

# Introduction to Serverless Computing

IBM Code

Upkar Lidder  
Developer Advocate, IBM

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

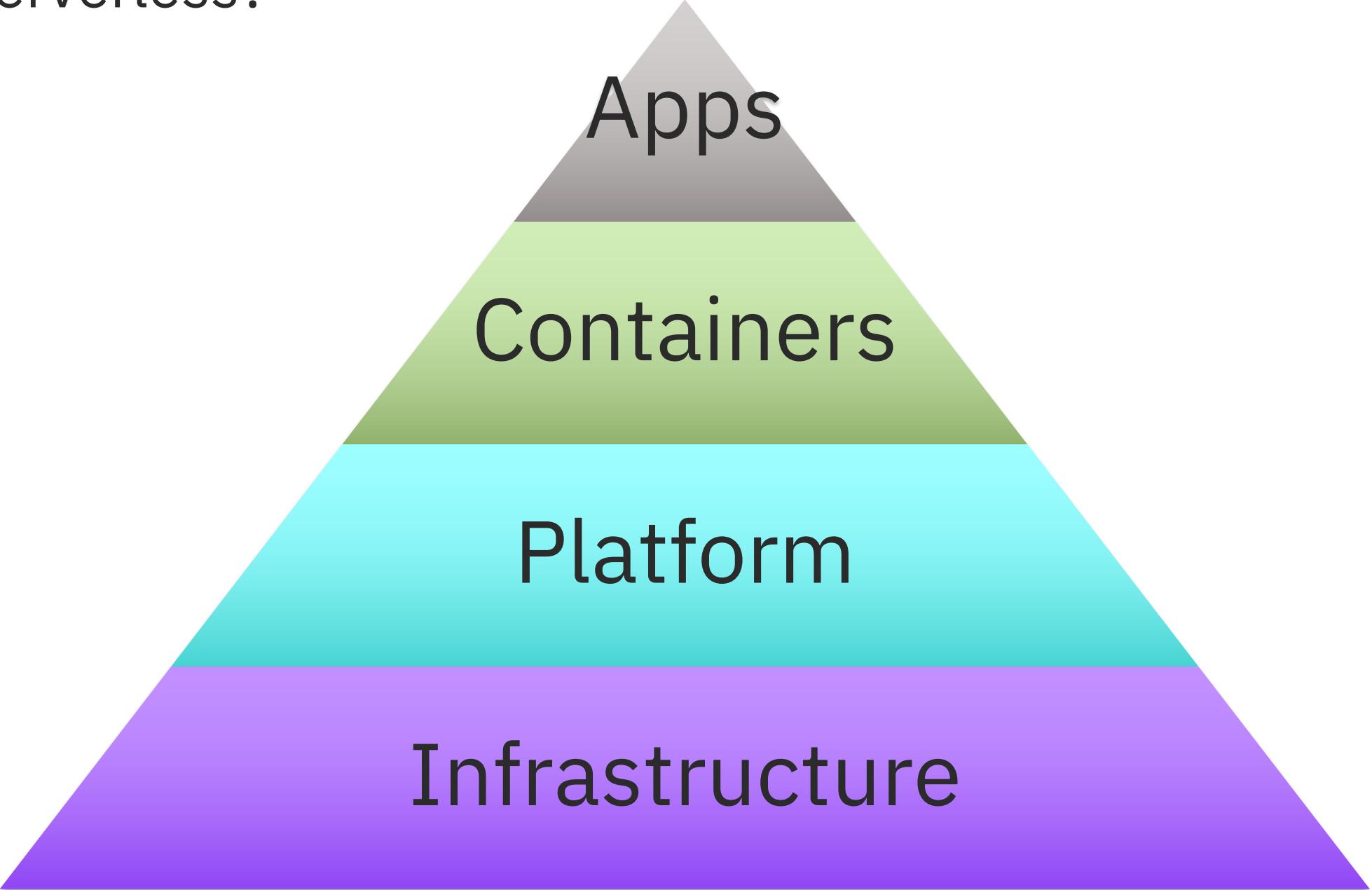
It describes a finer-grained deployment model where applications, **bundled as one or more functions**, are uploaded to a platform and then **executed, scaled, and billed** in response to the **exact demand needed** at the moment.

