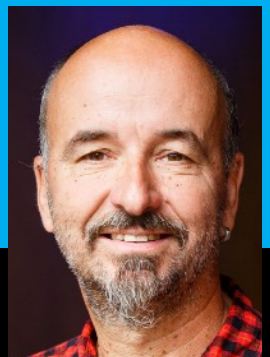


adaptTo()

APACHE SLING & FRIENDS TECH MEETUP

2 - 4 SEPTEMBER 2019

Sling & Serverless - Best Friends Forever?

Bertrand Delacrétaz - Principal Scientist, Adobe

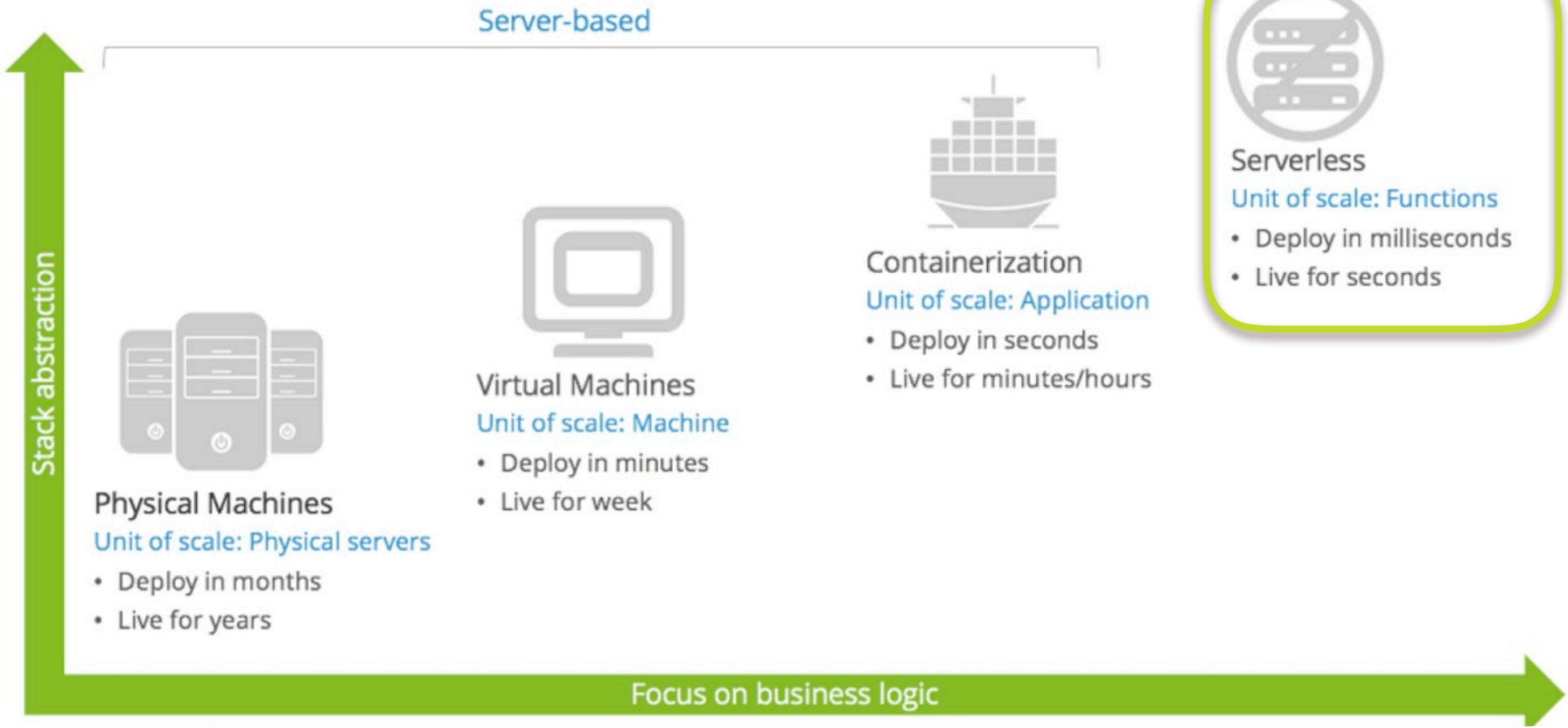
Adobe

Images: stock.adobe.com, unless otherwise specified

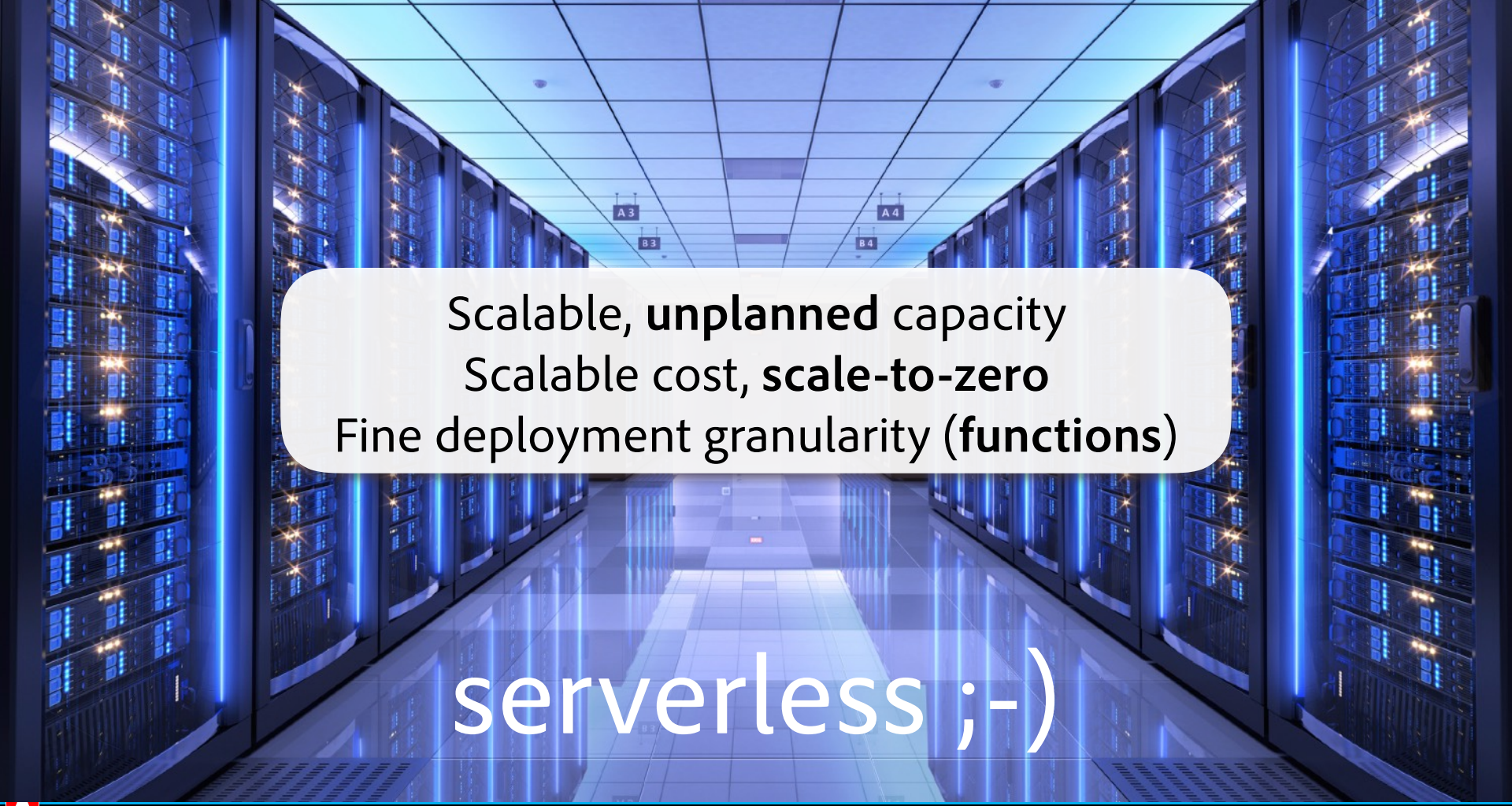
slides revision: 2019-09-03

Serverless?

Cloud Technologies Evolution



Source: Deloitte Consulting LLP, via Alexander Klimetschek, @alexkli


Scalable, **unplanned** capacity
Scalable cost, **scale-to-zero**
Fine deployment granularity (**functions**)
serverless ;-)

Hello, Serverless World!

```
// The actual OpenWhisk action code
function main(params) {
  const name = params.name || 'World';

  const content = `
    <html>
      <body>
        <h1>Hello, ${escapeForHTML(name)}!</h1>
      </body>
    </html>
    `;

  console.log(content);
  return {body: content };
}
```

Installation:

```
wsk action update web-hello web-hello.js --web true
```

URL:

```
wsk -i action get web-hello --url
```

Units of Deployment

Serverless: Containers or Functions ?

