



**Architecting on AWS
Student Guide
Version 3.1**

100-ARC-31-EN-SG

Copyright © 2013, 2014 Amazon Web Services, Inc. and its affiliates.
All rights reserved.

This work may not be reproduced or redistributed, in whole or in part,
without prior written permission from Amazon Web Services, Inc.

Commercial copying, lending, or selling is prohibited.

Corrections or feedback on the course, please email us at:
aws-course-feedback@amazon.com.

For all other questions, please email us at aws-training-info@amazon.com.

The Architecting on AWS course will enable you to design scalable, elastic, secure, and highly available applications on AWS. In this course, we cover architecture patterns for common solutions running on AWS, including Web Applications, Batch Processing, and hosting internal IT Applications. You'll have an opportunity to explore AWS products and services through the use of hands-on labs, and build your knowledge of how to connect AWS components through whiteboard and group discussion sessions.

Table of Contents

Knowledge Check	1
Architecting in the Cloud	23
Security and Compliance	44
Amazon VPC	90
Identity, Authentication, and Authorization	120
Overview of Services for Web Applications	179
Elasticity, Scalability, and Bootstrapping	227
Data Storage Scaling	317
Overview of Application Services	371
Designing for Cost	400
Disaster Recovery and High Availability	447
Migrating Applications to the AWS Cloud	494
Reference Architecture	540



Module 1: Knowledge Check

Architecting on AWS – Knowledge Check

 Amazon
Training and
Certification 1

Notes:

In the next few slides, we'll go over a few of the basic concepts related to Amazon Web Services.

Knowledge Check

What is the Cloud?

Notes:

Cloud Computing is the use of computing resources (hardware and software) that are delivered as a service over a network (typically the Internet). The name comes from the use of a cloud- shaped symbol as an abstraction for the complex infrastructure it contains in system diagrams. Cloud computing entrusts remote services with a user's data, software and computation.

From *Module 1: Introduction to AWS, page 10, AWS Essential Student Guide, version 3.1.*

More information available at: <http://aws.amazon.com/what-is-cloud-computing/>.

Knowledge Check

How does AWS cloud computing differ from traditional IT infrastructure?

Notes:

AWS customers can leverage the cost effectiveness, scalability and flexibility of running on AWS infrastructure. AWS provides a massive global cloud infrastructure that allows companies to quickly innovate, experiment and iterate.

Instead of waiting weeks or months for hardware, you can instantly deploy new applications, scale up as the workload grows, and scale down based on demand. With AWS there is zero capital expenditure and no long-term contract.

From *Module 1: Introduction to AWS, page 15, AWS Essential Student Guide, version 3.1.*

More information available at: <http://aws.amazon.com/what-is-cloud-computing/>.

Knowledge Check

There are 3 main methods of interacting with AWS. Can you list them? Which method is most commonly used by individuals new to AWS?

Notes:

These methods are:

- AWS Management Console
- AWS Command Line Interface (CLI)
- Software Development Kits (SDKs)

You can also use query APIs, but this method is not as common as the other three.

More information available at:

<http://docs.aws.amazon.com/gettingstarted/latest/awsgsg-intro/gsg-aws-what-tools.html> .

Knowledge Check

What is an AWS Region?

Notes:

A region is a collection of AWS resources located in a specific geographic area. Each region is completely independent.

To see the benefits of regions, consider Amazon EC2. Amazon EC2 provides multiple regions so that you can launch Amazon EC2 instances in locations that meet your requirements. For example, you might want to launch instances in Europe to be closer to your European customers or to meet legal requirements.

More information available at:

<http://docs.aws.amazon.com/AWSEC2/latest/UserGuide/using-regions-availability-zones.html>.

Knowledge Check

What are regions made of?

Notes:

Each region has multiple, isolated locations known as Availability Zones. Availability Zones in a region are connected through low-latency links. You can design your application to use multiple availability zones. This way, if resources in one Availability Zone fails, the resources in another Availability Zone can handle requests.

More information available at:

<http://docs.aws.amazon.com/AWSEC2/latest/UserGuide/using-regions-availability-zones.html> .

Knowledge Check

Where can you get the latest information on the number of regions and Availability Zones within AWS?

Notes:

An up-to-date list of AWS regions and Availability Zones can be found here:
<http://aws.amazon.com/about-aws/globalinfrastructure/>.

Knowledge Check

Where does AWS responsibility for security end and yours begin?

Notes:

Because you're building systems on top of the AWS cloud infrastructure, the security responsibilities will be shared: AWS has secured the underlying infrastructure and you must secure anything you put on the infrastructure. This includes your AWS EC2 instances and anything you install on them, any accounts that access your instances, the security group that allows outside access to your instances, the VPC subnet that the instances reside within if you've chosen this option, the external access to your S3 buckets, etc.

More information available at: <http://aws.amazon.com/security/>.

Knowledge Check

Why would you use a security group? Can you give an example of how multiple security groups would work together?

Notes:

A *security group* acts as a virtual firewall that controls the traffic for one or more instances. When you launch an instance, you associate one or more security groups with the instance. You add rules to each security group that allow traffic to or from its associated instances. You can modify the rules for a security group at any time; the new rules are automatically applied to all instances that are associated with the security group. When we decide whether to allow traffic to reach an instance, we evaluate all the rules from all the security groups that are associated with the instance.

More information available at:

<http://docs.aws.amazon.com/AWSEC2/latest/UserGuide/using-network-security.html>.

Knowledge Check

How do you extend your on-premise network into AWS?

Notes:

Amazon Virtual Private Cloud (Amazon VPC) lets you provision a logically isolated section of the Amazon Web Services (AWS) Cloud where you can launch AWS resources in a virtual network that you define. You have complete control over your virtual networking environment, including selection of your own IP address range, creation of subnets, and configuration of route tables and network gateways.

More information available at: <http://aws.amazon.com/vpc/>.

Knowledge Check

What type of storage system is S3?

Notes:

S3 is object storage (as opposed to a traditional file system). The fundamental components of Amazon S3 are objects, which are in turn stored in buckets.

More information available at:

<http://docs.aws.amazon.com/AmazonS3/latest/dev/Introduction.html>

Knowledge Check

How do different EC2 instance types differ?

Notes:

Instance types comprise varying combinations of CPU, memory, storage, and networking capacity and give you the flexibility to choose the appropriate mix of resources for your applications. Each instance type includes one or more instance sizes, allowing you to scale your resources to the requirements of your target workload. For the best experience, you should launch on instance types that are appropriate for your applications. You have the flexibility to choose the combination of instance types most appropriate for your application today, and then choose a different instance size as your business and application needs change. To make it easier for you to select the best option for your applications, Amazon EC2 instance types are grouped together into families based on target application profiles. Below is a quick guide to help you think about choosing the right instance type.

More information at: <http://aws.amazon.com/ec2/instance-types/>.

Knowledge Check

There are three ways you can pay for EC2 instances. Can you name them?

Notes:

Amazon EC2 provides customers three different purchasing models that give you the flexibility to optimize your costs. On-Demand Instances allow you to pay a fixed rate by the hour with no commitment; with Reserved Instances you pay a low, one-time fee and in turn receive a significant discount on the hourly charge for that instance; and Spot Instances enable you to bid whatever price you want for instance capacity, providing for even greater savings if your applications have flexible start and end times.

More information at: <http://aws.amazon.com/ec2/purchasing-options/>.

Knowledge Check

How many providers of Amazon Machine Images (AMIs) are there? Can you name them?

Notes:

There are three providers of AMIs:

- Amazon Web Services
- Third parties, such as community AMIs or AMIs purchased from the AWS Marketplace (https://aws.amazon.com/marketplace?b_k=291)
- You! (The AMIs you create are considered custom AMIs)

More information at:

<http://docs.aws.amazon.com/AWSEC2/latest/UserGuide/AMIs.html>

Knowledge Check

Why would you choose EBS over instance storage?

Notes:

Amazon Elastic Block Store (Amazon EBS) provides persistent block level storage volumes for use with Amazon EC2 instances in the AWS Cloud. Each Amazon EBS volume is automatically replicated within its Availability Zone to protect you from component failure, offering high availability and durability. Amazon EBS volumes offer the consistent and low-latency performance needed to run your workloads. With Amazon EBS, you can scale your usage up or down within minutes – all while paying a low price for only what you provision.

More information at: <http://aws.amazon.com/ebs/> and
<http://docs.aws.amazon.com/AWSEC2/latest/UserGuide/RootDeviceStorage.html>

Knowledge Check

Why would you want to scale out your application, instead of scaling up?

Notes:

In a typical business situation, when your web application starts to get more traffic, you either add more servers or increase the size of your existing servers to handle the additional load. Similarly, if the traffic to your web application starts to slow down, you either terminate the under-utilized servers or decrease the size of your existing servers. Depending on your infrastructure, vertical scaling might involve changes to your server configurations every time you scale. With horizontal scaling, you simply increase or decrease the number of servers according to your application's demands. The decision when to scale vertically and when to scale horizontally depends on factors such as your use case, cost, performance, and infrastructure.

When you scale using Auto Scaling you can increase the number of servers you're using automatically when the user demand goes up to ensure that performance is maintained, and you can decrease the number of servers when demand goes down to minimize costs. Auto Scaling helps you make efficient use of your compute resources by automatically doing the work of scaling for you. This automatic scaling is the core value of the Auto Scaling service.

More information at:

<http://docs.aws.amazon.com/AutoScaling/latest/DeveloperGuide/WhatIsAutoScaling.html>

Knowledge Check

Why should you avoid using your AWS Master Account?

Notes:

The access key ID and secret access key for your AWS account give you access to all your resources, including your billing information. Also, you cannot restrict permissions on a master account.

Therefore, protect your AWS account credentials like you would your credit card numbers or any other secret.

More information at:

<http://docs.aws.amazon.com/IAM/latest/UserGuide/IAMBestPractices.html>

Knowledge Check

What is the difference between a CloudFormation template and a CloudFormation stack?

Notes:

A stack is a collection of AWS resources. A template is a text file whose format complies with the JSON standard that describes your AWS infrastructure requirements.

More information at:

<http://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/concept-stack.html> and

<http://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/concept-template.html>

Knowledge Check

What advantages does a CloudFormation template have over a traditional network diagram?

Notes:

AWS CloudFormation enables you to create and provision AWS infrastructure deployments predictably and repeatedly. It helps you leverage AWS products such as Amazon Elastic Compute Cloud, Amazon Elastic Block Store, Amazon Simple Notification Service, Elastic Load Balancing and Auto Scaling to build highly reliable, highly scalable, cost-effective applications without worrying about creating and configuring the underlying AWS infrastructure.

More information at:

[© 2013, 2014 Amazon Web Services, Inc. and its affiliates. All rights reserved.](http://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide>Welcome.html</p></div><div data-bbox=)

Knowledge Check

Where you can you go to find a variety of CloudFormation templates that you can try out?

Notes:

A variety of templates are available at

<http://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/example-templates.html> and <http://aws.amazon.com/cloudformation/aws-cloudformation-templates/>.

Knowledge Check

How can you keep track of your applications metrics in AWS? What types of metrics are available to you?

Notes:

You can use CloudWatch to collect and track metrics, which are the variables you want to measure for your resources and applications. CloudWatch alarms send notifications or automatically make changes to the resources you are monitoring based on rules that you define. For example, you can monitor the CPU usage and disk reads and writes of your Amazon Elastic Compute Cloud (Amazon EC2) instances and then use this data to determine whether you should launch additional instances to handle increased load. You can also use this data to stop under-used instances to save money. In addition to monitoring the built-in metrics that come with AWS, you can monitor your own custom metrics. With CloudWatch, you gain system-wide visibility into resource utilization, application performance, and operational health.

CloudWatch metrics focus on metrics available at the hypervisor level. You can view what metrics are available to you at

http://docs.aws.amazon.com/AmazonCloudWatch/latest/DeveloperGuide/viewing_metrics_with_cloudwatch.html.

More information available at:

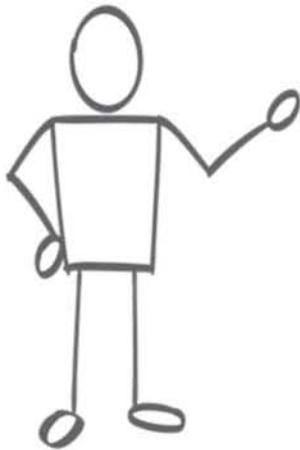
<http://docs.aws.amazon.com/AmazonCloudWatch/latest/DeveloperGuide/WhatIsCloudWatch.html>.

Copyright © 2013, 2014 Amazon Web Services, Inc. and its affiliates. All rights reserved.

This work may not be reproduced or redistributed, in whole or in part, without prior written permission from Amazon Web Services, Inc. Commercial copying, lending, or selling is prohibited.

Errors or corrections? Email us at aws-course-feedback@amazon.com.
Other questions? Email us at aws-training-info@amazon.com.

All trademarks are the property of their owners.



Module 2: Architecting in the Cloud

Topics

- 💡 Five benefits of the cloud
- 💡 Seven best practices for building systems with AWS

[Course Title - Module Title]



Notes:

Topics

- 💡 Five benefits of the cloud
- 💡 Seven best practices for building systems with AWS

[Course Title - Module Title]



Notes:

What makes the cloud attractive?

💡 Abstract Resources

- Focus on your needs, not hardware specs. As needs change, so should your resources

💡 On-Demand Provisioning

- Ask for what you need, exactly when you need it; get rid of it when you do not

💡 Scalability in Minutes

- Scale out or in, up or down, depending on usage or needs

Notes:

Abstract Resources: Abstract resources means that you can change components

without needing to re-design your system. On-Demand Provisioning: by

provisioning resources only when you need them, you can run your applications

more efficiently. Scalability in Minutes: Scalability is critical: the cloud allows you to

scale up AND down, depending on what you need.

What makes the cloud attractive? (continue)

💡 Pay Per Consumption

- No long-term commitments. Pay only for what you use

💡 Efficiency of Experts

- Utilize the skills, knowledge and resources of experts

Notes:

Another benefit of the cloud: you pay for what you use. This makes it easy to test changes out, scale systems up and down, and keep costs under control. You and your team can spend more time working on the hard problems.

Topics

- 💡 Five benefits of the cloud
- 💡 Seven best practices for building systems with AWS

[Course Title - Module Title]



Notes:

Seven best practices for building systems with AWS

- 💡 Design for failure and nothing fails
- 💡 Loose coupling sets you free
- 💡 Implement elasticity
- 💡 Build security in every layer
- 💡 Don't fear constraints
- 💡 Think parallel
- 💡 Leverage different storage options

Notes:

Let's list through the seven best practices, and then we'll go into each one in more detail.

Seven best practices for building systems with AWS

“Everything fails, all the time.”

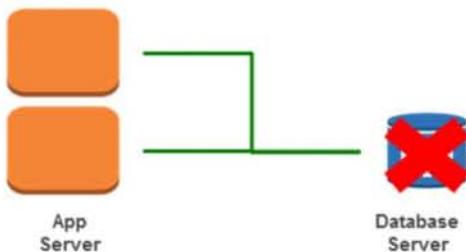
Werner Vogels, CTO, Amazon.com

Notes:

This quote from our CTO Werner Vogels, is a great mindset when designing any system—whether on-premises or on AWS. If you design your architecture around that premise – specifically, assuming that any component will eventually fail – then your application won't fail when an individual component does. For example, one goal when designing for failure would be to see your application survive when the underlying physical hardware fails on one of your servers.

Seven best practices: Design for failure

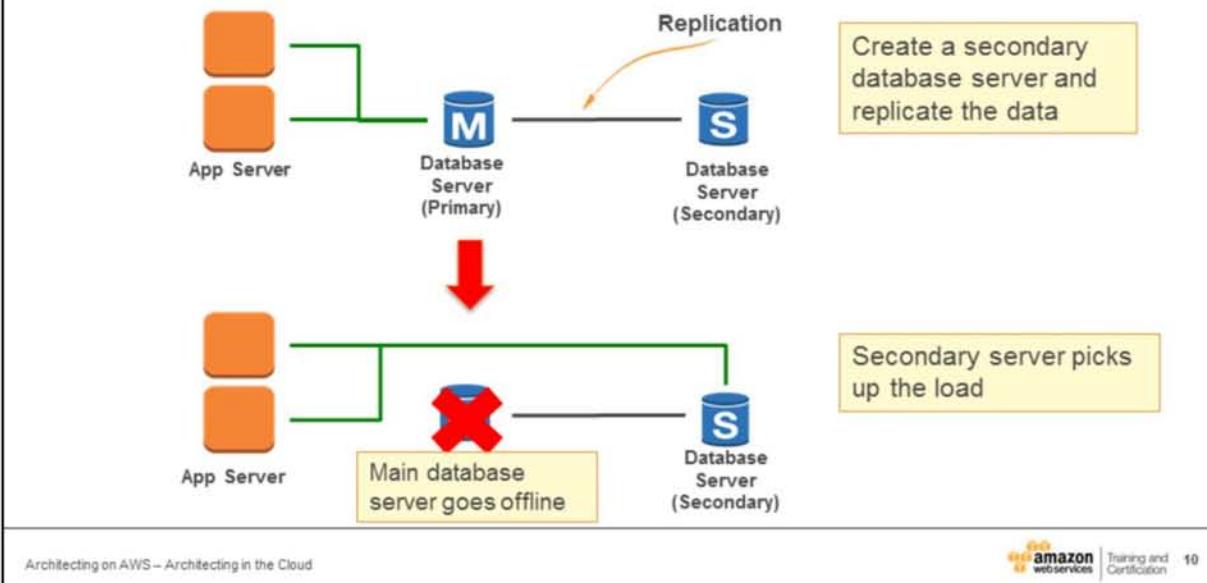
- 💡 Avoid single points of failure
- 💡 Assume everything fails and design backwards
 - **Goal:** Applications should continue to function even if the underlying physical hardware fails or is removed/replaced.



Notes:

This (simple) system shows two application servers connected to a single database server. The database server represents a single point of failure. When it goes down, so does the app. The single point of failure is exactly what you want to avoid. Reiterate the goal shown in the slide: apps should continue to function even if the underlying physical hardware fails or is removed/replaced.

Seven best practices: Design for failure (continue)



Notes:

A common way to solve the problem of a single database server is to create a secondary server and replicate the data.

If the main database server goes offline, the secondary server can pick up the load. Notice that, when the main database goes offline, the app servers need to automatically send their requests to the secondary database.

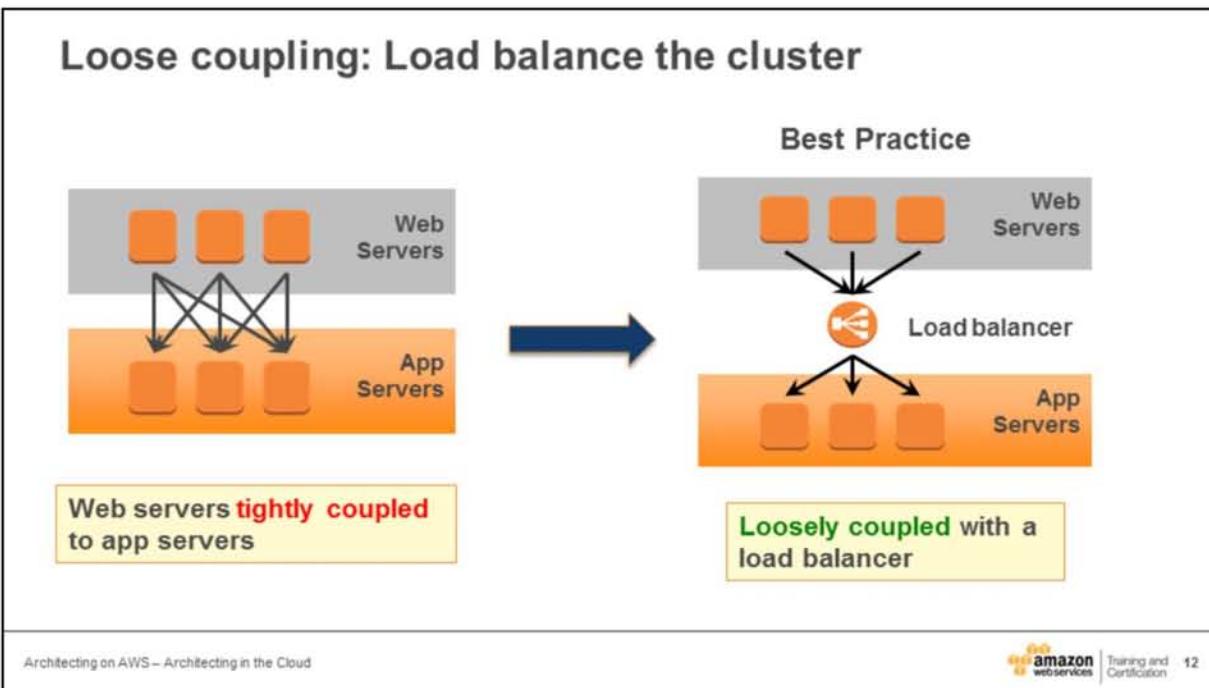
Seven best practices: Loose coupling

- 💡 Design architectures with independent components
 - The more loosely they are coupled, the bigger they scale
- 💡 Design every component as a black box
- 💡 Load balance clusters
- 💡 Use a queue to pass messages between components

Notes:

A loosely coupled system means each component of that system operates independently. This independence can help you create systems that scale.

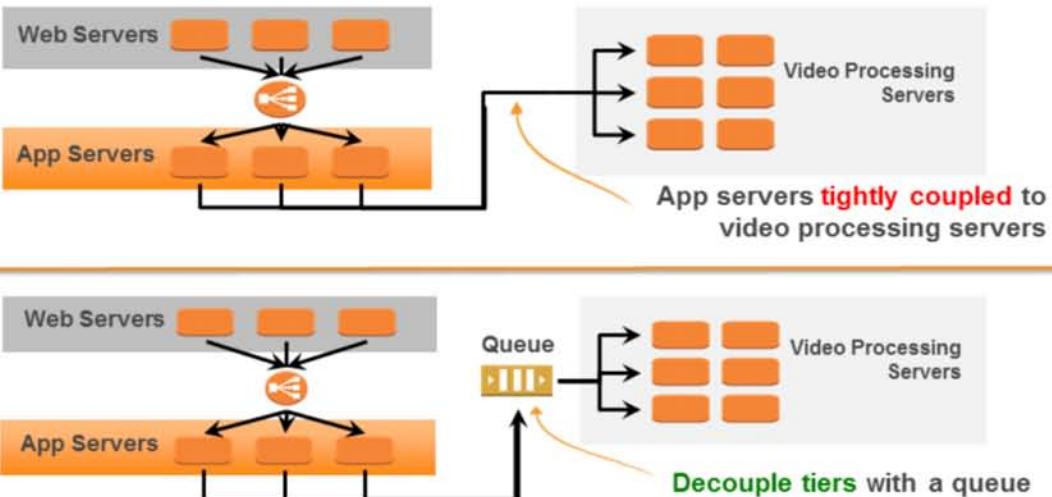
Components in loosely-coupled systems operate without knowing the details of other components—each component is basically a “black box.” A common way to create a loosely-coupled system is to load balance requests across multiple systems. Another strategy uses queues to pass messages from one component to another.



Notes:

On the left, the diagram illustrates a collection of web and app servers that are tightly-coupled. On the right, the diagram shows a load balancer (in this case, Amazon Elastic Load Balancer) that routes requests between the web servers and the app servers.

Loose coupling: Use a queue to pass messages



[Course Title - Module Title]

amazon
aws | Training and Certification 13

Notes:

On the top, we have app servers that are tightly coupled to backend video processing servers. In order to achieve loose coupling, use a queue in the middle to manage requests from the app servers to the video processing servers.

Seven best practices: Implement elasticity

- 💡 Elasticity is a fundamental property of the cloud
- 💡 Do not assume the health, availability, or fixed location of components
- 💡 Use designs that are resilient to reboot and re-launch
- 💡 Bootstrap your instances
 - When an instance launches, it should ask “Who am I and what is my role?”
- 💡 Favor dynamic configuration

Notes:

Elasticity is a fundamental property of the cloud. When you design with elasticity in mind, the components of your system don't assume that other components are available. In addition, elastic systems remain functional as components are rebooted, re-launched, or replaced.

A key requirement of elasticity is bootstrapping. With bootstrapping, new components are told who they are and what role they play as they come online. Elasticity also gives you the ability to dynamically configure components to adapt to changing circumstances.

Seven best practices: Build security in every layer

💡 Security is a shared responsibility. You decide how to:

- Encrypt data in transit and at rest
- Enforce principle of least privilege
- Create distinct, restricted Security Groups for each application role
 - Restrict external access via these security groups
- Use multi-factor authentication

Notes:

Security is always important—whether your app is hosted on-premises, in the cloud, or a combination of the two.

Even if data is in the cloud, it's still your responsibility to apply the appropriate levels of encryption when the data is in transit and when it's at rest. The principle of least privilege means that people and systems are only given privileges that are essential to that person or system's role. (You can learn more here: http://en.wikipedia.org/wiki/Principle_of_least_privilege.) In AWS, create security groups and use them to restrict external access to your system.

Multi-factor authentication is always highly recommended.

Seven best practices: Do not fear constraints

💡 Need more RAM?

- Consider distributing load across machines or a shared cache

💡 Need better IOPS for database?

- Instead, consider multiple read replicas, sharding, or DB clustering

💡 Hardware failed or configuration got corrupted?

- “Rip and replace”—Simply toss bad instances and instantiate replacement

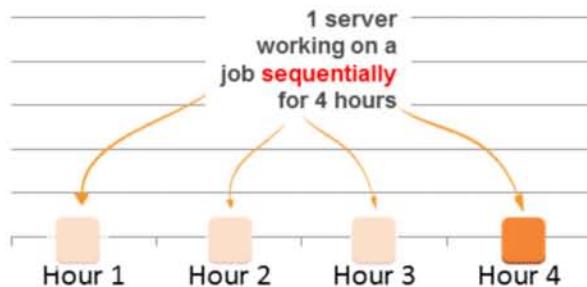
Notes:

A great benefit of the cloud is that you're free to adapt your systems as necessary. Changes that are difficult or potentially more costly when done on-premises (such as adding more RAM) are easy to do in the cloud. You can replace and remove bad instances with new ones.

The cloud makes it easier to create multiple replicas of databases. (We also have services, such as RDS and DynamoDB, that can help.)

Seven best practices: Think parallel

- Experiment with parallel architectures

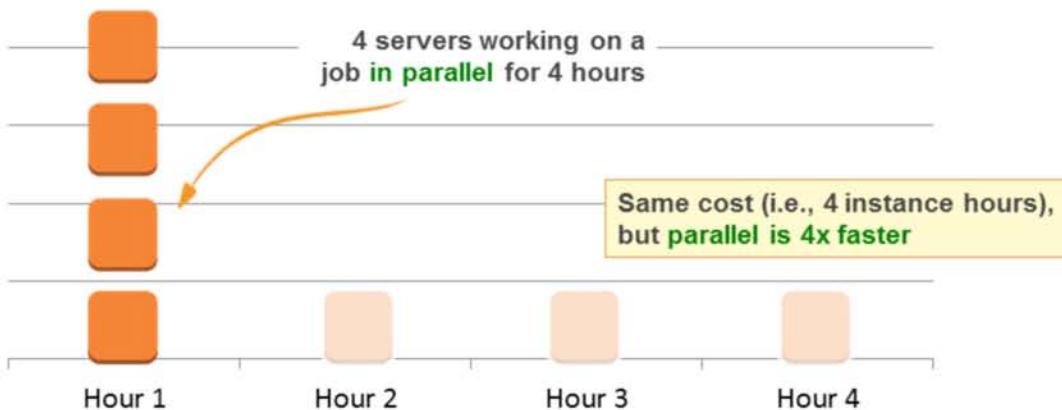


Notes:

Often, we think of scaling a system vertically: we increase RAM, hard drive space, and so on. In the cloud, you could also scale horizontally: instead of increasing the capability of a single system, you can distribute the workload across multiple systems. These types of systems are parallel architectures.

Here, we have a single server working on a job sequentially for 4 hours. As time passes, the same server continues to work until finally the job is finished.

Seven best practices: Think parallel (continue)



Notes:

Instead of having one server work for four hours, what about having 4 servers work for one hour?

In this example, both sequentially and parallel architectures cost the same (4 instance hours), but with a parallel system, the process is done 4x faster.

What if we are talking about one month instead of one hour?

(1 instance in 750 hours) = (1 hour for 750 instances)

Seven best practices: Leverage many storage options

💡 One size does not fit all

- Object storage
- Content delivery network/edge caching
- Block storage
- Relational database
- NoSQL

Notes:

As you can see in this slide, there are lots of storage options available, because different systems have different (and often multiple) storage needs.

Module review

- 💡 What are the benefits cloud services offer?
- 💡 List the seven AWS best practices

Answers:

Five benefits:

1. Abstract resources
2. On-demand provisioning
3. Scalability in minutes
4. Pay per consumption
5. Efficiency of Experts

Seven best practices.

1. Design for failure and nothing fails
2. Loose coupling sets you free
3. Implement elasticity
4. Build security in every layer
5. Don't fear constraints
6. Think parallel
7. Leverage different storage options

Copyright © 2013, 2014 Amazon Web Services, Inc. and its affiliates. All rights reserved.

This work may not be reproduced or redistributed, in whole or in part, without prior written permission from Amazon Web Services, Inc. Commercial copying, lending, or selling is prohibited.

Errors or corrections? Email us at aws-course-feedback@amazon.com.

Other questions? Email us at aws-training-info@amazon.com.

All trademarks are the property of their owners.



Module 3: Security and Compliance

Before we start...

💡 Identify the correct statements:

Security and patching of the operating system and the application is the responsibility of the customer.

Penetration testing is a violation of the AWS Terms of Service.

Data on block storage devices (ephemeral storage and EBS) is encrypted by default.

Port scanning is performed by AWS to check for vulnerabilities in your application.

AWS is PCI DSS Level 1 certified, but customers are responsible for managing PCI compliance and certification for their own applications.

Each AWS Region has at least one Disaster Recovery Availability Zone.

Notes:

This slide provides an opportunity to test those ideas. Here are the answers:

- Security and patching of the OS: **True**.
- Penetration testing is a violation of the Terms of Service: **True**.
- Data on block storage is encrypted by default: **False**.
- Port scanning is performed by AWS: **False**.
- AWS is PCI DSS Level 1 certified: **True**.
- Each AWS region has one Disaster Recovery Availability zone. **False**. (Although every region has multiple Availability Zones, it is up to the customer to use them effectively for disaster recovery.)

Topics

- 💡 The shared responsibility security model
- 💡 AWS role in security
- 💡 Your role in security
- 💡 Securing networks with Security Groups

Notes:

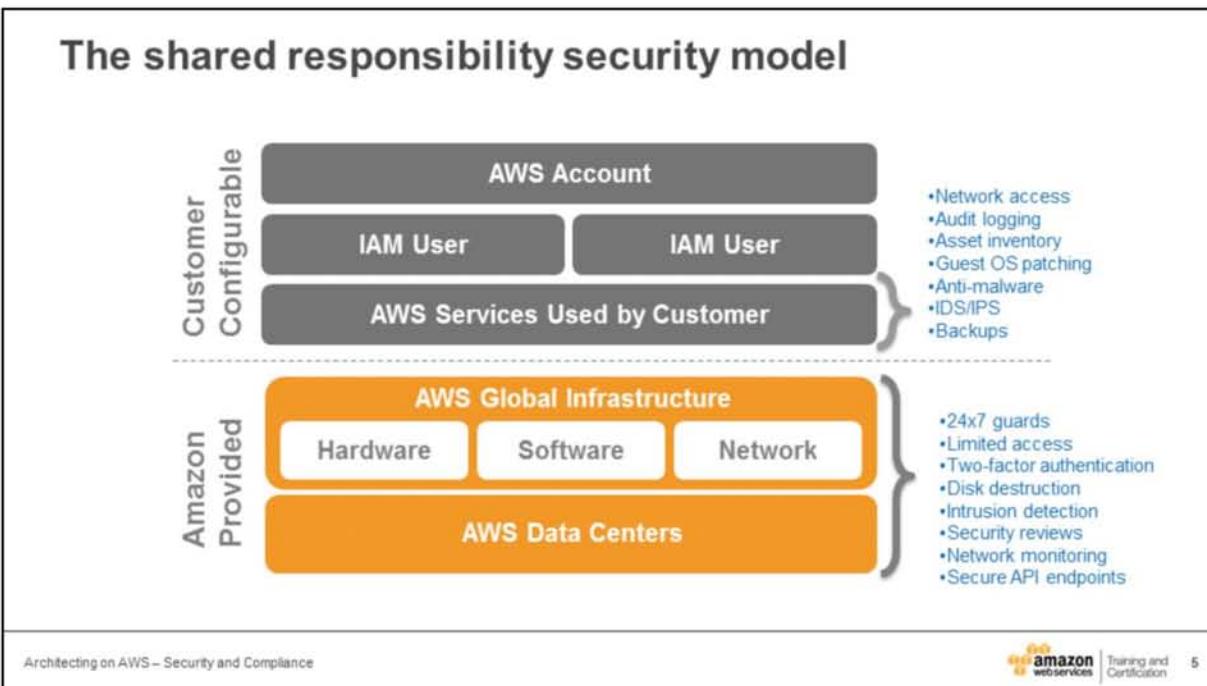
In this module, we'll cover the following topics:

- The Shared Responsibility Security Model
- AWS role in security
- Your role in security
- Securing networks with Security Groups

Topics

- 💡 The shared responsibility security model
- 💡 AWS role in security
- 💡 Your role in security
- 💡 Securing networks with Security Groups

Notes:



Notes:

Security within the cloud is a joint effort between you and the cloud provider (AWS). This diagram shows that AWS secures the underlying infrastructure, and you secure anything you put on top of that infrastructure.

What does “Securing the underlying infrastructure” mean? With AWS, it means we secure:

- Data centers – Non-descript facilities, 24x7 security guards, two-factor authentication, access logging and review, video surveillance, disk degaussing and destruction
- Hardware infrastructure – Servers, storage devices, and other appliances on which all of our services rely
- Software infrastructure – Host operating systems, service applications, and virtualization software
- Network infrastructure – Routers, switches, load balancers, firewalls, cabling, etc. Includes continuous network monitoring at external boundaries, secure access points, and redundant infrastructure.

You (the customer) are responsible for what you put on or connect to AWS. The security steps you need to take depend on the services you use and the complexity of your system. For example:

EC2

- Patching the guest OS on the instance

- Apps installed on the instance
- Firewalls
- Security Groups

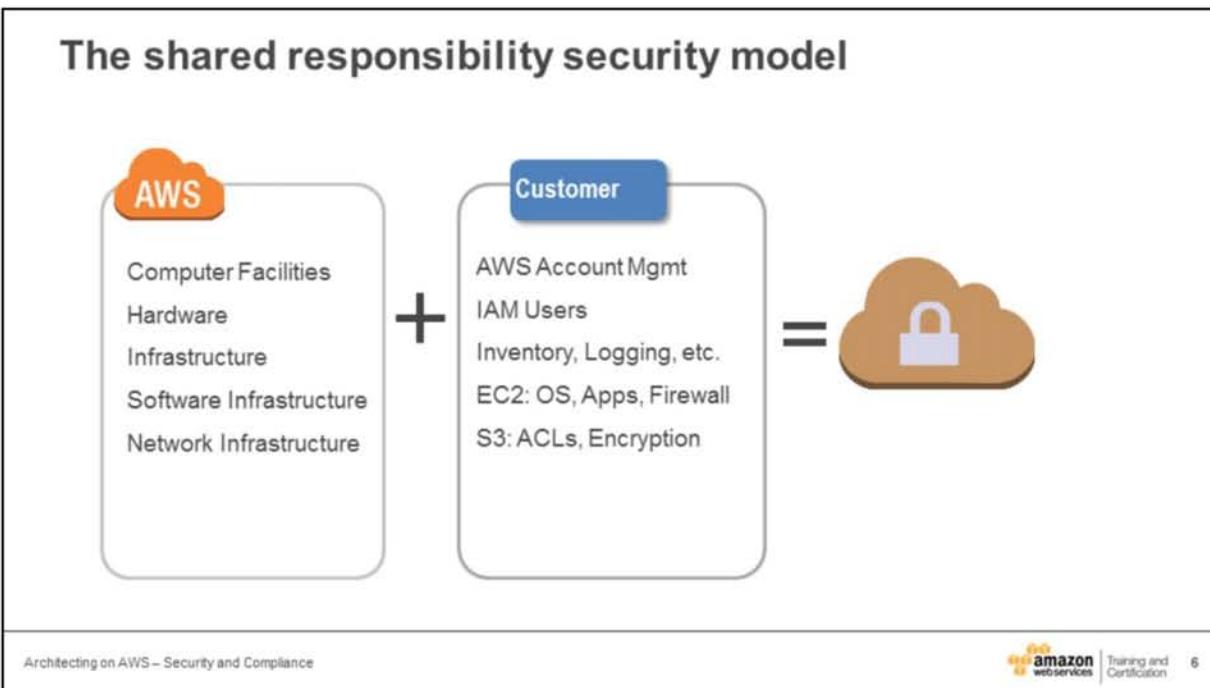
S3

- External access to S3
- Encryption

RDS:

- A managed service, so AWS handles things like backups and patching.
- You still need to configure network access and are responsible for the master account

And **ALWAYS** protect your AWS account!



Notes:

This diagram shows that the actions AWS takes, plus the action you take, help make your applications secure.

Topics

- 💡 The shared responsibility security model
- 💡 AWS role in security
- 💡 Your role in security
- 💡 Securing networks with Security Groups

Notes:

Discuss more details on the AWS role in security.

AWS role in Shared Responsibility Security Model

AWS

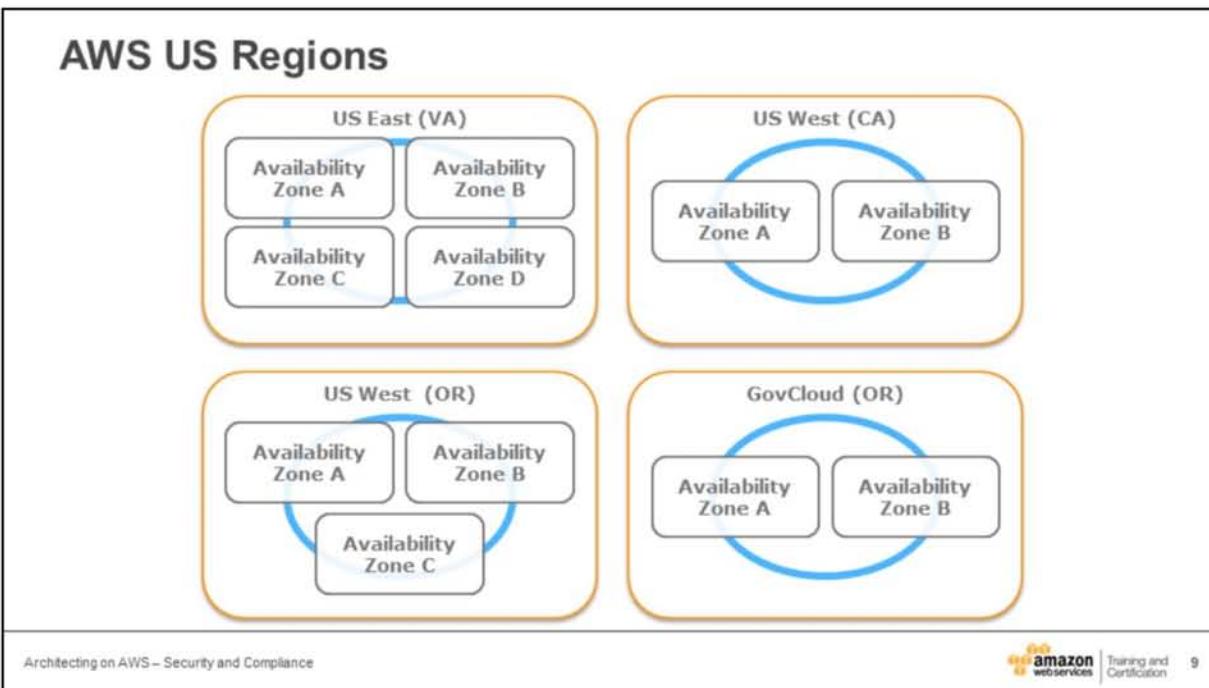
- Facilities
- Physical Security
 - Physical infrastructure
 - Network infrastructure
- Virtualization infrastructure
- Third-Party Attestations, Reports, and Certifications for the above

Customer

- Operating system
- Application
- Security groups
- OS Firewalls
- Network configuration
- Account Management
- Certifying your applications

Notes:

This slide shows the security elements AWS is responsible for.

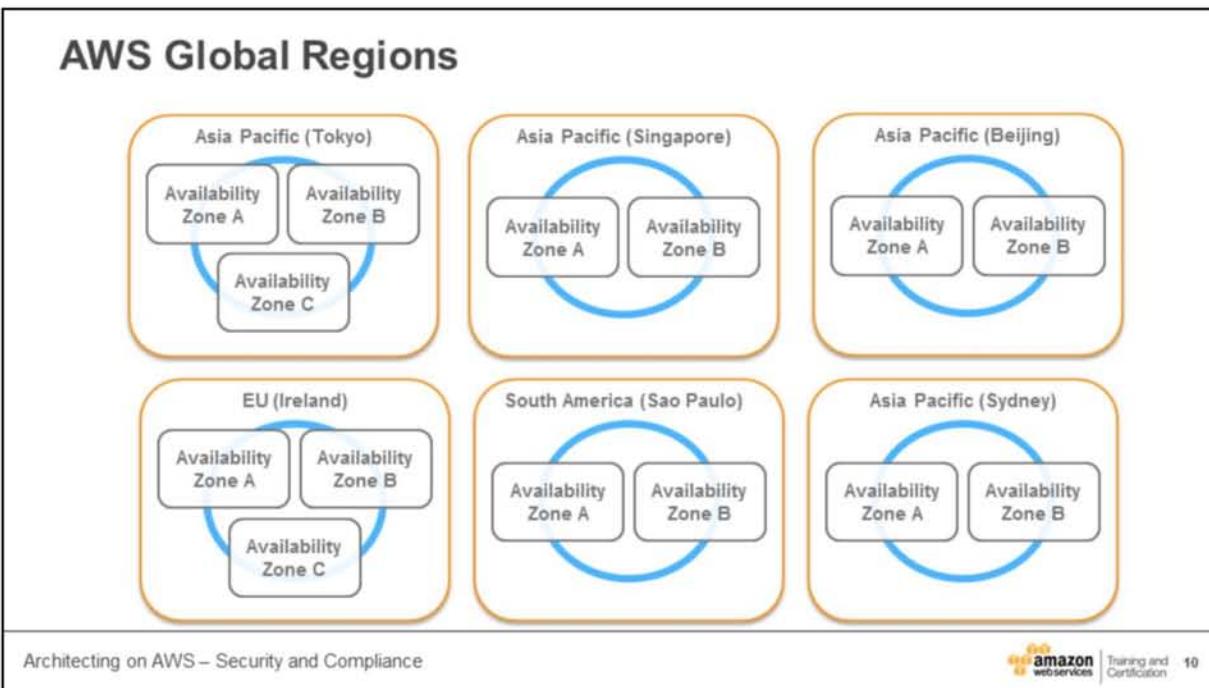


Notes:

AWS provides customers the flexibility to place instances and store data within multiple geographic Regions. Each Region is an independent collection of AWS resources in a defined geography.

AWS currently supports multiple regions; see <http://aws.amazon.com/about-aws/globalinfrastructure/> for a complete list.

The Amazon S3 US Standard Region includes the US East facilities in Northern Virginia and facilities in Western Washington State. The selection of a Region within an acceptable geographic jurisdiction to the customer provides a solid foundation to meeting location-dependent privacy and compliance requirements, such as the EU Data Privacy Directive.



Notes:

Data is not replicated between Regions unless proactively done so by the customer, thus allowing customers with these types of data placement and privacy requirements the ability to establish compliant environments.

All communications between Regions is across public Internet infrastructure.

Appropriate encryption methods should be used to protect sensitive data.

Amazon S3, Amazon Simple Notification Service (SNS), and Amazon Simple Queue Service (SQS) do not expose the concept of Availability Zones to customers. With these services, data is automatically stored on multiple devices across multiple facilities within a Region.

Physical Security

- 💡 Controlled, need-based access
 - All access is logged and reviewed
 - Multi-factor authentication
- 💡 Separation of Duties
 - Employees with physical access do not have logical access
- 💡 24 x 7 security guards

Notes:

Amazon has many years of experience in designing, constructing, and operating large-scale datacenters. This experience has been applied to the AWS platform and infrastructure.

AWS datacenters are housed in nondescript facilities

Physical access is strictly controlled both at the perimeter and at building ingress points by professional security staff utilizing video surveillance, intrusion detection systems, and other electronic means

Authorized staff must pass two-factor authentication a minimum of two times to access datacenter floors. All visitors and contractors are required to present identification and are signed in and continually escorted by authorized staff.

AWS only provides datacenter access and information to employees and contractors who have a legitimate business need for such privileges.

When an employee no longer has a business need for these privileges, his or her access is immediately revoked, even if they continue to be an employee of Amazon or Amazon Web Services. All physical access to datacenters by AWS employees is logged and audited routinely.

Network Security

- ⚠ Distributed Denial of Service (DDoS)
 - Standard mitigation techniques in effect
- ⚠ Man in the Middle (MITM)
 - All API endpoints protected by SSL
- ⚠ IP Spoofing
 - Prohibited at host OS level
- ⚠ Unauthorized Port Scanning
 - Violation of TOS
 - Detected, stopped, and blocked
- ⚠ Packet Sniffing
 - Promiscuous mode ineffective
 - Protection at hypervisor level

Notes:

Network security is the next important layer. This slide shows three common network security concerns, and how they can be addressed.

Storage Device Decommissioning

💡 Uses techniques from:

- DoD 5220.22-M ("National Industrial Security Program Operating Manual")
- NIST 800-88 ("Guidelines for Media Sanitization")

💡 Ultimately, all devices are:

- Degaussed
- Physically destroyed

Notes:

As the second bullet shows, ultimately all devices are degaussed and physically destroyed.

Virtual Memory and Local Disk Space

- 💡 Proprietary disk management prevents one instance from reading disk contents of another
- 💡 Disk is wiped upon creation
- 💡 Disks can be encrypted by customer

Notes:

This slide shows how security affects virtual memory and local disk space.

AWS Third-Party Attestations, Reports, and Certifications

AWS Environment

- Service Organization Controls (SOC) Reports
 - SOC 1 Type II (SSAE 16/ISAE 3402/formerly SAS70)
 - SOC 2 Type II
 - SOC 3
- Payment Card Industry Data Security Standard (PCI DSS) Level 1 Certification
- ISO 27001 Certification
- FedRAMP
- DIACAP and FISMA
- ITAR
- FIPS 140-2

Additional information available at
<https://aws.amazon.com/compliance/>

Note: The following applies to both slide 20 and 21.

While customers don't communicate their use and configurations to AWS, AWS does communicate its security and control environment relevant to customers.

AWS communicates its security and control environments by:

- Obtaining industry certifications and independent third party attestations described in this document
- Publishing information about the AWS security and control practices in whitepapers and web site content

Additional details about the AWS Compliance assurance programs at
www.aws.amazon.com/compliance/.

We also recommend you review the AWS Security Whitepaper, located at
www.aws.amazon.com/security/.

AWS Third-Party Attestations, Reports and Certifications (continue)

- Customers have deployed various compliant applications:
 - Sarbanes-Oxley (SOX)
 - HIPAA (healthcare)
 - FedRAMP (US Public Sector)
 - FISMA (US Public Sector)
 - ITAR (US Public Sector)
 - DIACAP MAC III Sensitive IATO

Note:

In addition, the flexibility and control that the AWS platform provides allows customers to deploy solutions that meet several industry-specific standards, including:

- HIPAA
- Cloud Security Alliance (CSA)
- Motion Picture Association of America (MPAA)

AWS provides a wide range of information regarding its IT control environment to customers through white papers, reports, certifications, accreditations, and other third-party attestations.

Topics

- 💡 The shared responsibility security model
- 💡 AWS role in security
- 💡 Your role in security
- 💡 Securing networks with Security Groups

Your role in Shared Responsibility Security Model

AWS

- Facilities
- Physical Security
 - Physical infrastructure
 - Network infrastructure
- Virtualization infrastructure
- Third-Party Attestations, Reports, and Certifications for the above

Customer

- Operating system
- Application
- Security groups
- OS Firewalls
- Network configuration
- Account Management
- Certifying your applications

EC2

Notes:

As an AWS customer, your role in security falls primarily into the categories shown here. Notice that the following security layers only apply if you're running EC2 instances:

- Operating System
- Application
- Security Groups
- OS Firewalls

Account Management

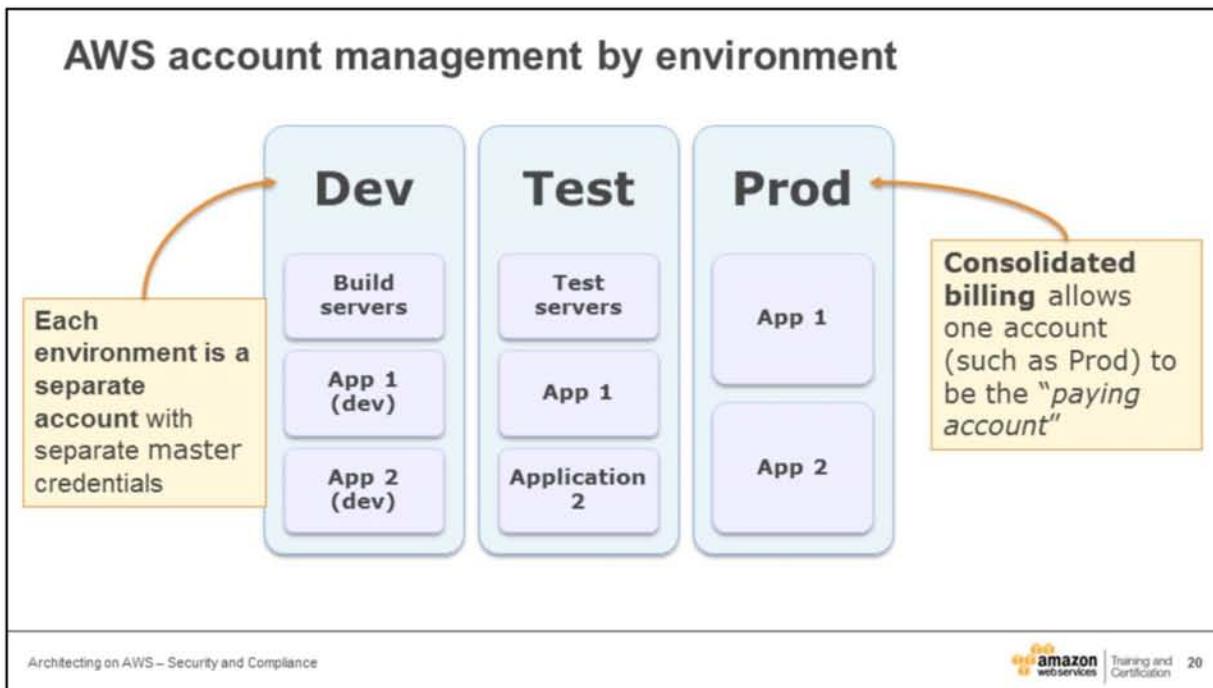
- 💡 Master (root) account has root/admin-level access
- 💡 Multiple accounts may be created to isolate resources
- 💡 Accounts may be isolated by:
 - Environment (dev, test, prod)
 - Major System
 - Line of business / function
 - Customer
 - Risk level

Notes:

Account Management is a critically important part of security—this is especially true with AWS. An AWS Account is a valuable asset, and must be kept safe.

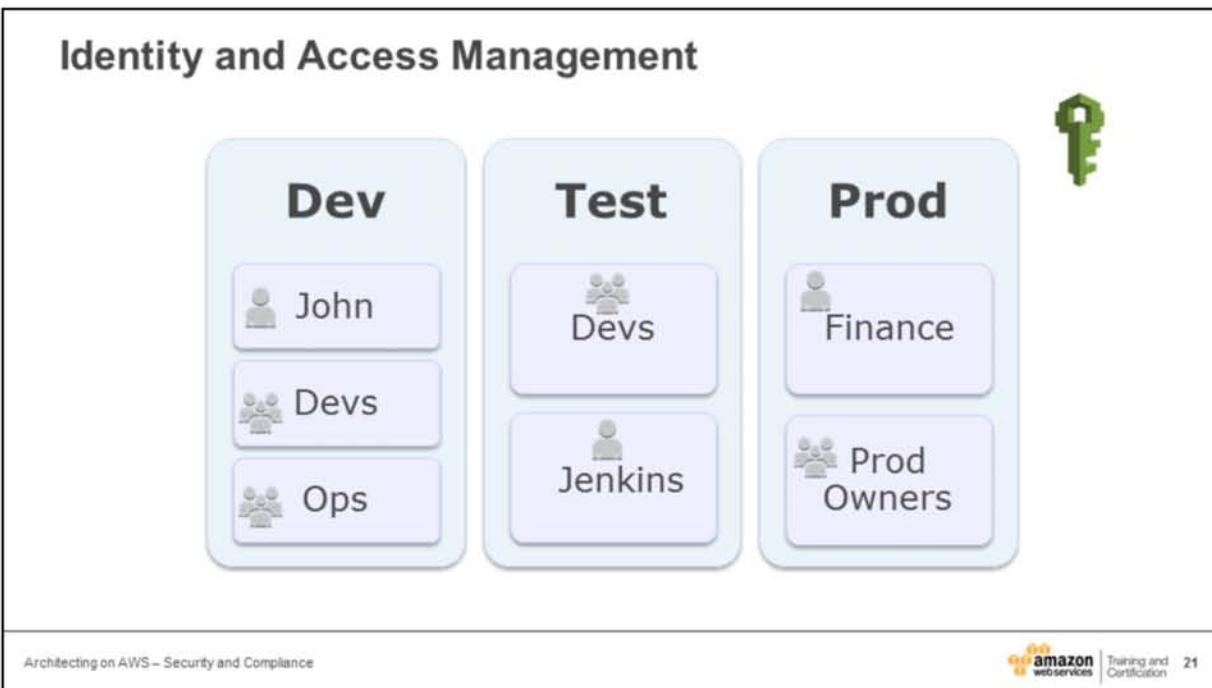
Often, companies just have a single AWS Account. This is not recommended.

Instead, create multiple accounts, and use various methods to isolate these accounts from each other.



Notes:

In this slide, you can see how you can separate AWS accounts by which environment they need to access. When you create multiple accounts within AWS, you can consolidate billing, so only one account is the "paying" account.



Notes:

IAM provides a great way of keeping with our “least privilege” concept that we discussed earlier in the day. Here’s the same example from earlier, but using IAM to create different accounts under a single AWS account.

Operating system security

💡 Guest (instance) operating system

- Customer controlled (customer owns root/admin)
- AWS admins cannot log in ← Why not?

💡 EC2 Key Pairs

- You (and only you) have the private half of the key
- You (and only you) can:
 - SSH to the instance (Linux)
 - Decrypt the Administrator password (Windows)

Notes:

OS security is controlled by you, and is separate from AWS accounts.

Just because you have an AWS account, doesn't mean you can log into an instance.

AWS uses public-key cryptography to encrypt and decrypt login information.

Public-key cryptography uses a public key to encrypt a piece of data, such as a password, then the recipient uses the private key to decrypt the data. The public and private keys are known as a key pair.

Operating system security

💡 You still need to patch

- Most traditional tools will work
- Emerging options
 - Chef (www.opscode.com/chef)
 - Puppet (www.puppetlabs.com)
 - Fabric/Cuisine (www.fabfile.org)
 - Capistrano (<https://github.com/capistrano/capistrano/wiki>)
 - Amazon OpsWorks (<https://aws.amazon.com/opsworks/>)

Notes:

OS security is another level of security for which you're responsible. This slide shows some areas of OS security that customers are responsible for.

Your Data

- 💡 Protect privacy and enforce your policies with data encryption
- 💡 Encrypt data in transit
 - (SSL/TLS)
- 💡 Encrypt data at rest
 - Consider encrypted file systems for sensitive data
 - Encrypt objects before storing them
 - Encrypt records before writing in database
- 💡 EBS and Ephemeral volumes can be encrypted
- 💡 Variety of options
 - EncFS, Loop-AES, dm-Crypt, TrueCrypt, etc...

Notes:

In AWS, volume storage comes in two varieties: Elastic Block Storage (which persists after an instance is terminated) or Ephemeral volumes (which don't survive termination). You can encrypt both storage types in a variety of ways. As you would do in your own datacenters, you should encrypt data in motion or at rest as appropriate.

If you store your policies in revision control, you can periodically check that the policy in use matches what you have in revision control, and perhaps alert if it has been changed.

Encryption: File Systems

Managing encryption keys

- Study key management capabilities of encryption product(s) you choose
- Establish a procedure that minimizes possibility of losing keys
- AWS CloudHSM
 - Securely generate, store and manage cryptographic keys used for data encryption
 - Dedicated SafeNet Luna SA

Notes:

It's up to you to know and understand encryption options for the products you use.

AWS CloudHSM is one option to help keep things secure. The AWS CloudHSM service provides customers with dedicated access to a hardware security module (HSM) appliance designed to provide secure cryptographic key storage and operations within an intrusion-resistant, tamper-evident device. You can generate, store, and manage the cryptographic keys used for data encryption such that they are accessible only by you.

When you sign up for CloudHSM, you receive dedicated single tenant access to CloudHSM appliances. Each appliance appears as a resource in your Amazon VPC. You, not AWS, initialize and manage the cryptographic domain of the CloudHSM. The cryptographic domain is a logical and physical security boundary that restricts access to your keys. Only *you* can control your keys and operations performed by the CloudHSM. AWS administrators manage, maintain, and monitor the health of the CloudHSM appliance, but do not have access to the cryptographic domain. After you initialize the cryptographic domain, you can configure clients on your EC2 instances that allow applications to use the APIs provided by CloudHSM.

To learn more, go to <http://aws.amazon.com/cloudhsm>.

Use Multiple Security Layers

- 💡 Security Groups (EC2, VPC, RDS, ElastiCache, Redshift)
- 💡 Bastion Host
- 💡 Host-based Firewalls*
- 💡 IDS*

* Third-Party tools/solutions

Notes:

Good security requires multiple layers. These are just a few examples of how multiple layers of security can help protect your AWS systems.

Topics

- 💡 The shared responsibility security model
- 💡 AWS role in security
- 💡 Your role in security
- 💡 Securing networks with Security Groups

Network Security: Security Groups

- 💡 Control inbound traffic
 - VPC groups can also control outbound
- 💡 Apply many Security Groups to one instance
- 💡 Default group: no access
- 💡 Several services use Security Groups
 - EC2
 - VPC (more advanced features)
 - RDS
 - ElastiCache
 - Redshift

Notes:

Here's an overview of what security groups can do. You can use security groups for a variety of AWS services, such as those shown in this slide.

Network Security: Security Groups (continue)

- 💡 When defining inbound rules, specify source by:
 - CIDR address
 - 0.0.0.0/0 for Internet, 10.0.0.0/16 for EC2 private, and so on
 - Security Group Name
 - Restrict access to other EC2 instances in the specified security group

Notes:

CIDR Notation: An Overview

- 💡 CIDR notation is useful for expressing a range of IP addresses
- 💡 Consider this IP(v4) address:

216.173.122.34


- Each number can have a decimal value between 0 and 255
- Each number is a single byte (8 bits)

Notes:

Let us quickly review CIDR notation and how to define CIDR block. Feel free to move quickly through these slides (or skip them) if you do not require to review CIDR notation.

CIDR Notation: An Overview

- 💡 What if you wanted to express a firewall rule that allowed traffic from any address in the last octet?

216.173.122.* 

- Specify the first valid number in the range. If we want to allow all values in the last octet, the first allowable value is 0

216.173.122.0


- In this case, we want to freeze the first 3 octets: **3 octets == 3 bytes == 24 bits**
Therefore → 216.173.122.0/**24**

CIDR Notation: A few more examples

- Match an exact address:

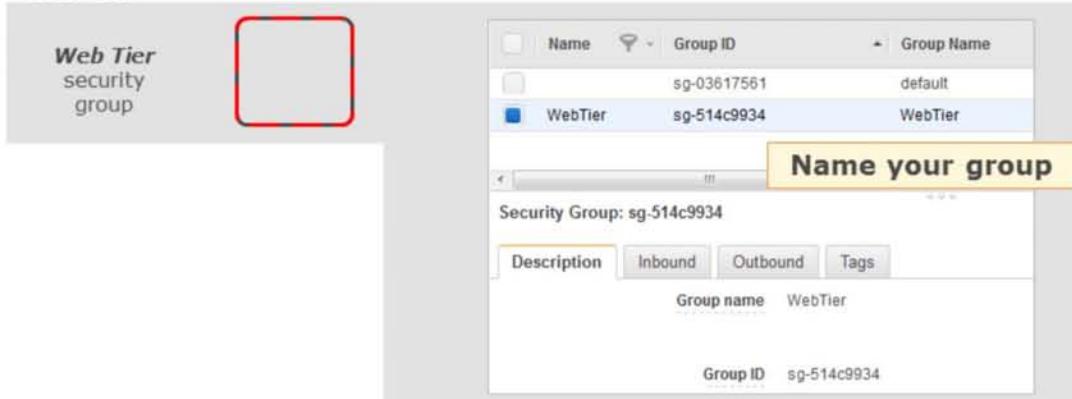
216.173.122.34  216.173.122.34/32

- Match any address:

..*.*  0.0.0.0/0

Security Groups Example: Web Server Instance (1 of 3)

- Design a security group for Apache web servers in your application's web tier



Notes:

In the next few slides, we'll show how security groups work with a Web Server instance.

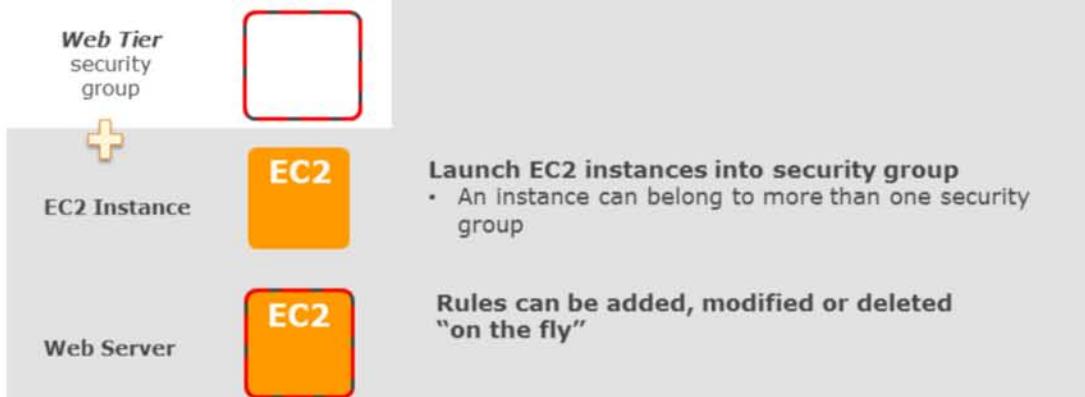
Security Groups Example: Web Server Instance (2 of 3)



Notes:

Next, we define some rules for the group, such as ports, protocols, and source.

Security Groups Example: Web Server Instance (3 of 3)



Notes:

With our Security Group in place, we can now launch instances into it. You have the option to change the rules for a Security Group at any time.

Security Groups Example: Multi-tier Security Group Activity

Define the Groups

Tier	Port	Source
Web	80	0.0.0.0/0
	443	0.0.0.0/0
	22	Bastion
App	22	Bastion
	8000	Web
DB	3306	207.171.191.92/32
	3306	App
	22	Bastion
Bastion	22	207.171.191.60/32

Architecting on AWS – Security and Compliance

amazon web services | Training and Certification 36

Notes:

As this slide shows, we have three servers: a web server, an app server, and a database server.

- Web server communicates with app server through port 8000 (app specific)
- App server communicates with database server through port 3306 (MySQL)
- Web server accessible to Internet through port 80 (HTTP) and/or port 443 (HTTPS)

All three servers permit administrator access through port 22 (SSH), but only from inside the company's network.

Notice that:

- Anyone can access the Web server through HTTP
- Administrators can remote into a Bastion Host to SSH into the other systems
- All other Internet ports are blocked by default.

Let's look at the security group table on the left. First, we have our Bastion Host, which is only accessible through port 22.

Secondly, we have our database, which is accessible as follows:

- Bastion Host can access through port 22
- App server can access through port 3306
- Specific IP address can access through port 3306 for DB synchronization

Next, we have the App servers, which are accessible by:

- The Bastion Host, through port 22
- The Web server, through port 8000

Lastly, we have the Web servers, which are accessible by:

- Anyone on the Internet, through port 80 and port 443
- The Bastion Host, through port 22
- By creating these security groups, you can help keep your applications secure

Most security best practices still apply in the Cloud

- Secure coding standards
- Perform penetration testing
 - <http://aws.amazon.com/security/penetration-testing>
- Antivirus where appropriate
- Intrusion Detection
 - Host-based Intrusion Detection (such as OSSEC)
- Log events
- Role-based access control
 - AWS Identity & Access Management
 - LDAP and/or Active Directory for Operating Systems & Applications

Notes:

Your security measures that you already have in place probably also apply to the cloud as well.

Module review

- 💡 What are the five main layers of security for cloud architecture?
- 💡 What security model is used with AWS services
- 💡 What areas of security is AWS responsible for?
- 💡 What areas of security are you, the customer, responsible for?

Answers:

What are the five main layers of security for cloud architecture?

- * Physical
- * Network
- * Data (in transit and at rest)
- * OS
- * API credentials

What security model is used with AWS services?

- * Shared Responsibility security model

What areas of security is AWS responsible for?

- * Facilities
- * Physical security
- * Virtualization infrastructure
- * Certifications for the above

What areas of security are you, the customer, responsible for?

- * Operating system
- * Application
- * Security groups
- * OS Firewalls
- * Network configuration

- * Account management
- * Certifying your apps

Copyright © 2014 Amazon Web Services, Inc. or its affiliates. All rights reserved.

This work may not be reproduced or redistributed, in whole or in part, without prior written permission from Amazon Web Services, Inc. Commercial copying, lending, or selling is prohibited.

Errors or corrections? Email us at aws-course-feedback@amazon.com.
Other questions? Email us at aws-training-info@amazon.com.

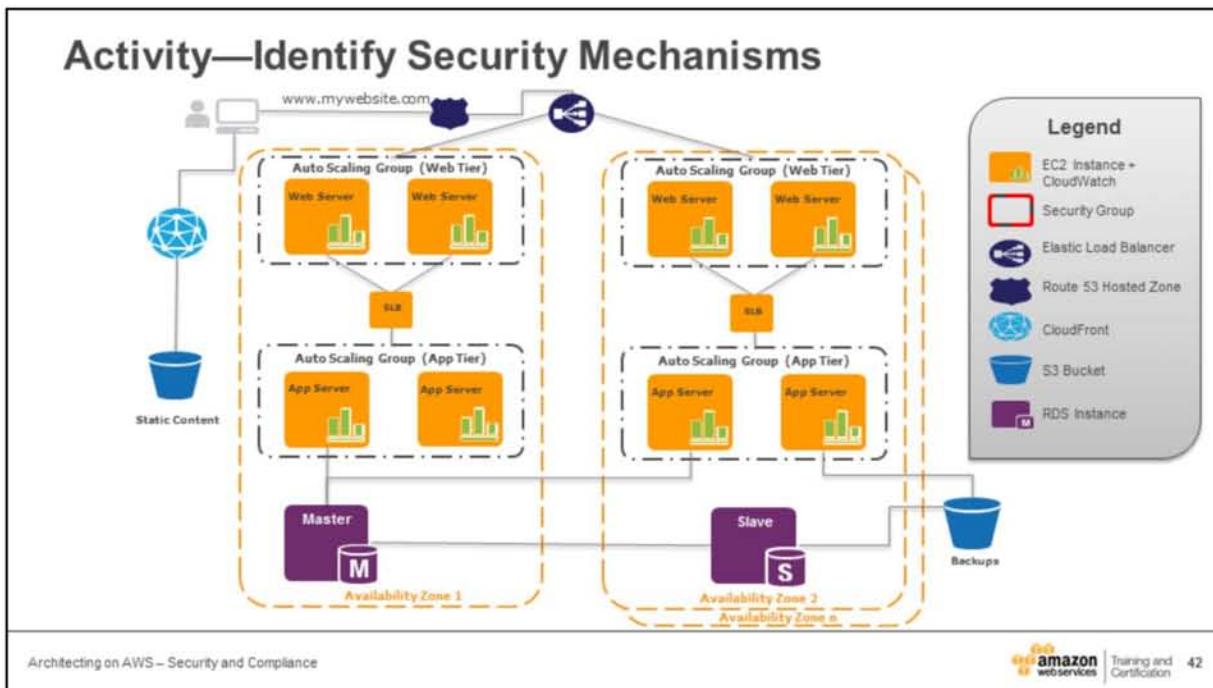
All trademarks are the property of their owners.

Appendix

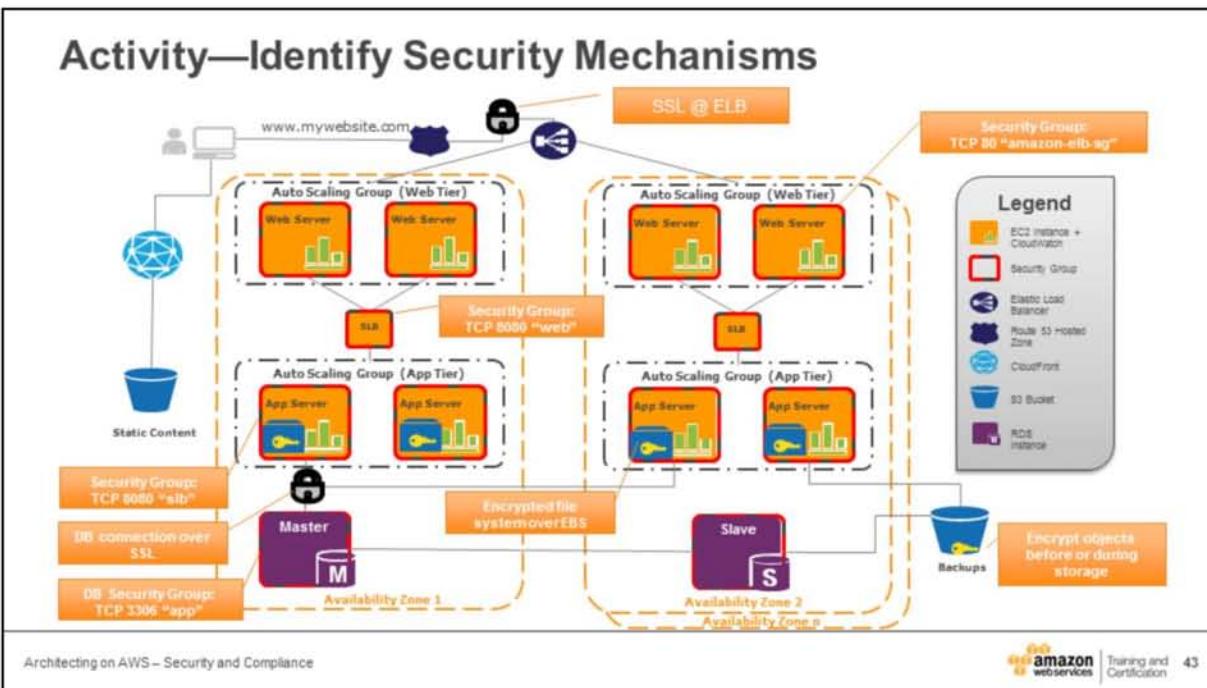
Activity—Identify Security Mechanisms

- 💡 Consider the architecture for a scalable web application. How do you secure it? Address the following aspects of security:
- Physical
 - Network
 - Data (in transit and at rest)
 - Operating system
 - Security credential management
 - Logging

The answers to this question are available in the following slides.



Here's a basic system. Remember, it is important to build security into every layer of your design.



Here is one way to implement security across multiple layers of a system. Notice that there are security groups for:

- Web servers
- App servers
- Load Balancers
- Databases

In addition:

- Data is encrypted as external users access the site through SSL
- Data in the app servers are encrypted on Amazon Elastic Block Storage (EBS)
- Data between app servers and databases are encrypted through SSL
- Backups are encrypted during storage



Module 4: Amazon VPC

Topics

- 💡 What is Amazon VPC?
- 💡 Subnets, gateways, and routes
- 💡 Advanced Features

Topics

- 💡 What is Amazon VPC?
- 💡 Subnets, gateways, and routes
- 💡 Advanced Features

What is Amazon VPC?

Specify your own private IP address range from any ranges you choose

Divide your private IP address range into one or more public or private subnets



Amazon Virtual Private Cloud (Amazon VPC) enables you to launch Amazon Web Services (AWS) resources into a **virtual network** that you've defined.

Control inbound and outbound access to and from individual instances

Bridge your VPC and your onsite IT infrastructure with an encrypted VPN connection

Assign multiple IP address and attach multiple ENIs and EIPs to EC2 instances

Notes:

Amazon Virtual Private Cloud (Amazon VPC) enables you to launch Amazon Web Services (AWS) resources into a **virtual network** that you've defined.

More information on Amazon VPC is available at <http://aws.amazon.com/vpc>.

Topics

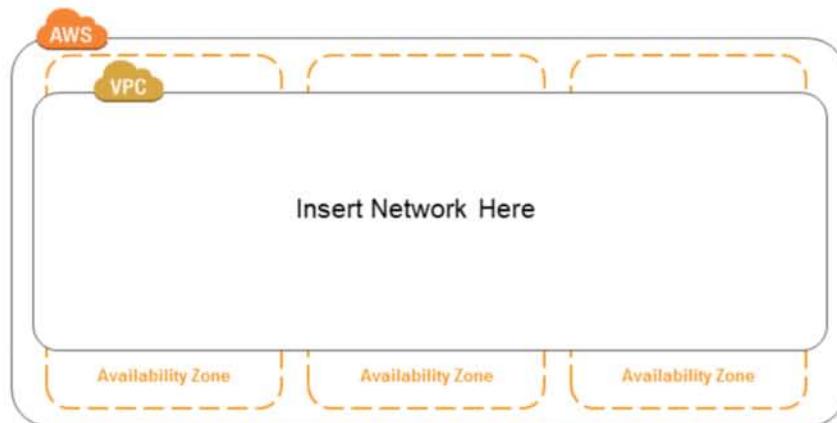
- 💡 What is Amazon VPC?
- 💡 Subnets, gateways, and routes
- 💡 Advanced Features

Notes:

With Amazon VPC, you can define a virtual network topology that closely resembles a traditional network that you might operate in **your own data center**.

Subnets, Gateways and routes

- Select an AWS region for your VPC



Architecting on AWS – Amazon VPC

 Training and Certification 6

Notes:

Over the next few slides, we'll talk about how Amazon VPC fits into your system architecture. The first step is to select a region for your VPC.

Components of Amazon VPC

Amazon VPC is comprised of a variety of objects that will be familiar to customers with existing networks:

- A Virtual Private Cloud (VPC): a logically isolated virtual network in the AWS cloud. You define a VPC's IP address space from a range you select.
- Subnet: a segment of a VPC's IP address range where you can place groups of isolated resources.
- Internet Gateway: the Amazon VPC side of a connection to the public Internet.
- NAT Instance: An EC2 instance that provides Port Address Translation for non-EIP instances to access the Internet via the Internet Gateway.
- Hardware VPN Connection: a hardware-based VPN connection between your Amazon VPC and your datacenter, home network, or co-location facility.
- Virtual Private Gateway: the Amazon VPC side of a VPN Connection.
- Customer Gateway: Your side of a VPN Connection.
- Router: Routers interconnect Subnets and direct traffic between Internet Gateways, Virtual Private Gateways, NAT instances and Subnets.
- Peering Connection: A peering connection enables you to route traffic via private IP addresses between two peered VPCs.

Subnets, Gateways and routes

- Specify the size of our network with a CIDR block



Architecting on AWS – Amazon VPC

 amazon web services | Training and Certification 7

Notes:

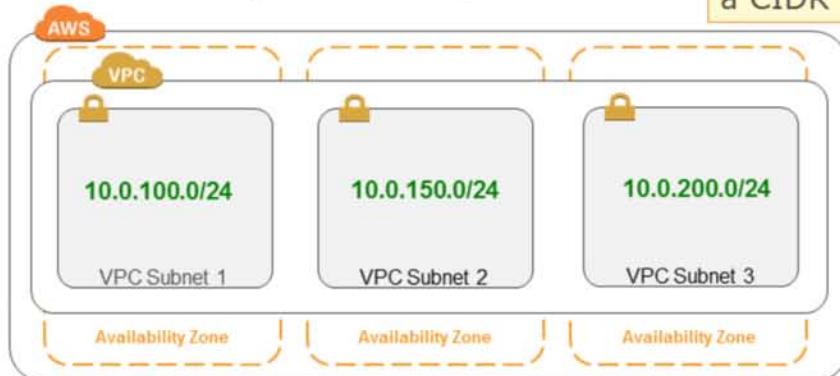
Next, you select an IP address range that the VPC will use. Consider carefully what this range should be. For example

- Allocate unique addresses
 - Provide a large enough range to account for future growth.
- AWS is flexible, but nothing substitutes good planning!

Subnets, Gateways and routes

- 💡 Create **subnets** inside the VPC
 - Subnets **do not** span Availability Zones

Each subnet has a CIDR range



Notes:

After you select a region and define your IP address range, you can start to create subnets within the VPC. Subnets don't span Availability Zones.

Each subnet has a CIDR range—in this case:

- Subnet 1: 10.0.100.0/24
- Subnet 2: 10.0.150.0/24
- Subnet 3: 10.0.200.0/24

Subnets, Gateways and routes

- 💡 Attach gateway devices to network



Architecting on AWS – Amazon VPC

amazon web services | Training and Certification 9

Notes:

Next, you need to add gateway devices to allow access to the VPC. An Internet Gateway (IGW) is critical to allow subnets to route traffic to the Internet. In other words, IGW is Amazon VPC side of a connection to the public Internet.

Subnets, Gateways and routes

- VPN Gateway (VGW) allows subnet(s) to route to on-premises network over IPSEC VPN



Architecting on AWS – Amazon VPC

amazon
web services | Training and
Certification 10

Notes:

A VPN gateway allows for users within your network to access subnets over IPSEC VPN.

By default, instances that you launch into a virtual private cloud (VPC) can't communicate with your own network. You can enable access to your network from your VPC by attaching a virtual private gateway to the VPC, creating a custom route table, and updating your security group rules. For more information about adding a hardware virtual private gateway, refer to online documentation → http://docs.aws.amazon.com/AmazonVPC/latest/UserGuide/VPC_VPN.html

Subnets, Gateways and routes

- Define custom routing rule(s) for each subnet



Architecting on AWS – Amazon VPC

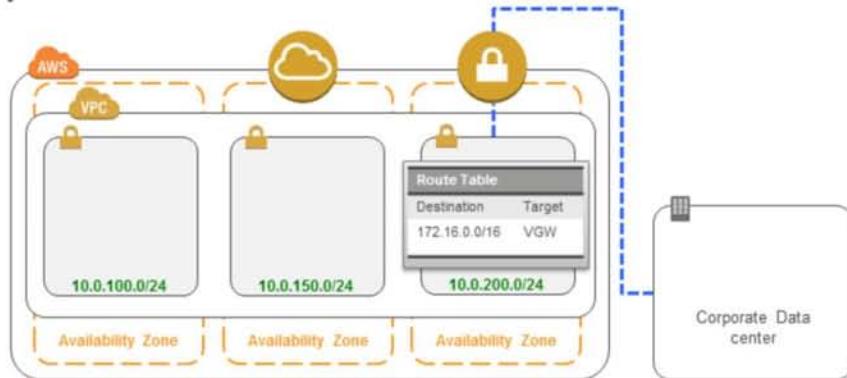
amazon
web services | Training and
Certification 11

Notes:

Another option is create custom routing rules for each subnet. Let's look at how this works.

Subnets, Gateways and routes

- Private subnet with IPSEC VPN to corporate datacenter via VGW



Architecting on AWS – Amazon VPC

amazon web services | Training and Certification 12

Notes:

In this example, Subnet 3 is a private subnet that access an on-premises data center through a Virtual Gateway.

Subnets, Gateways and routes

- Public subnet with inbound/outbound Internet connectivity via IGW



Architecting on AWS – Amazon VPC

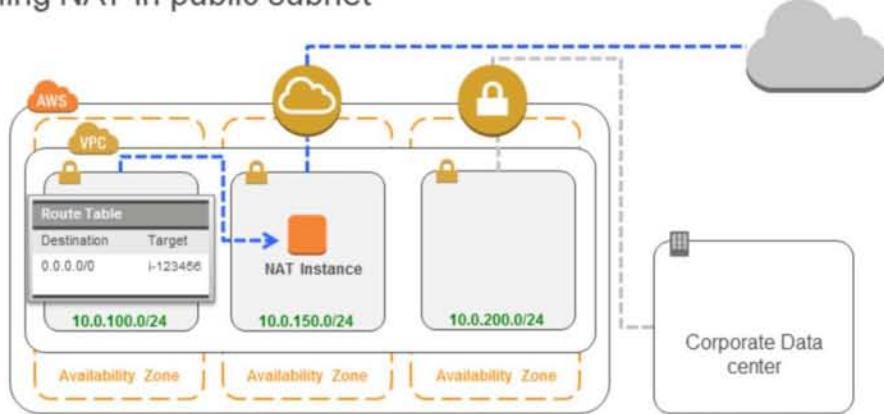
amazon web services | Training and Certification 13

Notes:

Subnet 2, on the other hand, is a public subnet, with Internet connectivity through an Internet Gateway.

Subnets, Gateways and routes

- Private subnet with outbound Internet connectivity via EC2 Instance running NAT in public subnet



Architecting on AWS – Amazon VPC

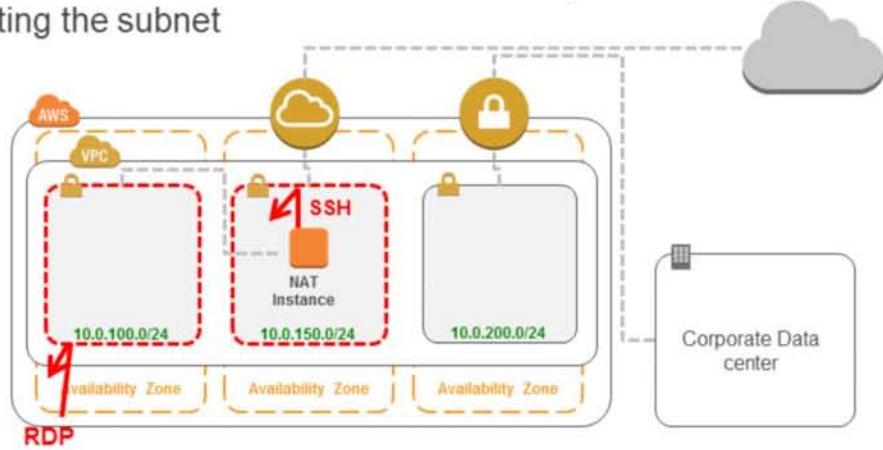
amazon web services | Training and Certification 14

Notes:

Lastly, Subnet 1 is a private subnet that can connect to the Internet, but only through an EC2 instance running in Subnet 2, which is a public subnet.

Subnets, Gateways and routes

- Network Access Control Lists allow or deny traffic entering or existing the subnet



Architecting on AWS – Amazon VPC

amazon web services | Training and Certification 15

Notes:

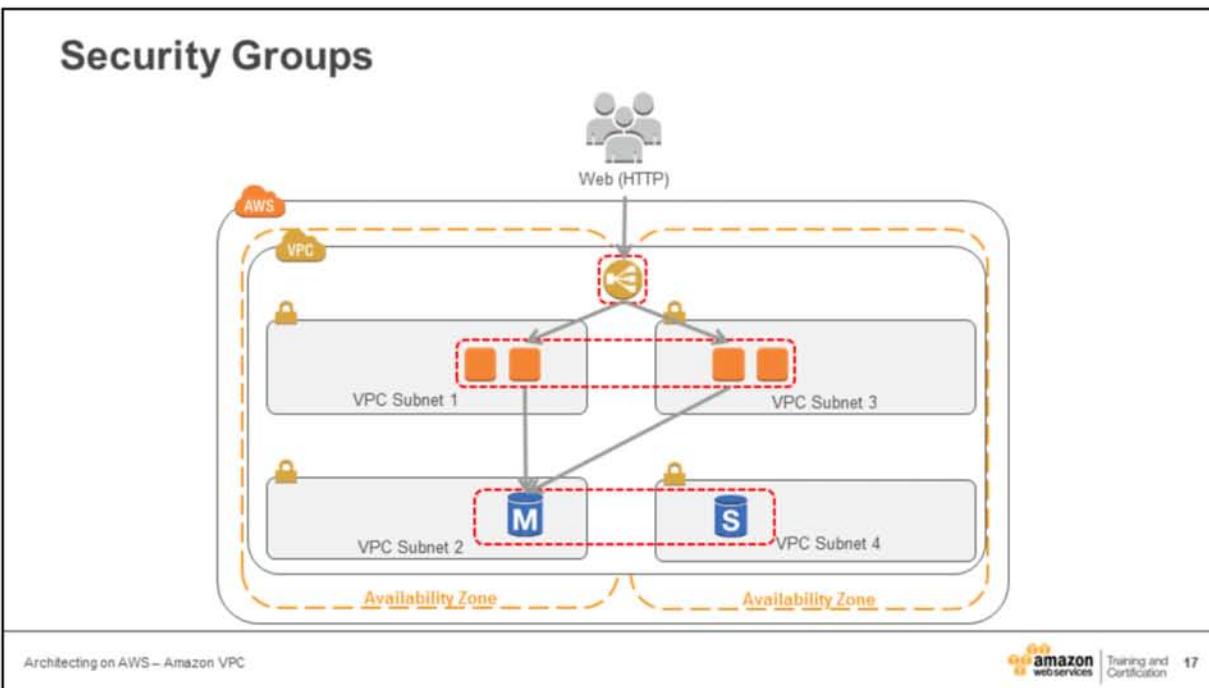
Notice that our VPC configuration prevents RDP traffic from entering Subnet 1. In addition, instances in Subnet 2 can't SSH into other systems.

Topics

- 💡 What is Amazon VPC?
- 💡 Subnets, gateways, and routes
- 💡 Advanced Features

Notes:

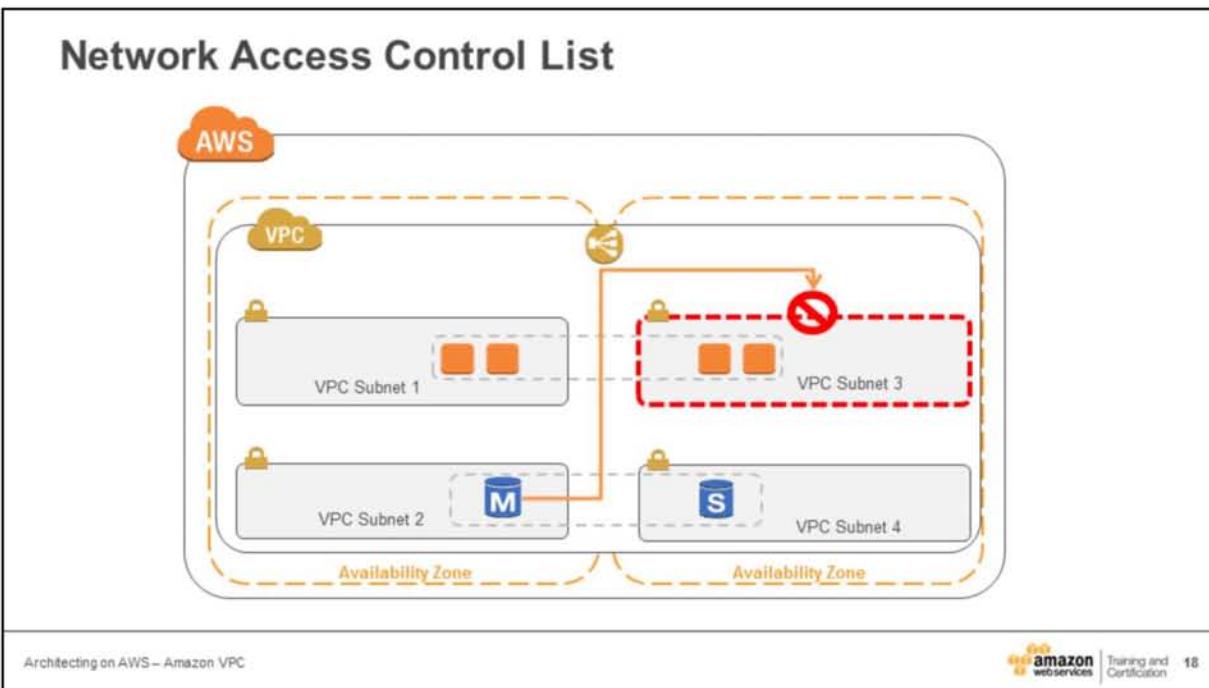
Amazon Virtual Private Cloud (Amazon VPC) includes features such as security groups, network access control lists, and Elastic Network Interfaces (ENIs).



Notes:

Security groups provides full control over inbound and outbound traffic. You can control traffic at the IP or security group level. Security groups can reference each other using ingress and egress rules. This allows you to move beyond traditional network security approaches. You can secure assets based on function or data classification. Remember, security groups are dynamic, which allows your firewall infrastructure to be just as elastic as your cloud resources.

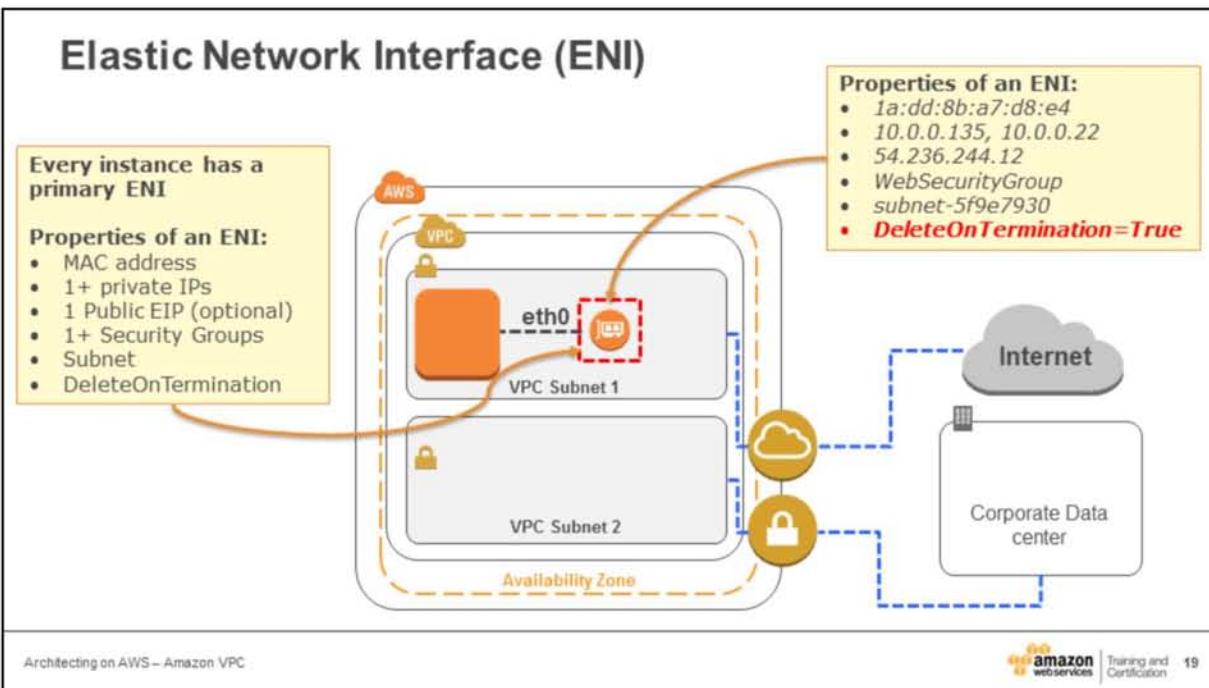
This example shows a 3-tier network topology. In this example, a 3-tier hierarchical network topology is implemented where users can only access a web site through a load balancer, only the load balancer can connect to the web servers, and only the web servers can connect to the database servers.



Notes:

In this example:

- Users can only access a web site through a load balancer
- Only the load balancer can connect to the web servers
- Only the web servers can connect to the database servers

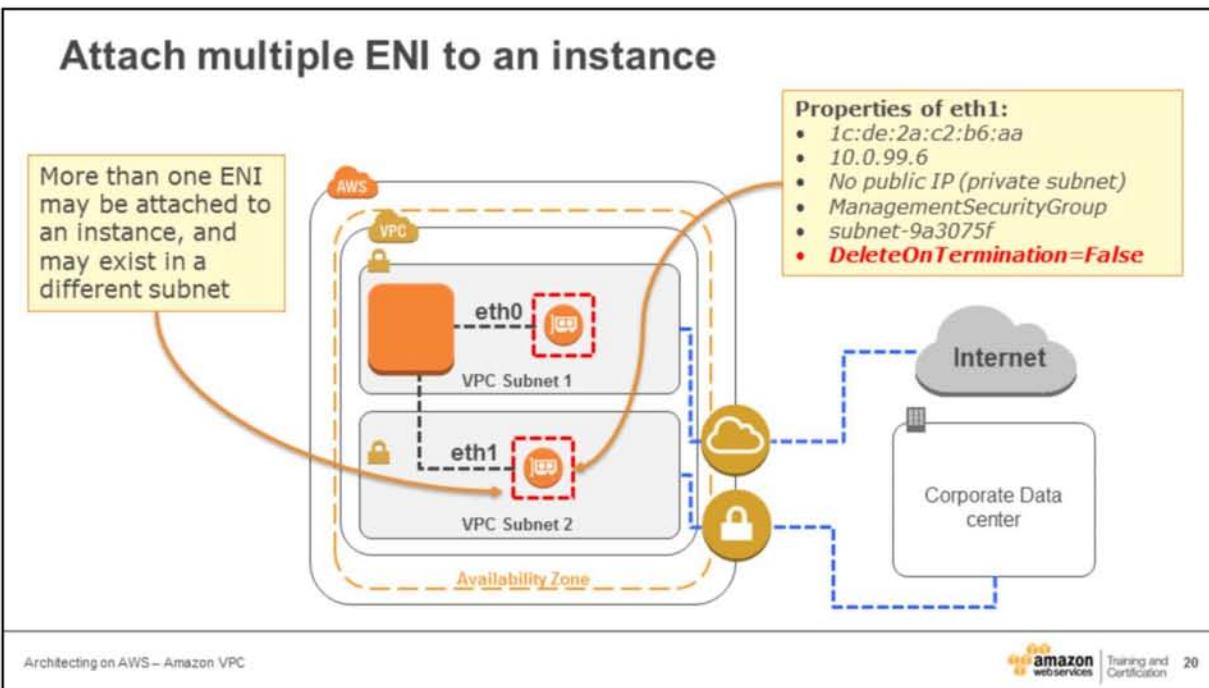


Notes:

Another component of Amazon VPC is the Elastic Network Interface, or ENI. An ENI is a virtual network interface that you can attach to an instance in a VPC. For access to S3, SQS, SNS (and other API interactions), the instance needs to have public access or access to the local AWS public subnet service APIs (as depicted in the diagram). Except for default VPCs, EC2 instances are not automatically allocated an Elastic IP address.

As you can see here, an ENI can have a variety of properties, including MAC address, or one more private IPs, at least one security group, and so on.

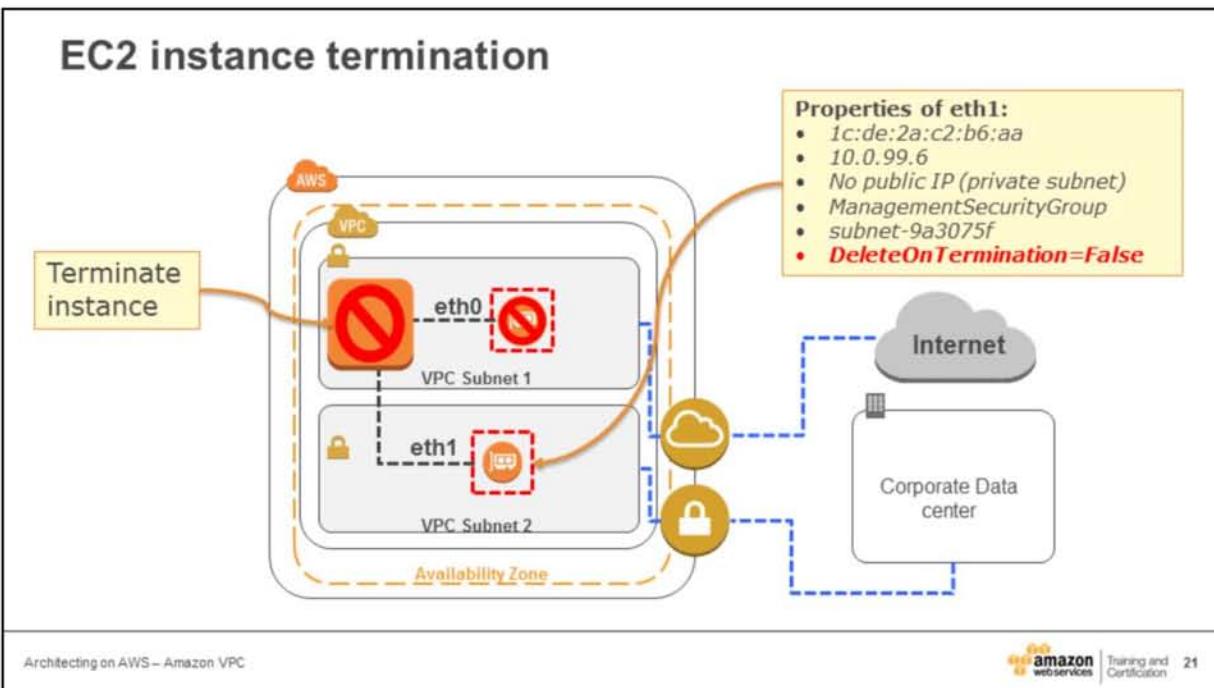
Notice that, in this case, we've set *DeleteOnTermination* to **True**.