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

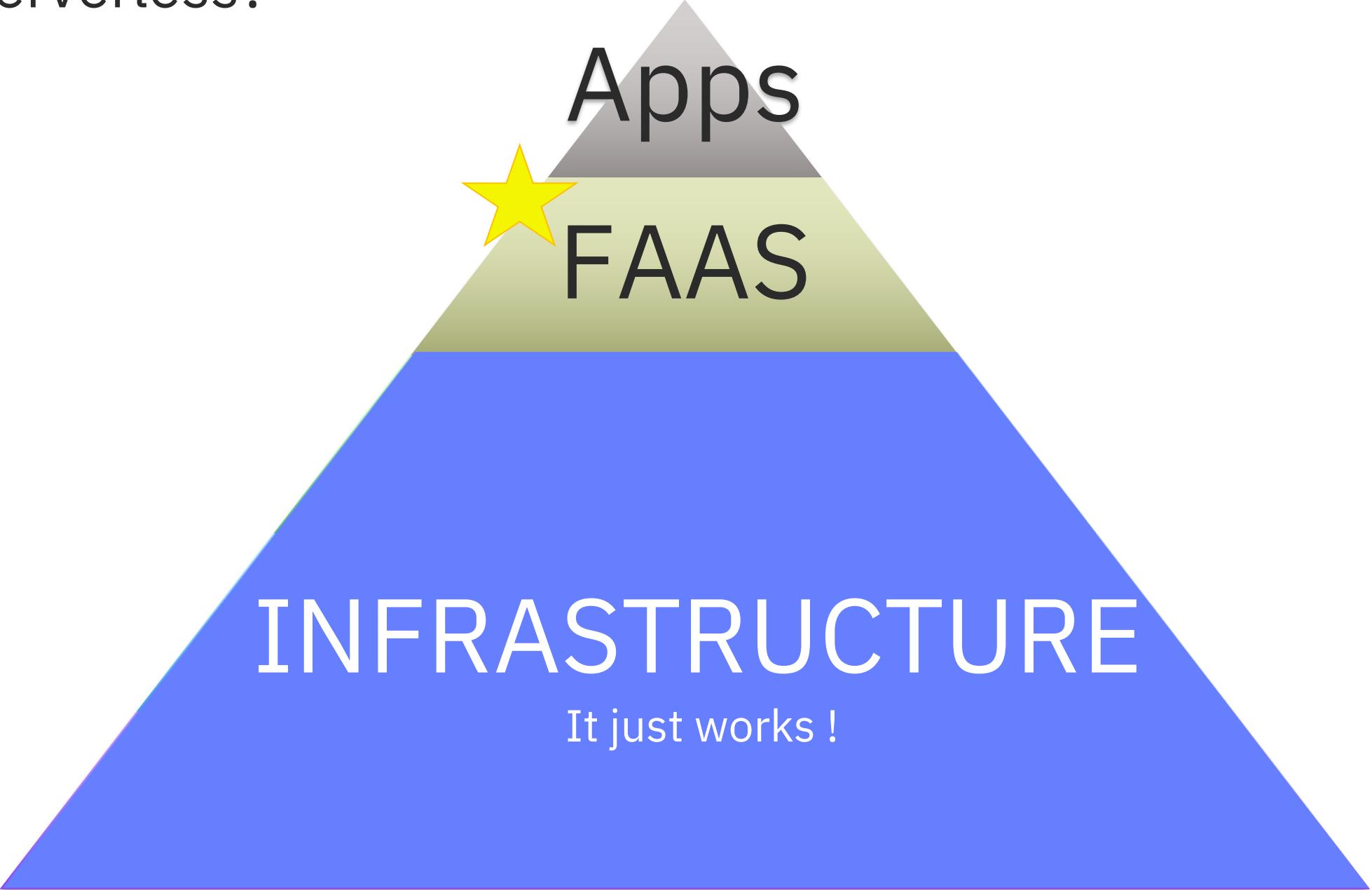
- **Cloud Native Computing Foundation**

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

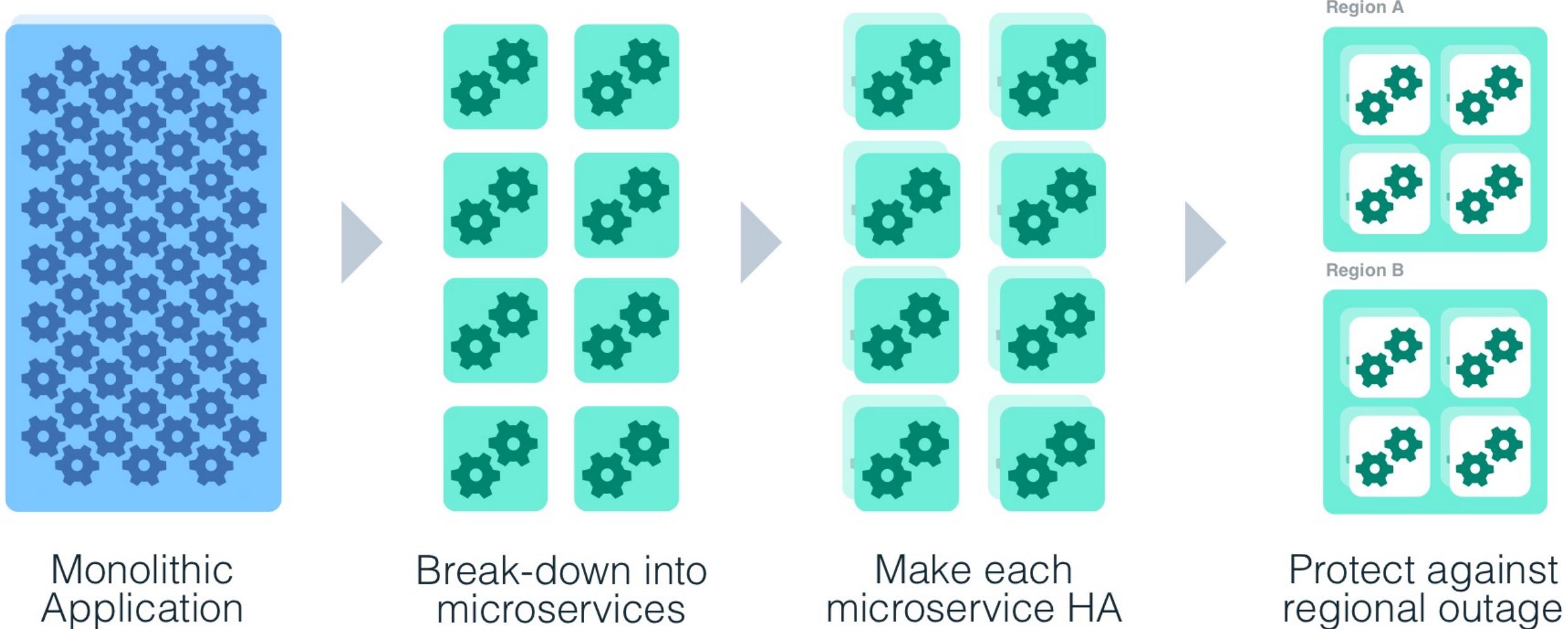
# What is Serverless?



