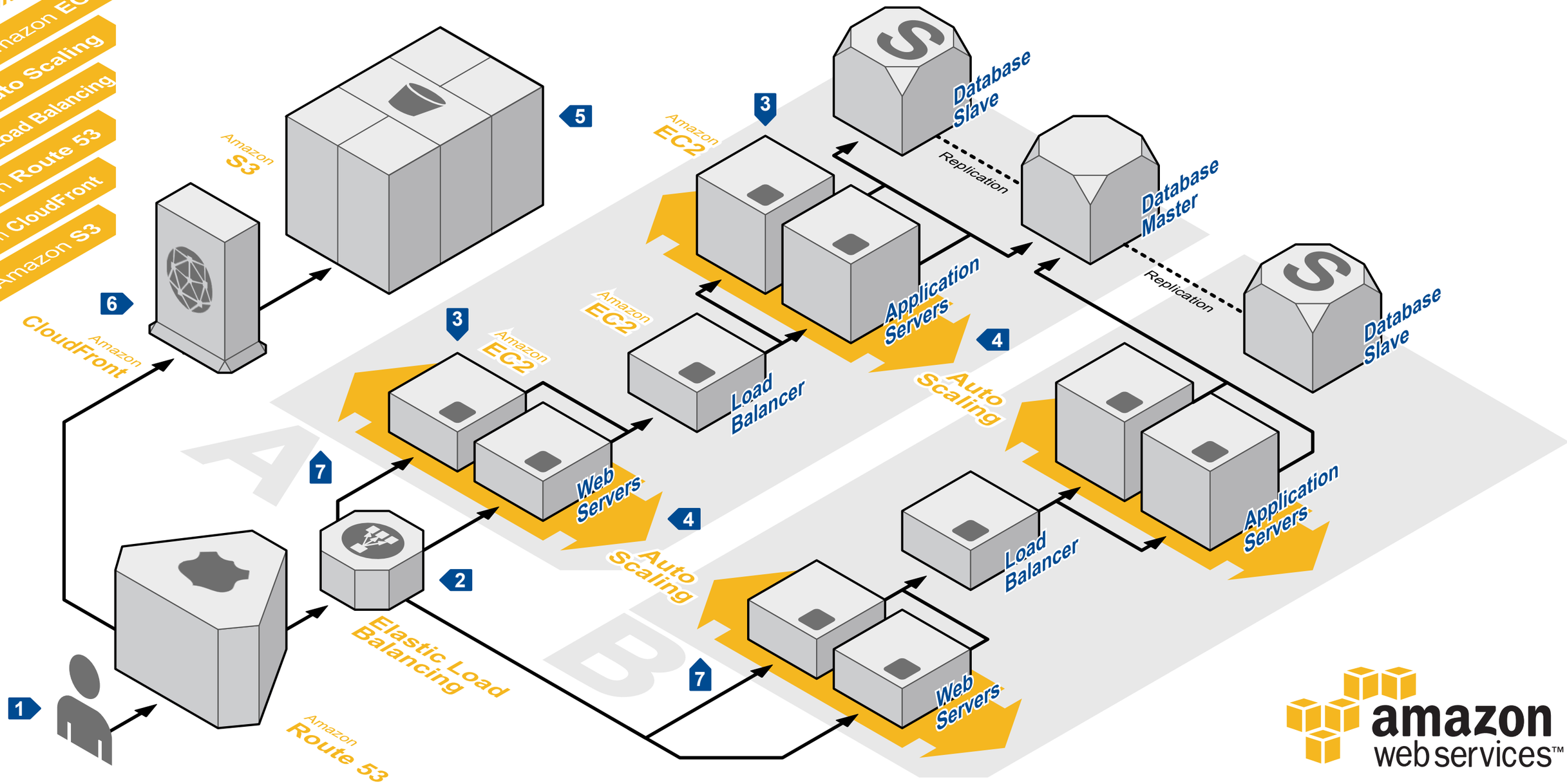
AWS Reference Architectures

- Amazon EC2
- Auto Scaling
- Elastic Load Balancing
- Amazon Route 53
- Amazon CloudFront
- Amazon S3

WEB APPLICATION HOSTING

Highly available and scalable web hosting can be complex and expensive. Dense peak periods and wild swings in traffic patterns result in low utilization rates of expensive hardware. Amazon Web Services provides the reliable, scalable, secure, and high-performance infrastructure required for web applications while enabling an elastic, scale out and scale down infrastructure to match IT costs in real time as customer traffic fluctuates.



amazon web services™

System Overview

1 The user's DNS requests are served by **Amazon Route 53**, a highly available Domain Name System (DNS) service. Network traffic is routed to infrastructure running in Amazon Web Services.

2 HTTP requests are first handled by Elastic Load Balancing, which automatically distributes incoming application traffic across multiple **Amazon Elastic Compute Cloud (EC2)** instances across Availability Zones (AZs). It enables even greater fault tolerance in your applications, seamlessly providing the amount of load balancing capacity needed in response to incoming application traffic.

3 Web servers and application servers are deployed on **Amazon EC2** instances. Most organizations will select an **Amazon Machine Image (AMI)** and then customize it to their needs. This custom AMI will then be used as the starting point for future web development.

4 Web servers and application servers are deployed in an **Auto Scaling** group. Auto Scaling automatically adjusts your capacity up or down according to conditions you define. With Auto Scaling, you can ensure that the number of **Amazon EC2** instances you're using increases seamlessly during demand spikes to maintain performance and decreases automatically during demand lulls to minimize costs.

5 Resources and static content used by the web application are stored on **Amazon Simple Storage Service (S3)**, a highly durable storage infrastructure designed for mission-critical and primary data storage.

6 Static and streaming content is delivered by **Amazon CloudFront**, a global network of edge locations. Requests are automatically routed to the nearest edge location, so content is delivered with the best possible performance.

7 **Availability zones (AZs)** are distinct geographic locations that are engineered to insulate against failures in other AZs. Multiple AZs are combined into a region. Here, the entire web application is deployed in two different AZs for high availability.

