Nº39 marzo 2018

CompartiMOSS

REVISTA ESPECIALIZADA EN TECNOLOGÍAS MICROSOFT

Entrevista Gastón Cruz

MSAL: Introducción y ejemplo de uso para llamar a Graph API

Sitios Web Estáticos en Azure Storage

Convert Classic root sites to Modern SharePoint

Contenido

Comparti MOSS

Staff

DIRECCIÓN GENERAL

- Gustavo Velez
- Juan Carlos Gonzalez
- Fabian Imaz
- Alberto Diaz

DISEÑO Y DIAGRAMACIÓN

- Santiago Porras Rodríguez

Contacte con nosotros

revista@compartimoss.com
gustavo@gavd.net
jcgonzalezmartin1978@hotmail.com
fabian@siderys.com.uy
adiazcan@hotmail.com

BLOGS
http://www.gavd.net
http://geeks.ms/blogs/jcgonzalez
http://blog.siderys.com
http://geeks.ms/blogs/adiazmartin

REDES SOCIALES
Facebook:
http://www.facebook.com/group.php?gid=128911147140492
LinkedIn:
http://www.linkedin.com/groups/CompartiMOSS-3776291
Twitter:
@CompartiMOSScom

03

Editorial

Con el inicio del nuevo año, vemos que Microsoft sigue en su ruta de “todo-en-la-nube”. Acercándose la nueva versión de Windows 10, está claro que la compañía apuesta cada vez más a esta ruta como demuestra la reciente noticia de que una gran parte del equipo de desarrollo de Windows ha sido movido al equipo de Azure. ¿Cuánto tiempo más antes de que llegue la tan mencionada versión en la nube de Windows?

Por el momento seguimos viendo también que, según las cifras de negocios presentadas por Microsoft hace unas cuantas semanas, Azure disminuye un poco en crecimiento, lo que es bastante comprensible después de años de doblar su rendimiento cada año, y que Office es todavía la gallina de los huevos de oro. Mejor dicho, nada nuevo bajo el sol. Y que, por supuesto Microsoft ha recuperado el puesto número uno como compañía más valiosa del mundo, algo que era impensable hace cinco años.

En CompartiMOSS seguimos, como es nuestra costumbre, mostrando lo más nuevo y más interesante técnicamente que nos ofrece Microsoft. Tanto los autores como los editores les ofrecemos un nuevo numero de la revista, que es de todos, y les deseamos que la disfruten tanto leyéndola y aplicando las ideas presentadas, como nosotros, los autores y responsables del magazín, disfrutamos creándola.

El Equipo Editorial de CompartiMOSS

4

Guardar firma desde la entrada de escritura manual de PowerApps en una lista de SharePoint

Creando un conector genérico con Flow para guardar un archivo

Vamos a crear un conector genérico que luego podremos usar en PowerApps para subir la imagen. Para ello iremos a nuestro sitio de Flow y crearemos un flujo desde cero, a la hora de buscar conectores y desencadenadores seleccionaremos “request/response” (traducido a Flow en español como “solicitud”).

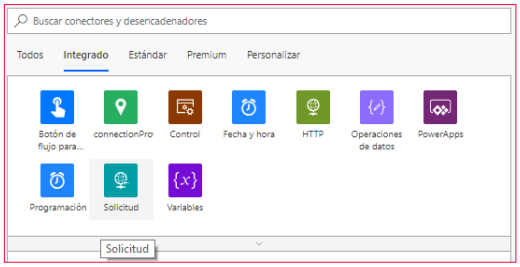


Imagen 1.- Conector de solicitud.

La url que pide para el post la dejaremos, ya veremos que se rellena automáticamente, vamos primero a agregar otro paso más a nuestro Flow, de las acciones de operaciones de datos vamos a quedarnos con “compose” (traducido como “Redactar”).

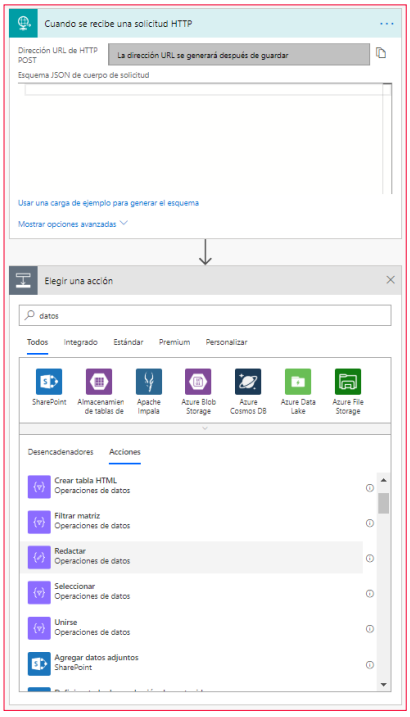


Imagen 2.- Componer la llamada.

A continuación, vamos a configurar la entrada de esta acción con la expresión Trigger() la cual si nos fijamos en su descripción permite derivar la salida de un desencadenador en tiempo de ejecución, es decir, el desencadenador anterior “cuando se recibe una solicitud HHTP”.

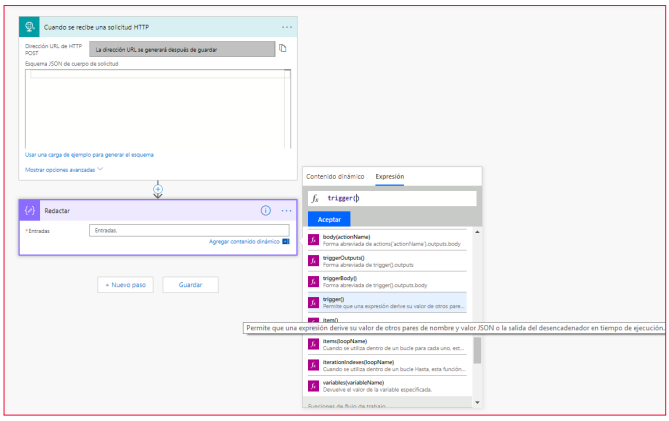


Imagen 3.- Configurar el trigger.

“En este artículo se describirá como desde una aplicación móvil de PowerApps utilizando la entrada”

Dejamos la configuración de esta manera, indicamos un nombre al flujo y al pulsar sobre guardarle veremos que la dirección url que antes hemos dejado vacía se ha rellenado

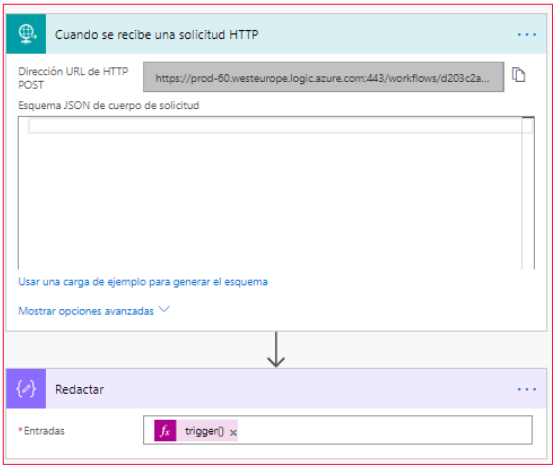


Imagen 4.- Url de solicitud.

Vamos a probar esta url para ver más claro que y como necesitamos enviarle al flujo todo lo necesario para que

se guarde una imagen, que será el nombre de la imagen y la imagen en sí. Para verlo vamos a usar la aplicación postman ("https://www.getpostman.com"):

- Indicamos la Url anterior como Url de la solicitud http.
- En la pestaña body de postman indicamos que la petición será form-data.
- Como key file seleccionamos una imagen cualquiera de nuestro pc.
- En la url agregamos el parámetro para el nombre de imagen.

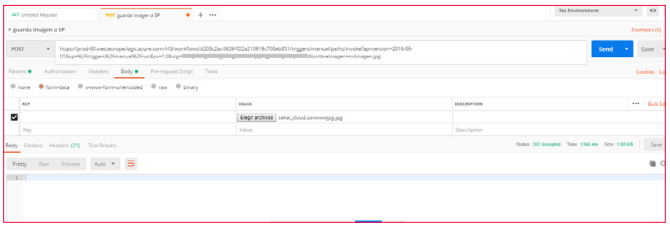
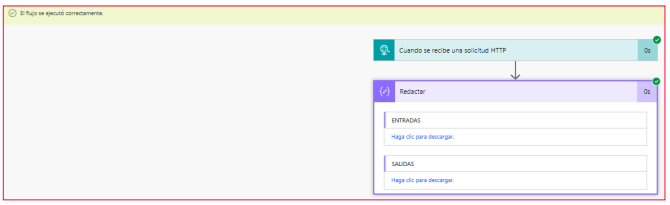
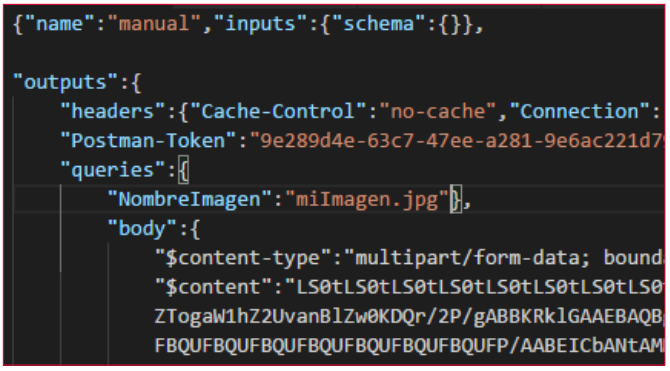


Imagen 5.- Testeando las peticiones con postman.

Después de enviar la petición de postman veremos en el historial de ejecuciones de nuestro flujo el resultado, donde en el JSON resultante, dentro de "outputs" tenemos "queries" con el nombre de la imagen que pasamos como parámetro y el "$content" dentro de "body" con la imagen en base64





```
{"name":"manual","inputs":{"schema":{}},

"outputs":{
    "headers":{"Cache-Control":"no-cache","Connection":
    "Postman-Token":"9e289d4e-63c7-47ee-a281-9e6ac221d7
    "queries":{
        "NombreImagen":"miImagen.jpg"},
        "body":{
            "$content-type":"multipart/form-data; bound
            "$content":"LS0tLS0tLS0tLS0tLS0tLS0tLS0
            ZTogaW1hZ2UvanBlZw0KDQr/2P/gABBKRklGAAEBAQB
            FBQUFBQUFBQUFBQUFBQUFBQUFP/AABEICbANtAM
```

Imagen 6.- Ejemplo de respuesta.

Vamos a editar nuestro flujo y coger primero el nombre que tendrá la imagen configurando el paso de redactar:

```
trigger()['outputs']['queries']['NombreImagen']
```

"el ejemplo vamos a partir de una aplicación para los repartidores de paquetes de una empresa"

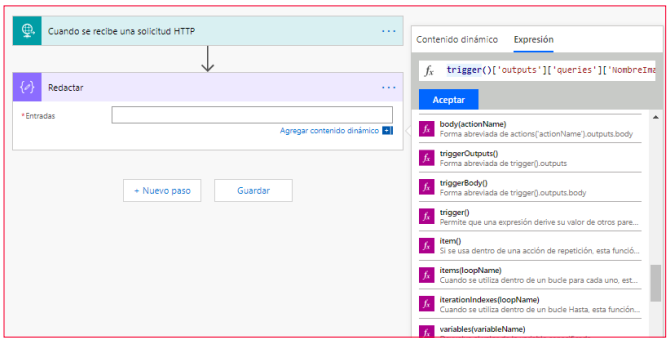


Imagen 7.- Configuración de compose.

Para obtener la imagen en sí, tendremos que añadir otro paso de "redactar". En este caso lo que queremos es el contenido en base64 de la imagen que contiene el body del resultado de la petición y que, como podemos ver en el JSON anterior, es de tipo "multipart/form-data" así que usaremos la expresión TriggermultipartBody().

Como posteriormente queremos que esta imagen se quede guardada en SharePoint aquí será necesario transformar esa imagen en base64 a binario y subir la imagen, vamos a verlo en la imagen como queda.

```
base64ToBinary(triggerMultipartBody(0)['$content'])
```

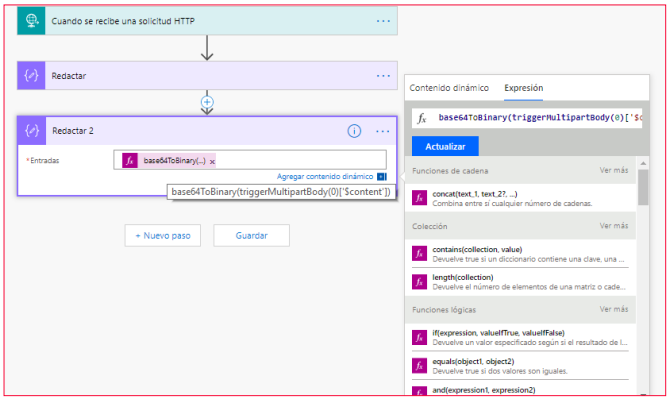


Imagen 8.- Obtención de la imagen.

Ya solo nos quedaría la parte de guardar en una biblioteca de SharePoint vamos entonces a agregar un paso más a nuestro flujo para ello, los pasos anteriores les cambié el nombre para que resulte más fácil de identificar

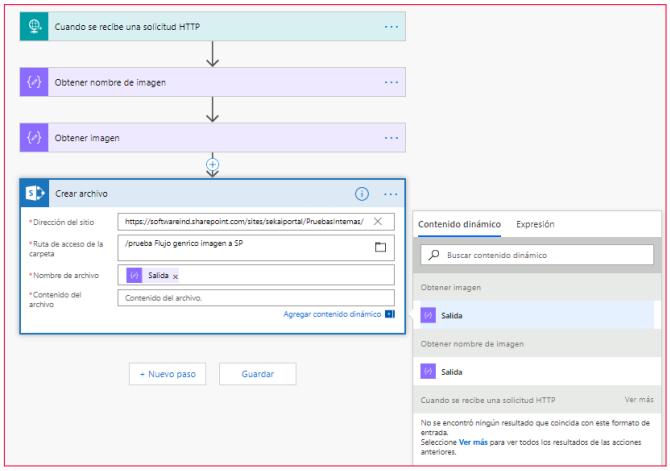


Imagen 9.- Guardar imagen.

