

Serverless Swift - OpenWhisk

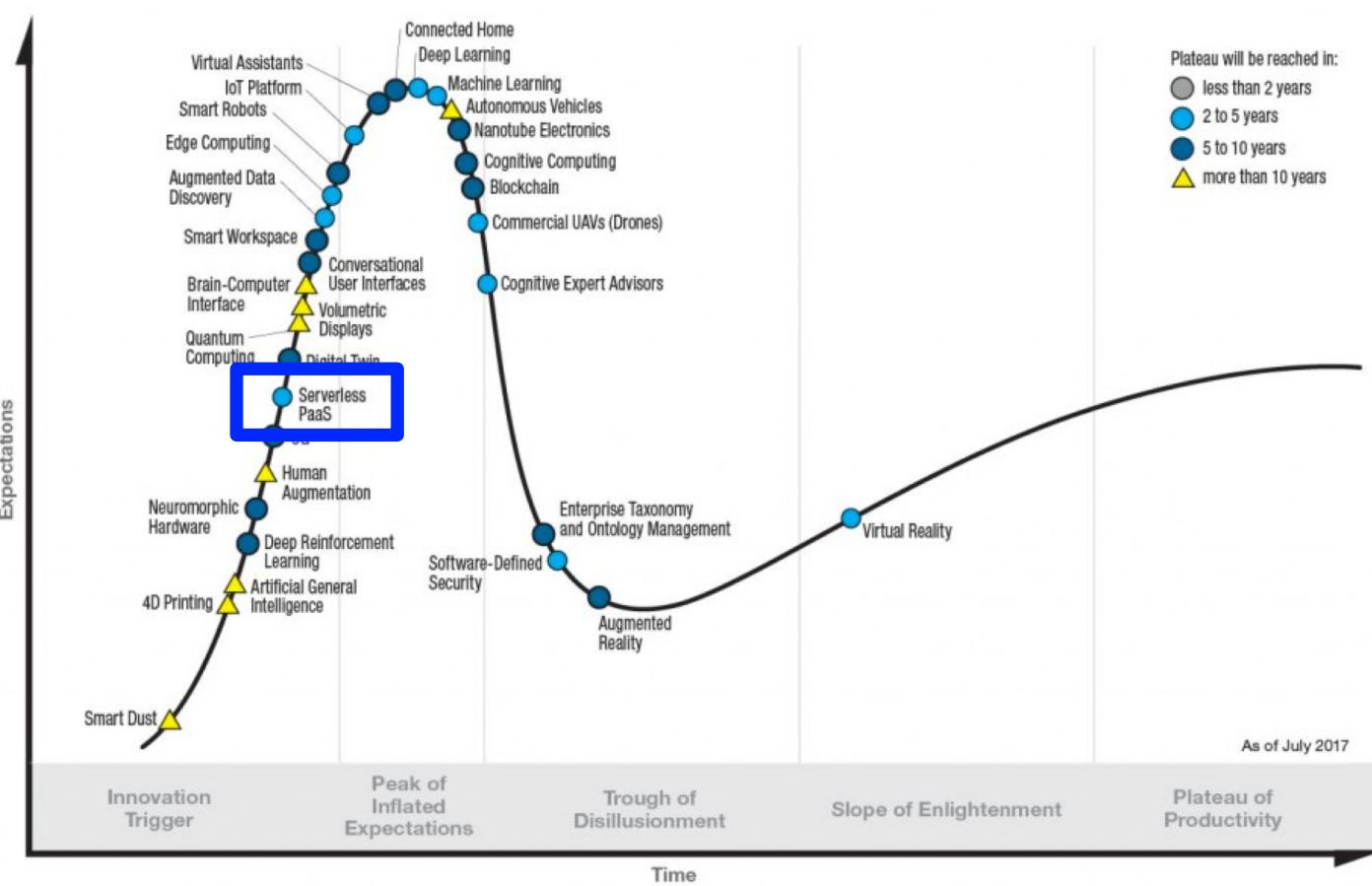
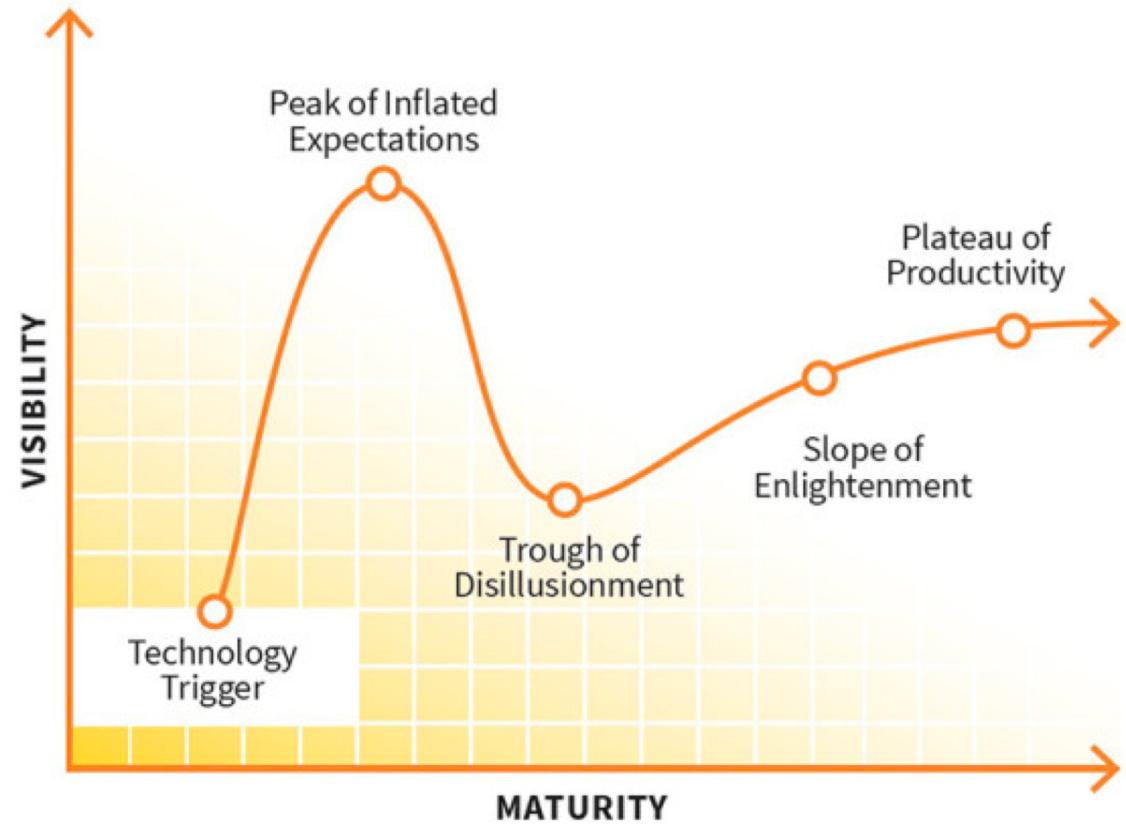
Carlos Santana

