

LOS IMPRESCINDIBLES

**AZURE
SERVICES**





Azure es todavía un universo por explorar para muchos de nosotros. Sus posibilidades, alcance e infraestructura, están en constante expansión y liderar esta tecnología no es nada fácil...¿o sí?

En esta serie de artículos sobre los "Azure Services" vamos a dejarnos llevar por la experiencia de **Alberto Díaz, Luise López, Carlos López y Felipe López**, para descubrir las mejores prácticas sobre el desarrollo en la nube de Microsoft e ir conociendo sus productos y servicios. Y así, poniendo al alcance de todos la nube de Azure, conseguiremos que su éxito llegue a cada empresa y organización ;)

Happy Coddling! :)

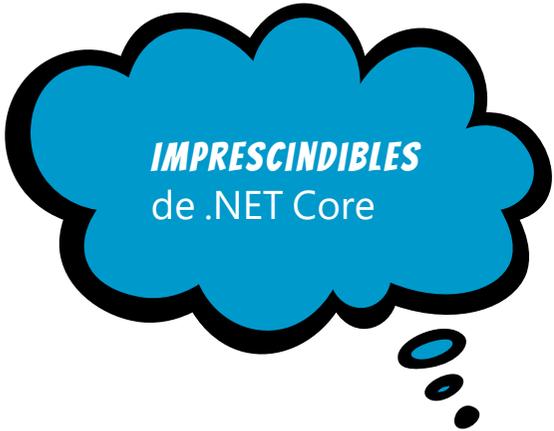


Índice de
contenidos



IMPRESINDIBLES
de Azure Services

- > Containerless, una versión Serverless de Contenedores en Azure
- > Cómo activar HTTP/2 en un App Service de Azure
- > Serverless. Sin servidores y no morir en el intento
- > Azure Container Registry: cómo configurarlo
- > Azure Resource Manager
- > Orquestando proyectos: VST, Slack, SonarQube



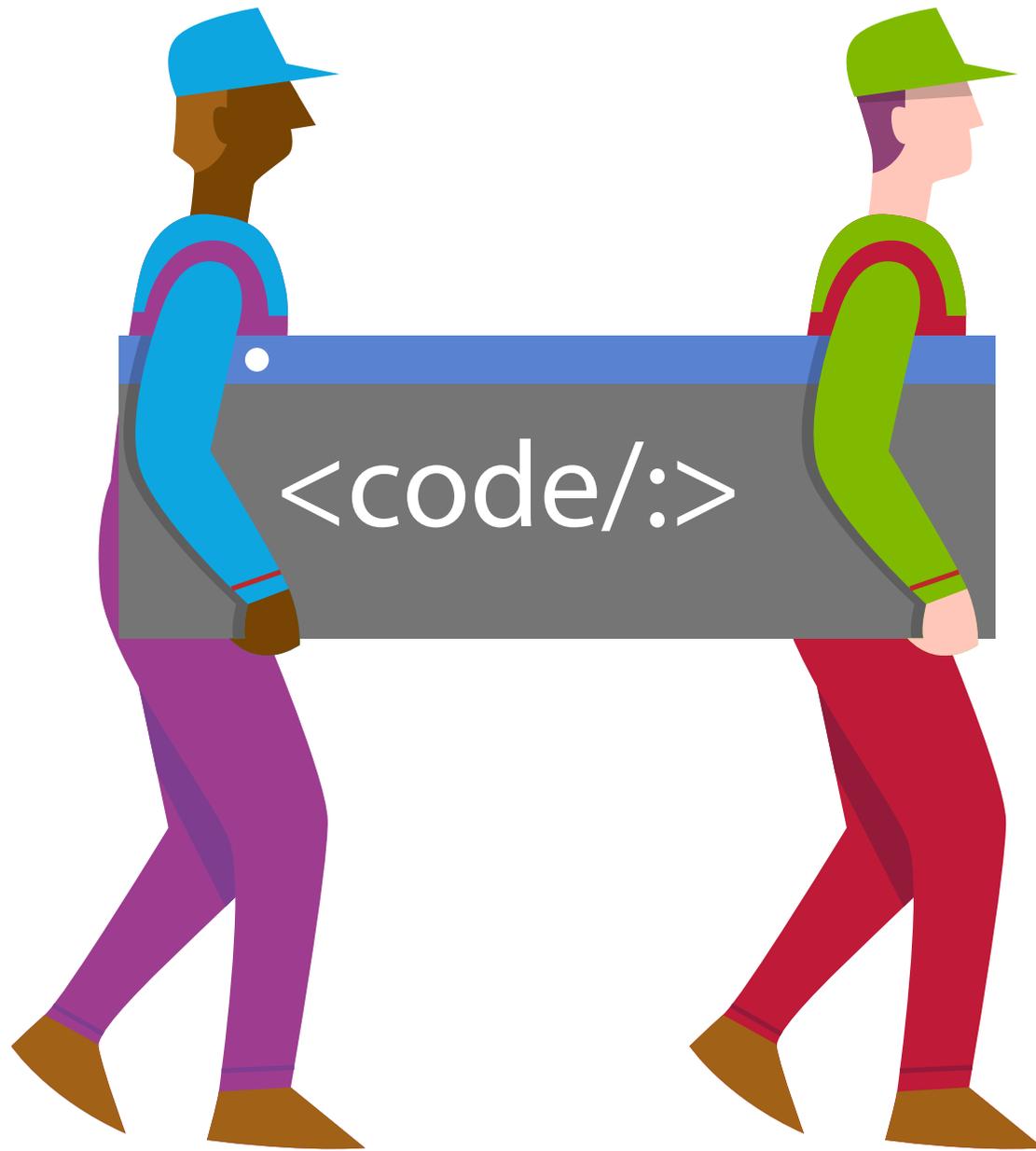
IMPRESINDIBLES
de .NET Core

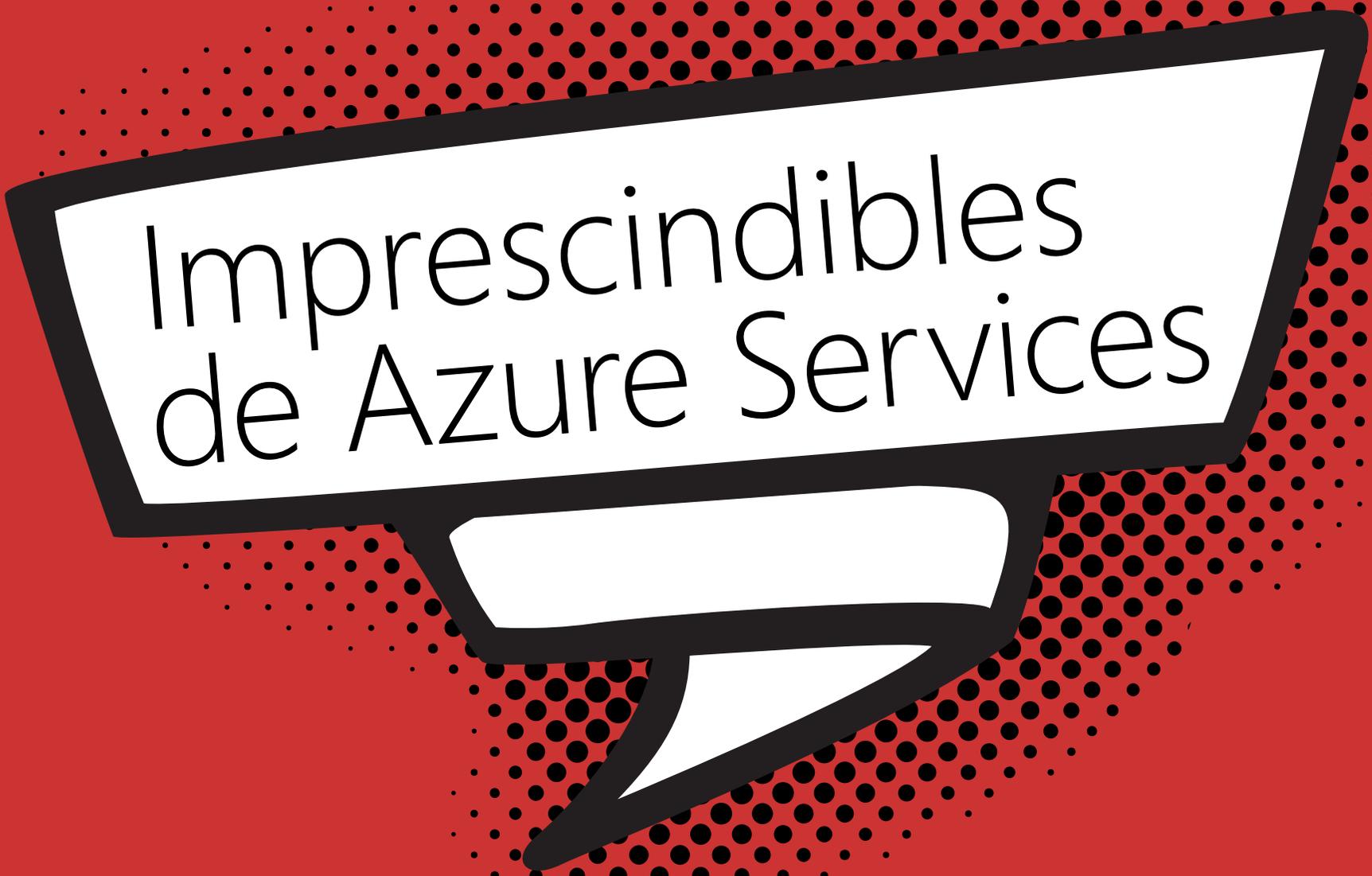


IMPRESINDIBLES
de SharePoint



IMPRESINDIBLES
de Seguridad en
Azure y Office 365

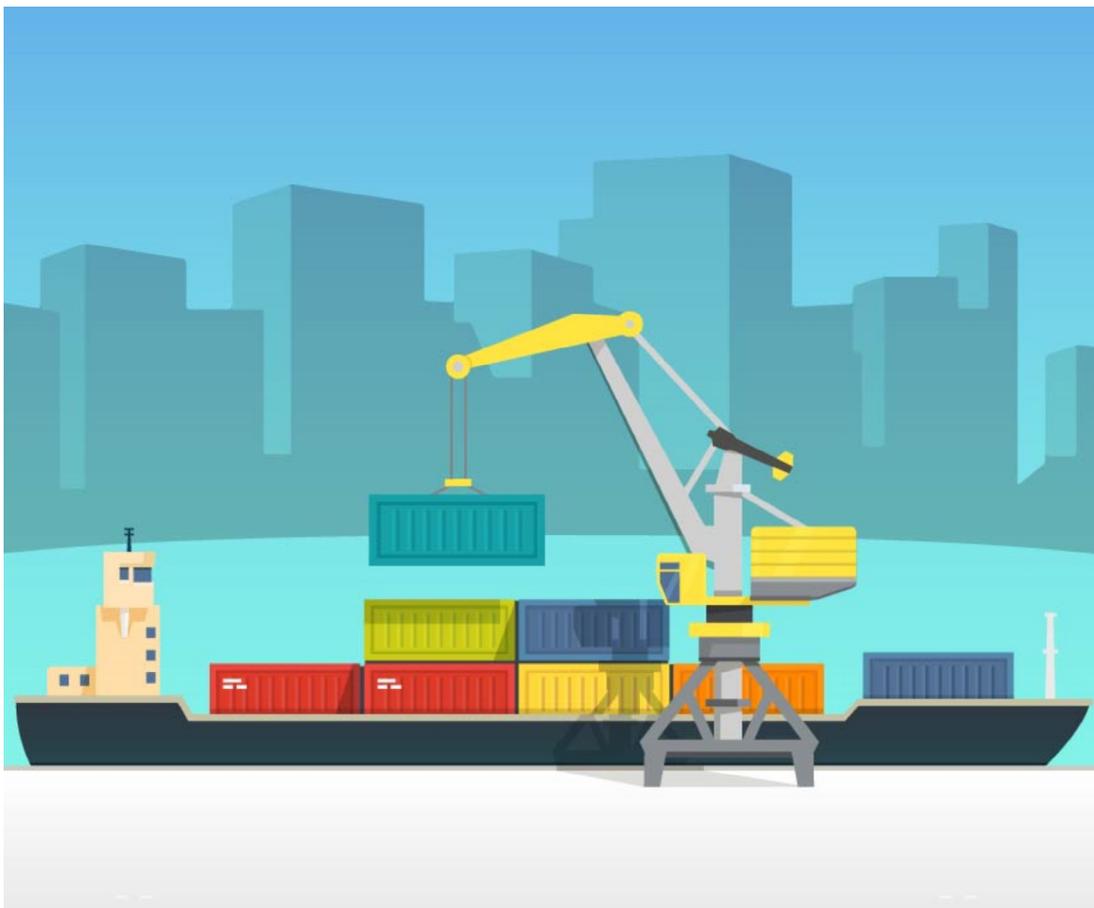




Imprescindibles
de Azure Services



Containerless, una versión Serverless de Contenedores en Azure



Azure [Container Instances](#) es la versión Serverless para ejecutar contenedores en Azure, lo que nos simplifica la administración o el provisionado de servidores que conformen un cluster. Simplemente instanciamos un contenedor y se ejecuta, sin saber dónde ni cómo.

