AWS Architecture Monthly

August 2019

Serverless

Notices

Editor's Note

This month’s magazine is all about serverless—the native architecture of the cloud that enables you to shift more of your operational responsibilities to AWS and allows you to build and run applications and services without thinking about servers.

Think about it: you can eliminate management tasks like patching or server and cluster provisioning and spend time on other things to help grow your business. In fact, serverless applications such as AWS Lambda, Amazon S3, Amazon DynamoDB, and API proxy don’t require provisioning, maintaining, and administering servers for backend components such as compute, databases, storage, stream processing, and message queueing.

Whatever will you do with all that saved time and money? Agility and innovation, here you come.

We hope you’ll find this edition of Architecture Monthly useful, and we’d like your feedback. Please give us a star rating and your comments on Amazon. You can also reach out to aws-architecture-monthly@amazon.com anytime.

For August’s magazine, we’ve assembled architectural best practices about serverless architectures learning from all over AWS, and we’ve made sure that a broad audience can appreciate it.

- **Interview**: Chris Munns, Principal Developer Advocate
- **Training**: Building A Serverless Data Lake on AWS
- **Blog post**: How to Design Your Serverless Apps for Massive Scale
- **Solution**: Serverless Image Handler
- **Podcast**: Understanding the AWS Serverless Application Model (SAM)
- **Whitepaper**: Implementing Microservices on AWS

Annik Stahl, Editor

Interview:

Chris Munns, Principal Developer Advocate

I recently met with Chris Munns, AWS Principal Developer Advocate and a serverless guru.

What are some of the architectural patterns that we see customers use in the field?

When it come to the serverless space, we typically think about the kind of origin use cases that we see for serverless applications. You bucket those primarily into five types of use cases.

The first is web applications, when you need to be able to power, whether it be a single page app for some sort of more modern, reactive Web interface, or maybe even something where it's a little bit more legacy but you still have a separation between the front end in the back end. Then serverless becomes a really easy way to do that.

We also talk to people building true internal back ends for services, whether you want to call it an internal microservice or a back end for a mobile app or IoT device—that also becomes a common use case. And typically these were both more traditional API Gateway-type driven workloads.

We see people doing data processing where they're streaming or batch processing, and whether they're plugging into something like Kinesis or S3, there's a lot of data processing that goes on a really large scale in AWS Lambda, and it's one of the largest drivers of Lambda today.

And then we see use cases like Chatbots and Alexa, which are increasingly popular ways that companies are interfacing both internally and externally. You know that we obviously use Alexa pretty heavily internally.

And the last thing would be the glue use cases of IT governance, security, developer tools, management tools, kind of all of those things that you could just kind of plot a little Lambda function into and have it do some sort of business logic for you. That is also kind of a way that a lot of people typically dip their toes into the space of using things like Lambda. It's one of the most common is that we see out there.

When is it a good time for customers to think about planning for serverless?

I don't know if I would say that there's any specific time. I know that we say that if you're thinking about a greenfield application, we have a lot of customers that are thinking about serverless first. They say, “Can we do this in a serverless way? Can we use Lambda? Can we use API Gateway, step functions, or something else to glue this together, like Kinesis or S3?”

A lot of companies consider if serverless will work for the greenfield projects. We also see some companies that are looking to essentially re-platform or rebuild something, so they're also looking at serverless. They’re taking maybe a traditional enterprise application and they're reimagining that with something like API Gateway and Lambda and this sort of static hosting or maybe something like Amplify Console now to house the front end.

Where do you see the highest level of adoption of serverless architecture in the industry?

I don’t think that we see any one category of customer using it more heavily than others.

We have customers, such as large financial institutions, which have used Lambda both for powering Web applications and internal governance tools. And FINRA has been using Lambda for data processing—in near real-time—hundreds of thousands of actions.

We also see it with small startups. Bustle, a media organization that focuses on women-directed media companies, has acquired various media properties—websites that they basically reinvent as a serverless product, while lowering their costs to run them, and they're doing that with serverless technologies behind the scene.

