Amazon Elastic Compute Cloud (Amazon EC2) provides scalable computing capacity in the Amazon Web Services (AWS) cloud. Using Amazon EC2 eliminates your need to invest in hardware up front, so you can develop and deploy applications faster. You can use Amazon EC2 to launch as many or as few virtual servers as you need, configure security and networking, and manage storage. Amazon EC2 enables you to scale up or down to handle changes in requirements or spikes in popularity, reducing your need to forecast traffic.

## **Features of Amazon EC2**

Amazon EC2 provides the following features:

* Virtual computing environments, known as instances
* Preconfigured templates for your instances, known as Amazon Machine Images (AMIs), that package the bits you need for your server (including the operating system and additional software)
* Various configurations of CPU, memory, storage, and networking capacity for your instances, known as instance types
* Secure login information for your instances using key pairs (AWS stores the public key, and you store the private key in a secure place)
* Storage volumes for temporary data that's deleted when you stop or terminate your instance, known as instance store volumes
* Persistent storage volumes for your data using Amazon Elastic Block Store (Amazon EBS), known as Amazon EBS volumes
* Multiple physical locations for your resources, such as instances and Amazon EBS volumes, known as Regions and Availability Zones
* A firewall that enables you to specify the protocols, ports, and source IP ranges that can reach your instances using security groups
* Static IPv4 addresses for dynamic cloud computing, known as Elastic IP addresses
* Metadata, known as tags, that you can create and assign to your Amazon EC2 resources
* Virtual networks you can create that are logically isolated from the rest of the AWS cloud, and that you can optionally connect to your own network, known as virtual private clouds (VPCs)

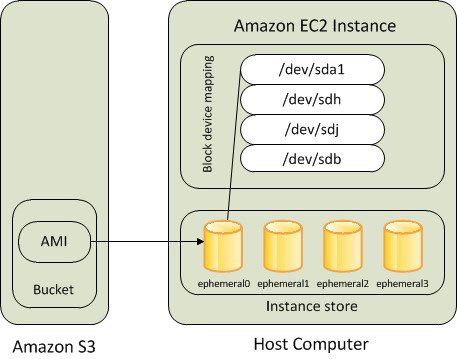
**EBS volume is network attached drive which results in slow performance but data is persistent meaning even if you reboot the instance data will be there.**

**Instance store is physically attached device which gives better performance but data will be lost once instance is rebooted.**

EC2 instances support two types for block level storage: EC2 Instances can be launched using either Elastic Block Store (EBS) or Instance Store volume as root volumes and additional volumes.

EC2 instances can be launched by choosing between AMIs backed by EC2 instance store and AMIs backed by EBS. However, AWS recommends use of EBS backed AMIs, because they launch faster and use persistent storage.

**Instance Store**



Instance Store

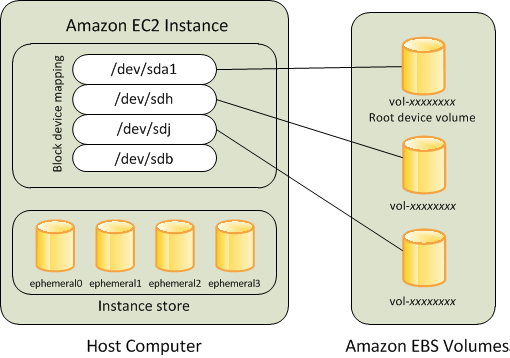
* Instance store backed instance is an EC2 instance using an Instance store as root device volume created from a template stored in S3.
* An instance store is ephemeral storage that provides temporary block level storage for your instance. Instance store is ideal for temporary storage like buffers, caches, and other temporary content.
* Instance store volumes accesses storage from disks that are physically attached to the host computer.
* When an Instance stored instance is launched, the image that is used to boot the instance is copied to the root volume (typically sda1).
* Instance store provides temporary block-level storage for instances.
* Data on an instance store volume persists only during the life of the associated instance; if an instance is stopped or terminated, any data on instance store volumes is lost.

## **Key points for Instance store backed Instance**

1. Boot time is slower then EBS backed volumes and usually less then 5 min
2. Can be selected as Root Volume and attached as additional volumes
3. Instance store backed Instances can be of maximum 10GiB volume size
4. Instance store volume can be attached as additional volumes only when the instance is being launched and cannot be attached once the Instance is up and running
5. Instance store backed Instances cannot be stopped, as when stopped and started AWS does not guarantee the instance would be launched in the same host and hence the data is lost
6. Data on Instance store volume is NOT LOST when the instance is rebooted
7. AMI creation requires usage on AMI tools and needs to be executed from within the running instance
8. Instance store backed Instances cannot be upgraded
9. For EC2 instance store-backed instances AWS recommends to:- 1. Distribute the data on the instance stores across multiple AZs 2. Back up critical data from the instance store volumes to persistent storage on a regular basis.
10. Data on Instance store volume is LOST in following scenarios :-

* Failure of an underlying drive
* Stopping an EBS-backed instance where instance store are attached as additional volumes
* Termination of the Instance

# Amazon Elastic Block Store (EBS)



* An “EBS-backed” instance means that the root device for an instance launched from the AMI is an EBS volume created from an EBS snapshot
* An EBS volume behaves like a raw, unformatted, external block device that can be **attached to a single instance** and are not physically attached to the Instance host computer (more like a network attached storage).
* Volume persists independently from the running life of an instance. After an EBS volume is attached to an instance, you can use it like any other physical hard drive.
* EBS volume can be detached from one instance and attached to another instance.
* EBS volumes can be created as encrypted volumes using the EBS encryption feature.
* EBS is block store which is separately attached to EC2. Also its design such a way that it will be replicated within its availability zone so it provides high availability and durability.
* And the additional advantage of it is, you can have back-ups for EBS by creating Snapshots which is not possible instance store. So that whenever you want to retrieve the data you can just create the EBS volume from the snapshot.

