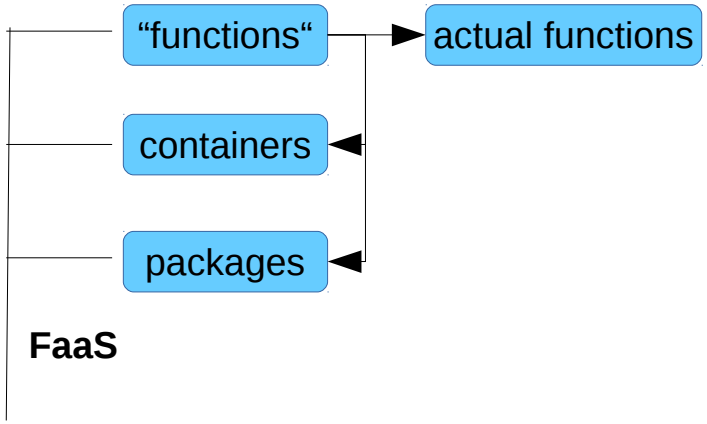
Serverless Computing: FaaSter, Better, Cheaper and *More Pythonic*

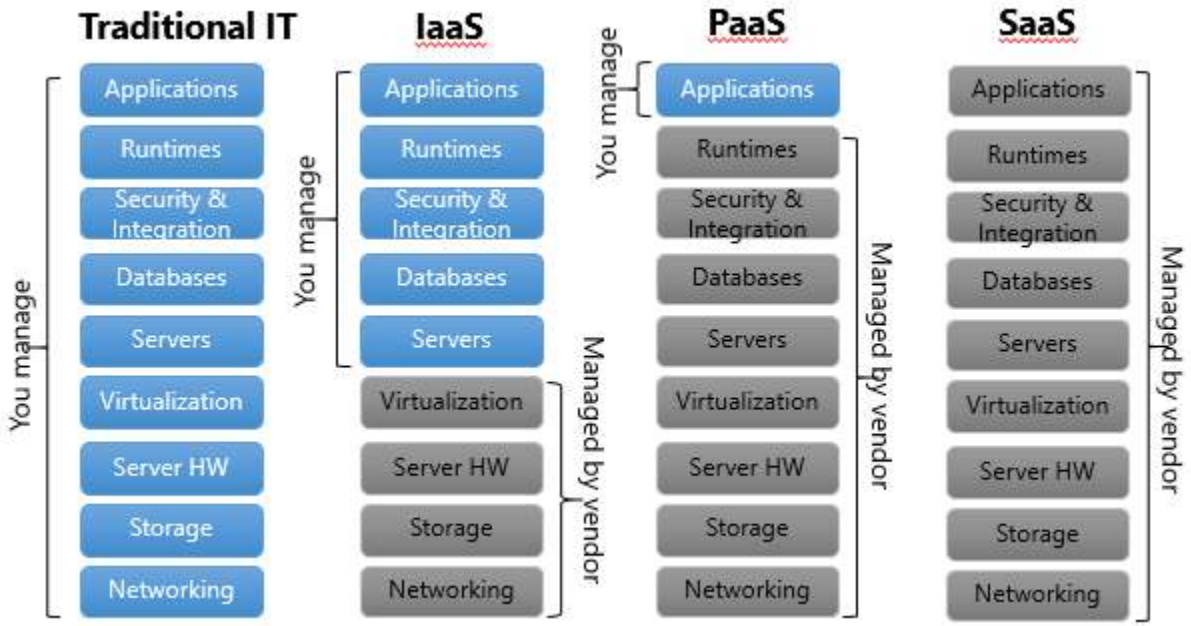
Josef Spillner <josef.spillner@zhaw.ch>
Service Prototyping Lab (blog.zhaw.ch/icclab)

February 16, 2018 | Swiss Python Summit

What‘s Function-as-a-Service (FaaS)?

