

CCA Unit 3 – Architecting on AWS

CCA 3.01: Review of AWS Fundamentals

Section 1: Introduction to System Design

► CCA 3.01: Review of AWS Fundamentals

CCA 3.02: Designing Your Environment

CCA 3.03: Design for High Availability

Section 2: Automation and Serverless Architectures

Section 3: Well-Architected Best Practices

Section 4: Deployment and Implementation

© 2017 Amazon Web Services, Inc. or its affiliates. All rights reserved.



This module begins CCA Unit 3 – Architecting on AWS. Before starting Unit 3, you should have completed Units 1 and 2.

Unit 3 is divided in four sections starting with **Section 1, Introduction to System Design:**

- * **CCA 3.01** with a **review AWS fundamentals** (should have covered in Units 1 & 2).
- * In **CCA 3.02** we'll discuss **Designing Your Environment** and general design principles followed by
- * **CCA 3.03, Design for High Availability**

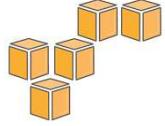
What's In This Module

Review of the following...

- Foundational Services
- Security
- Database Options
- Elasticity

This module covers a quick review of the following...

- Foundational Services
- Security
- Database Options
- Elasticity



Part 1

Foundational Services

© 2017 Amazon Web Services, Inc. or its affiliates. All rights reserved.

 Academy

Part 1: Foundational Services

EC2

Q1 You created an EC2 instance in Lab 01 using *Windows*. What is the other **operating system** you can create an EC2 instance for?

Answer:

Q2 An AMI defines which of the following:
(select all that apply)

- Root volume (OS, app server, etc.)
- Launch permissions
- What block volumes to attach

Q3 When launching an EC2 instance from an AMI, you can:
(select the best answer)

- Launch a single instance of a single type
- Launch a single instance of a different types
- Launch multiple instances of a single type
- Launch multiple instances of a different types

Q4 What family of instance types would you use for a system designed for data warehousing, logging, or data-processing applications?

- Compute-optimized
- Memory-optimized
- Storage-optimized

Discuss your answers with the class.

(For review, go to CCA 2.01 – Part 1)

VPCs and Subnets

Q1 You can customize the network configuration for your VPC, such as:
(select all that apply)

- selection of IP address range
- creation of subnets
- configuration of route tables
- network gateways

Q2 Match the server type to the best use of public/private subnets:



Q3 What would you use as a firewall to control inbound and outbound traffic? Match:



Q4 You can create a VPN connection to your remote network by using an Amazon EC2 instance in your VPC that's running a software VPN appliance.

- True
 False

Discuss your answers with the class.
(For review, go to CCA 2.01 – Part 2)

Storage Solutions

Q1 List some common use cases for Amazon S3:
(identify as many as you can)

Answer:

Storage and backup
Application file hosting
Media hosting
Software delivery
Store AMIs and snapshots

Q2 Data is stored using Amazon S3 as:

- Files
- Buckets
- Objects

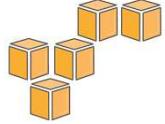
Q3 What storage option would be best suited for an application that require a database, file system, or access to raw block-level storage:

- S3
- Glacier
- EBS

Q4 When an EBS volume is attached to an instance that stops, what happens to the EBS volume?

- The EBS volume is also stopped
- The EBS volume persists

Discuss your answers with the class.
(For review, go to CCA 2.01 – Part 3)



Part 2 Security

© 2017 Amazon Web Services, Inc. or its affiliates. All rights reserved.

 **Amazon** | Academy

Part 2: Security

Shared Responsibility Model

- Q1** Under the Shared Responsibility Model, security of the platform, OS, firewall configuration, and network traffic protection are responsibilities of:
- The customer
 AWS
- Q2** AWS allows HTTPS access so that you can establish secure communication sessions with your AWS services, including SSL and TLS using *customer access points*, also called:
- Answer:** API endpoints
- Q3** What rule would you define for your security group to prevent individual clients from overloading a single server?
- Answer:** Accept traffic only from a load balancer

AWS Identity and Access Management (IAM)

Q1 List three entities to which IAM policies may be assigned:

Answer:

IAM Users
IAM Groups
IAM Roles

Q2 List three entities which may assume IAM Roles:

Answer:

IAM Users
applications
services

Q3 AWS IAM _____ appropriate for OS and application authentication

is
 is not

Q4 It is considered a best practice to use roles for applications that run on Amazon EC2 instances

True
 False

Discuss your answers with the class.
(For review, go to CCA 2.02 – Part 2)

AWS CloudTrail

Q1

CloudTrail can be used to...

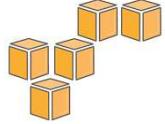
record **AWS API calls** for accounts

deliver **log files** to an Amazon S3 bucket

make calls using **AWS Management Console, AWS SDKs, AWS CLI and higher-level AWS services**
(list three)

Discuss your answers with the class.

(For review, go to CCA 2.02 – Part 2)



Part 3

Database Options

© 2017 Amazon Web Services, Inc. or its affiliates. All rights reserved.

 **Amazon**
webservices | Academy

Part 3: Database Options

Amazon Relational Database Service (RDS)

Q1

What database types you can use with RDS? (*List as many as you can think of*)

Answer: MySQL, MariaDB, Microsoft SQL Server, Oracle, or PostgreSQL

Q2

You must enable automatic backups on RDS which will allow you to restore your database to a point in time. [Automatic backups are enabled by default.](#)

True
 False

Q3

How do security groups help protect your databases?

Answer: Security groups control which IP addresses or EC2 instances can connect to your databases on a DB instance.

Q4

What can you do to provide failover for your database?

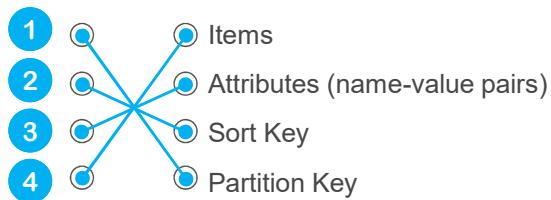
Answer: Provision a Multi-AZ DB instance which will automatically replicate the data to a standby instance in a different AZ.

Discuss your answers with the class.

(For review, go to CCA 2.03 – Part 2)

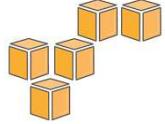
DynamoDB Data Model

Match the following:



Discuss your answers with the class.

(For review, go to CCA 2.03 – Part 3)



Part 4

Elasticity

© 2017 Amazon Web Services, Inc. or its affiliates. All rights reserved.

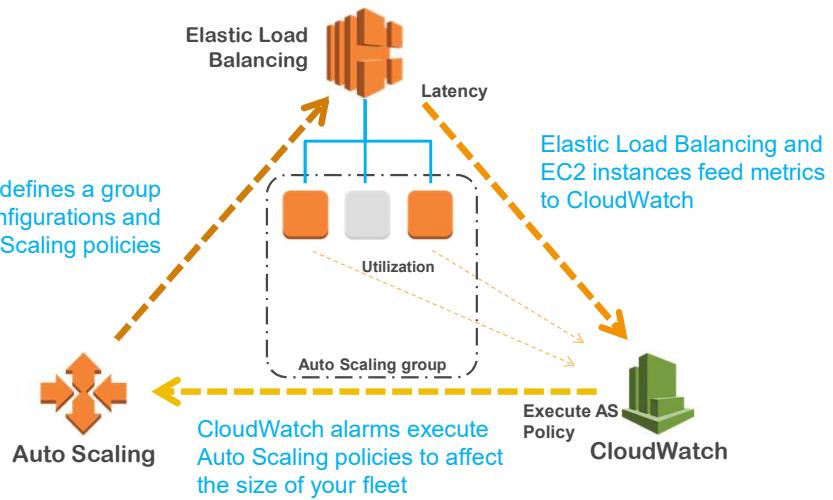
 Academy

Part 4: Elasticity

Elastic Load Balancing

Describe how ELB works with CloudFront and Auto Scaling:

Auto Scaling defines a group with launch configurations and Auto Scaling policies



Auto Scaling works as a triad of services working in sync. All of these services work well individually, but together they become more powerful and increase control and flexibility.

Discuss your answers with the class.

(For review, go to CCA 2.04 – Part 1)

Amazon CloudWatch

Q1 What does CloudWatch do?

Answer: CloudWatch is a metrics repository. AWS products put metrics into the repository, and you retrieve statistics based on the metrics. Statistics can be graphically presented in the CloudWatch console.

Q2 What can you do to receive notifications on specific metrics?

Answer: Set alarms

Q3 You can use CloudWatch only on AWS services.

You can create custom application-specific metrics of your own.

True
 False

Discuss your answers with the class.

(For review, go to CCA 2.03 – Part 2)

Auto Scaling

Q1 Describe what an Auto Scaling *launch configuration* is?

Answer: a template that an Auto Scaling group uses to launch EC2 instances.

Q2 With respect to Auto Scaling Groups, define the following:

Minimum: The least number of instances in the group, ensuring your group never goes **below** this size

Maximum: The largest number of instances in the group, ensuring your group never goes **over** this size

Desired: Targets this number of instances, automatically launching another instance if one goes down.

Scaling Policies: Tells Auto Scaling when to launch or terminate instances as demand on your application increases or decreases.

Q3 Auto Scaling is available at no additional charge.

True
 False

Discuss your answers with the class.

(For review, go to CCA 2.03 – Part 3)

AWS Trusted Advisor

Q1 What does Trusted Advisor do?

Answer: Provides AWS customers with performance and security recommendations in four categories: - Cost optimization; - Security; - Fault tolerance; - Performance improvement

Q2 Trusted Advisor can help improve performance by checking service limits where usage is more than 80% of the service limit.

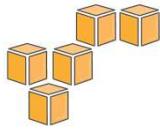
True
 False

Q3 List available fault tolerance checks:

(List as many as you can.)

Amazon EBS Snapshots
Load Balancer Optimization
Auto Scaling Group Resources
Amazon RDS Multi-AZ
Amazon Route 53 Name Server Delegations
ELB Connection Draining

Discuss your answers with the class.
(For review, go to CCA 2.03 – Part 4)



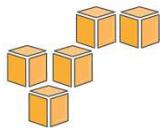
In review...

- Foundational Services
- Security
- Database Options
- Elasticity

In review...

- Foundational Services
- Security
- Database Options
- Elasticity

This module does not include a knowledge assessment.



Up Next...

[Section 1: Introduction to System Design](#)

CCA 3.01 – Review of AWS Fundamentals



CCA 3.02 – Designing Your Environment

CCA 3.03 – Design for High Availability

[Section 2: Automation and Serverless Architectures](#)

[Section 3: Well-Architected Best Practices](#)

[Section 4: Deployment and Implementation](#)

© 2017 Amazon Web Services, Inc. or its affiliates. All rights reserved.



Next we'll continue with Unit 3, Architecting on AWS which is broken into four sections:

[Section 1: Introduction to System Design](#)

[Section 2: Automation and Serverless Architectures](#)

[Section 3: Well-Architected Best Practices](#)

[Section 4: Deployment and Implementation](#)

Up next is **CCA 3.02 – Designing Your Environment**.

© 2017 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.

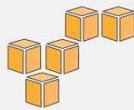
For all other questions, contact us at
<https://aws.amazon.com/contact-us/aws-training/>.

All trademarks are the property of their owners.

© 2017 Amazon Web Services, Inc. or its affiliates. All rights reserved.



Do not speak over this slide – just let it play for 8 seconds.



CCA Unit 1 – Introduction to System Design

CCA 3.02: Designing Your Environment

Section 1: Introduction to System Design

CCA 3.01 AWS Essentials Review

► CCA 3.02 Designing Your Environment

CCA 3.03 System Design for High Availability

Section 2: Automation and Serverless Architectures

Section 3: Well-Architected Best Practices

Section 4: Deployment and Implementation

© 2017 Amazon Web Services, Inc. or its affiliates. All rights reserved.



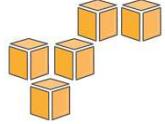
Welcome to **CCA 3.02, Designing Your Environment**.

What's In This Module

- Choosing a Region
- Use Multiple AZs
- Use Multiple VPCs
- Dividing VPCs Into Subnets
- Managing VPC Traffic
- Connecting Multiple VPCs
- Integrating On-prem Components
- Default VPCs and Subnets

This module covers...

- Choosing a Region
- Use Multiple AZs
- Use Multiple VPCs
- Dividing VPCs Into Subnets
- Managing VPC Traffic
- Connecting Multiple VPCs
- Integrating On-prem Components
- Default VPCs and Subnets



Part 1

How do you choose a region?

© 2017 Amazon Web Services, Inc. or its affiliates. All rights reserved.

 **Amazon**
webservices | Academy

Part 1: How do you choose a region?

How Do You Choose A Region?

Criteria:

Data Sovereignty

Distance to Users

Services/Features

Cost Efficiency

1. Does the region meet your environment's **data sovereignty and compliance** requirements?

- Do you meet data security laws of the country and locality in which your data is stored?
- Can your customer's data legally exist outside of the country where you are operating?
- Will you be able to meet your governance requirements by placing your environment in the region where you plan to place it?

Your data will be subject to the laws of the country and locality in which it's stored. In addition, some laws dictate that if you're operating your business in their jurisdiction, you cannot store that data anywhere else. Similarly, compliance standards (such as the Health Insurance Portability and Accountability Act in the United States) have strict guidelines on how and where data can be stored in order to comply. Take all of these things into account when evaluating where to place your environment.

How Do You Choose A Region?

Criteria:

Data Sovereignty

Distance to Users

Services/Features

Cost Efficiency

2. How close is the region to your users or data centers?

- 2006 Study: Every 100-ms delay on Amazon.com costs 1% in sales on the website.
- Most organizations choose an initial region closest to data centers or customers, e.g.:
 - New York: US East (N. Virginia)
 - Netherlands: EU (Ireland) or EU (Frankfurt)
 - Philippines: Asia Pacific (Singapore)
- Two equidistant regions to choose from? Consider which has lower costs.

Proximity is a major reason behind choosing your region, especially when latency is critically important, as it is in most applications. In most cases, the latency difference between using the closest region and the farthest away region is relatively small, but even small differences in latency can impact customer experience. An internal study in 2006 found that for every 100-ms Amazon.com is delayed, there was a corresponding 1% drop in sales. A Google study in 2006 similarly found that a 500ms delay for displaying their search results caused a 20% drop in traffic and revenue. Customers expect responsive environments, and as time goes by and technology becomes more and more powerful, those expectations rise as well.

To find more resources on the relationship between latency and revenue, see:

<http://highscalability.com/latency-everywhere-and-it-costs-you-sales-how-crush-it>

How Do You Choose A Region?

Criteria:

Data Sovereignty

Distance to Users

Services/Features

Cost Efficiency



3. Does the region you're considering offer **all of the services and features** your environment might require?
 - Not all services and features are available in all regions.
 - Although some services might not be available in your region, typically you can still use those services within your environment, but they may experience increased latency.
 - New services typically launch in a limited number of regions, with more regions added regularly.

While we strive to make all of our services and features available everywhere, the complications which arise from having a global reach make accomplishing that goal extremely challenging. But rather than wait until a service is available everywhere before launching it, we release our service when it's ready, and expand its availability as soon as possible.

How Do You Choose A Region?

Criteria:

Data Sovereignty
Distance to Users
Services/Features
Cost Efficiency

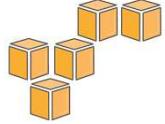
4. Are you choosing the most **cost-effective region?**

- Service costs vary by region.
- Some services (like Amazon S3) have costs when you transfer data out of their original region.
- If you want to replicate your environment across more than one region, is that the most cost-effective solution?

Service costs can differ depending on which region they're used. An Amazon EC2 instance in US-East 1 may not cost the same as if it were running in EU-West 1, as an example. Typically, the difference in cost may not be enough to supersede the other three considerations, however, in cases where the latency/compliance/service availability differences between regions are minimal, you may be able to save a lot of expense using the lower-cost region for your environment.

In circumstances where your customers are in different areas of the globe, you may consider optimizing your customer's experience by replicating your environment in multiple regions that are closer to your customers. Since you would then be distributing your load across multiple environments, your costs for components in each environment may go down even as you add more infrastructure. For example, adding a second application environment might allow you to cut your processing and storage capacity requirements in half in each environment. Since AWS is designed to allow you that kind of flexibility, and since you only really pay for what you use, you could easily scale your existing environment down as a way to mitigate the cost of adding another environment.

The downside to that approach is that you now have two environments to manage, and that not all of your components will scale down enough to mitigate all of the new component costs. Additionally, you may have to maintain one single storage "source of truth" in one region (such as a Master RDS instance), which your secondary region would have to communicate with, increasing latency and cost for those operations.



Part 2

How many Availability Zones should you use?

Part 2: How many Availability Zones should you use?

How Many Availability Zones Should I Use?

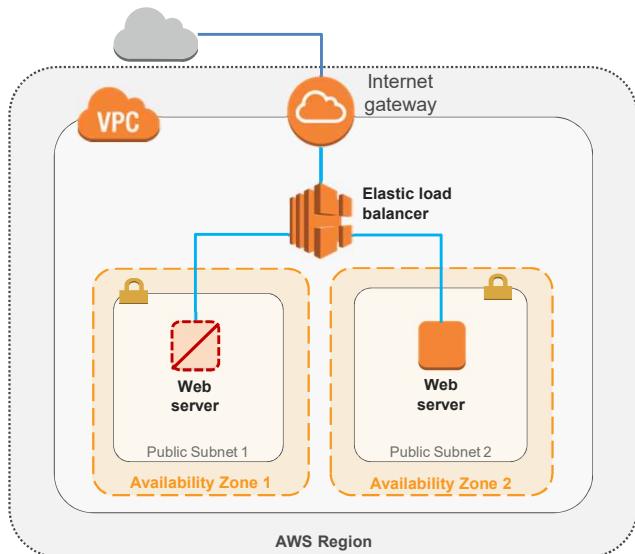
Recommendation: Start with **two** Availability Zones per region.

- **Best practice:** If resources in one Availability Zone are unreachable, your application shouldn't fail.
- Most applications can support two Availability Zones.
- Using more than two Availability Zones for HA is not usually cost-effective.

Most applications can be designed to support 2 AZs but may not benefit from more due to utilizing data sources that only support primary/secondary failover. Availability Zones are spread out physically, so you won't receive much benefit from duplicating your resources in three or more Availability Zones in one region.

For heavy Amazon EC2 Spot instance usage or data sources that go beyond active/passive, such as Amazon DynamoDB, there may be a benefit to using more than 2 AZs.

Using Two Availability Zones



In this basic pattern, the two web servers are positioned behind an Elastic Load Balancing load balancer, which distributes traffic between them.

1. If one of the servers becomes unavailable, the load balancer recognizes this.
2. It stops distributing traffic to the unhealthy instance.

This ensures that in case there's a problem in one of the AZs where a component resides, your application is still available.

You can further increase the availability of your infrastructure using other methods, which we will discuss in a later module.

Other Reasons To Use Two Availability Zones

How many Availability Zones would be recommended for each scenario?

1

Q: Applications heavily use Amazon EC2 Spot Instances:

A: Two Availability Zones or more for more **price options**

2

Q: Applications have data sources such as MySQL, MS SQL Server, and Oracle:

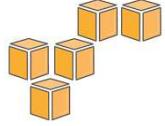
A: Two Availability Zones to **support active/passive**

3

Q: Applications have data sources such as Cassandra or MongoDB:

A: Two Availability Zones or more for **extremely high availability**

Since Amazon EC2 Spot instances are priced according to Availability Zone, you could leverage two Availability Zones to get the best price even when the prices change.



Part 3

Should you just fit everything into one VPC?

© 2017 Amazon Web Services, Inc. or its affiliates. All rights reserved.

 Academy

The Amazon Web Services logo, featuring the word "amazon" in lowercase with a yellow arrow above it, followed by "webservices" in smaller letters, and the word "Academy" to its right.

Part 3: Should you just fit everything into one VPC?

Using One VPC

There are **limited** use cases where one VPC could be appropriate:

- High-performance computing
- Identity management
- Small, single applications managed by one person or very small team

For **most** use cases, there are two primary patterns for organizing your infrastructure:

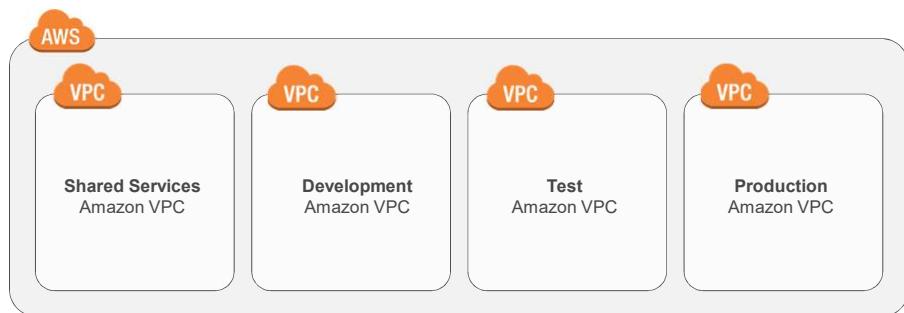
Multi-VPC and **Multi-Account**

High-performance computing environments may work best entirely within a single VPC, as a single VPC environment will have lower latency than one spread across multiple VPCs.

Identity management environments may best be limited to one VPC for best security. For small applications supported by a small team (or one person), it may be easiest to use one VPC.

AWS Infrastructure Patterns

VPC pattern



Account pattern



Choosing A Pattern

How do you know which pattern to use?

- The primary factors for determining this are the **complexity** of your organization and your **workload isolation** requirements:
 - Single IT team? **Multi-VPC**
 - Large organization with many IT teams? **Multi-account**
 - High workload isolation required? **Multi-account**

Multi-VPC Pattern

Features:

- Uses one AWS Account
- Uses two or more VPCs to organize application environments

Best suited for:

Single team or single organizations, such as Managed Service Providers

Why?

Limited teams make maintaining standards and managing access far easier

Exception:

Governance and compliance standards may require workload isolation regardless of organizational complexity

Multi-VPC patterns are best suited for a single team or organization that maintains full control over the provisioning and management of all resources in each application environment. For example, a single team developing a large e-commerce application may use this pattern when the developers have full access to the Dev & Prod environments. Also, this pattern is very common with Managed Service Providers (MSPs) managing all resources in Test & Prod.

Multi-Account Pattern

Features:

- Uses **two or more AWS accounts** to organize application environments
- Uses **one VPC** per AWS account

Best suited for:

- **Large organizations** and **organizations with multiple IT teams**, such as Enterprise-level corporations or government agencies
- **Medium-sized organizations** that anticipate rapid growth

Why?

Managing access and **standards** can be more challenging in more complex organizations.

Multi-account patterns are best suited for Enterprise customers or organizations deploying applications managed across multiple teams. For example, an organization supporting two or more teams may use this pattern to support developers having full access to the Dev environment resources but limited or no access to Prod.

Other Important Considerations

The majority of AWS services **do not actually sit within a VPC**.

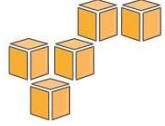
- For these services, a VPC **cannot provide any isolation** outside of connectivity.
- Communication between resources based in a VPC and resources outside of that VPC traverses the **public AWS network** by default.

Amazon S3 offers **VPC endpoints** to connect without traversing the public Internet:

- Endpoints are supported within the same region only.
- Support for endpoints with other services will be added in the future.

For more, see:

http://docs.aws.amazon.com/AmazonVPC/latest/UserGuide/VPC_Networking.html



Part 4

How should you divide your VPCs into subnets?

© 2017 Amazon Web Services, Inc. or its affiliates. All rights reserved.

 Academy

Part 4: How should you divide your VPCs into subnets?

VPCs And IP Addresses

- When you create your VPC, you specify its set of IP addresses with CIDR notation
- Classless Inter-Domain Routing (CIDR) notation is a simplified way to show a specific range of IP addresses
- Example: 10.0.0.0/16 = all IPs from 10.0.0.0 to 10.0.255.255
- How does that work? What does the 16 define?

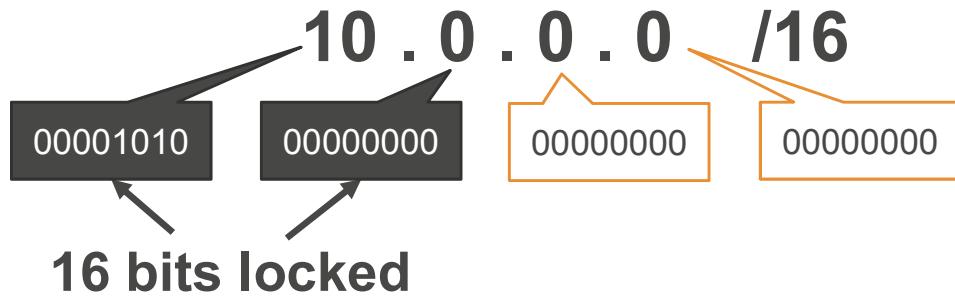
IPs and CIDR

Every set of 3 digits in an IP address represents a set of 8 binary values (8 bits).

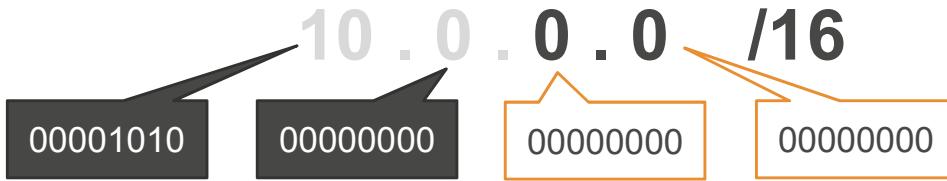


IPs and CIDR

The 16 in the CIDR notation example represents how many of those bits are "locked down" and cannot change.



IPs and CIDR



The unlocked bits can change between 1 and 0, allowing the full range of possible values.

CIDR Example: 10.0.0.0/16



VPCs and IP Addresses

- AWS VPCs can use CIDR ranges between /16 and /28.
- For every one step a CIDR range increases, the total number of IPs is cut in half:

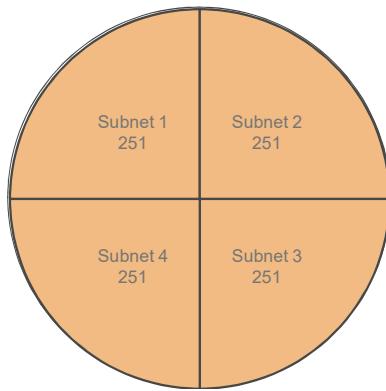
CIDR / Total IPs						
/16	/17	/18	/19	/20	/21	/22
65,536	32,768	16,384	8,192	4,096	2,048	1,024
/23	/24	/25	/26	/27	/28	
512	256	128	64	32	16	

What Are Subnets?

Subnets are **segments** or **partitions** of a network, divided by **CIDR range**.

Example:

A VPC with **CIDR /22** includes 1,024 total IPs



Note: In every subnet,
the first four and last
one IP addresses are
reserved for AWS use.

The first four IP addresses and the last IP address in each subnet CIDR block are not available for you to use, and cannot be assigned to an instance. For example, in a subnet with CIDR block 10.0.0.0/24, the following five IP addresses are reserved:

- 10.0.0.0: Network address.
- 10.0.0.1: Reserved by AWS for the VPC router.
- 10.0.0.2: Reserved by AWS for mapping to the Amazon-provided DNS.
- 10.0.0.3: Reserved by AWS for future use.
- 10.0.0.255: Network broadcast address. We do not support broadcast in a VPC, therefore we reserve this address.

How to Use Subnets

Recommendation: Use subnets to define Internet accessibility.

Public subnets

Include a routing table entry to an **Internet gateway** to support inbound/outbound access to the public Internet.

Private subnets

Do not have a routing table entry to an Internet gateway and are **not directly accessible** from the public Internet.

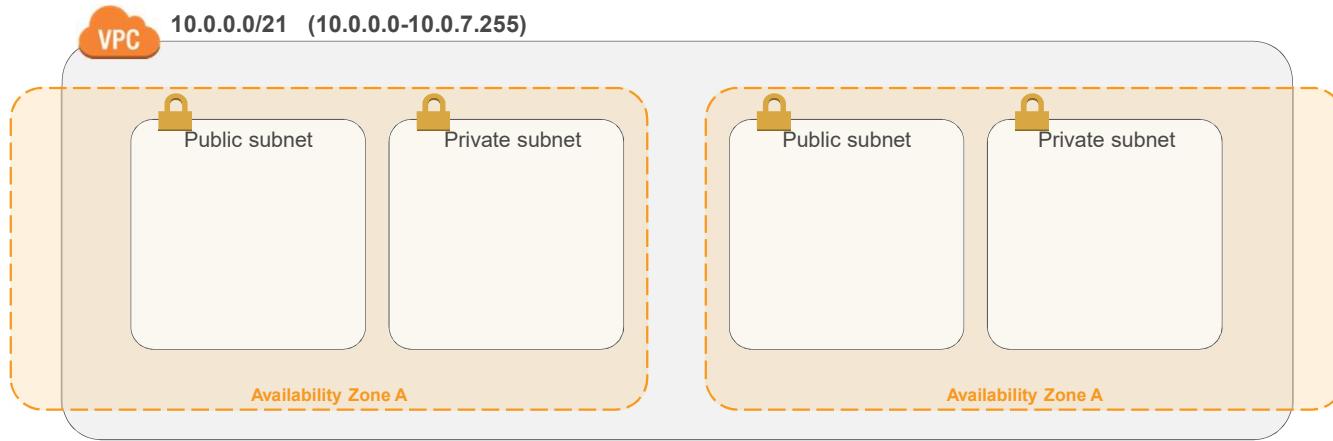
Typically use a "jump box" (NAT/proxy/bastion host) to support restricted, **outbound-only** public Internet access.

Rather than define your subnets based on application or functional tier (web/app/data/etc), you should organize your subnets based on Internet accessibility. This allows you to define clear, subnet-level isolation between public and private resources.

Note: In certain circumstances, such as for PCI compliance, where extremely sensitive data cannot have any direct or indirect connection to the Internet, that subnet is referred to as "protected" subnet.

Subnets

Recommendation: Start with **one public** and **one private** subnet per Availability Zone.



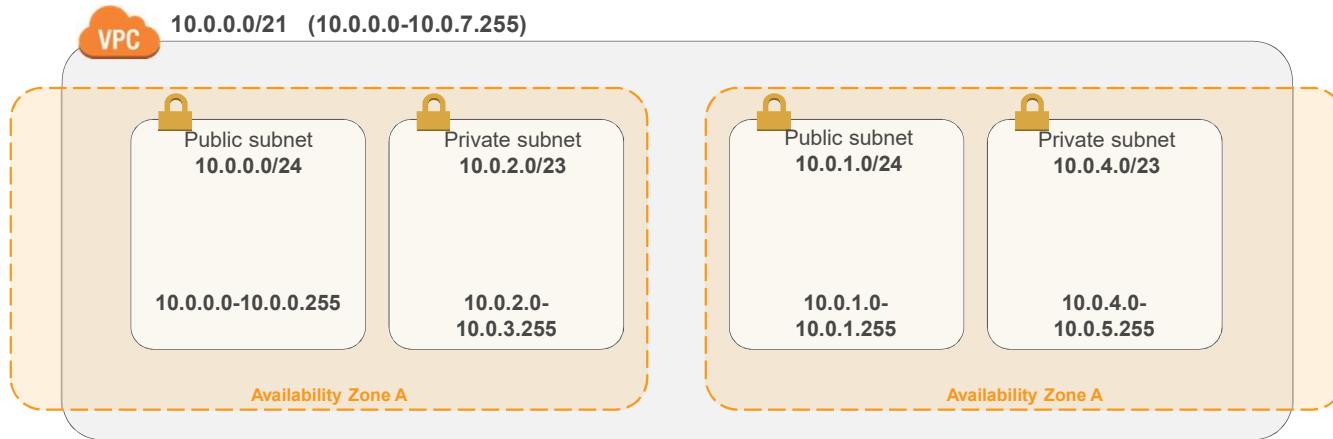
As subnets should be used to define Internet accessibility, there may not be a good reason to have more than one public and one private subnet per Availability Zone. In this environment, all of your resources that require direct access to the Internet (public-facing load balancers, NAT instances, bastion hosts, etc.) would go into the public subnet, while all other instances would go into your private subnet (exception: resources which require absolutely no access to the Internet, either directly or indirectly, would go into a separate private subnet).

Some environments try to use subnets to create layers of separation between "tiers" of resources, such as putting your back-end application instances and your data resources into separate private subnets. This practice requires you to more accurately predict how many hosts you will need in each subnet, making it more likely that you will either run out of IPs more quickly, or leave too many IPs unused, when they could be used elsewhere.

While subnets can provide a very basic element of segregation between resources using network ACL rules, security groups can provide an even more finely grained level of traffic control between your resources, without the risk of overcomplicating your infrastructure and wasting or running out of IPs. With this approach, you just need to anticipate how many public and how many private IPs your VPC needs, and use other resources to create segregation between resources within a subnet.

Subnets

Recommendation: Start with **one public** and **one private** subnet per Availability Zone.



Subnet Sizes

Recommendation: Consider larger subnets over smaller ones (/24 and larger).

Simplifies workload placement:

Choosing where to place a workload among 10 small subnets is more complicated than with one large subnet.

Less likely to waste or run out of IPs:

If your subnet runs out of available IPs, you can't add more to that subnet.

Example: If you have 251 IPs in a subnet that's using only 25 of them, you can't share the unused 226 IPs with another subnet that's running out.

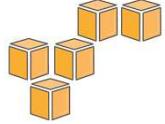
Note: It is no longer necessary to limit Address Resolution Protocol (ARP) broadcast domains as this is solved by the VPC.

Subnet Types

Which subnet type (public or private) should you use for these resources ?

	Public	Private
Datastore instances	<input type="checkbox"/>	<input checked="" type="checkbox"/>
Batch processing instances	<input type="checkbox"/>	<input checked="" type="checkbox"/>
Back-end instances	<input type="checkbox"/>	<input checked="" type="checkbox"/>
Web application instances	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>

- * While you can put web-tier instances into a public subnet, we actually recommend you place your web-tier instances inside of private subnets, behind a load balancer placed in a public subnet. Some environments require web application instances to be attached to Elastic IPs directly (even though you can also attach an Elastic IP to a load balancer), and in those cases web application instances would need to be in a public subnet. We will talk more about load balancers in a later module.



Part 5

How do you control your VPC traffic?

- Route tables
- Security groups
- Network ACLs
- Internet gateways

Part 5: How do you control your VPC traffic?

Route Tables

Directing Traffic Between VPC Resources

- Determine where network traffic is routed
- Main and custom route tables
- VPC route table: *Local route*
- Only one route table per subnet



Best practice:

For better security, use **custom** route tables for **each subnet**.

A route table contains a set of rules, called routes, that are used to determine where network traffic is directed.

When you create a VPC, it automatically has a main route table. Initially, the main route table (and every route table in a VPC) contains only a single route: a local route that enables communication within the VPC. You can't modify the local route in a route table. Whenever you launch an instance in the VPC, the local route automatically covers that instance; you don't need to add the new instance to a route table. You can create additional custom route tables for your VPC.

Each subnet in your VPC must be associated with a route table, which controls the routing for the subnet. If you don't explicitly associate a subnet with a particular route table, the subnet is implicitly associated with and uses the main route table. A subnet can be associated with only one route table at a time, but you can associate multiple subnets with the same route table.

One way to protect your VPC is to leave the main route table in its original default state (with only the local route), and explicitly associate each new subnet you create with one of the custom route tables you've created. This ensures that you must explicitly control how each subnet's outbound traffic is routed.

Security Groups

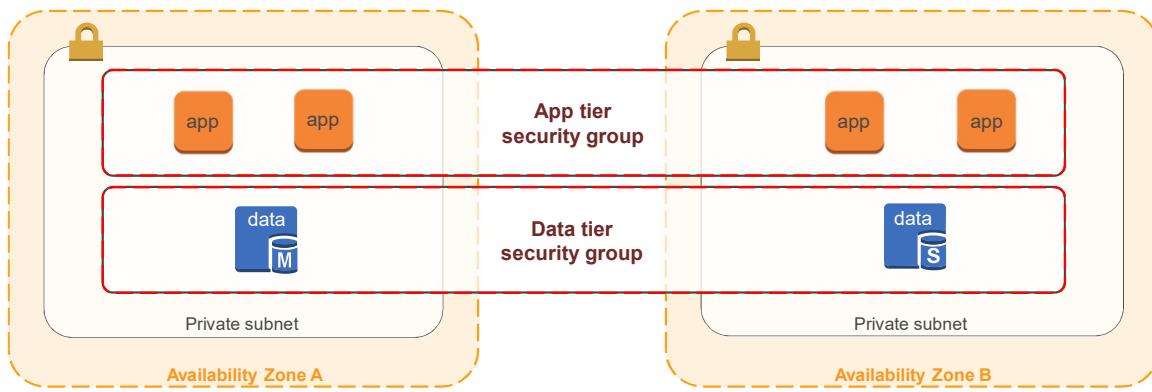
Securing VPC Traffic With Security Groups

- Are **virtual firewalls** that control inbound and outbound traffic for one or more instances.
- Deny all incoming traffic by default and use *allow* rules that can filter based on TCP, UDP, and ICMP protocols.
- Are **stateful**, which means that if your inbound request is allowed, the outbound response does not have to be inspected/tracked, and vice versa.
- Can define a **source/target** as either a **CIDR block** or **another security group** to handle situations like auto scaling.

Regarding stateful rules: for example, if you initiate an ICMP *ping* command to your instance from your home computer, and your inbound security group rules allow ICMP traffic, information about the connection (including the port information) is tracked. Response traffic from the instance for the ping command is not tracked as a new request, but instead as an established connection, and is allowed to flow out of the instance, even if your outbound security group rules restrict outbound ICMP traffic. Not all flows of traffic are tracked. If a security group rule permits TCP or UDP flows for all traffic (0.0.0.0/0), and there is a corresponding rule in the other direction that permits the response traffic, that flow of traffic is not tracked. The response traffic is therefore allowed to flow based on the inbound or outbound rule that permits the response traffic, and not on tracking information.

Security Groups

Use security groups to control traffic **into, out of, and between resources**.

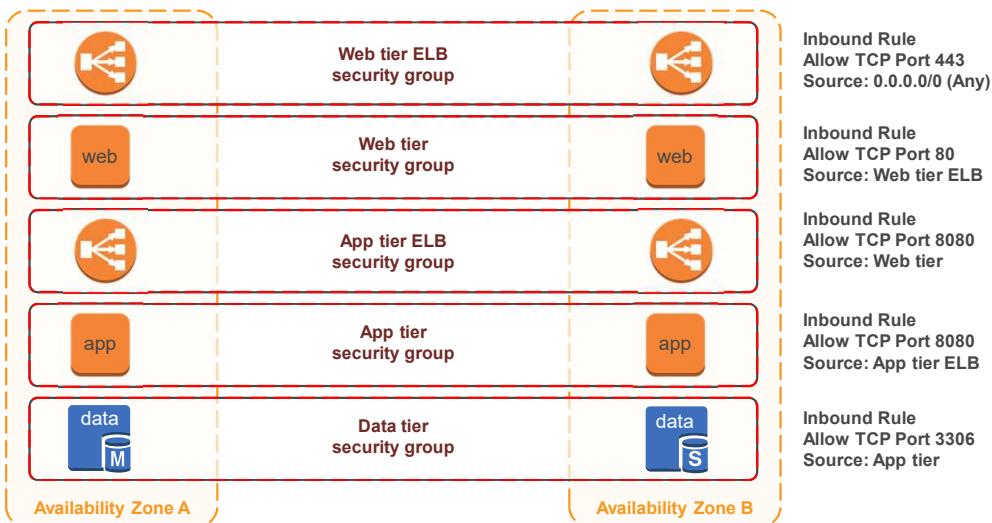


How Security Groups Are Configured

- By default, all newly created security groups **allow all outbound traffic** to all destinations. Modifying the default outbound rule on security groups increases complexity and is not recommended unless required for compliance.
- Most organizations create security groups with **inbound rules** for **each functional tier** (web/app/data/etc.) within an application.

Security Group Chaining Diagram

Security group rules per application tier



Here's an example of a chain of security groups. The inbound and outbound rules are set up in a way that traffic can only flow from the top tier to the bottom tier and back up again. The security groups act as firewalls to prevent a security breach in one tier to automatically provide subnet-wide access of all resources to the compromised client.

Network ACLs

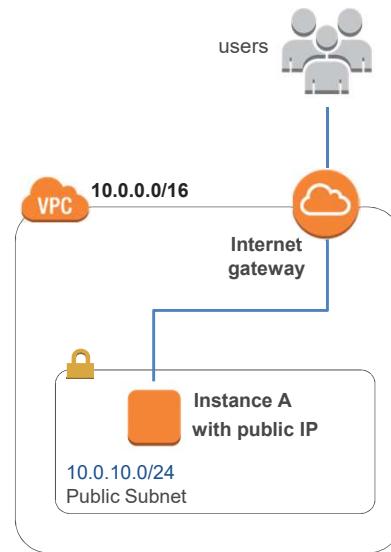
- Are **optional virtual firewalls** that control traffic in and out of a subnet.
- **Allow all** incoming/outgoing traffic by default and use **stateless** rules to allow or deny traffic.
"Stateless rules" inspect all inbound and outbound traffic and do not keep track of connections.
- Enforce rules only at the **boundary of the subnet**, not at the instance-level, like security groups.

Security groups provide far more granular control over traffic than NACLs. In some cases, NACLs are explicitly required for certain compliance standards. In this case, we recommend your NACLs act as redundantly with your security groups as possible.

Internet gateways

Directing Traffic To Your VPC

- Allow communication between instances in your VPC and the Internet.
- Are horizontally scaled, redundant, and highly available by default.
- Provide a target in your VPC route tables for Internet-routable traffic.



Directing Traffic To Your VPC

To enable access to or from the Internet for instances in a VPC subnet, you must:

- Attach an Internet gateway to your VPC
- Ensure that your subnet's route table points to the Internet gateway
- Ensure that instances in your subnet have public IP addresses or Elastic IP addresses
- Ensure that your NACLs and security groups allow the relevant traffic to flow to and from your instance

What About Outbound Traffic From Private Instances?

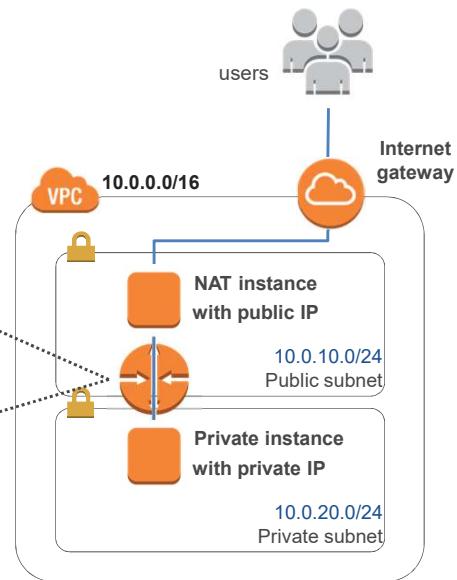
Network Address Translation services:

- Enable instances in the private subnet to initiate outbound traffic to the Internet or other AWS services.
- Prevent private instances from receiving inbound traffic from the Internet.

Two primary options:

- Amazon EC2 instance set up as a NAT in a public subnet

Destination	Target
10.0.0.0/16	local
0.0.0.0/0	NAT



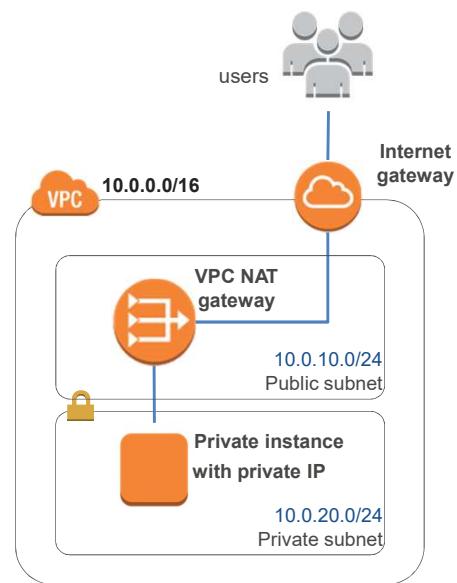
What About Outbound Traffic From Private Instances ?

Network Address Translation services:

- Enable instances in the private subnet to initiate outbound traffic to the Internet or other AWS services.
- Prevent private instances from receiving inbound traffic from the Internet.

Two primary options:

- Amazon EC2 instance set up as a NAT in a public subnet
- VPC NAT Gateway

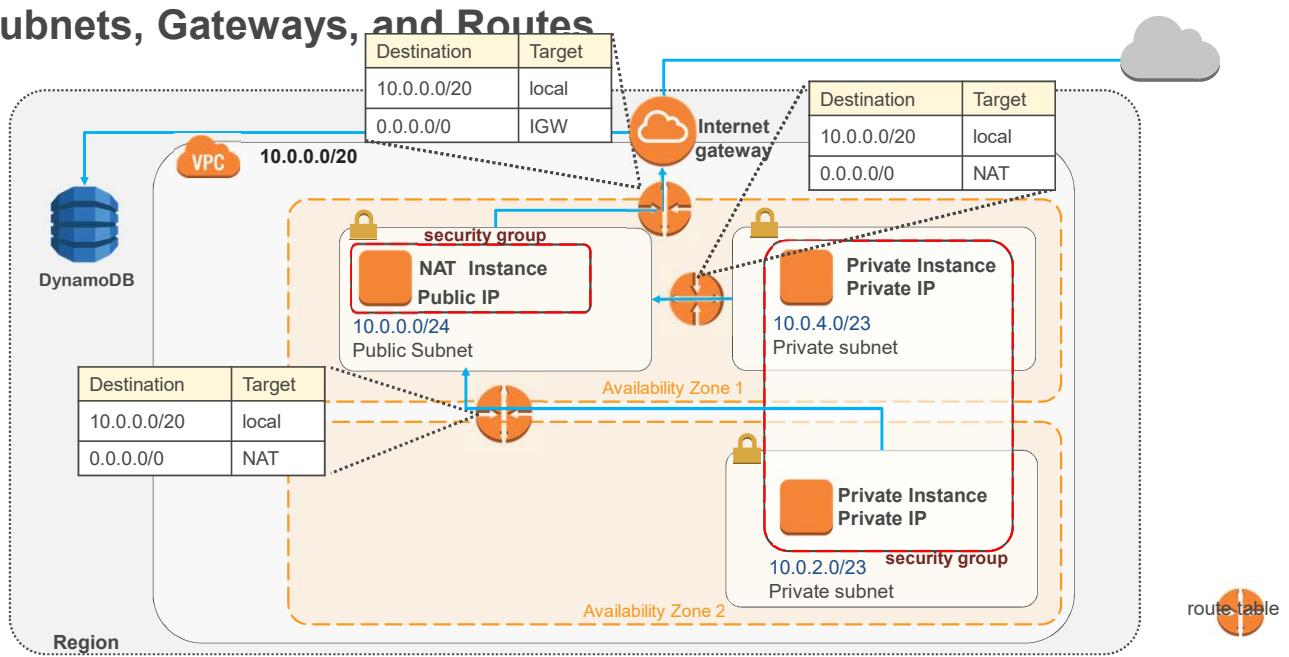


VPC NAT Gateways vs. NAT Instances On Amazon EC2

	VPC NAT gateway	NAT instance
Availability	Highly available by default	Use script to manage failover
Bandwidth	Bursts to 10 Gbps	Based on bandwidth of instance type
Maintenance	Managed by AWS	Managed by you
Security	NACLs	Security groups and NACLs
Port forwarding	Not supported	Supported

You should carefully consider your decision on which Network Address Translation solution to use. While VPC NAT gateways offer all the advantages of being a service managed by AWS (such as being inherently highly available), they may not provide the exact level of control your application needs. There are also cost differences to consider between the two options.

Subnets, Gateways, and Routes



© 2017 Amazon Web Services, Inc. or its affiliates. All rights reserved.

1. To make a subnet public, it must be routed through an Internet gateway.
2. If instances in your private subnets need to send data out to the Internet, you'll need to route traffic to something like a NAT instance (or NAT gateway).

Logging VPC Traffic

Amazon VPC Flow Logs

- Captures traffic flow details in your VPC
Accepted and rejected traffic
- Can be enabled for VPCs, subnets, and ENIs
- Logs published to CloudWatch Logs

Use cases:

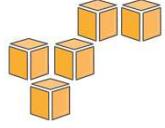
- Troubleshoot connectivity issues.
- Test network access rules.
- Monitor traffic.
- Detect and investigate security incidents.

How can you verify that the configured network access rules are working as expected? Many organizations typically collect, store, monitor, and analyze network flow logs for various purposes, including troubleshooting connectivity and security issues and testing network access rules.

VPC Flow Logs is a feature that enables you to capture information about the IP traffic going to and from network interfaces in your VPC. The flow log will capture accepted and rejected traffic flow information for all network interfaces in the selected resource. The information captured by flow logs can help you with a number of tasks; for example, troubleshooting why specific traffic is not reaching an instance, which in turn can help you diagnose overly restrictive security group rules. You can also use flow logs as a security tool to monitor the traffic that is reaching your instance. You can create alarms to notify you if certain types of traffic are detected; you can also create metrics to help you to identify trends and patterns.

You can create a flow log for a VPC, a subnet, or a network interface. If you create a flow log for a subnet or VPC, each network interface in the VPC or subnet is monitored. Flow log data is published to a log group in CloudWatch Logs, and each network interface has a unique log stream. Log streams contain flow log records, which are log events consisting of fields that describe the traffic for that network interface. Amazon CloudWatch and CloudWatch Logs are covered later.

You can analyze the data captured from flow logs using your own applications or using solutions available from AWS Marketplace.



Part 6

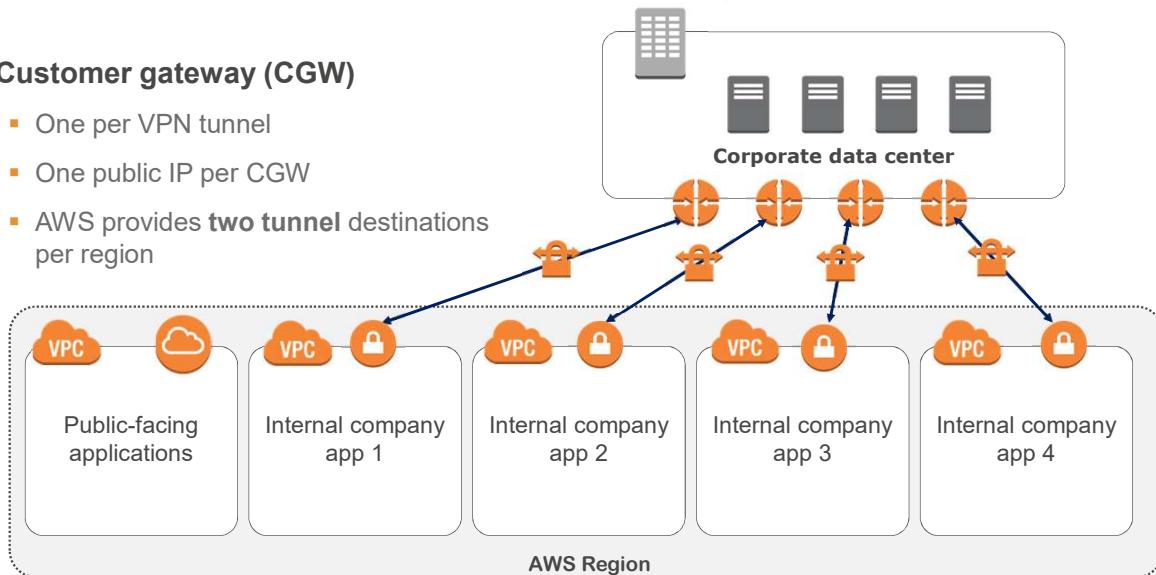
Can you connect multiple VPCs to each other?

Part 6: Can you connect multiple VPCs to each other?

Is This the Most Efficient Way Of Connecting VPCs?

Customer gateway (CGW)

- One per VPN tunnel
- One public IP per CGW
- AWS provides **two tunnel destinations** per region



If multiple VPCs need to connect to the data center, you can clearly see the challenge of using CGW. You would have to maintain unique IP addresses on your end and route. You can have VPN hub-and-spoke architecture to work around this; however, is it really an efficient solution?

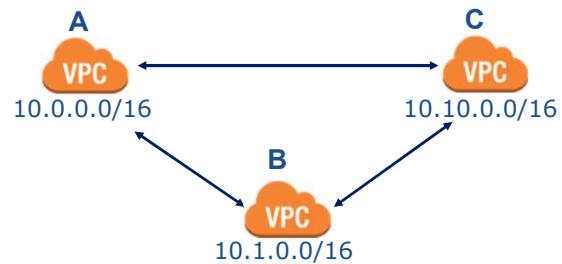
VPN hub-and-spoke architecture:

- Amazon EC2 VPN instances to central VGW
- For high availability, two EC2-based VPN endpoints in each spoke
- Central VPC (hub) contains common services for all app VPCs
- Dynamic routing protocol between spokes and hub

Resolution: VPC Peering

VPC peering connection allows you to route traffic between the peer VPCs.

- Use private IP addresses.
- VPCs reside in the same region.
- IP space cannot overlap.
- Only one between any two VPCs.
- Transitive peering relationships are not supported



Instances in either VPC can communicate with each other as if they are within the same network.

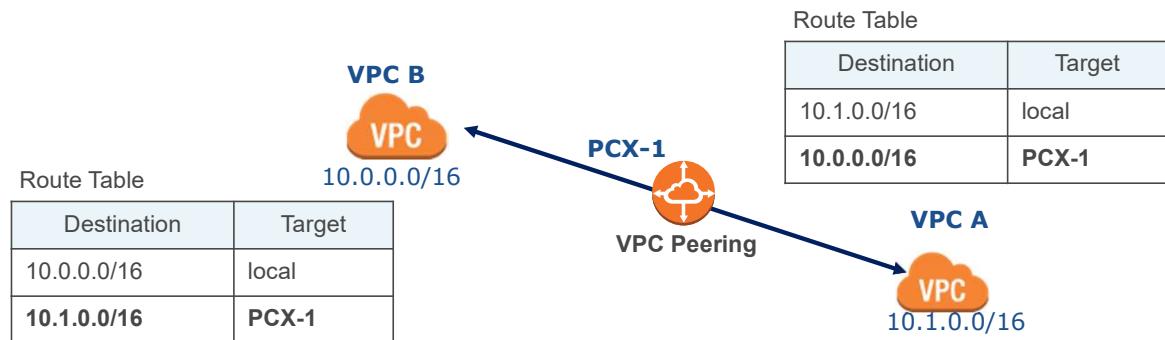
In the diagram, VPCs A and B are peered, which does not mean that C can talk to A. By default, VPC peering does not allow C to connect to A unless they are explicitly established as peers. Therefore, you control which VPCs can talk to each other.

To establish a VPC peering connection, the owner of the requester VPC (or local VPC) sends a request to the owner of the peer VPC to create the VPC peering connection. The peer VPC can be owned by you or another AWS account, and cannot have a CIDR block that overlaps with the requester VPC's CIDR block. The owner of the peer VPC has to accept the VPC peering connection request to activate the VPC peering connection. To enable the flow of the traffic between the peer VPCs using private IP addresses, add a route to one or more of your VPC's route tables that points to the IP address range of the peer VPC. The owner of the peer VPC adds a route to one of their VPC's route tables that points to the IP address range of your VPC. You may also need to update the security group rules that are associated with your instance to ensure that traffic to and from the peer VPC is not restricted.

A VPC peering connection is a one-to-one relationship between two VPCs. You can create multiple VPC peering connections for each VPC that you own, but transitive peering relationships are not supported: you will not have any peering relationship with VPCs that your VPC is not directly peered with.

How Does VPC Peering Work?

- No Internet gateway or virtual gateway required
- No single point of failure
- No bandwidth bottlenecks



The example on this slide shows that an entire CIDR block is opened, but you can specify an IP on the route table (10.0.150.38/32).

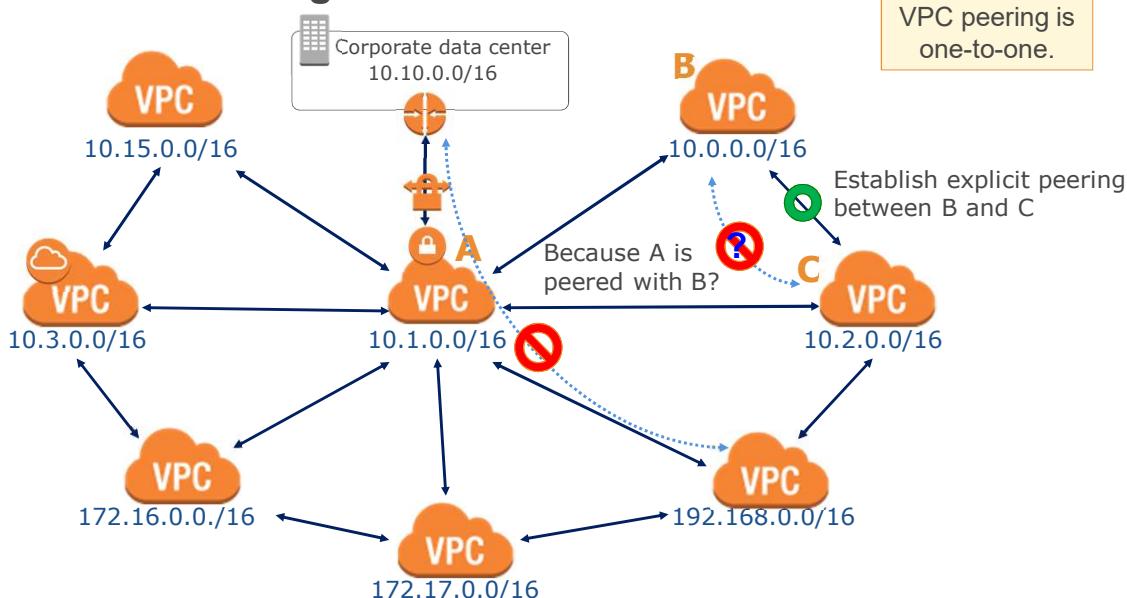
To create a VPC peering connection with another VPC, you need to be aware of the following limitations and rules:

- You cannot create a VPC peering connection between VPCs in different regions.
- There is a limit on the number of active and pending VPC peering connections that you can have per VPC.
- VPC peering does not support transitive peering relationships; in a VPC peering connection, your VPC will not have access to any other VPCs that the peer VPC may be peered with. This includes VPC peering connections that are established entirely within your own AWS account.
- You cannot have more than one VPC peering connection between the same two VPCs at the same time.
- The Maximum Transmission Unit (MTU) across a VPC peering connection is 1500 bytes.
- A placement group can span peered VPCs; however, you will not get full-bisection bandwidth between instances in peered VPCs.
- Unicast reverse path forwarding in VPC peering connections is not supported.
- You cannot reference a security group from the peer VPC as a source or destination for ingress or egress rules in your security group. Instead, reference CIDR blocks of the peer VPC as the source or destination of your security

group's ingress or egress rules.

- Private DNS values cannot be resolved between instances in peered VPCs.

Rules Of VPC Peering



In the diagram, VPCs A and B are peered, which does not mean that C can talk to B. By default, VPC peering does not allow C to connect to B unless they are explicitly established as peers. The same for corporate data center access. VPN connections to one VPC cannot be shared across a peered connection.

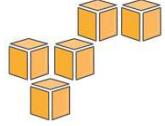
You can configure all the VPCs as peered to each other. You may think that is too much, but it is not, because VPC peering does not involve any hardware or external configuration. Even peering between 50 VPCs is reasonable.

Consider a case where your organization wants to share resources across the entire organization. For example, a VPC dedicated for the accounting department and a VPC for the engineering department both need to access shared services such as a management approval service or procurement service. However, the engineering VPC should not be able to access the accounting VPC. In such a case, you create VPC peering between the accounting VPC and the shared services VPC, and also create VPC peering between the engineering VPC and the shared services VPC.

In this scenario, only A has a connection to the corporate data center, and can serve as a hub. There are various peering scenarios that you can configure. Refer to the online documentation for more scenarios: <http://docs.aws.amazon.com/AmazonVPC/latest/PeeringGuide/peering-configurations.html>

VPC Peering Security

- Two-way handshake to establish a peering connection
- Routing controls: Routing tables control the local subnets that can route to remote subnets
- Security groups control what traffic an instance can send or receive
- Network ACLs control what traffic a subnet can send or receive
- No edge-to-edge routing or transitive trusts: Reduces inadvertently creating unexpected network connections



Part 7

How do you integrate on-premises components into your environment?

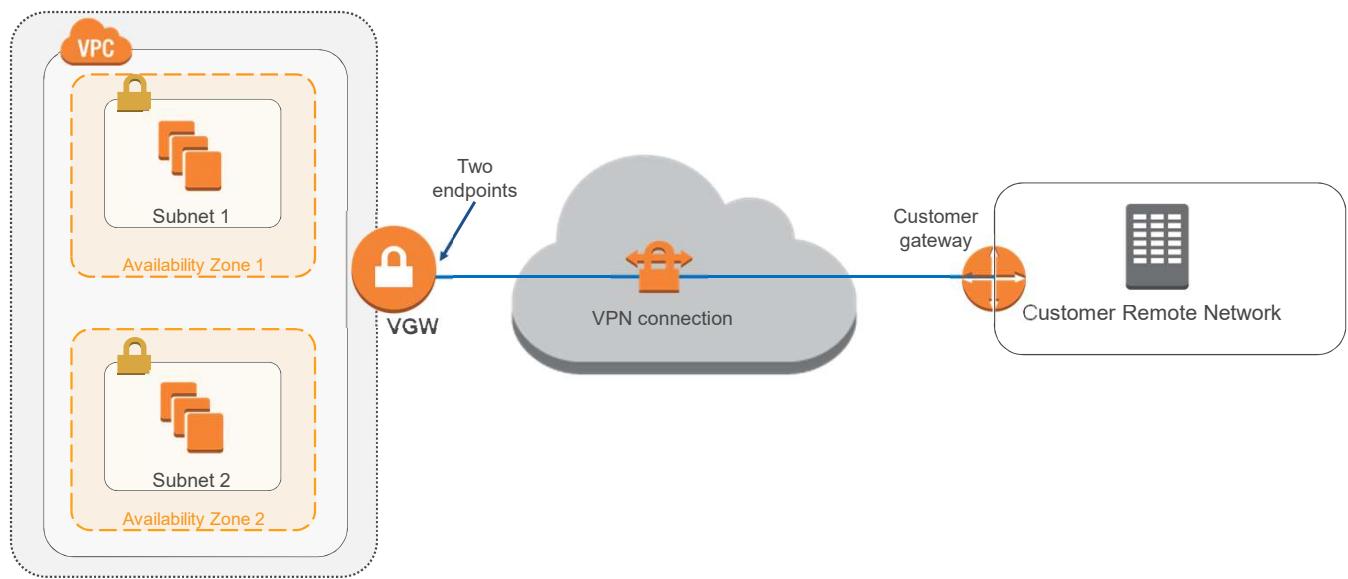
© 2017 Amazon Web Services, Inc. or its affiliates. All rights reserved.

 Academy

The Amazon Web Services logo, featuring the brand name in a stylized font above the words "webservices". To the right, the word "Academy" is written in a smaller, regular font.

Part 7: How do you integrate on-premises components into your environment?

Extending On-Premises Network To AWS: VPN Connections



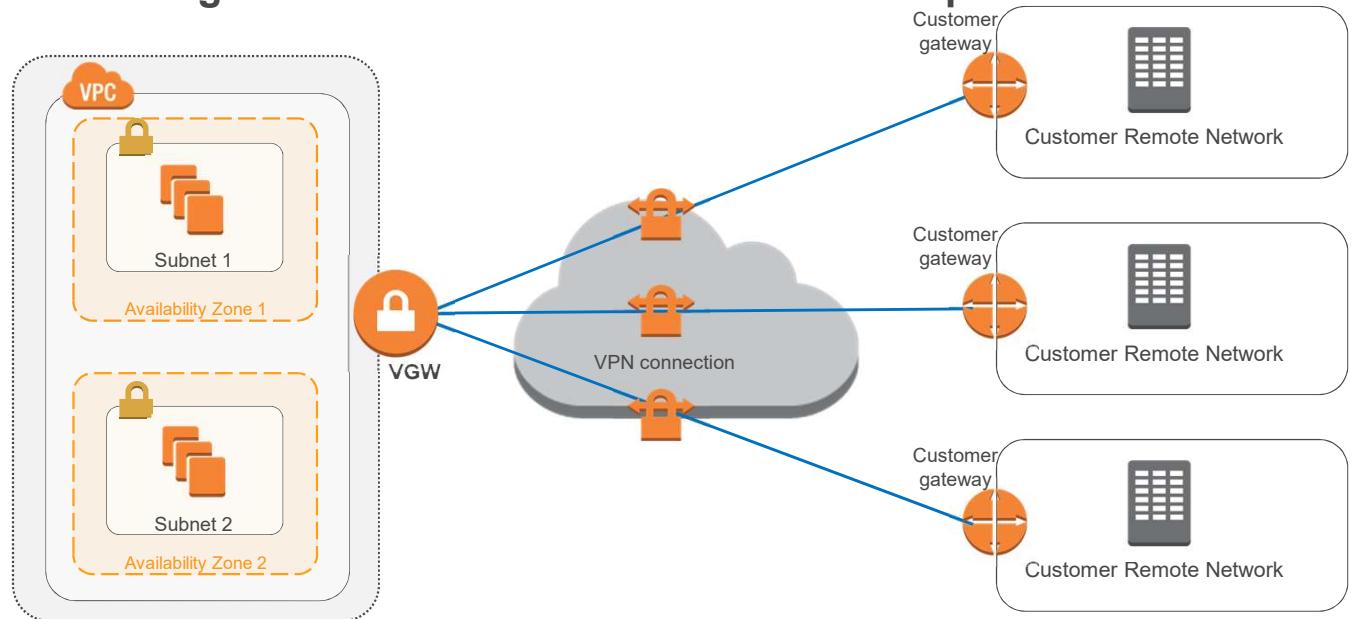
© 2017 Amazon Web Services, Inc. or its affiliates. All rights reserved.



One solution is to use a VPN connection between your VPC's Virtual Gateway and your data center. With an AWS hardware VPN, you are provided with two VPN endpoints to provide basic, automatic failover. To create an AWS hardware VPN, go here: http://docs.aws.amazon.com/AmazonVPC/latest/UserGuide/VPC_VPN.html

You can also create a VPN connection to your remote network by using an Amazon EC2 instance in your VPC that's running a software VPN appliance. AWS does not provide or maintain software VPN appliances; however, you can choose from a range of products provided by partners and open source communities on the AWS Marketplace.

Extending On-Premises Network To AWS: Multiple VPN



© 2017 Amazon Web Services, Inc. or its affiliates. All rights reserved.



Amazon VGW also supports and encourages multiple customer gateway connections so that customers can implement redundancy and failover on their side of the VPN connection, as shown on this slide. Both dynamic and static routing options are provided to give customers flexibility in their routing configuration. Dynamic routing leverages BGP peering to exchange routing information between AWS and these remote endpoints. Dynamic routing also allows customers to specify routing priorities, policies, and weights (metrics) in their BGP advertisements and to influence the network path between their networks and AWS.

It is important to note that when BGP is used, both the IPSec and the BGP connections must be terminated on the same customer gateway device, so it must be capable of terminating both IPSec and BGP connections.



AWS Direct Connect

- **AWS Direct Connect** provides you with a private network connection between AWS and your data center.
- It is a network service alternative to using the Internet to access AWS cloud services.

Benefits:

- Reduced network transfer costs
- Improved application performance with predictable metrics
- Transferring large data sets
- Security and compliance
- Hybrid cloud architectures
- Private data center expansion
- Alternative to Internet-based IPSec VPN connections (IPSec VPN connections can be used as a failover)

AWS Direct Connect is a unique solution to go beyond simple connectivity over the Internet and, instead, get access with scale, speed, and consistency to the AWS network for these important applications. AWS Direct Connect does not involve the Internet; instead, it uses dedicated, private network connections between your on-premises solutions and AWS.

Service Benefits

There are several scenarios for which AWS Direct Connect is useful. Some common scenarios are described below.

Transferring Large Data Sets

Consider an HPC application that operates on large data sets that must be transferred between your data center and the AWS cloud. For such applications, connecting to the AWS cloud using AWS Direct Connect is a good solution:

- Network transfers will not compete for Internet bandwidth at your data center or office location.
- The high bandwidth link reduces the potential for network congestion and degraded application performance.

Reduced Network Transfer Costs

By using AWS Direct Connect to transfer large data sets, you can limit the Internet bandwidth used by your application. By doing so, you can reduce network fees that you pay to your Internet Service Provider and avoid having to pay for increased Internet bandwidth commitments or new contracts. In addition, all data transferred over AWS Direct Connect is charged at the reduced AWS Direct

Connect data transfer rate rather than Internet data transfer rates, which can greatly reduce your network costs.

Improved Application Performance

Applications that require predictable network performance can also benefit from AWS Direct Connect. Examples include applications that operate on real-time data feeds, such as audio or video streams. In such cases, a dedicated network connection can provide more consistent network performance than standard Internet connectivity.

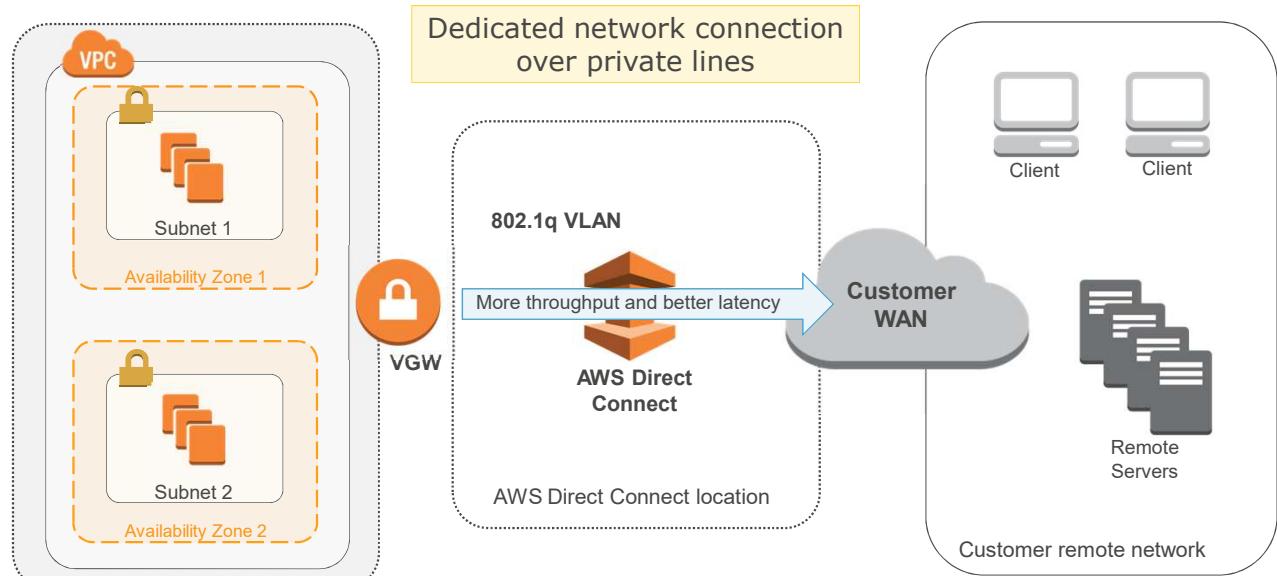
Security and Compliance

Enterprise security or regulatory policies sometimes require applications hosted on the AWS cloud to be accessed through private network circuits only. AWS Direct Connect is a natural solution to this requirement because traffic between your data center and your application flows through the dedicated private network connection.

Hybrid Cloud Architectures

Applications that require access to existing data center equipment that you own can also benefit from AWS Direct Connect. The next section discusses this use case and illustrates different scenarios that can be supported by AWS Direct Connect.

Extending On-Premises Network to AWS: AWS Direct Connect



© 2017 Amazon Web Services, Inc. or its affiliates. All rights reserved.



Benefits:

- More predictable network performance
- Reduced bandwidth costs
- 1- or 10-Gbps provisioned connections
- Supports BGP peering and routing policies

We have partnered closely with Equinix, Coresite, Eircom, TelecityGroup, and Terramark to create global Direct Connect access to every AWS region in the world. In a few of our locations, particularly in LA, NYC, and London, we have extended the capability of the service to provide access to additional IT hot spots.

Limitation:

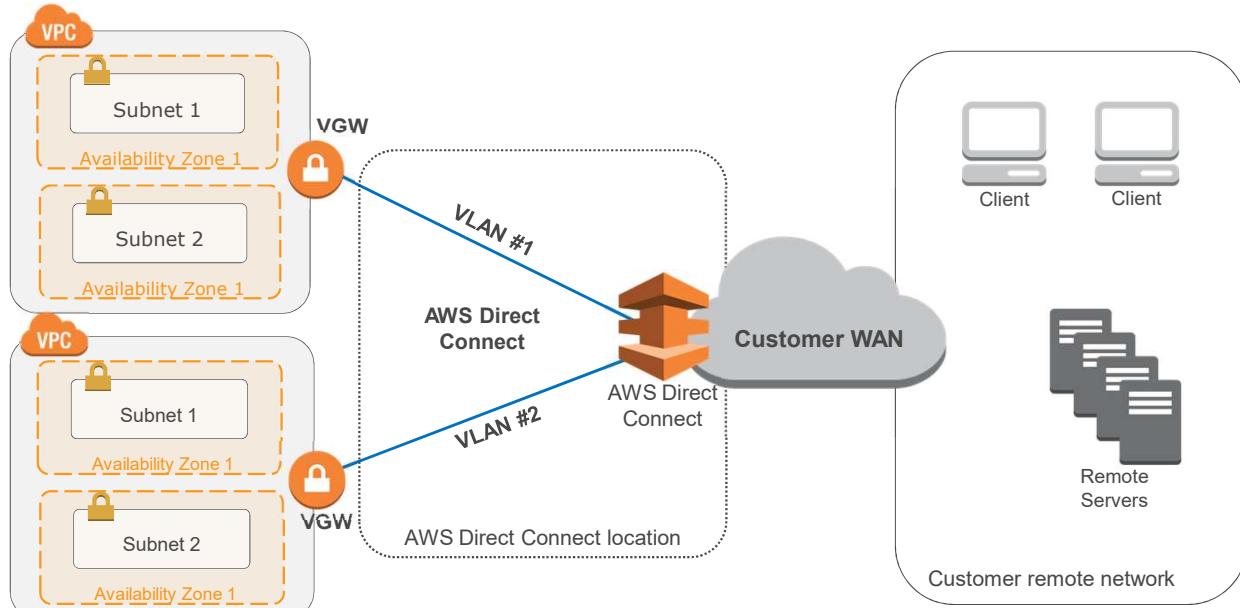
May require additional telecom and hosting provider relationships or new network circuits to be provisioned.

AWS Direct Connect makes it easy to establish a dedicated network connection from on-premises to Amazon VPC. Using AWS Direct Connect, customers can establish private connectivity between AWS and their data center, office, or colocation environment. This private connection can reduce network costs, increase bandwidth throughput, and provide a more consistent network experience than Internet-based connections can.

AWS Direct Connect lets customers establish 1-Gbps or 10-Gbps dedicated network connections (or multiple connections) between AWS networks and one of the AWS Direct Connect locations, and uses industry standard VLANs to access Amazon EC2 instances running within a VPC using private IP addresses. Customers may choose from an ecosystem of WAN service providers for integrating their AWS Direct Connect endpoint in an AWS Direct Connect location with their remote networks.

For a list of current AWS Direct locations, see
<http://aws.amazon.com/directconnect/details/>.

VPC-to-VPC: AWS Direct Connect



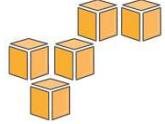
© 2017 Amazon Web Services, Inc. or its affiliates. All rights reserved.



Remember that VPC peering does not currently support cross-regions. This option can potentially reduce network costs, increase bandwidth throughput, and provide a more consistent network experience than the other VPC-to-VPC connectivity options.

A physical AWS Direct Connect connection can be divided into multiple logical connections, one for each VPC. These logical connections can then be used for routing traffic between each VPC as shown on this slide. In addition to intra-region routing, customers can connect AWS Direct Connect locations in other regions using their existing WAN providers and leverage AWS Direct Connect to route traffic between regions over their WAN backbone network.

We recommend this approach for existing AWS Direct Connect customers or customers who would like to take advantage of the reduced network costs, increased bandwidth throughput, and more consistent network experience of AWS Direct Connect. It can provide very efficient routing, because traffic can take advantage of 1-Gbps or 10-Gbps fiber connections physically attached to the AWS network in each region. Additionally, it provides the customer with the most flexibility for controlling and managing routing on their local and remote networks, as well as the potential ability to reuse AWS Direct Connect connections.



Exercise:

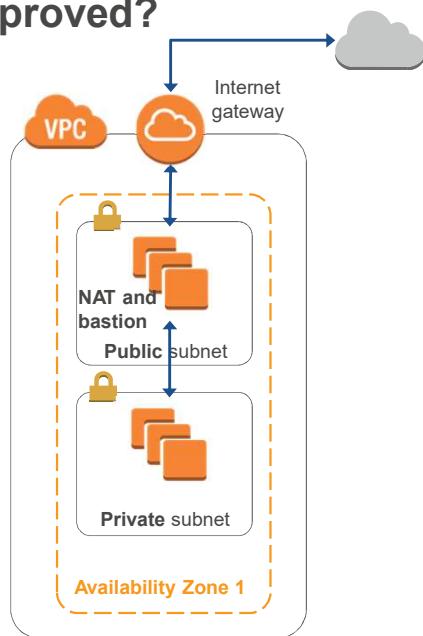
Improve This Architecture

© 2017 Amazon Web Services, Inc. or its affiliates. All rights reserved.