Una vez guardado, si todo ha ido bien podremos probarlo llamándolo otra vez desde postman y visualizar el resultado en el sitio de SharePoint.

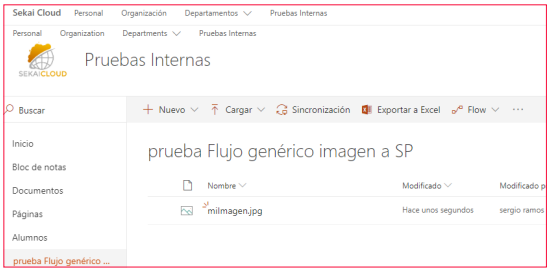


Imagen 10.- Resultado obtenido.

Guardar imagen desde la entrada de escritura manual de PowerApps

Vamos a ver como adjuntar una firma manual insertada desde una aplicación de PowerApps, para el ejemplo vamos a partir de una aplicación para los repartidores de paquetes de una empresa en la que firme el destinatario y los datos de la entrega con su firma se queden reflejados en una lista, vamos a ello, lo primero la lista de ejemplo con el destinatario dirección, etc. y un número de seguimiento del paquete.

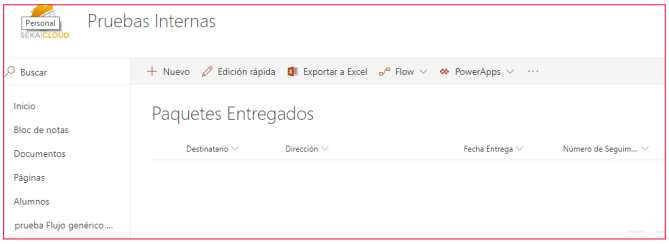
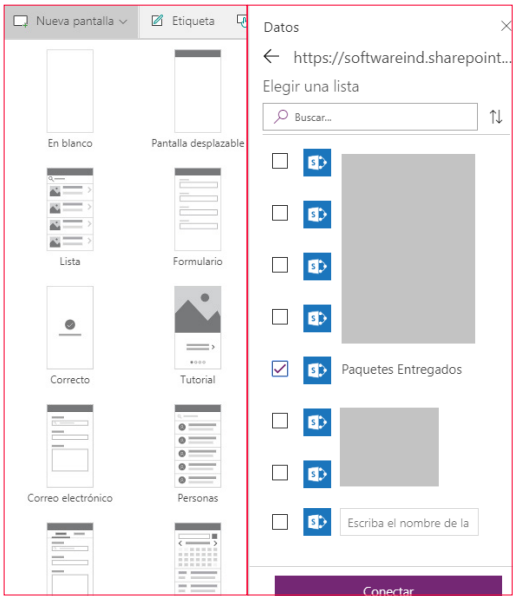
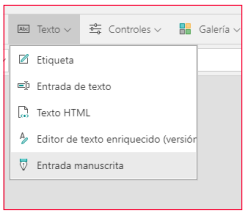


Imagen 11.- Creación de lista.

A partir de una aplicación PowerApps de lienzo en blanco, nos conectaremos a los datos de la lista, agregaremos una pantalla de formulario y haremos una conexión a la lista. Al formulario también habrá que cambiarle la propiedad de modo predeterminado a “nuevo”, también elimine el campo título que no tiene mucho sentido y agregamos el control de entrada para la firma.





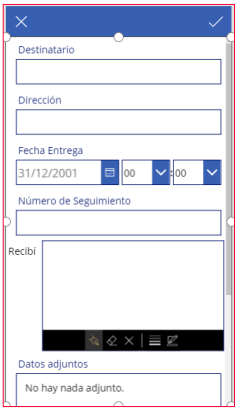


Imagen 12.- PowerApp creada.

Hasta aquí bien, el problema es que si intentamos guardar la imagen que contiene el control veremos algo del estilo “blob:https://create.PowerApps.com/……” y nos informará que eso no existe, es aquí donde haremos uso del workflow creado anteriormente. Para lo que queremos hacer, generaremos una plantilla para la llamada haciendo uso de la página http://specgen.apistudio.io/ con la url de la petición del post del http (la url que obtuvimos en el Flow).

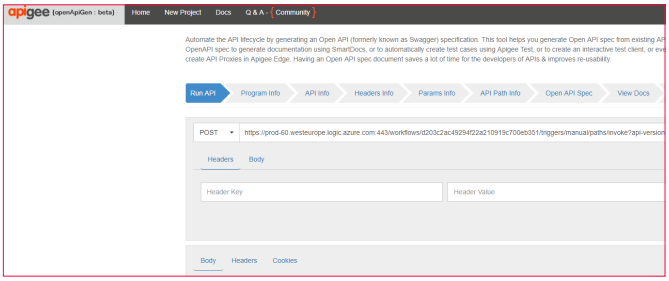


Imagen 13.- Generar llamada.

El paso siguiente de “program info” permite indicar datos de contacto, el título de la API, etc. Cambiamos el título a SubirArchivo, el e-mail de contacto y el puerto para la API. Dejamos el resto de los datos con los valores por defecto.

En el paso de “API Info” configuramos el parámetor API Base Path para que aparezca el valor “/workflows” e indicamos un nombre en “Operation id”.

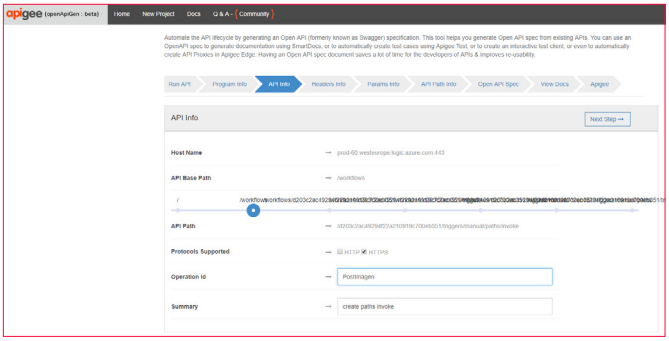


Imagen 14.- Paso “API Info”.

Los siguientes pasos los dejaremos como están hasta lle-

gar al paso “Open API Spec” que nos permitirá descargar el JSON Que se ha formado. Este JSON puede contener valores “null” por lo que será necesario editarlo para realizar las correcciones necesarias que nos eviten errores posteriormente.

```
"post": {
  "summary": "create paths invoke",
  "operationId": "PostImagen",
  "produces": [
    null
  ],
  "responses": {
    "202": {
      "description": "Accepted"
    }
  },
  "parameters": [
    {
      "name": "content-type",
      "in": "header",
      "description": "",
```

Imagen 15.- Arreglar el código generado.

En el json en la sección “parameters” podemos ver la definición de los parámetros de la petición http del workflow, a estos les tenemos que agregar los valores default que ya tenemos, y eliminar todo el parámetro content-type, se quedarían así

```
  "name": "content-type",
  "in": "header",
  "description": "",
  "required": true,
  "type": "string"
},
{
  "name": "api-version",
  "in": "query",
  "type": "string",
  "default":"2016-06-01",
  "description": "",
  "required": false
},
{
  "name": "sp",
  "in": "query",
  "type": "string",
  "default":"/triggers/manual/run",
  "description": "",
  "required": false
},
{
  "name": "sv",
  "in": "query",
  "type": "string",
  "default":"1.0",
  "description": "",
  "required": false
},
{
  "name": "sig",
  "in": "query",
  "type": "string",
  "default":"
  "description": "",
  "required": false
},
```

Imagen 16.- Agregando propiedades al JSON.

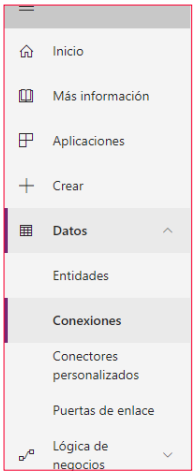
“la imagen de “tick” de la app en su OnSelect vamos a llamar a subirArchivo”

También agregamos nuestro contenido de la imagen con lo que se quedaría esta parte así

```
{
  "name": "NombreImagen",
  "in": "query",
  "type": "string",
  "description": "",
  "required": true
},
{
  "name": "Imagen",
  "in": "formData",
  "type": "file",
  "description": "",
  "required": true
}
```

Imagen 17.- Agregando propiedades.

Volvemos al entorno de creación de PowerApps PowerApps y hacemos clic en la parte de conexiones. Crearemos una nueva conexión en conectores personalizados, donde importaremos el archivo JSON que acabamos de modificar.



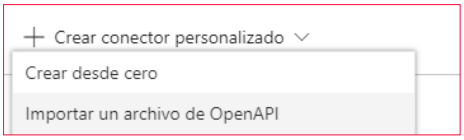


Imagen 18.- Creando una nueva conexión.

Tras esto en el paso 2 de autenticación lo dejaremos sin autenticación y en el 3 veremos el id de operación que será el que usemos más adelante.

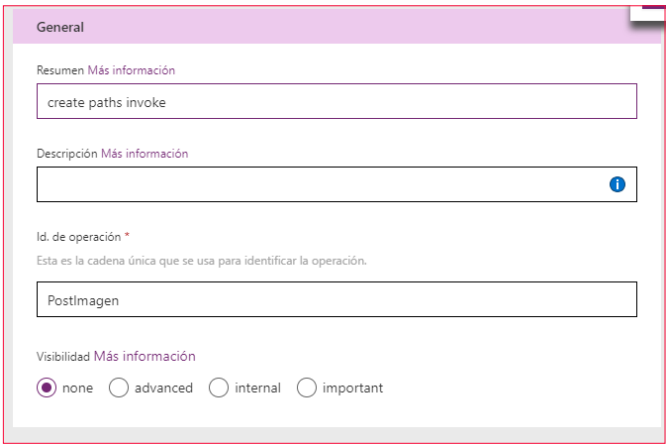


Imagen 19.- Configuración de la conexión.

Y también podemos ver los parámetros que se enviarán, tendremos que editar api-version, sp,sv,sig y ponerlos como de visibilidad interna para que se oculten al usuario.

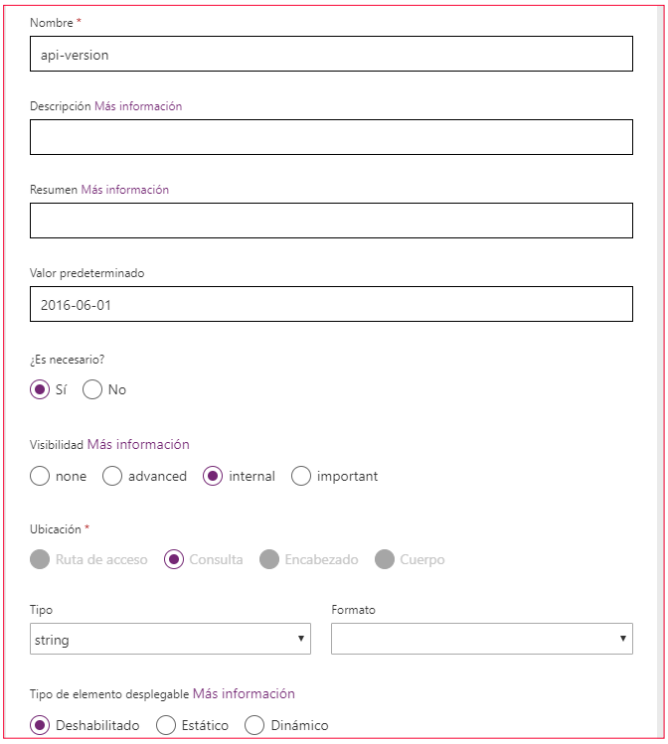


Imagen 20.- Parámetros adicionales en la configuración de la conexión.

Tras esto ya podremos guardarlo, y agregarlo como conexión

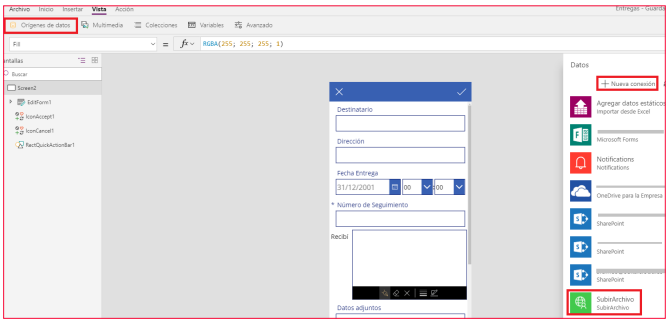


Imagen 21.- Conexión configurada y disponible para su uso.

