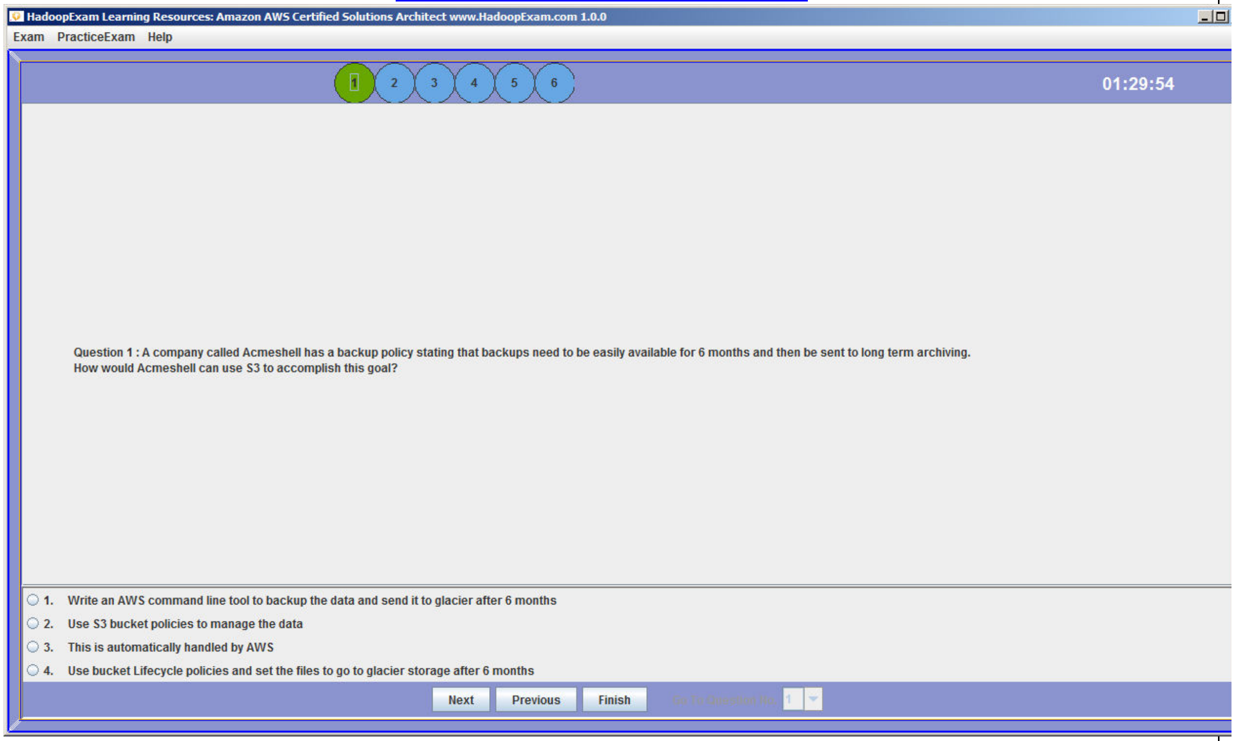
Amazon WebService Tutorial by www.HadoopExam.com

HadoopExam Learning Resources


HadoopExam Learning Resources: Amazon AWS Certified Solutions Architect www.HadoopExam.com 1.0.0
Exam
PracticeExam
Help
1
2
3
4
5
6
01:29:54
Next
Previous
Finish

Question 1 : A company called Acmeshell has a backup policy stating that backups need to be easily available for 6 months and then be sent to long term archiving.
How would Acmeshell can use S3 to accomplish this goal?

1. Write an AWS command line tool to backup the data and send it to glacier after 6 months
2. Use S3 bucket policies to manage the data
3. This is automatically handled by AWS
4. Use bucket Lifecycle policies and set the files to go to glacier storage after 6 months

S3 Lifecycle Management

Lifecycle management defines how Amazon S3 manages objects during their lifetime.
Some objects that you store in an Amazon S3 bucket might have a well defined lifecycle:

If you are uploading periodic logs to your bucket, your application might need these logs for a week or a month after creation, and after that you might want to delete them.
Some documents are frequently accessed for a limited period of time. After that, you might not need real time access to these objects, but your organization might require you to archive them for a longer period and then optionally delete them later. Digital media archives, financial and healthcare records, raw genomics sequence data, long term database backups, and data that must be retained for regulatory compliance are some kinds of data that you might upload to Amazon S3 primarily for archival purposes.

[Type the abstract of the document here. The abstract is typically a short summary of the contents of the document. Type the abstract of the document here. The abstract is typically a short summary of the contents of the document.]

For such objects, you can define rules that identify the affected objects, a timeline, and specific actions you want Amazon S3 to perform on the objects.
Amazon S3 manages object lifetimes with a lifecycle configuration, which is assigned to a bucket and defines rules for individual objects. Each rule in a lifecycle configuration consists of the following:
An object key prefix that identifies one or more objects to which the rule applies.
An action or actions that you want Amazon S3 to perform on the specified objects.

A date or a time period, specified in days since object creation, when you want Amazon S3 to perform the specified action.
You can add these rules to your bucket using either the Amazon S3 console or programmatically.

S3 Buckets

Suppose you have a website with domain name (www.hadoopexam.com or hadoopexam.com) with links to photos and videos stored in your Amazon S3 bucket, examplebucket. By default, all the Amazon S3 resources are private, so only the AWS account that created the resources can access them. To allow read access to these objects from your website, you can add a bucket policy that allows s3:GetObject permission with a condition, using the aws:referer key, that the get request must originate from specific webpages. The following policy specifies the StringLike condition with the aws:Referer condition key.

S3 Bucket Permissions :

A pre-signed URL gives you access to the object identified in the URL, provided that the creator of the pre-signed URL has permissions to access that object. That is, if you receive a pre-signed URL to upload an object, you can upload the object only if the creator of the pre-signed URL has the necessary permissions to upload that object.