# What is Serverless?



# What is Serverless?



# Use Cases

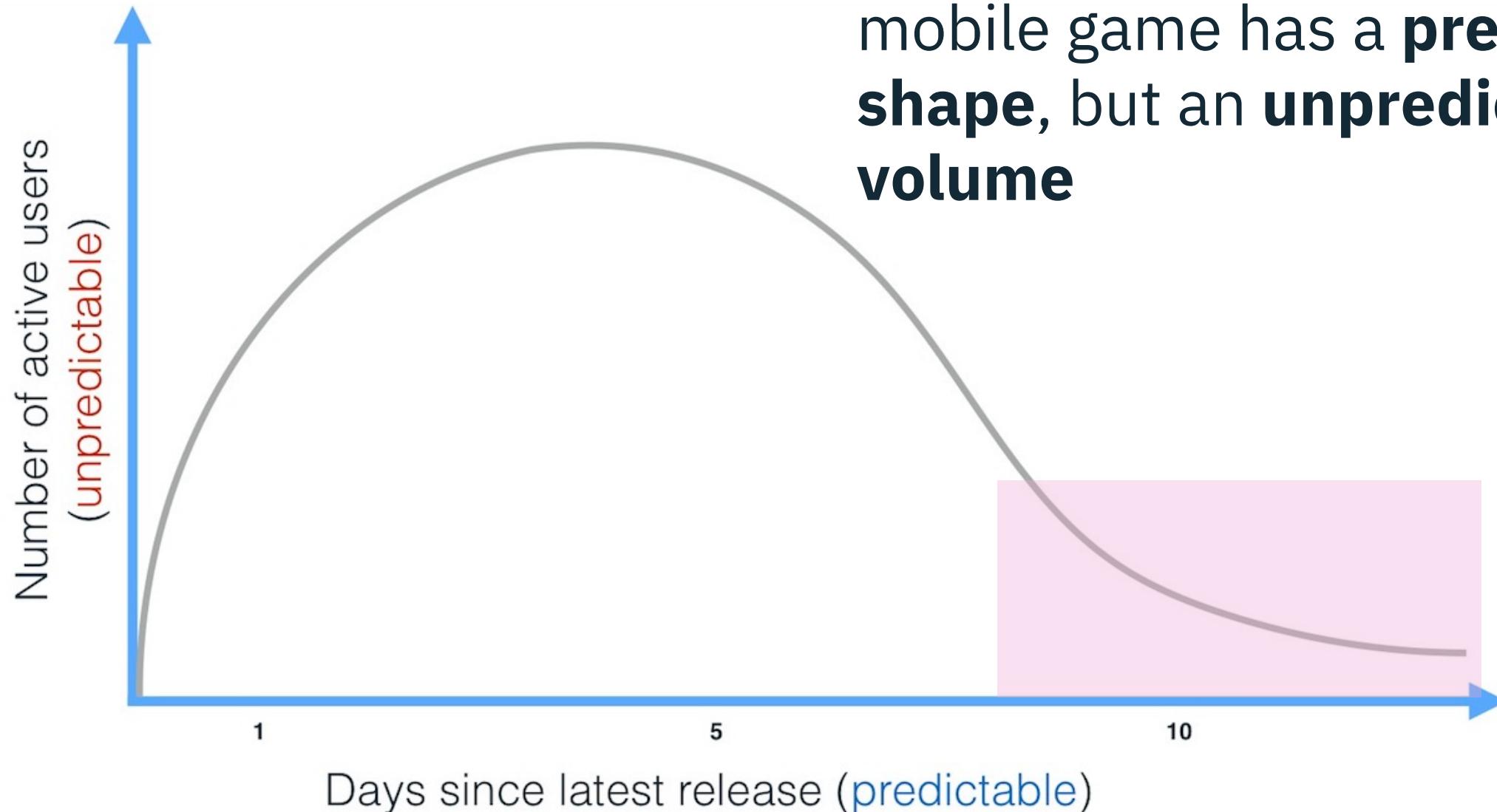
Small, focused, asynchronous, concurrent, easy to parallelize into independent units of work

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

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

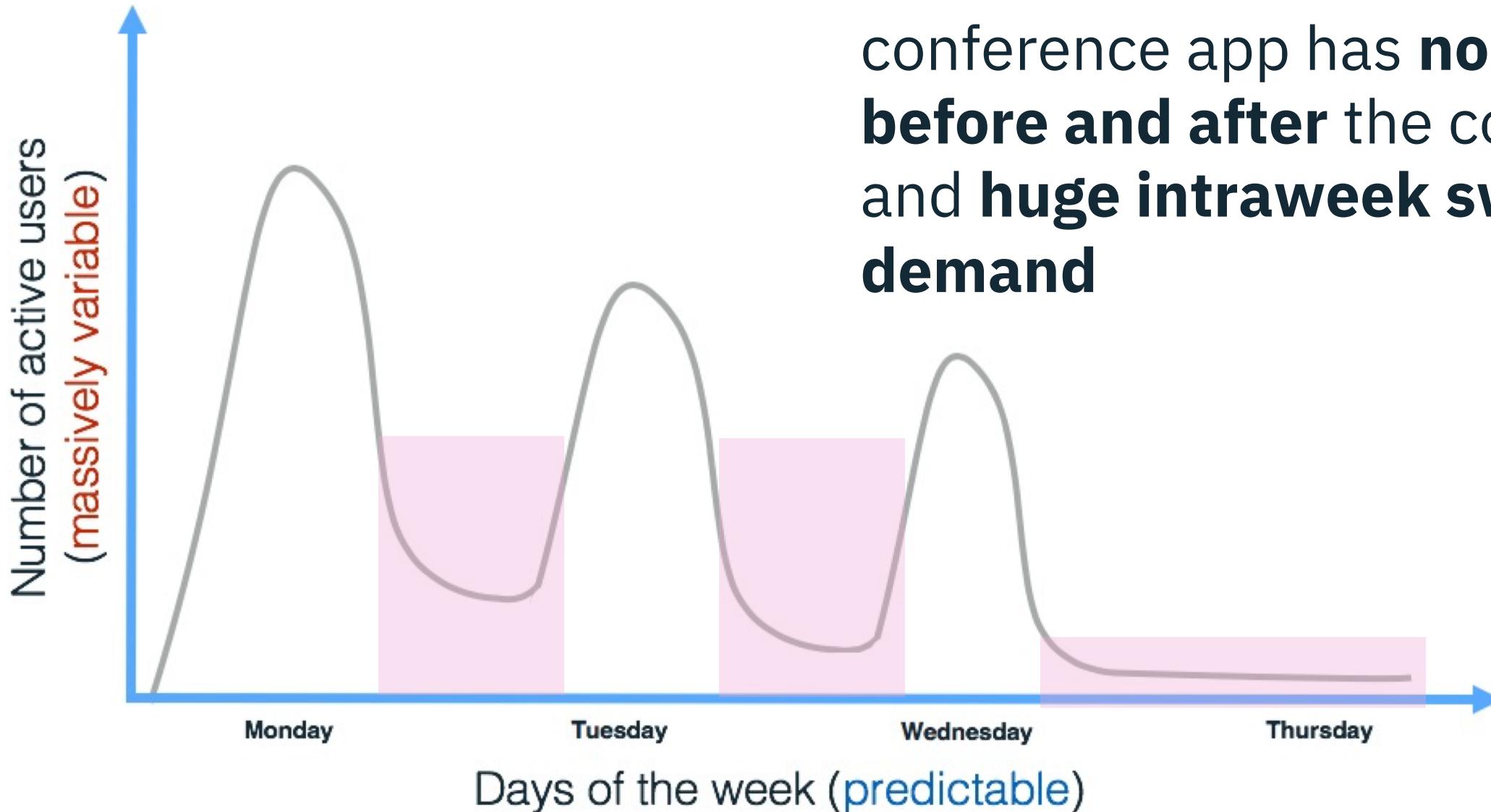
Highly dynamic in terms of changing business requirements that drive a need for accelerated developer velocity