IBM Software Engineer (STSM)

@csantanapr

Serverless EVERYTHING...

@csantanapr



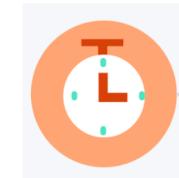
Gartner Hype Cycle
Gartner, July 2017

Serverless EVERYTHING... okay maybe not quite everything, but a lot!

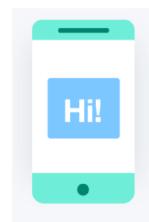
@csantanapr



Web App Backends



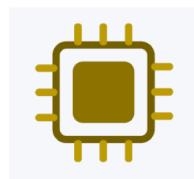
Scheduled Tasks



Mobile Backends



Conversational Scenarios



Internet of Things



Data Processing

FaaS platform to execute code in response to events

Open Source via Apache Incubation

<http://openwhisk.org> <http://github.com/apache/incubator-openwhisk>

Deploy via:

Docker compose, Ansible, Vagrant, Kubernetes via Helm

Don't Deploy ☺:



IBM Cloud Functions (Free 400GBs monthly) <http://bluemix.net/openwhisk>



Apache OpenWhisk

@csantanapr



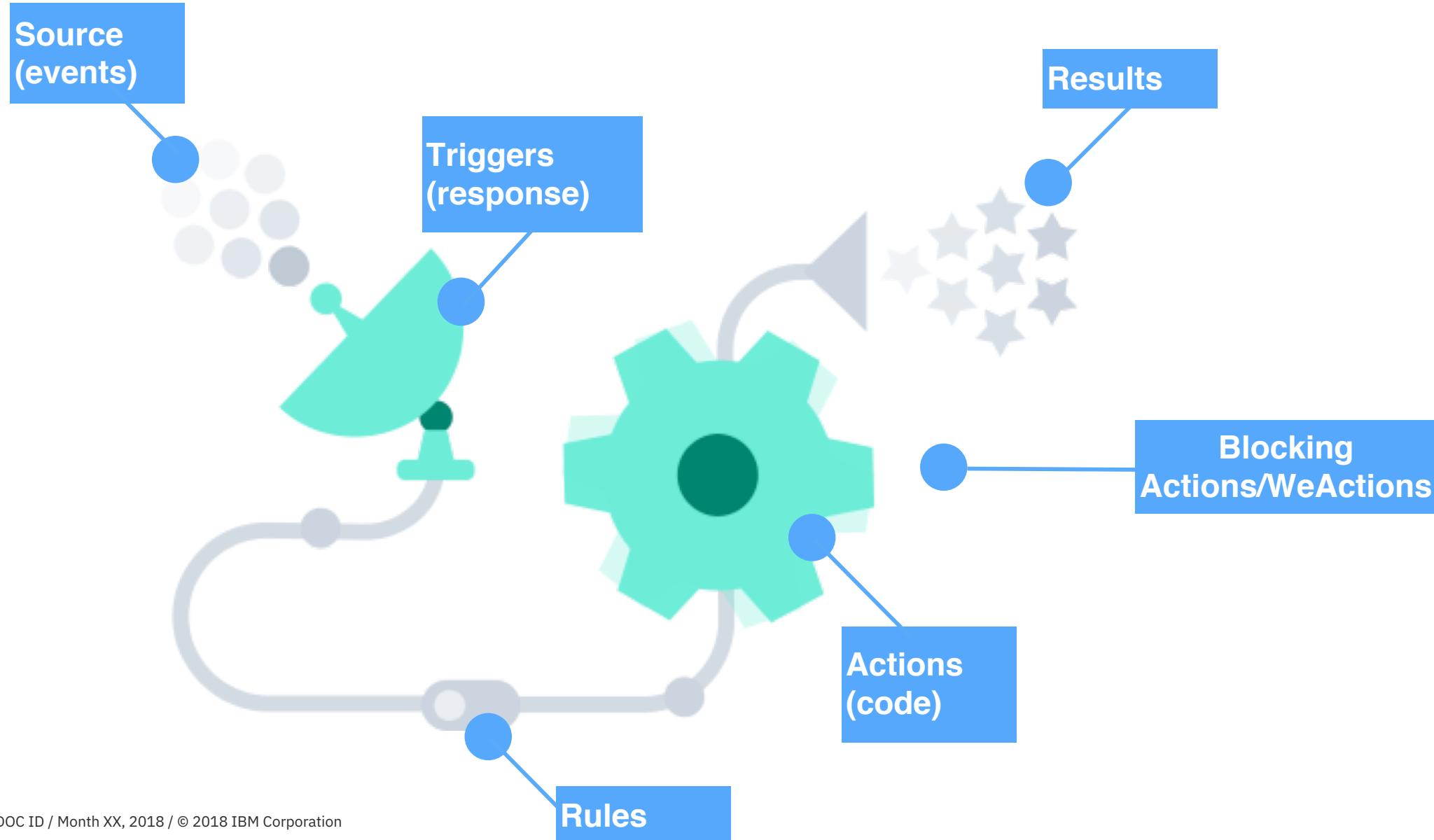
<https://github.com/apache/incubator-openwhisk>

www.openwhisk.org



Apache OpenWhisk Concepts

@csantanapr



Why do developers want to use **Serverless** to build **Mobile** **Backends** for their apps?

- Scalability
- Pay per Use
- Decreased Time to Market
- Event Driven Scenarios
- CPU Intensive Tasks offloaded to the Cloud
- Integration with Cloud Services



Swift for Backends

- Swift is fun Among the most loved languages of 2018
- Translate skills from the front end to the back end
- Shared application code between back end and front end, reuse of packages

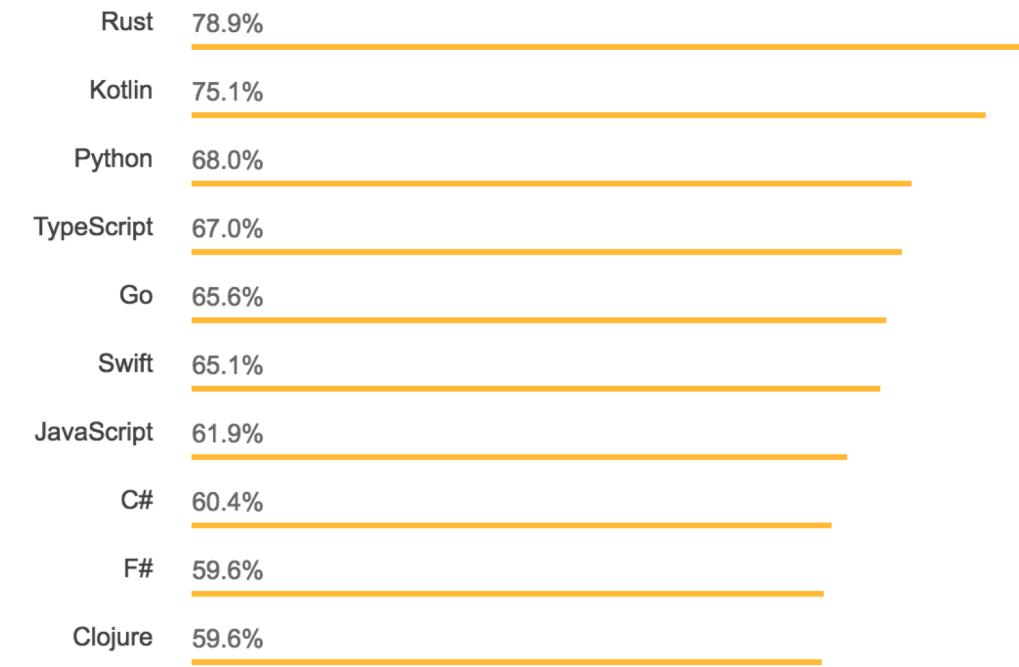
Outsource compute-intensive tasks to a powerful & scalable serverless platform and implement your actions even without changing the programming language.



