Function-as-a-Service:

A Pythonic Perspective on Serverless Computing

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

Jun 13, 2017 | PyParis

Your Tutorial Agenda

50‘ FaaS overview and some existing tools

20‘ Lambada: Decompose your functions

20‘ Snafu: Run your functions

Your Tutorial Instructor

Josef Spillner <josef.spillner@zhaw.ch>

- works at Zurich University of Applied Sciences
- lectures Python programming to undergraduates & masters of advanced studies
- performs research in the Service Prototyping Lab

- co-authored «Architectural Transformations in Network Services and Distributed Systems»
- wrote many rarely used Python things since 2003

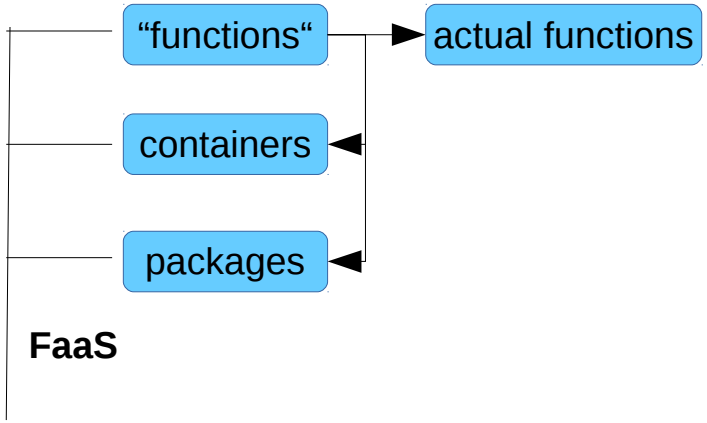
[LS16]

