Introduction to Serverless Computing

Upkar Lidder
Developer Advocate, IBM

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

IBM **Code**

What is Serverless?

What is Serverless?

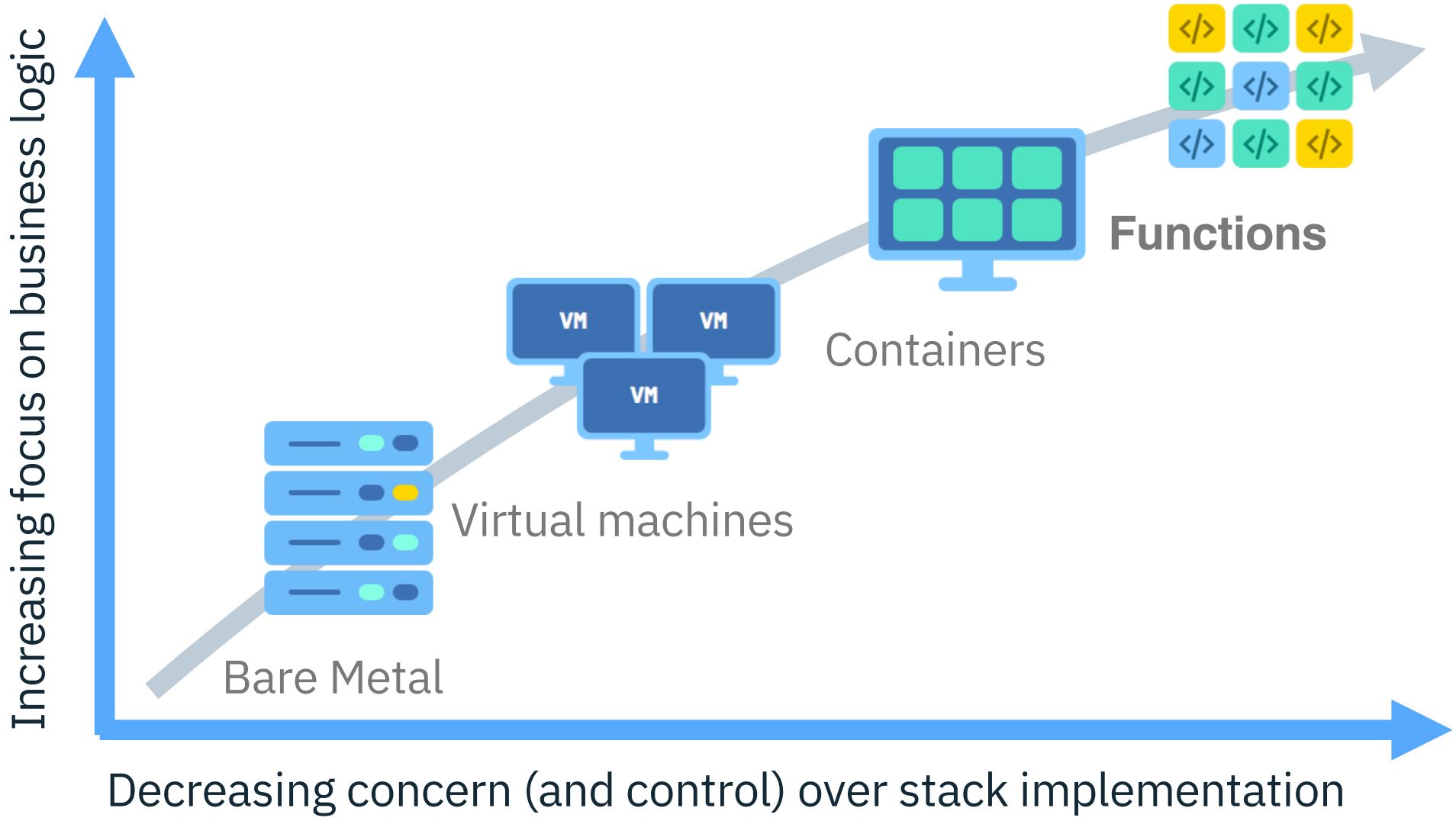
Serverless computing refers to the concept of building and running applications that do not require server management.

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?



Use Cases

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

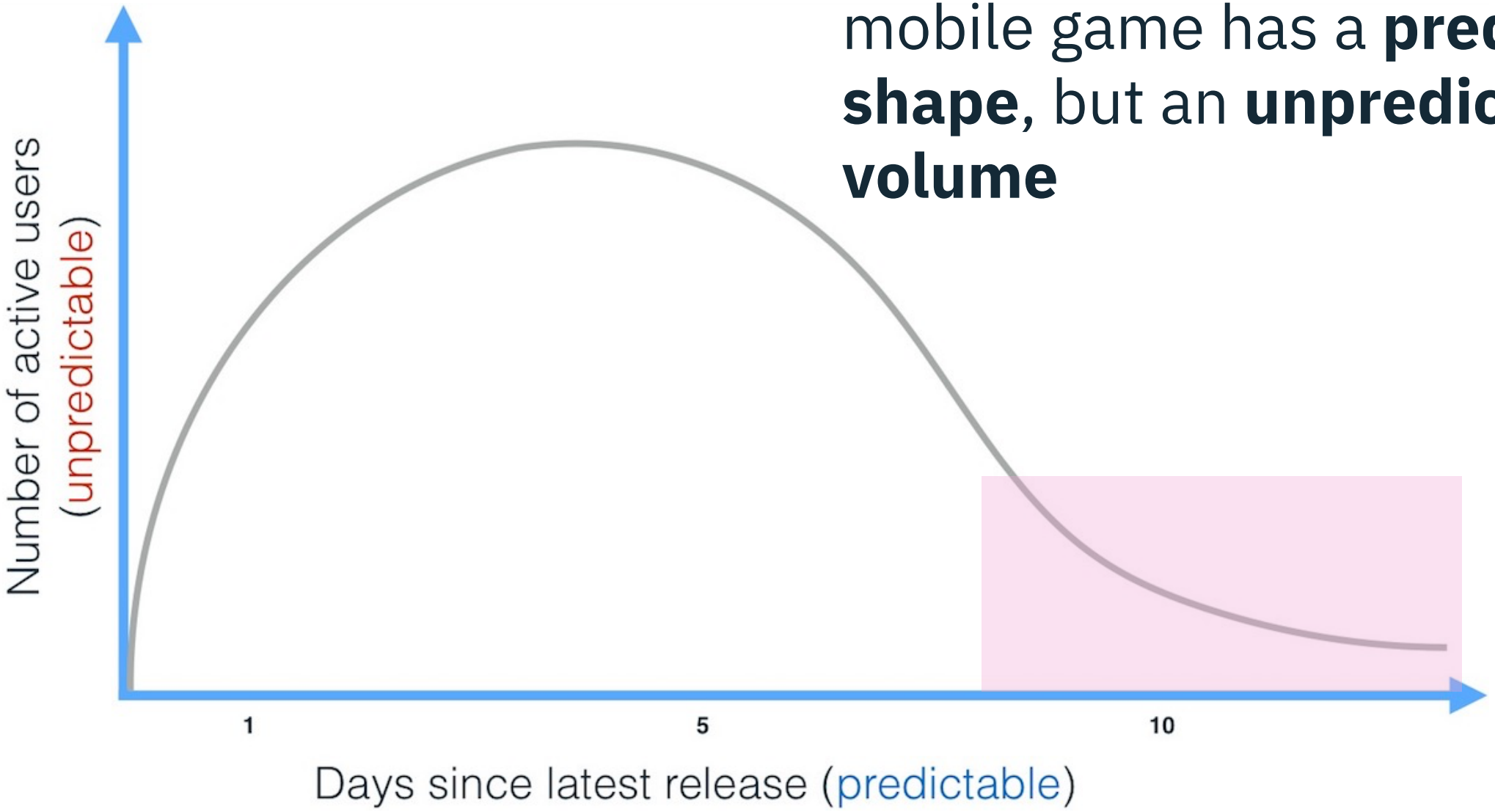
Infrequent or has sporadic demand, with large, unpredictable variance in scaling requirements

