

# Future Cloud Applications #2

## More on FaaS: The Swiss Army Knife of Serverless Computing

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

Apr 27, 2017 | Future Cloud Applications

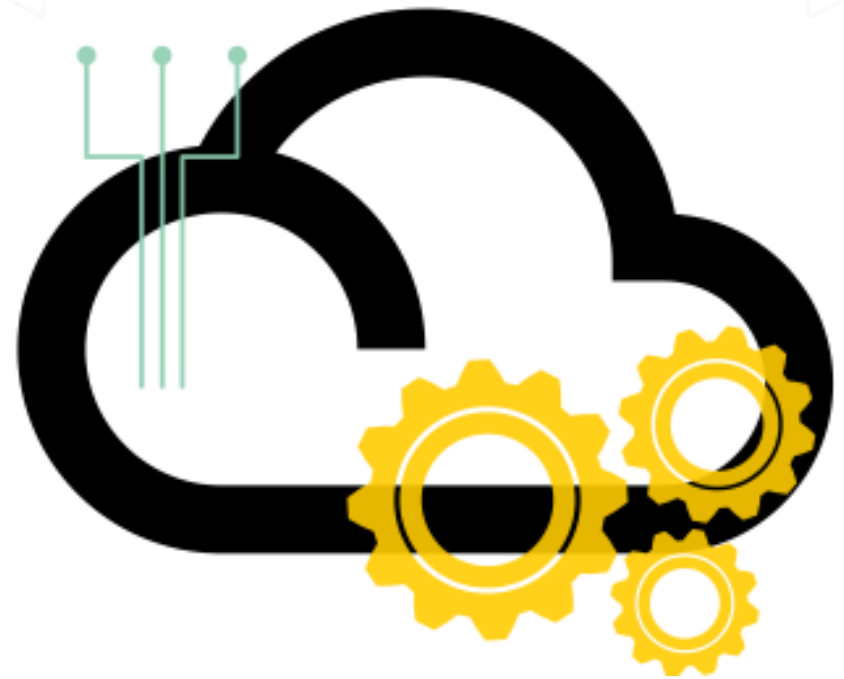
# This Event

## Future Cloud Applications

- link academic research  $\leftarrow \rightarrow$  industry
- highly technical
- no specific technology preference
  - for this, other meetups exist
- ~every 2-3 months
  - to be calibrated

## Participation

- meetup group (55+ people)
- open for all
- open for talk proposals!

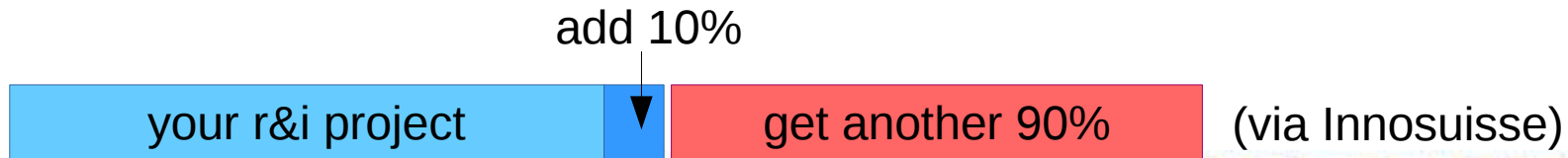


[cyprus-property-buyers.com (no, really)]

# Service Prototyping Lab

## Research approach

- ambitious long-term research initiatives
  - cloud-native applications, **service tooling**, cloud accounting & billing
- transfer of results into Swiss companies



## Publishing approach

- preprint-first - rapid dissemination
- open source, Labsite
- blog posts, events

# Service Prototyping Lab + ICCLab



# The FaaS Space



**AWS Lambda**



Google Cloud Platform  
Functions



**OpenWhisk**



Fission



OpenLambda

Webtask.io

Hook.io

Step Functions

PyWren

Zappa

X-Ray

Serverless Framework

Apex

Kubeless

Funktion

Picasso

Effe

Docker-LambdaCI

Lever OS



# FaaS Examples

## 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:

```
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)

# Our Tools for FaaS

Podilizer  
(Java)

Lambada  
(Python)

*today*

Web2Cloud  
(JavaScript)



Lambbackup  
(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

## Decomposition Categories:

- static code analysis
- dynamic code analysis

→ Lambada: FaaSification for Python



# Lambda

## 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

