Go serverless with Azure Functions

Andrea Giunta

Azure MVP

Senior architech @OrangeDev

CTO @ ICSoftsolutions

Agenda

- A quick brief on cloud
- What is Serverless
- Introduction to Azure Functions
- Demo (Hello from the clouds)
- A «where am I running» application
- Demo
- Q&A

The cloud, but what is the cloud?

- Cloud computing gain access to hardware and software resources, based on software architecture needs.
- The main benefits are
- Scaling and redudancy
  - Vertical scaling
  - Horizontal scaling
  - Geographic redundancy
- Cost
  - Pay as you go plans
  - Elastic management of resource

Cloud nomenclatures

We can differentiate cloud services in three principal segments

- IaaS
- Paas
- SaaS

But what happen really on PaaS layer?

- PaaS is generic, we have different platforms, so we have DaaS, MBaaS, HaaS, FaaS and others

What is Serverless?

Nothing to get scared about, really.

End user (developer) doesn’t have to think about were the application will run (on virtual hardware level)

How you can achieve this?

Serverless and Azure Functions

Serverless services can be placed inside Function as a Service segment inside PaaS

Serverless architecture allow us to make improve a decomposition of our backend and initialize resources only for our needs, running our code on demand

Azure functions is, of course, a FaaS that allow your app to run on demand

How can I write and deploy an AF?

Visual studio 2015 has a preview toolkit for function

Visual studio 2017 still not (at 8/4/17)

You can handle and write functions from azure portal

Or?

Since A.F supports Git for CI/CD you can use this as «workaround»

What if i want to make something big?

Call a Function technically is like calling an API, main difference is architecture

We can also mix these architecture in same scenario (Mixed architecture)

Let’s see what could be needed in a development phase..

…making a serverless vehicle managment system

Authentication

Requests to functin is authenticate via the function api, but our users should authenticate themselves.

But, in a serverless scenario how i should remember my user status?

Viable solutions are provided by tokens especially (IMHO) JWT

JWT – Json Web Token

A JWT is made by three sections

- Encrypt algorithm information
- Claims
- Signature

I suggest this kind of token because it contains all information on user (claims)

Sql server

We can set connection string atop of the functions

So we can connect to any sql server we should need

Let’s see where this can be configured and how can be used

SQL - Demo

Decomposition

Since microservices and serverless could coincide, we can use a few techniques from «both» world

Decomposition is a common base technique to define how we should develop our architecture, the main options are

- By name (a service for each entity: e.g. users, addressbook)
- By verbs (a service for a business operation: e.g checkout, login)

Our service actors

- Vehicle
- Assignee
- Maintenance note

With following relations

- Vehicle 1-N Maintenance note

I’ll choose a verb decomposition

Vehicle

- Id
- Name
- Model
- Type
- License number
- Fuel type

- Function to access and insert new vehicle

Assignee

- Id
- Full name
- Mobile phone
- Email

- Function to access and insert new Assignee

Maintenance note

- Id
- Date
- VehicleId
- Note
- Cost

- Function to access and insert new note