 **Amazon**
webservices | Academy

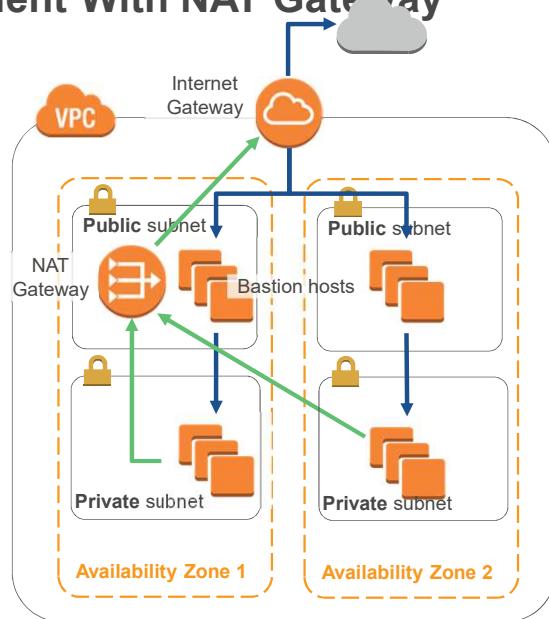
Exercise: Improve This Architecture

How Should This Be Improved?



Using the things covered in this module, how can this architecture be improved?

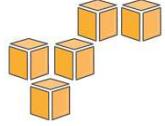
Multi-AZ Deployment With NAT Gateway



In this example, the architecture has been replicated across two Availability Zones.

Additionally, NAT Gateway is being used in place of NAT instances.

Note: NAT gateways are not always more appropriate for your solution than NAT instances on Amazon EC2. Refer to earlier slides to determine which would be right for your environment. Additionally, it's important to note that while NAT gateways are placed inside of one public subnet, it is a managed service and is therefore highly available by design and does not need to be replicated across a second Availability Zone.



Part 8

What are default VPCs and default subnets, and when should you use them?

Part 8: What are default VPCs and default subnets, and when should you use them?

What Is A Default VPC?

💡 Details about default VPCs:

- Each region in your account has a default VPC.
- Default CIDR is 172.31.0.0/16.
- If you create a VPC-based resource (Amazon EC2, Amazon RDS, Elastic Load Balancing, etc.) but don't specify a custom VPC, it will be placed in your default VPC in that region.
- Includes a default subnet, IGW, main route table connecting default subnet to the IGW, default security group, and default NACL.
- Configurable the same as other VPCs; e.g., adding more subnets.

What Is A Default Subnet?

💡 Default subnets in default VPCs:

- Created **within each Availability Zone** for each default VPC.
- **Public** subnet with a CIDR block of **/20** (4,096 IPs).
- You can convert it (and any public subnet) into a **private** subnet by removing its route to the IGW.
- When a new Availability Zone is added to a region, your default VPC in that region gets a subnet placed in the new Availability Zone (unless you've made modifications to that VPC).

When Should I Use Default VPCs And Subnets?

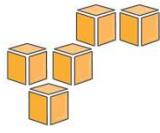
 **Recommendation:** Use default VPCs and their subnets only for **experimenting** in your AWS account.

- Default VPCs are a **quick start** solution.
 - They provide an easy way to test launching instances of your VPC-based resources, without having to set up a new VPC.
- For real-world applications, **create your own** VPCs and subnets.
 - You'll have greater control/knowledge of their configurations.
 - You can delete them and create new ones easily.

VPC Considerations And Best Practices

- Choose CIDR blocks wisely. Plan ahead.
- Use large subnets instead of a higher number of small subnets.
- Keep subnets simple and divide by Internet accessibility (public/private).
- Use Multi-AZ deployments in VPC for high availability.
- Use security groups to control traffic between resources.
- Use VPC Flow Logs to track and monitor your VPC traffic.
- Check the health of your VPN link via API calls or the AWS Management Console.

- Pick CIDR block (IP ranges) wisely by selecting something big enough for future growth (or run multiple VPCs): start with “/16.”
- The primary purpose of subnets is to divide resources based on access, so they should be leveraged that way.
- Use Multi-AZ deployments in VPC for high availability (more on that in the next module).
- Use security groups. You can specify granular security policy for Amazon EC2 instances using a security group.
- VPC Flow Logs stores traffic logs for a particular VPC, VPC subnet, or Elastic Network Interface to CloudWatch Logs, where they can be accessed by third-party tools for storage and analysis.



In review...

- 💡 Choosing a Region
- 💡 Use Multiple AZs
- 💡 Use Multiple VPCs
- 💡 Dividing VPCs Into Subnets
- 💡 Managing VPC Traffic
- 💡 Connecting Multiple VPCs
- 💡 Integrating On-prem Components
- 💡 Default VPCs and Subnets



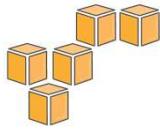
Knowledge Assessment

© 2017 Amazon Web Services, Inc. or its affiliates. All rights reserved.

In review...

- Choosing a Region
- Use Multiple AZs
- Use Multiple VPCs
- Dividing VPCs Into Subnets
- Managing VPC Traffic
- Connecting Multiple VPCs
- Integrating On-prem Components
- Default VPCs and Subnets

To complete this module, please remember to finish the corresponding knowledge assessment.



Up Next...



LAB 07 - Deploy a Web Application on AWS



Discussion 1 - Forklift an Existing Application onto AWS



CCA 3.03 - System Design for High Availability

Before you continue, please complete **Lab 07, Deploy a Web Application on AWS**. Following the labs is classroom Discussion 1, *Forklift an Existing Application onto AWS*.

Up next is **CCA 3.03 - System Design for High Availability**.

© 2017 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.

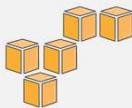
Errors or corrections? Email us at aws-course-feedback@amazon.com.

For all other questions, contact us at
<https://aws.amazon.com/contact-us/aws-training/>.

All trademarks are the property of their owners.

© 2017 Amazon Web Services, Inc. or its affiliates. All rights reserved.





CCA Unit 1 – Introduction to System Design

CCA 3.03: System Design for High Availability

Section 1: Introduction to System Design

CCA 3.01 AWS Essentials Review

CCA 3.02 Designing Your Environment

► **CCA 3.03** System Design for High Availability

Section 2: Automation and Serverless Architectures

Section 3: Well-Architected Best Practices

Section 4: Deployment and Implementation

© 2017 Amazon Web Services, Inc. or its affiliates. All rights reserved.



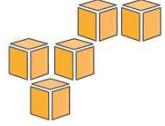
Welcome to **CCA 3.03, System Design for High Availability**.

What's In This Module

- 💡 Introduction to High Availability
- 💡 When to Scale
- 💡 Amazon Route 53
- 💡 Connecting Outside AWS

This module covers...

- Introduction to High Availability
- When to Scale
- Amazon Route 53
- Connecting Outside AWS



Part 1

What is High Availability?

© 2017 Amazon Web Services, Inc. or its affiliates. All rights reserved.

 Academy

The Amazon Web Services logo, featuring the word "amazon" in lowercase with a yellow arrow above it, followed by "webservices" in smaller letters. To the right, the word "Academy" is written in a grey sans-serif font.

Part 1: What is High Availability?

What Is High Availability?

High availability (HA) is about ensuring that your application's **downtime is minimized** as much as possible, without the need for human intervention.

Levels of Availability:

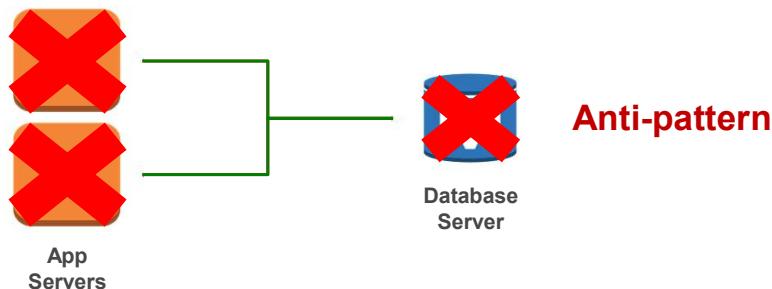
	Percent of Uptime	Max Downtime per Year	Equivalent Downtime per Day
1 Nine	90%	36.5 days	2.4 hrs
2 Nines	99%	3.65 days	14 min
3 Nines	99.9%	8.76 hrs	86 sec
4 Nines	99.99%	52.6 min	8.6 sec
5 Nines	99.999%	5.25 min	.86 sec

Availability refers to the amount of time your system is in a functioning condition. In general terms, your availability is referred to as 100% minus your system's downtime. Since events which may disrupt your system's availability are never entirely predictable, there are always ways to make an application more available, however, improving availability typically leads to increased cost. When considering how to make your environment more available, it's important to balance the cost of the improvement with the benefit to your users. HA might mean that you ensure your application is always alive/reachable or does it mean that the app is servicing requests within an acceptable level of performance?

Best Practice: Avoid Single Points Of Failure

*Assume everything fails,
and design backwards.*

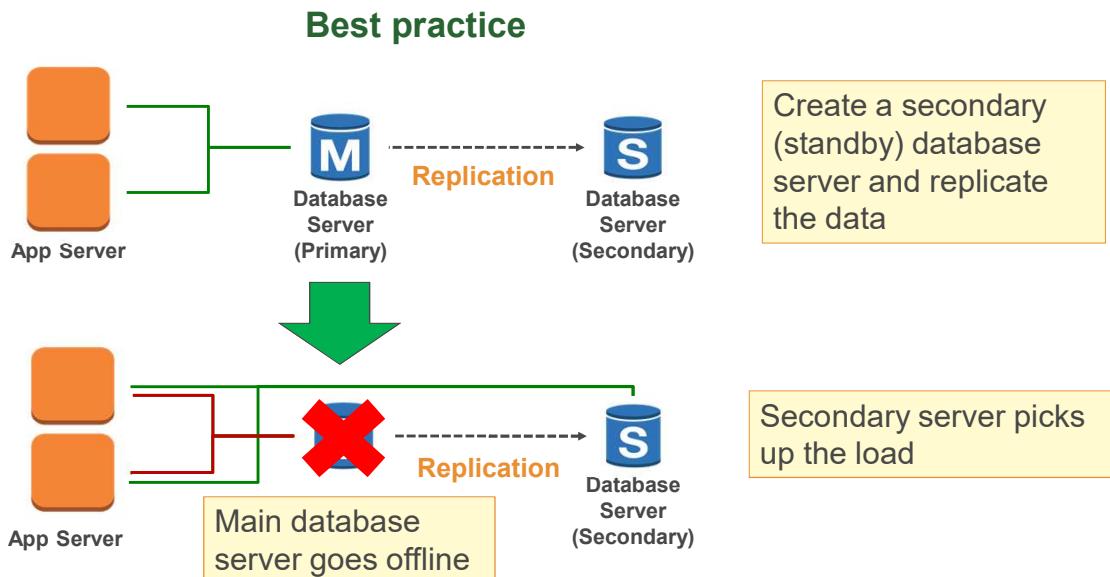
Implement redundancy where possible in order to prevent single failures from bringing down an entire system.



Wherever possible, eliminate single points of failure from your architectures. This doesn't mean every component you have has to be duplicated at all times, though. Depending on your downtime SLAs, you can leverage automated solutions that only launch when needed, or a managed service where AWS automatically replaces malfunctioning underlying hardware for you.

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.

Best Practice: Avoid Single Points Of Failure

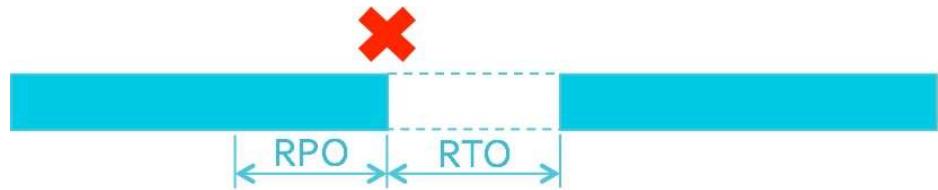


A common way to solve the problem of a single database server is to create a secondary server (standby) and replicate the data.

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. This goes back to Best Practice #2: treat resources as disposable, and design your applications to support changes in hardware.

High Availability: Driven By Your Requirements

- Recovery Time Objective (RTO)
How quickly must the system recover?
- Recovery Point Objective (RPO)
How much data can you afford to lose?
- How much money do you need to invest to meet those objectives?



Essentially, non-functional requirements define the design of your infrastructure. A highly available and fault-tolerant environment expands multi-AZ and cross-regions; however, there are costs associated with that design. We discussed in the *Designing for Cost* module that pricing can differ from region to region.

High Availability Factors

- **Fault tolerance:**
The **built-in redundancy** of an application's components.
- **Recoverability:**
The process, policies, and procedures related to **restoring service** after a catastrophic event.
- **Scalability:**
The ability of an application to **accommodate growth** without changing design.

Fault tolerance, recoverability, and scalability are three primary factors that determine the overall availability of your application.

Fault tolerance is often confused with high availability, but fault tolerance refers to the built-in redundancy of an application's components. Does it avoid single points of failure? We will talk more about fault tolerance within this module.

Recoverability is often overlooked as a component of availability. In the event a natural disaster makes one or more of your components unavailable, or destroys your primary data source, are you able to restore service quickly and without lost data? We will talk more about specific disaster recovery strategies in a later module.

Scalability is a question of how quickly your application's infrastructure can respond to increased capacity needs to ensure your application is available and performing within your required standards. It does not guarantee availability, but is one part of your application's availability.

On-Premises HA vs. HA On AWS

In **traditional, on-premises** IT, high availability:

- Is very expensive
- Is suitable only for absolutely mission-critical applications

AWS **expands availability and recoverability** options by enabling you to use:

- Multiple servers
- Isolated redundant data centers within each Availability Zone
- Multiple Availability Zones within each region
- Regions across the globe
- Fault-tolerant services

High Availability And AWS Regions

- Multi-region deployment
- increases:
 - Availability
 - Cost
 - Complexity

Default to a **single region**, unless multi-region deployment is necessary.

If you use a single region, **multi-AZ is the bare minimum HA solution.**



AWS Services And High Availability

Inherently HA services

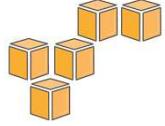
- ✓ Amazon S3 and Amazon Glacier
- ✓ DynamoDB
- ✓ Amazon CloudFront
- ✓ Amazon SWF
- ✓ Amazon SQS
- ✓ Amazon SNS
- ✓ Amazon SES
- ✓ Amazon Route 53
- ✓ Elastic Load Balancing
- ✓ IAM
- ✓ Amazon CloudWatch
- ✓ Amazon CloudSearch
- ✓ AWS Data Pipeline
- ✓ Amazon Kinesis
- ✓ Auto Scaling
- ✓ Amazon Elastic File System
- ✓ AWS CloudFormation
- ✓ Amazon WorkMail
- ✓ AWS Directory Service
- ✓ AWS Lambda
- ✓ Amazon EBS
- ✓ Amazon RDS

HA with the right architecture

- Amazon EC2
- Amazon VPC
- Amazon Redshift
- Amazon ElastiCache
- AWS Direct Connect

****Not all services are listed here.**

While some AWS services are not HA inherently, they can be designed to be HA. Throughout the rest of the course, ways to accomplish this will be explored.



Part 2
Elastic Load Balancing (ELB)

© 2017 Amazon Web Services, Inc. or its affiliates. All rights reserved.

 **Amazon**
webservices | Academy

Part 2: Elastic Load Balancing (ELB)

Elastic Load Balancing (ELB)



A managed load balancing service that **distributes incoming application traffic** across multiple Amazon EC2 instances.

The foundation of the web tier includes the use of Elastic Load balancers in the architecture. These load balancers not only send traffic to EC2 instances, but can also send metrics to Amazon Cloudwatch, a managed monitoring service provided. The metrics from EC2 and the ELB can act as triggers—so that if we notice a particularly high latency or that our servers are becoming over utilized, we're able to take advantage of Auto Scaling to add more capacity to our web server fleet.

AWS Load Balancing Solution: Elastic Load Balancing

- Elastic Load Balancing:
- Distributes load between instances.
- Recognizes and responds to unhealthy instances.
- Can be public or internal-facing.
 - Uses HTTP, HTTPS, and TCP protocols.
 - Each load balancer is given a public DNS name.
 - ▶ Internet-facing load balancers have DNS names which publicly resolve to the public IP addresses of the load balancer's nodes.
 - ▶ Internal load balancers have DNS names which publicly resolve to the private IP addresses of the load balancer's nodes.

DIY Load Balancer vs. Elastic Load Balancing

DIY	Elastic Load Balancing
<ul style="list-style-type: none">• Un-managed (managed by you)• You create a load balancer on Amazon EC2 that fits your requirements• Scaling must be handled by you	<ul style="list-style-type: none">• Managed by AWS• Usually the most cost-efficient solution• Automatically scaled

Why Choose Elastic Load Balancing?

Elastic Load Balancing provides the following features:

- Health checks
- Cross-zone load balancing
- Proxy Protocol
- Sticky sessions
- Connection draining



Elastic Load Balancing offers features that are ready for you to take advantage of. DIY takes a certain amount of effort to provide these features.

Health Checks

To discover the availability of your Amazon EC2 instances, the load balancer periodically sends pings, attempts connections, or sends requests to test the Amazon EC2 instances. These tests are called *health checks*. Each registered Amazon EC2 instance must respond to the target of the health check with an HTTP status code 200 to be considered healthy by your load balancer.

Cross-Zone Load Balancing

You can set up your load balancer to distribute incoming traffic across more than one Availability Zone or multiple Availability Zones within a region. By default, the load balancer node routes traffic to back-end instances within the same Availability Zone. To ensure that your back-end instances are able to handle the request load in each Availability Zone, it is important to have approximately equivalent numbers of instances in each zone. For example, if you have ten instances in Availability Zone us-east-1a and two instances in us-east-1b, the traffic will still be equally distributed between the two Availability Zones. As a result, the two instances in us-east-1b will have to serve the same amount of traffic as the ten instances in us-east-1a.

If you want the request traffic to be routed evenly across all back-end instances, regardless of the Availability Zone that they may be in, enable cross-zone load balancing on your load balancer. Cross-zone load balancing allows each load balancer node to route request across multiple Availability Zones for higher fault tolerance.

Proxy Protocol

The Proxy Protocol header helps you identify the IP address of a client when you use a load balancer configured for TCP/SSL connections. Because load balancers intercept traffic between clients and your back-end instances, the access logs from your back-end instance contain the IP address of the load balancer instead of the originating client. When Proxy Protocol is enabled, the load balancer adds a human-readable format header that contains the connection information, such as the source IP address, destination IP address, and port numbers of the client. The header is then sent to the back-end instance as a part of the request. You can parse the first line of the request to retrieve your client's IP address and the port number.

Sticky Sessions

By default, a load balancer routes each request independently to the application instance with the smallest load. However, you can use the *sticky session* feature (also known as session affinity), which enables the load balancer to bind a user's session to a specific application instance. This ensures that all requests coming from the user during the session will be sent to the same application instance.

Sticky Sessions With Elastic Load Balancing

Sticky sessions enable the load balancer to **bind a user's session** to a specific instance for the life of a session.

- Duration-based session stickiness uses a **load balancer-generated cookie** to track the application instance for each request.
- Application-controlled session stickiness uses a cookie to associate the session with the original server that handled the request.

You can use sticky sessions for only HTTP/HTTPS load balancer listeners.

By default, a load balancer routes each request independently to the application instance with the smallest load. However, you can use sticky session features (also known as session affinity) which enables the load balancer to bind a user's session to a specific application instance.

Duration-based session stickiness

The load balancer uses a special load balancer-generated cookie to track the application instance for each request. When the load balancer receives a request, it first checks to see whether this cookie is present in the request. If so, the request is sent to the application instance specified in the cookie. If there is no cookie, the load balancer chooses an application instance based on the existing load balancing algorithm. A cookie is inserted into the response for binding subsequent requests from the same user to that application instance. The stickiness policy configuration defines a cookie expiration, which establishes the duration of validity for each cookie. The cookie is automatically updated after its duration expires.

Application-controlled session stickiness

The load balancer uses a special cookie to associate the session with the original server that handled the request, but follows the lifetime of the application-generated cookie corresponding to the cookie name specified in the policy configuration. The load balancer only inserts a new stickiness cookie if the application response includes a new application cookie.

The load balancer stickiness cookie does not update with each request. If the application cookie is explicitly removed or expires, the session stops being sticky until a new application cookie is issued.

Connection Draining For Your Load Balancer

Enabling *connection draining* causes the load balancer to **stop sending new requests** to the back-end instances when instances are **de-registering** or **become unhealthy**.

Why is this important?

You can perform maintenance without affecting your end users.

When you enable *connection draining* on a load balancer, any back-end instances that you deregister will complete requests that are in progress before deregistration.

Likewise, if a back-end instance fails health checks, the load balancer will not send any new requests to the unhealthy instance but will allow existing requests to be completed, while ensuring that in-flight requests continue to be served. This means that you can perform maintenance such as deploying software upgrades or replacing back-end instances without affecting your customers' experience.

Connection draining is also integrated with Auto Scaling, which makes it even easier to manage the capacity behind your load balancer. When connection draining is enabled, Auto Scaling will wait for outstanding requests to be completed before terminating instances.

You can enable connection draining through:

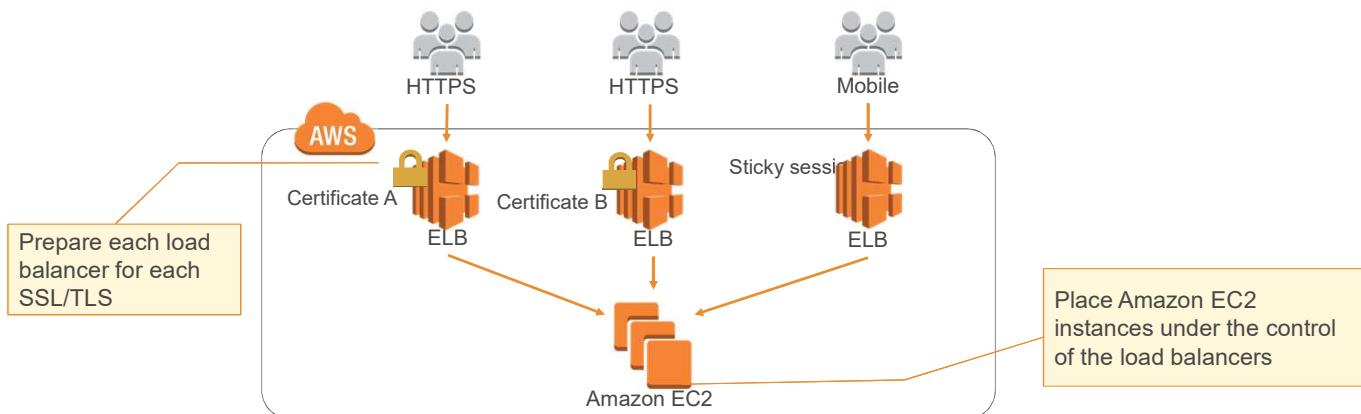
- AWS Management Console
- API
- Command Line Interface (CLI)
- AWS CloudFormation

For more information, see

<http://docs.aws.amazon.com/ElasticLoadBalancing/latest/DeveloperGuide/how-elb-works.html>

Cloud Design Pattern: Multi-Load Balancer Pattern

Assign multiple load balancers with different settings for the types of devices that are accessing the web application.



Problem to be solved:

When a web application is multi-device compatible, there will be access from PCs, mobile phones, and smart phones. At this time, if it is necessary to perform a setup such as for SSL/TLS or to assign sessions for individual access devices, and if the setup is performed by the EC2 instances themselves, any change to the settings would become extremely laborious as the number of servers increases.

Cloud Pattern:

In groups of virtual servers, you can solve this problem by assigning multiple virtual load balancers with different settings. That is, rather than modifying the servers, you can change the behavior relative to access by the different devices by changing the virtual load balancer for routing the access. For example, you can apply this to settings such as for sessions, health checks, and HTTPS.

Implementation:

Assign multiple virtual load balancers to a single Amazon EC2 instance. You can use the SSL Termination function of the load balancer to perform the HTTPS (SSL) process.

- Place an Amazon EC2 instance under the control of the load balancers.
- Prepare load balancers with different settings for sessions, health checks, HTTPS, and the like, and switch between them for the same Amazon EC2 instance.

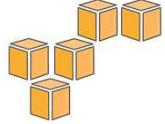
Benefits:

- The behavior (on the load balancer level) for mobile sites and PC sites can be different, even when using the same Amazon EC2 instance.
- Even when multiple SSLs (HTTPS) are used by the same Amazon EC2 instance, you can

prepare load balancers for each SSL (HTTP).

Cautions:

- Note that when you cut off an Amazon EC2 instance from a load balancer to perform maintenance, for example, you have to cut off the Amazon EC2 instance from all of the load balancers.
- When you use the SSL Termination function of a load balancer, the Amazon EC2 instance side will be able to receive requests via HTTP, making it difficult to evaluate the HTTPS connection by the applications.



Exercise:

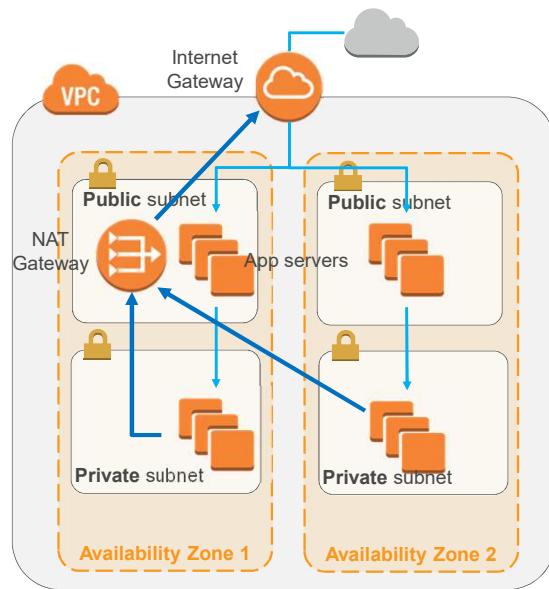
Improve This Architecture

© 2017 Amazon Web Services, Inc. or its affiliates. All rights reserved.

 **Amazon**
webservices | Academy

Exercise: Improve This Architecture

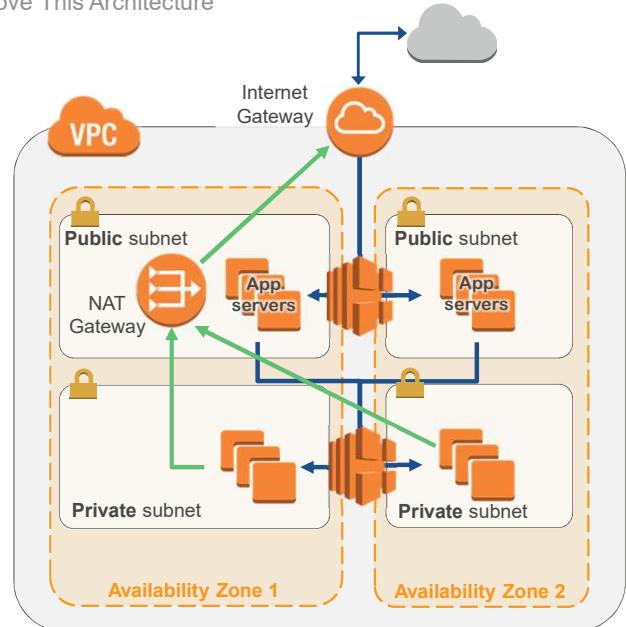
How Can The Availability Of This Environment Be Improved?



HA With Load Balancing

- **Internet-facing** load balancer
 - Deployed in **public** subnet
 - Balancing traffic to app servers in two Availability Zones

- **Internal** load balancer
 - Intranet-facing or internal
 - Balancing traffic from NATs to private instances



When you create your load balancer in a VPC, you can make your load balancer internal (private) or Internet-facing (public). When you make your load balancer internal, a DNS name will be created, and it will contain the private IP address of your load balancer. An internal load balancer is not exposed to the Internet. When you make your load balancer Internet-facing, a DNS name will be created with the public IP address. The DNS records are publicly resolvable in both cases.

Elastic IP Addresses Provide Greater Fault Tolerance

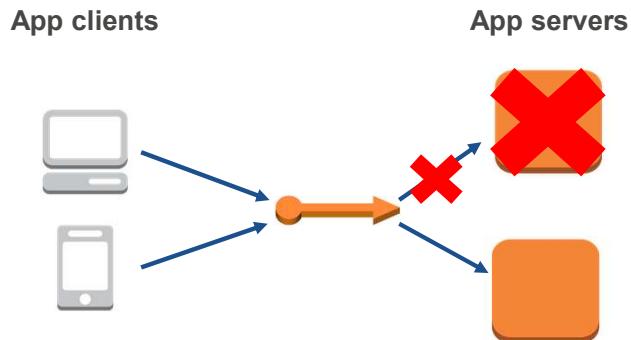


Elastic IP Addresses:

- Are **static IP addresses** designed for dynamic cloud computing.
- Can be attached to **Amazon EC2 instances**.
- Enable you to **mask the failure** of an instance or software by allowing your users and clients to **use the same IP address with replacement resources**.

The foundation of the web tier includes the use of Elastic Load balancers in the architecture. These load balancers not only send traffic to EC2 instances, but can also send metrics to Amazon Cloudwatch, a managed monitoring service provided. The metrics from EC2 and the ELB can act as triggers—so that if we notice a particularly high latency or that our servers are becoming over utilized, we're able to take advantage of Auto Scaling to add more capacity to our web server fleet.

Using Elastic IP Addresses to Enable High Availability

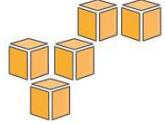


If one instance goes down, clients can use the same IP address to reach the replacement instance.

Elastic Load Balancers

What about HA across regions?

The answer is no. Load balancers only work across Availability Zones within a region.



Part 3

Amazon Route 53

© 2017 Amazon Web Services, Inc. or its affiliates. All rights reserved.



Part 3: Amazon Route 53

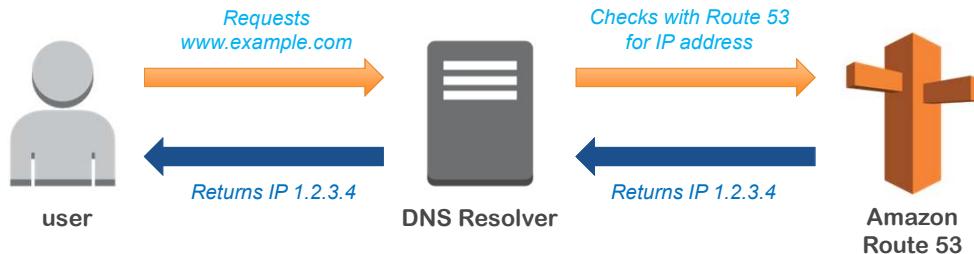
Amazon Route 53: AWS's HA Domain Name Service (DNS)

Amazon Route 53 is **an authoritative DNS service from AWS** with the following characteristics:

- DNS translates domain names (like www.amazon.com) into IP addresses .
- The name refers to the fact that DNS servers respond to queries on port 53.



Amazon Route 53 DNS Resolution



Amazon Route 53

Reliable

- Redundant locations
- Backed with 100% Service Level Agreement (SLA)

Fast

- Worldwide anycast network
- Fast propagation of changes

Integrated with AWS

- ELB-Alias Queries
- Latency-based routing

Easy to Use

- Console
- Programmatic API
- Domain name management

Cost-Effective

- Inexpensive rates
- Pay-as-you-go model

Flexible

- Geolocation routing
- Weighted round robin
- Self-aliasing

Amazon Route 53 is a Tier 0 service, meaning it is designed and is maintained with the highest level of availability in mind.

What Kinds Of Routing Does Amazon Route 53 Support?

Simple routing: Single server environments.

Weighted round robin: Assign weights to resource record sets to specify the frequency.

Latency-based routing: Helps to improve your global applications.

Health check and DNS failover: Fail over to a backup site if your primary site becomes unreachable.

Geolocation routing: Specify **geographic locations** by continent, by country, or by state in the United States.

Simple routing (round robin) uses a simple routing policy when you have a single resource that performs a given function for your domain, for example, one web server that serves content for the example.com website. In this case, Amazon Route 53 responds to DNS queries based only on the values in the resource record set, for example, the IP address in an A record.

Weighted round robin allows you to assign weights to resource record sets in order to specify the frequency with which different responses are served. You may want to use this capability to do A/B testing, sending a small portion of traffic to a server on which you've made a software change. For instance, suppose you have two record sets associated with one DNS name: one with weight 3 and one with weight 1. In this case, 75% of the time Amazon Route 53 will return the record set with weight 3 and 25% of the time Amazon Route 53 will return the record set with weight 1. Weights can be any number between 0 and 255.

Latency-based routing (LBR) helps you improve your application's performance for a global audience. LBR works by routing your customers to the AWS endpoint (e.g., Amazon EC2 instances, EIPs or load balancers) that provides the fastest experience based on actual performance measurements of the different AWS regions where your application is running.

Geolocation routing lets you choose the resources that serve your traffic based on the geographic location of your users (the origin of DNS queries). When you use geolocation routing, you can localize your content and present some or all of your website in the language of your users. You can also use geolocation routing to restrict distribution of

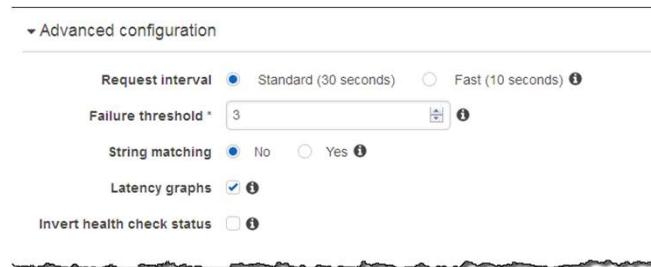
content to only the locations in which you have distribution rights. Another possible use is for balancing load across endpoints in a predictable, easy-to-manage way, so each end-user location is consistently routed to the same endpoint.

With *DNS failover*, Amazon Route 53 can help detect an outage of your website and redirect your end users to alternate locations where your application is operating properly. When you enable this feature, Amazon Route 53 health-checking agents will monitor each location/endpoint of your application to determine its availability. You can take advantage of this feature to increase the availability of your customer-facing application.

Amazon Route 53 DNS Failover

Improve availability of your applications running on AWS by:

- Configuring backup and failover scenarios for your own applications.
- Enabling highly available multi-region architectures on AWS.
- Improving health checks by combining multiple health checks, domain name-based health checks, string matching, specifying the request interval, and more.



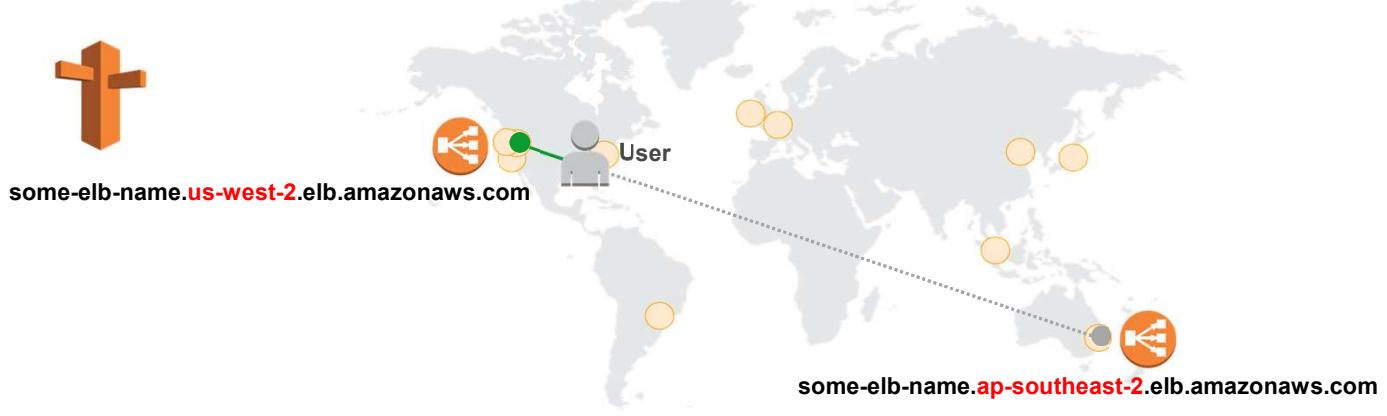
© 2017 Amazon Web Services, Inc. or its affiliates. All rights reserved.

 **amazon**
web services | Academy

Route 53 lets you track the health status of your resources and take action when an error occurs.

String Matching: the health check looks for a particular string in the page body (within the first 5kb of content). You could use this to confirm whether the DB instance is working (by matching on a string that the page would only contain if it successfully connects to the DB). Failover to the static backup site would then be triggered if either the web server goes down or the DB instance goes down (or if the web server hangs or returns garbage content for some reason).

Use Case: Multi-Region Deployment



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

© 2017 Amazon Web Services, Inc. or its affiliates. All rights reserved.

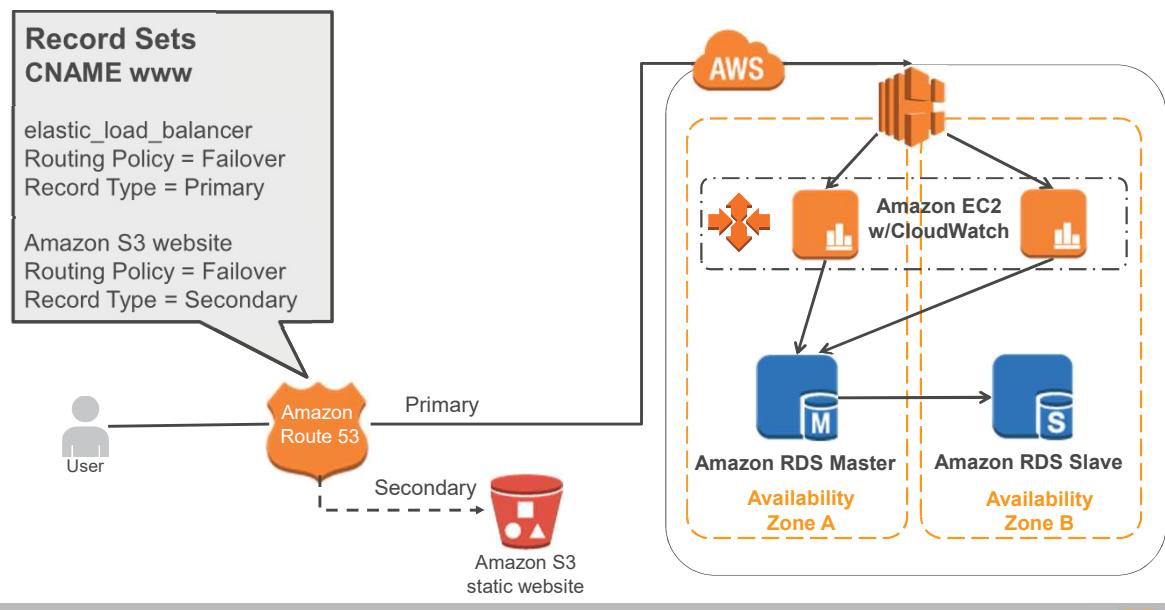


With Amazon Route 53, the user is automatically directed to the Elastic Load Balancing load balancer that's also in the United States, because it's closer to the user.

Benefits:

- Latency-based routing to region
- Load balancing routing to AZ

Typical Architecture



© 2017 Amazon Web Services, Inc. or its affiliates. All rights reserved.

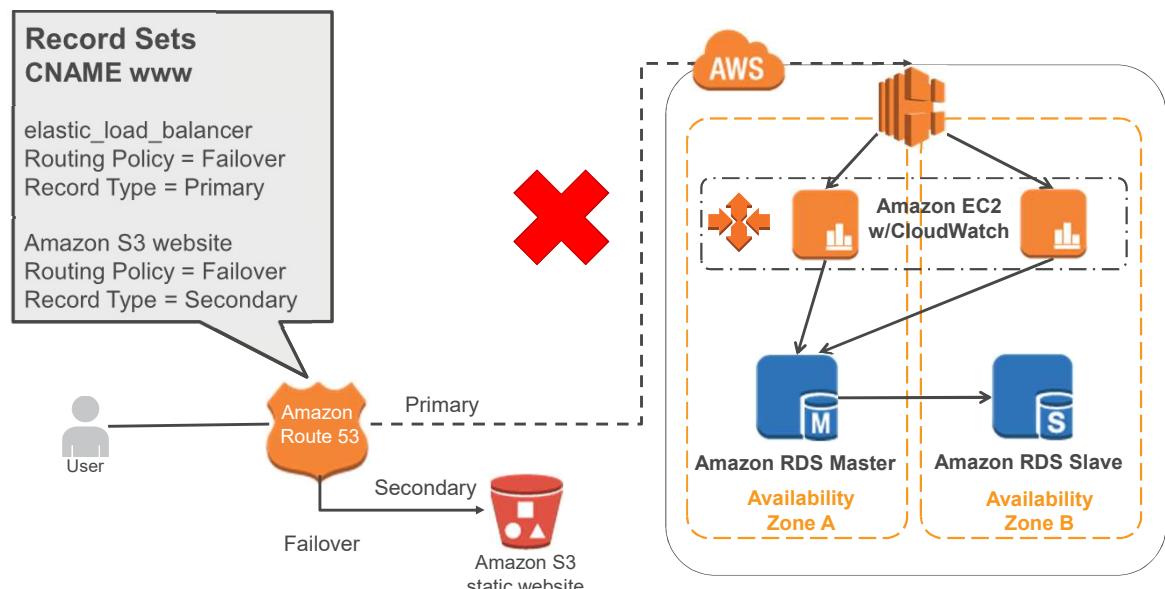


The slide shows how DNS failover works in a typical architecture for a multi-tiered web application. Route 53 passes traffic on to an Elastic Load Balancer, which then passes traffic on to a fleet of autoscaling EC2 instances.

We can do the following with Route 53 to ensure high availability:

1. Create two DNS records for the CNAME www with a routing policy of Failover Routing. The first record is the primary and points to the ELB for your web application. The second record is the “Secondary” route policy and points to your static S3 website
2. Use Route 53 health checks to make sure the primary is up. If it is, all traffic will default to your web application stack.

Typical Architecture



© 2017 Amazon Web Services, Inc. or its affiliates. All rights reserved.



The slide shows how DNS failover works in a typical architecture for a multi-tiered web application. Route 53 passes traffic on to an Elastic Load Balancer, which then passes traffic on to a fleet of autoscaling EC2 instances.

We can do the following with Route 53 to ensure high availability:

1. Create two DNS records for the CNAME www with a routing policy of Failover Routing. The first record is the primary and points to the ELB for your web application. The second record is the “Secondary” route policy and points to your static S3 website
2. Use Route 53 health checks to make sure the primary is up. If it is, all traffic will default to your web application stack.

Getting Started With Amazon Route 53

We offer guides for each of the following tasks on our website:

- Creating a domain that uses Amazon Route 53 as the DNS service.
- Migrating an existing domain to Amazon Route 53.
- Creating a subdomain that uses Amazon Route 53 without migrating the parent domain.
- Migrating a subdomain to Amazon Route 53 without migrating the parent domain.
- Creating alias record sets for Elastic Load Balancing.

(Links to each guide are in the notes section of this slide)

- Create a domain that uses Amazon Route 53 as the DNS service: see
<http://docs.aws.amazon.com/Route53/latest/DeveloperGuide/>
- Migrate an existing domain to Amazon Route 53: see
<http://docs.aws.amazon.com/Route53/latest/DeveloperGuide/MigratingDNS.html>
- Create a subdomain that uses Amazon Route 53 without migrating the parent domain: see
<http://docs.aws.amazon.com/Route53/latest/DeveloperGuide/CreatingNewSubdomain.html>
- Migrate a subdomain to Amazon Route 53 without migrating the parent domain: see
<http://docs.aws.amazon.com/Route53/latest/DeveloperGuide/MigratingSubdomain.html>
- Create alias record sets for Elastic Load Balancing:
<http://docs.aws.amazon.com/Route53/latest/DeveloperGuide/resource-record-sets-choosing-alias-non-alias.html>

Alta Veículos Needed A Highly Available Environment

“

Our application availability increased by 30% while costs were reduced 25%

Ronie Redi
IT Manager, Alta Veículos



”

- Alta needed to improve availability and reliability of its email services, and improve durability of backups
- Used Route 53, Amazon ELB, Amazon EBS, Amazon EC2, Amazon S3 and Amazon Glacier to migrate its email platform (Zimbra) and host its Enterprise Resource Planning (ERP) backups
- Application availability improvement increased customer satisfaction. Costs saved were reinvested in the business.

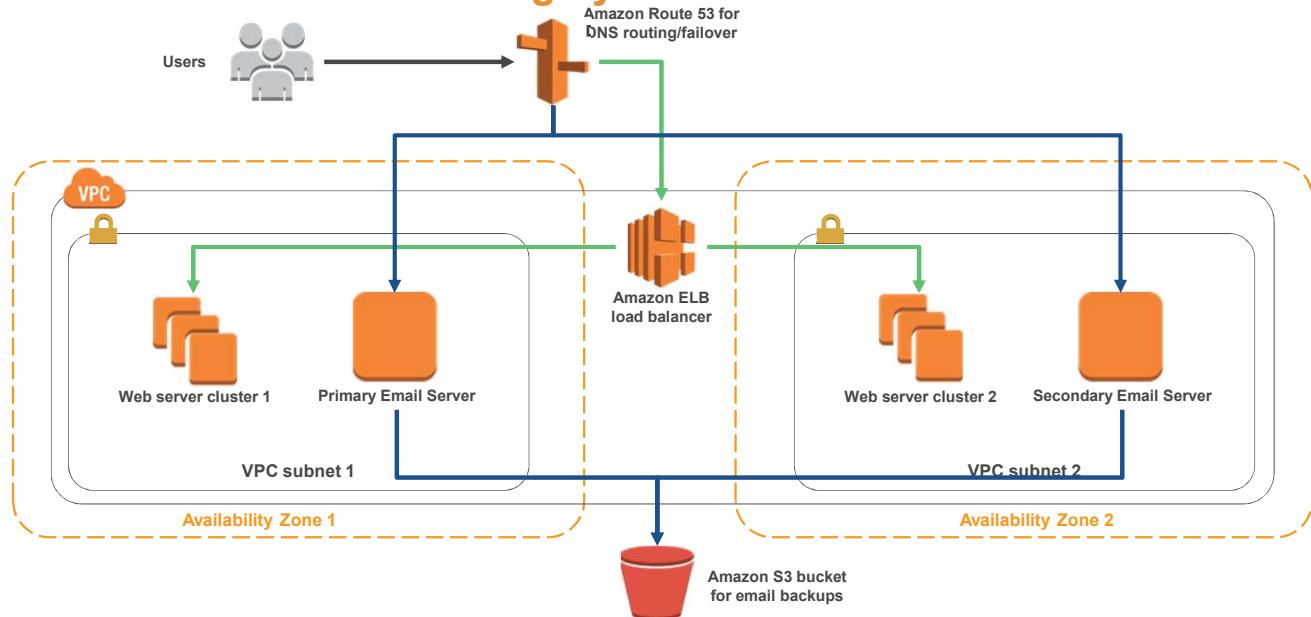
Alta Veículos offers new and pre-owned vehicles, fleet services, parts, accessories, garage services and financing.

© 2017 Amazon Web Services, Inc. or its affiliates. All rights reserved.



Alta Veículos is a Brazilian company that came to AWS because it needed an IT infrastructure that was more reliable, available, and scalable in order to meet the needs of its growing userbase.

Alta Veículos Needed A Highly Available Environment



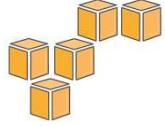
© 2017 Amazon Web Services, Inc. or its affiliates. All rights reserved.



Alta Veículos's environment has users access the portal they need via Amazon Route 53. It's split between two Availability Zones (both in the São Paulo region). Their VPC has two subnets to increase availability.

1. If users are trying to access their web servers, Route 53 routes that request to their Amazon ELB load balancer,
2. From there, the load balancer distributes the requests evenly between the two web server clusters.
3. Users trying to access the email servers are routed by Route 53 to the primary email server first, but in the event that that server is unavailable, requests are failed over to their secondary email server.
4. The email servers use Amazon S3 to backup and restore emails.

For more on this case study, see: <http://aws.typepad.com/brasil/case-studies/>

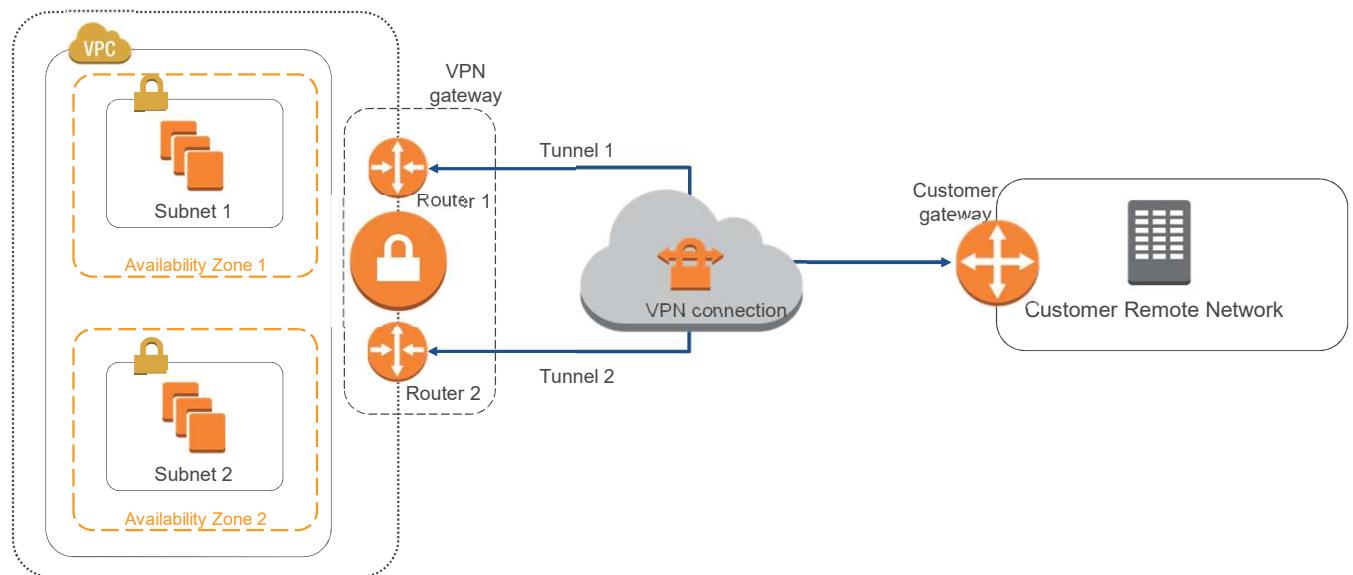


Part 4

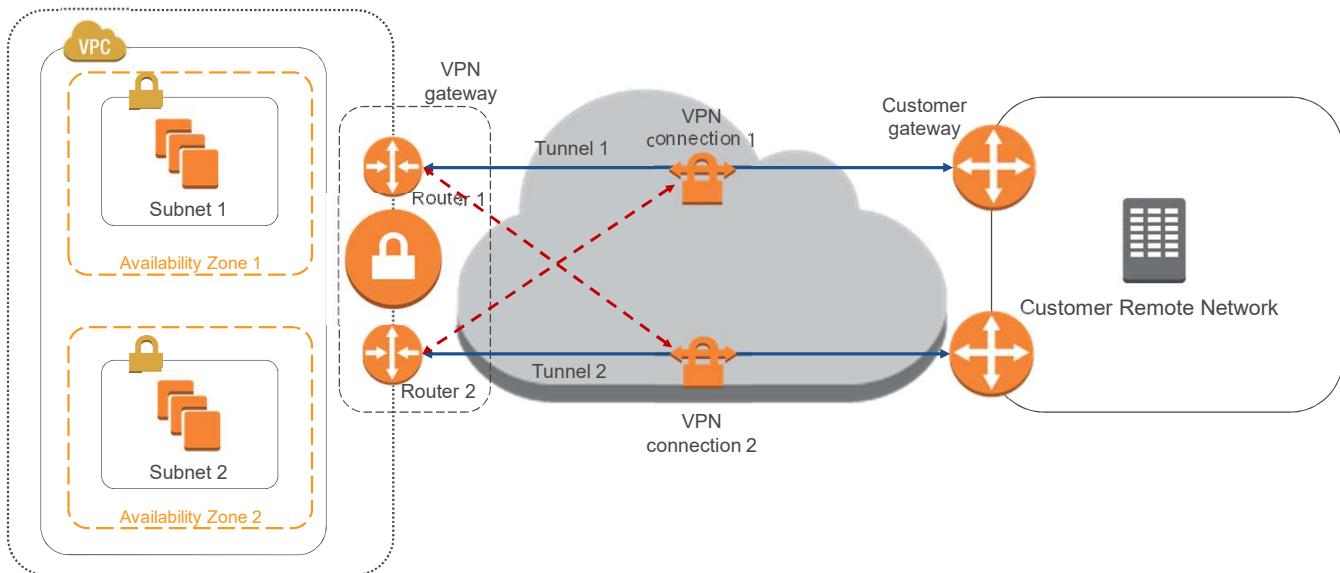
What about connections to components outside of AWS?

Part 4: What about connections to components outside of AWS?

VPN Connection: Is This Highly Available?



Adding Connection Redundancy With AWS Hardware VPNs



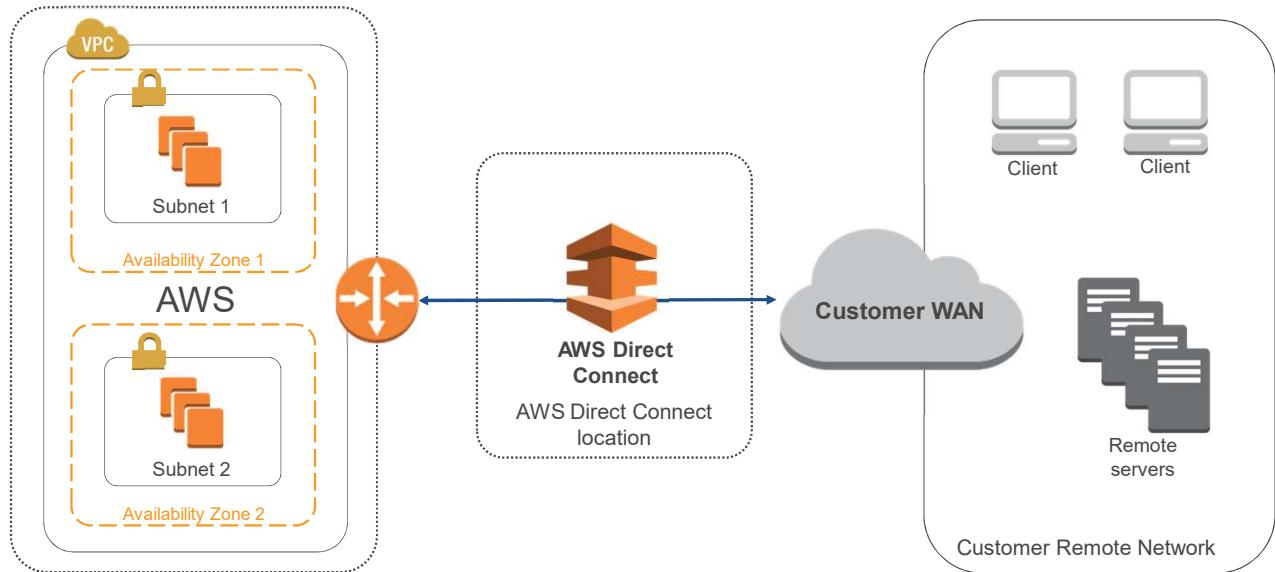
© 2017 Amazon Web Services, Inc. or its affiliates. All rights reserved.



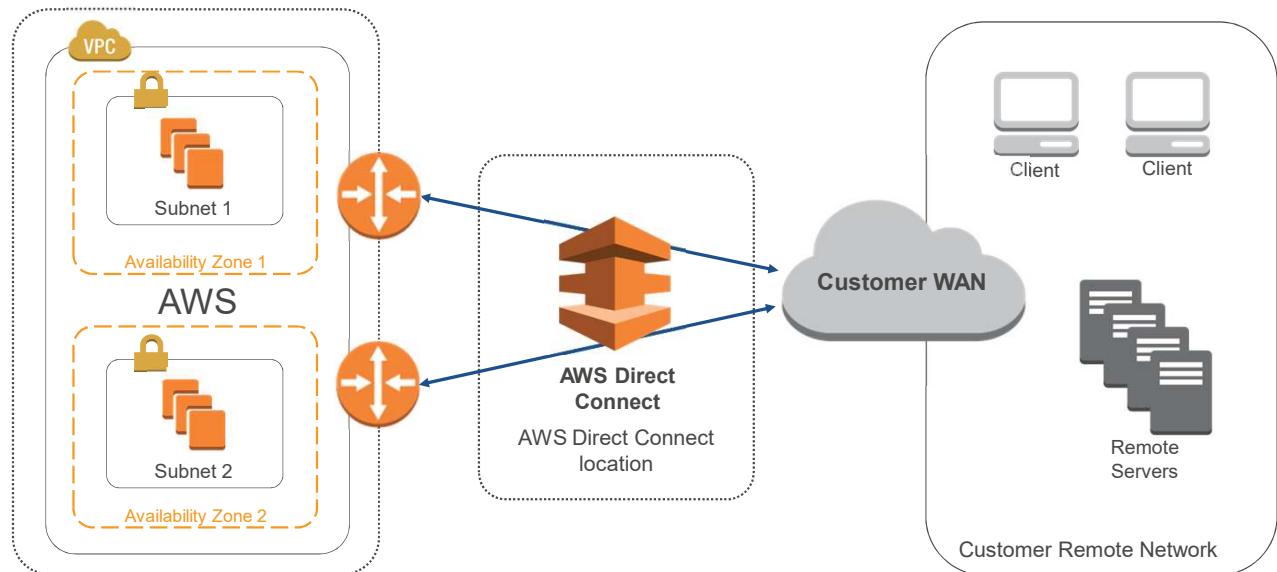
To protect against a loss of connectivity in case your customer gateway becomes unavailable, you can set up a second VPN connection to your VPC and virtual private gateway by using a second customer gateway. By using redundant VPN connections and customer gateways, you can perform maintenance on one of your customer gateways while traffic continues to flow over the second customer gateway's VPN connection. To establish redundant VPN connections and customer gateways on your network, you need to set up a second VPN connection. The customer gateway IP address for the second VPN connection must be publicly accessible.

Dynamically routed VPN connections use the Border Gateway Protocol (BGP) to exchange routing information between your customer gateways and the virtual private gateways. Statically routed VPN connections require you to enter static routes for the network on your side of the customer gateway. BGP-advertised and statically entered route information allow gateways on both sides to determine which tunnels are available and reroute traffic if a failure occurs. We recommend that you configure your network to use the routing information provided by BGP (if available) to select an available path. The exact configuration depends on the architecture of your network.

AWS Direct Connect: Is This Highly Available?



Direct Connect Single Router And Dual Ports



© 2017 Amazon Web Services, Inc. or its affiliates. All rights reserved.



The slide shows a default configuration where you establish 2 physical links from your router to Different AWS Routers, providing redundancy.

Redundancy is based on the BGP policies. Customers can set BGP policies (local preference and MED) that causes one link to be preferred for both inbound and outbound traffic. If the local preference and MED values are the same between the two links, traffic will be roughly load balanced across both.

For more information, including steps for how to set up redundant Direct Connect links, see:
<http://docs.aws.amazon.com/directconnect/latest/UserGuide/getstarted.html>



CCA Lab-08

Making Your Environment Highly Available

(Approx. 60 minutes)

© 2017 Amazon Web Services, Inc. or its affiliates. All rights reserved.

 Academy

Lab 8: Scenario

In this lab, you will take a web application environment and make it **highly available**. You will also create a **private instance layer**.

Here are the services you will be using:



Amazon EC2



Amazon VPC



Amazon EC2
AMI



Elastic Load
Balancing



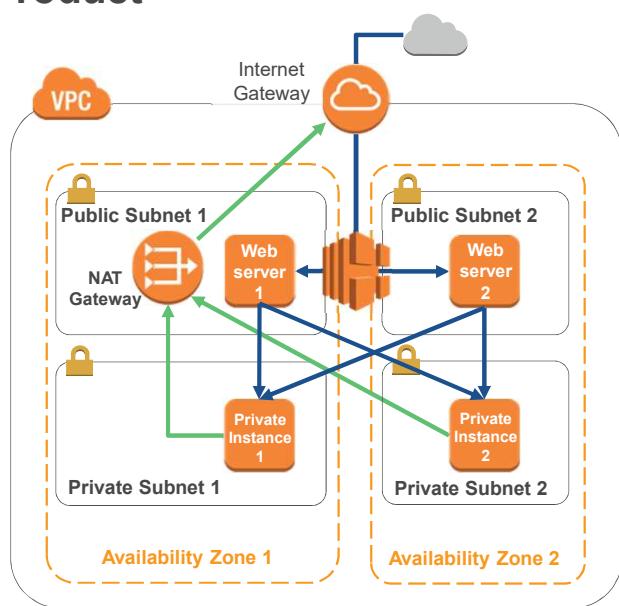
NAT
Gateway

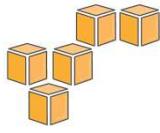
Lab 8: Tasks

Here are the tasks the lab guide will walk you through:

1. Inspect the resources created automatically for your lab.
2. Start your web server's PHP application
3. Create an AMI of your web server.
4. Create a second public subnet in a second AZ and launch a new web server in that subnet.
5. Create a load balancer and attach it to both subnets.
6. Create two private subnets with two instances inside of them.
7. Create a NAT Gateway and route traffic from private instances to it.
8. Test the private instances' connections to the NAT Gateway.

Lab 8: Final Product





In review...

- 📦 Introduction to High Availability
- 📦 When to Scale
- 📦 Amazon Route 53
- 📦 Connecting Outside AWS



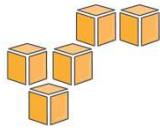
Knowledge Assessment

© 2017 Amazon Web Services, Inc. or its affiliates. All rights reserved.

In review...

- Introduction to High Availability
- When to Scale
- Amazon Route 53
- Connecting Outside AWS

To complete this module, please remember to finish the corresponding knowledge assessment.



Up Next...



LAB 08 - Making Your Environment Highly Available



Section 2 - Automation and Serverless Architectures

Before you continue, please complete **Lab 08, Making Your Environment Highly Available**.

This concludes Section 1, *Introduction to System Design*. **Section 2** will cover **Automation and Serverless Architectures**.

© 2017 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.

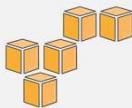
Errors or corrections? Email us at aws-course-feedback@amazon.com.

For all other questions, contact us at
<https://aws.amazon.com/contact-us/aws-training/>.

All trademarks are the property of their owners.

© 2017 Amazon Web Services, Inc. or its affiliates. All rights reserved.





CCA Unit 3 – Architecting on AWS

CCA 3.04: Event-Driven Scaling

Section 1: Introduction to System Design

Section 2: Automation and Serverless Architectures

► CCA 3.04 Event-Driven Scaling

CCA 3.05 Automating Your Infrastructure

CCA 3.06 Decoupling Your Infrastructure

CCA 3.07 Designing Web-Scale Storage

Section 3: Well-Architected Best Practices

Section 4: Deployment and Implementation

© 2017 Amazon Web Services, Inc. or its affiliates. All rights reserved.



Academy

This module begins **Section 2 of Unit 3 – Automation and Serverless Architectures**.

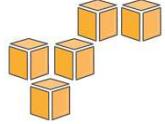
We'll start by discussing Event-Driven Scaling in CCA 3.04.

What's In This Module

- 💡 Enable Scalability
- 💡 How do you know if scaling is needed?
- 💡 Auto Scaling
 - 💡 EC2 Auto Recovery
 - 💡 Scaling Data Stores
 - 💡 AWS Lambda and Event-Driven Scaling

This module covers...

- Enable Scalability
- How do you know if scaling is needed?
- Auto Scaling
- EC2 Auto Recovery
- Scaling Data Stores
- AWS Lambda and Event-Driven Scaling



Part 1

Enable Scalability

© 2017 Amazon Web Services, Inc. or its affiliates. All rights reserved.

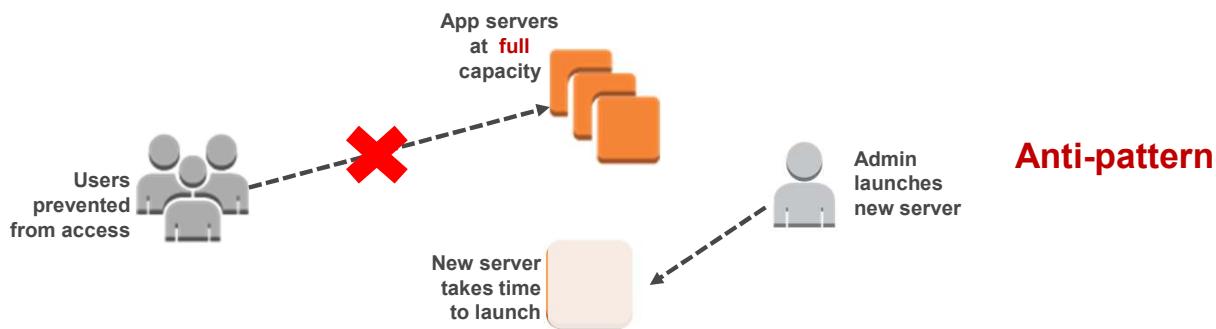
 Academy

Part 1: Enable Scalability

Best Practice: Enable Scalability

Ensure that your architecture can handle changes in demand.

A key advantage of a cloud-based infrastructure is how **quickly** you can respond to changes in resource needs.



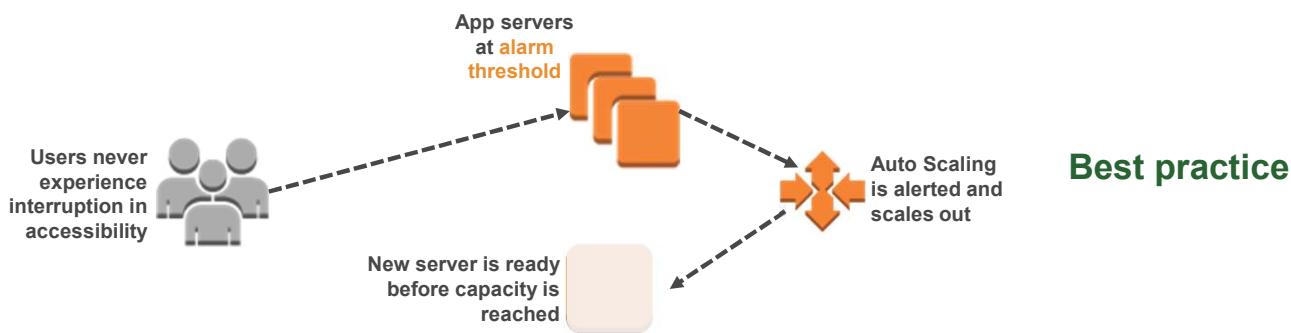
Easy and even seamless scalability of infrastructure is one of the biggest reasons for running your workloads on the AWS cloud. So make sure that your infrastructure is taking advantage of scalability at every layer.

1. In this anti-pattern (an example of a poorly designed system), scaling has to be done reactively and manually
2. The application servers reach their full capacity.
3. With no more room, users are being prevented from accessing the application.
4. The admin finds out that the servers are at their full capacity and starts launching a new instance to take some of the load off.
5. Unfortunately, since there's always a few minutes between when an instance is launched and when it's available to be used, that is more time that the users are not able to access the application.

Best Practice: Enable Scalability

Ensure that your architecture can handle changes in demand.

A key advantage of a cloud-based infrastructure is how **quickly** you can respond to changes in resource needs.



1. Using the best practice of enabling scalability, this pattern is designed to anticipate need and have more capacity available before it's too late.
 2. A monitoring solution (such as Amazon CloudWatch) detects that the total load across the fleet of servers has reached a specified threshold of load. This could be anything, such as "Stayed above 60% CPU utilization for longer than 5 minutes", or anything related to the use of resources, and with CloudWatch, you can even design custom metrics based around your specific application, which can trigger scaling in whatever way you need. When that alarm is triggered, Amazon EC2 Auto Scaling immediately launches a new instance.
 3. That instance is then ready before capacity is reached, providing a seamless experience for users, who never know that capacity was in danger of being reached.
- Ideally, you should also design this system to scale down once demand drops off again, so that you're not running instances that are no longer needed.

Vertical vs. Horizontal Scaling

Vertical scaling

Scale up and down

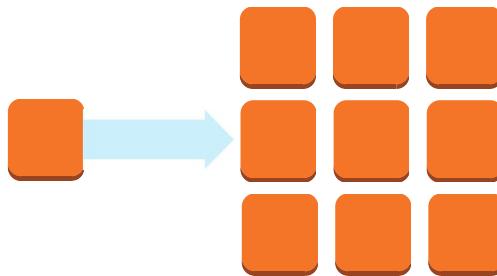
Change in the **specifications** of instances
(more CPU, memory, etc.)



Horizontal scaling

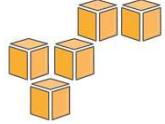
Scale in and out

Change in the **number** of instances (Add and remove instances as needed)



Vertical scaling keeps adding memory and CPUs as the workload increases. Eventually, this will hit the limit. Also, there are some performance concerns. If you are running Java applications, more memory (more Java heap) means it takes longer for garbage collection to run. It may introduce longer pause time. Another point to consider is that vertical scaling may require the server to be re-booted.

Horizontal scaling is virtually limitless. Needless to say, horizontal scaling is the ideal solution to handle growing workload, but it's important that if you're going to rely on horizontal scaling, your application be designed to fully take advantage of it.



Part 2

How do you know if scaling is needed?

© 2017 Amazon Web Services, Inc. or its affiliates. All rights reserved.

 **Amazon**
webservices | Academy

Part 2: How do you know if scaling is needed?

You Are Flying Blind Without Instrumentation

To leverage AWS in an **efficient** way, you need **insight** into your AWS resources, e.g.:

- How much of your infrastructure is actually being used?
- Is your application's performance or availability being affected by a lack of sufficient capacity?



*How do you capture
this information?*

Collecting Metrics: Amazon CloudWatch



Amazon CloudWatch:

- ❖ Monitors components in your infrastructure based on what you identify.
- ❖ Sends notifications and triggers auto scaling actions based on metrics you specify.

The foundation of the web tier includes the use of ELBs in the architecture. These load balancers not only send traffic to EC2 instances, but can also send metrics to Amazon CloudWatch, a managed monitoring service provided. The metrics from EC2 and the ELB can act as triggers—so that if we notice a particularly high latency or that our servers are becoming over utilized, we're able to take advantage of Auto Scaling to add more capacity to our web server fleet.

Monitor AWS Resources With Amazon CloudWatch

Amazon CloudWatch offers:

- ─ A distributed statistics gathering system; it collects and tracks metrics.
- ─ Metrics that are seamlessly collected at the hypervisor level by default.
- ─ **The ability to create and use custom metrics** of data generated by your own applications and services.

Examples:

- The time web pages take to load
- Request error rates
- Number of simultaneous processes or threads

The metrics are collected on the hypervisor level, so the existing instance metrics include:

- CPU utilization
- I/O (bytes and operations)
- Network (bytes and operations)

Custom metrics:

- Publish Custom Metrics: see
<http://docs.aws.amazon.com/AmazonCloudWatch/latest/DeveloperGuide/publishingMetrics.html>
- Monitoring scripts for Amazon EC2 instances: see
<http://docs.aws.amazon.com/AmazonCloudWatch/latest/DeveloperGuide/monitoringScripts.html>

Q: When would you use a custom metric over having your program emit a log to CloudWatch logs?

A: You can monitor your own data using custom metrics, CloudWatch logs, or both. You may want to use custom metrics if your data is not already produced in log format, for example operation system processes or performance measurements. Or, you may want to write your own application or script, or one provided by an APN partner. If you want to store and save individual measurements along with additional details, you may want to use CloudWatch logs.

CloudWatch Alarm Examples

Amazon
EC2



If CPU utilization is > 60% for 5 minutes...

Amazon
RDS



If number of simultaneous connections is > 10 for one minute...

Amazon
ELB

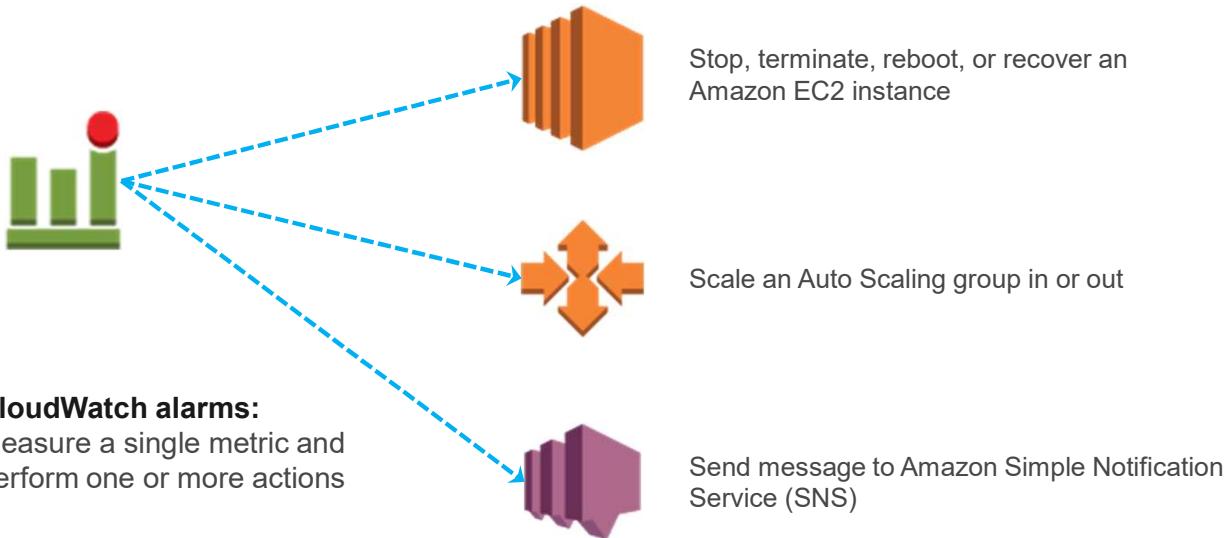


If number of healthy hosts is < 5 for 10 minutes...

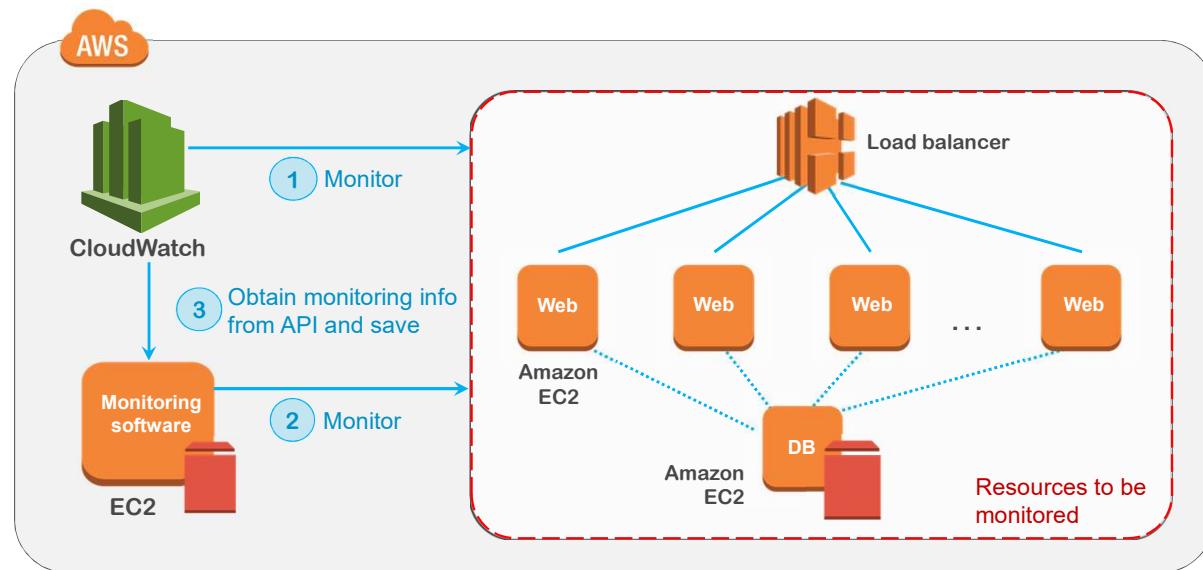
With these examples, the triggering of the alarm would start some other action, such as executing an auto scaling policy, sending a notification (to an Ops team, for instance), etc.

Actions can also be executed when an alarm is NOT triggered.

CloudWatch Alarms And Actions



Cloud Design Pattern: Monitoring Integration Pattern



© 2017 Amazon Web Services, Inc. or its affiliates. All rights reserved.



Problem to be solved:

Monitoring is necessary in system operation. A monitoring service is provided by the Amazon cloud. However, because the monitoring service in the Amazon cloud cannot monitor the internal workings of a virtual server (the operating system, middleware, applications, and so forth), you need to have an independent monitoring system.

Explanation of the cloud pattern:

The virtual server, etc., are monitored by the AWS cloud monitoring service, but you need to use your own system to monitor the operating system, middleware, applications, and so forth. The cloud monitoring service provides an API, enabling you to use your monitoring system to perform centralized control, including of the cloud side, through this API, to obtain information from the cloud monitoring system.