En la PowerApp creada, el control de entrada de escritura que se ha agregado se llama PenInput3 y tiene la propiedad Image. En la imagen de "tick" de la app en su OnSelect vamos a llamar a subirArchivo, tendrá de nombre el número de seguimiento y subiremos la imagen del recibí que dibujemos, esta llamada la voy a hacer directamente y antes del submit para no andar poniendo validaciones etc, pero no es la forma más correcta

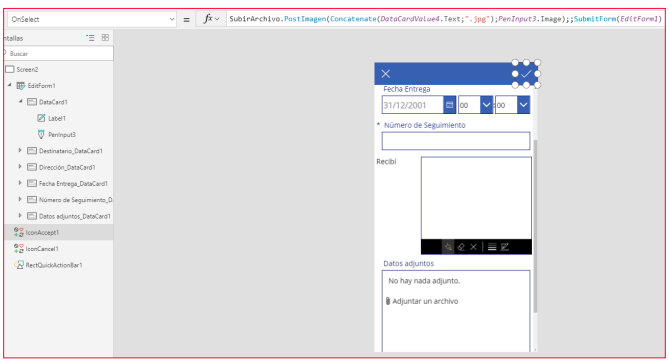
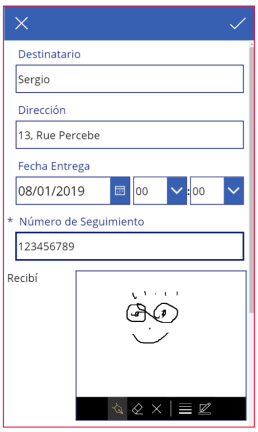
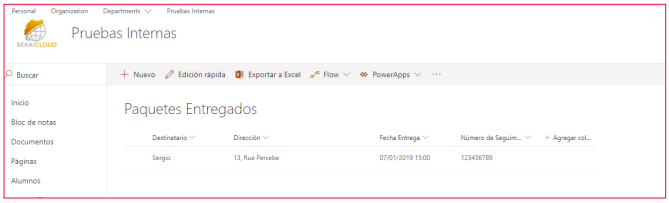


Imagen 22.- Configurando el control penInput.

Y la prueba:





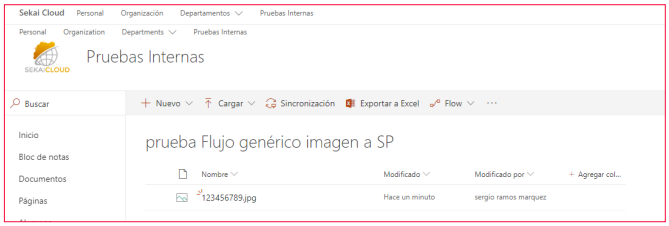


Imagen 23.- Prueba del Control penInput y de cómo la imagen generada se guarda en SharePoint.

Agregar imagen como adjunto con Flow

En este último paso vamos a ver como agregar como adjunto la imagen de firma que guardamos desde la aplicación creada anteriormente, en el teníamos una lista "paquetes entregado" y una biblioteca que almacena las firmas, vamos entonces a la biblioteca de firmas y crearemos un Flow desde cero para ella con el desencadenador "cuando se crea un archivo".

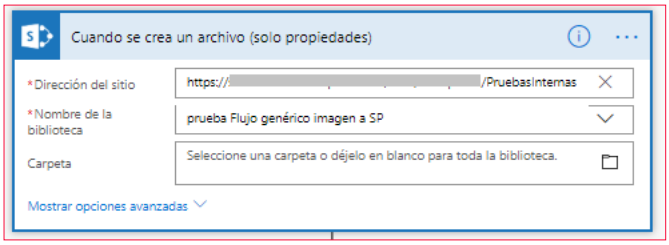


Imagen 24.- Trigger de inicio del Flow.

Obtendremos el contenido del archivo y crearé una variable para el nombre, o podría coger del mismo nombre del

archivo, pero ni se por qué de esa manera no me funcionaba bien

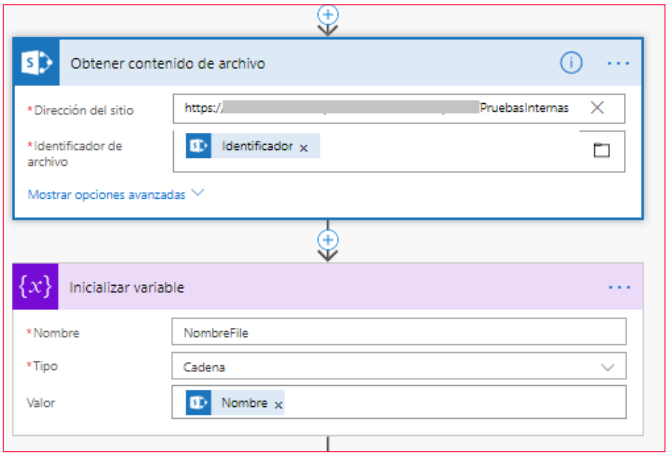


Imagen 25.- Procesado del contenido del archivo.

Obtendremos los elementos de la lista:

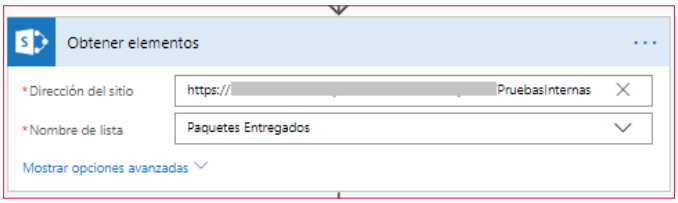


Imagen 27.- Obtención de los elementos de la lista.

Y en el último paso le damos el nombre de la imagen y le decimos que el contenido es el que hemos recuperado anteriormente

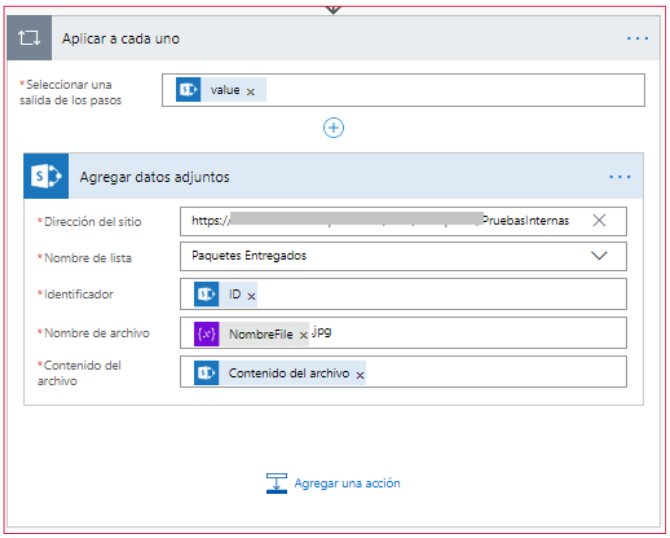


Imagen 26.- Paso final del Flow.

Conclusión

En este artículo hemos podido ver como interconectando los diferentes productos, Flow PowerApps y SharePoint podemos construir soluciones de negocio versátiles sin necesidad de código

SERGIO RAMOS
Sekai Cloud Co-Founder
sramos@sekaicloud.es
https://www.linkedin.com/in/sergioramosmarquez/
http://sekaicloud.es/

10

MSAL: Introducción y ejemplo de uso para llamar a Graph API

Si eres desarrollador en el ecosistema Cloud de Microsoft (AKA Azure), más pronto que tarde vas a necesitar consumir un servicio securizado con Azure Active Directory (AAD), por ejemplo, la ultra conocida Graph API, pero también otros servicios como la API REST de SharePoint, o incluso una API desarrollada por nosotros y protegida con Azure AD.

Sirva la siguiente imagen como referencia de arquitectura con varios clientes consumiendo APIs seguras en Azure AD, donde una de ellas podría ser perfectamente la Graph API:

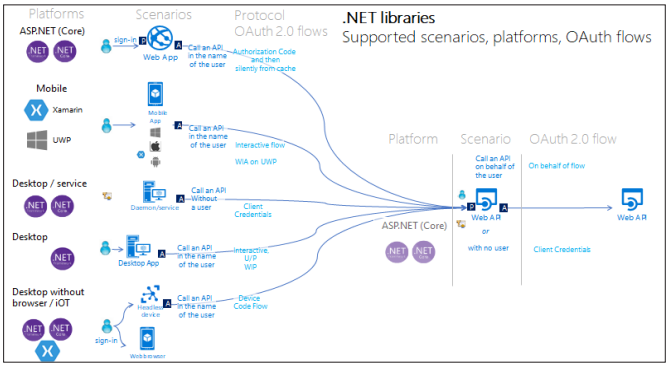


Imagen 1.- Arquitectura de referencia para consumir APIs de forma segura mediante Azure AD.

Cualquier servicio securizado en Azure AD, estará basado en los protocolos estándar OpenIdConnect y oAuth2, por lo que antes de ser consumido, deberemos obtener un Token válido para ese servicio.

El negociado de ese Token dependerá del escenario de consumo, pero siempre se hará siguiendo uno de los Flows definidos por OpenIdConnect y oAuth2. Dicho "Flow", no es más que un baile entre el cliente y Azure AD, que dará como resultado un Token válido para el servicio / API. Ese "baile", será diferente en función del tipo de cliente (ver imagen arriba): Si el cliente es una aplicación web, y se pretende consumir Graph API con permisos del usuario logado en la web app, seguiremos un Flow llamado "Authorization Code Flow". Si por el contrario tenemos una aplicación de consola y queremos consumir Graph con permisos de aplicación (sin intervención de ningún usuario), utilizaremos un Flow distinto: "Client credentials Flow".

A día de hoy, Azure AD expone 2 versiones para negociado de Tokens: endpoint v1 / v2. Podéis encontrar las principales diferencias en el siguiente enlace: https://docs.microsoft.com/en-us/azure/active-directory/develop/azure-ad-endpoint-comparison. Actualmente se recomienda utilizar el V2 siempre que sea posible, ya que cumple mejor con los estándares de OpenIdConnect y oAuth2 (entre otras ventajas). El endpoint v2, soporta los siguiente flows:

- Open ID Connect (Sign in) Flow
- OAuth 2.0 Auth code grant.
- OAuth 2.0 Implicit grant.
- Oauth 2.0 On-behalf-of.
- Oauth 2.0 Client Credentials.
- Oauth 2.0 Resource Owner password credentials grant.
- Refresh Token.
- Device Code Flow (personal accounts NOT supported).

Tenéis muy bien documentados todos estos flows en la documentación de Azure AD, que podéis encontrar en el siguiente enlace: https://docs.microsoft.com/en-us/azure/active-directory/develop/active-directory-v2-protocols

Por suerte, no es necesario conocer todos estos flows (además de otras cosas complejas como Signatures, Cryptography, etc), para poder negociar un Token con Azure AD. Es aquí donde aparece la librería MSAL (Microsoft Authentication Library). MSAL nos va a facilitar mucho la vida a la hora obtener un token, y nos va a abstraer bastante de la implementación específica de cada Flow.

> ***"más pronto que tarde vas a necesitar consumir un servicio securizado con Azure Active Directory"***

En este artículo nos vamos a centrar en MSAL.net, que es la versión .net, pero existen otras versiones de MSAL para otras tecnologías: msal.js, para aplicaciones JavaScript o NodeJS, además de versiones para IOS, Android e incluso Python.

Actualmente podemos instalar MSAL como un paquete más de NuGet.

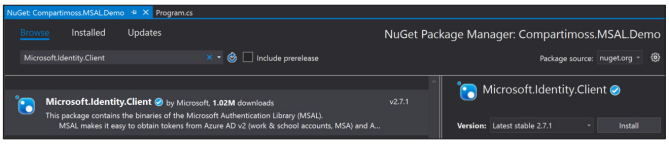


Imagen 2.- Paquete NuGet de MSAL.

Registrar App para consumir MS Graph API

Previamente al negociado de Token, debemos registrar una App en Azure AD, y configurarla para poder consumir la Graph API. En este ejemplo vamos a utilizar el Azure AD v2 endpoint, así que haremos uso de la opción del portal "App registrations (preview)".

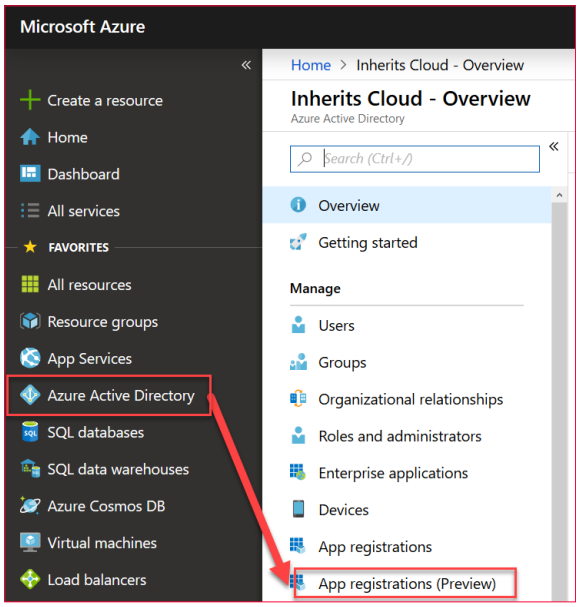


Imagen 3.- Acceso al registro de la App en Azure AD.

Como podemos observar, esta opción todavía está en "preview", sin embargo, se refiere a la experiencia de registro (UI y UX), es decir, que el endpoint V2 es perfectamente valido para producción. Una vez allí, crearemos una nueva app, simplemente dando un nombre para la App. Una vez registrada, necesitamos unos pasos extras para poder utilizar Graph desde nuestra App. Primero vamos a necesitar crear un Secret:

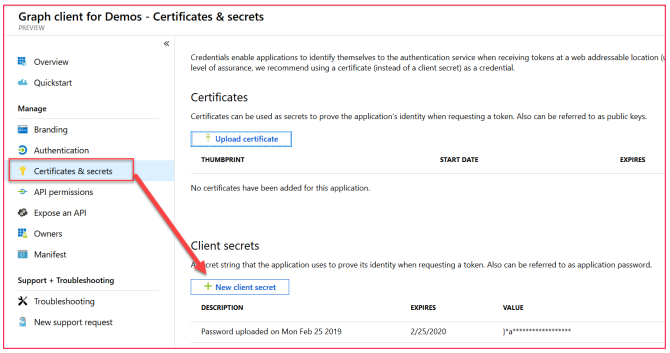


Imagen 4.- Generación del Client Secret.

