Backup and Recovery Approaches Using Amazon Web Services

December 2012

Simon Elisha

Abstract

Traditional enterprise backup and recovery strategies typically take an agent-based approach whereby the entire contents of a server are backed up over either the local area network (LAN) or the storage area network (SAN). Traditional architectures have required this approach because replacing failed components is complex, time consuming, and operationally intensive. This has, in turn, created a backup environment that is complex to manage and resource intensive to operate—requiring technologies such as data de-duplication and virtual tape libraries to cope with ever-increasing workloads.

The AWS platform enables a far more lightweight approach to backup and recovery due, in part, to the following characteristics:

- Computers are now virtual abstract resources instantiated via code rather than being hardware-based.
- Capacity is available at incremental cost rather than up-front cost.
- Resource provisioning takes place in minutes, lending itself to real-time configuration.
- Server "images" are available on-demand, can be maintained by an organization, and can be activated immediately.

These characteristics offer you opportunities to recover deleted or corrupted data with less infrastructure overhead.

This paper is intended to describe some of the high-level concepts you can leverage to deliver less complex, lightweight data backup and recovery capabilities.

Protecting Configurations Rather Than Servers

The Amazon Elastic Compute Cloud (Amazon EC2[1]) service enables the backup and recovery of a standard server, such as a web server or application server, so that you can focus on protecting configuration and stateful data—rather than the server itself. This set of data is much smaller than the aggregate set of server data, which typically includes various application files, operating system files, temporary files, and so on. This change of approach means that regular nightly incremental or weekly full backups can take far less time and consume less storage space.

When a compute instance is started in Amazon EC2, it is based upon an Amazon Machine Image (AMI)[2] and can also connect to existing storage volumes—for example, Amazon Elastic Block Store (Amazon EBS)[3]. In addition, when launching a new instance, it is possible to pass "user data"[4] to the instance that can be accessed internally as dynamic configuration parameters.

A sample workflow is as follows:

- Launch a new instance of a web server, passing it the "identity" of the web server and any security credentials required for initial setup. The instance is based upon a pre-built AMI that contains the operating system and relevant web-server application (e.g., Apache or IIS).
- Upon startup, a boot script accesses a designated and secured Amazon Simple Storage Service (Amazon S3)[5] bucket that contains the specified configuration file(s).
- The configuration file contains various instructions for setting up the server (e.g., web server parameters, locations of related servers, additional software to install, and patch updates).
- The server executes the specified configuration and is ready for service. An open source tool for performing this process, called cloud-init[6], is already installed on Amazon Linux AMIs and is also available for a number of other Linux distributions.

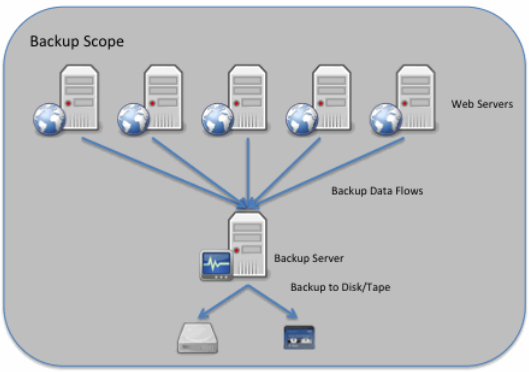


Figure 1: Traditional Backup Approach

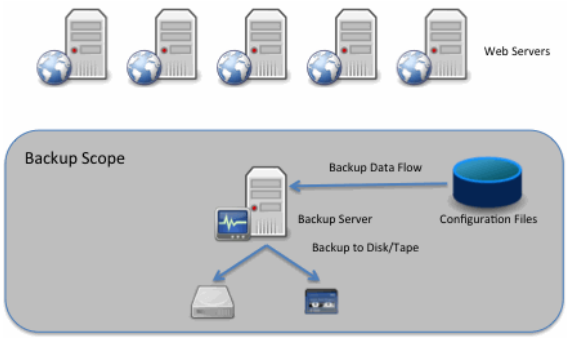


Figure 2: Amazon EC2 Backup Approach

[1] http://aws.amazon.com/ec2/
[2] http://docs.amazonwebservices.com/AWSEC2/latest/UserGuide/index.html?AMIs.html
[3] http://aws.amazon.com/ebs/
[4] http://docs.amazonwebservices.com/AWSEC2/latest/UserGuide/index.html?AESDG-chapter-instancedata.html
[5] http://aws.amazon.com/s3/
[6] https://launchpad.net/cloud-init