Most Loved, Dreaded, and Wanted

Most Loved, Dreaded, and Wanted Languages

Loved Dreaded Wanted



Swift for Server-side programming

```
import Kitura

let router = Router()

router.get("/") { request, response, next in
    response.send("Hello world")
    next()
}

Kitura.addHTTPServer(onPort: 8080, with: router)
Kitura.run()
```



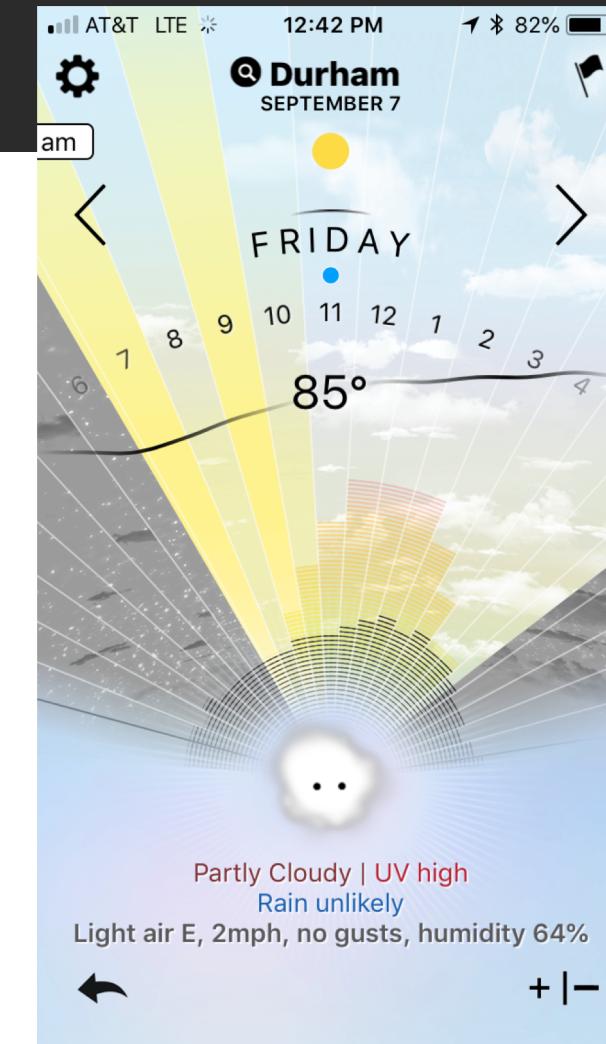
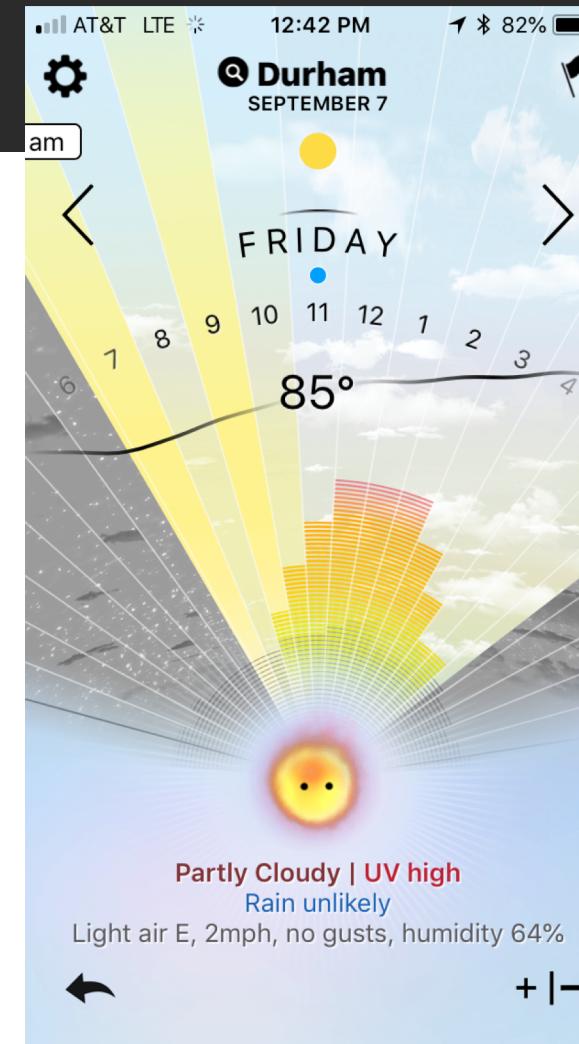
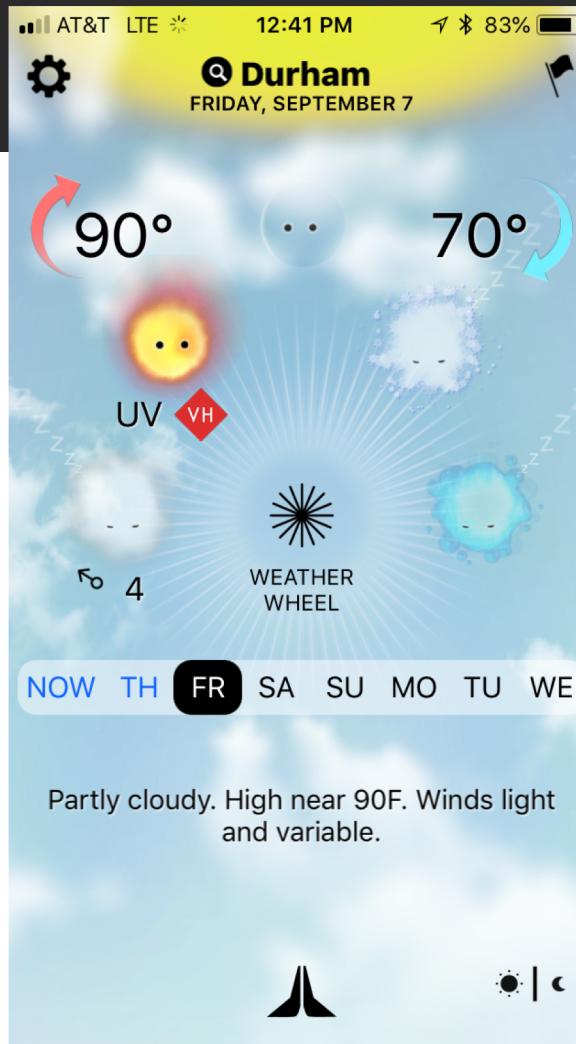
KITURA



