Service Engineering Research Area
ICCLAB SPLAB

Zurich University of Applied Sciences
zh aw School of Engineering
InIT Institute of Applied Information Technology

Serverless Applications:

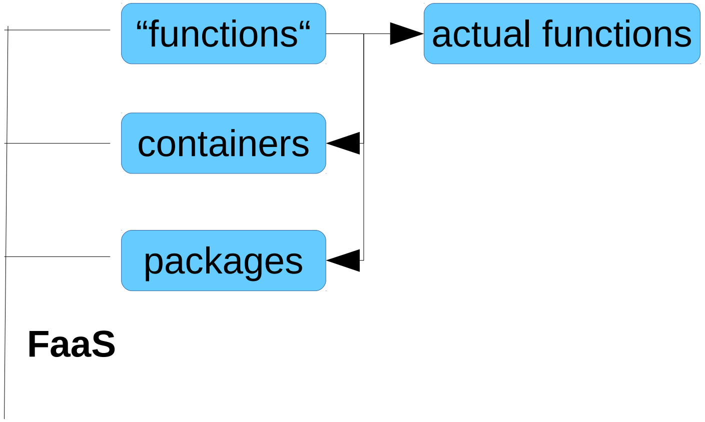
Tools, Languages, Providers and (Research) Challenges

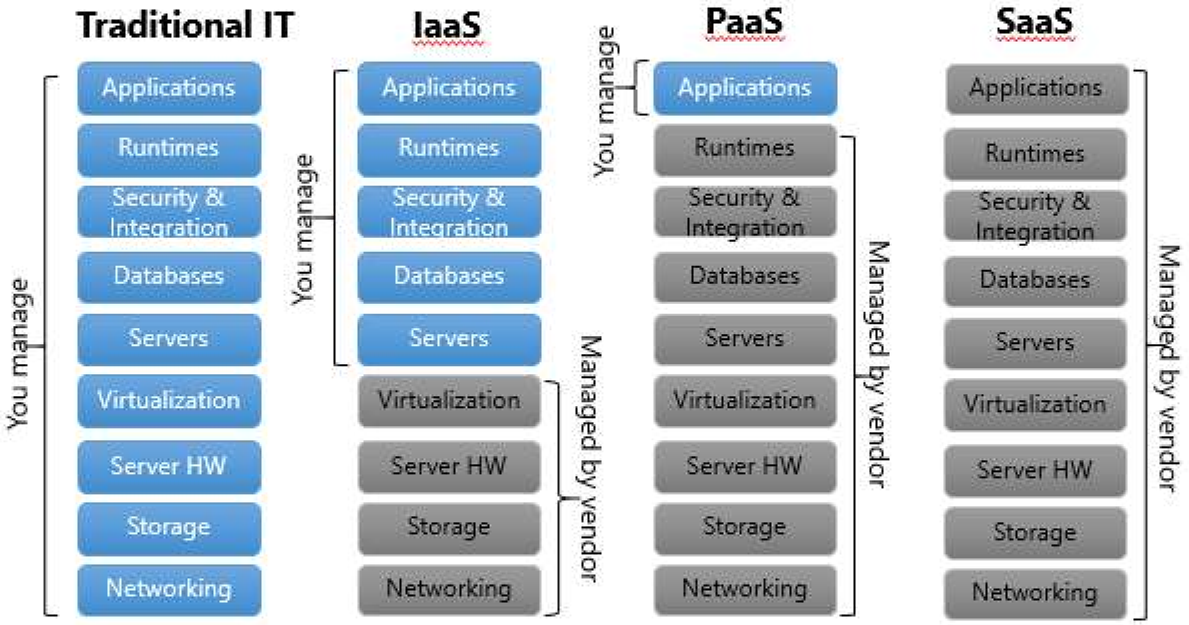
Josef Spillner <josef.spillner@zhaw.ch>
Service Prototyping Lab (blog.zhaw.ch/icclab)
Zurich University of Applied Sciences

Jun 22, 2017 | Serverless Meetup

What is FaaS?

- running functions in the cloud (“hosted functions“)
- real “pay per use“ (per invocation, per load × time unit, e.g. MB/100ms)
- seemingly “serverless“