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





[mazikglobal.com]

Service Engineering Research Area
ICCLAB SPLAB

Developer‘s Vision: Rapid Prototyping

Applied research mission

- technological immersion combined with scientific excellence
- supporting local (Swiss) development & devops companies

Applied to current serverless computing/FaaS environments:

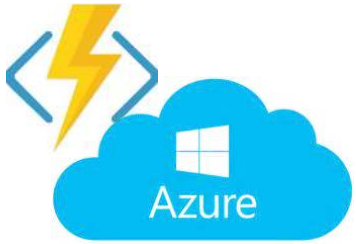
[9gag.com]

Service Engineering Research Area
ICCLAB SPLAB

The FaaS Space - in Python

AWS Lambda

IBM Cloud Functions



FaaS (Docker)

Kubeless

Funktion

Apache OpenWhisk

Fission

Picasso

Fn

Docker-LambCI

Elfe

Lever OS

OpenLambda

Snafu

IBM Composer
[OpenWhisk]

Whisk-Mocha
[OpenWhisk]

X-Ray
[Lambda]

Podilizer

Chalice
[Lambda]

Zappa
[Lambda]

PyWren
[Lambda]

Step Functions
[Lambda]

Fission Workflows
[Fission]

Termite

Dawson
[Lambda]

Apex
[Lambda]

LambDash
[Lambda]

Lambada

Serverless Framework
[Lambda, OW, GCF, AF]

MR Refarg
[Lambda]

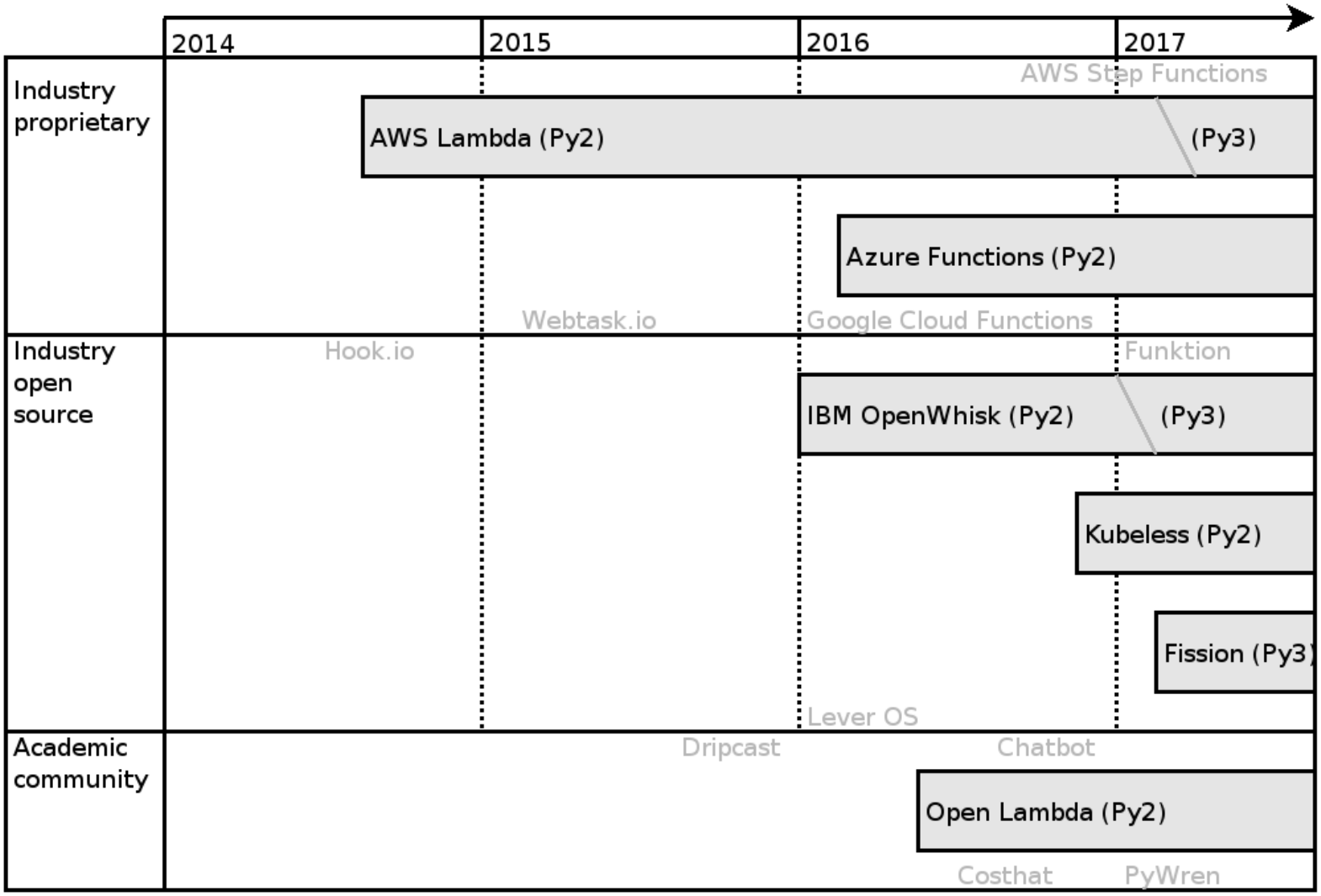
Service Engineering Research Area
ICCLAB SPLAB