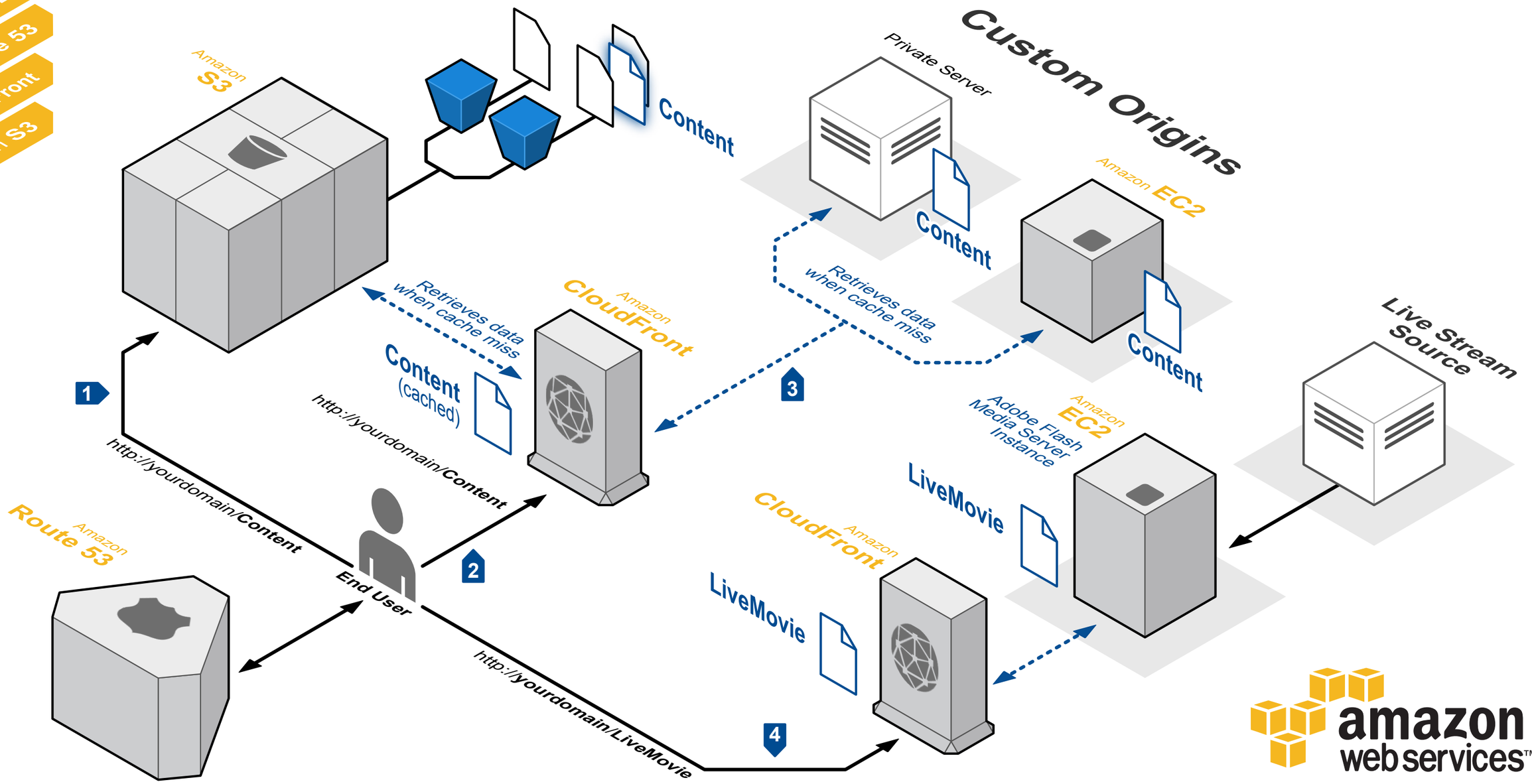
AWS Reference Architectures

- Amazon EC2
- Amazon Route 53
- Amazon CloudFront
- Amazon S3

CONTENT & MEDIA SERVING

Serving digital content is one of the most basic and straightforward tasks—that is, until you have serious requirements for low latency, high availability, durability, access control, and millions of views on or under budget. In addition, because of "spiky" usage patterns, operations teams often need to provision static hardware, network, and management resources to support the maximum expected need, which guarantees waste outside of peak hours.

AWS provides a suite of services specifically tailored to deliver high-performance media serving. Each service features pay as you go pricing on an elastic infrastructure, meaning that you can scale up and down according to your demand curve while paying for only the resources you use. Because this infrastructure is programmable, it can react quickly. Our advanced API provides detailed control over the infrastructure that powers your system.



System Overview

1 **Simple and Secure** — This reference architecture uses **Amazon Simple Storage Service (S3)** to host static content on the web. **Amazon S3** is highly available, highly durable, and designed for web scale. It provides a great way to offload the work of serving static content from your web servers. You can also provide secure access to this content over HTTPS.

2 **Faster and Edge Cached** — As your customer base grows and becomes more geographically distributed, using a high- performance edge cache like **Amazon CloudFront** can provide substantial improvements in latency, fault tolerance, and cost. By using **Amazon S3** as the origin server for the **Amazon CloudFront** distribution, you gain the advantages of fast in-network data transfer rates, simple publishing/caching workflow, and a unified security framework. **Amazon S3** and **Amazon CloudFront** can be configured by a web service, the AWS Management Console, or a host of third-party management tools.

3 Alternatively, you could use **Amazon Elastic Compute Cloud (EC2)** as origin server of **Amazon S3** for hosting static content. Using **Amazon EC2** could allow you a greater degree of control, logging, and feature richness in serving content. For static content, you could also substitute your own on-premises or cohosted private servers as origin servers for **Amazon CloudFront**.

4 **Live Streaming** — Featuring the power of Adobe Flash Media Server hosted on **Amazon EC2**, combined with **Amazon CloudFront** for stream distribution and caching, live streaming works seamlessly on the AWS platform. This configuration uses a web server to host the manifest.xml file, **Amazon DevPay** EC2 instances to host Flash Media Server with hourly license pricing, and **Amazon CloudFront** to serve the stream.