## **Key points for EBS backed Instance**

1. Boot time is very fast usually less then a min
2. Can be selected as Root Volume and attached as additional volumes
3. EBS backed Instances can be of maximum 16TiB volume size depending upon the OS
4. EBS volume can be attached as additional volumes when the Instance is launched and even when the Instance is up and running
5. When EBS-backed instance is in a stopped state, various instance– and volume-related tasks can be done for e.g. you can modify the properties of the instance, you can change the size of your instance or update the kernel it is using, or you can attach your root volume to a different running instance for debugging or any other purpose
6. EBS volumes are AZ scoped and tied to a single AZ in which created
7. EBS volumes are automatically replicated within that zone to prevent data loss due to failure of any single hardware component
8. AMI creation is easy using a Single command
9. EBS backed Instances can be upgraded for instance type, Kernel, RAM disk and user data
10. Data on the EBS volume is LOST :- 1. For EBS Root volume, if Delete on termination flag is enabled (enabled, by default) 2. For attached EBS volumes, if the Delete on termination flag is enabled (disabled, by default).
11. Data on EBS volume is NOT LOST in following scenarios :-

* Reboot on the Instance
* Stopping an EBS-backed instance
* Termination of the Instance for the additional EBS volumes. Additional EBS volumes are detached with their data intact

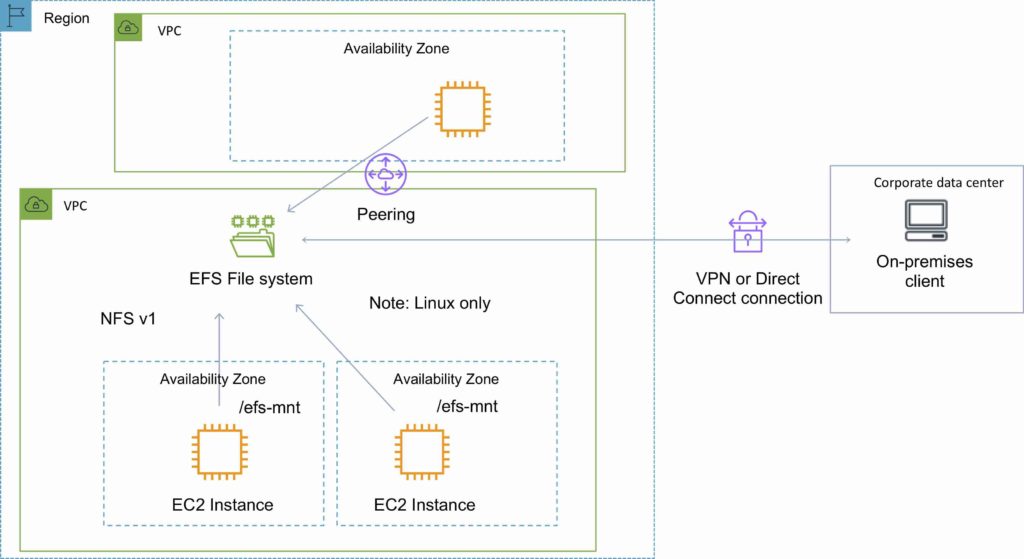
# Performance

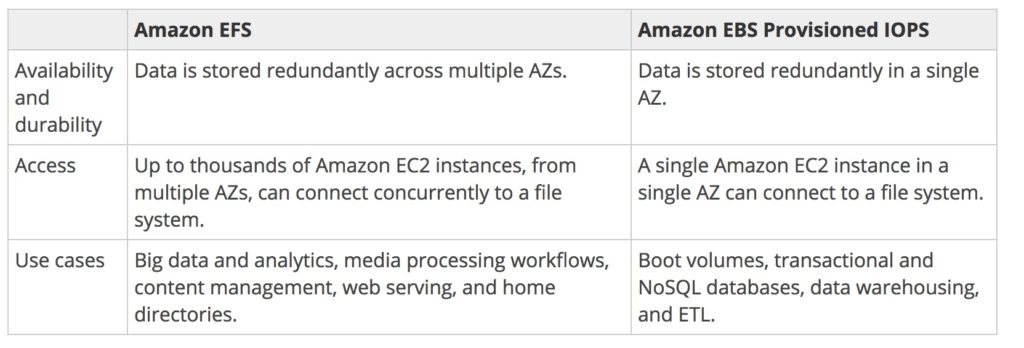
* EBS-backed AMIs launch faster than EC2 instance store-backed AMIs.
* When an EC2 instance store-backed AMI is launched, all the parts have to be retrieved from S3 before the instance is available.
* With an EBS-backed AMI is launched, parts are lazily loaded and only the parts required to boot the instance need to be retrieved from the snapshot before the instance is available.
* However, the performance of an instance that uses an EBS volume for its root device is slower for a short time while the remaining parts are retrieved from the snapshot and loaded into the volume.
* When you stop and restart the instance, it launches quickly, because the state is stored in an EBS volume.
* EBS volume must be in same AZ as the EC2 instance they are attached to.
* Each EBS volume is automatically replicated within availability zone to protect from component failure

## **EBS SNAPSHOTS**

* Snapshots capture a point-in-time state of an instance.
* Cost-effective and easy backup strategy.
* Share data sets with other users or accounts.
* Can be used to migrate a system to a new AZ or region.
* Can be used to convert an unencrypted volume to an encrypted volume.
* Snapshots are stored on Amazon S3.
* Does not provide granular backup (not a replacement for backup software).
* If you make periodic snapshots of a volume, the snapshots are incremental, which means that only the blocks on the device that have changed after your last snapshot are saved in the new snapshot.
* Even though snapshots are saved incrementally, the snapshot deletion process is designed so that you need to retain only the most recent snapshot in order to restore the volume.
* Snapshots can only be accessed through the EC2 APIs(CLI or Console).
* EBS volumes are AZ specific but snapshots are region specific.
* Volumes can be created from EBS snapshots that are the same size or larger.
* Snapshots can be taken of non-root EBS volumes while running.
* To take a consistent snapshots writes must be stopped (paused) until the snapshot is complete – if not possible the volume needs to be detached, or if it’s an EBS root volume the instance must be stopped.
* To lower storage costs on S3 a full snapshot and subsequent incremental updates can be created.
* You are charged for data traffic to S3 and storage costs on S3.
* You are billed only for the changed blocks.
* Deleting a snapshot removes only the data not needed by any other snapshot.
* You can resize volumes through restoring snapshots with different sizes (configured when taking the snapshot).
* Snapshots can be copied between regions (and be encrypted). Images are then created from the snapshot in the other region which creates an AMI that can be used to boot an instance.
* You can create volumes from snapshots and choose the availability zone within the region.
* Once you create the AMI using a snapshot ,that snapshot cannot be deleted because the AMI is backed by this snapshot. You have to delete AMI first then snapshot.
* Amazon EBS fast snapshot restore enables you to create a volume from a snapshot that is fully-initialized at creation. This eliminates the latency of I/O operations on a block when it is accessed for the first time. Volumes created using fast snapshot restore instantly deliver all of their provisioned performance.