# Why Serverless?



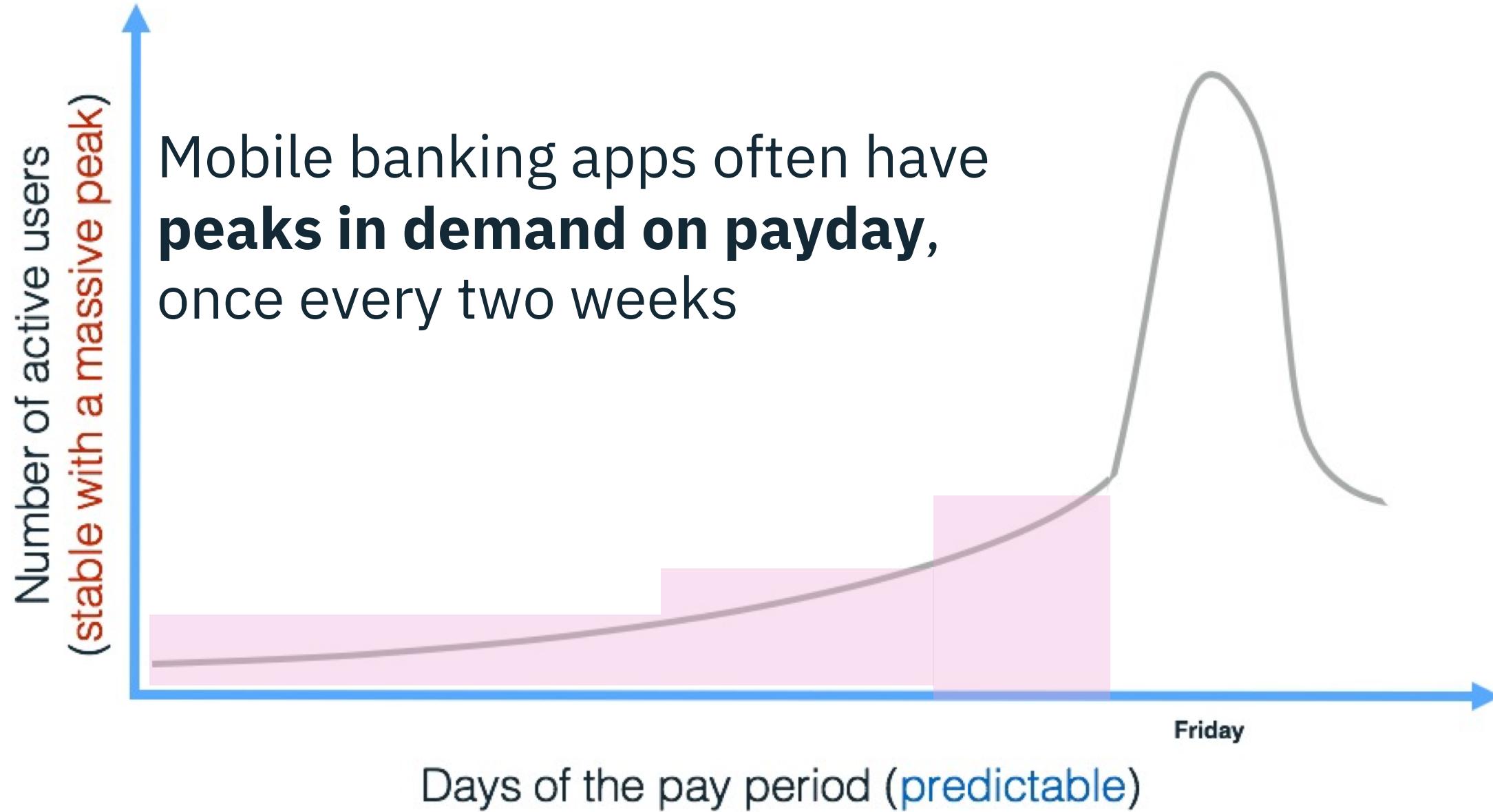
Active users curve for a new mobile game has a **predictable shape**, but an **unpredictable volume**

# 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

## Apache OpenWhisk

- 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)
- Based on open source OpenWhisk serverless platform. Can create your own serverless platform based on OpenWhisk