In this case, there is no need to back up the server itself. The relevant configuration is contained in the combination of the AMI and the configuration file(s). So the only components requiring backup and recovery are the AMI and configuration file(s).

Consider a web farm of 10 servers, each with an operating system and related configuration files of 5 GB per server—requiring 50 GB of storage and network capacity to do a full backup. (Web content is typically stored in a separate repository that is backed up independently.) Contrast this to the AWS approach where you need to protect only the AMI (of, say, 5 GB) and the relevant configuration files (typically tens of KB). This dramatically reduces the overhead of backup and recovery, eliminates "backup windows," and provides effective version control for the environment.

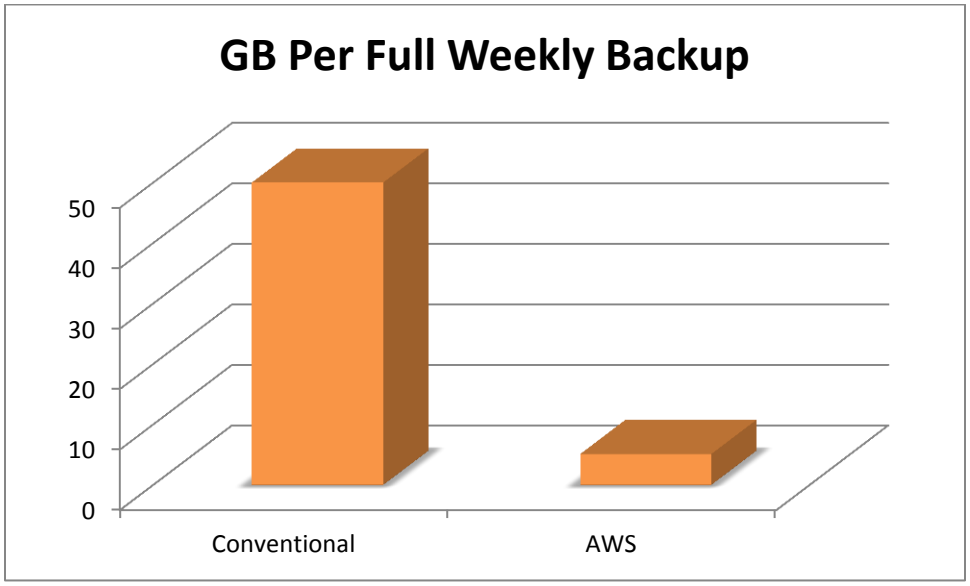


Figure 3: Example reduction in backup data volume

Self-Configuring Instances – Creating Flexibility and Deployment Options

Because you can start and stop instances at will, and have different versions of an application running concurrently, you can leverage more sophisticated and flexible deployment options. The self-configuration of instances enables you to implement techniques such as rolling-upgrades and A/B testing in the environment.

For example, to implement a new version of an application server in the architecture, you can take the following approach:

1. Create a new instance of the application based upon the correct AMI version and relevant configuration files. In our example, we call this "Application Version 2.0."
2. Map "Application Version 2.0" to the relevant load balancer so it is now "in the rotation" of servers available to service a customer request.
3. Once you confirm that "Application Version 2.0" is in production, you can stop or terminate the existing "Application Version 1.0" instances.
4. At this point, the entire application is operating in version 2.0 mode without outages and with simple rollback capability to version 1.0 using the stopped instances.

To take this example further, you may want to utilize A/B testing of new application capabilities or features. In the same way that you were able to introduce a new version of the application server into the architecture for a rolling upgrade, you can use the load balancer to direct certain customers to particular (new version) instances of the application, while the remaining customers continue to use the existing version of the application.

