4. Think Parallel

- **Serial** and **Sequential** is now history
- The cloud is designed to handle massively parallel operations when it comes to accessing (retrieving and storing) data: **leverage request parallelization**
- **Multi-threading** your requests by using multiple concurrent threads
- The processes of a cloud application should be made **thread-safe** through a **share-nothing** philosophy
- **Distribute** the incoming requests across **multiple asynchronous** web servers using **load balancer**

5. Leverage Storage Options

- In the cloud, you are **paying for bandwidth** in and out of the cloud
- Transfer and the cost can add up very quickly.
- Keep **dynamic data** closer to the compute element
- Keep **static data** closer to the end-user
- If a large quantity of data that needs to be processed resides outside of the cloud, use **Sneakernet :-)**
- If the data is static and not going to change often (for example, images, video, audio, PDFs, JS, CSS files), it is advisable to take advantage of a **content delivery service** so that the static data is cached at an edge location closer to the end-user (requester) thereby lowering the access latency.

5. AWS Tactics

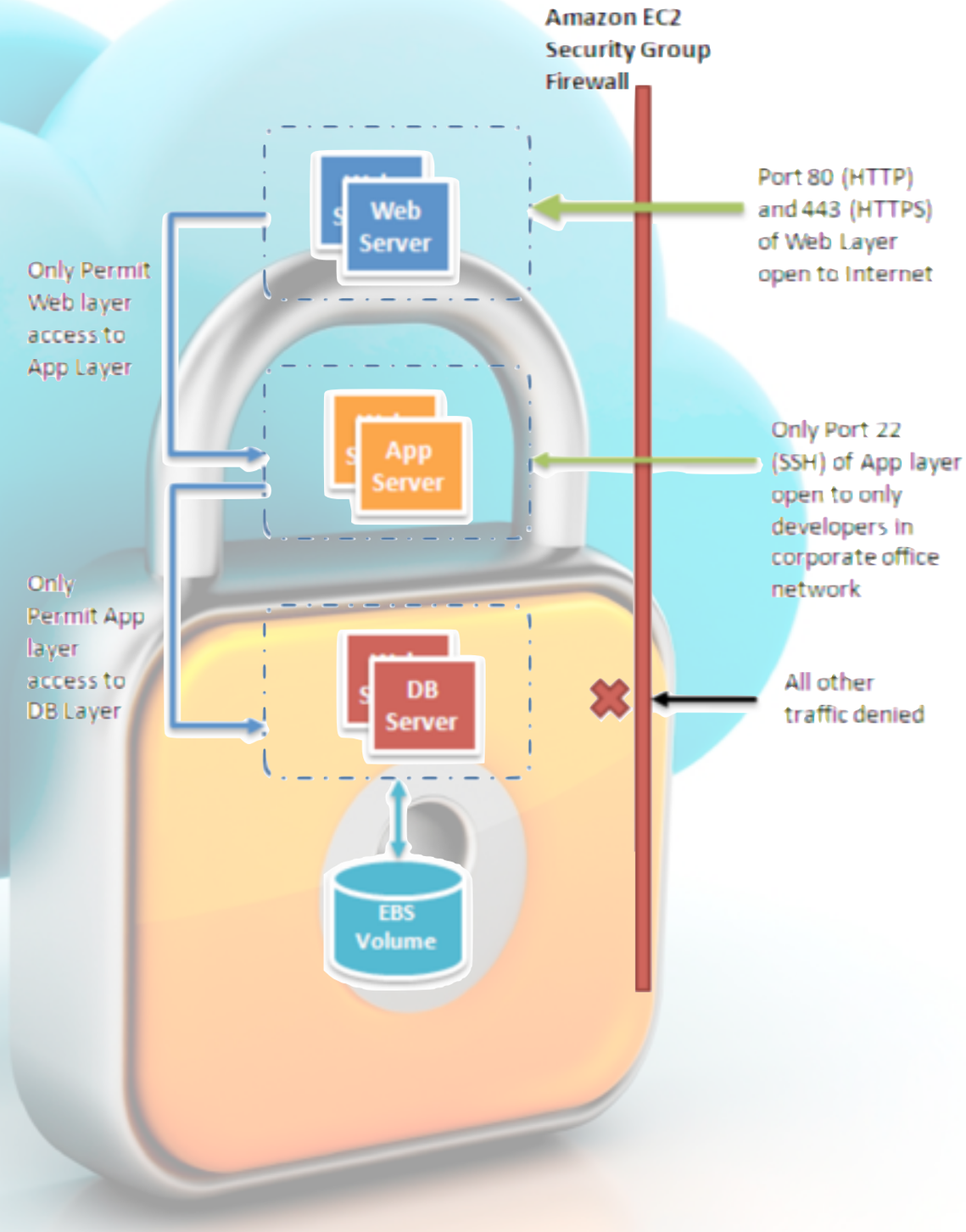
	Amazon S3 + CF	Amazon EC2 Ephemeral Store	Amazon EBS	Amazon SimpleDB	Amazon RDS
Ideal for	Storing Large write-once, read-many types of objects, Static Content Distribution	Storing non-persistent transient updates	Off-instance persistent storage for any kind of data,	Querying light-weight attribute data	Storing and querying structured Relational and referential Data
Ideal examples	Media files, audio, video, images, Backups, archives, versioning	Config Data, scratch files, TempDB	Clusters, boot data, Log or data of commercial RDBMS like Oracle, DB2	Querying, Mapping, tagging, click-stream logs, metadata, shared-state management, indexing	Complex transactional systems, inventory management and order fulfillment systems
Not recommended for	Querying, Searching	Storing Database logs or backups, customer data		Relational (joins) query	
Not recommended examples	Database, File Systems	Sensitive data	Content Distribution	OLTP, DW cube rollups	Simple lookups

