

Serverless Architectures Review, Future Trend and the Solutions to Open Problems

Manoj Kumar*

Solutions Architect, Master of Science in Software Engineering, Specialization in Cloud and Mobile Computing, Computer Engineering, San Jose State University, CA, USA

*Corresponding author: manojsjsu@gmail.com

Received December 11, 2018; Revised January 15, 2019; Accepted January 25, 2019

Abstract Lately, the popularity and adoption of serverless computing or Function-as-a-Service have been grown substantially, and it emerges as a better way to manage cost, reliability, availability, and scalability. This paper presents a detail of serverless offerings from leading cloud providers such as AWS, Azure, Google Cloud Platform and some open source. It compares them side by side in the relevant category like compute, storage, database, messaging, API management, and tooling. Also provides comparative analysis on available serverless architectures for the most common use cases within cloud provider's environment. It will also emphasize on benefits, open problems, possible solutions, and the future of the technology.

Keywords: *serverless architectures, serverless; serverless computing, Function-as-a-Service, FaaS, Backend-as-a-Service, BaaS, AWS lambda, azure functions, google cloud functions, open whisk, stateful serverless, real-time serverless architecture, compare serverless offerings*

Cite This Article: Manoj Kumar, "Serverless Architectures Review, Future Trend and the Solutions to Open Problems." *American Journal of Software Engineering*, vol. 6, no. 1 (2019): 1-10. doi: 10.12691/ajse-6-1-1.

1. Introduction

Serverless computing is an evolutionary technology that enables the developer to build and run code without worrying about servers. In these type of implementation, no need to own any infrastructure. In case of three-tier architecture, presentation tier turns in thin client-side code placed in object storage delivered as Storage as a service (SaaS), domain logic can run as Function as a

service (FaaS), and data storage tier is replaced with Backend as a Service or (BaaS). In the serverless architectures, major portion of the application run in ephemeral stateless containers provided by cloud providers. These containers trigger on events and terminate after execution. So, this type of cloud computing is otherwise called Function as a Service (FaaS). Over 10 years prior journey began with virtualization then Platform-as-a-Service (PaaS), and now Function-as-a-Service. This evolution is well drafted in this timeline [Figure 1](#).

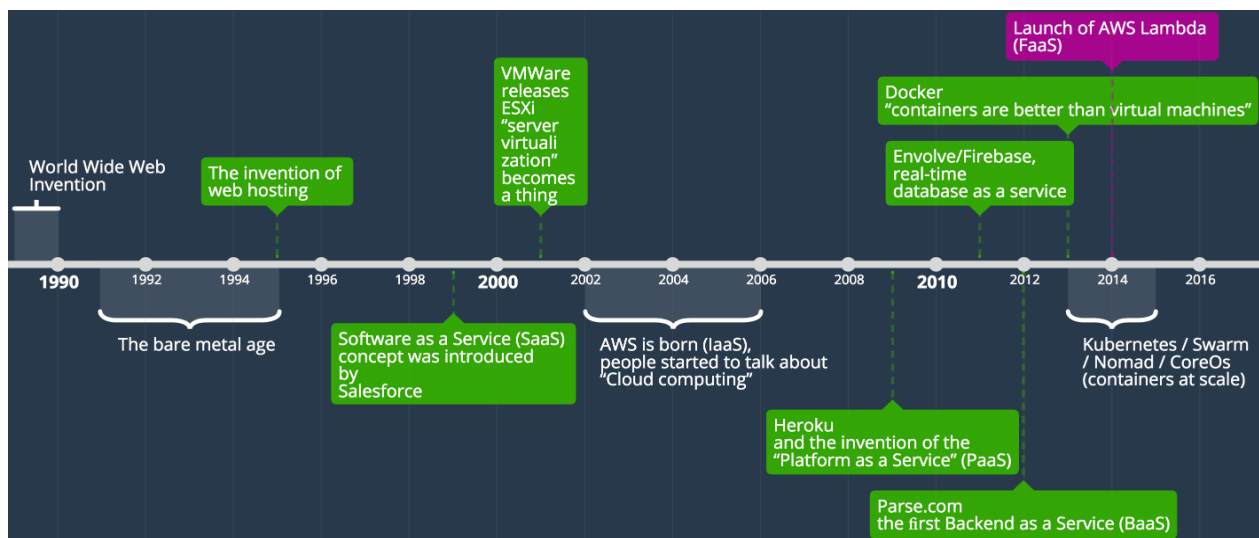


Figure 1. Evolution from Bare metal to Serverless (adapted from [1])

FaaS enables code to run without provisioning or managing servers. Some characteristics are as follows.

1. Any type of backend code in any popular programming language
2. Just worry about uploading code, nothing else. No administration required.
3. Auto Scale
4. High availability
5. Code can be triggered by most of the cloud services as well as APIs

In serverless applications, all the components which form the architecture don't require provisioning, maintenance, and administration of the servers. Depending on the use case these components may include compute, storage, databases, machine learning, stream processing, message queuing, streaming, etc.

2. Serverless Solutions

Serverless is a combination of 'Function as a Service' and 'Backend as a Service' [2]. At high level 'Platform as a Service' looks similar to serverless approach, however, it is not. As per Adrian Cockcroft, "If your PaaS can efficiently start instances in 20ms that run for half a second, then call it serverless.". PaaS platform is not scalable as FaaS does. [2]. I would say, Serverless is like a public transport. Use it and pay just for the use. It can scale as well.

2.1. Serverless Solutions from Leading Providers

Serverless technology provides an abstraction to servers, infrastructures and operating systems. In recent years

almost, all the major cloud provides extended their cloud serverless offerings. Following Table 1 is listing down all the serverless offering from top providers.

2.1.1. AWS

Amazon Web Services (AWS) [3] is the market leader in cloud space they have the most advanced set of serverless products. These fully managed services enable developers to build and run serverless applications. Recently AWS launched Serverless Application Repository which provides starting point for serverless projects. This repository has several publicly available serverless applications which can be searched, deployed and published in few clicks as shown in Figure 2. Developer can search multiple serverless applications deploy and modify them as per their requirement and even integrate them.