Implementation:

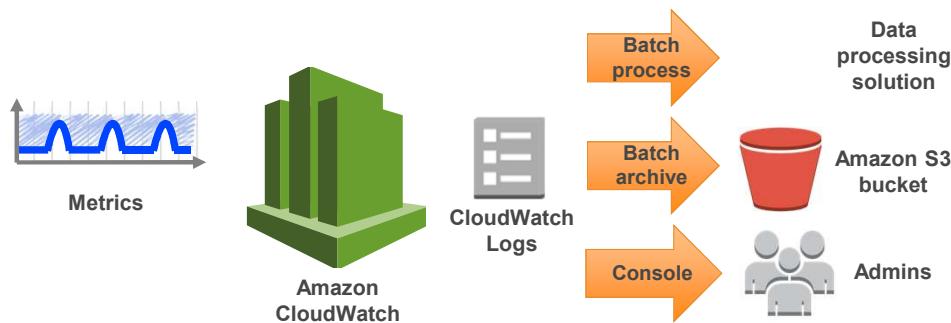
Install monitoring software on the Amazon EC2 instance to obtain monitoring information from the CloudWatch monitoring service.

- Install monitoring software (Nagios, Zabbix, Munin, or the like).
- Use a plug-in to obtain monitoring information using the CloudWatch API and to write that information to the monitoring software.
- Use the plug-in to perform monitoring, including the information from AWS.

For information about the Cloud Design Pattern, see

http://en.clouddesignpattern.org/index.php/CDP:Monitoring_Integration_Pattern

Store And Monitor Application Log Files With CloudWatch Logs



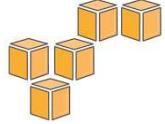
Your metrics can be stored durably in CloudWatch as CloudWatch Logs:

- Admins and other parties can review CloudWatch logs directly in the AWS Management Console
- Logs can be stored in Amazon S3, to be accessed by other services or another user
- Logs can be streamed in real time to data-processing solutions like Amazon Kinesis Streams or AWS Lambda

You can use CloudWatch Logs to store your log data in highly durable storage. You can change the log retention setting so that any log events older than this setting are automatically deleted. The CloudWatch Logs agent makes it easy to quickly send both rotated and non-rotated log data off a host and into the log service. You can then access the raw log data when you need it.

For more on CloudWatch Logs, see:

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



Part 2

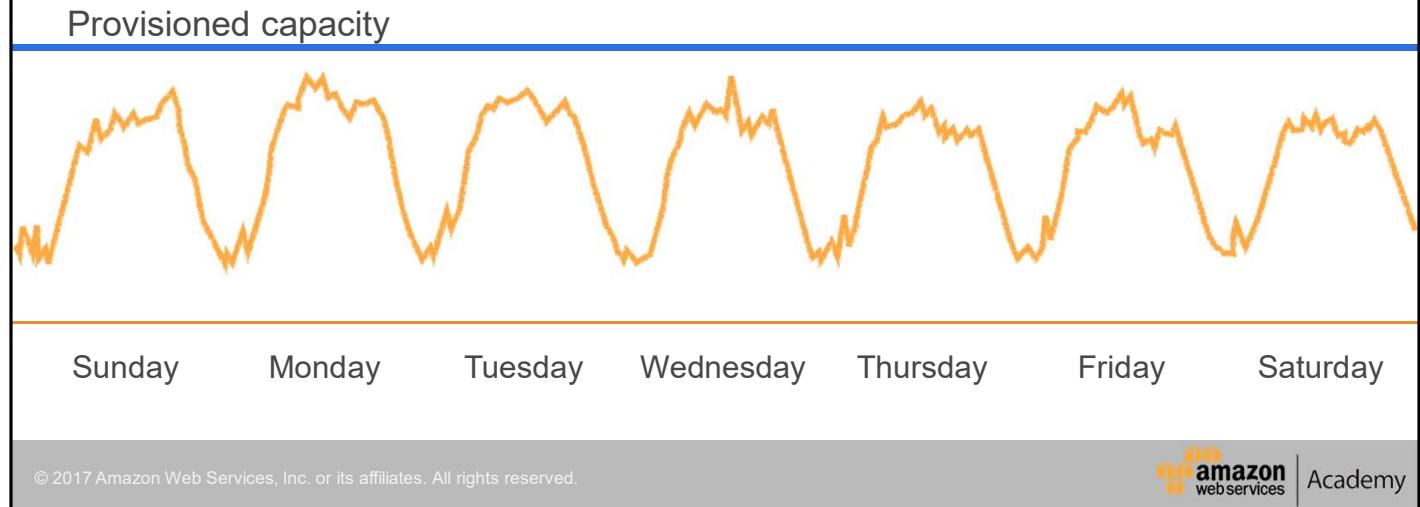
Auto Scaling

© 2017 Amazon Web Services, Inc. or its affiliates. All rights reserved.

 **Amazon**
webservices | Academy

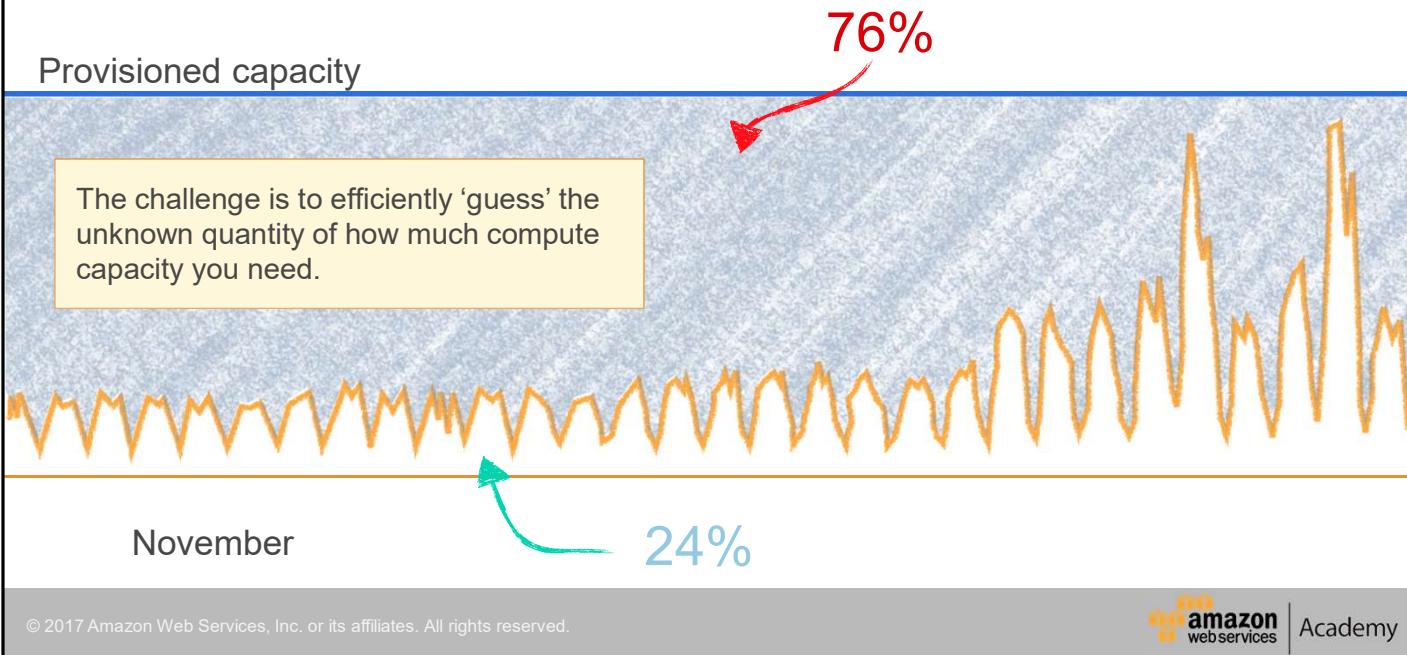
Part 3: Auto Scaling

Typical Weekly Traffic At Amazon.com



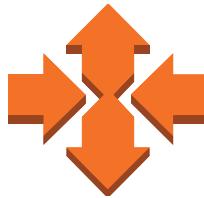
Needless to say, the retail company Amazon.com is one of the largest AWS customers. Typically, the incoming traffic is very predictable. Before Amazon.com moved their infrastructure onto AWS, they had a traditional data center, as many companies had. In order to support the peak load, your data center must provide enough hardware and software to support the capacity.

November Traffic To Amazon.com



Amazon.com experiences a seasonal peak in November (Black Friday). Because of this peak in late November, they had to invest in enough resources to support this seasonal peak, knowing that this only occurs a certain time of the year. As the business grew, Amazon.com had to keep investing in additional hardware and software. At some point, they ran out of space so they had to add a new data center. The problem is that about 76% of the resources are idle for most of the year. But if you don't have enough compute capability to support the seasonal peak, the server can crash and your business can lose customer confidence.

Auto Scaling

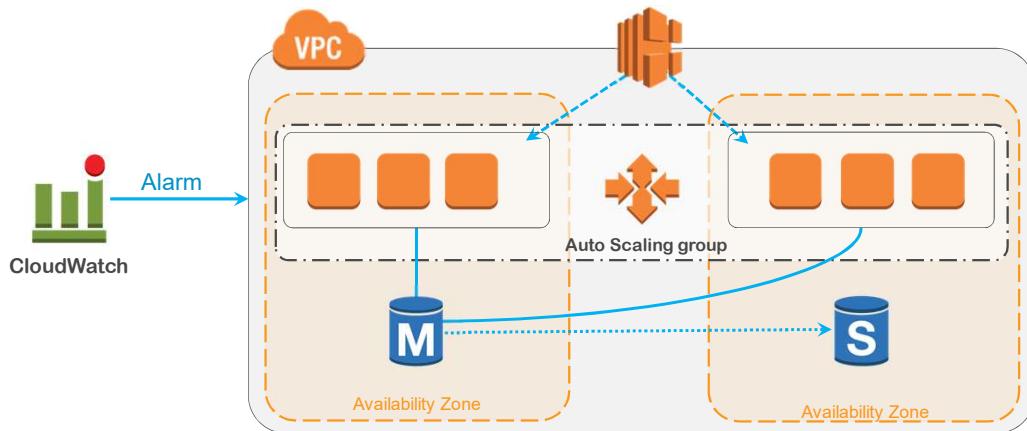


- Launches or terminates instances based on specified conditions.
- Automatically registers new instances with load balancers when specified.
- Can launch across Availability Zones.

The foundation of the web tier includes the use of Elastic Load balancers in the architecture. These load balancers not only send traffic to EC2 instances, but can also send metrics to Amazon CloudWatch, a managed monitoring service provided. The metrics from EC2 and the ELB can act as triggers—so that if we notice a particularly high latency or that our servers are becoming over utilized, we're able to take advantage of Auto Scaling to add more capacity to our web server fleet.

Do Not Guess About Resource Needs

Build a **flexible** system that will react to changes in customer demand and manage costs dynamically.



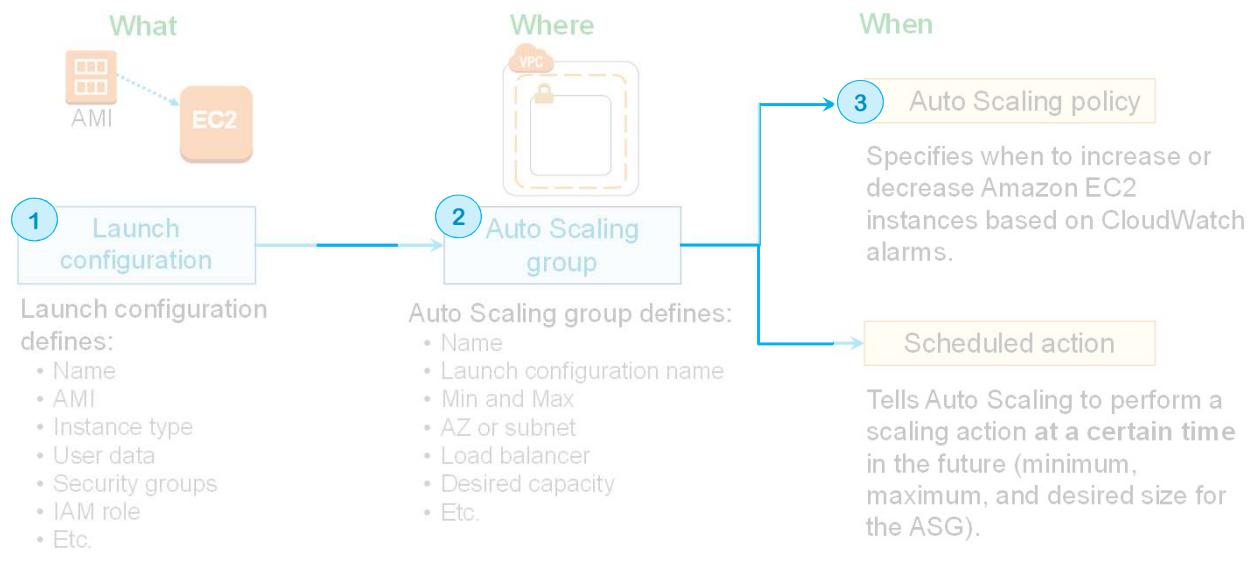
Auto Scaling: Don't Fear Constraints

Avoid limiting the ability to create new resources on-demand with just-in-time and/or scheduled provisioning.

How can you make this happen?

- APIs
- Scripts
- Auto Scaling

How Does Auto Scaling Work?



Auto Scaling components

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

Launch configuration

Launch configuration defines how Auto Scaling should launch your Amazon EC2 instances (similar to ec2-run-instances API). Auto Scaling provides you with an option to create a new launch configuration using the attributes from an existing Amazon EC2 instance. When you use this option, Auto scaling copies the attributes from the specified instance into a template from which you can launch one or more Auto Scaling groups.

Auto Scaling group

Your Auto Scaling group uses a launch configuration to launch Amazon EC2 instances. You create the launch configuration by providing information about the image you want Auto Scaling to use to launch Amazon EC2 instances. The information can be the image ID, instance type, key pairs, security groups, and block device mapping.

Auto Scaling policy

An Auto Scaling group uses a combination of policies and alarms to determine when the specified conditions for launching and terminating instances are met. An alarm is an object that watches over a single metric (for example, the average CPU utilization of your Amazon EC2 instances in an Auto Scaling group) over a time period that you specify. When the value

of the metric breaches the thresholds that you define, over a number of time periods that you specify, the alarm performs one or more actions. An action can be sending messages to Auto Scaling. A policy is a set of instructions for Auto Scaling that tells the service how to respond to alarm messages.

Scheduled action

Scaling based on a schedule allows you to scale your application in response to predictable load changes.

To configure your Auto Scaling group to scale based on a schedule, you need to create scheduled actions. A scheduled action tells Auto Scaling to perform a scaling action at a certain time in the future. To create a scheduled scaling action, you specify the start time at which you want the scaling action to take effect, and you specify the new minimum, maximum, and desired size you want for that group at that time. At the specified time, Auto Scaling updates the group to set the new values for minimum, maximum, and desired sizes, as specified by your scaling action.

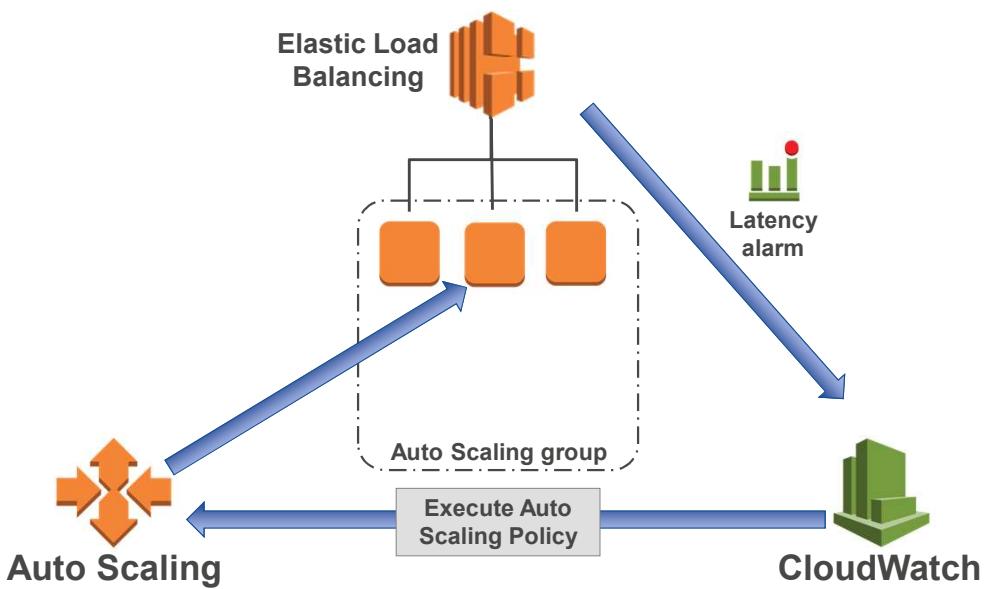
Auto Scaling Policy vs. Scheduled Actions

Auto Scaling Policy	Scheduled Action
<ul style="list-style-type: none"> Manually initiate a scaling activity Dynamic scaling with CloudWatch to initiate a scaling activity based on a metric 	<ul style="list-style-type: none"> Prior knowledge so you can be proactive (predictable schedule)
Parameters example: <ul style="list-style-type: none"> AutoScalingGroupName HonorCooldown PolicyName 	Parameters example: <ul style="list-style-type: none"> ScheduledActionName AutoScalingGroupName DesiredCapacity StartTime
Usage scenario: If CPU utilization reaches 70%, add an instance. In contrast, when the CPU utilization is low, remove an instance to reduce the cost.	Usage scenario: Traffic to your web application starts to increase on Wednesday, remains high on Thursday, and starts to decrease on Friday.

If you have unpredictable load/peak, configure an Auto Scaling policy using the information captured by CloudWatch. By doing so, you can avoid having servers crash due to out-of-memory or CPU starvation, etc.

On the other hand, you may have a system where the workload is predictable. Applications that are used internally within a company to process operational tasks (such as payroll, or some sort of batch processing to be executed at the end of each day) have a more distinctive pattern in memory and CPU consumption. In such a case, you may choose to define a scheduled scaling.

Elastic Load Balancing, CloudWatch, And Auto Scaling



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

1. In this scenario, a CloudWatch alarm assigned to the load balancer keeps track of latency. When it is triggered, the alarm notifies Amazon CloudWatch.
2. This triggers CloudWatch to execute an Auto Scaling policy.
3. Auto Scaling scales the assigned Auto Scaling group out and adds another instance.

How Do You Decide On Minimum Capacity Size?

- Auto Scaling group defines:
 - Desired capacity
 - Minimum capacity
 - Maximum capacity
- What would be a good **minimum** capacity to set it to?
- What would be a good **maximum** capacity to set it to?



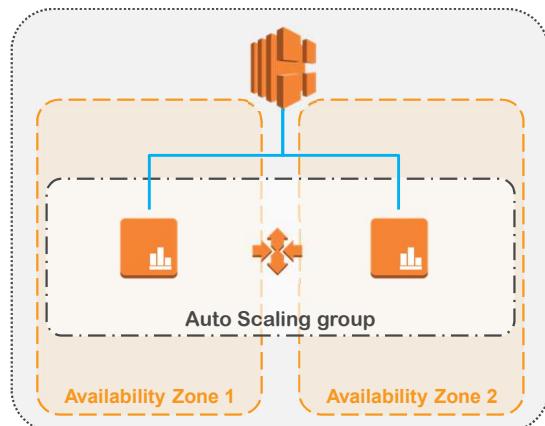
You don't have to specify the desired capacity because minimum capacity is the initial desired capacity. You pay for what you use; therefore, you don't need to launch more instances than you need and auto scale based on demand.

Deciding on the minimum capacity size depends on the type of applications that you are running. Here are some things to consider:

- If you are processing batches that run once a week, you might want to set the minimum to zero.
- Remember that it takes minutes to launch a new instance (depending on the complexity of your launch configuration). You may not be able to afford zero minimum capacity to start with.

How Do You Decide On Minimum Capacity Size?

What about high availability?



Minimum = two instances (# of AZs)

Desired capacity = two instances (Min.)

Maximum Capacity Size And Auto Scaling

CPU utilization triggers the alarm: capacity is doubled until CPU utilization drops below 60% or max capacity is reached.

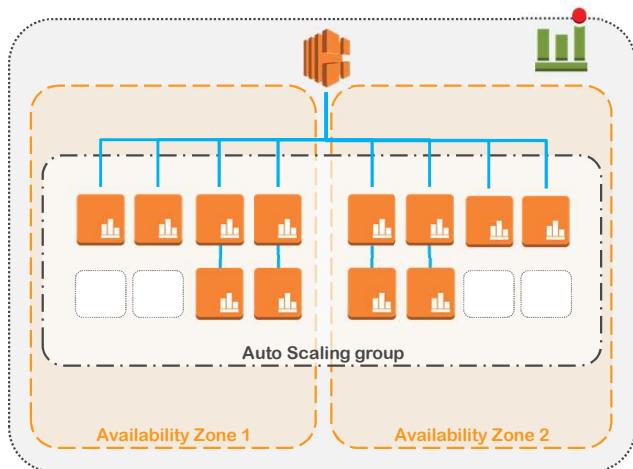
Scenario:

Auto Scaling group

- Minimum = 2
- Maximum = **12**

Auto Scaling policy

- When CPU utilization is greater than 60%
- Add 100% of group = **double the capacity**



This scenario demonstrates how maximum capacity size plays a role in the way Auto Scaling works.

As a discussion, you set the Auto Scaling policy to **double** the current group capacity when the CPU utilization becomes greater than 60%. You start with a minimum of two instances.

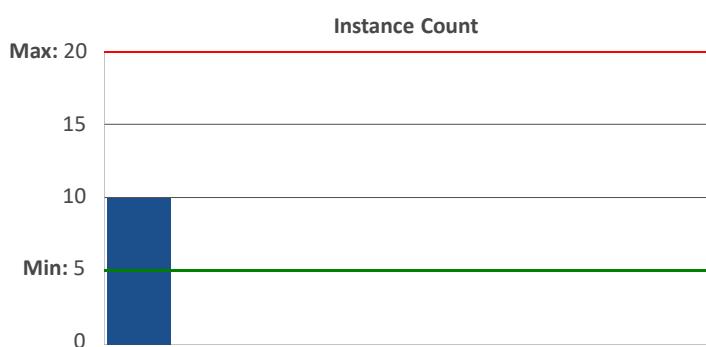
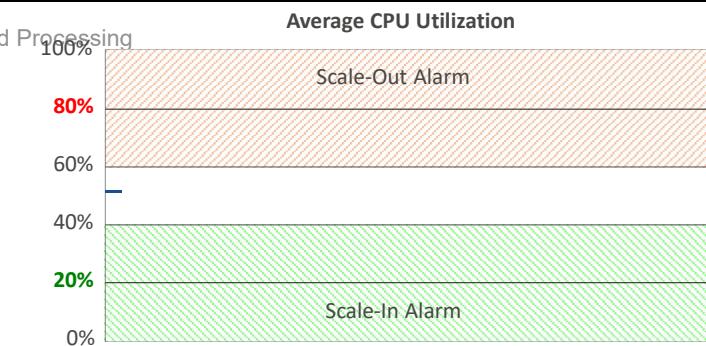
1. The first time the alarm triggers the Auto Scaling policy to take effect, the desired capacity becomes four (2x2).
2. Load keeps increasing and when CPU utilization reaches greater than 60%, the desired capacity becomes eight (2x4).
3. The next time the alarm gets triggered, you can no longer double the current group capacity because the maximum capacity is set to 12.
4. Auto Scaling adds four more instances equally balanced across the two AZs.

Selecting a reasonable maximum capacity size depends on your **application** and possibly a **budget**. You pay for what you use; therefore, your organization may limit you from running more than a certain number of instances. Even if you don't have a budget restriction, there is no single answer.

Auto Scaling Steps

Alarms:

- Add 2 instances when average CPU is 80-100%
- Add 1 instance when average CPU is 60-80%
- Remove 1 instance when average CPU is 20-40%
- Remove 2 instances when average CPU is 0-20%



The following slides will demonstrate how Auto Scaling response to scale in and scale out alarms.

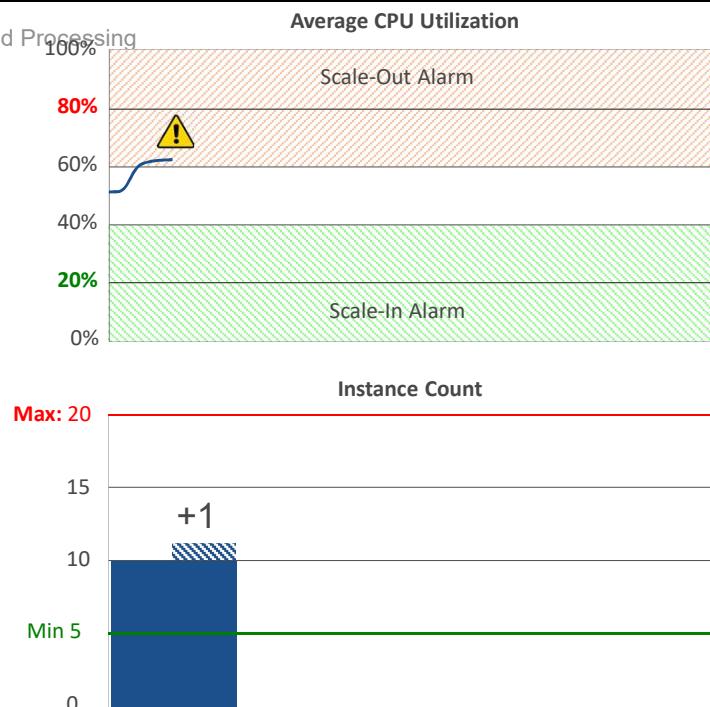
Auto Scaling Steps

As usage increases:

- CPU utilization goes up

When CPU utilization is 60-80%:

- Scale-out alarm is triggered
- Add 1 step policy is applied
- New instance is launched but not added to the aggregated group metrics until after warm up period expires



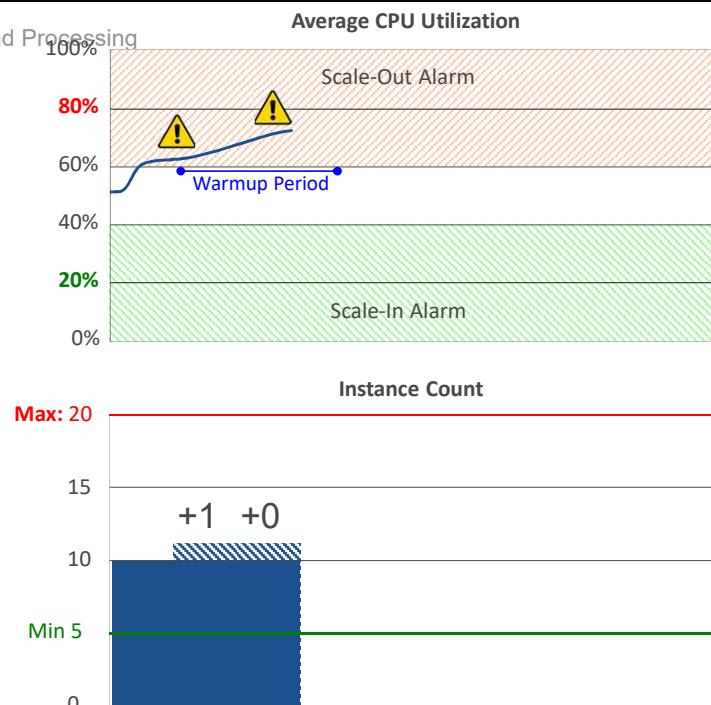
Auto Scaling Steps

As usage increases:

- CPU utilization goes up.

While waiting for new instance:

- CPU utilization remains high.
- Another alarm period is triggered.
- Since current capacity is still 10 during the warmup period, and desired capacity is already 11, no additional instances are launched.



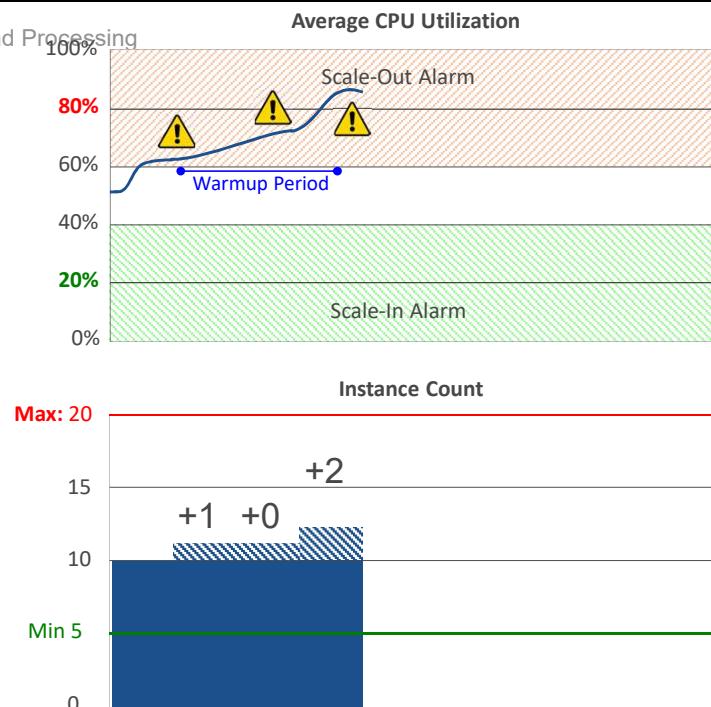
Auto Scaling Steps

As usage increases further:

- CPU utilization goes up

When CPU utilization is 80-100%:

- Scale-out alarm is triggered
- Add 2 step policy is applied
- Since the alarm occurred during a warm up period, two instances are launched less the one instance added during the first alarm
- Again new instances are not added to aggregated group metrics



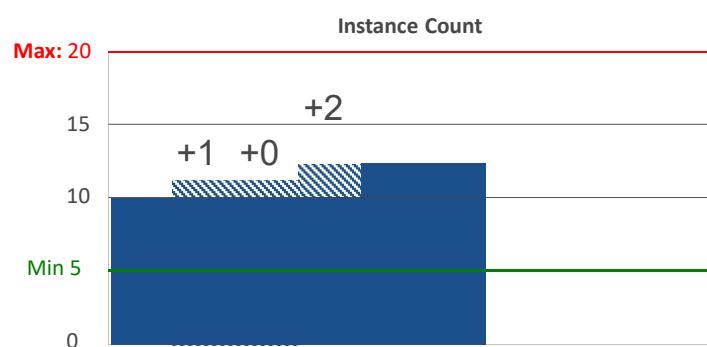
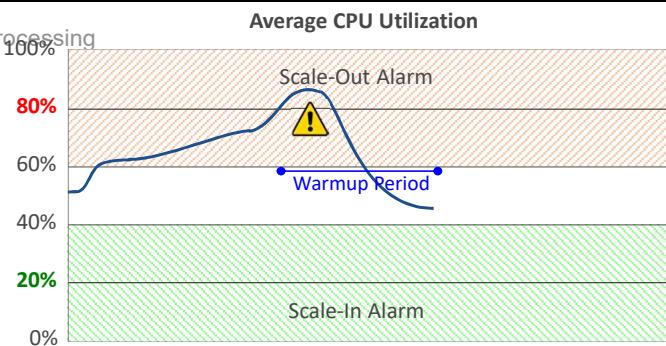
Auto Scaling Steps

As capacity matches usage:

- CPU utilization stabilizes

When CPU utilization is 40-60%:

- No alarms are triggered
- After warm up periods expire, new instances are added to the aggregated group metrics



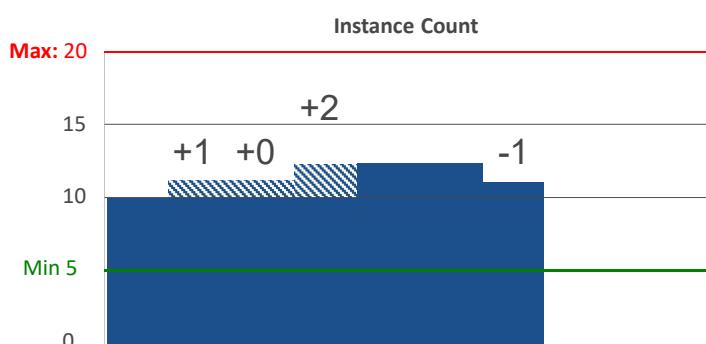
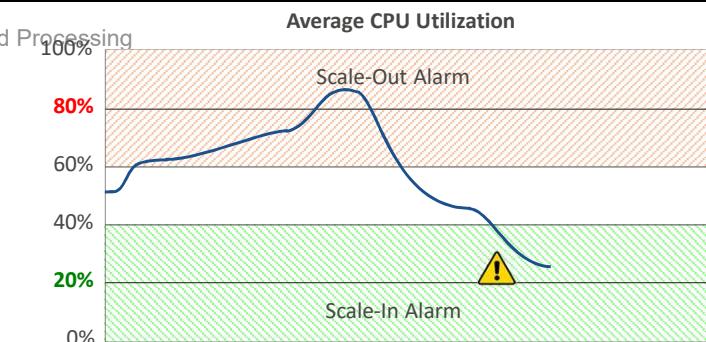
Auto Scaling Steps

As usage decreases:

- CPU utilization goes down

When CPU utilization is 20-40%:

- Scale-in alarm is triggered
- Remove 1 step policy is applied
- An instance is removed from the Auto Scaling group and from the aggregated group metrics



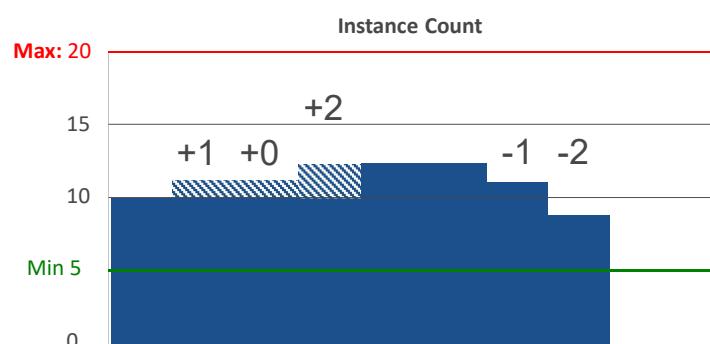
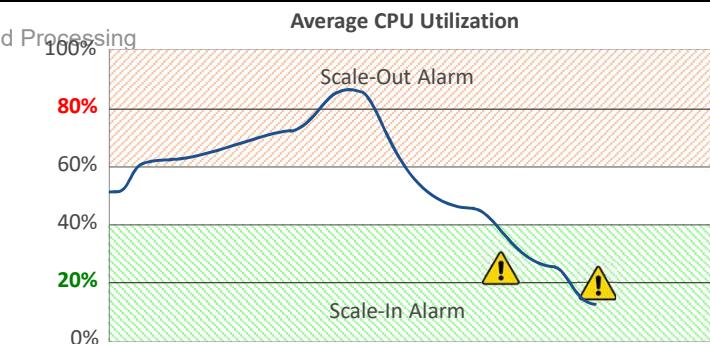
Auto Scaling Steps

As usage decreases:

- CPU utilization goes down further

When CPU utilization is 0-20%:

- Scale in alarm is triggered
- Remove 2 step policy is applied
- Two instances are removed from the Auto Scaling group and from the aggregated group metrics



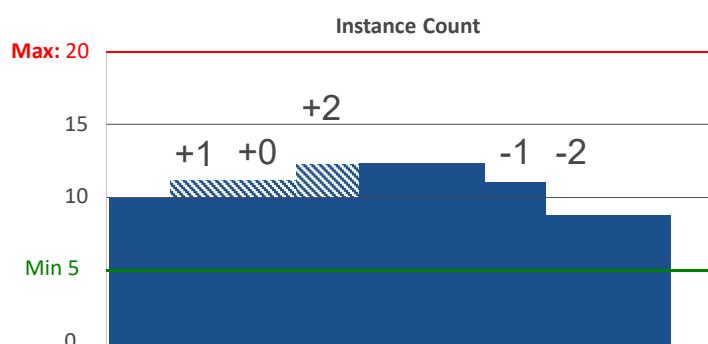
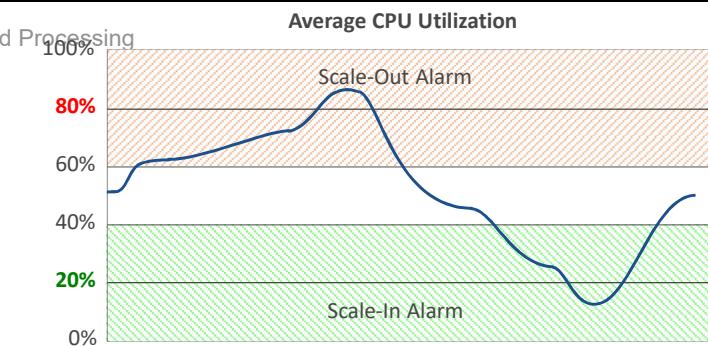
Auto Scaling Steps

As capacity matches usage:

- CPU utilization stabilizes

When $40\% < \text{CPU Utilization} < 60\%$:

- No alarm is triggered
- No step policies are applied
- No instances are added or removed from service



Auto Scaling Considerations

- Avoid Auto Scaling thrashing.
 - Be more cautious about scaling in; avoid aggressive instance termination.
 - Scale out early, scale in slowly.
- Set the min and max capacity parameter values carefully.
- Use lifecycle hooks.
 - Perform custom actions as Auto Scaling launches or terminates instances.
- Stateful applications will require additional automatic configuration of instances launched into Auto Scaling groups.

Remember: Instances can take several minutes after launch to be **fully usable**.

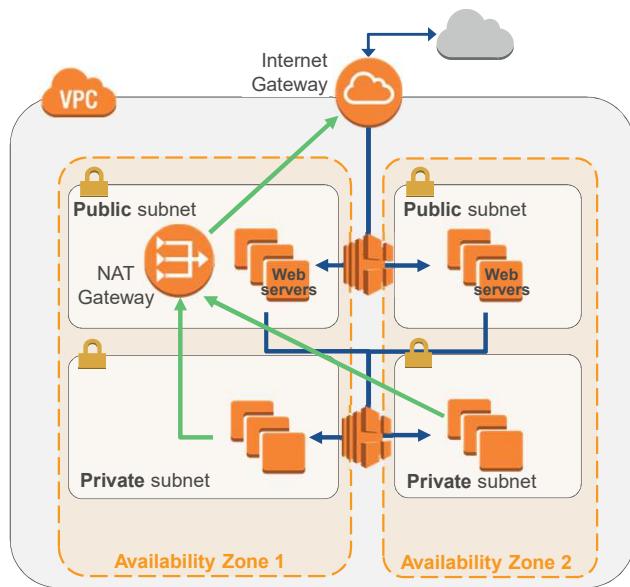
Scaling in means you are decreasing the computing capacity due to low utilization. It is possible that the workload spikes right after you scale in if you have an unpredictable workload. Depending on your launch configuration, it may take a few minutes for Auto Scaling to launch additional instances. Therefore, you should scale in slowly rather than aggressively. Being cost conscious is good but your applications should not suffer.

Desired capacity is different from minimum capacity. An Auto Scaling group's desired capacity is the default number of instances that should be running. A group's minimum capacity is the fewest number of instances the group can have running. For instance, a group with a desired capacity of 4 will run 4 instances when there are no scaling policies in effect. If the group has a minimum capacity of 2, then any scale-in policy which goes into effect can not reduce the total number of instances in the group below 2.

More on Auto Scaling lifecycle hooks:

<http://docs.aws.amazon.com/AutoScaling/latest/DeveloperGuide/lifecycle-hooks.html>

How Can You Increase The Availability Of This System?



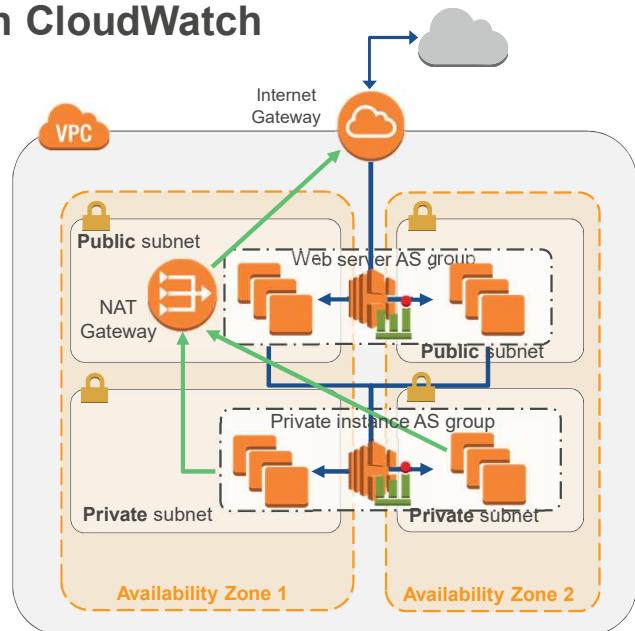
HA With Auto Scaling And Amazon CloudWatch

Amazon CloudWatch

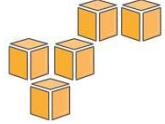
- Monitors the health of your ELBs and/or your instances and launches actions appropriately.

Auto Scaling

- Receives request from Amazon CloudWatch.
- Adds or subtracts instances.



To make this application environment even more highly available, you can monitor the ELB load balancers, the instances, or both, to determine when new instances need to be launched. When a CloudWatch alarm is triggered based on the metrics you've specified for it to monitor, it takes the action you've specified; that could include sending a notification to an Amazon SNS topic or triggering a scaling action, or both. We will spend more time on event-driven scaling in a later module, but for now it's important to understand this basic pattern for making a web-tier highly available.



Part 4

EC2 Auto Recovery

© 2017 Amazon Web Services, Inc. or its affiliates. All rights reserved.



Part 4: EC2 Auto Recovery

EC2 Auto Recovery

Replace impaired Amazon EC2 instances automatically with Amazon EC2's auto recovery feature.

- Conditions (detected by CloudWatch) that cause an instance to be impaired:
 - Loss of network connectivity
 - Loss of system power
 - Software/hardware issues on host
- Replacement instances:
 - Maintain same instance ID/metadata, IP addresses
 - Cannot use in-memory data from impaired instance (that data is lost)

For more information, see:

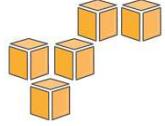
<http://docs.aws.amazon.com/AWSEC2/latest/UserGuide/ec2-instance-recover.html>

EC2 Auto Recovery

- Currently supported instance types:
C3, C4, M3, M4, R3, and T2
- GovCloud and China regions are not supported yet.
- Instances must be in a VPC and use shared tenancy.
- Instance storage cannot be used; you must use EBS-backed storage exclusively.
- Use instead of Auto Scaling when your job needs to maintain identical instance metadata or storage volume.

For more information, see:

<http://docs.aws.amazon.com/AWSEC2/latest/UserGuide/ec2-instance-recover.html>



Part 5

Scaling Data Stores

© 2017 Amazon Web Services, Inc. or its affiliates. All rights reserved.

 **Amazon**
webservices | Academy

Part 5: Scaling Data Stores

How To Scale With Amazon RDS

With scaling on Amazon RDS you can:

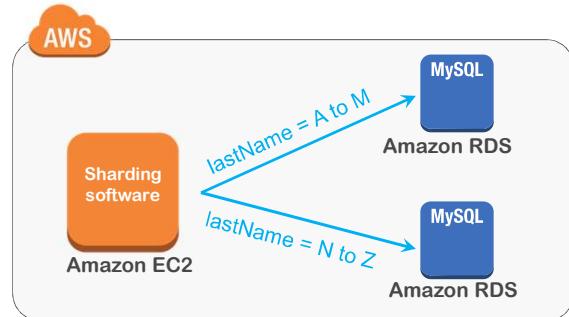
- Scale up or down with resizable instance types
- Scale your storage up with a few clicks or via the API
Easy conversion from standard to Provisioned IOPS storage
- Offload read traffic to Read Replicas

For increased performance, put a cache in front of Amazon RDS, such as:

- Amazon ElastiCache for Memcached or Redis
- Your preferred cache solution, self-managed in Amazon EC2

Scaling Amazon RDS Writes With Database Sharding

- Without shards, all data resides in one partition.
Example: Users by last name, A to Z, in one database.
- With **sharding**, split your data into **large chunks** (shards).
Example: Users by last name, A through M, in one database; N through Z in another database.
- In many circumstances, sharding gives you **higher performance** and **better operating efficiency**.



Sharding is a technique for improving the performance of writing with multiple database servers. Fundamentally, databases with identical structures are prepared and divided using appropriate table columns as keys, to distribute the writing processes. The use of the RDBMS service provided in the AWS cloud lets you perform this sharding to achieve an increase in availability and operating efficiency.

You can use Amazon RDS in sharding backend databases. Install sharding software such as MySQL server combined with a Spider Storage Engine on an Amazon EC2 instance. Prepare multiple RDSs and use them as the sharding backend databases. You can distribute the RDSs to multiple regions.

Refer to Cloud Design Pattern, “Sharding Write Pattern” for more information:

http://en.clouddesignpattern.org/index.php/CDP:Sharding_Write_Pattern

Horizontal Scaling With Read Replicas – Amazon RDS

Add **Read Replicas**

- Horizontally scale for read-heavy workloads
- Offload reporting

Keep in mind...

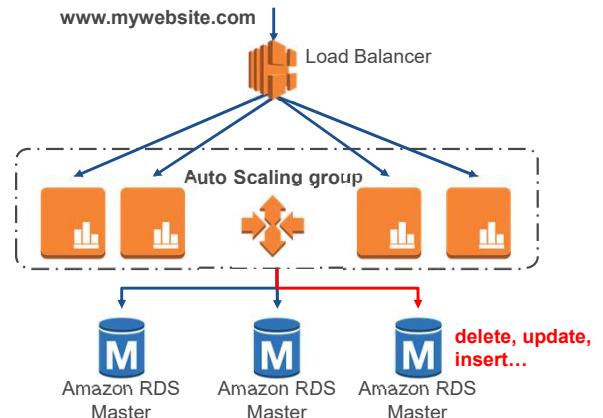
- Replication is **asynchronous**
- Currently available for Amazon Aurora, MySQL, and PostgreSQL (9.3.5 and later)

Read replicas with PostgreSQL have specific requirements, outlined here:

http://docs.aws.amazon.com/AmazonRDS/latest/UserGuide/USER_ReadRepl.html#USER_ReadRepl.PostgreSQL

Scaling Amazon RDS: Push-button Scaling

- Scale nodes vertically up or down
From micro to 8xlarge and everything in-between
- Scale out nodes horizontally
Shard based on data or workload characteristics
- Scale storage vertically without incurring downtime



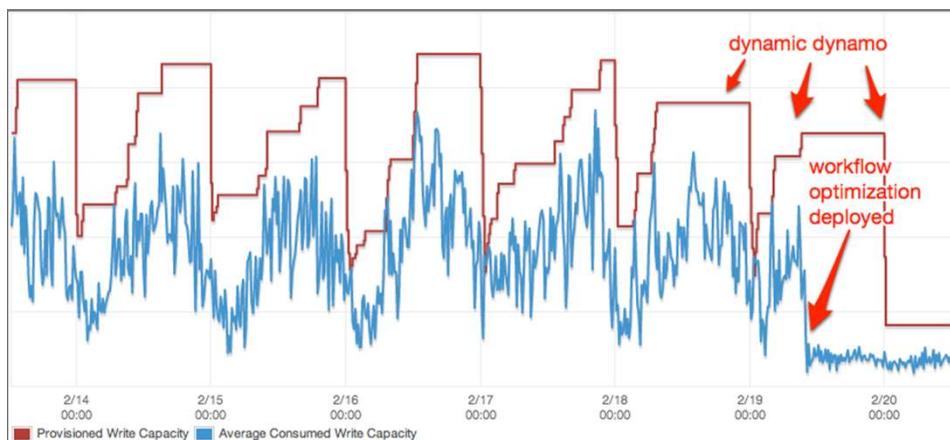
Using the Amazon RDS APIs or a few clicks of the AWS Management Console, you can scale the compute and memory resources powering your deployment up or down. Scaling operations typically finish within a few minutes.

As your storage requirements grow, you can provision additional storage on-the-fly with zero downtime. If you are using RDS PIOPS (with the exception of Amazon RDS with SQL Server), you can also scale the throughput of your DB instance by specifying the IOPS rate from 1,000 IOPS to 30,000 IOPS in 1,000 IOPS increments and storage from 100GB and 6TB.

[Amazon RDS for SQL Server](#) does not currently support increasing storage or IOPS of an existing SQL Server DB instance.

Auto Scaling With Dynamic DynamoDB

- Third-party library for automating scaling decisions.
- Scale up for service levels, scale down for cost.
- CloudFormation template for fast deployment.



When you create a DynamoDB table, you provision the desired amount of request capacity, taking into account the amount of read and write traffic and the average size of each item. You can change this capacity (up or down) as you gain a better understanding of your application's requirements.

Many independent developers have built and published tools to facilitate the use of DynamoDB in a wide variety of environments.

A CloudFormation template has been created that enables you to use Dynamic DynamoDB with just a couple of clicks. You can configure Dynamic DynamoDB to scale your tables up and down automatically, and you can restrict scaling activities to certain time slots. You can scale read and write throughput capacity independently using upper and lower thresholds, and you can set minimums and maximums for each value. Finally, Dynamic DynamoDB supports a circuit-breaker. Before it performs any scaling activities, it can verify that your application is up and running. This final check will avoid spurious scale-down activities if your application is experiencing other problems.

This template lives at <https://raw.githubusercontent.com/sebdah/dynamic-dynamodb/master/cloudformation-templates/dynamic-dynamodb.json>.

The template launches a t1.micro Amazon EC2 instance with the Dynamic DynamoDB package pre-installed. The instance is managed by an Auto Scaling group and will be replaced if it fails.

For more information, see:

- <http://aws.amazon.com/blogs/aws/auto-scale-dynamodb-with-dynamic-dynamodb/>
- <http://dynamic-dynamodb.readthedocs.org/en/latest/index.html>

Peak Games Uses AWS To Scale Its Games Platform

“

We had to provision servers through email and wait four to six hours. If there was a problem, we had to manually import all the backups to new servers.

Serdar Sahin
Head of Cloud and Big Data Services,
Peak Games



”

Peak Games is the largest and fastest-growing gaming company focused on the emerging markets of Turkey, Middle East and North Africa.

Challenge:

- Peak Games, based in Turkey, is a gaming company with 25 million players across the Middle East and North Africa
- In 2011, they were using a co-located server solution and could not keep up with managing scaling, backups, provisioning and setting up new servers, HA, and DR
- Needed to avoid a large up-front investment for new servers, and wanted greater scalability for unpredictable game launches

Peak Games Uses AWS To Scale Its Games Platform

Solution: Move to the AWS Cloud

Peak Games uses:

- **Amazon EC2 in Auto Scaling groups** for its game service
- **Amazon RDS with MySQL** for storing data
- **Amazon S3** to store backups and game assets
- **Amazon ElastiCache** to improve performance of their web applications
- **Elastic Load Balancing** to distribute traffic across applications
- **Amazon SNS** for internal applications (e.g. to push marketing notifications directly to users)

“

Whenever we launch a new application or game, we first determine the AWS services we can use, which greatly reduces time to market and devops overhead. We then scale our infrastructure based on the load.

Serdar Sahin
Head of Cloud and Big Data Services,
Peak Games



”

© 2017 Amazon Web Services, Inc. or its affiliates. All rights reserved.



Important to note that all of the services chosen scale either inherently or by implementation within Peak Games' environment.

Peak Games Uses AWS To Scale Its Games Platform

“

We're developing new games constantly. Thanks to AWS, our devops team can set up the infrastructure for our new games while supporting our existing games.

Serdar Sahin
Head of Cloud and Big Data Services,
Peak Games



”

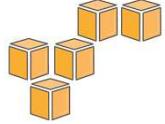
Results:

- By moving to AWS, Peak Games reduced provisioning time from 4-6 hours to 10 minutes.
- Auto Scaling their game and data layers allows them to handle sharp changes in demand over the course of a day.
- They also reduced time to market by 75%, as well as cut operational costs and staff needs, while increasing flexibility, game availability, and scalability.

© 2017 Amazon Web Services, Inc. or its affiliates. All rights reserved.



For more on the Peak Games case study, see: <https://aws.amazon.com/solutions/case-studies/peak-games/>



Part 6

AWS Lambda and Event-Driven Scaling

© 2017 Amazon Web Services, Inc. or its affiliates. All rights reserved.



Part 6: AWS Lambda and Event-Driven Scaling

AWS Lambda



- Fully managed compute service that runs stateless code (Node.js, Java, and Python) in response to an event or on a time-based interval.
- Allows you to run code without managing infrastructure like Amazon EC2 instances and Auto Scaling groups.

The foundation of the web tier includes the use of Elastic Load balancers in the architecture. These load balancers not only send traffic to EC2 instances, but can also send metrics to Amazon Cloudwatch, a managed monitoring service provided. The metrics from EC2 and the ELB can act as triggers—so that if we notice a particularly high latency or that our servers are becoming over utilized, we're able to take advantage of Auto Scaling to add more capacity to our web server fleet.

AWS Lambda



AWS Lambda handles:

- Servers
- Capacity needs
- Deployment
- Scaling and fault tolerance
- OS or language updates
- Metrics and logging

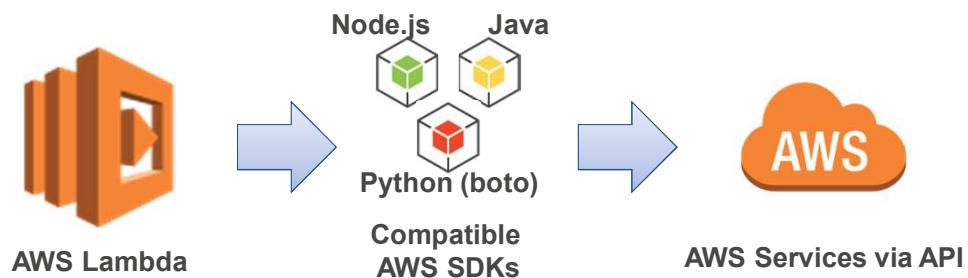
AWS Lambda enables you to:

- Bring your own code (even native libraries)
- Run code in parallel
- Create back ends, event handlers, and data processing systems
- Never pay for idling resources!

How Can AWS Lambda Be Used With Scaling?

Scaling events can trigger AWS Lambda functions. In addition, code run in AWS Lambda has access to the AWS API and can grant permissions via AWS IAM roles.

This means you can use a Lambda function to **automatically make API calls to other AWS services** when scaling happens.



How Can AWS Lambda Be Used With Scaling?

Examples of scaling-related operations you could perform with AWS Lambda:

- Scale container-based instances (Docker, Amazon Elastic Container Service, etc.)
- Scale more intelligently using functions (e.g.: analyze a stream of performance data looking for patterns rather than just events)
- Since AWS Lambda can scale automatically, consider replacing some Amazon EC2 instances with Lambda functions where appropriate



CCA Lab-09

Using Auto Scaling with AWS Lambda and Lifecycle Hooks

(Approx. 45 minutes)

© 2017 Amazon Web Services, Inc. or its affiliates. All rights reserved.

 Academy

Lab Scenario

In this lab, you will add an **AWS Lambda function** to an Auto Scaling group. This function will **automatically tag and create an snapshot** of any new instances launched into the group.

Here are the services that will be used:



Amazon
SNS



Auto
Scaling



AWS
Lambda



AWS IAM



Amazon EBS
snapshot



Amazon
CloudWatch

Lab Tasks

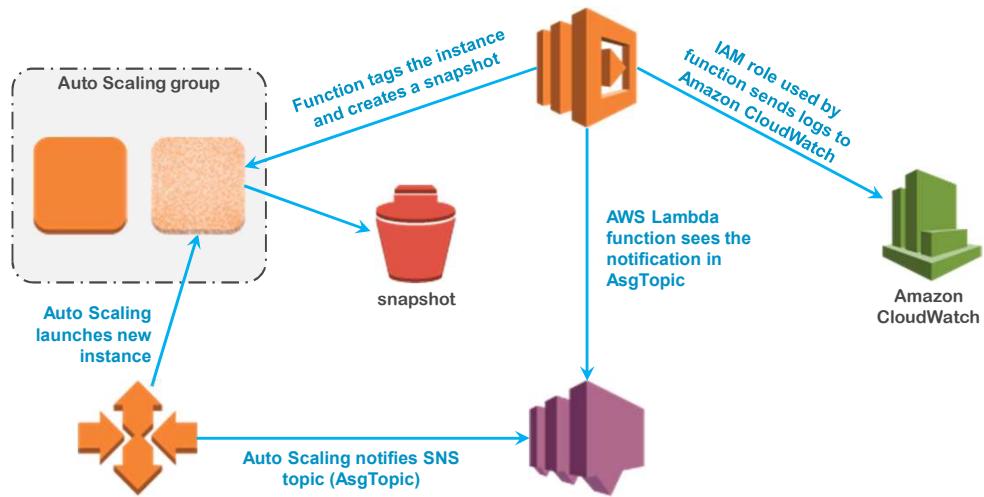


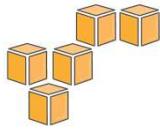
~ 45 min.

Here are the tasks the lab guide will walk you through:

1. Inspect the resources created automatically for your lab.
2. Create an SNS topic to receive notifications when your Auto Scaling group scales out.
3. Create an IAM role that will allow your AWS Lambda function to interact with Amazon EC2, Auto Scaling, and Amazon CloudWatch.
The function is triggered by a notification sent to your SNS topic. When it runs, it finds the instance being launched by Auto Scaling, adds a tag to it, and when the instance is ready it creates a snapshot of it.
4. Create an AWS Lambda function.
5. Manually scale your Auto Scaling group out to test your function.

Lab 3: Final Product





In review...

- ─ Enable Scalability
- ─ How do you know if scaling is needed?
- ─ Auto Scaling
- ─ EC2 Auto Recovery
- ─ Scaling Data Stores
- ─ AWS Lambda and Event-Driven Scaling



Knowledge Assessment

© 2017 Amazon Web Services, Inc. or its affiliates. All rights reserved.

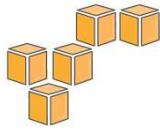


In review...

- Enable Scalability
- How do you know if scaling is needed?
- Auto Scaling
- EC2 Auto Recovery
- Scaling Data Stores
- AWS Lambda and Event-Driven Scaling

To complete this module, please remember to finish the corresponding knowledge assessment.

To complete this module, please remember to finish the corresponding knowledge assessment.



Up Next...



LAB 09 - Using Auto Scaling with AWS Lambda and Lifecycle Hooks



CCA 3.05 - Automating Your Infrastructure

Before you continue, please complete **Lab 09, Using Auto Scaling with AWS Lambda and Lifecycle Hooks**.

Up next is **CCA 3.05 - Automating Your Infrastructure**.

© 2017 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.

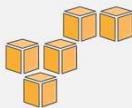
Errors or corrections? Email us at aws-course-feedback@amazon.com.

For all other questions, contact us at
<https://aws.amazon.com/contact-us/aws-training/>.

All trademarks are the property of their owners.

© 2017 Amazon Web Services, Inc. or its affiliates. All rights reserved.





CCA Unit 3 – Architecting on AWS

CCA 3.05: Automating Your Infrastructure

Section 1: Introduction to System Design

Section 2: Automation and Serverless Architectures

CCA 3.04 Event-Driven Scaling

► **CCA 3.05** Automating Your Infrastructure

CCA 3.06 Decoupling Your Infrastructure

CCA 3.07 Designing Web-Scale Storage

Section 3: Well-Architected Best Practices

Section 4: Deployment and Implementation

© 2017 Amazon Web Services, Inc. or its affiliates. All rights reserved.



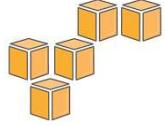
Welcome to CCA-3.05, Automating Your Infrastructure.

What's In This Module

- Automate Your Environment
- Infrastructure as code using CloudFormation
- Using Templates
- Extending CloudFormation

This module covers...

- Automate Your Environment
- Infrastructure as code using CloudFormation
- Using Templates
- Extending CloudFormation



Part 1

Automate Your Environment

© 2017 Amazon Web Services, Inc. or its affiliates. All rights reserved.

 Academy

Part 1: Automate Your Environment

Manual Configuration Challenges

Creating and configuring AWS services and resources through a management console is a **manual** process.

What are the challenges and concerns for a manual process?

- Reliability
- Reproducibility
 - DEV
 - TEST
 - PROD
- Documentation

Repeating the same steps to configure each environment manually can be error prone. There is always a chance of human error such as a typo (e.g., database name). Frequently, you need to apply the same configuration that you have on your DEV environment to QA environments. There may be multiple QA environments for each stage of testing (e.g., functional testing, user acceptance test, and stress testing environments). Often a QA tester discovers a defect caused by wrongly configured resources, which could introduce further delay in the test schedule. Most importantly, you cannot afford to have a configuration error in production servers.

In order to reproduce exactly the same configuration, you may need to document a step-by-step configuration instruction.

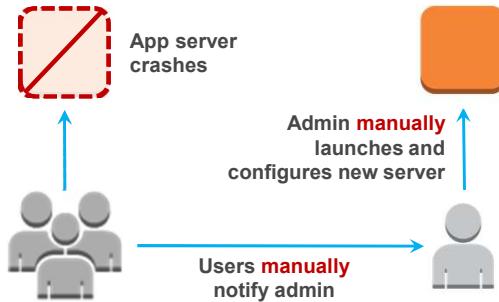
The solution to this challenge is to automate these steps by creating a script. The automation script itself can be the documentation. As long as the script is written correctly, it is more reliable than manual configuration. It is certainly reproducible.

Best Practice: Automate Your Environment

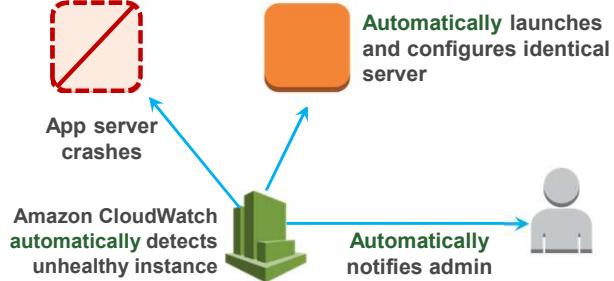
Where possible, automate the provisioning, termination, and configuration of resources.

Improve your system's **stability** and **consistency**, as well as the **efficiency** of your organization, by removing manual processes.

Anti-pattern



Best practice



© 2017 Amazon Web Services, Inc. or its affiliates. All rights reserved.



AWS offers built-in monitoring and automation tools at virtually every layer of your infrastructure. Take advantage of these resources to ensure that your infrastructure can respond quickly to changes in needs. Detecting unhealthy resources and launching replacement resources can be automated, and you can even be notified when resources are changed.

Best Practice: Use Disposable Resources

Take advantage of the dynamically provisioned nature of cloud computing.

Think of servers and other components as temporary resources.

Anti-pattern

- Over time, different servers end up in different configurations
- Resources run when not needed
- Hardcoded IP addresses prevent flexibility
- Difficult/inconvenient to test new updates on hardware that's in use

Best practice

- Automate deployment of new resources with identical configurations
- Terminate resources not in use
- Switch to new IP addresses automatically
- Test updates on new resources, and then replace old resources with updated ones

This best practice returns to the idea of thinking of your infrastructure as software instead of hardware. With hardware, it's easy to "buy in" too much on specific components, which makes it harder to upgrade when necessary because you have too much sunk cost.

On AWS, we recommend that you think of your resources the opposite way: migrating between instances or other discrete resources is fairly simple, which means that you can (and should) treat your resources as easily replaceable. This enables you to move more quickly to respond to changes in capacity needs and upgrade applications and underlying software.

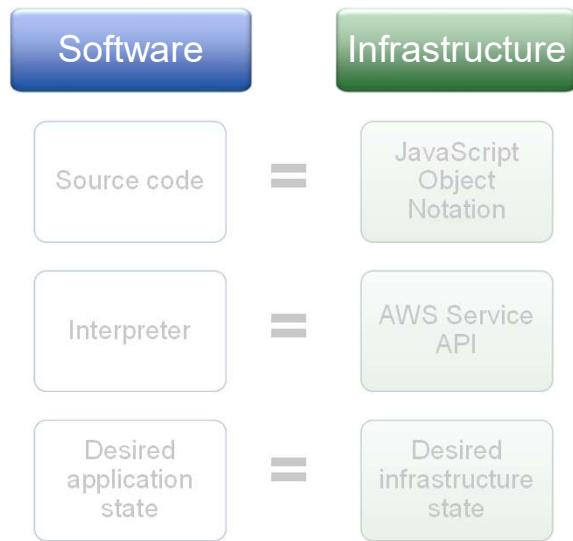
What Does Infrastructure As Code Mean?



Techniques, practices, and tools from **software development** applied to creating **reusable**, **maintainable**, **extensible** and **testable** infrastructure.

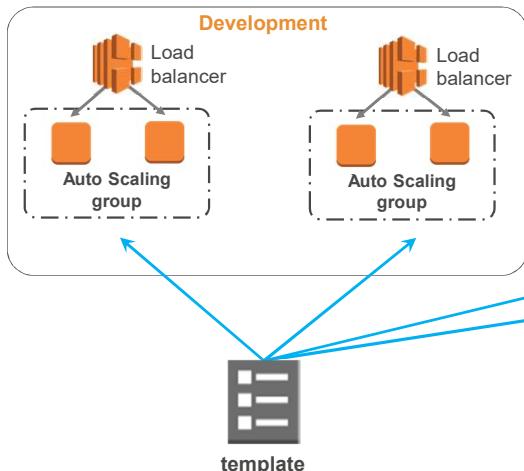
There are many components to automating your infrastructure, but the most critical one is defining your infrastructure not as bundles of hardware components, but as code.

Build And Operate Your Infrastructure Like Software

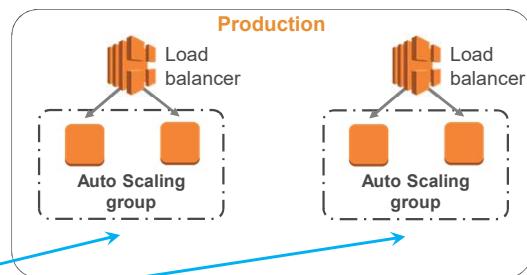


Benefits Of Treating Infrastructure As Code

Repeatability



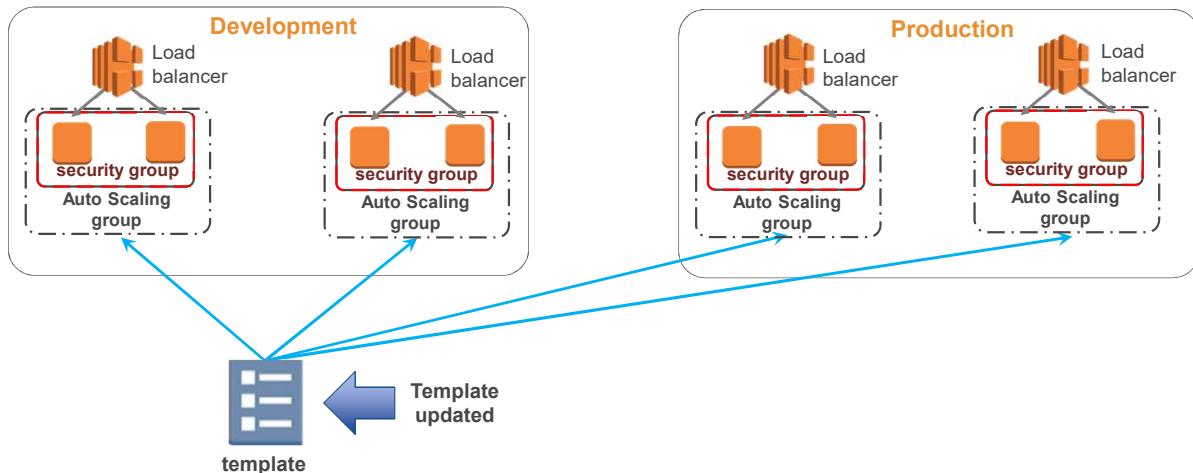
Reusability



If you build infrastructure with code, you gain the benefits of repeatability and reusability while building your environments. With one template (or a combination of templates), you can build the same complex environments over and over again. And when doing this with AWS, you can even create environments dependent upon conditions, so that what ends up being built is specific to the context in which you've created it. For instance, a template can be designed so that different AMIs are used based on whether or not this template was launched into the development or the production environments.

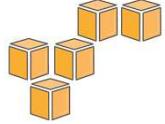
Benefits Of Treating Infrastructure As Code

Maintainability, Consistency, and Parallelization



In this scenario, the template has been updated to add new security groups to the instance stacks. With one change to the template, all four environments can have the new security group resource added.

This feature provides the benefit of easier maintainability of resources, as well as greater consistency and a reduction in effort through parallelization.



Part 2:

Infrastructure as code on AWS : *AWS CloudFormation*

© 2017 Amazon Web Services, Inc. or its affiliates. All rights reserved.

amazon
webservices | Academy

CloudFormation: Infrastructure As Code

AWS CloudFormation allows you to **launch, configure, and connect AWS resources** with JavaScript Object Notation (JSON) templates.

Template



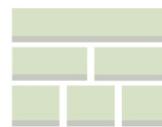
- JSON-formatted file describing the resources to be created
- Treat it as source code: put it in your repository

AWS CloudFormation Engine



- AWS service component
- Interprets AWS CloudFormation template into stacks of AWS resources

Stack



- A collection of resources created by AWS CloudFormation
- Tracked and reviewable in the AWS Management Console

Additional points for an AWS CloudFormation template:

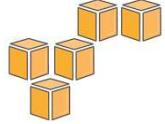
- Treat it as code and manage it using your choice of version control (e.g., Git or SVN).
- Define an entire application stack (all resources required for your application) in a JSON template file.
- Define runtime parameters for a template (Amazon EC2 Instance Size, Amazon EC2 Key Pair, etc.).

Ways To Work With CloudFormation Templates



- Simple JSON text editor
- VisualOps.io
 - APN Partner tool
 - WYSIWYG editor for CloudFormation templates
- CloudFormation Designer
 - Available via the AWS Management Console
 - Lets you drag and drop resources onto a design area to automatically generate a JSON-formatted CloudFormation template
 - Edit the properties of the JSON template on the same page
 - Open and edit existing CloudFormation templates using the CloudFormation Designer tool

AWS CloudFormation templates can be created several different ways. The most traditional is using a code editor that supports JSON syntax, such as Atom or Sublime Text. However, you can also build templates visually using our own CloudFormation Designer tool, available in the AWS Management Console, or with a third party WYSIWYG editor, such as VisualOps (www.visualops.io)



Part 3

How should resources be grouped together into templates?

Part 3: How should resources be grouped together into templates?

Organizing Your CloudFormation Templates

Assign resources to CloudFormation templates based on **ownership and application lifecycles**.

At a minimum separate network resources, security resources, and application resources into their own templates.

Example: a network resource template named “`NetworkSharedTierVpcIgwNat.template`” may include definitions for the following resources: VPCs, subnets, IGWs, route tables, and Network ACLs.

While templates can be re-used to create multiple environments or parts of environments, we do not recommend building all of an application's environment within one template. Resources should be grouped into templates based on their ownership and place in the application's lifecycle. For instance, a test environment and a production environment should probably not share the same templates in most cases: resources in a test environment will need to change frequently, while resources in a production environment should be relatively stable. In addition, sharing templates across management teams is not recommended as different needs and standards may impact teams inappropriately.

Organizing Your CloudFormation Templates

Avoid sharing a single template across applications for resources of the same type unless you are deliberately centralizing control of that resource type.

Example: Don't use one template to define the security groups of 10 applications.

If you have one application template that supports one application, changes to the template only affect one application. If you have one application template that supports several applications, changes to the template will affect several applications and cause them all to be retested. For this reason, we do not recommend using a single template across multiple applications.

Example CloudFormation Groups



When thinking about how to bundle resources into your CloudFormation templates, a good guideline is to organize it like it's software. Think about the tightly connected components to your infrastructure and put them in the same templates. In this example, CloudFormation resources are grouped into five different templates: Front-end services, back-end services, shared services, base network services, and identity resources.

Anatomy Of A CloudFormation Template

```
"Description" : "JSON  
string",  
"Metadata" : {  
    template metadata },  
"Parameters" : {  
    set of parameters },  
"Mappings" : {  
    set of mappings },  
"Conditions" : {  
    set of conditions },  
"Resources" : {  
    set of resources },  
"Outputs" : {  
    set of outputs }
```

Description:

- Text string that describes the template
- Literal string between 0 and 1024 bytes long
 - Cannot use a parameter or function to specify it

```
"Description" : "This template builds a  
VPC with one public and one private  
subnet",
```

For more information, see

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

Anatomy Of A CloudFormation Template

```

"Description" : "JSON
string",
"Metadata" : {
    template metadata },
"Parameters" : {
    set of parameters },
"Mappings" : {
    set of mappings },
"Conditions" : {
    set of conditions },
"Resources" : {
    set of resources },
"Outputs" : {
    set of outputs }

```

Metadata:

JSON objects that provide additional details about the template

- Settings or configuration information that some CloudFormation features need to retrieve
- Can be specified at the template or resource level

"Metadata" :

```

"Instances" : {"Description" : "<Information
about the instances>" },
"Databases" : {"Description" : "<Information
about the databases>" } }

```

Some AWS CloudFormation features retrieve settings or configuration information that you define from the Metadata section. You define this information in the following AWS CloudFormation-specific metadata keys:

AWS::CloudFormation::Init

Defines configuration tasks for the cfn-init helper script. This script is useful for configuring and installing applications on EC2 instances. For more information, see AWS::CloudFormation::Init.

AWS::CloudFormation::Interface

Defines the grouping and ordering of input parameters when they are displayed in the AWS CloudFormation console. By default, the AWS CloudFormation console alphabetically sorts parameters by their logical ID. For more information, see AWS::CloudFormation::Interface.

AWS::CloudFormation::Designer

Describes how your resources are laid out in AWS CloudFormation Designer (Designer). Designer automatically adds this information when you use it to create and update templates. For more information, see What Is AWS CloudFormation Designer?.

More here:

<http://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/metadata-section-structure.html>

Anatomy Of A CloudFormation Template

```

"Description" : "JSON
string",
"Metadata" : {
  template metadata },
"Parameters" : {
  set of parameters },
"Mappings" : {
  set of mappings },
"Conditions" : {
  set of conditions },
"Resources" : {
  set of resources },
"Outputs" : {
  set of outputs }

```

Resources:

Services (and their settings) that you want to launch in the stack

Properties:

- Each resource must be declared separately (except multiple instances of the same resource)
- Resource declaration has the resource's attributes

```

"Resources" : {
  "Logical ID" : {
    "Type" : "Resource type",
    "Properties" : {
      Set of properties } } }

```

The required Resources section declares the AWS resources that you want as part of your stack, such as an Amazon EC2 instance or an Amazon S3 bucket. You must declare each resource separately; however, you can specify multiple resources of the same type. If you declare multiple resources, separate them with commas.

For more information on how to create and use resources:

<http://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/resources-section-structure.html>

Anatomy Of CloudFormation Template: Resources

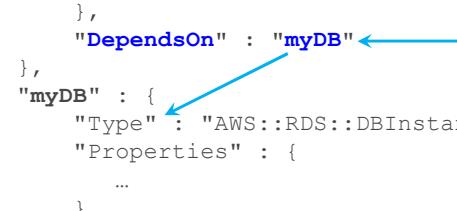
```
"Resources" : {
    "MyInstance" : {
        "Type" : "AWS::EC2::Instance",
        "Properties" : {
            "UserData" : {
                "Fn::Base64" : {
                    "Fn::Join" : [ "", [ "Queue=", { "Ref" : "MyQueue" } ] ]
                }
            },
            "AvailabilityZone" : "us-east-1a",
            "ImageId" : "ami-20b65349"
        },
        "MyQueue" : {
            "Type" : "AWS::SQS::Queue",
            "Properties" : { } }
    }
}
```

This shows a resource declaration. It defines two resources: an Amazon EC2 instance ("MyInstance") and an Amazon SQS queue ("MyQueue"). The MyInstance resource includes the MyQueue resource as part of its UserData property, as well as specifications for its AMI ID and Availability Zone (these properties could also be set in the Parameters or Conditions sections).

Resource Attribute: DependsOn

The "DependsOn" attribute specifies that the creation of a specific resource follows another. You can use the DependsOn attribute with any resource.

```
"Resources" : {  
    "AppServerInstance" : {  
        "Type" : "AWS::EC2::Instance",  
        "Properties" : {  
            "ImageId" : {  
                "Fn::FindInMap" : [ "RegionMap", { "Ref" : "AWS::Region" }, "AMI" ]  
            }  
        },  
        "DependsOn" : "myDB"  
    },  
    "myDB" : {  
        "Type" : "AWS::RDS::DBInstance",  
        "Properties" : {  
            ...  
        }  
    }  
}
```



Creates the Amazon EC2 instance
only after the RDS database
instance has been created.

DependsOn is how you specify that CloudFormation should wait to launch a resource until a specific, different resource has already finished being created.

When A DependsOn Attribute Is Required

The following resources depend on a VPC gateway attachment when they have an associated public IP address and are in a VPC:

- Auto Scaling group
- Amazon EC2 instances
- Elastic Load Balancing load balancers
- Elastic IP address

Some resources in a VPC require a gateway (either an Internet or VPN gateway). If your AWS CloudFormation template defines a VPC, a gateway, and a gateway attachment, any resources that require the gateway are dependent on the gateway attachment. For example, an Amazon EC2 instance with a public IP address is dependent on the VPC gateway attachment if the VPC and Internet gateway resources are also declared in the same template.

Special Resource: Wait Condition

Wait conditions are special CloudFormation resources that **pause the creation of the stack** and wait for a signal before it continues.

