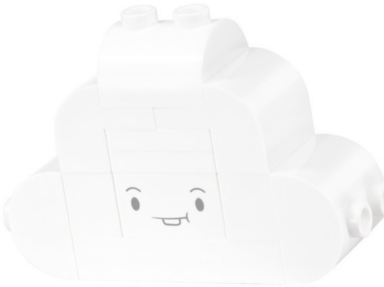
Serverless computing:
An overview

Andrea Lisi

Programming Tools and Techniques in the Pervasive Parallelism Era
Università degli studi di Pisa

Roadmap



1. Preliminaries: Cloud computing
2. Serverless computing service model
3. Serverless platforms
4. Conclusions

Preliminaries: Cloud computing

Cloud computing

Cloud computing is a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction.[NIST]

Cloud computing

Essential characteristics

- On-demand self-service
- Broad network access
- Resource pooling
- Rapid elasticity
- Measured service

Service Models

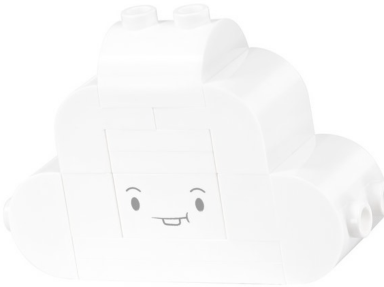
- Infrastructure - aaS
- Platform - aaS
- Software - aaS

Deployment Models

- Private cloud
- Community cloud
- Public cloud

[NIST]

Cloud computing



Essential characteristics

- On-demand self-service
- Broad network access
- Resource pooling
- Rapid elasticity
- Measured service

Service Models

- Infrastructure - aaS
- Platform - aaS
- Software - aaS

Deployment Models

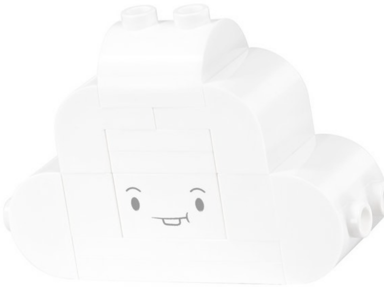
- Private cloud
- Community cloud
- Public cloud

[NIST]

Cloud computing

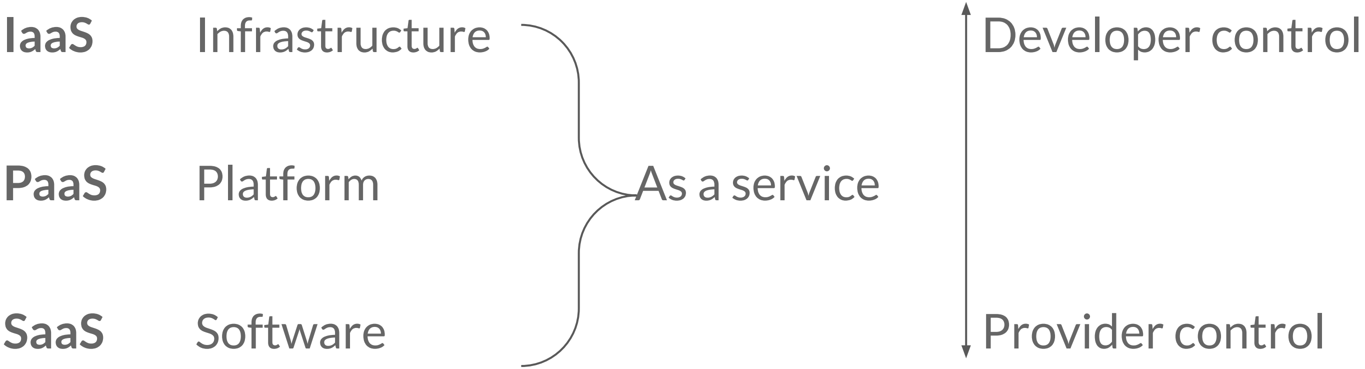
Main service models in Cloud computing:

IaaS	Infrastructure	As a service
PaaS	Platform	
SaaS	Software	



Cloud computing

Main service models in Cloud computing:



IaaS Infrastructure

PaaS Platform

SaaS Software

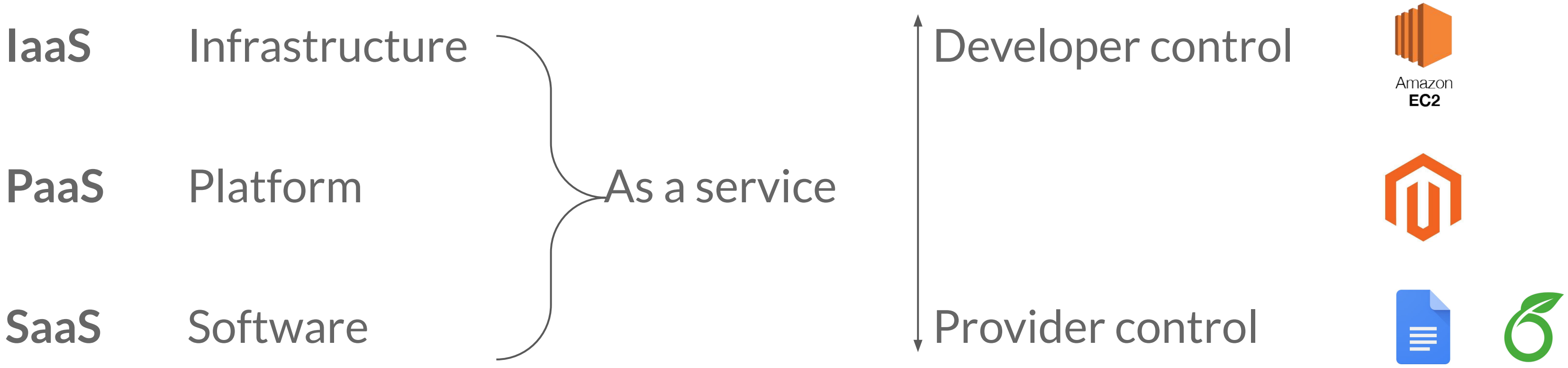
As a service

Developer control

Provider control

Cloud computing

Main service models in Cloud computing:



IaaS Infrastructure

PaaS Platform

SaaS Software

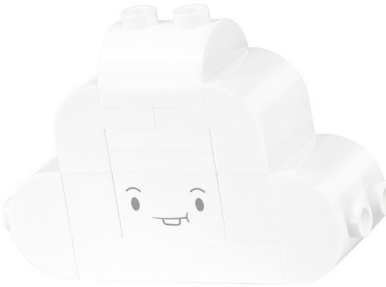
As a service

Developer control

Provider control

Amazon
EC2

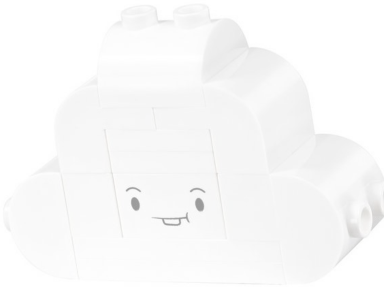
Cloud computing



	Service	Disadvantages
IaaS	The developer can customize aspects down to OS	Effort and time have to be spent over cloud platform management
PaaS	Cloud provider handles the resource management, the developer focuses only on the business logic	The customer is charged by resources allocation, also when idle
SaaS	A customer subscribe and utilize an application fully controlled by the provider	Execution of user-provided functions limited to the application domain

Serverless Computing

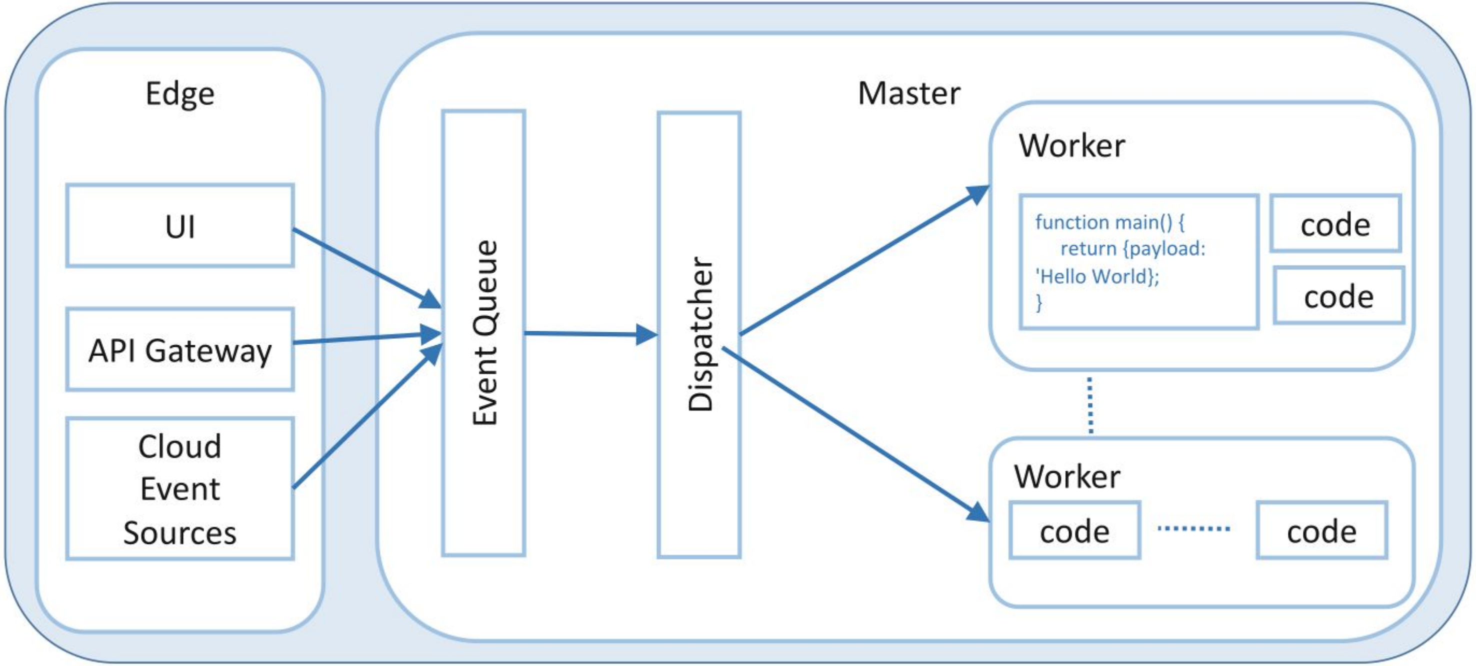
Serverless computing



Serverless computing is a service model similar to PaaS

- Server management is delegated to the cloud provider (server**-less**)
- It is based on stateless computation
- It follows a event-based logic
  - Triggers and Actions

Architecture



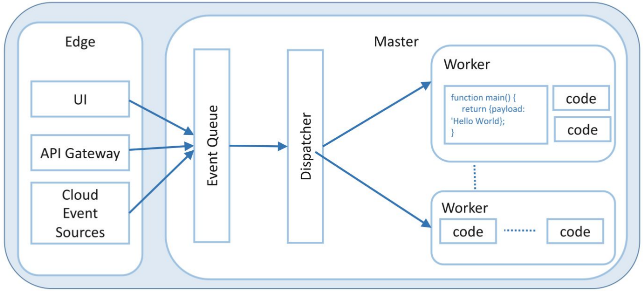
[Baldini]

Serverless computing

As a developer:

- Adopt a framework
- Identify the event sources
- Write the code for every action
- Connect events to the correct code

