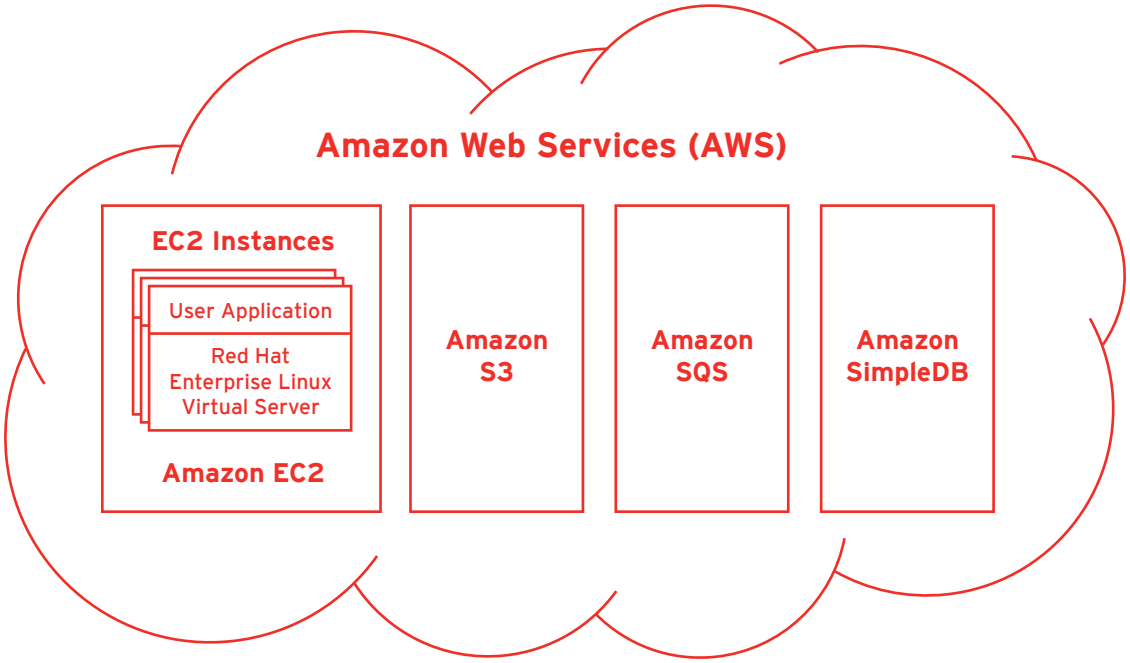
Red Hat Reference Architecture Series

Getting Started with Cloud Computing: Amazon EC2 on Red Hat Enterprise Linux



Version 1.0

June 2008

redhat

amazon.com

Getting Started with Cloud Computing:
Amazon EC2 on Red Hat Enterprise Linux

1801 Varsity Drive
Raleigh NC 27606-2072 USA
Phone: +1 919 754 3700
Phone: 888 733 4281
Fax: +1 919 754 3701
PO Box 13588
Research Triangle Park NC 27709 USA

The GPG fingerprint of the security@redhat.com key is:
CA 20 86 86 2B D6 9D FC 65 F6 EC C4 21 91 80 CD DB 42 A6 0E

Table of Contents

1. Introduction & Executive Summary

Cloud computing with Red Hat Enterprise Linux (RHEL) is a web-scale virtual computing environment powered by Amazon Web Services. It provides everything needed to develop and host applications: compute capacity, network bandwidth, storage capacity, and the leading open source operating system platform, Red Hat Enterprise Linux.

Red Hat's Linux Automation strategy promotes running any application, any-where, anytime including: running applications on-premises (using physical and virtual) servers, running applications in application/server hosted facilities, and finally running application in the cloud, i.e., Amazon's Elastic Compute Cloud (Amazon EC2).

Amazon Elastic Compute Cloud, also known as "EC2", is a commercial web service which allows paying customers to rent computers to run computer applications on. EC2 allows scalable deployment of applications by providing a web services interface through which customers can request an arbitrary number of Virtual Machines, i.e. server instances, on which they can load any software of their choice. Current users are able to create, launch, and terminate server instances on demand, hence the term "elastic". The Amazon implementation allows server instances to be created in zones that are insulated from correlated failures. EC2 is one of several Web Services provided by Amazon.com under the umbrella term Amazon Web Services (AWS).

EC2 uses Xen Virtualization. Each virtual machine, called an instance, is a virtual private server and can be one of several sizes. Instances are sized based on EC2 Compute Units which is the equivalent CPU capacity of physical hardware.

In November 2007, Red Hat announced the availability of Red Hat Enterprise Linux On Demand on Amazon Elastic Compute Cloud (Amazon EC2), a web service that provides resizeable compute capacity in the cloud. This collaboration makes all the capabilities of Red Hat Enterprise Linux, including the Red Hat Network management service, world-class technical support and over 3,400 certified applications, available to customers on Amazon's proven network infrastructure and datacenters.

The combination of Red Hat Enterprise Linux and Amazon EC2 changes the economics of computing by allowing customers to pay only for the infrastructure software services and capacity that they actually use. Red Hat Enterprise Linux for cloud computing makes it easy to develop, deploy, and manage your new and existing applications in a virtual computing environment. Compute capacity can be scaled up or down on demand to accommodate changing workloads and business requirements. Red Hat Enterprise Linux on Amazon EC2 enables customers to increase or decrease capacity within minutes, removing the need to over-buy software and hardware capacity as a set of resources to handle periodic spikes in demand.

Red Hat Enterprise Linux, with integrated virtualization, provides a seamless deployment solution bridging both on-premise and cloud computing. As part of this solution, **Red Hat**

Network (RHN) offers a common set of management and automation tools across on-premises deployments and the Amazon EC2 cloud computing environment. Red Hat will provide technical support and maintenance of Red Hat Enterprise Linux on Amazon EC2. This is the first commercially supported operating system available on Amazon EC2.

"In collaboration with Amazon Web Services, we are now able to offer customers yet another choice in how they deploy the Red Hat Enterprise Linux platform. This offering will be appealing to developers, customers looking to quickly and cost-effectively deploy web-scale services and businesses that require rapidly scaled compute resources," said Donald Fischer, vice president of Online Services at Red Hat. "The marriage of Red Hat Enterprise Linux with Amazon's EC2 service makes the promise of professional web scale computing a reality."
"We are pleased to collaborate with Red Hat in making more choices available for Amazon EC2 users," said Adam Selipsky, vice president, Product Management and Developer Relations, Amazon Web Services. "This offering will further help customers to avoid the heavy lifting of deploying and managing their own infrastructure, while paying as they go for Red Hat's proven Enterprise Linux solution."

2. AWS Infrastructure Services

2.1 Amazon Elastic Compute Cloud (Amazon EC2)

Amazon Elastic Compute Cloud (Amazon EC2) is a web service that provides resizable compute capacity in the cloud. It is designed to make web-scale computing easier for developers.

Amazon EC2's simple web service interface allows you to obtain and configure capacity with minimal friction. It provides you with complete control of your computing resources and lets you run on Amazon's proven computing environment. Amazon EC2 reduces the time required to obtain and boot new server instances to minutes, allowing you to quickly scale capacity, both up and down, as your computing requirements change. Amazon EC2 changes the economics of computing by allowing you to pay only for capacity that you actually use. Amazon EC2 provides developers the tools to build failure resilient applications and isolate themselves from common failure scenarios.

2.2 Amazon Simple Storage Service (Amazon S3)

Amazon S3 is storage for the Internet. It is designed to make web-scale computing easier for developers.

Amazon S3 provides a simple web services interface that can be used to store and retrieve any amount of data, at any time, from anywhere on the web. It gives any developer access to the same highly scalable, reliable, fast, inexpensive data storage infrastructure that Amazon uses to run its own global network of web sites. The service aims to maximize benefits of scale and to pass those benefits on to developers.

2.3 Amazon Simple Queue Service (Amazon SQS)

Amazon Simple Queue Service (Amazon SQS) offers a reliable, highly scalable, hosted queue for storing messages as they travel between computers. By using Amazon SQS, developers can simply move data between distributed components of their applications that perform different tasks, without losing messages or requiring each component to be always available. Amazon SQS makes it easy to build an automated workflow, working in close conjunction with the Amazon Elastic Compute Cloud (Amazon EC2) and the other AWS infrastructure web services.

Amazon SQS works by exposing Amazon's web-scale messaging infrastructure as a web service. Any computer on the Internet can add or read messages without any installed software or special firewall configurations. Components of applications using Amazon SQS

can run independently, and do not need to be on the same network, developed with the same technologies, or running at the same time.