Use a wait condition to coordinate the creation of stack resources with other configuration actions external to the stack creation.

```
"myWaitCondition" : {  
    "Type" : "AWS::CloudFormation::WaitCondition",  
    "DependsOn" : "Ec2Instance",  
    "Properties" : {  
        "Handle" : { "Ref" : "myWaitHandle" },  
        "Timeout" : "4500"  
    }  
}
```

Wait condition that begins after the successful creation of the “Ec2Instance” resource

The AWS::CloudFormation::WaitConditionHandle type has no properties. When you reference the WaitConditionHandle resource by using the Ref function, AWS CloudFormation returns a pre-signed URL. You pass this URL to applications or scripts that are running on your Amazon EC2 instances to send signals to that URL. An associated AWS::CloudFormation::WaitCondition resource checks the URL for the required number of success signals or for a failure signal.

The timeout value is in seconds (e.g., 4500 seconds).

For more information, see

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

Using Creation Policies In A Template

Pause the creation of the stack and wait for a specified number of success signals before it continues:

```
"AutoScalingGroup": {  
    "Type": "AWS::AutoScaling::AutoScalingGroup",  
    "Properties": {  
        "AvailabilityZones": { "Fn::GetAZs": "" },  
        "LaunchConfigurationName": { "Ref": "LaunchConfig" },  
        "DesiredCapacity": "3",  
        "MinSize": "1",  
        "MaxSize": "4"  
    },  
    "CreationPolicy": {  
        "ResourceSignal": {  
            "Count": "3",  
            "Timeout": "PT15M"  
        }  
    },  
},
```

Creation policy that waits for three success signals but times out after 15 minutes.

The creation policy on the slide is associated with the creation of an Auto Scaling group. The default count is 1 and the default timeout period is five minutes (PT5M). The value for count must be an integer, and the value for timeout must be a string that is in ISO8601 duration format, in the form “PT#H#M#S” where # is the number of hours, minutes, and seconds, respectively.

Set your timeouts so that they give your resources plenty of time to get up and running. When the timeout period expires (or a failure signal is received), the resource creation fails and AWS CloudFormation rolls the stack back.

For more information, see

<https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/aws-attribute-creationpolicy.html>

What Is Going To Break?

If you share your template, what could **potentially break?**

Things that are specific to your environment, such as:

- Amazon EC2 key pairs
- Security group names
- Subnet ID
- EBS snapshot IDs

```
"Resources" : {
    "Ec2Instance" : {
        "Type" :
        "AWS::EC2::Instance",
        "Properties" : {
            "KeyName" : "MyKeyPair",
            "ImageId" : "ami-
75g0061f",
            "InstanceType" :
            "m1.medium"
        ...
    }
}
```

How can you fix this?

Parameters, Mappings, and Conditions

Anatomy Of A CloudFormation Template

```

"Description" : "JSON
string",
"Metadata" : {
  template metadata },
"Parameters" : {
  set of parameters },
"Mappings" : {
  set of mappings },
"Conditions" : {
  set of conditions },
"Resources" : {
  set of resources },
"Outputs" : {
  set of outputs }

```

Parameters:

Values you can pass in to your template at runtime

- Allow stacks to be customized at launch of a template
- Can specify allowed and default values for each parameter

You declare parameters in a template's *Parameters* object. A parameter contains a list of attributes that define its value and constraints against its value. The only required attribute is *Type*, which can be *String*, *Number*, or *CommaDelimitedList*. You can also add a *Description* attribute that tells a user more about what kind of value they should specify. The parameter's name and description appear in the Specify Parameters page when a user uses the template in the Create Stack wizard.

For more information about how to create and use parameters, see

<https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/parameters-section-structure.html>.

CloudFormation Template: Parameters Example

```
"Parameters" : {  
    "WebAppInstanceTypeParameter" : {  
        "Type" : "String",  
        "Default" : "t1.micro",  
        "AllowedValues" : ["t1.micro", "m1.small", "m1.medium"],  
        "Description" : "Enter t1.micro, m1.small, or m1.large. Default  
        is t1.micro." } }  
  
"Resources" : {  
    "WebAppInstance" : {  
        "Type" : "AWS::EC2::Instance",  
        "Properties" : {  
            "InstanceType" : { "Ref" : "WebAppInstanceTypeParameter" },  
            "ImageId" : "ami-2f726546"  
        }  
    }  
}
```

In this example, the `WebAppInstanceTypeParameter` specifies a default Amazon EC2 instance type of `t1.micro`, but allows users to choose from a `t1.micro`, `m1.small`, or `m1.medium` instance type when they invoke the template. It also provides a description, which appears in the CloudFormation Console when launching the template.

Then, when an Amazon EC2 instance is launched in the Resources section of the template, the Properties section of the instance can reference the `WebAppInstanceTypeParameter` specification. In this example, the "`WebAppInstance`" resource, which is an Amazon EC2 instance, references the `WebAppInstanceTypeParameter` specification for its instance type. You could also specify details like the range of acceptable AMI IDs, key pairs, subnets, or basically any properties a resource needs to have specified.

Anatomy Of A CloudFormation Template

```

"Description" : "JSON
string",
"Metadata" : {
  template metadata },
"Parameters" : {
  set of parameters },
"Mappings" : {
  set of mappings },
"Conditions" : {
  set of conditions },
"Resources" : {
  set of resources },
"Outputs" : {
  set of outputs }

```

Mappings:

Keys and associated values that specify conditional parameter values

```

"Mappings" : {
  "RegionMap" : {
    "us-east-1"      : { "64" : "ami-6411e20d" },
    "eu-west-1"      : { "64" : "ami-37c2f643" },
    "ap-southeast-1" : { "64" : "ami-66f28c34" },
  }
}

```

Mappings allow you to customize a resource's properties based on certain conditions. This enables you to have fine-grained control over how your templates are launched. For example, because an AMI ID is unique to a region, and the person who received your template may not necessarily know which AMI to use, you can provide the look-up list using the *Mappings* parameter.

For more information on how to create and use Mappings:

<https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/mappings-section-structure.html>

CloudFormation Template: Mappings Example

```
"Mappings" : {  
    "RegionAndInstanceTypeToAMIID" : {  
        "us-east-1": {  
            "m1.small": "ami-1ccae774",  
            "t2.micro": "ami-1ecae776"  
        },  
        "us-west-2" : {  
            "m1.small": "ami-ff527ecf",  
            "t2.micro": "ami-e7527ed7"  
        }  
    }  
}
```

Specify multiple mapping levels

Anatomy Of A CloudFormation Template

```

"Description" : "JSON
string",
"Metadata" : {
  template metadata },
"Parameters" : {
  set of parameters },
"Mappings" : {
  set of mappings },
"Conditions" : {
  set of conditions },
"Resources" : {
  set of resources },
"Outputs" : {
  set of outputs }

```

Conditions:

Control whether certain resources are created or certain properties are assigned a value during stack creation or update

The optional Conditions section includes statements that define when a resource is created or when a property is defined. For example, you can compare whether a value is equal to another value. Based on the result of that condition, you can conditionally create resources. If you have multiple conditions, separate them with commas.

You might use conditions when you want to reuse a template that can create resources in different contexts, such as a test environment versus a production environment. In your template, you can add an EnvironmentType input parameter, which accepts either prod or test as inputs. For the production environment, you might include Amazon EC2 instances with certain capabilities; however, for the test environment, you want to use reduced capabilities to save money. With conditions, you can define which resources are created and how they're configured for each environment type.

Conditions are evaluated based on input parameter values that you specify when you create or update a stack. Within each condition, you can reference another condition, a parameter value, or a mapping. After you define all your conditions, you can associate them with resources and resource properties in the Resources and Outputs sections of a template.

For more information on how to create and use Conditions:

<http://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/conditions-section-structure.html>

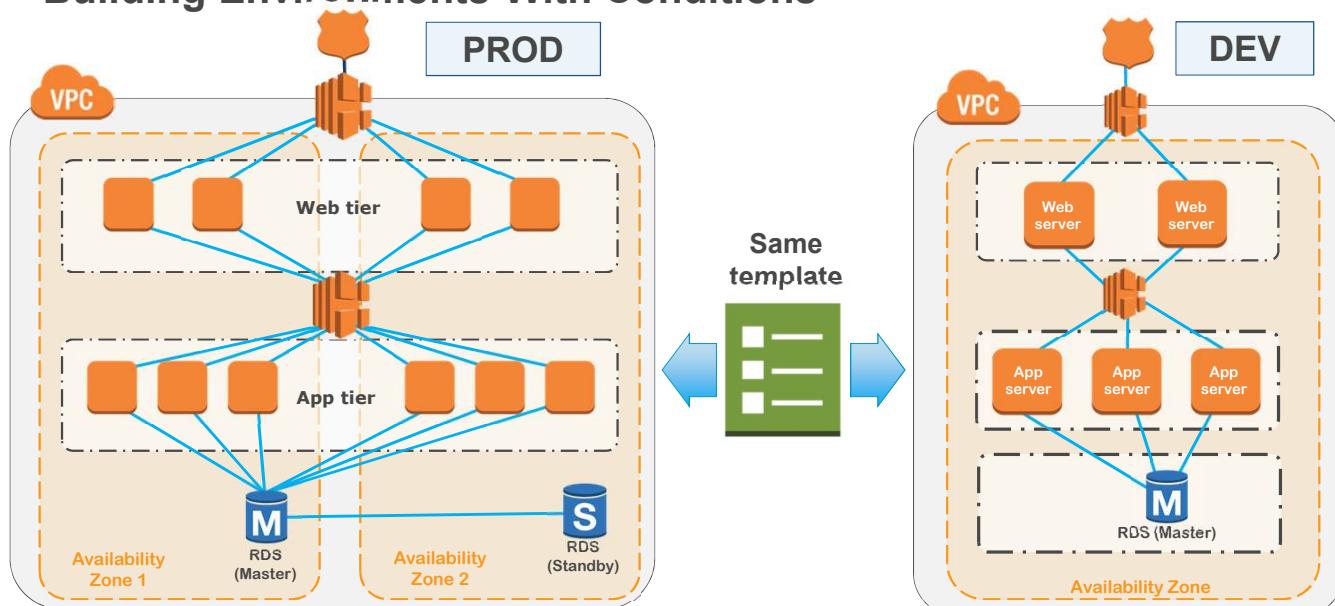
Anatomy Of CloudFormation Template: Conditions

```
"Parameters" : {  
    "EnvType" : {  
        "Description" : "Specifies if this is a Dev, QA or Prod  
environment",  
        "Type" : "String",  
        "Default" : "Dev",  
        "AllowedValues" : ["Dev", "QA", "Prod"]  
    },  
    ...  
},  
  
"Conditions" : {  
    "CreateProdResources" : {"Fn::Equals" : [{"Ref" : "EnvType"}, "Prod"]}  
},  
...
```

CreateProdResources condition evaluates to
true if EnvType is Prod

In this example, the EnvType parameter specifies whether you want to create a Dev, QA, or Prod environment. Depending on the environment, you may want to specify different configurations, such as which database it points to. You can use “Condition” to evaluate this and specify appropriate resources for each environment.

Building Environments With Conditions



© 2017 Amazon Web Services, Inc. or its affiliates. All rights reserved.



Academy

Your production environment and development environment must have the same stack in order to ensure that your application works the way it was designed. Your DEV environment and QA environment must have the same stack of applications and same configuration. There might be several QA environments for functional testing, user acceptance testing, load testing, and so on. Creating those environments manually can be error prone. You can use a *Conditions* statement in the template to solve this problem.

Anatomy Of A CloudFormation Template

```

"Description" : "JSON
string",
"Metadata" : {
  template metadata },
"Parameters" : {
  set of parameters },
"Mappings" : {
  set of mappings },
"Conditions" : {
  set of conditions },
"Resources" : {
  set of resources },
"Outputs" : {
  set of outputs }

```

Outputs:

Values returned whenever you view your stack's properties

Properties:

Declares output values that you want to view from the CloudFormation console or that you want to return in response to **describe-stack** calls

```

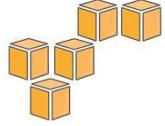
"Outputs" : {
  "<Logical ID>" : {
    "Description" : "<Information about the value>",
    "Value" : "<Value to return>" } }

```

Outputs can specify the string output of any logical identifier available in the template. It's a convenient way to capture important information about your resources or input parameters.

For more information on how to create and use Outputs, see:

<http://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/outputs-section-structure.html>



Part 4

What about resources and features not directly supported by CloudFormation?

Part 4: What about resources and features not directly supported by CloudFormation?

CloudFormation Is **Extensible** With Custom Resources

Use CloudFormation's custom resources feature to **plug in your own logic** as part of stack creation.

Examples:

1. Provision a 3rd-party application subscription and pass the authentication key back to the Amazon EC2 instance that needs it.
2. Use an AWS Lambda function to peer a new VPC with another VPC.

Custom resources enable you to write custom provisioning logic in templates that AWS CloudFormation runs anytime you create, update, or delete stacks. For example, you might want to include resources that aren't available as AWS CloudFormation resource types. You can include those resources by using custom resources. That way you can still manage all your related resources in a single stack.

Use the `AWS::CloudFormation::CustomResource` or `Custom::String` resource type to define custom resources in your templates. Custom resources require one property: the service token, which specifies where AWS CloudFormation sends requests to, such as an Amazon SNS topic.

For more on custom resources with CloudFormation, see:

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

Other infrastructure as code services on AWS:



AWS Elastic
Beanstalk



AWS
OpsWorks



Amazon
EC2 Run and
Command

Third-Party
Applications
on Amazon
EC2

While AWS CloudFormation is the most common way to deploy infrastructure automatically, there are other options within AWS that may be more suitable for specific use cases.

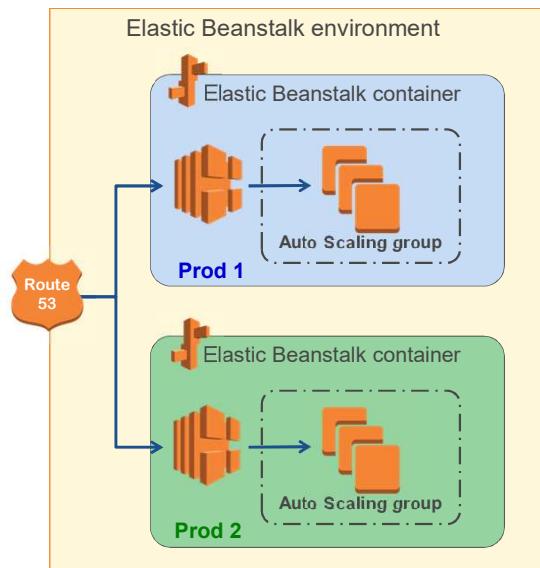
AWS Elastic Beanstalk



- AWS Elastic Beanstalk is an **automated deployment and scaling service** for web applications.
- AWS Elastic Beanstalk:
 - Accepts Java, .NET, PHP, Node.js, Python, Ruby, Go, or Docker code
 - Deploys on Apache, Nginx, Passenger, and IIS servers
- A deployment with AWS Elastic Beanstalk can automatically handle:
 - Load balancing
 - Health monitoring
 - Auto scaling
 - Application platform management
 - Code deployment

Blue-Green Deployment On AWS Elastic Beanstalk

- Fully scaled production
- Minimally scaled pre-production
- Keep both Elastic Load Balancing load balancers warm
- Roll back quickly if anything goes wrong



In case of a 24-Hour Flash Sale, you may not worry about the deployment of a web application. However, in general, you want to minimize the down-time for your web application; therefore, blue-green deployment is highly desirable.

One of the challenges with automating deployment is the cut-over itself, taking software from the final stage of testing to live production. So how can you quickly deploy without any downtime?

One simple technique is to use blue-green deployments, where the live production environment is “blue,” and the matching environment is “green.”

Green is your new deployment. You deploy updates to the green deployment and attach it to your load balancer. Once that is complete and functional, you can begin to shut down or upgrade the blue deployment. This approach also gives you the opportunity for a rapid rollback by switching back to the blue deployment if things are not working properly after switching to the green environment.

AWS Elastic Beanstalk:

Automated infrastructure management and code deployment for your application.

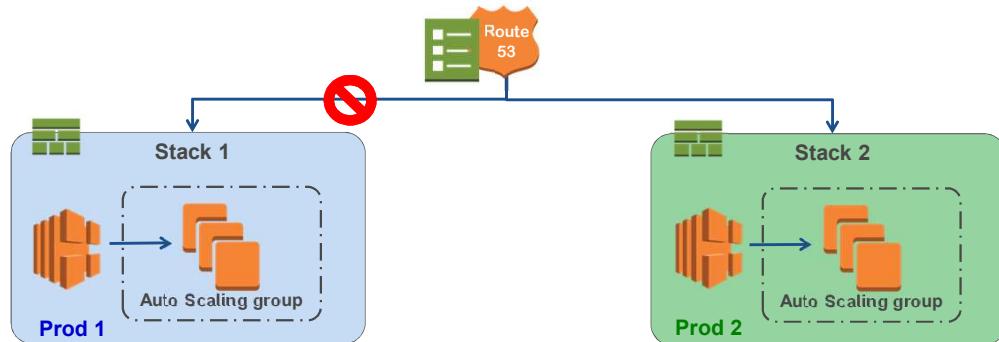
Includes:

- Load balancing
- Health monitoring
- Autoscaling
- Application platform management
- Code deployment

Blue-Green Deployment On AWS CloudFormation

It takes little more effort than the Beanstalk approach.

Customer story: An internal security team has not approved the usage of Beanstalk; therefore, the customer had to use CloudFormation to implement blue-green deployment.





AWS OpsWorks

AWS OpsWorks is a **configuration management service** that helps you configure and operate applications of all shapes and sizes using Chef.

- Define the application's whole architecture and specifications, including:
 - Package installation
 - Software configuration
 - Resources (compute, storage, etc.)
- Start from templates for common technologies (app servers, databases, etc.) or build your own.
- Use AWS OpsWorks as a lifecycle tool to:
 - Simplify management of an application
 - Reduce the number of deployment cycles

Amazon EC2 Run Command



Amazon EC2 Run Command enables you to **securely manage the configuration** of your Amazon EC2 instances. Run Command:

- Provides a simple way to automate common administrative tasks, such as:
 - Executing Shell scripts and commands on Linux.
 - Running PowerShell commands on Windows.
 - Installing software or patches.
- Allows you to execute commands across multiple instances.
- Offers visibility into the results.

3rd Party Automation Options On Amazon EC2



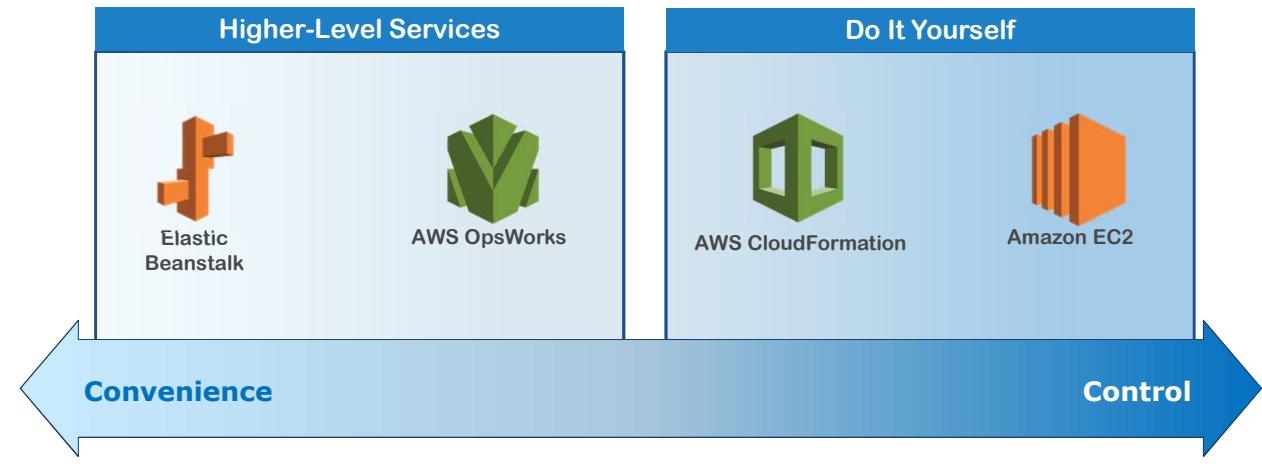
Launch and run any automation solutions on Amazon EC2, including Chef, Puppet, Ansible, and Salt.

- These solutions offer a variety of launch and configuration automation options.
- 1-click deployment for Chef is available via the AWS Marketplace.

For more on using these solutions on AWS, see:

- <https://www.chef.io/solutions/aws/>
- <https://puppet.com/product/managed-technology/aws>
- <https://www.ansible.com/aws>
- <https://docs.saltstack.com/en/latest/topics/cloud/aws.html>

Choosing The Right Solution



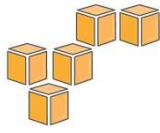
© 2017 Amazon Web Services, Inc. or its affiliates. All rights reserved.



A frequently asked question is about the multiple services that provide application management capabilities, and where the line between them is. It really depends on the level of convenience and control that you need.

Elastic Beanstalk is an easy-to-use application service for building web applications with a popular container: Java, PHP, Node.js, Python, Ruby, and Docker. If you want to upload your code and don't need to customize your environment, Elastic Beanstalk may be a good choice for you.

AWS OpsWorks lets you launch an application and define its architecture and the specification of each component, including package installation, software configuration and resources, such as storage. You can use templates for common technologies (app servers, databases, etc.) or you can build your own template instead.



In review...

- 💡 Automate Your Environment
- 💡 Infrastructure as code using CloudFormation
- 💡 Using Templates
- 💡 Extending CloudFormation



Knowledge Assessment

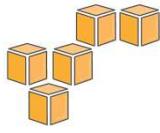
© 2017 Amazon Web Services, Inc. or its affiliates. All rights reserved.

 Academy

In review...

- Automate Your Environment
- Infrastructure as code using CloudFormation
- Using Templates
- Extending CloudFormation

To complete this module, please remember to finish the corresponding knowledge assessment.



Up Next...



LAB 10 - Creating an Amazon VPC with CloudFormation



Mid-Curriculum Project

© 2017 Amazon Web Services, Inc. or its affiliates. All rights reserved.



Before you continue, please complete **Lab 10, Creating an Amazon VPC with CloudFormation**.

Following the lab, you will complete the mid-curriculum project for Cloud Computing Architecture.

© 2017 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.

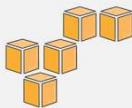
Errors or corrections? Email us at aws-course-feedback@amazon.com.

For all other questions, contact us at
<https://aws.amazon.com/contact-us/aws-training/>.

All trademarks are the property of their owners.

© 2017 Amazon Web Services, Inc. or its affiliates. All rights reserved.





CCA Unit 3 – Architecting on AWS

CCA 3.06: Decoupling Your Infrastructure

Section 1: Introduction to System Design

Section 2: Automation and Serverless Architectures

CCA 3.04 Event-Driven Scaling

CCA 3.05 Automating Your Infrastructure

► **CCA 3.06 Decoupling Your Infrastructure**

CCA 3.07 Designing Web-Scale Storage

Section 3: Well-Architected Best Practices

Section 4: Deployment and Implementation

© 2017 Amazon Web Services, Inc. or its affiliates. All rights reserved.



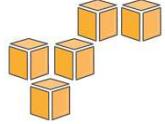
Welcome to CCA-3.06, Decoupling Your Infrastructure.

What's In This Module

- Decoupling Your Components
- Infrastructure as Code
- Using Simple Queue Service (SQS)
- Loose Coupling with DynamoDB
- Amazon API Gateway
- Processing Data with Lambda
- Decoupling Examples

This module covers...

- Decoupling Your Components
- Infrastructure as Code
- Using Simple Queue Service (SQS)
- Loose Coupling with DynamoDB
- Amazon API Gateway
- Processing Data with Lambda
- Decoupling Examples



Part 1

Decoupling Your Components

© 2017 Amazon Web Services, Inc. or its affiliates. All rights reserved.

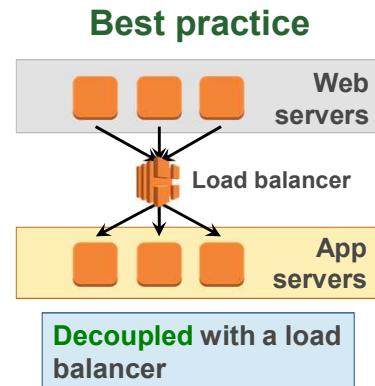
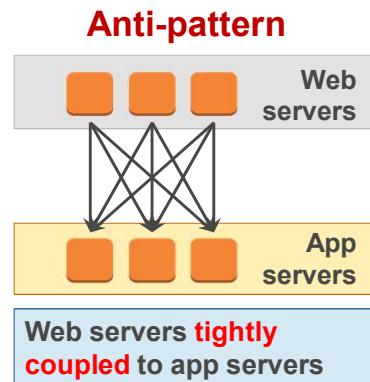
 Academy

Part 1: Decoupling Your Components

Best Practice: Loosely Couple Your Components

Design architectures with independent components.

Reduce interdependencies so that the change or failure of one component does not affect other components.



Traditional infrastructures revolve around chains of tightly integrated servers, each with a specific purpose. When one of those components/layers goes down, however, the disruption to the system can ultimately be fatal. Additionally, it impedes scaling; if you add or remove servers at one layer, every server on every connecting layer has to be connected appropriately also.

With loose coupling, where possible, you leverage managed solutions as intermediaries between layers of your system. This way, failures and scaling of a component or a layer are automatically handled by the intermediary.

Two of the primary solutions for decoupling your components are load balancers and message queues.

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, an Amazon Elastic Load Balancing instance) that routes requests between the web servers and the app servers. On the right, if one of the app servers goes down, the load balancer will automatically start directing all traffic to the two healthy servers. On the left, if one of the app servers goes down, the connection between the web servers and that server will cause an error as they try to access the unhealthy app server.

Decoupling

The more loosely your system is coupled...
the more easily it scales.



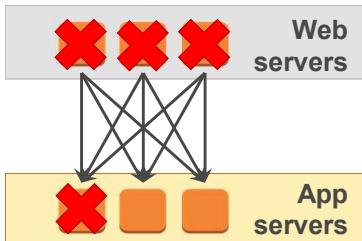
As demonstrated in the example above, loose coupling simplifies the process of scaling.
If you're using a distributed system at multiple tiers, a managed load balancer or
queueing system provides a single endpoint where traffic can go and be retrieved.

Decoupling

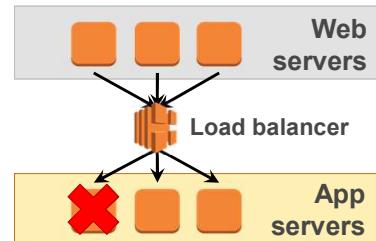
The more loosely your system is coupled...

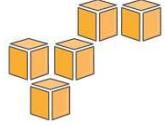
the more easily it scales.
the more fault-tolerant it can be.

Tightly coupled



Loosely coupled





Part 2

Loose Coupling Strategies

© 2017 Amazon Web Services, Inc. or its affiliates. All rights reserved.

 Academy

Part 2: Loose Coupling Strategies

Best Practice: Design Services, Not Servers

Leverage the breadth of AWS services; don't limit your infrastructure to servers.

Managed services and serverless architectures can provide greater **reliability** and **efficiency** in your environment.

Anti-pattern

- Simple applications run on persistent servers.
- Applications communicate directly with one another.
- Static web assets are stored locally on instances.
- Back-end servers handle user authentication and user state storage.

Best practice

- Serverless solution is provisioned at time of need.
- Message queues handle communication between applications.
- Static web assets are stored externally, such as on Amazon S3.
- User authentication and user state storage are handled by managed AWS services.

Although Amazon EC2 offers a tremendous amount of flexibility regarding how to set up your solution and what solutions are possible, it shouldn't be the first solution for all needs. Serverless solutions and managed services offered by AWS can solve many needs without your having to provision, configure, and manage an entire Amazon EC2 instance. Solutions like AWS Lambda, Amazon Simple Queue Service, Amazon DynamoDB, Elastic Load Balancing, Amazon Simple Email Service, and Amazon Cognito can replace server-based solutions at a lower cost with managed solutions that have a lower profile and are more performant.

Implement A Service-Oriented Architecture (SOA)

- An architectural approach in which application components **provide services** to other components via a communications protocol.
- **Services** are self-contained units of functionality.

Microservices Architectures And Decoupling

- Small, independent processes within an SOA.
- Each process is focused on doing one small task.
- Processes communicate to each other using language-agnostic APIs.

What do you get with Microservices? Microservices are independently scalable, which makes it easier to expand or shrink the relevant components of your system while leaving the rest untouched.

A system that employs microservices can more easily withstand incidents. Availability can be degraded gracefully. There should no cascading failure. Your system becomes fault tolerant, built with failure in mind.

These services are meant to deliver the base functionality required. Further customization is performed at higher application layers.

Microservices are swappable. Updating or upgrading a specific part translates to replacing one component with another one.

Microservices are minimal. *Minimal* means “small enough to be rewritten in a few days” to some people; to others it means that they do one thing and do it right. It can also mean “as small as possible while adding business value.” Ultimately, “minimal” is defined by your organization as *what makes sense*. Teams start feeling when a service is not so micro.

The clear advantage of microservices is that you have a smaller surface area of code to maintain.

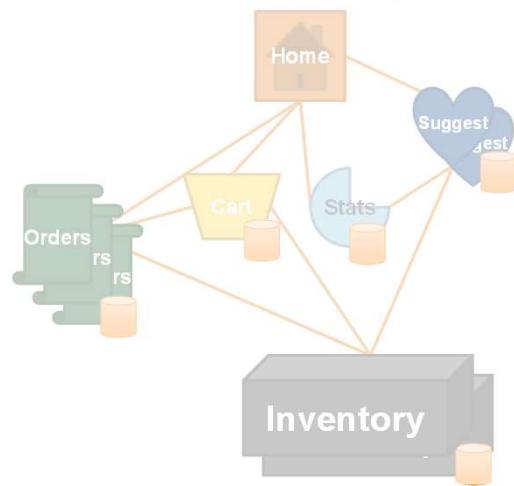
Microservices should also be complete. You can build with no external dependencies; all prerequisites are included in the microservice, which enables loose coupling.

Comparing Architectural Styles

Traditional app architectures are **monolithic**:



Microservice-based architectures are **loosely coupled**:



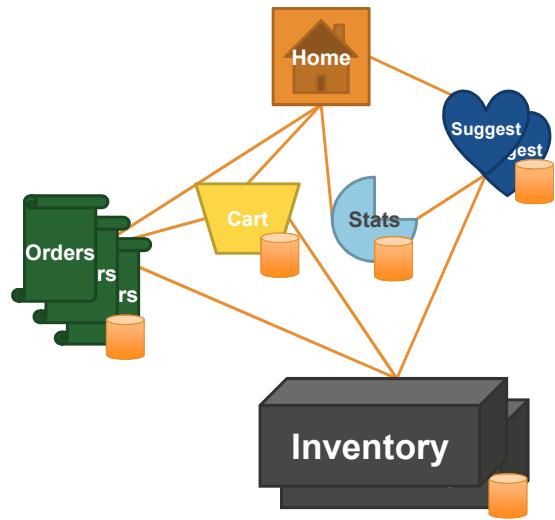
© 2017 Amazon Web Services, Inc. or its affiliates. All rights reserved.

Traditional application architectures are **monolithic**: all components of the system are tightly coupled together.

Microservice-based architectures think of components as independent services connected together in a loosely coupled manner.

Microservices

- Split features into individual components
- Have smaller parts to iterate on
- Have a reduced test surface area
- Benefit from a lower risk of change
- Use individual horizontally scalable parts



What are the advantages of microservices?

You can create applications made of small independent components, which constitute smaller parts to iterate on. This means that the cost, size and risk of change is reduced, which increases the rate of change.

Microservices Concepts: Bounding

💡 Each business domain can be divided into **contexts**.

- Contexts are made up of microservices.
- Each context has its own functions, objects, and language.
- Contexts can comprise single operations like:
 - Transaction handling
 - User registration, authentication, tracking
 - Content publishing or syndication
 - Catalog, warehouse management

Context

The other overarching concept of microservices is *bounded contexts*, which are the blocks that combine together to make one business domain. A business domain could be something like car manufacturing, book selling, or social network interactions; anything that involves a complete business process.

We can divide business domains into bounded contexts, each with its own functions, objects, and language. Each context has clear boundaries (warehouse vs. order handling), and within each context you have a finite set of functionalities. Contexts have interfaces with other contexts (e.g., ordering impacts the warehouse), which means that these contexts communicate with each other, and do so in terms of business concepts. Some data or processes are not meant to be shared; they're useful internally, and kept within the boundaries. Think carefully about each context regarding which parts need to be exposed and which can remain completely bounded.

Best Practices

- Change components without breaking them:
 - The interface is a contract
 - Modifying capabilities should not affect consumers
- Use a simple API:
 - Lowers the cost of using your service
 - More complexity means more resistance to change
 - The less you share, the less will break
 - Allows you to hide the details
- Keep it technology-agnostic:
 - Enable change: be ready to embrace the next evolution
 - Be tech-omnivorous
- Design with failure in mind:
 - "Everything fails, all the time"
- Monitor your environment:
 - Not just the infrastructure
 - Extracting health status means collecting it from services

In order to replicate the business domain you need to integrate the services that you have built. How can you do this in the best possible fashion? First, always be aware that you are maintaining a contract with your subscribers/clients. You want to iterate faster but not break functionality in the process. The interface is crucial. You need to be considerate of other teams that depend on your product.

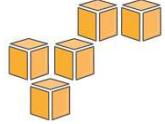
Second, make the API as simple as possible. This way it will be easier to maintain and change later. Because interfacing is so important, choosing a unified solution for all this inter-service communication to occur is tempting. Be careful to avoid integrating with smart, controlling, complex middleware which would force tighter coupling.

Best Practices

- Change components without breaking them:
 - The interface is a contract
 - Modifying capabilities should not affect consumers
- Use a simple API:
 - Lowers the cost of using your service
 - More complexity means more resistance to change
 - The less you share, the less will break
 - Allows you to hide the details
- Keep it technology-agnostic:
 - Enable change: be ready to embrace the next evolution
 - Be tech-omnivorous
- Design with failure in mind:
 - "Everything fails, all the time"
- Monitor your environment:
 - Not just the infrastructure
 - Extracting health status means collecting it from services

Microservices allow you to choose the best technology for your workload. For example you might use in-memory cache such as Memcached or Redis for fast access to your data, but in another service you might be better served with a traditional relational database. Same goes for your chosen programming language and other technology choices: remaining flexible about them is both a benefit and a mandate.

Good monitoring is the difference between reacting to an outage and proactively preventing one with proper rerouting, scaling and managed degradation. You want services to offer and push their own health status to the monitoring layer, because who knows better about status than the service itself? This can be done in many ways, such as plugins, or by writing to a monitoring API.



Part 3

What can be used to easily and reliably communicate between components?

Amazon Simple Queue Service (SQS)

© 2017 Amazon Web Services, Inc. or its affiliates. All rights reserved.

 Academy

Part 3: Simple Queue Service (SQS)

Amazon Simple Queue Service (SQS)



Amazon SQS is a fully managed **message queueing service**. Transmit **any volume of messages** at **any level of throughput** without losing messages or requiring other services to be always available.

Messages



- Generated by one component to be consumed by another.
- Can contain 256 KB of text in any format.

Amazon SQS



- Ensures delivery of each message at least once.
- Supports multiple readers and writers on the same queue.
- Does not guarantee first in, first out.

Queues



- Repository for messages awaiting processing.
- Acts as a buffer between the components which produce and receive data.

Amazon SQS is a fully managed **message queueing service**. Transmit **any volume of messages** at **any level of throughput** without losing messages or requiring other services to be always available.

Messages

- Generated by one component to be consumed by another.
- Can contain 256 KB of text in any format.

Amazon SQS

- Ensures delivery of each message at least once.
- Supports multiple readers and writers on the same queue.
- Does **not** guarantee first in, first out.

Queues

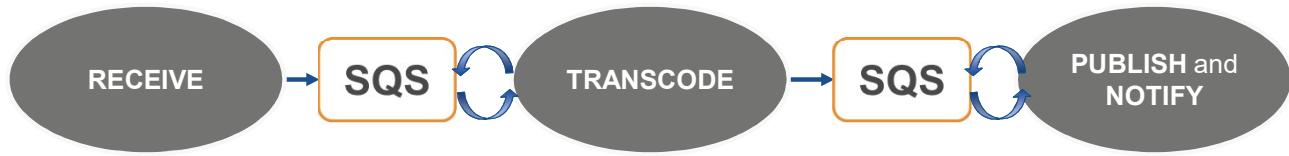
- Repository for messages awaiting processing.
- Acts as a buffer between the components which produce and receive data.

Tightly Coupled Systems



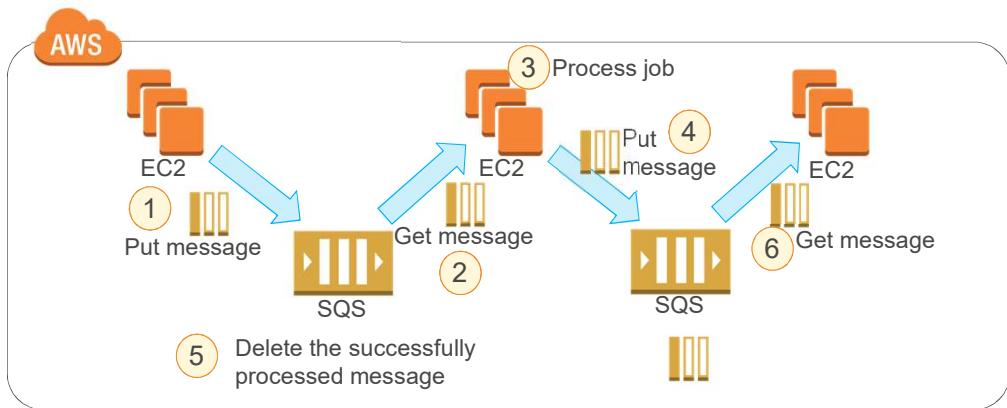
The slide shows a tightly coupled system.

Loosely Coupled Systems



Loose Coupling With Amazon SQS

The queuing chain pattern enables **asynchronous processing**:



Queuing chain pattern is one of the cloud design patterns. For more information, see http://en.clouddesignpattern.org/index.php/CDP:Queuing_Chain_Pattern

Problem to be solved

An example of image processing: the sequential operations of uploading, storing, and encoding the image, creating a thumbnail, and copyrighting are tightly linked to each other. This tight linkage complicates the recovery operations when there has been a failure.

Queuing Chain Pattern

You can achieve loose coupling of systems by using queues between systems and exchanging messages that transfer jobs. This enables asynchronous linking of systems. This method lets you increase the number of virtual servers that receive and process the messages in parallel. If there is no image to process, you can configure autoscaling to terminate the servers that are excess.

Although you can use this pattern without cloud technology, the queue itself is provided as an AWS cloud service (Amazon SQS), which makes it easier for you to use this pattern.

Benefits

- Use asynchronous processing to return responses quickly.
- Structure the system through loose coupling of Amazon EC2 instances.
- Handle performance and service requirements through merely increasing or decreasing the number of Amazon EC2 instances used in job processing.
- Even if an Amazon EC2 instance fails, a message remains in the queue service, which enables processing to be continued immediately upon recovery of the Amazon EC2 instance and produces a system that is robust to failure.

Why Choose Amazon Simple Queue Service (SQS)?

- Extremely scalable
Potentially millions of messages
- Extremely reliable
All messages are stored redundantly on multiple servers and in multiple data centers
- Simple to use
Messages get sent in, messages get pulled out
- Simultaneous read/write
- Secure
API credentials are needed

A queue can be created in any region.

Amazon SQS offers a free tier: new and existing customers receive one million queuing requests for free each month. Some applications may be able to operate within this free tier limit.

Understand The Properties Of Distributed Queues

- Message Order
- At-Least-Once Delivery
- Message Sample

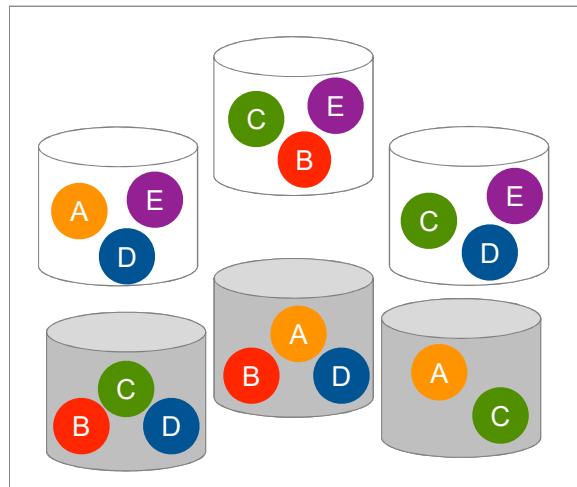
Your distributed system's components



Messages received from sampled servers



Your queue
(Distributed on SQS servers)



The diagram on the slide shows the short polling behavior of a message being returned after one of your system components makes a receive request. Amazon SQS samples several of the servers (in gray) and returns the messages from those servers (Message A, C, D, and B). Message E is not returned to this particular request, but it would be returned to a subsequent request.

Message Order

Amazon SQS makes a best effort to preserve order in messages, but because of the distributed nature of the queue, we cannot guarantee that you will receive messages in the exact order you sent them. If your system requires that order be preserved, we recommend that you place sequencing information in each message so you can reorder the messages upon receipt.

At-Least-Once Delivery

Amazon SQS stores copies of your message on multiple servers for redundancy and high availability. On rare occasions, one of the servers that is storing a copy of a message might be unavailable when you receive or delete the message. If that occurs, the copy of the message will not be deleted on that unavailable server, and you might get that message copy again when you receive messages. Because of this, you must design your application to be idempotent (i.e., it must not be adversely affected if it processes the same message more than once).

Message Sample

The behavior of retrieving messages from the queue depends on whether you are using short (standard) polling—the default behavior—or long polling. With short polling, when you retrieve messages from the queue, Amazon SQS samples a subset of the servers (based on a weighted random distribution) and returns messages from just those servers. This means that a particular receive request might not return all your messages. Or, if you have

a small number of messages in your queue (less than 1000), it means a particular request might not return any of your messages, whereas a subsequent request will. If you keep retrieving from your queues, Amazon SQS will sample all of the servers, and you will receive all of your messages.

Amazon SQS Key Feature: Visibility Timeout

Visibility timeout prevents multiple components from processing the same message.

- When a message is received, it becomes “locked” while being processed. This prevents it from being processed by other computers.
- The component that receives the message processes it and then deletes it from the queue.
- If the message processing fails, the lock will expire and the message will be available again (fault tolerance).

Visibility timeout is the period of time that a message is invisible to the rest of your application after an application component gets it from the queue. During the visibility timeout, the component that received the message usually processes it and then deletes it from the queue. This prevents multiple components from processing the same message.

When the application needs more time for processing, the “invisible” timeout can be changed dynamically via the ChangeMessageVisibility operation.

Amazon SQS Key Feature: Dead Letter Queues

A **dead letter queue** is a queue of messages that **could not be processed**.

Why use a dead letter queue?

It can sideline and isolate the unsuccessfully processed messages.

Note: A dead letter queue must reside in the same account and AWS region as the other queues that use the dead letter queue.

A dead letter queue (DLQ) receives messages after a maximum number of processing attempts has been reached. It provides the ability to isolate messages that could not be processed for analysis out of band.

A DLQ is just like any other Amazon SQS queue. Messages can be sent to it and received from it like any other SQS queue. You can create a DLQ from the SQS API and the SQS console.

Resource-based Permissions: Sharing A Queue

- Shared queues
 - Queues can be shared with other AWS accounts and anonymously.
 - A **permission** gives access to another person to use your queue in some particular way.
 - A **policy** is the actual document that contains the permissions you granted.
- Who pays for shared queue access?
 - The queue **owner** pays for shared queue access.

A developer associates an access policy statement (specifying the permissions being granted) with the queue to be shared. Amazon SQS provides APIs to create and manage the access policy statements: AddPermission, RemovePermission, SetQueueAttributes, and GetQueueAttributes. Refer to the latest API specification for more details.

Amazon SQS in each region is totally independent in message stores and queue names; therefore, the messages cannot be shared between queues in different regions.

Amazon SQS Use Cases

- Work queues
- Buffering batch operations
- Request offloading
- Fan-out
- Auto Scaling

Work Queues

Decouple components of a distributed application that may not all process the same amount of work simultaneously.

Buffering Batch Operations

Add scalability and reliability to your architecture and smooth out temporary volume spikes without losing messages or increasing latency.

Request Offloading

Move slow operations off of interactive request paths by enqueueing the request.

Fan-out

Combine Amazon SQS with Amazon SNS to send identical copies of a message to multiple queues in parallel for simultaneous processing.

Auto Scaling

Use Amazon SQS queues to help determine the load on an application, and when combined with Auto Scaling, you can scale the number of Amazon EC2 instances out or in, depending on the volume of traffic.

Other decoupling services on AWS:



Amazon Simple
Notification
Service (SNS)



Amazon
DynamoDB



Amazon API
Gateway



AWS Lambda

Amazon Simple Notification Service (SNS)



Amazon SNS enables you to **set up, operate, and send notifications** to subscribing services or other applications.

- Messages published to topic
- Topic subscribers receive message

Subscriber types:

- Email (plain or JSON)
- HTTP/HTTPS
- Short Message Service (SMS) clients (USA only)
- Amazon SQS queues
- Mobile push messaging
- AWS Lambda Function

Amazon Simple Notification Service is a web service that makes it easy to set up, operate, and send notifications from the cloud.

When using Amazon SNS, you (as the owner) create a topic and control access to it by defining policies that determine which publishers and subscribers can communicate with the topic. A publisher sends messages to topics they have created or to topics they have permission to publish to. Instead of including a specific destination address in each message, a publisher sends a message to the topic. Amazon SNS matches the topic to a list of subscribers who have subscribed to that topic, and delivers the message to each of those subscribers. Each topic has a unique name that identifies the Amazon SNS endpoint for publishers to post messages and subscribers to register for notifications. Subscribers receive all messages published to the topics to which they subscribe, and all subscribers to a topic receive the same messages.

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 more.

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.

Characteristics Of Amazon SNS



- Single published message
- Order is not guaranteed
- No recall
- HTTP/HTTPS retry
- 256 KB max per message

Single published message

All notification messages will contain a single published message.

Order is not guaranteed

Amazon SNS will attempt to deliver messages from the publisher in the order they were published into the topic. However, network issues could potentially result in out-of-order messages at the subscriber end.

No recall

When a message is delivered successfully, there is no recall feature.

HTTP/HTTPS retry

An Amazon SNS Delivery Policy can be used to control the retry pattern (linear, geometric, exponential backoff), maximum and minimum retry delays, and other parameters.

64 KB per message (non-SMS)

XML, JSON, and unformatted text.

256 KB max (four requests of 64 KB each)

Each 64-KB chunk of published data is billed as one request. For example, a single API call with a 256-KB payload will be billed as four requests.

How Is Amazon SNS Different From Amazon SQS?

Amazon SQS and Amazon SNS are both messaging services within AWS.

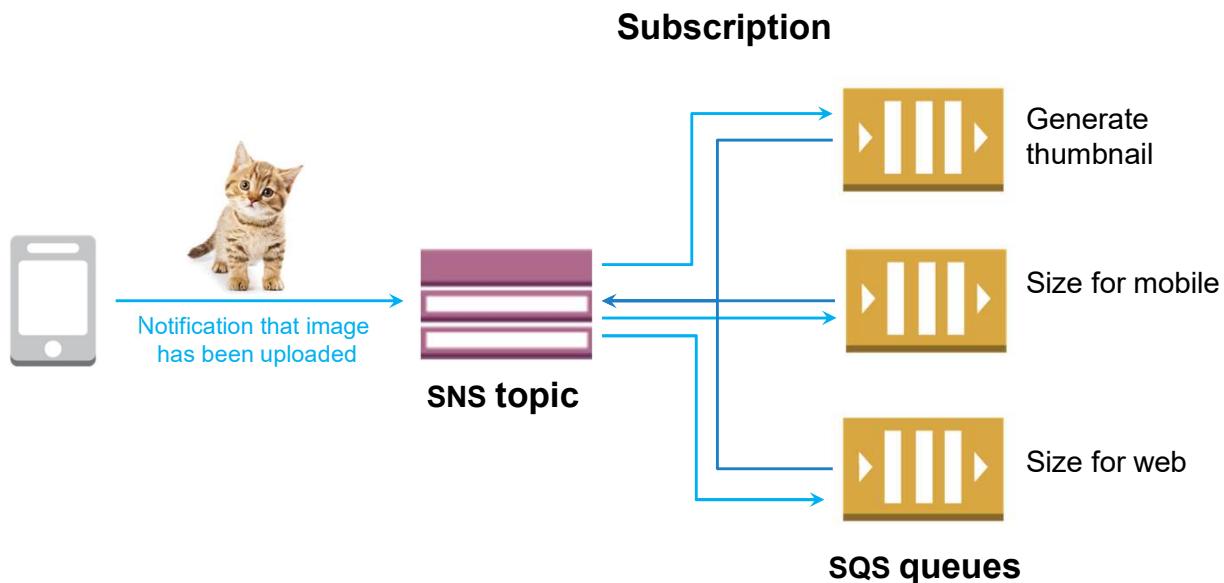
	 Amazon SNS	 Amazon SQS
Message persistence	No	Yes
Delivery mechanism	Push (Passive)	Poll (Active)
Producer/consumer	Publish/subscribe	Send/receive

Amazon SNS allows applications to send time-critical messages to multiple subscribers through a push mechanism.

Amazon SQS exchanges messages through a polling model: sending and receiving components are decoupled.

Amazon SQS provides flexibility for distributed components of applications to send and receive messages without requiring each component to be concurrently available.

Use Case: Fan-Out



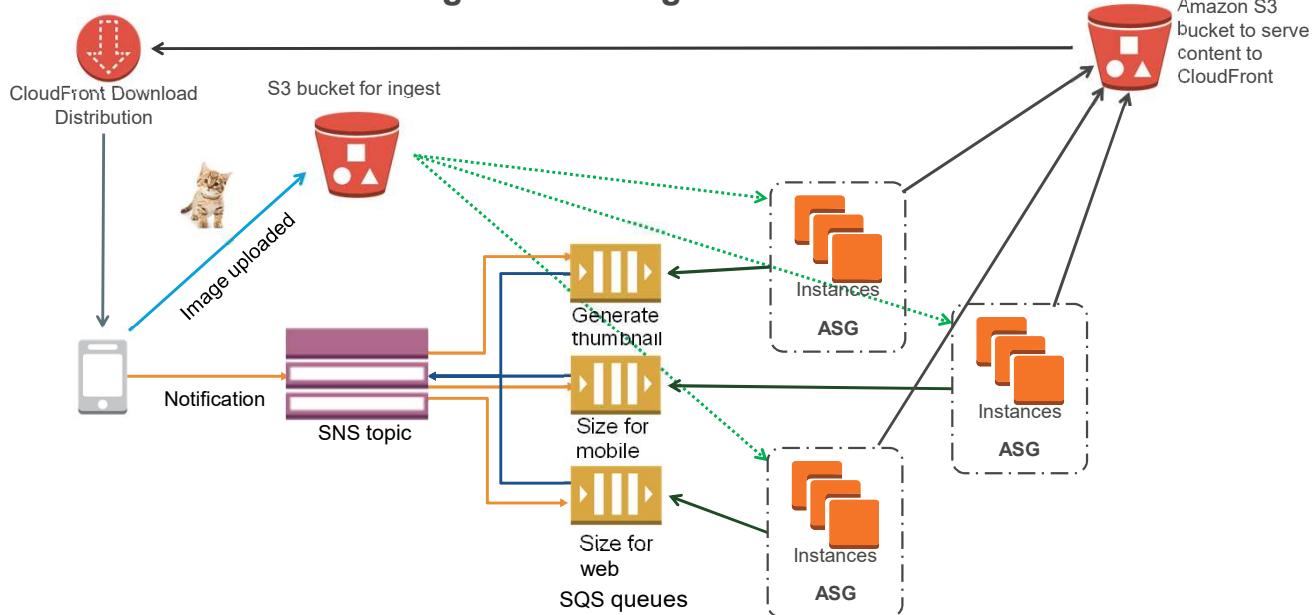
© 2017 Amazon Web Services, Inc. or its affiliates. All rights reserved.



Amazon SNS works closely with Amazon Simple Queue Service (Amazon SQS). Both services provide different benefits for developers. Amazon SNS allows applications to send time-critical messages to multiple subscribers through a “push” mechanism, which eliminates the need to periodically check or “poll” for updates. Amazon SQS is a message queue service used by distributed applications to exchange messages through a polling model and can be used to decouple sending and receiving components without requiring each component to be concurrently available. By using Amazon SNS and Amazon SQS together, messages can be delivered to applications that require immediate notification of an event and also persisted in an Amazon SQS queue for other applications to process at a later time.

When you subscribe an Amazon SQS queue to an Amazon SNS topic, you can publish a message to the topic, and Amazon SNS sends an Amazon SQS message to the subscribed queue. The Amazon SQS message contains the subject and message that were published to the topic along with metadata about the message in a JSON document.

End-to-End Scenario: Image Processing

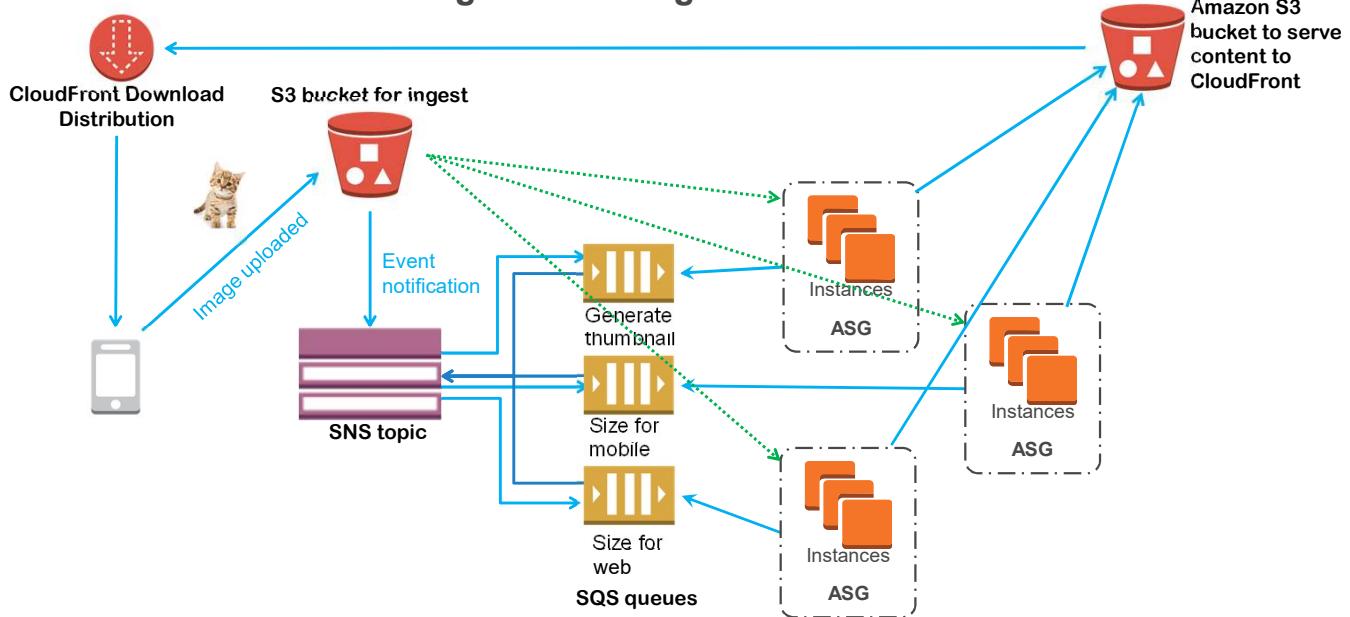


© 2017 Amazon Web Services, Inc. or its affiliates. All rights reserved.

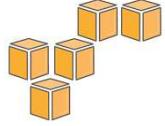


In this scenario, the user uploads the image and then manually triggers a function that sends the necessary message to a designated SNS topic, which starts the rest of the workflow.

End-to-End Scenario: Image Processing – S3 Event Notifications



In this alternative scenario, uploading the image to Amazon S3 triggers an event notification, which sends the message to the SNS topic automatically.



Part 4

Loose Coupling with DynamoDB

© 2017 Amazon Web Services, Inc. or its affiliates. All rights reserved.

 Academy

Part 4: Loose Coupling with DynamoDB

Use DynamoDB With Loosely Coupled Infrastructure



DynamoDB is a great solution for **storing and retrieving your processing output** with high throughput.

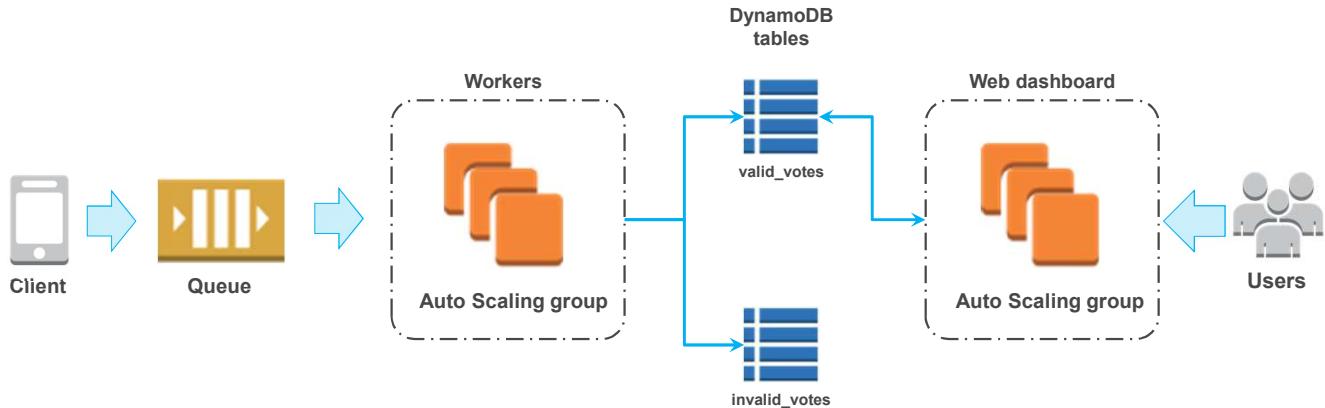
DynamoDB is:

- Highly available
- Fault-tolerant
- Fully managed

Loosely coupled systems can work very well with a managed NoSQL database solution like DynamoDB.

Example Pattern With Amazon SQS And DynamoDB

An example mobile voting platform:



TOKYU HANDS Used DynamoDB To Avoid Back-End System Outages

“

For a few hundred dollars a month, we get a fast, highly-available and scalable database. We don't need to spend any money on expensive hardware or personnel to operate the database.

Hideki Hasegawa
CTO,
TOKYU HANDS



Challenge:

- TOKYU HANDS operates 40 physical stores across Japan and Singapore and an online store worldwide, offering an enormous variety of crafting items for purchase.
- Prior to AWS, their IT systems were located in an on-premises data center which struggled to scale with spikes in traffic, particularly with their point-of-sale (POS) system.
- During their 2012 and 2013 "Hands Messe" sales events, service interruptions due to their overloaded back-end led a direct loss of revenue.

TOKYU HANDS is one of Japan's biggest and most popular retailers.

TOKYU HANDS Used DynamoDB To Avoid Back-End System Outages

Solution: DynamoDB

- In just a few months, TOKYU HANDS built a brand new POS and merchandising system around Amazon DynamoDB.
- The solution also used Amazon S3, Amazon SQS, and Amazon SNS and was hosted entirely in AWS.
- Using all of these managed solutions together enabled TOKYU HANDS to scale easily with demand, with little need to worry about component failures.

AWS offers a wide variety of fully managed services that makes it easy for us to architect our entire IT system. Amazon DynamoDB is one such service that is at the core of critical applications.

Hideki Hasegawa
CTO,
TOKYU HANDS



TOKYU HANDS Used DynamoDB To Avoid Back-End System Outages

“

Most of our team is made up of former clerks at our retail stores. They know our customers in and out, and we want them to use that knowledge to build powerful applications and not worry about infrastructure. With AWS, we are able to do just that.

Hideki Hasegawa
CTO,
TOKYU HANDS



Results:

- Using a multi-AZ architecture with DynamoDB, TOKYU HANDS's 2014 "Hands Messe" sale ensured their tables were highly available.
- Quickly scaling the throughput of their tables up prior to the sale meant that unlike in 2012 and 2013, none of their database requests were rejected due to capacity restraints.
- When the sale finished, TOKYU HANDS scaled its throughput capacity back down to save money.

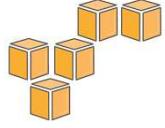
”

© 2017 Amazon Web Services, Inc. or its affiliates. All rights reserved.

Academy

For more on how TOKYU HANDS uses AWS, see:

<https://aws.amazon.com/blogs/aws/how-tokyu-hands-architected-a-cost-effective-shopping-system-with-amazon-dynamodb/>



Part 5

Amazon API Gateway

© 2017 Amazon Web Services, Inc. or its affiliates. All rights reserved.



Part 5: Amazon API Gateway



Amazon API Gateway

- Allows you to **create APIs** that act as "front doors" for your applications **to access data, business logic, or functionality** from your back-end services.
- Fully managed and handles all tasks involved in accepting and processing up to hundreds of thousands of concurrent API calls.
- Can handle workloads running on:
 - Amazon EC2
 - AWS Lambda
 - Any web application



Features Of Amazon API Gateway

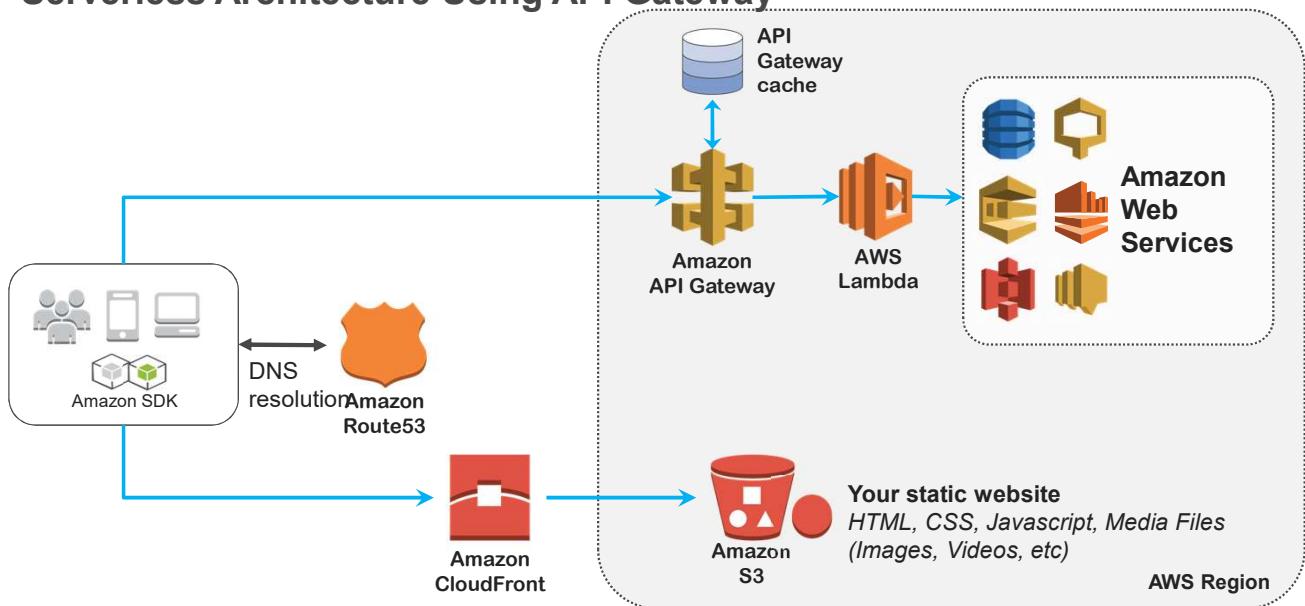
- Host and use multiple versions and stages of your APIs.
- Create and distribute API keys to developers.
- Leverage signature version 4 to authorize access to APIs.
- Throttle and monitor requests to protect your back end.
- Deeply integrated with AWS Lambda.



Benefits Of Amazon API Gateway

- Managed cache to store API responses
- Reduced latency and Distributed Denial of Service (DDoS) protection through Amazon CloudFront
- SDK generation for iOS, Android, and JavaScript
- OpenAPI Specification (Swagger) support
- Request/response data transformation

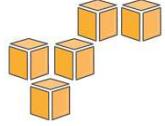
Serverless Architecture Using API Gateway



© 2017 Amazon Web Services, Inc. or its affiliates. All rights reserved.



After API Gateway receives a response to a request to your back end, it caches that response in its own cache. When the same request comes through again, API Gateway checks its cache for this request and returns that without having to check with your back end.



Part 6

Serverless architectures: Do you really need all of your instances?

AWS Lambda

© 2017 Amazon Web Services, Inc. or its affiliates. All rights reserved.

 **Amazon**
webservices | Academy

Part 6: Decoupling Using Lambda

Use AWS Lambda To Decouple Your Infrastructure



AWS Lambda is a great solution for **processing data** with high availability and a limited cost footprint.

AWS Lambda allows you to further decouple your infrastructure by **replacing traditional servers** with simple microprocesses.

Serverless Computing With AWS Lambda



AWS Lambda starts code within milliseconds of an event such as:

- An image upload
- In-app activity
- A website click
- Output from a connected device

Consider AWS Lambda if:

- You're using entire instances to run simple functions or processing applications
- You don't want to worry about HA, scaling, deployment, or management

Triggers For AWS Lambda Functions



Amazon
DynamoDB



Amazon
S3



Amazon
SNS



Amazon
Kinesis



Amazon
SES



Scheduled
Events



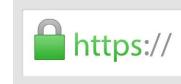
Amazon
Cognito



AWS SDKs via
Amazon API
Gateway



Amazon
CloudWatch



HTTPS via API
Gateway





How To Use AWS Lambda

1. Upload your code to AWS Lambda (in .zip form)
You can also write your code directly into an editor in the console or import it from an Amazon S3 bucket.
2. Scheduled function? Specify how often it will run.
Event-driven function? Identify the event source.
3. Specify its necessary compute resources (128MB-1.5GB of memory).
4. Specify its timeout period.
5. Specify the VPC whose resources it needs to access (if applicable).
6. Launch the function.

That's it. From there, AWS deploys and manages it for you.

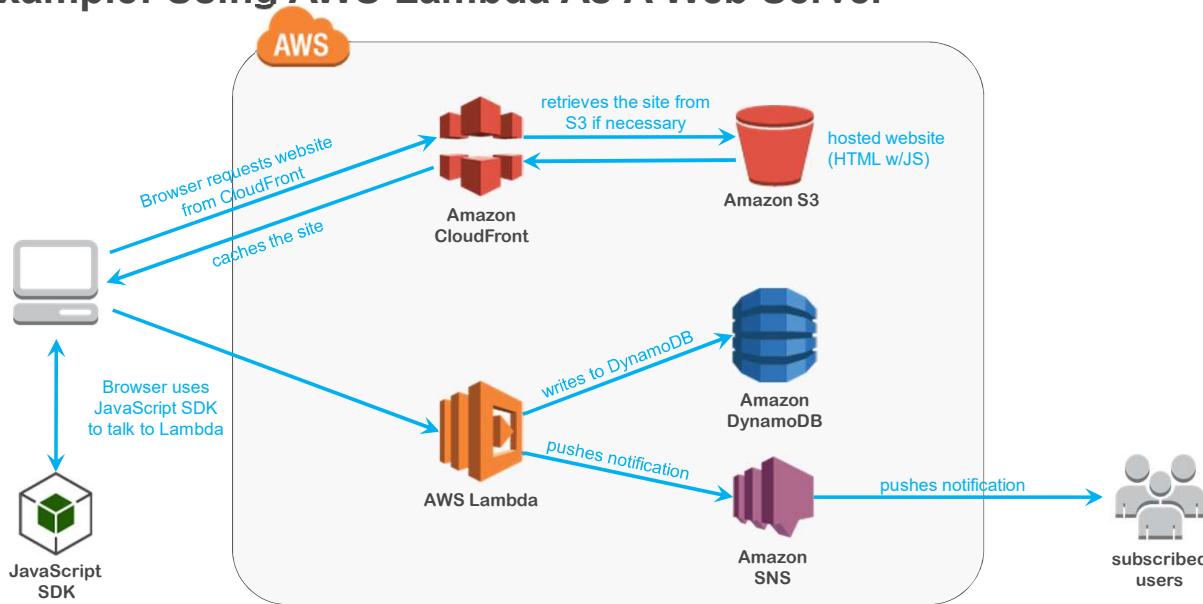
Resource Sizing With AWS Lambda

- AWS Lambda currently offers **23 different levels** of resource allocation, which range from:
 - 128 MB of memory and the lowest CPU power, to
 - 1.5 GB of memory and the highest CPU power
- **More resources = lower latency** for CPU-intensive workloads
- Compute price **scales** with resource level
- Functions can run **between 100 ms and five minutes** in length
- **Free tier:** 1M free requests and 400,000 GB-s/mo of compute time

Compute time varies based on what level of resource allocation your function runs. For example, at 128 MB, you would have 3.2 million free seconds per month, and at 1.5 GB, you would have 266,667 free seconds per month. Correspondingly, when you no longer have free tier seconds, the price per second increases with a corresponding increase in resource allocation level.

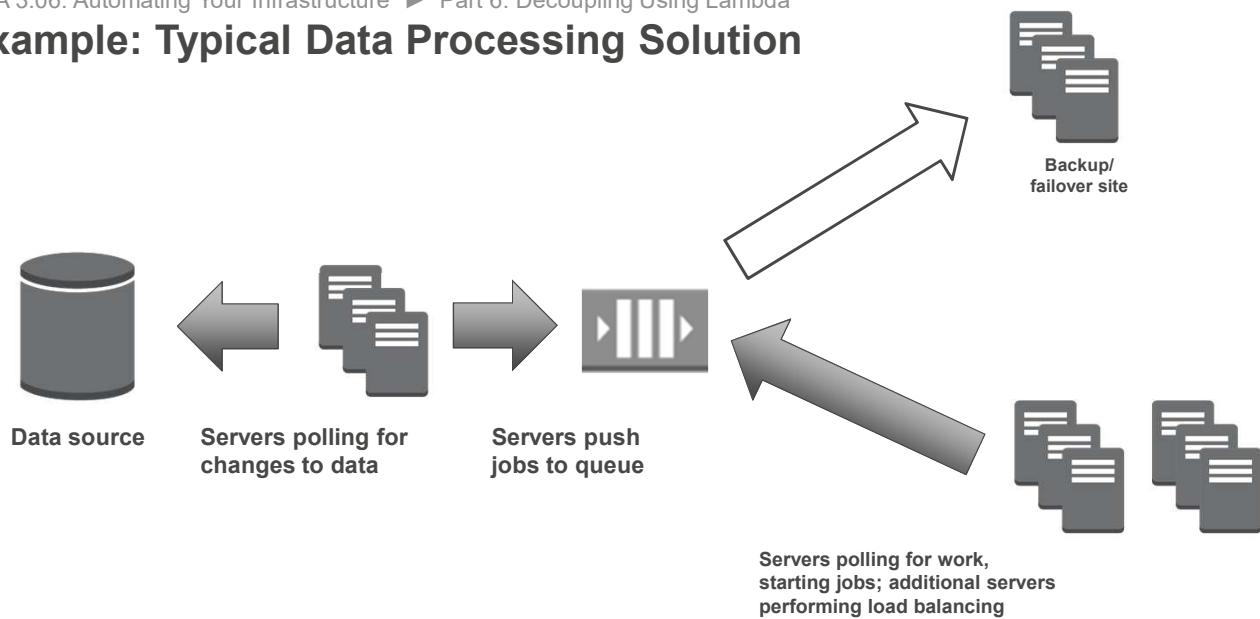
For more information about the AWS Lambda free tier, see
<https://aws.amazon.com/lambda/pricing/>

Example: Using AWS Lambda As A Web Server



© 2017 Amazon Web Services, Inc. or its affiliates. All rights reserved.

Example: Typical Data Processing Solution



Here's an example of a typical data processing solution done on-premises:

1. You start with the source of your data, be it your RDBMS, a data warehouse, a search or message service, or any other kind of large scale data store.
 2. Then, a fleet of servers are polling the data source for any changes: updated fields, new data, deleted data, etc; whatever their application is designed to look for, they're running on a constant basis for these changes to occur.
 3. When specified changes are detected, those servers push jobs to a queue service.
 4. Fleets of workers are polling this queue for work, and when jobs are discovered, taking them down and performing processing.
 5. To ensure high availability of this solution, a backup/failover site has to also be created and maintained so that work won't stop if part of the process breaks down.
- This is a very complex, multi-component system that introduces a lot of maintenance and wasted resources.

Example: Data Processing With AWS Lambda

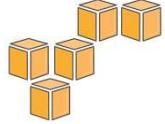


AWS Lambda handles:

- Listening/polling
- Queueing
- Processing
- Auto Scaling
- Redundancy
- Load balancing

To create the same system with AWS Lambda, here are the steps:

1. You start with the same data source as before. This could be based in AWS or not.
2. AWS Lambda runs.
3. That's it. If your application can perform its processing operations on it, AWS Lambda handles basically everything else. All you have to do is tell it what to do with the data when it's done.



Part 7

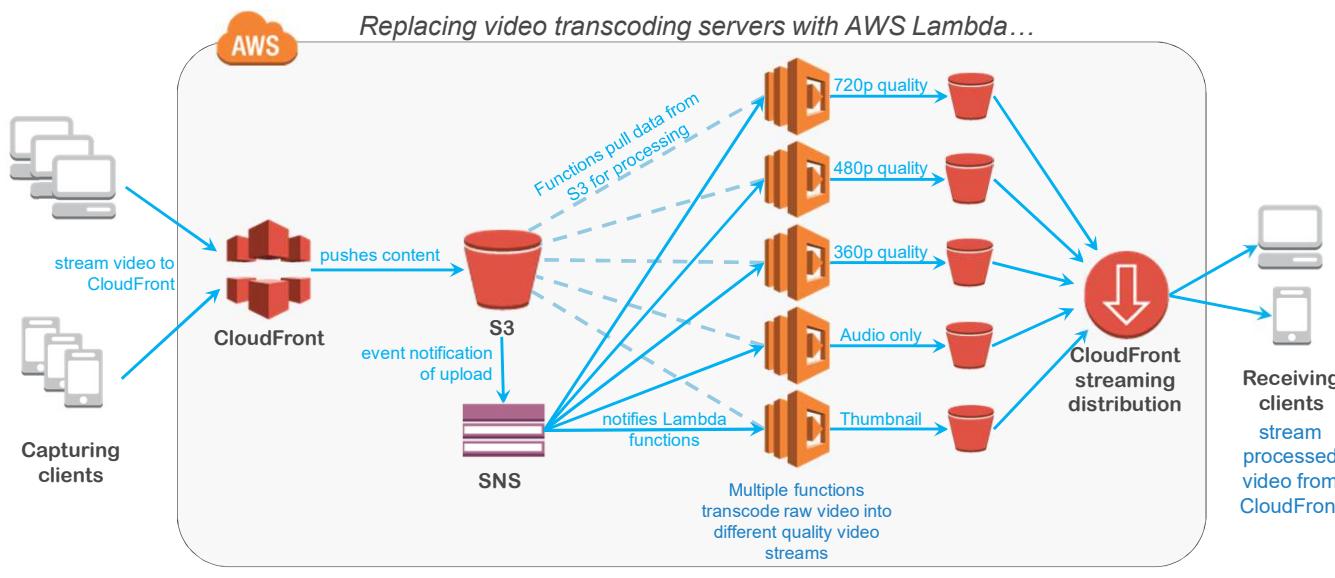
Decoupling Examples

© 2017 Amazon Web Services, Inc. or its affiliates. All rights reserved.

 **amazon**
webservices | Academy

Part 7: Decoupling Examples

Example: Livestreaming Company



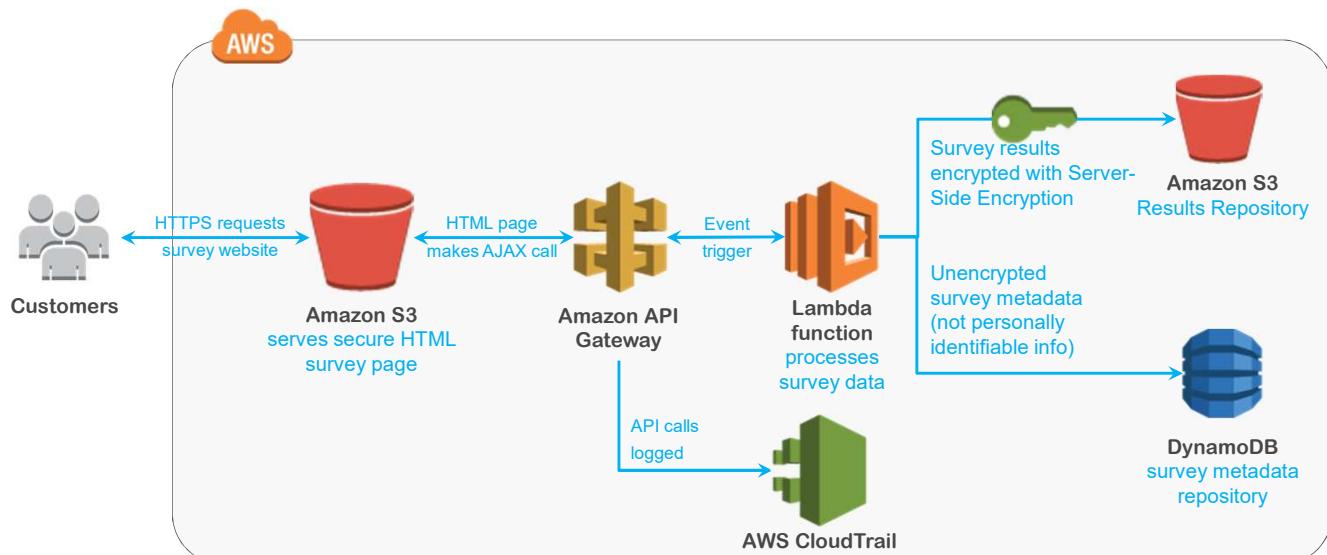
© 2017 Amazon Web Services, Inc. or its affiliates. All rights reserved.



