CHALMERS
UNIVERSITY OF TECHNOLOGY

UNIVERSITY OF GOTHENBURG

ICET
LAB

# AWS Lambda and #serverless.

## What’s all the fuzz about?

*Joint work with*
Erik Wittern and Waldemar Hummer
Josef Spillner

**Philipp Leitner**

✉ philipp.leitner@chalmers.se

🐦 @xLeitix

ICET LAB

https://icet-lab.eu

@IcetLab

Universität Zürich UZH

# We are hiring a postdoc!!

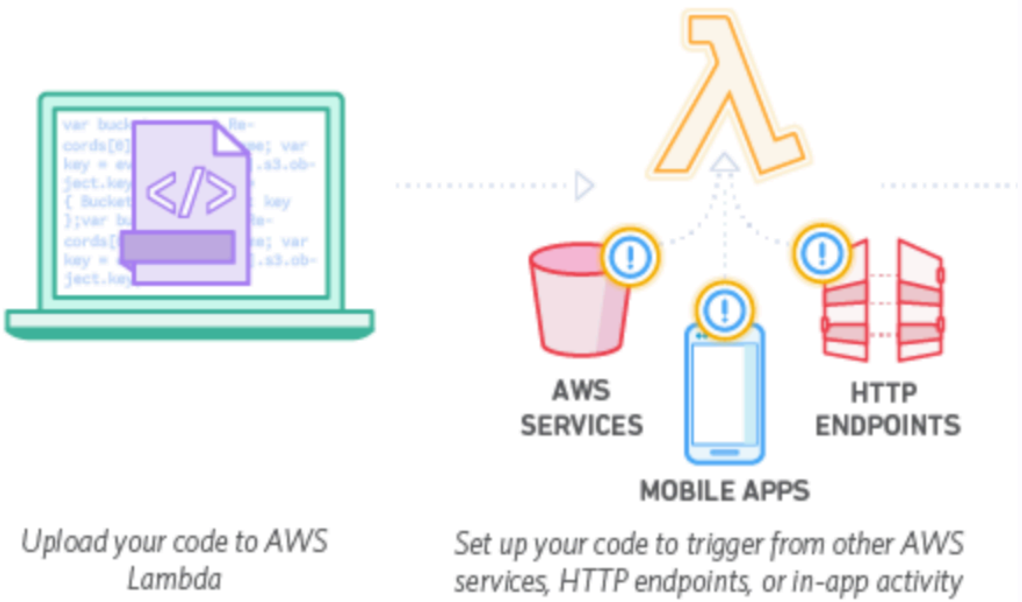
# Function-as-a-Service?

# Function-as-a-Service?

# Serverless?

Upload your code to AWS Lambda

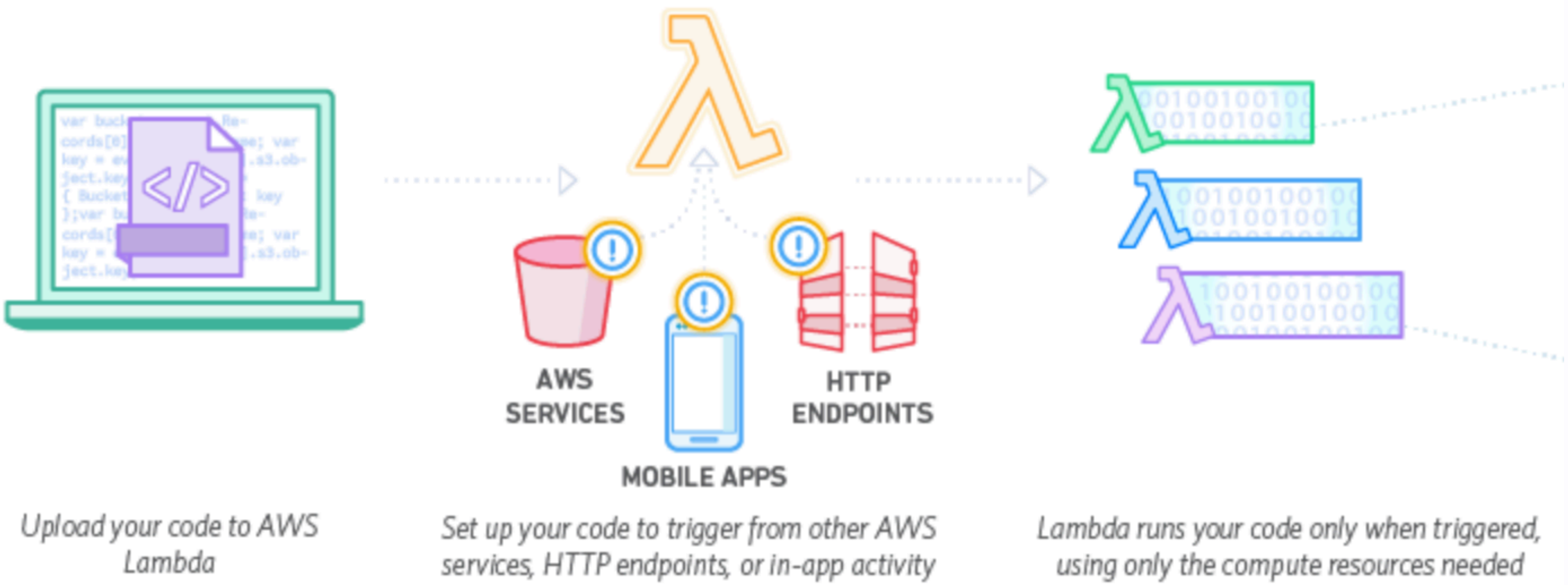
**Source:** https://aws.amazon.com/lambda/