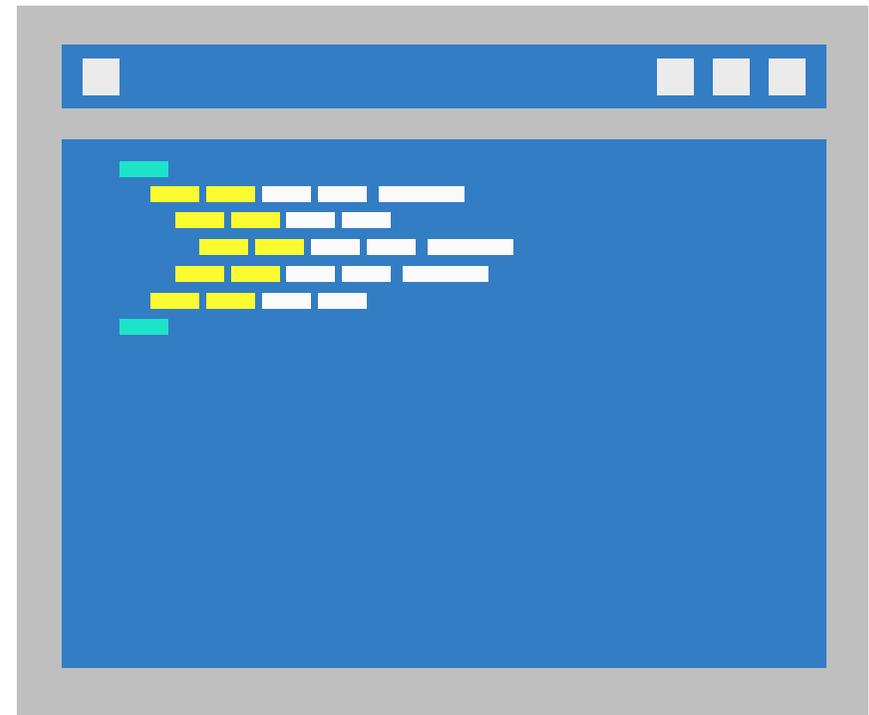
Azure nos ofrece diversos servicios con arquitectura Serverless, por ejemplo, **Azure Functions**, que es un servicio en el cual ejecutamos nuestro código ante una petición y, salvo que cambiemos el modo de ejecución a no-serverless (App Service Plan), nos ofrece como máximo 10 minutos para la ejecución del proceso.

¿Cómo funciona Azure Function?

En realidad, **Azure Functions** empaqueta nuestro código (C#, nodejs, ...) en un contenedor y cuando se desencadena la ejecución, instancia el contenedor en alguno de los nodos disponibles para que se ejecute con la variables de contextos adecuadas y los datos necesarios.



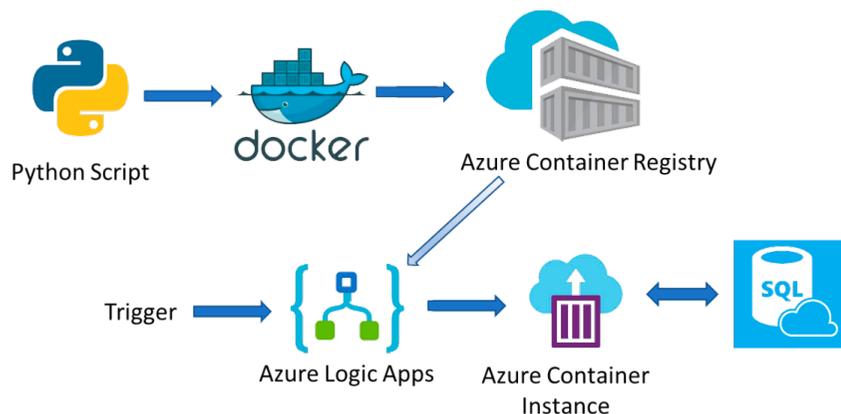
En función del número de desencadenadores, **Azure Functions** instanciará tantos contenedores como sean necesarios y, si los estresamos un poco, podemos ver como aumenta el número de instancias y los nodos o servidores que están atendiendo las peticiones.



¿Cómo funciona Azure Container Instances?

Azure Container Instances (ACI) ejecuta un contenedor en base a una imagen (pública o privada), con la memoria RAM y la CPU que necesitemos. Además de poder añadir una dirección IP pública o montar un volumen de Azure File Share.

Sin ningún tipo de desencadenador implementado en el servicio, nos ofrece una API de administración para crear e instanciar contenedores, además de eliminarlos y conocer su estado.



ACI no ha sido diseñado para ejecutar aplicaciones basadas en Microservicios, ya que no disponemos de ningún tipo de orquestador, como en **Azure Container Services** o en Kubernetes. Sin embargo, puede ser una buena opción para aumentar la capacidad de nuestro cluster en determinados momentos.

¿Cuándo usar ACI?

ACI encaja perfectamente fuera de los límites que Azure Functions tiene por diseño del servicio. Con ACI podemos ejecutar procesos **sin límite de tiempo**, instalar artefactos en el contenedor o montar un volumen usando Azure File Shares.

Por ejemplo, un simple proceso que crea una infraestructura de sitios en SharePoint Online.

Si lo planteamos usando **Azure Functions**, sabemos que los tiempos de respuesta pueden ser superiores a 10 minutos, por lo tanto, tenemos que cambiar a un modo no-serverless (App Service Plan) o utilizar el patrón de Workflow de las Durable Functions para dividir en subprocessos todos los pasos que necesitamos ejecutar.

Con ACI, simplemente tendríamos un contenedor, con el código que necesitamos para crear la infraestructura. Este contenedor, se podría instanciar bien desde una Azure Logic App, que tiene unas actividades específicas para crear instancias, ejecutarlas y eliminarlas.

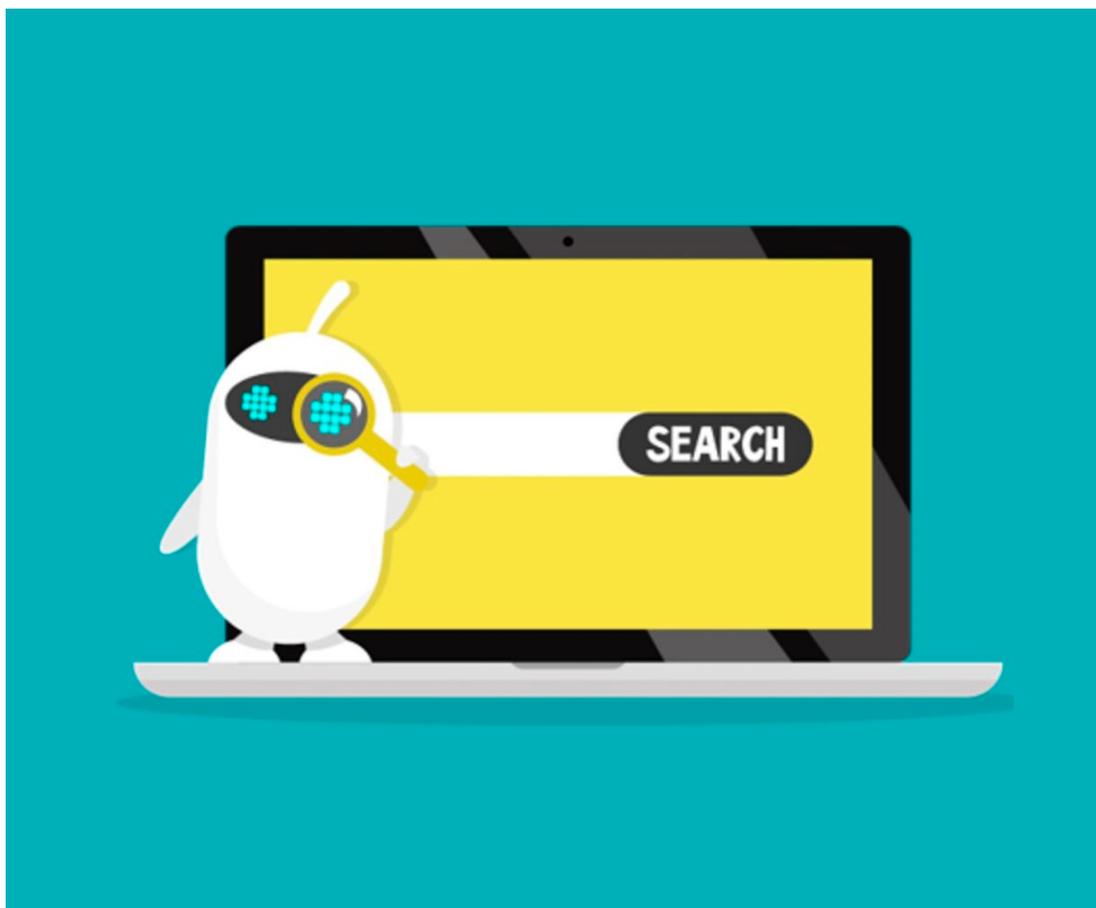
Espero que te haya gustado el artículo y sobretodo, que te haya inspirado :)



Alberto Díaz Martín
CTIO



Cómo activar HTTP/2 en un App Service de Azure

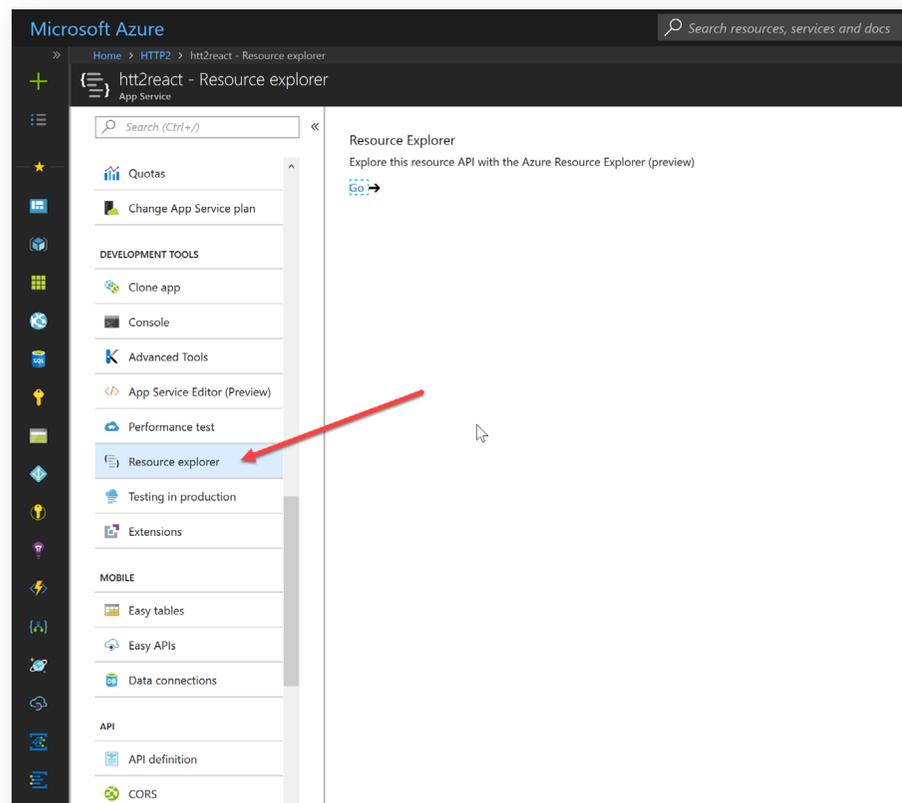


Con [HTTP/2](#) mejoraremos sustancialmente el rendimiento de nuestras aplicaciones que se ejecutan en el App Service de Azure, ¿a qué esperas para activarlo?

