In this reading, you will find further information about the topics that Morgan and Raf talked about in the video where they both played Solutions Architects.

## Caching for Amazon DynamoDB by using Amazon DynamoDB Accelerator

Raf mentioned that if you need to improve Amazon DynamoDB performance to microsecond latency, to look into using Amazon DynamoDB Accelerator (DAX).   
  
DAX is a fully managed, highly available, in-memory [cache](https://aws.amazon.com/caching/) for DynamoDB that’s designed to deliver up to a 100-times performance improvement—from milliseconds to microseconds—even at millions of requests per second.  
  
DAX does the heavy lifting that’s required to add in-memory acceleration to your DynamoDB tables—and developers don’t need to manage cache invalidation, data population, or cluster management.  
  
A benefit of using DAX is that you don’t need to modify the application logic because DAX is compatible with existing DynamoDB API calls.  
  
DAX is designed to run within an Amazon Virtual Private Cloud (Amazon VPC) environment. Amazon VPC defines a virtual network that closely resembles a traditional data center. With a VPC, you have control over its IP address range, subnets, routing tables, network gateways, and security settings. You can launch a DAX cluster in your virtual network, and control access to the cluster by using Amazon VPC security groups.  
  
For more information about DAX, see [Amazon DynamoDB Accelerator.](https://aws.amazon.com/dynamodb/dax/)

## Optimizing AWS Lambda

### AWS Lambda Power Tuning

Morgan mentioned that you can fine-tune the memory or power configuration for your AWS Lambda functions to potentially increase performance and lower costs.  
  
[AWS Lambda Power Tuning](https://github.com/alexcasalboni/aws-lambda-power-tuning) is an open-source tool that helps you visualize and fine-tune the memory or power configuration of Lambda functions. It runs in your own AWS account, and it supports three optimization strategies: cost, speed, and balanced.  
  
AWS Lambda Power Tuning is a state machine that’s powered by AWS Step Functions. It helps you optimize your Lambda functions for cost or performance in a data-driven way.  
  
The state machine is designed to be easy to deploy and fast to execute. Also, it's language agnostic, so you can optimize any Lambda functions in your account.  
  
To work with AWS Lambda Power Turning, you provide a Lambda function Amazon Resource Name (ARN) as input. The state machine then invokes that function with multiple power configurations (from 128 MB to 10 GB—you decide which values). Then, it analyzes all the execution logs and suggests the best power configuration to minimize cost or maximize performance.  
  
Note that the input function will run in your AWS account, which means that it will perform HTTP requests, SDK calls, cold starts, and so on. The state machine also supports cross-Region invocations, and you can enable parallel execution to generate results in a few seconds.  
  
The state machine generates a visualization of average cost and speed for each power configuration.  
  
For example, the following diagram shows results for two CPU-intensive functions, which become both cheaper and faster with more power:  
For more information, see [AWS Lambda Power Tuning](https://github.com/alexcasalboni/aws-lambda-power-tuning).

### AWS Lambda Powertools

Morgan also mentioned that you can use another suite of tools called AWS Lambda Powertools to optimize your Lambda functions and use best practices. AWS Lambda Powertools is a suite of utilities for AWS Lambda functions that is designed to make it easier to adopt best practices such as tracing, structured logging, custom metrics, idempotency, batching, and more.   
  
For more information, see [AWS Lambda Powertools](https://awslabs.github.io/aws-lambda-powertools-python/latest/).

### AWS Lambda execution environment reuse

Another optimization technique that Morgan mentioned for Lambda is to move certain initialization tasks in your code so they are outside the handler. These tasks can then be reused across invocations (which is also known as execution environment reuse).  
  
You can take advantage of execution environment reuse to improve the performance of your function. To do this, initialize SDK clients and database connections outside of the function handler, and cache static assets locally in the /tmp directory. Subsequent invocations that are processed by the same instance of your function can reuse these resources. This reuse saves cost by reducing function run time.   
  
To avoid potential data leaks across invocations, don’t use the execution environment to store user data, events, or other information with security implications. If your function relies on a mutable state that can’t be stored in memory within the handler, consider creating a separate function or separate versions of a function for each user.  
  
To learn more, see [Best practices for working with AWS Lambda.](https://docs.aws.amazon.com/lambda/latest/dg/best-practices.html)

### Resources

* For more information about how to optimize serverless applications for cost, see [Building well-architected serverless applications: Optimizing application costs](https://aws.amazon.com/blogs/compute/building-well-architected-serverless-applications-optimizing-application-costs/) in the *AWS Compute Blog*.
* For more information about well-architected serverless applications, visit [Serverless Applications Lens - AWS Well-Architected Framework](https://docs.aws.amazon.com/wellarchitected/latest/serverless-applications-lens/welcome.html?did=wp_card&trk=wp_card).

Skip to main content

Customer #1: Use Case and Requirements

8:09 / 8:09

Press UP to enter the speed menu then use the UP and DOWN arrow keys to navigate the different speeds, then press ENTER to change to the selected speed.

Click on this button to mute or unmute this video or press UP or DOWN buttons to increase or decrease volume level.

Maximum Volume.

Video transcript

Start of transcript. Skip to the end.

- [Morgan] All right, I hope you're ready to dive in

and begin architecting solutions on AWS.

Let's call our first customer

and have a conversation about their workload.

(telephone ringing)

- [Raf] Hi there!

I am a technical architect here at Any Company Ecommerce,

happy to meet you.

- [Morgan] Hello there!

Nice to meet you too,

because we've clearly never met before.

My name is Morgan and I'm a Solutions Architect here at AWS,

and I hear you're trying to move a specific workload to AWS,

and are looking for some help on how to get there.

- [Raf] Yeah, yeah, that's true.

We've been working on moving our systems to AWS

over the last few years,

and we already have a decent amount of things on AWS,

but we have mostly just moved those workloads to AWS

without much you know, refactoring.

Now, we have an opportunity to completely rewrite

our next workload as part of a migration we are doing here.

- [Morgan] Oh, that's awesome.

I can definitely help you with that.

So, it sounds like before, you were using

a lift-and-shift model for migration,

whereas now, you're looking to use

more cloud-native technologies

so that you can better take advantage

of the cloud and its benefits.

Can you tell me more about your business

and the project that you want help with?

- [Raf] Yeah, absolutely.

So, we sell cleaning products all over the world,

and we have a website and an app open to the public,

where customers shop and place orders.

We also have a website for buyers

to purchase our products wholesale,

for resale in their own shops.

All of the orders from across these various frontends

come in to our orders service, hosted on premises,

that validates, authenticates, accepts,

orders, and processes, and stores the order in a database,

and then also makes calls downstream to other services,

like the inventory fulfillment and accounting service,

to reflect the confirmed order.

The inventory fulfillment and accounting services

are already hosted in the AWS Cloud,

and the orders service is the last major piece

that needs to be moved over.

- [Morgan] Okay, all right, I understand.

So, what about payment processing?

Is payment processing in scope for this project?

- [Raf] No, we use - we use a payment gateway service.

So, by the time this request hits the backend,

the payment will already be taken care of.

We just need to accept the order into our backend services,

perform some business logic,

and then send it to the downstream services

for further processing.

- [Morgan] Okay great, thanks for that context.

And how are you accepting a request for the services?

Do you have a web server running somewhere,

or are you hosting an API

that's the entry point for this?

What do you have going on?

- [Raf] Yeah, that's a good question.

Right now, we are using a web server

that is also hosted on the same server

with the application, to route requests

to the backend service.

- [Morgan] Okay, cool.

So, what about storing orders that come in?

What type of database are you using for those orders?

- [Raf] Oh, the database, right.

We are currently using MySQL database on-prem

for this service to store the order data.

The other downstream services

will work with their own data store.

So, we are open to changing databases,

but it's very important that

to be highly available and durable.

We only store the orders in one table,

so maintaining a whole MySQL instance

for this has been a bit of a time-consuming task

for the minimal data we store in the database.

I think, ideally, we would have a database in AWS

that is simpler to manage,

and it's a little bit of more hands off,

so we don't spend that much time maintaining it.

- [Morgan] Yeah, that makes sense.

It sounds like we can probably find

a simpler solution for you.

So, I'm hearing that you want to host a web backend on AWS,

as well as a database for the order service.

And I'm thinking we will try to go serverless

for this as much as possible,

given it sounds like you want to reduce

any operational overhead.

Can you tell me some more about the problems

you're trying to solve with this migration?

What types of pain points are driving your desire

to completely rewrite the service,

instead of just lifting and shifting it,

like you did with the other services?

- [Raf] Yeah, so most of the issues we run into for this app

come from the application layer,

because of the way all of this business logic

is all tied together in one order service.

When that service gets overloaded,

it can slow down, or even crash.

This means that orders are slow to get accepted,

and sometimes fail when the application becomes overwhelmed.

We have a hard time scaling on premises to meet our demands,

and we are looking for a solution with automatic scaling

that doesn't require time to set up or operate.

There is also an issue, where if the application crashes,

and a process is finishing up

making the calls to downstream APIs,

so let's say the first call to the inventory service

encounters is successful,

but then the application crashes, right?

This means that the application

doesn't get the chance to send the information

for successfully process orders

to the downstream services, like fulfillment and accounting.

So in a nutshell, most of our problems

come from not being able to scale quickly enough

to meet demand, and then, secondarily,

our components are too tightly coupled.

So if one API call fails,

then the next few ones don't get completed.

This causes inconsistency with our orders data

and causes headaches for our customers.

- [Morgan] Yeah okay, I got it.

I can definitely see how that would cause

some frustration, and why you would want to fix that.

So, I'm hearing that you want managed scaling in place,

and that you want this application

to be decoupled to increase resilience.

I also have, as a requirement,

that we need to come up with a solution

for sending the downstream calls to those other services,

independent of each other,

so that they don't impact the orders processing.

Now, what about demand for this application?

Is it a steady demand with low volume, or high volume?

Is it cyclical, or do you have a spiky demand?

- [Raf] Yeah, our app has pretty spiky demand.

When we launch online sales or send out coupon codes,

we see, like, huge traffic.

But then, at other times, we have essentially zero demand.

And as you know, it can be hard

to operate on demand like that

because we have to overprovision hardware

in the data centers, right?

We have to buy more servers and put it there.

- [Morgan] Cool, all right, so no problem.

We can try to come up with a solution here

that will scale up when you need it.

And then more importantly, scale back down

when demand goes down,

which, like you're saying, is harder to do on premises,

where you have to overprovision things all the time.

- [Raf] So, we are ready to begin planning

to move these orders service to AWS.

And as part of this process,

we are open to rewrite the codes

to go cloud native, whenever possible.

I think that going cloud native would let us

take advantage of the benefits offered by the cloud.

We are also hoping that,

as part of this migration and refactoring,

we will be able to fix some of the issues

we have been encountering in the code.

- [Morgan] Yep, I think I understand

what you're looking for.

Now beyond all that, do you have any other requirements

for this that you have laid out already?

- [Raf] Yes, I do.

This is something we have been spending

a lot of time talking about internally, here, in meetings.

I think that, at this point, we have covered most things,

but something specific, is that we want

monitoring and logging to be easy to put in place.

And ideally, we want everything

to use the same logging system.

- [Morgan] Okay great, so are there

any other aspects of this

that you're trying to optimize?

- [Raf] Mm, yeah.

One more thing I can think of right now,

is that we want to try to optimize this solution

for cost and performance as much as possible.

- [Morgan] Wow, all right, so you are super prepared.

- [Raf] Thank you. (laughing) - [Morgan] That's awesome.

I have all of my notes up here and I'm going to take those

and go digest this information.

And I'll likely be calling you

for more information as needed as I have questions

as I'm designing this.

I hope that's alright.

- [Raf] Absolutely yeah, so thank you for helping us

designing something that meets our needs.

Looking forward to see what you come with.

Call me anytime if you have any questions, okay?

- [Morgan] All right, thanks.

- [Raf] All right, bye-bye.

End of transcript. Skip to the start.

Downloads and transcripts

Video

Share this video

Share on twitter Share on facebook Share on linkedin

https://courses.edx.org/videos/block-v1:AWS+AWS-ARCH-1+3T2022+type@video+block@8

Start of transcript. Skip to the end.

- [Morgan] Okay, so we've just heard from our customer,

and there are a few key points I want to discuss

before we begin designing this architecture.

The customer stated that they're moving

from an application hosted on premises

to a service-oriented application hosted on AWS.

Some services have already been migrated to AWS,

and they're looking for advice on their order service,

which currently includes a couple of different functions,

and is acting as one monolithic code base.

Based on my notes,

this is what their on-premises solution

looks like right now.

It's good to keep this architecture in mind as we go along,

so that we don't forget anything.

Now, a big takeaway I heard from the customer,

is that they want this service to be more reliable

and resilient, so that if one part of the application fails,

it doesn't impact the other processes running downstream.

The current problem is that the orders acceptance

and downstream calls are all handled

by one code package as a monolith,

and we need to break that into multiple components,

and have them loosely coupled to avoid having

a single point of failure, like they have now.

Now, for the requirements that we gathered.

First off, managed scaling

for compute and database components, when possible,

is going to be important.

This one makes me immediately think that we will

be using serverless services as much as possible.

Serverless services scale in and out with demand

on their own, without the customer needing to step in.

For some services, you can set parameters

around how scaling happens, or set limits to scaling,

but it's totally managed behind the scenes.

So, I'm going to keep serverless top of mind

as we go through the different potential components,

which, spoiler alert, means I probably won't

be looking at EC2 a lot for this solution.

Now, for the next requirement,

we have decoupling components to maximize resilience.

This one is really about eliminating

single points of failure for the app.

I'm thinking of breaking the order service down

into multiple components, and following

an event-driven architecture,

so we can more loosely couple each piece.

What I mean by that, is that the order acceptance,

order processing, and then the downstream calls,

should all be independent of each other,

so that if one has a hiccup or fails,

it won't take down everything else.

We'll keep talking about this throughout the week.

Another requirement from the customer is that they wanted

centralized monitoring and logging.

They want to be able to see the state

of their application all in one place,

and have a centralized point to come to

for debugging or troubleshooting.

This won't be a problem,

especially with the serverless services,

which all tend to be integrated with Amazon CloudWatch

and Amazon CloudWatch Logs.

Then finally, there's our last requirement,

which is all about optimizing for cost,

performance efficiency, and operational overhead.

Cost is going to be one of the main drivers

for what services I choose, and then, secondarily,

performance efficiency and operational overhead.

It can be difficult to balance these things.

But again, serverless really tends to optimize

in the direction of these three things anyways,

so I'm not too concerned here.

We will build out the architecture first, piece by piece,

put it all together,

and then circle back and talk about the different choices

that we can make to optimize for cost and performance.

End of transcript. Skip to the start.

मुख्य सामग्री पर जाएँ ग्राहक #1: उपयोग मामला और आवश्यकताएँ 8:09 / 8:09 स्पीड मेनू में प्रवेश करने के लिए UP दबाएँ, फिर अलग-अलग गति को नेविगेट करने के लिए UP और DOWN तीर कुंजियों का उपयोग करें, फिर चयनित गति में बदलने के लिए ENTER दबाएँ। इस वीडियो को म्यूट या अनम्यूट करने के लिए इस बटन पर क्लिक करें या वॉल्यूम स्तर को बढ़ाने या घटाने के लिए UP या DOWN बटन दबाएँ। अधिकतम वॉल्यूम। वीडियो ट्रांसक्रिप्ट ट्रांसक्रिप्ट की शुरुआत। अंत पर जाएँ। - [मॉर्गन] ठीक है, मुझे आशा है कि आप इसमें गोता लगाने के लिए तैयार हैं और AWS पर समाधान तैयार करना शुरू करें। आइए अपने पहले ग्राहक को कॉल करें और उनके कार्यभार के बारे में बातचीत करें। (टेलीफोन बज रहा है) - [राफ] नमस्ते! मैं यहाँ Any Company Ecommerce में एक तकनीकी आर्किटेक्ट हूँ, आपसे मिलकर खुशी हुई। - [मॉर्गन] नमस्ते! आपसे मिलकर मुझे भी अच्छा लगा, क्योंकि हम स्पष्ट रूप से पहले कभी नहीं मिले हैं। मेरा नाम मॉर्गन है और मैं AWS में एक सॉल्यूशन आर्किटेक्ट हूँ, और मैंने सुना है कि आप एक खास कार्यभार को AWS में स्थानांतरित करने का प्रयास कर रहे हैं, और वहाँ पहुँचने के तरीके के बारे में कुछ मदद की तलाश कर रहे हैं। - [राफ़] हाँ, हाँ, यह सच है। हम पिछले कुछ वर्षों से अपने सिस्टम को AWS में स्थानांतरित करने पर काम कर रहे हैं, और हमारे पास पहले से ही AWS पर काफी चीज़ें हैं, लेकिन हमने ज़्यादातर कार्यभार को बिना ज़्यादा रिफ़ैक्टरिंग के AWS में स्थानांतरित कर दिया है। अब, हमारे पास यहाँ किए जा रहे माइग्रेशन के हिस्से के रूप में अपने अगले कार्यभार को पूरी तरह से फिर से लिखने का अवसर है। - [मॉर्गन] ओह, यह बहुत बढ़िया है। मैं निश्चित रूप से इसमें आपकी मदद कर सकता हूँ। तो, ऐसा लगता है कि पहले, आप माइग्रेशन के लिए लिफ्ट-एंड-शिफ्ट मॉडल का उपयोग कर रहे थे, जबकि अब, आप अधिक क्लाउड-नेटिव तकनीकों का उपयोग करना चाहते हैं ताकि आप क्लाउड और इसके लाभों का बेहतर लाभ उठा सकें। क्या आप मुझे अपने व्यवसाय और उस प्रोजेक्ट के बारे में अधिक बता सकते हैं जिसके लिए आप मदद चाहते हैं? - [राफ़] हाँ, बिल्कुल। तो, हम पूरी दुनिया में सफाई उत्पाद बेचते हैं, और हमारे पास एक वेबसाइट और एक ऐप है जो जनता के लिए खुला है, जहाँ ग्राहक खरीदारी करते हैं और ऑर्डर देते हैं। हमारे पास खरीदारों के लिए एक वेबसाइट भी है जो हमारे उत्पादों को थोक में खरीदते हैं, और अपनी दुकानों में पुनर्विक्रय करते हैं। इन विभिन्न फ्रंटएंड से सभी ऑर्डर हमारे ऑर्डर सेवा में आते हैं, जो परिसर में होस्ट की जाती है, जो पुष्टि करती है, प्रमाणित करती है, स्वीकार करती है, ऑर्डर करती है, और प्रक्रिया करती है, और डेटाबेस में ऑर्डर संग्रहीत करती है, और फिर पुष्टि किए गए ऑर्डर को दर्शाने के लिए इन्वेंट्री पूर्ति और लेखा सेवा जैसी अन्य सेवाओं को डाउनस्ट्रीम कॉल भी करती है। इन्वेंट्री पूर्ति और लेखा सेवाएँ पहले से ही AWS क्लाउड में होस्ट की गई हैं, और ऑर्डर सेवा अंतिम प्रमुख हिस्सा है जिसे स्थानांतरित करने की आवश्यकता है। - [मॉर्गन] ठीक है, ठीक है, मैं समझ गया। तो, भुगतान प्रसंस्करण के बारे में क्या? क्या इस परियोजना के लिए भुगतान प्रसंस्करण दायरे में है? - [राफ़] नहीं, हम - हम एक भुगतान गेटवे सेवा का उपयोग करते हैं। तो, जब तक यह अनुरोध बैकएंड पर पहुंचेगा, तब तक भुगतान का पहले ही ध्यान रखा जा चुका होगा। हमें बस अपनी बैकएंड सेवाओं में ऑर्डर स्वीकार करना होगा, कुछ व्यावसायिक तर्क निष्पादित करने होंगे, और फिर इसे आगे की प्रक्रिया के लिए डाउनस्ट्रीम सेवाओं को भेजना होगा। - [मॉर्गन] ठीक है बढ़िया, उस संदर्भ के लिए धन्यवाद। और आप सेवाओं के लिए अनुरोध कैसे स्वीकार कर रहे हैं? क्या आपके पास कहीं कोई वेब सर्वर चल रहा है, या आप कोई API होस्ट कर रहे हैं जो इसके लिए प्रवेश बिंदु है? आपके पास क्या चल रहा है? - [राफ] हाँ, यह एक अच्छा सवाल है। अभी, हम एक वेब सर्वर का उपयोग कर रहे हैं जो बैकएंड सेवा के लिए अनुरोधों को रूट करने के लिए एप्लिकेशन के साथ एक ही सर्वर पर होस्ट किया गया है। - [मॉर्गन] ठीक है, बढ़िया। तो, आने वाले ऑर्डर को संग्रहीत करने के बारे में क्या? आप उन ऑर्डर के लिए किस प्रकार का डेटाबेस उपयोग कर रहे हैं? - [राफ] ओह, डेटाबेस, ठीक है। हम वर्तमान में ऑर्डर डेटा संग्रहीत करने के लिए इस सेवा के लिए ऑन-प्रिमाइसेस MySQL डेटाबेस का उपयोग कर रहे हैं। अन्य डाउनस्ट्रीम सेवाएँ अपने स्वयं के डेटा स्टोर के साथ काम करेंगी। इसलिए, हम डेटाबेस बदलने के लिए तैयार हैं, लेकिन यह बहुत महत्वपूर्ण है कि यह अत्यधिक उपलब्ध और टिकाऊ हो। हम केवल एक टेबल में ऑर्डर स्टोर करते हैं, इसलिए इसके लिए एक संपूर्ण MySQL इंस्टेंस को बनाए रखना थोड़ा समय लेने वाला काम रहा है डेटाबेस में संग्रहीत न्यूनतम डेटा के लिए। मुझे लगता है, आदर्श रूप से, हमारे पास AWS में एक डेटाबेस होगा जिसे प्रबंधित करना आसान है, और यह थोड़ा और आसान है, इसलिए हमें इसे बनाए रखने में इतना समय नहीं लगाना पड़ता। - [मॉर्गन] हाँ, यह समझ में आता है। ऐसा लगता है कि हम शायद आपके लिए एक सरल समाधान पा सकते हैं। इसलिए, मैं सुन रहा हूँ कि आप AWS पर एक वेब बैकएंड होस्ट करना चाहते हैं, साथ ही ऑर्डर सेवा के लिए एक डेटाबेस भी। और मुझे लगता है कि हम इसके लिए यथासंभव सर्वरलेस होने की कोशिश करेंगे, यह देखते हुए कि ऐसा लगता है कि आप किसी भी परिचालन ओवरहेड को कम करना चाहते हैं। क्या आप मुझे उन समस्याओं के बारे में कुछ और बता सकते हैं जिन्हें आप इस माइग्रेशन के साथ हल करने की कोशिश कर रहे हैं? किस तरह की परेशानियाँ आपको सेवा को पूरी तरह से फिर से लिखने की इच्छा को प्रेरित कर रही हैं,

क्या आप इसे उठाकर दूसरी जगह शिफ्ट करना चाहते हैं, जैसा कि आपने दूसरी सेवाओं के साथ किया था? - [राफ] हाँ, तो इस ऐप के लिए हमारे सामने आने वाली ज़्यादातर समस्याएँ एप्लिकेशन लेयर से आती हैं, क्योंकि इस तरह से सभी व्यावसायिक तर्क एक ऑर्डर सेवा में एक साथ बंधे होते हैं। जब वह सेवा ओवरलोड हो जाती है, तो यह धीमी हो सकती है या क्रैश भी हो सकती है। इसका मतलब है कि ऑर्डर स्वीकार होने में धीमे होते हैं और कभी-कभी जब एप्लिकेशन ओवरलोड हो जाता है तो विफल हो जाते हैं। हमें अपनी मांगों को पूरा करने के लिए परिसर में स्केलिंग करने में कठिनाई होती है और हम स्वचालित स्केलिंग के साथ एक समाधान की तलाश कर रहे हैं जिसे सेट अप या संचालन के लिए समय की आवश्यकता नहीं होती है। एक समस्या यह भी है कि अगर एप्लिकेशन क्रैश हो जाता है और डाउनस्ट्रीम API को कॉल करने की प्रक्रिया समाप्त हो रही है, तो मान लें कि इन्वेंट्री सेवा के लिए पहला कॉल सफल होता है, लेकिन फिर एप्लिकेशन क्रैश हो जाता है, है न? इसका मतलब है कि एप्लिकेशन को ऑर्डर को सफलतापूर्वक प्रोसेस करने के लिए डाउनस्ट्रीम सेवाओं, जैसे कि पूर्ति और लेखांकन, को जानकारी भेजने का मौका नहीं मिलता है। तो संक्षेप में, हमारी अधिकांश समस्याएँ मांग को पूरा करने के लिए पर्याप्त तेज़ी से स्केल न कर पाने से आती हैं, और फिर, दूसरी बात, हमारे घटक बहुत कसकर युग्मित हैं। इसलिए यदि एक API कॉल विफल हो जाती है, तो अगले कुछ पूर्ण नहीं होते। इससे हमारे ऑर्डर डेटा में असंगति होती है और हमारे ग्राहकों के लिए सिरदर्द पैदा होता है। - [मॉर्गन] हाँ ठीक है, मैं समझ गया। मैं निश्चित रूप से देख सकता हूँ कि इससे कुछ निराशा कैसे होगी, और आप इसे क्यों ठीक करना चाहेंगे। तो, मैं सुन रहा हूँ कि आप प्रबंधित स्केलिंग चाहते हैं, और आप चाहते हैं कि इस एप्लिकेशन को लचीलापन बढ़ाने के लिए अलग किया जाए। मेरी एक आवश्यकता यह भी है कि हमें उन अन्य सेवाओं को डाउनस्ट्रीम कॉल भेजने के लिए एक समाधान के साथ आने की आवश्यकता है, एक दूसरे से स्वतंत्र, ताकि वे ऑर्डर प्रोसेसिंग को प्रभावित न करें। अब, इस एप्लिकेशन की मांग के बारे में क्या? क्या यह कम मात्रा के साथ एक स्थिर मांग है, या उच्च मात्रा? क्या यह चक्रीय है, या क्या आपके पास एक नुकीला मांग है? - [राफ] हाँ, हमारे ऐप की मांग बहुत ज़्यादा है। जब हम ऑनलाइन बिक्री शुरू करते हैं या कूपन कोड भेजते हैं, तो हमें बहुत ज़्यादा ट्रैफ़िक मिलता है। लेकिन, दूसरी बार, हमारे पास बिल्कुल भी मांग नहीं होती। और जैसा कि आप जानते हैं, इस तरह की मांग पर काम करना मुश्किल हो सकता है क्योंकि हमें डेटा सेंटर में हार्डवेयर को ज़्यादा प्रोविज़न करना पड़ता है, है न? हमें और सर्वर खरीदने होंगे और उन्हें वहाँ लगाना होगा। - [मॉर्गन] बढ़िया, ठीक है, तो कोई समस्या नहीं है। हम यहाँ एक समाधान निकालने की कोशिश कर सकते हैं जो ज़रूरत पड़ने पर स्केल अप करेगा। और फिर सबसे महत्वपूर्ण बात, जब मांग कम हो जाए, तो स्केल डाउन करना, जो, जैसा कि आप कह रहे हैं, परिसर में करना मुश्किल है, जहाँ आपको हर समय चीज़ों को ज़्यादा प्रोविज़न करना पड़ता है। - [राफ] तो, हम इन ऑर्डर सेवा को AWS में स्थानांतरित करने की योजना बनाने के लिए तैयार हैं। और इस प्रक्रिया के हिस्से के रूप में, जब भी संभव हो, हम क्लाउड नेटिव जाने के लिए कोड को फिर से लिखने के लिए तैयार हैं। मुझे लगता है कि क्लाउड नेटिव जाने से हम क्लाउड द्वारा दिए जाने वाले लाभों का लाभ उठा पाएँगे। हम यह भी उम्मीद कर रहे हैं कि, इस माइग्रेशन और रीफैक्टरिंग के हिस्से के रूप में, हम कोड में आने वाली कुछ समस्याओं को ठीक कर पाएंगे। - [मॉर्गन] हाँ, मुझे लगता है कि मैं समझ गया हूँ कि आप क्या चाहते हैं। अब इन सबसे परे, क्या आपके पास इसके लिए कोई अन्य आवश्यकताएँ हैं जो आपने पहले ही निर्धारित कर दी हैं? - [राफ़] हाँ, मेरे पास हैं। यह कुछ ऐसा है जिसके बारे में हम यहाँ बैठकों में आंतरिक रूप से बात करने में बहुत समय बिता रहे हैं। मुझे लगता है कि, इस बिंदु पर, हमने अधिकांश चीजों को कवर कर लिया है, लेकिन एक खास बात यह है कि हम चाहते हैं कि निगरानी और लॉगिंग को लागू करना आसान हो। और आदर्श रूप से, हम चाहते हैं कि सभी चीजें एक ही लॉगिंग सिस्टम का उपयोग करें। - [मॉर्गन] ठीक है बढ़िया, तो क्या इसके कोई अन्य पहलू हैं जिन्हें आप अनुकूलित करने का प्रयास कर रहे हैं? - [राफ़] हम्म, हाँ। एक और बात जो मैं अभी सोच सकता हूँ, वह यह है कि हम इस समाधान को लागत और प्रदर्शन के लिए यथासंभव अनुकूलित करने का प्रयास करना चाहते हैं। - [मॉर्गन] वाह, ठीक है, तो आप बहुत अच्छी तरह से तैयार हैं। - [राफ] धन्यवाद। (हँसते हुए) - [मॉर्गन] यह बहुत बढ़िया है। मेरे पास यहाँ मेरे सभी नोट्स हैं और मैं उन्हें लेकर जा रहा हूँ और इस जानकारी को पचाने जा रहा हूँ। और मैं संभवतः आपको अधिक जानकारी के लिए कॉल करूँगा क्योंकि मेरे पास प्रश्न हैं क्योंकि मैं इसे डिज़ाइन कर रहा हूँ। मुझे उम्मीद है कि यह ठीक है। - [राफ] बिल्कुल हाँ, तो हमारी ज़रूरतों को पूरा करने वाली चीज़ को डिज़ाइन करने में हमारी मदद करने के लिए धन्यवाद। यह देखने के लिए उत्सुक हूँ कि आप क्या लेकर आते हैं। अगर आपके कोई प्रश्न हों तो मुझे कभी भी कॉल करें, ठीक है? - [मॉर्गन] ठीक है, धन्यवाद। - [राफ] ठीक है, अलविदा। प्रतिलेख समाप्त। शुरुआत पर जाएँ। डाउनलोड और ट्रांसक्रिप्ट वीडियो इस वीडियो को शेयर करें ट्विटर पर शेयर करें फेसबुक पर शेयर करें लिंक्डइन पर शेयर करें https://courses.edx.org/videos/block-v1:AWS+AWS-ARCH-1+3T2022+type@video+block@8

kya aap ise uthaakar doosaree jagah shipht kar

ट्रांसक्रिप्ट की शुरुआत। अंत पर जाएँ। - [मॉर्गन] ठीक है, तो हमने अभी-अभी अपने ग्राहक से सुना है, और कुछ मुख्य बिंदु हैं जिन पर मैं चर्चा करना चाहता हूँ इससे पहले कि हम इस आर्किटेक्चर को डिज़ाइन करना शुरू करें। ग्राहक ने कहा कि वे परिसर में होस्ट किए गए एप्लिकेशन से AWS पर होस्ट किए गए सेवा-उन्मुख एप्लिकेशन में जा रहे हैं। कुछ सेवाएँ पहले ही AWS में माइग्रेट हो चुकी हैं, और वे अपनी ऑर्डर सेवा पर सलाह की तलाश कर रहे हैं, जिसमें वर्तमान में कुछ अलग-अलग फ़ंक्शन शामिल हैं, और यह एक मोनोलिथिक कोड बेस के रूप में कार्य कर रहा है। मेरे नोट्स के आधार पर, यह उनका ऑन-प्रिमाइसेस समाधान है अभी ऐसा दिखता है। इस आर्किटेक्चर को ध्यान में रखना अच्छा है, ताकि हम कुछ भी न भूलें। अब, ग्राहक से मैंने जो बड़ी बात सुनी, वह यह है कि वे चाहते हैं कि यह सेवा अधिक विश्वसनीय हो और लचीली हो, ताकि यदि एप्लिकेशन का एक भाग विफल हो जाए, तो यह डाउनस्ट्रीम चल रही अन्य प्रक्रियाओं को प्रभावित न करे। वर्तमान समस्या यह है कि ऑर्डर स्वीकृति और डाउनस्ट्रीम कॉल सभी को एक कोड पैकेज द्वारा मोनोलिथ के रूप में नियंत्रित किया जाता है, और हमें इसे कई घटकों में विभाजित करने की आवश्यकता है, और उन्हें एकल विफलता बिंदु से बचने के लिए शिथिल रूप से युग्मित करना होगा, जैसा कि अब है। अब, उन आवश्यकताओं के लिए जिन्हें हमने एकत्र किया है। सबसे पहले, जब संभव हो, तो कंप्यूट और डेटाबेस घटकों के लिए प्रबंधित स्केलिंग महत्वपूर्ण होने जा रही है। यह मुझे तुरंत सोचने पर मजबूर करता है कि हम यथासंभव सर्वरलेस सेवाओं का उपयोग करेंगे। सर्वरलेस सेवाएँ मांग के साथ अपने आप स्केल इन और आउट होती हैं, बिना ग्राहक के हस्तक्षेप के। कुछ सेवाओं के लिए, आप स्केलिंग कैसे होती है, इसके बारे में पैरामीटर सेट कर सकते हैं या स्केलिंग की सीमाएँ निर्धारित कर सकते हैं, लेकिन यह पूरी तरह से पर्दे के पीछे प्रबंधित होता है। इसलिए, मैं सर्वरलेस को ध्यान में रखूँगा क्योंकि हम विभिन्न संभावित घटकों से गुज़रते हैं, जिसका मतलब है कि मैं शायद इस समाधान के लिए EC2 पर बहुत अधिक ध्यान नहीं दूँगा। अब, अगली आवश्यकता के लिए, हमारे पास लचीलेपन को अधिकतम करने के लिए घटकों को अलग करना है। यह वास्तव में ऐप के लिए विफलता के एकल बिंदुओं को समाप्त करने के बारे में है। मैं ऑर्डर सेवा को कई घटकों में विभाजित करने और एक ईवेंट-संचालित आर्किटेक्चर का पालन करने के बारे में सोच रहा हूं, ताकि हम प्रत्येक भाग को अधिक शिथिल रूप से जोड़ सकें। मेरा मतलब यह है कि ऑर्डर स्वीकृति, ऑर्डर प्रोसेसिंग और फिर डाउनस्ट्रीम कॉल, सभी एक दूसरे से स्वतंत्र होने चाहिए, ताकि अगर किसी में कोई रुकावट आती है या विफल हो जाती है, तो यह बाकी सब को बंद न कर दे। हम इस बारे में पूरे सप्ताह बात करते रहेंगे। ग्राहक की एक और आवश्यकता यह है कि वे केंद्रीकृत निगरानी और लॉगिंग चाहते हैं। वे अपने एप्लिकेशन की स्थिति को एक ही स्थान पर देखना चाहते हैं और डिबगिंग या समस्या निवारण के लिए एक केंद्रीकृत बिंदु चाहते हैं। यह कोई समस्या नहीं होगी, खासकर सर्वर रहित सेवाओं के साथ, जो सभी Amazon CloudWatch और Amazon CloudWatch लॉग के साथ एकीकृत होते हैं। फिर अंत में, हमारी अंतिम आवश्यकता है, जो लागत, प्रदर्शन दक्षता और परिचालन ओवरहेड के लिए अनुकूलन के बारे में है। लागत मेरे द्वारा चुनी जाने वाली सेवाओं के लिए मुख्य चालकों में से एक होगी, और फिर, दूसरे स्थान पर, प्रदर्शन दक्षता और परिचालन ओवरहेड। इन चीजों को संतुलित करना मुश्किल हो सकता है। लेकिन फिर से, सर्वरलेस वास्तव में इन तीन चीजों की दिशा में अनुकूलन करता है, इसलिए मैं यहां बहुत चिंतित नहीं हूं। हम पहले आर्किटेक्चर का निर्माण करेंगे, टुकड़े-टुकड़े करके, इसे एक साथ रखेंगे, और फिर वापस आकर विभिन्न विकल्पों के बारे में बात करेंगे जिन्हें हम लागत और प्रदर्शन के लिए अनुकूलित कर सकते हैं। ट्रांसक्रिप्ट का अंत। शुरुआत पर जाएं।

- [Morgan] Let's take a closer look at the order service

that is currently running on premises,

and think about it from the perspective of compute.

If we pull up the architecture diagram

for the existing solution,

we need to host this code somewhere using a compute service.

So, we will tackle that first.

This order service is currently running

as a single code package,

and is running across multiple virtual machines

in the customer's data center.

We want to break this up into smaller pieces,

so we can decouple each component in the application.

And we also want to pick a compute service

that will optimize for automated scaling,

high performance, and reduced operational overhead.

So right away, I'm thinking, let's go ahead

and bring up the AWS webpage for compute,

and take a look at the different compute services

that we have available to us.

AWS is always changing, so it's a good idea to take a look

and see what is out there,

so that we are aware of all of our options.

So we've navigated to aws.amazon.com/products/compute,

and I'm going to go ahead and scroll down here,

to this handy table that lays out

the different compute services from a high level,

and tells us what the use cases are.

So we can see here, we have some different categories.

We have the first category, which is instances,

and under this one, you can see we have EC2 listed.

And we also have services like Lightsail,

and services like Batch.

Now, with these services,

these are going to have instances,

which meaning that they are not serverless services.

So things like EC2,

that's our virtual machines on the cloud.

We have Lightsail, which is basically an easier way

to use EC2 for specific use cases.

And then we have AWS Batch, which is for batch processes.

So, we can see already that each individual compute service

is meant for a specific purpose.

So now, if we keep scrolling down this page,

you can see that the next category

that we have is for containers.

And under containers, we have a couple of different

AWS services here, as well.

You can see we have Amazon Elastic Container Service,

which is a container orchestration tool.

We have Amazon Elastic Kubernetes Service,

and then we also have the compute platform, AWS Fargate.

So, these container services are pretty interesting.

I think that these could potentially be good

for our customer, but let's go ahead

and keep scrolling down here.

Then, you can see we also have,

under the serverless category,

which this category is particularly interesting

for our customer, knowing that we want to use serverless.

We have AWS Lambda listed, here.

So, it's just a good idea to know where to go

to look for information about AWS.

So, we're going to be pulling up these webpages

throughout the course, so that way, you understand

where you can go to find more information.

Now, before we discuss some of these serverless options,

I want to talk about why I am not considering EC2.

A lot of the time,

people look to EC2 as the default option.

You can spin up virtual machines on demand,

and you can choose your OS,

and take full control over what happens on your instance.

And this is all great,

and when I think about the customer's application,

we could, of course, lift and shift

the entire order service directly onto EC2.

A solution could use EC2 and work just fine.

Now, here's the important part.

Just because a solution could work,

doesn't mean that you should stop there and call it a day.

There can be many potential solutions to a problem,

and we should evaluate them all

and see which one is the best fit, not which one is a fit.

That being said, and knowing we want to make this

a serverless solution

because of the requirements our customer gave to us,

let's first take a look at the AWS container services.

For this category, we have Amazon ECS and Amazon EKS,

both of which can be run on Amazon EC2 or on AWS Fargate,

which is a serverless compute platform for containers.

Running the orders application and containers

on a serverless compute platform like AWS Fargate

could be a good pick, here.

The scaling is totally managed.

Logs and metrics go to CloudWatch.

Performance efficiency is high.

Operational overhead is low.

Costs can be kept low, as well,

using things like AWS Fargate Spot.

Everything is good.

The only thing is,

I don't know if our customer is interested

in using container technology,

which could be a blocker for this option.

Now, the next compute option

I want to consider is AWS Lambda.

Lambda is cloud native,

meaning that it was built specifically for the cloud,

and ther

efore was built to take advantage

of all of the benefits that the cloud has to offer.

AWS Lambda is totally serverless,

and scales up and down automatically

without a lot of operational overhead.

You may have to tweak some configurations over time,

depending on your workload, and there are things you can do

to make scaling happen faster and make your code run faster.

But overall, the operational overhead

with Lambda is significantly lower

when you compare it with options like EC2.

When using AWS Lambda, each request or event that comes in

spins up its own instance of a Lambda function.

The Lambda function runs in a microVM

powered by something called Firecracker,

and this microVM will shut down on its own

after a period of time.

So, if you have no demand,

you won't be dealing with idling instances

of your application code running.

It only runs when it needs to run.

Just like AWS Fargate, scaling is managed,

operational overhead is low, performance efficiency is high.

Let's go ahead and call our customer

and ask them what they think

about using containers or AWS Lambda.

(phone ringing)

- [Raf] Hello, how are you doing, Morgan?

- [Morgan] Hello there.

I'm doing great.

I'm looking at what compute services to use

for your order service,

and I ran into a question

that I need you to answer.

- [Raf] Sure, yeah, yeah.

What's up?

- [Morgan] Well, I am choosing between using AWS Lambda

or AWS Fargate for hosting

your order service application code.

The thing is, AWS Fargate requires

that you use containers and a container orchestration tool,

like Amazon ECS or Amazon EKS.

So my questions are,

are you familiar with these services?

And are you open to using container services

to host this workload?

- [Raf] So yeah, I am familiar actually,

from a high level, at least.

I do know we don't currently use containers at our company.

We don't really have the skill set

to manage containers in-house right now.

Our developers and DevOps are already beginning to learn

the basics for cloud-native stuff, like Lambda and DynamoDB,

since we have already started some migrations.

I think we might be using Lambda

in different parts of the business, now that you said.

- [Morgan] Well, all right, I think

that's all I needed to know.

Thank you.

- [Raf] Yeah, sure.

Thanks, anytime.

- [Morgan] Okay, so that is really good information

to gather from our customer.

Now, I know that Amazon ECS and AWS Fargate

are not really going to be the best choice for this,

even though from a technical perspective,

it would work fine.

That being said, let's go ahead

and add our first block to this architecture.

AWS Lambda.

Lambda will host the application code for the order service.

This will require a rewrite of the code,

so that Lambda can invoke it on request,

but I don't think that's going to be an issue.

When you use Lambda, you create what is called a function,

and then you upload your code,

you select a trigger for the function to run.

A trigger can be some sort of event,

perhaps coming from another AWS service

or an external event, like an HTTP request

through an Application Load Balancer,

or coming through something like Amazon API Gateway.

API Gateway is a natural choice

to front the Lambda function for our customer's use case.

This means that they will no longer need to maintain

a web server to accept the requests from the clients.

Instead, you can define an API in API Gateway,

and set off the HTTP methods and resources

to forward the incoming requests to the backend.

API Gateway can do a lot more for us, as well.

For example, according to my notes from the customer call,

the orders processing service is currently doing

authentication on the request, and basic validation

that the data is in the right format.

API Gateway can handle these sorts of things for you.

So that means less code has to be written to do those tasks.

And I can actually picture our customer reacting

to writing less code for this migration.

I think that they are going to love that.

Logs and metrics from both Lambda and API Gateway

can be easily sent to Amazon CloudWatch

and Amazon CloudWatch Logs,

which can then be analyzed in one central place

across the instances of your function.

Both API Gateway and AWS Lambda

have managed scaling as well,

which are good choices for cost effectiveness,

and have high performance efficiency.

This seems to meet all of our requirements,

and doesn't involve introducing a new technology

into our customer's stack,

like containerizing their applications.

So, that being said,

time to add the second block to

our architecture, here.

API Gateway is going to front

our compute choice, AWS Lambda.

We are going to add more blocks to this setup

as we go throughout the week, to complete this architecture.

We're just getting started with this solution,

so let's keep going.

End of transcript. Skip to the start.

- [मॉर्गन] आइए वर्तमान में परिसर में चल रही ऑर्डर सेवा पर करीब से नज़र डालें, और इसके बारे में कंप्यूट के नज़रिए से सोचें। अगर हम मौजूदा समाधान के लिए आर्किटेक्चर आरेख खींचते हैं, तो हमें कंप्यूट सेवा का उपयोग करके इस कोड को कहीं होस्ट करने की आवश्यकता है। इसलिए, हम पहले उससे निपटेंगे। यह ऑर्डर सेवा वर्तमान में एकल कोड पैकेज के रूप में चल रही है, और ग्राहक के डेटा सेंटर में कई वर्चुअल मशीनों पर चल रही है। हम इसे छोटे-छोटे टुकड़ों में तोड़ना चाहते हैं, ताकि हम एप्लिकेशन में प्रत्येक घटक को अलग कर सकें। और हम एक ऐसी कंप्यूट सेवा भी चुनना चाहते हैं जो स्वचालित स्केलिंग, उच्च प्रदर्शन और कम परिचालन ओवरहेड के लिए अनुकूल हो। तो अभी, मैं सोच रहा हूँ, चलो आगे बढ़ते हैं और कंप्यूट के लिए AWS वेबपेज लाते हैं, और हमारे पास उपलब्ध विभिन्न कंप्यूट सेवाओं पर एक नज़र डालते हैं। AWS हमेशा बदलता रहता है, इसलिए यह देखना अच्छा विचार है कि वहाँ क्या है, ताकि हम अपने सभी विकल्पों से अवगत हों। इसलिए हमने aws.amazon.com/products/compute पर नेविगेट किया है, और मैं आगे जाकर नीचे स्क्रॉल करने जा रहा हूँ, इस आसान तालिका पर जो उच्च स्तर से विभिन्न कंप्यूट सेवाओं को प्रस्तुत करती है, और हमें बताती है कि उपयोग के मामले क्या हैं। तो हम यहाँ देख सकते हैं, हमारे पास कुछ अलग-अलग श्रेणियाँ हैं। हमारे पास पहली श्रेणी है, जो इंस्टेंस है, और इसके अंतर्गत, आप देख सकते हैं कि हमारे पास EC2 सूचीबद्ध है। और हमारे पास लाइटसेल जैसी सेवाएँ भी हैं, और बैच जैसी सेवाएँ भी हैं। अब, इन सेवाओं के साथ, इनमें इंस्टेंस होंगे, जिसका अर्थ है कि वे सर्वर रहित सेवाएँ नहीं हैं। तो EC2 जैसी चीजें, ये क्लाउड पर हमारी वर्चुअल मशीनें हैं। हमारे पास लाइटसेल है, जो मूल रूप से विशिष्ट उपयोग के मामलों के लिए EC2 का उपयोग करने का एक आसान तरीका है। और फिर हमारे पास AWS बैच है, जो बैच प्रक्रियाओं के लिए है। तो, हम पहले से ही देख सकते हैं कि प्रत्येक व्यक्तिगत कंप्यूट सेवा एक विशिष्ट उद्देश्य के लिए है। तो अब, अगर हम इस पेज को नीचे स्क्रॉल करते रहें, तो आप देख सकते हैं कि अगली श्रेणी जो हमारे पास है वह कंटेनरों के लिए है। और कंटेनरों के अंतर्गत, हमारे पास यहाँ कुछ अलग-अलग AWS सेवाएँ भी हैं। आप देख सकते हैं कि हमारे पास Amazon Elastic Container Service है, जो एक कंटेनर ऑर्केस्ट्रेशन टूल है। हमारे पास Amazon Elastic Kubernetes Service है, और फिर हमारे पास कंप्यूट प्लेटफ़ॉर्म, AWS Fargate भी है। तो, ये कंटेनर सेवाएँ बहुत दिलचस्प हैं। मुझे लगता है कि ये हमारे ग्राहक के लिए संभावित रूप से अच्छी हो सकती हैं, लेकिन चलिए आगे बढ़ते हैं और यहाँ नीचे स्क्रॉल करते रहते हैं। फिर, आप देख सकते हैं कि हमारे पास सर्वरलेस श्रेणी के अंतर्गत भी है, जो कि हमारे ग्राहक के लिए विशेष रूप से दिलचस्प है, यह जानते हुए कि हम सर्वरलेस का उपयोग करना चाहते हैं। हमारे पास यहाँ AWS Lambda सूचीबद्ध है। तो, यह जानना एक अच्छा विचार है कि AWS के बारे में जानकारी कहाँ देखें। तो, हम पूरे कोर्स में इन वेबपेजों को खोलते रहेंगे, ताकि आप समझ सकें कि आप अधिक जानकारी पाने के लिए कहाँ जा सकते हैं। अब, इन सर्वर रहित विकल्पों में से कुछ पर चर्चा करने से पहले, मैं इस बारे में बात करना चाहता हूँ कि मैं EC2 पर विचार क्यों नहीं कर रहा हूँ। बहुत बार, लोग EC2 को डिफ़ॉल्ट विकल्प के रूप में देखते हैं। आप मांग पर वर्चुअल मशीन बना सकते हैं, और आप अपना OS चुन सकते हैं, और अपने इंस्टेंस पर होने वाली हर चीज़ पर पूरा नियंत्रण रख सकते हैं। और यह सब बढ़िया है, और जब मैं ग्राहक के आवेदन के बारे में सोचता हूँ, तो हम निश्चित रूप से, पूरी ऑर्डर सेवा को सीधे EC2 पर ले जा सकते हैं। एक समाधान EC2 का उपयोग कर सकता है और ठीक से काम कर सकता है। अब, यहाँ महत्वपूर्ण हिस्सा है। सिर्फ़ इसलिए कि कोई समाधान काम कर सकता है, इसका मतलब यह नहीं है कि आपको वहीं रुक जाना चाहिए और इसे एक दिन के लिए टाल देना चाहिए। किसी समस्या के कई संभावित समाधान हो सकते हैं, और हमें उन सभी का मूल्यांकन करना चाहिए और देखना चाहिए कि कौन सा सबसे उपयुक्त है, न कि कौन सा उपयुक्त है। यह कहने के बाद, और यह जानते हुए कि हम इसे सर्वर रहित समाधान बनाना चाहते हैं क्योंकि हमारे ग्राहक ने हमें जो आवश्यकताएँ दी हैं, तो सबसे पहले AWS कंटेनर सेवाओं पर एक नज़र डालते हैं। इस श्रेणी के लिए, हमारे पास Amazon ECS और Amazon EKS हैं, जिनमें से दोनों को Amazon EC2 या AWS Fargate पर चलाया जा सकता है, जो कंटेनरों के लिए एक सर्वर रहित कंप्यूट प्लेटफ़ॉर्म है। ऑर्डर एप्लिकेशन और कंटेनरों को AWS Fargate जैसे सर्वर रहित कंप्यूट प्लेटफ़ॉर्म पर चलाना यहाँ एक अच्छा विकल्प हो सकता है। स्केलिंग पूरी तरह से प्रबंधित है। लॉग और मेट्रिक्स CloudWatch पर जाते हैं। प्रदर्शन दक्षता उच्च है। ऑपरेशनल ओवरहेड कम है। लागत को भी कम रखा जा सकता है, AWS Fargate Spot जैसी चीज़ों का उपयोग करके। सब कुछ ठीक है। केवल एक चीज़ है, मुझे नहीं पता कि हमारे ग्राहक कंटेनर तकनीक का उपयोग करने में रुचि रखते हैं या नहीं, जो इस विकल्प के लिए अवरोधक हो सकता है। अब, अगला कंप्यूट विकल्प जिस पर मैं विचार करना चाहता हूँ, वह है AWS Lambda। Lambda क्लाउड नेटिव है, जिसका अर्थ है कि इसे विशेष रूप से क्लाउड के लिए बनाया गया था, और वहाँ

- [morgan] aaie vartamaan mein

efore को क्लाउड द्वारा प्रदान किए जाने वाले सभी लाभों का लाभ उठाने के लिए बनाया गया था। AWS लैम्बडा पूरी तरह से सर्वर रहित है, और बहुत अधिक परिचालन ओवरहेड के बिना स्वचालित रूप से ऊपर और नीचे स्केल करता है। आपको अपने कार्यभार के आधार पर समय के साथ कुछ कॉन्फ़िगरेशन में बदलाव करना पड़ सकता है, और स्केलिंग को तेज़ बनाने और अपने कोड को तेज़ी से चलाने के लिए आप कुछ चीज़ें कर सकते हैं। लेकिन कुल मिलाकर, लैम्बडा के साथ परिचालन ओवरहेड काफी कम है जब आप इसकी तुलना EC2 जैसे विकल्पों से करते हैं। AWS लैम्बडा का उपयोग करते समय, आने वाला प्रत्येक अनुरोध या ईवेंट लैम्बडा फ़ंक्शन का अपना इंस्टेंस स्पिन करता है। लैम्बडा फ़ंक्शन एक माइक्रोवीएम में चलता है जिसे फायरक्रैकर नामक किसी चीज़ द्वारा संचालित किया जाता है, और यह माइक्रोवीएम एक समय अवधि के बाद अपने आप बंद हो जाएगा। इसलिए, यदि आपकी कोई मांग नहीं है, तो आपको अपने एप्लिकेशन कोड के निष्क्रिय इंस्टेंस से निपटना नहीं पड़ेगा। यह केवल तभी चलता है जब इसे चलाने की आवश्यकता होती है। AWS Fargate की तरह, स्केलिंग प्रबंधित है, परिचालन ओवरहेड कम है, प्रदर्शन दक्षता अधिक है।

हमारी वास्तुकला, यहाँ।

API गेटवे हमारे कंप्यूट विकल्प, AWS लैम्ब्डा के सामने आने वाला है।

हम इस सेटअप में और ब्लॉक जोड़ने जा रहे हैं

जैसे-जैसे हम पूरे सप्ताह आगे बढ़ेंगे, इस वास्तुकला को पूरा करने के लिए।

हम अभी इस समाधान के साथ शुरुआत कर रहे हैं,

तो चलिए आगे बढ़ते हैं।

ट्रांसक्रिप्ट का अंत। शुरुआत पर जाएँ।

## AWS Lambda

For this weeks architecture, we have chosen AWS Lambda as the compute service due to it’s serverless nature and ability to support a web backend.   
  