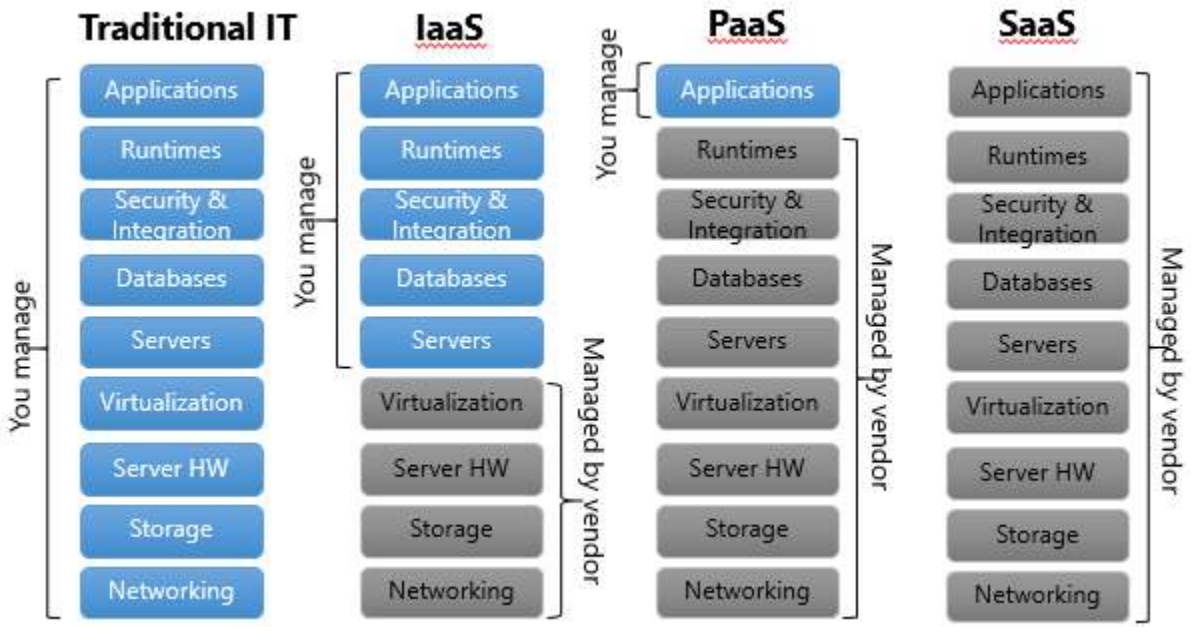
Service Prototyping Lab + ICCLab

Service Engineering Research Area
ICCLAB SPLAB

What is 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“

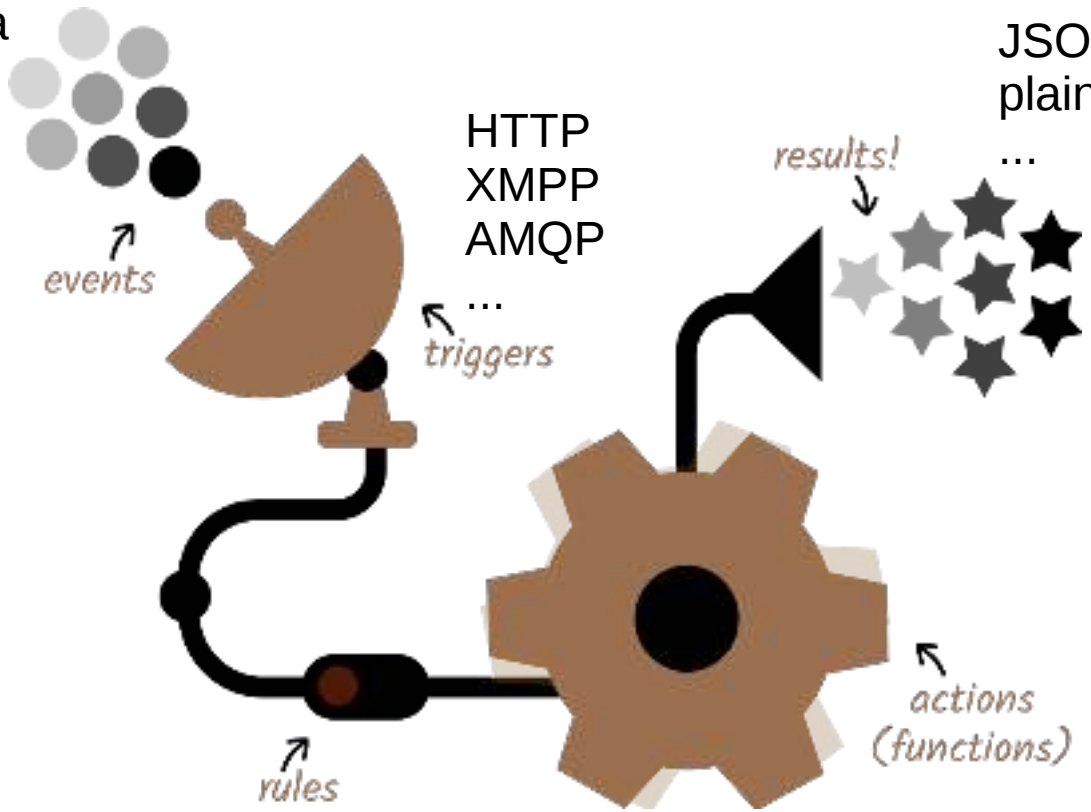




[mazikglobal.com]

Examples of FaaS

monitoring event
sensor data
log entry
git push
...



HTTP
XMPP
AMQP
...

JSON
plain text
...

Your Python functions!

max 1 per hour
...

[openwhisk.org]

The FaaS Space - in Python

AWS Lambda

Kubeless

Funktion

Snafu

Fission

Picasso

OpenWhisk

Elfe

Docker-LambCI

OpenLambda

Lever OS

PyWren
[Lambda]

Zappa
[Lambda]

Apex
[Lambda]

Serverless Framework
[Lambda, OW, GCF, AF]

X-Ray
[Lambda]

Step Functions
[Lambda]