Upload your code to AWS Lambda

Set up your code to trigger from other AWS services, HTTP endpoints, or in-app activity

**Source:** https://aws.amazon.com/lambda/

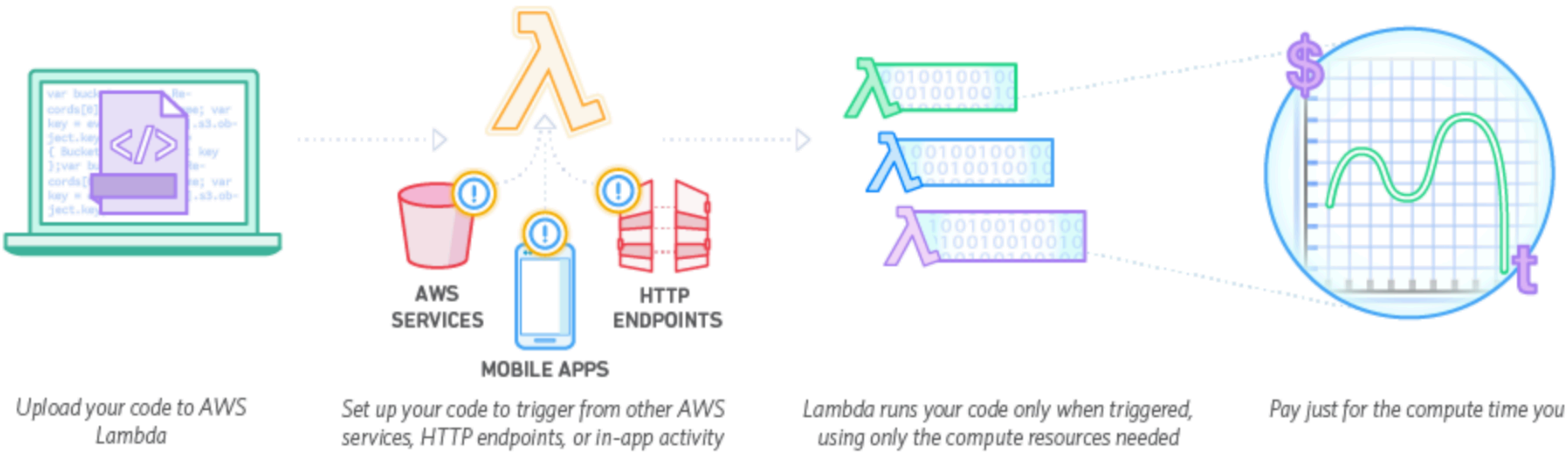


Upload your code to AWS Lambda

Set up your code to trigger from other AWS services, HTTP endpoints, or in-app activity

Lambda runs your code only when triggered, using only the compute resources needed

**Source:** https://aws.amazon.com/lambda/



*Upload your code to AWS Lambda*

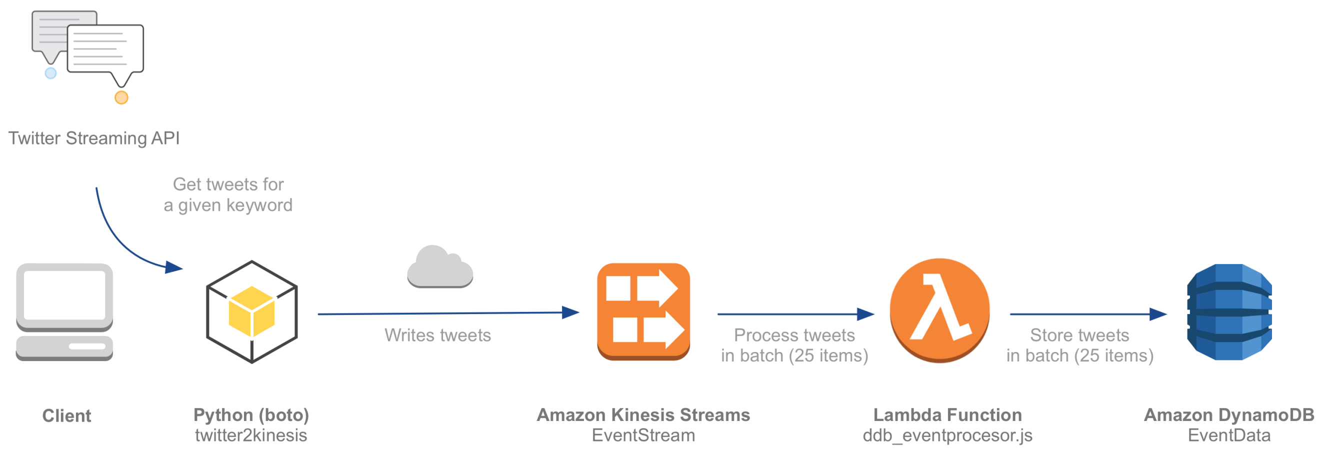
*Set up your code to trigger from other AWS services, HTTP endpoints, or in-app activity*

