Web Application Hosting in the AWS Cloud

Best Practices

September 2012

Matt Tavis, Philip Fitzsimons

Abstract

Highly available and scalable web hosting can be a complex and expensive proposition. Traditional scalable web architectures have not only needed to implement complex solutions to ensure high levels of reliability, but they have also required an accurate forecast of traffic to provide a high level of customer service. Dense peak traffic periods and wild swings in traffic patterns result in low utilization rates of expensive hardware, yielding high operating costs to maintain idle hardware, as well as an inefficient use of capital for underutilized hardware.

Amazon Web Services provides a reliable, scalable, secure, and highly performing infrastructure for the most demanding web applications, an infrastructure that matches IT costs with customer traffic patterns in real time.

Audience

This whitepaper is for IT managers and system architects who are looking to the cloud to help them achieve the scalability that they need to meet their on-demand computing needs.

An Overview of Traditional Web Hosting

Scalable web hosting is a well-known problem space, so the following figure, which depicts a conventional web hosting model, should not contain any surprises. We present it for comparison with a similar architecture that is implemented in the cloud. It is the cloud deployment that is the subject of this paper.

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

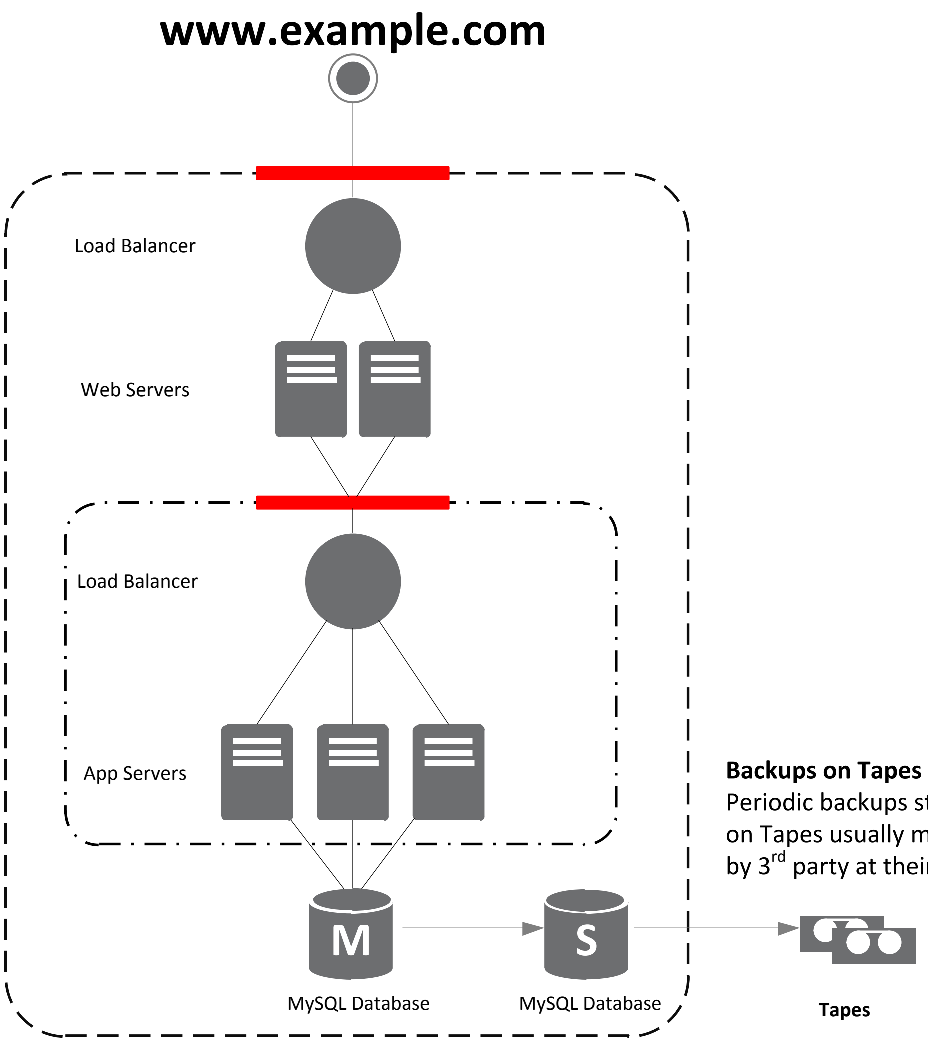
Web Tier
Fleet of machines handling HTTP requests

