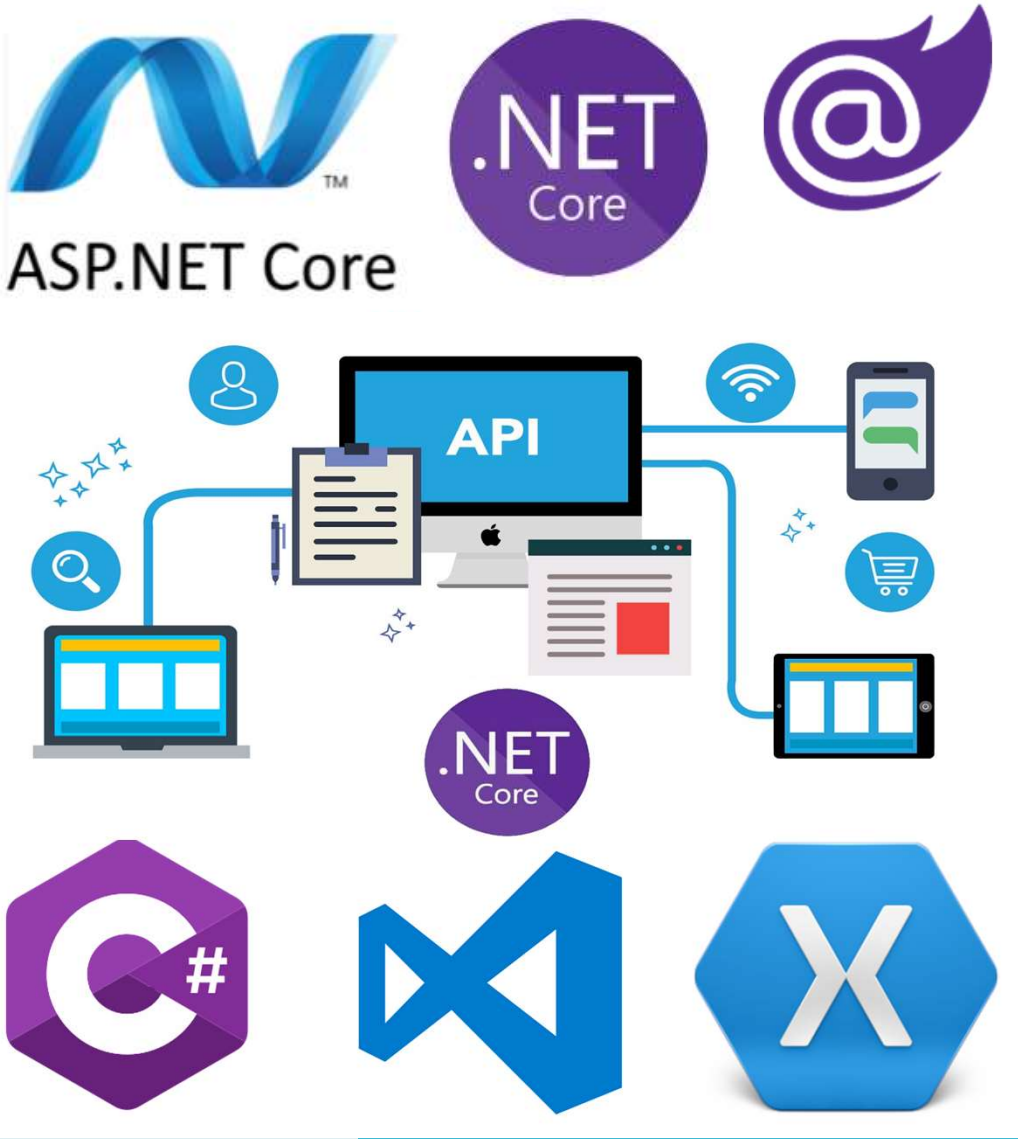
Curso: RESTful Web Services



Agenda

1 - Introducción al curso y herramientas

2 – Presentacion individual

3 – Presentacion del grupo

4 – Asp.NET Core presente y futuro

5 – Ejemplos

Sobre mí

Esteban Solano Granados

Senior Software Engineer

- Remote contractor / Instructor
- .Net / Mobile / Xamarin / Web Dev
- Mobile CR Developers

Twitter: @stvansolano

mobile cr developers

MVP Microsoft Most Valuable Professional

stvansolano@outlook.com

http://stvansolano.github.io/blog

Sobre mí


Microsoft
Microsoft Azure
Developer Camp
Microsoft
Microsoft Azure
Developer Camp
Microsoft
Microsoft Azure
Developer Camp
Microsoft Azure
Proyecte su jornada de modernización de aplicaciones con contenedores en Microsoft Azure
Esteban Solano Granados
@stvansolano
https://stvansolano.github.io/blog

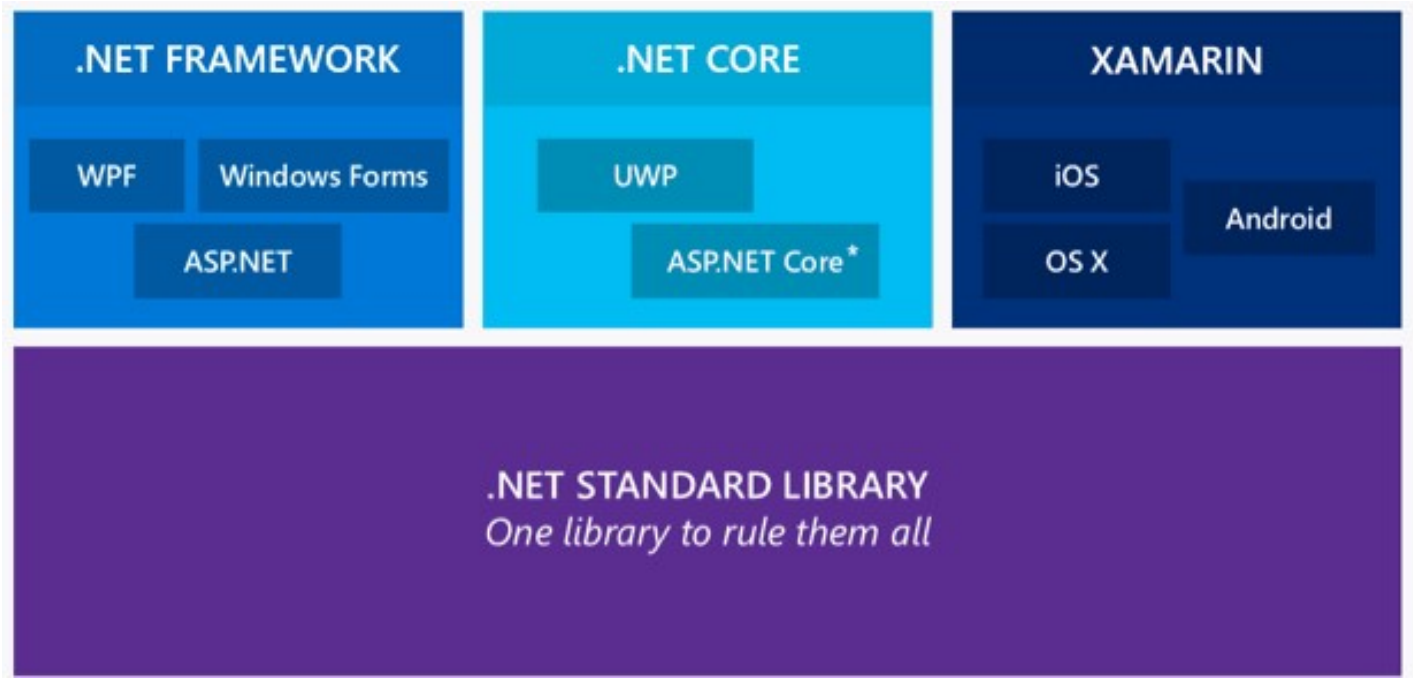
Recursos para el curso

Preparando el equipo

- VS Code / VS Studio 2017 / 2019
- Motores de Base de Datos (SQL / NoSQL)
- DotNet Core SDK 2.2
- Console Apps, Web Apps, Mobile Apps
- NodeJS

stvansolano@outlook.com http://stvansolano.github.io/blog

Presente - .NET Framework y .NET Core

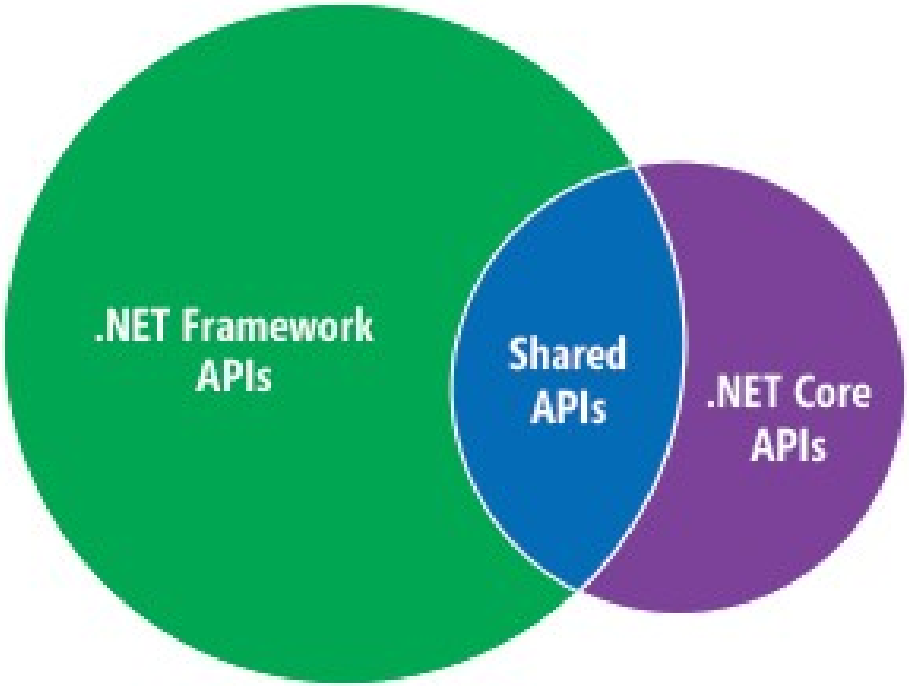


Twitter: @stvansolano

stvansolano@outlook.com

http://stvansolano.github.io/blog

.NET Presente y futuro - Código Compartido



Twitter: @stvansolano

stvansolano@outlook.com

http://stvansolano.github.io/blog

WebAPI

SignalR

MVC, Razor

Blazor (.NET Core 3)