6. Security

- In the cloud, **security** should be implemented **in every layer** of the cloud application architecture
- **Physical security** is typically handled by your service provider
- **Network** and **application-level security** is your responsibility
- Protect your data **in transit**
- Protect your data **at rest**
- Protect your **AWS credentials**
- Manage multiple Users and their permissions with **IAM**

6. AWS Tactics

- Every Amazon EC2 instance is protected by one or more **security groups**
- **Named sets of rules** that specify which ingress (i.e., incoming) network traffic should be delivered to your instance.
- You can specify TCP and UDP ports, ICMP types and codes, and source addresses.
- Security groups give you basic **firewall-like protection** for running instances.



Web Application Hosting Example

Traditional Architecture

Exterior Firewall
Hardware or Software Solution to open standard ports (80, 443)

Web Load Balancer
Hardware or Software solution to distribute traffic over web servers

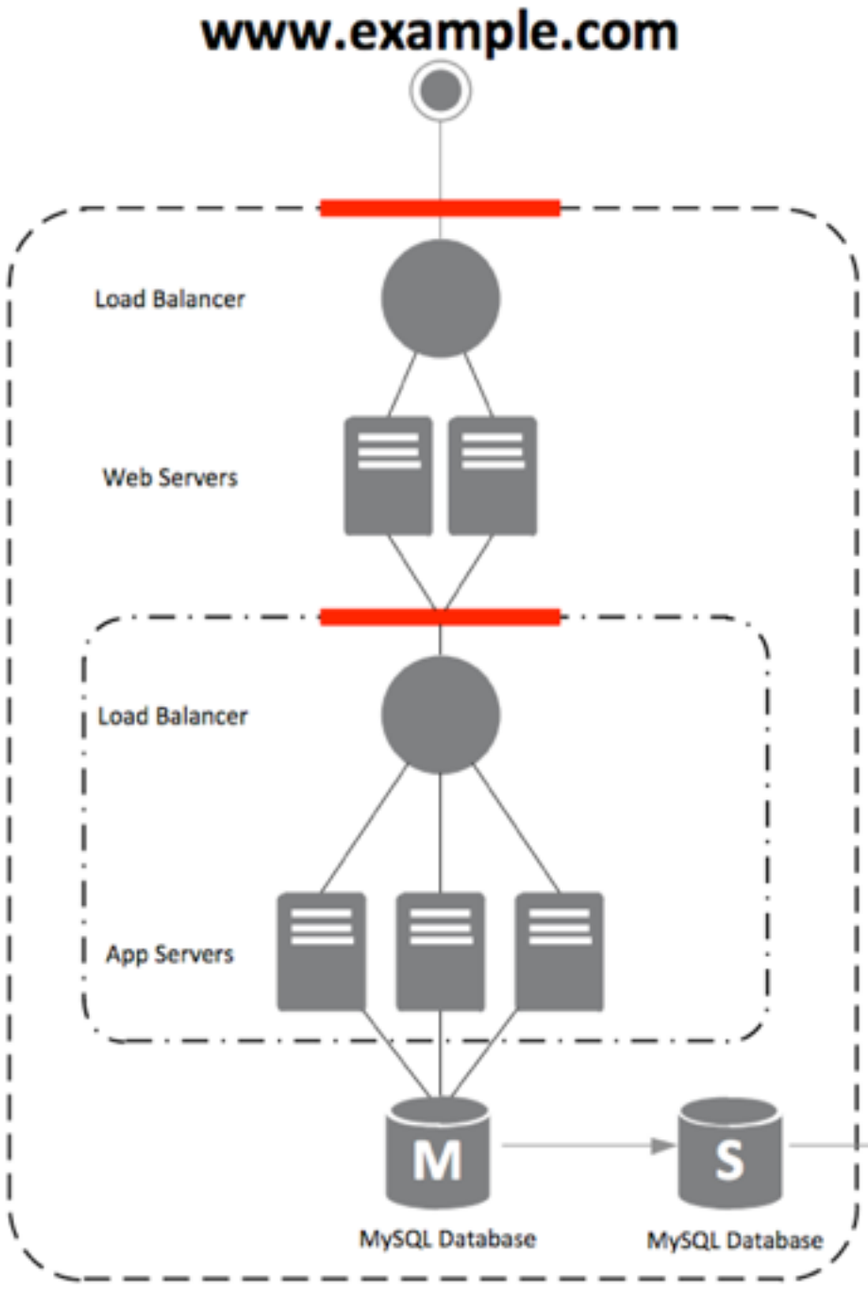
Web Tier
Fleet of machines handling HTTP requests

Backend Firewall limits access to application tied from web tier

App Load Balancer
Hardware or Software solution to spread traffic over app servers

App Server Tier
Fleet of machines handling Application specific workloads
Caching server machines can be implemented at this layer

Data Tier
Database Server machines with master and local running separately, Network storage for static objects



taken from: http://media.amazonwebservices.com/AWS_Web_Hosting_Best_Practices.pdf

Web Application Hosting Example

Amazon WS Architecture

Route 53
Provides DNS services to simplify domain management and zone APEX support (http://example.com)

