Serverless apps with Apache OpenWhisk

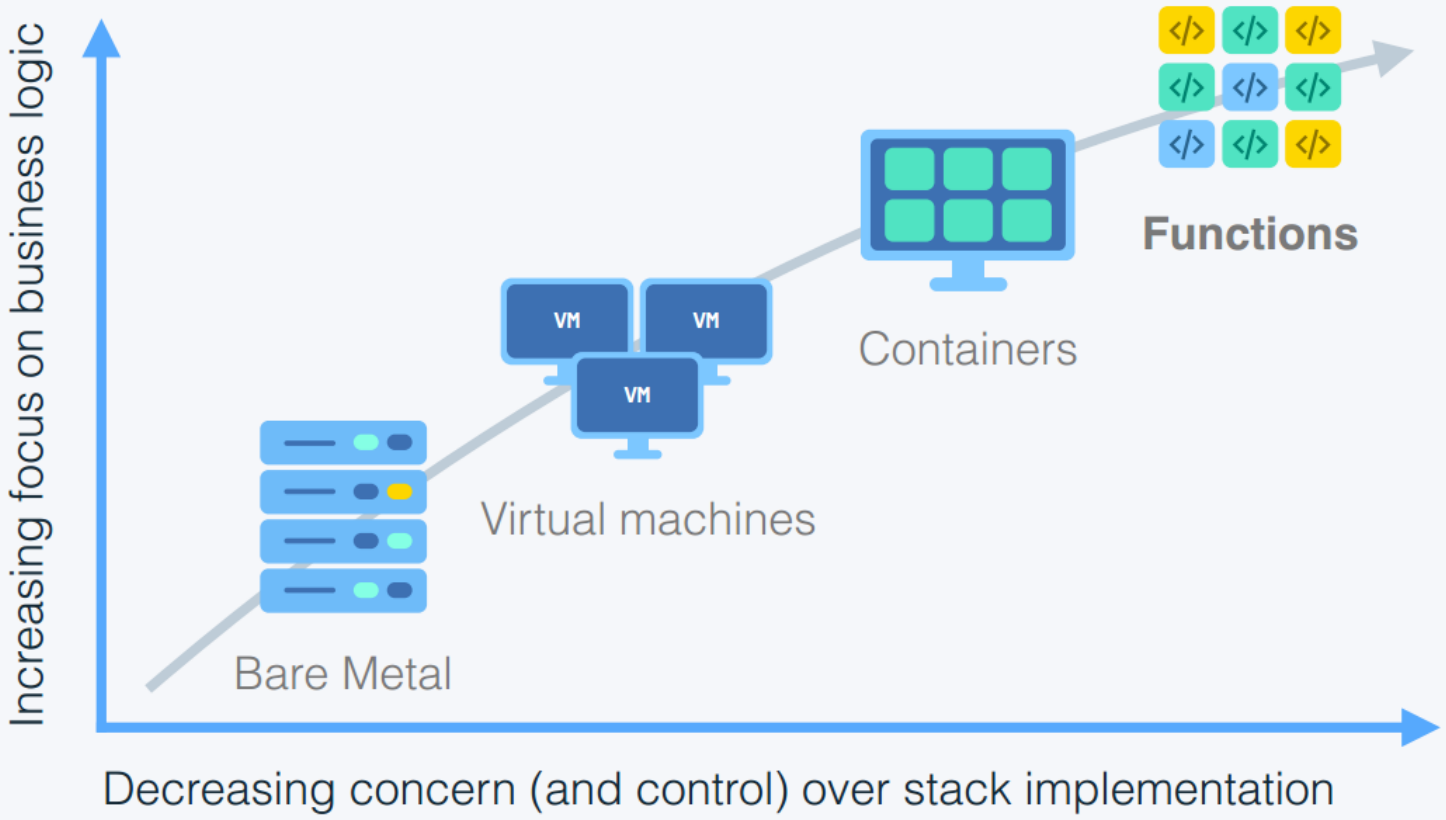
CLOUD NATIVE ● EVENT DRIVEN ● MICROSERVICES

Agenda

- Evolution of Serverless
- Introducing OpenWhisk
- OpenWhisk Architecture
- OpenWhisk & Containers
- Demos & Use cases
- Customers & Partners

What makes serverless, event driven computing so attractive?