2.4 Amazon SimpleDB

Amazon SimpleDB is a web service for running queries on structured data in real time. This service works in close conjunction with Amazon Simple Storage Service (Amazon S3) and Amazon Elastic Compute Cloud (Amazon EC2), collectively providing the ability to store, process and query data sets in the cloud. These services are designed to make web-scale computing easier and more cost-effective for developers.

Traditionally, this type of functionality has been accomplished with a clustered relational database that requires a sizable upfront investment, brings more complexity than is typically needed, and often requires a DBA to maintain and administer. In contrast, Amazon SimpleDB is easy to use and provides the core functionality of a database - real-time lookup and simple querying of structured data - without the operational complexity. Amazon SimpleDB requires no schema, automatically indexes your data and provides a simple API for storage and access. This eliminates the administrative burden of data modeling, index maintenance, and performance tuning. Developers gain access to this functionality within Amazon's proven computing environment, are able to scale instantly, and pay only for what they use.

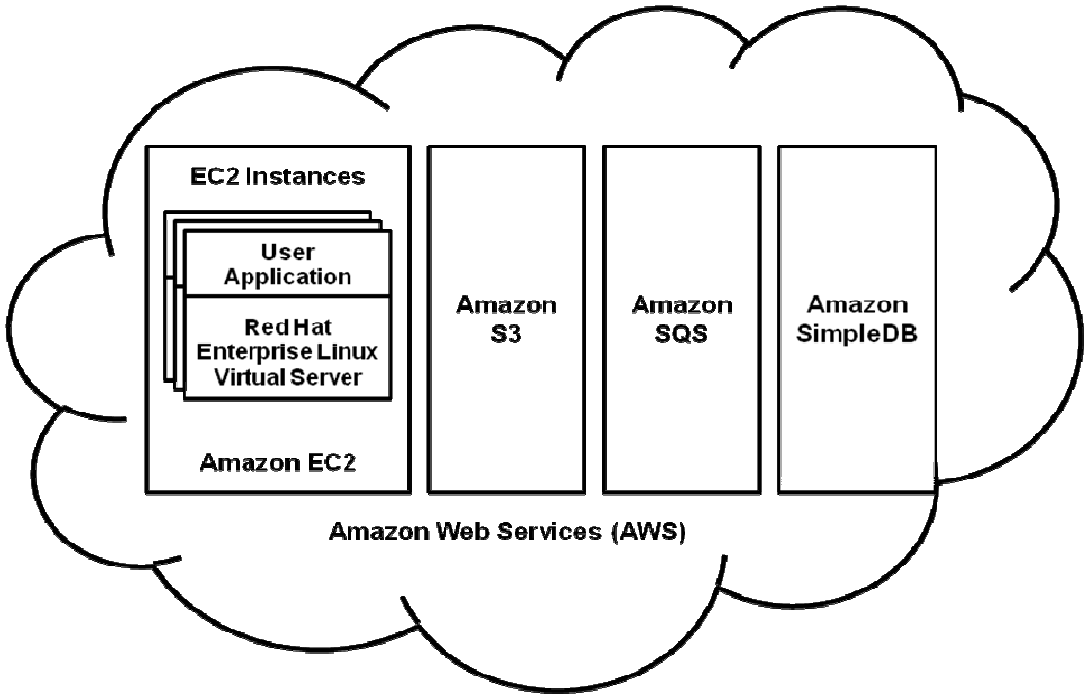


Figure 1

3. Amazon Elastic Compute Cloud (Amazon EC2)

Amazon Elastic Compute Cloud (Amazon EC2) is a web service that provides resizable compute capacity in the cloud. It is designed to make web-scale computing easier for developers.

Amazon EC2's simple web service interface allows you to obtain and configure capacity with minimal friction. It provides you with complete control of your computing resources and lets you run on Amazon's proven computing environment. Amazon EC2 reduces the time required to obtain and boot new server instances to minutes, allowing you to quickly scale capacity, both up and down, as your computing requirements change. Amazon EC2 changes the economics of computing by allowing you to pay only for capacity that you actually use. Amazon EC2 provides developers the tools to build failure resilient applications and isolate themselves from common failure scenarios.

3.1 Amazon EC2 Core Concepts

3.1.1 Amazon Machine Image (AMI)

An Amazon Machine Image (AMI) is an encrypted file stored in Amazon S3. It contains all the information necessary to boot instances of your software.

3.1.2 Amazon EC2 Instance

The running system based on an AMI is referred to as an instance. All instances based on the same AMI begin executing identically. Any information on them is lost when the instances are terminated or if they fail.

3.2 Amazon EC2 Functionality

Amazon EC2 presents a true virtual computing environment, allowing you to use web service interfaces to requisition machines for use, load them with your custom application environment, manage your network's access permissions, and run your image using as many or few systems as you desire.

To use Amazon EC2, you simply:

- Create an Amazon Machine Image (AMI) containing your applications, libraries, data and associated configuration settings. Or use pre-configured, templated images to get up and running immediately.
- Upload the AMI into Amazon S3. Amazon EC2 provides tools that make storing the AMI simple. Amazon S3 provides a safe, reliable and fast repository to store your images.
- Use Amazon EC2 web service to configure security and network access.
- Start, terminate, and monitor as many instances of your AMI as needed, using the web service APIs.
- Pay only for the resources that you actually consume, like instance-hours or data transfer.

3.3 Service Highlights

- **Elastic**
  Amazon EC2 enables you to increase or decrease capacity within minutes, not hours or days. You can commission one, hundreds or even thousands of server instances simultaneously. Of course, because this is all controlled with web service APIs, your application can automatically scale itself up and down depending on its needs.

- **Completely Controlled**
  You have complete control of your instances. You have root access to each one, and you can interact with them as you would any machine. Instances can be rebooted remotely using web service APIs. You also have access to console output of your instances.

- **Flexible**
  You have the choice of several instance types, allowing you to select a configuration of memory, CPU, and instance storage that is optimal for your application.

- **Designed for use with other Amazon Web Services**
  Amazon EC2 works in conjunction with Amazon Simple Storage Service (Amazon S3), Amazon SimpleDB and Amazon Simple Queue Service (Amazon SQS) to provide a complete solution for computing, query processing and storage across a wide range of applications.

- **Reliable**
  Amazon EC2 offers a highly reliable environment where replacement instances can be rapidly and reliably commissioned. The service runs within Amazon's proven network infrastructure and datacenters.

- **Features for Building Failure Resilient Applications**
  Amazon EC2 provides powerful features to build failure resilient applications including:

- **Multiple Locations**
  Amazon EC2 provides the ability to place instances in multiple locations. Amazon EC2 locations are composed of regions and Availability Zones. Regions are geographically dispersed and will be in separate geographic areas or countries. Currently, Amazon EC2 exposes only a single region. Availability Zones are distinct locations that are engineered to be insulated from failures in other Availability Zones and provide inexpensive, low latency network connectivity to other Availability Zones in the same region. Regions consist of one or more Availability Zones. By launching instances in separate Availability Zones, you can protect your applications from failure of a single location.

- **Elastic IP Addresses**
  Elastic IP addresses are static IP addresses designed for dynamic cloud computing. An Elastic IP address is associated with your account not a particular instance, and you control that address until you choose to explicitly release it. Unlike traditional static IP addresses, however, Elastic IP addresses allow you to mask instance or Availability Zone failures by programmatically remapping your public IP addresses to any instance in your account. Rather than waiting on a data technician to reconfigure or replace your host, or waiting for DNS to propagate to all of your customers, Amazon EC2 enables you to engineer around problems with your instance or software by quickly remapping your Elastic IP address to a replacement instance.

- **Secure**
  Amazon EC2 provides web service interfaces to configure firewall settings that control network access to and between groups of instances.

- **Inexpensive**
  Amazon EC2 passes on to you the financial benefits of Amazon's scale. You pay a very low rate for the compute capacity you actually consume. Compare this with the significant up-front expenditures traditionally required to purchase and maintain hardware, either in-house or hosted. This frees you from many of the complexities of capacity planning, transforms what are commonly large fixed costs into much smaller variable costs, and removes the need to over-buy "safety net" capacity to handle periodic traffic spikes.

3.4 Amazon EC2 Instances

Amazon Elastic Compute Cloud (Amazon EC2) provides the flexibility to choose from a number of different instance types to meet your computing needs. Each instance provides a predictable amount of dedicated compute capacity and is charged per instance-hour consumed.

