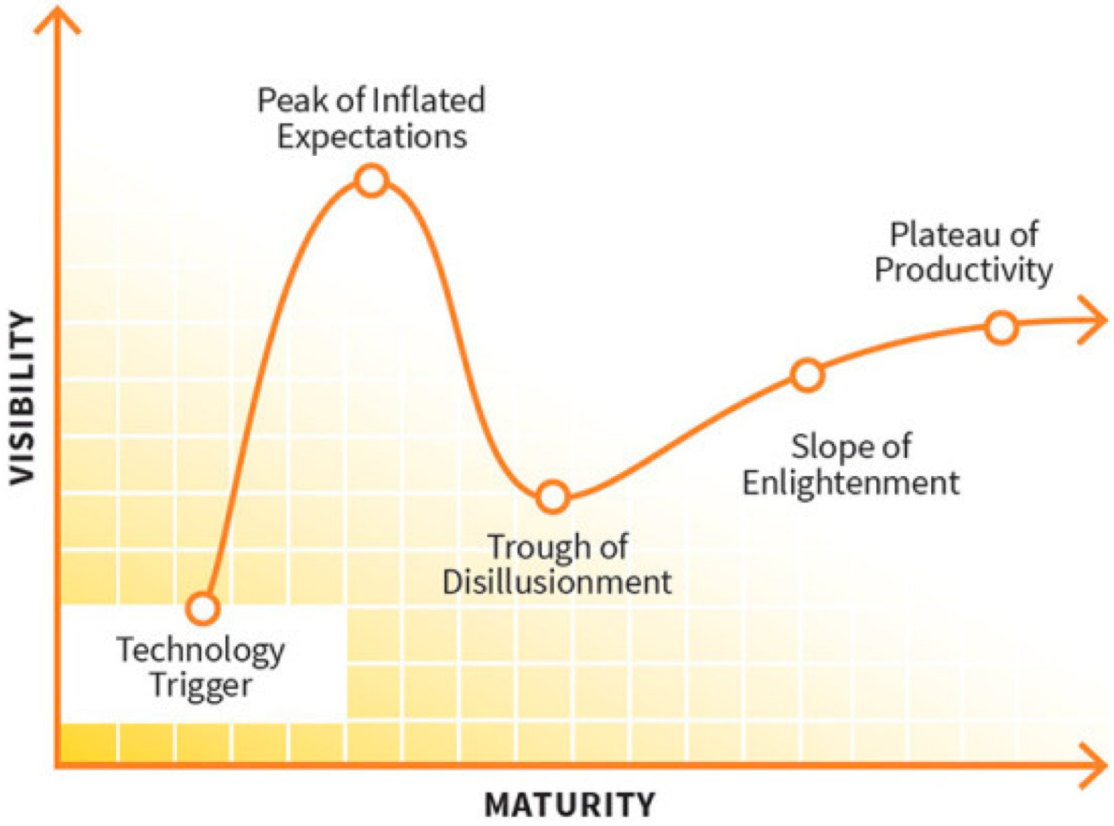
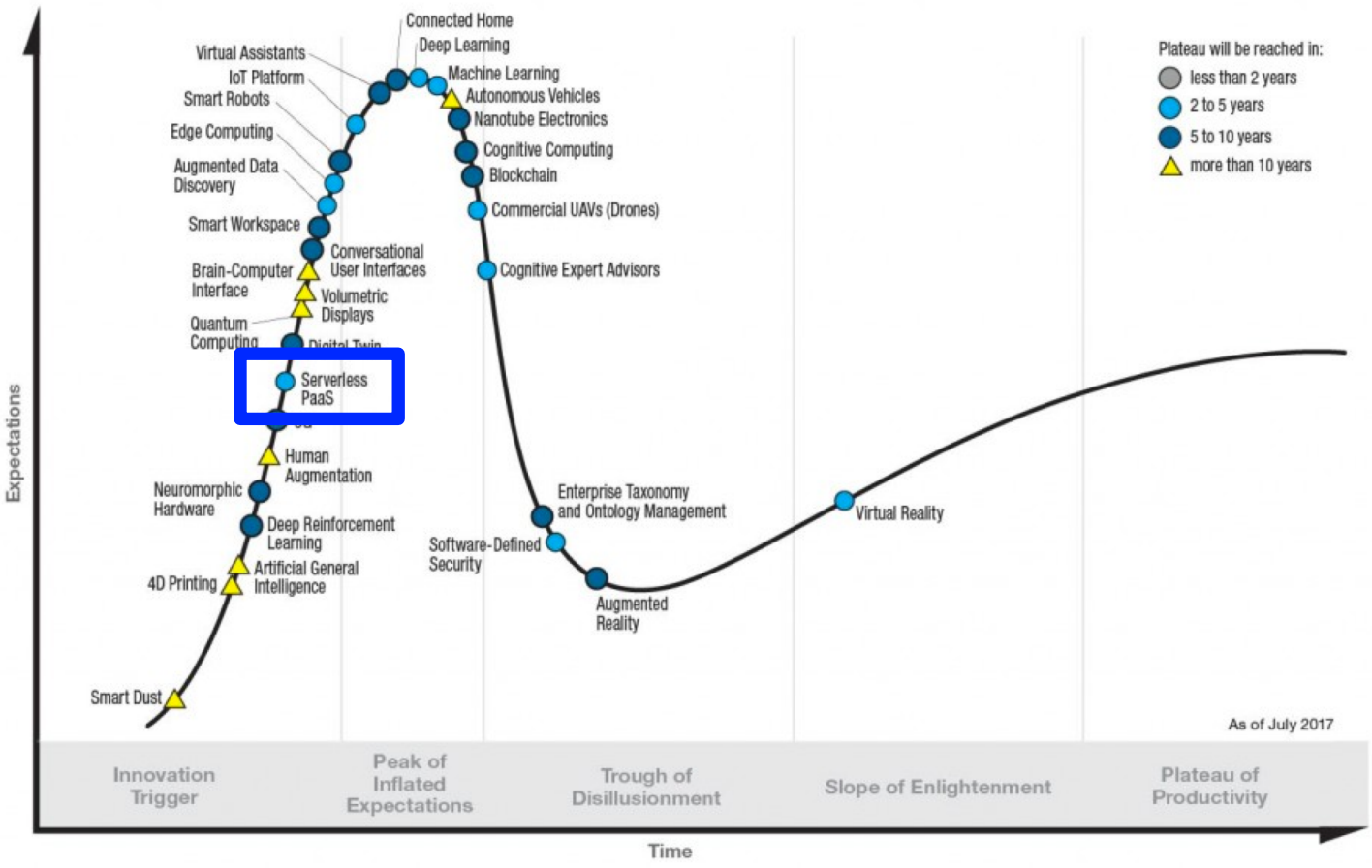
Serverless Swift - OpenWhisk