Runtime Overview: Providers&Stacks

Implementation	Languages	Availability
AWS Lambda	Node.js, Java, Python / C#	Service
Google Cloud Functions	Node.js	Service
Apache OpenWhisk	Node.js, Swift, Docker* / Python	OSS
→ IBM Cloud Functions	-"-	Service
Azure Functions	Node.js, C# / F#, Python, PHP, ...	Service
OVH Functions	Node.js, Python, Perl, Go, Bash	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
IronFunctions	Node.js, Java, Python, Go, ...	OSS
→ Fn	-"-	OSS

Service Engineering Research Area
ICCLAB SPLAB

Runtime Overview: Python Evolution



Open Source Tools for FaaS

Good News:

- almost all tools in this sphere are open source

Bad News:

- almost none of the large provider runtimes
  - AWS Lambda, MS Azure Functions, Google Cloud Functions, OVH Functions, ...
  - changing now? OpenWhisk, Fn
- first-use barrier
- heterogeneous approaches, no standards → synopsis, deployment, ...
- didactic usefulness
- research/experimentation flexibility, e.g. high-performance execution without isolation or authentication

FaaS Synopsis: Python Examples

AWS Lambda:

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

OpenWhisk/IBM Functions:

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

OVH Functions:

```
def handler(input):
    “‘
    input: dict
    returns: str
    “‘
    # ...
    return ““
```

Fission:

```
def main():
    “‘
    input: 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 naming (mangling on client or service side)
- function granularity (number of entry points)

→ Deployment: provider tools, lambda-uploader, serverless fw, ...

Service Engineering Research Area
ICCLAB SPLAB

Overlay Approach: PyWren

Improved conveyance of “serverless“ paradigm

- no explicit deployment prior to execution
- rather, deploys while executing

```
def my_function(b):
  x = np.random.normal(0, b, 1024)
  A = np.random.normal(0, b, (1024, 1024))
  return np.dot(A, x)

pwex = pywren.default_executor()
res = pwex.map(my_function, np.linspace(0.1, 100, 1000))
```

How it works:

- tightly bound to AWS
- cloudpickle to AWS S3
- executes Lambda function which reads/writes from/to S3
- parallelisation through map functions

Service Engineering Research Area
ICCLAB SPLAB

Overlay Approach: Gee‘s Lambada

Deployment with dependencies

- requirements.txt file references Lambada framework

```
def my_function(b):
  x = np.random.normal(0, b, 1024)
  A = np.random.normal(0, b, (1024, 1024))
  return np.dot(A, x)

tune = Lambada(role='...', region='...', memory=128)

@tune.dancer
def my_function_lambda(e, c):
  my_function(e['stddev'])
```

How it works:

- again, tightly bound to AWS
- creation of ZIP packages for manual or automated deployment

Service Engineering Research Area
ICCLAB SPLAB

Excursus: “Lambada“ projects

Name	Purpose	First Commit	Python
Carson Gee‘s Lambada [https://github.com/Superpedestrian/lambada]	Building multiple Lambdas in one library	26.09.2016	**X**
Josef Spillner‘s Lambada [https://gitlab.com/josefspillner/lambada]	Extraction and transformation of Python functions to Lambda	18.04.2016	**X**
Çağatay Gürtürk‘s Lambada [https://github.com/lambadaframework]	JAX-RS API framework for Java Lambdas and API Gateway	31.03.2016	
Aldrin Leal‘s Lambada [https://github.com/ingenieux/lambada]	Maven integration for Java Lambdas	28.12.2015	
uSwitch‘s Lambada [https://github.com/uswitch/lambada]	Developing Java Lambdas in Closure	16.06.2015	

Lambada: FaaS Code Transformer

Rapid prototyping through semi-automated transformation
Support for annotations
PyPI: `pip install lambadatransformer`