Lambda is a compute service that provides serverless compute functions that run in response to events or triggers. When an event or trigger is detected, a Lambda function is spun up in its own secure and isolated runtime environment, which is called an execution environment.   
  
Lambda functions can run for up to 15 minutes. Any processes that need longer than 15 minutes to run should use other compute services on AWS for hosting.  
  
Each execution environment stays active for a period of time, and then it shuts down on its own.   
  
When you use Lambda, you are responsible only for your code, which can make it easier to optimize for operational efficiency and low operational overhead. Lambda manages the compute fleet, which offers a balance of memory, CPU, network, and other resources to run your code. Because Lambda manages these resources, you can’t log in to compute instances or customize the operating system on the provided runtimes. Lambda performs operational and administrative activities on your behalf, including managing capacity, monitoring, and logging your Lambda functions.  
  
If you need to manage your own compute resources, AWS has other compute services that can meet your needs. For example:

* Amazon Elastic Compute Cloud (Amazon EC2) offers a wide range of EC2 instance types to choose from. With Amazon EC2, you can customize operating systems, settings for network and security, and the entire software stack. You are responsible for provisioning capacity, monitoring fleet health and performance, and using Availability Zones for fault tolerance.
* AWS Elastic Beanstalk is a service that you can use to deploy and scale applications on Amazon EC2. You retain ownership and full control over the underlying EC2 instances.

Lambda can be used for virtually any application or backend that requires compute and that runs in under 15 minutes. Common use cases are web backends, Internet of Things (IoT) backends, mobile backends, file or data processing, stream or message processing, and more.   
  
Lambda is a good choice for use cases where the requirements include reducing operational overhead, optimizing for cost, or optimizing for performance efficiency.  
  
Lambda works well for these use cases because it’s a managed service and you only pay for what you use. There are no idling resources when working with AWS Lambda, which means that each Lambda function is highly performant and cost efficient.

* To gain a deeper understanding of Lambda, see [AWS Lambda FAQs](https://aws.amazon.com/lambda/faqs/).
* To learn more about Lambda, see the [AWS Lambda Developer Guide](https://docs.aws.amazon.com/lambda/latest/dg/welcome.html) or the [AWS Lambda Operator Guide](https://docs.aws.amazon.com/lambda/latest/operatorguide/intro.html).
* To learn more about architecting and best practices for Lambda, see [Architecture and Best Practices](https://docs.aws.amazon.com/lambda/latest/operatorguide/architecture-best-practice.html).
* For a list of technical talks that cover Lambda, see [AWS Lambda - Technical Talks](https://aws.amazon.com/lambda/resources/webinars-and-talks/).
* For a list of technical tutorials that use Lambda, see [AWS Lambda - Workshops & Tutorials](https://aws.amazon.com/lambda/resources/workshops-and-tutorials/).

## Amazon API Gateway

After Morgan selected Lambda for the compute backend, she needed to find a way to expose the backend Lambda function. Amazon API Gateway integrates with Lambda, thus providing a way to expose the backend service without exposing to the open internet. This week’s customer might decide to add authentication to API Gateway to secure it further.  
  
API Gateway is a fully managed service that makes it easier for developers to create, publish, maintain, monitor, and secure APIs at any scale. APIs act as the front door for applications, so that the applications can access data, business logic, or functionality from your backend services. By using API Gateway, you can create RESTful APIs and WebSocket APIs that enable real-time two-way communication applications. API Gateway supports containerized and serverless workloads, as well as web applications.  
  
API Gateway handles all the tasks involved in accepting and processing up to hundreds of thousands of concurrent API calls, including traffic management, CORS support, authorization and access control, throttling, monitoring, and API version management. API Gateway has no minimum fees or startup costs. You pay for the API calls you receive and the amount of data transferred out and, with the API Gateway tiered pricing model, you can reduce your cost as your API usage scales.

* For more information about Amazon API Gateway, see [Amazon API Gateway](https://aws.amazon.com/api-gateway/) or [Amazon API Gateway FAQs](https://aws.amazon.com/api-gateway/faqs/).

## Amazon EC2

Amazon EC2 is a service that provides resizable compute capacity in the cloud, which means that it provides virtual machines in the cloud. Amazon EC2 is a flexible service that offers multiple instance types, sizes, and pricing models to meet specific requirements. Because you can choose your operating system and configurations for your instance, you can configure Amazon EC2 to work with virtually any workload.  
  
You can use Amazon EC2 when you want to run applications on AWS, but still want to retain control over the underlying infrastructure.   
  
Morgan didn’t choose Amazon EC2 as the compute service for this customer’s use case because of the operational overhead that Amazon EC2 requires. You have a lot of control over Amazon EC2, but that control also means that you will have overhead for managing the service. The customer had a straightforward use case and was willing to rewrite the code to use Lambda. The customer also had a spiky demand for their workload. Thus, choosing a service such as Lambda minimizes idling resources during low volume times, which can be more difficult to achieve with Amazon EC2.

* For a beginner tutorial about Amazon EC2, see [Get started with Amazon EC2 Linux instances](https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/EC2_GetStarted.html).
* To gain a deeper understanding of Amazon EC2, see [What is Amazon EC2?](https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/concepts.html) or [Amazon EC2 FAQs](https://aws.amazon.com/ec2/faqs/).

## AWS container services

Container management tools can be divided into three categories: registry, orchestration, and compute. AWS offers services that give you a secure place to store and manage your container images, orchestration that manages when and where your containers run, and flexible compute engines to power your containers. AWS can help manage your containers and their deployments for you, so you don't have to worry about the underlying infrastructure. No matter what you're building, AWS makes it easy and efficient to build with containers.  
  
Container services were not chosen for this architecture because the customer did not want to integrate this technology into their stack. So, even though running a container on Amazon ECS using AWS Fargate as the compute platform would technically work it was not chosen because of other customer preferences.

### Amazon ECS

Amazon Elastic Container Service (Amazon ECS) is a fully managed container orchestration service that you can use to deploy, manage, and scale containerized applications. It integrates with the rest of the AWS Cloud to provide a secure and easy-to-use solution for running container workloads in the cloud or on premises.   
  
Key features of Amazon ECS:

* Serverless by default with AWS Fargate: Fargate is built into Amazon ECS, and it reduces the time you need to spend on managing servers, handling capacity planning, or figuring out how to isolate container workloads for security. With Fargate, you define your application’s requirements and select Fargate as your launch type in the console or AWS Command Line Interface (AWS CLI). Then, Fargate takes care of all the scaling and infrastructure management that’s needed to run your containers.
* Security and isolation by design: Amazon ECS natively integrates with the tools you already trust for security, identity, and management and governance. This can help you get to production quickly and successfully. You can assign granular permissions for each of your containers, giving you a high level of isolation when you build your applications. You can launch your containers with the security and compliance levels that you have come to expect from AWS.
* Autonomous control plane operations: Amazon ECS is a fully-managed container orchestration service, with AWS configuration and operational best practices built-in—with no control plane, nodes, or add-ons for you to manage. It natively integrates with both AWS and third-party tools to make it easier for teams to focus on building the applications, not the environment.

### Amazon EKS

Amazon Elastic Kubernetes Service (Amazon EKS) is a managed service that you can use to run Kubernetes on AWS without needing to install, operate, and maintain your own Kubernetes control plane or nodes. Kubernetes is an open-source system for automating the deployment, scaling, and management of containerized applications. Amazon EKS offers the following features:

* It runs and scales the Kubernetes control plane across multiple AWS Availability Zones to ensure high availability.
* It also automatically scales control plane instances based on load, detects and replaces unhealthy control plane instances, and provides automated version updates and patching for them.
* It is integrated with many AWS services to provide scalability and security for your applications, including the following capabilities:
* Amazon Elastic Container Registry (Amazon ECR for container images).
* Elastic Load Balancing for load distribution.
* AWS Identity and Access Management (IAM) for authentication.
* Amazon Virtual Private Cloud (VPC) for isolation.
* It runs up-to-date versions of Kubernetes, so you can use all of the existing plugins and tooling from the Kubernetes community.

Applications that run on Amazon EKS are fully compatible with applications that run on any standard Kubernetes environment—it doesn’t matter whether they run in on-premises data centers or public clouds. This means that you can migrate any standard Kubernetes application to Amazon EKS with virtually no code modification.

### AWS Fargate

AWS Fargate is a technology that you can use with Amazon ECS to run containers without managing servers or clusters of EC2 instances. AWS Fargate reduces your need to provision, configure, or scale clusters of virtual machines to run containers. Thus, it also minimizes your need to choose server types, decide when to scale your clusters, or optimize cluster packing.  
  
When you run your tasks and services with the Fargate launch type, you package your application in containers, specify the CPU and memory requirements, define networking and IAM policies, and launch the application. Each Fargate task has its own isolation boundary and doesn’t share the underlying kernel, CPU resources, memory resources, or elastic network interface with another task.  
  
With Amazon ECS on AWS Fargate capacity providers, you can use both Fargate and Fargate Spot capacity with your Amazon ECS tasks. With Fargate Spot, you can run interruption-tolerant Amazon ECS tasks at a discounted rate, compared to the Fargate price. Fargate Spot runs tasks on spare compute capacity. When AWS needs the capacity back, your tasks will be interrupted with a 2-minute warning notice.

* To learn more about Amazon EKS, see the [Amazon EKS User Guide](https://docs.aws.amazon.com/eks/latest/userguide/what-is-eks.html).
* To learn more about AWS container services, see [Containers on AWS.](https://aws.amazon.com/containers/services/)
* To learn more about AWS Fargate, see [Amazon ECS on AWS Fargate.](https://docs.aws.amazon.com/AmazonECS/latest/developerguide/AWS_Fargate.html)
* For hands-on tutorials using Amazon ECS, see [Amazon ECS Workshop.](https://ecsworkshop.com/)

# What did you learn?

Help future students by selecting the skills or subjects you learned from this unit.

Management Operations Communications Governance Operating Systems Amazon Web Services Scalability Tooling Application Programming Interface (API) Linux Authentications Amazon Elastic Compute Cloud Public Cloud Virtual Machines Data Processing Internet Of Things (IoT) Business Logic Fault Tolerance Data Centers Automation Kubernetes Open Source Technology RESTful API Capacity Planning Tutorials Nodes (Networking) Infrastructure Management Authorization (Computing) Version Control AWS Elastic Beanstalk AWS CLI (Command Line Interface) AWS Lambda Amazon Elastic Container Service Serverless Computing Amazon API Gateway Access Controls Amazon Virtual Private Cloud (VPC) Amazon Elastic Container Registry Amazon Elastic Kubernetes Service API Gateway AWS Identity And Access Management (IAM) Runtime Systems Operational Efficiency Managed Services Throttling User Guide Network Interface WebSocket

None of these subjects were covered

<https://learning.edx.org/course/course-v1:AWS+AWS-ARCH-1+3T2022/block-v1:AWS+AWS-ARCH-1+3T2022+type@sequential+block@16d555216961430994773b43deb832aa/block-v1:AWS+AWS-ARCH-1+3T2022+type@vertical+block@5489ec6eff8a4a9381271d10795ac7f5>

Start of transcript. Skip to the end.

- [Morgan] Time to think about databases.

Before we dive in, let's think about what we know already

about the customer's current setup and their goals.

We know they are currently storing orders

in a MySQL database hosted on premises.

We heard from our customer that managed scaling

and reduced operations is really important.

We also heard that they're really storing

some pretty simple data in their orders database,

and that other systems use their own databases.

Let's go ahead and pull up the databases page

at aws.amazon.com/products/databases on AWS,

and get a quick overview of what's out there.

All right, so if I go ahead and scroll down,

we have another handy table, here.

And you can see that we have a bunch

of different types of databases,

and all of these are purpose built,

meaning that they were built

with specific use cases in mind.

This is important as you evaluate

each database for the solution.

You may find that by choosing one database type

over another, you can save a lot of time

in database setup, maintenance, or even coding,

if the database supports features

that you may have otherwise built into the application side.

So, if we go ahead and run down this list, here,

of the databases offered by AWS,

we can start to narrow it down.

So we can see on the left-hand side,

that we have the database type

as relational, key-value, in-memory.

So all of these are the different types of databases.

If we start up here at the top,

we have our relational databases,

and these are going to be the type of database

that people generally think about

when they think of databases.

So, for things like traditional applications,

where you have a bunch of tables that have a schema,

and all of them relate to each other.

And you can run SQL queries on the information

in those tables.

So for that, we have Amazon RDS,

or Amazon Relational Database Service,

Amazon Aurora,

which Amazon Aurora has an option

called Amazon Aurora Serverless.

That might be good for our customer,

so I'm going to keep that one in mind.

And then, we also have our data warehousing,

which is Amazon Redshift.

Now one category down here, we have the key-value databases.

And for this one, we have Amazon DynamoDB listed.

Amazon DynamoDB is also another serverless database,

so I'm going to keep that one in mind, as well.

And then, as we keep going down this list,

you can see there's other categories, like in-memory.

This is going to be for caching.

So for caching, we have things like Amazon ElastiCache.

That's going to be a different use case

than something like a relational or key-value database.

So you can see, that we have these purpose-built databases

for specific use cases.

And if we keep going down here,

you can see we have even really specific use cases,

like a graph database,

which is going to be the database, Amazon Neptune.

So instead of trying to make a relational database

work like a graph database,

you could just use the graph database.

So, after quickly narrowing things down here,

I am really going to take a good look

at both Amazon DynamoDB

and Amazon Aurora Serverless as our two database options.

First up, let's talk about Amazon Aurora Serverless.

Amazon Aurora is a relational database

that is available through Amazon RDS.

So, it's kind-of under the Amazon RDS umbrella.

Aurora is built for massive scale and for performance.

This is really a database that is meant for large workloads

that you would usually run on a commercial database

that can offer the level of data storage and performance

that you would need for enterprise applications.

It supports MySQL or PostgreSQL workloads,

which is part of the reason why I wanted to look

into this service.

I know that our customer is using MySQL,

which would be drop-in compatible with Aurora.

We also know, additionally,

the customer wants to reduce operations.

Amazon Aurora is a service that is not totally managed,

in the sense that you can use it in a way

where you are managing the underlying instances,

such as planning DB instance sizes

and resizing DB instances,

or adding more DB instances as the workload changes.

Those are the types of management operations

that you could make.

This is where something called Amazon Aurora Serverless

comes in to simplify that.

This would reduce the operational load

that you may normally associate with using Amazon RDS

because you can determine the appropriate minimum

and maximum capacity by running the workload,

and checking how much the DB instances actually scale.

Another great thing about Aurora

Serverless

is that you only pay for what you use

because it scales in and out as demand changes.

This would be a good fit for our customer's spiky workload

because they don't need the database

running at full capacity at all times.

All right, so using Amazon Aurora Serverless

would be a decent fit for our use case.

Now, let's talk about our other option

if we are looking into, Amazon DynamoDB.

DynamoDB is a non-relational database option

that is also designed for massive scale

and high performance.

Aurora and DynamoDB are similar in that way.

They are both designed for scale.

DynamoDB is best used for applications

that are looking to store key-value data,

where the application has potentially a high number

of concurrent users and connections to the database.

It is also serverless,

but not as a feature, like Amazon Aurora.

DynamoDB is fully serverless,

and it scales the underlying storage

of your data under the hood.

There's nothing you need to do

to scale the storage of the data,

which definitely helps with reducing operational overhead.

For example, I used to be a database admin

and I was always working

on adding more storage database clusters,

or adding more table space for quickly growing datasets.

Those types of administrative tasks

just simply don't exist for DynamoDB

because all of that storage scaling is handled for you,

which is great.

Instead of worrying about data storage,

you instead can focus on your application's usage patterns

and throughput needs.

This means you can provision the throughput needed

ahead of time, meaning how much data

you read or write per second.

Or you can use on-demand throughput,

where it scales up and down as your demand changes.

For a spiky access pattern like our customer has,

the best fit for them would be to use the on-demand mode

for their DynamoDB tables.

Beyond throughput when working with DynamoDB,

you should have a good understanding

of your data-access patterns.

So what I mean by that is, what types of queries

is your customer running against their data?

With DynamoDB, you don't create a database

with multiple tables.

Instead, you have tables that are standalone,

with no built-in ability to do joins across these tables.

You instead model your table with indexes

that allow you to run the queries that you need.

This is a bit different than you would do

with something like Aurora, which uses standard SQL queries

and has the ability to do complex SQL queries across tables.

With that in mind, DynamoDB is not a great fit

for every use case.

If you do need complex SQL queries or joins,

then it likely isn't the best choice.

Let's go ahead and call up our customer to discuss this.

(phone rings)

- [Raf] Hello, Morgan, what's up?

- [Morgan] Hello there.

I'm working on building the architecture here

for the order service,

and I have some questions for you

about the types of data queries

that you're making to your orders database.

- [Raf] Uh-hmm, go ahead.

What do you need to know?

- [Morgan] Well, I'm looking into potentially

using DynamoDB for the database,

but to really use the service the right way,

you need to know what queries are being made up front

so that we can model a table correctly.

So, what queries are being made

against the orders database right now?

- [Raf] Yeah, we need to be able to create, read, update,

and delete orders based on an order ID.

We also need to be able to pull back any orders

associated with a customer ID.

All in all, it's a pretty simple table.

We are not doing anything like super complicated or...

Yeah.

- [Morgan] All right, cool.

So, there are no complex cross-table SQL queries

that are usually run as a part of your day to day?

- [Raf] Correct, we use the table as a place

to stash order information, and that's,

that's really it.

- [Morgan] Awesome.

All right, I think DynamoDB would be a good fit

for this use case,

but I need to think it through a little bit more

to make sure that I'm understanding

all of the different features

and make sure that it's the best choice.

So, we'll talk soon.

- [Raf] All right, talk to you soon. Bye-bye.

- [Morgan] All right.

So knowing what we know now,

that data modeling won't be an issue

because this is a pretty straightforward use case,

I am leaning towards DynamoDB as the best fit here.

Amazon Aurora Serverless is a great choice as well,

but our customer doesn't need the complex features

that Aurora offers.

And DynamoDB is easier to get started with

and to operate over time.

Also, we are using AWS Lambda as the compute service.

Think about what happens w

ith database connections

when you spin up a Lambda function

and you scale it really big.

If there are 100 unique Lambda execution environments,

each one would create its own connection to the database.

During sales events, when their customers are cleaning out

their supplies website, buying everything up,

the website could flood the database

with connection requests, which can cause issues.

Luckily, with Amazon Aurora Serverless,

this problem has been solved.

You'd need to use something like Amazon RDS Proxy

to sit in between the Lambda functions

and the Aurora database.

But then, you have another service to manage

or to think about.

With DynamoDB, that problem doesn't exist

and doesn't need to be solved,

so there is no need to bring in another service

like RDS Proxy.

The simplicity that comes along with using DynamoDB

is a big reason why I like it for our customer,

who wants to go as cloud native as possible.

And a cloud-native database like DynamoDB

is a great choice for trying to keep it simple for them.

So let's go ahead and add another block to our diagram.

So far, we have API Gateway handling the API layer,

and we have AWS Lambda

handling the orders processing service,

and now, we can add DynamoDB to store our order data.

We have a long way to go here,

before this meets our requirements.

Up next, we need to think about these downstream calls

for accounting,

order fulfillment,

and inventory.

Hmm.

End of transcript. Skip to the start.

डेटाबेस कनेक्शन के साथ

जब आप लैम्बडा फ़ंक्शन को स्पिन करते हैं

और आप इसे वास्तव में बड़ा स्केल करते हैं।

यदि 100 अद्वितीय लैम्बडा निष्पादन वातावरण हैं,

तो प्रत्येक डेटाबेस से अपना कनेक्शन बनाएगा।

बिक्री कार्यक्रमों के दौरान, जब उनके ग्राहक अपनी आपूर्ति वेबसाइट को साफ कर रहे होते हैं, सब कुछ खरीद रहे होते हैं,

वेबसाइट डेटाबेस को कनेक्शन अनुरोधों से भर सकती है, जिससे समस्याएँ हो सकती हैं।

सौभाग्य से, Amazon Aurora Serverless के साथ,

यह समस्या हल हो गई है।

आपको लैम्बडा फ़ंक्शन और Aurora डेटाबेस के बीच में बैठने के लिए Amazon RDS Proxy जैसी किसी चीज़ का उपयोग करने की आवश्यकता होगी।

लेकिन फिर, आपके पास प्रबंधित करने के लिए या विचार करने के लिए एक और सेवा है।

DynamoDB के साथ, वह समस्या मौजूद नहीं है

और इसे हल करने की आवश्यकता नहीं है,

इसलिए RDS Proxy जैसी कोई अन्य सेवा लाने की कोई आवश्यकता नहीं है।

DynamoDB का उपयोग करने के साथ आने वाली सरलता

एक बड़ा कारण है कि मैं इसे अपने ग्राहक के लिए पसंद करता हूँ,

जो जितना संभव हो सके क्लाउड नेटिव बनना चाहता है।

और DynamoDB जैसा क्लाउड-नेटिव डेटाबेस उनके लिए इसे सरल रखने की कोशिश करने के लिए एक बढ़िया विकल्प है। तो चलिए आगे बढ़ते हैं और अपने आरेख में एक और ब्लॉक जोड़ते हैं। अब तक, हमारे पास API गेटवे है जो API लेयर को संभालता है, और हमारे पास AWS लैम्ब्डा है जो ऑर्डर प्रोसेसिंग सेवा को संभालता है, और अब, हम अपने ऑर्डर डेटा को स्टोर करने के लिए DynamoDB जोड़ सकते हैं। हमें यहाँ एक लंबा रास्ता तय करना है, इससे पहले कि यह हमारी आवश्यकताओं को पूरा करे। आगे, हमें अकाउंटिंग, ऑर्डर पूर्ति और इन्वेंट्री के लिए इन डाउनस्ट्रीम कॉल के बारे में सोचना होगा। हम्म। ट्रांसक्रिप्ट का अंत। शुरुआत पर जाएँ।

<https://aws.amazon.com/products/>

<https://aws.amazon.com/products/databases/>

# Databases on AWS

Databases are purpose-built on AWS, which means that each AWS database service is built for a specific use case or set of use cases. Using a database that is a best fit for the use case can save a lot of time in development hours. In the past, it was common to use relational databases for everything because they were the most commonly operated database on premises. With AWS, you can run different types of databases more easily without managing the infrastructure yourself. This can lead to making decisions that are more aligned with the use case and aren’t limited to in-house skill for database administration.   
  
For this weeks customer, Morgan chose Amazon DynamoDB as the database choice because the customer is using it as a simple lookup table, there is no need for complex SQL queries or joins across tables, and the serverless nature of the table makes it easy to operate over time.  
  
For a high-level overview of the AWS database services, see [AWS Cloud Databases.](https://aws.amazon.com/products/databases/)

## Amazon Aurora

Amazon Aurora is a fully managed relational database engine that's compatible with MySQL and PostgreSQL. You can use the code, tools, and applications for your existing MySQL and PostgreSQL databases with Aurora.   
  
Aurora wasn’t chosen for this architecture because the customer doesn’t need the complex, enterprise-database features that Aurora offers.  
  
As an enterprise-level database, Aurora can—with some workloads—deliver up to five times the throughput of MySQL and up to three times the throughput of PostgreSQL without requiring changes to most of your existing applications.  
  
Aurora includes a high-performance storage subsystem. Its MySQL-compatible and PostgreSQL-compatible database engines are customized to take advantage of that fast, distributed storage. The underlying storage grows automatically as needed. An Aurora cluster volume can grow to a maximum size of 128 tebibytes (TiB). Aurora also automates and standardizes database clustering and replication, which are typically among the most challenging aspects of database configuration and administration.  
  
Aurora is part of the managed database service Amazon Relational Database Service (Amazon RDS). Amazon RDS is a web service that makes it easier to set up, operate, and scale a relational database in the cloud.   
  
Aurora Serverless v2is an on-demand, automatic scaling configuration for Aurora. Aurora Serverless v2 helps automate the processes of monitoring the workload and adjusting the capacity for your databases. Capacity is adjusted automatically based on application demand. You're charged only for the resources that your database clusters consume. Thus, Aurora Serverless v2 can help you to stay within budget and reduce the need to pay for computer resources that you don't use.  
  
This type of automation is especially valuable for multitenant databases, distributed databases, development and test systems, and other environments with highly variable and unpredictable workloads.

* For more information on Amazon Aurora Serverless, see [Using Aurora Serverless v2](https://docs.aws.amazon.com/AmazonRDS/latest/AuroraUserGuide/aurora-serverless-v2.html).
* For links to tutorials for Amazon Aurora, see [Getting started with Amazon Aurora](https://aws.amazon.com/rds/aurora/getting-started/).

## Amazon RDS Proxy

By using Amazon RDS Proxy, your applications can pool and share database connections to improve their ability to scale. RDS Proxy makes applications more resilient to database failures by automatically connecting to a standby DB instance, while preserving application connections. By using RDS Proxy, you can also enforce AWS Identity and Access Management (IAM) authentication for databases, and securely store credentials in AWS Secrets Manager.  
  
With RDS Proxy, you can handle unpredictable surges in database traffic that otherwise might cause issues because of oversubscribing connections or creating new connections at a fast rate. RDS Proxy establishes a database connection pool and reuses connections in this pool without the memory and CPU overhead of opening a new database connection each time. To protect the database against oversubscription, you can control the number of database connections that are created.  
  
RDS Proxy queues or throttles application connections that can't be served immediately from the pool of connections. Although latencies might increase, your application can continue to scale without abruptly failing or overwhelming the database.

* For more information about Amazon RDS Proxy, see [Using Amazon RDS Proxy.](https://docs.aws.amazon.com/AmazonRDS/latest/AuroraUserGuide/rds-proxy.html)

## Amazon DynamoDB

Amazon DynamoDB is a fully managed NoSQL database service that provides fast and predictable performance with seamless scalability. By using DynamoDB, you can offload the administrative burdens of operating and scaling a distributed database so that you can reduce your need to handle hardware provisioning, setup and configuration, replication, software patching, or cluster scaling. DynamoDB also offers encryption at rest, which reduces your operational burden and the complexity involved in protecting sensitive data.   
  
With DynamoDB, you can create database tables that can store and retrieve virtually any amount of data and serve virtually any level of request traffic. You can scale up or scale down your tables' throughput capacity with minimal downtime or performance degradation.  
  
If you are an application developer, you might have some experience using a relational database management system (RDBMS) and Structured Query Language (SQL). As you begin working with Amazon DynamoDB, you will encounter many similarities, but also many things that are different.   
  
*NoSQL* is a term used to describe nonrelational database systems that are highly available, scalable, and optimized for high performance. Instead of the relational model, NoSQL databases (such as DynamoDB) use alternate models for data management, such as key-value pairs or document storage.   
  
In DynamoDB, tables, items, and attributes are the core components that you work with. A table is a collection of items, and each item is a collection of attributes. DynamoDB uses primary keys to uniquely identify each item in a table, and secondary indexes to provide more querying flexibility. You can use DynamoDB Streams to capture data modification events in DynamoDB tables.

* For more information about Amazon DynamoDB, see [What is Amazon DynamoDB?](https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/Introduction.html) or [Amazon DynamoDB FAQs](https://aws.amazon.com/dynamodb/faqs/).
* For more information about Amazon DynamoDB data modeling, see [Example of modeling relational data in DynamoDB](https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/bp-modeling-nosql-B.html).
* For hands-on learning with Amazon DynamoDB, see [Hands-on labs for DynamoDB](https://amazon-dynamodb-labs.com/hands-on-labs.html).

1. - [Morgan] During our customer call earlier this week,
2. we heard from them about some of the issues
3. that their code has.
4. Because the order-service code
5. is all wrapped up into one application,
6. if any part of that process breaks, the entire order fails,
7. or some of the downstream calls to inventory, accounting,
8. or fulfillment can break.
9. Then, the next subsequent downstream call is not made.
10. Part of our plan for the order service
11. is to break it down into smaller, more decoupled components
12. to solve these problems,
13. while also meeting the requirements of improved resiliency,
14. lower cost and operational overhead,
15. as well as managed scaling.
16. So far, we have designed our proposed solution
17. using API Gateway, AWS Lambda, and DynamoDB,
18. where API Gateway accepts and validates the order,
19. then sends it to the backend Lambda function,
20. which then will do any processing,
21. and then we'll save that order to the DynamoDB table.
22. Now, we need to build out the part of the solution
23. that handles the calls made to the downstream services,
24. while keeping the customer's requirements in mind.
25. Now, you could of course code the Lambda function
26. to make the calls to the downstream services,
27. but then those calls are still tightly coupled
28. with the business logic of processing the orders,
29. which is the same problem that they have now.
30. If I ever, down the road, want to change the way
31. we fan out these messages to the downstream services,
32. I would have to change the order-processing code,
33. which could inadvertently introduce potential bugs.
34. We want to decouple the application code
35. from the downstream calls
36. so that the Lambda function focuses on taking in
37. the incoming order, processing it,
38. and then saving it to the database.
39. From there, we can build out the solution
40. to capture the event of the order being written
41. to the database, and send it to the downstream services
42. using a publish/subscribe model,
43. without needing to have
44. the orders-processing Lambda function do that work directly.
45. So, knowing that there are three downstream services
46. already hosted on AWS,
47. and we need to fan out the order that is placed
48. to all of those services as a message.
49. I am thinking of using an event-driven architecture here,
50. which can help us out.
51. Let's start by pulling up the AWS documentation
52. to see what it has to say
53. about event-driven architectures,
54. at aws.amazon.com/event-driven-architecture.
55. If we scroll down to the diagram on this page,
56. it shows an example of an event-driven architecture,
57. and in this case, we have three components.
58. We have the event producer,
59. the event router, here in the middle,
60. and then, we have the event consumer on the right.
61. A producer publishes an event to the router,
62. which can then filter and push the events to consumers.
63. Producer services and consumer services are decoupled,
64. which means that they can be scaled,
65. updated, and deployed independently.
66. Recalling the architecture decisions
67. we have already made for this order service,
68. we have API Gateway sitting in front of Lambda,
69. which then writes to a DynamoDB table.
70. AWS Lambda is a great choice for event-driven architectures
71. because it doesn't run all of the time.
72. It runs in response to events
73. triggering the Lambda's functions invocation.
74. So, we are already in a good starting place
75. for event-driven architectures.
76. We need to consider now how to route the event
77. to the downstream services to be processed.
78. Back to the event-driven architectures webpage,
79. if I scroll down this page,
80. we can then review what AWS services exist
81. that can help us send the messages
82. to the downstream services,
83. since we don't want our Lambda function to do it directly.
84. Under this section titled Where to start,
85. I can see we essentially have two options
86. to push these messages to our consumers.
87. We have Amazon EventBridge here, and we have
88. Amazon Simple Notification Service, or SNS, here.
89. Starting with Amazon Simple Notification Service,
90. let me open my browser tab to aws.amazon.com/sns.
91. So, if we go ahead and scroll down this webpage
92. to this helpful diagram, we can see how SNS works.
93. SNS is a fully managed messaging system
94. that uses a publish/subscribe, or pub/sub model,
95. where a publisher can send or publish messages or events
96. to something called an SNS topic,
97. and then, you can subscribe multiple consumers
98. to this topic so they receive any published messages.
99. SNS has a one-to-many relationship
100. for published messages and subscribers,
101. where one message is consumed by all subscribers to a topic,
102. which is like the fan-out method that I mentioned earlier.
103. Optional filtering can be applied to topics as well,
104. so if you want to set it up where only some subscribers
105. received specific messages but not others,
106. that is possible with SNS.
107. **Potential subscribers for SNS topics could be things like,**
108. **an HTTPS endpoint, AWS Lambda functions, Amazon SQS,**
109. or other AWS services.
110. With our customer's use case,
111. the order being placed would be the message to be published,
112. and the subscribers would be the three downstream services,
113. who all need to consume the same message.
114. This seems like it could be a good fit for our use case.
115. Let's now go ahead and evaluate Amazon EventBridge,
116. and see what that service is all about.
117. Bringing up the Amazon EventBridge page
118. at aws.amazon.com/eventbridge.
119. If I go ahead and scroll down this page,
120. we have another diagram that can help us
121. understand EventBridge.
122. EventBridge is a serverless event bus
123. that connects applications using events.
124. This is very similar to SNS,
125. but there are a few differences.
126. First is that EventBridge supports a lot more targets
127. for consumers or subscribers than SNS does,
128. including third-party software solutions
129. like software-as-a-service applications,
130. and that also includes the other side of things
131. for the event sources.
132. EventBridge supports a lot
133. of different event sources, as well,
134. like software-as-a-service, or SaaS, applications.
135. It also supports message filtering by using rules,
136. whereas SNS only supports filtering
137. based on message attributes.
138. EventBridge can do much more deeper filtering.
139. This can be helpful if you have a variety
140. of messages being published ,that you need to route
141. those specific messages to specific consumers.
142. In our use case, we know that all three downstream consumers
143. will consume the same message,
144. so this doesn't really help us out a lot,
145. even though it is a cool feature.
146. EventBridge also has something called the schema registry,
147. which stores schemas generated
148. by your organization's applications,
149. AWS services, or SaaS applications.
150. A schema includes information like the title, format,
151. and validation rules for event data.
152. These can then be used to help you define
153. how you want your messages structured,
154. which can be particularly handy
155. if you are integrating from outside third-party systems.
156. Again, with our use case,
157. we really only have one message type,
158. and we control the shape of that message, as well,
159. so this isn't super helpful to us.
160. With all that being said,
161. I think EventBridge is a more complex
162. and powerful tool than SNS.
163. Both SNS and EventBridge are serverless,
164. so we don't need to worry about scaling here,
165. and both services would help us decouple our order service
166. and improve its resilience.
167. So they both technically meet the requirements.
168. However, SNS wins over EventBridge for cost,
169. and it wins for simplicity.
170. With our customer's simple use case,
171. SNS can handle the job very well.
172. I'm going to go ahead and add SNS
173. to our architecture diagram,
174. and this will send the messages to our downstream services.
175. This looks great, but there is still one piece missing.
176. How are we going to get the event
177. that an order was added to the DynamoDB table
178. to be published to the SNS topic?
179. This is where something called DynamoDB Streams
180. and AWS Lambda come into play.
181. DynamoDB Streams can be enabled on a DynamoDB table,
182. so that when changes are made to the table,
183. that event is captured in an ordered flow
184. of information called a stream,
185. and that stream can then be processed by applications.
186. In our case, we can process the stream
187. of new, incoming orders
188. by setting up another Lambda function,
189. and designating the DynamoDB stream as the trigger.
190. This Lambda function will receive the information
191. about the item being added to the DynamoDB table
192. in the payload of the request that gets sent to Lambda.
193. It will then take that data and submit an API call to SNS,
194. publishing the event to the topic.
195. So let's go ahead and add the DynamoDB stream here,
196. as well as our second Lambda function to our diagram.
197. The great thing about this design is that the mechanism
198. of sending the message to the downstream services
199. is handled outside of the order-processing code.
200. So if, at any point, this changes or has issues,
201. it is totally decoupled
202. from the application's business logic.
203. Additionally, if any new downstream services get added,
204. like an analytics service,
205. we can simply just subscribe the HTTPS endpoint
206. of the new service to the SNS topic as the subscriber,
207. and it will start to receive messages,
208. all without making any coding changes
209. to the existing solution, which is the key here.
210. This solution is shaping up nicely,
211. but there are a few more things that I want to add.
212. Let's talk about decoupling some more, coming up next.
213. [End of transcript. Skip to the start.](https://courses.edx.org/xblock/block-v1:AWS+AWS-ARCH-1+3T2022+type@vertical+block@ab7aeb60812547fe88392580c0c099e6?exam_access=&recheck_access=1&show_bookmark=0&show_title=0&view=student_view#transcript-start-3c3303ea3a3f4d76945d0e1e534463d2)
214. [Start of transcript. Skip to the end.](https://courses.edx.org/xblock/block-v1:AWS+AWS-ARCH-1+3T2022+type@vertical+block@0e5551ed0ea642f2aee204316da87da2?exam_access=&recheck_access=1&show_bookmark=0&show_title=0&view=student_view#transcript-end-fd8a6c78bf76417e8a79ab86427c8d41)
215. - [Morgan] In this video, we are going to explore
216. Amazon SNS, or Simple Notification Service.
217. So, I'm going to go ahead and navigate
218. to Simple Notification Service.
219. We'll select this service.
220. And this brings us to the landing page.
221. From this page, you can,
222. first thing, it's asking you to do,
223. is create a topic.
224. So, if we remember, Amazon Simple Notification Service
225. is a pub/sub messaging service for microservices
226. and serverless applications.
227. So, it essentially allows you to send a message
228. to what's called a topic, which is a message channel.
229. And then, this message channel will fan out the message,
230. or send the message to all of the subscribed endpoints.
231. So, let's go ahead and create a topic
232. and see what sorts of configurations are available to us.
233. So I'm just going to go ahead and name this test.
234. And then, click Next step.
235. Now, from here, if we explore this Create topic page,
236. right away, I'm noticing that we have
237. two different types of topics that are available to us.
238. We can do a FIFO, or first-in, first-out, or Standard.
239. So, a FIFO SNS topic is going to do
240. a strictly preserved order, messaging order.
241. So, if for some reason,
242. the order that the messages were received
243. matters to your application,
244. you would want to select this.
245. This will have exactly once message delivery,
246. so you're not going to get
247. multiple messages delivered multiple times.
248. And then, also, this provides you with a high throughput,
249. up to 300 publishes per second.
250. Whereas with a Standard SNS topic,
251. this gives you a lot more throughput
252. because this is best-effort message ordering.
253. So, that means that the order that it comes in
254. may not be the order that it comes out.
255. So a lot of applications,
256. the order doesn't necessarily matter.
257. And in our use case,
258. the order won't necessarily matter too much, here,
259. because the payment's already been processed
260. and things like that.
261. So, we are going to go ahead and select Standard.
262. And then, we can go down and select some of the other types
263. of configurations that we have for SNS topics.
264. One is that you can provide encryption in transit.
265. And so, encryption in transit is enabled by default.
266. And you can also add enabling server-side encryption,
267. which will add encryption at rest for your topic.
268. So, you have to kind of go out of your way here,
269. to enable encryption at rest.
270. And we will do that.
271. And what this will do, is it encrypts your message
272. as soon as it's arrived, or as soon as it's received.
273. And then, your message will be decrypted prior to delivery.
274. So, that means that when SNS has your message,
275. it's going to be encrypted the entire time
276. that the service is handling that message.
277. So now, if we keep scrolling down here,
278. we can also open up this Access policy,
279. and this is going to be the last thing
280. that I want to highlight for this service.
281. And here, you can see that we can create policies
282. that define who can access your topic.
283. So, AWS has these things called resource policies,
284. where you can attach a policy directly to an AWS resource
285. and restrict who can access this resource.
286. So here, you can see we have
287. who can publish messages to your topic.
288. You can select: Only the topic owner,
289. Everyone, Only the specified AWS accounts.
290. You can also say who can subscribe to this topic.
291. Is it only the topic owner?
292. Is it everyone? Only AWS accounts?
293. Or is it only requesters with certain endpoints?
294. By selecting these, it will kind-of change
295. this JSON preview for you, over on the right-hand side.
296. But you also can select the Advanced mode, here,
297. and directly edit this JSON.
298. Now, we can scroll down here,
299. and click Create topic.
300. And our topic was created.
301. And at this point, we could then go down.
302. We could create some subscriptions for this.
303. So, a subscription could be a:
304. Kinesis Data Firehose, an SQS queue,
305. a Lambda function, an email address,
306. it could be an HTTPS endpoint, or it could be SMS for texts.
307. So, you can use SNS to send updates to email addresses,
308. or phone numbers, or all sorts of things.
309. So, this is where you would come in
310. and you would create your subscriptions.
311. I'm going to go ahead and click Cancel for now.
312. And that is all we have for this SNS exploration video.
313. [End of transcript. Skip to the start.](https://courses.edx.org/xblock/block-v1:AWS+AWS-ARCH-1+3T2022+type@vertical+block@0e5551ed0ea642f2aee204316da87da2?exam_access=&recheck_access=1&show_bookmark=0&show_title=0&view=student_view#transcript-start-fd8a6c78bf76417e8a79ab86427c8d41)

<https://aws.amazon.com/sns/>

<https://aws.amazon.com/eventbridge/>

1. [Start of transcript. Skip to the end.](https://courses.edx.org/xblock/block-v1:AWS+AWS-ARCH-1+3T2022+type@vertical+block@ab7aeb60812547fe88392580c0c099e6?exam_access=&recheck_access=1&show_bookmark=0&show_title=0&view=student_view#transcript-end-3c3303ea3a3f4d76945d0e1e534463d2)

# Event-driven architectures on AWS

This week’s customer currently uses a synchronous web application to host the orders service, which is causing various issues—for example, the code is too tightly coupled with downstream API calls. Morgan suggested that they move to an event-driven architecture to solve this problem.   
  
An event-driven architecture uses events to invoke and communicate between decoupled services. It’s a common architecture in modern applications that are built with microservices. An event is a change in state, or an update, such as placing an item in a shopping cart on an ecommerce website. Events can either carry the state (the item purchased, its price, and a delivery address) or events can be identifiers (a notification that an order was shipped).  
  
Event-driven architectures have three key components: event producers, event routers, and event consumers. A producer publishes an event to the router, which filters and pushes the events to consumers. Producer services and consumer services are decoupled, which means that they can be scaled, updated, and deployed independently.  
  
The following diagram shows an example of an event-driven architecture for an ecommerce site. By using this architecture, the site can react to changes from various sources during times of peak demand, without crashing the application or overprovisioning resources.  
For more information about event-driven architectures, see [What is an Event-Driven Architecture?](https://aws.amazon.com/event-driven-architecture/)

## Amazon EventBridge compared to Amazon SNS

You can use both Amazon EventBridge and Amazon Simple Notification Service (Amazon SNS) to develop event-driven applications. Your choice will depend on your specific needs.   
  
We recommend EventBridge when you want to build an application that reacts to events from software as a service (SaaS) applications or AWS services. EventBridge is the only event-based service that integrates directly with third-party SaaS AWS Partners. EventBridge also automatically ingests events from over 90 AWS services without requiring developers to create any resources in their account. In addition, EventBridge uses a defined, JSON-based structure for events, and you can also select events to forward to a target by creating rules that are applied across the entire event body. EventBridge currently supports over 15 AWS services as targets, including AWS Lambda, Amazon Simple Queue Service (Amazon SQS), Amazon SNS, Amazon Kinesis Data Streams, and Amazon Kinesis Data Firehose, and more. At launch, EventBridge has limited throughput (see the service limits), which can be increased on request. It also has a typical latency of about half a second.  
  
We recommend Amazon SNS when you want to build an application that reacts to high throughput or low-latency messages that are published by other applications or microservices. Amazon SNS provides nearly unlimited throughput. You can also use it for applications that need very high fan-out (thousands or millions of endpoints). Messages are unstructured and can be in any format. Amazon SNS supports forwarding messages to six different types of targets, including AWS Lambda, Amazon SQS, HTTP/S endpoints, Short Message Service (SMS), mobile push, and email. The typical latency of Amazon SNS typical is under 30 milliseconds. A range of AWS services—more than 30, including Amazon Elastic Compute Cloud (Amazon EC2), Amazon Simple Storage Service (Amazon S3), and Amazon Relational Database Service (Amazon RDS)—send SNS messages by configuring the service to send them.

## Amazon EventBridge

EventBridge is a serverless event bus service that you can use to connect your applications with data from various sources. EventBridge delivers a stream of real-time data from your applications, software as a service (SaaS) applications, and AWS services to targets such as AWS Lambda functions, HTTP invocation endpoints using API destinations, or event buses in other AWS accounts.  
  
EventBridge receives an event, which is an indicator of a change in environment. EventBridge then applies a rule to route the event to a [target](https://docs.aws.amazon.com/eventbridge/latest/userguide/eb-targets.html). Rules match events to targets based on either the structure of the event (which is called an *e*vent pattern), or on a schedule. For example, when an EC2 instance changes from pending to running, you can have a rule that sends the event to a Lambda function. For more information about events, see [Amazon EventBridge events](https://docs.aws.amazon.com/eventbridge/latest/userguide/eb-events.html). For more information about rules, see [Amazon EventBridge rules](https://docs.aws.amazon.com/eventbridge/latest/userguide/eb-rules.html). Finally, for more information about event patterns, see [Amazon EventBridge event patterns](https://docs.aws.amazon.com/eventbridge/latest/userguide/eb-event-patterns.html).  
  
  
All events that come to EventBridge are associated with an event bus. Rules are tied to a single event bus, so they can only be applied to events on that event bus. Your account has a default event bus, which receives events from AWS services. You can also create custom event buses to send or receive events from a different account or Region. For more information about event buses, see [Amazon EventBridge event buses](https://docs.aws.amazon.com/eventbridge/latest/userguide/eb-event-bus.html).

* For more information about Amazon EventBridge, see the [Amazon EventBridge FAQs](https://aws.amazon.com/eventbridge/faqs/) or [What is Amazon EventBridge?](https://docs.aws.amazon.com/eventbridge/latest/userguide/eb-what-is.html)

## Amazon SNS

Amazon SNS is a managed service that provides message delivery from publishers to subscribers (which are also known as producers and consumers). Publishers communicate asynchronously with subscribers by sending messages to a topic, which is a logical access point and communication channel. Clients can subscribe to the SNS topic and receive published messages by using a supported endpoint type, such as Amazon Kinesis Data Firehose, Amazon SQS, AWS Lambda, HTTP, email, mobile push notifications, and mobile text messages through Short Message Service (SMS).  
  
Morgan chose Amazon SNS the customer’s architecture because it’s simple to use and supports a straightforward way to send messages between application components.

**Service Update:** Amazon SNS now supports payload-based message filtering [click here](https://aws.amazon.com/blogs/compute/introducing-payload-based-message-filtering-for-amazon-sns/) for more information.

* For more information about Amazon SNS, see the [Amazon SNS FAQs](https://aws.amazon.com/sns/faqs/) or [What is Amazon SNS?](https://docs.aws.amazon.com/sns/latest/dg/welcome.html)
* For a tutorial on how to set up an Amazon SNS topic, see [Getting started with Amazon SNS.](https://docs.aws.amazon.com/sns/latest/dg/sns-getting-started.html#step-create-queue)

## Amazon DynamoDB Streams

DynamoDB Streams captures a time-ordered sequence of item-level modifications in any DynamoDB table, and stores this information in a log for up to 24 hours. Applications can access this log and view the data items as they appeared, before and after they were modified, in near-real time.  
  
Encryption at rest encrypts the data in DynamoDB streams.  
  
DynamoDB Streams helps ensure the following:

* Each stream record appears exactly one time in the stream.
* For each item that is modified in a DynamoDB table, the stream records appear in the same sequence as the actual modifications to the item.

DynamoDB Streams writes stream records in near-real time so that you can build applications that consume these streams and take action based on the contents.  
  
You can enable a stream on a new table when you create it by using the AWS Command Line Interface (AWS CLI) or one of the AWS SDKs. You can also enable or disable a stream on an existing table, or change the settings of a stream. DynamoDB Streams operates asynchronously, so table performance isn’t affected if you enable a stream.  
  
All data in DynamoDB Streams is subject to a 24-hour lifetime. You can retrieve and analyze the last 24 hours of activity for any given table. However, data that is older than 24 hours is susceptible to trimming (removal) at any moment.