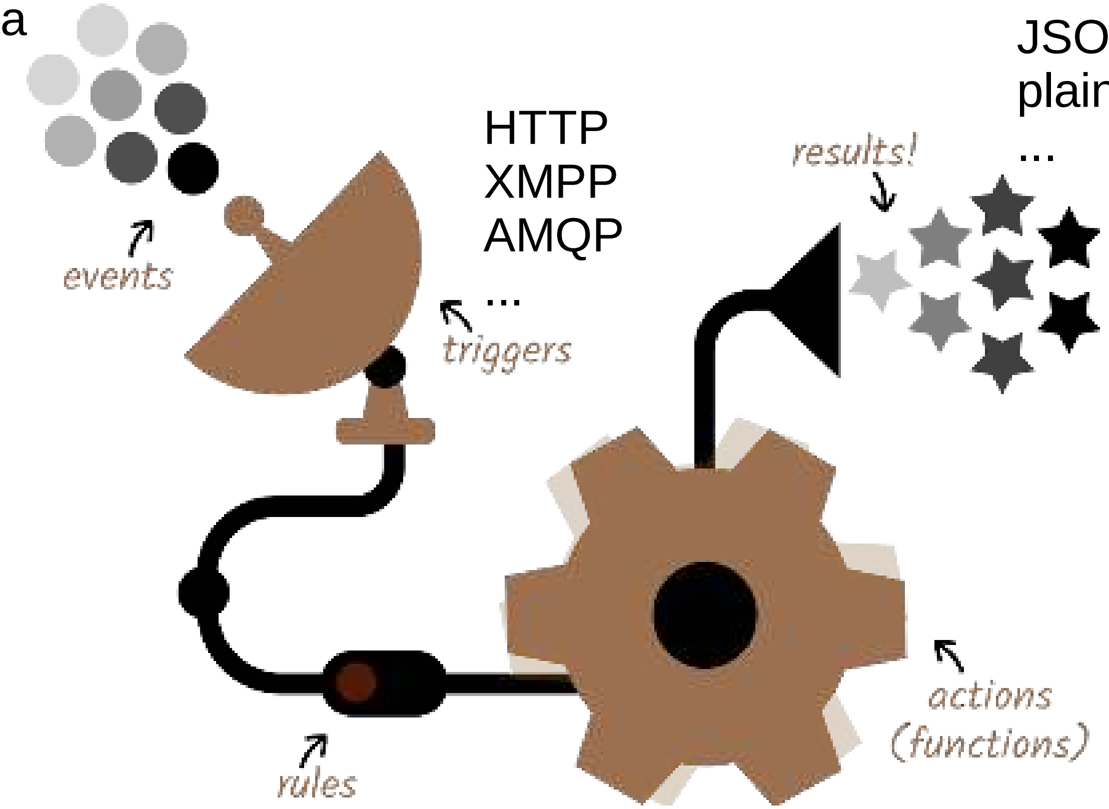




[mazikglobal.com]

Examples of FaaS: Process

monitoring event
sensor data
log entry
git push
...



HTTP
XMPP
AMQP
...

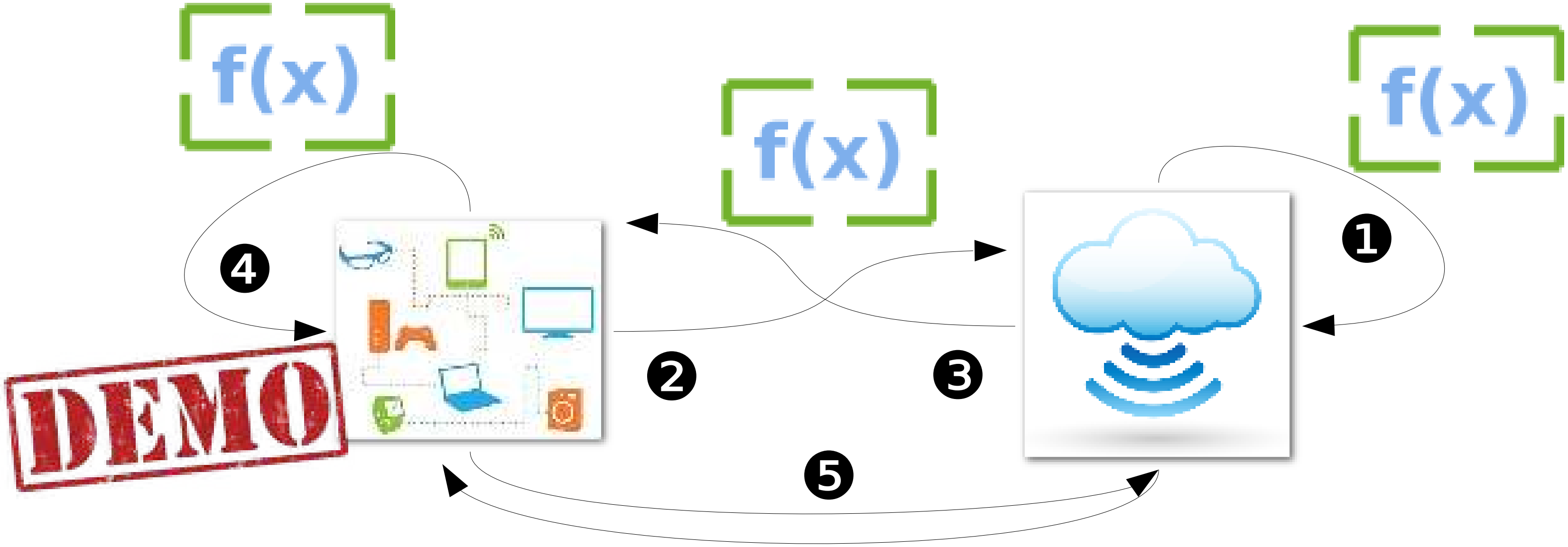
JSON
plain text
...