**Amazon EFS**

* EFS is a fully-managed service that makes it easy to set up and scale file storage in the Amazon Cloud.
* Implementation of an NFS file share and is accessed using the NFSv4.1 protocol.
* Elastic storage capacity and pay for what you use (in contrast to EBS with which you pay for what you provision).
* Multi-AZ metadata and data storage.
* Can configure mount-points in one, or many, AZs.
* Can be mounted from on-premises systems ONLY if using Direct Connect or a VPN connection.
* Alternatively, use the EFS File Sync agent.
* Good for big data and analytics, media processing workflows, content management, web serving, home directories etc.
* Pay for what you use (no pre-provisioning required).
* Can scale up to petabytes.
* EFS is elastic and grows and shrinks as you add and remove data.
* Can concurrently connect 1 to 1000s of EC2 instances, from multiple AZs.
* A file system can be accessed concurrently from all AZs in the region where it is located.
* The following diagram depicts the various options for mounting an EFS filesystem:
* By default you can create up to 10 file systems per account.
* Access to EFS file systems from on-premises servers can be enabled via Direct Connect or AWS VPN.
* You mount an EFS file system on your on-premises Linux server using the standard Linux mount command for mounting a file system via the NFSv4.1 protocol.
* Can choose General Purpose or Max I/O (both SSD).
* The VPC of the connecting instance must have DNS hostnames enabled.
* EFS provides a file system interface, file system access semantics (such as strong consistency and file locking).
* Data is stored across multiple AZ’s within a region.
* Read after write consistency.
* Need to create mount targets and choose AZ’s to include (recommended to include all AZ’s).
* EFS is compatible with all Linux-based AMIs for Amazon EC2.
* Instances can be behind an ELB
* There are two performance modes:
  + “General Purpose” performance mode is appropriate for most file systems.
  + “Max I/O” performance mode is optimized for applications where tens, hundreds, or thousands of EC2 instances are accessing the file system.
* Amazon EFS is designed to burst to allow high throughput levels for periods of time.



|  |  |
| --- | --- |
| EC2 Instance Store Volume | Elastic block store(EBS) volume |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

Spot Instances

* Amazon EC2 Spot Instances let you take advantage of unused EC2 capacity in the AWS cloud.
* Spot Instances are available at up to a 90% discount compared to On-Demand prices.
* You can use Spot Instances for various stateless, fault-tolerant, or flexible applications such as big data, containerized workloads, CI/CD, web servers, high-performance computing (HPC), and other test & development workloads.
* Because Spot Instances are tightly integrated with AWS services such as Auto Scaling, EMR, ECS, CloudFormation, Data Pipeline and AWS Batch, you can choose how to launch and maintain your applications running on Spot Instances.
* Moreover, you can easily combine Spot Instances with On-Demand and RIs to further optimize workload cost with performance.
* Due to the operating scale of AWS, Spot Instances can offer the scale and cost savings to run hyper-scale workloads.
* You also have the option to hibernate, stop or terminate your Spot Instances when EC2 reclaims the capacity back with two-minutes of notice.
* Only on AWS, you have easy access to unused compute capacity at such massive scale - all at up to a 90% discount.

Why Spot Instances ?

### **Low, predictable prices**

You can purchase Spot Instances at prices up to 90% lower than On-Demand instances. Moreover, you can provision capacity across Spot, On-Demand, and RIs [using EC2 Auto Scaling](https://www.youtube.com/watch?v=mXX1dgmStlo&feature=youtu.be) to optimize workload cost with performance.

### **Massive scale**

Spot offers you the advantages of the massive operating scale of AWS. You can run hyperscale workloads at a significant cost savings or you can accelerate your workloads by running parallel tasks.

### **Easy to use**

It’s easy to launch, scale and manage Spot Instances through AWS services, like EC2 Auto Scaling and ECS, or integrated third parties, like Terraform and Jenkins.

**Spot Best Practices**

Your instance type requirements, budget requirements, and application design will determine how to apply the following best practices for your application:

• Be flexible about instance types.

Test your application on different instance types when possible. Because prices fluctuate independently for each instance type in an Availability Zone, you can often get more compute capacity for the same price when you have instance type flexibility. Request all instance types that meet your requirements to further reduce costs and improve application performance. Spot Fleets enable you to request multiple instance types simultaneously.

• Choose pools where prices haven't changed much. Because prices adjust based on long-term demand, popular instance types (such as recently launched instance families), tend to have more price adjustments. Therefore, picking older-generation instance types that are less popular tends to result in lower costs and fewer interruptions. Similarly, the same instance type can have different prices in different Availability Zones.

• Minimize the impact of interruptions. Amazon EC2 Spot's Hibernate feature allows you to pause and then resume Amazon EBS backed instances when capacity is available. Hibernate is just like closing and opening your laptop lid, with your application starting up right where it left off

# How Spot Fleet works

A Spot Fleet is a collection, or fleet, of Spot Instances, and optionally On-Demand Instances.

The Spot Fleet attempts to launch the number of Spot Instances and On-Demand Instances to meet the target capacity that you specified in the Spot Fleet request. The request for Spot Instances is fulfilled if there is available capacity and the maximum price you specified in the request exceeds the current Spot price. The Spot Fleet also attempts to maintain its target capacity fleet if your Spot Instances are interrupted.

You can also set a maximum amount per hour that you’re willing to pay for your fleet, and Spot Fleet launches instances until it reaches the maximum amount. When the maximum amount you're willing to pay is reached, the fleet stops launching instances even if it hasn’t met the target capacity.

A Spot Instance pool is a set of unused EC2 instances with the same instance type (for example, m5.large), operating system, Availability Zone, and network platform. When you make a Spot Fleet request, you can include multiple launch specifications, that vary by instance type, AMI, Availability Zone, or subnet. The Spot Fleet selects the Spot Instance pools that are used to fulfill the request, based on the launch specifications included in your Spot Fleet request, and the configuration of the Spot Fleet request. The Spot Instances come from the selected pools.