1. [tart of transcript. Skip to the end.](https://courses.edx.org/xblock/block-v1:AWS+AWS-ARCH-1+3T2022+type@vertical+block@db6e843cfae4401e9571359097015e09?exam_access=&recheck_access=1&show_bookmark=0&show_title=0&view=student_view#transcript-end-312502f7db544810a77c77db05457fec)
2. - [Morgan] In the last video,
3. we started to break down the different components
4. of our customer's application,
5. and begin to decouple these components from each other,
6. using services like Amazon SNS,
7. and following an event-driven model.
8. This solved the problem where the application
9. was tightly coupled with the downstream calls being made,
10. but we have not addressed the latency issues
11. with the order-acceptance component of their application.
12. Right now, the way their application is written on premises,
13. it's using a synchronous model,
14. where the user submits their order to the order service,
15. the service accepts and validates the order,
16. process the order,
17. stores it in a database,
18. and then calls the downstream services for inventory,
19. fulfillment, and accounting before it responds to the user.
20. We have heard from our customer that this whole process
21. can be slow and takes too long to complete.
22. If we picture our customer's problem,
23. their end users are probably sitting there,
24. watching the browser just clock
25. and look frozen before it finally responds back to the user.
26. I know I get frustrated when websites are slow
27. and I think most people do,
28. so it makes sense that our customer
29. wants to solve this problem.
30. We have already decoupled the application
31. by separating the business logic from the downstream calls,
32. but we haven't addressed the synchronous aspect of this
33. that is causing the latency for the end user.
34. This especially is a problem during scaling events,
35. when the backend starts rejecting requests
36. because it's overwhelmed.
37. I'm going to recommend to our customer
38. that they move to an asynchronous architecture,
39. where the request would come in,
40. API Gateway would accept and validate the request,
41. and then, from there,
42. instead of sending the request directly to Lambda,
43. it would send it to storage first.
44. This is following a storage-first pattern,
45. which is used to reduce API latency.
46. This is how it works.
47. Usually, you have this type of setup
48. where you have your API,
49. which sends to your compute,
50. which then processes, writes to storage,
51. and then responds to the API layer,
52. which then sends that response to the client.
53. This is where we are at now,
54. which is causing some latency
55. while the processing is being done in the compute layer.
56. What we want to do instead,
57. is move the storage up closer to the API,
58. so we can instead have the API accept the request,
59. immediately write the data to storage somewhere,
60. and then respond back to the user quickly and say,
61. "Hey, we got the data needed.
62. You can move on."
63. Then, the compute component can read that data from storage
64. and do its work asynchronously without the user waiting.
65. This is a nice option when working with API Gateway
66. for a few reasons.
67. First of all, API Gateway can handle a lot for you,
68. like authenticating requests,
69. as well as validating and transforming payloads
70. before they ever reach your compute layer.
71. So because API Gateway is such a powerful service,
72. you can offload some of that work
73. from your backend compute layer to your API.
74. All of that logic happens closer to the client,
75. which can reduce some latency, there.
76. The second piece needed
77. for this storage-first thing to work,
78. is the fact that API Gateway can directly integrate
79. with other AWS services,
80. meaning that API Gateway can send the appropriate API calls
81. to other AWS services.
82. It knows how to interact with other AWS APIs,
83. so API Gateway can do things like write an item to DynamoDB,
84. or add a message to an SQS queue,
85. or publish a message to SNS.
86. API Gateway can do all of this
87. without you needing to write something
88. like a Lambda function to do it.
89. So because of these features,
90. you can store your event first, respond to your user,
91. then your compute layer can process asynchronously.
92. This sounds like it'll fix our user's issues,
93. but right now, our architecture looks like this.
94. We have API Gateway talking directly to Lambda.
95. So what we need to do, is make some space
96. between these two components,
97. and add a storage layer between them to create a buffer.
98. This means that we can follow the storage-first pattern.
99. I'm thinking about either using DynamoDB
100. as the storage option here, or Amazon SQS,
101. or Simple Queue Service.
102. First, let's explore DynamoDB.
103. This option would seem to make a lot of sense here
104. because we are already storing our data in DynamoDB.
105. So theoretically, since API Gateway
106. is doing the authentication
107. and the data validation of the order,
108. and API Gateway has a service integration with DynamoDB,
109. I'm thinking I can maybe just get rid of the Lambda function
110. that is accepting and writing the orders
111. to DynamoDB altogether.
112. This idea hinges on the Lambda service
113. not doing any real business processing.
114. If there is business logic happening
115. in that Lambda function, then this won't work.
116. But if there isn't,
117. then this is a pretty interesting idea,
118. and one I'm keeping in mind.
119. Now for our other option, Amazon SQS.
120. Amazon SQS is a service
121. that is a fully managed message queuing service
122. that enables you to decouple and scale microservices,
123. distributed systems, and serverless applications.
124. The way it works is you create a queue,
125. and this queue then acts as a place to send messages
126. that go between system components.
127. In our case, the message would be sent to the queue
128. from API Gateway,
129. and it would then stay in the queue
130. until that message was processed and deleted.
131. SQS is a supported event source by AWS Lambda,
132. so this would work pretty seamlessly
133. from an integration standpoint.
134. I think that both of these options seem great.
135. They're both decoupling the API from the backend logic,
136. which would help with this latency issue.
137. Both options also increase resilience,
138. so that if there was a large scaling event,
139. even if Lambda is scaled up to any limits
140. set by our customer, the orders would be stored somewhere
141. waiting to be processed.
142. So, no orders would be dropped at any point.
143. Both options are also serverless,
144. which is important for our customer,
145. who wants to reduce operational overhead and cost,
146. and also wants to optimize for performance efficiency.
147. With serverless, you scale up and down on demand,
148. so performance efficiency
149. is really built into both of these solutions.
150. Let's go ahead and call our customer
151. and see if they can give us some more information
152. so that we can make this decision.
153. (phone rings)
154. - [Raf] Oh, hello.
155. Hello, Morgan.
156. I'm just in the lab here testing out some,
157. some new gloves, a products that we are going to add
158. to our website.
159. What's up?
160. - [Morgan] Oh, nice.
161. That's a little bit unusual, having a tech worker
162. also working directly with the products you sell,
163. but that's pretty cool.
164. - [Raf] Yeah, we are pretty much hands-on all around here.
165. Anyway, how is it going?
166. - [Morgan] Well, it's going pretty well.
167. I'm almost done designing the solution
168. for your order service,
169. but I do have a question for you
170. that would help provide me with some clarity.
171. So, for this order service,
172. is the service performing any business logic
173. beyond basic data validation or storing the order?
174. - [Raf] Uh-hmm, yeah.
175. Well, it is.
176. It does have some complex business logic
177. beyond just validating and passing data
178. from one place to another.
179. So yeah, the answer is yes, it does.
180. - [Morgan] Okay, well, that is really all I needed to know.
181. But before I let you go,
182. I really love those gloves that you have there.
183. - [Raf] Yeah, cool.
184. I'll send you a pair.
185. - [Morgan] All right, great, thank you.
186. - [Raf] Bye-bye, thank you.
187. - [Morgan] All right, well, that settles it for me.
188. Since we cannot remove the Lambda function
189. doing the processing from the solution,
190. we should use Amazon SQS as the decoupling component
191. between API Gateway and AWS Lambda.
192. Now, our solution will look like this.
193. We have API Gateway accepting the request from the frontend
194. and sending it to SQS,
195. then immediately responding to the user.
196. From there, AWS Lambda will recognize
197. that there is an event in the queue,
198. run the business logic and store the information
199. in the DynamoDB table.
200. A stream entry will then be created,
201. which then triggers another Lambda function.
202. And this Lambda function will then call SNS
203. to publish a message to the downstream services.
204. It's time to finalize this architecture
205. and prepare to present it to our customer.
206. [End of transcript. Skip to the start.](https://courses.edx.org/xblock/block-v1:AWS+AWS-ARCH-1+3T2022+type@vertical+block@db6e843cfae4401e9571359097015e09?exam_access=&recheck_access=1&show_bookmark=0&show_title=0&view=student_view#transcript-start-312502f7db544810a77c77db05457fec)

### Downloads and transcripts

#### Video

1. - [Morgan] In this video,
2. we are going to explore the types
3. of configurations that we can make for an SQS queue.
4. So, I will first navigate to the SQS service.
5. And from this page, I can go ahead
6. and click the Create queue button.
7. From here, we can see the types of configuration details
8. that you can make for a queue,
9. and notice how this is already looking fairly similar
10. to the video that we made for the SNS exploration.
11. So there are differences between SQS and SNS,
12. and I highly recommend
13. that you understand those differences deeply.
14. Notice, though, how we can make a standard SQS queue,
15. or a FIFO, or first-in, first-out, queue.
16. So, that is very similar to the way that SNS worked.
17. Now, that being said, with SQS
18. what happens, is whenever you create your messages,
19. they remain in the queue until they are processed.
20. Whereas with SNS messages that you send through
21. just get sent directly to the subscriber.
22. Whether the subscriber successfully received it
23. or not is kind-of on the end of the subscriber.
24. Whereas with SQS, if you send a message
25. into a standard queue, for example, that message will remain
26. in that queue until it is processed.
27. So now, I'm going to go ahead and give this queue a name,
28. and I will go ahead and just name it test-queue.
29. And remember FIFO - first-in, first-out - that means
30. that you have preserved message ordering in that queue.
31. All right, so now scrolling down, there's some
32. configurations that are important to know for SQS.
33. And the first one that I want to call out,
34. is this maximum message size.
35. Messages can be between one kilobyte and 256 kilobytes.
36. So this is really not for large messages.
37. This is for small messages
38. that you're sending between components.
39. You would have to come up with a different solution
40. if you wanted to send large messages
41. between different components using a queue.
42. One way you could do that is maybe
43. by sending an index or a reference to where
44. the next component can go find the message, like a database,
45. or something like that.
46. But the maximum message size is 256 kilobytes.
47. The next thing I want to show, here,
48. is this message retention period.
49. And it can be between 1 minute or 14 days.
50. So, your messages will remain in your queue
51. until they're processed, up until 14 days, or whatever
52. you want to configure here.
53. We'll go ahead and make it 14 days,
54. the maximum amount of time for message retention.
55. Then, we have this receive message wait time.
56. And if we click on Info here, it will tell us
57. some information about this.
58. So, the receive message wait time is the maximum amount
59. of time that polling will wait for messages
60. to become available to receive.
61. The minimum value is 0 and the maximum is 20.
62. So with SQS, we have something called polling.
63. So, SQS is a polling mechanism, where the consumers
64. have to poll the queue to say:
65. Hey, do you have any messages for me yet?
66. Do you have any messages for me yet?
67. And so, whenever SQS receives a poll,
68. it can take a minimum
69. of 0 seconds to respond, or a maximum of 20.
70. If you have a low-volume queue, you may want
71. to have a longer received message wait time.
72. And that would reduce the amount
73. of times that you're polling an empty queue.
74. So, if you have a low-volume queue, maybe use long polling.
75. If you have a high-volume queue,
76. you can probably use short polling.
77. It just depends on the type of messages that you have.
78. Now, the other one I want to call out, here,
79. is the visibility timeout.
80. Visibility timeout sets the length of time that a message
81. received from a queue or by one consumer will not be visible
82. to the other message consumers.
83. And so, what this is used for, is let's say you have
84. five different consumers
85. all polling the same queue and processing the messages.
86. Let's say you have five EC2 instances
87. all using the same queue.
88. You want to make sure that all five EC2 instances,
89. aren't processing the exact same message
90. because then you would have duplication
91. of that message being processed.
92. So, whenever a message is claimed from a queue,
93. the visibility timeout begins.
94. And during that time,
95. no other consumers can receive that message.
96. So, the visibility timeout is another thing
97. that's very important to understand how it works.
98. If we can keep scrolling down here, you can see
99. that we have another opportunity to set a resource policy
100. on this SQS queue.
101. So, this will determine who can send messages
102. to the queue and who can receive messages from the queue.
103. And this is a resource policy.
104. And then, I'm going to just go ahead and click Create queue.
105. And then from here, we can view some other information
106. once the queue is completed.
107. You can see if you want to create any SNS subscriptions.
108. So, if we click info here,
109. you can subscribe one or more SQS queues to an SNS topic.
110. So, the SNS topic could send messages to the queue.
111. And then, we also have things like Lambda triggers.
112. So, is this queue going to be triggering
113. a Lambda function, right?
114. And then, you can also view monitoring through CloudWatch.
115. And this is going to be very interesting.
116. If you have operational issues, you could come in here
117. and you could see potentially, let's say, you have a bunch
118. of messages that are showing up and flooding the queue.
119. And you notice that the number
120. of messages deleted is really low.
121. Maybe that means that you have a problem with your code,
122. whether they aren't deleting the messages
123. after they are processed.
124. So, that is all I want to cover
125. for SQS during this exploratory video.
126. And as always, please check out the readings,
127. where you can find links
128. to tutorials and blogs to learn more about this service.
129. [End of transcript. Skip to the start.](https://courses.edx.org/xblock/block-v1:AWS+AWS-ARCH-1+3T2022+type@vertical+block@62f5496ed7eb4450bb247e130b13f094?exam_access=&recheck_access=1&show_bookmark=0&show_title=0&view=student_view#transcript-start-848dfd015e5444518c559c565c5ee0b6)

# Decoupling solutions on AWS

This week’s customer had issues with latency for their end users because they were using a synchronous model for their web backend. Morgan suggested that they migrate to an asynchronous model, where the order is first stored, and then processed shortly after. With this model, end user can receive a response more quickly from the backend. When the backend system receives an order, it can immediately respond and then process the request asynchronously.   
  
A loosely coupled architecture minimizes the bottlenecks that are caused by synchronous communication, latency, and I/O operations. Amazon Simple Queue Service (Amazon SQS) and AWS Lambda are often used to implement asynchronous communication between different services.  
  
You should consider using this pattern if you have the following requirements:

* You want to create loosely coupled architecture.
* All operations don’t need to be completed in a single transaction, and some operations can be asynchronous.
* The downstream system can’t handle the incoming transactions per second (TPS) rate. The messages can be written to the queue and processed based on the availability of resources.

A disadvantage of this pattern is that the actions of business transaction are synchronous. Even though the calling system receives a response, some part of the transaction might still continue to be processed by downstream systems.  
  
For more information, see [Patterns for integrating microservices](https://docs.aws.amazon.com/prescriptive-guidance/latest/modernization-integrating-microservices/integrating-patterns.html).

## Building storage-first applications

Building a storage-first application can help an application be more reliable when working with events. You can read more about this topic in the blog post [Building storage-first serverless applications with HTTP APIs service integrations](https://aws.amazon.com/blogs/compute/building-storage-first-applications-with-http-apis-service-integrations/).

## Amazon SQS

Amazon SQS is a fully managed, message queuing service that you can use to decouple and scale microservices, distributed systems, and serverless applications. Amazon SQS reduces the complexity and overhead associated with managing and operating message-oriented middleware, which means that developers can focus on differentiating work. By using Amazon SQS, you can send, store, and receive messages between software components at virtually any volume, without losing messages or requiring other services to be available.  
  
When you [create](https://docs.aws.amazon.com/AWSSimpleQueueService/latest/SQSDeveloperGuide/sqs-configure-create-queue.html) or [edit](https://docs.aws.amazon.com/AWSSimpleQueueService/latest/SQSDeveloperGuide/sqs-configure-edit-queue.html) a queue, you can configure the following parameters:

* Visibility timeout: The length of time that a message received from a queue (by one consumer) won't be visible to the other message consumers.
* Message retention period: The amount of time that Amazon SQS retains messages that remain in the queue. By default, the queue retains messages for 4 days. You can configure a queue to retain messages for up to 14 days.
* Delivery delay: The amount of time that Amazon SQS will delay before it delivers a message that is added to the queue. For more information, see [Amazon SQS delay queues](https://docs.aws.amazon.com/AWSSimpleQueueService/latest/SQSDeveloperGuide/sqs-delay-queues.html).
* Maximum message size: The maximum message size for the queue.
* Receive message wait time: The maximum amount of time that Amazon SQS waits for messages to become available after the queue gets a receive request.
* Enable content-based deduplication: Amazon SQS can automatically create deduplication IDs based on the body of the message.
* Enable high throughput FIFO: This feature enables high throughput for messages in the queue. Choosing this option changes the related options (deduplication scope and FIFO throughput limit) to the required settings for enabling high throughput for FIFO queues.
* Redrive allow policy: This policy defines which source queues can use this queue as the dead-letter queue.

### Short polling compared to long polling

When you consume messages from a queue by using short polling, Amazon SQS samples a subset of its servers (based on a weighted random distribution) and returns messages from only those servers. Thus, a particular ReceiveMessage request might not return all of your messages. However, if you have fewer than 1,000 messages in your queue, a subsequent request will return your messages. If you keep consuming from your queues, Amazon SQS samples all of its servers, and you receive all of your messages.  
  
The following diagram shows the short-polling behavior of messages returned from a standard queue after one of your system components makes a receive request. Amazon SQS samples several of its servers (in gray) and returns messages A, C, D, and B from these servers. Message E isn't returned for this request, but it's returned for a subsequent request.

When the wait time for the ReceiveMessage API action is greater than 0, *long polling* is in effect. The maximum long polling wait time is 20 seconds. Long polling helps reduce the cost of using Amazon SQS by reducing the number of empty responses (when there are no messages available for a ReceiveMessage request) and false empty responses (when messages are available, but aren't included in a response). For information about enabling long polling for a new or existing queue using the Amazon SQS console, see [Configuring queue parameters (console)](https://docs.aws.amazon.com/AWSSimpleQueueService/latest/SQSDeveloperGuide/sqs-configure-queue-parameters.html). For best practices, see [Setting up long polling](https://docs.aws.amazon.com/AWSSimpleQueueService/latest/SQSDeveloperGuide/working-with-messages.html#setting-up-long-polling).   
  
Long polling offers the following benefits:

* Reduces empty responses by letting Amazon SQS wait until a message is available in a queue before it sends a response. Unless the connection times out, the response to the ReceiveMessage request contains at least one of the available messages, up to the maximum number of messages specified in the ReceiveMessage action. In rare cases, you might receive empty responses even when a queue still contains messages, especially if you specify a low value for the ReceiveMessageWaitTimeSeconds parameter.
* Reduces false empty responses by querying all—instead of a subset of—Amazon SQS servers.
* Returns messages as soon as they become available.

### Resources

* For more information about Amazon SQS, see [What is Amazon Simple Queue Service?](https://docs.aws.amazon.com/AWSSimpleQueueService/latest/SQSDeveloperGuide/welcome.html) or [Amazon SQS FAQs](https://aws.amazon.com/sqs/faqs/).
* For a tutorial about setting up Amazon SQS to invoke Lambda functions, see [Using AWS Lambda with Amazon SQS](https://docs.aws.amazon.com/lambda/latest/dg/with-sqs.html).

# Decoupling solutions on AWS

This week’s customer had issues with latency for their end users because they were using a synchronous model for their web backend. Morgan suggested that they migrate to an asynchronous model, where the order is first stored, and then processed shortly after. With this model, end user can receive a response more quickly from the backend. When the backend system receives an order, it can immediately respond and then process the request asynchronously.   
  
A loosely coupled architecture minimizes the bottlenecks that are caused by synchronous communication, latency, and I/O operations. Amazon Simple Queue Service (Amazon SQS) and AWS Lambda are often used to implement asynchronous communication between different services.  
  
You should consider using this pattern if you have the following requirements:

* You want to create loosely coupled architecture.
* All operations don’t need to be completed in a single transaction, and some operations can be asynchronous.
* The downstream system can’t handle the incoming transactions per second (TPS) rate. The messages can be written to the queue and processed based on the availability of resources.

A disadvantage of this pattern is that the actions of business transaction are synchronous. Even though the calling system receives a response, some part of the transaction might still continue to be processed by downstream systems.  
  
For more information, see [Patterns for integrating microservices](https://docs.aws.amazon.com/prescriptive-guidance/latest/modernization-integrating-microservices/integrating-patterns.html).

## Building storage-first applications

Building a storage-first application can help an application be more reliable when working with events. You can read more about this topic in the blog post [Building storage-first serverless applications with HTTP APIs service integrations](https://aws.amazon.com/blogs/compute/building-storage-first-applications-with-http-apis-service-integrations/).

## Amazon SQS

Amazon SQS is a fully managed, message queuing service that you can use to decouple and scale microservices, distributed systems, and serverless applications. Amazon SQS reduces the complexity and overhead associated with managing and operating message-oriented middleware, which means that developers can focus on differentiating work. By using Amazon SQS, you can send, store, and receive messages between software components at virtually any volume, without losing messages or requiring other services to be available.  
  
When you [create](https://docs.aws.amazon.com/AWSSimpleQueueService/latest/SQSDeveloperGuide/sqs-configure-create-queue.html) or [edit](https://docs.aws.amazon.com/AWSSimpleQueueService/latest/SQSDeveloperGuide/sqs-configure-edit-queue.html) a queue, you can configure the following parameters:

* Visibility timeout: The length of time that a message received from a queue (by one consumer) won't be visible to the other message consumers.
* Message retention period: The amount of time that Amazon SQS retains messages that remain in the queue. By default, the queue retains messages for 4 days. You can configure a queue to retain messages for up to 14 days.
* Delivery delay: The amount of time that Amazon SQS will delay before it delivers a message that is added to the queue. For more information, see [Amazon SQS delay queues](https://docs.aws.amazon.com/AWSSimpleQueueService/latest/SQSDeveloperGuide/sqs-delay-queues.html).
* Maximum message size: The maximum message size for the queue.
* Receive message wait time: The maximum amount of time that Amazon SQS waits for messages to become available after the queue gets a receive request.
* Enable content-based deduplication: Amazon SQS can automatically create deduplication IDs based on the body of the message.
* Enable high throughput FIFO: This feature enables high throughput for messages in the queue. Choosing this option changes the related options (deduplication scope and FIFO throughput limit) to the required settings for enabling high throughput for FIFO queues.
* Redrive allow policy: This policy defines which source queues can use this queue as the dead-letter queue.

### Short polling compared to long polling

When you consume messages from a queue by using short polling, Amazon SQS samples a subset of its servers (based on a weighted random distribution) and returns messages from only those servers. Thus, a particular ReceiveMessage request might not return all of your messages. However, if you have fewer than 1,000 messages in your queue, a subsequent request will return your messages. If you keep consuming from your queues, Amazon SQS samples all of its servers, and you receive all of your messages.  
  
The following diagram shows the short-polling behavior of messages returned from a standard queue after one of your system components makes a receive request. Amazon SQS samples several of its servers (in gray) and returns messages A, C, D, and B from these servers. Message E isn't returned for this request, but it's returned for a subsequent request.

When the wait time for the ReceiveMessage API action is greater than 0, *long polling* is in effect. The maximum long polling wait time is 20 seconds. Long polling helps reduce the cost of using Amazon SQS by reducing the number of empty responses (when there are no messages available for a ReceiveMessage request) and false empty responses (when messages are available, but aren't included in a response). For information about enabling long polling for a new or existing queue using the Amazon SQS console, see [Configuring queue parameters (console)](https://docs.aws.amazon.com/AWSSimpleQueueService/latest/SQSDeveloperGuide/sqs-configure-queue-parameters.html). For best practices, see [Setting up long polling](https://docs.aws.amazon.com/AWSSimpleQueueService/latest/SQSDeveloperGuide/working-with-messages.html#setting-up-long-polling).   
  
Long polling offers the following benefits:

* Reduces empty responses by letting Amazon SQS wait until a message is available in a queue before it sends a response. Unless the connection times out, the response to the ReceiveMessage request contains at least one of the available messages, up to the maximum number of messages specified in the ReceiveMessage action. In rare cases, you might receive empty responses even when a queue still contains messages, especially if you specify a low value for the ReceiveMessageWaitTimeSeconds parameter.
* Reduces false empty responses by querying all—instead of a subset of—Amazon SQS servers.
* Returns messages as soon as they become available.

### Resources

* For more information about Amazon SQS, see [What is Amazon Simple Queue Service?](https://docs.aws.amazon.com/AWSSimpleQueueService/latest/SQSDeveloperGuide/welcome.html) or [Amazon SQS FAQs](https://aws.amazon.com/sqs/faqs/).
* For a tutorial about setting up Amazon SQS to invoke Lambda functions, see [Using AWS Lambda with Amazon SQS](https://docs.aws.amazon.com/lambda/latest/dg/with-sqs.html).

<https://docs.aws.amazon.com/lambda/latest/dg/with-sqs.html>

[https://docsYou are a solutions architect who is working with a customer, a startup travel company, to design solutions on AWS to meet their business needs. The startup has a travel website that helps users search for deals on flights and hotels based on data that the user inputs on the website. Currently, the code for the startup’s application has been written and they want to host a web backend for one of their APIs, which is used for making queries against their data. This web backend must accept requests from external clients that make HTTPS calls. The clients will make GET requests from the frontend. These requests will require complex Structured Query Language (SQL) queries against the data. The customer plans to use a relational database, such as MySQL, to store this data. This customer also wants to use serverless technologies when possible. Write a few paragraphs that answer the following questions:](https://docs.aws.amazon.com/AWSSimpleQueueService/latest/SQSDeveloperGuide/welcome.html)

* [What components are needed to host a web backend on AWS? (For example: compute service, database service, or others)](https://docs.aws.amazon.com/AWSSimpleQueueService/latest/SQSDeveloperGuide/welcome.html)
* [What AWS services would you consider using to build this solution?](https://docs.aws.amazon.com/AWSSimpleQueueService/latest/SQSDeveloperGuide/welcome.html)
* [What follow-up questions would you ask this customer to get more information about their requirements so that you could better determine which services to choose?](https://docs.aws.amazon.com/AWSSimpleQueueService/latest/SQSDeveloperGuide/welcome.html)

[.aws.amazon.com/AWSSimpleQueueService/latest/SQSDeveloperGuide/welcome.html](https://docs.aws.amazon.com/AWSSimpleQueueService/latest/SQSDeveloperGuide/welcome.html)

* For more information about Amazon DynamoDB Streams, see [Change data capture for DynamoDB Streams.](https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/Streams.html)
* For a tutorial about how to process DynamoDB streams with an AWS Lambda function, see [Tutorial: Process new items with DynamoDB Streams and Lambda](https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/Streams.Lambda.Tutorial.html)

<https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/Streams.html>

<https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/Streams.Lambda.Tutorial.html>

<https://home.edx.org/>

<https://home.edx.org/>

sns

sqs

1. [Start of transcript. Skip to the end.](https://courses.edx.org/xblock/block-v1:AWS+AWS-ARCH-1+3T2022+type@vertical+block@b36719869871424fbd818a3bda7966bf?exam_access=&format=Quiz&recheck_access=1&show_bookmark=0&show_title=0&view=student_view#transcript-end-71b19e768d35459da2d49d52068b390c)
2. - [Morgan] Great news.
3. We are ready to present our solution
4. for the order service to our customer.
5. Now, it's good to remember,
6. before we get into our customer meeting,
7. that this initial design will likely change
8. as the solution is implemented
9. and more requirements come to light.
10. I, personally, have never been
11. on a software or IT-related project
12. that didn't change as it went along.
13. There are still many unknowns
14. at this point in the process.
15. That being said, we have a really great solution
16. that meets the current requirements,
17. and I think our customer will be excited to move forward
18. with building it out.
19. And the next phase for this would also likely involve
20. building a proof of concept for our customer.
21. Let's go ahead and give them a call.
22. (phone ringing)
23. - [Raf] Hello, Morgan.
24. I hear you have our solution designed.
25. I'm looking forward to see it.
26. - [Morgan] Yes, I do.
27. Let me go ahead and share it with you.
28. Let's dive in.
29. I approach this as a greenfield service,
30. knowing that you are open to totally rewriting the code.
31. So with that in mind,
32. try to forget about the way things are currently done,
33. and instead let's focus on this solution
34. and how it will meet your requirements.
35. - [Raf] All right, great.
36. - [Morgan] Okay.
37. So you have the frontend clients making requests,
38. sending in orders through their phones, browsers, et cetera.
39. All these requests are going to be directed
40. to Amazon API Gateway,
41. which acts as the front door for your API.
42. API Gateway will handle the authentication for the request,
43. and it will also validate the format
44. of the incoming request
45. to verify that all the necessary fields
46. are included in the payload of the request.
47. Once it passes through the authentication and validation,
48. API Gateway will then send the message to an SQS queue.
49. The message will remain in the queue
50. until an AWS Lambda function
51. is spun up to process the message.
52. This happens quickly, and it's all automated
53. because there is a polling mechanism built into AWS Lambda
54. that will read the messages from the queue.
55. Putting SQS between API Gateway and Lambda
56. is decoupling the API from compute.
57. So that way, if you have a large scaling event
58. and you reach any predefined limits for the Lambda,
59. the messages will be in the queue and will not be lost.
60. Then, Lambda can churn through the messages and catch up.
61. Or to catch up even faster,
62. you can raise your Lambda limits
63. and process the messages that are in the queue.
64. Now, for the Lambda function.
65. This Lambda function will contain the application code
66. for the order service related to order processing
67. and storing the orders.
68. The orders will be stored in an Amazon DynamoDB table.
69. Once an order is stored in the table,
70. an entry gets added to the DynamoDB stream,
71. which then will need to be processed
72. to send the order to the downstream functions.
73. This is where another Lambda function comes into play.
74. And this Lambda function
75. will read the information on the DynamoDB stream
76. and publish the order information to Amazon SNS.
77. SNS, then following a fan-out pattern
78. that will send that message
79. to all of the subscribed endpoints.
80. Those endpoints being the three downstream services
81. for fulfillment, accounting, and inventory.
82. - [Raf] Wow! Okay, this looks awesome.
83. But I have a question for you.
84. What about monitoring and logging?
85. - [Morgan] Yeah, so all of these services
86. are serverless in nature,
87. and have built-in integrations to send metrics and logs
88. to Amazon CloudWatch and Amazon CloudWatch Logs.
89. - [Raf] Right.
90. It's good to hear that we won't need to do
91. a ton of extra configuration to that,
92. get working and in place.
93. What about scaling?
94. Is everything here using managed scaling?
95. - [Morgan] Yes, so the scaling is all managed
96. for these services.
97. Though, for some of them,
98. you can set limits or rates,
99. like scaling with Lambda
100. or managing the throughput for DynamoDB.
101. So, you can still control scaling events,
102. but it's much more hands-off
103. than when you compare it to services like EC2, for example.
104. - [Raf] Cool. Yeah, that's a relief.
105. Hopefully, I won't get paged
106. for outages anymore (chuckles)
107. when we have sales in our store.
108. That was always the worst.
109. Like being excited for a sales event to happen,
110. only for our infrastructure to fail
111. when we needed it most, right?
112. - [Morgan] Yeah, I totally understand that.
113. Now, for your other requirements that we went over,
114. we designed this with decoupling components, top of mind,
115. and with services that scale in
116. when you are no longer using them
117. to optimize for cost and for performance efficiency.
118. So overall, I think you're going to be really happy
119. with operating this solution over time.
120. - [Raf] Thank you so much.
121. I can't wait to take this back to the team
122. and show them what we will be working next.
123. - [Morgan] You're welcome.
124. And I am sure we'll be chatting soon
125. as you begin to dive into the details
126. on how all of this will actually work.
127. [End of transcript. Skip to the start.](https://courses.edx.org/xblock/block-v1:AWS+AWS-ARCH-1+3T2022+type@vertical+block@b36719869871424fbd818a3bda7966bf?exam_access=&format=Quiz&recheck_access=1&show_bookmark=0&show_title=0&view=student_view#transcript-start-71b19e768d35459da2d49d52068b390c)

{

"Version": "2012-10-17",

"Statement": [

{

"Effect": "Allow",

"Action": [

"sts:AssumeRole"

],

"Principal": {

"Service": [

"lambda.amazonaws.com"

]

}

}

]

}

1. [Start of transcript. Skip to the end.](https://courses.edx.org/xblock/block-v1:AWS+AWS-ARCH-1+3T2022+type@vertical+block@3dedd8d93ef44d19b86510fcae3d71d0?exam_access=&format=Quiz&recheck_access=1&show_bookmark=0&show_title=0&view=student_view#transcript-end-64b576a3c4054e9f9d84f375cd3acd63)
2. - [Morgan] All right, so in this video, I'm here with Raf,
3. he's a solutions architect now,
4. so we both have our solutions architects' hats on.
5. And what we're going to do, is we're going to take a look
6. at the architecture that we built this week.
7. And we're going to think about ways
8. that we could enhance this architecture,
9. either to be aligned with some of the things
10. that our customers were looking for,
11. like cost or performance, or maybe operational overhead,
12. but we're also going to look at things like,
13. just in general, what types of enhancements
14. could we make to these architectures?
15. So we'll be doing one video like this
16. at the end of every week, with four different architectures.
17. So, that being said,
18. Raf, why don't you go ahead and kick us off.
19. - [Raf] All right, ready to start here.
20. So the first architecture improvements that I would make
21. in this architecture is implementing DAX.
22. DAX is a cache layer for, it's DynamoDB Accelerator.
23. It's a caching layer for DynamoDB.
24. So, you can implement that and the Lambda functions.
25. Whenever going to DynamoDB, the Lambda function can use DAX,
26. which is DynamoDB Accelerator, so it caches content
27. so you don't have database hits all the time
28. on the DynamoDB table.
29. - [Morgan] Right, so with DynamoDB Accelerator,
30. it can get you to, like, microsecond
31. - [Raf] Microsecond latency
32. - [Raf] is super fast, - [Morgan] level response.
33. - [Raf] but it comes with a cost.
34. - [Morgan] It does.
35. - [Raf] Right, so depending on what the customer
36. wants to optimize for, you may use a DAX layer
37. here, between Lambda functions and DynamoDB,
38. or you could just use DynamoDB with read capacity units.
39. That would suffice as well.
40. - [Morgan] Right. It really depends on the goals
41. - [Morgan] of the customer. - [Raf] Yes.
42. - [Morgan] So, I would say DynamoDB accelerator is something
43. that you would introduce after you've determined
44. that the performance
45. - [Morgan] isn't good enough. - [Raf] Yes, yes.
46. - [Morgan] The other thing you could do with DynamoDB,
47. in terms of performance, is you could potentially
48. remodel your table
49. - [Morgan] with the indexes. - [Raf] Yes.
50. - [Raf] Yes, DynamoDB tuning is a big part of infrastructure
51. optimization, right?
52. And you can experiment with, with DAX, without DAX.
53. So you can, you can do tests.
54. If you're have infrastructure as code, you can even
55. mimic that entire environment
56. and do stress tests with DAX, without DAX.
57. Compare them.
58. - [Morgan] Do some A/B testing there.
59. - [Raf] Yes.
60. - [Morgan] All right, so the next thing
61. that I would recommend
62. is going to be for AWS Lambda.
63. We have our Lambda functions, here.
64. And so, with Lambda, you get to choose the amount
65. of CPU and memory that you have allocated
66. for your Lambda functions.
67. And so, something that is kind-of counterintuitive,
68. is that if you up the amount of memory
69. that you give a Lambda function,
70. technically, you're paying more for that Lambda function.
71. However, the code can oftentimes run faster
72. on Lambda functions that have more memory.
73. So what you can do, is you can use a library
74. called AWS Lambda Power Tuning,
75. and that will help you find the most optimal memory
76. and CPU allocation for your Lambda functions.
77. And that's going to help you have your Lambda functions
78. hopefully run more performantly,
79. and then, actually lower your costs.
80. So it's kind-of, it's counterintuitive
81. because the more memory that you give,
82. you can actually achieve lower costs
83. because your Lambda function runs for less time.
84. And with Lambda, you get charged per time.
85. - [Raf] Yes. Yes.
86. That's a good one.
87. Another optimization on the Lambdas, as well,
88. could be using a feature called Lambda layers.
89. Lambda layers allow you to get code that is reused
90. across multiple Lambda functions.
91. In this case, you only have two,
92. but as this architecture grow,
93. you may want to have more Lambda functions,
94. and you may potentially have code that is written
95. and replicated across multiple Lambda functions.
96. - [Morgan] Like, common libraries and,
97. - [Raf] Common libraries, exactly.
98. So you can get these and add a Lambda layer for that.
99. And you can, and you can reference
100. the Lambda layer in the Lambda function.
101. So, if you wants to change that library,
102. you don't need to change every single Lambda function
103. that has, that uses that code.
104. You just change the layer.
105. - [Morgan] Yeah. It also helps with faster code deployments.
106. So, it can speed up deploying new code
107. to your Lambda functions, as well,
108. because you only would deploy a small amount of code,
109. not the full amount
110. with the libraries included.
111. - [Raf] Yes. - [Morgan] And so continuing on
112. with the Lambda functions here, as well.
113. Another thing you can do, is look at
114. the code and the way the code is written.
115. So with Lambda, you have this handler,
116. and that's where the execution begins for the program.
117. But what you can do, is you can declare variables
118. outside of that handler, and then
119. those variables can be reused across execution environments.
120. So for example, if you have to create some objects
121. every single time a Lambda function is run,
122. you can declare those outside of the handler.
123. And then, if this one has, let's say 100 executions,
124. you could reuse that same variable across them.
125. So that would help you have
126. your code initialization be shorter.
127. So, that's a little bit of a programmer one,
128. and I'm a software developer myself,
129. so I can't help myself.
130. But yeah, so that's what I have to say for Lambda on that.
131. - [Raf] Yes, another enhancements I would suggest
132. was potentially, in the future,
133. if there are more needs to do filtering using Amazon SNS,
134. you can replace Amazon SNS by Amazon EventBridge.
135. EventBridge can use SNS as one of the consumers,
136. as one of the of the consumers of the data
137. that gets into the event bus in Amazon EventBridge.
138. And, in addition, you could have more than one.
139. So with EventBridge, you could, you could do filtering,
140. and you could have more than one consumer.
141. So, if you wants to get data from this Lambda function
142. to an SNS topic and an SQS queue, for example,
143. you can put EventBridge in the middle.
144. It would also integrate with software as a service,
145. like SaaS tools from AWS Partners and
146. and other integration parts, as well.
147. - [Morgan] And with the message filtering for SNS,
148. you can only filter based off of message attributes.
149. So, if you needed to do like, deeper, more specific
150. message filtering, or like, let's say,
151. the messages coming through weren't uniform,
152. then it would be a good idea to upgrade
153. with something like EventBridge.
154. And then, one final thing back to our Lambda functions here,
155. is we could optimize logging output and retention
156. for our logs from Lambda.
157. So, these Lambda functions are sending logs
158. to Amazon CloudWatch Logs,
159. and you could consider using something
160. like AWS Lambda Powertools.
161. And this is a suite of utilities for AWS Lambda functions,
162. where you can ease adopting the best practices for Lambda,
163. such as tracing using things like X-Ray,
164. or structured logging, creating custom metrics,
165. things like that.
166. So, using the AWS Powertools utility,
167. you can optimize your Lambda functions
168. on that level, as well.
169. - [Raf] All right, that's good.
170. Thanks, Morgan, for helping our students understand,
171. and how to mock-up an architecture
172. for a serverless web backend.
173. I think that was a strong week
174. that helped our customers a lot on understanding
175. and decoupling event-driven architectures.
176. [End of transcript. Skip to the start.](https://courses.edx.org/xblock/block-v1:AWS+AWS-ARCH-1+3T2022+type@vertical+block@3dedd8d93ef44d19b86510fcae3d71d0?exam_access=&format=Quiz&recheck_access=1&show_bookmark=0&show_title=0&view=student_view#transcript-start-64b576a3c4054e9f9d84f375cd3acd63)

In this reading, you will find further information about the topics that Morgan and Raf talked about in the video where they both played Solutions Architects.

## Caching for Amazon DynamoDB by using Amazon DynamoDB Accelerator

Raf mentioned that if you need to improve Amazon DynamoDB performance to microsecond latency, to look into using Amazon DynamoDB Accelerator (DAX).   
  
DAX is a fully managed, highly available, in-memory [cache](https://aws.amazon.com/caching/) for DynamoDB that’s designed to deliver up to a 100-times performance improvement—from milliseconds to microseconds—even at millions of requests per second.  
  
DAX does the heavy lifting that’s required to add in-memory acceleration to your DynamoDB tables—and developers don’t need to manage cache invalidation, data population, or cluster management.  
  
A benefit of using DAX is that you don’t need to modify the application logic because DAX is compatible with existing DynamoDB API calls.  
  
DAX is designed to run within an Amazon Virtual Private Cloud (Amazon VPC) environment. Amazon VPC defines a virtual network that closely resembles a traditional data center. With a VPC, you have control over its IP address range, subnets, routing tables, network gateways, and security settings. You can launch a DAX cluster in your virtual network, and control access to the cluster by using Amazon VPC security groups.  
  
For more information about DAX, see [Amazon DynamoDB Accelerator.](https://aws.amazon.com/dynamodb/dax/)

## Optimizing AWS Lambda

### AWS Lambda Power Tuning

Morgan mentioned that you can fine-tune the memory or power configuration for your AWS Lambda functions to potentially increase performance and lower costs.  
  
[AWS Lambda Power Tuning](https://github.com/alexcasalboni/aws-lambda-power-tuning) is an open-source tool that helps you visualize and fine-tune the memory or power configuration of Lambda functions. It runs in your own AWS account, and it supports three optimization strategies: cost, speed, and balanced.  
  
AWS Lambda Power Tuning is a state machine that’s powered by AWS Step Functions. It helps you optimize your Lambda functions for cost or performance in a data-driven way.  
  
The state machine is designed to be easy to deploy and fast to execute. Also, it's language agnostic, so you can optimize any Lambda functions in your account.  
  
To work with AWS Lambda Power Turning, you provide a Lambda function Amazon Resource Name (ARN) as input. The state machine then invokes that function with multiple power configurations (from 128 MB to 10 GB—you decide which values). Then, it analyzes all the execution logs and suggests the best power configuration to minimize cost or maximize performance.  
  
Note that the input function will run in your AWS account, which means that it will perform HTTP requests, SDK calls, cold starts, and so on. The state machine also supports cross-Region invocations, and you can enable parallel execution to generate results in a few seconds.  
  
The state machine generates a visualization of average cost and speed for each power configuration.  
  
For example, the following diagram shows results for two CPU-intensive functions, which become both cheaper and faster with more power:  
For more information, see [AWS Lambda Power Tuning](https://github.com/alexcasalboni/aws-lambda-power-tuning).

### AWS Lambda Powertools

Morgan also mentioned that you can use another suite of tools called AWS Lambda Powertools to optimize your Lambda functions and use best practices. AWS Lambda Powertools is a suite of utilities for AWS Lambda functions that is designed to make it easier to adopt best practices such as tracing, structured logging, custom metrics, idempotency, batching, and more.   
  
For more information, see [AWS Lambda Powertools](https://awslabs.github.io/aws-lambda-powertools-python/latest/).

### AWS Lambda execution environment reuse

Another optimization technique that Morgan mentioned for Lambda is to move certain initialization tasks in your code so they are outside the handler. These tasks can then be reused across invocations (which is also known as execution environment reuse).  
  
You can take advantage of execution environment reuse to improve the performance of your function. To do this, initialize SDK clients and database connections outside of the function handler, and cache static assets locally in the /tmp directory. Subsequent invocations that are processed by the same instance of your function can reuse these resources. This reuse saves cost by reducing function run time.   
  
To avoid potential data leaks across invocations, don’t use the execution environment to store user data, events, or other information with security implications. If your function relies on a mutable state that can’t be stored in memory within the handler, consider creating a separate function or separate versions of a function for each user.  
  
To learn more, see [Best practices for working with AWS Lambda.](https://docs.aws.amazon.com/lambda/latest/dg/best-practices.html)

### Resources

* For more information about how to optimize serverless applications for cost, see [Building well-architected serverless applications: Optimizing application costs](https://aws.amazon.com/blogs/compute/building-well-architected-serverless-applications-optimizing-application-costs/) in the *AWS Compute Blog*.
* For more information about well-architected serverless applications, visit [Serverless Applications Lens - AWS Well-Architected Framework](https://docs.aws.amazon.com/wellarchitected/latest/serverless-applications-lens/welcome.html?did=wp_card&trk=wp_card).

<https://docs.aws.amazon.com/lambda/latest/dg/best-practices.html>

<https://aws.amazon.com/dynamodbaccelerator/>

<https://github.com/alexcasalboni/aws-lambda-power-tuning>

<https://docs.aws.amazon.com/lambda/latest/dg/best-practices.html>

<https://aws.amazon.com/blogs/compute/building-well-architected-serverless-applications-optimizing-application-costs/>

<https://docs.aws.amazon.com/wellarchitected/latest/serverless-applications-lens/welcome.html?did=wp_card&trk=wp_card>

--=======================

<https://learning.edx.org/course/course-v1:AWS+AWS-ARCH-1+3T2022/block-v1:AWS+AWS-ARCH-1+3T2022+type@sequential+block@97b4a06a6c8a42f09a37623385de1efd/block-v1:AWS+AWS-ARCH-1+3T2022+type@vertical+block@2b92367e7e894c9fa3ea27fbd55f2521>

<https://aws-tc-largeobjects.s3.us-west-2.amazonaws.com/DEV-AWS-MO-Architecting/exercise-1-serverless.html>

1. [Start of transcript. Skip to the end.](https://courses.edx.org/xblock/block-v1:AWS+AWS-ARCH-1+3T2022+type@vertical+block@c499c667f3314bee95b9f62768f95c4b?exam_access=&recheck_access=1&show_bookmark=0&show_title=0&view=student_view#transcript-end-881b2c90da024339897589477554f47f)
2. - [Raf] I hope you are enjoying the course so far,
3. and welcome to the second week
4. of Architecting Solutions on AWS.
5. You may know me as the customer
6. in the first week of the course.
7. During this week, I am going to play the role
8. of a solutions architect helping a customer
9. to achieve their business goals.
10. What I know so far, is that the customer
11. is a software company that develops a solution
12. for providing restaurant menus by using QR codes.
13. I've also heard that the customer has good coding expertise,
14. but they need some help architecting
15. their data analytics infrastructure to ingest,
16. process, and visualize data.
17. This feature will be in addition to their main product
18. and will make them more competitive in the market.
19. In this week, you can expect to learn more about
20. Amazon S3, data ingestion, processing, and visualization,
21. all matching customer requirements,
22. which I will collect in the next video.
23. Ah, and remember to check the readings between the videos.
24. There is good detailed content in there.
25. That way, you can dive deeper into the content
26. that is more relevant to your own learning.
27. Ready? Join us for the second week,
28. where I will teach you how to map
29. AWS data analytics services towards customer needs.
30. Let's continue because we have a customer
31. waiting for our call.
32. [End of transcript. Skip to the start.](https://courses.edx.org/xblock/block-v1:AWS+AWS-ARCH-1+3T2022+type@vertical+block@c499c667f3314bee95b9f62768f95c4b?exam_access=&recheck_access=1&show_bookmark=0&show_title=0&view=student_view#transcript-start-881b2c90da024339897589477554f47f)
33. [Start of transcript. Skip to the end.](https://courses.edx.org/xblock/block-v1:AWS+AWS-ARCH-1+3T2022+type@vertical+block@c0b2af5f3eed498fbaed74c0546b5702?exam_access=&recheck_access=1&show_bookmark=0&show_title=0&view=student_view#transcript-end-0691bbb1818f4c02a4f1c3aaea9999ba)
34. - [Raf] All right, let's call our customer for the week
35. and learn more about
36. what their current architecture state is,
37. and what they expect to be done.
38. During this call, I will also try to outline
39. what is in scope and out of scope for the architecture.
40. (phone ringing)
41. - [Morgan] Hi, Raf, great to hear from you.
42. - [Raf] Hello, Morgan, how are you doing?
43. How are things going?
44. - [Morgan] Things are pretty good.
45. How are you doing?
46. - [Raf] I'm doing pretty well.
47. Is this a good time to talk about the use case
48. and the requirements for the data analytics solution
49. we talked about the other day at the cafeteria?
50. - [Morgan] Sure, let me tell you the current state
51. of what we already have,
52. and then we can talk about what we need.
53. Does that work?
54. - [Raf] Yeah, sure, that's perfect.
55. I'll be picturing the architecture in my mind
56. as you speak, so I'll be mostly listening,
57. but I'm paying attention.
58. - [Morgan] Cool, so currently,
59. we offer a solution that uses an S3 static website
60. to host an HTML webpage containing restaurant menus.
61. We have also developed a solution,
62. where restaurant admins
63. can log in to a system of ours
64. and update that object in the bucket.
65. They do that when they want to edit the menu.
66. We also have a system to generate a QR code pointing
67. to the bucket, so restaurants use that to print the QR code
68. and add it to their restaurant tables.
69. All of that is already done and we maintain it.
70. - [Raf] Cool.
71. That's pretty cool and useful during times
72. when we don't want to touch menus with our hands.
73. It is also a solution that helps save paper.
74. I love it.
75. To see if I understood it correctly,
76. when customers are in the restaurant,
77. they scan the QR, then they choose what they want,
78. but they still have to wait for someone
79. to come to their table.
80. Is that right?
81. - [Morgan] Yeah, yeah, that's right.
82. And it's not optimal for restaurants and customers
83. because it adds time
84. between when the customer makes their choice
85. and then actually ordering.
86. We also have a third-party payment-processing service
87. that we use for other solutions,
88. and we would like to add an order this item option
89. to the static HTML websites,
90. that calls an API that we already have.
91. - [Raf] Great, yeah, that all makes sense.
92. That would be a nice upgrade to your product.
93. Tell me more.
94. - [Morgan] Yeah, so we know
95. that we need to do one QR code per table
96. because that will be the order identifier, and that's fine.
97. But since we are doing this product upgrade,
98. we would also like to have data analytics
99. that will help restaurants gain insights
100. into their menus.
101. So for example, we want to be able to see
102. what dishes in the menu people are viewing.
103. Insights can show us
104. if people are just scrolling through the whole menu,
105. or if they stop after appetizers,
106. and they don't get to entrees, things like that.
107. We also want to help our customers
108. optimize their menu design through analytics.
109. - [Raf] Yeah.
110. With that info, you can also discover things
111. like items where customers see the description of it
112. but don't add to their tab, or the amount of time
113. between a customer expands an item's detail
114. and they pay for it.
115. With this kind of information,
116. your BI analysts can make some reports
117. to help restaurants, and even suggest specific dishes
118. based on session data,
119. like if you order this from restaurant A,
120. then you might be interested on ordering this
121. from restaurant B.
122. That's called session data.
123. This whole process is called clickstream data analysis.
124. I can help you design a durable
125. and resilient architecture
126. that will receive, store, and process this data.
127. This is perfectly possible using
128. AWS data analytics services.
129. Let me just ask you another question.
130. Do you already have the client-side part
131. that will send that clickstream data?
132. How are you going to send data to be ingested?
133. - [Morgan] So, we wrote a JavaScript library
134. that we can use for that.
135. All we need is an endpoint where we can do HTTP POSTs,
136. and then the analytics side.
137. So, we'd prefer a solution
138. that accepts data through REST standards.
139. - [Raf] All right, so the coding on the client side
140. and the payment-gateway connection
141. are out of scope for this project, right?
142. - [Morgan] Right, we are happy
143. if we have a RESTful HTTPS endpoint
144. where we can send stuff.
145. - [Raf] Perfect.
146. Now, in addition, do you have any other requirements?
147. - [Morgan] Yes, we do.
148. Cost is a priority for us,
149. so we would prefer to use AWS services
150. that bill per refined usage, and not per time.
151. We also would prefer using managed services, when possible,
152. because we are operating at a reduced staff,
153. so that means, like, no EC2 instances
154. or operating systems to maintain, if possible.
155. We'd also prefer to have the data ingested
156. and available as a backup
157. in another AWS Region with encryption.
158. - [Raf] Absolutely, that seems the right thing to do.
159. I assume you would like encryption both at rest
160. and in transit, right?
161. There is even a quote from Werner Vogels,
162. Amazon's CTO, that says:
163. "Dance like nobody's watching,
164. encrypt like everyone is."
165. (Raf laughs)
166. Thanks for coming to me
167. with well-defined requirements.
168. That's really rare.
169. (Raf laughs)
170. So, we will likely use Amazon S3 for storage,
171. which checks many of these boxes.
172. I will call you with more questions
173. as I progress through the architecture of this solution.
174. Is that okay?
175. - [Morgan] Yeah, yeah, please do call me.
176. Call me any time you have any additional questions.
177. - [Raf] All right.
178. Thank you, bye bye, see you soon.
179. - [Morgan] Bye.
180. [End of transcript. Skip to the start.](https://courses.edx.org/xblock/block-v1:AWS+AWS-ARCH-1+3T2022+type@vertical+block@c0b2af5f3eed498fbaed74c0546b5702?exam_access=&recheck_access=1&show_bookmark=0&show_title=0&view=student_view#transcript-start-0691bbb1818f4c02a4f1c3aaea9999ba)
181. [Start of transcript. Skip to the end.](https://courses.edx.org/xblock/block-v1:AWS+AWS-ARCH-1+3T2022+type@vertical+block@eff20b74d8cd4585aafe32847e7127ad?exam_access=&recheck_access=1&show_bookmark=0&show_title=0&view=student_view#transcript-end-9b034241adcc4e5aa6f25d7e8ce09d69)
182. - [Raf] It is clear that we are going to use
183. AWS data analytics services.
184. The first thing we need to think about
185. when talking about data analytics
186. is how to use the collected data to produce information
187. that would be useful for business needs.
188. Those are called insights.
189. Sometimes, this journey of generating insights
190. out of data can be complex,
191. involving the use of machine learning,
192. or sometimes it can be quick and simple.
193. I usually say that if you control how the data is produced,
194. things are easier because you don't need to apply
195. data transformation when it arrives.
196. Regardless of whether you are going simple or complex,
197. AWS offer data services to transform data
198. while it's being ingested, or after it has arrived.
199. In our scenario, the customer will be using
200. a JavaScript library to produce data,
201. and they are controlling how that data is sent.
202. That makes things simpler
203. because we would just need a service to ingest
204. and store data as-is.
205. That's an important consideration.
206. The number one tip is to always think
207. about how data is going to come
208. into the architecture you were designing.
209. This becomes more important
210. if you were building a generic solution
211. that is capable of accepting data from different types,
212. sources, and devices, such as images, audio,
213. log files, pictures, and et cetera.
214. You may also have heard the saying,
215. "Use the right tool for the job."
216. The reason for that is because
217. there are so many data services out there,
218. in a way that choosing the most appropriate one
219. will usually perform better, while saving some money.
220. Which brings us to the question: which one will I choose?
221. So, the best way to choose a service,
222. is by understanding what each service does.
223. That way, you can map a customer requirement
224. to a specific service that will do a specific task.
225. Let me point my web browser to
226. aws.amazon.com/big-data/datalakes-and-analytics.
227. If I scroll down on this page,
228. I have the AWS analytics services.
229. And AWS puts its data services in four categories.
230. Analytics,
231. data movement,
232. data lake,
233. and predictive analytics and machine learning.
234. If you need interactive query, you can use Amazon Athena,
235. which is an interactive query service
236. that makes it easier to analyze data in Amazon S3
237. using standard SQL.
238. Athena is serverless,
239. so there is no infrastructure to manage,
240. and you pay only for the queries that you run.
241. Definitely a service that charges for a refined usage.
242. If you need big data processing,
243. you can use Amazon EMR or AWS Glue,
244. which are services that can run
245. managed Hadoop clusters or jobs.
246. These are useful for further data transformation.
247. They are cloud big data platforms
248. for doing massive parallel processing of distributed data.
249. It uses open-source analytics frameworks
250. such as Apache Spark, Apache Hive, Presto, and others.
251. Remember when I talked about advanced data transformation?
252. Amazon EMR would help you with that.
253. There are so many services, and the goal of this video
254. is not to double-click on each one.
255. The reading you are going to see next will do that for you.
256. The main message here, is to choose
257. the right tool for the job.
258. And the job, in this case,
259. are our customer requirements and needs.
260. Sometimes, these requirements and needs are satisfied
261. by more than one specific service.
262. When that happens, it's time to call our customer
263. asking for more detailed information.
264. To give you an idea, here are some questions
265. that would help you decide which service to use.
266. What latency is acceptable between ingesting the data
267. and making it available?
268. Can the customer afford to wait some time to get the data?
269. Where is the data coming from?
270. Are we talking about a database migration,
271. clickstream data ingestion, or batch file uploads?
272. Sometimes, customers wants to use solutions
273. they created, or software
274. they already have licenses and support for.
275. Sometimes, customers may want to use
276. their own credential federation,
277. and integration with existing identity providers.
278. That's fine.
279. If that's the case, you may need to configure AWS services
280. in order to make a bridge between the existing resources
281. and the new architecture you are creating.
282. Over the next videos, I will be discussing some
283. of the data analytics services for clickstream solutions,
284. explaining why you might choose one service over another.
285. For the sake of matching an AWS service
286. with the customer need,
287. I may need to call Morgan while architecting the solution.
288. [End of transcript. Skip to the start.](https://courses.edx.org/xblock/block-v1:AWS+AWS-ARCH-1+3T2022+type@vertical+block@eff20b74d8cd4585aafe32847e7127ad?exam_access=&recheck_access=1&show_bookmark=0&show_title=0&view=student_view#transcript-start-9b034241adcc4e5aa6f25d7e8ce09d69)

In this course, the Morgan and Raf cover some of the AWS data services that can be used for a solution that ingests clickstream data. Clickstream is the term for small events that contain pieces of data that are generated continuously with high speed and volume. Remember, clickstream data is usually data that’s collected by systems (mostly frontends) regarding user interactions with that system. In this week’s scenario, clickstream data will be used to collect information about users’ behaviors when they interact with a restaurant menu. Regardless of whether you are architecting for clickstream data, AWS provides various analytics services that meet your needs for data analytics. Organizations of all sizes and industries can use these services to reinvent their business with data.

## AWS data services

From data movement, data storage, data lakes, big data analytics, log analytics, streaming analytics, business intelligence, and machine learning (ML) to many things in between, AWS offers purpose-built services that provide price performance, scalability, and low costs. [Data Lakes and Analytics on AWS](https://aws.amazon.com/big-data/datalakes-and-analytics/) lists all the AWS services that can be used for data analytics, and places the services into four distinct categories. This reading summarizes the most popular services in each category.

## Data lakes and data storage

#### Amazon S3

Amazon Simple Storage Service (Amazon S3) is an object storage service that offers scalability, data availability, security, and performance. Customers of all sizes and industries can store and protect virtually any amount of data for virtually any use case, such as data lakes, cloud-native applications, and mobile apps. With cost-effective storage classes and easy-to-use management features, you can optimize costs, organize data, and configure fine-tuned access controls to meet specific business, organizational, and compliance requirements.  
The following list details some use cases for Amazon S3:

* Archive data at the lowest cost: Move data archives to the Amazon Simple Storage Service Glacier (Amazon S3 Glacier storage classes to lower costs, reduce operational complexities, and gain new insights.
* Run cloud-native applications: Build fast, powerful, mobile and web-based cloud-native applications that scale automatically in a highly available configuration, such as static websites that use the client side for coding.
* Build a data lake: Run big data analytics, artificial intelligence (AI), machine learning (ML), and high performance computing (HPC) applications to unlock data insights.
* Back up and restore critical data: Meet Recovery Time Objectives (RTO), Recovery Point Objectives (RPO), and compliance requirements with the robust replication features of Amazon S3.

For more information, see [Amazon S3](https://aws.amazon.com/s3/).