Your Python
functions!

max 1 per hour
triggers/actions
default params
...

[openwhisk.org]

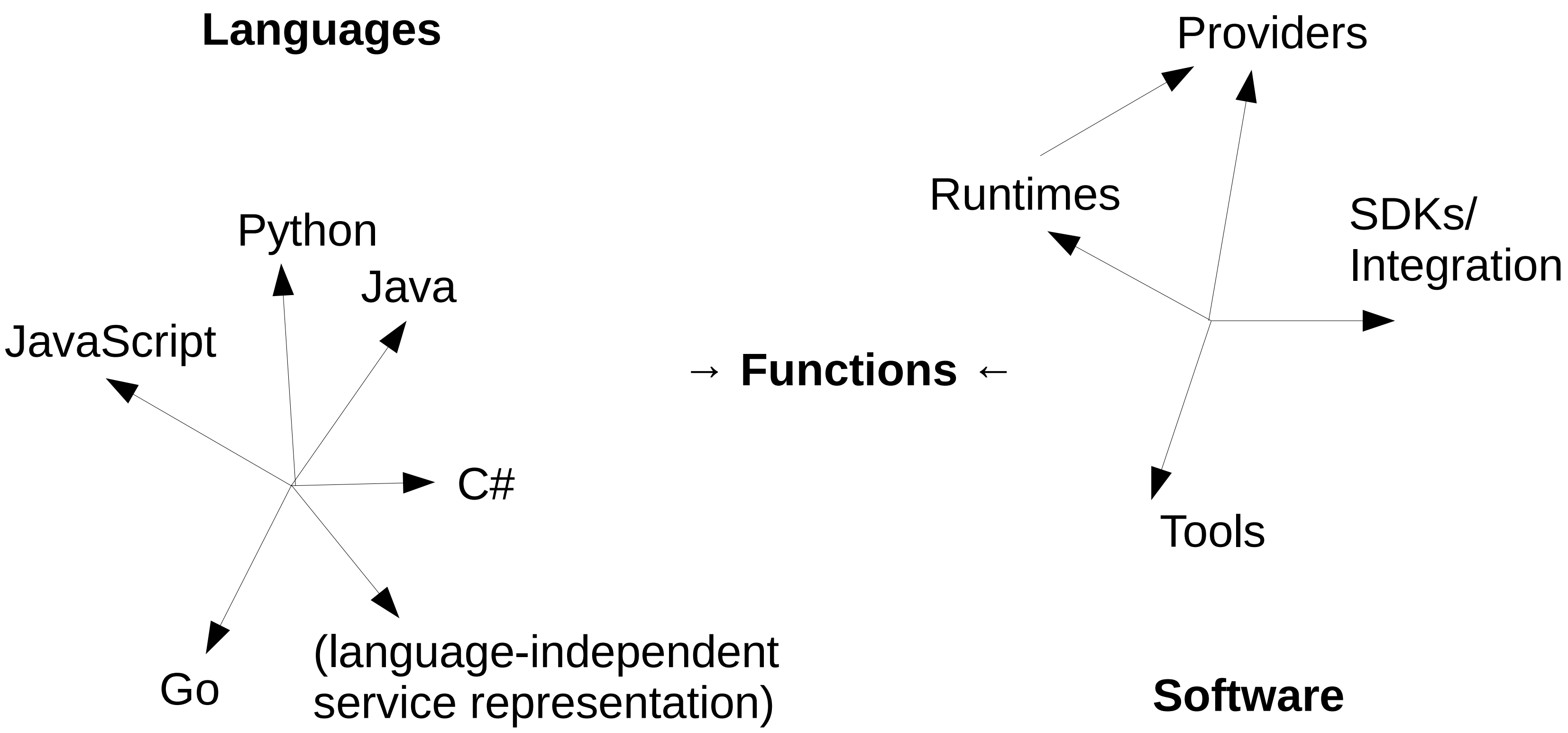
Examples of FaaS: Environment



1) input from cloud, output to cloud
   e.g. incident management system
2) input from devices, ...
   e.g. robot vision processing
3) e.g. cloud-controlled CPS (door locks)
4) e.g. microphone to USB light
5) e.g. 1000s of microphones to USB lights