3.4.1 Available Instance Types

Standard Instances

Instances of this family are well suited for most applications.

- **Small Instance (default)***
  1.7 GB memory
  1 EC2 Compute Unit (1 virtual core with 1 EC2 Compute Unit)
  160 GB instance storage (150 GB plus 10 GB root partition)
  32-bit platform
  I/O Performance: Moderate
  Price: $0.10 per instance hour

- **Large Instance**
  7.5 GB memory
  4 EC2 Compute Units (2 virtual cores with 2 EC2 Compute Units each)
  850 GB instance storage (2 x 420 GB plus 10 GB root partition)
  64-bit platform
  I/O Performance: High
  Price: $0.40 per instance hour

- **Extra Large Instance**
  15 GB memory 8 EC2 Compute Units (4 virtual cores with 2 EC2 Compute Units each)
  1,690 GB instance storage (4 x 420 GB plus 10 GB root partition)
  64-bit platform
  I/O Performance: High
  Price: $0.80 per instance hour

High-CPU Instances

Instances of this family have proportionally more CPU resources than memory (RAM) and are well suited for compute-intensive applications.

- **High-CPU Medium Instance**
  1.7 GB of memory
  5 EC2 Compute Units (2 virtual cores with 2.5 EC2 Compute Units each)
  350 GB of instance storage
  32-bit platform
  I/O Performance: Moderate
  Price: $0.20 per instance hour

- **High-CPU Extra Large Instance**
  7 GB of memory
  20 EC2 Compute Units (8 virtual cores with 2.5 EC2 Compute Units each)
  1690 GB of instance storage

64-bit platform
I/O Performance: High
Price: $0.80 per instance hour

3.4.2 Selecting Instance Types

Amazon EC2 instances are grouped into two families: Standard and High-CPU. Standard Instances have memory to CPU ratios suitable for most general purpose applications; High-CPU instances have proportionally more CPU resources than memory (RAM) and are well suited for compute-intensive applications. When choosing instance types, you should consider the characteristics of your application with regards to resource utilization and select the optimal instance family and size.

One of the advantages of EC2 is that you pay by the instance hour, which makes it convenient and inexpensive to test the performance of your application on different instance families and types. One good way to determine the most appropriate instance family and instance type is to launch test instances and benchmark your application.

3.4.3 Measuring Compute Resources

Transitioning to a utility computing model fundamentally changes how developers have been trained to think about CPU resources. Instead of purchasing or leasing a particular processor to use for several months or years, you are renting capacity by the hour. Because Amazon EC2 is built on commodity hardware, over time there may be several different types of physical hardware underlying EC2 instances. Our goal is to provide a consistent amount of CPU capacity no matter what the actual underlying hardware.

Amazon EC2 uses a variety of measures to provide each instance with a consistent and predictable amount of CPU capacity. In order to make it easy for developers to compare CPU capacity between different instance types, we have defined an Amazon EC2 Compute Unit. The amount of CPU that is allocated to a particular instance is expressed in terms of these EC2 Compute Units. We use several benchmarks and tests to manage the consistency and predictability of the performance of an EC2 Compute Unit. **One EC2 Compute Unit provides the equivalent CPU capacity of a 1.0-1.2 GHz 2007 Opteron or 2007 Xeon processor**. This is also the equivalent to an early-2006 1.7 GHz Xeon processor referenced in our original documentation. Over time, we may add or substitute measures that go into the definition of an EC2 Compute Unit, if we find metrics that will give you a clearer picture of compute capacity.

To find out which instance will work best for your application, the best thing to do is to launch an instance and benchmark your own application. One of the advantages of EC2 is that you pay by the hour which makes it convenient and inexpensive to test the performance of your application on different instance types.

3.4.4 I/O Performance

Amazon EC2 provides virtualized server instances. While some resources like CPU, memory and instance storage are dedicated to a particular instance, other resources like the network and the disk subsystem are shared among instances. If each instance on a physical host tries to use as much of one of these shared resources as possible, each will receive an equal share of that resource. However, when a resource is under-utilized you will often be able to consume a higher share of that resource while it is available.

The different instance types will provide higher or lower minimum performance from the shared resources depending on their size. Each of the instance types has an I/O performance indicator (moderate or high). Instance types with high I/O performance have a larger allocation of shared resources. Allocating larger share of shared resources also reduces the variance of I/O performance. For many applications, moderate I/O performance is more than enough. However, for those applications requiring greater or more consistent I/O performance, you may want to consider instances with high I/O performance

*The Small Instance type is equivalent to the original Amazon EC2 instance type that has been available since Amazon EC2 launched. This instance type is currently the default for all customers. To use other instance types, customers must specifically request them via the RunInstances API.

3.5 Amazon EC2 Data Transfer

Internet Data Transfer
$0.100 per GB - all data transfer in
$0.170 per GB - first 10 TB / month data transfer out
$0.130 per GB - next 40 TB / month data transfer out
$0.110 per GB - next 100 TB / month data transfer out
$0.100 per GB - data transfer out / month over 150 TB

Data transfer "in" and "out" refers to transfer into and out of Amazon EC2.

Data transferred between Amazon EC2 and Amazon S3-US or Amazon SimpleDB, is free of charge (i.e., $0.00 per GB), except data transferred between Amazon EC2 and Amazon S3-Europe, which will be charged at regular rates.

Amazon S3 usage is billed separately from Amazon EC2; charges for each service will be billed at the end of the month.

Availability Zone Data Transfer
$0.00 per GB - all data transferred between instances in the same Availability Zone

using private IP addresses.

Regional Data Transfer
$0.01 per GB in/out - all data transferred between instances in different Availability Zones in the same region.

Public and Elastic IP Data Transfer
$0.01 per GB in/out - If you choose to communicate using your Public or Elastic IP address inside of the Amazon EC2 network, you'll pay Regional Data Transfer rates even if the instances are in the same Availability Zone. For data transfer within the same Availability Zone, you can easily avoid this charge (and get better network performance) by using your private IP whenever possible.

* Please note that Regional Data Transfer and Public and Elastic IP Data Transfer will be charged at $0.00 per GB through June 30, 2008. This is to allow current Amazon EC2 users to optimize the placement of instances before incurring any charges. See Availability Zones for tools to describe instance location.

3.6 Amazon EC2 Elastic IP Addresses

No cost for Elastic IP addresses while in use
$0.01 per non-attached Elastic IP address per complete hour
$0.00 per Elastic IP address remap - first 100 remaps / month
$0.10 per Elastic IP address remap - additional remap / month over 100

3.7 Using Amazon EC2 to Run Instances

Amazon EC2 allows you to set up and configure everything about your instances from your operating system up to your applications. An Amazon Machine Image (AMI) is simply a packaged-up environment that includes all the necessary bits to set up and boot your instance. Your AMIs are your unit of deployment. You might have just one AMI or you might compose your system out of several building block AMIs (e.g., webservers, appservers, and databases). Amazon EC2 provides a number of command line tools to make creating an AMI easy. Once you create a custom AMI, you will need to upload it to Amazon S3. Amazon EC2 uses Amazon S3 to provide reliable, scalable storage of your AMIs so that we can boot them when you ask us to do so.

You can also choose from a library of globally available AMIs that provide useful instances. For example, if you just want a simple Linux server, you can choose one of the standard Linux distribution AMIs. Once you have set up your account and uploaded your AMIs, you are ready to boot your instance. You can start your AMI on any number and any type of instance by calling the RunInstances API. If you wish to run more than 20 instances or if you

feel you need more than 5 Elastic IP addresses, please complete the Amazon EC2 instance request form or the Elastic IP request form and your increase request will be considered.

3.8 Paying for What You Use

You will be charged at the end of each month for your EC2 resources actually consumed.

As an example, assume you launch 100 instances of the Small type costing $0.10 per hour at some point in time. The instances will begin booting immediately, but they won't necessarily all start at the same moment. Each instance will store its actual launch time. Thereafter, each instance will charge for its hours (at $.10/hour) of execution at the beginning of each hour relative to the time it launched. Each instance will run until one of the following occurs: you terminate the instance with the TerminateInstances API call (or an equivalent tool), the instance shuts itself down (e.g. Unix "shutdown" command), or the host terminates due to software or hardware failure. Partial instance hours consumed are billed as full hours.

4. Amazon Simple Storage Service (Amazon S3)

Amazon S3 is storage for the Internet. It is designed to make web-scale computing easier for developers.