Carlos Santana
IBM Software Engineer (STSM)
@csantanapr

Serverless EVERYTHING...

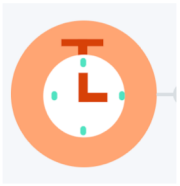




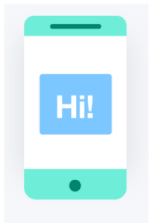
Gartner Hype Cycle
Gartner, July 2017

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

Web App Backends

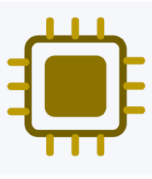


Scheduled Tasks



Mobile Backends

Conversational Scenarios



Internet of Things

Data Processing

Apache OpenWhisk

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 ☺:

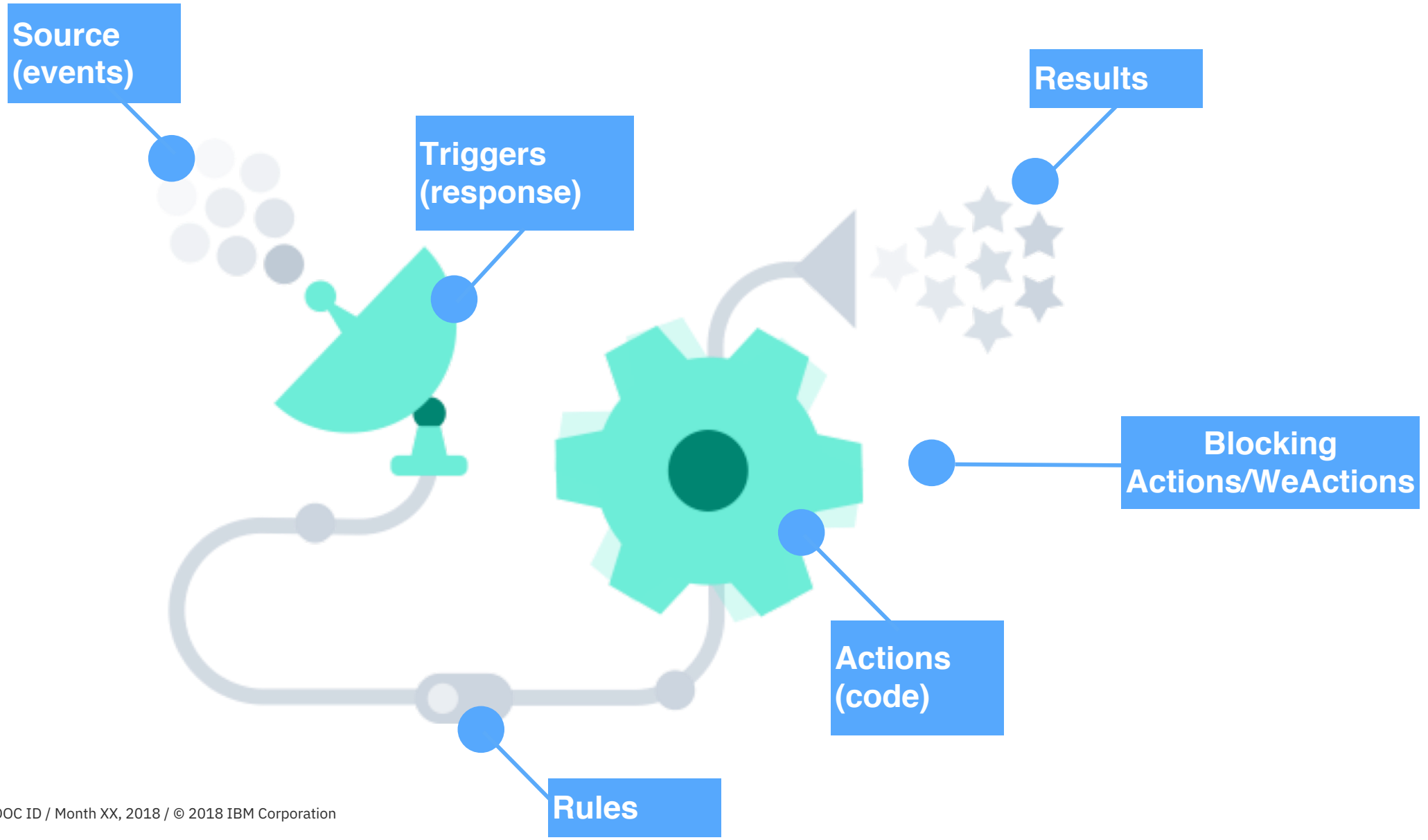
IBM Cloud Functions (Free 400GBs monthly) http://bluemix.net/openwhisk