Service Engineering Research Area
ICCLAB
SPLAB

The FaaS Space



Languages

Python

Java

JavaScript

C#

Go

(language-independent service representation)

→ **Functions** ←

Providers

Runtimes

SDKs/ Integration

Tools

Software

The FaaS Space

AWS Lambda

Google Cloud Platform
Functions

OpenWhisk

Azure

webtask

hook.io

PyWren
[Lambda]

Chalice
[Lambda]

Apex
[Lambda]

Zappa
[Lambda]

Dawson
[Lambda]

Serverless Framework
[Lambda, OW, GCF, AF]

Service Engineering
Research Area
ICCLAB
SPLAB

FaaS (Docker)

Kubeless

Funktion

Fission

Picasso

Effe

Docker-LambCI

OpenLambda

Lever OS

Whisk-Mocha
[OpenWhisk]

X-Ray
[Lambda]

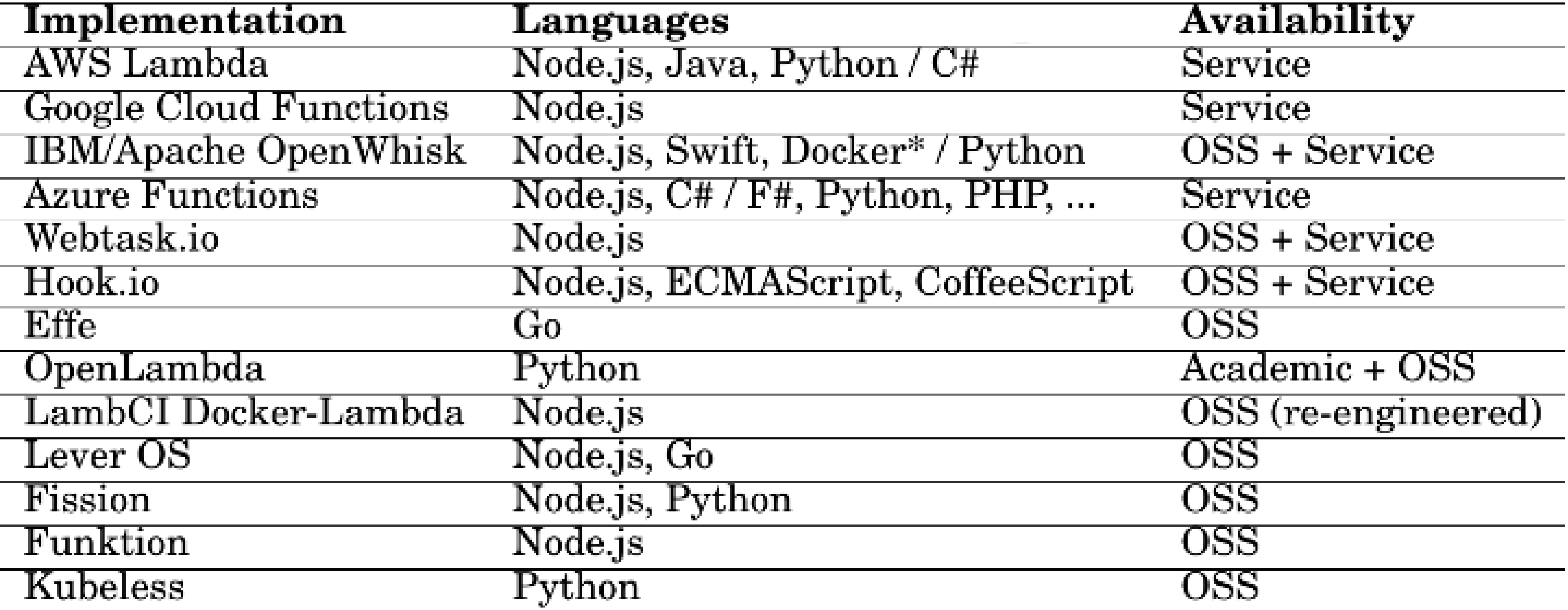
Step Functions
[Lambda]

MR Refarg
[Lambda]

LambDash
[Lambda]