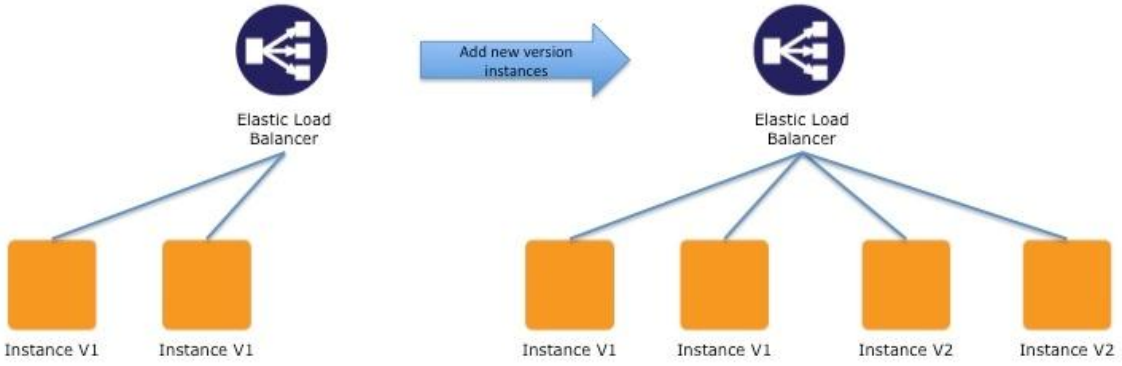


Figure 4: Rolling Upgrade – Adding new version instances

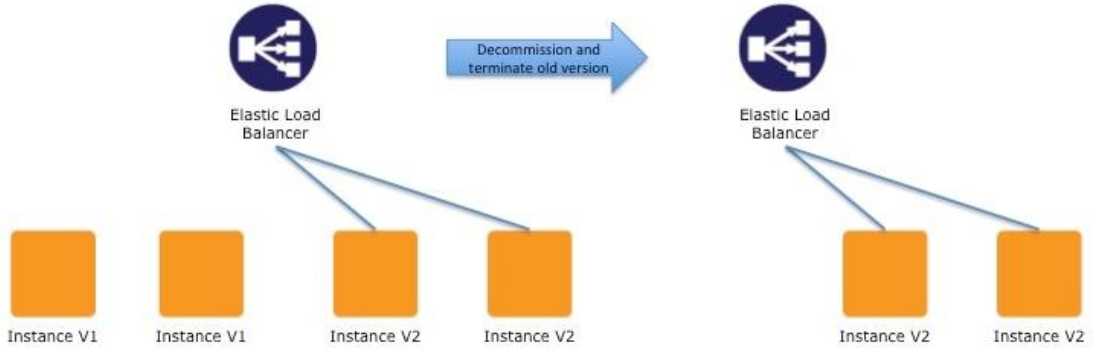


Figure 5: Rolling Upgrade – Decommission and terminate old versions

Other important aspects to consider are security and remediation of compromises. Because instances are easily replaced, you can focus on a strategy of "replace" rather than "repair." This strategy significantly reduces response speed and complexity.

For example, consider a content management system (CMS) that hosts your Internet presence. For some reason, the latest version of the code has not been deployed, and hackers know about and are exploiting a security breach on your site. Forensically analyzing which instances are compromised is time consuming, and trying to "sanitize" each one is often impossible to do with 100% certainty. Instead, you simply terminate the compromised instances and replace them with "fresh" ones. These new instances would then leverage updated configuration files to ensure that the latest patched versions of the software are always deployed. By taking this approach, you eliminate the risk of a security breach that is not completely remediated and guarantee that the new instances are not compromised.

This approach also provides an effective way to "architect for failure," which is a key design pattern when deploying distributed systems at scale. Because you can automatically replace components at will, unexpected failures need not affect service delivery.