Apache OpenWhisk

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

www.openwhisk.org

Apache OpenWhisk Concepts



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

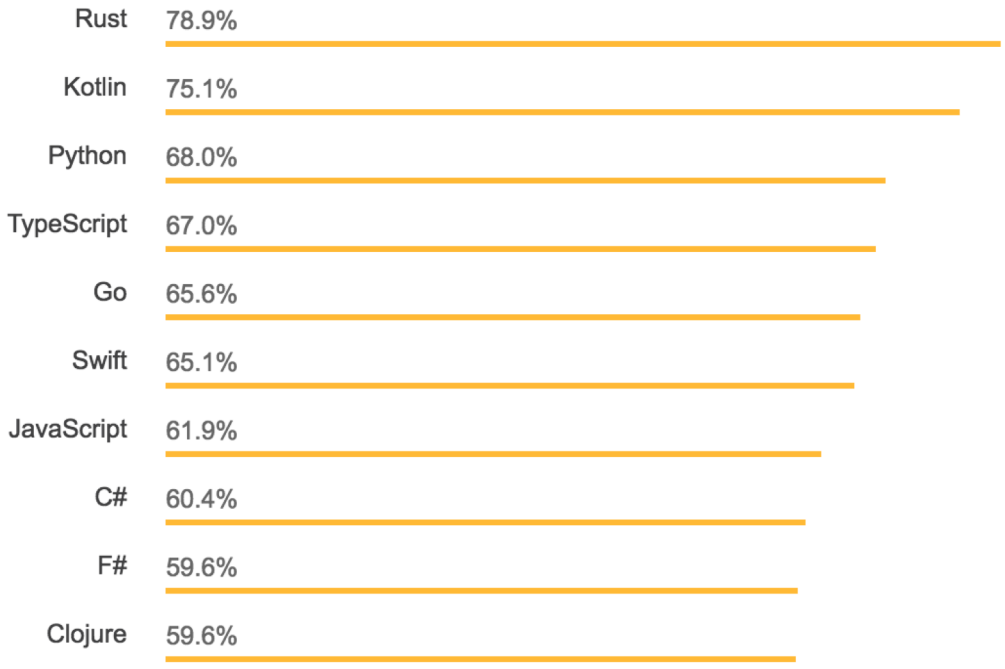
- Swift is fun Among the most loved languages of 2018
- Translate skills from the front end to the back end
- Shared application code between back end and front end, reuse of packages