All objects and buckets by default are private. The pre-signed URLs are useful if you want your user/customer to be able upload a specific object to your bucket, but you don't require them to have AWS security credentials or permissions. When you create a pre-signed URL, you must provide your security credentials, specify a bucket name an object key, an HTTP method (PUT of uploading objects) and an expiration date and time. The pre-signed URLs are valid only for the specified duration.

You can generate a pre-signed URL programmatically using the AWS SDK for Java or the AWS SDK for .NET. If you are using Visual Studio, you can also use the AWS Explorer to generate a pre-signed object URL without writing any code. Anyone who receives a valid pre-signed URL can then programmatically upload an object.

Note

Anyone with valid security credentials can create a pre-signed URL. However, in order to successfully upload an object, the pre-signed URL must be created by someone who has permission to perform the operation that the pre-signed URL is based upon.

Amazon S3 Object Size

Depending on the size of the data you are uploading, Amazon S3 offers the following options:

Upload objects in a single operation With a single PUT operation you can upload objects up to 5 GB in size.

Upload objects in parts Using the Multipart upload API you can upload large objects, up to 5 TB.

The Multipart Upload API is designed to improve the upload experience for larger objects. You can upload objects in parts. These object parts can be uploaded independently, in any order, and in parallel. You can use a Multipart Upload for objects from 5 MB to 5 TB in size. For more information, see Uploading Objects Using Multipart Upload.

Amazon encourages Amazon S3 customers to use Multipart Upload for objects greater than 100 MB.

The total volume of data and number of objects you can store are unlimited. Individual Amazon S3 objects can range in size from 1 byte to 5 terabytes. The largest object that can be uploaded in a single PUT is 5 gigabytes.

Bucket Namespace is global

The bucket namespace is global - just like domain names. You can't create a bucket with a name that is already used by another Amazon S3 user.In most cases, the common names, such as "images" or "users" will not be available. If you try to create a bucket with a name that already exists, you will get a 409 Conflict.

S3 and Regions

You can choose the geographical Region where Amazon S3 will store the buckets you create. You might choose a Region to optimize latency, minimize costs, or address regulatory requirements. Amazon S3 currently supports the following Regions:

US Standard Uses Amazon S3 servers in the United States : Provides eventual consistency for all requests. This region automatically routes requests to facilities in Northern Virginia or the Pacific Northwest using network maps.

US West (Oregon) Region Uses Amazon S3 servers in Oregon : Provides read-after-write consistency for PUTS of new objects in your Amazon S3 bucket and eventual consistency for overwrite PUTS and DELETES.

US West (Northern California) Region Uses Amazon S3 servers in Northern California : Provides read-after-write consistency for PUTS of new objects in your Amazon S3 bucket and eventual consistency for overwrite PUTS and DELETES.

EU (Ireland) Region Uses Amazon S3 servers in Ireland : Provides read-after-write consistency for PUTS of new objects in your Amazon S3 bucket and eventual consistency for overwrite PUTS and DELETES.

Asia Pacific (Singapore) Region Uses Amazon S3 servers in Singapore : Provides read-after-write consistency for PUTS of new objects in your Amazon S3 bucket and eventual consistency for overwrite PUTS and DELETES.

Asia Pacific (Sydney) Region Uses Amazon S3 servers in Sydney : Provides read-after-write consistency for PUTS of new objects in your Amazon S3 bucket and eventual consistency for overwrite PUTS and DELETES.

Asia Pacific (Tokyo) Region Uses Amazon S3 servers in Tokyo : Provides read-after-write consistency for PUTS of new objects in your Amazon S3 bucket and eventual consistency for overwrite PUTS and DELETES.

South America (Sao Paulo) Region Uses Amazon S3 servers in Sao Paulo : Provides read-after-write consistency for PUTS of new objects in your Amazon S3 bucket and eventual consistency for overwrite PUTS and DELETES.

Objects stored in a Region never leave the Region unless you explicitly transfer them to another Region. For example, objects stored in the EU (Ireland) Region never leave it.

Object Key and Metadata

Each Amazon S3 object has data, a key, and metadata. When you create an object you specify the key name. This key name uniquely identifies the object in the bucket. For example, in Amazon S3 console (see AWS Management Console), when you highlight a bucket, a list of objects in your bucket appear. These names are the object keys. The name for a key is a sequence of Unicode characters whose UTF-8 encoding is at most 1024 bytes long.

Note: If you anticipate that your workload against Amazon S3 will exceed 100 requests per second, follow the Amazon S3 key naming guidelines for best performance. For information, see Request Rate and Performance Considerations.

In addition to the key, each Amazon S3 object has metadata. It is a set of name-value pairs. You can set object metadata at the time you upload it. After you upload the object, you cannot modify object metadata. The only way to modify object metadata is to make copy of the object and set the metadata. For more information, go to PUT Object - Copy in the Amazon Simple Storage Service API Reference. You can use the Amazon S3 management console to update the object metadata but internally it makes an object copy replacing the existing object to set the metadata.There are two kinds of metadata: system metadata and user-defined metadata.

Encryption provides added security for your object data stored in your buckets in Amazon S3. You can encrypt data on your client-side and upload the encrypted data to Amazon S3. In this case, you manage encryption process, the encryption keys, and related tools. Optionally, you might want to use the server-side encryption feature in which Amazon S3 encrypts your object data before saving it on disks in its data centers and decrypts it when you download the objects, freeing you from the tasks of managing encryption, encryption keys, and related tools. You can also use your own encryption keys with the Amazon S3 server-side encryption feature

Server-side encryption encrypts only the object data. Any object metadata is not encrypted.

Instead of using Amazon S3's server-side encryption, you also have the option of encrypting your data before sending it to Amazon S3. You can build your own library that encrypts your objects data on the client side before uploading it to Amazon S3. Optionally, you can use the AWS SDK for Java, which you can use to automatically encrypt your data before uploading it to Amazon S3.