*Lambda runs your code only when triggered, using only the compute resources needed*

*Pay just for the compute time you use*

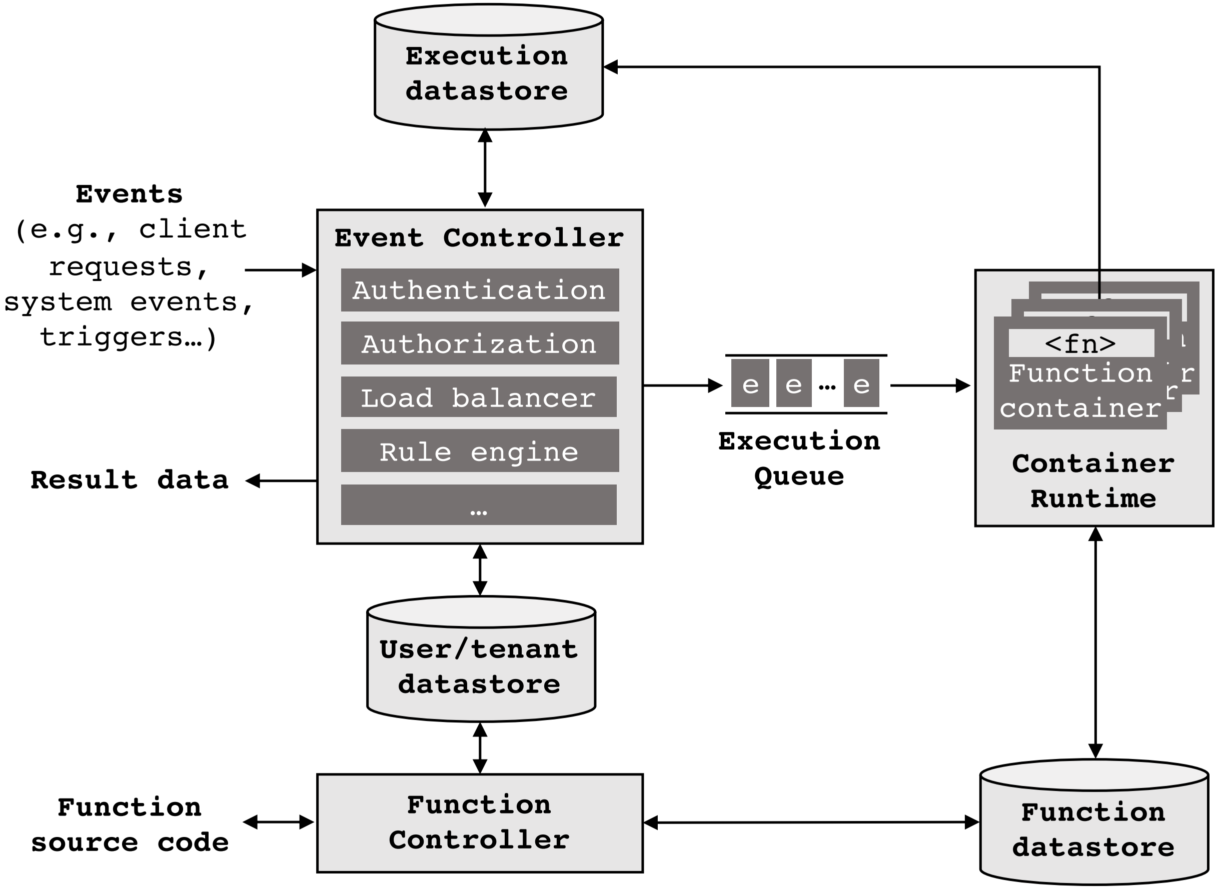
**Source:** https://aws.amazon.com/lambda/