Backup and Recovery of the Amazon Machine Image (AMI)

AMIs that you register are automatically stored in your account using Amazon EBS snapshots. These snapshots reside in Amazon S3 and are highly durable[7]. This means that the underlying storage mechanism for the AMIs is protected from multiple failure scenarios.

It is also possible to share AMIs between separate AWS accounts. Consequently, you can create totally independent copies of the AMI by:

- Sharing the original AMI to another specified AWS account that you control
- Starting a new instance based upon the shared AMI
- Creating a new AMI from that running instance

The new AMI is then stored in the second account and is an independent copy of the original AMI. Of course, you can also create multiple copies of the AMI within the same account.

Backup and Recovery of Configuration Files

Customers use a variety of version management approaches for configuration files, and you can follow the same regime for the files used to configure your Amazon EC2 instances. For example, you could store different versions of configuration files in designated locations and securely control them like any other code. You then back up these code repositories using the appropriate backup cycle (e.g., daily, weekly, monthly) and snapshots to protected locations.

Furthermore, you could use Amazon S3 to store your configuration files, taking advantage of the durability of the service e in addition to backing up the files to an alternate location on a regular basis. Bootstrap approaches are limited only by your imagination. We recommend using AWS CloudFormation templates as you can describe your AWS resources, and any associated dependencies or runtime parameters in a simple JSON file.

Backing Up Database and File Servers

Backing up data for database and file servers differs from the web and application layers. In general, database and file servers contain larger amounts of business data (tens of GB to multiple TB) that must be retained and protected at all times. In these cases, you can leverage efficient data movement techniques such as snapshots to create backups that are fast, reliable, and space efficient.

For databases that are built upon RAID-sets of Amazon EBS volumes (and have total storage less than 1 TB), an alternative backup approach is to asynchronously replicate data to another database instance built using a single Amazon EBS volume. While the destination Amazon EBS volume will have slower performance, it is not being used for data access and can be easily snapshotted to Amazon S3 using the Amazon EBS snapshot capability (see the Snapshot Options for Amazon EBS section).

[7] http://aws.amazon.com/s3/ - protecting

Alternative to Backing Up Static Content

If you manage large data sets of static information (e.g., map tiles or web site graphics), you can opt to migrate that data into Amazon S3, which is designed to provide 99.999999999% durability of object storage. This enables the data to be both highly durable while also being served directly from Amazon S3 rather than via web servers—potentially improving application performance.

To protect against logical corruption, you can also use techniques such as object versioning[8], MFA Delete[9] and simply copying the data to another Amazon S3 bucket.

Snapshot Options for Amazon EBS

Amazon EC2 volumes use Amazon EBS to store block-based data. Examples of this are file systems and databases. Amazon EBS natively enables you to create a snapshot of a volume to Amazon S3 using the AWS Management Console, the command line interface (CLI), or the APIs. Using the console, clicking the **Create Snapshot** option commences the creation of a snapshot to Amazon S3.