Notes:

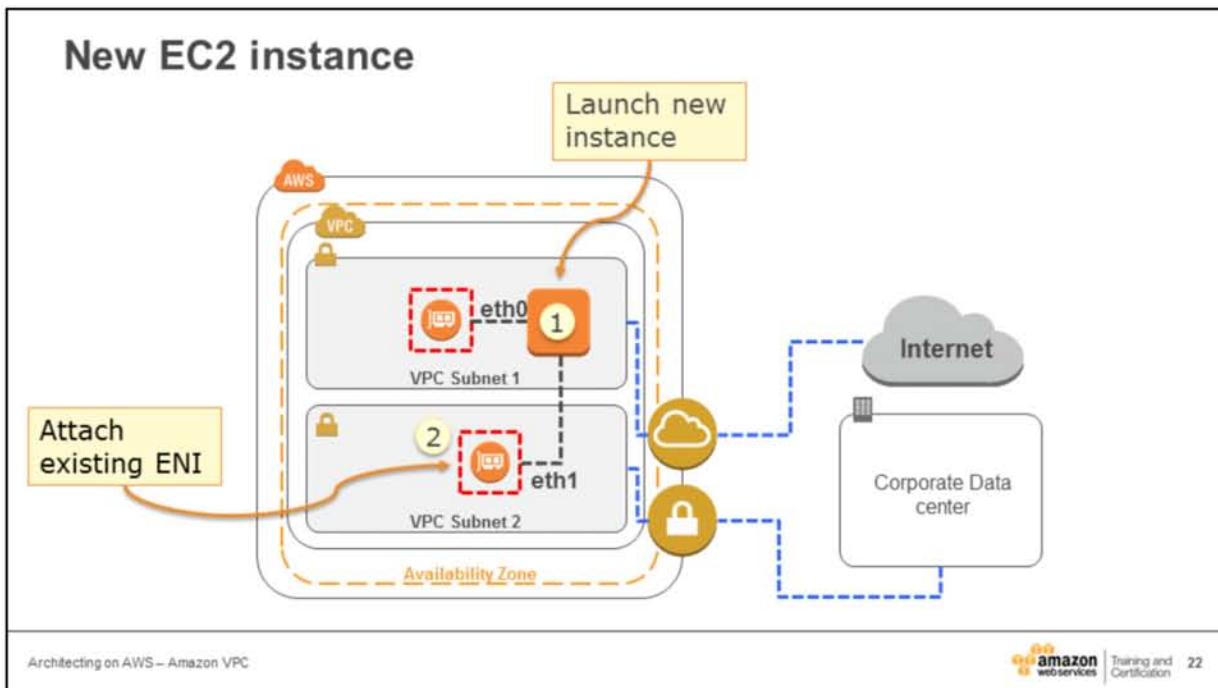
You can attach more than one ENI to an instance, detach it, and re-attach it to a new instance as needed. As you can see here, we've attached to ENIs to the same EC2 instance, with the second ENI belonging to a completely different subnet from the first one.

Notice that, unlike the other ENI, we set `DeleteOnTermination` to **False**.



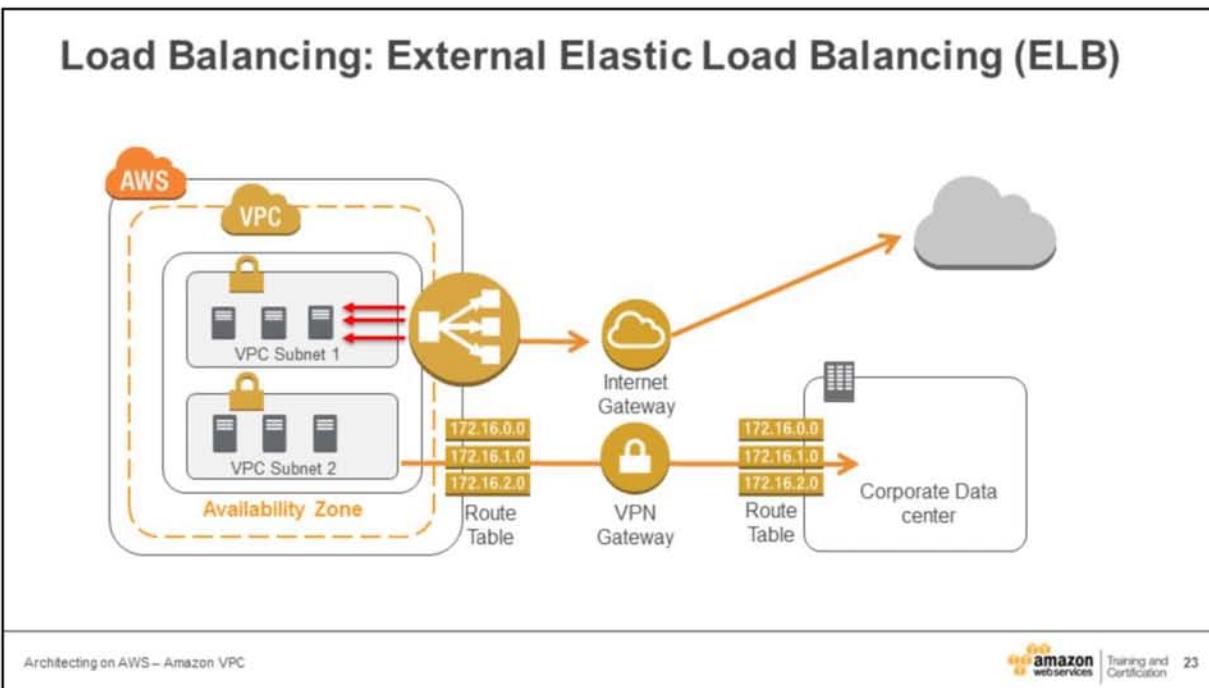
Notes:

This means that, when we terminate the instance. Our second ENI still exists. Why would we do this?



Notes:

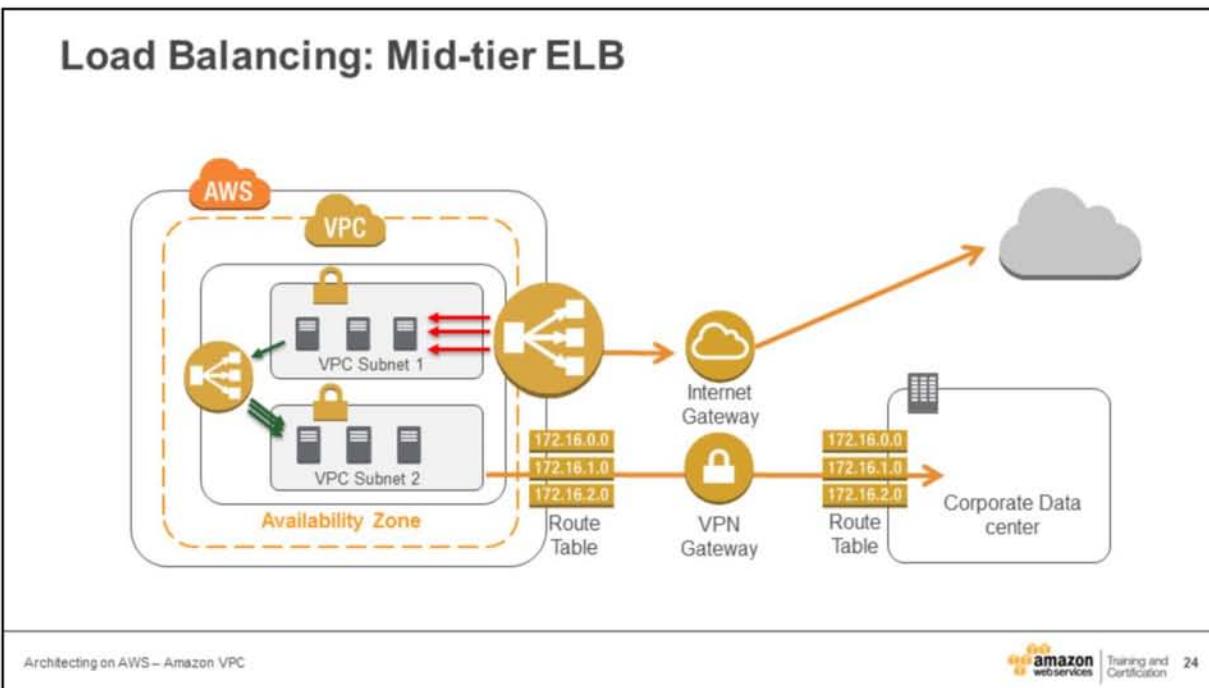
1. If we spin up a new EC2 instance
2. We can attach the existing ENI to it



Notes:

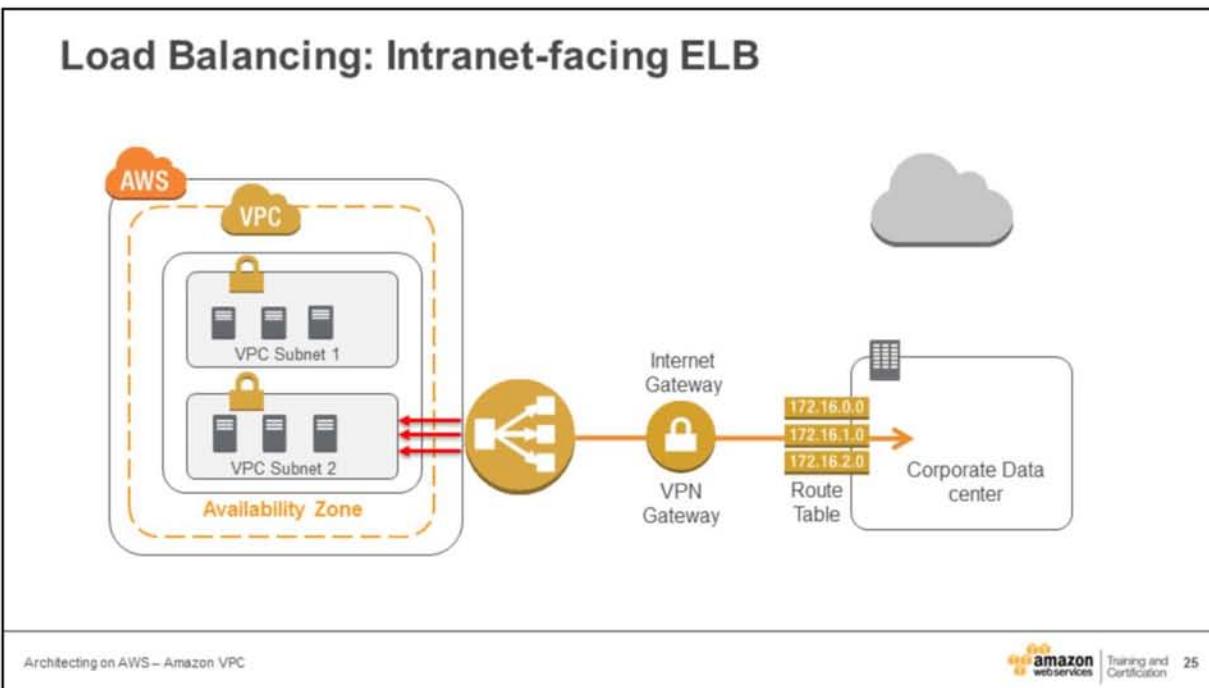
Another important feature of Amazon VPC is how it integrates with Amazon Elastic Load Balancing (ELB).

ELBs come in two varieties. First, external load balancing. In this case, the ELB provides an externally accessible endpoint that balances external traffic from the Internet across a farm of servers.



Notes:

Another option is internal load balancing. In this scenario, the ELB is an internally accessible endpoint that load balances traffic from internal networks or data centers.



Notes:

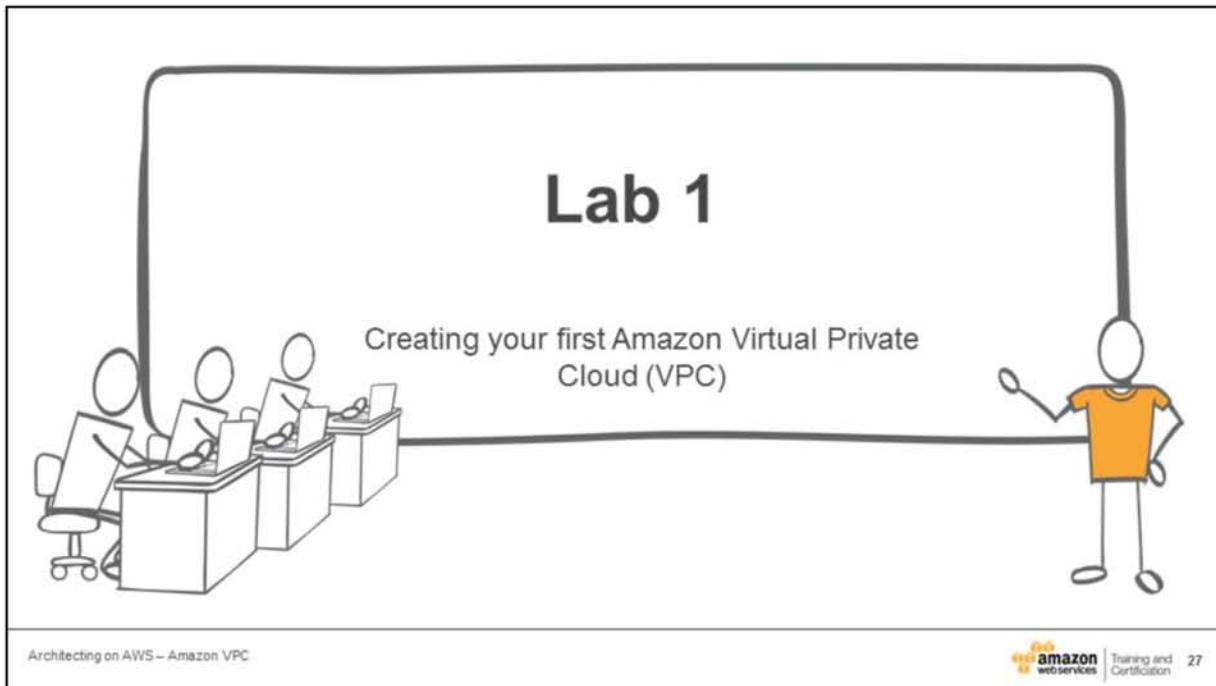
You can also use it as a mid-tier load balancer—balancing traffic from front-end web servers to back-end application servers.

Module review

- 💡 **True or False?** In VPC, you can assign multiple IP address and attach multiple ENIs and EIPs to EC2 instances
- 💡 **True or False?** Private subnet can span across multiple Availability Zone
- 💡 Describe how you can integrate ELB into your VPC

Answers:

- In VPC, you can assign multiple IP address and attach multiple ENIs and EIPs to EC2 instances → True
- Private subnet can span across multiple Availability Zone → False. Regardless of public or private subnet, a subnet cannot span across multiple AZs
- In this module, we discussed the usage of ELB facing external traffic, intermediate (middle-tier) load balancing, and intranet-facing ELB. If you are migrating an existing system to Amazon cloud, you can replace the load balancing solution (e.g. F5) with ELB.



Architecting on AWS – Amazon VPC

 amazon
web services | Training and
Certification 27

Notes:

Approximate timing: 1 hour 30 minutes

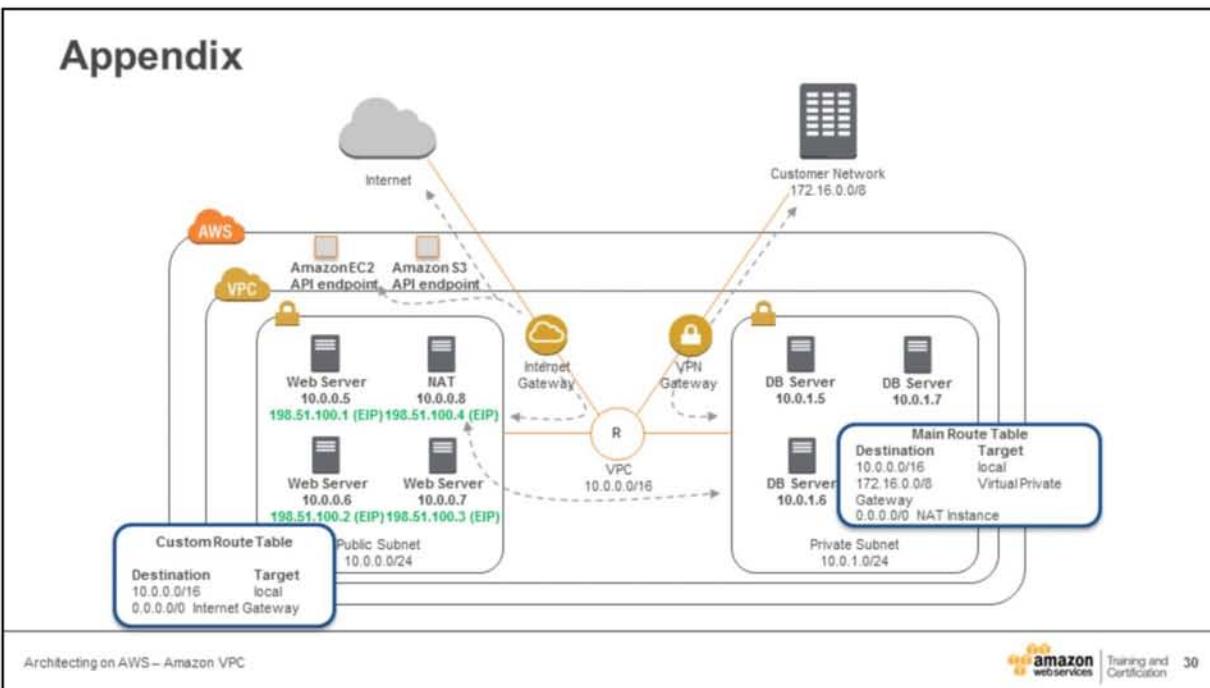
Copyright © 2013, 2014 Amazon Web Services, Inc. and its affiliates. All rights reserved.

This work may not be reproduced or redistributed, in whole or in part, without prior written permission from Amazon Web Services, Inc. Commercial copying, lending, or selling is prohibited.

Errors or corrections? Email us at aws-course-feedback@amazon.com.
Other questions? Email us at aws-training-info@amazon.com.

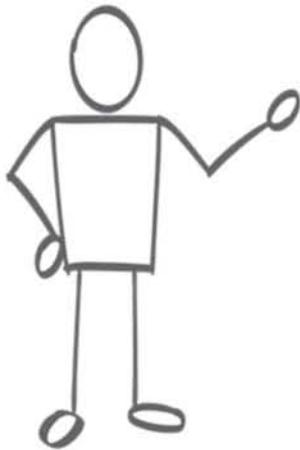
All trademarks are the property of their owners.

Appendix



This diagram uses a variety of Amazon services and configurations. We have a VPC with two subnets, an Internet Gateway, and on.

- How does traffic flow from one location to another?
- What traffic pathways are allowed or denied by this configuration?



Module 5: Identity, Authentication, and Authorization

Architecting on AWS – Identity, Authentication, and Authorization

 Amazon
Training and
Certification 1

Topics

- 💡 Authentication, authorization, and where they apply
- 💡 Authentication to AWS Service APIs
- 💡 Authorization Policies
- 💡 Temporary credentials with the Security Token Service
- 💡 Service-specific, OS, and application authentication

Notes:

Topics

- 💡 Authentication, authorization, and where they apply
- 💡 Authentication to AWS Service APIs
- 💡 Authorization Policies
- 💡 Temporary credentials with the Security Token Service
- 💡 Service-specific, OS, and application authentication

Notes:

First, let's talk about authentication and authorization, and where they apply.

Question: what's the difference between authentication and authorization?

- Authentication: you are who you say you are.
- Authorization: what you're allowed to do.

Authentication, authorization, and where they apply

- 💡 The three major realms where authentication and authorization occur within AWS
- 💡 Multi-factor authentication and how to implement it
- 💡 Your AWS master account
- 💡 Creating users and groups with IAM
- 💡 The role of authorization policies

Notes:

The three realms: WordPress example

💡 We want to run *WordPress* on AWS

1. Login to **Management console** and launch EC2 instance
→ Management level authentication and authentication
2. Login to **instance**, install WordPress and configure DB connection
→ Component level authentication and authentication
3. Login to **WordPress** and write a blog post
→ Application level authentication and authentication

Notes:

Let's use an example to illustrate different types, or *realms*, of authentication. We'll use WordPress here, but keep in mind this is just an example. Our first assumption is that we want to run WordPress on AWS. What would we need to do?

To start with, we need to log into the Management Console and launch an EC2 instance. Next, we need to log into the instance, install WordPress and configure a database connection to store blog posts and other data. After we install WordPress, we need to log in. Then we can write a blog post, create a page, and so on.

Login to Management console and launch EC2 instance

💡 Authentication and authorization to AWS APIs:

- Everything is an API at AWS
- You have to make authenticated API requests
 - Examples of API requests:
EC2 → RunInstance

Notes:

Normally, you'd think of logging in as accessing the URL of the Management Console and supplying your credentials—this isn't always the case. You can also use our APIs to do the same thing. Everything at AWS is an API, and to make an API request, you must be authenticated.

One common (and pretty straightforward) task is to use the API to launch an instance. For more information, go to

<http://docs.aws.amazon.com/cli/latest/userguide/cli-ec2-launch.html>.

Login to instance, install WordPress and configure DB connection

💡 Authentication and authorization to **OS**:

- Local Linux user (for example: `root@`, `ubuntu@`, `ec2-user@`)
- Local Windows user (Administrator)

💡 Authentication and authorization to **database**:

- MySQL username and password
- SQL Server username and password

Notes:

With an instance up and running, you need to log into it and install WordPress.

Of course, how you log in depends on whether you launch a Linux or Windows EC2 instance. You also need credentials to access the WordPress database.

Login to WordPress and write a blog post

💡 Authentication and authorization to the application:

- WordPress authenticates to a database
- Some applications authenticate to Active Directory
- Others authenticate via OAuth 2.0, and so on

Notes:

Lastly, you need to have credentials to access the app itself.

WordPress Example

Task	Can AWS help?
Login to Management console and launch EC2 Instance	Yes, a lot
Login to instance, install WordPress and configure DB connection	Yes, some
Login to WordPress and write a blog post	Depends on the application

Notes:

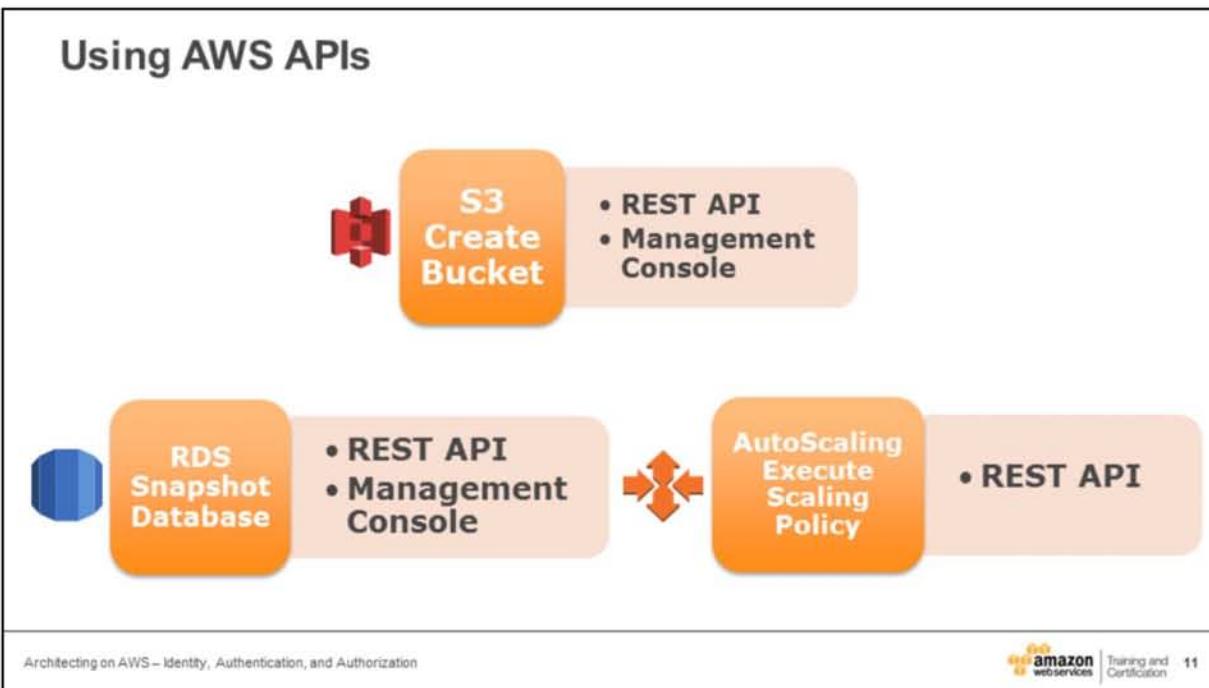
How does AWS fit into these three different realms of authentication and authorization? As you can see here, it varies. Naturally, AWS has a lot of tools to help you access AWS-specific resources. We can help a bit with logging into instance and bootstrapping them to install and configure apps and services. After that, it really depends on the application.

Topics

- 💡 Authentication, authorization, and where they apply
- 💡 Authentication to AWS Service APIs
- 💡 Authorization Policies
- 💡 Temporary credentials with the Security Token Service
- 💡 Service-specific, OS, and application authentication

Notes:

Let's look at how you can authenticate to AWS service APIs.



Notes:

For more detail, please refer to online API reference documentation.

- S3 → <http://docs.aws.amazon.com/AmazonS3/latest/API/IntroductionAPI.html>
- RDS → <http://docs.aws.amazon.com/AmazonRDS/latest/APIReference/Welcome.html>
- Auto Scaling → <http://docs.aws.amazon.com/AutoScaling/latest/APIReference/Welcome.html>

Using API example:

- Using an API to create an Amazon S3 bucket. You can accomplish this using a REST API or the Management Console. (Note that the SOAP support over HTTP is deprecated, but it is still available over HTTPS. New Amazon S3 features will not be supported for SOAP, so the recommendation is to use either the REST API or the AWS SDKs.)
- Creating a snapshot of a database in Amazon RDS. There are two options: REST API or Management Console.
- Auto Scaling policy. Currently one API option: REST API. (Don't forget there's still our really robust Command Line Interface!)

Three major interfaces to AWS

	For all services?	Credential
REST API	YES	Access key Secret Key
Management Console	NO	Username Password
SOAP API	NO	X.509 Certificate

Notes:

Here, you can see the three different sets of credentials used for the different AWS interfaces.

Multi-factor Authentication (MFA):

Optional but recommended

Physical



Virtual

Android

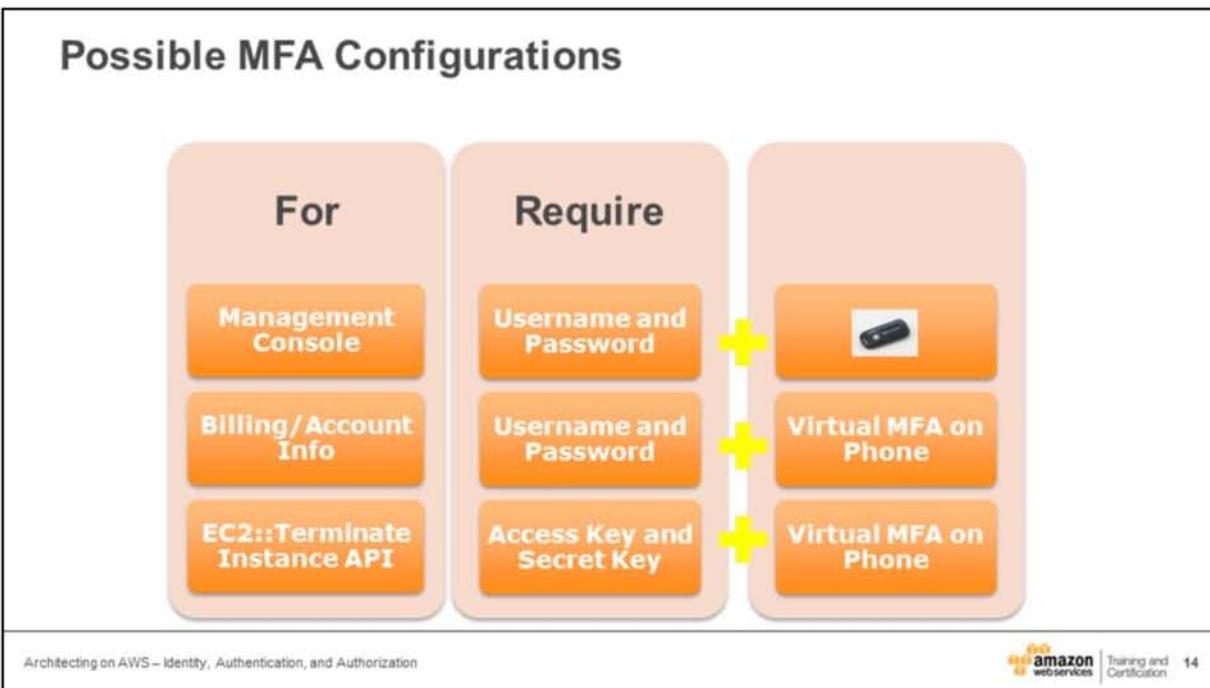
iOS

Windows Phone

Blackberry

Notes:

Let's spend some time talking about multi-factor authentication (MFA). You're not required to use MFA, but we strongly recommend doing so.



Notes:

Here are a few possible configurations of MFA that you might implement. These are just suggestions—confer with your security teams to find a system that works best for your organization.

Master Account

- 💡 Every account has a master user
 - Equivalent to Root/Administrator
- 💡 Every master user has:
 - A Management Console login
 - An Access Key/Secret Key
- 💡 Best practices:
 - Do not use the access key/secret key from the Master Account
 - Apply a physical MFA to the Management Console login
 - Use Identity and Access Management

Notes:

Your AWS master account is one of the most important assets as you incorporate the Cloud into your infrastructure.

Every master account has both a Management Console login and an access key/secret key.

Here's where it gets interesting: while you *get* an access key/secret key with your Master Account, we don't recommend you use it. Instead, we recommend you use Amazon Identity and Access Management to create users and groups. Add a physical MFA to your master account, and then use that account only when absolutely necessary.

Identity and Access Management (IAM)

💡 Within a Master Account, create:

- Users
- Groups
- Roles



Create users within a master account

- 💡 No credentials or privilege by default
- 💡 Credentials can be any of:
 - Console login
 - Access key/secret key
 - MFA
 - X.509 cert
- 💡 Privilege via:
 - Individual authorization
 - Group membership

Best Practices:

- Rotate access key/secret key
- Apply a password policy

Notes:

With IAM, you can use your master account to create users (with no credentials or privileges by default). These user accounts can use one of several different credential options. You can grant user privileges through individual authorization, or through group membership.

Create groups within a master account

Groups

- A collection of users 
- Defines privilege of members via authorization policies

Notes:

Groups should be a familiar concept to most of you.

Create roles within a master account

💡 Roles

- Allow your applications (such as Java) running on EC2 to securely access other services (S3, SQS, and so on)
- Allow cross-account management/access

Example:

- Jane in Account A may assume a Role in Account B, giving Jane an Access Key/Secret Key TokenName that may be used to make API calls to Account B.

Notes:

Roles provide another way of controlling access to AWS resources.

Accessing DynamoDB from Java application on EC2

- 💡 An example of EC2 not using IAM role

```
AWSCredentials creds = new BasicAWSCredentials(  
    "AKIAIOSFODNN7EXAMPLE",  
    "wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY");  
CredentialProvider session = new STSSessionCredentialsProvider(creds);  
AmazonDynamoDB dynamo = new AmazonDynamoDBClient(session);
```

Credentials embedded
in code!!!

Notes:

In this example, we have a Java App running on an EC2 instance and accessing DynamoDB. One option might be to embed credentials in your code. Needless to say, this is not ideal.

Accessing DynamoDB from Java application on EC2

- EC2 using an IAM role

```
AmazonDynamoDB dynamo = new AmazonDynamoDBClient();
```

Credentials automatically
retrieved from IAM Role

Notes:

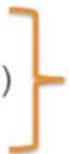
Instead, we can actually retrieve credentials automatically. From here, a user can take advantage of a role to access resources they might not normally need to access.

Topics

- 💡 Authentication, authorization, and where they apply
- 💡 Authentication to AWS Service APIs
- 💡 Authorization Policies**
- 💡 Temporary credentials with the Security Token Service
- 💡 Service-specific, OS, and application authentication

Authorization Policies

- 💡 Defining fine-grain privileges for IAM Users, Groups, and Roles
- 💡 Authorization Policy Documents
 - JSON format
 - Action (API)
 - Resource (some services)
 - Condition (optional)



Define **least-privilege access** for each user, group, or role in your AWS account

Notes:

Policies give you the opportunity to fine tune privileges granted to IAM users, groups, and roles. Policies can be in a variety of formats. Since policies are in JSON format, they can be used in conjunction with a version control system. It's a good idea to define least-privilege access to each user, group, or role. Then, you can customize access to specific resources using an Authorization Policy.

Authorization Policy Documents (1 of 2)

JSON format

Action (API)

➤ Specific API(s) that you can call, such as:

- S3::GetObject
- S3::GetObjectVersion
- **S3::Get***

Resource (some services)

➤ Applies to specific resources, such as:

- arn:aws:s3:::bucketname/keyname
- arn:aws:s3:::my_website/images/header.jpg
- arn:aws:s3:::my_website/images/*

Notes:

Here are a few examples of how you can scope an authorization policy to a set of specific API calls. Alternatively, you can create a policy that applies to specific AWS resources.

Authorization Policy Documents (2 of 2)

💡 Condition (optional)

- Applies to specific conditions, such as:
 - SSL required
 - Request must originate from specific IP range (CIDR)
 - Request requires MFA
 - Request valid until (or after) some date/time

Notes:

You can also create policies that apply to specific conditions, such as an IP address range, or time of day.

Authorization Policies Example

```
{ "Statement" : [  
    {  
        "Effect" : "Allow",  
        "Action" : "s3:Get*",  
        "Resource" : "arn:aws:s3:::my-bucket/secure/*",  
        "Condition" : {  
            "IpAddress" : {  
                "aws:SourceIp" : [ "174.128.53.0/24" ]  
            }  
        },  
        { ANOTHER STATEMENT... } ] }
```

Notes:

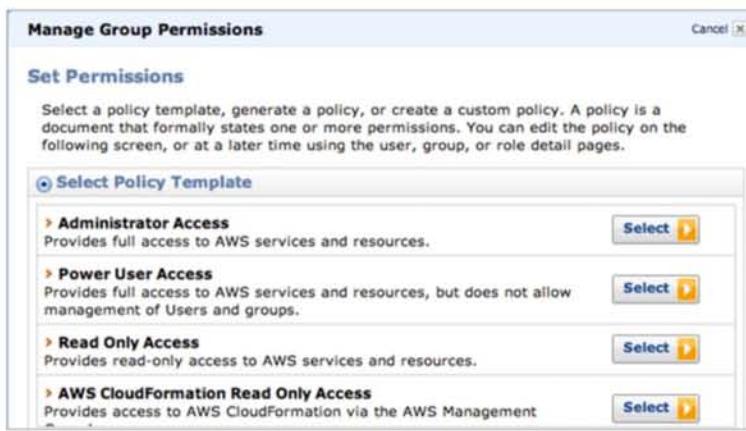
Here's an example of an authorization policy.

Question: what does this policy allow?

- Specific API call (S3:Get*)
- Specific resource (bucket)
- Request must come from a specific IP address range.

Creating a policy document: Use pre-defined policies

💡 In IAM Management console at console.aws.amazon.com/iam



Architecting on AWS – Identity, Authentication, and Authorization

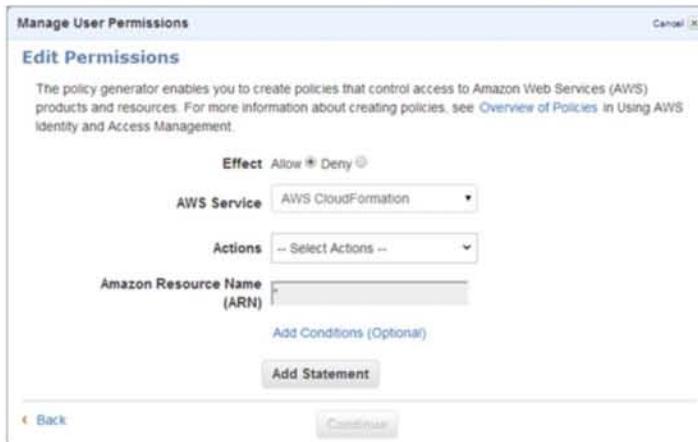
 Training and Certification 27

Notes:

The Management Console includes several pre-defined policies that you can use.

Creating a policy document: Use Policy Generator

In IAM Management console at console.aws.amazon.com/iam



Architecting on AWS – Identity, Authentication, and Authorization

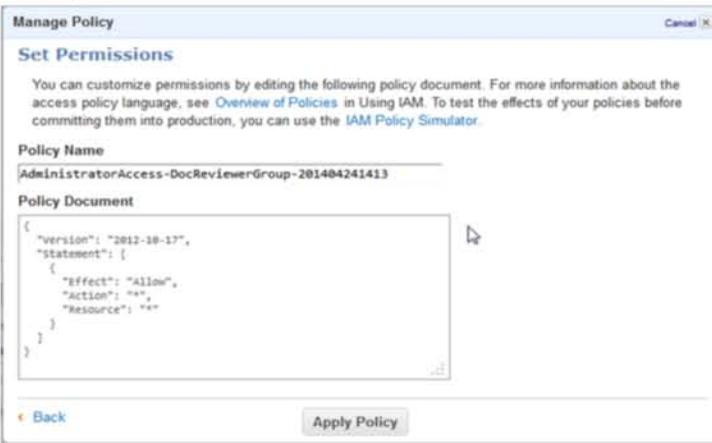
 Training and Certification 28

Notes:

In addition, you can create a policy through the Policy Generator in the Management Console.

Creating a policy document: Define custom policies

💡 In IAM Management Console or APIs



Architecting on AWS – Identity, Authentication, and Authorization

Amazon
Training and
Certification 29

Notes:

You can even use the Policy Generator to create custom policies.

Creating a policy document: Simulate policies

💡 Use policy simulator in IAM management console

The screenshot shows the AWS IAM Policy Simulator interface. On the left, there is a preview of a JSON policy document titled "AdministratorAccess-DocReviewerGroup-201404241413". The policy contains a single statement allowing all actions on all resources. On the right, the "Policy Simulator" interface is displayed with various configuration options like "Select service", "Select actions", and "Run Simulation". The "Simulation Settings" section shows the selected service as "Amazon EC2" and the resource name format as "arn:aws:ec2:<region>:<account>:<resource>". Below this, a note states "No condition keys appear in the selected policies." At the bottom, the "Results" section indicates "0 actions selected, 0 actions not simulated, 0 actions allowed, 0 actions denied." A table header for "Service", "Action", "Permission", and "Description" is shown but no data is present. The footer of the screenshot includes the text "Architecting on AWS – Identity, Authentication, and Authorization" and the Amazon logo with "Training and Certification 30".

Notes:

The IAM policy simulator enables you to test the effects of IAM access control policies before committing them into production, making it easier to verify and troubleshoot permissions. Simply choose the policy you want to evaluate, select from the list of AWS actions, and click a button to simulate whether the policy will allow or deny the selected actions.

User guide →

<http://docs.aws.amazon.com/IAM/latest/UsingPolicySimulatorGuide/using-iam-policy-simulator.html>

Topics

- 💡 Authentication, authorization, and where they apply
- 💡 Authentication to AWS Service APIs
- 💡 Authorization Policies
- 💡 Temporary credentials with the Security Token Service
- 💡 Service-specific, OS, and application authentication

Notes:

In addition to the authentication and authorization measures we've talked about, you can also grant temporary credentials using the Security Token Service.

Security Token Service

- 💡 Generate temporary credentials for an IAM User or for users that you authenticate (federated users).
- 💡 Useful for improving security posture, mobile applications, and identity federation.

Temporary Credentials

- Access Key, Secret Key, and Token
- Expire automatically (15 minutes ~ 36 hours)

IAM Users

- Can create temporary credentials for themselves

Federated Users

- Authenticate users to your identity store
- SSO to AWS Management Console
- Enhanced security for mobile applications

Roles

- Allow trusted entity to assume role
- Entity = EC2 Instance(s), or an IAM user in another account

Notes:

This service generates temporary credentials that you authenticate. This is very useful when dealing with federated users.

Let's look at the different types of credentials that you can use in your security processes.

1. **Temporary credentials** in some cases may be valid for up to 36 hours. In other cases (when using AssumeRole), they're only valid up to 1 hour.
2. **IAM users** can create temporary credentials for themselves.
3. **Federated users** first go through your own authentication process (ActiveDirectory, and so on). Can use Single Sign-on to access the management console.
4. You can allow trusted entities, such as an EC2 instance or an IAM user in another account, to assume a **role**.

IAM users can use the AWS Security Token Service GetSessionToken API action to create temporary security credentials for themselves. This enables access for IAM users or AWS accounts whose permissions are already defined. Because the credentials are temporary, they provide enhanced security when you have an IAM user who will be accessing your resources through a less secure environment, such as a mobile device or web browser.

By default, temporary security credentials for an IAM user are valid for a maximum

of 12 hours, but you can request a duration as short as 15 minutes, or as long as 36 hours. For security reasons, a token for an AWS account's root identity is restricted to a duration of one hour.

Federated Users

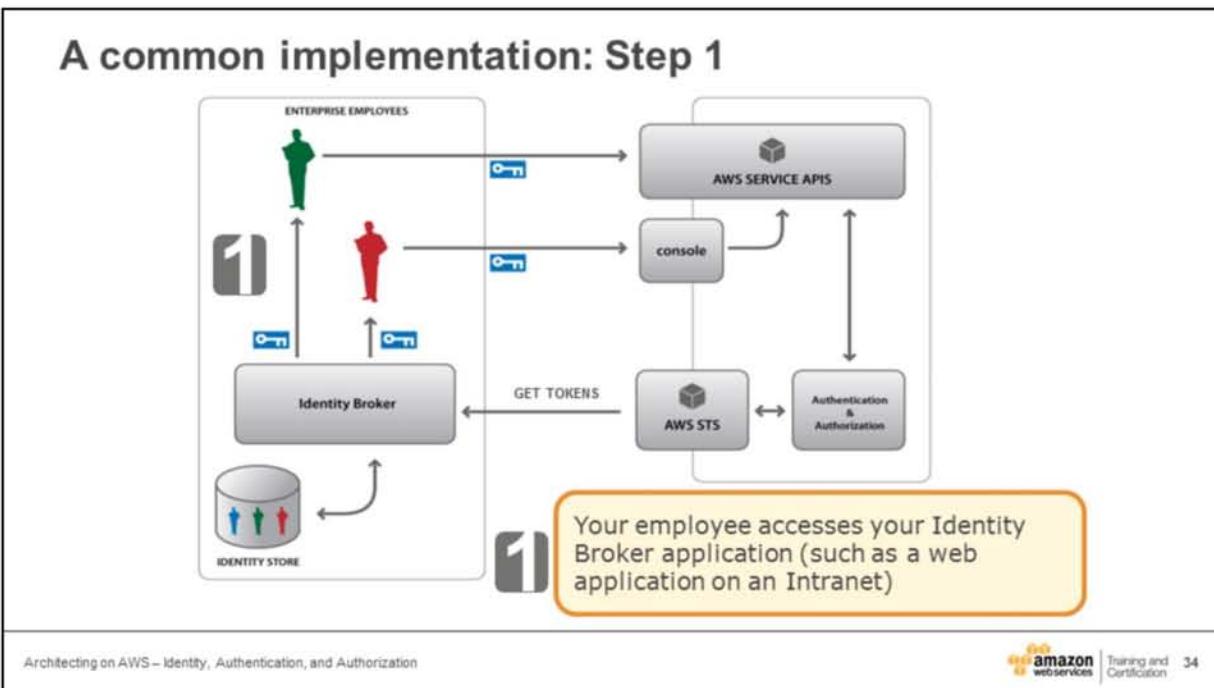
Federated Users

- **Authenticate users to your own identity store**

- You write an "identity broker application"
- Users authenticate to your identity broker
- Your identity broker provisions temporary credentials via STS
- **SSO:** Temporary credentials can be used to sign user directly into the AWS Management Console
- **Let's look at an example...**

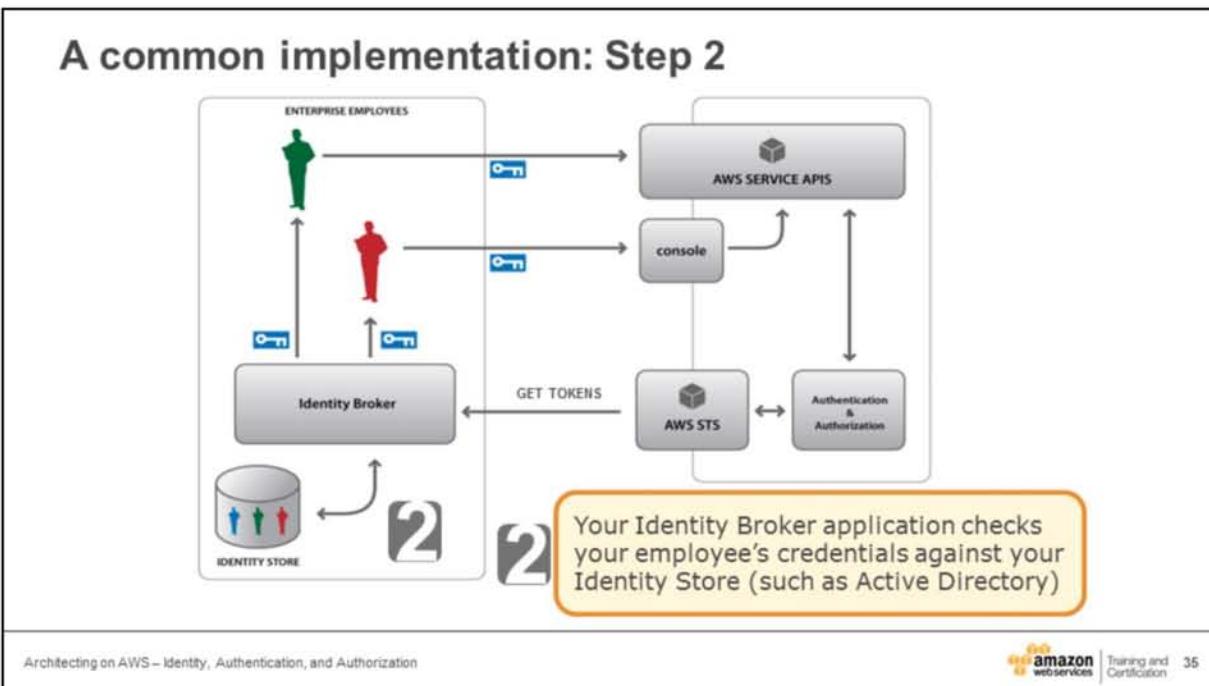
Notes:

Linking federated users with AWS is pretty common. The next few slides will walk through a typical implementation.



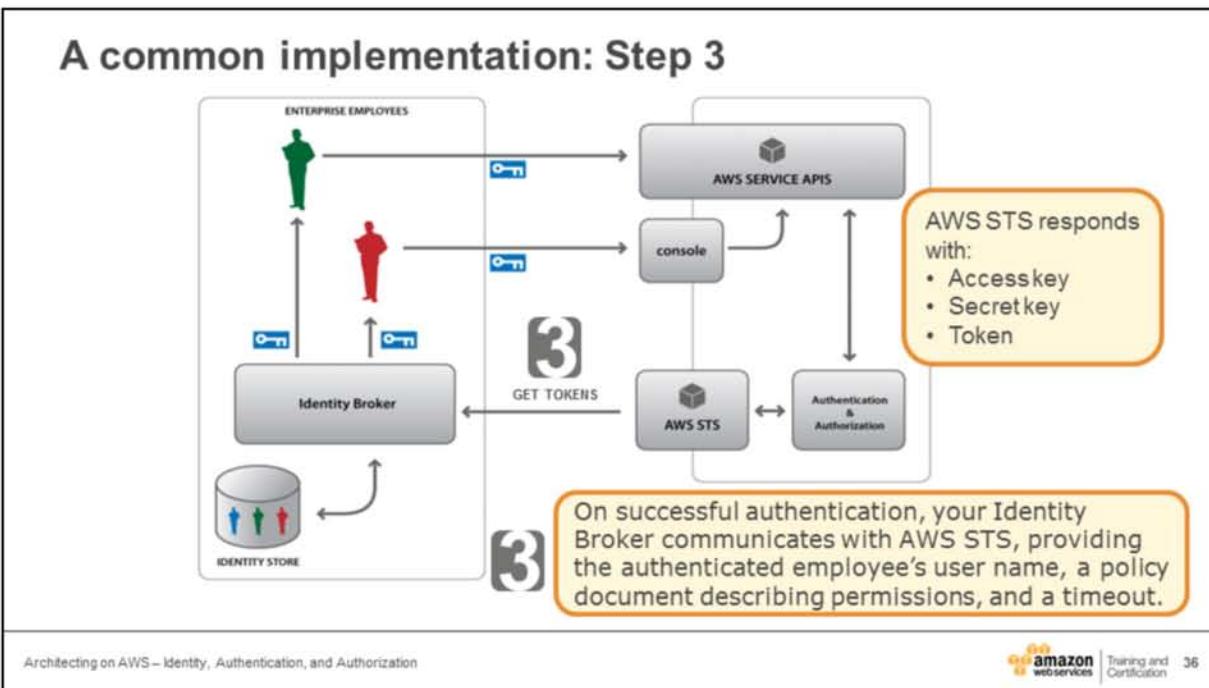
Notes:

Step 1: Your employee access your Identity Broker application.



Notes:

Step 2: Your Identity Broker app checks the user's credentials.

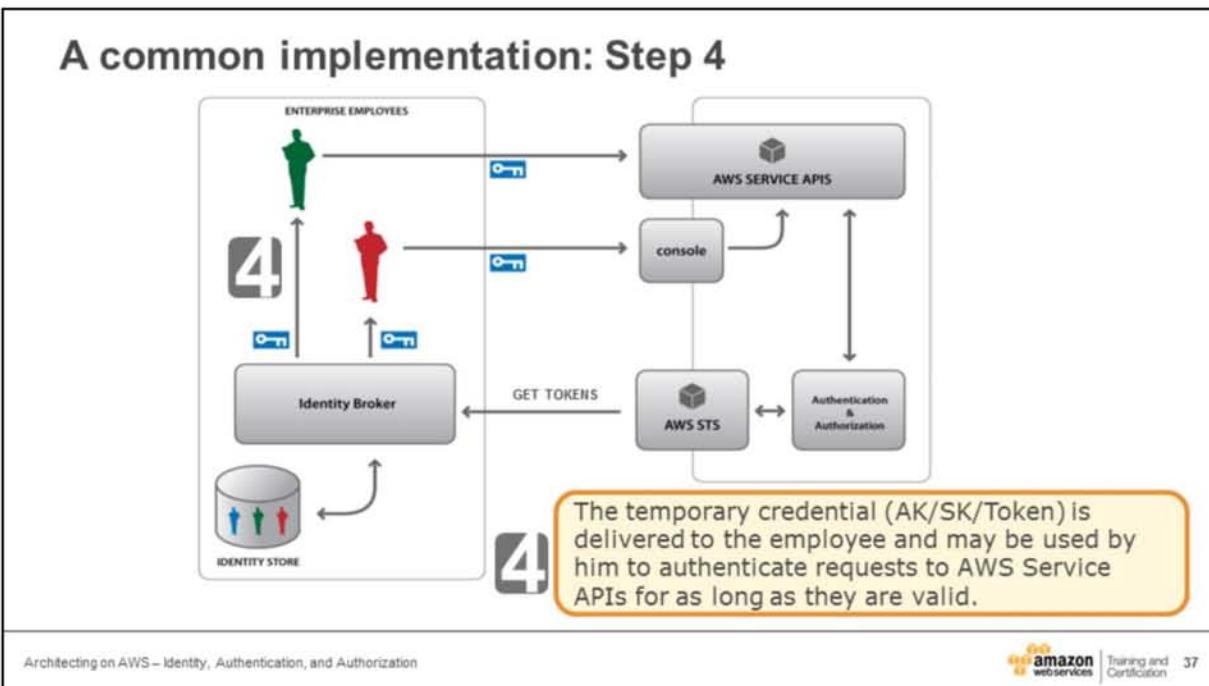


Notes:

Step 3: After successful authentication, the Identity Broker communicates with AWS STS, providing:

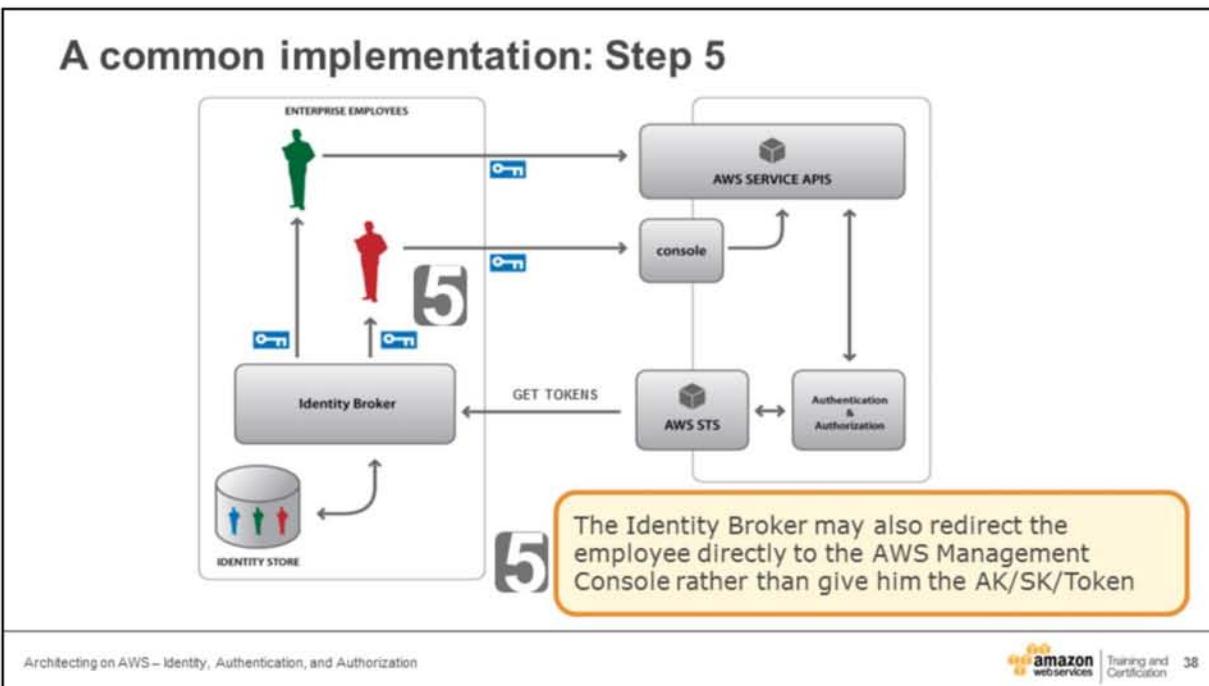
- Employee name
- Policy document
- Timeout

AWS STS responds with an Access Key, Secret Key, and Token. This credential has the permissions and expiration specified by the Identity Broker



Notes:

Step 4: User receives token and uses it authenticate requests.



Notes:

Step 5: User receives token and uses it authenticate requests.

Topics

- 💡 Authentication, authorization, and where they apply
- 💡 Authentication to AWS Service APIs
- 💡 Authorization Policies
- 💡 Temporary credentials with the Security Token Service
- 💡 Service-specific, OS, and application authentication

Service-specific policy documents



S3

Some services have additional mechanisms for authentication and authorization.



SNS

You may apply authorization policy documents to these individual services



SQS

For example, a policy applied to an S3 bucket may make some objects publically readable

Notes:

Some (but not all) services allow you to use additional authentication and authorization mechanisms.

S3 Bucket Policy Example

```
{  
    "Statement": [  
        {  
            "Sid": "AddCannedAcl",  
            "Effect": "Allow",  
            "Principal": {  
                "AWS": [  
                    "arn:aws:iam::111122223333:root",  
                    "arn:aws:iam::444455556666:root"  
                ]  
            },  
            "Action": ["s3:PutObject", "s3:PutObjectAcl"],  
            "Resource": ["arn:aws:s3:::bucket/*"],  
            "Condition": {  
                "StringEquals": {  
                    "s3:x-amz-acl": ["public-read"]  
                }  
            }  
        }  
    ]  
}
```

Allow... these two AWS accounts (principals) to make...
...these two S3 API calls
...anywhere in this bucket
...as long as the object they are trying to put includes the 'public-read' ACL.

Architecting on AWS – Identity, Authentication, and Authorization

amazon web services | Training and Certification 41

Notes:

The next five slides show how a bucket policy allows certain accounts (note that they're referred to as "principals") to make specific API requests to a specific S3 bucket.

Operating System Authentication

- 💡 Initial OS login restricted by key pair
 - You maintain the private key (.pem file)
 - Has nothing to do with IAM
 - After initial login, you can implement your own authentication system (Active Directory, and so on)



Notes:

OS Authentication is also an important method of authentication. Remember that OS Authentication is not related to IAM.

RDS Database Authentication

- 💡 RDS database has its own username/password
 - Manage username/password with AWS Service APIs
 - Connect to database with username/password using normal conventions (such as JDBC)
 - **Has nothing to do with IAM**

Notes:

RDS Database authentication is the same as OS Authentication. Again, it has nothing to do with IAM.

Application Authentication

💡 Application authentication up to the user

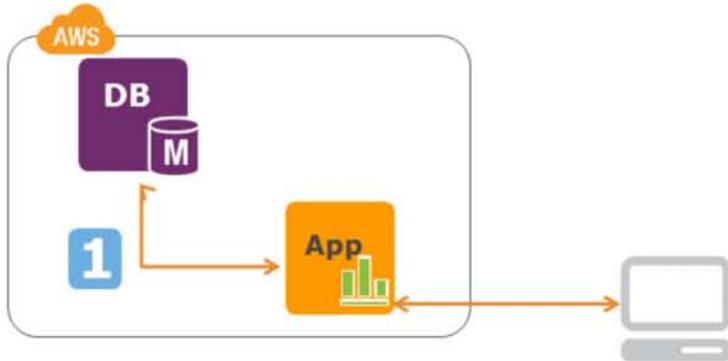
- IAM is strictly for authenticating/authorizing AWS Service APIs.
IAM is not suitable for application authentication.
- Let us look at 3 possibilities for application authentication in EC2...

Notes:

Some people ask if IAM is appropriate for application authentication. It's not.
Let's look at 3 ways to implement application authentication on EC2.

Application authentication up to the user

1. Use a local database for credentials and application roles

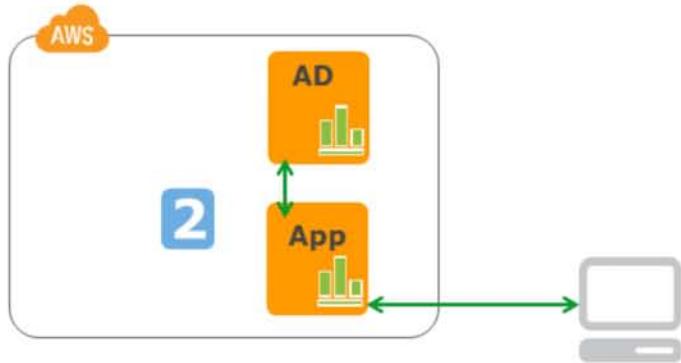


Notes:

One common solution: use a local database for credentials.

Application authentication up to the user

2. Use Active Directory or LDAP to store application credentials and roles

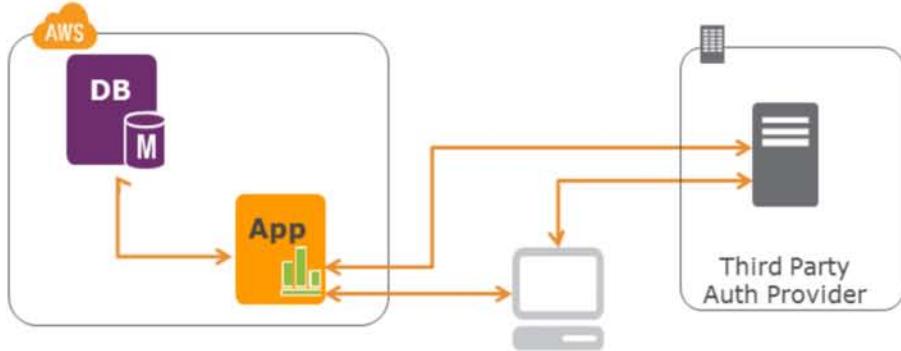


Notes:

Another option: use Active Directory or LDAP.

Application authentication up to the user

3. Integrate the application with a third-party provider (such as OAuth 2.0) and store roles in a local database



Notes:

Yet another option: connect the application to a third-party provider.

Module review

- 💡 What are the 3 major realms where authentication and authorization occur within AWS?
- 💡 What credentials are required to use an AWS API? The Management Console?
- 💡 What is the role of a policy document? How can you create one?
- 💡 What service allows federated access to AWS?
- 💡 Can you use IAM for application authentication?

Answers:

Three major realms

1. Management
2. Component
3. Application

API Credentials

Access key/secret key

Management Console

Username/password

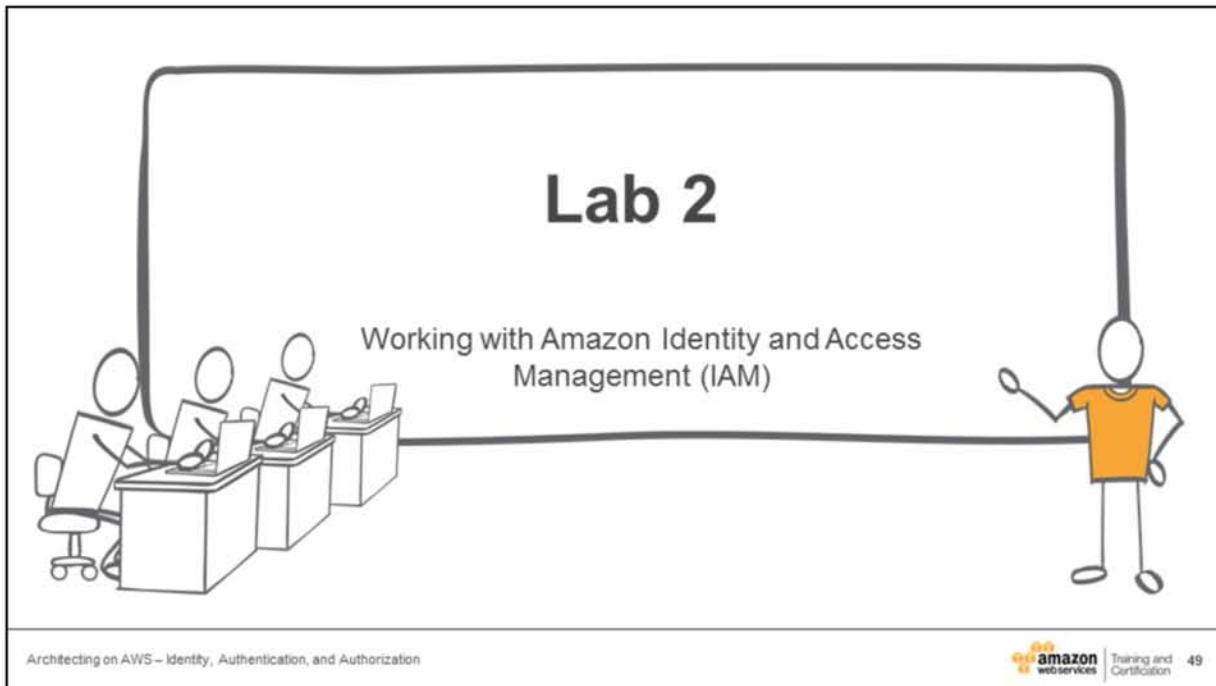
Policy document

Allows granular access to specific AWS resources or services. Create using management console: use policy generator, pre-defined, or build your own custom policy.

Federated access

Amazon Security Token Service

IAM for app authentication? → No.



Architecting on AWS – Identity, Authentication, and Authorization

 Amazon
Training and
Certification 49

Notes:

Approximate timing: 1 hour

Copyright © 2013, 2014 Amazon Web Services, Inc. and its affiliates. All rights reserved.

This work may not be reproduced or redistributed, in whole or in part, without prior written permission from Amazon Web Services, Inc. Commercial copying, lending, or selling is prohibited.

Errors or corrections? Email us at aws-course-feedback@amazon.com.
Other questions? Email us at aws-training-info@amazon.com.

All trademarks are the property of their owners.

Appendix

Service-specific, OS, and application authentication

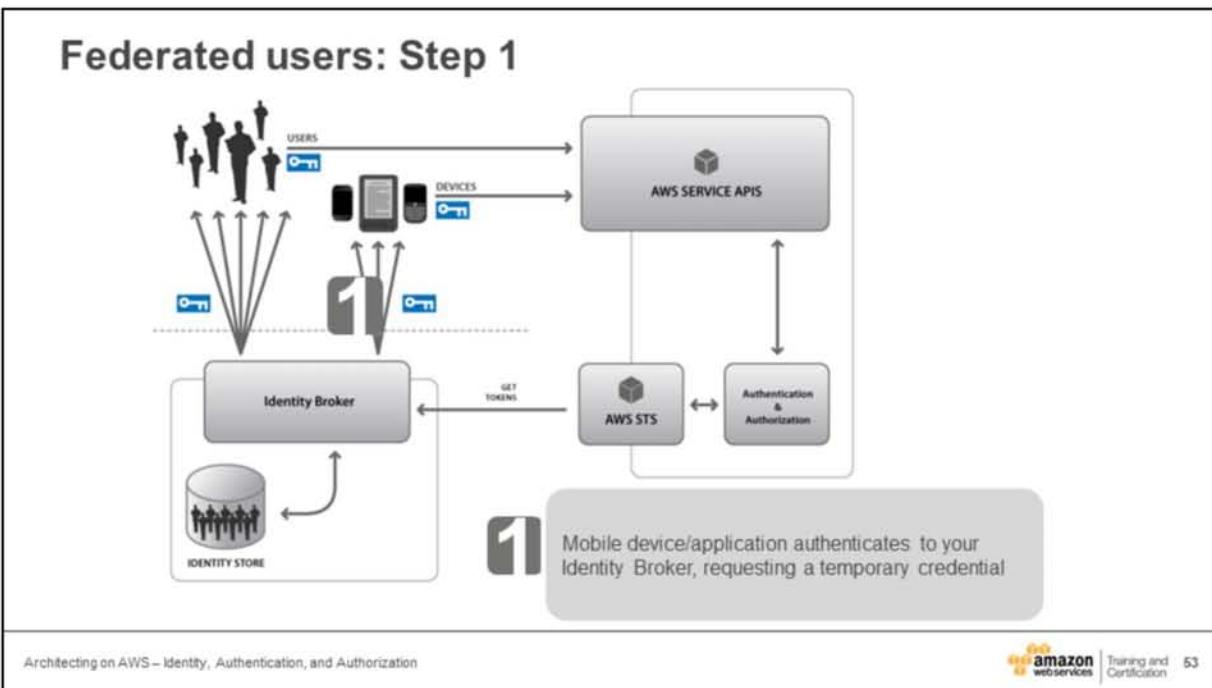
Federated users (1 of 7)

- 💡 Allow mobile devices to communicate directly with AWS
 - **Example:** allow a mobile application to upload photos or video directly to S3
 - No limit to number of credentials that you can generate
 - **Expire automatically**
 - **Let's look at an example**

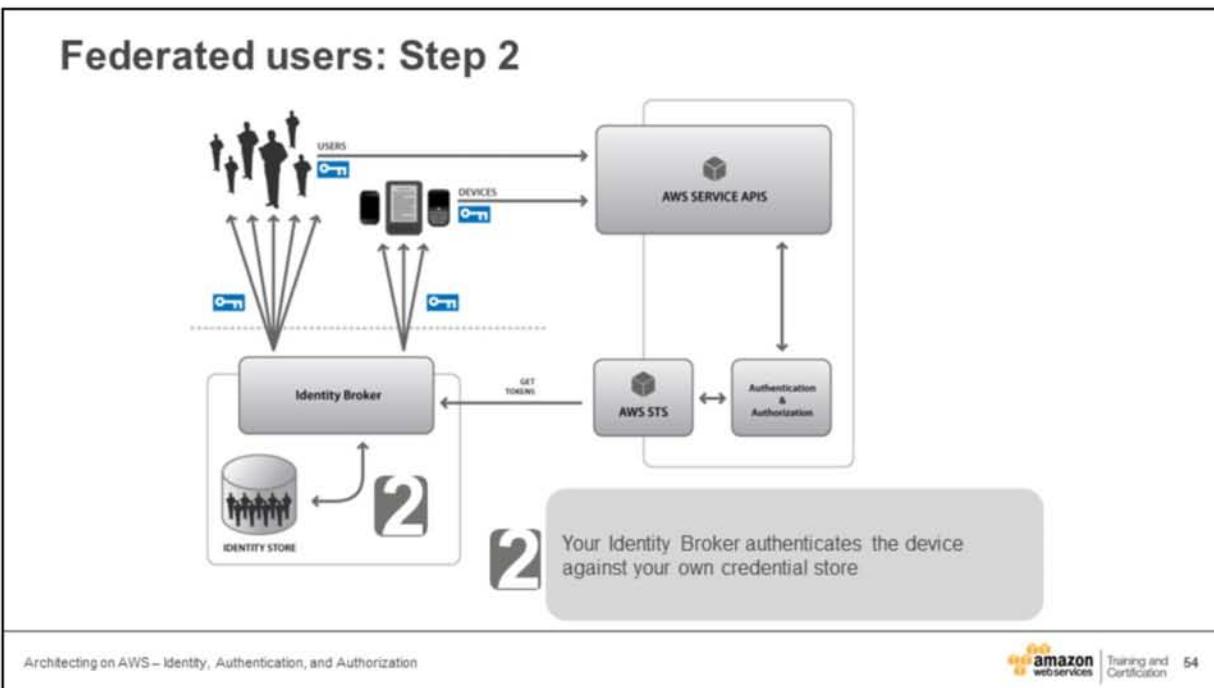
[Course Title - Module Title]



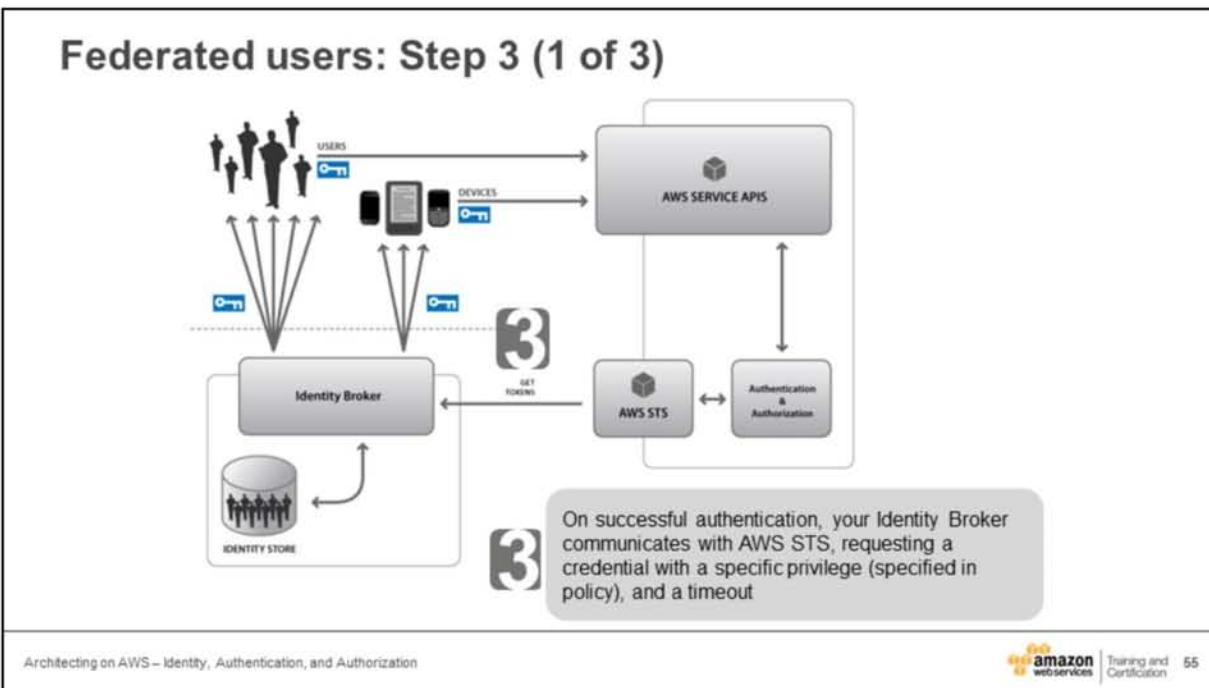
Training and
Certification 52



Step 1: Mobile device authenticates to your Identity Broker.



Step 2: Identity Broker authenticates the device.



After successful authentication, the Identity Broker communicates with AWS STS:

- Requests specific privilege
- Timeout

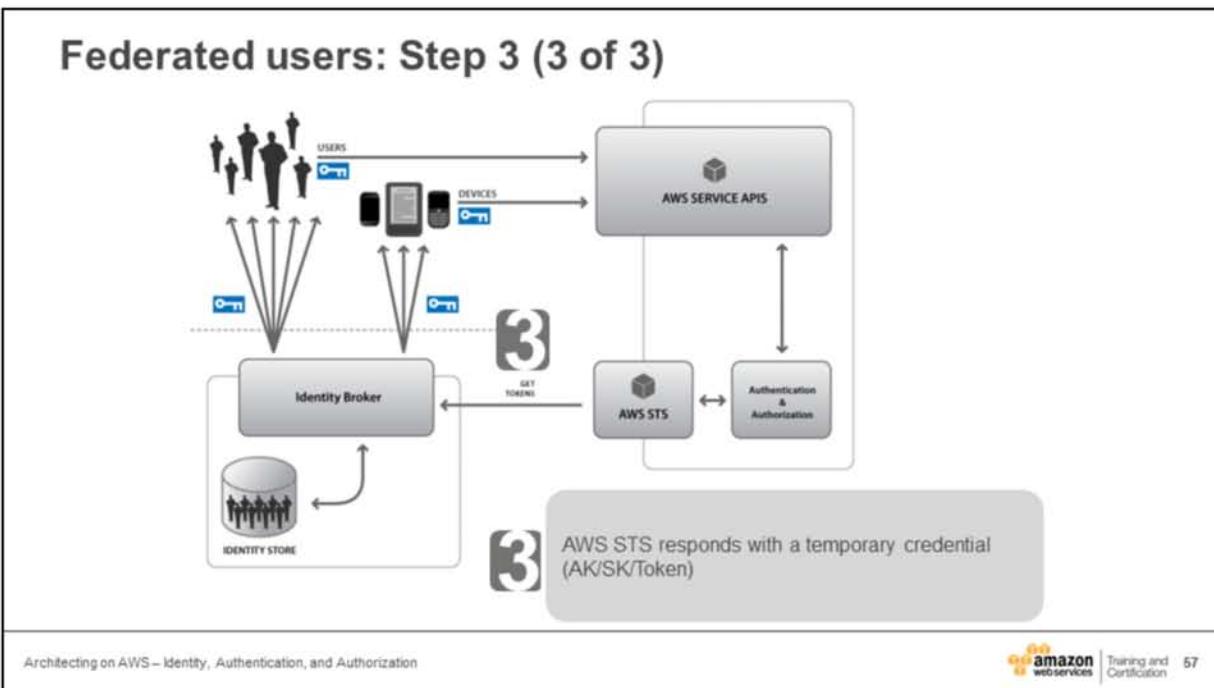
Federated users: Step 3 (2 of 3)



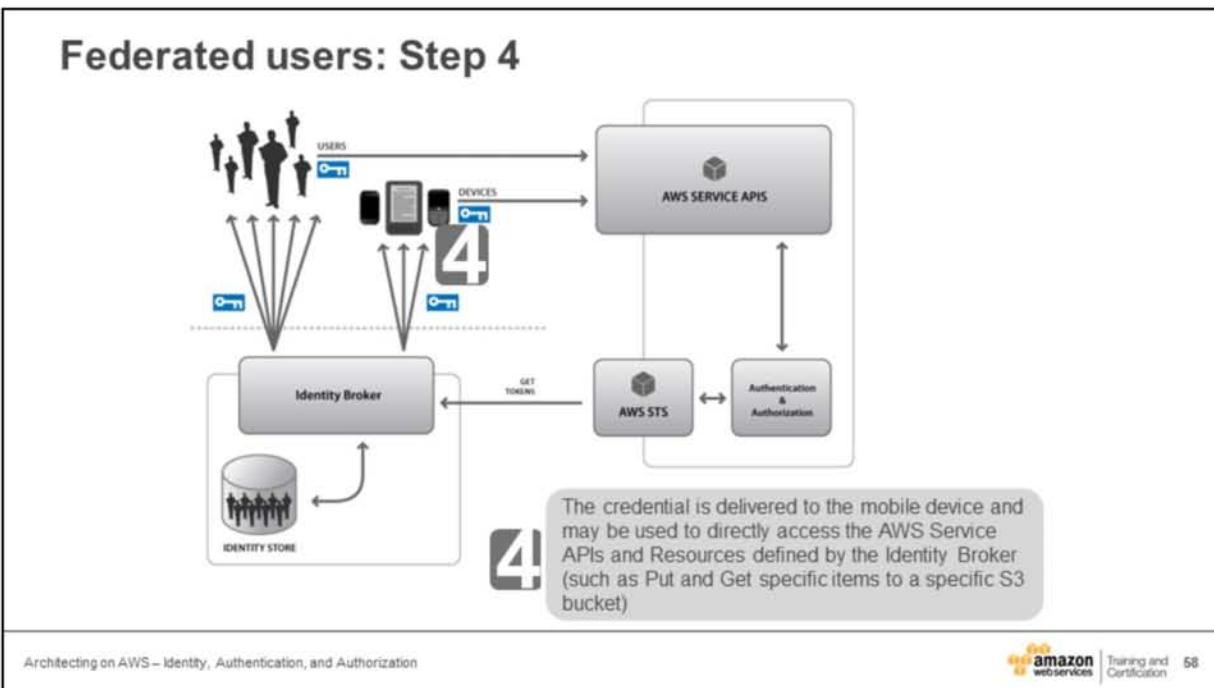
Architecting on AWS – Identity, Authentication, and Authorization

 Amazon
Training and
Certification 56

An example of a privilege, as specified by a policy document



AWS STS responds with temporary credential



Step 4: Mobile device receives credential.



Module 6: Overview of Services for Web Applications

Architecting on AWS – Overview of Services for Web Applications

 Amazon
Training and
Certification 1

Topics

- 💡 AWS products for network content and delivery
 - Amazon Route 53
 - Amazon Elastic Load Balancer
 - Amazon CloudFront
- 💡 AWS products for deployment and management
 - Amazon CloudWatch
 - Amazon Elastic Beanstalk
 - AWS CloudFormation

Notes:

Topics

- 💡 AWS products for network content and delivery
 - Amazon Route 53
 - Amazon Elastic Load Balancer
 - Amazon CloudFront
- 💡 AWS products for deployment and management
 - Amazon CloudWatch
 - Amazon Elastic Beanstalk
 - AWS CloudFormation

Notes:

Network and Content Delivery

- Amazon Route 53 

- Amazon Elastic Load Balancer 

- Amazon CloudFront 

Architecting on AWS – Overview of Services for Web Applications

 Training and Certification 4

Notes:

Specifically, we'll look at Amazon Route 53, Amazon Elastic Load Balancer, and Amazon CloudFront.

Amazon Route 53

- Global network of DNS servers that answer DNS queries with low latency
- Queries for your domain are **automatically routed to the nearest DNS server**, and thus answered with the best possible performance
- You pay only for managing domains through the service and the number of queries that the service answers

Notes:

Amazon Route 53 is a global network of DNS servers. It answers DNS queries with low latency.

Route 53 is an “authoritative DNS” system. An authoritative DNS system provides an update mechanism that developers use to manage their public DNS names. It then answers DNS queries, translating domain names into IP address so computers can communicate with each other. The name for our service (Route 53) comes from the fact that DNS servers respond to queries on port 53 and provide answers that route end users to your applications on the Internet. In the future, we will add additional routing capabilities to Route 53 to better help your users find the best way to your website or application.

Route 53 example: amdocstore.com



Name	Type	Value
amdocstore.com	A	215.92.110.63
www.amdocstore.com	CNAME	amdocstore.com
assets.amdocstore.com	CNAME	amdocstore-assets.s3.amazonaws.com

Architecting on AWS – Overview of Services for Web Applications

 Amazon
Training and
Certification

6

Notes:

In this diagram, shows a typical configuration of Amazon Route 53. Notice that the Web Server resides on the West Coast of the United States, while the assets for the web site reside on the East Coast.

Route 53 example: Round Robin



Resolve to different values for the same record with equal probability

Name	Type	Value
amdocstore.com	A	215.92.110.63
amdocstore.com	A	215.92.110.64

Notes:

In this diagram, we have two web servers running. With Amazon Route 53, requests for a web application are split evenly between the two servers.

Route 53 example: Weighted Round Robin



Resolve to different values for the same record with different, user-controlled probabilities

Name	Type	Value	Weight
amdocstore.com	A	215.92.110.63	4
amdocstore.com	A	215.92.110.64	1

Architecting on AWS – Overview of Services for Web Applications

Amazon
Training and
Certification 8

Notes:

As you can see, one web server is larger than the other. This represents the fact that Amazon Route 53 is configured to send 4 times as much traffic to that server than to the smaller one.

Route 53 example: ALIAS record



Resolve zone apex (such as amdocstore.com)
to an Elastic Load Balancer

Name	Type	Value
amdocstore.com	ALIAS	some-elb-name.us-west-2.elb.amazonaws.com
www.amdocstore.com	CNAME	amdocstore.com

Architecting on AWS – Overview of Services for Web Applications

Amazon web services | Training and Certification 9

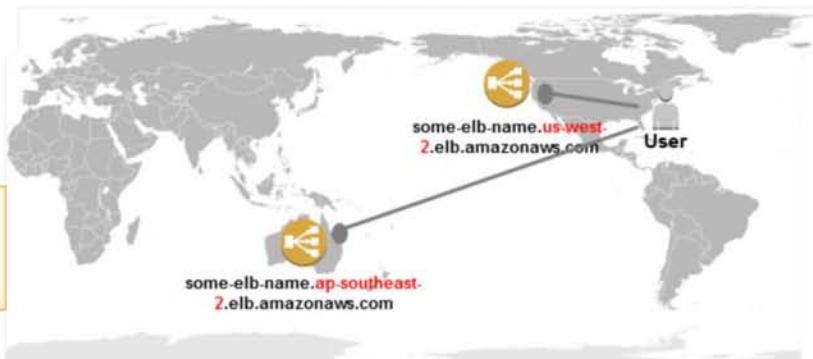
Notes:

Here's an example of Amazon Route 53 configured to send traffic to an Amazon Elastic Load Balancer. Amazon Route 53 also works well with Amazon Elastic Load Balancer. You just need to configure your ALIAS record to resolve the zone apex to the ELB.

Route 53 example: Latency Based Routing



Return address
nearest the
user/requester



Name	Type	Value
amdocstore.com	ALIAS	some-elb-name.us-west-2.elb.amazonaws.com
amdocstore.com	ALIAS	some-elb-name.ap-southeast-2.elb.amazonaws.com

Architecting on AWS – Overview of Services for Web Applications

Amazon web services | Training and Certification 10

Notes:

Amazon Route 53 has another load balancing option: Latency Based Routing. If you choose this option, Amazon Route 53 selects the address that's nearest to the user or requestor. Notice the user is located in the United States.

Route 53 example: Latency Based Routing (continue)



Name	Type	Value
amdocstore.com	ALIAS	some-elb-name.us-west-2.elb.amazonaws.com
amdocstore.com	ALIAS	some-elb-name.ap-southeast-2.elb.amazonaws.com

Architecting on AWS – Overview of Services for Web Applications

Training and Certification 11

Notes:

With Amazon Route 53, the user is automatically directed to the Amazon Elastic Load Balancer that's also in the United States, because it's closer to the user.

Network and Content Delivery

Amazon Route 53 

Amazon Elastic Load Balancer 

Amazon CloudFront 

Notes:

As we're already talking about Amazon Elastic Load Balancer, let's look at its features in more detail.

Amazon Elastic Load Balancer

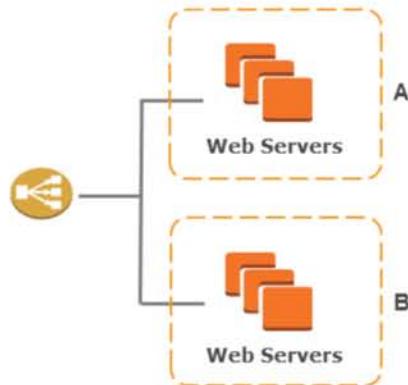
- Supports the routing and load balancing of HTTP, HTTPS and TCP traffic to EC2 instances
- Supports health checks to detect and remove failing instances
- Dynamically grows and shrinks based on traffic
- Seamlessly integrates with Auto-scaling to add and remove instances based on scaling activities
- Single CNAME provides stable entry point for DNS configuration

Notes:

Amazon Elastic Load Balancer has a number of features, but it basically does what its name implies: it load balances traffic to EC2 instances. It's considered elastic because it dynamically grows and shrinks, depending on traffic. You'll use an ELB (or more than one) often if you use Auto Scaling groups.

Amazon Elastic Load Balancer

- Distribute load across Availability Zones



Architecting on AWS – Overview of Services for Web Applications

Amazon Web Services | Training and Certification | 14

Notes:

Here's an example of how Amazon Elastic Load Balancer works.

Amazon Elastic Load Balancer

💡 Health Check example:

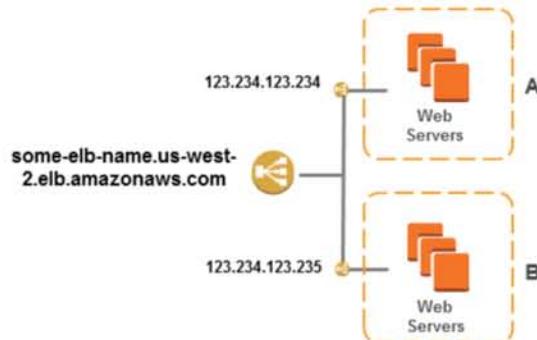
- HTTP GET /health.php
- Every x seconds
- Instance marked unhealthy after y consecutive failures

Notes:

How does Amazon Elastic Load Balancer determine the health of an EC2 instance? It periodically pings a PHP page. You can configure how many failures will cause the ELB to mark an EC2 as unhealthy or unavailable.

Amazon Elastic Load Balancer Scalability

- Stable DNS host name resolves via round robin to ELB IP addresses in each Availability Zone.



Architecting on AWS – Overview of Services for Web Applications

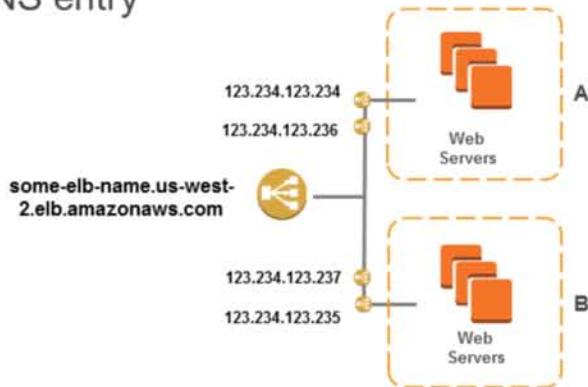
amazon
web services | Training and
Certification 16

Notes:

Amazon Elastic Load Balancers are designed for scalability and work across Availability Zones.

Amazon Elastic Load Balancer Scalability

- As traffic increases, AWS adds IP addresses to ELB's DNS entry



Architecting on AWS – Overview of Services for Web Applications

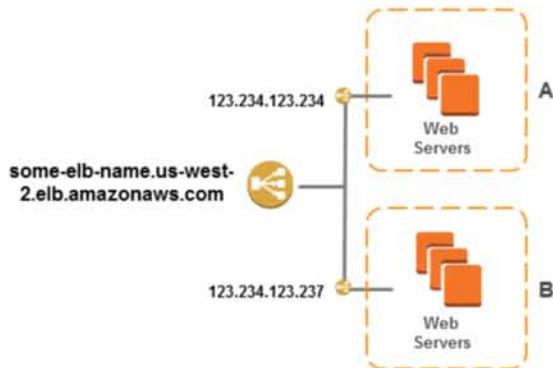
amazon
web services | Training and
Certification | 17

Notes:

As traffic increases, AWS adds more IP addresses to the ELB to compensate.

Amazon Elastic Load Balancer Scalability

- As traffic decreases, AWS automatically removes IP address from the ELB's DNS entry



Architecting on AWS – Overview of Services for Web Applications

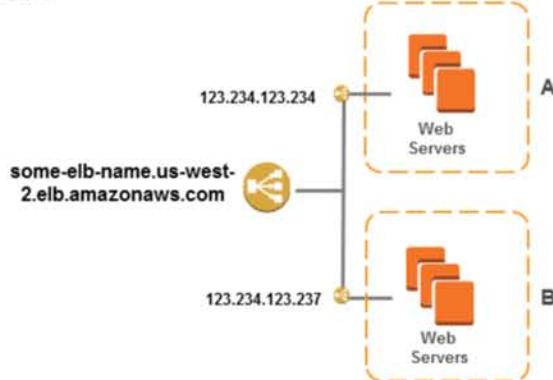
amazon web services | Training and Certification | 18

Notes:

And, just as important, ELBs scale down when traffic demands decrease.

Amazon Elastic Load Balancer Scalability

- Never refer to an ELB by its IP address. Always use its A Record



Architecting on AWS – Overview of Services for Web Applications

amazon
web services | Training and Certification | 19

Notes:

As this slide shows, never refer to an ELB by its IP address. Instead, always use its A record.

Network and Content Delivery

Amazon Route 53 

Amazon Elastic Load Balancer 

Amazon CloudFront 

Notes:

Moving on, let's talk about Amazon CloudFront.

Amazon CloudFront

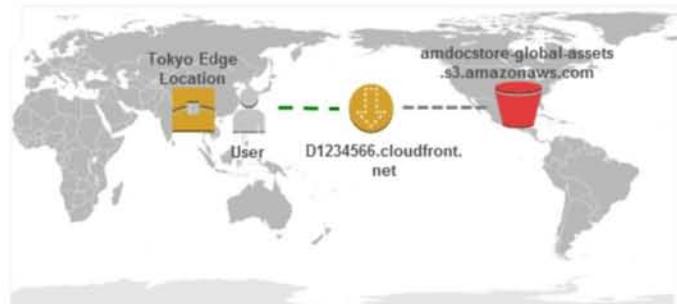
- Web service for content delivery
- Distribute content to end users with low latency, high data transfer speeds, and no commitments
- Delivers your content using a global network of edge locations
- Supports download, dynamic, streaming and live streaming

Notes:

Amazon CloudFront is a web service for content delivery.

Amazon CloudFront: Static content and S3 origin

- User in Japan requests content from S3 via CloudFront distribution
- User receives content from CloudFront edge location in Tokyo



Architecting on AWS – Overview of Services for Web Applications

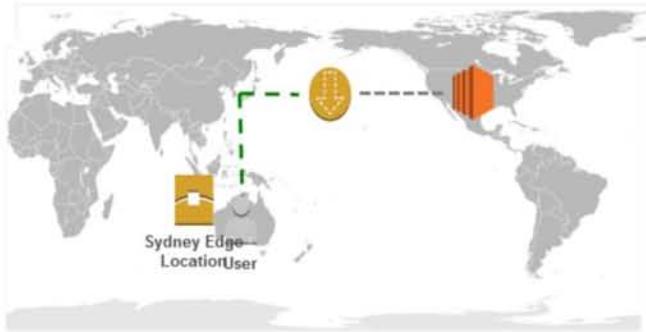
amazon web services | Training and Certification 22

Notes:

Here's an example of CloudFront in action. A user, located in Japan, goes to a web site that's delivered through CloudFront. CloudFront automatically sends the content to the user, using the closest edge location (in this case, Tokyo).

Amazon CloudFront: Dynamic content and EC2 origin

- User in Australia requests dynamic content from EC2 via CloudFront distribution.
- User receives dynamic content from CloudFront edge location in Sydney.



Architecting on AWS – Overview of Services for Web Applications

amazon
web services | Training and
Certification 23

Notes:

Now a user in Australia goes to the same web site, and Amazon CloudFront automatically serves the content using an edge location in Sydney.

Amazon CloudFront: Common Use Cases

- 💡 Delivering your entire website or web application by caching static content, proxy requests for dynamic content to the origin server
- 💡 Distributing software or other large files. Amazon CloudFront also offers lower prices than Amazon S3 at higher usage tiers
- 💡 Delivering media files using streaming of pre-recorded media and live events

Notes:

There are many great use cases for Amazon CloudFront, including:

Delivering your entire website or web application

A typical website generally contains a mix of static content and dynamic content. Static content includes images or style sheets; dynamic or application generated content includes elements of your site that are personalized to each viewer. Amazon CloudFront can help improve performance of your entire website in the following ways:

Amazon CloudFront can cache static content at each edge location. This means that your popular static content (e.g., your site's logo, navigational images, cascading style sheets, JavaScript code, etc.) will be available at a nearby edge location for the browsers to download with low latency and improved performance for viewers.

Caching popular static content with Amazon CloudFront also helps you offload requests for such files from your origin sever – CloudFront serves the cached copy when available and only makes a request to your origin server if the edge location receiving the browser's request does not have a copy of the file. Amazon CloudFront proxies requests for dynamic or interactive content back to your origin running in an AWS Region. Each of your end users is routed to the edge location closest to them, in terms of internet latency. Then, their requests are carried back to your origin server running in AWS on connections that Amazon monitors and optimizes for performance. Amazon CloudFront also reuses existing connections between the CloudFront edge and the origin server reducing connection setup latency for each origin request. Other connection optimizations

are also applied to avoid internet bottlenecks and fully utilize available bandwidth between the edge location and the viewer. This means that Amazon CloudFront can speed-up the delivery of your dynamic content and provide your viewers with a consistent and reliable, yet personalized experience when navigating your web application.

You can use a single CloudFront distribution to deliver your entire website, including both static and dynamic or interactive content. This means that you can continue to use a single domain name (e.g., www.mysite.com) for your entire website without the need to separate your static and dynamic content. Meanwhile, you can still continue to use separate origin servers for different types of content on your website. Amazon CloudFront provides you with granular control for configuring multiple origin servers and caching properties for different URLs on your website. These performance optimizations and functionality can help speed up the download of your entire website which can help lower site abandonment.

Distributing software or other large files

Amazon CloudFront is a good choice for software developers who wish to distribute applications, updates or other downloadable software to end users. Amazon CloudFront's high data transfer rates speed up downloading your applications, improving the customer experience and lowering your costs . Amazon CloudFront also offers lower prices than Amazon S3 at higher usage tiers.

Delivering media files

If your application involves rich media – audio or video – that is frequently accessed, you will benefit from Amazon CloudFront's lower data transfer prices and improved data transfer speeds. Amazon CloudFront offers multiple options for delivering your media files – both pre-recorded media and live media.

Streaming of pre-recorded media: You can deliver your on-demand media using Adobe's Real Time Messaging Protocol (RTMP) streaming via Amazon CloudFront. You store the original copy of your media files in Amazon S3 and use Amazon CloudFront for low-latency delivery of your media content. Amazon CloudFront integrates with Amazon S3 so you can configure media streaming by making a simple API call or with a few clicks in the AWS Management Console. You also benefit for the high throughput delivery of your media when using Amazon CloudFront, so you can deliver content in full HD quality to your viewers.

Progressive download of on-demand media: You can store the original versions of your media content in Amazon S3 and configure an Amazon CloudFront download distribution for progressive download of your video and audio files. Popular media files are cached at the edge to help you scale and give your viewers the best possible performance.

Delivering live events: If you need to deliver a live event – audio or video – to a global audience, Amazon CloudFront can improve performance and help offload requests to your origin infrastructure by caching your live media for a short period of time and collapsing simultaneous requests for the same media fragment to a smaller number of requests sent to the origin. In addition, Amazon CloudFront's live HTTP solutions give you the ability to deliver your live event to viewers using different

device platforms, including both Flash based and Apple iOS devices.

Amazon CloudFront

- Streaming media
- Live media streaming via Adobe Flash Media Server (FMS) to both Flash Player and Apple iOS devices
- Live Smooth Streaming using Windows Media Services
- Stream pre-recorded media stored in S3 via FMS or via progressive-download
- Supports Custom SSL
- Collapse simultaneous requests for the same object before contacting your origin server
- Multiple CNAMEs for increased simultaneous connections

Notes:

Refer to online documentation about working with streaming distribution →
<http://docs.aws.amazon.com/AmazonCloudFront/2012-05-05/DeveloperGuide/WorkingWithStreamingDistributions.html>

Custom SSL Support: Amazon CloudFront gives you three options for accelerating your entire website while delivering your content securely over HTTPS from all of CloudFront's edge locations. In addition to delivering securely from the edge, you can also configure CloudFront to use HTTPS connections for origin fetches so that your data is encrypted end-to-end from your origin to your end users. By default, you can deliver your content to viewers over HTTPS by using your CloudFront distribution domain name in your URLs, for example, <https://dxxxxx.cloudfront.net/image.jpg>. If you want to deliver your content over HTTPS using your own domain name and your own SSL certificate, you can use one our Custom SSL certificate support features. For more detail, please refer to online documentation →<http://aws.amazon.com/cloudfront/custom-ssl-domains/>

From Edge Location to Origin - The nature of dynamic content requires repeated back and forth calls to the origin server. CloudFront edge locations collapse multiple concurrent requests for the same object into a single request. They also maintain persistent connections to the origins (with the large window size). Connections to other parts of AWS are made over high-quality networks that are monitored by Amazon for both availability and performance. This monitoring has the beneficial side effect of keeping error rates low and window sizes high. These

optimizations further help reduce the need to scale your origin infrastructure as your website becomes more popular.

(Optional) You can associate **one or more CNAME aliases** with a distribution so that you can use your domain name (for example, example.com) in the URLs for your objects instead of using the domain name that CloudFront assigned when you created your distribution. For more information, refer to online documentation → <http://docs.aws.amazon.com/AmazonCloudFront/2012-05-05/DeveloperGuide/CNAMEs.html>

Caching:

With Amazon CloudFront's Dynamic Content Delivery, you can configure multiple cache behaviors for your download distribution. You can include these configuration values: origin server name, viewer connection protocol, minimum expiration period, query string parameters, cookies and trusted signers in cases of private content in each cache behavior. For more information, refer to online documentation → <http://docs.aws.amazon.com/AmazonCloudFront/latest/DeveloperGuide/RequestAndResponseBehaviorCustomOrigin.html>

You can control how long your objects stay in a CloudFront cache before CloudFront forwards another request to your origin. Reducing the duration allows you to serve dynamic content. Increasing the duration means your customers get better performance because your objects are more likely to be served directly from the edge cache. A longer duration also reduces the load on your origin. (Minimum TTL of 0 for 'not cached') Refer to online documentation → <http://docs.aws.amazon.com/AmazonCloudFront/latest/DeveloperGuide/Expiration.html>

Topics

- ➲ AWS products for network content and delivery
 - Amazon Route 53
 - Amazon Elastic Load Balancer
 - Amazon CloudFront
- ➲ AWS products for deployment and management
 - Amazon CloudWatch
 - Amazon Elastic Beanstalk
 - AWS CloudFormation

AWS Products for Deployment and Management

- Amazon CloudWatch



- Amazon Elastic Beanstalk



- AWS CloudFormation



Architecting on AWS – Overview of Services for Web Applications

 Training and Certification 27

Notes:

Here are three tools we'll go over:

- Amazon CloudWatch
- Amazon Elastic Beanstalk
- AWS CloudFormation

Amazon CloudWatch

- Monitor AWS resources automatically without installing additional software
- Alarm when a value is breached, triggering an action (send an e-mail, add/remove EC2 instances to an Auto Scaling Group, and so on)
- Visibility into resource utilization, operational performance, and overall demand patterns
- Metrics, including CPU utilization, disk I/O, and network traffic
- Custom application-specific metrics of your own
- Accessible via AWS Management Console, APIs, SDK, or CLI

Notes:

Amazon CloudWatch supports a number of features. Here are the most relevant ones.