ICCLAB SPLAB

The FaaS Space: Runtimes

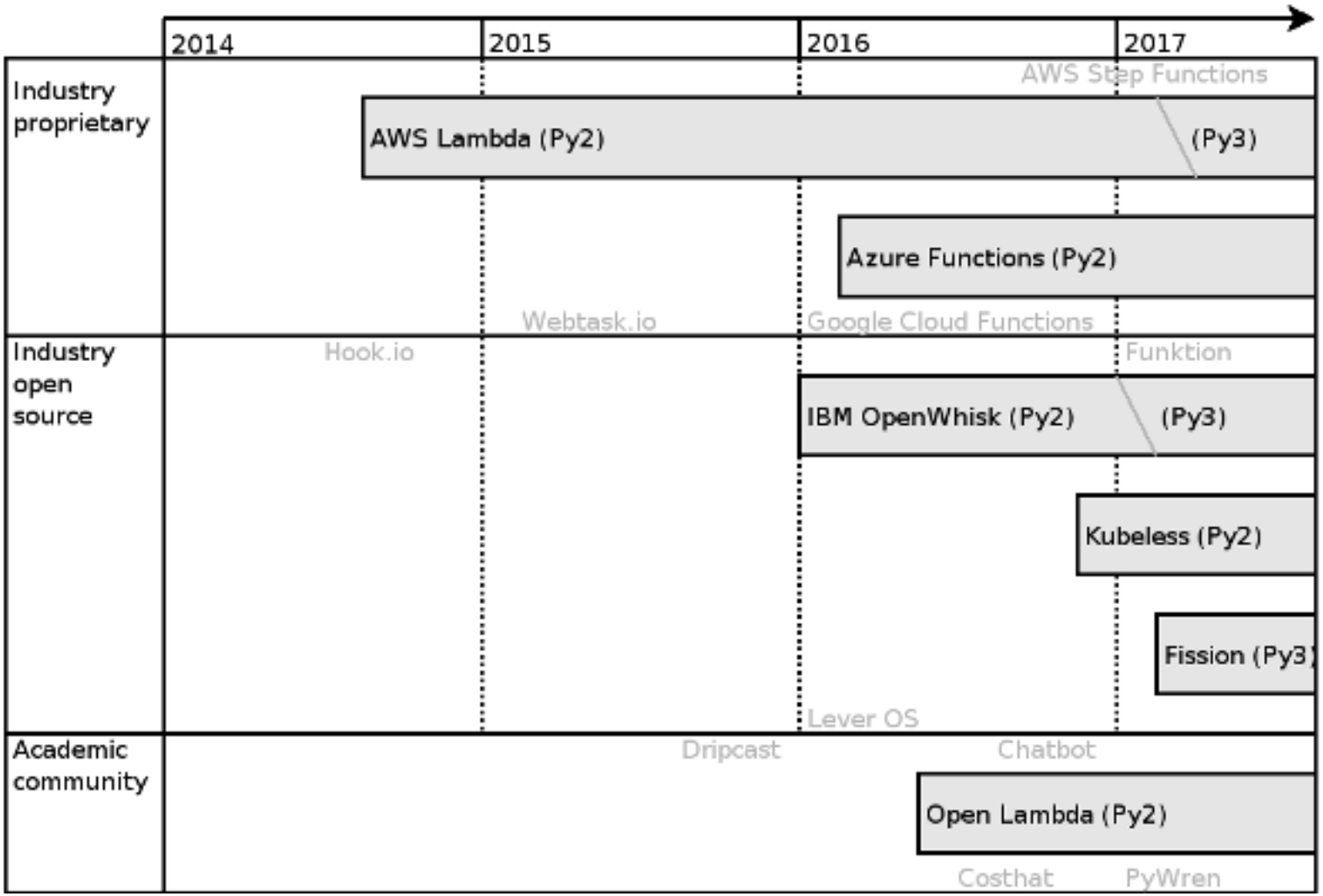
Function-as-a-Service offerings in greater detail...

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

ICCLAB SPLAB

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)

ICCLAB SPLAB

The World‘s Tools for FaaS Devs

OpenWhisk

$ wsk

does not compile

openlambda

$ bin/admin

(invoke read error)