```
@cloudfunction(memory=128, duration=5)
def my_function(b):
  x = np.random.normal(0, b, 1024)
  A = np.random.normal(0, b, (1024, 1024))
  return np.dot(A, x)

$ ./lambada --annotations my_function.py

>>> from lambadalib import lambada
>>> lambada.move(globals(), endpoint=..., local=True)
```

Source level: `ast, codegen`
Object level: `inspect`

Target platform: AWS Lambda

Service Engineering Research Area
ICCLAB SPLAB

Lambada Signature Converter

Going beyond just Lambda: Portable cloud functions

```
»» faasconverter: track module: test
»» faasconverter: convert function foo (x)
»» faasconverter: converted to module: test_portable.py

def foo(x):
        return 2*x

# FaaS-Converter wrapper for aws
def lambda_handler(event, context):
        return foo(event['x'])
# FaaS-Converter wrapper for ibm
def main(dict):
        return foo(event['x'])
```

Work in progress - helping hands welcome!

Service Engineering Research Area
ICCLAB SPLAB

Snafu: The “Swiss Army Knife“

Good News:

- developer tooling is improving
- Serverless framework, PyWren, several Lambada‘s...

Better News:

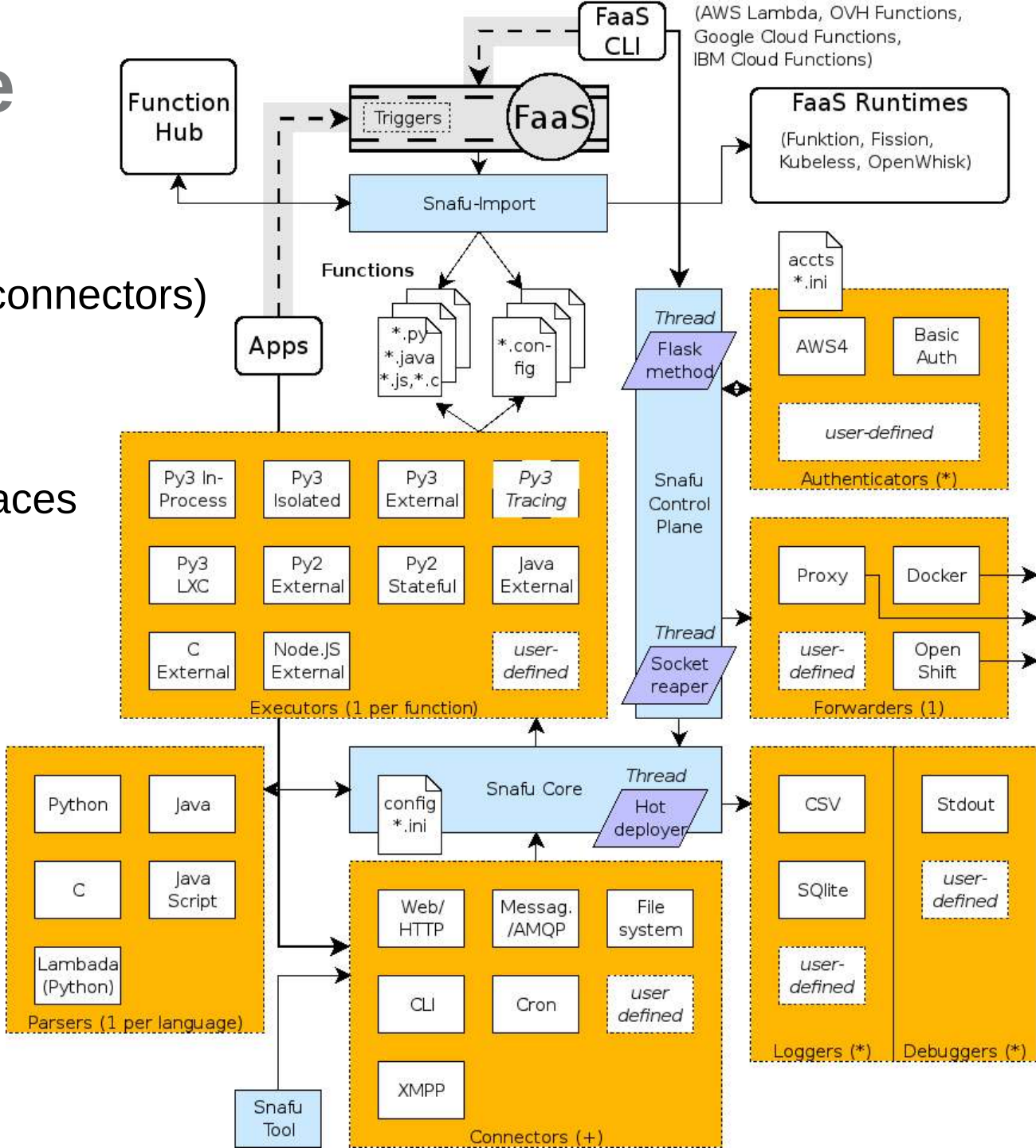
- more choice when deploying, executing, testing, migrating, sharing...

Service Engineering Research Area
ICCLAB SPLAB

Architecture

Subsystems
- parsing functions
- triggering functions (connectors)
- authentication
- forwarding
- executing functions
- logging output and traces

Language support
- Python
- Java, C, JavaScript