Specifying Encryption Metadata Storage Location : When the Amazon S3 client (using the AmazonS3EncryptionClient class) encrypts data and uploads it to Amazon S3, the encrypted envelope symmetric key is also stored in S3. By default, the encrypted key is stored as user-defined object metadata. After you upload an encrypted object, you can view its properties and see the additional metadata name-value pairs related to encryption. For example, the key name x-amz-meta-x-amz-key and key value equal to the envelope key are set on an client-side encrypted object uploaded to Amazon S3. Optionally, you can also choose to store encryption metadata as an instruction file stored at the same location as the encrypted file. The instruction file will have the same key name as the encrypted data file but with the extension ".instruction" appended. You should use an instruction file when the strength of your encryption key results in a symmetric key that is too big for the object metadata. Metadata should be less than 2 KB. Encryption metadata is either stored as object metadata or an instruction file, but not both.

Managing data consistency

Amazon S3 provides eventual consistency for some operations, so it is possible that new data will not be available immediately after the upload, which could result in an incomplete data load or loading stale data. All uploads to buckets in the US Standard Region are eventually consistent. All other regions provide read-after-write consistency for uploads of new objects with unique object keys. For more information about data consistency, see Amazon S3 Data Consistency Model in the Amazon Simple Storage Service Developer Guide.

To ensure that your application loads the correct data, we recommend the following practices: Create new object keys.

Amazon S3 provides eventual consistency in all regions for overwrite operations. Creating new file names, or object keys, in Amazon S3 for each data load operation provides strong consistency in all regions except US Standard.

Use a manifest file with your COPY operation. The manifest explicitly names the files to be loaded. Using a manifest file enforces strong consistency, so it is especially important for buckets in the US Standard region, but it is a good practice in all regions. Use a named endpoint.

If your cluster is in the US East (N. Virginia) Region, you can improve data consistency and reduce latency by using a named endpoint when you create your Amazon S3 bucket.

Amazon EC2

Amazon Elastic Compute Cloud (Amazon EC2) is a web service that provides resizable compute capacity in the cloud.

It is designed to make web-scale computing easier for developers. Amazon EC2s simple web service interface allows you to obtain and configure capacity with minimal friction. It provides you with complete control of your computing resources and lets you run on Amazons proven computing environment.

Amazon EC2 reduces the time required to obtain and boot new server instances to minutes, allowing you to quickly scale capacity, both up and down, as your computing requirements change.

EBS Storage

The data stored on a local instance store will persist only as long as that instance is alive. However, data that is stored on an Amazon EBS volume will persist independently of the life of the instance. Therefore, we recommend that you use the local instance store for temporary data and, for data requiring a higher level of durability, we recommend using Amazon EBS volumes or backing up the data to Amazon S3. If you are using an Amazon EBS volume as a root partition, you will need to set the Delete On Terminate flag to "N" if you want your Amazon EBS volume to persist outside the life of the instance.

Amazon EBS provides two volume types: Standard Volumes and Provisioned IOPS Volumes. They differ in performance characteristics and price, allowing you to tailor your storage performance and cost to the needs of your applications.

Standard volumes offer storage for applications with moderate or bursty I/O requirements.

These volumes deliver approximately 100 IOPS on average with a best effort ability to burst up to hundreds of IOPS. Standard volumes are also well suited for use as boot volumes, where the burst capability provides fast instance start-up times. Provisioned IOPS volumes are designed to deliver predictable, high performance for I/O intensive, random read and write workloads such as databases. With Provisioned IOPS, you specify an IOPS rate when creating a volume, and then Amazon EBS provisions that rate for the lifetime of the volume.

Amazon EBS currently supports up to 4000 IOPS per Provisioned IOPS volume.

You can stripe multiple volumes together to deliver thousands of IOPS per Amazon EC2 instance to your application.

Unlike the data stored on a local instance store (which persists only as long as that instance is alive), data stored on an Amazon EBS volume can persist independently of the life of the instance. Therefore, we recommend that you use the local instance store only for temporary data. For data requiring a higher level of durability, we recommend using Amazon EBS volumes or backing up the data to Amazon S3. If you are using an Amazon EBS volume as a root partition, set the Delete on termination flag to "No" if you want your Amazon EBS volume to persist outside the life of the instance.

Provisioned IOPS

For any production application that requires fast and consistent I/O performance, we recommend Provisioned IOPS (input/output operations per second) storage.
Provisioned IOPS storage is a storage option that delivers fast, predictable,and consistent throughput performance. When you create a DB instance, you specify an IOPS rate and storage space allocation. Amazon RDS provisions that IOPS rate and storage for the lifetime of the DB instance or until you change it. Provisioned IOPS storage is optimized for I/O intensive, online transaction processing (OLTP) workloads that have consistent performance requirements.

A virtual private cloud is a virtual network that is logically isolated from other virtual networks in the AWS cloud. Amazon Virtual Private Cloud (VPC) lets you launch AWS resources, such as an Amazon RDS or Amazon EC2 instance, into a VPC. The VPC can either be a default VPC that comes with your account or it could be one that you create. All VPCs are associated with your AWS account.

IOPS is available for all your RDS instances it is not depend on which database engine or their deployment strategy (either inside VPC or outside VPC). You can provision a MySQL, PostgreSQL, or Oracle DB instance with up to 30,000 IOPS and 3 TB of allocated storage. You can provision a SQL Server DB instance with up to 10,000 IOPS and 1 TB of allocated storage.

Using Provisioned IOPS Storage with Multi-AZ, Read Replicas, Snapshots, VPC, and DB Instance Classes
For production OLTP use cases, we recommend that you use Multi-AZ deployments for enhanced fault tolerance and Provisioned IOPS storage for fast and predictable performance.
In addition to Multi-AZ deployments, Provisioned IOPS storage complements the following features:

Amazon VPC for network isolation and enhanced security.