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

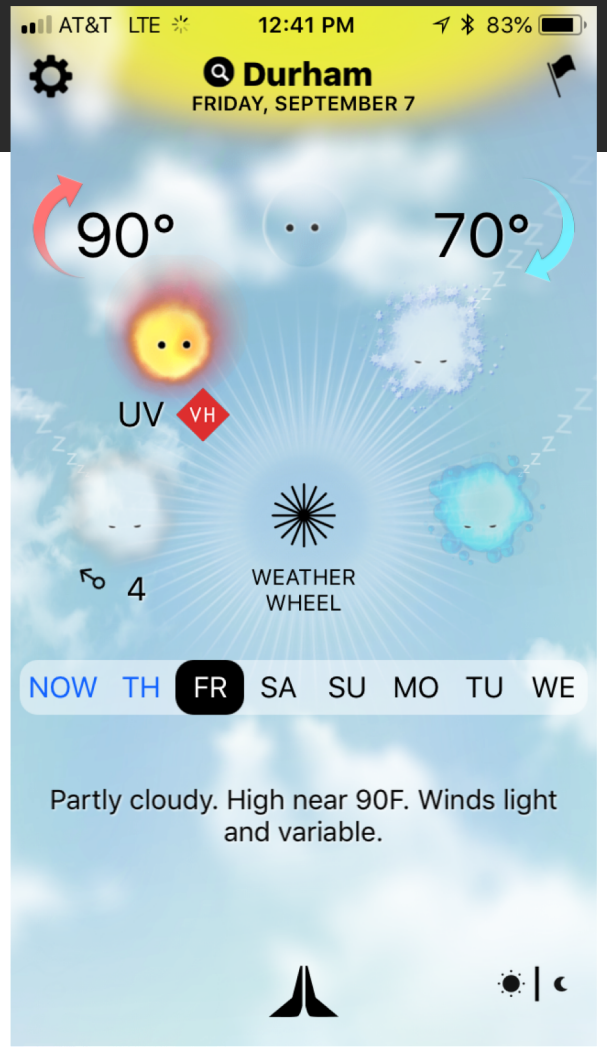


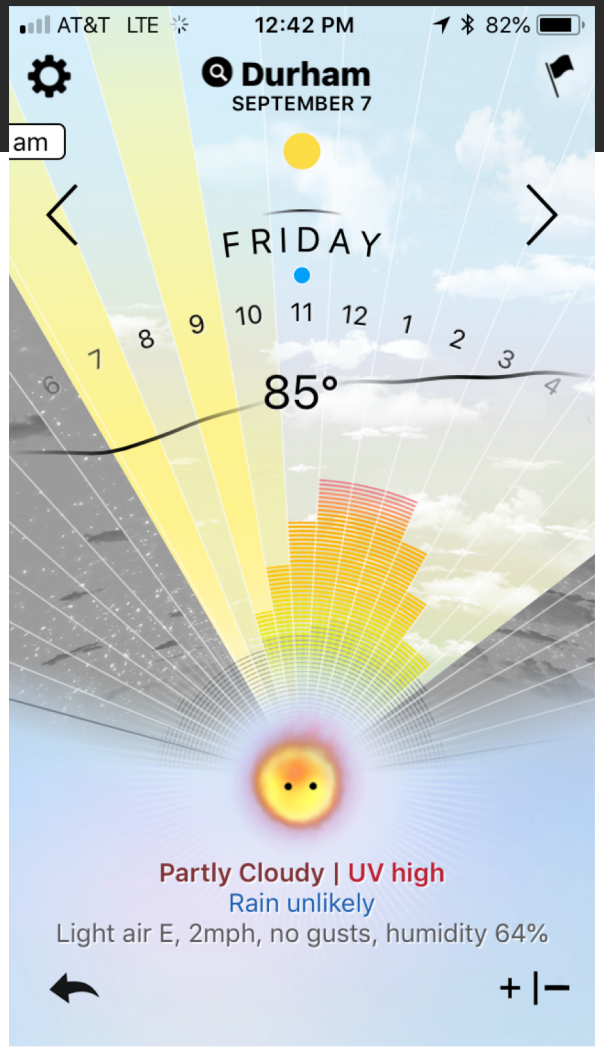
Swift for Server-side programming

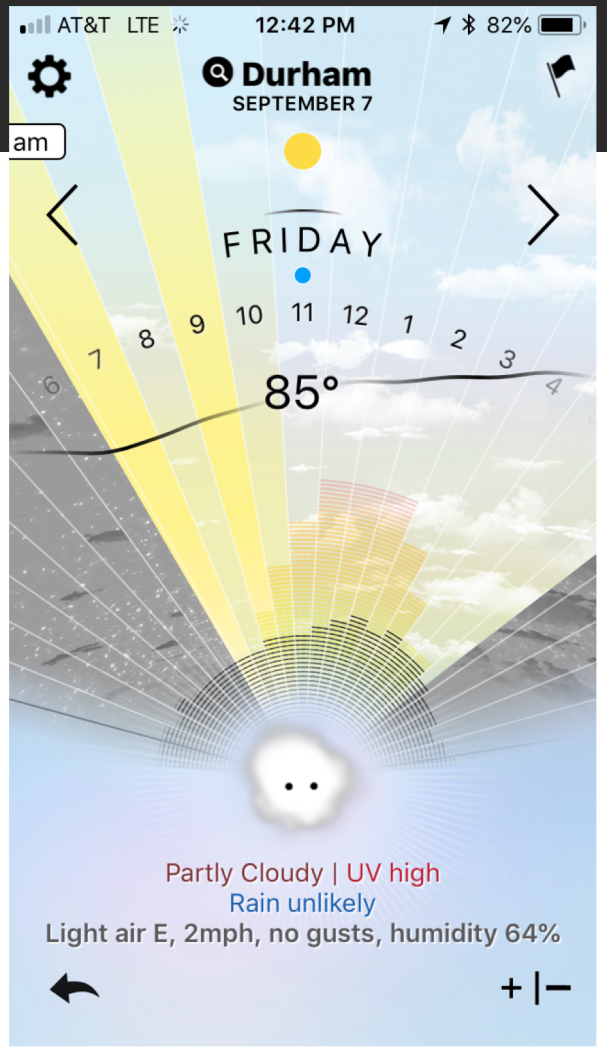
```swift
import Kitura

let router = Router()

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

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