Scalable, **unplanned** capacity + Scalable cost, **scale-to-zero**


```
// The actual OpenWhisk action code
function main(params) {
  const name = params.name || 'World';

  const content = `
    <html>
      <body>
        <h1>Hello, ${escapeForHTML(name)}!</h1>
      </body>
    </html>
    `;

  console.log(content);
  return {body: content };
}
```

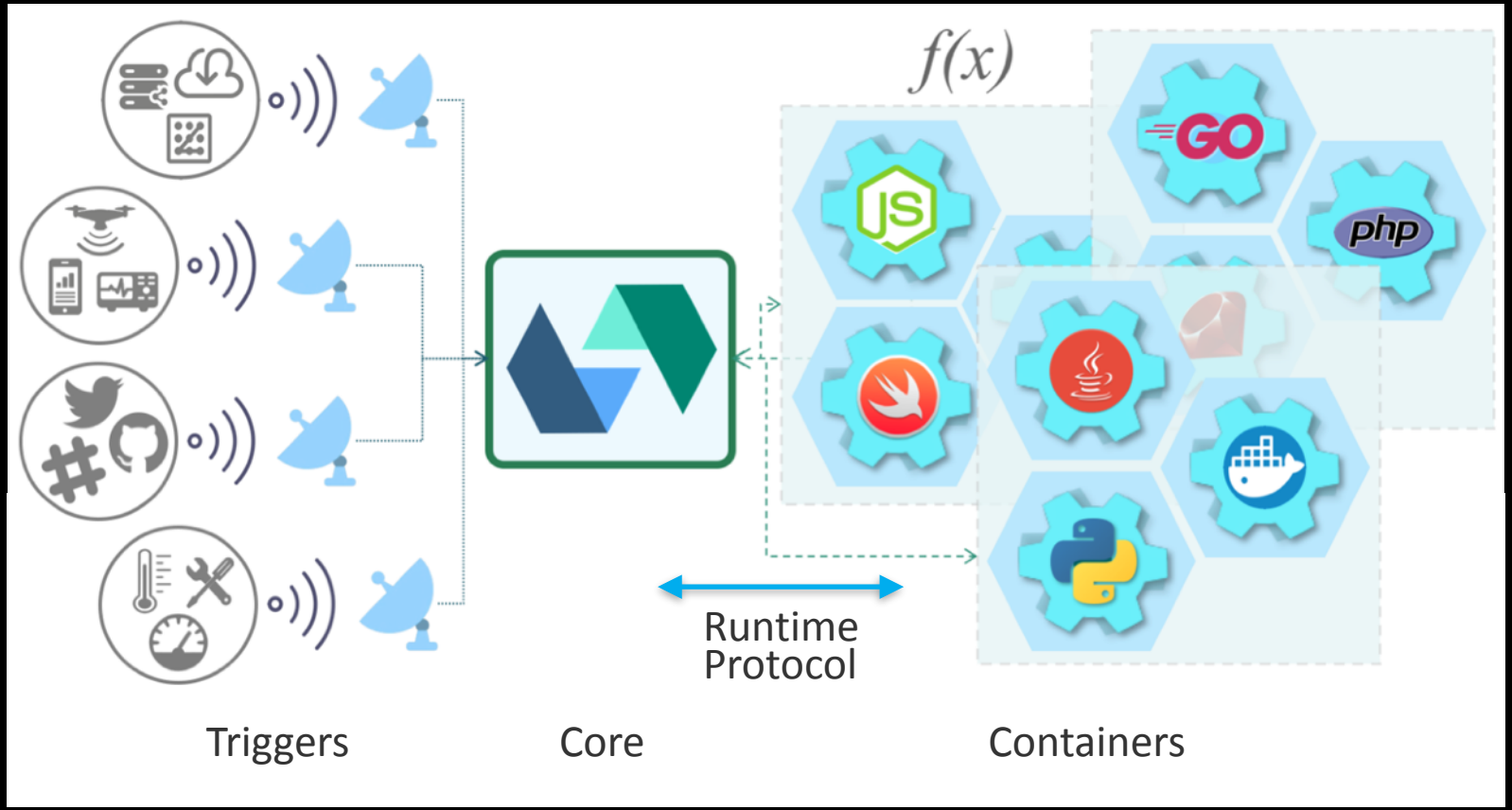
Fine deployment granularity ?

Needs fast startup

OpenWhisk: simple init/run HTTP API

Limited choice of languages?