Read more here: **http://www.adobe.com/go/fmsaws**

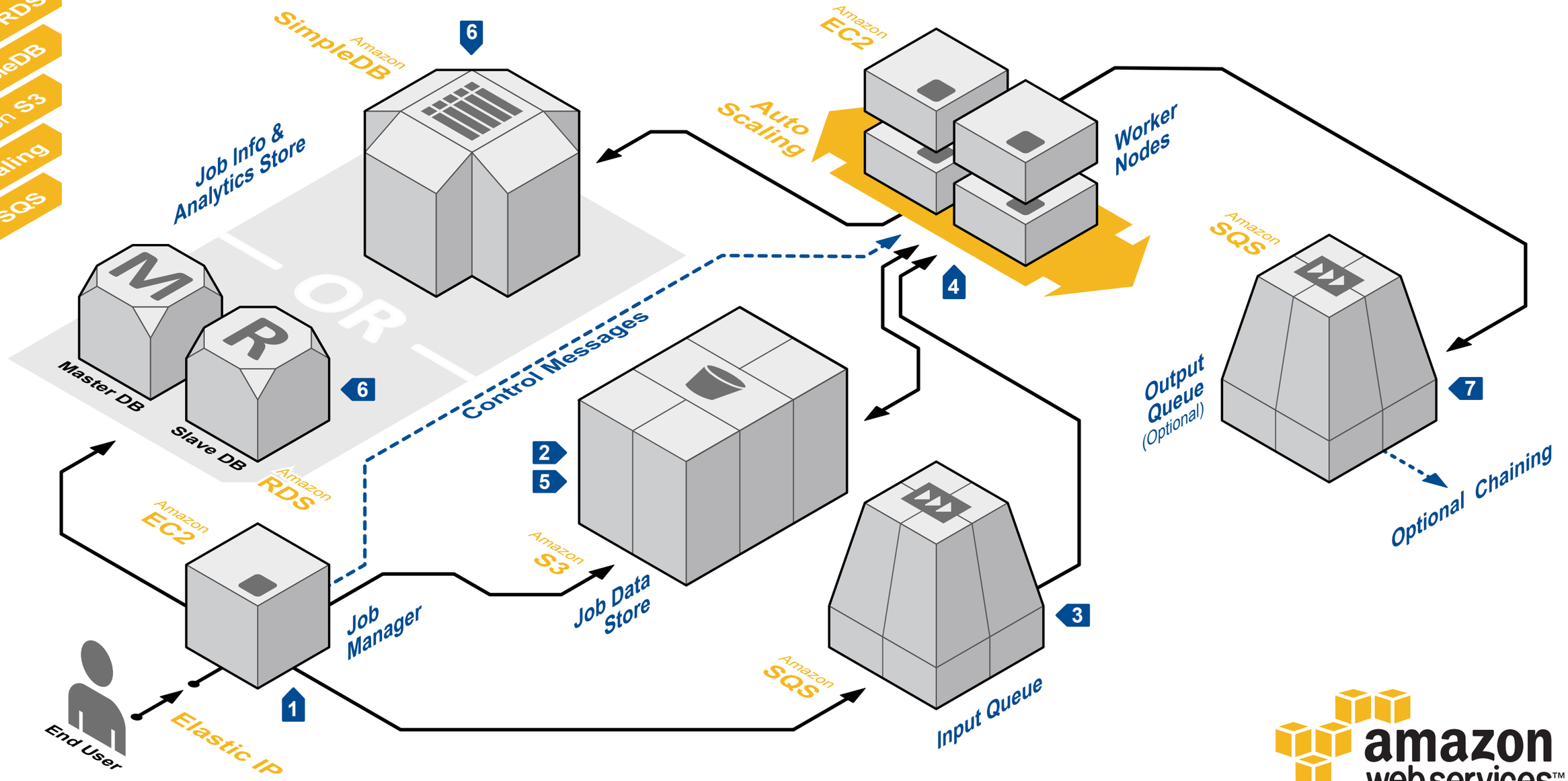
AWS Reference Architectures

- Amazon EC2
- Amazon RDS
- Amazon SimpleDB
- Amazon S3
- Auto Scaling
- Amazon SQS

BATCH PROCESSING

Batch processing on AWS allows for the on-demand provisioning of a multi-part job processing architecture that can be used for instantaneous or delayed deployment of a heterogeneous, scalable “grid” of worker nodes that can quickly crunch through large batch processing tasks in parallel. There are numerous batch oriented applications in place today that can leverage this style of on-demand processing, including claims processing, large scale transformation, media transcoding and multi-part data processing work.

Batch processing architectures are often synonymous with highly variable usage patterns that have significant usage peaks (e.g., month-end processing) followed by significant periods of underutilization.
There are numerous approaches to building a batch processing architecture. This document outlines a basic batch processing architecture that supports job scheduling, job status inspection, uploading raw data, outputting job results, grid management, and reporting job performance data.



amazon web services™

System Overview

1 Users interact with the Job Manager application which is deployed on an **Amazon Elastic Computer Cloud (EC2)** instance. This component controls the process of accepting, scheduling, starting, managing, and completing batch jobs. It also provides access to the final results, job and worker statistics, and job progress information.

2 Raw job data is uploaded to **Amazon Simple Storage Service (S3)**, a highly-available and persistent data store.

3 Individual job tasks are inserted by the Job Manager in an **Amazon Simple Queue Service (SQS)** input queue on the user’s behalf.

4 Worker nodes are **Amazon EC2** instances deployed on an **Auto Scaling** group. This group is a container that ensures health and scalability of worker nodes. Worker nodes pick up job parts from the input queue automatically and perform single tasks that are part of the list of batch processing steps.

5 Interim results from worker nodes are stored in **Amazon S3**.

6 Progress information and statistics are stored on the analytics store. This component can be either an **Amazon SimpleDB** domain or a relational database such as an **Amazon Relational Database Service (RDS)** instance.

7 Optionaly, completed tasks can be inserted in an **Amazon SQS** queue for chaining to a second processing stage.

Amazon's Corporate IT Deploys SharePoint 2010 to the Amazon Web Services Cloud

September 2011

This paper describes how and why Amazon.com's corporate IT organization deployed a self-service portal platform for internal sites that uses Microsoft SharePoint 2010[1] and a SQL Server 2008 Enterprise[2] storage database to the Amazon Web Services (AWS) cloud. Throughout the process, we[3] engaged with AWS like any vendor would. In turn, AWS treated us as it would treat any enterprise customer.

This paper shares our story as an AWS reference customer of this six-month cloud deployment project. Our story is broken down into the following sections:

- Where We Were
- Where We Wanted to Go and Why
- Our Deployment Success Criteria
- How We Launched
- Lessons We Learned
- What Benefits Emerged
- Where We Go From Here

Where We Were

We launched SharePoint 2007 when it was released. We did this to provide teams worldwide with self-service portal site creation features and to give our corporate offices in various countries content publishing tools to manage our new global intranet site. Our employees use the self-service features to create and manage sites that help them accomplish many tasks, including controlling access to centrally managed documents, communicating to their customers, publishing processes and procedures, sharing ideas and knowledge, implementing business workflows, capturing and reporting on metrics, and other mission-critical use cases. Employees were using SharePoint to drive mission critical processes and store highly confidential corporate data.

Over the past three years, usage and capacity have exploded due the massive growth across all teams worldwide. More than 30,000 employees per month worldwide regularly use our corporate intranet. Employees create separate Team Sites to collaborate with other team mates and managers create "My Sites" for projects and programs. The system now consists of thousands of Team Sites and My Sites, several SQL Server database servers, and a few web and application servers for serving content, central administration, and search.

