

Functions as a Service

- ForwardJS

IBM Code

Upkar Lidder
Developer Advocate, IBM

> ulidder@us.ibm.com
> @lidderupk
> blog.upkarlidder.com

What is Serverless?

What is Serverless?

Serverless computing refers to the concept of building and running applications that **do not require server management**.

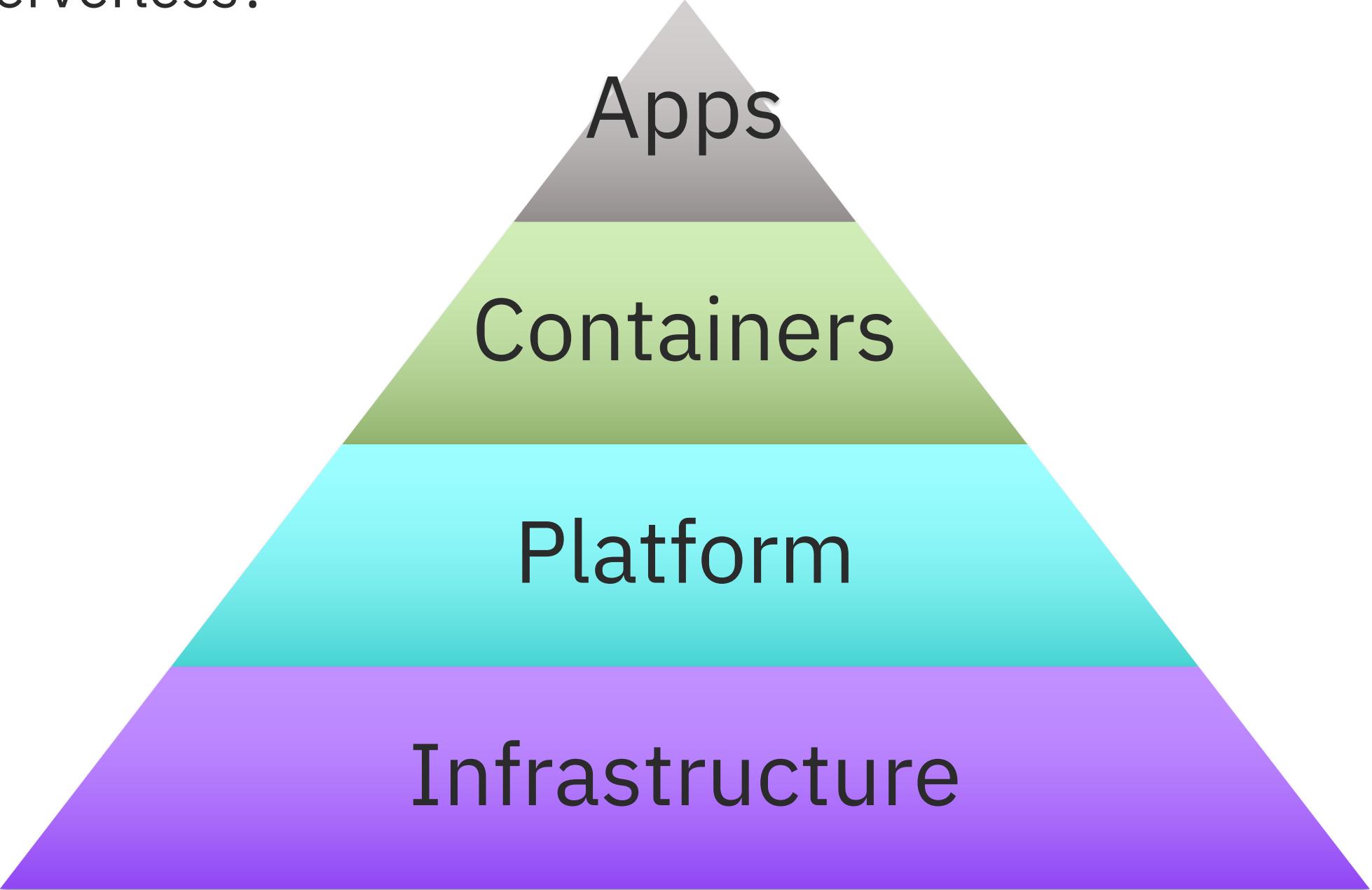
It describes a finer-grained deployment model where applications, **bundled as one or more functions**, are uploaded to a platform and then **executed, scaled, and billed** in response to the **exact demand needed** at the moment.

It refers to the idea that consumers of serverless computing **no longer need to spend time and resources on server provisioning, maintenance, updates, scaling, and capacity planning**. Instead, all of these tasks and capabilities are handled by a serverless platform and are completely abstracted away from the developers

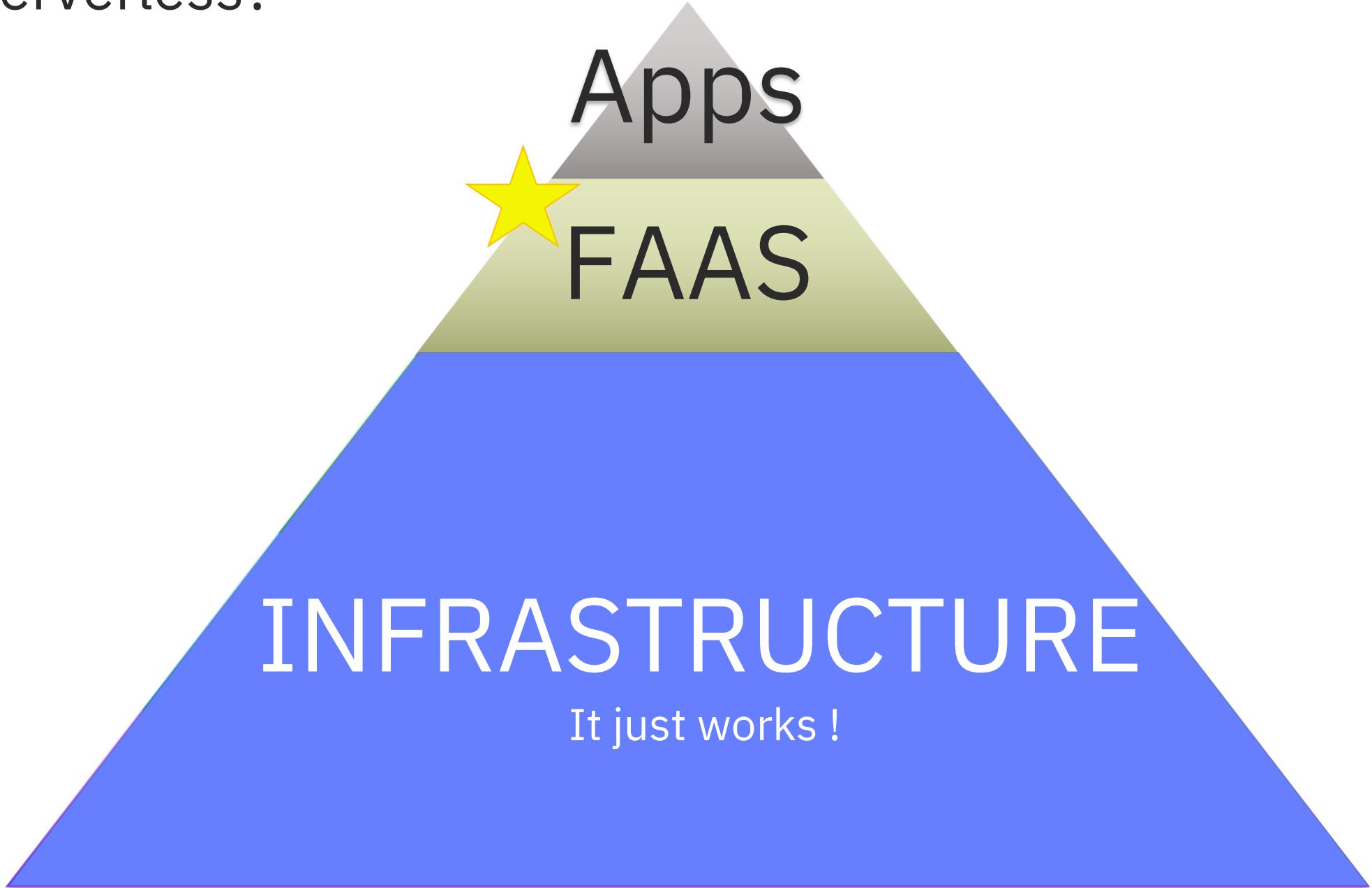
- **Cloud Native Computing Foundation**

<https://github.com/cncf/wg-serverless/tree/master/whitepapers/serverless-overview>

What is Serverless?