2.1.1.1. Serverless Application Use cases

Web applications and backends: For example, we can consider serverless News application. In this case, we can host website code in S3, Lambda can be used to perform data processing unit. It will get News data from DynamoDB and expose it through API gateway. Figure 3 is demonstrating this architecture.

Following Figure 4 is demonstrating serverless backend for mobile apps.

2.1.1.2. Data processing

The real-time data processing system can be build using AWS Lambda, Amazon Kinesis, S3, and DynamoDB. Figure 5 is suggesting sample serverless architecture for photo thumbnail creation.

Following Figure 6 demonstrates real-time data processing.

Table 1. Comparing Serverless Offerings Side by Side. AWS [3], Azure [4], Google Cloud [5]

Type of Service	Leading Cloud Providers		
	AWS aws.amazon.com	Azure azure.microsoft.com	Google cloud.google.com
Compute	AWS Lambda enables the developer to run functions without provisioning any infrastructure or server. It out of the box provides scalability and availability. Lambda@Edge allows developers to run code at edge locations on the CloudFront events.	Azure Functions: event-driven FaaS solution is similar to AWS Lambda. Functions on Azure IoT Edge: Runs code at the edge IoT device even in intermittent connectivity conditions.	Cloud Functions: Event-driven serverless compute platform. Still in Beta.
Storage	Amazon Simple Storage Service (Amazon S3) is object storage to store and retrieve data at any scale from anywhere.	Azure Storage: massively scalable cloud object storage, highly available, and durable.	Cloud Storage: object storage
Database	DynamoDB: NoSQL database service supports both document and key-value store models. AWS AppSync: real-time data update to clients	Azure Cosmos DB: schema agnostic multimodal globally distributed NoSQL database.	Cloud Datastore: auto-scaling NoSQL Database as a Service (DBaaS) on the Google Cloud Platform. Cloud Bigtable: Massively Scalable NoSQL Big Data database service. Firebase Real-time Database: Real-time db for web and mobile apps.
Security and access control	Amazon Cognito enables user authentication and access, IAM	Azure Active Directory: cloud-based identity and access management.	Firebase Authentication: provides backend services, SDKs, and UI libraries to authenticate users to the mobile app.
Cloud messaging	Amazon SNS: pub/sub messaging service. Amazon SQS: message queuing service.	Event Grid: managed event routing service that eliminates the need for polling. Service Bus: messaging infrastructure that enables connection between private and public cloud environments for distributed and cloud solutions.	Cloud Pub/Sub: Ingest event streams for real-time stream analytics. Firebase Cloud Messaging: Cloud messaging platform

Type of Service	Leading Cloud Providers		
	AWS aws.amazon.com	Azure azure.microsoft.com	Google cloud.google.com
Workflow orchestration	AWS Step Functions provides a visual workflow to coordinate between components. In serverless world it enables to coordinate between lambdas. As per my experience, this is in the very early stage.	Logic Apps: serverless workflows that enable developers to integrate data with apps. No need to write complicated integration code between different systems.	An open-source tool Fantasm can be used for workflow.
API management	Amazon API Gateway is fully managed service which enables developers to create, publish, test, maintain, monitor and secure APIs at any scale.	API Management: to build, administer, monitor, and secure APIs at any scale. Azure Functions Proxies: enable developers to create microservice architectures by splitting monolith API surfaces into various function apps. Still, it presents single API interface to clients.	Cloud Endpoints: build, deploy, secure and manage APIs on any Google Cloud backend. Apigee: A cross-cloud API platform which enables control and visibility into the APIs.
Analytics	Amazon Kinesis is a streaming data platform to analyze real-time data. Amazon Athena enables SQL querying for data in S3.	Azure Stream Analytics: Real-time streaming data and SQL like language querying Event Hubs: to process, route, and store IOT devices data.	Cloud Dataflow: managed service for transforming and processing real-time data stream or in batch.
Intelligence	Amazon Machine Learning: service to perform predictions in real-time at scale. SageMaker: managed platform to build, train, and deploy ML models at scale. Amazon Rekognition Image: image and video analysis. Amazon Polly is a Text-to-Speech (TTS), Amazon Lex is a service to build conversational interfaces (bot).	Azure Bot Service: intelligent bot service with support to channels like Slack, Twitter, Skype, Microsoft Teams, Office 365 etc. Cognitive Services: enables vision, hear, speak, understand, and interpret user needs using natural methods of communication.	Cloud ML Engine: Serverless machine learning services built on Tensor Processing units. Other services like API.AI, Cloud Vision API, Cloud Speech API, Cloud Translation API. Google cloud is pretty rich in AI services.
Developer tooling	Frameworks: AWS Serverless Application Model (SAM) CI/CD: CodeStar, CodePipeline, AWS CodeBuild, and CodeDeploy Monitoring, Logging, and diagnostics: X-Ray to debug and trace, CloudWatch is a monitoring service also stores Lambda logs Integrated Development Experience (IDE): Cloud9 IDE, Plugin for Eclipse IDE and Visual Studio SDK: AWS SDK for Java, .NET, Python, Node.js, Go.	Frameworks: Serverless Framework: an open source application framework with a plugin for Azure Function CI/CD: Visual Studio Team Services Monitoring, Logging, and diagnostics: Application Insights: Service to monitor, log, and diagnose problems IDE: Visual Studio developer tools for Functions and Logic Apps. SDK: Azure SDKs and tools for all major platforms and languages.	Frameworks: Serverless Framework: an open source application framework with a plugin for Google Cloud Function CI/CD: Cloud Deployment Manager Monitoring, Logging, and diagnostics: Stackdriver Monitoring: provides a log of metrics, events, and metadata from Google Cloud Platform, Amazon Web Services. IDE: Cloud tools for all major IDEs. SDK: client libraries are available in Java, NodeJs, Python, .NET, Ruby, Go, and PHP.