Due to the rapid growth of the service, corporate IT was looking to simplify the management of the hardware infrastructure, and the performing of common tasks such as lease returns, provisioning new hardware, supporting data center moves, and expanding SharePoint to support new geocentric requirements.

[1] See http://sharepoint.microsoft.com.
[2] See http://www.microsoft.com/sqlserver.
[3] Throughout this document, the term 'we' refers to Amazon corporate IT.

Just like any other central IT department of any large enterprise, we also spent several weeks procuring hardware and several hours installing and configuring Windows operating systems. Typically, our new hardware procurement was taking four to six weeks. Our team was spending approximately six hours per server rebuilding the hardware with the latest Windows operating system and patches. We were approaching the capacity limits of our current hardware and would need to make a significant investment to support continued growth "on-premise."

Additionally, the business was demanding enterprise features that were available only in SharePoint 2010. In discussions with Microsoft, we determined that multiple disruptive upgrades would have to take place on our production farm to support an in-place upgrade to SharePoint 2010, so it was a logical time to consider relocating the service.

Where We Wanted to Go and Why

Given our desire to reduce infrastructure administration costs and to be able to scale on demand, we quickly realized that the combination of SharePoint 2010 and AWS was the perfect marriage.

Some of the main reasons that the combination of these two technologies create a best-in-class service include the following:

- Windows servers that typically take four to six weeks to procure can be acquired and provisioned in minutes.
- By bundling a Windows Server 2008 Amazon Machine Image (AMI)[4], the host build process is completely automated.
- Amazon Elastic Compute Cloud (EC2)[5] hosts would take us out of the business of lease returns.
- Amazon Elastic Block Store (EBS)[6] would be used to scale SharePoint storage capacity within minutes independently from its compute resources.
- Amazon Virtual Private Cloud (VPC)[7] would let us keep the SharePoint installation inside our corporate network perimeter.
- We wanted to prove that cloud resources would offer cost savings when compared to our corresponding hardware total cost of ownership (TCO).
- Unhealthy SharePoint servers can be instantly replaced. The replaced servers can either be discarded or saved for in-depth troubleshooting offline, reducing the customer impact.
- The new SharePoint 2010 service application architecture would allow services to be scaled independently—making scaling SharePoint 2010 application services and server roles a great fit for deployment to the cloud.

Taking SharePoint to the cloud would allow us to realize the benefits of reduced hardware overhead, simplified operational support, improved service scalability, and infrastructure cost savings. Even with little knowledge of SharePoint 2010, we could learn as we built knowing that it is cheap to experiment in the cloud. If at any point we ran into problems, we could simply tear it down, evaluate what we learned, and start over with new design ideas.

[4] See http://aws.amazon.com/amis?_encoding=UTF8&jiveRedirect=1.
[5] See http://aws.amazon.com/ec2.
[6] See http://aws.amazon.com/ebs.
[7] See http://aws.amazon.com/vpc.

Deploying a new instance of SharePoint 2010 would enable us to validate the functionality, benefits, and supportability of both the new release and enterprise features that have been in such high demand by the business. Additionally, it would allow our users to begin realizing the benefits of SharePoint 2010 now, while pursuing a more leisurely migration from SharePoint 2007.

Launching SharePoint 2010 in the cloud quickly rose to the top of our project list after considering the preceding benefits.

Our Deployment Success Criteria

An enterprise cloud deployment requires a significant investment of engineering time and resources. Having clearly defined success criteria ensures continued momentum when challenges arise. During our system assessment phase, we identified four main criteria as keys to success and summarized why SharePoint 2010 met these success criteria. This gave us the justification to proceed with our cloud deployment plan.

1. **Strong executive commitment.** Top management must support the application as a viable candidate for migrating to the cloud. After reviewing our case, our senior executives were heavily in favor of investing in a SharePoint cloud deployment. Our organization had senior executives who were passionate about the scalability of a cloud IT infrastructure and made our cloud migration program one of our organization's top priorities. Strong executive commitment ensured that our engineers stayed the course in the face of challenges. We knew that without such support, the deployment would have been much more difficult to accomplish.

2. **Motivated Engineers.** The engineers who owned the application had to be excited about the promise of the cloud, and had to be willing to tackle challenging problems head on. Our SharePoint engineers were excited about the opportunity to blaze a trail for other teams while working with new technology. The engineers wanted to stop managing hardware infrastructure processes in general and start creating and customizing nimble enterprise applications geared toward our employees' needs. The team knew that the knowledge we gained deploying these applications in the cloud would help us to discover and resolve perceived obstacles for future cloud deployments.

3. **High cloud readiness and low migration effort.** The application had to lend itself to a cloud deployment, as was the case with both SharePoint 2010 and SQL Server 2008. The number of Working Front End (WFE) and Web Application servers was easily scaled horizontally as needed in a SharePoint farm, and application services could also be scaled independently of each other. We felt that these scalability characteristics were a great fit for the flexibility and scalability of EC2 and EBS resources. Furthermore, SharePoint 2010 provided application-level data persistence and availability solutions that would protect us in the unlikely event of an EC2 or EBS failure. As we opted to implement a stand-alone instance of SharePoint 2010, site migration could occur at a later date using one of the many third-party tools recommended by Microsoft.

4. **Strong vendor partnership on cloud licensing and support.** The vendor has to allow customers to leverage their application licenses on the AWS cloud. By leveraging Microsoft's License Mobility[8] through Software Assurance, we could deploy SharePoint 2010 and SQL Server 2008 Enterprise Edition with active Software Assurance in the cloud under our existing Microsoft Volume Licensing agreement. We are therefore able to license and operate our SharePoint and SQL Server environments on AWS the same as we do for an on-premise environment.

[8] See http://aws.amazon.com/windows/mslicensemobility.

How We Launched

Deployment Support

The deployment plan brought a number of enterprise and cloud technologies together for the first time in corporate IT. We opted to purchase AWS Premium Support so that we could maximize pre-implementation support advice (solution architects) and receive post-implementation assistance (support engineers). AWS solution architects helped us solve key challenges and share best practices from the field. AWS Premium Support was engaged as needed to help resolve issues and keep us on track. Having access to AWS Premium Support was critical to the project's success.

