

Azure Functions y Xamarin



Luis Beltrán

Microsoft MVP

Xamarin Certified Mobile Developer

 @darkicebeam



<http://bit.ly/XamarinDiplomadoITC>



<http://bit.ly/MeetupCelayaNet>



<http://icebeamwp.blogspot.mx>



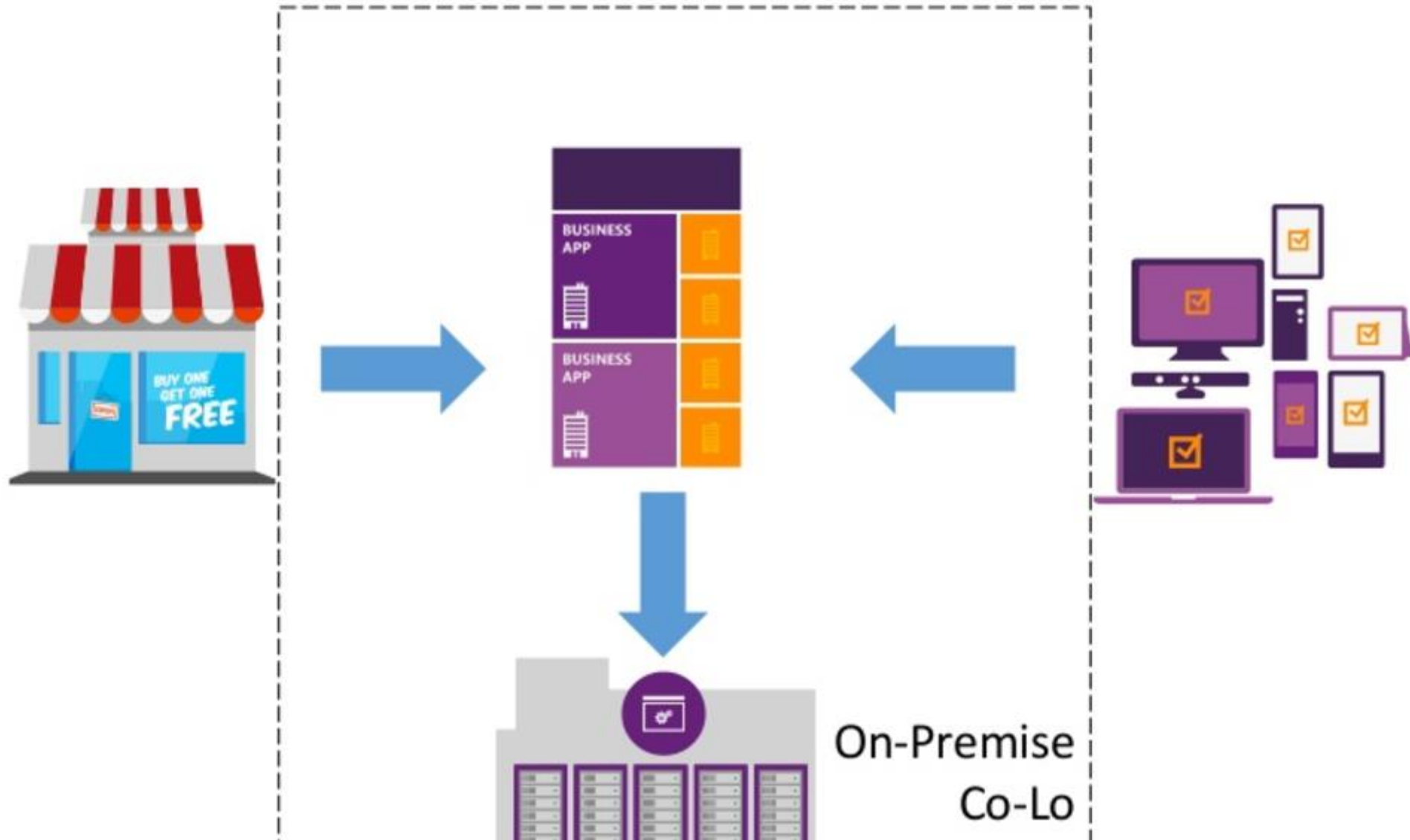
<https://github.com/icebeam7/XamarinCompanyFunctions>



Azure Functions

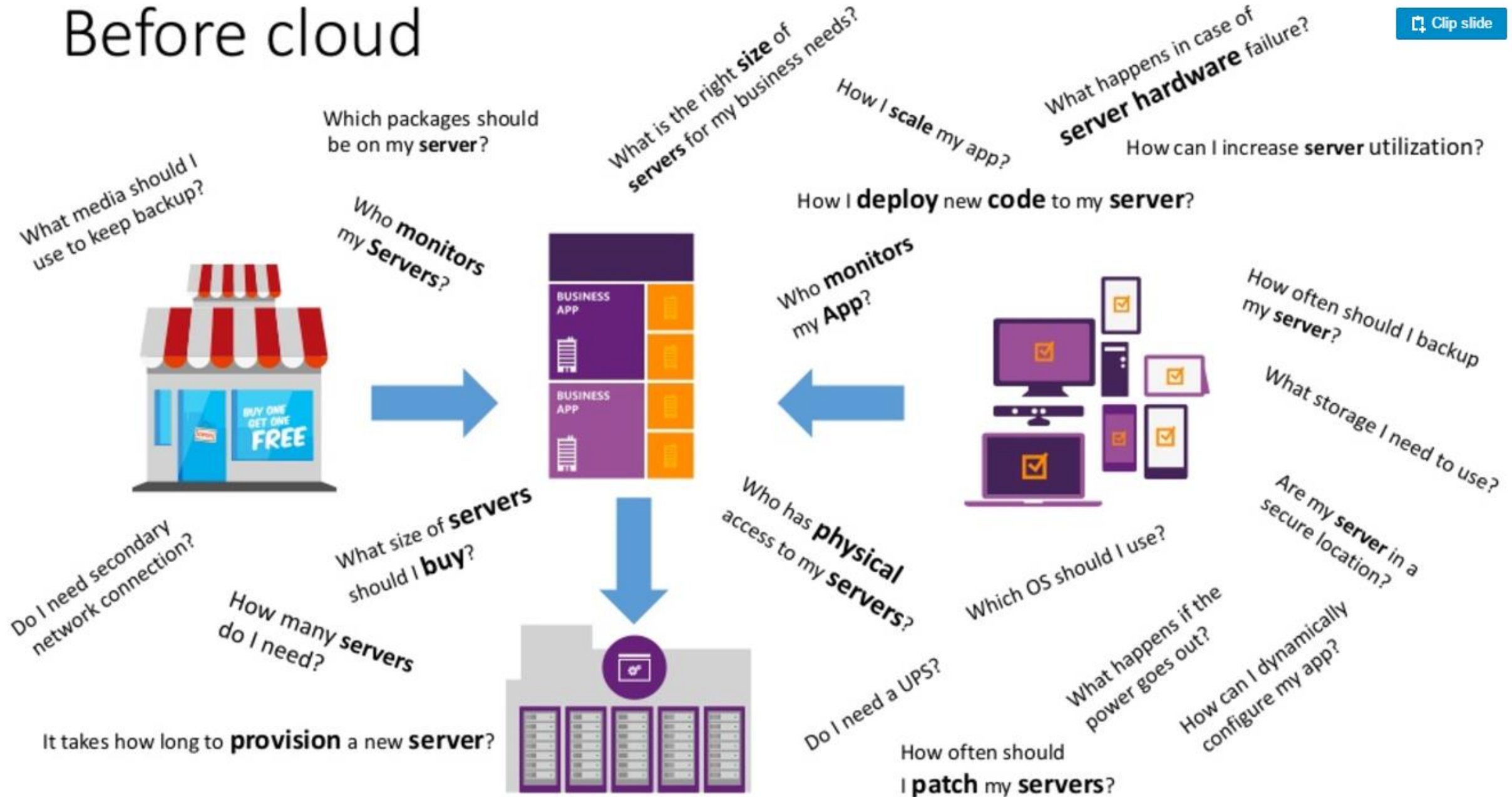
- Azure Functions es uno de los nuevos servicios proporcionados por la nube de Azure.
- Es una solución que permite ejecutar de manera sencilla pequeños fragmentos de código (“funciones”) en la nube, simplemente escribiendo el código que necesitas SIN preocuparte de crear una aplicación completa o la infraestructura necesaria para ejecutarla. En otras palabras, ejecuta *serverless applications* (una arquitectura de código sin servidor).