Twitter: @stvansolano

stvansolano@outlook.com

http://stvansolano.github.io/blog

ASP.NET Core

>WebAPI

SignalR

MVC, Razor

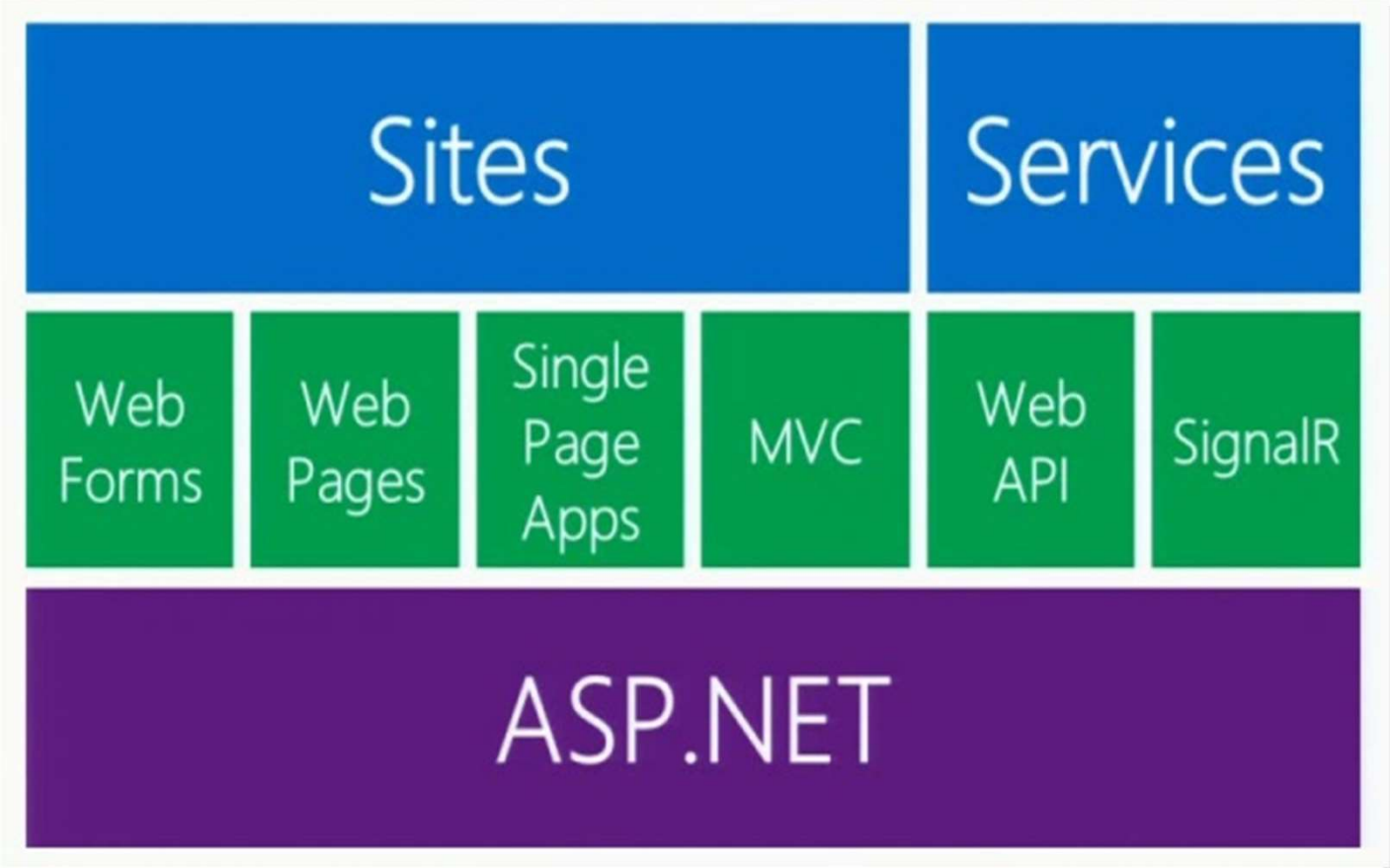
Blazor (.NET Core 3)