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

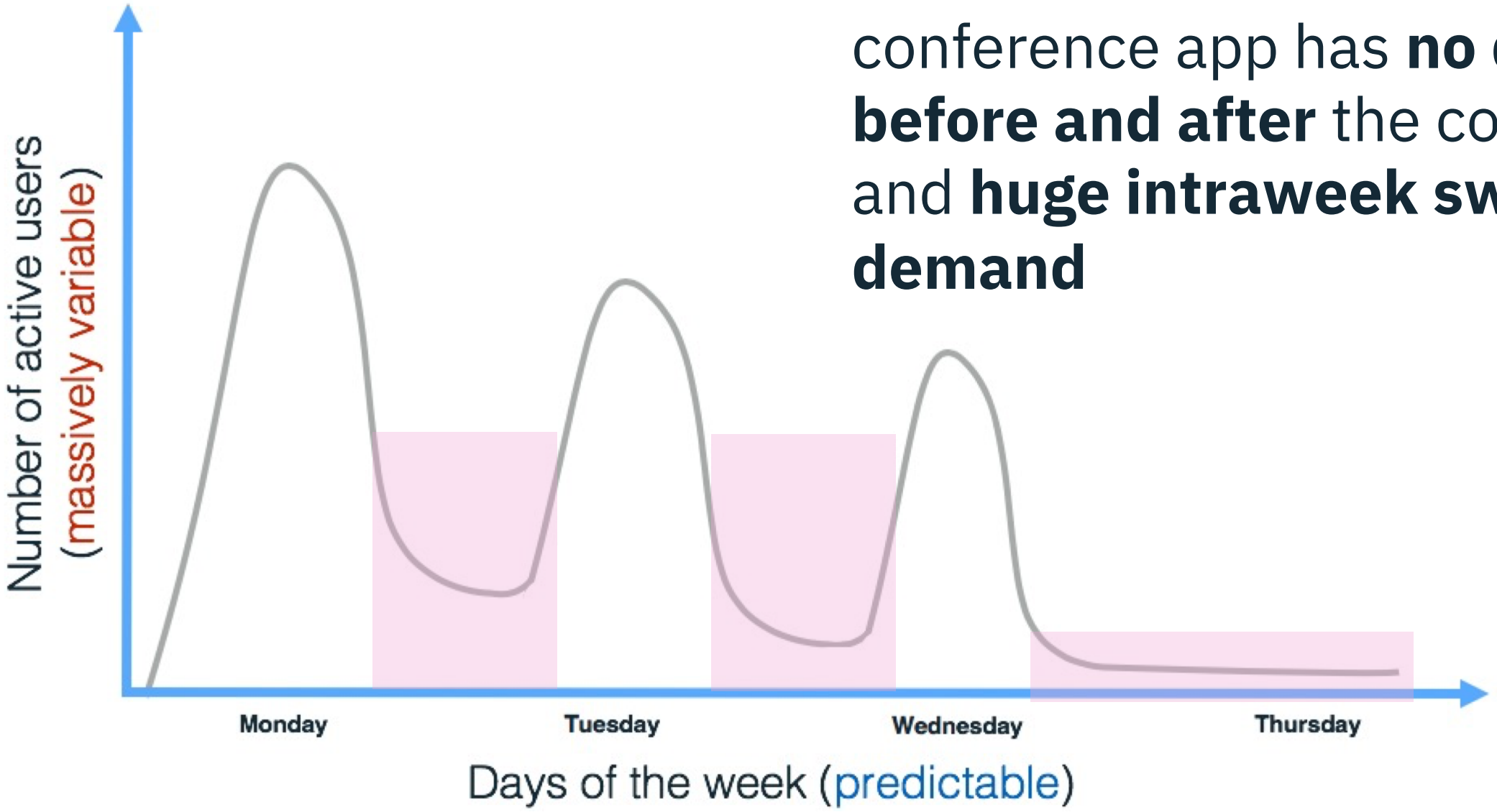
Why Serverless?

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

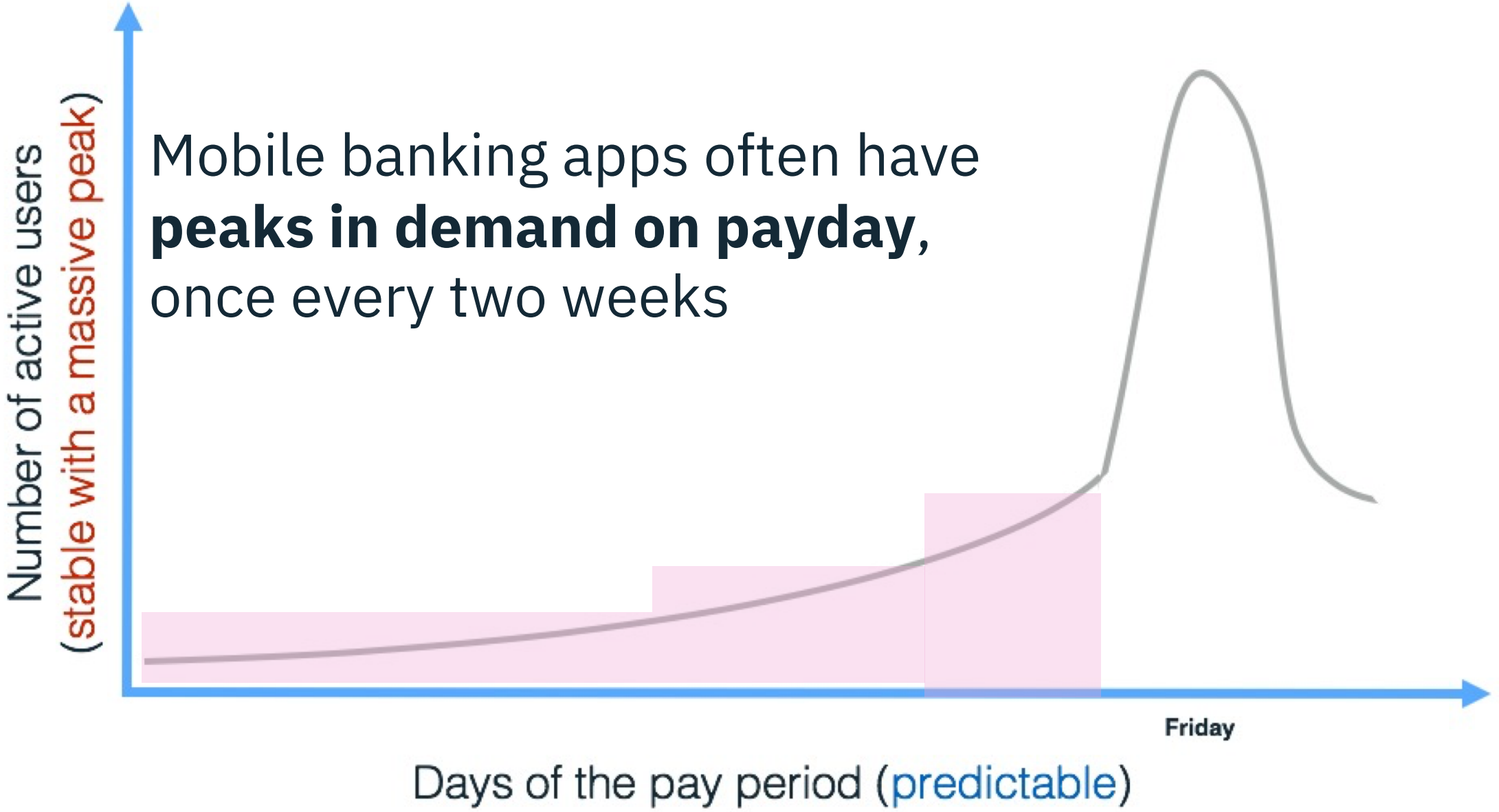


Why Serverless?