- generic (containers)

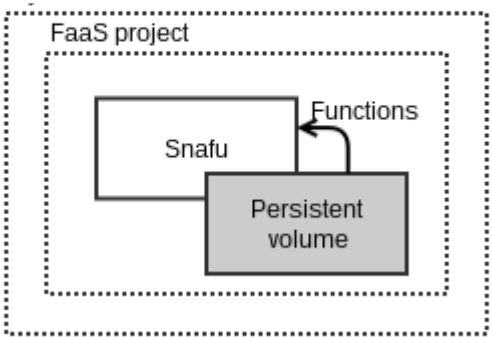


Service Engineering Research Area
ICCLAB SPLAB

Snafu Use Cases

Toying/Prototyping/Debugging

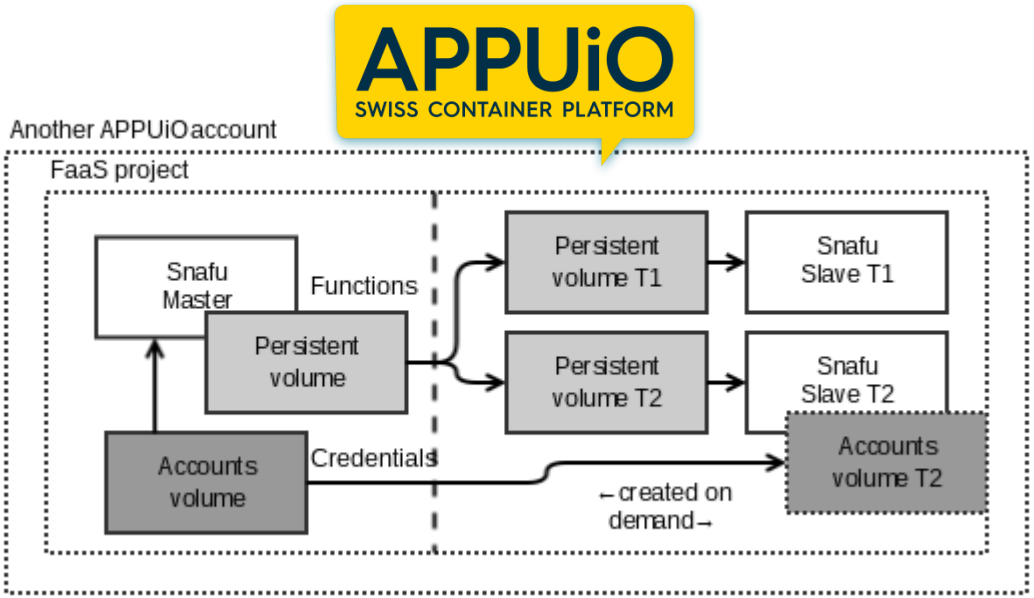
- directly from Git: `git clone https://github.com/serviceprototypinglab/snafu`
- or from PyPI: `pip install snafu`



Single-Tenant Operations

`docker run -ti jszhaw/snafu`

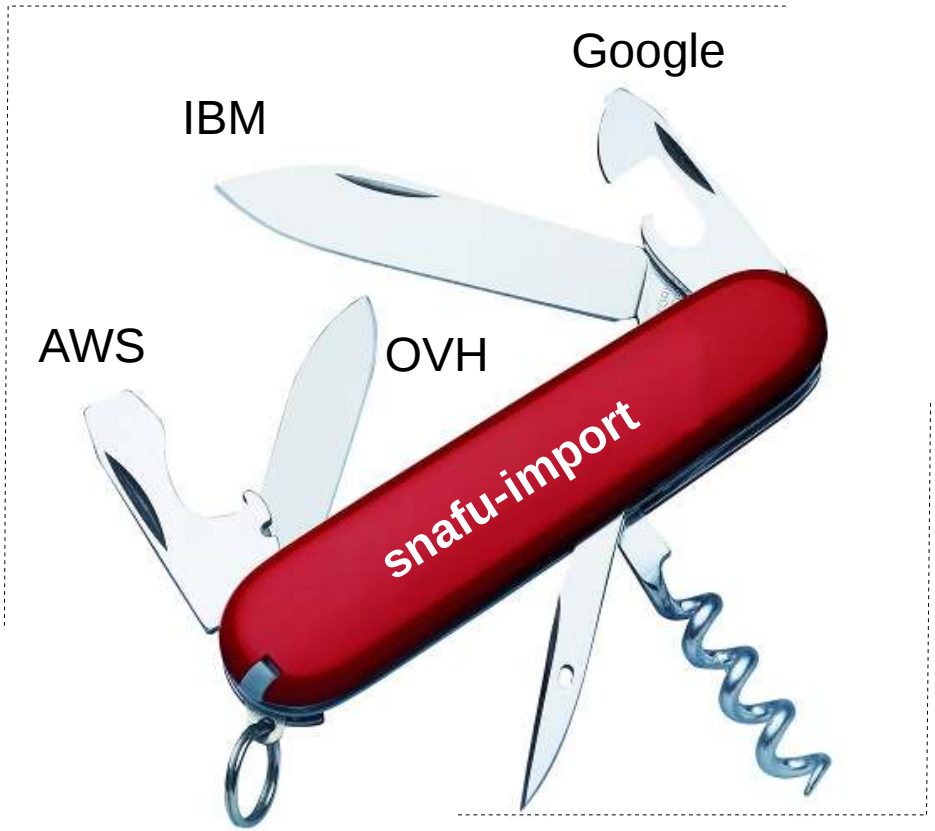
Multi-Tenant Operations



Service Engineering Research Area
ICCLAB SPLAB

Snafu Examples

Integration into the wider FaaS ecosystem




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

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

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

$ ./tools/patch-gcloud
```