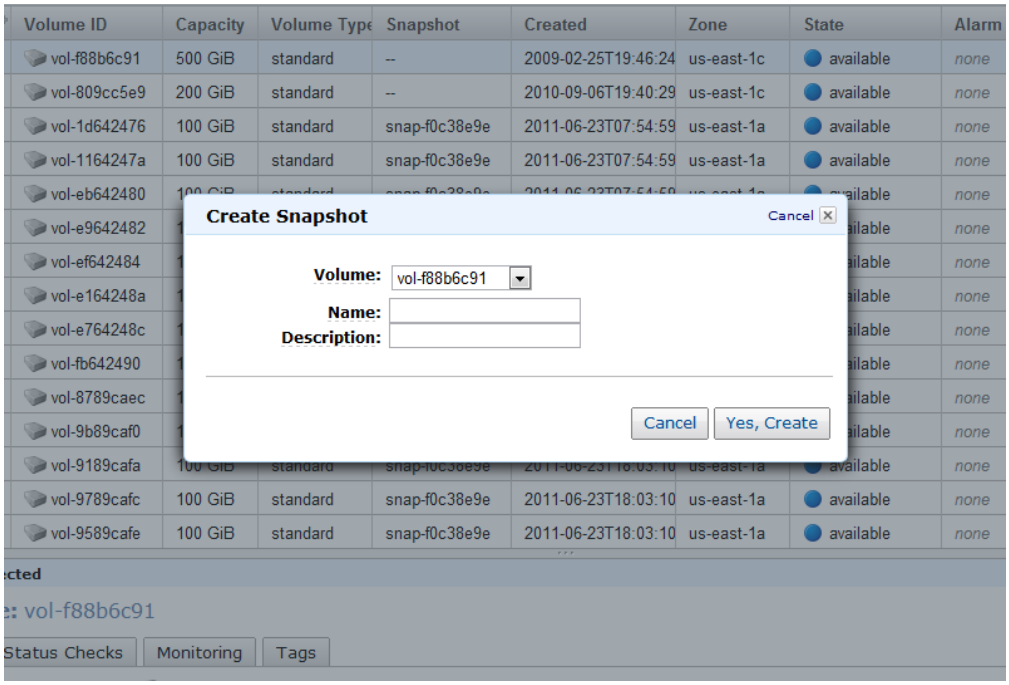


Figure 3 – Creating a snapshot from Amazon EBS using the console.

You can also create the snapshot using the `ec2-create-snapshot` command.

When you apply these commands to a backup strategy, you protect your data directly to durable disk-based storage. You can schedule and issue the commands on a regular basis, and due to the economical pricing of Amazon S3, you can retain many generations of data. Further, because snapshots are block-based, you consume space only for changed data after the initial snapshot is created.

[8] http://docs.amazonwebservices.com/AmazonS3/latest/dev/Versioning.html
[9] http://docs.amazonwebservices.com/AmazonS3/latest/dev/UsingMFADelete.html

To restore data from a snapshot, use the console or the CLI command `ec2-create-volume` to create a new volume from an existing snapshot. For example, to restore a volume to a prior point-in-time backup, you could use the following sequence:

1. Create a new volume from the backup snapshot using the following command:

   `ec2-create-volume -z us-west-1b -snapshot MySnapshotName`

2. Within the Amazon EC2 instance, un-mount the existing volume (e.g., by using `umount` in Linux or the Logical Volume Manager in Windows).
3. Detach the existing volume from the instance using the following command:

   `ec2-detach-volume OldVolume`

4. Attach the new volume that was created from the snapshot using the following command:

   `ec2-attach-volume VolumeID -I InstanceID -d Device`

5. Remount the volume on the running instance.

This process enables a fast and reliable way to restore full volume data as needed. If you need only a partial restore, you can attach the volume to the running instance under a different device name, mount it, and then use operating system copy commands to copy the data from the backup volume to the production volume.

Amazon EBS snapshots can also be copied between AWS Regions using the Amazon EBS snapshot copy capability via the Console, API or GUI[10]. This enables data to be protected out of region without having to manage the underlying replication technology.

Creating Consistent or "Hot" Backups

When you back up a system, it is ideal to have the system in a "quiet" state where it is not performing any processing. From a backup perspective, the "ideal" state is a machine that is accepting no traffic—but this ideal is increasingly rare as 24/7 IT operations become the norm.

As such, it is necessary to "quiesce" the file system or database in order to take a "clean" backup. How you do this depends on your database and/or file system—so due diligence is required. To summarize the process for a database:

- If possible, put the database into "hot backup mode." Alternatively, create a "read replica" copy of the database; this is a copy of the database that is up to date, but runs on a separate instance. Keep in mind that, on AWS, you can run this instance for the duration required to perform the backup and then close it down—saving resources. Also note that there may be a performance impact on the primary database during the existence of the read replica due to additional replication workload.
- Issue the relevant Amazon EBS snapshot commands.
- Take the database out of hot backup mode, or if using a read replica, terminate the read replica instance.

