State of Serverless

Doug Davis (dug@us.ibm.com | @duginabox)
Agenda

• What is Serverless?
• Why and when to use Serverless?
  – Serverless vs ...
  – Use cases
• CNCF Serverless Working Group
• CloudEvents and beyond
But first...Functions as a Service (FaaS)

Increasing focus on business logic

Decreasing concern (and control) over infrastructure implementation

- Faster start-up times
- Better resource utilization
- Finer-grained management
- Splitting up the monolith
What is a Function?

- Event-driven
- Short duration
- Stateless
- Lower cost
What is a Function?

• Example:

```javascript
/* Javascript example */
function main(params) {
  var name = params.name || 'World';
  return {payload: 'Hello, ' + name + '!'};
}
```

E.g.

https://openwhisk.ng.bluemix.net/api/v1/web/dug%40us.ibm.com_dev/default/test.json

• Framework handles hosting and infrastructure to deal with incoming messages and response
  – Provide access via an HTTP(s) API
  – Connect to a set of "Actions"
  – Chaining functions to orchestrate
Functions as a Service

Event Sources

Actions

Function Execution

Backend Services

Kubernetes, Docker, and/or Hypervisor

Compute, Network, Storage

Business Intelligence

Analytics

Databases
• Serverless takes FaaS and adds the notion of:
  • Infrastructure manages the auto-scaling of the functions based on demand
  • Infrastructure manages the scaling down to zero instances when not being invoked
  • Fine grained, pay just for what you use, cost model
  • Zero cost when not being executed (more on this later)
  • Except for persistent storage type of resources
• "Serverless" means not needing to worry about managing the server
Serverless vs PaaS / CaaS

• Very similar
  – Especially if PaaS / CaaS has auto-scaling feature

• Similar mind-shift for VM -> PaaS / CaaS
  – Remove the OS and just deploy your app
  – Remove the "app" and just deploy a "set of functions (APIs)"

• Biggest difference is the scope of the code being deployed

• Function vs Application
  – Decompose monolithic app to individual function endpoints
  – E.g. can scale just "GETs" vs "Entire App" (or microservice) based on demand
## Serverless / FaaS use cases

<table>
<thead>
<tr>
<th>Use Case</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>micro-service</td>
<td>Easily implement fine-grained, micro-service APIs.</td>
</tr>
<tr>
<td>IoT</td>
<td>Power various mobile, web and IoT app use cases by scaling and simplifying the programming model of orchestrating various services.</td>
</tr>
<tr>
<td>Batch and Stream Processing</td>
<td>Automate and control batch and stream processing</td>
</tr>
<tr>
<td>DevOps</td>
<td>Automate DevOps pipeline based on events triggered from successful builds or completed staging or a go-live event.</td>
</tr>
<tr>
<td>IT/Ops</td>
<td>Allow an easier deployment model for administrative functions (bots) to run for IT/Ops.</td>
</tr>
</tbody>
</table>

**Net:** Event Driven & Reusable Utilities
CNCF Serverless Working Group
CNCF Serverless Working Group

- June 2017 at the request of CNCF Technical Oversight Committee (TOC)
- State of tech/community & recommendations for possible involvement
- Most key Serverless players involved
  - IBM, VMWare, Google, Red Hat, Huawei, Microsoft, AWS, SolarWinds, Docker, iguazio, Amazon, MasterCard, Pivotal, Serverless Inc., Clay Labs, The New Stack, A Cloud Guru, Platform9, Bitnami, Auth0, Hyper, ...
  - To date, 51 different companies have been involved
  - On average ~30 people join our weekly calls
CNCF Serverless WG: White Paper

- Describes & defined Serverless as it exists today in the community
  - Common vocabulary
  - Differentiates Serverless from FaaS, PaaS, CaaS and Container Orchestration
  - Describes the mechanics of a generic Serverless system
  - Roles: Provider vs Developer
  - Zero cost when idle (except e.g. stateful storage costs)
  - Public vs Private