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

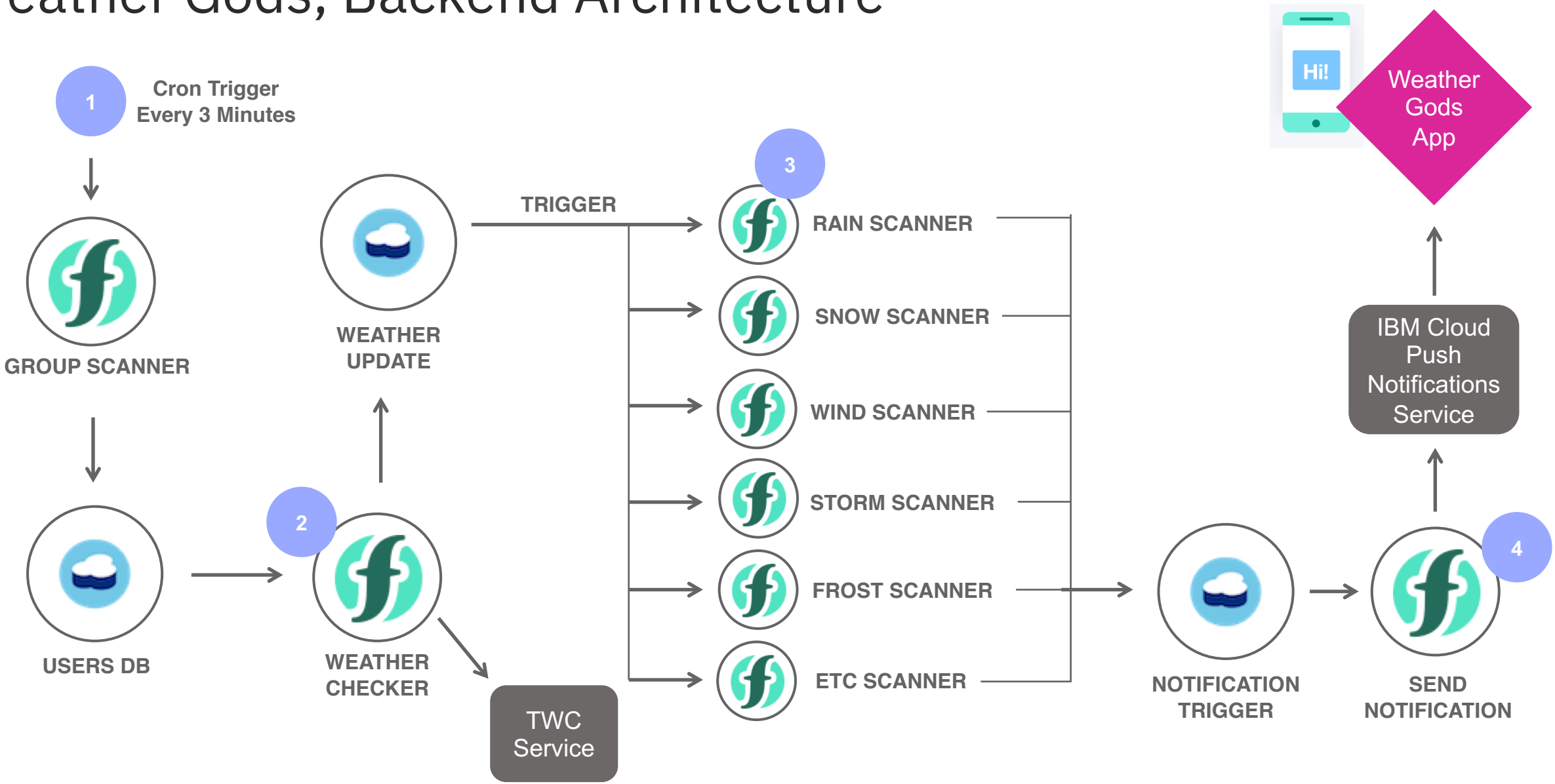
Weather Gods



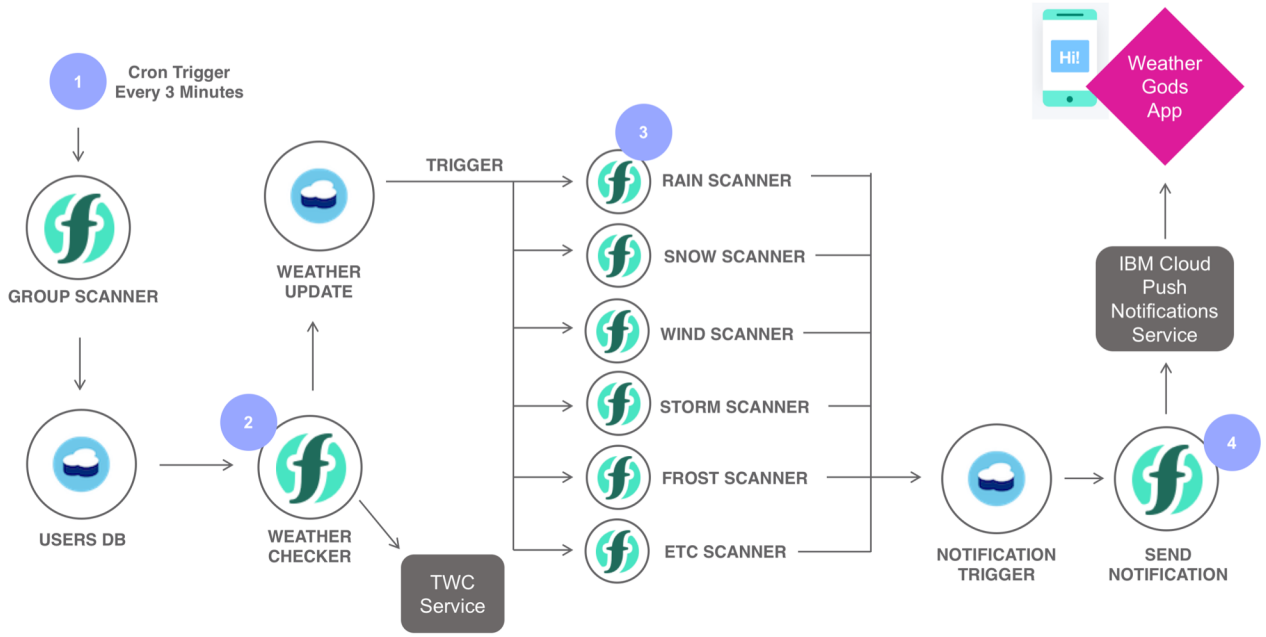




Weather Gods, Backend Architecture



Weather Gods, Why Serverless?



Integrated Platform Services

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

Scheduled Tasks