# An Example: Real-Time Tweet Processing



**Source:** https://github.com/aws-samples/lambda-refarch-streamprocessing

# Internally



**Source:** loosely based on IBM's OpenWhisk architecture

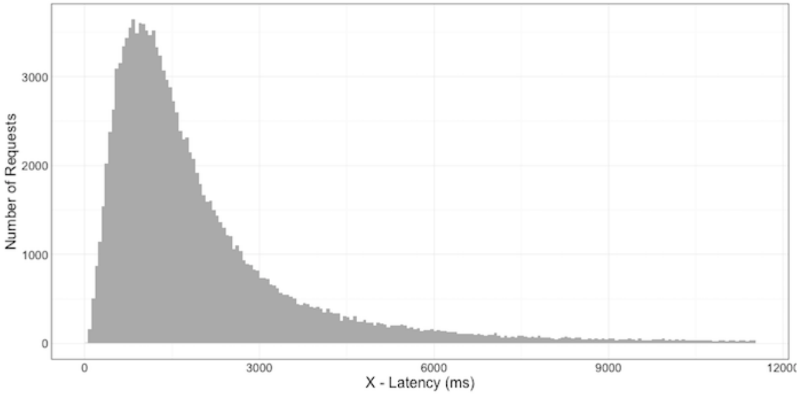
# No state

# No state

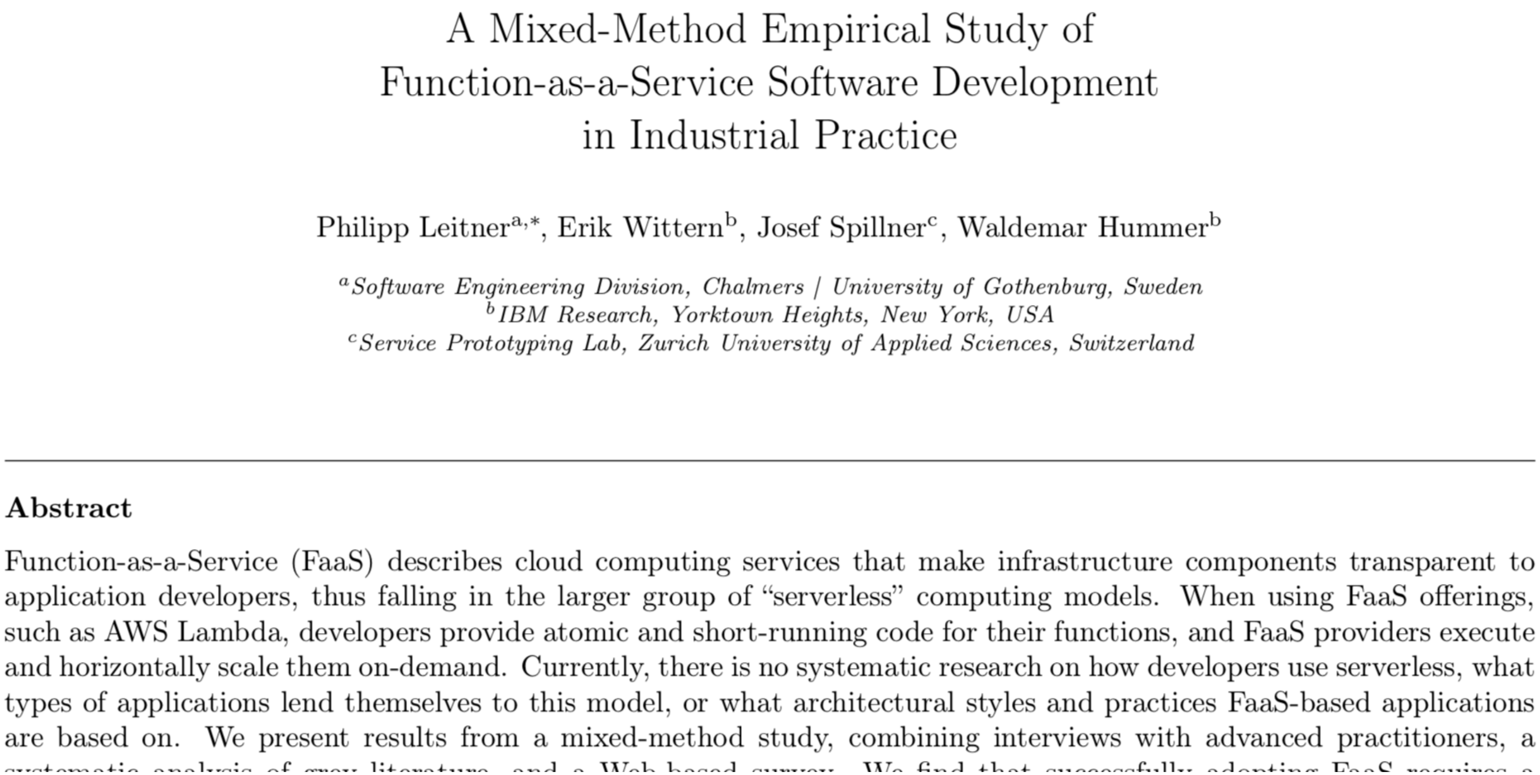
Hard cap on max. execution time ✗

# No state

## Hard cap on max. execution time



**Extreme tail latency**



# A Mixed-Method Empirical Study of Function-as-a-Service Software Development in Industrial Practice

Philipp Leitner[a,*], Erik Wittern[b], Josef Spillner[c], Waldemar Hummer[b]

[a] *Software Engineering Division, Chalmers | University of Gothenburg, Sweden*
[b] *IBM Research, Yorktown Heights, New York, USA*
[c] *Service Prototyping Lab, Zurich University of Applied Sciences, Switzerland*