In this example, serverless architecture, a livestreaming company replaces all of its complicated video transcoding servers with AWS Lambda.

1. First, the live streaming application encodes the video and uploads it to the closest edge location in Amazon CloudFront. This reduces the latency between the user and the AWS environment.
2. CloudFront then pushes that content into an Amazon S3 bucket.
3. When a new stream is detected by the bucket, it sends an event notification to an Amazon SNS topic.
4. When the topic receives a notification, that triggers the Lambda functions
5. The Lambda functions pull the video from the Amazon S3 bucket and perform their designated processing.
6. Each processed video file is stored into its own Amazon S3 bucket.
7. Those videos are made available via a CloudFront streaming distribution.
8. Clients can then stream the video from CloudFront to their devices.

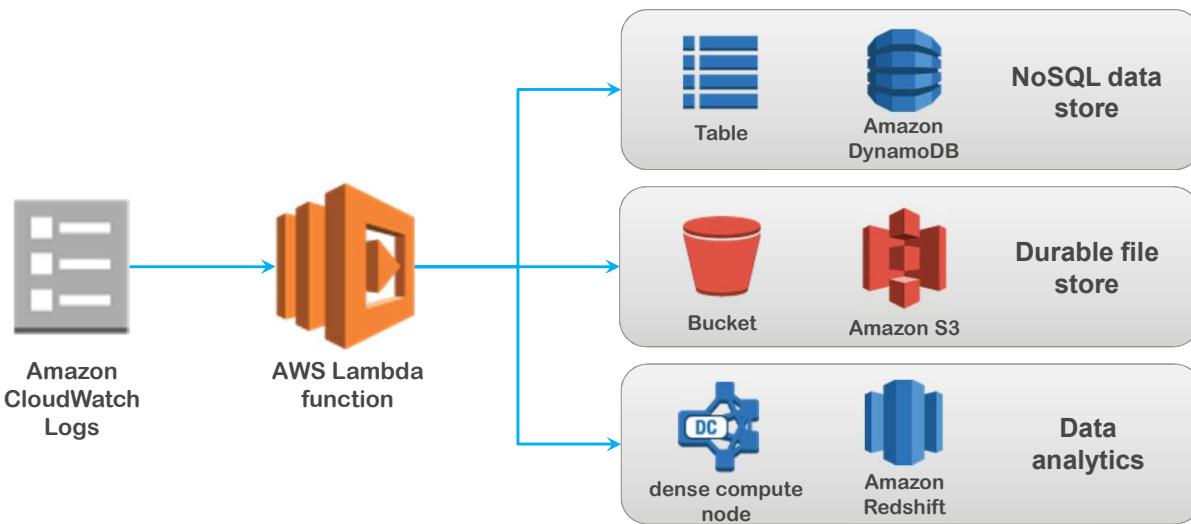
Example: Serving Secure Surveys



In this example, a serverless architecture serves, delivers, and processes secure surveys, all on managed services.

1. First, a customer requests the website over HTTPS. The webpage is served directly from Amazon S3.
2. The customer's survey is submitted via an AJAX call to Amazon API Gateway
3. Amazon API Gateway logs this to Amazon CloudTrail. In case a survey's results are lost, or one of the AJAX calls includes malicious activity of some sort, these logs may be helpful in identifying and fixing the problem.
4. Amazon API Gateway then turns the AJAX call into an event trigger for an AWS Lambda function, which pulls the survey data and processes it.
5. The results of the survey are sent by the AWS Lambda function to an Amazon S3 bucket, where they are secured with Server-Side Encryption.
6. Metadata from the survey which does not include any personally identifiable information is then written and stored in a DynamoDB table. This could be used for later queries and analysis.

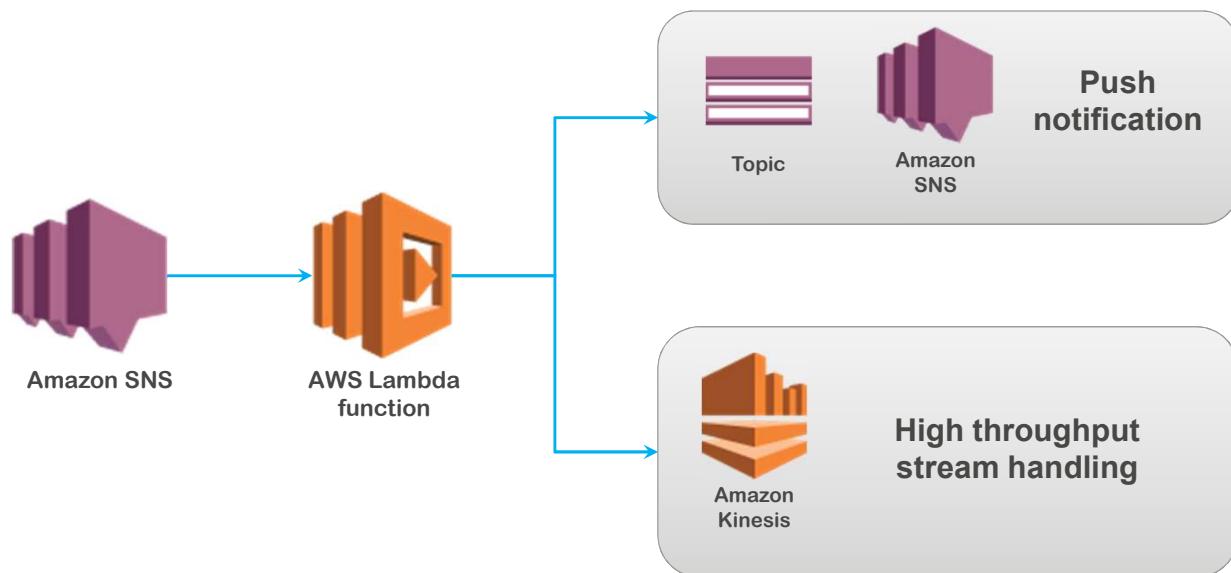
Example: Processing Amazon CloudWatch Logs



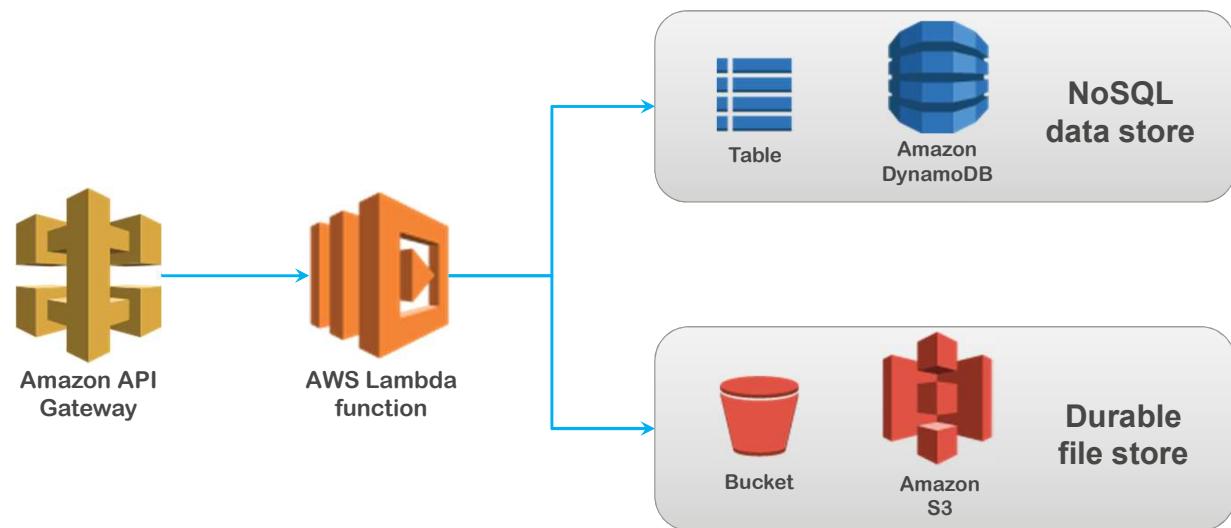
In this example, a serverless architecture processes logs for a variety of uses:

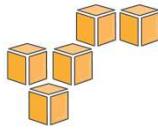
1. The Lambda function pulls a log stored as soon as it's been stored into a designated Amazon S3 bucket. From there, it could process the data in multiple useful ways.
2. It could read the log and store important findings into a DynamoDB table, where the data could be read by another application.
3. It could convert the logs into a different format and store the new log file back into Amazon S3.
4. It could also send pre-processed data to an Amazon Redshift cluster, which can then use it to perform analytics operations on the logs, providing insight for end-users in a targeted manner. Amazon Redshift will be discussed more later in the course.

Example: Real-Time Message Handling Workflow



Example: CRUD Backend Workflow





In review...

- Decoupling Your Components
- Infrastructure as Code
- Using Simple Queue Service (SQS)
- Loose Coupling with DynamoDB
- Amazon API Gateway
- Processing Data with Lambda
- Decoupling Examples



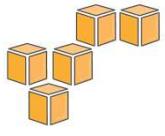
Knowledge Assessment

© 2017 Amazon Web Services, Inc. or its affiliates. All rights reserved.

In review...

- Decoupling Your Components
- Infrastructure as Code
- Using Simple Queue Service (SQS)
- Loose Coupling with DynamoDB
- Amazon API Gateway
- Processing Data with Lambda
- Decoupling Examples

To complete this module, please remember to finish the corresponding knowledge assessment.



Up Next...



CCA 3.07 - Designing Web-Scale Storage

© 2017 Amazon Web Services, Inc. or its affiliates. All rights reserved.

 **Amazon**
webservices | Academy

Up next is **CCA 3.07 - Designing Web-Scale Storage**.

© 2017 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.

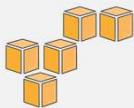
Errors or corrections? Email us at aws-course-feedback@amazon.com.

For all other questions, contact us at
<https://aws.amazon.com/contact-us/aws-training/>.

All trademarks are the property of their owners.

© 2017 Amazon Web Services, Inc. or its affiliates. All rights reserved.





CCA Unit 3 – Architecting on AWS

CCA 3.07: Designing Web-Scale Storage

Section 1: Introduction to System Design

Section 2: Automation and Serverless Architectures

CCA 3.04 Event-Driven Scaling

CCA 3.05 Automating Your Infrastructure

CCA 3.06 Decoupling Your Infrastructure

► CCA 3.07 Designing Web-Scale Storage

Section 3: Well-Architected Best Practices

Section 4: Deployment and Implementation

© 2017 Amazon Web Services, Inc. or its affiliates. All rights reserved.



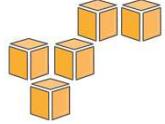
Welcome to CCA-3.07, Designing Web-Scale Storage.

What's In This Module

- How Web Apps Work
- Store Static Assets in S3
- Serve Frequently Accessed Assets from CloudFront
- Non-relational Data and DynamoDB
- Relational Data and RDS

This module covers...

- How Web Apps Work
- Store Static Assets in S3
- Serve Frequently Accessed Assets from CloudFront
- Non-relational Data and DynamoDB
- Relational Data and RDS



Part 1

How Web-based Applications Work

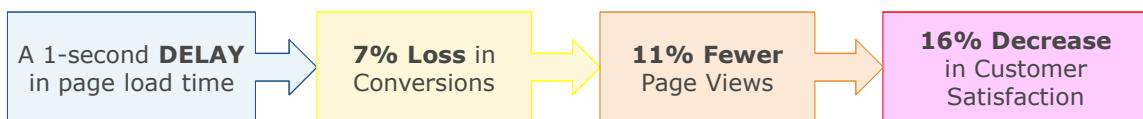
© 2017 Amazon Web Services, Inc. or its affiliates. All rights reserved.

 **Amazon**
webservices | Academy

Part 1: Automate Your Environment

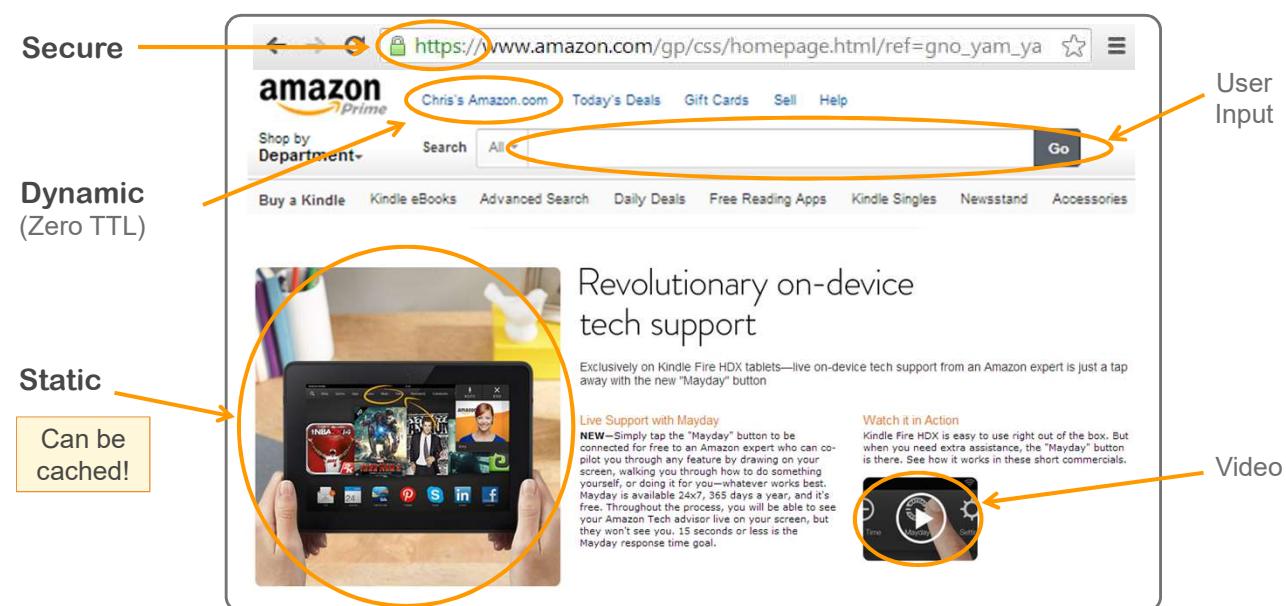
Reality Of Web-based Applications

- Your application being unavailable:
 - Leads to revenue loss
 - Has an impact on customer loyalty and your brand image
- Performance translates to:
 - High page views
 - Better customer experience
 - Higher conversion rates



by strangeloop

Deliver All Your Content



© 2017 Amazon Web Services, Inc. or its affiliates. All rights reserved.



With the caching and acceleration technology of CloudFront, AWS can deliver all of your content from static images to user-inputted content.

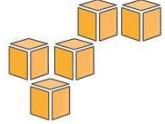
Static: images, js, html, etc. with high time-to-live (TTL)

Video: rtmp and http streaming support

Dynamic: customizations and non-cacheable content

User input: http action support including Put/Post

Secure: Serve the content securely with SSL (https)



Part 2

Store Static Assets in Amazon S3

© 2017 Amazon Web Services, Inc. or its affiliates. All rights reserved.

 Academy

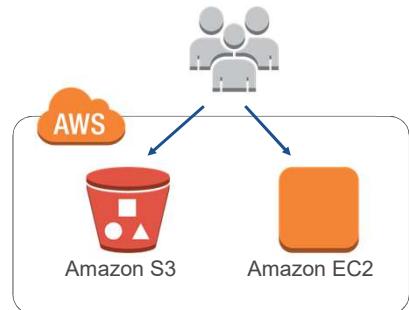
The Amazon Web Services logo, featuring the word "amazon" in lowercase with a yellow arrow above it, followed by "webservices" in smaller letters, positioned next to the word "Academy".

Part 2: Store Static Assets in Amazon S3

Web Storage Pattern

Problem

- Delivery of large files from a web server can become a problem in terms of network latency.
- User-generated content needs to be distributed across all your web servers.



Solution

- Store static asset files (CSS, multimedia, etc.) in Amazon S3 and deliver the files from there.
- Objects that are stored in Amazon S3 can be accessed directly by users if set to being public.

The use of Amazon S3 **eliminates the need to worry about network loads and data capacity on your web servers.**

Amazon S3 performs backups in at least three different data centers and thus has extremely high durability.

CloudFront CDN can be leveraged as a global caching layer in front of Amazon S3 to accelerate content to your end users.

Why Store Static Content In Amazon S3?

Items stored in Amazon S3 are highly durable:

- Amazon S3 **redundantly stores** your objects in multiple facilities in a region.
- Amazon S3 **verifies the integrity** of your data using checksums.
- Versioning allows you to **preserve, retrieve, and restore every version** of every object.

The Amazon S3 data consistency model:

- Provides **read-after-write** consistency for PUTS of new objects.
- Provides **eventual consistency** for overwrite PUTS and Deletes.

Data consistency model

- Read after Write consistency
- Read after Update not consistent
- Read after Delete not consistent
- List after Delete not consistent

How Do I Get The Most Out Of Amazon S3?

Choosing a region:

- Performance depends on the **proximity** to your users.
- Co-locating with compute and other AWS resources affects performance.

Pay attention to your **object naming scheme** if:

- You want consistent performance from a bucket.
- You want a bucket capable of routinely exceeding 100 PUT/LIST/DELETE or 300 GET requests per second.

Data stored in an Amazon S3 bucket is replicated across multiple data centers in a geographical region. Because response latencies grow when requests have to travel long distances over the Internet, your choice of which region to place your object in is important.

Naming Buckets and Keys

Although buckets can be named with any alphanumeric character, following some simple naming rules will ensure that you can reference your bucket using the convention <bucketname>.s3.amazonaws.com.

- Use 3 to 63 characters.
- Use only lowercase letters, numbers, periods (.), and hyphens (-).
- Don't start or end the bucket name with a hyphen, and don't follow or precede a period with a hyphen.

Keys can be named with any properly encoded UTF-8 character.

What If Your Bucket Is Constantly Under Load ? (1 of 3)

Amazon S3 automatically partitions your buckets according to the prefixes of your files:

Partition 1

```
/album1/photo1.jpg  
/album1/photo2.jpg  
/album1/photo3.jpg  
/album1/photo4.jpg  
  
/album2/photo1.jpg  
/album2/photo2.jpg  
/album2/photo3.jpg  
/album2/photo4.jpg
```



If a process requests the photos from album1, all requests will go to the **same partition.**

Partition 2

Anti-pattern

© 2017 Amazon Web Services, Inc. or its affiliates. All rights reserved.

 Academy

If you are storing a few objects, incrementing the last number of your file name is fine. If you store thousands of objects in an Amazon S3 bucket, this file name could limit the performance when Amazon S3 partitions your bucket. Adding a random value to the key can provide higher performance.

What if your bucket is constantly under load? (2 of 3)

You can add a hex hash prefix to the key to reduce the chance that a request reads from the same partition multiple times at once:

/2e4f/album2/photo7.jpg
/3a79/album1/photo2.jpg
/7b54/album1/photo3.jpg
/8761/album2/photo6.jpg
/921c/album1/photo4.jpg
/b71f/album1/photo1.jpg
/ba65/album2/photo5.jpg
/c34a/album2/photo8.jpg

If a process requests the first four photos, the requests will be split between the partitions.

Note: This example partitioning scheme is for demonstration purposes only. Your Amazon S3 bucket may be partitioned differently based on how its contents are named.

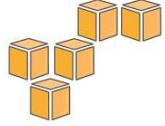
Although the example mentioned in the previous slide (increment the last number) can limit performance, it has its benefits listed on this slide. Systematic discoverable keys allow you to work with stored objects more easily.

What If Your Bucket Is Constantly Under Load? (3 of 3)

To more easily retrieve your files in useful ways, maintain a **secondary index** and hash all key names.

Example secondary index you could store in a DynamoDB table:

App ID	Month	Key
38229	04	/2e4f/var/logs/system.log.6
49113	03	/3a79/var/logs/system.log.1
49113	03	/7b54/var/logs/system.log.2
67812	04	/8761/var/logs/system.log.5



Part 3

Serve Frequently Accessed Assets from CloudFront

© 2017 Amazon Web Services, Inc. or its affiliates. All rights reserved.

 Academy

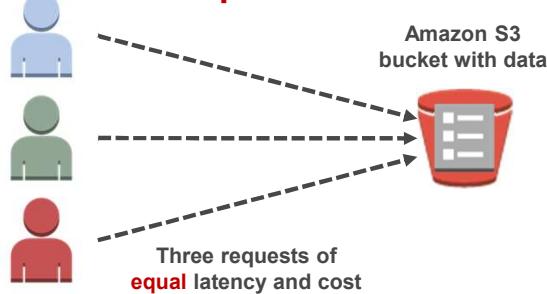
Part 3: Serve Frequently Accessed Assets from CloudFront

Best Practice: Use Caching

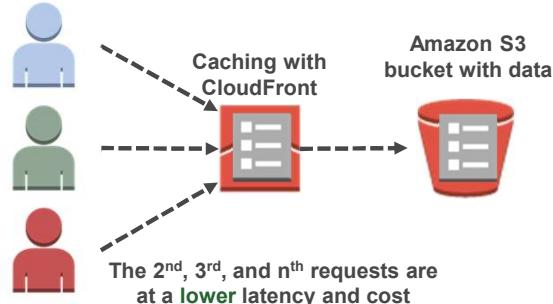
Use caching to minimize redundant data retrieval operations.

Implement caching at **multiple layers** of an architecture. It can **reduce cost and latency** and **increase performance** of applications.

Anti-pattern



Best practice

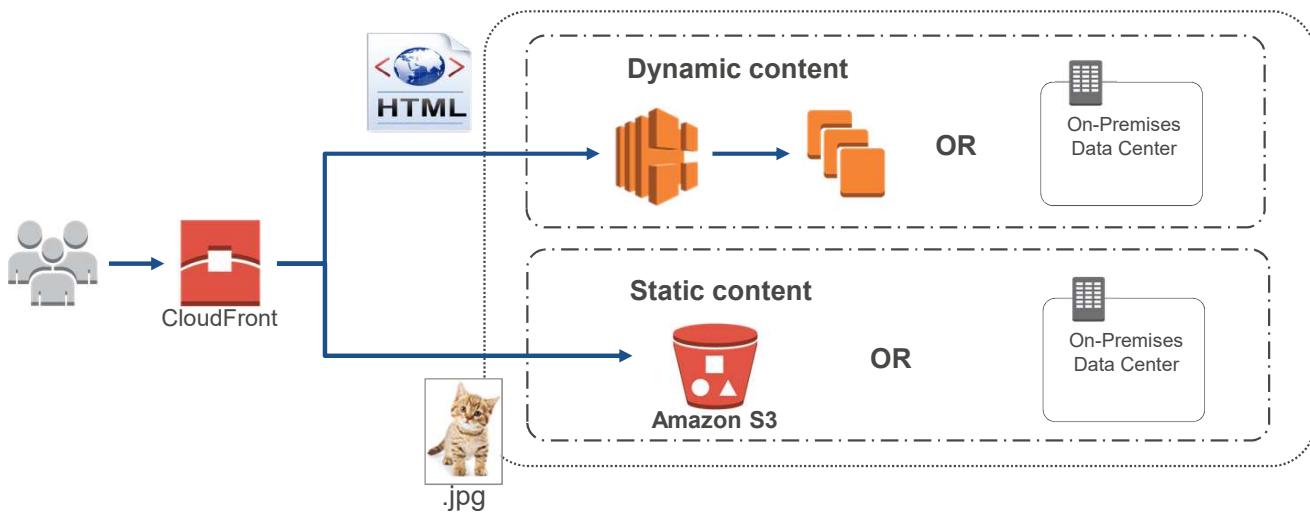


Caching is the process of temporarily storing data or files in an intermediary location between the requester and the permanent storage, for the purpose of making future requests faster and reducing network throughput.

1. For instance, in the anti-pattern above, there is no caching service being used by your Amazon S3 bucket.
2. Three users request a file from one of your Amazon S3 buckets, one at a time.
3. The file is delivered to each the same way, resulting in each request taking the same amount of time to complete and costing the same amount of money.
4. Lets compare that with a better pattern.
5. In the best practice pattern, your infrastructure has Amazon CloudFront, which offers caching, in front of Amazon S3.
6. In this scenario, the first request checks for the file in CloudFront. Upon not finding it there, it pulls the file from Amazon S3, stores a copy of the file in CloudFront at an edge location closest to the user, and then sends another copy to the user who made the request.
7. Now, when any other users request that file, it's retrieved from the closer edge location in CloudFront, rather than going all the way to Amazon S3 to get it.

This reduces both latency AND cost, since in the requests after the first one, you're no longer paying for the file to be transferred out of Amazon S3.

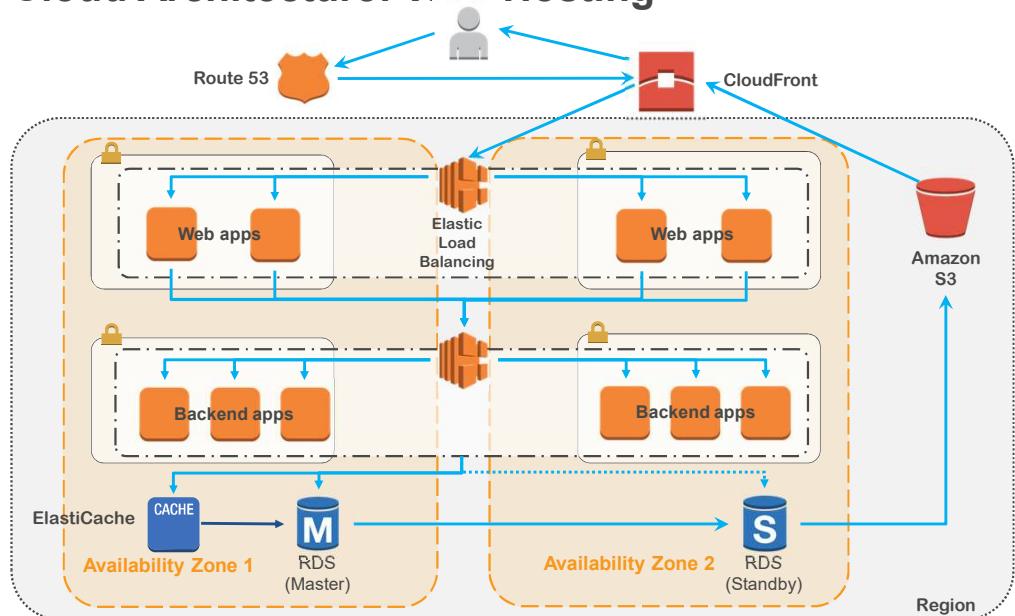
Cache Static And Re-usable Content



In general, you only cache static content; however, dynamic or unique content affects the performance of your application. Depending on the demand, you might still get some performance gain by caching the dynamic or unique content in Amazon S3.
Static content off-loading:

- Create absolute URL references to your static assets instead of relative.
- Store static assets in Amazon S3.
- Optimize for “Write Once Read Many.”

AWS Cloud Architecture: Web Hosting



© 2017 Amazon Web Services, Inc. or its affiliates. All rights reserved.



Traditional web hosting architecture implements a common three-tier web application model that separates the architecture into presentation, application, and persistence layers. Scalability is provided by adding hosts at the presentation, persistence, or application layers.

Web application hosting in the AWS cloud

The traditional web hosting architecture is easily portable to the cloud services provided by AWS products with only a small number of modifications, but the first question that should be asked concerns the value of moving a classic web application hosting solution into the AWS cloud. If you decide that the cloud is right for you, you'll need a suitable architecture.

- *Amazon Route 53* provides DNS services to simplify domain management and zone APEX support.
- *Amazon CloudFront* provides edge caching for high-volume contents.
- *Elastic Load Balancing* spreads traffic to web server Auto Scaling groups in this diagram.
- *Exterior Firewall* is moved to every web server instance via security groups.
- *Backend Firewall* is moved to every back-end instance.
- *App server load balancer* on Amazon EC2 instances spreads traffic over the app server cluster.
- *Amazon ElastiCache* provides caching services for the app, which removes load from the database tier.

How Can Amazon CloudFront Help?

Content Delivery Network (CDN)

- Your content is cached all around the world
- Geographic proximity means **lower latency**
- Better user experience
- Less stress on your core infrastructure



Independent of Region

Key Features

- TCP/IP optimizations for the network path
- Keep-alive connections to reduce round-trip time
- SSL/TLS termination close to viewers
- POST/PUT upload optimizations
- Latency-based routing

Amazon CloudFront is a web service that speeds up distribution of your static and dynamic web content—for example, .html, .css, .php, and image files—to end users. CloudFront delivers your content through a worldwide network of data centers called *edge locations*. When a user requests content that you are serving with CloudFront, the user is routed to the edge location that provides the lowest latency (time delay), so content is delivered with the best possible performance. If the content is already in the edge location with the lowest latency, CloudFront delivers it immediately. If the content is not currently in that edge location, CloudFront retrieves it from an Amazon S3 bucket or an HTTP server (for example, a web server) that you have identified as the source for the definitive version of your content.

In addition to what is listed in the slide, the key features of CloudFront are:

- RTMP (Flash) and HTTP delivery
- Live and video on-demand (VOD) streaming
- Adaptive bitrate streaming
- HTTP/HTTPS file delivery
- Private content
- Programmatic invalidation
- Industry-compliant, detailed access logs
- Full control via APIs
- Origin choice (Amazon S3, Amazon EC2, etc.)
- Typical use cases of CloudFront:

- Static websites
- Video and rich media
- Online gaming
- Interactive agencies
- Software downloads

How Do You Enable CloudFront?

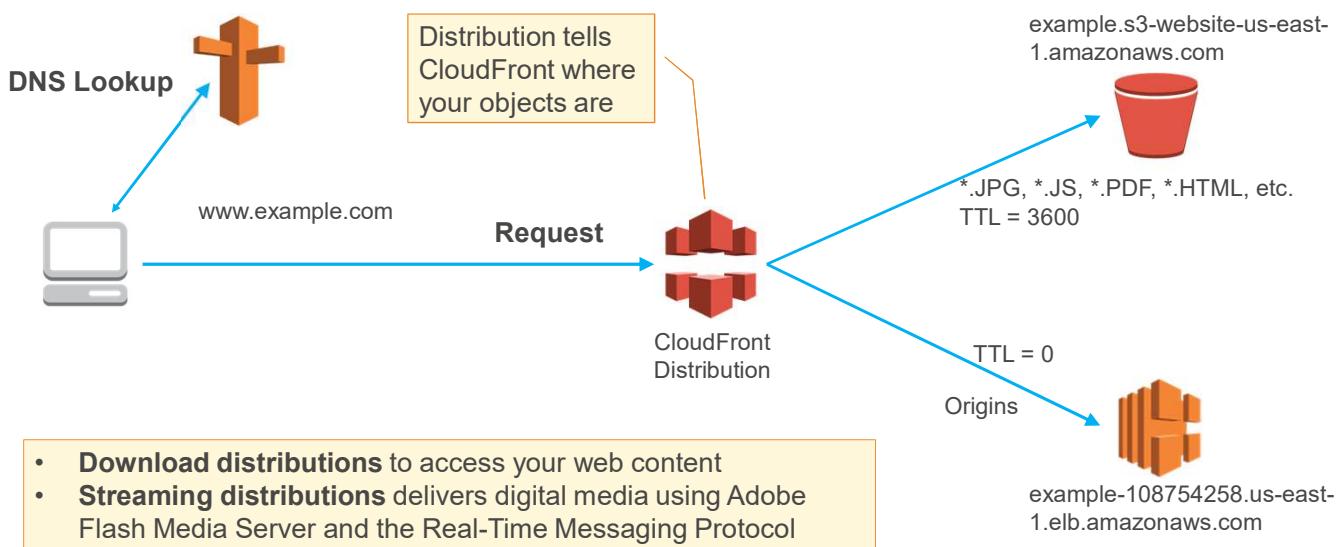
Use a separate CNAME for static content

- Static content cached, dynamic content straight from origin
- Most efficient
- More effort to set up and manage

Point entire URL to CloudFront

- Easiest to manage
- Use URL patterns to stage dynamic content
- ALL content goes through edge locations

CloudFront Distributions



When you set up CloudFront, you create what's called a *distribution*; your DNS then would point to the distribution. You configure the distribution to point to your source content, called an *origin*. If you have content in multiple locations, your distribution can have multiple origins and get routed to the correct origin based on URL pattern.

Two distribution types:

- **WEB distribution:** To access your web content in any combination of up to 10 Amazon S3 buckets and custom origins.
- **RTMP distribution:** The origin for a RTMP distribution is always an Amazon S3 bucket.

How Does CloudFront Speed Up My Website?

Use cache control headers

- Cache control header set on your files identifies static and dynamic content
- Delivering all your content using a single Amazon CloudFront distribution helps to ensure performance optimization on your entire website

How long is my file kept at the edge location?

- Set expiration period by setting the cache control headers on your files in your origin

Amazon CloudFront uses these cache control headers to determine how frequently it needs to check the origin for an updated version of that file. For an expiration period set to 0 seconds, Amazon CloudFront will revalidate every request with the origin server. If your files don't change very often, the best practice is to set a long expiration period and implement a versioning system to manage updates to your files.

By default, if no cache control header is set, each edge location checks for an updated version of your file whenever it receives a request more than 24 hours after the previous time it checked the origin for changes to that file.

How To Expire Contents

Time to Live (TTL)

- Fixed period of time (expiration period).
- Set by *you*.
- GET request to origin from CloudFront will use **If-Modified-Since** header.

Change object name

- `Header-v1.jpg` becomes `Header-v2.jpg`.
- New name forces refresh.

Invalidate object

- Last resort: very inefficient and very expensive.

There are three ways to wear out cached content: the first two are much preferred, and TTL is easiest if the replacement does not need to be immediate.

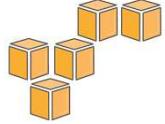
(<http://docs.aws.amazon.com/AmazonCloudFront/latest/DeveloperGuide/RequestAndResponseBehaviorS3Origin.html>)

If you set the TTL for a particular origin to 0, CloudFront will still cache the content from that origin. It will then make a GET request with an If-Modified-Since header, thereby giving the origin a chance to signal that CloudFront can continue to use the cached content if it hasn't changed at the origin.

The second way requires more effort but is immediate (there might be some support for this in some CMS systems). Although you can update existing objects in a CloudFront distribution and use the same object names, it is not recommended.

CloudFront distributes objects to edge locations only when the objects are requested, not when you put new or updated objects in your origin. If you update an existing object in your origin with a newer version that has the same name, an edge location won't get that new version from your origin until both of the listed events occur.

The third method should only be used sparingly for individual objects: it is a bad solution (the system must forcibly interact with all edge locations).



Part 4

Store Non-relational Data in a NoSQL Database Such as DynamoDB

© 2017 Amazon Web Services, Inc. or its affiliates. All rights reserved.

 Academy

The Amazon Web Services logo, featuring the word "amazon" in lowercase with a yellow smiley face icon above it, followed by "webservices" in a smaller font, and the word "Academy" to its right.

Part 4: Store Non-relational Data in a NoSQL Database Such as DynamoDB

Best Practice: Choose The Right Database Solutions

Match the technology to the workload, not the other way around.

Choose from an array of relational database engines, NoSQL solutions, data warehousing options, and search-optimized data stores.

Things to consider:

- Read and write needs
- Total storage requirements
- Typical object size and nature of access to these objects
- Durability requirements
- Latency requirements
- Maximum concurrent users to support
- Nature of queries
- Required strength of integrity controls

In traditional data centers and on-premises environments, limitations on what hardware and licenses are available can constrain your choice of data store solution. AWS offers data store options that provide more flexibility in how you choose your data store. Consequently, we recommend that you choose your data store based on your application environment's needs.

Relational/SQL And NoSQL Databases

	Relational/SQL	NoSQL								
Data Storage	Rows and Columns	Key-Value, Documents, and Graphs								
Schemas	Fixed	Dynamic								
Querying	Using SQL	Focused on collection of documents								
Scalability	Vertical	Horizontal								
Example	<table border="1"> <thead> <tr> <th>ISBN</th> <th>Title</th> <th>Author</th> <th>Format</th> </tr> </thead> <tbody> <tr> <td>9182932465265</td> <td>Cloud Computing Concepts</td> <td>Wilson, Joe</td> <td>Paperback</td> </tr> </tbody> </table>	ISBN	Title	Author	Format	9182932465265	Cloud Computing Concepts	Wilson, Joe	Paperback	<pre>{ ISBN: 9182932465265, Title: "Cloud Computing Concepts", Author: "Wilson, Joe", Format: "Paperback" }</pre>
ISBN	Title	Author	Format							
9182932465265	Cloud Computing Concepts	Wilson, Joe	Paperback							

A SQL database stores data in rows and columns. Rows contain all the information about one entry, and columns are the attributes that separate the data points. A SQL database schema is fixed: columns must be locked before data entry. Schemas can be amended if the database is altered entirely and taken offline. Data in SQL databases is queried using SQL (Structure Query Language), which can allow for complex queries. SQL databases scale vertically by increasing hardware power.

A NoSQL database stores data using one of many storage models, including key-value pairs, documents, and graphs. NoSQL schemas are dynamic. A 'row' doesn't have to contain data for each 'column'. Data in NoSQL databases is queried by focusing on collections of documents. NoSQL databases scale horizontally by increasing servers.

NoSQL Databases

- Can be an alternative to relational databases for some types of applications
- Can process large amounts of data with high availability (depending on the NoSQL solution, configuration, and architecture)
- Form a broad category with different implementations and data models
- Have the common feature of distributed fault tolerance

NoSQL Databases

Does your app need transaction support, ACID compliance, joins, SQL?

- Can it do without these for all, some, or part of its data model?
Refactor database hotspots to NoSQL solutions
- NoSQL databases can offer increases in flexibility, availability, scalability, and performance

ACID refers to atomicity, consistency, isolation, and durability, which are the four primary attributes that guarantee that database transactions are processed reliably. For more on the ACID model, see:

<http://databases.about.com/od/specificproducts/a/acid.htm>

NoSQL Databases

NoSQL on Amazon EC2:

- Cassandra, HBase, Redis, MongoDB, Couchbase, and Riak

Managed NoSQL:

- Amazon DynamoDB
- ElastiCache with Redis
- Amazon Elastic Map Reduce supports HBase

Shifting Functionality To NoSQL

- Think about *when* you need NoSQL instead of SQL.
- Leverage managed services like Amazon DynamoDB.
- **Use cases:**
 - Leaderboards and scoring
 - Rapid ingest of clickstream or log data
 - Temporary data needs (cart data)
 - Hot tables
 - Metadata or lookup tables
 - Session data

An AWS customer, Scopely, decided to move game-state information into DynamoDB when they maxed out an Amazon RDS instance. Instead of spending time sharding or optimizing RDS, they decided that a NoSQL database is exactly what they need to store the game-state information.

Key Characteristics Of DynamoDB

Fully managed NoSQL database service.

Zero Administration!

Low latency

- SSD-based storage nodes
- Latency = single-digit milliseconds

Massive and seamless scalability

- No table size or throughput limits
- Live repartitioning for changes to storage and throughput

Predictable performance

- Provisioned throughput model

Durable and available

- Consistent, disk-only writes

Amazon DynamoDB is a fast, NoSQL database service that makes it simple and cost-effective to store and retrieve any amount of data and serve any level of request traffic. Its throughput and single-digit millisecond latency make it a great fit for gaming, ad tech, mobile, and many other applications. DynamoDB delivers seamless throughput and storage scaling via API and an easy-to-use management console, so you can easily scale up or down to meet your needs. Many AWS customers have, with the click of a button, created DynamoDB deployments in a matter of minutes that are able to serve trillions of database requests per year.

DynamoDB tables do not have fixed schemas, and each item may have a different number of attributes. Multiple data types add richness to the data model. Secondary indexes add flexibility to the queries you can perform, without affecting performance. Performance, reliability, and security are built in, with SSD (solid state drive) storage and automatic three-way replication. DynamoDB uses proven cryptographic methods to securely authenticate users and prevent unauthorized data access.

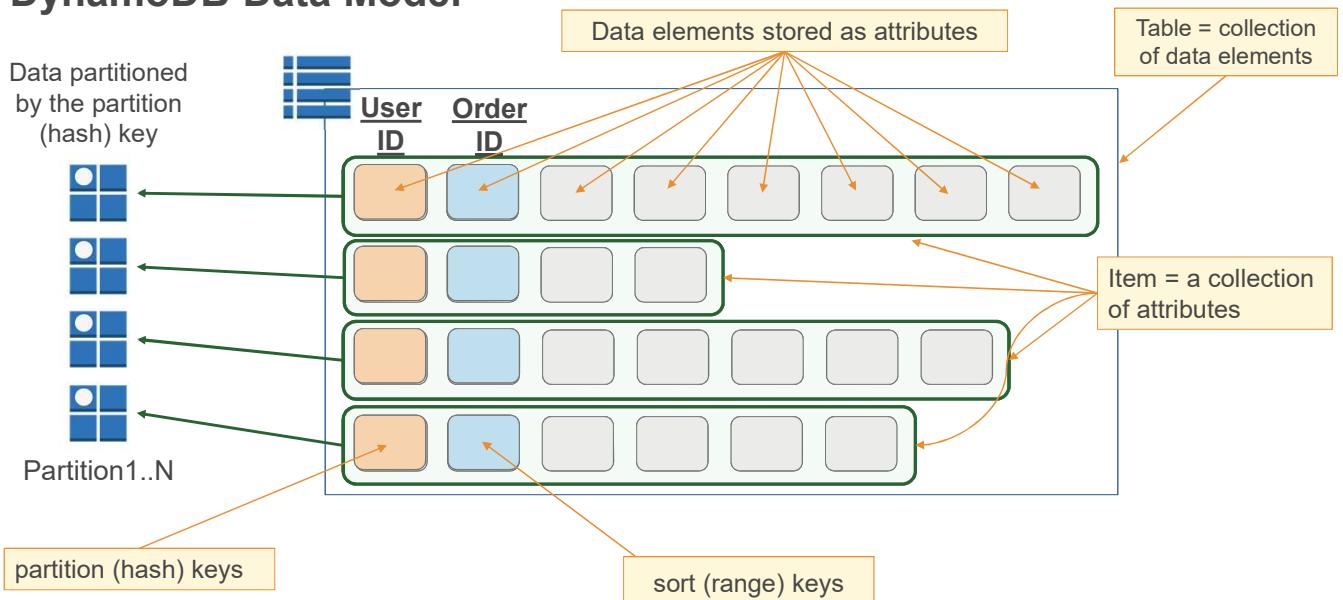
DynamoDB also integrates with Amazon EMR. Amazon EMR allows businesses to perform complex analytics of their large datasets using a hosted pay-as-you-go Hadoop framework on AWS. With DynamoDB, it is easy for customers to use Amazon EMR to analyze datasets stored in DynamoDB and archive the results, while keeping the original dataset in DynamoDB intact.

Businesses can also use Amazon EMR to access data in multiple stores (DynamoDB, Amazon RDS, and Amazon S3), do complex analysis over this combined dataset, and store the results of this work in Amazon S3. Amazon EMR also supports direct interaction with DynamoDB using Hive.

You can run other NoSQL data stores on AWS for use with EMR; for example, Cassandra,

CouchDB, Riak, MongoDB, and HBase.

DynamoDB Data Model



© 2017 Amazon Web Services, Inc. or its affiliates. All rights reserved.



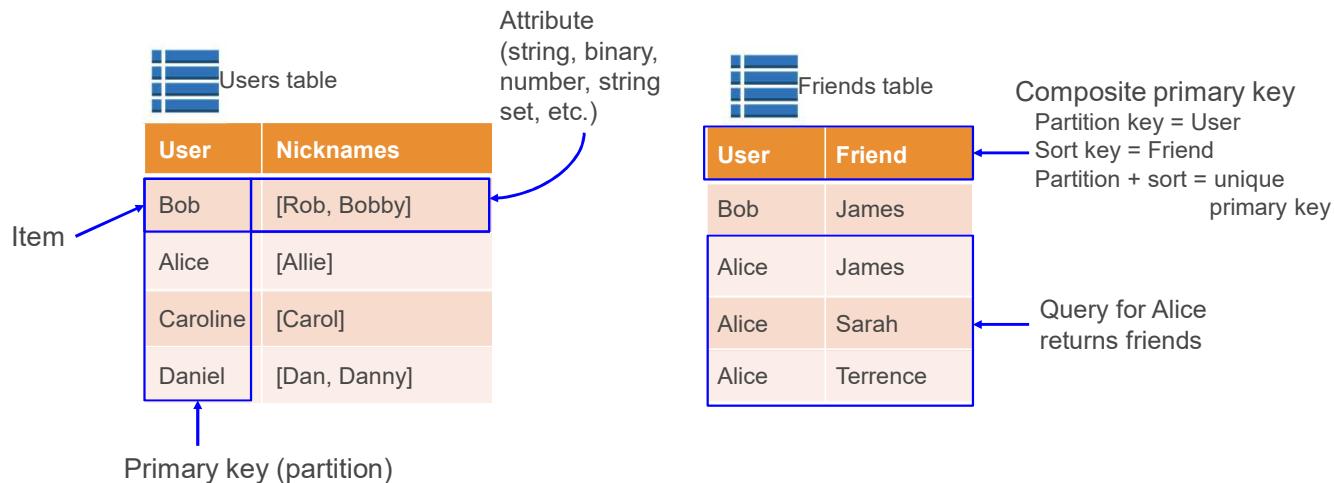
In the world of DynamoDB, there is no schema. In a table, you have items. An item has *variable attributes* that have key-value pairs (similar to associative array or hash). In DynamoDB, they are called *key* and *attribute*.

DynamoDB supports key-value GET/PUT operations using a user-defined primary key (partition key, also known as a hash key). The primary key is the only required attribute for items in a table and it uniquely identifies each item. You specify the primary key when you create a table. In addition, DynamoDB provides flexible querying by allowing queries on non-primary key attributes using Global Secondary Indexes and Local Secondary Indexes.

A primary key can be a single-attribute partition key or a composite partition-sort key. A composite partition-sort key is indexed as a partition key element and sort (also known as range) key element. This multi-part key could be a combination of “UserID” (partition) and “Timestamp” (sort). Holding the partition key element constant, you can search across the sort key element to retrieve items. For example, you can retrieve all items for a single UserID across a range of timestamps.

Auto-partitioning occurs with data set size growth and as provisioned capacity increases.

Use Case: Social Network



Take a social network as an example. Users have friends, so you create “Users” and “Friends” tables. In the Users table, you use ‘User’ as a partition key to query the data. For each item of a user, you have attributes, which are nicknames in this example. As you can see, the attributes do not have to be all the same length. Attributes can be string, number, binary, string-set, number-set, etc.

In the Friends table, you use a composite primary key. The partition key is “User” and the sort key is “Friend.” Combining the user partition key and the friend sort key together gives you a unique primary key for identifying that friendship.

Item size cannot exceed 400 KB, which includes both attribute name binary length (UTF-8 length) and attribute value lengths (again, binary length). The attribute name counts towards the size limit. For example, consider an item with two attributes: one attribute named "shirt-color" with value "R" and another attribute named "shirt-size" with value "M". The total size of that item is 23 bytes.

DynamoDB Consistency

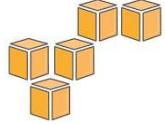
- Each DynamoDB table is automatically replicated across multiple Availability Zones in an AWS region.
- Read consistency: manner and timing of the read operation of a successfully written or updated data item.
- You can specify, at the time of the read request, whether a read should be **eventually** or **strongly** consistent.

DynamoDB Best Practices

- Keep item size small
- Store metadata in DynamoDB and large BLOBS in Amazon S3
- Use table per day, week, month, etc., for storing time series data
- Use conditional or Optimistic Concurrency Control (OCC) updates
- Avoid hot keys and hot partitions

Optimistic concurrency control (OCC) is a concurrency control method applied to transactional systems such as relational database management systems and software transactional memory. OCC assumes that multiple transactions can frequently be completed without interfering with each other. While they are running, transactions use data resources without acquiring locks on those resources. Before being committed, each transaction verifies that no other transaction has modified the data it has read. If the check reveals conflicting modifications, the committing transaction rolls back and can be restarted.

OCC is generally used in environments with low data contention. When conflicts are rare, transactions can be completed without the expense of managing locks and without having transactions wait for other transactions' locks to clear, which leads to higher throughput than other concurrency control methods. However, if contention for data resources is frequent, the cost of repeatedly restarting transactions hurts performance significantly; it is commonly thought that other concurrency control methods have better performance under these conditions. However, locking-based ("pessimistic") methods also can deliver poor performance, because locking can drastically limit effective concurrency even when deadlocks are avoided.



Part 5

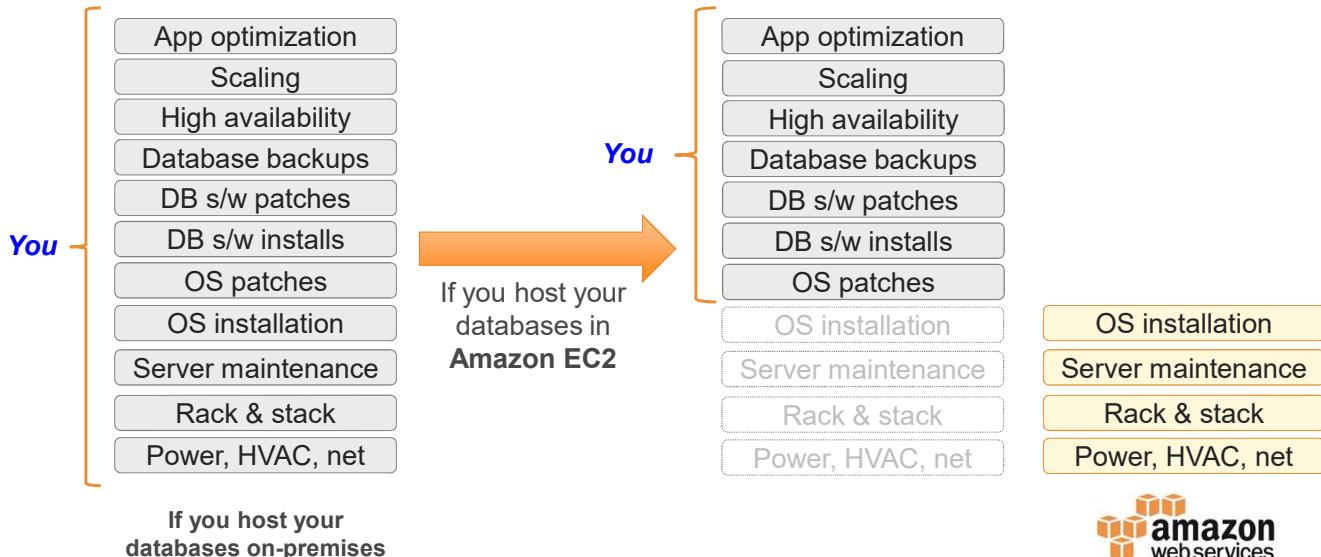
Store Relational Data in Amazon RDS

© 2017 Amazon Web Services, Inc. or its affiliates. All rights reserved.

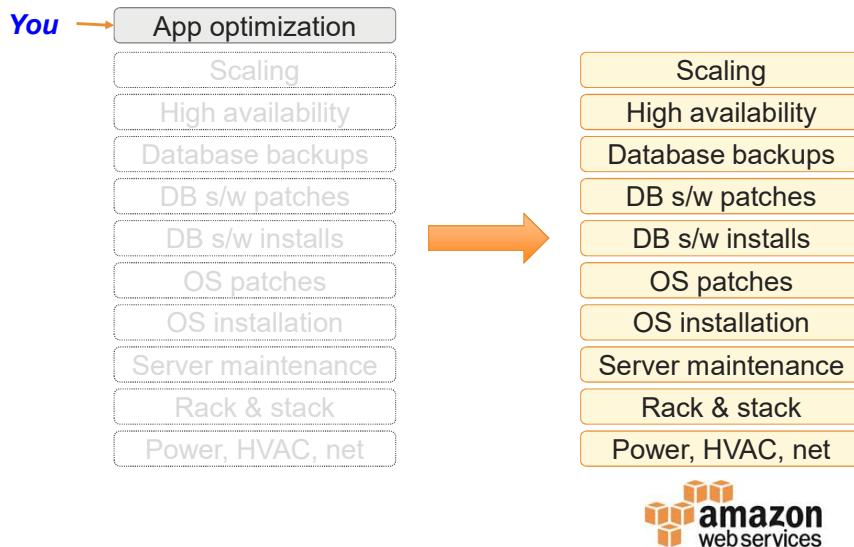


Part 5: Store Relational Data in Amazon RDS

Your Responsibilities



If You Choose A Managed DB Service Like Amazon RDS



© 2017 Amazon Web Services, Inc. or its affiliates. All rights reserved.

amazon
webservices | Academy

All the time that's freed up by offloading undifferentiated labor to AWS can be used to do the app optimizations you always wanted to do.

Amazon RDS Optimizes Developer Productivity

Let developers focus on innovation:

- Query construction
- Query optimization



Offload the operational burdens:

- Migration
- Backup and recovery
- Patching
- Software upgrades
- Storage upgrades
- Frequent server upgrades
- Hardware crash

Amazon RDS Security

- Control Amazon RDS DB instance access via security groups.
- Encrypted instances are currently available for all database engines supported in Amazon RDS.
- Use IAM to control which RDS operations each individual user has permission to call.
- Encrypt connections between your application and your DB instance using SSL/TLS.
- Transparent Data Encryption (TDE) is supported for SQL Server and Oracle.
- You can receive notifications of important events that can occur on your RDS instance.

Access Control

When you first create a DB instance within Amazon RDS, you create a master user account, which is used only within the context of Amazon RDS to control access to your DB instances. The master user account is a native database user account that allows you to log on to your DB instance with all database privileges. You can specify the master user name and password you want associated with each DB instance when you create the DB instance. After you create your DB instance, you can connect to the database using the master user credentials. Subsequently, you can create additional user accounts so that you can restrict who can access your DB instance.

You can control Amazon RDS DB instance access via DB security groups, which are similar to Amazon EC2 security groups but not interchangeable. DB security groups act like a firewall that controls network access to your DB instance. Database security groups default to a “deny all” access mode, and customers must specifically authorize network ingress. There are two ways to do this: authorize a network IP range, or authorize an existing Amazon EC2 security group. DB security groups only allow access to the database server port (all others are blocked) and can be updated without restarting the Amazon RDS DB instance, which gives a customer seamless control of their database access. Using AWS IAM, you can further control access to your RDS DB instances. AWS IAM enables you to control which RDS operations each individual AWS IAM user has permission to call.

Encryption

Amazon RDS DB instances and snapshots at rest can be encrypted. When encryption has been enabled, the instance’s automated backups, Read Replicas, and snapshots are

encrypted using AES-256. Amazon RDS then handles authentication of access and decryption of this data with a minimal impact on performance, and with no need to modify your database client applications. This encryption is available for Amazon RDS with MySQL, PostgreSQL, Oracle, and SQL Server DB instances; however, it is not available for DB instances in the AWS GovCloud (us-gov-west-1) Region.

You can use SSL to encrypt connections between your application and your DB instance. For MySQL and SQL Server, RDS creates an SSL certificate and installs the certificate on the DB instance when the instance is provisioned. For MySQL, launch the MySQL client using the `--ssl_ca` parameter to reference the public key in order to encrypt connections. For SQL Server, download the public key and import the certificate into your Windows operating system. Oracle RDS uses Oracle native network encryption with a DB instance. You simply add the native network encryption option to an option group and associate that option group with the DB instance. When an encrypted connection is established, data transferred between the DB instance and your application will be encrypted during transfer. You can also require your DB instance to only accept encrypted connections. Note that SSL support within Amazon RDS is for encrypting the connection between your application and your DB instance; it should not be relied on for authenticating the DB instance itself. Although SSL offers security benefits, be aware that SSL encryption is a compute-intensive operation and will increase the latency of your database connection.

Event Notification

You can receive notifications of a variety of important events that can occur on your RDS instance, such as whether the instance was shut down, a backup was started, a failover occurred, the security group was changed, or your storage space is low. Amazon RDS groups events into categories that you can subscribe to so that you can be notified when an event in that category occurs. You can subscribe to an event category for a DB instance, DB snapshot, DB security group, or DB parameter group. RDS events are published via Amazon SNS and sent to you as an email or text message.

Amazon Aurora Overview

- A relational database delivered using service-oriented architectures.
 - Similar to how Amazon DynamoDB is a managed version of an Amazon EC2-hosted NoSQL engine.
- Scalable and highly available, leveraging Amazon S3.
- Drop-in compatible with MySQL 5.6.
 - Existing applications “just work.”
 - Amazon RDS MySQL customers can select options for migration.
 - Amazon EC2 (or on-premises) MySQL users can import their data file.

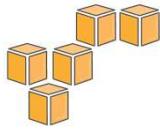
<https://aws.amazon.com/blogs/aws/highly-scalable-mysql-compat-rds-db-engine/>

Amazon Aurora: Scalable Performance

- Integrated with Amazon S3 for continuous backup.
 - Six copies replicated across three Availability Zones.
- Moved logging and storage layer into multi-tenant, scalable service layer.
- SSD data plane up to 64 TB max DB volume
 - Pay only for consumed amount.
 - Added in 10-GB increments.

Amazon Aurora: Resilient Design

- 99.99% available.
- Instant crash recovery.
 - Compare to traditional DBs that replay all logs over single thread.
 - Aurora replays redo records as part of a disk read.
 - Parallel, distributed asynchronous recovery.
- Cache layer survives DB restarts, improving read response.
- Read replicas designed for instant promotion.
 - Up to 15 replicas.
 - ~10-ms replica lag (compared to seconds or minutes with MySQL).



In review...

- 📦 How Web Apps Work
- 📦 Store Static Assets in S3
- 📦 Serve Frequently Accessed Assets from CloudFront
- 📦 Non-relational Data and DynamoDB
- 📦 Relational Data and RDS



Knowledge Assessment

© 2017 Amazon Web Services, Inc. or its affiliates. All rights reserved.



In review...

- How Web Apps Work
- Store Static Assets in S3
- Serve Frequently Accessed Assets from CloudFront
- Non-relational Data and DynamoDB
- Relational Data and RDS

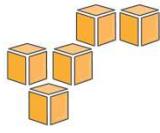
To complete this module, please remember to finish the corresponding knowledge assessment.

How should web-accessible content be stored?

1. Store static assets in Amazon S3.
2. Serve frequently accessed assets from Amazon CloudFront.
3. Store non-relational data in a NoSQL database such as Amazon DynamoDB.
4. Store relational data in Amazon RDS.

How should web-accessible content be stored?

1. Store static assets in Amazon S3.
2. Serve frequently accessed assets from Amazon CloudFront.
3. Store non-relational data in a NoSQL database such as Amazon DynamoDB.
4. Store relational data in Amazon RDS.



Up Next...



LAB 11 - Caching Static Files with Amazon CloudFront



LAB 12 - Implementing a Serverless Architecture with AWS Managed Services



Discussion 2 - Build an Environment for a New Web Application

Before you continue, please complete **Lab 11, Caching Static Files with Amazon CloudFront**; and **Lab 12, Implementing a Serverless Architecture with AWS Managed Services**.

Following the labs is classroom Discussion 2, *Build an Environment for a New Web Application*.



CCA Lab-12

Implementing a Serverless Architecture with AWS Managed Services

(Approx. 45 minutes)

© 2017 Amazon Web Services, Inc. or its affiliates. All rights reserved.

 Academy

Lab 12: Scenario

In this lab, you will put together a **serverless architecture** that processes a text file, stores the information intelligently into DynamoDB tables, and sends notifications based on a condition within one of the tables.

Here are the services that will be used:



Amazon
SNS



Amazon
S3



AWS
Lambda



AWS IAM



Amazon
DynamoDB

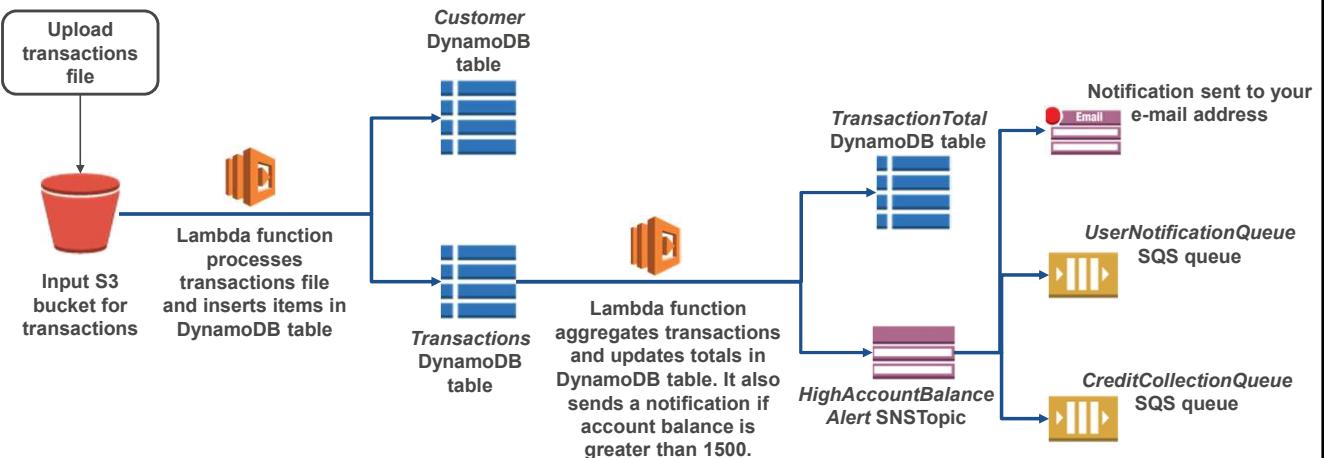


Amazon
SQS

Lab 12: Tasks

1. Inspect the resources created automatically for your lab.
2. Create an SNS topic to receive notifications from one of the AWS Lambda functions and subscribe to it with your e-mail.
3. Create two SQS queues and subscribe them to the SNS topic.
4. Create an AWS Lambda function that will detect an upload to an Amazon S3 bucket, process the file, store the customer identification data it finds into one DynamoDB table, and store the purchase amounts into another DynamoDB table.
5. Create a second AWS Lambda function that detects additions to the second table, adds the values together, and sends a notification to the SNS topic if the total purchases amount to more than \$1500.
6. Test the environment by uploading a sample text file to your bucket.

Lab 12: Final Product



While using another load balancer for the private instances would make this more highly available, in the interest of time we have omitted that step.

© 2017 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.

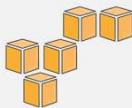
Errors or corrections? Email us at aws-course-feedback@amazon.com.

For all other questions, contact us at
<https://aws.amazon.com/contact-us/aws-training/>.

All trademarks are the property of their owners.

© 2017 Amazon Web Services, Inc. or its affiliates. All rights reserved.





CCA Unit 3 – Architecting on AWS

CCA 3.08: Introducing the Well-Architected Framework

Section 1: Introduction to System Design

Section 2: Automation and Serverless Architectures

Section 3: Well-Architected Best Practices

► **CCA 3.08** Introducing the Well-Architected Framework

CCA 3.09 Well-Architected Pillar 1: Security

CCA 3.10 Well-Architected Pillar 2: Reliability

CCA 3.11 Well-Architected Pillar 3: Performance Efficiency

CCA 3.12 Well-Architected Pillar 4: Cost Optimization

Section 4: Deployment and Implementation

© 2017 Amazon Web Services, Inc. or its affiliates. All rights reserved.



This module begins **Section 3** of Unit 3 – **Well-Architected Best Practices**. We'll start by introducing the Well-Architected framework in this module and then cover each of the four pillars of the framework in the following modules.

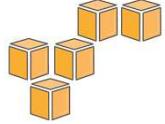
What's In This Module

- 💡 What is the AWS Well-Architected Framework?
- 💡 Well-Architected Pillars Overview

This begins **CCA 3.09** introducing the Well-Architected framework.

This module covers...

- What is the AWS Well-Architected Framework?
- Well-Architected Pillars Overview



Part 1

Introducing the AWS Well-Architected Framework

© 2017 Amazon Web Services, Inc. or its affiliates. All rights reserved.



Part 1: Introducing the AWS Well-Architected Framework

- .
- .
- .

<speaker notes>

What is the AWS Well-Architected Framework?

- The goal of this framework is to enable customers to:
 - Assess and improve their architectures
 - Better understand the business impact of their design decisions
- It provides a **set of questions developed by AWS experts** to helps customers think critically about their architecture.
- It asks, "Does your infrastructure follow best practices?"

AWS Well-Architected Outcomes

Architects should leverage the AWS Well-Architected Framework in order to:

- Increase awareness of **architectural best practices**.
- Address **foundational areas** that are often neglected.
- **Evaluate** architectures using a consistent set of principles.

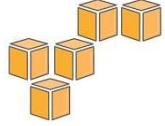
AWS Well-Architected Guidance

■ The AWS Well-Architected Framework **does not** provide:

- Implementation details
- Architectural patterns
- Relevant case studies

■ However, it **does** provide:

- Questions centered on critically understanding architectural decisions
- Services and solutions relevant to each question
- References to relevant resources



Part 2

Well-Architected Pillars Overview

© 2017 Amazon Web Services, Inc. or its affiliates. All rights reserved.

 Academy

Part 2: Well-Architected Pillars Overview

Pillars Of The Well-Architected Framework

Security

Protect and monitor systems.



Reliability

Recover from failure and mitigate disruption.



Performance Efficiency

Use resources sparingly.



Cost Optimization

Eliminate unneeded expense.



The pillars of the Well-Architected Framework guide you through looking at the design of your application from four different perspectives, or *pillars*.

Security



💡 The ability to protect:

- Information
- Systems
- Assets

💡 While delivering business value through:

- Risk assessments
- Mitigation strategies

Reliability

Reliability

Recover from failure and mitigate disruption.



The ability of a system to:

- 💡 Recover from infrastructure or service failures
- 💡 Dynamically acquire computing resources to meet demand
- 💡 Mitigate disruptions such as:
 - Misconfigurations
 - Transient network issues

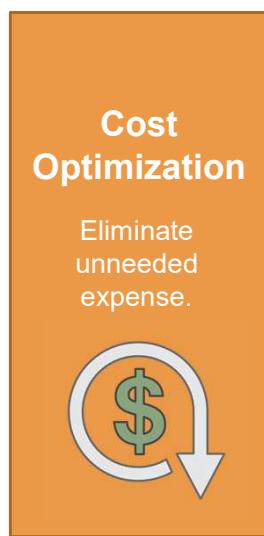
Performance Efficiency



The ability to:

- ◆ Use computing resources efficiently to meet system requirements
- ◆ Maintain that efficiency as demand changes and technologies evolve

Cost Optimization



The ability to avoid or eliminate:

- ─ Unneeded cost
- ─ Suboptimal resources

Pillars Of The Well-Architected Framework

Security

Protect and monitor systems.



Reliability

Recover from failure and mitigate disruption.



Performance Efficiency

Use resources sparingly.

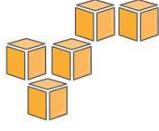


Cost Optimization

Eliminate unneeded expense.



(INSTRUCTOR NOTE: This is just to let the learner see all of the pillars together again before moving on.)



In review...

- 💡 What is the AWS Well-Architected Framework?
- 💡 Well-Architected Pillars Overview



Knowledge Assessment

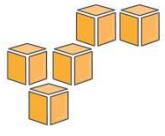
© 2017 Amazon Web Services, Inc. or its affiliates. All rights reserved.



In review...

- What is the AWS Well-Architected Framework?
- Well-Architected Pillars Overview

To complete this module, please remember to finish the corresponding knowledge assessment.



Up Next...



CCA 3.09 - Well-Architected Pillar 1: Security

© 2017 Amazon Web Services, Inc. or its affiliates. All rights reserved.

 **Amazon**
webservices | Academy

Up next is **CCA 3.09** covering the **Security Pillar** of the Well-Architected framework.

© 2017 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.

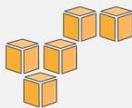
Errors or corrections? Email us at aws-course-feedback@amazon.com.

For all other questions, contact us at
<https://aws.amazon.com/contact-us/aws-training/>.

All trademarks are the property of their owners.

© 2017 Amazon Web Services, Inc. or its affiliates. All rights reserved.





CCA Unit 3 – Architecting on AWS

CCA 3.09: Security Pillar

Section 1: Introduction to System Design

Section 2: Automation and Serverless Architectures

Section 3: Well-Architected Best Practices

CCA 3.08 Introducing the Well-Architected Framework

► **CCA 3.09** Well-Architected Pillar 1: Security

CCA 3.10 Well-Architected Pillar 2: Reliability

CCA 3.11 Well-Architected Pillar 3: Performance Efficiency

CCA 3.12 Well-Architected Pillar 4: Cost Optimization

Section 4: Deployment and Implementation

© 2017 Amazon Web Services, Inc. or its affiliates. All rights reserved.



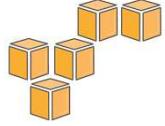
Welcome to **CCA 3.09** – featuring the **Security Pillar** of the Well-Architected framework.

What's In This Module

- Security Pillar Overview
- Principles of the Security Pillar & Key Services
- Preventing Common Security Exploits
- Securing & Encrypting Data
- Securing Data At Rest on S3
- Authentication

This module covers...

- Security Pillar Overview
- Principles of the Security Pillar & Key Services
- Preventing Common Security Exploits
- Securing & Encrypting Data
- Securing Data At Rest on S3
- Authentication



Part 1

Security Pillar Overview

© 2017 Amazon Web Services, Inc. or its affiliates. All rights reserved.



Part 1: Security Pillar Overview

Best Practice: Secure Your Infrastructure Everywhere

Build security into every layer of your infrastructure.

Physical data centers typically rely on security at the perimeter. AWS enables you to implement security at the perimeter as well as **within and between your resources**.

Things to consider:

- 💡 Isolate parts of your infrastructure
- 💡 Encrypt data in transit and at rest
- 💡 Enforce access control granularly, using the principle of least privilege
- 💡 Use multi-factor authentication
- 💡 Leverage managed services
- 💡 Log access of resources
- 💡 Automate your deployments to keep security consistent

Security isn't just about getting through the outer boundary of your infrastructure; it's also about ensuring that your individual environments and their components are secured from each other.

For example, in Amazon EC2, you can set up security groups that allow you to determine which ports on your instances can send and receive traffic and where that traffic can come from or go to. You can use this to reduce the probability that a security threat on one instance will spread to every other instance in your environment. Similar precautions should be taken with other services. Specific ways to implement this best practice are discussed throughout the course.

Securely Store and Analyze Terabytes of Data

Mount Sinai uses AWS to securely store and analyze 100 TB of data...

“ By using AWS, we met strict HIPAA standards for confidentiality... and we can store source files securely and cost-effectively.

Dr. John A. Martignetti
Icahn School of Medicine at Mount Sinai



Icahn School of Medicine
at Mount Sinai

- Needed to search and analyze terabytes of genomics data to investigate the causes of aggressive forms of cancer: breast and ovarian.
- Using AWS gives researchers the ability to mine data from cancer patients all over the world.
- Analyzes more than 100 TB of data each time new information comes in.
- Can store source files securely and cost-effectively with durability and accessibility.

Internationally recognized leader in medical and scientific training, biomedical research, and patient care.



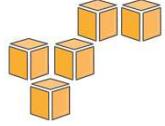
Mount Sinai leveraged Station X's GenePool platform, which is built on the AWS Cloud, to complete this project.



Academy

© 2017 Amazon Web Services, Inc. or its affiliates. All rights reserved.

For more on how Mount Sinai uses AWS, see: <https://aws.amazon.com/solutions/case-studies/mt-sinai/>



Part 2

Principles of the Security Pillar

© 2017 Amazon Web Services, Inc. or its affiliates. All rights reserved.

 **Amazon**
webservices | Academy

Part 2: Principles of the Security Pillar

Security Pillar Principles

💡 Apply Security at All Layers

Instead of just running security appliances (e.g., firewalls) at the edge of your infrastructure, **use firewalls and other security controls on all of your resources:**

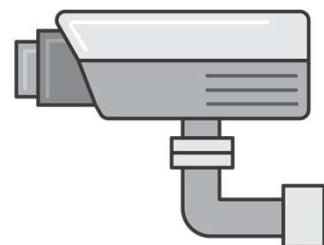
- ✓ Every virtual server
- ✓ Every load balancer
- ✓ Every network subnet

💡 Enable Traceability

💡 Automate Responses To Security Events

💡 Focus On Securing Your System

💡 Automate Security Best Practices



Apply security at all layers: Instead of just running security appliances (e.g., firewalls) at the edge of your infrastructure, use firewalls and other security controls on all of your resources (e.g., every virtual server, load balancer, and network subnet).

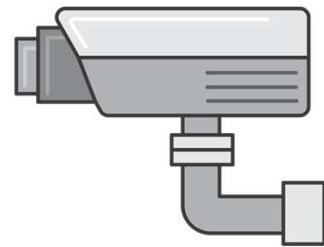
Security Pillar Principles

💡 Apply Security at All Layers

💡 Enable Traceability

Log and audit all action and changes to your environment and access to your services.

- 💡 Automate Responses To Security Events
- 💡 Focus On Securing Your System
- 💡 Automate Security Best Practices

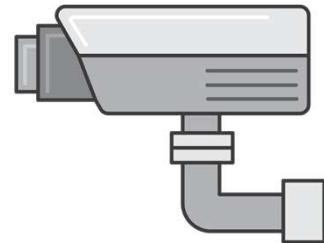


Log and audit all action and changes to your environment and access to your services.

Security Pillar Principles

- 💡 **Apply Security at All Layers**
- 💡 **Enable Traceability**
- 💡 **Automate Responses To Security Events**

Monitor and automatically trigger responses to event-driven or condition-driven alerts.



- 💡 **Focus On Securing Your System**
- 💡 **Automate Security Best Practices**

Monitor and automatically trigger responses to event-driven or condition-driven alerts.

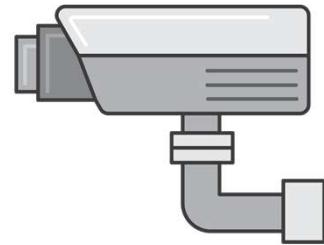
Security Pillar Principles

- 💡 **Apply Security at All Layers**
- 💡 **Enable Traceability**
- 💡 **Automate Responses To Security Events**
- 💡 **Focus On Securing Your System**

With the AWS Shared Responsibility Model:

- ✓ AWS provides secure infrastructure and services.
- ✓ You can focus on securing your application, data, and operating systems.

- 💡 **Automate Security Best Practices**

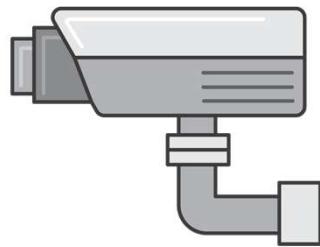


Focus on securing your system: With the AWS shared responsibility model, you can focus on securing your application, data, and operating systems, while AWS provides secure infrastructure and services.

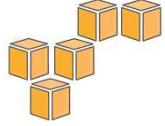
AWS shared responsibility model: <https://aws.amazon.com/compliance/shared-responsibility-model/>

Security Pillar Principles

- 💡 **Apply Security at All Layers**
- 💡 **Enable Traceability**
- 💡 **Automate Responses To Security Events**
- 💡 **Focus On Securing Your System**
- 💡 **Automate Security Best Practices**
 - ✓ Use **software-based security mechanisms** to improve your ability to securely scale more rapidly and cost-effectively.
 - ✓ Create and save a **custom baseline image** of a virtual server, and then use that image automatically on each new server you launch.
 - ✓ Create an **entire infrastructure** that is defined and managed in a **template**.



Automate security best practices: Software-based security mechanisms improve your ability to securely scale more rapidly and cost-effectively. Create and save a custom baseline image of a virtual server, and then use that image automatically on each new server you launch. Create an entire infrastructure that is defined and managed in a template.



Part 3

Key Services for Security

© 2017 Amazon Web Services, Inc. or its affiliates. All rights reserved.

 Academy

The Amazon Web Services logo consists of a stylized orange 'a' icon followed by the text 'amazon webservices'. To the right of the logo, the word 'Academy' is written in a smaller, dark font.

Part 3: Key Services

Key Services for Security

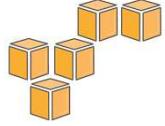
Areas	Key Services									
Data protection		Elastic Load Balancing		Amazon EBS		Amazon S3		Amazon RDS		AWS Key Management Service (KMS)
Privilege management		AWS IAM		MFA	MFA token					
Infrastructure protection		Amazon VPC								
Detective controls		AWS CloudTrail		AWS Config		Amazon CloudWatch				

© 2017 Amazon Web Services, Inc. or its affiliates. All rights reserved.



The AWS service that is essential to security is AWS Identity and Access Management (IAM), which allows you to securely control access to AWS services and resources for your users. The following services and features support the four areas of security:

- **Data protection:** Services such as Elastic Load Balancing, Amazon Elastic Block Store (EBS), Amazon Simple Storage Service (S3), and Amazon Relational Database Service (RDS) include encryption capabilities to protect your data in transit and at rest. AWS Key Management Service (KMS) makes it easier for customers to create and control keys used for encryption.
- **Privilege management:** IAM enables you to securely control access to AWS services and resources. Multi-factor authentication (MFA), adds an extra layer of protection on top of your user name and password.
- **Infrastructure protection:** Amazon Virtual Private Cloud (VPC) lets you provision a private, isolated section of the AWS cloud where you can launch AWS resources in a virtual network.
- **Detective controls:** AWS CloudTrail records AWS API calls, AWS Config provides a detailed inventory of your AWS resources and configuration, and Amazon CloudWatch is a monitoring service for AWS resources.