- Chronological Trigger every 3 minutes

Event Driven Code Execution

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

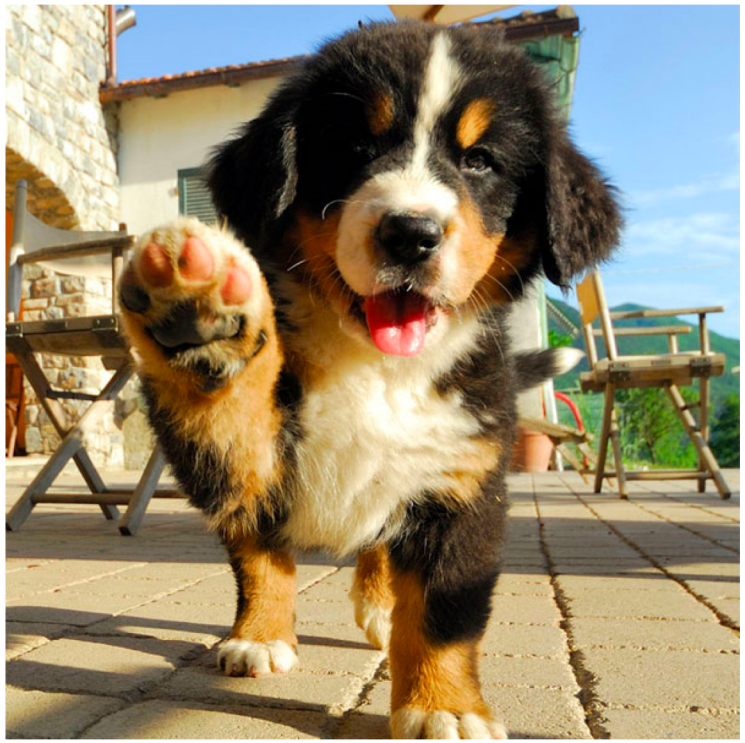
Parallel Code Execution

- 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

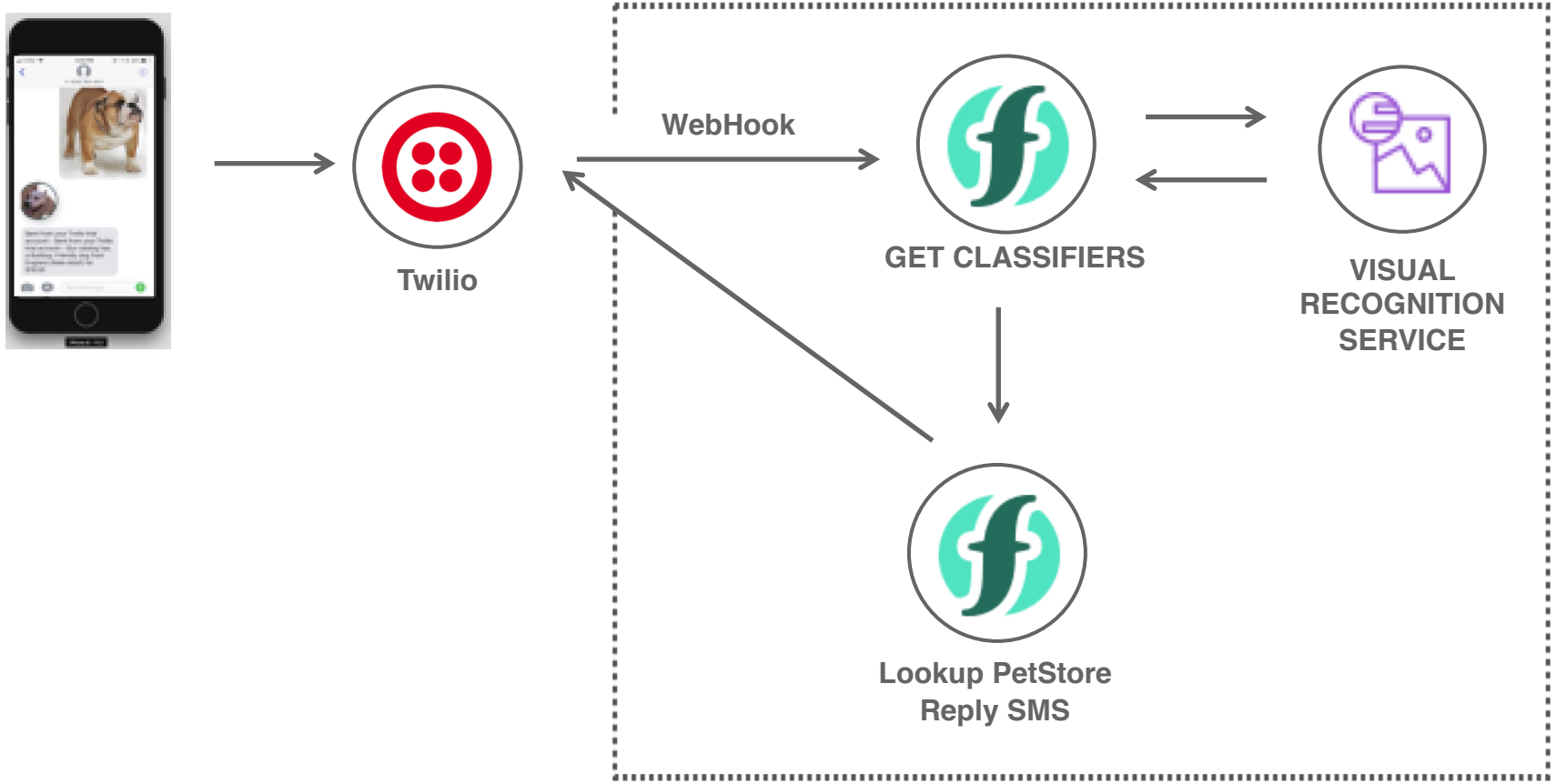
[input]

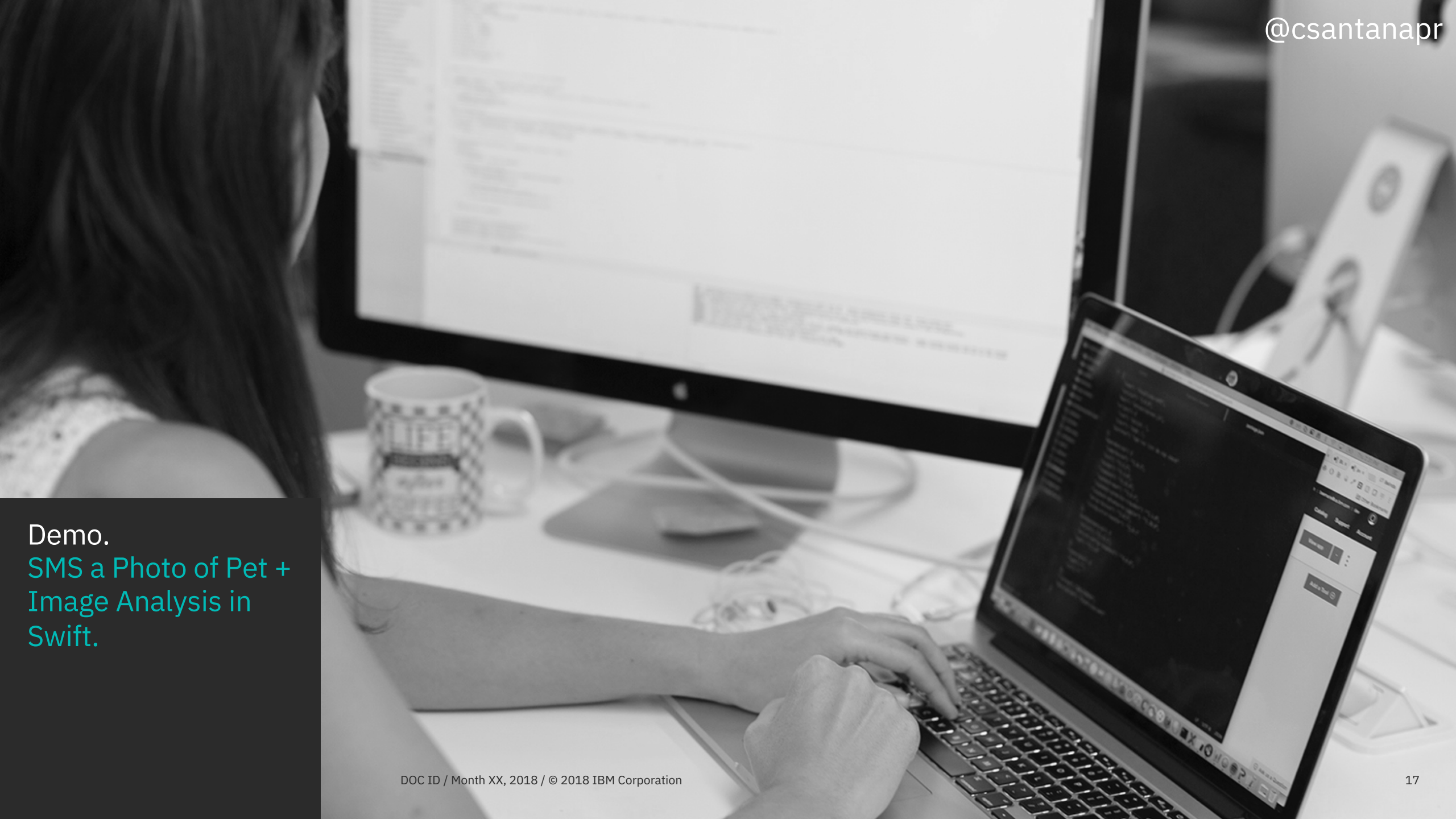


[output]