Amazon S3 provides a simple web services interface that can be used to store and retrieve any amount of data, at any time, from anywhere on the web. It gives any developer access to the same highly scalable, reliable, fast, inexpensive data storage infrastructure that Amazon uses to run its own global network of web sites. The service aims to maximize benefits of scale and to pass those benefits on to developers.

4.1 Amazon S3 Functionality

Amazon S3 is intentionally built with a minimal feature set.

- Write, read, and delete objects containing from 1 byte to 5 gigabytes of data each. The number of objects you can store is unlimited.
- Each object is stored in a bucket and retrieved via a unique, developer-assigned key.
- A bucket can be located in the United States or in Europe. All objects within the bucket will be stored in the bucket's location, but the objects can be accessed from anywhere.
- Authentication mechanisms are provided to ensure that data is kept secure from unauthorized access. Objects can be made private or public, and rights can be granted to specific users.
- Uses standards-based REST and SOAP interfaces designed to work with any Internet-development toolkit.
- Built to be flexible so that protocol or functional layers can easily be added. Default download protocol is HTTP. A BitTorrent(TM) protocol interface is provided to lower costs for high-scale distribution. Additional interfaces will be added in the future.
- Reliability backed with the Amazon S3 Service Level Agreement.

4.2 Pay for What You Use

Pay only for what you use. There is no minimum fee. Estimate your monthly bill using the AWS Simple Monthly Calculator.

We charge less where our costs are less, thus some prices vary across geographic regions and are based on the location of the bucket.

United States

Storage
$0.15 per GB-Month of storage used

Data Transfer
$0.100 per GB - all data transfer in

$0.170 per GB - first 10 TB / month data transfer out
$0.130 per GB - next 40 TB / month data transfer out
$0.110 per GB - next 100 TB / month data transfer out
$0.100 per GB - data transfer out / month over 150 TB

Requests
$0.01 per 1,000 PUT, POST, or LIST requests
$0.01 per 10,000 GET and all other requests*
* No charge for delete requests

Europe

Storage
$0.18 per GB-Month of storage used

Data Transfer
$0.100 per GB - all data transfer in

$0.170 per GB - first 10 TB / month data transfer out
$0.130 per GB - next 40 TB / month data transfer out
$0.110 per GB - next 100 TB / month data transfer out
$0.100 per GB - data transfer out / month over 150 TB

Requests
$0.012 per 1,000 PUT, POST, or LIST requests
$0.012 per 10,000 GET and all other requests*
* No charge for delete requests

Data transfer "in" and "out" refers to transfer into and out of Amazon S3. Data transferred between Amazon EC2 and Amazon S3, is free of charge (i.e., $0.00 per GB), except data transferred between Amazon EC2 and Amazon S3-Europe, which will be charged at regular rates.

4.3 Amazon S3 Design Requirements

Amazon S3 is based on the idea that quality Internet-based storage should be taken for granted. It helps free developers from worrying about how they will store their data, whether it will be safe and secure, or whether they will have enough storage available. It frees them

from the upfront costs of setting up their own storage solution as well as the ongoing costs of maintaining and scaling their storage servers. The functionality of Amazon S3 is simple and robust: Store any amount of data inexpensively and securely, while ensuring that the data will always be available when you need it. Amazon S3 enables developers to focus on innovating with data, rather than figuring out how to store it.

Amazon S3 was built to fulfill the following design requirements:

- **Scalable**: Amazon S3 can scale in terms of storage, request rate, and users to support an unlimited number of web-scale applications. It uses scale as an advantage: Adding nodes to the system increases, not decreases, its availability, speed, throughput, capacity, and robustness.

- **Reliable**: Store data durably, with 99.99% availability. There can be no single points of failure. All failures must be tolerated or repaired by the system without any downtime.

- **Fast**: Amazon S3 must be fast enough to support high-performance applications. Server-side latency must be insignificant relative to Internet latency. Any performance bottlenecks can be fixed by simply adding nodes to the system.

- **Inexpensive**: Amazon S3 is built from inexpensive commodity hardware components. As a result, frequent node failure is the norm and must not affect the overall system. It must be hardware-agnostic, so that savings can be captured as Amazon continues to drive down infrastructure costs.

- **Simple**: Building highly scalable, reliable, fast, and inexpensive storage is difficult. Doing so in a way that makes it easy to use for any application anywhere is more difficult. Amazon S3 must do both.

A forcing-function for the design was that a single Amazon S3 distributed system must support the needs of both internal Amazon applications and external developers of any application. This means that it must be fast and reliable enough to run Amazon.com's websites, while flexible enough that any developer can use it for any data storage need.

4.4 Amazon S3 Design Principles

The following principles of distributed system design were used to meet Amazon S3 requirements:

- **Decentralization**: Use fully decentralized techniques to remove scaling bottlenecks and single points of failure.

- **Asynchrony**: The system makes progress under all circumstances.

- **Autonomy**: The system is designed such that individual components can make

decisions based on local information.

- **Local responsibility**: Each individual component is responsible for achieving its consistency; this is never the burden of its peers.
- **Controlled concurrency**: Operations are designed such that no or limited concurrency control is required.
- **Failure tolerant**: The system considers the failure of components to be a normal mode of operation, and continues operation with no or minimal interruption.
- **Controlled parallelism**: Abstractions used in the system are of such granularity that parallelism can be used to improve performance and robustness of recovery or the introduction of new nodes.
- **Decompose into small well-understood building blocks**: Do not try to provide a single service that does everything for everyone, but instead build small components that can be used as building blocks for other services.
- **Symmetry**: Nodes in the system are identical in terms of functionality, and require no or minimal node-specific configuration to function.
- **Simplicity**: The system should be made as simple as possible (but no simpler).

5. Amazon Simple Queue Service (Amazon SQS)

Amazon Simple Queue Service (Amazon SQS) offers a reliable, highly scalable, hosted queue for storing messages as they travel between computers. By using Amazon SQS, developers can simply move data between distributed components of their applications that perform different tasks, without losing messages or requiring each component to be always available. Amazon SQS makes it easy to build an automated workflow, working in close conjunction with the Amazon Elastic Compute Cloud (Amazon EC2) and the other AWS infrastructure web services.

Amazon SQS works by exposing Amazon's web-scale messaging infrastructure as a web service. Any computer on the Internet can add or read messages without any installed software or special firewall configurations. Components of applications using Amazon SQS can run independently, and do not need to be on the same network, developed with the same technologies, or running at the same time.

5.1 Amazon SQS Functionality

- Developers can create an unlimited number of Amazon SQS queues, each of which can send and receive an unlimited number of messages.
- New messages can be added to a queue at any time. The message body can contain up to 8 KB of text in any format.
- A computer can check a queue at any time for messages waiting to be read.
- A message is "locked" while a computer is processing it, keeping other computers from trying to process it simultaneously. If processing fails, the lock will expire and the message will again be available.
- Messages can be retained in queues for up to 4 days.
- Developers can access Amazon SQS through standards-based SOAP and Query interfaces designed to work with any Internet-development toolkit.

5.2 SQS Service Highlights

Amazon SQS provides significant advantages over the complexity of home-grown messaging systems or the ongoing expense of licensed or hosted systems.

- **Reliable** - Amazon SQS runs within Amazon's high-availability data centers, so queues will be available whenever applications need them. To prevent messages from being lost or becoming unavailable, all messages are stored redundantly across multiple servers and data centers.

- **Simple** - Developers can utilize Amazon SQS queues by using only four APIs: CreateQueue, SendMessage, ReceiveMessage, and DeleteMessage. The SOAP and

query APIs can be used with virtually any language and platform.

- **Scalable** - Amazon SQS was designed to enable an unlimited number of computers to read and write and unlimited number of messages at any time.

- **Secure** - Authentication mechanisms are provided to ensure that messages stored in Amazon SQS queues are secured against unauthorized access.

- **Inexpensive** - No up-front or fixed expenses. The only costs of sending messages through Amazon SQS are small per-request handling fees and data transfer fees. For $1 a user can transmit over a half a million (500,000) messages.

5.3 Using Amazon SQS with Other AWS Infrastructure Web Services

Amazon SQS can be used with Amazon EC2, as well as Amazon S3 and Amazon SimpleDB, to make applications more flexible and scalable. A common use case is to create an integrated and automated workflow, where multiple components or modules need to communicate with each other, but can't all process the same amount of work simultaneously. In this case, SQS queues carry messages to be processed in an orderly fashion by the user's application running on Amazon EC2 instances. The Amazon EC2 instances can read the queue, process the job, and then post the results as messages to another SQS queue (possibly for further processing by another application). Because Amazon EC2 allows applications to scale up and down dynamically, application developers can easily vary the number of compute instances based on the amount of work in the SQS queues, to ensure that jobs are executed in a timely manner.