You can publish your own metrics to CloudWatch with the put-metric-data command (<http://docs.aws.amazon.com/cli/latest/reference/cloudwatch/put-metric-data.html>). You can view statistical graphs of your published metrics with the AWS Management Console.

If you call put-metric-data with a new metric name, CloudWatch creates a new metric for you. Otherwise, CloudWatch associates your data with the existing metric that you specify.

Example:

Discussion Forum has a post about custom metric to monitor memory →
<https://forums.aws.amazon.com/message.jspa?messageID=266893>

Amazon CloudWatch

- 💡 Many AWS resources provide metrics automatically, and at no charge:

- | | |
|---------------|---------------------|
| ➤ EC2 | ➤ Elastic MapReduce |
| ➤ DynamoDB | ➤ Redshift |
| ➤ EBS | ➤ Route53 |
| ➤ ElastiCache | ➤ Opsworks |
| ➤ RDS | ➤ Storage Gateways |
| ➤ SNS | ➤ ELB |
| ➤ SQS | ➤ Billing |

Notes:

Best of all, many AWS resources provide CloudWatch metrics automatically—and at no additional charge. (So it's a good idea to take advantage of it!)

Here's the current list of AWS services that provide Amazon CloudWatch metrics. We'll look at a few of these in more detail.

CloudWatch monitors AWS resources automatically, without installing additional software:

- **Basic Monitoring** for Amazon EC2 instances: Seven pre-selected metrics at five-minute frequency and three status check metrics at one-minute frequency, free of charge.
- **Detailed Monitoring** for Amazon EC2 instances: Ten pre-selected metrics at one-minute frequency, for an additional charge. Aggregate metrics across groups of similar instances with Detailed Monitoring enabled are included.
- **DynamoDB tables**: seven pre-selected metrics at five-minute frequency, free of charge.
- **EBS PIOPS volumes**: ten pre-selected metrics at one-minute frequency, free of charge.
- **EBS standard volumes**: eight pre-selected metrics at five-minute frequency, free of charge.
- **ElastiCache nodes**: thirty-nine pre-selected metrics at one-minute frequency, free of charge.

- **RDS DB instances:** fourteen pre-selected metrics at one-minute frequency, free of charge.
- **SNS topics:** four pre-selected metrics at five-minute frequency, free of charge.
- **SQS queues:** eight pre-selected metrics at five-minute frequency, free of charge.
- **Elastic MapReduce job flows:** twenty-six pre-selected metrics at five-minute frequency, free of charge.
- **Redshift:** Sixteen pre-selected metrics at one-minute frequency, free of charge.
- **Route53 health checks:** One pre-selected metric at one-minute frequency, free of charge.
- **Opsworks:** fifteen pre-selected metrics at one-minute frequency, free of charge.
- **Storage Gateways:** eleven pre-selected gateway metrics and five pre-selected storage volume metrics at five-minute frequency, free of charge.
- **Auto Scaling groups:** seven pre-selected metrics at one-minute frequency, optional and charged at standard pricing.
- **Elastic Load Balancers:** thirteen pre-selected metrics at one-minute frequency, free of charge.
- **Estimated charges on your AWS bill:** you can also choose to enable metrics to monitor your AWS charges. The number of metrics depends on the AWS products and services that you use, and these metrics are free of charge.

Amazon CloudWatch: EC2

- EC2: 5-minute frequency for no charge, 1-minute frequency for a small fee, for example:
 - CPU Utilization (%)
 - NetworkOut
 - NetworkIn
 - DiskReadBytes
 - DiskWriteBytes
 - And so on

Notes:

This slide shows the CloudWatch metrics for EC2.

Amazon CloudWatch: ELB

- ELB: 1-minute frequency for no charge:

- RequestCount
- HealthyHostCount
- UnHealthyHostCount
- Latency
- HTTPCode_Backend_2XX
- HTTPCode_Backend_3XX
- HTTPCode_Backend_4XX
- HTTPCode_Backend_5XX
- And so on

Notes:

In this slide, you can see the CloudWatch metrics provided for Elastic Load Balancer.

Amazon CloudWatch

💡 Alarms

- When a metric is breached (for example, instance CPU utilization > 80% for 5 minutes), take some action, such as:
 - Send an e-mail
 - Add or remove an EC2 instance

Notes:

Amazon CloudWatch uses alarms to inform you when a threshold's been met. An alarm can perform one of several actions: such as send an email, or add or remove an EC2 instance.

Amazon CloudWatch

- Visualize metrics in the AWS Management Console, or download via the API



Architecting on AWS – Overview of Services for Web Applications

amazon web services | Training and Certification 33

Notes:

You can view metrics through the AWS Management Console, or you can download them through its API.

AWS Products for Deployment and Management

Amazon CloudWatch



Amazon Elastic Beanstalk



AWS CloudFormation



Amazon Elastic Beanstalk

- 💡 Simply upload your application (or push with git) and Beanstalk deploys to an environment
- 💡 Environment includes an Elastic Load Balancer, Auto Scaling/EC2, and Notifications
- 💡 Manage multiple application versions across different environments (dev, test, prod)
- 💡 Supported containers include Java, .NET, PHP, Python, node.js, and Ruby
- 💡 Application and server logs pushed to S3 automatically

Notes:

Amazon Elastic Beanstalk basically operates as PaaS—Platform as a Service.

AWS Products for Deployment and Management

Amazon CloudWatch 

Amazon Elastic Beanstalk 

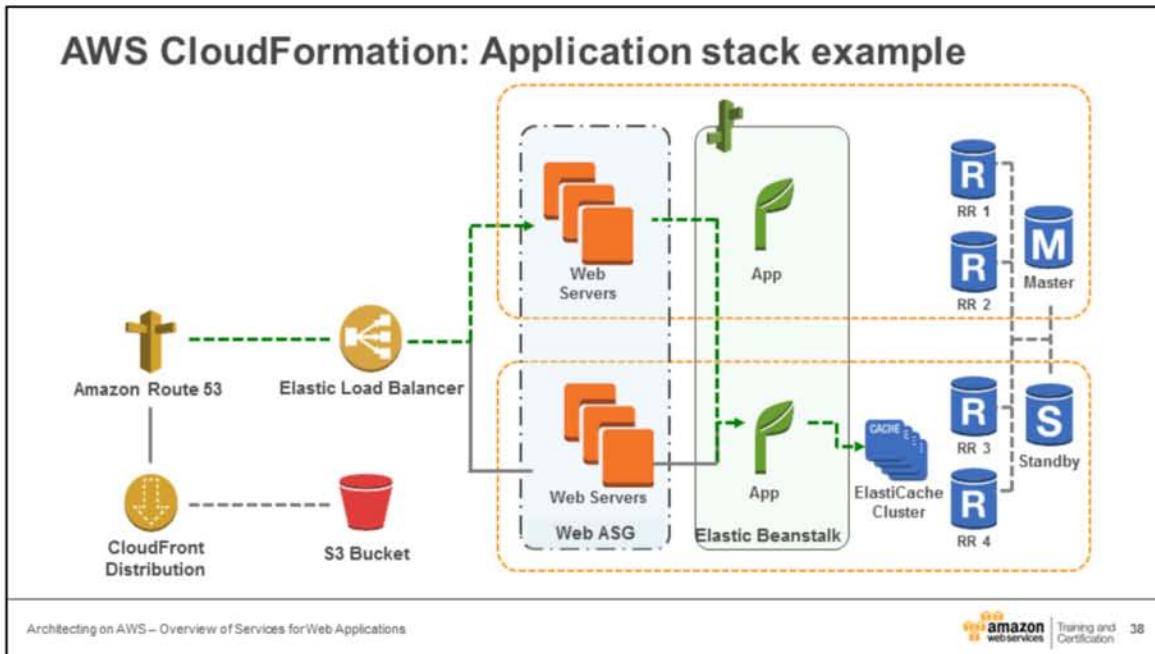
AWS CloudFormation 

AWS CloudFormation

- Infrastructure as code, suitable for change management in version control (git, svn, and so on)
- Define an entire application stack (all resources required for your application) in a JSON template file
- Define runtime parameters for a template (EC2 Instance Size, EC2 Key Pair, and so on)

Notes:

AWS CloudFormation allows you to build your infrastructure as code. This makes it great for version control.

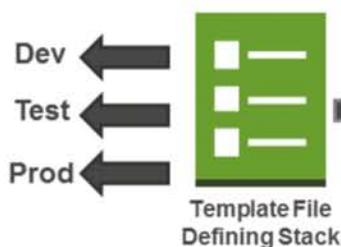


Notes:

In this diagram, we have a full application stack running in AWS.

AWS CloudFormation: Application stack example (continue)

Build out multiple environments, such as for Development, Test, and Production using the template



Use the version control system of your choice to store and track changes to this template

The entire application can be represented in an AWS CloudFormation template.

Architecting on AWS – Overview of Services for Web Applications

 Amazon Web Services | Training and Certification 39

Notes:

1. The entire application can be represented in an AWS CloudFormation template.
2. You can use the version control system of your choice to store and track changes to this template.
3. You can use the template to quickly build out multiple environments, such as for Development, Test, and Production.

AWS CloudFormation Example (1 of 3)

```
{  
    "Description" : "Create an EC2 instance running the Amazon Linux 32 bit AMI.",  
    "Parameters" : {  
        "KeyPair" : {  
            "Description" : "The EC2 Key Pair to allow SSH access to the instance",  
            "Type" : "String"  
        }  
    },  
    "Resources" : {  
        "Ec2Instance" : {  
            "Type" : "AWS::EC2::Instance",  
            "Properties" : {  
                "KeyName" : { "Ref" : "KeyPair" },  
                "ImageId" : "ami-75g0061f",  
                "InstanceType" : "m1.medium"  
            }  
        }  
    },  
    "Outputs" : {  
        "InstanceId" : {  
            "Description" : "The InstanceId of the newly created EC2 instance",  
            "Value" : { "Ref" : "Ec2Instance" }  
        }  
    }  
}
```

Architecting on AWS – Overview of Services for Web Applications

 Training and Certification 40

Notes:

Here's an example of an AWS CloudFormation Template. We'll go into this in more detail later.

AWS CloudFormation Example (2 of 3)

```
{  
    "Description" : "Create an EC2 instance running the Amazon Linux 32 bit AMI.",  
    "Parameters": {  
        "KeyPair" : {  
            "Description" : "The EC2 Key Pair to allow SSH access to the instance",  
            "Type" : "String"  
        }  
    },  
    "Resources" : {  
        "Ec2Instance" : {  
            "Type" : "AWS::EC2::Instance",  
            "Properties" : {  
                "KeyName" : { "Ref" : "KeyPair" },  
                "ImageId" : "ami-75g0061f",  
                "InstanceType" : "m1.medium"  
            }  
        }  
    },  
    "Outputs" : {  
        "InstanceId" : {  
            "Description" : "The InstanceId of the newly created EC2 instance",  
            "Value" : { "Ref" : "Ec2Instance" }  
        }  
    }  
}
```

Notice that you need to use
an EC2 KeyPair for the
CloudFormation template to
work.

AWS CloudFormation Example (3 of 3)

```
{  
    "Description" : "Create an EC2 instance running the Amazon Linux 32 bit AMI.",  
    "Parameters" : {  
        "KeyPair" : {  
            "Description" : "The EC2 Key Pair to allow SSH access to the instance",  
            "Type" : "String"  
        }  
    },  
    "Resources" : {  
        "Ec2Instance" : {  
            "Type" : "AWS::EC2::Instance",  
            "Properties" : {  
                "KeyName" : { "Ref" : "KeyPair" },  
                "ImageId" : "ami-75g0061f",  
                "InstanceType" : "m1.medium"  
            }  
        }  
    },  
    "Outputs" : {  
        "InstanceId" : {  
            "Description" : "The InstanceId of the newly created EC2 instance",  
            "Value": { "Ref" : "Ec2Instance" }  
        }  
    }  
}
```

You can define exactly what type of EC2 instance you want to launch.

Module review

- 💡 List the three main AWS products for network and content delivery
- 💡 List the three main AWS products for deployment and management

Notes:

AWS products for network and content delivery:

- Amazon Route 53
- Amazon Elastic Load balancer
- Amazon CloudFront

AWS products for deployment and management:

- Amazon CloudWatch
- Amazon Elastic Beanstalk
- Amazon CloudFormation

Copyright © 2013, 2014 Amazon Web Services, Inc. and its affiliates. All rights reserved.

This work may not be reproduced or redistributed, in whole or in part, without prior written permission from Amazon Web Services, Inc. Commercial copying, lending, or selling is prohibited.

Errors or corrections? Email us at aws-course-feedback@amazon.com.
Other questions? Email us at aws-training-info@amazon.com.

All trademarks are the property of their owners.



Module 7: Elasticity, Scalability, and Bootstrapping

Architecting on AWS – Elasticity, Scalability, and Bootstrapping

 Amazon
Training and
Certification 1

Notes:

In this module, we'll discuss elasticity, scalability, and bootstrapping—three elements that can help you maximize your use of AWS resources while keeping costs under control.

Topics

- 💡 Basic tenets of AWS
- 💡 Patterns and (anti-patterns) for creating scalable architectures in AWS
- 💡 Bootstrapping EC2 Instances
- 💡 Building with CloudFormation
- 💡 Components of Auto Scaling

Notes:

During this module, we'll discuss the following:

- **Basic Tenets of AWS.** These key ideas can help you build reliable, scalable systems with AWS.
- **Patterns (and anti-Patterns) for creating scalable architectures in AWS.** Here, we'll discuss some good—and some not-so-good—methods of using AWS resources.
- **Bootstrapping EC2 instances.** This module is where we'll get into how to spin up instances pre-configured for your applications.
- **Building with CloudFormation.** We'll show you how this tool can help you spin up entire systems within AWS with just a few clicks.
- **Components of Auto Scaling.** A feature that's talked about a lot—Auto Scaling lets you scale up and scale in as necessary.

Topics

💡 Basic tenets of AWS

- Review how traditional architectures accommodate expected load variation
- Anti-patterns for elastic, scalable architectures
- 4 patterns for elastic, scalable architectures

💡 Patterns and (anti-patterns) for creating scalable architectures in AWS

- 💡 Bootstrapping EC2 Instances
- 💡 Building with CloudFormation
- 💡 Components of Auto Scaling

Notes:

In "Basic tenets of AWS," we will discuss:

- Review how traditional architectures accommodate expected load variation
- Anti-patterns for elastic, scalable architectures
- 4 patterns for elastic, scalable architectures

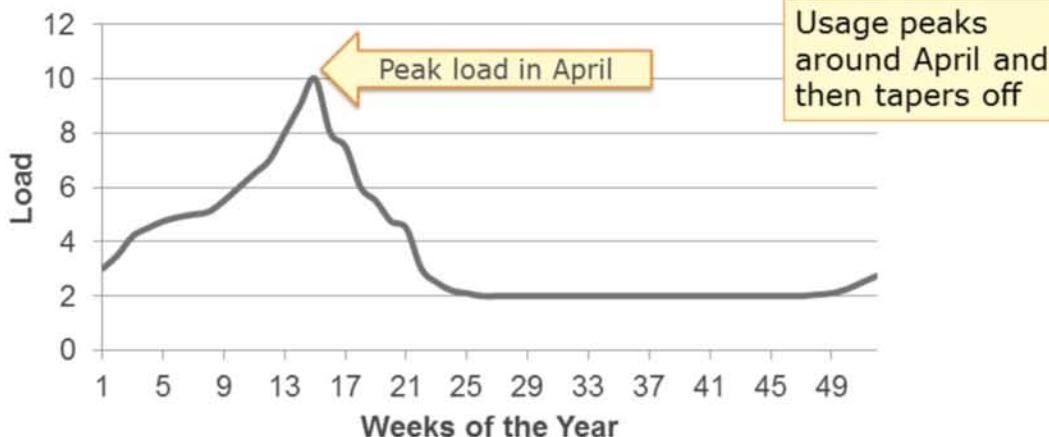
Expected Load Variation

- 💡 Non-cloud systems typically over-provision and under-utilize
- 💡 Under-utilization costs:
 - Capital
 - Space
 - Power
 - Cooling
 - Maintenance

Notes:

A first tenet to consider is expected load variation. A common challenge for any IT department is to figure out how much capacity is necessary. It's very easy (and often costly) to over-provision and under-utilize.

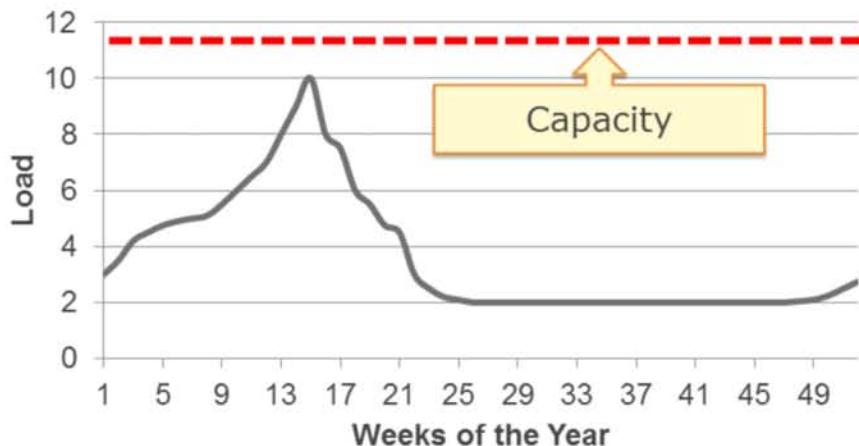
Expected Load Variation Example (1 of 6)



Notes:

This graph shows a typical usage scenario for an online shopping app. Notice that usage peaks around April, then tapers off.

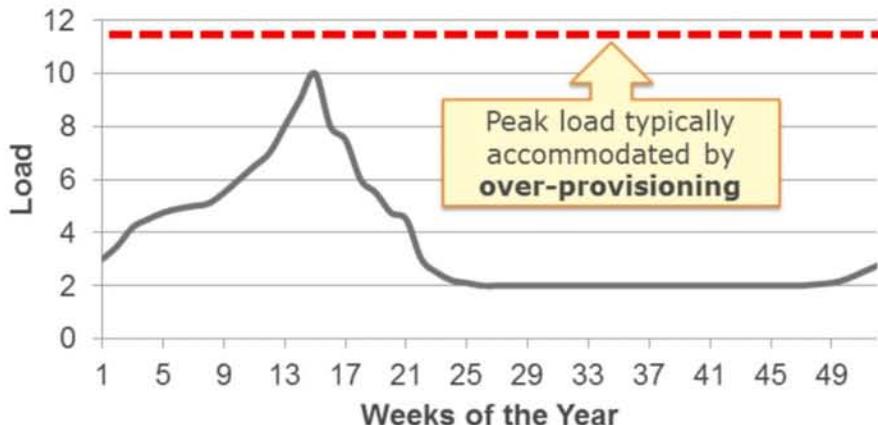
Expected Load Variation Example (2 of 6)



Notes:

To ensure that the app has enough capacity, it's common to provision based on peak usage.

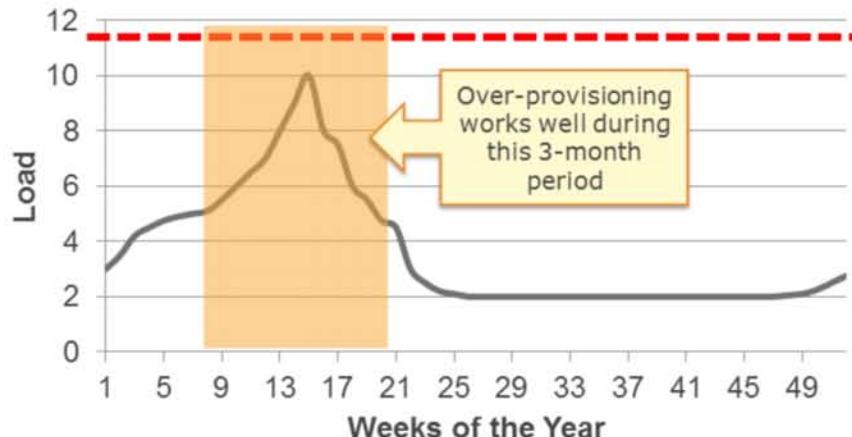
Expected Load Variation Example (3 of 6)



Notes:

This usually means that the app is over-provisioned.

Expected Load Variation Example (4 of 6)



Notes:

Over-provisioning isn't a big deal when app usage is at its peak.

Expected Load Variation Example (5 of 6)



Architecting on AWS - Elasticity, Scalability, and Bootstrapping

Amazon
Training and
Certification 9

Notes:

But that capacity goes to waste the rest of the year.

Expected Load Variation Example (6 of 6)



Architecting on AWS - Elasticity, Scalability, and Bootstrapping

Amazon
Training and
Certification 10

Notes:

If we could find a way to reduce capacity without sacrificing performance during peak usage, we could save a significant amount of time and money. In some cases, a savings of almost 50%.

Elastic environments

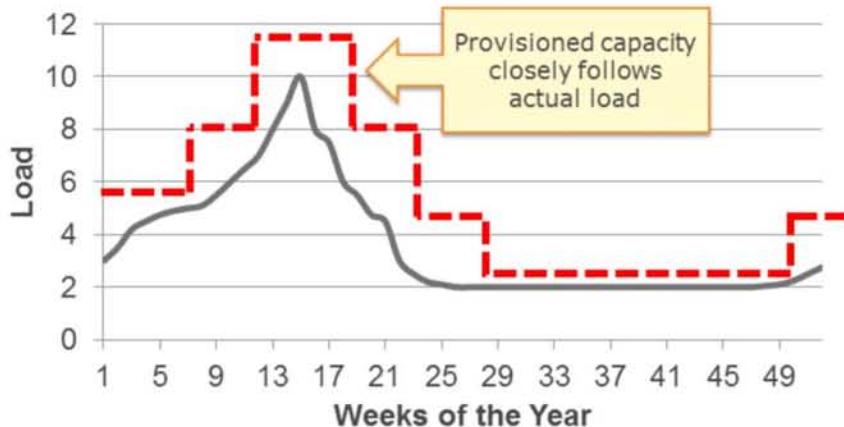
- 💡 Highly utilized all the time
- 💡 Provision resources “just in time”

Notes:

A better solution would be a more elastic environment. These types of environments have two main qualities:

- They're highly utilized, all the time
- They provision resources in a “just in time” fashion

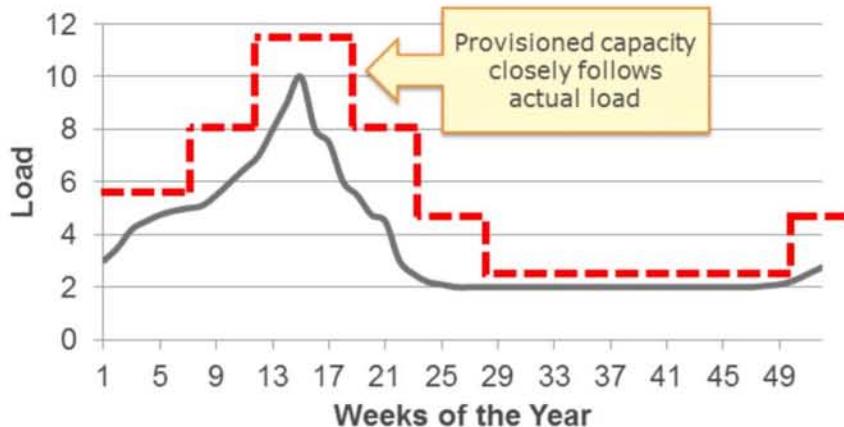
Elastic environments scenario (1 of 4)



Notes:

Let's go back to our online shopping app and see how an elastic infrastructure changes things. Here, the dotted red line shows the ideal level of capacity we'd like to have (and pay for) throughout the year. An elastic environment tries to use provisioned capacity that closely follows actual load.

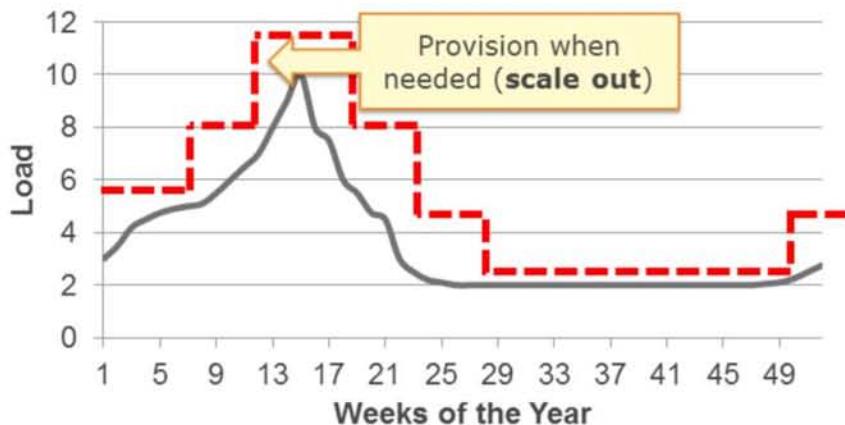
Elastic environments scenario (2 of 4)



Notes:

An elastic environment tries to use provisioned capacity that closely follows actual load.

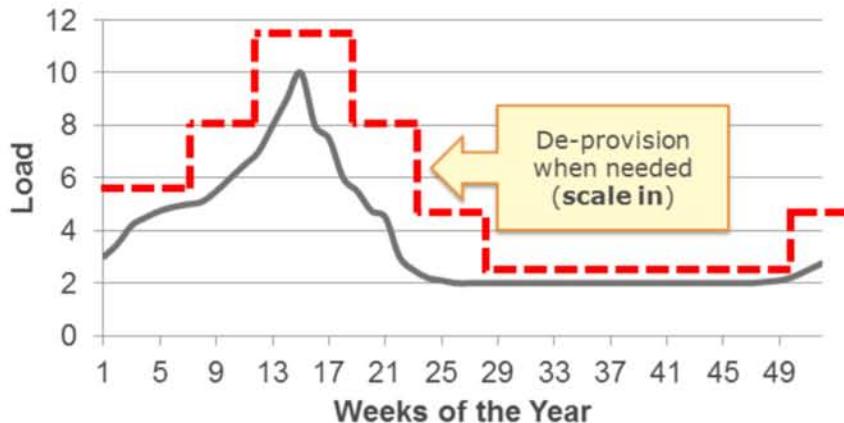
Elastic environments scenario (3 of 4)



Notes:

As usage increases, we provision more resources. (This is **scaling out**.)

Elastic environments scenario (4 of 4)



Notes:

Equally important is what happens when usage decreases. That's when we de-provision resources. (This is called **scaling in**.)

To sum up, the basic tenets for building systems with AWS are:

- Provision only what you need, when you need it.
- Scale up when usage increases
- Scale down when usage decreases

Topics

- 💡 Basic tenets of AWS
- 💡 Patterns and (anti-patterns) for creating scalable architectures in AWS
 - 💡 Bootstrapping EC2 Instances
 - 💡 Building with CloudFormation
 - 💡 Components of Auto Scaling

Notes:

Tenets are great, but how do we put them to work? Let's look at some common patterns (and anti-patterns) for building scalable systems with AWS.

Patterns and anti-patterns

💡 Anti-Pattern: Manual Processes

- When direct, manual intervention is required to start new resources—or scale existing ones—will be a blocker at scale

💡 Pattern: Automated Processes

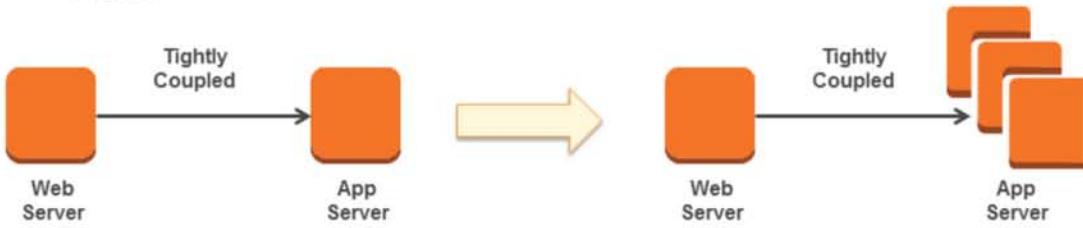
Notes:

First, let's look at an anti-pattern: manual processes. This is when manual intervention is required to start new resources, or scale existing ones. A better option: automated processes. AWS provides you with a lot of tools to help automate the creation of new systems and the scaling of existing ones. (We'll talk more about this when we get to CloudFormation and Auto Scaling.)

Patterns and anti-patterns: Integration

⚠ Anti-Pattern: Tightly-coupled

- Application components in which a single unit depends on another specific single unit behave poorly when the dependency fails or needs to **subdivide (for example, grow horizontally)** to scale.

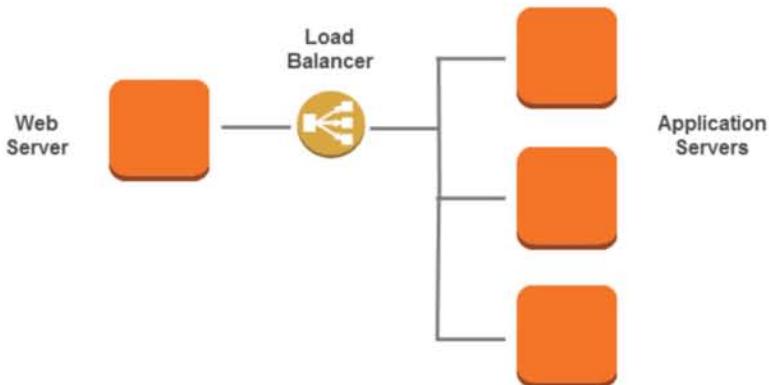


Notes:

Another anti-pattern that's really common: tightly coupled systems. Here, we're talking about systems in which one unit depends on a specific instance of another unit. When that unit goes down, everything comes to a halt. Tightly-coupled systems also perform poorly when the system needs to scale horizontally. In this example, the problem is that the Web server has to be told that there are new app servers available.

Patterns and anti-patterns: Integration (continue)

💡 Pattern: Loosely-coupled



Architecting on AWS - Elasticity, Scalability, and Bootstrapping

Amazon
Training and
Certification 19

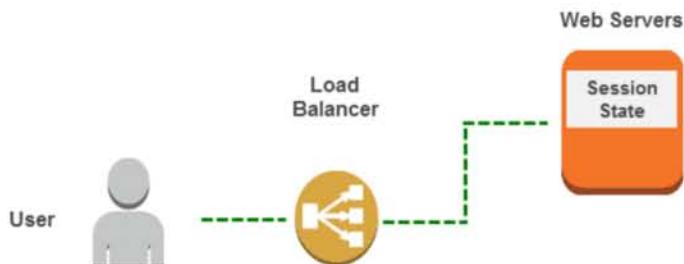
Notes:

A better option: loosely-coupled systems. A common example of a loosely coupled system (as shown here) is a load balancer, that acts as an intermediary between the Web server and one or more app servers. With a loosely coupled system like this, you can start up another app server, and the Web server doesn't need to know about it—the load balancer distributes traffic accordingly.

Patterns and anti-patterns: Session state

⚠ Anti-Pattern: Stateful

- Applications that store state on one instance are more challenging to scale horizontally



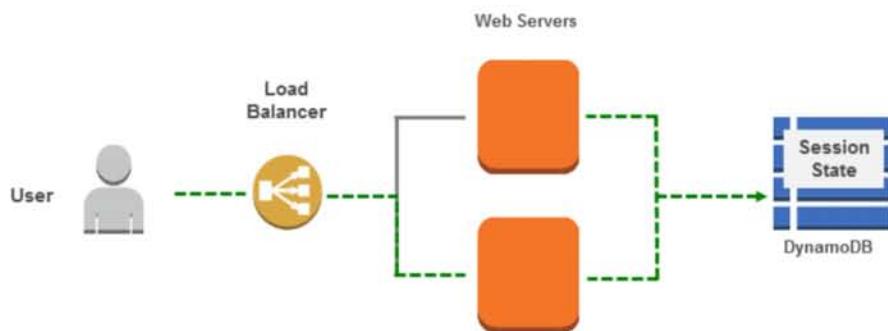
Notes:

Let's look at another anti-pattern: stateful apps. Here, we're talking about apps that store a state in one (and only one) instance.

Patterns and anti-patterns: Session state (continue)

Pattern: Stateless

- Move state to a shared, accessible location



Notes:

A better option: stateless apps, where any instance can access the state of the activity at any time.

Patterns and anti-patterns

⚠ Anti-Pattern: Vertical

- Vertical scaling (more CPU, memory, and so on) will eventually run out of room



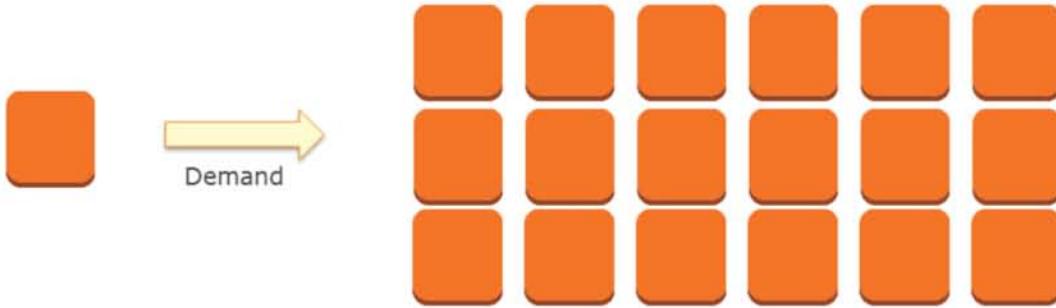
Notes:

Vertical scaling is also an anti-pattern. Sooner or later, you reach a point where you cannot add any more CPU, memory, and so on. You can only upgrade a server to a certain point!

Patterns and anti-patterns

Pattern: Horizontal

- Add and remove instances as needed



Notes:

Instead, build your systems so they can scale horizontally, and add or remove systems as needed. With AWS, you can continue to add or remove instances horizontally, far exceeding the capabilities of vertical scaling.

Topics

- 💡 Basic tenets of AWS
- 💡 Patterns and (anti-patterns) for creating scalable architectures in AWS
- 💡 Bootstrapping EC2 Instances
- 💡 Building with CloudFormation
- 💡 Components of Auto Scaling

Notes:

Now that we have a good idea of what to think about as we build systems with AWS, let's talk about bootstrapping EC2 instances.

Bootstrapping

💡 The process of automatically setting up your servers

💡 Examples:

- Install latest software
- Copy data from S3
- Register with DNS
- Start services
- Update packages
- Reboot
- Open port 80
- Register with load balancer
- Mount devices

Notes:

Bootstrapping is the process of automatically setting up servers. As you can imagine, this is an important capability when building scalable systems. Bootstrapping actions can be almost anything. Here are some examples.

Bootstrapping tools

- 💡 Scripts on instance
- 💡 Configuration Management Tools
- 💡 Amazon OpsWorks

Notes:

With AWS, you have several tools to help you bootstrap instances. These include (but aren't necessarily limited to) bash and PowerShell scripts, as well as configuration tools like Chef and Puppet.

EC2 Metadata and UserData (1 of 4)

- 💡 Every EC2 instance has access to local instance Metadata and UserData service
- 💡 Metadata: immutable information about the instance
- 💡 Accessible from within the instance via HTTP at <http://169.254.169.254/latest/meta-data/>

Notes:

To maximize your use of bootstrapping with AWS, you should know how to access the MetaData and UserData of the EC2 instance. The MetaData of an EC2 instance is immutable information about the instance. You can get this data from within the instance by using the URL shown here. An EC2 instance can also include UserData. This are instructions that a script on the instance can parse and use to configure the system. Note that UserData has a maxmize size of 16KB.

EC2 Metadata and UserData (2 of 4)

- 💡 Script(s) on instance may retrieve useful information about the instance, such as:
 - Host name
 - AMI ID
 - Instance ID
 - Public/Private DNS
 - Availability Zone
- 💡 Pass up to 16KB of text to an instance on launch
- 💡 Text can be parsed by script on instance and used to configure the machine

Notes:

Your scripts can use this Metadata to get a lot useful information about the instance.
The Metadata server also stores IAM role credentials.

EC2 Metadata and UserData (3 of 4)

- 💡 Pass up to 16KB of text to an instance on launch
- 💡 Text can be parsed by script on instance and used to configure the machine

Notes:

An EC2 instance can also include UserData. This are instructions that a script on the instance can parse and use to configure the system. Note that UserData has a maximize size of 16KB.

EC2 Metadata and UserData (4 of 4)

Request Instances Wizard

CHOOSE AN AMI INSTANCE DETAILS CREATE KEY PAIR

Number of Instances: 1

Advanced Instance Options

Here you can choose a specific kernel or RAM disk to use Detailed Monitoring or enter data that will be available from the instance.

Kernel ID: Use Default

Monitoring: Enable CloudWatch detailed monitoring for this instance (additional charges will apply)

User Data:

as text
 as file

ROLE = App Server
DB_ADDR = 10.28.117.88
EIP_TO_ATTACH = 16.17.18.19

(Use shift+enter to insert a newline)
 base64 encoded

Custom script on AMI
(`script_runner.py`) fetches
userdata, parses it, and
configures EC2 Instance on
boot

Notes:

You can specify UserData in a couple of ways. The Management Console, for example, provides you with the option of including UserData when you spin up a new EC2 instance.

UserData and CloudInit

- 💡 CloudInit executes UserData on first boot if UserData begins with:
 - `#!` (Linux)
 - `<script>` (Windows; technically, EC2Config, not CloudInit, does this)
- 💡 CloudInit is installed on Amazon Linux, Ubuntu, and RHEL AMIs
- 💡 EC2Config is installed on Windows Server AMIs
- 💡 Both may be installed on other distributions via a package repo or source

Notes:

So, I mentioned that there needs to be a script on the EC2 instance that can parse the UserData. Most EC2 instances include CloudInit, which can parse your UserData and run its instructions. To use CloudInit on Linux machines, start your UserData with `#!`.

To use CloudInit on Windows machines, use `<script>`. (Technically, EC2Config parses these instructions, but the result is the same.)

I said that *most* EC2 instances use CloudInit (or EC2Config). Let's get more specific:

- CloudInit is available on Amazon Linux, Ubuntu, and RHEL Amazon Machine Images (AMIs).
- EC2Config is on all Windows Server AMIs.

Either one may be available on other distributions, however.

UserData and CloudInit (continue)

- 💡 UserData to install Apache and MySQL on boot, and attach an EIP:

```
#!/bin/bash

# Install Apache, PHP, and MySQL
yum install -y httpd mysql-server

# Attach an Elastic IP to this instance
ec2-associate-address \
23.34.45.56 \
-i $(curl http://169.254.169.254/latest/meta-data/instance-id)
```

Notes:

Here's an example of UserData that:

- Installs Apache
- Installs MySQL
- Attaches an Amazon Elastic IP address (EIP)

Bootstrapping: 3 ways

- 💡 Fully-functional
- 💡 Partially configured
- 💡 Base OS, configured with code

Notes:

So how much configuration can you do with bootstrapped AMIs? Generally, there are three categories:

- Fully functional
- Partially configured
- Configured with code

AMIs and Bootstrapping



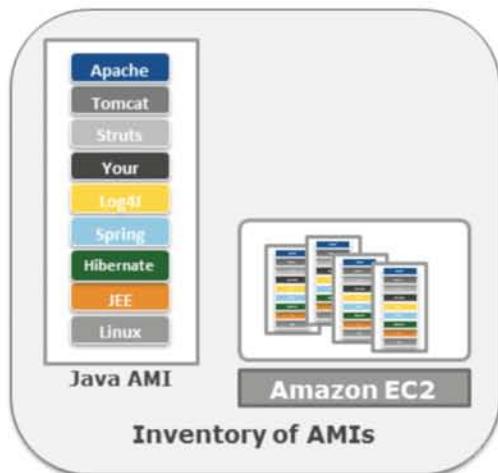
An example of full stack required to run your application.

There are 3 AMI/bootstrapping techniques to configure this stack.

Notes:

Let's look at these three categories and how they affect spinning up new instances.

Bootstrapping: Fully-functional AMI

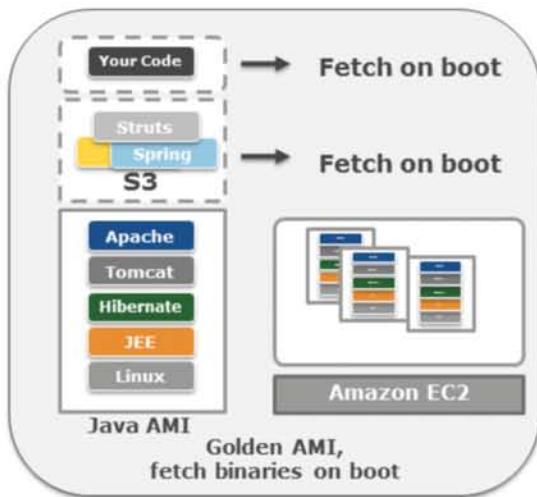


Notes:

With a fully-functional AMI, everything that your app needs is installed and ready to go as the instance spins up.

Even in the fully functioning work, it may still be worth having "all pending updates" applied as part of bootstrap. That means at day 0, there are no updates, but day 60 may have some new updates, we can then re-make a master MAI with new updates applied, but not have running instances without these updates installed. So deploy time increases over time until we re-master the AMI.

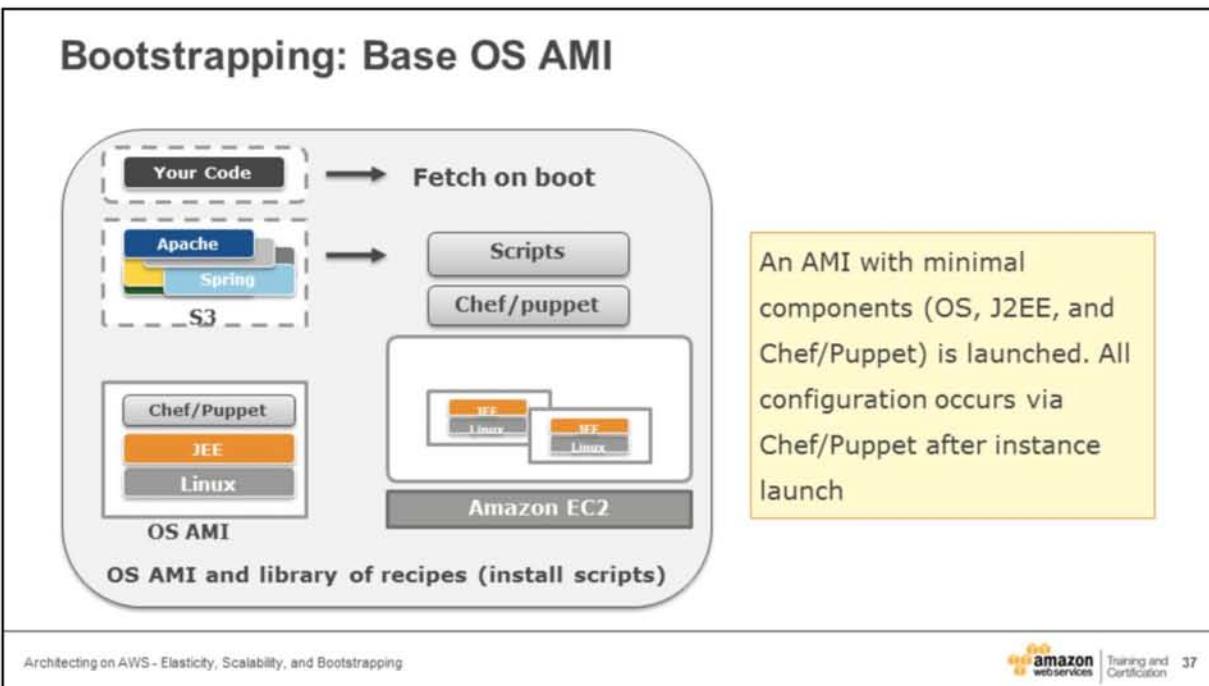
Bootstrapping: Partially-configured AMI



A “**Golden Image**” is launched, with scripts fetching/installing app code and other supporting components on boot

Notes:

With a partially configured AMI, you launch a “golden image” of the system, with scripts fetching and installing additional components on boot.



Notes:

An AMI configured with code has only the base OS and other minimal components. Tools like Chef or Puppet do the rest of the configuration after the instance launches.

Topics

- 💡 Basic tenets of AWS
- 💡 Patterns and (anti-patterns) for creating scalable architectures in AWS
- 💡 Bootstrapping EC2 Instances
- 💡 Building with CloudFormation
- 💡 Components of Auto Scaling

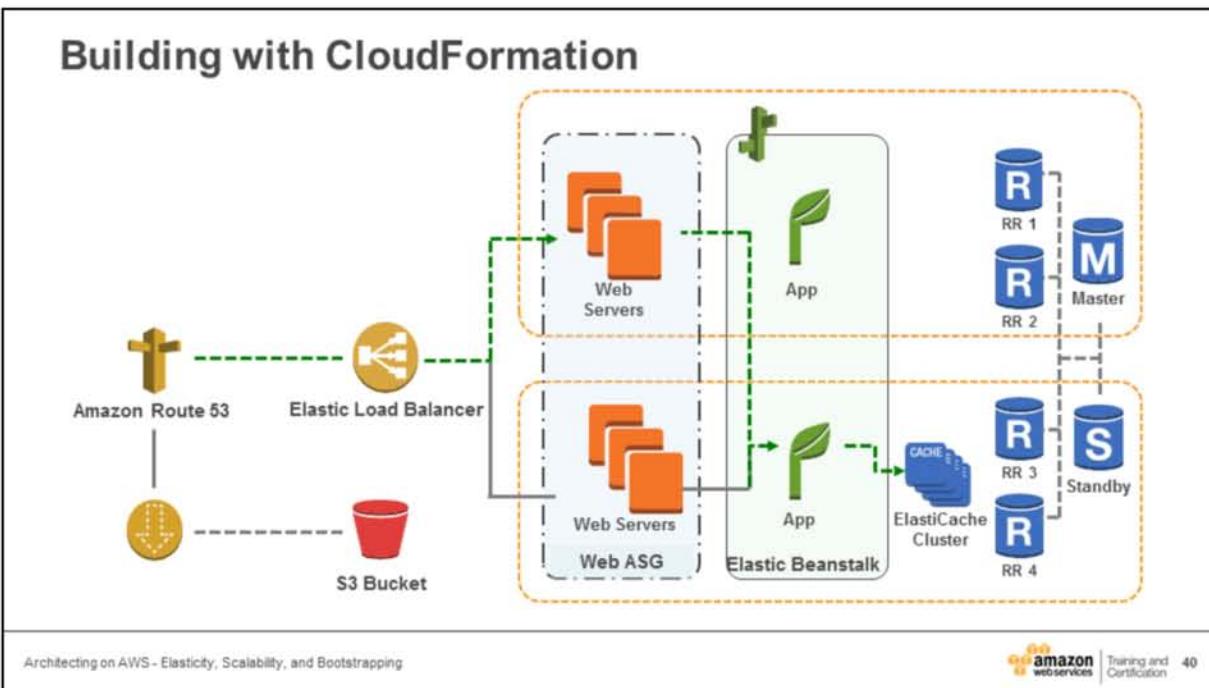
Building with CloudFormation

- Infrastructure as code, suitable for change management in version control (git, svn, and so on)
- Define an entire application stack (all resources required for your application) in a JSON template file
- Define runtime parameters for a template (EC2 Instance Size, EC2 Key Pair, and so on)
- Generate templates from running environments with CloudFormer

Notes:

CloudFormation basically is infrastructure as code. It's a set of instructions that not only include how to spin up EC2 instances, but the entire application stack.

A common question is how to get started using CloudFormation—especially if you already have a functional system running in AWS. There's a tool, CloudFormer, that can help: <http://aws.amazon.com/developertools/6460180344805680>.

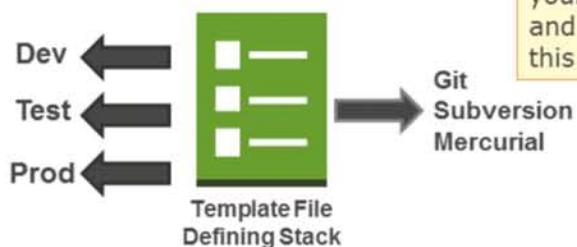


Notes:

How does CloudFormation work? Let's take a look at a system built in AWS. This entire system is considered the stack.

Building with CloudFormation (continue)

Build out multiple environments, such as for Development, Test, and Production using the template



Use the version control system of your choice to store and track changes to this template

The entire application can be represented in an AWS CloudFormation template.

Notes:

A quick review of CloudFormation basic before jumping into the template anatomy.

1. The entire application can be represented in an AWS CloudFormation template.
2. You can use the version control system of your choice to store and track changes to this template.
3. You can use the template to quickly build out multiple environments, such as for Development, Test, and Production.

CloudFormation Template Anatomy

```
{  
    "Description" : "Create an EC2 instance.",  
    "Resources" : {  
        "Ec2Instance" : {  
            "Type" : "AWS::EC2::Instance",  
            "Properties" : {  
                "KeyName" : "my-key-pair",  
                "ImageId" : "ami-75g0061f",  
                "InstanceType" : "m1.medium"  
            }  
        }  
    }  
}
```

Architecting on AWS - Elasticity, Scalability, and Bootstrapping



Training and
Certification 42

Notes:

Here's a basic CloudFormation template.

CloudFormation Template Anatomy: Resources

```
{  
    "Description" : "Create an EC2 instance.",  
    "Resources" : {  
        "Ec2Instance" : {  
            "Type" : "AWS::EC2::Instance",  
            "Properties" : {  
                "KeyName" : "my-key-pair",  
                "ImageId" : "ami-75g0061f",  
                "InstanceType" : "m1.medium"  
            }  
        }  
    }  
}
```

Resources are the AWS things you want to create

Notes:

Each template has a Resources property. This property contains the AWS resources that you want to create.

CloudFormation Template Anatomy: Resource name

```
{  
    "Description" : "Create an EC2 instance.",  
    "Resources" : {  
        "Ec2Instance" : {  
            "Type" : "AWS::EC2::Instance",  
            "Properties" : {  
                "KeyName" : "my-key-pair",  
                "ImageId" : "ami-75g0061f",  
                "InstanceType" : "m1.medium"  
            }  
        }  
    }  
}
```

Logical resource
name, anything
you like

Notes:

Each resource can have a logical name, which can be almost anything.

CloudFormation Template Anatomy: Resource type

```
{  
    "Description" : "Create an EC2 instance.",  
    "Resources" : {  
        "Ec2Instance" : {  
            "Type" : "AWS::EC2::Instance",  
            "Properties" : {  
                "KeyName" : "my-key-pair",  
                "ImageId" : "ami-75g0061f",  
                "InstanceType" : "m1.medium"  
            }  
        }  
    }  
}
```

The type of the
resource to create

Notes:

Each resource has a type property, which specifies the type of resource to create.

CloudFormation Template Anatomy: Properties

```
{  
    "Description" : "Create an EC2 instance.",  
    "Resources" : {  
        "Ec2Instance" : {  
            "Type" : "AWS::EC2::Instance",  
            "Properties" : {  
                "KeyName" : "my-key-pair",  
                "ImageId" : "ami-75g0061f",  
                "InstanceType" : "m1.medium"  
            }  
        }  
    }  
}
```

Properties define how CloudFormation will call the ec2-run-instance API

Architecting on AWS - Elasticity, Scalability, and Bootstrapping

 Amazon
Training and Certification 46

Notes:

Resources also have properties, which define how CloudFormation calls the ec2-run-instance API. (This is the API that controls spinning up new EC2 instances.)

CloudFormation Template Anatomy: Key-pair

```
{  
    "Description" : "Create an EC2 instance.",  
    "Resources" : {  
        "Ec2Instance" : {  
            "Type" : "AWS::EC2::Instance",  
            "Properties" : {  
                "KeyName" : "my-key-pair",  
                "ImageId" : "ami-75g0061f",  
                "InstanceType" : "m1.medium"  
            }  
        }  
    }  
}
```

We should allow the user of this template to specify her own EC2 Key Pair rather than hard-coding it.

Notes:

Notice that, in this template, we hard-coded the EC2 Key pair. In reality, it's probably better to let the user of the template specify the keypair instead.

CloudFormation Template Anatomy: Input

```
{  
    "Description" : "Create an EC2 instance.",  
    "Parameters" : {  
        "UserKeyName" : {  
            "Description" : "The EC2 Key Pair to allow SSH access to the instance",  
            "Type" : "String"  
        }  
    },  
    "Resources" : {  
        "Ec2Instance" : {  
            "Type" : "AWS::EC2::Instance",  
            "Properties" : {  
                "KeyName" : { "Ref" : "UserKeyName" },  
                "ImageId" : "ami-75g0061f",  
                "InstanceType" : "m1.medium"  
            }  
        }  
    }  
}
```

Parameters allow user of template to provide input

Notes:

We can allow the user to specify information (such as the keypair), by adding a parameters property to the CloudFormation template.

CloudFormation Template Anatomy: Reference a parameter

```
{  
    "Description" : "Create an EC2 instance.",  
    "Parameters" : {  
        "UserKeyName" : {  
            "Description" : "The EC2 Key Pair to allow SSH access to the instance",  
            "Type" : "String"  
        }  
    },  
    "Resources" : {  
        "Ec2Instance" : {  
            "Type" : "AWS::EC2::Instance",  
            "Properties" : {  
                "KeyName" : { "Ref" : "UserKeyName"},  
                "ImageId" : "ami-75g0061f",  
                "InstanceType" : "m1.medium"  
            }  
        }  
    }  
}
```

The **KeyName** property
now references the
UserKeyName parameter

Notes:

We can then refer to the Parameter in the properties of the resource.

CloudFormation Template Anatomy: Instance type

```
{  
    "Description" : "Create an EC2 instance.",  
    "Parameters" : {  
        "UserKeyName" : {  
            "Description" : "The EC2 Key Pair to allow SSH access to the instance",  
            "Type" : "String"  
        }  
    },  
    "Resources" : {  
        "Ec2Instance" : {  
            "Type" : "AWS::EC2::Instance",  
            "Properties" : {  
                "KeyName" : { "Ref" : "UserKeyName" },  
                "ImageId" : "ami-75g0061f",  
                "InstanceType" : "ml.medium"  
            }  
        }  
    }  
}
```

Let the user choose
Instance Type, but with
some restrictions...

Notes:

Here's another example where we hard-coded information into the template—in this case, the instance type. Let's use the parameters property to let the user specify the instance type instead. This time, however, let's add some restrictions, so we can keep users from spinning up the really large systems whenever they want.

CloudFormation Template Anatomy: Instance type (continue)

```
{  
    "Description" : "Create an EC2 instance.",  
    "Parameters" : {  
        "UserKeyName" : {  
            "Description" : "The EC2 Key Pair to allow SSH access to the instance",  
            "Type" : "String"  
        },  
        "InstanceType" : {  
            "Description" : "The EC2 Instance Type to launch.",  
            "Type" : "String",  
            "AllowedValues" : ["t1.micro", "m1.small", "m1.medium"]  
        }  
    },  
    "Resources" : {  
        "Ec2Instance" : {  
            "Type" : "AWS::EC2::Instance",  
            "Properties" : {  
                "KeyName" : { "Ref" : "UserKeyName"},  
                "ImageId" : "ami-75g0061f",  
                "InstanceType" : { "Ref" : "InstanceType" }  
            }  
        }  
    }  
}
```

The `InstanceType` parameter defines the `AllowedValues` the user may provide when creating a stack from this template

Notes:

Here, we have a new parameter, `InstanceType`. Notice that, unlike our `UserKeyName` parameter, we've added an `AllowedValues` property. This property contains the types of EC2 instances the user can select.

CloudFormation Template Anatomy: Outputs

```
{  
    "Description" : "Create an EC2 instance.",  
    ...  
},  
"Resources" : {  
    "Ec2Instance" : {  
        "Type" : "AWS::EC2::Instance",  
        "Properties" : {  
            "KeyName" : { "Ref" : "UserKeyName"},  
            "ImageId" : "ami-75g0061f",  
            "InstanceType" : { "Ref" : "InstanceType" }  
        }  
    }  
},  
"Outputs" : {  
    "InstancePublicDnsName" : {  
        "Description" : "The public DNS name of the newly created EC2 instance",  
        "Value" : { "Fn::GetAtt" : [ "Ec2Instance", "PublicDnsName" ] }  
    }  
}
```

Finally, we want to know the public DNS of the EC2 Instance that CloudFormation creates. Outputs allow us to output information about the resources created in the template

Notes:

The last thing we want to do is figure out the public DNS of the EC2 instance that the CloudFormation template creates.

We can get this information by adding an Outputs property to the template. As the name implies, this property allows us to output information about the resources created in the template.

CloudFormation Template Anatomy: Public DNS

```
{  
    "Description" : "Create an EC2 instance.",  
    ...  
},  
"Resources" : {  
    "Ec2Instance" : {  
        "Type" : "AWS::EC2::Instance",  
        "Properties" : {  
            "KeyName" : { "Ref" : "UserKeyName"},  
            "ImageId" : "ami-75g0061f",  
            "InstanceType" : { "Ref" : "InstanceType" }  
        }  
    }  
},  
"Outputs" : {  
    "InstancePublicDnsName" : {  
        "Description" : "The public DNS name of the newly created EC2 instance",  
        "Value" : { "Fn::GetAtt" : [ "Ec2Instance", "PublicDnsName" ] }  
    }  
}
```

Fn::GetAtt allows us to retrieve the 'PublicDnsName' attribute of the 'Ec2Instance' resource

Notes:

In this case, we use the command Fn::GetAtt to get the PublicDnsName of the instance.

CloudFormation Template Anatomy: UserData

```
...
"UserData": {
    "Fn::Base64": {
        "Fn::Join": [
            "",
            [
                "#!/bin/bash -ex\n",
                "yum -y install git-core\n",
                "yum -y install php-pear\n",
                "pear install Crypt_HMAC2-1.0.0\n",
                "pear install HTTP_Request-1.4.4\n",
                "pear install aws/sdk\n",

```

Bootstrap with User Data

Notes:

CloudFormation templates can be really flexible! In this example, we've added UserData content that we talked about earlier.

CloudFormation Template Anatomy: Re-use templates

```
...  
"AppDatabase": {"Type": "AWS::CloudFormation::Stack",  
"Metadata": { ... },  
"Properties": {  
    "TemplateURL": {  
        "Fn::Join": [  
            "/",  
            [  
                { ... },  
                "RDS_SQL_55.template"  
            ]  
        ]  
    },  
},  
}  
]  
}
```

Embed and re-use templates

Notes:

You can also include CloudFormation templates within other CloudFormation templates.

Building with CloudFormation: Metadata and cfn-init

- CloudFormation Metadata and cfn-init
- Declare metadata dynamically configure instances
- Works on Linux and Windows Server instances

```
"Ec2Instance": {  
    "Metadata": {  
        "AWS::CloudFormation::Init": {  
            "config": {  
                "sources" : {  
                    "/usr/local/bin/s3cmd" : "https://github.com/s3tools/s3cmd"  
                },  
                "packages": {  
                    "yum": { "git": [] }  
                }  
            }  
        }  
    }  
}
```

Define Instance Configuration

Notes:

CloudFormation also uses a helper script, called cfn-init. This script reads metadata from the template and acts on it.

Here's an example of using cfn-init to define the configuration of the instance.

You can learn more here:

<http://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/cfn-init.html>.

Topics

- 💡 Basic tenets of AWS
- 💡 Patterns and (anti-patterns) for creating scalable architectures in AWS
- 💡 Bootstrapping EC2 Instances
- 💡 Building with CloudFormation
- 💡 Components of Auto Scaling

Notes:

Another key part of building scalable systems with AWS is Auto Scaling. Let's look at the components of Auto Scaling now.

Auto Scaling

- 💡 Scale your Amazon EC2 capacity up or down automatically according to conditions you define
- 💡 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

Notes:

With Auto Scaling, you can scale your EC2 instances up and down automatically. You can also ensure that EC2 instances increase and decrease seamlessly.

Auto Scaling Components

💡 Components

- Launch configuration
- Group
- Scaling policy (optional)
- Scheduled action (optional)

💡 Auto Scaling Launch Configuration

- Define how Auto Scaling should launch your EC2 instances
- Similar to ec2-run-instances API

Notes:

Auto Scaling basically consists of four components. One of the first things you should consider when you create an Auto Scaling configuration is how it launches EC2 instances.

Auto Scaling: Configure through EC2 Dashboard

The screenshot shows the AWS EC2 Dashboard. On the left, there is a navigation menu with several sections: EC2 Dashboard, Events, Tags, Reports, Instances (with sub-options like Instances, Spot Requests, Reserved Instances), Images (AMIs, Bundle Tasks), Elastic Block Store (Volumes, Snapshots), Network & Security (Security Groups, Elastic IPs, Placement Groups, Load Balancers, Key Pairs, Network Interfaces), and Auto Scaling (Launch Configurations, Auto Scaling Groups). The 'Auto Scaling Groups' option is highlighted with an orange box. The main content area is titled 'Welcome to Auto Scaling' and contains a note about managing EC2 capacity automatically. It features three benefit cards: 'Reusable Instance Templates' (provision instances based on a reusable template), 'Automated Provisioning' (keep your Auto Scaling group healthy and balanced), and 'Adjustable Capacity' (maintain a fixed group size or adjust dynamically based on CloudWatch metrics). A 'Create Auto Scaling group' button is also present.

Architecting on AWS - Elasticity, Scalability, and Bootstrapping

amazon web services Training and Certification 60

Notes:

You can configure Auto Scaling through the EC2 dashboard, as shown here.

Auto Scaling: Create launch configuration

Create Launch Configuration

An AMI is a template that contains the software configuration (operating system, application server, and applications) required to launch your instance. You can select an AMI provided by AWS, our user community, or the AWS Marketplace; or you can select one of your own AMIs.

Quick Start

My AMIs

Amazon Linux **Amazon Linux AMI 2014.03.1** - ami-043a5034 (64-bit) / ami-1e3a502e (32-bit)

Free tier eligible

Red Hat Enterprise Linux 6.4 (PV) - ami-bfa63b88 (64-bit) / ami-baa63b8a (32-bit)

Free tier eligible

SuSE Linux Enterprise Server 11 sp3 (PV) - ami-dfb429e8 (64-bit) / ami-9eb429ee (32-bit)

Free tier eligible

Ubuntu Server 12.04 LTS (PV) - ami-fa9cfc1ca (64-bit) / ami-ff9cfc1c8 (32-bit)

Free tier eligible

Ubuntu Server 12.10 (PV) - ami-7eaec4e (64-bit) / ami-7caeccc4c (32-bit)

Free tier eligible

Amazon Linux AMI (HVM) 2014.03.1 - ami-3031a000

Cancel and Exit

Select

* 64-bit 32-bit

Architecting on AWS - Elasticity, Scalability, and Bootstrapping

Training and Certification 61

Notes:

Creating a launch configuration is almost identical to launching an EC2 instance. You specify options for the AMI, storage, tags, and so on.

Auto Scaling: Create launch configuration (continue)

The screenshot shows the 'Create Launch Configuration' wizard in the AWS Management Console. The current step is 'Configure details'. The 'Name' field is filled with 'Web launch configuration'. Under 'Monitoring', the 'Enable CloudWatch detailed monitoring' checkbox is checked. In the 'User data' section, a shell script is pasted into the text area:

```
#!/bin/bash
yum -y install httpd php mysql php-mysql
chkconfig httpd on
/etc/init.d/httpd start
cd /tmp
```

Under 'IP Address Type', the radio button for 'Only assign a public IP address to instances launched in the default VPC and subnet (default)' is selected. A note below states: 'Note: this option only affects instances launched into an Amazon VPC'.

At the bottom left, it says 'Architecting on AWS - Elasticity, Scalability, and Bootstrapping'. At the bottom right, there is an 'Amazon Web Services' logo and 'Training and Certification' text.

Notes:

Auto Scaling configurations can also take bootstrapping scripts and configuration instructions, just like EC2 instances.

Notice also that you need to give every launch configuration that you create a name.

Auto Scaling: Auto Scaling Group

1. Configure Auto Scaling group details 2. Configure scaling policies 3. Configure Notifications 4. Review

Create Auto Scaling Group

Launch Configuration: Web launch configuration
Group name: web-group
Group size: Start with 1 instances
Network: vpc-310cb53 (172.31.0.0/16) (default)
Subnet: subnet-b64299d4(172.31.32.0/20) Default in us-west-2a
subnet-141b3160(172.31.16.0/20) Default in us-west-2a
Create new VPC
Create new subnet
Each instance in this Auto Scaling group will be assigned a public IP address.
Advanced Details
Load Balancing: Receive traffic from Elastic Load Balancer(s)
auto-scaling-wb-x
Health Check Type: ELB EC2
Health Check Grace Period: 300 seconds
Monitoring: Enable CloudWatch detailed monitoring
Learn more

Define where (AZs or VPC Subnets) Auto Scaling should launch instances

Optionally, select ELB(s) to register with

Architecting on AWS - Elasticity, Scalability, and Bootstrapping

amazon web services Training and Certification 63

Notes:

Another component of Auto Scaling is the Auto Scaling group. This group defines the Availability Zones and VPC subnets into which Auto Scaling should launch instances. You also have the option of specifying Amazon Elastic Load Balancers, tags, and other information.

You can create an Auto Scaling group using the AWS Management Console. With an auto scaling group, you define:

- The name of the group
- The launch configuration you want to use
- The size of the auto scaling group
- Whether to launch the Auto Scaling group within a VPC or not
- How to load balance between auto scaling group instances
- How to monitor the health of instances in an auto scaling group

Auto Scaling: Scaling policies

1. Configure Auto Scaling group details 2. Configure scaling policies 3. Configure Notifications 4. Review

Create Auto Scaling Group

You can optionally add scaling policies if you want to adjust the size (number of instances) of your group automatically. A scaling policy is a set of instructions for making such adjustments in response to an Amazon CloudWatch alarm that you assign to it. In each policy, you can choose to add or remove a specific number of instances or a percentage of the existing group size, or you can set the group to an exact size. When the alarm triggers, it will execute the policy and adjust the size of your group accordingly. [Learn more about scaling policies.](#)

Keep this group at its initial size
 Use scaling policies to adjust the capacity of this group

Scale between **1** and **4** instances. These will be the minimum and maximum size of your group.

Increase Group Size

Name: Increase Group Size
Execute policy when: High CPU usage [Edit Remote](#)
Breaches the alarm threshold: CPUUtilization >= 80 for 300 seconds
for the metric dimensions AutoScalingGroupName = web-group
Take the action: Add - 1 instances
And then wait: 300 seconds before allowing another scaling activity

Decrease Group Size

Name: Decrease Group Size
Execute policy when: No alarm selected [Add new alarm](#)

You can establish auto scaling policies based on CloudWatch alarms

Architecting on AWS - Elasticity, Scalability, and Bootstrapping

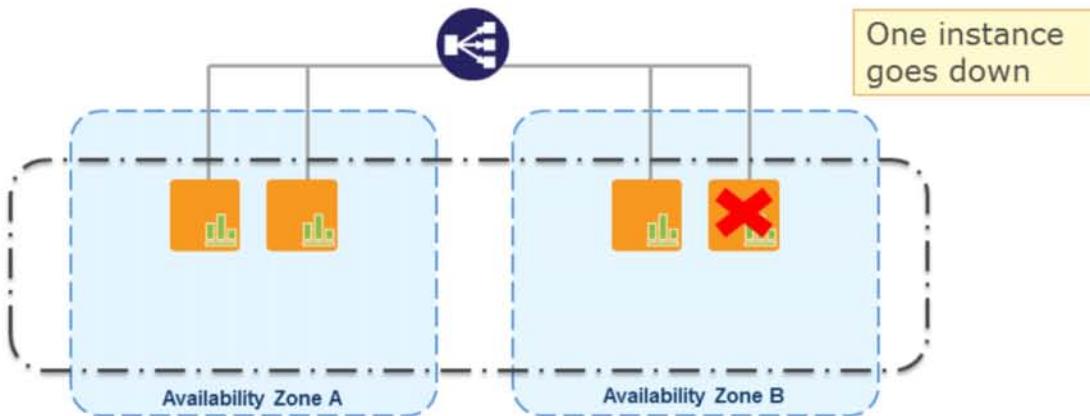
amazon web services | Training and Certification 64

Notes:

In addition, you can also establish auto scaling policies, which dynamic increase or decrease EC2 instances based on CloudWatch alarms.

Note that it's always a good idea to ensure that your auto scaling group handles both scaling UP (adding instances) and scaling DOWN (removing instances). This is one way you can help maximize your IT budget—by adding and removing instances when needed.

Auto Scaling Group in working (1 of 4)



Architecting on AWS - Elasticity, Scalability, and Bootstrapping

Amazon Web Services | Training and Certification 65

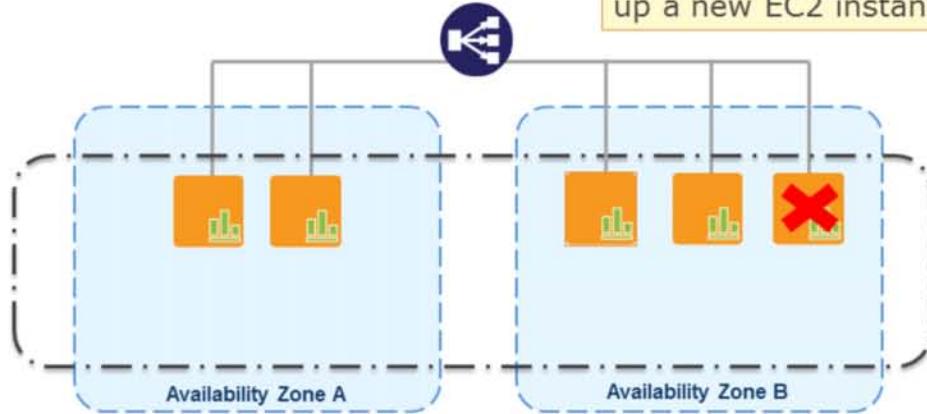
Notes:

Here's what this Auto Scaling group looks like. Notice that we have four instances (as we specified in the desired-capacity parameter) running across two Availability Zones, all registered with one ELB.

If one instance goes down...

Auto Scaling Group in working (2 of 4)

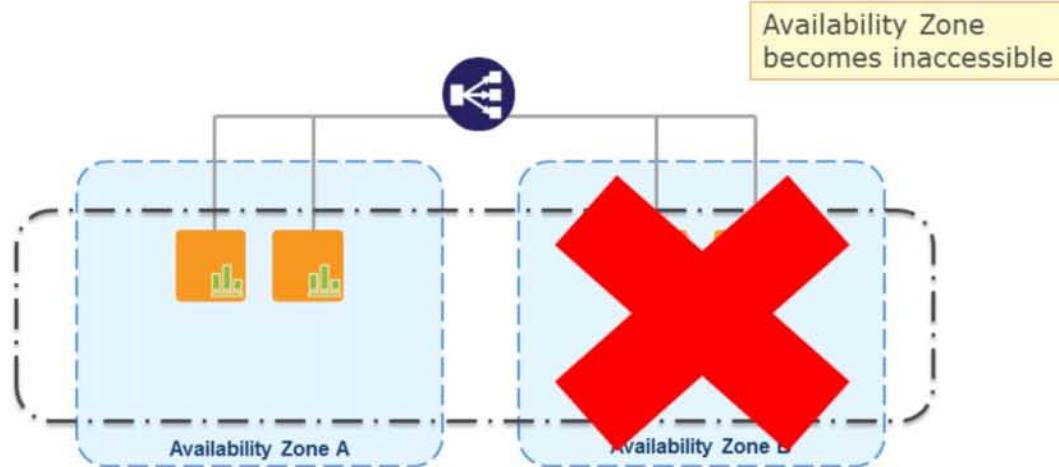
The Auto Scaling group spins up a new EC2 instance



Notes:

The Auto Scaling group spins up a new one.

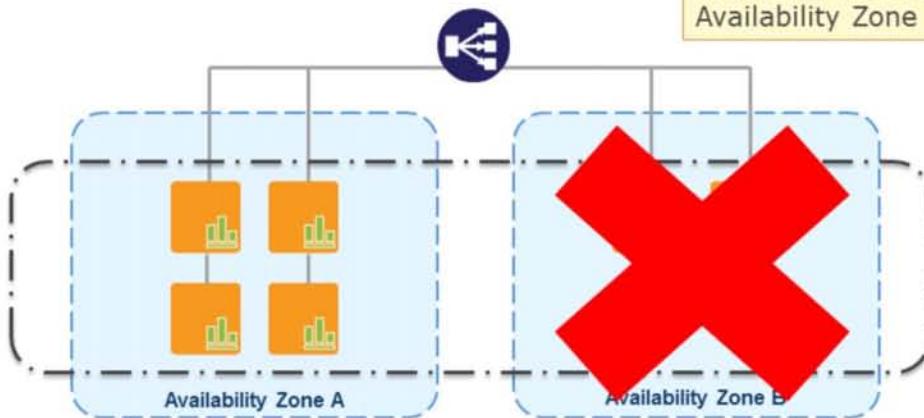
Auto Scaling Group in working (3 of 4)



Notes:

If the instances in one Availability Zone aren't accessible...

Auto Scaling Group in working (4 of 4)



Architecting on AWS - Elasticity, Scalability, and Bootstrapping

Amazon web services | Training and Certification 68

Notes:

New instances are started in the other Availability Zone.

Auto Scaling Policy

The screenshot shows the AWS Auto Scaling Policy configuration interface. It displays two policy definitions:

- Increase Group Size:** Triggered by a High CPU usage alarm (CPUUtilization >= 60 for 300 seconds). Action: Add 1 instance. Wait: 300 seconds.
- Decrease Group Size:** Triggered by No alarm selected. Action: Remove 0 instances. Wait: 300 seconds.

A yellow callout box highlights the text: "Define scaling actions that describes how many EC2 instances to add or remove to a Group".

Architecting on AWS - Elasticity, Scalability, and Bootstrapping

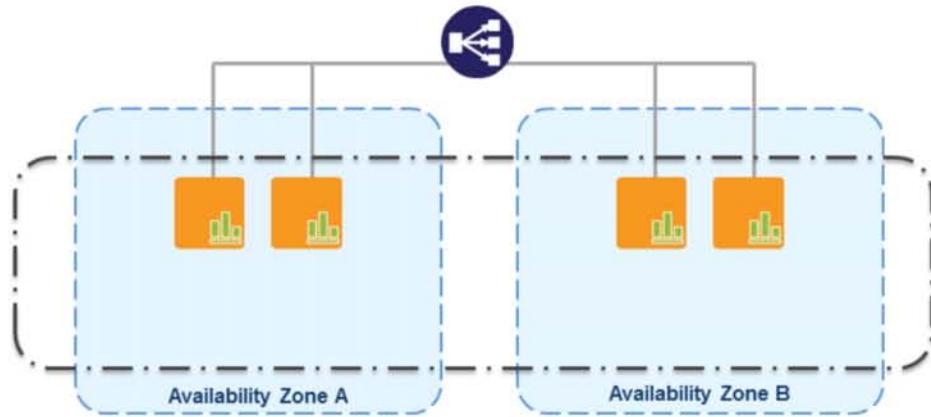
amazon web services | Training and Certification 69

Notes:

An Auto Scaling policy defines how to add and remove EC2 instances from an Auto Scaling group. You can create an auto scaling policy at any time—whether it's when you first create the auto scaling group, or later on. Notice that the UI encourages you to create policies in pairs—one to determine when to scale out, and another to determine when to scale in. This is important—you should always consider both aspects of auto scaling when you create an auto scaling policy.

Auto Scaling Policy: Remove servers

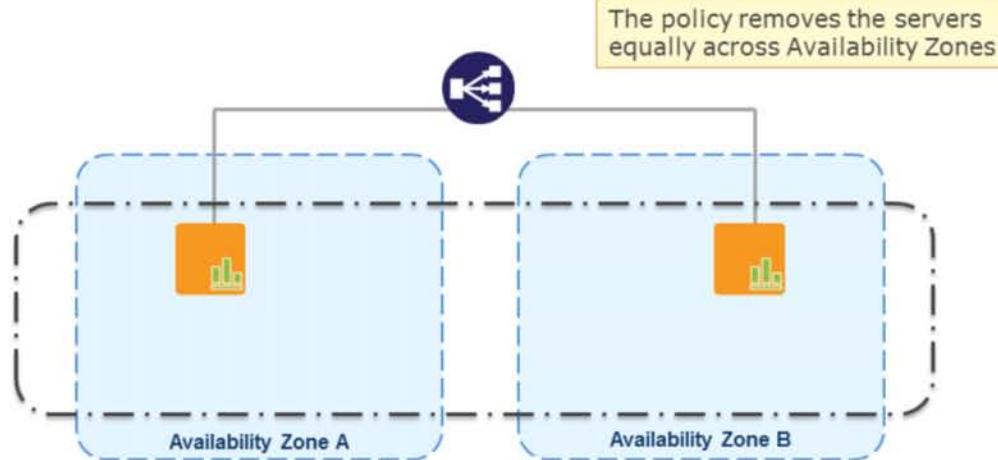
```
as-execute-policy web-remove-two-servers
```



Notes:

For example, here's a policy that removes two servers from the group.

Auto Scaling Policy: Remove servers (continue)

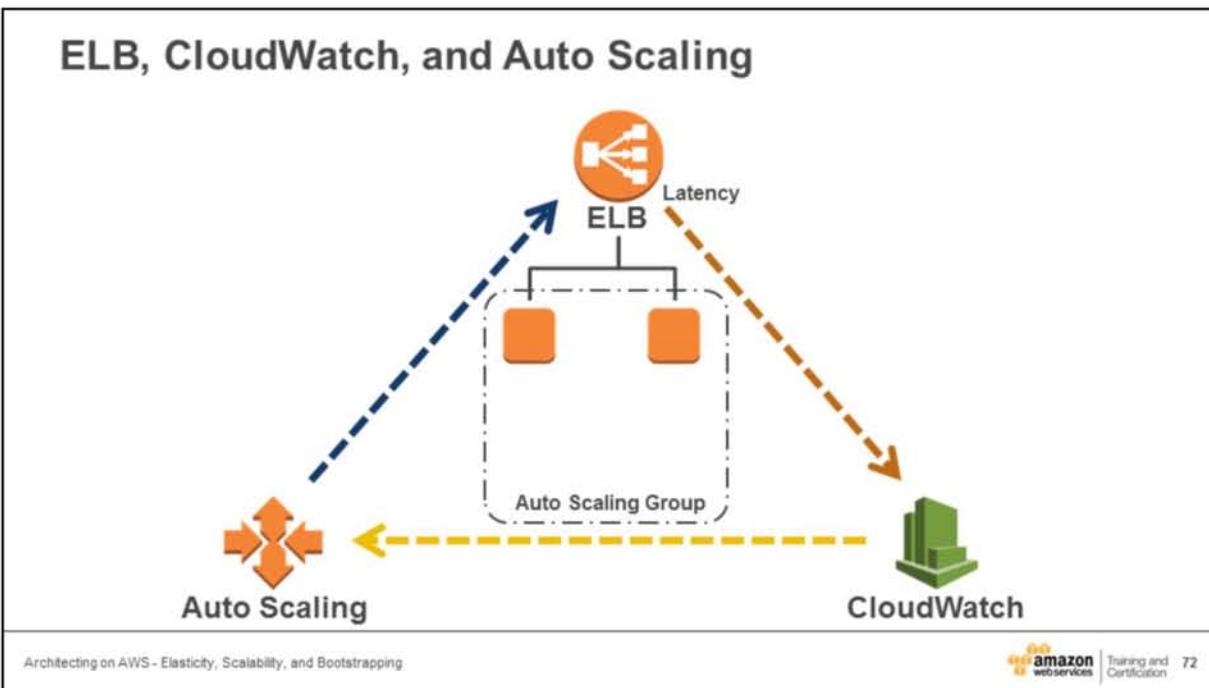


Architecting on AWS - Elasticity, Scalability, and Bootstrapping

Amazon
Training and
Certification 71

Notes:

Notice that the policy removes the servers equally across Availability Zones (in this case), one from each zone.



Notes:

All of these service work well individually, but together they become more powerful and increase the control and flexibility our customers demand.

Module review

- 💡 What are the four patterns for building scalable architectures?
- 💡 Can you give examples of bootstrapping an instance?
- 💡 How do you get the metadata for any given instance?
- 💡 How do you build a basic CloudFormation template?
- 💡 What are the four components of auto scaling?

Answers:

Four patterns for building scalable architectures are:

- Automated scaling processes
- Loosely-coupled
- Stateless
- Horizontal scaling

Instance bootstrapping examples are:

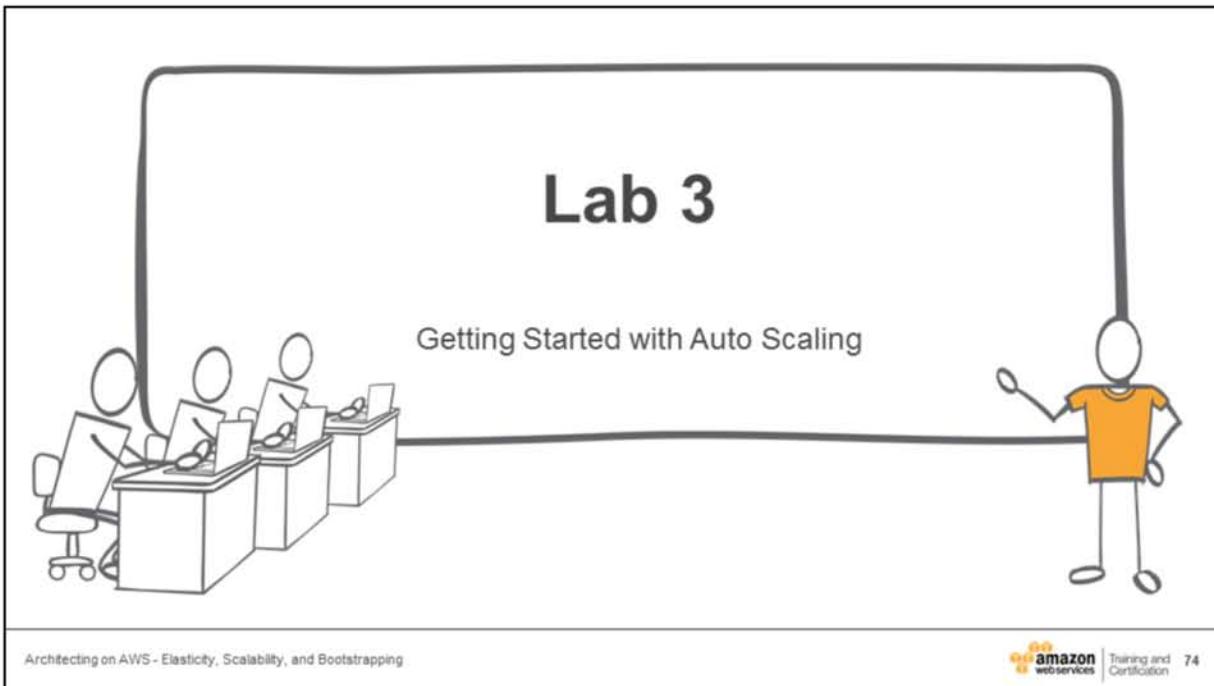
- Install latest software
- Copy data from S3
- Register with DNS
- Start services
- Update packages
- Reboot
- Open port 80
- Register with load balancer
- Mount devices

You can get the metadata via HTTP at → <http://169.254.169.254/latest/meta-data/>

Four components of auto scaling are:

1. Launch configuration
2. Group

3. Scaling policy (optional)
4. Scheduled action (optional)



Architecting on AWS - Elasticity, Scalability, and Bootstrapping

amazon web services | Training and Certification 74

Notes:

Approximate timing: 1 hour 30 minutes

Copyright © 2013, 2014 Amazon Web Services, Inc. and its affiliates. All rights reserved.

This work may not be reproduced or redistributed, in whole or in part, without prior written permission from Amazon Web Services, Inc. Commercial copying, lending, or selling is prohibited.

Errors or corrections? Email us at aws-course-feedback@amazon.com.
Other questions? Email us at aws-training-info@amazon.com.

All trademarks are the property of their owners.

Appendix: Auto Scaling

The following slides are intended for when you need to go into more detail on Auto Scaling. For more information, please visit: <http://aws.amazon.com/autoscaling/>

Auto Scaling

- 💡 Automatically Scale Server Farms
 - Scale out and in
 - (Re)Balance Across AZs
 - Add/Remove from ELB if applicable
- 💡 Set a Thermostat
 - Do not manage the furnace burners

Types of Auto Scaling

Manual

- Send an API call or use CLI to launch/terminate instances
- Only need to specify capacity change (+/-)

By Schedule

- Scale up/down based on date and time

By Policy

- Scale in response to changing conditions, based on user configured real-time monitoring and alerts

Automatic Rebalance

- Instances are automatically launched/terminated to ensure the application is balanced across multiple AZs

Auto Scaling Components

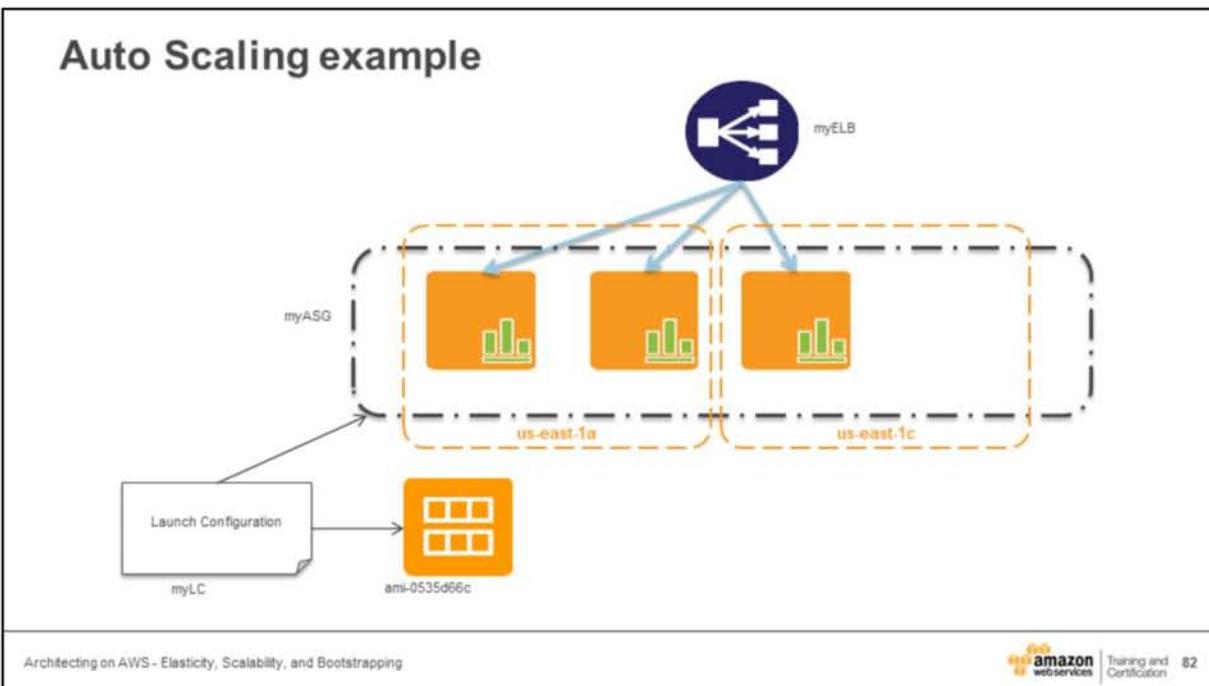
- 💡 Launch Configuration
- 💡 Auto Scaling Group
- 💡 Auto Scaling Policy
- 💡 CloudWatch Alarms

Auto Scaling Launch Configuration

- Describes what Auto Scaling will create when adding instances
 - Name (MyLC)
 - AMI (ami-0535d66c)
 - Instance Type (m1.medium)
 - Security Group (SSH, Web, aws-elb-sg)
 - Instance Key Pair (myKeyPair)
- Only one active launch configuration at a time
- Auto Scaling will terminate instances with old launch configurations first
 - Rolling software updates

Auto Scaling Group

- ─ Auto Scaling managed grouping of EC2 instances
- ─ Automatic health check to maintain pool size
- ─ Automatically scale the number of instances by policy
 - Min, Max, Desired (how many initially)
- ─ Automatic Integration with ELB
- ─ Automatic Integration with AZs
 - Automatic distribution & balancing across AZs

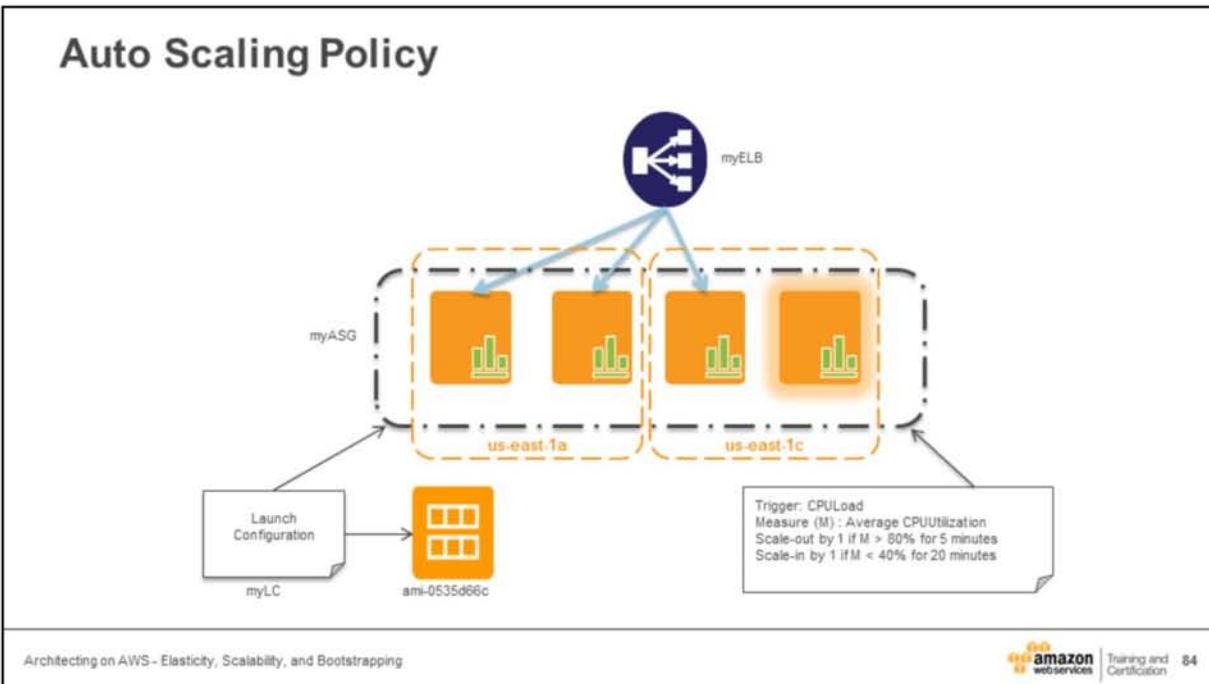


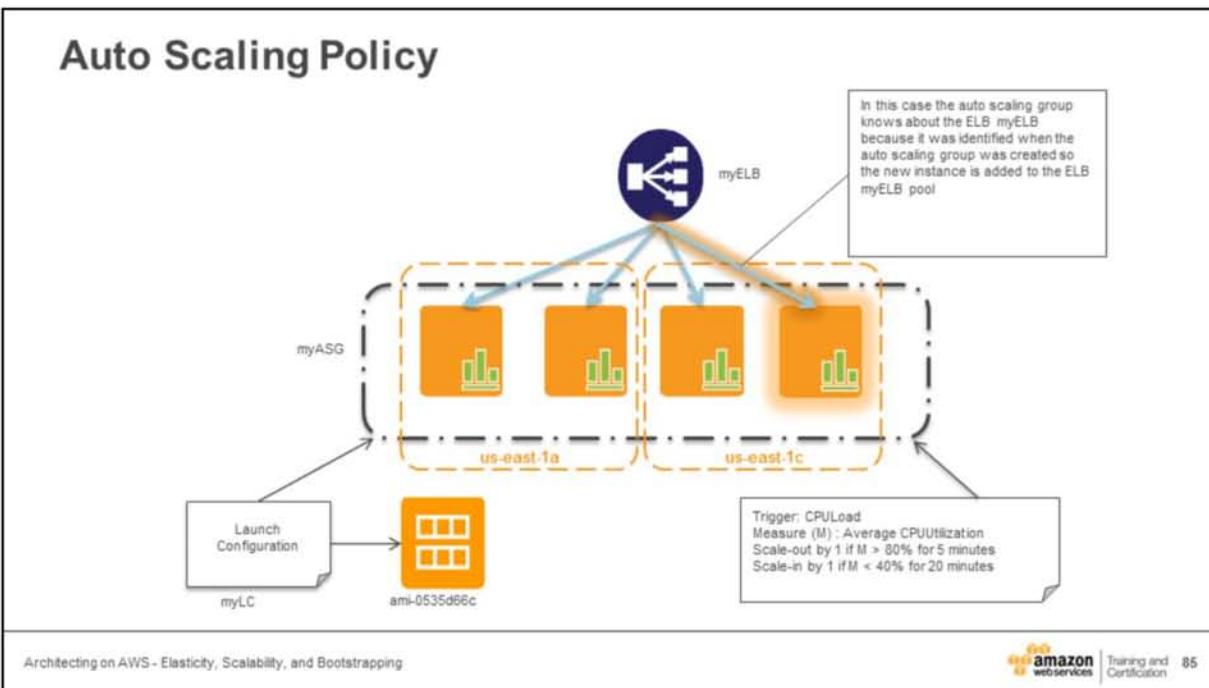
Auto Scaling Policy

- 💡 Parameters for performing an Auto Scaling action
 - Scale Up/Down
 - By how much
 - ChangeInCapacity (+/- #)
 - ExactCapacity (#)
 - ChangeInPercent (+/- %)
 - Cool Down (seconds)
- 💡 Policy can be triggered by CloudWatch Events

Trigger: CPULoad
Measure (M) : Average CPUUtilization
Scale-out by 1 if M > 80% for 5 minutes
Scale-in by 1 if M < 40% for 20 minutes

Auto Scaling Policy





Appendix: Bootstrapping

Architecting on AWS - Elasticity, Scalability, and Bootstrapping



More information available at:

<http://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/cfn-windows-stacks-bootstrapping.html>.

Bootstrapping (1 of 3)

- Bootstrapping is the process of automatically setting up your servers after they boot
- Auto scaling strategies must include proper bootstrapping of provisioned EC2 instances
 - AMI management
 - Software to install or configure (including rebooting)
 - Discovery or registration of new instances
- Low touch and as dynamic as possible is needed to meet high availability SLAs
- Graceful departure is just as important

Bootstrapping (2 of 3)

- 💡 A MySQL read replica fails (unresponsive, too slow, and so on)
- 💡 First, you have to detect it
 - Is the instance running at all (pingable)?
 - Issue a standard "Show Slave Status" MySQL command against the Read Replica and look at "Seconds_Behind_Master"
 - "Show Slave Status" is also published as an Amazon CloudWatch metric ("Replica Lag") available via the AWS Management Console or Amazon CloudWatch APIs
- 💡 CloudWatch Alarm to SNS, Monitoring Agents and a monitoring console, or a witness instance are some ways to detect a bad read replica
- 💡 After its found, a bad read replica can be deleted and replaced

Bootstrapping (3 of 3)

- 💡 So, the bad read replica is detected and replaced, but how does an app tier instance know about the new read replica?
- 💡 You can delete the bad read replica and create a new one with the same endpoint by using the same DB Instance Identifier and Source DB Instance Identifier as the deleted read replica. Automated or Manual through the AWS Console.
- 💡 If you deleted the bad read replica and replaced it with a new one with a different DB Instance Identifier and Source DB Instance Identifier
 - Automated - Something has to call the `DescribeDBInstance` API to retrieve the endpoint for the new read replica, then update the app tier instances(s) with the endpoint of the new read replica
 - Manual – Use the AWS Management Console to retrieve the endpoint for the new read replica and update the app tier



Module 8: Data Storage Scaling

Identify the correct statements

Multiple EBS Volumes may be attached to the same EC2 instance.

EC2 Instance Store (ephemeral volumes) are a local resource directly attached to the host that an EC2 instance runs in.

Because of its scalable and durable nature, S3 is a good candidate for hosting live database transaction log and data files.

CloudFront supports caching of static objects as well as RTMP and RTMPE streaming and HTTP progressive download.

EBS volumes have a maximum size of 2TB.

Sequential IO is a better fit for EBS volumes compared to EC2 Instance Store (ephemeral volumes).

Frequent EBS snapshots increase EBS volume durability.

Multiple EC2 instances with multiple EBS volumes can be pooled to create a multi-petabyte Distributed Network File System (DNFS)

Notes:

Before we begin, let's take a look at these statements. Which are true and which are false?

Row 1: True, True, False

Row 2: True, False (1TB), False

Row 3: True, True

CloudFront supports various protocols for streaming. To learn more about working with streaming distribution, refer to the online documentation → <http://docs.aws.amazon.com/AmazonCloudFront/2012-05-05/DeveloperGuide/WorkingWithStreamingDistributions.html>

Topics

- 💡 Data storage options
- 💡 Amazon EBS
- 💡 Instance storage
- 💡 Amazon S3 and Amazon CloudFront

Notes:

Here's what we'll cover in this module.

Topics

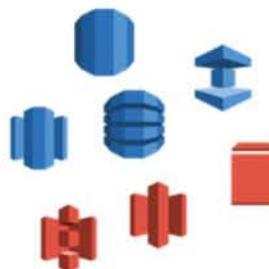
- 💡 Data storage options
 - AWS storage options
 - Best practices for data storage
- 💡 Amazon Elastic Block Storage (EBS)
- 💡 Instance storage
- 💡 Amazon S3 and Amazon CloudFront

Notes:

Specifically, we'll look at what options are available to you, and some best practices you should follow.

AWS Storage Options

- **Block Storage:** Instance Store, Amazon EBS
- **Object Storage:** Amazon S3, Amazon Glacier, Amazon CloudFront
- **Sync Volumes:** AWS Storage Gateway
- **Relational Databases:** Amazon RDS
- **NoSQL Databases:** Amazon DynamoDB
- **In-memory Cache:** ElastiCache
- **Content Cache:** Amazon CloudFront



Notes:

AWS offers many variations of data storage, from block storage to caching content.

Best practices on choosing the right data storage solution

- 💡 Understand the variety of storage options on AWS
- 💡 Access performance, durability, cost, and interface
- 💡 POSIX versus Object store—choose where appropriate
- 💡 Use multiple cloud storage options—storage hierarchy
- 💡 Horizontal versus Vertical scaling
- 💡 Be creative. Use storage alternatives like in-memory caches

It's all about performance-oriented and cost-oriented choice.

Notes:

AWS offers many variations of data storage, from block storage to caching content.

Topics

- >Data storage options
 - Amazon Elastic Block Storage (EBS)
 - Benefits of using Amazon EBS
 - Standard volumes versus Provisioned IOPS volumes
 - Amazon EBS pricing
 - Instance storage
 - Amazon S3 and Amazon CloudFront

Notes:

In this topic, we are going to discuss the benefits of using Amazon EBS. The difference between the standard volumes and the provisioned IOPS volumes as well as EBS pricing will be covered.