Serverless developers focus more on code, less on infrastructure

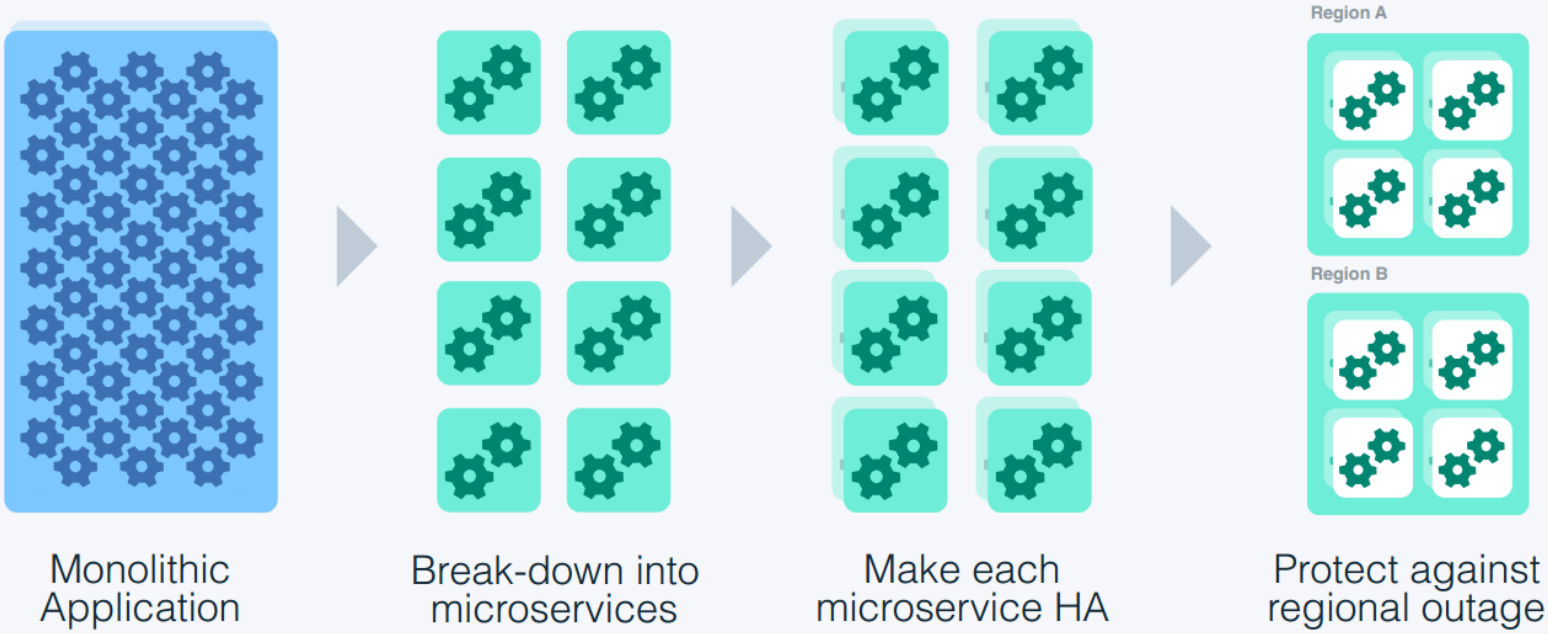


Runs code **only** on-demand on a per-request basis

No servers

Just code

Problem: Microservices can be hard to manage at scale



Serverless can handle many cloud native app 12 Factors

1. Codebase
   One codebase tracked in revision control, many deploys

   Handled by developer

2. Dependencies
   Explicitly declare and isolate dependencies

   Handled by developer, facilitated by platform

3. Configuration
   Store config in the environment

   Handled by platform

4. Backing services
   Treat backing services as attached resources

   Handled by platform

5. Build, release, run
   Strictly separate build and run stages

   Handled by platform

6. Processes
   Execute the app as one or more stateless processes

   Handled by platform

Serverless can handle many cloud native app 12 Factors

7. Port binding
   Export services via port binding

   Handled by platform

8. Concurrency
   Scale out via the process model

   Handled by platform

9. Disposability
   Maximize robustness with fast startup and graceful shutdown

   Handled by platform

10. Dev/prod parity
    Keep development, staging, and production as similar as possible

    Handled by developer

11. Logs
    Tread logs as event streams

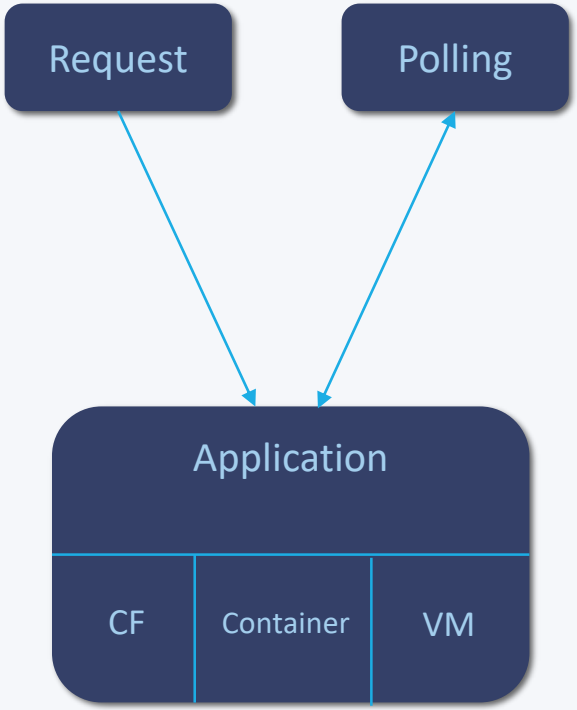
    Handled by platform

12. Admin processes
    Run admin/management tasks as one-off processes

    Handled by developer

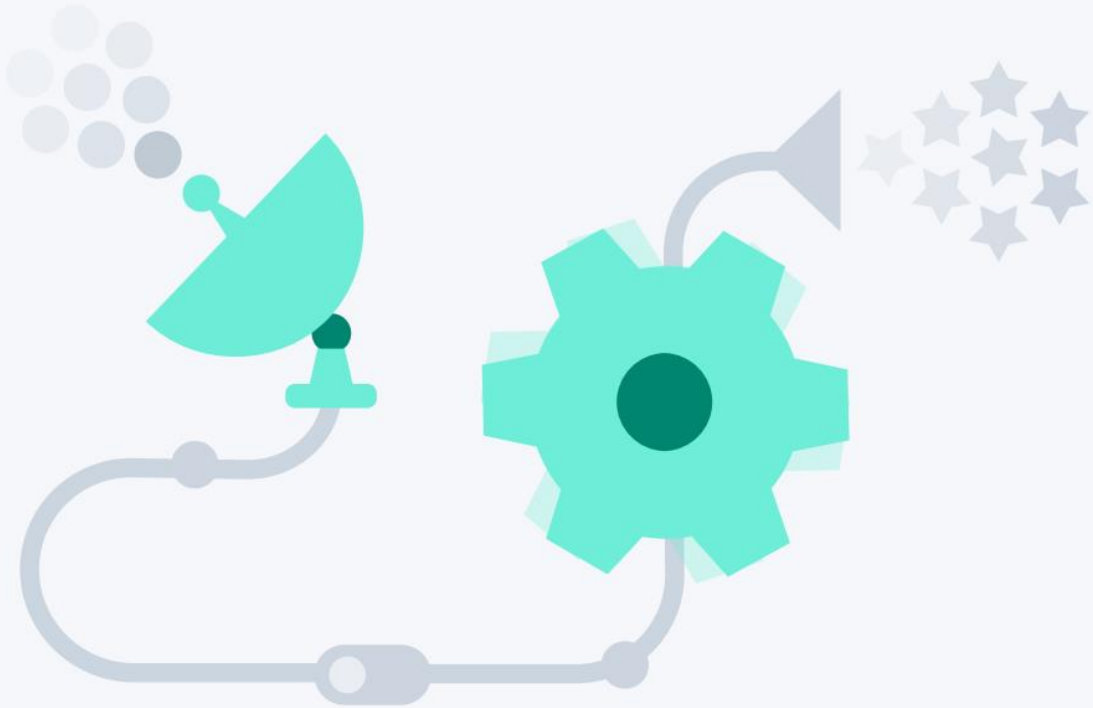
Problem: Programming and pricing models aren't efficient

- Continuous polling needed in the absence of an event driven programming model
- Charged for resources, even when idle
- Worries persist about capacity management



Event-programming model

- Runs code **in response** to events

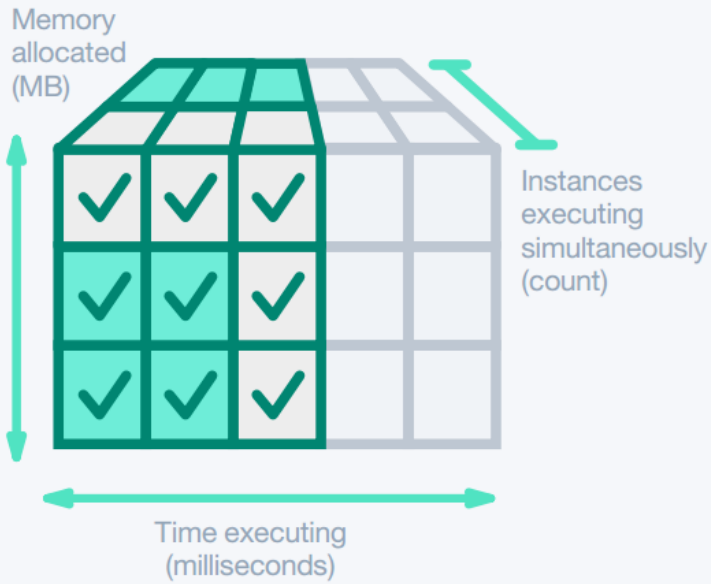


Emerging workloads are a good fit for event-driven programming

- Execute logic in response to database change
- Perform analytics on sensor input messages
- Provide cognitive computing via chatbots
- Schedule tasks performed for a short time
- Invoke autoscaled APIs and mobile backends

New cost models more accurately charge for usage

- **Cloud resource cost** better matches **business value gained**



Technological and business factors make serverless compelling

Platforms evolving to facilitate cloud native design for developers

↓

Growth of event driven workloads that need automated scale

↓

Cost models getting more granular and efficient

↓

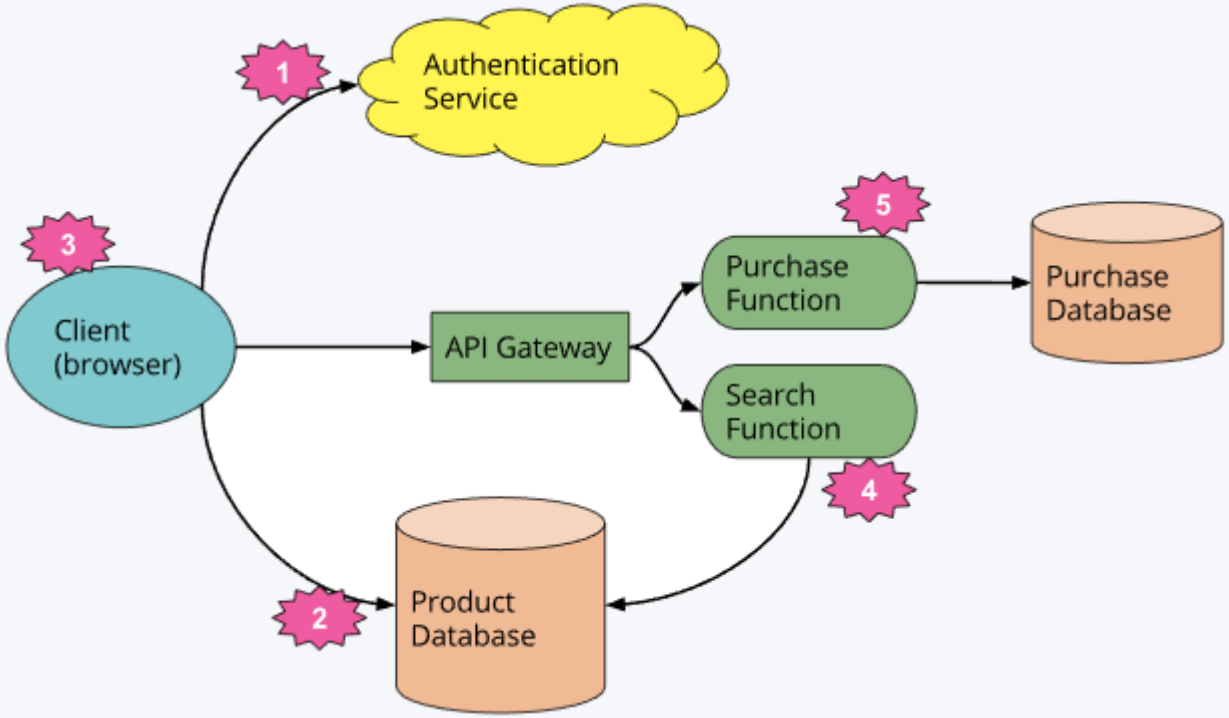
Serverless architectures are gaining traction