Read Replicas The type of storage on a read replica is independent of that on the master DB instance. For example, if the master DB instance uses standard storage, you can add read replicas that use Provisioned IOPS storage and vice versa. If you use standard storage based read replicas with a master DB instance that uses Provisioned IOPS storage, the performance of your read replicas may differ considerably from that of a configuration in which both the master DB instance and the read replicas are using Provisioned IOPS storage.

DB Snapshots If you are using a DB instance that uses Provisioned IOPS storage, you can use a DB snapshot to restore an identically configured DB instance, regardless of whether the target DB instance uses standard storage or Provisioned IOPS storage. If your DB instance uses standard storage, you can use a DB snapshot to restore only a DB instance that uses standard storage.

You can use Provisioned IOPS storage with any DB instance class that runs the DB engine you want; however, smaller DB instance classes will not consistently make the best use Provisioned IOPS storage.
We recommend that you use Provisioned IOPS Optimized storage.

Provisioned IOPS Storage Costs
Because Provisioned IOPS storage reserves resources for your use, you are charged for the resources whether or not you use them in a given month. When you use Provisioned IOPS storage, you are not charged the monthly Amazon RDS I/O charge. If you prefer to pay only for I/O that you consume, a DB instance that uses standard storage may be a better choice.

PUT Bucket notification

This implementation of the PUT operation uses the notification subresource to enable notifications of specified events for a bucket. Currently, the s3:ReducedRedundancyLostObject event is the only event supported for notifications. The s3:ReducedRedundancyLostObject event is triggered when Amazon S3 detects that it has lost all replicas of an object and can no longer service requests for that object.

If the bucket owner and Amazon SNS topic owner are the same, the bucket owner has permission to publish notifications to the topic by default. Otherwise, the owner of the topic must create a policy to enable the bucket owner to publish to the topic. For more information about creating this policy, go to Example Cases for Amazon SNS Access Control.

By default, only the bucket owner can configure notifications on a bucket. However, bucket owners can use a bucket policy to grant permission to other users to set this configuration with s3:PutBucketNotification permission.

After you call the PUT operation to configure notifications on a bucket, Amazon S3 publishes a test notification to ensure that the topic exists and that the bucket owner has permission to publish to the specified topic. If the notification is successfully published to the SNS topic, the PUT operation updates the bucket configuration and returns the 200 OK response with a x-amz-sns-test-message-id header containing the message ID of the test notification sent to topic.

To turn off notifications on a bucket, you specify an empty NotificationConfiguration element in your request

Amazon SWF

Amazon Simple Workflow (Amazon SWF) is a task coordination and state management service for cloud applications. With Amazon SWF, you can stop writing complex glue-code and state machinery and invest more in the business logic that makes your applications unique. The Amazon Simple Workflow Service (Amazon SWF) makes it easier to develop asynchronous and distributed applications by providing a programming model and infrastructure for coordinating distributed components and maintaining their execution state in a reliable way. By relying on Amazon SWF, you are freed to focus on building the aspects of your application that differentiate it.

The fundamental concept in Amazon SWF is the workflow. A workflow is a set of activities that carry out some objective, together with logic that coordinates the activities. For example, a workflow could receive a customer order and take whatever actions are necessary to fulfill it.
Each workflow runs in an AWS resource called a domain, which controls the workflow's scope.
An AWS account can have multiple domains, each of which can contain multiple workflows, but workflows in different domains cannot interact.

When designing an Amazon SWF workflow, you precisely define each of the required activities. You then register each activity with Amazon SWF as an activity type. When you register the activity, you provide information such as a name and version, and some timeout values based on how long you expect the activity to take. For example, a customer may have an expectation that an order will ship within 24 hours. Such expectations would inform the timeout values that you specify when registering your activities.

In the process of carrying out the workflow, some activities may need to be performed more than once, perhaps with varying inputs. For example, in a customer-order workflow, you might have an activity that handles purchased items. If the customer purchases multiple items, then this activity would have to run multiple times. Amazon SWF has the concept of an activity task that represents one invocation of an activity. In our example, the processing of each item would be represented by a single activity task.

An activity worker is a program that receives activity tasks, performs them, and provides results back. Note that the task itself might actually be performed by a person, in which case the person would use the activity worker software for the receipt and disposition of the task. An example might be a statistical analyst, who receives sets of data, analyzes them, and then sends back the analysis.

Let's have an example of SWF, in a customer-order workflow, you might have an activity that handles purchased items. If the customer purchases multiple items, then this activity would have to run multiple times. Amazon SWF has the concept of an activity task that represents one invocation of an activity. In this example, the processing of each item would be represented by a single activity task.

RDS:

Amazon RDS DB snapshots and automated backups are stored in S3.

You can use the AWS Management Console or ModifyDBInstance API to manage the period of time your automated backups are retained by modifying the RetentionPeriod parameter. If you desire to turn off automated backup's altogether, you can do so by setting the retention period to 0 (not recommended). You can manage your user-created DB Snapshots via the DB Snapshots section of the AWS Management Console. Alternatively, you can see a list of the user-created DB Snapshots for a given DB Instance using the DescribeDBSnapshots API and delete snapshots with the DeleteDBSnapshot API.

Amazon RDS enables automated backups of your DB Instance with a 1 day retention period. Free backup storage is limited to the size of your provisioned database and only applies to active DB Instances. For example, if you have 10GB-months of provisioned database storage, we will provide at most 10GB-months of backup storage at no additional charge. If you would like to extend your backup retention period beyond one day, you can do so using the CreateDBInstance API (when creating a new DB Instance) or ModifyDBInstance API (for an existing DB Instance). You can use these APIs to change the RetentionPeriod parameter from 1 to the desired number of days.