## Abstract

Function-as-a-Service (FaaS) describes cloud computing services that make infrastructure components transparent to application developers, thus falling in the larger group of "serverless" computing models. When using FaaS offerings, such as AWS Lambda, developers provide atomic and short-running code for their functions, and FaaS providers execute and horizontally scale them on-demand. Currently, there is no systematic research on how developers use serverless, what types of applications lend themselves to this model, or what architectural styles and practices FaaS-based applications are based on. We present results from a mixed-method study, combining interviews with advanced practitioners, a systematic analysis of grey literature, and a Web-based survey. We find that successfully adopting FaaS requires a

# Methodology

**Interviews** (n = 12)

Analysis of **Grey Literature** (n = 50)

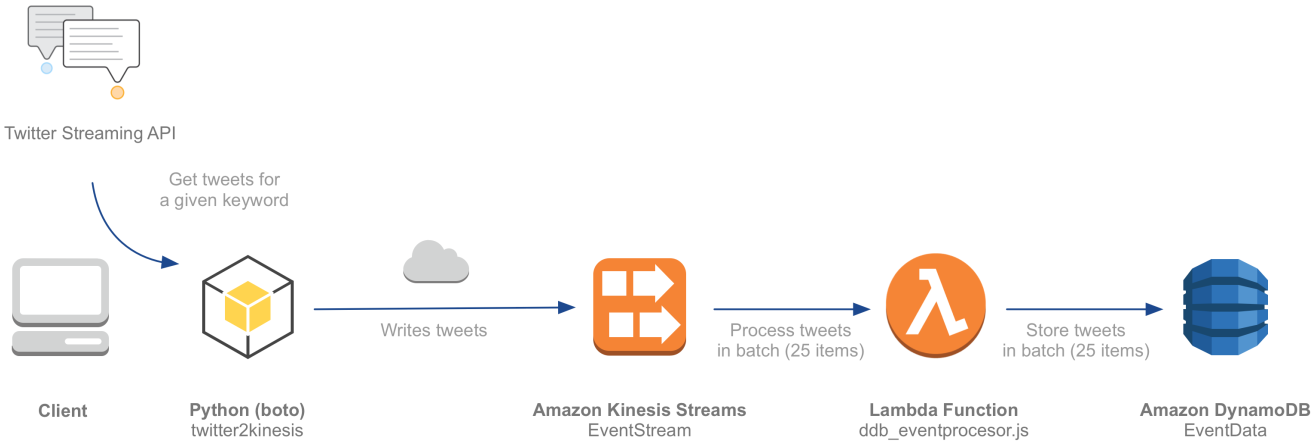
**Web Survey** (n = 182)

# Main Findings

## Compositional Application Model

*“Microservices on steroids”*

# Recall: Real-Time Tweet Processing



amazon web services

**Source:** https://github.com/aws-samples/lambda-refarch-streamprocessing

# Compositional Application Model

> "*I think the term 'application' is oftentimes not really that applicable anymore (...) it's really hard to say, like, what is the application anymore [and what is part of the cloud or infrastructure.].*" -I6

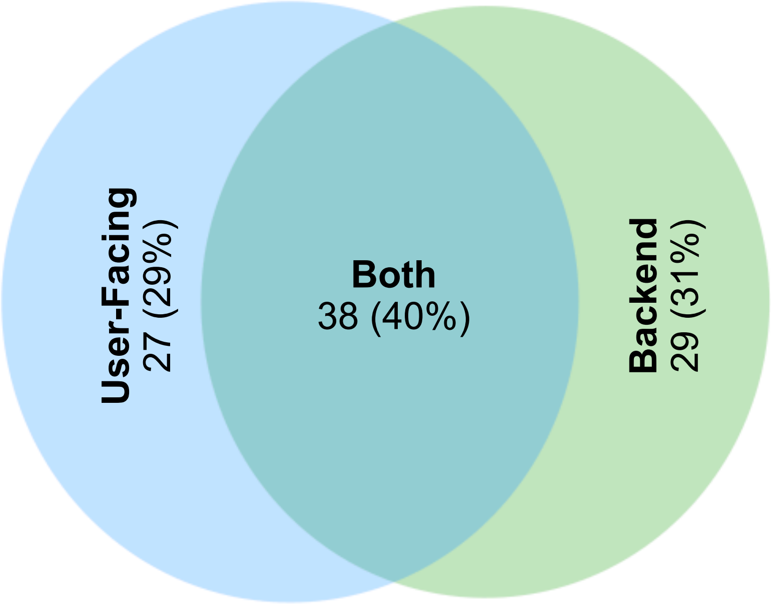
> "*AWS API Gateway, S3, Kinesis, SNS, DynamoDB, Step-Functions, or their Azure and GCP siblings — are at play with any serverless solution*" -A9