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



Backups on Tapes
Periodic backups stored on Tapes usually managed by 3rd party at their site

Figure 1 A traditional web application architecture

This traditional web hosting architecture is implements a common three-tier web application model that separates the architecture into presentation, application, and persistence layers. Scalability is provided by adding additional hosts at the presentation, persistence, or application layers. The architecture also has built-in performance, failover, and availability features. The following sections will look at why and how such an architecture should be and could be deployed in the Amazon Web Services cloud.

Web Application Hosting in the Cloud using Amazon Web Services

The traditional web hosting architecture (Figure 1) is easily portable to the cloud services provided by the AWS products with only a small number of modifications, but the first question that should be asked concerns the value of moving a classic web application hosting solution into the AWS cloud. If you decide that the cloud is right for you, you'll need a suitable architecture. This section helps you evaluate an AWS cloud solution. It compares deploying your web application in the cloud to an on-premises deployment, presents an AWS cloud architecture for hosting your application, and discusses the key components of this solution.

How AWS Can Solve Common Web Application Hosting Issues

If you are responsible for running a web application, you face a variety of infrastructure and architectural issues for which AWS can provide seamless and cost-effective solutions. The following are just some of the benefits of using AWS over a traditional hosting model:

A Cost-Effective Alternative to Oversized Fleets Needed to Handle Peaks

In the traditional hosting model, servers need to be provisioned to handle peak capacity, and unused cycles are wasted outside of peak periods. AWS-hosted web applications can leverage on-demand provisioning of additional servers, so you can constantly adjust capacity and costs to actual traffic patterns.

For example, the following graph shows a web application with a usage peak from 9 AM to 3 PM and less usage for the remainder of the day. An auto-scaling approach based on actual traffic trends, which provisions resources only when needed, would result in less wasted capacity and a cost reduction of greater than 50%.

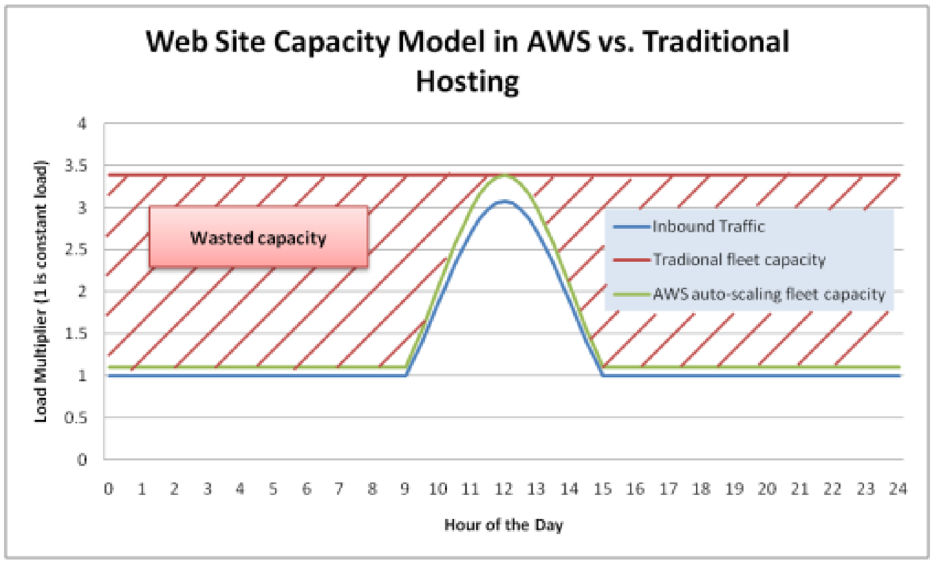


Figure 2 - An Example of Wasted Capacity in a Classic Hosting Model

A Scalable Solution to Handling Unexpected Traffic Peaks

An even more dire consequence of the slow provisioning associated with a traditional hosting model is the inability to respond in time to unexpected traffic spikes. There are many stories about web applications going down because of an unexpected spike in traffic after the site is mentioned in the popular media. The same on-demand capability that helps web applications scale to match regular traffic spikes can also handle an unexpected load. New hosts can be launched and ready in a matter of minutes, and they can be taken offline just as quickly when traffic returns to normal.