When you delete a DB Instance, you can create a final DB Snapshot upon deletion; if you do, you can use this DB Snapshot to restore the deleted DB Instance at a later date. Amazon RDS retains this final user-created DB Snapshot along with all other manually created DB Snapshots after the DB Instance is deleted. Automated backups are deleted when the DB Instance is deleted. Only manually created DB Snapshots are retained after the DB Instance is deleted.

RDS Logs :

You can use the mysqlbinlog utility to download or stream binary logs from Amazon RDS instances running MySQL 5.6. The binary log is downloaded to your local computer, where you can perform actions such as replaying the log using the mysql utility.

Amazon RDS normally purges a binary log as soon as possible, but the binary log must still be available on the instance to be accessed by mysqlbinlog. To specify the number of hours for RDS to retain binary logs, use the mysql.rds_set_configuration stored procedure. Specify a period in which have time to download the logs. If you set the retention period, monitor storage usage for the instance to verify that the instance starts has sufficient storage.

However, Amazon RDS does not currently provide direct access to the binary logs for your Database Instance.

Reduced Redundancy Storage (RRS)

Reduced Redundancy Storage (RRS) is a new storage option within Amazon S3 that enables customers to reduce their costs by storing non-critical, reproducible data at lower levels of redundancy than Amazon S3s standard storage. It provides a cost-effective, highly available solution for distributing or sharing content that is durably stored elsewhere, or for storing thumbnails, transcoded media, or other processed data that can be easily reproduced. Amazon S3s standard and reduced redundancy options both store data in multiple facilities and on multiple devices, but with RRS, data is replicated fewer times, so the cost is less. Amazon S3 standard storage is designed to provide 99.999999999% durability and to sustain the concurrent loss of data in two facilities, while RRS is designed to provide 99.99% durability and to sustain the loss of data in a single facility. Both the standard and RRS storage options are designed to be highly available, and both are backed by Amazon S3s Service Level Agreement.

RRS is designed to provide 99.99% durability of objects over a given year. This durability level corresponds to an average annual expected loss of 0.01% of objects. For example, if you store 10,000 objects using the RRS option, you can on average expect to incur an annual loss of a single object (i.e. 0.01% of 10,000 objects). This annual loss represents an expected average and does not guarantee the loss of 0.01% of objects in a given year.

The RRS option stores objects on multiple devices across multiple facilities, providing 400 times the durability of a typical disk drive, but does not replicate objects as many times as standard Amazon S3 storage, and thus is even more cost effective. In addition, RRS is designed to sustain the loss of data in a single facility.

If an RRS object has been lost, Amazon S3 will return a 405 error on requests made to that object. Amazon S3 also offers notifications for Reduced Redundancy Storage (RRS) object loss. Customers can configure their bucket so that when Amazon S3 detects the loss of an RRS object, a notification will be sent through Amazon Simple Notification Service (SNS). This enables customers to replace lost RRS objects.

DynamoDB and Hash Key

If a specific hash key element has a large range key element set, and the results cannot be retrieved in a single Query request, the ExclusiveStartKey continuation parameter allows you to submit a new query request from the last retrieved item without re-processing the data already retrieved.

Pagination, LastEvaluatedKey, and ExclusiveStartKey

DynamoDB paginates the results from Query and Scan operations. With pagination,
Query and Scan results are divided into distinct pieces; an application can process the first page of results, then the second page, and so on. The data returned from a Query or Scan operation is limited to 1 MB; this means that if you scan a table that has more than 1 MB of data, you'll need to perform another Scan operation to continue to the next 1 MB of data in the table.

If you query for specific attributes that match values that amount to more than 1 MB of data, you'll need to perform another Query request for the next 1 MB of data. The second query request uses a starting point (ExclusiveStartKey) based on the key of the last returned value (LastEvaluatedKey) so you can progressively query or scan for new data in 1 MB increments.

When the entire result set from a Query or Scan has been processed, the LastEvaluatedKey is null.
This indicates that the result set is complete (i.e. the operation processed the last page of data).

If LastEvaluatedKey is anything other than null, this does not necessarily mean that there is more data in the result set. The only way to know when you have reached the end of the result set is when LastEvaluatedKey is null.

A Scan result is an eventually consistent read, meaning that changes to data immediately before the scan takes place might not be included in the scan results. The result set will not contain any duplicate items.

A Query result is an eventually consistent read, but you can optionally request a strongly consistent read instead. An eventually consistent read might not reflect the results of a recently completed PutItem or UpdateItem operation

Eventually Consistent Reads

When you read data (GetItem, BatchGetItem, Query or Scan operations), the response might not reflect the results of a recently completed write operation (PutItem, UpdateItem or DeleteItem).
The response might include some stale data. Consistency across all copies of the data is usually reached within a second; so if you repeat your read request after a short time, the response returns the latest data. By default, the Query and GetItem operations perform eventually consistent reads, but you can optionally request strongly consistent reads. BatchGetItem operations are eventually consistent by default, but you can specify strongly consistent on a per-table basis. Scan operations

are always eventually consistent. For more information about operations in DynamoDB, see Using the DynamoDB API.

Strongly Consistent Reads
When you issue a strongly consistent read request, DynamoDB returns a response with the most up-to-date data that reflects updates by all prior related write operations to which DynamoDB returned a successful response. A strongly consistent read might be less available in the case of a network delay or outage. For the query or get item operations, you can request a strongly consistent read result by specifying optional parameters in your request.

CIDR notation
CIDR notation like this: 10.10.1.32 with a mask of 18 bits allows 14 bits (18 + 14 = 32) to be used for host addresses. This notation is important when defining rules for a security group in order to control the inbound traffic that's allowed to reach your instances.

In CIDR notation, an IP address is represented as A.B.C.D /n, where "/n" is called the IP prefix or network prefix. The IP prefix identifies the number of significant bits used to identify a network.
For example, 192.9.205.22 /18 means, the first 18 bits are used to represent the network and the remaining 14 bits are used to identify hosts. Common prefixes are 8, 16, 24, and 32.