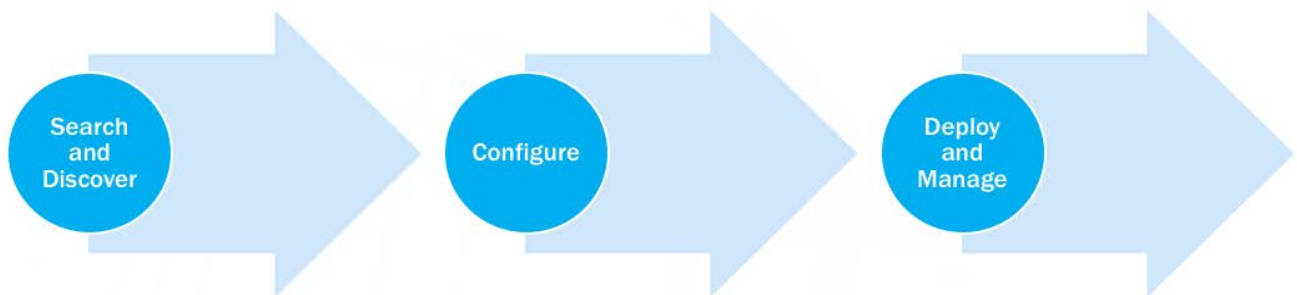


Figure 2. Application Development with AWS Serverless Application Repository

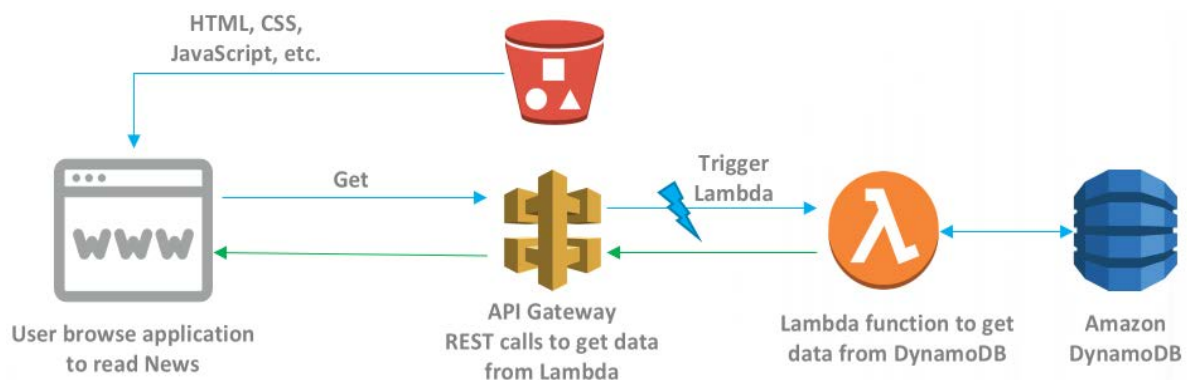


Figure 3. Serverless Web Application Architecture with AWS (adapted from [6])

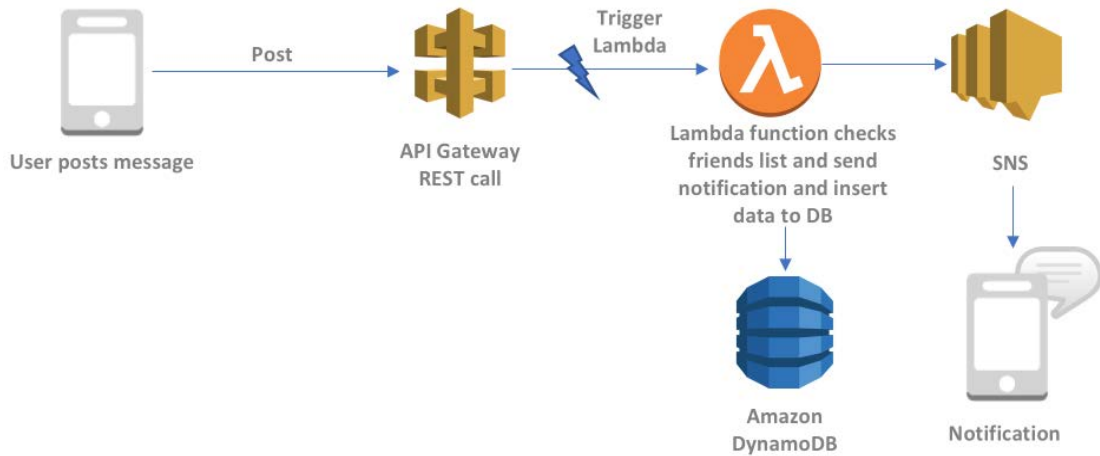


Figure 4. Serverless Web Application Architecture with AWS (adapted from [3])



Figure 5. Photo Thumbnail Creation with AWS (adapted from [3])

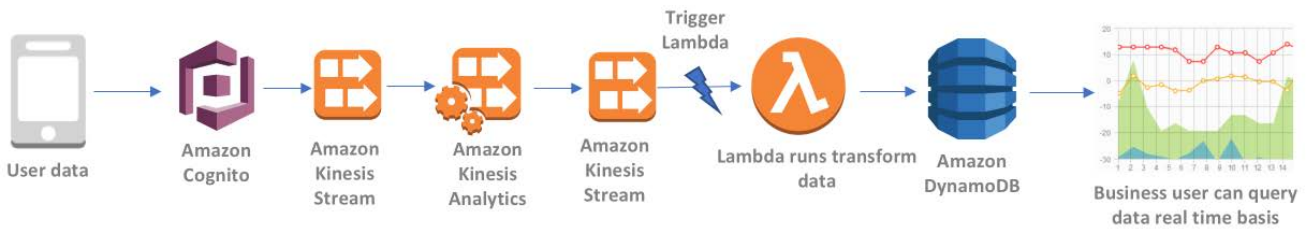


Figure 6. Analysis of Streaming data processing with AWS (adapted from [3])

2.1.1.3. Scheduled Tasks, Automated Backups, and log analysis

Lambda is great for scheduling tasks, log and build scheduling. Figure 7 shows log collection process and log analytics process using kinesis and Elasticsearch.

As a complete serverless web application, developer can envision something like as in Figure 8. In this application website code is hosted at S3. When a browser makes a call for webpage, Web page code checks for authentication token in case of new login call redirects to Amazon Cognito for authentication. Once web page gets

token it makes a call to Lambda using API Gateway. Lambda gets data from DynamoDB and returns back to browser through API Gateway.

2.1.2. Azure

Microsoft Azure [4] holds second position in the cloud market share. They provide a good set of toolsets in serverless space. Azure can be the best choice if respective organization is already using a significant number of Microsoft software.