An On-Demand Solution for Test, Load, Beta and Pre-Production Environments

The hardware costs of building out a traditional hosting environment for a production web application don't stop with the production fleet. Quite often pre-production, beta, and testing fleets also need to be created to ensure the quality of the web application at each stage of the development lifecycle. While various optimizations can be made to ensure the highest possible utilization of this testing hardware, these parallel fleets are not always utilized optimally: a lot of expensive hardware sits unused for long periods of time. In the AWS cloud, you can provision testing fleets only when you need them. Additionally, you can simulate user traffic on the AWS cloud during load testing. You can also use these parallel fleets as a staging environment for a new production release, which allows for quick switchover from current production to a new application version with little or no service outages.

An AWS Cloud Architecture for Web Hosting

Below is another look at that classic web application architecture and how it could leverage the AWS cloud computing infrastructure:

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

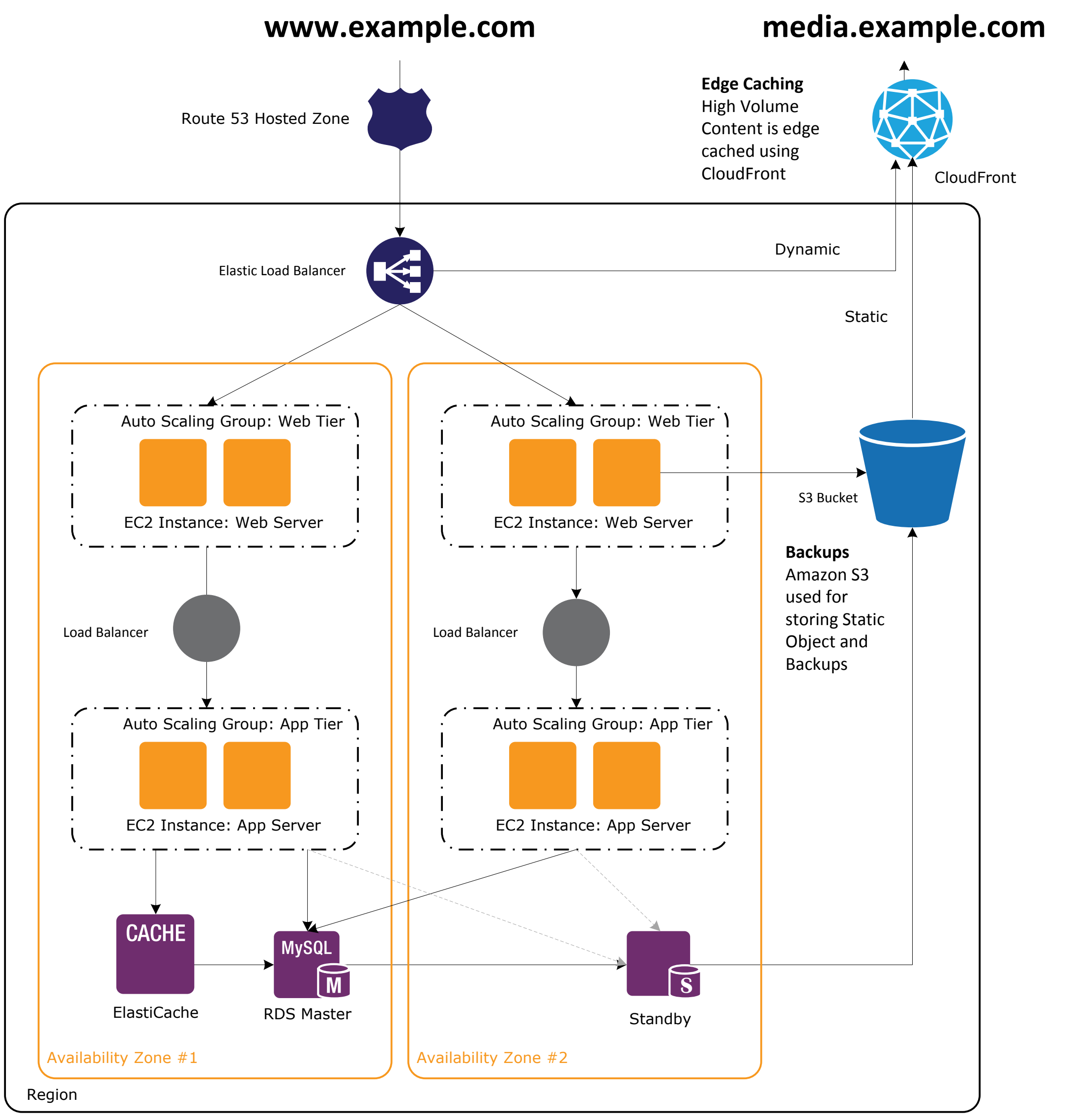


Figure 3 - An Example of a Web Hosting Architecture on AWS

Key Components of an AWS Web Hosting Architecture

The following sections outline some of the key components of a web hosting architecture deployed in the AWS cloud and explain how they differ from a traditional web hosting architecture.

Content Delivery

Edge caching is still relevant in the Amazon Web Service cloud computing infrastructure. Any existing solutions in your web application infrastructure should work just fine in the AWS cloud. One additional option, however, is made available when using AWS, which is to utilize the Amazon CloudFront service[1] for edge caching your website.