What is Serverless?



Use cases

Leans well to ...

Small, focused, asynchronous, concurrent, easy to parallelize into independent units of work

Infrequent or has sporadic demand, with large, unpredictable variance in scaling requirements

Stateless, ephemeral, without a major need for instantaneous cold start time

Highly dynamic in terms of changing business requirements that drive a need for accelerated developer velocity

Use cases

Existing use cases ...

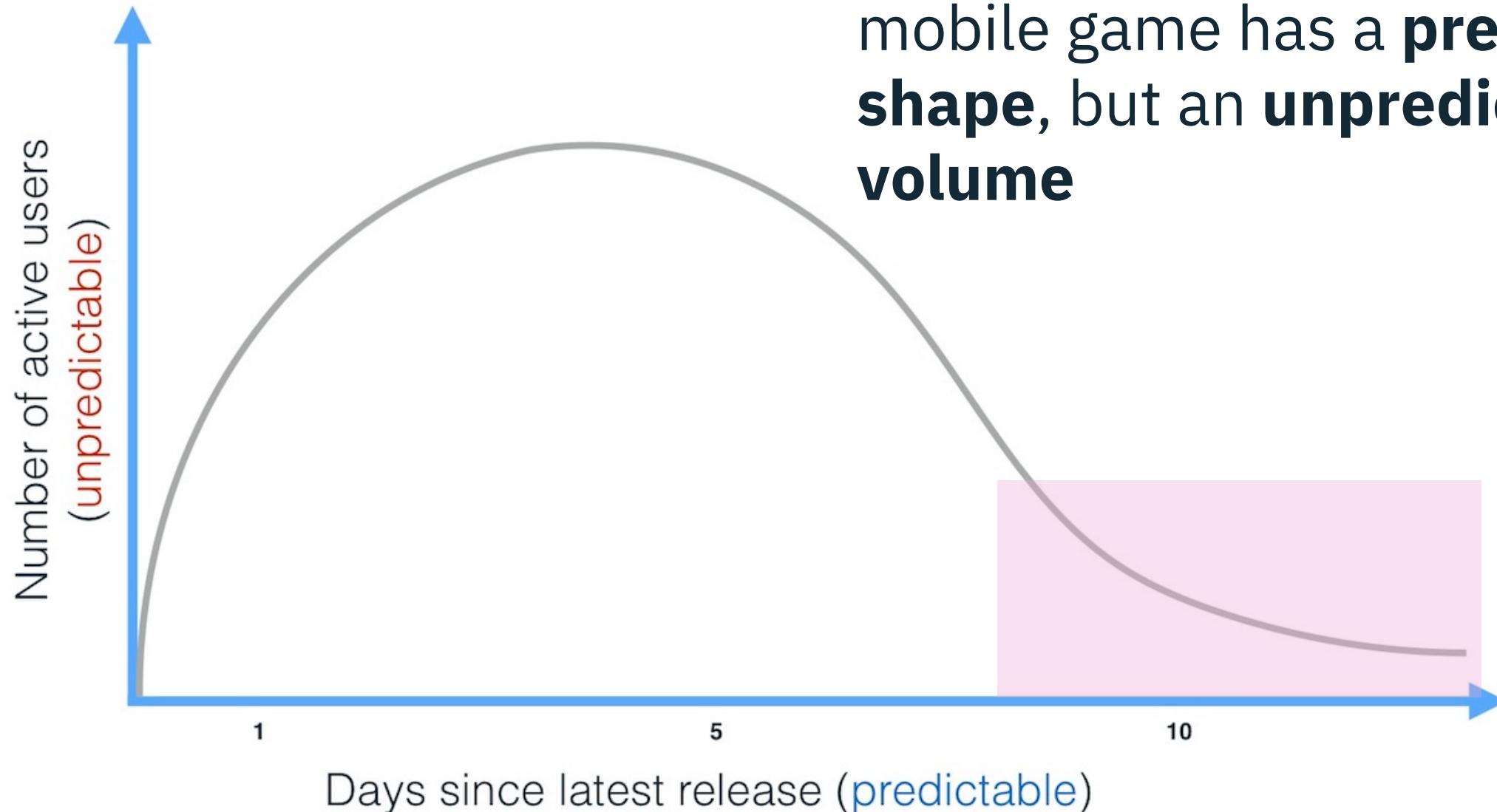
Secure and scalable APIs

Backend services / features

Event Driven Programming

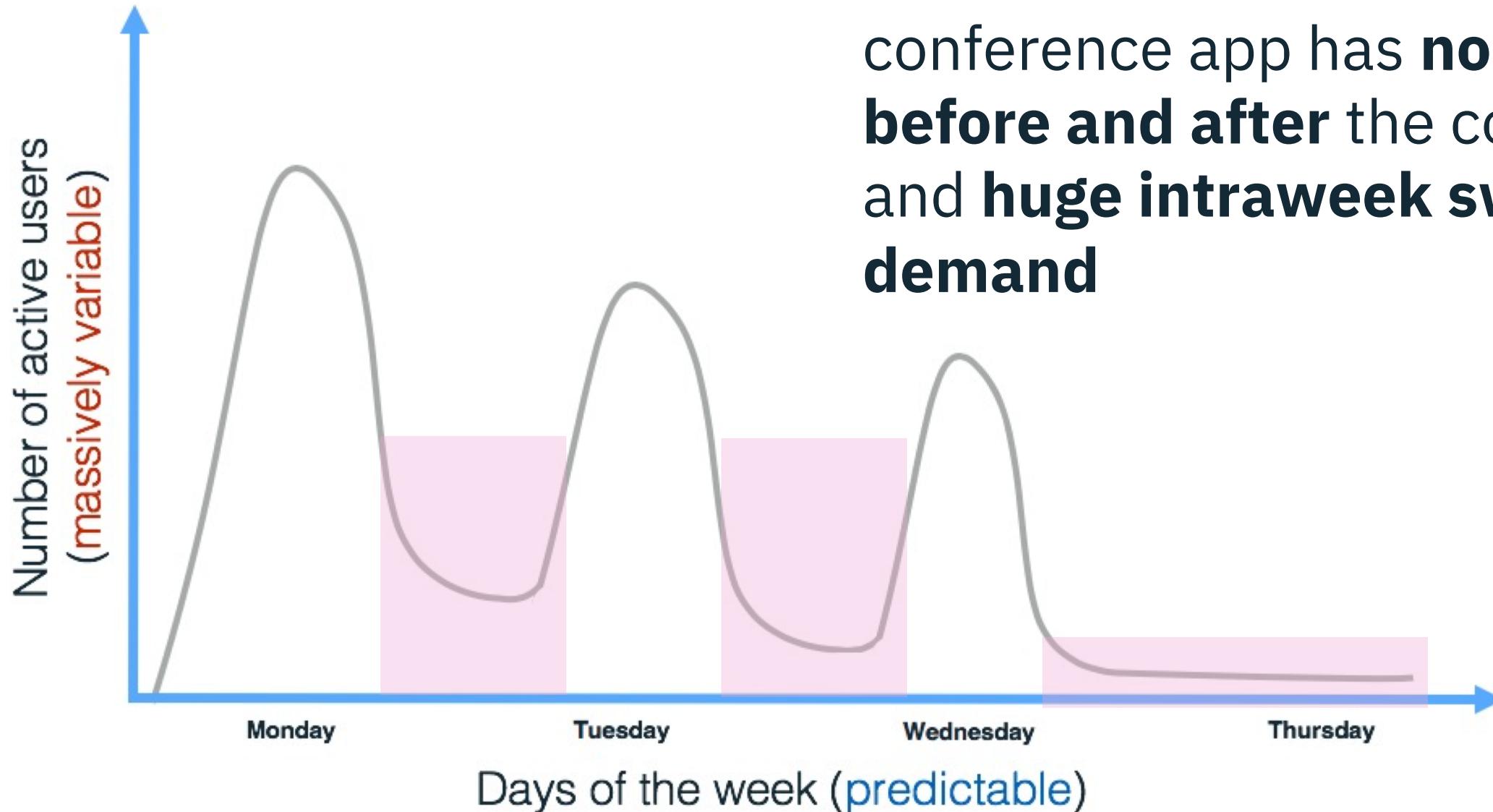
Small focused tasks with infrequent and/or unpredictable traffic

Why Serverless?