OpenWhisk: Orchestrated Containers



OpenWhisk runtime protocol

The /init end point

After starting the Docker container, the invoker will POST to `/init` the following payload:

```
{
    "value": {
        "name" : "helloPHP",
        "main" : "main",
        "binary": false,
        "code" : "<?php …",
    }
}
```

The /run end point

To run the action, the invoker will POST to `/run`. This may happen multiple times, but never concurrently. The payload contains the arguments to be passed to the function to be executed. For example:

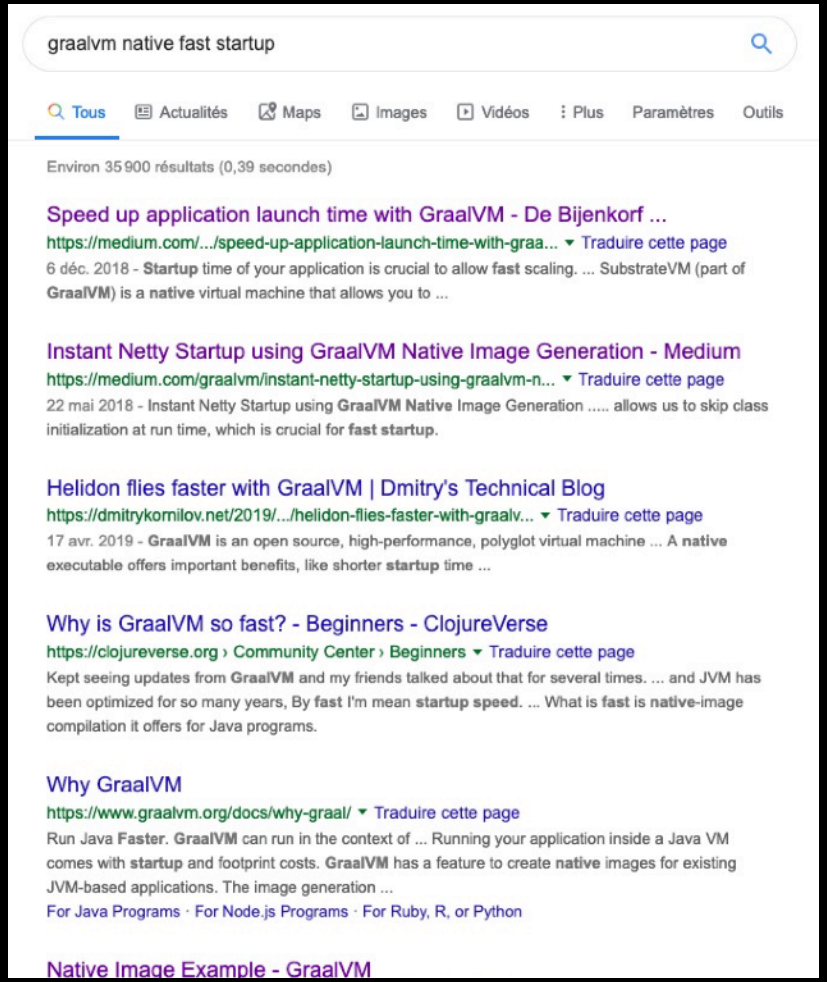
```
{
    "value":

    {
        "name" : "Rob",
    }
}
```

Runtime container responds to **/init** and **/run** HTTP requests
-> easy to create **any** runtime

GraalVM - native Java

(in containers?)



Impressive startup speed...assuming you pass the native-image tool hurdle!

SubstrateVM limitations

What	Support Status
Dynamic Class Loading / Unloading	Not supported
Reflection	Supported (Requires Configuration)
Dynamic Proxy	Supported (Requires Configuration)
Java Native Interface (JNI)	Mostly supported
Unsafe Memory Access	Mostly supported
Class Initializers	Supported
InvokeDynamic Bytecode and Method Handles	Not supported
Lambda Expressions	Supported
Synchronized, wait, and notify	Supported
Finalizers	Not supported
References	Mostly supported
Threads	Supported
Identity Hash Code	Supported
Security Manager	Not supported
JVMTI, JMX, other native VM interfaces	Not supported
JCA Security Services	Supported

https://github.com/oracle/graal/blob/master/substratevm/LIMITATIONS.md

Sling code -> native code experiments

```java
/** This is where we wire the system, like the OSGi framework
 *  would do. As it seems hard to run that framework in a GraalVM
 *  environment for now, we wire things statically.
 */
private static OsgiContext initialize() {
    final OsgiContext result = new OsgiContext();

    // This would be automatic in a JUnit environment
    result.registerInjectActivateService(new MockEventAdmin());

    // Our minimal resource provider
    final MockResourceProvider mrp = new MockResourceProvider();
    result.registerInjectActivateService(mrp);

    // SlingRequestProcessor
    result.registerInjectActivateService(new SlingRequestProcessorWrapper(result.bundleContext()));

    // ResourceResolver
    //result.registerInjectActivateService(new MockResourceResolver(mrp));
    result.registerInjectActivateService(new MockServiceUserMapper());
    result.registerInjectActivateService(new ResourceAccessSecurityTracker());
    final ResourceResolverFactoryActivator rrfa = new ResourceResolverFactoryActivator();
    result.registerInjectActivateService(rrfa);
    result.registerInjectActivateService(new ResourceResolverFactoryService(rrfa));

    return result;
}
```