Elastic Load Balancer
ELB to spread traffic to Web Server Autoscaling groups

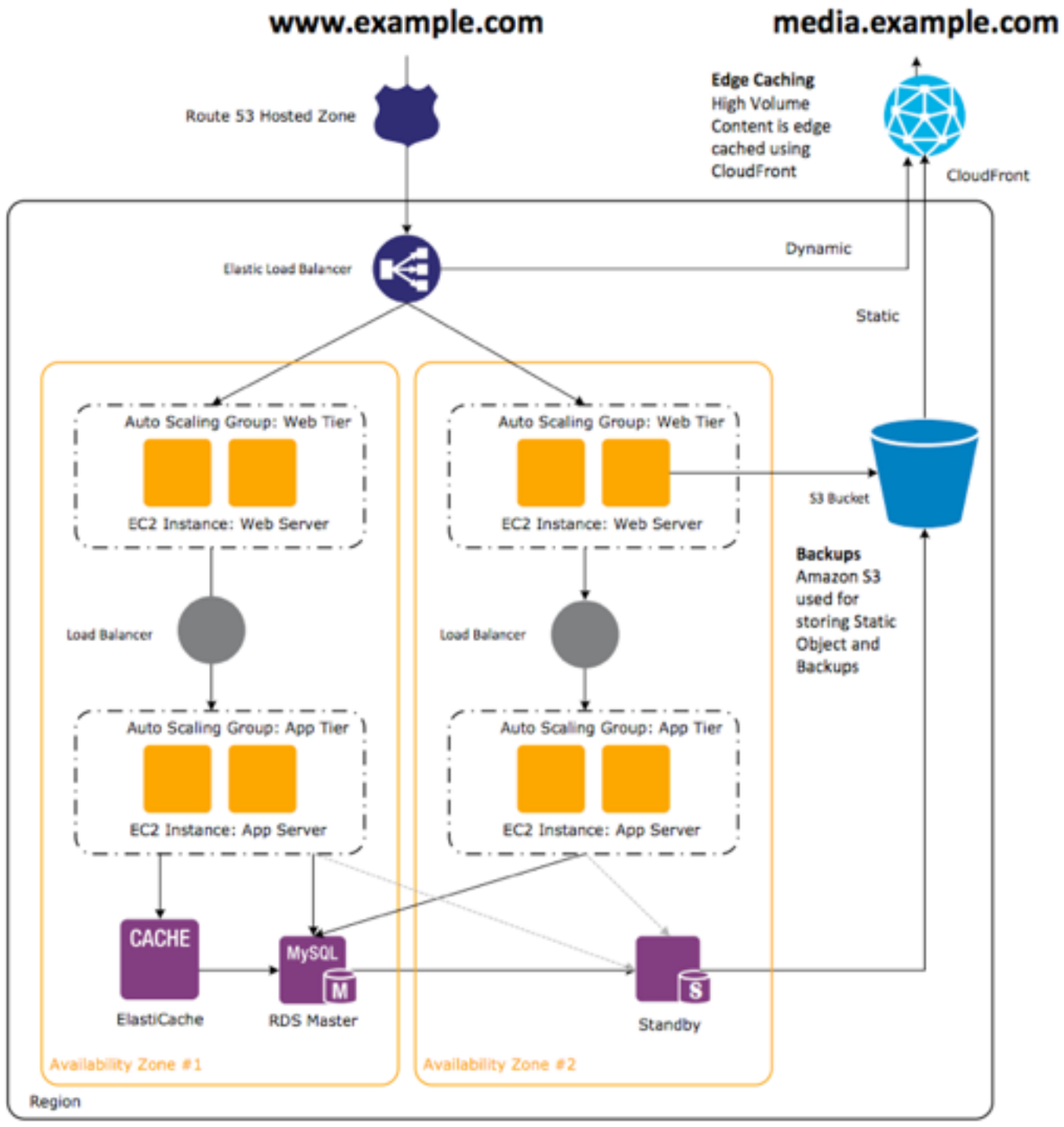
Exterior Firewall moved to every Web Server instance via Security Groups
Auto Scaling Web Tier
Group of EC2 instances handling HTTP requests.
Backend Firewall moved to every back-end instance

App Server Load Balancer
Software LB (e.g. HAProxy) on EC2 instances to spread traffic over app server cluster

Auto-scaling App Tier
Group of EC2 instances running the actual app. Instances belong to Auto Scaling group

ElastiCache
Provides caching services for app, removing load from database tier

DB Tier
MySQL RDS DB creates a highly available architecture with multi-AZ deployments. Read-only replicas can also be used to scale read intensive applications



taken from: http://media.amazonwebservices.com/AWS_Web_Hosting_Best_Practices.pdf

Top 10 Obstacles for Cloud Computing

- **Availability of a service:**
  - Organizations worry about whether Cloud services will have adequate availability
  - Very (very) high availability can be achieved by adopting multiple Cloud Computing providers
  - Even if the Cloud provider has multiple data centers in different geographic regions, it may have common software infrastructure and accounting systems, or the company may even go out of business
- **Data lock-in:**
  - Software stacks have improved interoperability among platforms, but the APIs for Cloud Computing itself are still essentially proprietary
  - Customers cannot easily extract their data and programs from one site to run on another