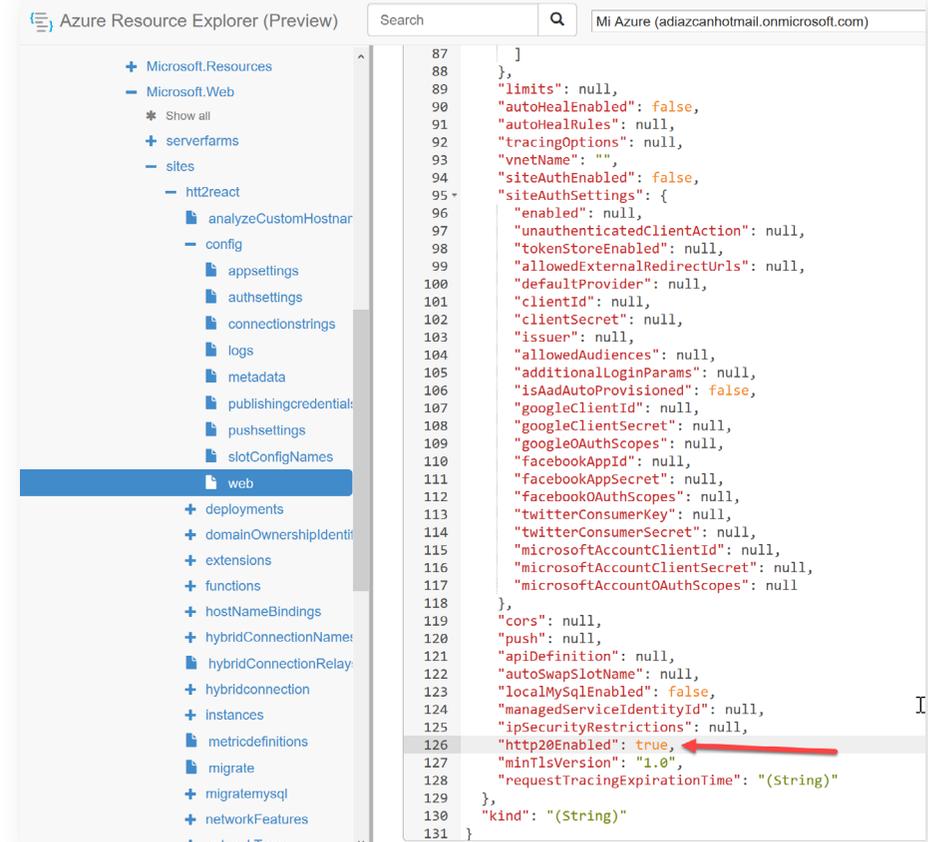
HTTP/2 es el mayor avance desde que hace casi 20 años se publicó HTTP 1.1 y, entre diversos cambios y mejoras, nos permite mantener una única conexión TCP con múltiples peticiones en simultáneo, con un protocolo nativo binario.

Configurar un App Service

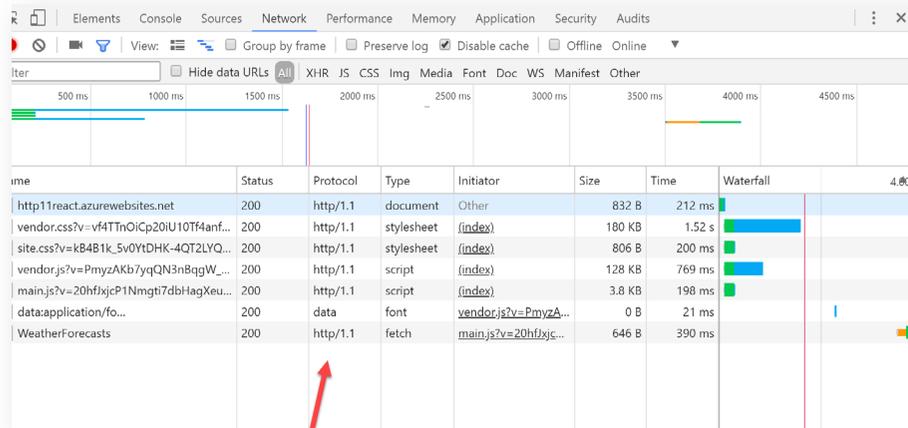
En Azure, ya podemos habilitar HTTP/2 en los App Service, aunque todavía no tenemos interfaz de usuario y hay que modificar el recurso manualmente. Para esto, nos vamos al **Resource Explorer del App Service**.



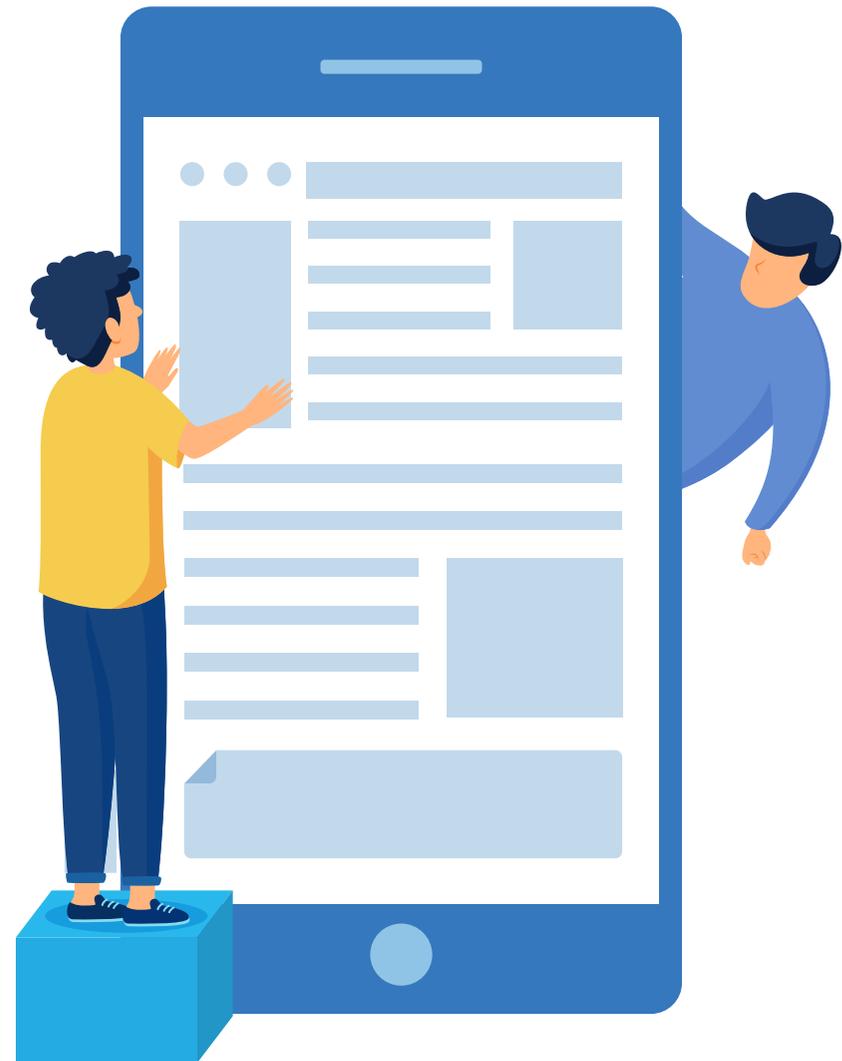
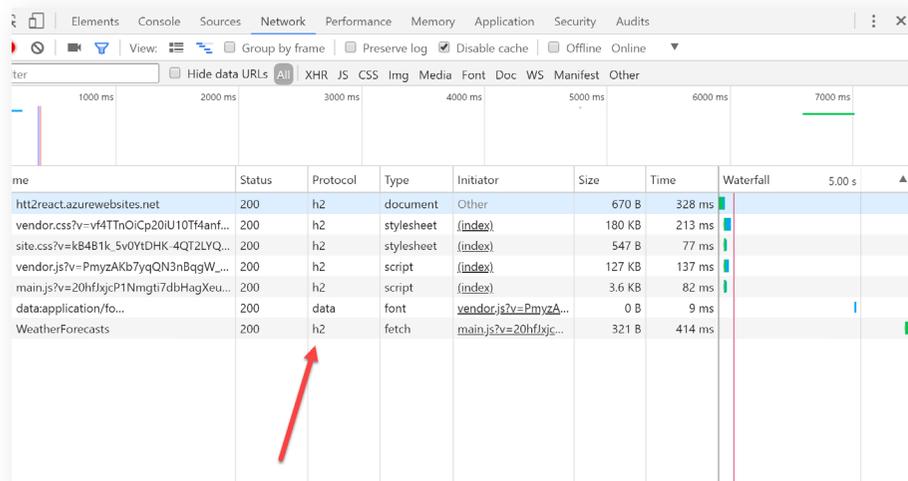
Tenemos que editar la variable `http20Enabled` que se encuentra en Microsoft Web > sites > "nombre del site" > config > web para poner un «true» y que se active el protocolo HTTP/2.



Después de actualizar el recurso, deberíamos de reiniciar el App Service para que pasemos de HTTP/1.1:

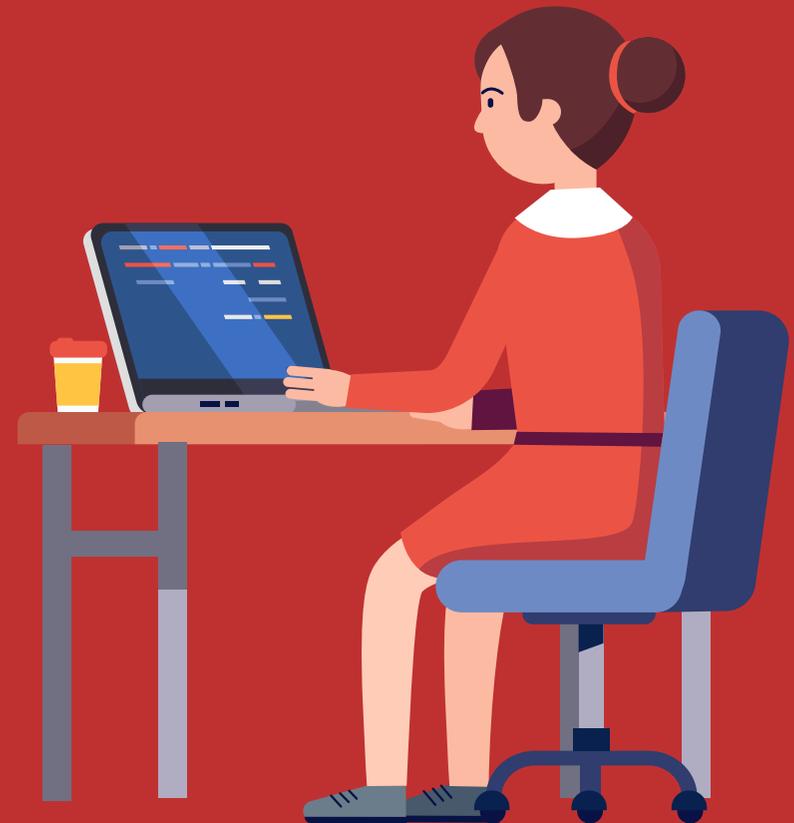
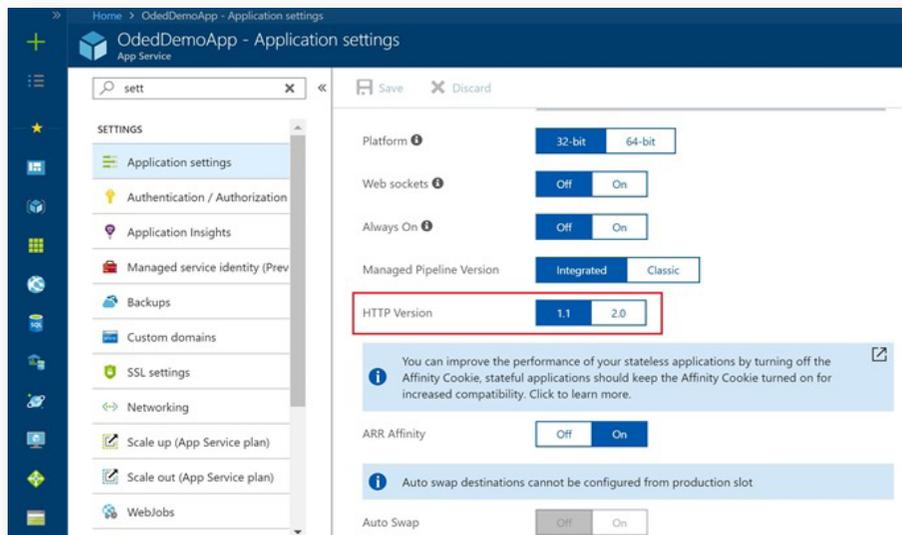


A realizar las conexiones con HTTP/2 (o h2 como nos indican algunos navegadores):



Si te ha resultado útil la información, lo dicho ¿a qué esperas para activarlo? :)