Initial Requirements Analysis

Working with AWS solution architects, we studied the application dependencies and requirements. We selected the following EC2 types for our SharePoint server roles:

SharePoint Role	EC2 Type
Working Front End (WFE) Servers (x2)	High Memory Quad Extra Large (m2.4xl)
Web Application Servers (x2)	High Memory Quad Extra Large (m2.4xl)
SQL Server Database (Content, Witness)	High Memory Quad Extra Large (m2.4xl)

We used Microsoft's SharePoint architecture design recommendations[9] because they were the simplest and were recommended for most large companies. The configuration works well for hosting a large number of sites for a single company on the same farm. This design would enable us to optimize the EC2 resources required to run services within a farm with all services available to all web application servers. All sites would have access to all of the service applications that are deployed in the farm. We would measure the performance of the farm, determine which services were overused, and consider whether to scale these on independent web application servers as needed.

Security Review

All applications underwent a risk assessment by our corporate information security team. The security team set a high security bar for SharePoint because they determined that the employees would store highly sensitive company data. Because the SharePoint data was so sensitive, the requirements from our security review included the following:

- SharePoint must be deployed in a Virtual Private Cloud (VPC)[10] to restrict access to only our corporate network.
- All SharePoint data must be encrypted, both at rest and in flight.
- All SharePoint traffic in the VPC must be encrypted, in addition to traffic coming in and out of the VPC.
- SharePoint encryption keys must be protected and cannot be stored in the same location as the data they are protecting.

These significant requirements are due to the sensitive nature of the data stored in our SharePoint application, and were taken into consideration as the team revisited the application design.

[9] See http://download.microsoft.com/download/2/E/5/2E54AB34-2AB4-4779-9C61-1370381FEF67/SvsSingleFarm_SharePointProducts2010.pdf.
[10] See http://aws.amazon.com/vpc.

Application Security Design

We implemented the following design elements to accommodate both application and security requirements.

Data Encryption at Rest

While a number of encryption solutions were considered, BitLocker[11] met the security requirements as a block-level encryption solution that was included with Windows at no additional cost. We used Windows Server 2008 BitLocker to encrypt all SharePoint application logs and indexes to protect confidential data on EBS volumes. The operating system volume did not need to be encrypted because temporary files that were normally stored on the operating system volume were instead stored on a separate encrypted EBS data volume.

Transparent Data Encryption (TDE) in SQL Server 2008 Enterprise was the optimal choice for bulk database encryption and it met data security standards. Because TDE acts at the application level and possesses knowledge of database file formats, it performs more optimally than BitLocker. Therefore, we decided to use TDE to encrypt SharePoint databases in SQL Server 2008 databases that contained confidential information.

The following is an overview of transparent data encryption:

- TDE encrypts databases, log files, and any information written to TempDB, snapshots, backups, and mirrored DB instances, if applicable.
- TDE operates at the input/output level through the buffer pool, so any data written into the database file (*.mdf) is encrypted.
- TDE can be selectively enabled on specific databases.
- Backups cannot be restored to other servers without a copy of the private key, so stolen MDF files do not disclose any confidential data.
- Easier administration, minimal server resources required (3 to 5 percent performance overhead).

Data Encryption in Flight

We implemented several methods for securing communication with EC2 hosts in our VPC. The following describes our security design for data in flight at the network and application level.

- All Windows hosts were configured to request IPsec connections via Group Policy Object (GPO).
- All SharePoint services running in the VPC require clients to connect over HTTPS (SSL).
- All SQL Server deployments to the VPC force all client connections to use SSL.
- Remote Desktop Protocol (RDP) connections enforce 128-bit SSL encryption.

[11] See http://technet.microsoft.com/en-us/library/cc731549(WS.10).aspx.

Windows DPAPI Encryption Key Management

One of our biggest challenges was to solve the problem of how we can securely manage our BitLocker keys. BitLocker keys needed to be managed such that Windows can auto-unlock EBS volumes on reboot. However, we did not have a Windows-based key management service that could be used to automate this process. As an alternative to implementing a custom Windows key-store service, we chose to use the Windows Data Protection API (DPAPI)[12].

DPAPI is a core service of Windows and is used by many parts of the operating system: Internet Explorer (7, 8, and 9), Encrypting File System (EFS), Outlook (S/MIME), Internet Information Services Secure Socket Layer/Transport Layer Security (SSL/TLS), Rights Management Services (RMS), 802.1x (EAP/TLS), CredMan, and almost all .NET apps requiring some form of protected storage. DPAPI uses the host machine account password, which is randomly generated when the machine account is created, to generate a key that protects the BitLocker encryption keys. The computer account password is never stored in clear text. An irreversible hash (known as a "password verifier") is stored in Credential Manager and is accessible only via Local Security Authority Subsystem (LSASS). The BitLocker keys are encrypted using DPAPI (configured to use AES 128-bit encryption) and stored in the host registry with access control lists (ACLs) applied to the hive node.

This can be done using a script in conjunction with a service and a key-store file protected by the server's machine account DPAPI key. The following summarizes the process:

1. When a data volume is encrypted with BitLocker, the script retrieves the volume's encryption key and passes it to the service (which is running as LocalService).
2. The service manages the key-store file and keeps its contents encrypted using DPAPI. (This is very similar to how BitLocker protects data volume keys when a Trusted Platform Module (TPM) is present; however, in this case, the master key is the DPAPI key rather than the BitLocker key.)
3. When a server boots, the script calls the service to retrieve the encryption key for each volume and passes that key to BitLocker's command-line utility to mount the encrypted volume.
4. Because the key-store file is encrypted with the computer's DPAPI key, the file cannot be used on any machine other than the one that initially created and encrypted it.

Proof of Concept

The proof of concept was an integral step in demonstrating the applicability of the AWS platform, our Windows infrastructure, our first SharePoint 2010 design and deployment, and our security requirements. We were able to prototype each element independently, and tear down the environment after each proof while spending very little money.

Windows Infrastructure

Implementing Windows infrastructure included the following:

1. Connecting a VPC located in the AWS US East region to our corporate network in a nearby on-premise East Coast data center.
2. Submitting an EC2 limit increase request to change the maximum on-demand instances limit per AWS Account[13] to the maximum number needed for our organization.
3. Provisioning Windows Server instances on Amazon EC2 and attaching Amazon EBS Volumes using the AWS Management Console.

---

[12] See http://msdn.microsoft.com/en-us/library/ms995355.aspx.
[13] Form to increase to your Amazon EC2 instance limit: http://aws.amazon.com/contact-us/ec2-request.

4. Bundling corporate IT standard patches, updates, and settings on the Windows Server 2008 64-bit Data Center Edition Amazon Machine Image (AMI) with patches, updates, and settings for internal use.
5. Configuring a test domain that is separate from the corporate domain using two cloud-based test domain controllers.
6. Launching Windows Server 2008 EC2 hosts using the custom AMI and joining the hosts to the test domain.