So, you can see the big gap between a bank and financial regulations agency down to a media and entertainment startup, and there's lots of stuff in between there as well.

Why would customers not switch everything to serverless from an application architecture perspective?

Well, it's not an “everything type” of conversation, and this would be one of the stranger misconceptions that are out there. At the end of the day, when we talk about products like AWS Lambda and Amazon API Gateway and the rest of the portfolio of products, they're not meant to be the only way. There are instances when people will use containers or regular EC2 and maybe some sort of other high-level managed service, and one of the things that we look for and try to encourage is figuring out where people are spending their time and how can they reduce the effort, reduced the operations, and find things that are a good fit.

Generally speaking, I don't think we'll ever reach a point where there's just one ring to rule them all, as it were, when it comes to a few technologies. But we do see that the serverless products, again, offer the benefits of the lowest kind of operability overhead and they’ve the best cost alignment with use cases. So we do see many organizations that say that they are serverless first but again, serverless first doesn't mean serverless only.

How do you see the field of serverless evolving?

We keep seeing companies having that “Aha!” moment, when they start thinking about all the things they’ve been doing in the past to manage all these resources or run all these servers. And now they *don’t* have to do all that stuff anymore so they start to kind of ask themselves, “Well, what do we do now with our time?” And it becomes a great enabler of increased agility and faster time-to-market. And they start to be able to then spend time on some of things that they couldn’t before like testing, operations, operability, and raising the bar internally for themselves.

At the last re:Invent I had the opportunity to talk with both a large bank and a company called Centrica, which is a UK utilities-based organization. Both these organizations talked about just this: how they took a three- or four-tier application and reimagined it as something serverless and just almost immediately saw benefits such as lower costs, lower overhead, better reliability, better scale. And they looked at it and thought, well this is one app that we have, but we have dozens or hundreds or apps like this and so the potential for those benefits then be magnified is an incredible impact on their business.

How can companies get started with serverless computing?

We have a whole lot of getting started material out there today. If folks go to https://aws.amazon.com.serverless, they’ll find links to all sorts of resources and info about our products, our partners, getting started, customer case studies. It's a great place make your first step into the space.

Available online at: https://amzn.to/AWS-SL-Datalake-training

Description

In this one-day advanced course, you will learn to design, build, and operate a serverless data lake solution with AWS services. This course will include topics such as ingesting data from any data source at large scale, storing the data securely and durably, enabling the capability to use the right tool to process large volumes of data, and understanding the options available for analyzing the data in near-real time.

Intended Audience

This course is intended for:

- Solutions architects
- Big Data developers
- Data architects and analysts
- Data analysis practitioners

Course Objectives

In this course, you will learn to:

- Collect large amounts of data using services such as Amazon Kinesis Data Streams and Amazon Kinesis Data Firehose and store the data durably and securely in Amazon Simple Storage Service (Amazon S3)
- Create a metadata index of your data lake
- Choose the best tools for ingesting, storing, processing, and analyzing your data in the lake
- Apply the knowledge to hands-on labs that provide practical experience with building an end-to-end solution
- Configure Amazon Simple Notification Service (Amazon SNS) to audit, monitor, and receive event notifications about activities in the data warehouse
- Prepare for operational tasks, such as resizing Amazon Redshift clusters and using snapshots to back up and restore clusters

- Use a business intelligence (BI) application to perform data analysis and visualization tasks against your data

Find a class near you:
https://www.aws.training/training/schedule?courseId=13623&src=detail

Real-World Example

Viber: Massive Data Lakes on AWS

https://amzn.to/AWS-SL-Viber

See how Viber built a data storage and processing solution that handles 10-15 billion events per day, peaking at 300k events per second, for its over 1B users worldwide. You'll see how they use a combination of AWS services, open source tools, and AWS partner solutions to build a flexible, end-to-end solution.