Before cloud



Before cloud

Clip slide



Is it PaaS time?

Clip slide

Which packages should be on my **server**?

What is the right **size** of **servers** for my business needs?

How I **scale** my app?

How can I increase **server** utilization?

How I **deploy** new **code** to my **server**?

Who **monitors** my **App**?

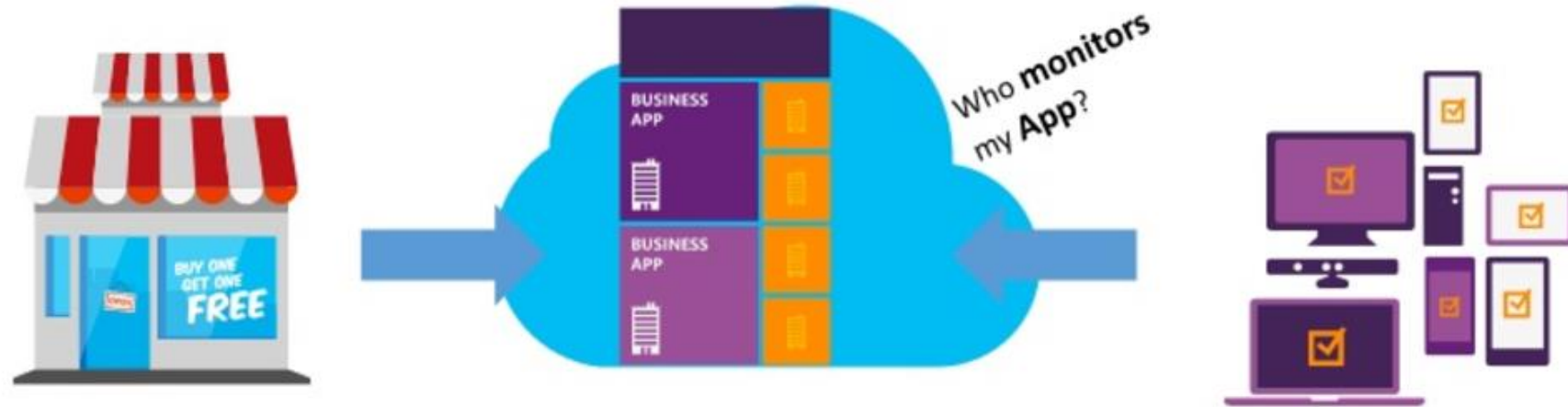
How often should I backup my **server**?

Which **OS** should I use?

How can I dynamically configure my app?

How often should I **patch** my **servers**?

How many **servers** do I need?



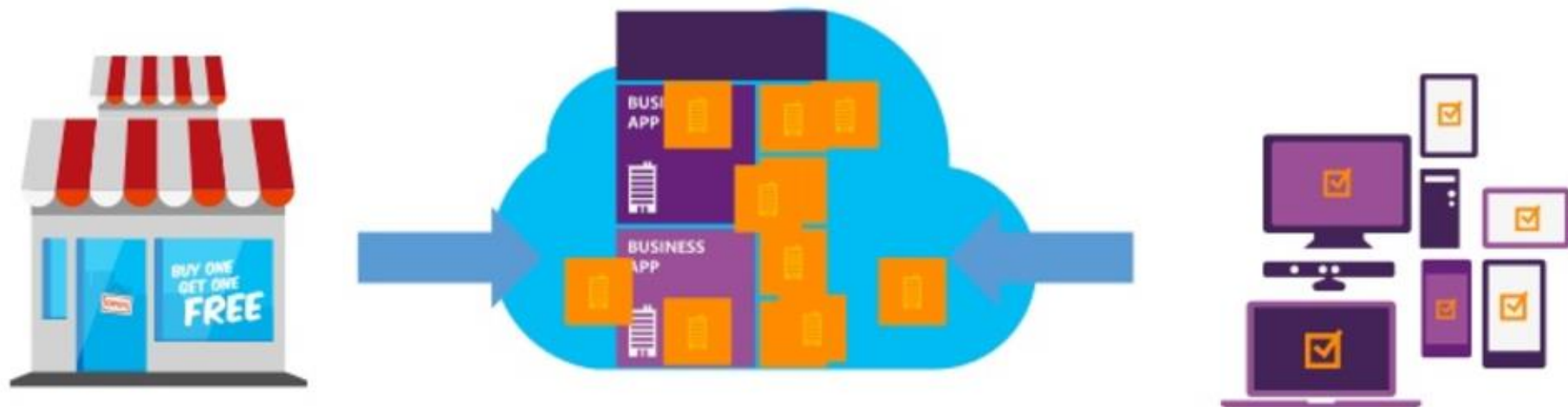
Serverless . . .

Clip slide

What is the right **size** of
servers for my business needs?

How I **scale** my app?

How can I increase **server** utilization?



How many **servers**
do I need?

What is Serverless?