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



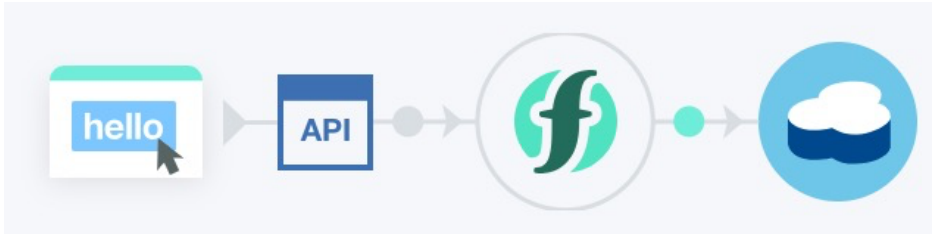
Why Serverless?



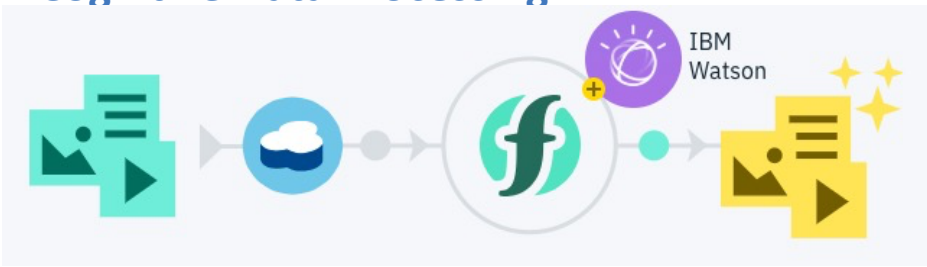
Mobile banking apps often have **peaks in demand on payday**, once every two weeks

