Build a Delightful API with Serverless Technology

Rob Allen, Nineteen Feet

January 2018 ~ @akrabat

I write APIs

Deployment options

1. Physical servers
2. Virtual machines
3. Containers

Deployment options

1. Physical servers
2. Virtual machines
3. Containers
    a. Platform (e.g. Kubernetes)
    b. Application (e.g. Cloud Foundry)
    c. Serverless (e.g. OpenWhisk)

Serverless?

"The first thing to know about serverless computing is that "serverless" is a pretty bad name to call it."

Brandon Butler, Network World

AKA: Functions as a Service

- A runtime to execute your functions
- No capacity planning or load balancing
- Pay for execution, not when idle

Use-cases

Synchronous

Service is invoked and provides immediate response
(HTTP requests: APIs, chat bots)

Asynchronous

Push a message which drives an action later
(web hooks, timed events, database changes)

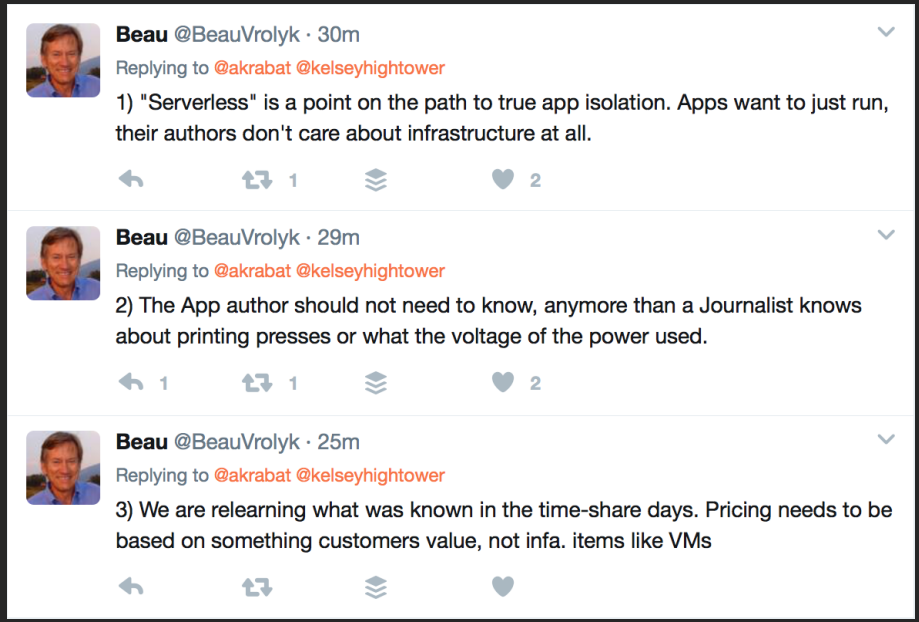
Benefits

- No need to think about servers
- Concentrate on application code
- Pay only for what you use, when you use it
- Language agnostic: NodeJS, Swift, Python, PHP Java, C#, etc

Challenges

- Start up latency
- Time limit
- State is external
- DevOps is still a thing

It's about value



Beau @BeauVrolyk · 30m

Replying to @akrabat @kelseyhightower

1) "Serverless" is a point on the path to true app isolation. Apps want to just run, their authors don't care about infrastructure at all.

1 2

Beau @BeauVrolyk · 29m

Replying to @akrabat @kelseyhightower

2) The App author should not need to know, anymore than a Journalist knows about printing presses or what the voltage of the power used.

