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)

Apr 27, 2017 | Future Cloud Applications
This Event

Future Cloud Applications
- link academic research ↔ 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!
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

your r&i project \[\Rightarrow\] get another 90% (via Innosuisse)
The FaaS Space

AWS Lambda

OpenWhisk

Webtask.io

Hook.io

PyWren

Zappa

Serverless Framework

Apex

Kubeless

Funktion

Fission

Picasso

Effe

Docker-LambCI

OpenLambda

Lever OS

X-Ray

Step Functions
FaaS Examples

AWS Lambda:

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

OpenWhisk:

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

Fission:

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

Azure:

```python
def main():
    from AzureHTTPHelper import\n    HTTPHelper
    input = HTTPHelper().post
    # ...
    open(os.environ["res"], "w").write(\n    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)

Web2Cloud (JavaScript)

Lambackup (file backups)

Lama (relational data)

Snafu (FaaS host)

today

today
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
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

```python
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)
```
Lambada

Code Transformation

Rewrite rules, via AST:

```python
return 9  
print("hello")

return 9

return \{"ret": 9\}

return \{"ret": 9, "stdout": "hello"\}
```

Stubs, via templates:

```python
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:
    def __init__(self):
        self.x = 9
    def test(self):
        return self.x * 2

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

class Proxy:
    def __new__(cls, clsname, p=True):
        if p: # __new__ must return callable
            return lambda: Proxy(clsname, False)
        else:
            return object.__new__(cls)

    def __init__(self, clsname, ignoreproxy):
    def __getattr__(self, nime):

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

Genuine, authentic Chinese imitation of a real SWISS TYPE ARMY KNIFE
13 blades or functions.

$1.97

[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

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:  [github.com/serviceprototypinglab]
Next Future Cloud Applications Event!

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