Case Study :
Serverless Computing AWS Lambda

CSL7091 - Cloud Computing and Virtualization

Dr. Sumit Kalra
Shashwat Kathuria - B17CS050
Abhinav Pandey - B17CS001

Serverless Computing

Introduction

- Cloud Computing Model
- Cloud provider runs the server and dynamically manages the allocation of machine resources
- Pricing is based on the actual amount of resources consumed by the application, rather than on pre-purchased units of capacity
- A form of utility computing

Benefits

- Easy to Deploy
- Low Cost
- More time available for coding, UI, UX, etc
- Better scalability
- Improved latency
- Improved flexibility

AWS Lambda

Introduction

- Event - driven, serverless computing platform provided by Amazon
- A computing service that runs code in response to events and automatically manages the computing resources required by that code

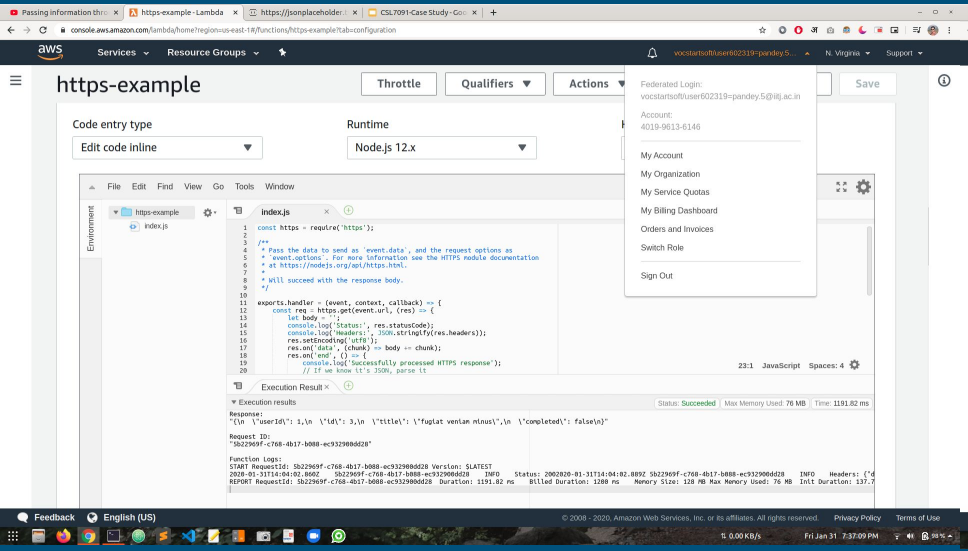
Features

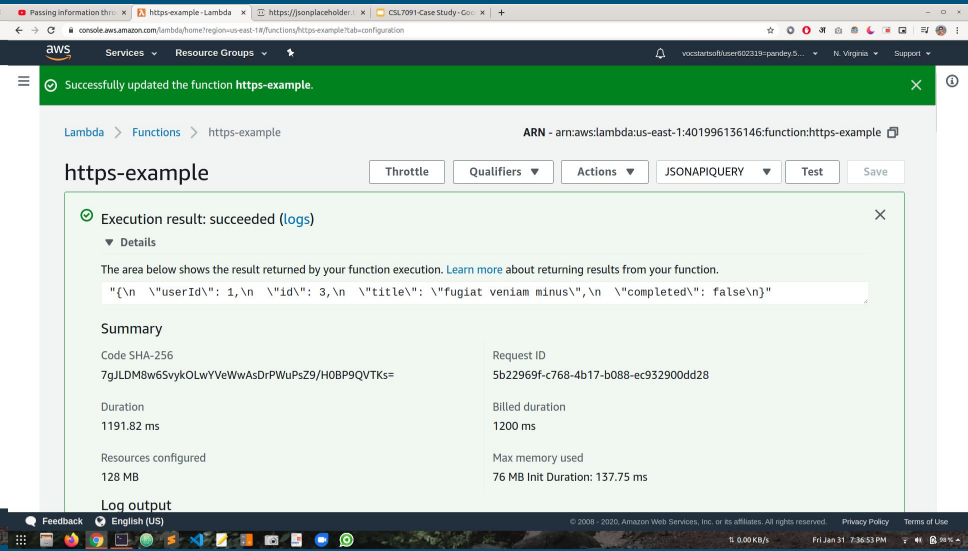
- No servers to manage - only write and upload code
- Continuous Scaling - scaling precisely with the size of the workload
- Subsecond metering - pay only for compute time
- Consistent performance - optimize code execution time