A network access control list (ACL)

A network access control list (ACL) is an optional layer of security that acts as a firewall for controlling traffic in and out of a subnet. You might set up network ACLs with rules similar to your security groups in order to add an additional layer of security to your VPC.

A network ACL is a numbered list of rules that we evaluate in order, starting with the lowest numbered rule, to determine whether traffic is allowed in or out of any subnet associated with the network ACL. The highest number that you can use for a rule is 32766. We suggest that you start by creating rules with rule numbers that are multiples of 100, so that you can insert new rules where you need to later on.

A network ACL has separate inbound and outbound rules, and each rule can either allow or deny traffic. Your VPC automatically comes with a modifiable default network ACL; by default, it allows all inbound and outbound traffic.

Each subnet must be associated with a network ACL; if you don't explicitly associate a subnet with a network ACL, the subnet is automatically associated with the default network ACL. Network ACLs are stateless; responses to allowed inbound traffic are subject to the rules for outbound traffic.

ACLs and Ephemeral Ports
The Network ACL uses an ephemeral port range of 49152-65535. However, you might want to use a different range for your network ACLs. This section explains why. The client that initiates the request chooses the ephemeral port range. The range varies depending on the client's operating system.

Many Linux kernels (including the Amazon Linux kernel) use ports 32768-61000. Requests originating from Elastic Load Balancing use ports 1024-65535. Windows operating systems through Windows Server 2003 use ports 1025-5000. Windows Server 2008 uses ports 49152-65535. Therefore, if a request comes in to a web server in your VPC from a Windows XP client on the Internet, your network ACL must have an outbound rule to enable traffic destined for ports 1025-5000.

If an EC2 instance in your VPC is the client initiating a request, your network ACL must have an inbound rule to enable traffic destined for the ephemeral ports specific to the type of instance (Amazon Linux, Windows Server 2008, and so on.)

In practice, to cover the different types of clients that might initiate traffic to public-facing instances in your VPC; you need to open ephemeral ports 1024-65535. However, you can also add rules to the ACL to deny traffic on any malicious ports within that range. Make sure to place the DENY rules earlier in the table than the rule that opens the wide range of ephemeral ports.

Amazon IAM

AWS Identity and Access Management is a web service that enables Amazon Web Services (AWS) customers to manage users and user permissions in AWS. The service is targeted at organizations with multiple users or systems that use AWS products such as Amazon EC2, Amazon RDS, and the AWS Management Console. With IAM, you can centrally manage users, security credentials such as access keys, and permissions that control which AWS resources users can access.

Without IAM, organizations with multiple users and systems must either create multiple AWS accounts, each with its own billing and subscriptions to AWS products, or employees must all share the security credentials of a single AWS account. Also, without IAM, you have no control over the tasks a particular user or system can do and what AWS resources they might use.

IAM addresses this issue by enabling organizations to create multiple users (each user is a person, system, or application) who can use AWS products, each with individual security credentials, all controlled by and billed to a single AWS account. With IAM, each user is allowed to do only what they need to do as part of the user's job.

To change a user's name or path, you must use the IAM CLI or API. There is no option in the console to rename a user.
To rename IAM users, use the following commands:
CLI: aws iam update-user
API: UpdateUser
When you change a user's name or path, the following happens:
Any policies attached to the user stay with the user under the new name.
The user stays in the same groups under the new name.
The unique ID for the user remains the same. For more information about unique IDs, see Unique IDs.
Any resource or role policies that refer to the user as the principal (the user is being granted access) are automatically updated to use the new name or path. For example, any queue-based policies in

Amazon SQS or resource-based policies in Amazon S3 are automatically updated to use the new name and path.

IAM does not automatically update policies that refer to the user as a resource to use the new name or path; you must manually do that. For example, imagine that user Bob has a policy attached to him that lets him manage his security credentials. If an administrator renames Bob to Robert, the administrator also needs to update that policy to change the resource from this:

arn:aws:iam::account-number-without-hyphens:user/division_abc/subdivision_xyz/Bob
to this:
arn:aws:iam::account-number-without-hyphens:user/division_abc/subdivision_xyz/Robert
This is true also if an administrator changes the path; the administrator needs to update the policy to reflect the new path for the user.

AWS Identity and Access Management is a feature of your AWS account offered at no additional charge. You will be charged only for use of other AWS services by your IAM users.

When IAM creates a user, group, instance profile, or server certificate, it assigns to each entity a unique ID that looks like the following example:

AIDAJQABLZS4A3QDU576Q

For the most part, you use friendly names and ARNs when you work with IAM entities, so you don't need to know the unique ID for a specific entity. However, the unique ID can sometimes be useful when it isn't practical to use friendly names.

One example pertains to reusing friendly names in your AWS account. Within your account, a friendly name for a user or group must be unique. For example, you might create an IAM user named David. Your company uses Amazon S3 and has a bucket with folders each employee; the bucket has an IAM resource policy (a bucket policy) that lets users access only their own folders in the bucket. Suppose that the employee named David leaves your company and you delete the corresponding IAM user. But later another employee named David starts and you create a new IAM user named David. If the resource policy on the bucket is set using the IAM user named David, the policy could end up granting the new David access to information in the Amazon S3 bucket that was left by the former David.

However, every IAM user has a unique ID, even if you create a new IAM user that reuses a friendly name that you deleted before. In the example, the old IAM user David and the new IAM user David have different unique IDs. If you create resource policies for Amazon S3 buckets that grant access by unique ID and not just by user name, it reduces the chance that you could inadvertently grant access to information that an employee should not have.

Another example where user IDs can be useful is if you maintain your own database (or other store) of IAM user information. The unique ID can provide a unique identifier for each IAM user you create, even if over time you have IAM users that reuse a name, as in the previous example.

IAM includes the following features:
Central control of users and security credentials You can control creation, rotation, and revocation of each users AWS security credentials (such as access keys) Central control of user access You can control what data in the AWS system users can access and how they access it Shared AWS resources Users can share data for collaborative projects.