Use cases

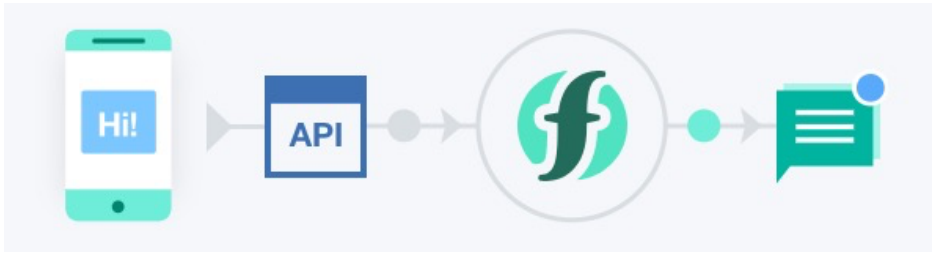
Serverless Backends



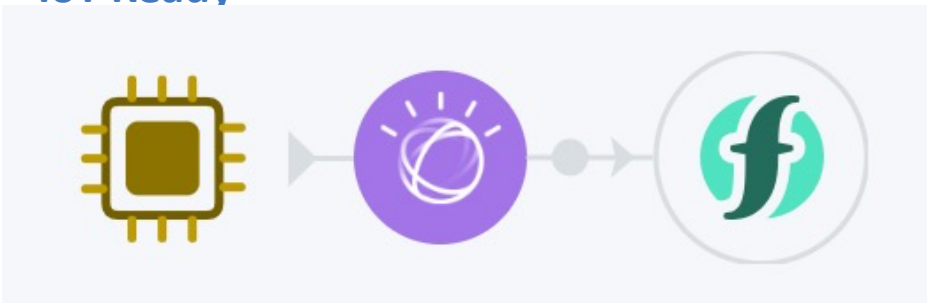
Cognitive Data Processing



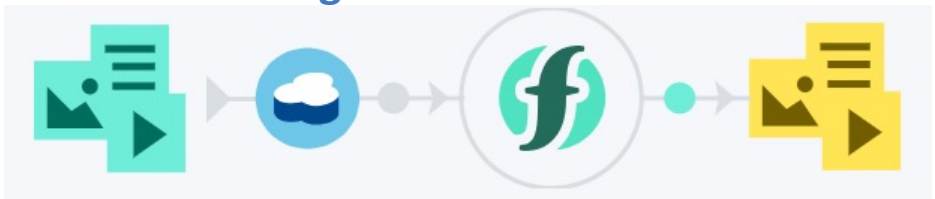
Mobile Backend



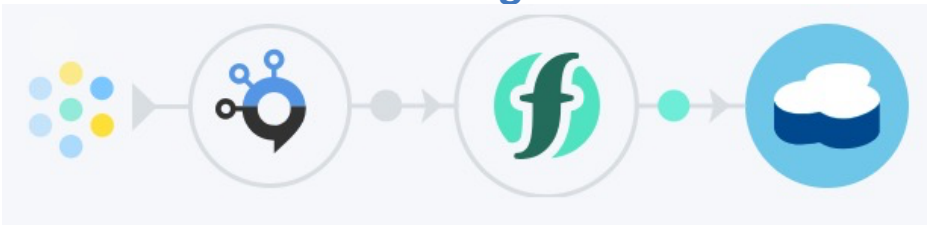
IoT Ready



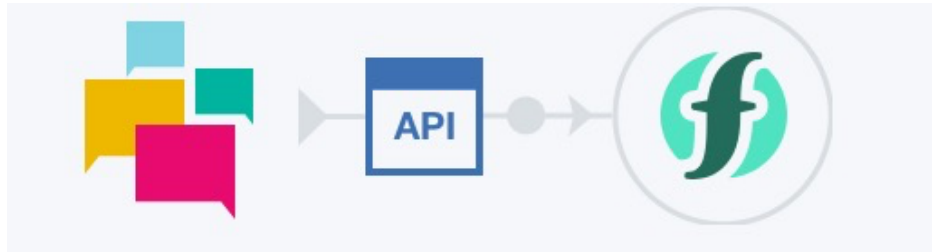
Data Processing



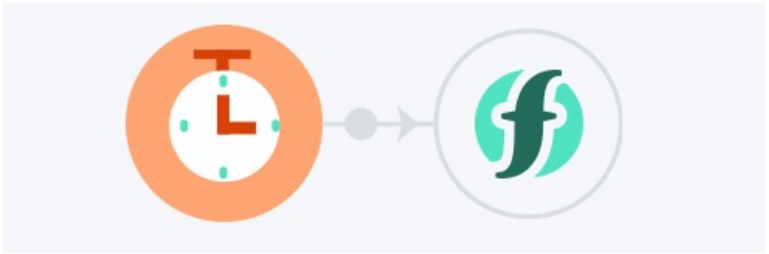
Event Stream Processing



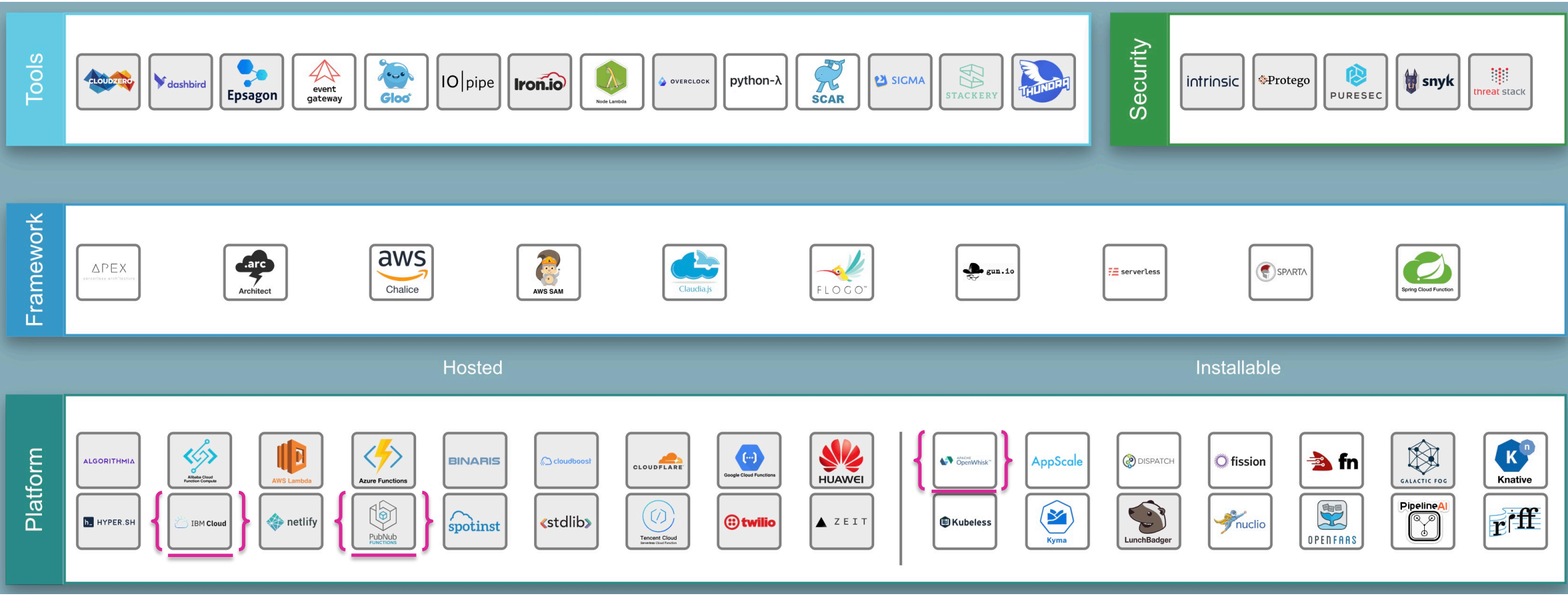
Conversational Scenarios



Scheduled Tasks



Server Landscape (Cloud Native Computing Foundation)