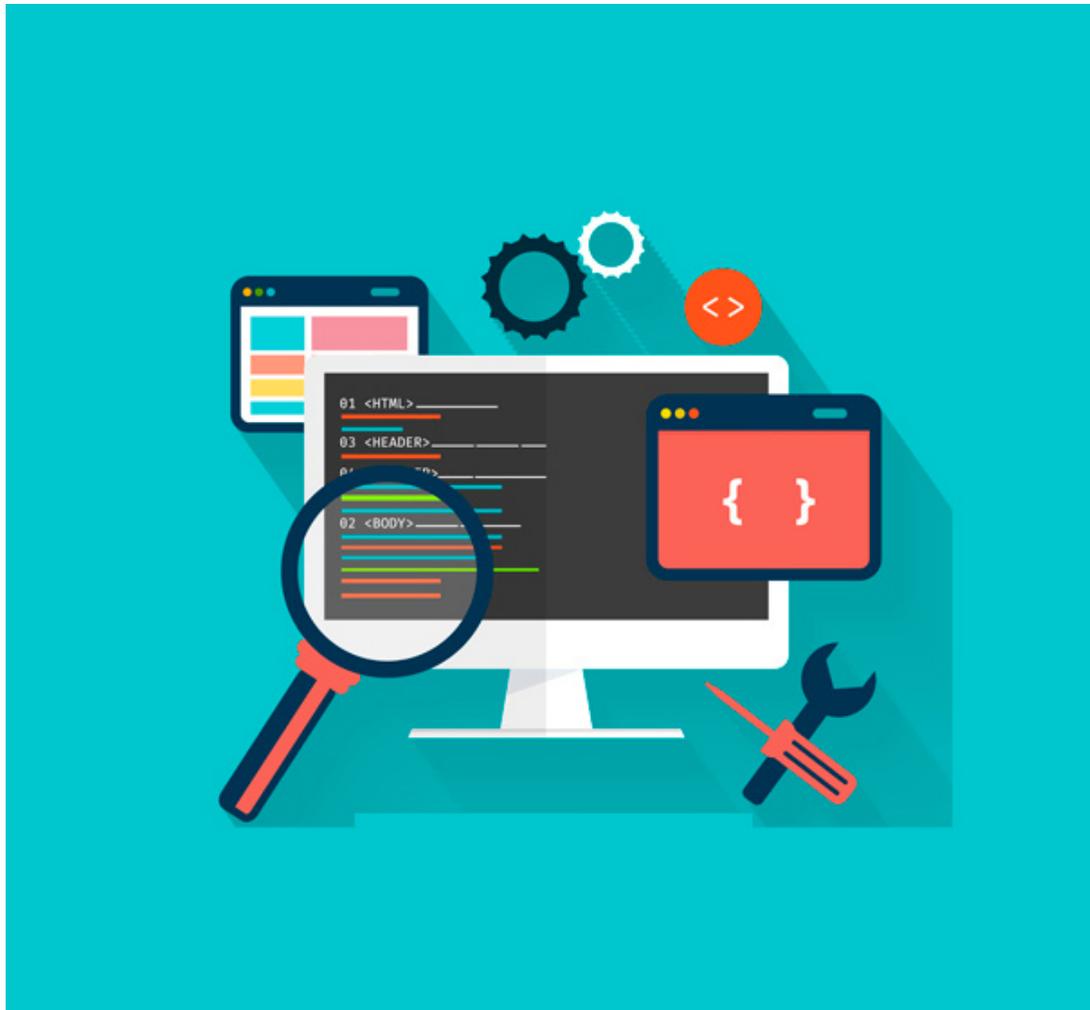
[UPDATE] Ya tenemos interfaz de usuario en el Portal de Azure para activar HTTP/2 en la configuración del App Service.



Alberto Díaz Martín
CTIO



Serverless. Sin servidores y no morir en el intento



La revista [CompartiMOSS](#) ha empezado un ciclo de webcast de debate sobre la tecnología que usamos en nuestro día a día. Este primer webcast se centra en las Arquitecturas Serverless y cómo implementarlo en Azure, con la participación de [Carlos Mendible](#), [Robert Bermejo](#), [Sergio Hernandez](#) y un servidor. Os invito a todos a escucharlo. Entre risa y risa, intentamos dar nuestra visión sobre este tipo de Arquitecturas :)

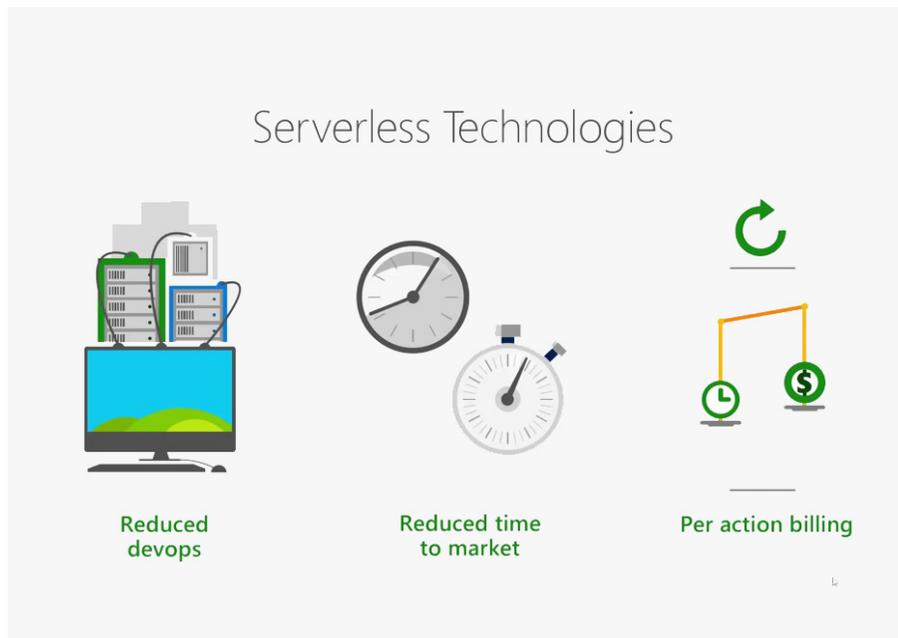


Sin servidores y no morir en el intento

Con @cmendible, @robertbemejo, @shmancebo y @adiazcan

¿Qué es Serverless?

Una arquitectura Serverless se basa en el uso de servicios de terceros o en código que se ejecuta en supuestos contenedores que nos aíslan de las típicas necesidades tradicionales de dimensionar servidores y nos permiten centrarnos en el contexto de ejecución de nuestra aplicación. Este modelo, nos va a llevar a reducir el coste de ejecución y la complejidad de nuestras aplicaciones.



Serverless en Azure

Con esta definición, vamos a necesitar servicios donde nuestra aplicación se ejecute. En Azure tenemos los siguientes:

- > [Azure Functions](#) un servicio para publicar nuestro código que escala bajo demanda para atender las necesidades de ejecución. Como lenguaje de programación podemos usar JavaScript, C#, Java, Python, PHP, Bash Batch y PowerShell, lo que nos abre un mundo de posibilidades para llevar nuestras aplicaciones a este servicio, el cual se ejecuta a partir de suscripción a eventos como peticiones HTTP, programaciones, subida de un fichero a un storage, un nuevo mensaje en una cola de mensajería, ...
- > [Logic App](#) permite crear workflows como procesos de integración de aplicaciones. Uno de sus potenciales reside en los más de 200 conectores de aplicaciones SaaS y sistemas OnPremises, entre los que podemos destacar SQL Server, Office 365, SharePoint, Dynamics 365, Salesforce, etc. Además,

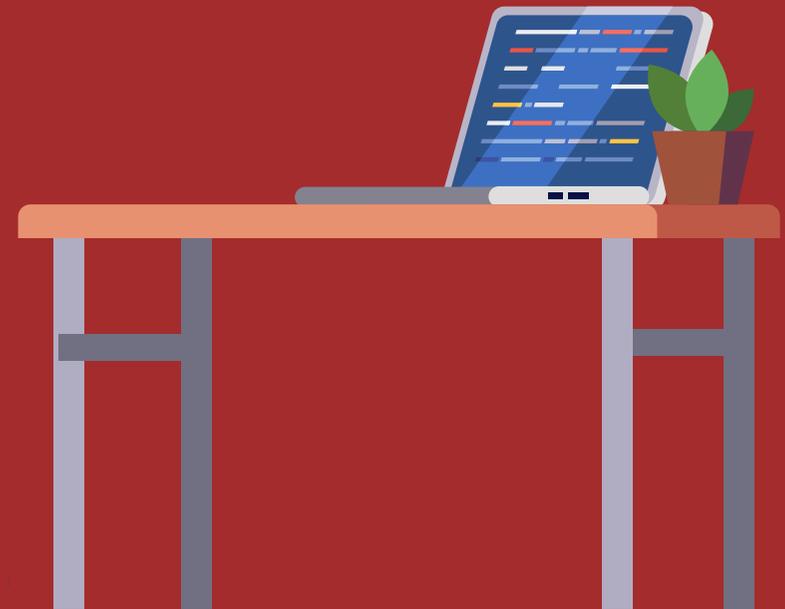
incluye capacidades de mensajería empresarial con conectores EDIFACT, X12 y AS2, entre otros.

El objetivo es hacer transformaciones e integraciones de aplicaciones con un workflow como base del proceso.

> [Event Grid](#) responde a una necesidad de recibir en tiempo real los eventos que se producen en nuestras aplicaciones. En realidad, es un servicio que permite administrar el enrutamiento de los mensajes que se reciben basado en el concepto publicador/es y suscriptor/es. Nuestras aplicaciones tendrán la capacidad de enviar o recibir mensajes a través del protocolo HTTP.

Con estas piezas del puzle, tendremos que diseñar e implementar nuestra aplicación a la que seguro le podremos poner elementos Serverless.

Alberto Díaz Martín
CTIO



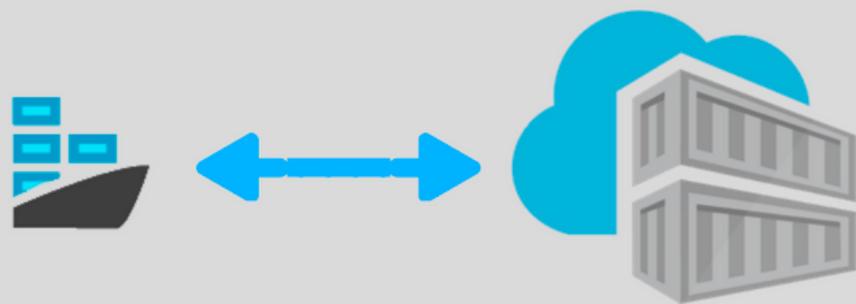




¡NO PUEDO PREDICAR
EL ODIO Y LA
GUERRA CUANDO SOY
UNA DISCÍPULA
DE AZURE!



Azure Container Registry: cómo configurarlo

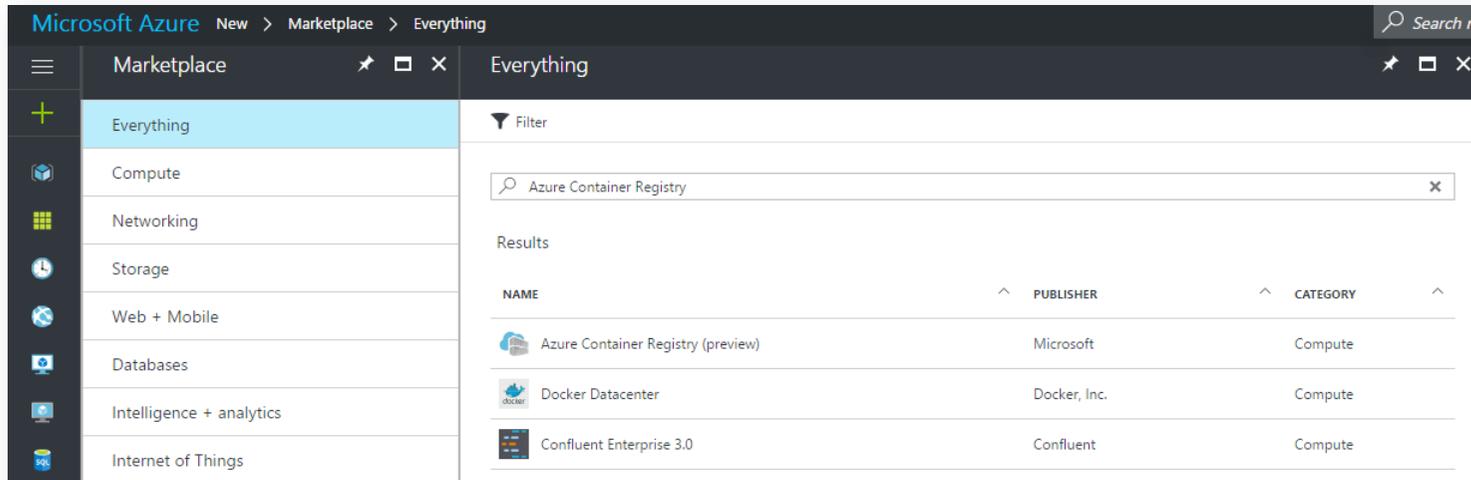


Azure Container Registry

En el post de hoy, vamos a mostrar cómo configurar un **Azure Container Registry** (ACR a partir de ahora). Este servicio nos proporciona un **repositorio** para colocar nuestras imágenes **Docker**, para que las podamos usar fácilmente desde cualquier administrador de contenedores, sobre todo los que proporciona Azure mediante **Azure Container Service** (como Kubernetes, Mesosphere y DC/OS) y que cubriremos en posteriores entradas.