Backing up a file system works similarly, and depends highly on the capabilities of the particular operating system or file system. An example of a file system that can flush its data for a consistent backup is xfs (`xfs_freeze`). If the file system in question does not support the ability to freeze, you should un-mount it, issue the snapshot command, and

[10] http://docs.aws.amazon.com/AWSEC2/latest/UserGuide/ebs-copy-snapshot.html

then re-mount the file system. Alternatively, you can facilitate this process by using a logical volume manager that supports freezing of I/O.

Because the snapshot process is fast to execute, and captures a "point in time," the volumes you are backing up only need be un-mounted for a matter of seconds. This ensures that the backup "window" is as small as possible, and that outage time is predictable and can be effectively scheduled. While the data copy process of creating the snapshot may take longer, the snapshot activity requiring the volume to be un-mounted is very quick. Don't confuse the two processes when structuring your backup regime.

Backups for Amazon Relational Database Service

The Amazon Relational Database Service (Amazon RDS)[11] includes automated backups. This means that you do not need to issue specific commands to create backups of your database.

Amazon RDS provides two different methods for backing up and restoring your DB Instance(s); automated backups and database snapshots (DB Snapshots).

- Automated backups enable point-in-time recovery of your DB Instance. When automated backups are turned on for your DB Instance, Amazon RDS automatically performs a full daily backup of your data (during your preferred backup window) and captures transaction logs (as updates to your DB Instance are made). When you initiate a point-in-time recovery, transaction logs are applied to the most appropriate daily backup in order to restore your DB Instance to the specific time you requested. Amazon RDS retains backups of a DB Instance for a limited, user-specified period of time called the retention period, which by default is one day but can be set to up to thirty-five days. You can initiate a point-in-time restore and specify any second during your retention period, up to the Latest Restorable Time. You can use the `DescribeDBInstances` call to return the latest restorable time for your DB Instance(s), which is typically within the last five minutes. Alternatively, you can find the Latest Restorable Time for a DB Instance by selecting it in the AWS Management Console and looking in the **Description** tab in the lower panel of the console.

- DB Snapshots are user-initiated and enable you to back up your DB Instance in a known state as frequently as you wish, and then restore to that specific state at any time. DB Snapshots can be created with the AWS Management Console or by using the `CreateDBSnapshot` call and are kept until you explicitly delete them with the console or the `DeleteDBSnapshot` call.

Note that when you restore to a point in time or from a DB Snapshot, a new DB Instance is created with a new endpoint. (If you want to, you can delete the old DB Instance by using the AWS Management Console or a `DeleteDBInstance` call.) You do this so you can create multiple DB Instances from a specific DB Snapshot or point in time.

Multi-Volume Backups

In some cases, you may stripe data across multiple Amazon EBS volumes using a logical volume manager in order to increase potential throughput. When using a logical volume manager (e.g., mdadm or LVM), it is important to perform the backup from the volume manager layer rather than the underlying devices. This ensures all metadata is consistent and that the various sub-component volumes are coherent. In these cases, you can use the `ec2-create-snapshot` command for this type of backup with the logical volume manager. You can take a number of approaches to accomplish this, an example being the script created by alestic.com (http://alestic.com/2009/09/ec2-consistent-snapshot).

[11] http://aws.amazon.com/rds/

You can also perform backups of this nature from the logical volume manager or file system level. In these cases, using a "traditional" backup agent enables the data to be backed up over the network. When using tools such as Zmanda, NetBackup, or CommVault, it is important to remember that they expect a consistent server name/IP address. As a result, using these tools in concert with instances deployed in a Virtual Private Cloud (VPC)[12] is the best method to ensure reliability.

An alternative approach is to create a replica of the primary system volumes that exist on a single large volume. This simplifies the backup process, as only one large volume needs to be backed up, and the backup does not take place on the primary system. However, it is important to ascertain whether the single volume can perform sufficiently to maintain changes during the backup and whether the maximum volume size is appropriate for the application.

Other Backup and Recovery Integration Points

Oracle Backup Using the Oracle Secure Backup Cloud Module to Amazon S3

Database administrators are always seeking efficient ways to protect the data contained in Oracle databases. Oracle has made available the ability to backup data directly from the Oracle database to Amazon S3 buckets. This means that backups take advantage of the economical and durable storage made available by Amazon S3 with native integration into the Oracle database framework and operational procedures using RMAN.