#### Amazon S3 Glacier

**Note:** Amazon S3 storage classes will have their own dedicated reading later in this week.  
  
The Amazon S3 Glacier storage classes are purpose-built for data archiving. They are designed to provide you with high performance, retrieval flexibility, and low-cost archive storage in the cloud. All S3 Glacier storage classes provide virtually unlimited scalability and are designed for 99.999999999 percent (11 nines) of data durability. In addition to low-cost storage, the S3 Glacier storage classes also deliver options for fast access to your archival data.  
  
For more information, see [Amazon S3 Glacier storage classes](https://aws.amazon.com/s3/storage-classes/glacier/).

#### AWS Lake Formation

AWS Lake Formation is a service that you can use to set up a secure data lake in days. A data lake is a centralized, curated, and secured repository that stores all your data, both in its original form and prepared for analysis. You can use a data lake to break down data silos and combine different types of analytics to gain insights and guide better business decisions.  
  
For more information, see [AWS Lake Formation](https://aws.amazon.com/lake-formation/?whats-new-cards.sort-by=item.additionalFields.postDateTime&whats-new-cards.sort-order=desc).

## Data analytics

#### Amazon Athena

Amazon Athena is an interactive query service that you can use to analyze data in Amazon S3 by using standard Structured Query Language (SQL). Athena is serverless, so you don’t need to manage infrastructure, and you pay only for the queries that you run.  
  
Using Athena is straightforward. You point to your data in Amazon S3, define the schema, and start querying by using standard SQL. Most results are delivered within seconds. With Athena, you don’t need complex extract, transform, and load (ETL) jobs to prepare your data for analysis. Anyone with SQL skills can use Athena to quickly analyze large-scale datasets.  
  
For more information, see [Amazon Athena](https://aws.amazon.com/athena/).

#### Amazon EMR

Amazon EMR is a big data solution for petabyte-scale data processing, interactive analytics, and machine learning that use open-source frameworks, such as Apache Spark, Apache Hive, and Presto.  
  
In this course, we didn’t use or explore Amazon EMR for this week’s scenario because the customer was short-staffed, and Amazon EMR requires a learning curve to operate the open-source frameworks that it uses. In fact, Amazon EMR has so many features that we could make an entire 4-week course about it! The following list details some of the use cases for Amazon EMR. (Notice that the third item aligns well with the week’s scenario—Amazon EMR would a good candidate for the task if the customer already had big data knowledge.)

* Run large-scale data processing and what-if analysis by using statistical algorithms and predictive models to uncover hidden patterns, correlations, market trends, and customer preferences.
* Extract data from various sources, process it at scale, and make the data available for applications and users.
* Analyze events from streaming data sources in real time to create long-running, highly available, and fault-tolerant streaming data pipelines.
* Analyze data using open-source ML frameworks, such as Apache Spark MLlib, TensorFlow, and Apache MXNet.
* Connect to Amazon SageMaker Studio for large-scale model training, analysis, and reporting.

For more information, see [Amazon EMR](https://aws.amazon.com/emr/?c=a&sec=srv).

#### Amazon OpenSearch Service

You can use Amazon OpenSearch Service to perform interactive log analytics, real-time application monitoring, website search, and more. OpenSearch is an open source, distributed search and analytics suite that is derived from Elasticsearch. Amazon OpenSearch Service is the successor to Amazon Elasticsearch Service. It offers the latest versions of OpenSearch, support for 19 versions of Elasticsearch, and visualization capabilities that are powered by OpenSearch Dashboards and Kibana. Amazon OpenSearch Service currently has tens of thousands of active customers, with hundreds of thousands of clusters under management, processing hundreds of trillions of requests per month.  
  
For more information, see [Amazon OpenSearch Service](https://aws.amazon.com/opensearch-service/?c=a&sec=srv).

## Data movement

#### Amazon Kinesis

**Note:** Later this week, Amazon Kinesis will have its own dedicated reading that further explains each of the Kinesis services (such as Amazon Kinesis Data Streams, Amazon Kinesis Data Firehose, and Amazon Kinesis Data Analytics).  
  
With Amazon Kinesis, you can collect, process, and analyze real-time, streaming data so that you can get timely insights and react quickly to new information. Amazon Kinesis offers key capabilities to cost-effectively process streaming data at virtually any scale, along with the flexibility to choose the tools that best suit the requirements of your application. With Amazon Kinesis, you can ingest real-time data such as video, audio, application logs, website clickstreams, and Internet of Things (IoT) telemetry data for machine learning, analytics, and other applications. You can use Amazon Kinesis to process and analyze data as it arrives, which means that you can respond quickly—you don’t need to wait for all your data to be collected before processing can begin.  
  
For more information, see [Amazon Kinesis](https://aws.amazon.com/kinesis/).

#### AWS Glue

AWS Glue is a serverless data integration service that you can use to discover, prepare, and combine data for analytics, machine learning, and application development. AWS Glue provides capabilities that are needed for data integration so that you can start analyzing your data and using your data in minutes instead of months. Data integration is the process of preparing and combining data for analytics, machine learning, and application development. It involves multiple tasks, such as discovering and extracting data from various sources; enriching, cleaning, normalizing, and combining data; and loading and organizing data in databases, data warehouses, and data lakes. These tasks are often handled by different types of users who each use different products.  
  
For more information, see [AWS Glue](https://aws.amazon.com/glue/?whats-new-cards.sort-by=item.additionalFields.postDateTime&whats-new-cards.sort-order=desc).

#### AWS DMS

AWS Database Migration Service (AWS DMS) helps you migrate databases to AWS quickly and securely. The source database remains fully operational during the migration, which minimizes downtime to applications that rely on the database. AWS DMS can migrate your data to and from the most widely used commercial and open-source databases.  
  
For more information, see [AWS Database Migration Service](https://aws.amazon.com/dms/).

## Predictive analytics and machine learning

You won’t explore this area much in this course, but—as was mentioned earlier in this reading—regardless of the kind of architecture you want to create, AWS offers services that can help. (That is, use the right tool for the job!) Though you might not see the following AWS services typically listed with AWS data services, you can use them to handle or process data for predictive analytics and machine learning (ML).

#### Amazon SageMaker

SageMaker can be used for any generic ML solution. You can use it to build, train, and deploy ML models for virtually any use case with fully managed infrastructure, tools, and workflows. SageMaker requires a learning curve to use, but it’s a managed serverless service that many people can use to innovate with ML through a choice of tools—such as integrated development environments (IDEs) for data scientists and no-code interfaces for business analysts.  
  
For more information, see [Amazon SageMaker](https://aws.amazon.com/sagemaker/).

#### Amazon Rekognition

Amazon Rekognition is one of Raf’s favorite ML services from the entire list of AWS services! It is easy to use, serverless, and abstracted, in the sense that you interact with it by doing API calls. With Amazon Rekognition, you can automate image and video analysis by adding pretrained or customizable computer vision API operations to your applications without building ML models and infrastructure from scratch.  
  
On its own, Amazon Rekognition is not a data analytics service. However, it’s listed here because you can use it as part of a data analytics solution. It is designed to analyze millions of images, streaming, and stored videos within 3 seconds. Check out some examples:  
In this image, Amazon Rekognition is detecting objects (such as packages, pets, or people) in real time from live video streams.  
In this image, Amazon Rekognition is detecting faces that appear in images and videos, and recognizing attributes (such as open eyes, glasses, and facial hair) for each face.  
For more information, see [Amazon Rekognition](https://aws.amazon.com/rekognition/).

#### Amazon Comprehend

Amazon Comprehend is a natural-language processing (NLP) service that uses ML to uncover valuable insights and connections in text, which is instrumental for a data analytics solution. For example, you could mine business and call center analytics or process financial documents. For medical use cases, you can use Amazon Comprehend Medical, which focuses on extracting information accurately and quickly from unstructured medical text.  
  
For more information, see [Amazon Comprehend](https://aws.amazon.com/comprehend/) and [Amazon Comprehend Medical](https://aws.amazon.com/comprehend/medical/).

1. [Start of transcript. Skip to the end.](https://courses.edx.org/xblock/block-v1:AWS+AWS-ARCH-1+3T2022+type@vertical+block@bc482f9e6d86404590461547c40e8361?exam_access=&recheck_access=1&show_bookmark=0&show_title=0&view=student_view#transcript-end-bf3d22b676bd42f19b2ce83e1fa345c2)
2. - [Raf] The analytics solution
3. will need a place to store the clickstream data.
4. In this video,
5. I will talk about storage by comparing some AWS services,
6. and tell you why Amazon S3 will be
7. the chosen one. (voice echoes)
8. There are many storage options to consider,
9. such as Amazon EFS, Amazon EBS, and Amazon S3.
10. And some other services,
11. more suitable for data migration, file transfer, or backups.
12. Let me briefly talk about these three,
13. and consider if they should be part of our architecture,
14. or not.
15. So let me bring to our table, Amazon EBS,
16. Amazon EFS,
17. and Amazon S3.
18. Amazon EBS, or Amazon Elastic Block Store,
19. is a service that provides block-level access
20. when you need a file system.
21. It provides high-end performance
22. and can handle workloads at virtually any scale.
23. But the reason why I'm not going to use EBS volumes
24. in our architecture,
25. is because it does not align well with the requirements
26. and needs.
27. Don't get me wrong,
28. I love EBS volumes.
29. And let me explain why I am not using it.
30. The reason why I am not using, in this case,
31. is because they need an EC2 instance to be attached.
32. And their billing is not as granular as Amazon S3.
33. Meaning that, if you allocate a 50-gigabyte volume,
34. you'll have to pay for 50 gigabytes,
35. regardless if you only have a couple of gigabytes to store.
36. Also, the durability standards of Amazon S3
37. are higher than Amazon EBS,
38. since S3 replicates data across an entire AWS Region.
39. For these reasons,
40. I am not considering Amazon EBS for this storage layer.
41. Okay.
42. What about the next one on the list?
43. Amazon EFS, or Amazon Elastic File System.
44. Amazon Elastic File System
45. can replicate data across an AWS Region.
46. So, it is a service that has a higher data durability
47. when compared to Amazon EBS.
48. But it also relies on file systems
49. that needs to be attached to resources,
50. such as EC2 instances, containers, or servers.
51. So although it checks the durability box,
52. it doesn't address a very common ask
53. for most data analytics architectures,
54. which is having your storage layer
55. that lives by itself,
56. and does not need to be attached to anything
57. in order to store data.
58. Because of that,
59. I am not using Amazon EFS for the storage layer.
60. That's where Amazon S3 shines.
61. And that's why we usually say,
62. it's the storage for the internet.
63. You can interact with Amazon S3 by doing API calls,
64. such as PutObject, GetObject, DeleteObject, or others.
65. And you can do these API calls from your laptop,
66. from a webpage, from a server, from a mobile app,
67. and virtually everywhere.
68. Because of that nature,
69. S3 is ideal for separating storage from data processing.
70. And that is an important thing to have in data analytics
71. or big data operations.
72. Let me explain why.
73. Having a layer that stores only data is convenient
74. because you can use different data-processing services
75. or ingestion mechanisms.
76. Remember the saying, "Use the right tool for the job"?
77. So, detaching storage from processing
78. gives me the flexibility
79. of using specific data-processing services
80. that focus on the right thing.
81. That's why Amazon S3 is also commonly used as a data lake.
82. This term comes from nature,
83. where people usually do an analogy with different biomes
84. and data in nature.
85. That way, data is accessible for specific tools to process
86. and analyze data.
87. Most of AWS, and many commercial solutions,
88. already know how to work with S3 as the data layer.
89. Now, if we go back to our requirements,
90. you will see that Amazon S3 checks all the boxes.
91. One of the requirements is
92. that data should be stored in a durable way
93. with cross-Region replication and encryption.
94. Well, Amazon S3 is known
95. for providing high data durability standards.
96. It is designed to provide 11 nines, or
97. 99.999999999 (popping sounds)
98. percent (bell rings)
99. durability for objects,
100. over in a given year.
101. **S3 gets that high durability**
102. **because every single object in a bucket gets replicated**
103. **over different physical facilities in an AWS Region.**
104. **I know that cost savings is also another requirement,**
105. **and S3 helps by only charging for refined usage,**
106. **meaning that it charges for 5 megabytes of data,**
107. **if you just store 5 megabytes.**
108. **Or 1 petabyte of data, if you store 1 petabyte.**
109. **Also, speaking about large data amounts,**
110. **Amazon S3 provides a feature called S3 Intelligent-Tiering,**
111. **which transitions objects from one storage class**
112. **to another automatically, based on access patterns.**
113. **This does automate cost savings,**
114. **which is always something my customer will welcome.**
115. **In a nutshell,**
116. **Amazon S3 charges per usage,**
117. **refined usage.**
118. **It easily enables cross-Region replication.**
119. **It encrypts data, both in transit and at rest,**
120. **at no additional cost.**
121. **And let me use multiple data ingestion**
122. **and processing systems,**
123. **by being a managed service that does not need instances**
124. **or servers to operate.**
125. **It is perfect for our use case, and it checks all the boxes.**
126. **That's why I am using it for the storage layer**
127. **of my data analytics solution.**
128. [End of transcript. Skip to the start.](https://courses.edx.org/xblock/block-v1:AWS+AWS-ARCH-1+3T2022+type@vertical+block@bc482f9e6d86404590461547c40e8361?exam_access=&recheck_access=1&show_bookmark=0&show_title=0&view=student_view#transcript-start-bf3d22b676bd42f19b2ce83e1fa345c2)

<https://docs.aws.amazon.com/AmazonS3/latest/userguide/access-policy-language-overview.html?icmpid=docs_amazons3_console>

{

"Id": "Policy1718166020898",

"Version": "2012-10-17",

"Statement": [

{

"Sid": "Stmt1718166015671",

"Action": [

"s3:AbortMultipartUpload",

"s3:AssociateAccessGrantsIdentityCenter",

"s3:BypassGovernanceRetention",

"s3:CreateAccessGrant",

"s3:CreateAccessGrantsInstance",

"s3:CreateAccessGrantsLocation",

"s3:CreateAccessPoint",

"s3:CreateAccessPointForObjectLambda"

],

"Effect": "Allow",

"Resource": "arn:aws:s3:::djsresturant",

"Principal": {

"AWS": [

"djspolicy1"

]

}

}

]

}

<https://awspolicygen.s3.amazonaws.com/policygen.html>

**{**

**"Version":"2012-10-17",**

**"Statement":[**

**{**

**"Sid":"PolicyForAllowUploadWithACL",**

**"Effect":"Allow",**

**"Principal":{"AWS":"111122223333"},**

**"Action":"s3:PutObject",**

**"Resource":"arn:aws:s3:::myresturantbucket/\*",**

**"Condition": {**

**"StringEquals": {"s3:x-amz-acl":"bucket-owner-full-control"}**

**}**

**}**

**]**

**}**

**==**

**{**

**"Version": "2012-10-17",**

**"Statement": [**

**{**

**"Sid": "PolicyForAllowUploadWithACL",**

**"Effect": "Allow",**

**"Principal": {**

**"AWS": "111122223333"**

**},**

**"Action": [**

**"s3:PutObject",**

**"s3:\*"**

**],**

**"Resource": "arn:aws:s3:::myresturantbucket/\*",**

**"Condition": {**

**"StringEquals": {**

**"s3:x-amz-acl": "bucket-owner-full-control"**

**}**

**}**

**}**

**]**

**}https://docs.aws.amazon.com/AmazonS3/latest/userguide/example-bucket-policies.html?icmpid=docs\_amazons3\_console**

**arn:aws:s3:::myresturantbucket**

etter solution:

1. Create an IAM policy that gives access to the bucket
2. Assign it to a group
3. Put user into that group

Instead of saying "This bucket is allowed to be touched by this user", you can define "These are the people that can touch this".

It sounds silly right now, but wait till you add 42 more buckets and 60 users to the mix. Having a central spot to manage all resource access will save the day.

The value for Principal should be user arn which you can find in Summary section by clicking on your username in IAM. It is because so that specific user can bind with the S3 Bucket Policy In my case, it is arn:aws:iam::332490955950:user/sample ==> sample is the username

1. [Start of transcript. Skip to the end.](https://courses.edx.org/xblock/block-v1:AWS+AWS-ARCH-1+3T2022+type@vertical+block@bb429acda4d946e8a2ec6d079e0ad5aa?exam_access=&recheck_access=1&show_bookmark=0&show_title=0&view=student_view#transcript-end-6b05a2f913fb4b189ce948b6eb0c3bce)
2. - [Raf] All right. Let me load my Amazon S3 web console to
3. show you some exploration we can do in this service.
4. When you open Amazon S3 webpage,
5. which I already have here from the AWS Management Console,
6. you are presented with an option of creating a new bucket.
7. There is a button on the upper-right corner
8. that I can choose, Create bucket.
9. When I click in that option, I can specify the bucket name.
10. So, let me put my bucket name as
11. raf-restaurant-1.
12. And I can choose a Region
13. where I want the bucket to be created.
14. In my case, it is us-east-1.
15. So, buckets are Regional.
16. You don't need to specify an Availability Zone
17. for a bucket, that doesn't exist.
18. You just specify the Region
19. you want the bucket to be created.
20. Scrolling down,
21. you have some options, such as
22. blocking public access settings for this bucket.
23. The important word here is settings.
24. By unchecking the box that says Block all public access,
25. you are not making the bucket public.
26. You are allowing a bucket configuration
27. to make this bucket public.
28. Okay. So I am disabling that, for now,
29. which doesn't mean that the bucket is public.
30. I'm just allowing the bucket to be configured
31. in a public manner.
32. So scrolling down, I acknowledge
33. that the current settings might result
34. in this bucket and the objects within becoming public.
35. I kind-of wants to mimic what our customer have, here.
36. They have a bucket that is publicly accessible,
37. and they have HTML files with the restaurant menus.
38. So, I'm calling my bucket named raf-restaurant-1
39. in Virginia, and this bucket is not blocking
40. all the public access settings.
41. If I scroll down,
42. there is an option that says, Create bucket.
43. So I'm just going to choose that option, Create bucket,
44. and now, the bucket is created.
45. What I am going to do, is just open the bucket
46. and upload a file to a bucket.
47. I can do that by choosing the option that says Upload.
48. I'm going to hit Upload, drag and drop a file
49. that I already have here,
50. index.html.
51. So, I am uploading a file to this bucket,
52. called index.html.
53. As from the moment I click Upload, this file
54. will become a replicated object in my bucket.
55. Now, I have a bucket with a file that has 32 bytes.
56. It's a very small file in my bucket, raf-restaurant-1.
57. Now, how to make this file, this object public?
58. The first thing I should do, is actually go to Amazon S3,
59. and there is a setting in my account
60. that automatically blocks public access settings
61. for the account.
62. By default, this is on.
63. I went here, I clicked Edit, and I turned it off.
64. So, by default, it looked like this.
65. You, you must type confirm.
66. By default,
67. it looks like everything is blocked,
68. and this is in the account level.
69. Even though we had overwritten that
70. for that specific bucket, you must disable this
71. in the account, right?
72. So, you must go and hit Edit and block all public access.
73. You want to disable that.
74. Save changes, type confirm,
75. and now, your bucket will be effectively able to
76. be public when you configure a bucket policy to do so.
77. So, we have our bucket,
78. which is called raf-restaurant-1 with an object inside.
79. And first, before making the object, the bucket, public.
80. I would like to go on Properties
81. and scroll down everything,
82. and enable a feature called static website hosting,
83. which is what our customers have.
84. So, I'm going to enable static website hosting
85. and I can specify an index document,
86. which will be served
87. when, whenever someone accesses the URL for that bucket.
88. I'll just choose index.html, save changes,
89. and now I can go in Permissions,
90. Bucket policy, and edit the bucket policy.
91. I am going to place a bucket policy that allow
92. this bucket to be public, which is read like this.
93. This is a policy that has a statement ID, or SID,
94. that says PublicRead,
95. allowing everyone to make the operations of GetObject
96. and GetObjectVersion everywhere in my bucket.
97. That's how you read this policy.
98. I am going to scroll down and hit Save changes.
99. Now that I have a public bucket
100. that says publicly accessible,
101. and I have static website hosting enabled,
102. I can go back to Properties
103. and scroll down just to get the static website hosting URL.
104. So, each bucket has a URL
105. when you have static website hosting enabled,
106. and that's one of the reasons
107. for bucket names needing to be globally unique,
108. because you cannot have two URLs
109. in the internet with the same name
110. pointing to two different locations, right?
111. So, this URL is pointing to the bucket.
112. If I open a new browser tab,
113. or if I simply click in that URL,
114. I have my index.html served automatically, right?
115. And I don't need to type /index.html
116. because the static website hosting feature
117. is doing that for me.
118. If I type index.html, it's the same thing, right?
119. And this could be a restaurant menu.
120. This is an HTML file that I have, with just that
121. "This could be a restaurant menu!" string in it.
122. But it could be a full CSS style sheets,
123. with JavaScript libraries running things
124. on my client's side, such as ingesting clickstream data
125. to an HTTPS endpoint provided by API Gateway, right?
126. So more, more to come in the course, stay with us,
127. and this was the exploring Amazon S3 video.
128. [End of transcript. Skip to the start.](https://courses.edx.org/xblock/block-v1:AWS+AWS-ARCH-1+3T2022+type@vertical+block@bb429acda4d946e8a2ec6d079e0ad5aa?exam_access=&recheck_access=1&show_bookmark=0&show_title=0&view=student_view#transcript-start-6b05a2f913fb4b189ce948b6eb0c3bce)

## S3 CRR

With S3 Cross-Region Replication (CRR), you can replicate objects—and their respective metadata and object tags—into other AWS Regions for reduced latency, compliance, security, disaster recovery, and other use cases. CRR can be configured from a single, source Amazon Simple Storage Service (Amazon S3) bucket to replicate objects into one or more destination buckets in another AWS Region.  
  
CRR automatically replicates data between buckets across different AWS Regions. With CRR, you can set up replication at a bucket level, a shared prefix level, or an object level (by using Amazon S3 object tags).   
  
You can use CRR to provide lower-latency data access in different geographic regions. CRR can also help if you have a compliance requirement to store copies of data hundreds of miles apart. You can use CRR to change account ownership for the replicated objects to protect data from accidental deletion. For more information about CRR, see [Replicating objects](https://docs.aws.amazon.com/AmazonS3/latest/userguide/replication.html) in the *Amazon S3 User Guide*.  
  
Example use cases include the following:

* Compliance: Amazon S3 stores your data across multiple geographically distant Availability Zones by default. However, compliance requirements might require you to store data at even greater distances. You can use CRR to replicate data between distant AWS Regions to satisfy these requirements.
* Latency performance: If your customers or end users are distributed across one or more geographic locations, you can minimize latency for data access by maintaining multiple object copies in AWS Regions that are geographically closer to your customers.
* Regional efficiency: If you have compute clusters in two or more AWS Regions that analyze the same set of objects, you might choose to maintain object copies in all of those AWS Regions.

## S3 Lifecycle

You can add rules in an S3 Lifecycle configuration to tell Amazon S3 to transition objects to another Amazon S3 storage class. For more information about storage classes, see. Some examples of when you might use S3 Lifecycle configurations in this way include the following:

* When you know that objects are infrequently accessed, you might transition them to the S3 Standard-IA storage class.
* You might want to archive objects that you don't need to access in real time to the S3 Glacier Flexible Retrieval storage class.

However, what if you’re not sure about the data access patterns?

### Amazon S3 Intelligent-Tiering storage class

S3 Intelligent-Tiering is a good storage class for data with unknown, changing, or unpredictable access patterns—independent of object size or retention period. You can use S3 Intelligent-Tiering as the default storage class for virtually any workload, especially data lakes, data analytics, new applications, and user-generated content.  
  
S3 Intelligent-Tiering is the only cloud storage class that’s designed to deliver automatic storage cost savings when data access patterns change, without performance impact or operational overhead. S3 Intelligent-Tiering is designed to optimize storage costs by automatically moving data to the most cost-effective access tier when access patterns change. For a small monthly object monitoring and automation charge, S3 Intelligent-Tiering monitors access patterns and automatically moves objects that haven’t been accessed to lower-cost access tiers.  
  
There are no retrieval charges in S3 Intelligent-Tiering. S3 Intelligent-Tiering has no minimum eligible object size, but objects smaller than 128 KB are not eligible for automatic tiering. These smaller objects can be stored, but they will always be charged at the Frequent Access tier rates and don’t incur the monitoring and automation charge.  
For more information, see [Amazon S3 Intelligent-Tiering storage class](https://aws.amazon.com/s3/storage-classes/intelligent-tiering/).

### Amazon S3 Glacier storage classes

You can use Amazon S3 Lifecycle policies to transition objects from one storage class to another. These storage classes include Amazon Simple Storage Service Glacier (Amazon S3 Glacier) storage classes, which are purpose-built for data archiving. S3 Glacier storage classes provide you with low-cost archive storage in the cloud. All S3 Glacier storage classes provide virtually unlimited scalability and are designed for 99.999999999 percent (11 nines) of data durability. The S3 Glacier storage classes deliver options for fast access to your archive data and low-cost archive storage in the cloud.  
For more information, see [Amazon S3 Glacier storage classes](https://aws.amazon.com/s3/storage-classes/glacier/).

**{**

**"Version": "2012-10-17",**

**"Statement": [**

**{**

**"Sid": "PublicRead",**

**"Effect": "Allow",**

**"Principal": "\*",**

**"Action": [**

**"s3:GetObject",**

**"s3:GetObjectVersion"**

**],**

**"Resource": "arn:aws:s3:::djsresturant/\*"**

**}**

**]**

**}**

**------------------------**

**{**

**"Version":"2012-10-17",**

**"Statement":[**

**{**

**"Sid":"PublicRead",**

**"Effect":"Allow",**

**"Principal": {**

**"AWS": "${aws\_cloudfront\_origin\_access\_identity.my\_aws\_cloudfront\_origin\_access\_identity.iam\_arn}"**

**},**

**"Action": [**

**"s3:GetObject",**

**"s3:GetObjectVersion"**

**],**

**"Resource":[**

**"arn:aws:s3:::myrestaurantbucket/\*"**

**]**

**}**

**]**

**}**

## Steps to Reproduce (for bugs)

1. ./mc mb minio/mybucket
2. create file policy.json:

{

"Version": "2012-10-17",

"Statement": [

{

"Sid": "AllowThisBucketOnly",

"Effect": "Allow",

"Action": "s3:\*",

"Resource": [

"arn:aws:s3:::mybucket/\*",

"arn:aws:s3:::mybucket"

]

}

]

}

1. ./mc policy set-json policy.json minio/mybucket  
   mc: Unable to set-json policy policy.json for minio/mybucket. Policy has invalid resource.
2. I was facing the same problem. I was not using the correct resource name. I did change the resource name to exactly the bucket for which I was creating the bucket policy e.g
3. "Resource": "arn:aws:s3:::abc/\*"
4. to
5. "Resource": "arn:aws:s3:::abc12/\*"
6. [Start of transcript. Skip to the end.](https://courses.edx.org/xblock/block-v1:AWS+AWS-ARCH-1+3T2022+type@vertical+block@3e6eb5af948a4683916e01bd30776184?exam_access=&recheck_access=1&show_bookmark=0&show_title=0&view=student_view#transcript-end-f0ba99708b08466a87dbf7154bd250de)
7. - [Raf] Now that we know we are going to use Amazon S3
8. for the storage layer, it is time to talk
9. about which service we are going to use to bring
10. the data to S3 in a scalable way.
11. As usual, there are a variety of services
12. to choose from.
13. While some are ideal for bringing data in batches,
14. some others are most suitable for IoT, database migrations,
15. or even offline data transfer in a truck.
16. In this video, I will compare some of these services
17. and justify my choice for the service that I choose
18. to handle clickstream data ingestion.
19. Before we start talking about services,
20. allow me to make some comments about clickstream data.
21. Clickstream is the term for when we have
22. small events containing pieces of data
23. that are generated continuously, with high speed and volume.
24. Usually, clickstream events are generated by user actions,
25. such as in this case, navigating through a restaurant menu
26. using HTML code.
27. Most clickstream data is used to analyze user behavior
28. in a website by logging the user's navigation,
29. such as the amount of time the user spends in the webpage,
30. where they usually begin the navigation,
31. how it ends, and et cetera.
32. By tracking these user behavior in real time,
33. you can update recommendations,
34. perform advanced A/B testing,
35. send push notifications based on session length,
36. and much more.
37. This is the exact need of our customer.
38. As usual, there are many ways to achieve the same results.
39. I will bring the services to the table,
40. and I will remove them one by one
41. until we find the most appropriate one for our scenario.
42. Let me bring the candidates, which are AWS Data Exchange,
43. AWS DMS,
44. Amazon EMR,
45. Kinesis Data Firehose,
46. Kinesis Data Streams,
47. and Kinesis Data Analytics.
48. Let me start with Amazon EMR.
49. A way of doing clickstream ingestion
50. is by using Amazon EMR with a managed Hadoop cluster
51. and installing a streaming framework,
52. such as Spark Streaming.
53. This could do the job,
54. but they run in an Amazon EMR cluster,
55. which although it's a managed service,
56. charges per time and requires
57. some big data knowledge to operate.
58. Since our customer is running operations with reduced staff,
59. they may not have someone with expertise
60. in big data frameworks, and having a service
61. that charges per time instead of per granular usage
62. would not meet one of the requirements.
63. That being said, I would not consider Amazon EMR
64. for the data ingestion in this case.
65. Out of the equation for now.
66. Another data ingestion service is AWS DMS,
67. or AWS Database Migration Service.
68. It is a service suitable
69. for when you want to bring a database to the cloud.
70. It does that by helping you migrate databases,
71. while keeping the source database running
72. during the migration.
73. In our scenario, there are no databases to be migrated,
74. just clickstream data that needs to be sent.
75. So, for that reason,
76. AWS DMS is not a player in the game here.
77. Another data ingestion service would be AWS Data Exchange,
78. but AWS Data Exchange is not designed
79. for our customer's purpose.
80. AWS Data Exchange is a data analytics service
81. that provides data catalogs, and is more suitable
82. for integrating third-party data
83. into a data lake for further analysis.
84. So, I am not using AWS Data Exchange for now.
85. It seems that the most appropriate suggestion
86. for this architecture is to use one
87. of the Amazon Kinesis services.
88. Amazon Kinesis is a perfect match for this job.
89. It is designed to ingest large amounts
90. of small bits of data.
91. Kinesis is a family of services,
92. and three of them could be used for our scenario.
93. The first is Amazon Kinesis Data Analytics.
94. This service is suitable for real-time data processing
95. of data that is being ingested.
96. That processing can be data transformation, aggregation,
97. filtering, cleaning, or any other manipulation.
98. Since our customer will have control
99. over how the data is being sent,
100. not requiring any transformation at all,
101. I am not going to use Kinesis Data Analytics for now.
102. The remaining services of the Amazon Kinesis family
103. that I'll talk about are Amazon Kinesis Data Streams
104. and Amazon Kinesis Data Firehose.
105. There are some differences among them,
106. which you can find in the next reading of this course.
107. One of the main differences between Kinesis Data Streams
108. and Kinesis Data Firehose is that Kinesis Data Streams
109. requires writing some additional code
110. for the data producer and consumer,
111. so it's less convenient than Kinesis Data Firehose,
112. but it supports lower latency
113. between the moments the data is ingested
114. and when it is available for querying.
115. Mm, this latency thing is something I did not capture
116. in my call with the customer.
117. So let me call them to ask something real quick.
118. (phone rings)
119. - [Morgan] Yello.
120. - [Raf] Hi, Morgan.
121. How are you doing, my friend?
122. - [Morgan] Oh, I'm doing great.
123. I'm actually spending some time
124. out at the golf course, you know, just enjoying my day.
125. How are you doing?
126. - [Raf] Nice, nice.
127. I'm good too, hey, sorry to interrupt your golf.
128. Listen, I have something to ask you.
129. I'm thinking about which service to use
130. as part of the data ingestion mechanism.
131. And I would like to know what your latency requirement
132. is from the moment that you receive the clickstream data
133. from the frontend.
134. I mean, can you afford having a couple
135. of minutes before that data is available in the S3 bucket?
136. - [Morgan] Oh yeah, absolutely.
137. There's really no rush on that at all.
138. This is kind-of analytics
139. for asynchronous processing, anyways,
140. so we don't have anything
141. like real-time dashboards or anything like that.
142. We just need to ingest data to further process it
143. and make it available later.
144. So, there's no real-time dashboards or anything like that.
145. So, some minutes there are totally fine.
146. - [Raf] All right.
147. Thanks a lot for clarifying that.
148. That's all I need to decide which service I will pick.
149. Enjoy your golf session.
150. - [Morgan] All right, yeah, thanks Raf.
151. And call me if you need anything else.
152. - [Raf] Yeah, bye-bye Morgan, always good talking to you.
153. - [Morgan] Bye. - [Raf] Bye.
154. All right, so the customer does not have
155. low latency requirements.
156. So, I am definitely using Kinesis Data Firehose, here,
157. because it is more convenient.
158. I think Morgan would appreciate not having
159. to write additional code, especially
160. because they are short staffed.
161. As you can see, the difference
162. between Kinesis Data Streams and Kinesis Data Firehose
163. is mostly convenience versus control,
164. and there is no best one.
165. There is only the most recommended according to a scenario.
166. In our case, the customer is looking for convenience.
167. Now, the thing is, in order to interact with Amazon Kinesis,
168. we need to perform Kinesis API calls, such as PutRecord,
169. regardless whether we are using Kinesis Data Firehose
170. or Kinesis Data Streams.
171. There are at least two services that could help with that.
172. So, let me put the question mark here
173. because a service needs to do that integration,
174. and let me bring Amazon Cognito.
175. The first is by using an authentication service,
176. like Amazon Cognito.
177. It would provide an identity pool
178. where you can use AWS credentials
179. in order to perform Kinesis API operations.
180. But there is one customer requirement that will
181. make using Amazon Cognito a little more difficult.
182. Do you remember which requirements I'm thinking about?
183. Remember that the customer already has a JavaScript library
184. that they wants to use, and that JS library
185. is looking to make an HTTPS POST.
186. You cannot do that directly to Kinesis.
187. You have to use one of the Kinesis API calls.
188. What service do you think we could use to proxy
189. these HTTPS requests to Kinesis in order to avoid
190. exposing our Kinesis Data Firehose delivery streams
191. over the internet?
192. If you thought about API Gateway, you got it right.
193. So let me consider API Gateway here, as well.
194. We can use the API Gateway service integration
195. to send data out to our Kinesis stream.
196. That way, our customer does not need to change
197. the logic in their JavaScript library.
198. So, I am adding API Gateway to this architecture right now,
199. pretty close to Amazon Kinesis, in our architecture diagram.
200. API Gateway is actually not a data ingestion service itself,
201. but it can be used to proxy things and proxy
202. the HTTPS requests to other AWS services,
203. which is very handy in this.
204. [End of transcript. Skip to the start.](https://courses.edx.org/xblock/block-v1:AWS+AWS-ARCH-1+3T2022+type@vertical+block@3e6eb5af948a4683916e01bd30776184?exam_access=&recheck_access=1&show_bookmark=0&show_title=0&view=student_view#transcript-start-f0ba99708b08466a87dbf7154bd250de)

**Amazon Kinesis Family**  
The Amazon Kinesis Family makes it easier to collect, process, and analyze real-time, streaming data so that you can get timely insights and react quickly to new information. Amazon Kinesis offers key capabilities to cost-effectively process streaming data at virtually any scale, along with the flexibility to choose the tools that best suit the requirements of your application. With Amazon Kinesis, you can ingest real-time data such as video, audio, application logs, website clickstreams, and Internet of Things (IoT) telemetry data for machine learning, analytics, and other applications. You can use Amazon Kinesis to process and analyze data as it arrives, which means that you can respond quickly instead of waiting until all your data is collected before it’s processed.  
  
For more information, see [Amazon Kinesis](https://aws.amazon.com/kinesis/).  
  
**Amazon Kinesis Data Streams**  
Amazon Kinesis Data Streams is a massively scalable and durable real-time data streaming service. Kinesis Data Streams is designed to continuously capture gigabytes of data per second from hundreds of thousands of sources, such as website clickstreams, database event streams, financial transactions, social media feeds, IT logs, and location-tracking events. The data that’s collected is typically available in milliseconds, which means that you can use Kinesis Data Streams with use cases for real-time analytics, such as real-time dashboards, real-time anomaly detection, dynamic pricing, and more.  
  
For more information, see [Amazon Kinesis Data Streams](https://aws.amazon.com/kinesis/data-streams/).  
  
**Amazon Kinesis Data Firehose**  
Amazon Kinesis Data Firehose is designed to reliably load streaming data into data lakes, data stores, and analytics services. It can capture, transform, and deliver streaming data to Amazon Simple Storage Service (Amazon S3), Amazon Redshift, Amazon Elasticsearch Service, generic HTTP endpoints, and service providers like Datadog, New Relic, MongoDB, and Splunk. It is a fully managed service that automatically scales to match the throughput of your data and requires virtually no ongoing administration. It can also batch, compress, transform, and encrypt your data streams before loading, which minimizes the amount of storage that you use and increases security. Although it’s easier to operate when compared with Amazon Kinesis Data Streams, Kinesis Data Firehose delivery streams have a higher latency from the moment that data is ingested. For example, you can set the batch interval to 60 seconds if you want to receive new data within 60 seconds of sending it to your delivery stream. However, you could have latencies that are lower than 1 second when you use Amazon Kinesis Data Streams.  
  
For more information, see [Amazon Kinesis Data Firehose](https://aws.amazon.com/kinesis/data-firehose/?kinesis-blogs.sort-by=item.additionalFields.createdDate&kinesis-blogs.sort-order=desc).  
  
**Amazon Kinesis Data Analytics**  
Amazon Kinesis Data Analytics is designed to transform and analyze streaming data in real time with Apache Flink. Apache Flink is an open-source framework and engine for processing data streams. Amazon Kinesis Data Analytics reduces the complexity of building, managing, and integrating Apache Flink applications with other AWS services. Amazon Kinesis Data Analytics is designed to take care of everything that’s required to run streaming applications continuously. It also scales automatically to match the volume and throughput of your incoming data. With Amazon Kinesis Data Analytics, there are no servers to manage, no minimum fee or setup cost, and you only pay for the resources that your streaming applications consume.  
  
For more information, see [Amazon Kinesis Data Analytics](https://aws.amazon.com/kinesis/data-analytics/).  
  
**Amazon Kinesis Video Streams**  
Amazon Kinesis Video Streams is designed to securely stream video from connected devices to AWS for analytics, machine learning (ML), playback, and other forms of processing. Kinesis Video Streams automatically provisions and elastically scales the infrastructure that’s needed to ingest streaming video data from millions of devices. It’s designed to durably store, encrypt, and index video data in your streams, and you can access your data through the Kinesis Video Streams APIs. You can use Kinesis Video Streams to play back video for live and on-demand viewing. You can also use it to quickly build applications that take advantage of computer vision and video analytics through integration with Amazon Rekognition Video and libraries for ML frameworks (such as Apache MXNet, TensorFlow, and OpenCV). Kinesis Video Streams also supports WebRTC, an open-source project that uses APIs to facilitate real-time media streaming and interaction between web browsers, mobile applications, and connected devices. Typical uses for WebRTC include video chat and peer-to-peer media streaming.  
  
For more information, see [Amazon Kinesis Video Streams](https://aws.amazon.com/kinesis/video-streams/).

1. [Start of transcript. Skip to the end.](https://courses.edx.org/xblock/block-v1:AWS+AWS-ARCH-1+3T2022+type@vertical+block@91198d9c148d40f49119cc3416af36df?exam_access=&recheck_access=1&show_bookmark=0&show_title=0&view=student_view#transcript-end-2df7741e64054f57850034dc9f38203d)
2. - [Raf] For the services in our data analytics architecture,
3. we already know we are using Amazon S3
4. as the storage layer,
5. and Amazon Kinesis for the ingestion mechanism,
6. with API Gateway in front of it.
7. In this video, I will cover some options
8. that can access the ingested data that is now in S3,
9. always remembering matching customer needs
10. in order to find the most appropriate service.
11. Let's get started.
12. As we know, separating data storage from processing
13. is important because we can then use the right tool
14. for the job.
15. In our case, we want to process clickstream data
16. that sits in S3 with large amounts of small files.
17. Let me show you some services that could do that,
18. and discuss them one by one.
19. Candidates are Amazon S3 Select,
20. AWS Glue,
21. Amazon EMR,
22. and Amazon Athena.
23. Starting with Amazon S3 itself.
24. There is an S3 feature called S3 Select
25. that could be used for accessing the data.
26. You can use Amazon S3 Select
27. to perform SQL queries to filter the contents
28. of Amazon S3 objects.
29. It meets customer requirements
30. because it is serverless and managed,
31. and it seems to be a good candidate, right?
32. Well, kind-of, because the issue here
33. is that Amazon S3 Select
34. can only query one file per query, one file at a time,
35. and not lots of small files.
36. Our solution is very likely to produce multiple objects,
37. unless we are running something else
38. to aggregate the data, which we are not doing for now.
39. That being said, I am not using Amazon S3 Select for that.
40. How about AWS Glue or Amazon EMR?
41. These services are strong candidates for data processing,
42. but they really shine when it comes
43. to processing unstructured data using big data frameworks.
44. I remember the customer saying they are running
45. on reduced staff,
46. and these services have a bit of a learning curve.
47. They could be used for converting data formats,
48. performing data aggregation,
49. or using AI/ML when processing data,
50. which, for now, does not seem to be a need.
51. I love AWS Glue, but one of our customer requirements
52. is to keep the architecture the most minimal possible
53. because of reduced staff, cost, and maintenance.
54. If we have that in mind,
55. we can exclude Amazon EMR and AWS Glue,
56. leaving the best choice to be Amazon Athena.
57. Let me open my web browser at the page aws.amazon.com/athena
58. and see what the webpage tells,
59. hoping to read something that matches precisely
60. what we need.
61. Amazon Athena, start querying data instantly.
62. Get results in seconds.
63. Pay only for the queries you run.
64. Amazon Athena is an interactive query service
65. that makes it easy to analyze data in Amazon S3
66. using standard SQL.
67. Athena is serverless,
68. so there is no infrastructure to manage,
69. and you pay only for the queries that you run.
70. Athena is easy to use.
71. Simply point to your data in Amazon S3,
72. define the schema,
73. and start querying using standard SQL.
74. Most results are delivered within seconds.
75. With Athena, there is no need for complex ETL jobs
76. to prepare your data for analysis.
77. This makes it easy for anyone with SQL skills
78. to quickly analyze large-scale datasets.
79. Anyone with SQL skills?
80. I'm in.
81. That should be easy enough
82. and not demand any technical skills.
83. This seems to be very well-aligned
84. with our customer needs.
85. Let me dive a little further into Athena
86. to explain you some concepts,
87. just the necessary information to help you understand Athena
88. on a Solutions Architect - Associate level.
89. The first thing to know is that Athena
90. does not make a copy of your data.
91. When you create an Athena table,
92. you have to specify an S3 bucket prefix
93. for where Athena should look for
94. when running your query.
95. That's why the SQL command for creating a table
96. in Athena is CREATE EXTERNAL TABLE.
97. That's why the word EXTERNAL is in there.
98. Data remains on S3,
99. still providing a centralized data repository,
100. and allowing us to use the best tool for the job.
101. When creating Athena tables,
102. you need to create a table that matches the format
103. of your data in the bucket.
104. Let's say the files are in the CSV format
105. and they contain something like CustomerID,
106. SessionTime, session end, and NumberOfClicks
107. in the webpage.
108. You need to create a table that match that structure.
109. You can also create multiple Athena tables
110. and join them.
111. These tables don't even need to reference the same bucket.
112. The only thing we need is having the data in the bucket
113. matching the table schema.
114. So if the data is in CSV,
115. the table needs to be prepared to query CSV data.
116. Same with JSON or other formats.
117. Let me call my customer and ask something
118. about data formats.
119. Okay, take it off, and then put the number.
120. (buttons dialing)
121. - [Raf] Dialing to Morgan.
122. (phone ringing)
123. - [Morgan] Hi, Raf, how you doing?
124. - [Raf] Good, good.
125. I have a question about the JavaScript library.
126. Is this a good time to chat really quick?
127. - [Morgan] Yeah, sure.
128. - [Raf] So, I'm thinking of using Amazon Athena
129. to access the ingested data,
130. but the thing is that Athena needs a serializer/deserializer
131. on the table to be able to understand
132. what is in the files.
133. And among some of the supported SerDes,
134. which is the term for Serializer/Deserializer,
135. there is CSV, JSON, there's also a SerDe
136. for regular expressions,
137. which gives us more flexibility.
138. My question is, are you sending the data
139. in one of these formats, CSV, JSON?
140. I am asking that because I want
141. to avoid using another service
142. to prepare and normalize the data to be friendly for Athena.
143. - [Morgan] Hm, you know, that's a good question, Raf.
144. I'm not sure, but we have some developers
145. in the office today.
146. Let me go ahead and ask them.
147. They're the ones who wrote the code,
148. so just one second.
149. - [Raf] Sure.
150. - [Morgan] Hey, do you guys know
151. what the format of the data's in?
152. Oh, it's in - it's in JSON?
153. Okay, okay, cool, all right, cool.
154. So yeah, it sounds like it's in JSON format.
155. - [Raf] Wow, that helped me a lot
156. because I can ingest data as-is
157. and query with no modification.
158. As long as you send data in the same format,
159. it's all good, that's all I need for today.
160. Thank you, bye bye.
161. - [Morgan] All right, bye.
162. - [Raf] It is clear
163. that Amazon Athena is a good choice here.
164. Now, one thing is to access the data.
165. Another thing is to have a graph.
166. I can use Athena to make SQL queries based
167. on clickstream data.
168. But how about presenting that data in a pie chart?
169. Or comparing data among different queries?
170. That's where BI and analytics applications come in.
171. Stay with me for the next video.
172. [End of transcript. Skip to the start.](https://courses.edx.org/xblock/block-v1:AWS+AWS-ARCH-1+3T2022+type@vertical+block@91198d9c148d40f49119cc3416af36df?exam_access=&recheck_access=1&show_bookmark=0&show_title=0&view=student_view#transcript-start-2df7741e64054f57850034dc9f38203d)
173. [Start of transcript. Skip to the end.](https://courses.edx.org/xblock/block-v1:AWS+AWS-ARCH-1+3T2022+type@vertical+block@81e6afb1ef0843e0988091d7f9cdb16e?exam_access=&recheck_access=1&show_bookmark=0&show_title=0&view=student_view#transcript-end-f8b8f82c0bb74c8ab3842505261a64c1)
174. - [Raf] Let's explore Amazon Athena,
175. a service that allow you to start querying data instantly.
176. I am going to Amazon Athena in the AWS Management Console.
177. And first thing I see when I go to Amazon Athena
178. is an option saying,
179. Explore the query editor.
180. If I choose that option of exploring the query editor,
181. I already have some Athena tables here that are pointing
182. to CSV files in the last three bucket that I have.
183. One of the tables is called year\_2018.
184. And if you click on the three dots to show more options
185. in that table and choose Generate table DDL,
186. DDL as in Data Definition Language,
187. it will send Athena a command that will show me
188. the CREATE TABLE command that was used to create this table.
189. So, CREATE EXTERNAL TABLE
190. because the data actually resides in S3,
191. then table name,
192. and then I have the fields
193. such as vendorid, passenger\_count.
194. This is a table that points to CSV files that I have
195. in a bucket in my account called nyc-tlc-raf.
196. So, the bucket contains prefixes,
197. and within each one of these prefixes,
198. I have CSV files.
199. So, and these files are in CSV,
200. and I am using the SerDe,
201. the Serializer/Deserializer represented by CSV.
202. So, I have fields terminated by comma,
203. so, because they are CSV files.
204. If I do queries in this table,
205. such as, for example,
206. SELECT \* FROM table limit 10
207. and clicking Run,
208. I will have Athena doing a SQL query
209. in CSV files that are in my bucket.
210. This is just dummy data.
211. This is just data that I have to show you
212. this exploratory Athena demo.
213. And this is how easy to use Athena.
214. You create the tables.
215. You point the tables to buckets.
216. Buckets must have files that must be
217. in the same formats that Athena understands
218. how to deal with with these formats,
219. either if they are JSON, or CSV, or Parquet,
220. which is a columnar data storage.
221. Once you have the files in the bucket
222. and the Athena table correctly configured
223. to know where to look for,
224. you can start querying the data,
225. and even joining multiple Athena tables.
226. [End of transcript. Skip to the start.](https://courses.edx.org/xblock/block-v1:AWS+AWS-ARCH-1+3T2022+type@vertical+block@81e6afb1ef0843e0988091d7f9cdb16e?exam_access=&recheck_access=1&show_bookmark=0&show_title=0&view=student_view#transcript-start-f8b8f82c0bb74c8ab3842505261a64c1)
227. [Start of transcript. Skip to the end.](https://courses.edx.org/xblock/block-v1:AWS+AWS-ARCH-1+3T2022+type@vertical+block@d675baea26574adeaee3dd3a16a16032?exam_access=&recheck_access=1&show_bookmark=0&show_title=0&view=student_view#transcript-end-38bfd504e8fd4b37afa4daeae474e3a5)
228. - [Raf] Our data analytics architecture is almost ready,
229. and I can't wait to show it to the customer.
230. We have decided to use API Gateway,
231. and Amazon Kinesis Data Firehose for data ingestion,
232. Amazon S3 for storage,
233. and Amazon Athena for acquiring the data.
234. Now, it is time to add a data visualization layer
235. to complete the solution.
236. The visualization layer needs to connect with Amazon Athena,
237. which is our query service.
238. There are multiple data visualization options in AWS.
239. Among them, candidates for our needs are
240. Amazon CloudWatch,
241. Amazon OpenSearch Service,
242. Amazon Managed Grafana,
243. and Amazon QuickSight.
244. Let me briefly discuss each one,
245. starting with Amazon CloudWatch.
246. Amazon CloudWatch offers dashboards,
247. and that's why I listed it here, under data visualization.
248. Amazon CloudWatch monitors your AWS resources in real time,
249. but it is primarily a monitoring service
250. for operational metrics,
251. such as CPU utilization for EC2 instances,
252. number of objects in an S3 bucket,
253. and things that are more related
254. to the operation of the services.
255. Although it can create alarms and detect anomalous data,
256. it is not intended to be used for BI
257. as part of a data analytics architecture.
258. Amazon CloudWatch is more used
259. by system administrators looking for operational metrics.
260. So, I am not going to use it
261. for our data visualization, here.
262. Amazon OpenSearch Service would also give me a dashboard.
263. OpenSearch is a distributed, community-driven,
264. Apache 2.0 licensed,
265. 100 percent open-source search and analytics suite
266. that is used for a broad set of use cases.
267. The issue here is that when I create an OpenSearch cluster,
268. it comes with storage and processing power embedded.
269. And I want to decouple that by using Amazon S3 for storage.
270. Because it has processing,
271. the cost of that cluster may be higher
272. than the scenario using Athena for query
273. and another service only providing data visualization.
274. I love Amazon OpenSearch Service,
275. but there are better services
276. to satisfy the requirements here.
277. Another option would be Amazon Managed Grafana.
278. Grafana is a popular open-source analytics platform
279. that you can use to query, visualize, alert,
280. and understand your metrics,
281. no matter where they are stored.
282. Grafana really shines
283. when you need to visualize time series data.
284. Time series databases are built specifically
285. for handling metrics or measurements that are timestamped.
286. Since our clickstream data may have that property,
287. it could work.
288. I will keep it on the table for now.
289. Although Grafana dashboards are mostly used
290. for monitoring and operational dashboards.
291. It is pretty well suited for BI and data analytics, as well.
292. Another offering is Amazon QuickSight,
293. which is a data visualization service that integrates
294. with other AWS services, such as,
295. guess what, Amazon Athena.
296. This makes it an ideal candidate for our architecture.
297. Amazon QuickSight has no servers to manage,
298. offers a pay-per-use pricing model,
299. provides native AWS service integrations
300. with built-in security,
301. also enabling data insights.
302. Hmm...
303. Since both Amazon Managed Grafana
304. and Amazon QuickSight can do the job here,
305. let me call the customer and ask
306. if they are already using one
307. of these services for another project,
308. because if they are,
309. I can use the same one for our project,
310. saving on potential licensing costs
311. or reducing learning curve.
312. Remember that cost savings is a requirement.
313. (phone ringing)
314. - [Morgan] Hello again, my friend.
315. What's up?
316. - [Raf] Hi Morgan.
317. Is this a good time to talk?
318. - [Morgan] Sure, yeah.
319. How's the architecture going?
320. I've heard some good things so far.
321. - [Raf] Yeah, it's going pretty well.
322. I am now choosing what data visualization service to use.
323. And I have a question for you.
324. By any chance, by any chance,
325. are you already using Amazon Managed Grafana
326. or Amazon QuickSight for another project?
327. - [Morgan] You know, now that I think about it,
328. yeah, we are.
329. We are using Amazon QuickSight
330. for the sales dashboards and another project.
331. I'm not sure if we can use the same license there,
332. but we would prefer using it
333. because our BI team is already familiar with the tool.
334. - [Raf] Right, yeah.
335. That's good to hear.
336. I was in between choosing Amazon Managed Grafana
337. or Amazon QuickSight,
338. but knowing that your team already knows
339. how to use Amazon QuickSight makes me think,
340. as the service used for data visualization here,
341. sometimes the best service is the service
342. that your team already knows how to use, right?
343. - [Morgan] Yeah, definitely.
344. I agree.
345. We can have some more internal trainings here,
346. so that other people can help the BI analysts
347. learn how to provide insights for the restaurants
348. by looking at the QuickSight dashboards.
349. So, this sounds great and would be super handy.
350. - [Raf] Okay.
351. I got all I need, the architecture is almost finished.
352. I'll call you soon to schedule a date for you to come
353. at our office to see my architecture proposal.
354. Is that good?
355. - [Morgan] All right, looking forward to it.
356. Talk to you later.
357. - [Raf] Cool, bye-bye.
358. All right.
359. Amazon QuickSight it is, then.
360. I think this closes the gap between the part
361. that is in scope for our architecture.
362. In the next video,
363. I'll call the customer again to show the full picture.
364. [End of transcript. Skip to the start.](https://courses.edx.org/xblock/block-v1:AWS+AWS-ARCH-1+3T2022+type@vertical+block@d675baea26574adeaee3dd3a16a16032?exam_access=&recheck_access=1&show_bookmark=0&show_title=0&view=student_view#transcript-start-38bfd504e8fd4b37afa4daeae474e3a5)

QuickSight powers millions of dashboard views weekly for customers, which means that their end users can make better, data-driven decisions. Here are some key features and benefits of QuickSight:

* Connect and scale all your data
  + Connect to all your data in AWS, in third-party cloud service providers, or on premises
  + Use SPICE in-memory storage to scale data exploration to thousands of users
  + Combine data from multiple sources and create complex data models for governed data sharing
* Build customizable dashboards
  + Use dashboard design to create customized, use-case-specific dashboards
  + Deliver personalized email reports and alerts to end users
  + Access information from virtually anywhere by using QuickSight access for iOS, Android, or mobile web applications
* Use machine learning (ML) integrations for insights
  + Use Anomaly Detection to continuously analyze all your data for anomalies and variations
  + Forecast business metrics and perform interactive what-if analyses
  + Customize Auto-Narratives and weave them into dashboards to provide deeper context for users
* Enable self-service business intelligence (BI) for everyone
  + Dive deep into data through simple questions, without BI training
  + Create visual analyses of data by using a web-based authoring interface
  + Embed QuickSight capabilities in applications for data-driven user experiences
* Use native integration with AWS
  + Use private virtual private cloud (VPC) connectivity for secure AWS access to Amazon Redshift, Snowflake, Exasol, Amazon Relational Database Service (Amazon RDS), and more
  + Use native AWS Identity and Access Management (IAM) permissions for Amazon Simple Storage Service (Amazon S3) and Amazon Athena, with fine-grained access control for serverless data exploration
  + Use Amazon SageMaker integration to incorporate sophisticated ML models without complex data pipelines
* Take advantage of no servers to manage, and paying by usage
  + Automatically scale serverless architecture to hundreds of thousands of users with high availability, with no need to overprovision for peak usage
  + Provide consistent, fast response times for end users and analysts through the auto-scaling capabilities of the SPICE in-memory engine, with no need to scale databases for high workloads
  + Optimize costs through the pay-per-session model by paying only for actual usage, with no need to buy thousands of end-user licenses for large-scale BI or embedded analytics
* Take advantage of built-in security, governance, and compliance
  + Use end-to-end data encryption for data, and use encryption at rest for data in SPICE
  + Use row-level and column-level security with API support for control at the user or group level.
  + Take advantage of rapid deployment for regulated workloads with compliance support for the Health Insurance Portability and Accountability Act (HIPAA), HITRUST CSF, General Data Protection Regulation (GDPR), System and Organizational Controls (SOC), Payment Card Industry (PCI), International Organization for Standardization (ISO) 27001, Federal Risk and Authorization Management Program (FedRAMP) High, and more

You are a solutions architect who is working with a customer to design solutions on AWS to meet their business needs. Your customer wants to create a solution for ingesting data into a data lake on AWS. This data lake must be scalable and highly available. It must also be able to store both structured and unstructured data in the same storage location. After the data is stored, it will need to be transformed into different formats so that it can be used for analytics and to derive insights. The customer wants to store raw data without any modification, and they also want to store processed data in a separate location.   
  