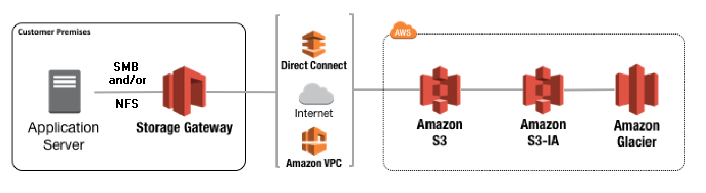
**AWS Storage Gateway** connects an on-premises software appliance with cloud-based storage to provide seamless integration with data security features between your on-premises IT environment and the AWS storage infrastructure. You can use the service to store data in the AWS Cloud for scalable and cost-effective storage that helps maintain data security.

Is a hybrid cloud storage service that gives you on-premises access to virtually unlimited cloud storage

AWS Storage Gateway offers file-based, volume-based, and tape-based storage solutions:

**File Gateway**

* A file gateway supports a file interface into Amazon Simple Storage Service (Amazon S3) and combines a service and a virtual software appliance. By using this combination, you can store and retrieve objects in Amazon S3 using industry-standard file protocols such as Network File System (NFS) and Server Message Block (SMB).
* The software appliance, or gateway, is deployed into your on-premises environment as a virtual machine (VM) running on VMware ESXi, Microsoft Hyper-V, or Linux Kernel-based Virtual Machine (KVM) hypervisor.
* The gateway provides access to objects in S3 as files or file share mount points. With a file gateway, you can do the following:
* You can store and retrieve files directly using the NFS version 3 or 4.1 protocol.
* You can store and retrieve files directly using the SMB file system version, 2 and 3 protocol.
* You can access your data directly in Amazon S3 from any AWS Cloud application or service.
* You can manage your Amazon S3 data using lifecycle policies, cross-region replication, and versioning. You can think of a file gateway as a file system mount on S3.



A file gateway simplifies file storage in Amazon S3, integrates to existing applications through industry-standard file system protocols, and provides a cost-effective alternative to on-premises storage.

It also provides low-latency access to data through transparent local caching.

A file gateway manages data transfer to and from AWS, buffers applications from network congestion, optimizes and streams data in parallel, and manages bandwidth consumption.

File gateways integrate with AWS services, for example with the following:

* Common access management using AWS Identity and Access Management (IAM)
* Encryption using AWS Key Management Service (AWS KMS)
* Monitoring using Amazon CloudWatch (CloudWatch)
* Audit using AWS CloudTrail (CloudTrail)
* Operations using the AWS Management Console and AWS Command Line Interface (AWS CLI)
* Billing and cost management

After the file gateway is activated, you create and configure your file share and associate that share with your Amazon S3 bucket.

Doing this makes the share accessible by clients using either the NFS or SMB protocol.

Files written to a file share become objects in Amazon S3, with the path as the key. There is a one-to-one mapping between files and objects, and the gateway asynchronously updates the objects in Amazon S3 as you change the files.

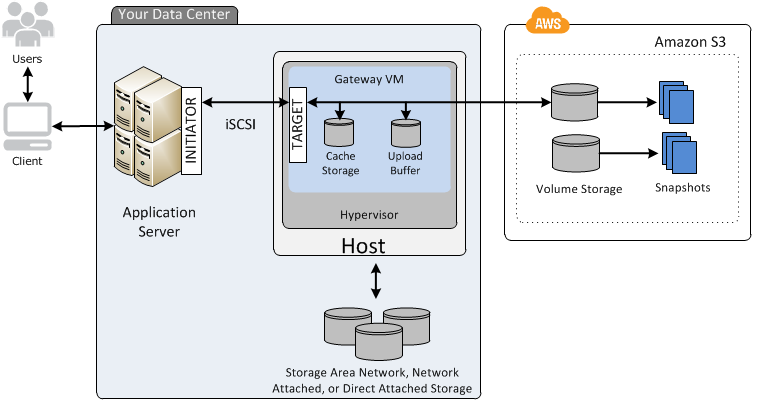
Existing objects in the bucket appear as files in the file system, and the key becomes the path.

Objects are encrypted with Amazon S3–server-side encryption keys (SSE-S3). All data transfer is done through HTTPS.

**Volume Gateway**

A volume gateway provides cloud-backed storage volumes that you can mount as Internet Small Computer System Interface (iSCSI) devices from your on-premises application servers.

The volume gateway is deployed into your on-premises environment as a VM running on VMware ESXi, KVM, or Microsoft Hyper-V hypervisor.

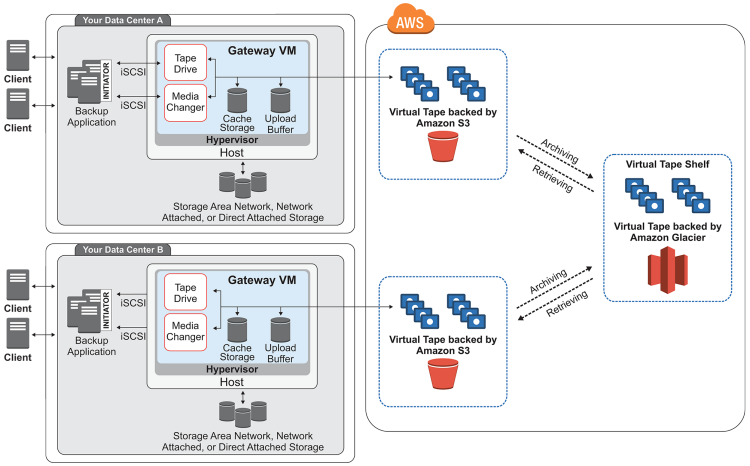


The gateway supports the following volume configurations:

**Cached volumes** –

* You store your data in Amazon Simple Storage Service (Amazon S3) and retain a copy of frequently accessed data subsets locally.
* Cached volumes offer a substantial cost savings on primary storage and minimize the need to scale your storage on-premises.
* You also retain low-latency access to your frequently accessed data.
* **Upload buffer disk** − This type of storage disk is used to store the data before it is uploaded to Amazon S3 over SSL connection. The storage gateway uploads the data from the upload buffer over an SSL connection to AWS.
* **Snapshots** − Sometimes we need to back up storage volumes in Amazon S3. These backups are incremental and are known as **snapshots**. The snapshots are stored in Amazon S3 as Amazon EBS snapshots. Incremental backup means that a new snapshot is backing up only the data that has changed since the last snapshot. We can take snapshots either at a scheduled interval or as per the requirement.