```
import time
import math

level = 12
counter = 0

def fib(x):
    global counter
    counter += 1
    for i in range(counter):
        a = math.sin(counter)
    if x in (1, 2):
        return 1
    return fib(x - 1) + fib(x - 2)

if __name__ == "__main__":
    fib(level)
```

# Lambda

## Code Transformation

Rewrite rules, via AST:

<code>return 9</code>	<code>print("hello")</code>	<code>local_func()</code>
-----	<code>return 9</code>	-----
<code>return {"ret": 9}</code>	-----	<code>local_func_stub()</code>
	<code>return {"ret: 9", "stdout": "hello"}</code>	

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"))
```

# 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

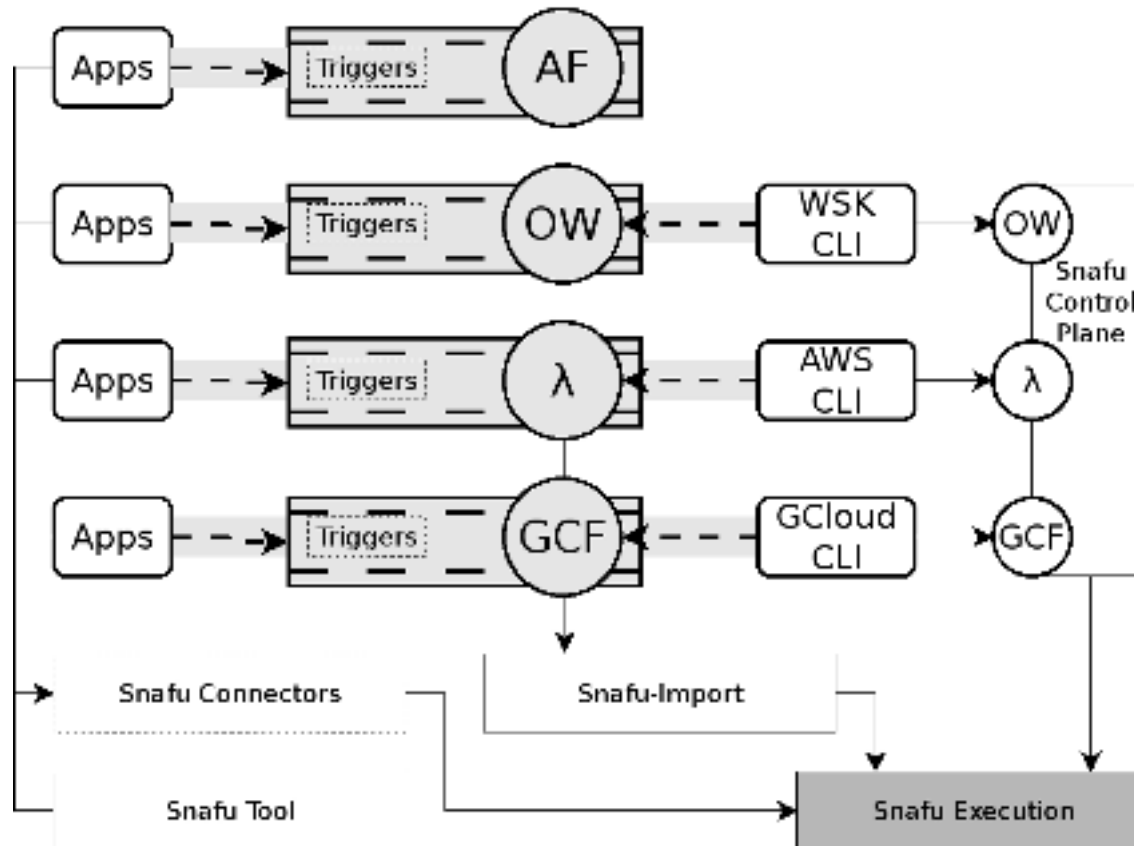
# Lambada - Demo Time!