Amazon CloudFront can be used to deliver your website, including dynamic, static and streaming content using a global network of edge locations. Requests for your content are automatically routed to the nearest edge location, so content is delivered with the best possible performance. Amazon CloudFront is optimized to work with other Amazon Web Services, like Amazon Simple Storage Service[2] (Amazon S3) and Amazon Elastic Compute Cloud[3] (Amazon EC2). Amazon CloudFront also works seamlessly with any non-AWS origin server, which stores the original, definitive versions of your files.

Like other Amazon Web Services, there are no contracts or monthly commitments for using Amazon CloudFront – you pay only for as much or as little content as you actually deliver through the service.

Managing Public DNS

Moving a web application to the AWS cloud requires some DNS changes to take advantage of the multiple availability zones that AWS provides. To help you manage DNS routing, AWS provides Amazon Route 53[4], a highly available and scalable DNS web service. Queries for your domain are automatically routed to the nearest DNS server and thus are answered with the best possible performance. Route 53 resolves requests for your domain name (for example, www.example.com) to your Elastic Load Balancer, as well as your zone apex record (example.com).

Host Security

Unlike a traditional web hosting model, inbound network traffic filtering should not be confined to the edge; it should also be applied at the host level. Amazon Elastic Compute Cloud (EC2) provides a feature called *security groups*. A security group is analogous to an inbound network firewall, for which you to specify the protocols, ports, and source IPs ranges that are allowed to reach your EC2 instances. Each EC2 instance can be assigned one or more security groups, each of which routes the appropriate traffic to each instance. Security groups can be configured so that only specific subnets or IP addresses have access to an EC2 instance, or they can reference other security groups to limit access to EC2 instances that are in specific groups.