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

IBM Cloud Functions, Swift Demo



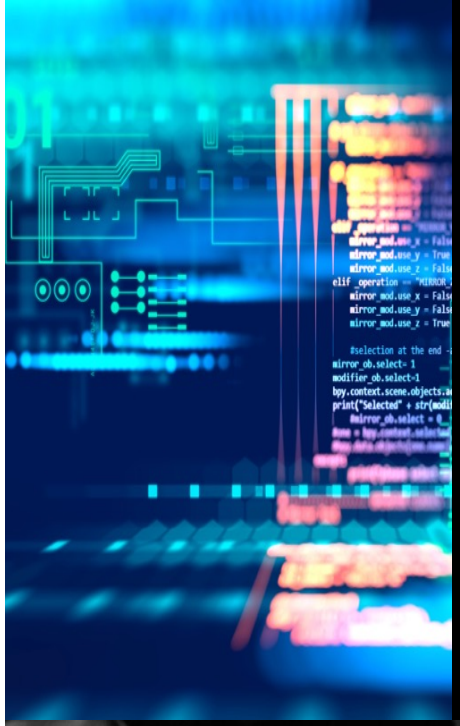


Demo.

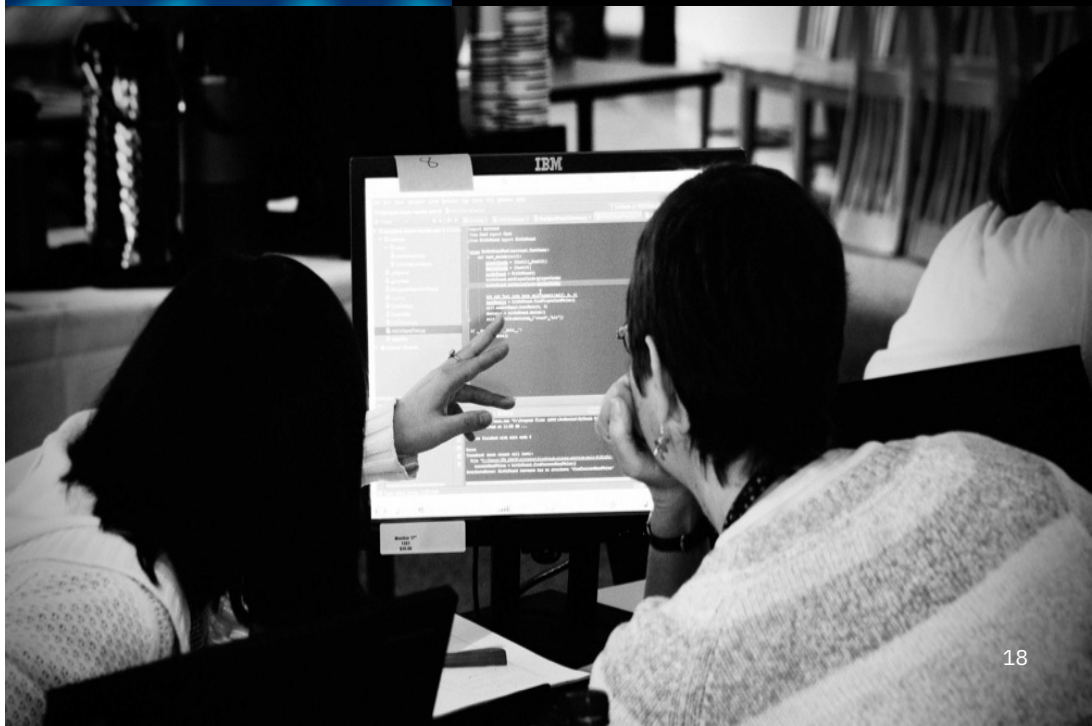
SMS a Photo of Pet + Image Analysis in Swift.

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

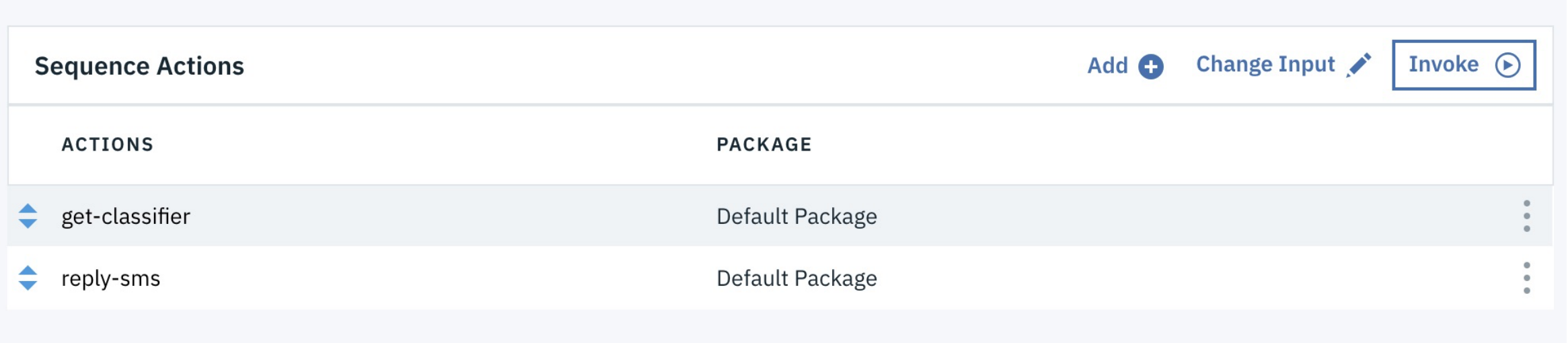
```swift
import VisualRecognitionV3