Finalmente, configuramos los permisos que queremos que nuestra App tenga en Graph API. En nuestro ejemplo, nuestro cliente va a ser una aplicación de consola, y vamos a querer llamar a Graph con permisos de aplicación, para que no sea necesaria la intervención de un usuario (login). Un caso de uso bastante común sería esto mismo en una Azure Function que se ejecuta con un Timer (típico proceso de background). Siendo así, seleccionamos permisos de Aplicación:

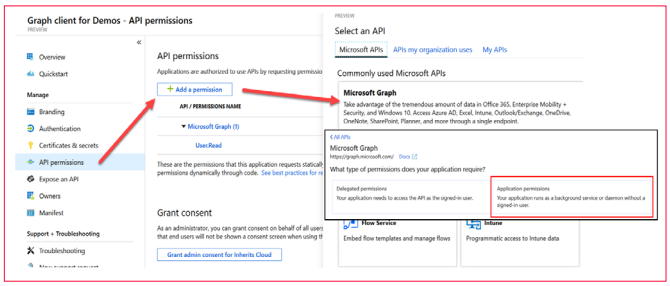


Imagen 5.- Añadiendo permisos a la aplicación.

Para nuestro ejemplo vamos a leer los grupos, así que solicitamos permisos para ello:

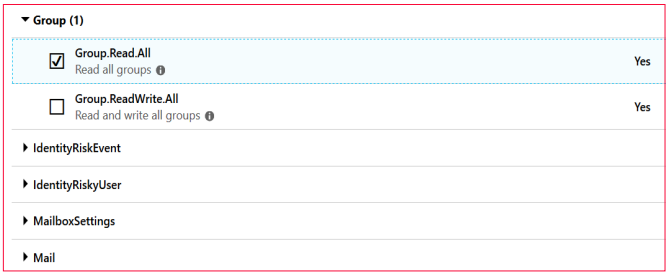


Imagen 6.- Permisos configurados.

"nos vamos a centrar en MSAL.net, que es la versión .net, pero existen otras versiones de MSAL"

Los permisos de aplicación siempre van a requerir que un administrador los "consienta" (AKA Admin consent). Para ello podríamos acceder a otro endpoint de Azure AD pasando ciertos parámetros, un usuario administrador haría el Login, y aceptaría los permisos requeridos. Sin embargo, el portal nos lo pone mucho más fácil con el siguiente botón:

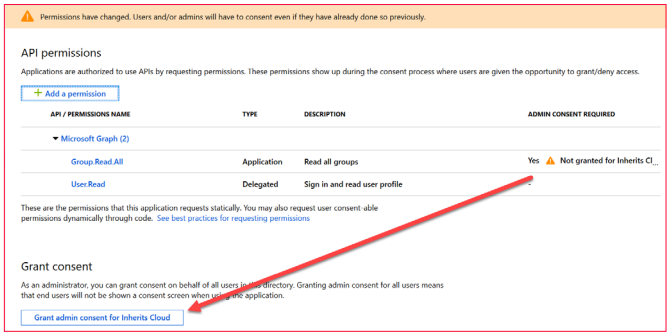


Imagen 7.- Garantizando el consentimiento requerido por la aplicación.

Obtención del token

Llegado a este punto, ya tenemos nuestra App registrada en Azure AD, y lista para usarla desde nuestro código y obtener un Token para Graph. Vamos ahora al código. En una aplicación de consola, podemos usar el siguiente snippet para obtener un token válido para Graph API con permisos de Aplicación:

```
class Program
{
    private const string TenantId = "cd8[redacted]3";
    private const string ClientId = "bbb1[redacted]2c2";
    private const string Secret = "}*aR?*@|$[redacted]k:-[^W";

    0 references
    static void Main(string[] args)
    {
        try
        {
            var credentials = new ClientCredential(Secret);

            var application = new ConfidentialClientApplication(
                ClientId,
                $"https://login.microsoftonline.com/{TenantId}",
                "https://daemon",
                credentials,
                null,
                new TokenCache());

            var authenticationResult = application.AcquireTokenForClientAsync(
                new[] { "https://graph.microsoft.com/.default" })
                .GetAwaiter().GetResult();

            Console.WriteLine(authenticationResult.AccessToken);
        }
        catch (Exception ex)
        {
            Console.WriteLine(ex);
        }

        Console.WriteLine("Done.");
        Console.ReadLine();
    }
}
```

Básicamente son solo 2 pasos: Primero creamos un objeto "ConfidentialClientApplication", que va a representar nuestra App registrada en Azure AD. Existen dos tipos de clientes OpenIdConnect / oAuth2:

- Confidential: Un cliente "confidencial" es aquel que, por su naturaleza, puede guardar una key/secret de manera segura, como por ejemplo una Azure Function, una app de consola (daemons en general), o una aplicación de web con código de servidor.
- Public: un cliente público no es capaz de almacenar una key de forma segura, como por ejemplo una aplicación browser-base (una SPA de JavaScript), o aplicaciones nativas, mobile, etc.

"con unas pocas líneas de código y sin tener que conocer las tripas de los diferentes flows de Azure AD"

Y así ha sido reflejado en la libraría MSAL:

- ConfidentialClientApplication
- PublicClientApplication

En el constructor de ConfidentialClientApplication tenemos:

1.– Client ID de nuestra App registrada en Azure AD.

2.– Authority: esto es la URL de login a nuestro tenant, y se compone tal y como tenemos en el código, usando nuestro Tenant ID.

3.– Redirect URI: no aplica a nuestro caso, ya que es una aplicación de consola (no una web app). Sin embargo, es necesario especificar cualquier valor que sea una URL HTTPS válida.

4.– ClientCrendetials usando el Secret obtenido al registrar la app en Azure AD.

5.– User Token Cache: null, puesto que obtenemos el token con permisos de Aplicación, sin usuario logado.

6.– Application Token Cache: MSAL tiene una cache de Tokens interna, para evitar llamadas innecesarias a Azure AD una vez obtenido el Token. En muchos escenarios, la TokenCache interna de MSAL no será suficiente, y habrá que extenderla (por ejemplo, para usar alguna DB o RedisCache). Hoy en día, MS proporciona un ejemplo basado en Http Session, que mejora la interna para escenarios web, pero que sigue sin ser óptimo.

Una vez tenemos el objeto ConfidentialClientApplication, sólo tenemos que llamar al método "AcquireTokenForClientAsync" pasando los scopes para los que solicitamos el Token. Para el caso de Graph, existen numerosos scopes: Groups.Read.All, Email.Send, etc. Sin embargo, cuando usamos permisos de Aplicación, debemos pasar el scope por defecto de Graph: https://graph.microsoft.com/.default, y de esta manera, la App tendrá los permisos registrados en Azure AD (que en nuestro caso era Groups.Read.All).

Llamando a Graph API para obtener grupos en la tenant

Una vez obtenido el token, basta con incluirlo en cualquier petición a Graph. En nuestro caso vamos a llamar al endpoint de groups, para obtener los primeros 5 grupos de la tenant:

```
var httpClient = new HttpClient();
httpClient.DefaultRequestHeaders.Authorization =
    new AuthenticationHeaderValue("Bearer", authenticationResult.AccessToken);

var response = httpClient.GetAsync("https://graph.microsoft.com/v1.0/groups?$select=Id,DisplayName&$top=5")
    .GetAwaiter()
    .GetResult();

var json = response.Content.ReadAsStringAsync().GetAwaiter().GetResult();

Console.WriteLine(json);
```

Como vemos, con unas pocas líneas de código y sin tener que conocer las tripas de los diferentes flows de Azure AD, hemos sido capaces de obtener un Token y hacer una llamada a Graph API.

LUIS MAÑEZ
SharePoint/Cloud Solutions Architect en ClearPeople LTD
@luismanez
https://medium.com/inherits-cloud

13

Análisis de Logs ULS usando ElasticStack (Beats, Logstash, Elasticsearch y Kibana)

El stack de Elastic sigue la siguiente estructura:

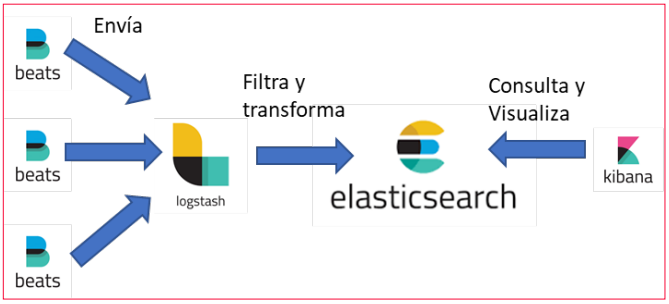


Imagen 1.- Arquitectura de ElasticStack.

Kibana actúa como interfaz gráfica para todo el sistema, donde se visualizan los datos y se realizan las agrupaciones necesarias.

"incluye los logs de SharePoint en sus distintas versiones"

Elasticsearch es el repositorio de datos transformados (sin ser una base de datos), cuyos índices son la principal base para la representación de datos

Logstash es donde se realiza la agrupación, filtrado y transformación de los datos. Esto es importante porque es quien consumirá más recursos.

Beats. Es un pequeño agente que instalamos en los servidores y que nos permite el envío de datos recolectados en nuestro servidor a un procesador (Logstash) o directamente al repositorio (Elasticsearch. Cabe destacar que existen distintos tipos de Beats, según nuestra necesidad: FileBeat (para el envío de logs/ficheros), WinLogBeat (envío de eventos en un servidor Windows), MetricBeat (envío de métricas), etc.

Sobre la arquitectura:

- Los distintos Beats se instalan sobre los servidores/aplicación a analizar. En este caso particular, una granja de SharePoint 2013, pero puede hacerse sobre aplicaciones web (apache, IIS, etc).
- Logstash puede instalarse en un servidor como rol dedicado, dependiendo de la cantidad de beats que vaya a recibir. A mayor número de beats/servidores, más poder va a requerir.
- ElasticSearch puede instalarse como servidor StandAlone, compartiendo recursos con Logstash aunque puede escalarse hacia una arquitectura más robusta: un clúster con un par de nodos.
- Kibana funciona como un servidor web, por lo que puede instalarse como servidor separado.

Como arquitectura, se ha instalado Beats en un servidor de Desarrollo, enviando logs a un servidor standAlone que contiene el resto de la arquitectura (Logstash, Elasticsearch & Kibana). La configuración (fichero filebeats.yml) es la siguiente:

```
filebeat.prospectors:

- type: log
  enabled: true
  paths:
    - C:\Elastic\Data\*.log

output.logstash:
  hosts: ["DAPERE759VM:5044"]
```

En el servidor con el resto de las funciones, hay 2 elementos importantes a "retocar", ambos en la carpeta de Logstash:

En *%Logstash_home%/Config*, creamos un fichero llamado Sharepoint_ULS.conf, con el siguiente contenido:

```
input {
        beats {
     port => "5044"
   }
}
filter{
        grok {
                match => ["message", "(?<sptimestamp>%{MONTHNUM}/%{MONTHDAY}/%{YEAR} %{HOUR}:%{MINUTE}:%{SECOND}\*?*)%{SPACE}%{PROG:sp_process}%{SPACE}\(%{WORD:sp_pid}\)%{SPACE}%{WORD:sp_tid}%{SPACE}+%{CATEGORY:sp_area}%{SPACE}+%{CATEGORY:sp_category}%{SPACE}%{WORD:sp_eventid}%{SPACE}%{WORD:severity}%{SPACE}%{GREEDYDATA:sp_eventmessage}"]
        }
        date {
                match => ["sptimestamp", "MM/dd/YYYY HH:mm:ss.SSS"]
                target => "sptimestamp"
        }
}
output {
 stdout {
         codec => json
```

```
        }
elasticsearch {
        hosts => ["127.0.0.1:9200"]
        index => "uls.%{+YYYY.MM.dd}"
}
}
```

Esto es tal vez el mayor quebradero de cabeza, puesto que es donde hacemos la transformación de los datos. Así que lo despiezo:

- Input { beats { port => "5044"} } Esto le dice a logstash de dónde va a obtener los datos, en este caso serán Beats que vendrán a través del puerto 5044. Estos BEATS vienen como un JSON.
- Filter { Esto es el inicio de las transformaciones que hará el Logstash. Tiene dos partes:
  - el GROK que es una estructura basada en RegularExpressions. Este filtro es lo que nos permite separar el contenido de cada línea e identificar los distintos campos que vienen informados en el log ULS: Fecha/hora, Proceso, PID, Area, Categoria, Severidad, Mensaje, etc.
  - La segunda parte es la transformación del campo fecha. Es importante porque la configuración predeterminada de Logstash no entiende que en el JSON venga informado un campo numérico/fecha con un cero previo: Prefiere 1/15/2018 en vez de 01/15/2018.
- output. Esta sección indica los distintos tipos de salida y pueden ser por pantalla (en cuyo caso le digo que quiero que lo formatee como un JSON), a una consola o a un repositorio (ElasticSearch). En este último caso hay que indicar el servidor al que debe ir y cómo se van a nombrar los índices.

"Una vez se levanta filebeats comenzará a recoger información de los ficheros de log"

El segundo fichero por modificar se llama grok-patterns y está ubicado en %logstash_home%\vendor\bundle\jruby\2.3.0\gems\logstash-patterns-core-4.1.2\patterns. Importante: Este fichero es parte del sistema, así que algo de cuidado! Contiene básicamente los distintos tipos de RegularExpressions que se utilizan para definir/acotar campos. Agregamos una línea con la definición de CATEGORY al final:

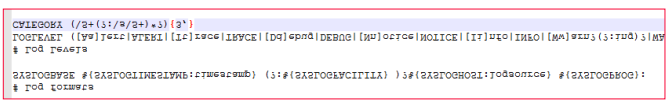


Imagen 2.- Fichero grok-patterns modificado.