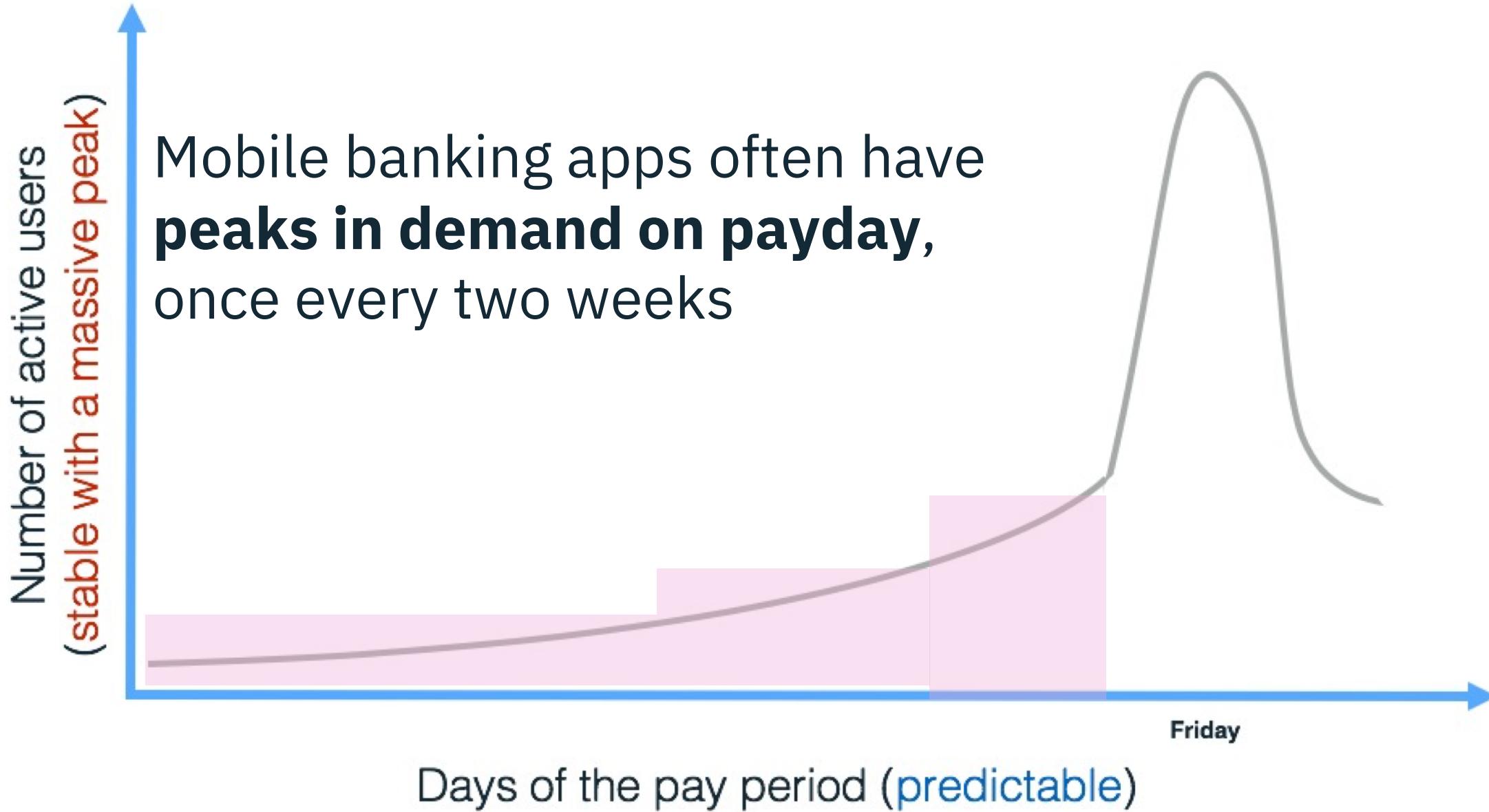
Active users curve for a new mobile game has a **predictable shape**, but an **unpredictable volume**

Why Serverless?



Active users curve for a mobile conference app has **no demand before and after** the conference, and **huge intraweek swings in demand**

Why Serverless?



Server Landscape (Cloud Native Computing Foundation)

Tools



Security



Framework



Hosted

Installable

Platform



Server Landscape – features

Amazon Lambda

- Node.js, Python, Java, C# and Go

IBM Cloud Functions

- Node.js 8, Node.js 6, Python 3.6.4, Python 3.6.1, PHP 7.1, PHP 7.2, and Swift 4, Swift 3.1.1, Ruby 2.5
- Other languages can be added via Docker container (for example Java)
- Hosted and managed open source OpenWhisk serverless platform. Also provides DIY approach.
- Recent change – 2GB max memory and 10 minutes run limit.

Microsoft Azure

- C#, F#, Node.js (in GA)
- Java, Python, PHP, TypeScript, Bash, PowerShell (experimental mode)

Google Cloud Function

- Node.js, Python
- **Many many more and constantly changing**

Apache OpenWhisk

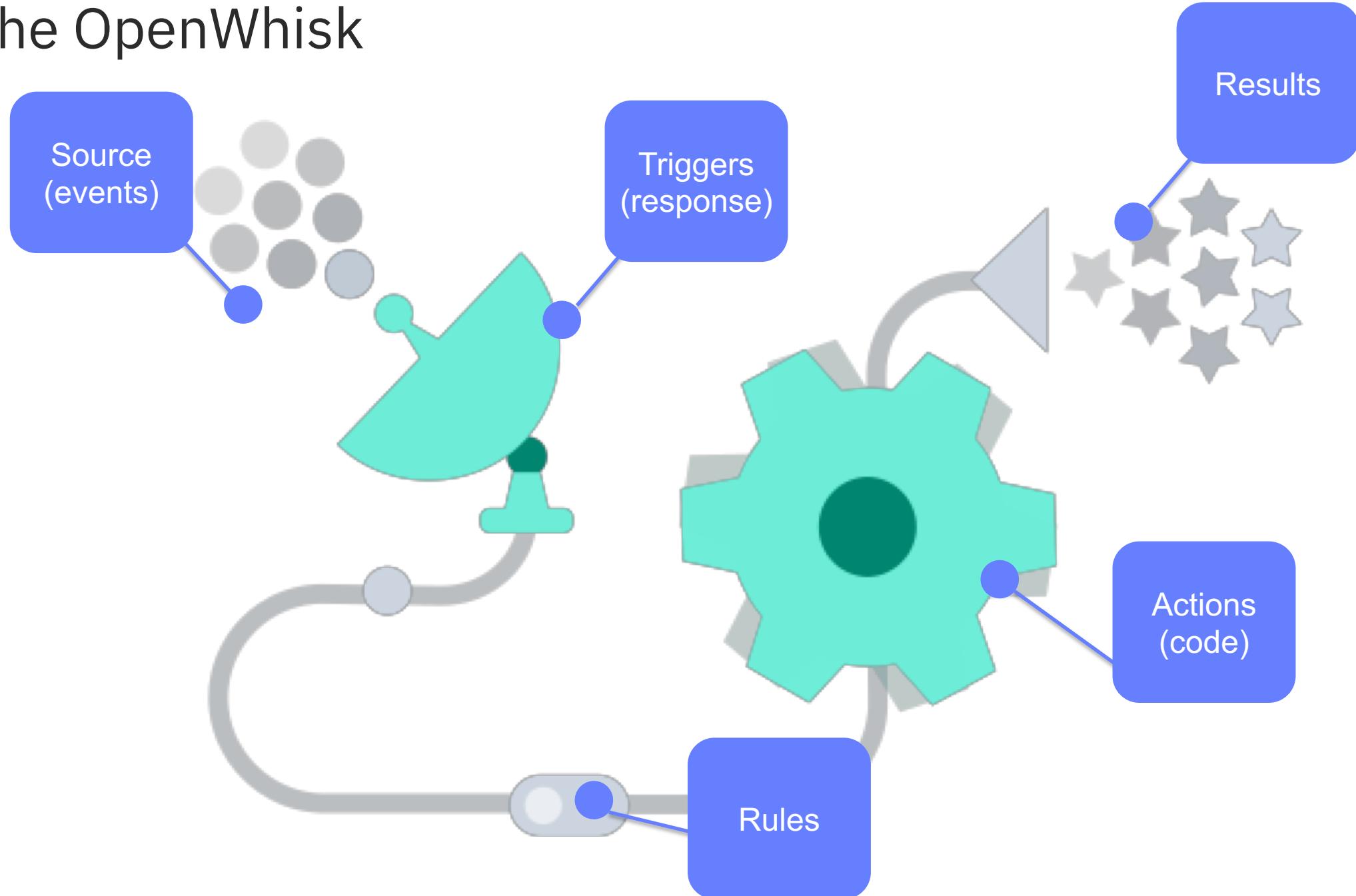
Apache OpenWhisk is an open source, distributed Serverless platform that supports a programming model in which developers write functional logic (called **Actions**), in any supported programming language, that can be dynamically scheduled and run in response to associated **events** (via **Triggers**) from external sources (**Feeds**) or from HTTP requests.