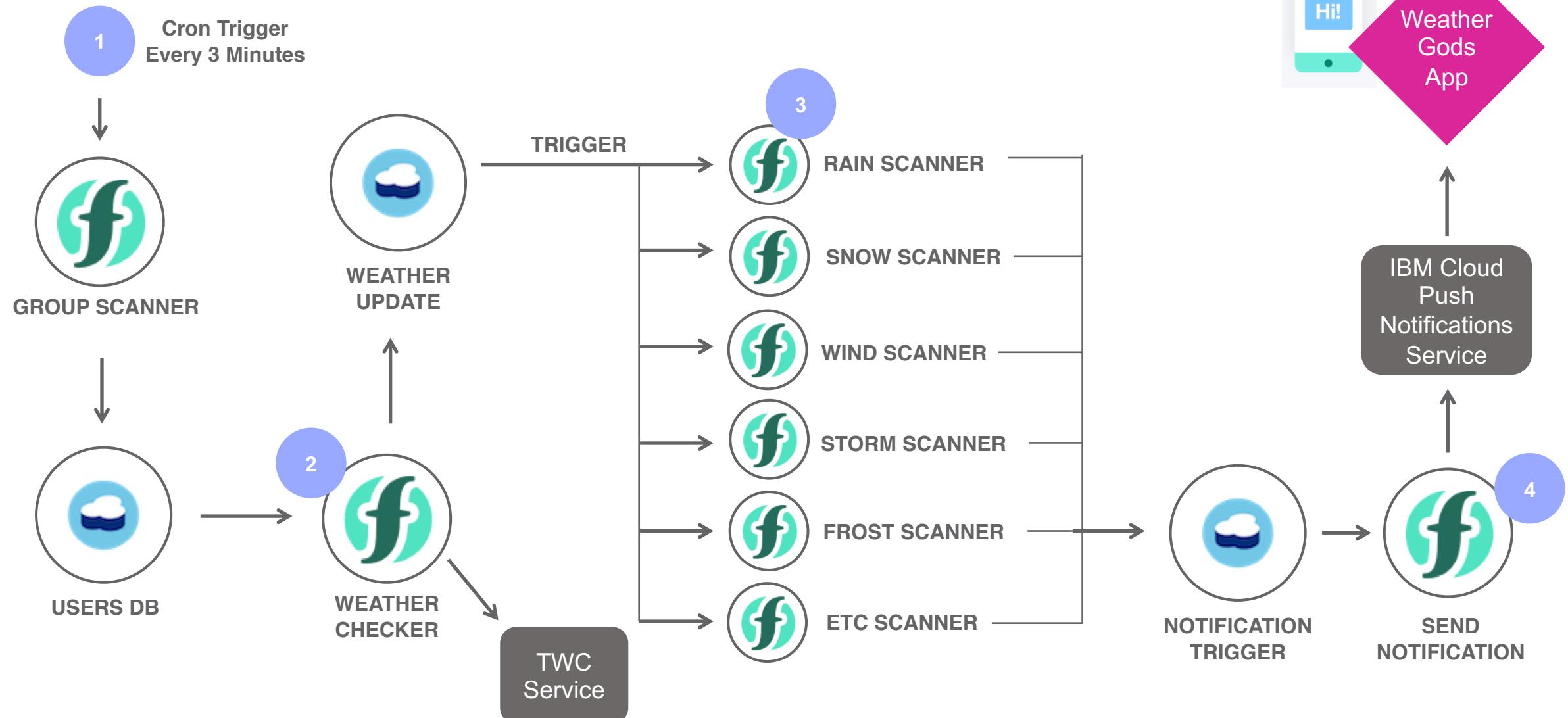
That's interesting and all, but wouldn't it be more fun to see a real App?