Top 10 Obstacles for Cloud Computing

- **Data confidentiality and auditability:**
  - My sensitive corporate data will never be in the cloud
  - Current Cloud offerings are essentially public (rather than private) networks, exposing the system to more attacks
  - There are also requirements for auditability and privacy laws (many Nations have laws requiring SaaS providers to keep customer data and copyrighted material within national boundaries)
  - There are no fundamental obstacles to making a Cloud Computing environment as secure as the vast majority of in-house IT environments, well-understood technologies (e,g., encrypted storage, VPN, firewalls,...)
- **Data Transfer Bottlenecks:**
  - Applications continue to become more data-intensive, significant costs in the Cloud
  - Avoid Internet transfers by shipping disks
  - Data Storage for free, CPU cycles sustain the business
  - WAN and LAN bandwidth are still bottlenecks

Top 10 Obstacles for Cloud Computing

- **Performance unpredictability:**
  - Multiple Virtual Machines can share CPUs and main memory surprisingly well in Cloud Computing, but I/O sharing is more problematic
  - Improve architectures and operating systems to efficiently virtualize interrupts and I/O channels
  - Flash memory adoption will decrease I/O interference
- **Scalable storage:**
  - Short-term usage, no up-front cost, and infinite capacity on-demand is more difficult to achieve with persistent storage with respect to computation
- **Bugs in large-scale distributed systems:**
  - Removing errors in very large scale distributed systems is very challenging
  - A common occurrence is that bugs cannot be reproduced in smaller configurations, so the debugging must occur at scale in the production data centers

Top 10 Obstacles for Cloud Computing

- **Scaling quickly:**
  - Pay-as-you-go certainly applies to storage and to network bandwidth, both of which count bytes used
  - Computation is slightly different, depending on the virtualization level:
    - Google AppEngine automatically scales in response to load increases and decreases, and users are charged by the cycles used
    - Amazon EC2 charges by the hour for the number of instances you occupy, even if your machine is idle
  - Automatically scale quickly up and down in response to load is important in order to save money, but without violating service level agreements
- **Reputation fate sharing:**
  - Reputations do not virtualize well. One customer's bad behavior can affect the reputation of the cloud as a whole
  - For instance, blacklisting of Amazon EC2 IP addresses by spam-prevention services may limit which applications can be effectively hosted
  - Legal issues
- **Software licensing:**
  - Current software licenses commonly restrict the computers on which the software can run

Amazon Web Services - AWS

- A set of building- block services, designed to work independently but can work well together:
  - Share a common naming convention and authentication
  - Minimize internal connections and dependencies
- Every function in AWS can be accessed by making a web service call:
  - Start a server, create a load balancer, allocate an IP address, or attach a persistent storage volume,...

AWS Key Concepts

- **Availability Zone:**
  - A set of distinct locations within an AWS Region
  - Each Availability Zone has independent power grid and network connections so that it's protected from failures in other Availability Zones
  - The zones within a Region are connected to each other with inexpensive, low-latency connections
- **Region:**
  - A set of AWS Availability Zones that are located in one geographic area
  - Use of multiple Regions for business, legal, or performance reasons

AWS Key Concepts

- **Amazon Machine Image (AMI):**
  - Contains the operating system and can also include additional software and layers of your application such as database servers, middleware, web servers,...
  - Prebuilt AMIs can be started
  - AMIs can be customized, shared, or even sold
  - Each AMI has a unique ID (e.g., ami-bf5eb9d6)
- **Instance:**
  - Represents one running copy of an AMI
  - Any number of copies of the same AMI can be launched
- **Elastic IP Address:**
  - AWS allows you to allocate fixed (static) IP addresses and then attach (or route) them to your instances
  - Each instance can have at most one such address attached

AWS Key Concepts

- **Elastic Block Store (EBS) Volume:**
  - An addressable disk volume which can be attached to any running instance in the same Availability Zone
  - The volume can then be formatted, mounted, and used as if it were a local disk drive
  - Volumes have a lifetime independent of any particular instance
  - A volume can persist even when no instances are running
- **Security Group:**
  - Defines the allowable set of inbound network connections for an instance
  - Each group is named and consists of a list of protocols, ports, and IP address ranges
  - A group can be applied to multiple instances, and a single instance can be regulated by multiple groups

AWS Infrastructure Services

- Simple Storage Service (S3)
- CloudFront
- Simple Queue Service (SQS)
- SimpleDB
- Relational Database Service (RDS)
- Elastic Compute Cloud (EC2)
- Elastic MapReduce
- Other Services...