Server Landscape – feature matrix

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

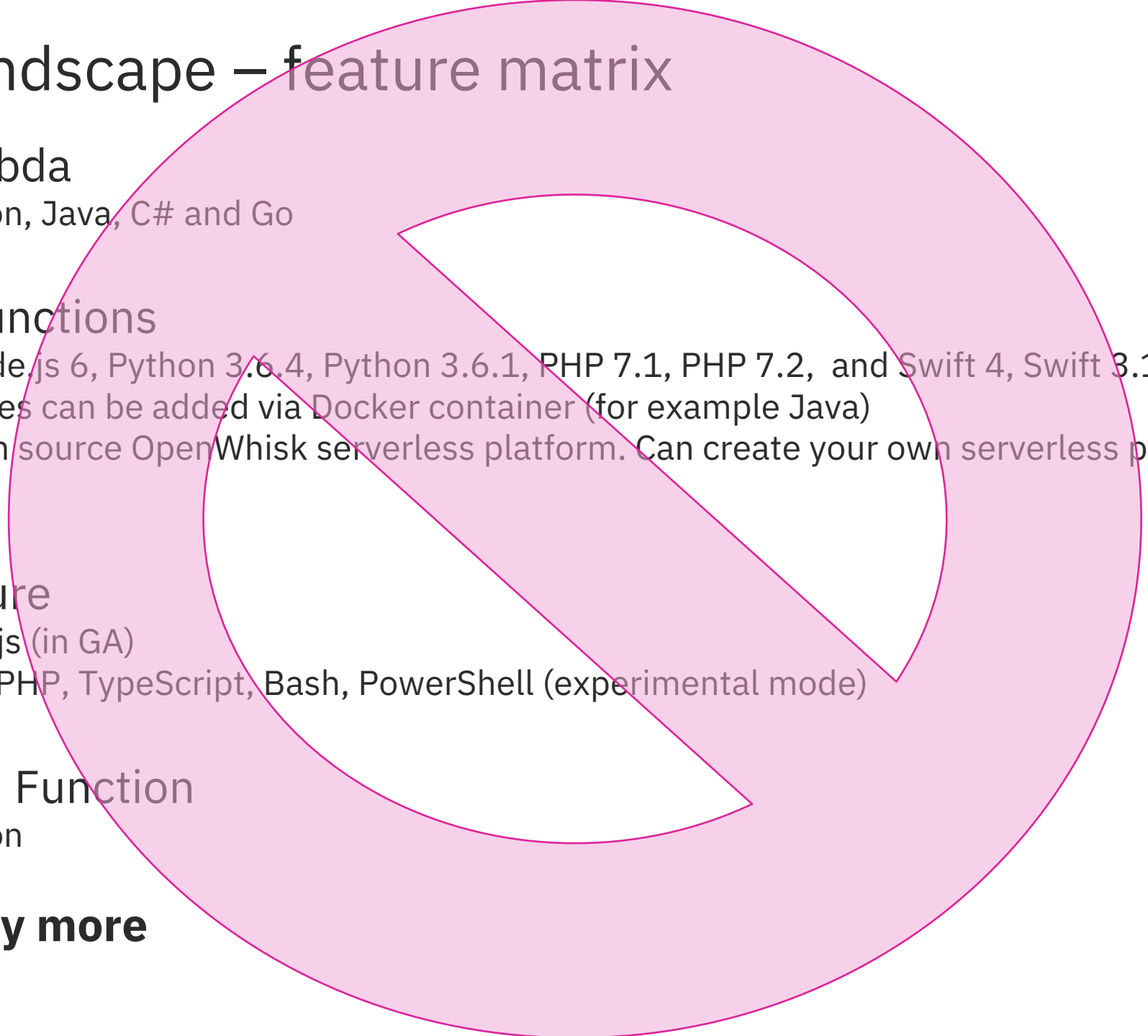
Microsoft Azure

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

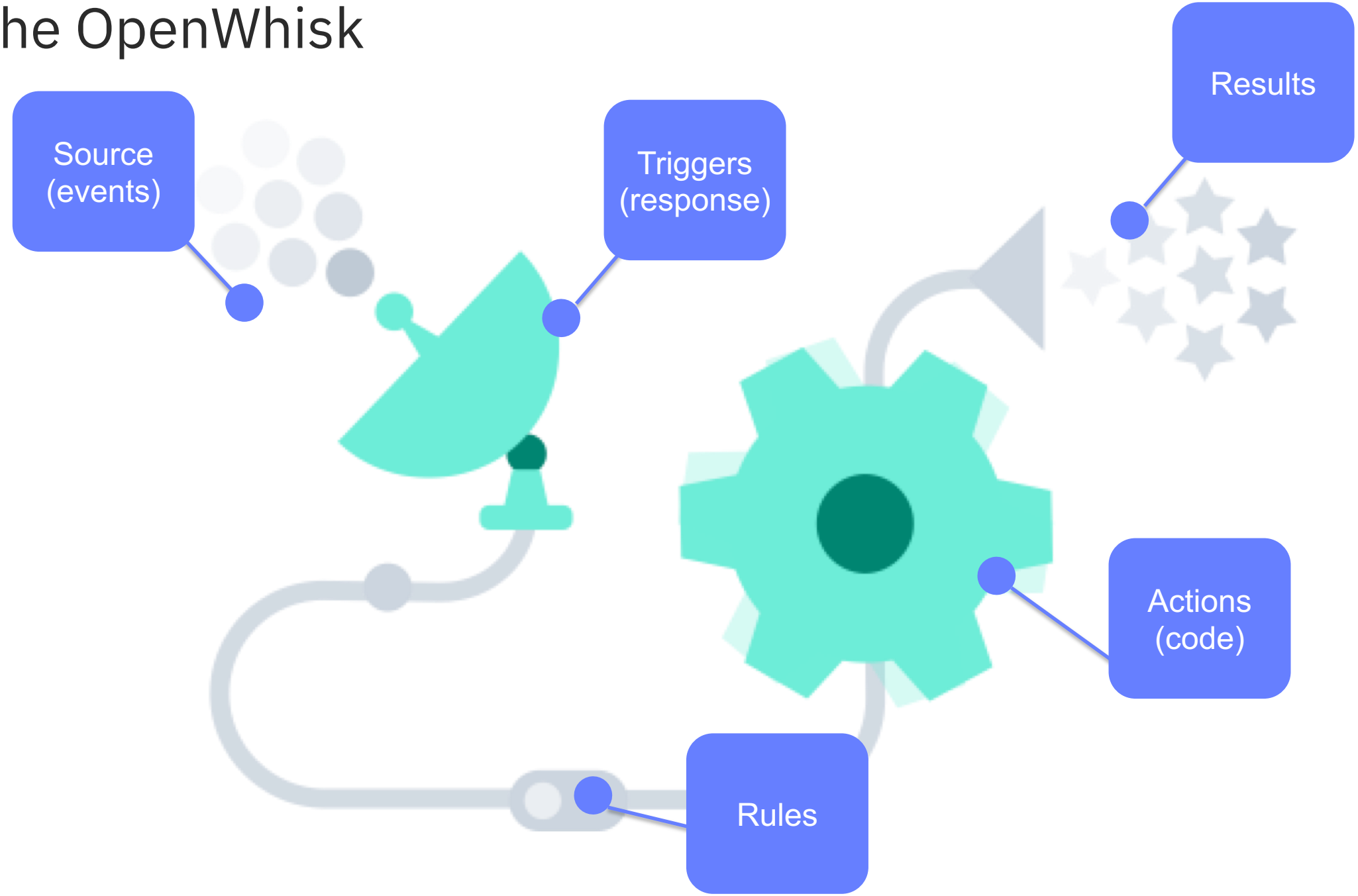
Google Cloud Function

- Node.js, Python

- **Many many more**



Apache OpenWhisk



Apache OpenWhisk

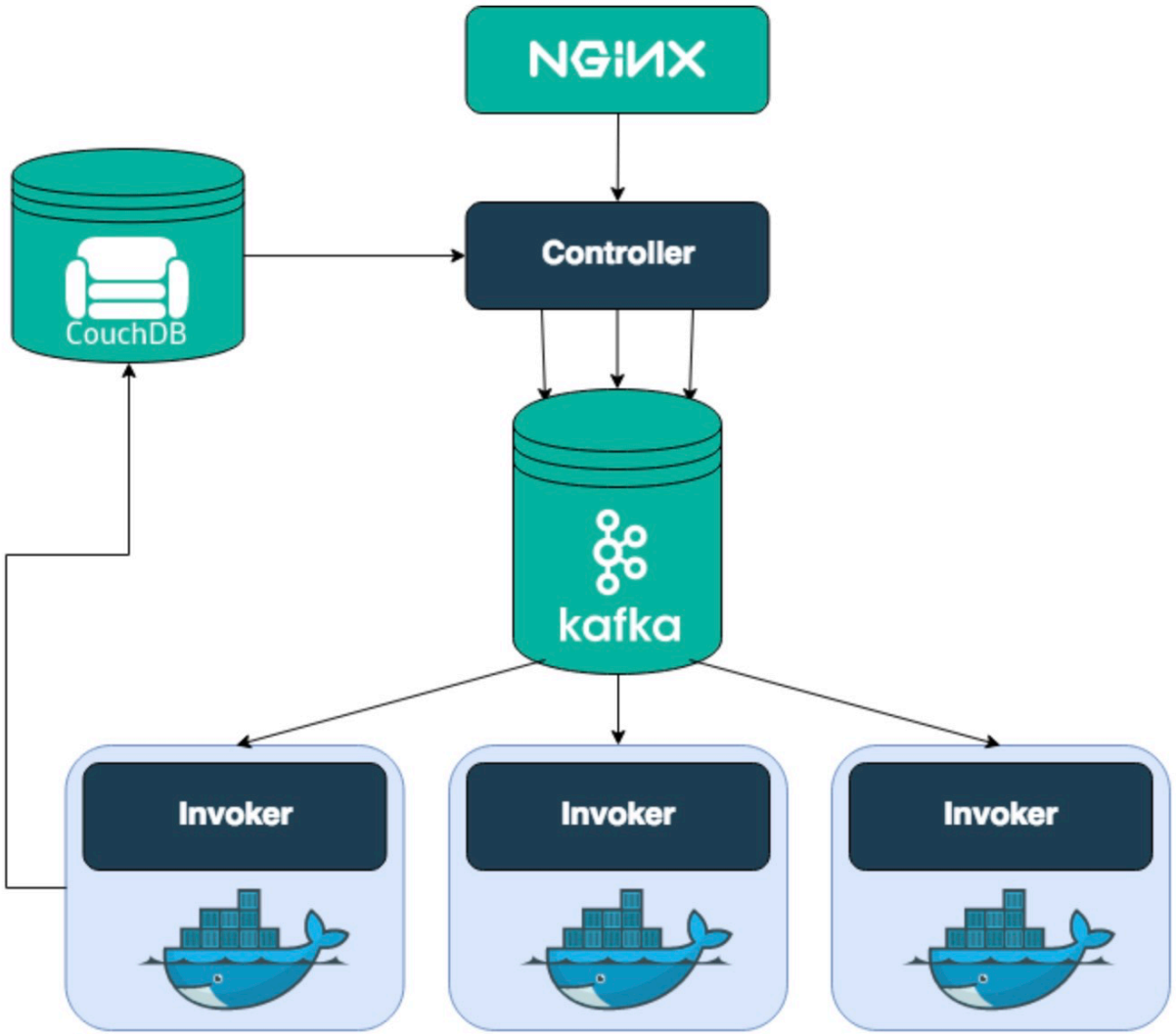
T *A class of events that can occur*

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	…

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
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
a high-throughput, distributed, publish-subscribe messaging system

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

...