Security

Implementing the security design included the following:

1. Testing Windows BitLocker configuration on EBS volumes.
2. Writing and testing a script to retrieve the encryption key during boot-up that passes the keys to automatically unlock any BitLocker-encrypted volume on reboot using DPAPI.
3. Enabling SQL Server 2008 Enterprise and enabling TDE on databases.
4. Issuing SSL certificates and applying the certificates to Internet Information Services (IIS).
5. Enabling IPsec for all traffic between Windows hosts.

SQL Server Deployment

Implementing the SQL Server deployment included the following:

1. Deploying mirrored primary and secondary servers running SQL Server.
2. Configuring SharePoint databases.
3. Testing SQL Server failover.

SharePoint Deployment

Implementing the SharePoint deployment included the following:

1. Installing SharePoint using the default configuration wizard.
2. Learning how to configure SharePoint 2010 to corporate IT specification.
3. Executing a SharePoint functionality test plan.
4. Executing a SharePoint WFE and application server failover test plan.

Production Deployment

The knowledge gained in the proof of concept enabled us to develop a solid deployment plan, which we executed as follows.

EC2 Capacity

As we moved into production, we were in a position to pay for reserved EC2 instances[14]. This ensured that we got more favorable reserved pricing for "always on" SharePoint capacity. This pricing option for EC2 enabled us to make a low one-time payment to further reduce hourly usage charges. Reserved Instances complement our existing EC2 on-demand instances to reduce our EC2 costs.

[14] To learn more about Reserved Instances, go to http://aws.amazon.com/ec2/reserved-instances.

Windows Infrastructure

First, we extended the corporate active directory and domain to the VPC. This enabled us to leverage our existing Windows identity and access model into AWS.

Windows IPsec

A GPO was created to enable IPsec between Windows hosts in the VPC.

Windows Encryption and Key Management

Windows BitLocker was used to encrypt all SharePoint volumes containing potentially confidential data. SQL Server TDE was used to encrypt the SharePoint databases. As an alternative to implementing a custom Windows key-store service, we used DPAPI, custom scripts, and a service to manage the BitLocker keys that encrypted log, index, and temporary file volumes.

SharePoint Availability

We chose to implement high availability using SharePoint instances on-premise in a nearby corporate IT data center. This ensured that our deployment had a failover solution off the cloud for each of our SharePoint Working Front End servers, Web Application servers, and SQL Server instances. A primary and secondary SQL Server pair were installed with database mirroring. The primary server was installed in the VPC, with the secondary deployed to a nearby on-premise East Coast data center. A separate witness server was deployed to the VPC running SQL Server Express to monitor the databases and handle failovers.

SharePoint Data Redundancy

SharePoint WFE and application servers run stateless services that do not require any data persistence. Should any of the corresponding EC2 hosts become corrupted or lost, they would simply be replaced by a freshly configured SharePoint EC2 host and EBS volume. The only data that must be persisted is stored in SQL Server databases and these databases are automatically mirrored to a secondary on-premise server to ensure that data is persisted off the cloud on every transaction.

SharePoint Backups and Recovery

The only backups that need to take place are at the database level. We backup the database every eight hours to mitigate major disasters that might cause the loss of significant amounts of data. Should a disaster occur, a full database restore would be implemented to recover the farm to the last database snapshot.

SharePoint and SQL Server Monitoring

Our SharePoint and SQL Server cloud and on-premise deployments both used Microsoft System Center Operations Manager (SCOM) as the monitoring solution. SharePoint and SQL Server management packs were installed to help detect and respond to critical events generated by these specific applications.

Architecture Diagram

The following diagram outlines how we deployed our SharePoint 2010 architecture across both our corporate AWS VPC and a nearby on-premise east coast data center. When we designed this solution, AWS VPC was available only in a single Availability Zone in the AWS Region where we were deploying (AWS has since made VPC available in multiple AZ's per Region[15]). Deploying each SharePoint component across an AWS Availability Zone in our VPC and our own east coast data center ensured high availability across two data centers. On-premise load balancers direct traffic appropriately across the four SharePoint front-end servers that are connected to four application servers to balance the processing of the application services load. The primary SQL Server database handles all data, with SQL Server mirroring set up to handle data redundancy to a secondary on-premise server.