Abstraction
of servers

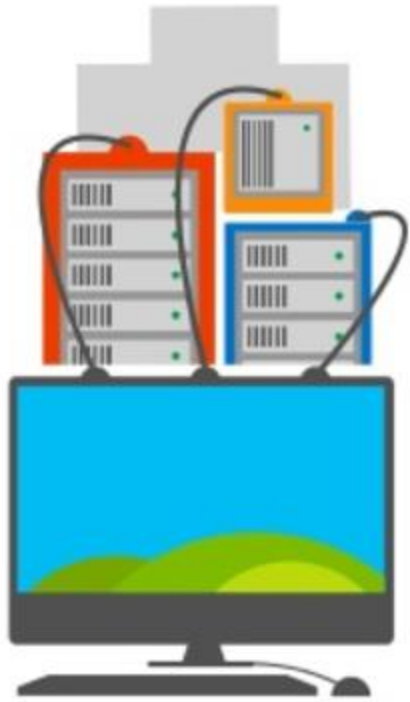


Event-driven/
instant scale



Sub-second
billing

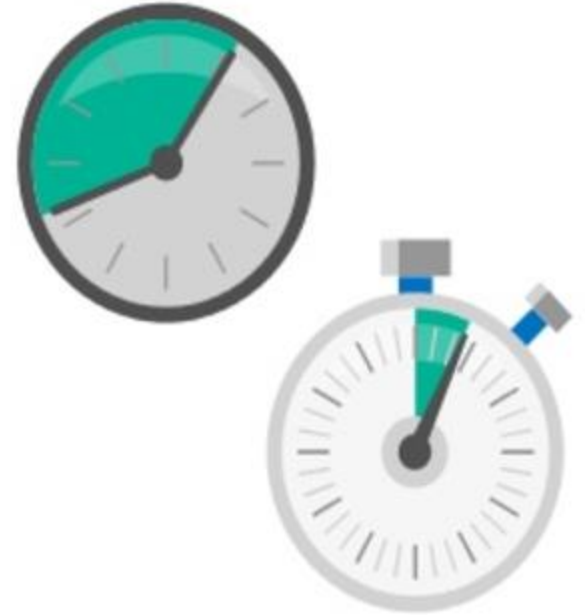
Benefits of Serverless?



Reduced
DevOps



Focus on
Business
Logic



Reduced Time
To Market



Azure Functions

Process events with Serverless code.

Make composing Cloud Apps insanely easy

Develop Functions in C#, Node.js, F#, Python, PHP, Batch and more

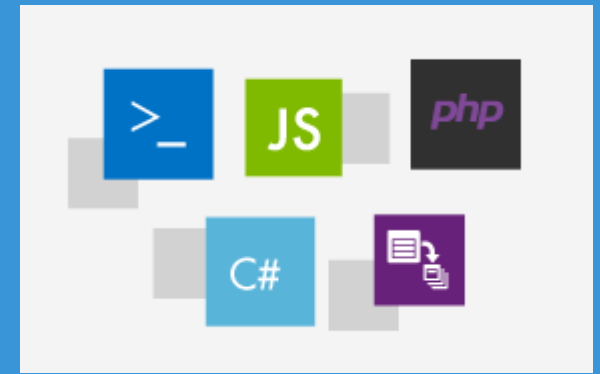
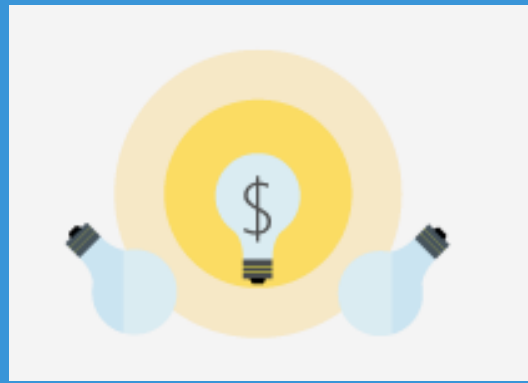
Easily schedule event-driven tasks across services

Expose Functions as HTTP API endpoints

Scale Functions based on customer demand

Easily integrate with Logic Apps

Características



- **Elección del lenguaje:** escriba funciones con C#, F#, Node.js, Python, PHP, batch, bash o cualquier archivo ejecutable.
- **Modelo de precios de pago por uso:** pague solo el tiempo que haya empleado ejecutando el código.
- **Traiga sus propias dependencias:** NuGet para que pueda usar sus bibliotecas favoritas.
- **Seguridad integrada:** proteja las funciones desencadenadas por HTTP con los proveedores de OAuth como Azure Active Directory, Facebook, Google, Twitter y cuenta Microsoft.

Características

- **Integración simplificada:** fácil aprovechamiento de los servicios de Azure y ofertas de software como servicio (SaaS). Para ver algunos ejemplos, consulte la sección de integraciones.
- **Desarrollo flexible:** codifique las funciones directamente en el portal o configure la integración continua e implemente el código mediante GitHub, Visual Studio Team Services y otras herramientas de desarrollo compatibles.
- **Código abierto:** el tiempo de ejecución de Funciones de Azure es de código abierto y está disponible en GitHub.

Situaciones clave (plantillas)

- **BlobTrigger:** procesar blobs de Almacenamiento de Azure cuando se agregan a los contenedores. Esta función se puede usar para cambiar el tamaño de las imágenes.
- **EventHubTrigger:** responder a eventos proporcionados a un Centro de eventos de Azure. Especialmente útil en escenarios de IoT, procesamiento del flujo de trabajo o de la experiencia del usuario y en instrumentación de aplicaciones.
- **Webhook genérico:** procesar las solicitudes HTTP de webhook de cualquier servicio que admita webhooks.

Situaciones clave (plantillas)

- **GitHub webhook:** responder a los eventos que se producen en los repositorios de GitHub.
- **HTTPTrigger:** desencadenar la ejecución del código mediante una solicitud HTTP.
- **QueueTrigger:** responder a mensajes conforme llegan a una cola de Almacenamiento de Azure.

Situaciones clave (plantillas)

- **ServiceBusQueueTrigger:** permite conectar el código a otros servicios de Azure o servicios locales, mediante la escucha de las colas de mensajes.
- **ServiceBusTopicTrigger:** permite conectar el código a otros servicios de Azure o a servicios locales mediante la suscripción a temas.
- **TimerTrigger:** ejecutar limpieza u otras tareas de lote dentro de una programación predefinida.

Escenarios comunes de Azure Functions

Procesamiento basado en temporizador

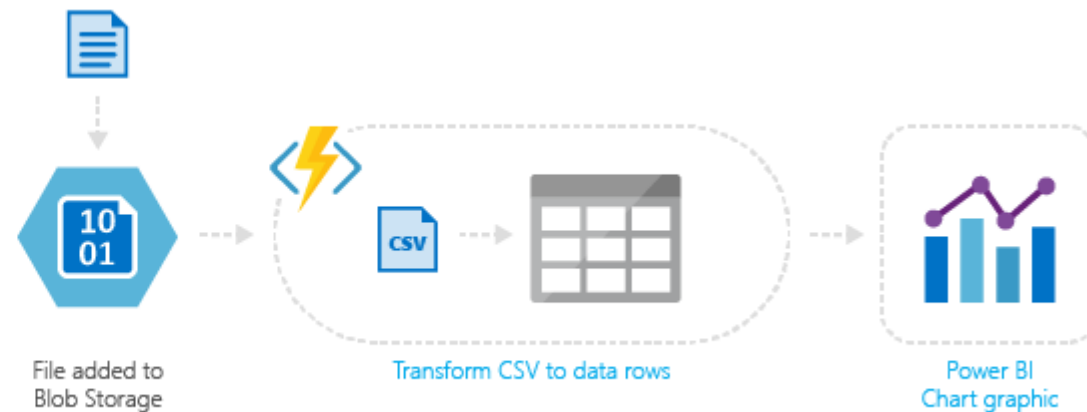
Azure Functions admite un evento basado en un temporizador mediante la sintaxis de trabajos CRON. Por ejemplo, escriba código que se ejecute cada 15 minutos y limpie una tabla de base de datos según una lógica de negocios personalizada.