[d0wn.com]

# Snafu

The Swiss Army Knife of Serverless Computing



# Snafu

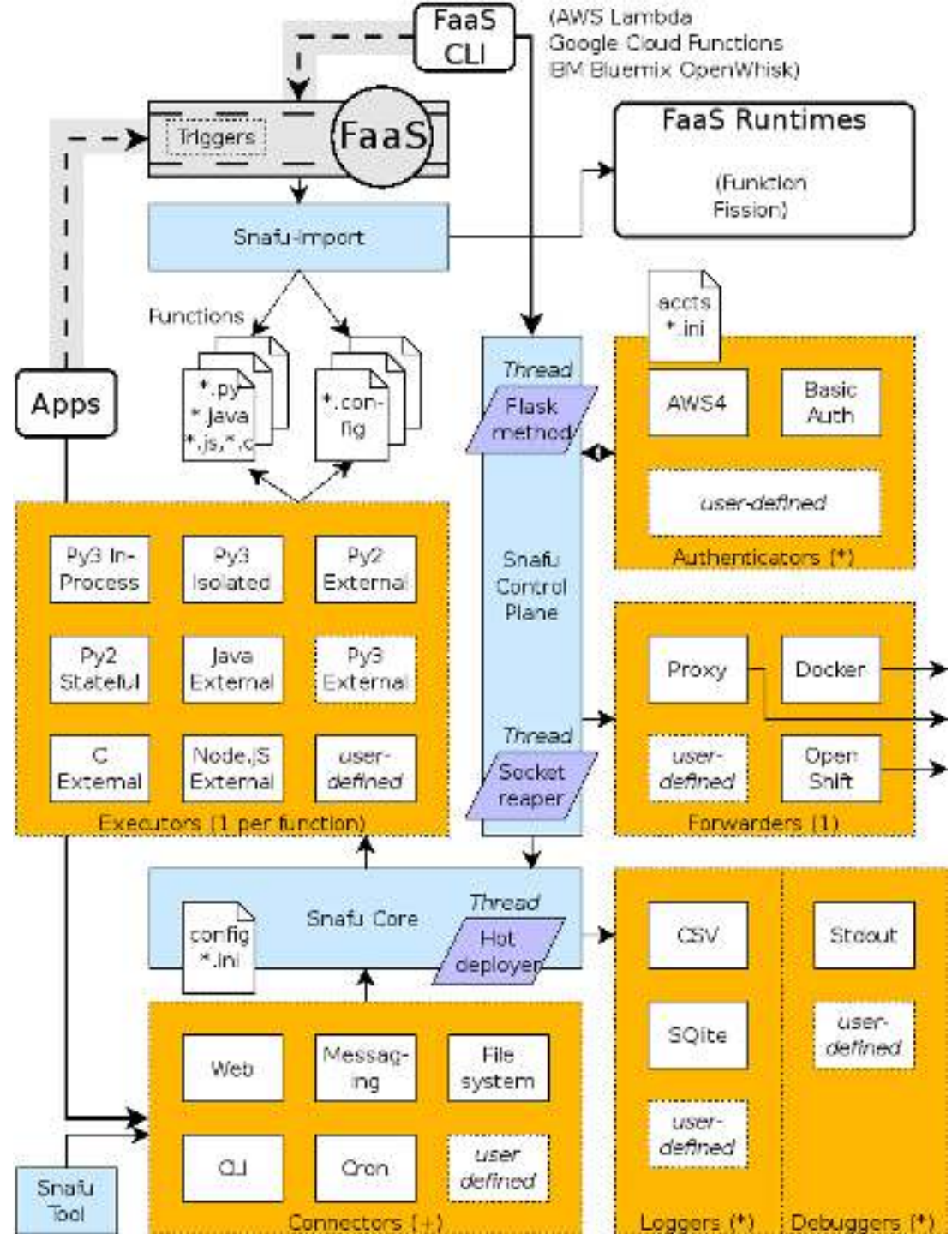
## Current Implementation

- 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>]
```