For example, here is how a video transcoding website uses Amazon EC2, Amazon SQS, Amazon S3, and Amazon SimpleDB together. End users submit videos to be transcoded to the website. The videos are stored in Amazon S3, and a message ("the request message") is placed in an Amazon SQS queue ("the incoming queue") with a pointer to the video and to the target video format in the message. The transcoding engine, running on a set of Amazon EC2 instances, reads the request message from the incoming queue, retrieves the video from Amazon S3 using the pointer, and transcodes the video into the target format. The converted video is put back into Amazon S3 and another message ("the response message") is placed in another Amazon SQS queue ("the outgoing queue") with a pointer to the converted video. At the same time, metadata about the video (e.g., format, date created and length) can be indexed into Amazon SimpleDB for easy querying. During this whole workflow, a dedicated Amazon EC2 instance can constantly monitor the incoming queue and, based on the number of messages in the incoming queue, is able to dynamically adjust the number of transcoding Amazon EC2 instances to meet customers' response time requirements.

5.4 Pay for What You Use

Pay only for what you use. There is no minimum fee. Estimate your monthly bill using AWS Simple Monthly Calculator.

Please note that Amazon SQS introduced a new WSDL and pricing plan on February 6, 2008. Pricing for the current WSDL (version 2008-01-01)

Requests
$0.01 per 10,000 Amazon SQS Requests ($0.000001 per Request)

Amazon SQS requests are CreateQueue, ListQueues, DeleteQueue, SendMessage, ReceiveMessage, DeleteMessage, SetQueueAttributes and GetQueueAttributes

Data Transfer
$0.100 per GB - all data transfer in

$0.170 per GB - first 10 TB / month data transfer out
$0.130 per GB - next 40 TB / month data transfer out
$0.110 per GB - next 100 TB / month data transfer out
$0.100 per GB - data transfer out / month over 150 TB

Data transfer "in" and "out" refers to transfer into and out of Amazon SQS. Data transferred between Amazon SQS and Amazon EC2 is free of charge (i.e., $0.00 per GB)

5.5 SQS - Detailed Description

5.5.1 Basic Queue Requests

The Amazon Simple Queue Service employs a simple interface that is easy to use and highly flexible. The following requests are provided:

- *CreateQueue*: Create queues for use with your AWS account.
- *ListQueues*: List your existing queues.
- *DeleteQueue*: Delete one of your queues.
- *SendMessage*: Add any data entries to a specified queue.
- *ReceiveMessage*: Return one or more messages from a specified queue.
- *DeleteMessage*: Remove a previously received message from a specified queue.
- *SetQueueAttributes*: Control queue settings like the amount of time that messages are locked after being read so they cannot be read again.
- *GetQueueAttributes*: See information about a queue like the number of messages in it.

5.5.2 Amazon SQS Message Lifecycle

Messages that are stored in Amazon SQS have a lifecycle that is easy to manage but ensures that all messages are processed.

1 A system that needs to send a message will find an Amazon SQS queue, and use *SendMessage* to add a new message to it.
2 A different system that processes messages needs more messages to process, so it calls *ReceiveMessage*, and this message is returned.
3 Once a message has been returned by *ReceiveMessage*, it will not be returned by any other *ReceiveMessage* until the visibility timeout has passed. This keeps multiple computers from processing the same message at once.
4 If the system that processes messages successfully finishes working with this message, it calls *DeleteMessage*, which removes the message from the queue so no one else will ever process it. If this system fails to process the message, than it will be read by another *ReceiveMessage* call as soon as the visibility timeout passes.

6. Amazon SimpleDB™

Amazon SimpleDB is a web service for running queries on structured data in real time. This service works in close conjunction with Amazon Simple Storage Service (Amazon S3) and Amazon Elastic Compute Cloud (Amazon EC2), collectively providing the ability to store, process and query data sets in the cloud. These services are designed to make web-scale computing easier and more cost-effective for developers.

Traditionally, this type of functionality has been accomplished with a clustered relational database that requires a sizable upfront investment, brings more complexity than is typically needed, and often requires a DBA to maintain and administer. In contrast, Amazon SimpleDB is easy to use and provides the core functionality of a database - real-time lookup and simple querying of structured data - without the operational complexity. Amazon SimpleDB requires no schema, automatically indexes your data and provides a simple API for storage and access. This eliminates the administrative burden of data modeling, index maintenance, and performance tuning. Developers gain access to this functionality within Amazon's proven computing environment, are able to scale instantly, and pay only for what they use.

6.1 Amazon SimpleDB Functionality

Amazon SimpleDB provides a simple web services interface to create and store multiple data sets, query your data easily, and return the results.

You organize your structured data into domains and can run queries across all of the data stored in a particular domain. Domains are comprised of items, and items are described by attribute-value pairs. To understand these elements, consider the metaphor of data stored in a spreadsheet table. An Amazon SimpleDB domain is like a worksheet, items are like rows of data, attributes are like column headers, and values are the data entered in each of the cells.

However unlike a spreadsheet, Amazon SimpleDB allows for multiple values to be associated with each "cell" (e.g., for item "123," the attribute "color" can have both value "blue" and value "red"). Additionally, in Amazon SimpleDB, each item can have its own unique set of associated attributes (e.g., item "123" might have attributes "description" and "color" whereas item "789" has attributes "description," "color" and "material"). Amazon SimpleDB automatically indexes your data, making it easy to quickly find the information that you need. There is no need to pre-define a schema or change a schema if new data is added later.

To use Amazon SimpleDB you:

- CREATE a new domain to house your unique set of structured data.
- GET, PUT or DELETE items in your domain, along with the attribute-value pairs that you associate with each item. Amazon SimpleDB automatically indexes data as it is

added to your domain so that it can be quickly retrieved; there is no need to pre-define a schema or change a schema if new data is added later. Each item can have up to 256 attribute values. Each attribute value can range from 1 to 1,024 bytes.

- QUERY your data set using this simple set of operators: =, !=, <, > <=, >=, STARTS-WITH, AND, OR, NOT, INTERSECTION AND UNION. Query execution time is currently limited to 5 seconds. Amazon SimpleDB is designed for real-time applications and is optimized for those use cases.

- Pay only for the resources that you consume.

6.2 Service Highlights

- **Simple to use**
  Amazon SimpleDB provides streamlined access to the lookup and query functions that traditionally are achieved using a relational database cluster - while leaving out other complex, often-unused database operations. The service allows you to quickly add data and easily retrieve or edit that data through a simple set of API calls. Accessing these capabilities through a web service also eliminates the complexity of maintaining and scaling these operations.

- **Flexible**
  With Amazon SimpleDB, it is not necessary to pre-define all of the data formats you will need to store; simply add new attributes to your Amazon SimpleDB data set when needed, and the system will automatically index your data accordingly. The ability to store structured data without first defining a schema provides developers with greater flexibility when building applications.

- **Scalable**
  Amazon SimpleDB allows you to easily scale your application. You can quickly create new domains as your data grows or your request throughput increases. For the Beta release, a single domain is limited in size to 10 GB and you are limited to a maximum of 100 domains; however, over time these limits may be raised.

- **Fast**
  Amazon SimpleDB provides quick, efficient storage and retrieval of your data to support high performance web applications.

- **Reliable**
  The service runs within Amazon's high-availability data centers to provide strong and consistent performance. To prevent data from being lost or becoming unavailable, your fully indexed data is stored redundantly across multiple servers and data centers.

- **Designed for use with other Amazon Web Services**
  Amazon SimpleDB is designed to integrate easily with other web-scale services such

as Amazon EC2 and Amazon S3. For example, developers can run their applications in Amazon EC2 and store their data objects in Amazon S3. Amazon SimpleDB can then be used to query the object metadata from within the application in Amazon EC2 and return pointers to the objects stored in Amazon S3.

- **Inexpensive**
  Amazon SimpleDB passes on to you the financial benefits of Amazon's scale. You pay only for resources you actually consume. Compare this with the significant up-front expenditures traditionally required to obtain software licenses and purchase and maintain hardware, either in-house or hosted. This frees you from many of the complexities of capacity planning, transforms large capital expenditures into much smaller operating costs, and eliminates the need to over-buy "safety net" capacity to handle periodic traffic spikes.

6.3 Pay for What you Use

Pay only for what you use. There is no minimum fee.

