Functions-as-a-Service & Workflows

Building a server-less application using FaaS Workflows

SPEAKER: DIEGO MARTIN

SPLAB

INDEX OF CONTENTS

- Stage 0
    - Functions-as-a-Services, a brief introduction
    - FaaS Workflows
    - Working with the technology: providers
- Stage 1
    - Building a ToDo-App from scratch

A BRIEF INTRO

Functions-as-a-Services, a brief introduction

Funtion-as-a-Service (aka FaaS) is a specialized kind of Platform-as-a-Service (aka PaaS) in witch everything, asides the functional code itself, is abstracted and handled by the provider

It can be modeled like the image above in which both sides (the input and the output of the function) can be anything the provider allows you: databases, HTTP triggers, events, files, devices, etc

FaaS WORKFLOWS

FaaS Workflows

FaaS Workflows arise from the need to connect two (or more) functions in a logic and function-agnostic way

This connections can be as simple as a sequence or chain, or evolve into a more complex relation of functions that can result in a complete state machine (which could be considered as States-as-a-Service)

PROVIDERS

Working with the technology

Until today, there's a few providers offering this technology:

- AWS Step Functions
- IBM Composer
- Platform9 Fission Workflows
- Microsoft Azure Logic Apps
- Oracle Fn Project (Fn Flow)

Working with the technology

AWS Setp Functions

```
{
    "Comment": "A simple minimal example of the States language",
    "StartAt": "Hello World",
    "States": {
    "Hello World": {
      "Type": "Task",
      "Resource": "arn:aws:lambda:us-east-1:123456789012:function:HelloWorld",
      "End": true
    }
  }
}
```

Fission Workflows

```
apiVersion: 1
output: WhaleWithFortune
tasks:
  GenerateFortune:
    run: fortune
    inputs: "{$.Invocation.Inputs.default}"

  WhaleWithFortune:
    run: whalesay
    inputs: "{$.Tasks.GenerateFortune.Output}"
    requires:
    - GenerateFortune
```

IBM Composer

```
composer.try(
    composer.sequence(
        'myWatsonTranslator/languageId',
        composer.if(
            p => p.language !== 'en',
            composer.sequence(
                p => ({translateFrom: p.language, translateTo: 'en', payload: p.payload}),
                'myWatsonTranslator/translator'
            ),
            composer.sequence(
                p => ({text: p.payload}),
                'en2shakespeare'
            )
        )
    ),
    err => ({payload: 'Sorry we cannot translate your text'})
)
```

Working with the technology

AWS Step Functions

Graph Code

Success Failed Needs retry In progress

Start
FetchAnOrder
RegionChoice
CreateOrderA
CreateOrderB
OrderOK
DatabaseError
UnservedRegion
ProcessOrder
NoOrderPossible
End

Execution Details

Info Input Output

Execution Status
Succeeded

Started
Nov 20, 2016 9:58:28 AM

Closed
Nov 20, 2016 9:58:32 AM

Step Details

ID	Type	Timestamp
1	ExecutionStarted	Nov 20, 2016 9:58:28 AM
2	TaskStateEntered	Nov 20, 2016 9:58:28 AM
3	LambdaFunctionScheduled	Nov 20, 2016 9:58:28 AM

IBM Composer

APP myApp v0.0.1

You are in edit mode, viewing the **currently deployed version**

```
1 // try typing "composer." to begin your composition
2 composer.sequence(args => ({msg: `hello ${args.name}!`}))
```

Entry
args => ({msg: `hello ${args.name}!`})
Exit

Deploy Revert

AZURE Logic Apps

When a feed item is published

Condition

And

Feed su... × contains Microsoft

Add dynamic content

Add

If true

Office 365 Outlook - Send an email

Add an action More

If false

Add an action More

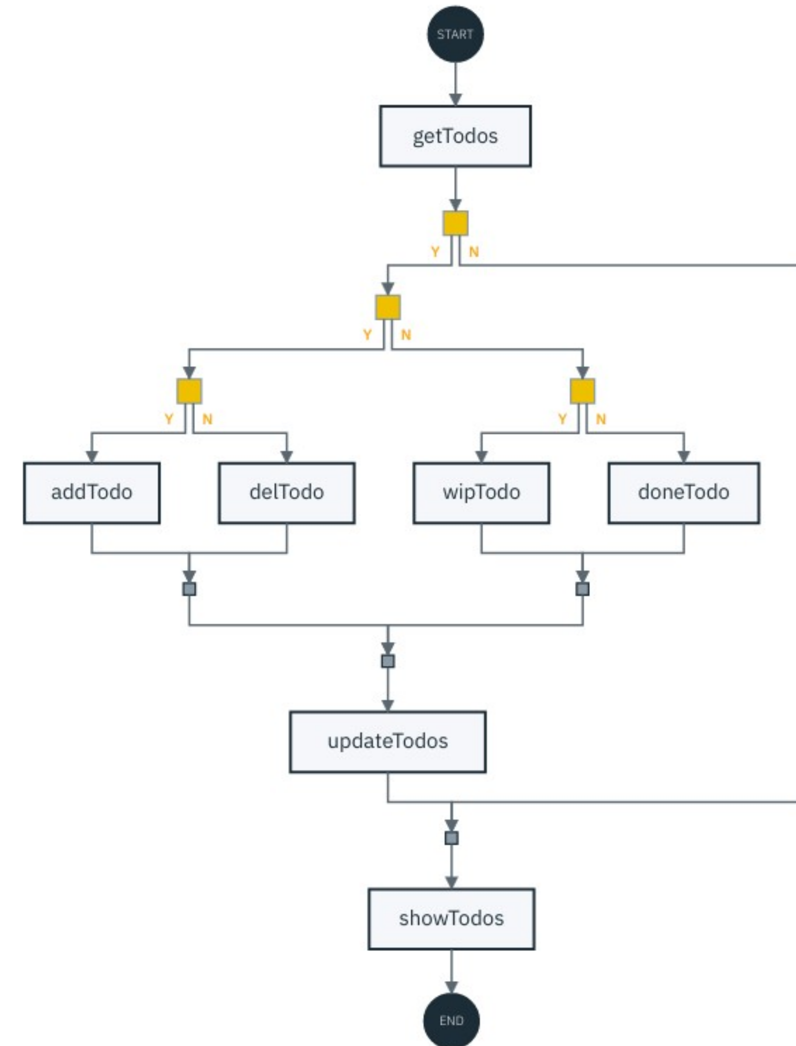
The Hands-on!

The composition

7 Simple functions

2 kind of connectors:
- sequence
- if-then-else

```
composer.sequence('FWTt/getTodos',
    composer.if(params => (!!params.action) && (params.action.constructor === Object),
        composer.sequence(
            composer.if(params => params.action.add || params.action.del,
                composer.if(params => params.action.add,
                    'FWTt/addTodo',
                    'FWTt/delTodo'
                ),
                composer.if(params => params.action.wip,
                    'FWTt/wipTodo',
                    'FWTt/doneTodo'
                )
            ),
            'FWTt/updateTodos'
        ),
        composer.empty()
    ),
    'FWTt/showTodos'
);
```



The Hands-on!

THANKS!