Procesamiento de eventos de servicio de Azure

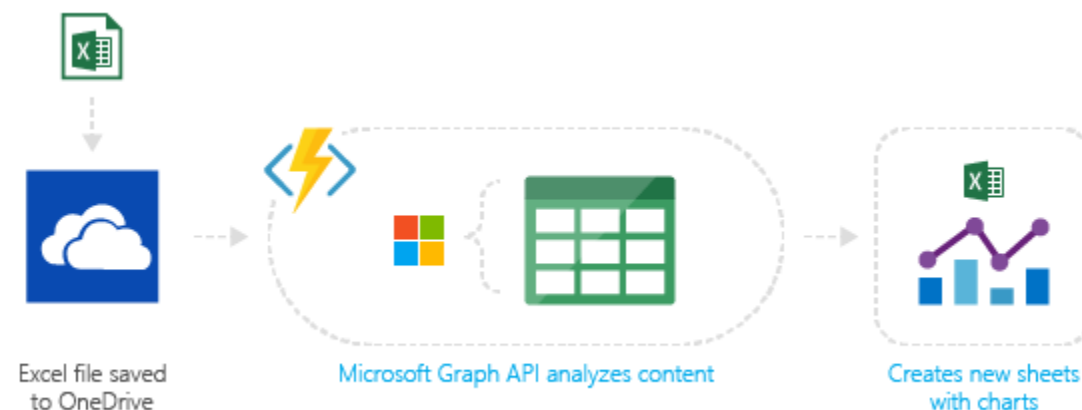
Azure Functions permite desencadenar un evento en función de una actividad en un servicio de Azure. Por ejemplo, ejecute código sin servidor que lea archivos de registro de prueba recién detectados en un contenedor de Azure Blob Storage y transfórmelo en una fila de una tabla de Azure SQL Database.

[Función de Azure en C# para reaccionar a Azure Insights Events >](#)



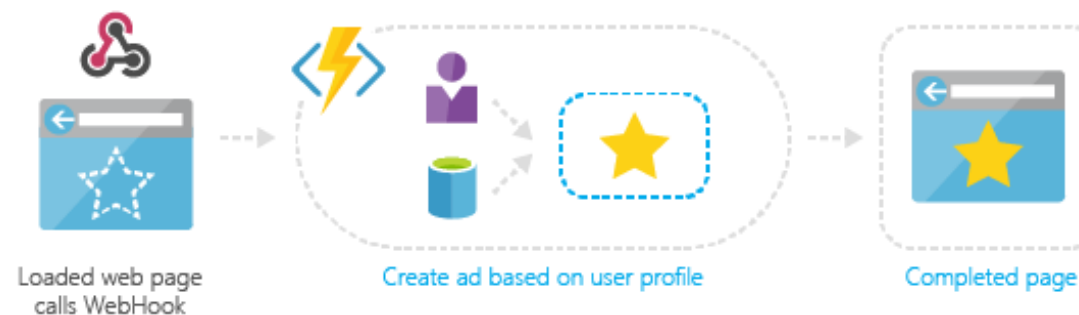
Procesamiento de eventos SaaS

Azure Functions admite desencadenadores en función de la actividad en un servicio SaaS. Por ejemplo, guarde un archivo en OneDrive, que desencadena una función que usa la API Graph de Microsoft para modificar la hoja de cálculo y crea gráficos adicionales y datos calculados.



Arquitecturas de aplicaciones web sin servidor

Azure Functions puede mejorar una aplicación de una sola página. La aplicación llama a las funciones con la URL de WebHook, guarda los datos de usuario y decide qué datos se muestran. O bien, realice personalizaciones sencillas, como cambiar el destino de anuncios mediante una llamada a una función y pasarle información del perfil del usuario.

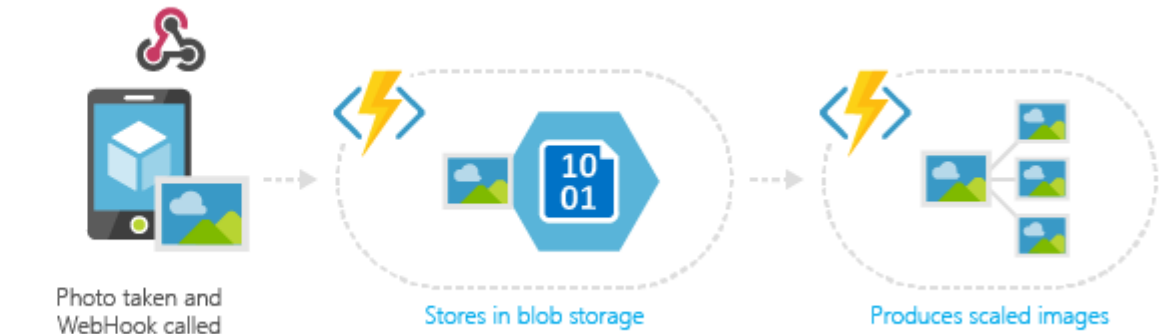


Back-end móviles sin servidor

Un back-end móvil puede ser un conjunto de API HTTP a las que se llama desde un cliente móvil con la URL de WebHook. Por ejemplo, una aplicación móvil puede capturar una imagen y luego llamar a una función de Azure para obtener un token de acceso para cargar en Blob Storage. La carga del blob desencadena una segunda función de Azure, con la que se ajusta el tamaño de la imagen al dispositivo móvil.

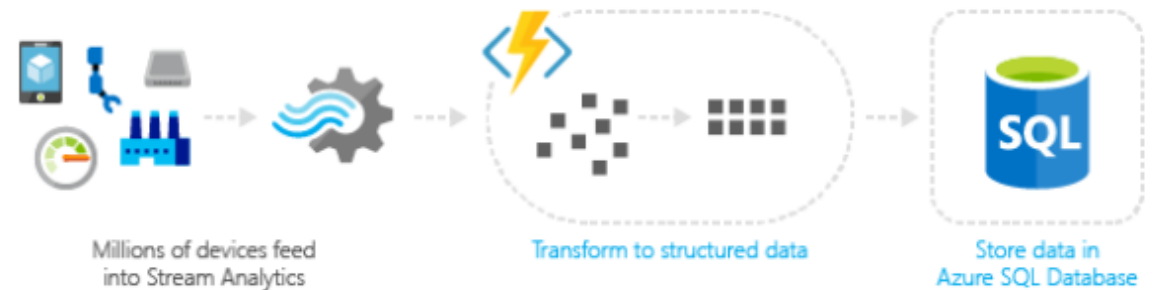
[Función de Azure en Node.js para generar tokens de SAS >](#)

[Función de Azure en C# para generar tokens de SAS >](#)



Procesamiento de datos en tiempo real

Por ejemplo, los dispositivos Internet de las cosas (IoT) envían mensajes a Azure Stream Analytics, que luego llama a una función de Azure para transformar el mensaje. Esta función procesa los datos y crea otro registro en una instancia de Azure SQL Database.