Part 4

Preventing Common Security Exploits

© 2017 Amazon Web Services, Inc. or its affiliates. All rights reserved.

 **Amazon**
webservices | Academy

Part 4: Preventing Common Security Exploits

DDoS (Distributed Denial of Service) Attacks

- 💡 A Denial of Service (DoS) attack attempts to make your website or application **unavailable** to your end users.
- 💡 To achieve this, attackers use a variety of techniques that **consume network or other resources**, thus interrupting access for legitimate end users.
- 💡 The attackers use **multiple hosts** to orchestrate an attack against a target.

This section gives an overview of Distributed Denial of Service (DDoS) attacks and discusses techniques using AWS services as well as security solutions from the AWS Marketplace to help build resilience into your architecture. We will discuss anti-DDoS features available in AWS and how these features can be used jointly to help protect your services and applications.

DDoS has the potential to impact that availability of services and applications. If you put no defense in place, even a small DDoS can have a significant impact on that availability. We are going to look at techniques on AWS that you can use to deploy a DDoS resilient service.

Distributed Denial of Service (DDoS) Protection

Protecting against attacks is a **shared responsibility** between AWS and you.

AWS	Customer
<ul style="list-style-type: none">▪ AWS API endpoints are hosted on large, Internet-scale, world-class infrastructure.▪ Proprietary DDoS mitigation techniques are used.▪ AWS networks are multi-homed across a number of providers to achieve Internet access diversity.	<ul style="list-style-type: none">▪ Front your application with AWS Services▪ Safeguard exposed resources▪ Minimize the attack surface▪ Evaluate soft limits and request increases ahead of time▪ Learn normal behavior▪ Create a plan for attacks

DDoS is a type of DoS attack where multiple compromised systems (typically infected with Trojans) are used to target a single system. Victims of a DDoS attack consist of both the end targeted system and all systems maliciously used and controlled by the hacker in the distributed attack.

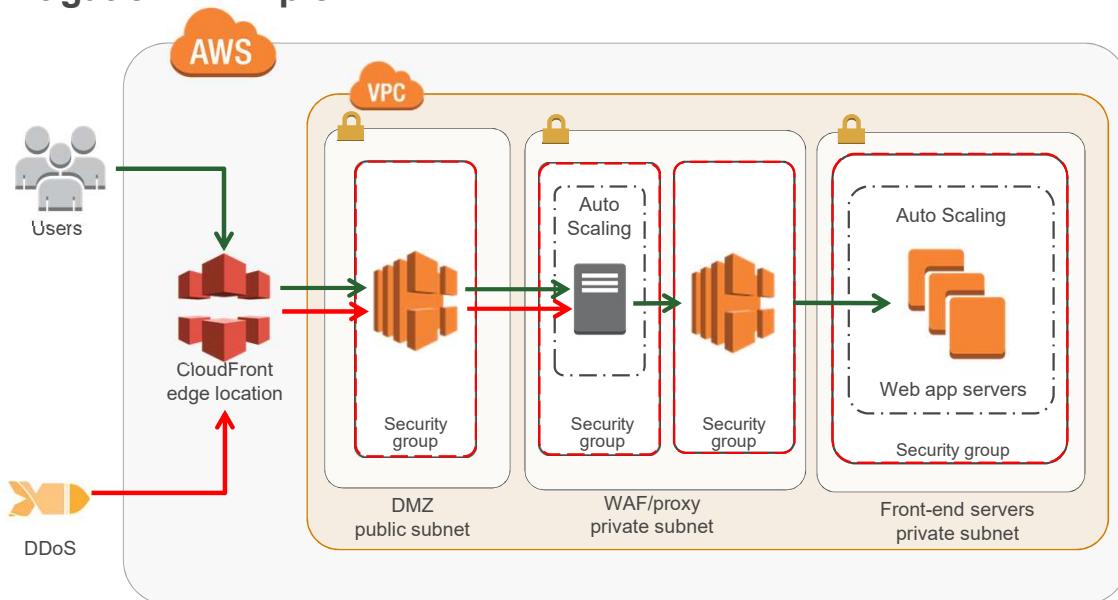
Some services, such as Amazon S3, use a shared infrastructure, which means that multiple AWS accounts access and store data on the same set of Amazon S3 infrastructure components. In this case, a DoS/DDoS attack on abstracted services will probably affect multiple customers. AWS provides both mitigation and protection controls for DoS/DDoS on abstracted services to minimize the impact to you in the event of such an attack. You are not required to provide additional DoS/DDoS protection of such services, but we do advise that you follow the best practices outlined in the AWS Security Best Practices whitepaper

(http://media.amazonwebservices.com/AWS_Security_Best_Practices.pdf).

Video: DDoS Resilience with Amazon Web Services:

<http://www.slideshare.net/AmazonWebServices/ddos-resiliency-with-amazon-web-services-sec305-aws-reinvent-2013>

DDoS Mitigation Example



© 2017 Amazon Web Services, Inc. or its affiliates. All rights reserved.



This is an example of a resilient architecture that can help to prevent or mitigate DDoS attacks.

The strategy to minimize the attack surface area is to (a) reduce the number of necessary Internet entry points, (b) eliminate non-critical Internet entry points, (c) separate end user traffic from management traffic, (d) obfuscate necessary Internet entry points to the level that untrusted end users cannot access them, and (e) decouple Internet entry points to minimize the effects of attacks. This strategy can be accomplished with Amazon Virtual Private Cloud.

Within AWS, you can take advantage of two forms of scaling: horizontal and vertical scaling. In terms of DDoS, there are three ways to take advantage of scaling in AWS: (1) select the appropriate instance types for your application, (2) configure services such as Elastic Load Balancing and Auto Scaling to automatically scale, and (3) use the inherent scale built into the AWS global services like Amazon CloudFront and Amazon Route 53.

Because ELB only supports valid TCP requests, DDoS attacks such as UDP and SYN floods are not able to reach your instances.

You can set a condition to incrementally add new instances to the Auto Scaling group when network traffic is high (typical of DDoS attacks).

Amazon CloudFront also has filtering capabilities to ensure that only valid TCP connections and HTTP requests are made while dropping invalid requests.

A WAF (web application firewall) is a tool that applies a set of rules to HTTP traffic, in order

to filter web requests based on data such as IP addresses, HTTP headers, HTTP body, or URI strings. They can be useful for mitigating DDoS attacks by offloading illegitimate traffic.

AWS now offers a managed WAF service. For more on AWS WAF, see:

<http://docs.aws.amazon.com/waf/latest/developerguide/what-is-aws-waf.html>

Whitepaper: AWS Best Practices for DDoS Resiliency:

https://d0.awsstatic.com/whitepapers/DDoS_White_Paper_June2015.pdf

Using Amazon Inspector To Prevent Common Exploits

- Is an **automated security assessment service** that assesses applications for:
 - Vulnerabilities
 - Deviations from best practices
- Produces a **detailed report with prioritized steps for remediation** after performing the assessment.



© 2017 Amazon Web Services, Inc. or its affiliates. All rights reserved.

Amazon | Academy

Amazon Inspector is an automated security assessment service that helps improve the security and compliance of applications deployed on AWS. Amazon Inspector automatically assesses applications for vulnerabilities or deviations from best practices. After performing an assessment, Amazon Inspector produces a detailed report with prioritized steps for remediation.

Application testing is key to moving fast but staying safe at speed can be challenging. Security assessments are highly manual, resulting in delays or missed security checks. Valuable security subject matter experts are spending too much of their time on route security assessments. Amazon Inspector can help in these areas.

For more on Amazon Inspector, see <https://aws.amazon.com/inspector/>

Video: AWS re:Invent 2015 | (SEC324) New Launch! Introducing Amazon Inspector: Security Insights into Your Application Deployments:

https://www.youtube.com/watch?v=HjuEtMrWc_w

Amazon Inspector Documentation:

http://docs.aws.amazon.com/inspector/latest/userguide/inspector_introduction.html

Amazon Inspector API:

<http://docs.aws.amazon.com/inspector/latest/APIReference/Welcome.html>

Amazon Inspector Rules

Amazon Inspector includes a knowledgebase with **hundreds of rules** that are

- 💡 Mapped to:
 - Common security compliance standards
 - Vulnerability definitions
- 💡 Regularly updated by AWS security researchers.

Examples of Built-In Rules



Remote root login
being enabled



Vulnerable software
versions installed

To help you get started quickly, Amazon Inspector includes a knowledge base of hundreds of rules mapped to common security compliance standards (e.g. PCI DSS) and vulnerability definitions. Examples of built-in rules include checking for remote root login being enabled, or vulnerable software versions installed. These rules are regularly updated by AWS security researchers.

Amazon Inspector Prioritized List of Findings

Amazon Inspector - Findings

Inspector findings are potential security issues discovered during Inspector's assessment of the specified application. Learn more.

Add/Edit attributes					
Last updated on September 24, 2015 4:12:42 PM (20m ago) 					
Filter Viewing 1-10 of					
	Severity	Application	Assessment	Rule package	Finding
<input type="checkbox"/>	High ⓘ	Customer Processing	Comprehensive-Assessment	Authentication Best Practices	Instance i-aac4c4f is
<input type="checkbox"/>	High ⓘ	Customer Processing	Comprehensive-Assessment	Common Vulnerabilities and Ex...	Instance i-aac4c4f is
<input type="checkbox"/>	High ⓘ	Customer Processing	Comprehensive-Assessment	Authentication Best Practices	No password complex
<input type="checkbox"/>	Informational ⓘ	Customer Processing	Initial app	PCI DSS 3.0 Readiness	Instance i-aac4c4f wa
<input type="checkbox"/>	Informational ⓘ	Customer Processing	Initial app	PCI DSS 3.0 Readiness	The machine i-aac4c4
<input type="checkbox"/>	Informational ⓘ	Customer Processing	Comprehensive-Assessment	Operating System Security Best...	No potential security is
<input type="checkbox"/>	Informational ⓘ	Customer Processing	Comprehensive-Assessment	PCI DSS 3.0 Readiness	The machine i-aac4c4
<input type="checkbox"/>	Informational ⓘ	Customer Processing	Comprehensive-Assessment	Network Security Best Practices	No potential security is
<input type="checkbox"/>	Informational ⓘ	Customer Processing	Comprehensive-Assessment	PCI DSS 3.0 Readiness	Instance i-aac4c4f wa
<input type="checkbox"/>	Informational ⓘ	Customer Processing	Initial app	PCI DSS 3.0 Readiness	A machine with instan

© 2017 Amazon Web Services, Inc. or its affiliates. All rights reserved.



Findings are potential security issues discovered during the Amazon Inspector's assessment of the selected assessment target. Findings contain both a detailed description of the security issues and recommendations for how to solve them. The details of the finding include the following:

- Name of the assessment target that includes the EC2 instance where this finding was registered
- Name of the assessment template that was used to produce this finding
- Assessment run start time
- Assessment run end time
- Assessment run status
- Name of the rules package that includes the rule that triggered this finding
- Name of the finding
- Severity of the finding
- Description of the finding
- Remediation steps

Amazon Inspector Findings:

http://docs.aws.amazon.com/inspector/latest/userguide/inspector_findings.html

Amazon Inspector Detailed Remediation Recommendations

Finding for application - Customer Processing

Application name Customer Processing

Assessment name Comprehensive-Assessment

Assessment start Today at 3:51 PM (GMT-4)

Assessment end Today at 4:12 PM (GMT-4)

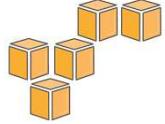
Status COMPLETED

Rule package Authentication Best Practices

Finding Instance i-aac4c46f is configured to allow users to log in with root credentials over SSH

Severity High ⓘ

This is an example of a recommended remediation step that you can complete to fix the potential security issue described by the finding.



Part 5
Securing Data

© 2017 Amazon Web Services, Inc. or its affiliates. All rights reserved.

 **Amazon**
webservices | Academy

Part 5: Securing Data

CloudFront Custom SSL Support

By default, your content is delivered to viewers over HTTPS by using a CloudFront distribution domain name such as <https://dxxxxx.cloudfront.net/image.jpg>.

Custom SSL certificate support features let you use your own domain name and your own SSL certificate.

- Server Name Indication (SNI) Custom SSL
 - Allows multiple domains to serve SSL traffic over the same IP address.
- Dedicated IP Custom SSL
 - To deliver content to browsers that do not support SNI.

SNI Custom SSL (works for most clients)

Server Name Indication (SNI) Custom SSL relies on the SNI extension of the Transport Layer Security protocol, which allows multiple domains to serve SSL traffic over the same IP address. Amazon CloudFront delivers your content from each edge location and offers the same security as the Dedicated IP Custom SSL feature (see below).

When you use SNI Custom SSL, some users may not be able to access your content because some older browsers do not support SNI and will not be able to establish a connection with CloudFront to load the HTTPS version of your content. There is no separate pricing for this feature. You can use SNI Custom SSL with no upfront or monthly fees for certificate management; you simply pay normal Amazon CloudFront rates for data transfer and HTTPS requests.

For more information, see <http://aws.amazon.com/cloudfront/custom-ssl-domains/>

Dedicated IP Custom SSL (works for all clients)

If you need to deliver content to browsers that don't support SNI, you can use the Dedicated IP Custom SSL feature. For this feature, CloudFront allocates dedicated IP addresses to serve your SSL content at each CloudFront edge location.

To sign up for Dedicated IP Custom SSL certificate support, complete this form:

<http://aws.amazon.com/cloudfront/custom-ssl-domains/>

When AWS approves your request, you can upload an SSL certificate and use the AWS Management Console to associate it with your CloudFront distributions. If you need to associate more than one custom SSL certificate with your CloudFront distribution, please include details about your use case and the number of custom SSL certificates you intend to use in the "Use Case and # of SSL Certs You Intend to Use" section of the form.

Use Alternate Domain Names With HTTPS

1. To use **dedicated IP**, request permission for your AWS account (not necessary for SNI).
2. Upload your SSL certificate to the IAM certificate store.

```
aws iam upload-server-certificate --server-certificate-name  
CertificateName --certificate-body file://public_key_certificate_file  
--private-key file://privatekey.pem --certificate-chain  
file://certificate_chain_file --path /cloudfront/path/
```

3. Update your distribution to include your domain names.
 - Specify which SSL certificate you want to use.
 - Specify dedicated IP address or SNI.
 - Add or update DNS records.

To request permission, go to <https://aws.amazon.com/cloudfront/custom-ssl-domains/>

By default, when you request permission to use an alternate domain name with HTTPS, AWS updates your account so you can associate two custom SSL certificates with your CloudFront distributions. Typically, you'll use the second certificate only temporarily, when you have more than one distribution and you need to rotate certificates. If you need to permanently associate two or more certificates with your distributions, indicate how many certificates you need and describe the circumstances in your request.

Using IAM CLI to upload your SSL certificate to the IAM certificate store

You must upload your certificate to the IAM certificate store using the same AWS account that you used to create your CloudFront distribution.

- When you upload your certificate to IAM, the value of the -path parameter (certificate path) must start with /cloudfront/; for example, /cloudfront/production/ or /cloudfront/test/. The path also must end with a /.
- If you plan to use the CloudFront API to create or update your distribution, make note of the alphanumeric string that the AWS CLI returns, for example AS1A2M3P4L5E67SIIXR3J. This is the value that you will specify in the IAMCertificateId element. You don't need the IAM ARN, which is also returned by the CLI.

After you associate your SSL certificate with your CloudFront distribution, do not delete the certificate from the IAM certificate store until you remove the certificate from all distributions and until the status of the distributions has changed to *Deployed*.

CloudFront Security: Advanced SSL Features Support

High-security ciphers

Improve the security of HTTPS connections.

Perfect Forward Secrecy

Provides additional safeguards against eavesdropping of encrypted data through a unique random session key.

OCSP Stapling

Improves the time taken for individual SSL/TLS handshakes by moving the Online Certificate Status Protocol (OSCP) check.

Session Tickets

Helps speed up the time spent restarting or resuming an SSL/TLS session.

High security ciphers improve the security of HTTPS connections. Amazon CloudFront edge servers and clients (e.g., browsers) automatically agree on a cipher as part of the SSL handshake process, and now the connections can use ciphers with advanced features such as Elliptic Curve signatures and key exchanges.

Perfect Forward Secrecy provides additional safeguards against the eavesdropping of encrypted data through the use of a unique random session key. This prevents the decoding of captured data, even if the secret long-term key is compromised.

OCSP Stapling improves the time taken for individual SSL handshakes by moving the Online Certificate Status Protocol (OSCP) check, a call used to obtain the revocation status of an SSL certificate, from the client to a periodic, secure check by the Amazon CloudFront servers. With OCSP Stapling, the client no longer needs to handle certificate validation, which improves performance.

Session Tickets help speed up the time spent restarting or resuming an SSL session. Amazon CloudFront encrypts SSL session information and stores it in a ticket that the client can use to resume a secure connection instead of repeating the SSL handshake process.

How To Make Content Private

Restrict access to objects in your Amazon S3 bucket.

Require that users use signed URLs.

- Create CloudFront key pairs for your trusted signers.
- Write the code that generates signed URLs.
 - Typically, you'll write an application that automatically generates signed URLs.
 - Alternatively, use a web interface to create signed URLs.
- Add trusted signers to your distribution.
- **Note:** Once you add a trusted signer to your distribution, users must use signed URLs to access the corresponding content.

A signed URL includes additional information, such as an expiration date and time that gives you more control over access to your content. This additional information appears in a policy statement which is based on either a **canned** policy or a **custom** policy:

- Use a **canned policy** to restrict access to a single object.
- Use a **custom policy** to restrict access to one or more objects using pattern matching.

For more information about creating a signed URL using a **canned** policy, see
<http://docs.aws.amazon.com/AmazonCloudFront/latest/DeveloperGuide/private-content-creating-signed-url-canned-policy.html>

For more information about creating a signed URL using a **custom** policy, see
<http://docs.aws.amazon.com/AmazonCloudFront/latest/DeveloperGuide/private-content-creating-signed-url-custom-policy.html>

Origin Access Identity To Restrict Access

Restrict access to Amazon S3 content by creating an **origin access identity** (OAI), which is a special CloudFront user.

- CloudFront origin access identity gets the objects from Amazon S3 on your users' behalf.
- Direct access to the objects through Amazon S3 URLs will be denied.

Procedure:

1. Create an origin access identity and add it to your distribution.
2. Change the permissions either on your Amazon S3 bucket or the objects in your bucket so that only the origin access identity has read permission.

Typically, if you use an Amazon S3 bucket as the origin for a CloudFront distribution, you grant everyone permission to read the objects in your bucket. This allows anyone to access your objects using either the CloudFront URL or the Amazon S3 URL. CloudFront doesn't expose Amazon S3 URLs, but your users may have those URLs if your application servers access objects directly from Amazon S3 or if anyone gives out direct links to specific objects in Amazon S3.

If you want to use CloudFront-signed URLs to provide access to objects in your Amazon S3 bucket, you probably also want to prevent users from accessing your Amazon S3 objects by using Amazon S3 URLs. If users access your objects directly in Amazon S3, they bypass the controls provided by CloudFront-signed URLs, including control over when a URL expires and control over which IP addresses can be used to access the objects. In addition, if users access objects using both CloudFront URLs and Amazon S3 URLs, CloudFront access logs are less useful because they are incomplete.

You restrict access to Amazon S3 content by creating an origin access identity, which is a special CloudFront user. You change Amazon S3 permissions to give the origin access identity permission to access your objects, and to remove permissions from everyone else. When your users access your Amazon S3 objects using CloudFront URLs, the CloudFront origin access identity gets the objects on your user's behalf. If your users try to access objects using Amazon S3 URLs, they are denied access. The origin access identity has permission to access objects in your Amazon S3 bucket, but users don't.

You can create *origin access identity* using the CloudFront console or the CloudFront API.

- For more information about using the CloudFront console, see
<http://docs.aws.amazon.com/AmazonCloudFront/latest/DeveloperGuide/private-content-restricting-access-to-s3.html#private-content-creating-oai-console>
- For more information about using the CloudFront API, see
<http://docs.aws.amazon.com/AmazonCloudFront/latest/DeveloperGuide/private-content-restricting-access-to-s3.html#private-content-creating-oai-api>

Use IAM To Control API Access To CloudFront Resources

Control a user's API access to CloudFront with IAM policies.

Group policy example:

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": ["cloudfront:*"],
            "Resource": "*",
            "Condition": {
                "Bool": {
                    "aws:SecureTransport": "true"
                }
            }
        }
    ]
}
```

Grant permission to access all CloudFront actions for the group this policy is attached to...

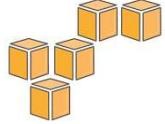
...with condition that actions require use of SSL/TLS

This is an example of an IAM policy that is attached to a group. This policy allows a group to retrieve CloudFront distribution data, but only if it is using SSL with the request.

In an IAM policy, you can specify any and all API actions that CloudFront offers. The action name must be prefixed with the lowercase string “cloudfront:” for example, “cloudfront:GetDistributionConfig”.

For the list of canonical names for all CloudFront actions, see

<http://docs.aws.amazon.com/AmazonCloudFront/latest/DeveloperGuide/UsingWithIAM.html>



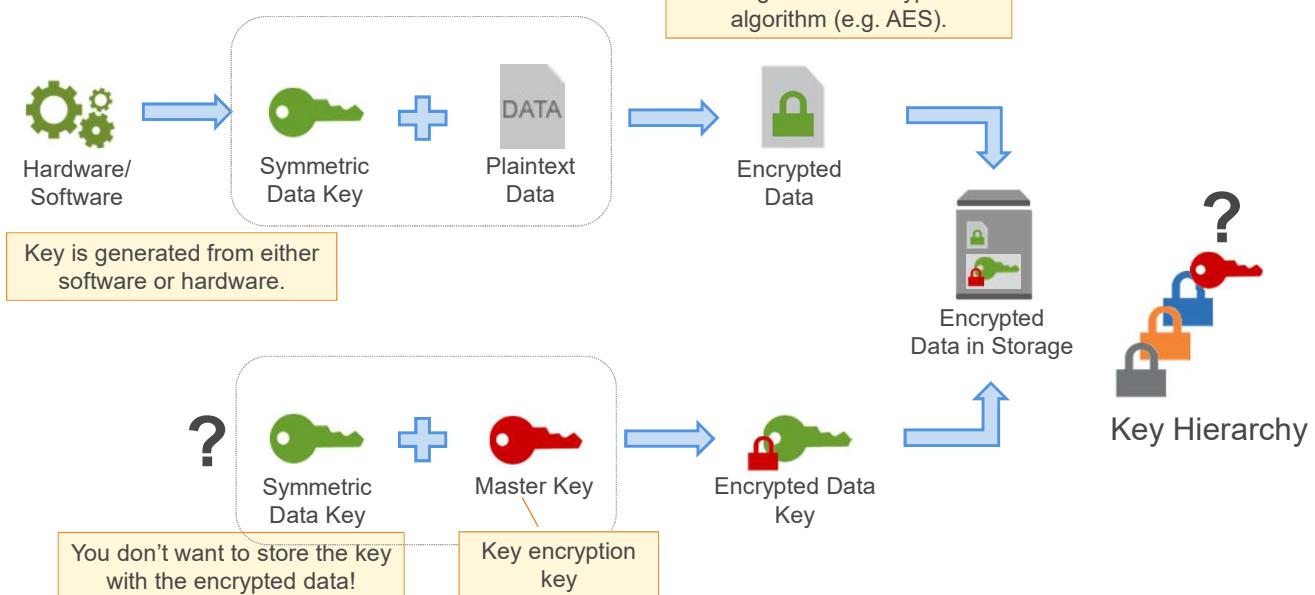
Part 6
Encrypting Data

© 2017 Amazon Web Services, Inc. or its affiliates. All rights reserved.

 **Amazon**
webservices | Academy

Part 6: Encrypting Data

Encryption Primer



© 2017 Amazon Web Services, Inc. or its affiliates. All rights reserved.



Before we discuss specific encryption and key management functions in AWS, let's review how data encryption and key management is typically implemented.

A symmetric data key is generated from either software or hardware. Symmetric keys are preferable to asymmetric keys when you want to encrypt data of an arbitrary size and have it be fast.

The key is used along with an encryption algorithm (like AES), and the resulting ciphertext is stored.

But what about the symmetric key you just used? You can't store it with the encrypted data. You have to protect that key somehow.

The best practice is to encrypt the data key with yet another key, called a key-encrypting key. This key can be symmetric or asymmetric, but it should be derived and stored in a separate system than the one you're processing your data in.

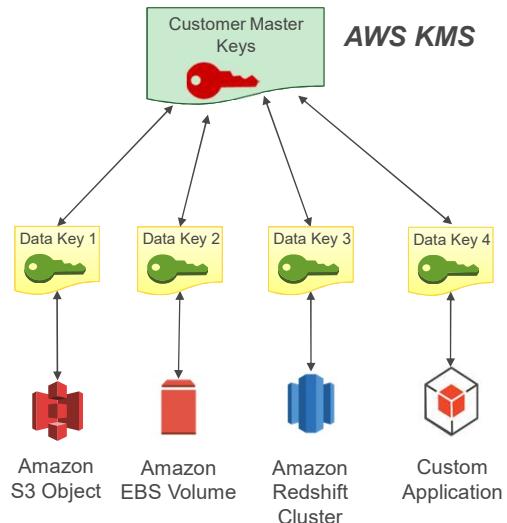
After you encrypt the data key with the key-encrypting key, you can then store the resulting ciphertext along with the encrypted data.

But what about the key-encrypting key? How do you protect that? You can iterate on the process of enveloping this key with additional keys as many times as you want, creating a key hierarchy. At some point, you'll need to access a plaintext key that starts the "unwrapping" process so you can derive the final data key to decrypt the data. The location and access controls around this key should be distinct from the ones used with the original data.

What Is AWS Key Management Service (KMS)?

AWS KMS is a managed encryption service that enables you to easily encrypt your data.

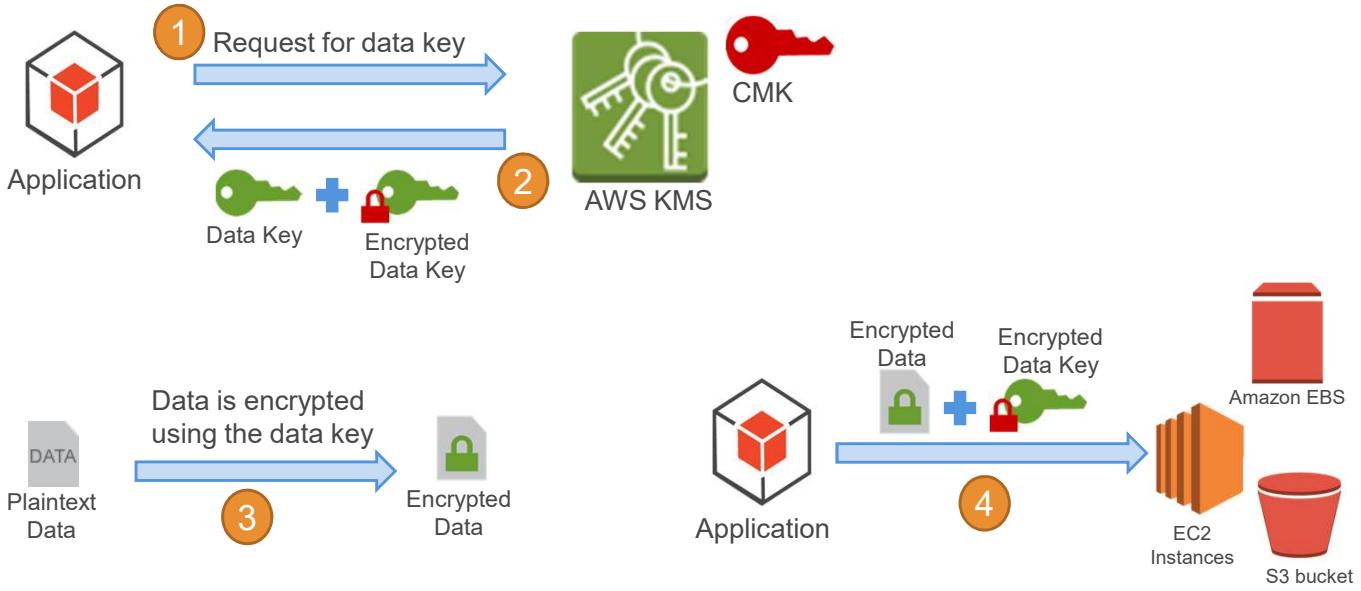
- Two-tiered key hierarchy using envelope encryption.
- Data keys are unique.
- AWS KMS master keys encrypt data keys.
- AWS KMS master keys never leave the AWS KMS system.



AWS KMS is a managed encryption service that enables you to easily encrypt your data. AWS KMS provides a highly available key storage, management, and auditing solution for you to encrypt your data across AWS services and within your own applications. If you are a developer who needs to encrypt data in your applications, use the AWS SDKs with AWS KMS support to easily use and protect encryption keys. If you're an IT administrator looking for a scalable key management infrastructure to support your developers and their growing number of applications, use AWS KMS to reduce your licensing costs and operational burden. If you're responsible for providing data security for regulatory or compliance purposes, use AWS KMS to verify that data is encrypted consistently across the applications where it is used and stored.

AWS KMS uses envelope encryption to protect data. AWS KMS creates a data key, encrypts it under a customer master key, and returns plaintext and encrypted versions of the data key to you. You use the plaintext key to encrypt data and store the encrypted key alongside the encrypted data. The key should be removed from memory as soon as is practical after use. You can retrieve a plaintext data key only if you have the encrypted data key and you have permission to use the corresponding master key.

AWS KMS Customer Master Key (CMK) And Data Keys



© 2017 Amazon Web Services, Inc. or its affiliates. All rights reserved.



AWS KMS uses a type of key called a customer master key (CMK) to encrypt and decrypt data. CMKs are the fundamental resources that AWS KMS manages. CMKs can be either customer-managed keys or AWS-managed keys. They can be used inside of AWS KMS to encrypt or decrypt up to 4 kilobytes of data directly. They can also be used to encrypt generated data keys, which are then used to encrypt or decrypt larger amounts of data outside of the service. CMKs can never leave AWS KMS unencrypted, but data keys can. There is one AWS-managed key for each account for each service that is integrated with AWS KMS. This key is referred to as the *default* key for the service under your account. This key is used to encrypt data keys used by AWS services to protect data when you don't specify a CMK while creating the encrypted resource. If you need more granular control, you can specify a customer-managed key. For example, if you choose to encrypt an Amazon EBS volume, you can specify the AWS-managed default EBS key for the account or a CMK you created within AWS KMS. The key you selected is then used to protect the data key used to encrypt the volume.

You can only create CMKs if you have the appropriate permissions. You can provide an alias (display name) and a description for the key and define which IAM users or roles within an account can manage and use the key. You can also choose to allow AWS accounts other than your own to use the key. You use data keys to encrypt large data objects within your own application outside AWS KMS. When you call **GenerateDataKey**, AWS KMS returns a plaintext version of the key and cipher text that contains the key encrypted under the specified CMK. AWS KMS tracks which CMK was used to encrypt the data key. You use the

plaintext data key in your application to encrypt data, and you typically store the encrypted key alongside your encrypted data. Security best practices suggest that you should remove the plaintext key from memory as soon as is practical after use. To decrypt data in your application, pass the encrypted data key to the Decrypt function. AWS KMS uses the associated CMK to decrypt and retrieve your plaintext data key. Use the plaintext key to decrypt your data and then remove the key from memory.

Benefits Of Using AWS KMS

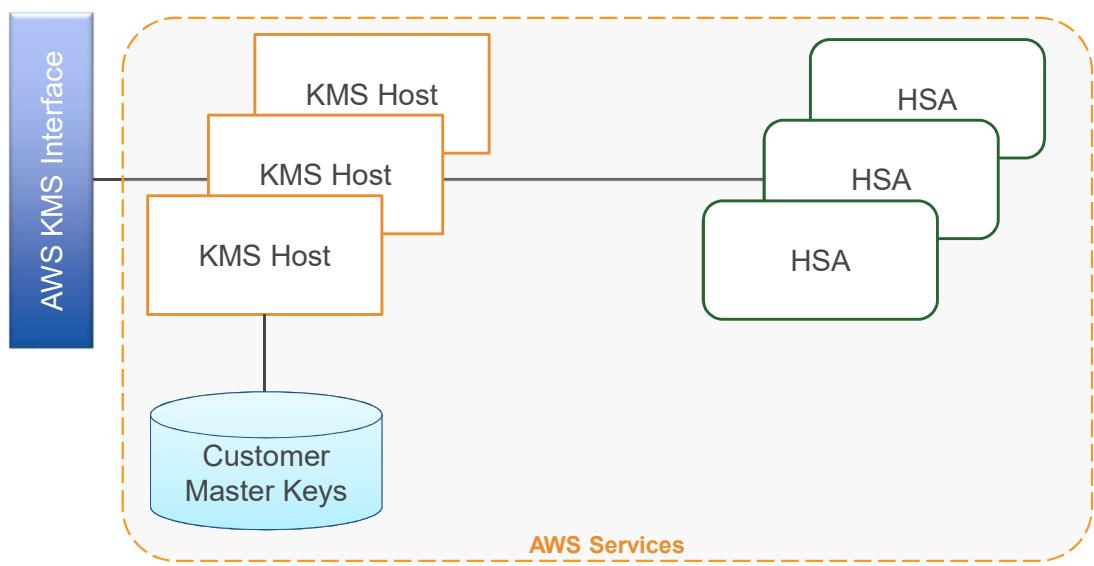
- Only data keys are available directly to the customer, and these are unique to each item encrypted.
If one was compromised, it would not allow decryption of other objects.
- The risk of a compromised data key is limited.
- The performance for encrypting large data is improved.
- It is easier to manage a small number of master keys than millions of data keys.

AWS KMS allows you to centrally manage and securely store your keys. These keys can be used from within your applications and supported AWS cloud services to protect your data, but the key never leaves AWS KMS. You submit data to AWS KMS to be encrypted or decrypted under keys that you control. You set usage policies on these keys that determine which users can use them to encrypt and decrypt data. All requests to use these keys are logged in AWS CloudTrail so you can understand who used which key when.

You can perform the following key management functions in AWS KMS:

- Create keys with a unique alias and description.
- Define which IAM users and roles can manage keys.
- Define which IAM users and roles can use keys to encrypt and decrypt data.
- Choose to have AWS KMS automatically rotate your keys on an annual basis.
- Temporarily disable keys so they cannot be used by anyone.
- Re-enable disabled keys.
- Audit use of keys by inspecting logs in AWS CloudTrail.

How Does AWS KMS work?



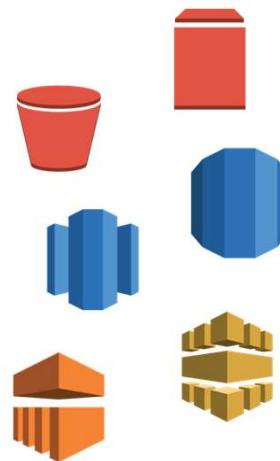
AWS KMS provides a simple web interface in the AWS Management Console and RESTful APIs to access an elastic, multi-tenant, hardened security appliance (HSA). You can establish your own HSA-based cryptographic contexts under your master keys. These keys are only accessible on the HSAs, and they can be used to perform HSA-resident cryptographic operations, including the issuance of application data keys (encrypted under your master key). You can create multiple master keys, each represented with an HSA-based customer master key. You can use the AWS KMS console to manage a set of master keys that protect application-specific keys for services in AWS, such as Amazon Elastic Block Store (EBS) volume encryption and Amazon Simple Storage Service (S3) encryption, and to manage developer access to a cloud-based cryptographic service provider to securely obtain, use, and store keys protected by master keys. Access can be managed per master key, per API.

AWS KMS is a tiered service consisting of web-facing KMS hosts and a tier of HSAs. The grouping of these tiered hosts forms a logical entity called a domain. All customer requests to AWS KMS are made over the Secure Sockets Layer protocol (SSL) and terminate on a KMS host. The AWS KMS hosts use protocols and procedures to fulfill those requests through the HSAs. AWS KMS authorizes customer requests through existing AWS authentication schemes from AWS Identity and Access Management (IAM).

AWS KMS Integration With Other AWS Services

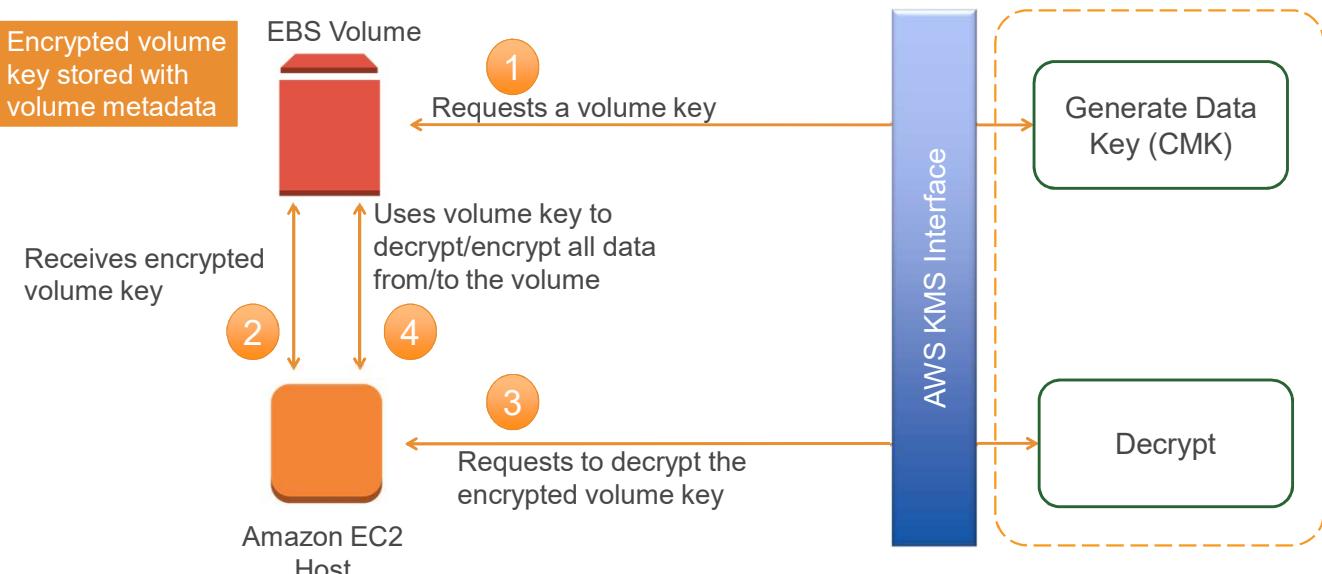
The following services are integrated with AWS KMS to simplify data encryption:

- Amazon Elastic Block Store (EBS)
- Amazon Simple Storage Service (S3)
- Amazon RDS
- Amazon Redshift
- Amazon Elastic Transcoder
- Amazon WorkMail
- Amazon EMR



AWS KMS is integrated with other AWS services including Amazon Elastic Block Store (Amazon EBS), Amazon Simple Storage Service (Amazon S3), Amazon Redshift, Amazon Elastic Transcoder, Amazon WorkMail, and Amazon Relational Database Service (Amazon RDS) to make it simple to encrypt your data with encryption keys that you manage. AWS KMS is also integrated with AWS CloudTrail to provide you with key usage logs to help meet your regulatory and compliance needs.

EBS Volume Encryption Using AWS KMS



© 2017 Amazon Web Services, Inc. or its affiliates. All rights reserved.



Amazon Elastic Block Store (EBS) offers a volume encryption. Each volume is encrypted using AES-256-XTS. This requires two 256-bit volume keys, which you can think of as one 512-bit volume key. The volume key is encrypted under a customer master key in your account. For Amazon EBS to encrypt a volume for you, it must have access to a CMK in the account. You do this by providing a grant for Amazon EBS to the CMK to create data keys and to encrypt and decrypt these data keys. Now, Amazon EBS will use the CMK to generate AWS KMS-encrypted volume keys. The basic steps to encrypt data being written to an EBS volume are:

1. Amazon EBS obtains an encrypted volume key under a customer master key through AWS KMS and stores the encrypted key with the volume metadata.
2. When the EBS volume is mounted, the encrypted volume key is retrieved.
3. A call to AWS KMS over SSL is made to decrypt the encrypted volume key.
4. AWS KMS identifies the CMK, makes an internal request to an HSA in the fleet to decrypt the volume key, and returns the volume key back to the customer over the SSL session.
5. The volume key is stored in memory and used to encrypt and decrypt all data going to and from the attached EBS volume.
6. Amazon EBS retains the encrypted volume key for later use in case the volume key in memory is no longer available.

The XTS-AES mode of the AES algorithm is an industry-recognized standard for maintaining confidentiality of storage devices. For more information, see <http://csrc.nist.gov/publications/nistpubs/800-38E/nist-sp-800-38E.pdf>.

AWS CloudHSM



- Protects your cryptographic keys using a dedicated, tamper-resistant Hardware Security Module (HSM).
- Helps you comply with strict cryptographic key management requirements.
Designed to meet [FIPS 140-2](#) and [Common Criteria EAL4+](#) standards.

© 2017 Amazon Web Services, Inc. or its affiliates. All rights reserved.



The AWS CloudHSM service helps you meet corporate, contractual, and regulatory compliance requirements for data security by using dedicated Hardware Security Module appliances within the AWS cloud. A Hardware Security Module (HSM) is a hardware appliance that provides secure key storage and cryptographic operations within a tamper-resistant hardware device. HSMs are designed to securely store cryptographic key material and use the key material without exposing it outside the cryptographic boundary of the appliance.

AWS and AWS Marketplace partners offer a variety of solutions for protecting sensitive data within the AWS platform, but additional protection may be necessary for some applications and data that are subject to strict contractual or regulatory requirements for managing cryptographic keys. AWS CloudHSM allows you to protect your encryption keys within HSMs that are designed and validated to government standards for secure key management. You can securely generate, store, and manage the cryptographic keys used for data encryption in a way that ensures that only you have access to the keys. AWS CloudHSM helps you comply with strict key management requirements within the AWS cloud without sacrificing application performance.

As part of the service, AWS currently provides Luna SA 7000 HSM appliances from SafeNet, Inc. These appliances are designed to meet Federal Information Processing Standard (FIPS) 140-2 and Common Criteria EAL4+ standards. For more information about Luna SA HSM appliances, see <http://www.safenet-inc.com/data-encryption/hardware-security-modules-hsms/luna-hsms-key-management/luna-sa-network-hsm/>.

For more on FIPS 140-2, see: https://en.wikipedia.org/wiki/FIPS_140-2

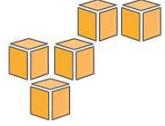
For more on Common Criteria EAL4+, see: https://en.wikipedia.org/wiki/Common_Criteria

AWS CloudHSM vs AWS KMS

AWS CloudHSM	AWS KMS
Single-tenant HSM	Multi-tenant AWS service
Customer-managed durability and availability	Highly available and durable key storage and management
Customer managed root of trust	AWS managed root of trust
Broad third-party app support	Broad support for AWS services
Symmetric and asymmetric options	Symmetric encryption only

AWS CloudHSM provides you with a dedicated hardware device installed in your virtual private cloud that provides a FIPS 140-2 Level 2 validated single-tenant HSM to store and use your keys. You have total control over your keys and the application software that uses them with AWS CloudHSM.

AWS KMS allows you to control the encryption keys used by your applications and supported AWS services in multiple regions around the world from a single console. Centralized management of all your keys in AWS KMS lets you enforce who can use your keys, when they get rotated, and who can manage them. AWS KMS integration with AWS CloudTrail gives you the ability to audit the use of your keys to support your regulatory and compliance activities.



Part 7

Encrypting Source and Output Data at Rest in Amazon S3

© 2017 Amazon Web Services, Inc. or its affiliates. All rights reserved.

 Academy

Part 7: Encrypting Source and Output Data at Rest in Amazon S3

Source And Output Data At Rest In Amazon S3

- 💡 Data stored in Amazon S3 is private by default, requires AWS credentials for access
 - Access to Amazon S3 can be over HTTP or HTTPS
 - Amazon S3 logging allows auditing of access to all objects
 - Amazon S3 supports access control lists and policies for every bucket, prefix (directory/folder), and object
- 💡 Amazon S3 provides server-side encryption (AES-256) using AWS maintained keys or customer provided keys
 - AWS encryption keys are further encrypted with a rotating key
- 💡 Can also encrypt data before storage in Amazon S3 (client-side encryption)

Server-side encryption is about data encryption at rest; that is, Amazon S3 encrypts your data as it writes it to disks in its data centers and decrypts it for you when you access it. If you authenticate your request and have access permissions, there is no difference in the way you access encrypted or unencrypted objects. Amazon S3 manages encryption and decryption for you. For example, if you share your objects using a pre-signed URL, the pre-signed URL works the same way for both encrypted and unencrypted objects.

In client-side encryption, you manage encryption/decryption of your data, the encryption keys, and related tools. Server-side encryption is an alternative to client-side encryption in which Amazon S3 manages the encryption of your data, which frees you from the tasks of managing encryption and encryption keys.

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 encrypt data using any encryption method you want, and then upload the encrypted data using the Amazon Simple Storage Service (Amazon S3) APIs. Most common application languages include cryptographic libraries that allow you to perform encryption in your applications. Two commonly available open source tools are Bouncy Castle and OpenSSL. After you encrypt an object and safely store the key in your KMI, the encrypted object can be uploaded to Amazon S3 directly with a PUT request. To decrypt this data, you issue the GET request in the Amazon S3 API and then pass the encrypted data to your local application for decryption.

AWS provides an alternative to these open source encryption tools with the Amazon S3 encryption client, which is an open source set of APIs embedded into the AWS SDKs. This client allows you to supply a key from your KMI that can be used to encrypt or decrypt your data as part of the call to Amazon S3. The Amazon S3 encryption client is integrated into the AWS SDKs for Java, Ruby, and .NET and provides a transparent drop-in replacement for any cryptographic code you may have used previously with your application that interacts with Amazon S3. Although Amazon provides the encryption method, you control the security of your data because you control the keys for that engine to use. If you're using the Amazon S3 encryption client on-premises, AWS never has access to your keys or unencrypted data. If you're using the client in an application running in Amazon EC2, a best practice is to pass keys to the client using secure transport (e.g., SSL or SSH) from your KMI to help ensure confidentiality.

For more information about securing data at rest with encryption, see

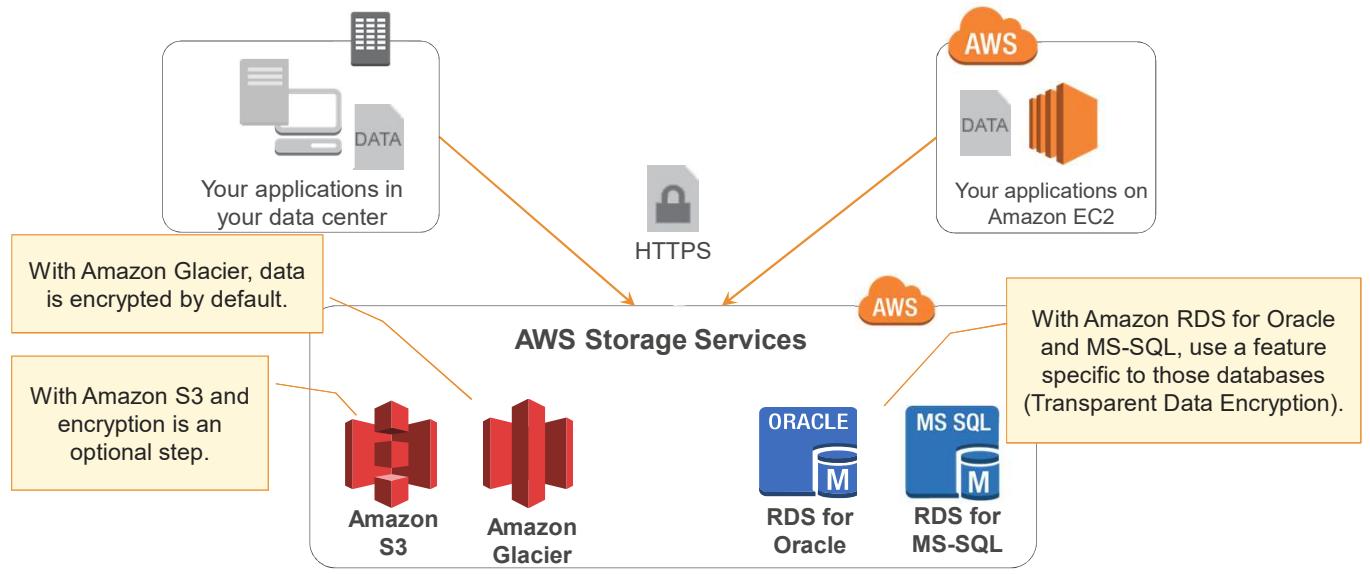
http://media.amazonwebservices.com/AWS_Securing_Data_at_Rest_with_Encryption.pdf

Source And Output Data At Rest In Amazon S3

- 💡 Data stored in Amazon S3 is private by default, requires AWS credentials for access
 - Access to Amazon S3 can be over HTTP or HTTPS
 - Amazon S3 logging allows auditing of access to all objects
 - Amazon S3 supports access control lists and policies for every bucket, prefix (directory/folder), and object
- 💡 Amazon S3 provides server-side encryption (AES-256) using AWS maintained keys or customer provided keys
 - AWS encryption keys are further encrypted with a rotating key
- 💡 Can also encrypt data before storage in Amazon S3 (client-side encryption)

With SSE-C, Amazon S3 encrypts your data on your behalf, using keys that you provide. Your key is discarded immediately after your request, and your key is never stored by Amazon S3. Because Amazon S3 performs the encryption for you, you get the benefits of using your encryption keys without the cost of writing or executing your own encryption code. In addition to REST API updates, the AWS SDKs for Java, Ruby, PHP, and .NET provide the necessary functionality to leverage SSE-C.

AWS Server-Side Encryption



© 2017 Amazon Web Services, Inc. or its affiliates. All rights reserved.



Your source data comes from either systems in your data center or an Amazon EC2 instance. You can upload that data over a secure HTTPS connection to any of five AWS services that support automatic server-side encryption. The service endpoint will handle the encryption and key management processes for you.

With Amazon S3, encryption is an optional step you determine at the time you upload your data to the service.

With Amazon Glacier, all data is encrypted by default.

Amazon RDS for Oracle and Microsoft SQL use a feature specific to those database packages called Transparent Data Encryption, or TDE. TDE uses keys created in the database application with keys created by AWS to protect your data.

Amazon RDS Security Groups



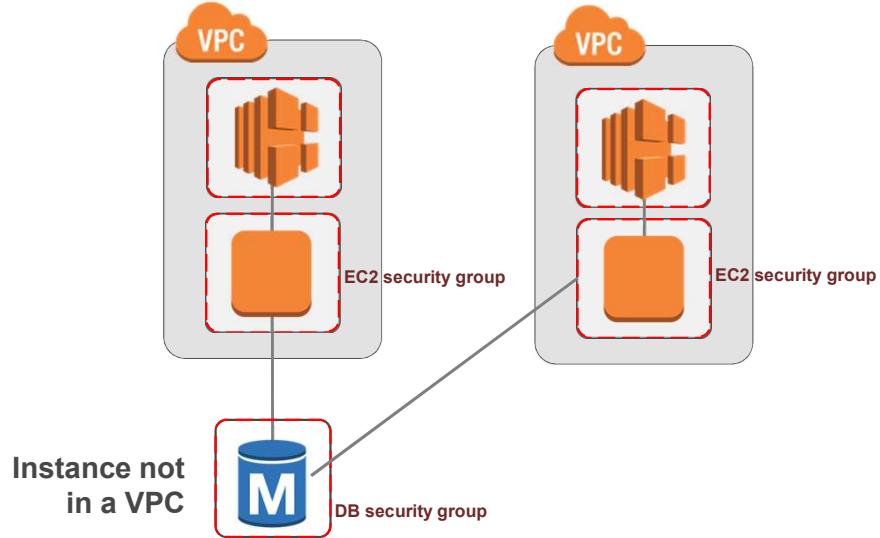
- Control traffic in and out of a DB instance.
- No network access by default.
- Three types:
 - DB security groups
Control access to a DB instance that is not in a VPC.
 - VPC security groups
Control access to a DB instance inside a VPC.
 - EC2 security groups
Control access to EC2 instances.

Security groups control the access that traffic has in and out of a DB instance. Three types of security groups are used with Amazon RDS: DB security groups, VPC security groups, and EC2 security groups. In simple terms, a DB security group controls access to a DB instance that is not in a VPC, a VPC security group controls access to a DB instance (or other AWS instances) inside a VPC, and an EC2 security group controls access to an EC2 instance.

It is no longer possible to launch an Amazon RDS instance without putting it inside a VPC. However, some environments still include Amazon RDS instances hosted outside of VPCs. Those instances will need to use DB security groups.

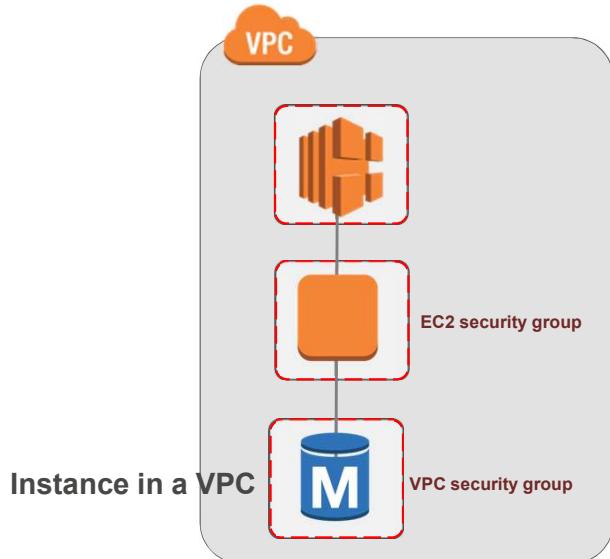
By default, network access is turned off to a DB instance. You can specify rules in a security group that allows access from an IP address range, port, or EC2 security group. When ingress rules are configured, the same rules apply to all DB instances that are associated with that security group. You can specify up to 20 rules in a security group.

Amazon RDS Security Groups



Each DB security group rule enables a specific source to access a DB instance that is associated with that DB security group. The source can be a range of addresses (e.g., 203.0.113.0/24) or an EC2 security group. When you specify an EC2 security group as the source, you allow incoming traffic from all EC2 instances that use that EC2 security group. Note that DB security group rules apply to inbound traffic only; outbound traffic is not currently permitted for DB instances. DB security groups can be created using the Amazon RDS APIs or the Amazon RDS section of the AWS Management Console.

Amazon RDS Security Groups



Each VPC security group rule enables a specific source to access a DB instance in a VPC that is associated with that VPC security group. The source can be a range of addresses (e.g., 203.0.113.0/24) or another VPC security group. By specifying a VPC security group as the source, you allow incoming traffic from all instances (typically application servers) that use the source VPC security group. VPC security groups can have rules that govern both inbound and outbound traffic, although the outbound traffic rules do not apply to DB instances. Note that you must use the Amazon EC2 API or the Security Group option on the VPC Console to create VPC security groups.

DB instances deployed within a VPC can be configured to be accessible from the Internet or from EC2 instances outside the VPC. If a VPC security group specifies a port access such as TCP port 22, you would not be able to access the DB instance because the firewall for the DB instance provides access only via the IP addresses specified by the DB security groups the instance is a member of and the port defined when the DB instance was created. You should use TCP as the protocol for any VPC security group created to control access to a DB instance. The port number for the VPC security group should be the same port number as that used to create the DB instance.

Encrypting Amazon RDS Connections

- 💡 You are responsible for encryption of your **data in-transit**.
- 💡 Use **SSL/TLS** to encrypt connections between applications and DB Instances.
- 💡 Configure your DB instance to **accept only encrypted connections**.
- 💡 Encryption of data connections helps meet **compliance standards**.

You can use SSL/TLS to encrypt connections between your application and your DB instance . For MySQL and SQL Server, RDS creates an SSL certificate and installs the certificate on the DB instance when the instance is provisioned. For MySQL, you launch the MySQL client using the --ssl_ca parameter to reference the public key in order to encrypt connections. For SQL Server, download the public key and import the certificate into your Windows operating system. Oracle RDS uses Oracle native network encryption with a DB instance. You simply add the native network encryption option to an option group and associate that option group with the DB instance. Once an encrypted connection is established, data transferred between the DB Instance and your application will be encrypted during transfer. You can also require your DB instance to only accept encrypted connections.

Note that SSL/TLS support within Amazon RDS is for encrypting the connection between your application and your DB Instance; it should not be relied on for authenticating the DB Instance itself.

While SSL/TLS offers security benefits, be aware that SSL/TLS encryption is a compute intensive operation and will increase the latency of your database connection.

Encrypting Amazon RDS Resources

- 💡 You are responsible for encrypting your **data at-rest**.
- 💡 Use encryption option to encrypt data at-rest.
 - Underlying storage for the instance, its backups, logs, Read Replicas, and snapshots.
 - Uses AES-256 encryption algorithm.
- 💡 Encrypting data-at-rest helps meet compliance standards.
- 💡 Automatic handling of authentication and decryption.
- 💡 Supports TDE for Oracle and SQL Server DB instances.
- 💡 Use AWS Key Management Service to manage keys.

You can encrypt your Amazon RDS instances and snapshots at rest by enabling the encryption option for your Amazon RDS DB instance. Data that is encrypted at rest includes the underlying storage for a DB instance, its automated backups, Read Replicas, and snapshots.

Amazon RDS–encrypted instances use the industry standard AES-256 encryption algorithm to encrypt your data on the server that hosts your Amazon RDS instance. When your data is encrypted, Amazon RDS handles authentication of access and decryption of your data transparently with a minimal impact on performance. You don't need to modify your database client applications to use encryption.

Amazon RDS–encrypted instances provide an additional layer of data protection by securing your data from unauthorized access to the underlying storage. You can use Amazon RDS encryption to increase data protection of your applications and to fulfill compliance requirements for data-at-rest encryption.

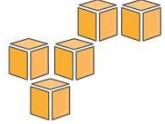
Amazon RDS–encrypted instances are currently available for MySQL, PostgreSQL, Oracle, and SQL Server DB instances.

Amazon RDS also supports encrypting an Oracle or SQL Server DB instance with Transparent Data Encryption (TDE). TDE can be used in conjunction with encryption at rest, although using TDE and encryption at rest simultaneously might slightly affect the performance of your database. You must manage different keys for each encryption method. After you have created an encrypted DB instance, you cannot change the encryption key for that instance; therefore, be sure to determine your encryption key requirements before you create your encrypted DB instance.

To manage the keys used for encrypting and decrypting your Amazon RDS resources, you use the AWS Key Management Service (AWS KMS). AWS KMS combines secure, highly

available hardware and software to provide a key management system scaled for the cloud. Using AWS KMS, you can create encryption keys and define the policies that control how these keys can be used. AWS KMS supports CloudTrail, so you can audit key usage to verify that keys are being used appropriately.

All logs, backups, and snapshots are encrypted for an Amazon RDS–encrypted instance. A Read Replica of an Amazon RDS–encrypted instance is also encrypted using the same key as the master instance.



Part 8

Authentication

© 2017 Amazon Web Services, Inc. or its affiliates. All rights reserved.

amazon
webservices | Academy

Part 8: Authentication



AWS Directory Service

💡 AWS Directory Service is a managed service to:

- Run Microsoft AD as a managed service within AWS Directory Service
- Connect your AWS resources with an existing on-premises Microsoft Active Directory (AD Connector)
- Set up a new, stand-alone directory in the AWS Cloud (Simple AD)

💡 AWS Directory Service allows use of existing corporate credentials for:

- Accessing AWS services (e.g. Amazon WorkSpaces, and Amazon WorkDocs)
- Accessing the AWS Management Console through IAM Roles

💡 Options:

- Run Microsoft AD as a managed service within AWS Directory Service
- Use AD Connector
- Use Simple AD

AWS Directory Service is a managed service that allows you to connect your AWS resources with an existing on-premises Microsoft Active Directory or to set up a new, stand-alone directory in the AWS Cloud.



AWS Directory Service

💡 AWS Directory Service is a managed service to:

- Run Microsoft AD as a managed service within AWS Directory Service
- Connect your AWS resources with an existing on-premises Microsoft Active Directory (AD Connector)
- Set up a new, stand-alone directory in the AWS Cloud (Simple AD)

💡 AWS Directory Service allows use of existing corporate credentials for:

- Accessing AWS services (e.g. Amazon WorkSpaces, and Amazon WorkDocs)
- Accessing the AWS Management Console through IAM Roles

💡 Options:

- Run Microsoft AD as a managed service within AWS Directory Service
- Use AD Connector
- Use Simple AD

Connecting to an on-premises directory is easy and once this connection is established, all users can access AWS resources and applications with their existing corporate credentials. You can also launch managed, Samba-based directories in a matter of minutes, simplifying the deployment and management of Windows workloads in the AWS Cloud.



AWS Directory Service

💡 AWS Directory Service is a managed service

💡 AWS Directory Service allows use of existing corporate credentials

💡 Options:

- Run Microsoft AD as a managed service within AWS Directory Service
 - Powered by Windows Server 2012 R2
 - Created as a highly available pair of domain controllers connected to your VPC
- Use AD Connector
 - Connect to your on-premises Active Directory via VPC VPN connection or AWS Direct Connect
 - Users access AWS applications with existing credentials
 - Integrate with existing RADIUS-based MFA solutions

© 2017 Amazon Web Services, Inc. or its affiliates. All rights reserved.



AD Connector enables you to easily connect your Microsoft Active Directory to the AWS Cloud without requiring complex directory synchronization technologies or the cost and complexity of hosting a federation infrastructure. Once set up, your end users and IT administrators can use their existing corporate credentials to log on to AWS applications such as Amazon Workspaces or Amazon WorkDocs, and to manage AWS resources (e.g. Amazon EC2 instances or Amazon S3 buckets) via AWS Identity and Access Management (IAM) role-based access to the AWS Management Console. As well as providing a streamlined experience for your end users and IT administrators, this also means that your existing security policies such as password expiration, password history, and account lockouts can be enforced consistently whether users or IT administrators are accessing resources in your on-premises infrastructure or the AWS Cloud. You can also use AD Connector to enable multi-factor authentication by integrating with your existing RADIUS-based MFA infrastructure to provide an additional layer of security when users access AWS applications.

A Simple AD is a managed directory which is powered by Samba 4 Active Directory Compatible Server. It supports commonly used features such as user accounts, group memberships, domain-joining Amazon EC2 instances running Microsoft Windows, Kerberos based single sign-on (SSO), and Group Policies. This makes it even easier to manage Amazon EC2 instances running Windows and deploy Windows applications in the AWS Cloud. Many of the applications and tools you use today that require Microsoft Active Directory support can be used with a Simple AD. User accounts in a Simple AD can also be used to access AWS Applications such as

Amazon WorkSpaces and Amazon WorkDocs, and to manage AWS resources via AWS Identity and Access Management (IAM) role-based access to the AWS Management Console.

A Simple AD also provides you with audit trails and event logs so you can track the usage of your directory including which users logged in, when they logged in, and any accounts that have been locked because of failed login attempts. You can use standard Microsoft Windows event log tools to review directory audit and event logs. A Simple AD also provides daily automated snapshots by default to enable point-in-time recovery.

AWS Security Token Service (STS)

STS is a lightweight web service that enables you to request **temporary, limited-privilege credentials** for IAM users or for users that you authenticate (federated users).

Allow trusted entity to assume a role by calling the **AssumeRole** APIs of STS

To view security credentials of a running Amazon EC2 instance, use the following Instance Metadata Service (IMDS) URL:

<http://169.254.169.254/latest/meta-data/iam/security-credentials/>

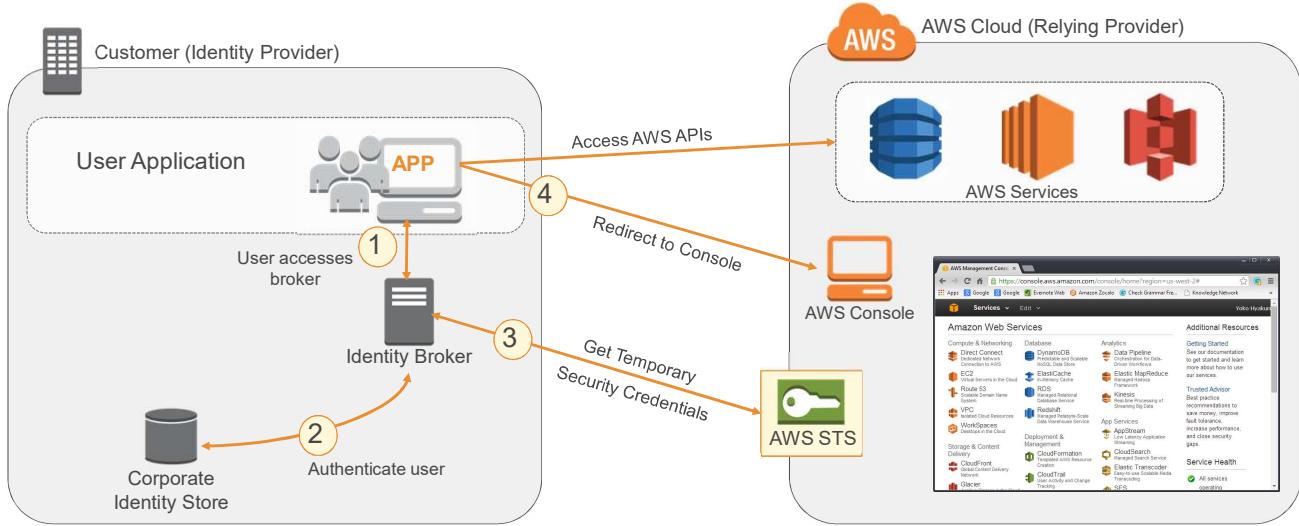
Federated Users

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.
- **Single Sign-On (SSO):** Temporary credentials can be used to sign user directly into the AWS Management Console.

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

Use Case: STS Identity Broker



© 2017 Amazon Web Services, Inc. or its affiliates. All rights reserved.



Academy

In this scenario:

- The identity broker application has permissions to access the AWS STS API to create temporary security credentials.
- The identity broker application is able to verify that employees are authenticated within the existing authentication system.
- Users are able to get a temporary URL that gives them access to the AWS Management Console (which is referred to as single sign-on).

IAM user groups from another AWS account:

You can establish cross-account access by using IAM roles. Using roles for cross-account access lets you grant access to any resources that the trusting account (Account A) has access to, as long as the resource is in a service that supports roles.

IAM users within the current account:

For mission-critical permissions that IAM users might not frequently use, you can separate those permissions from their normal day-to-day permissions by using roles. Users would then have to actively assume a role, which can prevent them from accidentally performing disruptive actions.

For example, you might have Amazon EC2 instances that are critical to your organization. Instead of directly granting administrators permission to terminate the instances, you can create a role with those privileges and allow administrators to

assume the role.

Administrators won't have permission to terminate those instances; they must first assume a role. By using a role, administrators must take an additional step to assume a role before they can stop an instance that's critical to your organization.

Third Parties:

When third parties require access to your organization's AWS resources, you can use roles to delegate API access to them. For example, a third party might provide a service for managing your AWS resources. With IAM roles, you can grant these third parties access to your AWS resources without sharing your AWS security credentials. Instead, they can assume a role that you created to access your AWS resources.

Third parties must provide you the following information for you to create a role that they can assume:

- The AWS account ID that the third party's IAM users use to assume your role. You specify their AWS account ID when you define the trusted entity for the role.
- An external ID that the third party can associate with your role. You specify the ID that was provided by the third party when you define the trusted entity for the role.
- The permissions that the third party requires in order to work with your AWS resources. You specify these permissions when defining the role's permission policy. This policy defines what actions they can take and what resources they can access.

After you create the role, you must share the role's Amazon Resource Name (ARN) with the third party. They require your role's ARN in order to assume the role.

Identity Broker:

- Used to query STS
- Determines user from a web request
- Uses AWS credentials (service account) to authenticate to AWS
- Issues temporary security credentials to access AWS APIs (through STS)
- AWS permissions are configured by the administrator of the identity broker
- Configurable timeout; 1-36 hours
- Sample IIS authentication proxy code provided →
<http://aws.amazon.com/code/1288653099190193>

SSO Federation Using SAML

Amazon STS supports **SAML 2.0**.

Benefits:

- Open standards
- Quicker and easier to implement federation
- Leverage existing identity management software to manage access to AWS resources
- No coding required

AWS Management Console SSO:

- IdP-initiated web SSO via SAML 2.0 using the HTTP-POST binding (web SSO profile)
- New sign-in URL that greatly simplifies SSO:
`https://signin.aws.amazon.com/saml<SAML_AuthN_response>`
- API federation using new assumeRoleWithSAML operation

AWS supports identity federation that makes it easier to manage users by maintaining their identities in a single place which includes the support for the Security Assertion Markup Language (SAML) 2.0, an open standard used by many identity providers. This feature enables federated single sign-on, or SSO, empowering users to sign in to the AWS Management Console or make programmatic calls to AWS APIs, by using assertions from a SAML-compliant identity provider, such as Shibboleth and Windows Active Directory Federation Services.

There are many great use cases that illustrate how identity federation makes user administration easier. For instance, if a user leaves your company, you can simply delete the user's corporate identity, which then also revokes access to AWS. End users also benefit because they only need to remember one user name and password. Using SAML can make configuring federation with AWS easy, because system administrators can set it up using identity provider software instead of writing code.

SSO Federation Using SAML

Amazon STS supports **SAML 2.0**.

Benefits:

- Open standards
- Quicker and easier to implement federation
- Leverage existing identity management software to manage access to AWS resources
- No coding required

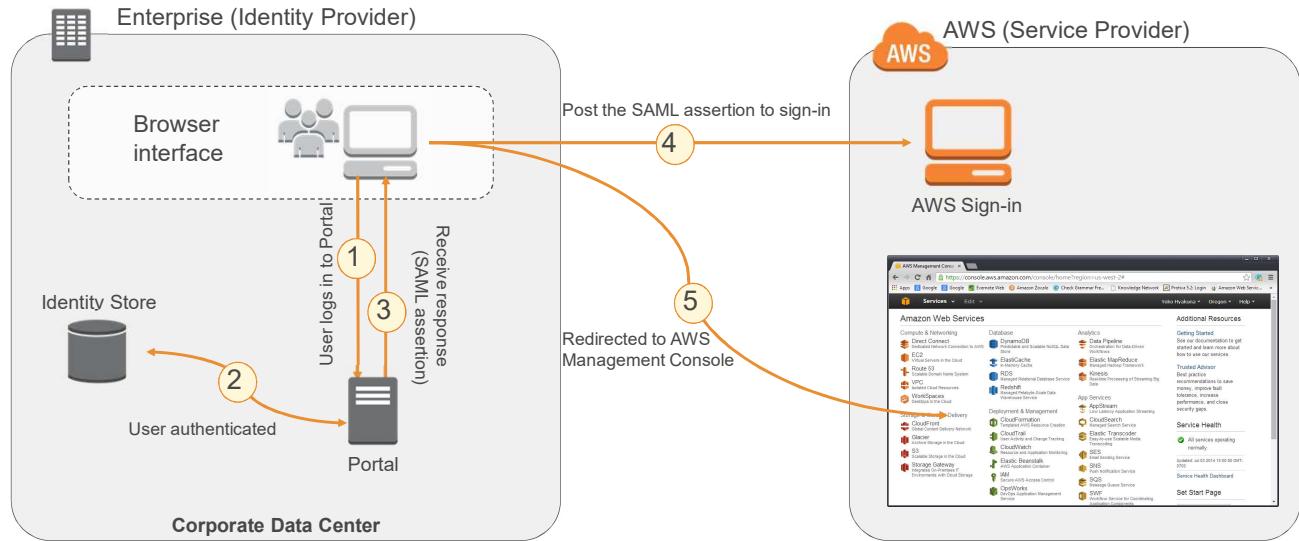
AWS Management Console SSO:

- IdP-initiated web SSO via SAML 2.0 using the HTTP-POST binding (web SSO profile)
- New sign-in URL that greatly simplifies SSO:
https://signin.aws.amazon.com/saml<SAML_AuthN_response>
- API federation using new assumeRoleWithSAML operation

You create a SAML provider in IAM if you want to establish trust between AWS and an Identity Provider (IdP) such as Shibboleth or Active Directory Federation Services, and you want to federate user identities so that users in your organization can access AWS resources. SAML providers in IAM are used as principals in an IAM trust policy.

You can create and manage a SAML provider in the AWS Management Console or by using the CLI or API calls.

SSO Federation Using SAML



© 2017 Amazon Web Services, Inc. or its affiliates. All rights reserved.



Academy

1. User browses to a URL: A user in your organization browses to an internal portal in your network. The portal also functions as the IdP that handles the SAML trust between your organization and AWS.
2. User authenticated: The identity provider (IdP) authenticates the user's identity against AD.
3. Receives authentication response: The client receives a SAML assertion (in the form of authentication response) from the IdP.
4. Post to sign-in passing AuthN: The client posts the SAML assertion to the new AWS sign-in endpoint. Behind the scenes, sign-in uses the AssumeRoleWithSAML API to request temporary security credentials and construct a sign-in URL.
5. Redirect client to AWS Management Console: The user's browser receives the sign-in URL and is redirected to the AWS Management Console.

From the user's perspective, the process happens transparently. The user starts at your organization's internal portal and ends up at the AWS Management Console without ever having to supply any AWS credentials.

The basic steps:

1. Generate a metadata document using your IdP software.
2. Create a SAML provider using that metadata document.
3. Create one or more IAM roles that establish trust with the SAML provider and define permissions for the federated user.
4. Configure your IdP software to map attributes (such as AD groups) to the IAM roles.
5. Finished.

Web Identity Federation

💡 Use STS API, `AssumeRoleWithWebIdentity`

- Lets you request temporary security credentials to access AWS resources.

💡 Supported web identity providers:

- Amazon
- Google
- Facebook

💡 A mobile app can be developed without server-side code and without distributing long-term credentials with the mobile app.

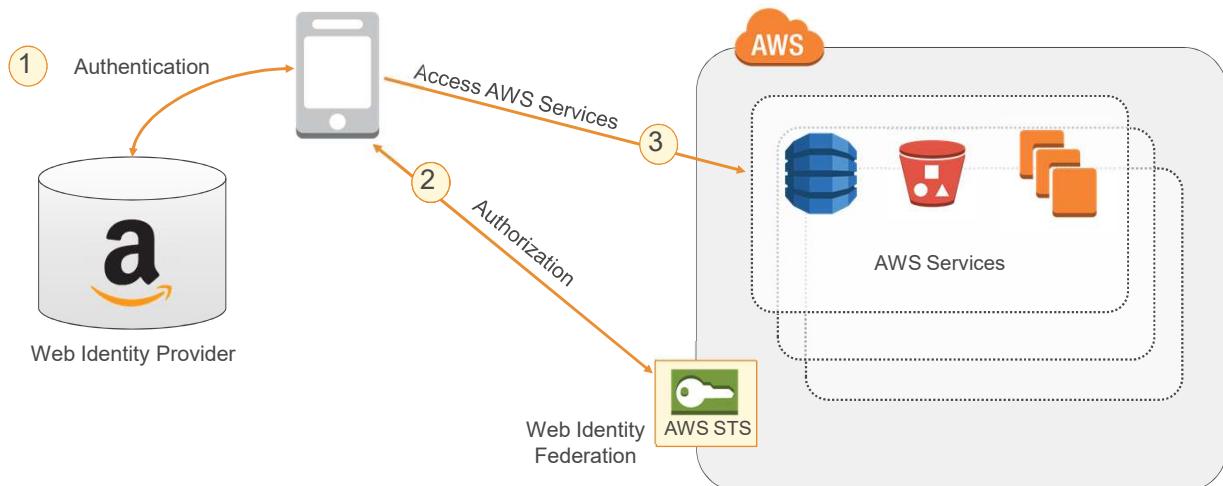
To configure your application and vend federated credentials using trusted identity providers such as Login with Amazon, Facebook, or Google, you want to:

1. Register an application with the identity provider.
2. Create an IAM role for the identity provider.
3. Set up permissions for the IAM role.
4. Use the identity provider's SDK to get an access to token after logging in.
5. Use the AWS SDK for JavaScript to get temporary credentials to your application.

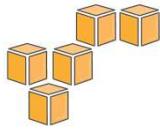
For more information, see the online documentation at

<http://docs.aws.amazon.com/amazondynamodb/latest/developerguide/WIF.RunningYourApp.html>

Use Case: Web Identity Federation



1. The user needs to be authenticated. For example, using the *Login with Amazon* SDK, the app authenticates the user and receives a token from Amazon.
2. The user needs to be authorized to access resources in his/her AWS account. The app makes an unsigned AssumeRoleWithWebIdentity request to STS, passing the token from the previous step. STS verifies the authenticity of the token; if the token is valid, STS returns a set of temporary security credentials to the app. By default, the credentials can be used for one hour.
3. The app uses the temporary security credentials to make signed requests for resources in your Amazon S3 bucket (as an example). Because the role's access policy used variables that reference the app ID and the user ID, the temporary security credentials are scoped to that end user and will prevent him/her from accessing objects owned by other users.



In review...

- Security Pillar Overview
- Principles of the Security Pillar & Key Services
- Preventing Common Security Exploits
- Securing & Encrypting Data
- Securing Data At Rest on S3
- Authentication



Knowledge Assessment

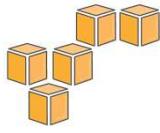
© 2017 Amazon Web Services, Inc. or its affiliates. All rights reserved.



In review...

- Security Pillar Overview
- Principles of the Security Pillar & Key Services
- Preventing Common Security Exploits
- Securing & Encrypting Data
- Securing Data At Rest on S3
- Authentication

To complete this module, please remember to finish the corresponding knowledge assessment.



Up Next...



[CCA 3.09 Worksheet - Improve This Architecture Exercise](#)



[CCA 3.10 - Well-Architected Pillar 2: Reliability](#)

Before you continue, please complete the exercise in the **CCA 3.09 Worksheet** and follow the instructions to complete the exercise and lab. The worksheet also contains links to additional resources you may find useful.

Up next is **CCA 3.10** featuring the **Reliability Pillar** of the Well-Architected framework.

© 2017 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.

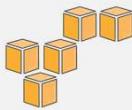
Errors or corrections? Email us at aws-course-feedback@amazon.com.

For all other questions, contact us at
<https://aws.amazon.com/contact-us/aws-training/>.

All trademarks are the property of their owners.

© 2017 Amazon Web Services, Inc. or its affiliates. All rights reserved.





CCA Unit 3 – Architecting on AWS

CCA 3.10: Reliability Pillar

Section 1: Introduction to System Design

Section 2: Automation and Serverless Architectures

Section 3: Well-Architected Best Practices

CCA 3.08 Introducing the Well-Architected Framework

CCA 3.09 Well-Architected Pillar 1: Security

► **CCA 3.10 Well-Architected Pillar 2: Reliability**

CCA 3.11 Well-Architected Pillar 3: Performance Efficiency

CCA 3.12 Well-Architected Pillar 4: Cost Optimization

Section 4: Deployment and Implementation

© 2017 Amazon Web Services, Inc. or its affiliates. All rights reserved.



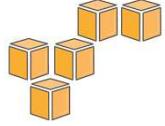
Welcome to **CCA 3.10** – featuring the **Reliability Pillar** of the Well-Architected framework.

What's In This Module?

- Reliability Pillar Overview
- Disaster Recovery
- Managed Desktops in the Cloud
- Reliability Case Study

This module covers...

- An overview of the Reliability Pillar
- Architectural patterns for Disaster Recovery
- Managed Desktops in the Cloud
- A case study looking into a real-life application of the reliability principles



Part 1

Reliability Pillar Overview

© 2017 Amazon Web Services, Inc. or its affiliates. All rights reserved.



Part 1: Reliability Pillar Overview

Pillar 2: Reliability

The ability of a system to:

- **recover** from infrastructure or service failures
- **dynamically acquire** computing resources to meet demand
- **mitigate disruptions** such as misconfigurations or transient network issues



(INSTRUCTOR NOTE: To have a positive effect on learning transfer and retention, your narrative needs to *match* what is on the screen. In e-learning, discordance between screen and narrative can cause split attention.)

“Reliability refers to the ability of a system to [1] **recover** from infrastructure or service failures, [2] **dynamically acquire** computing resources to meet demand, and to [3] **mitigate disruptions** such as misconfigurations or transient network issues.”

In the cloud, several principles can help you increase reliability. The goal is to keep the impact of any type of failure to the smallest area possible. By preparing your system for the worst, you can implement a variety of mitigation strategies for the different components of your infrastructure and applications.

Before a failure occurs, you want to test your recovery procedures. It is important to try to automate recovery as much as possible to reduce the possibility of human error.

Scaling can also help with achieving high availability.

Test Recovery Procedures

In an on-premises environment, testing is often conducted to prove that the system works in a particular scenario; testing is not typically used to validate recovery strategies.

In the cloud, you can **test how your system fails**, and you can **validate recovery procedures**.

- ❶ **Use automation** to simulate different failures or to recreate scenarios that led to failures before.
- ❷ **Expose failure pathways** so you can test and rectify before a real failure scenario, thus reducing the risk of failure for untested components.



Test recovery procedures:

In an on-premises environment, testing is often conducted to prove the system works in a particular scenario; testing is not typically used to validate recovery strategies.

[1] In the cloud, you can test how your system fails, and you can validate your recovery procedures.

- You can [2] **use automation** to simulate different failures or to recreate scenarios that led to failures before.
- This [3] **exposes failure pathways** that you can test and rectify *before* a real failure scenario, reducing the risk of components failing that have not been tested before.

Automatically Recover From Failure

By monitoring a system for **key performance indicators (KPIs)** (KPIs), you can trigger automation when a threshold is breached.

- Allows for **automatic notification and tracking of failures** for automated recovery processes that work around or repair the failure.
- With more sophisticated automation, you can **anticipate and remediate failures** before they occur.



Automatically recover from failure: By monitoring a system for key performance indicators (KPIs), you can trigger automation when a threshold is breached. This allows for automatic notification and tracking of failures and for automated recovery processes that work around or repair the failure. With more sophisticated automation, it is possible to anticipate and remediate failures before they occur.

Scale Horizontally To Increase Aggregate System Availability

- Replace one large resource with multiple small resources to reduce the impact of a single failure on the overall system.
- Distribute requests across multiple smaller resources to ensure that they don't share a common point of failure.



Scale horizontally to increase aggregate system availability: Replace one large resource with multiple small resources to reduce the impact of a single failure on the overall system. Distribute requests across multiple smaller resources to ensure that they don't share a common point of failure.

Stop Guessing Capacity

- 💡 A common cause of failure in on-premises systems is **resource saturation**, when the demands placed on a system exceed the capacity of that system (such as denial-of-service attacks).
- 💡 In the cloud, you can **monitor demand and system utilization** and **automate** the addition or removal of resources. This allows you to maintain the level of resources that will satisfy demand without over- or under-provisioning.



Stop guessing capacity: A common cause of failure in on-premises systems is resource saturation, when the demands placed on a system exceed the capacity of that system (this is often the objective of denial-of-service attacks). In the cloud, you can monitor demand and system utilization and automate the addition or removal of resources to maintain the optimal level to satisfy demand without over- or under-provisioning.

Suggested: In the cloud, you can monitor demand and system utilization and automate the addition or removal of resources. This allows you to maintain the level of resources that will satisfy demand without over- or under-provisioning.

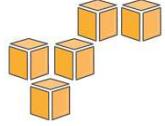
Key Services For Reliability



Areas	Key Services
Foundations	 AWS IAM  Amazon VPC
Change management	 AWS CloudTrail  AWS Config
Failure management	 AWS CloudFormation

The AWS service that is key to ensuring reliability is [1] Amazon CloudWatch, which monitors run-time metrics. Other services and features that support the three areas of Reliability are as follows:

- **[2] Foundations:** AWS Identity and Access Management (IAM) enables you to securely control access to AWS services and resources. Amazon VPC lets you provision a private, isolated section of the AWS cloud where you can launch AWS resources in a virtual network.
- **[3] Change management:** AWS CloudTrail records AWS API calls for your account and delivers log files to you for auditing. AWS Config provides a detailed inventory of your AWS resources and configuration, and continuously records configuration changes.
- **[4] Failure management:** AWS CloudFormation enables the template creation of AWS resources and provisions them in an orderly and predictable fashion.



Part 2

Disaster Recovery

Making Your Infrastructure More Reliable

© 2017 Amazon Web Services, Inc. or its affiliates. All rights reserved.



Part 2: Disaster Recovery

Common Practices For Disaster Recovery On AWS

Example architectural patterns:

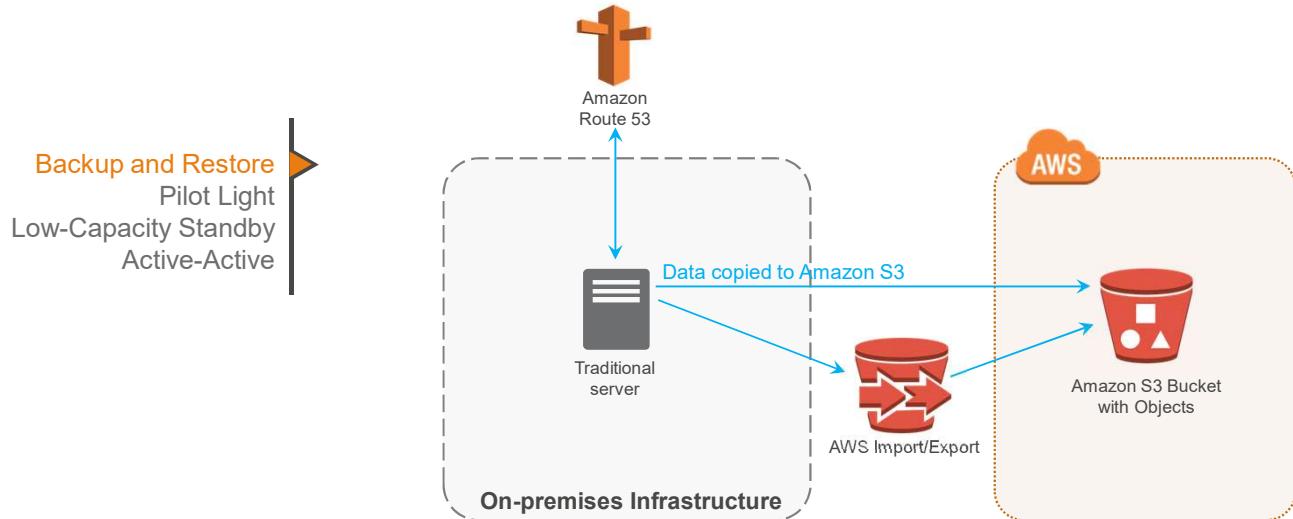


- Backup and Restore
- Pilot Light
- Fully Working Low-Capacity Standby
- Multi-Site Active-Active

We will touch on the following disaster recovery patterns (sorted from highest to lowest RTO/RPO):

- Backup and Restore
- Pilot Light
- Fully Working Low-Capacity Standby
- Multi-Site Active-Active

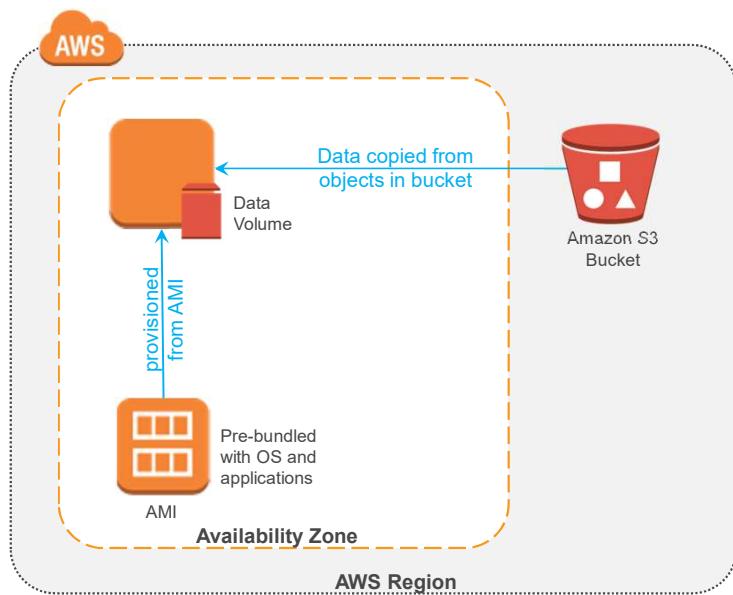
Backup and Restore



This diagram shows a basic disaster recovery system. In this diagram, the data is in a traditional data center, with backups stored in AWS. AWS Import/Export is used to get the data into AWS, and the data is then stored in Amazon S3.

Backup and Restore

Backup and Restore
Pilot Light
Low-Capacity Standby
Active-Active



To recover the data, use an AMI (custom or pre-configured) to create one or more Amazon EC2 instances. Then, add the Amazon S3 data by copying it to the Amazon EC2 instances.

Backup and Restore

Backup and Restore
Pilot Light
Low-Capacity Standby
Active-Active

Advantages

- Simple to get started
- Extremely cost effective (mostly backup storage)

Preparation Phase

- Take backups of current systems
- Store backups in Amazon 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.

This disaster recovery pattern is easy to set up and rather inexpensive. You do need to think about what AMIs you need for recovery.

Backup and Restore

Backup and Restore
Pilot Light
Low-Capacity Standby
Active-Active

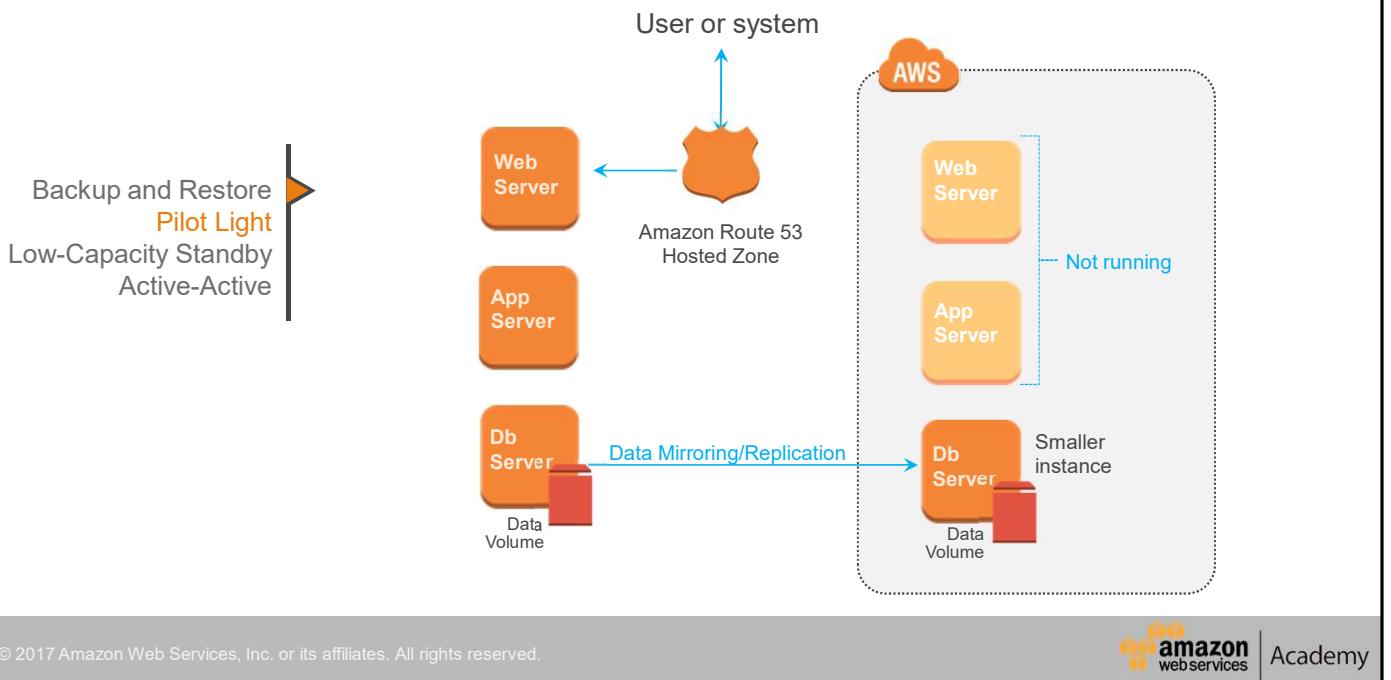
In case of disaster

- Retrieve backups from Amazon S3.
- Bring up required infrastructure.
 - Amazon EC2 instances with prepared AMIs, Elastic Load Balancing, etc.
 - Use Amazon CloudFormation to automate deployment of core networking.
- Restore system from backup.
- Switch over to the new system.
 - Adjust DNS records to point to AWS.

Objectives

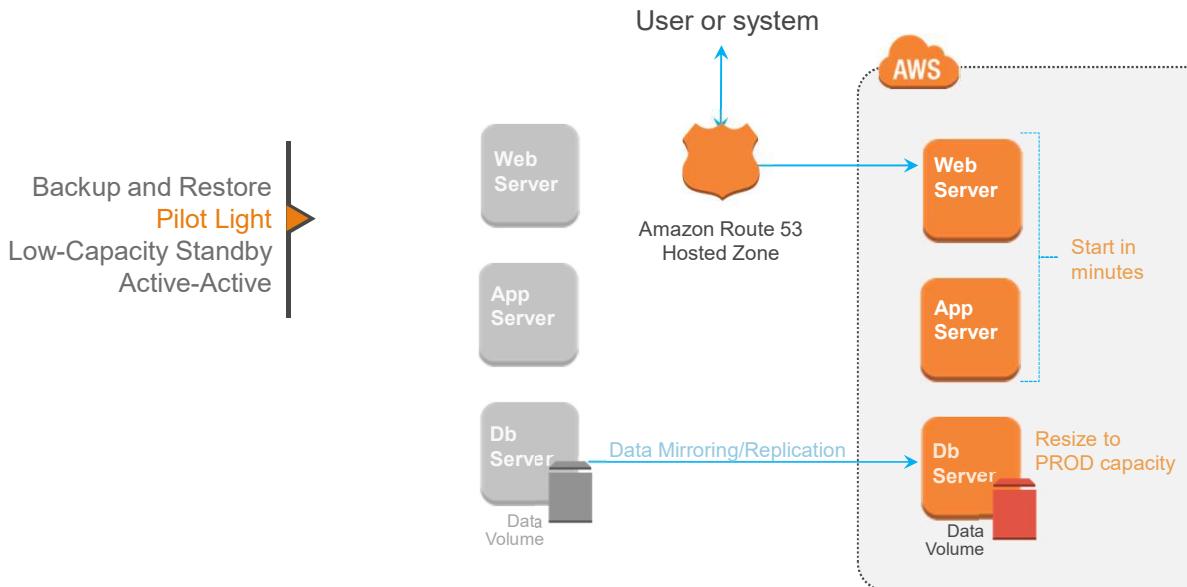
- **RTO:** as long as it takes to bring up infrastructure and restore system from backups.
- **RPO:** time since last backup.

Pilot Light



This diagram shows a Pilot Light disaster recovery pattern. In this case, the database data is replicated into AWS, with Amazon EC2 instances of the web servers and application servers ready to go, but currently not running. Notice that Amazon Route 53 is also used here.

Pilot Light



© 2017 Amazon Web Services, Inc. or its affiliates. All rights reserved.



Now, if the primary system fails, when you start up the Amazon EC2 instances, they use the latest copy of the data. Then you redirect Amazon Route 53 to point to the new web server.

Pilot Light

Backup and Restore
Pilot Light
Low-Capacity Standby
Active-Active

Advantage

- 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, Elastic Load Balancing, etc.
- Reserved Instances

This pattern is relatively inexpensive to implement.

Pilot Light

Backup and Restore
Pilot Light
Low-Capacity Standby
Active-Active

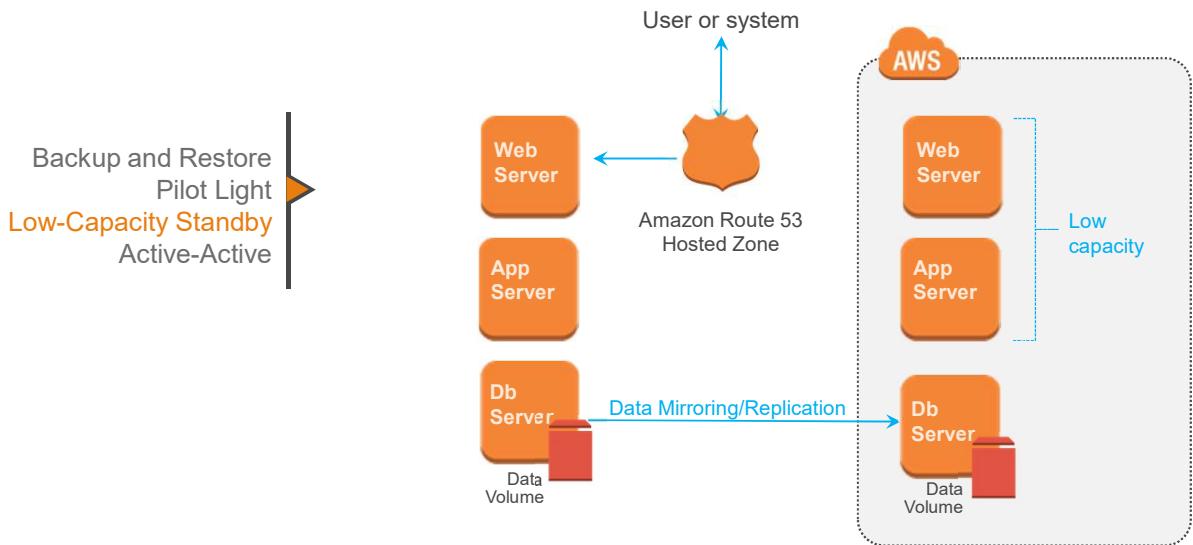
In case of disaster

- 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.

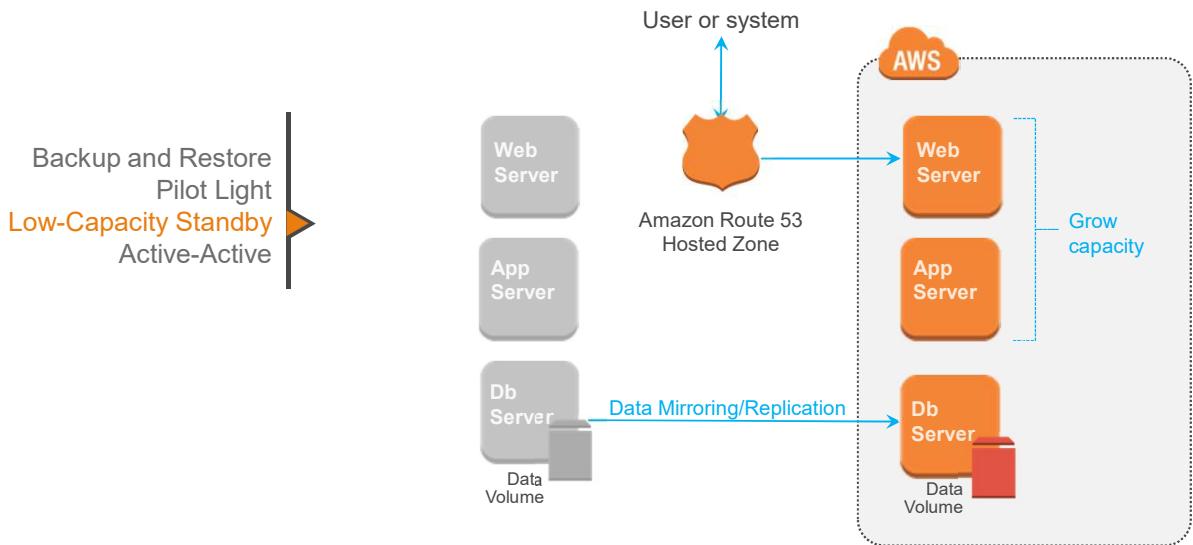
Objectives

- **RTO:** as long as it takes to detect need for DR and automatically scale up replacement system.
- **RPO:** depends on replication type.

Fully Working Low-Capacity Standby



Fully Working Low-Capacity Standby

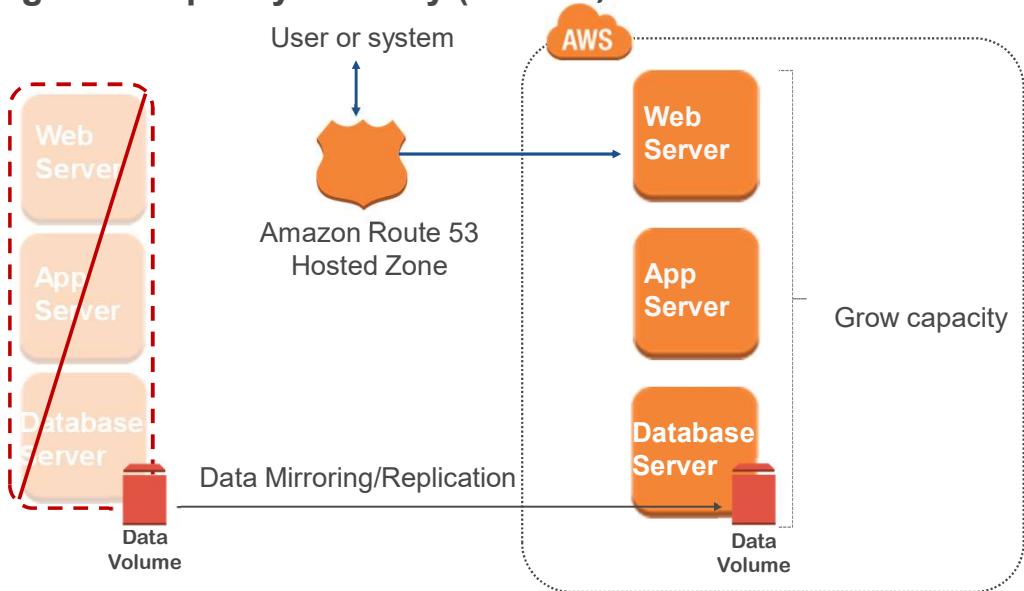


© 2017 Amazon Web Services, Inc. or its affiliates. All rights reserved.

amazon webservices | Academy

If the primary environment is unavailable, Amazon Route 53 switches over to the secondary system, which is designed to automatically scale its capacity up in the event of a failover from the primary system.

Fully Working Low-Capacity Standby (Cont'd.)



Fully Working Low-Capacity Standby (Cont'd.)

Backup and Restore
Pilot Light
Low-Capacity Standby
Active-Active



💡 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

This pattern is more expensive because active systems are running.

Fully Working Low-Capacity Standby

Backup and Restore
Pilot Light
Low-Capacity Standby
Active-Active

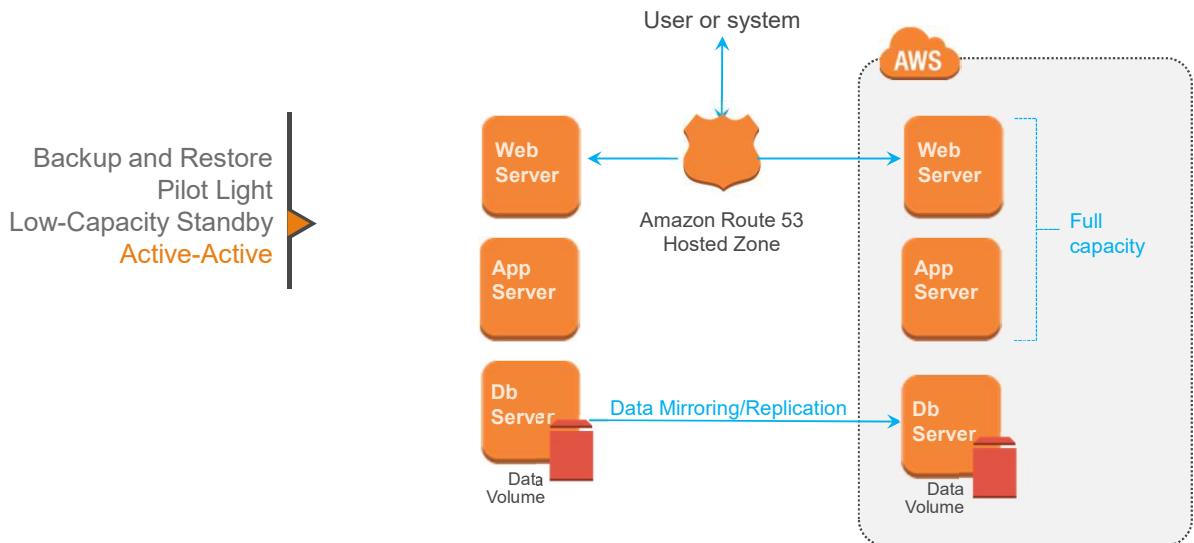
💡 In case of disaster

- 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

Multi-Site Active-Active



The next level of disaster recovery is to have a fully functional system running in AWS at the same time as the on-premises systems.

Multi-Site Active-Active (Cont'd.)

Backup and Restore
Pilot Light
Low-Capacity Standby
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

Objectives

- **RTO:** as long as it takes to fail over
- **RPO:** depends on replication type

This pattern potentially has the least downtime of all. It does, of course, have more costs associated with it, because more systems are running.

Best Practices For Being Prepared

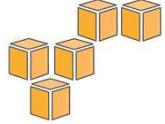
- Start simple and work your way up
 - Backups in AWS as a first step
 - Incrementally improve RTO/RPO as a continuous effort
- Check for any software licensing issues
- Exercise your DR Solution
 - Practice "Game Day" exercises
 - Ensure backups, snapshots, AMIs, etc. are working
 - Monitor your monitoring system

As you start thinking about disaster recovery and AWS, here is some important advice:

- Start small and build as needed.
- Check your licenses!
- Test your solutions often.

Advantages Of Disaster Recovery With AWS

- Various building blocks available
- Fine control over cost vs. RTO/RPO tradeoffs
- Ability to easily and effectively test your DR plan
- Availability of multiple locations worldwide
- Managed desktops available
- Amazon WorkSpaces provides a fully managed desktop computing service in the cloud
- Variety of solution providers



Part: 3

Managed Desktops in the Cloud

© 2017 Amazon Web Services, Inc. or its affiliates. All rights reserved.

 Academy

Part 3: Managed Desktops in the Cloud

Amazon WorkSpaces



A fully managed desktop computing services in the cloud.

Key benefits:

- Fully managed
- Supports multiple devices
- Keeps data secure and available
- You can choose software and hardware
- Price per month
- Corporate directory integration



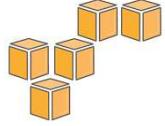
© 2017 Amazon Web Services, Inc. or its affiliates. All rights reserved.

Amazon
webservices | Academy

Amazon WorkSpaces is a fully managed service. All the heavy lifting and provisioning of hardware and software is done for customers by AWS. All they need to do is launch the number of workspaces they need. The service takes care of all underlying provisioning processes; when the workspaces are ready, users will receive an email providing them with instructions on how to obtain a client and connect to their workspaces.

Amazon WorkSpaces provides customers with a choice of devices they can use to connect to their desktop. They can use an iPad, a Kindle Fire HDK (including the ability to use a keyboard and mouse), or a Windows or Mac desktop. iPad and Android clients have numerous optimizations to make a desktop experience on the device intuitive, such as a slide-out radial control to access commonly used functions and a choice of mouse modes. Amazon WorkSpaces delivers only pixels to users, using the Teradici PCoIP protocol, and customer data does not stay on the end user's device. The user volume provided for a user's workspaces is regularly backed up to Amazon S3 as a snapshot, which helps ensure data durability even in the case of hardware failure.

Amazon WorkSpaces integrates with customers' corporate directories. This means that all workspaces provisioned by a customer will join the customer's Active Directory domain. Users can continue to use their existing corporate credentials to get seamless access to corporate resources (e.g., Exchange, SharePoint, and other internal applications). This also means that for administrators, as the workspaces join the customer's Active Directory domain, they can be managed just like any other desktops with management tools or processes that customers are already using.



Section 4

Reliability Case Study

© 2017 Amazon Web Services, Inc. or its affiliates. All rights reserved.

 **amazon**
webservices Academy

Section 4: Reliability Case Study

Unilever Uses AWS To Standardize Deployments

“ We needed to standardize our environment to support a faster time-to-market.

Sreenivas Yalamanchili
Global Technical Manager,
Unilever Digital marketing



Unilever is a global company with more than 400 individual brands sold in 180 countries. Today, the company's product line covers all facets of daily life, from food to cleaning to health and well-being.

- Unilever is an international consumer goods corporation with a presence in more than 190 countries
- Needed an infrastructure capable of supporting its digital marketing campaigns
- Unilever used AWS to build a standardized digital marketing platform, migrated 500 web properties in less than 5 months, and reduce time to launch new projects by 75%

Unilever Used AWS To Enable Better Disaster Recovery



Amazon S3

Backup data, snapshots, product and recipe media files stored durably and made available to all regions



Amazon EBS

EBS Snapshot Copy used to backup SQL Server on Amazon EC2 data volumes and restore them to the disaster recovery region when necessary



Amazon EC2

Created ~400 AMIs of Windows and Linux instances to provide flexible deployment across environments

In DR region, all instances except master SQL Server are kept on standby, activated when needed

Instances kept on standby in DR region: CMS LDAP, Active Directory, and SQL Server standby and witness instances. Master SQL Server instance in DR region kept on and mirrored with master SQL Server instance in primary region.

Disaster Recovery on AWS: Unilever Digital Marketing

“

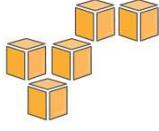
“We designed a disaster recovery solution to protect our content management system, content deployment architecture, and many GOLD-classified web properties—and to give the business confidence in the AWS Cloud.

Sreenivas Yalamanchili
Global Technical Manager, Unilever Digital marketing



”

For more on how Unilever uses AWS, see: <https://aws.amazon.com/solutions/case-studies/unilever/>



In review...

- 💡 Reliability Pillar Overview
- 💡 Disaster Recovery
- 💡 Managed Desktops in the Cloud
- 💡 Reliability Case Study



Knowledge Assessment

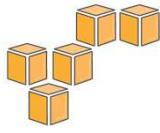
© 2017 Amazon Web Services, Inc. or its affiliates. All rights reserved.

 Academy

In review...

- Reliability Pillar Overview
- Disaster Recovery
- Managed Desktops in the Cloud
- Reliability Case Study

To complete this module, please remember to finish the corresponding knowledge assessment.



Up Next...



CCA 3.10 Worksheet - Improve This Architecture Exercise



LAB 13 - Multi-Region Failover with Amazon Route 53



CCA 3.11 - Well Architected Pillar 3: Performance Efficiency

Before you continue, please complete the exercise in the **CCA 3.10 Worksheet** and **Lab 13**, Multi-Region Failover with Amazon Route 53.

Up next is **CCA 3.11** featuring the **Performance Efficiency** pillar of the Well-Architected framework.

© 2017 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.

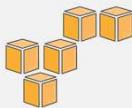
Errors or corrections? Email us at aws-course-feedback@amazon.com.

For all other questions, contact us at
<https://aws.amazon.com/contact-us/aws-training/>.

All trademarks are the property of their owners.

© 2017 Amazon Web Services, Inc. or its affiliates. All rights reserved.





CCA Unit 3 – Architecting on AWS

CCA 3.11: Performance Efficiency Pillar

Section 1: Introduction to System Design

Section 2: Automation and Serverless Architectures

Section 3: Well-Architected Best Practices

CCA 3.08 Introducing the Well-Architected Framework

CCA 3.09 Well-Architected Pillar 1: Security

CCA 3.10 Well-Architected Pillar 2: Reliability

► CCA 3.11 Well-Architected Pillar 3: Performance Efficiency

CCA 3.12 Well-Architected Pillar 4: Cost Optimization

Section 4: Deployment and Implementation

© 2017 Amazon Web Services, Inc. or its affiliates. All rights reserved.



Welcome to **CCA 3.11** – featuring the **Performance Efficiency Pillar** of the Well-Architected framework.

What's In This Module?

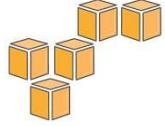
💡 Performance Efficiency Pillar Overview

💡 Performance Optimization

This module covers...

- An **overview** of the Performance Efficiency Pillar
- Making your perform more efficiently with **performance optimization**

NOTE: At the end of this module you will download a **worksheet** to complete an exercise for the Performance Efficiency pillar.



Part 1

Performance Pillar Overview

© 2017 Amazon Web Services, Inc. or its affiliates. All rights reserved.

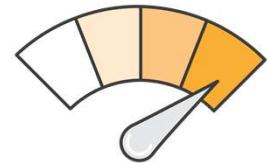
 Academy

Part 1: Performance Pillar Overview

Pillar 3: Performance Efficiency

The ability to **use computing resources efficiently** to meet system requirements and to **maintain that efficiency** as demand changes and technologies evolve.

- Democratize advanced technologies
- Go global in minutes
- Use "serverless" architectures
- Experiment more often



In the cloud, several principles can help you achieve performance efficiency. The goal here is really about getting the most out of your architecture.

Using advanced technologies, such as edge locations or edge services, can reduce latency for services and reduce the load on your system. This can reduce cost at the same time.

Deploy an infrastructure without creating or having to patch EC2 instances by using a "serverless" architecture. This is becoming more and more common. A big driver of this is the availability of managed services on AWS such as AWS Lambda and Amazon API Gateway, and regional services like Amazon S3 and Amazon DynamoDB.

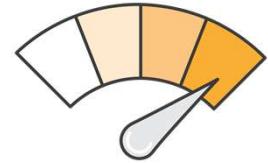
Democratize Advanced Technologies

Technologies that are difficult to implement can become easier to consume by **pushing that knowledge and complexity** into the cloud vendor's domain.

Instead of your IT team having to learn how to host and run a new technology, they can simply **consume it as a service**.

Example:

NoSQL databases, media transcoding, and machine learning are all technologies that require expertise to implement. In the cloud, these technologies become services that your team can consume.

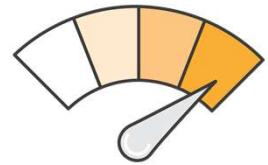


Democratize advanced technologies: Technologies that are difficult to implement can become easier to consume by pushing that knowledge and complexity into the cloud vendor's domain. Instead of your IT team learning how to host and run a new technology, they can simply consume it as a service. For example, NoSQL databases, media transcoding, and machine learning are all technologies that require expertise that is not evenly dispersed across the technical community. In the cloud, these technologies become services that your team can consume while focusing on product development instead of resource provisioning and management.

Go Global In Minutes

Easily deploy your system in **multiple regions** around the world with just a few clicks.

This allows you to **provide lower latency** and a **better experience** for your customers simply and at a minimal cost.



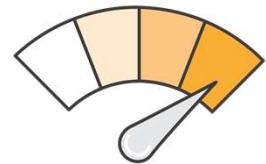
Go global in minutes: Easily deploy your system in multiple regions around the world with just a few clicks. This allows you to provide lower latency and a better experience for your customers simply and at minimal cost.

User Serverless Architectures

In the cloud, serverless architectures remove the need for you to run and maintain servers to carry out traditional compute activities.

For example: storage services can act as static websites, removing the need for web servers, and event services can host your code for you.

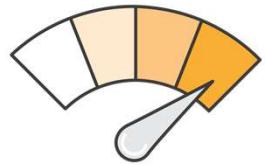
This not only removes the operational burden of managing these servers, but also can lower transactional costs because these managed services operate at cloud scale.



Use *serverless* architectures: In the cloud, serverless architectures remove the need for you to run and maintain servers to carry out traditional compute activities. For example, storage services can act as static websites, thus removing the need for web servers; event services can host your code for you. This not only removes the operational burden of managing these servers, but also can lower transactional costs because these managed services operate at cloud scale.

Experiment More Often

With virtual and automated resources, you can **quickly carry out comparative testing** using different types of instances, storage, or configurations.



Experiment more often: With virtual and automatable resources, you can quickly carry out comparative testing using different types of instances, storage, or configurations.

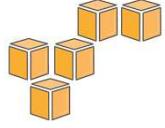
Key Services For Performance Efficiency



Areas	Key Services						
Compute			Auto Scaling				
Storage			Amazon EBS		Amazon S3		Amazon Glacier
Database			Amazon RDS		Amazon DynamoDB		
Space-time trade-off			Amazon CloudFront				

The key AWS service for performance efficiency is Amazon CloudWatch, which monitors your resources and systems, providing visibility into your overall performance and operational health. The following services are important in the four areas of performance efficiency:

- **Compute:** Auto Scaling is key to ensuring that you have enough instances to meet demand and maintain responsiveness.
- **Storage:** Amazon EBS provides a wide range of storage options (such as SSD and PIOPS) that allow you to optimize for your use case. Amazon S3 provides reduced-redundancy storage, lifecycle policies to Amazon Glacier (archival storage), and server-less content delivery.
- **Database:** Amazon RDS provides a wide range of database features (such as provisioned IOPS, and read replicas) that allow you to optimize for your use case. Amazon DynamoDB provides single-digit millisecond latency at any scale.
- **Space-time trade-off:** AWS has regions across the globe, allowing you to choose the optimal location for your resources, data, and processing. Use Amazon CloudFront to cache content even closer to your users.



Section 2

Performance Optimization

Making Your Infrastructure More Reliable

© 2017 Amazon Web Services, Inc. or its affiliates. All rights reserved.

 Academy

Section 2: Performance Optimization - Making Your Infrastructure Perform More Efficiently

Choosing The Right EC2 Instance Size

- Evaluate the most important performance metrics for your application and choose the appropriate instance family and instance type.
- Avoid over-provisioning and under-provisioning.
- Change instance types and sizes as your needs change.
- Analyze if your application can scale out across multiple EC2 instances by design. Design applications that are resilient to reboot and relaunch to allow for scaling horizontally instead of vertically where possible.

Storage Comparison

	Instance Store	Amazon EBS	Amazon S3	Amazon Glacier
Average latency	ms	ms	ms, sec, min (~ size)	hrs
Data volume	4 GB – 48 TB	1 GiB – 16 TiB	No limit	No limit
Item size	Block storage	Block storage	5 TB max	40 TB max
Request rate	Very High	Very High	Low–Very High (no limit)	Very Low (no limit)
Cost/GB per month	Amazon EC2 instance cost	¢¢	¢	¢
Durability	Low	High	Very High	Very High

Hot  Cold

Database Comparison

	DynamoDB	Amazon RDS	Amazon Redshift
Average latency	ms	ms, sec	sec,min
Data volume	No limit	5 GB – 64 TB max (Max varies depending on engine)	1.6 PB max
Item size	KB (400 KB max)	KB (~rowsize)	KB (64 K max)
Request rate	Very High	High	Low
Cost/GB per month	¢¢	¢¢	¢
Durability	Very High	High	High

Hot



Cold

Amazon ElastiCache

- Web service that makes it easy to deploy, operate, and scale an **in-memory cache** in the cloud.
- **Improves performance** of web applications by allowing you to retrieve information from fast, managed, in-memory caches, instead of relying entirely on slower disk-based databases.
- **Supports Memcached and Redis** open-source in-memory caching engines.

Amazon ElastiCache can be used to significantly improve latency and throughput for many read-heavy application workloads or compute-intensive workloads by allowing you to store the objects that are often read in cache. Moreover, with Redis's support for advanced data structures, you can augment the database tier to provide features that are not easily achievable via databases in a cost-effective way.

Amazon ElastiCache simplifies and offloads the management, monitoring, and operation of in-memory cache environments, enabling you to focus on the differentiating parts of your applications.

Amazon ElastiCache provides:

- Support for two engines: Memcached and Redis
- Ease of management via the AWS Management Console.
- Compatibility with the specific engine protocol. This means most of the client libraries will work with the respective engines they were built for - no additional changes or tweaking required.
- Detailed monitoring statistics for the engine nodes at no extra cost via Amazon CloudWatch
- Pay only for the resources you consume based on node hours used

Amazon ElastiCache: <http://aws.amazon.com/elasticsearch/>

Video: Introduction to Amazon ElastiCache: <https://youtu.be/8eD2eNljURE>

Amazon Kinesis

- Enables you to build custom applications that process or analyze **streaming data**.
- Can continuously capture and store **terabytes of data per hour** from **hundreds of thousands of sources**, such as
 - Website clickstreams
 - Financial transactions
 - Social media feeds
 - IT logs

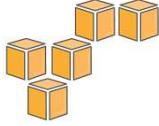
NOTE: Amazon Kinesis will be discussed in depth in a later module.

Amazon Kinesis Streams enables you to build custom applications that process or analyze streaming data for specialized needs. You can configure hundreds of thousands of data producers to continuously put data into an Amazon Kinesis stream. For example, data from website clickstreams, application logs, and social media feeds. Within less than a second, the data will be available for your Amazon Kinesis Applications to read and process from the stream.

With Amazon Kinesis Client Library (KCL), you can build Amazon Kinesis Applications and use streaming data to power real-time dashboards, generate alerts, implement dynamic pricing and advertising, and more. You can also emit data from Amazon Kinesis Streams to other AWS services such as Amazon Simple Storage Service (Amazon S3), Amazon Redshift, Amazon EMR, and AWS Lambda.

Amazon Kinesis Streams: <https://aws.amazon.com/kinesis/streams/>

Video: Introduction to Amazon Kinesis Streams: <https://youtu.be/MbEfiX4sMXc>



In review...

- 💡 Performance Efficiency Pillar Overview
- 💡 Performance Optimization



Knowledge Assessment

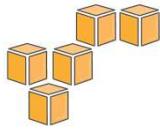
© 2017 Amazon Web Services, Inc. or its affiliates. All rights reserved.

 Academy

In review...

- Performance Efficiency Pillar Overview
- Performance Optimization

To complete this module, please remember to finish the corresponding knowledge assessment.



Up Next...



[CCA 3.11 Worksheet - Improve This Architecture Exercise](#)



[CCA 3.12 - Well Architected Pillar 4: Cost Optimization](#)

Before you continue, please complete the exercise in the [CCA 3.11 Worksheet](#).

Up next is **CCA 3.12** featuring the **Cost Optimization** pillar of the Well-Architected framework.

© 2017 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.

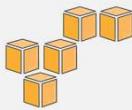
Errors or corrections? Email us at aws-course-feedback@amazon.com.

For all other questions, contact us at
<https://aws.amazon.com/contact-us/aws-training/>.

All trademarks are the property of their owners.

© 2017 Amazon Web Services, Inc. or its affiliates. All rights reserved.





CCA Unit 3 – Architecting on AWS

CCA 3.12: Cost Optimization Pillar

Section 1: Introduction to System Design

Section 2: Automation and Serverless Architectures

Section 3: Well-Architected Best Practices

CCA 3.08 Introducing the Well-Architected Framework

CCA 3.09 Well-Architected Pillar 1: Security

CCA 3.10 Well-Architected Pillar 2: Reliability

CCA 3.11 Well-Architected Pillar 3: Performance Efficiency

► CCA 3.12 Well-Architected Pillar 4: Cost Optimization

Section 4: Deployment and Implementation

© 2017 Amazon Web Services, Inc. or its affiliates. All rights reserved.



Welcome to **CCA 3.10** – featuring the **Reliability Pillar** of the Well-Architected framework.

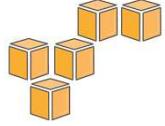
What's In This Module?

- Cost Optimization Pillar overview
- Key services for optimizing cost
- Pricing options and instance types
- Cost management tools

This module covers...

- An **overview** of the Cost Optimization Pillar
- key services for optimizing costs,
- pricing options and instance types, and
- cost management tools

NOTE: At the end of this module you will download a **worksheet** to complete an exercise for the Cost Optimization pillar.



Part 1

Cost Optimization Pillar Overview

© 2017 Amazon Web Services, Inc. or its affiliates. All rights reserved.



Part 1: Cost Optimization Pillar Overview

Best Practice: Optimize For Cost

Take advantage of AWS's flexible platform to increase your cost efficiency.

Ensure that your resources are **sized appropriately**, that they **scale in and out** based on need, and that you're taking advantage of **different pricing options**.

Things to consider:

- 💡 Are my resources the right size and type for the job?
- 💡 What metrics should I monitor?
- 💡 How do I make sure that resources not in use are turned off?
- 💡 How often will I need to use this resource?
- 💡 Can I replace any of my servers with managed services?

Another advantage to the flexible platform from AWS is the ability to match your costs to your specific needs; because you're replacing capital expense with variable expense, the best way to build your infrastructure for cost is to make sure you're only paying for what you need.

Additionally, within each service there are frequently different pricing tiers and models or different configurations within each service that you can leverage to optimize for cost.

Pillar 4: Cost Optimization

The ability to avoid or eliminate unneeded cost or suboptimal resources.



In the cloud you can follow several principles that help you achieve cost optimization. If you follow cost optimization best practices, you should have a good cost comparison with on-premises, but it's always possible to reduce your costs in the cloud as your applications and environments are migrated and mature over time. Cost optimization should never end until the cost of identifying money-saving opportunities is more than the amount of money you are actually going to save. Until that point is reached, you should continually monitor your expenditures and look for new ways to save on cost. For example, evaluate when new features are available.

Many details about cost optimization are in the AWS Well-Architected Framework whitepaper: https://d0.awsstatic.com/whitepapers/architecture/AWS_Well-Architected_Framework.pdf

Transparently Attribute Expenditure

The cloud helps to:

- ─ Identify the cost of a system and attribute IT costs to individual business owners.
- ─ Identify return on investment (ROI). Consequently, this gives owners an incentive to optimize their resources and reduce costs.



Transparently attribute expenditure: The cloud makes it easier to identify the cost of a system and attribute IT costs to individual business owners. This helps identify return on investment and, consequently, gives those owners an incentive to optimize their resources and reduce costs.

For instance, you can use tagging on most of your resources. This can help you understand what is costing you each month, whether it's a specific department or a specific project.

Use Managed Services To Reduce Cost Of Ownership

- Managed services remove the operational burden of maintaining servers for tasks such as sending email or managing databases.
- Because managed services operate at cloud scale, they can offer a lower cost per transaction or service.



Use managed services to reduce cost of ownership: In the cloud, managed services remove the operational burden of maintaining servers for tasks such as sending email or managing databases. Additionally, because managed services operate at cloud scale, they can offer a lower cost per transaction or service.

Trade Capital Expense For Operating Expense

- Do not invest heavily in data centers and servers before you know how you're going to use them.
- Pay only for the computing resources you consume, when you consume them.

Example:

Development and test environments are typically only used for eight hours during the working week. Stop these resources when not in use, for a potential cost savings of 75% (40 hours versus 168 hours).



Trade capital expense for operating expense: Instead of investing heavily in data centers and servers before you know how you're going to use them, pay only for the computing resources you consume, when you consume them. For example, development and test environments are typically only used for eight hours per day during the working week, so you can stop these resources when not in use for a potential cost savings of 75% (40 hours versus 168 hours).

Benefit From Economies Of Scale

- Achieve a lower variable cost in the cloud than you could on your own because AWS can achieve **higher economics of scale**.
- Hundreds of thousands of customers are **aggregated** in the AWS cloud, which translates to lower pay-as-you-go prices.



Benefit from economies of scale: By using cloud computing, you may achieve a lower variable cost than you could on your own because AWS can achieve higher economies of scale. Hundreds of thousands of customers are aggregated in the AWS cloud, which translates into lower pay-as-you-go prices. As the number of AWS customers increases, AWS can pass on more economies-of-scale savings .

In addition, as customers use more AWS resources, they can receive higher discounts with tiered pricing that is available for some services. Amazon S3, for example, has a tiered pricing plan with lower cost for larger volumes of stored data.

Stop Spending Money On Data Center Operations

- 💡 AWS does the difficult work of **racking, stacking, and powering** servers.
- 💡 You can **focus on your customers and business projects** instead of on IT infrastructure.



Stop spending money on data center operations: AWS does the difficult work of racking, stacking, and powering servers, so you can focus on your customers and business projects instead of on IT infrastructure.

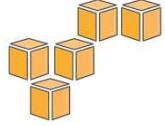
Key Services For Cost Optimization



Areas	Key Services
Matched supply and demand	 Auto Scaling
Cost-effective resources	 Spot and Reserved Instances
Expenditure awareness	 Amazon CloudWatch
Optimizing over time	 AWS Blog & What's New
	 AWS Trusted Advisor
	 Amazon SNS
	 AWS Trusted Advisor

The key AWS feature that supports cost optimization is cost allocation tags, which help you to understand the costs of a system. The following services and features are important in the four areas of cost optimization:

- **Matched supply and demand:** Auto Scaling allows you to add or remove resources to match demand without overspending.
- **Cost-effective resources:** You can use Reserved Instances and prepaid capacity to reduce your cost. AWS Trusted Advisor can be used to inspect your AWS environment and find opportunities to save money.
- **Expenditure awareness:** Amazon CloudWatch alarms and Amazon Simple Notification Service (SNS) notifications will warn you if you go, or are forecasted to go, over your budgeted amount.
- **Optimizing over time:** The AWS Blog and *What's New* section on the AWS website are resources for learning about newly launched features and services. AWS Trusted Advisor inspects your AWS environment and finds opportunities to save money by eliminating unused or idle resources or committing to Reserved Instance capacity.



Part 2

Pricing Options

Optimizing Your Infrastructure Cost

- AWS Free Tier
- Amazon EC2 Pricing Options
- Dedicated Instances and Dedicated Hosts

© 2017 Amazon Web Services, Inc. or its affiliates. All rights reserved.

 Academy

Part 2: Pricing Options - Optimizing Your Infrastructure Cost

AWS Free Tier

- 💡 Get hands-on experience with AWS Cloud Services.
- 💡 Use services for **12 months** following account sign-up for free at certain usage limits.

💡 Other Free Options:

- AWS Free Tier Eligible Software
- AWS Free Tier Non-Expiring Offers

The **Amazon Web Services (AWS) Free Tier** is designed to enable you to get hands-on experience with AWS Cloud Services. The AWS Free Tier includes services with a free tier available for 12 months following your AWS sign-up date.

For example, you can use 750 hours per month of Amazon EC2, or you can use 5 GB of standard Amazon S3 storage with up to 20,000 get requests and 2,000 put requests. Free usage of other services is also available.

AWS Free Tier Non-Expiring Offers: Some services have non-expiring offers that continue after your 12-month term. For example, up to 1,000 Amazon SWF workflow executions are free each month.

AWS Free Tier Eligible Software: The AWS Marketplace offers more than 700 free and paid software products that run on the AWS Free Tier. If you qualify for the AWS Free Tier, you can use these products on an Amazon EC2 t2.micro instance for up to 750 hours per month and pay no additional charges for the Amazon EC2 instance (during the 12 months). Software charges may still apply for paid software, but some software applications are free.

After creating your AWS account, you can use any of the products and services for free within certain usage limits as detailed here: <https://aws.amazon.com/free/>

Amazon EC2 Purchasing Options

- ─  On-Demand Instances
- ─  Spot Instances
- ─  Reserved Instances
- ─  Dedicated Instances
- ─  Dedicated Hosts

As part of the **Free Tier** from AWS, new AWS customers can get started with Amazon EC2 and some other services each month for up to one year after sign-up. What's available for free through the free tier varies from service to service.

On-Demand Instances let you pay for compute capacity by the hour with no long-term commitments or upfront payments.

Spot Instances allow customers to purchase compute capacity with no upfront commitment and at hourly rates usually lower than the On-Demand rate. If the Spot Price moves higher than a customer's maximum price, the customer's instance will be shut down by Amazon EC2.

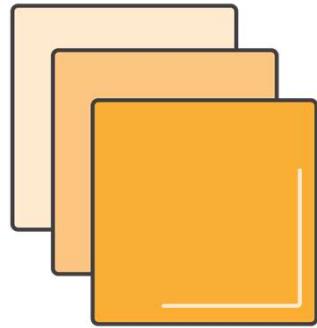
Dedicated Hosts are physical EC2 servers with instance capacity fully dedicated for your use. Dedicated Hosts can help you reduce costs by allowing you to use your existing server-bound software licenses. Dedicated Hosts can be purchased On-Demand or as a Reserved Instance.

Whitepaper: How AWS Pricing Works:

https://d0.awsstatic.com/whitepapers/aws_pricing_overview.pdf

Amazon EC2 On-Demand Instances

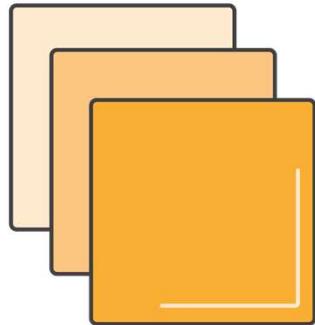
- 💡 Pay for compute capacity by the hour
- 💡 No long-term commitments
- 💡 No upfront payments
- 💡 Increase or decrease your compute capacity depending on the demands of your application



Amazon EC2 On-Demand Instances

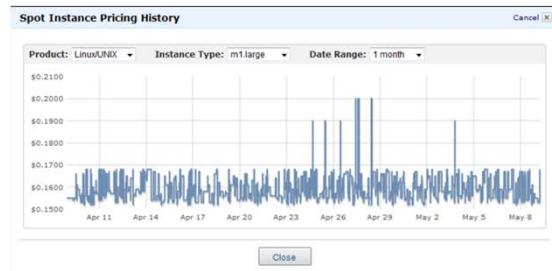
Recommended for:

- Users that want the low cost and flexibility of Amazon EC2 without an up-front payment or long-term commitment
- Applications with short term, spiky, or unpredictable workloads that cannot be interrupted
- Applications being developed or tested on Amazon EC2 for the first time



Amazon EC2 Spot Instances

- Bid for unused AWS capacity
- Prices controlled by AWS based on supply and demand
- Termination Notice provided 2 minutes prior to termination, stored in metadata
- Best approach to temporary requests for large numbers of servers



© 2017 Amazon Web Services, Inc. or its affiliates. All rights reserved.



With Amazon EC2 Spot Instances, you get to name your own prices for the same range of Amazon EC2 instance types that are available in On-Demand Pricing. At the end of the billing hour, you pay the market rate. If the market rate changes and goes above your maximum bid, you lose your compute resource (your instance is terminated).

If your instance is marked for termination, the Termination Notice will be stored in the instance's metadata two minutes before its termination time. The notice is accessible at <http://169.254.169.254/latest/meta-data/spot/termination-time>, and includes the time when the shutdown signal will be sent to the instance's operating system.

Best Practice: Relevant applications on Spot Instances should poll for the termination notice at five-second intervals, which gives the application almost the entire two minutes to complete any needed processing before the instance is terminated and taken back by AWS. More about the Spot Instance Termination Notice here:

<http://aws.amazon.com/blogs/aws/new-ec2-spot-instance-termination-notices/>

So, be sure to architect your infrastructure for interruption:

- Decouple components.
 - Separate interactive processes and back-end processes.
 - Use frameworks such as Elastic Map Reduce.
- Design for interruption.
 - Use Amazon SQS or Amazon SWF.
 - Place data in durable stores such as Amazon S3 or DynamoDB.
 - Save progress regularly.

Spot Use Cases

Use Case	Types of Applications
Batch processing	Generic background processing (scale out computing)
Web/data crawling	Analyze data
Financial	Hedge fund analytics, energy trading, etc.
Elastic Map Reduce	Hadoop (large data processing)
Grid computing	Scientific trials/simulations in chemistry, physics, and biology
Transcoding	Transform videos into specific formats
Gaming	Back-end servers for Facebook games
Testing	Scale to large server pool to test software, websites, etc.

Although Spot Instances are terminated when the market price exceeds your bid price, there are still many different kinds of workloads that Spot is good for.

Vimeo's Spot Market Considerations

Key Considerations:

- Never bid more than threshold (80% of on-demand price)
- No more than 10 open spot requests at any time
- Bid 10% more than the average price over last hour
- Use spots for low-priority and less time-critical jobs
- Have more retries for jobs running on spots
- Watch out for open spot requests (add expiry to your requests)
- Billed hour forward unless terminated by AWS
- For *long running jobs*, either bid higher on spot or use on-demand instances
- Fail over to on-demand when spot market is saturated

Vimeo.com is a video sharing service that uses Amazon EC2 instances to transcode videos after they've been uploaded. Using Spot instances has saved them up to 50% over On-Demand instances. They've shared how they achieved these savings in a slide deck you can find here: <http://www.slideshare.net/ptrmcrrhr/vimeo-ec2>

This slide details the key considerations they had to make when planning to replace On-Demand instances with Spot instances.

For more information on how Vimeo used Spot instances, see:
<http://www.slideshare.net/ptrmcrrhr/vimeo-ec2>

Amazon EC2 Reserved Instance Types

No Upfront

- Access a Reserved Instance without an upfront payment
- Discounted effective hourly rate for every hour within the term, regardless of usage
- 1-year reservation available

Up to 75%
discount
compared to
On-Demand
Instance
pricing

Partial Upfront

- Part of the Reserved Instance must be paid at the start of the term
- Discounted effective hourly rate for the remainder of the term, regardless of usage
- 1-year or 3-year reservations available

All Upfront

- Full payment made at the start of the term
- No other costs incurred for the remainder of the term, regardless of usage
- 1-year or 3-year reservations available

This slide presents a more detailed look at **Reserved Instances**, which can provide you with a significant discount (up to 75%) compared to On-Demand Instance pricing. You are assured that your Reserved Instance will always be available for the operating system and Availability Zone in which you purchased it. For applications that have steady-state or predictable usage, Reserved Instances can provide significant savings compared to using On-Demand Instances.

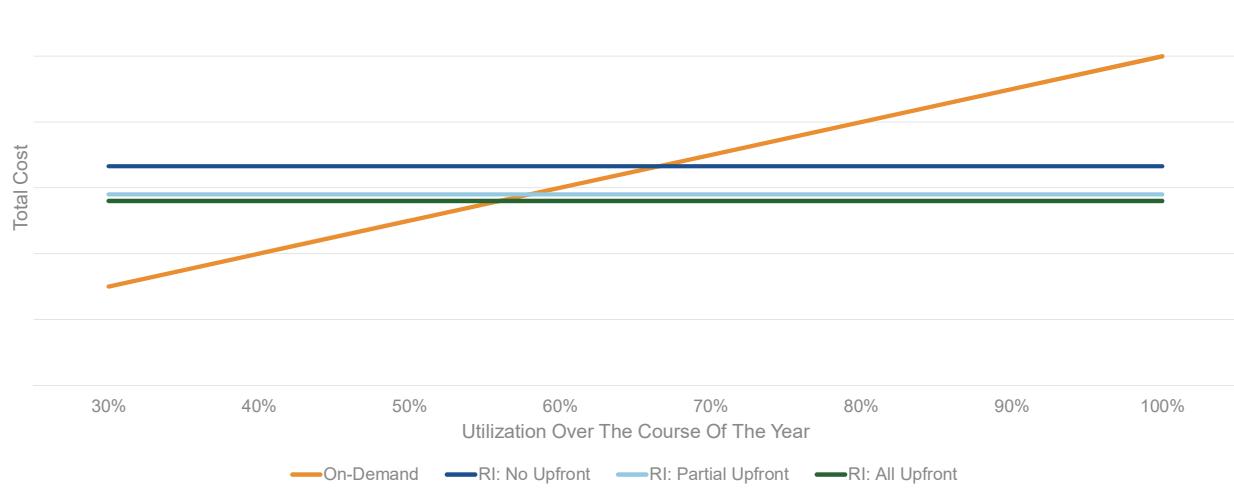
There are three payment options with Reserved Instances:

- **All Upfront:** Pay for the entire Reserved Instance with one upfront payment.
- **Partial Upfront:** Make a low upfront payment, and then you are charged a discounted hourly rate for the instance for the duration of the Reserved Instance term.
- **No Upfront:** Does not require any upfront payments and provides a discounted hourly rate for the duration of the term.

When you purchase Reserved Instances, note that you are required to specify instance type, operating system, and Availability Zone. In addition, you are billed for every hour for the term, even when the instance is not in use. For situations that require only limited use of instances, we recommend using On Demand or Spot Instances instead.

On-Demand vs. Reserved Instance Break-Even Points

m3.xlarge 1 year On-Demand vs. Reserved Instances



© 2017 Amazon Web Services, Inc. or its affiliates. All rights reserved.



So what are the break-even points of these various RI purchasing options compared to purchasing instances On-Demand? Let's look at the following scenario: You decide you need an m3.xlarge instance running Linux (in US-East) and are wondering at what level of usage (the number of hours in a month you will run that instance). Does it make sense to purchase an RI compared to running that instance on an On-Demand basis? In this scenario, you will begin saving money over On-Demand by purchasing an All Upfront Reserved Instance if you are running the instance at least 60% of the time. If you are not in a position to purchase All Upfront, consider the Partial Upfront or No Upfront options where the break even points are at ~65% usage and ~72% usage, respectively.

In other words, if you are running the instance at least 65% of the time, and cannot make an All Upfront purchase, you will save money over the year by purchasing a partial up-front Reserved Instance.

If you would rather make no up-front payments at all, but can commit to running this instance at least 72% of the time for at least a year, the No Upfront Reserved Instance payment option allows you to pay a low monthly fee to take advantage of savings over running that same instance as On-Demand.

Reserved Instance Marketplace Beta

Flexibility

- Sell your unused Amazon EC2 Reserved Instances
- Buy Amazon EC2 Reserved Instances from other AWS customers
- As your needs change, change your Reserved Instances

Diverse term and pricing options

- Shorter terms
- Opportunity to save on upfront pricing

Identical capacity reservations

Along with the implementation of the new Reserved Instance pricing model, AWS has released the Reserved Instance Marketplace, currently in beta.

Flexibility: You can sell your unused Amazon EC2 Reserved Instances to other businesses and organizations. You can also buy Amazon EC2 Reserved Instances from other AWS customers (listed as third-party sellers). Third-party sellers on the Marketplace provide a wide selection of term lengths and pricing options to choose from. Throughout the course of your term, your needs may change. You may need more or fewer instances than anticipated, or you may just need to move an instance to a new AWS Region or change other options for your instance, such as instance type.

Diverse term and pricing options: Buying Reserved Instances from the marketplace provides a wider selection of prices and terms than buying Reserved Instances directly from AWS. When purchasing a Reserved Instance from the marketplace, you will be taking over the third-party seller's original one- or three-year term. As such, you will only be required to fulfill the remainder of the term from the point at which you purchased the Reserved Instance from a third-party seller. In addition, the up-front cost is determined by the third-party seller, providing you with the opportunity to spend less on your upfront costs compared with the price of a Reserved Instance purchased directly from AWS. However, usage or recurring fees, such as monthly fees, will remain the same as the fees set when the Reserved Instances were originally purchased from AWS.

Identical capacity reservations: Reserved Instances from the marketplace offer the same capacity reservations as those purchased directly from AWS.

Reserved Instances: Scheduled Instances

Reserve instances for **predefined blocks of time on a recurring basis** for a one-year term, with prices that are generally 5 to 10% lower than the equivalent On-Demand rates.

- 💡 Run reserved instances on a daily, weekly, or monthly basis.
- 💡 After you purchase your reserved instances, they are available to launch during the time windows that you've specified.
- 💡 Can be launched via the AWS Management Console, AWS CLI, AWS Tools for Windows PowerShell, and the `RunScheduledInstances` function.

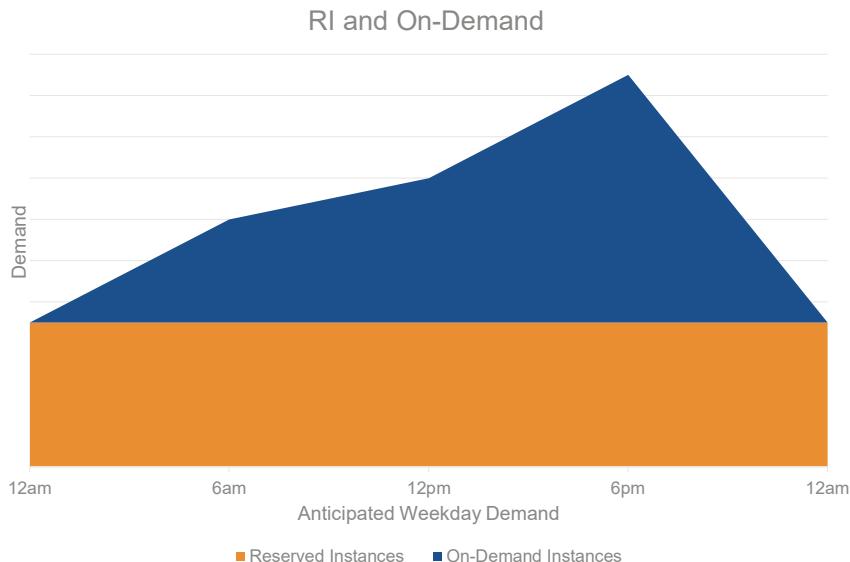
Scheduled Reserve Instance pricing is available if you purchase Reserved Instances on a recurring schedule, which allows you to pay for compute capacity by the hour and obtain a capacity reservation ahead of time for only the time periods you will need it.

Reserved Instances: Scheduled Instances

Use cases:

- ⌚ A bank performs Value at Risk calculations every weekday afternoon
- ⌚ A phone company does a multi-day bill calculation run at the start of each month
- ⌚ A trucking company optimizes routes and shipments on Monday, Wednesday, and Friday mornings
- ⌚ An animation studio performs a detailed, compute-intensive 3D rendering every night

Leveraging EC2's Different Pricing Models

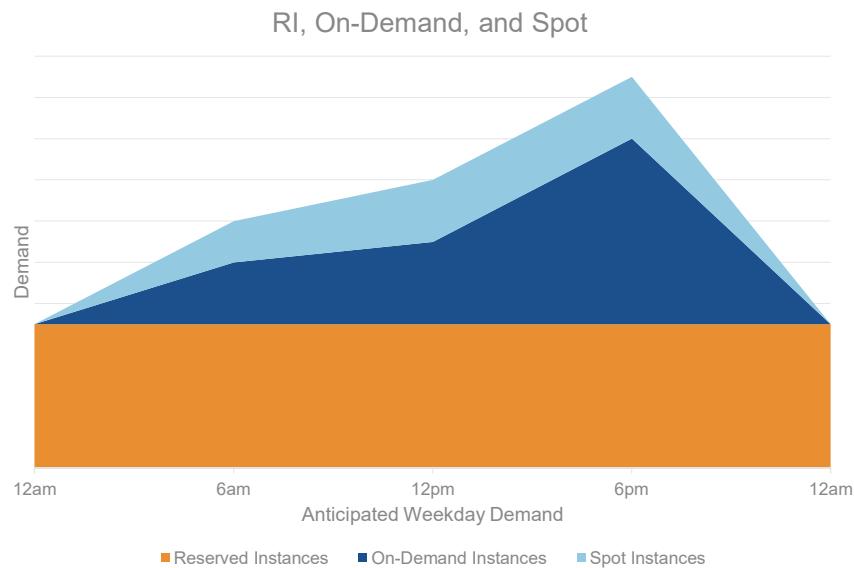


© 2017 Amazon Web Services, Inc. or its affiliates. All rights reserved.



This chart illustrates a very basic way to leverage Reserved Instances and On-Demand Instances together to accommodate for fluctuations in demand over time. In this example, multiple Reserved Instances have been purchased and are running, but as the day goes on and demand increases, the need for more instances increases as well. This customer supplements their capacity with On-Demand Instances, which can be shut down when they're not needed later in the evening.

Leveraging EC2's Different Pricing Models



© 2017 Amazon Web Services, Inc. or its affiliates. All rights reserved.



In this second chart, this customer takes a more complex approach which attempts to leverage all three price models. In this case, some of the supplemental instances come first as On-Demand, but further needs are addressed with Spot Instances. This allows them to save money by using Spot over On-Demand; however it exposes them to unexpected instance termination, because they would lose those instances if they're outbid. This could lead to lost data or insufficient capacity for their customers, which means a model such as this should only be implemented in circumstances where sudden termination of instances is acceptable and handled appropriately.

One AWS customer that has leveraged all three models together is Pinterest. For more information, see <http://www.allthingsdistributed.com/2012/08/tco-and-return-on-agility.html>

Blended Approach

- 💡 Choose instance type that matches requirements
 - Start with memory requirements and architecture type (32-bit or 64-bit)
 - Then choose the closest number of virtual cores required
- 💡 Scale across Availability Zones
 - Smaller increments give more granularity for deploying to multiple AZs
- 💡 Start with on-demand and then assess utilization for RIs

If you know the baseline utilization, using Reserved Instances can save you money over time because they offer discounted (effective) hourly rates. At seasonal peaks, you can use On-Demand Instances to handle the increased load.

Dedicated Instances and Dedicated Hosts

Comparing dedicated instances and dedicated hosts:

Characteristic	Dedicated Instances	Dedicated Hosts
Dedicated physical servers	✓	✓
Per instance billing (subject to a \$2 per region fee)	✓	✗
Per host billing	✗	✓
Visibility of sockets, cores, host ID	✗	✓
Affinity between a host and instance	✗	✓
Targeted instance placement	✗	✓
Automatic instance placement	✓	✓
Add capacity using an allocation request	✗	✓

Dedicated Instances

- Single customer tenancy on compute hardware.
- Physically isolated at the host hardware level:
 - From other instances that are not dedicated instances.
 - From instances that belong to other AWS accounts.
- Useful for workloads with **unique** compliance requirements.
 - Example: HIPAA PHI
- May not be needed unless physical isolation is required.

AWS provides various options for provisioning your EC2 instances such as On-Demand, Reserved, Spot, and Dedicated Instances. Of these, the Dedicated Instance option is very useful for customers with unique compliance requirements (like HIPAA PHI).

Dedicated Instances are Amazon EC2 instances that run in a virtual private cloud (VPC) on hardware that is dedicated to a single customer for additional isolation. These instances are physically isolated at the host hardware level from instances that are not dedicated instances and from instances that belong to other AWS accounts.

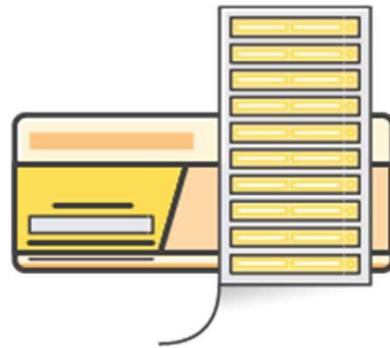
For more details on architecting for HIPAA security and compliance on AWS, see https://d0.awsstatic.com/whitepapers/compliance/AWS_HIPAA_Compliance_Whitepaper.pdf.

Dedicated Hosts

A dedicated host is a physical EC2 server with instance capacity **fully dedicated** for your use.

Dedicated hosts can help you:

- Use your existing server-bound licenses
- Meet compliance and regulatory requirements



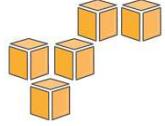
A Dedicated Host is a physical EC2 server with instance capacity fully dedicated for your use. Dedicated Hosts can help you reduce costs by allowing you to use your existing server-bound software licenses, including Windows Server, SQL Server, and SUSE Linux Enterprise Server (subject to your license terms), and can also help you meet compliance requirements. Dedicated Hosts can be purchased On-Demand (hourly). Reservations can provide up to a 70% discount compared to the On-Demand price.

Dedicated Host benefits:

- **Save Money on Licensing Costs:** Dedicated Hosts can enable you to save money by using your own per-socket or per-core software licenses in Amazon EC2.
- **Help Meet Compliance and Regulatory Requirements:** Dedicated Hosts allow you to place your instances in a VPC on a specific, physical server. This enables you to deploy instances using configurations that help address corporate compliance and regulatory requirements

For more information about Dedicated Hosts, see

<https://aws.amazon.com/ec2/dedicated-hosts/>.



Section 3

Cost Management Tools

Optimizing Your Infrastructure Cost

- AWS Trusted Advisor
- Optimizing Cost with Caching
- AWS Cost Calculation Tools

Part 3: Cost Management Tools - [Optimizing Your Infrastructure Cost](#)

Using AWS Trusted Advisor

AWS Trusted Advisor provides best practices (checks) in:

- Cost Optimization
- Security
- Fault Tolerance
- Performance Improvement

Best practices available to all customers:

- Service Limits
- Security Groups – Specific Ports Unrestricted
- IAM Use
- MFA on Root Account

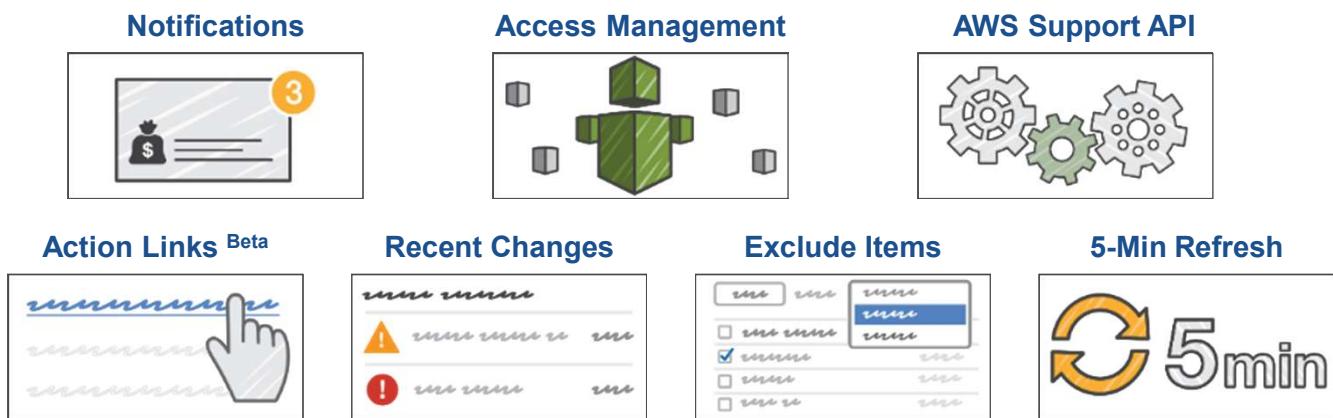
Over 40 Trusted Advisor checks available with Business and Enterprise Support plans

AWS Trusted Advisor is like your customized cloud expert. It provides four of the most popular performance and security recommendations to all AWS customers. The complete set of checks and guidance is available with Business and Enterprise Support plans. AWS Trusted Advisor helps you to provision your resources following best practices to improve system performance and reliability, increase security, and look for opportunities to save money.

You can see the comprehensive list of Trusted Advisor best practices (checks) here:
<https://aws.amazon.com/premiumsupport/trustedadvisor/best-practices/>.

Trusted Advisor Features And Functionalities

AWS Trusted Advisor provides a suite of features for you to customize recommendations and to proactively monitor your AWS resources:



© 2017 Amazon Web Services, Inc. or its affiliates. All rights reserved.

 Academy

Trusted Advisor Notifications helps you stay up-to-date with your AWS resource deployment. You will be notified by weekly email when you opt in for this service, and it is free.

You can use **AWS Identity and Access Management (IAM)** to control access to specific checks or check categories.

You can retrieve and refresh Trusted Advisor results programmatically using the **AWS Support API**.

Action Links are hyperlinks on items within a Trusted Advisor report that take you directly to the AWS Management Console where you can take action on the Trusted Advisor recommendations.

With the **Recent Changes** feature, you can track recent changes of check status on the console dashboard. The most recent changes appear at the top of the list to bring them to your attention.

The **Exclude Items** feature allows you to customize the Trusted Advisor report. You can exclude items from the check result if they are not relevant.

You can refresh individual checks or refresh all the checks at once by clicking the Refresh All button in the summary dashboard. A check is eligible for **refresh five minutes** after it was last refreshed.

For more information about Trusted Advisor, see <https://aws.amazon.com/premiumsupport/trustedadvisor/>.

AWS Trusted Advisor Dashboard

The screenshot shows the AWS Trusted Advisor Dashboard. On the left, a sidebar lists 'Dashboard', 'Cost Optimizing', 'Performance', 'Security', 'Fault Tolerance', and 'Preferences'. The main area is titled 'Trusted Advisor Dashboard' and contains four sections: 'Cost Optimizing' (with a dollar sign icon), 'Performance' (with a speedometer icon), 'Security' (with a lock icon), and 'Fault Tolerance' (with an umbrella icon). Each section displays a summary of checked (green checkmark), warning (yellow triangle), and error (red exclamation) items, along with potential monthly savings. Below these are 'Recent Changes' and 'What's New' sections, and a language selection dropdown set to English.

© 2017 Amazon Web Services, Inc. or its affiliates. All rights reserved.



Academy

This is an example of the Trusted Advisor Dashboard. It shows a summary of the four best practice categories as well as recent changes.

You can visit the Trusted Advisor Console here:

<https://console.aws.amazon.com/trustedadvisor/>

AWS Trusted Advisor Case Study – Hungama

“

Using AWS Trusted Advisor helped us save 33% on our monthly bill, and we'll continue to use it to optimize our infrastructure and costs on AWS.

Amit Vora
CTO, Hungama Digital Media



”

Hungama uses AWS Trusted Advisor to Optimize Usage and Cut Costs

- Hungama has used AWS for server and storage management since 2008.
- They deliver content to consumers in 47 countries across mobile, Internet, and Internet protocol television (IPTV) services.
- The company uses Amazon S3 to host more than 60 TB of content and Amazon EC2 and Amazon RDS for server and storage management.
- As the company grew rapidly, more departments used AWS for development, causing an increase in monthly costs.

Hungama is a leading aggregator, developer, publisher and distributor of Bollywood and South-Asian entertainment content.

AWS Trusted Advisor Case Study – Hungama

Optimize Usage and Cut Costs

Hungama reduced monthly costs by 33% by using AWS Trusted Advisor.

How did they do that?

By using Trusted Advisor's Cost Optimizing checks.

Here are a few things they did:



The Low Utilization Amazon EC2 Instances check revealed over-provisioned instance sizes, and instances spun up for special projects were not terminated after completion.

In response, the audit team used this information to right-size their instances. They also categorized production and development servers and automated the process of shutting down development servers during non-business hours.

The Low Utilization Amazon EC2 Instances check on AWS Trusted Advisor checks the Amazon Elastic Compute Cloud (Amazon EC2) instances that were running at any time during the last 14 days and alerts you if the daily CPU utilization was 10% or less and network I/O was 5 MB or less on 4 or more days.

Estimated monthly savings are calculated by using the current usage rate for On-Demand Instances and the estimated number of days the instance might be underutilized. Actual savings will vary if you are using Reserved Instances or Spot Instances, or if the instance is not running for a full day. To get daily utilization data, download the report for this check.

AWS Trusted Advisor Case Study – Hungama

Optimize Usage and Cut Costs

Hungama reduced monthly costs by 33% by using AWS Trusted Advisor.

How did they do that?

By using Trusted Advisor's Cost Optimizing checks.

Here are a few things they did:



The Reserved Instance Optimization check identified additional opportunities for optimization of the RI instances they had purchased.

In response, Hungama changed how they reserved their instances and based reservations on the specific usage patterns of their different instance categories (dev/prod/test/etc).

Hungama's RI changes occurred prior to the implementation of the current pricing model (No/Partial/All Upfront), however this check is still extremely useful to customers using RI instances.

The Reserved Instance Optimization check with AWS Trusted Advisor checks your Amazon EC2 computing consumption history and calculates an optimal number of Partial Upfront Reserved Instances. Recommendations are based on the previous calendar month's hour-by-hour usage aggregated across all consolidated billing accounts.

AWS Trusted Advisor Case Study – Hungama

Optimize Usage and Cut Costs

Hungama reduced monthly costs by 33% by using AWS Trusted Advisor.

How did they do that?

By using Trusted Advisor's Cost Optimizing checks.

Here are a few things they did:



Amazon EBS

The Underutilized Amazon EBS volumes check identified a number of unused or underutilized EBS volumes that were often leftover from previous test projects.

In response, the audit team created snapshots of many of the underutilized EBS volumes, which they stored on Amazon S3, and then deleted the volumes. This resulted in a reduction of over 90% on the number of snapshots generated weekly.

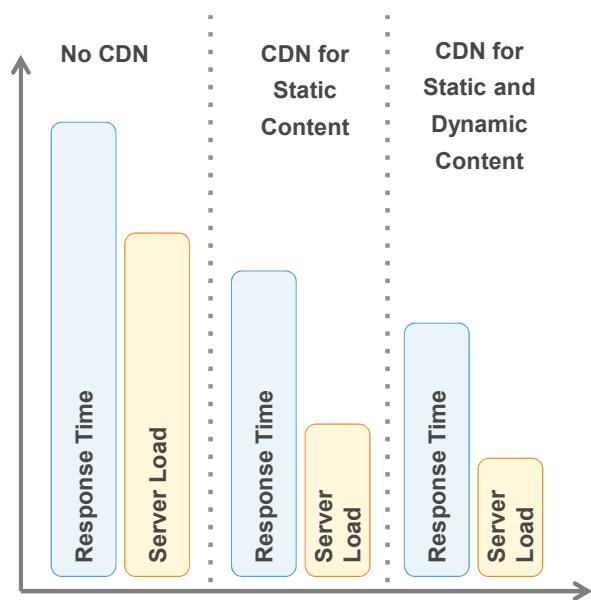
The Underutilized Amazon EBS volumes check on AWS Trusted Advisor checks Amazon EBS volume configurations and warns when volumes appear to be underused. If a volume remains unattached or has very low write activity (excluding boot volumes) for a period of time, the volume is probably not being used.

For more on how Hungama uses AWS, see: <https://aws.amazon.com/solutions/case-studies/hungama/>

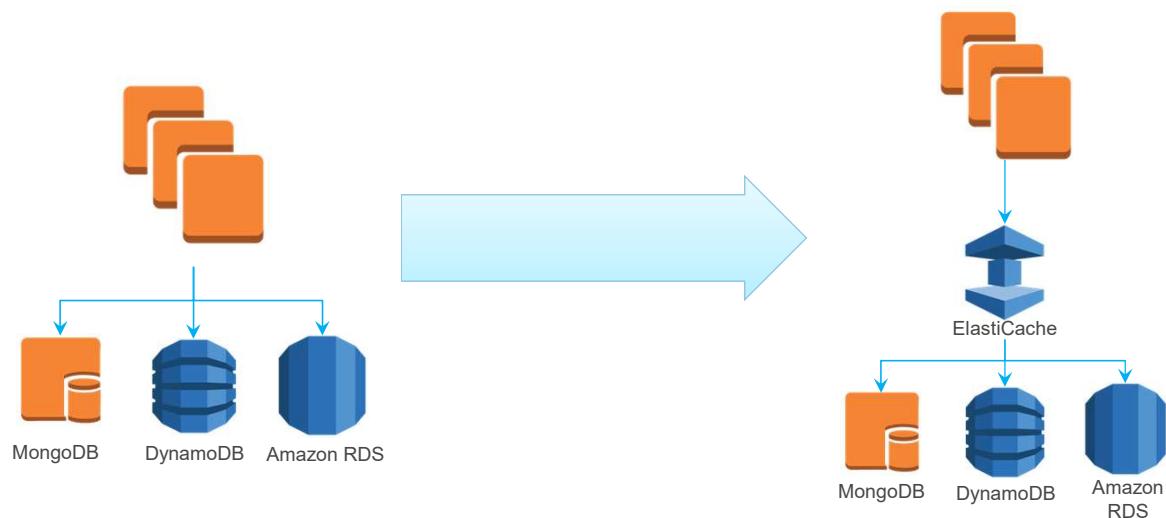
Optimize Cost with Caching

The more you can offload, the less infrastructure you need to maintain, scale, and pay for.

- 💡 Offload popular traffic to Amazon CloudFront and Amazon S3
- 💡 Introduce caching



Cache Everything



AWS Simple Monthly Calculator

Estimate the cost of running your application or solution in the AWS cloud based on usage:

Language: English ▾

Get Started with AWS: [Learn more about our Free Tier](#) or [Sign Up for an AWS Account](#)

FREE USAGE TIER: New Customers get free usage tier for first 12 months

Services		Estimate of your Monthly Bill (\$ 215.67)					
Choose region: US-East / US Standard (Virginia) ▾		Inbound Data Transfer is Free and Outbound Data Transfer is 1 GB free per region per month					
Amazon Elastic Compute Cloud (Amazon EC2) is a web service that provides resizable compute capacity in the cloud. It is designed to make web-scale computing easier for developers. Amazon Elastic Block Store (EBS) provides persistent storage to Amazon EC2 instances.		Clear Form					
Compute: Amazon EC2 Instances:		Description	Instances	Usage	Type	Billing Option	Monthly Cost
		Add New Row					
Storage: Amazon EBS Volumes:		Description	Volumes	Volume Type	Storage	IOPS	Snapshot Storage
		Add New Row					

© 2017 Amazon Web Services, Inc. or its affiliates. All rights reserved.



Academy

The AWS Simple Monthly Calculator is a tool that can help you estimate the cost of running your application or solution in the AWS cloud based on usage.

The AWS Simple Monthly Calculator is here:

<https://calculator.s3.amazonaws.com/index.html>

Video: Getting Started with the AWS Simple Monthly Calculator:

<https://youtu.be/54TVCueOoAc?list=PLhr1KZpdzukcAtqFF32cjGUNNT5GOzKQ8>

AWS Total Cost Of Ownership (TCO) Calculator

- ─ Estimate cost savings when using AWS
- ─ Use a detailed set of reports that can be used in executive presentations
- ─ Modify assumptions that best meet your business needs

1. Describe your infrastructure in four steps, or enter detailed configurations

What type of environment are you comparing against? On-Premises Colocation

Which AWS region is ideal for your geo requirements? US East (N. Virginia)

Servers

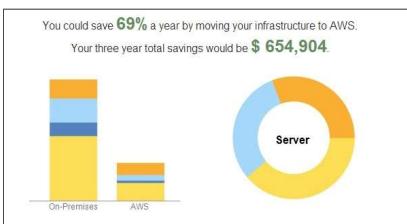
Are you comparing physical servers or virtual machines? Physical Servers Virtual Machines

App. Name	Number of VMs	CPU Cores	Memory(GB)	Hypervisor	Guest OS	VM Usage (%)
100	4	16	Vmware	Linux	100	

Storage

Storage Type	RAV Storage Capacity	Max IOPS for Application	Backup % / Month
SAN	40 TB	1000	10

2. Get an instant summary report



3. Download a full report including detailed cost breakdowns

amazon web services

Contact Sales

Calculations

Methodology

Assumptions

FAQ

You can use the below URL to retrieve your calculations or Share it with the world:
<https://www.awstcocalculator.com/Output/LoadOff2cc0130443427588768972>

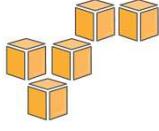
© 2017 Amazon Web Services, Inc. or its affiliates. All rights reserved.



AWS helps you reduce Total Cost of Ownership (TCO) by reducing the need to invest in large capital expenditures and providing a pay-as-you-go model that empowers you to invest in the capacity you need and use it only when the business requires it.

The AWS TCO calculators allow you to estimate the cost savings when using AWS and provide a detailed set of reports that can be used in executive presentations. The calculators also give you the option to modify assumptions that best meet your business needs.

You can launch the TCO Calculator here: <https://awstcocalculator.com/>



In review...

- 💡 Cost Optimization Pillar overview
- 💡 Key services for optimizing cost
- 💡 Pricing options and instance types
- 💡 Cost management tools



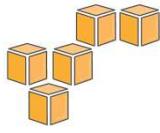
Knowledge Assessment

© 2017 Amazon Web Services, Inc. or its affiliates. All rights reserved.

In review...

- Cost Optimization Pillar overview
- Key services for optimizing cost
- Pricing options and instance types
- Cost management tools

To complete this module, please remember to finish the corresponding knowledge assessment.



Up Next...



[CCA 3.12 Worksheet - Improve This Architecture Exercise](#)



[LAB 14 - Launching EC2 Spot Instances with Auto Scaling and CloudWatch](#)



[Unit 3 – Section 4: Deployment and Implementation](#)

Before you continue, please complete the exercise in the [CCA 3.12 Worksheet](#) and [Lab 14](#), Launching EC2 Spot Instances with Auto Scaling and CloudWatch.

Up next is **Section 4** featuring the **Deployment and Implementation**.

© 2017 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.

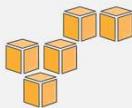
Errors or corrections? Email us at aws-course-feedback@amazon.com.

For all other questions, contact us at
<https://aws.amazon.com/contact-us/aws-training/>.

All trademarks are the property of their owners.

© 2017 Amazon Web Services, Inc. or its affiliates. All rights reserved.





CCA Unit 3 – Architecting on AWS

CCA 3.13: Troubleshooting

- Section 1: Introduction to System Design
 - Section 2: Automation and Serverless Architectures
 - Section 3: Well-Architected Best Practices
 - Section 4: Deployment and Implementation
- **CCA 3.13 Troubleshooting**
CCA 3.14 Design Patterns and Sample Architectures

© 2017 Amazon Web Services, Inc. or its affiliates. All rights reserved.



This module begins **Section 4 of Unit 3 – Deployment and Implementation**. We'll start by covering approaches for troubleshooting (CCA 3.13) followed by Design Patterns and Sample Architectures.

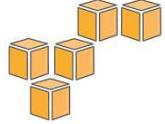
What's In This Module?

- Instance connection
- Network performance
- Storage I/O rates to EBS
- CPU load on RDS instances
- Permissions

This module covers common troubleshooting issues including...

- Instance connections timing out
- [2] network performance,
- [3] storage I/O rates to EBS
- [4] CPU load on RDS instances, and
- [5] permissions issues.

NOTE: At the end of this module you will download a **worksheet** to complete an exercise for the Performance Efficiency pillar.



Part 1

“My instance connection timed out.”

Part 1: “My instance connection timed out.”

"My instance connection timed out."

*Check your **routes**.*

Is your **routing table** configured correctly?

Public subnets need Internet-bound traffic routed to an IGW.

Destination	Target
10.0.0.0/16	local
0.0.0.0/0	IGW

If you're using a Virtual Private Gateway, is your **VPN** routed correctly?

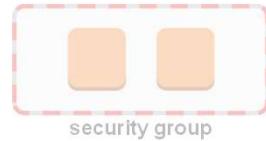
Make sure you're using the correct routing type for your VPN router (Dynamic or Static).



"My instance connection timed out."

*Check your **security group rules**.*

Instances need both **inbound** and **outbound** rules permitting traffic.
Without rules, security groups deny all traffic by default.



But don't leave your security group **completely open**.
Publicly-accessible hosts should be behind another layer of access that offers control. (DNS, WAF, ELB)



"My instance connection timed out."

Check your **Network ACLs**.

Verify that your Network ACLs allow traffic to and from your computer.

If applicable, check your corporate network's **internal firewall**.

Port 22 for Linux instances and port 3389 for Windows instances must be open.

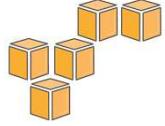
"My instance connection timed out."

Make sure your instance has a **public IP address**.

If you forgot to give it one, you can attach an Elastic IP address to it without having to restart your instance.

Check the **CPU load** on your instance – it may be overloaded.

Use Amazon CloudWatch to check CPU utilization. If your instance is overloaded, consider scaling up to a larger instance type, or scaling out to more instances running in parallel behind a load balancer.



Part 2

“My network performance is poor.”

Part 2: “My network performance is poor.”

"My network performance is poor."

Consider changing your **instance type**.

Are you using an instance type with **enhanced networking**?

Enhanced networking provides higher performance (packets per second), lower latency, and lower jitter.



Amazon
EC2

If you're using a **NAT instance** on Amazon EC2, does it need to be scaled up to a larger size?

AWS NAT Gateways are also better equipped for handling high network throughput needs.



NAT gateway

"My network performance is poor."

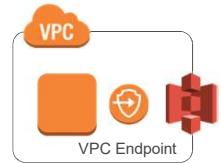
- 💡 If used, make sure **jumbo frames** are enabled correctly.

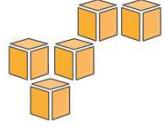
If one instance has jumbo frames enabled, make sure all instances (including NAT instances) it communicates with also have jumbo frames enabled.



- 💡 Consider **VPC endpoints** where possible.

Connections between VPC resources and Amazon S3 will likely be faster if they use an Amazon S3 VPC endpoint instead of traversing the Internet.





Part 3

"My I/O to my EBS volumes is too low."

Part 3: "My I/O to my EBS volumes is too low."

"My I/O to my EBS volumes is too low."

Review your **instance** and **EBS types**:

- 💡 Are you using **EBS-optimized** instance types?

EBS-optimized instance types are designed for applications with heavy disk I/O.



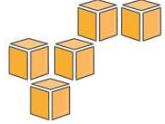
Amazon
EC2

- 💡 Are you using an EBS type with high I/O?

Provisioned IOPS SSDs can provision up to 20,000 IOPS per volume.



Amazon EBS



Section 4

"The CPU load on my RDS instances is too high."

Part 4: 2: "The CPU load on my RDS instances is too high."

"The CPU load on my RDS instances is too high."

💡 Optimize your queries.

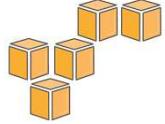
Determine which of your queries are slowest, and review them to determine if they can be optimized.

💡 Use read replicas.

Letting a read replica handle read requests will relieve the CPU load on your master RDS instance.

💡 Ensure you're using the best instance type.

Your queries may require more CPU or memory; test your queries against more powerful instance types to see if you should switch.



Part 5

"I get *access denied* when I make a request to an AWS service."

Part 5: "I get *access denied* when I make a request to an AWS service."

"I get 'access denied' when I make a request to an AWS service."

- 💡 Verify you have **permission** to call that action on that resource.

If any conditions are set, you must meet those conditions as well.

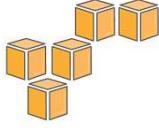


permissions

- 💡 Verify that any **resource policies** specify you as a principal and grant you access.

Services like Amazon S3, Amazon SNS, and Amazon SQS have resource-based policies.





In review...

- Instance connection
- Network performance
- Storage I/O rates to EBS
- CPU load on RDS instances
- Permissions



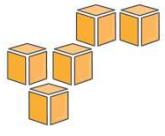
Knowledge Assessment

© 2017 Amazon Web Services, Inc. or its affiliates. All rights reserved.

In review...

- Instance connection
- Network performance
- Storage I/O rates to EBS
- CPU load on RDS instances
- Permissions

To complete this module, please remember to finish the corresponding knowledge assessment.



Up Next...



CCA 3.13 Worksheet - Improve This Architecture Exercise



CCA 3.14: Design Patterns and Sample Architectures

© 2017 Amazon Web Services, Inc. or its affiliates. All rights reserved.



Before you continue, please complete the exercise in the **CCA 3.13 Worksheet**.

Up next is **CCA 3.14** featuring the **Design Patterns and Sample Architectures**.

© 2017 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.

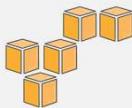
Errors or corrections? Email us at aws-course-feedback@amazon.com.

For all other questions, contact us at
<https://aws.amazon.com/contact-us/aws-training/>.

All trademarks are the property of their owners.

© 2017 Amazon Web Services, Inc. or its affiliates. All rights reserved.





CCA Unit 3 – Architecting on AWS

CCA 3.14: Design Patterns and Sample Architectures

Section 1: Introduction to System Design

Section 2: Automation and Serverless Architectures

Section 3: Well-Architected Best Practices

Section 4: Deployment and Implementation

CCA 3.13 Troubleshooting

► **CCA 3.14** Design Patterns and Sample Architectures

© 2017 Amazon Web Services, Inc. or its affiliates. All rights reserved.



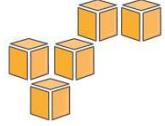
This module begins **Section 4** of Unit 3 – **Deployment and Implementation**. We'll start by covering approaches for troubleshooting (CCA 3.13) followed by Design Patterns and Sample Architectures.

What's In This Module?

- High Availability Design Patterns
- Stream Processing Example
- Sensor Network Data Ingestion and Processing Example
- Application Back-End Example
- Transcoding and Serving Video Files

This module covers common troubleshooting issues including...

- High Availability Design Patterns
- Stream Processing Example
- Sensor Network Data Ingestion and Processing Example
- Application Back-End Example
- Transcoding and Serving Video Files



Part 1

High Availability Design Patterns

© 2017 Amazon Web Services, Inc. or its affiliates. All rights reserved.

 **amazon**
webservices | Academy

Part 1: High Availability Design Patterns

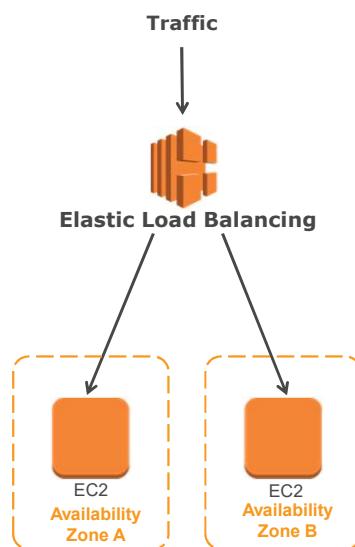
Review: Multi-AZ Pattern

Pros:

- If an Availability Zone fails, the system is still available as a whole.

Implementation:

- Create an AMI for your instance.
- Spin up multiple instances using that AMI in multiple AZs.
- Create a load balancer in multiple AZs and attach the instances.
- Confirm instances are attached to load balancer and are in a healthy state.



- Need to store static objects such as uploads in a common location such as Amazon S3 or NFS so all EC2 instances have access to these resources
- Need to use sticky sessions with Elastic Load Balancing or move session state out of web server

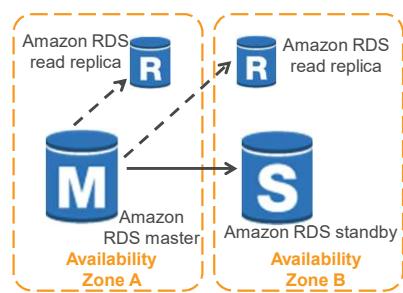
Review: High Availability Database Pattern

Pros:

- One connection string for master and slave with automatic failover
- Maintenance does not bring down DB but causes failover
- Read replicas take load off of master

Implementation:

- Create an Amazon RDS instance (MySQL or Oracle)
- Deploy in Multi-AZ
- Create read replicas for each zone (MySQL or Aurora)



Review: Floating IP Pattern

Problem: Your instance fails or you need to upgrade it, so you need to push traffic to another instance with the same public IP address.

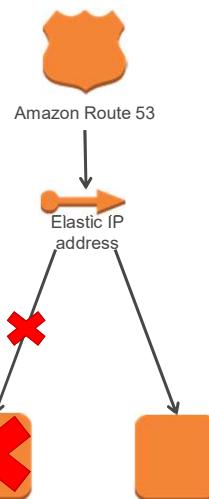
Solution: Use an **Elastic IP** address.

Pros:

- Since we are moving the EIP, DNS will not need to be updated.
- Fallback is as easy as moving the EIP back to the original instance.
- Elastic IP addresses can be moved across instances in different zones in the same region.

Implementation:

- Allocate the Elastic IP address for the EC2 instance.
- Upon failure or upgrade, launch a new EC2 instance.
- Disassociate the Elastic IP address from the EC2 instance and associate it to the new EC2 instance.



- Moving the Elastic IP address between instances can be automated.
- The EC2 Auto Recovery feature can be used with certain instance types.

Floating Interface Pattern

Problem: In case an instance fails or needs to be upgraded, I need to push traffic to another instance with the **same public and private IP addresses** and **same network interface**.

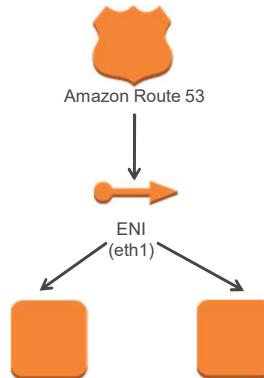
Solution: Deploy your application in VPC and **use an elastic network interface (ENI)** on eth1 that can be **moved** between instances.

Advantages:

- DNS will not need to be updated.
- Easy rollback: move the ENI back to the original instance.
- Anything pointing to the public or private IP on the instance will not need to be updated.
- ENIs can be moved across instances in a subnet.

Implementation:

- Allocate the ENI for the instance.
- Upon failure or upgrade, launch a new instance.
- Detach the ENI from the instance and attach it to the new instance.



Moving the ENI between instances can be automated. Additionally, the ENIs cannot be moved, but all other ENIs can.

State-Sharing

Problem: You want your app to be **stateless** in order to better scale horizontally.

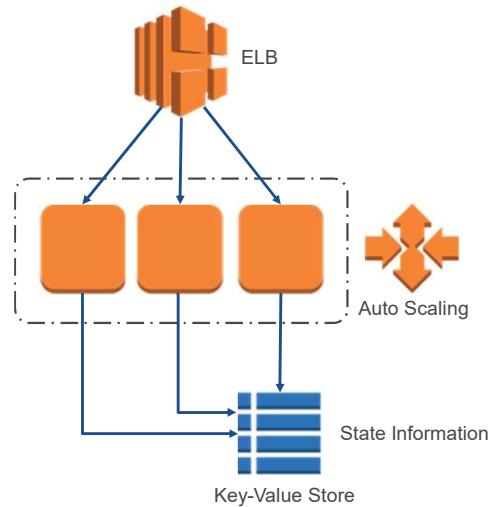
Solution: Move state off of your server and into a **key-value store (KVS)**.

Advantages:

- This lets you use the scale-out pattern without having to worry about inheritance or loss of state information.

Implementation:

- Use ElastiCache and DynamoDB for data storage.
- Prepare a data store for storing the state information.
- Use, as a key in the data store, an ID that identifies the user (a session ID or user ID), and store the user information as a value.
- Store, reference, and update the state information in the data store, instead of storing it in the web/APP server.



State sharing is important to implement for an auto scaled fleet.

Because access to state information from multiple web/AP servers is concentrated on a single location, you must use caution to prevent the performance of the data store from becoming a bottleneck.

Scheduled Scale-Out

Problem: An application's traffic does not scale organically, but has huge jumps at specific periods of the day or for an event.

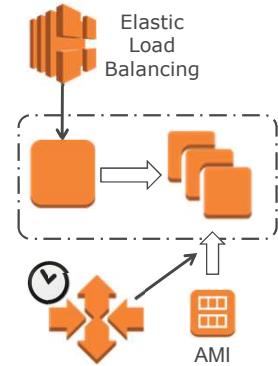
Solution: Use [Scaling by Schedule](#) or [Scaling by Policy](#).

Advantages:

- Allows you to scale in advance of a traffic spike that you know will occur.

Implementation:

- Create a customized AMI.
- Create a Launch Config for your Auto Scaling group.
- Create an Auto Scaling group for your instances (behind a load balancer).
- Options:
 - Create Schedule Update to launch or terminate instances at a specified time.
 - Create Scale by Recurrence policy that will automatically scale your instances based upon cron.



Job Observer Pattern

Problem: You need to manage resources against the depth of your work queue.

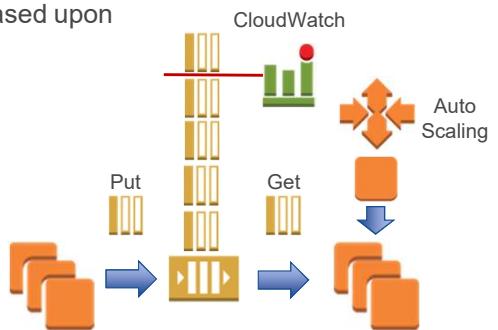
Solution: Create an Auto Scaling group to scale compute resources based upon queue depth.

Advantages:

- Compute scales with job size, providing efficiency and savings.
- Job can be completed in a shorter timeframe.
- Even if a work item fails to complete, process is resilient.

Implementation:

- Work items for batch job placed in Amazon SQS queue as messages.
- Auto Scaling group should be created to scale compute resources up or down based upon Amazon Cloudwatch queue depth metric.
- Batch processing servers grab work items from Amazon SQS to complete job.



Bootstrap Instance

Problem: Code releases happen often. Creating a new AMI every time you have a release and managing these AMIs across multiple regions is difficult.

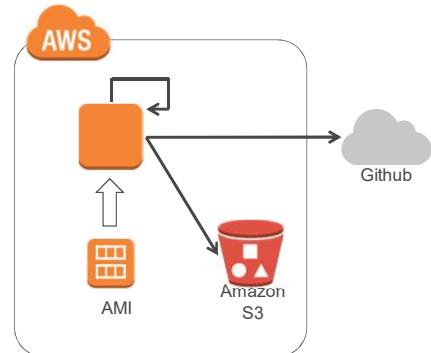
Solution: Develop a base AMI, and then bootstrap the instance during the boot process to install software, get updates, and install source code so that your AMI rarely or never changes.

Advantage:

- Do not need to update AMI regularly and move customized AMI between regions for each software release.

Implementation:

- Identify a base AMI to start from.
- Create a repository where your code is located.
- Identify all packages and configs that need to occur at boot of the instance.
- During boot, pass user data to your EC2 instances that will be executed to bootstrap your instance.

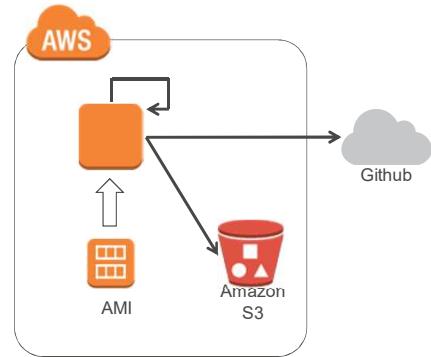


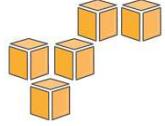
- During boot, it will probably take more time to install and perform configuration than it would with a golden AMI.
- Bootstrapping can also be done through Auto Scaling and AWS CloudFormation.

Bootstrap Instance - Example