**Stored volumes** –

* If you need low-latency access to your entire dataset, first configure your on-premises gateway to store all your data locally.
* Then asynchronously back up point-in-time snapshots of this data to Amazon S3. This configuration provides durable and inexpensive offsite backups that you can recover to your local data center or Amazon Elastic Compute Cloud (Amazon EC2).
* For example, if you need replacement capacity for disaster recovery, you can recover the backups to Amazon EC2.
* 
* The diagram identifies the following tape gateway components:

**Tape Gateway**

A tape gateway provides cloud-backed virtual tape storage.

The tape gateway is deployed into your on-premises environment as a VM running on VMware ESXi, KVM, or Microsoft Hyper-V hypervisor.

With a tape gateway, you can cost-effectively and durably archive backup data in GLACIER or DEEP\_ARCHIVE.

A tape gateway provides a virtual tape infrastructure that scales seamlessly with your business needs and eliminates the operational burden of provisioning, scaling, and maintaining a physical tape infrastructure.

You can run AWS Storage Gateway either on-premises as a VM appliance, as a hardware appliance, or in AWS as an Amazon EC2 instance.

You deploy your gateway on an EC2 instance to provision iSCSI storage volumes in AWS. You can use gateways hosted on EC2 instances for disaster recovery, data mirroring, and providing storage for applications hosted on Amazon EC2.

Tape Gateway offers a durable, cost-effective solution to archive your data in the AWS Cloud. With its virtual tape library (VTL) interface, you use your existing tape-based backup infrastructure to store data on virtual tape cartridges that you create on **your tape gateway**. Each tape gateway is preconfigured with a media changer and tape drives. These are available to your existing client backup applications as iSCSI devices. You add tape cartridges as you need to archive your data.

* **Virtual tape** – A virtual tape is like a physical tape cartridge. However, virtual tape data is stored in the AWS Cloud. Like physical tapes, virtual tapes can be blank or can have data written on them. You can create virtual tapes either by using the Storage Gateway console or programmatically by using the Storage Gateway API. Each gateway can contain up to 1,500 tapes or up to 1 PiB of total tape data at a time. The size of each virtual tape, which you can configure when you create the tape, is between 100 GiB and 5 TiB.
* **Virtual tape library (VTL)** – A VTL is like a physical tape library available on-premises with robotic arms and tape drives. Your VTL includes the collection of stored virtual tapes. Each tape gateway comes with one VTL.

The virtual tapes that you create appear in your gateway's VTL. Tapes in the VTL are backed up by Amazon S3. As your backup software writes data to the gateway, the gateway stores data locally and then asynchronously uploads it to virtual tapes in your VTL—that is, Amazon S3.

* + **Tape drive** – A VTL tape drive is analogous to a physical tape drive that can perform I/O and seek operations on a tape. Each VTL comes with a set of 10 tape drives, which are available to your backup application as iSCSI devices.
  + **Media changer** – A VTL media changer is analogous to a robot that moves tapes around in a physical tape library's storage slots and tape drives. Each VTL comes with one media changer, which is available to your backup application as an iSCSI device.
* **Archive** – Archive is analogous to an offsite tape holding facility. You can archive tapes from your gateway's VTL to the archive. If needed, you can retrieve tapes from the archive back to your gateway's VTL.
  + **Archiving tapes** – When your backup software ejects a tape, your gateway moves the tape to the archive for long-term storage. The archive is located in the AWS Region in which you activated the gateway. Tapes in the archive are stored in the virtual tape shelf (VTS). The VTS is backed by [S3 Glacier](https://docs.aws.amazon.com/amazonglacier/latest/dev/introduction.html) or [S3 Glacier Deep Archive](https://docs.aws.amazon.com/amazonglacier/latest/dev/introduction.html), low-cost storage service for data archiving, backup, and long-term data retention.
  + **Retrieving tapes** – You can't read archived tapes directly. To read an archived tape, you must first retrieve it to your tape gateway by using either the Storage Gateway console or the Storage Gateway API.

AWS provides the service AWS Secrets Manager for easier management of secrets. Secrets can be database credentials, passwords, third-party API keys, and even arbitrary text. You can store and control access to these secrets centrally by using the Secrets Manager console, the Secrets Manager command line interface (CLI), or the Secrets Manager API and SDKs.

In the past, when you created a custom application to retrieve information from a database, you typically embedded the credentials,the secret, for accessing the database directly in the application. When the time came to rotate the credentials, you had to do more than just create new credentials. You had to invest time to update the application to use the new credentials. Then you distributed the updated application. If you had multiple applications with shared credentials and you missed updating one of them, the application failed. Because of this risk, many customers have chosen not to regularly rotate credentials, which effectively substitutes one risk for another.

Secrets Manager enables you to replace hardcoded credentials in your code, including passwords, with an API call to Secrets Manager to retrieve the secret programmatically. This helps ensure the secret can't be compromised by someone examining your code, because the secret no longer exists in the code. Also, you can configure Secrets Manager to automatically rotate the secret for you according to a specified schedule. This enables you to replace long-term secrets with short-term ones, significantly reducing the risk of compromise.

## **Getting Started with Secrets Manager**

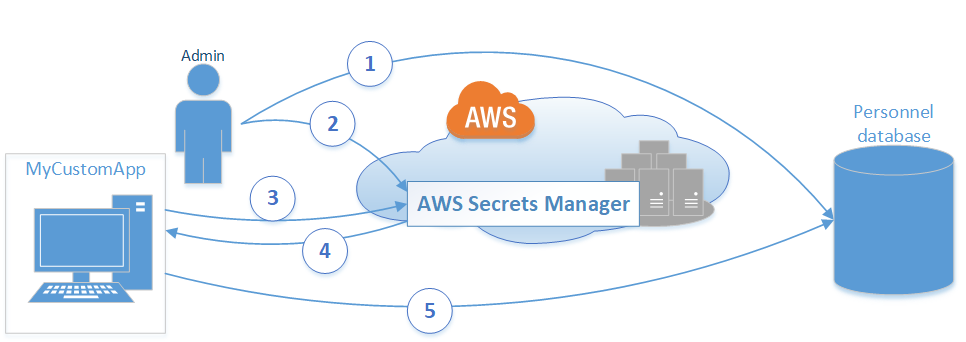
For a list of terms and concepts you need to understand to make full use of Secrets Manager, see [Key Terms and Concepts for AWS Secrets Manager](https://docs.aws.amazon.com/secretsmanager/latest/userguide/terms-concepts.html).