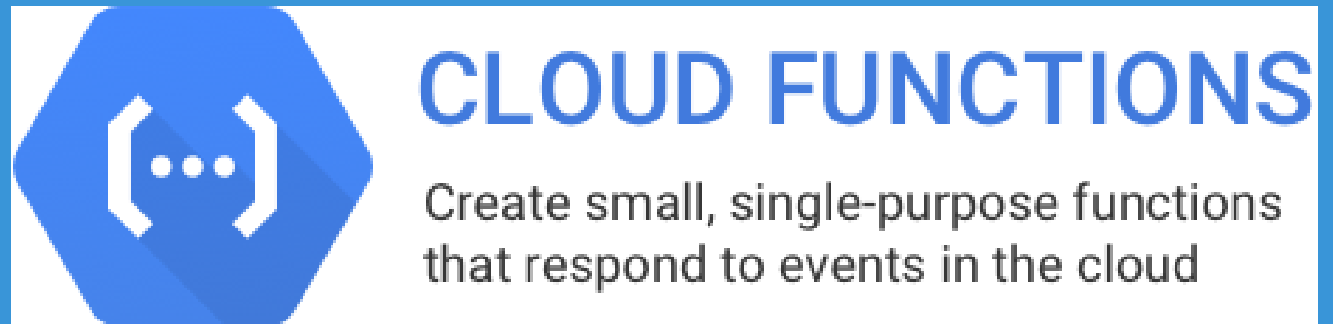
Mensajería de bots en tiempo real

Utilice Azure Functions para personalizar el comportamiento de un bot con la utilización de un webhook. Por ejemplo, cree una función de Azure que procese un mensaje con Cortana Analytics y llame a esta función con Microsoft Bot Framework.



Tecnologías similares

- Google Cloud Functions
- AWS Lambda



Costo

¿Cuánto cuesta Funciones de Azure?

Funciones de Azure tiene dos tipos de planes de precios, elija el que mejor se adapte a sus necesidades:

- **Plan de Consumo:** cuando se ejecuta la función, Azure proporciona todos los recursos informáticos necesarios. No tiene que preocuparse de la administración de recursos y solo paga por el tiempo que haya empleado ejecutando el código.
- **Plan del Servicio de aplicaciones :** se ejecutan las funciones igual que aplicaciones web, móviles y de API. Cuando ya se usa el Servicio de aplicaciones para las otras aplicaciones, las funciones pueden ejecutarse en el mismo plan sin costo adicional.

Puede encontrar todos los detalles de precios en la [página de Precios de Funciones](#). Para más información acerca de cómo escalar sus funciones, consulte [Escalado de Funciones de Azure](#).

Costo

<https://azure.microsoft.com/es-mx/pricing/details/functions/>

Precios de Azure Functions

El plan de consumo de Azure Functions se factura en función del **consumo de recursos** y las **ejecuciones**. Los precios del plan de consumo incluyen una concesión mensual gratuita de 1 millones de solicitudes y 400,000 GB-segundos de consumo de recursos. Los clientes también puede ejecutar Functions dentro de su plan de App Service a las **tarifas** normales del plan de App Service.

Seleccionar columnas ▼

MEDIDOR	PRECIO	CONCESIÓN GRATUITA (POR MES)
Tiempo de ejecución*	MXN\$0.000309/GB/s	400,000 GB/s
Ejecuciones totales*	MXN\$3.86 por millón de ejecuciones	1 millón de ejecuciones

*Las concesiones gratuitas se aplican solo a las suscripciones de consumo de pago.

Nota: Con cada Function App, se crea una cuenta de almacenamiento de forma predeterminada. La cuenta de almacenamiento no está incluida en la concesión gratuita y su uso se cobra a las **tarifas de almacenamiento** normales.