A couple of examples

- UI-driven application
  - Traditional architecture

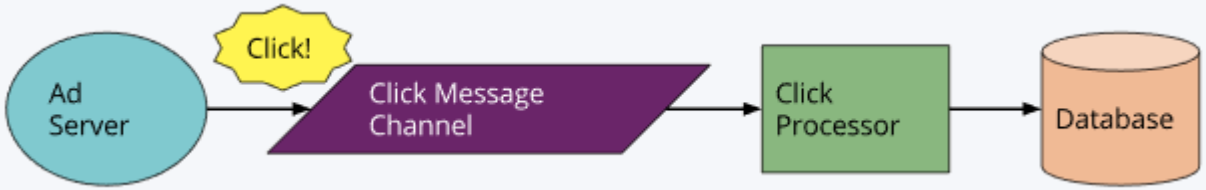
A couple of examples

- UI-driven application
  - Serverless BaaS architecture



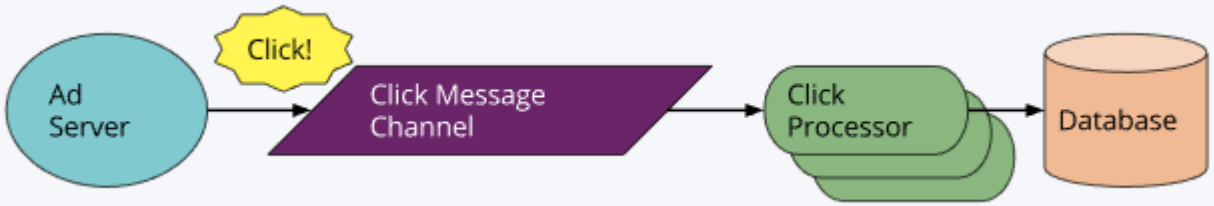
A couple of examples

- Message-driven application
  - Traditional architecture

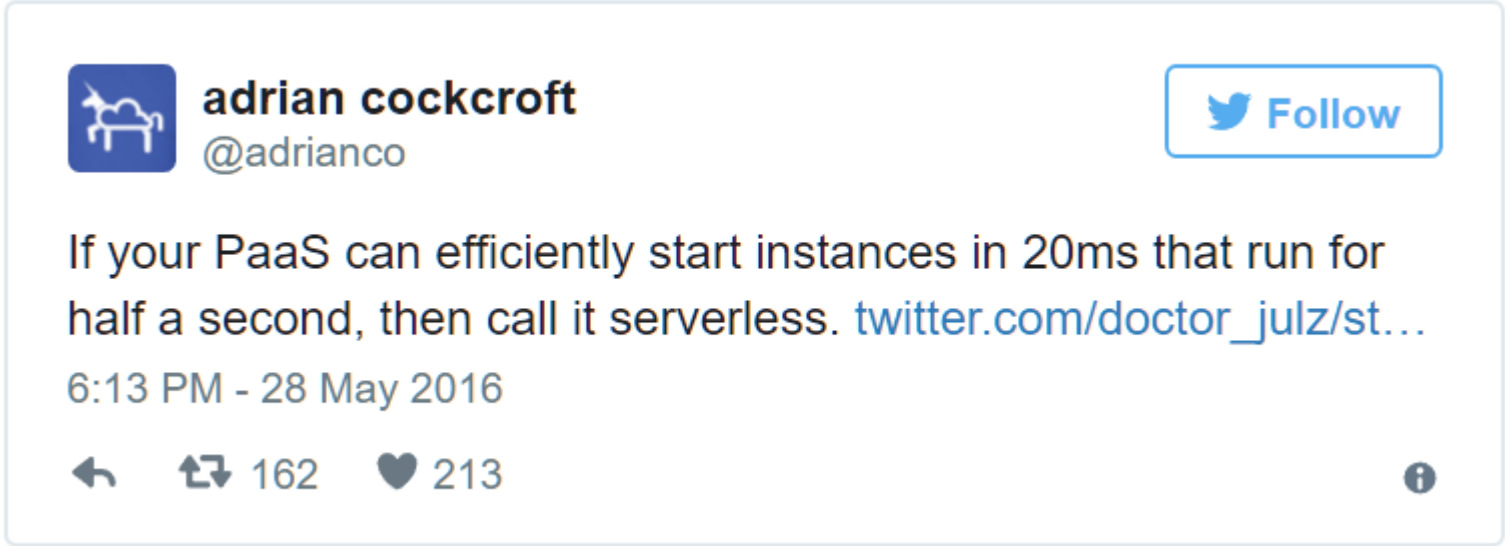


A couple of examples

- Message-driven application
  - Serverless FaaS architecture



Comparison with PaaS



Available Serverless Solutions

FUNCTION AS A SERVICE (FAAS)

- Microsoft Azure Functions
- Google Cloud Functions
- Amazon Lambda
- IBM/Apache OpenWhisk
- Iron.io IronWorker
- Joyent Manta Functions
- PubNub BLOCKS
- Serverless Docker

BACKEND AS A SERVICE (BAAS)

- Amazon API Gateway
- Amazon Cognito
- AWS DynamoDB
- Google Cloud Datastore
- Google Firebase
- AnyPresence
- Appery.io
- BaaSBox

Drawbacks

- Vendor control
- Multitenancy Problems
- Vendor lock-in
- Security concerns
- Loss of Server optimizations
- No in-server state for Serverless FaaS

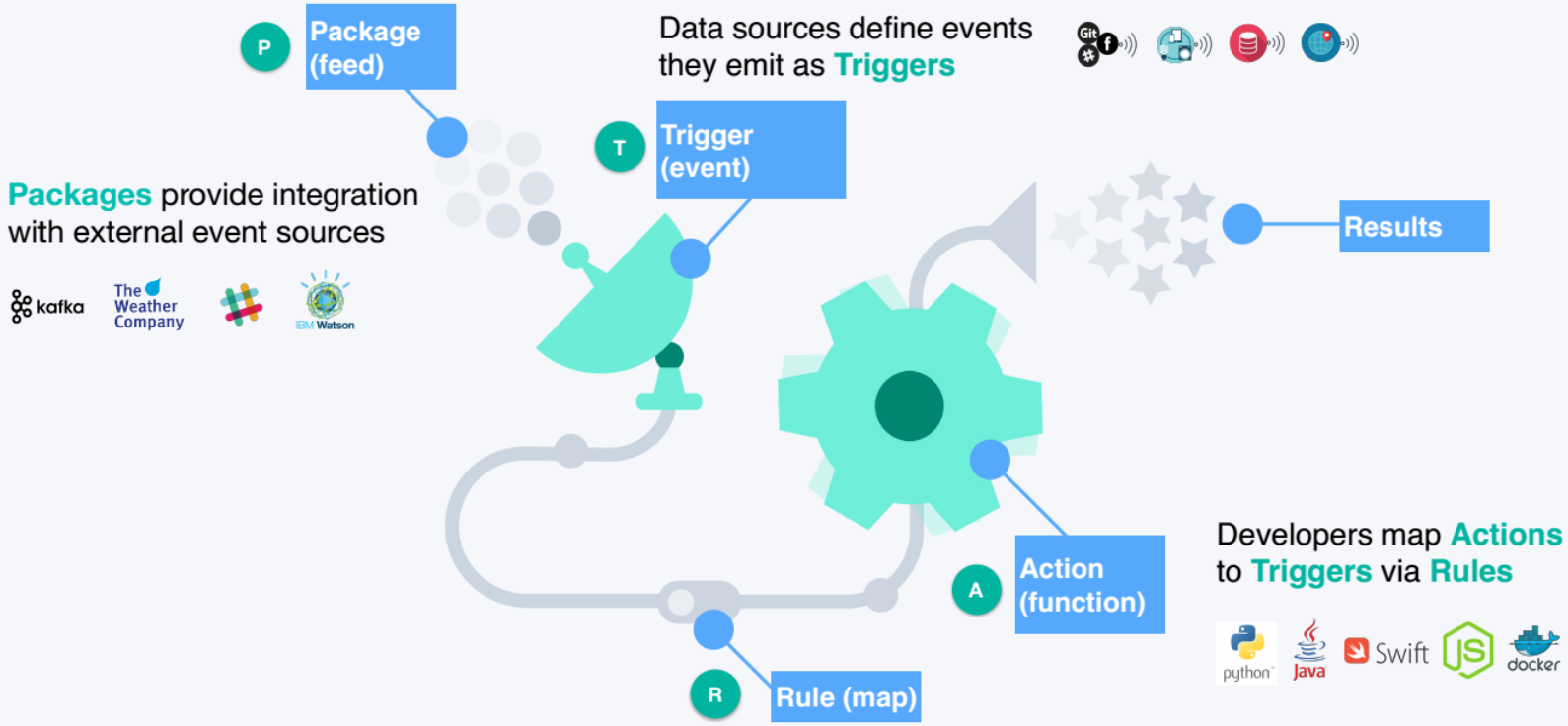
Introducing OpenWhisk

FaaS platform to execute code in response to events

- Provides serverless deployment and operations model
- Runs code only on-demand on a per-request basis
- Optimized utilization, fine-grained metering at any scale
- Flexible, extensible, polyglot programming model
- Open source and open ecosystem (Apache Incubator)
- Ability to run in public, private, and hybrid models

OpenWhisk is a cloud platform that **executes code** in **response to events**

Developers work with packages, triggers, actions, and rules



Triggers

T A class of events that can occur

Actions

A

Code that runs in response to an event
(that is, an event handler)

Actions

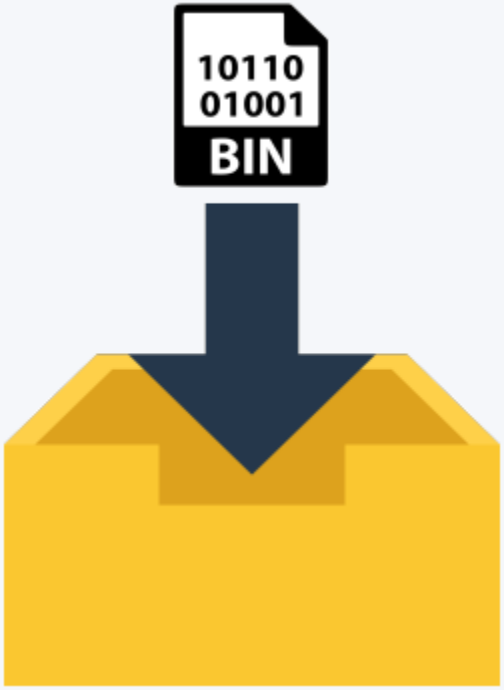
A

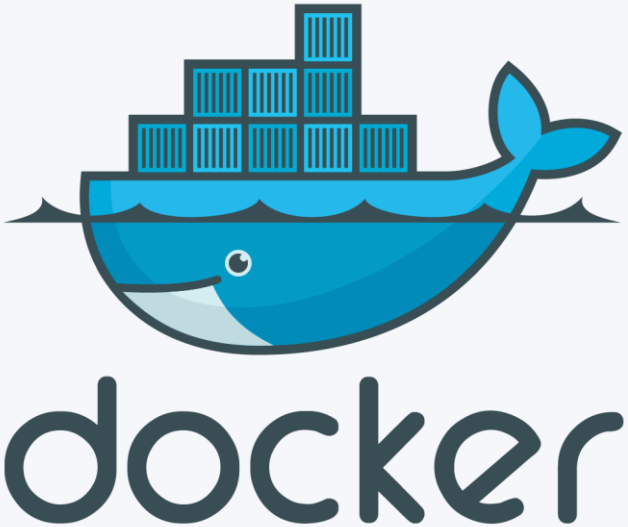
Can be written in a variety of languages, such as JavaScript, Python, Java, Swift, …

```
function main(params) {
    return { message: 'Hello, ' + params.name + ' from ' + params.place };
};
```