Cabe destacar que los instaladores de Elastic (cualquiera de los productos del stack) suelen descargarse como un .zip pero alguno de ellos incluye un .msi para desplegar con mayor facilidad. En todo caso, posteriormente los ejecutables pueden arrancarse como servicios (Filebeat y ElasticSearch por ejemplo). También podemos arrancarlos de forma manual. En todo caso es importante arrancarlos de forma ordenada:

1.– Elasticsearch.

2.– Logstash (fundamentalmente, porque tarda algo de tiempo en arrancar).

3.– Kibana.

4.– FileBeats.

El orden es importante ya que las aplicaciones deben estar levantadas en sus respectivos puertos para que la siguiente aplicación pueda conectar: Si Logstash no está levantado, FileBeats nos da error. Para este demo, ElasticSearch se levanta como servicio y Kibana, Logstash y filebeats como comandos de consola (y así apreciar las salidas y confirmar que funciona correctamente)

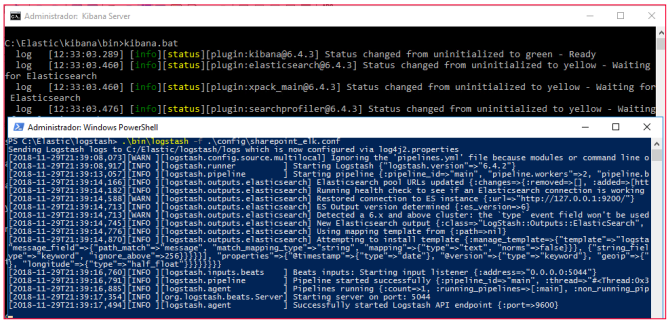


Imagen 3.- Kibana y Logstash levantados en consola.

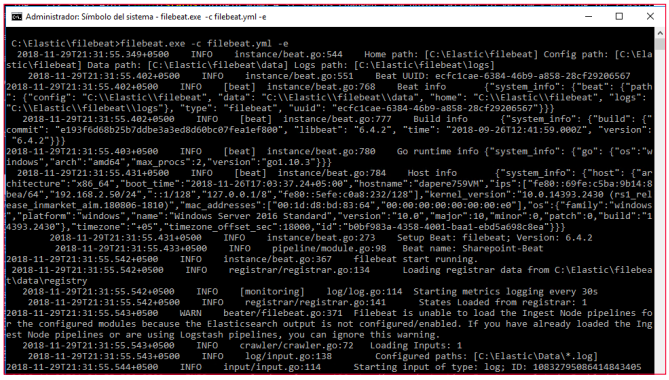


Imagen 4- FileBeat levantado en consola.

Una vez se levanta filebeats comenzará a recoger información de los ficheros de log y enviarlas a Logstash para su procesamiento y de allí pasarán a ElasticSearch. ¿Cómo confirmar que funciona correctamente? En un navegador, abrimos la url de kibana y en "Management > Index Management" podemos ver si nuestro índice está creado y la cantidad de elementos que tiene:

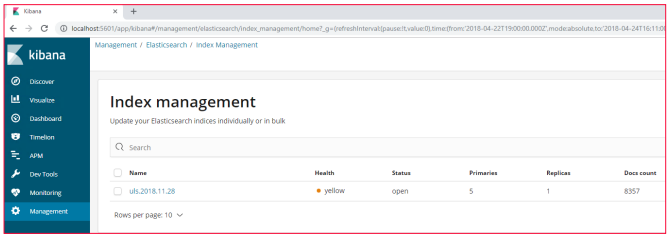


Imagen 5.- Confirmación de la creación del índice.

Para saber si nuestro contenido tiene los campos que nos interesan, en la sección “Dev tools” podemos ejecutar una consulta de ejemplo:

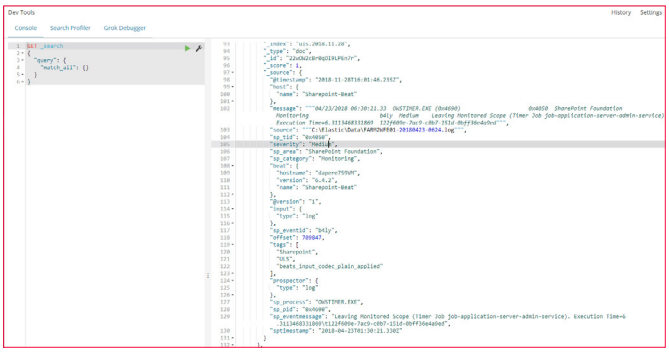


Imagen 6.- Consulta al índice y visualización de campos.

La ventaja de utilizar la salida por consola es apreciar errores antes de verlos en el índice. En la siguiente imagen (7: Error en Logstash al convertir datos) se puede apreciar un error en la salida de logstash porque el filtro GROK no es capaz de convertir los datos de forma correcta (_grokparsefailure). Cabe destacar que aunque se presente este tipo de error, se genera un índice dentro de ElasticSearch que contiene el timeStamp (fecha/hora en que se envía el JSON) y como campo Message, la línea entera del log.

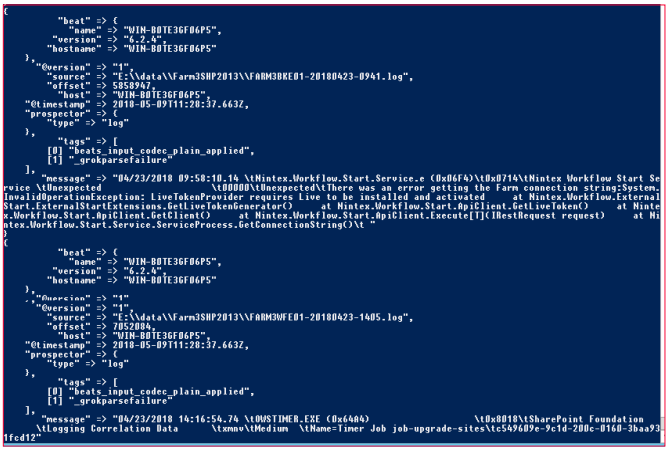


Imagen 7.- Error en Logstash al convertir datos.

Una vez llegados a este punto, podemos comenzar con la exploración del contenido de los logs. Para ello, en “Discover” comenzamos por definir un patrón de índice, desde “Management > Kibana > Create Index Pattern”. Lo primero que nos solicita es el patrón del nombre. Si tenemos algún índice creado nos coincidirá y podremos partir desde allí:

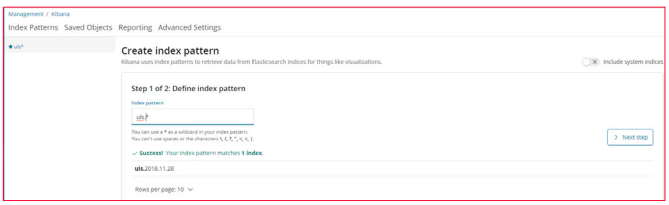


Imagen 8.- Crear un patrón de índice.

“antes de generar los gráficos es importante tener una idea de lo que queremos comparar.”

Al presionar el botón “Next step”, nos lleva a una nueva ventana donde debemos seleccionar el campo de fecha asociado, que nos permita diferenciar elementos en el tiempo.

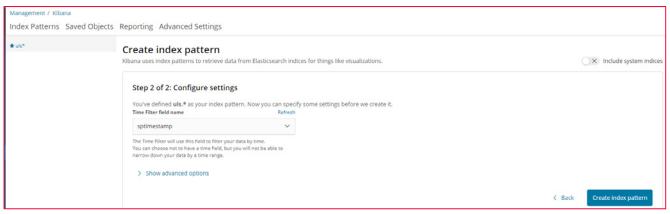


Imagen 9.- Configurar campos en el índice.

Una vez creado nuestro patrón y asociado al índice, podemos comenzar a realizar búsquedas. Esto es clave para la creación de gráficos y Dashboards. En este caso generamos una búsqueda llamada “Process” y agregamos como campos sp_timestamp, sp_category, sp_area, sp_process y severity.

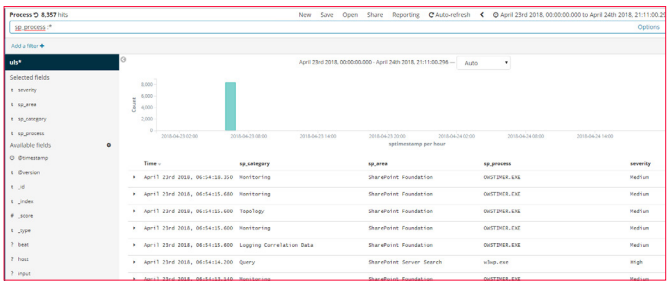


Imagen 10.- Creación de búsqueda y campos a visualizar.

Una vez hecho y guardado, podemos seguir con crear paneles de visualización. Para ello tenemos varios tipos: área, mapa de calor, barras verticales/horizontales, datatable, línea, y dos que particularmente son muy útiles: tarta (pie) y Nube de etiquetas (tag cloud). Una vez seleccionado el tipo, debemos decir cuál es la base de nuestra gráfica. Puede ser una consulta que hayamos hecho o una nueva. Si seleccionamos la consulta que hemos guardado previamente, en el titulo aparece “linked to saved search Process”. Un aspecto importante es lo que Kibana llama “buckets” que pueden ser Split Series (nos permite superponer datos) o Split Charts (que nos permite gráficas separadas)

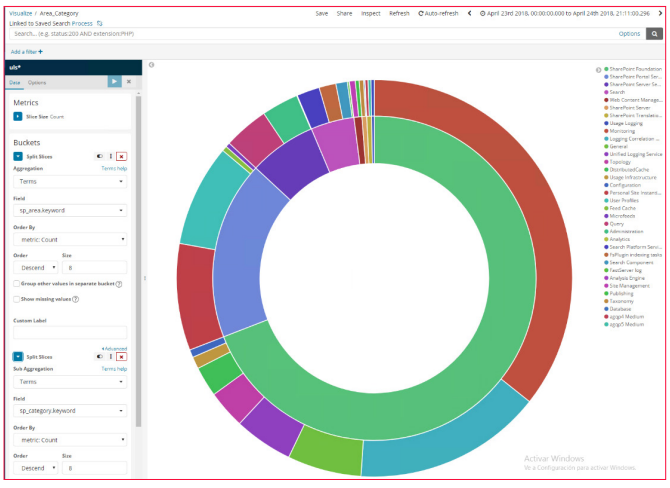


Imagen 11.- SP_Category vs SP_Area.

Aquí un truco importante: antes de generar los gráficos

es importante tener una idea de lo que queremos comparar. Por ejemplo, para cada proceso (w3wp, owstimer, noderunner, etc), saber los tipos (Categorías) de mensajes que aparecen y la cantidad de veces que aparecen. Esto nos puede dar una idea de las cosas que se hacen o dónde debemos poner nuestro foco.

Por último, una vez generadas varias visualizaciones podemos crear un dashboard, que nos proporcione la información según necesitemos ver los datos:

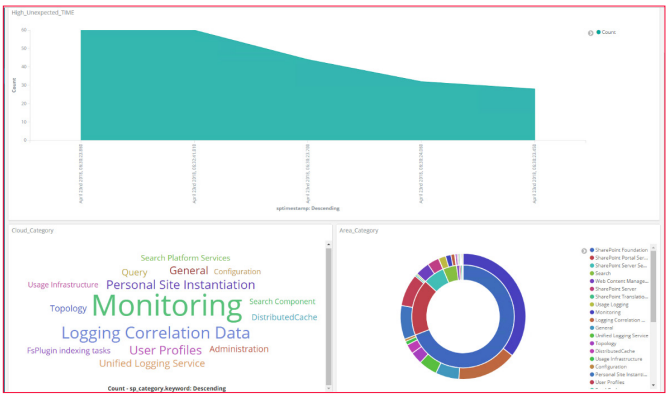


Imagen 12.- Dashboard para el análisis.

Conclusión

El análisis de logs, y en particular de IIS y SharePoint nos permite en primera instancia revisar los puntos que peor se comportan en nuestra plataforma y así dedicar esfuerzos en su mejora. Igualmente podemos identificar cuellos de botella y los elementos que presentan mayor uso, de forma que podamos reforzarnos o escalar la plataforma en dichos puntos.

Si bien es cierto que hoy en día con topologías híbridas o basadas en la nube disponemos de una variedad de opciones desde enviar logs a una BBDD y extraer los datos o usar eventos que puedan disparar flujos de escalado, no podemos descartar el uso de herramientas de análisis y representación vía Cuadros de mando como Power BI, Kibana, splunk, grafana e incluso, comenzar a pensar en análisis de series (para verificar cada cuanto se produce un error) e identificar patrones (errores en indexados FULL, por ejemplo)

DANIEL PÉREZ AGUIAR
perezaguiar@programmer.net

17

Entrevista Gastón Cruz

Soy Gastón Cruz, crecí en Colonia del Sacramento y me desarrollé como profesional en Montevideo, Uruguay.

Me he desempeñado como Arquitecto de Soluciones, y líder de proyectos en Uruguay, Chile, Argentina, Perú y USA. Hace ya 20 años que estoy vinculado a tecnologías Microsoft. Particularmente a Power BI desde sus comienzos.

He sido galardonado como MVP en Data Platform en (2016-2017) y (2017-2018).

Actualmente me encuentro liderando proyectos de Data Analytics en Arkano.

¿Por qué y cómo empezaste en el mundo de la tecnología?

Comencé como muchos niños de mi época, con juegos en la vieja TK90, y luego con lenguajes de programación como LOGO y BASIC en mi primera PC XT con Procesador Intel 8086, que me permitieron implementar desde sencillos juegos, hasta sistemas de organización de bibliotecas y logística. Desde esa época, y mientras cursaba secundaria me interesó mucho el tema entrenamientos, por lo que dictaba cursos personalizados en Zona Franca Colonia, y en algunas otras empresas y estudios contables en Colonia del Sacramento.