Typical users of Secrets Manager can have one or more of the following roles:

* Secrets Manager administrator – Administers the Secrets Manager service. Grants permissions to individuals who can then perform the other roles listed here.
* Database or service administrator – Administers the database or service with secrets stored in Secrets Manager. Determines and configures the rotation and expiration settings for their secrets.
* Application developer – Creates the application, and then configures the application to request the appropriate credentials from Secrets Manager.

## **Basic Secrets Manager Scenario**

The following diagram illustrates the most basic scenario. The diagram displays you can store credentials for a database in Secrets Manager, and then use those credentials in an application to access the database.



1. The database administrator creates a set of credentials on the Personnel database for use by an application called MyCustomApp. The administrator also configures those credentials with the permissions required for the application to access the Personnel database.
2. The database administrator stores the credentials as a secret in Secrets Manager named *MyCustomAppCreds*. Then, Secrets Manager encrypts and stores the credentials within the secret as the protected secret text.
3. When MyCustomApp accesses the database, the application queries Secrets Manager for the secret named *MyCustomAppCreds*.
4. Secrets Manager retrieves the secret, decrypts the protected secret text, and returns the secret to the client app over a secured (HTTPS with TLS) channel.
5. The client application parses the credentials, connection string, and any other required information from the response and then uses the information to access the database server.

**Note:**

Secrets Manager supports many types of secrets. However, Secrets Manager can natively rotate credentials for [supported AWS databases](https://docs.aws.amazon.com/secretsmanager/latest/userguide/intro.html#full-rotation-support) without any additional programming. However, rotating the secrets for other databases or services requires creating a custom Lambda function to define how Secrets Manager interacts with the database or service. You need some programming skill to create the function.

When storing credentials, different secured services might require different pieces of information. Secrets Manager provides this flexibility by storing the secret as key-value pairs of text strings. If you choose a database supported by Secrets Manager , Secrets Manager defines the key-value pairs according to the requirements of the rotation function for the chosen database. Secrets Manager formats the pairs as [JSON](http://json.org/) text. If you choose some other service or database Secrets Manager doesn't provide the Lambda function for, then you can specify your secret as a user-defined JSON key-value pairs.

**Security in amazon VPC.**

**Security Groups:**

* A *security group* acts as a virtual firewall for your instance to control inbound and outbound traffic.
* When you launch an instance in a VPC, you can assign up to five security groups to the instance.
* Security groups act at the instance level, not the subnet level. Therefore, each instance in a subnet in your VPC can be assigned to a different set of security groups.
* If you launch an instance using the Amazon EC2 API or a command line tool and you don't specify a security group, the instance is automatically assigned to the default security group for the VPC.
* If you launch an instance using the Amazon EC2 console, you have an option to create a new security group for the instance.
* For each security group, you add *rules* that control the inbound traffic to instances, and a separate set of rules that control the outbound traffic.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.

## **Security group basics**

## The following are the basic characteristics of security groups for your VPC:

* There are quotas on the number of security groups that you can create per VPC, the number of rules that you can add to each security group, and the number of security groups that you can associate with a network interface. For more information, see [Amazon VPC quotas](https://docs.aws.amazon.com/vpc/latest/userguide/amazon-vpc-limits.html).
* You can specify allow rules, but not deny rules.
* You can specify separate rules for inbound and outbound traffic.
* When you create a security group, it has no inbound rules. Therefore, no inbound traffic originating from another host to your instance is allowed until you add inbound rules to the security group.
* By default, a security group includes an outbound rule that allows all outbound traffic. You can remove the rule and add outbound rules that allow specific outbound traffic only. If your security group has no outbound rules, no outbound traffic originating from your instance is allowed.
* Security groups are stateful — if you send a request from your instance, the response traffic for that request is allowed to flow in regardless of inbound security group rules. Responses to allowed inbound traffic are allowed to flow out, regardless of outbound rules.
* Instances associated with a security group can't talk to each other unless you add rules allowing the traffic (exception: the default security group has these rules by default).
* Security groups are associated with network interfaces. After you launch an instance, you can change the security groups that are associated with the instance, which changes the security groups associated with the primary network interface (eth0). You can also specify or change the security groups associated with any other network interface. By default, when you create a network interface, it's associated with the default security group for the VPC, unless you specify a different security group. For more information about network interfaces, see [Elastic network interfaces](https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/using-eni.html).
* When you create a security group, you must provide it with a name and a description. The following rules apply:
  + Names and descriptions can be up to 255 characters in length.
  + Names and descriptions are limited to the following characters: a-z, A-Z, 0-9, spaces, and .\_-:/()#,@[]+=&;{}!$\*.
  + A security group name cannot start with sg- as these indicate a default security group.
  + A security group name must be unique within the VPC.

**AWS VPN**

Establish a secure and private encrypted tunnel from your network or device to the AWS global network.

AWS VPN is comprised of two services: AWS Site-to-Site VPN and AWS Client VPN.

**AWS Client VPN** is a fully-managed, elastic VPN service that automatically scales up or down the number of available Client VPN connections based on user demand.

Because it runs in the cloud, you don’t need to install and manage either a hardware or software VPN solution and you don’t need to over-provision for peak demand.

AWS Client VPN is designed so your employees can access any company resource, both in AWS and on premises, from any location.

Client VPN also provides quick and easy connectivity to your workforce and business partners using OpenVPN-enabled devices such as Mac, Windows, iOS, Android, and Linux.

A free AWS VPN client is available for AWS Client VPN. It provides an end-to-end VPN encryption experience that goes from user devices to AWS and on-premises networks.

**AWS Site-to-Site VPN**

By default, instances that you launch into an Amazon VPC cannot communicate with your on-premises network. You can securely extend your data center or branch office network to the cloud with an AWS Site-to-Site VPN (Site-to-Site VPN) connection. It uses internet protocol security (IPSec) communications to create encrypted VPN tunnels between two locations.

You also have the option to use Accelerated Site-to-Site VPN, which is built for cross-continent applications using AWS Global Accelerator. It provides higher performance and consistency, particularly for your business-critical applications that span continents.