Benefits of using Amazon EBS

- Network attached, block storage
- Data lifetime independent of Amazon EC2 instance lifetime
- User-controlled snapshots
- Provisioned performance (up to 4,000 16k IOPS per volume)
- Large data storage capacity (up to 1TB per volume)
- Portability between instances (detach/attach)

Notes:

EBS offers a lot of advantages. Here are some of them. We'll look into each of these in more detail.

Network attached block storage

- 💡 Each volume is like a hard drive on a physical server
- 💡 Attach multiple volumes to an Amazon EC2 instance
- 💡 Volumes cannot be shared with multiple Amazon EC2 instances
- 💡 Ideal for:
 - OS boot device
 - File systems
 - Databases
 - Raw block devices

Notes:

EBS is popular in part because it's familiar. It's just like a hard drive attached to a physical server.

Amazon EBS performance

	Standard Amazon EBS	Provisioned IOPS Amazon EBS
IOPS	100 IOPS steady-state, with best-effort bursts to hundreds	Within 10% of up to 4,000 IOPS, 99.9% of the time, as provisioned
Throughput	Best effort to 10's of MB/sec	16 KB per IO = up to 64 MB/sec. It can increase in bursts on a best-effort basis.
Latency	Reads typically <20ms writes typically <10ms	Each IO has a service time of provisioned IOPS/s

- Use EBS-optimized Amazon EC2 instances when attaching Provisioned IOPS volume for dedicated network bandwidth
- Stripe multiple volumes for more IOPS (e.g., (10) x 4,000 IOPS volumes in RAID0 for 40,000 IOPS)

Notes:

EBS has two performance options: standard and Provisioned IOPS. Choose the option that best fits the needs of your system architecture.

To get 10% of PIOPS 99.9%, make sure you use EBS-optimized instances when attaching a provisioned IOPS volume.

Consider striping multiple volumes for more IOPS. (e.g., (10) x 4,000 IOPS volumes in RAID0 for 40,000 IOPS).

The following example usage scenarios work well with Amazon EBS standard volumes:

- File server
- Log processing
- Low-traffic websites
- Analytics
- System boot volume

The following example usage scenarios work well with Amazon EBS Provisioned IOPS volumes:

- Business applications
- Database workloads, such as:
 - MongoDB
 - Microsoft SQL Server
 - MySQL

- PostgreSQL
- Oracle

Amazon EBS pricing

	Standard	Provisioned IOPS
Storage	\$0.05 per GB-month	\$0.125 per GB-month
IOPS	\$0.05 per 1 million I/O requests	\$0.10 per provisioned IOPS-month
Snapshots	\$0.095 per GB-month of data stored	

Notes:

This is the EBS pricing in US East (N. Virginia) region as of **April, 2014**. As you can see, Provisioned IOPS offers more capability, but there are cost differences to consider. Let's take a look.

EBS cost analysis: Standard vs. Provisioned IOPS

- 💡 **Steady, predictable IO patterns:** cost effective to use Provisioned IOPS Amazon EBS
- 💡 **Bursty IO patterns:** analyze pattern and choose Amazon EBS type



Notes:

Standard EBS volumes deliver approximately 100 IOPS on average, with burst capability of up to hundreds of IOPS. Provisioned IOPS volumes are designed to offer consistent high performance.

Remember: Provisioned IOPS EBS is not always more expensive than Standard EBS volumes. Just because PIOPS has a higher monthly storage charge it won't necessarily be more expensive overall, depending on actual level of IO activity to the volume.

Amazon EBS: Native redundancy; optimized for random IO

- Replicated within single AZ
- 0.1% - 0.5% AFR using snapshots
- Optimized for random I/O
- Can be striped using RAID 0 or LVM

Notes:

EBS volumes have native redundancy—data is replicated across servers within a single Availability Zone. EBS volumes have an Annual Failure Rate (AFR) of between .1% and .5%

Volumes are optimized for random input/output use. You can stripe EBS volumes using RAID 0 or a Logical Volume Manager (LVM).

Amazon EBS snapshots

- 💡 Snapshots are stored in S3
- 💡 May be migrated across regions
- 💡 New volumes can be created from Amazon EBS Snapshots and placed in desired Availability Zone
- 💡 AMIs can be created from Amazon EBS Snapshots

Notes:

Amazon EBS provides the ability to back up point-in-time snapshots of your data to Amazon S3 for durable recovery.

Amazon EBS snapshots are incremental backups, meaning that only the blocks on the device that have changed since your last snapshot will be saved.

When you delete a snapshot, only the data not needed for any other snapshot is removed. Even if prior snapshots have been deleted, all active snapshots will contain all the information needed to restore the volume. In addition, the time to restore the volume is the same for all snapshots, offering the restore time of full backups with the space savings of incremental.

You can use snapshots to instantiate multiple new volumes, expand the size of a volume or move volumes across Availability Zones.

By optionally specifying a different volume size or a different Availability Zone, this functionality can be used as a way to increase the size of an existing volume or to create duplicate volumes in new Availability Zones.

Amazon EBS also provides the ability to copy snapshots across AWS regions, making it easier to leverage multiple AWS regions for geographical expansion, data center migration and disaster recovery.

AMIs can be created from EBS Snapshots and new instances can be created and placed in desired Availability Zone.

Amazon EBS capacity

Grow a volume:

1. Snapshot → Create new
2. Select larger volume from snapshot
3. Detach existing volume
4. Attaching the new volume

Allocate space proactively for root volumes to prevent out-of-space issues

1GB to 1TB per volume

Notes:

Be aware that when striping beyond 6-8 disks, the overhead of increased cost and management might not be worth the performance gains.

Split applications between different disk volumes will reduce cluttering of volumes. E.g. use the C drive for the operating system, the D drive for applications, the E drive for the pagefile, and the F drive for log files.

Reference

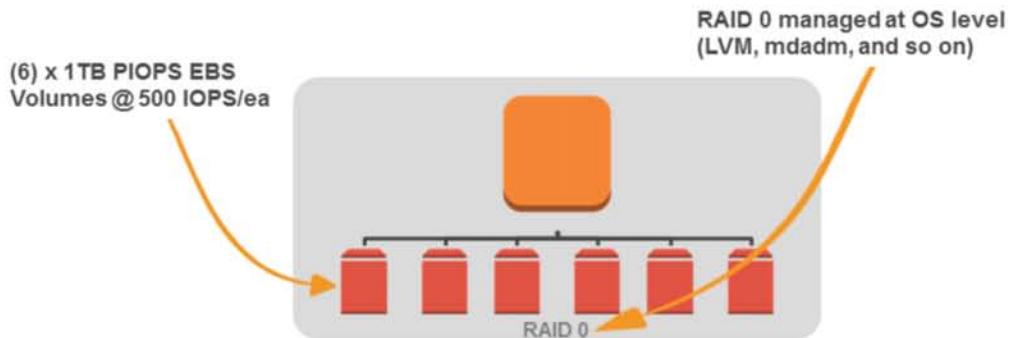
By default, each Amazon EC2 customer has a limit of:

- 5,000 total volumes per account
- 10000 total snapshots per account
- 20 TiB in total Standard volume storage
- 10,000 Provisioned IOPS or 20 TiB in total Provisioned IOPS volume storage (whichever is reached first) size

EBS volumes can range in size from 1GB to 1TB per volume.

Amazon EBS capacity scenario: 6TB block storage

- 6 TB of block storage at 3,000 IOPS on an EC2 instance

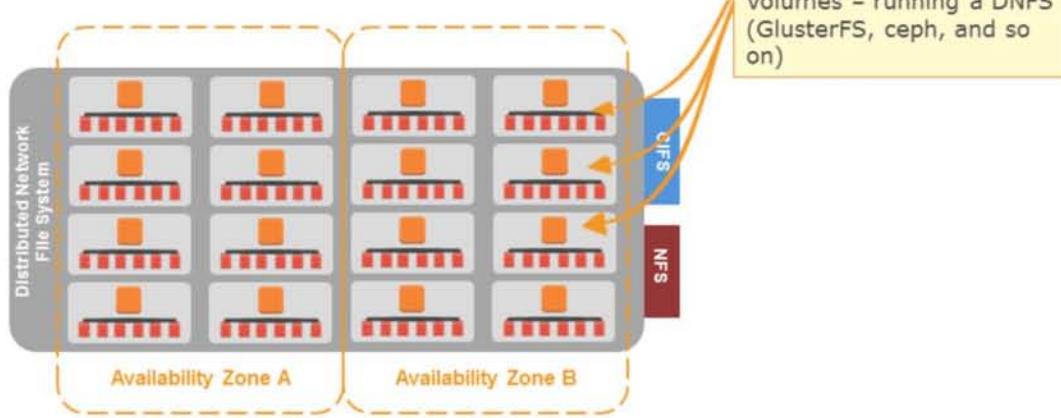


Notes:

Let's look at the following scenario: you need 6TB of block storage at 3,000 IOPS on EC2. To accomplish this, connect 6 1TB EBS volumes, at 500 each. Connect these volumes at the OS level.

Amazon EBS capacity scenario: 48TB shared block storage (1 of 3)

- 48TB of shared block storage



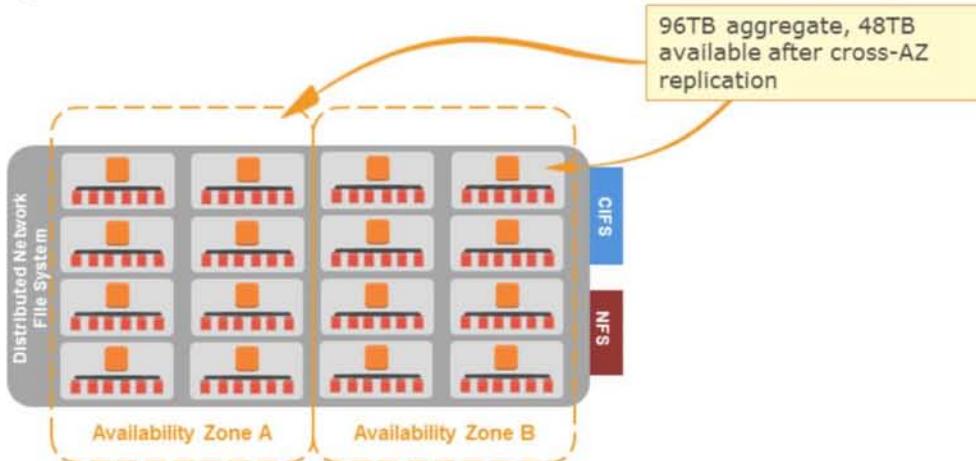
Architecting on AWS – Data Storage Scaling

amazon web services | Training and Certification 17

Notes:

Let's look at another scenario: you need 48TB of shared block storage. To accomplish this, you need 16 EC2 instances, each with 6 1TB EBS volumes, running a distributed file system.

Amazon EBS capacity scenario: 48TB shared block storage (2 of 3)

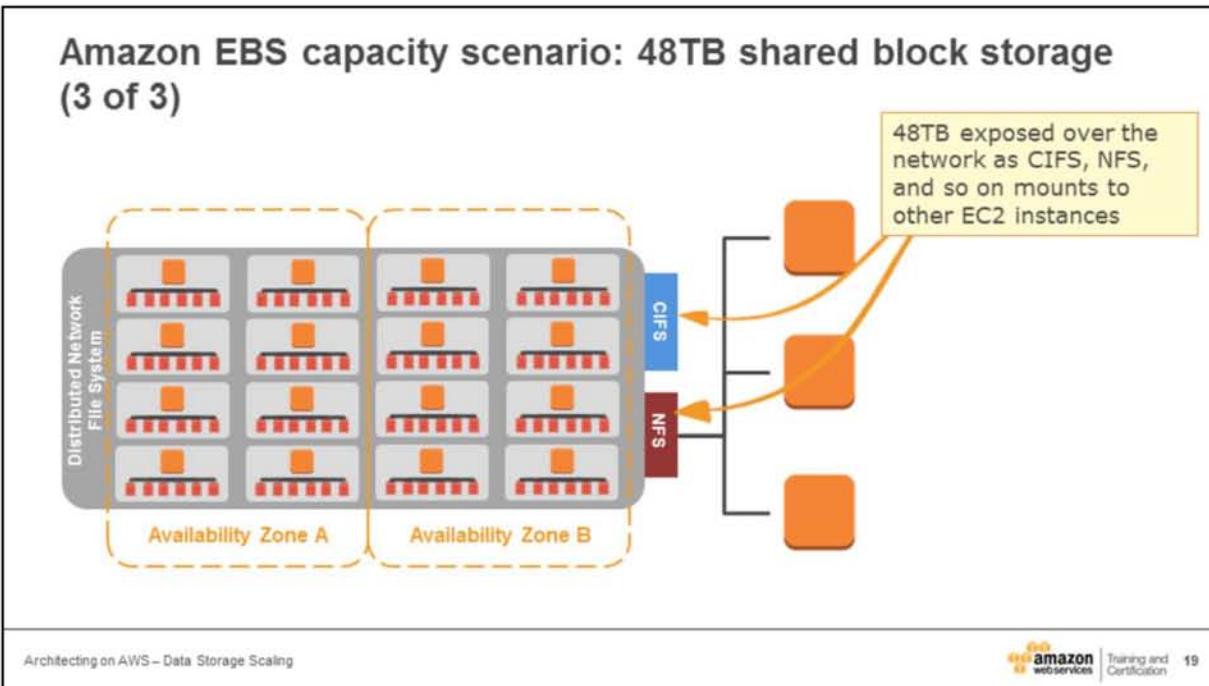


Architecting on AWS – Data Storage Scaling

amazon web services | Training and Certification 18

Notes:

With this configuration, you have 96TB of aggregate storage, with 48TB available after cross-availability zone replication.



Notes:

These 48TB are exposed over the network either as CIFS, NFS, or similar to mount to other EC2 instances.

Amazon EBS Anti-Patterns

- ⚠ Temporary Storage
 - Consider using Amazon EC2 Instance Storage
- ⚠ Very high durability storage
 - Consider using Amazon S3 or snapshots
- ⚠ Storing static web content
 - Consider using Amazon S3
- ⚠ Storing structured data or Key-Value pairs
 - Consider using DynamoDB or Amazon RDS

Notes:

There are a lot of great things you can do with EBS. But a few uses that we don't recommend:

- Temporary storage (ephemeral volumes are better)
- High Durability (S3 or snapshots are the better choice)
- Storing static web content (again, S3 is the better choice)
- Storing structured data (Amazon DynamoDB or Amazon RDS are better solutions)

Amazon EBS Review Questions

- 💡 Describe benefits of using Amazon EBS
- 💡 Differentiate Amazon EBS Standard volumes from Amazon EBS Provisioned IOPS volume
- 💡 Describe Amazon EBS pricing

Notes:

By now you should have a good understanding of the benefits of EBS, how to distinguish Provisioned IOPS from standard volumes, and understand a bit about EBS pricing.

Topics

- ─ Data storage options
- ─ Amazon Elastic Block Storage (EBS)
- ─ Instance storage
 - Benefits of Amazon Instance Store
 - Instance Store Best Practices
- ─ Amazon S3 and Amazon CloudFront

Notes:

Let's move on to instance storage. We will look at the benefits of instance storage, as well as some best practices to consider.

Instance Store (1 of 3)

- No additional charge beyond your Amazon EC2 Instance
- Number and size of volumes varies by Amazon EC2 instance type
 - Larger instances have larger/more volumes
 - hi1.4xlarge = 2 x 1024GB SSD
 - c1.xlarge = 4 x 450GB
 - t1.micro = none
- Not automatically attached; Must request at instance launch
- Volatile
 - No persistence
 - Data is gone when an Amazon EC2 instance stops, fails or is terminated

Notes:

Amazon EC2 instances are divided into different instance types, which determine the size of the instance store available on the instance by default.

The instance type also determines the type of hardware for your instance store volumes.

Example: A high I/O instance (hi1.4xlarge) uses solid state drives (SSD) to deliver very high random I/O performance. This is a good option when you need storage with very low latency, but you don't need it to persist when the instance terminates, or you can take advantage of fault tolerant architectures.

Instance store volumes must be mounted using block device mapping before you can use them. Each instance store volume is pre-formatted with the ext3 file system. You can reformat volumes with the file system of your choice after you launch your instance. A Windows instance uses a built-in tool, EC2Config Service, to reformat the instance store volumes available on an instance with the NTFS file system.

Instance Store (2 of 3)

- 💡 Zero network overhead; local, direct attached resource
 - No network variability
 - Not optimized for random I/O
 - Generally better for sequential I/O
- 💡 Root volume and data volume are lost on physical disk failure, stopping, or terminating of instance
- 💡 Ideal for storing temporary data like buffers, caches, scratch data, and other temporary content, or for data that is replicated across a fleet of instances, such as a load-balanced pool of web servers

Notes:

Instance stores are ideal for storing temporary data, like buffers, caches, and so on. For more robust uses, consider EBS or other storage options.

Instance Store (3 of 3)

💡 High-performance SSD option

- hi1.4xlarge EC2 instance type
- (2) x 1TB SSD local to instance
- ~120,000 random read IOPS (4 KB blocks)
- ~10,000-85,000 random write IOPS (4 KB blocks)

💡 High-storage

- Hs1.8xlarge EC2 instance type
- (24) x 2TB disks local to instance = 48TiB

Notes:

Instance store offers consistent performance for sequential reads and writes.

However, the first write to any location on a standard instance store volume performs more slowly than subsequent writes. (The performance of the solid state drives (SSD) used by high I/O instances is not affected this way.)

If you require high disk performance, we recommend that you initialize your drives by writing once to every drive location before production use.

The data in an instance store persists only during the lifetime of its associated Amazon EC2 instance. Data does persist if an instance reboots. Data is lost under the following circumstances: Failure of an underlying drive, Stopping an Amazon EBS-backed instance, Terminating an instance.

High I/O Instances which is based on solid-state drive (SSD) technology offers a maximum sequential throughput on all AMI types (Linux PV, Linux HVM, and Windows) per second is approximately 2 GB read and 1.1 GB write.

Using Linux paravirtual (PV) AMIs, high I/O instances can deliver more than 120,000 4 KB random read IOPS and between 10,000 and 85,000 4 KB random write IOPS (depending on active logical block addressing span) to applications across two 1 TiB data volumes. For hardware virtual machines (HVM) and Windows AMIs, performance is approximately 90,000 4 KB random read IOPS and between 9,000 and 75,000 4 KB random write IOPS.

They are well suited for NoSQL databases (for example, Cassandra and MongoDB), Clustered databases and Online transaction processing (OLTP) systems.

Instance Store Anti-Patterns

- ⚠ Persistent storage
 - Consider Amazon EBS
- ⚠ Database / Structure Storage
 - Consider Amazon RDS, DynamoDB, and so on
- ⚠ Shareable storage
 - Local instance storage volumes cannot be shared
 - Consider Amazon EBS
- ⚠ Backups
 - Consider Amazon EBS and Amazon EBS Snapshots

Notes:

Instance storage is not a good choice for the following:

- Persistent storage (Amazon EBS is better)
- Database/structure storage (Amazon RDS, or services like DynamoDB, are better)
- Shareable storage (instance storage can't be shared)
- Backups (EBS and EBS Snapshots are better)

Instance Store Review Questions

- 💡 Describe benefits of Amazon Instance Store
- 💡 Describe best practices of using Instance Store

Notes:

At this point, you should have a good understand of Amazon Instance Store, and how to use it effectively.

Topics

- 💡 Data storage options
- 💡 Amazon Elastic Block Storage (EBS)
- 💡 Instance storage
- 💡 Amazon S3 and Amazon CloudFront

Notes:

In this topic, we will discuss the following:

- Amazon S3 and storage classes
- Amazon S3 namespaces
- Amazon S3 Server-side Encryption
- Amazon S3 access controls
- Website hosting and Amazon S3
- Multi-part upload, object versioning, and server access
- Overview of Amazon CloudFront

Amazon Simple Storage Service (S3)



- An object store, not a file system
- Write once, read many (WORM)
- Eventually consistent
- Unlimited storage capacity; pay for what you actually use
- Highly scalable and available data storage
- Objects stored in a region never leaves the region unless explicitly transferred
- Storage for backups
- Provides a REST and a SOAP interface

Notes:

Amazon Simple Storage Service (Amazon S3) is a web service that enables you to store data in the cloud. You can then download the data or use the data with other AWS services such as EC2.

We will discuss Eventual Consistency in the next slide.

Amazon S3 achieves high availability by replicating data across multiple servers within Amazon's data centers. After a "success" is returned, your data is safely stored. It offers 11 9s of durability. S3 offers unlimited storage capacity. Pricing for Amazon S3 is designed so that you don't have to plan for the storage requirements of your application. Amazon S3 charges you only for what you actually use, with no hidden fees and no overage charges.

Amazon S3 achieves high availability by replicating data across multiple servers within Amazon's data centers. After a "success" is returned, your data is safely stored. Objects stored in a Region never leave the Region unless you explicitly transfer them to another Region.

Snapshots of EBS volumes are stored in S3 for durable recovery. You pay 9.5 cents per GB/Month.

Amazon S3 offers APIs in REST and SOAP. Using REST, you use standard HTTP requests to create, fetch, and delete buckets and objects. The REST API uses the standard HTTP headers and status codes, so that standard browsers and toolkits work as expected.

Eventually consistent

■ New Objects

- Synchronously stores your data across multiple facilities before returning SUCCESS
- Read-after-write consistency (Note: except US-STANDARD region)

■ Updates

- Write then read: could report key does not exist
- Write then list: might not include key in list
- Overwrite then read: old data could be returned

■ Deletes

- Delete then read: could still get old data
- Delete then list: deleted key could be included in list

Notes:

New objects:

The US Standard Region provides eventual consistency for all requests. All other regions provide read-after-write consistency for PUTS of new objects and eventual consistency for overwrite PUTS and DELETES.

Amazon S3 achieves high availability by replicating data across multiple servers within Amazon's data centers. After a "success" is returned, your data is safely stored. However, information about the changes might not immediately replicate across Amazon S3 and you might observe the behaviors listed on the slide.

Amazon S3 buckets in the US West (Oregon), US West (Northern California), EU (Ireland), Asia Pacific (Singapore), Asia Pacific (Tokyo), Asia Pacific (Sydney) and South America (Sao Paulo) Regions provide **read-after-write consistency** for PUTS of new objects and **eventual consistency** for overwrite PUTS and DELETES. Amazon S3 buckets in the US Standard Region provide eventual consistency.

Updates:

Updates to a single key are atomic. For example, if you PUT to an existing key, a subsequent read might return the old data or the updated data, but it will never write corrupted or partial data.

Deletes:

Deletes also can take time to become consistent.

Amazon S3 Storage Classes



Standard

- Designed to provide **high durability** and **high availability** of objects over a given year
- Designed to sustain the concurrent loss of data in two facilities
- Objects you want to have high durability e.g. master copy of movie media

Reduced Redundancy Storage (RRS)

- Reduces costs by storing data at lower levels of redundancy than the Standard storage
- Designed to provide high durability and high availability of objects over a given year
- Objects you can afford to lose or can recreate e.g. different encodings of movie media

Notes:

Amazon S3 stores objects according to their storage class, which Amazon S3 assigns to an object when it is written to Amazon S3. The default storage class is STANDARD. You can assign objects a specific storage class (STANDARD or REDUCED_REDUNDANCY) only when writing the objects or when copying objects stored in Amazon S3. RRS is specified per object, at the time the object is written.

RRS enables customers to reduce their costs by storing non-critical, reproducible data at lower levels of redundancy than Amazon S3's standard storage. E.g. distributing or sharing content that is durably stored elsewhere, or for storing thumbnails, transcoded media, or other processed data that can be easily reproduced. The RRS option stores objects on multiple devices across multiple facilities, providing 400 times the durability of a typical disk drive, but does not replicate objects as many times as standard Amazon S3 storage, and thus is even more cost effective. RRS provides 99.99% durability of objects over a given year. This durability level corresponds to an average expected loss of 0.01% of objects annually. You are charged less for using RRS than for standard Amazon S3 storage.

Amazon S3 Storage Classes (continue)



Glacier

- Suitable for archiving data.
- Data retrieval time is 3-5 hour
- Designed for high durability of archives
- Cost effective - Write-once, read-never.
 - \$0.01 per GB for storage
- Pay for accessing data

Notes:

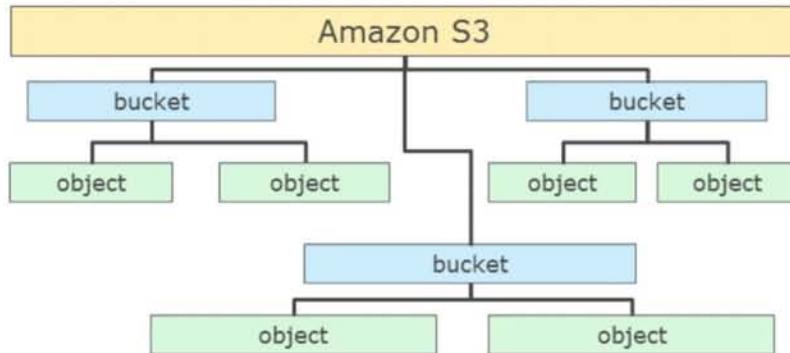
The Glacier storage class is suitable for archiving data, where data access is infrequent and a retrieval time of several hours is acceptable. The Glacier storage class uses the very low-cost Amazon Glacier storage service, but you still manage objects in this storage class through Amazon S3. You cannot associate an object with the Glacier storage class as you upload it. You transition existing Amazon S3 objects to the Glacier storage class by using lifecycle management.

For the most current pricing in your region, please refer to online documentation → <http://aws.amazon.com/glacier/pricing/>

Buckets, objects and keys



- Bucket name = globally unique
- Maximum of 100 buckets with unlimited object capacity



Notes:

S3 content is stored in buckets. Buckets are similar to Internet domain names. Just as Amazon is the only owner of the domain name Amazon.com, only one person or organization can own a bucket within Amazon S3. Once you create a uniquely named bucket in Amazon S3, you can organize and name the objects within the bucket in any way you like and the bucket will remain yours for as long as you like and as long as you have the Amazon S3 account. Each AWS account can own up to 100 buckets at a time. Bucket ownership is not transferable.

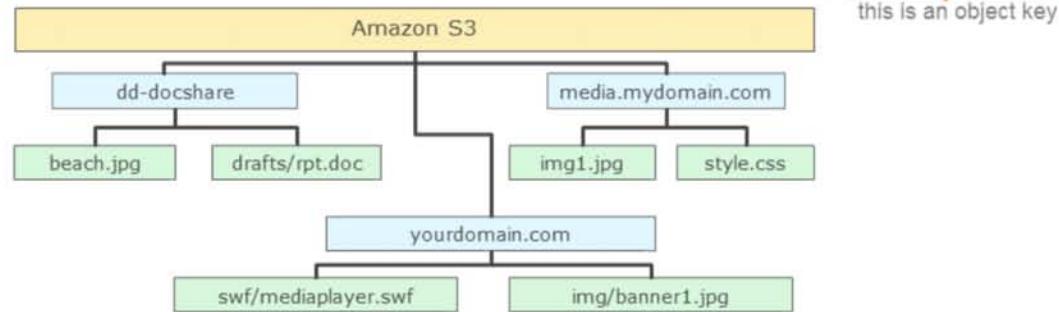
The similarities between buckets and domain names is not a coincidence—there is a direct mapping between Amazon S3 buckets and subdomains of s3.amazonaws.com. Objects stored in Amazon S3 are addressable using the REST API under the domain `bucketname.s3.amazonaws.com`. For example, if the object `homepage.html` is stored in the Amazon S3 bucket `mybucket` its address would be `http://mybucket.s3.amazonaws.com/homepage.html`.

Buckets, objects and keys (continue)



- 💡 Object key = unique within a bucket
- 💡 Bucket name + object name (key) = globally unique
 - Max 1024 bytes UTF-8
 - Including 'path' prefixes

`drafts/rpt.doc`
this is an object key



Notes:

A key is the unique identifier for an object within a bucket. Every object in a bucket has exactly one key. Because the combination of a bucket, key, and version ID uniquely identify each object, Amazon S3 can be thought of as a basic data map between "bucket + key + version" and the object itself. Every object in Amazon S3 can be uniquely addressed through the combination of the web service endpoint, bucket name, key, and optionally, a version. For example, in the URL `http://dd-docshare.s3.amazonaws.com/drafts/rpt.doc`, "dd-docshare" is the name of the bucket and "drafts/rpt.doc" is the key. The rules for bucket names in the US Standard region are similar but less restrictive. Bucket names can be as long as 255 characters. Bucket names can contain any combination of uppercase letters, lowercase letters, numbers, periods (.), dashes (-) and underscores (_). In other regions, at least 3 and no more than 63 characters long. Bucket names can contain any combination of lowercase letters, numbers, periods (.), dashes (-) but must start and end with lowercase letter or number. These naming rules for US Standard region can result in a bucket name that is not DNS-compliant. For example, MyAWSBucket, is a valid bucket name, with uppercase letters in its name. If you try to access this bucket using a virtual hosted-style request, `http://MyAWSBucket.s3.amazonaws.com/yourobject`, the URL resolves to the bucket myawsbucket and not the bucket MyAWSBucket. In response, Amazon S3 will return a bucket not found error. To avoid this problem, we recommend as a best practice that you always use DNS-compliant bucket names regardless of the region in which you create the bucket.

Server Side Encryption (SSE)



- Automatic encryption of data at rest
- Durable - Amazon S3 key storage
- Secure - 3-way simultaneous access
- Self managed - No need to manage a key store
- Strong - AES-256
- Simple - Additional PUT header

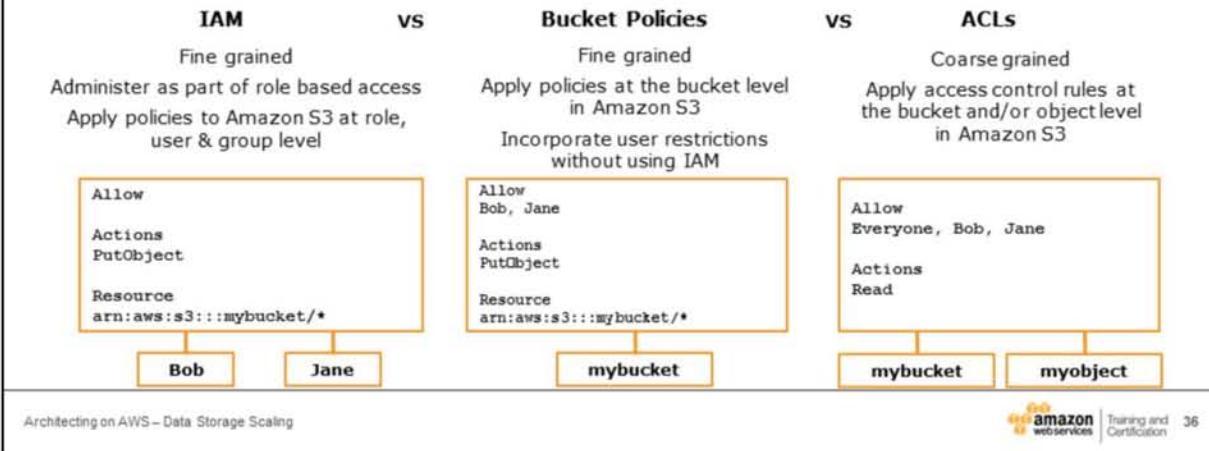
Notes:

Encryption provides added security for your object data stored in your buckets in Amazon S3. You can encrypt data on your client-side and upload the encrypted data to Amazon S3. In this case, you manage encryption process, the encryption keys, and related tools. Optionally, you might want to use the server-side encryption feature in which Amazon S3 encrypts your object data before saving it on disks in its data centers and decrypts it when you download the objects, freeing you from the tasks of managing encryption, encryption keys, and related tools. Amazon S3 Server Side Encryption employs strong multi-factor encryption. Amazon S3 encrypts each object with a unique key. As an additional safeguard, it encrypts the key itself with a master key that it regularly rotates. Amazon S3 Server Side Encryption uses one of the strongest block ciphers available, 256-bit Advanced Encryption Standard (AES-256), to encrypt your data. You can specify data encryption at the object level. When you upload an object, you can explicitly specify in your request if you want Amazon S3 to save your object data encrypted. Server-side encryption is optional. Your bucket might contain both encrypted and unencrypted objects. Amazon S3 supports bucket policy that you can use if you require server-side encryption for all objects that are stored in your bucket. For example, the following bucket policy denies upload object (s3:PutObject) permission to everyone if the request does not include the x-amz-server-side-encryption header requesting server-side encryption. Server-side encryption encrypts only the object data. Any object metadata is not encrypted. The object creation REST APIs provide a request header, x-amz-server-side-encryption that you can use to request server-side encryption.

Access Controls



- 💡 Use Amazon S3 policies, ACLs or IAM to define rules
- 💡 Apply policies to buckets and objects



Notes:

Amazon S3 enables you to manage access to objects and buckets using access control lists (ACLs), bucket policies and IAM policies. You can use them independently or together.

With ACLs, you can only grant other AWS accounts access to your Amazon S3 resources. With IAM policies, you can only grant users within your own AWS account permission to your Amazon S3 resources. With bucket policies, you can do both.

The IAM policy (on the left) allows the Amazon S3 PutObject action for the bucket called mybucket in your AWS account, and it's attached to the users Bob and Jane. The bucket policy (on the right) is attached to mybucket. As with the IAM policy, the bucket policy gives Bob and Jane permission to access PutObject on mybucket. In this example, IAM policy and a bucket policy that are equivalent.

IAM policies lets you manage access to your Amazon S3 resources based on user; whereas bucket policies let you manage access based on the specific resources.

Amazon S3: Sample Bucket Policy

```
{"Statement": [{"Effect": "Allow",  
"Principal": {"AWS": ["4649-6425", "5243-0045"]},  
"Action": "*",  
"Resource": "/mybucket/*",  
"Condition": {"IpAddress": {"AWS:SourceIp": "176.13.0.0/12"}  
}]}]
```

Diagram annotations:

- Accounts to allow**: A yellow box with an orange arrow pointing to the "Principal" field, which contains two AWS account IDs: "4649-6425" and "5243-0045".
- Resource**: A yellow box with an orange arrow pointing to the "Resource" field, which is set to "/mybucket/*".
- Source address to allow**: A yellow box with an orange arrow pointing to the "Condition" field, specifically to the "IpAddress" key and its value "176.13.0.0/12".



Notes:

Let's look at a sample bucket policy.

The policy itself is written in JSON and uses the access policy language. You can use the AWS Policy Generator tool to create a bucket policy for your Amazon S3 bucket. Policy Generator can be accessed from S3 console or using <http://awspolicygen.s3.amazonaws.com/policygen.html>

Policy keys let you restrict access to resources based on information other than just the API action being requested. They let you restrict access based on contextual information about the request, such as the IP address of the requester, the time and date of the request, etc.

AWS provides a set of common keys supported by all AWS products that adopt the access policy language for access control. These keys are:

- `aws:CurrentTime`—Key to use with date conditions to restrict access based on request time. For date/time conditions
- `aws:MultiFactorAuthAge`—Key that provides a numeric value indicating how long ago (in seconds) the temporary credential for Multi-Factor Authentication (MFA) validation, included in the request, was created.
- `aws:SecureTransport`—Boolean representing whether the request was sent

using SSL

- **aws:SourceIp**—The requesters IP address, for use with IP address conditions. If the request comes from an Amazon EC2 instance, we evaluate the instance's public IP address to determine if access is allowed.
- **aws:UserAgent**—Information about the requesters client application, for use with string conditions
- **aws:EpochTime**—Number of seconds since epoch.
- **aws:Referer**—Same as the HTTP *referrer* field.

The key names are case insensitive. For example, `aws:CurrentTime` is equivalent to `AWS:currenttime`.

Website hosting using Amazon S3

Static sites with client-side scripts

Setting default documents

Redirecting requests

Architecting on AWS – Data Storage Scaling

Training and Certification 38

Notes:

You can host a static website on Amazon S3 and provide low latency delivery to your end users with Amazon CloudFront. On a static website, individual web pages include static content. They may also contain client-side scripts. By contrast, a dynamic website relies on server-side processing, including server-side scripts such as PHP, JSP, or ASP.NET. To host a dynamic website or other web applications, you can use Amazon Elastic Compute Cloud (Amazon EC2).

To host your static website, you configure an Amazon S3 bucket for website hosting and then upload your website content to the bucket. The website is then available at the region-specific website endpoint of the bucket: <bucket-name>.s3-website-<AWS-region>.amazonaws.com E.g. <http://examplebucket.s3-website-us-east-1.amazonaws.com/>

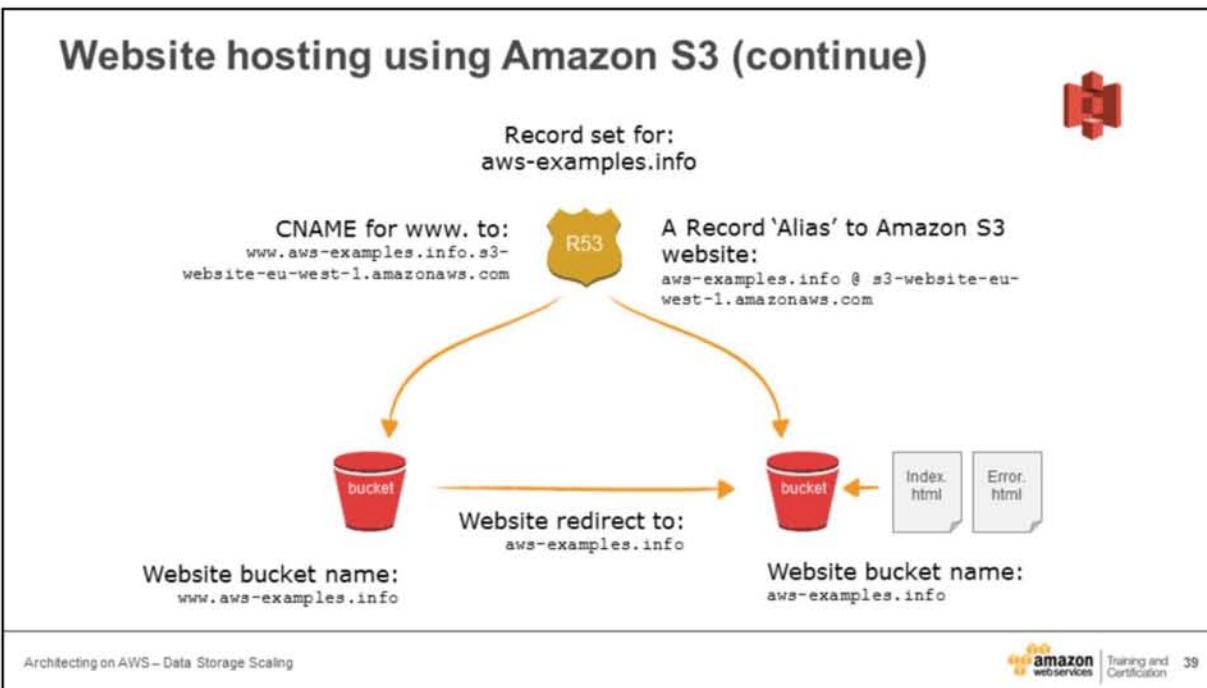
Instead of accessing the website by using an Amazon S3 website endpoint, you can use your own domain, such as example.com to serve your content. Amazon S3, in conjunction with Amazon Route 53, supports hosting a website at the root domain. For example if you have the root domain example.com and you host your website on Amazon S3, your website visitors can access the site from their browser by typing either <http://www.example.com> or <http://example.com>.

Index Documents - Amazon S3 returns this index document when requests are made to the root domain

Error document - If an error occurs, Amazon S3 returns an HTML error document. For 4XX class errors, you can optionally provide your own custom error document.

Redirects all requests - If you root domain is example.com and you want to serve requests for both http://example.com and http://www.example.com, you can create two buckets named example.com and www.example.com, maintain website content in only one bucket, say, example.com, and configure the other bucket to redirect all requests to the example.com bucket.

Advanced conditional redirects - You can conditionally route requests according to specific object key names or prefixes in the request, or according to the response code. For example, suppose that you delete or rename an object in your bucket. You can add a routing rule that redirects the request to another object.



Notes:

You can use the Amazon Route 53 service to create a CNAME record for your domain and the Amazon S3 bucket name or the Amazon CloudFront distribution name. Note that you will also need to configure your Amazon S3 bucket or your Amazon CloudFront distribution respectively with the CNAME entry to completely establish the alias between your domain name and the AWS domain name for your bucket or distribution.

For Amazon S3 buckets configured to host static websites, you also have the option of creating an 'Alias' record that maps to your S3 website bucket. Alias records have two advantages: first, unlike CNAMEs, you can create an Alias record for your zone apex (e.g. example.com, instead of www.example.com), and second, queries to Alias records are free of charge.

IP addresses associated with Amazon S3 website endpoints (e.g. s3.amazonaws.com) can change at any time due to scaling up, scaling down, or software updates. Route 53 responds to each request for an Alias record with one IP address for the bucket.

DNS Considerations

One of the design requirements of Amazon S3 is extremely high availability. One of the ways we meet this requirement is by updating the IP addresses associated with the Amazon S3 endpoint in DNS as needed. These changes are automatically reflected in short-lived clients, but not in some long-lived clients. Long-lived clients will need to take special action to re-resolve the Amazon S3 endpoint periodically to

benefit from these changes.

For Java, Sun's JVM caches DNS lookups forever by default;

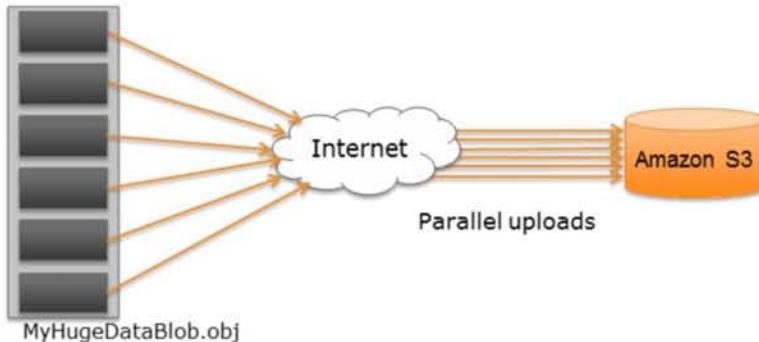
networkaddress.cache.ttl (default: -1) A value of -1 indicates "cache forever". Refer the "InetAddress Caching" section of the InetAddress documentation.

For PHP, the persistent PHP VM that runs in the most popular deployment configurations caches DNS lookups until the VM is restarted. Refer the `getHostByName` PHP docs.

Multipart Upload



- Upload as 100 MB per part
- Multipart Upload supports a maximum of 10,000 parts



Notes:

Depending on the size of the data you are uploading, Amazon S3 offers the following options:

Upload objects in a single operation—With a single PUT operation you can upload objects up to 5 GB in size.

Upload objects in parts—Using the Multipart upload API you can upload large objects, up to 5 TB.

The Multipart Upload API is designed to improve the upload experience for larger objects. You can upload objects in parts. These object parts can be uploaded independently, in any order, and in parallel. You can use a Multipart Upload for objects from 5 MB to 5 TB in size. We encourage Amazon S3 customers to use Multipart Upload for objects greater than 100 MB.

Advantages

Improved throughput—You can upload parts in parallel to improve throughput.

Quick recovery from any network issues - If transmission of any part fails, you can retransmit that part without affecting other parts. After all parts of your object are uploaded, Amazon S3 assembles these parts and creates the object.

Pause and resume object uploads - You can upload object parts over time.

Once you initiate a multipart upload there is no expiry; you must explicitly complete or abort the multipart upload.

Begin an upload before you know the final object size - You can upload an object as you are creating it.

All objects and buckets by default are private. The pre-signed URLs are useful if you want your user/customer to be able upload a specific object to your bucket, but you don't require them to have AWS security credentials or permissions. When you create a pre-signed URL, you must provide your security credentials, specify a bucket name an object key, an HTTP method (PUT of uploading objects) and an expiration date and time. The pre-signed URLs are valid only for the specified duration. You can generate a pre-signed URL programmatically using the AWS SDK for Java or the AWS SDK for .NET. If you are using Visual Studio, you can also use the AWS Explorer to generate a pre-signed object URL without writing any code. Anyone who receives a valid pre-signed URL can then programmatically upload an object. You can use the AWS SDK to upload objects. The SDK provides wrapper libraries for you to upload data easily. However, if your application requires it, you can send REST requests directly. You can send a PUT request to upload data in a single operation.

Amazon S3 Object Versioning

The diagram consists of two curved arrows. The left arrow points from the text "Bucket level Automatically preserves all copies of objects" to the "Versioning" section of the screenshot. The right arrow points from the text "Persistent Even deleted object history is held" to the same section. The screenshot shows the "Versioning" configuration page with the "Enabled" radio button selected. It includes a descriptive text about versioning, a note that once enabled it cannot be disabled, and a "Save" and "Cancel" button at the bottom.

Preserve object histories

Bucket level
Automatically preserves all copies of objects

Persistent
Even deleted object history is held

Versioning

Versioning allows you to preserve, retrieve, and restore every version of every object stored in this bucket. This provides an additional level of protection by providing a means of recovery for accidental overwrites or deletions.

Once enabled, Versioning cannot be disabled.

Enabled Suspended

Save Cancel

Architecting on AWS – Data Storage Scaling

amazon web services Training and Certification 41

Notes:

Versioning enables you to keep multiple versions of an object in one bucket. You must explicitly enable versioning on your bucket. By default versioning is disabled. You might enable versioning to recover from unintended overwrites and deletions or to archive objects so that you can retrieve previous versions of them.

Enabling and suspending versioning is done at the bucket level. When you enable versioning for a bucket, all objects added to it will have a unique version ID. Unique version IDs are randomly generated, Unicode, UTF-8 encoded, URL-ready, opaque strings that are at most 1024 bytes long. Only Amazon S3 generates version IDs. They cannot be edited.

When you DELETE an object, all versions remain in the bucket and Amazon S3 inserts a delete marker. By default, GET requests retrieve the most recently stored version. Performing a simple GET Object request when the latest version is a delete marker returns a 404 Not Found error.

You can, however, GET an older version of an object by specifying its version ID. You can permanently delete an object by specifying the version you want to delete. Only the owner of an Amazon S3 bucket can permanently delete a version.

You can add additional security by configuring a bucket to enable MFA (Multi-Factor Authentication) Delete. When you do, the bucket owner must include two forms of authentication in any request to delete a version or change the versioning state of the bucket.

The screenshot shows the AWS S3 console with the 'Service Access Logging' dialog open for the bucket 'example0422'. The dialog includes fields for 'Enabled' (checked), 'Target Bucket' (set to 'example0422'), and 'Target Prefix' (set to 'logs/'). A callout box highlights the logging configuration area with the text: 'Enable logging on the target bucket and choose location where logs are stored'. At the bottom are 'Save' and 'Cancel' buttons. The top right corner features the AWS logo.

Notes:

An Amazon S3 bucket can be configured to create access log records for the requests made against it. An access log record contains details about the request such as the request type, the resource with which the request worked, and the time and date that the request was processed.

By default, server access logs are not collected for a bucket. Once logging is enabled for a bucket, available log records are aggregated into log files and delivered to you via an Amazon S3 bucket of your choosing on an hourly basis.

Object Lifecycle

Bucket: mybucket-0422

Bucket: mybucket-0422
Region: Oregon
Creation Date: Tue Apr 22 18:05:37 GMT-700 2014
Owner: Me

- + Permissions
- + Static Website Hosting
- + Logging
- + Notifications
- Lifecycle

You can manage the lifecycle of objects by using lifecycle rules. Rules enable you to move objects to Amazon Glacier and remove them after a pre-defined time period. Each rule can share a common prefix.

No rules added... [Add rule](#)

Lifecycle Rule

Create a lifecycle rule to schedule the archival of objects to Glacier and/or permanent removal of objects. Objects transitioned to Glacier will no longer be immediately accessible. Most restores for transitioned objects will take 3 to 5 hours. Learn more.

Enabled:

Name (Optional): Rule Name (auto-generate)

Apply to Entire Bucket:

Prefix:

Time Period Format: Days from the creation date Effective from date

No actions added...

Move to Glacier Expiration

Note: A lifecycle rule is a bulk operation that can potentially affect a large number of objects. See Glacier pricing considerations.

Save Cancel

Save Cancel

Architecting on AWS – Data Storage Scaling

 Training and Certification 43

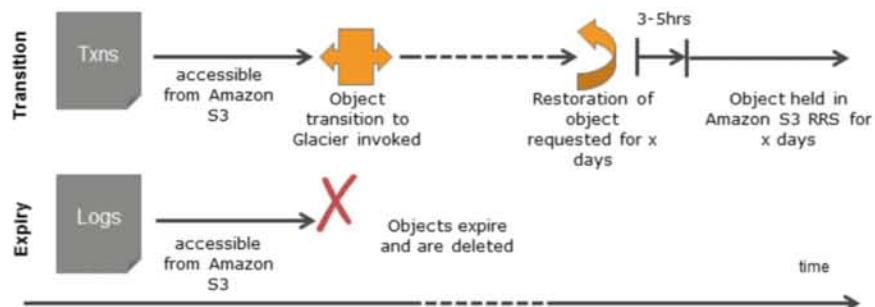
Notes:

You can create and assign lifecycle rules to transition an object to Glacier or expire it. Let discuss more on Object Lifecycle in the next slide.

Object Lifecycle Management



- Automated management of objects
- Object Expiration - Permanently delete objects from Amazon S3
- Object Archival - Move objects to Glacier and out of Amazon S3 storage



Notes:

In-Context: If you are uploading periodic logs to your bucket, your application might need these logs for a week or a month after creation, and after that you might want to delete them.

Object Expiration - When the specified time period is reached in the object's lifetime, Amazon S3 deletes it. Expiration applies to all Amazon S3 objects, including those that are archived in Amazon Glacier.

Object Transition - When a specified date or time period in the object's lifetime is reached, Amazon S3 sets the storage class to Glacier. Digital media archives, financial and healthcare records, raw genomics sequence data, long-term database backups, and data that must be retained for regulatory compliance are some kinds of data that you might upload to Amazon S3 primarily for archival purposes. A lifecycle configuration can contain as many as 1000 rules. If your bucket is version-enabled or versioning is suspended, you cannot add a lifecycle configuration.

Object Lifecycle Management (continue)



Considerations before archiving objects

- Not available in real time
- Transition action is only one-way
- Visible and available only through Amazon S3
- Rule with an empty key prefix

Before You Decide to Expire Objects

- Expiration action deletes objects
- Rule with an empty key prefix

Notes:

Not available in real time - Archived objects are Amazon S3 objects, but before you can access an archived object, you must first restore a temporary copy of it. The restored object copy is available only for the duration you specify in the restore request. After that, Amazon S3 deletes the temporary copy, and the object remains archived in Amazon Glacier. Note that object restoration from an archive can take from three to five hours.

Transition action is only one-way - You cannot use a lifecycle configuration rule to convert a Glacier object to a Standard or Reduced Redundancy Storage (RRS) object. If you want to change the storage class of an already archived object to either Standard or RRS, you must use the restore operation to make a temporary copy first. Then use the copy operation to overwrite the object as the Standard or the RRS object.

Visible and available only through Amazon S3 - Amazon S3 stores the archived objects in Amazon Glacier; however, these are Amazon S3 objects, and you can access them only by using the Amazon S3 console or the API. You cannot access the archived objects through the Amazon Glacier console or the API.

Rule with an empty key prefix - If you specify an empty prefix, the rule applies to all objects in the bucket. So if the rule specifies a transition action with an empty prefix, you are requesting Amazon S3 to archive all the objects in the bucket at a specific period in the objects' lifetime.

Expiration action deletes objects - You might have objects in Amazon S3 or

archived to Amazon Glacier. No matter where these objects are, Amazon S3 will delete them. You will no longer be able to access these objects.

Rule with an empty key prefix - If you specify an empty prefix, the rule applies to all objects in the bucket. So if the rule specifies an expiration action with an empty prefix, you are requesting Amazon S3 to delete all the objects in the bucket at a specific period in the objects' lifetime.

Amazon S3 Review Questions

- 💡 Describe Amazon S3 and storage classes
- 💡 Describe Amazon S3 namespaces
- 💡 Describe Amazon S3 Server Side Encryption
- 💡 Define Amazon S3 access controls
- 💡 Describe website hosting using Amazon S3

Copyright © 2013, 2014 Amazon Web Services, Inc. and its affiliates. All rights reserved.

This work may not be reproduced or redistributed, in whole or in part, without prior written permission from Amazon Web Services, Inc. Commercial copying, lending, or selling is prohibited.

Errors or corrections? Email us at aws-course-feedback@amazon.com.
Other questions? Email us at aws-training-info@amazon.com.

All trademarks are the property of their owners.



Module 9: Overview of Application Services

Topics

- 💡 AWS application services
- 💡 Roles of application services in AWS architecture
 - Amazon Simple Queue Service (SQS)
 - Amazon Simple Notification Service (SNS)
 - Amazon Simple Workflow Service (SWF)
 - Amazon Simple Email Service (SES) – *beta*
 - Amazon CloudSearch - *beta*

Notes:

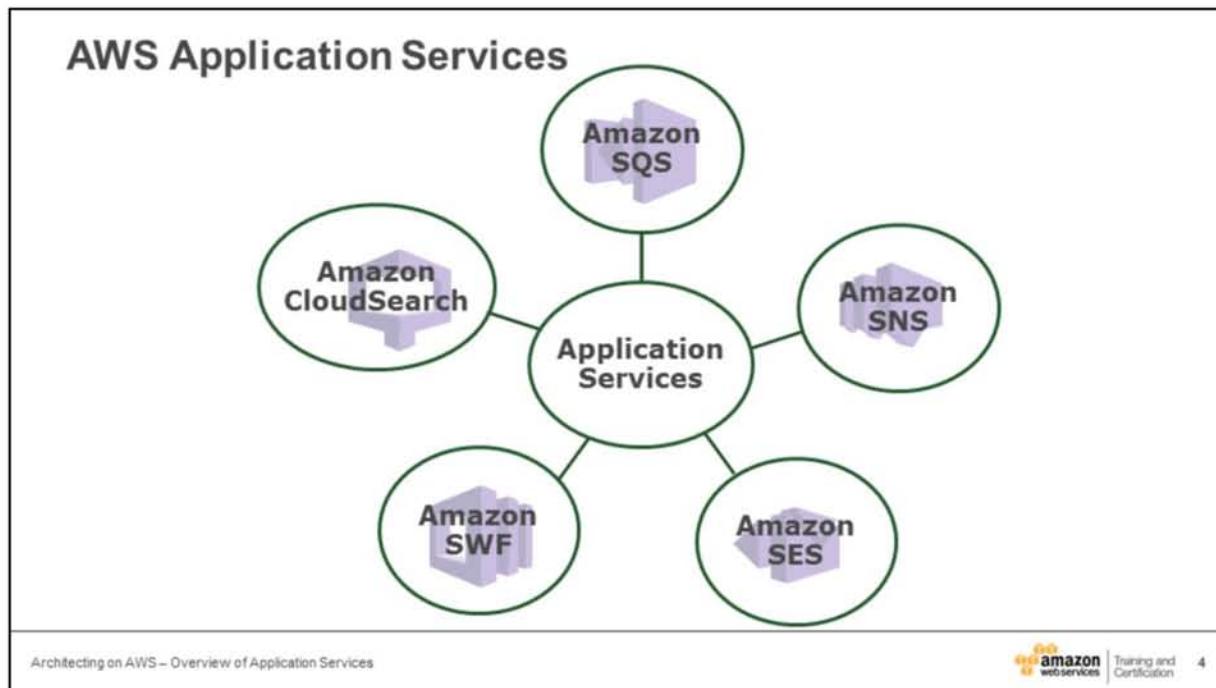
We'll start by going over each service. Then, we'll look at how to use these services when building your applications.

Topics

AWS application services

Roles of application services in AWS architecture

- Amazon Simple Queue Service (SQS)
- Amazon Simple Notification Service (SNS)
- Amazon Simple Workflow Service (SWF)
- Amazon Simple Email Service (SES) – *beta*
- Amazon CloudSearch - *beta*



Notes:

There are five application services you should be aware of.

First is Amazon Simple Queue Service (SQS). Amazon SQS provides a hosted queue for storing messages as they travel between computers, making it easy to build automated workflow between Web services.

Learn more at <http://aws.amazon.com/sqs>.

Next is Amazon Simple Notification Service (SNS). Amazon Simple Notification Service is a web service that makes it easy to set up, operate, and send notifications from the cloud.

Learn more at <http://aws.amazon.com/sns>.

Then, we have Amazon Simple Email Service (SES). Amazon Simple Email Service is a highly scalable and cost-effective bulk and transactional email-sending service for the cloud.

Learn more at <http://aws.amazon.com/ses>.

Amazon Simple Workflow Service (SWF) helps you coordinate the processing steps in your applications and manage distributed execution state.

Learn more at <http://aws.amazon.com/swf>.

Last, we have Amazon CloudSearch. Amazon CloudSearch is a fully-managed

search service in the cloud that allows customers to easily integrate fast and highly scalable search functionality into their applications. Learn more at <http://aws.amazon.com/cloudsearch>.

If you go to <http://aws.amazon.com/products>, you'll see that Amazon Elastic Transcoder is also listed as an application service. But because this service is highly specialized, we're not covering it today. If you think this service is something you might use, check out <http://aws.amazon.com/elastictranscoder>.

Topics

- 💡 AWS application services
- 💡 Roles of application services in AWS architecture
 - Amazon Simple Queue Service (SQS)
 - Amazon Simple Notification Service (SNS)
 - Amazon Simple Workflow Service (SWF)
 - Amazon Simple Email Service (SES) – *beta*
 - Amazon CloudSearch - *beta*

Notes:

Now, you know what application services AWS provides. Let's take a look at how to put them to work.

Amazon Simple Queue Service (SQS)

- 💡 Hosted queue for storing messages as they travel between computers or processes
- 💡 Move data between distributed components of their applications



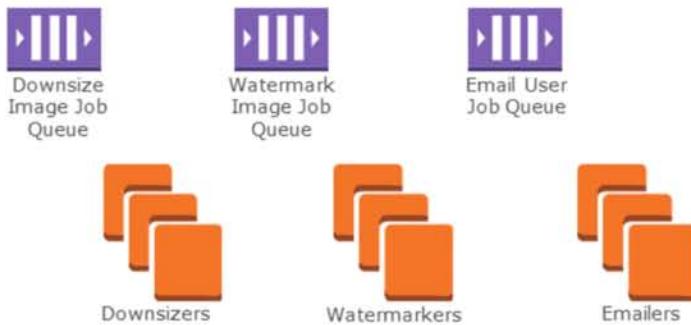
Notes:

As you saw earlier, Amazon Simple Queue Service provides a hosted queue for storing messages as they travel between computers, making it easy to build automated workflow between Web services.

Amazon SQS: Watermarking application example (1 of 4)

User uploads an image

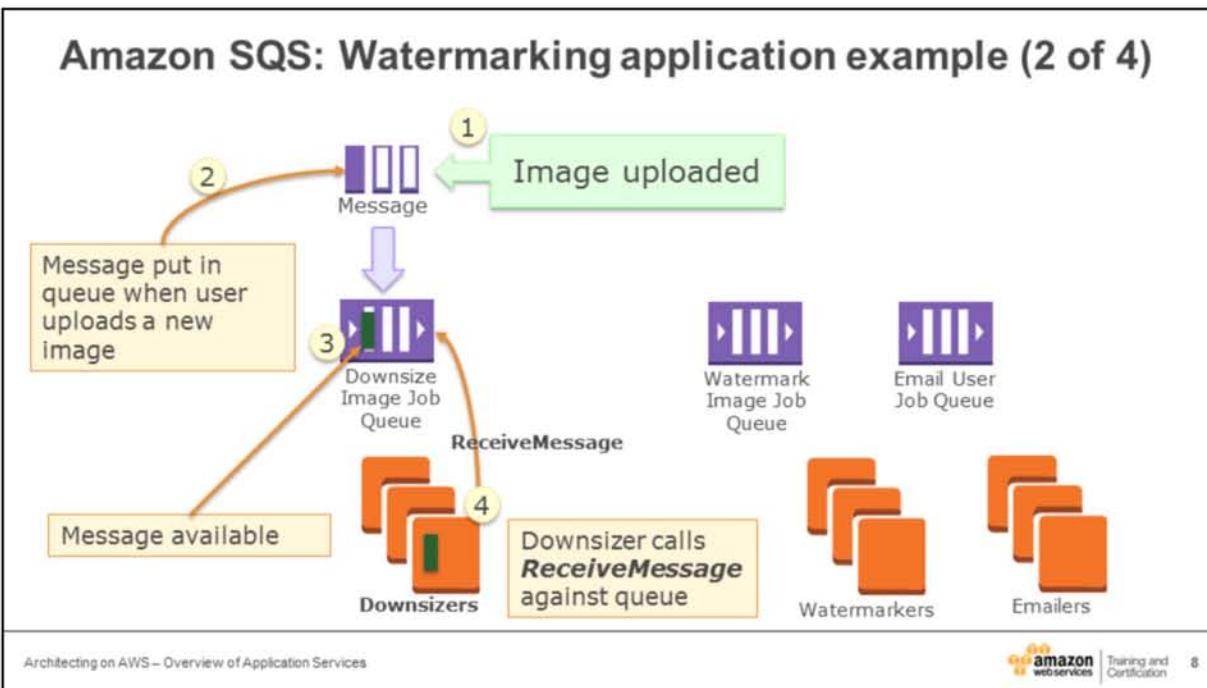
- Image downsized
- Image watermarked
- User e-mailed



Notes:

For this example, consider an app that adds a watermark to an image. This app has three main stages, as shown here.

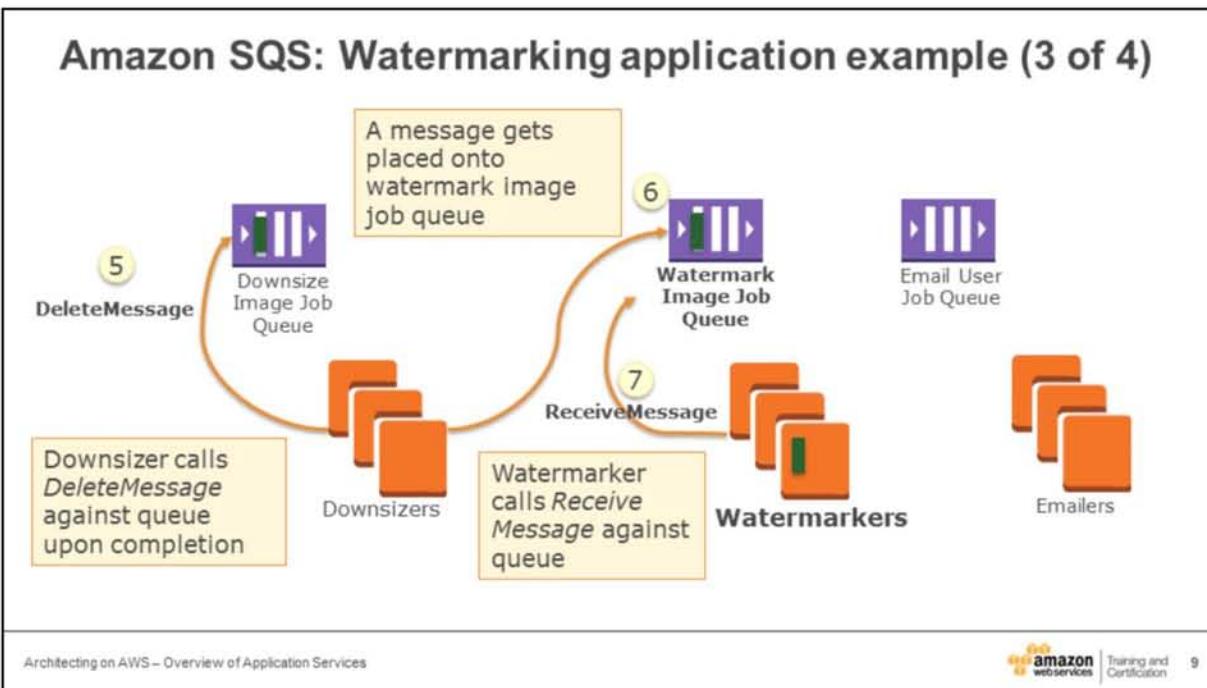
Three groups of EC2 instances handle each of these stages, with SQS queues managing the communication between these groups as an image moves through the process.



Notes:

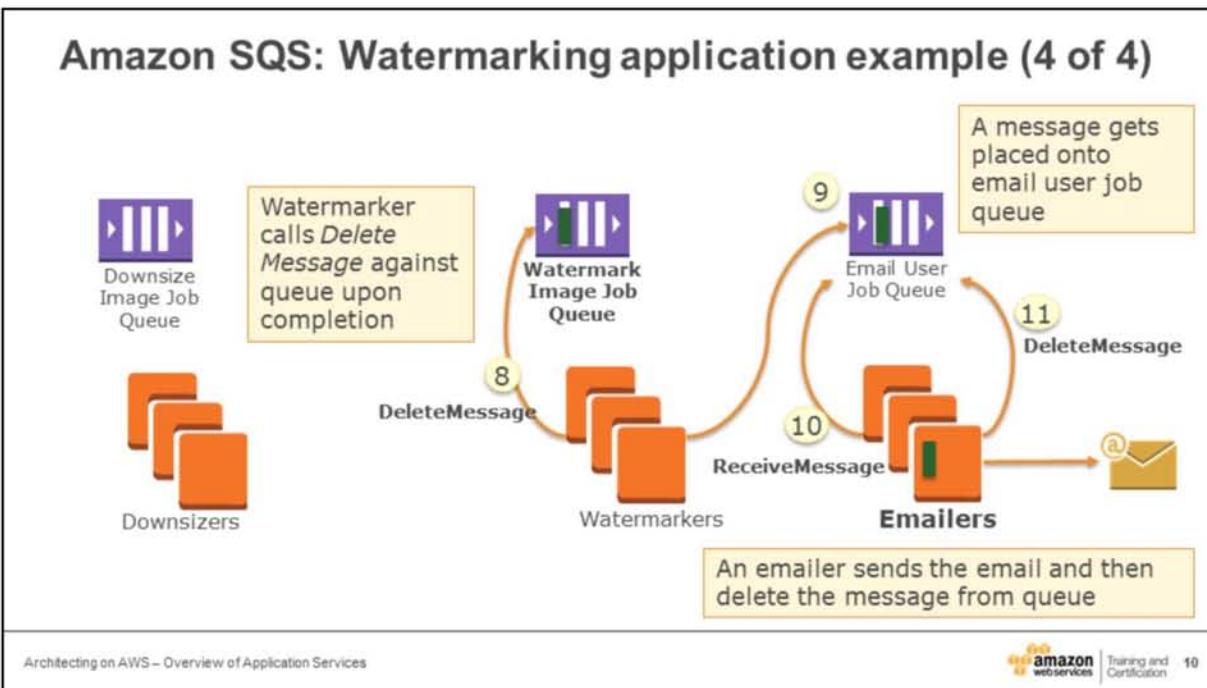
The process starts when the user uploads an image.

1. An image was uploaded
2. A message gets placed onto the queue
3. This information is entered into the “downsize image job queue”
4. The EC2 instances that downsize images call `ReceiveMessages` to get messages from the queue



Notes:

1. An image was uploaded
2. A message gets placed onto the queue
3. This information is entered into the “downsize image job queue”
4. The EC2 instances that downsize images call ReceiveMessages to get messages from the queue
5. When the EC2 instances downsize the image, they call DeleteMessage to remove the message from the queue.
6. Next, the EC2 instances puts a message into the next queue—in this case, a watermark image queue.
7. Just as before, an EC2 instance in the Watermark Image group gets the message by calling ReceiveMessages.



Notes:

1. An image was uploaded
2. A message gets placed onto the queue
3. This information is entered into the “downsize image job queue”
4. The EC2 instances that downsize images call `ReceiveMessages` to get messages from the queue
5. When the EC2 instances downsize the image, they call `DeleteMessage` to remove the message from the queue.
6. Next, the EC2 instances puts a message into the next queue—in this case, a watermark image queue.
7. Just as before, an EC2 instance in the Watermark Image group gets the message by calling `ReceiveMessages`.
8. Deletes the message when finished.
9. Watermarker instance puts the message into the email queue.
10. An emailer receives the message, sends the mail
11. Finally, emailer deletes the message.

Topics

- 💡 AWS application services
- 💡 Roles of application services in AWS architecture
 - Amazon Simple Queue Service (SQS)
 - Amazon Simple Notification Service (SNS)
 - Amazon Simple Workflow Service (SWF)
 - Amazon Simple Email Service (SES) – *beta*
 - Amazon CloudSearch - *beta*

Notes:

Amazon Simple Notification Service (SNS)

- 💡 Set up, operate, and send notifications
- 💡 Publish messages from an application and immediately deliver them to subscribers or other applications
- 💡 Messages published to topic
- 💡 Topic subscribers receive message
- 💡 Subscriber types:
 - Email (plain or JSON)
 - HTTP/HTTPS
 - SMS
 - SQS
 - Mobile Push Messaging

Notes:

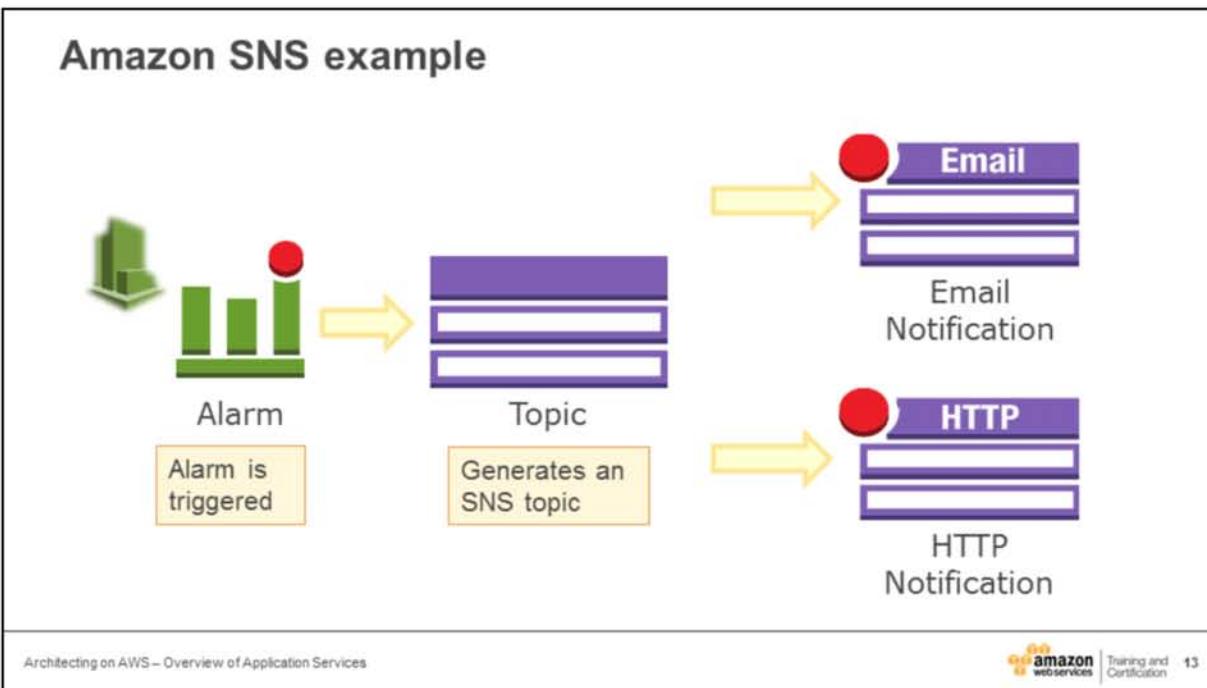
Amazon Simple Notification Service is a web service that makes it easy to set up, operate, and send notifications from the cloud.

Mobile Push Messaging:

Amazon SNS lets you push messages to mobile devices or distributed services, via API or an easy-to-use management console. You can seamlessly scale from a handful of messages per day to millions of messages or higher.

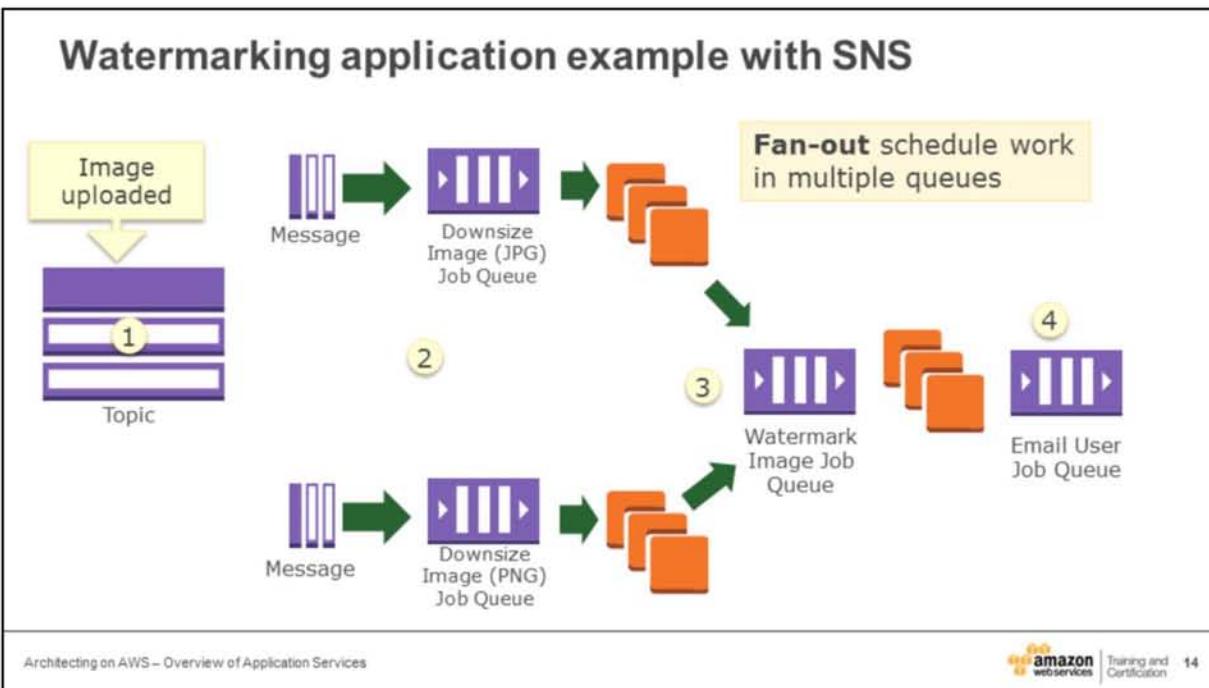
With SNS you can publish a message once, and deliver it one or more times. So you can choose to direct unique messages to individual Apple, Google or Amazon devices, or broadcast deliveries to many mobile devices with a single publish request.

SNS allows you to group multiple recipients using topics. A topic is an “access point” for allowing recipients to dynamically subscribe for identical copies of the same notification. One topic can support deliveries to multiple endpoint types -- for example, you can group together iOS, Android and SMS recipients. When you publish once to a topic, SNS delivers appropriately formatted copies of your message to each subscriber.



Notes:

Let's say you have a CloudWatch alarm set, and it goes off which generates an SNS topic. Amazon SNS can then send notifications to multiple sources at once—in this case, we send an email notification and an HTTP notification.



Notes:

Let's look at our watermark application again. We have a new requirement: we want to downsize an image both in a JPG and PNG file format. Instead of doing this one step at a time, we'll use Amazon SNS.

1. Image uploaded which creates a topic
2. SNS sends a message to two queues: one that downsizes the image into a JPG format, and one that downsizes it into a PNG format
3. Each downsizer group completes its work, the images are sent to the Watermark Image queue, just as before.
4. The images then proceed through the app until the final result is emailed to the user.

Topics

- 💡 AWS application services
- 💡 Roles of application services in AWS architecture
 - Amazon Simple Queue Service (SQS)
 - Amazon Simple Notification Service (SNS)
 - Amazon Simple Workflow Service (SWF)
 - Amazon Simple Email Service (SES) – *beta*
 - Amazon CloudSearch - *beta*

Amazon Simple Workflow Service (SWF)

- Manage workflows, including state, decisions, executions, tasks and logging
- Coordinate processing steps across distributed systems
- Ensure tasks are executed reliably, in order, and without duplication
- HTTP API: executed from code written in any language
- Use from any app that supports HTTP, from EC2 to mobile device in a warehouse

Notes:

Amazon SWF helps you coordinate the processing steps in your applications and manage distributed execution state.

Amazon SWF: Decider and Activity Worker

-Decider

- Decision logic (i.e., flow control)
- Decoupled from application logic
- Makes decision based on events assigned by SWF

-Activity Worker

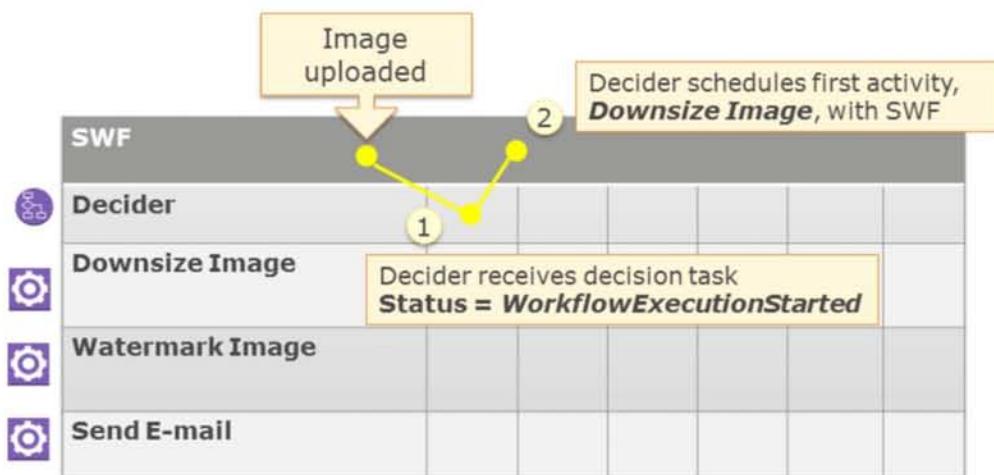
- Does work assigned by SWF
- Work scheduled by Decider
- Run anywhere

Notes:

Amazon SWF is powerful, but requires more setup work than Amazon SQS and Amazon SNS. With Amazon SWF, you must write a decider and an activity worker.

The decider controls your decision logic. Your activity worker does the actual work.

Watermarking application example with SWF (1 of 4)



Architecting on AWS – Overview of Application Services

 Amazon
Training and
Certification 18

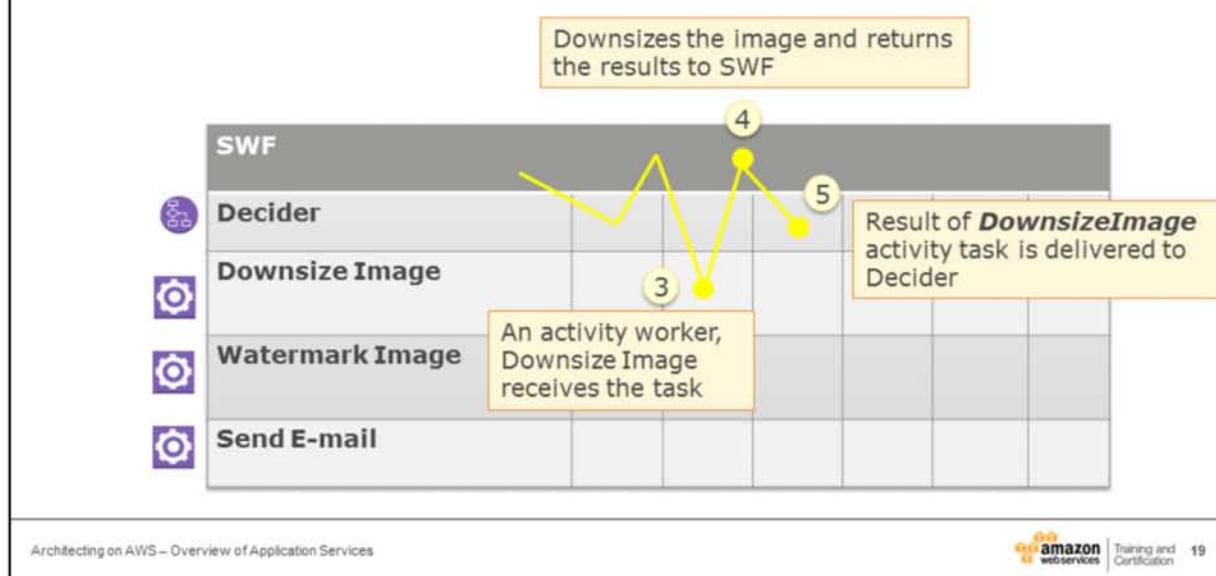
Note:

This time, we have we'll use Amazon SWF to handle moving the image through the different stages of the app.

One decider and three activity tasks: downsize image, watermark image and send e-mail.

1. Decider receives decision task: **Status = WorkflowExecutionStarted**
2. Decider schedules first activity, **Downsize Image**, with SWF

Watermarking application example with SWF (2 of 4)

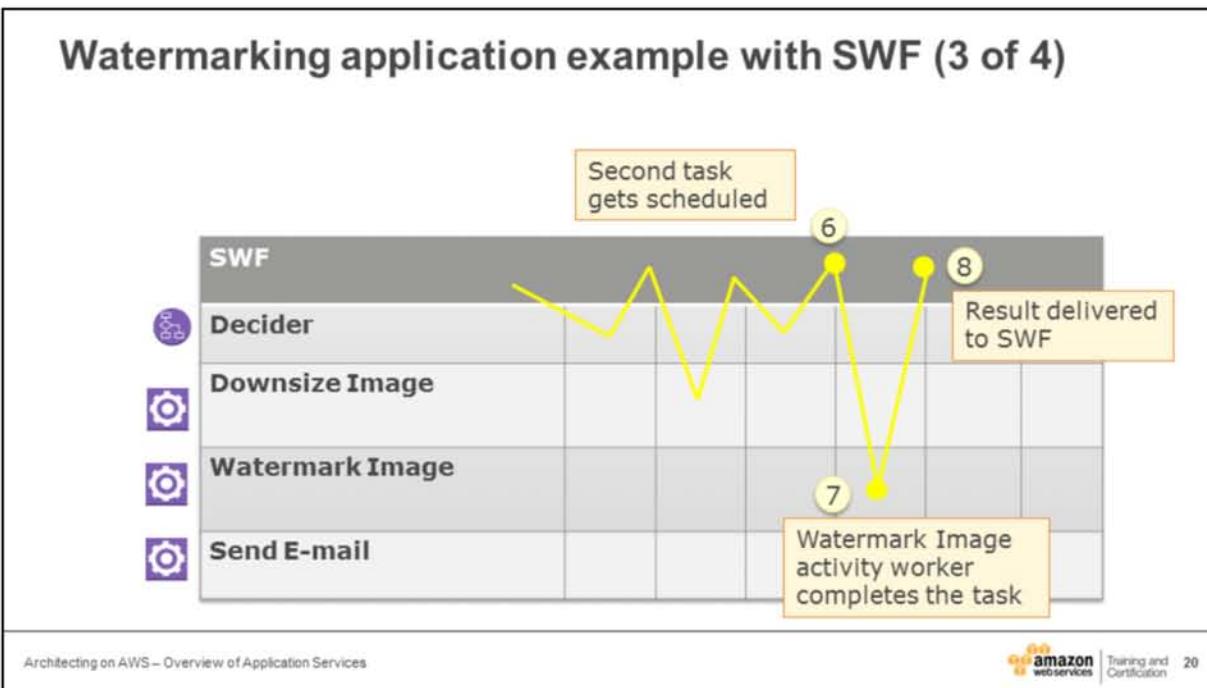


Note:

This time, we have we'll use Amazon SWF to handle moving the image through the different stages of the app.

One decider and three activity tasks: downsize image, watermark image and send e-mail.

1. Decider receives decision task: Status = *WorkflowExecutionStarted*
2. Decider schedules first activity, *Downsize Image*, with SWF
3. **An activity worker receives the task**
4. **Downsizes the image, and returns the results to SWF**
5. **Result of *DownsizeImage* activity task is delivered to Decider**



Note:

This time, we have we'll use Amazon SWF to handle moving the image through the different stages of the app.

One decider and three activity tasks: downsize image, watermark image and send e-mail.

1. Decider receives decision task: **Status = WorkflowExecutionStarted**
2. Decider schedules first activity, **Downsize Image**, with SWF
3. An activity worker receives the task
4. Downsizes the image, and returns the results to SWF
5. Result of **DownsizeImage** activity task is delivered to Decider
6. **Decider schedules second task**
7. **Image watermarked by Activity Worker**
8. **Result delivered to SWF**

Watermarking application example with SWF (4 of 4)



Architecting on AWS – Overview of Application Services

amazon
web services | Training and Certification 21

Notes:

This time, we have we'll use Amazon SWF to handle moving the image through the different stages of the app.

One decider and three activity tasks: downsize image, watermark image and send e-mail.

1. Decider receives decision task: Status = *WorkflowExecutionStarted*
2. Decider schedules first activity, *Downsize Image*, with SWF
3. An activity worker receives the task
4. Downsizes the image, and returns the results to SWF
5. Result of *DownsizeImage* activity task is delivered to Decider
6. Decider schedules second task
7. Image watermarked by Activity Worker
8. Result delivered to SWF
9. Repeat the process until the workflow completes

Topics

📦 AWS application services

📦 Roles of application services in AWS architecture

- Amazon Simple Queue Service (SQS)
- Amazon Simple Notification Service (SNS)
- Amazon Simple Workflow Service (SWF)
- Amazon Simple Email Service (SES) – *beta*
- Amazon CloudSearch - *beta*

Notes:

Amazon Simple Email Service (SES)



- Bulk and transactional email-sending service
- Eliminates the hassle of email server management, network configuration, and meeting rigorous Internet Service Provider (ISP) standards
- Provides a built-in feedback loop, which includes notifications of bounce backs, failed and successful delivery attempts, and spam complaints

Notes:

As of April, 2014, Amazon Simple Email Service is still in *beta*. Please follow the update at AWS web site for any up-coming news → <http://aws.amazon.com/new/>

Amazon Simple Email Service is a highly scalable and cost-effective bulk and transactional email-sending service for the cloud.

Topics

📦 AWS application services

📦 Roles of application services in AWS architecture

- Amazon Simple Queue Service (SQS)
- Amazon Simple Notification Service (SNS)
- Amazon Simple Workflow Service (SWF)
- Amazon Simple Email Service (SES) – *beta*
- Amazon CloudSearch - *beta*

Amazon CloudSearch



- Fully-managed search service
- Integrate fast and highly scalable search functionality into applications
- Scales automatically: with increases in searchable data or as query rate changes
- You can assign weights to selected fields for customized relevance ranking
- AWS manages hardware provisioning, data partitioning, and software patches
- Supports 33 languages and popular search features such as highlighting, autocomplete, and geospatial search

Notes:

As of May, 2014, Amazon CloudSearch is still in *beta*. Please follow the update at AWS web site for any up-coming news → <http://aws.amazon.com/new/>

Amazon CloudSearch is a fully-managed search service in the cloud that allows customers to easily integrate fast and highly scalable search functionality into their applications.

CloudSearch supports two types of text fields, text and literal. Text fields are processed according to the language configured for the field to determine individual words that can serve as matches for queries. Literal fields are not processed and must match exactly, including case. CloudSearch also supports four numeric types: int, double, date, and latlon. Int fields hold 64-bit, signed integer values. Double fields hold double-width floating point values. Date fields hold dates specified in UTC (Coordinated Universal Time) according to IETF RFC3339: yyyy-mm-ddT00:00:00Z. Latlon fields contain a location stored as a latitude and longitude value pair.

You can assign weights to selected fields so you can boost the relevance _score of documents with matches in key fields such as a title field, and minimize the impact of matches in less important fields. By default all fields have a weight of 1. To learn more, refer to online document → <http://docs.aws.amazon.com/cloudsearch/latest/developerguide/weighting.html>

fields.html

Q: How does my search domain scale to meet my application needs?

Search domains scale in two dimensions: data and traffic. As your data volume grows, you need more (or larger) Search instances to contain your indexed data, and your index is partitioned among the search instances. As your request volume or request complexity increases, each Search Partition must be replicated to provide additional CPU for that Search Partition. For example, if your data requires three search partitions, you will have 3 search instances in your search domain. As your traffic increases beyond the capacity of a single search instance, each partition is replicated to provide additional CPU capacity, adding an additional three search instances to your search domain. Further increases in traffic will result in additional replicas, to a maximum of 5, for each search partition.

Module review

- 💡 List the five main AWS application services
- 💡 What role does SQS play in designing loosely-coupled systems?
- 💡 What subscriber types does SNS support?
- 💡 What are the two components of SWF?

Notes:

Five services:

SQS, SNS, SES, SWF, CloudSearch

Role:

Messaging queue between two systems (Web server and app server, etc.)

SNS formats:

Email, HTTP/HTTPS, SMS, SQS, Mobile Push Messaging

Two components of SWF:

Decider, Activity Worker

Copyright © 2013, 2014 Amazon Web Services, Inc. and its affiliates. All rights reserved.

This work may not be reproduced or redistributed, in whole or in part, without prior written permission from Amazon Web Services, Inc. Commercial copying, lending, or selling is prohibited.

Errors or corrections? Email us at aws-course-feedback@amazon.com.
Other questions? Email us at aws-training-info@amazon.com.

All trademarks are the property of their owners.



Module 10: Designing for Cost

Topics

- 💡 Cost model
- 💡 Services and feature costs
- 💡 Billing options
- 💡 Best practices

[Course Title - Module Title]



Notes:

During this module, we'll discuss:

- The AWS cost model
- Services and feature costs
- Billing options
- Best practices to help create cost-effective systems

Topics

- 💡 Cost model
- 💡 Services and feature costs
- 💡 Billing options
- 💡 Best practices

[Course Title - Module Title]



Notes:

Let's first talk about the AWS cost model.

Cost Model

- 💡 Amazon wants customers to pay for exactly what they use
- 💡 Do not pay for unutilized feature or services
- 💡 This model translates into a very granular cost structure
- 💡 Every Application has different component bounding (CPU, Memory, Disk I/O), pay for what you use
- 💡 Customers have control of how they utilize our products and service, which leads to control over cost expenditures

Notes:

AWS works hard to keep costs down. More importantly, AWS provides you with the ability to control your costs at a granular level. Keep this in mind as you consider which AWS services you want to use.

Topics

- 💡 Cost model
- 💡 Services and feature costs
- 💡 Billing options
- 💡 Best practices

[Course Title - Module Title]



Notes:

Let's look at the costs of AWS services and features.

ELB, EIP, and CloudWatch Costs

EIP

- Free when associated with an EC2 instance
- \$0.005 per hour unassociated
- \$0.10 per 100 remaps

CloudWatch

- Detailed monitoring \$3.50 per instance per month
- Custom metrics \$0.50 per metric per month

ELB

- \$0.025 per ELB-hour
- \$0.008 per GB of bandwidth

Note: Pricing defers from region to region and subject to change. See our web site for the most up-to-date pricing information

Note:

This pricing information on this slide is based on US East (N. Virginia) region as of April 2014. For the most current pricing in your region, please refer to our online documentation: <http://aws.amazon.com/ec2/pricing/>

These are the costs of Elastic Load Balancing, Elastic IP addresses, and CloudWatch. With Elastic IP addresses, remember that you don't pay when it's used, you pay when it is NOT used.

EBS Service and Feature Costs

Standard EBS Volume

- \$0.05 per GB/month
- \$0.05 per million I/O requests

Provisioned IOPS EBS Volumes

- \$0.125 per GB-month of provisioned storage
- \$0.10 per Provisioned IOPS-month

EBS Snapshots to Amazon S3

- \$0.095 per GB-month of data stored

Note:

This pricing information on this slide is based on US East (N. Virginia) region as of April 2014. For the most current pricing in your region, please refer to our online documentation: <http://aws.amazon.com/ebs/pricing/>

Provisioned IOPS Volumes

Volume storage for Provisioned IOPS volumes is charged by the amount you provision in GB per month, until you release the storage. With Provisioned IOPS volumes, you are also charged by the amount you provision in IOPS (input/output operations per second) multiplied by the percentage of days you provision for the month.

For example, if you provision a volume with 1000 IOPS, and keep this volume for 15 days in a 30 day month, then in a Region that charges \$0.10 per provisioned IOPS-month, you would be charged \$50 for the IOPS that you provision:

$$(\$0.10 \text{ per provisioned IOPS-month}) * (1000 \text{ IOPS provisioned}) * (15 \text{ days}) / 30$$

You will be charged for the IOPS provisioned on a Provisioned IOPS volume even when the volume is detached from an instance.

S3, S3 RRS, Glacier Costs

- Pay for capacity used. For the first 1TB per month:
 - S3 Standard Storage - \$0.03/GB
 - S3 Reduced Redundancy Storage - \$0.024/GB
 - Glacier - \$0.01/GB
- S3 PUT, COPY, POST, LIST - \$0.005 per 1,000 requests
- Glacier Archive/Restore - \$0.05 per 1,000 requests
- DELETE - Free
- GET and all other requests - \$0.004 per 10,000 requests
- Data transfer OUT from S3 to Internet - \$0.12 per GB up to 10TB per month

Note:

This pricing information on this slide is based on US East (N. Virginia) region as of April 2014. For the most current pricing in your region, please refer to our online documentation: <http://aws.amazon.com/s3/pricing/>

RDS Service and Feature Costs

- Multiple instance types to choose from
- Provisioned IOPS (up to 30,000 per DB) optional
- Data Transfer Out of a Region - \$0.02/GB
- Reserved billing model available

Note:

This pricing information on this slide is based on US East (N. Virginia) region as of April 2014. For the most current pricing in your region, please refer to our online documentation: <http://aws.amazon.com/rds/pricing/>

RDS has a few ways to configure costs, including the ability to choose multiple instance types, and the option to include Provisioned IOPS. There's also a reserved billing option.

DynamoDB Service Costs

- Provision IOPS capacity
 - \$0.0065 per hour for every 10 units of write capacity
 - \$0.0065 per hour for every 50 units of read capacity
- Indexed data storage
 - First 100MB stored per month is free
 - \$0.25 per GB-month thereafter
- Data transfer OUT - \$0.12/GB up to 10TB/month
- Reserved billing model available

Note:

This pricing information on this slide is based on US East (N. Virginia) region as of April 2014. For the most current pricing in your region, please refer to our online documentation: <http://aws.amazon.com/dynamodb/pricing/>

DynamoDB offers a similar pricing structure as RDS.

R53 and CloudFront Service Costs

Route 53

- \$0.50 per hosted zone/month for the first 25 hosted zones
- \$0.50 per million queries for the first 1 billion queries/month
- \$0.75 per million for latency based queries for the first 1 billion queries/month
- \$0.50 per health check per month inside AWS, \$0.75 outside (S3 Endpoints are free) – Basic Health Check

CloudFront

- Bandwidth out \$0.12 per GB (US and Europe) for the first 10TB/month
- \$0.0075 per 10,000 requests (US) for all HTTP methods
- Pricing based on edge location. For details see:
<http://aws.amazon.com/cloudfront/pricing/>

Note:

This pricing information on this slide is based on US East (N. Virginia) region as of April 2014. For the most current pricing in your region, please refer to our online documentation: <http://aws.amazon.com/route53/pricing/>

SQS, SNS, SES Service and Feature Costs

■ SQS

- First 1 million SQS requests/month are free. \$0.50 per 1 million SQS requests/month thereafter

■ SNS

- \$0.06 per million HTTP/s notifications
- \$2.00 per 100,000 Emails notifications
- \$0.75 per 100 SMS notifications
- No charge for SQS notifications

■ SES

- \$0.10 per 1000 Emails out
- Data transfer inside a region is free
- \$0.12 per GB of attachments sent

Note:

This pricing information on this slide is based on US East (N. Virginia) region as of April 2014. For the most current pricing in your region, please refer to our online documentation:

- <http://aws.amazon.com/sqs/pricing/>
- <http://aws.amazon.com/sns/pricing/>
- <http://aws.amazon.com/ses/pricing/>

Topics

- 💡 Cost model
- 💡 Services and feature costs
- 💡 Billing options
- 💡 Best practices

[Course Title - Module Title]



Notes:

Free Services and Features

- ─ Free Tier Utilization
- ─ VPC
- ─ Auto Scaling
- ─ Cloud Watch standard metrics
- ─ CloudFormation
- ─ IAM
- ─ OpsWorks
- ─ Elastic Beanstalk

Notes:

First off, there are several services that you can access for free. For information on the AWS free usage tier, see: aws.amazon.com/free

EC2 Billing Options

- Prices vary by instance type - optimized to fit different use cases
- On Demand prices should be considered retail rate
- Reserved Instance billing model available
- Unique Spot Market available

Notes:

As part of AWS's free usage tier, new AWS customers can get started with Amazon EC2 for free. Upon sign-up, new AWS customers receive the following EC2 services each month for one year:

- 750 hours of EC2 running Linux/Unix Micro instance usage
- 750 hours of EC2 running Microsoft Windows Server Micro instance usage
- 750 hours of Elastic Load Balancing plus 15 GB data processing
- 30 GB of Amazon EBS Standard volume storage plus 2 million I/Os and 1 GB snapshot storage
- 15 GB of bandwidth out aggregated across all AWS services
- 1 GB of Regional Data Transfer

On-Demand Instances let you pay for compute capacity by the hour with no long-term commitments. This frees you from the costs and complexities of planning, purchasing, and maintaining hardware and transforms what are commonly large fixed costs into much smaller variable costs.

Your choice of Amazon EC2 instances matters

- 💡 A larger compute instance will sometimes save you not only time but money too. Paying more per hour for a shorter amount of time can be less expensive
- 💡 Instances come in multiple sizes, allowing you to optimally scale resources to the requirements of your workload. As you choose an instance type, consider the following:
 - Core count
 - Memory size
 - Storage size & type
 - Network performance

Notes:

Your choice of Amazon EC2 instances matters. Matching it to your workload can save time and money. A lowest-price per hour instance is not necessarily a money saver – a larger compute instance will sometimes save you both money and time.