Write a few paragraphs that describe which AWS services you might use for this architecture. Define the scope of each service, and explain how they work together to form a solution to solve the customer’s business need.

### Discussion - Consider this Scenario

Topic: Week 2 / Consider this Scenario-Week2

Filter:

Sort:

### Consider this Scenario

[discussion](https://courses.edx.org/courses/course-v1:AWS+AWS-ARCH-1+3T2022/discussion/forum/2a84d5b4006efe68ed5d4700f7a2ce6405ee8968/threads/6664cb3c507a7a04d9867834)

[Solution propose](https://courses.edx.org/courses/course-v1:AWS+AWS-ARCH-1+3T2022/discussion/forum/2a84d5b4006efe68ed5d4700f7a2ce6405ee8968/threads/6664cb3c507a7a04d9867834)

[S3(Simple Storage Service) to store raw and processed data to act as data lake, supporting scalability and availability to store strucutred and unstructured data. Glue to perform Extract, transfor and Load to transform raw data and save in another format storing in a s3 bucket as well.](https://courses.edx.org/courses/course-v1:AWS+AWS-ARCH-1+3T2022/discussion/forum/2a84d5b4006efe68ed5d4700f7a2ce6405ee8968/threads/6664cb3c507a7a04d9867834)

[1 comments](https://courses.edx.org/courses/course-v1:AWS+AWS-ARCH-1+3T2022/discussion/forum/2a84d5b4006efe68ed5d4700f7a2ce6405ee8968/threads/6664cb3c507a7a04d9867834)

[discussion](https://courses.edx.org/courses/course-v1:AWS+AWS-ARCH-1+3T2022/discussion/forum/2a84d5b4006efe68ed5d4700f7a2ce6405ee8968/threads/6629dcdc89a40604ce9f2c5e)

[architecture solution](https://courses.edx.org/courses/course-v1:AWS+AWS-ARCH-1+3T2022/discussion/forum/2a84d5b4006efe68ed5d4700f7a2ce6405ee8968/threads/6629dcdc89a40604ce9f2c5e)

[Amazon web services](https://courses.edx.org/courses/course-v1:AWS+AWS-ARCH-1+3T2022/discussion/forum/2a84d5b4006efe68ed5d4700f7a2ce6405ee8968/threads/6629dcdc89a40604ce9f2c5e)

[1 comments](https://courses.edx.org/courses/course-v1:AWS+AWS-ARCH-1+3T2022/discussion/forum/2a84d5b4006efe68ed5d4700f7a2ce6405ee8968/threads/6629dcdc89a40604ce9f2c5e)

[discussion](https://courses.edx.org/courses/course-v1:AWS+AWS-ARCH-1+3T2022/discussion/forum/2a84d5b4006efe68ed5d4700f7a2ce6405ee8968/threads/65f1cfa85c98f204a497142b)

[Possible solution](https://courses.edx.org/courses/course-v1:AWS+AWS-ARCH-1+3T2022/discussion/forum/2a84d5b4006efe68ed5d4700f7a2ce6405ee8968/threads/65f1cfa85c98f204a497142b)

[Use Amazon S3, Athena could be great choices.](https://courses.edx.org/courses/course-v1:AWS+AWS-ARCH-1+3T2022/discussion/forum/2a84d5b4006efe68ed5d4700f7a2ce6405ee8968/threads/65f1cfa85c98f204a497142b)

[1 comments](https://courses.edx.org/courses/course-v1:AWS+AWS-ARCH-1+3T2022/discussion/forum/2a84d5b4006efe68ed5d4700f7a2ce6405ee8968/threads/65f1cfa85c98f204a497142b)

[discussion](https://courses.edx.org/courses/course-v1:AWS+AWS-ARCH-1+3T2022/discussion/forum/2a84d5b4006efe68ed5d4700f7a2ce6405ee8968/threads/65d9224fa44b7304a0e83d25)

[create a scenario](https://courses.edx.org/courses/course-v1:AWS+AWS-ARCH-1+3T2022/discussion/forum/2a84d5b4006efe68ed5d4700f7a2ce6405ee8968/threads/65d9224fa44b7304a0e83d25)

[To create a scalable and highly available data lake on AWS for both structured and unstructured data, we'll use Amazon S3. It's like a big, reliable storage space. For getting data into this lake, we can use AWS Glue, which helps organize and prepare the data. The original, untouched data goes straight into a specific S3 space, and the processed data goes into a different one. When it's time to analyze the data, we can use Amazon Athena. It lets us ask questions about the data stored in S3 without any fancy setup. So, in simple terms, S3 holds all the data, Glue helps organize it, and Athena lets us look for insights without making things complicated. This way, the customer can store, process, and analyze their data as they need.](https://courses.edx.org/courses/course-v1:AWS+AWS-ARCH-1+3T2022/discussion/forum/2a84d5b4006efe68ed5d4700f7a2ce6405ee8968/threads/65d9224fa44b7304a0e83d25)

[1 comments](https://courses.edx.org/courses/course-v1:AWS+AWS-ARCH-1+3T2022/discussion/forum/2a84d5b4006efe68ed5d4700f7a2ce6405ee8968/threads/65d9224fa44b7304a0e83d25)

[discussion](https://courses.edx.org/courses/course-v1:AWS+AWS-ARCH-1+3T2022/discussion/forum/2a84d5b4006efe68ed5d4700f7a2ce6405ee8968/threads/65ce74e74a178904b0b45e92)

[Solution for Customer](https://courses.edx.org/courses/course-v1:AWS+AWS-ARCH-1+3T2022/discussion/forum/2a84d5b4006efe68ed5d4700f7a2ce6405ee8968/threads/65ce74e74a178904b0b45e92)

[Here are the Amazon services that Morgan will use. Amazon API Gateway --> Kenesis Data Firehose --> Amazon S3 --> Amazon Athena --> Amazon QuickSight](https://courses.edx.org/courses/course-v1:AWS+AWS-ARCH-1+3T2022/discussion/forum/2a84d5b4006efe68ed5d4700f7a2ce6405ee8968/threads/65ce74e74a178904b0b45e92)

[1 comments](https://courses.edx.org/courses/course-v1:AWS+AWS-ARCH-1+3T2022/discussion/forum/2a84d5b4006efe68ed5d4700f7a2ce6405ee8968/threads/65ce74e74a178904b0b45e92)

[discussion](https://courses.edx.org/courses/course-v1:AWS+AWS-ARCH-1+3T2022/discussion/forum/2a84d5b4006efe68ed5d4700f7a2ce6405ee8968/threads/65c11ed9752ae404c8d18a7d)

[Designing a Scalable and Highly Available AWS Data Lake Architecture for Enhanced Analytics and Insight Generation](https://courses.edx.org/courses/course-v1:AWS+AWS-ARCH-1+3T2022/discussion/forum/2a84d5b4006efe68ed5d4700f7a2ce6405ee8968/threads/65c11ed9752ae404c8d18a7d)

[the architecture will leverage Amazon S3 for scalable and reliable storage, AWS Glue for data cataloging and transformation, Amazon Kinesis for data ingestion, Amazon Athena or Amazon Redshift for analytics, AWS Lake Formation for managing the data lake, and Amazon S3 Glacier for archival needs. Together, these services will provide a comprehensive, scalable, and efficient solution for creating and managing a data lake on AWS, catering to both the storage and analytical needs of the customer.](https://courses.edx.org/courses/course-v1:AWS+AWS-ARCH-1+3T2022/discussion/forum/2a84d5b4006efe68ed5d4700f7a2ce6405ee8968/threads/65c11ed9752ae404c8d18a7d)

[1 comments](https://courses.edx.org/courses/course-v1:AWS+AWS-ARCH-1+3T2022/discussion/forum/2a84d5b4006efe68ed5d4700f7a2ce6405ee8968/threads/65c11ed9752ae404c8d18a7d)

[discussion](https://courses.edx.org/courses/course-v1:AWS+AWS-ARCH-1+3T2022/discussion/forum/2a84d5b4006efe68ed5d4700f7a2ce6405ee8968/threads/653b80cd756c080496285fb6)

[Ingesting data into a data lake Solution](https://courses.edx.org/courses/course-v1:AWS+AWS-ARCH-1+3T2022/discussion/forum/2a84d5b4006efe68ed5d4700f7a2ce6405ee8968/threads/653b80cd756c080496285fb6)

[API gateway - facade Kinesis Analytics - storage and transformation S3 - store processed and raw data from kinesis Clicksight - visualization](https://courses.edx.org/courses/course-v1:AWS+AWS-ARCH-1+3T2022/discussion/forum/2a84d5b4006efe68ed5d4700f7a2ce6405ee8968/threads/653b80cd756c080496285fb6)

[1 comments](https://courses.edx.org/courses/course-v1:AWS+AWS-ARCH-1+3T2022/discussion/forum/2a84d5b4006efe68ed5d4700f7a2ce6405ee8968/threads/653b80cd756c080496285fb6)

[discussion](https://courses.edx.org/courses/course-v1:AWS+AWS-ARCH-1+3T2022/discussion/forum/2a84d5b4006efe68ed5d4700f7a2ce6405ee8968/threads/64efeb840b881604f9e3baf3)

[Consider this Scenario: Scalable Storage](https://courses.edx.org/courses/course-v1:AWS+AWS-ARCH-1+3T2022/discussion/forum/2a84d5b4006efe68ed5d4700f7a2ce6405ee8968/threads/64efeb840b881604f9e3baf3)

[One possible solution for creating a data lake on AWS is to use the following services: - Amazon S3: This is a scalable and highly available object storage service that can store any type of data, such as structured, unstructured, or semi-structured data. Amazon S3 can also store data in different formats, such as JSON, CSV, Parquet, or ORC. Amazon S3 can be used as the primary storage location for the data lake, where raw data can be stored in one S3 bucket, and processed data can be stored in another S3 bucket. Amazon S3 also supports encryption, versioning, lifecycle management, and access control policies to secure and manage the data. - AWS Glue: This is a fully managed service that provides a serverless data integration platform. AWS Glue can be used to discover, catalog, and prepare the data for analytics. AWS Glue can crawl the data in Amazon S3 and create a metadata catalog that describes the data structure and schema. AWS Glue can also run ETL (extract, transform, and load) jobs that can transform the data into different formats and store them in the processed data S3 bucket. AWS Glue can also use AWS Glue DataBrew, a visual data preparation tool, to clean and enrich the data. - Amazon Athena: This is a serverless interactive query service that can analyze data in Amazon S3 using standard SQL. Amazon Athena can be used to query both raw and processed data in the data lake without any loading or transformation. Amazon Athena can leverage the AWS Glue catalog to access the metadata and schema of the data. Amazon Athena can also integrate with other AWS services, such as Amazon QuickSight, a business intelligence tool, to visualize and share the insights derived from the data. By using these services, the customer can create a scalable and highly available data lake on AWS that can store and process both structured and unstructured data in the same storage location. The customer can also use various tools and services to analyze the data and derive insights to meet their business needs.](https://courses.edx.org/courses/course-v1:AWS+AWS-ARCH-1+3T2022/discussion/forum/2a84d5b4006efe68ed5d4700f7a2ce6405ee8968/threads/64efeb840b881604f9e3baf3)

[1 comments](https://courses.edx.org/courses/course-v1:AWS+AWS-ARCH-1+3T2022/discussion/forum/2a84d5b4006efe68ed5d4700f7a2ce6405ee8968/threads/64efeb840b881604f9e3baf3)

[discussion](https://courses.edx.org/courses/course-v1:AWS+AWS-ARCH-1+3T2022/discussion/forum/2a84d5b4006efe68ed5d4700f7a2ce6405ee8968/threads/64c7c9862a472d04a586b713)

[Use Kinesis,AWS Lambda](https://courses.edx.org/courses/course-v1:AWS+AWS-ARCH-1+3T2022/discussion/forum/2a84d5b4006efe68ed5d4700f7a2ce6405ee8968/threads/64c7c9862a472d04a586b713)

[- add API Gateway to get Kinesis ingests data directly and puts that unstructured data to AWS S3 - add AWS Lambda that would be triggered to process commin in data put them to other AWS S3.](https://courses.edx.org/courses/course-v1:AWS+AWS-ARCH-1+3T2022/discussion/forum/2a84d5b4006efe68ed5d4700f7a2ce6405ee8968/threads/64c7c9862a472d04a586b713)

[1 comments](https://courses.edx.org/courses/course-v1:AWS+AWS-ARCH-1+3T2022/discussion/forum/2a84d5b4006efe68ed5d4700f7a2ce6405ee8968/threads/64c7c9862a472d04a586b713)

[discussion](https://courses.edx.org/courses/course-v1:AWS+AWS-ARCH-1+3T2022/discussion/forum/2a84d5b4006efe68ed5d4700f7a2ce6405ee8968/threads/64b5795f0a891e04b20a7d31)

[week 2, task](https://courses.edx.org/courses/course-v1:AWS+AWS-ARCH-1+3T2022/discussion/forum/2a84d5b4006efe68ed5d4700f7a2ce6405ee8968/threads/64b5795f0a891e04b20a7d31)

[Flow: 1) API gateway to send data via HTTP requests 2) Amazon Glue to ingest data because it can work with different types of data and meets Data Lake reqs 3) S3 (Tier to choose the cheapest way of data storing) to store data separately and in the diff region as anticipated 4) Amazon EMR to process data cause it works with diff formats and work with lots of data 5) No reqs to clean up data so no need to choose AWS Glue DataBrew, so I would go with Grafana or QuickSight](https://courses.edx.org/courses/course-v1:AWS+AWS-ARCH-1+3T2022/discussion/forum/2a84d5b4006efe68ed5d4700f7a2ce6405ee8968/threads/64b5795f0a891e04b20a7d31)

[1 comments](https://courses.edx.org/courses/course-v1:AWS+AWS-ARCH-1+3T2022/discussion/forum/2a84d5b4006efe68ed5d4700f7a2ce6405ee8968/threads/64b5795f0a891e04b20a7d31)

[discussion](https://courses.edx.org/courses/course-v1:AWS+AWS-ARCH-1+3T2022/discussion/forum/2a84d5b4006efe68ed5d4700f7a2ce6405ee8968/threads/6482fab00a891e04b2064e71)

[The Scenario is very accurate to reality.](https://courses.edx.org/courses/course-v1:AWS+AWS-ARCH-1+3T2022/discussion/forum/2a84d5b4006efe68ed5d4700f7a2ce6405ee8968/threads/6482fab00a891e04b2064e71)

[The lecture is very good.](https://courses.edx.org/courses/course-v1:AWS+AWS-ARCH-1+3T2022/discussion/forum/2a84d5b4006efe68ed5d4700f7a2ce6405ee8968/threads/6482fab00a891e04b2064e71)

[1 comments](https://courses.edx.org/courses/course-v1:AWS+AWS-ARCH-1+3T2022/discussion/forum/2a84d5b4006efe68ed5d4700f7a2ce6405ee8968/threads/6482fab00a891e04b2064e71)

[discussion](https://courses.edx.org/courses/course-v1:AWS+AWS-ARCH-1+3T2022/discussion/forum/2a84d5b4006efe68ed5d4700f7a2ce6405ee8968/threads/63b7568f3caa560502e9ccff)

[Solution for ingesting data into a data lake on AWS](https://courses.edx.org/courses/course-v1:AWS+AWS-ARCH-1+3T2022/discussion/forum/2a84d5b4006efe68ed5d4700f7a2ce6405ee8968/threads/63b7568f3caa560502e9ccff)

[As a first step, a review of the requirements suggests the following: Ingestion of Data into AWS Data Lake. There is no indication of th](https://courses.edx.org/courses/course-v1:AWS+AWS-ARCH-1+3T2022/discussion/forum/2a84d5b4006efe68ed5d4700f7a2ce6405ee8968/threads/63b7568f3caa560502e9ccff)

1. [Start of transcript. Skip to the end.](https://courses.edx.org/xblock/block-v1:AWS+AWS-ARCH-1+3T2022+type@vertical+block@9840939e3c6c45c5bdfed7abec7e11be?exam_access=&format=Quiz&recheck_access=1&show_bookmark=0&show_title=0&view=student_view#transcript-end-b5b4ce70d9c740c3a169153754a3c01b)
2. - [Raf] Our solution seems to be complete,
3. and we are ready to call our customer and talk about it.
4. So, let me call Morgan and explain
5. why I choose these services.
6. (phone ringing)
7. - [Morgan] Hey Raf, how's it going?
8. - [Raf] I'm doing fine, thanks for asking.
9. Came to the office today. Traffic wasn't so bad.
10. How about you?
11. - [Morgan] I'm doing great.
12. Do you have any questions about the architecture?
13. - [Raf] No, I actually finished that,
14. and would like to show you,
15. explaining why I chose one service versus another.
16. Can you come to the office today so I can show you?
17. - [Morgan] Yeah, sure. I can do that. I can drive on in.
18. I'll be over there soon.
19. (video game beeps)
20. (popping sound)
21. - [Raf] Wow, that was fast, (Morgan laughs)
22. that was fast, I must say.
23. - [Morgan] Yeah, I'm a very fast driver,
24. you know, in the highways, (Raf laughs)
25. there wasn't a lot of traffic,
26. so, you know, I got here really quick.
27. - [Raf] Awesome. Ready to get started in the architecture?
28. - [Morgan] I am. I'm ready to see,
29. - [Morgan] let's get started. - [Raf] Okay.
30. - [Morgan] Just as a reminder, one of the things
31. that we were looking for was,
32. we have this clickstream data JavaScript library
33. that we built, - [Raf] Yup.
34. and we need an HTTPS endpoint to send that data to,
35. so that we can then do analytics on it in the background.
36. So, my first question for you is,
37. which one of these services here is providing that
38. - [Morgan] HTTPS? Yeah. - [Raf] HTTPS. Yeah.
39. - [Raf] So, it's API Gateway.
40. It's Amazon API Gateway, which is integrating
41. with Kinesis Data Firehose.
42. There is a service integration between Amazon API Gateway
43. and Kinesis Data Firehose that can
44. deliver data to Firehose.
45. Our primary data ingestion mechanism
46. is Kinesis Data Firehose, but we don't want to expose
47. Kinesis Data Firehose API endpoints on the internet,
48. especially because you are doing HTTPS POSTs
49. from the JavaScript library, right?
50. So I am fronting that with API Gateway,
51. which gives you more ability of specifying
52. your custom domain name, your own SSL certificates
53. if you want, and you don't need to change your code.
54. - [Morgan] Okay, that's awesome.
55. So, it sounds like API Gateway's
56. basically just acting as a proxy for this architecture,
57. and then Data Firehose is the actual
58. data ingestion mechanism.
59. - [Raf] Yes, exactly. - [Morgan] Okay, awesome.
60. So, it looks like Firehose then sends that data to
61. - [Morgan] S3. So - - [Raf] S3, yeah.
62. - [Raf] And I'm choosing S3, here.
63. I had many storage options services to choose from,
64. and the reason why I chose S3 is because it allows me,
65. choosing the right tool for the job.
66. In other words, if I want an ML data processing engine,
67. I can use S3 with that.
68. If I want to have data delivered
69. to another Kinesis data stream, or if if I wanted
70. to have execution of Lambda functions, for example,
71. I can use S3 for that, because S3 is detached
72. from any processing or compute or visualization layer,
73. allowing me to use the one that best fits for the job.
74. Which in our case is, Amazon Athena, right?
75. So, remember when I called you asking if the data is in
76. CSV or JSON or any other format?
77. - [Morgan] Yeah, I do remember that.
78. - [Raf] Yeah, it is because data will come
79. from your frontend through API Gateway,
80. Kinesis Firehose, it will land in an S3 bucket.
81. And once this data in, is in the S3 bucket,
82. you're going to use an Amazon Athena table.
83. So you're going to do a SQL query on the table,
84. and the table must match the same structure
85. that you have in the files.
86. - [Morgan] Got it. Okay, so with Athena,
87. we can just use regular SQL for that?
88. We don't have to, like, learn anything new for that?
89. - [Raf] You can use regular SQL, yeah.
90. - [Morgan] Okay, yeah, that's great news.
91. - [Raf] Yeah, and because you are controlling
92. how that data is produced,
93. we are not modifying the data before it lands
94. in the S3 bucket, so as long as you send the data
95. on the same way it is produced, which are JSON files.
96. And you have the table schema in Athena,
97. pointing to the S3 bucket, you can do SQL queries.
98. Now, when you do SQL queries in Amazon Athena,
99. you're going to have a table as an output of the query.
100. - [Morgan] Right, and so the output is just like a list
101. - [Morgan] of data, right? - [Raf] Yes.
102. - [Morgan] And so, ideally we want to be able
103. to visualize that data,
104. and actually draw insights from it.
105. So, I'm assuming that's where QuickSight comes in.
106. - [Raf] Exactly, yeah. Remember when I called you about
107. Amazon QuickSight or Amazon Managed Grafana,
108. if you're already using one of these?
109. - [Morgan] Yes.
110. - [Raf] Is because I was thinking about which
111. data visualization service to choose from, from the list.
112. - [Morgan] Okay, great. And then,
113. we're already using QuickSight
114. like I had mentioned, so that's great to hear,
115. because then we can just reuse that license,
116. and we don't have to train our business intelligence team
117. on a new tool.
118. But something you mentioned, I thought was interesting.
119. So, you're saying we're not modifying the data as it comes
120. through this pipeline, here, but we have
121. potentially thousands of restaurants,
122. all generating clickstream data.
123. And so, what would happen if one of those websites
124. was misconfigured, and was sending data in the wrong format?
125. How would we fix that?
126. - [Raf] Very good question. So far in this architecture,
127. we are not modifying the data that comes, but if you want to
128. have that option, there are some data manipulation
129. mechanisms in Kinesis Firehose.
130. And there is also a feature called bucket actions,
131. where you can run Lambda function code
132. in response to PutObject events in the bucket.
133. So, for every single JSON file that you have,
134. and Firehose segregates them,
135. so for every file that you have that becomes an object
136. in the bucket, you can run a Lambda function,
137. and that Lambda function can sanitize the data.
138. For example, by cleaning fields that came
139. that shouldn't be there, or things like that.
140. - [Morgan] Right. Because as we know, this is the real world
141. - [Morgan] right? - [Raf] The real world.
142. - [Morgan] Things don't ever work the way, exactly the way
143. - [Morgan] we want them to. - [Raf] Yeah, yeah.
144. - [Morgan] So, that makes sense.
145. So, we're going to use Firehose to send to S3.
146. And it also, it's kind-of cool
147. that it doesn't do every single data point,
148. but it will aggregate into files.
149. And then, that way the Lambda function isn't
150. running all the time, it's just when a new file gets added?
151. - [Raf] Only when new files
152. - [Raf] get added. - [Morgan] Okay, awesome.
153. You could also have the option of preserving the original
154. data in one S3 bucket, and then, when you run
155. the Lambda functions to modify that data,
156. you can use a secondary bucket.
157. This is a pretty common approach as well.
158. - [Morgan] Okay. Yeah, I think I understand.
159. - [Raf] This way, I am expecting that no data modification
160. will be needed, but if you need, you can work
161. with two S3 buckets.
162. One for raw data that comes,
163. and one for the processed data.
164. And that Lambda function that runs can do other things
165. as well, like querying DynamoDB tables,
166. or querying RDS databases, and enrich that data,
167. and put that enriched data in a secondary.
168. - [Morgan] So then, we could have even more data
169. to draw insights from at that point.
170. - [Raf] Yes, - [Morgan] Okay.
171. - [Raf] you could have that option.
172. - [Morgan] Well, I think this looks awesome.
173. It seems to meet all of our requirements
174. that we had set out.
175. So, I'm really excited to take this back to the team,
176. and I'm also really excited that everything's serverless,
177. and it also doesn't require us writing any code.
178. So, I think this is great.
179. And thank you so much for putting it together.
180. - [Raf] Yeah, sure. I remember that cost was a concern.
181. So I only chose services that bills per refined billing.
182. - [Morgan] Yeah, I appreciate that.
183. - [Raf] There's no cost per hour if you're not using,
184. so at 2:00 AM, 3:00 AM, 4:00 AM,
185. where the restaurants don't have people,
186. and you're not generating insights.
187. The cost is lower because all of these services here,
188. they bill by a granular usage.
189. - [Morgan] Okay, perfect.
190. - [Raf] So yeah, I hope you enjoy.
191. Let's present that to your team later,
192. and thanks for trusting me on designing
193. the data analytics solution for you.
194. - [Morgan] Yep, thank you.
195. - [Raf] This was the proposal
196. of the architecture to the customer.
197. In this meeting, Morgan also had the chance to ask questions
198. and ask me to rethink some aspects of it.
199. There may be some last-minute requirements that would need
200. an architecture revision, which is not ideal,
201. but unfortunately, can happen.
202. As a solutions architect, I can stop here,
203. or I can move further, possibly with additional
204. consulting hours or implementing a proof of concept.
205. It various case by case.
206. In our scenario, the architecture was simple
207. with the usage of just five AWS services,
208. which was ideal for this course, so I could talk about them
209. a little bit, and explain my reasons on which one to choose.
210. It may make sense for solutions architects
211. to stop at the architectural part,
212. because scenarios can get way more complex
213. with third-party integration for identity federation,
214. dozens of AWS services, data ingestion
215. from different sources, and et cetera.
216. I hope this week helped you learn more
217. about each AWS service that we presented.
218. Again, our goal is not to double-click and dive deep
219. into each of the services, but instead, show you an approach
220. for how to decide which service to use,
221. according to the requirements and customer scenario.
222. [End of transcript. Skip to the start.](https://courses.edx.org/xblock/block-v1:AWS+AWS-ARCH-1+3T2022+type@vertical+block@9840939e3c6c45c5bdfed7abec7e11be?exam_access=&format=Quiz&recheck_access=1&show_bookmark=0&show_title=0&view=student_view#transcript-start-b5b4ce70d9c740c3a169153754a3c01b)

[Skip to main content](https://courses.edx.org/xblock/block-v1:AWS+AWS-ARCH-1+3T2022+type@vertical+block@604b4669a7ba4de5986e2d3b11bed65a?exam_access=&format=Quiz&recheck_access=1&show_bookmark=0&show_title=0&view=student_view#main)

### Week Wrap-Up: Taking this Architecture to the Next Level

that is out of scope of the solution.

0:52 / 7:47

Press UP to enter the speed menu then use the UP and DOWN arrow keys to navigate the different speeds, then press ENTER to change to the selected speed.

Click on this button to mute or unmute this video or press UP or DOWN buttons to increase or decrease volume level.

Maximum Volume.

### Video transcript

1. [Start of transcript. Skip to the end.](https://courses.edx.org/xblock/block-v1:AWS+AWS-ARCH-1+3T2022+type@vertical+block@604b4669a7ba4de5986e2d3b11bed65a?exam_access=&format=Quiz&recheck_access=1&show_bookmark=0&show_title=0&view=student_view#transcript-end-35af63abca89411f9bb6bf9e51ca6b7f)
2. - [Morgan] All right. So, now that we've built
3. and presented this architecture to the customer,
4. - [Morgan] or me, (Raf laughs)
5. - [Morgan] we will both now be playing
6. the solutions architects,
7. and we'll discuss some of the things that can be enhanced,
8. or at least considered,
9. for the sake of making this architecture better.
10. Rethinking current architectures
11. should be a never-ending effort.
12. And it is a good practice to revisit solutions
13. from time to time because new services may be announced,
14. or conditions may have changed.
15. So, if we're ready to get going here,
16. if we look at the blocks,
17. we have these extra blocks that were not
18. on the table before.
19. So, can you explain a little bit about what those are for?
20. - [Raf] Correct. We have the full picture here, right?
21. So, we have both the part that is in scope,
22. which are the blocks at the bottom of the table,
23. the data analytics solution,
24. API Gateway, Kinesis Data Firehose,
25. Amazon S3, Amazon Athena, and Amazon QuickSight.
26. And on top of the table, we have the part
27. that is out of scope of the solution.
28. Meaning that, customers on their mobile phones,
29. they will scan the QR code.
30. Once they scan, they scan this QR code,
31. they will be served with an HTML webpage out of Amazon S3.
32. So, Amazon S3 is serving that webpage.
33. Customer also has a system where restaurant admins
34. can use that system to update the QR codes.
35. - [Morgan] And this system is currently being run
36. on EC2, I think, right?
37. - [Raf] Yes, yes. - [Morgan] Okay.
38. - [Raf] So, they upload the S3 bucket
39. with new items in the menu.
40. If they wants to change prices,
41. if they wants to change descriptions,
42. whatever modification they wants to do
43. to the restaurant menu.
44. So, it goes to S3.
45. And then, S3 starts serving that webpage
46. for customers who, customers who scan the QR code
47. - [Morgan] Makes sense. All right.
48. So, kind-of knowing that we have this full system here,
49. we're going to also suggest improvements
50. to the existing out of scope.
51. - [Raf] Yes. - [Morgan] So,
52. - [Morgan] with that in mind,
53. I want to make our first suggestion for this architecture.
54. And that would be to potentially look
55. at using a serverless service for this menu update system.
56. So, right now, this is running on EC2 somewhere,
57. and that server is running all of the time,
58. but these restaurants aren't updating their menus
59. all of the time, at all hours of the day.
60. You know, that's not something that happens constantly.
61. So instead, what we could maybe do, is try to change
62. this into a serverless architecture
63. for this part of the system.
64. And so, if we think about it, like, this one block
65. would probably become many blocks, right?
66. As we break it down into multiple serverless components.
67. So that is going to be my first recommendation
68. for this architecture.
69. - [Raf] Yes.
70. Another thing I have in mind here is,
71. now we have two S3 buckets,
72. one for the data analytics
73. that is in scope of what I suggested to the customer.
74. And another one that is out of scope,
75. which is the bucket that is serving the restaurant menus.
76. There is an AWS service called Amazon CloudFront,
77. which is a content delivery network,
78. where you can use Amazon S3 as origin.
79. So, Amazon CloudFront can cache contents before delivering
80. to customers.
81. And it has a cheaper data transfer out
82. when compared to Amazon S3.
83. - [Morgan] Som instead of the clients coming directly
84. to S3 to get the HTML, they would hit
85. - [Morgan] CloudFront first. - [Raf] They would hit
86. - [Raf] CloudFront first. - [Morgan] And so,
87. - [Morgan] if that data,
88. that HTML data, is already cached at CloudFront,
89. that request would never come to S3.
90. - [Raf] Exactly. - [Morgan] And so with S3,
91. you pay per request, as well as the data storage.
92. So, if you can reduce the amount of requests
93. that are actually being sent to S3,
94. then, you can save some costs there.
95. - [Raf] Yes, absolutely.
96. And it's also faster
97. because it made the content, may be cached
98. in a location that is potentially closer
99. to where the client is.
100. - [Morgan] Right. - [Raf] Right.
101. - [Morgan] All right, so the next thing
102. I want to recommend for this,
103. would involve getting rid of Amazon API Gateway.
104. So, API Gateway was used in this architecture
105. so that we didn't expose Kinesis Data Firehose
106. directly to the internet, right?
107. We want to have some sort of authentication,
108. or some sort of barrier, between our Firehose
109. and the internet.
110. So, what we could do instead, is get rid of API Gateway,
111. and instead introduce Amazon Cognito.
112. And Amazon Cognito is an AWS service
113. that provides identity pools for authorization.
114. So, you can set up authorization workflows with Cognito,
115. and then you can integrate that directly
116. into the JavaScript.
117. So, this would require some changes
118. to that JavaScript client library.
119. So, our customer would have to make changes to this code,
120. and that would authenticate with Cognito.
121. And then, once you're authenticated,
122. you could then send the calls
123. from the JavaScript directly to Firehose.
124. So that means you wouldn't need to host and maintain
125. and pay for Amazon API Gateway
126. - [Morgan] anymore. - [Raf] Yes.
127. - [Morgan] If you're using Cognito.
128. - [Raf] And you can use the AWS JavaScript SDK
129. for that, right, on the HTML files?
130. - [Morgan] Right, right. - [Raf] Right.
131. - [Raf] Another suggestion that I would do
132. to enhance this architecture,
133. is making a CloudFormation template
134. out of this, of this entire solution,
135. in a way that you have the entire set
136. in a, in a state where you can reproduce
137. in another AWS account.
138. - [Morgan] All right, so why would
139. we want to have this architecture
140. in different accounts per customer?
141. - [Raf] Yeah. Well, spoiler alert, week 4,
142. we'll talk about multi-account strategies,
143. and we'll do a good job on explaining
144. the advantages of doing that.
145. But, you can have better governance,
146. you can have separated billing per account.
147. So, if the customer wants to charge a specific
148. restaurant chains for their whole infrastructure stack,
149. in addition to the value they add to them
150. with the system to update the menus and everything,
151. you can have that on separate stacks.
152. Also reduces manual mistakes
153. when operating the infrastructure.
154. So, it is a good idea to have that
155. into CloudFormation templates,
156. which is an infrastructure as code tool,
157. which is the first step that you would like to have,
158. to have this same environment on multiple AWS accounts.
159. - [Morgan] Right. Okay, that makes sense.
160. - [Raf] Yeah. What else? - [Morgan] All right, so -
161. - [Raf] What else? - [Morgan] So, the
162. - [Morgan] next thing that I'm thinking here,
163. is again with that client library,
164. so I'm going to keep picking
165. on that one there, but for that client-side library,
166. we have it currently sending requests
167. to API Gateway, and the same thing would apply
168. even if it was talking directly to Firehose.
169. You want to make sure
170. that you have appropriate retry logic embedded
171. into the client side.
172. And you want to also make sure that any retry logic
173. that you have has something called exponential backoff.
174. So, for example, let's say you're sending requests
175. from the JavaScript to API Gateway,
176. and API Gateway in, let's say, the Region
177. that you're using, is experiencing some intermittent issues.
178. You don't want to just continue to flood that service
179. just keep retrying and retrying and retrying.
180. Instead, what you do is you retry, and then you retry less
181. and less over time as it continues to fail.
182. So, implementing things like exponential backoff
183. is a good idea,
184. but having retry logic on the client side is important.
185. And luckily in the JavaScript SDK for AWS,
186. some of that retry logic is built into the SDK, as well.
187. - [Raf] Yes. Another optimization that I think about here,
188. is to change content in the bucket, to be more friendly
189. to Athena, for Athena to perform better.
190. You can partition the content,
191. you can transform the content to a columnnar data store
192. by converting files from CSV or JSON
193. to Parquet, Apache Parquet.
194. And then, you can have Athena doing faster queries,
195. faster and cheaper queries.
196. - [Morgan] All right, so I think that's all
197. the enhancements we have for this architecture.
198. So thank you, Raf, for designing in this week
199. to help our students better understand
200. how to use and mock up the data analytics services.
201. And I think this was a really strong week
202. to help everybody understand a little bit more
203. about how to use each individual AWS data analytics service,
204. and when to use them to match the customer requirement.
205. - [Raf] All right.
206. [End of transcript. Skip to the start.](https://courses.edx.org/xblock/block-v1:AWS+AWS-ARCH-1+3T2022+type@vertical+block@604b4669a7ba4de5986e2d3b11bed65a?exam_access=&format=Quiz&recheck_access=1&show_bookmark=0&show_title=0&view=student_view#transcript-start-35af63abca89411f9bb6bf9e51ca6b7f)

### Downloads and transcripts

1. Video[tart of transcript. Skip to the end.](https://courses.edx.org/xblock/block-v1:AWS+AWS-ARCH-1+3T2022+type@vertical+block@604b4669a7ba4de5986e2d3b11bed65a?exam_access=&format=Quiz&recheck_access=1&show_bookmark=0&show_title=0&view=student_view#transcript-end-35af63abca89411f9bb6bf9e51ca6b7f)
2. - [Morgan] All right. So, now that we've built
3. and presented this architecture to the customer,
4. - [Morgan] or me, (Raf laughs)
5. - [Morgan] we will both now be playing
6. the solutions architects,
7. and we'll discuss some of the things that can be enhanced,
8. or at least considered,
9. for the sake of making this architecture better.
10. Rethinking current architectures
11. should be a never-ending effort.
12. And it is a good practice to revisit solutions
13. from time to time because new services may be announced,
14. or conditions may have changed.
15. So, if we're ready to get going here,
16. if we look at the blocks,
17. we have these extra blocks that were not
18. on the table before.
19. So, can you explain a little bit about what those are for?
20. - [Raf] Correct. We have the full picture here, right?
21. So, we have both the part that is in scope,
22. which are the blocks at the bottom of the table,
23. the data analytics solution,
24. API Gateway, Kinesis Data Firehose,
25. Amazon S3, Amazon Athena, and Amazon QuickSight.
26. And on top of the table, we have the part
27. that is out of scope of the solution.
28. Meaning that, customers on their mobile phones,
29. they will scan the QR code.
30. Once they scan, they scan this QR code,
31. they will be served with an HTML webpage out of Amazon S3.
32. So, Amazon S3 is serving that webpage.
33. Customer also has a system where restaurant admins
34. can use that system to update the QR codes.
35. - [Morgan] And this system is currently being run
36. on EC2, I think, right?
37. - [Raf] Yes, yes. - [Morgan] Okay.
38. - [Raf] So, they upload the S3 bucket
39. with new items in the menu.
40. If they wants to change prices,
41. if they wants to change descriptions,
42. whatever modification they wants to do
43. to the restaurant menu.
44. So, it goes to S3.
45. And then, S3 starts serving that webpage
46. for customers who, customers who scan the QR code
47. - [Morgan] Makes sense. All right.
48. So, kind-of knowing that we have this full system here,
49. we're going to also suggest improvements
50. to the existing out of scope.
51. - [Raf] Yes. - [Morgan] So,
52. - [Morgan] with that in mind,
53. I want to make our first suggestion for this architecture.
54. And that would be to potentially look
55. at using a serverless service for this menu update system.
56. So, right now, this is running on EC2 somewhere,
57. and that server is running all of the time,
58. but these restaurants aren't updating their menus
59. all of the time, at all hours of the day.
60. You know, that's not something that happens constantly.
61. So instead, what we could maybe do, is try to change
62. this into a serverless architecture
63. for this part of the system.
64. And so, if we think about it, like, this one block
65. would probably become many blocks, right?
66. As we break it down into multiple serverless components.
67. So that is going to be my first recommendation
68. for this architecture.
69. - [Raf] Yes.
70. Another thing I have in mind here is,
71. now we have two S3 buckets,
72. one for the data analytics
73. that is in scope of what I suggested to the customer.
74. And another one that is out of scope,
75. which is the bucket that is serving the restaurant menus.
76. There is an AWS service called Amazon CloudFront,
77. which is a content delivery network,
78. where you can use Amazon S3 as origin.
79. So, Amazon CloudFront can cache contents before delivering
80. to customers.
81. And it has a cheaper data transfer out
82. when compared to Amazon S3.
83. - [Morgan] Som instead of the clients coming directly
84. to S3 to get the HTML, they would hit
85. - [Morgan] CloudFront first. - [Raf] They would hit
86. - [Raf] CloudFront first. - [Morgan] And so,
87. - [Morgan] if that data,
88. that HTML data, is already cached at CloudFront,
89. that request would never come to S3.
90. - [Raf] Exactly. - [Morgan] And so with S3,
91. you pay per request, as well as the data storage.
92. So, if you can reduce the amount of requests
93. that are actually being sent to S3,
94. then, you can save some costs there.
95. - [Raf] Yes, absolutely.
96. And it's also faster
97. because it made the content, may be cached
98. in a location that is potentially closer
99. to where the client is.
100. - [Morgan] Right. - [Raf] Right.
101. - [Morgan] All right, so the next thing
102. I want to recommend for this,
103. would involve getting rid of Amazon API Gateway.
104. So, API Gateway was used in this architecture
105. so that we didn't expose Kinesis Data Firehose
106. directly to the internet, right?
107. We want to have some sort of authentication,
108. or some sort of barrier, between our Firehose
109. and the internet.
110. So, what we could do instead, is get rid of API Gateway,
111. and instead introduce Amazon Cognito.
112. And Amazon Cognito is an AWS service
113. that provides identity pools for authorization.
114. So, you can set up authorization workflows with Cognito,
115. and then you can integrate that directly
116. into the JavaScript.
117. So, this would require some changes
118. to that JavaScript client library.
119. So, our customer would have to make changes to this code,
120. and that would authenticate with Cognito.
121. And then, once you're authenticated,
122. you could then send the calls
123. from the JavaScript directly to Firehose.
124. So that means you wouldn't need to host and maintain
125. and pay for Amazon API Gateway
126. - [Morgan] anymore. - [Raf] Yes.
127. - [Morgan] If you're using Cognito.
128. - [Raf] And you can use the AWS JavaScript SDK
129. for that, right, on the HTML files?
130. - [Morgan] Right, right. - [Raf] Right.
131. - [Raf] Another suggestion that I would do
132. to enhance this architecture,
133. is making a CloudFormation template
134. out of this, of this entire solution,
135. in a way that you have the entire set
136. in a, in a state where you can reproduce
137. in another AWS account.
138. - [Morgan] All right, so why would
139. we want to have this architecture
140. in different accounts per customer?
141. - [Raf] Yeah. Well, spoiler alert, week 4,
142. we'll talk about multi-account strategies,
143. and we'll do a good job on explaining
144. the advantages of doing that.
145. But, you can have better governance,
146. you can have separated billing per account.
147. So, if the customer wants to charge a specific
148. restaurant chains for their whole infrastructure stack,
149. in addition to the value they add to them
150. with the system to update the menus and everything,
151. you can have that on separate stacks.
152. Also reduces manual mistakes
153. when operating the infrastructure.
154. So, it is a good idea to have that
155. into CloudFormation templates,
156. which is an infrastructure as code tool,
157. which is the first step that you would like to have,
158. to have this same environment on multiple AWS accounts.
159. - [Morgan] Right. Okay, that makes sense.
160. - [Raf] Yeah. What else? - [Morgan] All right, so -
161. - [Raf] What else? - [Morgan] So, the
162. - [Morgan] next thing that I'm thinking here,
163. is again with that client library,
164. so I'm going to keep picking
165. on that one there, but for that client-side library,
166. we have it currently sending requests
167. to API Gateway, and the same thing would apply
168. even if it was talking directly to Firehose.
169. You want to make sure
170. that you have appropriate retry logic embedded
171. into the client side.
172. And you want to also make sure that any retry logic
173. that you have has something called exponential backoff.
174. So, for example, let's say you're sending requests
175. from the JavaScript to API Gateway,
176. and API Gateway in, let's say, the Region
177. that you're using, is experiencing some intermittent issues.
178. You don't want to just continue to flood that service
179. just keep retrying and retrying and retrying.
180. Instead, what you do is you retry, and then you retry less
181. and less over time as it continues to fail.
182. So, implementing things like exponential backoff
183. is a good idea,
184. but having retry logic on the client side is important.
185. And luckily in the JavaScript SDK for AWS,
186. some of that retry logic is built into the SDK, as well.
187. - [Raf] Yes. Another optimization that I think about here,
188. is to change content in the bucket, to be more friendly
189. to Athena, for Athena to perform better.
190. You can partition the content,
191. you can transform the content to a columnnar data store
192. by converting files from CSV or JSON
193. to Parquet, Apache Parquet.
194. And then, you can have Athena doing faster queries,
195. faster and cheaper queries.
196. - [Morgan] All right, so I think that's all
197. the enhancements we have for this architecture.
198. So thank you, Raf, for designing in this week
199. to help our students better understand
200. how to use and mock up the data analytics services.
201. And I think this was a really strong week
202. to help everybody understand a little bit more
203. about how to use each individual AWS data analytics service,
204. and when to use them to match the customer requirement.
205. - [Raf] All right.
206. [End of transcript. Skip to the start.](https://courses.edx.org/xblock/block-v1:AWS+AWS-ARCH-1+3T2022+type@vertical+block@604b4669a7ba4de5986e2d3b11bed65a?exam_access=&format=Quiz&recheck_access=1&show_bookmark=0&show_title=0&view=student_view#transcript-start-35af63abca89411f9bb6bf9e51ca6b7f)

### Downloads and transcripts

#### Video

### Reading 2.5: Architecture Optimizations for Week 2

In this reading, you will find further information about the topics that Morgan and Raf talked in the video where they both played solutions architects.

## Serverless on AWS

Morgan commented about changing the system that generates QR codes into something serverless, or a container. AWS offers technologies for running code, managing data, and integrating applications—all without managing servers. Serverless technologies feature automatic scaling, built-in high availability, and a pay-for-use billing model to increase agility and optimize costs. These technologies also reduce infrastructure management tasks, such as capacity provisioning and patching, so that you can focus on writing code that serves your customers. Serverless applications start with AWS Lambda, an event-driven compute service that natively integrates with over 200 AWS services and software as a service (SaaS) applications. For more information, see [Serverless on AWS](https://aws.amazon.com/serverless/).

## Amazon CloudFront

Then, Raf commented that Amazon CloudFront could be used with the Amazon Simple Storage Service (Amazon S3) bucket that is used to display the restaurant menus. Amazon CloudFront is a web service that speeds up the distribution of your static and dynamic web content (such as .html, .css, .js, and image files) to your users. CloudFront delivers your content through a worldwide network of data centers that are called edge locations. When a user requests content that you're serving with CloudFront, the request is routed to the edge location that provides the lowest latency (time delay), so that content is delivered with the best possible performance. In addition, Amazon CloudFront allows custom SSL certificates (issued by AWS Certificate Manager, or ACM), custom domain names, and further distributed denial of service (DDoS) protection that is powered by AWS WAF and AWS Shield.  
For more information, see [Amazon CloudFront](https://aws.amazon.com/cloudfront/).

## Amazon Cognito

Morgan commented about using Amazon Cognito instead of Amazon API Gateway to send data to Amazon Kinesis. This solution would require refactoring code in the JavaScript library that sends the clickstream data. However, it would also bypass API Gateway, and therefore reduce cost. You can use Amazon Cognito to add user sign-up, sign-in, and access control to your web applications and mobile apps. Amazon Cognito scales to millions of users and supports sign-in with social identity providers, such as Apple, Facebook, Google, and Amazon, and enterprise identity providers through SAML 2.0 and OpenID Connect.  
  
For more information, see [Amazon Cognito](https://aws.amazon.com/cognito/).  
  
With Amazon Cognito, you can have a mobile JavaScript library that assumes a role with a web identity. That role must have a strict policy associated with it—for example, the policy could allow only the PutRecord API operation on a specific Kinesis data stream.  
  
For more information, see [AssumeRoleWithWebIdentity](https://docs.aws.amazon.com/STS/latest/APIReference/API_AssumeRoleWithWebIdentity.html) in the *AWS Security Token Service API Reference*.

## Amazon CloudFormation and IaC

Later in the conversation, Raf talked about transforming the entire environment into a template that could be further replicated for each of the customer’s clients. This can be achieved by treating infrastructure as code (IaC) with a managed AWS service called AWS CloudFormation. CloudFormation helps you model and set up your AWS resources so that you can spend less time managing those resources, and more time focusing on your applications that run in AWS. You create a template that describes all the AWS resources that you want—for example, Amazon Elastic Compute Cloud (Amazon EC2) instances or Amazon Relational Database Service (Amazon RDS) DB instances—and CloudFormation provisions and configures those resources for you. You don't need to individually create and configure AWS resources and figure out which resource is dependent on another resource.  
  
For example, this YAML template snippet creates an S3 bucket with the following configurations:

* Static website hosting that uses index.html as the index document and error.html as the error document
* Bucket policy that allows everyone to GET objects (otherwise, the bucket is private)
* Deletion policy that asks CloudFormation to retain the bucket if the CloudFormation stack is deleted (this is a good practice when you use data services, such as buckets and databases)

The snippet also outputs the https:// URL that’s used by static website hosting, which you can send to someone else, or copy and paste into a web browser after the bucket is created.

AWSTemplateFormatVersion: 2010-09-09  
Resources:  
 S3Bucket:  
 Type: 'AWS::S3::Bucket'  
 Properties:  
 AccessControl: PublicRead  
 WebsiteConfiguration:  
 IndexDocument: index.html  
 ErrorDocument: error.html  
 DeletionPolicy: Retain  
 BucketPolicy:  
 Type: 'AWS::S3::BucketPolicy'  
 Properties:  
 PolicyDocument:  
 Id: MyPolicy  
 Version: 2012-10-17  
 Statement:  
 - Sid: PublicReadForGetBucketObjects  
 Effect: Allow  
 Principal: '\*'  
 Action: 's3:GetObject'  
 Resource: !Join   
 - ''  
 - - 'arn:aws:s3:::'  
 - !Ref S3Bucket  
 - /\*  
 Bucket: !Ref S3Bucket  
Outputs:  
 WebsiteURL:  
 Value: !GetAtt   
 - S3Bucket  
 - WebsiteURL  
 Description: URL for website hosted on S3  
 S3BucketSecureURL:  
 Value: !Join   
 - ''  
 - - 'https://'  
 - !GetAtt   
 - S3Bucket  
 - DomainName  
 Description: Name of S3 bucket to hold website content

For more examples of how to use CloudFormation templates, see [Template snippets](https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/CHAP_TemplateQuickRef.html) in the *AWS CloudFormation User Guide*.   
  
Creating infrastructure templates is important because you can replicate customer workloads across multiple AWS accounts or Regions. You will learn more about how relevant and important this is in week 4 of this course.  
  
For more information, see [AWS CloudFormation](https://aws.amazon.com/cloudformation/).

## Error retries and exponential backoff in AWS

Morgan also talked about error retries and exponential backoffs. Although this topic seems complicated, here are some best practices and observations about error retries and exponential backoffs.  
  
Numerous components on a network—such as DNS servers, switches, load balancers, and others—can generate errors anywhere in the life of a given request. The usual technique for dealing with these error responses in a networked environment is to implement retries in the client application. This technique increases the reliability of the application and reduces operational costs for the developer.  
  
If you are using one of the AWS SDKs, they already practice exponential backoffs by default. This default behavior would be one benefit of using Amazon Cognito on the client side to send data to Kinesis, instead of using a custom routine that sends data to API Gateway. You should implement a maximum delay interval and a maximum number of retries. The maximum delay interval and maximum number of retries are not necessarily fixed values. These values should be set based on the operation that’s being performed and other local factors, such as network latency.  
  