$ az

not scriptable

$ kubeless

breaks minikube

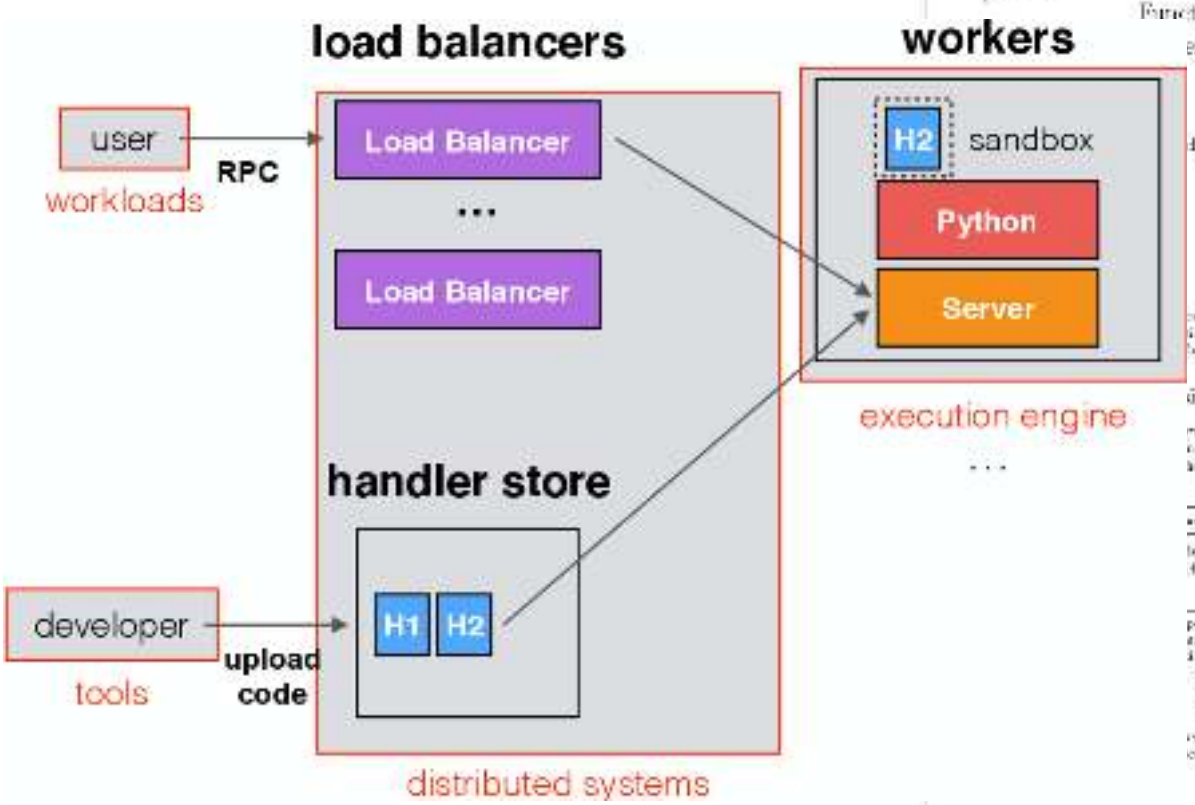
AWS Lambda

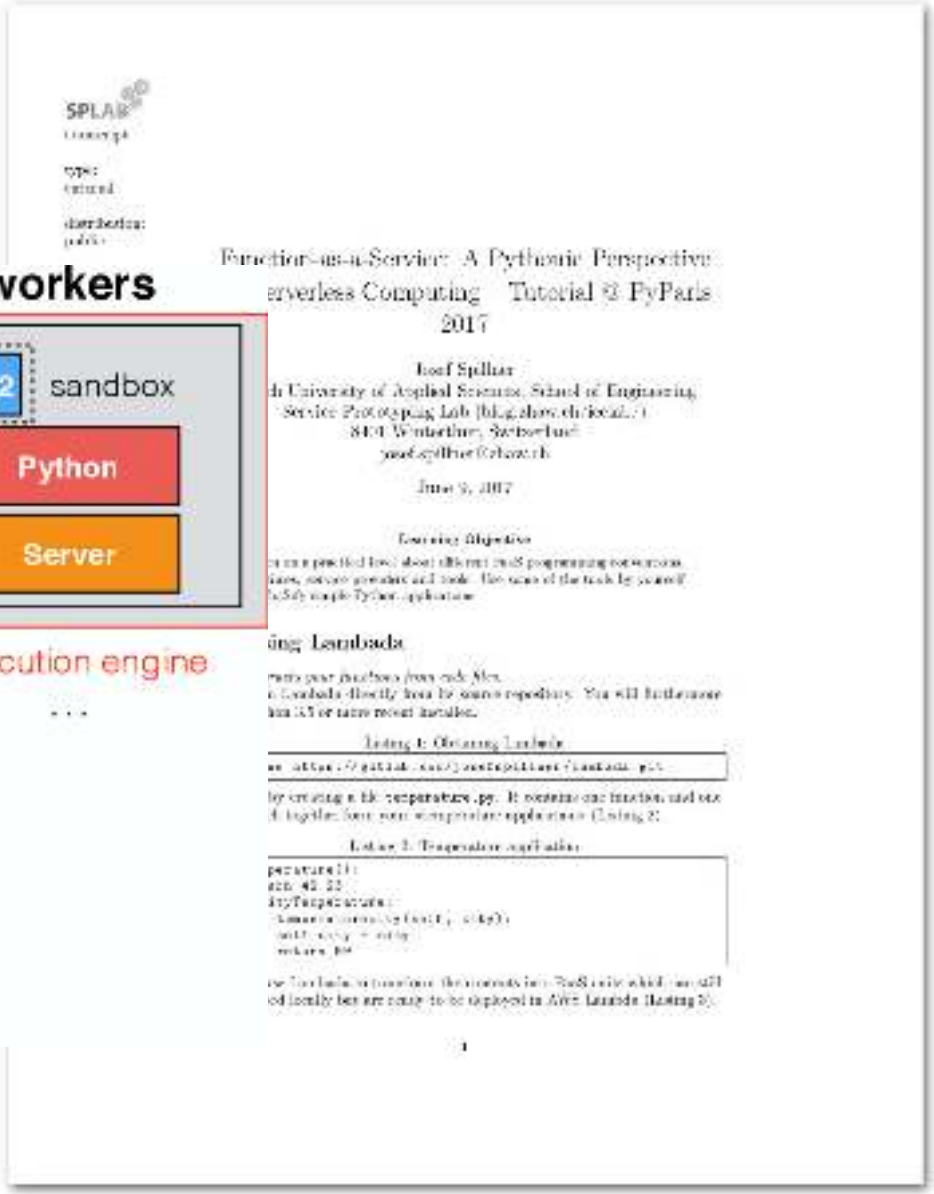
$ aws lambda

requires account

$ fission

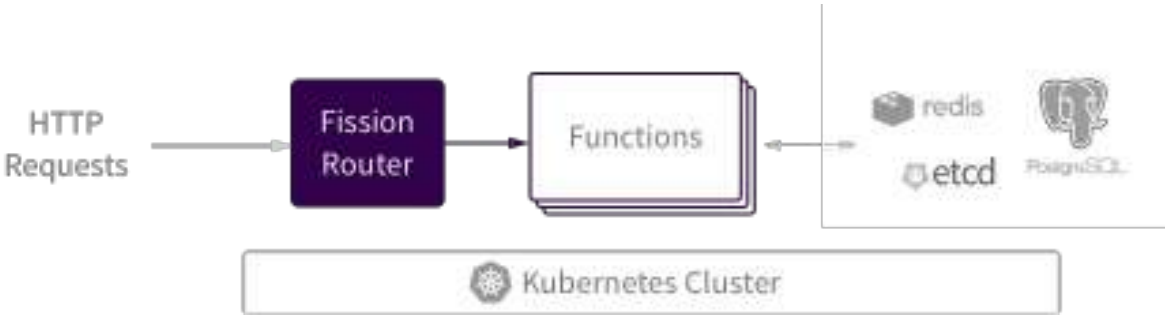
Open Lambda - Hands-on Time!





Fission - Hands-on Time!

fission



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:

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


Questions?

Our Tools for FaaS Devs

Podilizer
(Java)

Lambada
(Python)

today

Web2Cloud
(JavaScript)

Lambackup
(file backups)

Lama
(relational data)

Snafu
(FaaS host)

today

Lambada

Definition of “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

