lab 2-3: Using the Bucket Policy Generator for Amazon S3**

In this lab, you will generate a bucket policy using the AWS Policy Generator. The AWS Policy Generator is a tool that enables you to create various policies and use them to control access to AWS products and resources.

**1.**   Go to the URL <https://awspolicygen.s3.amazonaws.com/policygen.html>.

**2.**   In the Select Policy Type drop-down list, select S3 Bucket Policy.

**3.**   In Add Statement(s), do the following:

**A.**   Set Effect to Allow.

**B.**   Set Principal to \* (this gives access to everyone).

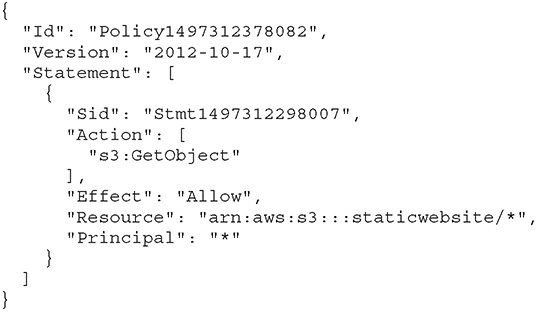
**C.**   Set AWS Service to Amazon S3.

**D.**   In Actions, select GetObject.

**E.**   In ARN, enter **arn:aws:s3:::staticweb site/\***.

**4.**   Click the button Add Statement.

**5.**   Click Generate Policy. You will notice that the system has generated the following policy for you. You can use this policy to grant anonymous access to your static web site.



Questions

[**1**](https://learning.oreilly.com/library/view/aws-certified-solutions/9781260470192/ch2.xhtml#ch2_1)**.**   What is the main purpose of Amazon S3 Glacier? (Choose all that apply.)

**A.**   Storing hot, frequently used data

**B.**   Storing archival data

**C.**   Storing historical or infrequently accessed data

**D.**   Storing the static content of a web site

**E.**   Creating a cross-region replication bucket for Amazon S3

[**2**](https://learning.oreilly.com/library/view/aws-certified-solutions/9781260470192/ch2.xhtml#ch2_2)**.**   What is the best way to protect a file in Amazon S3 against accidental delete?

**A.**   Upload the files in multiple buckets so that you can restore from another when a file is deleted

**B.**   Back up the files regularly to a different bucket or in a different region

**C.**   Enable versioning on the S3 bucket

**D.**   Use MFA for deletion

**E.**   Use cross-region replication

[**3**](https://learning.oreilly.com/library/view/aws-certified-solutions/9781260470192/ch2.xhtml#ch2_3)**.**   Amazon S3 provides 99.999999999 percent durability. Which of the following are true statements? (Choose all that apply.)

**A.**   The data is mirrored across multiple AZs within a region.

**B.**   The data is mirrored across multiple regions to provide the durability SLA.

**C.**   The data in Amazon S3 Standard is designed to handle the concurrent loss of two facilities.

**D.**   The data is regularly backed up to AWS Snowball to provide the durability SLA.

**E.**   The data is automatically mirrored to Amazon S3 Glacier to achieve high availability.

[**4**](https://learning.oreilly.com/library/view/aws-certified-solutions/9781260470192/ch2.xhtml#ch2_4)**.**   To set up a cross-region replication, what statements are true? (Choose all that apply.)

**A.**   The source and target bucket should be in a same region.

**B.**   The source and target bucket should be in different region.

**C.**   You must choose different storage classes across different regions.

**D.**   You need to enable versioning and must have an IAM policy in place to replicate.

**E.**   You must have at least ten files in a bucket.

[**5**](https://learning.oreilly.com/library/view/aws-certified-solutions/9781260470192/ch2.xhtml#ch2_5)**.**   You want to move all the files older than a month to S3 IA. What is the best way of doing this?

**A.**   Copy all the files using the S3 copy command

**B.**   Set up a lifecycle rule to move all the files to S3 IA after a month

**C.**   Download the files after a month and re-upload them to another S3 bucket with IA

**D.**   Copy all the files to Amazon S3 Glacier and from Amazon S3 Glacier copy them to S3 IA

[**6**](https://learning.oreilly.com/library/view/aws-certified-solutions/9781260470192/ch2.xhtml#ch2_6)**.**   What are the various way you can control access to the data stored in S3? (Choose all that apply.)

**A.**   By using an IAM policy

**B.**   By creating ACLs

**C.**   By encrypting the files in a bucket

**D.**   By making all the files public

**E.**   By creating a separate folder for the secure files

[**7**](https://learning.oreilly.com/library/view/aws-certified-solutions/9781260470192/ch2.xhtml#ch2_7)**.**   How much data can you store on S3?

**A.**   1 petabyte per account

**B.**   1 exabyte per account

**C.**   1 petabyte per region

**D.**   1 exabyte per region

**E.**   Unlimited

[**8**](https://learning.oreilly.com/library/view/aws-certified-solutions/9781260470192/ch2.xhtml#ch2_8)**.**   What are the different storage classes that Amazon S3 offers? (Choose all that apply.)