Twitter: @stvansolano

stvansolano@outlook.com

http://stvansolano.github.io/blog

.NET Core

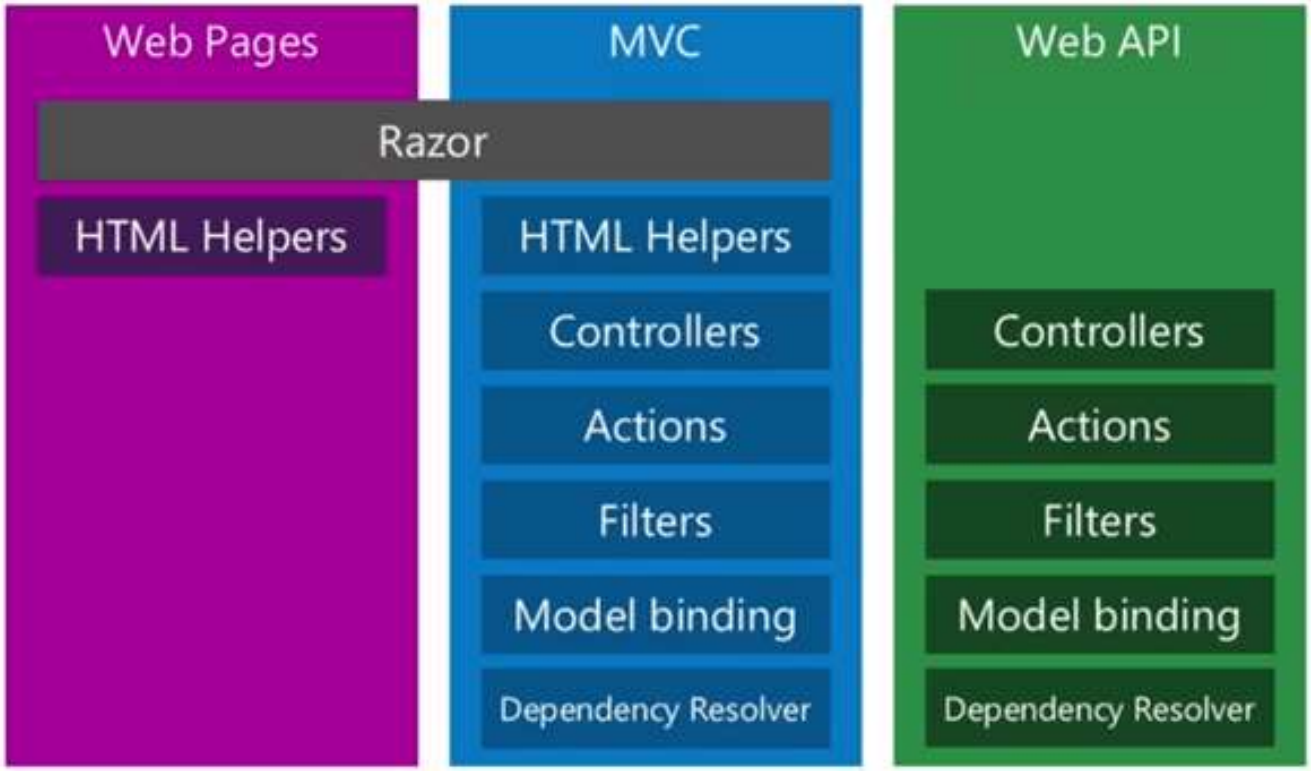


Twitter: @stvansolano

stvansolano@outlook.com

http://stvansolano.github.io/blog

Componentes



Twitter: @stvansolano

stvansolano@outlook.com

http://stvansolano.github.io/blog

.NET Core y futuro

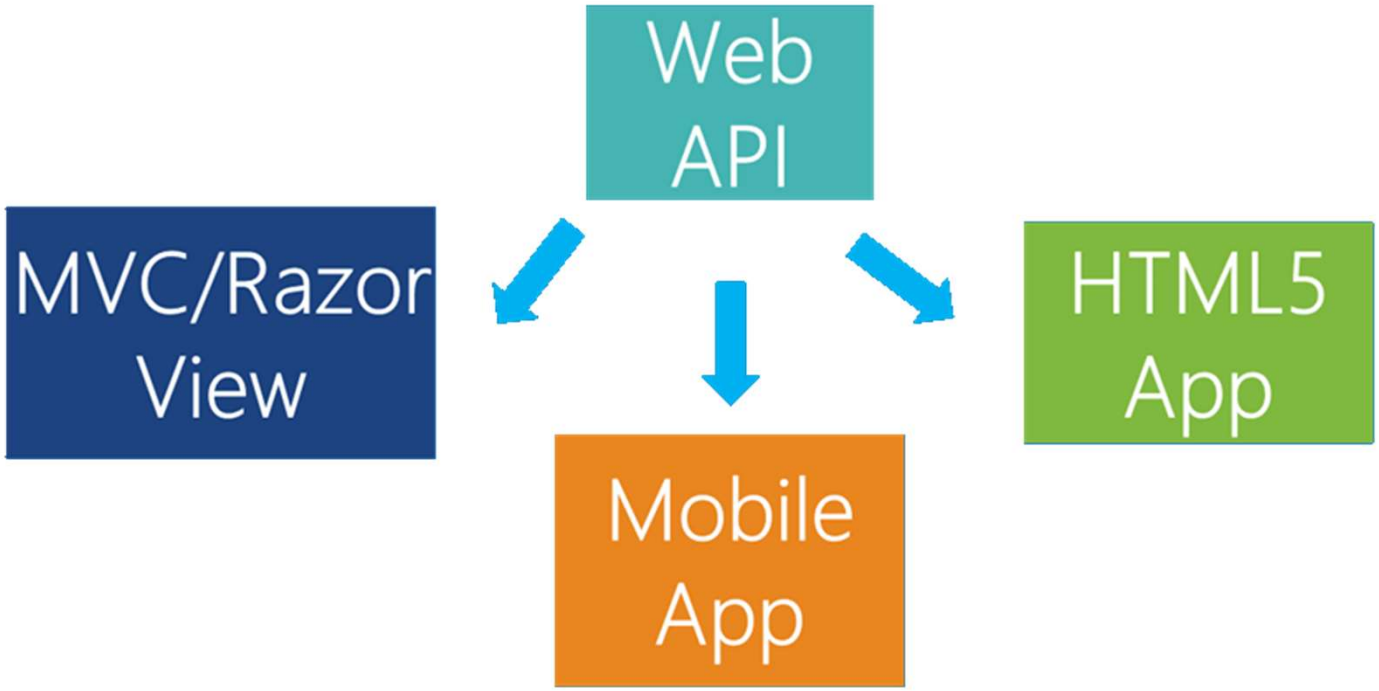
.NET – A unified platform

Twitter: @stvansolano

stvansolano@outlook.com

http://stvansolano.github.io/blog

Web API

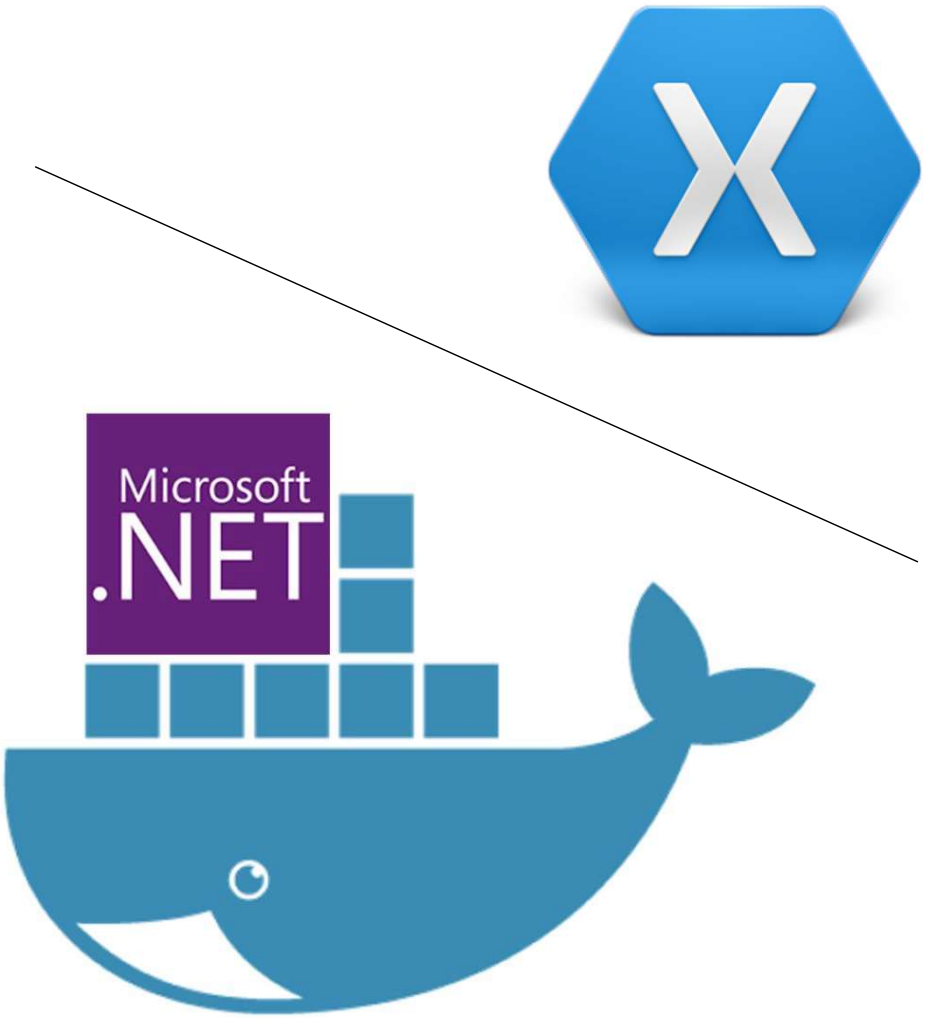


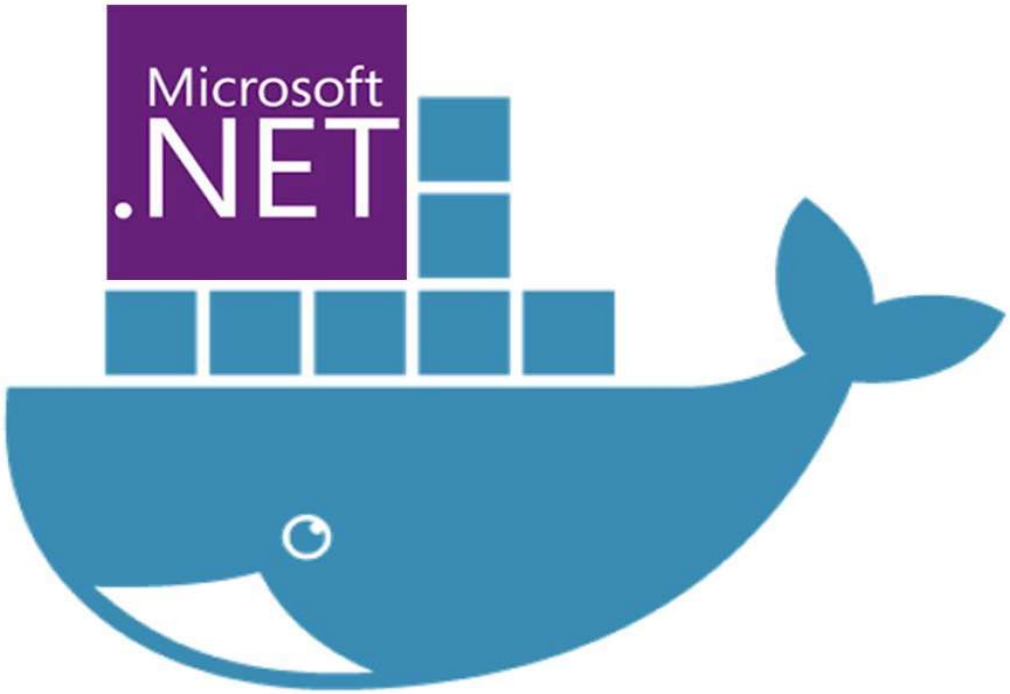
Twitter: @stvansolano

stvansolano@outlook.com

http://stvansolano.github.io/blog

Demo





Web API + Xamarin

- Swagger
- Docker
- Azure y contenedores (ACI)

Recursos Azure

- Azure Container Image
- Azure Container Instance
- Azure Container Registry

¿Preguntas?

Escríbeme

Twitter: @stvansolano

stvansolano@outlook.com

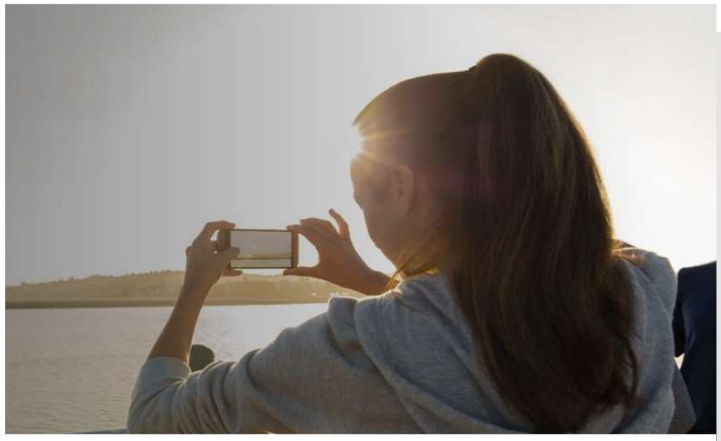
Recursos para el curso

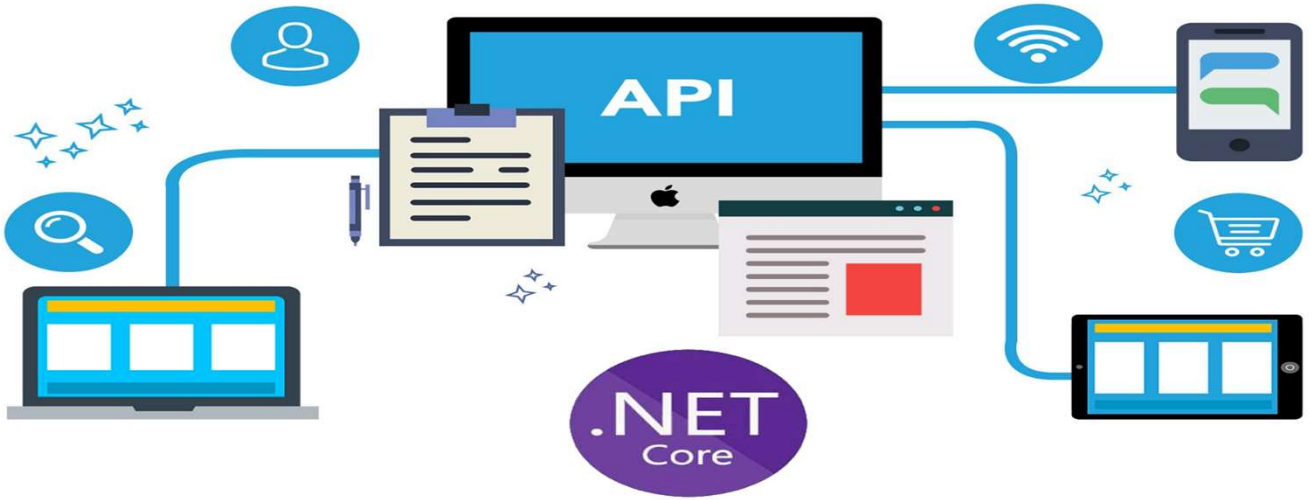
Manos a la obra

- VS Code / VS Studio 2017 / 2019
- Motores de Base de Datos (SQL / NoSQL)*
- DotNet Core SDK 2.2
- Console Apps, Web Apps, Mobile Apps
- NodeJS

stvansolano@outlook.com http://stvansolano.github.io/blog

RESTful Web Services





.NET Core y futuro

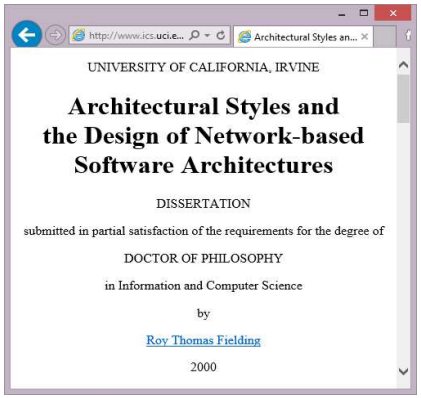
.NET – A unified platform

Twitter: @stvansolano

stvansolano@outlook.com

http://stvansolano.github.io/blog

REST


UNIVERSITY OF CALIFORNIA, IRVINE

Architectural Styles and the Design of Network-based Software Architectures

DISSERTATION

submitted in partial satisfaction of the requirements for the degree of

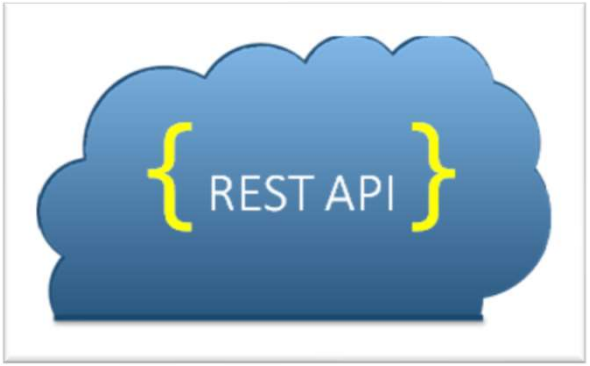
DOCTOR OF PHILOSOPHY

in Information and Computer Science

by

Roy Thomas Fielding

2000



REST

Representational State Transfer

Arquitectura para crear aplicaciones distribuidas y modeladas alrededor de la especificación HTTP