[1] Amazon CloudFront - http://aws.amazon.com/cloudfront/
[2] Amazon S3 – http://aws.amazon.com/s3
[3] Amazon EC2 – http://aws.amazon.com/ec2/
[4] Amazon Route 53 – http://aws.amazon.com/route53/

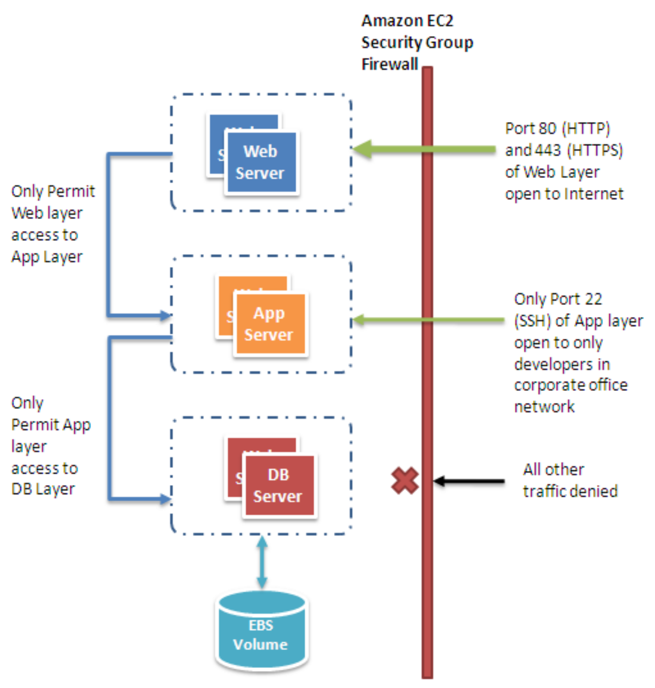

Amazon EC2
Security Group
Firewall
Web Server
App Server
DB Server
EBS Volume
Only Permit Web layer access to App Layer
Only Permit App layer access to DB Layer
Port 80 (HTTP) and 443 (HTTPS) of Web Layer open to Internet
Only Port 22 (SSH) of App layer open to only developers in corporate office network
All other traffic denied

Figure 4: Security Groups in a Web Application

For instance, in the example AWS web hosting architecture above, the security group for the web server cluster might allow access for any host only over TCP on ports 80 and 443 (HTTP and HTTPS) and from instances in the application server security group on port 22 (SSH) for direct host management. The security group of the application server cluster, on the other hand, might allow access from the Web Server security group for handling web requests and from your organization's subnet over TCP on port 22 (SSH) for direct host management. In this model, shown above, your support engineers could log in directly to the application servers from the corporate network and then access the other clusters from the application server boxes. For a deeper discussion on security, go to the AWS Security Center[5]. The center contains security bulletins, certification information, and security whitepapers that explain the security capabilities of AWS.

Load balancing across clusters

Hardware load balancers are a common network appliance used in traditional web application architectures. AWS provides this capability through the Elastic Load Balancing[6] service, a configurable load-balancing solution that supports health checks on hosts, distribution of traffic to EC2 instances across multiple availability zones, and dynamic addition and removal of Amazon EC2 hosts from the load-balancing rotation. Elastic Load Balancing can also dynamically grow and shrink the load-balancing capacity to adjust to traffic demands while providing a predictable entry point by using a persistent CNAME. The Elastic Load Balancing service also supports sticky sessions to address more advanced routing needs. If your application requires more advanced load-balancing capabilities you could run a software load-balancing

[5] AWS Security Center – http://aws.amazon.com/security
[6] Amazon Elastic Load Balancing - http://aws.amazon.com/elasticloadbalancing

package (e.g., Zeus, HAProxy, or nginx) on EC2 instances . You could then assign Elastic IP[7] addresses to those load-balancing EC2 instances in order to minimize DNS changes.

Finding other hosts and services

In the traditional web hosting architecture, most of your hosts have static IP addresses; in the cloud, most of your hosts will have dynamic IP addresses. Although every EC2 instance can have both public and private DNS entries and will be addressable over the Internet, the DNS entries and the IP addresses are assigned dynamically when you launch the instance, and they cannot be manually assigned. Static IP addresses (Elastic IP addresses in AWS terminology) can be assigned to running instances after they are launched. You should use Elastic IP addresses for instances and services that require consistent endpoints, such as, master databases, central file servers, and EC2-hosted load balancers.

Server roles that can easily scale out and in, such as web servers, should be made discoverable at their dynamic endpoints by registering their IP address with a central repository. Because most web application architectures have a database server that is always on, the database server is a common repository for discovery information. For situations where consistent addressing is needed instances can be allocated an Elastic IP addresses from a pool of addresses by a bootstrapping script when the instance is launched.