Service Engineering Research Area
ICCLAB SPLAB

More Snafu Examples

```
# Zero-configuration interactive mode
$ snafu

# Invocation of a specific function from a known module
$ snafu -q -x helloworld.helloworld functions/helloworld.py

# Combination of various parameters: Java method with Lambda semantics
$ snafu -l sqlite -e java -c lambda -C messaging

# Using the Lambada function/method parser
$ snafu -f lambada -s test.ini
```

```
# Run function externally
$ snafu -X gfunctions

# Import/export
$ snafu-import -s gfunctions -t funktion -c myfunction
```

Service Engineering Research Area
ICCLAB SPLAB

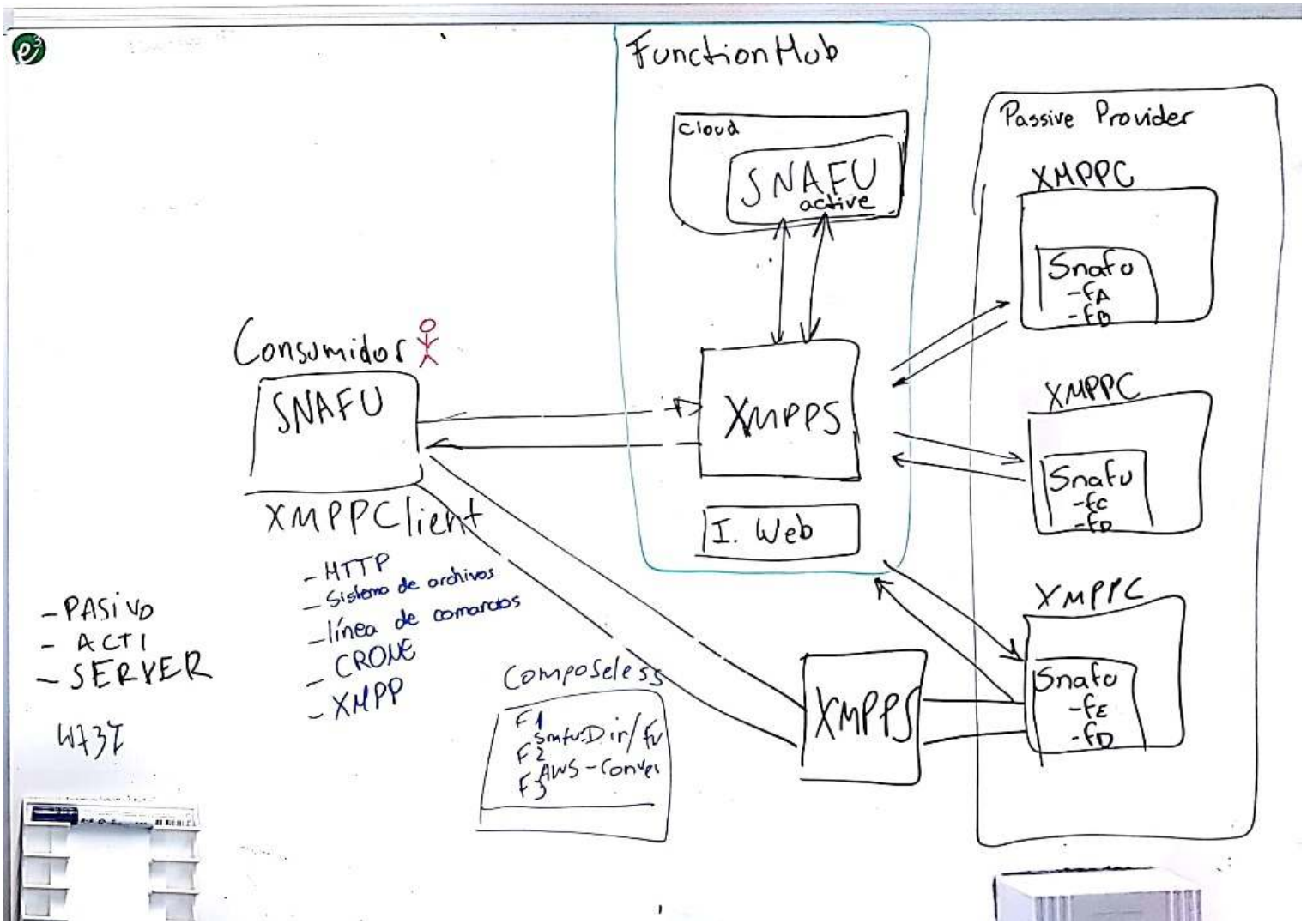
Even More Snafu Examples

```
# Zero-configuration FaaS daemon
$ snafu-control

# Lambda compatibility mode
$ snafu-control -a aws -r -d -e docker

# Multi-tenancy account management
$ snafu-accounts --add -k <k> -s <s> -e <ep>

# Safe mode
$ snafu-control -P
```