REST Dissertation

Tesis de Roy Fielding ~2000

Cliente <-> Servidor

- Cache
- Stateless
- Uniform Interface
- Code-On-Demand
- Connectors
- Proxy

https://www.ics.uci.edu/~fielding/pubs/dissertation/fielding_dissertation.pdf

REST 101

REST está diseñado para tomar ventaja de la arquitectura de la WWW

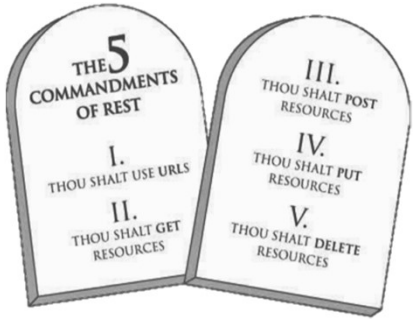
- Operaciones implementadas por verbos HTTP
- Utiliza URLs que representan recursos accesibles

C = Create=> POST

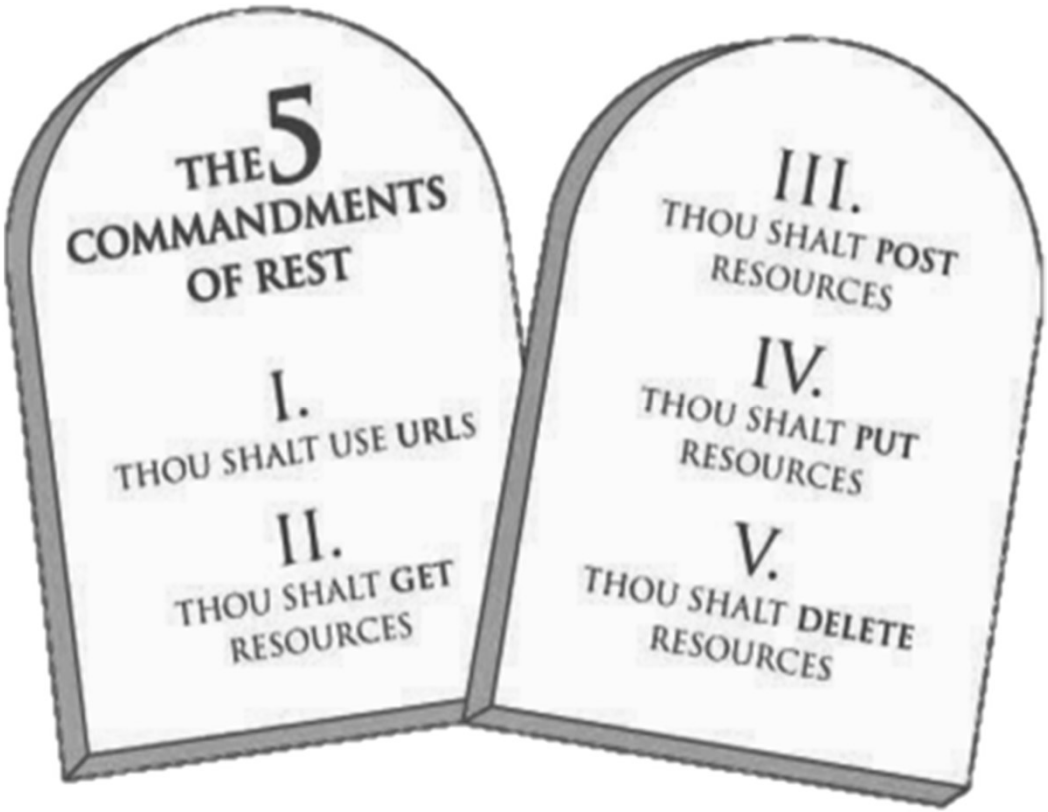
R = Read=> GET

U = UPdaTe => PUT

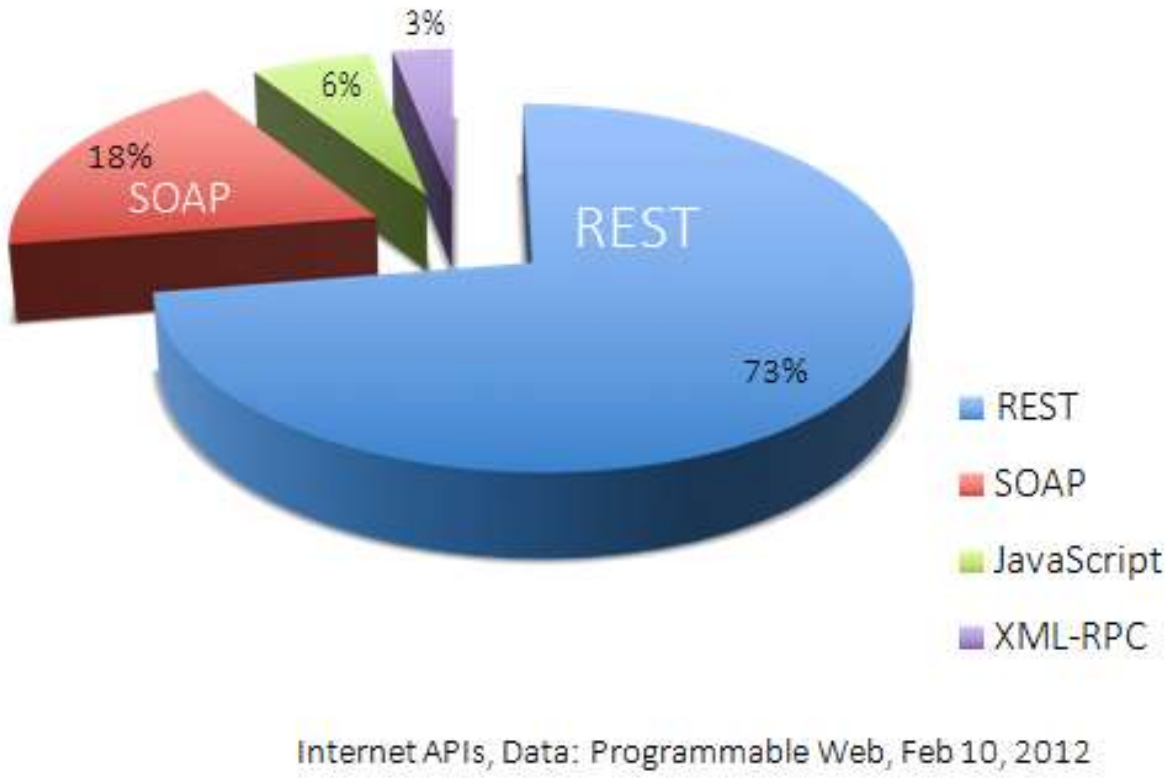
D = Delete=> DELETE



REST 101


THE 5 COMMANDMENTS OF REST
I.
THOU SHALT USE URLS
II.
THOU SHALT GET RESOURCES
III.
THOU SHALT POST RESOURCES
IV.
THOU SHALT PUT RESOURCES
V.
THOU SHALT DELETE RESOURCES

¿Por qué utilizar REST? (II)



Internet APIs, Data: Programmable Web, Feb 10, 2012

Antes - SOAP


Response 1
XML
Raw
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/" x
<soapenv:Header/>
<soapenv:Body>
<v1:GetObjectListResponse>
<v1:GetObjectListResult success="true" errorMsg="">
<v1:Object id="5">
<v1:StringVal name="Surname">Kowalski</v1:StringVal>
</v1:Object>
</v1:GetObjectListResult>
</v1:GetObjectListResponse>
</soapenv:Body>
</soapenv:Envelope>

REST = URLs + Operaciones


```
GET https://www.some_address.com/customers/12345

GET https://www.some_address.com/customers?id=12345
```

```
HTTP/1.1 200 OK
Content-Type: text/xml; charset=utf-8
Content-Length: ####
...
```

```
"category-0":{
   "Articles":[
      {
         "Text":"Aqui va el texto",
         "Title":"Asp Net 5"
      },
      {
         "Text":"Aqui va el texto",
         "Title":"Web API"
      }
   ],
   "Title":"Asp.Net 5"
},
```

{JSON}

- JavaScript Object Notation es un formato de serialización muy conocido que utiliza pares de nombre/valor en forma jerárquica

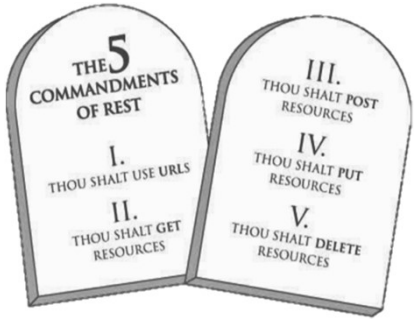
¿Preguntas?