## 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 changings**

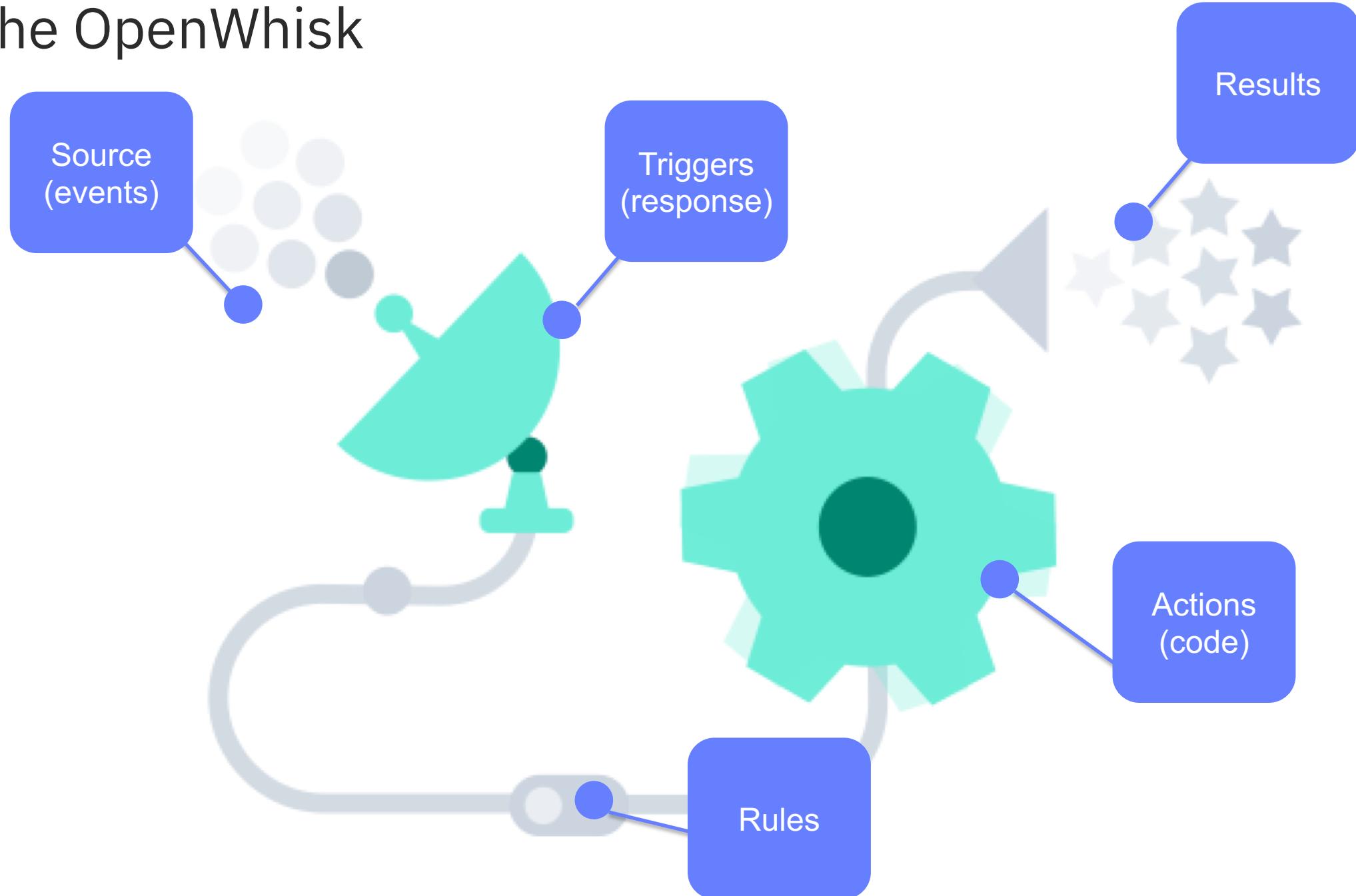
# Apache OpenWhisk

Apache OpenWhisk (Incubating) is an open source, distributed Serverless platform that executes functions (fx) in response to events at any scale.

The OpenWhisk platform 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.

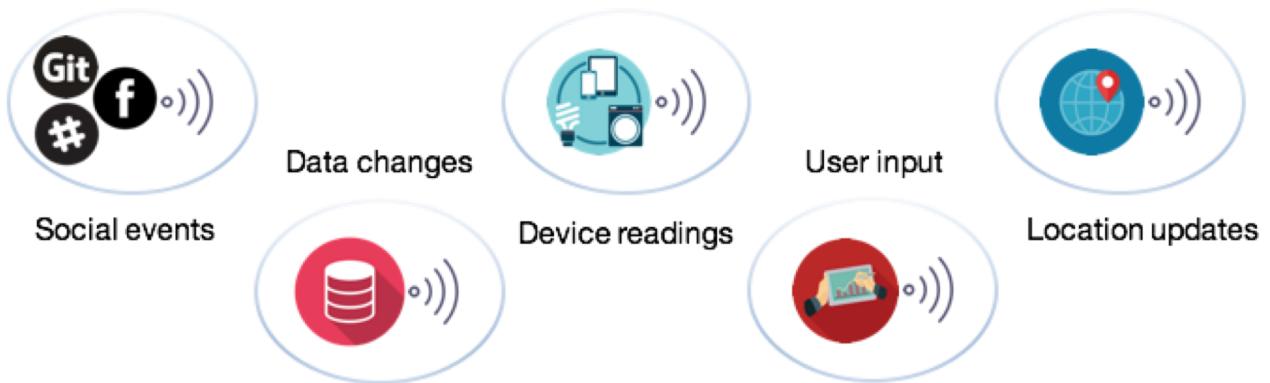


# Apache OpenWhisk



# 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



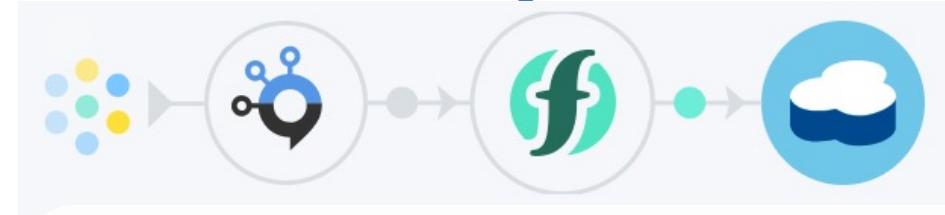
### Cognitive Data Processing



### IoT Ready



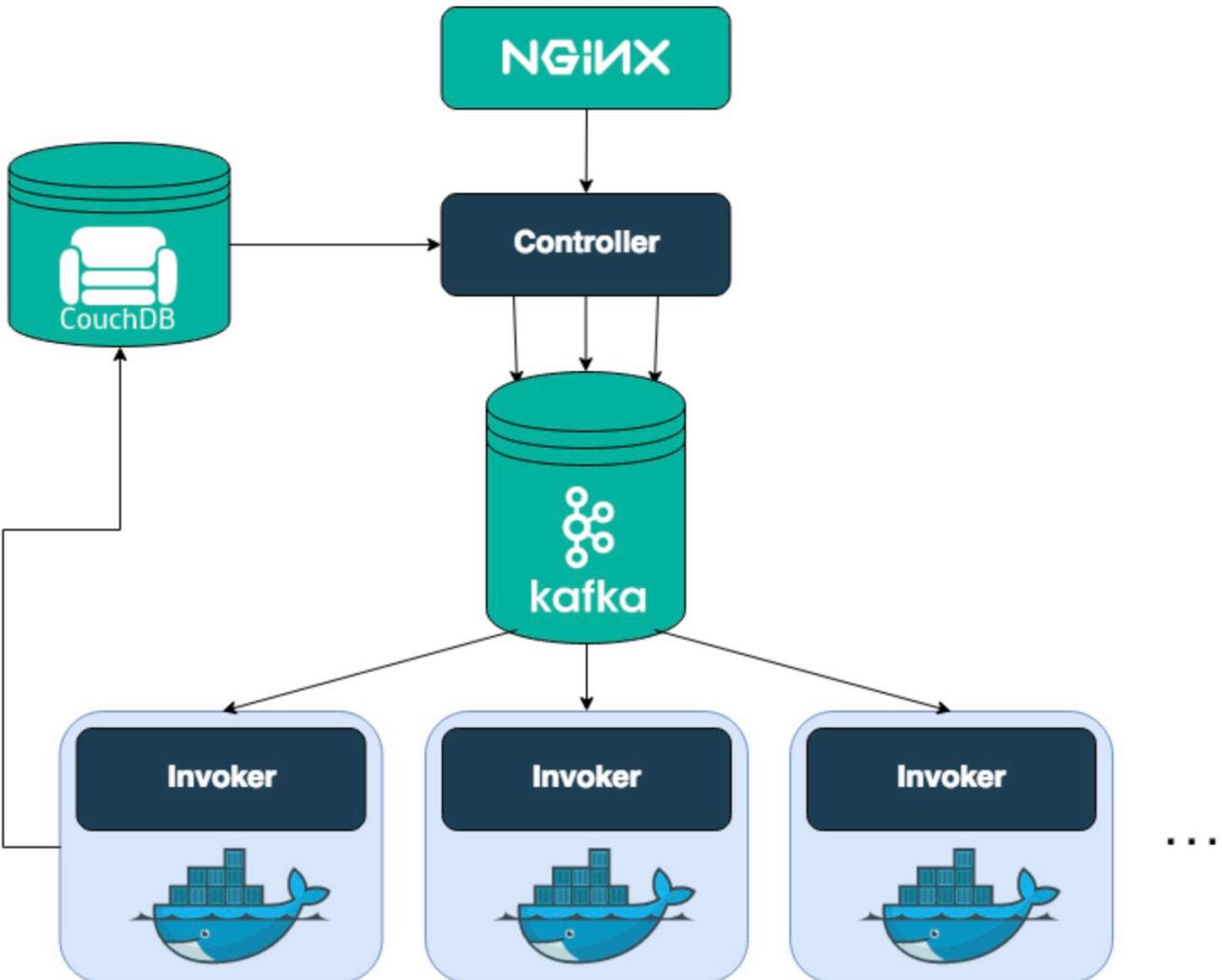
### Event Stream Processing



### 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, including CRUD requests for your entities in OpenWhisk and invocation of actions