Some of the following use cases and architectures recommended by Azure.

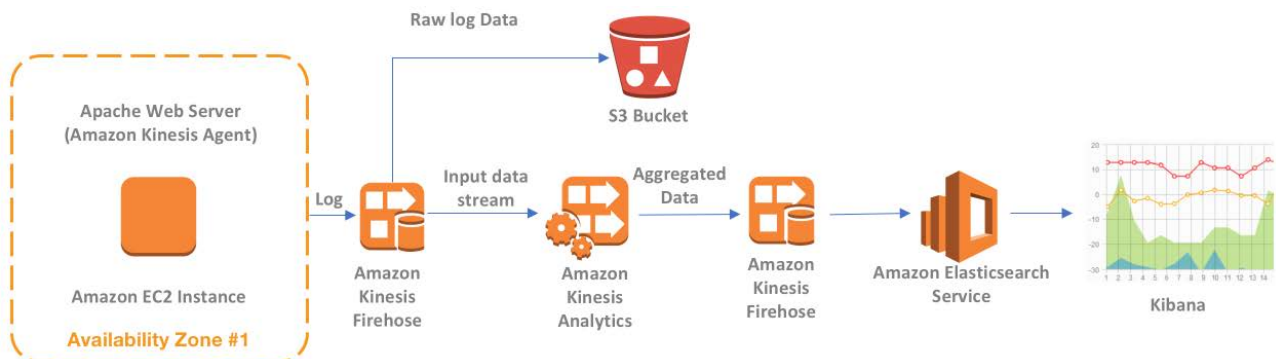


Figure 7. Log Analysis with AWS (adapted from [7])

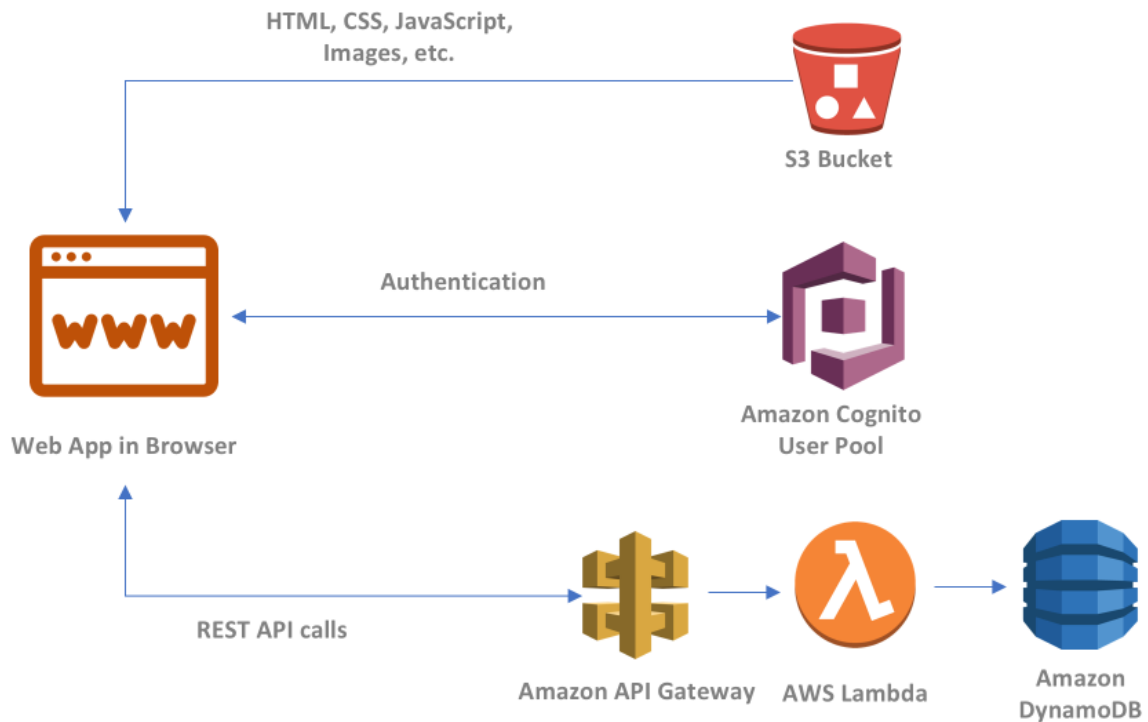


Figure 8. Complete Serverless Web Application with AWS (adapted from [6])

2.1.2.1. Web Application Architecture

Azure Functions are FaaS offering from Azure and can be exposed using WebHook URL in order to work as a microservices. This is good architecture to perform CRUD operations of a single web page. Figure 9 demonstrates targeted advertising app as per the user preferences.

2.1.2.2. IoT backend

Azure Stream Analytics receives messages from the Internet of Things (IoT) devices then it calls an Azure

function to process, transform and insert in Azure CosmosDB as describes in Figure 10.

2.1.2.3. SaaS integration

Functions support triggers based on activity in a Software as a service (SaaS)-based application. Following Figure 11 demonstrates an example to save a file in OneDrive, which invokes an Azure function. In the function, we can modify excel and can create analytical charts using Microsoft Graph API.

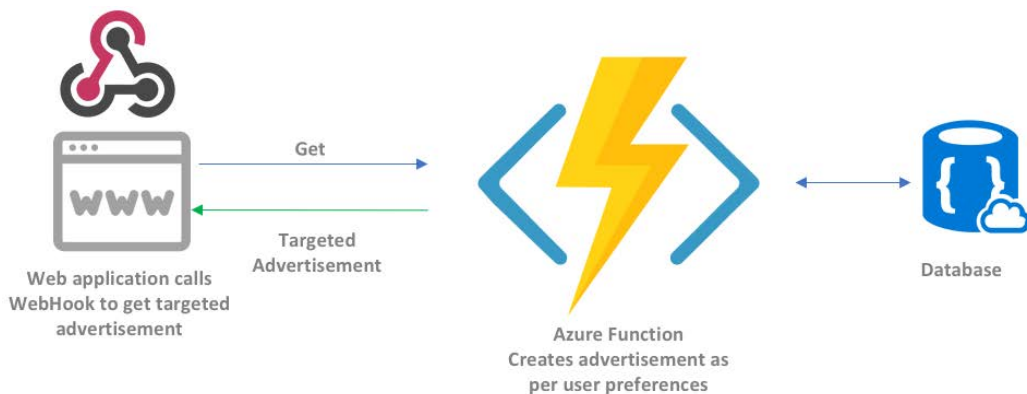


Figure 9. Azure Serverless Architecture for Targeted Ad (adapted from [4])