Actions

A Or any other language by packaging with Docker





Actions

A

Can be composed to create sequences that increase flexibility and foster reuse

$$A_A := A_1 + A_2 + A_3$$

$$A_B := A_2 + A_1 + A_3$$

$$A_C := A_3 + A_1 + A_2$$

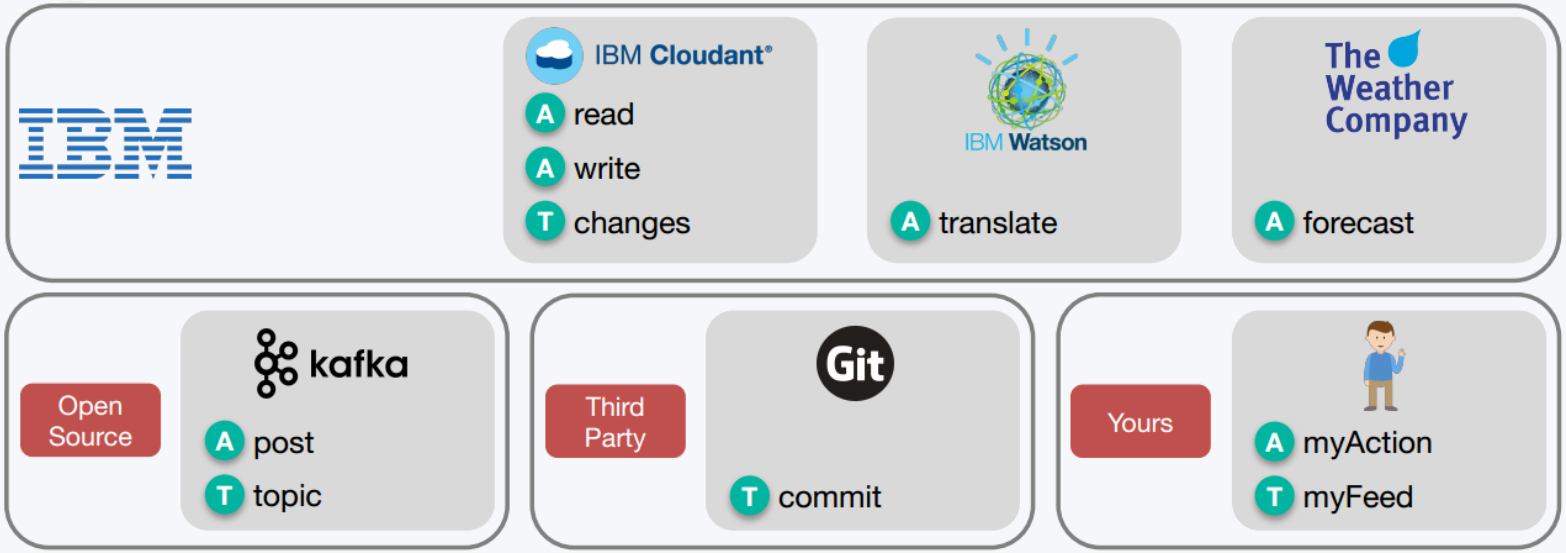
Rules

R

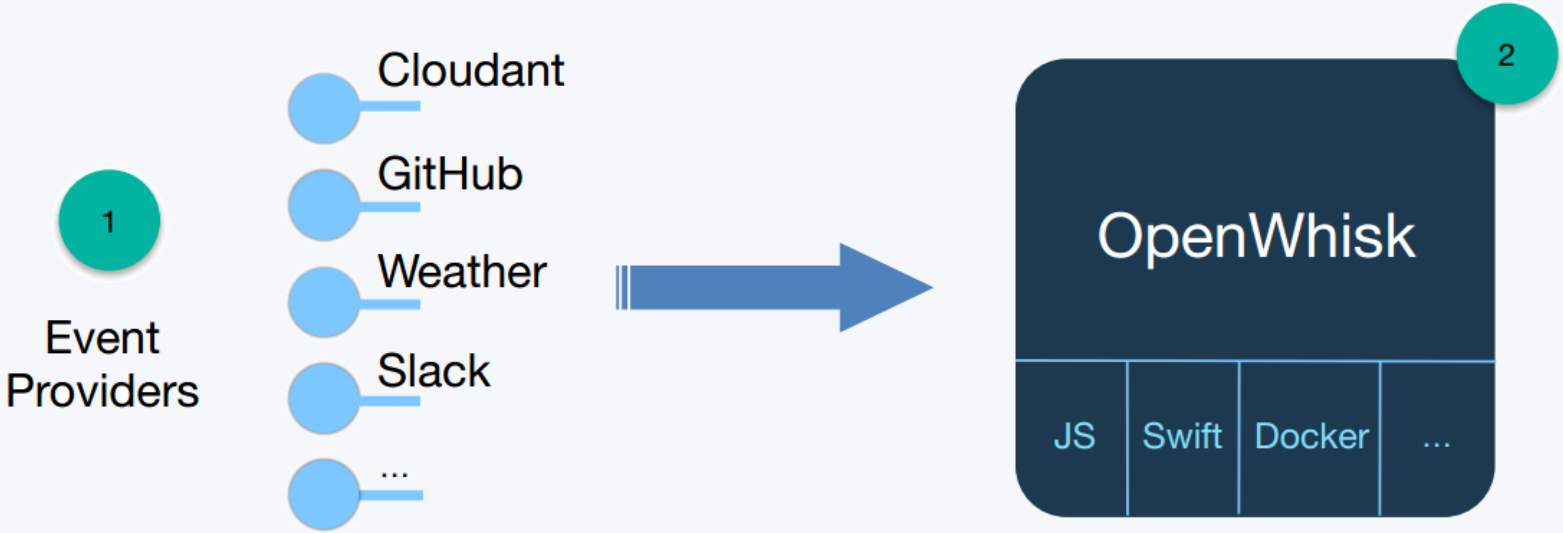
An association of a trigger to an action in a many to many mapping.

Packages

P A shared collection of triggers and actions



OpenWhisk enables event driven applications

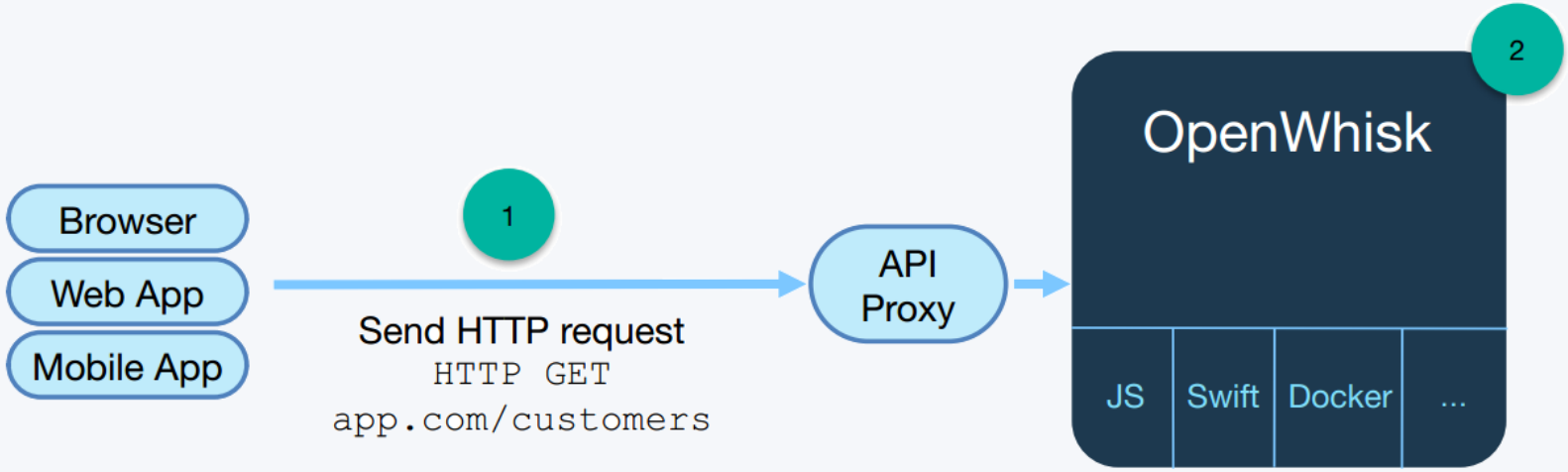


An event occurs, for example

- Commit pushed to Github repository
- Data changed in Cloudant

Which triggers execution of associated OpenWhisk action

OpenWhisk can implement REST microservices



Creating the action

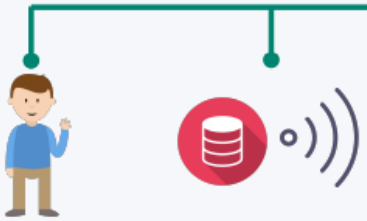
```
function main() {
    console.log('Hello World');
    return { hello: 'world' };
}
```

```
wsk action create myAction action.js
```