# Main Findings

Compositional Application Model

Use Cases

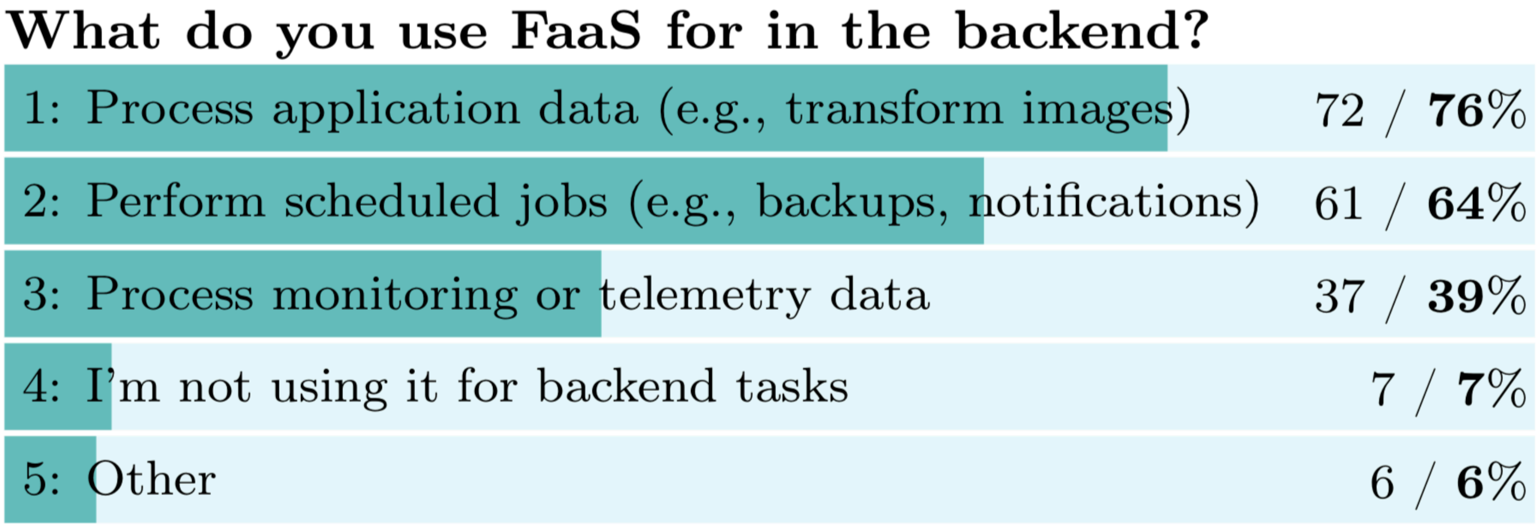
# Main Findings



**But:**

many challenges in user-facing apps

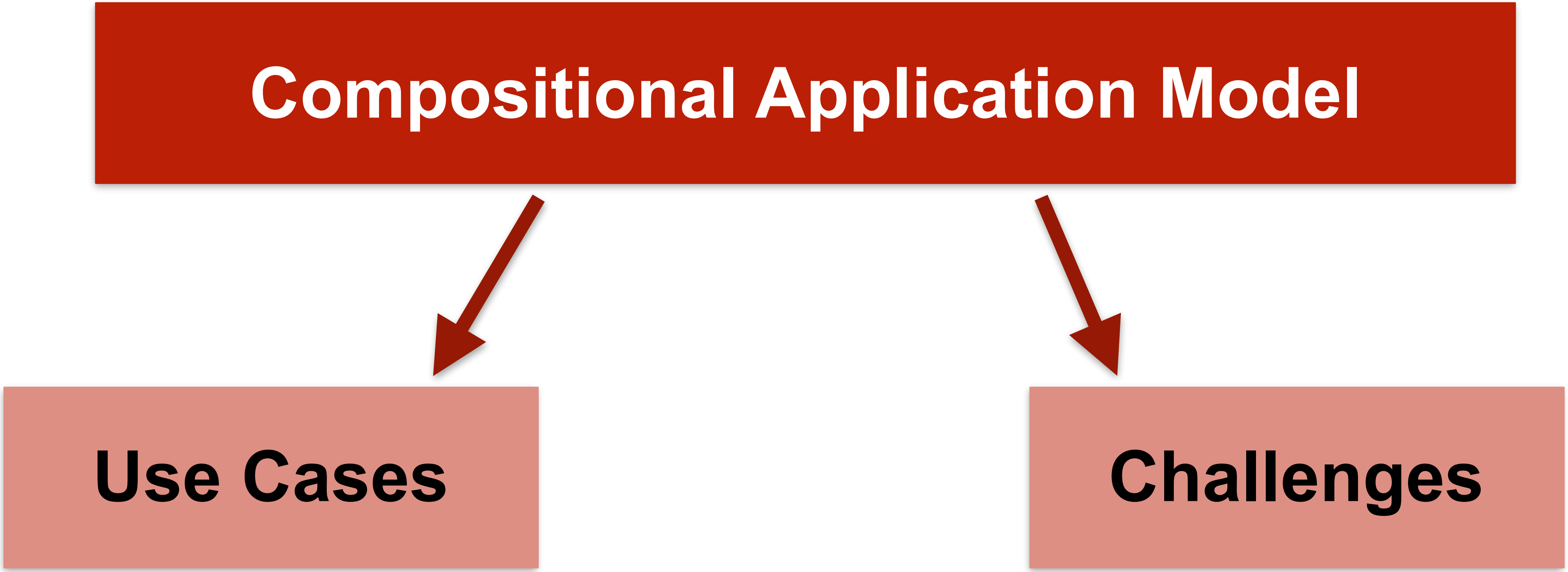
# Use Cases - Backend



**What do you use FaaS for in the backend?**

| | | | |
|---|---|---|---|
| 1: | Process application data (e.g., transform images) | 72 | **76%** |
| 2: | Perform scheduled jobs (e.g., backups, notifications) | 61 | **64%** |
| 3: | Process monitoring or telemetry data | 37 | **39%** |
| 4: | I'm not using it for backend tasks | 7 | **7%** |
| 5: | Other | 6 | **6%** |