Weather Gods

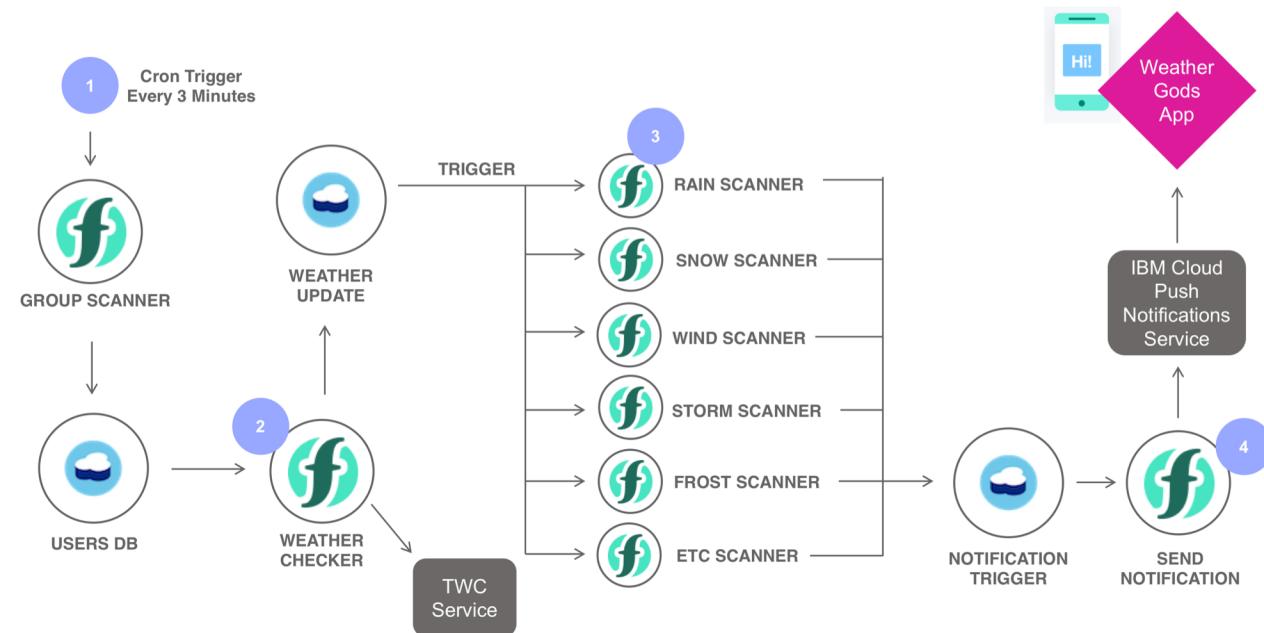
@csantanapr



Weather Gods, Backend Architecture



Weather Gods, Why Serverless?



Integrated Platform Services

- The Weather Company Data
- Cloudant NoSQL DB
- Notifications Service

Event Driven Code Execution

- User location updated
- New user in DB
- New weather item in DB

Scheduled Tasks

- Chronological Trigger every 3 minutes

Parallel Code Execution

- Run all “weather scanners” in parallel
- Scale these up and down as required based on user preferences

I'm ready to write my
next app. How do I **get**
started with Serverless
Swift?