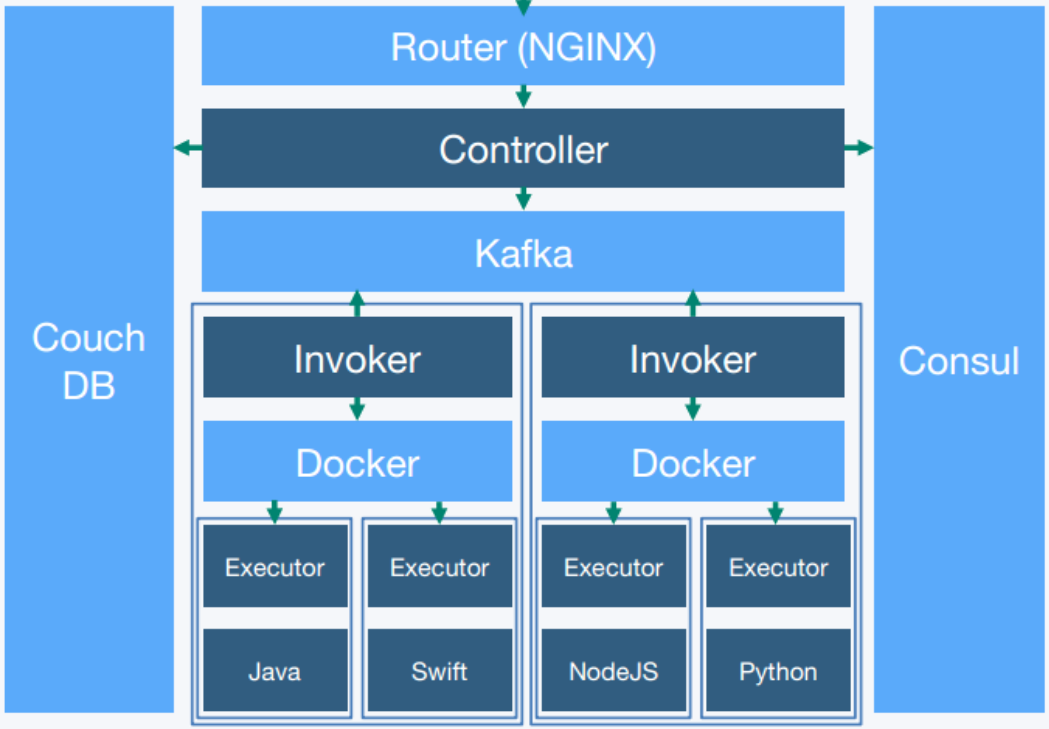
```
wsk action invoke myAction
```

OpenWhisk Architecture

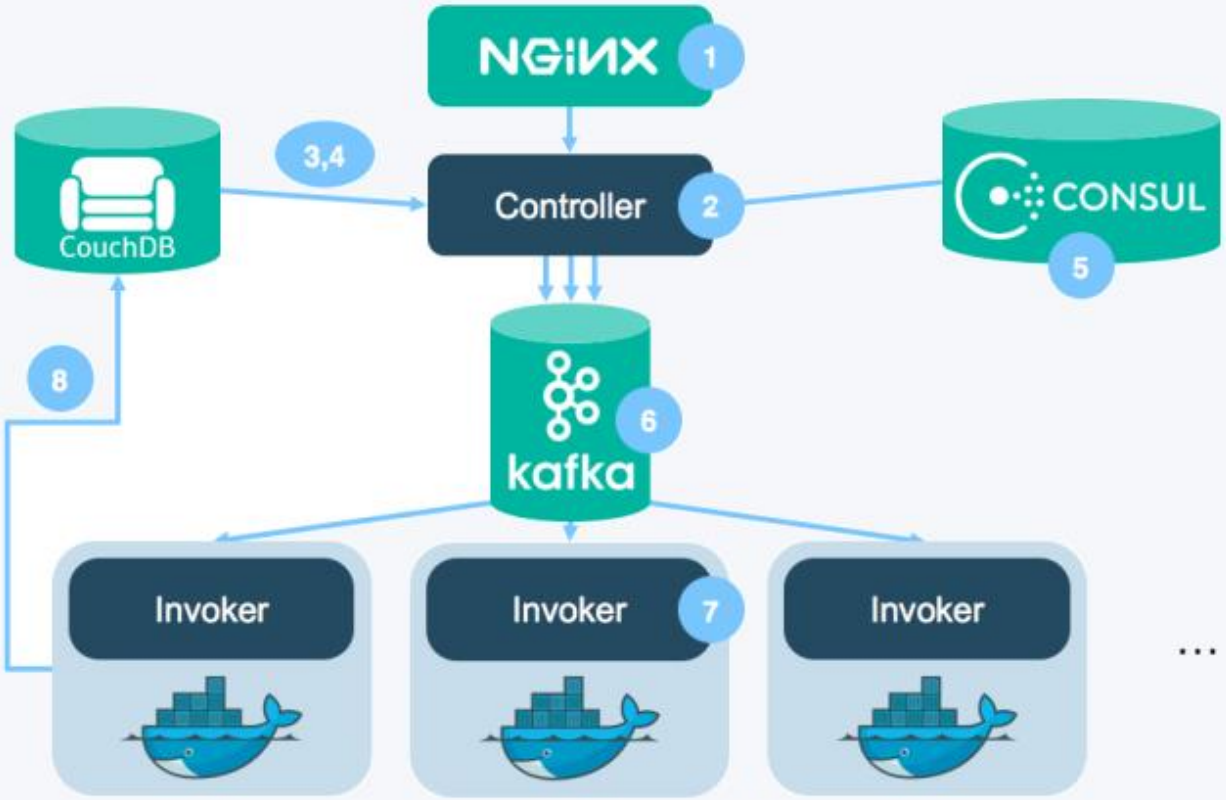
OpenWhisk under the hood: Developer view



1. Router receives request to API via CLI or UI
2. Controller checks entitlement and dispatches requests to Kafka
3. Invokers pull requests from CouchDB and start execution of the action



OpenWhisk under the hood: A deeper look



Entering the NGINX

```
wsk action invoke myAction
```

```
POST /api/v1/namespaces/$userNamespace/actions/myAction
Host: $openwhiskEndpoint
```

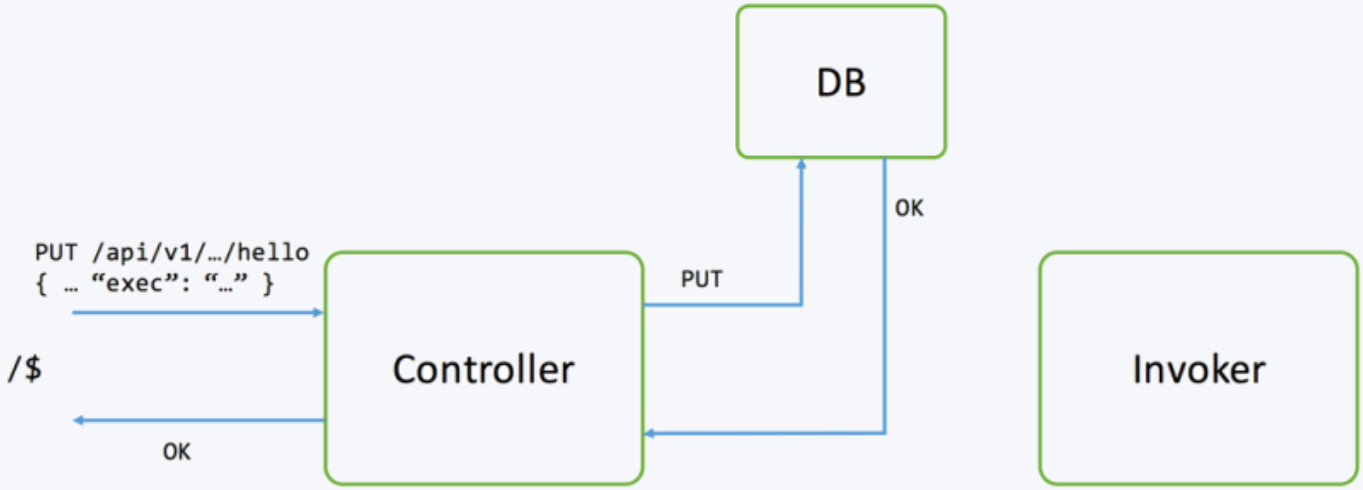
Storing the results

```
{
   "activationId": "31809ddca6f64cfc9de2937ebd44fbb9",
   "response": {
       "statusCode": 0,
       "result": {
           "hello": "world"
       }
   },
   "end": 1474459415621,
   "logs": [
       "2016-09-21T12:03:35.619234386Z stdout: Hello World"
   ],
   "start": 1474459415595,
}
```

```
wsk activation get 31809ddca6f64cfc9de2937ebd44fbb9
```

OpenWhisk container model

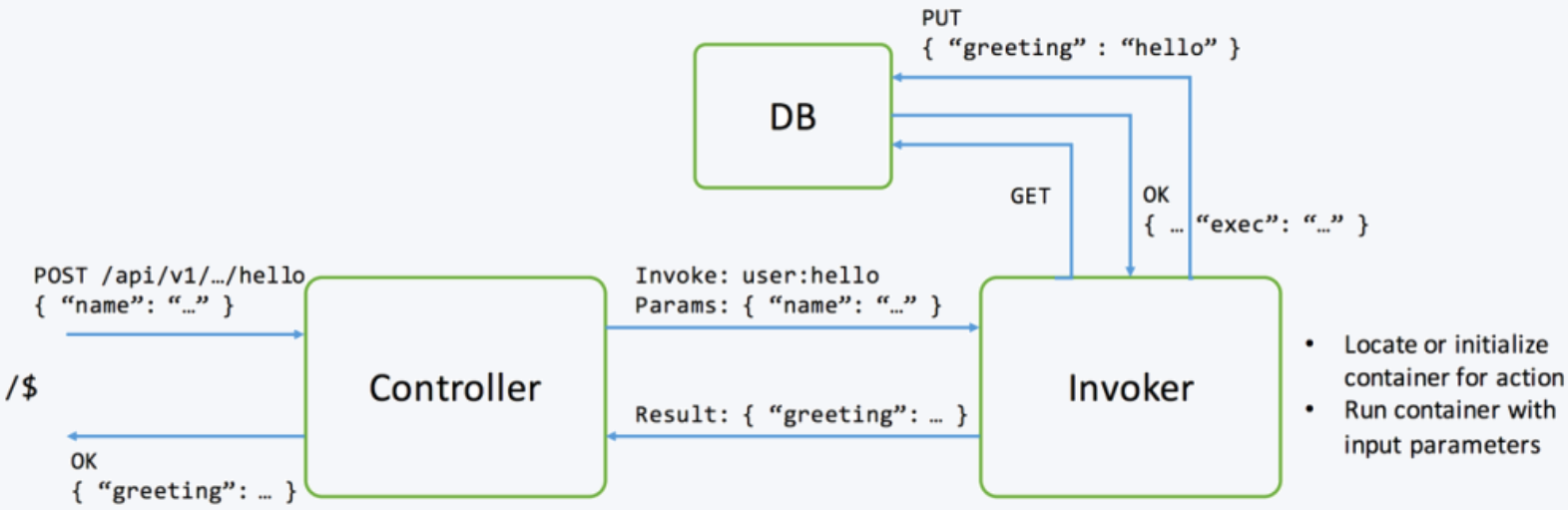
Action creation




```
/$ wsk -v action create hello-js hello.js
```