No think about resource management

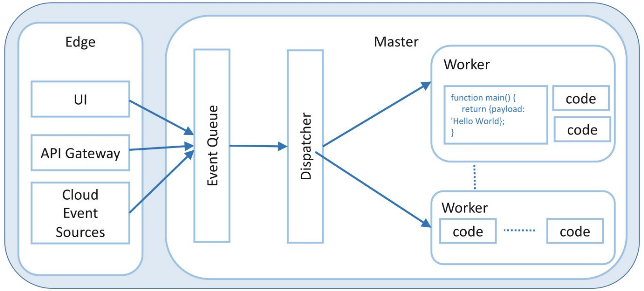


```
function main() {
    return {payload:
'Hello World};
}
```

Serverless computing

As a cloud provider:

- Queue up incoming events
- Efficiently manage workers lifecycle
  - ○ Start-up, execution, de-allocation
- Auto-scale the workers to satisfy the needs of the application
- Manage failures in a cloud environment



```
function main() {
    return {payload:
'Hello World};
}
```

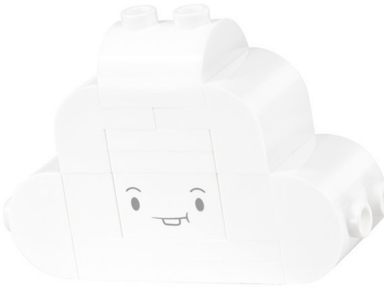
Serverless computing

If the core deployment unit is a function, the service model is typically known as **Function as a Service** (FaaS)

- Server management is delegated to the cloud provider (server-**less**)
- It is based on stateless functions
- Functions follows a event-based logic
  - Events and Callbacks

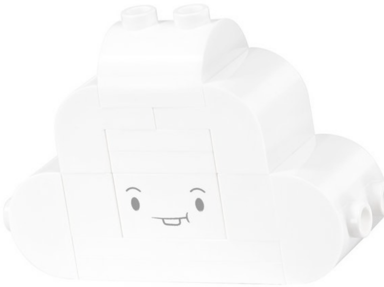
However the distinctions between the twos can be fuzzy[LiquidWeb] [Geoffrey]

Functions



The goal is to break down a monolithic system into a set of independent processes similar to microservices

- Each process is associated to an event and a function
    - The event might be a HTTP request
    - The function executes the logic to satisfy that request
- Composing processes accordingly creates the whole application

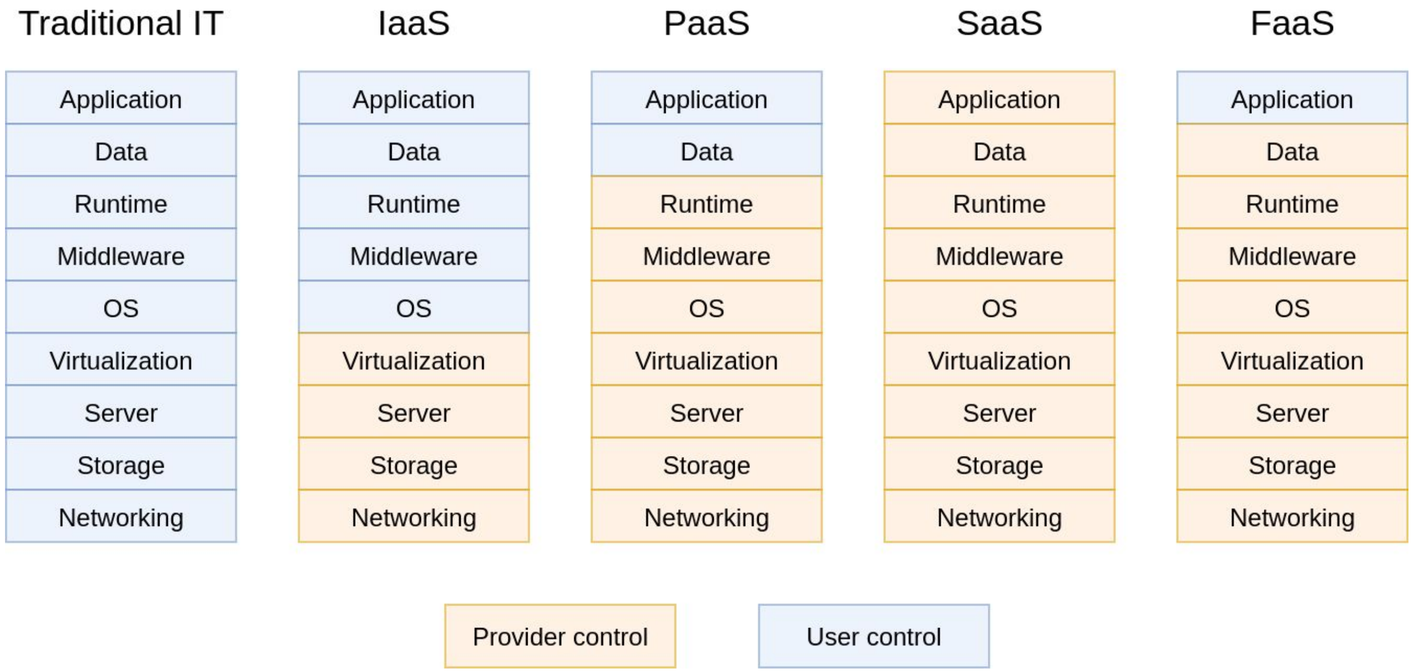


Functions

Serverless functions do not rely on a machine state

- Is not guaranteed the second invocation of the same function will start with a state stored during its first execution[Flower]
- It is possible to run multiple instances without race conditions issues and safely scale up on-demand
- If a state is needed, it should be externalized