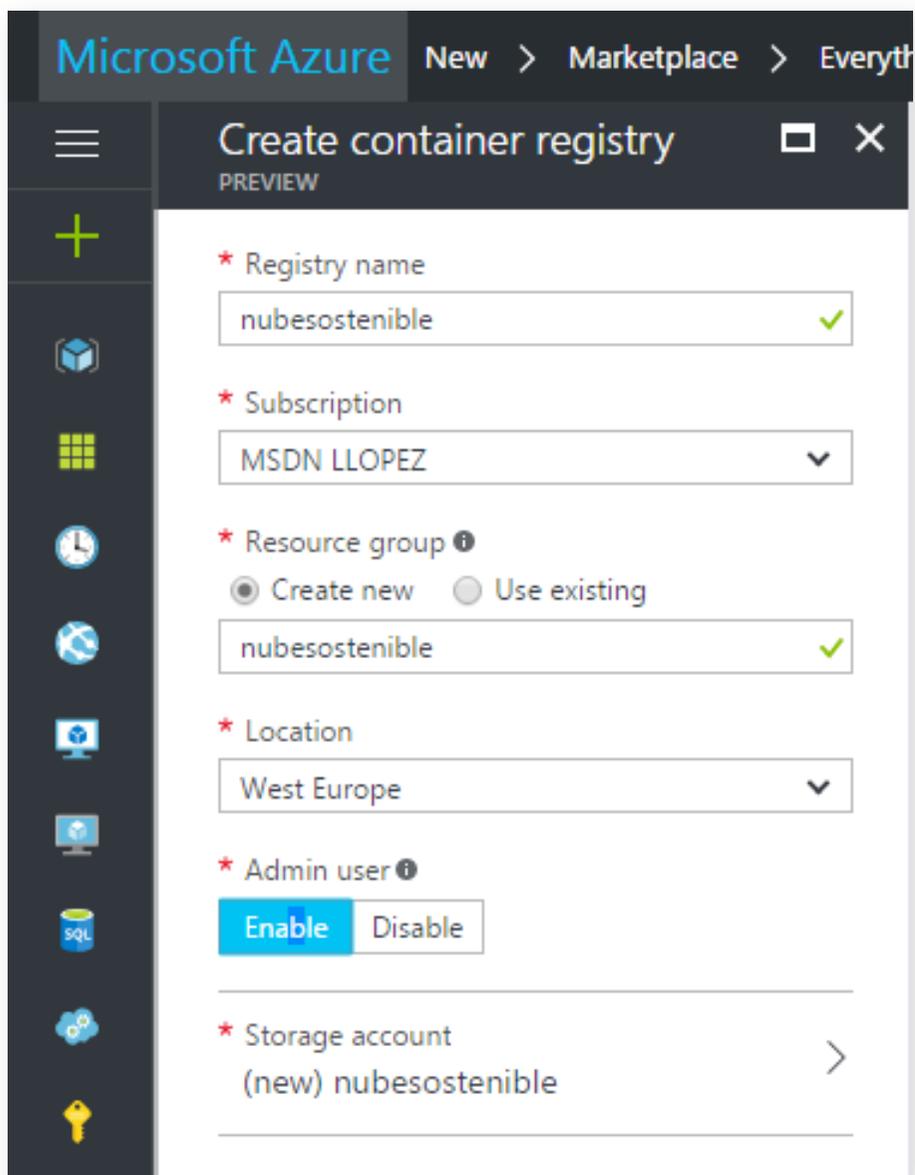
Este servicio se construye sobre un grupo de recursos y una cuenta de almacenamiento, que configuraremos en el momento de la creación.

En primer lugar, desde el **Marketplace** de Azure, buscaremos **'Azure Container Registry'** y tras seleccionarlo, pulsaremos el botón de crear en la nueva cuchilla que se despliega. Tras esto, es el momento de rellenar el nombre del grupo de recursos y de la cuenta de almacenamiento que se crearán para el ACR.



Es importante tener en cuenta, que si queremos hacerlo fácil, debemos **habilitar el usuario administrador**. Si no lo hacemos, deberemos **crear SPNs en Azure Active Directory** y darles privilegios para que puedan logarse en el contenedor. Esta acción esta más allá del alcance de este post.

Nosotros trabajaremos con el **usuario admin**. Esta configuración puede cambiarse en cualquier momento.



Después del asistente, se creará nuestro ACR. Es el momento de comenzar a trabajar.

Siguiendo con el propósito de demostración de este post, vamos a realizar algunas operaciones básicas en nuestro nuevo ACR. En primer lugar, **descargar una imagen** del registro público de Docker, **cambiarle el TAG** y **subirla** a nuestro repositorio privado. Después, **arrancaremos un contenedor** desde nuestro registro.

Debido a que nuestro contenedor es privado, necesitamos **credenciales** para utilizarlo. Estas son el usuario administrador que hemos configurado antes. Las podemos localizar en el **portal de Azure** (si vamos al ACR, en la pestaña '**Access Key**'). Aquí tenemos el nombre del servidor, el usuario y la contraseña.



nubesostenible - Access key

Container registry - PREVIEW

Search (Ctrl+/)

- Overview
- Activity log
- Access control (IAM)
- Tags
- Quick start

SETTINGS

- Access key**
- Repositories
- Locks
- Automation script

SUPPORT + TROUBLESHOOTING

- New support request

Registry name

nubesostenible

Login Server

nubesostenible-on.azurecr.io

Admin user ⓘ

Username

nubesostenible

Password

cY=2uV4==Y/A=kbawDJF+xW60ByEL/NG

Ya estamos listos para comenzar. Vamos a **descargar la imagen de NGINX** del repositorio de Docker.

```
Administrador: Windows PowerShell
PS C:\Windows\system32> docker pull nginx
Using default tag: latest
latest: Pulling from library/nginx
5040bd298390: Pull complete
31123d939af1: Pull complete
23f1bdd267a9: Pull complete
Digest: sha256:4296639ebdf92f035abf95fee1330449e65990223c899838283c9844b1aac4c
Status: Downloaded newer image for nginx:latest
PS C:\Windows\system32>
```

Ahora es el momento en el que debemos hacer **login en nuestro ACR** con los datos recuperados del portal.

```
Administrador: Windows PowerShell
PS C:\Windows\system32> docker login nubesostenible-on.azurecr.io -u nubesostenible -p cY=2uV4==Y/A=kbawDJF+xW60ByEL/NG
Login Succeeded
PS C:\Windows\system32>
```

Ya casi estamos listos para subir la imagen al ACR, pero antes, debemos **añadir el TAG** adecuado a la imagen, para que apunte a nuestro registro.

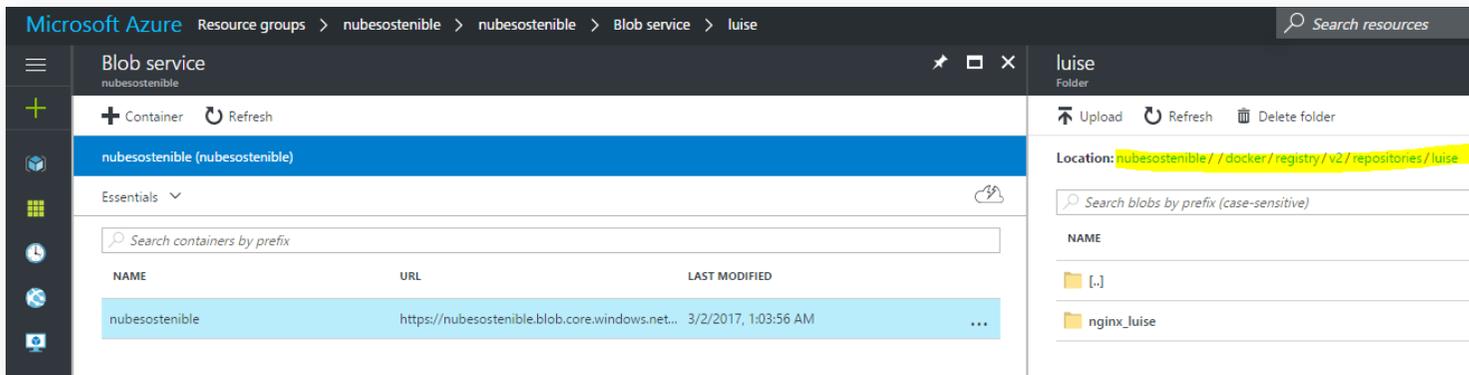
```
Administrador: Windows PowerShell
PS C:\Windows\system32> docker tag nginx nubesostenible-on.azurecr.io/luise/nginx_luise
PS C:\Windows\system32>
```

Tras esto, la subimos:

```
Administrador: Windows PowerShell
PS C:\Windows\system32> docker push nubesostenible-on.azurecr.io/luise/nginx_luise
The push refers to a repository [nubesostenible-on.azurecr.io/luise/nginx_luise]
a82b6c66a6d4: Pushed
1941ca4a7a84: Pushed
a2ae92ffcd29: Pushed
latest: digest: sha256:9e81e4ce4899448e5e7aea69a72dfd1df989a7a0fe7365ad63be1133f05acf10 size: 948
PS C:\Windows\system32>
```

Ya tenemos la imagen en el contenedor.

Si ahora indagamos en la **cuenta de almacenamiento** asociada al ACR, veremos que se ha generado una **estructura de directorios** que corresponde con la definida en el TAG.



Vamos a probar a **levantar la imagen** de nuestro contenedor.

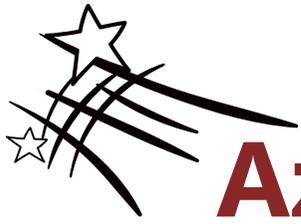
```
Administrador: Windows PowerShell
PS C:\Windows\system32> docker pull nubesostenible-on.azurecr.io/luise/nginx_luise
Using default tag: latest
latest: Pulling from luise/nginx_luise
Digest: sha256:9e81e4ce4899448e5e7aea69a72dfd1df989a7a0fe7365ad63be1133f05acf10
Status: Image is up to date for nubesostenible-on.azurecr.io/luise/nginx_luise:latest
PS C:\Windows\system32> docker run nubesostenible-on.azurecr.io/luise/nginx_luise
PS C:\Windows\system32> docker ps
CONTAINER ID   IMAGE                                COMMAND                                CREATED
STATUS        PORTS                               NAMES
490f569988eb   nubesostenible-on.azurecr.io/luise/nginx_luise   "nginx -g 'daemon ..."   Less than a second ago
Up 44 seconds  80/tcp, 443/tcp                       brave_tesla
```

OK. Hemos sido capaces de **descargar la imagen de nuestro registro y ejecutarla**.
Así que ya esta!!

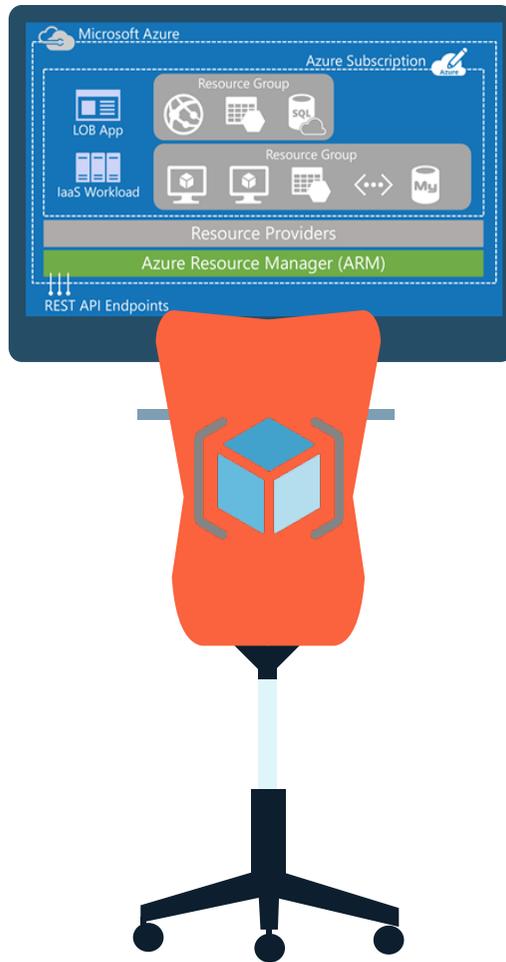
Have Fun!!!