Supporters

The following companies and organizations have acknowledged support of the Apache OpenWhisk project as contributors or users of the technology.

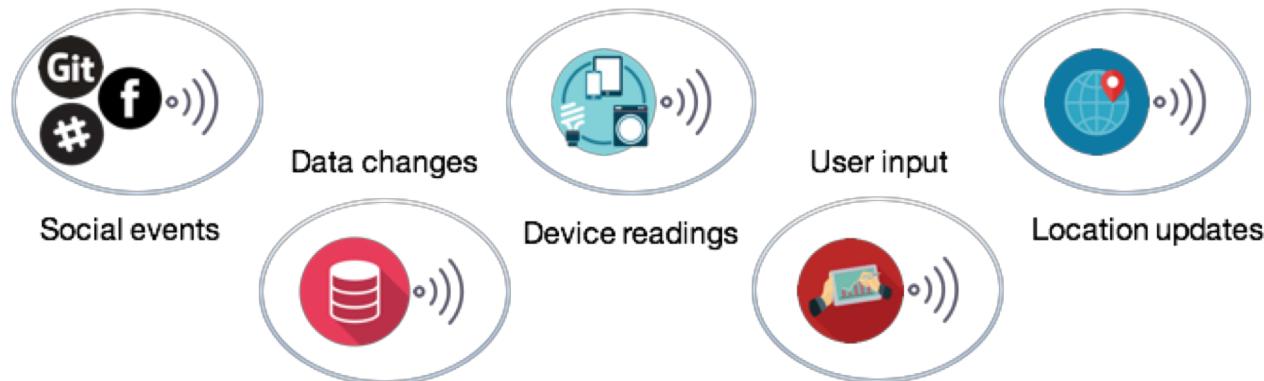


Apache OpenWhisk



Apache OpenWhisk

T A class of events that can occur



R

An association of a trigger to an action in a many to many mapping.



A Can be written in a variety of languages, such as JavaScript, Python, Java, PHP, and Swift

JS/NodeJS 8

Swift 4

Go

Ruby

Java

Docker

Rust

bash

Python 3

PHP 7

C

...

Use cases

Serverless Backends



Mobile Backend



Data Processing



Conversational Scenarios



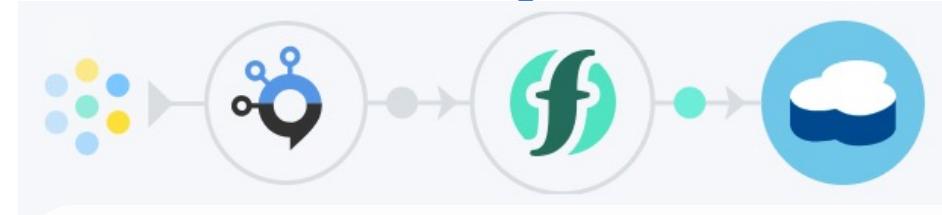
Cognitive Data Processing



IoT Ready



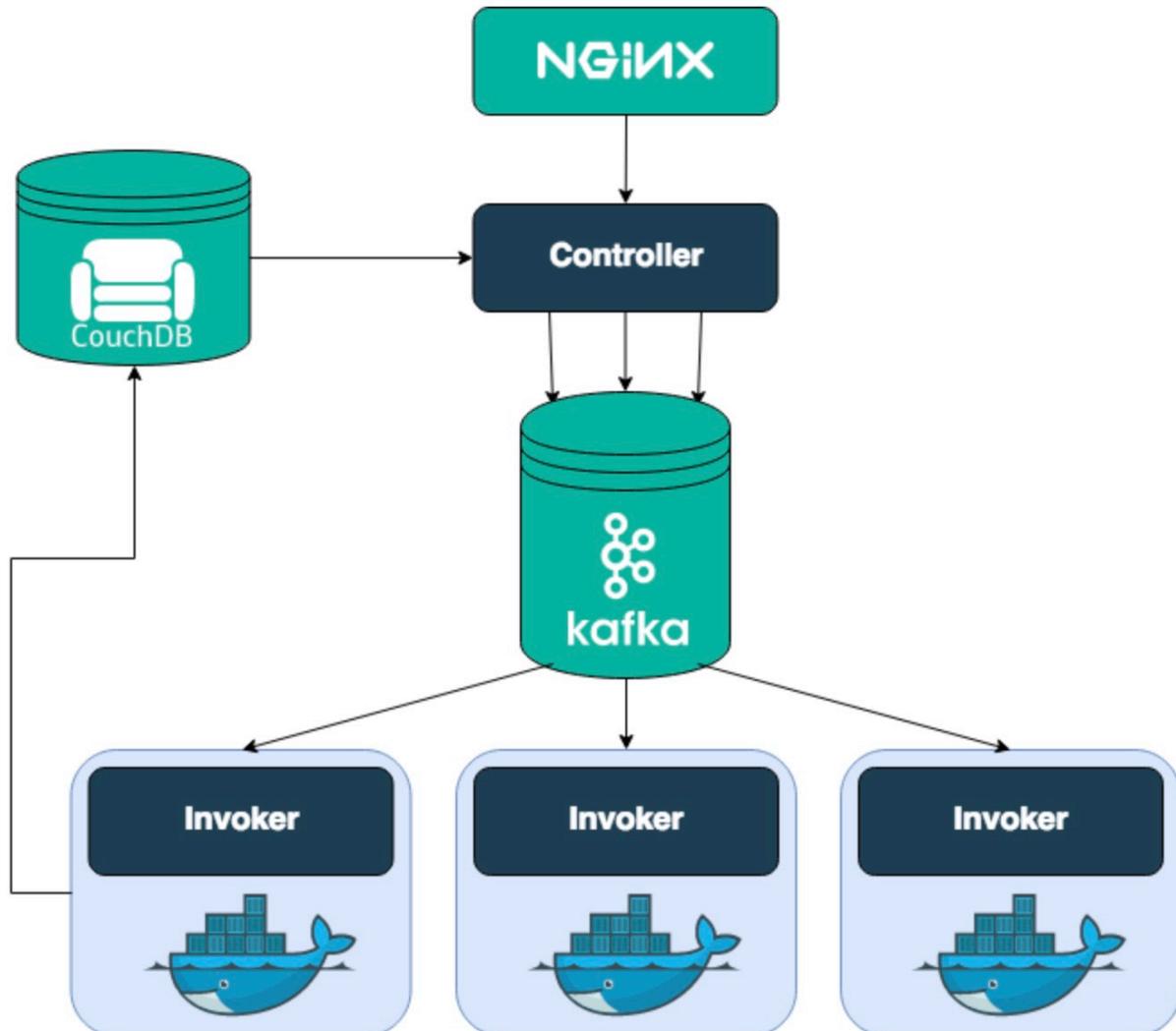
Event Stream Processing



Scheduled Tasks



Apache OpenWhisk



Entering the system: nginx
“an HTTP and reverse proxy server”.