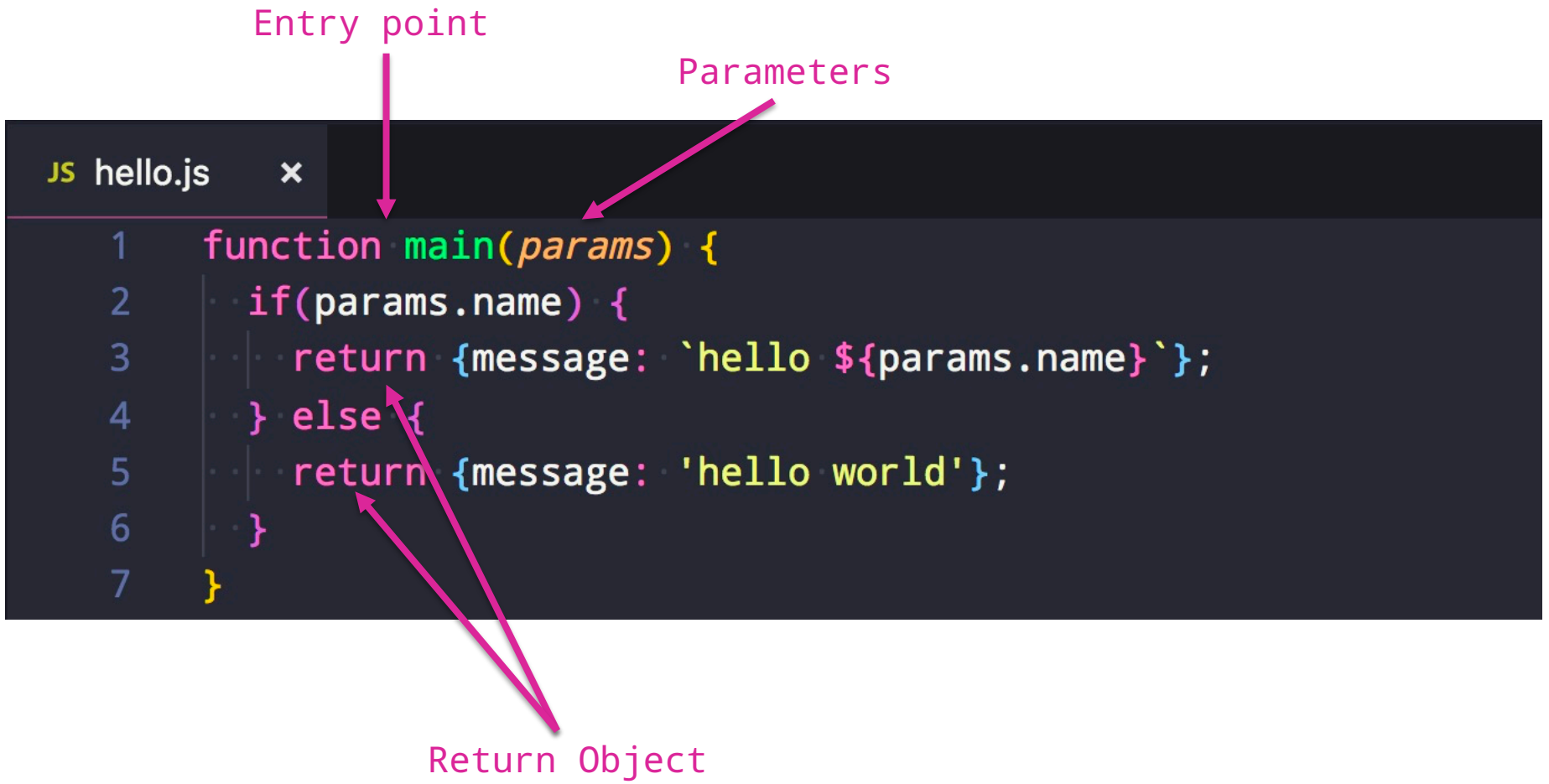
Storing the results: CouchDB again
Store the returned result, logs written, start and end time of the action.

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

Apache OpenWhisk

```
~/D/c/o/incubator-openwhisk-devtools  …  docker-compose  docker ps --format "{{.ID}}: {{.Names}} {{.Image}}"
9c38cfa1cf6a: 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
```

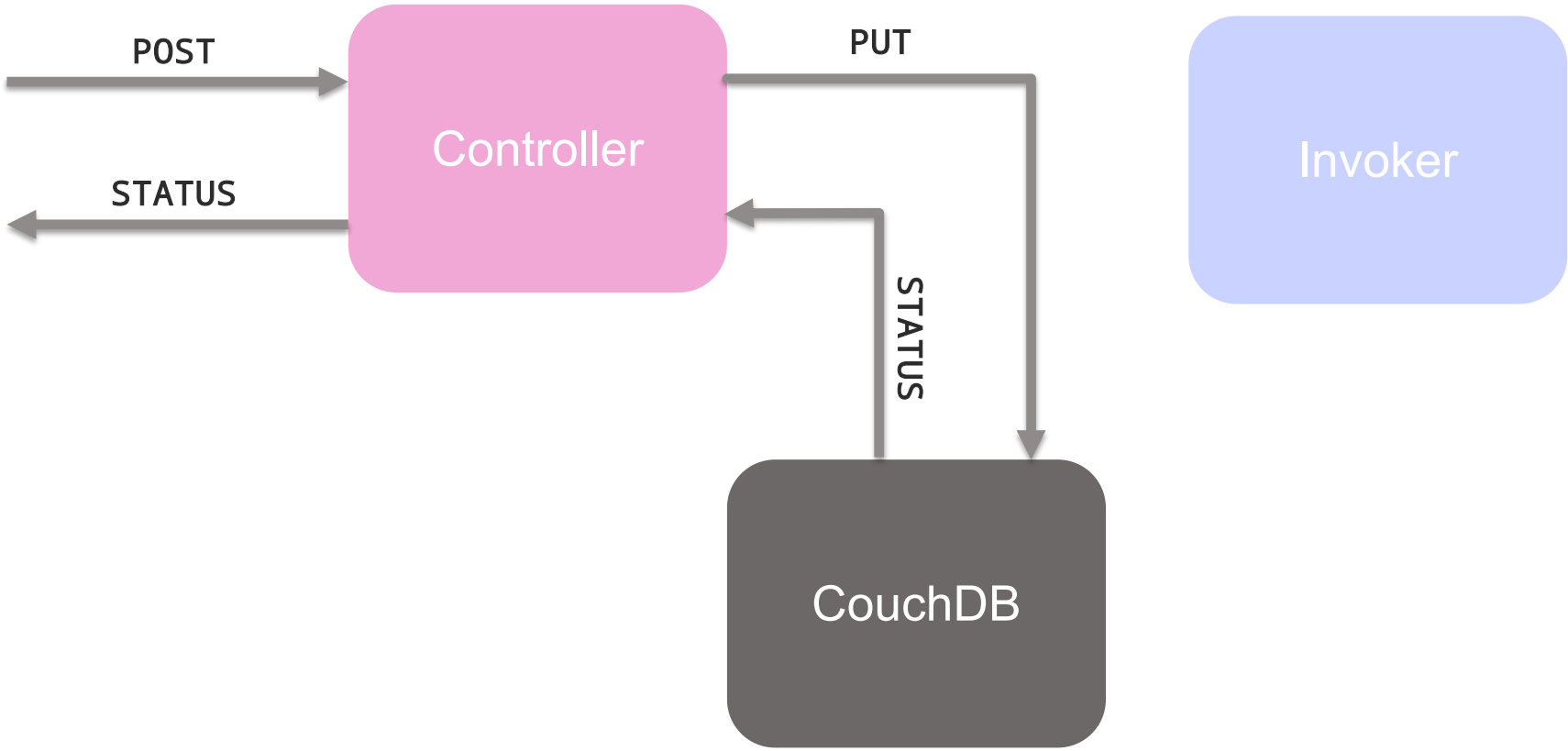
```
~/D/c/openwhisk-example wsk action list -i
actions
```

```
~/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

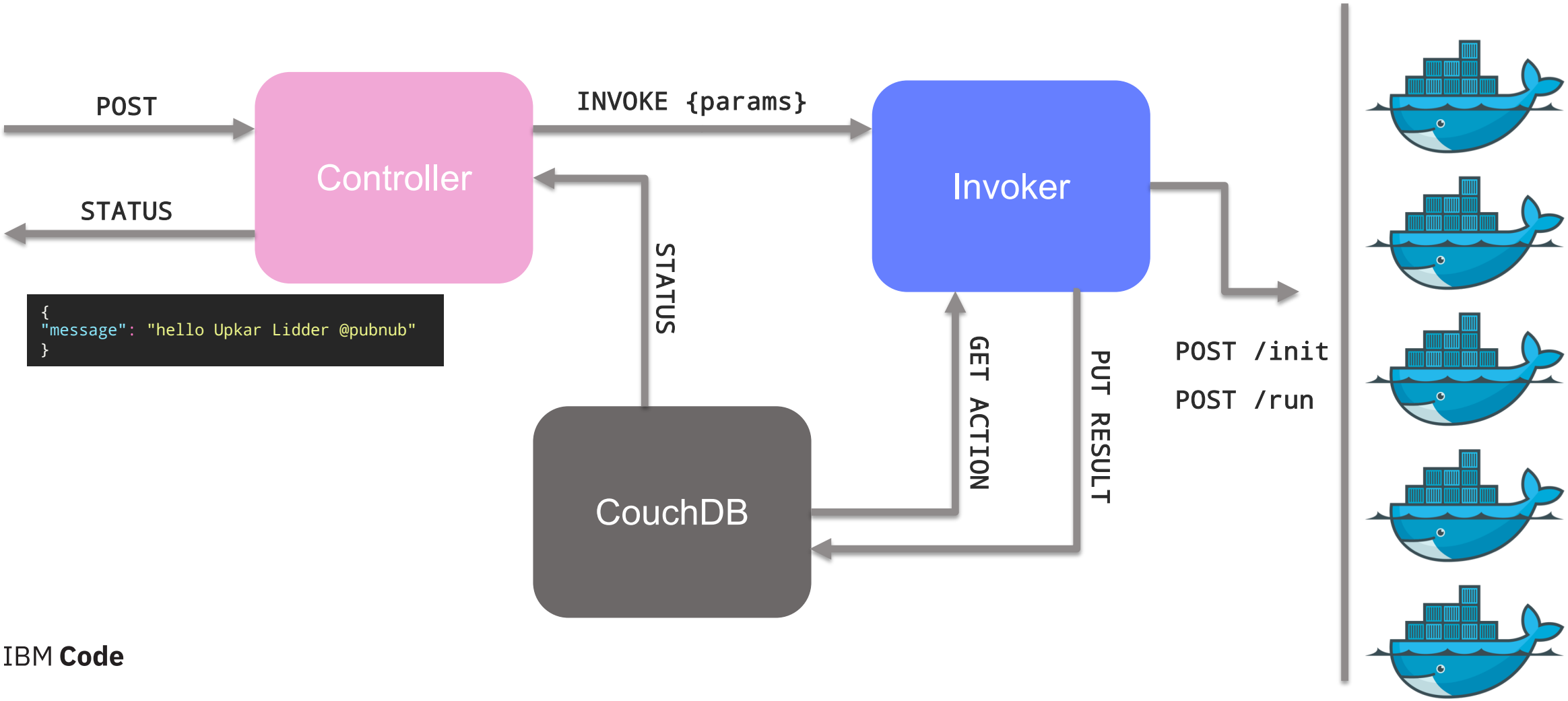
```
~/D/c/openwhisk-example wsk action invoke hello -i -r -v -p name "Upkar Lidder @pubnub"
REQUEST:
[POST]  https://localhost/api/v1/namespaces/guest/actions/hello?blocking=true&result=true
Req Headers
{
  "Authorization": [
    "Basic
  ],
  "Content-Type": [
    "application/json"
  ],
  "User-Agent": [
    "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": [
    "27363aab6d244db9b63aab6d24ddb909"
  ],
  "X-Request-Id": [
    "ZHki96YE5NiUVNCnOCcGeRKmRPkNumXB"
  ]
}
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"
```



Apache OpenWhisk - wsk