OpenWhisk container model

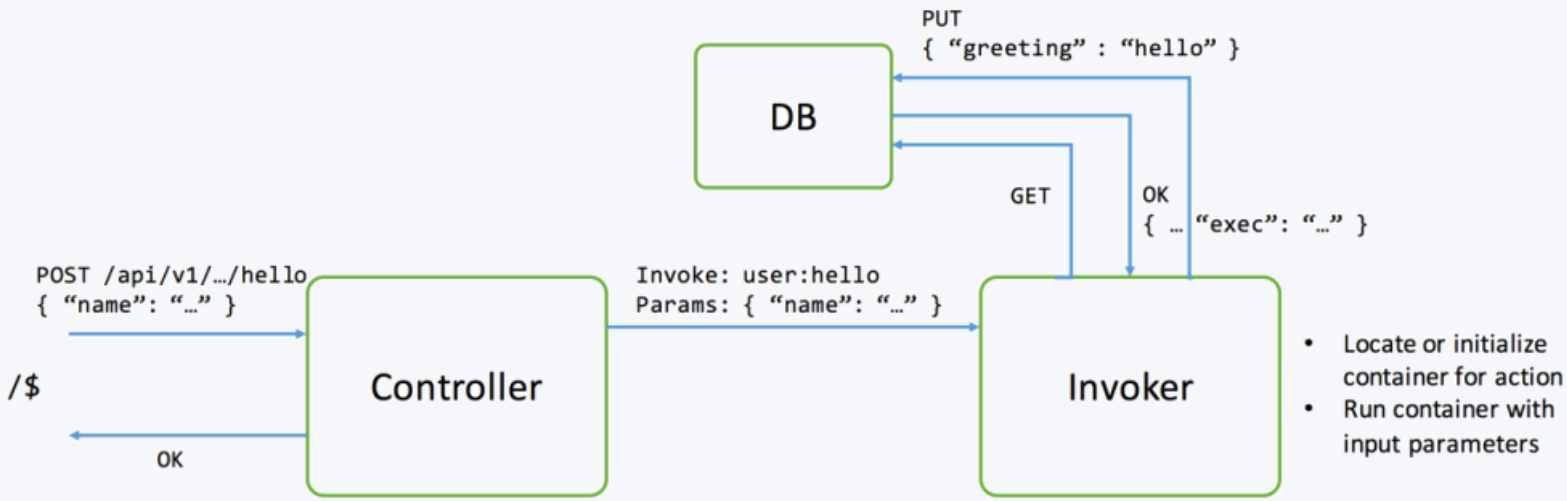
Action invocation (blocking)




```
/$ wsk -v action invoke –b hello-js -p ...
```

OpenWhisk container model

Action invocation (non-blocking)

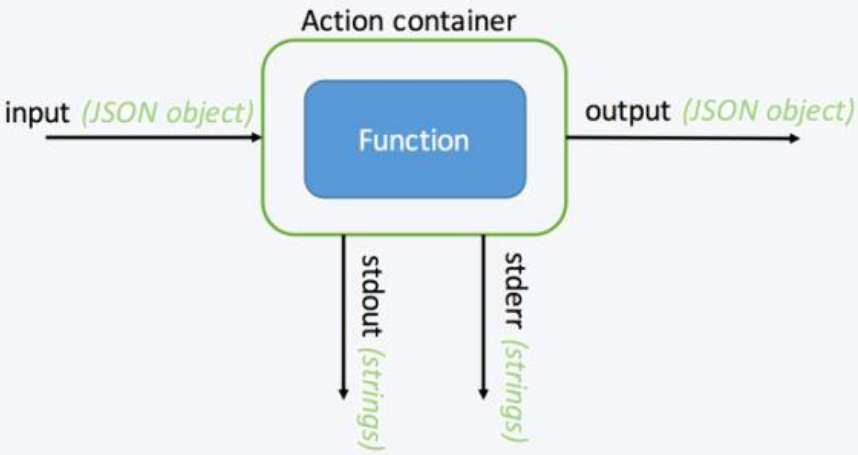



```
/$ wsk -v action invoke hello-js -p ...
```

OpenWhisk container model

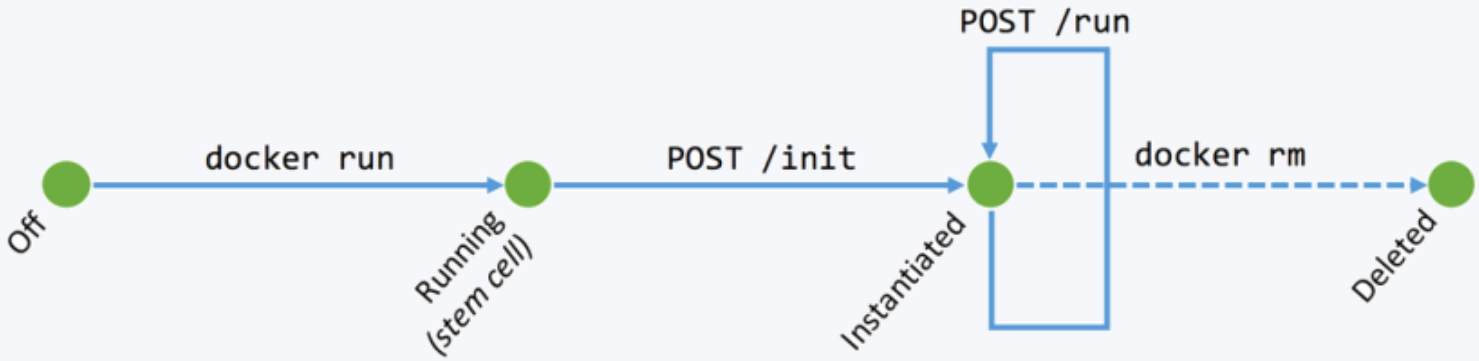
Action containers

- Host user-written function
- Maintain the illusion that "action ≈ function"
- Provide a simple REST API to:
  - Initialize the container
  - Run the function



OpenWhisk container model

- Action container lifecycle



(Additionally: `docker unpause` / `docker pause` before / after each POST.)

OpenWhisk & Containers

Behind the scenes: It’s about containers

- Basically, OpenWhisk is based on Docker… but we added some smartness to meet our performance goals…