**Network ACLs**

A network access control list (ACL) is an optional layer of security for your VPC that acts as a firewall for controlling traffic in and out of one or more subnets. 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. For more information about the differences between security groups and network ACLs, see [Comparison of security groups and network ACLs](https://docs.aws.amazon.com/vpc/latest/userguide/VPC_Security.html#VPC_Security_Comparison).

The following are the basic things that you need to know about network ACLs:

* Your VPC automatically comes with a modifiable default network ACL. By default, it allows all inbound and outbound IPv4 traffic and, if applicable, IPv6 traffic.
* You can create a custom network ACL and associate it with a subnet. By default, each custom network ACL denies all inbound and outbound traffic until you add rules.
* Each subnet in your VPC 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.
* You can associate a network ACL with multiple subnets. However, a subnet can be associated with only one network ACL at a time. When you associate a network ACL with a subnet, the previous association is removed.
* A network ACL contains a numbered list of rules. We evaluate the rules 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 recommend that you start by creating rules in increments (for example, increments of 10 or 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.
* Network ACLs are stateless, which means that responses to allowed inbound traffic are subject to the rules for outbound traffic (and vice versa).

## **Network ACL rules**

You can add or remove rules from the default network ACL, or create additional network ACLs for your VPC. When you add or remove rules from a network ACL, the changes are automatically applied to the subnets that it's associated with.

The following are the parts of a network ACL rule:

* **Rule number**. **Rules are evaluated starting with the lowest numbered rule. As soon as a rule matches traffic, it's applied regardless of any higher-numbered rule that might contradict it.**
* **Type**. The type of traffic; for example, SSH. You can also specify all traffic or a custom range.
* **Protocol**. You can specify any protocol that has a standard protocol number. For more information, see [Protocol Numbers](http://www.iana.org/assignments/protocol-numbers/protocol-numbers.xhtml). If you specify ICMP as the protocol, you can specify any or all of the ICMP types and codes.
* **Port range**. The listening port or port range for the traffic. For example, 80 for HTTP traffic.
* **Source**. [Inbound rules only] The source of the traffic (CIDR range).
* **Destination**. [Outbound rules only] The destination for the traffic (CIDR range).
* **Allow/Deny**. Whether to allow or deny the specified traffic.

## **Default network ACL**

The default network ACL is configured to allow all traffic to flow in and out of the subnets with which it is associated. Each network ACL also includes a rule whose rule number is an asterisk. This rule ensures that if a packet doesn't match any of the other numbered rules, it's denied. You can't modify or remove this rule.

The following is an example default network ACL for a VPC that supports IPv4 only.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Inbound** | | | | | |
| **Rule #** | **Type** | **Protocol** | **Port range** | **Source** | **Allow/Deny** |
| 100 | All IPv4 traffic | All | All | 0.0.0.0/0 | ALLOW |
| \* | All IPv4 traffic | All | All | 0.0.0.0/0 | DENY |
| **Outbound** | | | | | |
| **Rule #** | **Type** | **Protocol** | **Port range** | **Destination** | **Allow/Deny** |
| 100 | All IPv4 traffic | All | All | 0.0.0.0/0 | ALLOW |
| \* | All IPv4 traffic | All | All | 0.0.0.0/0 | DENY |

If you create a VPC with an IPv6 CIDR block or if you associate an IPv6 CIDR block with your existing VPC, we automatically add rules that allow all IPv6 traffic to flow in and out of your subnet. We also add rules whose rule numbers are an asterisk that ensures that a packet is denied if it doesn't match any of the other numbered rules. You can't modify or remove these rules. The following is an example default network ACL for a VPC that supports IPv4 and IPv6.

**Note**

If you've modified your default network ACL's inbound rules, we do not automatically add an allow rule for inbound IPv6 traffic when you associate an IPv6 block with your VPC. Similarly, if you've modified the outbound rules, we do not automatically add an allow rule for outbound IPv6 traffic.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Inbound** | | | | | |
| **Rule #** | **Type** | **Protocol** | **Port range** | **Source** | **Allow/Deny** |
| 100 | All IPv4 traffic | All | All | 0.0.0.0/0 | ALLOW |
| 101 | All IPv6 traffic | All | All | ::/0 | ALLOW |
| \* | All traffic | All | All | 0.0.0.0/0 | DENY |
| \* | All IPv6 traffic | All | All | ::/0 | DENY |
| **Outbound** | | | | | |
| **Rule #** | **Type** | **Protocol** | **Port range** | **Destination** | **Allow/Deny** |
| 100 | All traffic | All | All | 0.0.0.0/0 | ALLOW |
| 101 | All IPv6 traffic | All | All | ::/0 | ALLOW |
| \* | All traffic | All | All | 0.0.0.0/0 | DENY |
| \* | All IPv6 traffic | All | All | ::/0 | DENY |

## **Custom network ACLs and other AWS services**

If you create a custom network ACL, be aware of how it might affect resources that you create using other AWS services.

With Elastic Load Balancing, if the subnet for your backend instances has a network ACL in which you've added a deny rule for all traffic with a source of either 0.0.0.0/0 or the subnet's CIDR, your load balancer can't carry out health checks on the instances.

**What is Eventual Consistency?**

Consistency is a key concept in data storage: it describes when changes committed to a system are visible to all participants. Classic transactional databases employ various levels of consistency, but the golden standard is that after a transaction commits the changes are guaranteed to be visible to all participants. A change committed at millisecond 1 is guaranteed to be available to all views of the system – all queries – immediately thereafter.

Eventual consistency relaxes the rules a bit, allowing a time lag between the point the data is committed to storage and the point where it is visible to all others. A change committed at millisecond 1 *might* be visible to all immediately. It might not be visible to all until millisecond 500. It might not even be visible to all until millisecond 1000. But, *eventually* it will be visible to all clients. Eventual consistency is a key engineering tradeoff employed in [building distributed systems](http://www.allthingsdistributed.com/2007/12/eventually_consistent.html).

One issue with eventual consistency is that there’s no theoretical limit to how long you need to wait until all clients see the committed data. A delay must be employed (either explicitly or implicitly) to ensure the changes will be visible to all clients.

**What is Read-After-Write Consistency?**

Read-after-write consistency tightens things up a bit, guaranteeing immediate visibility of *new* data to all clients.

With read-after-write consistency, a newly created object or file or table row will immediately be visible, without any delays.

Note that read-after-write is not complete consistency: there’s also read-after-update and read-after-delete.

Read-after-update consistency would allow edits to an existing file or changes to an already-existing object or updates of an existing table row to be immediately visible to all clients.

