Serverless
Where Have We Come?
Where Are We Going?

Stephen Fink
Distinguished Engineer
IBM Watson

@sjfink
VISIBILITY

- Peak of Inflated Expectations
- Plateau of Productivity
- Slope of Enlightenment
- Trough of Disillusionment
- Technology Trigger

TIME
You are here
You are here?

Serverless 2018?

- Peak of Inflated Expectations
- Plateau of Productivity
- Slope of Enlightenment
- Trough of Disillusionment

TIME

VISIBILITY

Technology Trigger
Are you ready?
About your speaker

2014-2017  IBM’s OpenWhisk/Cloud Functions
About your speaker

2014-2017 IBM’s OpenWhisk/Cloud Functions
Serverless Advocate
Gave talks like ....

The Serverless Revolution

Stephen Fink
Chief Architect
IBM Watson Programming Models

@sjfink
With slides like …

Serverless

“Making Development Fun Again”
About your speaker

2014-2017  IBM’s OpenWhisk/Cloud Functions
2017-2018  Architect for IBM Watson
large development organization invested in microservices typical business pressure
Serverless for Watson
Hype and Reality
Write functions. Not plumbing.

Platform handles all infrastructure transparently

No servers

Just code
conventional microservice architecture

Developer Responsibilities

- business logic
- service integrations
  - IaaS: programmatic
  - PaaS: declarative
- containers
- middleware
- autoscaling
- load balancing
- service discovery
- fault tolerance
- logging
- messaging
- security patches

serverless architecture

Developer Responsibilities

- business logic
- service integrations (declarative)
- REST API definition (declarative)

Platform Responsibilities

- containers
- middleware
- autoscaling
- load balancing
- service discovery
- fault tolerance
- logging
- messaging
- security patches
build a robust, fault-tolerant, scalable microservice in minutes
Ship it!
Ship it!
**Hype:** Just deploy your code. The system handles logging and monitoring automatically!
**Reality**: All Watson systems already have extensive management, logging, DevOps, and monitoring. Serverless doesn’t help integrate into existing production environment.
**Hype:** Just deploy your code. Specify an API declaratively. API Gateway does the rest!
*Reality:* Conventional web frameworks have tools, libraries, ecosystems tuned precisely for this problem. A serverless REST endpoint doesn't save that much code.
**Hype**: specify your service integrations declaratively. Wrap services as functions, compose at will, and go!
Reality: Service integrations have many flavors and options. Code and SDKs is still the easiest way to invoke Watson in realistic use cases.

```javascript
document.querySelector('#button').onclick = function () {
  fetch('/api/speech-to-text/token')
    .then(function (response) {
      return response.text();
    }).then(function (token) {

      var stream = WatsonSpeech.SpeechToText.recognizeMicrophone({
        token: token,
        objectMode: true, // send objects instead of text
        format: false // optional - performs basic formatting on the results such as capitals an periods
      });

      stream.on('data', function (data) {
        console.log(data);
      });

      stream.on('error', function (err) {
        console.log(err);
      });

      document.querySelector('#stop').onclick = stream.stop.bind(stream);
    }).catch(function (error) {
      console.log(error);
    });
};
```
A Mixed-Method Empirical Study of Function-as-a-Service Software Development in Industrial Practice

Philipp Leitner\textsuperscript{a}, Erik Witters\textsuperscript{b}, Josef Spillner\textsuperscript{b}, Waldemar Hummer\textsuperscript{b}

\textsuperscript{a}Software Engineering Division, Chalmers / University of Gothenburg, Sweden
\textsuperscript{b}IBM Research, Yorktown Heights, New York, USA
\textsuperscript{c}Service Prototyping Lab, Zurich University of Applied Sciences, Switzerland

<table>
<thead>
<tr>
<th>Challenges</th>
<th>Percentage</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lack of tooling (e.g., testing, deployment)</td>
<td>51 / 55%</td>
</tr>
<tr>
<td>Integration testing</td>
<td>37 / 40%</td>
</tr>
<tr>
<td>Vendor lock-in</td>
<td>30 / 32%</td>
</tr>
<tr>
<td>Container start-up latency</td>
<td>27 / 29%</td>
</tr>
<tr>
<td>Managing state in functions</td>
<td>25 / 27%</td>
</tr>
<tr>
<td>Unit testing</td>
<td>17 / 18%</td>
</tr>
<tr>
<td>Little support for reusing functions</td>
<td>13 / 14%</td>
</tr>
<tr>
<td>Lack of documentation</td>
<td>12 / 13%</td>
</tr>
<tr>
<td>Finding/hiring developers familiar with FaaS</td>
<td>11 / 12%</td>
</tr>
<tr>
<td>Little support for composition of functions</td>
<td>11 / 12%</td>
</tr>
<tr>
<td>CPU or processing limitations</td>
<td>8 / 9%</td>
</tr>
<tr>
<td>Memory limitation</td>
<td>5 / 5%</td>
</tr>
<tr>
<td>Other</td>
<td>3 / 3%</td>
</tr>
</tbody>
</table>
Initial Prototype

Move to Production
Software as a service (SaaS /sæs/)\(^1\) is a software licensing and delivery model in which software is licensed on a subscription basis and is centrally hosted.\(^2\)[3] It is sometimes
You need to add custom logic to a hosted SaaS chatbot.
SaaS tooling with serverless extension points.