Service Engineering Research Area
ICCLAB SPLAB

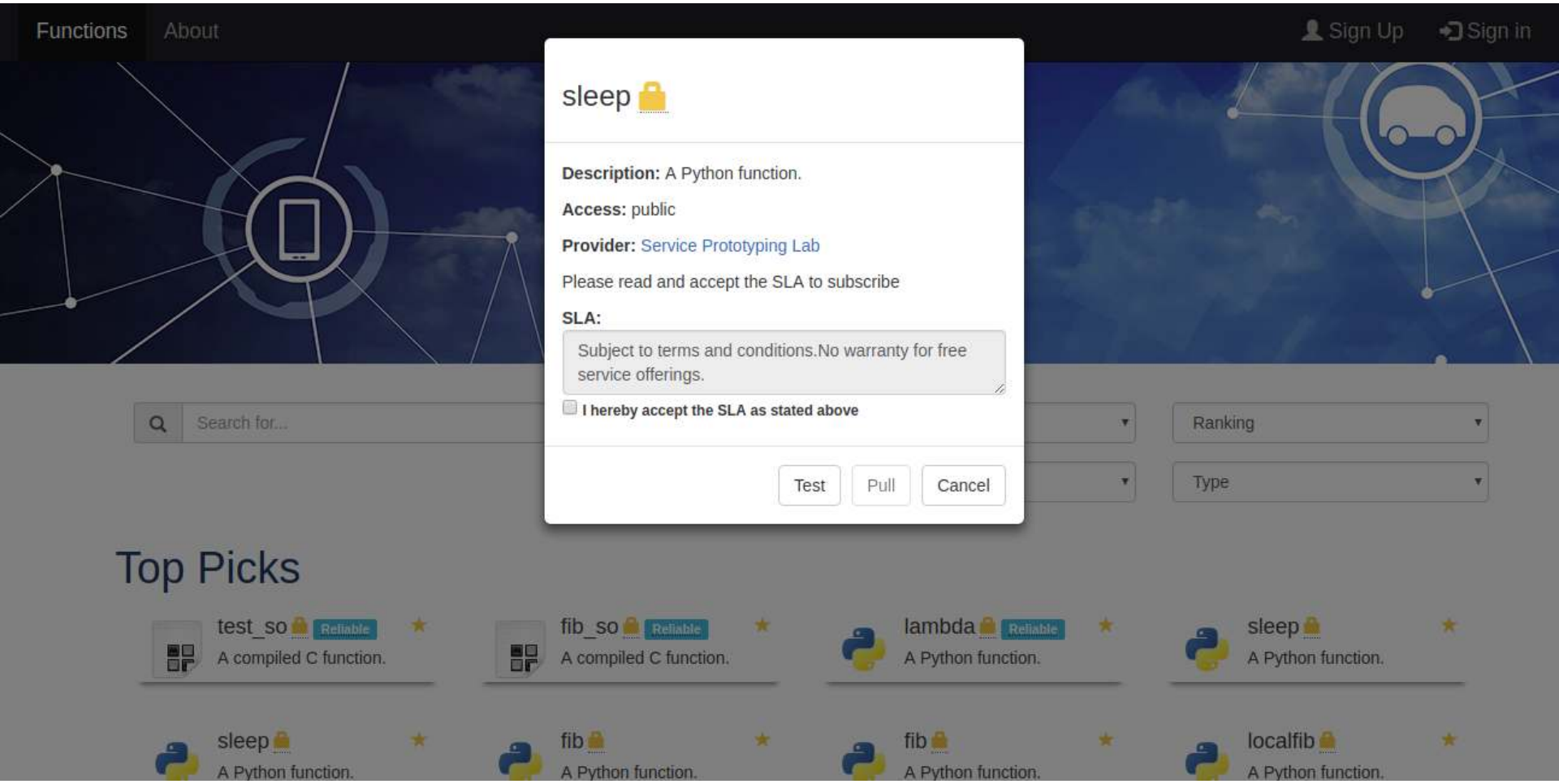
Open Function Ecosystems

Towards vibrant decentralised cloud function communities



Open Function Ecosystems

Our “Function Hub“ and “composeless“ prototypes



Work in progress - helping hands welcome (again)!

The Future of FaaS

Converters

Dev Tools

Optimisers

Marketplaces

Composers

Transformers

Runtimes

Debuggers

Deployers

European Symposium on Serverless Computing and Applications (ESSCA) - December 21, 2018, Zurich Toni-Areal - essca2018.servicelaboratory.ch

Service Engineering Research Area
ICCLAB SPLAB