# Main Findings

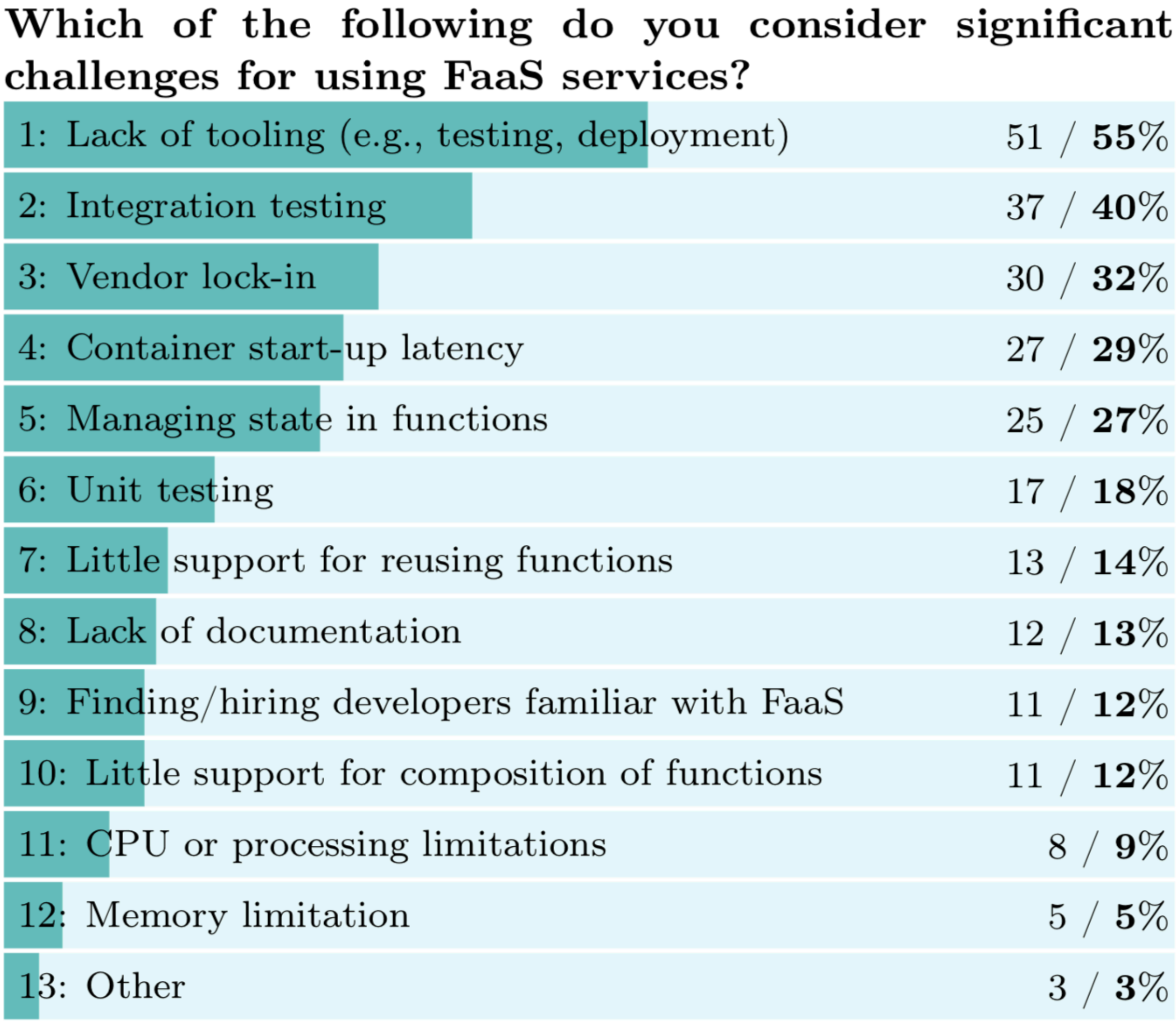


Compositional Application Model

Use Cases

Challenges

# Challenges



**Which of the following do you consider significant challenges for using FaaS services?**

| | |
|---|---|
| 1: Lack of tooling (e.g., testing, deployment) | 51 / **55%** |
| 2: Integration testing | 37 / **40%** |
| 3: Vendor lock-in | 30 / **32%** |
| 4: Container start-up latency | 27 / **29%** |
| 5: Managing state in functions | 25 / **27%** |
| 6: Unit testing | 17 / **18%** |
| 7: Little support for reusing functions | 13 / **14%** |
| 8: Lack of documentation | 12 / **13%** |
| 9: Finding/hiring developers familiar with FaaS | 11 / **12%** |
| 10: Little support for composition of functions | 11 / **12%** |
| 11: CPU or processing limitations | 8 / **9%** |
| 12: Memory limitation | 5 / **5%** |
| 13: Other | 3 / **3%** |