Blog:

How to Design Your Serverless Apps for Massive Scale

By George Mao

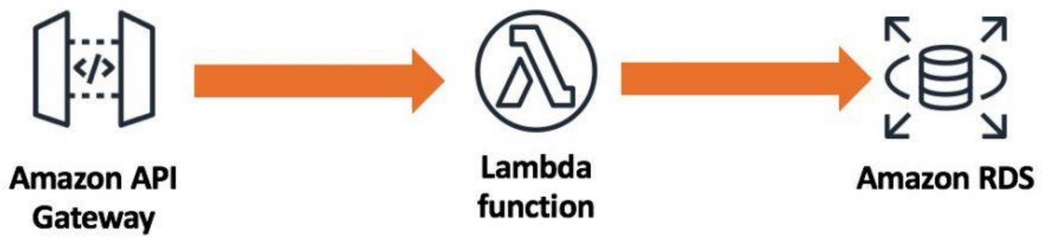
Available online at: https://amzn.to/AWS-SL-scaling

Serverless is one of the hottest design patterns in the cloud today, allowing you to focus on building and innovating, rather than worrying about the heavy lifting of server and OS operations. In this series of posts, we'll discuss topics that you should consider when designing your serverless architectures. First, we'll look at architectural patterns designed to achieve massive scale with serverless.

Scaling Considerations

In general, developers in a "serverful" world need to be worried about how many total requests can be served throughout the day, week, or month, and how quickly their system can scale. As you move into the serverless world, the most important question you should understand becomes: "What is the concurrency that your system is designed to handle?"

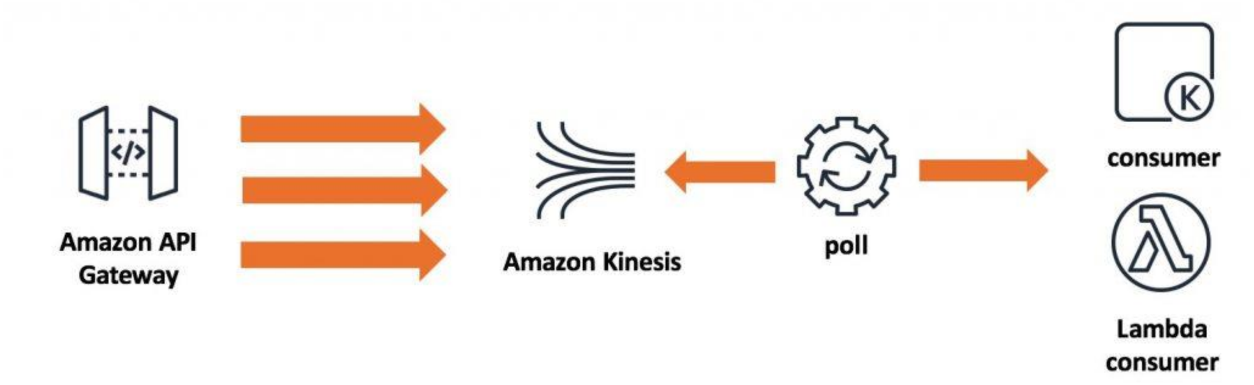
The AWS Serverless platform allows you to scale very quickly in response to demand. Below is an example of a serverless design that is fully synchronous throughout the application. During periods of extremely high demand, Amazon API Gateway and AWS Lambda will scale in response to your incoming load. This design places extremely high load on your backend relational database because Lambda can easily scale from thousands to tens of thousands of concurrent requests. In most cases, your relational databases are not designed to accept the same number of concurrent connections.



This design risks bottlenecks at your relational database and may cause service outages. This design also risks data loss due to throttling or database connection exhaustion.

Cloud Native Design

Instead, you should consider decoupling your architecture and moving to an asynchronous model. In this architecture, you use an intermediary service to buffer incoming requests, such as Amazon Kinesis or Amazon Simple Queue Service (SQS). You can configure Kinesis or SQS as out of the box event sources for Lambda. In design below, AWS will automatically poll your Kinesis stream or SQS resource for new records and deliver them to your Lambda functions. You can control the batch size per delivery and further place throttles on a per Lambda function basis.