The FaaS Space: Runtimes

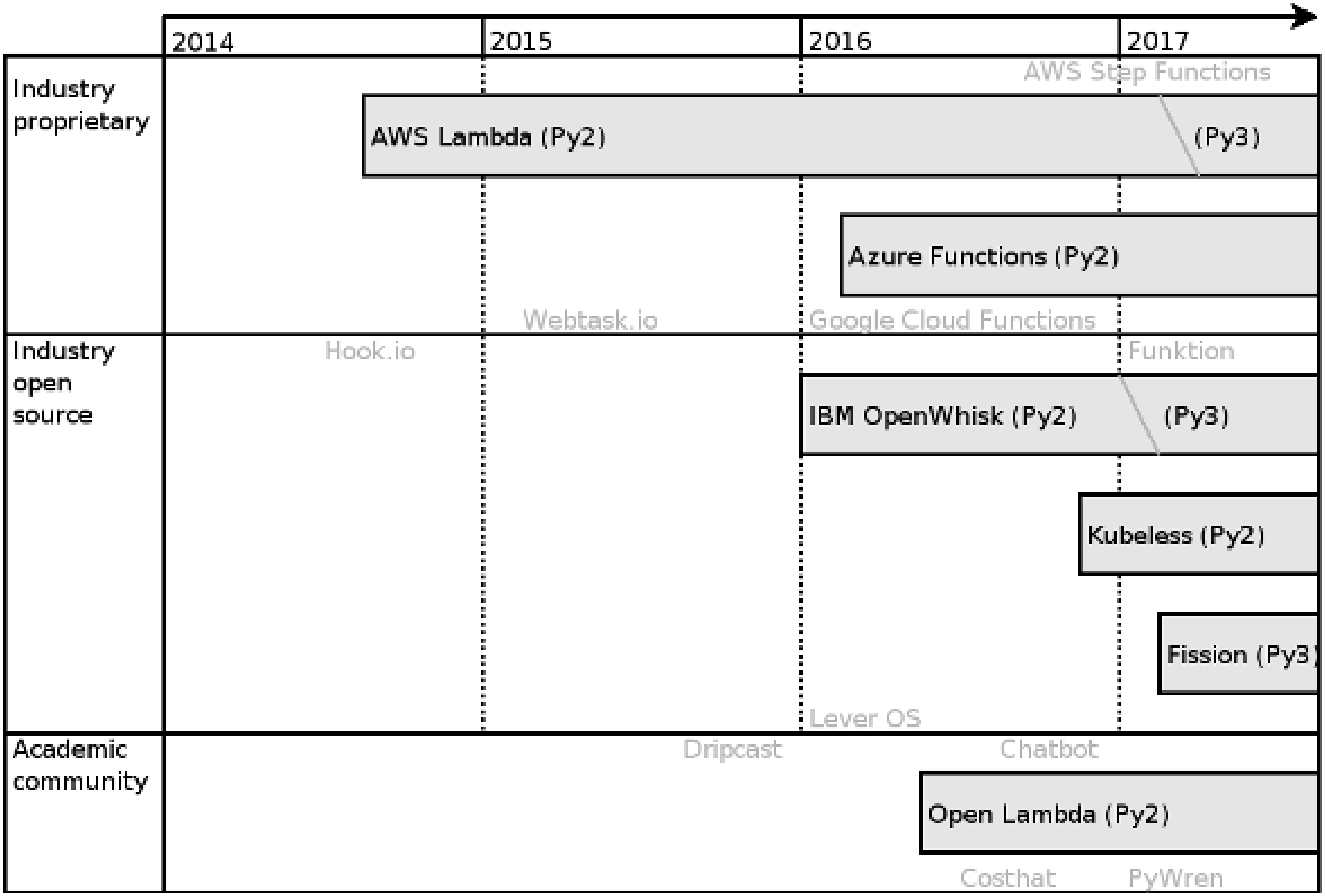
Function-as-a-Service offerings in greater detail...



Implementation	Languages	Availability
AWS Lambda	Node.js, Java, Python / C#	Service
Google Cloud Functions	Node.js	Service
IBM/Apache OpenWhisk	Node.js, Swift, Docker* / Python	OSS + Service
Azure Functions	Node.js, C# / F#, Python, PHP, ...	Service
Webtask.io	Node.js	OSS + Service
Hook.io	Node.js, ECMAScript, CoffeeScript	OSS + Service
Effe	Go	OSS
OpenLambda	Python	Academic + OSS
LambCI Docker-Lambda	Node.js	OSS (re-engineered)
Lever OS	Node.js, Go	OSS
Fission	Node.js, Python	OSS
Funktion	Node.js	OSS
Kubeless	Python	OSS

Trend: Sooner or later → gaps will be filled

The FaaS Space: Python runtimes



FaaS Synopsis in Python

AWS Lambda:

```
def lambda_handler(event, context):
        ‘‘‘
        event: dict
        context: meta information object
        returns: dict, string, number, ...
        ‘‘‘
        # ...
        return “result“
```

OpenWhisk:

```
def handler(input):
        ‘‘‘
        input: dict
        returns: dict
        ‘‘‘
        # ...
        return {}
```

Fission:

```
def main():
        ‘‘‘
        input: via flask.request.get_data()
        returns: str
        ‘‘‘
        # ...
        return “result“
```