```
}
~/D/c/openwhisk-example wsk


      ____      ___                   _    _ _     _     _
     /\   \    / _ \ _ __   ___ _ __ | |  | | |__ (_)___| | __
  /\  /__\   \  | | | '_ \ / _ \ '_ \| |  | | '_ \| / __| |/ /
 /  \____ \  /  | |_| | |_) |  __/ | | | |/\| | | | | \__ \   <
 \   \  /  \/    \___/| .__/ \___|_| |_|__/\__|_| |_|_|___/_|\_\
  \___\/ tm           |_|

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 an 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 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

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/

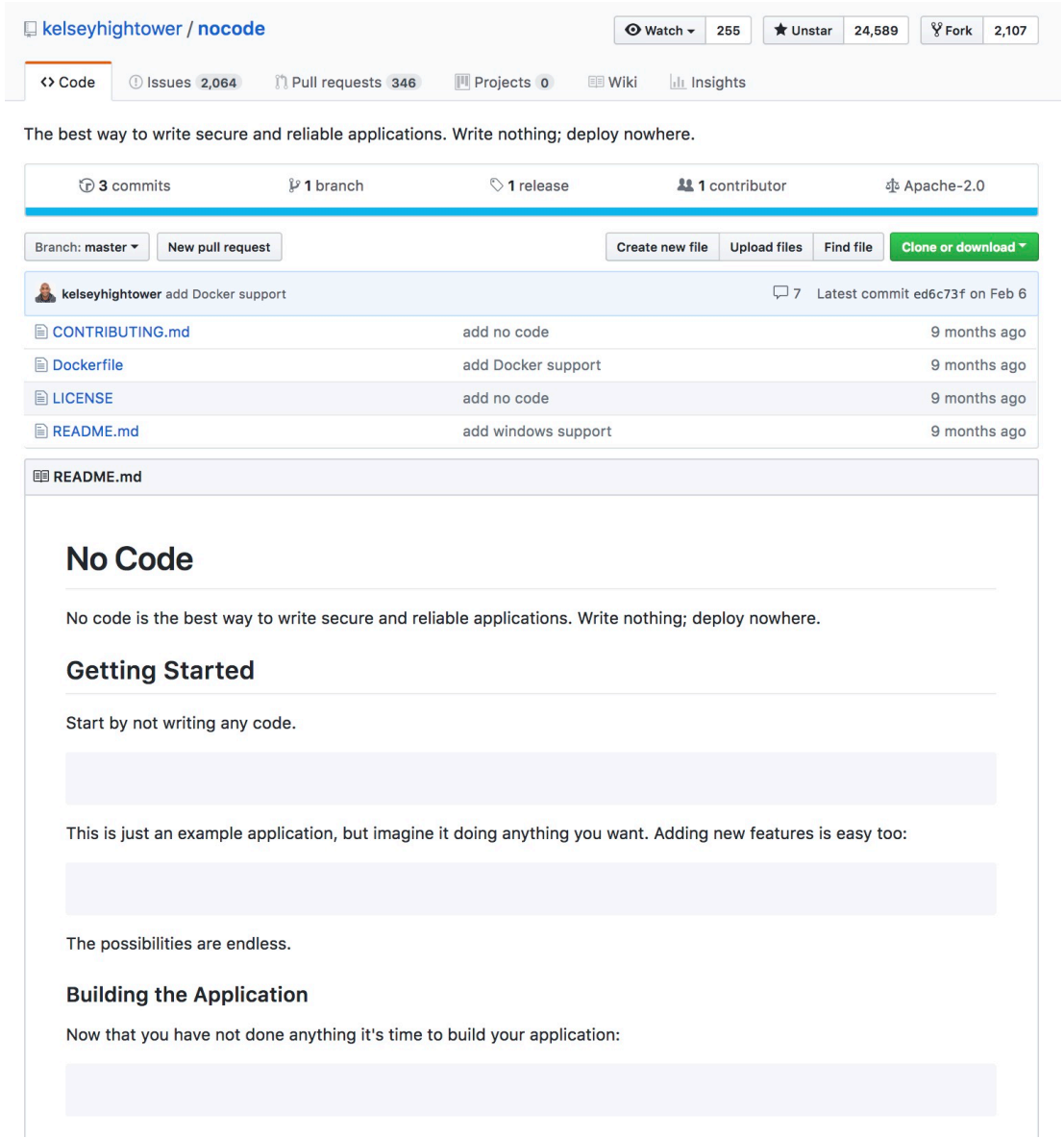
Code Patterns

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

Serverless architecture

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

Future of Serverless ?