Our Case Studies

Used Lambda to **parse a response from an API Gateway** as an example

Very easy to **set up**, **scale** and **manage resources** with **only compute time cost**, **no servers to maintain**, everything done on cloud

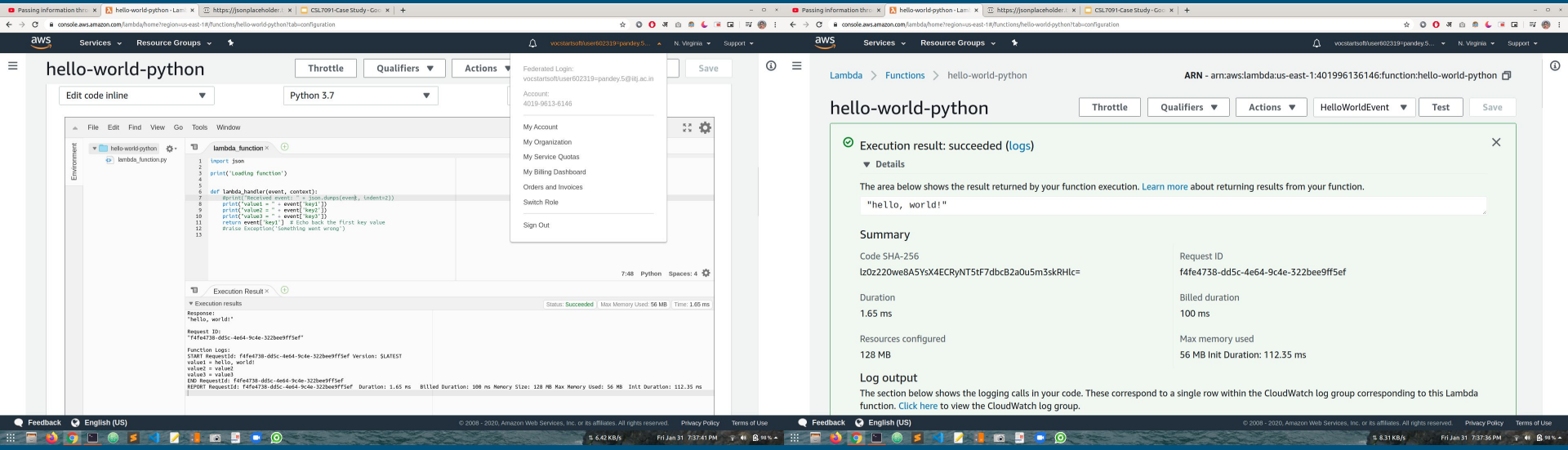




Our Case Studies

Used Lambda to **print a statement (or any metric as an example after processing event data)**

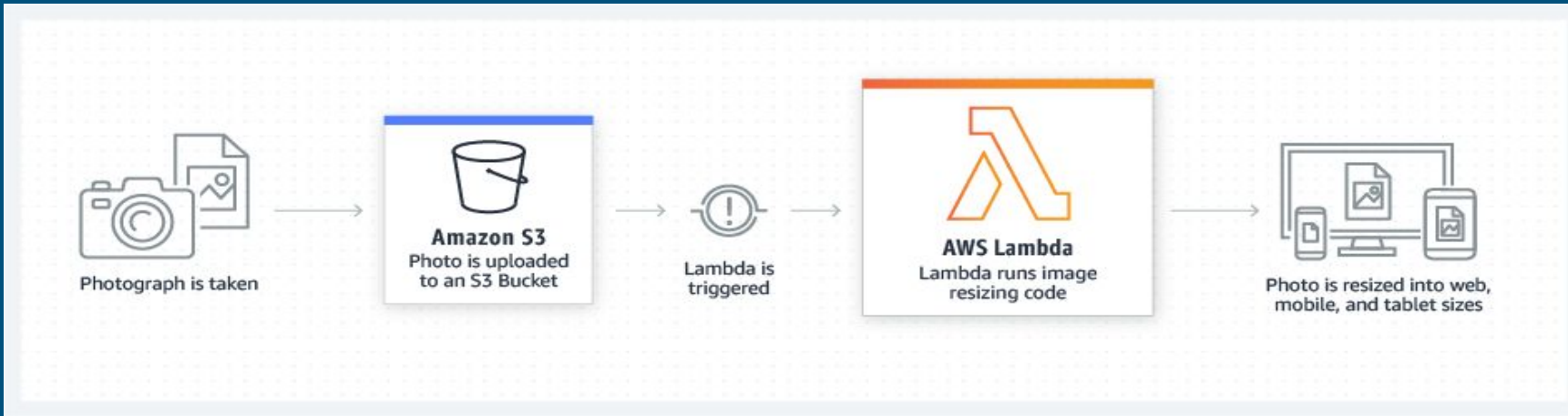
Very easy to **set up**, **scale** and **manage resources** with **only compute time cost**, **no servers to maintain**, everything done on cloud