Escríbeme

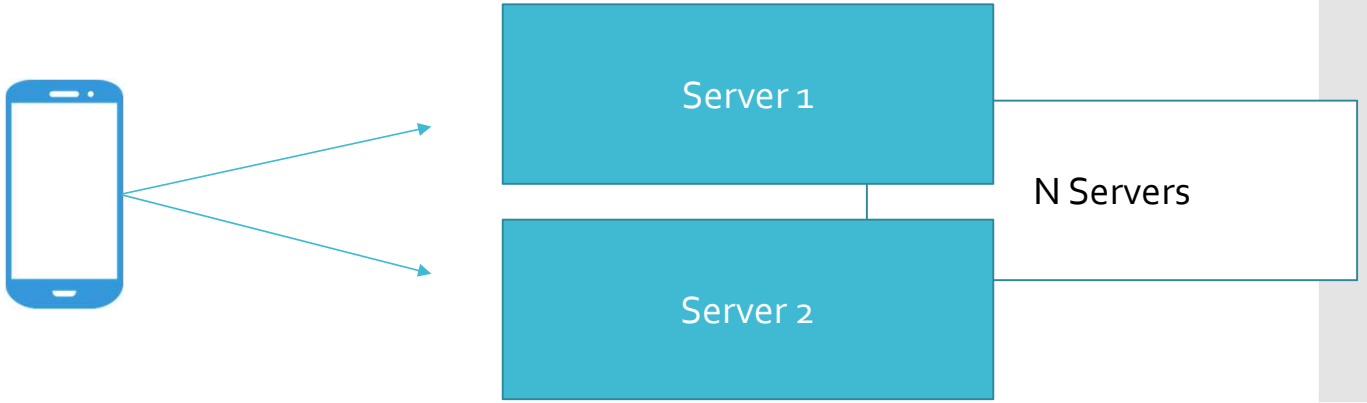
Twitter: @stvansolano

stvansolano@outlook.com

REST 101



- Uniform
- Stateless
- **HATEOAS - ??**



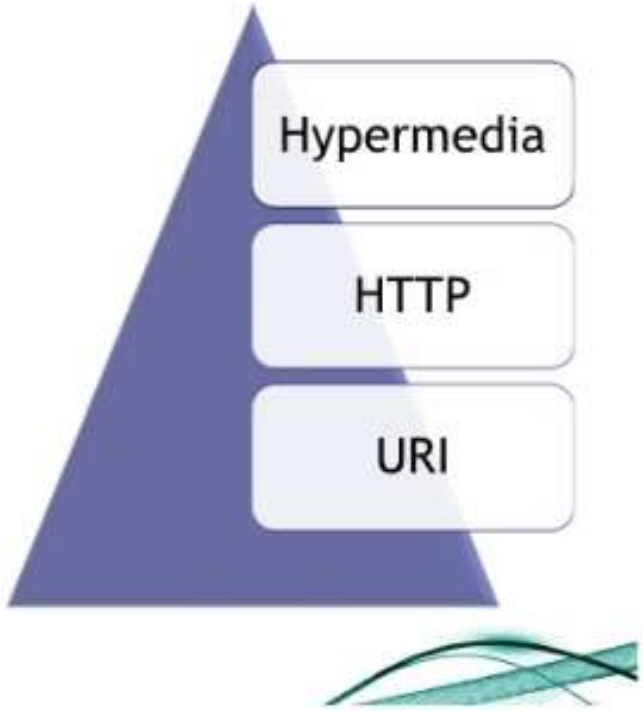
HATEOAS

- **HATEOAS=Hypermedia As the Engine of the Application State**

RESTful Maturity levels

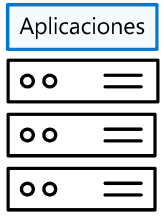
The Richardson Maturity Model

- Level 0
  - SOAP, XML RPC, POX
  - Single URI
- Level 1
  - URI Tunnelling
  - Many URIs, Single verb
- Level 2
  - Many URIs, many verbs
  - CRUD services (e.g. Amazon S3)
- Level 3
  - Level 2 + Hypermedia
  - RESTful Services



Contenedores Docker

¿Qué es un contenedor?



Máquinas virtuales

Virtualizar el hardware

Máquinas virtuales como unidades de escalado

Contenedores

Virtualizar el sistema operativo

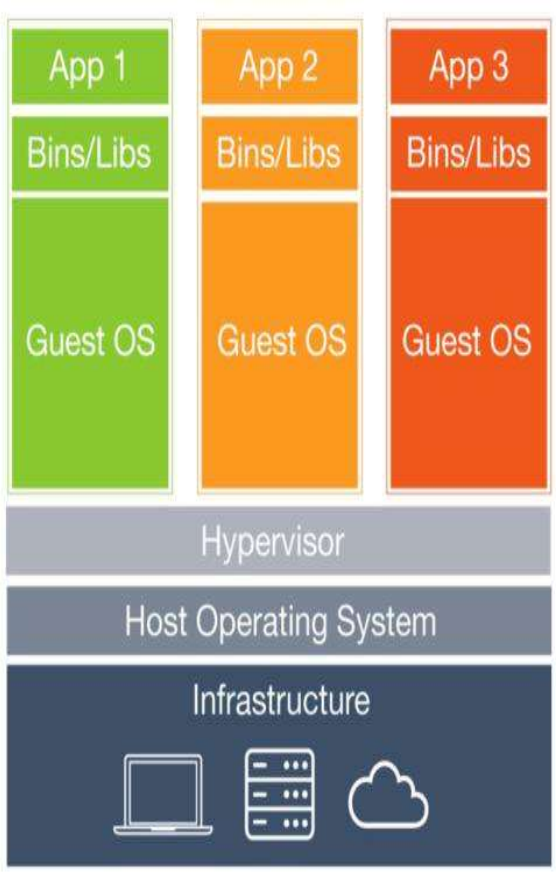
Aplicaciones como unidades de escalado

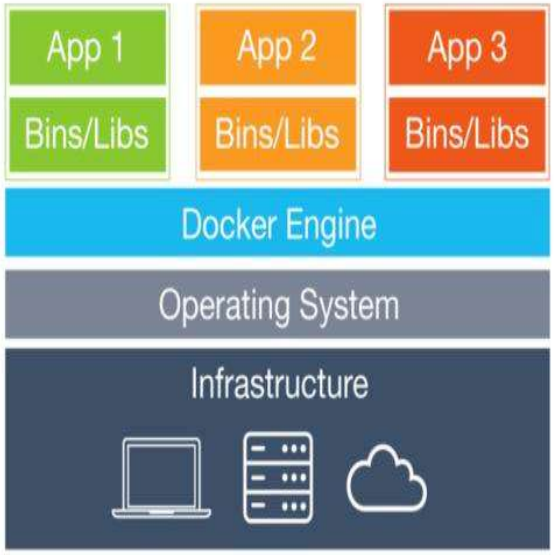
Twitter: @stvansolano

stvansolano@outlook.com

http://stvansolano.github.io/blog

VMs versus Containers



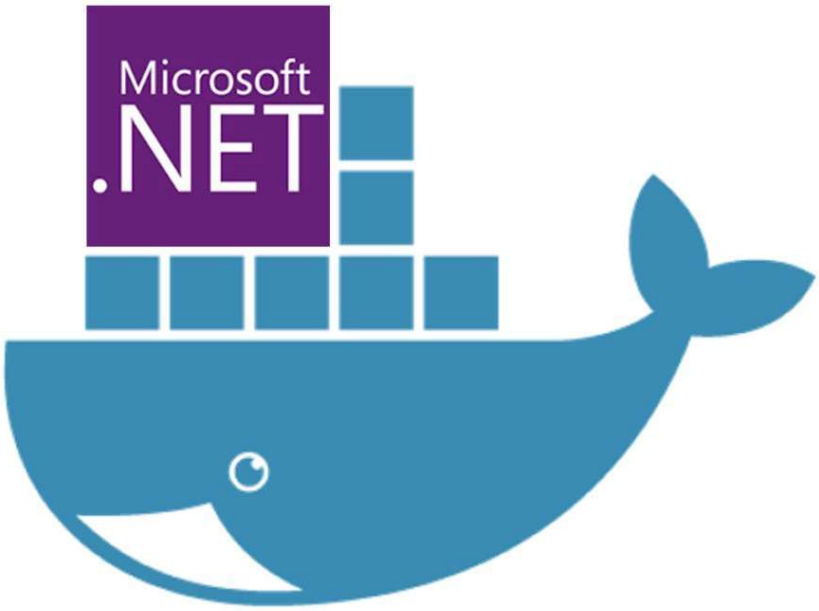


Twitter: @stvansolano

stvansolano@outlook.com

http://stvansolano.github.io/blog

Demos



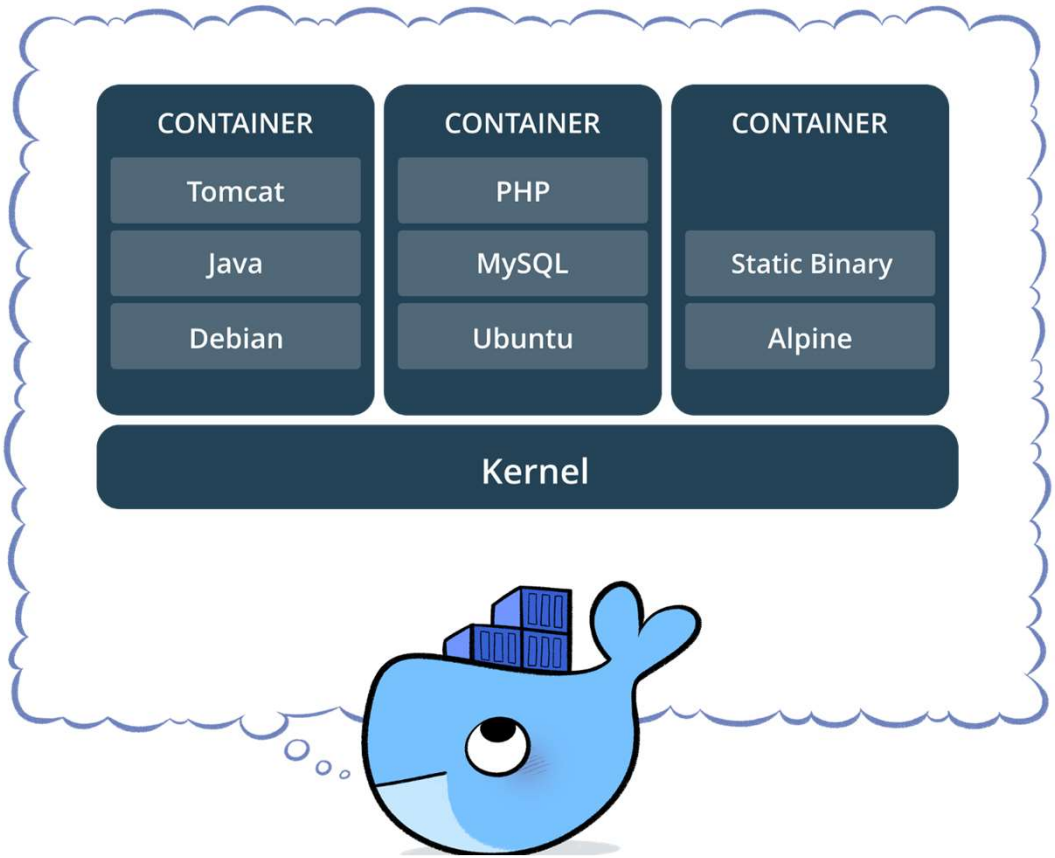
¿Preguntas?