kelseyhightower / nocode

Watch 255 | Unstar 24,589 | Fork 2,107

Code | Issues 2,064 | Pull requests 346 | Projects 0 | 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:

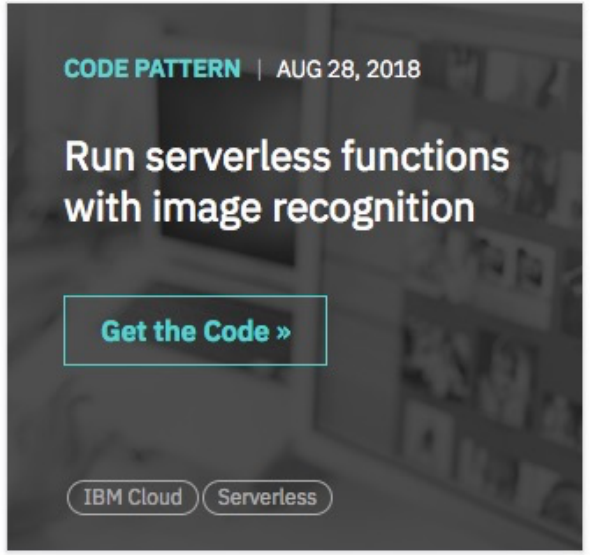
I know right?

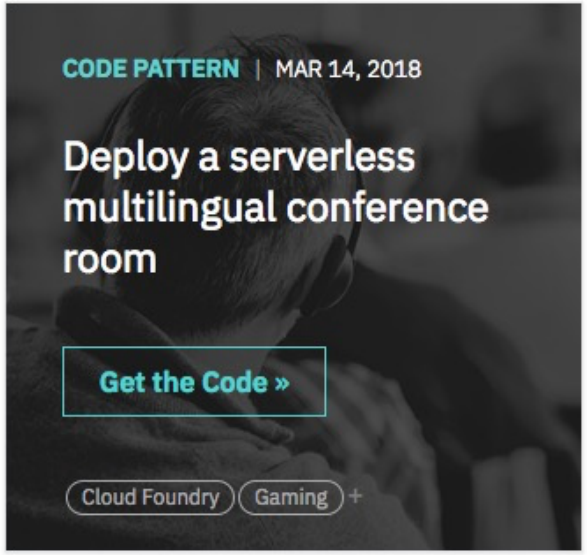
Contributing

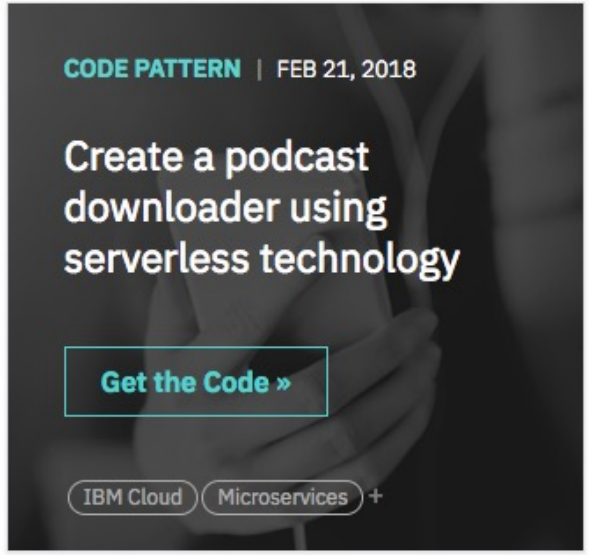
You don't.

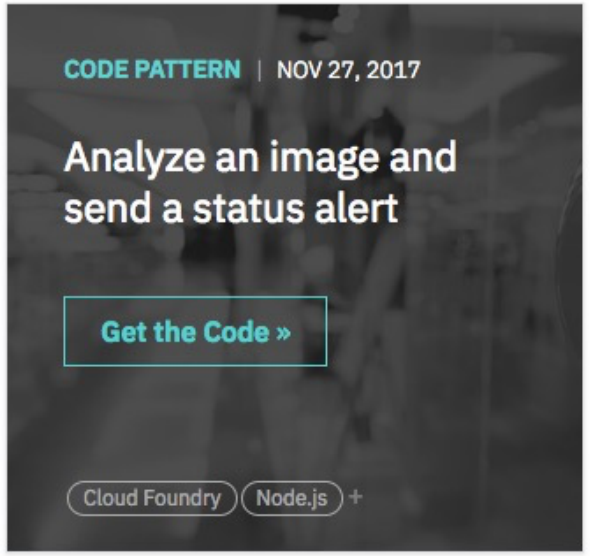
© 2018 GitHub, Inc. Terms Privacy Security Status Help — Contact GitHub Pricing API Training Blog About

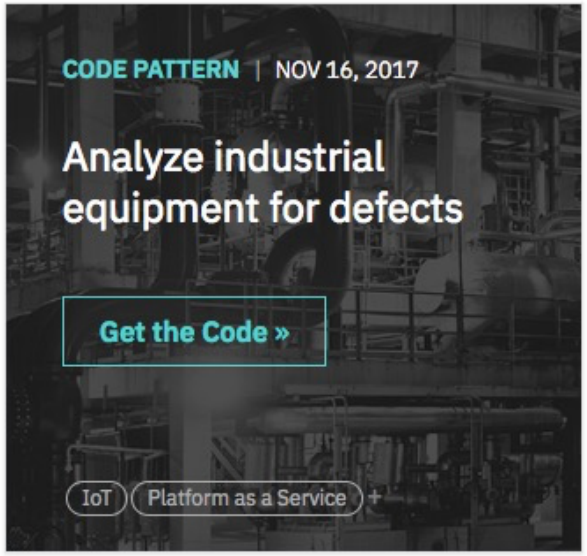
Code Patterns 🛠️

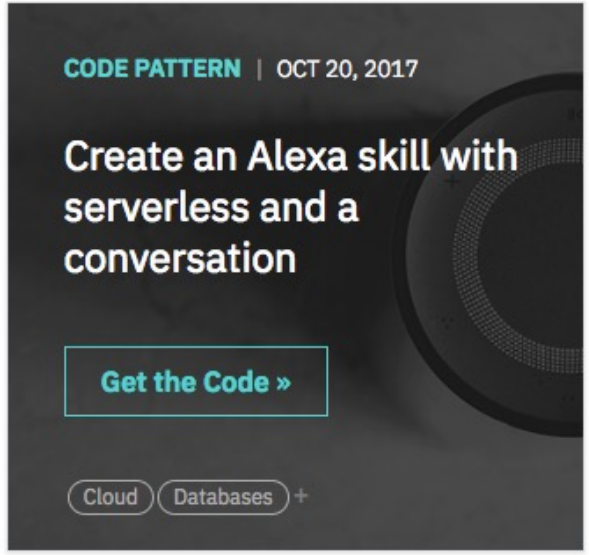












Thank you!

Upkar Lidder
Developer Advocate, IBM

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

IBM **Code**