Using this model, newly added hosts can request the list of necessary endpoints for communications from the database as part of a bootstrapping phase. The location of the database can be provided as user data [8] that is passed into each instance as it is launched. Alternatively, you can use the Amazon SimpleDB service to store and maintain configuration information. Amazon SimpleDB is a highly-available service that is available at a well-known endpoint.

Caching within the web application

In-memory application caches can reduce load on services and improve performance and scalability on the database tier by caching frequently used information. Amazon ElastiCache[9] is a web service that makes it easy to deploy, operate, and scale an in-memory cache in the cloud. The in-memory cache that you create can be configured to automatically scale with load and to automatically replace failed nodes. Amazon ElastiCache is protocol-compliant with Memcached, which simplifies migration from your current on-premises solution.

[7] Elastic IP addresses are static IP addresses designed for dynamic cloud computing, that you can move from one instance to another
[8] User Data - http://docs.amazonwebservices.com/AWSEC2/latest/APIReference/index.html?ApiReference-ItemType-RunInstancesType.html
[9] Amazon ElastiCache – http://aws.amazon.com/elasticache

Database configuration, backup and failover

Many web applications contain some form of persistence, usually in the form of a relational or NoSQL database. AWS offers both relational and NoSQL database infrastructure. Alternatively, you can deploy your own database software on an Amazon EC2 instance. The following table summarizes these options, which are then discussed in greater detail.

	Relational Database Solutions	NoSQL Solutions
Managed Database Service	Amazon Relational Database Service (RDS) – MySQL, Oracle, SQL Server	Amazon DynamoDB
Self-Managed	Hosting a relational DBMS on an Amazon EC2 instance	Hosting a NoSQL solution on an Amazon EC2 instance

Amazon Relational Database Service (RDS)

Amazon RDS gives you access to the capabilities of a familiar MySQL, Oracle, or Microsoft SQL Server database engine. The code, applications, and tools that you are already using can be used with Amazon RDS. Amazon RDS automatically patches the database software and backs up your database, and it stores backups for a user-defined retention period. It also supports point-in-time recovery. You benefit from the flexibility of being able to scale the compute resources or storage capacity associated with your relational database instance by making a single API call.

In addition, Amazon RDS Multi-AZ deployments increase your database availability and protect your database against unplanned outages. Amazon RDS Read Replicas provide read-only replicas of your database, so you can scale out beyond the capacity of a single database deployment for read-heavy database workloads. As with all Amazon Web Services, there are no up-front investments required, and you pay only for the resources you use.

Hosting a relational database (RDBMS) on an Amazon EC2 instance

In addition to the managed Amazon RDS offering, you can install your choice of RDBMS (such as MySQL, Oracle, SQL Server, or DB2) on an EC2 instance and manage it yourself. AWS customers hosting a database on Amazon EC2 have successfully used a variety of master/slave and replication models, including mirroring for read-only copies and log shipping for always-ready passive slaves.

An additional consideration when managing your own database software directly on Amazon EC2 is the availability of fault-tolerant and persistent storage. For this purpose, we recommend that databases running on Amazon EC2 utilize Amazon Elastic Block Storage (Amazon EBS) volumes, which are similar to network-attached storage. For EC2 instances running a database, all database data and logs should be placed on Amazon EBS volumes, which will remain available even if the database host fails. This configuration allows for a simple failover scenario, in which a new EC2 instance can be launched if a host fails, and the existing Amazon EBS volumes can be attached to the new instance. The database can then pick up where it left off.

Amazon EBS volumes automatically provide redundancy within the Availability Zone, which increases their availability over simple disks. If the performance of a single Amazon EBS volume is not sufficient for your databases needs, volumes can be striped to increase IOPS performance for your database. For demanding workloads you can also use EBS

Provisioned IOPS, where you specify the IOPS required. If you use Amazon RDS, the service manages its own storage so you can focus on managing your data.