1 1 2

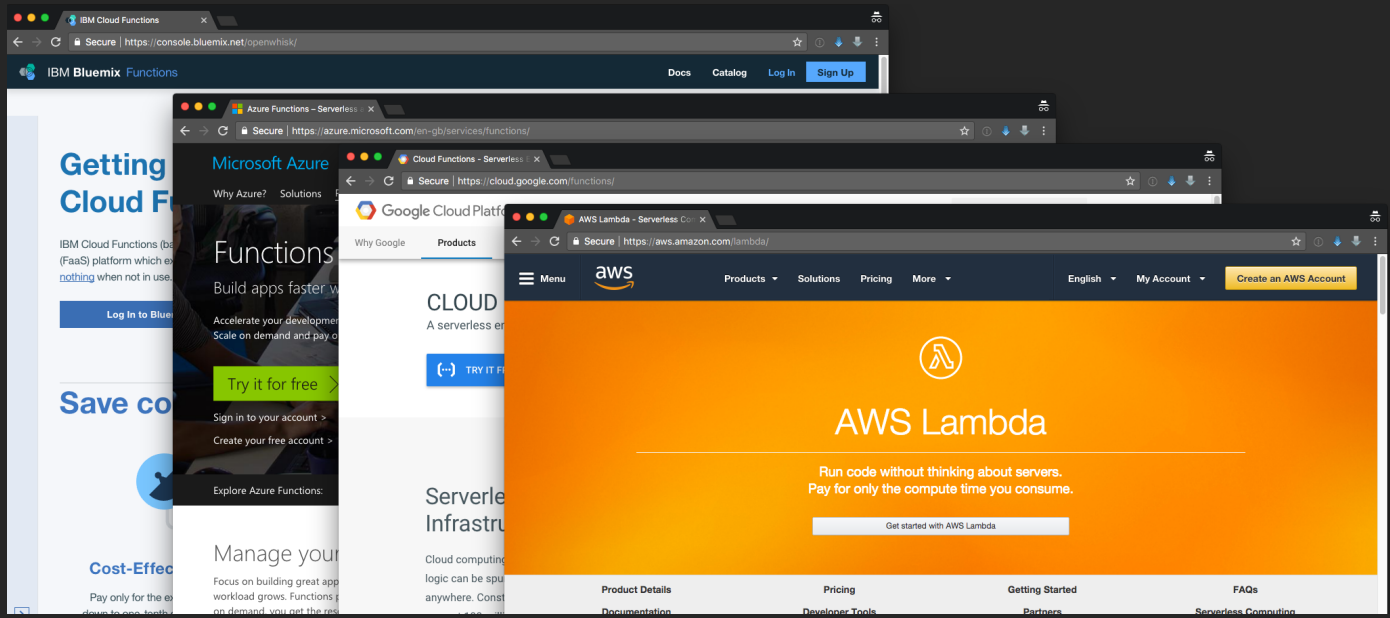
Beau @BeauVrolyk · 25m

Replying to @akrabat @kelseyhightower

3) We are relearning what was known in the time-share days. Pricing needs to be based on something customers value, not infa. items like VMs

You are charged by duration of execution & memory used

Serverless providers



OpenWhisk

Open source

Multiple providers:

- IBM
- Adobe
- RedHat

Swift is a general-purpose programming language built using a modern approach to safety, performance, and software design patterns.

-- swift.org

Major features

Strong typing
Type inference
Optionals
Closures
ARC

Custom operators
Tuples
Generics
Serialisation
Interoperable with C

Rock-Paper-Scissors

```
 1 import Foundation
 2
 3 let shapes = ["rock", "paper", "scissors"]
 4
 5 for count in 1...3 {
 6     print(count)
 7     sleep(1)
 8 }
 9
10 srandom(UInt32(Date().timeIntervalSince1970))
11 let chosenShape = random() % shapes.count
12 print(shapes[chosenShape]);
```

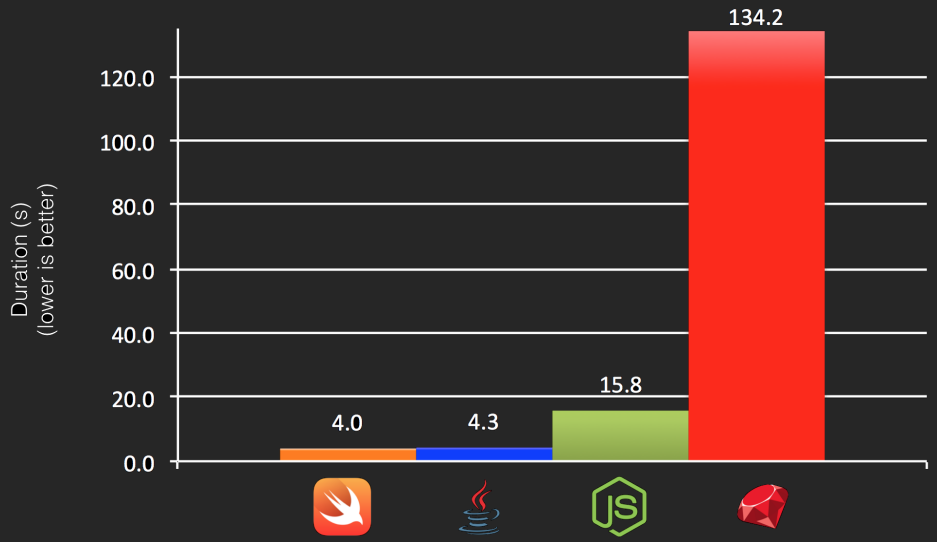
Result

```
$ swift rock-paper-scissors.swift
1
2
3
scissors
```