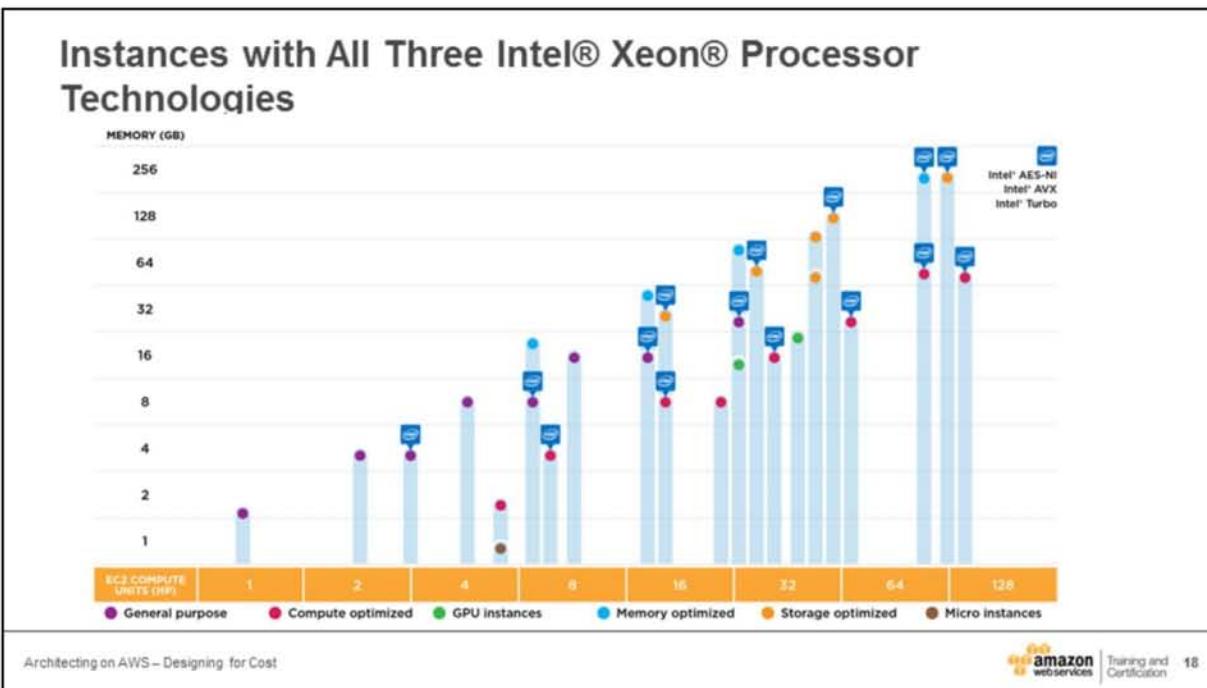
Additional Features that impact your workload

- Intel AES-NI1 – Intel processors that support these new encryption instructions allow you to enable encryption for enhanced data security without paying a performance penalty
- Intel AVX – Get dramatically better performance for highly parallel HPC workloads such as life science engineering, data mining, financial analysis, or other technical computing applications. AVX also enhances image, video, and audio processing.
- Intel Turbo Boost Technology2 – Get a turbo boost of compute speed, accelerating performance for peak loads. This Instance is appropriate for traditional non-parallel workloads.

Note:

1. Intel AES-New Instructions (Intel AES-NI) requires a system with an AES-NI enabled processor, as well as non-Intel software to execute the instructions in the correct sequence. AES-NI is available on select Intel processors. For more information, see <http://software.intel.com/en-us/articles/intel-advanced-encryption-standard-instructions-aes-ni/>.
2. Intel Turbo Boost Technology requires a system with Intel Turbo Boost Technology capability. Performance varies depending on hardware, software, and system configuration. For more information, see <http://www.intel.com/content/www/us/en/architecture-and-technology/turbo-boost/turbo-boost-technology.html>

Simply put, if you can get your workload done faster you can save money. Sometimes paying more per an hour but running an Instance for a shorter amount of time could save money.



Note:

There are a variety of considerations when choosing an Instance type. The resources provided by the Instance such as core count, memory size, and storage size. Infrastructure strength such as storage type & speed, EBS optimization, and network performance should also be considered.

Something to consider is the capabilities directly proved by the Intel processors powering the instance such as:

AES-NI – Intel processors that support these new encryption instructions allow you to turn on encryption for enhance data security without paying a performance penalty.

Intel® AVX – get dramatically better performance for highly parallel HPC workloads such as life science engineering, data mining, financial analysis, or other technical computing applications. AVX also enhances image video, or audio processing.

Intel® Turbo Boost Technology – get a turbo burst of computing accelerating performance for peak loads. Appropriate for traditional non-parallel workloads.

Amazon EC2 Intel Processor Specifications

Instance Family	Instance Type	Processor Arch	vCPU	ECU	Physical Processor	Intel® AES-NI	Intel® AVX†	Intel® Turbo
General purpose	m3.medium	64-bit	1	3	Intel Xeon E5-2670	Yes	Yes	Yes
General purpose	m3.large	64-bit	2	6.5	Intel Xeon E5-2670	Yes	Yes	Yes
General purpose	m3.xlarge	64-bit	4	13	Intel Xeon E5-2670	Yes	Yes	Yes
General purpose	m3.2xlarge	64-bit	8	26	Intel Xeon E5-2670	Yes	Yes	Yes
Compute optimized	c3.large	64-bit	2	7	Intel Xeon E5-2680 v2	Yes	Yes	Yes
Compute optimized	c3.xlarge	64-bit	4	14	Intel Xeon E5-2680 v2	Yes	Yes	Yes

Architecting on AWS – Designing for Cost



Training and
Certification 19

Notes:

The following table lists the processor specifications for each Amazon EC2 Instance Type. Here we have highlighted some of the Intel Instances. A complete list of Amazon EC2 Instance Types can be found online:
<http://aws.amazon.com/ec2/instance-types/>.

EC2 Billing Options

On-demand instances

Unix/Linux instances start at \$0.02/hour

Pay as you go for compute power

Low cost and flexibility

Pay only for what you use, no up-front commitments or long-term contracts

Use Cases:

Applications with short term, spiky, or unpredictable workloads;

Application development or testing

Reserved instances

1 or 3 year terms

Pay low up-front fee, receive significant hourly discount

Low Cost / Predictability

Use Cases:

Applications with steady state or predictable usage

Applications that require reserved capacity, including disaster recovery

Spot instances

Bid on unused EC2 capacity

Spot Price based on supply/demand, determined automatically

Cost / Large Scale, dynamic workload handling

Use Cases:

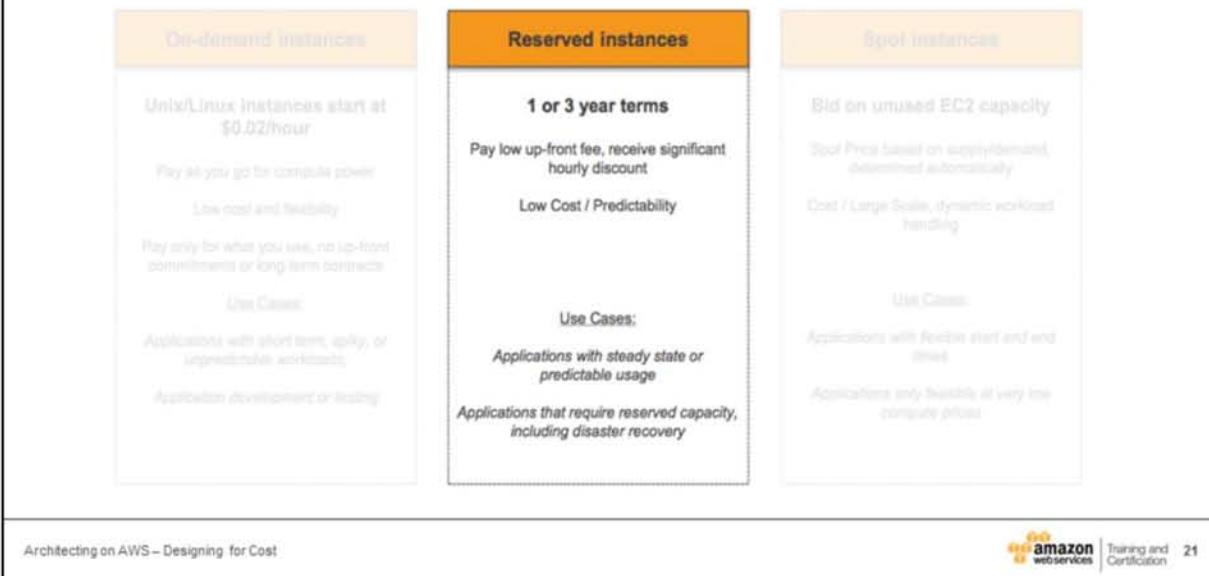
Applications with flexible start and end times

Applications only feasible at very low compute prices

Notes:

Most people new to AWS think of EC2 and on-demand pricing. This billing model offers a lot of flexibility—but it isn't always the most cost-effective one.

EC2 Billing Options



Architecting on AWS – Designing for Cost

Amazon web services | Training and Certification 21

Notes:

Reserved instances are great for when you know the baseline capacity your application needs. These instances are considerably less expensive than on-demand instances, and require a commitment of 1 or 3 years.

18 instance types are available: see <http://aws.amazon.com/ec2/reserved-instances/#3> for a complete list.

Reserved Instance Cost Savings Over On-Demand (m1.large – Linux – One Year RI)

Annual Utilization	On Demand	Light Utilization RI	Medium Utilization RI	Heavy Utilization RI
10%	\$234.00	-77.95%	-210.43%	-479.49%
20%	\$468.00	-18.97%	-73.68%	-189.74%
30%	\$702.00	0.68%	-28.09%	-93.16%
40%	\$936.00	10.51%	-5.30%	-44.87%
50%	\$1,170.00	16.41%	8.38%	-15.90%
60%	\$1,404.00	20.34%	17.49%	3.42%
70%	\$1,638.00	23.15%	24.00%	17.22%
80%	\$1,872.00	25.26%	28.89%	27.56%
90%	\$2,106.00	26.89%	32.69%	35.61%
100%	\$2,340.00	28.21%	35.73%	42.05%

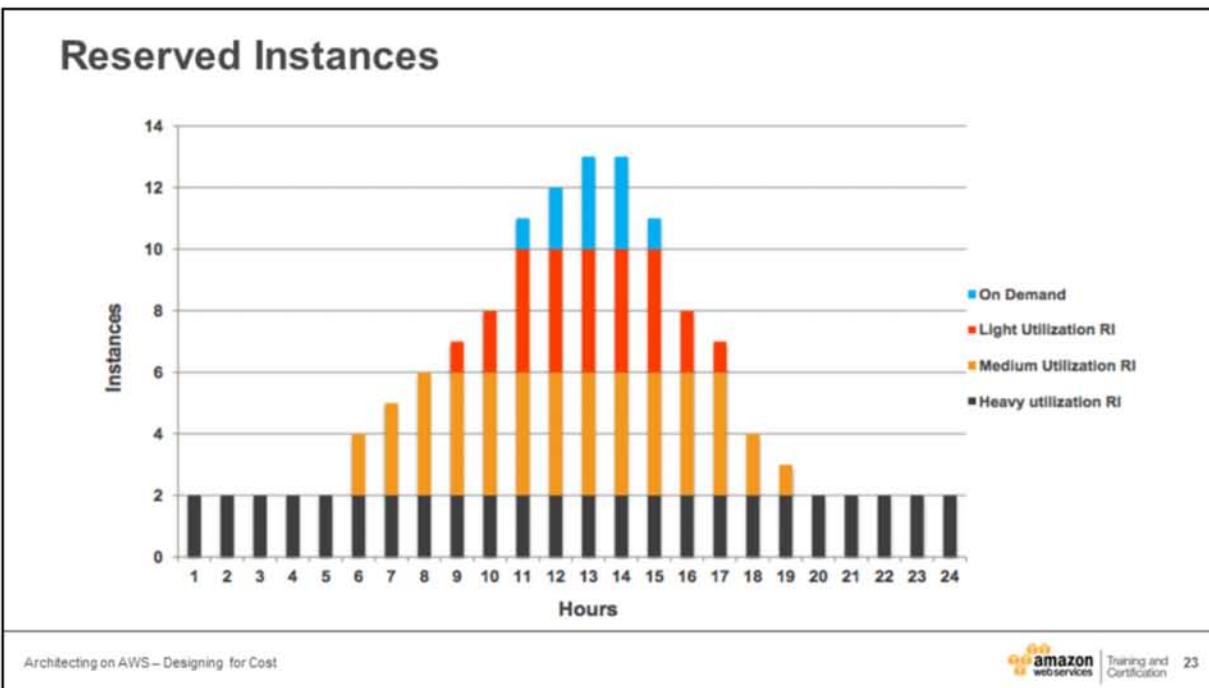
█ Optimal Savings

█ Sub-Optimal Savings

█ Least Savings

Notes:

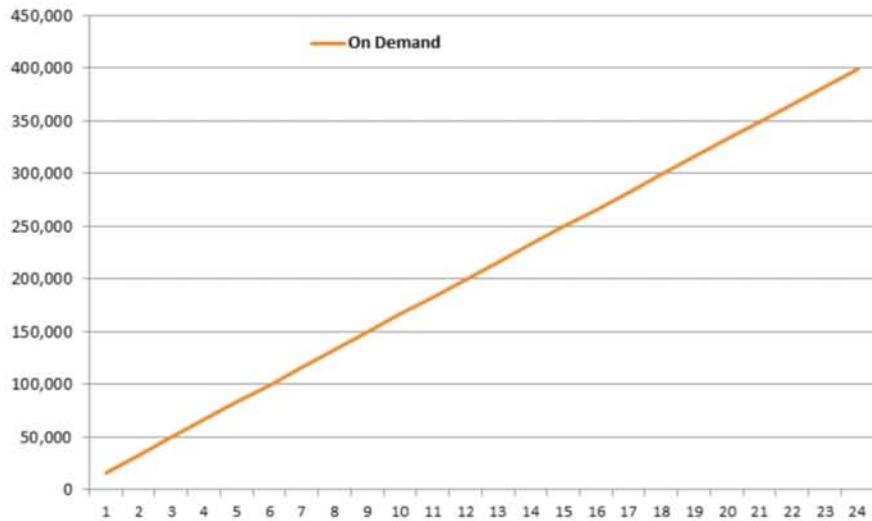
Here's a look at how reserved instances can reduce costs over on-demand instances.



Notes:

This chart shows how on-demand instances compare to reserved instances, depending on how heavily those reserved instances are used.

Reserved vs. On-Demand



Notes:

In this graph, the blue line shows the cost of on-demand instances over time.

Reserved vs. On-Demand

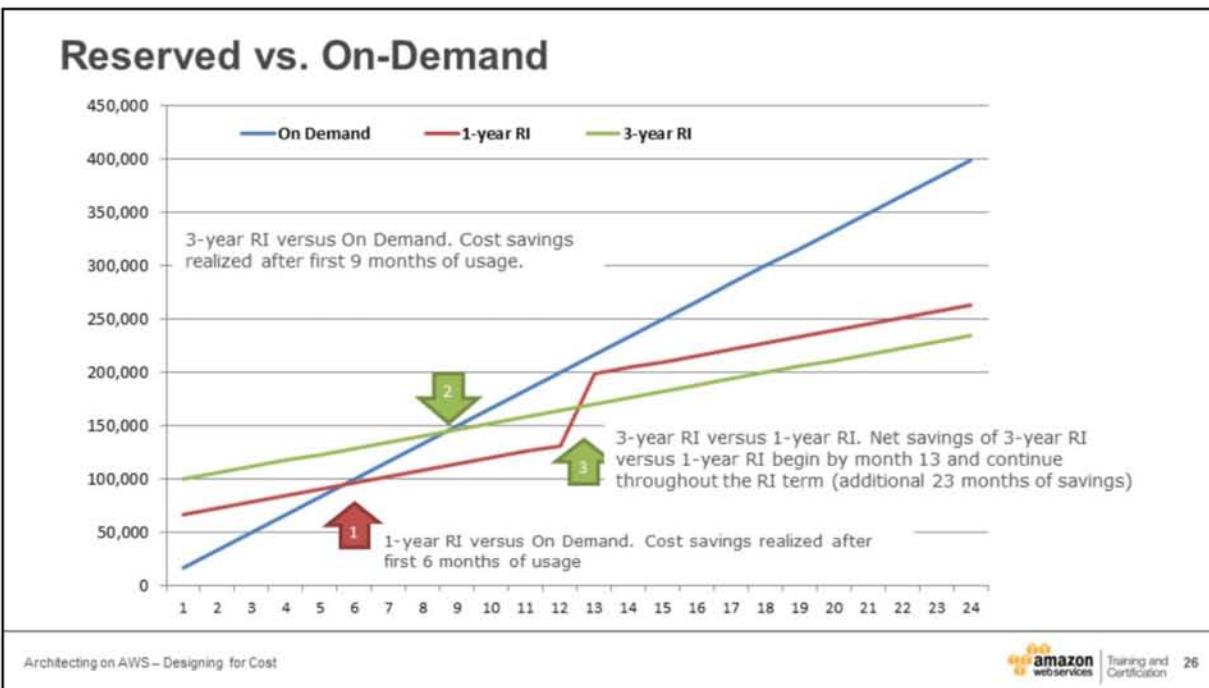


Architecting on AWS – Designing for Cost

amazon web services | Training and Certification 25

Notes:

This red line shows the cost of reserved instance for a 1 year term. As you can see, initially it's a little more expensive than on-demand instances, but over time, it is the less expensive option.



Notes:

The green line represents reserved instances with a 3-year commitment. Again, this is a little more expensive initially, but becomes the least expensive option over time.

Spot Market

- 💡 Spot instances often offer a significant savings over on-demand
- 💡 After an architecture is built for elasticity, leveraging spot instances can be a simple change

Notes:

EC2 offers another billing model: spot markets. Spot markets are best used when your system is already running and built for elasticity. (In fact, elasticity is critical for maximizing the use of the spot market.)

Please note the following important points:

- Spot Instances perform exactly like other Amazon EC2 instances while running. They only differ in their pricing model and the possibility of being interrupted when the Spot price exceeds your max bid.
- You will never pay more than your maximum bid price per hour. By placing your max bid at the maximum you're willing to pay per Spot instance-hour, you set the upper bound on your Spot computational costs.
- If your Spot instance is interrupted by Amazon EC2, you will not be charged for any partial hour of usage. For example, if your Spot instance is interrupted 59 minutes after it starts, we will not charge you for that 59 minutes. However, if you terminate your instance, you will pay for any partial hour of usage as you would for On-Demand Instances.
- You should always be prepared for the possibility that your Spot Instance may be interrupted. A high max bid price may reduce the probability that your Spot Instance will be interrupted, but cannot prevent interruption. If the Spot price exceeds your max bid or there is no longer spare EC2 capacity in a given Spot pool, your instances will be terminated.



Notes:

Here's a comparison of spot market instances versus on-demand. Notice that the spot price is considerably cheaper than on-demand.

Spot Market

Use Case	Types of Applications
Batch Processing	Generic background processing (scale out computing)
Hadoop	Hadoop/MapReduce processing type jobs (Search, Big Data, and so on)
Scientific Computing	Scientific trials/simulations/analysis in chemistry, physics, and biology
Video and Image Processing/Rendering	Transform videos into specific formats
Testing	Provide testing of software, web sites, etc.
Web/Data Crawling	Analyzing data and processing it
Financial	Hedge fund analytics, energy trading, etc.
HPC	Utilize HPC servers to do embarrassingly parallel jobs
Cheap Compute	Backend servers for Facebook games

Notes:

It takes a little more planning to make use of spot instances. Here are a few scenarios in which spot instances can be helpful.

Spot Market

💡 Best practices for using spot instances:

- Save your work frequently
- Add checkpoints
- Split up your work
- Test your application
- Use spot and on-demand in hybrid fashion → Master node in cluster is on-demand instance, worker nodes are Spot Instances

Notes:

It's important to use some best practices when using spot. Here are a few to consider.

- **Save Your Work Frequently:** Because Spot Instances can be terminated with no warning, it is important to build your applications in a way that allows you to make progress even if your application is interrupted. There are many ways to accomplish this, two of which are adding checkpoints to your application or splitting your work into small increments.
- **Add Checkpoints:** Depending on fluctuations in the Spot Price caused by changes in the supply or demand for Spot capacity, Spot Instance requests may not be fulfilled immediately and may be terminated without warning. In order to protect your work from potential interruptions, we recommend inserting regular checkpoints to save your work periodically. One way to do this is by saving all of your data to an Amazon EBS volume. Another approach is to run your instances using Amazon EBS-backed AMIs. By setting the
- DeleteOnTermination flag to false as part of your launch request, the Amazon EBS volume used as the instance's root partition will persist after instance termination, and you can recover all of the data saved to that volume. You can read more details on the use of Amazon EBS-backed AMIs here.
- **Note:** When using this technique with a persistent request, bear in mind that a new EBS volume will be created for each new Spot Instance.
- **Split up Your Work:** Another best practice is to split your workload into small increments if possible. Using Amazon SQS, you can queue up work increments

and keep track of what work has already been done (as in the example from the previous section). When using this approach, ensure that processing a unit of work is idempotent (can be safely processed multiple times) to ensure that resuming an interrupted task doesn't cause problems. You can do this by queuing a message to your Amazon SQS queue for each increment of work. You can then build an AMI that, when run, discovers the queue from which to pull its work. Discovery can be done by building it into the AMI, passing in user data or by storing the configuration remotely (for example in Amazon S3), which will tell the AMI in which queue to look. More details on using Amazon SQS with Amazon EC2 and a detailed walkthrough on how to set up this type of architecture can be found [here](#).

- **Test Your Application:** When using Spot Instances, it is important to make sure that your application is fault tolerant and will correctly handle interruptions. While we attempt to cleanly terminate your instances, your application should be prepared to deal with sudden shutdowns. You can test your application by running an On-Demand Instance and then terminating it. This can help you to determine whether your application is sufficiently fault tolerant and is able to handle unexpected interruptions.
- Configure your apps to spin up spot instances first; then fall back to on-demand instances if you don't get the spot instance in under 15 minutes.
- Mix the use of spot instances and on-demand instances.
- **You should always be prepared for the possibility that your Spot Instance may be interrupted.** A high max bid price may reduce the probability that your Spot Instance will be interrupted, but cannot prevent interruption. If the Spot price exceeds your max bid or there is no longer spare EC2 capacity in a given Spot pool, your instances will be terminated.

For more information, refer to the online documentation →
<http://aws.amazon.com/ec2/purchasing-options/spot-instances/>

Topics

- 💡 Cost model
- 💡 Services and feature costs
- 💡 Billing options
- 💡 Best practices

[Course Title - Module Title]



Notes:

While we're talking about best practices, let's go over some for managing cost in general.

Minimize Always On instances

- 💡 Elasticity is one of the fundamental properties of the cloud that drives many of its economic benefits
- 💡 Optimize your usage based on real-time demand
 - Reduce the number of web servers during off-peak periods
 - Shut down processing nodes unused at night
 - Purchase Reserved Instances for the Always On fleet

Notes:

Elasticity is critical for managing costs. Make sure you not only scale up when necessary, but scale down when demand diminishes.

Scale-in automatically

- 💡 Scaling out is key to serving customer needs but scaling in is where the savings occur
 - Auto-scaling works well for stateless components
 - Scripted scaling makes sense for stateful components
- 💡 Examples
 - Auto-scaling based on Network I/O for web servers
 - De-scaling RDS instances on the weekend
 - Shutting down all workers when batch processing queues are empty

Notes:

Scaling out is what most people think about in terms of moving services to the cloud. But scaling in is where you save money.

Scripted scaling

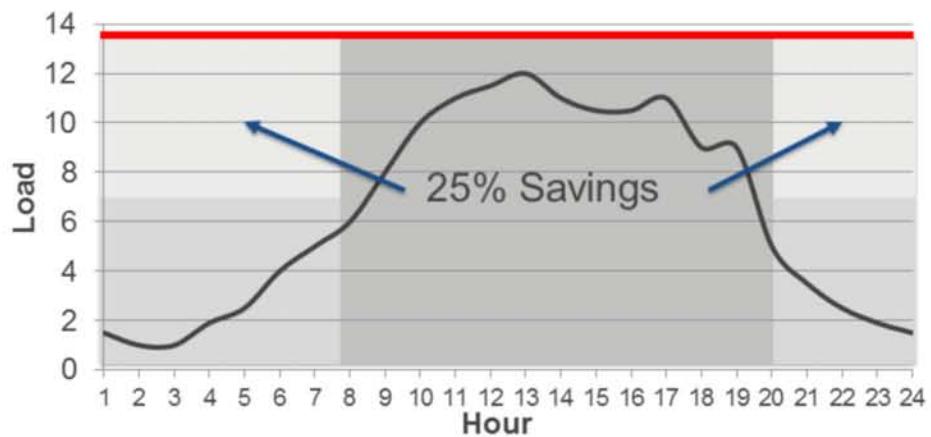
- 💡 Some scenarios do not lend themselves to automated de-scaling due to complex application logic
 - Shutting down worker nodes only when they are not currently working
 - Shutting down RDS read replicas during weekends
- 💡 Every AWS service has an API and command line tools for managing it
 - Scripting of scaling activities can be a significant cost savings for little effort

Notes:

Make use of scripts to help your applications scale in effectively. Often applications are too complex for automated scaling. Scripts can help a lot here.

Optimize by time of day

Daily CPU Load



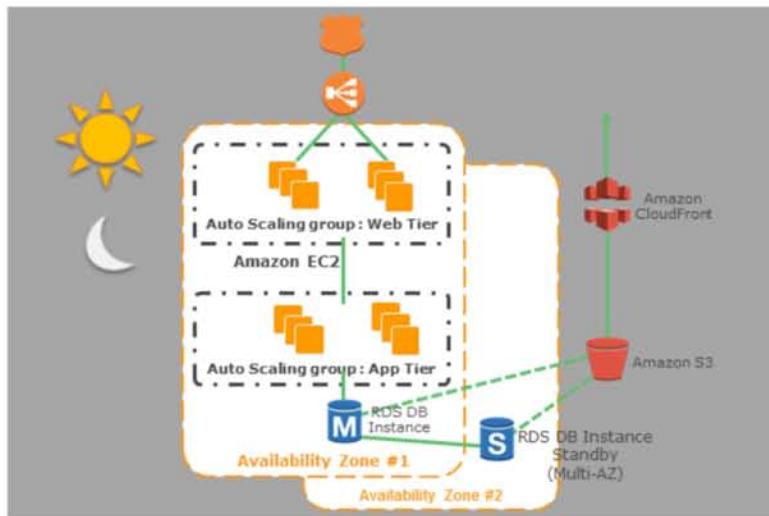
Architecting on AWS – Designing for Cost

amazon
web services | Training and
Certification 35

Notes:

One way scripts can help: scaling resources in based on the time of day.

Optimize by time of day



[Course Title - Module Title]
Architecting on AWS – Designing for Cost

amazon
aws web services | Training and Certification 36

End of month processing

- 💡 Expand the cluster at the end of the month
 - Expand/Shrink feature in Amazon Elastic MapReduce
- 💡 Vertically Scale up at the end of the month
 - Modify-DB-Instance (in Amazon RDS) (or a New RDS DB Instance)
 - CloudFormation Script (in Amazon EC2)

Notes:

Another option: adjust scaling to account for the demands of end-of-month processing.

Optimize during the month



Notes:

This graph shows the potential cost savings for this type of configuration.

Leverage scalable, on-demand services

- 💡 EC2 can run almost anything but there are many cases where it is not cost effective
- 💡 AWS offers many scalable and cost-effective options for common application needs:
 - ELB instead of a software load balancer on EC2
 - SQS instead of a queue on EC2

Notes:

It's tempting to simply spin up EC2 instances and install whatever apps you want on them. However, there are often more cost-effective options.

Best Practices

Software LB on EC2

Pros

- Application-tier load balancer

Cons

- SPOF
- Elasticity has to be implemented manually
- Not as cost-effective

ELB

Pros

- Pay as you go
- Scalability
- Availability
- High performance

[Course Title - Module Title]



Notes:

Here's an example of how EC2 may not always be the best choice. Notice that a software load balancer on EC2 might be more familiar to implement, but isn't as effective as Amazon Elastic Load Balancer.

Best Practices (continue)

\$0.025
per hour



VS.

\$0.06
per hour
(m1.small)



Architecting on AWS – Designing for Cost

amazon web services | Training and Certification 41

Notes:

From a cost perspective, Amazon Elastic Load Balancer is much less expensive than a software load balancer running on EC2.

Best Practices

Software on EC2

Pros

- Custom features

Cons

- Requires an instance
- SPOF
- Limited to one AZ
- DIY administration

SNS, SQS, SES

Pros

- Elastic and Fault-tolerant
- Auto scaling
- Monitoring included
- IPV6

Cons

- Internal load balancing only in VPC

[Course Title - Module Title]



Notes:

Messaging services, such as Amazon Simple Queue Service, are also an important way to save costs.

Best Practices

\$0.01
per
10,000 Requests
(\$0.000001 per Request)



\$0.095
per hour
(small instance)



Architecting on AWS – Designing for Cost

Amazon web services | Training and Certification 43

Notes:

Again, setting up a software queue on EC2 might be more familiar, but it's nearly 10x more expensive than using SQS.

Clean up after yourself

- 💡 When it is easy to create resources, it can be easy to forget about them
 - Use tagging to identify the purpose of resources
 - Use CloudWatch to identify underutilized resources
 - Keep track of objects in S3 and clean up unused content
 - Release unused Elastic IPs
- 💡 Examples
 - Daily report on utilization of resources
 - Clean up script to delete old S3 objects
- 💡 Make use of Trusted Advisor

Notes:

Another important aspect of managing costs on AWS: clean up after yourself! Keep track of what resources you are using, and shut down resources that aren't being used. Use tags to stay informed of what resources are performing what role.

Economics Center

<http://aws.amazon.com/economics>

Notes:

The AWS economics center is a great resource to help you manage costs. For example, the TCO comparison calculator for Web applications allows you to compare the cost of AWS versus running Web applications on-premises:
<http://aws.amazon.com/tco-calculator/>

Copyright © 2013, 2014 Amazon Web Services, Inc. and its affiliates. All rights reserved.

This work may not be reproduced or redistributed, in whole or in part, without prior written permission from Amazon Web Services, Inc. Commercial copying, lending, or selling is prohibited.

Errors or corrections? Email us at aws-course-feedback@amazon.com.
Other questions? Email us at aws-training-info@amazon.com.

All trademarks are the property of their owners.



Module 11: Disaster Recovery and High Availability

Architecting on AWS – Disaster Recovery and High Availability

 Amazon
Training and
Certification 1

Topics

- 💡 How High Availability and Disaster Recovery work together
- 💡 Building highly available systems on AWS
- 💡 Best practices for high availability and disaster recovery
- 💡 Common patterns of disaster recovery on AWS

Notes:

In this module, we'll cover these key concepts.

Topics

- 💡 How High Availability and Disaster Recovery work together
- 💡 Building highly available systems on AWS
- 💡 Best practices for high availability and disaster recovery
- 💡 Common patterns of disaster recovery on AWS

Notes:

Let's first take a look at the relationship between High Availability and Disaster Recovery.

How Availability and Disaster Recovery

- 💡 Think of it as a spectrum
- 💡 It's part of a business continuity plan
- 💡 It's not an all or nothing proposition
- 💡 In the face of internal or external events, how do you...
 - Keep your applications running 24x7
 - Make sure your data is safe
 - Get an application back up after a major disaster



Notes:

It's important to consider these two concepts as two parts of the same solution. Consider both high availability and disaster recovery at the same time—even if your immediate concerns focus more on one part than the other.

How Availability

- 💡 HA is one end of the spectrum:
 - Rather than recovery with defined RTO/RPO, design for continuous availability
- 💡 Goal: application never goes down
 - Eliminate single points of failure
 - “Self Healing” app recovers automatically from component failures
 - Uses graceful degradation if necessary
- 💡 Traditional IT model: HA is very expensive, suitable only for absolutely mission-critical apps

Notes:

Let's review some key concepts of highly-available systems.

How Availability Terms

- 💡 Uptime: period when system is available for use
- 💡 Downtime (or Outage): period when system is unavailable for normal function or offline
- 💡 Graceful Degradation: system continues to be available, but at a reduced level of service or function
- 💡 Availability: percentage uptime in a given period (nines)
 - 99.999% ("five nines") availability = no more than about 5 minutes downtime per year

Notes:

A few terms to make sure we're all on the same page.

Disaster Recovery on the spectrum

- 💡 Recover from any event
- 💡 Recovery Time Objective (RTO)
 - Acceptable time period within which normal operation (or degraded operation) needs to be restored after event
- 💡 Recovery Point Objective (RPO)
 - Acceptable data loss measured in time
- 💡 Traditional IT model has DR in a second physical site
 - Low end DR: off-site backups
 - High end DR: hot site active-active architecture

Notes:

Now, let's review disaster recovery.

How Does AWS Change Traditional DR / HA?

- AWS is useful for low-end traditional DR to high-end HA, but...
- AWS encourages a rethinking of traditional DR / HA practices
 - Everything in the cloud is "off-site" and (potentially) "multi-site"
 - Using multiple sites (multiple AZs) comes largely for free
 - Using multiple geographically-distributed sites (multiple Regions) is significantly cheaper and easier
- Tends to move the default design point away from "cold" Disaster Recovery toward "hot" High Availability
- Makes it easier to stack multiple mechanisms
 - e.g., Basic HA within one Region, DR site in second Region

Notes:

You may be familiar with High Availability and Disaster Recovery. Keep in mind that, to make the most of AWS products and services, you need to adapt your concepts and practices a little.

For example, with AWS, you get the resilience of multiple availability zones with little to no additional cost.

Another important consideration:

With AWS, you can think more in terms of high availability, and less in terms of disaster recovery. This is because, with AWS, you can build systems that can better withstand failures, outages, and so on.

Topics

- 💡 How High Availability and Disaster Recovery work together
- 💡 Building highly available systems on AWS
- 💡 Best practices for high availability and disaster recovery
- 💡 Common patterns of disaster recovery on AWS

Notes:

Now, we'll talk about how to build a highly available system on AWS.

AWS: Regions and Availability Zones

- Regions are completely separate clouds
- Multiple network-connected Availability Zones in each Region



Notes:

Understanding and using multiple Availability Zones is probably the single most important concept for implementing DR and building HA applications!

Each Region is conceptually a separate, independent cloud. Within a Region, each Availability Zone is conceptually a separate, fault-isolated data center.

Remember that Availability Zones have these traits:

- AZs are in distinct physical locations
- Low-latency network connections between AZs
- Independent power, cooling, network, security
- Always partition app stacks across 2 or more AZs
- Elastic Load Balance across AZs

Take advantage of multiple availability zones

- 💡 No cost difference between servers running in a single AZ versus multiple AZs
- 💡 Most services are designed with multiple AZ use in mind

Notes:

Use multiple Availability Zones! Most AWS services run across multiple AZs in a region. This practice helps your applications become highly available.

Next step: Building across regions

💡 Advantages

- reduce latency
- increase throughput

💡 Challenges

- Most services operate within a single region
 - S3, Auto Scaling, etc.
 - Several tools can help:
AMI copy, EBS Snapshot, Route 53, CloudFormation
 - Some AWS services are region-independent
Management Console, IAM, Route 53

Notes:

After you have a system up and running in one region, across multiple Availability Zones, you can start thinking about how to employ other regions.

There are a few reasons to consider cross-region apps:

- Reduce latency
- Keep data to one geographic region
- Disaster recovery

Don't treat regions as you would traditional data centers. In many cases, multiple-availability zones provides all the durability you might need.

Fault-Tolerant Building Blocks

- 💡 Many services are inherently fault-tolerant and highly available:
 - S3, SQS, RDS, DynamoDB, ELB, Route 53
- 💡 Services for building highly available apps:
 - Availability Zones, Elastic IPs, EBS/Snapshots, ELB, Auto Scaling, Route 53, etc.
 - EC2 (On-Demand and Reserved), AMIs, etc...

Notes:

For example, many AWS services make use of multiple availability zones to become fault-tolerant and highly available.

AWS Features: Compute

- 💡 Amazon Elastic Compute Cloud (EC2)
 - Virtual hosts that can be provisioned momentarily
- 💡 AMI – Amazon Machine Image
 - Packaged, reusable functionality: OS + software stack
 - Basic unit of EC2 deployment
- 💡 Reserved Instances
 - Reserved capacity when it is needed
 - For further cost savings, can be used for non-critical workloads when not used for DR

Notes:

As you build your systems, here are a few compute resources that are designed with high availability in mind.

AWS Features: Storage

- Amazon Simple Storage Service (S3)
 - Highly-durable file storage for archival and backup
- Elastic Block Store (EBS) and EBS Snapshots
 - Persistent Data volumes for EC2 instances
 - Redundant within a single Availability Zone
 - Snapshot backups provide long-term durability, and volume sharing / cloning capability within a Region

Notes:

Amazon S3 and Amazon EBS are storage options that are highly durable and fault-tolerant.

AWS Features: Data Migration

- ─ AWS Storage Gateway
- ─ Amazon VM Import
- ─ Amazon Import/Export
- ─ Other Mechanisms
 - Backup systems
 - Replication (mirroring, db replication, log shipping, etc.)
 - Managed File Transfer products
 - Scripted rsync, tsunami, etc.

Notes:

If you need to move your data from one region to another, or (more likely) from your on-premises data center to AWS, you have several options available to you.

AWS Features: Networking

- Amazon Virtual Private Cloud (VPC) & Direct Connect
 - VPC provisions an isolated area within EC2
 - DX provides dedicated fiber cross-connect
- Elastic IP Addresses (EIP)
 - Public Static IP addresses allocated to AWS account
 - EIP can be mapped / remapped to any running instance within a Region
- Amazon Route 53
 - Highly available DNS Service
 - Can quickly map URLs or hostnames between Regions
 - Supports Weighted Round Robin (even across Regions)

Notes:

Several AWS networking resources are built to support highly-available systems.
Here are a few examples.

AWS Features: Monitoring / Scaling / Load Balancing

- CloudWatch Monitoring
 - Metrics and Alarms for AWS resources
 - Use in conjunction with open source and third-party monitoring tools
- Auto Scaling
 - Respond to changing conditions by adding or terminating EC2 instances (triggered by CloudWatch alarm)
 - Maintain a fixed number of instances running, replacing them if they fail or become unhealthy
- Elastic Load Balancing (ELB)
 - Distributes incoming traffic across multiple instances, in single AZ or across AZs
 - Sends traffic only to healthy instances

Notes:

AWS services such as CloudWatch, and features such as Auto Scaling, can also be valuable tools as you build high availability into your applications.

A simple example:

Create an Auto Scaling group that has a minimum/maximum number of systems equal to 1. This means you always have one system up and running at any given time.

AWS Features: Automation

- 💡 “Everything is an API” philosophy enables automation of AWS resources
- 💡 AWS CloudFormation

Notes:

Make use of our APIs to further improve the resiliency of your applications.
Don't forget to use AWS CloudFormation to help you with creating systems in new regions or to improve your application's availability.

You can have CloudFormation templates ready, and launch them from the API (don't need someone to log in), so we can automate the DR process.

Topics

- 💡 How High Availability and Disaster Recovery work together
- 💡 Building highly available systems on AWS
- 💡 Best practices for high availability and disaster recovery
- 💡 Common patterns of disaster recovery on AWS

Notes:

We've discussed some of the building blocks for highly available applications. Let's look at a few best practices you should consider as you put these building blocks to work.

Best Practices for HA

- 💡 Build loosely coupled systems
 - Queues, load-balance tiers
- 💡 Implement elasticity
 - Bootstrapping, load balancing, Auto Scaling, etc...
 - Instance asks: "Who am I and what is my role?"
- 💡 Use abstract machine and system representations
 - Build images from recipes, stacks from CloudFormation

Notes:

Most of the best practices discussed in Architecting in the Cloud (on day 1) apply to building highly available applications.

Design for failure: Basic principles

- 💡 Goal: Applications should continue to function even if the underlying physical hardware fails or is removed or replaced.
 - Avoid single points of failure
 - Assume everything fails, and design backwards
 - Design your recovery process

Notes:

One way to think about high availability: it's targeted disaster recovery applied to each component in your application.

Best Practices: Test Your Application



<http://techblog.netflix.com/2010/12/5-lessons-weve-learned-using-aws.html>

Architecting on AWS – Disaster Recovery and High Availability

amazon
web services | Training and
Certification 23

Notes:

Of course, the best way to ensure your application is highly available is to test it. One tool that you can try is Chaos Monkey.

Best Practices: Test Your Application

From the Netflix blog:

One of the first systems our engineers built in AWS is called the Chaos Monkey. The Chaos Monkey's job is to randomly kill instances and services within our architecture. If we aren't constantly testing our ability to succeed despite failure, then it isn't likely to work when it matters most – in the event of an unexpected outage.

Simple monkey:

- Kill any instance in the account

Complex monkey:

- Kill instances with specific tags
- Introduce other faults (e.g. connectivity via Security Group)

Human monkey:

- Kill instances from the AWS Management Console

<http://techblog.netflix.com/2010/12/5-lessons-weve-learned-using-aws.html>

Architecting on AWS – Disaster Recovery and High Availability



Notes:

Chaos Monkey came from Netflix, and uses a variety of tests to ensure that application components are highly available.

Topics

- 💡 How High Availability and Disaster Recovery work together
- 💡 Building highly available systems on AWS
- 💡 Best practices for high availability and disaster recovery
- 💡 Common patterns of disaster recovery on AWS

Notes:

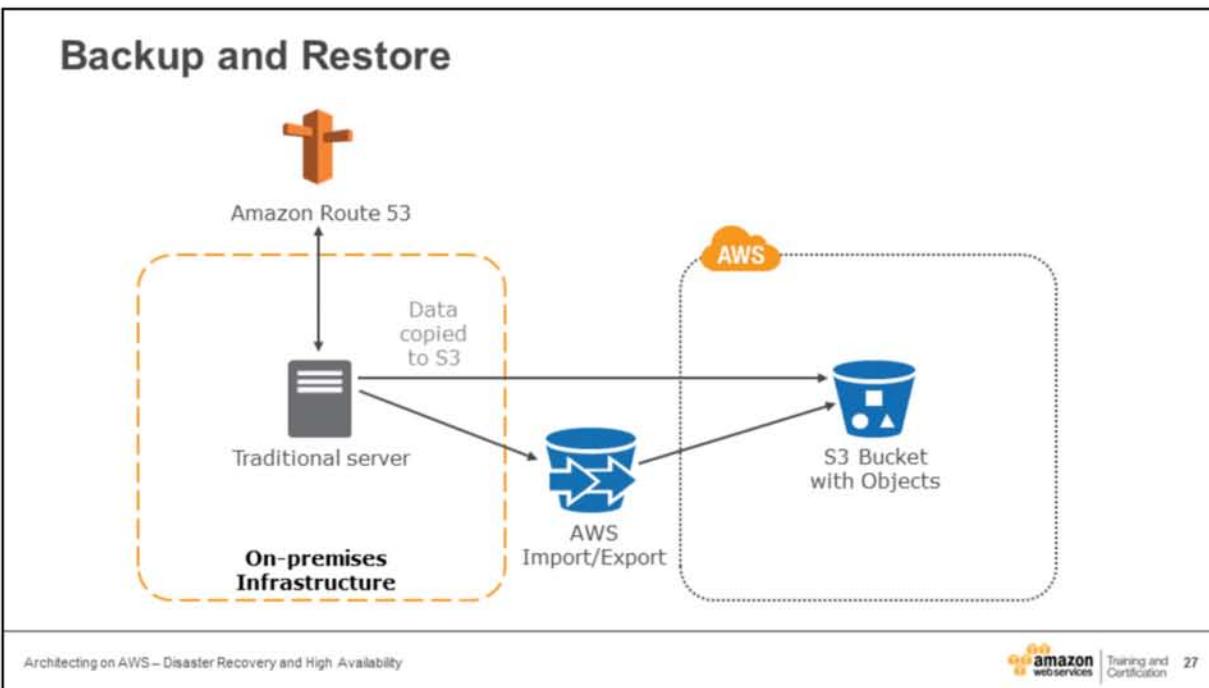
Now, we'll move to the other end of the spectrum and talk about disaster recovery.

Common Practices of Disaster Recovery on AWS

- 💡 Example Architectural Patterns (sorted by increasingly optimal RTO/RPO)
 - Backup and Restore
 - Pilot Light
 - Fully Working Low Capacity Standby
 - Multi-Site Hot Standby

Notes:

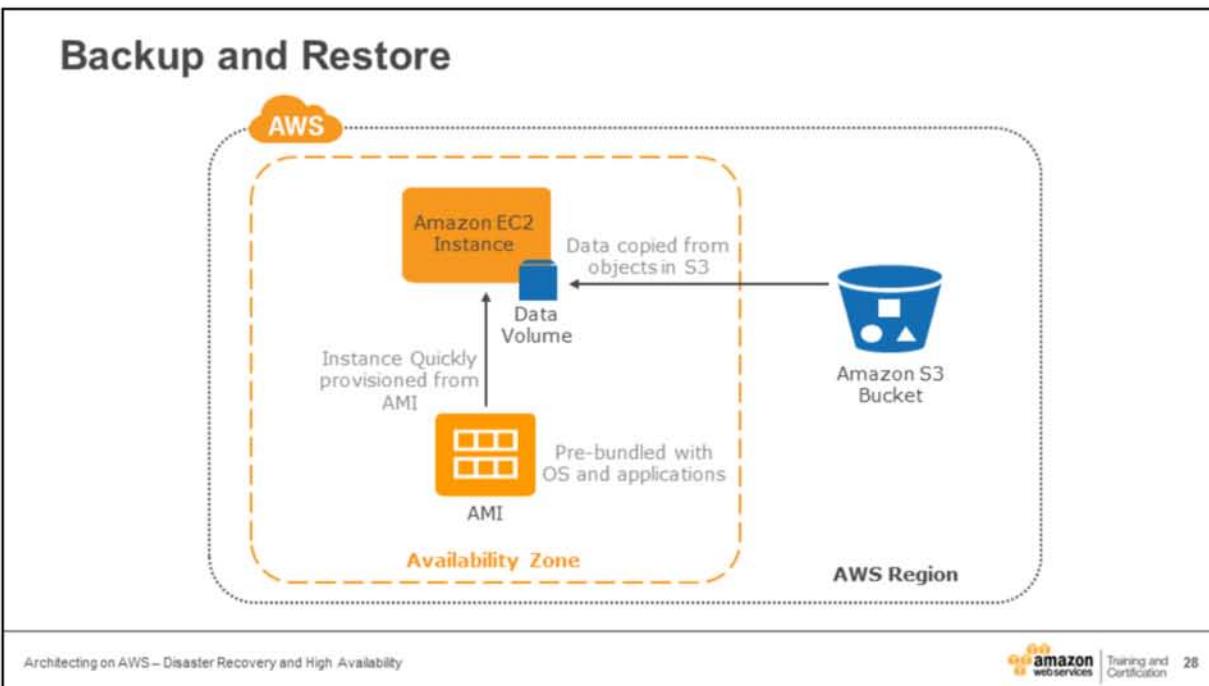
Here are a few different disaster recovery patterns to follow.



Notes:

This diagram shows a basic disaster recovery system.

In this diagram, our data is in a traditional data center, with backups stored in AWS. We use AWS Import/Export to get our data into AWS, and store the data in Amazon S3.



Notes:

To recover our data, we use an AMI (custom or pre-configured) to create one or more EC2 instances.

Then, we add our S3 data by copying it to the EC2 instances.

Backup and Restore

💡 Advantages

- Simple to get started
- Extremely cost effective (mostly backup storage)

💡 Preparation Phase

- Take backups of current systems
- Store backups in S3
- Describe procedure to restore from backup on AWS
 - Know which AMI to use, build your own as needed
 - Know how to restore system from backups
 - Know how to switch to new system
 - Know how to configure the deployment

Notes:

This disaster recovery pattern is easy to set up, and pretty inexpensive.
You do need to think about what AMIs you need for recovery.

Common Practices of Disaster Recovery on AWS

💡 In case of disaster

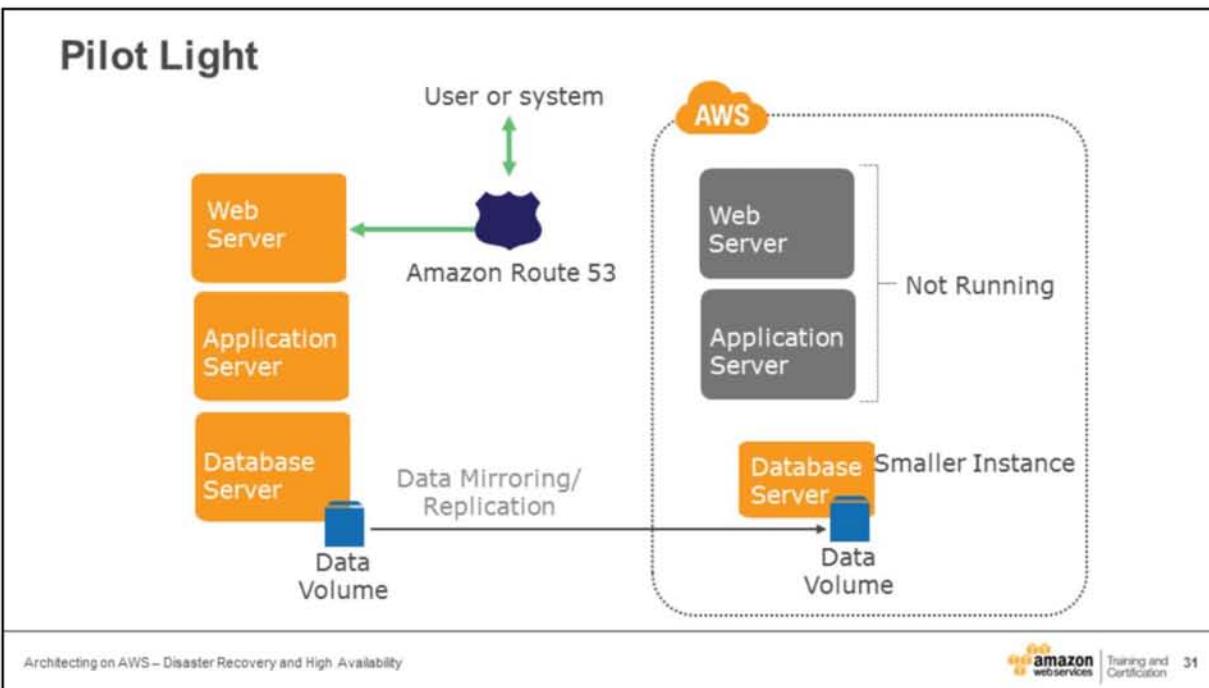
- Retrieve backups from S3
- Bring up required infrastructure
 - EC2 instances with prepared AMIs, Load Balancing, etc.
- Restore system from backup
- Switch over to the new system
 - Adjust DNS records to point to AWS

💡 Objectives

- RTO: as long as it takes to bring up infrastructure and restore system from backups
- RPO: time since last backup

Notes:

Here's how this pattern works in practice.

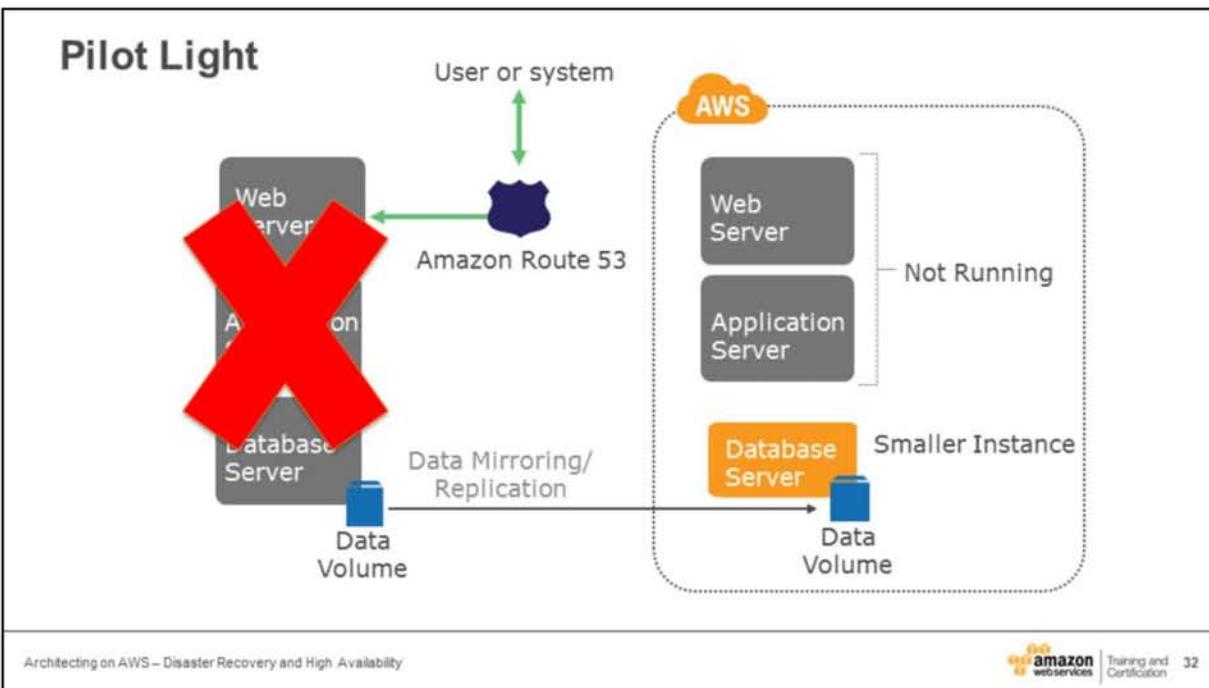


Notes:

This diagram shows a Pilot Light disaster recovery pattern.

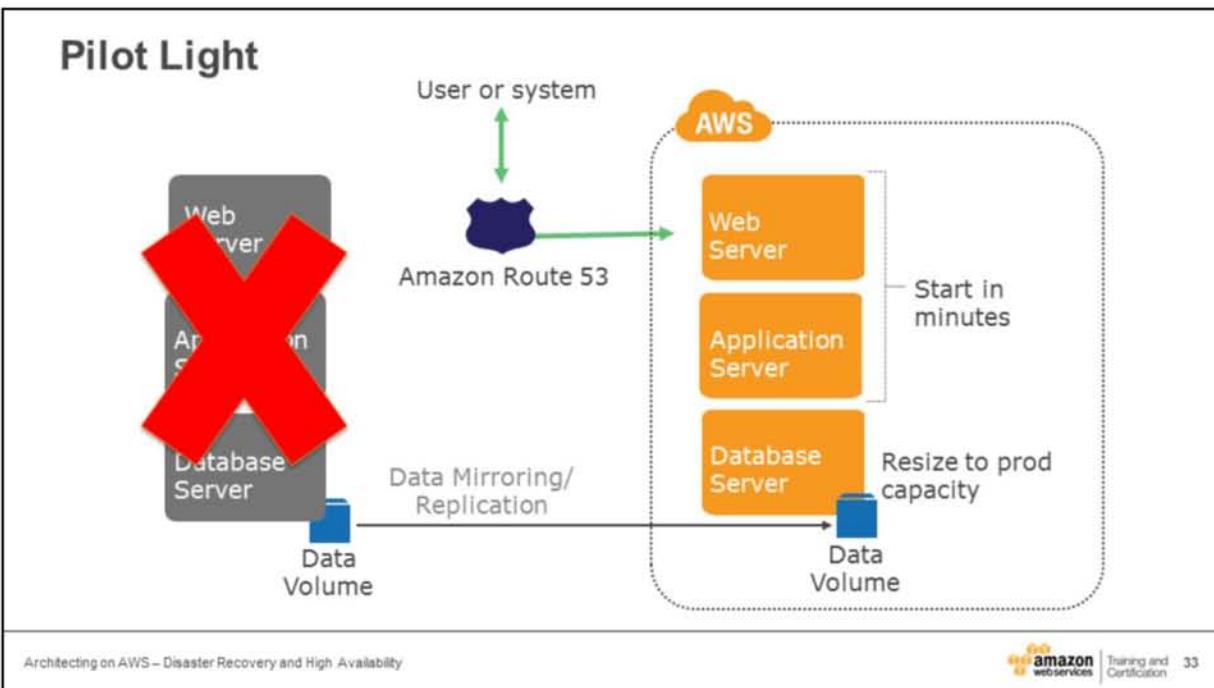
In this case, we have our database data replicated into AWS, with EC2 instances of our web servers and application servers ready to go, but currently not running.

Notice that we've Amazon Route 53 in use here as well.



Notes:

Now, should our primary system fail, when we start up our EC2 instances, it uses the latest copy of our data.



Notes:

Then we redirect Amazon Route 53 to point to the new web server.

Pilot Light

💡 Advantages

- Very cost effective (fewer 24/7 resources)

💡 Preparation Phase

- Enable replication of all critical data to AWS
- Prepare all required resources for automatic start
 - AMIs, Network Settings, Load Balancing, etc.
- Reserved Instances

Notes:

Again, this pattern is relatively inexpensive to implement.

Pilot Light

💡 In case of disaster

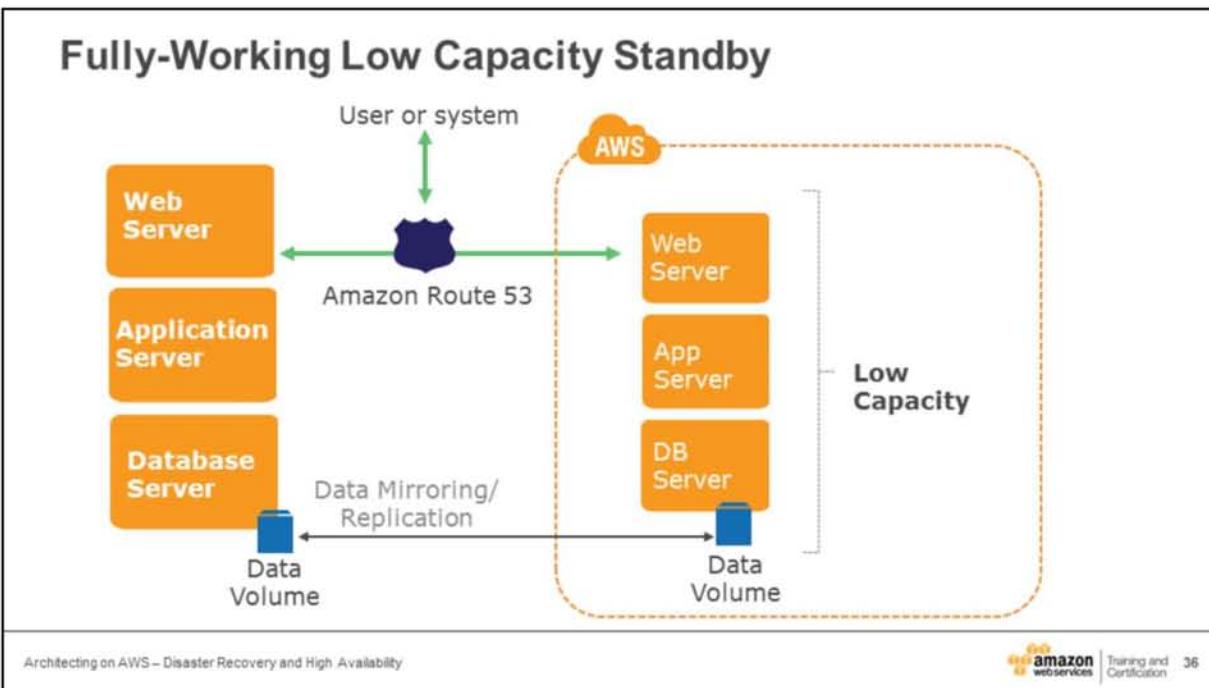
- Automatically bring up resources around the replicated core data set
- Scale the system as needed to handle current production traffic
- Switch over to the new system
 - Adjust DNS records to point to AWS

💡 Objectives

- RTO: as long as it takes to detect need for DR and automatically scale up replacement system
- RPO: depends on replication type

Notes:

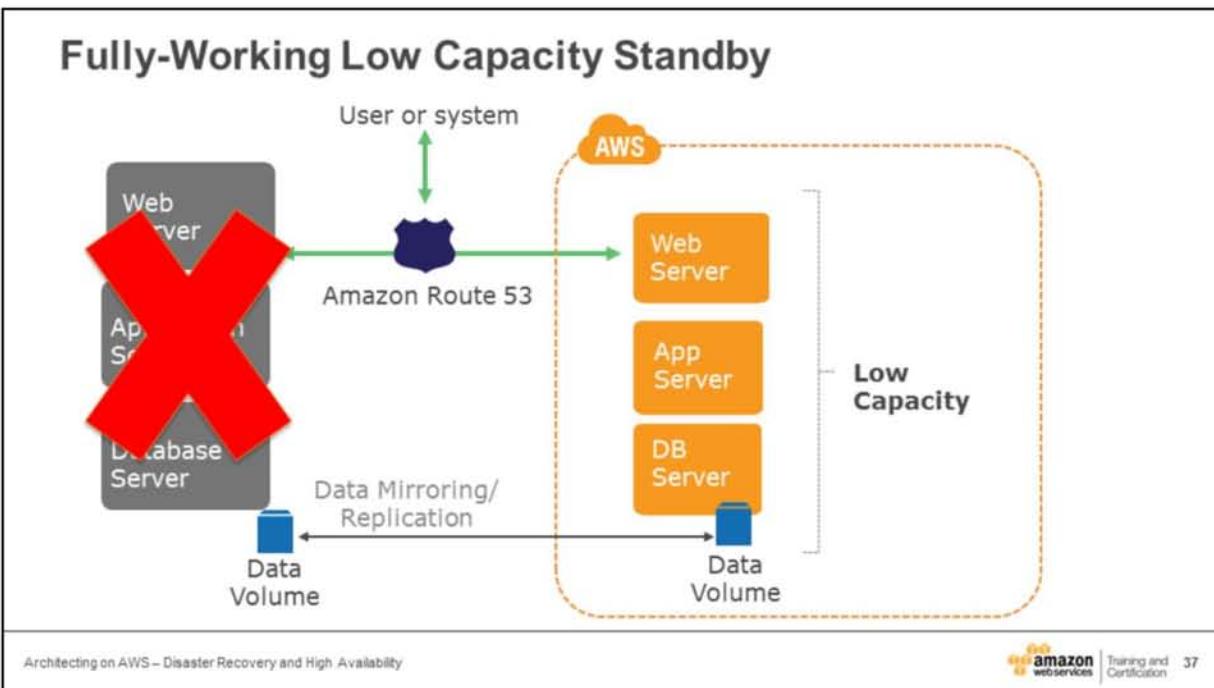
Here's how the pattern works in practice.



Notes:

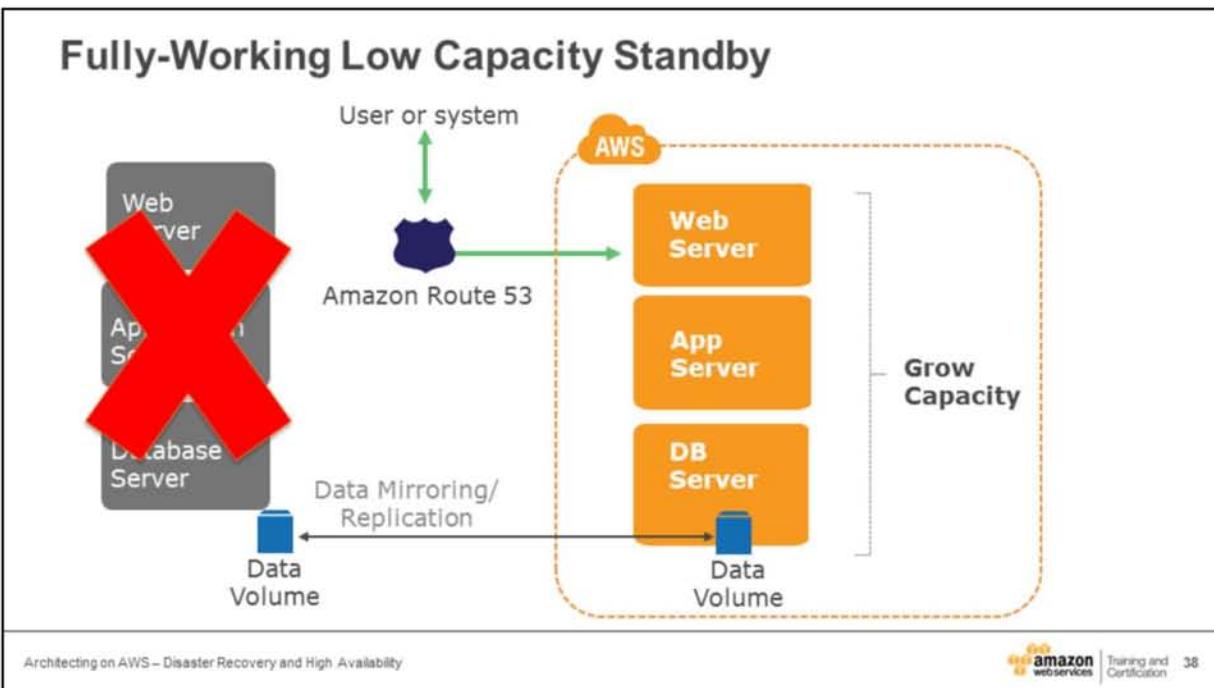
Low capacity standby is like the next level of Pilot Light.

Here, we have two systems running: our main system, and a low-capacity system running on AWS. We use Amazon Route 53 to distribute requests between our main system and our cloud system.



Notes:

Should our main system go down...



Fully-Working Low Capacity Standby

💡 Advantages

- Can take some production traffic at any time
- Cost savings (IT footprint smaller than full DR)

💡 Preparation

- Similar to Pilot Light
- All necessary components running 24/7, but not scaled for production traffic
- Best practice – continuous testing
 - “Trickle” a statistical subset of production traffic to DR site

Notes:

This pattern is more expensive, because we have active systems running.

Fully-Working Low Capacity Standby

💡 In case of disaster

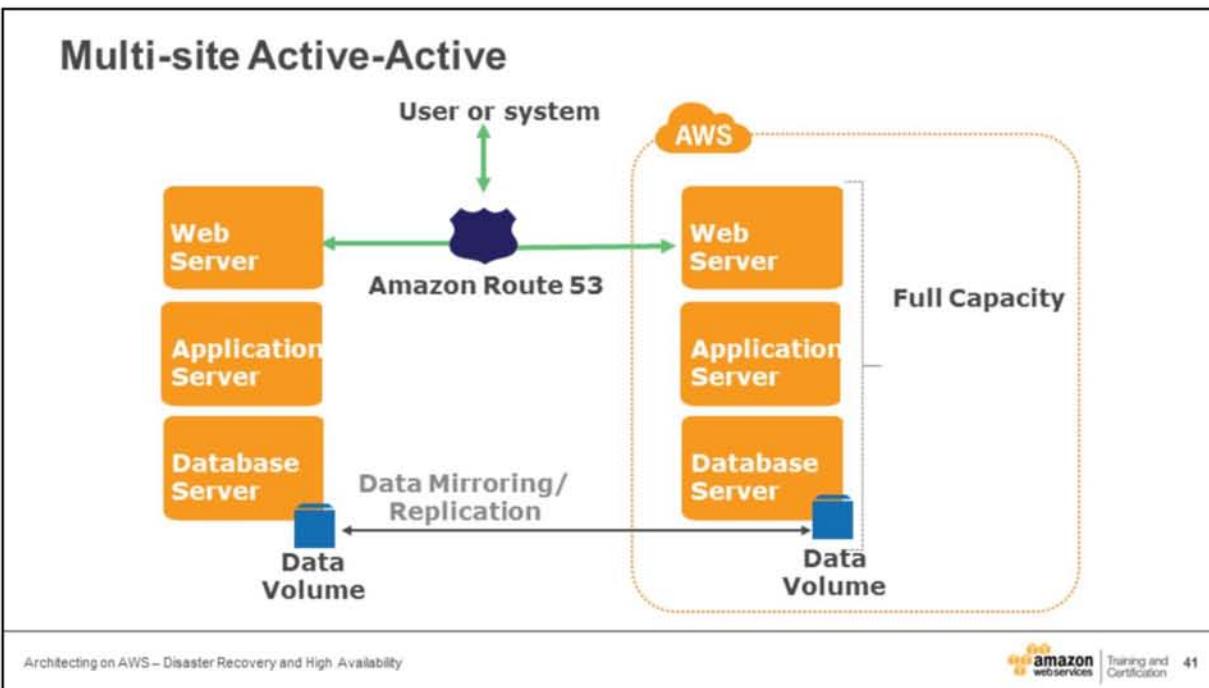
- Immediately fail over most critical production load
 - Adjust DNS records to point to AWS
- (Auto) Scale the system further to handle all production load

💡 Objectives

- RTO: for critical load: as long as it takes to fail over; for all other load, as long as it takes to scale further
- RPO: depends on replication type

Notes:

Here's how this pattern works in practice.



Notes:

And, of course, the next level of disaster recovery is to have a fully functional system running in AWS at the same time as our on-premises systems.

Multi-site Active-Active

Advantages

- At any moment can take all production load

Preparation

- Similar to Low-Capacity Standby
- Fully scaling in/out with production load

In Case of Disaster

- Immediately fail over all production load
 - Adjust DNS records to point to AWS

Objectives

- RTO: as long as it takes fail over
- RPO: depends on replication type

Notes:

This pattern potentially has the least downtime of all of the patterns we're discussing today. It does, of course, have more costs associated with it, because you have more systems running.

Hosted Desktops

💡 Advantages

- Replacement of workstations in case of disaster
- Pay only when used for DR

💡 Preparation

- Set up AMIs with appropriate working environment

💡 In Case of Disaster

- Launch desktop AMI and resume work

💡 Objectives

- RTO: as long as it takes to launch AMI and restore work environment on virtual desktop
- RPO: depends on state of AMI

Notes:

With AWS, you can also use hosted desktops as part of your disaster recovery process.

Best Practices for Being Prepared

- 💡 Start simple and work your way up
 - Backups in AWS as a first step
 - Incrementally improve RTO/RPO as a continuous effort
- 💡 Check for any software licensing issues
- 💡 Exercise your DR Solution
 - Game Day
 - Ensure backups, snapshots, AMIs, etc. are working
 - Monitor your monitoring system

Notes:

As you start thinking about disaster recovery and AWS, here are a few things to think about.

- Start small and build as needed
- Check your licenses!
- Test your solutions often

Conclusion – Advantages of disaster recovery with AWS

- Various building blocks available
- Fine control over cost vs. RTO/RPO tradeoffs
- Ability to scale up when needed
- Pay for what you use, and only when you use it (when an event happens)
- Ability to easily and effectively test your DR plan
- Availability of multiple locations world wide
- Hosted desktops available
- Variety of Solution Providers

Notes:

Let's review some advantages of disaster recovery on AWS.

Putting It Together: Incremental Improvement

- Start with existing on-premise app with traditional DR
- Gradually move DR (and thus the app) to the cloud:
 - Backup (to S3) and Restore
 - “Pilot Light” on AWS for Quick Recovery
 - Fully Working Low Capacity Standby on AWS
- Migrate to primary on AWS, DR on-premise
- Add hot standby in second AWS AZ or Region
- Incrementally add HA features to primary app
- End State: Use HA techniques in-Region, use DR techniques for Region-to-Region resiliency

Notes:

How do you start small and build up? Here's a set of steps that are pretty common.

Copyright © 2013, 2014 Amazon Web Services, Inc. and its affiliates. All rights reserved.

This work may not be reproduced or redistributed, in whole or in part, without prior written permission from Amazon Web Services, Inc. Commercial copying, lending, or selling is prohibited.

Errors or corrections? Email us at aws-course-feedback@amazon.com.
Other questions? Email us at aws-training-info@amazon.com.

All trademarks are the property of their owners.



Module 12: Migrating Applications to the AWS Cloud

Architecting on AWS – Migrating Applications to the AWS Cloud

 Amazon
Training and
Certification 1

Topics

- 💡 Defining cloud strategies
- 💡 Planning migrations
- 💡 Deploying applications
- 💡 Optimizing applications

Notes:

Here are the main areas we'll cover in this module.

Topics

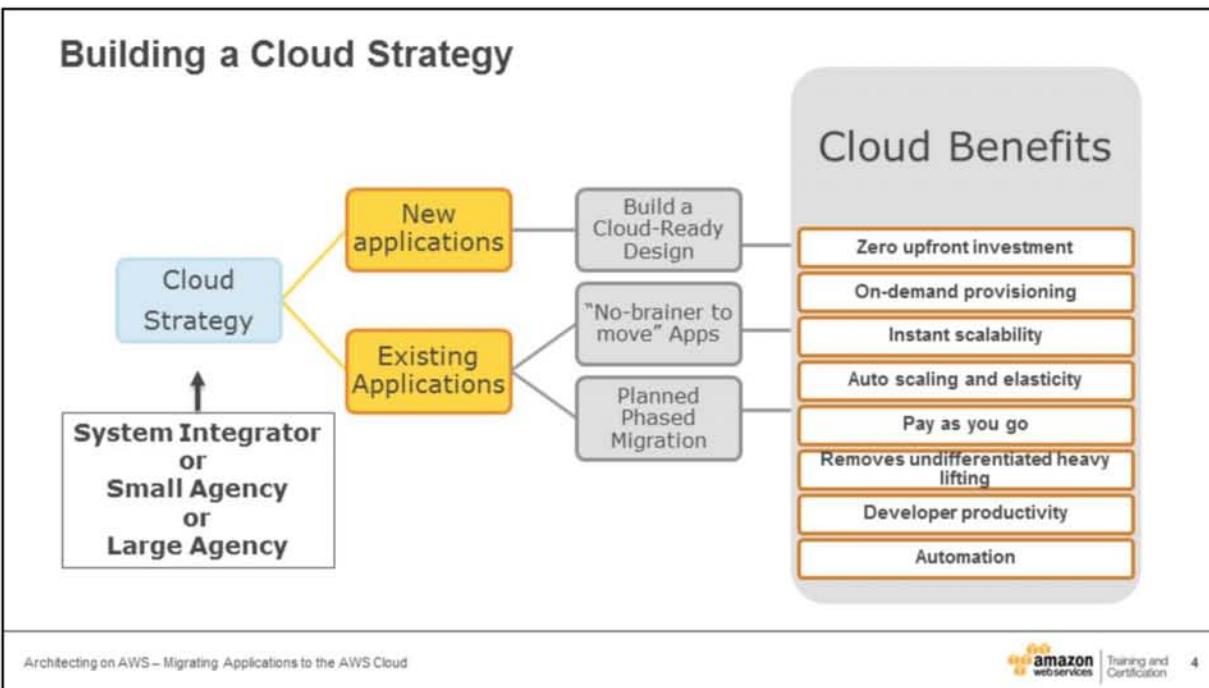
- 💡 Defining cloud strategies
- 💡 Planning migrations
- 💡 Deploying applications
- 💡 Optimizing applications

Notes:

We'll look at basic strategies to consider for new and existing applications, discuss what types of applications are easiest to migrate, and what the different phases of migration are.

In the first topic, *Defining cloud strategies*, we are going to discuss the following:

- Basic strategies for new applications
- Strategies for existing applications
- Types of applications that are easy to migrate
- Phases of migration



Notes:

There are several approaches to building a cloud strategy. We've seen customers from all size companies, from all industries get started with AWS in different ways. Building a cloud strategy really depends on the your organization's needs. We recommend that you have two strategies in mind: one for new applications, and one for existing applications.

For new applications, build and design new architectures with the cloud in mind. We have seen several customers, who were able to build cloud-ready applications from the ground up. These customers designed their applications using some or all of the cloud architecture patterns we've discussed—such as elasticity and loosely-coupled systems.

Existing applications can also benefit from the cloud. These applications require a different approach than new applications. For example,, we recommend defining a migration plan and transitioning application by application. This enables you to gain experience with the cloud at a pace that you define. As you define your migration plan, you'll find that some applications are ready to move right away. Other apps, however, might require a slower, more methodical plan.

This strategy has worked for several of our customers. Whether you are startup or an SMB or Large enterprise or an SI helping the customer, the strategy does not really need to change.

Let's look at how you might move existing applications to the cloud. These apps fall

into two categories:

- “No-brainer to move” apps. These are apps that can easily move into the cloud with a minimum of time and effort.
- Planned phased migration. These are apps that require more thought before moving them into the cloud.

“No-brainer to move” Apps

- 💡 Dev/Test applications
- 💡 Self-contained Web Applications
- 💡 Social Media Product Marketing Campaigns
- 💡 Customer Training Sites
- 💡 Video Portals (Transcoding and Hosting)
- 💡 Pre-sales Demo Portal
- 💡 Software Downloads
- 💡 Trial Applications

Notes:

What's a no-brainer app? It really depends on your organization. Here's a list of few types of apps that are often easy to move to AWS.

Topics

- 💡 Defining cloud strategies
- 💡 Planning migrations
- 💡 Deploying applications
- 💡 Optimizing applications

Notes:

In this topic, we are going to discuss the following:

- Two phases of migration planning
- Key questions to ask prior to migration
- Criteria for stack-ranking applications
- Two tools for cost management
- Licensing models and the cloud

Phased Approach to Plan Migrations

💡 Goal: Identify which application to move first

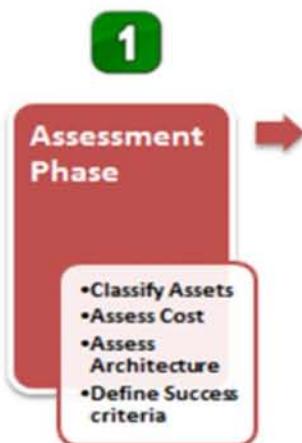


Most companies skip this phase!

Notes:

Planning an app migration consists of two phases. The first is the assessment phase—and this phase is often overlooked.

Planning Migrations: Assessment phase



Questions you need to ask:

- Which business applications should move to the cloud first?
- Does the cloud provide all of the infrastructure building blocks you require?
- Can you reuse your existing resource management and configuration tools?
- What are my legal, governance and compliance requirements?
- What are your criteria to measure success? How will you measure it ?

Notes:

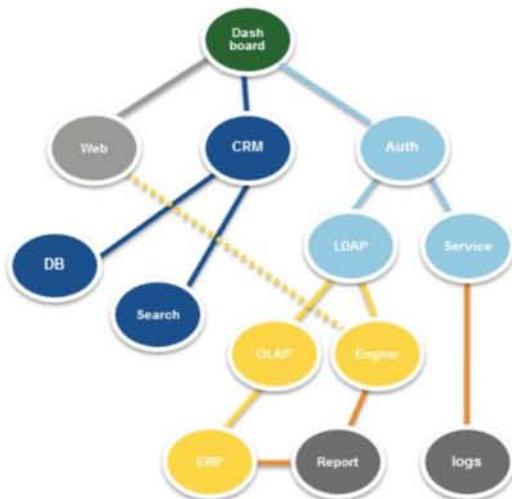
The assessment phase is when you look at your apps and figure out what it takes to move the app to the cloud.

Questions that you need to ask at this stage include:

- Which are business applications should move to the cloud?
- After the migration is in progress, which are the next that you could move?
- Does the cloud provide all the infrastructure services you require?
- Which tools you can reuse?

These questions are particularly useful at this time, but keep them in mind as you move into the other phases of migration.

Classifying your IT Assets



- 💡 List all your IT assets
- 💡 Identify upward and downward dependencies
- 💡 Start classifying your IT assets into different categories:
 - Applications with Classified, Sensitive, or Public data sets
 - Applications with low, medium and high compliance requirements
 - Applications that are internal-only, partner-only or customer-facing
 - Applications with low, medium and high coupling
 - Applications with strict, relaxed licensing

Notes:

The first step in the migration of existing applications comes to classifying your IT assets. List your IT assets and identify the upward and downward dependencies.

Within every organization there are variety of applications of different sizes, shapes, and characteristics. In some cases, moving an app to AWS might not make sense, but that doesn't mean that there aren't other apps to consider.

Stack rank your IT assets; select the low-hanging fruits first

- 💡 Search for under-utilized IT assets
- 💡 Applications that have immediate business need to scale
- 💡 Applications that are running out of capacity
- 💡 Easiest to move today
- 💡 That builds support within your organization and creates awareness and excitement

Rank	Asset Name	
1	Product Marketing Site	Today
2	Internal Batch Process	2 days
3	CRM System	5 days
4	Log Processing apps	1 week
5	ERP System	Phased Migration

Architecting on AWS – Migrating Applications to the AWS Cloud

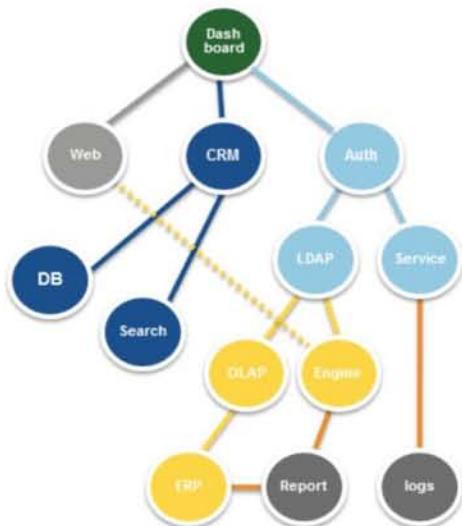
 Amazon
Training and
Certification 10

Notes:

After you identify your assets, stack rank them and prioritize the applications based on criteria you defined earlier.

It's not uncommon to find that some apps can move to the cloud and save you time and resources.

Pick the Low-hanging fruit



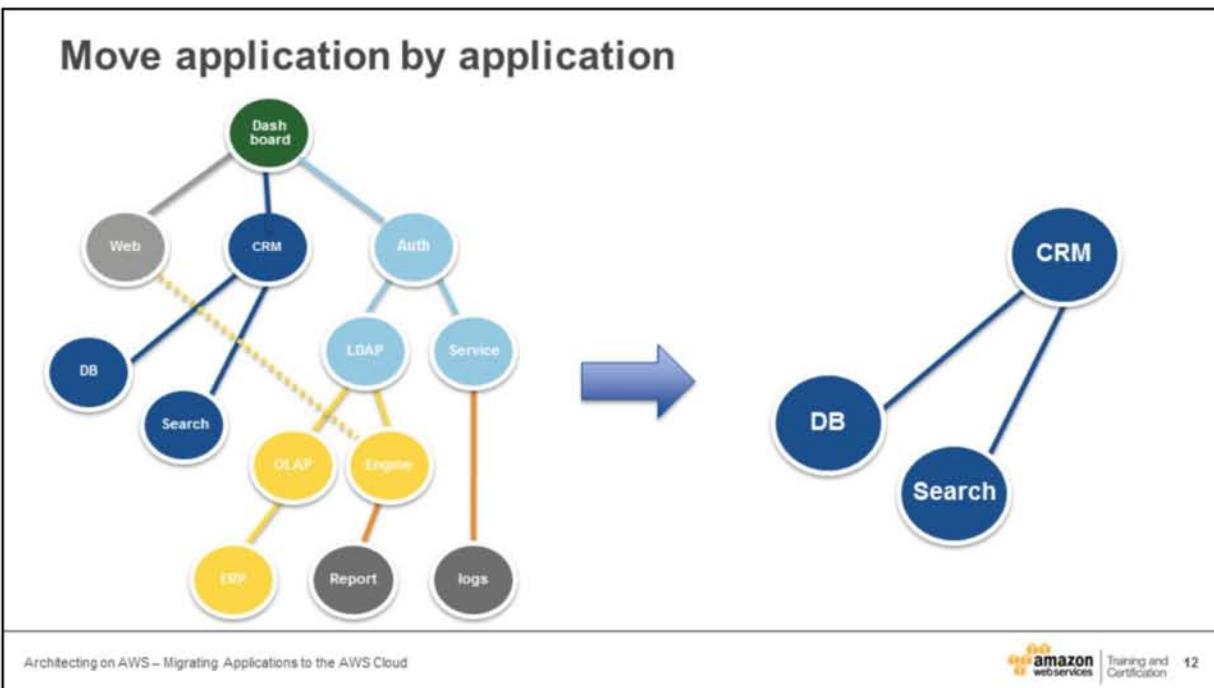
Examples:

- Web Applications
- Batch Processing systems
- Content Management Systems
- Digital Asset Management Systems
- Log Processing systems
- Collaborative Tools
- Big Data Analytics Platforms

Notes:

After listing them, he started to stack rank this IT assets based on the analysis. The applications that would take time to move to the cloud are NOT off the list they are just lower down the list. And the applications that are low hanging fruits are up the list. Pick the low hanging fruits first, gain some experience and then tackle the other applications.

For example web applications or content management systems etc. are all easy to move and can be forklifted to the cloud.



Notes:

After you identify which apps you want to move and in what order, you can start to move them into AWS. It's often best to do this one application at a time.

Manage Costs

Pricing Model	One-time Upfront			Monthly		
	AWS	Co-lo	On-Site	AWS	Co-lo	On-Site
Server Hardware	0	\$\$\$	\$\$	\$\$	0	0
Network Hardware	0	\$\$	\$\$	0	0	0
Hardware Maintenance	0	\$\$	\$\$	0	0	0
Software OS	0	\$\$	\$\$	\$	0	0
Power and Cooling and Data Center Efficiency	0	0	\$\$	0	0	\$
Data Center/co-lo Space	0	\$\$	\$\$	0	0	0
Personnel	0	\$\$	\$\$	\$	\$\$	\$\$\$
Storage and Redundancy	0	\$\$	\$\$	\$	0	0
Bandwidth	\$	\$\$	\$	\$\$	\$	\$
Resource Management Software	0	0	0	\$\$	\$	0
Total						

Architecting on AWS – Migrating Applications to the AWS Cloud



Notes:

Cost management is critical—whether you're using on-premises resources or AWS services.

One common challenge is finding the TCO of an individual app. It's also important to factor in "hidden" costs such as power, cooling, and so on.

When you use AWS, all these costs are already included.

Also, don't forget to consider using reserved instances when you calculate your long-term TCO. As we discussed earlier, these instance types can provide significant savings over on-demand instances.

Cost to run in AWS?

The screenshot shows the 'Amazon web services SIMPLE MONTHLY CALCULATOR'. It features a sidebar with links to various AWS services: Amazon EC2 (highlighted in orange), Amazon S3, Amazon SQS, Amazon SES, Amazon SNS, and Amazon Route 53. The main area has tabs for 'Services' and 'Estimate of your Monthly Bill'. A dropdown menu 'Choose region:' is set to 'US-East (Northern Virginia) & US-Standard'. Below it, a section titled 'Compute: Amazon EC2 On-Demand Instances:' shows a table with one row. The table columns are 'Instances', 'Description', 'Operating System', and 'Instance Type'. The first column has a red minus sign icon. The second column contains '0'. The third column is 'Linux/OpenSolaris'. The fourth column is 'Micro'. A note above the table states: '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 simple, efficient, and cost-effective for developers and IT professionals.' At the bottom, there is a footer with the text 'Architecting on AWS – Migrating Applications to the AWS Cloud' and the Amazon logo with 'Training and Certification 14'.

Notes:

Though the number and types of products offered by Amazon Web Services has increased dramatically in the past several years, Amazon's philosophy on pricing has not changed. You pay for what you use every month, and you can start or stop using a product anytime. No contracts are required.

Remember to use our economics center to help estimate the cost of running your app in AWS.

Flexible Licensing Options Available Today

- Bring Your Own License (BYOL)
- Use a utility style pricing model with a support package
- Use ISV Cloud Service



Notes:

A few things to keep in mind:

- AWS does not require developers to use any particular programming model, language, or operating system.
- AWS does not force developers to use the entire suite of services - they can use any of our infrastructure services individually or in any combination.
- AWS does not limit developers to a pre-set amount of storage, bandwidth, or computing resources they can consume - they can use as much or as little as they wish, and only pay for what they use.

Define your Success Criteria

Cloud is not just about saving money

- Developer Productivity
- Business Agility
- Reduced Time to Market
- Data center efficiency
- Redundancy
- Chargeback and Billing
- Eliminates “Heavy lifting”
- Foundation of 21st century architectures
- Reduced waste/recycle
- Hardware upgrades
- Less number of 24/7 Personnel

Notes:

Remember that cloud is not just about saving money. It offers you opportunities to increase developer productivity, improve business agility, and reduce time to market.

You don't have to worry about hardware upgrades. Also, if your company decides to move away from commercial development tools to open source ruby/rails (or vice versa), all you have to do is turn off your machines, dispose your instances and start new.

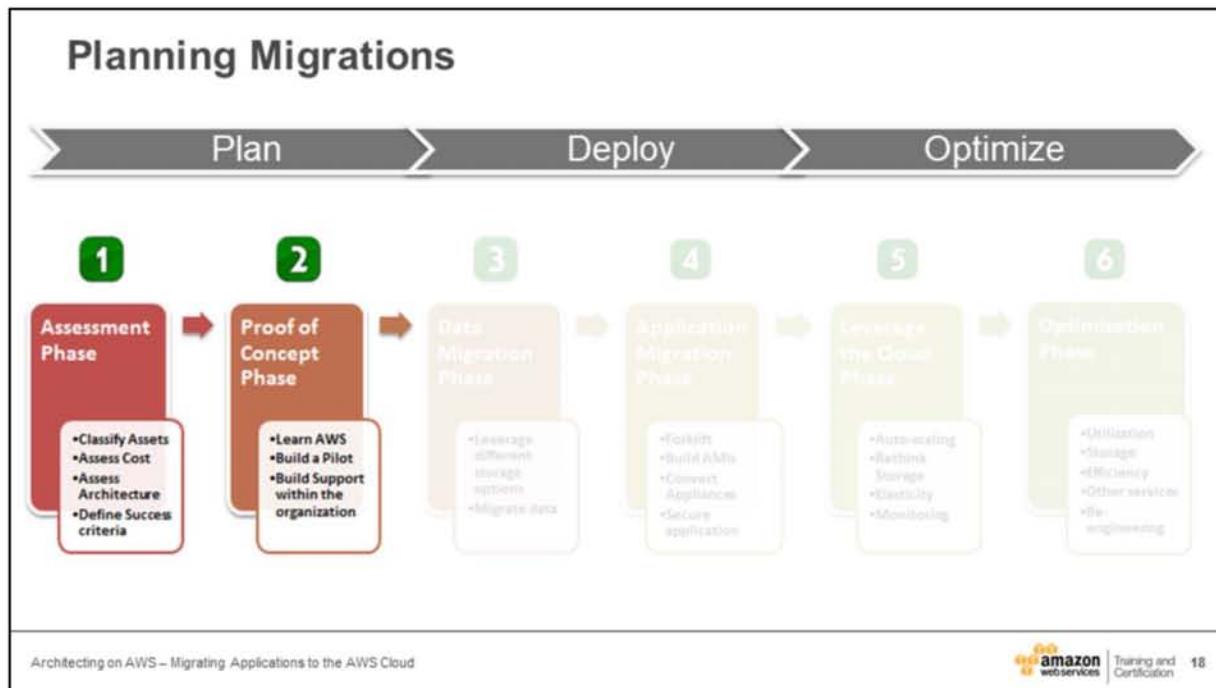
Define your Success Criteria and measure it

Success Criteria Examples	Old	New
Cost (CapEx)	\$1M	\$300K
Cost (OpEx)	\$20K/Year	\$10K/Year
Hardware procurement efficiency	10 machines in 7 months	100 machines in 5 minutes
Time to market	9 months	1 month
Reliability	unknown	Redundant
Security	5 products launched in 1 year	15 products launched
Flexibility and Productivity	Fixed Stack	Any Stack
New opportunities	10 projects backlog	0 backlog, 5 new projects identified

Notes:

Define your success criteria and measuring it at every stage to know your progress is very important. Here are some examples:

- **NASA JPL** had 10 projects in backlog pre-cloud and now they had 0 backlog and 5 new projects identified. They had a net result of 25 new projects initiated in 3 months.
- **Netflix** was able to go to time to market within 1 month when any new device comes in and they want to viewing support.



Notes:

After you've done the work of assessing your apps and defining what your goals are, you're ready to move to the proof of concept phase.

Planning Migrations: Proof of Concept



Questions you need to ask:

- Will I learn different aspects of the AWS cloud by building this proof of concept ?
- How much effort is required to port a small dataset and small app ?
- Will this proof of concept build support and create awareness within the organization ?
- What is the best way to capture all my lessons learned? A whitepaper?
- Which applications can I move immediately after this proof of concept?

Notes:

During this phase, you'll learn more about AWS, put a pilot app in place, and help others in your organization see the value of moving apps to AWS.

Invest in Proof of Concept Early

Proof of concept
will answer tons
of questions
quickly

- Get your feet wet with Amazon Web Services
 - Learning AWS
 - Build reference architecture
 - Be aware of the security features
- Build a Prototype/Pilot
 - Build support in your organization
 - Validate the technology
 - Test legacy software in the cloud
 - Perform benchmarks and set expectations

Notes:

It's tempting to skip the assessment phase and jump straight to the proof of concept phase. We recommend that the proof of concept phase build onto the work you've done in the assessment phase. After you enter the assessment phase, you'll get a chance to get your feet wet with Amazon Web Services, get trained from Amazon (like what you've been doing over the past few days).

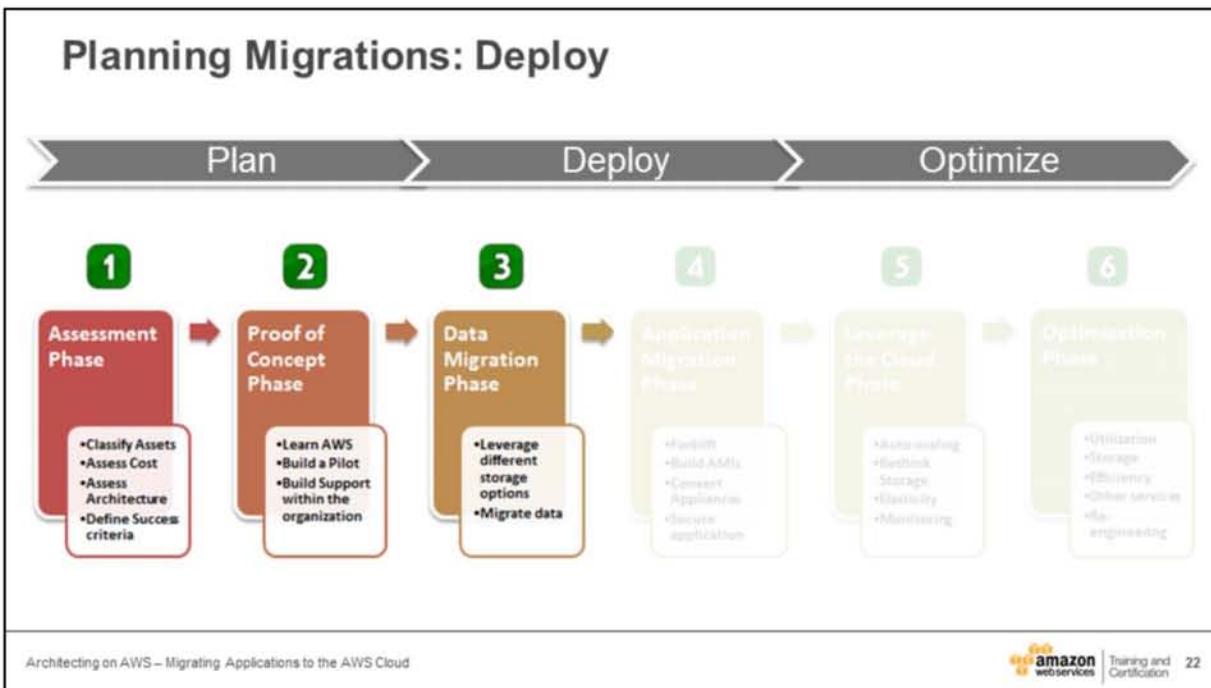
Topics

- 💡 Defining cloud strategies
- 💡 Planning migrations
- 💡 Deploying applications
- 💡 Optimizing applications

Notes:

In this topic, *Deploying applications*, we are going to discuss the following:

- Two phases of deployment
- How to get your data into AWS
- Handling your data
- Mirroring on-premises hardware and software

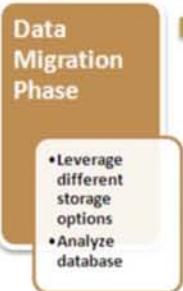


Notes:

First, let's talk about the data migration phase.

Planning Migrations: Deploying Applications

3

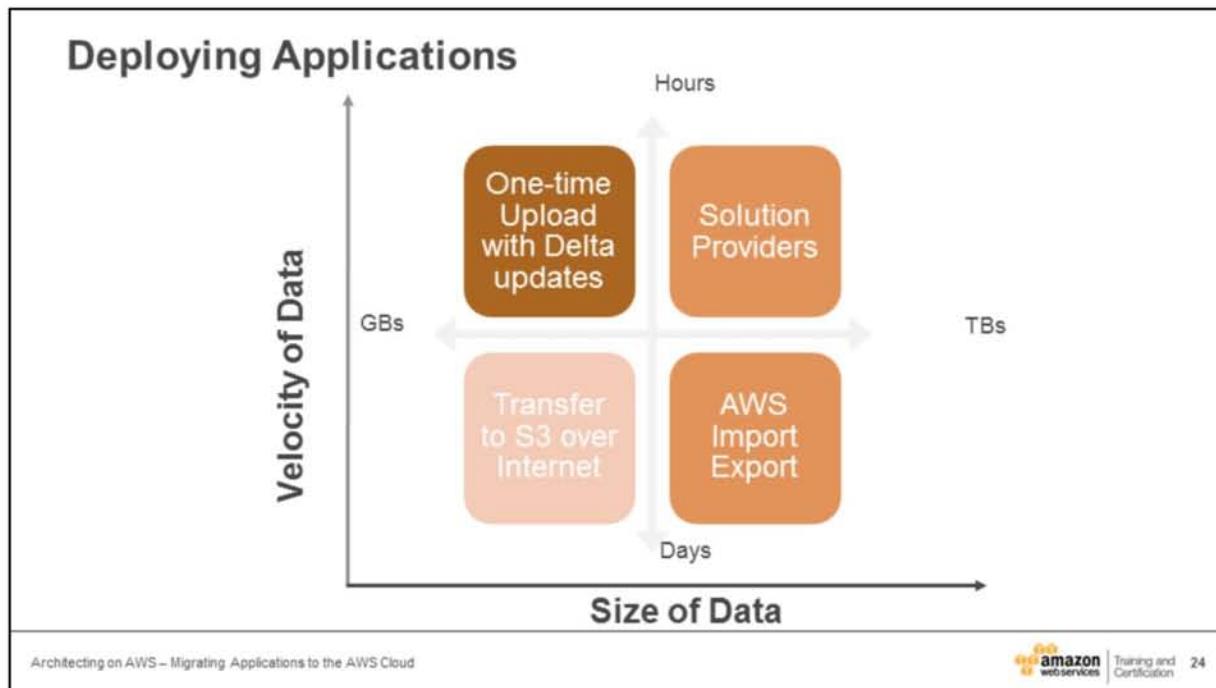


Includes:

- Learning about different database storage options available today
- Uploading/Moving your data in Batches
- Analyzing your database/datasets
- Build necessary tools and scripts to migrate data
- Security of your data (Encryption)

Notes:

In this phase, you identify what storage options make the most sense for the application.



Notes:

How you move your data into the cloud can take some thought. In this chart, you can see a few options depending on the velocity and size of the app's data.

Deploying Applications

💡 Cutting Over Your Master Data Store



Architecting on AWS – Migrating Applications to the AWS Cloud

 Amazon
Training and
Certification 25

Notes:

In some cases, you can do a one-time, bulk transfer, and then copy delta updates over as needed.