# The Future (as we see it)

***"Faster horses"***

vs

**Different Services**

(end user view)

(provider view)

# The Future (as we see it)

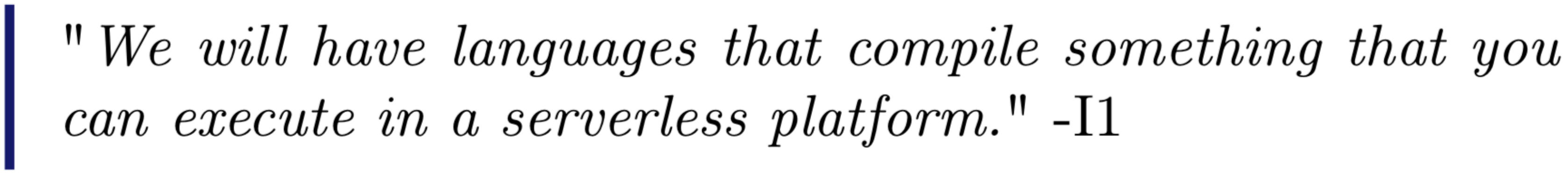
**Different Services**

*“Function-as-a-Service as the assembly language of the cloud”*

# The Future (as we see it)

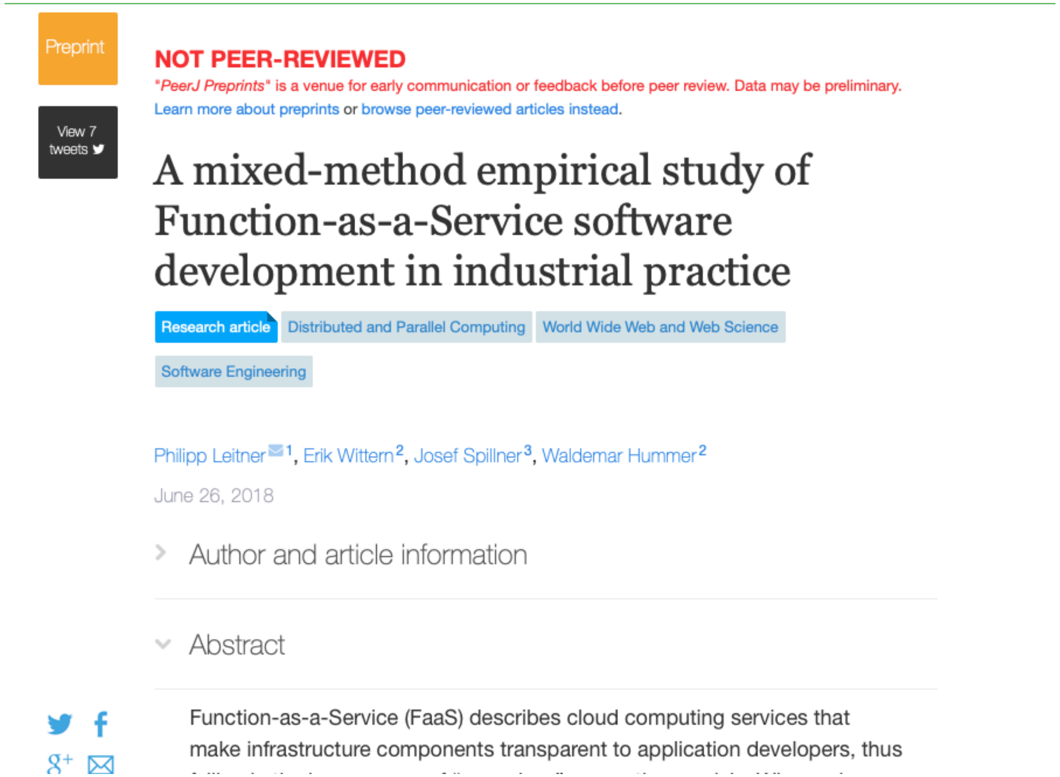
**Different Services**

*“Function-as-a-Service as the assembly language of the cloud”*



> *" We will have languages that compile something that you can execute in a serverless platform."* -I1

# Learn More



NOT PEER-REVIEWED

"*PeerJ Preprints*" is a venue for early communication or feedback before peer review. Data may be preliminary.

Learn more about preprints or browse peer-reviewed articles instead.

## A mixed-method empirical study of Function-as-a-Service software development in industrial practice

Research article
Distributed and Parallel Computing
World Wide Web and Web Science
Software Engineering

Philipp Leitner[1], Erik Wittern[2], Josef Spillner[3], Waldemar Hummer[2]

June 26, 2018

Abstract

Function-as-a-Service (FaaS) describes cloud computing services that make infrastructure components transparent to application developers, thus

**https://peerj.com/preprints/27005/**