Remember this?

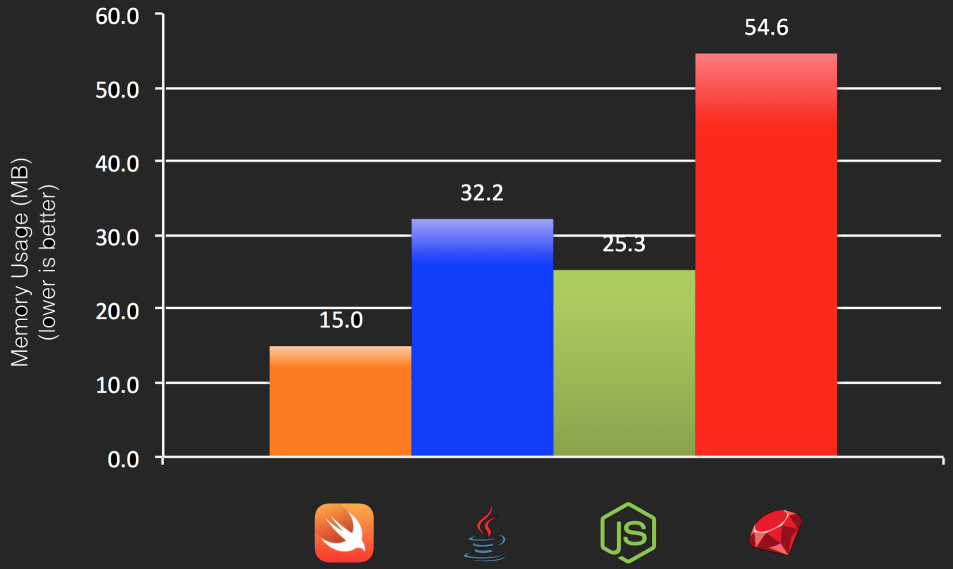
You are charged by duration of execution & memory used