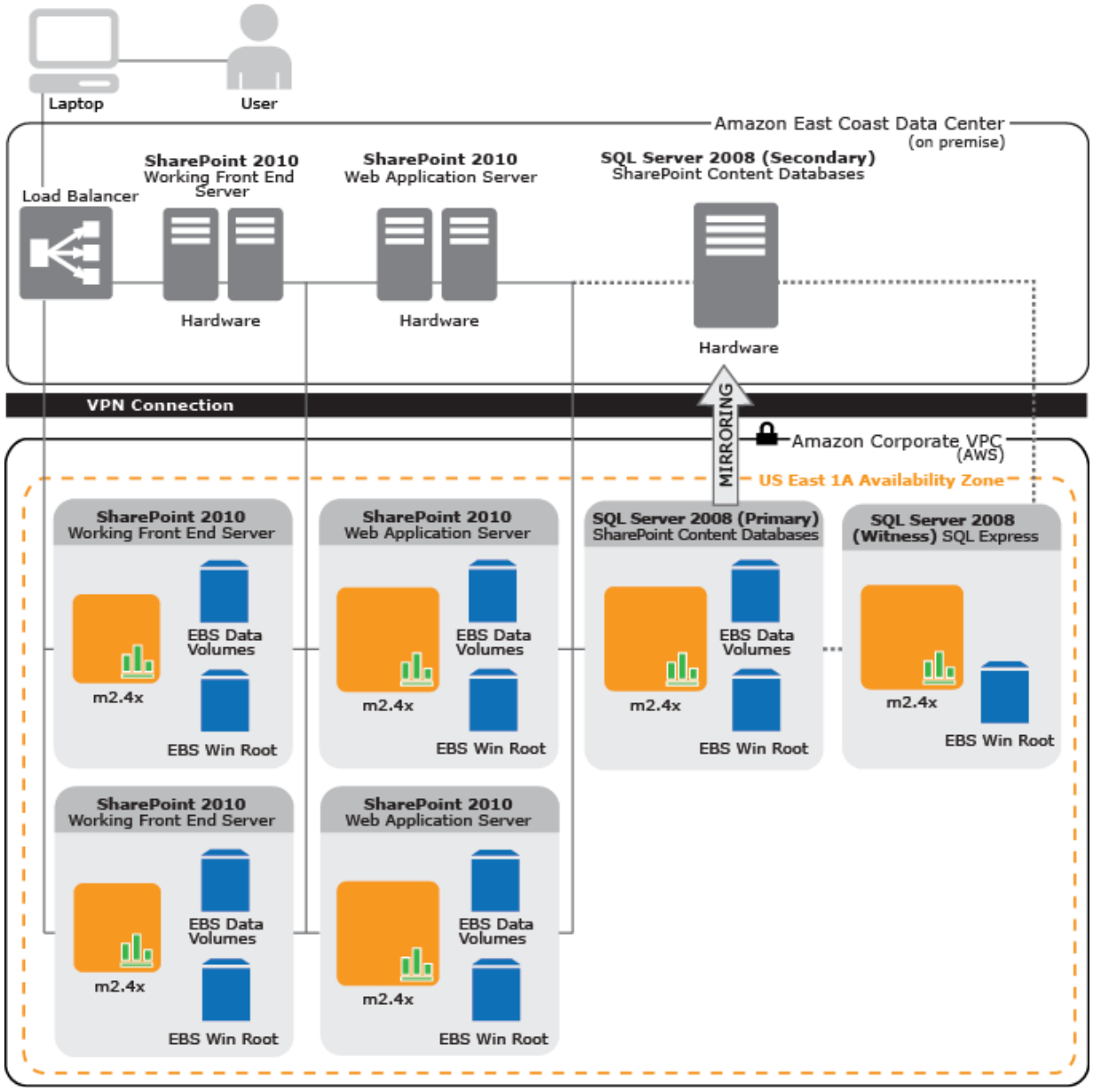


Figure 1. SharePoint 2010 Enterprise Component Breakdown

[15] http://aws.amazon.com/about-aws/whats-new/2011/08/03/Announcing-VPC-GA.

Lessons We Learned

As with any major enterprise migration, we encountered a few challenges during the deployment. We will apply the knowledge gained from these lessons to drive future cloud migration projects and share them with other teams.

Licensing

Many vendor licensing agreements are not yet written anticipating the use of a cloud-based infrastructure. This will change, but in the meantime, be sure to engage your vendor early in the project, as we did with our Microsoft account representatives, to understand how their licensing and support agreements can be applied to AWS. We learned that it is important to educate the vendor about the cloud architectural patterns and to help them to develop new licensing models to support the cloud if their existing licensing models don't apply. By investing in this process early, all parties were able to agree on the conditions that would allow corporate IT to launch both SharePoint Server 2010 and SQL Server 2008 Enterprise Edition for production use in our VPC.

Forecast VPC Resources

The goal for our corporate IT was to use a single AWS account for all EC2 capacity in our VPC. As a large enterprise customer, we went through a forecasting process with AWS to fulfill our VPC capacity requirements. This process began with a request to increase the EC2 limit for our AWS account[16]. We then worked with AWS sales and solution architects to determine how many reserved versus on-demand instances we would need over three years.

Use Standard Scalability and High Availability Practices

Corporate IT best practices state that we should deploy applications to at least two data centers for high availability. Therefore, we deployed SharePoint and SQL Server in the VPC and in an on-premise nearby data center. We also leveraged our existing on-premise load balancers to direct traffic across SharePoint web front-end instances on-premise and in the VPC. In the future, we want to expand our cloud footprint to a second VPC Availability Zone and use Amazon Elastic Load Balancing to direct traffic. This allows us to consider the option of decommissioning on-premise servers and load balancers.

Plan an Access Control Strategy for Cloud Resources

With the power and flexibility of having on-demand hosts, comes the power of terminating hosts just as easily. We needed to think through the processes and controls needed to protect these resources from accidental deletion or misidentification. Our approach relied on the use of AWS Identity and Access Management (IAM)[17] policies and tagging for resource identification. We assigned AWS user policies to engineers across teams where limiting access to various AWS APIs helped us to avoid accidental deletions. Defining a tagging convention also helped us to easily identify EC2 and Amazon EBS resources. We gave feedback to the AWS team about how resource-level controls could simplify and improve our IAM strategy.

[16] Form to increase to your AWS EC2 instance limit: http://aws.amazon.com/contact-us/ec2-request.
[17] See http://aws.amazon.com/documentation/iam.

Security Is a Shared Responsibility

Our Corporate Information Security team held AWS to the same standards as any external vendor. We learned that security is a shared responsibility. Although AWS offers a variety of different security features like Security Groups that help us protect instances, data volumes, and network traffic, it is our responsibility to protect the corporate data that is stored and running in the cloud. Because of the sensitive nature of SharePoint data, Corporate Information Security required us to encrypt all data at rest and in flight. From these requirements, the team learned about the architectural challenges and operational overhead that comes with supporting a robust data encryption and key management solution. We overcame these challenges by investing the time and resources needed to incorporate these critical security components into our cloud deployment. These are now core framework components that we will use in other similar Windows cloud deployment projects.

Start Conservatively and Scale Over Time

Through early experimentation, we discovered that Windows EBS volume performance varies over time. As an application, SharePoint has very specific storage performance requirements, which equate to about 0.75 IOPS/GB stored. Based on our limited experience with SharePoint 2010 requirements and the impact that storage performance variability will have on the SharePoint farm, we decided to start conservatively by initially launching only 100 production team sites. Over time, we will monitor performance of these initial SharePoint sites and architect the best cloud-based deployment model, which will likely require us to scale our cloud resources horizontally. Given the relatively high input/output requirements for SharePoint storage on SQL Server, we have requested that AWS consider providing more consistent, high-performance storage offerings that require less direct managing and mapping of Windows EBS volumes.

What Benefits Emerged

The deployment of SharePoint 2010 self-service portals and a back-end SQL Server 2008 Enterprise database farm to the AWS cloud was completed quickly and efficiently. The deployment enabled us to deliver the SharePoint 2010 Enterprise application's overall business value to our employees.

Specific benefits include the following:

- Infrastructure procurement time was reduced from over four to six weeks to minutes.
- Server image build process that had previously taken a half day is now automated.
- Annual infrastructure costs were cut by 22 percent when on-premise hardware was replaced with equivalent cloud resources.
- Operational overhead of server lease returns were eliminated, freeing up approximately 2 weeks of engineering overhead per year by replacing servers with equivalent cloud resources.

Where We Go From Here

With SharePoint now running on AWS in a VPC and key lessons learned in the process, we are sharing our story with internal peers and external customers. We solved a number of key availability, performance, scalability, and security concerns that can be leveraged within the company and to drive other internal teams to adopt AWS as their cloud provider. This will build visibility and separate the *truths* from *myths* that are prevalent among many enterprise IT organizations.