→ Lambada: FaaSification for Python
(currently limited to Lambdafication)

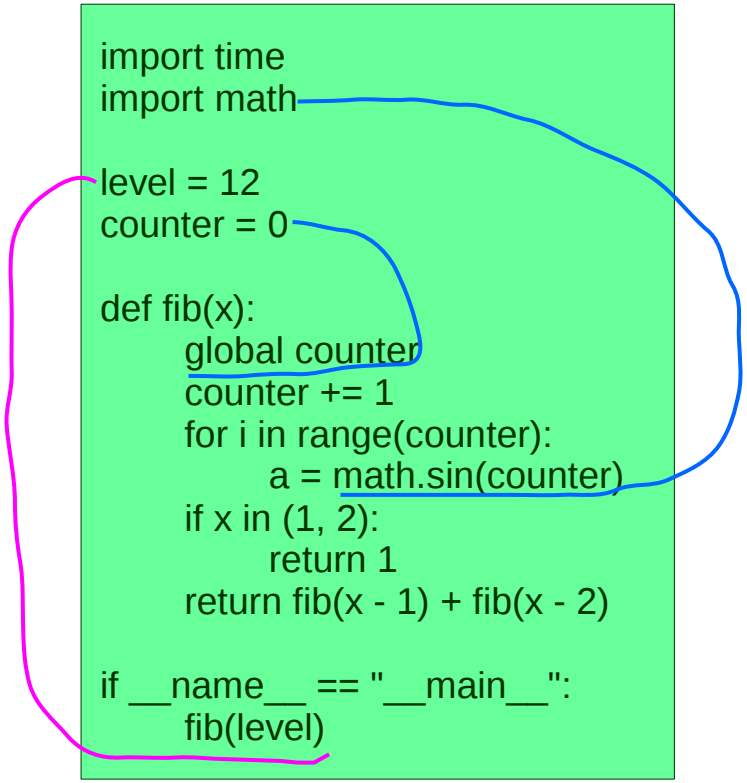
ICCLAB SPLAB Service Engineering Research Area

Lambada

Code Analysis

Dependencies
- imported modules
- global variables
- dependency functions
  - defined in other module
  - defined in same module

Input/Output
- printed lines
- input statements
  - tainting
  - stateful function splitting



Lambada

Code Transformation

Rewrite rules, via AST:

```
return 9
------------------
return {“ret“: 9}
```

```
print(“hello“)
return 9
----------------------------------------
return {“ret: 9“, “stdout“: “hello“}
```

```
local_func()
----------------------
local_func_stub()
```