Performance



http://benchmarksgame.alioth.debian.org/u64q/performance.php?test=spectralnorm

Memory

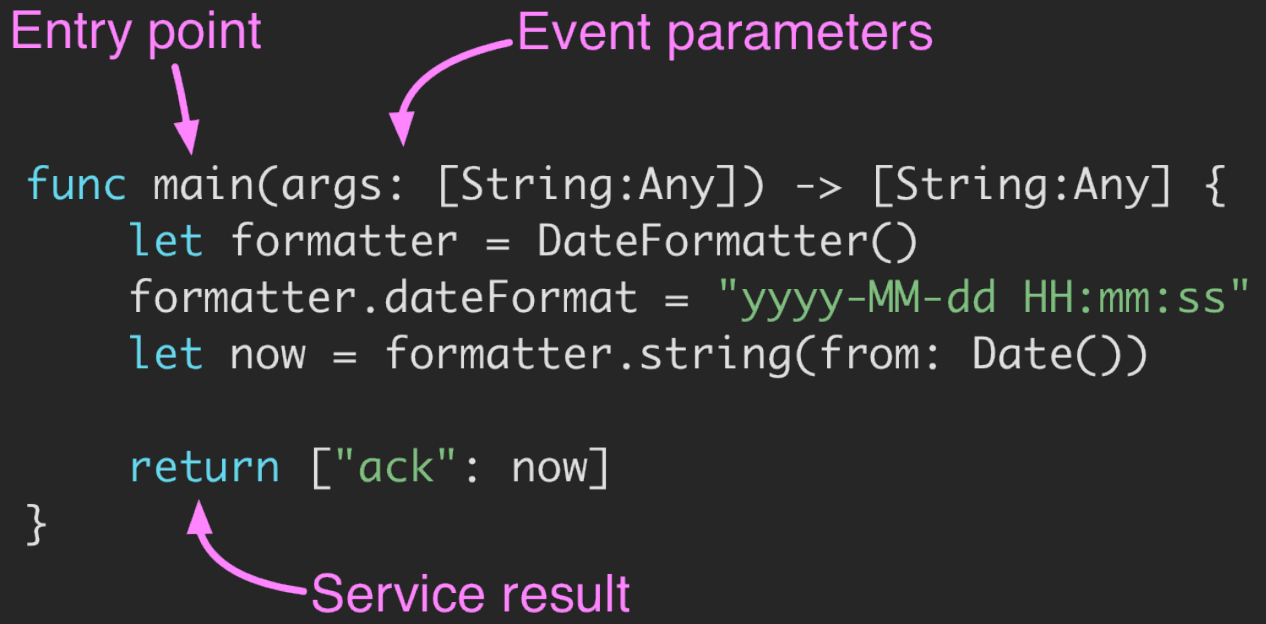


http://benchmarksgame.alioth.debian.org/u64q/performance.php?test=spectralnorm

Hello world in OpenWhisk

```
1 func main(args: [String:Any]) -> [String:Any] {
2     let formatter = DateFormatter()
3     formatter.dateFormat = "yyyy-MM-dd HH:mm:ss"
4     let now = formatter.string(from: Date())
5
6     return ["ack": now]
7 }
```

Hello world in OpenWhisk



Running your action

```
$ wsk action update ping ping.swift
ok: updated action ping

$ wsk action invoke ping --result
{
  "ack": "2018-01-10 09:15:00"
}
```

Public HTTP access

Web actions

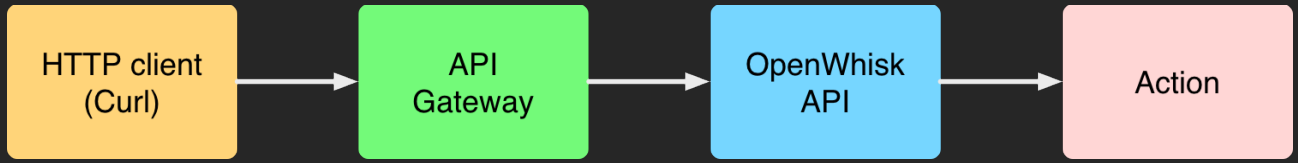

```
$ wsk action update ping ping.swift --web true

$ curl https://openwhisk.eu-gb.bluemix.net/api/v1/ \
    web/19FT_demo/default/ping.json
{
  "ack": "2018-01-10 09:15:00"
}
```

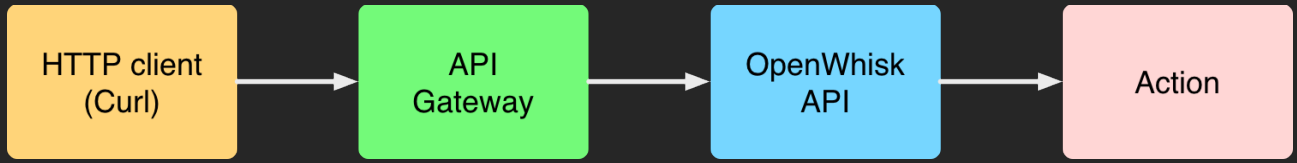
What about...

API routing?
Rate limiting?
Authentication?
Custom domains?

API Gateway



API Gateway




```
$ wsk api create /myapp /ping GET ping

$ curl https://service.eu.apiconnect.ibmcloud.com/gws/apigateway/api/ \
   d12f3ba5e86077e6f0c8...830db1e5a05c88edfb12/ping
{
  "ack": "2018-01-10 09:15:00"
}
```

Let's talk about HTTP APIs

HTTP APIs

Just because it's *serverless* doesn't mean we can ignore the basics!