Serverless: **perfect match** for extension points in hosted solutions.
Write functions.
Not plumbing.

Hype or Reality?
Write functions.
Not plumbing.

Hype or Reality?

Initial Prototype
Write functions. Not plumbing.

Hype or Reality?

Initial Prototype

Integration with Production Systems
Write functions. Not plumbing.

Hype or Reality?

Initial Prototype

Integration with Production Systems

SaaS Extension Points
Event-driven Programming

Functions run in response to events
WHEN a document is uploaded, inject the document into the knowledge store

WHEN the logs grow to certain size gather the logs, perform ETL, retrain a new model

WHEN a new model is available evaluate the accuracy

EVERY 15 minutes poll a stream for new data to inject
Event-driven programming in Watson AI

**Data**

- Private data
  - <HTML>
  - WORD
  - JSON
  - PDF

**Ingestion**

- Convert and enrich by leveraging Watson APIs to add NLP meta data to your content, making it easier to explore and discover insights.
- Clean and normalize through an automated processing of NLP results, improving data quality.

**Storage**

- Normalized data is indexed into a collection as part of your environment in the cloud.

**Query**

- Understand data faster, create better hypothesis and deliver better outcomes.

**Output**

- Actionable insights into your app.
WHEN a document is uploaded, ingest the document into the knowledge store
WHEN a document is uploaded, inject the document into the knowledge store.

WHEN the logs grow to certain size gather the logs, perform ETL, retrain a new model.
WHEN a document is uploaded, inject the document into the knowledge store

WHEN the logs grow to certain size gather the logs, perform ETL, retrain a new model

WHEN a new model is available evaluate the accuracy
WHEN a document is uploaded, ingest the document into the knowledge store

WHEN the logs grow to certain size gather the logs, perform ETL, retrain a new model

WHEN a new model is available evaluate the accuracy

EVERY 15 minutes poll a stream for new data to inject
Unlock hidden value in data to find answers, monitor trends and surface patterns, with the world's most advanced cloud-native insight engine.

Quickly find insights in the Watson Discovery News data collection of recent news articles. Easily explore a company's:

- Top stories over the last two months
- Top entities (people, topics, companies) mentioned in those articles
- Trend of public sentiment in news
- Anomalous periods of high press coverage
- Trend of most commonly paired entities (co-mentions)

Watson Discovery also lets you do the same analysis with your own data. Learn more [here](#).
Event-driven Programming

Functions run in response to events

Hype or Reality?
Event-driven Programming

Functions run in response to events

Hype or Reality?

+ not latency sensitive
+ highly elastic

★ ★ ★ ★ ★
Scale instantly.
No provisioning.

Enlist more resources automatically based on offered load.
Fine-grain pricing.

Pay only for the exact time your actions run, metered on the order of milliseconds.
### Do you think that using FaaS at the moment is cheap in terms of cloud hosting costs?

<table>
<thead>
<tr>
<th>Option</th>
<th>Percentage</th>
</tr>
</thead>
<tbody>
<tr>
<td>1: Total costs of FaaS are lower than its alternatives</td>
<td>65 / 71%</td>
</tr>
<tr>
<td>2: Costs do not matter to us at this point</td>
<td>20 / 22%</td>
</tr>
<tr>
<td>3: Total costs of FaaS are higher than its alternatives</td>
<td>3 / 3%</td>
</tr>
<tr>
<td>4: Other</td>
<td>3 / 3%</td>
</tr>
</tbody>
</table>
Total Cost of Solution (unscientific estimates)

Development
Operations
Support

Storage
Networking
Compute

Compute Cycles
Everything Else
Scale instantly. No provisioning. Fine-grain pricing.

Hype or Reality?
Scale instantly. No provisioning. Fine-grain pricing.

Hype or Reality?

Startup or hobby-ist
Scale instantly. No provisioning. Fine-grain pricing.

**Hype or Reality?**

- Startup or hobby-ist: ★★★★★
- Large Enterprise: ★★★★☆☆ often irrelevant
Cloud: an evolutionary story

- Increasing focus on business logic
- Decreasing concern (and control) over stack implementation

- Bare Metal
- Virtual machines
- Containers
- Functions

© 2018 IBM Corporation
Containers

+ Fine-Grain Metering
+ Faster Autoscaling
+ Event-driven Programming

Functions
Containers

- Tools
- Control and Flexibility
- De Facto Standards

Functions
Infrastructure Convergence

The End of the Road for Serverless?
Serverless
Innovation in Cloud infrastructure
Programming Model
Composing IBM Cloud Functions

https://ibm.biz/serverless-research

Kerry Chang, Olivier Tardieu
IBM Research
try {
    let zipCode = getZipCode(location);
    return getWeather(zipCode);
} catch(err) {
    return { message: `Unable to retrieve weather info: ${err}` };}

Does not work as a serverless app
• Time limit, double billing
try {
    let zipCode = getZipCode(location);
    return getWeather(zipCode);
} catch(err) {
    return { message: `Unable to retrieve weather info: ${err}` };
- if/else
- try/catch
- data forwarding
- inline functions for simple operations
Event-driven Programming → Serverless Event-Driven Workflow
In Conclusion
In Conclusion

1. Integration: You don’t own `main()`
2. Embrace containers - infrastructure convergence is coming
3. Innovate on event-driven programming model
Q&A