NoSQL solutions

In addition to support for relational databases, AWS also offers Amazon DynamoDB, a fully managed NoSQL database service that provides fast and predictable performance with seamless scalability. Using the AWS Management Console or the Amazon DynamoDB API, you can scale capacity up or down without downtime or performance degradation. Because Amazon DynamoDB handles the administrative burdens of operating and scaling distributed databases to AWS, you don't have to worry about hardware provisioning, setup and configuration, replication, software patching, or cluster scaling.

Amazon SimpleDB provides a lightweight, highly available, and fault-tolerant core non-relational database service that offers querying and indexing of data without the requirement of a fixed schema. SimpleDB can be a very effective replacement for databases in data access scenarios that require one big, highly-indexed and flexible schema table.

Additionally, Amazon EC2 can be used to host many other emerging technologies in the NoSQL movement, such as Cassandra, CouchDB, and MemcacheDB.

Storage and backup of data and assets

There are numerous options within the AWS cloud for storing, accessing, and backing up your web application data and assets. The Amazon Simple Storage Service (Amazon S3) provides a highly available and redundant object store. Amazon S3 is a great storage solution for somewhat static or slow-changing objects, such as images, videos, and other static media. Amazon S3 also supports edge caching and streaming of these assets by interacting with the Amazon CloudFront service.

For attached file system like storage, EC2 instances can have Amazon Elastic Block Storage volumes attached, which can act like mountable disks for running EC2 instances. Amazon EBS is great for data that needs to be accessed as block storage and that requires persistence beyond the life of the running instance, such as database partitions and application logs.

In addition to having a lifetime that is independent of the EC2 instance, snapshots of Amazon EBS volumes can be taken and stored in Amazon S3. Because EBS snapshots back up only changes since the previous snapshot, more frequent snapshots can reduce snapshot times. You can also use an Amazon EBS snapshot as a baseline for replicating data across multiple Amazon EBS volumes and attaching those volumes to other running instances.

Amazon EBS volumes can be as large as 1 TB, and multiple Amazon EBS volumes can be striped for even larger volumes or for increased I/O performance. To maximize the performance of your I/O-intensive applications, you can use Provisioned IOPS volumes. Provisioned IOPS volumes are designed to meet the needs of I/O-intensive workloads, particularly database workloads that are sensitive to storage performance and consistency in random access I/O throughput. You specify an IOPS rate when you create the volume and Amazon EBS provisions that rate for the lifetime of the volume. Amazon EBS currently supports up to 1,000 IOPS per volume. You can stripe multiple volumes together to deliver thousands of IOPS per instance to your application.

Auto Scaling the fleet

One of the key differences between the AWS cloud architecture and the traditional hosting model is that AWS can dynamically scale the web application fleet on demand to handle changes in traffic. In the traditional hosting model, traffic forecasting models are generally used to provision hosts ahead of projected traffic. In AWS, instances can be provisioned on the fly according to a set of triggers for scaling the fleet out and back in. Amazon Auto Scaling can create capacity groups of servers that can grow or shrink on demand. Auto Scaling also works directly with Amazon CloudWatch for metrics data and with the Elastic Load Balancing service to add and remove hosts for load distribution. For example, if the web servers are reporting greater than 80% CPU utilization over a period of time, an additional web server could be quickly deployed and then automatically added to the Elastic Load Balancer for immediate inclusion in the load-balancing rotation.

As shown in the AWS web hosting architecture model, multiple auto-scaling groups can be created for different layers of the architecture so that each layer can scale independently. For example, the web server auto-scaling group might trigger scaling out and in in response to changes in network I/O, whereas the application server auto-scaling group might scale out and in according to CPU utilization. You can set minimums and maximums to help ensure 24/7 availability and to cap to usage within a group.

Auto Scaling triggers can be set both to grow and to shrink the total fleet at a given layer to match resource utilization to actual demand. In addition to the Auto Scaling service, Amazon EC2 fleets can be scaled directly through the Amazon EC2 API, which allows for launching, terminating, and inspecting instances.