Further information about installation and operation of OSB Cloud Module can be found at http://aws.amazon.com/oracle.

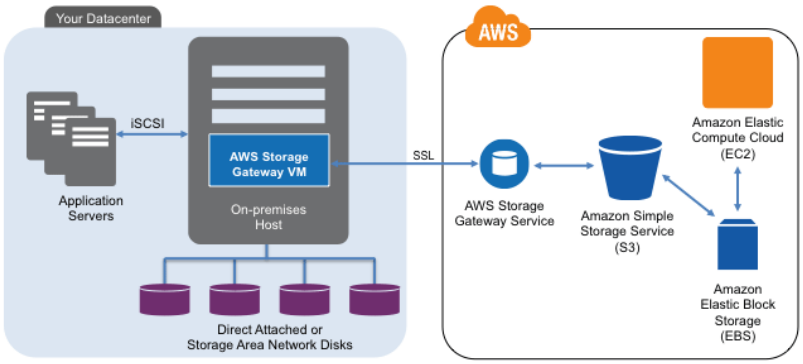
This approach to backup for Oracle enables low-cost, reliable off-premise backup of Oracle databases, and can apply to Oracle databases hosted both on-premise and in Amazon EC2.

Sending On-Premises Backups to Amazon S3

Many backup software vendors now support Amazon S3 as a backup destination (e.g., CommVault Simpana Software Cloud Storage Connector, SecoBackup, and Zmanda). Further, many storage gateways offer integration between existing backup software and Amazon S3 storage (e.g., Nasuni and Riverbed). This is useful in providing an off-site backup that is both durable and cost effective—eliminating the complexity and security risks of off-site tape management.

You can also leverage AWS Direct Connect[13] to provide a dedicated link into Amazon S3 over which your data is sent. This provides the potential for both higher dedicated bandwidth and private connectivity.

AWS Storage Gateway[14] also provides a useful method to send backups to Amazon S3, enabling seamless data migration between AWS's cloud storage and on-premises applications. AWS Storage Gateway stores volume data locally in your infrastructure, and in AWS. In addition to storage replication, it stores the data as an Amazon EBS Snapshot, which you can use to



[12] http://aws.amazon.com/vpc/
[13] http://aws.amazon.com/directconnect/
[14] http://aws.amazon.com/storagegateway

recover data and present it to your Amazon EC2 instances. This makes recovery processes efficient and repeatable.

Managing Backup Generations and Security

When performing backups on an ongoing basis, it is important to implement effective backup rotation strategies to reduce storage overhead, and to ensure the correct versions of data are maintained as per business requirements. A detailed discussion of backup rotation protocols is beyond the scope of this paper.