Stubs, via templates:

```
def func_stub(x):
    input = json.dumps({“x“: x})
    output = boto3.client(“lambda“).invoke(FN=“func“, Payload=input)
    y = json.loads(output[“Payload“].read().decode(“utf-8“))
```

ICCLAB SPLAB Service Engineering Research Area

Lambada

Code Transformation

Stateful proxies for Object-Oriented Programming:

```
class Test:                    →    class Proxy:
     def __init__(self):                def __new__(cls, clsname, p=True):
          self.x = 9                         if p: # __new__ must return callable
                                                   return lambda: Proxy(clsname, False)
     def test(self):                         else:
          return self.x * 2                        return object.__new__(cls)

                                        def __init__(self, clsname, ignoreproxy): ...
                                        def __getattr__(self, name): ...
```

→ Test becomes Proxy(“Test“), Test() then invokes proxy
→ test() becomes remote_test({“x“: 9}) through network proxy class
→ automatically upon import of class

Lambada

Code Deployment + Activation

```
$ lambada [--local] [--debug] [--endpoint <ep>] <file.py>
$ python3 -m lambada <file.py>

>>> import lambada
>>> lambada.move(globals() [, endpoint="..."])
```

Local mode: source code modified locally as copy

Remote mode: rewritten source code deployed and invoked

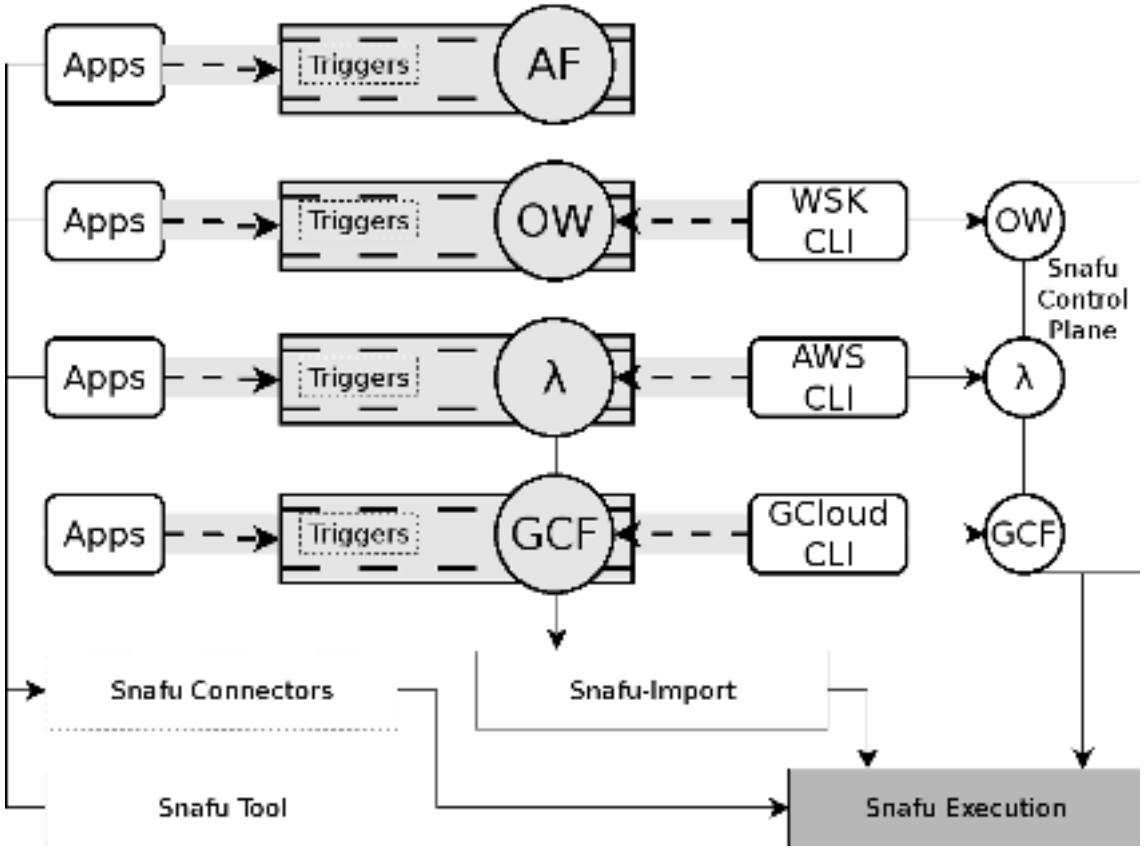
Lambada - Hands-on Time!

[d0wn.com]


Questions?