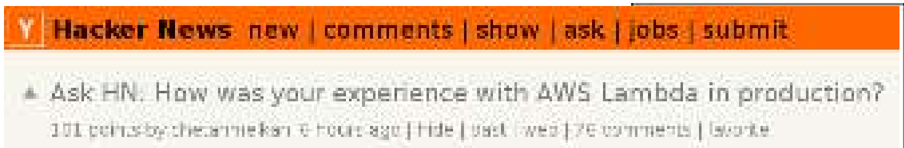
Azure Functions:

```
def main():
        from AzureHTTPHelper import\
        HTTPHelper
        input = HTTPHelper().post
        # ...
        open(os.environ[“res“], “w“).write(\
        json.dumps({“body“: “...“}))
main()
```

Further differences:

- function scoping (e.g. with/without export in JavaScript)
- function naming (mangling on client or service side)

FaaS Challenges (Engineers‘ View)



▲ Ask HN: How was your experience with AWS Lambda in production?

101 points by chetanmelkani 6 hours ago | hide | past | web | 76 comments | favorite

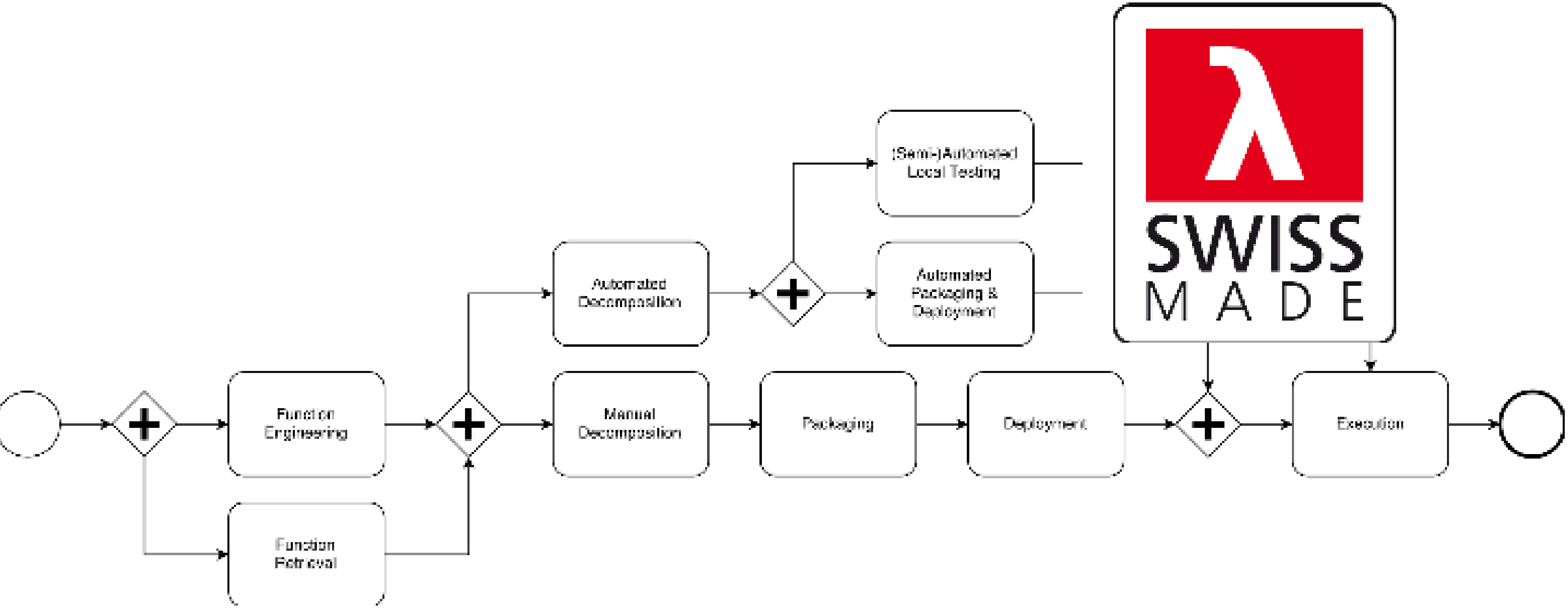
User	Development complicated	Monitoring insufficient	API/language substandard	Deployment slow	Expensive	Integration poor	all positive
callumlocke	X	X	X	X			
weeefun		X		X			
scrollaway	X	X	X		X		
CSDude		X		X			
davidvanleeuwen						X	
lanestp	X					X	
dblooman	X	X					
cameronmaske	X	X			X		
petarb	X	X		X		X	
beefsack							X
kehers			X		X	X	
tracker	X						
alexcasalboni							X
cntlzw		X				X	
xer							X
marcfowler						X	
Jdam							X
rmccue						X	
eknkc		X					
akhatri_aus		X		X			