Machine Utilization - $0.14 per Amazon SimpleDB Machine Hour consumed

Amazon SimpleDB measures the machine utilization of each request and charges based on the amount of machine capacity used to complete the particular request (**QUERY**, **GET**, **PUT**, etc.), normalized to the hourly capacity of a circa 2007 1.7 GHz Xeon processor.

Data Transfer
$0.100 per GB - all data transfer in

$0.170 per GB - first 10 TB / month data transfer out
$0.130 per GB - next 40 TB / month data transfer out
$0.110 per GB - next 100 TB / month data transfer out
$0.100 per GB - data transfer out / month over 150 TB

Data transfer "in" and "out" refers to transfer into and out of Amazon SimpleDB. Data transferred between Amazon SimpleDB and other Amazon Web Services is free of charge (i.e., $0.00 per GB).

Structured Data Storage - $1.50 per GB-month

Amazon SimpleDB measures the size of your billable data by adding the raw byte size of the data you upload + 45 bytes of overhead for each item, attribute name and attribute-value pair.

Amazon SimpleDB is designed to store relatively small amounts of data and is optimized for fast data access and flexibility in how that data is expressed. In order to minimize your costs across AWS services, large objects or files should be stored in Amazon S3, while the pointers and the meta-data associated with those files can be stored in Amazon SimpleDB. This will allow you to quickly search for and access your files, while minimizing overall storage costs. See below for detailed descriptions on calculating your own structured data storage requirements and for a more detailed explanation of how storage in Amazon SimpleDB and storage in Amazon S3 differ.

6.4 Detailed Description

6.4.1 The Data Model: Domains, Items, Attributes and Values

The data model used by Amazon SimpleDB makes it easy to input, manage and query your structured data. Developers organize their data-set into domains and can run queries across all of the data stored in a particular domain. Domains are collections of items that are described by attribute-value pairs.

Think of these terms as analogous to concepts in a traditional spreadsheet table. For example, take the details of a product catalog shown in the table below and consider how they would be represented in Amazon SimpleDB. The whole table/catalog would be your domain named "clothing." Individual products would be rows in the table or items in your domain. The characteristics of items would be described by column headers (attributes). Values are in individual cells. Now consider the items below - a sweater, a dress shirt and a pair of shoes - are new products you would like to add to your domain.

itemID	description	color	Material
123	sweater	blue, red	
456	dress shirt	white, blue	
789	Shoes	black	leather

In Amazon SimpleDB, to add the items above, you would PUT the three itemIDs into your domain along with the attribute-value pairs for each of the items. Without the specific syntax, it would look something like this:

PUT (item, 123), (description, sweater), (color, blue), (color, red)
PUT (item, 456), (description, dress shirt), (color, white), (color, blue)
PUT (item, 789), (description, shoes), (color, black), (material, leather)

Amazon SimpleDB differs from tables of traditional databases in several important ways. First, you have the flexibility to easily go back later on and add new attributes that only apply to certain items - for example, sleeve length for dress shirts. Additionally there is no need to pre-define data types. If you have an attribute called "size," it can have values of 10.5 for shoes and XL for sweaters, making the service extremely flexible and easy to use.

Amazon SimpleDB automatically indexes all of your data, enabling you to easily query for an item based on attributes and their values. In the above example, you could submit a query for items where (color = blue INTERSECTION description = dress shirt), and Amazon SimpleDB would quickly return item 456 as the result.

6.4.2 API Summary

Amazon SimpleDB provides a small number of simple API calls which implement writing, indexing and querying data. The interface and feature set are intentionally focused on core functionality, providing a basic API for developers to build upon and making the service easy to learn and simple to use.

- *CreateDomain* - Create a domain that contains your dataset.
- *DeleteDomain* - Delete a domain.
- *ListDomains* - List all domains and associated metadata.
- *PutAttributes* - Add or update an item and its attributes, or add attribute-value pairs to items that exist already. Items are automatically indexed as they are received.
- *GetAttributes* - Retrieve an item and all or a subset of its attributes and values.
- *DeleteAttributes* - Delete an item, an attribute, or an attribute value.
- *Query* - Query the dataset using a query expression which specifies value tests on one or more attributes. Supported value tests are: =, !=, <, > <=, >=, starts-with. Example: ["price" < "12.00"] INTERSECTION ["color" = "green"]

6.4.3 Amazon SimpleDB and Relational Databases within AWS

Today, many developers correlate the word "database" with Relational Database Management Systems (RDBMS). While RDBMS offerings provide deep functionality, for many use cases, they introduce more complexity (and more cost) than is necessary. Many developers simply want to store, process, and query their data without worrying about managing schemas, maintaining indexes, tuning performance or scaling access to their data. Amazon SimpleDB removes the need to maintain a schema, while your attributes are automatically indexed to provide fast real-time lookup and querying capabilities. This flexibility minimizes the performance tuning required as the demands for your data increase.

Amazon SimpleDB eliminates administrative complexity by providing a simple set of APIs focused on the core functionality necessary to store, process, and query your data. The simplicity of this set of APIs, and the ability to access this service "in the cloud," allow you to quickly develop sophisticated applications without employing a DBA. Amazon SimpleDB allows you to easily scale your application based on your needs. You can quickly create new domains as your data grows or your request throughput increases. You no longer have to be concerned about obtaining software licenses, purchasing and maintain hardware, and managing capacity. You pay only for what you use.

Some developers do require a complex schema or broader functionality, and will undertake the extra work required to run their own relational database. Many developers are doing just this by hosting their own databases inside the Amazon EC2 compute environment. This provides them complete control over whatever database they choose to run, while still accessing the benefits of Amazon's computing infrastructure and the ability to scale capacity up and down instantly.

Either choice is fine with us. Over time, we plan to continue to add features that make it as easy as possible for developers to pursue whichever option they prefer for obtaining database functionality.

6.4.4 Data Storage in Amazon SimpleDB vs. Data Storage in Amazon S3

Unlike Amazon S3, Amazon SimpleDB is not storing raw data. Rather, it takes your data as input and expands it to create indices across multiple dimensions, which enables you to quickly query that data. Additionally, Amazon S3 and Amazon SimpleDB use different types of physical storage. Amazon S3 uses dense storage drives that are optimized for storing larger objects inexpensively. Amazon SimpleDB stores smaller bits of data and uses less dense drives that are optimized for data access speed.

In order to optimize your costs across AWS services, large objects or files should be stored in Amazon S3, while smaller data elements or file pointers (possibly to Amazon S3 objects) are best saved in Amazon SimpleDB. Because of the close integration between services and the free data transfer within the AWS environment, developers can easily take advantage of both the speed and querying capabilities of Amazon SimpleDB as well as the low cost of storing data in Amazon S3, by integrating both services into their applications.

6.4.5 Calculating Your Storage Needs

The best way to predict the size of your structured data storage is as follows:

Raw byte size (GB) of all item IDs + 45 bytes per item +
Raw byte size (GB) of all attribute names + 45 bytes per attribute name +
Raw byte size (GB) of all attribute-value pairs + 45 bytes per attribute-value pair

To calculate your estimated monthly storage cost, take the resulting size in GB and multiply by $1.50

6.4.6 Machine Utilization Example

Amazon SimpleDB measures the machine utilization of each request and charges based on the amount of machine capacity used to complete the particular request (QUERY, GET, PUT,

etc.), normalized to the hourly capacity of a circa 2007 1.7 GHz Xeon processor. Machine utilization is driven by the amount of data (# of attributes, length of attributes) processed by each request. A GET operation that retrieves 256 attributes will use more resources than a GET that retrieves only 1 attribute. A multi-predicate QUERY that examines 100,000 attributes will cost more than a single predicate query that examines 250.

In the response message for each request, Amazon SimpleDB returns a field called Box Usage. Box Usage is the measure of machine resources consumed by each request. It does not include bandwidth or storage. Box usage is reported as the portion of a machine hour used to complete a particular request. The cost of an individual request is Box Usage (expressed in hours) * $0.14 per Amazon SimpleDB Machine hour. The cost of all your requests is the sum of Box Usage (expressed in hours) * $0.14.

For example, if over the course of a month, the sum of the Box Usage for your requests uses the equivalent of one 1.7 GHz Xeon processor for 9 hours, your charge will be:

9 hours * $0.14 per Amazon SimpleDB Machine hour = $1.26.