Snafu

The Swiss Army Knife of Serverless Computing



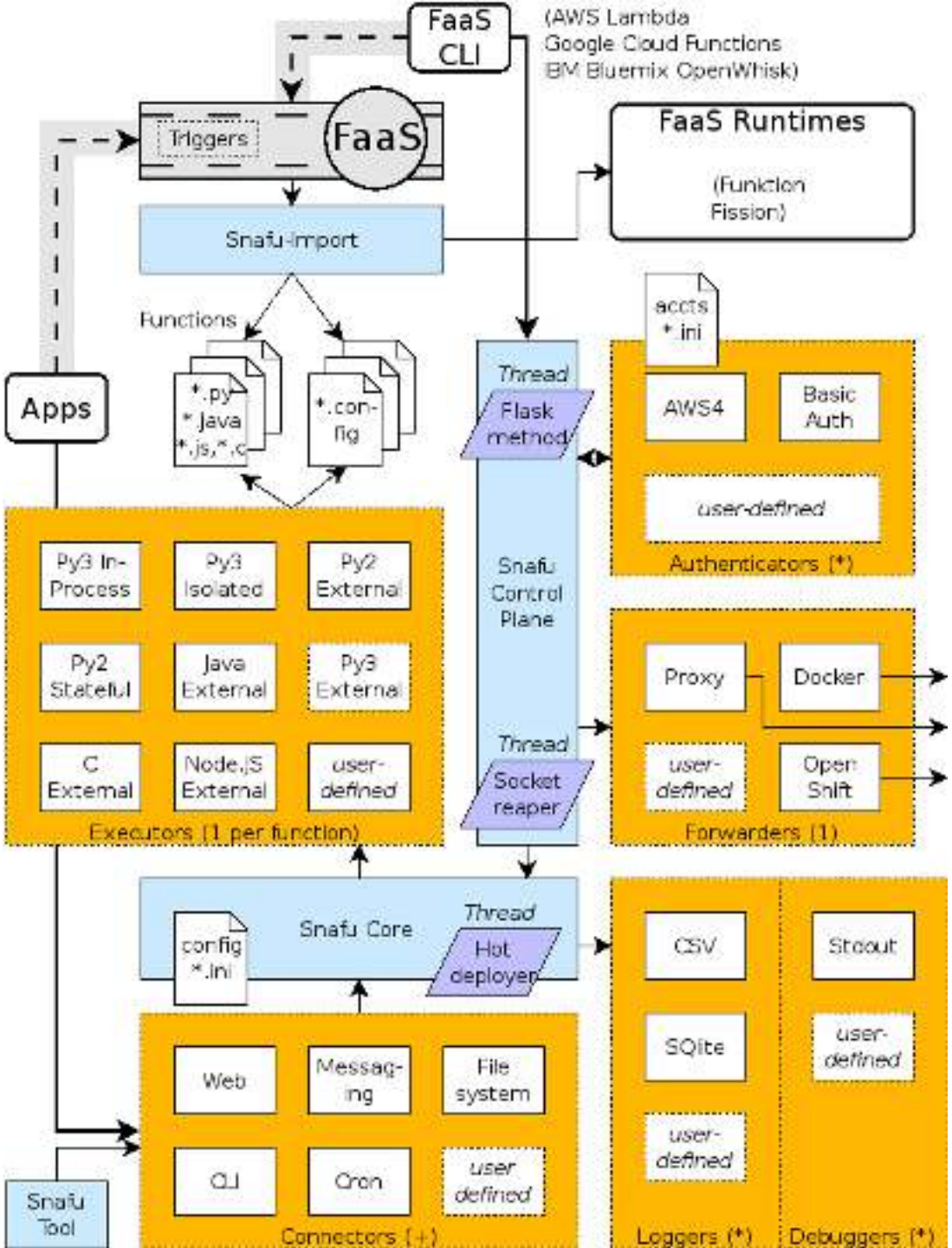
Snafu

Current Implementation

- scalable from developer single instance to multi-tenant deployments
- executes Python 2 & 3, Java, JavaScript, C
- integrates with FaaS ecosystem at-large
- extensible subsystems

SLOC: ~1800
(including subsystems: ~800)