≈

`wsk action invoke`

`docker run`

Behind the scenes: It’s about containers

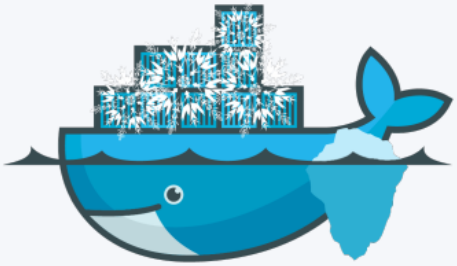
Start container
`docker run`

→

Initialize
`/init`

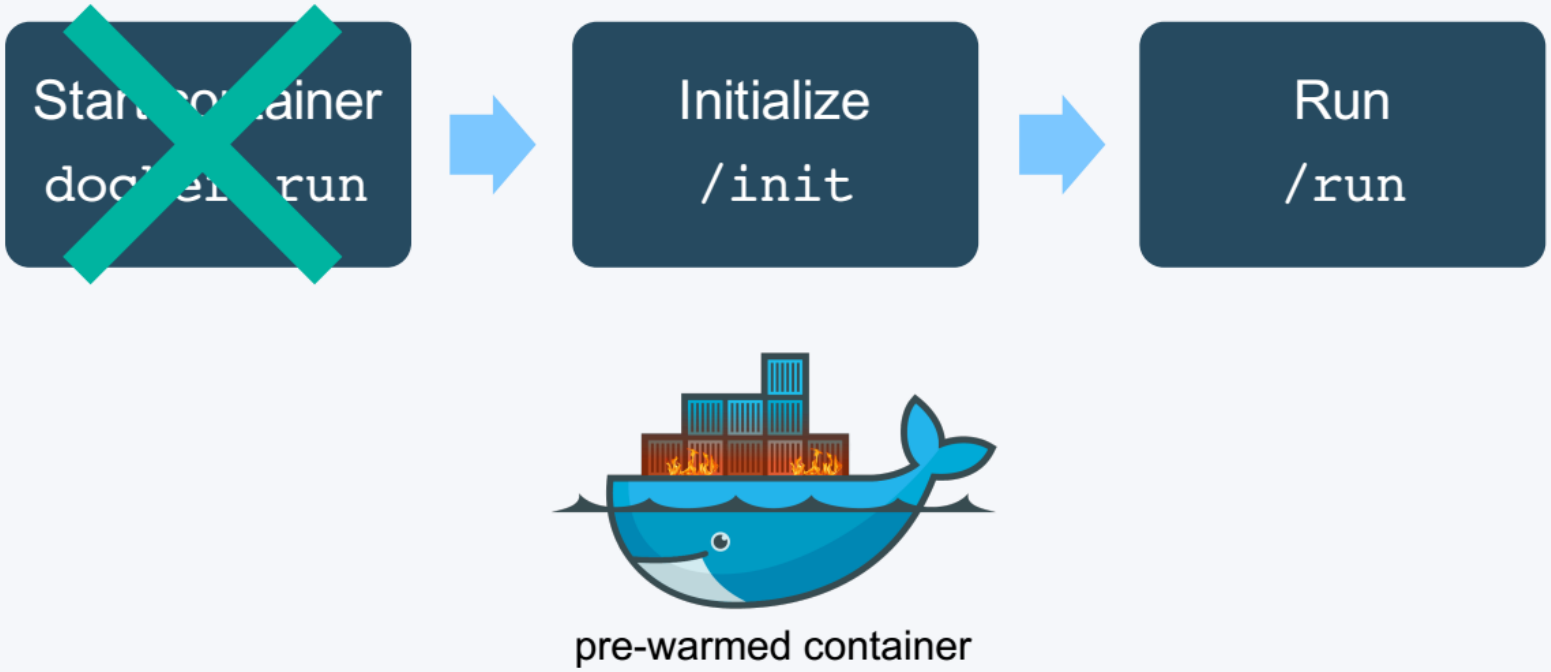
→

Run
`/run`

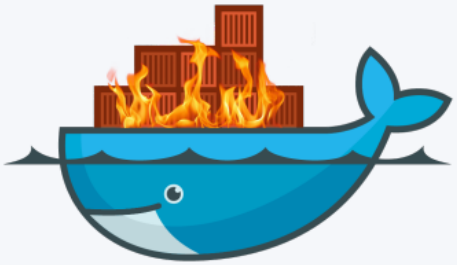


cold container

Behind the scenes: It’s about containers



Behind the scenes: It’s about containers

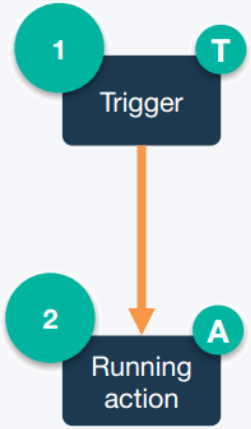


warm container

Performance is king…

Demos & Use cases

Create a timer triggered action



Cron syntax alarm

Log the current time

```
function main(params) {
    var date = new Date();
    var now = (date.getHours() + ":" +
               date.getMinutes() + ":" +
               date.getSeconds());
    console.log("Invoked at: " + now);
    return { message: "Invoked at: " + now };
}
```

```
$ wsk action create handler handler.js
```

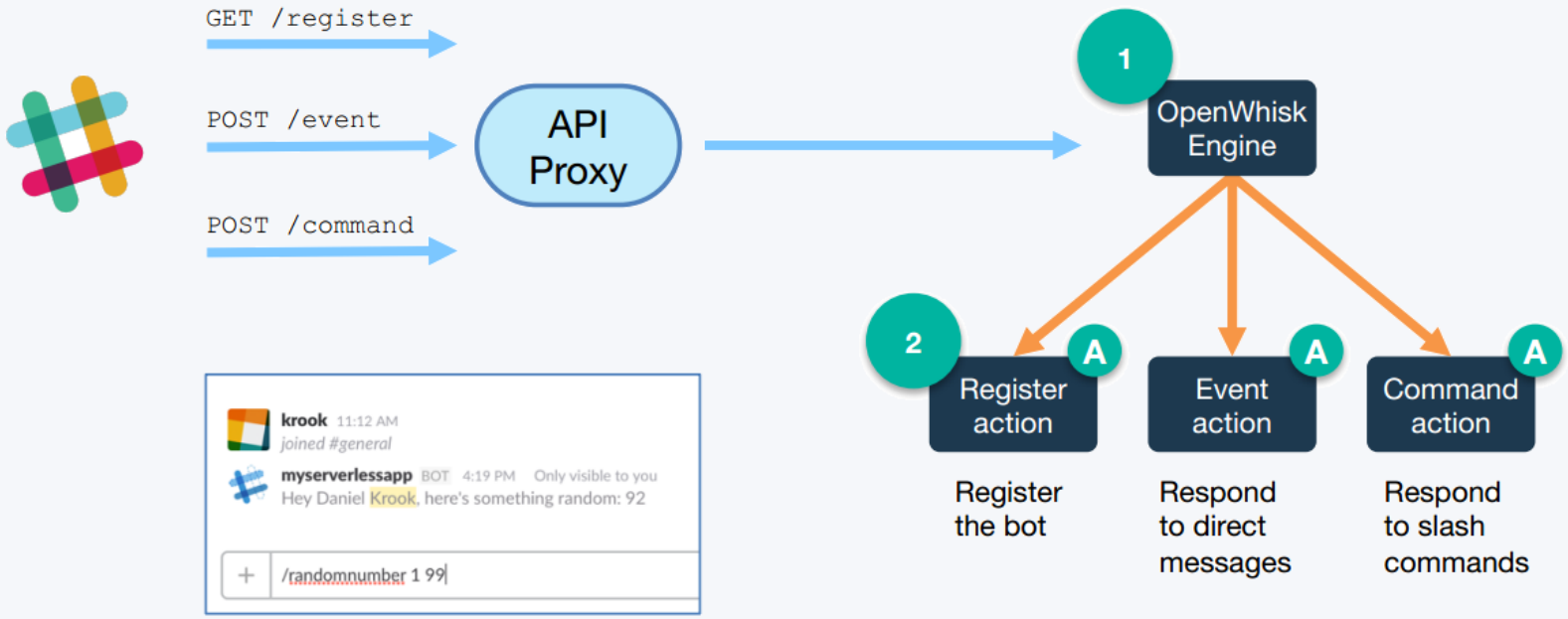
```
$ wsk action invoke --blocking handler
```

```
$ wsk trigger create every-20-seconds \
  --feed /whisk.system/alarms/alarm
  --param cron "*/20 * * * * *"
```

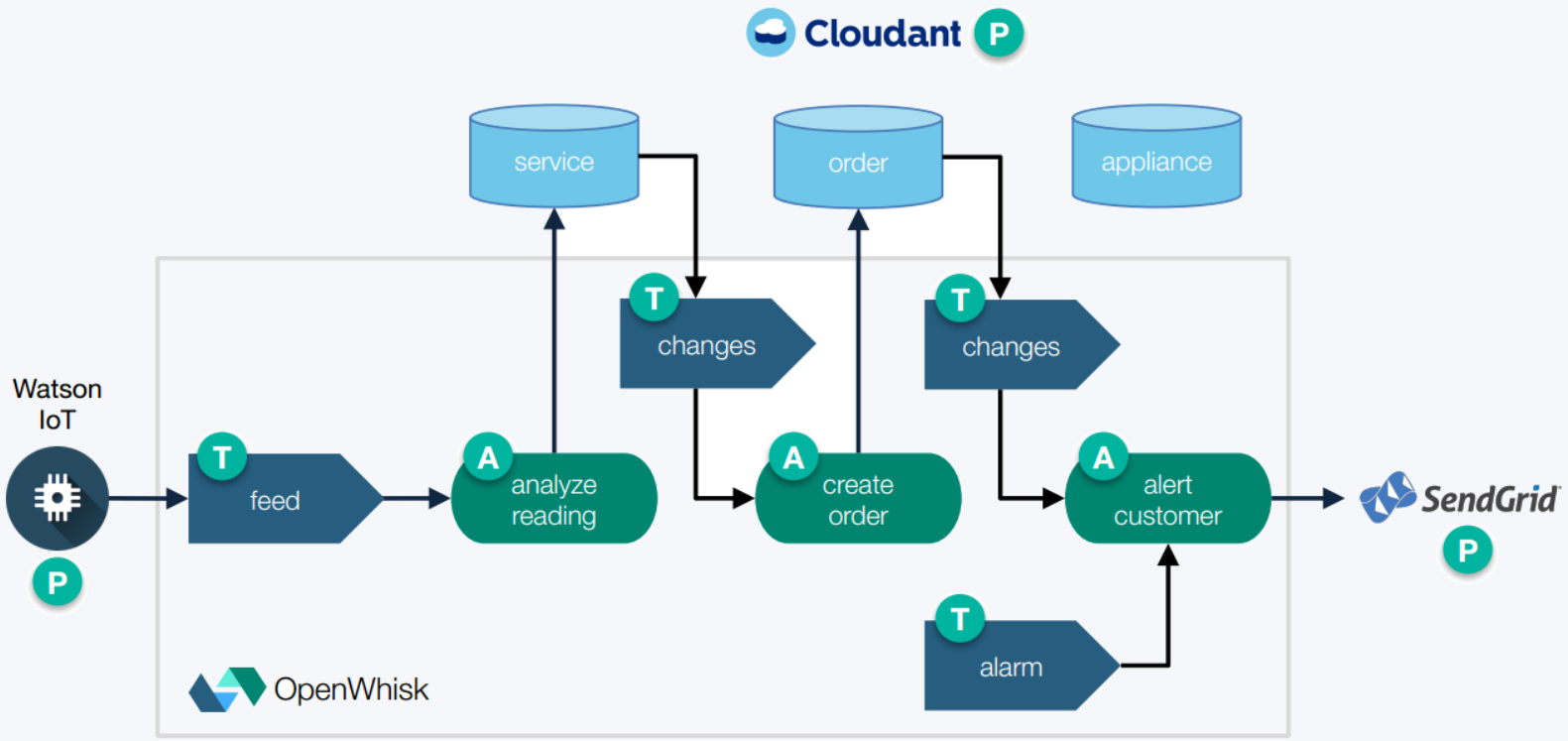
```
$ wsk rule create --enable \
  invoke-periodically \
  every-20-seconds \
  handler
```

```
$ wsk activation poll
```