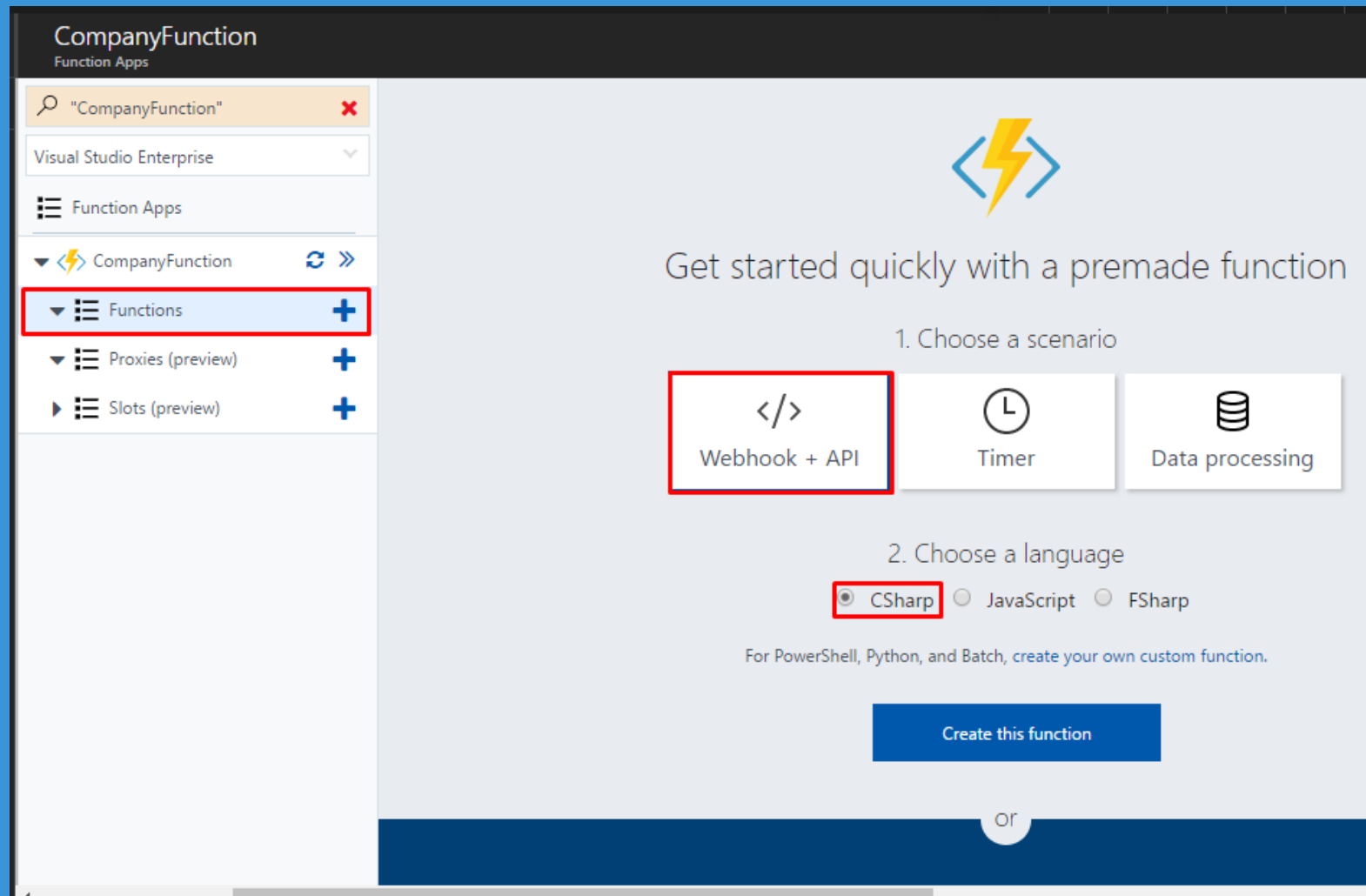
Demo

<https://github.com/icebeam7/XamarinCompanyFunctions>

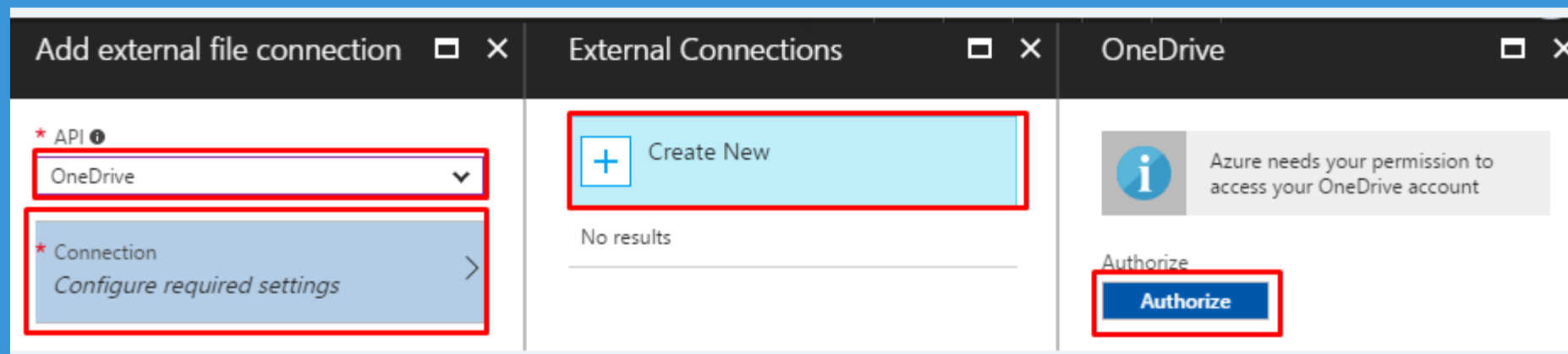
basado en

<http://blog.pieeatingninjas.be/2017/01/29/building-a-xamarin-app-with-azure-functions-and-onedrive/>

Demo



Demo



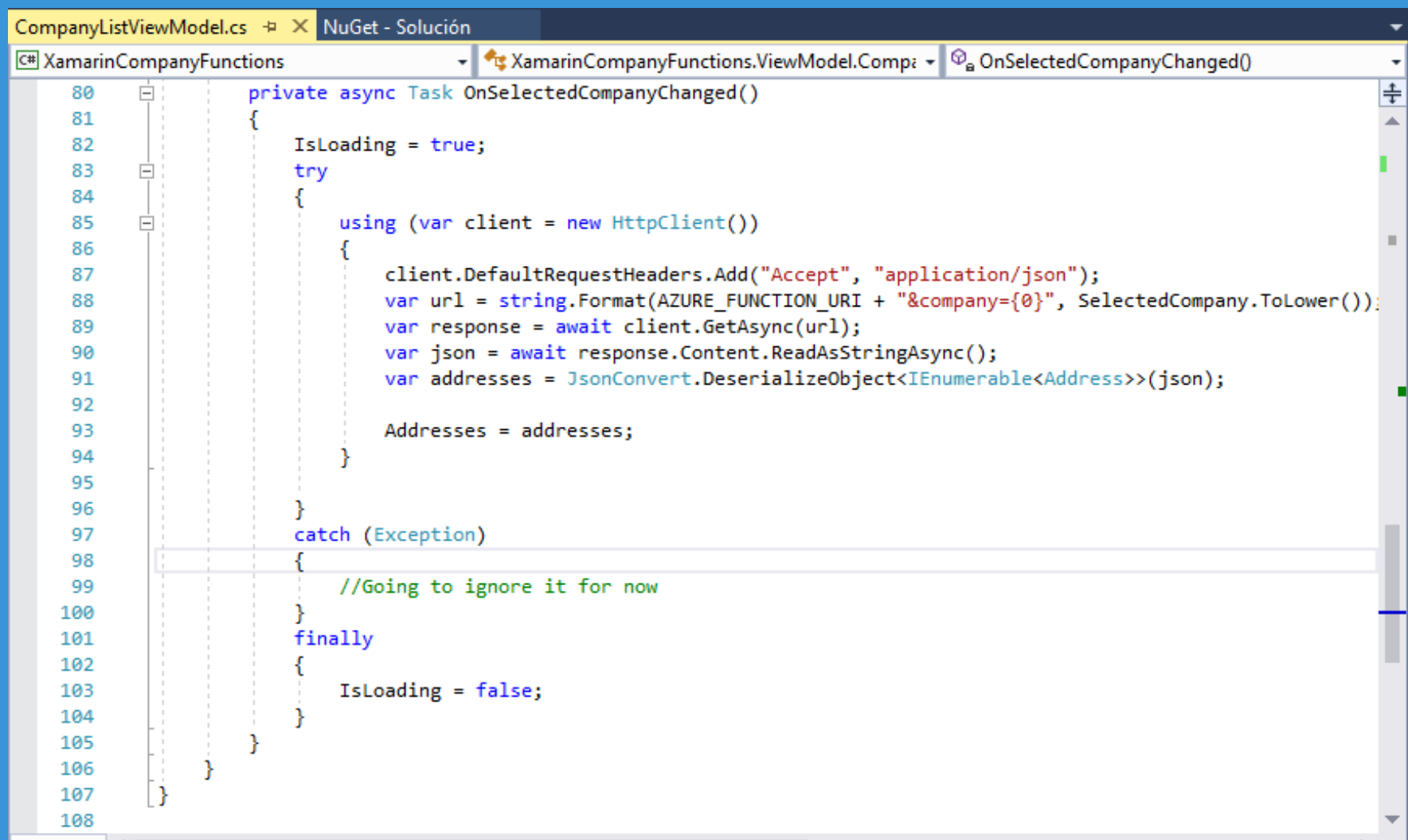
Archivos > AzureFunctions > Companies

```
1 <Addresses>
2   <Address name="Instituto Tecnológico de Celaya" company="TecNM">
3     <Street>A. Garcia Cubas</Street>
4     <Number>600</Number>
5     <City>Celaya</City>
6     <PostalCode>38300</PostalCode>
7     <Coords>20.5391485 -100.8191762</Coords>
8   </Address>
9   <Address name="Universidad de Guanajuato" company="UGto">
10    <Street>Pedro Lascurain de Retana</Street>
11    <Number>5</Number>
12    <City>Guanajuato</City>
13    <PostalCode>36000</PostalCode>
14    <Coords>21.0172925 -101.3233052</Coords>
15  </Address>
16  <Address name="Tecnológico de Queretaro" company="TecNM">
17    <Street>Av. Tecnológico</Street>
18    <Number>S/N</Number>
19    <City>Queretaro</City>
20    <PostalCode>76000</PostalCode>
21    <Coords>20.5943228 -100.4051776</Coords>
22  </Address>
23 </Addresses>
```