Luis Emilio López López
Cloud Solutions Specialist



Azure Resource Manager



Hasta hace poco cuando administrábamos una aplicación en Azure compuesta por varios recursos (máquina virtual, base de datos, red virtual, cuenta de almacenamiento...), lo hacíamos de forma individual. Con la llegada del nuevo portal de Azure, se ha introducido **Azure Resource Manager** (ARM), dejando de lado la administración clásica que hacíamos con Azure Service Management (ASM). Utilizando ARM agrupamos todos los elementos que componen una aplicación y mediante plantillas podemos implementar, actualizar o eliminar el conjunto de la solución en una sola operación.

Beneficios de usar ARM en lugar de ASM

> Implementar, administrar y supervisar todos los recursos de la solución en grupo en lugar de hacerlo de forma individual.



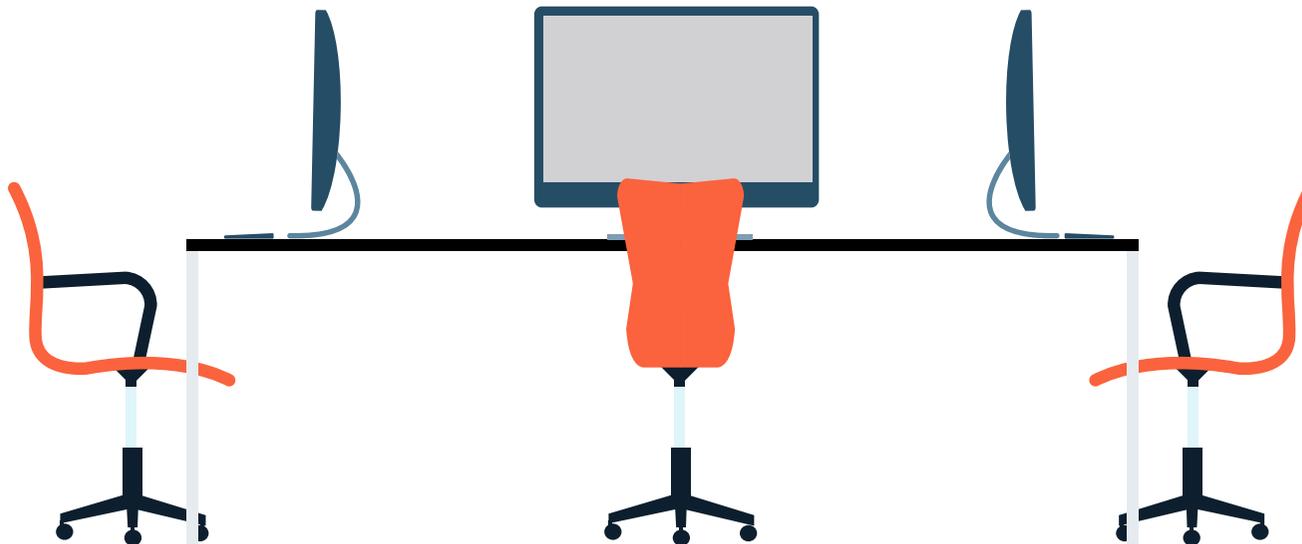
The screenshot shows the Microsoft Azure portal interface. The top navigation bar includes the Microsoft Azure logo and the text "Grupos de recursos". The left sidebar contains a navigation menu with various icons and labels, including "Grupos de recursos", "Inicio rápido", "Costos de los recursos", "Implementaciones", "Propiedades", "Bloqueos", "Script de automatización", "Métrica", "Reglas de alerta", "Registros de diagnósticos", "Application Insights", "Log analytics (OMS)", and "Búsqueda de registros". The main content area displays the "Información esencial" section for a resource group, showing the subscription name, subscription ID, and location (Oeste de Europa). Below this, there is a table with 7 elements, each with a name, type, and location. The table columns are labeled "NOMBRE", "TIPO", and "UBICACIÓN".

NOMBRE	TIPO	UBICACIÓN
[Redacted]	Red virtual	Oeste de Europa
[Redacted]	Cuenta de almacen...	Oeste de Europa
[Redacted]	Máquina virtual	Oeste de Europa
[Redacted]	Microsoft.Compute/...	Oeste de Europa
[Redacted]	Interfaz de red	Oeste de Europa
[Redacted]	Dirección IP pública	Oeste de Europa
[Redacted]	Grupo de seguridad...	Oeste de Europa

> Controlar el acceso basado en la asignación de roles (RBAC), otorgando el nivel de permiso específico que requieran los usuarios para poder realizar sus tareas.

The screenshot displays the Azure IAM control interface. The main window is titled 'Control de acceso (IAM)' and shows a list of roles. A secondary window titled 'Agregar acceso' is open, showing the process of adding access. The 'Agregar acceso' window has two steps: 1. 'Seleccionar rol Propietario' and 2. 'Agregar usuarios 1 Usuario, 0 Grupos'. A third window titled 'Seleccionar rol' is also open, showing a list of roles including 'Propietario', 'Colaborador', 'Lector', 'Administrador de acceso de usu...', and 'Administrador de seguridad'.

USUARIO	FUNCIÓN	ACCESO
Administradores de suscripciones	Propietario	Heredado



> Etiquetar los grupos para tenerlos organizados y poder revisar la facturación permitiéndonos conocer el coste de la solución.

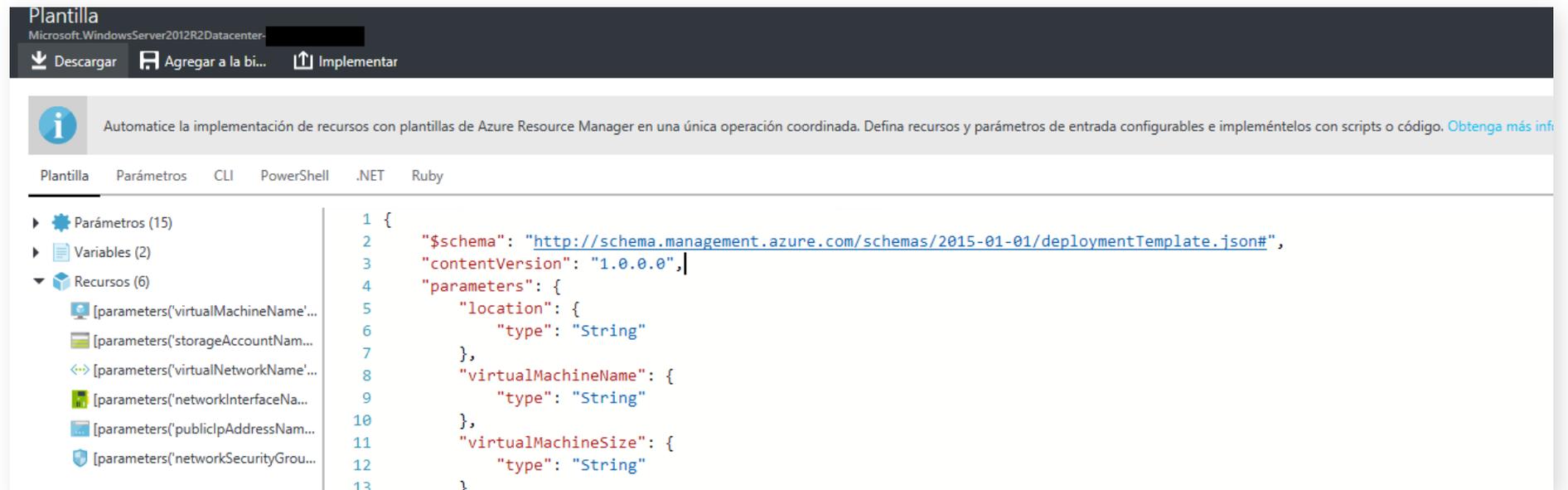
Costes de los recursos

NOMBRE	TIPO	GASTO (EUR)
[Redacted]	Cuenta de almacenamiento	3,17
[Redacted]	Máquina virtual	1,52
[Redacted]	Dirección IP pública	0,02
[Redacted]	Máquina virtual	0,00
[Redacted]	Interfaz de red	0,00
[Redacted]	Grupo de seguridad de red	0,00
[Redacted]	Red virtual	0,00

> Administrar la infraestructura mediante plantillas declarativas en lugar de scripts. Las plantillas son archivos de notación de objetos JavaScript (JSON) donde se definen los recursos.

Uso de plantillas

Para empezar a utilizar plantillas, podemos crearlas desde cero o ver, modificar y descargar la plantilla de alguna implementación que hayamos realizado de forma manual desde el portal.



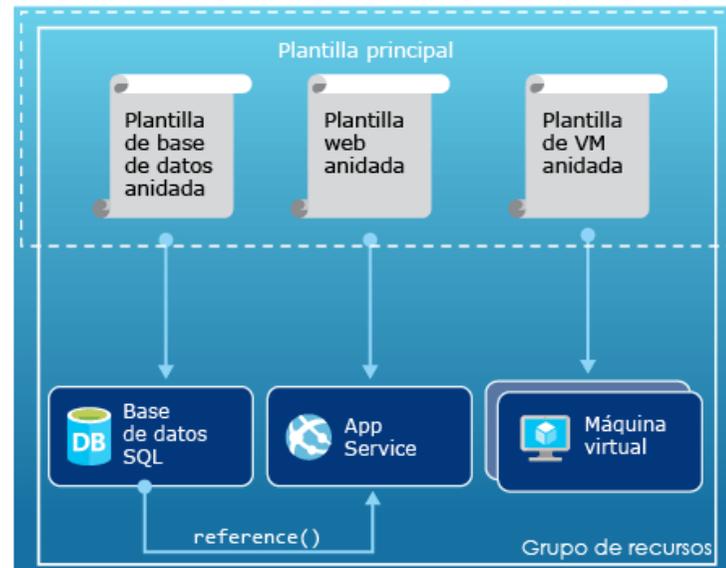
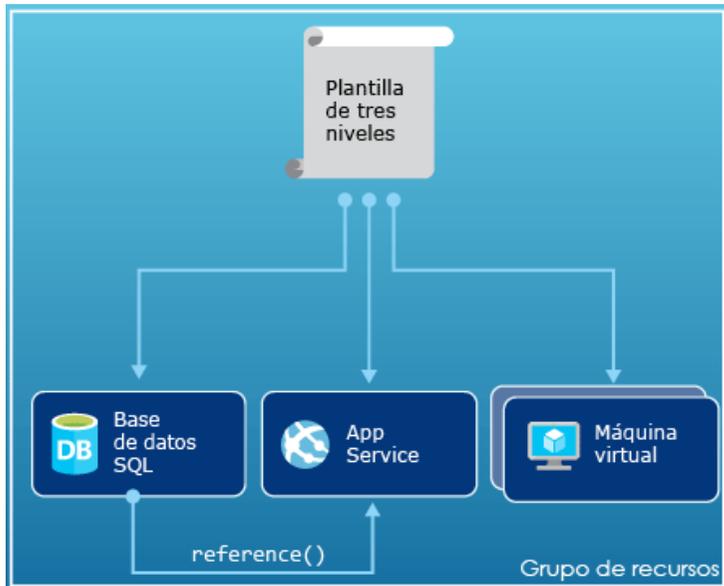
The screenshot shows the Azure Resource Manager template editor interface. At the top, there's a header with the title "Plantilla" and the path "Microsoft.WindowsServer2012R2Datacenter". Below the header, there are three buttons: "Descargar", "Agregar a la bi...", and "Implementar". A main banner contains an information icon and text: "Automatice la implementación de recursos con plantillas de Azure Resource Manager en una única operación coordinada. Defina recursos y parámetros de entrada configurables e impleméntelos con scripts o código. [Obtenga más inf](#)". Below the banner, there are tabs for "Plantilla", "Parámetros", "CLI", "PowerShell", ".NET", and "Ruby". The "Plantilla" tab is active, showing a JSON template. On the left, there's a tree view with categories: "Parámetros (15)", "Variables (2)", and "Recursos (6)". The "Recursos" category is expanded, showing several resource definitions with their parameter placeholders. The main area displays the JSON code for the template, which includes a schema reference, content version, and a parameters section with definitions for location, virtualMachineName, and virtualMachineSize.

```
1 {
2   "$schema": "http://schema.management.azure.com/schemas/2015-01-01/deploymentTemplate.json#",
3   "contentVersion": "1.0.0.0",
4   "parameters": {
5     "location": {
6       "type": "String"
7     },
8     "virtualMachineName": {
9       "type": "String"
10    },
11    "virtualMachineSize": {
12      "type": "String"
13    }
14  }
```

También existen repositorios de plantillas que nos pueden ser de utilidad, por ejemplo en [GitHub](#).