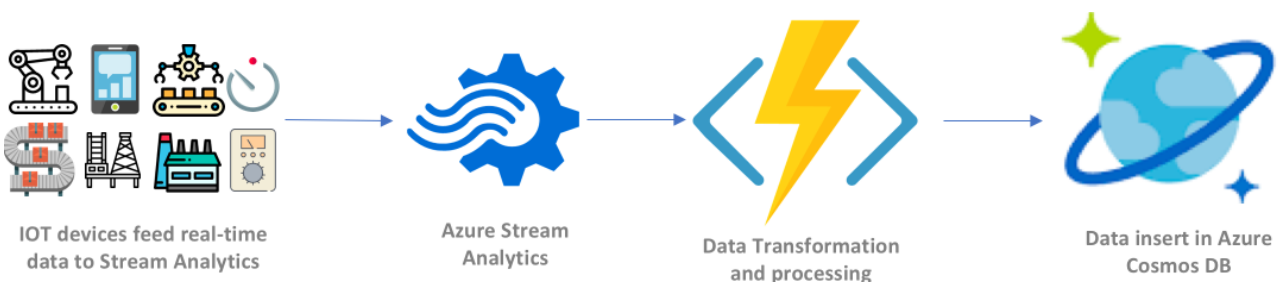


Figure 10. Azure Serverless Architecture for Real-time Stream Processing (adapted from [4])

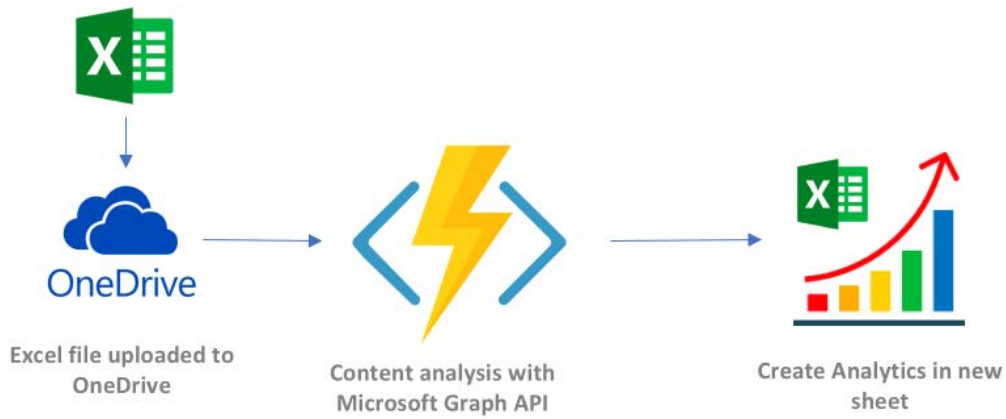


Figure 11. Azure Serverless Architecture for SaaS Add-on (adapted from [4])

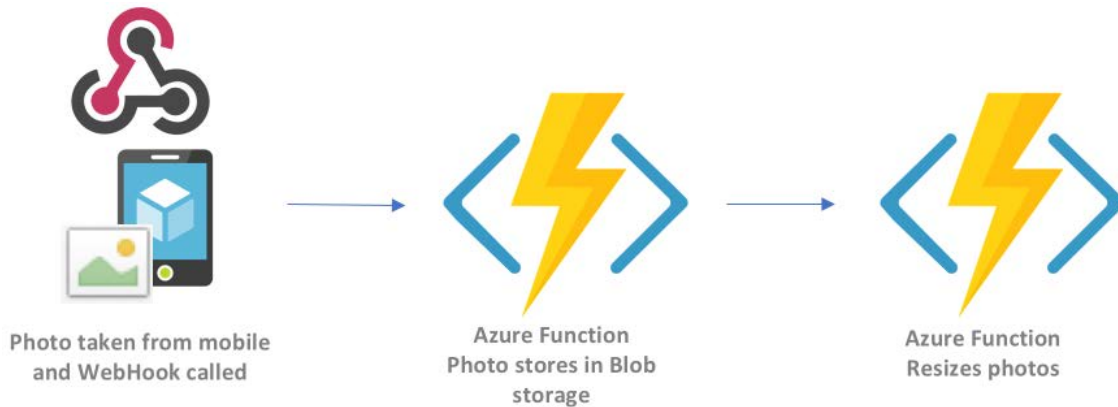


Figure 12. Azure Serverless Architecture for Mobile (adapted from [4])

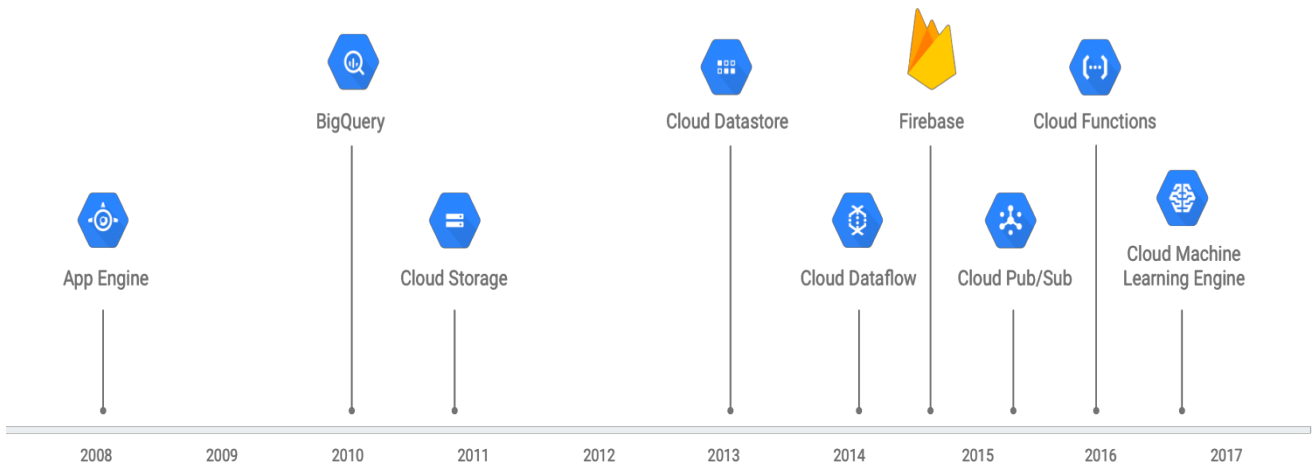


Figure 13. Google Serverless Evolution [5]