- Highlights promising use cases and areas where already proven value

- Recommendations for potential future CNCF activities
## CNCF Serverless WG: Landscape

<table>
<thead>
<tr>
<th>Project Name (Serverless/FaaS)</th>
<th>Sponsors</th>
<th>Homepage</th>
<th>Orchestration</th>
<th>Languages</th>
</tr>
</thead>
<tbody>
<tr>
<td>AWS Lambda</td>
<td>Amazon</td>
<td><a href="https://aws.amazon.com/lambda/">https://aws.amazon.com/lambda/</a></td>
<td>Node.js (JavaScript), Python</td>
<td>Node.js</td>
</tr>
<tr>
<td>Google Cloud Functions</td>
<td>Google</td>
<td><a href="https://cloud.google.com/functions/">https://cloud.google.com/functions/</a></td>
<td>Any language, Docker images</td>
<td>Node.js, Swift, Python, JavaScript</td>
</tr>
<tr>
<td>Hyper Func</td>
<td>Hyper.sh</td>
<td><a href="https://docs.hyper.sh/Feature/container/func.html">https://docs.hyper.sh/Feature/container/func.html</a></td>
<td>Node.js, Python, Java</td>
<td>Go/C (native), Python/Java</td>
</tr>
<tr>
<td>IBM Cloud Functions</td>
<td>IBM</td>
<td><a href="https://console.bluemix.net/openwhisk/">https://console.bluemix.net/openwhisk/</a></td>
<td>Node.js, Swift, Python, Java</td>
<td>Go/C (native), Python/Java</td>
</tr>
<tr>
<td>Apache OpenWhisk</td>
<td>OSS</td>
<td><a href="https://github.com/apache/incubator-openwhisk">https://github.com/apache/incubator-openwhisk</a></td>
<td>Kubernetes, Standalone, Docker</td>
<td>Node.js, Python, Java, Python, JavaScript</td>
</tr>
<tr>
<td>fission.io</td>
<td>OSS</td>
<td><a href="https://github.com/fission/fission">https://github.com/fission/fission</a></td>
<td>Kubernetes</td>
<td>Node.js, Python, Java, Python, JavaScript</td>
</tr>
<tr>
<td>OpenFaaS</td>
<td>OSS</td>
<td><a href="https://github.com/openfaas/faas">https://github.com/openfaas/faas</a></td>
<td>Docker Swarm, Kubernetes, any other</td>
<td>Python, Ruby, C#, Node.js, Go, Rust</td>
</tr>
<tr>
<td>Iron.io functions</td>
<td>OSS</td>
<td><a href="https://github.com/iron-io/functions">https://github.com/iron-io/functions</a></td>
<td>Any that supports Docker images</td>
<td>Any language, Docker images</td>
</tr>
<tr>
<td>kubeless</td>
<td>OSS</td>
<td><a href="https://github.com/kubeless/kubeless">https://github.com/kubeless/kubeless</a></td>
<td>Kubernetes</td>
<td>Python, Node.js, Ruby</td>
</tr>
<tr>
<td>microue</td>
<td>OSS</td>
<td><a href="https://github.com/Stackvana/microue">https://github.com/Stackvana/microue</a></td>
<td>Any Node.js HTTP middleware</td>
<td>Over 20 languages</td>
</tr>
<tr>
<td>Nuclio (by iguazio)</td>
<td>OSS</td>
<td><a href="https://github.com/nuclio/nuclio">https://github.com/nuclio/nuclio</a></td>
<td>Docker, Kubernetes, Single binary</td>
<td>Go/C (native), Python/Java</td>
</tr>
</tbody>
</table>

### Diagram

- Apache OpenWhisk
- fission
- OPENFAAS
- IronFunctions
- kubeless
- microue
- Nuclio

![CNCF Serverless WG: Landscape Diagram](https://example.com/landscape_diagram.png)
CNCF Serverless WG: Recommendations