A la hora de definir las plantillas y sus recursos, no es necesario utilizar una única plantilla para toda la solución.

En algunas ocasiones, conviene separar los recursos en diferentes plantillas, de esta forma podemos reutilizarlas en diferentes soluciones, vinculando las necesarias en cada caso.



Una vez tengamos definidas las plantillas, se pueden desplegar directamente desde Visual Studio o cargar la plantilla en Azure para guardarla en la biblioteca de soluciones e implementarla desde allí.

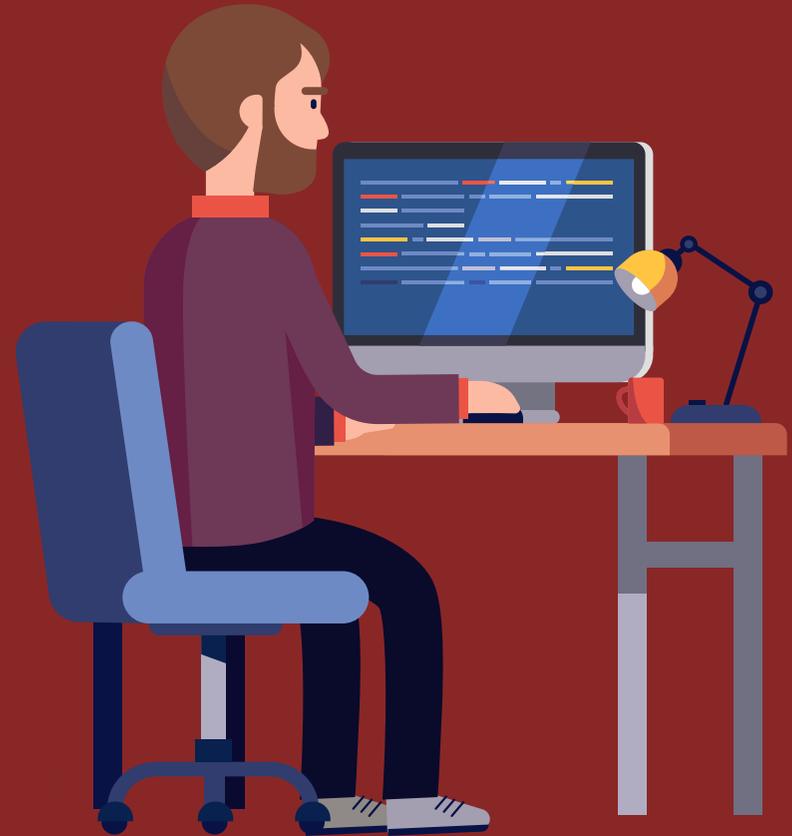


Conclusión

Usando **Azure Resource Manager** con plantillas, podemos garantizar que nuestras soluciones se despliegan y actualizan correctamente, ya que nos permite tener el control de código fuente necesario para asegurar que las soluciones no se desvíen de la configuración definida.

En implementaciones automáticas, esto es fundamental, debido a que los cambios realizados de forma manual no se reflejan en el código fuente.

Felipe López Agudo
IT & Cloud Specialist





Orquestando proyectos: VSTS, Slack, SonarQube



Cuando se está desarrollando una aplicación, y durante todo el ciclo de vida de la misma, se plantean una serie de puntos muy importantes como son: conseguir un producto libre de errores, mediante un desarrollo rápido y con una buena comunicación interna del equipo.

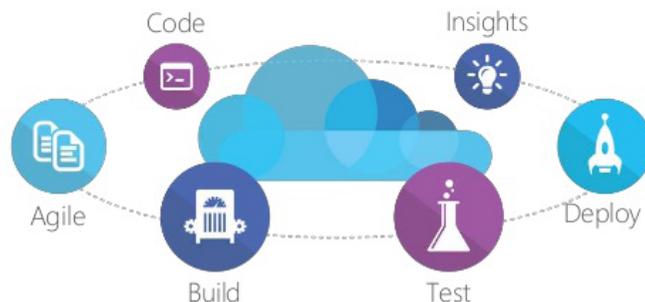
Para conseguir todo esto vamos a hablar sobre **Visual Studio Team Services**, o **VSTS**, que proporciona mecanismos para organizar proyectos, el código desarrollado, automatizar compilaciones, Test y despliegues en distintos entornos, **Slack** como herramienta de comunicación y **SonarQube** para asegurar que el código resultante sea más limpio y tenga menos errores en el futuro.

Visual Studio Team Services

Comenzamos repasando **Visual Studio Team Services** (a partir de ahora VSTS), la herramienta por excelencia de Microsoft para gestionar todas las fases del ciclo de vida.

Engloba varios “subproductos”. Uno de ellos es la **gestión de código fuente**, permitiendo trabajar varias personas en el mismo proyecto sin problemas y facilitando la clara separación de código desplegado en distintos entornos, gracias a la funcionalidad de las ramas y de combinar el código entre ellas.

Visual Studio Team Services



VSTS también permite plasmar **las tareas**, en agrupaciones por áreas e interacciones, y **casos de usuario**, que se deberán completar para llevar a cabo un proyecto, pudiendo describirlas, fijarles una prioridad y esfuerzo, asignarlas a usuarios que podrán cambiarlas de estado durante el progreso de la tarea, y siempre guardando un histórico de todos cambios sufridos.

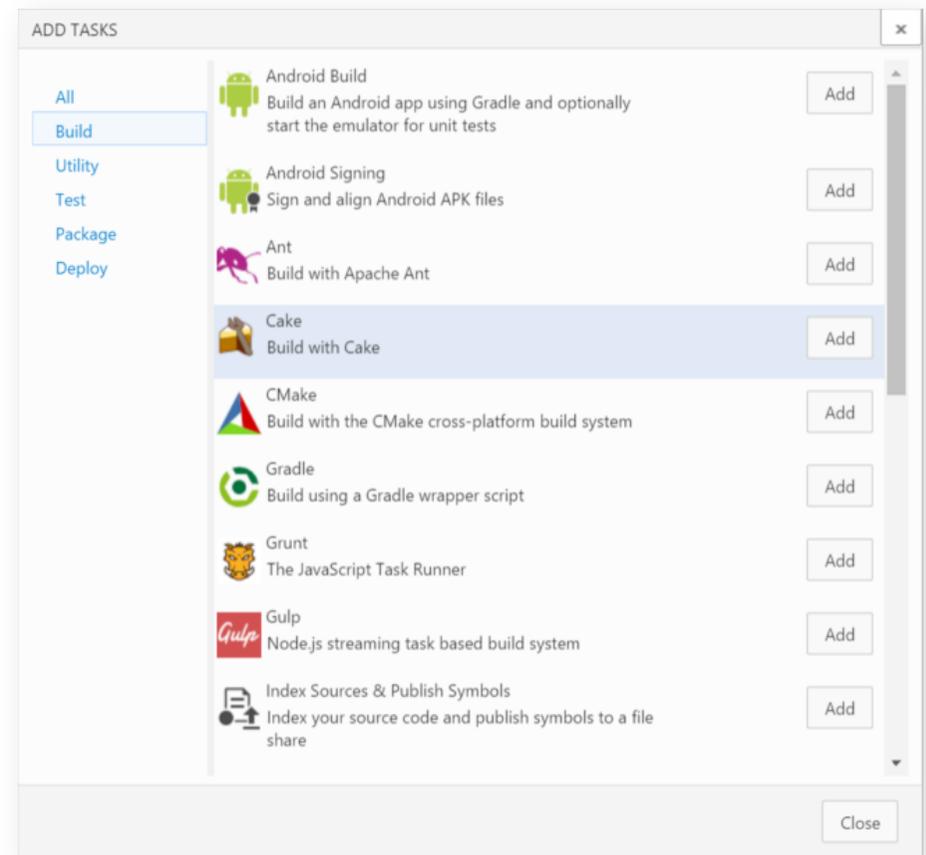
Otra ventaja de describir estas tareas y bugs es la de poder asociar conjuntos de cambios del código a éstos.

Otro de los grandes beneficios que ofrece VSTS, son las **definiciones de compilación**, o “**Build definitions**”. Atras quedaron aquellos tiempos de las Build XAML donde añadir una modificación era una tarea muy compleja y difícil.

Ahora las definiciones de compilación permiten realizar una serie de acciones secuenciales,

pudiendo tener en cuenta o no el estado resultante de la tarea anterior, y configurarlas con disparadores/triggers en función de eventos sobre el proyecto o en base a una programación establecida. Con estas definiciones se podría conseguir, por ejemplo, que con cada conjunto de cambios subido al repositorio se comprobara que el código compila sin errores y que pase los test desarrollados, que todas las noches se vuelva a comprobar la integridad de los desarrollos y que se desplieguen en un entorno de integración, o que cuando se suban cambios en una determinada rama se despliegue esa actualización en un determinado entorno.

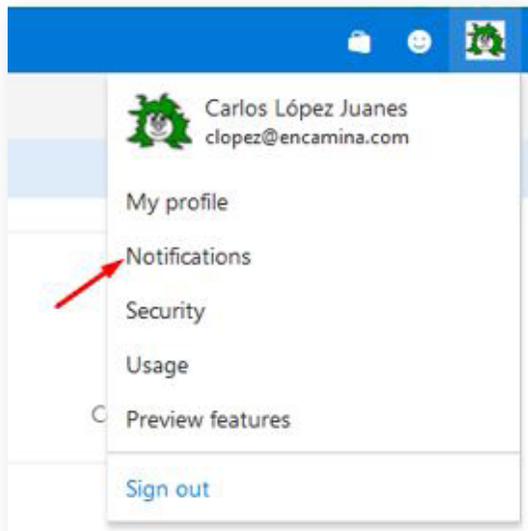
Para trabajar con las definiciones de compilación se hace desde la pestaña Build & Release, donde se almacenan todas las definiciones creadas y se pueden configurar agregando, o quitando, tareas desde el conjunto de disponibles:



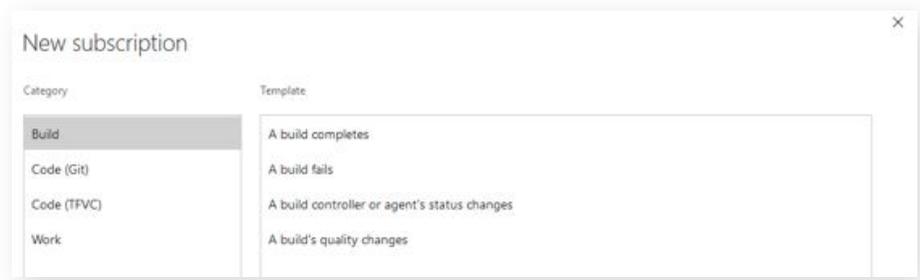
Entre las **acciones disponibles** se pueden encontrar de distintos tipos: compilación, test, utilidades, generación de paquetes y despliegues. Dentro de estas categorías se encuentran además interacciones con otros sistemas como puede ser Azure, NuGet, Sonar... o ejecutar acciones npm, Grunt, Gulp... Este listado de tareas es extensible en el Marketplace de VSTS.

Una característica de VSTS que ayuda a conocer el estado de las definiciones de compilación y de mucho más, son las **notificaciones configurables**. Éstas permiten conocer desde cuándo se ha creado, editado o borrado un elemento hasta crear avisos a partir de subidas de conjuntos de cambios de código o ejecuciones de definiciones de compilación.