## Authentication and Authorization: CouchDB

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

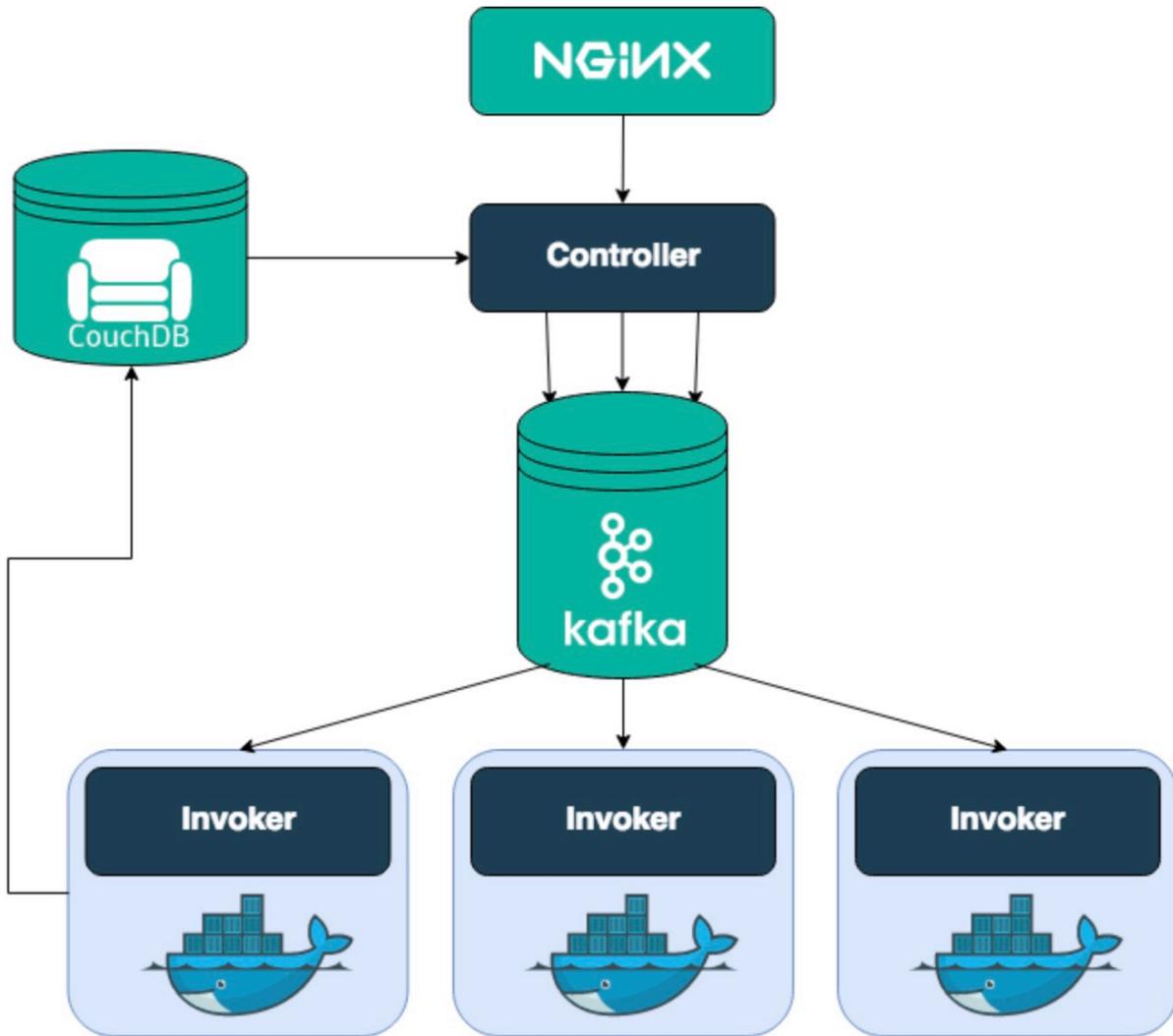
## Getting the action: CouchDB... again

Load the action from the **whisks** database in CouchDB. The record of the action contains mainly the code to execute (shown above) and default parameters that you want to pass to your action, merged with the parameters you included in the actual invoke request. It also contains the resource restrictions imposed on it in execution, such as the memory it is allowed to consume.

## Who's there to invoke the action: Load Balancer

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.

# Apache OpenWhisk



## Please form a line: Kafka

“a high-throughput, distributed, publish-subscribe messaging system”. Controller and Invoker solely communicate through messages buffered and persisted by Kafka.

## Actually invoking the code already: Invoker

The Invoker’s duty is to invoke an action. To execute actions in an isolated and safe way it uses **Docker**.

**Example:** the Invoker will start a Node.js container, inject the code from *myAction*, run it with no parameters, extract the result, save the logs and destroy the Node.js container again.

## Storing the results: CouchDB again

As the result is obtained by the Invoker, it is stored into the **activations** database as an activation. The record contains both the returned result and the logs written. It also contains the start and end time of the invocation of the action.

The system itself mainly consists of only two custom components, the **Controller** and the **Invoker**.