That’s not the same thing as read-after-write, which is only for new data.

Read-after-delete would guarantee that reading a deleted object or file or table row will fail for all clients, immediately.

That, too, is different from read-after-write, which only relates to the creation of data.

## **Amazon S3 Consistency Models**

[Amazon S3](https://cloudacademy.com/amazon-web-services/labs/create-your-first-amazon-s3-bucket-2/)**provides read-after-write consistency for PUTS of new objects in your S3 bucket and eventual consistency for overwrite PUTS and DELETES in all regions**.  So, if you add a new object to your bucket, you and your clients will see it. But, if you overwrite an object, it might take some time to update its replicas – hence the eventual consistency model is applied.

Amazon S3 guarantees high-availability by replicating data across many servers and AZs. It is obvious that data integrity should be maintained if a new record is added or a record/data is updated and deleted. The scenarios for above cases are as follows:

* **A new PUT request is made**. The object might not appear in the list if queried immediately until the changes are propagated to all the servers and AZs. The read-after-write consistency model is applied here.
* **An UPDATE request is made**. As eventual consistency model is applied for UPDATEs, a query to list the object might return an old value.
* **A DELETE request is made**. As eventual consistency model is applied for DELETEs, a query to list or read the object might return the deleted object.

**5 Pilliars of well-architectured framework:**

The [AWS Well-Architected Framework](https://aws.amazon.com/architecture/well-architected/) helps cloud architects build the most secure, high-performing, resilient, and efficient infrastructure possible for their applications. This framework provides a consistent approach for customers and [AWS Partner Network](https://aws.amazon.com/partners/) (APN) Partners to evaluate architectures, and provides guidance to implement designs that scale with your application needs over time.

Opertaion Excellence:

The operational excellence pillar includes the ability to run and monitor systems to deliver business value and to continually improve supporting processes and procedures.

There are six design principles for operational excellence in the cloud:

* Perform operations as code
* Annotate documentation
* Make frequent, small, reversible changes
* Refine operations procedures frequently
* Anticipate failure
* Learn from all operational failures

### Best Practices

Operations teams need to understand their business and customer needs so they can support business outcomes.

Ops creates and uses procedures to respond to operational events, and validates their effectiveness to support business needs. Ops also collects metrics that are used to measure the achievement of desired business outcomes.

Everything continues to change—your business context, business priorities, customer needs, etc.

It’s important to design operations to support evolution over time in response to change and to incorporate lessons learned through their performance.

Security:

The security pillar includes the ability to protect information, systems, and assets while delivering business value through risk assessments and mitigation strategies.

There are six design principles for security in the cloud:

* Implement a strong identity foundation
* Enable traceability
* Apply security at all layers
* Automate security best practices
* Protect data in transit and at rest
* Prepare for security events

Best Practices:

Before you architect any system, you need to put in place practices that influence security.

You will want to control who can do what. In addition, you want to be able to identify security incidents, protect your systems and services, and maintain the confidentiality and integrity of data through data protection.

You should have a well-defined and practiced process for responding to security incidents.

These tools and techniques are important because they support objectives such as preventing financial loss or complying with regulatory obligations.

The [AWS Shared Responsibility Model](https://aws.amazon.com/compliance/shared-responsibility-model/) enables organizations to achieve security and compliance goals. Because AWS physically secures the infrastructure that supports our cloud services, you can focus on using services to accomplish your goals.

Reliability Pillar:

The reliability pillar includes the ability of the system to **recover from infrastructure and service disruption**, dynamically **acquire computing resource** to meet the system needs and mitigate disruptions such as misconfiguration and transient network

There are five design principles for reliability in the cloud:

* Automatically recover from failure
* Stop guessing capacity
* Scale horizontally to increase aggregate system availability
* Test recovery procedures
* Manage change in automation.

Best Practices:

To achieve reliability, a system must have a well-planned foundation and monitoring in place, with mechanisms for handling changes in demand or requirements.

The system should be designed to detect failure and automatically heal itself.

Before architecting any system, foundational requirements that influence reliability should be in place. For example, you must have sufficient network bandwidth to your data center. These requirements are sometimes neglected (because they are beyond a single project’s scope). This neglect can have a significant impact on the ability to deliver a reliable system. In an on-premises environment, these requirements can cause long lead times due to dependencies and therefore must be incorporated during initial planning

With AWS, most of these foundational requirements are already incorporated or may be addressed as needed. The cloud is designed to be essentially limitless, so it is the responsibility of AWS to satisfy the requirement for sufficient networking and compute capacity, while you are free to change resource size and allocation, such as the size of storage devices, on demand.

Performance Efficiency:

This pillar includes the ability **to use the computing resources efficiently** to meet the system requirements and to **maintain that efficiency** as demand changes and technologies evolve.

Five design principles

* Democratize advance technologies.
* Use Serverless architectures
* Go global in minutes.
* Mechanical Sympathy
* Experiments more often.

Best Practices:

Take a data-driven approach to selecting a high-performance architecture. Gather data on all aspects of the architecture, from the high-level design to the selection and configuration of resource types.

By reviewing your choices on a cyclical basis, you will ensure you are taking advantage of the continually evolving AWS cloud. Monitoring will ensure you are aware of any deviance from expected performance and can take action on it.

Finally, your architecture can make tradeoffs to improve performance, such as using compression or caching, or relaxing consistency requirements.  
The optimal solution for a particular system will vary based on the kind of workload you have, often with multiple approaches combined. Well-architected systems use multiple solutions and enable different features to improve performance.

Cost Optimization:

This pillar includes the ability **to avoid or eliminate the unneeded** cost and suboptimal resources.

* Adopt consumption model
* Measure overall efficiency
* Stop spending money on data center operations
* Analyze and attribute expenditure.
* Use managed services to reduce the cost of ownership.

Design decisions are sometimes guided by haste as opposed to empirical data, as the temptation always exists to overcompensate “just in case” rather than spend time benchmarking for the most cost-optimal deployment.

This often leads to drastically over-provisioned and under-optimized deployments.

Using the appropriate instances and resources for your system is key to cost savings. For example, a reporting process might take five hours to run on a smaller server but one hour to run on a larger server that is twice as expensive. Both servers give you the same outcome, but the smaller one will incur more cost over time. A well-architected system will use the most cost-effective resources, which can have a significant and positive economic impact.