Escríbeme

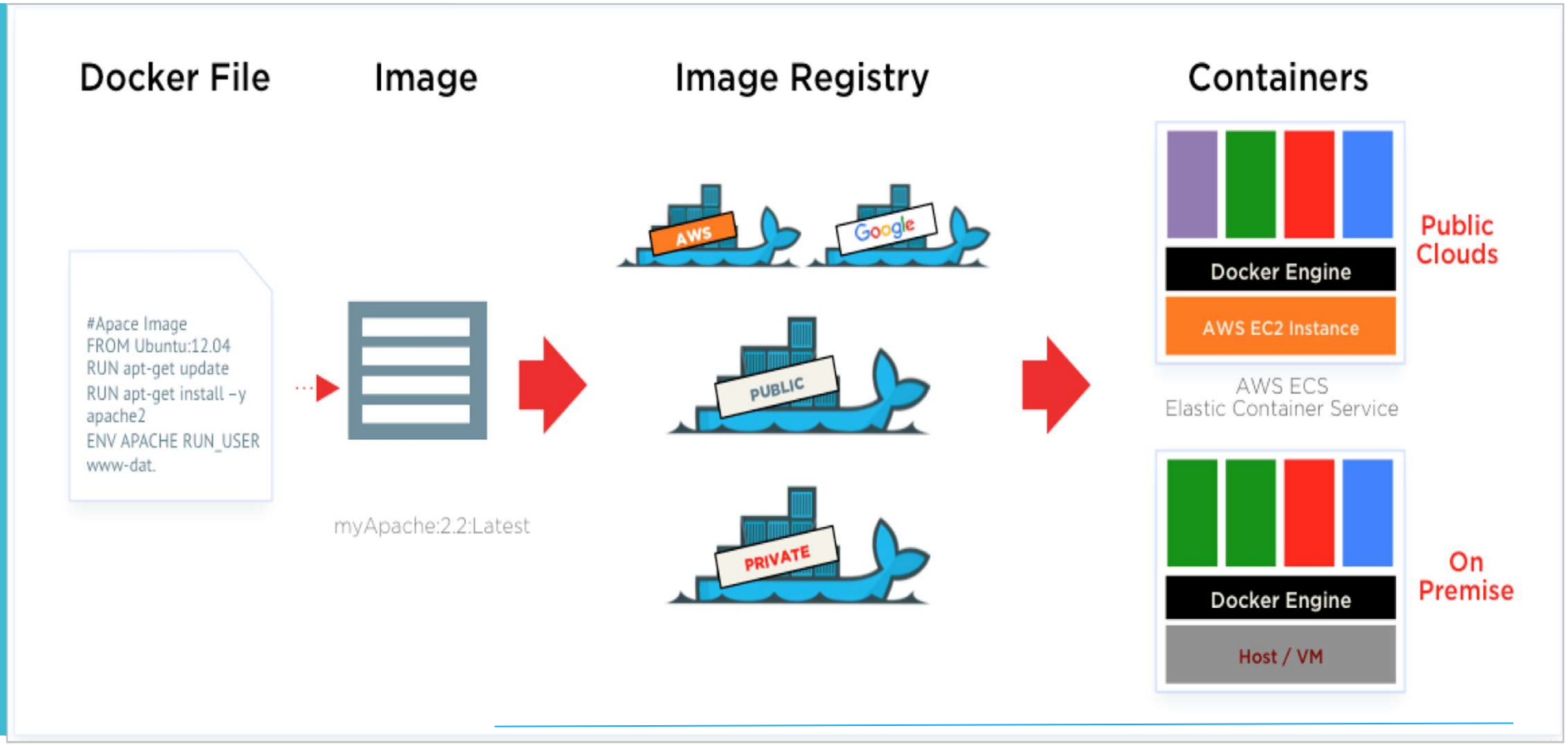
Twitter: @stvansolano

stvansolano@outlook.com

Docker 101 (Basico)



Docker workflows



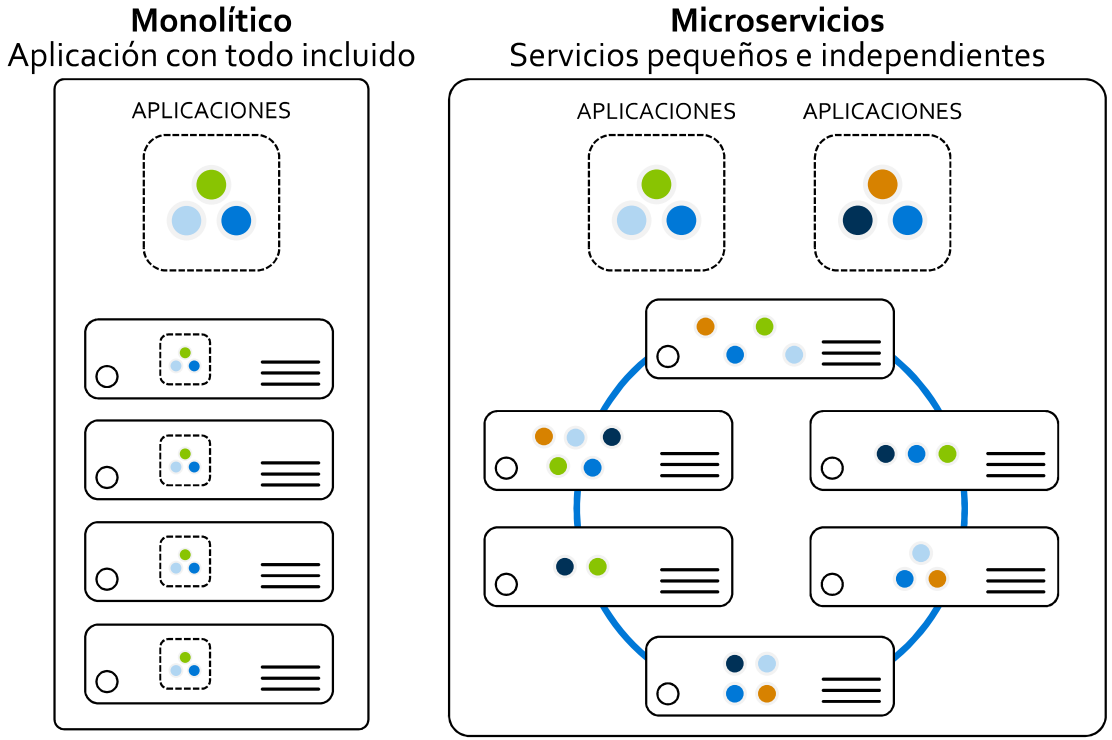
Twitter: @stvansolano

stvansolano@outlook.com

http://stvansolano.github.io/blog

Microservicios

¿Qué son los microservicios?



Microservicios

¿Qué son los microservicios?

Un estilo arquitectónico de software

Las aplicaciones se componen de pequeños módulos independientes que se comunican entre sí mediante API bien definidas. No específico de la plataforma

Desacoplados

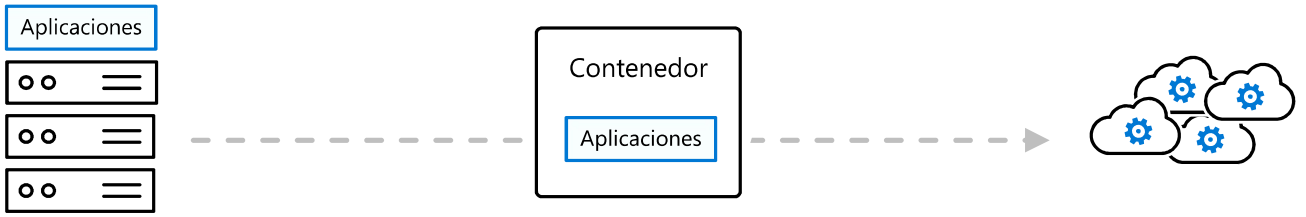
Estos módulos de servicio son bloques de construcción altamente desacoplados que son lo suficientemente pequeños para implementar una funcionalidad única pero que juntos pueden formar sistemas más grandes

Independientemente versionado, implementado y escalado

Con una arquitectura de microservicios, los desarrolladores pueden crear, gestionar y mejorar los servicios de aplicaciones de forma independiente, incluso utilizando diferentes idiomas.

Los contenedores proporcionan el formato consistente y el aislamiento deseado por los microservicios

El Viaje hacia la nube



Aplicaciones existentes/nuevas

Elevación y cambio a contenedores

Cada host suele ejecutar **8** contenedores a la vez[1]

Microservicios

Segundos para implementar, opera **12 veces** más rápido[1]

Innovación continua con DevOps

46 veces más frecuente en la entrega de código[2]

1: Informe Datalog: 8 hechos sorprendentes sobre la adopción de Docker; 2: Informe sobre el estado de los dispositivos en 2017

Best Practices

- Usar variables de entorno para ambientes Docker-izados / Compose / Kubernetes
- Content-negotiation: Usarlo / habilitarlo para escenarios específicos (No sólo JSON)
- HATEOAS: Linking y APIs relacionales (.Include & Anonymous types / VMs)

Best Practices (II)

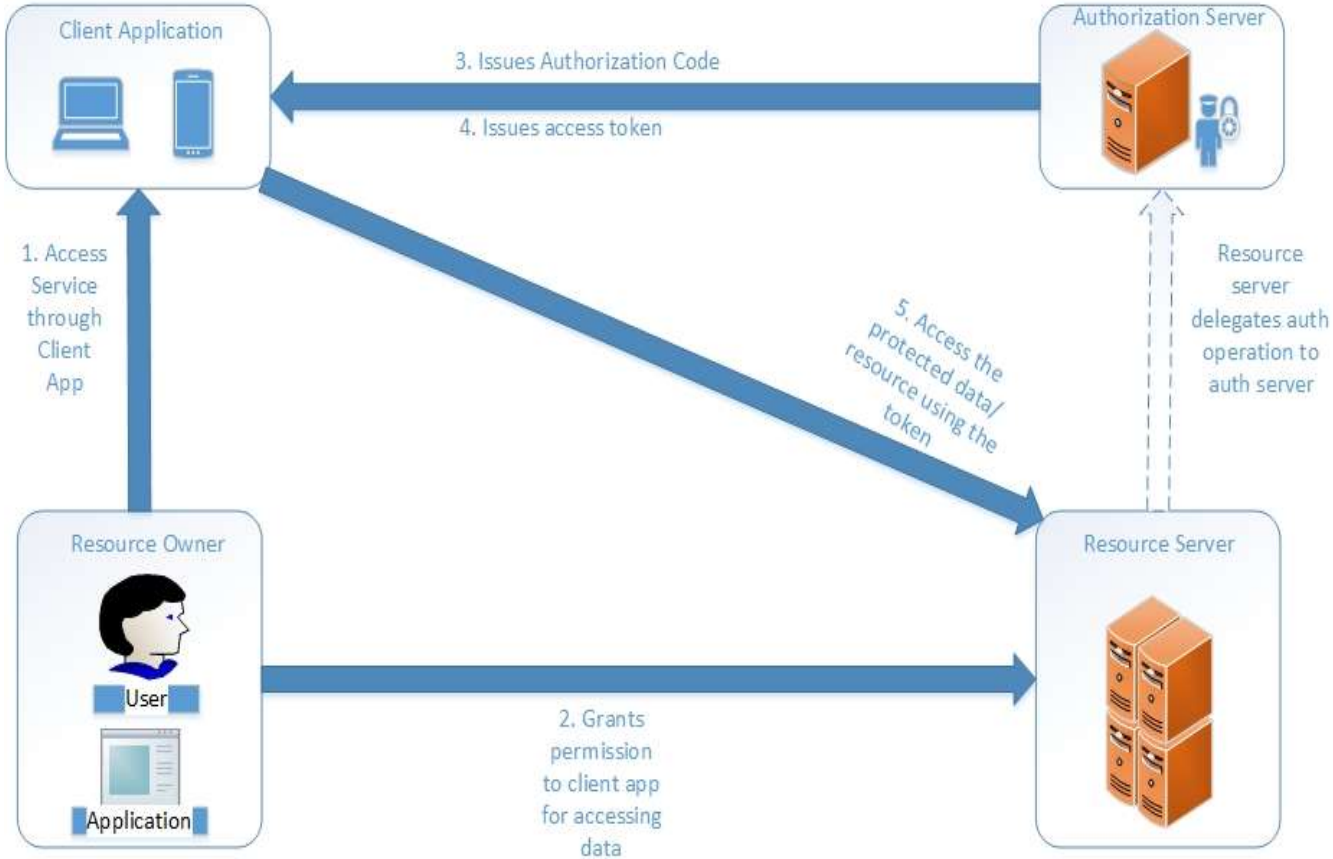
- Uso de Middleware para errores generales
- OData Protocol: Queries, filtering, pagination
- OAuth (flows) + Authorization