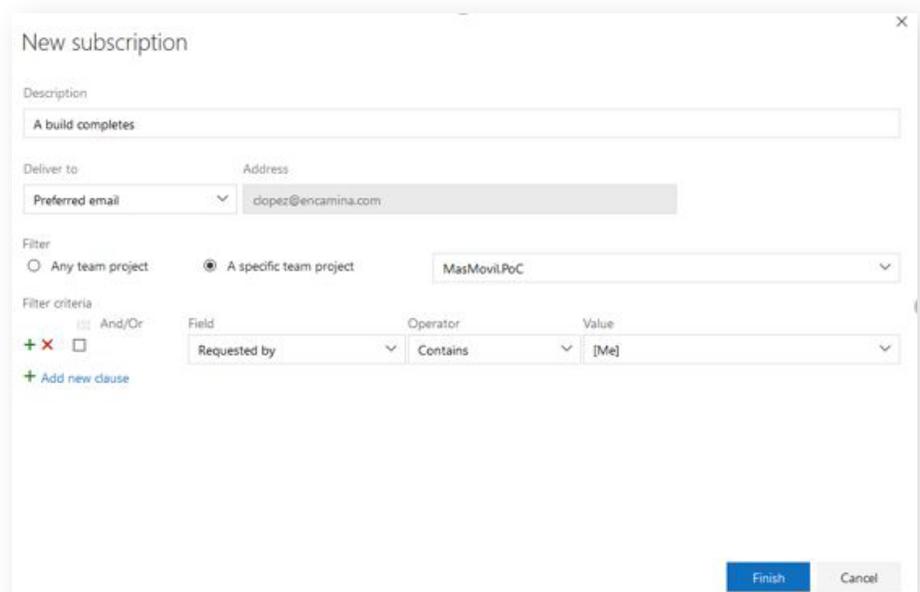
Esta configuración se realiza desde la opción **Notificaciones del menú de usuario**:



A continuación, se pueden crear y configurar las notificaciones necesarias:



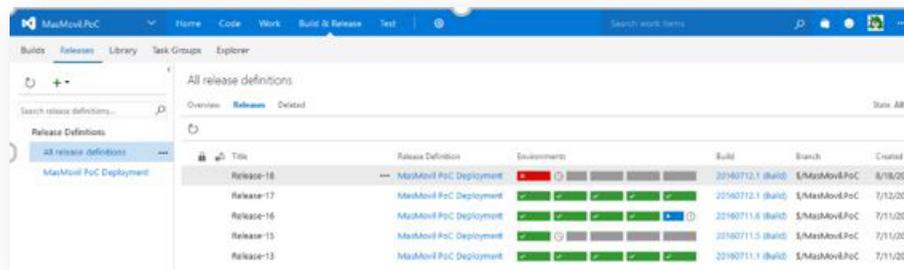
Por ejemplo, se puede configurar que se envíe una notificación cuando una definición de compilación lanzada por mí en un determinado proyecto termine:



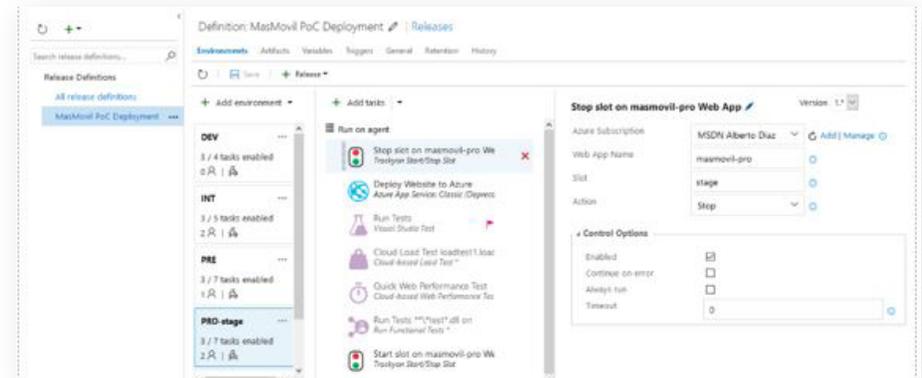
Otra característica de VSTS que permite tener un gran control de los despliegues de las

actualizaciones entre los distintos entornos es la Gestión de despliegues, o **Release Management**.

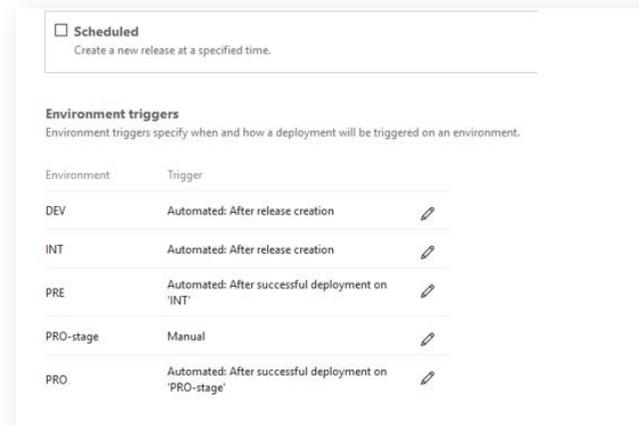
Esta funcionalidad permite, a partir de la definición de los entornos, aplicar a cada uno de ellos unos pasos como, por ejemplo, despliegues en aplicaciones web de Azure, ejecución de Test predefinidos, lanzar pruebas de rendimiento, gestionar los Slot de las aplicaciones web de Azure, etc. La Gestión de despliegues se encuentra en la sección Releases dentro del menú Build & Release:



Inicialmente se muestra un resumen de los despliegues en curso o finalizados, y también se puede acceder a la edición de la definición:



Las acciones que se pueden agregar son las que existen en las definiciones de compilación vistas anteriormente. Esos despliegues por entorno se ejecutarán en función de unos disparadores, o Triggers, en función de subidas de conjuntos de cambios, planificados en el tiempo o en base al estado de finalización de otra definición de despliegue:



Adicionalmente, por cada entorno, se pueden definir una serie de políticas de retención sobre los entornos:

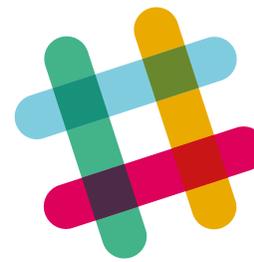
Define retention policy for releases deployed to each environment in this release definition. Learn more about [retention policy](#).

Environment : DEV	Days to retain a release : 60	Retain build : <input checked="" type="checkbox"/>
	Minimum releases to keep : 3	
Environment : INT	Days to retain a release : 60	Retain build : <input checked="" type="checkbox"/>
	Minimum releases to keep : 3	
Environment : PRE	Days to retain a release : 60	Retain build : <input checked="" type="checkbox"/>
	Minimum releases to keep : 3	
Environment : PRO-stage	Days to retain a release : 60	Retain build : <input checked="" type="checkbox"/>
	Minimum releases to keep : 3	
Environment : PRO	Days to retain a release : 60	Retain build : <input checked="" type="checkbox"/>
	Minimum releases to keep : 3	



Slack

Slack es una herramienta muy sencilla, pero a la vez potente, para **crear un canal de comunicación entre los integrantes de un proyecto**, permitiendo compartir comentarios, imágenes, ficheros, etc.



slack

Otra de las fortalezas de Slack es su facilidad para integrarse con otros sistemas, incluyendo VSTS. Con un par de sencillos pasos se puede configurar Slack para que muestre los resultados de compilaciones, u otros eventos de VSTS, conectándose mediante Webhooks. Tanto el tipo de evento a registrar como la personalización de la imagen y nombre que se quiere ver en el canal es posible.

Para agregar una integración a cualquier canal de Slack habrá que acceder a la dirección <https://mycompany.slack.com/apps> y configurarlo desde ese mismo punto:

Webhook URL
When setting up this integration, this is the URL that you will paste into Visual Studio Team Services.
[Show setup instructions](#)

Descriptive Label
Use this label to provide extra context in your list of integrations (optional).

Customize Name
Choose the username that this integration will post as.

Customize Icon
Change the icon that is used for messages from this integration.

 or

Donde la integración con VSTS es máxima al obtener también enlaces a las definiciones de compilación lanzadas, y a sus resultados. Un aspecto a tener en cuenta es que a la hora del día a día, el exceso de información puede ocasionar el efecto contrario al que buscamos y es que en lugar de utilizar la herramienta, esta se abandone por exceso de información o bien porque las notificaciones de VSTS se mezclen con las conversaciones de los integrantes. Una de las opciones, es crear un canal de Slack solamente para estas notificaciones y otro canal para las conversaciones del equipo.

Como resultado obtendremos mensajes del siguiente tipo:

Compilación **20170120.16** correcta

Requested by	Duration
Carlos Lopez	00:00:17
Build Definition	
Compilación Check-In	

Compilación **DebtRecovery.Recovery.DEV_20170120.1** correcta

Requested by	Duration
Carlos Lopez	00:00:34
Build Definition	
DebtRecovery.Recovery.DEV	

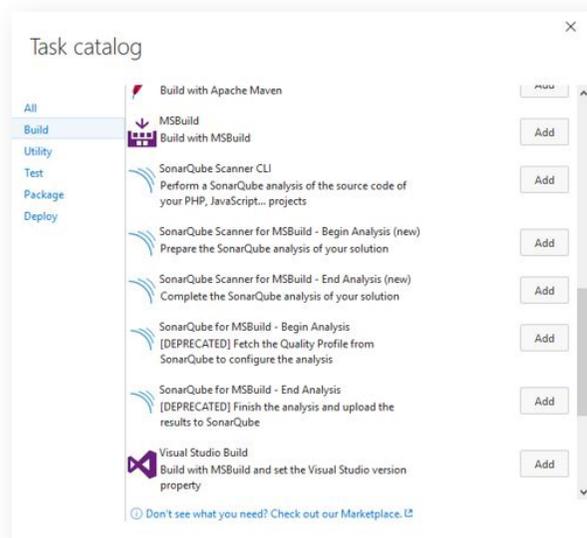
SonarQube

SonarQube permite **mantener una alta calidad del código desarrollado**, integrándose fácilmente con VSTS analiza el código en busca de vulnerabilidades, errores, Code Smells, etc.

Entre las **características del producto** se encuentra el ayudar a escribir un código más

limpio, detectar errores, o posibles puntos de error, que no se consiguen encontrar en la fase de compilación, es capaz de analizar más de 20 lenguajes distintos de programación (C#, JS, Java, C++, Python, ...), muy fácil de integrar durante la Integración Continua mediante el uso de Webhooks o su API REST, y provee de un único punto de acceso a los resultados en un panel donde poder navegar a través de distintos gráficos por la salud del código.

La integración de SonarQube con VSTS es tan sencillo como agregar pasos a las definiciones de compilación:



Conclusión

Hoy en día donde cada vez más la competencia es más alta, y los equipos de desarrollo pueden tender a estar separados geográficamente, se necesitan de mecanismos para mejorar la velocidad de desarrollo, construyendo de manera más libre de errores y hacer la comunicación interna del equipo más fluida y directa.

VSTS ayuda a la **organización de los proyectos**, asegura **obtener código limpio** de errores en fase de compilación y gestiona también **despliegues entre entornos**.

Con **Slack** se consigue una **comunicación rápida** entre los integrantes del equipo y con **SonarQube** es posible detectar puntos que en tiempo de ejecución producirán errores o bajadas de rendimiento.

Carlos López Juanes
Team Leader



Nuestros
autores:

Alberto Díaz Martín

CTIO



Con más de 15 años de experiencia en tecnologías Microsoft, actualmente es parte del equipo de Dirección de ENCAMINA. Organiza y participa en las conferencias más relevantes del mundo Microsoft en España. Autor de diversos libros, en 2013 entró a formar parte de la Dirección de CompartiMOSS, una revista digital sobre tecnologías Microsoft. Desde 2011 es Microsoft MVP en la categoría de Azure. Es fundador de TenerifeDev y coordinador de SUGES.

Luís Emilio López López

Cloud Solutions Specialist



Con más de 17 años de experiencia en el campo de las TIC, actualmente es IT and Cloud Manager en ENCAMINA. Posee las certificaciones MCSE en Cloud platform and infrastructure, Productivity, Messaging y SharePoint. También tiene las certificaciones MCSA en: Cloud Platform, Office 365, Windows Server 2012... Y en realidad le gusta definirse como «un GEEK aprendiz de todo y maestro de nada ;)».

Felipe López Agudo

IT & Cloud Specialist



Técnico de Sistemas con experiencia en el área de servidores y servicios de red, dedicado los últimos años a tecnologías Microsoft. Lleva 5 años trabajando en ENCAMINA y es MCSA en Windows Server 2012 y Office 365. Además está certificado en infraestructuras Azure y en instalaciones, despliegues y administración de CRM. Actualmente forma parte del equipo de sistemas del área de CE como IT & Cloud Specialist.

Carlos López Juanes

Team Leader



Ingeniero en Informática por la Universidad Politécnica de Valencia. MCSD en SharePoint Applications, Web Applications y App Builder, además de MCSA en Web Applications. Lleva desarrollando con tecnologías Microsoft más de 7 años, especialmente sobre entornos web y sobre la plataforma SharePoint. Actualmente es Team Leader en ENCAMINA.