This design allows you to accept extremely high volume of requests, store the requests in a durable datastore, and process them at the speed which your system can handle.

Conclusion

Serverless computing allows you to scale much quicker than with server-based applications, but that means application architects should always consider the effects of scaling to your downstream services. Always keep in mind cost, speed, and reliability when you're building your serverless applications.

Our next post in this series will discuss the different ways to invoke your Lambda functions and how to design your applications appropriately.

Real-World Example

HSBC: Using serverless at scale in banking

https://amzn.to/AWS-SL-HSBC

Building a scalable, elastic platform while maintaining bank-grade security.

Available online at: https://amzn.to/AWS-SL-Solution-imagehandler

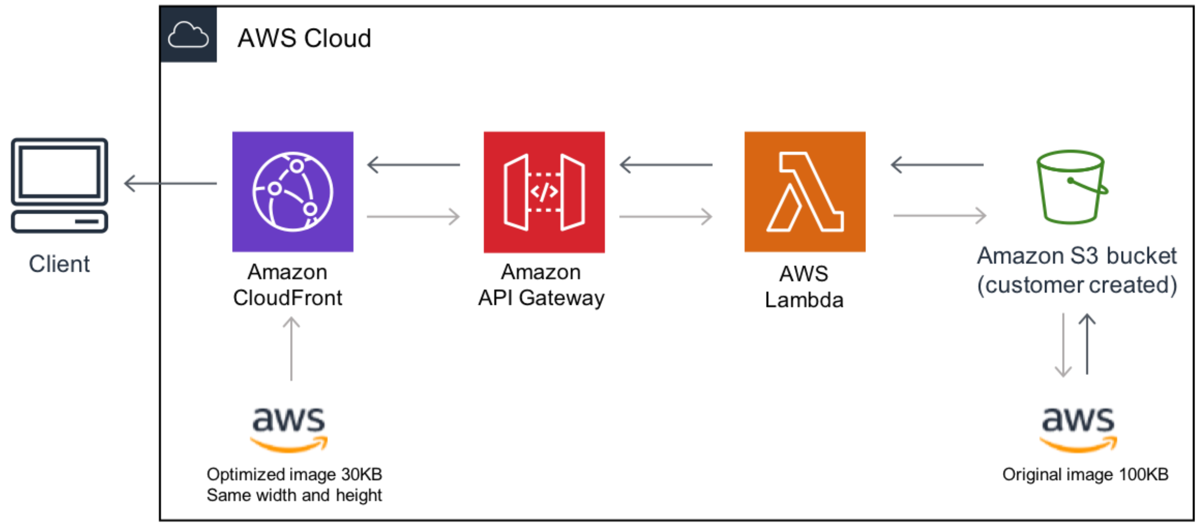
What does this AWS Solution do?

Many Amazon Web Services (AWS) customers use images on their websites and mobile applications to drive user engagement. Websites with large image files can experience high load times, so in order to ensure a great user experience across different devices, developers often provide multiple versions of each image to accommodate different bandwidth and layout constraints. This process can be difficult to manage and cause time delays, as it often requires version control, increased storage and compute costs for file reprocessing, and coordination with application teams and web developers to update image files.

To help customers provide a low-latency website response, and decrease the cost of image optimization, manipulation, and processing, AWS offers the Serverless Image Handler, a solution that combines highly available, trusted AWS services and the open source image processing suite Sharp to enable fast and cost-effective image manipulation on the AWS Cloud. This reference implementation automatically deploys and configures a serverless architecture that is optimized for dynamic image manipulation, and uses Amazon CloudFront for global content delivery and Amazon Simple Storage Service (Amazon S3) for reliable and durable cloud storage at a low cost.

AWS Solution Overview

AWS offers a simple solution that automatically deploys and configures a serverless architecture that is optimized for dynamic image manipulation. The diagram below presents the Serverless Image Handler architecture you can deploy in minutes using the solution's implementation guide and accompanying AWS CloudFormation template.