Permissions based on organizational groups You can restrict users AWS access based on their job duties (for example, admin, developer, etc.) or departments. When users move inside the organization, you can easily update their AWS access to reflect the change in their role
Central control of AWS resources Your organization maintains central control of the AWS data the users create, with no breaks in continuity or lost data as users move around within or leave the organization Control over resource creation You can help make sure that users create AWS data only in sanctioned places Networking controls You can help make sure that users can access AWS resources only from within the organizations corporate network, using SSL Single AWS bill Your organization's AWS account gets a single AWS bill for all your users AWS activity.

RDS and Read Replicas

You can create a second-tier Read Replica from an existing first-tier Read Replica. By creating a second-tier Read Replica, you may be able to move some of the replication load from the master database instance to a first-tier Read Replica. Please note that a second-tier Read Replica may lag further behind the master because of additional replication latency introduced as transactions are replicated from the master to the first tier replica and then to the second-tier replica.

A Read Replica will stay active and continue accepting read traffic even after its corresponding source DB Instance has been deleted. If you desire to delete the Read Replica in addition to the source DB Instance, you must explicitly delete the Read Replica using the DeleteDBInstance API or AWS Management Console.

Scaling up the storage or compute resources on a source MySQL instance should be applied
as well in the read replicas in order to avoid storage scarcity or replication lags in your replication setting. You can create up to 5 read replicas from one DB instance. In order for replication to operate effectively, each read replica should have as much compute and storage resources as the source DB instance. If you scale the source DB instance, you should also scale the read replicas.

You may find in some cases that your Read Replica(s) arent able to receive or apply updates from their source Multi-AZ DB Instance after a Mulit-AZ failover. This is because some MySQL binlog events were not flushed to disk at the time of the failover. After the failover, the Read Replica may ask for binlogs from the source that it doesnt have.

To resolve the current issue, you will need to delete the Read Replica and create a new one to replace it. To avoid this issue in the future, setting sync-binlog=1 will greatly reduce the chance that MySQL binlogs needed by the Read Replica will be lost during a crash/failover scenario. As the MySQL documentation explains, even this doesnt completely resolve the issue. To further reduce the likelihood of hitting this issue, set innodb_support_xa=1. Note that there may be a performance penalty for setting these variables. Both sync_binlog and innodb_support_xa are dynamic variables, so if you find that the performance penalty is too great, you can reset them without taking an outage.

This is ultimately a choice between performance and improving the automatic resynchronization of Read Replicas after a source Multi-AZ failover. One of the advantages of Amazon RDS Read Replicas is that they can be quickly re-instantiated when synchronization issues arise by deleting and re-creating them. As such, you dont have to take the performance hit from setting sync-binlog and/or innodb_support_xa if manually dropping out of sync Read Replicas and re-creating them meets your needs.

At this point in time, Amazon RDS allows you to create up to five (5) Read Replicas for a given source DB Instance.

If you promote a read replica that is in turn replicating to other read replicas, those replications stay active. Consider an example where MyDBInstance1 replicates to MyDBInstance2, and MyDBInstance2 replicates to MyDBInstance3. If you promote MyDBInstance2, there will no longer be any replication from MyDBInstance1 to MyDBInstance2, but MyDBInstance2 will still replicate to MyDBInstance3.

The following steps show the general process for promoting a read replica to a Single-AZ DB instance.

1.Stop any transactions from being written to the read replica source DB instance, and then wait for all updates to be made to the read replica. Database updates occur on the read replica after they have occurred on the source DB instance, and this replication "lag" can vary significantly. Use the Replica Lag metric to determine when all updates have been made to the read replica.

2.To be able to make changes to the read replica, you must the set the read_only parameter to 0 in the DB parameter group for the read replica.

3.Perform all needed DDL operations, such as creating indexes, on the read replica. Actions taken on the read replica do not affect the performance of the source DB instance.

4.Promote the read replica by using the Promote Read Replica option on the Amazon RDS console, the CLI command rds-promote-read-replica, or the PromoteReadReplica API operation.

Note
The promotion process takes a few minutes to complete. When you promote a read replica, replication is stopped and the read replica is rebooted. When the reboot is complete, the read replica is available as a Single-AZ DB instance.

<u>RDS and Sharding</u>

Sharding divides up data in your entire database by allocating individual machines or machine groups to provide a unique set of data according to an appropriate group. For example, you might put all users with a surname ending in the letters A-D onto a single server. When a user connects to the application and their surname is known, queries can be redirected to the appropriate MySQL server.
When using sharding with EC2, separate the web server and MySQL server into separate EC2 instances, and then apply the sharding decision logic into your application. Once you know which MySQL server you should be using for accessing the data you then distribute queries to the appropriate server.

Warning
With sharding and EC2, be careful that the potential for failure of an instance does not affect your application. If the EC2 instance that provides the MySQL server for a particular shard fails, then all of the data on that shard becomes unavailable.

Amazon VPC

Amazon VPC lets you create a virtual networking environment in a private, isolated section of the Amazon Web Services (AWS) cloud, where you can exercise complete control over aspects such as private IP address ranges, subnets, routing tables and network gateways. With Amazon VPC, you can define a virtual network topology and customize the network configuration to closely resemble a traditional IP network that you might operate in your own datacenter.

One of the scenarios where you may want to use Amazon RDS in VPC is if you want to run a public-facing web application, while still maintaining non-publicly accessible backend servers in a private subnet. You can create a public-facing subnet for your webservers that has access to the Internet, and place your backend RDS DB Instances in a private-facing subnet with no Internet access.

When you create a VPC, we automatically create a set of DHCP options and associate them with the VPC. This set includes only a single option: domain-name-servers=AmazonProvidedDNS. This is an Amazon DNS server, and this option enables DNS for instances that need to communicate over the VPC's Internet gateway. The string AmazonProvidedDNS maps to a DNS server running on a reserved IP address at the base of the VPC network range "plus two". For example, the DNS Server on a 10.0.0.0/16 network is located at 10.0.0.2.