Demo

```
1 #r "System.Xml.Linq"
2 using System.Net;
3 using System.Xml.Linq;
4
5 public static async Task<HttpStatusCode> Run(HttpRequestMessage req, string inputFile, TraceWriter log)
6 {
7     log.Info("C# HTTP trigger function processed a request.\n" + inputFile);
8
9     string requestedCompany = req.GetQueryNameValuePairs()
10         .FirstOrDefault(q => q.Key.ToLower() == "company").Value;
11     log.Info("Compañía solicitada: " + requestedCompany);
12
13     var addresses = XDocument.Parse(inputFile);
14     var filteredAddresses = addresses.Root.Elements("Address")
15         .Where(e => (e.Attribute("company")?.Value.ToLower() == requestedCompany ||
16             string.IsNullOrEmpty(requestedCompany)));
17
18     //Creating anonymous type here to return.
19     var addressesResult = filteredAddresses.Select(a => new
20     {
21         Name = a.Attribute("name").Value,
22         Company = a.Attribute("company").Value,
23         Street = a.Element("Street").Value,
24         Number = a.Element("Number").Value,
25         City = a.Element("City").Value,
26         PostalCode = a.Element("PostalCode").Value,
27         Coords = a.Element("Coords").Value,
28     });
29
30     return req.CreateResponse(HttpStatusCode.OK, addressesResult);
31 }
```

Demo

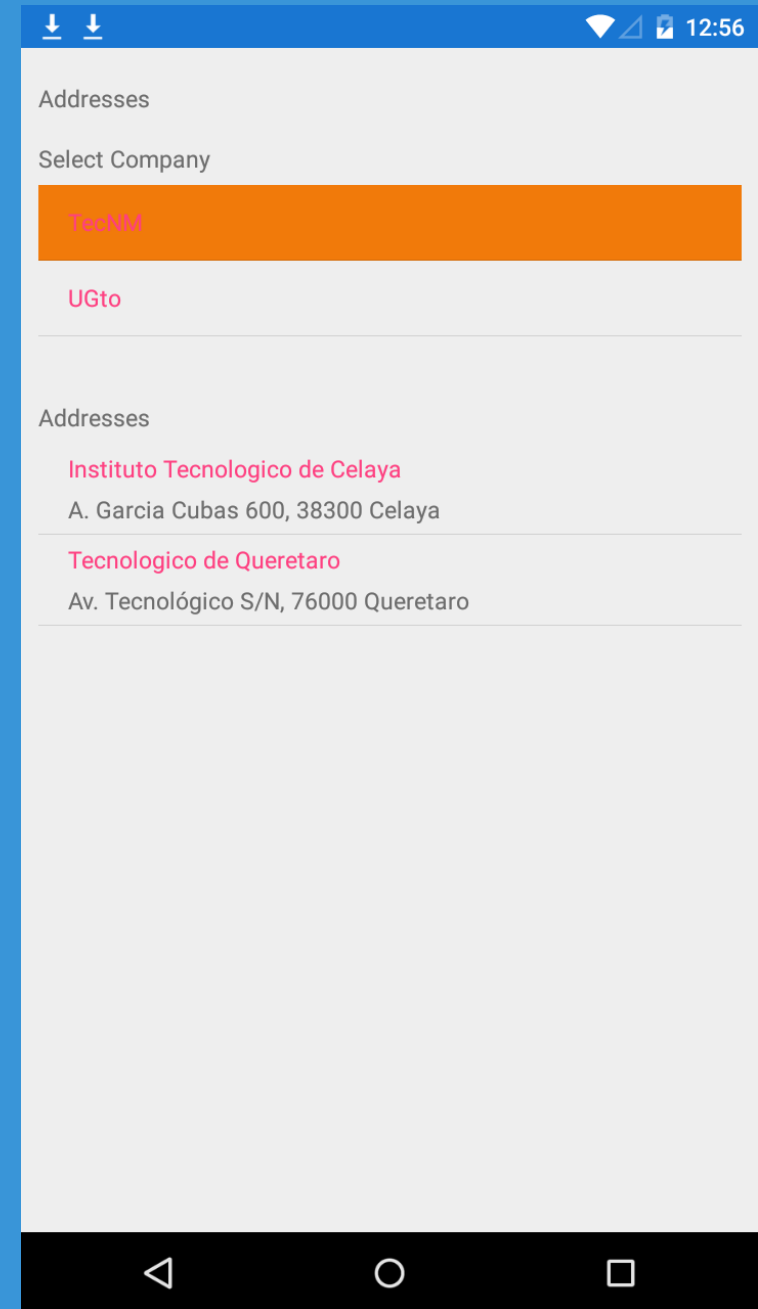
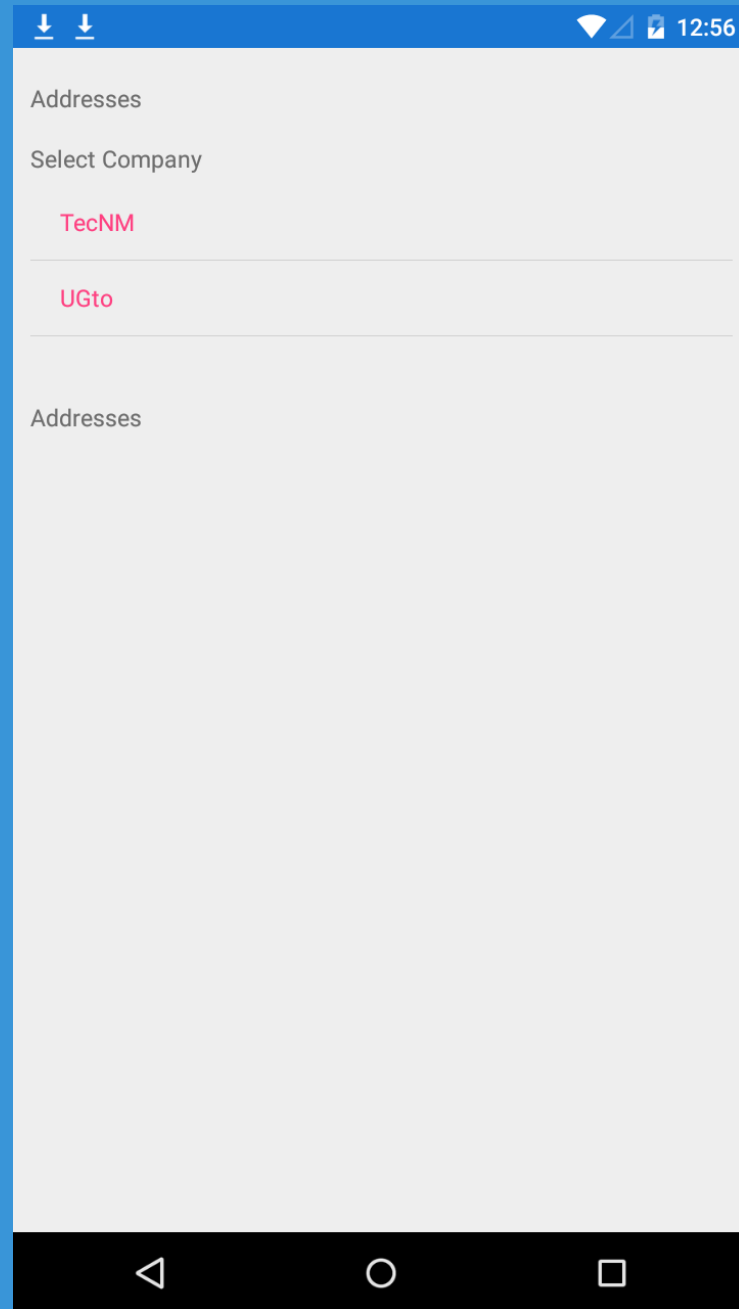