use frameworks

do some crazy stuff to get metrics

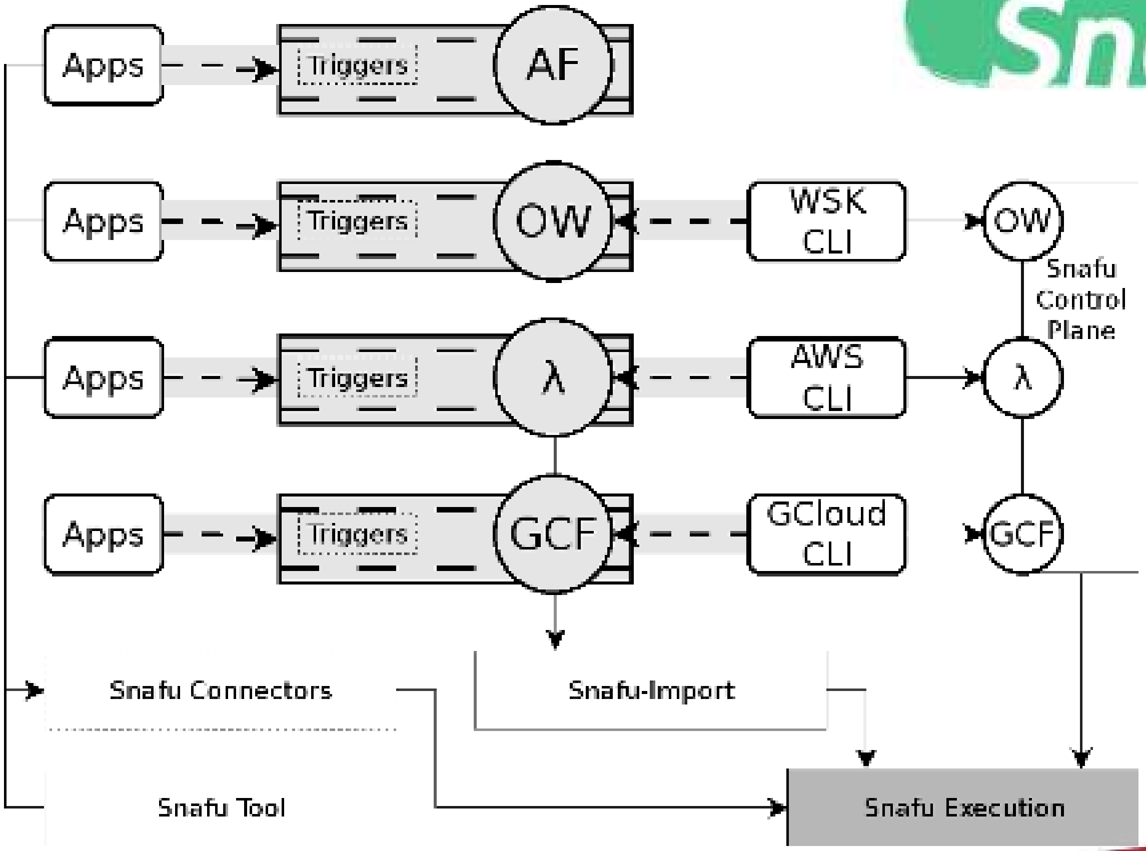
API gateway is terrible

FaaS Programmer Perspective



Snafu

The Swiss Army Knife of Serverless Computing



DEMO

FaaSification

→ Process of automated decomposition of software application into a set of deployed and readily composed function-level services.

FaaSification := code analysis + transformation + deployment + on-demand activation

Integration Categories:

- generic (code/function unit generation)
- single-provider integration
- multi-provider integration

Depth Categories:

- shallow (file to function)
- medium (function to lines)
- deep (line to many lines)

Decomposition Categories:

- static code analysis
- dynamic code analysis

“Lambdafication“:

- FaaSification to Lambda

Languages:
Java & Python

Termite

Podilizer

DEMO

The FaaS Space

AWS Lambda

Google Cloud Platform Functions

OpenWhisk

Azure

webtask

hook.io

PyWren [Lambda]

Chalice [Lambda]

Apex [Lambda]

Zappa [Lambda]

Dawson [Lambda]

Service Engineering Research Area ICCLAB SPLAB

Serverless Framework [Lambda, OW, GCF, AF]

FaaS (Docker)

Kubeless

Funktion

Snafu

Fission

Picasso

Effe

Docker-LambCI

OpenLambda

Lever OS

Termite

Whisk-Mocha [OpenWhisk]

Podilizer

X-Ray [Lambda]

Step Functions [Lambda]

Lambada

MR Refarg [Lambda]

LambDash [Lambda]

Further Reading and FaaS Fun

Lama, Lambackup:

- https://arxiv.org/abs/1701.05945

Podilizer:

- https://arxiv.org/abs/1702.05510

Snafu:

- https://arxiv.org/abs/1703.07562

Lambada

- https://arxiv.org/abs/1705.08169

On arXiv Analytics:

On GitHub:

[github.com/serviceprototypinglab]

PyParis

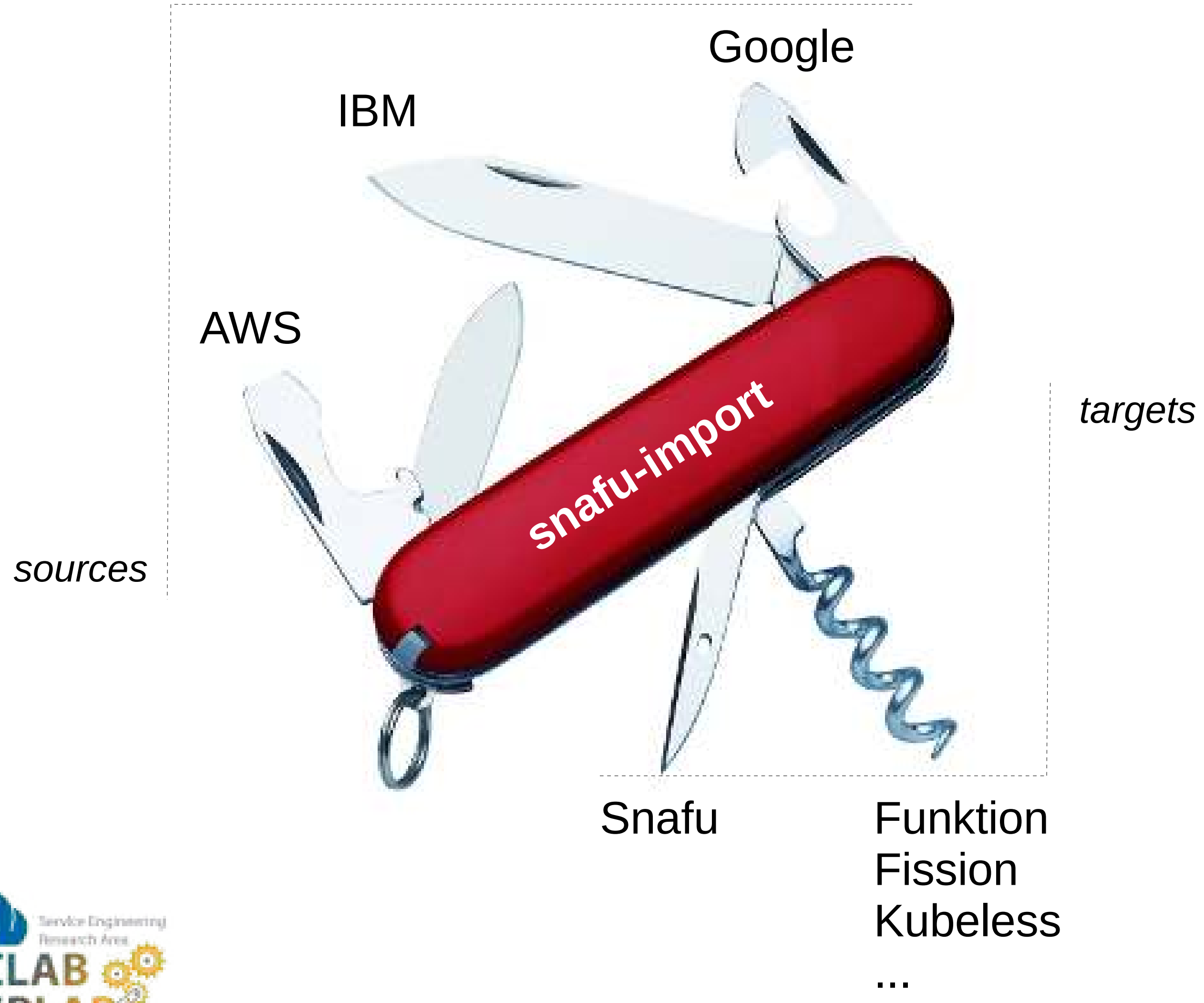
Tutorial slides + transcript @ blog.zhaw.ch/icclab

ICCLAB SPLAB Service Engineering Research Area

BACKUP...

Snafu

Integration into the wider FaaS ecosystem



Google

IBM

AWS

sources

targets

Snafu

Funktion
Fission
Kubeless
...

```
$ snafu-import \
--source <s> \
--target <t>
```

```
$ alias aws="aws \
--endpoint-url \
http://localhost:10000"

$ wsk property set \
--apihost \
localhost:10000

$ ./tools/patch-gcloud
```