Deploying Applications

	Amazon S3 + CloudFront	Amazon EC2 Ephemeral Store	Amazon EBS	Amazon DynamoDB	Amazon RDS
Ideal for	Storing large write-once, read-many types of objects, Static Content Distribution	Storing local caches of state that can be easily re-built when needed	Off-instance persistent storage for any kind of data including File systems	Query-able light-weight attribute data	Storing and querying structured relational and referential data
Ideal examples	Media files, audio, video, images, Backups, archives, versioning	Config data, scratch files, TempDB	Clusters, boot data, Log or data of commercial RDBMS like Oracle, DB2	Querying, Indexing, Mapping, tagging, click-stream logs, metadata, Configuration, catalogs	Web apps, Complex transactional systems, inventory management and order fulfillment systems
Not recommended for	Querying, Searching	Storing database logs or backups, customer data	Static data, Web-facing content, key-value data	Complex joins or transactions, BLOBs	Clusters
Not recommended examples	Database, File Systems	Shared drives, Sensitive data	Content Distribution	OLTP, DW cube rollups	Relational, Typed data Clustered DB, Simple lookups

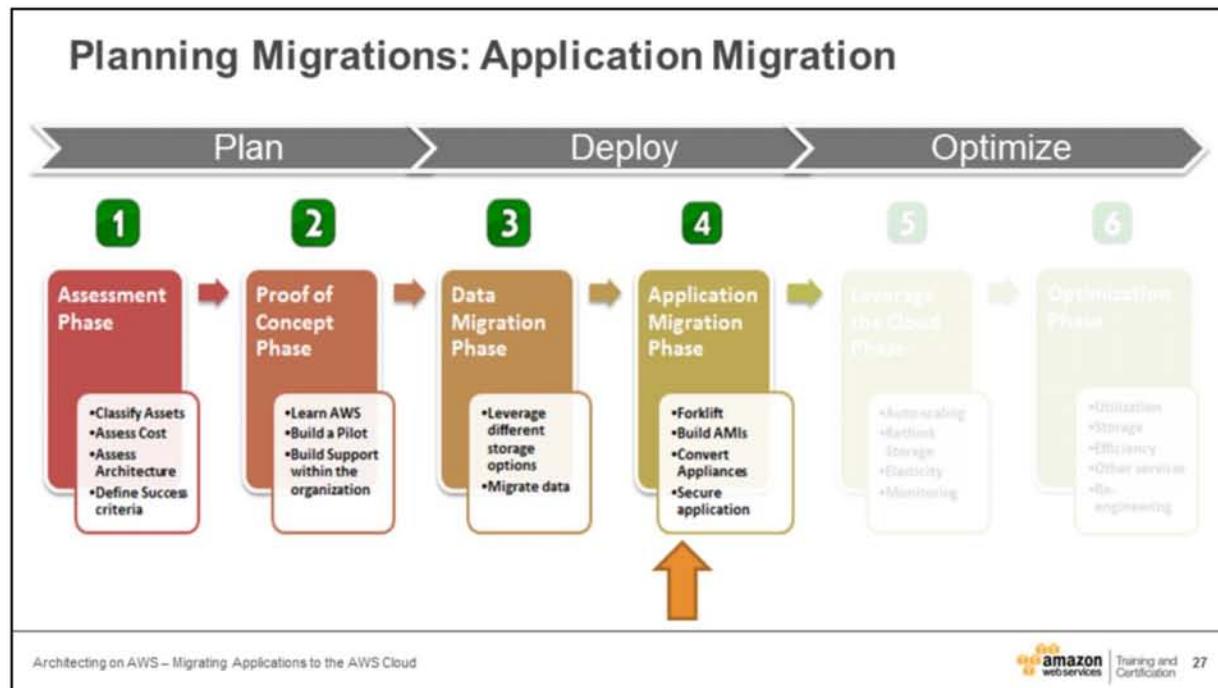
Architecting on AWS – Migrating Applications to the AWS Cloud



26

Notes:

Here's a chart that can help you identify which AWS storage option might make the most sense for your application.

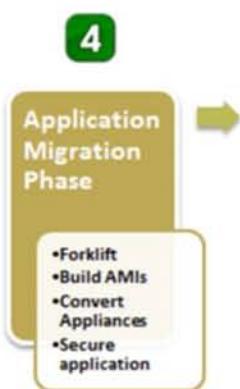


Notes:

The next phase is to migration your application. There are two approaches to consider:

- The forklift approach. This approach is often faster to implement. The disadvantage is that it doesn't take full advantage of cloud-based architecture (such as loosely-coupled systems).
- The redesign approach. This approach often results in greater long-term benefits to your organization. The disadvantage: it takes longer to implement.

Planning Migrations: Application Migration

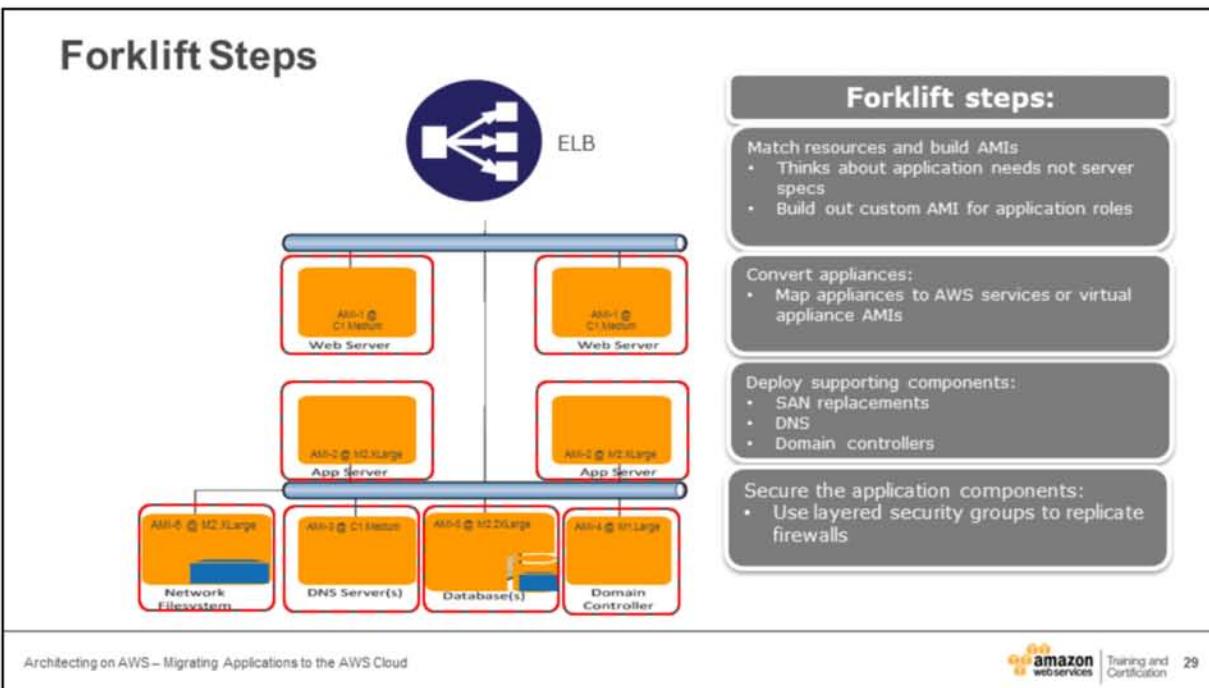


Includes - Forklift:

- Match your HW resources to the cloud
- Build AMIs
- Convert to virtual appliances
- Deploy supporting components (SAN, NAS, Domain controllers...)
- Secure your application
- Reuse existing management and monitoring tools or use cloud tools

Notes:

As this slide shows, the forklift approach consists of a few components, ranging from matching your resources to the cloud to security.

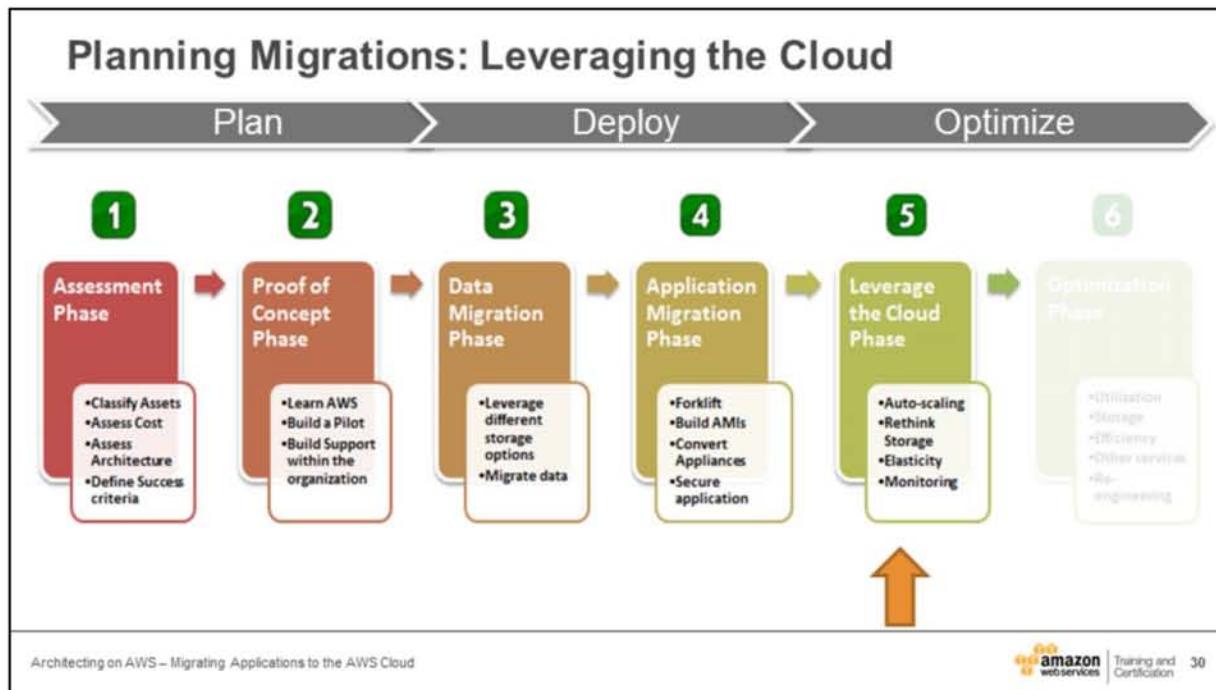


Notes:

During this phase, You have to build out AMIs that map to specific application roles in the existing architecture. You can see here that I now have an AMI for the Web server, App server and the Database. Next I convert my appliances. In this case, I swapped out a physical load balancer for an instance of ELB.

Now I build out AMIs for supporting components. In this case, I have added an AMI for the Storage Nodes (this could be a cluster) and the Domain Controller and the DNS server.

To secure the application, I used Security Groups to lock down traffic to only certain components between instances and that replaces the traditional firewall bottleneck.



Notes:

You're not completely done when you move your application to the cloud. Now is the time to start optimizing the app. First, you can start looking at how to leverage AWS services more effectively.

Planning Migrations: Leveraging the Cloud

5

Leverage the Cloud Phase

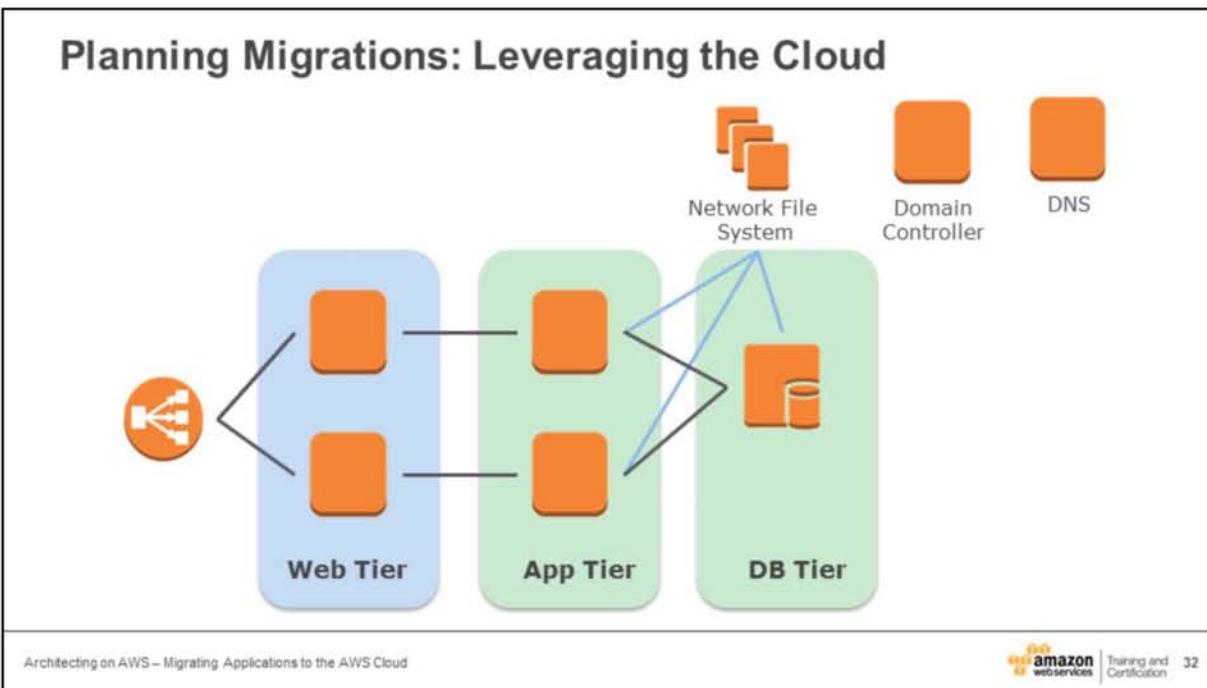
- Auto-scaling
- Rethink Storage
- Elasticity
- Monitoring



- 💡 Embrace and Implement Elasticity
- 💡 Bootstrap AMIs
- 💡 Automate processes
- 💡 Leverage Auto Scaling
- 💡 Leverage new storage options by AWS
- 💡 Harden Security (IAM)

Notes:

Leveraging the cloud takes more time than the forklift approach we discussed earlier. But it can result in greater scalability, less maintenance, and even reduced costs.

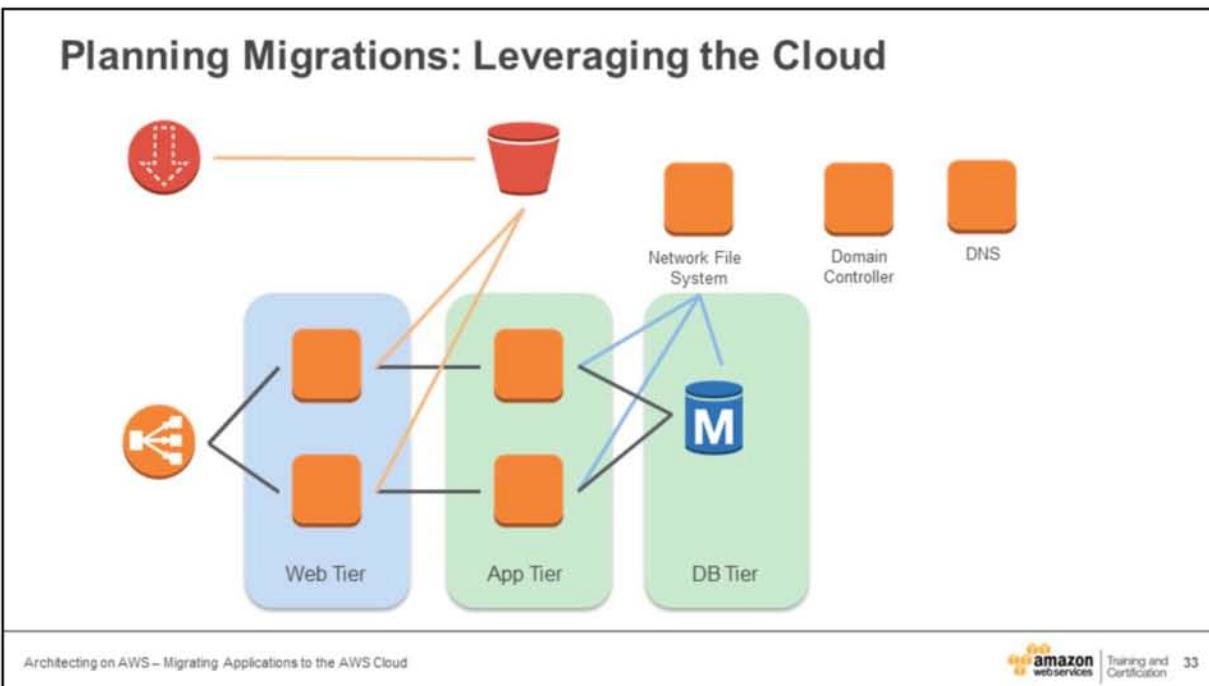


Notes:

Here is a simple network diagram of a web application running in AWS. Let's look at some ways we can better leverage AWS technologies to improve our application.

One option: rethink storage. By moving static content off for direct serving from S3 and switching to RDS Oracle you can reduce your reliance on and management of the Network Storage components. Additionally, you can use CloudFront to deliver my static content faster to customers.

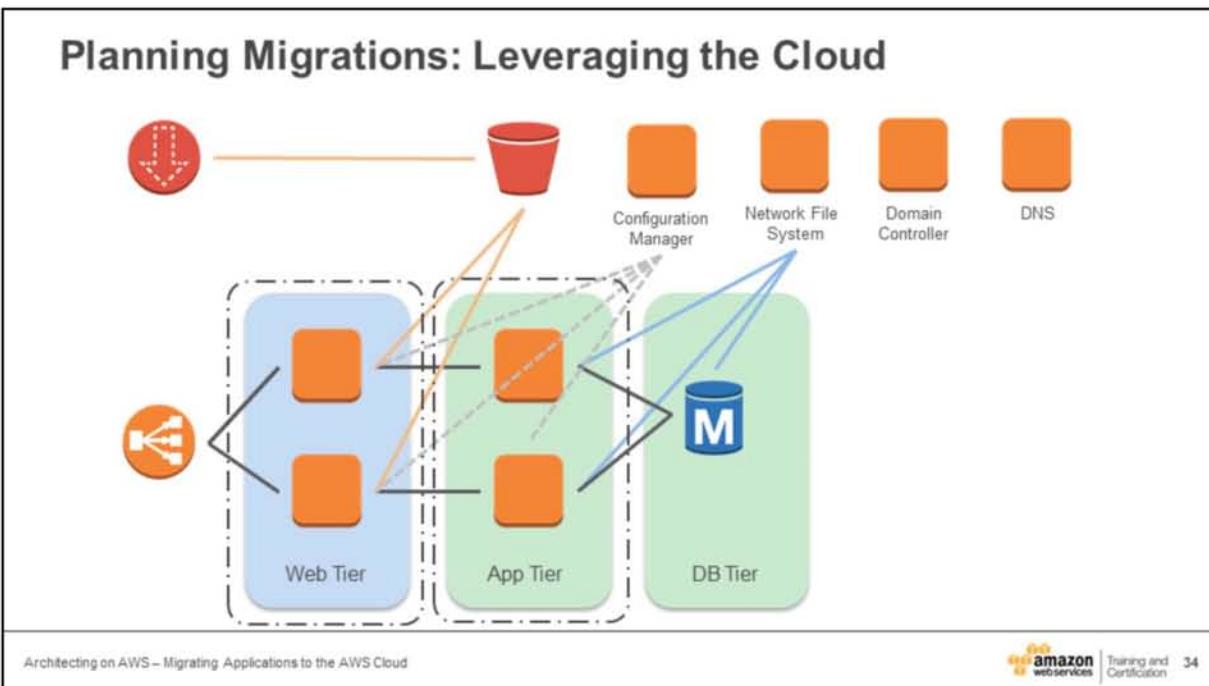
Next, implement elasticity. In this example, we have added a Configuration Management server to enable the dynamic configuration of each of instances. For scaling out and in, I have added Auto-scaling groups to the “stateless” layers. This combined with the configuration management server enables dynamic scale out and in without manual intervention.



Notes:

One option: rethink storage. By moving static content off for direct serving from S3 and switching to RDS Oracle you can reduce your reliance on and management of the Network Storage components. Additionally, you can use CloudFront to deliver static content faster to customers.

Next, implement elasticity. In this example we have added a Configuration Management server to enable the dynamic configuration of each of instances. For scaling out and in, we have added Auto-scaling groups to the “stateless” layers. This combined with the configuration management server enables dynamic scale out and in without manual intervention.



Notes:

Next, implement elasticity. In this example we have added a Configuration Management server to enable the dynamic configuration of each of instances. For scaling out and in, we have added Auto-scaling groups (indicated by the new boxes around our web and app tiers) to the “stateless” layers. This combined with the configuration management server enables dynamic scale out and in without manual intervention.

Accelerate the cloud adoption within your organization

Be the Cloud Champion within your company or team

- 💡 Be a Cloud Advocate
- 💡 Starting a weekly sync meeting
- 💡 Share Lessons Learned (Brownbags)
- 💡 Document Best Practices
- 💡 Reuse tools, scripts, How-Tos
- 💡 Start Cloud Computing practice or Cloud Computing Center Of Excellence
- 💡 Educate and Evangelize

Notes:

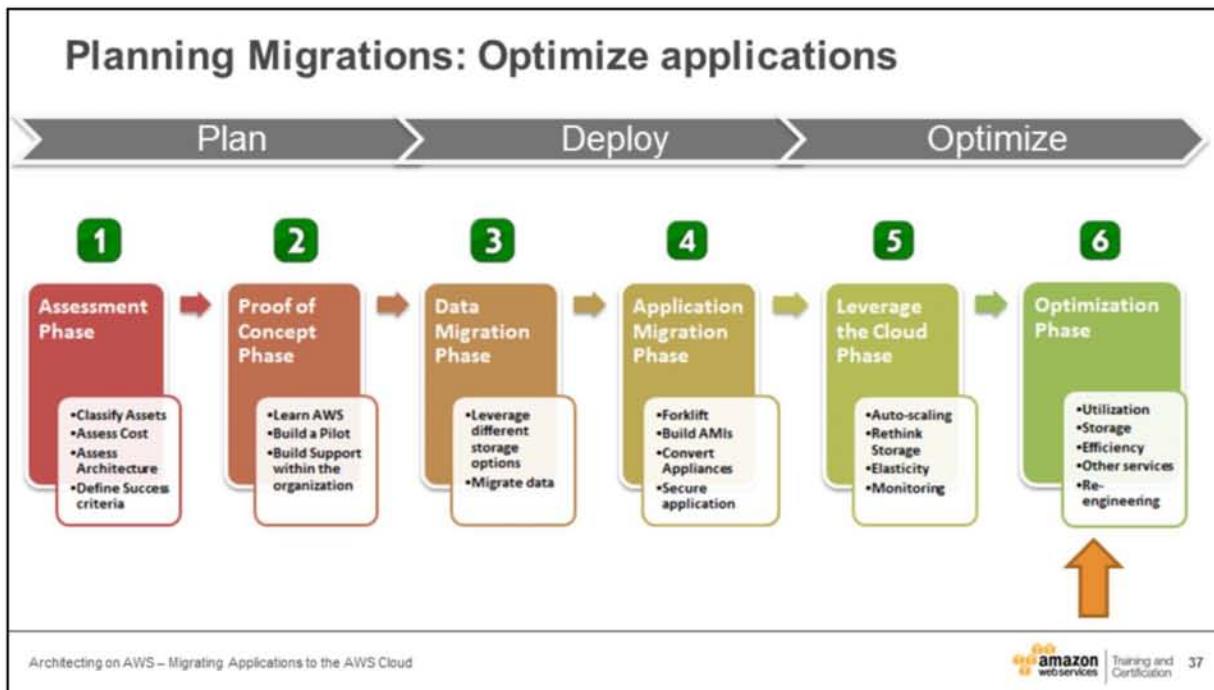
By the way, as you migrate your app to AWS, be sure to share what you've learned with others throughout your organization.

Topics

- 💡 Defining cloud strategies
- 💡 Planning migrations
- 💡 Deploying applications
- 💡 Optimizing applications

Notes:

The last phase of app migration is to optimize your app.



Notes:

Let's take a look at what it means to optimize an application.

Planning Migrations: Optimize applications



Improve Efficiency:

- Re-rethink Storage
- Parallel processing
- Optimize for cost (Use Spot)
- Optimize for availability
- Leverage scalable on-demand services like SNS, SQS

Notes:

In this phase, you can start thinking of things like parallel processing, spot instances, and so on.

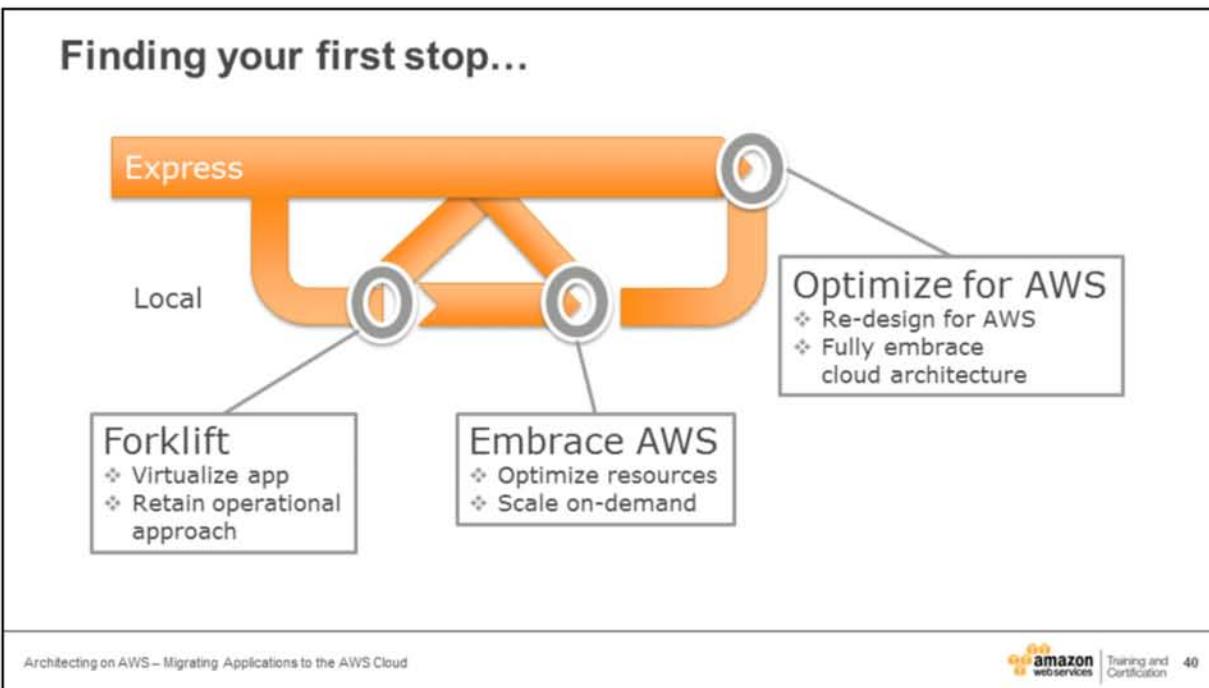
Optimize applications

- 💡 Re-re-think storage
 - Break up datasets across storage solutions based on best fit and scalability
- 💡 Parallelize processing
 - Spread loads across multiple resources
 - Decouple components for parallel processing
- 💡 Use spot instances where possible to reduce costs
- 💡 Embrace scalable, on-demand services
 - Scale out systems with minimal effort
 - Route 53
 - SES, SQS, SNS

Notes:

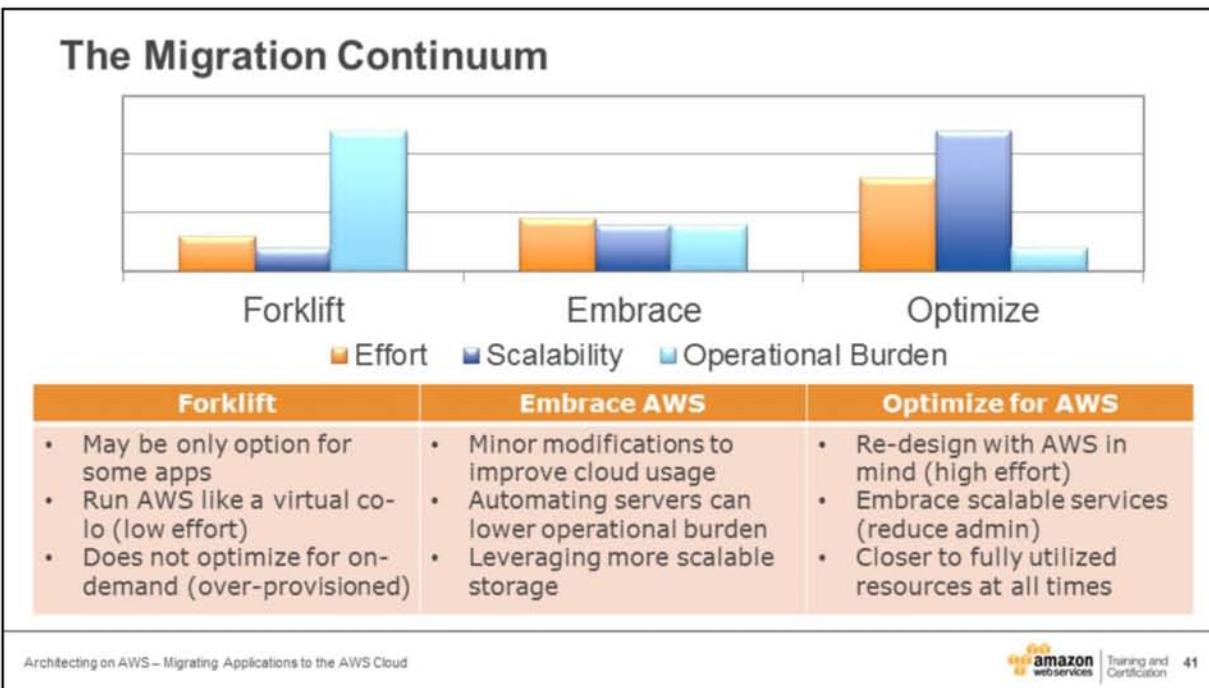
Ways you can optimize your applications include:

- Re-rethinking storage. For example, moving single tables to NoSQL style services like DynamoDB.
- Parallel processing. Look for opportunities to spread loads across more services for finer-grained resource management.
- Use spot instances. Opportunities to use spot is always something to look for to reduce cost and reduce processing time.
- Use scalable services. There are numerous scalable services that can be used in place of standalone instances. SQS, for example, can help coordinate App servers. Another example is Route 53, which can replace a DNS server.



Notes:

Here's another way of looking at the migration path, and the options that are available to you.



Notes:

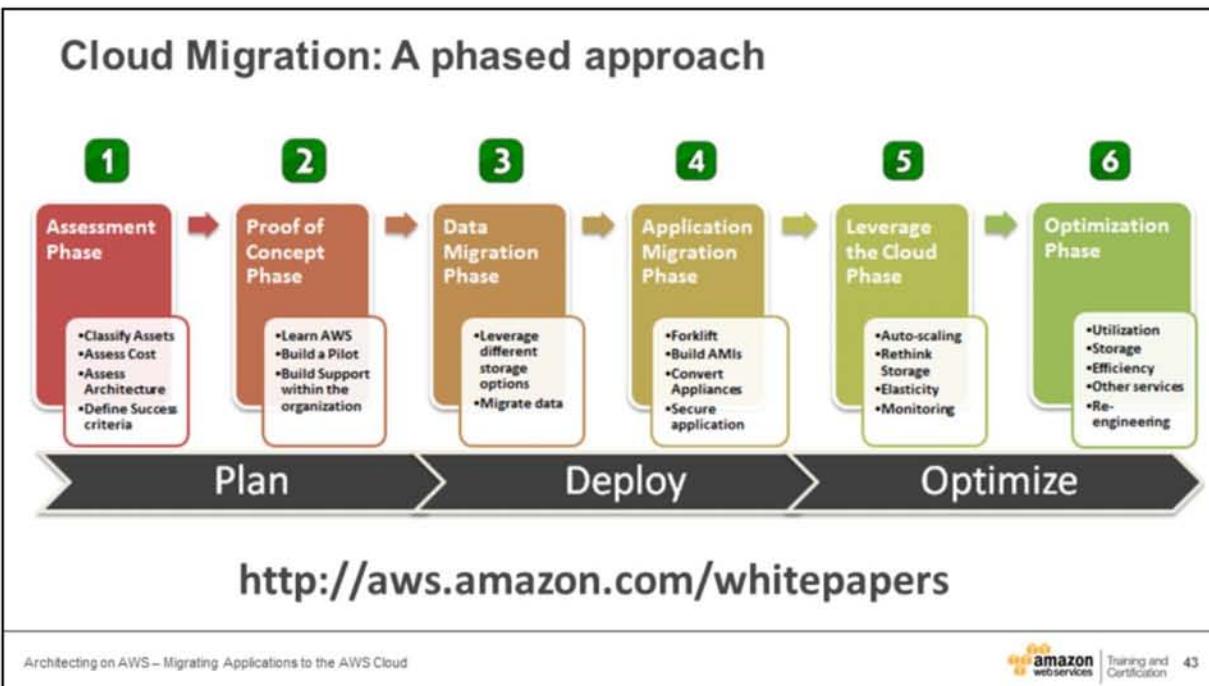
In this chart, you can see how the different migration phases affect the amount of effort, scalability, and operational burden.

Key Takeaways

- Classify and stack rank your apps and move the easy ones first, gain confidence and define your success criteria
- Dive into a Proof of Concept quickly as it will answer several questions quickly
- Leverage multiple storage options – one size does not fit all
- Migrate with confidence: Forklift – Leverage – Optimize
- Be the Cloud Champion within your agency, department or team

Notes:

Let's go over a few key thoughts on app migration.



Notes:

Following these phases can help you migrate your applications successfully. Review our whitepapers for more information and assistance.

Get Training!

Self-Paced Labs



Try products, gain new skills, and get hands-on practice working with AWS technologies

[aws.amazon.com/training/
self-paced-labs](http://aws.amazon.com/training/self-paced-labs)

Training



Skill up and gain confidence to design, develop, deploy and manage your applications on AWS

aws.amazon.com/training

Certification



Demonstrate your skills, knowledge, and expertise with the AWS platform

aws.amazon.com/certification

Notes:

We have several programs available to help you deepen your knowledge and proficiency with AWS. We encourage you to check out the following resources:

- Get hands-on experience testing products and gaining practical experience working with AWS technologies by taking an AWS Self-Paced Lab at run.qwiklab.com. Available anywhere, anytime, you have freedom to take self-paced labs on-demand and learn at your own pace. AWS self-paced labs were designed by AWS subject matter experts and provide an opportunity to use the AWS console in a variety of pre-designed scenarios and common use cases, giving you hands-on practice in a live AWS environment to help you gain confidence working with AWS. You have flexibility to choose from topics and products about which you want to learn more.
- Take an instructor-led AWS Training course. We have a variety of role-based courses to meet the requirements of your job role and business need, whether you're a Solutions Architect, Developer, SysOps Administrator, or just interested in learning AWS fundamentals.
- AWS Certification validates your skills, knowledge and expertise in working with AWS services. Earning certification enables you to gain visibility and credibility for your skills.

Copyright © 2013, 2014 Amazon Web Services, Inc. and its affiliates. All rights reserved.

This work may not be reproduced or redistributed, in whole or in part, without prior written permission from Amazon Web Services, Inc. Commercial copying, lending, or selling is prohibited.

Errors or corrections? Email us at aws-course-feedback@amazon.com.
Other questions? Email us at aws-training-info@amazon.com.

All trademarks are the property of their owners.



Module A: Reference Architecture

Architecting on AWS – Reference Architecture

 Amazon
Training and
Certification 1

DocStore

by aMedia Corp.

Architecting on AWS – Reference Architecture



Training and
Certification 2

DocStore

aMedia DocStore allows customers to upload and manage their documents

DocStore

aMedia DocStore allows customers to upload and manage their documents

- Upload and access content from anywhere – browser and mobile apps
- PDF, Office docs, images, videos – text content extracted, indexed, and searchable

DocStore

Two Account Types

Free

- 5GB storage
- Search on name, created/modified date, and tags
 - No full-text indexing
- All content may be securely downloaded by owner

DocStore

Two Account Types

Gift Premium

- 20 GB storage
- Full-text indexing and search for all documents (OCR on images)
- All content may be securely downloaded by owner

DocStore

Two Major Components

DocStore

#1: Web Application

- Allow users to upload, manage, and view all documents
- All content uploaded directly to S3

DocStore

#1: Web Application

- ☐ Both powered by public DocStore API
- ☐ Both must scale
- ☐ Both must be highly available

DocStore

#1: Web Application

- ❖ Multi-region deployment for global support
- ❖ User can log in to any region
 - Always redirected to “home” region (i.e., region where account was created)

DocStore

#1: Web Application

- Work in groups to design architecture that drives the web and mobile applications, including the DocStore API

DocStore Requirements

Account Types

- **Free** – 5GB cap, basic document indexing, download original files
- **Premium** – 20GB cap, full-text indexing/search + OCR

Content

- Stored in S3
- Efficient upload/download to/from S3
- Static assets in one S3 bucket; distributed globally

Web App/API

- User authentication
- Session state stored off-instance
- Powered by API tier
- Highly Available

Deployment

- Oregon (us-west-2) and Sydney (ap-southeast-2)
- Replicate minimal information (i.e., don't replicate uploaded content)

DocStore

Sample Architecture

Architecting on AWS – Reference Architecture



Application deployed in two regions: us-west-2 (Oregon)
and ap-southeast-2 (Sydney)

DocStore



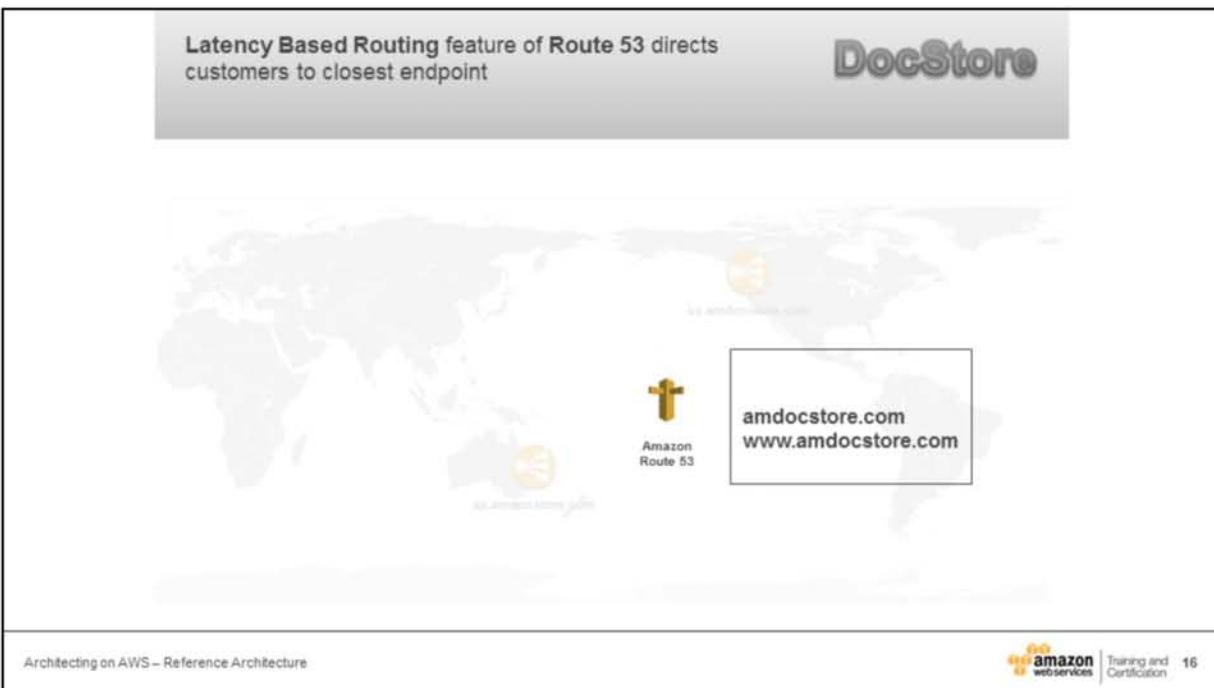
Each ELB has a region-specific CNAME in Route 53

DocStore



Architecting on AWS – Reference Architecture

amazon
web services | Training and
Certification 15









Let's focus on one region

DocStore



us.amdocstore.com

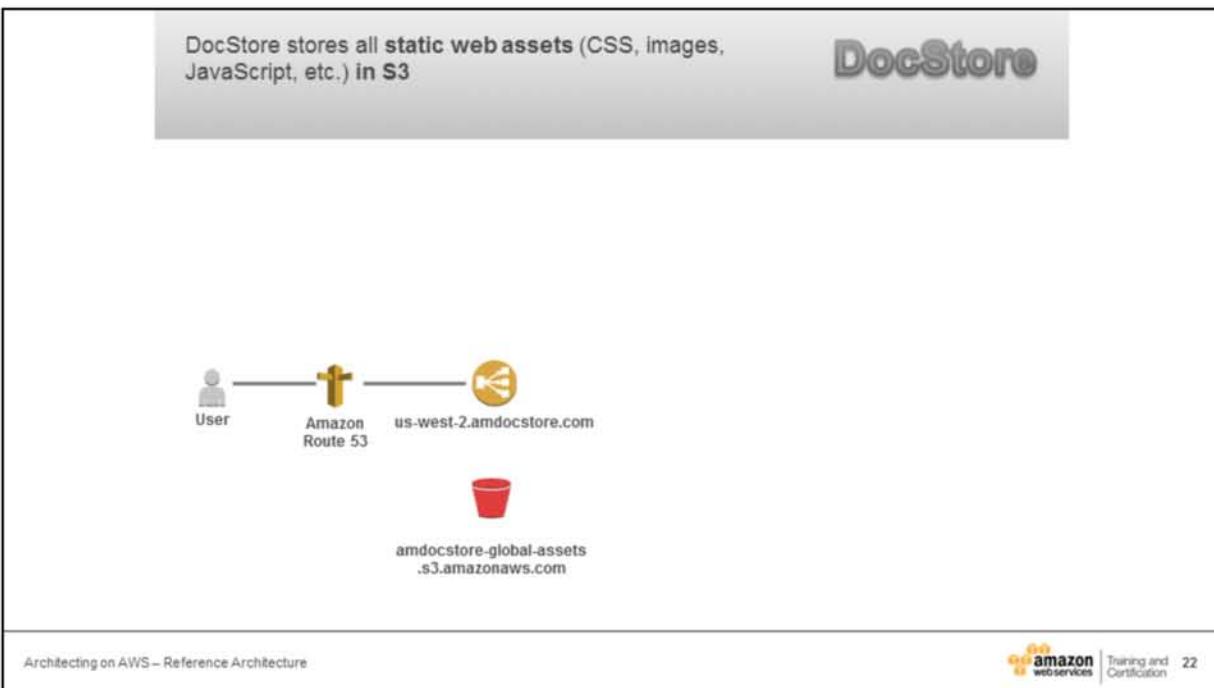
Architecting on AWS – Reference Architecture

 Amazon
Training and
Certification 20

Route 53 LBR directs a user to the nearest Elastic Load Balancer (in this case, the ELB in us-west-2)

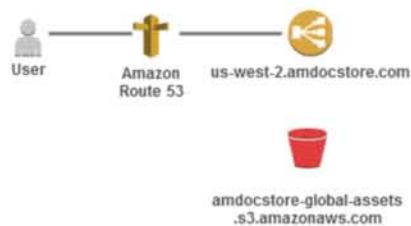
DocStore





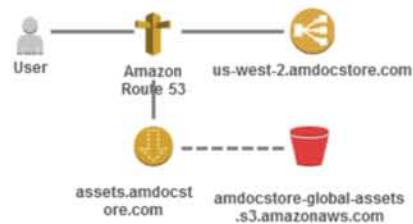
This bucket exists in the us-west-2 origin, but is the canonical source for static asset requests globally

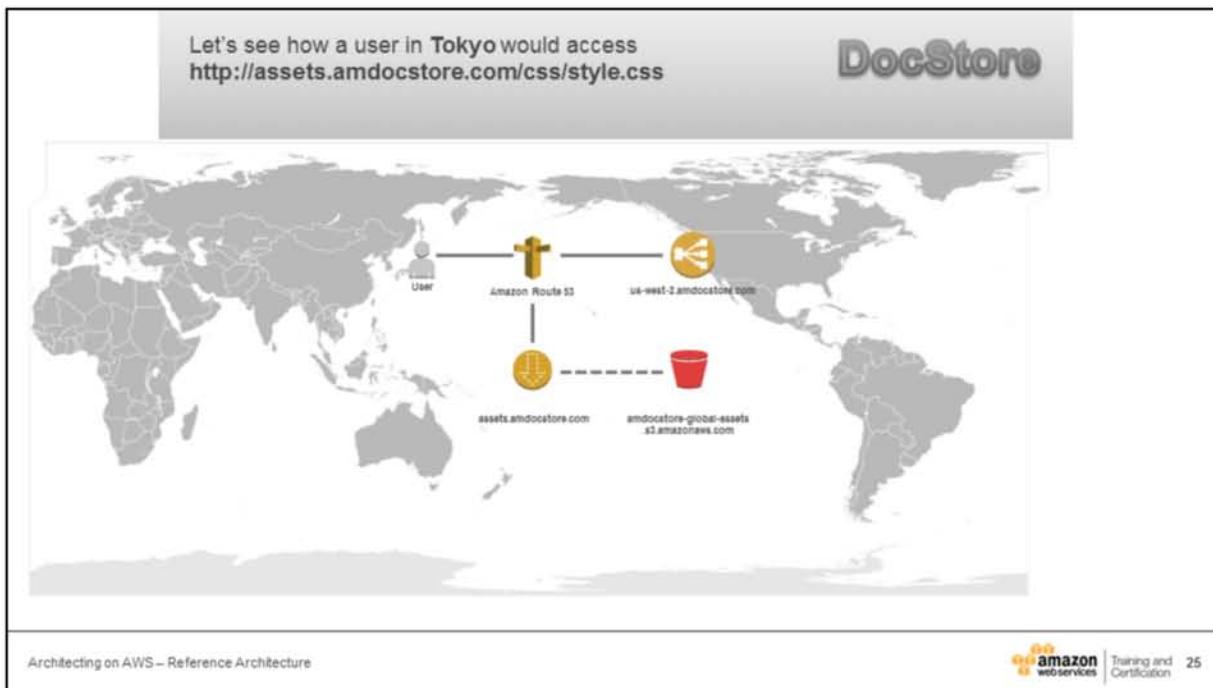
DocStore



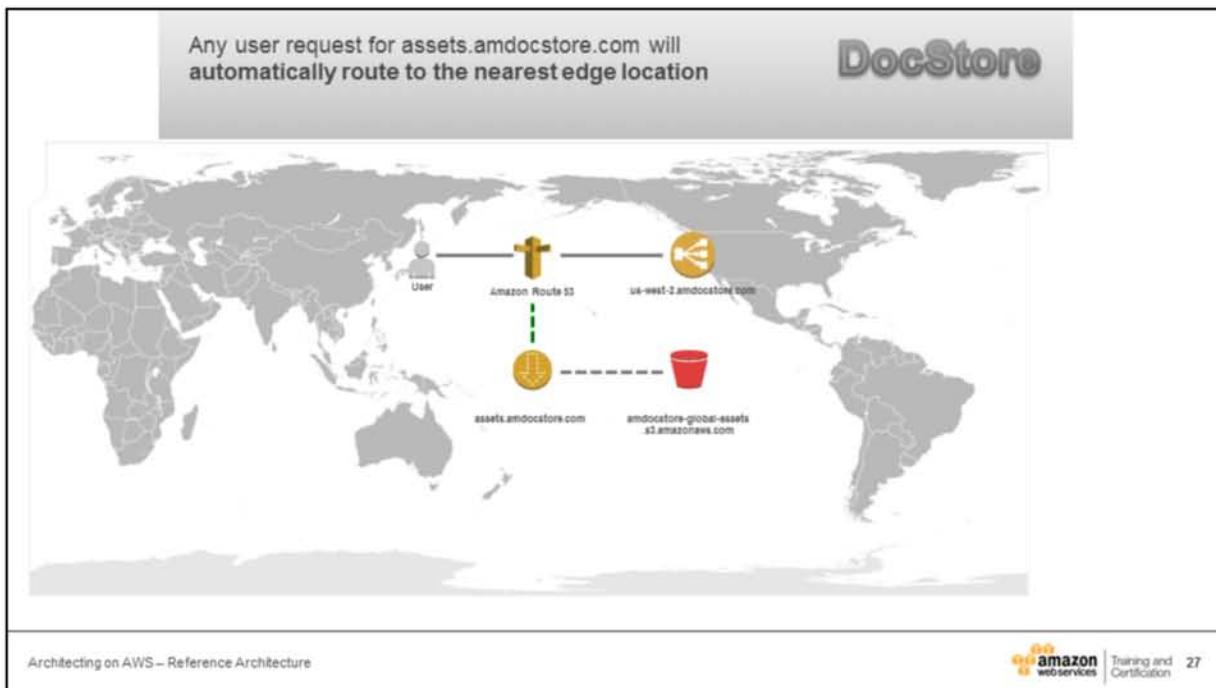
CloudFront distributes static web assets in S3 to end users with low latency using a global network of edge locations

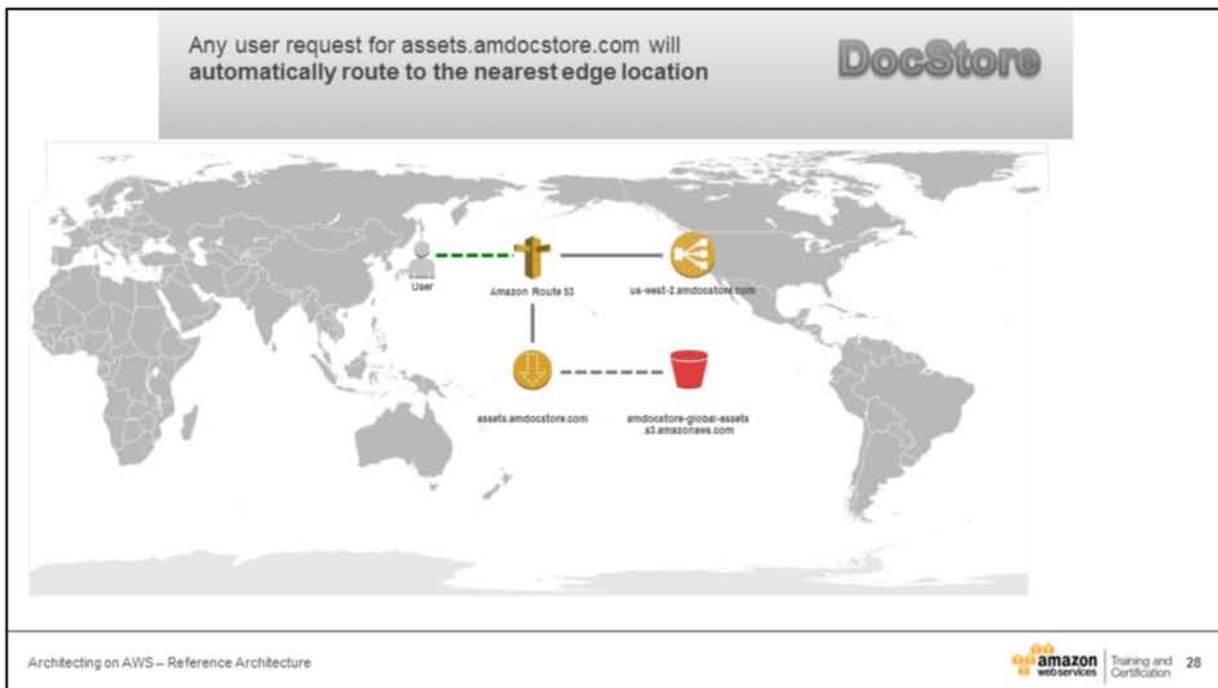
DocStore

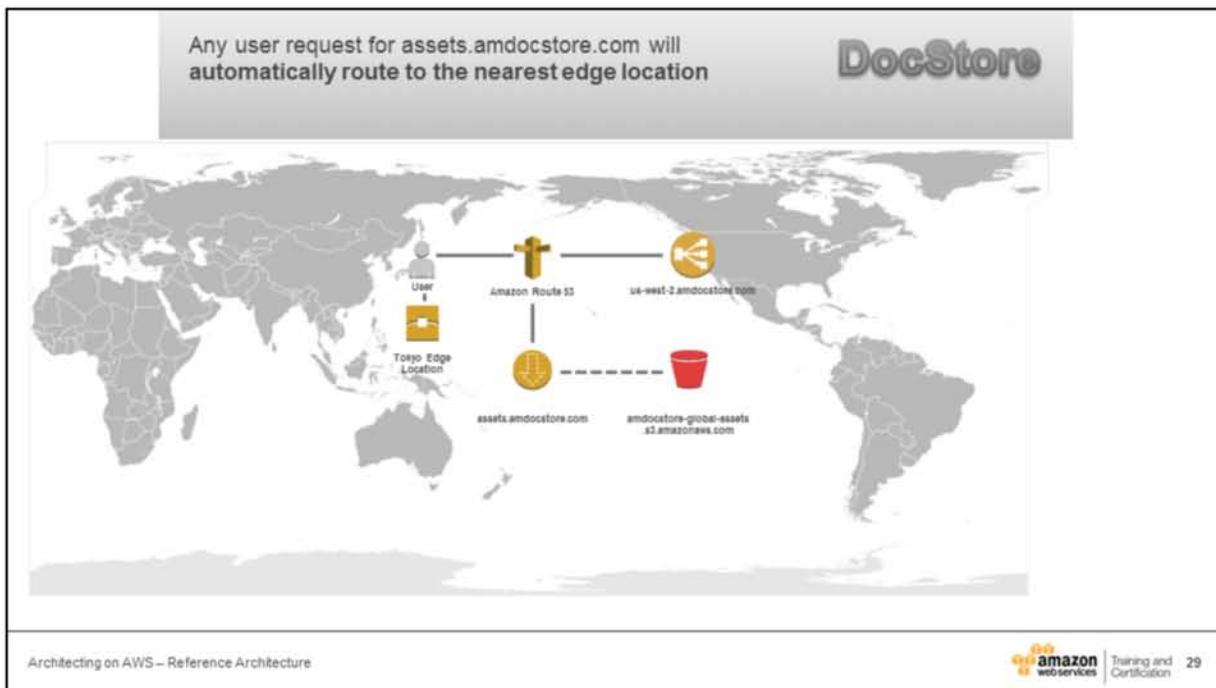


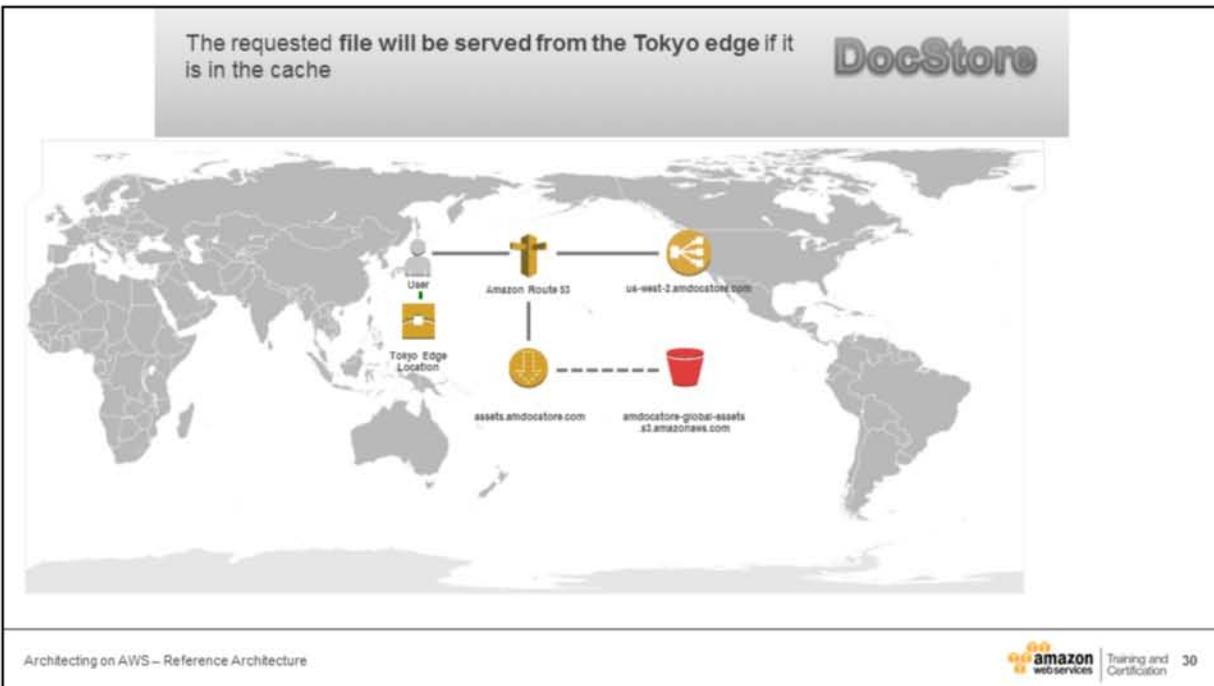


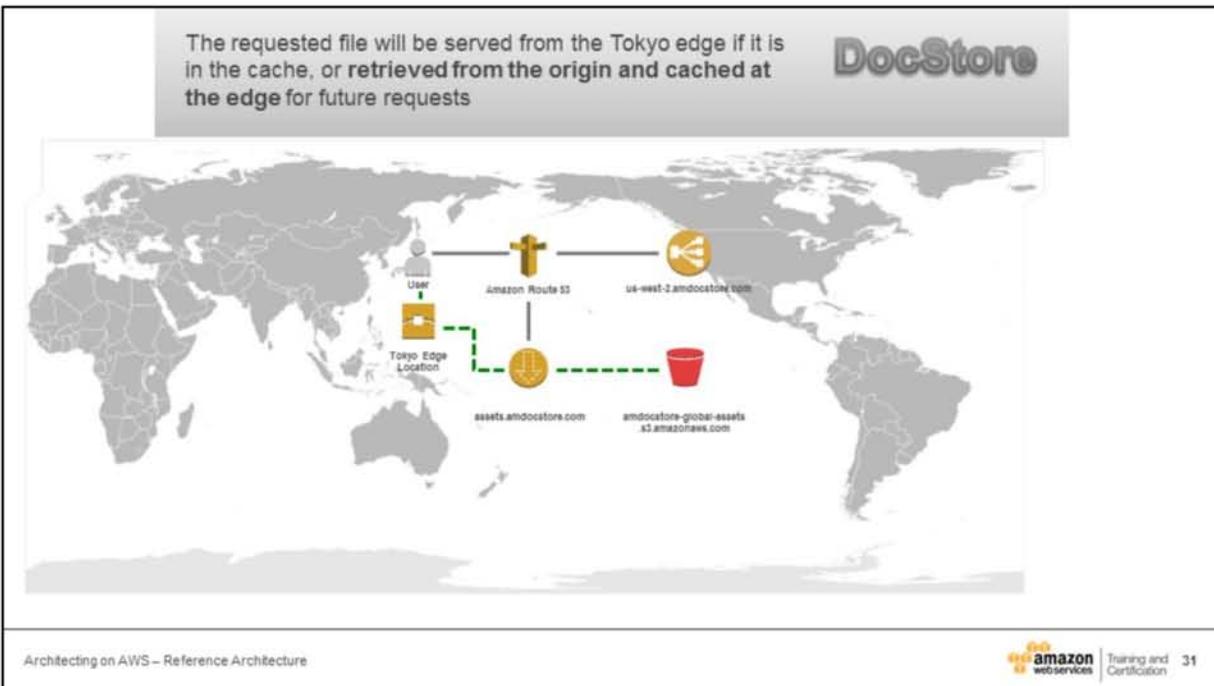


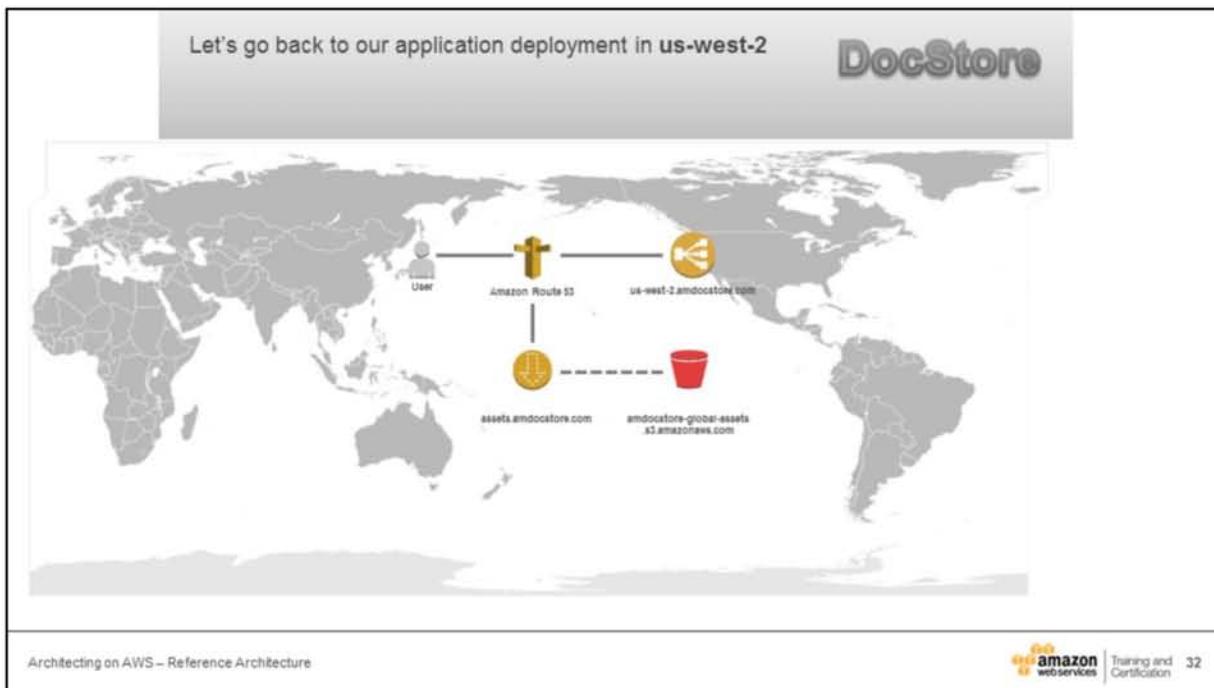












Let's go back to our application deployment in us-west-2

DocStore

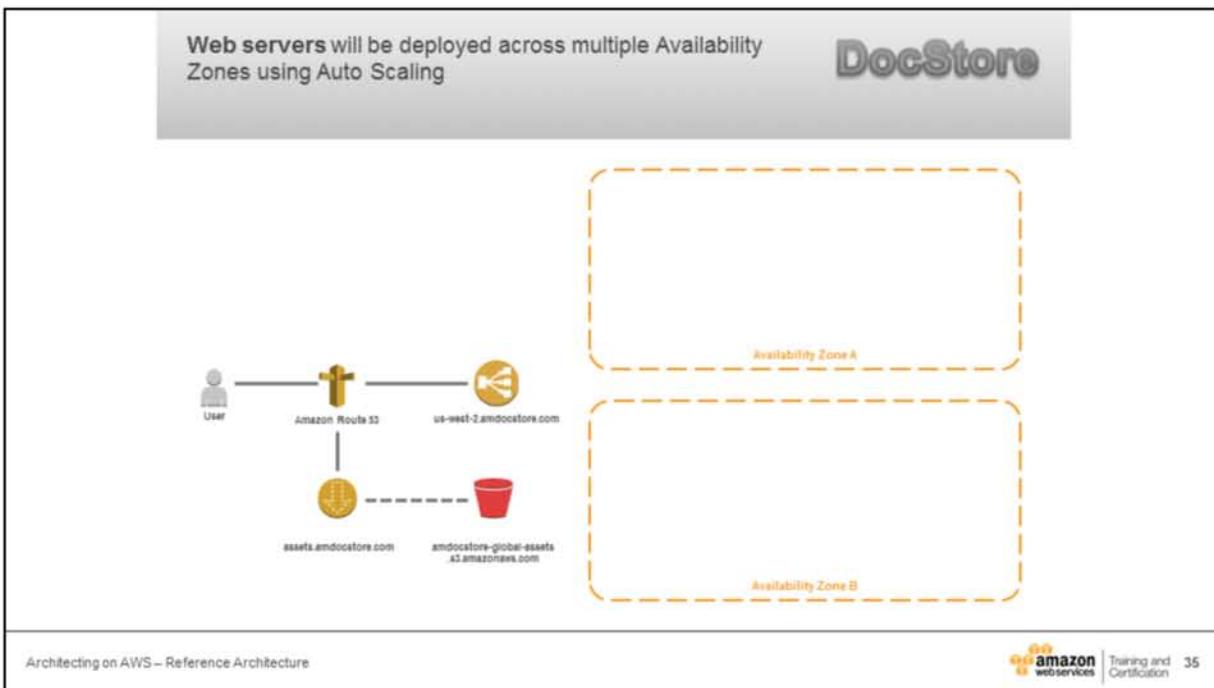


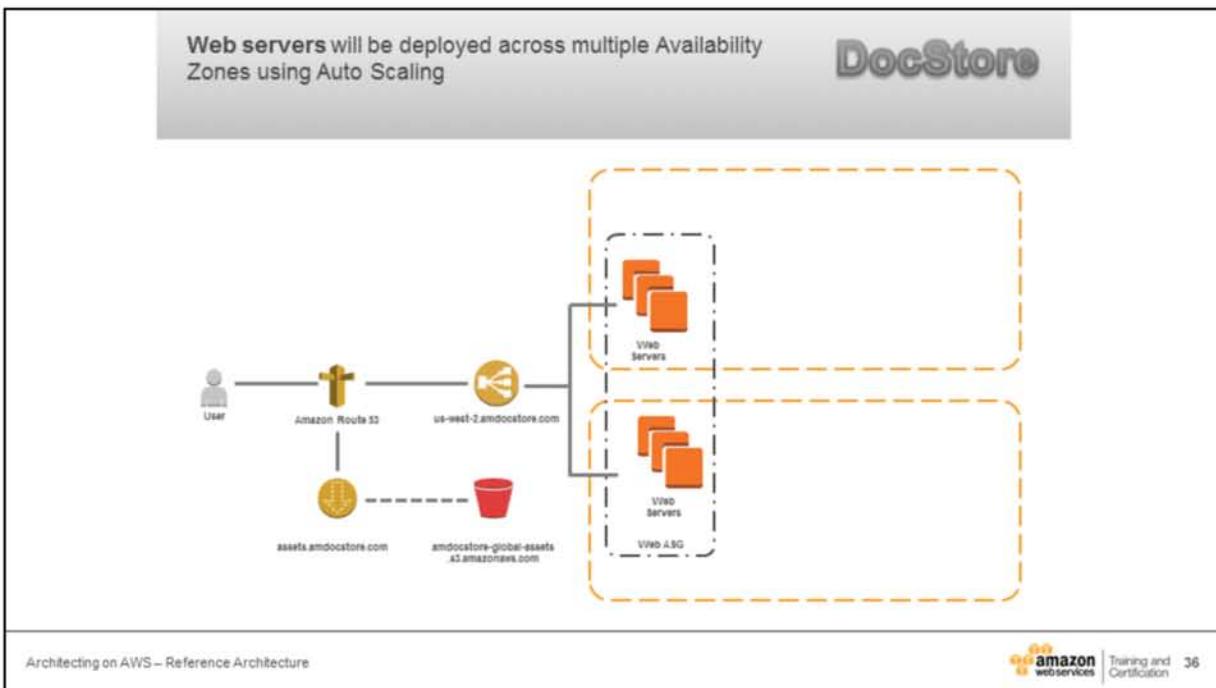
Architecting on AWS – Reference Architecture

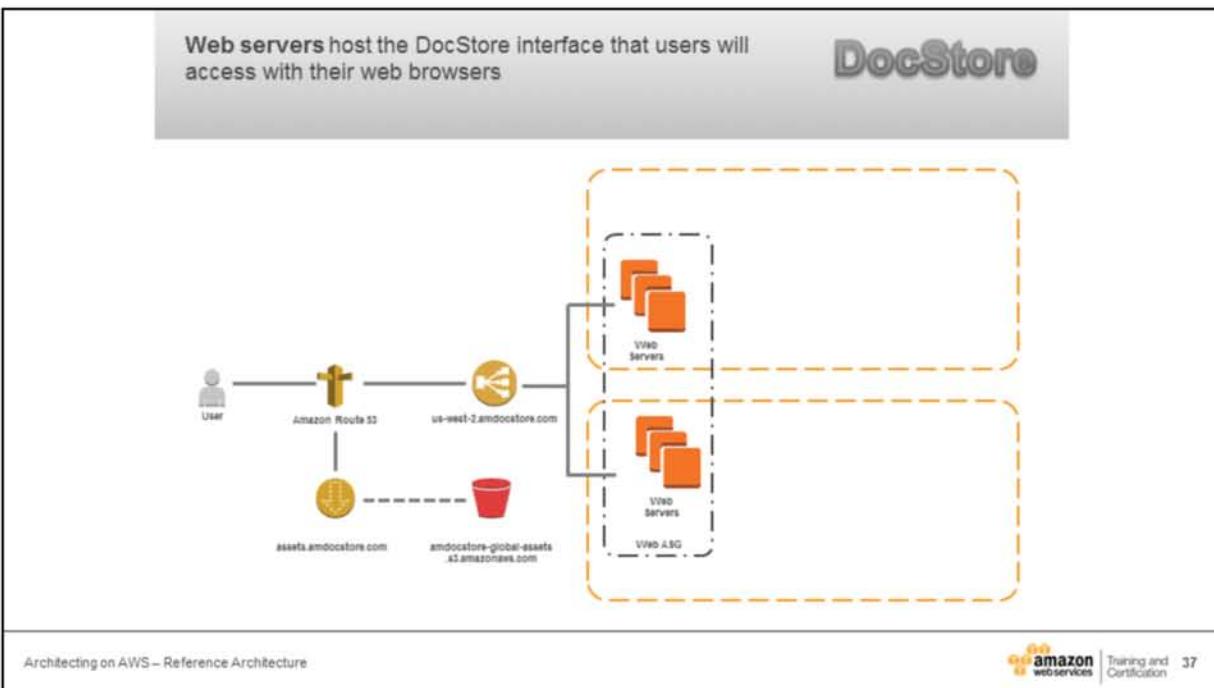
amazon web services | Training and Certification 33

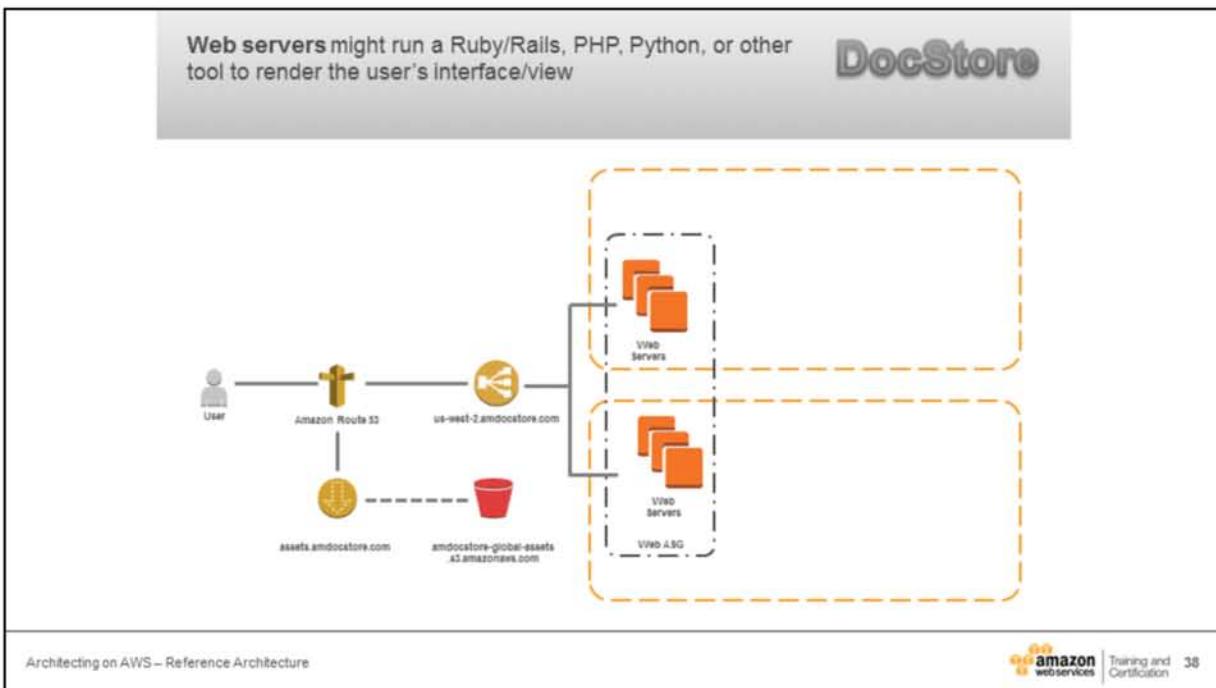
Let's go back to our application deployment in us-west-2 and focus on the browser component of DocStore

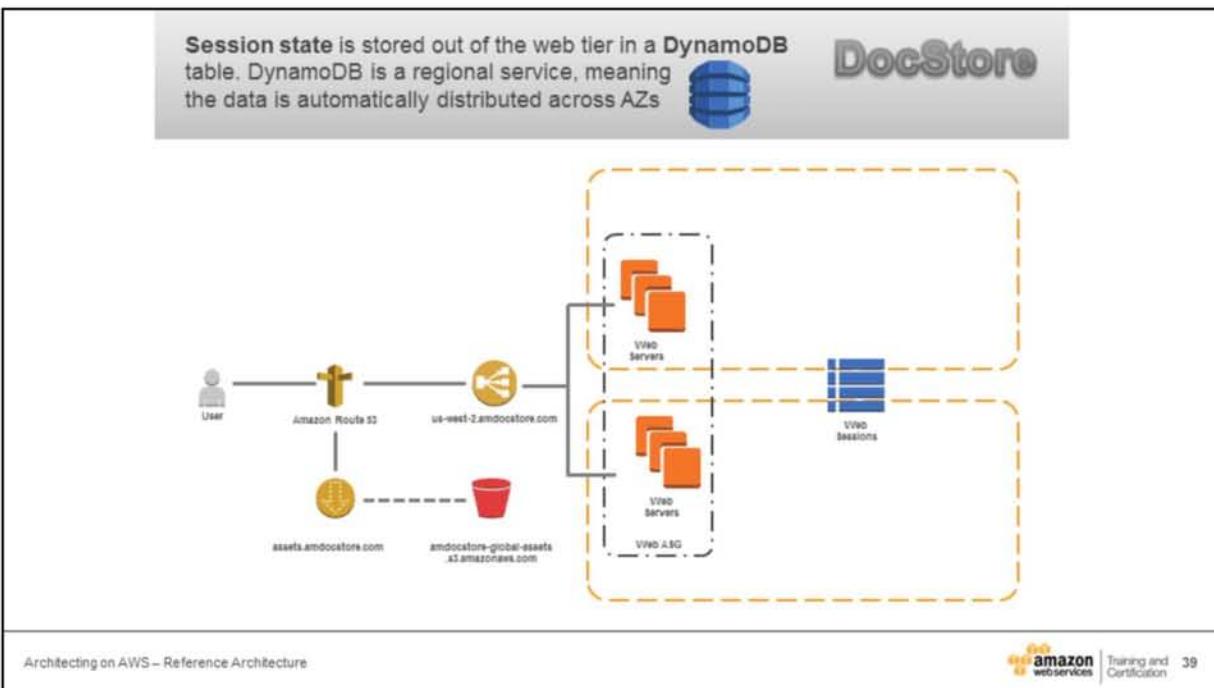


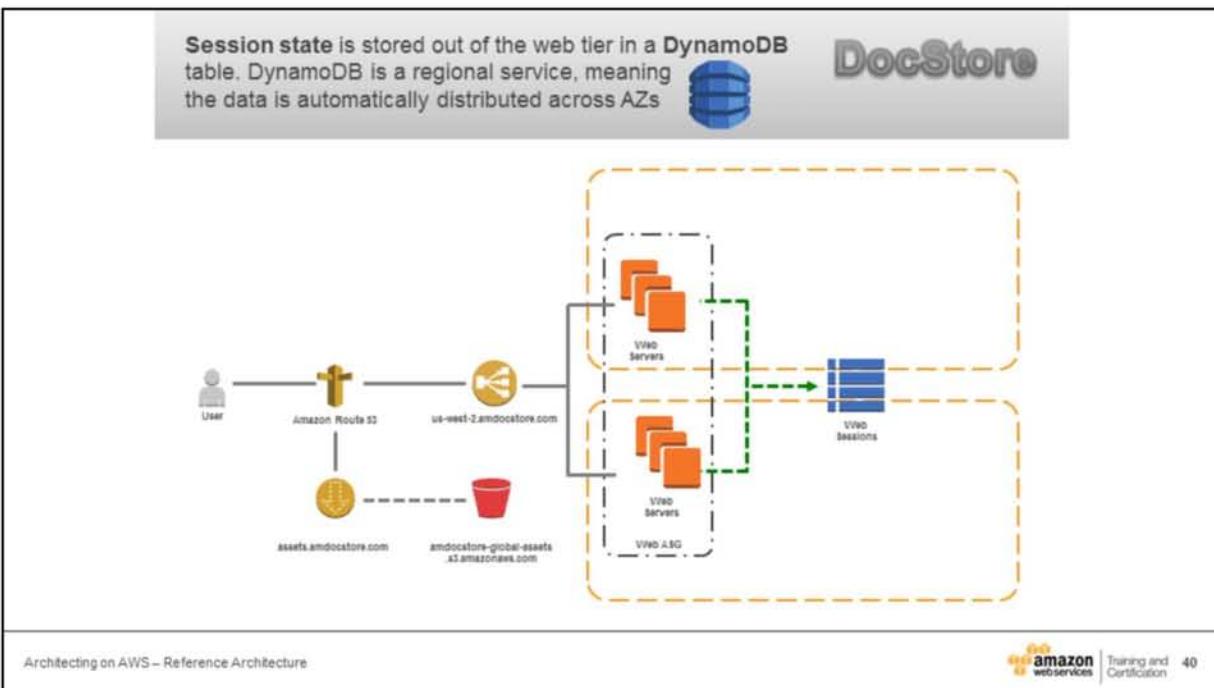


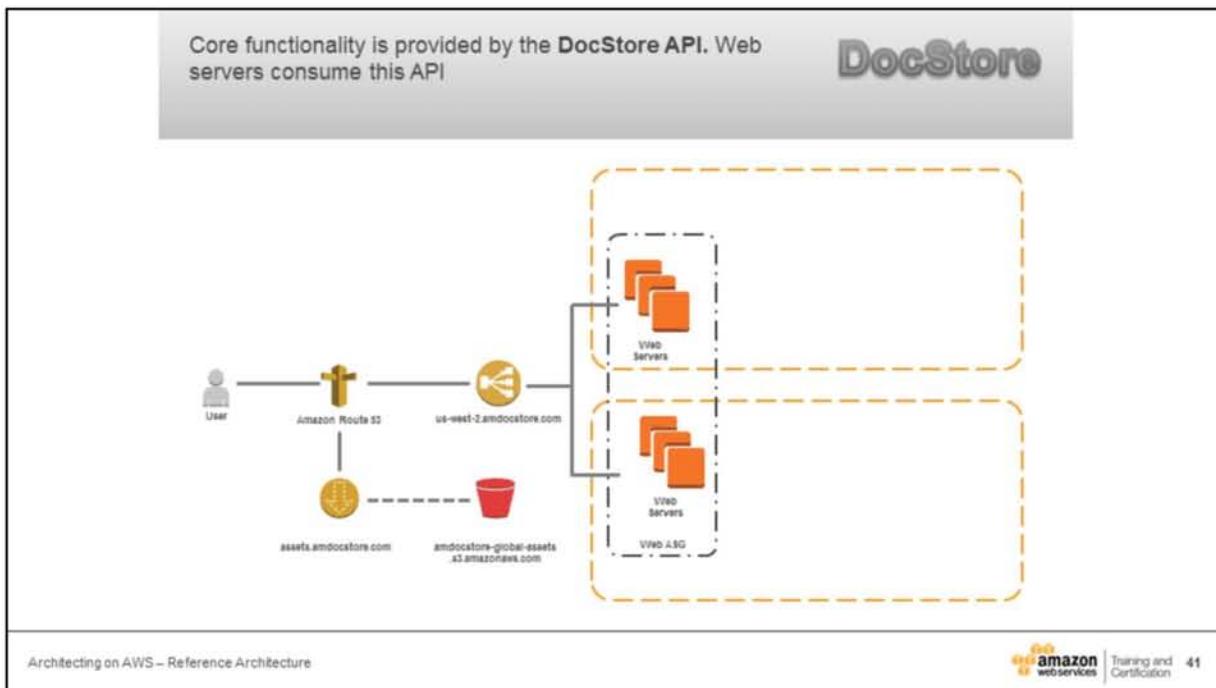


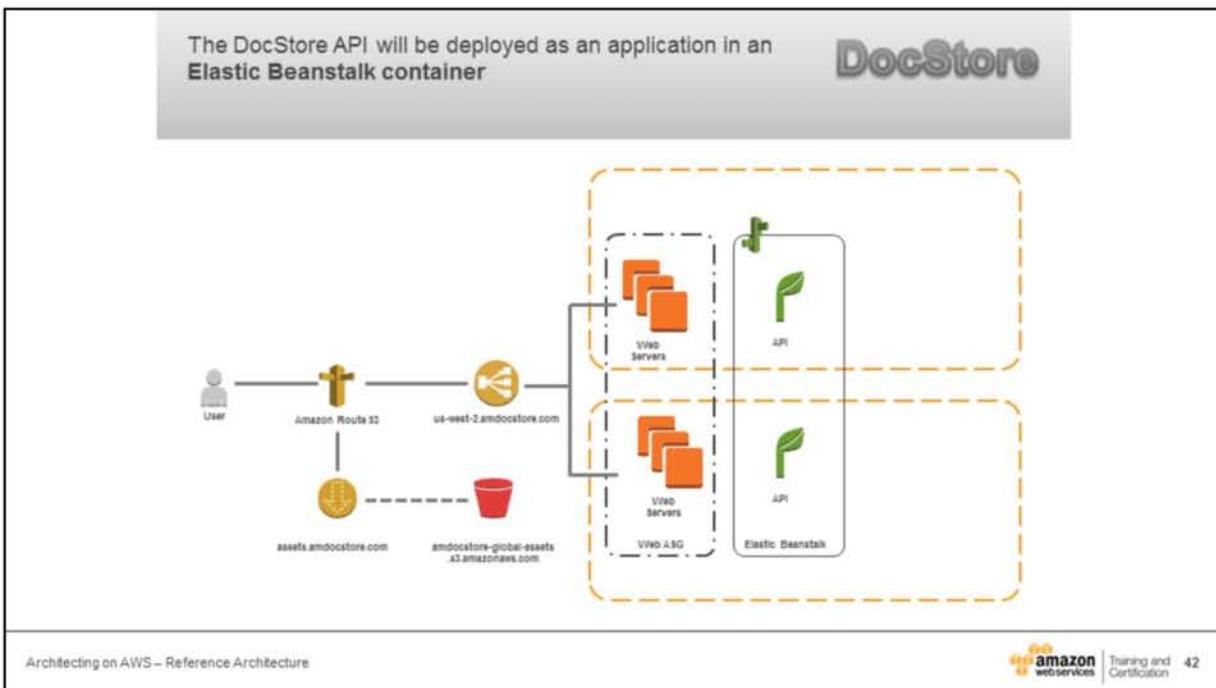


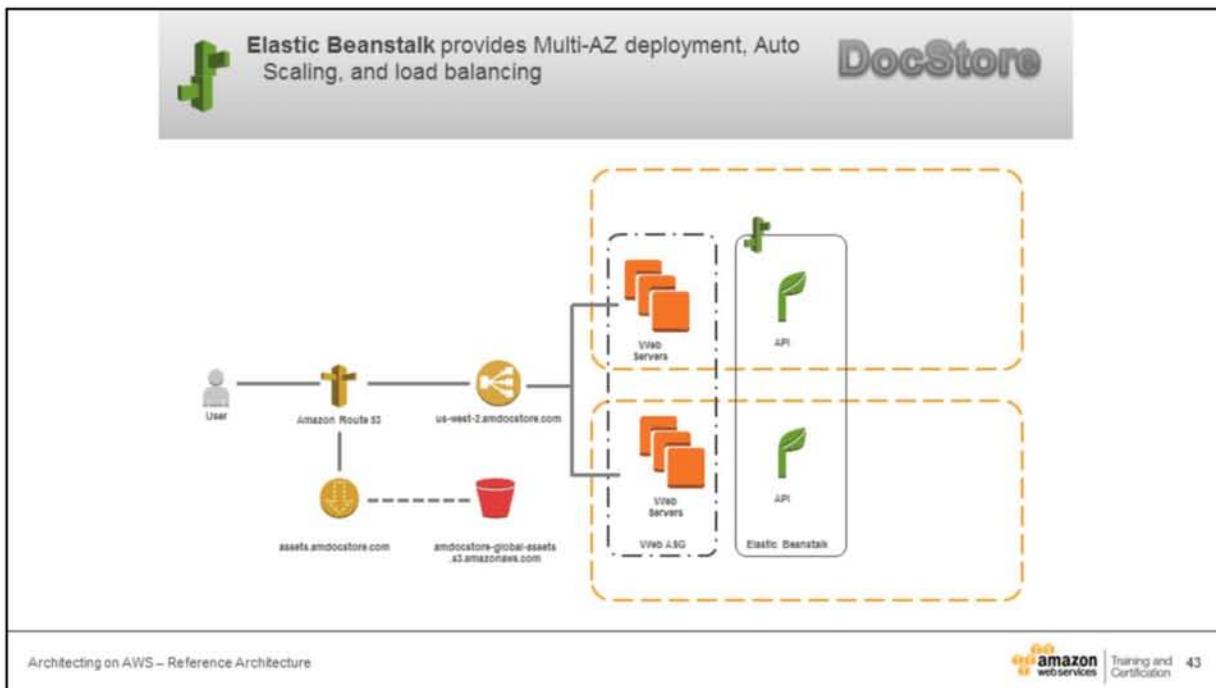


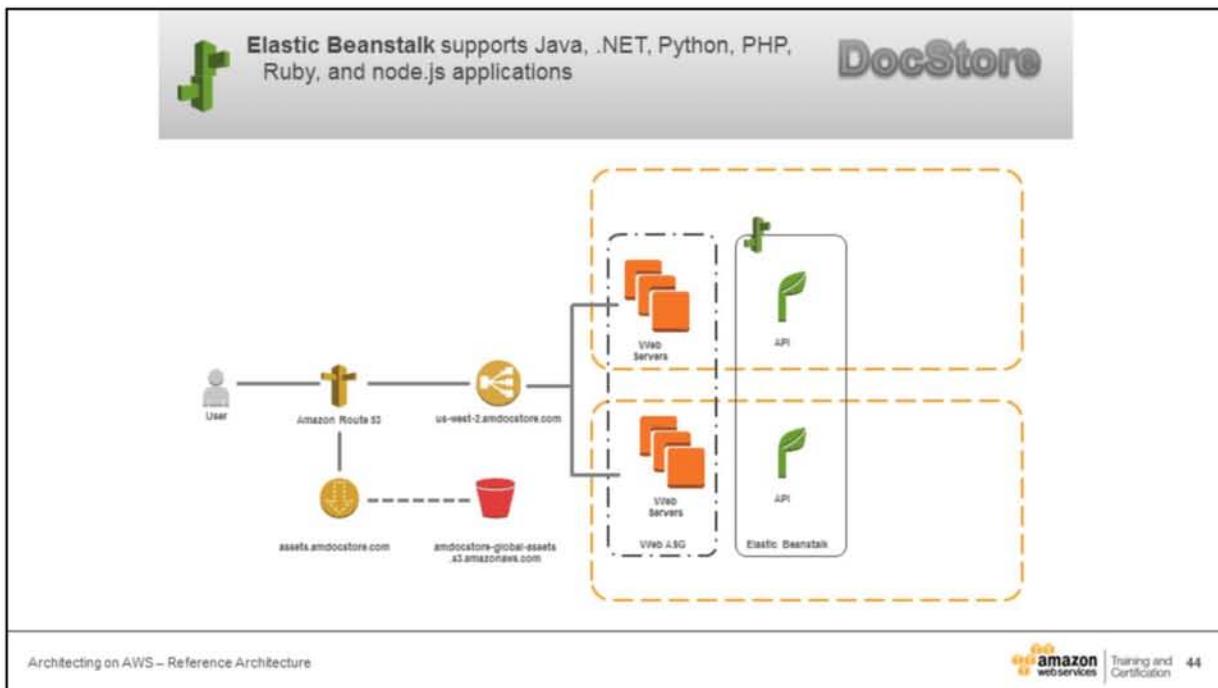






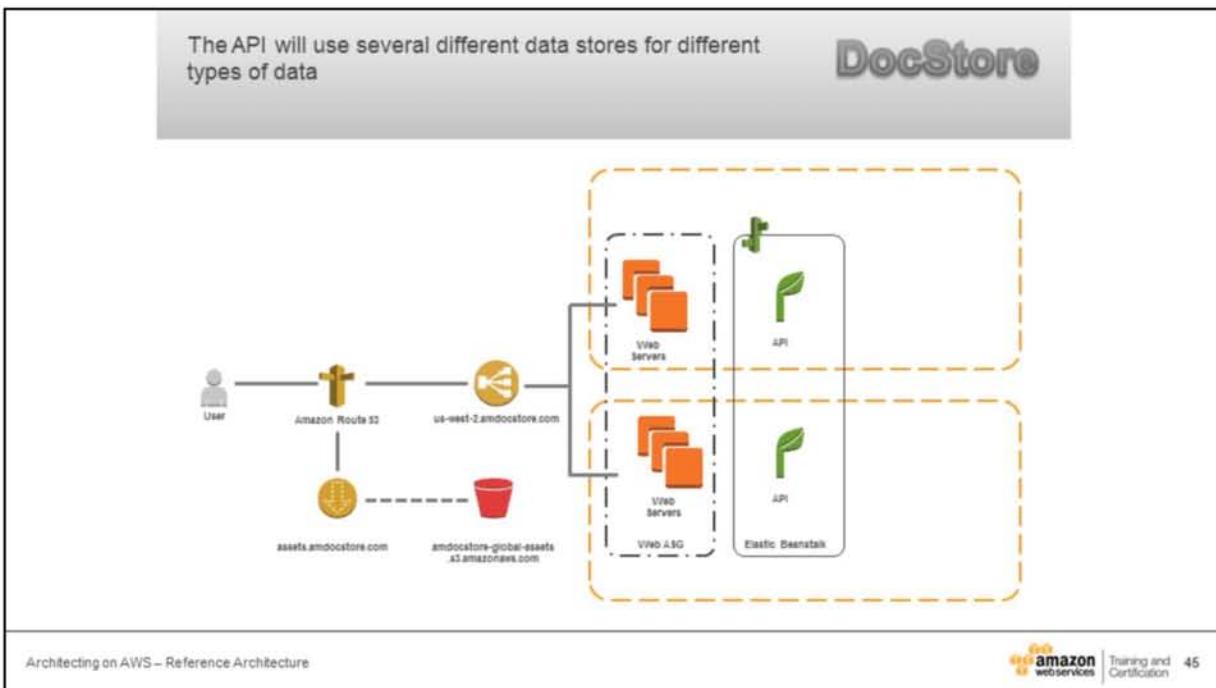


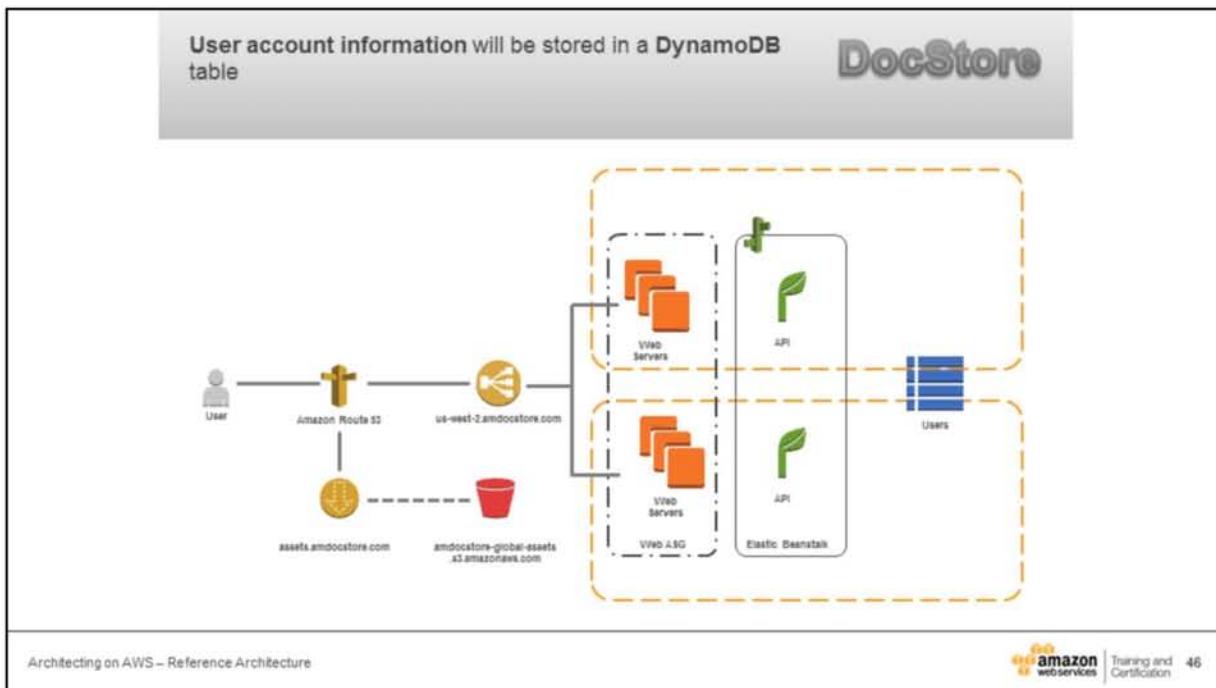


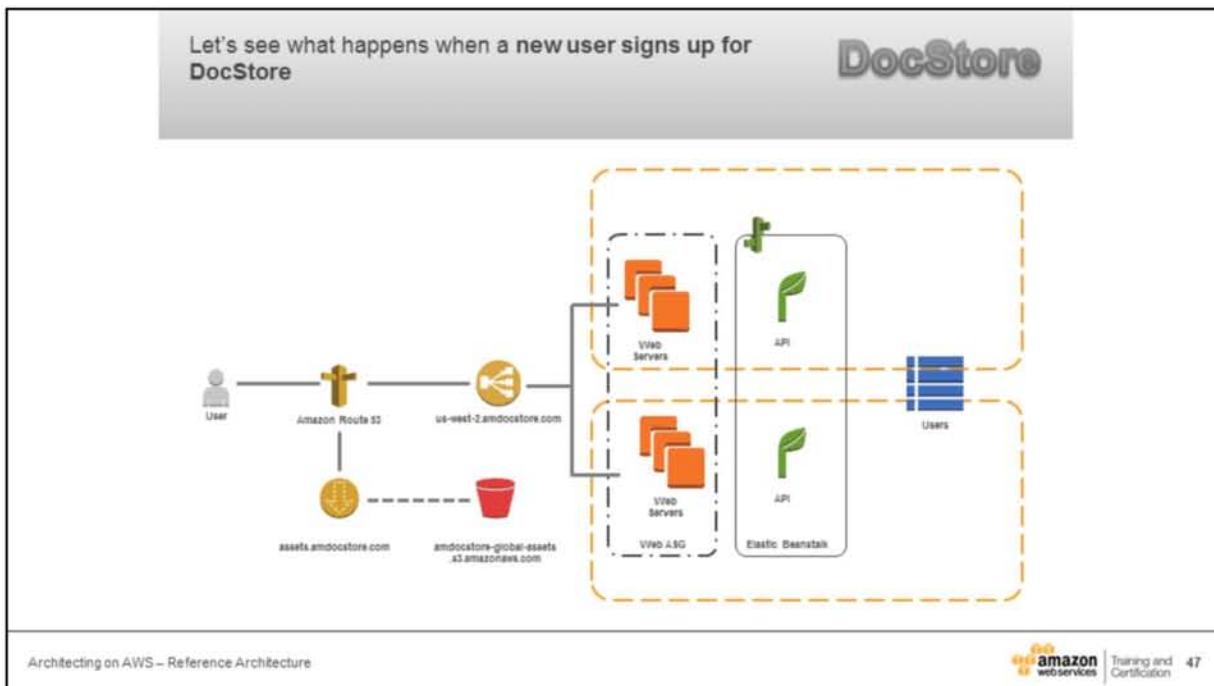


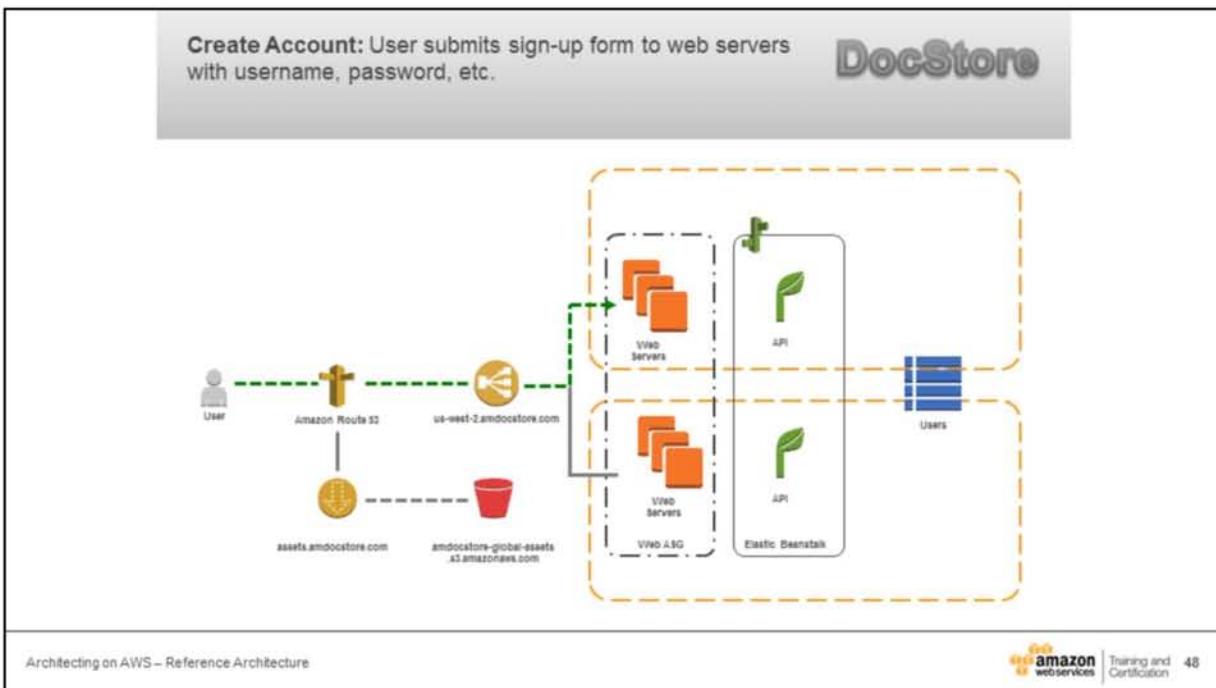
Architecting on AWS – Reference Architecture