**Everything else is already there, developed by so many people out there in the open-source community.**

# 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

```
import com.google.gson.JsonObject;
import com.google.gson.JsonPrimitive;

import java.util.logging.Logger;

/**
 * Hello FunctionApp
 */
public class FunctionApp {
    protected static final Logger logger = Logger.getLogger("basic");
    public static JsonObject main(JsonObject args) {
        JsonObject response = new JsonObject();
        JsonPrimitive nameArg = args.getAsJsonPrimitive("name");
        String result;
        if (nameArg == null) {
            result = "Hello! Welcome to OpenWhisk";
        } else {
            result = "Hello " + nameArg.getAsString() + " Welcome to OpenWhisk";
        }
        response.addProperty("greetings", result);

        logger.info("invoked with params:");
        return response;
    }
}
```

Entry point

Parameters

Return Object

# Apache OpenWhisk – create an action

```
<project
  xmlns="http://maven.apache.org/POM/4.0.0"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/maven-v4_0_0.xsd">
  <modelVersion>4.0.0</modelVersion>
  <groupId>com.example</groupId>
  <artifactId>hello-world-java</artifactId>
  <version>1.0-SNAPSHOT</version>
  <url>https://openwhisk.apache.org/</url>
  <properties>
    <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
    <maven.compiler.source>1.8</maven.compiler.source>
    <maven.compiler.target>1.8</maven.compiler.target>
    <gson.version>2.8.2</gson.version>
  </properties>
  <dependencies>
    <dependency>
      <groupId>com.google.code.gson</groupId>
      <artifactId>gson</artifactId>
      <version>${gson.version}</version>
    </dependency>
    <dependency>
      <groupId>junit</groupId>
      <artifactId>junit</artifactId>
      <version>4.12</version>
      <scope>test</scope>
    </dependency>
  </dependencies>
  <build>
    <finalName>hello-world-java</finalName>
    <plugins>
      <!-- This helps in packaging the function dependencies as uber-jar-->
      <plugin>
        <groupId>org.apache.maven.plugins</groupId>
        <artifactId>maven-shade-plugin</artifactId>
        <version>3.1.0</version>
        <executions>
          <execution>
            <phase>package</phase>
            <goals>
              <goal>shade</goal>
            </goals>
          </execution>
        </executions>
      </plugin>
    </plugins>
  </build>
</project>
```

gson Dependency

JAR

# 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



IBM Code

# Apache OpenWhisk – create an action

```
~/D/c/colorado-spring-openwhisk ➜ * ➜ serverless-java-mini-workshop ➤ tree src/
```

```
src/
└── main
    └── java
        └── com
            └── example
                └── FunctionApp.java
└── test
    └── java
        └── com
            └── example
                └── FunctionAppTest.java
```

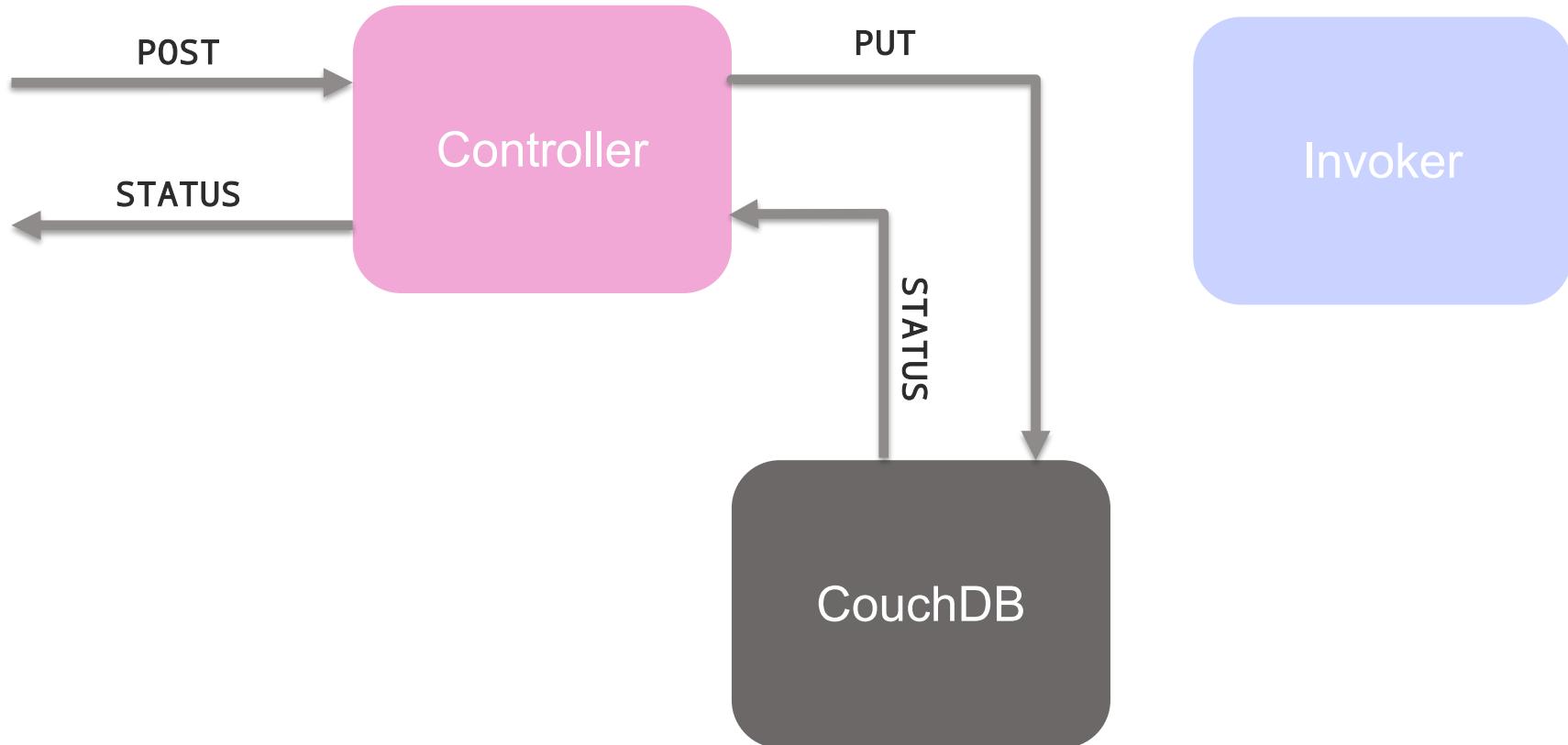
```
~/D/c/colorado-spring-openwhisk ➜ * ➜ serverless-java-mini-workshop ➤ wsk action list -i
actions
```

```
~/D/c/colorado-spring-openwhisk ➜ * ➜ serverless-java-mini-workshop ➤ wsk action create helloJava target/hello-world-java.jar \
--main com.example.FunctionApp -i
ok: created action helloJava
```

```
~/D/c/colorado-spring-openwhisk ➜ * ➜ serverless-java-mini-workshop ➤ wsk action list -i
actions
/guest/helloJava
private java
```

# Apache OpenWhisk – create an action

```
wsk action create helloJava target/hello-world-java.jar  
--main com.example.FunctionApp -i
```