Simple Storage Service - S3

- Store **binary data** objects for private or public use
- The implementation is fault-tolerant and assumes that hardware failures are a common occurrence
- S3 automatically makes multiple copies of each object to achieve high **availability** and **durability**
- **Objects** size 1B-5GB
- All objects reside in **buckets**
- S3 objects can be accessed by **HTTP requests**
- Other AWS services use S3 as a storage system for AMIs, access logs, and temporary files
- Amazon S3 charges: amount of data stored, amount of data transferred in and out of S3, and the number of requests made to S3

Simple Storage Service - S3

Region: US Standard

	Standard Storage	Reduced Redundancy Storage	Glacier Storage
First 1 TB / month	$0.095 per GB	$0.076 per GB	$0.010 per GB
Next 49 TB / month	$0.080 per GB	$0.064 per GB	$0.010 per GB
Next 450 TB / month	$0.070 per GB	$0.056 per GB	$0.010 per GB
Next 500 TB / month	$0.065 per GB	$0.052 per GB	$0.010 per GB
Next 4000 TB / month	$0.060 per GB	$0.048 per GB	$0.010 per GB
Over 5000 TB / month	$0.055 per GB	$0.037 per GB	$0.010 per GB

Request Pricing

Region: US Standard

	Pricing
PUT, COPY, POST, or LIST Requests	$0.005 per 1,000 requests
Glacier Archive and Restore Requests	$0.05 per 1,000 requests
Delete Requests	Free †
GET and all other Requests	$0.004 per 10,000 requests
Glacier Data Restores	Free ††

† No charge for delete requests of Standard or RRS objects. For objects that are archived to Glacier, there is a pro-rated charge of $0.03 per gigabyte for objects deleted prior to 90 days. Learn more.

†† Glacier is designed with the expectation that restores are infrequent and unusual, and data will be stored for extended periods of time. You can restore up to 5% of your average monthly Glacier storage (pro-rated daily) for free each month. If you choose to restore more than this amount of data in a month, you are charged a restore fee starting at $0.01 per gigabyte. Learn more.

Data Transfer Pricing

The pricing below is based on data transferred "in" to and "out" of Amazon S3.

Region: US Standard

	Pricing
Data Transfer IN To Amazon S3	
All data transfer in	$0.000 per GB
Data Transfer OUT From Amazon S3 To	
Amazon EC2 in the Northern Virginia Region	$0.000 per GB
Another AWS Region or Amazon CloudFront	$0.020 per GB
Data Transfer OUT From Amazon S3 To Internet	
First 1 GB / month	$0.000 per GB
Up to 10 TB / month	$0.120 per GB
Next 40 TB / month	$0.090 per GB
Next 100 TB / month	$0.070 per GB
Next 350 TB / month	$0.050 per GB
Next 524 TB / month	Contact Us
Next 4 PB / month	Contact Us
Greater than 5 PB / month	Contact Us

Cloud Front

- Content distribution service designed to work in conjunction with Amazon S3
- All Amazon S3 data is served from central locations in the US, Europe, and Asia, access from certain parts of the world can take several hundred milliseconds
- CloudFront addresses this speed limitation with a global network of edge locations (18 today) located near end users
- After you have stored your data in an S3 bucket, you can create a CloudFront **Distribution**
- Each distribution contains a unique URL, which you use in place of the bucket name and S3 domain to achieve content distribution
- CloudFront charges accrue based on the amount of data transferred out of CloudFront and the number of requests made to CloudFront