Entering the system: Controller
serves as the interface for everything a user can do

Authentication and Authorization: CouchDB
check that the user exists in OpenWhisk’s database and that it has the privilege to invoke the action myAction

Getting the action: CouchDB... again
Load the action from the **whisks** database in CouchDB.

Who’s there to invoke the action: Load Balancer / Controller
has a global view of the executors available in the system by checking their health status continuously. Those executors are called **Invokers**. The Load Balancer, knowing which Invokers are available, chooses one of them to invoke the action requested.

Please form a line: Kafka (communication between controller and invokers)
a high-throughput, distributed, publish-subscribe messaging system

Actually invoking the code already: Invoker
execute actions in an isolated and safe way using **Docker**.

Storing the results: CouchDB again
Store the returned result, logs written, start and end time of the action.

Serverless is an architecture, a way of thinking that abstracts away the underlying hardware and infrastructure from the developer completely.

Apache OpenWhisk - wsk

```
}
```

```
~/D/c/openwhisk-example ➤ wsk
```



Usage:

```
wsk [command]
```

Available Commands:

```
action      work with actions
activation   work with activations
package     work with packages
rule        work with rules
trigger     work with triggers
sdk         work with the sdk
property    work with whisk properties
namespace   work with namespaces
list        list entities in the current namespace
api         work with APIs
bluemix    bluemix integration
```

Flags:

```
--apihost HOST      whisk API HOST
--apiversion VERSION  whisk API VERSION
-u, --auth KEY       authorization KEY
--cert string        client cert
-d, --debug          debug level output
-h, --help            help for wsk
-i, --insecure       bypass certificate checking
--key string         client key
-v, --verbose        verbose output
```

```
~/D/c/openwhisk-example ➤ wsk action --help
```

work with actions

Usage:

```
wsk action [command]
```

Available Commands:

```
create      create a new action
update      update an existing action, or create an action if it does not exist
invoke      invoke action
get         get action
delete      delete action
list        list all actions in a namespace or actions contained in a package
```

Global Flags:

--apihost HOST	whisk API HOST
--apiversion VERSION	whisk API VERSION
-u, --auth KEY	authorization KEY
--cert string	client cert
-d, --debug	debug level output
-i, --insecure	bypass certificate checking
--key string	client key
-v, --verbose	verbose output

```
~/D/c/openwhisk-example ➤ wsk trigger --help
```

work with triggers

Usage:

```
wsk trigger [command]
```

Available Commands:

fire	fire trigger event
create	create new trigger
update	update an existing trigger, or create a trigger if it does not exist
get	get trigger
delete	delete trigger
list	list all triggers

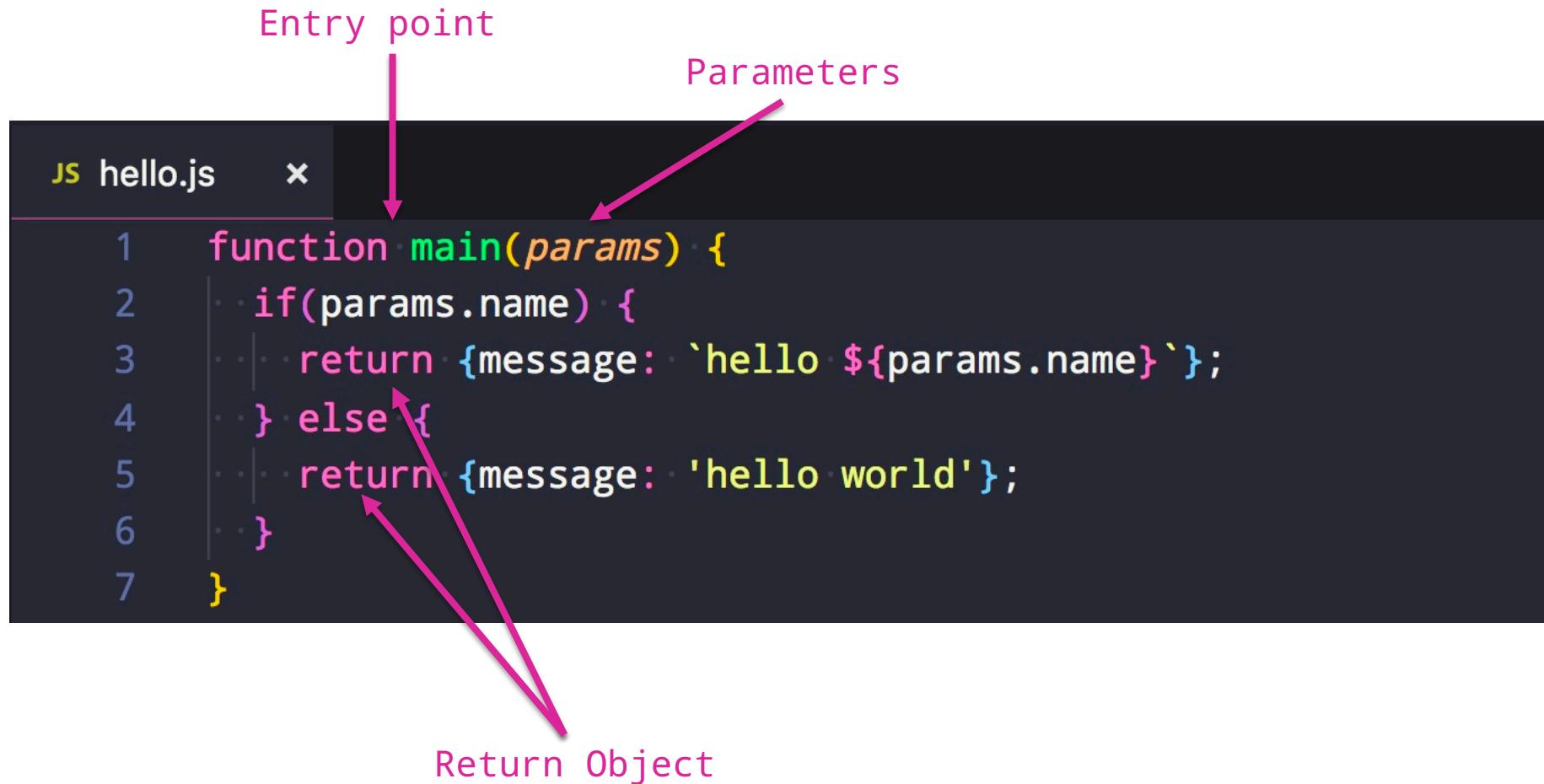
Global Flags:

--apihost HOST	whisk API HOST
--apiversion VERSION	whisk API VERSION
-u, --auth KEY	authorization KEY
--cert string	client cert
-d, --debug	debug level output
-i, --insecure	bypass certificate checking
--key string	client key
-v, --verbose	verbose output

Apache OpenWhisk

```
~/D/c/o/incubator-openwhisk-devtools ➤ docker-compose ➤ docker ps --format "{{.ID}}: {{.Names}} {{.Image}}"
9c38cf1cf6a: wsk0_36_prewarm_nodejs6 openwhisk/nodejs6action:latest
6b316404834c: wsk0_35_prewarm_nodejs6 openwhisk/nodejs6action:latest
33c7227b23af: openwhisk_apigateway_1 openwhisk/apigateway:latest
e62416c40058: openwhisk_invoker_1 openwhisk/invoker
d13f1d3157ec: openwhisk_controller_1 openwhisk/controller
672608ddb366: openwhisk_kafka-topics-ui_1 landoop/kafka-topics-ui:0.9.3
7654709edd2b: openwhisk_kafka-rest_1 confluentinc/cp-kafka-rest:3.3.1
af68360a86c2: openwhisk_kafka_1 wurstmeister/kafka:0.11.0.1
422c91ddf86a: openwhisk_db_1 apache/couchdb:2.1
dedb8e4552c8: openwhisk_zookeeper_1 zookeeper:3.4
d254ec1388ca: openwhisk_redis_1 redis:2.8
5ca234738543: openwhisk_minio_1 minio/minio:RELEASE.2018-07-13T00-09-07Z
```

Apache OpenWhisk – create an action



Apache OpenWhisk – create an action

```
~/D/c/openwhisk-example ➤ ls  
hello.js
```

```
~/D/c/openwhisk-example ➤ wsk action list -i  
actions
```