Comparison - 1



Traditional IT	IaaS	PaaS	SaaS	FaaS
Application	Application	Application	Application	Application
Data	Data	Data	Data	Data
Runtime	Runtime	Runtime	Runtime	Runtime
Middleware	Middleware	Middleware	Middleware	Middleware
OS	OS	OS	OS	OS
Virtualization	Virtualization	Virtualization	Virtualization	Virtualization
Server	Server	Server	Server	Server
Storage	Storage	Storage	Storage	Storage
Networking	Networking	Networking	Networking	Networking

Provider control

User control

Comparison - 2

Each cloud service model adopts different pricing models[Laatikainen]

- IaaS: pay for the cloud infrastructure
- PaaS: pay for the executable environment
- SaaS: subscribe to the service
- FaaS: pay only the execution of the function on demand
  - Unlike in PaaS, idle time is not charged
  - However the function initialization is charged, and it can cost a consistent amount if a function is frequently de-allocated

Functions

t0 ↑ 10 Requests

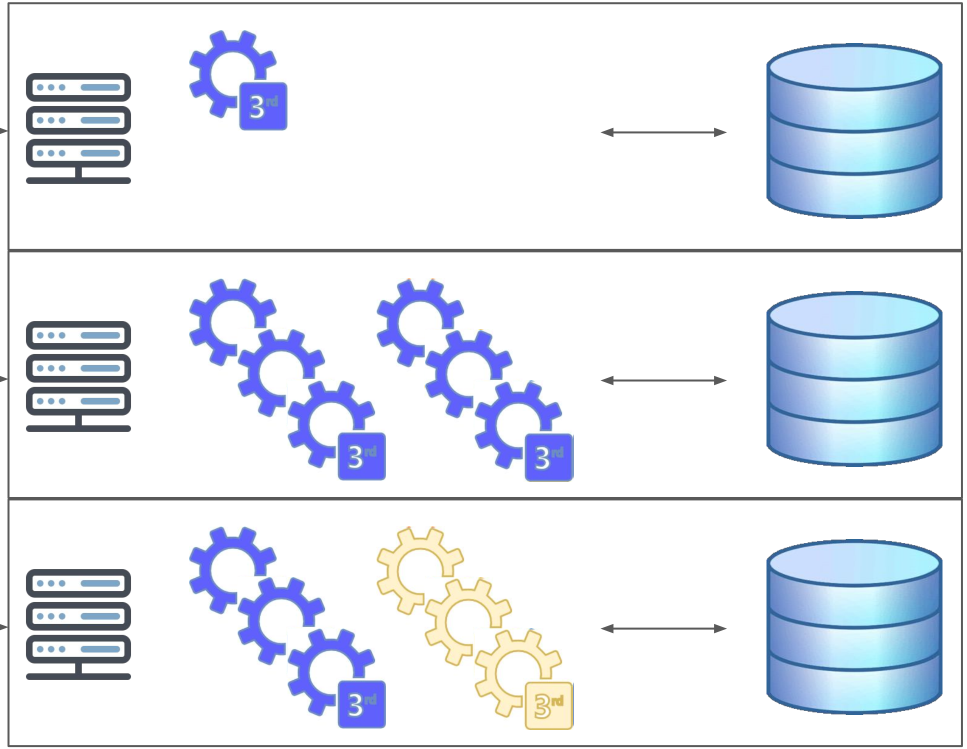
+ 10 start-ups
+ 10 executions

t1 ↑ 60 Requests

+ 50 start-ups
+ 60 executions

t2 ↓ 30 Requests

+ 30 executions

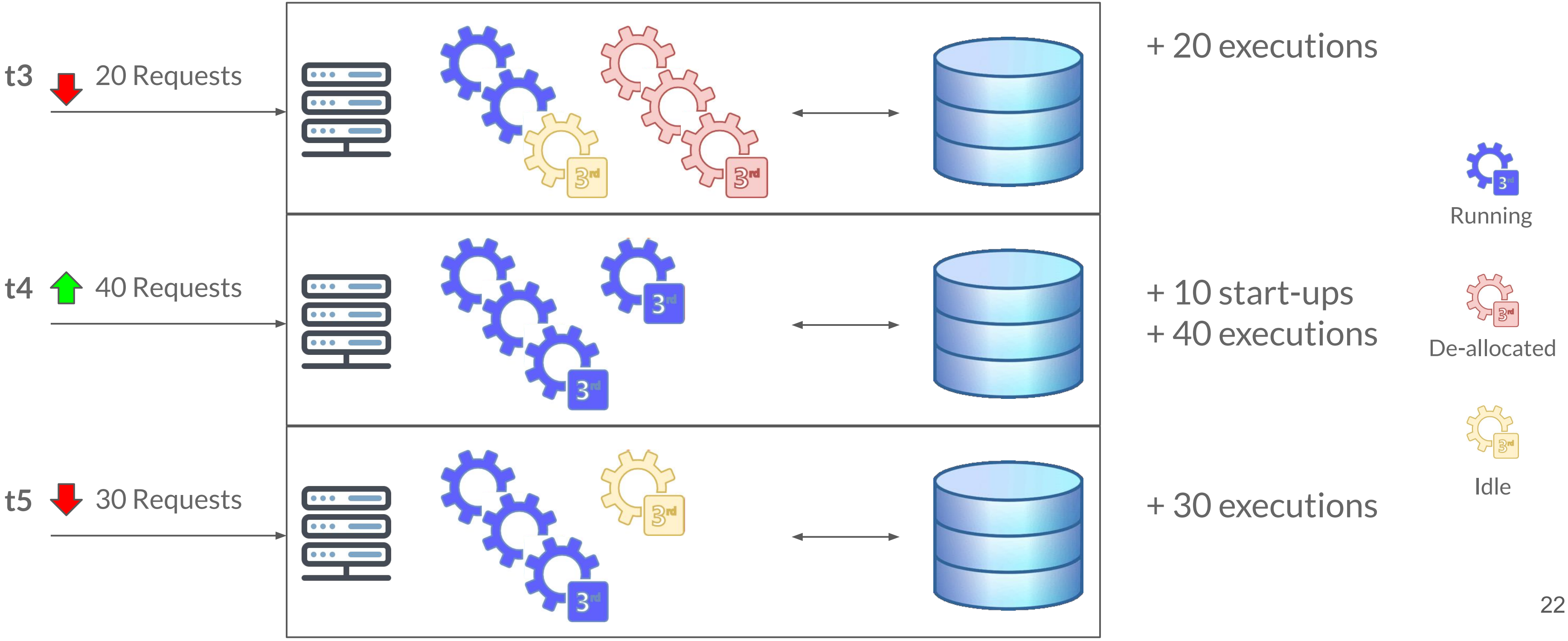


Running

De-allocated

Idle

Functions



t3 20 Requests

+ 20 executions

t4 40 Requests

+ 10 start-ups
+ 40 executions

t5 30 Requests

+ 30 executions

Running

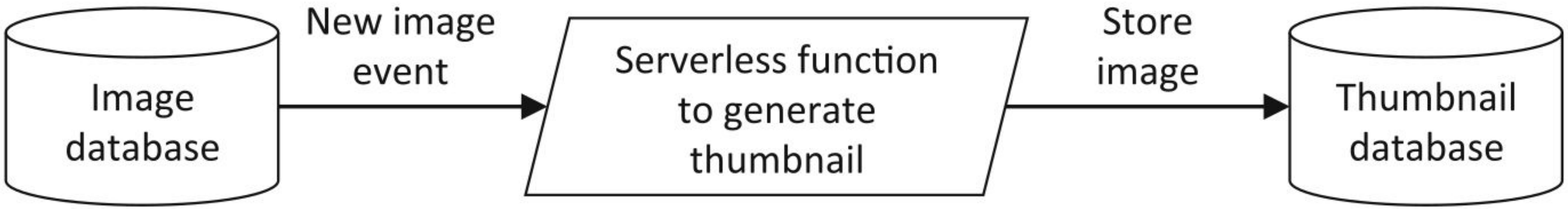
De-allocated

Idle

Use cases

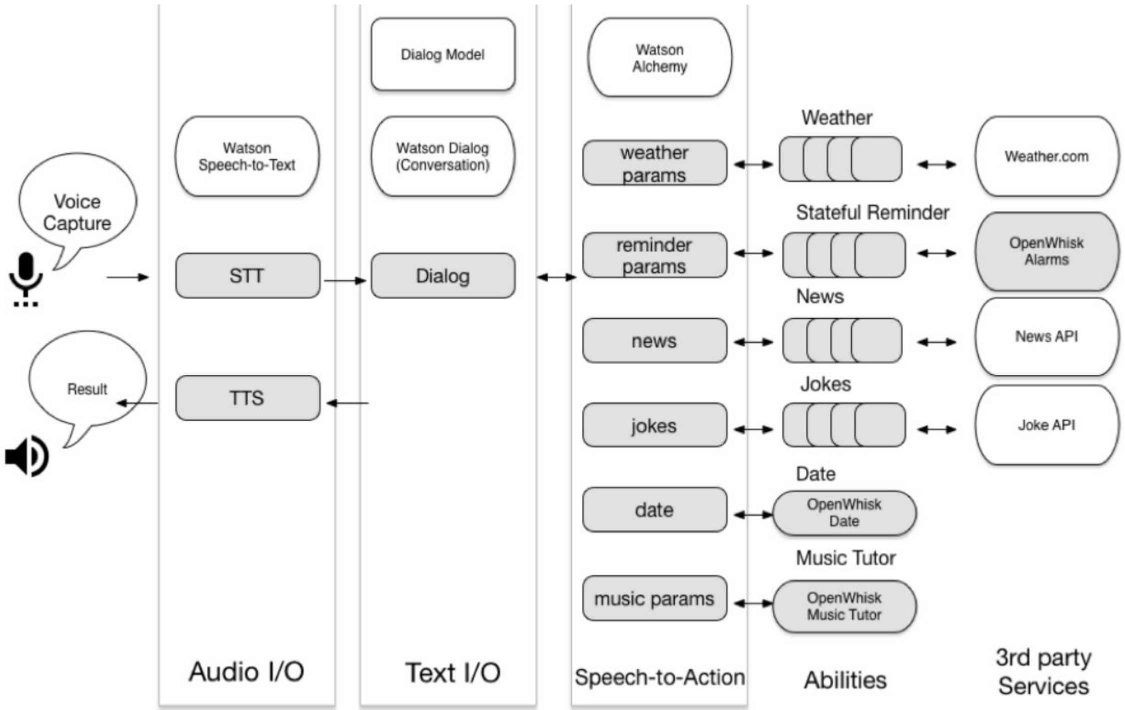
Use cases for serverless computing include event-based single task computation

Example: thumbnail production from large image



[Baldini]

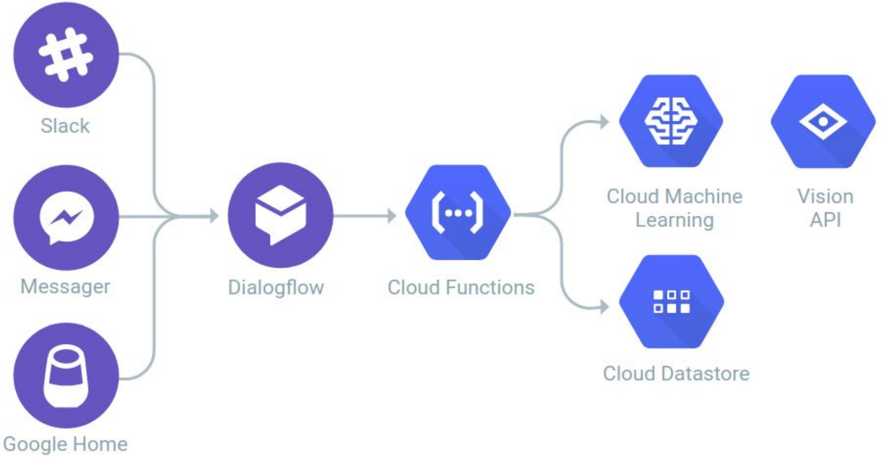
Use cases



[Yan]