Simple Queue Service - SQS

- Allows building highly scalable processing pipelines using loosely coupled parts (queues)
- Queues allow for flexibility, asynchrony, and fault tolerance
- Each step in the pipeline retrieves work units from an instance of the queue service, processes the work unit as appropriate, and then writes completed work into another queue for further processing
- Queues work well when the requirements—be it time, CPU, or I/O speed—for each processing step for a particular work unit vary widely
- SQS usage is charged based on the amount of data transferred and the number of requests made to SQS

SimpleDB

- Supports storage and retrieval of semi-structured data
- Unlike a traditional relational database, does not use a fixed database schema
  - Adapts to changes in the "shape" of the stored data on the fly
  - No need to update existing records when adding new fields
- Automatically indexes all stored data

- The SimpleDB data model is flexible and straightforward:
  - Similar data grouped into domains
  - Each domain can hold millions of items, each with a unique key
  - Each item can have a number of attribute/value pairs
  - The attribute names can vary from item to item as needed

Relational Database Service - RDS

- MySQL database instances
- Don’t worry about procuring hardware, installing and configuring an operating system or database engine, or finding storage for backups
- You can scale the amount of processing power up or down, and increase the storage allocation in a matter of minutes, so you can respond easily to changing circumstances
- DB Instance can be backed up into Amazon S3
- Multi-AZ option to run a redundant backup copy of your DB Instance for extra availability and reliability
- Amazon RDS charges accrue based on the amount of time that each DB Instance is running, and the amount of storage allocated to the instance

Other Services

- **CloudWatch:**
  - Provides monitoring within EC2 (CPU load average, disk I/O rate, and network I/O rate)
- **Elastic Load Balancer:**
  - Distributes web traffic across any number of EC2 instances (in the same Availability Zone or across several zones in a Region)
  - Performs also periodic health checks on the instances, and will stop sending traffic to unhealthy instances
- **Auto Scaling:**
  - Uses the data collected by CloudWatch to build systems that can scale out (adding more EC2 instances) and scale in (shutting down EC2 instances) within a defined auto scaling group
  - Usually triggers on CPU/memory utilization

Amazon EC2

- Launch server instances within a set of AMIs
- Instance **types** are available with a wide range of memory, processing power, and local disk storage
- Instances can be launched in any EC2 Region and an Availability Zone can be specified if needed
- Each instance is protected by a **firewall** which, by default, blocks all internal and external connectivity
- Instances can be associated to any number of **security groups**
- One important concept: **Persistent** and **Ephemeral** resources

Persistent Resources

- Once allocated to your account, can be expected to remain operational in the face of transient or permanent hardware or software failures
- The Amazon implementation of a persistent resource makes uses of redundancy, automated failover, and automatic recovery to provide stable resources
- The following EC2 resources are persistent:
  - Elastic IP Addresses
  - EBS Volumes
  - Elastic Load Balancers
  - Security Groups
  - AMIs stored in Amazon S3 or as Amazon EBS snap-shots

Ephemeral Resources

- Without built-in redundancy and will eventually fail
- In case of failures, stored data and state information is generally lost
- Need to use other EC2 facilities to implement your own redundancy, failover, and recovery mechanisms
- **EC2 instances are ephemeral**
- What? The instances can crash at any point and take my local data with them?
  - This is a feature, rather than a bug
  - One of the most important architectural aspects of a large-scale Internet site is that **you can think of the individual servers as if they were all transient and extremely unreliable**
- You can use software to build a reliable system from unreliable parts
- The cloud also makes it easy to simulate failures so that you can ensure recovery logic works as expected

Ephemeral Resources

- A **public IP address** is assigned to each instance as part of the launch process:
  - The IP address will have the same lifetime as the instance
  - The EC2 **Elastic IP address** feature supports allocation of public IP addresses that are stable and that have a lifetime independent of any particular EC2 instance
- The **local disk** storage included with each EC2 instance is ephemeral
  - The storage will remain intact if a running instance is rebooted
  - It's scrubbed and then reused after the instance has been terminated
  - EBS provides persistent storage with high reliability and availability:
    - Create an EBS volume and then attach it to any of your instances in the same Availability Zone
    - Create point-in-time snapshot backups to Amazon S3 and then restore the backups to the same volume