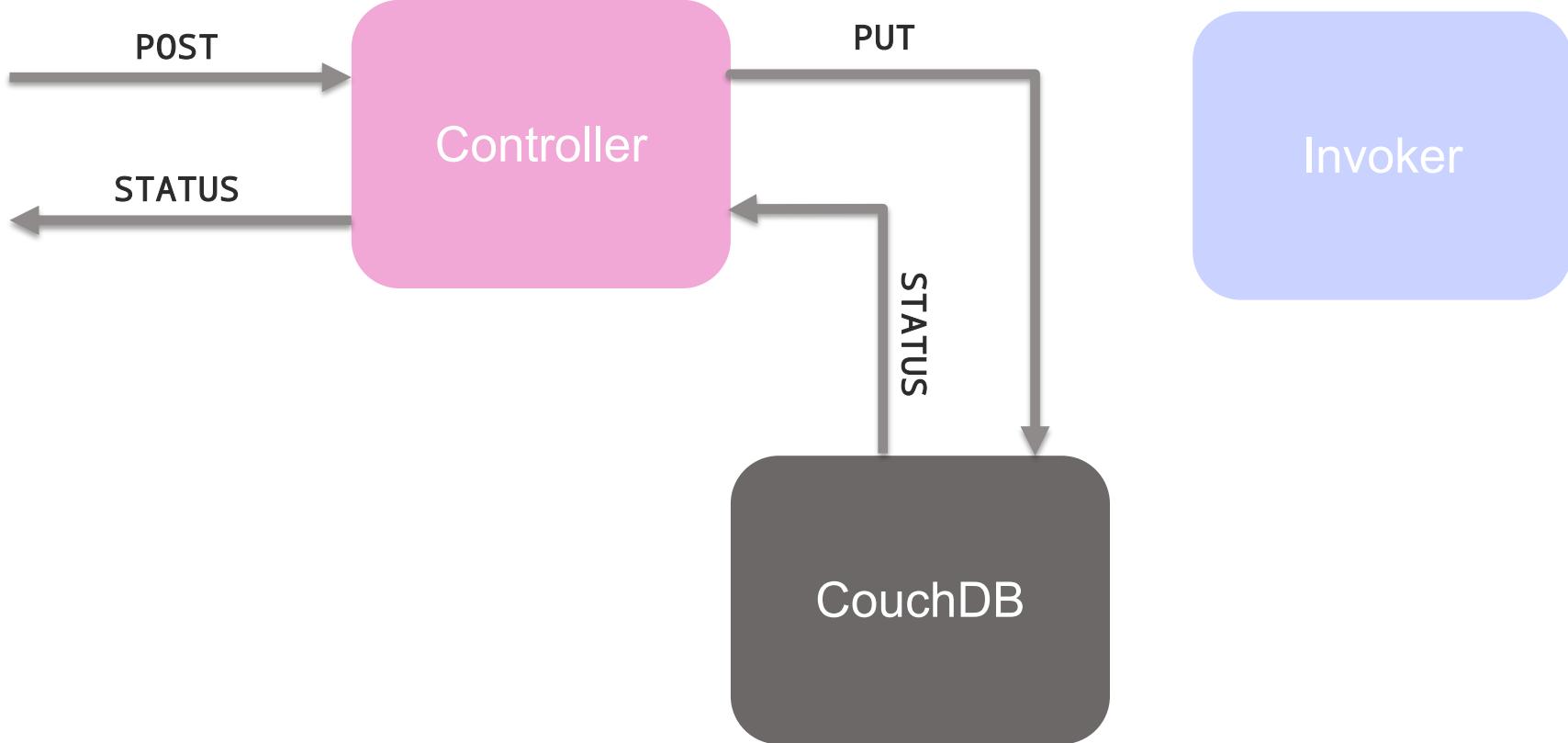
```
~/D/c/openwhisk-example ➤ wsk action create hello hello.js -i  
ok: created action hello
```

```
~/D/c/openwhisk-example ➤ wsk action list -i  
actions  
/guest/hello
```

private nodejs:6

Apache OpenWhisk – create an action

```
wsk action create hello hello.js -i
```



Apache OpenWhisk – invoke an action

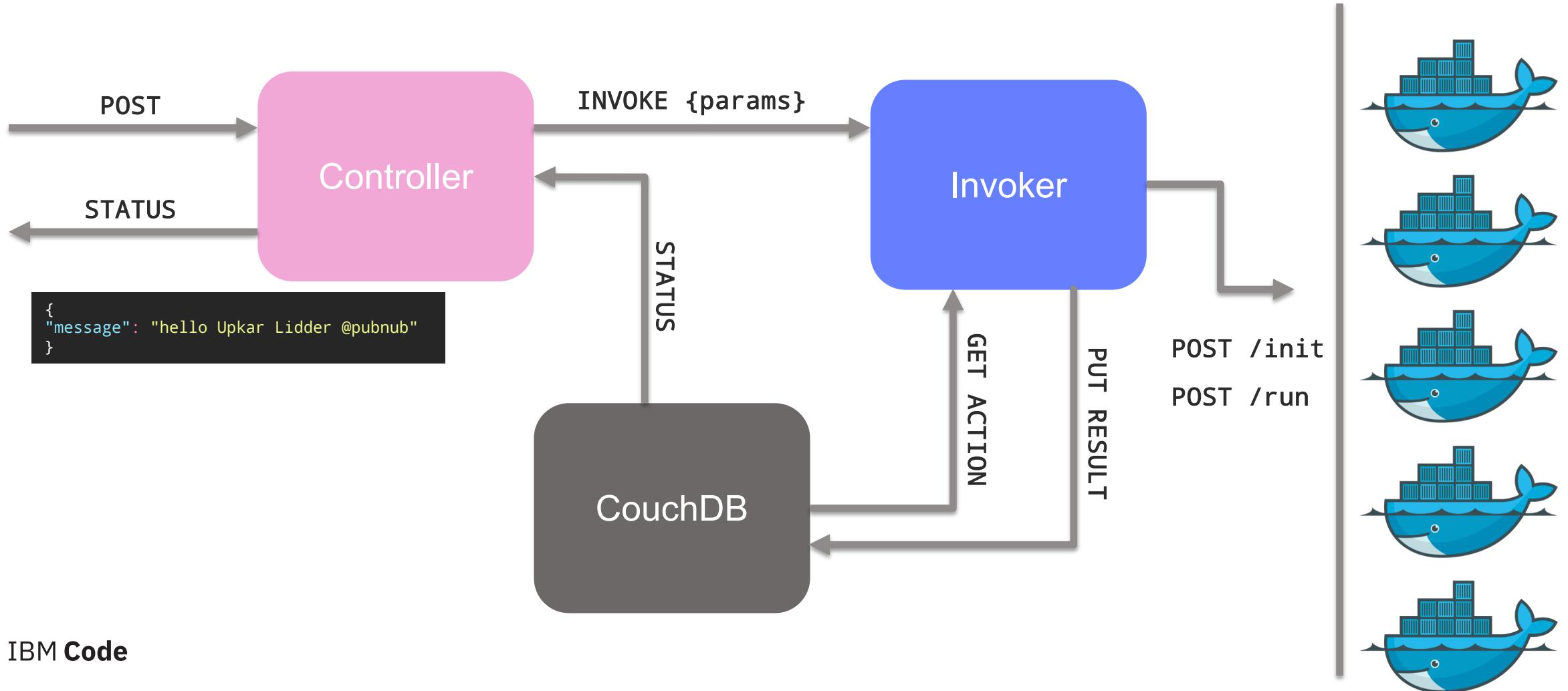
```
~/D/c/openwhisk-example ➜ wsk action invoke hello -i -r -v -p name "Upkar Lidder @pubnub"
REQUEST:
[POST] https://localhost/api/v1/namespaces/guest/actions/hello?blocking=true&result=true
Req Headers
{
  "Authorization": [
    "Basic [REDACTED]"
  ],
  "Content-Type": [
    "application/json"
  ],
  "User-Agent": [
    "CloudFunctions-CLI/1.0 (2018-07-12T22:20:03+00:00) darwin amd64"
  ]
}
Req Body
{"name":"Upkar Lidder @pubnub"}

RESPONSE:Got response with code 200
Resp Headers
{
  "Access-Control-Allow-Headers": [
    "Authorization, Origin, X-Requested-With, Content-Type, Accept, User-Agent"
  ],
  "Access-Control-Allow-Methods": [
    "GET, DELETE, POST, PUT, HEAD"
  ],
  "Access-Control-Allow-Origin": [
    "*"
  ],
  "Connection": [
    "keep-alive"
  ],
  "Content-Length": [
    "40"
  ],
  "Content-Type": [
    "application/json"
  ],
  "Date": [
    "Mon, 19 Nov 2018 20:09:24 GMT"
  ],
  "Server": [
    "openresty/1.13.6.2"
  ],
  "X-Openwhisk-Activation-Id": [
    "27363aab6d244db9b63aab6d24ddb999"
  ],
  "X-Request-Id": [
    "ZHi96YE5NiUVNCnOCcGeRKmRPkNumXB"
  ]
}
Response body size is 40 bytes
Response body received:
{"message":"hello Upkar Lidder @pubnub"}
{
  "message": "hello Upkar Lidder @pubnub"
}
```

```
{  
  ...|... "message": "hello Upkar Lidder @pubnub"  
}  
}
```

Apache OpenWhisk – invoke an action

```
wsk action invoke hello -i -r -p name "Upkar Lidder @pubnub"
```



ForwardJS – Serverless Workshop

The image displays a composite view of a browser and a terminal window illustrating a serverless architecture.