The screenshot shows a Visual Studio editor window with the following details:

- File Explorer:** Shows the project structure with 'CompanyListViewModel.cs' and 'NuGet - Solución'.
- Code Editor:** Displays the 'OnSelectedCompanyChanged()' method in 'XamarinCompanyFunctions.ViewModel.Comp:'. The code is as follows:

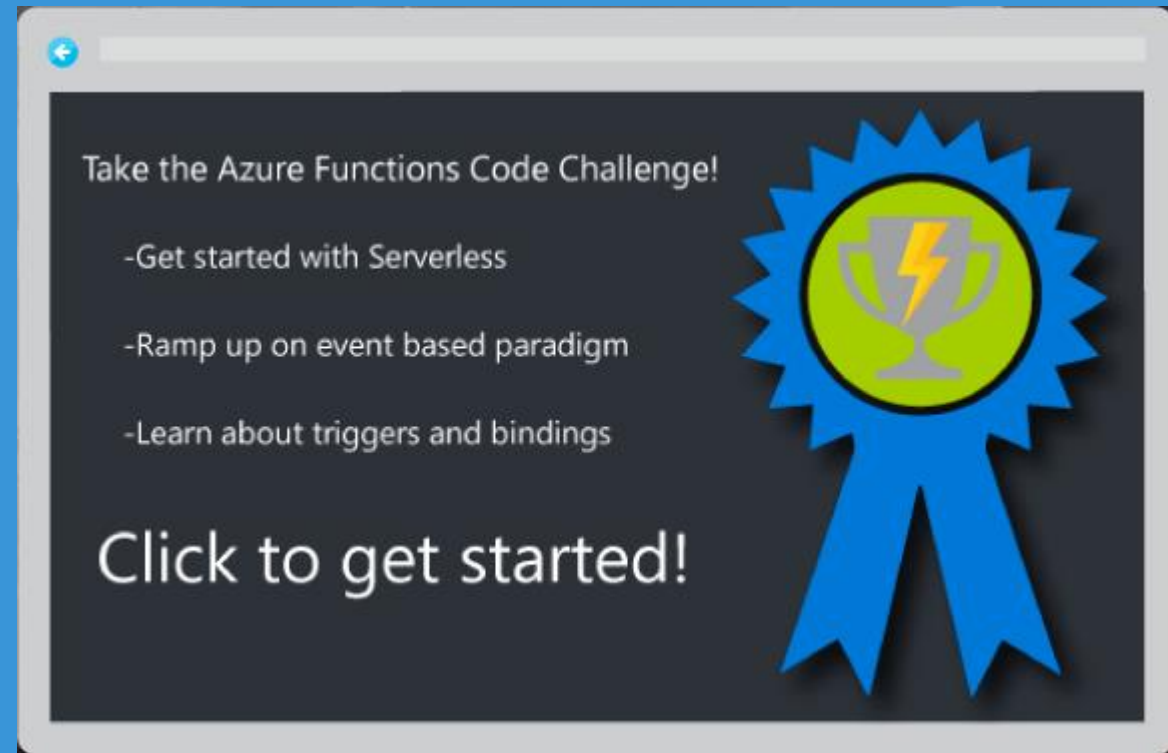
```
80 private async Task OnSelectedCompanyChanged()
81 {
82     IsLoading = true;
83     try
84     {
85         using (var client = new HttpClient())
86         {
87             client.DefaultRequestHeaders.Add("Accept", "application/json");
88             var url = string.Format(AZURE_FUNCTION_URI + "&company={0}", SelectedCompany.ToLower());
89             var response = await client.GetAsync(url);
90             var json = await response.Content.ReadAsStringAsync();
91             var addresses = JsonConvert.DeserializeObject<IEnumerable<Address>>(json);
92
93             Addresses = addresses;
94         }
95     }
96     catch (Exception)
97     {
98         //Going to ignore it for now
99     }
100 finally
101 {
102     IsLoading = false;
103 }
104 }
```
- Line Numbers:** The code is numbered from 80 to 108.
- UI Elements:** The editor includes a scrollbar on the right and a search icon in the top right corner.

Demo



¿Dudas?

<https://functionschallenge.azure.com>



Azure Functions Challenge

Test your coding skills and learn how to build solutions using Azure Functions at the same time. Earn badges for every challenge you complete and brag to your friends! You can code these challenges in the **FREE Azure Functions experience** or using your existing **Azure subscription**. Don't worry, Azure Functions has more than enough **free executions** for you to last the whole challenge experience! Also check out the new **Visual Studio Tools for Azure Functions**. Use it to build and debug your function locally and then publish or zip deploy to your test function. If you already have an **Azure subscription**, you can also remotely debug your functions.

¡Gracias por tu atención!

Luis Beltrán

Microsoft MVP

Xamarin Certified Mobile Developer

 @darkicebeam



<http://bit.ly/XamarinDiplomadoITC>



<http://bit.ly/MeetupCelayaNet>



<http://icebeamwp.blogspot.mx>



<https://github.com/icebeam7/XamarinCompanyFunctions>