2.1.2.4. Mobile backend

Azure Functions support HTTP triggers and output bindings. These HTTP triggers can be customized to respond to WebHooks and work like APIs. These APIs work as a backend to mobile apps. Figure 12 demonstrates an example, a mobile phone app captures the image and calls the Azure function to get authorization token to upload to blob storage. Another Azure function resizes image and uploads to blob storage.

2.1.3. Google

Google [5] is a third major player in the cloud market share. It also has a wide variety of serverless offerings.

Over the years, Google Cloud Platform (GCP) has launched several serverless products covering application development and analytics. Figure 13 gives high level evolution of serverless in GCP.

We can summarize Google serverless applications in three most used use cases.

- Web Backend: Browser → App Engine → Datastore
- Microservice: Microservice → Cloud Function → Datastore
- ETL: File → Cloud Dataflow → BigQuery

Following Table 2 shows common use cases with Google serverless technology.

Table 2. Serverless with Google [5]

USE CASE	SERVERLESS WITH GOOGLE
Mobile apps	Firebase/Firestore
Web clients	Firebase/Firestore
Web backend	App Engine -> Datastore
Microservices	Cloud Functions -> Datastore
Data Processing	Cloud Functions/Cloud Function for Firebase
Bots	Cloud Functions
IoT device messages	Cloud Pub/Sub -> Dataflow
NoSQL database	Cloud Datastore
ETL	Cloud Dataflow -> BigQuery
Blob file storage	Cloud Storage
Analytics warehouse (SQL)	BigQuery
Personalization	Cloud Machine Learning Engine

Following Figure 14 shows serverless architecture using Firebase and Google Cloud Function.

In this example python program is reading sensor data and inserting in the Firebase database using HTTPs call. Firebase updates all the devices in real-time. Firebase

update trigger notifies cloud function. In case of critical values of sensor data, it notifies the user using Firebase Cloud Messaging (FCM).

2.1.4. Open Source

IBM’s OpenWhisk [8] is event-driven compute service (FaaS) which enables the developer to create actions without worrying about server provisioning in IBM Bluemix. IBM Bluemix (now is IBM Cloud) is the place for developers to create, deploy and manage application in the cloud. IBM Bluemix provides startup kit for most of the application components with easy integration. It is the place for rapid development and deployment of the applications. Following Figure 15 shows IBM Cloud Functions (based on Apache OpenWhisk) Serverless Architecture for a web application.

‘Serverless Framework’ [9] is open source toolkit to develop and build web, mobile and IoT applications using AWS Lambda, Azure Functions, Google Cloud Functions and IBM OpenWhisk. It is very useful for rapid development, testing, and deployment in a single environment to supported cloud providers. No need to worry about infrastructure provisioning and scaling. Following Figure 16 shows quick hello-world FAAS deployment.



Figure 14. Real-Time Update Serverless Architecture for The Mobile Application

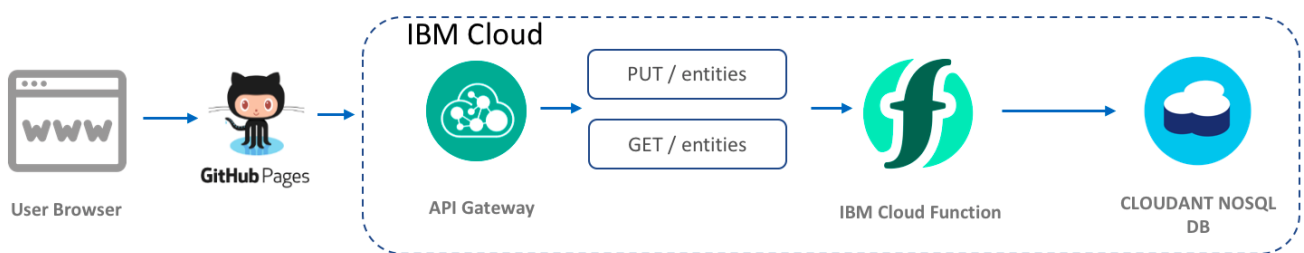
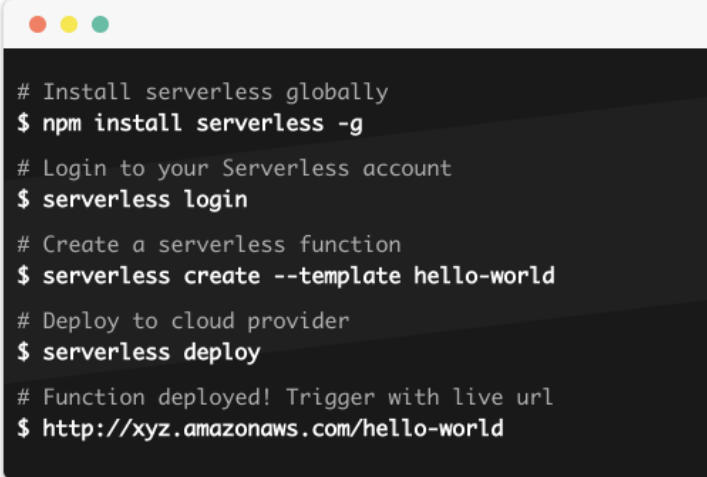


Figure 15. IBM Cloud Functions Serverless Architecture for Web Application (adapted from [8])



```

# Install serverless globally
$ npm install serverless -g

# Login to your Serverless account
$ serverless login

# Create a serverless function
$ serverless create --template hello-world

# Deploy to cloud provider
$ serverless deploy

# Function deployed! Trigger with live url
$ http://xyz.amazonaws.com/hello-world

```

Figure 16. Serverless Framework Commands [9]

There is an open source project name 'APEX' [10] enables developers in managing AWS Lambda. It empowers developers to develop Lambda functions in any language even that is not supported by AWS. APEX project contains project.json configuration file, that has the definition of Lambda functions. Deployment can be performed by just running simple command '\$ apex deploy'. It also provides tooling for testing, rollback, tailing logs, etc. Another related project 'APEX Up' is very useful if requirement is to develop a web application, API or static website. Basically, it provides an abstraction to Lambda.