7. Running RHEL Virtual Servers as EC2 Instances

7.1 Introduction

Red Hat Enterprise Linux on Amazon EC2 is the certified and supported operating platform purchasable by the hour—delivered by Amazon and supported by Red Hat. As this is Red Hat Enterprise Linux, most operational and management tasks are the same or very similar to managing Red Hat Enterprise Linux for on-premise deployments. Basic knowledge of management tasks is recommended, as well as a familiarity with the Red Hat Enterprise Linux System Administrator Guide (=> http://www.redhat.com/docs/manuals/enterprise/RHEL-4-Manual/en-US/System_Administration_Guide/) or Red Hat Enterprise Linux Deployment Guide (=> http://www.redhat.com/docs/manuals/enterprise/RHEL-5-manual/en-US/RHEL510/Deployment_Guide/index.html) as well as different courses offered by Red Hat on the subject.

This guide concentrates on differences specific to operating within the Amazon EC2 (Elastic Compute Cloud). Only basic EC2 operations are explained in this document, if you want to learn more and understand the full EC2 API, read the EC2 technical documentation, which is available in the Resource Center at Amazon Web Services, (=> www.amazon.com/aws).

Most major management tasks, for example running and stopping instances, are simplified if you install and use the EC2 Firefox extension that is available through the Amazon Web Service Resource Center (http://sourceforge.net/projects/elasticfox/ , (http://developer.amazonwebservices.com/connect/entry.jspa?entryID=609). Although not required for operation, many of the operations available within the EC2 API tools (for example “ec2run”) are also available and easily managed via the EC2 Firefox extension.

7.2 EC2 Constraints

One important feature of EC2 based virtual instances during the beta period is that they are always stateless. This means that whenever you restart an instance, it picks up the state of the underlying AMI (Amazon Machine Image). For many tasks suitable for virtual instances, this is a favorable model as you always know exactly what is the initial state of an instance, and can simplify the management of multiple servers processing same tasks or offering same services.

However, stateless nature requires you to closely follow the management procedure described below, to make sure that you don’t lose any configuration changes or updates and additional installations that you might perform.

Additionally, Red Hat Network (RHN => https://www.redhat.com/rhn/) stores the profile of all

connected servers. If you restart an instance of your profile on RHN, the corresponding profile becomes invalid and unusable. More about this requirement and the reasoning behind it is described in the Red Hat Network Channel Management Guide (=> https://rhn.redhat.com/rhn/help/channel-mgmt/).

While running instances themselves do not contain any persistent storage, Amazon offers the process of **rebundling**, which provides a method for storing the data and configuration of an instance and makes it available for multiple future executions of a virtual instance. Rebundling utilizes Amazon's S3 (Simple Storage Service). The management process described below utilizes rebundling to store the configuration of an instance after managing the instance with tools such as RHN.

Figure 2 shows the flowchart of the process you need to follow when managing Enterprise Linux Hosted images and instances.

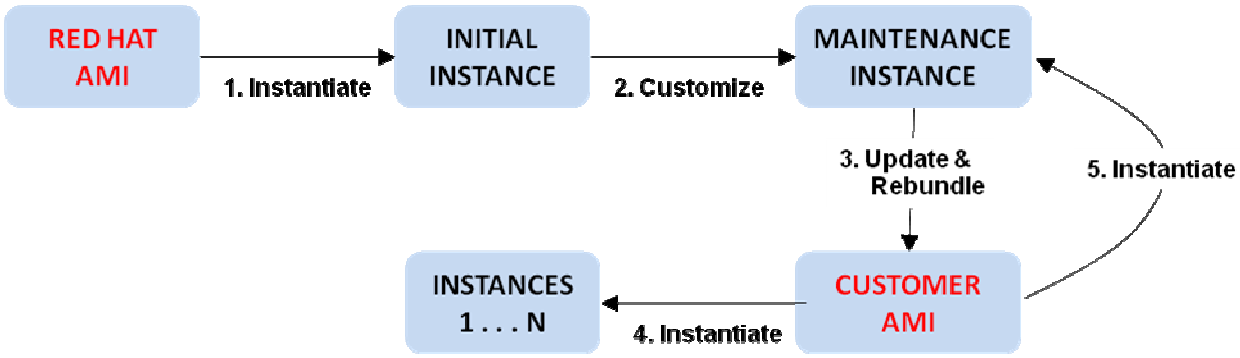


Figure 2

7.3 Getting Started Overview

The following steps presume that you have purchased a subscription to Red Hat Enterprise Linux on Amazon EC2, you have access to the AMI's that are provided by the subscription, have activated your Red Hat Network entitlement, and have accounts at both Amazon Web Services and Red Hat Network. If you have not completed these steps – please review the process at www.redhat.com/solutions/cloud.

- Start & Configuring the Initial Instance
- Storing the Configured Image
- Future Configuration and Maintenance of Images

7.4 Start & Configuring the Initial Instance

Using ‘ec2run’, start a new instance and wait for it to become available. Once the image is up and running, you can connect to it using an ssh client.

On the first log in, you are presented with a configuration interface, similar to the one used in the first boot step on physical servers. Use it to configure your virtual instance as well as register for RHN access. Once the first boot step is done, you can update the instance and install any additional required software, as well as perform any additional configurations.

Note: do not restart the instance before completing the next step.

7.5 Storing the Configured Image

Once you’re satisfied with the state of the running instance, you can prepare for restart. To avoid losing all configuration changes to the instantiated AMI, create your own AMI, using the provided **ec2-rhel-bundle** script, which should be accessible from the root command prompt. This script creates your custom image, storing it in S3 persistent storage, as well as disabling RHN access for any regular virtual instances that you might start in the future. Use the resulting custom image for all future virtual instances.

The script requires you to provide your account number, your access keys, your private key, and your certificate. All of this information can be located at the “AWS Access Identifiers” section of the Amazon Web Services web site. The private key and certificate are RSA encoded files that need to be uploaded to the instance via scp. Amazon suggests you place these files into /tmp or /mnt so they are not bundled into your maintenance image.

7.6 Future Configuration & Maintenance of Images

As previously mentioned, if you update or somehow change a running instance, these changes are lost after the instance restarts. Furthermore, if you perform an update, you risk making your RHN profile unusable in the future. Therefore, the proper way to do those changes is to designate one running instance as the maintenance instance, and perform all tasks on it. After completion, build a new custom image, as described above, and use it for all your future virtual instances. To designate an instance as maintenance, use the provided script ec2-rhel-maintenance, which is also accessible from the root command prompt inside the virtual instance. The script is very simple, and it enables the previously disabled RHN access. It is highly recommended that you restart your running virtual instances using the newly built image, to propagate the changes made in the maintenance instance.

7.7 Kernel Update Notice

EC2 currently doesn’t permit changing kernels associated with the available image, rather a new AMI is provided that is associated with appropriate kernels.

Part of the service of Red Hat Enterprise Linux Hosted is to provide new AMIs whenever a relevant updated kernel is available. During the BETA period, whenever a new kernel becomes available to Red Hat Enterprise Linux through RHN, Red Hat Enterprise Linux Hosted Subscribers also receive access to a new AMI contained within EC2. While users of the EC2 provided AMIs do not have the ability to upgrade kernels of running instances or stored AMIs, customers are encouraged to migrate to the latest version of the AMI whenever a kernel and resulting new AMI is available.

7.8 Pay for What You Use

You can now buy RHEL support with preconfigured AMI's on Amazon EC2 directly from Amazon. If building your own AMI is undesirable, Amazon and third parties provide AMIs you may use. The AMIs built by Amazon are free, while some built by third parties can be used for a fee. Amazon has provided a means of measuring and billing for instance time used by AMI instances. The goal is to attract vendors to build AMIs for specific purposes and to allow them to sell the rights to use them.

Red Hat has recently made available its Enterprise Linux software subscription with full support on Amazon's EC2 platform. RHEL on Amazon EC2 is available for a base price of $19 per month, per user and $0.21, $0.53 or $0.94 for every compute hour used on Amazon's EC2 service, depending on whether customers choose a small, large or extra-large compute instance size, plus network bandwidth and storage fees.

8. A Step-by-Step Guide to Setting up and Running RHEL AMIs in EC2 Instances

This section will take you through setting up an account on Amazon to use the EC2 Compute Cloud.

Red Hat's Linux Automation strategy promotes running any application, any-where, anytime including: running applications on-premises (using physical and virtual) servers, running applications in application/server hosted facilities, and finally running application in the cloud, i.e., Amazon's Elastic Compute Cloud (Amazon EC2).