Left Window (Browser): Gmail Inbox

The Gmail inbox shows three messages:

- A new message from "uldder" with subject "Thank you for joining forwardJS - This is a cool Exercise!" and timestamp "10:52 AM".
- A message from "IBM Cloud" with subject "Action required: Confirm your IBM Cloud account - Hello Upkar, Th..." and timestamp "10/7/18".
- A message from "ibmacct" with subject "New User Registration - IBMid Confirmation code Use this code to..." and timestamp "10/7/18".

Middle Window (Browser): Contract Registration Page

The page has a heading "Contract Registration Page" and a message "Hiya," followed by "Enter your email and/or phone number and see the OpenWhisk magic". It contains two input fields: "email@domain.com" and "+1XXX" (with the last few digits redacted), and a "Register" button.

Bottom Window (Terminal): Cloud Function Logs

The terminal shows logs for several cloud functions:

- Function 1 (stdout):

```
1. ic /Users/uldder (cloud-functions)
   ~L/M/c/D/c/o/forwardjs (fish)
   "2019-01-23T18:52:26.211340356Z stdout: "
]
```
- Function 2 (stdout):

```
Activation: 'read' (044e7439bb9347d48e7439bb9357d444)
[
  "2019-01-23T18:52:25.871398987Z stdout: success { _id: 'd2ddf03de371fdd410e481b82b76a81d', ",
  "2019-01-23T18:52:25.87146648Z stdout: _rev: '1-6f56bd4b7485bd444a47e863c754a0ae4', ",
  "2019-01-23T18:52:25.871473667Z stdout: user_email: 'upkar.ibm.watson@gmail.com', ",
  "2019-01-23T18:52:25.871481562Z stdout: user_phone: '+14155701199' }"
]
```
- Function 3 (stdout):

```
Activation: 'userchangeseq' (1c0be337e32744a18be337e327b4a1b0)
[
  "044e7439bb9347d48e7439bb9357d444",
  "26ccf4a5035a43f88cf4a5035a03f83c"
]
```
- Function 4 (stdout):

```
Activation: 'myCloudantTrigger' (efd4aa9cc9ac45be94aa9cc9ac45be61)
[
  "{\"statusCode\":0,\"success\":true,\"activationId\":\"1c0be337e32744a18be337e327b4a1b0\",\"rule\":\"upkar.ibm.watson@gmail.com_dev/upkar-cloudant-changed\\\",\"action\":\"upkar.ibm.watson@gmail.com_dev/forwardjs-dev/userchangeseq\\\"}"
]
```
- Function 5 (stdout):

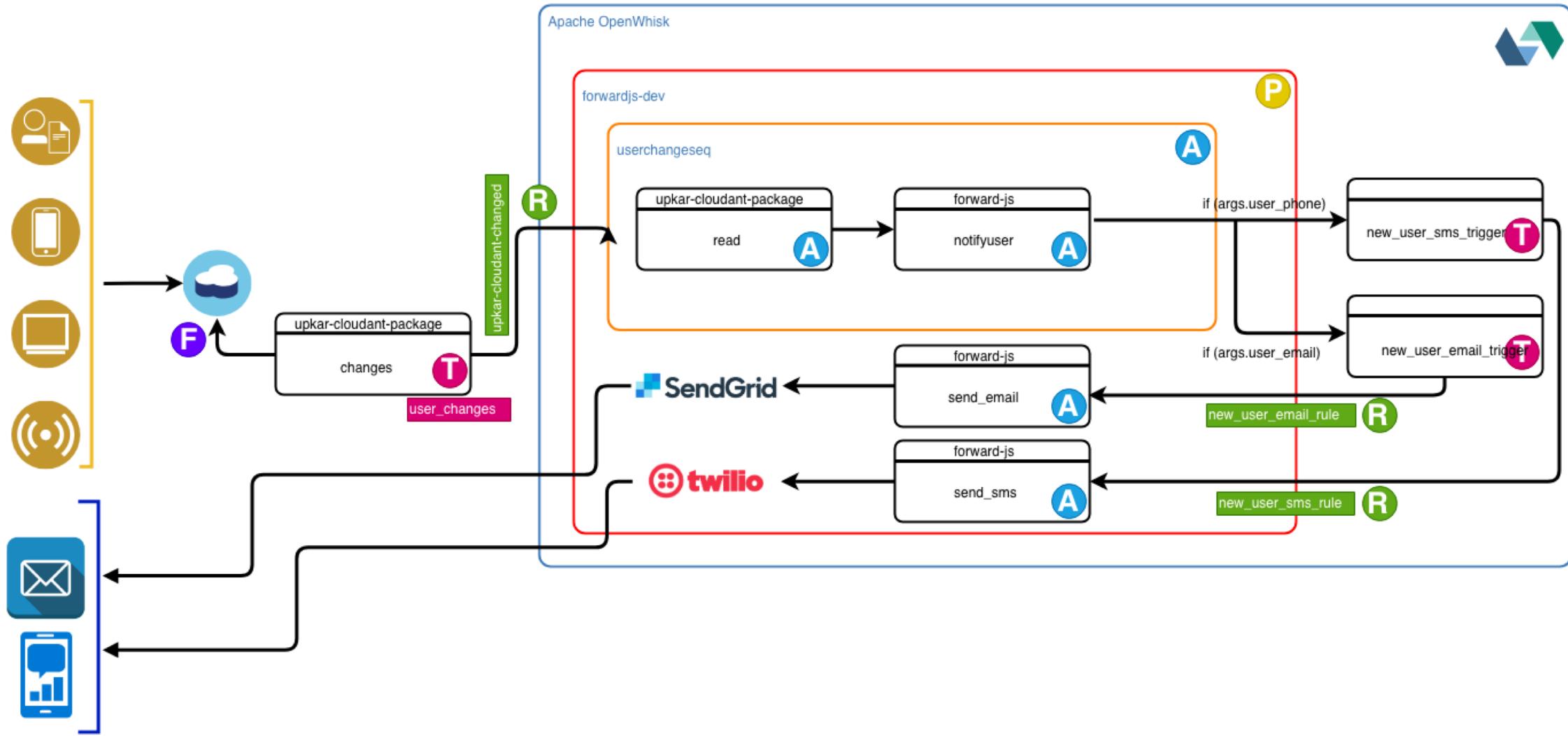
```
Activation: 'send_email' (9b3d319c50bd4995bd319c50bd09955b)
[
  "2019-01-23T18:52:27.482975979Z stdout: email sent to : upkar.ibm.watson@gmail.com"
]
```
- Function 6 (stdout):

```
Activation: 'send_sms' (4754357216f9455094357216f99550a2)
[
  "2019-01-23T18:52:28.652058204Z stdout: SMaba269f3ac9647778bca36198cdf82bf",
  "2019-01-23T18:52:28.652113684Z stdout: sent message from +14153221692 to +14155701199"
]
```