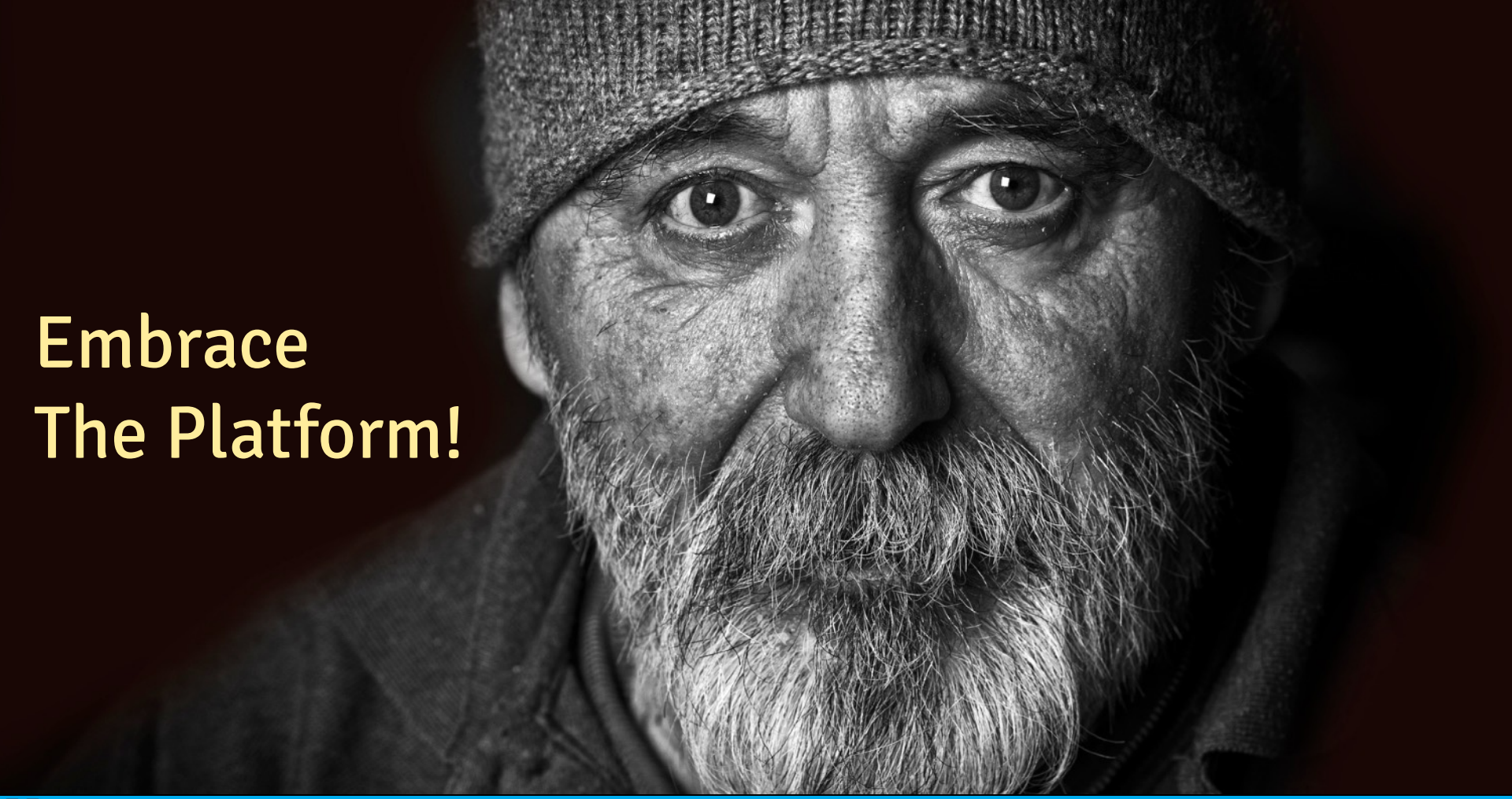
Static wiring with Sling OSGi mocks

Static wiring worked ~~well~~ ok (with lots of suffering) until hitting what's probably a native-image bug exposed by our legacy code.