Demonstrating the EC2 cloud can help our customers grasp Red Hat's Linux Automation Strategy. In addition, the EC2 cloud gives our sales and sales support persons & solution architects & professional service consultants a virtual lab environment that is always available if we have a network connection.

8.1 Setting up your Amazon Web Services (AWS) Account

You can use an existing Amazon account, or setup a new account solely for AWS:

1. Go to: http://aws.amazon.com, and select **Sign-up today**
2. Use your existing amazon account, or create a new account for AWS.
3. After signing up, under Next Steps for Using Amazon Web Services, select **Amazon Simple Storage Services** (or go to http://aws.amazon.com/s3)
4. Select **Sign Up for This Web Service**
5. Review Pricing and enter your Credit Card Information
6. Go to http://aws.amazon.com/ec2 , and select **Sign Up for This Web Service**
7. Complete your signup for the Amazon EC2 Web Service
8. On the Thank You page, select **Create a New X.509 Certificate**
9. Select **Yes** to create a new Certificate
10. Download your **Private Key** file and your **Certificate** file (for this example, I will place them in ~/ec2)
11. Note your AWS account ID # by going to http://aws.amazon.com/ec2 moving the

mouse over **Your Web Services Account**, and select **Your Account Activity**. Your Account Number will be at the top of this page in XXXX-XXXX-XXXX format.

8.2 Obtaining Tools and Setting Up Environment

1. Make sure you have a 1.5 compatible JVM installed, and that your JAVA_HOME environment variable is set correctly[1]

2. Go to the Amazon EC2 Resource Center: http://developer.amazonwebservices.com/connect/entry.jspa?externalID=351&categoryID=88 and click **Download the Amazon EC2 Command-Line Tools**

3. Unzip the tools (for this document, expanded files are placed in ~/ec2/ec2-api-tools)

4. Setup environment variables using certs downloaded in step 10 above. (optionally place in shell profile, or a script):

```
export EC2_HOME=$HOME/ec2/ec2-api-tools
export PATH=$PATH:$EC2_HOME/bin
export EC2_PRIVATE_KEY=$HOME/ec2/pk-XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX.pem
export EC2_CERT=$HOME/ec2/cert-XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX.pem
```

-

5 Sanity check now to make sure your environment is setup correctly. Run the following command and you should see similar output if you are configured correctly:

```
[localhost ec2]$ ec2-describe-images -o self -o amazon
IMAGE   ami-20b65349    ec2-public-images/fedora-core4-base.manifest.xml      amazon   available      public
IMAGE   ami-22b6534b    ec2-public-images/fedora-core4-mysql.manifest.xml     amazon   available      public
IMAGE   ami-23b6534a    ec2-public-images/fedora-core4-apache.manifest.xml    amazon   available      public
...
```

These are the available system images (AMIs) that are available from Amazon publicly. If you create any of your own AMIs, they will also be listed here.

Note: To get access to the RHEL5 base AMIs, send an e-mail to cloud-list@redhat.com , and provide your Account Number noted in step 11 above.

6. To use a public AMI, you will need to use a public/private keypair. Run the following command:

```
[localhost ec2]$ ec2-add-keypair ec2-keypair
```

7. Take the resulting output private key text (everything between and including the "-----BEGIN RSA PRIVATE KEY-----" and "-----END RSA PRIVATE KEY-----") , and save to a txt file. For this document, ~/ec2/id_rsa-ec2-keypair

8. Change permissions on the private key file:

```
[localhost ec2]$ chmod 600 id_rsa-ec2-keypair
```

1 http://docs.amazonwebservices.com/AWSEC2/2007-08-29/GettingStartedGuide/prerequisites.html#java-runtime

8.3 Starting and Stopping Instances

1. In Step 5 above, the output listed public AMIs available, and the names are formatted as ami-XXXXXXXX. Once you have access to the RHEL5 base AMIs, the names are as follows:

   32-bit RHEL5 test
   ami-97d035fe

   64-bit RHEL5 test
   ami-68d33601

2. Use one of the RHEL5 AMIs or one of the public ones, and run the following command:

```
[localhost ec2]$ ec2-run-instances ami-97d035fe -k ec2-keypair
RESERVATION     r-cff41ea6      265404266339    default
INSTANCE        i-72fe0d1b      ami-97d035fe                    pending ec2-keypair     0               m1.small
2008-01-21T20:52:58+0000
```

   You now have a RHEL5 (or your selected AMI) image being created.

3. To check the status of your instance, run the following:

```
[localhost ec2]$ ec2-describe-instances
RESERVATION     r-cff41ea6      265404266339    default

INSTANCE        i-72fe0d1b      ami-97d035fe    ec2-67-202-39-96.compute-1.amazonaws.com
domU-12-31-39-00-50-53.compute-1.internal       running ec2-keypair     0               m1.small
 2008-01-21T20:52:58+0000
```

   The system is ready when you see output similar to above where a hostname is provided, and its status is running. The name of the instance is listed as well, in the above example i-72fe0d1b.

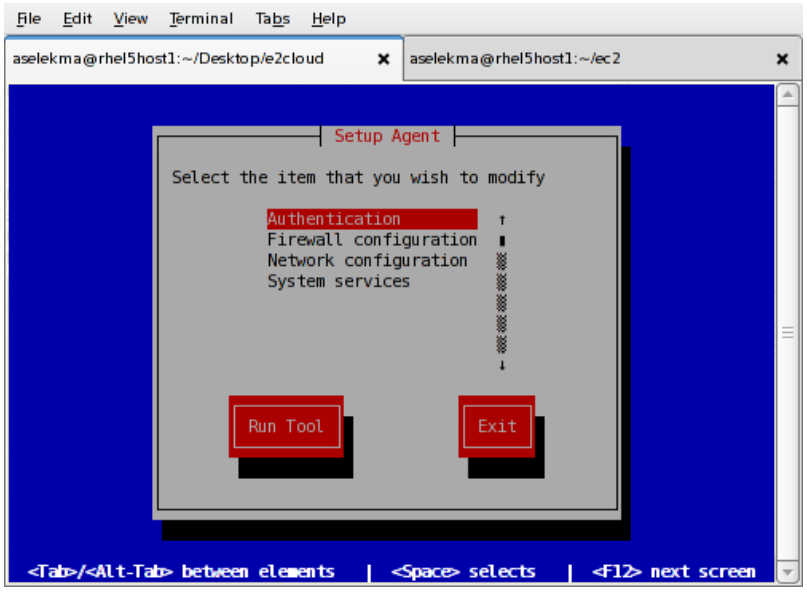
4. Enable port 22 to your default group (where this instance is currently running):

```
[localhost ec2]$ ec2-authorize default -p 22
PERMISSION      default  ALLOWS  tcp     22      22      FROM    CIDR    0.0.0.0/0
```

5. You can now ssh to your running instance using the private ssh key you created in the last section:

```
[localhost ec2]$ ssh -i id_rsa-ec2-keypair ec2-67-202-39-96.compute-1.amazonaws.com -l root
RSA key fingerprint is 71:14:41:cf:75:f3:2a:a2:ee:e8:8e:6e:f7:f7:07:65.
Are you sure you want to continue connecting (yes/no)? yes
```

6. If running the RHEL5 image, you will be in a text-based firstboot to setup the server and register it with RHN.



7. You now have a fully functional RHEL5 (or other) virtual server available to you. When you are done, shut down the machine with the following command:

```
[localhost e2]$ ec2-terminate-instances i-72fe0d1b
INSTANCE        i-72fe0d1b      running shutting-down
```

Note: All data is lost when an AMI is shut down. To have persistent storage, you must configure your machines to use Amazon's S3 storage. A step-by-step guide to using Amazon S3 is out of scope of this document.

9. References

1. Getting Started Guide: Red Hat Enterprise Linux on Amazon EC2
   http://www.redhat.com/f/pdf/EC2GettingStarted.pdf

2. http://www.redhat.com/solutions/cloud/gettingstarted/

3. Amazon EC2 RHEL5 Quick Start - Aryeh Selekman, Red Hat Solution Architect

4. Amazon Elastic Compute Cloud (Amazon EC2)
   http://aws.amazon.com/ec2

5. Amazon Simple Storage Service (Amazon S3)
   http://aws.amazon.com/ec2

6. Amazon Simple Queue Service (Amazon SQS)
   http://aws.amazon.com/sqs

7. Amazon SimpleDB™
   http://aws.amazon.com/SimpleDB

8. Amazon Elastic Compute Cloud - Getting Started Guide
   http://docs.amazonwebservices.com/AWSEC2/2007-08-29/GettingStartedGuide