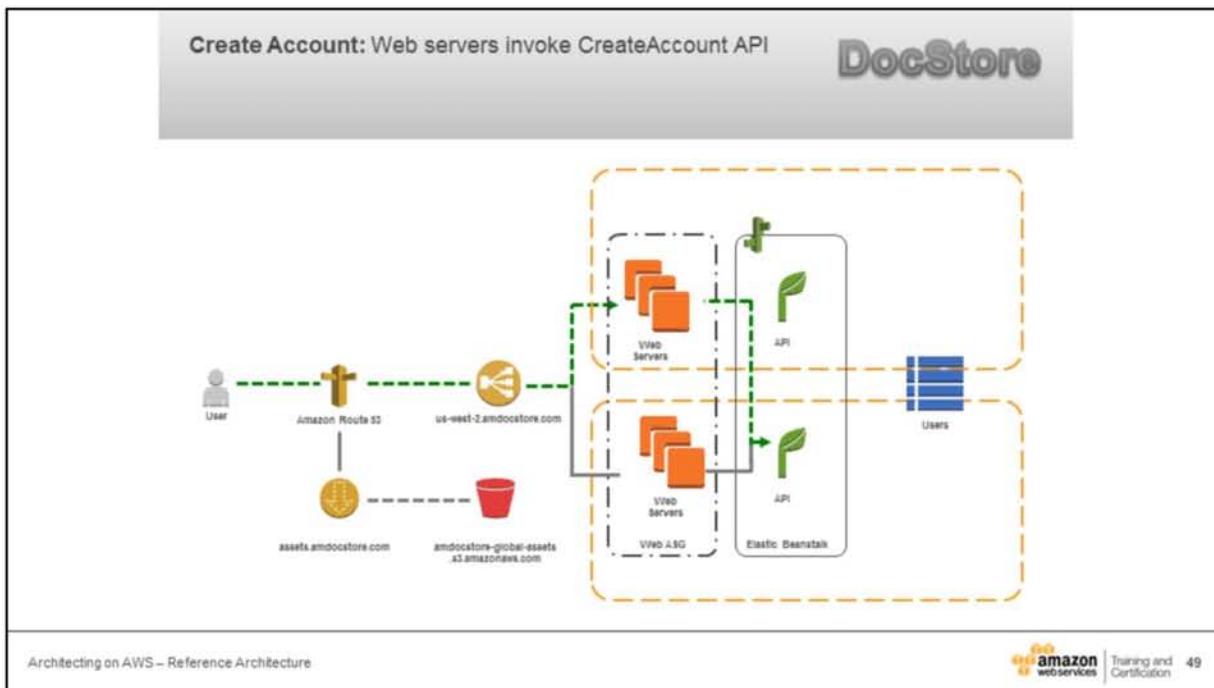
amazon web services | Training and Certification 44

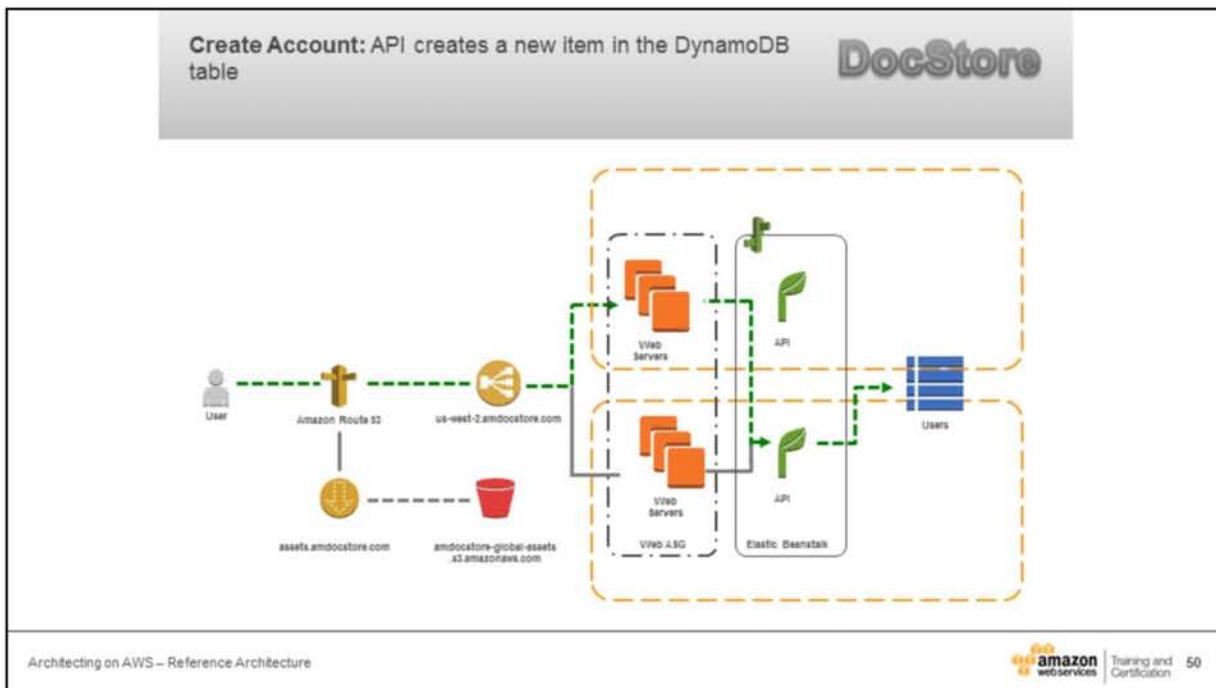


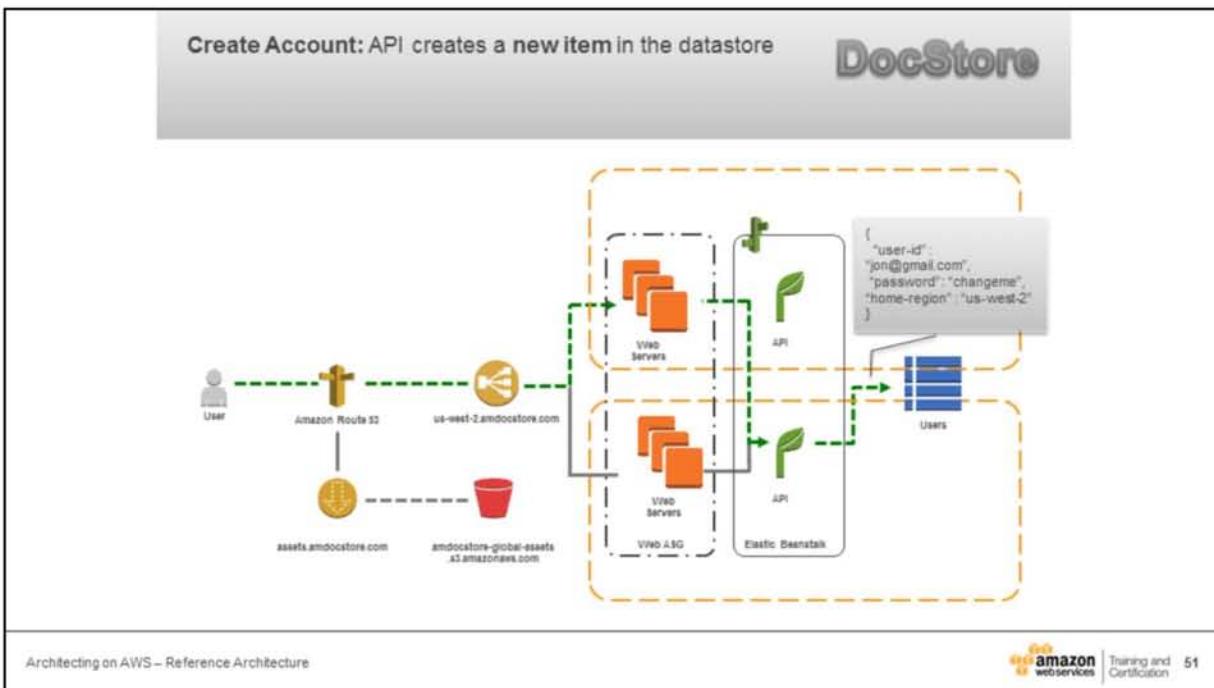


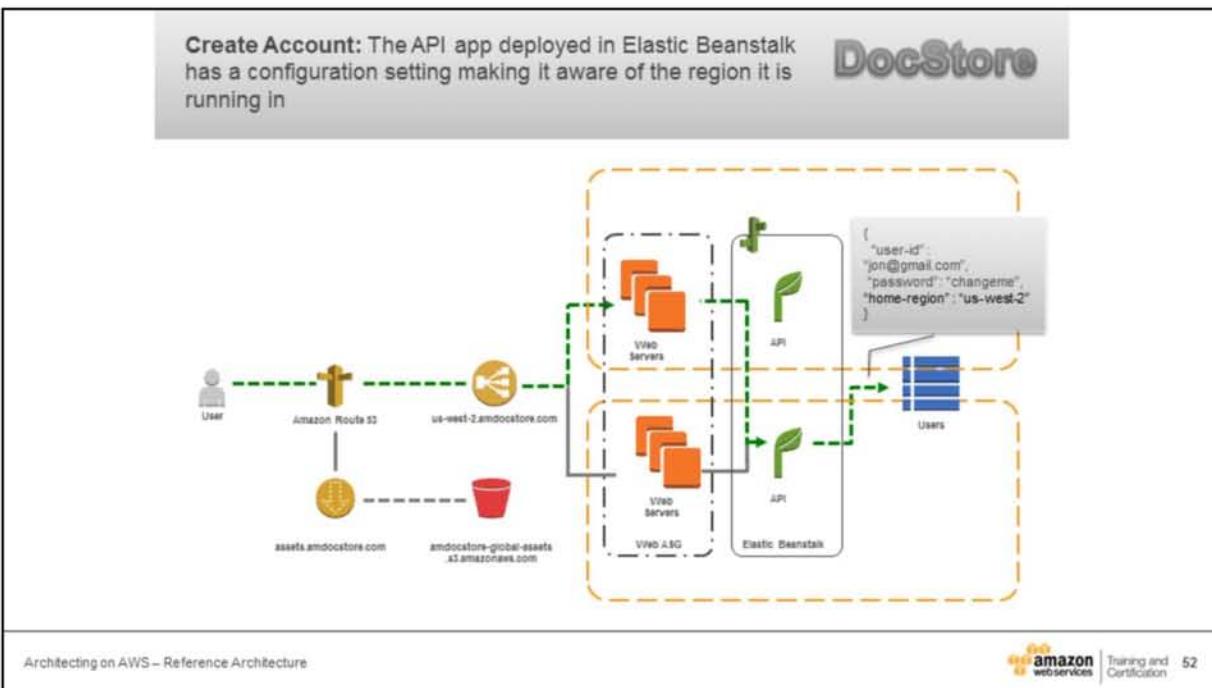


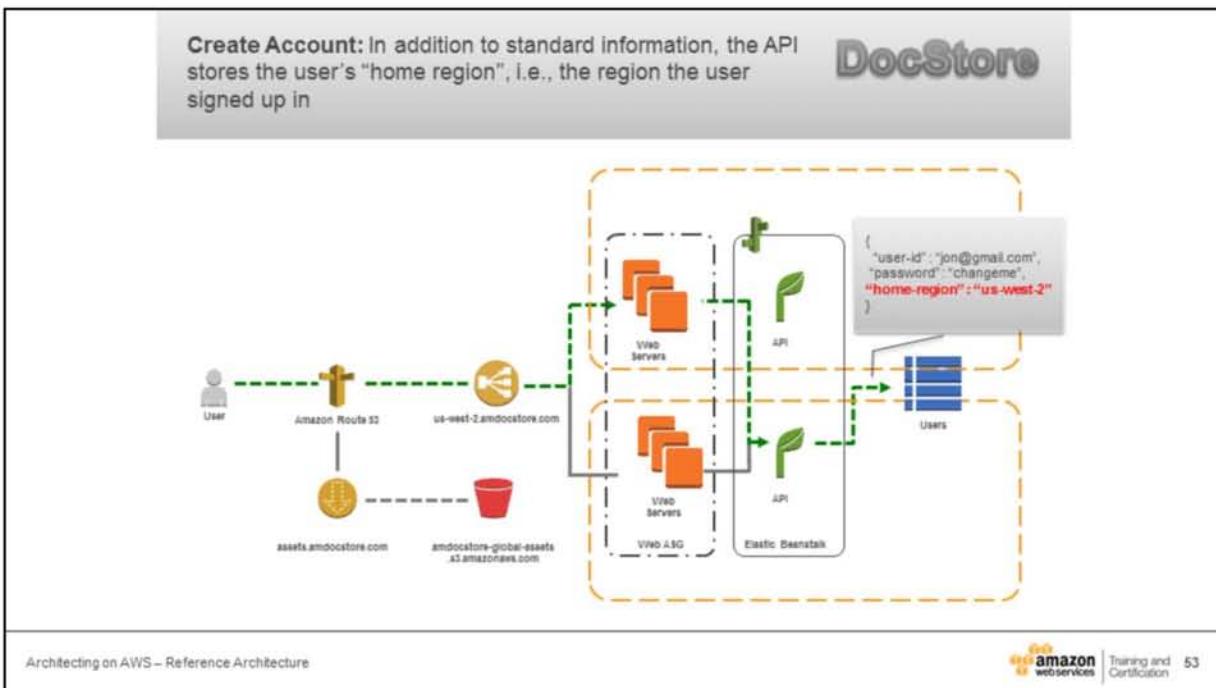


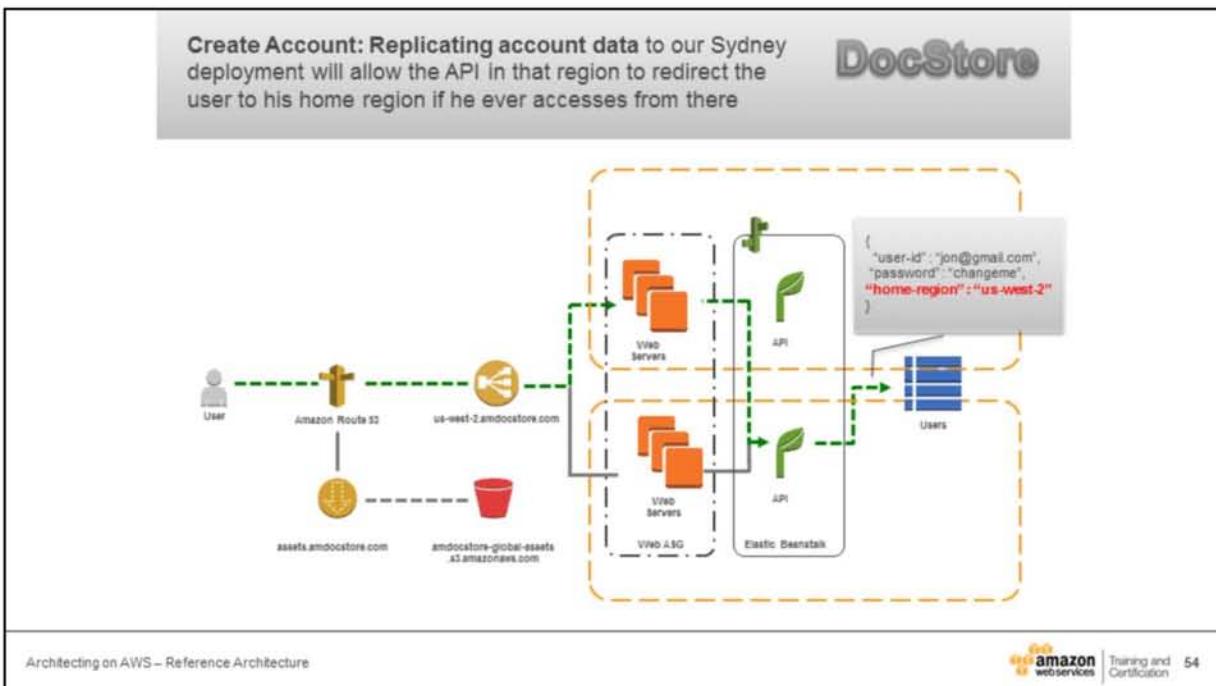


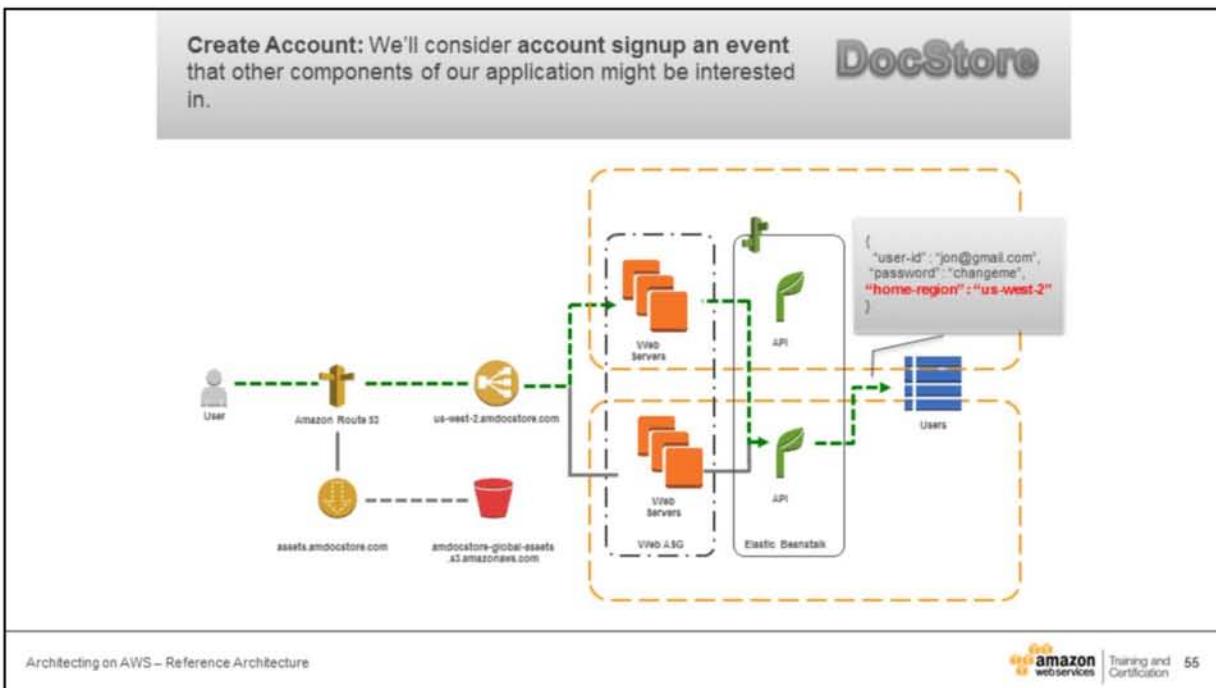


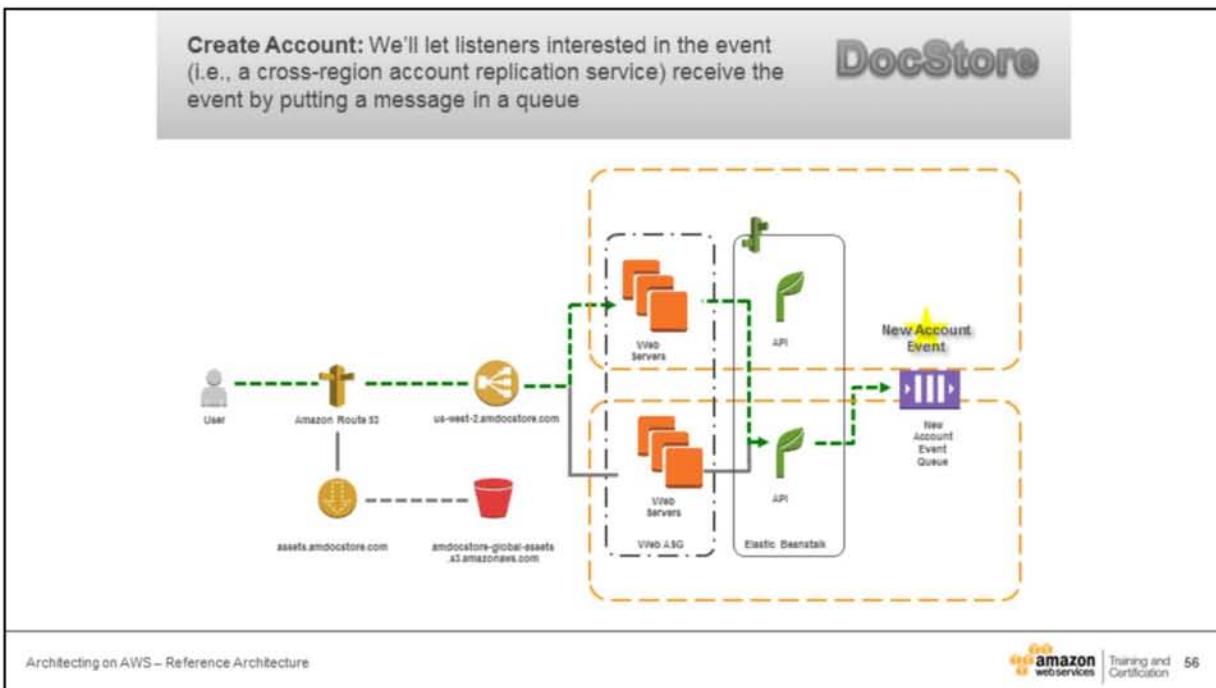


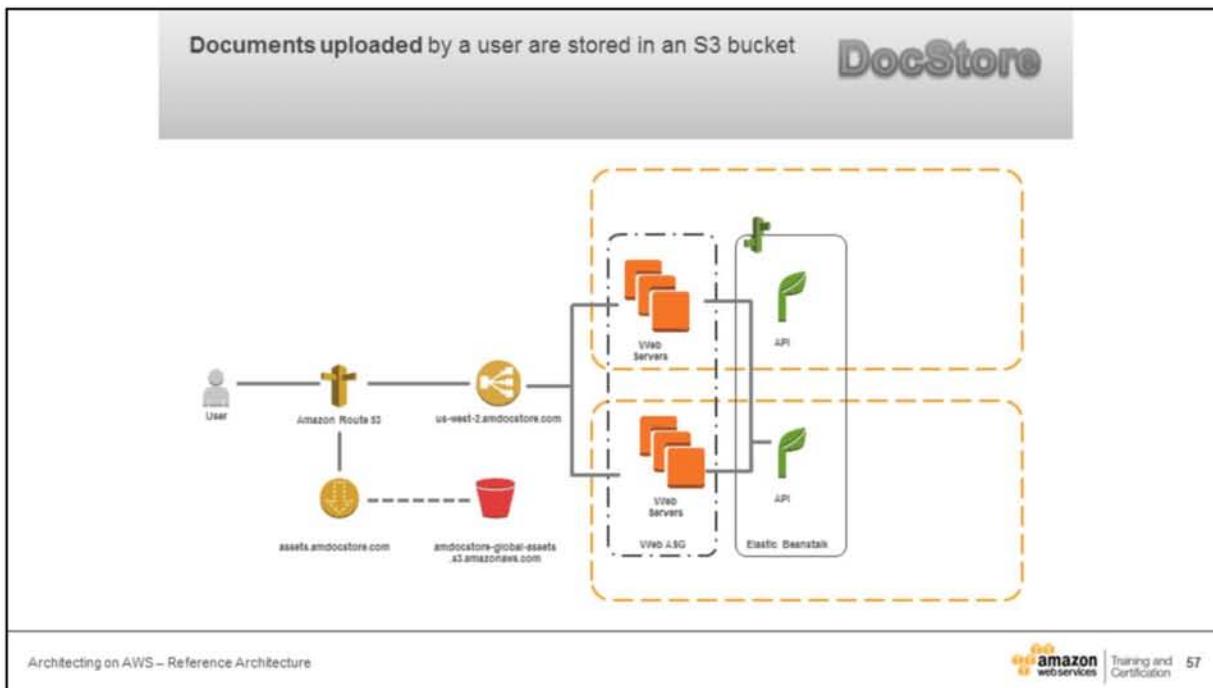


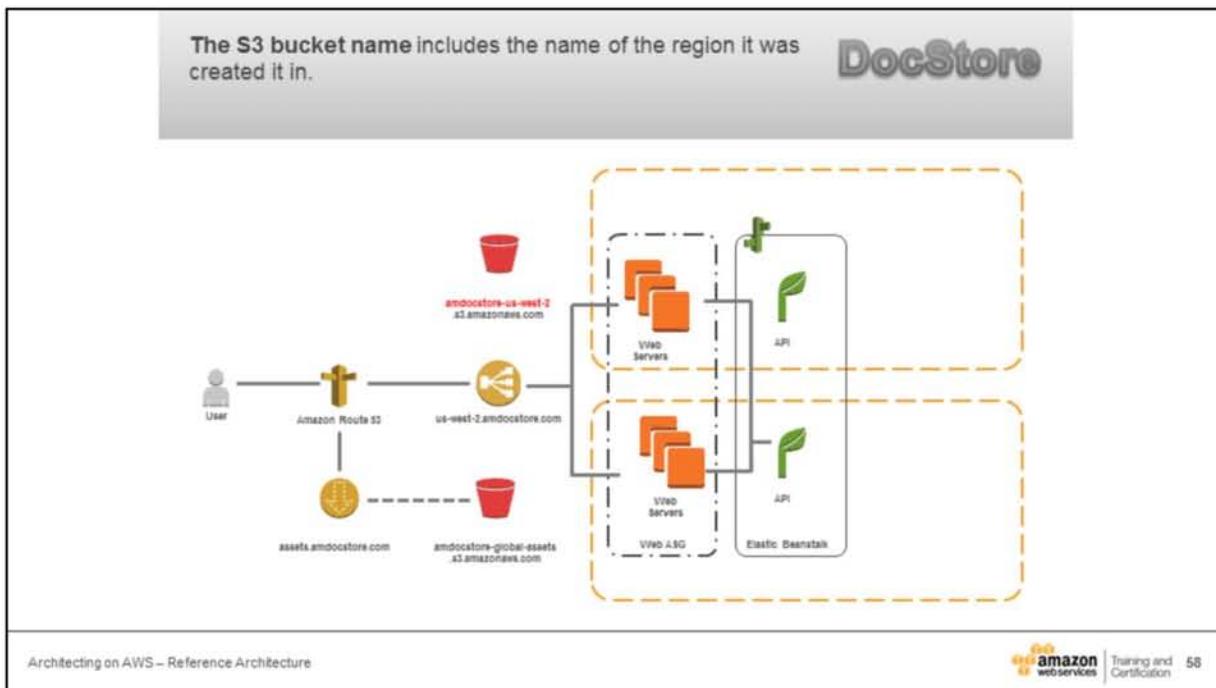


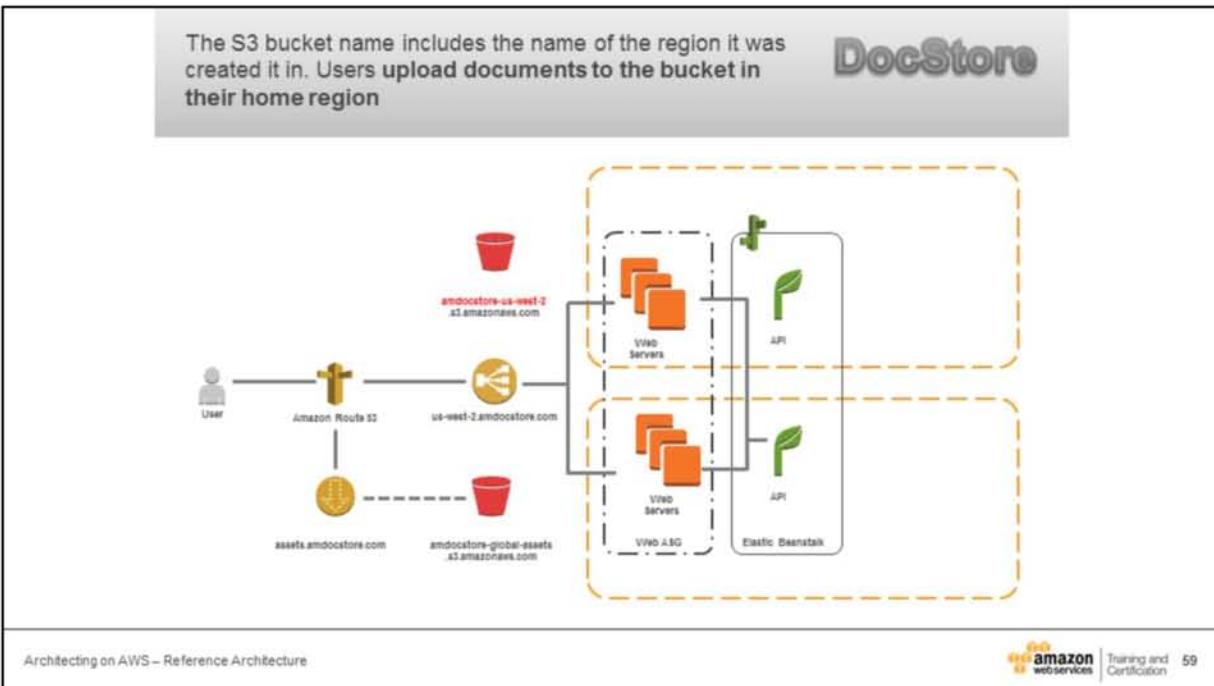


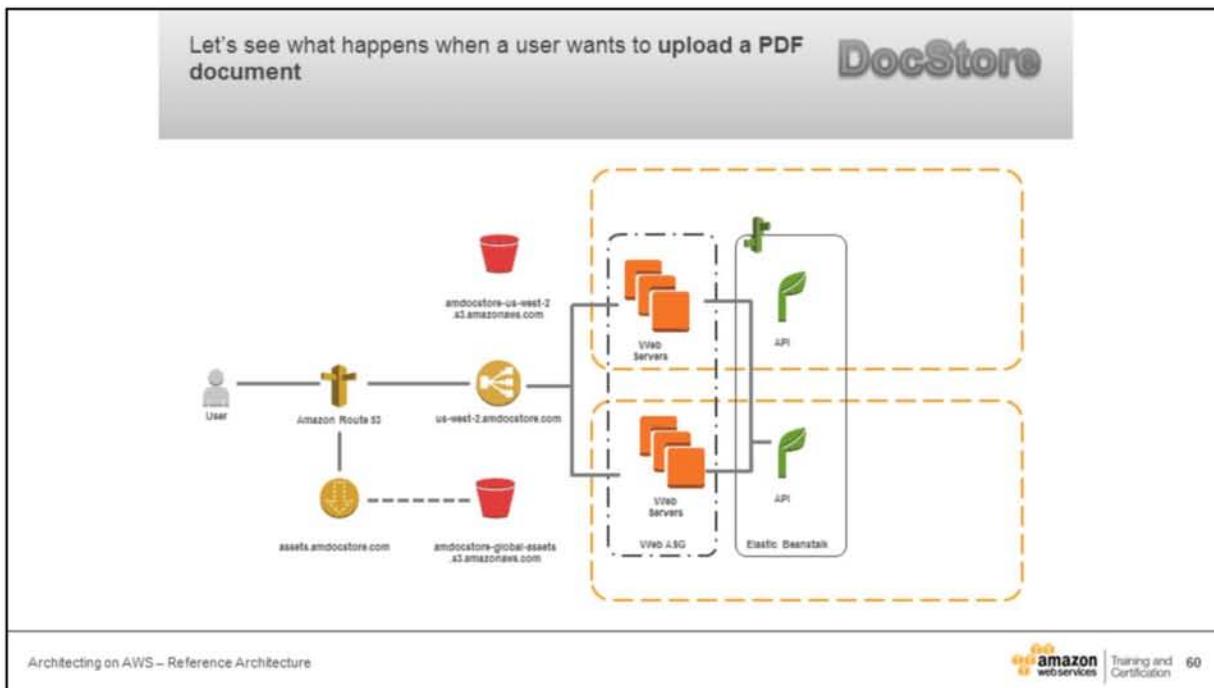


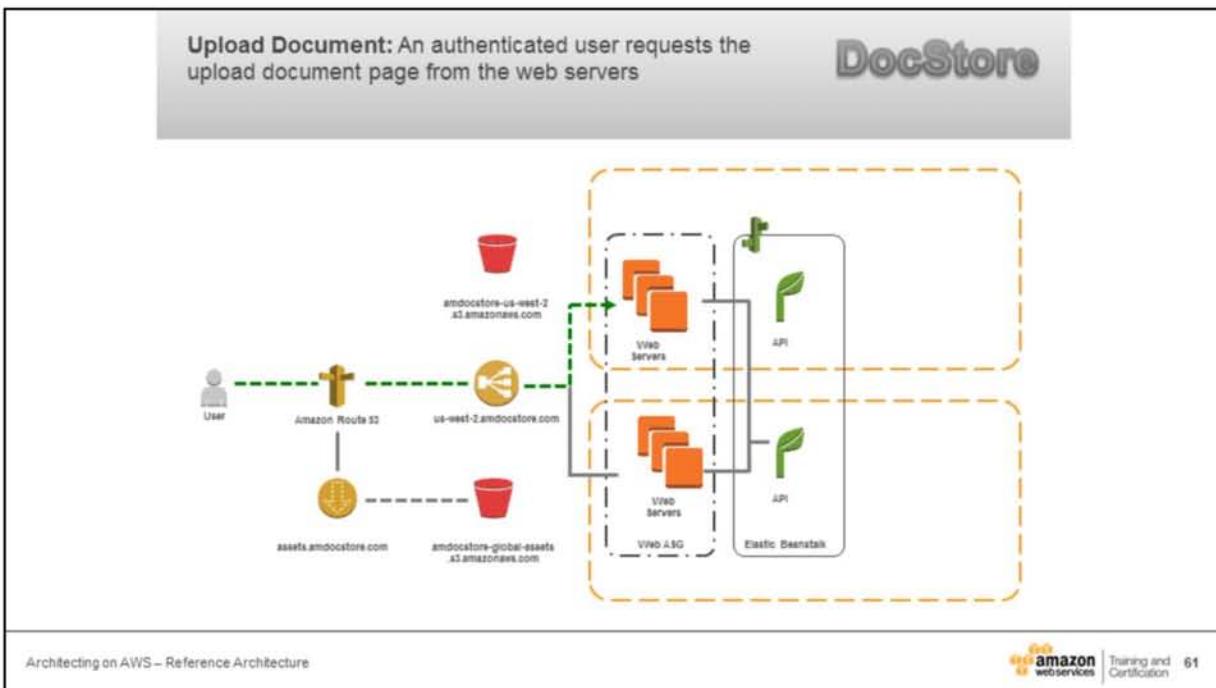


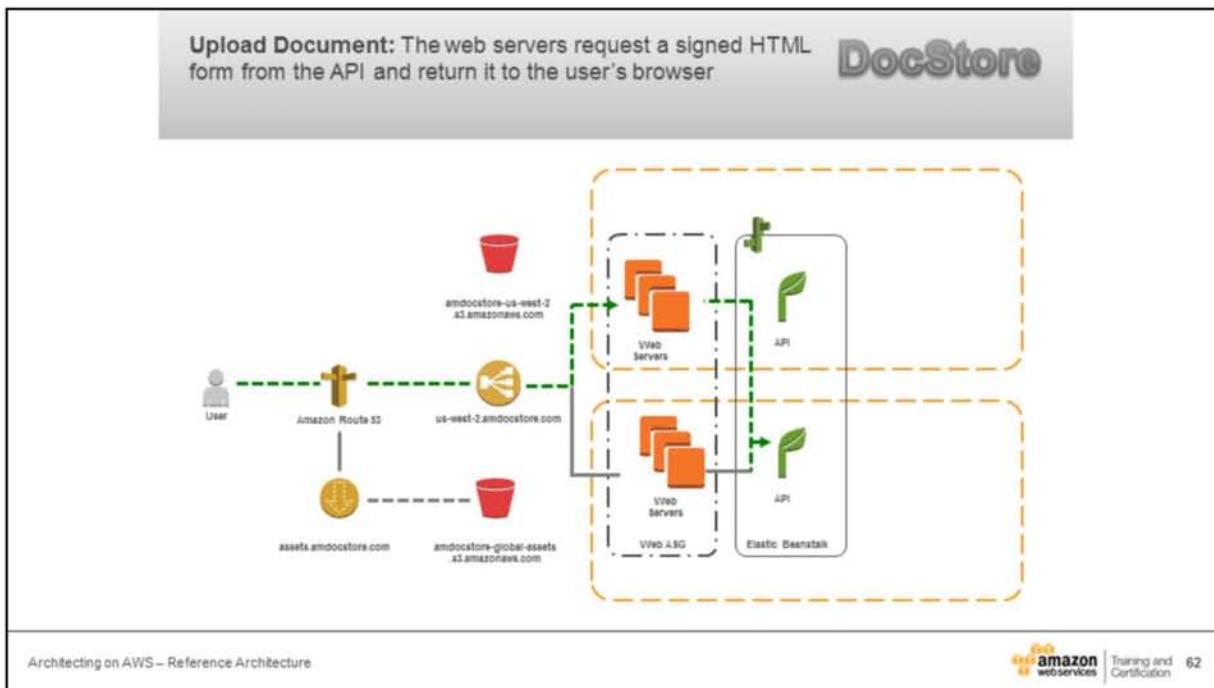


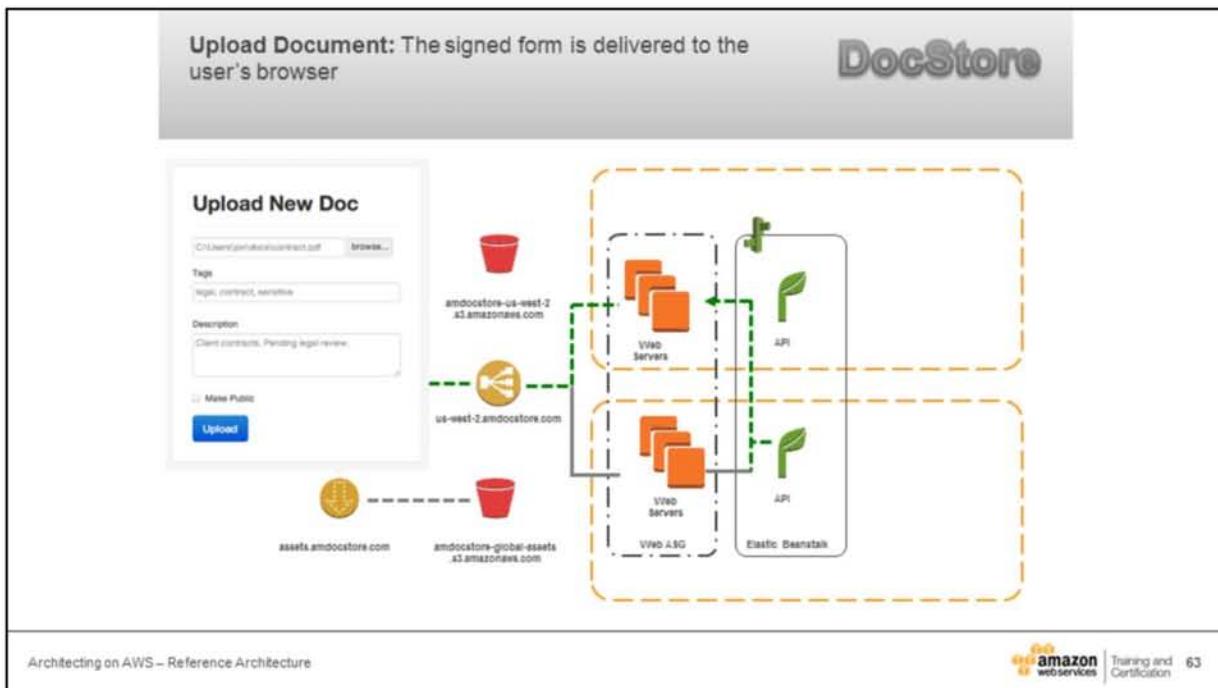


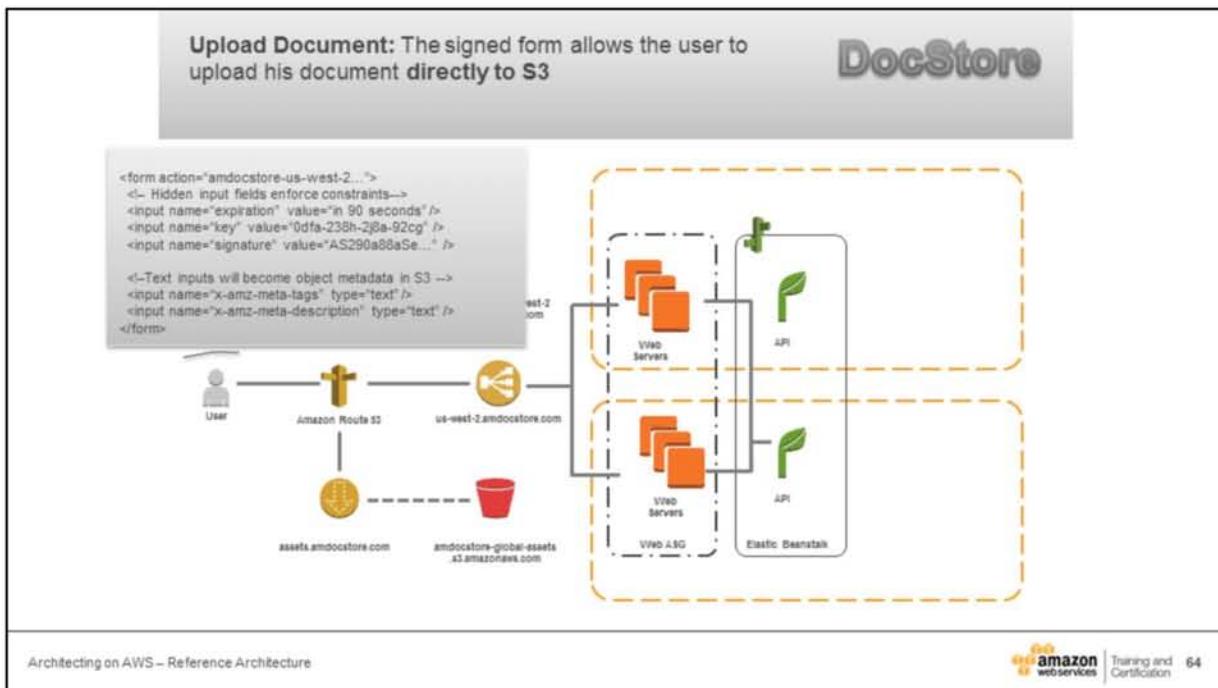


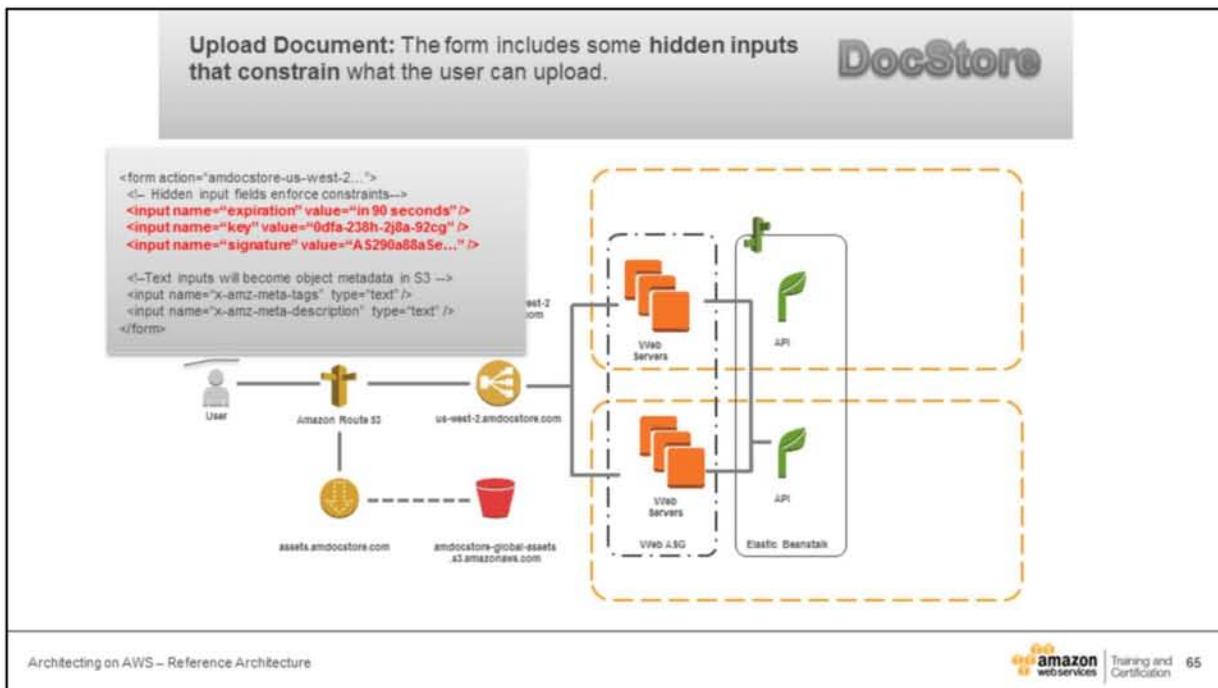


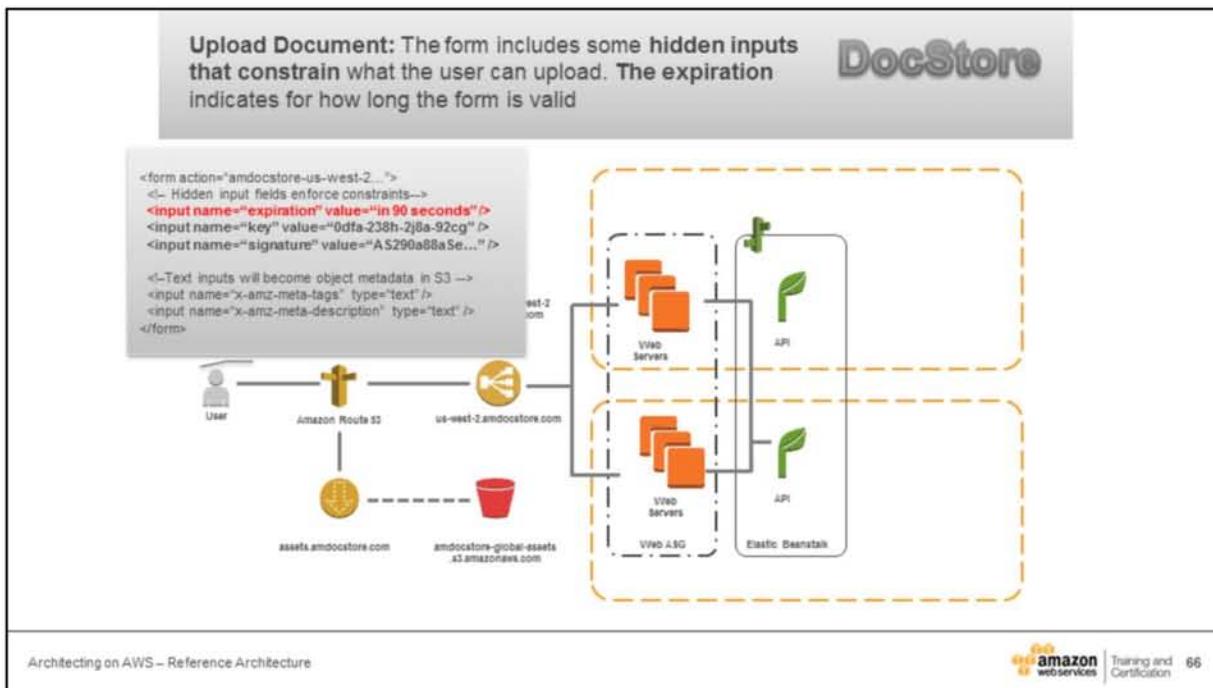


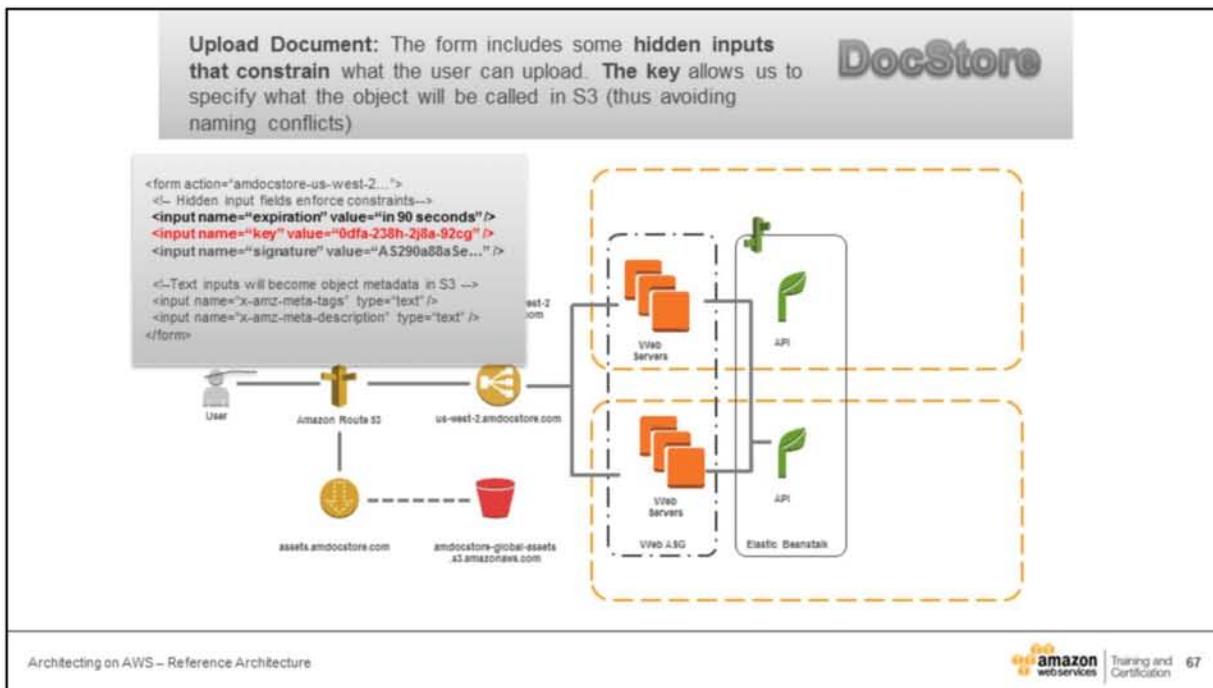


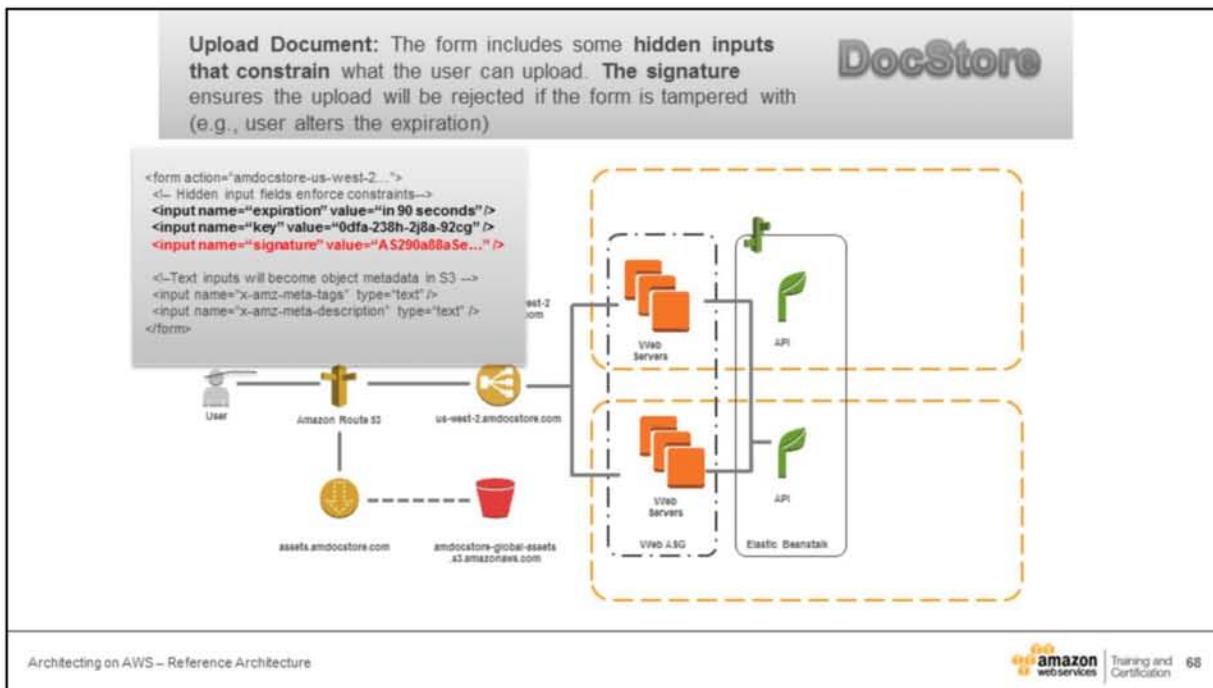












Upload Document: User-editable text input fields prefixed with x-amz-meta- will be associated with the uploaded object as S3 object metadata

DocStore

Upload New Doc

C:\Users\jon\docs\contract.pdf

Tags
legal, contract, sensitive

Description
Client contracts. Pending legal review.

Make Public

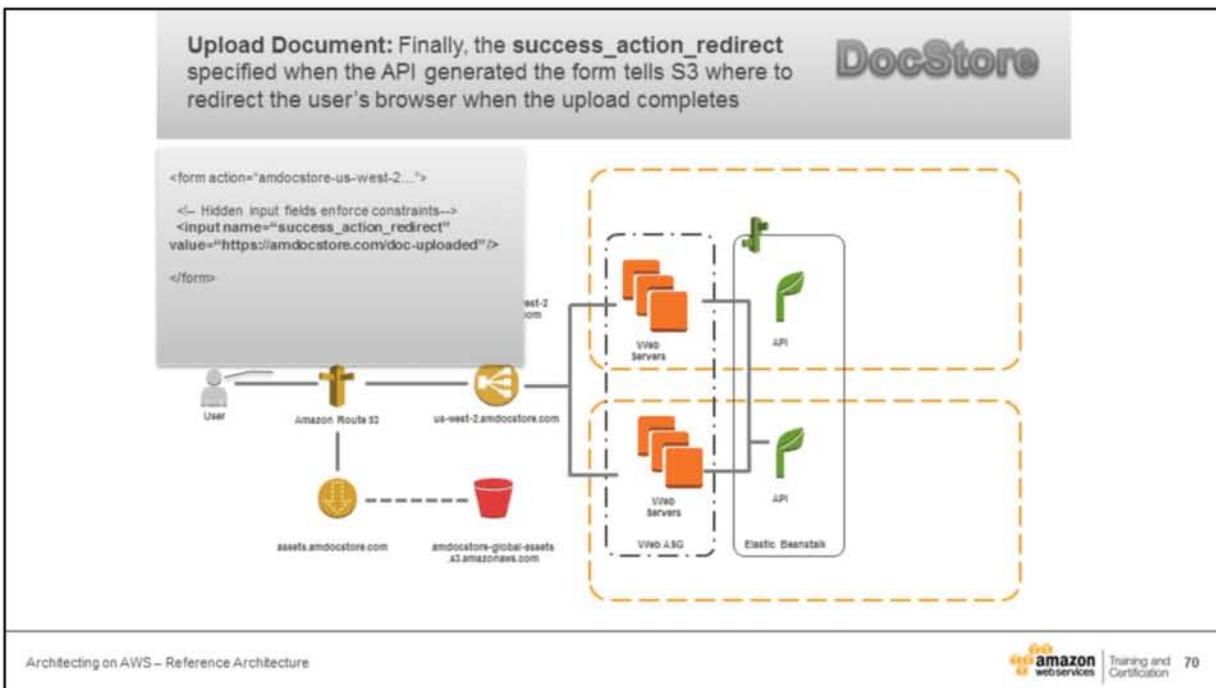
Diagram:

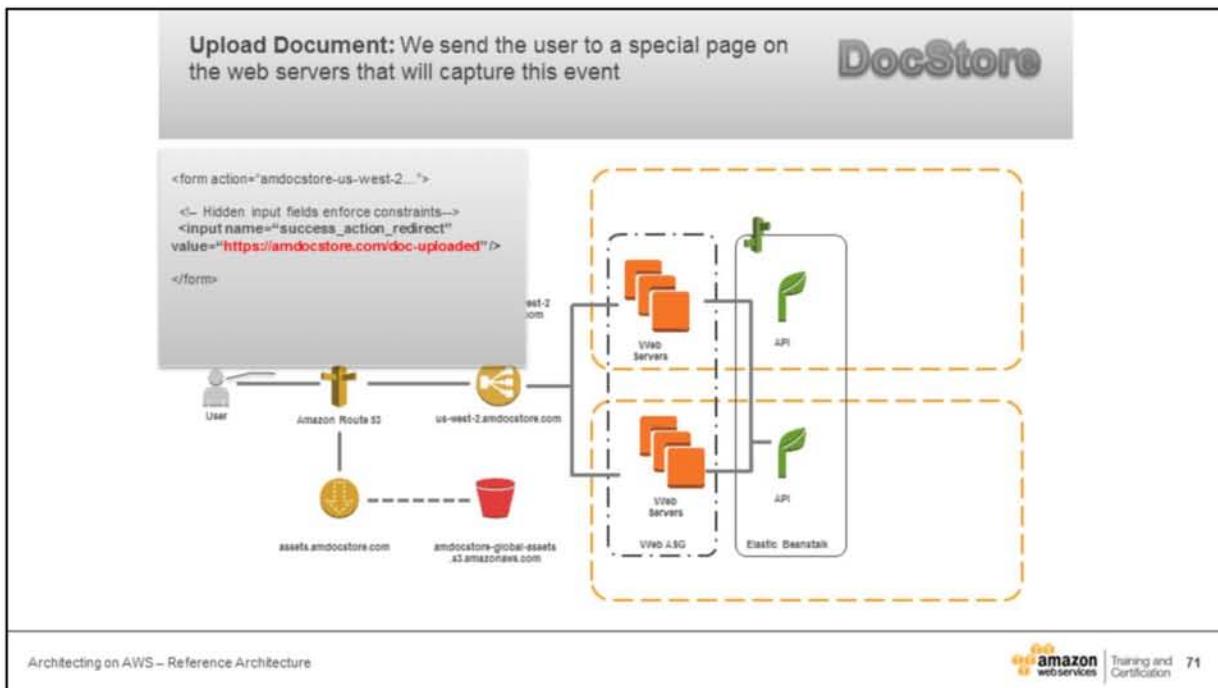
User → Amazon Route 53 → us-west-2.amdocstore.com

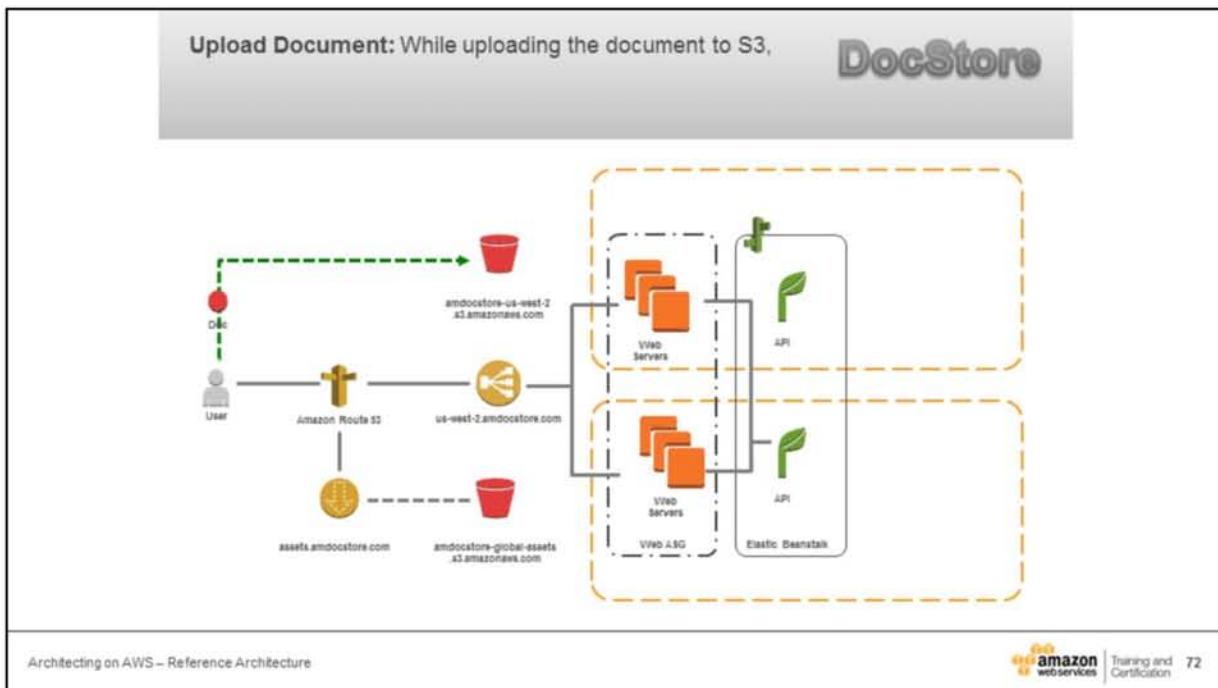
assetsx.amdocstore.com ← S3 Bucket → amdocstore-global-assets.s3.amazonaws.com

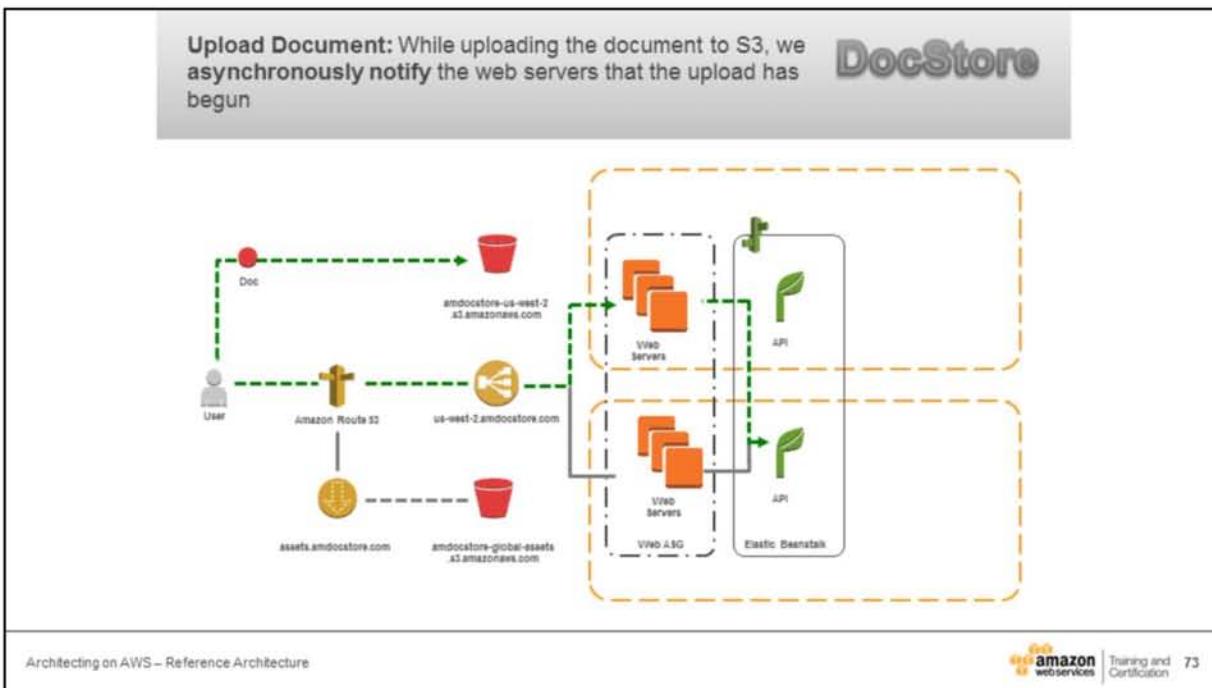
Architecting on AWS – Reference Architecture

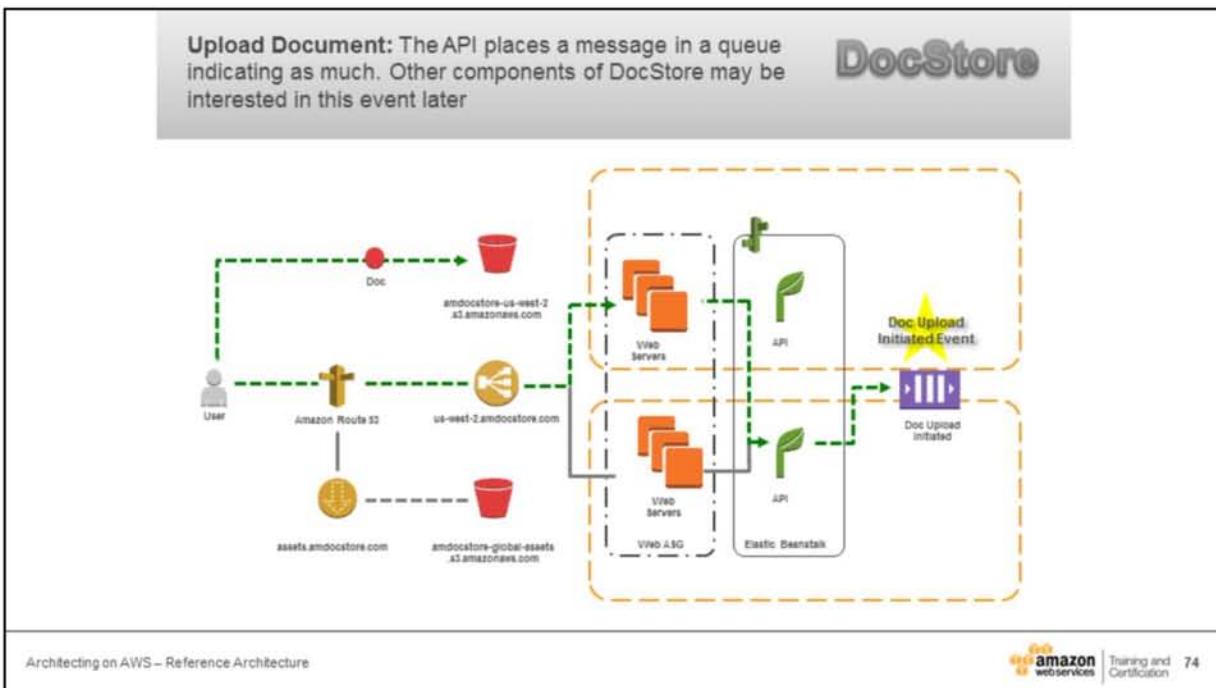
amazon web services | Training and Certification 69