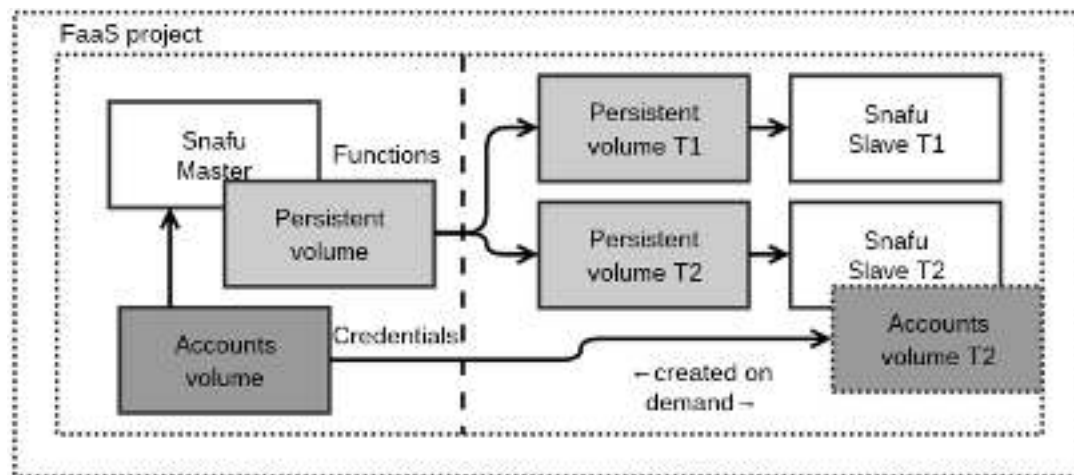
```
$ snafu -l sqlite -e java -c lambda -C messaging
```



# Snafu

## Daemon mode

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



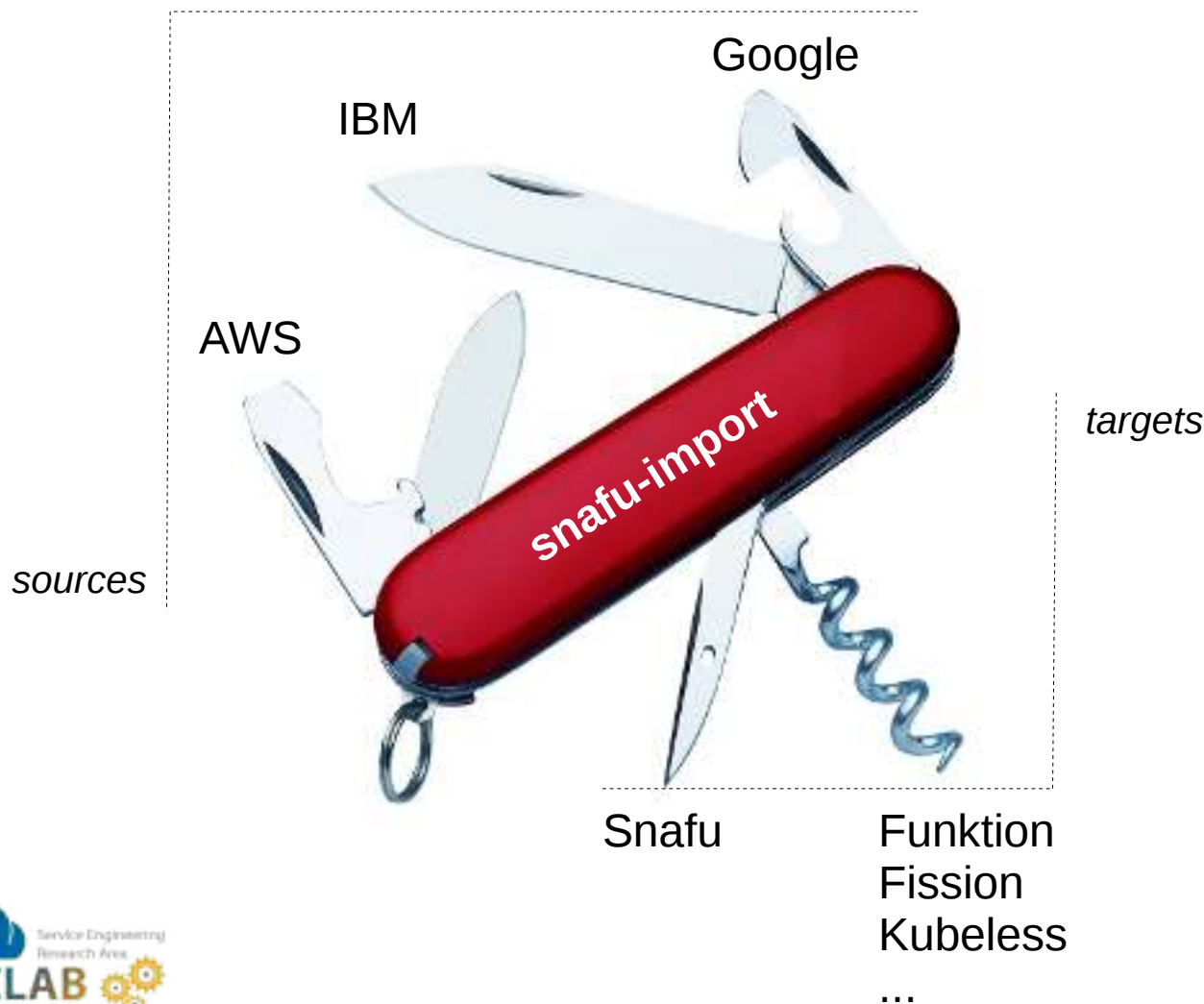
```
$ snafu-control
```