**A.**   S3 Standard

**B.**   S3 Global

**C.**   S3 CloudFront

**D.**   S3 US East

**E.**   S3 IA

[**9**](https://learning.oreilly.com/library/view/aws-certified-solutions/9781260470192/ch2.xhtml#ch2_9)**.**   What is the best way to delete multiple objects from S3?

**A.**   Delete the files manually using a console

**B.**   Use multi-object delete

**C.**   Create a policy to delete multiple files

**D.**   Delete all the S3 buckets to delete the files

[**10**](https://learning.oreilly.com/library/view/aws-certified-solutions/9781260470192/ch2.xhtml#ch2_10)**.**   What is the best way to get better performance for storing several files in S3?

**A.**   Create a separate folder for each file

**B.**   Create separate buckets in different regions

**C.**   Use a partitioning strategy for storing the files

**D.**   Use the formula of keeping a maximum of 100 files in the same bucket

[**11**](https://learning.oreilly.com/library/view/aws-certified-solutions/9781260470192/ch2.xhtml#ch2_11)**.**   The data across the EBS volume is mirrored across which of the following?

**A.**   Multiple AZs

**B.**   Multiple regions

**C.**   The same AZ

**D.**   EFS volumes mounted to EC2 instances

[**12**](https://learning.oreilly.com/library/view/aws-certified-solutions/9781260470192/ch2.xhtml#ch2_12)**.**   I shut down my EC2 instance, and when I started it, I lost all my data. What could be the reason for this?

**A.**   The data was stored in the local instance store.

**B.**   The data was stored in EBS but was not backed up to S3.

**C.**   I used an HDD-backed EBS volume instead of an SSD-backed EBS volume.

**D.**   I forgot to take a snapshot of the instance store.

[**13**](https://learning.oreilly.com/library/view/aws-certified-solutions/9781260470192/ch2.xhtml#ch2_13)**.**   I am running an Oracle database that is very I/O intense. My database administrator needs a minimum of 3,600 IOPS. If my system is not able to meet that number, my application won’t perform optimally. How can I make sure my application always performs optimally?

**A.**   Use Elastic File System since it automatically handles the performance

**B.**   Use Provisioned IOPS SSD to meet the IOPS number

**C.**   Use your database files in an SSD-based EBS volume and your other files in an HDD-based EBS volume

**D.**   Use a general-purpose SSD under a terabyte that has a burst capability

[**14**](https://learning.oreilly.com/library/view/aws-certified-solutions/9781260470192/ch2.xhtml#ch2_14)**.**   Your application needs a shared file system that can be accessed from multiple EC2 instances across different AZs. How would you provision it?

**A.**   Mount the EBS volume across multiple EC2 instances

**B.**   Use an EFS instance and mount the EFS across multiple EC2 instances across multiple AZs

**C.**   Access S3 from multiple EC2 instances

**D.**   Use EBS with Provisioned IOPS

[**15**](https://learning.oreilly.com/library/view/aws-certified-solutions/9781260470192/ch2.xhtml#ch2_15)**.**   You want to run a MapReduce job (a part of the big data workload) for a noncritical task. Your main goal is to process it in the most cost-effective way. The task is throughput sensitive but not at all mission critical and can take a longer time. Which type of storage would you choose?

**A.**   Throughput Optimized HDD (st1)

**B.**   Cold HDD (sc1)

**C.**   General-Purpose SSD (gp2)

**D.**   Provisioned IOPS (io1)

Answers

[**1**](https://learning.oreilly.com/library/view/aws-certified-solutions/9781260470192/ch2.xhtml#rch2_1)**.**   **B, C.** Hot and frequently used data needs to be stored in Amazon S3; you can also use Amazon CloudFront to cache the frequently used data. Amazon S3 Glacier is used to store the archive copies of the data or historical data or infrequent data. You can make lifecycle rules to move all the infrequently accessed data to Amazon S3 Glacier. The static content of the web site can be stored in Amazon CloudFront in conjunction with Amazon S3. You can’t use Amazon S3 Glacier for a cross-region replication bucket of Amazon S3; however, you can use S3 IA or S3 RRS in addition to S3 Standard as a replication bucket for CRR.

[**2**](https://learning.oreilly.com/library/view/aws-certified-solutions/9781260470192/ch2.xhtml#rch2_2)**.**   **C.** You can definitely upload the file in multiple buckets, but the cost will increase the number of times you are going to store the files. Also, now you need to manage three or four times more files. What about mapping files to applications? This does not make sense. Backing up files regularly to a different bucket can help you to restore the file to some extent. What if you have uploaded a new file just after taking the backup? The correct answer is versioning since enabling versioning maintains all the versions of the file and you can restore from any version even if you have deleted the file. You can definitely use MFA for delete, but what if even with MFA you delete a wrong file? With CRR, if a DELETE request is made without specifying an object version ID, Amazon S3 adds a delete marker, which cross-region replication replicates to the destination bucket. If a DELETE request specifies a particular object version ID to delete, Amazon S3 deletes that object version in the source bucket, but it does not replicate the deletion in the destination bucket.