IBM Cloud Functions, Swift Demo

@csantanapr

[input]

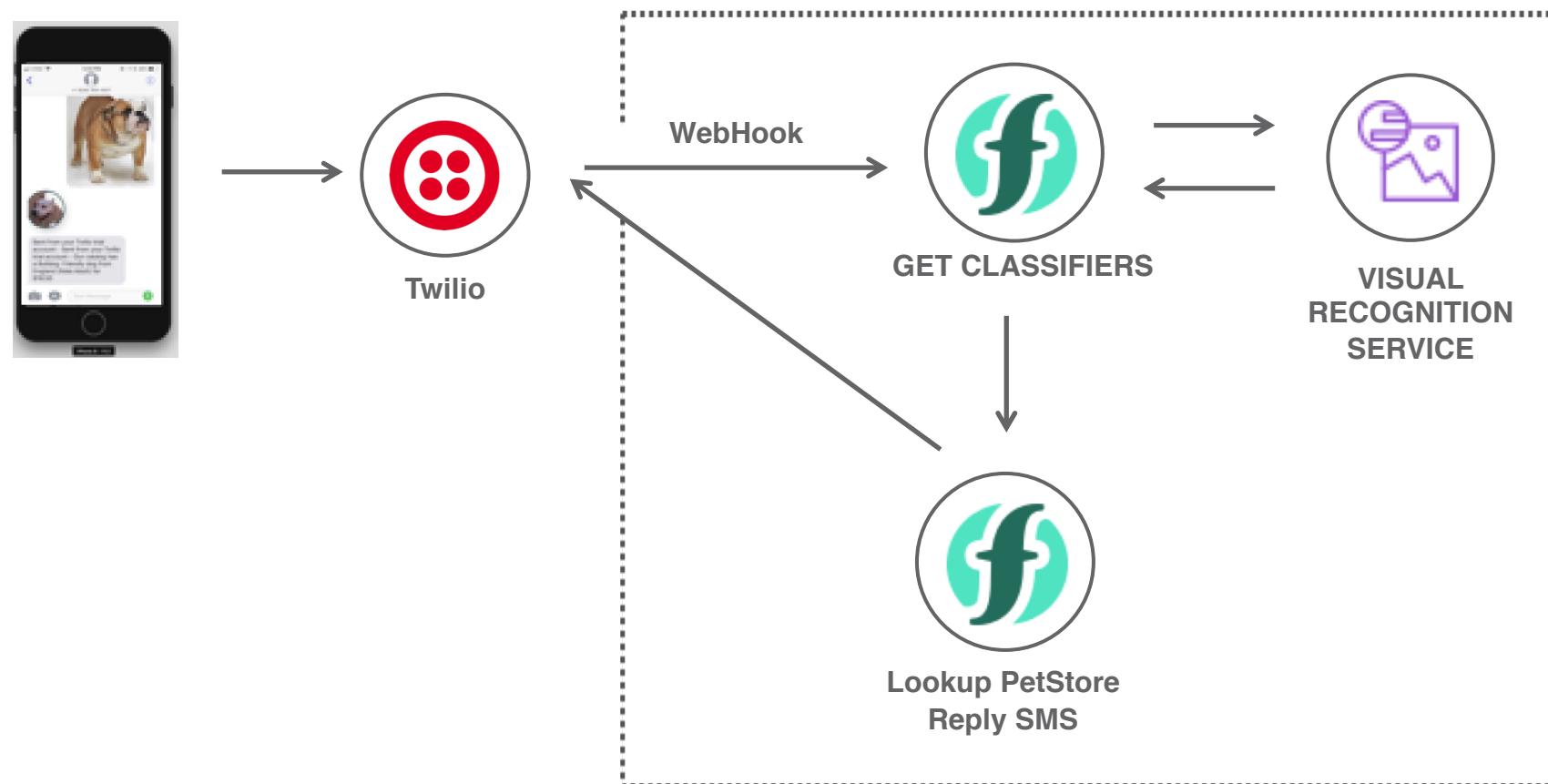


[output]

- “Bernese mountain dog”
- “dog”
- “animal”

IBM Cloud Functions, Swift Demo

@csantanapr



Demo.
SMS a Photo of Pet +
Image Analysis in
Swift.



Thank you

 bluemix.net/openwhisk

 twitter.com/csantanapr

 github.com/csantanapr/ato-serverless-swift

 developer.ibm.com/code



Feel free to reach out
for any questions or to
share what you're
working on.



get-classifier Action

Code  Swift 4 Edit mode - press **ESC** to exit Change Input  **Invoke** 

```
1 import VisualRecognitionV3
2
3 struct Input: Decodable {
4     let MediaUrl0: String?
5     let apiKey: String?
6     let __bx_creds: WatsonCredentials?
7 }
8
9 struct Output: Encodable {
10    let body: RecognitionTags
11 }
12
13 func classifyImage(param: Input, completion: @escaping (Output?, Error?) -> Void) {
14     // set up visual recognition sdk
15     let apiKey: String = param.apiKey ?? (param.__bx_creds?.watson_vision_combined.apikey)!
16     let imageUrl: String = param.MediaUrl0 ?? defaultImage
17     let visualRecognition = VisualRecognition(version: "2018-10-19", apiKey: apiKey)
18     let failure = { (error: Error) in print("err", error) }
19
20     // make call to visual recognition classify function
21     visualRecognition.classify(url: imageUrl, failure: failure) { classifiedImages in
22         let image = classifiedImages.images.first
23         let classifier = image?.classifiers.first
24         let classes = classifier?.classes
25         var resultTags = [RecognitionTag]()
26         for theclass in classes! {
27             resultTags.append(RecognitionTag(name: theclass.className, score: theclass.score ?? 0))
28         }
29         let result = Output(body: RecognitionTags(tags: resultTags, imageUrl: imageUrl))
30         completion(result, nil)
31     }
32 }
33 }
```