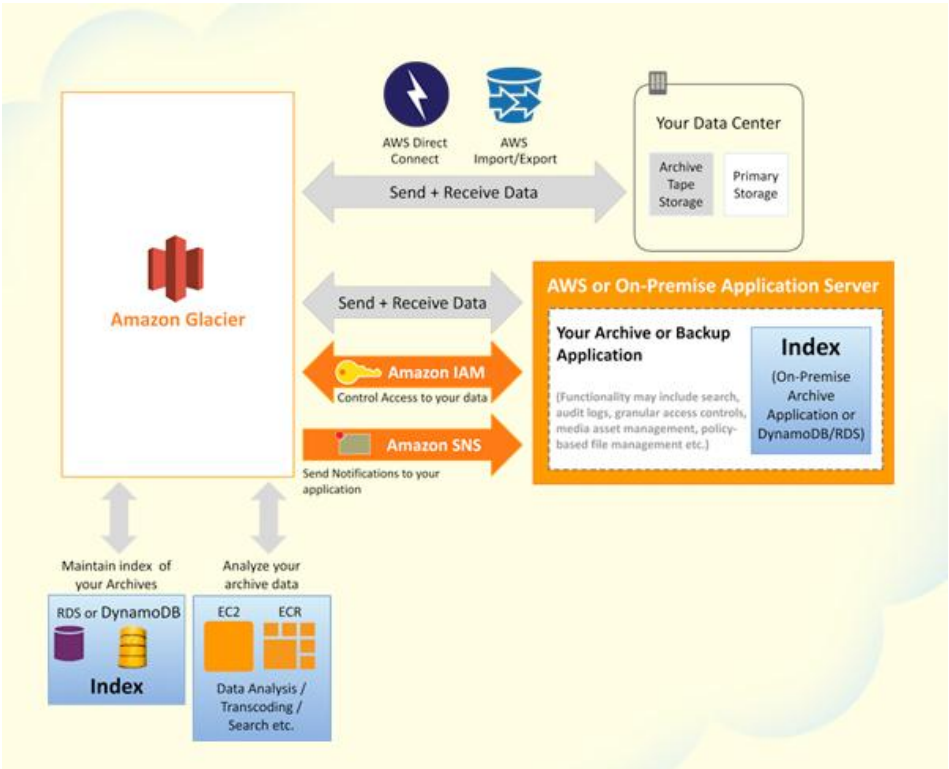
Protocols aside, if your data is sensitive, you should encrypt it while it's in transit and at rest as part of the backup process. An interesting solution to the backup rotation and encryption requirement is the s3napback tool: http://dev.davidsoergel.com/trac/s3napback/.

When using Amazon RDS, your backups are created automatically and retained for up to 8 days—enabling recovery of DB Instances to any second within that period up to the last 5 minutes.

Long-Term Data Archival

Many customers have a requirement to retain digital information for long periods of time (e.g., 7 years, 21 years, life of the patient, or indeterminate duration) in a format whereby it can be retrieved when needed, albeit infrequently. This presents a challenge in being able to store large (and continually growing) volumes of information in a manner that is durable, economical, and low-maintenance. The Amazon Glacier[15] service is designed to enable customers to efficiently and reliably store unlimited amounts of archival data at low cost, with high durability (i.e., designed to provide average annual durability of 99.999999999%), and for long periods of time. You can choose to retrieve your data anytime within a 3 to 5 hour time window, rather than instantaneously. This enables you to effectively meet the dual (and often conflicting) goals of cost effective long-term storage and near real-time data retrieval.

In Amazon Glacier, data is stored as archives that are uploaded to Amazon Glacier and organized into vaults, which customers can control access to using the AWS Identity and Access Management (IAM)[16] service. You retrieve data by scheduling a job, which typically completes within 3 to 5 hours.



Amazon Glacier integrates seamlessly with other AWS services such as Amazon S3 and the AWS storage and database services. Amazon S3 enables you to create lifecycle policies that will archive data to Glacier (and allow retrieval) automatically.[17]

Customers can integrate Amazon Glacier into their existing backup and

[15] http://aws.amazon.com/glacier
[16] http://aws.amazon.com/iam
[17] http://docs.aws.amazon.com/AmazonS3/latest/dev/object-archival.html

archive tools and processes such that it represents a new tier of storage useful for any data to be kept for long periods of time. Furthermore, if you have existing tape-based archives, you can migrate them to Amazon Glacier using the AWS Import/Export service[18] whereby physical devices can be shipped to AWS for direct ingestion into the relevant Amazon Glacier vaults.

Conclusion

The AWS platform provides new and more flexible options for infrastructure configuration that enable a far more efficient and cost-effective backup and recovery regime for enterprise customers. By evolving certain processes and procedures from current legacy approaches to the state-of-the-art "infrastructure as code" approach, you can achieve the correct level of backup and recovery for your applications while reducing backup infrastructure and complexity.

Further Reading

1. Backup and Storage Webpage - https://aws.amazon.com/backup-storage/
2. Step-by-step video series on how to backup your Oracle Databases to Amazon S3 using Oracle Secure Backup Cloud Module - https://aws.amazon.com/backup-storage/gsg-oracle-rman/

[18] http://aws.amazon.com/importexport