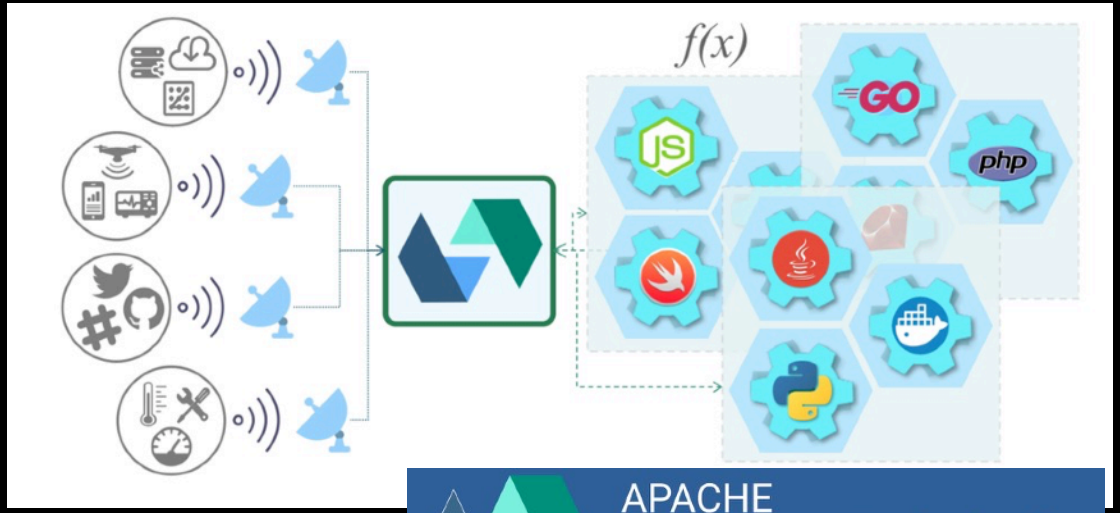
See SLING-8556 and sling-whiteboard/graalvm - and with more effort or once the tools mature that might work.


I GIVE UP!

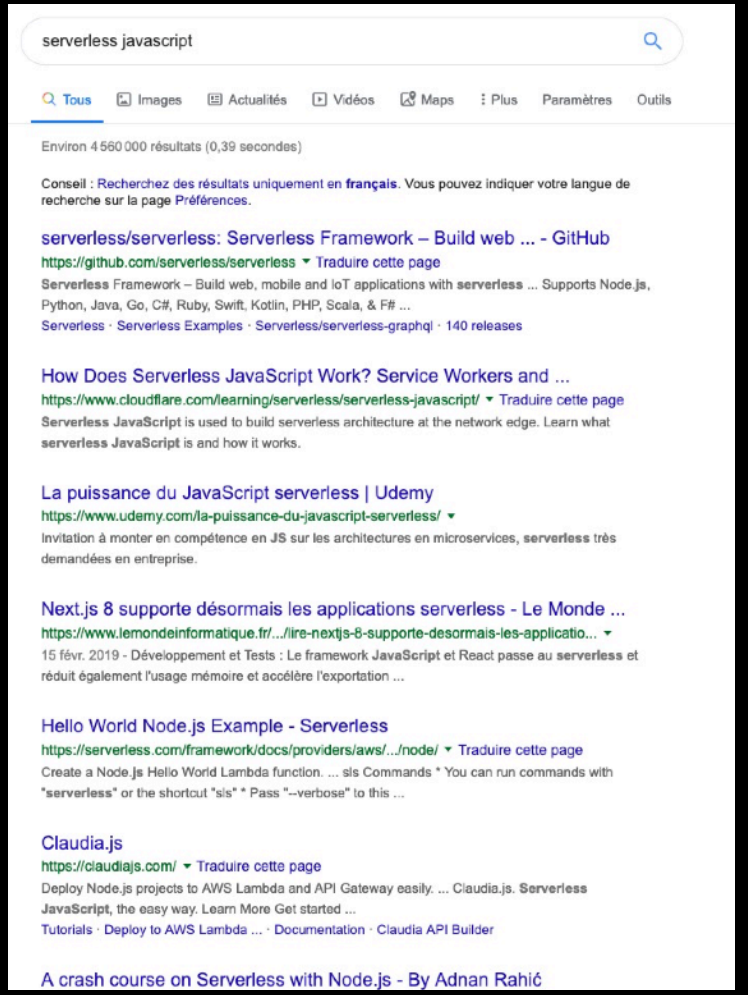
Embracing the Platform?


Embrace
The Platform!

serverless == JavaScript?



Are we doing Google Search Driven Architecture now?. Hmm...