Failover with AWS

Another key advantage of AWS over traditional web hosting is the Availability Zones that give you easy access to redundant deployment locations. Availability Zones are physically distinct locations that are engineered to be insulated from failures in other Availability Zones. They provide inexpensive, low-latency network connectivity to other Availability Zones in the same Region. As the AWS web hosting architecture diagram in this paper shows, we recommend that you deploy EC2 hosts across multiple Availability Zones to make your web application more fault-tolerant. It's important to ensure that there are provisions for migrating single points of access across Availability Zones in the case of failure. For example, a database slave should be set up in a second Availability Zone so that the persistence of data remains consistent and highly available even during an unlikely failure scenario.

While some architectural changes are often required while moving an existing web application to the AWS cloud, there are significant improvements to scalability, reliability, and cost-effectiveness that make using the AWS cloud well worth the effort. In the next section, we'll discuss those improvements.

Key Considerations when Using AWS for Web Hosting

There are some key differences between the AWS cloud and a traditional web application hosting model. The previous section highlighted many of the key areas that you should consider when deploying a web application to the cloud. This section points out some of the key architectural shifts that you need to consider when you bring any application into the cloud.

No more physical network appliances

You cannot deploy physical network appliances in AWS. For example, firewalls, routers, and load-balancers for your AWS applications can no longer reside on physical devices but must be replaced with software solutions. There is a wide variety of enterprise quality software solutions, whether for load balancing (e.g., Zeus, HAProxy, nginx, and Pound) or establishing a VPN connection (e.g., OpenVPN, OpenSwan, and Vyatta). This is not a limitation of what can be run on the AWS cloud, but it is an architectural change to your application if you use these devices today.

Firewalls everywhere

Where you once had a simple DMZ and then open communications among your hosts in a traditional hosting model, AWS enforces a more secure model, in which every host is locked down. One of the steps in planning an AWS deployment will be the analysis of traffic between hosts. This analysis will guide decisions on exactly what ports need to be opened. Security Groups within Amazon EC2 can be created for each type of host in your architecture, and a large variety of simple and tiered security models can be created to enable the minimum access among hosts within your architecture.

Multiple data centers available

Availability Zones within an AWS region should be thought of as multiple datacenters. EC2 instances in different Availability Zones are both logically and physically separated, and they provide an easy-to-use model for deploying your application across data centers for both high availability and reliability.

Treat hosts as ephemeral and dynamic

Probably the most important shift in how you might architect your AWS application is that Amazon EC2 hosts should be considered ephemeral and dynamic. Any application built for the AWS cloud should not assume that a host will always be available and should be designed in the knowledge that any data that is not on an Amazon EBS volume will be lost if an EC2 instance fails. Additionally, when a new host is brought up, no assumptions should be made about the IP address or location within an Availability Zone of the host. Your configuration model must be flexible, and your approach to bootstrapping a host must take the dynamic nature of the cloud into account. These techniques are critical for building and running a highly scalable and fault-tolerant application.

Closing Thoughts

There are numerous architectural and conceptual considerations when contemplating the migration of a web application into the AWS cloud, but the benefits of having a cost-effective, highly-scalable and fault-tolerant infrastructure that grows with your business far outstrips the efforts of migrating to the AWS cloud.

Additional Getting Started Resources

1. Getting started guide – AWS Web Application hosting for Linux
   http://docs.amazonwebservices.com/gettingstarted/latest/wah-linux/
2. Getting started guide – AWS Web Application hosting for Windows
   http://docs.amazonwebservices.com/gettingstarted/latest/wah/
3. Getting started Video Series: Linux Web Applications in the AWS Cloud
   http://aws.amazon.com/web-applications/gsg-webapps-linux/
4. Getting started Video Series: .NET Web Applications in the AWS Cloud
   http://aws.amazon.com/web-applications/gsg-webapps-windows/

Changes since last version (May 2010)

- Multiple sections updated to improve clarity
- Updated diagrams to use AWS icons
- Addition of Managing Public DNS section for detail on Route 53
- Finding other hosts and services section updated for clarity
- Database configuration, backup and failover updated for clarity and DynamoDB
- Storage and backup of data and assets expanded to cover EBS Provisioned IOPS volumes