Al mudarme a Montevideo para proseguir con estudios universitarios, continué con mis tareas como docente en BIOS (Instituto Privado) de Sistemas Contables, Excel Avanzado, Microsoft Project, y luego me transformé en Coordinador Académico, manejando equipo de docentes, preparando materiales, generando nuevas ofertas académicas.

En paralelo a mis estudios en Facultad de Ciencias Económicas realicé la Carrera de Analista de Sistemas donde descubrí que mi verdadera vocación era la de avanzar sobre lenguajes de programación, y tecnologías que le aportarán real valor a verticales de negocio.

Desde ese momento seguí mi camino en el mundo de IT como Arquitecto de Soluciones, Líder de Proyectos, Scrum Master en tecnologías como BizTalk, SQL Server, SharePoint, Project.

¿Cuáles son tus principales actividades tecnológicas hoy en día?

Actualmente me encuentro en un período de transición en lo laboral, preparándome para mudarme a Seattle, USA para incorporarme a una empresa de tecnología allí. Hace ya un tiempo mi foco es en Data Analytics, no sólo en Power BI sino en el vínculo con otras tecnologías y plataformas, que permiten llevar el Análisis de la Información un paso más allá como son: Azure Event Hubs, Azure Data Lake, Azure Data Factory, Azure Analysis Services, Azure SQL DW, Azure DataBricks, entre otras.

El roadmap que tiene Power BI es sumamente vertiginoso por lo que también me gusta mucho mantenerme al día con nuevas funcionalidades de la plataforma, y cómo se vincula con otros servicios y plataformas (SAP, Oracle, Teradata, Saleforce, Sybase).

Esto último me permite a su vez seguir desarrollándome como Trainer y compartir conocimientos en Webinars, Meetups, y conferencias, algo que también me apasiona.

¿Cuáles son tus principales actividades NO tecnológicas hoy en día?

Hoy en día me encuentro, junto a toda mi familia, preparando la mudanza a Seattle. Nos ha absorbido gran parte de nuestro tiempo, pero también nos mantiene muy entusiasmados estar juntos en estos preparativos.

En mis tiempos libres generalmente leo mucho (de tecno-

logía, literatura de suspenso, biografías).

Me gusta ir a la playa, leer un buen libro, jugar tenis de mesa con mi hijo (mañana casualmente cumple ya 16 años). Con mi hija estamos realizando tareas de investigación para ingreso a universidades ya que tiene que realizar esa elección.

Participo en varias comunidades y meetups, en Uruguay, y en la región. Me gusta mucho dar conferencias, y el networking que se genera en las mismas (y donde me he hecho de grandes amigos). Mis favoritas: NetConf Global y SQL Saturdays (en Uruguay, Argentina, Chile, Perú, Colombia).

¿Cuáles son tus hobbies?

Tengo varios, y a medida que pasa el tiempo noto que me gusta ir intercambiando entre uno y otro. Hubo un tiempo en el que tocaba muchísimo piano, ahora ya lo hago esporádicamente. En otros momentos los entrenamientos de tenis de mesa eran una pasión, y una forma de desconectarme del mundo. Ahora estoy comenzando con el running (me gustaría correr algún 10K) y me entusiasma mucho, además de permitirme tener un momento de relax.

¿Cuál es tu visión de futuro en la tecnología de acá a los próximos años?

En lo referente a Data Analytics veo una sinergia muy interesante en lo que se denomina Power Platform, a través de CDM (Common Data Model) y como vía de integración para lograr soluciones Enterprise robustas.

Por otro lado, IA (Inteligencia Artificial) se va a ir incorporando cada vez mas en nuestras tecnologías de base. Sin ir muy lejos, Power BI comenzó con QA, permitiendo que el usuario de negocio pueda realizar preguntas de lenguaje natural, y mediante insights se puedan descubrir correlaciones entre variables de nuestro modelo de negocio. En otro concepto, y hace un tiempo atrás mostraba en un Webinar como realizar un análisis de sentimiento, utilizando un modelo de datos en Power BI y llamando al servicio cognitivo de Azure (utilizando el lenguaje PowerQuery).

La nube llegó para quedarse, por lo que se van a seguir agregando muchos servicios a plataformas nube que nos permitan utilizarlos en nuestras soluciones, y permitiendo capacidades de cómputos que agilizarán la toma de decisiones, realizar análisis predictivo y prescriptivo, a partir de contextos y escenarios generados a través de la tecnología.

Se vienen cada vez mas desafíos, y la tecnología va acompañando, y permitiendo lograr cambios en nuestras vidas. Sin ir muy lejos, en temas de accesibilidad, en temas de salud, se han logrado avances muy interesantes, y muchos más que están por venir.

GASTÓN CRUZ
MVP Data Platform
@GastonFCruz

SharePoint y Azure: Azure Automation

Una de las características más poderosas de Azure como sistema de integración, es su capacidad para ser controlado programáticamente. Azure Automation ofrece un sofisticado servicio de configuración y automatización basado en la nube, que permite administración tanto de Azure mismo como de sistemas no localizados en Azure, tales como Office 365 y SharePoint Online.

Azure Automation ofrece la posibilidad de automatizar tareas frecuentes de administración en la nube, que potencialmente consumen mucho tiempo y son propensas a errores, reduciendo a su vez los costos operativos. El servicio permite crear scripts gráficamente, en PowerShell o con Python. Mediante el uso de scripts híbridos, se pueden combinar recursos en entornos locales con recursos de Azure. Los scripts de Automation se pueden hacer ejecutar automáticamente con un scheduler, manualmente desde el panel de control de Azure, por medio de PowerShell o utilizando un WebHook.

"ofrece un sofisticado servicio de configuración y automatización basado en la nube"

En el contexto de Office 365 y SharePoint Online, se puede pensar en el uso de Azure Automation para la creación de "Timer Jobs" como los que utiliza SharePoint OnPremises, tareas de mantenimiento como el aprovisionamiento de nuevos sitios o cuentas de usuarios o la ejecución automatizada de procesos dentro de SharePoint inicializados por otros sistemas (integración con sistemas externos).

Creación del Servicio de Azure Automation

Para poder utilizar el servicio es necesario primero crear una instancia en su suscripción de Azure. Utilice credenciales de administrador en el portal de Azure (https://portal.azure.com) y cree o reutilice un Grupo de Recursos. Dentro del Grupo, agregue un servicio del tipo "Automation" asignándole un nombre, suscripción, Grupo de Recursos y Localización del centro de datos. En el momento de escribir este articulo solamente existe un esquema de precios que incluye 500 minutos de ejecución de scripts gratis, y pago por cada minuto de ejecución extra cuando el limite gratuito se sobrepasa.

Cuando el servicio ha terminado de ser creado, desde su ventana de manejo se pueden monitorear scripts que están ejecutando o ya han ejecutado, y crear nuevos o manejar los que ya existen:

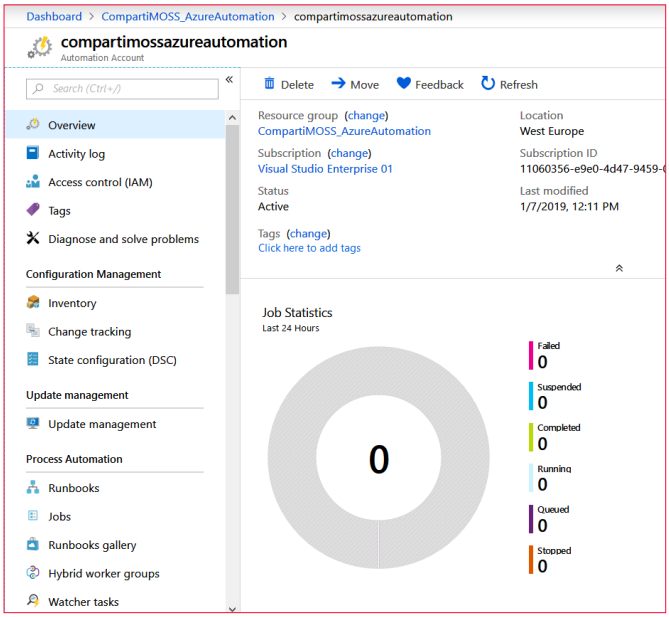


Imagen 1.- Ventana de manejo del Servicio de Automation.

Programación de un script de Automation

Para crear un script de PowerShell para Automation, desde la ventana de administración de servicio haga clic sobre "Runbooks", lo que abre una nueva ventana con la lista de todos los scripts que el servicio maneja (una instancia del servicio de Automation puede contener cientos de scripts):

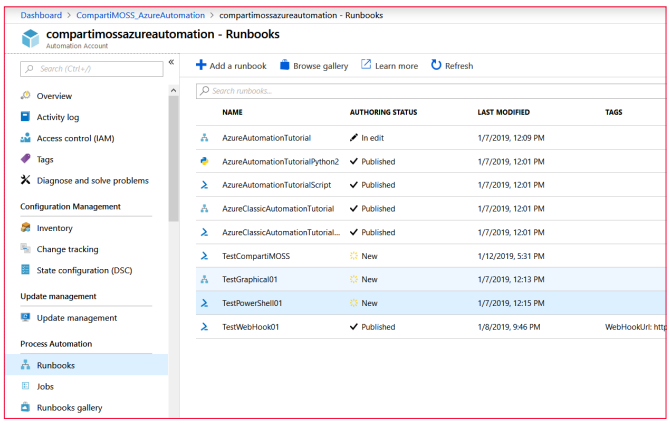


Imagen 2.- Ventana de manejo de Runbooks.

El servicio crea unos cuantos scripts de ejemplo que se pueden eliminar si no se utilizan. Haga clic sobre el botón “Add a runbook” lo que permite crear un nuevo script (“Runbook”) o importar uno desde un archivo. El botón “Create a new runbook” permite asignarle un nombre y seleccionar su tipo (PowerShell, Python, Graphical o PowerShell Workflow). El tipo “Graphical” permite crear wysiwyg scripts basados en “bloques” preexistentes de código de PowerShell. Para trabajar con SharePoint se utiliza siempre el tipo “PowerShell”.

Después de crear un Runbook de PowerShell aparece la pantalla donde se puede crear y editar el código de PowerShell:

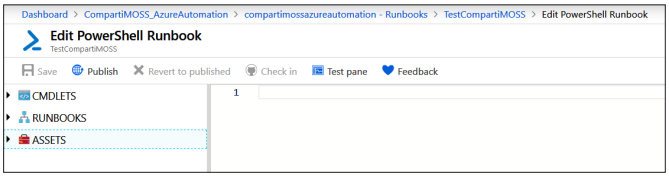


Imagen 3.- Ventana de manejo de un Runbook.

El menú del lado izquierdo permite examinar y agregar al código rápidamente todos los Cmdlets existentes por defecto en el servicio. De igual forma, se pueden conectar con otros Runbooks en el servicio para utilizar su funcionalidad. En la parte de “Assets” se pueden ver y agregar Variables, Conexiones, Credenciales y Certificados.

Una vez el script ha sido creado y testeado, se debe utilizar el botón de “Publish” para indicarle al sistema que el Runbook está listo para ser utilizado en producción.

Agregar Módulos de PowerShell en Azure Automation

Como se ha indicado, el servicio dispone de todos los Cmdlets por defecto que se pueden ver y utilizar en un sistema de Windows “normal”, es decir, todos los Cmdlets que existen en una consola de PowerShell de Windows estándar. Estos Cmdlets no permiten trabajar con Office 365 ni SharePoint Online, por lo que es necesario agregarle los correspondientes módulos o dlls antes de poder utilizar Azure Automation con SharePoint.

Hay tres tipos de módulos que se pueden utilizar para SharePoint (aunque usted puede crear sus propios Cmdlets o dlls, compilarlos y agregarlos a Automation de forma similar a como se indicara en un momento): los Cmdlets del módulo de MSOnline que permiten operaciones básicas con usuarios, grupos y en el Tenant de Office 365, los dlls de SharePoint CSOM (Client Side Object Model) que dan acceso a toda la funcionalidad de SharePoint y los Cmdlets de PnP que amplían la funcionalidad de los dlls de CSOM.

Agregar los Cmdlets de MSOnline

Desde la ventana de manejo de los Runbooks (Imagen 2), utilice el botón de “Modules” (menú izquierdo, bajo la sección de “Shared Resources”). La ventana muestra la lista con todos los módulos de PowerShell disponibles en el momento. Haga clic sobre “Browse gallery” y en la casilla de búsqueda escriba “msonline”. Haciendo clic sobre el módulo «MSOnline» (creado por «AzureADPowerShell») se instala el módulo automáticamente.

Agregar los dlls de SharePoint CSOM

Este módulo utiliza dos dlls de SharePoint: Microsoft.SharePoint.Client.dll y Microsoft.SharePoint.Client.Runtime.dll y permite utilizar las clases, métodos y propiedades presentes en ellos. Los dlls se pueden descargar desde https://www.microsoft.com/en-us/download/details.aspx?id=42038. Descargue desde ese sitio la última versión x64 del sharepointclientcomponents e instálelos o desempáquelos localmente. Copie los dos dlls indicados a un directorio llamado “Microsoft.SharePoint.Client” (utilice este nombre, no lo cambie). Haga un archivo zip del directorio “Microsoft.SharePoint.Client” (no incluya el directorio, los archivos tienen que estar directamente bajo la raíz del archivo zip).