Cloud Functions for Firebase allow developers to invoke backend code on Firebase feature events or HTTPS requests. The code is put away in Google's cloud and keeps running in an oversight situation as FaaS.

For Google Cloud Platform developers, Cloud Functions serve as a connective layer allowing the developer to weave logic between Google Cloud Platform (GCP) services by listening for and responding to events.

For Firebase developers, Cloud Functions for Firebase provides a way to extend the behavior of Firebase and integrate Firebase features through the addition of server-side code.

3. Benefits and Open Problems

Serverless technology is evolving so fast, many of the drawbacks of today will not be applicable after some time. There are several reasons to bat for serverless technology. Serverless can be the best fit for seasonal apps like tax filing, games, and chat apps.

Pros

- Scalability: Imagine game Pokémon GO, Is it possible without cloud. With serverless scaling is no limit. Provisioning infrastructure is high risk. Applications should scale or shrink as per demand.
- Low cost: User pays only for the workload what they run. No need to pay for the ideal time. No upfront set up cost.
- No infrastructure maintenance: In serverless approach user do not need to own any infrastructure.
- Easy deployment: Usually most of the cloud

providers provide command-line interface (CLI), console and cloud formation capabilities to deploy code in minutes.

- No infrastructure security required: As infrastructure is totally maintained by cloud provide so it's zero responsibility on that side.
- Availability to run close to the end user: Developer can improve latency by using edge location like AWS Lambda@edge.
- Faster time to market: Developers can create applications in hours' time in place of weeks or months. Developers can use cloud-optimized vast ecosystems like authentication service (AWS Cognito, Auth0, Azure Active Directory Federation Services, OAuth 2 Google service). AWS Kinesis and Azure Event Hubs are Kafka-like messaging/streaming systems which allow the developer to analyze and react on data at real-time basis. Several things can be tried in a couple of hours.

Cons

- Latency: For high-performance applications FaaS startup time might be significant. Cloud providers run containers whenever it is required to run the workload. However, startup time is not significant.
- Debugging and monitoring are complex in general in Serverless apps. Configuration management is not mature enough, hard to work in the team.
- New Technology: Still serverless technology is pretty new and not proven on large scale enterprise applications.
- Security Problems: Monitoring, support, maintenance, and testing of serverless applications are complex. Which leaves vulnerabilities open also it provides hacker wide attack surface due to highly distributed nature.

"According to the audit of more than 1,000 serverless apps by Israeli security firm PureSec, a fifth of them have critical security flaws. Most of these vulnerabilities were caused by copying and pasting insecure sample code into real-world projects, poor development practices, and lack of serverless education. The vendor has published a paper describing common errors to avoid. – NETWORK WORLD" [11].

- Vendor lock-in: This is the main concern and impediment in even cloud adoption. Serverless components are directly written in the provider’s environment. It is really complex to move these applications from one provider to another.
- No local data storage: FaaS is a stateless approach which does not retain any local data, so developers need to store data in the database only.
- No control to the underlying platform: Developer does not have access to the underlying environment in case they need to interact with the operating system.
- Execution Limit: There is an execution limit for programs to run. AWS Lambda has the hard limit of 5 minutes.

4. Future of Serverless and Solutions to Problems

Serverless systems are still in their infancy. Common standards such as platform independent function skeleton, the structure of passing event parameters, constant declaration, permission declaration, secret key management, and other configuration declaration are missing. In coming years, we will see more standardization around this. We can expect some solutions to the common pain points. Let’s talk about these pain points and solutions.

4.1. Serverless Application Development, Configuration, Debugging, Logging, Monitoring, and Deployment

Building, configuring, debugging, logging, monitoring, and deployment is the most critical problem in the serverless technology adoption. In coming years some standards will be set across cloud platforms as all the top cloud providers are pitching for it. These efforts should result in some kind of platform agnostic serverless platform. All cloud providers should support open source IDE for debugging, deployment and configuration. API Gateway configuration should be streamlined and should be configured through the simple configuration file. If developer deploys from IDE, it should create Lambda and API gateway. Good to see AWS has taken initiative in this direction and started building support tooling for its Lambda-based serverless infrastructure. At Serverless

Conf in Brooklyn 2016, Tim Wagner, general manager of AWS Serverless Compute (AWS Lambda, and Amazon API Gateway) announced “Flourish, a new open source project to create a full serverless runtime application environment” [12]. Hope, these things eliminate major hurdles in tooling. In the next 2-3 years, we can see the wide adoption of serverless.

4.2. Execution Duration, Startup Latency, and Session Management

First solution for this problem can be allowing Lambda to be instantiated on first application call with tentative active lifespan. Whenever any application starts and expected to call Lambda should activate the Lambda ahead of time for the tentative time span. Each client should pass unique globally unique identifier (GUID) in each call to identify Lambda instance and can be used for session management. If Lambda instance with GUID is not found or on first call its blank, new Lambda instance should be created and new GUID should be returned so that client can use for subsequent calls. If multiple Lambda instances are required multiple Lambdas with different GUIDs can be used. I agree in some cases client call will not find Lambda instance alive. That type of case can also happen in non-serverless environment. Figure 17 is demonstrating this logic.

Another solution can come at premium cost which can allow certain connections to be kept alive all the time and can be set as sticky endpoint.

Execution duration is usually enforced by cloud providers. Long running tasks can be broken down into reasonable chunks and can be run parallel in multiple lambdas.

4.3. Vendor lock-In

A common open source platform to develop and deploy on multiple cloud environments can eliminate vendor lock-in problem.

4.4. New Technology and Security

Over the time serverless technology will get mature and people will get more awareness. The developer will follow new evolved security guidelines in order to avoid vulnerabilities.