For more information, see [Error retries and exponential backoff in AWS](https://docs.aws.amazon.com/general/latest/gr/api-retries.html) in the *AWS General Reference*.

## Data optimizations for Amazon Athena

Last, but not least—in the conversation, Raf talked about data optimization for Amazon Athena. This topic is broad, and it even has dedicated AWS Blog posts about the [Top 10 Performance Tuning Tips for Amazon Athena](https://aws.amazon.com/blogs/big-data/top-10-performance-tuning-tips-for-amazon-athena/).  
  
The top three data best practices for Athena are compressing, partitioning, and converting your data into columnar formats.  
  
In brief, Athena is a service that charges per amount of data scanned. By data scanned, AWS means the amount of data that Athena fetches to perform a query. As you know, data compression can reduce file sizes, which means that Athena scans less data per query. Because Athena scans less data, the queries also run faster. You can save from 30 percent to 90 percent on your per-query costs, and get better performance by applying data best practices.  
  
Data partitioning is another technique to restrict the amount of data that’s scanned by each query. You can partition your data by any key. A common practice is to partition the data based on time, which often leads to a multi-level partitioning scheme. For example, a customer who has data coming in every hour might decide to partition by year, month, date, and hour. Say that you have a 5-TB file containing monthly information in it, and this file pertains to a whole year. If you constantly query things by adding statements such as WHERE Month = "February", the system will scan the entire 5-TB file for each query. The system will probably ignore data that doesn’t belong to the month of February. This process is slower and more expensive than partitioning the data so that you have files that belong to each individual month. If you partitioned the data by month, a query that contains a clause like WHERE Month = "February" would only scan data that pertains to the month of February, thus making the query faster and cheaper.  
  
For more information, see [Partitioning data in Athena](https://docs.aws.amazon.com/athena/latest/ug/partitions.html).  
  
Another strategy is to use an AWS service to convert bucket data into columnar data formats. Apache Parquet and ORC are columnar storage formats that are optimized for the fast retrieval of data and they are also used in AWS analytical applications. Columnar storage formats have the following characteristics that make them suitable for use with Athena:

* Compression by column, with the compression algorithm selected for the column data type to save storage space in Amazon S3, and reduce disk space and I/O during query processing.
* Predicate pushdown in Parquet and ORC enables Athena queries to fetch only the blocks that it needs, thus improving query performance. When an Athena query obtains specific column values from your data, it uses statistics from data block predicates, such as max or min values, to determine whether to read or skip the block.
* Splitting data in Parquet and ORC enables Athena to split the reading of data to multiple readers and increase parallelism when it processes queries.

For more information, see [Converting to columnar formats](https://docs.aws.amazon.com/athena/latest/ug/convert-to-columnar.html).

1. [ipt. Skip to the end.](https://courses.edx.org/xblock/block-v1:AWS+AWS-ARCH-1+3T2022+type@vertical+block@c3ba1732f4cb4b5d98914e221b3d37d5?exam_access=&format=Quiz&recheck_access=1&show_bookmark=0&show_title=0&view=student_view#transcript-end-cd2a003f8bab4ac5a0045fa4f83370ee)
2. - [Morgan] Hello, welcome, everybody, to this video.
3. In this video, we are going to do a walkthrough
4. of how to build a proof of concept
5. for this week's architecture.
6. Again, this is an optional step
7. if you would like to create this
8. in your own AWS account or follow along,
9. but we are going to include some instructions
10. if you do want to do that.
11. So just as a quick reminder,
12. we're going to be building a proof of concept,
13. so this isn't supposed to mimic exact real life conditions.
14. Instead, this is supposed to show our customer
15. how to build the solution that we designed,
16. and show them how these services integrate,
17. and show them a proof of concept
18. that this does work the way that we said it does.
19. So what we're going to do is build out the API Gateway,
20. that's the endpoint where the clickstream front end
21. is going to be sending the data.
22. We're going to be sending that data
23. to Amazon Kinesis Data Firehose,
24. which will then invoke a Lambda function
25. that does some really simple formatting.
26. And then, Amazon Kinesis Data Firehose
27. will send that data to Amazon S3 for storage.
28. Once the data is stored in S3,
29. we'll use Amazon Athena to query that data,
30. and then we'll use Amazon QuickSight to visualize that data.
31. So to get started, we need to create some IAM policies
32. and roles in order for this to work.
33. So I already did that.
34. Let's go ahead and navigate to IAM
35. and I will show you the roles
36. that I created for this demo.
37. All right, so from here I'm going to select Roles.
38. And then, we have this role here, API-Firehose,
39. if I click on this, and then scroll down,
40. we can see the different policies that exist,
41. I have two policies attached to this,
42. I have a customer-managed policy and an AWS-managed policy.
43. This first one is called API-Firehose
44. and this allows the action, Firehose PutRecord.
45. So this is the API call for Kinesis Data Firehose
46. of how you can send information
47. into your Kinesis Data Firehose.
48. So that's the API call, PutRecord.
49. And then, we're allowing it against star,
50. which is all Kinesis Data Firehose resources.
51. But you could also put the ARN
52. of the specific Data Firehose once you have it created.
53. And then, if I scroll down,
54. we also have this Amazon API Gateway push
55. to CloudWatch Logs.
56. This is allowing API Gateway to send logs
57. to Amazon CloudWatch Logs.
58. So this role is going to be used as the execution role
59. for our API Gateway,
60. which is going to integrate with Kinesis Data Firehose.
61. All right, so let's go ahead
62. and, first, let's create our S3 bucket.
63. So from here, I'm going to type in S3
64. and navigate to the S3 dashboard.
65. And from here, I'm going to select Create Bucket.
66. And now we need to give this bucket a name.
67. Bucket names need to be globally unique across all accounts.
68. So I'm going to go ahead and give this a name,
69. testbucket-morgan-2022,
70. and then I'll just go 1234.
71. Hopefully, that's unique enough.
72. And then, I will scroll down,
73. and we'll accept all the defaults and click Create Bucket.
74. Now this bucket by default's going to be private,
75. so that's the default configuration.
76. We will later modify the bucket access policy
77. for this bucket to allow Kinesis Data Firehose
78. to put objects in this bucket.
79. But for now, we're going to open up the bucket
80. and click on the Properties tab.
81. And then, I'm going to copy the ARN for this bucket,
82. because in order to integrate these services together,
83. you have to kind of point one to another
84. and manage permissions and things like that.
85. So I'm going to copy this ARN,
86. and then I have a text editor open on a different screen,
87. and I'm going to be copying and pasting as needed
88. for these tasks to create this POC.
89. So we have our bucket now
90. and we've copied the ARN for future use.
91. So the next thing that I want to do
92. is actually create the Lambda function
93. that Kinesis Data Firehose is going to be using.
94. So to do that, I'm going to navigate to the Lambda console.
95. And then, from here, I'm going to click Create Function.
96. And then, we're going to actually use
97. a blueprint for this one.
98. So it's very common to need a Lambda function
99. to work with Kinesis Data Firehose to transform your data
100. before it gets to S3.
101. So what we're going to do
102. is create one of these Lambda functions using a blueprint
103. and I'm going to type in Kinesis to filter them.
104. And then, we're looking for the one titled,
105. "Process records sent to a Kinesis Firehose stream",
106. specifically in Python 3.8.
107. You'll notice that there is one
108. with the same name for Node.js.
109. So you want to, I think it's, let's see, where is it?
110. Oh, here.
111. Yeah, so Process Records Sent to a Kinesis Firehose Stream,
112. you want to make sure you're picking the right one.
113. So choose the one that's Python 3.8
114. because that's what we have the code
115. written in for this step.
116. So make sure you're selecting the right one
117. if you choose to do this.
118. So, essentially, what this function is going to be doing
119. is modifying the data that comes in
120. through Kinesis Data Streams before it gets to S3.
121. And the reason for that is because with Amazon Athena,
122. it needs every piece of data to be on a new line
123. in order to register it as like
124. a unique item that it's querying.
125. So what we need to do is make sure
126. that each individual event
127. that comes through our clickstream process
128. gets put onto a new line.
129. And Kinesis Data Firehose right now,
130. as of the time of me creating this demo,
131. puts everything on one line.
132. So we want to make sure that we are allowing
133. those new lines to be put into place,
134. so that's what this is for.
135. So then we'll go ahead and click Configure.
136. And now we can give this function a name,
137. and I'm going to call this transform-data.
138. And then, we can go ahead and scroll down here
139. and we're going to leave this stubbed out code here for now.
140. I'm just going to go ahead and click Create Function
141. and we will modify this code in just a second.
142. I have the instructions open on another screen as well,
143. where I'm going to be copying and pasting
144. in different information,
145. like the code for this Lambda function.
146. All right, so I'm going to go ahead and scroll down.
147. Our Lambda function's been created.
148. And this is the Code source panel,
149. so this is where you can actually edit the code.
150. I'm going to highlight this code and blank it out,
151. and paste in the code from the instructions.
152. And so you'll notice that this is written in Python.
153. We have some import statements at the top
154. that we're using in the code below.
155. Boto3 is the AWS SDK for Python.
156. And then, we are setting up a variable called output,
157. and then we have our Lambda handler.
158. This is where the execution of this code is going to begin,
159. so this is like the entry point,
160. and this accepts an event in a context
161. which has information about the event.
162. So that would include things like the payload of the event,
163. and then context, this is about the execution.
164. And then, we have some code that's looping
165. over all of the records in the event.
166. And then, what we're doing is we are decoding that data.
167. We are adding a new line to the end of that data.
168. And then, we are re-encoding it.
169. And then, we are going to be appending all of that data,
170. formatting it, and then we're going to be appending
171. all of that data to itself,
172. kind of keeping a running tally here with each new event
173. having a new line at the end of it.
174. And then, we're going to be returning that data as records,
175. well, an output as JSON here.
176. All right, so now we can go ahead and click Deploy.
177. And then, the next thing I want to do
178. is click on the Configuration tab.
179. And I'm going to edit the general configuration,
180. and I want to change the timeout to be 10 seconds,
181. and this will just minimize
182. the potential of this function timing out
183. if you had like a large payload.
184. So if you had a ton of events that you were looping over,
185. then you may time out depending on, you know,
186. how many you have and the type of processing you're doing.
187. In our case, it probably won't time out at three seconds,
188. it'd be very unlikely,
189. but I'm going to go ahead and up that to 10 seconds anyways.
190. And the reason for that,
191. we most likely wouldn't time out is because this is a POC,
192. so we're not going to be sending in a ton of data.
193. We're just going to be sending in enough data
194. to prove that this solution does work.
195. So I'm going to go ahead and click Save.
196. And now I want to scroll down here
197. and I want to then grab the ARN.
198. So, sorry, I'm going to scroll up.
199. And then, I want to copy this function ARN,
200. and I'm going to paste that into my text document
201. off on another screen.
202. And again, this is going to be used for future steps
203. in building out this POC.
204. All right, so now we have our Lambda function created,
205. but this doesn't have any triggers,
206. it's not being used by anything yet.
207. So what we want to do next
208. is create the Kinesis Data Firehose
209. which will actually use this Lambda function.
210. So I'm going to go ahead and navigate to Kinesis
211. and click on Kinesis Service.
212. And then, we can click on, under the Get Started pane,
213. you want to select Kinesis Data Firehose,
214. and then click Create Delivery Stream.
215. And then, from here, we want to choose a source
216. and a destination.
217. Our source is going to be a Direct PUT
218. because API Gateway is going to be directly
219. putting events using that PutRecord API call
220. directly to Kinesis Data Firehose
221. using an AWS service integration.
222. And then, the destination is going to be S3.
223. And then, you can give this delivery stream
224. a name if you like.
225. I'm going to go ahead and just leave the generated name.
226. And then, I'm going to change this section here,
227. where it says Transform and Convert Records,
228. we do want to turn on data transformation,
229. so I'm going to select Enabled here.
230. And then, I'm going to paste in that ARN
231. for our Lambda function.
232. And then, we can choose an alias, which is going to be $LATEST.
233. And then, I'm going to scroll down,
234. and then we need to change this section here,
235. the Destination settings,
236. to where we going to be sending this data, downstream,
237. S3, that bucket we created.
238. So I'm going to go ahead and choose this bucket.
239. And then, you can also add a prefix there
240. if you would like to.
241. I'm going to go ahead and just let it put it directly
242. in the main folder of that bucket.
243. So I'm going to scroll down,
244. and then click Create delivery stream.
245. All right, so now our delivery stream is being created.
246. So you can see it's in this status, Creating.
247. I'm going to go ahead and copy the ARN for this
248. for now while this is creating.
249. And then, if we come back to delivery streams,
250. this is where you would kind of come back
251. and check to see when this is done creating.
252. So I'm going to take a pause here,
253. and then we'll grab some more information
254. about this delivery stream once it's created,
255. and then we'll continue on creating this POC.
256. This step can take some time.
257. So if it's in this creating phase,
258. just keep coming back to this page
259. and refreshing the screen,
260. and it will eventually say that it's ready,
261. it's been created,
262. but it does take a couple of minutes.
263. So be patient, and I will come back when this is done.
264. All right, so I'm back and we can now see
265. that this Kinesis Data Firehose delivery stream is active.
266. And the next thing I want to do
267. is open back up the stream,
268. and then I want to scroll down
269. and select the Configuration tab.
270. And then, I'm going to scroll down to
271. the Permissions section.
272. And then, note how this has an IAM role here.
273. So whenever we created
274. this Kinesis Data Firehose delivery stream,
275. it created an IAM role for us,
276. and we are going to need the ARN of this role.
277. So I'm going to go ahead and click this,
278. it will open up the IAM console for me
279. to that specific role.
280. And then, what I want to do is copy the ARN for that role,
281. and then I'm going to paste that
282. into my text document on the side.
283. And we are going to use this ARN
284. to modify the bucket access policy for our S3 bucket
285. that will allow Kinesis Data Firehose,
286. this specific delivery stream using this specific IAM role,
287. to put data into that bucket.
288. So with that being said,
289. let's go ahead and navigate to the S3 console and select S3.
290. And then, we want to select the bucket that we created,
291. and then select the Permissions tab,
292. and then scroll down to Bucket Policy.
293. Notice how it's currently empty,
294. we're going to add a bucket policy there, so click Edit.
295. And from here, you're going to paste in the JSON
296. from the instructions.
297. You also could use the policy generator
298. by going through and adding statements here
299. and things like that.
300. You can also look at policy examples,
301. use the policy generator, which is right here.
302. You could click that, open it in a new tab.
303. But notice how for the one that I just pasted in,
304. let's go ahead and read it.
305. We have a statement, and for this statement,
306. first we have one that says, I want to allow.
307. What principle? The AWS principle.
308. We need to paste in the ARN
309. for that Kinesis Data Firehose role,
310. so I'm going to go ahead and do that.
311. Notice how I'm getting rid of both of the carets there.
312. You want to make sure that yours looks like this,
313. where you have the complete ARN of that role
314. without the carets on either side,
315. and you will need to make sure
316. you have both double quotes as well,
317. because this is in JSON.
318. And then, for the action,
319. we have the different actions that we will need
320. in order to upload objects to this bucket.
321. And then, finally, we need to modify this last section,
322. which is against what resources can I make these actions?
323. Notice how I just copied the bucket ARN from up top here.
324. They make it very easy to do that.
325. So I'm going to paste that in here,
326. and then I'm going to paste it in again on the next line.
327. So I'm saying against what resources
328. can this role complete these actions?
329. This bucket and everything inside of this bucket, right?
330. Okay, so now we'll go ahead and click Save Changes.
331. All right, so now we have our bucket policy
332. has been edited successfully.
333. So the next step that we need to do is create the REST API,
334. which is going to be the endpoint
335. for our front-end clickstream client.
336. So I'm going to go ahead and navigate to API Gateway,
337. select API Gateway.
338. And then, I want to choose what API type I'm going to build.
339. I'm going to be using an AWS service integration
340. between API Gateway and Kinesis Data Firehose.
341. And as a part of that,
342. I want to be able to transform the payload
343. of that request using mappings,
344. so I need to choose a REST API to do that.
345. So I'm going to go ahead and select
346. REST API and click Build.
347. And then, I'm going to click Okay.
348. And then, under Create New API, I'm going to select New API.
349. And then, we can give this a name,
350. which I'm going to say clickstream-ingest-api,
351. or sorry, -poc.
352. And then, we can leave our endpoint type as Regional.
353. And then, you just click Create API.
354. All right, now we want to define some methods
355. and our resources.
356. So I'm going to click this Actions,
357. and then I'm going to click Create Resource.
358. And then, I want to name this first resource,
359. I'm just going to say is poc.
360. And then, I'll click Create Resource.
361. This is creating like where you'd send the endpoint.
362. So this API Gateway is going to give you a URL.
363. And then, if I did like that URL/poc,
364. and then make a post to it,
365. that's what I want our front end to do.
366. So I'm essentially creating a location
367. that you can send this data to.
368. And then, I need to create the method for this POC resource,
369. which is going to be, again, a POST,
370. because I expect our front-end client to make POSTs to this.
371. All right, next, we need to choose
372. the integration point for this method.
373. I'm going to use AWS Service for this,
374. because, again, we are using
375. an AWS service integration to do this.
376. I'm noting I'm in North Virginia,
377. I've been building everything in North Virginia.
378. So for the AWS region, I'm going to choose us-east-1,
379. which is the alias for the North Virginia region.
380. And then, for the AWS service,
381. I need to select Firehose here, which is right here.
382. Notice how there is also a selection for Kinesis.
383. You need to select Firehose,
384. because we're using Kinesis Data Firehose.
385. All right, so now for the AWS subdomain,
386. we're going to go ahead and leave that empty.
387. Then for the HTTP method, we're going to select POST,
388. because API Gateway is going to be making a POST
389. to Firehose once it accepts the request from the front end.
390. And then, we're going to leave this
391. Use Action Name bullet selected.
392. And then, under the Action Type,
393. we're going to put PutRecord.
394. This is the API call that's going to be made
395. to Kinesis Data Firehose.
396. So this is actually a Kinesis Data Firehose API.
397. And then, finally, the execution role,
398. this is that role that we created
399. in one of the first steps in IAM.
400. So we'll paste the ARN for that role
401. right here where it says Execution role.
402. And then, what we're going to do
403. is leave this content handling as Passthrough.
404. And then, we will also keep this check mark here
405. and just click Save.
406. All right, so now what we need to do
407. is modify this integration request further
408. so that this API can transform the incoming request
409. from the clickstream client into the shape
410. that the Kinesis Data Firehose needs.
411. So I'm going to select Integration Request.
412. And then, what I want to do is I want to scroll down here
413. and I want to select Mapping Templates,
414. and then scroll down.
415. And then what I want to do is I want to select,
416. When there are no templates defined.
417. And then, I want to add a mapping template,
418. under Content Type, I'm typing in application/json.
419. And then, checking this checkbox to get that to save.
420. Then, I'm going to click on this and scroll down.
421. And now what we need to do
422. is paste in this VTL from our,
423. which I forgot a curly bracket there,
424. this is from our instructions.
425. So VTL, this is going to be the code
426. that's going to be actually doing the transformation
427. on the payload.
428. So we have to shape the incoming data
429. to be matching the parameters
430. that Kinesis Data Firehose is expecting.
431. So Kinesis Data Firehose PutRecord
432. is looking for parameters,
433. the DeliveryStreamName,
434. and then the records of the actual data.
435. So I'm going to copy the name
436. of the Kinesis Data Firehose delivery stream
437. and go ahead and paste that right here.
438. And then, I have the record,
439. so if we just read through this,
440. there's nothing we need to change.
441. But, essentially, what we have here
442. is we have some VTL that is formatting the information
443. that's coming off of the input, which is the payload.
444. So we are transforming that to be recognized as JSON.
445. We're replacing some characters
446. and we're Base64 encoding it.
447. So I'm going to go ahead now and click Save.
448. So you need to make sure you click Save here,
449. it's a very important step.
450. All right, so now what we need to do
451. is scroll back up,
452. and now I'm going to click back
453. to the Method Execution page.
454. And now I want to click Test.
455. And so now from here,
456. what we're going to do is we're going to test this API out.
457. We're going to be simulating events
458. as if they were coming from our clickstream front end.
459. So I'm going to paste in some information
460. that's from the instructions.
461. This is just a really quick, mocked-up structure of JSON
462. of what it could look like coming from the clickstream.
463. So again, in the real world,
464. you would likely want to have a library
465. that sends a bunch of data
466. as it would would in the real world
467. if you're building this out to like QA it,
468. or something like that.
469. But for a proof of concept, this is perfectly fine.
470. So it has the element\_clicked is whatever entree they're on,
471. the time\_spent in seconds,
472. the restaurant\_name for the menu,
473. so like what menu are they on?
474. Because remember this is the software house,
475. they have multiple customers,
476. and then the date it's created at.
477. So I'm going to go ahead and click Test.
478. All right, and then, over here on the right-hand side,
479. we should be able to look through these logs
480. and see the response that we got back
481. from Kinesis Data Firehose.
482. So we can see the request is being sent
483. to Kinesis Data Firehose,
484. and then we can see that we received
485. the response status 200.
486. If you see status 400 here or 500, something like that,
487. 400, your stream might not be created yet.
488. So you might need to go and look and see
489. and make sure it's in that active state.
490. If it's not, wait a little bit, try again.
491. If this is 500,
492. then you might want to check the VTL that you pasted in,
493. make sure that's all structured correctly,
494. make sure that your integration request is created properly.
495. And then, if you have some sort of permissions error,
496. check your role that API Gateway is using
497. and make sure that that has the permissions
498. to post to your Kinesis Data Firehose.
499. But for us, right now,
500. it looks like this did work, so that's great.
501. I'm not done yet though.
502. What I'm going to do is I'm going to generate
503. a couple of different entrees to kind of simulate
504. what some data would look like
505. if somebody was clicking through the menu.
506. So I'm just going to do like,
507. let's say seven or eight different tests here.
508. And what Kinesis Data Firehose is going to do
509. is it's going to collect all this information
510. and it's going to aggregate it,
511. and then it's going to throw it into S3.
512. So whenever Kinesis Data Firehose does this aggregation,
513. there is a bit of a lag for time.
514. It can be like 60 seconds or something.
515. So we can verify that this is working
516. by actually querying the data in S3
517. or you can go to S3 directly.
518. So first, let's go to S3 directly
519. to see if that data's there.
520. It's not going to be there yet
521. because, again, this takes time
522. to actually get posted to S3.
523. But I can open up this bucket,
524. and then I'm going to pause here and we'll come back.
525. You can just kind of refresh this,
526. maybe give it a couple of minutes.
527. Refresh it, you should see some information pop up here.
528. So I'm going to pause it,
529. and then I'll come back when it's done.
530. (End of Part 1.)
531. [End of transcript. Skip to the start.](https://courses.edx.org/xblock/block-v1:AWS+AWS-ARCH-1+3T2022+type@vertical+block@c3ba1732f4cb4b5d98914e221b3d37d5?exam_access=&format=Quiz&recheck_access=1&show_bookmark=0&show_title=0&view=student_view#transcript-start-cd2a003f8bab4ac5a0045fa4f83370ee)

**import base64**

**import json**

**print('Loading function')**

**def lambda\_handler(event, context):**

**#print("Received event: " + json.dumps(event, indent=2))**

**for record in event['Records']:**

**# Kinesis data is base64 encoded so decode here**

**payload = base64.b64decode(record['kinesis']['data']).decode('utf-8')**

**print("Decoded payload: " + payload)**

**return 'Successfully processed {} records.'.format(len(event['Records']))**

**import base64**

**import json**

**print('Loading function')**

**def lambda\_handler(event, context):**

**#print("Received event: " + json.dumps(event, indent=2))**

**for record in event['Records']:**

**# Kinesis data is base64 encoded so decode here**

**payload = base64.b64decode(record['kinesis']['data']).decode('utf-8')**

**print("Decoded payload: " + payload)**

**return 'Successfully processed {} records.'.format(len(event['Records']))**

**import json**

**import boto3**

**import base64**

**output = []**

**def lambda\_handler(event, context):**

**for record in event ['records']:**

**payload= base64.b64decode(record['data']).decode('utf-8')**

**row\_w\_newline payload + "\n"**

**row\_w\_newline = base64.b64encode(row\_w\_newline.encode('utf-8'))**

**output\_record = {**

**}**

**'recordId': record['recordId'], 'result': 'Ok',**

**'data': row\_w\_newline**

**output.append(output\_record)**

**return {'records': output}**

**I**

[**https://www.udemy.com/course/aws-solutions-architect-professional/?utm\_source=adwords&utm\_medium=udemyads&utm\_campaign=DSA\_Catchall\_la.EN\_cc.INDIA&campaigntype=Search&portfolio=India&language=EN&product=Course&test=&audience=DSA&topic=&priority=&utm\_content=deal4584&utm\_term=\_.\_ag\_82569850245\_.\_ad\_533220805577\_.\_kw\_\_.\_de\_c\_.\_dm\_\_.\_pl\_\_.\_ti\_dsa-437115340933\_.\_li\_9185266\_.\_pd\_\_.\_&matchtype=&gad\_source=1&gclid=EAIaIQobChMI-5GN7oDWhgMVHS2DAx1yjw2cEAAYAiAAEgI61vD\_BwE&couponCode=IND21PM**](https://www.udemy.com/course/aws-solutions-architect-professional/?utm_source=adwords&utm_medium=udemyads&utm_campaign=DSA_Catchall_la.EN_cc.INDIA&campaigntype=Search&portfolio=India&language=EN&product=Course&test=&audience=DSA&topic=&priority=&utm_content=deal4584&utm_term=_._ag_82569850245_._ad_533220805577_._kw__._de_c_._dm__._pl__._ti_dsa-437115340933_._li_9185266_._pd__._&matchtype=&gad_source=1&gclid=EAIaIQobChMI-5GN7oDWhgMVHS2DAx1yjw2cEAAYAiAAEgI61vD_BwE&couponCode=IND21PM)

[**https://www.examtopics.com/discussions/amazon/view/117663-exam-aws-certified-solutions-architect-associate-saa-c03/**](https://www.examtopics.com/discussions/amazon/view/117663-exam-aws-certified-solutions-architect-associate-saa-c03/)

[**https://quizlet.com/513898750/aws-quiz-4-flash-cards/**](https://quizlet.com/513898750/aws-quiz-4-flash-cards/)

**arn:aws:lambda:ap-south-1:944346522085:function:kinesistransformdata**

**arn:aws:firehose:ap-south-1:944346522085:deliverystream/PUT-S3-m9Qr8**

[KinesisFirehoseServiceRole-PUT-S3-m9Qr8-ap-south-1-1718196392387](https://ap-south-1.console.aws.amazon.com/go/view?arn=arn%3Aaws%3Aiam%3A%3A944346522085%3Arole%2Fservice-role%2FKinesisFirehoseServiceRole-PUT-S3-m9Qr8-ap-south-1-1718196392387&source=firehose)

**arn:aws:iam::944346522085:role/service-role/KinesisFirehoseServiceRole-PUT-S3-m9Qr8-ap-south-1-1718196392387**

[**https://docs.aws.amazon.com/lambda/latest/dg/best-practices.html**](https://docs.aws.amazon.com/lambda/latest/dg/best-practices.html)

[**https://www.dhokiyas.com/**](https://www.dhokiyas.com/)

##### DHOKIYA'S RESEARCH ANALYST (OPC) PRIVATE LIMITED

|  |  |
| --- | --- |
| **CIN** | [**U66190GJ2024OPC148079**](https://www.falconebiz.com/company/DHOKIYA-S-RESEARCH-ANALYST-OPC-PRIVATE-LIMITED-U66190GJ2024OPC148079) |
| Company Status | **Active** |
| Registration Number | 148079 |
| Date of Incorporation | 29th January, 2024 |
| RoC | **RoC-Ahmedabad** |
| Company Age | **4 months & 15 days** |

### SEBI Registered Research Analyst & Investment Advisor

#### D-Street investment Advisor Pvt Ltd

Jan 2014 - Present 10 years 6 months

Surat, Gujarat, India

Chetan Dhokiya is Self Employed Person.  
he is SEBI Registered Research Analyst & Investment Advisor.

[**https://compliancemanager.co.in/blogs/**](https://compliancemanager.co.in/blogs/)

Athena now supports typeahead code suggestions to speed up SQL query development

Typeahead suggestions are turned on by default. You can change this setting in query editor preferences.

<https://learning.edx.org/course/course-v1:AWS+AWS-AWS-OTP-AWSD16+1T2023/block-v1:AWS+AWS-AWS-OTP-AWSD16+1T2023+type@sequential+block@7e1bb4fe1be1474fb4f1450c75ee8cab/block-v1:AWS+AWS-AWS-OTP-AWSD16+1T2023+type@vertical+block@6e97b69a0b5f49caadf0e4bf7681b4b1>

## [Amazon Athena](https://ap-south-1.console.aws.amazon.com/athena/home?region=ap-south-1#/landing-page)

[**https://us-east-1.quicksight.aws.amazon.com/sn/data-sets/new**](https://us-east-1.quicksight.aws.amazon.com/sn/data-sets/new)

**arn:aws:iam::944346522085:role/djsdynamo2**

**Gateway**

**arn:aws:iam::944346522085:role/dynamoapigateway**

**sqs**

[**https://sqs.ap-south-1.amazonaws.com/944346522085/djsdynamo3**](https://sqs.ap-south-1.amazonaws.com/944346522085/djsdynamo3)

[**https://aws.amazon.com/about-aws/whats-new/2023/10/amazon-sns-in-place-message-archiving-replay-fifo-topics/**](https://aws.amazon.com/about-aws/whats-new/2023/10/amazon-sns-in-place-message-archiving-replay-fifo-topics/)

[**https://learning.edx.org/course/course-v1:AWS+AWS-ARCH-1+3T2022/block-v1:AWS+AWS-ARCH-1+3T2022+type@sequential+block@97b4a06a6c8a42f09a37623385de1efd/block-v1:AWS+AWS-ARCH-1+3T2022+type@vertical+block@2b92367e7e894c9fa3ea27fbd55f2521**](https://learning.edx.org/course/course-v1:AWS+AWS-ARCH-1+3T2022/block-v1:AWS+AWS-ARCH-1+3T2022+type@sequential+block@97b4a06a6c8a42f09a37623385de1efd/block-v1:AWS+AWS-ARCH-1+3T2022+type@vertical+block@2b92367e7e894c9fa3ea27fbd55f2521)

**{**

**"Records": [**

**{**

**{**

**"receiptHandle":**

**"messageId": "19dd0b57-b21e-4ac1-bd88-01bbb068cb78", "MessageReceiptHandle", "body": "incoming order information will go here", "attributes":{**

**},**

**"ApproximateReceiveCount": "1", "SentTimestamp": "1523232000000", "SenderId": "123456789012",**

**"ApproximateFirstReceiveTimestamp": "1523232000001"**

**"messageAttributes": {},**

**"md50fBody": "{{{md5\_of\_body}}}",**

**"eventSource": "aws: sqs",**

**"eventSourceARN": "arn: aws: sqs:us-east-1:123456789012: M**

**"awsRegion": "us-east-1"**

**}**

**}**

**import boto3, uuid**

**client**

**table**

**boto3.resource('dynamodb') client.Table("orders")**

**def lambda\_handler(event, context):**

**for record in event['Records']: print("test")**

**payload record["body"]**

**print(str(payload))**

**table.put\_item(Item= {'orderID': str(uuid.uuid4()),'order': payload})**

**import boto3**

**import uuid**

**# Create a DynamoDB resource**

**dynamodb = boto3.resource('dynamodb')**

**table = dynamodb.Table("orders")**

**def lambda\_handler(event, context):**

**for record in event['Records']:**

**print("test")**

**payload = record["body"]**

**print(str(payload))**

**# Put the item into the DynamoDB table**

**table.put\_item(Item={**

**'orderID': str(uuid.uuid4()),**

**'order': payload**

**})**

**import json**

**import logging**

**import boto3**

**logger = logging.getLogger()**

**logger.setLevel(logging.INFO)**

**# Initialize the DynamoDB resource**

**dynamodb = boto3.resource('dynamodb')**

**# Reference the DynamoDB table**

**table = dynamodb.Table("orders")**

**def lambda\_handler(event, context):**

**logger.info(f"Event received: {event}")**

**try:**

**if 'orderID' not in event:**

**logger.error("orderID not found in the event")**

**raise KeyError('orderID')**

**for record in event['orderID']:**

**# Process each record**

**logger.info(f"Processing record: {record}")**

**return {**

**'statusCode': 200,**

**'body': json.dumps('Success')**

**}**

**except KeyError as e:**

**logger.error(f"KeyError: {e} not found in event")**

**return {**

**'statusCode': 400,**

**'body': json.dumps(f"Bad Request: {e} not found")**

**}**

**except Exception as e:**

**logger.error(f"Unexpected error: {str(e)}")**

**return {**

**'statusCode': 500,**

**'body': json.dumps(f"Internal Server Error: {str(e)}")**

**}**

**=====**

**[ERROR] KeyError: 'orderID'**

**Traceback (most recent call last):**

**File "/var/task/lambda\_function.py", line 19, in lambda\_handler**

**for record in event['orderID']:END RequestId: d9ac1ecf-dd7d-485d-b8c4-01035e3f8f0a**

And so this is going to be where you enter in the S3 prefix

[**https://ap-south-1.console.aws.amazon.com/s3/object/djskinessdatabucket?region=ap-south-1&bucketType=general&prefix=result/Unsaved/2024/06/15/55ba50f3-2831-4d73-813e-1f53aade6249.txt**](https://ap-south-1.console.aws.amazon.com/s3/object/djskinessdatabucket?region=ap-south-1&bucketType=general&prefix=result/Unsaved/2024/06/15/55ba50f3-2831-4d73-813e-1f53aade6249.txt)

[**https://ap-south-1.console.aws.amazon.com/s3/buckets/djskinessdatabucket?region=ap-south-1&bucketType=general&prefix=result/Unsaved/2024/06/15/&showversions=false**](https://ap-south-1.console.aws.amazon.com/s3/buckets/djskinessdatabucket?region=ap-south-1&bucketType=general&prefix=result/Unsaved/2024/06/15/&showversions=false)

## Select Dynamo Db Table 🡪 Export and Streams Tab🡪 DynamoDB stream details

**Select API Gateway🡪 integration request Tab🡪 integration request setting vit**

**{**

**"DeliveryStreamName": "<Enter the name of your delivery streams>",**

**"Record": {**

**"Data": "Sutil.base64 Encode($util.escape JavaScript($input.json('$' )).replace('\', ''))"**

**}**

**}**

**{**

**"DeliveryStreamName": "arn:aws:firehose:ap-south-1:944346522085:deliverystream/PUT-S3-m9Qr8",**

**"Record": {**

**"Data": "Sutil.base64 Encode($util.escape JavaScript($input.json('$' )).replace('\', ''))"**

**}**

**}**

[**https://ap-south-1.console.aws.amazon.com/apigateway/main/apis/sy9pdlif28/resources/?api=sy9pdlif28&experience=rest&region=ap-south-1**](https://ap-south-1.console.aws.amazon.com/apigateway/main/apis/sy9pdlif28/resources/?api=sy9pdlif28&experience=rest&region=ap-south-1)

**api🡪 integration Request Tab🡪**

**Api test in recourses method code**

**{**

**"element\_clicked":"entree\_1",**

**"time spent":67,**

**"source\_menu":"restaurant\_name",**

**"created\_at":"2022-09-11 23:00:00"**

**}**

 [Amazon Data Firehose](https://ap-south-1.console.aws.amazon.com/firehose/home?region=ap-south-1#/home)

 [Firehose streams](https://ap-south-1.console.aws.amazon.com/firehose/home?region=ap-south-1#/streams)

 PUT-S3-m9Qr8

**arn:aws:firehose:ap-south-1:944346522085:deliverystream/PUT-S3-m9Qr8**

**Dynamo Db Update Lambda Test code**

**{**

**"Records": [**

**{**

**"eventID": "c4ca4238a0b923820dcc509a6f75849b",**

**"eventName": "INSERT",**

**"eventVersion": "1.1",**

**"eventSource": "aws:dynamodb",**

**"awsRegion": "us-east-1",**

**"dynamodb": {**

**"Keys": {**

**"Id": {**

**"N": "101"**

**}**

**},**

**"NewImage": {**

**"Message": {**

**"S": "New item!"**

**},**

**"Id": {**

**"N": "101"**

**}**

**},**

**"ApproximateCreationDateTime": 1428537600,**

**"SequenceNumber": "4421584500000000017450439091",**

**"SizeBytes": 26,**

**"StreamViewType": "NEW\_AND\_OLD\_IMAGES"**

**},**

**"eventSourceARN": "arn:aws:dynamodb:us-east-1:123456789012:table/ExampleTableWithStream/stream/2015-06-27T00:48:05.899"**

**},**

**{**

**"eventID": "c81e728d9d4c2f636f067f89cc14862c",**

**"eventName": "MODIFY",**

**"eventVersion": "1.1",**

**"eventSource": "aws:dynamodb",**

**"awsRegion": "us-east-1",**

**"dynamodb": {**

**"Keys": {**

**"Id": {**

**"N": "101"**

**}**

**},**

**"NewImage": {**

**"Message": {**

**"S": "This item has changed"**

**},**

**"Id": {**

**"N": "101"**

**}**

**},**

**"OldImage": {**

**"Message": {**

**"S": "New item!"**

**},**

**"Id": {**

**"N": "101"**

**}**

**},**

**"ApproximateCreationDateTime": 1428537600,**

**"SequenceNumber": "4421584500000000017450439092",**

**"SizeBytes": 59,**

**"StreamViewType": "NEW\_AND\_OLD\_IMAGES"**

**},**

**"eventSourceARN": "arn:aws:dynamodb:us-east-1:123456789012:table/ExampleTableWithStream/stream/2015-06-27T00:48:05.899"**

**},**

**{**

**"eventID": "eccbc87e4b5ce2fe28308fd9f2a7baf3",**

**"eventName": "REMOVE",**

**"eventVersion": "1.1",**

**"eventSource": "aws:dynamodb",**

**"awsRegion": "us-east-1",**

**"dynamodb": {**

**"Keys": {**

**"Id": {**

**"N": "101"**

**}**

**},**

**"OldImage": {**

**"Message": {**

**"S": "This item has changed"**

**},**

**"Id": {**

**"N": "101"**

**}**

**},**

**"ApproximateCreationDateTime": 1428537600,**

**"SequenceNumber": "4421584500000000017450439093",**

**"SizeBytes": 38,**

**"StreamViewType": "NEW\_AND\_OLD\_IMAGES"**

**},**

**"eventSourceARN": "arn:aws:dynamodb:us-east-1:123456789012:table/ExampleTableWithStream/stream/2015-06-27T00:48:05.899"**

**}**

**]**

**}**

fter some researches, I found the library [smart-open](https://pypi.org/project/smart-open/) very useful and simply to use.

from smart\_open import open

import json

s3\_client = s3\_session.client("s3")

source\_uri = 's3://my-bucket/my-path'

for json\_line in open(source\_uri, transport\_params={"client": s3\_client}):

my\_json = json.loads(json\_line)

After you have the buffer, try the following

decompressed = gzip.decompress(buffer)

json\_lines = json.loads(decompressed)

for json\_obj in json\_lines:

**CREATE EXTERNAL TABLE IF NOT EXISTS `default`.`djskinessdatabucket` (`ID` varchar(500), `File` varchar(500))**

**ROW FORMAT SERDE 'org.apache.hadoop.hive.ql.io.parquet.serde.ParquetHiveSerDe'**

**STORED AS INPUTFORMAT 'org.apache.hadoop.hive.ql.io.parquet.MapredParquetInputFormat' OUTPUTFORMAT 'org.apache.hadoop.hive.ql.io.parquet.MapredParquetOutputFormat'**

**LOCATION 's3://djskinessdatabucket/'**

**TBLPROPERTIES ('classification' = 'parquet');**

[**Amazon Athena**](https://ap-south-1.console.aws.amazon.com/athena/home?region=ap-south-1#/landing-page) **Table**

**CREATE EXTERNAL TABLE IF NOT EXISTS my\_ingested\_data (**

**element\_clicked STRING,**

**time\_spent INT,**

**source\_menu STRING,**

**created\_at STRING**

**)**

**PARTITIONED BY (**

**datehour STRING**

**)**

**ROW FORMAT SERDE 'org.openx.data.jsonserde.JsonSerDe'**

**WITH SERDEPROPERTIES (**

**'paths' = 'element\_clicked,time\_spent,source\_menu,created\_at'**

**)**

**LOCATION 's3://djstestbucketnew1/'**

**TBLPROPERTIES (**

**'projection.enabled' = 'true',**

**'projection.datehour.type' = 'date',**

**'projection.datehour.format' = 'yyyy/MM/dd/HH',**

**'projection.datehour.range' = '2024/01/01/00,NOW',**

**'projection.datehour.interval' = '1',**

**'projection.datehour.interval.unit' = 'HOURS',**

**'storage.location.template' = 's3://djstestbucketnew1/${datehour}/'**

**);**

**Q**

[**https://repost.aws/**](https://repost.aws/)

1. [Start of transcript. Skip to the end.](https://courses.edx.org/xblock/block-v1:AWS+AWS-ARCH-1+3T2022+type@vertical+block@a10cd3b389734164b350159804bb64d9?exam_access=&format=Quiz&recheck_access=1&show_bookmark=0&show_title=0&view=student_view#transcript-end-93c19c87e1664943b854d310387a062c)
2. - [Morgan] All right, I'm back.
3. And we can now see that there's a folder here.
4. So if I drill down into this,
5. we can actually see the file that was dropped here
6. by Kinesis Data Firehose.
7. And so notice how this is partitioned via date.
8. So that's going to become important in the next step
9. as we create the Amazon Athena table
10. that we're going to use to actually query this object.
11. So let's go ahead and go to Athena, and select Athena.
12. So now we're at the Athena dashboard,
13. and you need to make sure that you first,
14. before you can create a table,
15. you need to go into settings and then select manage.
16. And then from here
17. you need to select a query result location.
18. And so this is going to be where you enter in the S3 prefix
19. in this region where you want your query result
20. to be saved as an object.
21. And so, if you browse
22. and then you can select the bucket that you created
23. in the earlier step,
24. and then I just appended 'result' on the end of that.
25. And then click save.
26. And you need to make sure you do this
27. before you create the Amazon Athena table.
28. So now we can go back to the editor and then you can copy
29. and paste in the query to create the table.
30. And there's a couple of things that we can notice
31. about this and a couple of things we have to change
32. before we can run it.
33. So this is creating a table called My Ingested data.
34. And this has the the rows, element clicked,
35. time spent, source menu, created at,
36. which is the exact same structure that was sent
37. in by our click stream data front end,
38. what we kind of mocked up in API Gateway.
39. And then down below we have partition by date hour string.
40. And then we have the serialization derealization, which is
41. it's going to be in JSON,
42. because again we're sending in our data via JSON.
43. And then with these properties, element clicked,
44. time spent, source menu, created at,
45. so it kind of knows how to map
46. that JSON into something that Athena can query.
47. And then the table properties,
48. this is going to be how it's partitioned.
49. So like I mentioned before,
50. it partitions the data over dates in a specific format.
51. So this is just kind of telling Athena like,
52. how you can find the files
53. to query based on the way this data's partitioned.
54. So now what we need to do is take our bucket name
55. that we created in that earlier step
56. and then you want to make sure
57. that you are replacing this location here
58. S3 colon slash slash, and then your bucket name.
59. And then down here line 20 as well.
60. You want to make sure that you replace it here also.
61. All right, so now that looks good.
62. Let's go ahead and click run.
63. Alright, so now our table has been created
64. and now if we want to actually query the data in this,
65. we can click on the plus sign, create New Query,
66. and then we can say select star
67. from my ingested data semicolon.
68. And this is going to pull back
69. all of the data in this table.
70. We should only have seven items in the table.
71. In the real world
72. you might have hundreds of thousands, right?
73. So if you do have hundreds of thousands
74. you want to make sure you're setting up query limits
75. or that you're filtering
76. or selecting on something more specific.
77. So that way you aren't running giant queries on accident
78. in Athena that cost a lot of money
79. and take a lot of time and resources.
80. So instead, you want to make sure
81. you are limiting these when possible.
82. But right now we know we only have seven in there.
83. And we're just trying to make sure that this works.
84. So now we're going to go ahead and run the query.
85. All right.
86. And we can see we have seven results, which is great.
87. We have all of our entrees, the amount of time they spent.
88. We can see this data's in here.
89. So now the last step is to go to QuickSight
90. and create a visualization there.
91. So let's go ahead and navigate to QuickSight.
92. And whenever you come to QuickSight for the first time
93. you may need to create an account.
94. I already have an account with QuickSight
95. but you might need to go ahead and create an account.
96. So once you have your account created,
97. something else you need to do is actually
98. give QuickSight permission to that bucket.
99. So I'm going to go ahead and click on Manage QuickSight.
100. And then on the left hand side I'm going to say
101. Security and Permissions.
102. And then I'm going to click on manage
103. under QuickSight access to AWS services.
104. And then I want to make sure
105. that this Amazon S3 is selected.
106. And then you want to make sure that your bucket is selected,
107. specifically the one that you created.
108. And then also click the checkbox
109. for Write Permission to Athena work group.
110. And then click Finish.
111. And then click save.
112. All right, so now this is looking good.
113. We can go back to QuickSight just by clicking
114. the upper left hand corner.
115. And next what I want to do is create a new dataset.
116. So I'm going to click on data sets
117. in the left hand side, select new dataset,
118. and then I want to select Athena.
119. And then I want to give this a data source.
120. So I'm going to go ahead and call this POC Clickstream
121. which is the data source name.
122. And then I'm going to click create data source.
123. And now we can select the table that we want to visualize
124. which is going to be
125. that My Ingested Data that we just created, click select.
126. And now I want to click Import to Spice
127. for quicker analytics and click Visualize.
128. All right.
129. And now what we want to do is actually
130. create a visualization here.
131. So you can see all the different visual types
132. that you have down here in the left hand corner.
133. It depends on how your data is shaped on,
134. like what's going to be the most important to you,
135. or not necessarily shaped, what the attributes are
136. and what insights you're trying to derive.
137. So you can go through
138. and you can explore these different types
139. of data sets if you would like.
140. We can see our field list is over here.
141. I'm going to go ahead and just select a pie chart for now
142. and then select element clicked.
143. And there we go.
144. It took a second to refresh
145. but now we can see we have equal amounts of clicks
146. for each entree
147. because we only entered in one for each thing here.
148. So this is how you do some visualizations.
149. You can go through
150. and you can click through these and explore them.
151. And we will create a link
152. in the instructions where you can read more
153. about how this works with QuickSight.
154. All right, so it looks like our POC is working.
155. The last step
156. for us that we need to do is delete all of these resources.
157. So in QuickSight to start, since we're already here
158. I'm going to go ahead and
159. go back to the QuickSight dashboard.
160. And I'm going to click on data sets.
161. And then I'm going to select my ingested data
162. and then click delete, and then click delete.
163. All right.
164. And then if we go back to analyses,
165. we can select our analyses
166. that we just created and click delete, and delete again.
167. All right, and now we're done with that.
168. So now let's switch to the Athena dashboard.
169. So now we're in Athena.
170. And what I want to do is delete this table.
171. So I'm actually going to just change this query
172. and say Drop table my, if I can type.
173. There we go.
174. Ingested data, semicolon, click run.
175. And that drops the table.
176. So it deletes that table.
177. And then we can go ahead and delete these queries
178. or close the query.
179. If you would like,
180. you can just exit out of that one.
181. All right, so now Athena's been deleted.
182. So now if we kind of work backwards from here,
183. let's go ahead and go to API Gateway and delete that.
184. So click on the API, actions, delete, click delete.
185. Looks good there.
186. Now we can go to Kinesis Data Firehose,
187. click on Kinesis and then click on the delivery streams.
188. And then select this Kinesis delivery stream.
189. And then we want to type in the name.
190. I'm just going to copy and paste it there.
191. Give this a second to delete.
192. Now you want to make sure that things actually delete in AWS
193. so that way you don't get charged
194. for things that didn't delete correctly.
195. So if you ever see a failure with the deletion of resource,
196. just make sure you open up an AWS support ticket
197. so that you can get some help with that.
198. There we go.
199. It's successfully deleted.
200. Great. So now we can see that's deleted.
201. So now let's go ahead and go to Lambda,
202. and we'll delete the Lambda function.
203. Select the Lambda function, click actions, click delete,
204. and then type in delete.
205. You want to spell delete correctly.
206. There we go.
207. Click delete, click close.
208. And what's next?
209. Probably the S3 bucket.
210. Select S3.
211. And then I want to select this test bucket that we created.
212. I want to first empty the bucket.
213. So select all the objects, click delete,
214. type in permanently delete.
215. I'm just going to go ahead and copy this and paste it below.
216. Delete objects.
217. Looks good.
218. Now I'm going to go back to the source bucket,
219. and go back to buckets,
220. select the bucket, and then I'm going to delete it.
221. And then I'm going to paste in the name,
222. copy and paste the name
223. of the bucket in there, click delete bucket.
224. And I think that that is everything.
225. You could also delete the IAM roles if you would like.
226. All right, so that is if for this POC.
227. If you are looking for troubleshooting tips,
228. I would recommend that you,
229. if you see that API Gateway is not integrating with Kinesis,
230. check the permissions between those two things.
231. Make sure Kinesis is actually created.
232. The next step is Kinesis and Lambda.
233. Make sure that you actually have the Lambda function
234. created correctly and that you have
235. enabled data transformation on the Kinesis Data Firehose.
236. The next step is S3.
237. Check your bucket policy.
238. If your bucket policy is not allowing that role
239. from Kinesis Data Firehose, then it's not going to work.
240. And then after that, we have Athena.
241. So you want to make sure
242. that you have that settings for your results set up
243. so it knows where to post the result object
244. you need that set up before you can run a query.
245. And then for QuickSight,
246. you also have to allow the permissions
247. to your specific S3 bucket in order for to read that data.
248. So you want to make sure that you have it there as well.
249. So again, you can kind of go step by step by step
250. to see like, where,
251. how far did the data get in the solution.
252. And then honestly, mostly the time the issue is permissions.
253. So you want to make sure you're double checking
254. your permissions every step that you get.
255. All right, thank you.
256. And that's it.
257. See you guys next time.
258. [End of transcript. Skip to the start.](https://courses.edx.org/xblock/block-v1:AWS+AWS-ARCH-1+3T2022+type@vertical+block@a10cd3b389734164b350159804bb64d9?exam_access=&format=Quiz&recheck_access=1&show_bookmark=0&show_title=0&view=student_view#transcript-start-93c19c87e1664943b854d310387a062c)

**{**

**"Version": "2012-10-17",**

**"Statement": [**

**{**

**"Effect": "Allow",**

**"Principal": {**

**"Service": "lambda.amazonaws.com"**

**},**

**"Action": "sts:AssumeRole"**

**}**

**]**

**}**

**{**

**"Version": "2012-10-17",**

**"Statement": [**

**{**

**"Effect": "Allow",**

**"Principal": {**

**"AWS": "arn:aws:iam::944346522085:role/kinesistransformdata-role-mp7kf4g5"**

**},**

**"Action": [**

**"dynamodb:Query",**

**"dynamodb:GetItem"**

**],**

**"Resource": "arn:aws:dynamodb:ap-south-1:944346522085:table/orders"**

**}**

**]**

**}**

[**https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/rbac-examples.html**](https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/rbac-examples.html)

**arn:aws:dynamodb:ap-south-1:944346522085:table/orders**

### Reading 3.1: Hybrid Networking and Connectivity

## Level 2 headings may be created by course providers in the future.

Bookmark this page

This week, the customer has a hybrid deployment, and they require a consistent connection between their data center and AWS. You should consider multiple architecture choices for this solution. However, Morgan ultimately chose AWS Direct Connect because the customer needed consistent, dedicated throughput for the high volume of data that flows between the data center and AWS.  
  
For more information about common networking setups with AWS, see [Network-to-Amazon VPC connectivity options](https://docs.aws.amazon.com/whitepapers/latest/aws-vpc-connectivity-options/network-to-amazon-vpc-connectivity-options.html).

## AWS Direct Connect

The AWS Direct Connect cloud service is the shortest path to your AWS resources. While your network traffic is in transit, it remains on the AWS global network and never touches the public internet. This isolation reduces the chance of encountering bottlenecks or unexpected increases in latency.  
  
When you create a new connection, you can choose a hosted connection that’s provided by an AWS Direct Connect Delivery Partner, or choose a dedicated connection from AWS—and deploy at over 100 AWS Direct Connect locations around the globe. With AWS Direct Connect SiteLink, you can send data between AWS Direct Connect locations to create private network connections between the offices and data centers in your global network.  
  
Each dedicated AWS Direct Connect connection consists of a single dedicated connection between ports on your router and an AWS Direct Connect device. We recommend establishing a second connection for redundancy.  
  
When you request multiple ports at the same AWS Direct Connect location, they will be provisioned on redundant AWS equipment.  
  
If you have configured a backup Internet Protocol security (IPsec) virtual private network (VPN) connection instead, all virtual private cloud (VPC) traffic will fail over to the VPN connection automatically. Traffic to or from public resources, such as Amazon Simple Storage Service (Amazon S3), will be routed over the internet. If you don’t have a backup AWS Direct Connect link or an IPsec VPN link, then Amazon VPC traffic will be dropped if a failure occurs. Traffic to or from public resources will be routed over the internet.  
  