OAuth

- Protocol for Authorization
- **Authorization != Authentication**

```
[ApiController]
[Route ("/api/[Controller]")]
[Authorize("MyPolicy")]
public class ProductsController : Controller {
```

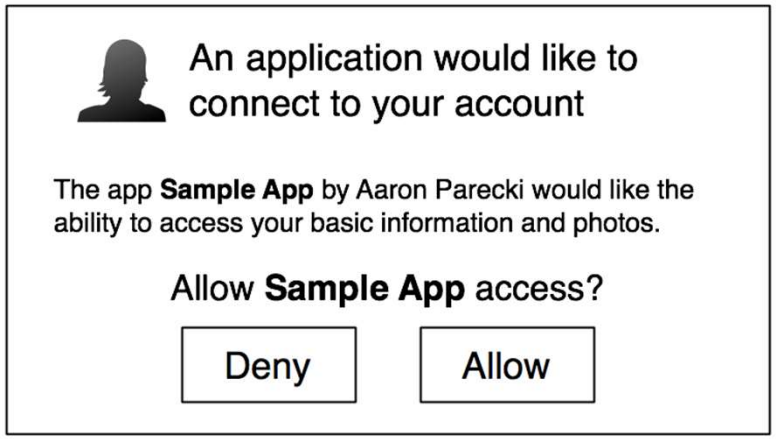
OAuth


Client Application
Authorization Server
3. Issues Authorization Code
4. Issues access token
1. Access Service through Client App
5. Access the protected data/ resource using the token
Resource server delegates auth operation to auth server
Resource Owner
User
Application
Resource Server
2. Grants permission to client app for accessing data

OAuth

OAuth Flows or "grant types"

- **Bearer Token**
- Client credentials (apps)
- Others (code)



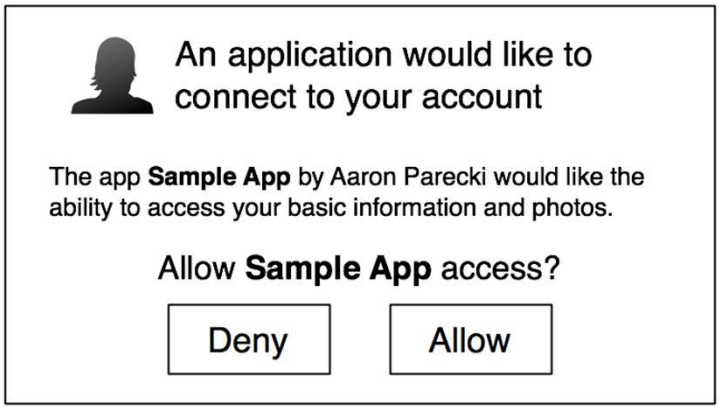
OAuth - Bearer Token

ASP.NET Core

- **Bearer Token**
- AuthController
- Claims
- Middleware
- Bearer token

OAuth Client Credentials

- Redirect URL
- **client_id / secret**
- Scope
- State




```
POST https://api.authorization-server.com/token
  grant_type=authorization_code&
  code=AUTH_CODE_HERE&
  redirect_uri=REDIRECT_URI&
  client_id=CLIENT_ID
```

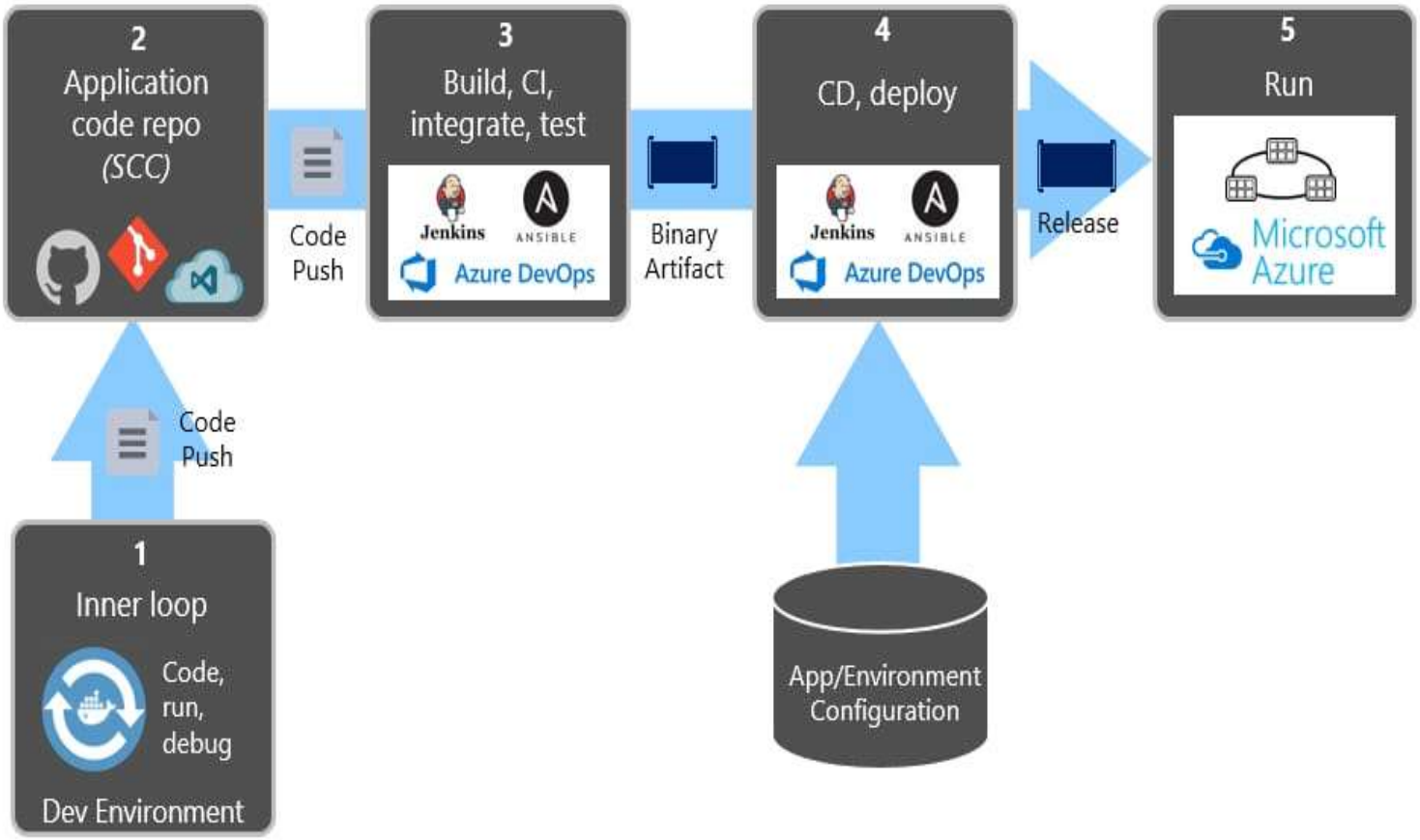
¿Preguntas?