Serverless Image Handler architecture

AWS Lambda retrieves images from your Amazon Simple Storage Service (Amazon S3) bucket and uses Sharp to return a modified version of the image to the Amazon API Gateway. The solution generates a Amazon CloudFront domain name that provides cached access to the image handler API.

Additionally, the solution deploys an optional demo user interface where you can interact directly with your image handler API endpoint using image files that already exist in your account. The demo UI is deployed in an Amazon S3 bucket to allow customers to immediately start manipulating images with a simple web interface. CloudFront is used to restrict access to the solution's website bucket contents.

View the deployment guide: https://amzn.to/AWS-SL-Image-deployment.

Real-World Example

Amazon Fresh: Serverless Product Selection at Amazon Fresh

http://bit.ly/AWS-SL-TMA-Fresh

See how Amazon Fresh team built a product selection workflow solution on AWS. This event-driven, serverless architecture leverages Lambda, S3, CloudFront, API Gateway, WAF, Cognito, RDS Aurora, CloudWatch, and SQS. You'll learn how adopting AWS allowed them to iterate more quickly, reduce operational overhead, and ultimately improve product selection for Amazon Fresh customers.

Do you want to deploy Serverless applications faster, easier, and more reliably? The AWS Serverless Application Model (SAM) might just fit the bill. AWS podcaster Simon Elisha speaks with Gerardo Estaba (Senior Partner Solutions Architect, AWS) about how to get started and how to get the best from it.

Listen to the podcast: https://amzn.to/AWS-SL-podcast321

Real-World Example

Macquarie Bank: Securing Payment Websites at the Edge Serverlessly

https://bit.ly/AWS-SL-Macquarie

Learn how Macquarie Bank has transformed its DEFT digital platform. DEFT is a payment and account receivable platform that processes millions of transactions & billions of dollars per year. You'll learn how this system evolved from an architecture based on Amazon ELB and Amazon EC2 to a serverless architecture largely at the edge, leveraging Amazon S3, AWS Lambda@Edge, Amazon Cloudfront, AWS WAF and AWS Shield.

Available online at: https://amzn.to/AWS-SL-WP-Microservices

Abstract

Microservices are an architectural and organizational approach to software development to speed up deployment cycles, foster innovation and ownership, improve maintainability and scalability of software applications, and scale organizations delivering software and services by using an agile approach that helps teams to work independently from each other. Using a microservices approach, software is composed of small services that communicate over well-defined APIs that can be deployed independently. These services are owned by small autonomous teams. This agile approach is key to successfully scale your organization.

There are three common patterns that we observe when our customers build microservices: API driven, event driven, and data streaming. In this whitepaper, we introduce all three approaches and summarize the common characteristics of microservices, discuss the main challenges of building microservices, and describe how product teams can leverage Amazon Web Services (AWS) to overcome these challenges.

Introduction

Microservices architectures are not a completely new approach to software engineering, but rather a combination of various successful and proven concepts such as:

- Agile software development
- Service-oriented architectures
- API-first design
- Continuous Integration/Continuous Delivery (CI/CD)

In many cases, design patterns of the Twelve-Factor App are leveraged for microservices.

We first describe different aspects of a highly scalable, fault-tolerant microservices architecture (user interface, microservices implementation, and data store) and how to build it on AWS leveraging container technologies. We then recommend the AWS services for implementing a typical serverless microservices architecture in order to reduce operational complexity.

Serverless is defined as an operational model by the following tenets:

- No infrastructure to provision or manage
- Automatically scaling by unit of consumption
- “Pay for value” billing model
- Built-in availability and fault tolerance

Finally, we look at the overall system and discuss the cross-service aspects of a microservices architecture, such as distributed monitoring and auditing, data consistency, and asynchronous communication.

Read the full whitepaper here: https://amzn.to/AWS-SL-WP-Microservices

Real-World Example

IBS: Mission Critical Microservices on AWS

https://amzn.to/AWS-SL-IBS-video

IBS is transforming its mission critical applications from monolith to micro services. Anil explains the deployment architecture, choice of databases, service discovery mechanism and log aggregation services built on top of the AWS platform.