struct Input: Decodable {
    let MediaUrl0: String?
    let apiKey: String?
    let __bx_creds: WatsonCredentials?
}

struct Output: Encodable {
    let body: RecognitionTags
}

func classifyImage(param: Input, completion: @escaping (Output?, Error?) -> Void) {
    // set up visual recogntiion sdk
    let apiKey: String = param.apiKey ?? (param.__bx_creds?.watson_vision_combined.apikey)!
    let imageUrl: String = param.MediaUrl0 ?? defaultImage
    let visualRecognition = VisualRecognition(version: "2018-10-19", apiKey: apiKey)
    let failure = { (error: Error) in print("err", error) }

    // make call to visual recognition classify function
    visualRecognition.classify(url: imageUrl, failure: failure) { classifiedImages in
        let image = classifiedImages.images.first
        let classifier = image?.classifiers.first
        let classes = classifier?.classes
        var resultTags = [RecognitionTag]()
        for theclass in classes! {
            resultTags.append(RecognitionTag(name: theclass.className, score: theclass.score ?? 0))
        }
        let result = Output(body: RecognitionTags(tags: resultTags, imageUrl: imageUrl))
        completion(result, nil)
    }
}
```

reply-sms Action

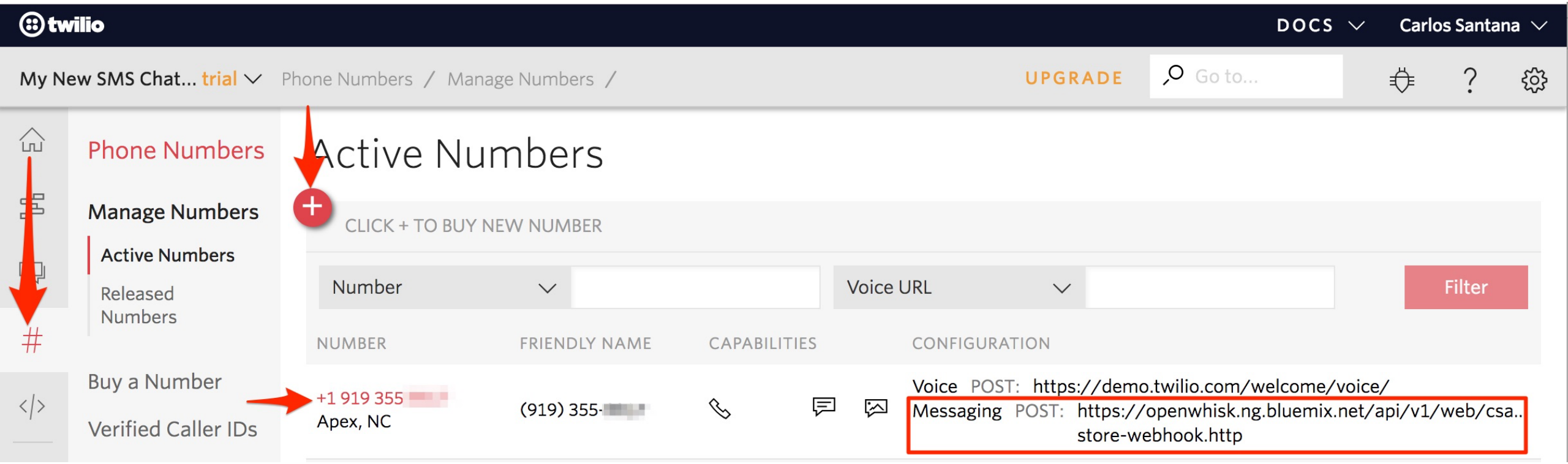
Code Swift 4 Edit mode - press ESC to exit Change Input Invoke

```swift

import Foundation

struct TwimlHeaders: Encodable {
    let content_type = "application/xml"
    enum CodingKeys: String, CodingKey {
        case content_type = "content-type"
    }
}
struct TwimlOutput: Encodable {
    let body: String
    let headers = TwimlHeaders()
}

func replySMSMessage(param: ImageTags, completion: @escaping (TwimlOutput?, Error?) -> Void) {
    let petDetected = param.body?.tags[0].name  ?? "no pet found"
    let petImage: String = param.body?.imageUrl ?? "no image"
    let petFound = lookupPet(pet: petDetected)
    let body = getBodyMessage(body: "\(petDetected), \(petFound.description), $\(petFound.price)", media: petImage)
    let twimlM = TwimlOutput(body: body)
    completion(twimlM, nil)
}
```

pet-store-webhook Sequence Action



Twilio setup - get a Number



Twilio setup – setup WebHook