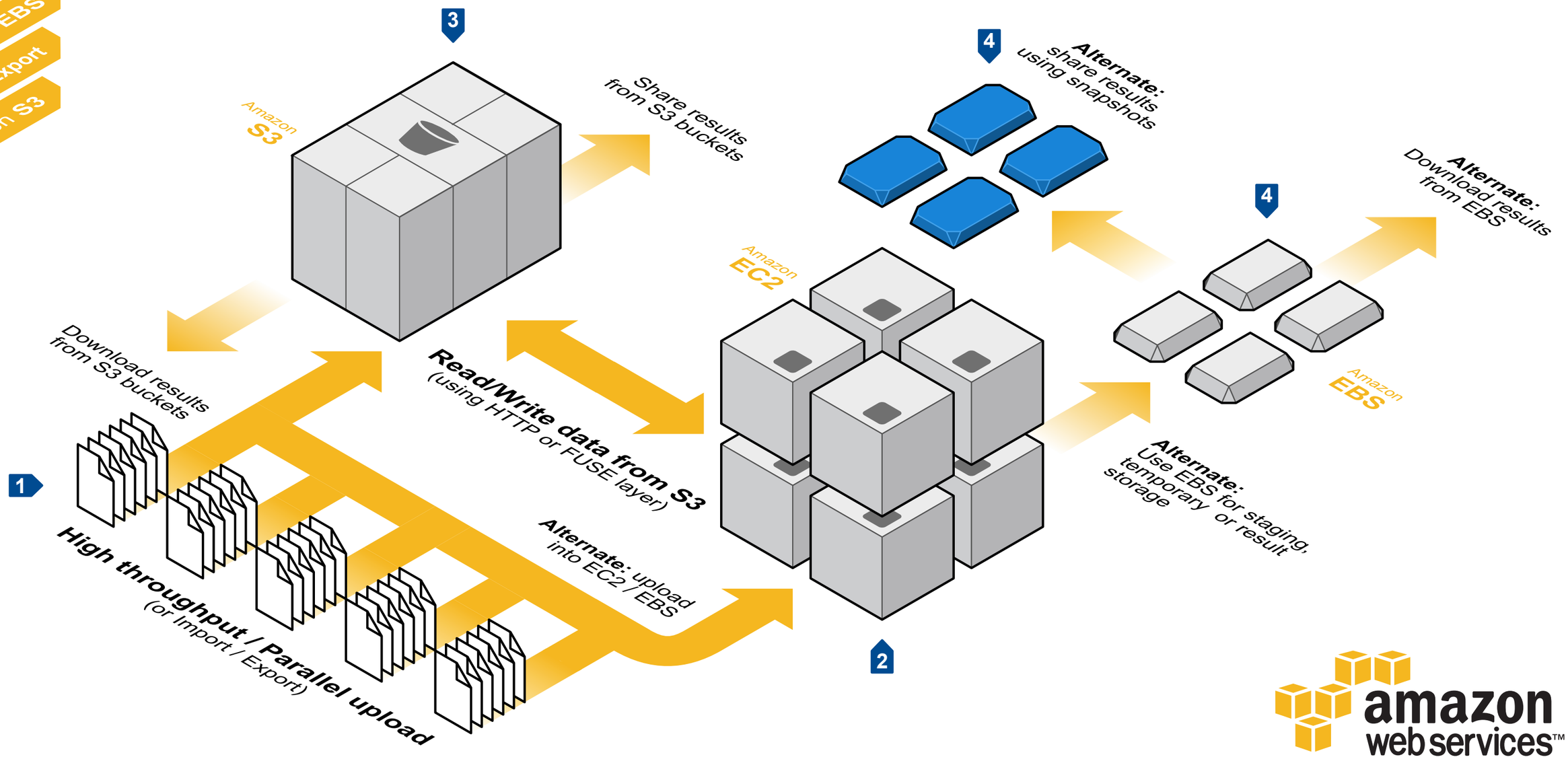
We are now in a position to make more cloud optimizations. We plan to optimize for the cloud by exploring ways we can further reduce operational overhead by automating SharePoint and SQL Server deployments. We are exploring how to expand our cloud footprint into other geographies to meet stakeholder demand for a SharePoint presence in Europe and Asia. In the end, our goal is to decommission all on-premise SharePoint hardware in favor of a fully redundant and highly available presence on the AWS cloud.

AWS Reference Architectures

- Amazon EC2
- Amazon EBS
- AWS Import / Export
- Amazon S3

LARGE SCALE COMPUTING & HUGE DATA SETS

Amazon Web Services is very popular for large-scale computing scenarios such as scientific computing, simulation, and research projects. These scenarios involve huge data sets collected from scientific equipment, measurement devices, or other compute jobs. After collection, these data sets need to be analyzed by large-scale compute jobs to generate result data sets. Ideally, results will be available as soon as the data is collected. Often, these results are then made available to a larger audience.



amazon web services™

System Overview

1 To upload large data sets into AWS, it is critical to make the most of the available bandwidth. You can do so by uploading data into **Amazon Simple Storage Service (S3)** in parallel from multiple clients, each using multithreading to enable concurrent uploads or multipart uploads for further parallelization. TCP settings like window scaling and selective acknowledgement can be adjusted to further enhance throughput. With the proper optimizations, uploads of several terabytes a day are possible. Another alternative for huge data sets might be **Amazon Import/Export**, which supports sending storage devices to AWS and inserting their contents directly into **Amazon S3** or **Amazon EBS** volumes.

2 Parallel processing of large-scale jobs is critical, and existing parallel applications can typically be run on multiple **Amazon Elastic Compute Cloud (EC2)** instances. A parallel application may sometimes assume large scratch areas that all nodes can efficiently read and write from. S3 can be used as such a scratch area, either directly using HTTP or using a FUSE layer (for example, s3fs or SubCloud) if the application expects a POSIX-style file system.

3 Once the job has completed and the result data is stored in **Amazon S3**, **Amazon EC2** instances can be shut down, and the result data set can be downloaded The output data can be shared with others, either by granting read permissions to select users or to everyone or by using time limited URLs.

4 Instead of using **Amazon S3**, you can use **Amazon EBS** to stage the input set, act as a temporary storage area, and/or capture the output set. During the upload, the concepts of parallel upload streams and TCP tweaking also apply. In addition, uploads that use UDP may increase speed further. The result data set can be written into EBS volumes, at which time snapshots of the volumes can be taken for sharing.