Cons of Serverless Computing

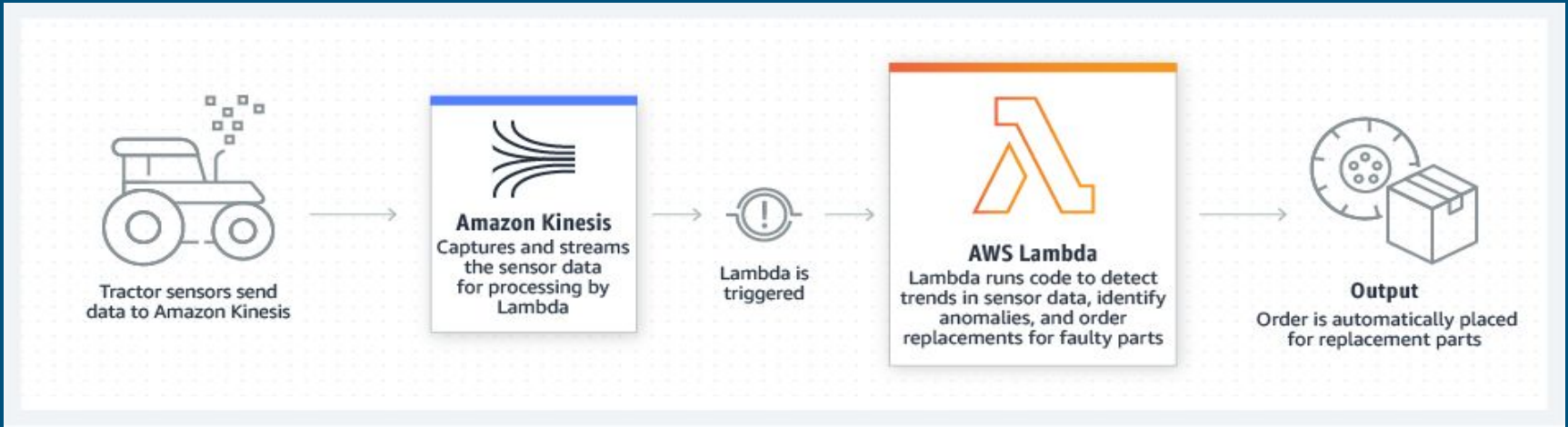
- ➔ **Vendor Lock-In**
  - Have to **conform to the rules** of the vendor
  - Currently **less freedom of languages** to choose from
- ➔ **Learning curve**
  - Learning curve for FaaS (Function-as-a-Service) tools is **pretty steep**
  - For easy migration, have to **split your monolith into microservices**, a **complicated** task
- ➔ **Unsuitable for long term tasks**
  - Lambda gives you five minutes to execute a task and if it takes longer, you'll have to **call another function**
  - Long duration operations such as uploading video files would require **additional FaaS functions** or be better with **"server-ful" architecture**

Real World Applications



An application of **Real Time File Processing**
Lambda **triggers on photo upload** and **runs resizing code**

Real World Applications



An application of **IoT Backend**
Lambda **triggers on sensor data collection** and runs **sensor maintenance code**

Real World Case Study

Context & Problem

- ➔ Wanted to update customer bonus points of their vending machines.
- ➔ The issue was that there needed to be introduced a delay for the same which built up costs due to delay of 90 seconds that was required for updating the records.