  
For more general information about AWS Direct Connect, see [What is AWS Direct Connect?](https://docs.aws.amazon.com/directconnect/latest/UserGuide/Welcome.html) and [AWS Direct Connect FAQs.](https://aws.amazon.com/directconnect/faqs/?nc=sn&loc=6)

## AWS Managed VPN

### AWS Site-to-Site VPN

By default, instances that you launch into a VPC can't communicate with your own (remote) network. You can enable access to your remote network from your VPC by creating an AWS Site-to-Site VPN connection, and configuring routing to pass traffic through the connection.  
  
*VPN connection* is a general term, but in AWS, a VPN connection refers to the connection between your VPC and your own on-premises network. Site-to-Site VPN supports IPsec VPN connections.

### Virtual private gateway

A virtual private gateway is the VPN concentrator on the Amazon side of the Site-to-Site VPN connection. You create a virtual private gateway and attach it to the VPC where you want to create the Site-to-Site VPN connection.  
  
  
Consider taking this approach when you want to take advantage of an AWS managed VPN endpoint that includes automated redundancy and failover built into the AWS side of the VPN connection.  
  
The virtual private gateway also supports and encourages multiple user gateway connections so that you can implement redundancy and failover on your side of the VPN connection, as shown in the following figure.  
  
  
The previous diagram also shows a customer gateway on the customer-network side. A customer gateway device is a physical or software appliance that you own or manage in your on-premises network (that is, on your side of a Site-to-Site VPN connection). You or your network administrator must configure the device to work with the Site-to-Site VPN connection.

### AWS Client VPN

AWS Client VPN is a fully managed, remote-access VPN solution that your remote workforce can use to securely access resources within both AWS and your on-premises network. It’s fully elastic, so it automatically scales up or down, based on demand. When you migrate applications to AWS, your users access the applications in the same way before, during, and after the move. AWS Client VPN, including the software client, supports the OpenVPN protocol.  
  
The following scenarios are examples of how you could use AWS Client VPN.

#### Scenario 1

The configuration for this scenario includes a single target VPC. We recommend this configuration if you need to give clients access to the resources inside only a single VPC.

#### Scenario 2

The configuration for this scenario includes access to only an on-premises network. We recommend this configuration if you need to give clients access to the resources inside only an on-premises network.

#### Scenario 3

In the configuration for the scenario shown in the following diagram, clients can access a single VPC, and clients can also route traffic to each other. We recommend this configuration if the clients that connect to the same client VPN endpoint also need to communicate with each other. Clients can communicate with each other by using the unique IP address that's assigned to them from the client Classless Inter-Domain Routing (CIDR) range when they connect to the client VPN endpoint.

* For more information about AWS Site-to-Site VPN, see [What is Amazon Site-to-Site VPN?](https://docs.aws.amazon.com/vpn/latest/s2svpn/VPC_VPN.html)

For more resources about AWS VPN services, see the following:

* For more information about AWS Client VPN, see [What is AWS Client VPN?](https://docs.aws.amazon.com/vpn/latest/clientvpn-admin/what-is.html)

## AWS Transit Gateway

Morgan mentioned that if multiple Regions or AWS accounts were all connected to a remote data center by using AWS Direct Connect, then she might recommend using AWS Transit Gateway.  
  
AWS Transit Gateway connects your VPCs and on-premises networks through a central hub. This arrangement simplifies your network and minimizes complex peering relationships. Transit Gateway acts as a cloud router—each new connection is made only once.  
  
As you expand globally, inter-Region peering connects transit gateways together through the AWS global network. Your data is automatically encrypted and never travels over the public internet.

### Without AWS Transit Gateway

The following diagram shows what a network solution might look like without the use of AWS Transit Gateway. Notice that there are many different connection points between the different components. This network configuration is complex, and it can be difficult to manage.

### With AWS Transit Gateway

In contrast, the following diagram shows what a network solution might look like with the use of AWS Transit Gateway. Transit Gateway is the hub for the connections between the different networks. Using a transit gateway can make the network easier to manage. You can apply route tables so that the transit gateway can connect networks to each other in a more centralized manner.  
The following example shows what using AWS Transit Gateway with AWS Direct Connect could look like for a solution that has a remote network and three AWS Regions. Each Region has multiple VPCs.  
  
  
For more information about AWS Transit Gateway, see [What is a transit gateway?](https://docs.aws.amazon.com/vpc/latest/tgw/what-is-transit-gateway.html)

[**https://learning.edx.org/course/course-v1:AWS+AWS-ARCH-1+3T2022/block-v1:AWS+AWS-ARCH-1+3T2022+type@sequential+block@38a938313cfc46f9b8ec2d20f15fd231/block-v1:AWS+AWS-ARCH-1+3T2022+type@vertical+block@5a2e161b6cc24a1182609a209bf6ce54**](https://learning.edx.org/course/course-v1:AWS+AWS-ARCH-1+3T2022/block-v1:AWS+AWS-ARCH-1+3T2022+type@sequential+block@38a938313cfc46f9b8ec2d20f15fd231/block-v1:AWS+AWS-ARCH-1+3T2022+type@vertical+block@5a2e161b6cc24a1182609a209bf6ce54)

1. AWS offers two container orchestration services:
2. Amazon Elastic Container Service
3. and Amazon Elastic Kubernetes Service.
4. These two services will manage containers
5. **on top of a compute platform,**
6. either a cluster of EC2 instances
7. or a serverless AWS Fargate cluster.
8. So, for the first decision,

### eading 3.2: Running Containers on AWS and NAT Gateways

Level 2 heading This week’s customer needed a solution to host containers on AWS. These containers will be running internal applications that don’t require inbound communication from the internet. However, the applications might require outbound communication to the internet so they can do tasks such as download updates from internet sources. The customer must use their own custom Amazon Machine Image (AMI) for the cluster that hosts the containers, and they must also have SSH access to underlying instances. For these reasons, Morgan chose Amazon Elastic Container Service (Amazon ECS) as the container orchestration tool and Amazon Elastic Compute Clear (Amazon EC2) as the launch type. Morgan also chose to include a NAT gateway in the architecture so that private instances could download information from the internet.

## Amazon ECS launch types

After you choose which container orchestration tool you want to use—either Amazon ECS or Amazon Elastic Kubernetes Services (Amazon EKS)—there are two launch types to pick from:

* EC2: Deploy and manage your own cluster of EC2 instances for running the containers
* AWS Fargate: Run containers directly, without any EC2 instances

Both launch types are good compute options for hosting your containers in a scalable and reliable way. Which launch type you choose will depend on which factors you want to optimize for.  
  
In the video, Morgan chose the EC2 launch type for containers that were being run by Amazon ECS. She chose the EC2 launch type because the customer wants to use a custom AMI. The customer also wants to maintain SSH access to underlying instances so they could try to have similar management operations across workloads. AWS Fargate doesn’t support either of these options. Therefore, EC2 was the correct choice for this use case.

### EC2 launch type

You can use the EC2 launch type to run your containerized applications on EC2 instances, which you register to your Amazon ECS cluster and manage yourself.  
  
The following diagram shows the general architecture.  
The diagram shows that the ECS agent is installed on each EC2 instance in the cluster. The ECS agent enables the orchestration tool that Amazon ECS uses to manage nodes.

## Fargate launch type

You can use the Fargate launch type to run your containerized applications without the need to provision and manage the underlying infrastructure. AWS Fargate is the serverless way to host your Amazon ECS workloads.  
  
The following diagram shows the general architecture.  
  
  
For more resources about containers on AWS, see the following:

* For general information about AWS container services, see [Containers at AWS](https://aws.amazon.com/containers/)or the [Customers FAQs](https://aws.amazon.com/containers/faqs/) on the AWS *Containers*site.
* For more information about ECS launch types, see [Amazon ECS launch types](https://docs.aws.amazon.com/AmazonECS/latest/developerguide/launch_types.html).

## NAT devices

You can use a NAT device to allow resources in private subnets to connect to the internet, other VPCs, or on-premises networks. These instances can communicate with services outside the VPC, but they can’t receive unsolicited connection requests.  
  
The NAT device replaces the source IPv4 address of the instances with the address of the NAT device. When the NAT device sends response traffic to the instances, the device translates the addresses back to the original source IPv4 addresses.  
  
You can use a managed NAT device that’s offered by AWS, which is called a *NAT gateway.* You can also create your own NAT device on an EC2 instance, which is called a *NAT instance*. We recommend that you use NAT gateways because they provide better availability and bandwidth, and administering NAT gateways requires less effort on your part.  
  
You can manage how the traffic flows from the private resources to the NAT device by using route tables.

### NAT instances

You can create your own AMI that provides network address translation, and use your AMI to launch an EC2 instance as a NAT instance. You can launch a NAT instance in a public subnet so that instances in a private subnet can initiate outbound IPv4 traffic to the internet or other AWS services without receiving inbound traffic that was initiated on the internet.

### NAT gateways

A NAT gateway is a NAT service. You can use a NAT gateway so that instances in a private subnet can connect to services that are outside your VPC, but external services can’t initiate a connection with those instances.  
  
When you create a NAT gateway, you specify one of the following connectivity types:

* Public (default): Instances in private subnets can connect to the internet through a public NAT gateway, but can’t receive unsolicited inbound connections from the internet. You create a public NAT gateway in a public subnet, and you must associate an elastic IP address with the NAT gateway at creation. You route traffic from the NAT gateway to the internet gateway for the VPC. Alternatively, you can use a public NAT gateway to connect to other VPCs or your on-premises network. In this case, you route traffic from the NAT gateway through a transit gateway or a virtual private gateway.
* Private: Instances in private subnets can connect to other VPCs or your on-premises network through a private NAT gateway. You can route traffic from the NAT gateway through a transit gateway or a virtual private gateway. You can’t associate an elastic IP address with a private NAT gateway. You can attach an internet gateway to a VPC with a private NAT gateway, but if you route traffic from the private NAT gateway to the internet gateway, the internet gateway drops the traffic.

The NAT gateway replaces the source IP address of the instances with the IP address of the NAT gateway. For a public NAT gateway, this is the elastic IP address of the NAT gateway. For a private NAT gateway, this is the private IP address of the NAT gateway. When the NAT device sends response traffic to the instances, it translates the addresses back to the original source IP address.  
  
For more resources about NAT devices, see the following:

* For more information, see [Connect to the internet or other networks using NAT devices](https://docs.aws.amazon.com/vpc/latest/userguide/vpc-nat.html).
* For information comparing NAT devices, see [Compare NAT gateways and NAT instances](https://docs.aws.amazon.com/vpc/latest/userguide/vpc-nat-comparison.html).

## s may be created by course providers in the future

## .

Bookmark this page

**Explain each activity in this conversation with steps and examples in aws mangement console**

### Reading 3.3: Amazon RDS

## Multi-AZ deployments

For this week’s customer, Morgan recommends that they migrate their on-premises database to Amazon Relational Database Service (Amazon RDS), and to use a Multi-AZ deployment for high availability.  
  
In a Multi-AZ deployment, Amazon RDS automatically creates a primary database (DB) instance and synchronously replicates the data to an instance in a different Availability Zone. When it detects a failure, Amazon RDS automatically fails over to a standby instance without manual intervention. This failover mechanism meets the customer’s need to have a highly available database.  
  
The following diagram shows a Multi-AZ deployment with one standby DB instance, and how it works.  
  
For even higher availability, the customer could explore deploying two standby DB instances, and use three Availability Zones instead of two.  
  
Say that you deploy MySQL or PostgreSQL databases in three Availability Zones by using Amazon RDS Multi-AZ with two readable standbys. With this configuration, you can see automatic failovers typically happen in under 35 seconds, and with a transaction-commit latency that can be up to two times faster when compared to an Amazon RDS Multi-AZ deployment with one standby. You can also gain additional read capacity, and a choice of AWS Graviton2–based or Intel–based instances for compute.  
  
The following diagram shows a Multi-AZ deployment with two standby DB instances, and how it works.

## Read replicas

Customers can also make RDS more highly available by using read replicas.  
  
Amazon RDS read replicas provide enhanced performance and durability for Amazon RDS DB instances. For read-heavy database workloads, read replicas make it easier to elastically scale out beyond the capacity constraints of a single DB instance.  
  
You can create one or more replicas of a given source DB instance and serve high-volume application read traffic from multiple copies of your data, which increases aggregate read throughput. Read replicas can also be promoted to become standalone DB instances, when needed.  
  
Read replicas are available in Amazon RDS for MySQL, MariaDB, PostgreSQL, Oracle, Microsoft SQL Server, and Amazon Aurora.  
  
For the MySQL, MariaDB, PostgreSQL, Oracle, and SQL Server database engines, Amazon RDS creates a second DB instance by using a snapshot of the source DB instance. Amazon RDS then uses the engine’s native asynchronous replication to update the read replica when there’s a change to the source DB instance.  
  
The read replica operates as a DB instance that allows only read-only connections. Applications can connect to a read replica like they would connect to any DB instance. Amazon RDS replicates all databases in the source DB instance.  
  
Here’s an example of when to use a read replica. Say that you’re running reports on your database, which is causing performance issues with CPU-intensive reads. You can use a read replica and direct all the reporting queries to that replica instead of to the primary instance. Offloading some of the intense queries to the replica should result in enhanced performance on the primary instance.

## Scaling Amazon RDS instances

### Scale your instance up vertically

When you create an RDS DB instance, you choose a database instance type and size.  
  
Amazon RDS provides a selection of instance types that are optimized to fit different use cases for relational databases. Instance types comprise varying combinations of CPU, memory, storage, and networking capacity. You have the flexibility to choose the appropriate mix of resources for your database. Each instance type includes several instance sizes, which means that you can scale your database to your target workload’s requirements.  
  
Not every instance type is supported for every database engine, version, edition or Region.  
  
When you want to scale your DB instance, you can vertically scale the instance and choose a larger instance size. This might be the route you choose to take when you need more CPU and storage ability for an instance.

### Use read replicas

If you need more CPU capabilities but don’t need more storage, you might choose to create read replicas to offload some of the workload to a secondary instance.

### Enable RDS Storage Auto Scaling

If you need more storage, but don’t need more CPU, then you could scale the storage horizontally. You can scale storage horizontally by allocating more storage volumes for your instance manually, or by enabling RDS Storage Auto Scaling. RDS Storage Auto Scaling automatically scales storage capacity in response to growing database workloads, with virtually zero downtime.  
  
Previously, you needed to manually provision storage capacity based on anticipated application demands. Underprovisioning could result in application downtime, and overprovisioning could result in underutilized resources and higher costs. With RDS Storage Auto Scaling, you set your desired maximum storage limit and Auto Scaling takes care of the rest.  
  
RDS Storage Auto Scaling continuously monitors actual storage consumption, and scales capacity up automatically when actual utilization approaches provisioned storage capacity.  
  
Auto Scaling works with new and existing database instances. You can enable Auto Scaling with a few clicks in the AWS Management Console.

### Change the storage type for increased performance

Finally, here’s one last thing to consider. If you’re looking for better performance, consider using a different storage type. For example, using Provisioned IOPS instead of General Purpose could give you some of the performance enhancements that you want.  
  
The following list briefly describes the three storage types:

* General Purpose SSD: General Purpose SSD volumes offer cost-effective storage that works well for a broad range of workloads. These volumes deliver single-digit millisecond latencies and the ability to burst to 3,000 IOPS for extended periods of time. Baseline performance for these volumes is determined by the volume's size.
* Provisioned IOPS: Provisioned IOPS storage is designed to meet the needs of I/O-intensive workloads—particularly database workloads—that require low I/O latency and consistent I/O throughput.
* Magnetic: Amazon RDS also supports magnetic storage for backward compatibility. We recommend that you use General Purpose SSD or Provisioned IOPS for any new storage needs. The maximum amount of storage that’s allowed for DB instances on magnetic storage is less than that of the other storage types.

For more resources about Amazon RDS, see the following:

* For more information about Amazon RDS Multi-AZ Deployments, see [Amazon RDS Multi-AZ](https://aws.amazon.com/rds/features/multi-az/).
* For more information about Amazon RDS read replicas, see [Amazon RDS Read Replicas](https://aws.amazon.com/rds/features/read-replicas/).
* For more information about Amazon RDS DB instances, see [Amazon RDS DB instances](https://docs.aws.amazon.com/AmazonRDS/latest/UserGuide/Overview.DBInstance.html).

## AWS DMS

Morgan suggested that this week’s customer use AWS Database Migration Service (Amazon DMS) to migrate data from their on-premises database to Amazon RDS.  
  
AWS DMS helps you migrate databases to AWS quickly and securely. The source database remains fully operational during the migration, which minimizes the downtime to applications that rely on the database. The AWS DMS can migrate your data to and from widely used commercial and open-source databases.  
  
At a basic level, AWS DMS is a server in the AWS Cloud that runs replication software. You create a source and target connection to tell AWS DMS where to extract from and load to. Then, you schedule a task that runs on this server to move your data. AWS DMS creates the tables and associated primary keys if they don't exist on the target. You can precreate the target tables yourself, if you prefer. You can also use AWS Schema Conversion Tool (AWS SCT) to create some or all of the target tables, indexes, views, triggers, and so on.  
  
The following diagram shows the AWS DMS replication process.  
  
  
This week’s customer would use AWS DMS to perform a homogenous database migration.  
  
In homogeneous database migrations, the source and target database engines are the same, or are compatible—such as Oracle to Amazon RDS for Oracle, MySQL to Amazon Aurora, MySQL to Amazon RDS for MySQL, or Microsoft SQL Server to Amazon RDS for SQL Server. Because the schema structure, data types, and database code are compatible between the source and target databases, this kind of migration is typically a one-step process. You create a migration task with connections to the source and target databases, and then start the migration. AWS DMS takes care of the rest. The source database can be located in your own premises outside of AWS, on an Amazon Elastic Compute Cloud (Amazon EC2) instance, or in an Amazon RDS DB instance. The target can be a database in Amazon EC2 or Amazon RDS.  
  
  
For more general information about AWS DMS, see [What is AWS Database Migration Service?](https://docs.aws.amazon.com/dms/latest/userguide/Welcome.html)

<https://ap-south-1.console.aws.amazon.com/rds/home?region=ap-south-1#databases>:

[**https://learning.edx.org/course/course-v1:AWS+AWS-ARCH-1+3T2022/block-v1:AWS+AWS-ARCH-1+3T2022+type@sequential+block@38a938313cfc46f9b8ec2d20f15fd231/block-v1:AWS+AWS-ARCH-1+3T2022+type@vertical+block@fb0363909921424d975c8d4c8354ecc8**](https://learning.edx.org/course/course-v1:AWS+AWS-ARCH-1+3T2022/block-v1:AWS+AWS-ARCH-1+3T2022+type@sequential+block@38a938313cfc46f9b8ec2d20f15fd231/block-v1:AWS+AWS-ARCH-1+3T2022+type@vertical+block@fb0363909921424d975c8d4c8354ecc8)

1. Let's go ahead and pull up aws.amazon.com/products/storage
2. Now, Amazon S3 itself does not support NFS,
3. like our customer needs,
4. but there's a service that is designed for hybrid solutions,
5. like our use case, called AWS Storage Gateway.
6. Pulling up the Storage Gateway webpage
7. **at aws.amazon.com/storagegateway.**

How the below Platform options can be used in aws console Host platform VMware ESXi Microsoft Hyper-V Linux KVM Amazon EC2 Hardware appliance

explain steps to configure the above platforms in aws console with steps and example

what to write in Hardware Appliance box Set up gateway in storage gateway

[**https://aws.amazon.com/remote-work-learning/?sc\_icampaign=Adoption\_Campaign\_pac-edm-2020-remote\_work-site\_merch-hero&sc\_ichannel=ha&sc\_icontent=awssm-4028\_f45&sc\_ioutcome=Enterprise\_Digital\_Marketing&sc\_iplace=hero&trk=ha\_a134p000006BkJTAA0~ha\_awssm-4028\_f45&trkCampaign=pac-edm-2020-remote\_work-pdp**](https://aws.amazon.com/remote-work-learning/?sc_icampaign=Adoption_Campaign_pac-edm-2020-remote_work-site_merch-hero&sc_ichannel=ha&sc_icontent=awssm-4028_f45&sc_ioutcome=Enterprise_Digital_Marketing&sc_iplace=hero&trk=ha_a134p000006BkJTAA0~ha_awssm-4028_f45&trkCampaign=pac-edm-2020-remote_work-pdp)

### Reading 3.4: AWS Storage Services

## Level 2 headings may be created by course providers in the future.

When you consider which storage service to choose for a specific use case, it’s important to be familiar with the different storage services that AWS has to offer. For this week’s customer, Morgan chose AWS Storage Gateway and Amazon Simple Storage Service (Amazon S3) as the services to support the solution. Morgan chose these services because the customer requires the Network File System (NFS) protocol to remain in place for all on-premises applications. However, the customer also wants to store the files (that they will access) in AWS. Storage Gateway supports this use case.

## AWS Storage Gateway

AWS Storage Gateway connects an on-premises software appliance with cloud-based storage to provide near-seamless integration with data security features between your on-premises IT environment and the AWS storage infrastructure. You can use the service to store data in the AWS Cloud for scalable and cost-effective storage that helps maintain data security. AWS Storage Gateway offers file-based, volume-based, and tape-based storage solutions.  
  
  
Morgan recommended Amazon S3 File Gateway for the customer’s solution.

## Amazon S3 File Gateway

Amazon S3 File Gateway supports a file interface into Amazon S3, and it combines a service and a virtual software appliance. By using this combination, you can store and retrieve objects in Amazon S3 by using industry-standard file protocols, such as NFS and Server Message Block (SMB). The software appliance (or gateway) is deployed into your on-premises environment as a virtual machine (VM) that runs on VMware ESXi, Microsoft Hyper-V, or Linux Kernel-based Virtual Machine (KVM) hypervisor. The gateway provides access to objects in Amazon S3 as files or file-share mount points. With S3 File Gateway, you can do the following:

* Store and retrieve files directly by using NFS version 3 or version 4.1.
* Store and retrieve files directly by using SMB file system version 2 or version 3.
* Access data directly in Amazon S3 from any AWS Cloud application or service.
* Manage your Amazon S3 data by using lifecycle policies, S3 Cross-Region Replication (CRR), and versioning. You can think of a S3 File Gateway as a file system mount on Amazon S3.

Amazon S3 File Gateway is designed to simplify file storage in Amazon S3. It integrates to existing applications through industry-standard file system protocols, and it provides a cost-effective alternative to on-premises storage. Amazon S3 File Gateway also provides low-latency access to data through transparent local caching. It manages data transfer to and from AWS, and it optimizes and streams data in parallel. Amazon S3 File Gateway also buffers applications from network congestion, and manages bandwidth consumption.  
  
For more resources about Amazon S3 File Gateway, see the following:

* For more information about Amazon S3 File Gateway, see [What is Amazon S3 File Gateway?](https://docs.aws.amazon.com/filegateway/latest/files3/what-is-file-s3.html)
* For more information about AWS Storage Gateway, see [AWS Storage Gateway Features](https://aws.amazon.com/storagegateway/features/).

## Amazon EBS

An Amazon Elastic Block Store (Amazon EBS) volume is a durable, block-level storage device that you can attach to your Amazon Elastic Compute Cloud (Amazon EC2) instances. After you attach a volume to an instance, you can use it as you would use a physical hard drive. EBS volumes are flexible. For current-generation volumes attached to current-generation instance types, you can dynamically increase size, modify the provisioned IOPS capacity, and change volume type on live production volumes.  
  
It’s important that you know about the different types of EBS volume types and sizes, and how IOPS are correlated with volume size and type.  
  
Amazon EBS provides the following volume types, which differ in performance characteristics and price, so that you can tailor your storage performance and cost to the needs of your applications.

* Solid-state drives (SSD): Optimized for transactional workloads involving frequent read/write operations with small I/O size, where the dominant performance attribute is IOPS. SSD-backed volume types include General Purpose SSD volumes and Provisioned IOPS SSD volumes
* Hard disk drives (HDD): Optimized for large, streaming workloads where the dominant performance attribute is throughput. HDD-backed volume types include Throughput Optimized HDD and Cold HDD volumes.
* Previous generation: Hard disk drives that you can use for workloads with small datasets, where data is accessed infrequently and performance is not a priority. We recommend that you consider a current-generation volume type instead.

Note that the number of available IOPs increases with the size of the volume. Thus, if you need more IOPs, you must vertically scale the volume.  
  
For more information, see [Amazon EBS volume types](https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/ebs-volume-types.html).

## Amazon EFS

Amazon Elastic File System (Amazon EFS) provides a simple, serverless elastic file system that you can use with AWS Cloud services and on-premises resources. It is built to scale on demand to petabytes without disrupting applications. Amazon EFS can grow and shrink automatically as you add and remove files, so it minimizes the need to provision and manage capacity to accommodate growth. Amazon EFS has a web services interface that you can use to create and configure file systems quickly and easily. The service manages all the file storage infrastructure for you, meaning that you can reduce the complexity of deploying, patching, and maintaining complex file system configurations.  
  
Amazon EFS supports NFS version 4 (NFSv4.1 and NFSv4.0), so the applications and tools that you use today work with Amazon EFS. Multiple compute instances—including Amazon EC2, Amazon Elastic Container Service (Amazon ECS), and AWS Lambda—can access an Amazon EFS file system at the same time. Amazon EFS can thus provide a common data source for workloads and applications that run on more than one compute instance or server.  
  
For more information, see [What is Amazon EFS?](https://docs.aws.amazon.com/efs/latest/ug/whatisefs.html)

## Amazon S3

Amazon S3 is an object storage service that stores data as objects within buckets. An *object* is a file and any metadata that describes the file. A *bucket* is a container for objects.  
  
To store your data in Amazon S3, you first create a bucket and specify a bucket name and AWS Region. Then, you upload your data to that bucket as objects in Amazon S3. Each object has a *key* (or *key name*), which is the unique identifier for the object within the bucket.  
  
S3 provides features that you can configure to support your specific use case. For example, you can use S3 Versioning to keep multiple versions of an object in the same bucket, which means that you can restore objects that are accidentally deleted or overwritten.

### Storage classes

Amazon S3 offers a range of storage classes that are designed for different use cases. For example, you could store mission-critical production data in S3 Standard for frequent access. You could also save on costs by storing infrequently accessed data in S3 Standard-IA or S3 One Zone-IA. Finally, you could archive data at a low costs in S3 Glacier Instant Retrieval, S3 Glacier Flexible Retrieval, and S3 Glacier Deep Archive.  
  
You can store data with changing or unknown access patterns in S3 Intelligent-Tiering, which optimizes storage costs by automatically moving your data between four access tiers when your access patterns change. These four access tiers include two low-latency access tiers, which are optimized for frequent and infrequent access. The four access tiers also include two opt-in archive access tiers that are designed for asynchronous access to rarely accessed data.  
  
For more resources about Amazon S3, see the following:

* For more information, see [What is Amazon S3?](https://docs.aws.amazon.com/AmazonS3/latest/userguide/Welcome.html)
* For more information, including links out to hands-on tutorials, see [Getting Started with Amazon S3](https://aws.amazon.com/s3/getting-started/?nc=sn&loc=6&dn=1).
* For more information about Amazon S3 storage classes, see [Amazon S3 Storage Classes](https://aws.amazon.com/s3/storage-classes/).
* at aws.amazon.com/hybrid,
* first need to create an activation key
* so you can register the VMs you plan to use
* for the containers on premises.
* Then, you need to install the AWS Systems Manager Agent,
* or the SSM Agent, and the ECS agent on those servers,
* using that activation key.
* We will talk more about Systems Manager
* in a few minutes, here.
* After that, you then define your application
* and deploy it the same way you would with ECS in AWS,
* and ECS can then manage those containers on premises.
* Okay, so back to our hybrid cloud webpage, here.
* The next category for services is Storage.
* You should be familiar with one of these already,
* as AWS Storage Gateway is listed here,
* and we do intend to use Storage Gateway for file storage.
* Then there is AWS Backup, which is a really awesome service
* that you should be familiar with.
* AWS Backup centralizes data-protection management
* and compliance for your applications running on AWS
* or in hybrid environments.
* Using AWS Backup, you can protect VMware workloads
* running on premises and on AWS,
* as well as data stored on AWS Storage Gateway volumes.
* Our customer may want to consider using AWS Backup
* to manage their backups across environments.
* So, let's go ahead and add AWS Backup
* to the architectural diagram here, as well.
* Our customer may or may not choose to use this,
* but I will include it on their architecture,
* and they can look further into whether it's a fit or not,
* but I want to make sure that we bring it to their attention.
* Now, on to networking.
* [AWS Systems Manager](https://ap-south-1.console.aws.amazon.com/systems-manager/home?region=ap-south-1&tab=Table)
* [Parameter Store](https://ap-south-1.console.aws.amazon.com/systems-manager/parameters?region=ap-south-1&tab=Table)
* [s3-bucket-for-app](https://ap-south-1.console.aws.amazon.com/systems-manager/parameters/s3-bucket-for-app/description?region=ap-south-1&tab=Table)
* **Overview**

arn:aws:ssm:ap-south-1:944346522085:parameter/s3-bucket-for-app

Value

Djsexamplebucket

https://ap-south-1.console.aws.amazon.com/systems-manager/run-command?region=ap-south-1#

### AWS Services for Hybrid Deployments

## Level 2 headings may be created by course providers in the future.

Bookmark this page

This week’s customer explained that they want to use the same management tools in both their data center and AWS when possible. To support this requirement, Morgan suggested that the customer migrate their on-premises containers to use Amazon Elastic Container Service (Amazon ECS) Anywhere so that they can use common tooling for running containers. Morgan also suggested that the customer use the following tools to manage resources across environments: AWS Systems Manager for operational support and tasks, and AWS Backup for managing backups in one centralized place.

## Amazon ECS Anywhere

Amazon ECS Anywhere is a feature of Amazon ECS that you can use to run and manage container workloads on customer-managed infrastructure.  
  
Amazon ECS Anywhere builds on the ease and simplicity of Amazon ECS to provide a consistent experience across your container-based applications for working with tooling and APIs. Whether on premises or in the cloud, cluster management, workload scheduling, and monitoring will be similar to what you already know from Amazon ECS. You can reduce costs and mitigate complex local container orchestration by taking advantage of the completely managed solution that Amazon ECS Anywhere provides. Amazon ECS Anywhere can help you meet compliance requirements and scale your business while retaining your on-premises investments.  
  
For a hands-on workshop about Amazon ECS Anywhere, see [Amazon ECS Workshop: ECS Anywhere](https://www.ecsworkshop.com/ecsanywhere/).

## AWS Systems Manager

By using AWS Systems Manager, you have visibility and control of your infrastructure on AWS. Systems Manager provides a unified user interface that you can use to view operational data from multiple AWS services and automate operational tasks across your AWS resources.  
  
With Systems Manager, you can group resources—such as Amazon Elastic Compute Cloud (Amazon EC2) instances, Amazon Simple Storage Service (Amazon S3) buckets, or Amazon Relational Database Service (Amazon RDS) instances—by application. You can also view operational data for monitoring and troubleshooting, and take action on your groups of resources.  
  
Systems Manager is design to simplify resource and application management, reduce the time needed to detect and resolve operational problems, and make it easier to operate and manage your infrastructure securely at scale.  
  
The following diagram shows how some Systems Manager capabilities perform actions on your resources. The diagram doesn't cover all capabilities. Each numbered interaction is described after the diagram.

1. Access Systems Manager: Use one of the available options for accessing Systems Manager, such as the AWS Management Console or the AWS Command Line Interface (AWS CLI).
2. Choose a Systems Manager capability: Determine which capability can help you with the action you want to perform on your resources. The diagram shows only a few of the capabilities that IT administrators and DevOps personnel use to manage their applications and resources.
3. Verification and processing: Systems Manager verifies that your AWS Identity and Access Management (IAM) user, group, or role has the needed permissions to perform the action that you specified. If the target of your action is a managed node, the Systems Manager Agent (SSM Agent) that runs on the node performs the action. For other types of resources, Systems Manager performs the specified action or communicates with other AWS services to perform the action on behalf of Systems Manager.
4. Reporting: Systems Manager, the SSM Agent, and other AWS services that performed an action on behalf of Systems Manager report their status. Systems Manager can send status details to other AWS services, if configured.
5. Systems Manager operations management capabilities: If you enable Systems Manager operations management capabilities—such as Explorer, OpsCenter, and Incident Manager—they can aggregate operations data or create artifacts in response to events or errors with your resources. These artifacts include operational work items (OpsItems) and incidents. The operations management capabilities from Systems Manager provide both operational insight into your applications and resources, and automated remediation solutions to help troubleshoot problems.

For more resources about Systems Manager, see the following:

* For a tutorial about how to use Systems Manager Run Command, see [Remotely Run Commands on an EC2 Instance with AWS Systems Manager](https://aws.amazon.com/getting-started/hands-on/remotely-run-commands-ec2-instance-systems-manager/).
* For a hands-on workshop about using Systems Manager, see [AWS Management and Governance Tools Workshop: AWS Systems Manager.](https://mng.workshop.aws/ssm.html)
* For more general information about Systems Manager, see [What is AWS Systems Manager?](https://docs.aws.amazon.com/systems-manager/latest/userguide/what-is-systems-manager.html)

## AWS Backup

You can use AWS Backup to centralize and automate data protection across AWS services and hybrid workloads. AWS Backup offers a cost-effective, fully managed, policy-based service that is designed to simplify data protection at scale.  
  
AWS Backup also helps you support your regulatory compliance or business policies for data protection. Together with AWS Organizations, you can use AWS Backup to centrally deploy data protection policies to configure, manage, and govern your backup activity across your company’s AWS accounts and resources. Supported resources include the following:

* EC2 instances
* Applications that are supported by Windows Volume Shadow Copy Service (VSS)—including Windows Server, Microsoft SQL Server, and Microsoft Exchange Server—on Amazon EC2
* Amazon Elastic Block Store (Amazon EBS) volumes
* S3 buckets
* Amazon RDS databases, including Amazon Aurora clusters
* Amazon DynamoDB tables
* Amazon Neptune databases
* Amazon DocumentDB (with MongoDB compatibility) databases
* Amazon Elastic File System (Amazon EFS) file systems
* Amazon FSx for NetApp ONTAP file systems
* Amazon FSx for Lustre file systems
* Amazon FSx for Windows File Server file systems
* Amazon FSx for OpenZFS file systems
* AWS Storage Gateway volumes
* VMware workloads on premises, on Amazon Outposts, and in VMware Cloud on AWS

For more information, see [What is AWS Backup?](https://docs.aws.amazon.com/aws-backup/latest/devguide/whatisbackup.html)

Add a Post

Filter:                                                           Sort:                                                           

[discussion](https://courses.edx.org/courses/course-v1:AWS+AWS-ARCH-1+3T2022/discussion/forum/03d0b2e2e945bd0d090d513e9dad8ea653a8c2cc/threads/6629dfac89a40604ce9f2c82)

[aws](https://courses.edx.org/courses/course-v1:AWS+AWS-ARCH-1+3T2022/discussion/forum/03d0b2e2e945bd0d090d513e9dad8ea653a8c2cc/threads/6629dfac89a40604ce9f2c82)

[architecture](https://courses.edx.org/courses/course-v1:AWS+AWS-ARCH-1+3T2022/discussion/forum/03d0b2e2e945bd0d090d513e9dad8ea653a8c2cc/threads/6629dfac89a40604ce9f2c82)

**[1comments](https://courses.edx.org/courses/course-v1:AWS+AWS-ARCH-1+3T2022/discussion/forum/03d0b2e2e945bd0d090d513e9dad8ea653a8c2cc/threads/6629dfac89a40604ce9f2c82)**

[discussion](https://courses.edx.org/courses/course-v1:AWS+AWS-ARCH-1+3T2022/discussion/forum/03d0b2e2e945bd0d090d513e9dad8ea653a8c2cc/threads/65fbbc5baafa00049d867574)

[Simple enhance](https://courses.edx.org/courses/course-v1:AWS+AWS-ARCH-1+3T2022/discussion/forum/03d0b2e2e945bd0d090d513e9dad8ea653a8c2cc/threads/65fbbc5baafa00049d867574)

[- Get at least 2 AZ when uses RDS - use more effective security](https://courses.edx.org/courses/course-v1:AWS+AWS-ARCH-1+3T2022/discussion/forum/03d0b2e2e945bd0d090d513e9dad8ea653a8c2cc/threads/65fbbc5baafa00049d867574)

**[1comments](https://courses.edx.org/courses/course-v1:AWS+AWS-ARCH-1+3T2022/discussion/forum/03d0b2e2e945bd0d090d513e9dad8ea653a8c2cc/threads/65fbbc5baafa00049d867574)**

[discussion](https://courses.edx.org/courses/course-v1:AWS+AWS-ARCH-1+3T2022/discussion/forum/03d0b2e2e945bd0d090d513e9dad8ea653a8c2cc/threads/65e78af8aafa00049d85319c)

[resiliency](https://courses.edx.org/courses/course-v1:AWS+AWS-ARCH-1+3T2022/discussion/forum/03d0b2e2e945bd0d090d513e9dad8ea653a8c2cc/threads/65e78af8aafa00049d85319c)

[A few ideas: 1. NAT gateway and EC2 instances are in a single AZ. If the configuration allows, split the ec2 workloads between AZs, or if they are monolithic traditional servers, use static scaling groups so that the ec2 instances can move between AZs as needed. Have a NAT gateway in each AZ. 2. RDS instance should have a 2nd AZ replica or read replica. 3. VPN connection has no redundancy. A DirectConnect + VPN connection would work, or, if there is not high bandwidth requirements, 2 VPN connections (assuming two different internet connections at the customer's premises). 4. Depending on what is being transferred between on-prem and AWS, a Storage Gateway could cache files locally and asynchronously sync changes to the AWS side. 5. AWS Backup or some other backup system to allow for backup and recovery. 6. AWS Systems Manager and Operations Manager to collect logs from on-prem and AWS data sources and create a single management dashboard.](https://courses.edx.org/courses/course-v1:AWS+AWS-ARCH-1+3T2022/discussion/forum/03d0b2e2e945bd0d090d513e9dad8ea653a8c2cc/threads/65e78af8aafa00049d85319c)

**[1comments](https://courses.edx.org/courses/course-v1:AWS+AWS-ARCH-1+3T2022/discussion/forum/03d0b2e2e945bd0d090d513e9dad8ea653a8c2cc/threads/65e78af8aafa00049d85319c)**

[discussion](https://courses.edx.org/courses/course-v1:AWS+AWS-ARCH-1+3T2022/discussion/forum/03d0b2e2e945bd0d090d513e9dad8ea653a8c2cc/threads/63d6248d3caa5604b7e9e8d6)

[Hybrid Architecture Recommendation for Resiliency](https://courses.edx.org/courses/course-v1:AWS+AWS-ARCH-1+3T2022/discussion/forum/03d0b2e2e945bd0d090d513e9dad8ea653a8c2cc/threads/63d6248d3caa5604b7e9e8d6)

[In re-architecting a Hybrid AWS system for resilience, the flowing elements must be considered. - Server resiliency - Database resiliency - Network Resiliency & - Application resiliency We are not given a lot of information at this stage such as what particular services are being delivered by this infrastructure. However a few give aways based on the architecture are: - There is limited traffic between the customers data center and the internet. Resources requiring access from the web are hosted in the cloud. - It seems from the architecture that several services are still hosted on prem and only some services are In the cloud. - There is still a significant number of servers on-premises. Below are some architectural consideration recommendations: Server & DB resiliency – This can be achieved by hosting a secondary RDS instance in another availability zone as well as secondary instances of the EC2 server. This will operate out of 2 VPCs. The RDS DB will be backed up to S3 Storage with intelligent tiering class to automatically determine what data to be archived and at what frequency. Data storage tiering could be manually setup depending on access patterns, regulatory requirements, and other usage considerations. Multiple Availability Zones (AZ) setup – This is recommended for hosting primary and secondary RDS instances and EC2 servers Network Resiliency – For network resiliency, the following architectural considerations are recommended: • AWS Direct Connect – In view of the number of on prem servers, the AWS Direct Connect is recommended to guarantee thorough for access and communication between on prem and cloud resources. Specifically, to guarantee DB write and read throughput, the AWS Direct Connect is recommended. • A Transit Gateway – This is required to support multi-AZ setup and enable communication between on-prem and cloud resources over the dedicated link. • NAT Gateway – This will allow Instances in the VPCs to establish a connection through the internet for internet-based users accessing the cloud-based resources. Also, the NAT gateway will allow Instances in one VPC to connect to the other VPC. • Internet Gateway – this allows internet-based resources connect to the cloud resources and vice versa. • AWS VPN – Communication through the public internet must be secure using AWS VPN resources. Application level resiliency can be achieved by backing up application files via AWS Backup Monitoring & service management - All cloud resources can be monitored via the AWS Cloudwatch. However to provide a central a signle montoring and service management interface for the customer, the AWS Systems Manager is recommended. AWS SM is a secure end-to-end management solution that can provide monitoring, management, and visibility for resources both on AWS cloud and on premises. By installing the SM agent on on prem and cloud resources, essential polls and management triggers can be set on each of the nodes/resources.](https://courses.edx.org/courses/course-v1:AWS+AWS-ARCH-1+3T2022/discussion/forum/03d0b2e2e945bd0d090d513e9dad8ea653a8c2cc/threads/63d6248d3caa5604b7e9e8d6)

**[2comments (2 unread comments)](https://courses.edx.org/courses/course-v1:AWS+AWS-ARCH-1+3T2022/discussion/forum/03d0b2e2e945bd0d090d513e9dad8ea653a8c2cc/threads/63d6248d3caa5604b7e9e8d6)**

[discussion](https://courses.edx.org/courses/course-v1:AWS+AWS-ARCH-1+3T2022/discussion/forum/03d0b2e2e945bd0d090d513e9dad8ea653a8c2cc/threads/653b93ab756c0804962860ba)

[My optimization recommendations for improving resiliency](https://courses.edx.org/courses/course-v1:AWS+AWS-ARCH-1+3T2022/discussion/forum/03d0b2e2e945bd0d090d513e9dad8ea653a8c2cc/threads/653b93ab756c0804962860ba)

[1) use 2 vpn connection with 2 different internet providers 2) use aws backup 3) use multi az rds db 4) use multi-region multi-az active/passive deployment 5) ASG with ALB for ec2 6) use waf, guard duty](https://courses.edx.org/courses/course-v1:AWS+AWS-ARCH-1+3T2022/discussion/forum/03d0b2e2e945bd0d090d513e9dad8ea653a8c2cc/threads/653b93ab756c0804962860ba)

**[1comments](https://courses.edx.org/courses/course-v1:AWS+AWS-ARCH-1+3T2022/discussion/forum/03d0b2e2e945bd0d090d513e9dad8ea653a8c2cc/threads/653b93ab756c0804962860ba)**

[discussion](https://courses.edx.org/courses/course-v1:AWS+AWS-ARCH-1+3T2022/discussion/forum/03d0b2e2e945bd0d090d513e9dad8ea653a8c2cc/threads/64ba4cf40a891e04b20aeaea)

[task - week 3](https://courses.edx.org/courses/course-v1:AWS+AWS-ARCH-1+3T2022/discussion/forum/03d0b2e2e945bd0d090d513e9dad8ea653a8c2cc/threads/64ba4cf40a891e04b20aeaea)

[I would deploy my fleet of instances in one more AZ. Also, I would add a System Manager to be able to manage my instances by mitigating human errors and more automatically.](https://courses.edx.org/courses/course-v1:AWS+AWS-ARCH-1+3T2022/discussion/forum/03d0b2e2e945bd0d090d513e9dad8ea653a8c2cc/threads/64ba4cf40a891e04b20aeaea)

**[1comments](https://courses.edx.org/courses/course-v1:AWS+AWS-ARCH-1+3T2022/discussion/forum/03d0b2e2e945bd0d090d513e9dad8ea653a8c2cc/threads/64ba4cf40a891e04b20aeaea)**

[discussion](https://courses.edx.org/courses/course-v1:AWS+AWS-ARCH-1+3T2022/discussion/forum/03d0b2e2e945bd0d090d513e9dad8ea653a8c2cc/threads/6499de362a472d04a582e47f)

[Systems Manager operations management capabilities](https://courses.edx.org/courses/course-v1:AWS+AWS-ARCH-1+3T2022/discussion/forum/03d0b2e2e945bd0d090d513e9dad8ea653a8c2cc/threads/6499de362a472d04a582e47f)

[If enable Systems Manager operations management capabilities—such as Explorer, OpsCenter, and Incident Manager—they can aggregate operations data or create artifacts in response to events or errors with resources. These artifacts include operational work items (OpsItems) and incidents. The operations management capabilities from Systems Manager provide both operational insight into applications and resources, and automated remediation solutions to help troubleshoot problems](https://courses.edx.org/courses/course-v1:AWS+AWS-ARCH-1+3T2022/discussion/forum/03d0b2e2e945bd0d090d513e9dad8ea653a8c2cc/threads/6499de362a472d04a582e47f)

**[1comments](https://courses.edx.org/courses/course-v1:AWS+AWS-ARCH-1+3T2022/discussion/forum/03d0b2e2e945bd0d090d513e9dad8ea653a8c2cc/threads/6499de362a472d04a582e47f)**

[discussion](https://courses.edx.org/courses/course-v1:AWS+AWS-ARCH-1+3T2022/discussion/forum/03d0b2e2e945bd0d090d513e9dad8ea653a8c2cc/threads/649940a50a891e04b208215b)

[Need AWS System manager and AWS backup service](https://courses.edx.org/courses/course-v1:AWS+AWS-ARCH-1+3T2022/discussion/forum/03d0b2e2e945bd0d090d513e9dad8ea653a8c2cc/threads/649940a50a891e04b208215b)

[With that diagram, we need to use AWS System Manager and AWS backup service for all system.](https://courses.edx.org/courses/course-v1:AWS+AWS-ARCH-1+3T2022/discussion/forum/03d0b2e2e945bd0d090d513e9dad8ea653a8c2cc/threads/649940a50a891e04b208215b)

### Reading 3.6: Architecture Optimizations for Week 3

## Level 2 headings may be created by course providers in the future.

Bookmark this page

In this reading, you will find further information about the topics that Morgan and Raf talked in the video where they both played solutions architects.  
  
The following diagram is a more detailed view of the solution that was covered this week.

## Disaster recovery

Morgan mentioned that because this week’s customer is an enterprise company, they could consider designing a disaster recovery (DR) plan for this workload.  
  
The DR strategies that are available to you within AWS can be broadly categorized into four approaches, which range from the low cost and low complexity of making backups to more complex strategies that use multiple active Regions. Active/passive strategies use an active site (such as an AWS Region) to host the workload and serve traffic. The passive site (such as a different AWS Region) is used for recovery. The passive site doesn’t actively serve traffic until a failover event is triggered.  
  
The following diagram outlines the different approaches to disaster recovery on AWS.

## Backup and restore

Backup and restore is a suitable approach for mitigating against data loss or corruption. This approach can also be used to mitigate against a regional disaster by replicating data to other AWS Regions, or to mitigate a lack of redundancy for workloads that are deployed to a single Availability Zone. In addition to data, you must redeploy the infrastructure, configuration, and application code in the recovery Region. To enable infrastructure to be redeployed quickly without errors, you should always deploy by using infrastructure as code (IaC), with services such as AWS CloudFormation or the AWS Cloud Development Kit (AWS CDK). Without IaC, it might be complex to restore workloads in the recovery Region, which can lead to increased recovery times and possibly exceed your Recovery Time Objective (RTO). In addition to backing up your user data, back up your code and configurations—including [Amazon Machine Images (AMIs)](https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/AMIs.html) that you use to create Amazon Elastic Compute Cloud (Amazon EC2) instances. You can use [AWS CodePipeline](http://aws.amazon.com/codepipeline) to automate the redeployment of your application code and configurations.

## Pilot light

With the pilot light approach, you replicate your data from one Region to another, and you provision a copy of your core workload infrastructure. Resources that are needed to support data replication and backup—such as databases and object storage—are always on. Other elements (such as application servers) are loaded with application code and configurations, but are turned off. They are only used during testing or when DR failover is invoked. In the cloud, you have the flexibility to deprovision resources when you don’t need them, and provision them when you do. A best practice for a resource that’s turned off is to not deploy the resource, and then create the configuration and capabilities to deploy it (turn on) when needed. Unlike the backup and restore approach, your core infrastructure is always available. Thus, you always have the option to quickly provision a full scale production environment by turning on and scaling out your application servers.

## Warm standby

The warm standby approach involves provisioning a scaled down—but fully functional—copy of your production environment in another Region. This approach extends the pilot-light concept and decreases the time to recovery because your workload is always-on in another Region. With this approach, you can perform testing or implement continuous testing more easily, which can increase your confidence in your ability to recover from a disaster.

## Multi-site active/active

You can run your workload simultaneously in multiple Regions as part of either a multi-site active/active strategy or a hot standby active/passive strategy. A multi-site active/active approach serves traffic from all Regions where it’s deployed. In contrast, hot standby approach serves traffic from only a single Region, and the other Regions are used only for DR. With a multi-site active/active approach, users can access your workload in any Regions where it’s deployed. This approach to DR is the most complex and most costly. However, it can reduce your recovery time to near zero for most disasters, with the correct technology choices and implementation. (Note that data corruption might need to rely on backups, which usually results in a non-zero recovery point.) Hot standby uses an active/passive configuration, where users are directed to only a single region and DR Regions don’t take traffic. Most customers find that if they’re going to stand up a full environment in the second Region, it makes sense to use it in an active/active approach. Alternatively, if you don’t want to use both Regions to handle user traffic, then warm standby offers an approach that’s more economical and operationally less complex.  
  
For more information about DR, see the [Disaster Recovery of Workloads on AWS: Recovery in the Cloud](https://docs.aws.amazon.com/whitepapers/latest/disaster-recovery-workloads-on-aws/disaster-recovery-workloads-on-aws.html) whitepaper.

## AWS Direct Connect with AWS VPN for failover

Raf commented that to make the connection to AWS more resilient, the customer can consider using Amazon Site-to-Site VPN as a failover for AWS Direct Connect so that the connection is redundant. This setup would protect the customer if the AWS Direct Connect connection became unavailable. With this setup, the customer would be able to fail over to the virtual private network (VPN) connection and remain connected to their AWS resources.  
  
For more information about how to set up Site-to-Site VPN as a failover for Direct Connect, see [How do I configure Direct Connect and VPN failover with Transit Gateway?](https://aws.amazon.com/premiumsupport/knowledge-center/dx-configure-dx-and-vpn-failover-tgw/)

## Automatic scaling for containers

Morgan commented that this week’s customer needs to look into how they will scale their containers. When the customer uses Amazon Elastic Container Service (Amazon ECS), they will need to think about scaling both their underlying EC2 cluster and the containers themselves.

### Scaling the cluster

Amazon ECS can manage the scaling of EC2 instances that are registered to your cluster. This capability is referred to as *Amazon ECS cluster auto scaling*, and is performed by an Amazon ECS Auto Scaling group capacity provider that has managed scaling turned on.  
  
When you use an Auto Scaling group capacity provider with managed scaling, Amazon ECS creates two custom Amazon CloudWatch metrics, and a target tracking scaling policy that attaches to your Auto Scaling group. Amazon ECS then manages the scale-in and scale-out actions of the Auto Scaling group based on the load that your tasks put on your cluster.

### Scaling the containers

*Automatic scaling* is the ability to increase or decrease the desired count of tasks in your Amazon ECS service automatically. Amazon ECS uses the Application Auto Scaling service to provide this functionality.  
  
Your Amazon ECS service can be optionally configured automatically scale its desired count of tasks in your Amazon ECS service, either up or down, in response to CloudWatch alarms.  
  
Amazon ECS Service Auto Scaling supports the following types of scaling policies:

* Target tracking scaling policies (Recommended): Increase or decrease the number of tasks that your service runs, based on a target value for a specific metric. This scaling approach is similar to the way that your thermostat maintains the temperature of your home. You select the temperature, and the thermostat manages the temperature.
* Step scaling policies: Increase or decrease the number of tasks that your service runs, based on a set of scaling adjustments, which are also known as step adjustments. These adjustments vary based on the size of the alarm breach.

For more resources about automatic scaling and Amazon ECS, see the following:

* For more information about how to scale an ECS cluster, see [Amazon ECS cluster Auto Scaling](https://docs.aws.amazon.com/AmazonECS/latest/developerguide/cluster-auto-scaling.html).
* For a hands-on tutorial about how to scale an ECS cluster, see the [Amazon ECS Workshop: Deploy ECS Cluster Auto Scaling](https://ecsworkshop.com/capacity_providers/ec2/).
* For more information about service auto scaling, see [Service auto scaling](https://docs.aws.amazon.com/AmazonECS/latest/developerguide/service-auto-scaling.html) in the *Amazon ECS Developer Guide*.

## Automatic scaling for Amazon RDS

Morgan commented that this week’s customer should look into RDS storage autoscaling.  
  
When you enable storage autoscaling, Amazon RDS automatically scales up your storage when it detects that you are running out of free database space.  
  
  
If your workload is unpredictable, you can enable storage autoscaling for an RDS DB instance. To do so, you can use the Amazon RDS console, the Amazon RDS API, or the AWS Command Line Interface (AWS CLI).  
  
For more information about Amazon RDS storage autoscaling, see [Managing capacity automatically with Amazon RDS storage autoscaling.](https://docs.aws.amazon.com/AmazonRDS/latest/UserGuide/USER_PIOPS.StorageTypes.html#USER_PIOPS.Autoscaling)

## Amazon S3 Intelligent-Tiering

Raf commented that this customer could also optimize for cost by using S3 Intelligent-Tiering on the S3 bucket where their data is being stored.  
  