# Apache OpenWhisk – invoke an action

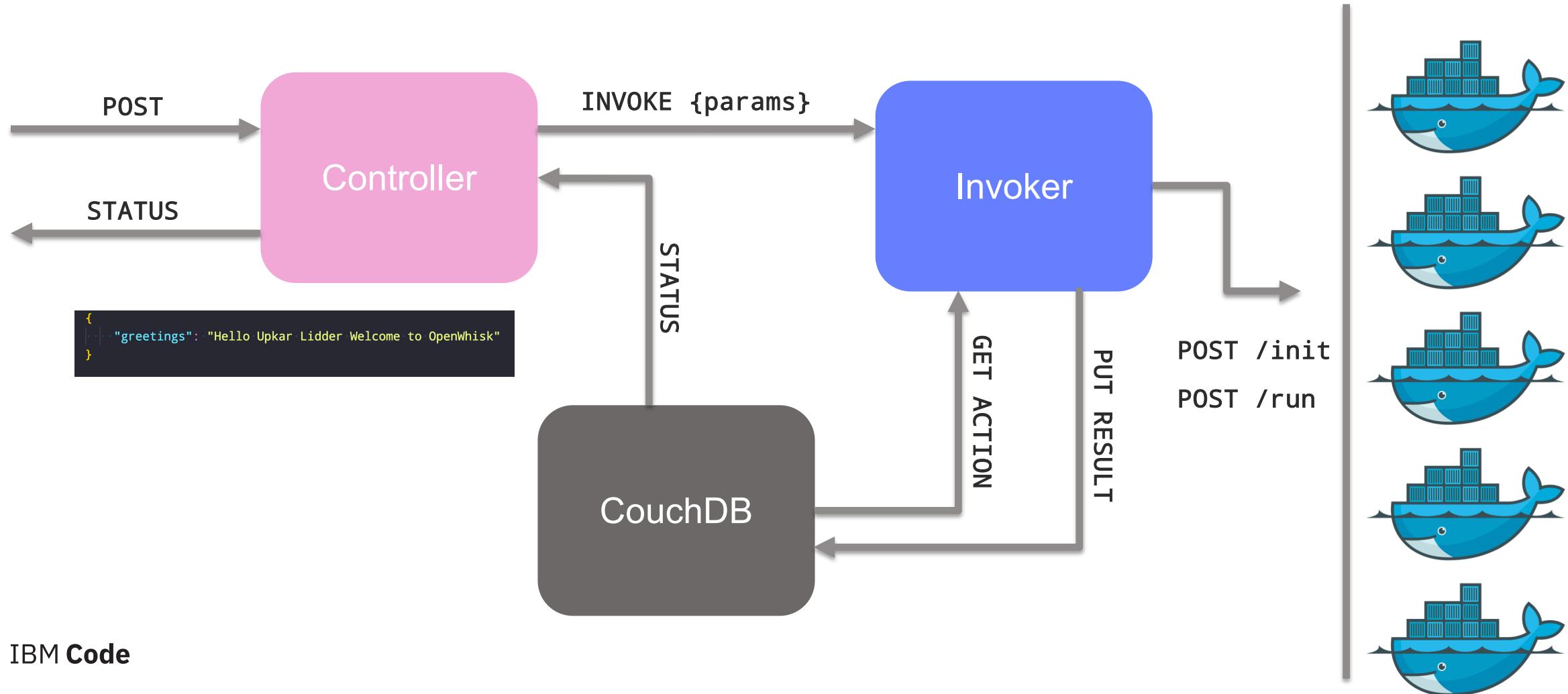
```
~/D/c/colorado-spring-openwhisk ✘ * ➔ serverless-java-mini-workshop ➔ wsk action invoke -ivr helloJava -p name "Upkar Lidder"
REQUEST:
[POST] https://localhost/api/v1/namespaces/_/actions/helloJava?blocking=true&result=true
Req Headers
{
  "Authorization": [
    "Basic MjNiYzQ2YjEtNzFmNi00ZWQ1LThjNTQtODE2YWE0ZjhjNTAy0jEyM3pPM3haQ0xyTU42djJCS0sxZFhZRnBYbFBrY2NPRnFtMTJDZEFzTWdSVTRWck5a0Wx5R1ZDR3VNREdJd1A="
  ],
  "Content-Type": [
    "application/json"
  ],
  "User-Agent": [
    "CloudFunctions-CLI/1.0 (2018-07-12T22:20:03+00:00) darwin amd64"
  ]
}
Req Body
{"name": "Upkar Lidder"}

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": [
    "55"
  ],
  "Content-Type": [
    "application/json"
  ],
  "Date": [
    "Mon, 03 Dec 2018 06:32:36 GMT"
  ],
  "Server": [
    "openresty/1.13.6.2"
  ],
  "X-Openwhisk-Activation-Id": [
    "8ef60362830641ddb60362830641dd7b"
  ],
  "X-Request-Id": [
    "b9C9bdibQFNmxP27NO4yLYxUvSjhK839"
  ]
}
Response body size is 55 bytes
Response body received:
{"greetings": "Hello Upkar Lidder Welcome to OpenWhisk"}
{
  "greetings": "Hello Upkar Lidder Welcome to OpenWhisk"
}

{...} "greetings": "Hello Upkar Lidder Welcome to OpenWhisk"
```

# Apache OpenWhisk – invoke an action

```
wsk action invoke -ivr helloJava -p name "Upkar Lidder"
```



# IBM Cloud Functions

	Open source	Hosted service
Serverless engine	<a href="#">Apache OpenWhisk</a>	<a href="#">IBM Cloud Functions</a>
API Gateway	<a href="#">LoopBack</a>	<a href="#">IBM API Gateway</a>
Databases	<a href="#">Apache CouchDB</a> <a href="#">MySQL</a>	<a href="#">IBM Cloudant</a> <a href="#">IBM Compose</a>
Message streams	<a href="#">Apache Kafka</a>	<a href="#">IBM Message Hub</a>



IBM Code

## 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

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	not 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

# 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.



# 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 +

# 1500 Drones Are Ready To Fly

## IBM Developer Drone Challenge

Challenge runs from November 12 through December 16 and those 18 years of age or older residing in the US and Canada are eligible to enter

Enter at <https://developer.ibm.com/contest>

Once per week during the 5 weeks of the contest a random drawing  
Will be held to determine the winners (watch for the drawing on Twitch)

Winners will receive a [DJI Tello programmable drone](#),  
an [IBM Developer T-shirt](#), and an [IBM Developer laptop sticker](#)

Winners will also receive a special code to unlock more challenges and given the opportunity to contribute back to the challenge



#IBMDroneDrop

# Thank you!

Upkar Lidder  
Developer Advocate, IBM

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

# IBM Code

# 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.