- Status codes
- HTTP method negotiation
- Content-type handling
- Error handling
- Media type format

Status codes

Send the right one for the right situation!

1xx	Informational
2xx	Success
3xx	Redirection
4xx	Client error
5xx	Server error

Full control over response

Return a dictionary with the keys: `statusCode`, `headers`, `body`

Do not use an extension on web actions:

```
curl https://openwhisk.eu-gb.bluemix.net/api/v1/ \
 web/19FT_demo/default/ping
```

Enable the *http* response on API Gateway:

```
wsk api create /myapp /ping GET hello --response-type http
```

Full control over response

```
1  func main(args: [String:Any]) -> [String:Any]
2  {
3    // our code here does something
4
5    return [
6      "statusCode" : 201,
7       "headers": [
8           "Content-Type": "application/json",
9       ],
10     "body" : ["result": "Item created"],
11   ]
12 }
```

HTTP verbs

Method	Used for	Idempotent?
GET	Retrieve data	Yes
PUT	Change data	Yes
DELETE	Delete data	Yes
POST	Change data	No
PATCH	Update data	No

In Serverless, prefer Idempotent

Handling incoming data

Parameters arrive in `args` dictionary

```
 1 func main(args: [String:Any]) -> [String:Any] {
 2     let formatter = DateFormatter()
 3     formatter.dateFormat = "yyyy-MM-dd HH:mm:ss"
 4     let now = formatter.string(from: Date())
 5
 6     var data = ["ack": now]
 7     if let state = args["state"] as? String {
 8         data["state"] = state
 9     }
10
11     return ["body": data]
12 }
```

Handling incoming data

Parameters arrive in `args` dictionary

```
 1 func main(args: [String:Any]) -> [String:Any] {
 2     let formatter = DateFormatter()
 3     formatter.dateFormat = "yyyy-MM-dd HH:mm:ss"
 4     let now = formatter.string(from: Date())
 5
 6     var data = ["ack": now]
 7     if let state = args["state"] as? String {
 8         data["state"] = state
 9     }
10
11     return ["body": data]
12 }
```

Handling incoming data

GET:

```
$ curl https://openwhisk.eu-gb.bluemix.net/api/.../ping?state=ABC
```

```
{ "ack": "2018-01-10 09:15:00", "state": "ABC" }
```

POST:

```
$ curl -X POST -d '{"state": "123"}' \
    https://openwhisk.eu-gb.bluemix.net/api/.../ping
```

```
{ "ack": "2018-01-10 09:15:00", "state": "123" }
```

HTTP request information

```
 1 func main(args: [String:Any]) -> [String:Any]
 2 {
 3     guard
 4         let method = args["__ow_method"] as? String,
 5         let path = args["__ow_path"] as? String,
 6         let headers = args["__ow_headers"] as? [String:Any]
 7     else {
 8         print("Error. Unable to unwrap HTTP request info.")
 9         return ["statusCode": 500, "error": "Internal Server Error"]
10     }
11
12     // `method`, `path` and `headers` are valid
```

Content negotiation

Correctly parse the request

- Read the `Content-Type` header
- Raise "415 Unsupported media type" status if unsupported

Create the correct response

- Read the `Accept` header
- Set the `Content-Type` header

Reading the HTTP body

JSON is parsed automatically, for everything else read `__ow_body`

```
 1 import AEXML;
 2
 3 func main(args: [String:Any]) -> [String:Any]
 4 {
 5     let headers = args["__ow_headers"] as? [String:Any]
 6     let body = args["__ow_body"] ?? ""
 7     let xmlDoc = try AEXMLDocument(xml: body)
 8
 9     // work with `xmlDoc`
10 }
```

Demo

To sum up

Resources

- https://github.com/SwiftOnTheServer/flashcards

akrabat.com/talks/build-a-delightful-api-with-serverless-technology-co

- https://swift.org
- http://openwhisk.incubator.apache.org
- https://medium.com/openwhisk

Thank you!

Please provide feedback via AttendeeHub

Rob Allen ~ @akrabat ~ https://akrabat.com