S3 Intelligent-Tiering is the only cloud storage class that delivers automatic storage cost savings when data access patterns change, thus reducing performance impact or operational overhead. The Amazon S3 Intelligent-Tiering storage class is designed to optimize storage costs by automatically moving data to the most cost-effective access tier when access patterns change. For a small monthly object monitoring and automation charge, S3 Intelligent-Tiering monitors access patterns and automatically moves objects that haven’t been accessed to lower-cost access tiers.  
  
S3 Intelligent-Tiering is a good storage class for data with unknown, changing, or unpredictable access patterns—independent of object size or retention period. You can use S3 Intelligent-Tiering as the default storage class for virtually any workload, especially data lakes, data analytics, new applications, and user-generated content.  
  
For more information about S3 Intelligent-Tiering, see [Amazon S3 Intelligent-Tiering storage class](https://aws.amazon.com/s3/storage-classes/intelligent-tiering/).

### Reading 4.1: Multi-account Strategies

## Level 2 headings may be created by course providers in the future.

Bookmark this page

This week’s customer is currently operating everything for all the clients that they support out of one AWS account. This situation isn’t ideal for a company that wants to organize and scale their usage of AWS as their usage grows.  
  
The practice of using multiple accounts has many advantages. To summarize, you can group workloads based on business purposes and ownership, centralize logging, and constrain access to sensitive data. You can also limit the scope of impact from adverse events, manage costs better, and distribute AWS service quotas and API request rate limits.  
  
Businesses that are starting to adopt AWS, expanding their footprint on AWS, or planning to enhance an established AWS environment need to ensure they have a [foundation on AWS](https://docs.aws.amazon.com/whitepapers/latest/establishing-your-cloud-foundation-on-aws/welcome.html) for their cloud environment. One important aspect of their foundation is to organize their AWS environment by following a multi-account strategy.  
  
By using multiple AWS accounts to help isolate and manage your business applications and data, you can optimize across most of the [AWS Well-Architected Framework](http://aws.amazon.com/architecture/well-architected/) pillars—including operational excellence, security, reliability, and cost optimization.

## Group workloads based on business purpose and ownership

You can group workloads with a common business purpose into distinct accounts. As a result, you can align the ownership and decision-making of those accounts. You can also avoid dependencies and conflicts with how workloads in other accounts are secured and managed.  
  
Different business units or product teams might have different processes. Depending on your overall business model, you might choose to isolate distinct business units or subsidiaries in different accounts. By isolating business units, they can operate with greater decentralized control—while still retaining the ability to provide overarching guardrails. This approach might also ease divestment of those units over time.  
  
Guardrails are governance rules for security, operations, and compliance that you can define and apply to align with your overall requirements.  
  
If you acquire a business that already operates in AWS, you can move the associated accounts into your existing organization intact. This movement of accounts can be an initial step toward integrating acquired services into your standard account structure.

## Apply distinct security controls by environment

Workloads often have distinct security profiles that require separate control policies and mechanisms to support them. For example, it’s common to apply different policies for security and operations to the non-production and production environments of a given workload. If you use separate accounts for the non-production and production environments, the resources and data that make up a workload environment are separated from other environments and workloads by default.

## Constrain access to sensitive data

When you limit sensitive data stores to an account that is built to manage it, you can more easily constrain the number of people and processes that can access and manage the data store. This approach simplifies the process of achieving least-privilege access. Limiting access at the coarse-grained level of an account helps contain exposure to highly sensitive data.  
For example, by designating a set of accounts to house publicly accessible Amazon Simple Storage Service (Amazon S3) buckets, you can implement policies that expressly forbid all other accounts from making S3 buckets publicly available.

## Promote innovation and agility

At AWS, we refer to your technologists as [builders](http://aws.amazon.com/campaigns/build-on-aws/) because they’re responsible for building value by using AWS products and services. Your builders might represent diverse roles, such as application developers, data engineers, data scientists, data analysts, security engineers, and infrastructure engineers.  
  
In the early stages of a workload’s lifecycle, you can help promote innovation by providing your builders with separate accounts in support of experimentation, development, and early testing. These environments often provide greater freedom than more tightly controlled production-like test and production environments. They do so by providing broader access to AWS services while also using guardrails that help prohibit access to (and the use of) sensitive and internal data.

* Sandbox accounts: Typically disconnected from your enterprise services and don’t provide access to your internal data. However, they offer the greatest freedom for experimentation.
* Development accounts: Typically provide limited access to your enterprise services and development data. However, they can more readily support day-to-day experimentation with your enterprise-approved AWS services, formal development, and early testing work.

In both cases, we recommend security guardrails and cost budgets so that you limit risks and proactively manage costs.  
You can support later stages of the workload lifecycle by using distinct test and production accounts for workloads or groups of related workloads. By having an environment for each set of workloads, owning teams can move faster by reducing dependencies on other teams and workloads, and by also minimizing the impact of changes.

## Limit scope of impact from adverse events

An AWS account applies boundaries for security, access, and billing boundaries to your AWS resources. These boundaries can help you achieve the independence and isolation of resources. By design, all resources that are provisioned within an account are logically isolated from resources that are provisioned in other accounts—even within your own AWS environment.  
  
This isolation boundary provides a way to limit the risks of an application-related issue, misconfiguration, or malicious actions. If an issue occurs within one account, impacts to workloads contained in other accounts can be either reduced or eliminated.

## Support multiple IT operating models

Organizations often have multiple IT operating models, or ways that they divide responsibilities among parts of the organization to deliver their application workloads and platform capabilities. The following diagram shows three example operating models:

### Example operating models

In the *Traditional Ops* model, teams who own custom and commercial off-the-shelf (COTS) applications are responsible for engineering their applications, but not for their production operations. A cloud platform engineering team is responsible for engineering the underlying platform capabilities. A separate cloud operations team is responsible for the operations of both applications and platform.  
  
In the *CloudOps model*, application teams are also responsible for production operations of their applications. In this model, a common cloud platform engineering team is responsible for both the engineering and operations of the underlying platform capabilities.  
  
In the *DevOps model*, the application teams take on the additional responsibilities of engineering and operating platform capabilities that are specific to their applications. A cloud platform engineering team is responsible for the engineering and operations of shared platform capabilities that are used by multiple applications.  
  
As a practice, IT Service Management (ITSM) is a common element across all of the models. Your overall goals and requirements for ITSM might not change across these models. However, the responsible individuals and solutions for meeting those goals and requirements can vary, depending on the model.  
  
Given the implications of centralized operations versus more distributed operational responsibilities, you might benefit from establishing separate groups of accounts in support of different operating models. By using separate accounts, you can apply distinct governance and operational controls that are appropriate for each of your operating models.

## Manage costs

An account is the default way that AWS costs are allocated. By using different accounts for different business units and groups of workloads, you can more easily report, control, forecast, and budget your cloud expenditures.  
  
In addition to cost reporting at the account level, AWS has built-in support to consolidate and report costs across your entire set of accounts. When you require fine-grained cost allocation, you can apply cost allocation tags to individual resources in each of your accounts.

## Distribute AWS service quotas and API request rate limits

AWS service quotas (also known as *limits*) are the maximum number of service resources or operations that apply to an account. For example, a service quota could be the number of S3 buckets that you can create for each account.  
  
You can use the Service Quotas service to help protect you from unexpected or excessive provisioning of AWS resources, and from malicious actions that could dramatically impact your AWS costs.  
  
AWS services can also throttle (or limit) the rate of requests that are made to their API operations.  
  
Because service quotas and request rate limits are allocated for each account, use separate accounts for workloads to help distribute the potential impact of the quotas and limits.  
  
For more information about multi-account strategies, see the following:

* For more information about building cloud foundations, see the [Establishing Your Cloud Foundation on AWS](https://docs.aws.amazon.com/whitepapers/latest/establishing-your-cloud-foundation-on-aws/welcome.html) whitepaper.
* For more information about organizing AWS accounts, see the [Organizing Your AWS Environment Using Multiple Accounts](https://docs.aws.amazon.com/whitepapers/latest/organizing-your-aws-environment/organizing-your-aws-environment.html) whitepaper.

<https://learning.edx.org/course/course-v1:AWS+AWS-ARCH-1+3T2022/block-v1:AWS+AWS-ARCH-1+3T2022+type@sequential+block@183aa53f87804ce284e2d589e8563bac/block-v1:AWS+AWS-ARCH-1+3T2022+type@vertical+block@04c635bbb53b4d3c83ab5d3fa1d8f08c>

1. AWS IAM provides, among other things,
2. IAM users,
3. IAM groups,
4. and IAM roles.
5. IAM users
6. and roles are IAM entities
7. that provide credentials
8. and permissions for access.
9. **IAM groups are collections of users.**
10. Users have permanent credentials
11. and roles provide a temporary credential
12. or a URL for sign in.
13. This operation is called "assume role".
14. Using roles is important
15. because when you have multiple accounts,
16. you would like to prevent
17. replicating IAM users on every account.
18. That's where IAM roles come i

### IAM Roles, Trust Relationships, and Permissions

## Level 2 headings may be created by course providers in the future.

Bookmark this page This week, Raf suggested that the customer use AWS Identity and Access Management (IAM) roles to support the multi-account strategy. He suggested using IAM roles to facilitate cross-account access for the other accounts that are used in this solution. The roles provide cross-account access without needing to create static credentials for each account, such as IAM users.

## IAM roles

An IAM role is an identity that you can create in your account, and it has specific permissions. An IAM role has some similarities to an IAM user. Roles and users are both AWS identities with permissions policies that determine what the identity can or can’t do in AWS. However, instead of being uniquely associated with one person, a role can be assumed by anyone who needs it. Also, a role doesn’t have standard long-term credentials (such as a password or access keys) associated with it. Instead, when you assume a role, it provides you with temporary security credentials for your role session.  
  
Roles can be used by the following entities:

* An IAM user in the same AWS account as the role
* An IAM user in a different AWS account than the role
* An AWS service, such as Amazon Elastic Compute Cloud (Amazon EC2)
* An external user who was authenticated by an external identity provider (IdP) service that’s compatible with Security Assertion Markup Language (SAML) 2.0, OpenID Connect (OIDC), or a custom-built identity broker.

When you work with AWS, you should also be familiar with the following terms, which are related to IAM roles.

### AWS service role

An AWS service role is a role that a service assumes to perform actions in your account on your behalf. When you set up some AWS service environments, you must define a role for the service to assume. This service role must include all the permissions that the service needs to access the AWS resources that it must work with. Service roles vary from service to service, but many allow you to choose your permissions if you meet the documented requirements for that service. You can create, modify, and delete a service role from within IAM.

### AWS service-linked role

An AWS service-linked role is a unique type of service role that’s linked directly to an AWS service. Service-linked roles are predefined by the service, and they include all permissions that the service needs to call other AWS services on your behalf. The linked service also defines how you create, modify, and delete a service-linked role. A service might automatically create or delete the role. It might allow you to create, modify, or delete the role as part of a wizard or process in the service. Or it might require you to use IAM to create or delete the role. Regardless of the method, service-linked roles make setting up a service easier because you don't need to manually add the necessary permissions.

### Delegation

Delegation is the granting of permissions to someone to allow access to resources that you control. Delegation involves setting up a trust between two accounts. The first account is the account that owns the resource (the trusting account). The second account is the account that contains the users that need to access the resource (the trusted account). The trusted and trusting accounts can be any of the following:

* The same account
* Separate accounts that are both under your organization's control
* Two accounts owned by different organizations

To delegate permissions to access a resource, you create an IAM role in the trusting account that has two [policies](https://docs.aws.amazon.com/IAM/latest/UserGuide/id_roles_terms-and-concepts.html#term_policy) attached. The *permissions policy* grants the user of the role the needed permissions to carry out the intended tasks on the resource. The *trust policy* specifies which trusted account members are allowed to assume the role.  
  
When you create a trust policy, you can’t specify a wildcard (\*) as a principal.  
  
The trust policy is attached to the role in the trusting account, and comprises half of the permissions. The other half is a permissions policy that’s attached to the user in the trusted account. The permissions policy allows that user to switch to (or assume) the role. A user who assumes a role temporarily gives up his or her own permissions, and instead takes on the role’s permissions. When the user exits (or stops using) the role, their original user permissions are restored. An additional parameter that’s called *external ID* can help ensure the secure use of roles between accounts that are not controlled by the same organization.

### Federation

Federation is the creation of a trust relationship between an external IdP and AWS. Users can sign in to a web identity provider, such as Login with Amazon, Facebook, Google, or any IdP that is compatible with OIDC. Users can also sign in to an enterprise identity system that’s compatible with SAML 2.0, such as Microsoft Active Directory Federation Services. When you use OIDC and SAML 2.0 to configure a trust relationship between these external IdPs and AWS, the user is assigned to an IAM role. The user also receives temporary credentials that allow the user to access your AWS resources.

### Federated user

Instead of creating an IAM user, you can use existing identities from AWS Directory Service, your enterprise user directory, or a web identity provider. These identities are known as *federated users*. AWS assigns a role to a federated user when access is requested through an identity provider.

### Trust policy

A trust policy is a JSON policy document where you define the principals that you *trust* to assume the role. A role trust policy is a required, resource-based policy that’s attached to a role in IAM. The principals that you can specify in the trust policy include users, roles, accounts, and services.

### Permissions boundary

A permissions boundary is an advanced feature where you use policies to limit the maximum permissions that an identity-based policy can grant to a role. You can’t apply a permissions boundary to a service-linked role.

### Principal

A principal is an entity in AWS that can perform actions and access resources. A principal can be an AWS account root user, an IAM user, or a role.  
  
For more information, see [Roles terms and concepts.](https://docs.aws.amazon.com/IAM/latest/UserGuide/id_roles_terms-and-concepts.html)

**aws.amazon.com/organizations**

**organizations Accounts**

[**https://learning.edx.org/course/course-v1:AWS+AWS-ARCH-1+3T2022/block-v1:AWS+AWS-ARCH-1+3T2022+type@sequential+block@183aa53f87804ce284e2d589e8563bac/block-v1:AWS+AWS-ARCH-1+3T2022+type@vertical+block@f6c93569ef0e418c9e50e31f6e392b0c**](https://learning.edx.org/course/course-v1:AWS+AWS-ARCH-1+3T2022/block-v1:AWS+AWS-ARCH-1+3T2022+type@sequential+block@183aa53f87804ce284e2d589e8563bac/block-v1:AWS+AWS-ARCH-1+3T2022+type@vertical+block@f6c93569ef0e418c9e50e31f6e392b0c)

[**https://learning.edx.org/course/course-v1:AWS+AWS-ARCH-1+3T2022/block-v1:AWS+AWS-ARCH-1+3T2022+type@sequential+block@183aa53f87804ce284e2d589e8563bac/block-v1:AWS+AWS-ARCH-1+3T2022+type@vertical+block@f62d52f7836549e4b5f5459a1b41a7cd**](https://learning.edx.org/course/course-v1:AWS+AWS-ARCH-1+3T2022/block-v1:AWS+AWS-ARCH-1+3T2022+type@sequential+block@183aa53f87804ce284e2d589e8563bac/block-v1:AWS+AWS-ARCH-1+3T2022+type@vertical+block@f62d52f7836549e4b5f5459a1b41a7cd)

### AWS IAM Identity Center

Raf suggested that this week’s customer use AWS IAM Identity Center (successor to AWS Single Sign-On) to manage single sign-on (SSO) for their AWS accounts. By doing so, the customer can use one place for workforce user and group access, and one place to manage AWS account access.  
  
AWS IAM Identity Center helps you securely create or connect your workforce identities and manage their access centrally across AWS accounts and applications. IAM Identity Center is the recommended approach for workforce authentication and authorization on AWS for organizations of any size and type.

### IAM Identity Center features

### IAM Identity Center includes the following core features: Workforce identities Human users who are members of your organization are also known as*workforce identities* or *workforce users*. You can create workforce users and groups in IAM Identity Center. You can also connect and synchronize to an existing set of users and groups in your own identity source for use across all your AWS accounts and applications. Supported identity sources include Microsoft Active Directory Domain Services, and external identity providers such as Okta Universal Directory or Microsoft Azure AD. Application assignments for SAML applications With application assignments, you can grant your workforce users in IAM Identity Center SSO access to Security Assertion Markup Language (SAML) 2.0 applications, such as Salesforce and Microsoft 365. Your users can access these applications in a single place, without the need for you to set up federation separately. Identity Center enabled applications AWS applications and services—such as Amazon Managed Grafana, Amazon Monitron, and Amazon SageMaker Studio Notebooks—discover and connect to IAM Identity Center automatically to receive sign-in and user directory services. This feature provides users with a consistent SSO experience for these applications, with no additional application configuration. Because the applications share a common view of users, groups, and group membership, users can also have a consistent experience when they share application resources with others. Multi-account permissions With multi-account permissions, you can plan for and centrally implement IAM permissions across multiple AWS accounts at one time, without the need for you to configure each of your accounts manually. You can create fine-grained permissions based on common job functions, or define custom permissions that meet your security needs. You can then assign those permissions to workforce users to control their access over specific accounts. AWS access portal The AWS access portal provides your workforce users with one-click access to all their assigned AWS accounts and cloud applications through a web portal.

formation, see [What is IAM Identity Center?](https://docs.aws.amazon.com/singlesignon/latest/userguide/what-is.html)

For more in Reflecting on AWS Best Practices

### Discussion - Reflecting on AWS Best Practices

**Topic:** Week 4 / Reflecting on AWS Best Practices-Week4

Hide Discussion

Add a Post

Filter:                                                           Sort:                                                           

[discussion](https://courses.edx.org/courses/course-v1:AWS+AWS-ARCH-1+3T2022/discussion/forum/e06fd9c8431a5908716b783ff9bbc0e4e22b1c07/threads/6629e1a189a40604ce9f2c99)

[aws](https://courses.edx.org/courses/course-v1:AWS+AWS-ARCH-1+3T2022/discussion/forum/e06fd9c8431a5908716b783ff9bbc0e4e22b1c07/threads/6629e1a189a40604ce9f2c99)

[architecture](https://courses.edx.org/courses/course-v1:AWS+AWS-ARCH-1+3T2022/discussion/forum/e06fd9c8431a5908716b783ff9bbc0e4e22b1c07/threads/6629e1a189a40604ce9f2c99)

**[1comments](https://courses.edx.org/courses/course-v1:AWS+AWS-ARCH-1+3T2022/discussion/forum/e06fd9c8431a5908716b783ff9bbc0e4e22b1c07/threads/6629e1a189a40604ce9f2c99)**

[discussion](https://courses.edx.org/courses/course-v1:AWS+AWS-ARCH-1+3T2022/discussion/forum/e06fd9c8431a5908716b783ff9bbc0e4e22b1c07/threads/65ea34afaafa00049d855cd4)

[Ideas on structuring accounts.](https://courses.edx.org/courses/course-v1:AWS+AWS-ARCH-1+3T2022/discussion/forum/e06fd9c8431a5908716b783ff9bbc0e4e22b1c07/threads/65ea34afaafa00049d855cd4)

[I'd look at setting up a shared services account, which will hold the settings for AWS Organizations and AWS Identity Manager. We'll setup other accounts based on the best fit for how the organization operates -- based on business unit, data security, as well as prod vs dev. We'll want to reduce the 'blast radius' for any data issues. We'll use IAMs ability to assume roles and trusting accross accounts, that way admins are not permanent admins of every account, but assume the role instead.](https://courses.edx.org/courses/course-v1:AWS+AWS-ARCH-1+3T2022/discussion/forum/e06fd9c8431a5908716b783ff9bbc0e4e22b1c07/threads/65ea34afaafa00049d855cd4)

**[1comments](https://courses.edx.org/courses/course-v1:AWS+AWS-ARCH-1+3T2022/discussion/forum/e06fd9c8431a5908716b783ff9bbc0e4e22b1c07/threads/65ea34afaafa00049d855cd4)**

[discussion](https://courses.edx.org/courses/course-v1:AWS+AWS-ARCH-1+3T2022/discussion/forum/e06fd9c8431a5908716b783ff9bbc0e4e22b1c07/threads/6588bb44752ae404c8cda1af)

[Write a few paragraphs that answer the following questions:](https://courses.edx.org/courses/course-v1:AWS+AWS-ARCH-1+3T2022/discussion/forum/e06fd9c8431a5908716b783ff9bbc0e4e22b1c07/threads/6588bb44752ae404c8cda1af)

[\*Limit scope of impact from adverse events \*Manage costs \*Group workloads based on business purpose and ownership](https://courses.edx.org/courses/course-v1:AWS+AWS-ARCH-1+3T2022/discussion/forum/e06fd9c8431a5908716b783ff9bbc0e4e22b1c07/threads/6588bb44752ae404c8cda1af)

**[2comments (2 unread comments)](https://courses.edx.org/courses/course-v1:AWS+AWS-ARCH-1+3T2022/discussion/forum/e06fd9c8431a5908716b783ff9bbc0e4e22b1c07/threads/6588bb44752ae404c8cda1af)**

[discussion](https://courses.edx.org/courses/course-v1:AWS+AWS-ARCH-1+3T2022/discussion/forum/e06fd9c8431a5908716b783ff9bbc0e4e22b1c07/threads/65e4c7aaaafa00049d85017b)

[multiple AWS accounts, patterns organizing resources, IAM roles for authenticating AWS accounts](https://courses.edx.org/courses/course-v1:AWS+AWS-ARCH-1+3T2022/discussion/forum/e06fd9c8431a5908716b783ff9bbc0e4e22b1c07/threads/65e4c7aaaafa00049d85017b)

[Using multiple AWS accounts to run workloads offers several benefits, enhancing security, simplifying billing, and facilitating resource management. By segmenting workloads across different accounts, organizations can apply the principle of least privilege, minimizing the potential impact of a security breach by isolating resources and permissions. This separation can prevent unauthorized access across different environments, such as development, testing, and production, ensuring that vulnerabilities in one area do not compromise the entire infrastructure. In terms of billing and cost management, multiple accounts allow for clearer tracking of expenses. Organizations can allocate costs more accurately and set individual budget limits per account, making it easier to identify and manage the financial aspects of different projects or departments. Additionally, this setup supports more granular access to billing information, enabling better analysis and optimization of AWS spending. When it comes to organizing resources within AWS accounts, common patterns include environment-based segregation, team or department-based segregation, and project-based segregation. Environment-based segregation involves creating separate accounts for development, testing, and production environments to reduce risks and provide stable operations. Team or department-based segregation assigns different accounts to specific teams or departments, allowing for tailored resource allocation and permission settings. Project-based segregation, on the other hand, involves creating separate accounts for individual projects or applications, which can help in managing lifecycle costs and decommissioning resources when projects end. The use of AWS Identity and Access Management (IAM) roles for authenticating into AWS accounts brings significant benefits, particularly in terms of security and access control. IAM roles provide a secure way to delegate permissions that don’t require sharing security credentials. Instead, trusted entities, such as IAM users, applications, or AWS services, assume roles to obtain temporary security credentials to make AWS API calls. This approach reduces the risk of credentials being compromised and facilitates the principle of least privilege by allowing the assignment of only the necessary permissions for a particular task. Additionally, IAM roles can be assumed by users from other AWS accounts, simplifying cross-account access without the need to create duplicate IAM users, thereby streamlining the management of user permissions across an organization’s AWS environment.](https://courses.edx.org/courses/course-v1:AWS+AWS-ARCH-1+3T2022/discussion/forum/e06fd9c8431a5908716b783ff9bbc0e4e22b1c07/threads/65e4c7aaaafa00049d85017b)

**[1comments](https://courses.edx.org/courses/course-v1:AWS+AWS-ARCH-1+3T2022/discussion/forum/e06fd9c8431a5908716b783ff9bbc0e4e22b1c07/threads/65e4c7aaaafa00049d85017b)**

[discussion](https://courses.edx.org/courses/course-v1:AWS+AWS-ARCH-1+3T2022/discussion/forum/e06fd9c8431a5908716b783ff9bbc0e4e22b1c07/threads/653b9ae4756c08049628612d)

[AWS Multi account Best Practices](https://courses.edx.org/courses/course-v1:AWS+AWS-ARCH-1+3T2022/discussion/forum/e06fd9c8431a5908716b783ff9bbc0e4e22b1c07/threads/653b9ae4756c08049628612d)

[\*\*What are the benefits of using multiple AWS accounts to run workloads? ->\*\* 1. apply fine grained customer managed policies for audit, compliance and billing purpose 2. for greater security 3. . manage quota limits \*\*What are some common patterns for organizing resources in AWS accounts?\*\* 4. segregate by BU and divisions and subdivisions within an organization with OU and sub-ou and applying Service control policy \*\*What is the benefit of using AWS Identity and Access Management (IAM) roles for authenticating into AWS accounts?\*\* 1. no need to have multiple user name / password per account 2. easy to switch and work on different account with different permissions 3. single place to manager access for multiple accounts](https://courses.edx.org/courses/course-v1:AWS+AWS-ARCH-1+3T2022/discussion/forum/e06fd9c8431a5908716b783ff9bbc0e4e22b1c07/threads/653b9ae4756c08049628612d)

**[1comments](https://courses.edx.org/courses/course-v1:AWS+AWS-ARCH-1+3T2022/discussion/forum/e06fd9c8431a5908716b783ff9bbc0e4e22b1c07/threads/653b9ae4756c08049628612d)**

[discussion](https://courses.edx.org/courses/course-v1:AWS+AWS-ARCH-1+3T2022/discussion/forum/e06fd9c8431a5908716b783ff9bbc0e4e22b1c07/threads/64a1634c2a472d04ac8021af)

[Answer about the AWS Best Practices](https://courses.edx.org/courses/course-v1:AWS+AWS-ARCH-1+3T2022/discussion/forum/e06fd9c8431a5908716b783ff9bbc0e4e22b1c07/threads/64a1634c2a472d04ac8021af)

[What are the benefits of using multiple AWS accounts to run workloads? With multi-account permissions, users can plan for and centrally implement IAM permissions across multiple AWS accounts at one time, without the need for users to configure each of their accounts manually. What are some common patterns for organizing resources in AWS accounts? These are the common patterns for organizing resources in AWS accounts: Workforce identities, Application assignments for SAML applications, Identity Center-enabled applications, Multi-account permissions, and AWS access portal. What is the benefit of using AWS Identity and Access Management (IAM) roles for authenticating into AWS accounts? AWS IAM Identity Center helps you securely create or connect your workforce identities and manage their access centrally across AWS accounts and applications](https://courses.edx.org/courses/course-v1:AWS+AWS-ARCH-1+3T2022/discussion/forum/e06fd9c8431a5908716b783ff9bbc0e4e22b1c07/threads/64a1634c2a472d04ac8021af)

**[1comments](https://courses.edx.org/courses/course-v1:AWS+AWS-ARCH-1+3T2022/discussion/forum/e06fd9c8431a5908716b783ff9bbc0e4e22b1c07/threads/64a1634c2a472d04ac8021af)**

[discussion](https://courses.edx.org/courses/course-v1:AWS+AWS-ARCH-1+3T2022/discussion/forum/e06fd9c8431a5908716b783ff9bbc0e4e22b1c07/threads/649b980e2a472d04a5830da6)

[Answer on AWS Best Practices](https://courses.edx.org/courses/course-v1:AWS+AWS-ARCH-1+3T2022/discussion/forum/e06fd9c8431a5908716b783ff9bbc0e4e22b1c07/threads/649b980e2a472d04a5830da6)

[- What are the benefits of using multiple AWS accounts to run workloads? You can group workloads based on business purposes and ownership, centralize logging, and constrain access to sensitive data. You can also limit the scope of impact from adverse events, manage costs better, and distribute AWS service quotas and API request rate limits. - What are some common patterns for organizing resources in AWS accounts? Sandbox accounts: Typically disconnected from your enterprise services and don’t provide access to your internal data. Development accounts: Typically provide limited access to your enterprise services and development data. - What is the benefit of using AWS Identity and Access Management (IAM) roles for authenticating into AWS accounts? AWS IAM Identity Center helps you securely create or connect your workforce identities and manage their access centrally across AWS accounts and applications.](https://courses.edx.org/courses/course-v1:AWS+AWS-ARCH-1+3T2022/discussion/forum/e06fd9c8431a5908716b783ff9bbc0e4e22b1c07/threads/649b980e2a472d04a5830da6)

**[1comments](https://courses.edx.org/courses/course-v1:AWS+AWS-ARCH-1+3T2022/discussion/forum/e06fd9c8431a5908716b783ff9bbc0e4e22b1c07/threads/649b980e2a472d04a5830da6)**

[discussion](https://courses.edx.org/courses/course-v1:AWS+AWS-ARCH-1+3T2022/discussion/forum/e06fd9c8431a5908716b783ff9bbc0e4e22b1c07/threads/63d955463caa5604c9e9e7be)

[Best Practices for AWS account administration](https://courses.edx.org/courses/course-v1:AWS+AWS-ARCH-1+3T2022/discussion/forum/e06fd9c8431a5908716b783ff9bbc0e4e22b1c07/threads/63d955463caa5604c9e9e7be)

[\*\*\*What are the benefits of using multiple AWS accounts to run workloads?\*\*\* Some benefits of using multiple accounts to run AWS workloads include the following: 1. Multiple accounts approach allows for better administrative oversight of the various AWS workloads and resources. They can be group based on their business purpose and ownership. This in turn facilitates auditing and accounting, and cost management. 2. Grouping workloads under numerous accounts enables the organization to apply security controls by user environments and also protect sensitive data by applying different access controls to different workloads. 3. Account level access enables ease of deployment and application of access control and rules on different resources by using automation tools and service control polices. This way, accounts can be automatically assigned to groups, organisations units that share common business objectives and access requirements. 4. Multi-account strategy protects the workloads from adverse events by limiting the possible scope of impact(blast radius) based on the different access rights and restrictions assigned each account. 5. Multiple accounts approach also allows for the distribution of resources such as API requests to avoid the limitations of accounts exhausting their resource quotas. 6. Finally, a multi-account approach enables innovation, where specific access rights and controls and guardrails are applied to innovation related workloads and accounts in a sandbox environment. \*\*\*What are some common patterns for organizing resources in AWS accounts?\*\*\* Common patterns for organising resources utilise the Organisation Unit (OU) structure described below. At the shared services account level, we have the several OU groups listed with their own structures: \*\*Management\*\* – a flat organisation consisting of root user and other management accounts \*\*Security\*\* – The security organisation unit is responsible for designing and implementing security policies and rules. They OU cost or 2 sub-OU’s namely testing and production, each with its own associated accounts. \*\*Infrastructure\*\* – Infrastructure OU contains shared infrastructure services. It holds accounts containing AWS infrastructure resources and is used solely for managing and administering infrastructure services such as VPC, firewalls, Transit Gateway, Direct Connect etc. The organisation should be made up of 2 sub-OU’s namely Test and Production. \*\*Sandbox\*\* – The Sandbox OU contains accounts in which developers and application builders are generally allowed to be innovative and creative using AWS services. The environment is usually disconnected from other resources such as internal networks and services and isolated from Workloads OU. The Sandbox accounts are also controlled with AWS Quotas. Sandbox OUs are usually flat in nature. \*\*Workload\*\* – The workload OUs host the business specific workloads – mostly commercial applications. They are also mostly shared resources used by other applications. The recommended organisation is made up of sub-OU’s namely Development, Testing and Production. At the next OU level are purpose specific accounts depending on the size and/or volume of workloads deployed. For each workload environment, appropriate guardrails are required to especially for the Development and Testing OUs. In more advanced organisations, Development and Testing OU’s are separate. \*\*Transitional\*\* – This organisational unit is a temporary holding area for workloads or accounts that are migrated or moved to an organisation. They are kept here before they are integrated into appropriate areas in the AWS structure. \*\*Policy Staging\*\* – This OU is created for teams that manage the overall policies for the AWS environment. It is designed to test policy designs and changes before implementing or applying them. Policy OU’s can be sub divided as the 2nd level into different areas of policy such as security, infrastructure, workload, sandbox etc, with each sub OU having a production and a test OU. Other possible OU’s include Individual Business OU, Deployments OU, Exceptions OU and Suspended OU. \*\*What are the Benefits of IAM roles?\*\* IAM roles allow for cross account access to services and resources without the need to create static credentials for every account that wants to access resource. This saves on administrative overheads and simplifies the overall administration of the services. I AM roles can also allow for third part or Independent Identity provider authentication by AWS. This simplified the role of adding users who have already been identified by other services.](https://courses.edx.org/courses/course-v1:AWS+AWS-ARCH-1+3T2022/discussion/forum/e06fd9c8431a5908716b783ff9bbc0e4e22b1c07/threads/63d955463caa5604c9e9e7be)

**[1comments](https://courses.edx.org/courses/course-v1:AWS+AWS-ARCH-1+3T2022/discussion/forum/e06fd9c8431a5908716b783ff9bbc0e4e22b1c07/threads/63d955463caa5604c9e9e7be)**

### Reading 4.4: Logging Services

Multiple AWS services do infrastructure logging, and you should be familiar with them: AWS CloudTrail, AWS Config, VPC Flow Logs, and Amazon GuardDuty. Raf recommended that the customer configure CloudTrail to concentrate data in the AWS Organizations management account, thus following best practices for account governance.

## AWS CloudTrail

CloudTrail monitors and records account activity across your AWS infrastructure, which gives you control over actions for storage, analysis, and remediation. Events include actions that are performed in the AWS Management Console, the AWS Command Line Interface (AWS CLI), and the AWS SDKs and APIs. The actions recorded by CloudTrail don’t include API calls that are made to a backend application that’s hosted on Amazon Elastic Compute Cloud (Amazon EC2). They also don’t include internal API calls that are made within an application, and don’t involve AWS API calls. CloudTrail captures AWS API call events and logs them.  
  
CloudTrail typically delivers logs within an average of about 15 minutes after an API call. (This time is not guaranteed.)  
  
By using CloudTrail, a user in an Organizations management account can create an organization trail that logs all events for all AWS accounts in that organization. Organization trails are automatically applied to all member accounts in the organization. Member accounts can see the organization trail, but they can't modify or delete the organization trail. By default, member accounts don't have access to the log files for the organization trail in the Amazon Simple Storage Service (Amazon S3) bucket. This feature helps you uniformly apply and enforce your event-logging strategy across the accounts in your organization.  
  
For more information about CloudTrail, see [How CloudTrail works](https://docs.aws.amazon.com/awscloudtrail/latest/userguide/how-cloudtrail-works.html).

## AWS Config

It’s important to know the difference between AWS CloudTrail and AWS Config.  
  
When you run your applications on AWS, you usually use AWS resources, which you must create and manage collectively. As the demand for your application grows, our need to keep track of your AWS resources also grows. AWS Config is designed to help you manage your application resources.  
  
AWS Config provides a detailed view of how AWS resources are configured in your AWS account. This configuration includes how resources are related to one another, and how they were configured in the past—which means that you can see how the configurations and relationships change over time. You can use AWS Config to get an inventory of the resources that you have in your AWS account, and then you can apply rules for how those resources are configured.  
  
An AWS *resource* is an entity that you can work with in AWS, such as an EC2 instance, an Amazon Elastic Block Store (EBS) volume, a security group, or a virtual private cloud (VPC).

### Resource administration

To exercise better governance over your resource configurations—and to detect resource misconfigurations—you need fine-grained visibility into what resources exist, and how these resources are configured at any time. You can use AWS Config to notify you when resources are created, modified, or deleted without needing to monitor these changes by polling the calls made to each resource.  
  
You can use AWS Config rules to evaluate the configuration settings of your AWS resources. When AWS Config detects that a resource violates the conditions in one of your rules, AWS Config flags the resource as noncompliant and sends a notification. AWS Config is designed to continuously evaluate your resources as they are created, changed, or deleted.

### Auditing and compliance

You might work with data that requires frequent audits to ensure compliance with internal policies and best practices. To demonstrate compliance, you need access to the historical configurations of your resources. AWS Config can provide this information.

### Managing and troubleshooting configuration changes

When you use multiple AWS resources that depend on one another, a change in the configuration of one resource might have unintended consequences on related resources. With AWS Config, you can view how the resource that you want to modify is related to other resources, and thus assess the potential impact of your change.  
  
You can also use the historical configurations of your resources that are provided by AWS Config to troubleshoot issues and to access the last-known good configuration of a problem resource.

### Security analysis

To analyze potential security weaknesses, you need detailed historical information about your AWS resource configurations. Examples include the AWS Identity and Access Management (IAM) permissions that are granted to your users, or the Amazon EC2 security group rules that control access to your resources.  
  
You can use AWS Config to view the IAM policy that was assigned to an IAM user, group, or role at any time when AWS Config was recording. This information can help you determine the permissions that belonged to a user at a specific time: for example, you can view whether the user *John Doe* had the permissions to modify Amazon Virtual Private Cloud (Amazon VPC) settings on Jan 1, 2015.  
  
You can also use AWS Config to view the configuration of your Amazon security groups, including the port rules that were open at a specific time. This information can help you determine whether a security group blocked incoming TCP traffic to a specific port.  
  
For more information, see [What is AWS Config?](https://docs.aws.amazon.com/config/latest/developerguide/WhatIsConfig.html)

## VPC Flow Logs

VPC Flow Logs is a feature that you can use to capture information about the IP traffic that goes to and from network interfaces in your VPC. Flow log data can be published to Amazon CloudWatch Logs or Amazon S3. After you create a flow log, you can retrieve and view its data in the chosen destination.  
  
Flow logs can help you with various tasks, such as the following:

* Diagnosing overly restrictive security group rules
* Monitoring the traffic that reaches your instance
* Determining the direction of the traffic to and from the network interfaces

Flow log data is collected outside the path of your network traffic. Therefore, it doesn’t affect network throughput or latency. You can create or delete flow logs with a minimal risk of impact to network performance.  
  
For more information, see [Logging IP traffic using VPC Flow Logs](https://docs.aws.amazon.com/vpc/latest/userguide/flow-logs.html).

## Amazon GuardDuty

Amazon GuardDuty is a near-continuous security monitoring service that analyzes and processes data sources, such as CloudTrail data events for Amazon S3 logs, CloudTrail management event logs, DNS logs, Amazon EBS volume data, Amazon Elastic Kubernetes Service (Amazon EKS) audit logs, and Amazon VPC flow logs.  
  
It uses threat intelligence feeds (such as lists of malicious IP addresses and domains) and machine learning to identify unexpected, potentially unauthorized, and malicious activity within your AWS environment. This activity can include issues such as the escalation of privileges, use of exposed credentials, communication with malicious IP addresses or domains, or the presence of malware on your EC2 instances and container workloads.  
  
For example, GuardDuty can detect compromised EC2 instances and container workloads that are serving malware or mining bitcoin. It also monitors AWS account access behavior for signs of compromise, such as unauthorized infrastructure deployments—for example, instances that are deployed in a Region that has never been used, or unusual API calls (such as a password policy change to reduce password strength).  
  
For more information, see [What is Amazon GuardDuty?](https://docs.aws.amazon.com/guardduty/latest/ug/what-is-guardduty.html)

### Automatic Account Provisioning

## Level 2 headings may be created by course providers in the future.

Bookmark this page

This week, Raf suggested that the customer automate as much as possible when they provision new AWS accounts and configure them for use. By using the AWS service stack, they would have a solution that creates new accounts with AWS Organizations, applies security guardrails with service control policies (SCPs), authenticates users into accounts with AWS IAM Identity Center (successor to AWS Single Sign-On), and has centralized logging through AWS CloudTrail. To configure newly created accounts, Raf suggested that the customer use AWS Control Tower, and use AWS Service Catalog to determine which portfolio of solutions is available in each account when they are configured.

## AWS CloudFormation

The services that we covered in this course—such as AWS Control Tower and AWS Service Catalog—use AWS CloudFormation templates. CloudFormation is an infrastructure as code (IaC) service. It helps you model and set up your AWS resources so that you can spend less time managing those resources, and more time focusing on your applications that run in AWS.  
  
With CloudFormation, you create a template that describes all the AWS resources that you want—such as Amazon Elastic Compute Cloud (Amazon EC2) instances or Amazon Relational Database Service (Amazon RDS) DB instances—and CloudFormation provisions and configures those resources for you. You don't need to individually create and configure AWS resources and determine what resource is dependent on what. Instead, CloudFormation handles provisioning and configuration.  
  
When you work with AWS, you should be very familiar with CloudFormation and its features. It’s a best practice to deploy infrastructure in an automated way, instead of doing everything manually in the console.  
  
For more resources about CloudFormation, see the following:

* For more information, see [What is AWS CloudFormation?](https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/Welcome.html)
* For a tutorial about how to use CloudFormation, see [Deploy a Reliable Multi-tier Infrastructure Using CloudFormation](https://www.wellarchitectedlabs.com/reliability/100_labs/100_deploy_cloudformation/).

## AWS Control Tower

AWS Control Tower offers a straightforward way to set up and govern an AWS multi-account environment that follows prescriptive best practices. AWS Control Tower orchestrates the capabilities of several other [AWS services](https://docs.aws.amazon.com/controltower/latest/userguide/integrated-services.html)—including AWS Organizations, AWS Service Catalog, and IAM Identity Center—to build a landing zone in typically less than an hour. Resources are set up and managed on your behalf.  
  
AWS Control Tower orchestration extends the capabilities of Organizations. To help protect your organizations and accounts from being affected by *drift*(or divergence from best practices), AWS Control Tower applies preventive and detective controls (or guardrails). For example, you can use guardrails to help ensure that security logs and necessary cross-account access permissions are created, but not altered.  
  
If you host more than a handful of accounts, it’s beneficial to have an orchestration layer that facilitates account deployment and account governance. You can adopt AWS Control Tower as your primary way to provision accounts and infrastructure. With AWS Control Tower, you can more easily adhere to corporate standards, meet regulatory requirements, and follow best practices.  
  
AWS Control Tower uses CloudFormation StackSets to set up resources in your accounts. Each stack set has stack instances that correspond to accounts, and to AWS Regions per account. AWS Control Tower deploys one stack set instance per account and Region.  
  
For more information, see [What is AWS Control Tower?](https://docs.aws.amazon.com/controltower/latest/userguide/what-is-control-tower.html)

## AWS Service Catalog

By using AWS Service Catalog, organizations can create and manage catalogs of IT services that are approved for AWS. These IT services can include virtual machine images, servers, software, databases, and more—they can even include complete, multi-tier application architectures.  
  
Organizations can use AWS Service Catalog to centrally manage commonly deployed IT services. AWS Service Catalog is designed to help organizations achieve consistent governance and meet compliance requirements. End users can quickly deploy only the approved IT services that they need, and these deployments will follow the constraints that your organization sets.  
  
AWS Service Catalog provides the following benefits:

* Standardization: Administer and manage approved assets by restricting where the product can be launched, the type of instance that can be used, and many other configuration options. The result is a standardized landscape for product provisioning for your entire organization.
* Self-service discovery and launch: Users browse listings of products (that is, services or applications) that they can access, locate the product that they want to use, and launch it on their own as a provisioned product.
* Fine-grained access control: Administrators assemble portfolios of products from their catalog, and add constraints and resource tags that will be used when the products are provisioned. Administrators then grant access to the portfolio through AWS Identity and Access Management (IAM) users and groups.
* Extensibility and version control: Administrators can add a product to different portfolios and restrict it without creating another copy. When the product is updated to a new version, the update is propagated to the product in every portfolio that references it.

For more i You are a consultant who is working with a customer that wants to use AWS to meet their business needs. This customer has multiple AWS accounts, but they want to better control the permissions scope of their accounts. You decide to recommend AWS Organizations for that task, with the help of service control policies (SCPs).

In a few sentences, explain how SCPs are used to define permissions and how this can help an AWS customer follow best practices.

### Discussion - Consider this Scenario

**Topic:** Week 4 / Consider this Scenario-Week4

Hide Discussion

Add a Post

Filter:                                                           Sort:                                                           

[discussion](https://courses.edx.org/courses/course-v1:AWS+AWS-ARCH-1+3T2022/discussion/forum/14945c424e8a44d970608dcd6a79f35211f068db/threads/6629e1fdc4684b04cfce6010)

[aws](https://courses.edx.org/courses/course-v1:AWS+AWS-ARCH-1+3T2022/discussion/forum/14945c424e8a44d970608dcd6a79f35211f068db/threads/6629e1fdc4684b04cfce6010)

[architecture](https://courses.edx.org/courses/course-v1:AWS+AWS-ARCH-1+3T2022/discussion/forum/14945c424e8a44d970608dcd6a79f35211f068db/threads/6629e1fdc4684b04cfce6010)

**[1comments](https://courses.edx.org/courses/course-v1:AWS+AWS-ARCH-1+3T2022/discussion/forum/14945c424e8a44d970608dcd6a79f35211f068db/threads/6629e1fdc4684b04cfce6010)**

[discussion](https://courses.edx.org/courses/course-v1:AWS+AWS-ARCH-1+3T2022/discussion/forum/14945c424e8a44d970608dcd6a79f35211f068db/threads/653b9fc6a7f342048db3df46)

[SCP Best practices](https://courses.edx.org/courses/course-v1:AWS+AWS-ARCH-1+3T2022/discussion/forum/14945c424e8a44d970608dcd6a79f35211f068db/threads/653b9fc6a7f342048db3df46)

[1. certain services can be denied using SCP polices at Ou level 2. privileges' can be provided using IAM at user account level. But SCP help provide guardrail to what a user can access 3. useful when user have access to multiple accounts with different roles/permissions](https://courses.edx.org/courses/course-v1:AWS+AWS-ARCH-1+3T2022/discussion/forum/14945c424e8a44d970608dcd6a79f35211f068db/threads/653b9fc6a7f342048db3df46)

**[5comments (5 unread comments)](https://courses.edx.org/courses/course-v1:AWS+AWS-ARCH-1+3T2022/discussion/forum/14945c424e8a44d970608dcd6a79f35211f068db/threads/653b9fc6a7f342048db3df46)**

[discussion](https://courses.edx.org/courses/course-v1:AWS+AWS-ARCH-1+3T2022/discussion/forum/14945c424e8a44d970608dcd6a79f35211f068db/threads/63dce7d205bf12050b507264)

[SCPs](https://courses.edx.org/courses/course-v1:AWS+AWS-ARCH-1+3T2022/discussion/forum/14945c424e8a44d970608dcd6a79f35211f068db/threads/63dce7d205bf12050b507264)

[Service Control Policies [SCPs] are used to manage permission limits within an organisation unit 9OU) in a multi account AWS organisation. The offer a central control over maximum allowable permissions. The define guardrails on the actions that users can take in an account. SCPs in themselves do not grant permissions to user accounts. The admin must still attach identity based or resource-based policies to IAM users. SCPs do not affect resource-based policies, only users and roles. They also do not affect users and roles from outs an organisation. In the scenario, SCPs will be firstly defined and created at the management account level. and then attached to organisation units within the customer account. It will not be applied the root level. Each OU can share similar SCPs or different SCPs can be defined with their own explicit statements and corollaries.](https://courses.edx.org/courses/course-v1:AWS+AWS-ARCH-1+3T2022/discussion/forum/14945c424e8a44d970608dcd6a79f35211f068db/threads/63dce7d205bf12050b507264)

**[1comments](https://courses.edx.org/courses/course-v1:AWS+AWS-ARCH-1+3T2022/discussion/forum/14945c424e8a44d970608dcd6a79f35211f068db/threads/63dce7d205bf12050b507264)**

nformation, see [What is AWS Service Catalog?](https://docs.aws.amazon.com/servicecatalog/latest/adminguide/introduction.html)

### Reading 4.6: Multi-account Best Practices

Implementing multi-account best practices is an ongoing effort. The following links and references provide resources so that you can learn more about the subject. Remember that the goal of this course is not to dive deep into each approach. Instead, the goal of this course is to maintain knowledge at a Solutions Architect - Associate level by highlighting the services that are used for the most common approaches.

## Multi-account environments

The *AWS for Industries Blog* includes a blog post called [Best Practices for AWS Organizations Service Control Policies in a Multi-Account Environment](https://aws.amazon.com/blogs/industries/best-practices-for-aws-organizations-service-control-policies-in-a-multi-account-environment/). This blog post talks about core concepts of AWS Organizations and service control policies (SCPs). It also provides suggestions about using different organizational unit (OU) structures for different use cases (such as a corporate OU, a production public-facing OU, a production internal-facing OU, and a security OU). At the end, you might have a complex structure that looks like the following diagram:  
  
[Organizing Your AWS Environments Using Multiple Accounts](https://docs.aws.amazon.com/whitepapers/latest/organizing-your-aws-environment/organizing-your-aws-environment.html) is an AWS Whitepaper that was published on July 26, 2022. It talks about the AWS Well-Architected Framework and much of the content that you learned about in this course. It’s a good read!  
  
The *AWS Organizations* website also has another reference for further reading, called [Establishing your best practice AWS environment](https://aws.amazon.com/organizations/getting-started/best-practices/).

## Tag policies and SCPs

With Organizations, you can set tag policies, in addition to defining SCPs. You can use tag policies to maintain standardized tags for AWS resources that are used with Organizations accounts. For example, the following example tag policy defines a tag with a key of *Environment*and a value of *Production*, and this tag is enforced for Amazon Elastic Compute Cloud (Amazon EC2) instances.

{  
 "tags": {  
 "Environment": {  
 "tag\_key": {  
 "@@assign": "Environment"  
 },  
 "tag\_value": {  
 "@@assign": [  
 "Production"  
 ]  
 },  
 "enforced\_for": {  
 "@@assign": [  
 "ec2:instance"  
 ]  
 }  
 }  
 }  
}

This tag policy prevents users from changing this tag on *existing*Amazon EC2 instances. *However, it doesn't prevent a user from launching new instances with non-compliant tags, or no tags.* This tag policy might be adequate if you use infrastructure as code (IaC) to provision environments. With IaC, environments will be created from a CloudFormation template, and you can thus embed the tags in the template.  
  
*However, if you want to prevent the creation of new AWS resources that aren't tagged, you need to use SCPs.* You could use the following example SCP to make sure that the AWS resources are created only if a certain tag is present. This SCP requires specific tags on specified created resources, and it uses explicit deny statements.

{  
 "Version": "2012-10-17",  
 "Statement": [  
 {  
 "Sid": "DenyCreateSecretWithNoProjectTag",  
 "Effect": "Deny",  
 "Action": "secretsmanager:CreateSecret",  
 "Resource": "\*",  
 "Condition": {  
 "Null": {  
 "aws:RequestTag/Project": "true"  
 }  
 }  
 },  
 {  
 "Sid": "DenyRunInstanceWithNoProjectTag",  
 "Effect": "Deny",  
 "Action": "ec2:RunInstances",  
 "Resource": [  
 "arn:aws:ec2:\*:\*:instance/\*",  
 "arn:aws:ec2:\*:\*:volume/\*"  
 ],  
 "Condition": {  
 "Null": {  
 "aws:RequestTag/Project": "true"  
 }  
 }  
 },  
 {  
 "Sid": "DenyCreateSecretWithNoCostCenterTag",  
 "Effect": "Deny",  
 "Action": "secretsmanager:CreateSecret",  
 "Resource": "\*",  
 "Condition": {  
 "Null": {  
 "aws:RequestTag/CostCenter": "true"  
 }  
 }  
 },  
 {  
 "Sid": "DenyRunInstanceWithNoCostCenterTag",  
 "Effect": "Deny",  
 "Action": "ec2:RunInstances",  
 "Resource": [  
 "arn:aws:ec2:\*:\*:instance/\*",  
 "arn:aws:ec2:\*:\*:volume/\*"  
 ],  
 "Condition": {  
 "Null": {  
 "aws:RequestTag/CostCenter": "true"  
 }  
 }  
 }  
 ]  
}

Do you see how SCPs and tag policies can—and should—be used together? For more information, see [Require a tag on specified created resources](https://docs.aws.amazon.com/organizations/latest/userguide/orgs_manage_policies_scps_examples_tagging.html#example-require-tag-on-create) in the *AWS Organizations User Guide*.

### Capstone Project

It’s time to put together everything you learned!

As part of this project, you will create a high-level architecture diagram that uses[AWS service icons and arrows](https://aws.amazon.com/architecture/icons/) to depict an AWS solution for the given scenario. Create your diagram by using a tool like[diagrams.net](https://app.diagrams.net/?splash=0&libs=aws4), or you can select a different tool by from the[AWS Architecture Icons](https://aws.amazon.com/architecture/icons/) page by scrolling to the **Drawing and diagramming tools section**.

**Scenario:**You are working for a customer that runs their workloads on premises. Your customer has two workloads:

* A three-tier architecture composed of a frontend (HTML, CSS, JavaScript), backend (Apache Web Server and a Java application), and database (MySQL). The three-tier application hosts a dynamic website that accepts user traffic from the internet.
* A data analytics workload that runs Apache Hadoop. The analytics workload analyzes a massive amount of data that stored on premises and it also uses visualization tools to derive insights.

These components are currently running in the data center on physical servers. Currently, if a power outage occurred in the data center, all systems would be brought offline. Because of this issue (in addition to other benefits of the cloud), your customer wants to migrate all components to the cloud and, when possible, use AWS services to replace on-premises components.

**Instructions:**You have been tasked with designing a solution that uses AWS services to decouple the application layers (frontend, backend, and database), and that hosts both the application and the data analytics workload in the cloud. You can use managed services and advocate for refactoring the code to take advantage of cloud-native technologies, or you can do a lift and shift and advocate for minimal refactoring. Also, the data analytics solution currently runs on Hadoop and you have a requirement to spin up an Amazon EMR cluster for it. However, it’s up to you to choose which AWS services you want to use for the ingestion, storage, and visualization of data.

Whichever architecture you choose to create, write an explanation that details how the solution works and why you chose to use the services that you selected. Also, create an architecture diagram that depicts how both solutions will be hosted on AWS.

**Explain each activity and** each service **in this conversation with steps and examples with aws mangement console for**

Explain each service with steps and examples with aws mangement console for

**https://pages.awscloud.com/ec2-virtual-workstations-contact-us.html**