Solution

- ➔ Used AWS Lambda to update the bonus points and pay only for the consume time.
- ➔ Introduced AWS Step State of 90 second delay which was not considered in costs.
- ➔ Efficiently and cost effectively solved the solution without managing servers.

Real World Case Study

Context & Problem

- ➔ Wanted to scale their new connected IoT robot vacuum cleaners cloud application.
- ➔ The issue was that they wanted an easy solution and did not want to go up with subscription services and the headache of scaling and maintaining.

Solution

- ➔ Used AWS Lambda and AWS IoT platform to update their IoT backend and provide connectivity layer between the robots and the iRobot cloud platform.
- ➔ Were also able to compute the metrics of their devices.
- ➔ Scaled their applications in a hassle-free way by using serverless computing.

Real World Case Study

Benchling

Context & Problem

- ➔ Wanted to scale their applications and fasten their genome search queries (biology related).
- ➔ The issue was that their servers were taking a lot of time, on the other hand they were also expecting more and more users so they needed to scale and the servers were being used in off hours leading to higher costs.

Solution

- ➔ They split their searches across several AWS Lambda by parallelizing and storing genome heavy data in Amazon S3.
- ➔ Reduced search times by 90%
- ➔ Were able to increase the number of genomes
- ➔ Reduced their costs and increased scalability

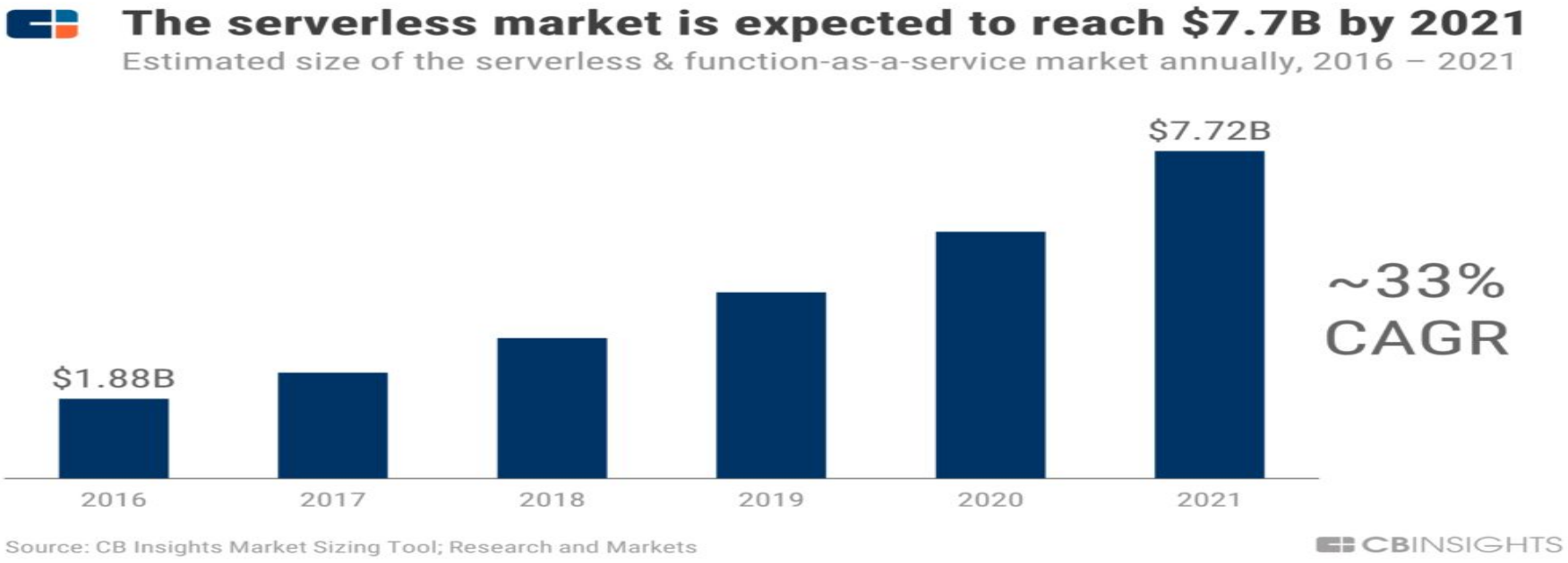
Competitors

Google Cloud Functions

IBM Cloud Functions

Azure Functions

Expected Trends



Thank You!