After you create a set of DHCP options, you can't modify them. If you want your VPC to use a different set of DHCP options, you must create a new set and associate them with your VPC. You can also set up your VPC to use no DHCP options at all.

You can have multiple sets of DHCP options, but you can associate only one set of DHCP options with a VPC at a time. If you delete a VPC, the DHCP options set associated with the VPC are also deleted.

After you associate a new set of DHCP options with a VPC, any existing instances and all new instances that you launch in the VPC use these options. You don't need to restart or relaunch the instances. They automatically pick up the changes within a few hours, depending on how frequently the instance renews its DHCP lease. If you want, you can explicitly renew the lease using the operating system on the instance.

DB Instance in VPC
Following are the pre-requisites necessary to create a DB Instances within a VPC:

You need to have a VPC set up with at least one subnet created in every Availability Zone in the Region you want to deploy your DB Instance. For information on creating Amazon VPC and subnets refer to the Getting Started Guide for Amazon VPC. You need to have a DB Subnet Group defined for your VPC.

You need to have a DB Security Group defined for your VPC (or you can use the default provided).
In addition, you should allocate adequately large CIDR blocks to each of your subnets so that there are enough spare IP addresses for Amazon RDS to use during maintenance activities including scale compute and failover etc.

The basic functionality of Amazon RDS is the same regardless of whether VPC is used. Amazon RDS manages backups, software patching, automatic failure detection, read replicas and recovery whether your DB Instances are deployed inside or outside a VPC.

Amazon RDS DB Instances deployed outside a VPC are assigned an external IP address (to which the Endpoint/DNS Name resolves) that provides connectivity from EC2 or the Internet. In Amazon VPC, Amazon RDS DB instances only have a private IP address (within a subnet that you de).

Scenario when Read Replica Needed
There are a variety of scenarios where deploying one or more Read Replicas for a given source DB Instance may make sense. Common reasons for deploying a Read Replica include:

Scaling beyond the compute or I/O capacity of a single DB Instance for read-heavy database workloads. This excess read traffic can be directed to one or more Read Replicas.

Serving read traffic while the source DB Instance is unavailable. If your source DB Instance cannot take I/O requests (e.g. due to I/O suspension for backups or scheduled maintenance), you can direct read traffic to your Read Replica(s). For this use case, keep in mind that the data on the Read Replica may be stale since the source DB Instance is unavailable.

Business reporting or data warehousing scenarios; you may want business reporting queries to run against a Read Replica, rather than your primary, production DB Instance.

Amazon Simple Email Service (Amazon SES):

Amazon Simple Email Service (Amazon SES) is a cost effective outbound only email sending service built on the reliable and scalable infrastructure that Amazon.com has developed to serve its own customer base. With Amazon SES, you can send transactional email, marketing messages, or any other type of high quality content and you only pay for what you use.

Along with high deliverability, Amazon SES provides easy, real time access to your sending statistics and built in notifications for bounces and complaints to help you fine-tune your email sending strategy.

In addition, the service integrates with other AWS services, making it easy to send emails from applications being hosted on AWS. With Amazon SES there is no long term commitment, minimum spend or negotiation required businesses can utilize a free usage tier, and beyond that pay only low fees for the number of emails sent plus data transfer fees.

Amazon SES is for applications that need to send arbitrary communications via email. Amazon SES supports custom email header fields, and many MIME types. Amazon Simple Email Service (Amazon SES) is a cost effective outbound only email sending service built on the reliable and scalable infrastructure that Amazon.com has developed to serve its own customer base.

You need to request production access to SES only once. If you have been granted production access to Amazon SES, then you can send email from anywhere including any of your Amazon EC2 instances. Amazon Web Services manages the IP addresses used by Amazon SES, and provides reverse DNS records for these addresses.

Amazon SES helps enable your emails to pass SPF policy checks enforced by many ISPs. We recommend that all Amazon SES users publish SPF records authorizing Amazon SES to send from their domains

Amazon SES supports many popular content formats, including documents, images, audio, and video.

Simply call the SendEmail or SendRawEmail APIs repeatedly for each email you would like to send. Software running on Amazon EC2, Amazon Elastic MapReduce, or your own servers can compose and deliver bulk emails via Amazon SES in whatever way best suits your business. If you already have your own bulk mailing software, its easy to update it to deliver through Amazon SES either by modifying the software to directly call Amazon SES, or reconfiguring it to deliver email through an Amazon SES SMTP relay as described above.

Amazon SES provides a full-featured SMTP interface for seamless integration with applications that can send email via SMTP. You can connect directly to this SMTP interface from your applications, or configure your existing email server to use this interface as an SMTP relay.

DHCP Options Sets

The Dynamic Host Configuration Protocol (DHCP) provides a standard for passing configuration information to hosts on a TCP/IP network. The options field of a DHCP message contains the configuration parameters. Some of those parameters are the domain name, domain name server, and the netbios-node-type.

DHCP options sets are associated with your AWS account so that you can use them across all of your virtual private clouds (VPC).

Instances launched into a non-default VPC are private, and as such no public IP addresses are given to them. Instead, a private IP address as well as an unresolvable host name is given to them. In order to provide your own host names you can configure the DHCP options of your VPC to point to your own DNS servers

The Amazon EC2 instances you launch into a nondefault VPC are private by default; they're not assigned a public IP address unless you specifically assign one during launch. By default, all instances in a nondefault VPC receive an unresolvable host name that AWS assigns (for example, ip-10-0-0-202). You can assign your own domain name to your instances, and use up to four of your own DNS servers. To do that, you must specify a special set of DHCP options to use with the VPC. This set can contain other commonly used DHCP options (see the following table for the full list of supported options).