[**3**](https://learning.oreilly.com/library/view/aws-certified-solutions/9781260470192/ch2.xhtml#rch2_3)**.**   **A, C.** By default the data never leaves a region. If you have created an S3 bucket in a global region, it will always stay there unless you manually move the data to a different region. Amazon does not back up data residing in S3 to anywhere else since the data is automatically mirrored across multiple facilities. However, customers can replicate the data to a different region for additional safety. AWS Snowball is used to migrate on-premises data to S3. Amazon S3 Glacier is the archival storage of S3, and an automatic mirror of regular Amazon S3 data does not make sense. However, you can write lifecycle rules to move historical data from Amazon S3 to Amazon S3 Glacier.

[**4**](https://learning.oreilly.com/library/view/aws-certified-solutions/9781260470192/ch2.xhtml#rch2_4)**.**   **B, D.** Cross-region replication can’t be used to replicate the objects in the same region. However, you can use the S3 copy command or copy the files from the console to move the objects from one bucket to another in the same region. You can choose a different class of storage for CRR; however, this option is not mandatory, and you can use the same class of storage as the source bucket as well. There is no minimum number of file restriction to enable cross-region replication. You can even use CRR when there is only one file in an Amazon S3 bucket.

[**5**](https://learning.oreilly.com/library/view/aws-certified-solutions/9781260470192/ch2.xhtml#rch2_5)**.**   **B.** Copying all the files using the S3 copy command is going to be a painful activity if you have millions of objects. Doing this when you can do the same thing by automatically downloading and re-uploading the files does not make any sense and wastes a lot of bandwidth and manpower. Amazon S3 Glacier is used mainly for archival storage. You should not copy anything into Amazon S3 Glacier unless you want to archive the files.

[**6**](https://learning.oreilly.com/library/view/aws-certified-solutions/9781260470192/ch2.xhtml#rch2_6)**.**   **A, B.** By encrypting the files in the bucket, you can make them secure, but it does not help in controlling the access. By making the files public, you are providing universal access to everyone. Creating a separate folder for secure files won’t help because, again, you need to control the access of the separate folder.

[**7**](https://learning.oreilly.com/library/view/aws-certified-solutions/9781260470192/ch2.xhtml#rch2_7)**.**   **E.** Since the capacity of S3 is unlimited, you can store as much data as you want there.

[**8**](https://learning.oreilly.com/library/view/aws-certified-solutions/9781260470192/ch2.xhtml#rch2_8)**.**   **A, E.** S3 Global is a region and not a storage class. Amazon CloudFront is a CDN and not a storage class. US East is a region and not a storage class.

[**9**](https://learning.oreilly.com/library/view/aws-certified-solutions/9781260470192/ch2.xhtml#rch2_9)**.**   **B.** Manually deleting the files from the console is going to take a lot of time. You can’t create a policy to delete multiple files. Deleting buckets in order to delete files is not a recommended option. What if you need some files from the bucket?

[**10**](https://learning.oreilly.com/library/view/aws-certified-solutions/9781260470192/ch2.xhtml#rch2_10)**.**   **C.** Creating a separate folder does not improve performance. What if you need to store millions of files in these separate folders? Similarly, creating separate folders in a different region does not improve the performance. There is no such rule of storing 100 files per bucket.

[**11**](https://learning.oreilly.com/library/view/aws-certified-solutions/9781260470192/ch2.xhtml#rch2_11)**.**   **C.** Data stored in Amazon EBS volumes is redundantly stored in multiple physical locations in the same AZ. Amazon EBS replication is stored within the same availability zone, not across multiple zones.

[**12**](https://learning.oreilly.com/library/view/aws-certified-solutions/9781260470192/ch2.xhtml#rch2_12)**.**   **A.** The only possible reason is that the data was stored in a local instance store that is not persisted once the server is shut down. If the data stays in EBS, then it does not matter if you have taken the backup or not; the data will always persist. Similarly, it does not matter if it is an HDD- or SSD-backed EBS volume. You can’t take a snapshot of the instance store.

[**13**](https://learning.oreilly.com/library/view/aws-certified-solutions/9781260470192/ch2.xhtml#rch2_13)**.**   **B.** If your workload needs a certain number of workloads, the best way would be is to use a Provisioned IOPS. That way, you can ensure the application or the workload always meets the performance metric you are looking for.

[**14**](https://learning.oreilly.com/library/view/aws-certified-solutions/9781260470192/ch2.xhtml#rch2_14)**.**   **B.** Use an EFS. The same EBS volume can’t be mounted across multiple EC2 instances.

[**15**](https://learning.oreilly.com/library/view/aws-certified-solutions/9781260470192/ch2.xhtml#rch2_15)**.**   **B.** Since the workload is not critical and you want to process it in the most cost-effective way, you should choose Cold HDD. Though the workload is throughput sensitive, it is not critical and is low priority; therefore, you should not choose st1. gp2 and io1 are more expensive than other options like st1.