- Maintain the landscape of Serverless implementations and features
- Produce additional documents and samples that educate community
- Document integration with other CNCF projects, such as how to monitor and observe
- Potential collaboration / harmonization on:
  - Event format
  - Function definition / packaging & deployment / workflow

Developer Interop
Proposed and got agreement from CNCF TOC to work on Events

Creating a common format for events
  - Useful across entire Cloud Native deployments, beyond just Serverless

Considering a few proposals as a starting point:
  - OpenEvents – (Serverless, Inc.)
  - Cloud-Native-Event-Mapping (CNEM) – (iguazio)
  - Cloud Auditing Data Federation (CADE) – (DMTF, IBM)

CloudEvents was born
  - https://cloudevents.io
CloudEvents Project

- Define the common metadata of an Event

```
{
    "cloudEventsVersion" : "0.1",
    "eventType" : "com.example.someevent",
    "eventTypeVersion" : "1.0",
    "source" : "/mycontext",
    "eventID" : "A234-1234-1234",
    "eventTime" : "2018-04-05T17:31:00Z",
    "contentType" : "text/xml",
    "data" : "<much wow="/"xml"/>"
}
```

It's not about data.
It's about metadata!
CloudEvents Use Cases

• Normalize events, web-hooks, across environments

• Facilitate integrations across platforms
  – Leave the event business logic processing to the application

• First step towards portability of functions
• **CloudEvents Specification** – define the metadata

• **Serialization Rules Specifications**
  – JSON event format
  – AMQP event format

• **Transport Bindings Specifications**
  – HTTP – binary and structured
  – MQTT
  – AMQP
  – NATS
  – Web-hooks

• **Primer**
CloudEvents Interop Demo – KubeCon EU

Publishers
- Amazon S3
- Azure Blob Storage

Middleware
- Serverless Event Gateway
- Azure Event Grid

Subscribers
- Google Cloud Function
- IBM Cloud Function
- Azure Function
- SAP Kubeless Function
- Nuclio Function
- VMWare Dispatch Function
- Oracle fn Function
- Huawei Function
- RedHat OpenWhisk
  + OpenShift Function
- AWS Lambda Function
- Alibaba Cloud Function

Log
- Twitter

https://youtu.be/TZPPiAy12K
CloudEvents Interop Demo – KubeCon EU

https://twitter.com/CloudEventsDemo/lists/demo
CloudEvents Status

- Released v0.1 in April 2018
- Multiple implementations planned
  - Kudos to Microsoft for already supporting it in Event Grid
- Approved as a CNCF Sandbox Project!
- Looking at next workstream... Function Workflow Definition
  - Chaining, orchestrating functions
- Considering others too...
Additional Information

- **CNCF Serverless Working Group**
  - [https://github.com/cncf/wg-serverless](https://github.com/cncf/wg-serverless)
  - Weekly calls on Thursdays at 8am PT  Come join in!

- **CNCF Serverless Working Group White Paper**
  - [https://docs.google.com/document/d/1UiW8bt5O8QBqQRIlJVKZJeilNnxl20AJu9wA8wcdI](https://docs.google.com/document/d/1UiW8bt5O8QBqQRIlJVKZJeilNnxl20AJu9wA8wcdI)

- **CNCF Serverless Landscape**
  - [https://docs.google.com/spreadsheets/d/10rSQ8rMHyDgf_ib3n6kfwFuoE88gr0amUPRxKbwVCk](https://docs.google.com/spreadsheets/d/10rSQ8rMHyDgf_ib3n6kfwFuoE88gr0amUPRxKbwVCk)

- **CloudEvents**
  - [https://github.com/cloudevents/spec](https://github.com/cloudevents/spec)
  - [http://cloudevents.io](http://cloudevents.io)
Thank You

Doug Davis
STSM, IBM (dug@us.ibm.com)
duglin @duginabox