```
$ 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 - Demo Time!




[pinterest.com]

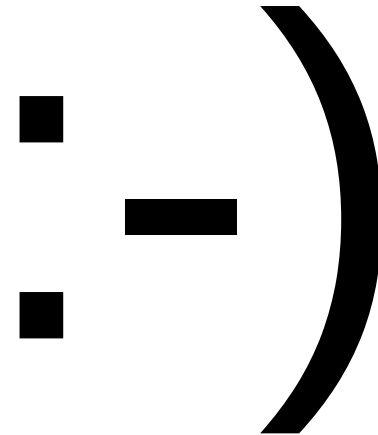
# One more thing...

## Challenges of Serverless

- Opinionated programming model
  - Aligned with 12-factor approach to cloud-native applications
- Per-handler resource allocation limits
- Per-invocation latency & overhead
- Lack of high-performance persistent state
- Ability to reuse and share handler functions ('marketplace')
- Lifecycle management of composite serverless applications
- Monitoring, error handling, testing, debugging



[slideshare.net/Alex Glikson]



*Full demo @ Open Cloud Day 14.06.2017 in Bern*

# Further Reading and FaaS Fun

Lama, Lambbackup:

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

Podilizer:

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

Snafu:

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

On arXiv Analytics:



On GitHub:



[[github.com/serviceprototypinglab](https://github.com/serviceprototypinglab)]



# Next Future Cloud Applications Event!

Our suggestion: around July 2017... topic: elasticity boundaries for compositions of stateful and stateless microservices

meetup

The screenshot shows a Meetup page for the group "Future Cloud Applications". The page has a red header with the group name. Below the header is a navigation bar with links for Home, Members, Photos, Discussions, and More, and a "Join us!" button. The main content area features a large image of a cloud with gears, a description of the event's aim, and a "Join us" button. There are also sections for "Who do I know here?" and "What's new".

## Future Cloud Applications

Home Members Photos Discussions More [Join us!](#)

The aim of this event is to bring together software developers with researchers. There are lots of interesting ideas floating around on how to construct the next-generation cloud applications and services. But some never leave the academic space due to a lack of communication. We can fix this: Join, see for yourself and discuss with us!

[Join us](#)

Join us and be the first to know when new Meetups are scheduled

[Who do I know here?](#)

Log in with Facebook to find out

Members: 88  
Upcoming Meetups: 1  
Calendar

Organizer: JS

Contact

### Welcome!

[Upcoming \(1\)](#) [Calendar](#)

#### Using AWS Lambda the cool way: Podlizer, Lambakup & Lama

ZHAW  
Technikumstr. 9, building TH 949, Winterthur (map)

Thu Feb 23  
6:00 PM  
RSVP

3 cancelled  
12 going  
0 comments

#### What's new

- ROMAN F. joined 1h ago
- SANDRO T. joined 3 days ago
- GORG A. RSVPed Yes for Using AWS Lambda the cool way: Podlizer, Lambakup & Lama 4 days ago
- GORG A. joined

**We're about:**  
Software Development - New Product  
Development: Software &

Service Engineering Research Area  
**ICCLAB**  
**SPLAB**