“desde la ventana de administración de servicio haga clic sobre “Runbooks”

Desde la ventana de manejo de los Runbooks (Imagen 2), utilice el botón de “Modules” (menú izquierdo, bajo la sección de “Shared Resources”). La ventana muestra la lista con todos los módulos de PowerShell disponibles en el momento. Haga clic sobre “Add a module” y seleccione el archivo zip creado anteriormente. El servicio sube los dlls a Azure y los configura en el sistema para que puedan ser utilizados por PowerShell.

Agregar los Cmdlets de PnP

Utilice el mismo procedimiento descrito para el módulo de MSOnline, pero en este caso busque por “ SharePointPnPPowerShellOnline”.

Creación de credenciales para Office 365

La idea de utilizar Azure Automation con SharePoint Online es que se puedan crear scripts que ejecuten automáticamente y por sí mismos, por lo que hay que configurar las credenciales de la cuenta de ejecución para que los scripts también se logueen automáticamente.

Desde la ventana de manejo de los Runbooks (Imagen 2), utilice el botón de “Credentials” (menú izquierdo, bajo la sección de “Shared Resources”) para crear credenciales que se puedan utilizar en los scripts. Es posible crear credenciales separadas para cada script, o una sola para todos. El botón de “Add a credential” permite configurar un nombre para la credencial, una corta descripción, la clave

y el usuario (Email). Las claves se conservan encriptadas y no es posible verlas en la interfaz de manejo del servicio.

Programación de scripts de Automation para SharePoint Online

Una vez se dispone de por lo menos uno de los módulos para SharePoint y una entrada de credenciales como se describe anteriormente, es posible comenzar a crear scripts para acceder al sistema.

Script utilizando MSOnline

Cree un nuevo Runbook como se describió anteriormente. En la ventana de edición agregue el siguiente código:

```
# Credenciales
$myCred = Get-AutomationPSCredential -Name "NombreCredenciales"

# Conectar a Office 365
Connect-MsolService -Credential $myCred

# Trabajar con SharePoint/Office
$myDomain = Get-MsolDomain

Write-Output $myDomain
```

La primera línea del script recolecta las credenciales del usuario del Tenant (utilizando el nombre de la entrada que define las credenciales) y la segunda línea las utiliza para logarse en Office 365. El Cmdlet "Get-MsolDomain" del módulo MSOnline recupera los datos del dominio y la cuarta línea los muestra por pantalla. Utilice cualquiera de los Cmdlets presentes en MSOnline para trabajar con SharePoint.

Use el botón de "Save" y luego el de "Test pane". En la ventana de testeo utilice el botón de "Start" para ejecutar el script y luego de unos cuantos segundos aparece el resultado:

"la parte de "Assets" se pueden ver y agregar Variables, Conexiones, Credenciales y Certificados"

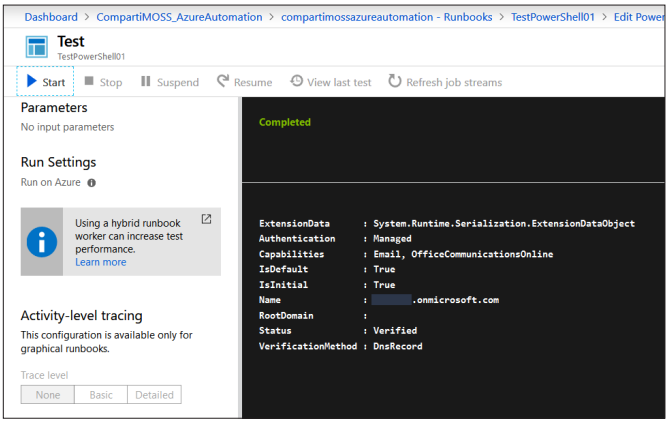


Imagen 4.- Testeo de un Runbook.

Script utilizando CSOM

Cree otro Runbook y utilice el siguiente código:

```
Add-Type -Path "C:\Modules\User\Microsoft.SharePoint.Client\Microsoft.SharePoint.Client.dll"

# Credenciales
$myCred = Get-AutomationPSCredential -Name "NombreCredenciales"
$myUser = $myCred.UserName
$myPW = $myCred.Password

# Conectar a Office 365
$cred = New-Object Microsoft.SharePoint.Client.SharePointOnlineCredentials($myUser, $myPW)
$mySite = "https://dominio.sharepoint.com/"
$myContext = New-Object Microsoft.SharePoint.Client.ClientContext($mySite)
$myContext.Credentials = $cred

# Trabajar con SharePoint
$myWeb = $myContext.Web
$myContext.Load($myWeb)
$myContext.ExecuteQuery()

Write-Output "Web Titulo: " $myWeb.Title
```

Note que en la primera línea se utiliza el nombre del archivo zip como parte de la ruta al dll de SharePoint que se configuró con el módulo indicado en la sección "Agregar los dlls de SharePoint CSOM".

Script utilizando PnP

Cree un nuevo Runbook y utilice el siguiente código:

```
# Credenciales
$myCred = Get-AutomationPSCredential -Name " NombreCredenciales"

# Conectar a Office 365
$mySite = "https://dominio.sharepoint.com/"
Connect-PnPOnline -Url $mySite -Credentials $myCred

# Trabajar con SharePoint
$myContext = Get-PnPContext

Write-Output $myContext
```

Como ejecutar un script de Azure Automation

Existen varias posibilidades para iniciar un script de Azure Automation:

- Manualmente desde la ventana de control de Azure Automation: Como se indicó anteriormente, un Runbook se puede hacer ejecutar manualmente desde la ventana de testeo directamente en el portal de Azure. También, después de publicado, se puede utilizar el botón de "Start" para hacerlo ejecutar manualmente. La ventana de "Overview" del Runbook muestra una lista de los "Recent Jobs" que se han ejecutado. Haciendo clic sobre uno de ellos, se puede obtener información completa sobre cómo fue esa ejecución en particular,

sus errores, output, etc.

- Desde otro Runbook: Un Runbook se puede hacer ejecutar desde otro Runbook, de la misma forma que un script de PowerShell se puede llamar desde otro script de PowerShell. Utilice el Cmdlet "Start-AzureRmAutomationRunbook" de la siguiente manera para ejecutar un script desde otro script:

```
Start-AzureRmAutomationRunbook `
  –AutomationAccountName 'NombreCredenciales' `
  –Name 'NombreRunbook' `
  -ResourceGroupName 'NombreGrupoDeRecursos' `
  -DefaultProfile $AzureContext `
  –Parameters $params –wait
```

Un archivo .ps1 de PowerShell se puede integrar (concatenar) también en otro archivo, de tal forma que el motor de PowerShell los ejecute como si fueran uno solo, de la misma manera que se hace en el funcionamiento normal de PowerShell. La siguiente línea de código integra el archivo "TestPowerShell01.ps1" en el archivo .ps1 actual:

```
.\TestPowerShell01.ps1
```

Ejecución automática por medio de un scheduler

Azure Automation permite acoplar un scheduler a un Runbook. Para hacerlo:

- Cree, testee y publique el Runbook.
- Desde la ventana de "Runbooks" (imagen 2) haga clic sobre "Schedules" (menú al lado izquierdo) y luego "Add a Schedule".
- Defina un nombre, descripción, fecha y hora para iniciar el Schedule y si el timer debe ejecutar una sola vez o recurrentemente.
- Si se utiliza un timer recurrente, defina cada cuanto debe ejecutar y si tiene un tiempo de finalización o debe ejecutar indefinidamente
- Guarde los cambios.

Para acoplar un Runbook a un scheduler, desde la lista de Runbooks seleccione el indicado y desde su ventana de manejo haga clic sobre "Link to Schedule". El botón de "Link a Schedule to your runbook" permite seleccionar uno de los scheduler creados anteriormente, o crear uno nuevo.

Por medio de PowerShell

Los scripts de Automation se pueden iniciar desde una consola local de PowerShell. Primero es necesario agregar el módulo de PowerShell para Azure por medio del Cmdlet "Install-Module AzureRM". Luego utilice un script similar al siguiente ejemplo que inicia logeando interactivamente al usuario, luego ejecuta uno de los Runbooks con un parámetro de entrada y finalmente, por medio de un loop, espera a que finalice la ejecución y muestra el resultado del script por pantalla:

```
# Login en Azure
Login-AzureRmAccount
#Get-AzureRmSubscription / Select-AzureRmSubscription # Por si es necesario cambiar de Subscripcion

$params = @{"ScriptData"="Gustavo estuvo aqui"}  # Use ; para separar dos o mas parametros
$job = Start-AzureRmAutomationRunbook -ResourceGroupName "CompartiMOSS_AzureAutomation" `
        -AutomationAccountName "compartimossazureautomation" -Name "TestRunbook01" `
        -Parameters $params

$doLoop = $true
While ($doLoop) {
  $job = Get-AzureRmAutomationJob -ResourceGroupName "CompartiMOSS_AzureAutomation" `
        -AutomationAccountName "compartimossazureautomation" -Id $job.JobId
 $status = $job.Status
 $doLoop = (($status -ne "Completed") -and ($status -ne "Failed") -and ($status -ne "Suspended") `
         -and ($status -ne "Stopped"))
}

Get-AzureRmAutomationJobOutput -ResourceGroupName "CompartiMOSS_AzureAutomation" `
        -AutomationAccountName "compartimossazureautomation" -Id $job.JobId -Stream Output
```

WebHooks para Azure Automation

Una forma más para hacer ejecutar un script de Azure Automation es por medio de un WebHook. Automation ofrece la posibilidad de crear un WebHook para cada Runbook, es decir, un URL en Internet, que se puede llamar con un método HTTP POST, pasándole parámetros si es necesario, de tal forma que cuando la llamada llega al sitio, el script de Automation comienza a funcionar automáticamente. La llamada al URL no depende de un lenguaje de programación determinado y se puede realizar por medio de múltiples maneras, de tal forma que es totalmente agnóstico en cuando a una u otra tecnología.

"La idea de utilizar Azure Automation con SharePoint Online es que se puedan crear scripts que ejecuten automáticamente y por sí mismos"

Para crear un Automation WebHook para un Runbook:

- Cree, guarde y testee el script de la forma indicada anteriormente.
- Publique el script.
- Desde la ventana del Runbook (después de publicado), haga clic sobre "Add webhook".
- En la nueva ventana, haga clic sobre "Create new webhook" y asígnele un nombre. En la casilla de "Expires" determine por cuanto tiempo el WebHook estará disponible. Copie en un sitio seguro el "URL" pues después de cerrar esta ventana será imposible recuperar el token desde Azure.

Un punto importante en la programación de PowerShell en el código del script es que solamente se puede leer un parámetro de entrada llamado (obligatoriamente) "$WebhookData" del tipo "object". Los nombres y valores de los parámetros se deben enviar en el header de la llamada HTTP y/o en su body. Para leer el parámetro, que contiene tanto el header como el body, primero testee si la variable contiene algún dato, y luego utilice sus valores si es "true", como muestra el siguiente fragmento de PowerShell:

```
Param(
 [string]$ScriptData,  # Para usar como parametro en un script directamente
  [object]$WebhookData  # Para usar como parametro en un WebHook
)

if ($WebhookData -ne $null) {
     Write-Output ("*** Informacion recibida usando un WebHook")
   Write-Output ("Name In Header - " + $WebhookData.RequestHeader.NameInHeader)
    Write-Output ("Text In Header - " + $WebhookData.RequestHeader.TextInHeader)
     Write-Output ("WebHook Name  -  " + $WebhookData.WebHookName)
   Write-Output ("Body - " + $WebhookData.RequestBody)
}
else {
   Write-Output ("*** Informacion recibida con una llamada directa al script")
   Write-Output ("Script Data - " + $ScriptData)
}
```

Para llamar el WebHook se puede utilizar cualquier tipo de lenguaje de programación. Por ejemplo, usando PowerShell:

```
$uri  =  "https://s2events.azure-automation.net/webhooks?token=dtb...MueTw%3d"
$headers = @{"NameInHeader"="nombre en header";"TextInHeader"="texto en header"}
$body = "texto en body"

#$response = Invoke-RestMethod -Method Post -Uri $uri -Headers $headers -Body $body
$response = Invoke-WebRequest -Method Post -Uri $uri -Headers $headers -Body $body


Write-Output ("Respuesta - " + $response)
```

Note que se puede utilizar tanto el Cmdlet "Invoke-RestMethod" como el Cmdlet "Invoke-WebRequest". También es importante observar que el WebHook no tiene ningún tipo de autorización, su única medida de seguridad es el Token en el URL. Si el token es conocido, cualquier sistema puede hacer ejecutar el script sin problemas (siempre es posible agregar código de autenticación personalizado, si es necesario). Finalmente, porque el Runbook ejecuta en forma asincrónica totalmente, es imposible recibir información desde la ejecución del script.

Puntos para tener en cuenta cuando se usa Automation

Cuando se usa Azure Automation se debe tener en cuenta los siguientes puntos:

- Interacción con usuarios: los scripts en Azure Automation ejecutan sin interacción con el usuario, por lo que cualquier error que ocurra en el script no es reportado. Asegúrese de utilizar código para atrapar errores y alguna forma para avisar a los administradores si ocurre uno (enviar un Email, por ejemplo).
- Seguridad: probablemente es una buena idea tener una cuenta de Azure dedicada solamente a la ejecución de scripts de Automation.
- Use "Write-Output" en lugar de "Write-Host" para mostrar información por pantalla.
- Otras posibilidades: existen otras posibilidades técnicas con Automation no mencionadas en este artículo. Por ejemplo, la creación de scripts gráficamente, el uso de "workflows" de PowerShell y el uso de scripts híbridos. Microsoft proporciona toda la información necesaria para usar esas otras opciones.

Conclusiones

Azure Automation es un servicio de Azure que se puede utilizar extensivamente con Office 365 y SharePoint Online, permitiendo la automatización del sistema por medio de scripts de PowerShell. El servicio es muy poco costoso para utilizar, muy poderoso técnicamente, y puede reemplazar otro tipo de soluciones fácil y rápidamente; por ejemplo, puede ser utilizado para crear "Timer Jobs" en SharePoint sin necesidad de crear y mantener toda la infraestructura de sitios web y/o funciones de Azure como se hace tradicionalmente.

