Serverless Applications:

Tools, Languages, Providers and (Research) Challenges

Josef Spillner <josef.spillner@zhaw.ch>
Service Prototyping Lab (blog.zhaiw.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"
Examples of FaaS: Process

monitoring event
sensor data
log entry
git push
...

HTTP
XMPP
AMQP

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

results!

Your Python
functions!

JSON
plain text

...
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
The FaaS Space

Languages

Python
JavaScript
Java
Go
C#

(language-independent service representation)

Providers

Runtimes

SDKs/Integration

Tools

Software

→ Functions ←
The FaaS Space: Runtimes

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

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

Trend: Sooner or later → gaps will be filled
The FaaS Space: Python runtimes

- **AWS Lambda (Py2)**
  - 2014-2015
- **Azure Functions (Py2)**
  - 2016
- **Google Cloud Functions**
  - 2016
- **IBM OpenWhisk (Py2)**
  - 2016
- **Kubeless (Py2)**
  - 2016
- **Fission (Py3)**
  - 2017
- **Open Lambda (Py2)**
  - Academic community
- **Dripcast**
  - 2016
- **Chatbot**
  - 2016
- **Lever OS**
  - 2016
- **Costhat**
  - 2016
- **PyWren**
  - 2016

**Industry proprietary**
- AWS Step Functions
- Webtask.io
- Facebook

**Industry open source**
FaaS Synopsis in Python

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 Functions:

```python
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)

- Use frameworks
- Do some crazy stuff to get metrics
- API gateway is terrible
FaaS Programmer Perspective
Snafu

The Swiss Army Knife of Serverless Computing
FaaSification

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

\[
\text{FaaSification} := \text{code analysis} + \text{transformation} + \text{deployment} + \text{on-demand activation}
\]

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

Decomposition Categories:
- static code analysis
- dynamic code analysis

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

“Lambdafication“:
- FaaSification to Lambda

Languages:
Java & Python

Tools:
- Termiter
- Podilizer
- ICLLAB SPLAB
- DEMO
The FaaS Space

AWS Lambda

OpenWhisk

Google Cloud Platform Functions

Microsoft Azure

Serverless Framework

Chalice [Lambda]

Zappa [Lambda]

Dawson [Lambda]

PyWren [Lambda]

Termite

whisk-mochi

Fission

FaaS (Docker)

Kubeless

Funktion

Snafu

Effe

Docker-LimbCl

OpenLambda

Lever OS

PyWren [Lambda]

Zappa [Lambda]

Dawson [Lambda]

Serverless Framework [Lambda, OW, GCF, AF]

MR Refarg [Lambda]

LambDash [Lambda]
Further Reading and FaaS Fun

Lama, Lambackup:

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

Snafu:

Lambada

On arXiv Analytics:

On GitHub:
[github.com/serviceprototypinglab]
BACKUP...
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