```

66     "LaunchConfig" : [
67         {
68             "Type" : "AWS::AutoScaling::LaunchConfiguration",
69             "Metadata" : {
70                 "Comment1" : "Configure the bootstrap helpers to install the Apache Web Server and PHP",
71                 "Comment2" : "The website content is downloaded from the index.zip file",
72             },
73             "AWS::CloudFormation::Init" : {
74                 "config" : {
75                     "packages" : {
76                         "yum" : [
77                             "httpd" : [],
78                             "php" : []
79                         ],
80                     }
81                 },
82                 "sources" : [
83                     {
84                         "/var/www/html" : "https://s3.amazonaws.com/bhol/index.zip"
85                     }
86                 ],
87                 "services" : {
88                     "sysvinit" : {
89                         "httpd" : {
90                             "enabled" : "true",
91                             "ensureRunning" : "true"
92                         }
93                     }
94                 }
95             },
96         }
97     ],
98 
```





Part 2

Stream Processing Example

The Amazon Associates program

© 2017 Amazon Web Services, Inc. or its affiliates. All rights reserved.

 Academy

Part 2: Stream Processing Example

The Amazon Associates Program

How to build this monthly payment system?



Registered Amazon Associate adds a link to Amazon.com in their website

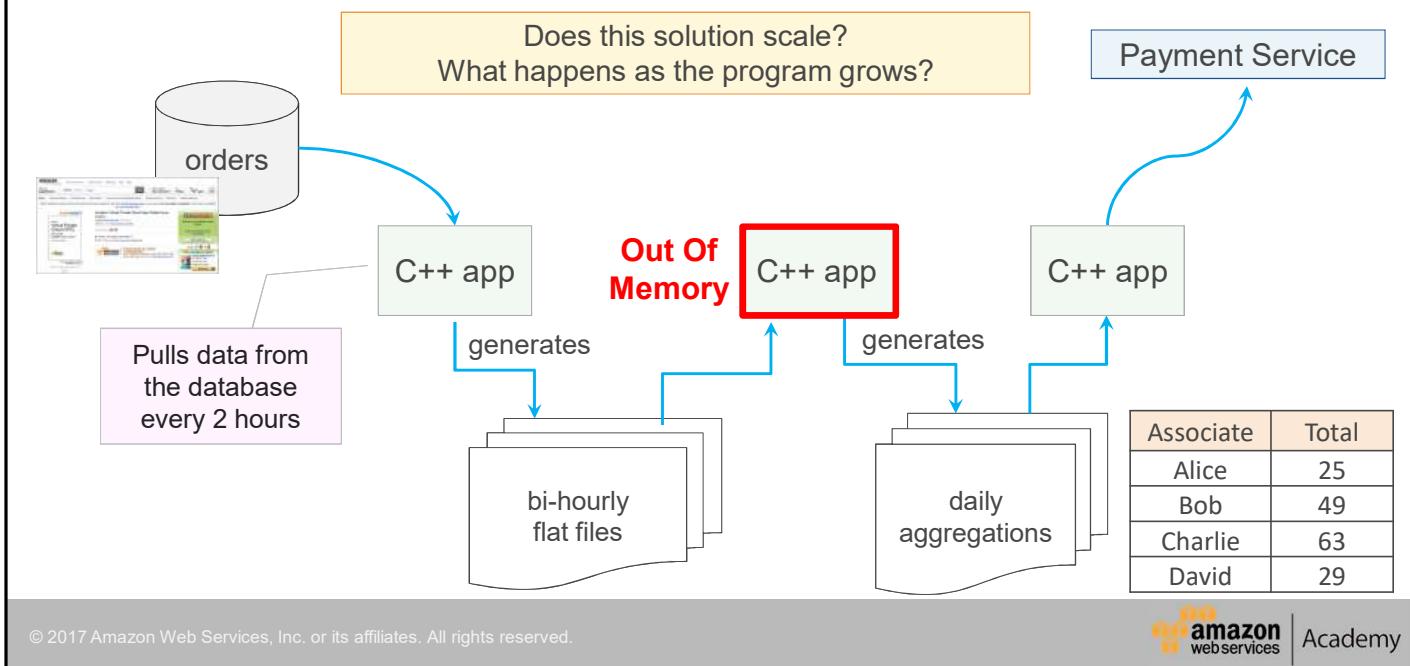
A screenshot of an Amazon product page for "Amazon Virtual Private Cloud User Guide (Kindle Edition)". The page shows the book cover, title, author (Amazon Web Services), price (\$0.00), and a "Buy now with 1-Click" button. A red circle highlights this button. Below it, there's a "Also available on Kindle" section.

Amazon Associate earns advertising fees for qualified purchases (monthly payout)

Payment Service

Purchase was made by the user who visited Amazon.com from Associate's link

The Amazon Associates Program – Generation 1



The Amazon Associates program is one of their oldest marketing programs. With this program, you add a link to Amazon.com on **your** website, and if people click on the link and buy something, Amazon will give you a share of that sale, up to 15%. (<https://affiliate-program.amazon.com/gp/associates/join/landing/main.html>)

How much to pay each associate every month is an extremely important problem. It is critical that Amazon is correct, but still it seems like a fairly straightforward problem: Amazon tracks every sale attributed to an associate link, adds up the sales at the end of the month, and pays at the appropriate rate. The reality is a little more complicated, though, because Amazon must track an order throughout its lifecycle to know whether the order was cancelled or returned.

This slide illustrates the system that Amazon built over 12 years ago. A **single-threaded** C++ application connected directly to the production orders database every two hours and dumped out the orders it was interested in to a series of flat files. Next, another monolithic C++ application ran once a day to aggregate the bi-hourly files into a daily view. Finally, another system ran once a month to scan over the daily aggregates, calculate how much to pay people, and give that information to the payments service, which was run by a totally separate team.

This system ran very smoothly until about 2008, when the scale of the associates program had grown to the point where the daily aggregations system started to run out of memory. The program needed a way to process large volumes of data quickly.

Solving The Problem With Amazon Elastic Mapreduce

Amazon Elastic MapReduce (EMR) is **Hadoop in the AWS cloud**.

What is Hadoop?

An open source project written in Java, managed by the Apache Software Foundation.

Built for:

- Handling high data variety, volume, and velocity
- Solution flexibility: Write your own data apps
- Processing large batches of data (such as logs)



Hadoop uses a distributed processing architecture called MapReduce in which a task is mapped to a cluster of commodity servers for processing. Each piece of work distributed to the cluster servers may be run or re-run on any of the servers. The cluster servers use HDFS to store data locally for processing. The results of the computation performed by those servers are then reduced to a single output set. One node, designated as the master node, controls the distribution of tasks and can automatically handle server failures.

Hadoop provides a number of benefits:

- *Handling uncertainty*: Hadoop facilitates data navigation, discovery, and ad-hoc data analysis. Hadoop allows you to compensate for unexpected occurrences by allowing you to analyze large amounts of data quickly in order to form a response.
- *Handling data variety*: Unlike traditional database systems, Hadoop can process structured, unstructured, or semi-structured data. This includes virtually any data format currently available. In addition to natively handling many types of data (such as XML, CSV, text, log files, objects, SQL, JSON, and binary), Hadoop can transform data into formats that allow better integration into your existing data sets. Also, you can store data with or without a schema, and you can perform large-scale ETL (extract, transform, load) to transform your data.
- *Providing flexibility*: Because Hadoop is open source, there are a number of ecosystem projects that can help you analyze the multiple types of data Hadoop can process and analyze. These projects allow you tremendous flexibility when developing big data solutions. Hadoop's programming frameworks (Hive, Pig, etc.) can support almost any big data use case for your applications.
- *Handling data volume and velocity*: Because of Hadoop's distributed architecture,

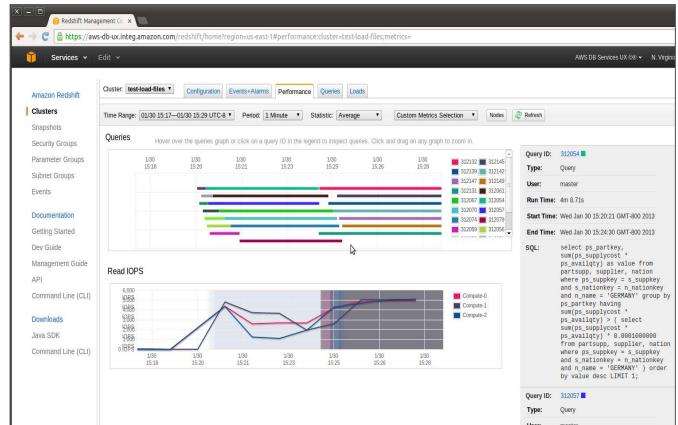
Hadoop clusters can handle tremendous amounts of data affordably. Adding additional data processing capability is as simple as adding additional servers to your cluster (linear scaling).



Build In Business Intelligence With Amazon Redshift

Amazon Redshift is a fully managed, petabyte-scale **data warehouse** service.

- Provision in minutes.
- Monitor query performance.
- Point and click resize.
- Built-in security.
- Automatic backups.



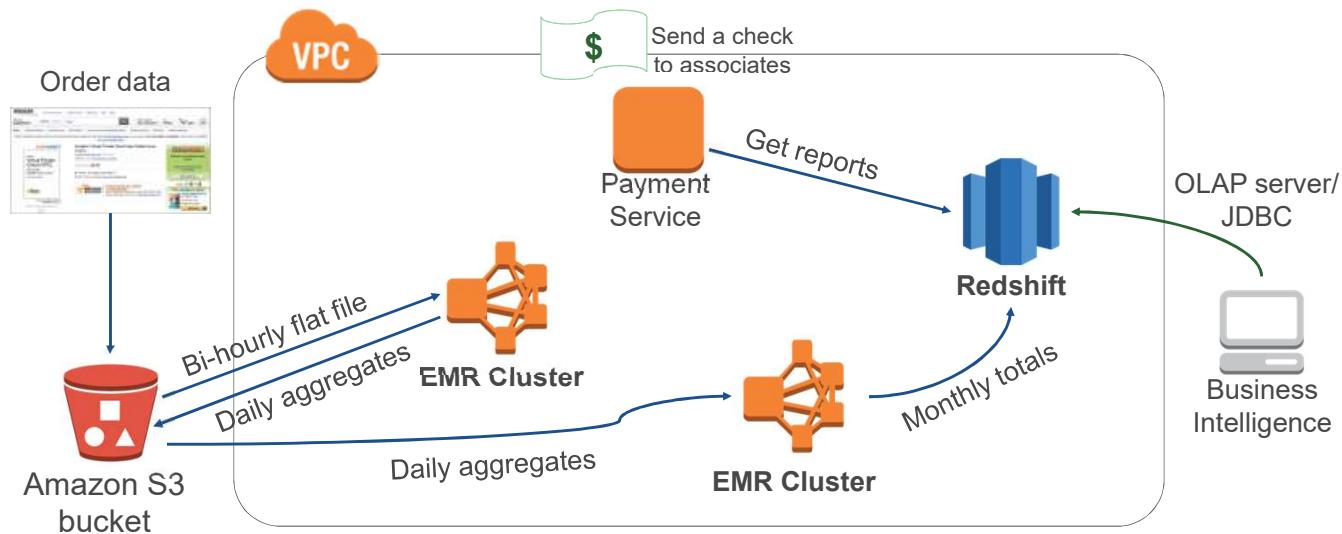
© 2017 Amazon Web Services, Inc. or its affiliates. All rights reserved.



Redshift features:

- SSL/TLS to secure data in transit.
- Encryption to secure data at rest.
 - AES-256; hardware accelerated
 - All blocks on disks and in Amazon S3 encrypted
 - HSM Support
- No direct access to compute nodes.
- Audit logging and AWS CloudTrail integration.
- Replication within the cluster and backup to Amazon S3 to maintain multiple copies of data at all times.
- Backups to Amazon S3 are continuous, automatic, and incremental.
 - Designed for eleven nines of durability.
- Continuous monitoring and automated recovery from failures of drives and nodes.
- Able to restore snapshots to any Availability Zone within a region.
- Easily enable backups to a second region for disaster recovery.

Handle Growing Volume With EMR – Generation 2



© 2017 Amazon Web Services, Inc. or its affiliates. All rights reserved.



To implement Amazon EMR as a solution for the Amazon Associates program's architecture, the team developed a system that publishes the raw orders stream to a set of registered listeners. The associates team wrote an application that listens to the orders stream, filters it, grabs any additional data it might need about the order from other services, and jams all that into Amazon S3. Then, a Hadoop cluster wakes up every day and pulls 60 days of order history out of Amazon S3. It adds up the month-to-date sales of every associate, computes pay rates, and stores all that back to Amazon S3. Now that the architecture has adopted a distributed solution, larger and larger data sizes are no longer a problem. As their volume grows, they don't have to scrape and struggle to figure out how to keep fitting it in memory on a single machine. All they have to do is add machines to their cluster. The code that runs in the cluster doesn't change at all.

To solve all these issues, Amazon moved to Amazon EMR, which is Hadoop in the cloud.

There are significant advantages to Amazon EMR:

- Cluster setup and resizing takes minutes.
- You can create application-specific cluster configurations.
- A Container-managed service makes software installation and upgrades trivial.
- Cost is usually much lower.

Consider the total cost of on-premises in comparison to Amazon EMR, including:

- The time and expense in sizing and ordering hardware.
- The development effort.
- The cost of maintaining the system and administrative personnel.

What's Next?

Generation 2 solution achieved the following:

- Replaced the file system with Amazon S3.
- Leveraged Elastic MapReduce to remove the performance bottleneck that was caused by the single-threaded C++ application.
- Added analytics and data insight with Amazon Redshift.

How do you further improve this system as the load increases?

- **Real-time** data processing

Batch Versus Streaming: Other Considerations

Batch processing (“cold” data)

- 💡 Hourly server logs
System issues an hour ago
- 💡 Weekly, monthly bill
Spending in past billing cycle
- 💡 Daily customer preference report from clickstream logs
- 💡 Daily fraud reports
Was there fraud yesterday?

Stream processing (“hot” data)

- 💡 Compiling CloudWatch metrics
Monitor issues immediately
- 💡 Real-time spending alerts, caps
Respond to sudden cost increases
- 💡 Real-time analysis
What to offer the customer now
- 💡 Real-time detection
Block fraudulent use now

Batch processing typically involves querying large amounts of “cold” data. In batch processing, it may take hours to get answers to business questions. For example, you may use batch processing to generate a billing report at the end of the month. Stream processing in real time, typically, involves querying small amounts of “hot” data, and it takes only a short amount of time to get answers.

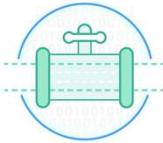
Amazon Kinesis



Ingest streaming data

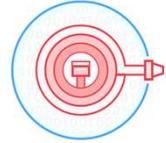
Process data in real-time

Store terabytes of data per hour



Amazon Kinesis Streams

Build your own custom applications that process or analyze streaming data



Amazon Kinesis Firehose

Easily load massive volumes of streaming data into Amazon S3 and Redshift



Amazon Kinesis Analytics

Easily analyze data streams using standard SQL queries

Amazon Kinesis Streams: Key benefits

Near real-time performance



Perform continual processing on streaming big data. Processing latencies fall to a few seconds, compared with the minutes or hours associated with batch processing.



Although Amazon Kinesis can be used to solve a variety of streaming data problems, a common use is the near real-time aggregation of data followed by loading the aggregate data into a data warehouse or Amazon EMR cluster.

Amazon Kinesis Streams: Key benefits



Near real-time performance

Perform continual processing on streaming big data. Processing latencies fall to a few seconds, compared with the minutes or hours associated with batch processing.



Build near real-time apps

Client libraries that enable developers to design and operate near real-time streaming data processing applications.

High throughput, elastic

Seamlessly scale to match your data throughput rate and volume.

You can easily scale up to gigabytes per second. The service can scale in or out based on your operational or business needs.



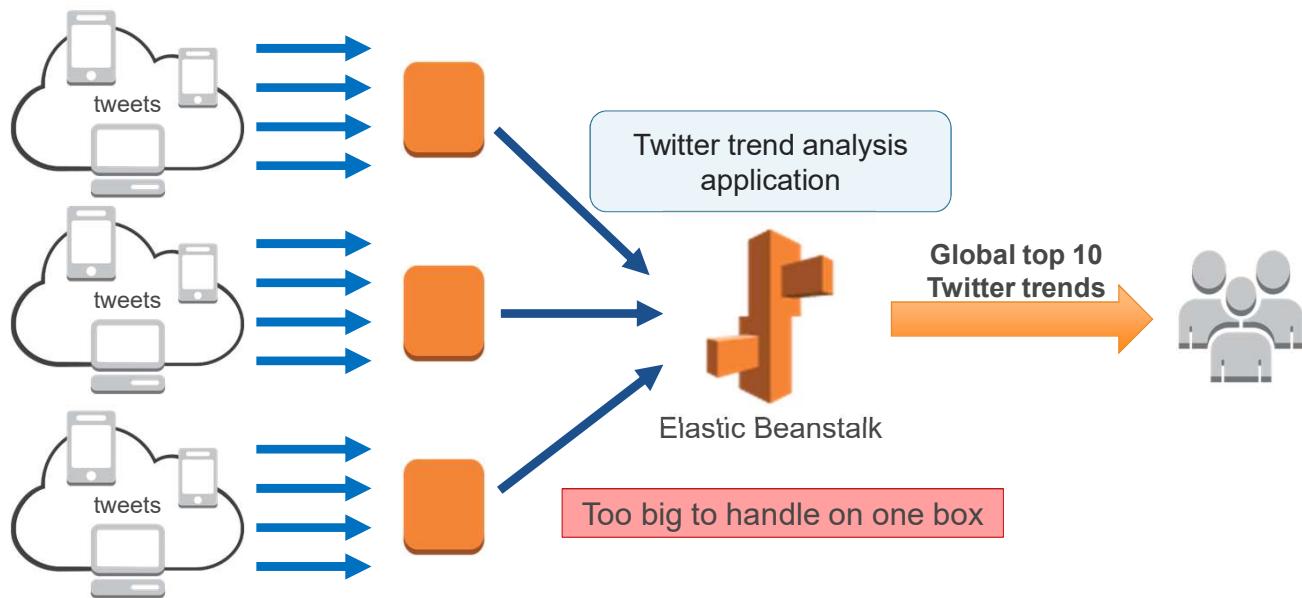
Low cost

Cost-efficient for workloads of any scale. You can get started by provisioning a small stream, and pay low hourly rates only for what you use.



Although Amazon Kinesis can be used to solve a variety of streaming data problems, a common use is the near real-time aggregation of data followed by loading the aggregate data into a data warehouse or Amazon EMR cluster.

Example Application: Twitter Trend Analysis Application



© 2017 Amazon Web Services, Inc. or its affiliates. All rights reserved.

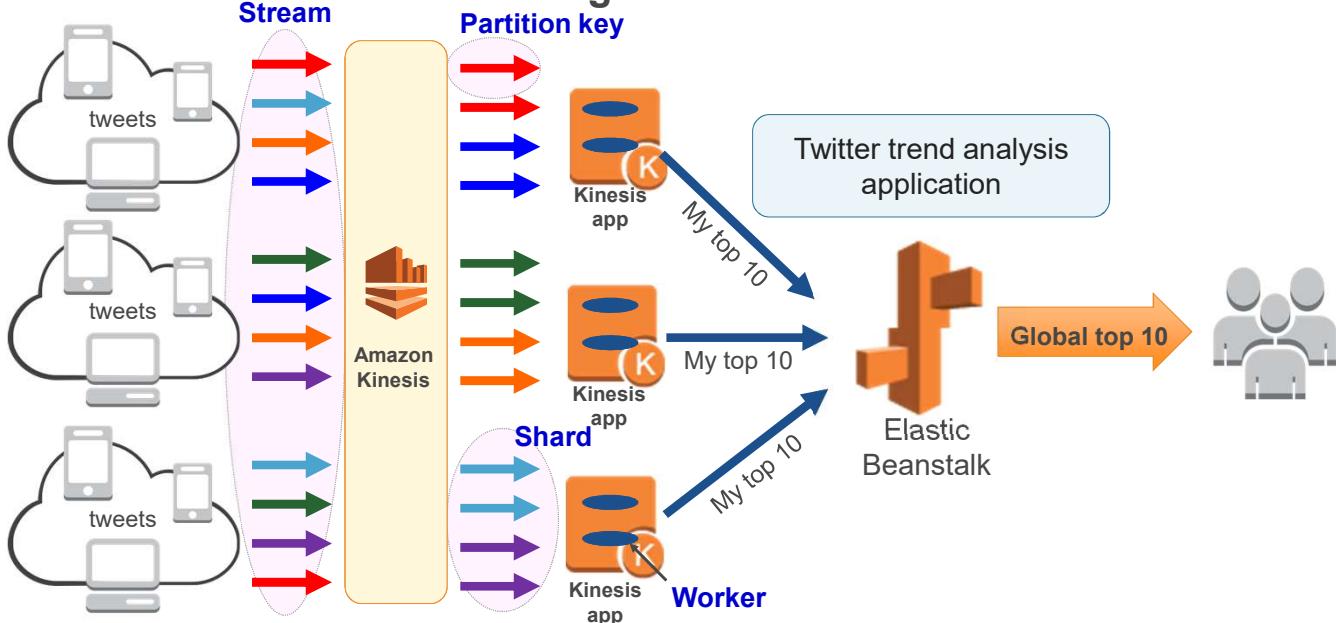


The desired functionality for the [twitter-trends.com](#) website is to display the top 10 current twitter trends. On AWS, you might begin by simply creating an AWS Elastic Beanstalk website.

Unfortunately, the Twitter fire-hose feed is quickly becoming too big to process on a single machine. You will need to divide up the work so that it can be done in parallel on multiple machines and then coalesced into an output that can be vended by the Elastic Beanstalk web server.

What you want to do is to compute a local top-10 value for all the tweets coming into each processing instance, and then combine all top-10 lists into a global top-10 list at the web server. The problem is that, without some sorting or grouping function, each processing instance will receive a random selection of tweets, which makes it impossible for any one instance to know how many tweets are occurring for any specific topic.

Solution: Stream Processing With Amazon Kinesis



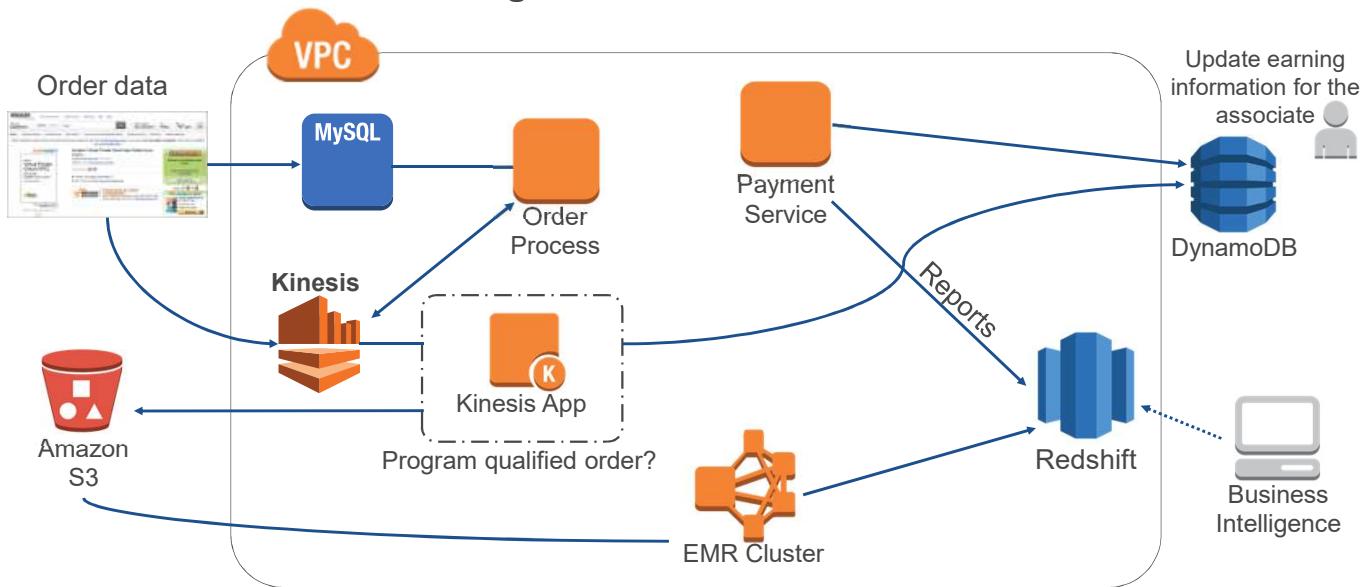
© 2017 Amazon Web Services, Inc. or its affiliates. All rights reserved.



You can solve this problem by taking a page from the map/reduce world. If you are computing the top 10 tweeted topics for a standalone batch of tweets, treat the topic of each tweet as a partition key (e.g., #WorldCup). Each map task would take in a fraction of all tweets, group by topic, and send the count for each topic to the appropriate reduce task responsible for that topic.

You can do something equivalent by introducing an intermediate stage that takes in the stream of tweets and orders them by topic. Each processing box would then pull only the tweets for the set of topics that it is responsible for. This would enable each box to be an authoritative counter of tweets for some subset of all currently active tweet topics. This is essentially the streaming equivalent of the map/reduce processing paradigm.

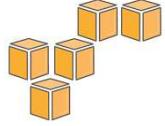
Real-Time Order Processing With Kinesis – Generation 3



© 2017 Amazon Web Services, Inc. or its affiliates. All rights reserved.

Amazon web services Academy

1. Every order that Amazon.com receives is pushed to Kinesis.
2. If the order is linked to an Associate program (qualified order), that order information is stored in an Amazon S3 bucket. Because the qualified order is mapped to a specific Amazon associate, the information can be stored in DynamoDB. The associate can log into a web application to see how many people came to his/her website and purchased a product at Amazon.com.
3. The EMR cluster retrieves the qualified order information from the Amazon S3 bucket and then the monthly total is stored in Redshift. A business intelligence application can run some analysis.
4. Payment service pulls the report and sends checks to the associates every month.



Part 3

Sensor Network Data Ingestion and Processing Example

Flu Outbreak Heat Map

© 2017 Amazon Web Services, Inc. or its affiliates. All rights reserved.

 **Amazon**
webservices | Academy

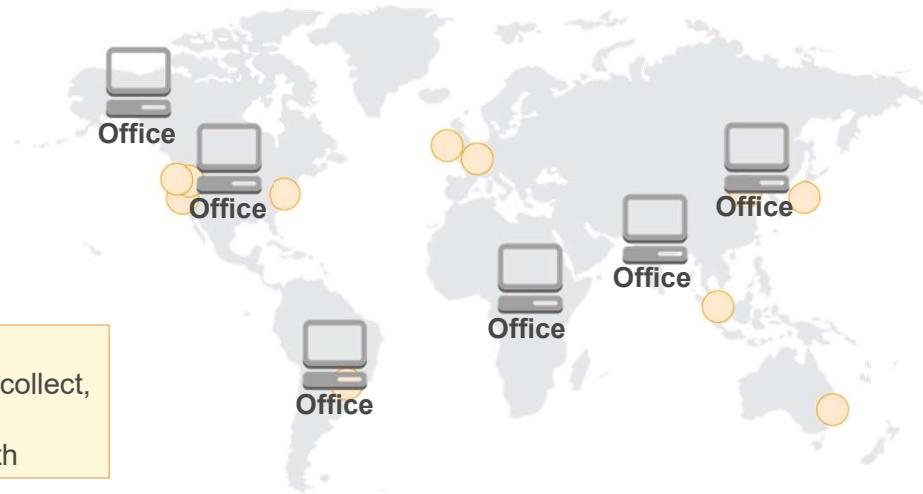
Part 3: Sensor Network Data Ingestion and Processing Example
Flu Outbreak Heat Map

Government Health Organization: Flu Data

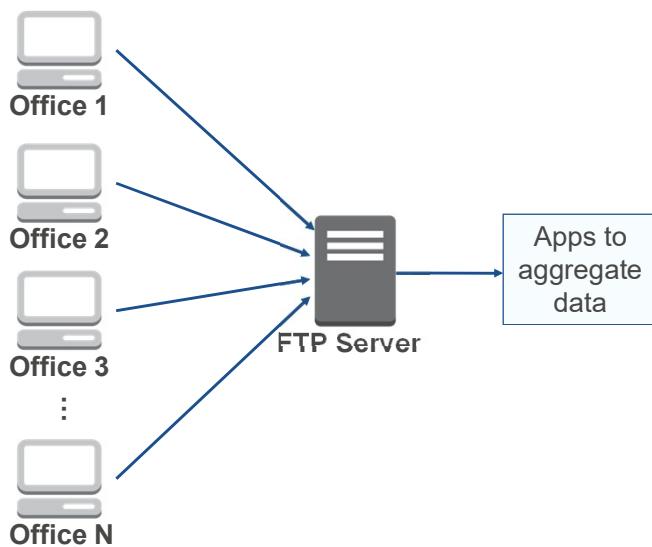
Scenario: Influenza outbreak around the world

Mission:

Based on the data you collect,
generate heat maps to
understand public health



Flu Heat Map – Generation 1



- Does this scale?
- Is there a concern for a single point of failure?
- What AWS services would you use?

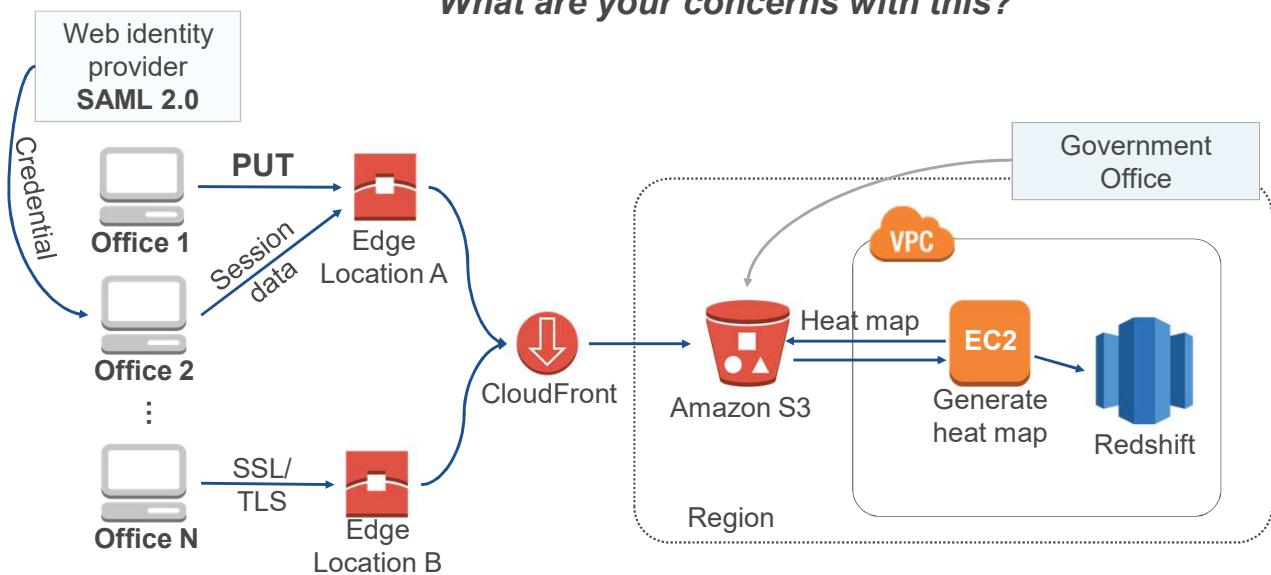
This is based on a real-world example. It is not rare to see a government organization using a very basic technology in place. In this case, all the federal offices send the flu data to a central FTP server.

This solution does not scale, and the FTP server can become a single point of failure. Because of the nature of file systems, this solution fails to generate a near real-time heat map for flu cases.

Would you use Kinesis? The problem with Kinesis is its payload size limitation. The data coming from global offices probably exceeds the allowed Kinesis payload size.

Flu Heat Map – Generation 2

What are your concerns with this?



© 2017 Amazon Web Services, Inc. or its affiliates. All rights reserved.

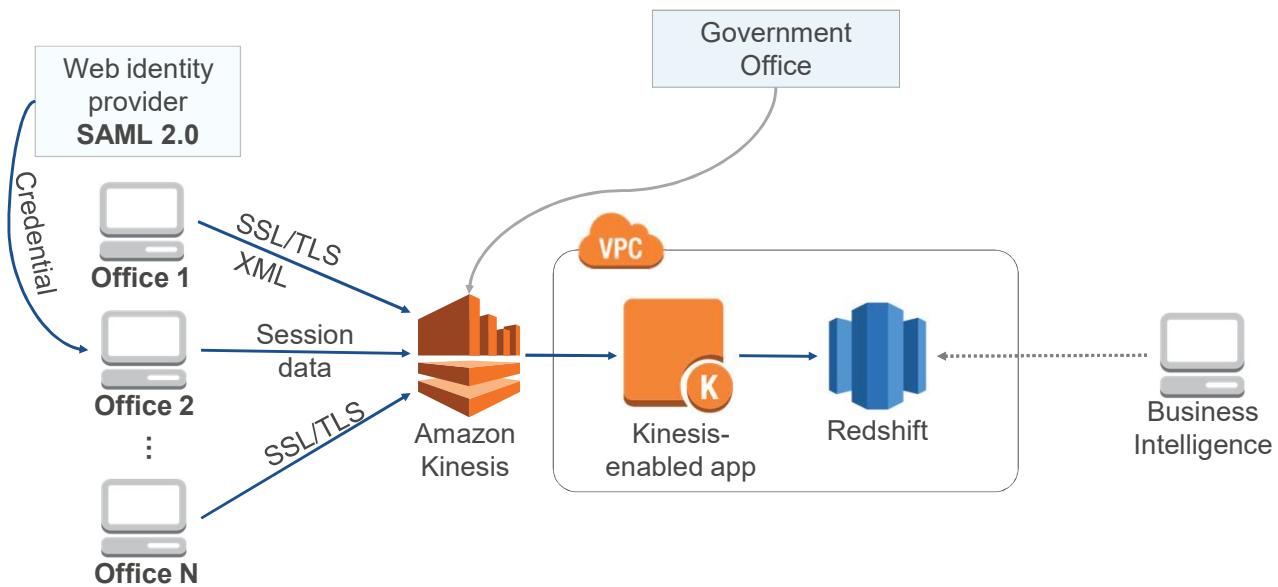
Amazon web services | Academy

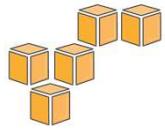
CloudFront

Compatible but does not cache the following operations:

- POST
- PUT
- PATCH
- DELETE
- OPTIONS

Flu Heat Map – Generation 3





Section 4

Application Back-End Example

Mobile Gaming

© 2017 Amazon Web Services, Inc. or its affiliates. All rights reserved.

 Academy

Part 4: Application Back-End Example
Mobile Gaming

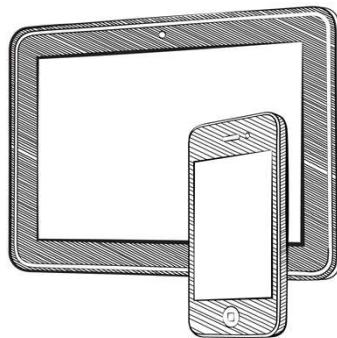
Mobile Games

Mobile app revenue is projected to grow continuously.



Mobile back-end technologies:

- HTTP-based
- External Social APIs
- Save state
- Database
- Static data store
- Mobile push
- Analytics



Mobile online features may include:

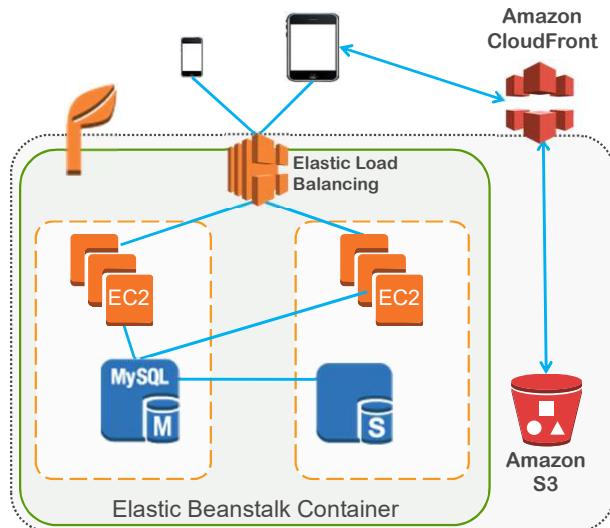
- Social login
- Friends
- Leaderboards
- Push messages
- Analytics

Mobile gaming customers that use AWS:

- Supercell with Hay Day and Clash of Clans
- Halfbrick with Fruit Ninja
- Wooga with Diamond Dash and Pocket Village
- Ubisoft, which leveraged AWS to launch 10 social games in 18 months
- SEGA, which migrated forums and some game servers to AWS and reported a 50% savings in server costs
- FunPlus, a successful Facebook game company that began without physical servers but quickly moved to AWS to gain agility and reduce costs

Mobile Games Back-End Concepts

- Think in terms of APIs
- GET friends, leaderboards
- HTTP + JSON
- Multiplayer servers
- Binary assets
- Game analytics



Game back ends are looking more and more like web applications. Lots of calls map to REST: get friends, leaderboards, active games, and even login. With multiplayer games, the core APIs, such as listing servers and joining games, map well to REST calls. Use stateful sockets sparingly.

- Amazon S3 for game data such as assets, UGC (User Generated Content), and analytics
- Auto Scaling group
 - Capacity on demand
 - Respond to users
- ElastiCache to offload the database
 - Memcached
 - Redis
- CloudFront CDN
 - DLC (Downloadable Content), assets
 - Game saves
 - UGC
- Beanstalk manages
 - Elastic Load Balancing
 - Amazon EC2
 - Auto Scaling
 - Monitoring
 - Amazon RDS

What Information Can Help Improve Your Game?

Sentiment analysis

- Enjoying game
- Engaged
- Like/dislike new content
- Stuck on a level
- Bored
- Abandonment

Players' behavior

- Hours played day/week
- Number of sessions/day
- Level progression
- Friend invites/referrals
- Response to mobile push
- Money spent/week



To get insight into how your application is perceived, it can be helpful to collect data about the behaviors of your users. This means ingesting and processing streams of data as your application is being used, and then performing analysis on that data to gain insights from it. This slide includes some examples of how to use data analysis to gain these kinds of business insights.

Data Analytics For Gaming

Batch processing examples:

- What game modes do people like best?
- How many people have downloaded DLC pack 2?
- Where do most people die on *map 4*?
- How many daily players are there on average?

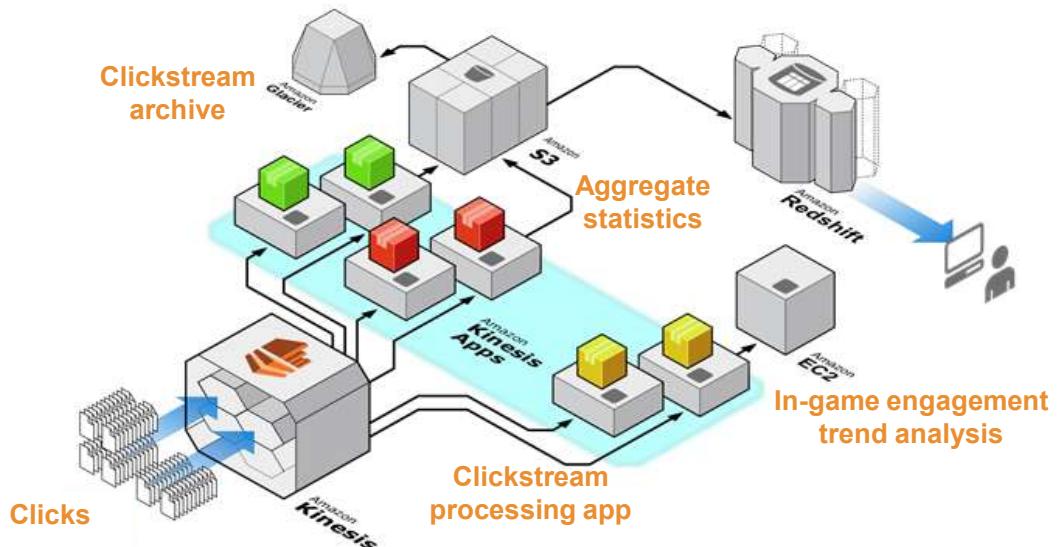
Real-time processing examples:

- What game modes are people playing now?
- Are more or fewer people downloading DLC today?
- Are people dying in the same places, or different places?
- How many people are playing today? Any variance?

Downloadable content (DLC) is additional content for a video game distributed through the Internet by the game's official publisher or other third-party content producers. DLC can be of several types, ranging from aesthetic outfit changes to a new, extensive storyline, similar to an expansion pack.

Both types of analysis are valuable, but obviously, when you can quickly determine what your players love, you can quickly make your game have more of that quality.

Reference Architecture: Data Analytics For Gaming



© 2017 Amazon Web Services, Inc. or its affiliates. All rights reserved.



Use cases for data analytics

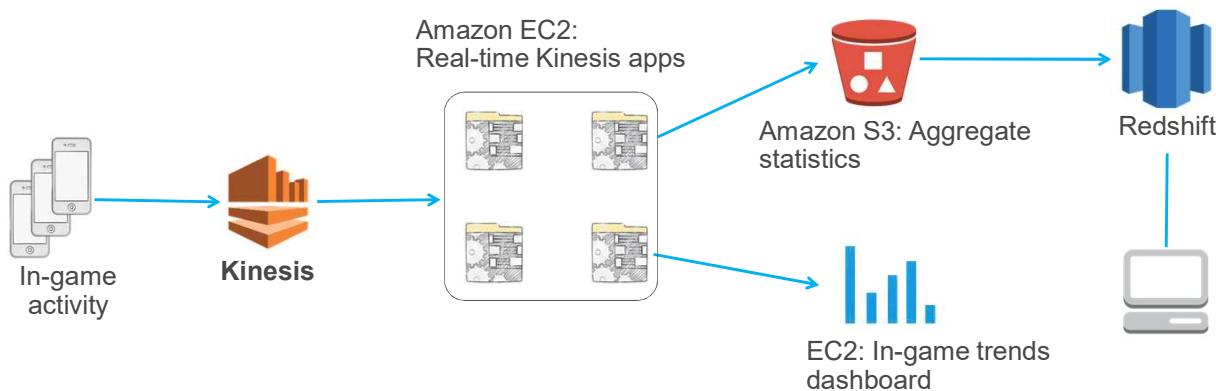
- Tracking player performance to keep games balanced and fair.
- Track game performance metrics by geography/region for online games where players sign in from around the world.
- Tracking sales performance of items sold in-game.
- Tracking performance of online ads served in-game.
- Accelerating to near real-time the delivery of the metrics described above.
- Continual in-game metrics on user engagement.
- Real-time analytics on user engagement in the game.
- Optimized predictions of when/where to sell in-game purchases.

Business benefits for data analytics

- Reduce operational burden on your developers.
- Scale your data analytics to match highly variable gaming loads without overpaying for spare capacity or increased time to results when usage is up.
- Try many more data experiments and find truly game-changing new metrics and analysis.
- Ingest and aggregate gaming logs that were previously uncollected or unused.
- Accelerate the time to results of batch processing of game logs [for example] to reduce the delivery of gaming metrics from every 48 hours to every 10 minutes.
- Deliver continuous/real-time game insight data from hundreds of game servers.

Case Study: Supercell

Analytics pipeline for Clash of Clans



© 2017 Amazon Web Services, Inc. or its affiliates. All rights reserved.



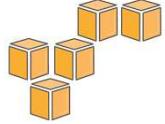
Finland-based Supercell, founded in 2010 by six game-industry veterans, is one of the fastest-growing social game developers in the world. With a staff of just over 100 employees, Supercell has three games—**Hay Day** (a social farming game), **Clash of Clans** (a social resource management game with strategic combat), and **Boom Beach** (combat strategy game, released in March 2014)—that attract tens of millions of players on iOS and Android devices every day.

Here, we are looking at Clash of Clans. In addition to loading into Redshift for business intelligence tasks, the data is also moved into Amazon S3 for long-term archiving. If the team ever needs to recover data, load retrospective information into a new data warehouse instance, or augment their reports with data that was not originally available, they can restore the vaults from Amazon Glacier.

Using Kinesis with Amazon EC2, Amazon S3, Amazon Glacier, and Redshift gives Supercell a complete, 360-degree view of how their customers are playing their games and provides the resources to dive into that data to answer specific questions such as who are the high scorers, which areas of the game world are most popular, or how their most recent updates are performing.

“We are using Amazon Kinesis for real-time delivery of game insight data sent by hundreds of our game engine servers. Amazon Kinesis also offloads a lot of developer burden in building a real-time, streaming data ingestion platform, and enables Supercell to focus on delivering games that delight players worldwide,” says Sami Yliharju, Services Lead at Supercell.

You can read more about the Supercell case study at
<http://aws.amazon.com/solutions/case-studies/supercell/>



Part 5

Transcoding and Serving Video Files Example

© 2017 Amazon Web Services, Inc. or its affiliates. All rights reserved.

 **Amazon**
webservices | Academy

Part 5: Transcoding and Serving Video Files Example

Transcoding Video Files Can Be A Challenge

- Challenging to set up:
 - Difficult to choose the right transcoding settings
 - Complicated to build a system that performs well
- Difficult to adapt:
 - Amount of content can scale up quickly
 - Output formats and device types change frequently
- Complex cost issues:
 - Building for peak is uneconomical
 - Storage and delivery costs on a large scale can be prohibitive



Many social media enable their users to share both photos and videos. Service providers face some challenges dealing with video files because people use multiple devices to access the videos (smartphones, PCs, tablets, etc.).

AWS Solution: Amazon Elastic Transcoder

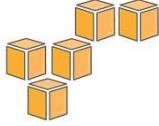


© 2017 Amazon Web Services, Inc. or its affiliates. All rights reserved.



Amazon Elastic Transcoder is a highly scalable, easy-to-use and cost-effective way for developers and businesses to convert (or “transcode”) video files from their source format into versions that will play back on devices like smartphones, tablets, and PCs. You can use Elastic Transcoder to convert files from different media formats into H.264/AAC/MP4 files at different resolutions, bitrates, and frame rates, and set up transcoding pipelines to transcode files in parallel.

Elastic Transcoder is for any customer with media assets stored in Amazon S3. Examples include developers who create apps or websites that publish user-generated content, enterprises and educational establishments that convert training and communication videos, and content owners and broadcasters that need to convert media assets into web-friendly formats.



In review...

- 💡 High Availability Design Patterns
- 💡 Stream Processing Example
- 💡 Sensor Network Data Ingestion and Processing Example
- 💡 Application Back-End Example
- 💡 Transcoding and Serving Video Files



Knowledge Assessment

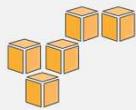
© 2017 Amazon Web Services, Inc. or its affiliates. All rights reserved.



In review...

- High Availability Design Patterns
- Stream Processing Example
- Sensor Network Data Ingestion and Processing Example
- Application Back-End Example
- Transcoding and Serving Video Files

To complete this module, please remember to finish the corresponding knowledge assessment.



CCA Unit 3 – Architecting on AWS

CCA 3.14: Design Patterns and Sample Architectures

Section 1: Introduction to System Design

Section 2: Automation and Serverless Architectures

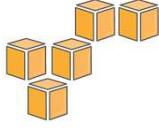
Section 3: Well-Architected Best Practices

Section 4: Deployment and Implementation

© 2017 Amazon Web Services, Inc. or its affiliates. All rights reserved.



Congratulations. This module concludes Unit 3, Architecting on AWS and the Cloud Computing Architecture curriculum.



Up Next...



Final Project

© 2017 Amazon Web Services, Inc. or its affiliates. All rights reserved.

 Academy

To complete the course, you will complete the final project.

© 2017 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.

For all other questions, contact us at
<https://aws.amazon.com/contact-us/aws-training/>.

All trademarks are the property of their owners.

© 2017 Amazon Web Services, Inc. or its affiliates. All rights reserved.