$ pip install snafu
$ docker run -ti jszhaw/snafu



Snafu

Standalone mode

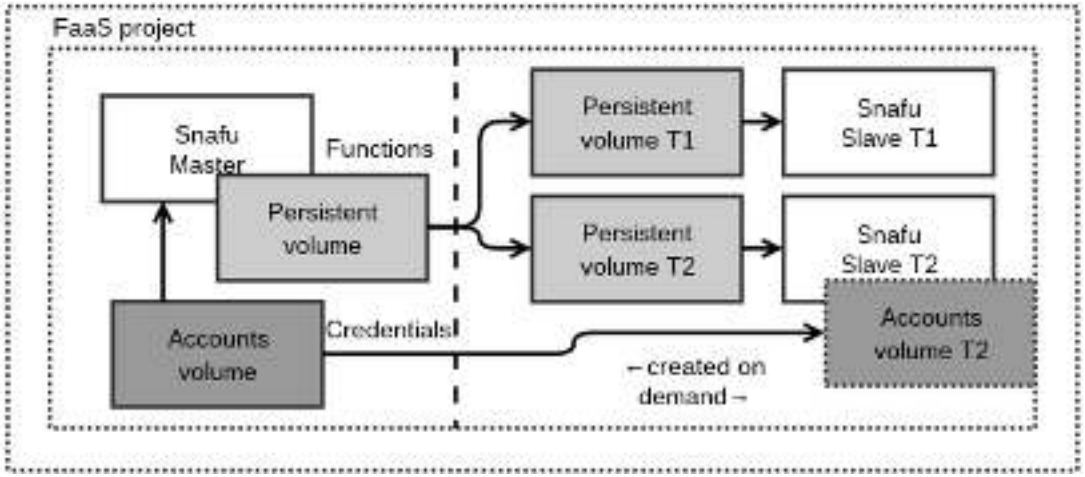
- call functions interactively
- batch mode with/without input pipe
- performance, robustness & correctness tests
- development

```
$ snafu

$ snafu -x <function> [<file/dir>]

$ snafu -l sqlite -e java -c lambda -C messaging
```

Snafu

Daemon mode (control plane)

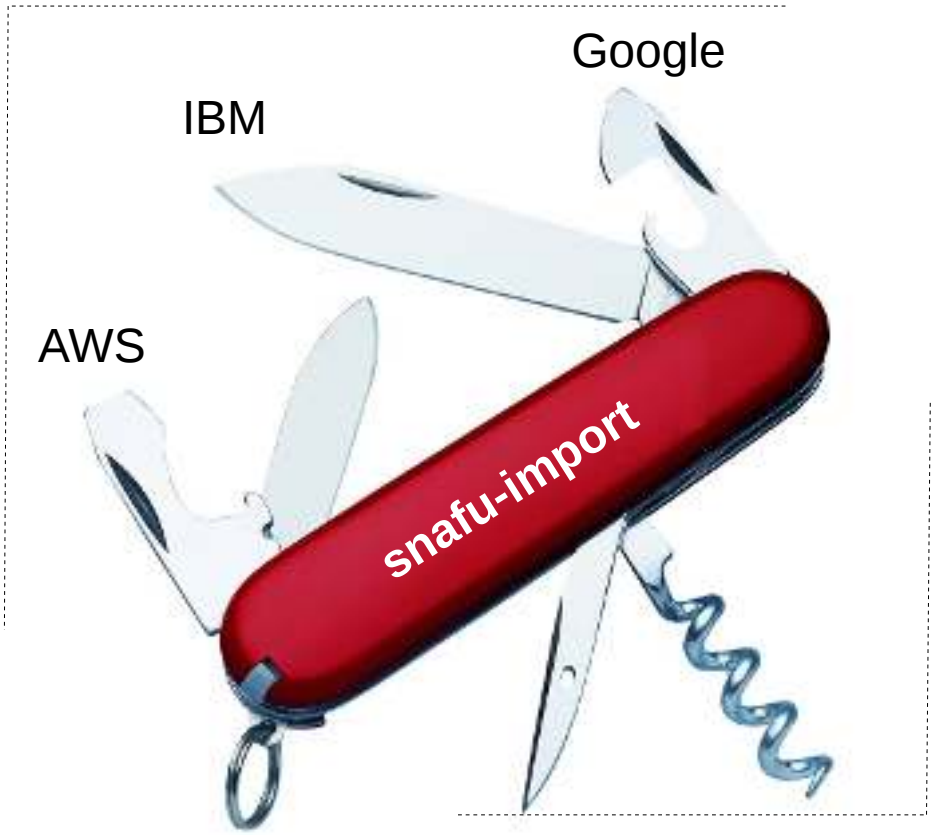
- hosted functions
- multi-tenant provisioning
- per-tenant isolation
- compatibility with existing client tools




```
$ snafu-control

$ snafu-control -a aws -r -d -e docker

# snafu-accounts --add -k <k> -s <s> -e <ep>
```

Snafu

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
```

Snafu - Hands-on Time!

[pinterest.com]

Q&A / Live help session

[dribbble.com]

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]