Escríbeme

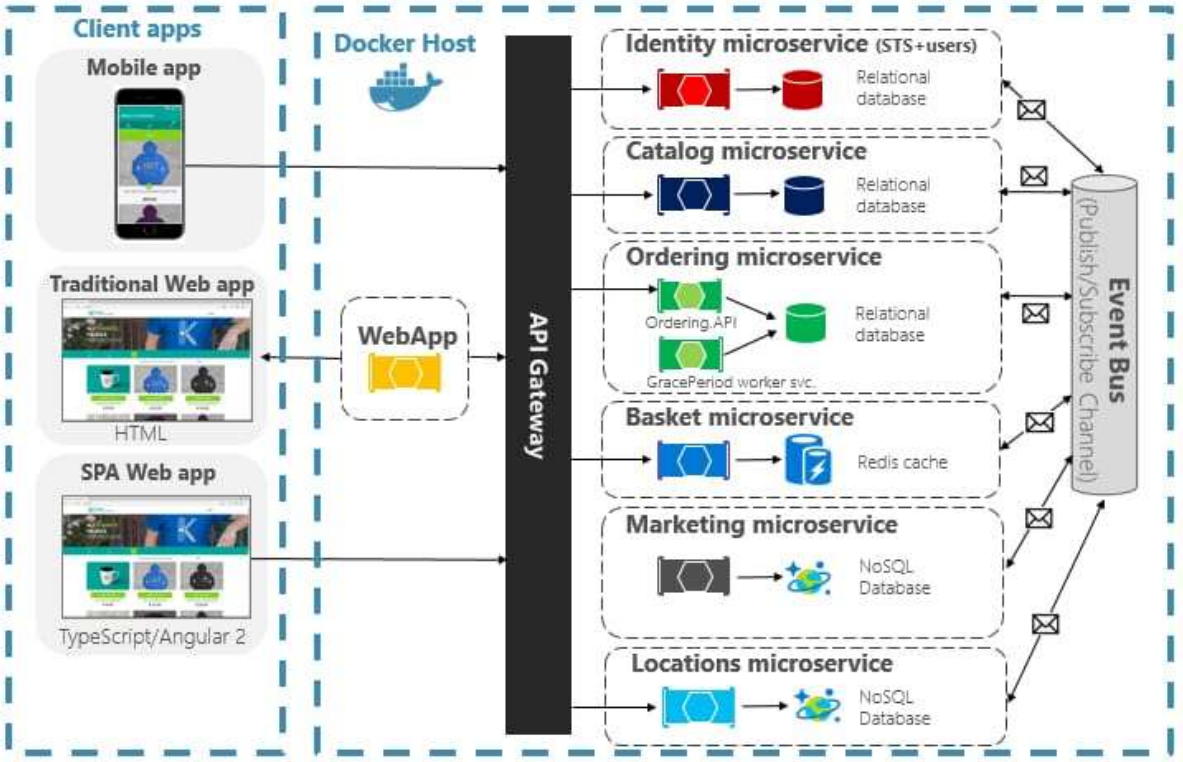
Twitter: @stvansolano

stvansolano@outlook.com

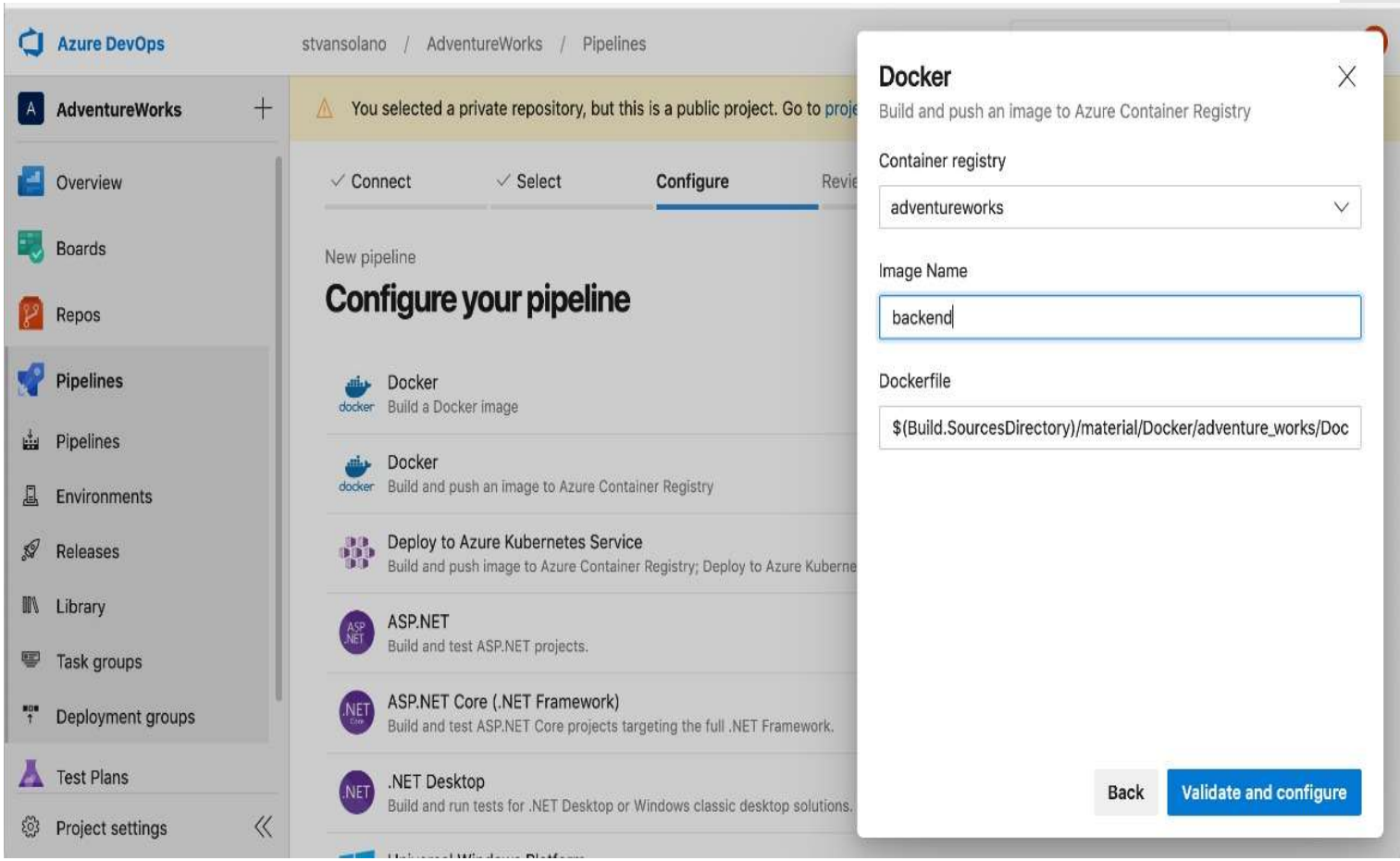
CI / CD Basics


2
Application code repo (SCC)
Code Push
3
Build, CI, integrate, test
Jenkins
ANSIBLE
Azure DevOps
Binary Artifact
4
CD, deploy
Jenkins
ANSIBLE
Azure DevOps
Release
5
Run
Microsoft Azure
Code Push
1
Inner loop
Code, run, debug
Dev Environment
App/Environment Configuration

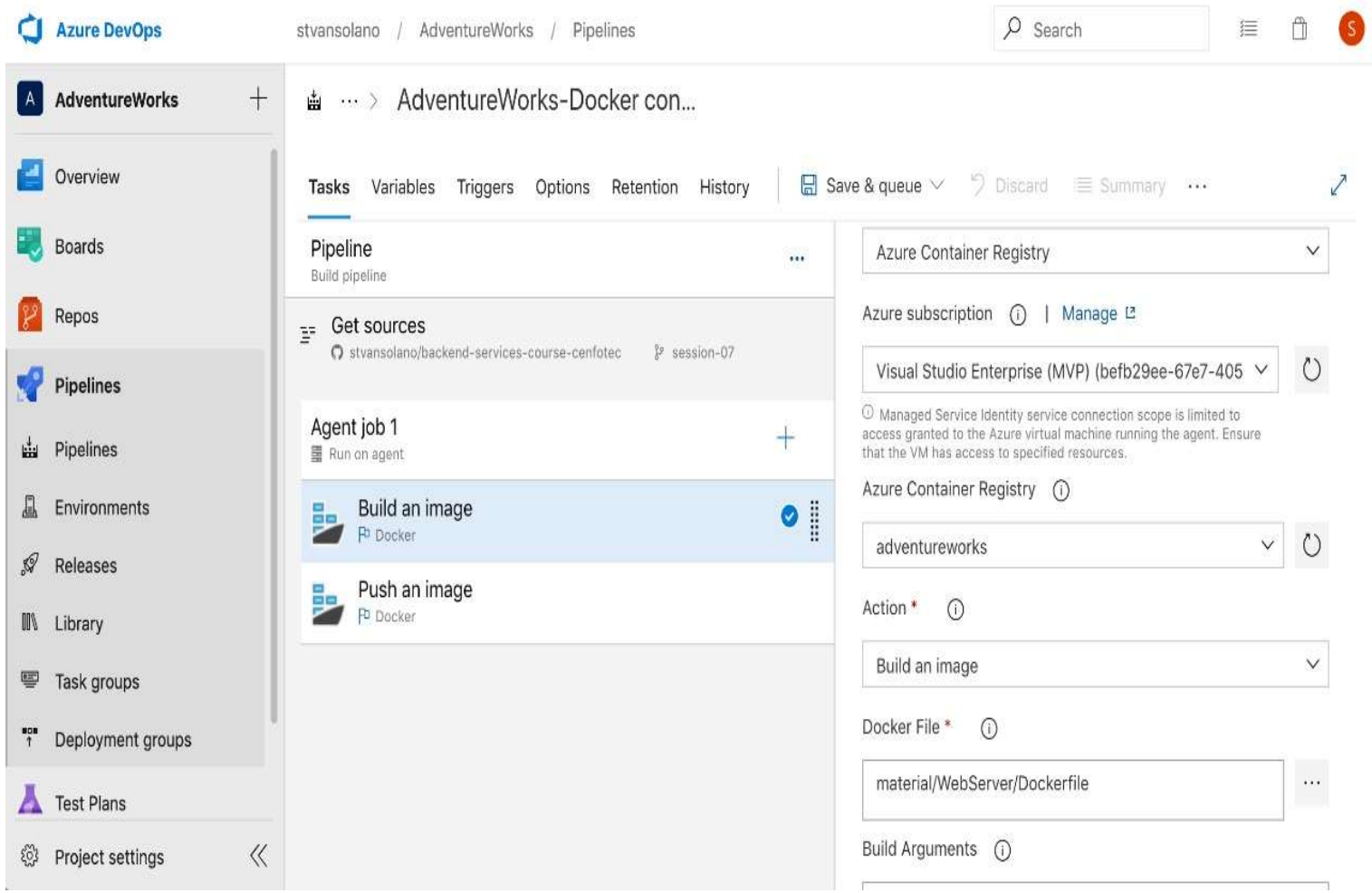
CI / CD + Microservices



Pipelines


Azure DevOps
stvansolano / AdventureWorks / Pipelines
AdventureWorks
Overview
Boards
Repos
Pipelines
Pipelines
Environments
Releases
Library
Task groups
Deployment groups
Test Plans
Project settings
You selected a private repository, but this is a public project. Go to proje
✓ Connect
✓ Select
Configure
New pipeline
Configure your pipeline
Docker
Build a Docker image
Docker
Build and push an image to Azure Container Registry
Deploy to Azure Kubernetes Service
Build and push image to Azure Container Registry; Deploy to Azure Kuberne
ASP.NET
Build and test ASP.NET projects.
ASP.NET Core (.NET Framework)
Build and test ASP.NET Core projects targeting the full .NET Framework.
.NET Desktop
Build and run tests for .NET Desktop or Windows classic desktop solutions.
Docker
Build and push an image to Azure Container Registry
Container registry
adventureworks
Image Name
backend
Dockerfile
$(Build.SourcesDirectory)/material/Docker/adventure_works/Doc
Back
Validate and configure

Docker + CI / CD


Azure DevOps
stvansolano / AdventureWorks / Pipelines
Search
AdventureWorks
Overview
Boards
Repos
Pipelines
Pipelines
Environments
Releases
Library
Task groups
Deployment groups
Test Plans
Project settings
AdventureWorks-Docker con...
Tasks
Variables
Triggers
Options
Retention
History
Save & queue
Discard
Summary
Pipeline
Build pipeline
Get sources
stvansolano/backend-services-course-cenfotec
session-07
Agent job 1
Run on agent
Build an image
Docker
Push an image
Docker
Azure Container Registry
Azure subscription
Manage
Visual Studio Enterprise (MVP) (befb29ee-67e7-405
Managed Service Identity service connection scope is limited to access granted to the Azure virtual machine running the agent. Ensure that the VM has access to specified resources.
Azure Container Registry
adventureworks
Action *
Build an image
Docker File *
material/WebServer/Dockerfile
Build Arguments

¿Preguntas?

Escríbeme

Twitter: @stvansolano

stvansolano@outlook.com

¡Gracias!

Escríbeme

Twitter: @stvansolano

stvansolano@outlook.com