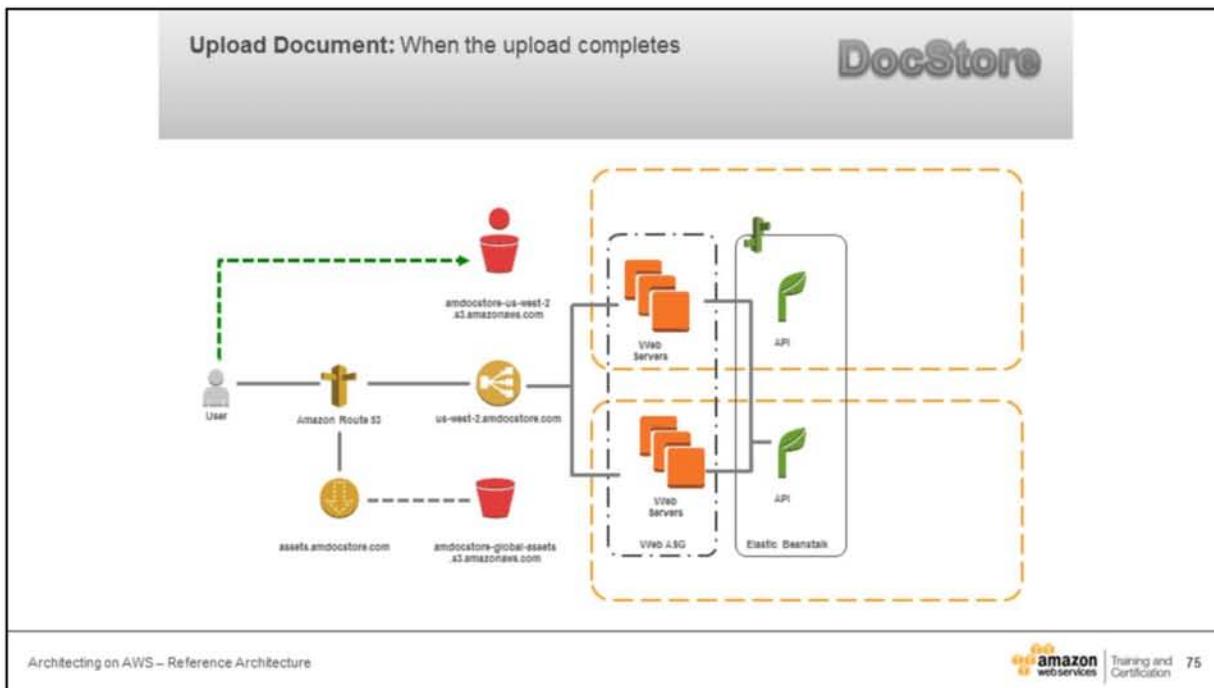


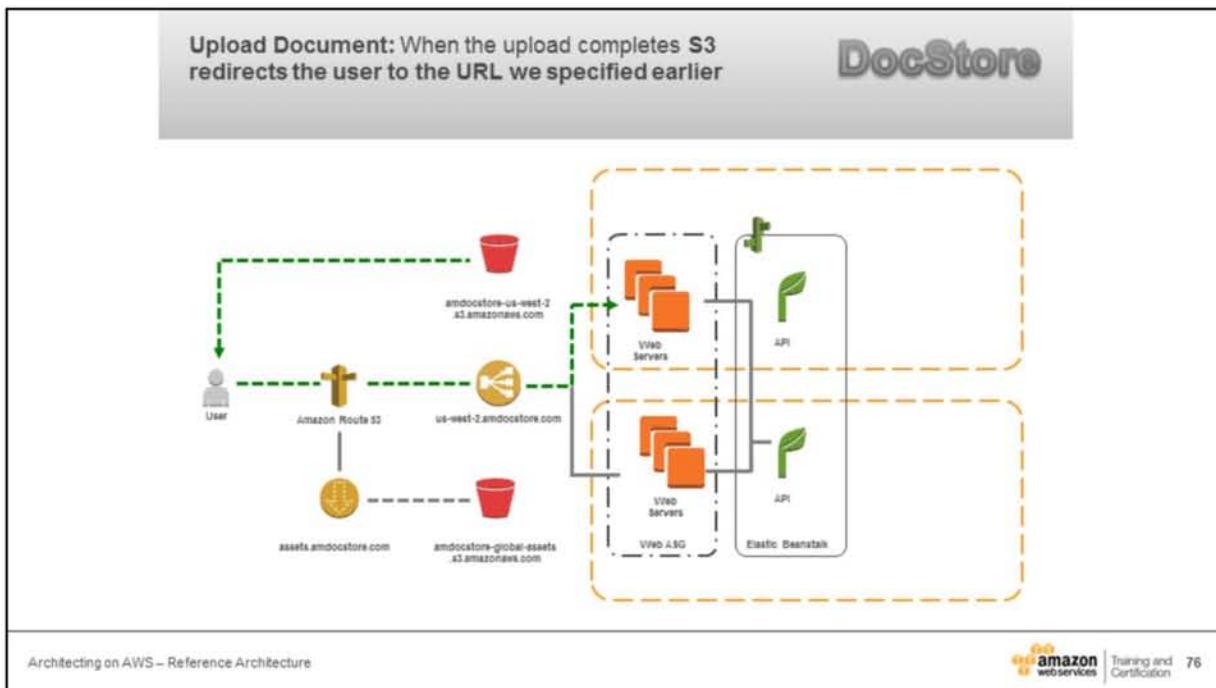


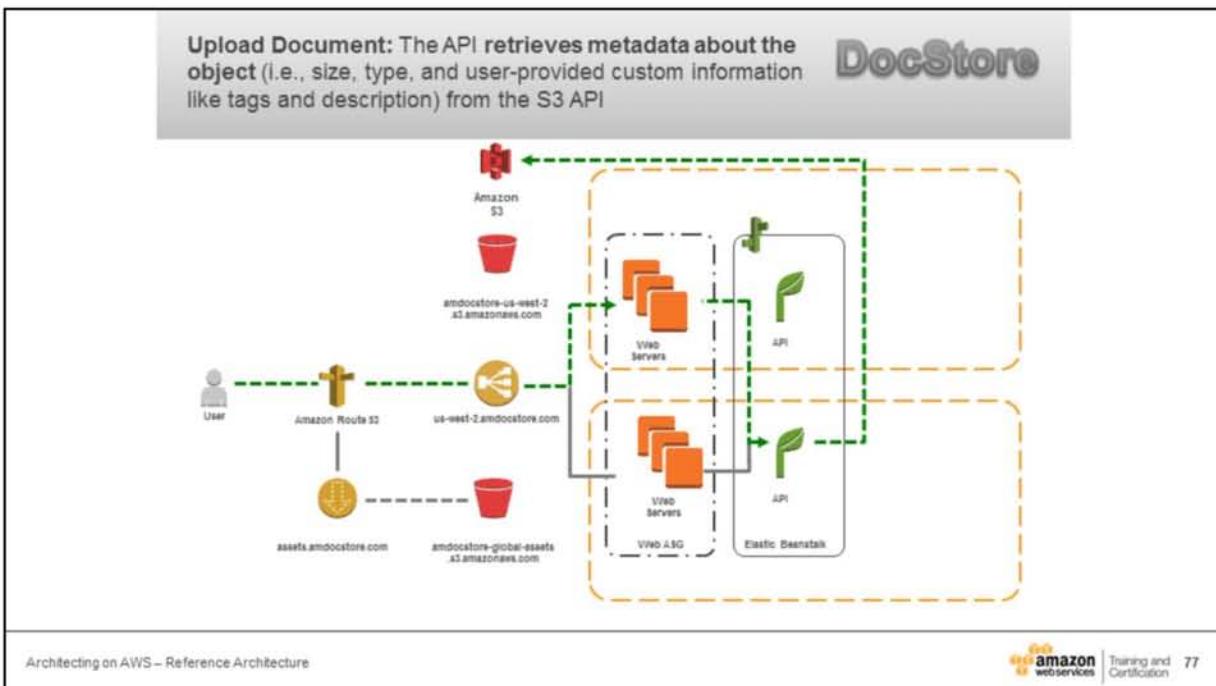


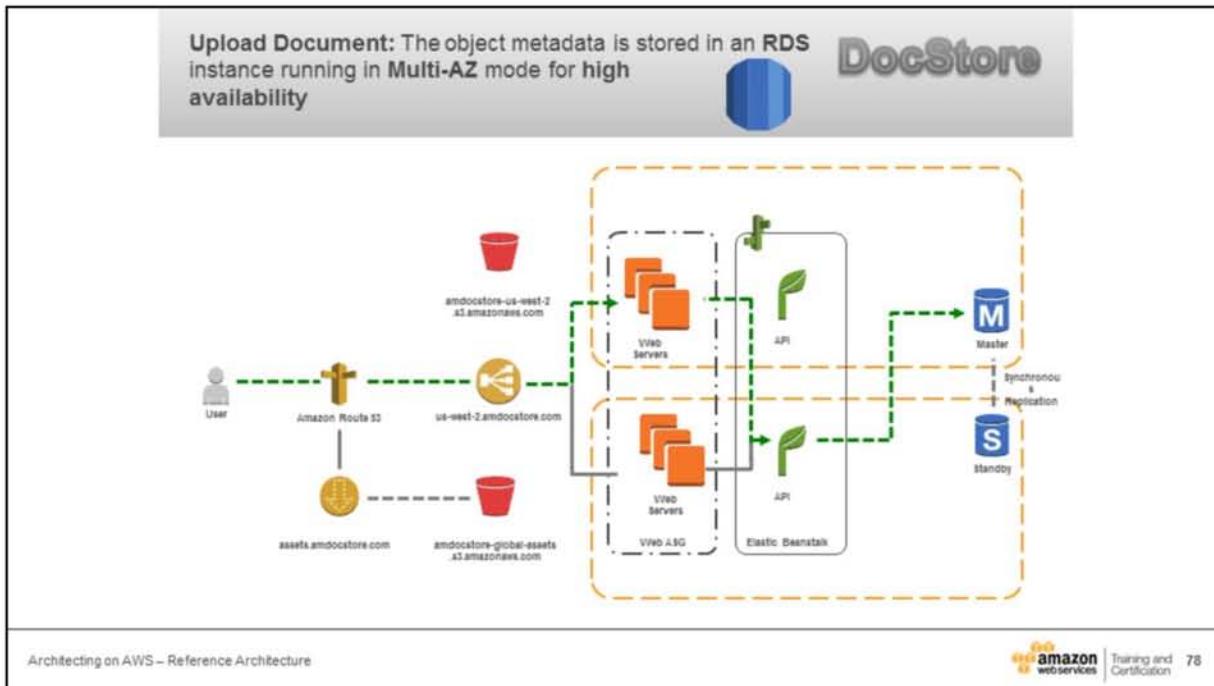


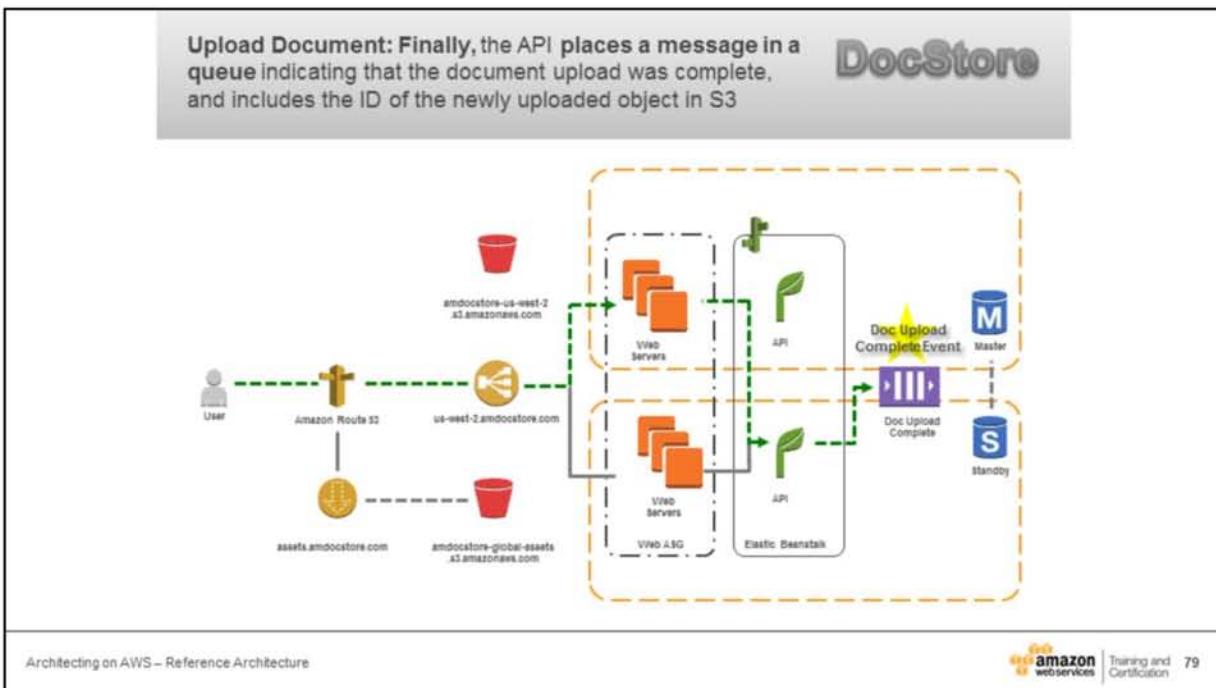


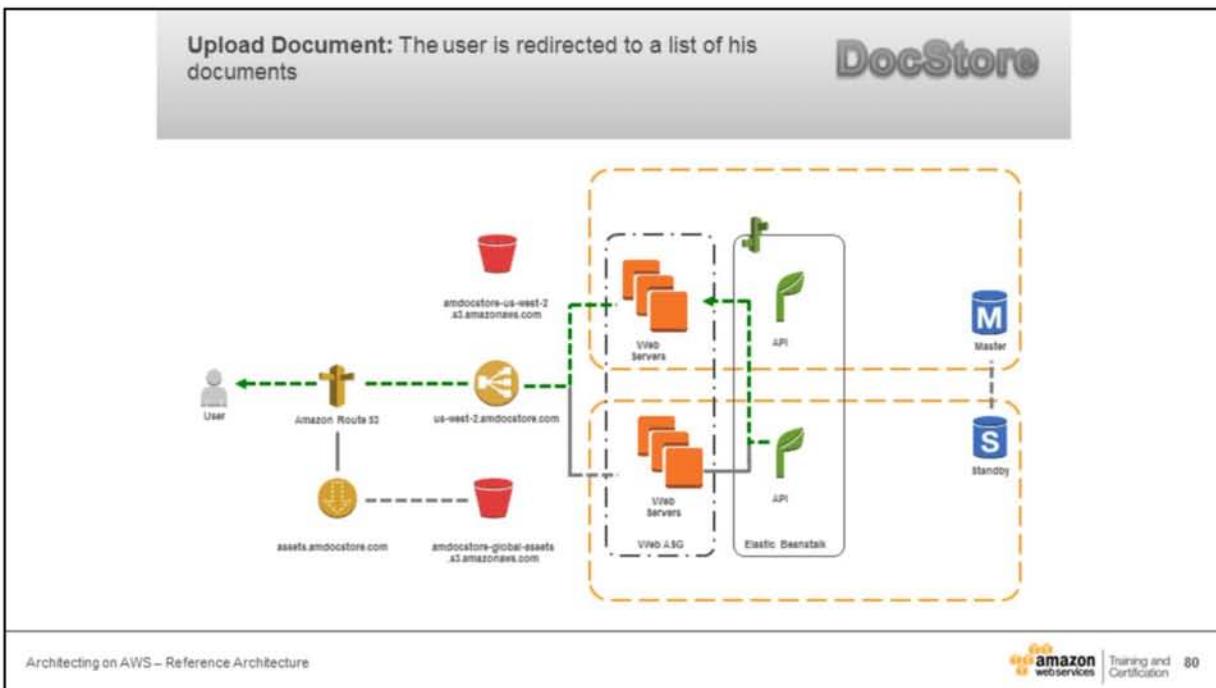


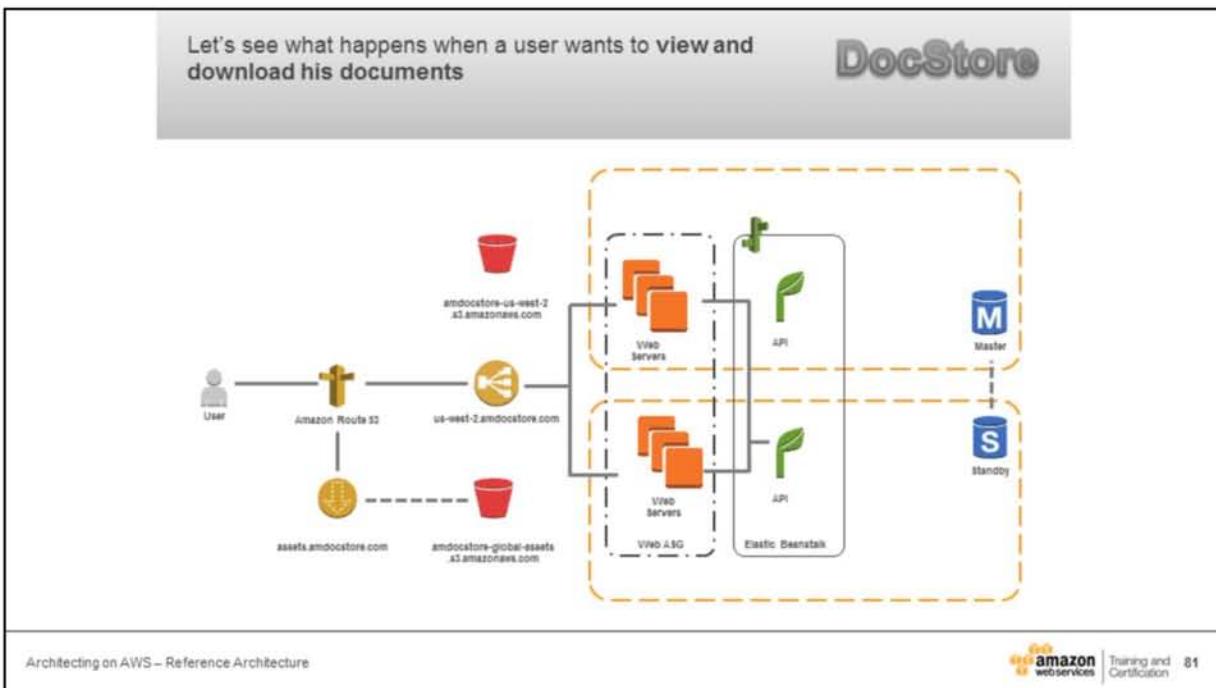


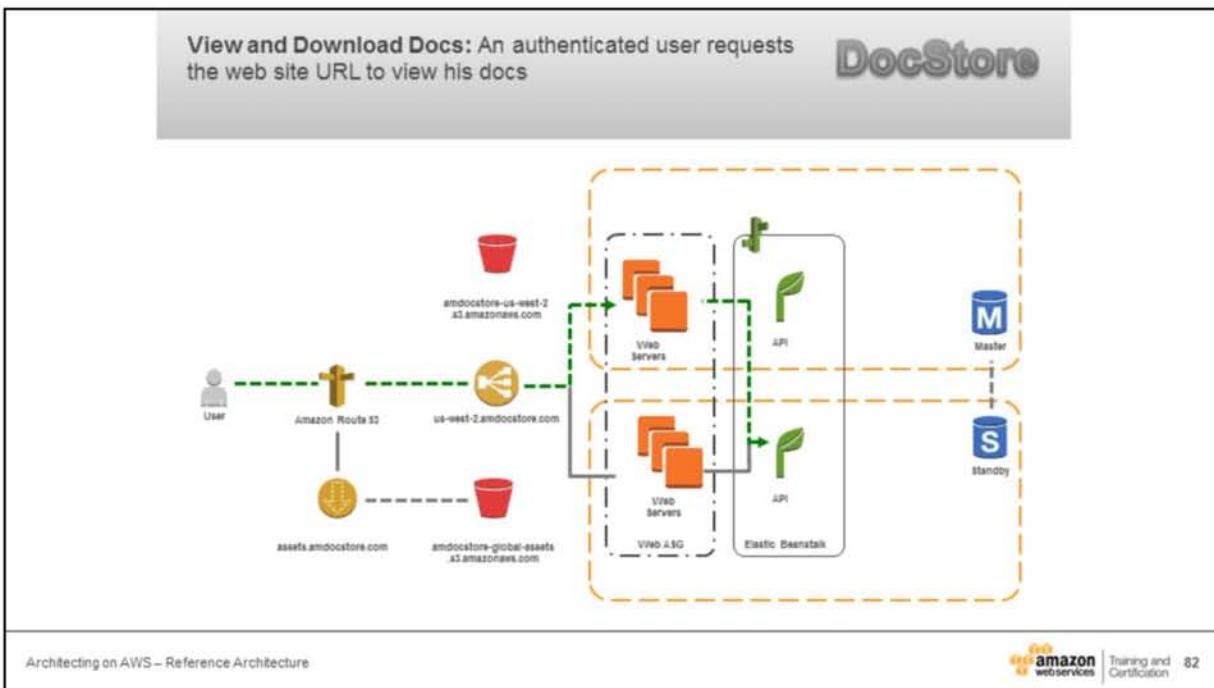


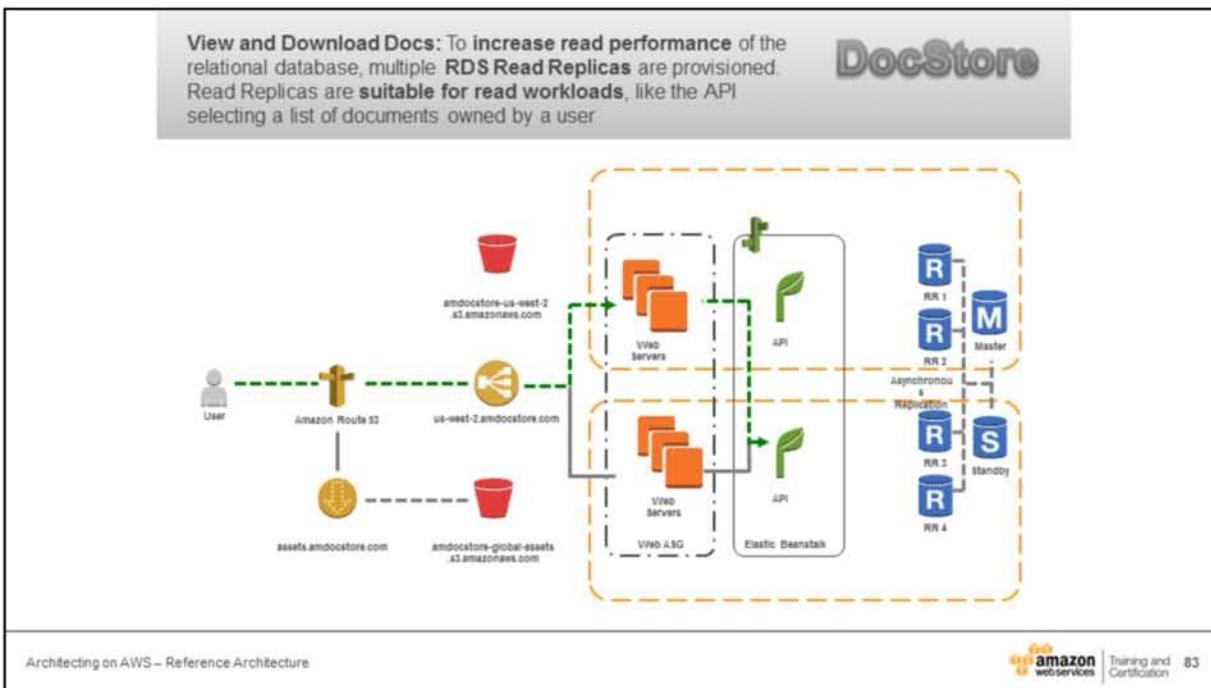


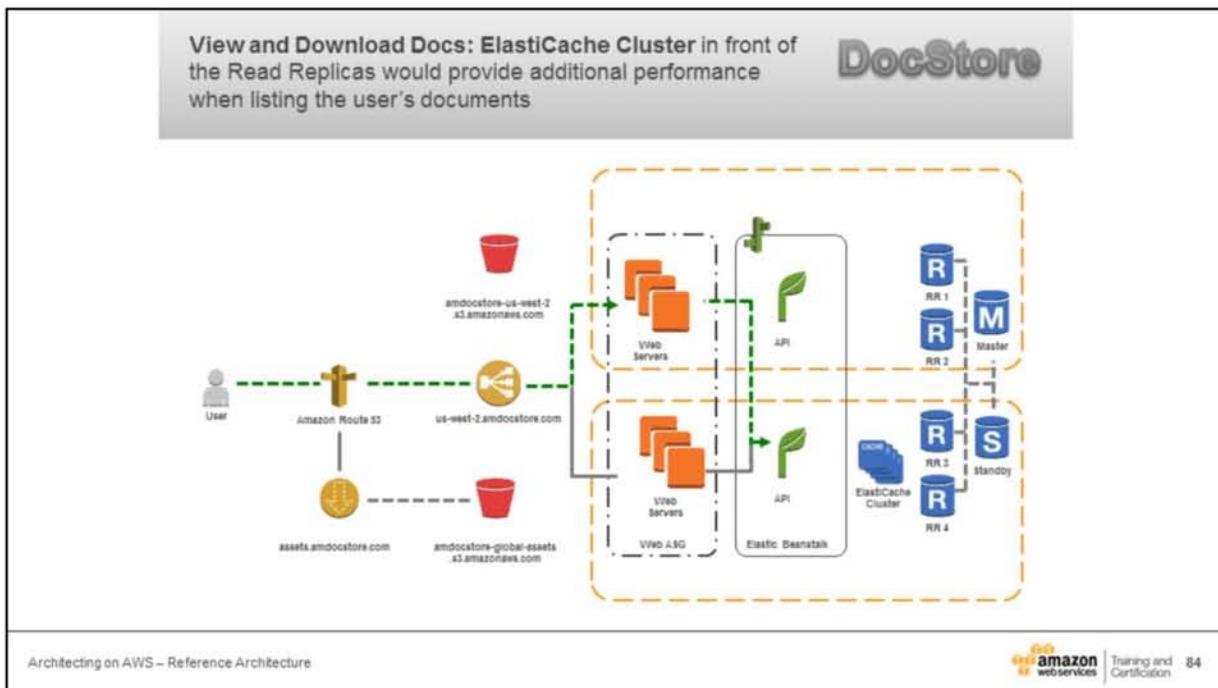


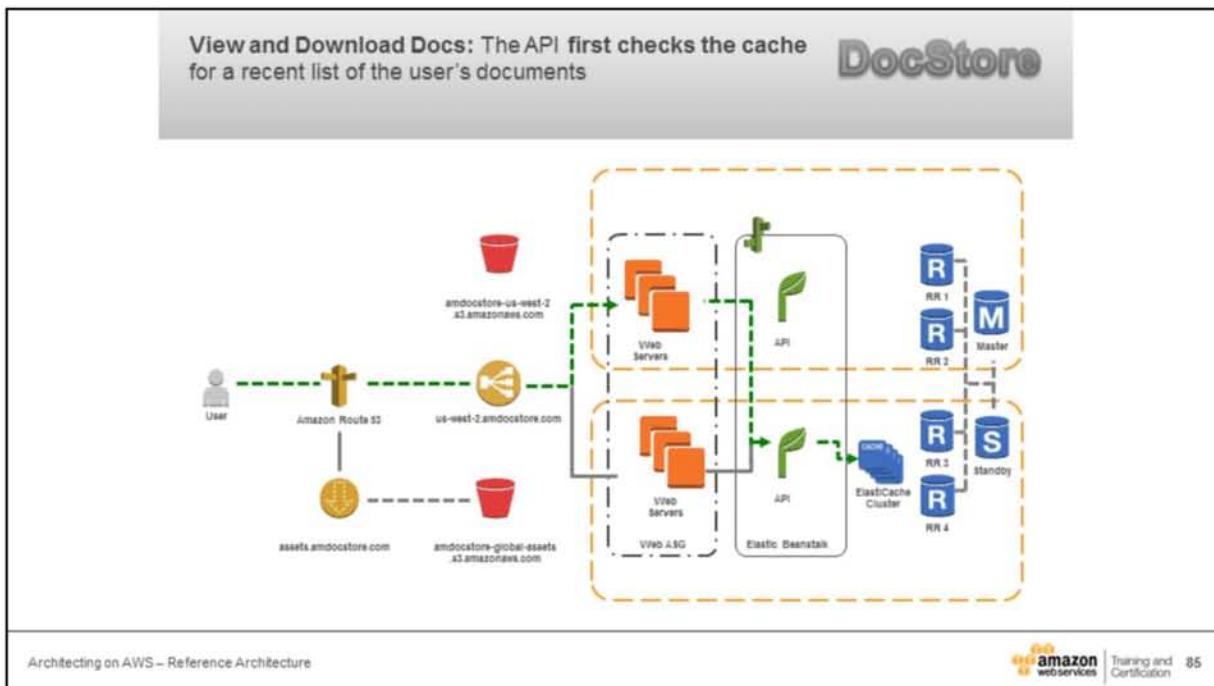


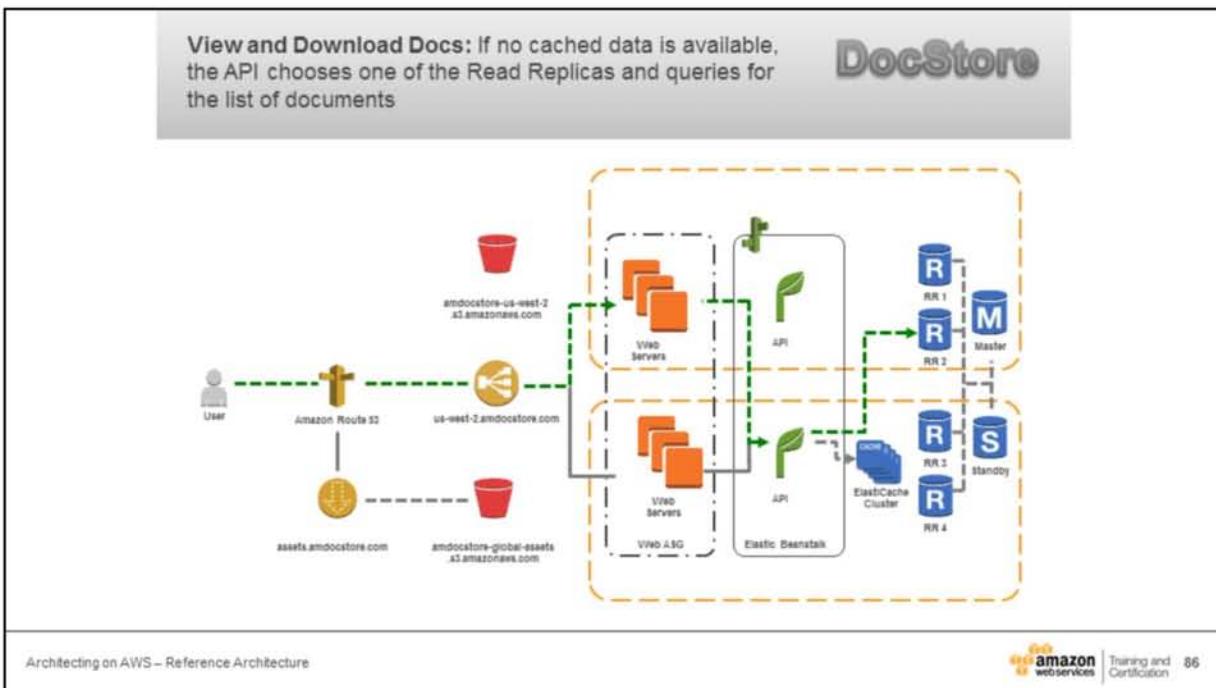


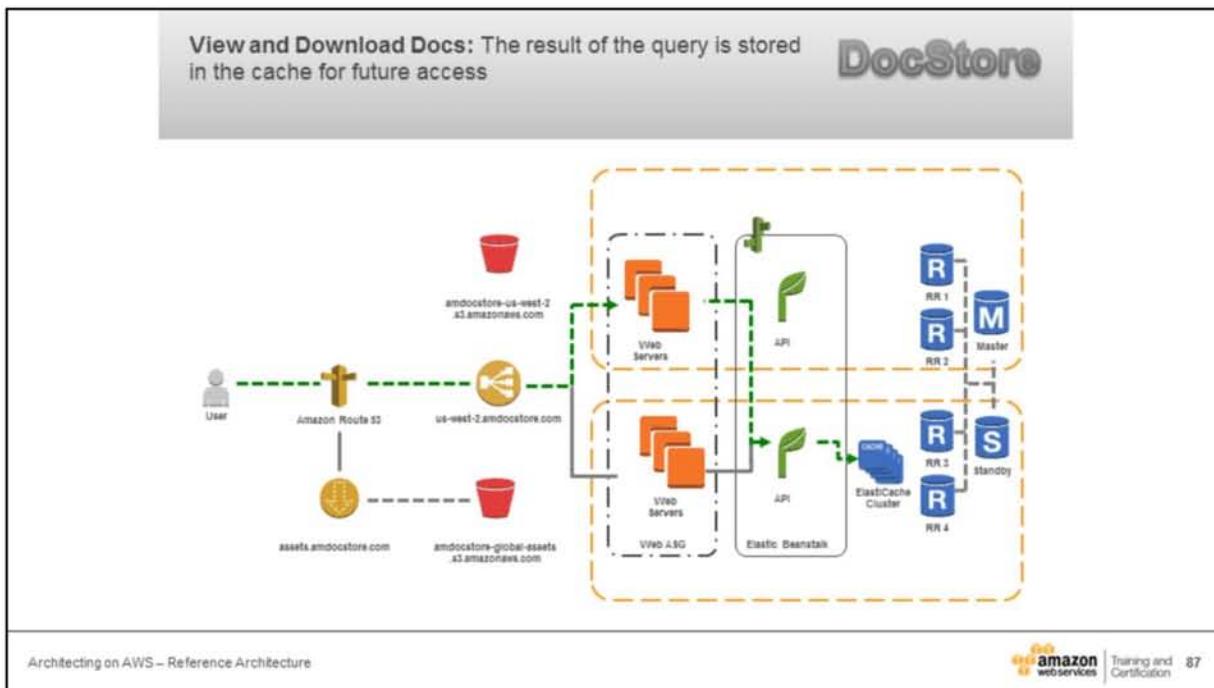


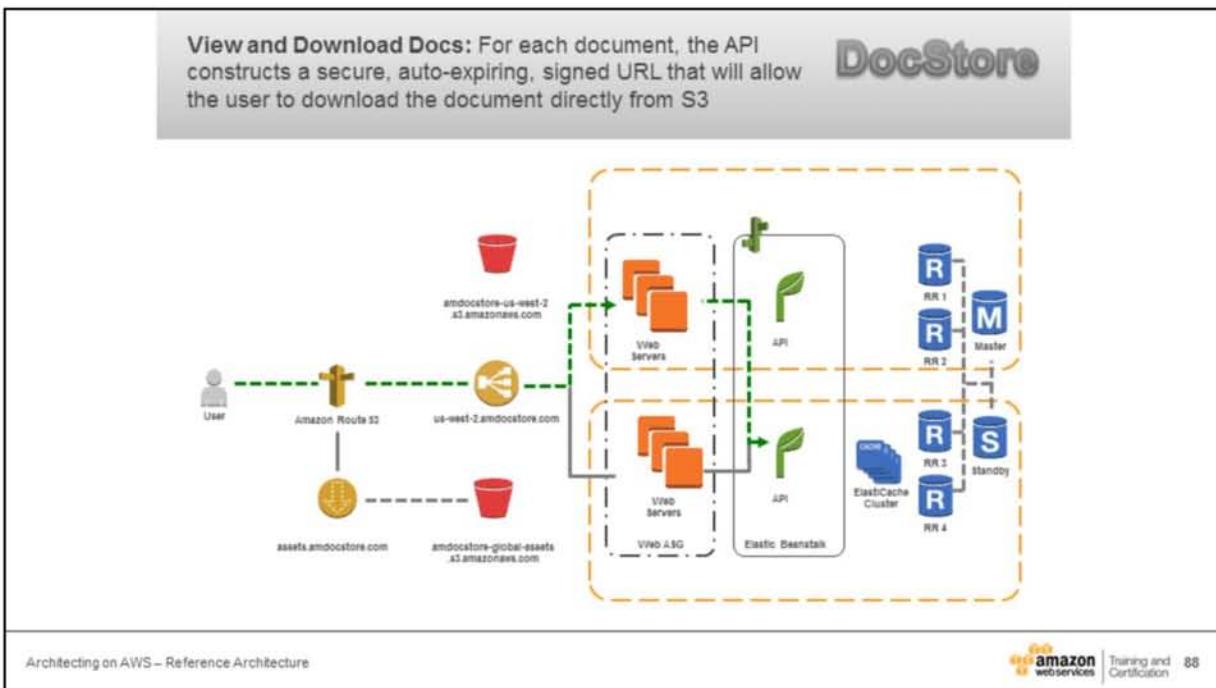


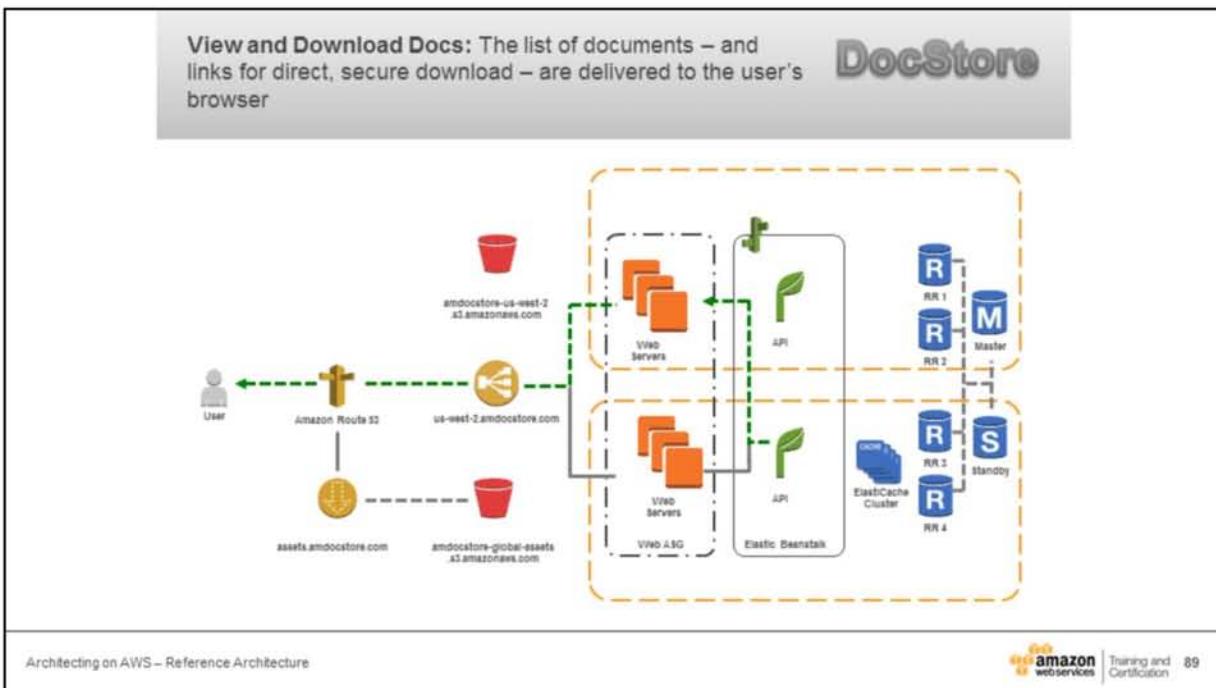


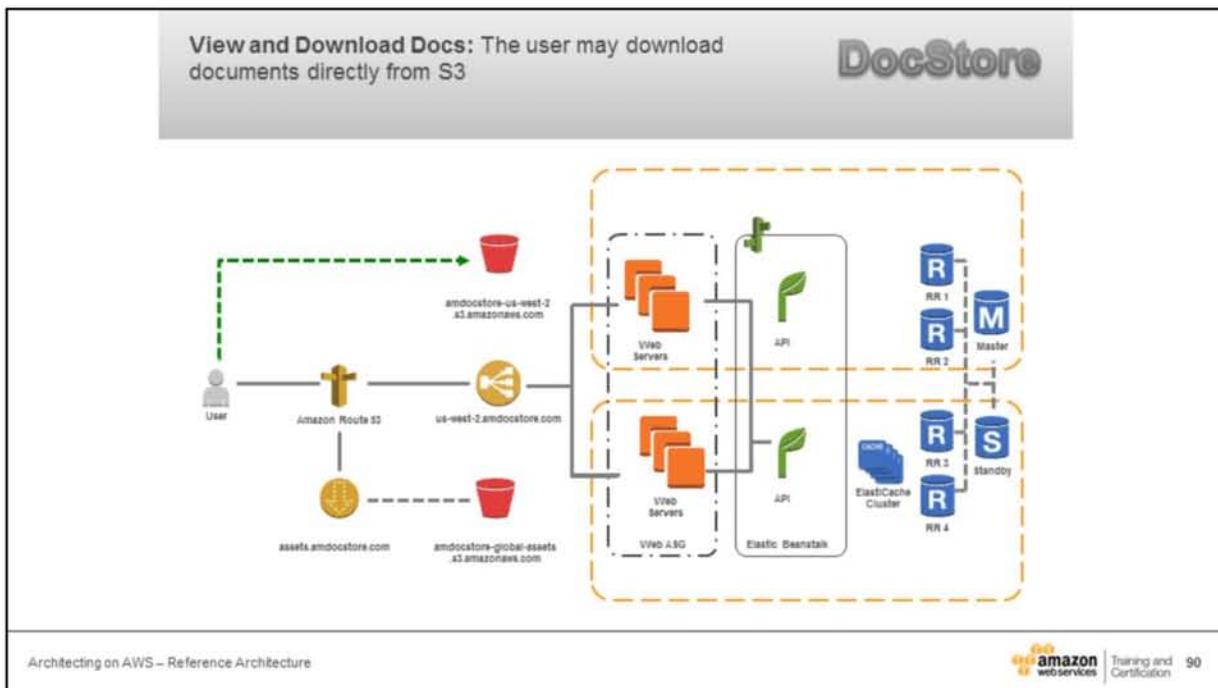


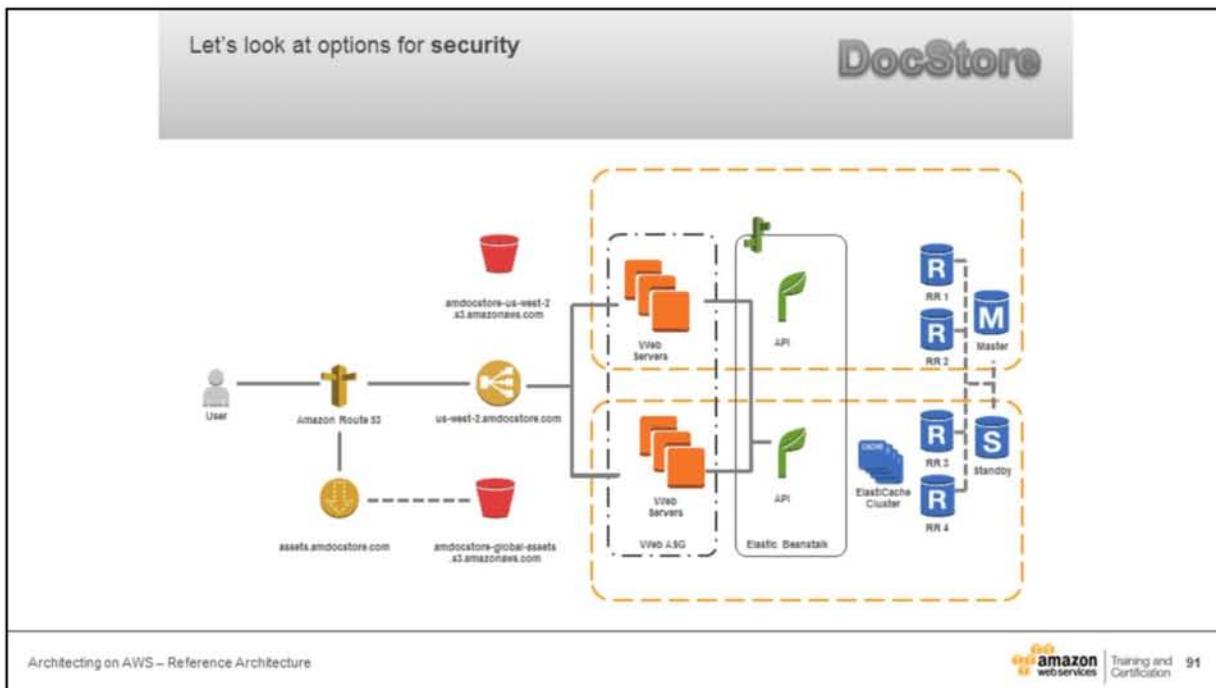


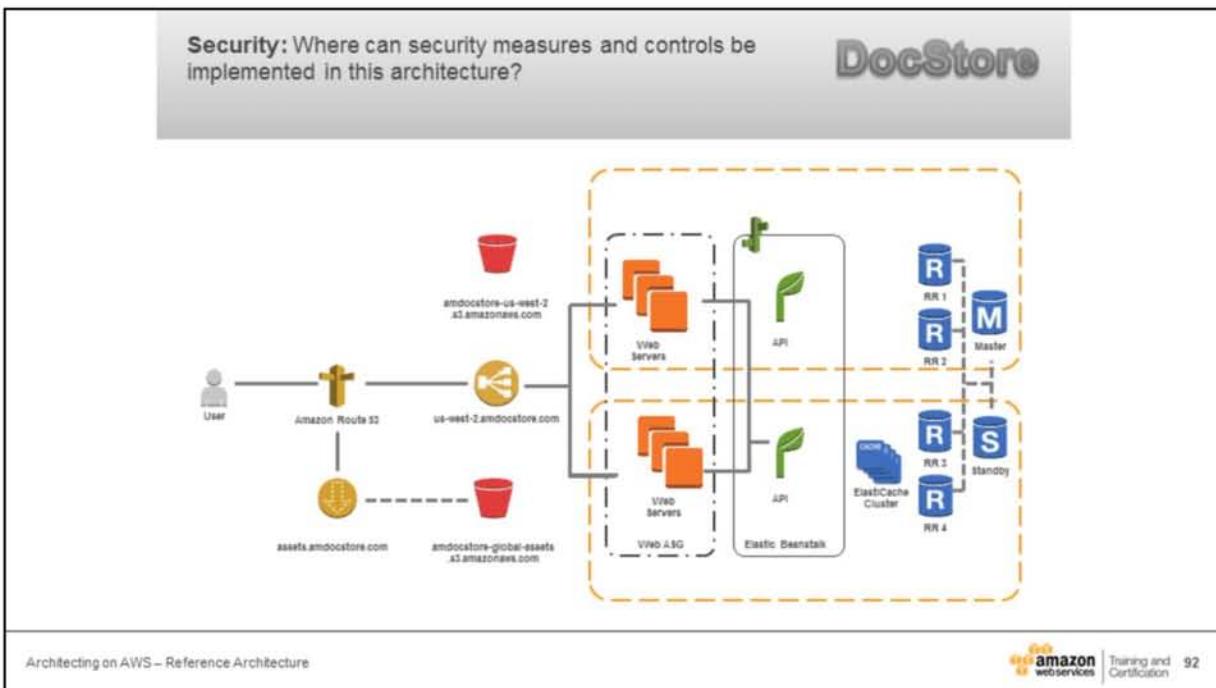


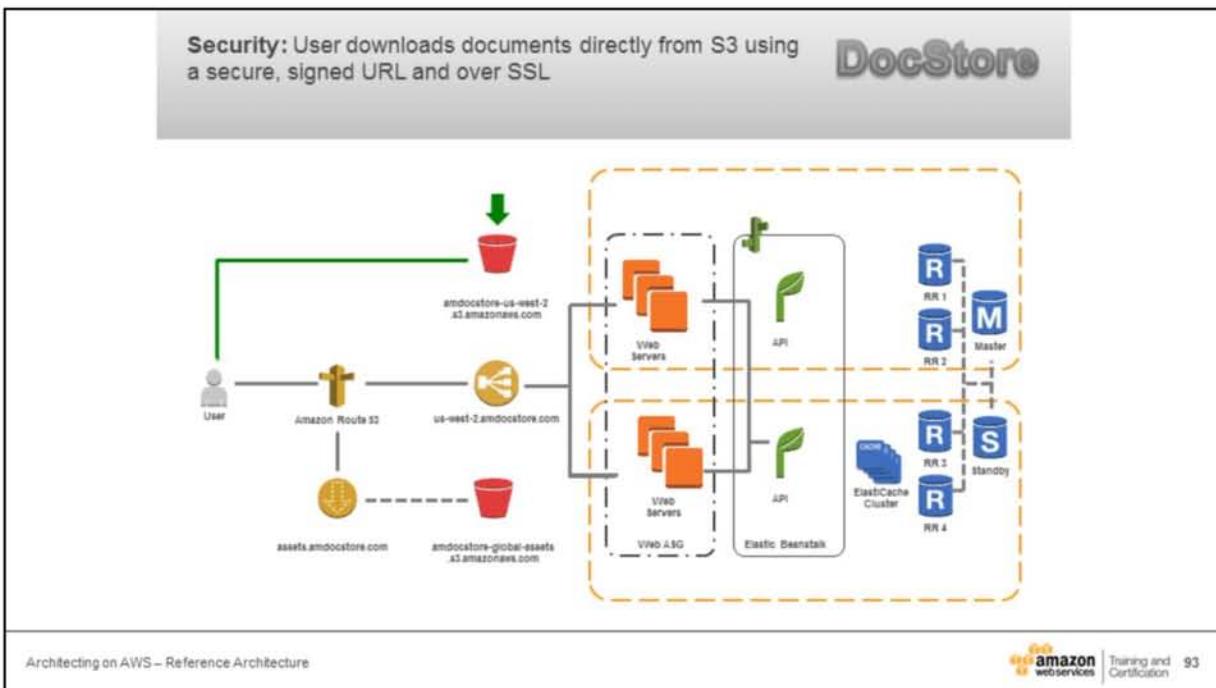


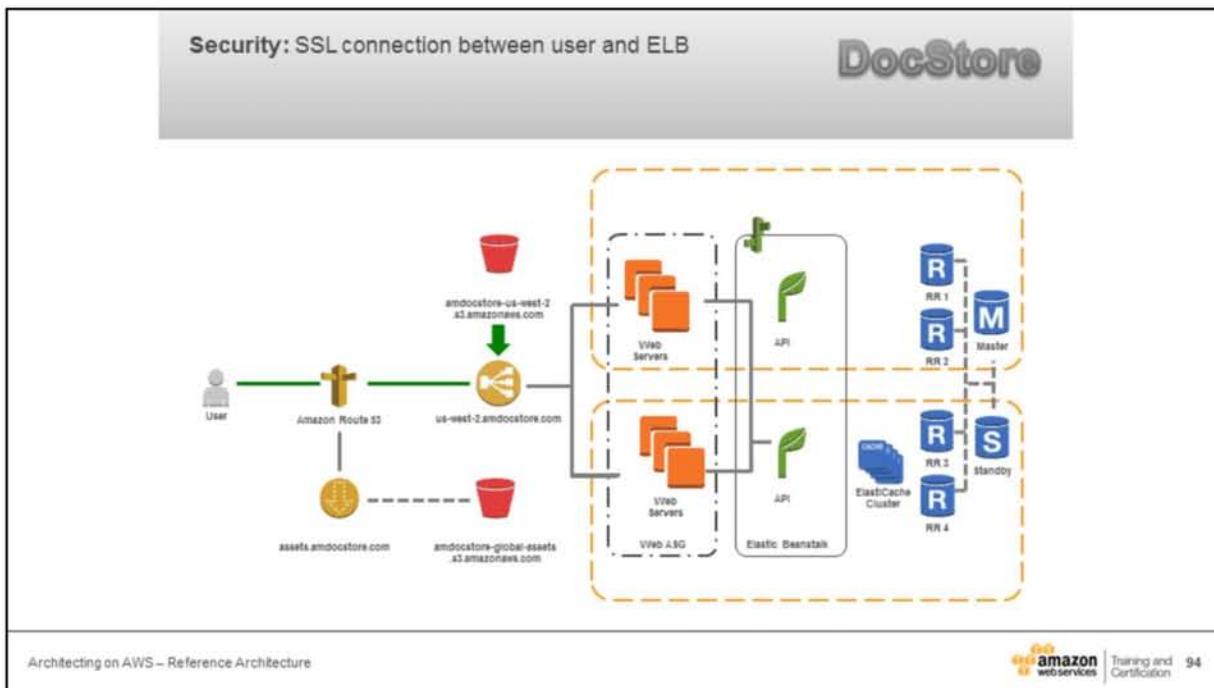


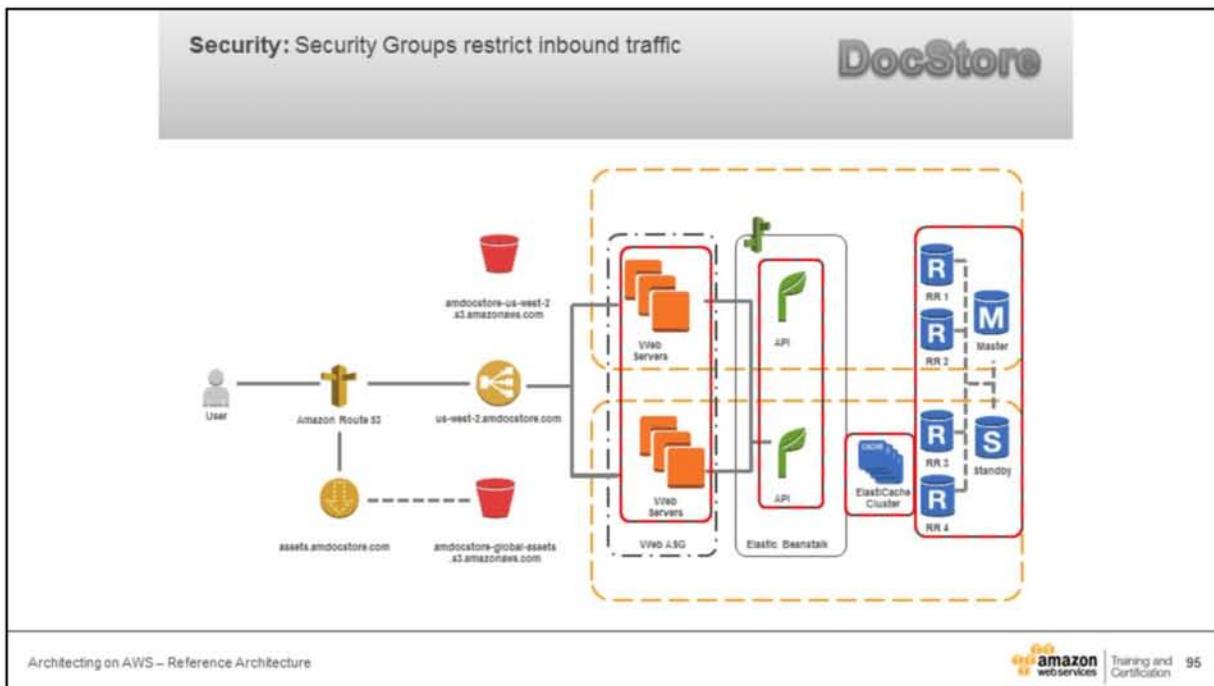


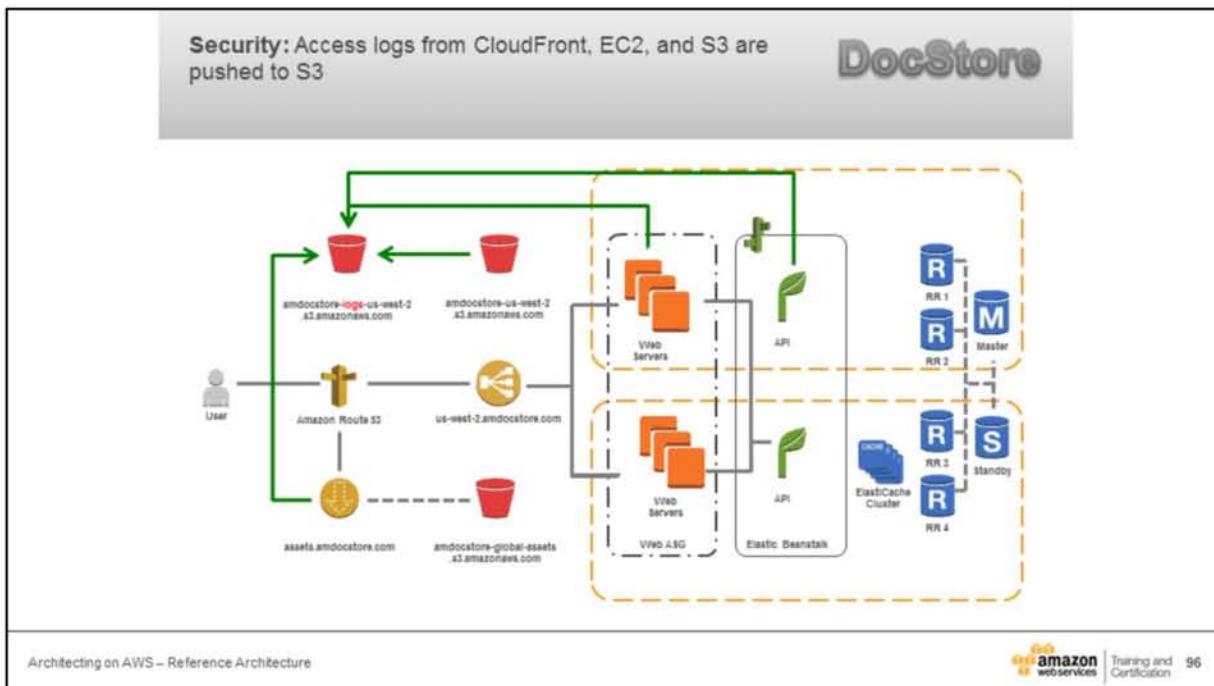


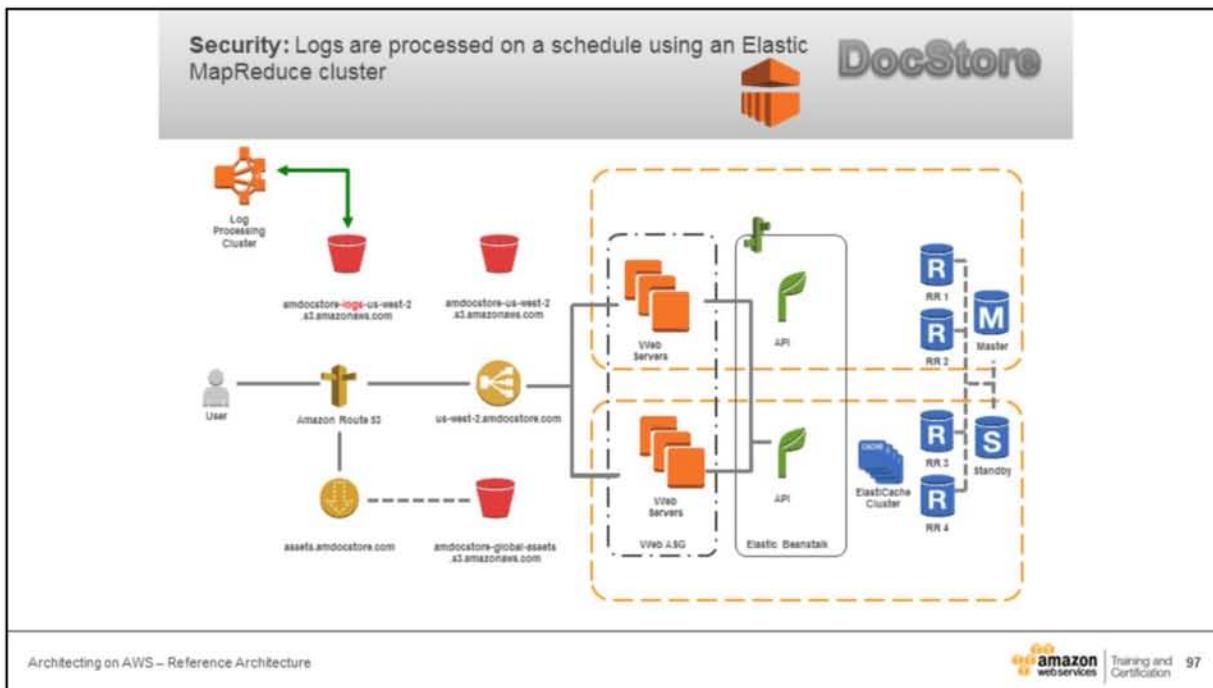


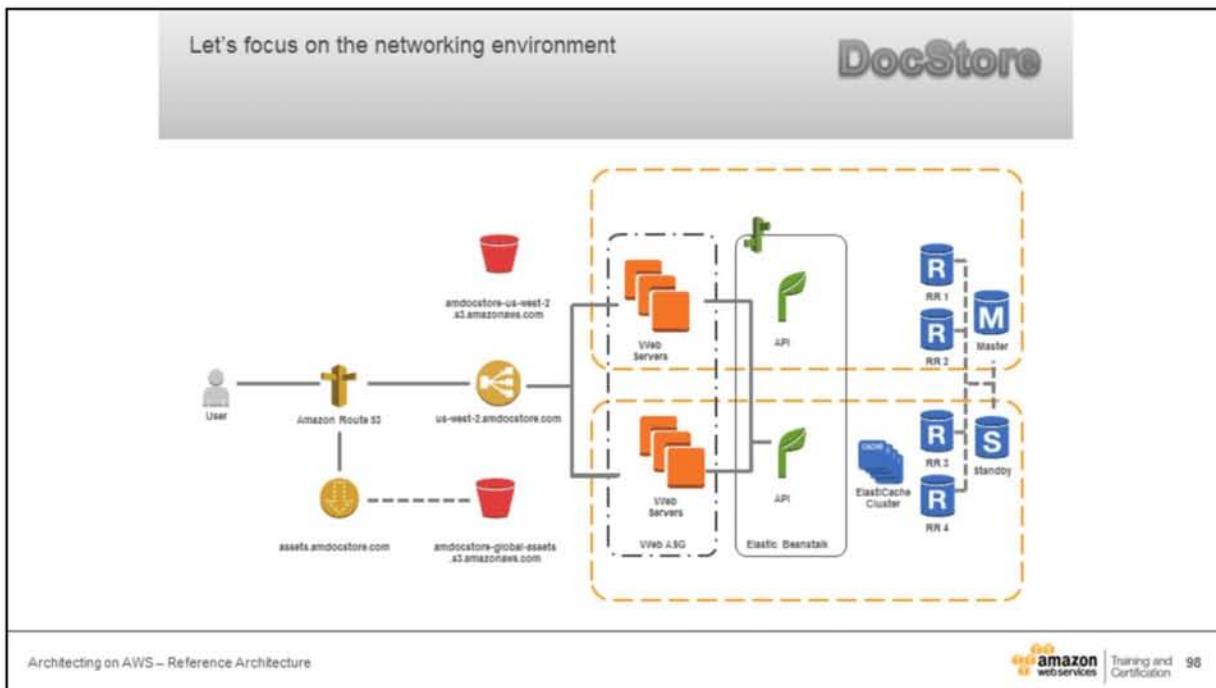


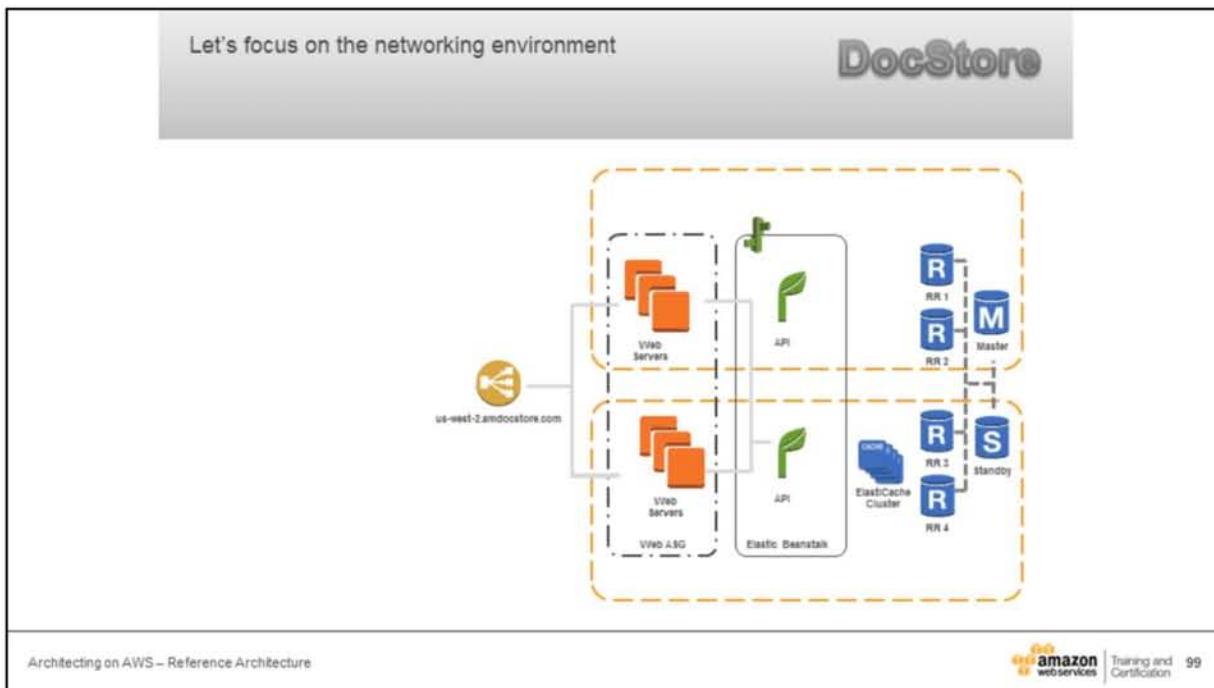


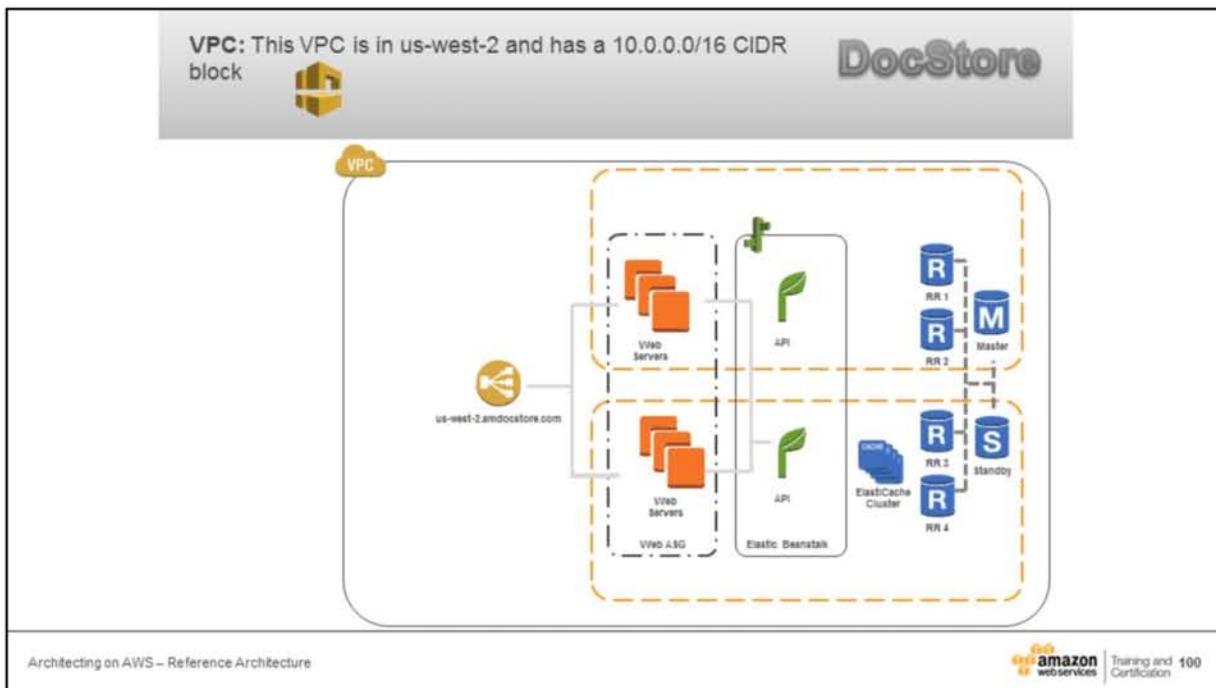


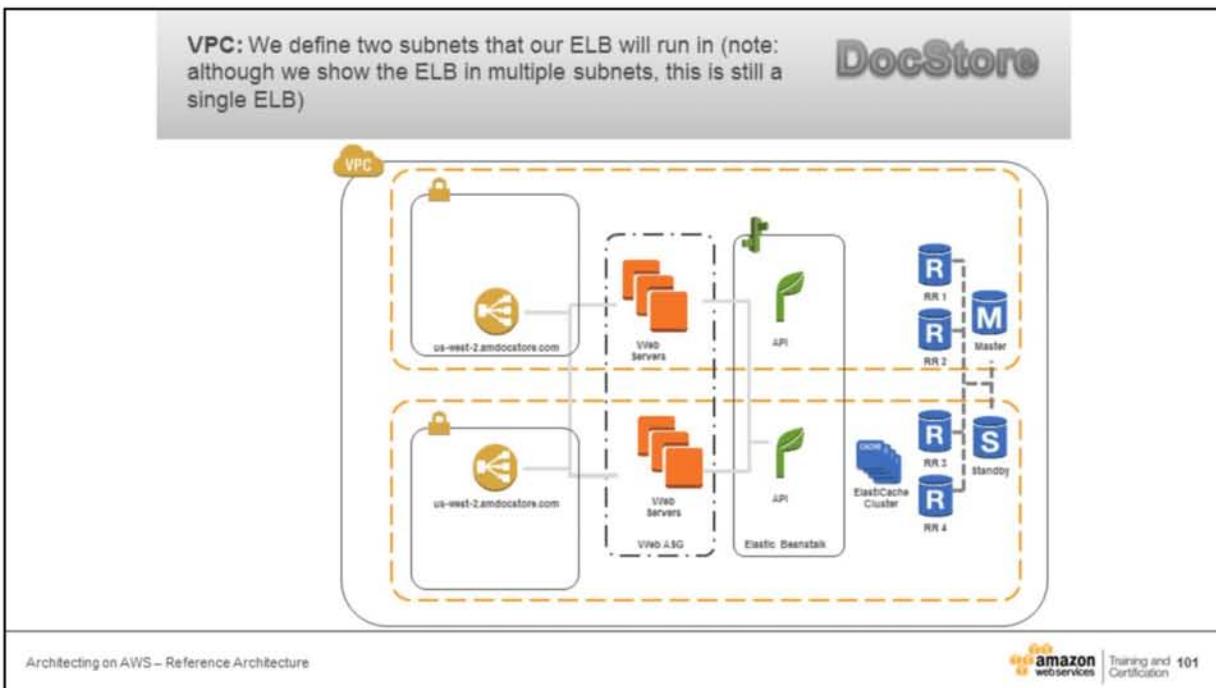


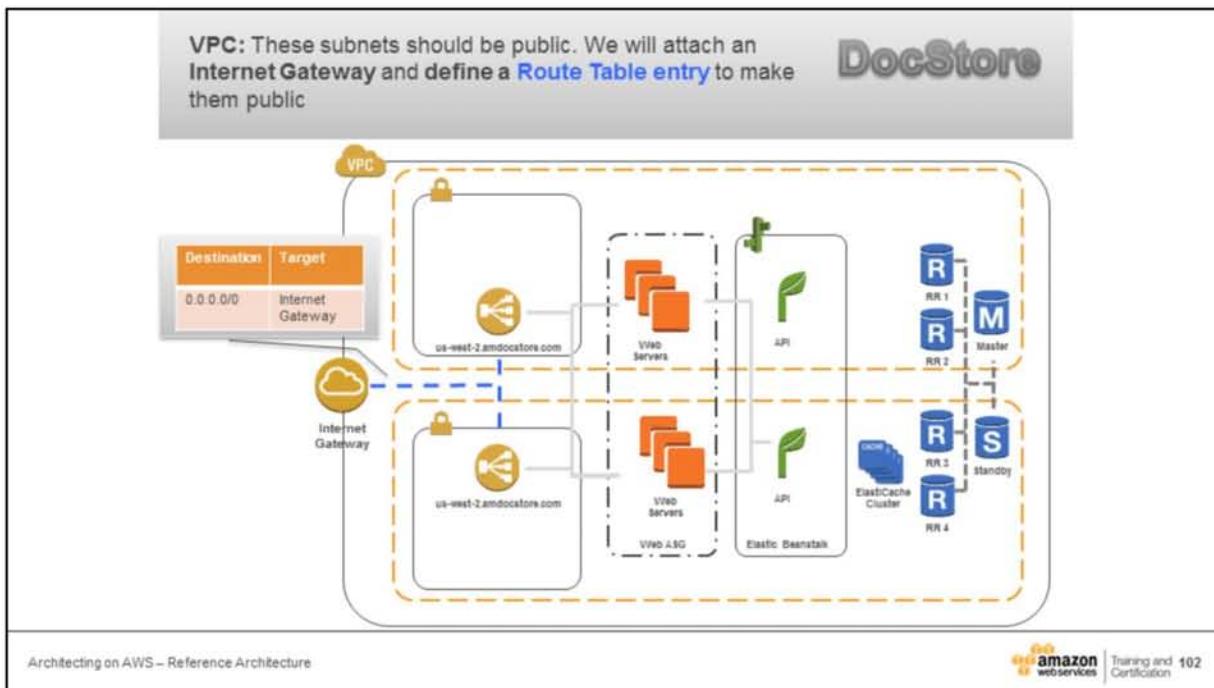


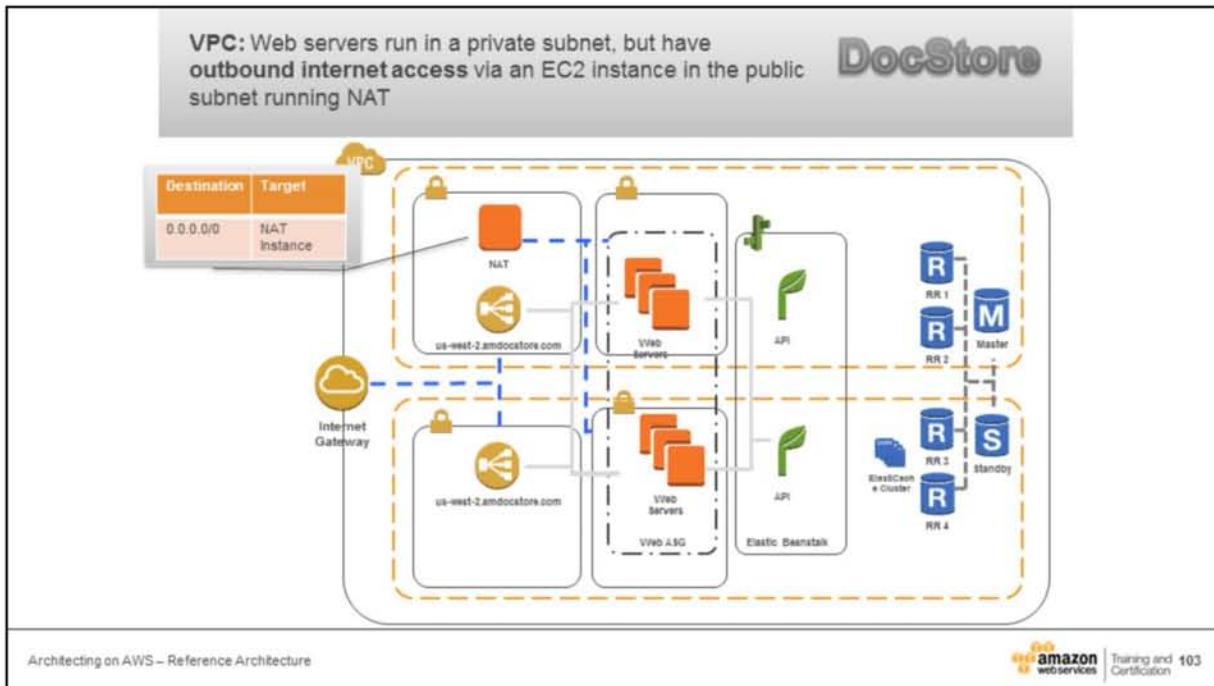


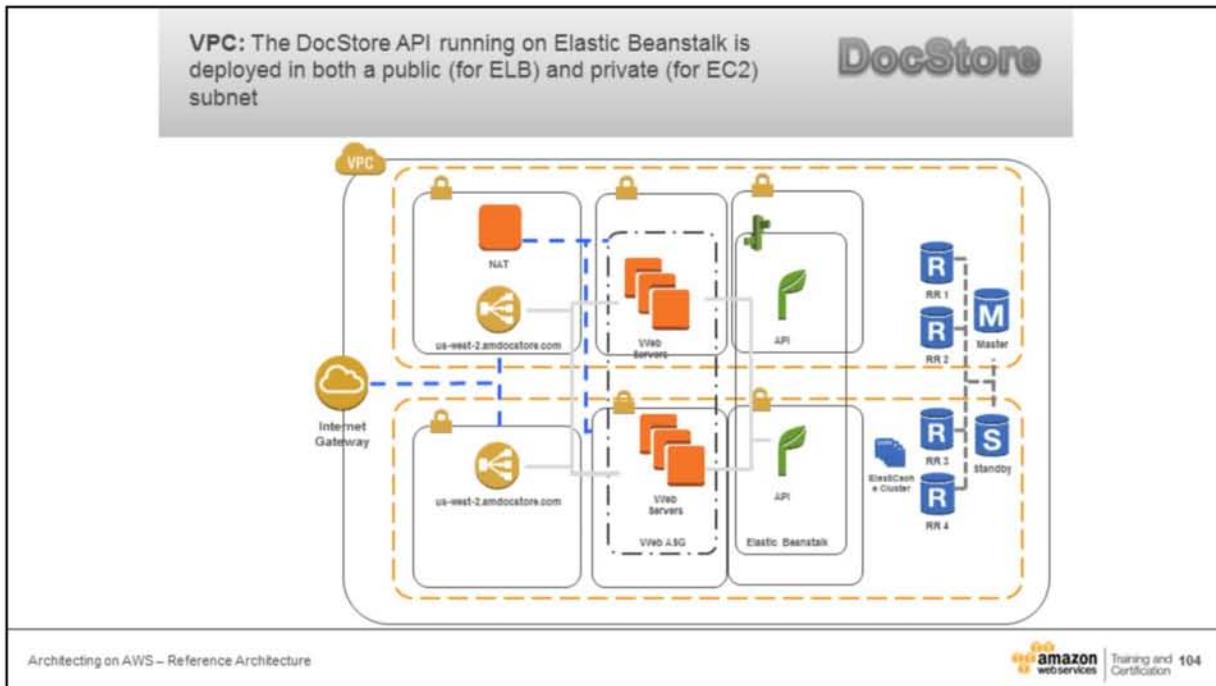


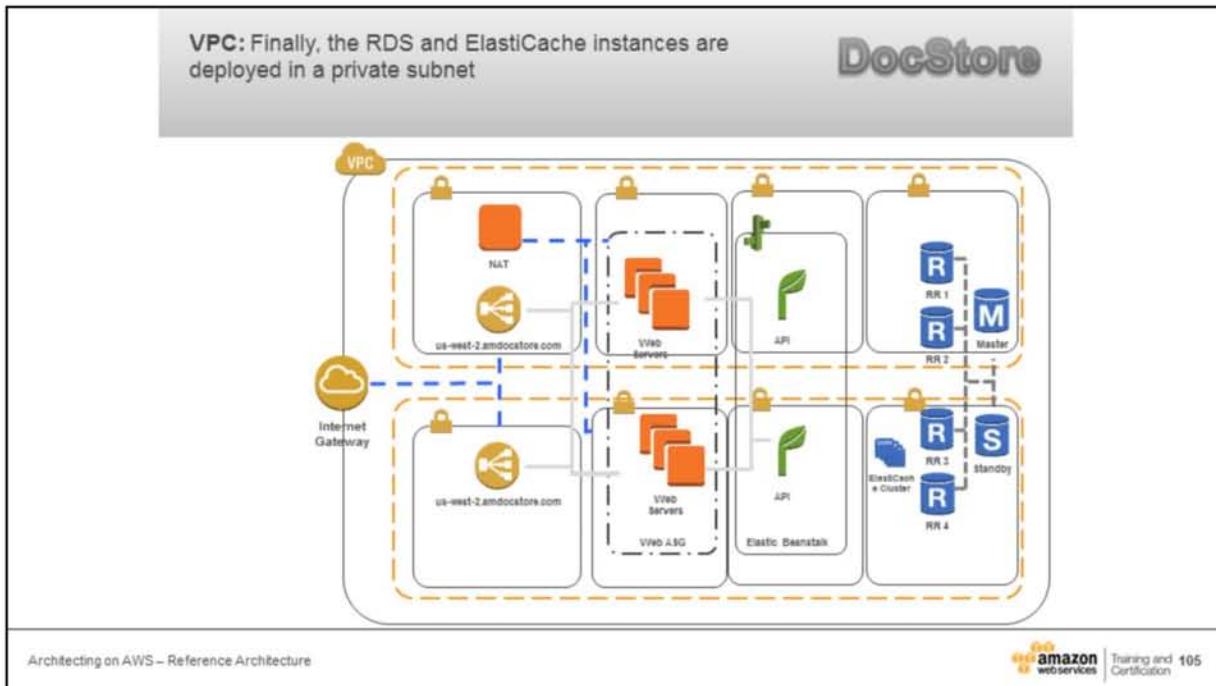










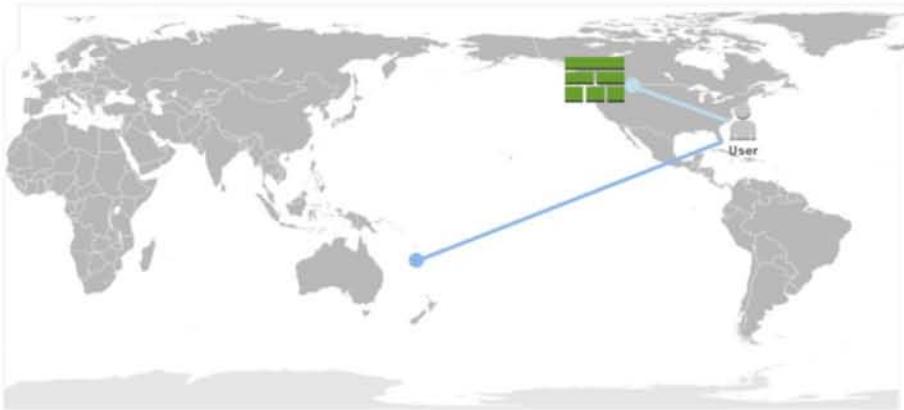


CloudFormation: Entire application is deployed as a stack from a template file



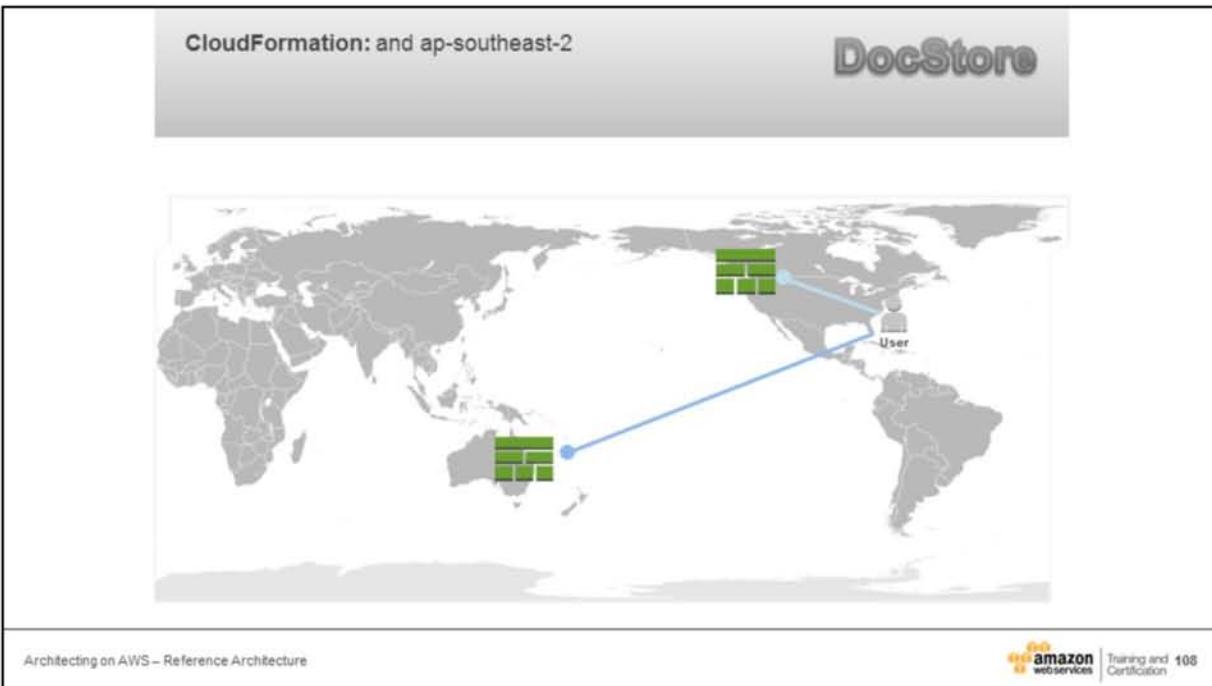
CloudFormation: The sample template file is used to deploy to us-west-2

DocStore



Architecting on AWS – Reference Architecture

amazon web services | Training and Certification 107



DocStore

#2: Batch Processing Back-End

Architecting on AWS – Reference Architecture



Training and 109
Certification

DocStore

#2: Batch Processing Back-End

- Sync user accounts
- Enforce max storage limits
- Extract and index text based on document type (i.e., PDF uses different filter than DOCX, etc.)
- Detailed dashboard with system-wide totals:
 - # docs, avg. doc size, total storage

DocStore

#2: Batch Processing Back-End

- Responds to events triggered by Web interface:

DocStore

#2: Batch Processing Back-End

- Responds to events triggered by Web interface:

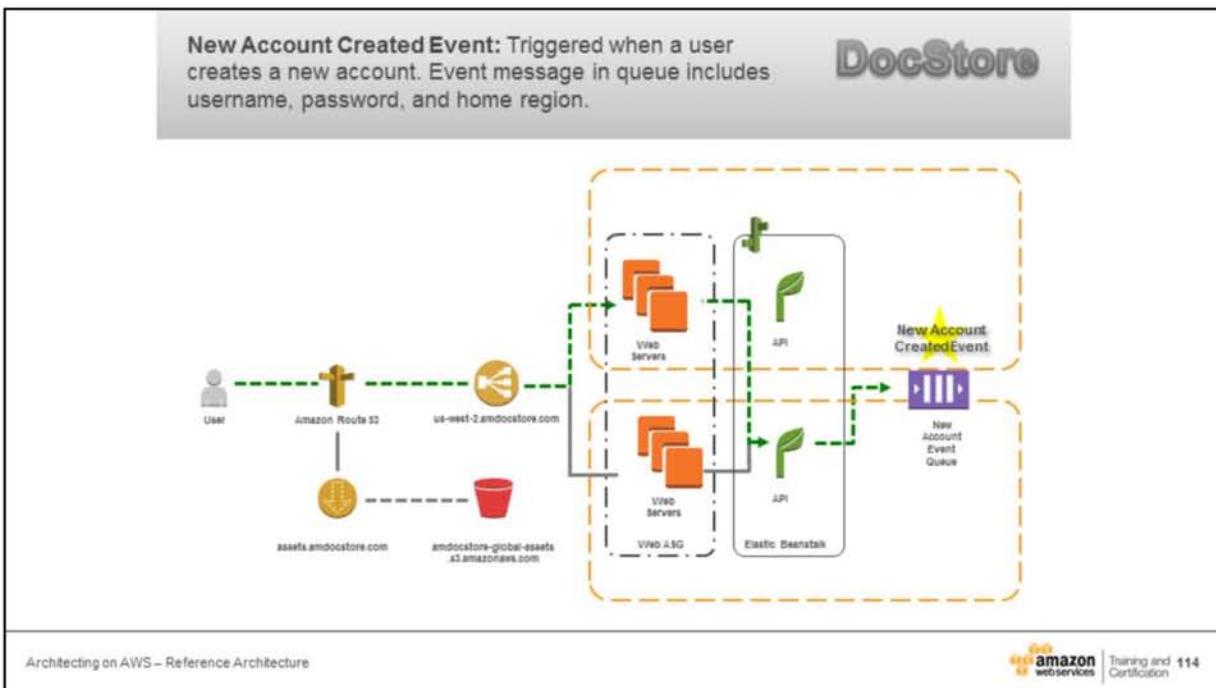


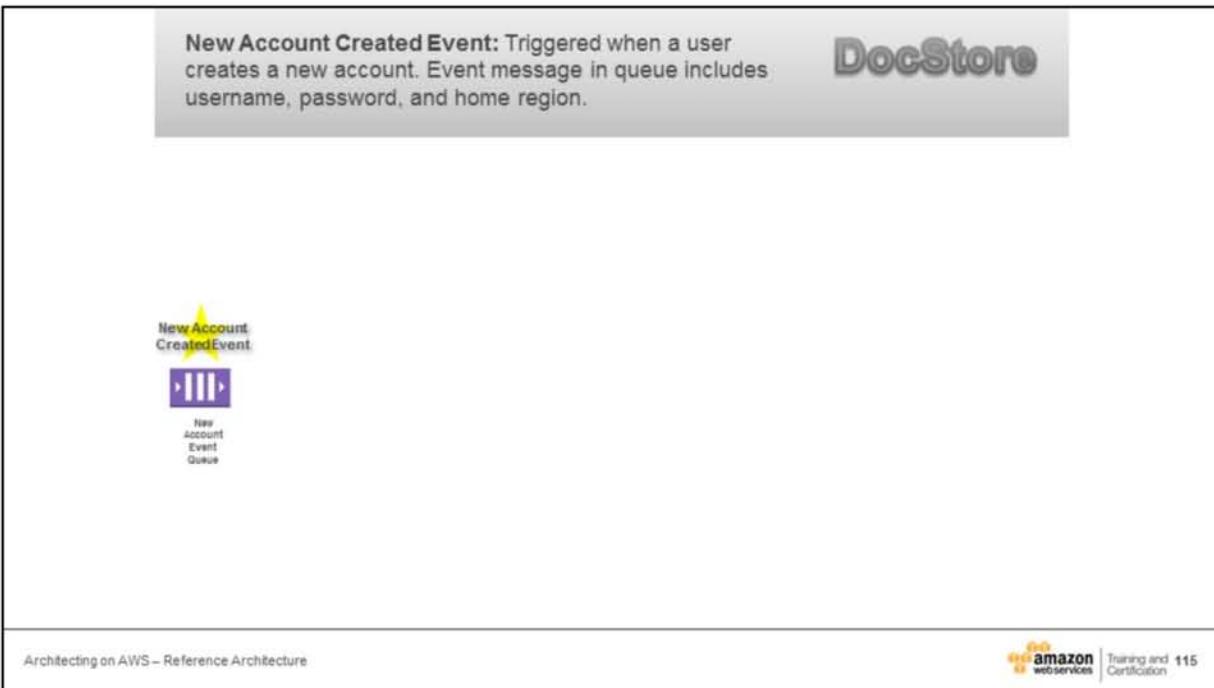
DocStore

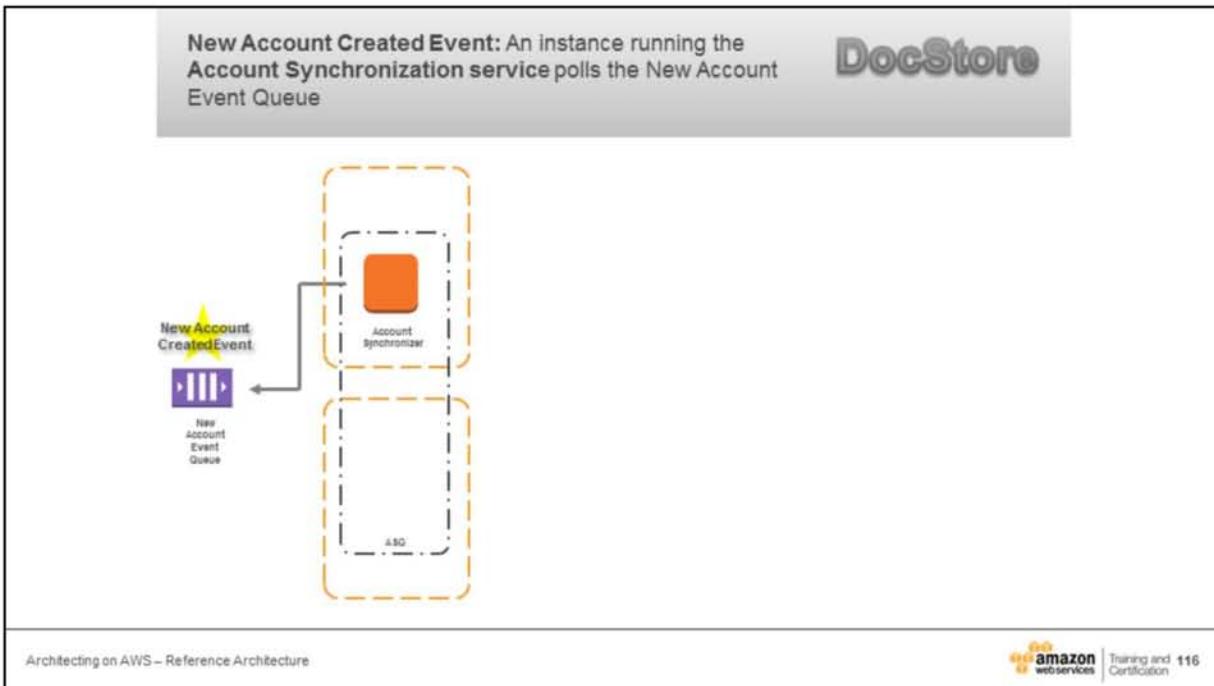
#2: Batch Processing Back-End

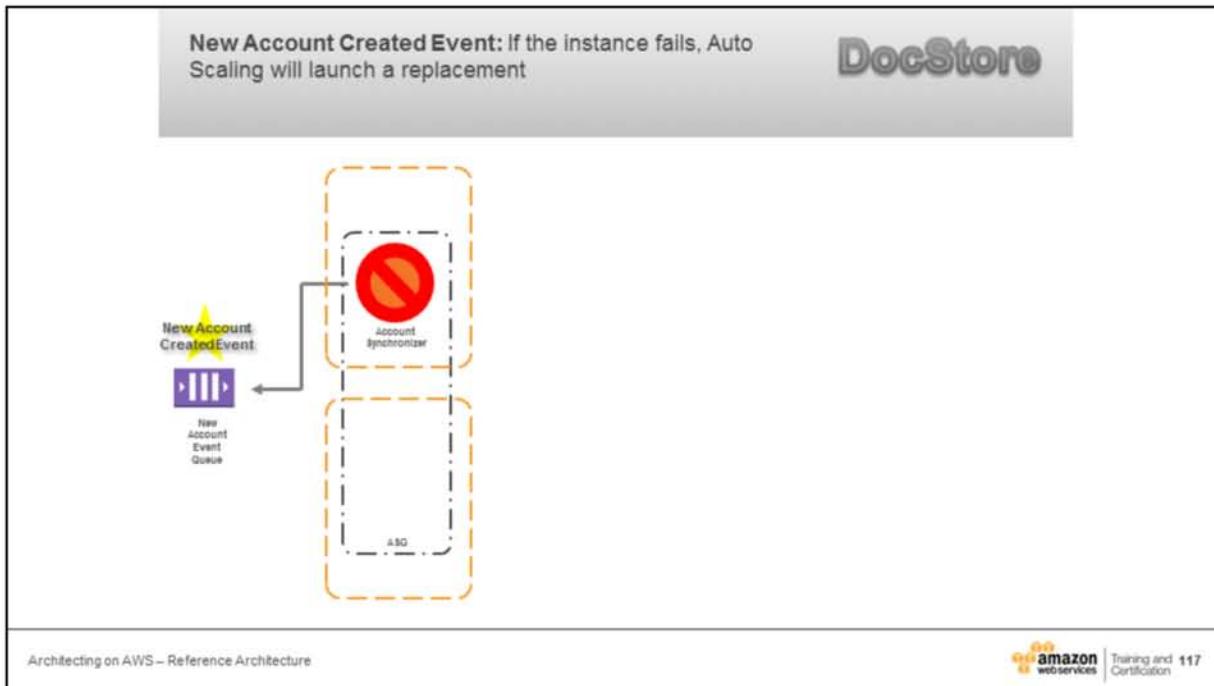
- Responds to events triggered by Web and Mobile interfaces:

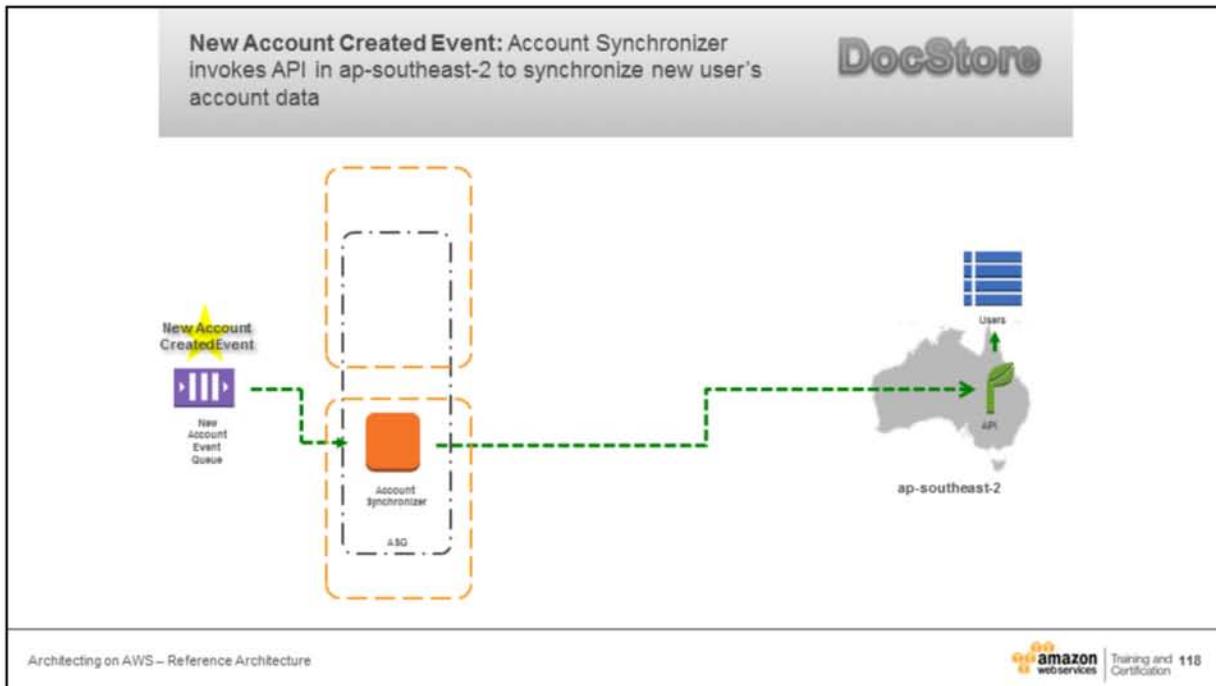








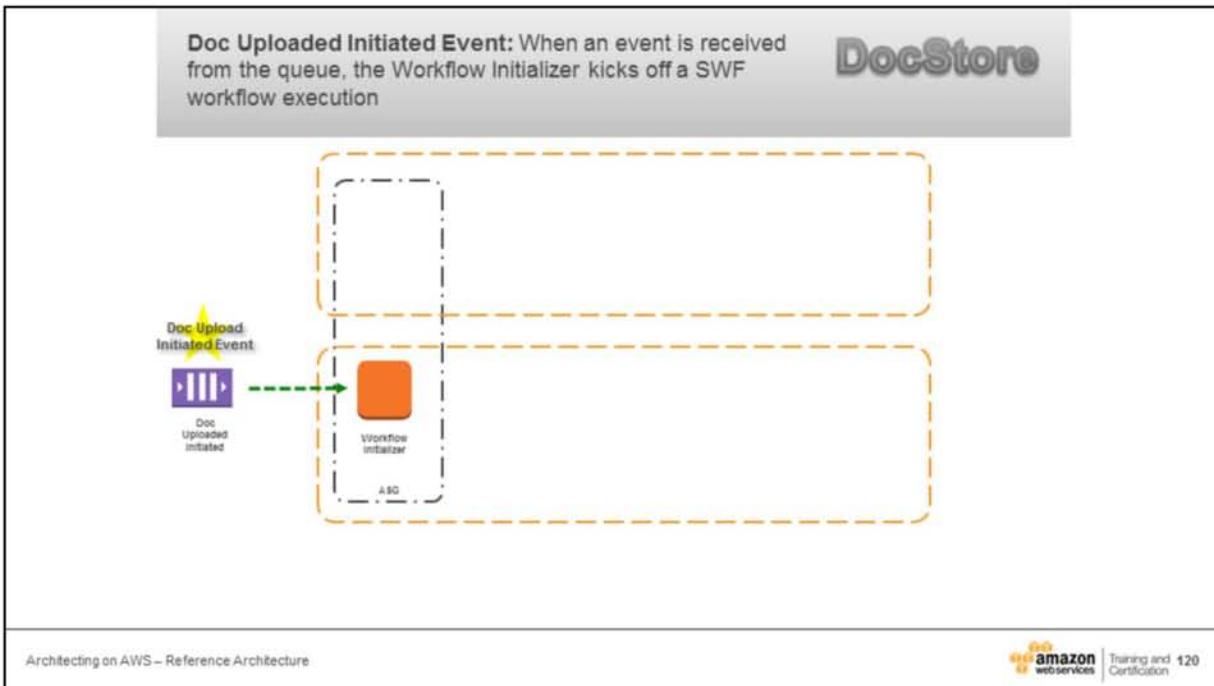


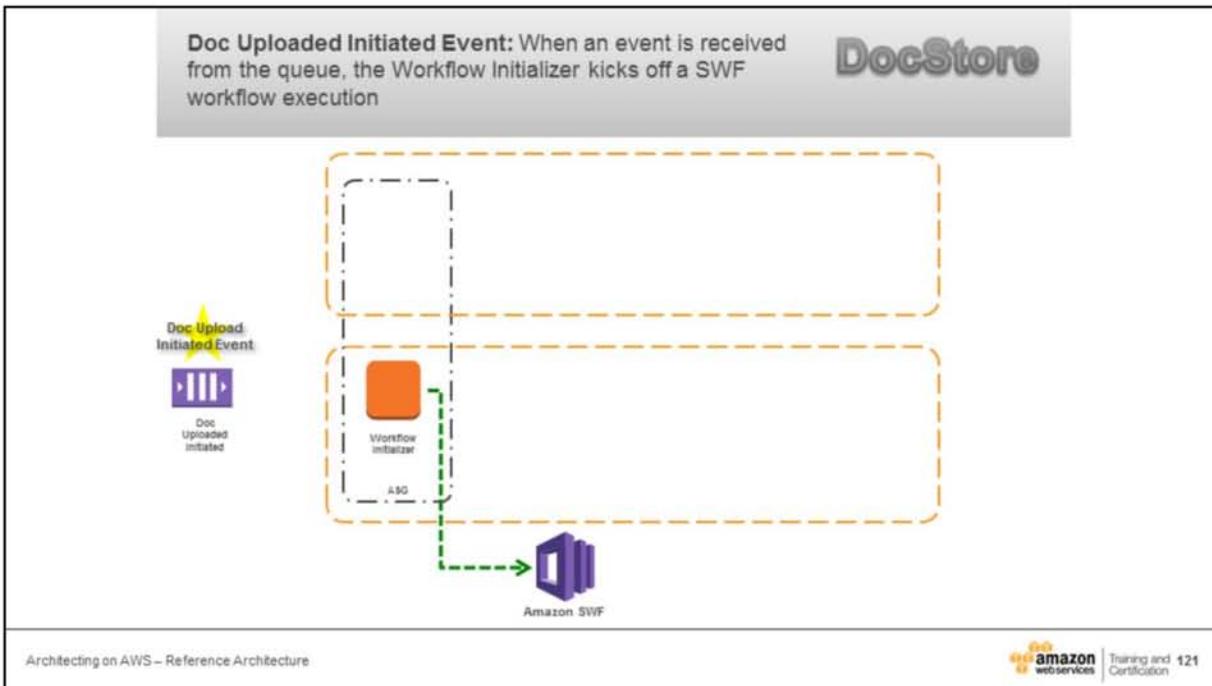


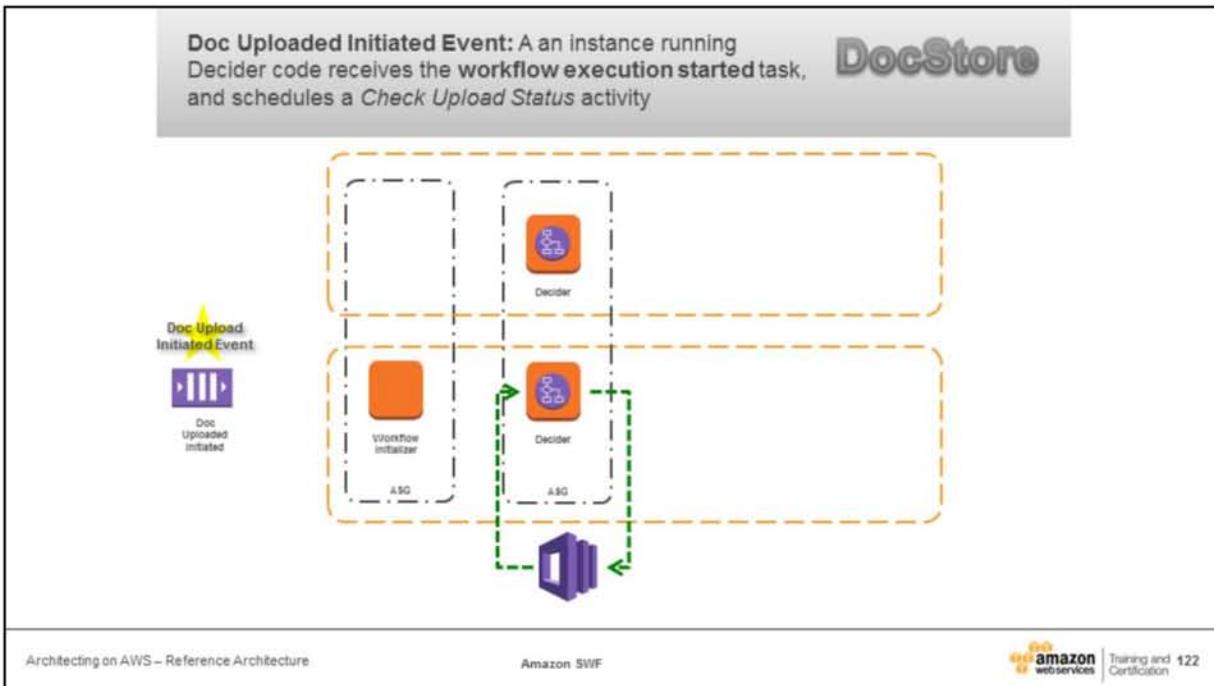
Doc Uploaded Initiated Event: When a user begins uploading a document to S3, an event is triggered so the upload can be tracked via a workflow

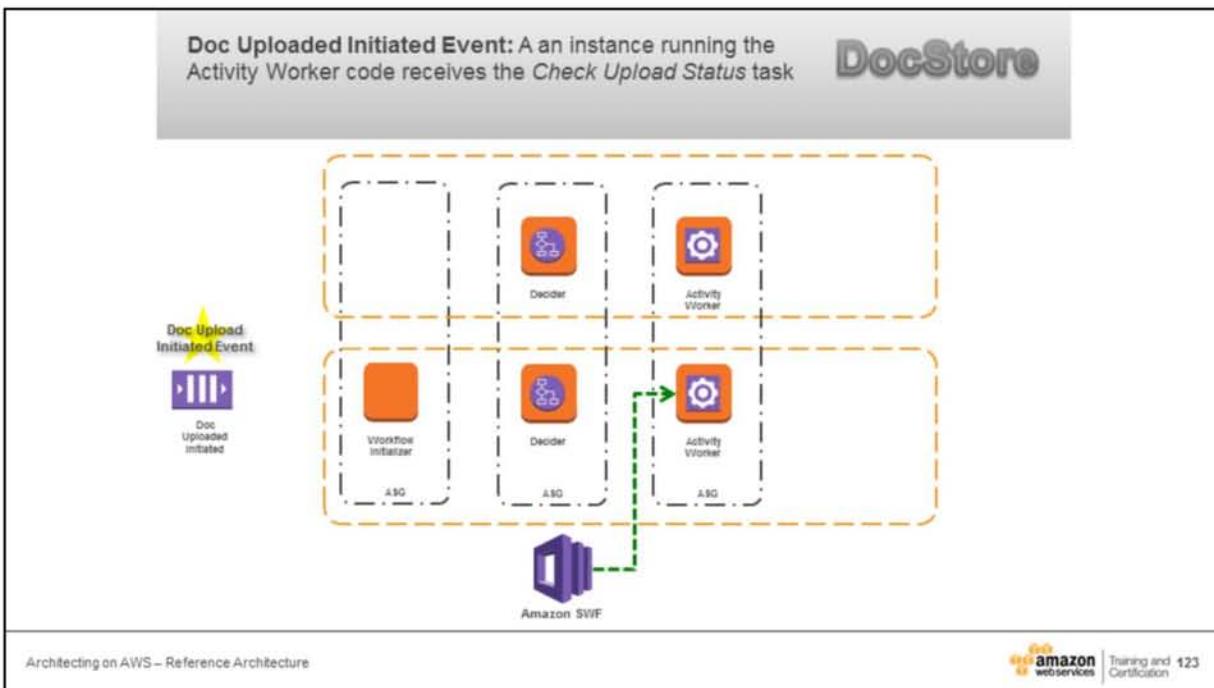
DocStore

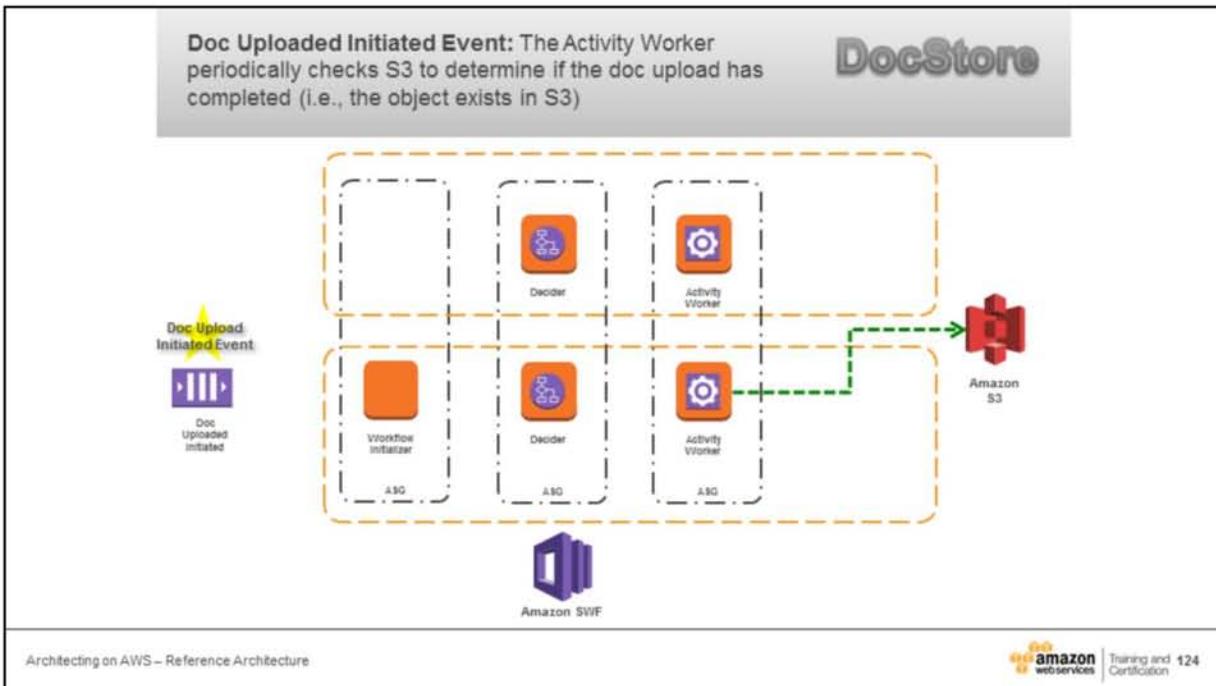


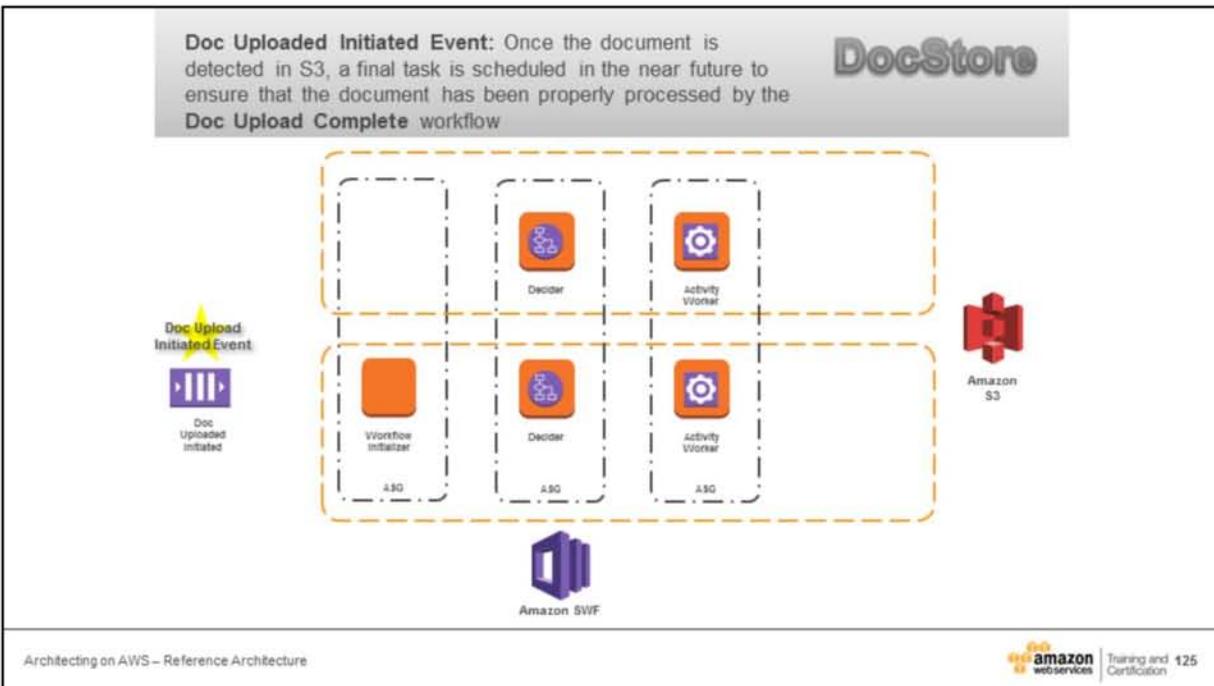






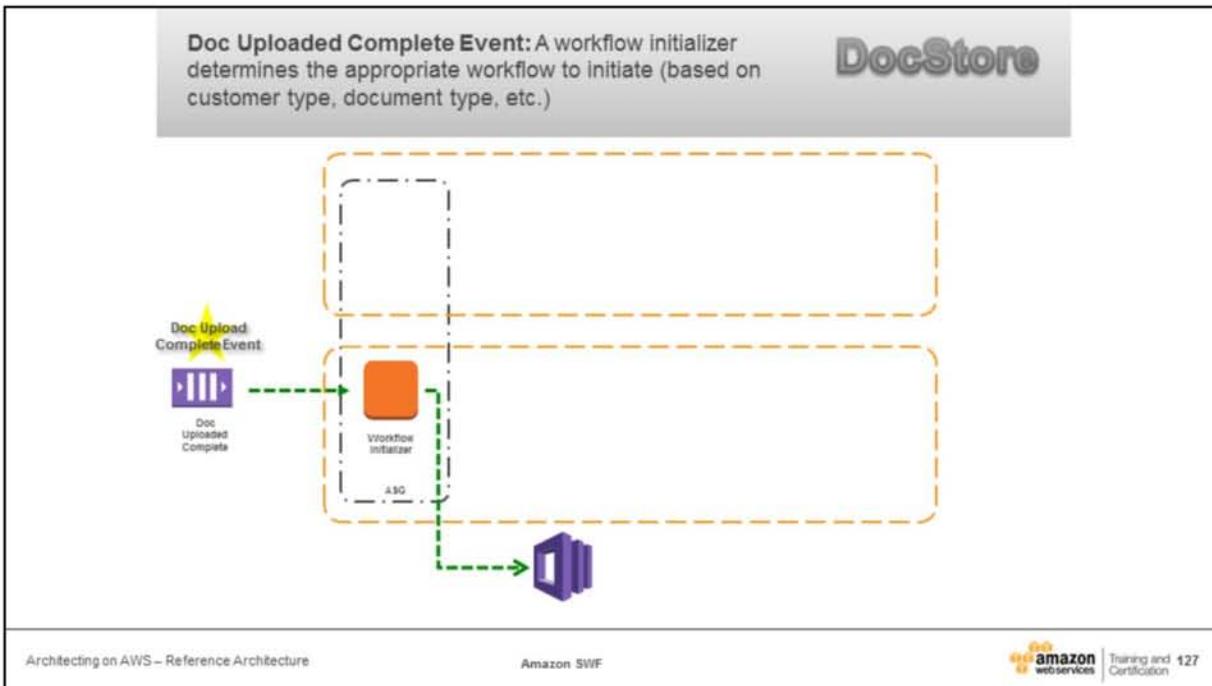


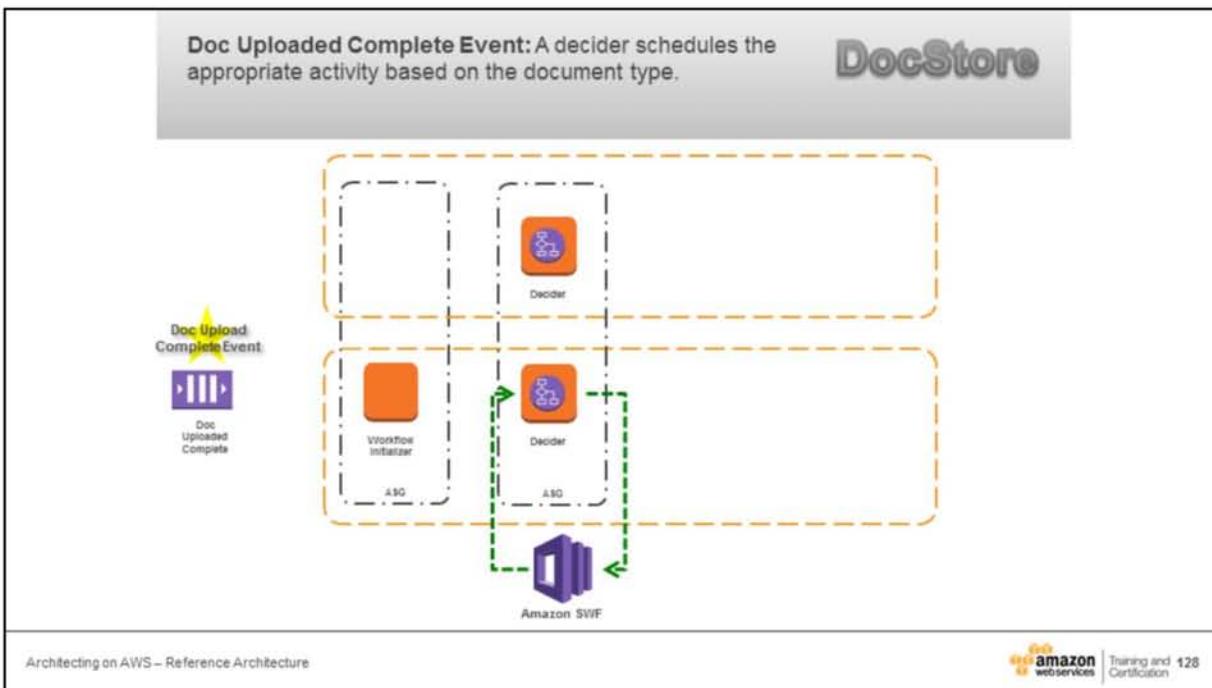


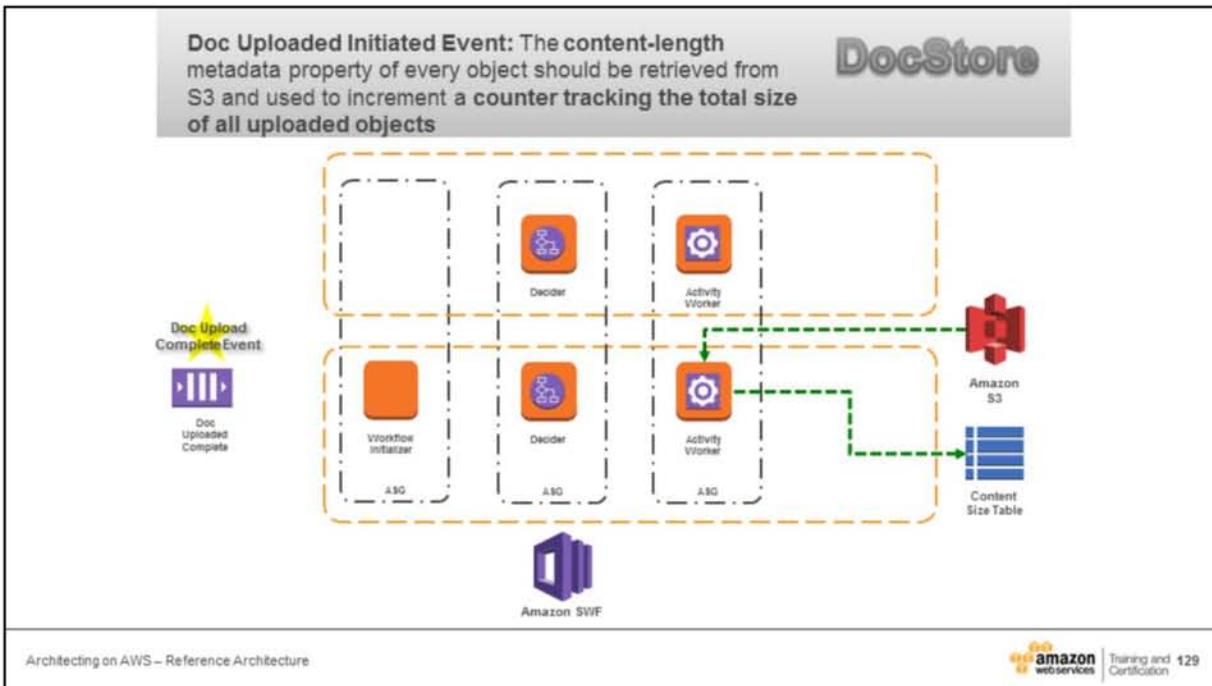


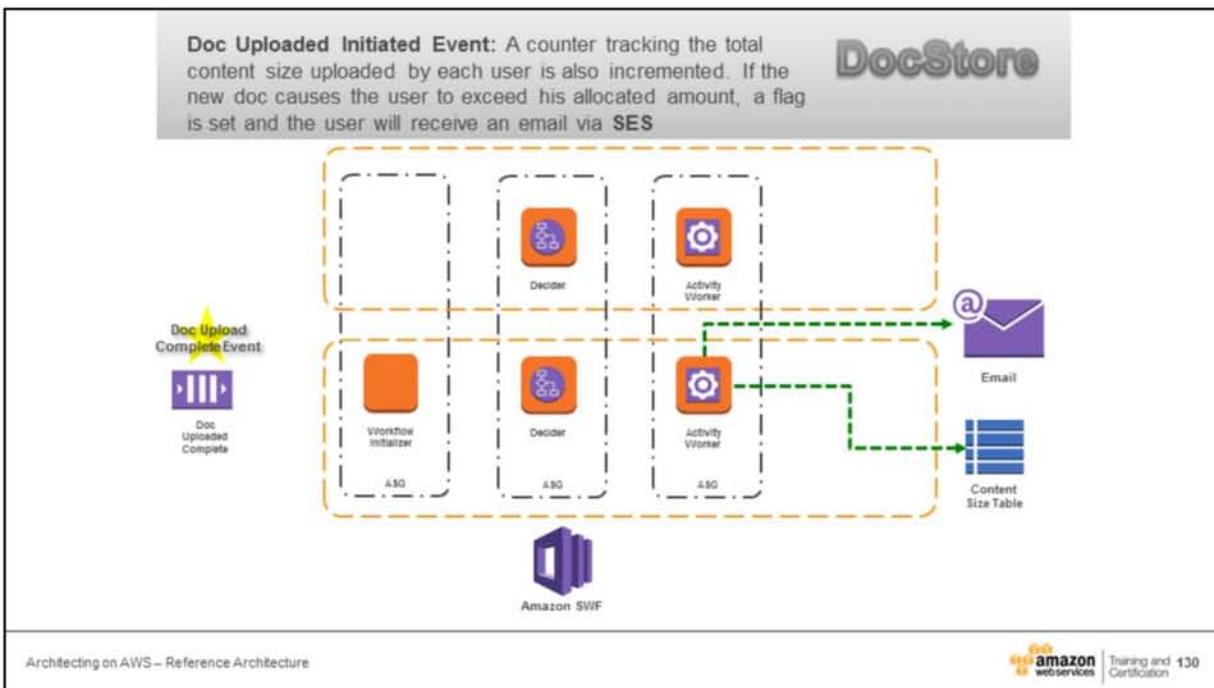
Doc Uploaded Complete Event: When a document upload has been complete, it needs to be processed

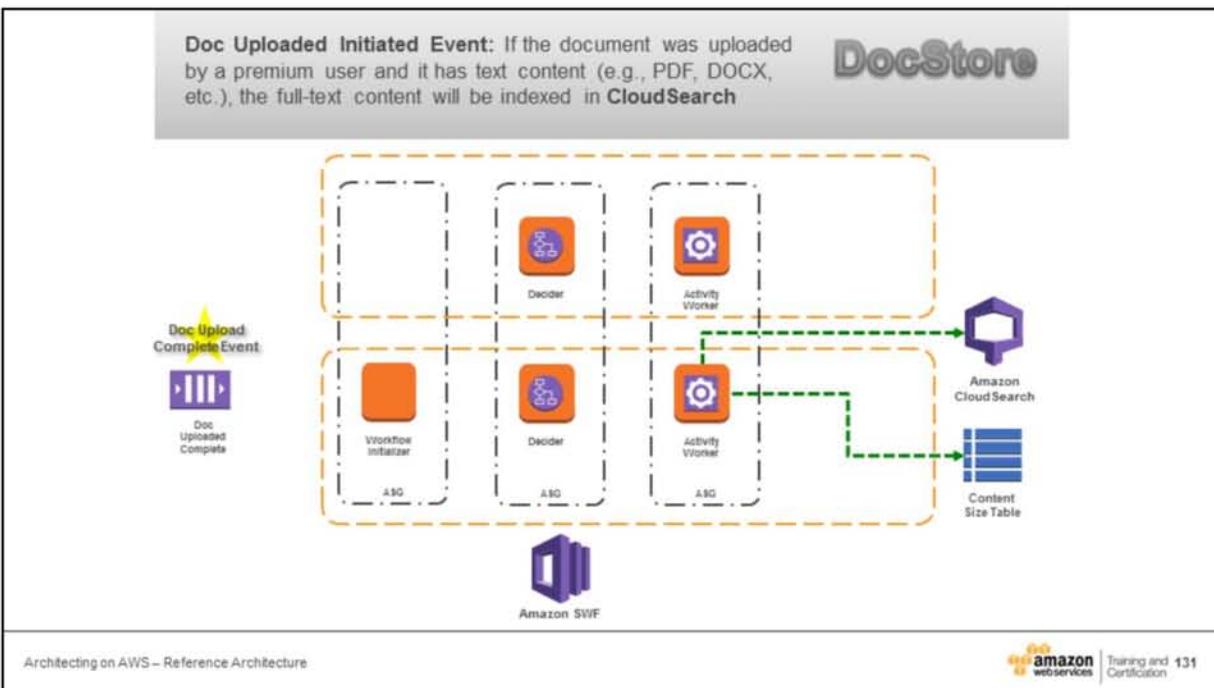


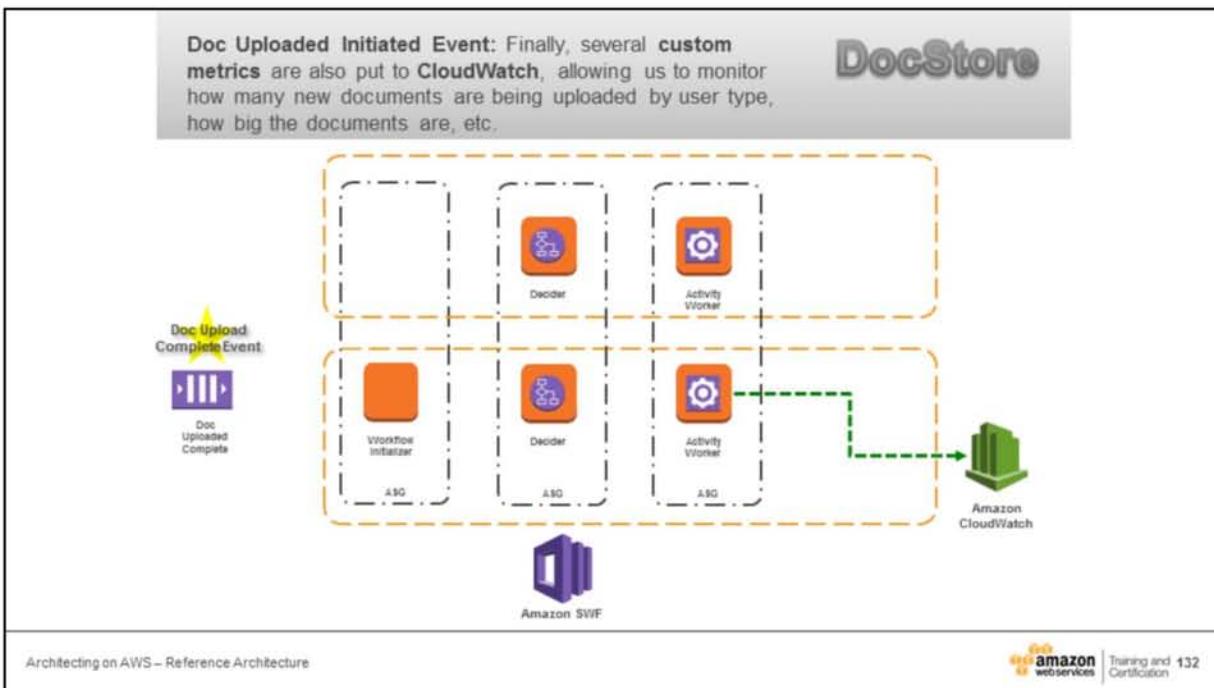












Copyright © 2013, 2014 Amazon Web Services, Inc. and its affiliates. All rights reserved.

This work may not be reproduced or redistributed, in whole or in part, without prior written permission from Amazon Web Services, Inc. Commercial copying, lending, or selling is prohibited.

Errors or corrections? Email us at aws-course-feedback@amazon.com.
Other questions? Email us at aws-training-info@amazon.com.

All trademarks are the property of their owners.