Use cases

Many uses cases that go under the roof of “Real-time data processing”, “Backend functionalities” and “third-party APIs integration”



[Google]

Serverless Platforms

Serverless platforms

Cost

Performance and limits

Programming languages

Programming model

Composability

Deployment

Security and accounting

Monitoring and debugging

[Baldini]

Serverless platforms

Customers rent from providers the infrastructure and the service of resource allocation and management

- Functions allocation, concurrency and networking
- Customers need to write functions, declare the resources (MB per function) and upload them
- Customers are charged for actual code execution in ms
    - The function start-up time is charged as well (cold-start)

Google Cloud Functions

Amazon Lambda

Amazon Lambda

Connect to the runtime an **handler** to a function

An handler has an **event** and **context** (e.g. request ID) as input

A code example[AmazonLambda]:

```
exports.handler = async function(event, context) {
 console.log("ENVIRONMENT VARIABLES\n" + JSON.stringify(process.env, null, 2))
 console.log("EVENT\n" + JSON.stringify(event, null, 2))
 return context.logStreamName
}
```

Amazon Lambda

A function can be invoked when needed

- Sync, or async invocation
- A second invocation
  - Will be served by the same instance if available
  - Otherwise a new instance will be created
- New instances can be created until the concurrency limit has been reached
  - Different strategies to handle concurrency

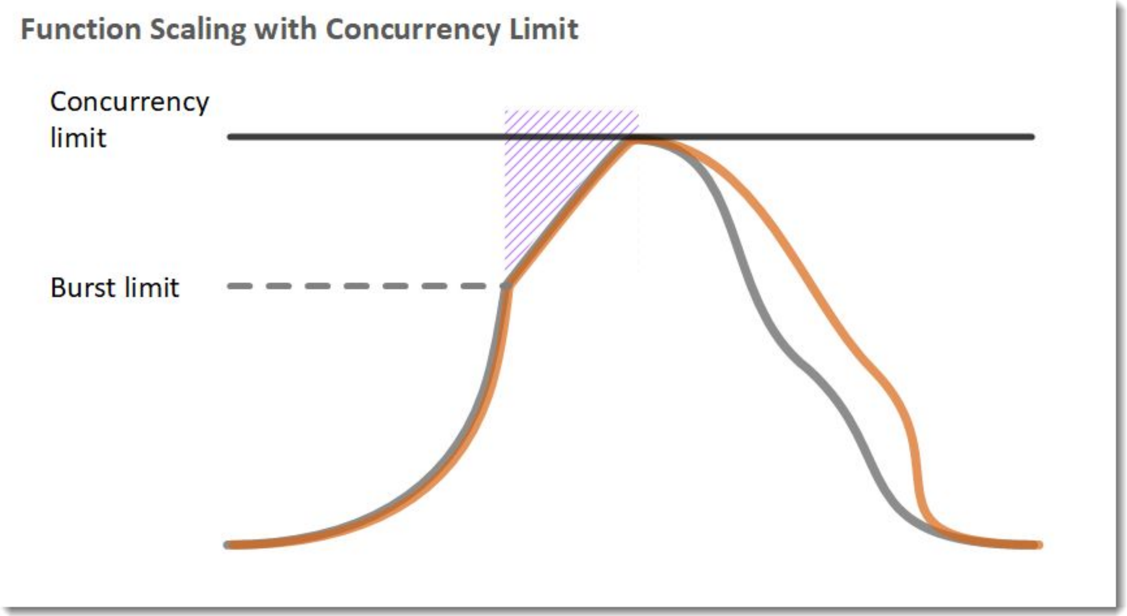
Amazon Lambda

Gray:

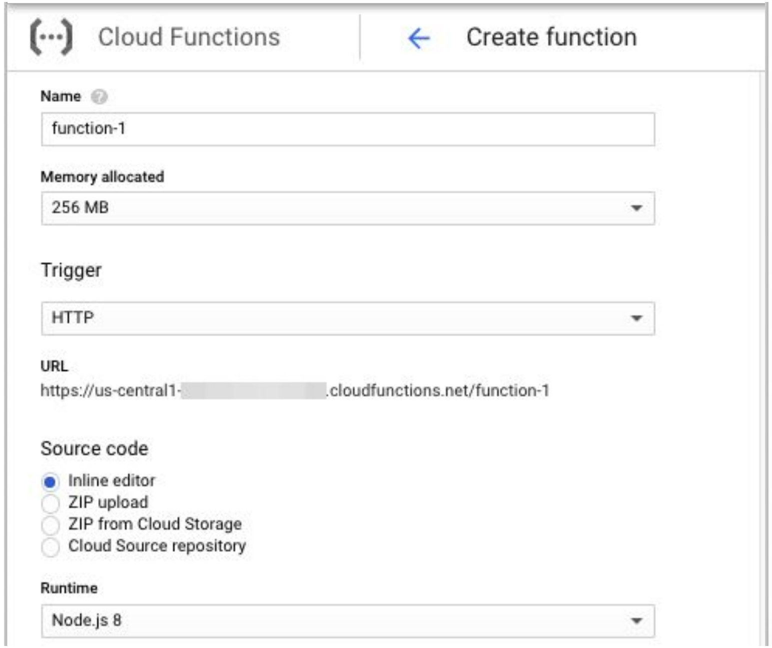
requests

Orange:

instances



Google Functions



Name
function-1

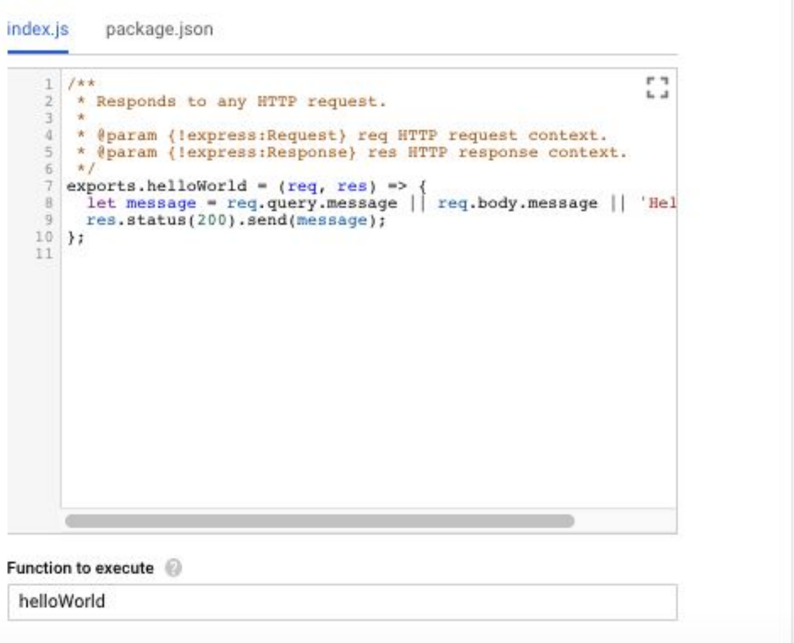
Memory allocated
256 MB

Trigger
HTTP

URL
https://us-central1-[redacted].cloudfunctions.net/function-1

Source code
● Inline editor
○ ZIP upload
○ ZIP from Cloud Storage
○ Cloud Source repository

Runtime
Node.js 8



```
/**
 * Responds to any HTTP request.
 *
 * @param {!express:Request} req HTTP request context.
 * @param {!express:Response} res HTTP response context.
 */
exports.helloWorld = (req, res) => {
  let message = req.query.message || req.body.message || 'Hel
  res.status(200).send(message);
};
```

Function to execute
helloWorld

Serverless platforms

Open source solutions provide the developer a framework for the management of the actions (functions) and events

- Flexibility and customization
- The infrastructure is responsibility of the developer, e.g. machines and VM[Flower]
    - Or rent from cloud providers
- They typically rely on Kubernetes for the orchestration of the functions[Li]

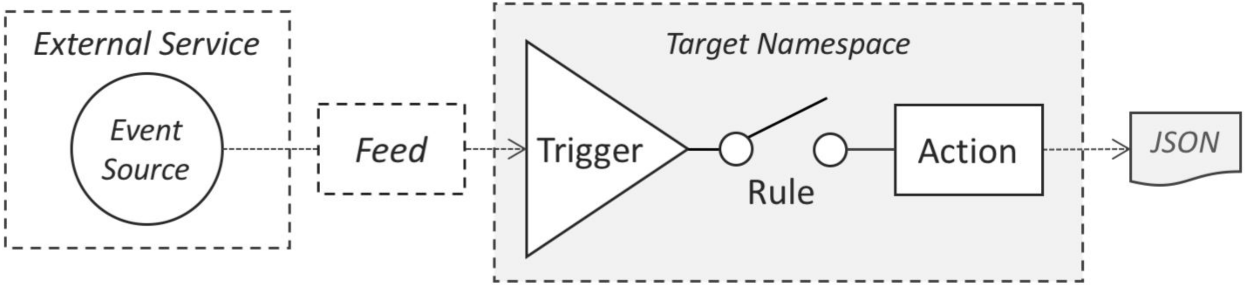
OpenWhisk

OpenFaas

OpenWhisk

The programming model of OpenWhisk make use of: [OpenWhisk]

- Triggers: event channels
- Rules: connect a Trigger to an Action
- Actions: stateless functions (logic)



OpenWhisk

OpenWhisk can be used either locally or remotely

The tool **wsk** allows a developer to create and interact with OpenWhisk entities

- Define a new function
- Invoke the function (sync or async)

OpenWhisk

wsk:

```
function main() {
 return {payload: 'Hello world'};
}
```

```
wsk action create helloJS hello.js
wsk action invoke helloJS --blocking
{
 "result": {
 "payload": "Hello world"
},
 "status": "success",
 "success": true }
```

OpenFaas

With the CLI tool **faas-cli** is possible to

- Create functions from templates
  - Many languages supported
- Create a Docker image of the function
- Deploy the function on a Kubernetes cluster
- Invoke the function
- Connect functions to HTTP triggers, or also other event sources like pub-sub brokers (e.g. Apache Kafka)