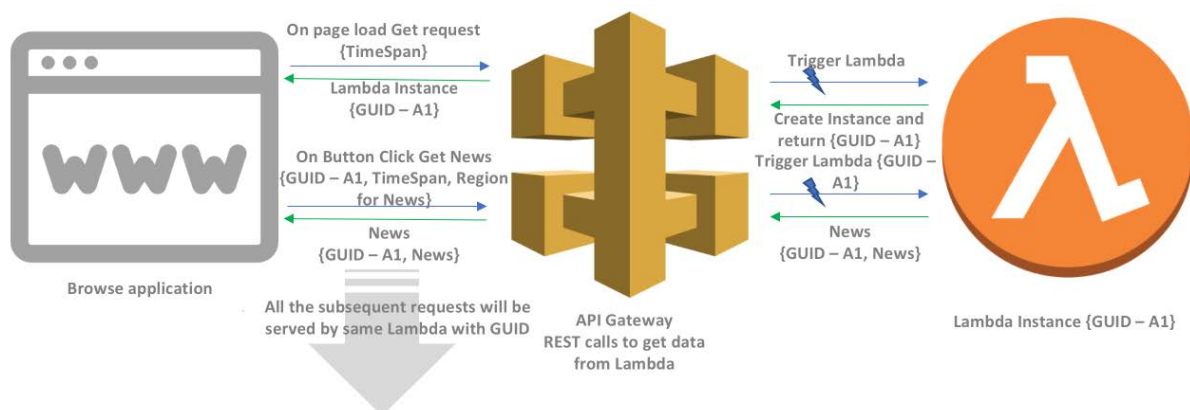


Figure 17. How to Solve Start-up Latency and Session Management Problems

4.5. Local Data Storage and Platform Access

These are not that critical. These requirements can be solved using the workarounds like storing in the database and for some special cases where platform access is required, containers can be used.

5. Conclusion

This paper provides a guide to technical audiences on how to best fit various serverless cloud component for the different kind of use cases and the serverless offerings from leading cloud providers. As we saw all cloud providers provide almost all kind of serverless components, but still AWS is leading the way. Google is good in Artificial Intelligence and mobile app use cases. Azure is well suited for companies who are already having a good establishment of Microsoft products.

After reviewing all these offerings in detail, it's very clear that AWS is much mature and have a variety of product line compared to others. AWS Lambda has several triggering points which provides more flexibility to architects. Lambda@Edge is really handy if it is required to run at edge location. GCF is not having this capability yet other hand Azure is limited to IoT only. Database and storage prospect all the three providers have the almost same capabilities. But still as per usability prospect, I liked GCP Firebase and it is quite famous in the mobile world. In messaging segment all three providers have good platforms. Workflow is the area where all the three need to go long way. I used AWS Step Functions which is quite promising but complex to work with. Till this time it is not allowing to modify workflow when deployed, developers need to recreate every time. This behavior is not developer friendly. There is K2 Workflow product which is having good workflow capabilities much advance than others. It is worth considering in workflow use cases. Azure is having the edge on SaaS add-on capabilities. In terms of AI and machine learning, Google is having a long history. It has some of the well-known services in Deep learning, TensorFlow, Virtual reality, and Augmented Reality. Everyone is looking for when open source platforms will come into existence that will allow developers and architects to choose appropriate serverless component

disregard of the cloud provider. Other hand Microservices are especially suited for serverless in terms of cost and performance. I will cover that area in next paper.

References

- [1] A. Mohamed, "A history of cloud computing," *Computer Weekly.com*, 2018. [Online]. Available: <https://www.computerweekly.com/feature/A-history-of-cloud-computing>.
- [2] M. Roberts, "Serverless Architectures," *MartinFowler.com*, May 22, 2018. [Online]. Available: <https://martinfowler.com/articles/serverless.html>.
- [3] Amazon Web Services, "Serverless Computing and Applications," AWS, 2018. [Online]. Available: <https://aws.amazon.com/serverless>.
- [4] Microsoft, "Serverless Computing," *Azure*, 2018. [Online]. Available: <https://azure.microsoft.com/en-us/overview/serverless-computing>.
- [5] Google, "Serverless," *GCP*, 2018. [Online]. Available: <https://cloud.google.com/serverless>.
- [6] Amazon Web Services, "Build Your First Serverless Web Application," AWS, 2018. [Online]. Available: <https://aws.amazon.com/serverless/build-a-web-app>.
- [7] Amazon Web Services, "Build a Log Analytics Solution," AWS, 2018. [Online]. Available: <https://aws.amazon.com/getting-started/projects/build-log-analytics-solution>.
- [8] IBM, "IBM Cloud Functions," *IBM Apache OpenWhisk*, 2018. [Online]. Available: <https://www.ibm.com/cloud/functions>.
- [9] Serverless, Inc., "The way cloud should be.," *Serverless*, 2018. [Online]. Available: <https://serverless.com>.
- [10] Open Source Collective 501c6 (Non Profit) - APEX Software, "APEX | Serverless Infrastructure.," *Github*, 2018. [Online]. Available: <https://github.com/apex/apex>.
- [11] A. Patrizio, "One in Five Serverless Apps has a Critical Security Vulnerability," *NetworkWorld*, Apr 12, 2018. [Online]. Available: <https://www.networkworld.com/article/3268415/security/one-in-five-serverless-apps-has-a-critical-security-vulnerability.html>.
- [12] M. Boyd, "Amazon Debuts Flourish, a Runtime Application Model for Serverless Computing," *TheNewStack*, May 26, 2016. [Online]. Available: <https://thenewstack.io/amazon-debuts-flourish-runtime-application-model-serverless-computing>.
- [13] A. Eivy, "Be Wary of the Economics of "Serverless" Cloud Computing," *IEEE Cloud Computing*, vol. 4, (2), pp. 6-12, 2017.
- [14] N. Bila et al, "Leveraging the serverless architecture for securing linux containers," in *2017 IEEE 37th International Conference on Distributed Computing Systems Workshops (ICDCSW)*, 2017.
- [15] T. Lynn et al, "A preliminary review of enterprise serverless cloud computing (function-as-a-service) platforms," in *2017 IEEE International Conference on Cloud Computing Technology and Science (CloudCom)*, 2017.