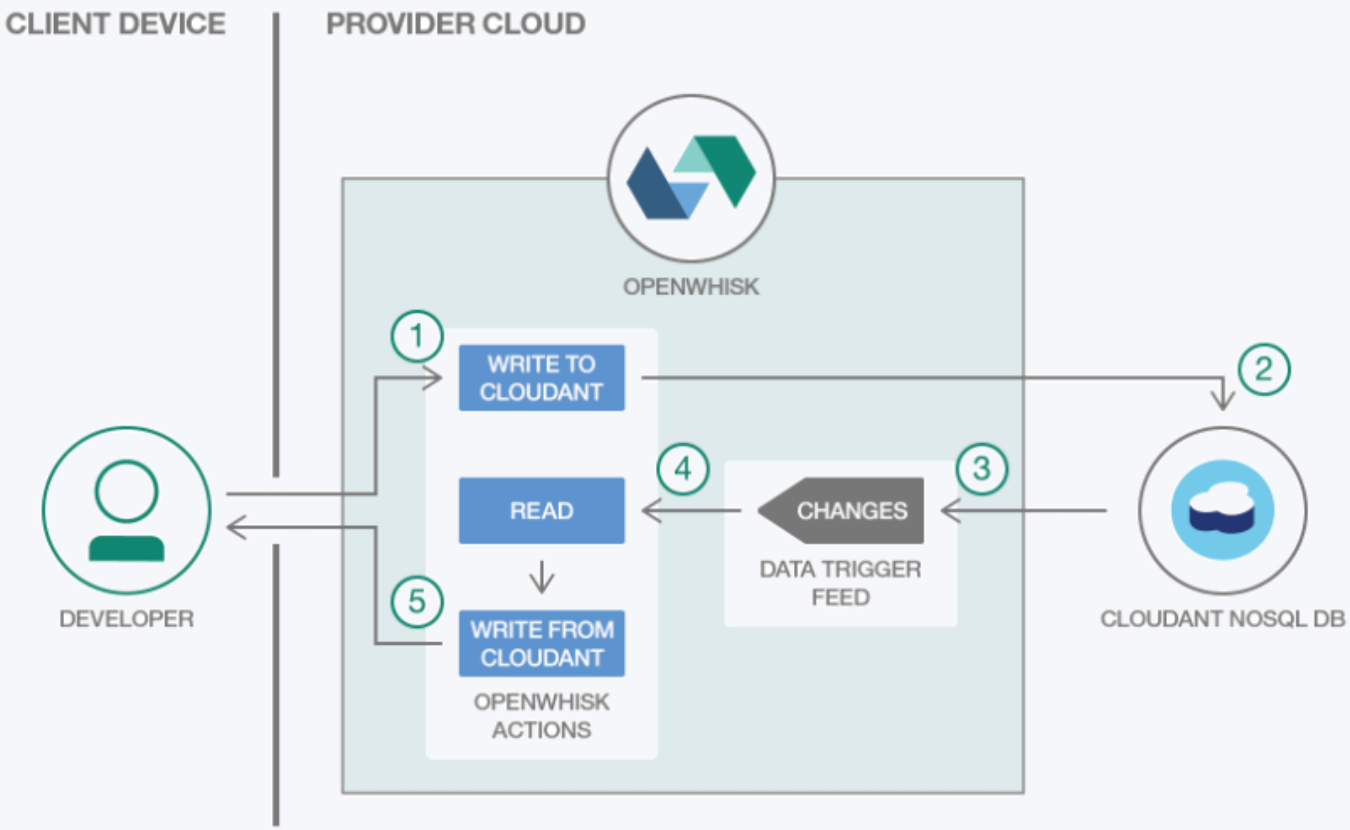
Create a Slack bot



IoT


Cloudant
P
service
order
appliance
T
changes
T
changes
Watson
IoT
P
T
feed
A
analyze
reading
A
create
order
A
alert
customer
SendGrid
P
T
alarm
OpenWhisk

Data Processing



Customers & Partners

Customers and Partners

Clients

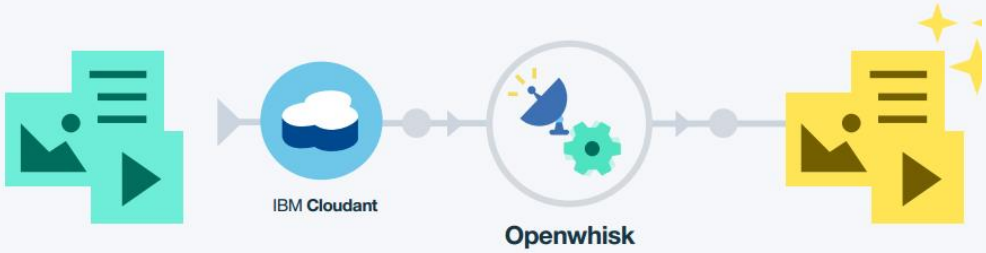
SiteSpirit
Santander
articoolo
AbilSense
BIGVU POCKET TV STUDIO
GreenQ Making Garbage Trucks Smarter
MAGENTIQ SEE BEYOND THE VISIBLE
The Weather Company
ADVISOR CONNECT

Partners

Adobe
THE APACHE SOFTWARE FOUNDATION
ALTOROS™
nepente ARTESANATO DE SOFTWARE
Miracle Systems
{ ••• } SWAGGERhub
PubNub

What do customers do with OpenWhisk?

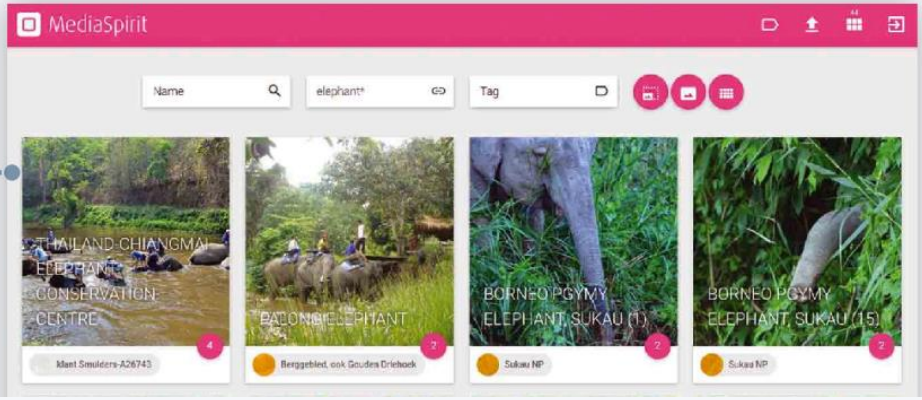
Data processing



SiteSpirit

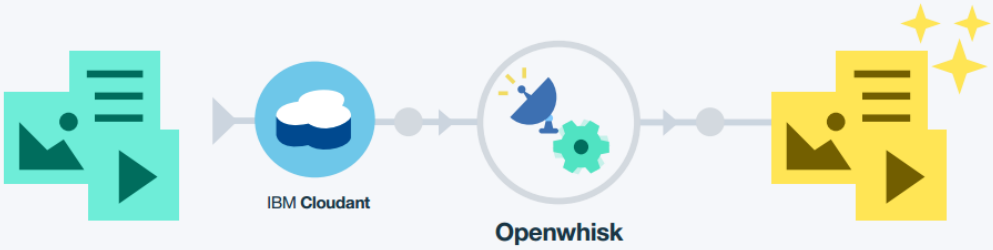
http://ecc.ibm.com/case-study/us-en/ECCF-CDC12387USEN

10x faster
90% less cost



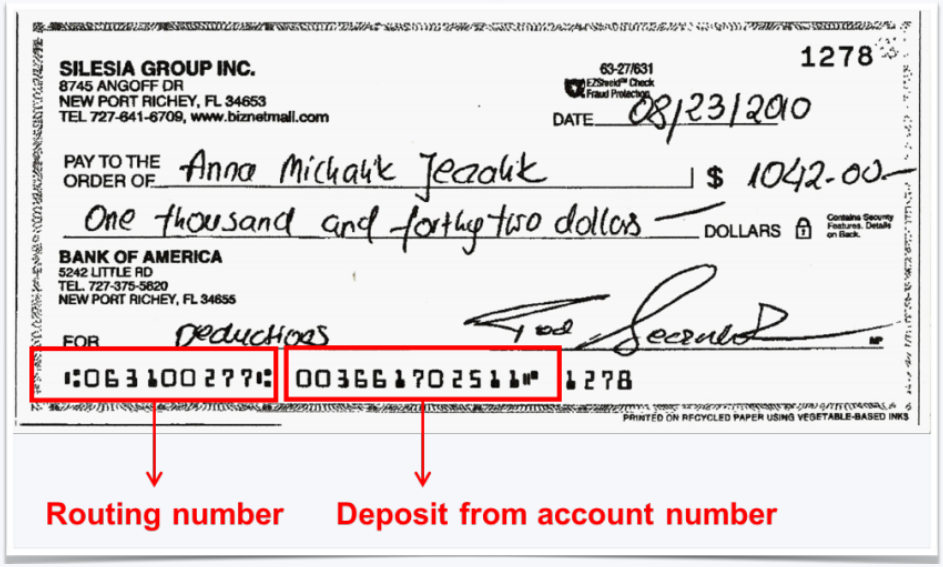
What do customers do with OpenWhisk?

Data processing



Santander

Less cost
<$2 for all paper checks processed within 1 year



Q&A