Real world case - Netflix

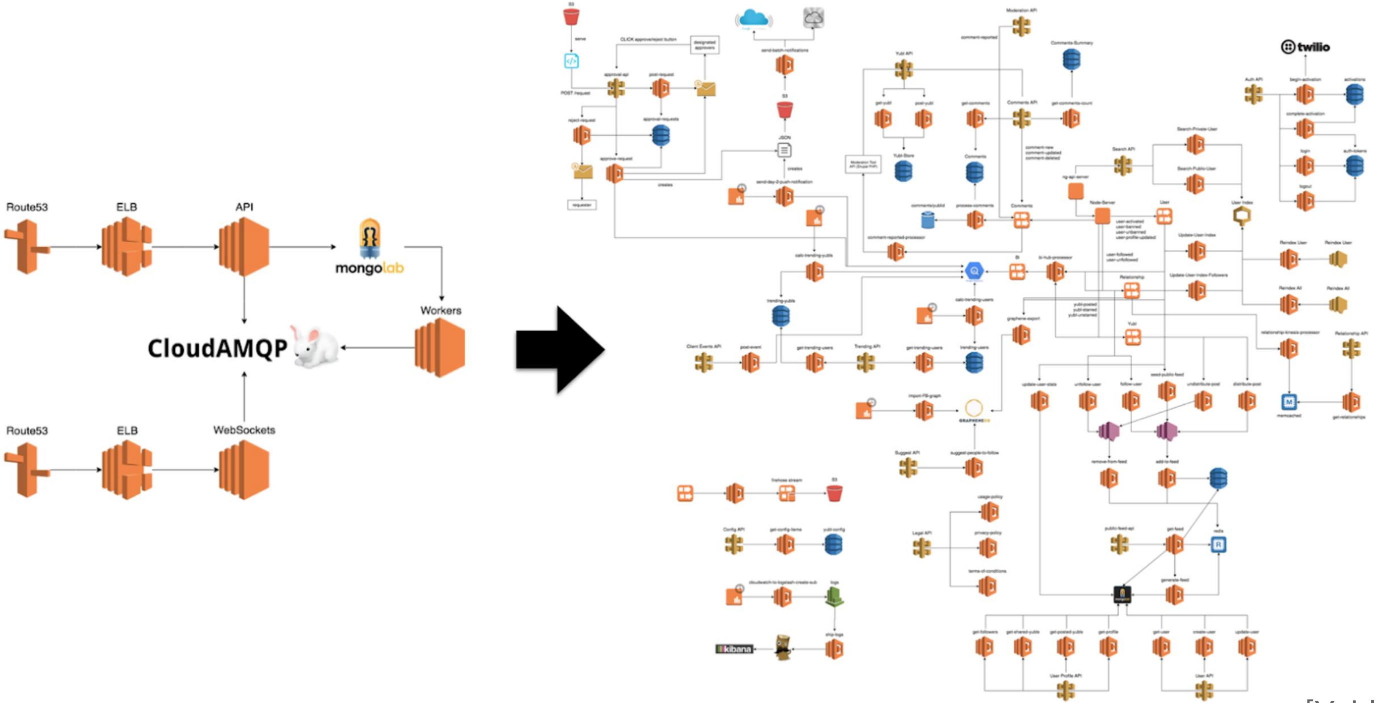
Netflix utilizes serverless computation to automatically manage

1. Backup for disaster recovery
2. Encoding media files
3. Security notifications
4. Metrics and dashboard

Netflix delivers about 7B hours of video di millions of users

[AWSNetflix]

Real world case - Yubl



[Yubl] [Adzic]

Conclusions and Challenges

Conclusions

Serverless computing, or FaaS, is a cloud service model

- Centered on stateless function-centric computation
- Focused on the application logic, and not the server management
- Pay-as-you-go
- Suitable for independent, single-task use cases
- Reduce the time-to-market, especially for new startups

Drawbacks - 1

Serverless computing comes with some drawback[Baldini] [Hellerstein]

- Stateless computation limits use cases like distributed computing
  - State can be propagated with slow storage means
  - This has also consequences on the pricing (I/O steps)
- Resources limitations and short-lived functions
  - 3GB of memory, 15 minutes in Amazon Lambda
- Too constraining for the developer, and probably lack of a particular execution environments (latest interpreter, etc)

Drawbacks - 2

Serverless computing comes with some drawback[Baldini] [Hellerstein]

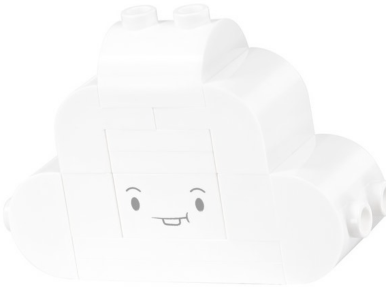
- Difficult for the provider to manage scaling and fault tolerance in an application agonistic manner
- Vendor lock-in
  - Such vendor tries to keep customer offering additional services

Future directions

Open research questions include

- Define its boundary with respect to the other service models
- Design efficient stateful serverless functions
- Design patterns to map applications into serverless functions
  - Need to identify resource requirements that fit in a serverless environment

Thank you!



References

[Baldini] **Serverless Computing: Current Trends and Open Problems**, Ioana Baldini et al

[Yan] **Building a Chatbot with Serverless Computing**, Mengting Yan et al

[McGrath] **Serverless Computing: Design, Implementation, and Performance**, Garrett McGrath et al

[Freet] **Cloud Forensics Challenges from a Service Model Standpoint: IaaS, PaaS and SaaS,** David Freet et al

[Begin] **Beginning Serverless Computing, Understanding Serverless Computing**, Chapter 1

[Geoffrey] **Report from workshop and panel on the Status of Serverless Computing and Function-as-a-Service (FaaS) in Industry and Research**, Geoffrey C. Fox et al

References

[Li] **Understanding Open Source Serverless Platforms: Design Considerations and Performance**, Junfeng Li et al

[Laatikainen] **Cloud Services Pricing Models**, Gabriella Laatikainen et al

[Hellerstein] **Serverless Computing: One Step Forward, Two Steps Back**, Joseph M. Hellerstein et al

[Adzic] **Serverless computing: economic and architectural impact**, Gojko Adzic et al

[Google] Google Cloud Functions Uses Cases (link)

[AmazonLambda] https://docs.aws.amazon.com/lambda/latest/dg/welcome.html

References

[OpenWhisk] https://openwhisk.apache.org/documentation.html

[Flower] https://martinfowler.com/articles/serverless.html#unpacking-faas

[NIST] National Institute of Standards and Technology

[AWSNetflix] https://aws.amazon.com/it/solutions/case-studies/netflix-and-aws-lambda/

[Yubl] https://hackernoon.com/yubls-road-to-serverless-part-1-overview-ca348370acde

[LiquidWeb] https://www.liquidweb.com/kb/serverless-vs-faas-a-beginners-guide/