ForwardJS – Serverless Workshop

```
~/t/openwhisk-registration ➜ master ➔ ic fn list
Entities in namespace: default
packages
/upkar.ibm.watson.3@gmail.com_dev/upkar-cloudant-package           private
/upkar.ibm.watson.3@gmail.com_dev/forwardjs-dev                      private
actions
/upkar.ibm.watson.3@gmail.com_dev/forwardjs-dev/userchangeseq        private sequence
/upkar.ibm.watson.3@gmail.com_dev/forwardjs-dev/notifyuser            private nodejs:6
/upkar.ibm.watson.3@gmail.com_dev/forwardjs-dev/send_sms              private nodejs:6
/upkar.ibm.watson.3@gmail.com_dev/forwardjs-dev/send_email            private nodejs:6
triggers
/upkar.ibm.watson.3@gmail.com_dev/user_changes_trigger                private
/upkar.ibm.watson.3@gmail.com_dev/new_user_sms_trigger                private
/upkar.ibm.watson.3@gmail.com_dev/new_user_email_trigger             private
rules
/upkar.ibm.watson.3@gmail.com_dev/upkar-cloudant-changed_rule         private
/upkar.ibm.watson.3@gmail.com_dev/new_user_sms_rule                  active
/upkar.ibm.watson.3@gmail.com_dev/new_user_email_rule                active
~/t/openwhisk-registration ➜ master ➔
```

ForwardJS – Serverless Workshop



IBM Cloud Functions

	Open source	Hosted service
Serverless engine	Apache OpenWhisk	IBM Cloud Functions
API Gateway	LoopBack	IBM API Gateway
Databases	Apache CouchDB MySQL	IBM Cloudant IBM Compose
Message streams	Apache Kafka	IBM Message Hub

Things to consider

- Functions are stateless. Need some sort of persistence between runs.
- Are you able to test and develop locally ? Does provider have CLI ?
- Can you easily version your functions ? Source control ?
- Can you easily monitor your functions ?
- Security and API gateway
- Avoid long-running loops / mini-monoliths ?
- Latency (cold, warm and hot loads)
- How do you track dependencies ?

System Limitations - OpenWhisk

limit	description	configurable	unit	default
timeout	a container is not allowed to run longer than N milliseconds	per action	milliseconds	60000
memory	a container is not allowed to allocate more than N MB of memory	per action	MB	256
logs	a container is not allowed to write more than N MB to stdout	per action	MB	10
concurrent	no more than N activations may be submitted per namespace either executing or queued for execution	per namespace	number	100
minuteRate	no more than N activations may be submitted per namespace per minute	per namespace	number	120
codeSize	the maximum size of the actioncode	configurable, limit per action	MB	48
parameters	the maximum size of the parameters that can be attached	not configurable, limit per action/package/trigger	MB	1
result	the maximum size of the action result	not configurable, limit per action	MB	1

System Limitations – IBM Cloud Functions

Limit	Description	Default	Min	Max
codeSize	The maximum size of the action code in MB.	48	1	48
concurrent	No more than N activations can be submitted per namespace either executing or queued for execution.	1000	1	1000*
logs	A container is not allowed to write more than N MB to stdout.	10	0	10
memory	A container is not allowed to allocate more than N MB of memory.	256	128	2048
minuteRate	No more than N activations can be submitted per namespace per minute.	5000	1	5000*
openulimit	The maximum number of open files for an action.	1024	0	1024
parameters	The maximum size of the parameters that can be attached in MB.	1	0	1
proclimit	The maximum number of processes available to an action.	1024	0	1024
result	The maximum size of the action invocation result in MB.	1	0	1
sequenceMaxActions	The maximum number of actions that comprise a given sequence.	50	0	50*
timeout	A container is not allowed to run longer than N milliseconds.	60000	100	600000

Resources

Lists

- <https://github.com/anaibol/awesome-serverless>
- <https://github.com/pmuens/awesome-serverless>
- <https://twitter.com/tmclaughbos/lists/serverless>

Email newsletter

- <https://serverless.email/>

Code Patterns

- <https://developer.ibm.com/patterns/category/serverless/>

Serverless architecture

- <https://martinfowler.com/articles/serverless.html>

Future of Serverless ?

[kelseyhightower / nocode](https://github.com/kelseyhightower/nocode)

Code Issues Pull requests Projects Wiki Insights

The best way to write secure and reliable applications. Write nothing; deploy nowhere.

3 commits 1 branch 1 release 1 contributor Apache-2.0

Branch: master New pull request Create new file Upload files Find file Clone or download

kelseyhightower add Docker support 7 Latest commit ed6c73f on Feb 6

CONTRIBUTING.md	add no code	9 months ago
Dockerfile	add Docker support	9 months ago
LICENSE	add no code	9 months ago
README.md	add windows support	9 months ago

README.md

No Code

No code is the best way to write secure and reliable applications. Write nothing; deploy nowhere.

Getting Started

Start by not writing any code.

This is just an example application, but imagine it doing anything you want. Adding new features is easy too:

The possibilities are endless.

Building the Application

Now that you have not done anything it's time to build your application:

Building the Application

Now that you have not done anything it's time to build your application:

Yep. That's it. You should see the following output:

Deploying

While you still have not done anything it's time to deploy your application. By running the following command you can deploy your application absolutely nowhere.

It's that simple. And when it comes time to scale the application, all you have to do is:

I know right?

Contributing

You don't.



Thank you!

Upkar Lidder
Developer Advocate, IBM

> ulidder@us.ibm.com
> [@lidderupk](https://twitter.com/lidderupk)
> blog.upkarlidder.com

IBM Code

Code Patterns

CODE PATTERN | SEP 19, 2018

Train an Anki Cozmo robot to recognize other toys

[Get the Code »](#)

Containers IBM Cloud +

CODE PATTERN | AUG 28, 2018

Run serverless functions with image recognition

[Get the Code »](#)

IBM Cloud Serverless

CODE PATTERN | MAR 14, 2018

Deploy a serverless multilingual conference room

[Get the Code »](#)

Cloud Foundry Gaming +

CODE PATTERN | FEB 21, 2018

Create a podcast downloader using serverless technology

[Get the Code »](#)

IBM Cloud Microservices +

CODE PATTERN | NOV 28, 2017

Develop protected serverless web applications

[Get the Code »](#)

Cloud IBM Cloud +

CODE PATTERN | NOV 27, 2017

Analyze an image and send a status alert

[Get the Code »](#)

Cloud Foundry Node.js +

CODE PATTERN | NOV 16, 2017

Analyze industrial equipment for defects

[Get the Code »](#)

IoT Platform as a Service +

CODE PATTERN | OCT 20, 2017

Create an Alexa skill with serverless and a conversation

[Get the Code »](#)

Cloud Databases +

OpenWhisk Deployment

OpenWhisk supports different installation methods for core OpenWhisk platform components. You can choose one of the following deployment methods based on your platform:

Kubernetes

OpenWhisk can be deployed using [Helm](#) charts on Kubernetes provisioned locally or from a public cloud provider.

Please refer to: [Deploy OpenWhisk to a Kubernetes Cluster](#) for detailed deployment instructions which includes specific [customizations](#) including Docker Desktop (Mac, Windows), Minikube, Google, IBM Cloud, etc.).

Docker Compose

One of the easiest way to start deploying OpenWhisk is to get Docker installed on Mac, Windows or Linux. This does not give you a production deployment but gives you enough of the pieces to start writing functions and seeing them executing. However, if you like using Docker directly, we recommend using one of our [Kubernetes-Docker](#) options.

Mesos

[Deploy OpenWhisk to a Mesos Cluster](#) is under active development.

OpenShift

[Here](#) are the instructions and details on how to deploy OpenWhisk on OpenShift.

Ansible

[Deploying OpenWhisk using Ansible](#) is generally done for creating a development environment. Most of the OpenWhisk tools follow this method of deployment in their CI/CD (Travis) pipeline. The OpenWhisk playbooks are structured such that it allows cleaning, deploying, or re-deploying a single component as well as the entire OpenWhisk stack.

Vagrant

A Vagrant machine is also available to run OpenWhisk on Mac, Windows PC or GNU/Linux. Downloading and install VirtualBox and Vagrant for your operating system and architecture. You can follow the steps under [Vagrant Setup](#) to run your first OpenWhisk action using Vagrant.

There are many different deployment methods available but we recommend [Deploy OpenWhisk to a Kubernetes Cluster](#) as its the most easy and quickest way of getting OpenWhisk deployed.