OpenWhisk Action Annotations

Annotations enable dynamic selection of functions, like we do for OSGi services in Sling

```
/**
 * A SlingSafeMethodsServlet that renders the current Resource as simple HTML
 */
@Component(service = Servlet.class,
    name="org.apache.sling.servlets.get.DefaultGetServlet",
    property = {
            "service.description=Default GET Servlet",
            "service.vendor=The Apache Software Foundation",

            // Use this as a default servlet for Sling
            "sling.servlet.resourceTypes=sling/servlet/default",
            "sling.servlet.prefix:Integer=-1",

            // Generic handler for all get requests
            "sling.servlet.methods=GET",
            "sling.servlet.methods=HEAD"
    })
```

OSGI service properties

```
{
  "namespace": "guest",
  "name": "somedoc-html",
  "version": "0.0.1",
  "exec": {
    "kind": "nodejs:10",
    "binary": false
  },
  "annotations": [
    {
      "key": "sling:contentType",
      "value": "text/html"
    },
    {
      "key": "sling:resourceType",
      "value": "microsling/somedoc"
    },
    {
      "key": "provide-api-key",
      "value": false
    },
    {
      "key": "sling:extensions",
      "value": "html"
    },
```

OpenWhisk Annotations

Serverless µSling!

2007...

Sling / SLING-47

microsling - simple webapp to demonstrate the core principles of Sling

CLOSED

Details

Type:	New Feature	Status:	Fixed
Priority:	Minor	Resolution:	None
Affects Version/s:	None	Fix Version/s:	
Component/s:	Engine		
Labels:	None		

Description

Following our recent API redesign discussions (see http://cwiki.apache.org/confluence/display/SLING/Sling+API+Redesign in particular), I have started working on "microsling", a webapp that demonstrates my understanding of the "most important parts" of Sling.

bertrand's brain grep

cat /dev/brain | egrep -i 'tech|thoughts|noise'

Home | Who?

microsling - Yet Another (cool) Web Applications Framework

October 12, 2007

From the *new and improved* department: I spent art of this week writing a "reduced to the max" rsion of the Sling core that I've called *microsling*.

's been a lot of fun of course, and I think the s are fairly impressive in terms of power per line of code, thanks to wer of the JCR API. Using "modern" Java, with scripting in the right helps a lot as well, but that's nothing new.

ent state, microsling allows you to use SlingServlets to process ests in a RESTful way, acting on abstract Resources and using request processing scripts in various languages.

lingServlets are provided to create content and to render it locity templates or server-side javascript.

an 1'200 lines of Java code, all inclusive.

microsling matches the complete vision of the Sling but this is already powerful stuff – and very simple to

sling-logo.png

You have searched the bertrand's brain grep blog archives for 'microsling'. If you are unable to find anything in these search results, you can try one of these links.

CAT MYSELF | SORT | HEAD -2

Bertrand Delacrétaz here – my "résumé" will tell you more.

The opinions expressed here are my own, I'm not representing any group or company on this blog. YMMV.

FOLLOW ME ON TWITTER

microsling Search

LS /VAR/CATEGORIES

Select Category

Actions selected based on Annotations

INPUT: resource type, request extension

FOR all OpenWhisk Actions A in this namespace,
ORDERED by preference:
  GET the annotations of action A
  IF annotations match
  THEN return A as the Action to use

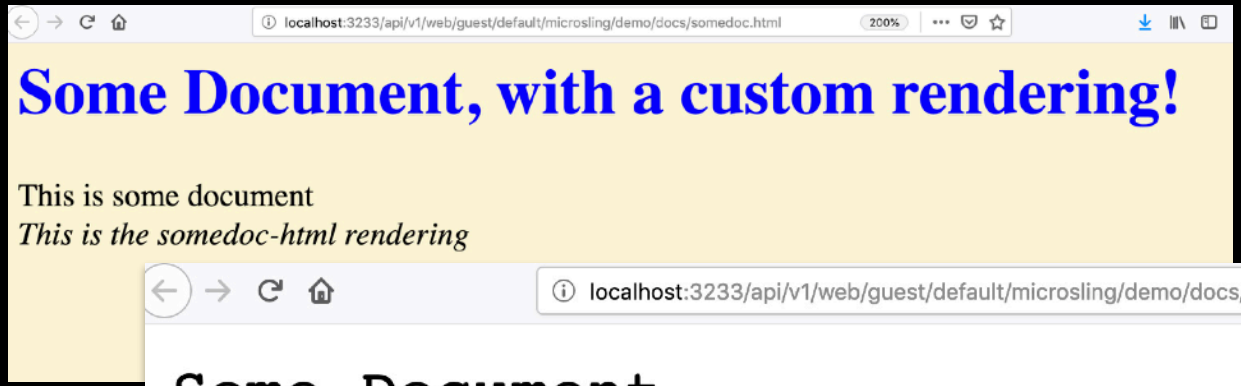
IF no Action found, use default renderers