reply-sms Action

Code  Swift 4

Edit mode - press **ESC** to exit

Change Input 

Invoke 

```
1
2 import Foundation
3
4 struct TwimlHeaders: Encodable {
5     let content_type = "application/xml"
6 enum CodingKeys: String, CodingKey {
7     case content_type = "content-type"
8 }
9 }
10 struct TwimlOutput: Encodable {
11     let body: String
12     let headers = TwimlHeaders()
13 }
14
15 func replySMSMessage(param: ImageTags, completion: @escaping (TwimlOutput?, Error?) -> Void) {
16     let petDetected = param.body?.tags[0].name ?? "no pet found"
17     let petImage: String = param.body?.imageUrl ?? "no image"
18     let petFound = lookupPet(pet: petDetected)
19     let body = getBodyMessage(body: "\(petDetected), \(petFound.description), \(petFound.price)", media: petImage)
20     let twimlM = TwimlOutput(body: body)
21     completion(twimlM, nil)
22 }
23
24
25
26
27
```

pet-store-webhook Sequence Action

Sequence Actions		Add	Change Input	Invoke
ACTIONS	PACKAGE			
get-classifier	Default Package			
reply-sms	Default Package			

Twilio setup - get a Number

The screenshot shows the Twilio Active Numbers page. On the left sidebar, under the 'Phone Numbers' section, 'Active Numbers' is selected and highlighted with a red arrow. In the center, the 'Active Numbers' heading is displayed above a 'CLICK + TO BUY NEW NUMBER' button. A red circle with a plus sign is overlaid on this button. Below it, there's a search bar with dropdown menus for 'Number' and 'Voice URL', and a 'Filter' button. The main table has columns for 'NUMBER', 'FRIENDLY NAME', 'CAPABILITIES', and 'CONFIGURATION'. A row for a number '+1 919 355 [REDACTED]' from 'Apex, NC' is shown. In the 'CONFIGURATION' column, the 'Voice' and 'Messaging' POST URLs are listed: 'Voice POST: https://demo.twilio.com/welcome/voice/' and 'Messaging POST: https://openwhisk.ng.bluemix.net/api/v1/web/csa.. store-webhook.http'. The 'Messaging' URL is highlighted with a red box.

NUMBER	FRIENDLY NAME	CAPABILITIES	CONFIGURATION
+1 919 355 [REDACTED]	(919) 355-[REDACTED]	[Phone icon] [Text icon] [Image icon]	Voice POST: https://demo.twilio.com/welcome/voice/ Messaging POST: https://openwhisk.ng.bluemix.net/api/v1/web/csa.. store-webhook.http

Twilio setup – setup WebHook

The screenshot shows the Twilio Active Numbers page. On the left sidebar, under the "Phone Numbers" section, the "Active Numbers" option is selected. A red arrow points from the top of the sidebar down to the "Active Numbers" link. Another red arrow points from the "Manage Numbers" link down to the "# Buy a Number" button. A third red arrow points from the "# Buy a Number" button down to the phone number "+1 919 355".

The main page title is "Active Numbers". Below it is a button labeled "CLICK + TO BUY NEW NUMBER". There are two dropdown filters: "Number" and "Voice URL". A red "Filter" button is also present.

NUMBER	FRIENDLY NAME	CAPABILITIES	CONFIGURATION
(919) 355-XXXX	Apex NC		Voice POST: https://demo.twilio.com/welcome/voice/ Messaging POST: https://openwhisk.ng.bluemix.net/api/v1/web/csantana-store-webhook.http

The screenshot shows the Twilio Messaging configuration page. At the top, there is a "CONFIGURE WITH" dropdown set to "Webhooks, TwiML Bins, Functions, Studio, or Proxy". A red box highlights the "A MESSAGE COMES IN" section, which includes a "Webhook" dropdown set to "https://openwhisk.ng.bluemix.net/api/v1/web/csantana-store-webhook.http" and an "HTTP POST" dropdown.

Below this, there is another section for "PRIMARY HANDLER FAILS" with a "Webhook" dropdown set to "https://demo.twilio.com/welcome/sms/reply/" and an "HTTP POST" dropdown.