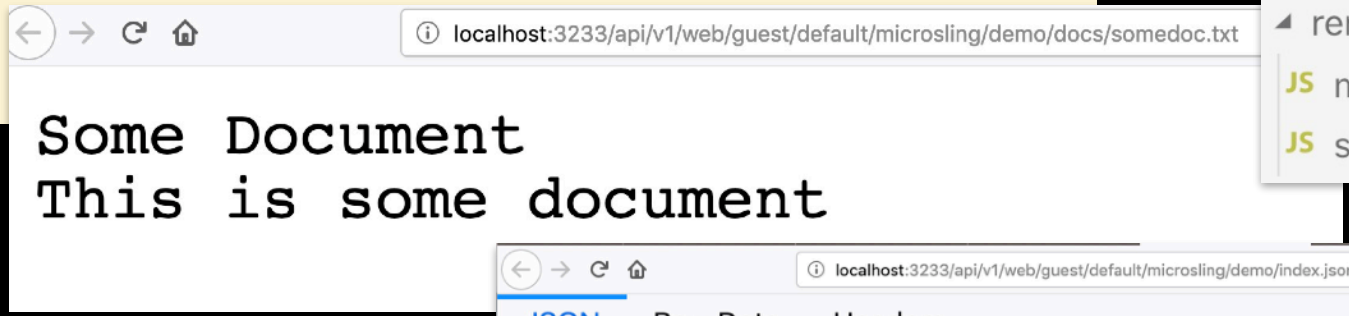
sling-whiteboard: serverless-microsling/lib/openwhisk-renderer.js

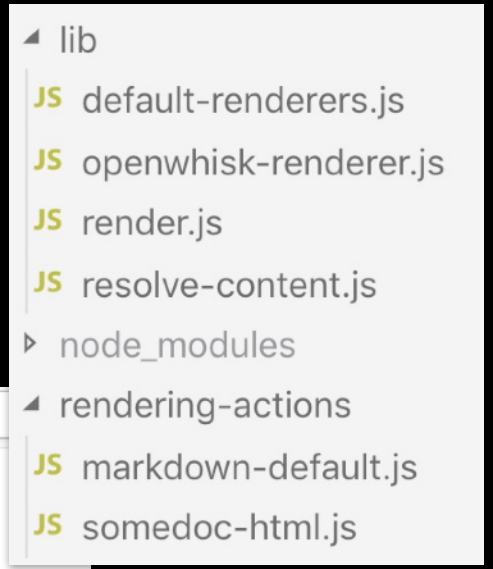

```
{
  "namespace": "guest",
  "name": "somedoc-html",
  "version": "0.0.1",
  "exec": {
    "kind": "nodejs:10",
    "binary": false
  },
  "annotations": [
    {
      "key": "sling:contentType",
      "value": "text/html"
    },
    {
      "key": "sling:resourceType",
      "value": "microsling/somedoc"
    },
    {
      "key": "provide-api-key",
      "value": false
    },
    {
      "key": "sling:extensions",
      "value": "html"
    },
```

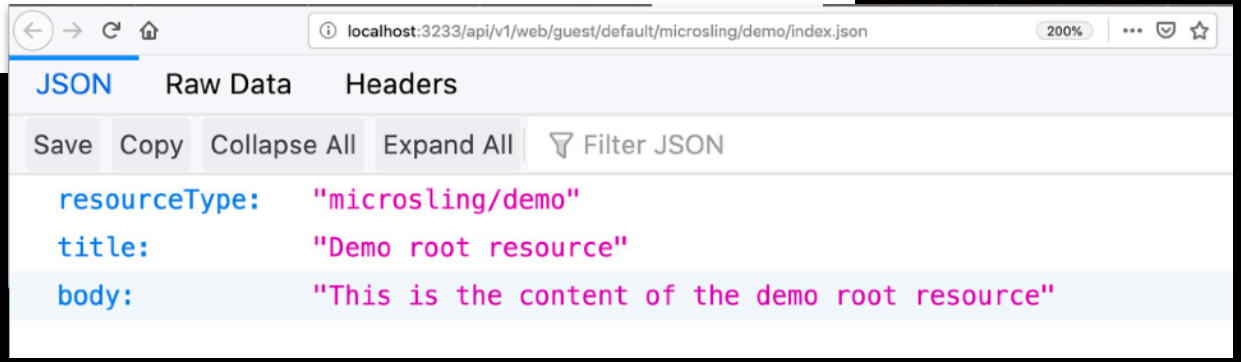

OpenWhisk Annotations

DEMO: Serverless Microsling









Dynamic selection

Overrides

New content types

CODA

CODA

GraalVM Native Images are difficult with existing code.

Embracing the platform usually helps - worked for our prototype.

Action annotations can be used in a similar way to OSGi service properties.

Next?

a) build on Karl & Radu's experiments for fast-starting Sling instances -> switch to containers -> "full" Sling?

b) build on the serverless microsling prototype to create rendering pipelines -> scalable rendering?

I'm @bdelacretaz - thank you!