GUSTAVO VELEZ
MVP Office Apps & Services
gustavo@gavd.net
http://www.gavd.net

24

Sitios Web Estáticos en Azure Storage

Desde hace muy pocos meses, ya podemos hospedar nuestras webs estáticas en el servicio de Azure Storage GPv2.

¿Qué es una web estática?

Una web estática es aquella aplicación que solo contiene ficheros HTML, CSS, JavaScript o imágenes, a diferencia de las webs dinámicas que dependes de código de servidor. Un ejemplo de una web estática es una aplicación Single Page Application, que solo contiene este tipo de ficheros y que llama a APIs para obtener los datos a mostrar, autenticarse, autorizarse, etc.

Hasta ahora si queríamos alojar una de estas webs en Azure lo teníamos que hacer en Azure Web Apps; también tenemos otros servicios, pero este sería el más sencillo y rápido. El problema que nos encontrábamos es el precio para pagar por alojar este tipo de webs y la rapidez en servir el contenido. Para dar salida a este problema, apareció la opción de alojar este tipo de webs en Azure Storage, lo que nos permite tener unos costes mucho más bajos con una escalabilidad mayor.

En este articulo vamos a ver como funciona y como podemos alojar nuestra web.

Habilitando Static Websites en Azure Storage.

Lo primero que tenemos que hacer es crear un Azure Storage GPv2.

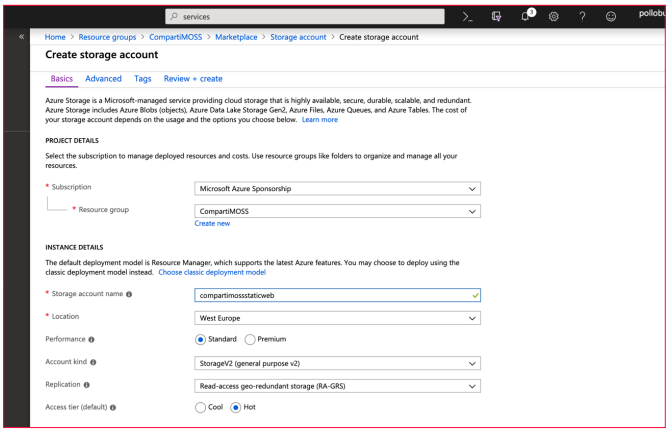


Imagen 1.- Creación de un Azure Storage GPv2.

Una vez lo tenemos creado, habilitaremos la opción de Static website:

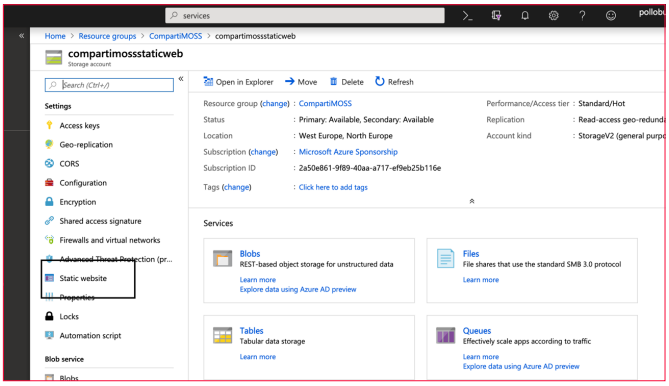


Imagen 2.- Acceso a la opción Static website.

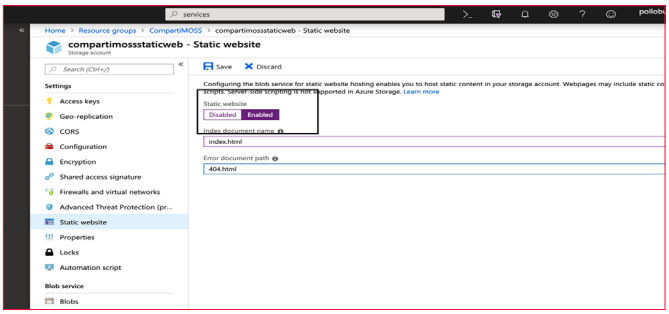


Imagen 3- Habilitando la opción Static website.

"es aquella aplicación que solo contiene ficheros HTML, CSS, JavaScript o imágenes"

Cómo funciona

Cuando se habilita la opción se debe indicar la página de inicio y la página de error 404 not found. Una vez añadimos esta información y guardamos, se crea un container con nombre $web, si ya existe no se crea, el comportamiento habitual de cualquier container en Azure Storage.

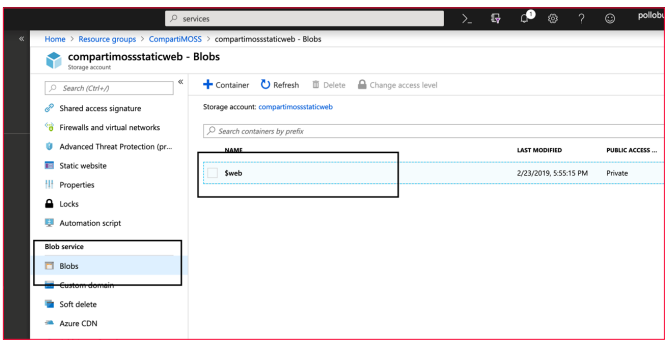


Imagen 4.- Container creado.

Una vez creado también nos da dos endpoints, uno principal y uno secundario que es donde estará alojada nuestra web.

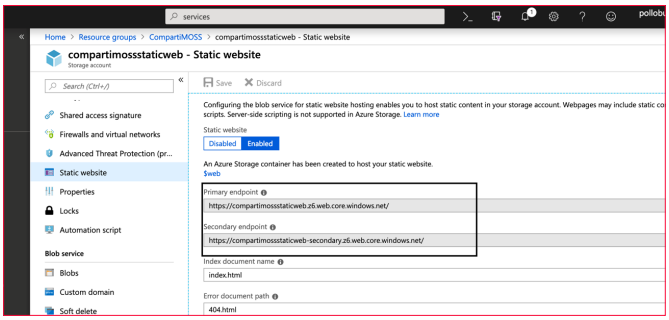


Imagen 5.- EndPoints primario y secundario.

Subiendo los archivos

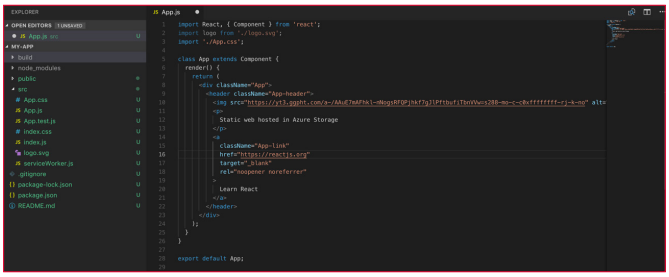
Lo único que debemos hacer es añadir nuestro archivo en el container creado $web, para ello podemos hacerlo de diferentes formas.

- AZzCopy.
- Storage Explorer.
- Azure Pipelines.
- Visual Studio Code Extension.
- Portal.

Ejemplo

Vale, hasta ahora son todo palabras; vamos a subir una web creada en React, la creada por defecto, a Azure Storage como static web. Todos los pasos para crear una aplicación React están aquí: https://github.com/facebook/create-react-app/. Primero creamos nuestra aplicación React: npm init react-app my-app

Después modificamos el fichero app.js para modificarlo y ver que todo funciona correctamente:



Ahora hacemos un npm start y vemos que nuestros cambios han tenido efecto:

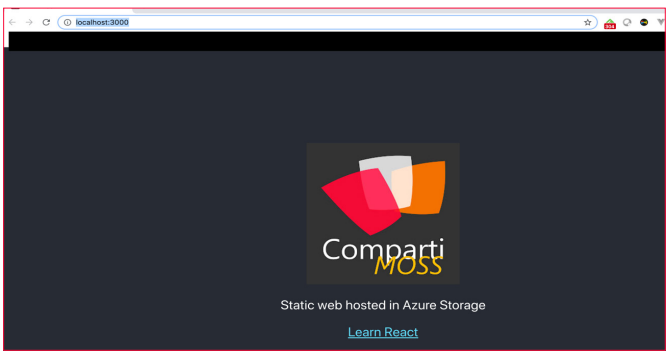


Imagen 6.- Web estática operativa.

Ahora vamos a preparar nuestra aplicación para subirla al Azure Storage, para ello ejecutaremos el comando npm run build. Este comando nos generará la carpeta build y dentro de ella todos los archivos necesarios para que nuestra web funcione.

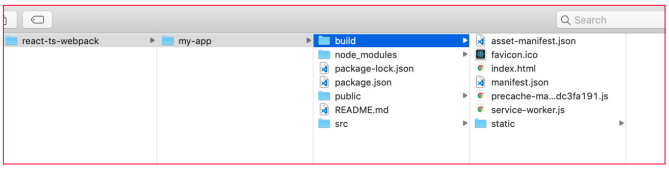


Imagen 7.- Archivos necesarios para que la Web estática se pueda publicar.

Ahora lo que tenemos que hacer es subir todos los ficheros de esta carpeta a nuestro container $web, para ello utilizaremos Microsoft Azure Storage Explorer.

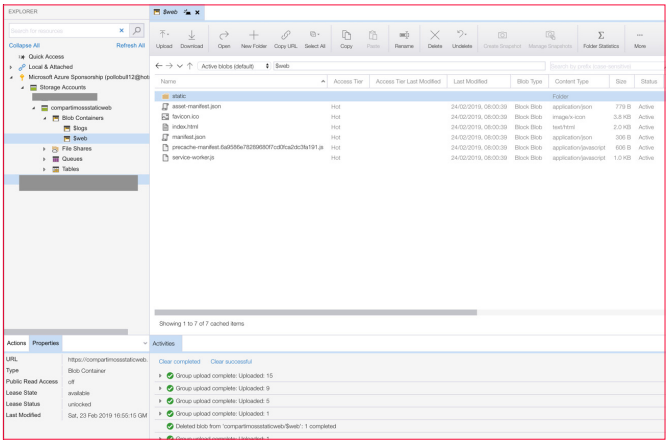


Imagen 8.- Subida de los archivos necesarios mediante Microsoft Azure Storage.

Ahora ya lo tenemos, si accedemos al primary point de nuestra static web en Azure Storage veremos nuestra web.

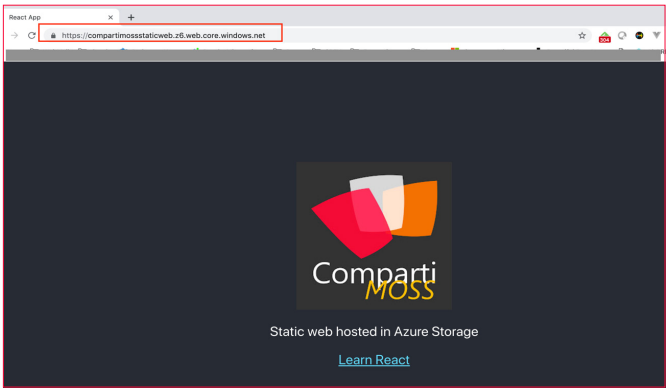


Imagen 9.- Web estática publicada y operativa

Como has podido ver subir una static web a Azure Storage es muy sencillo y rápido, y que las ventajas que nos ofrece son mucha y a tener en cuenta a la hora de alojar nuestras aplicaciones.

ROBERT BERMEJO
Cloud Architect & Technical Lead Consultant at TOKIOTA
Microsoft Azure MVP
bermejoblasco@live.com
@robertbemejo
www.robertbermejo.com

26

Asistentes con .Net – Parte I

Este es el comienzo de una serie de artículos en los que trataré el desarrollo con .net sobre algunos de los asistentes que hay actualmente en el mercado. Creo conveniente comenzar por: "¿Alexa y Assistant son bots o asistentes?" Desde mi punto de vista haría más una diferencia en cuanto a su dominio de trabajo y desarrollo de interfaces de comunicación. Tradicionalmente se ha hablado de los bots como sistemas que solo entendían una única entrada de información, texto y con un dominio muy concreto, como Irene, el "asistente" de Renfe (http://consulta.renfe.com), que aunque lo llamen asistente no lo deberíamos de confundir con Alexa o Assistant, los que yo considero realmente asistentes, ya que su dominio es mucho mayor y variado.

¿Pero entonces un bot y un asistente son distintos? Si y no. Funcionalmente son iguales, en cuanto a ámbito uno tiene mayor envergadura o conocimiento del entorno.

En estos artículos vamos a hablar de asistentes, ya que son los que actualmente nos permiten desarrollar funcionalidades extra sobre ellos (Google Actions y Alexa Skills por ejemplo).

¿Qué partes tiene un asistente?

Todos actualmente tienen las mismas partes y el funcionamiento es igual o muy similar, visto el caso sobre Google Assistant no te lleva más de 2 minutos entenderlo sobre Alexa.

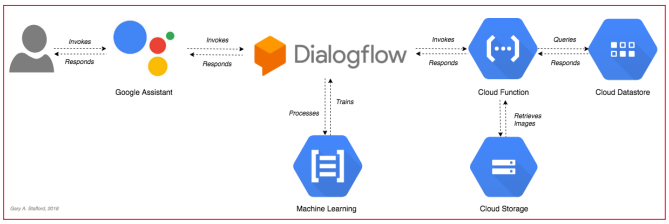


Imagen 1.- Bloques de un asistente.

En el caso de Assistant hay ciertas piezas que no podremos cambiar, que son: Google Assistant y Dialogflow. Assistant es la interfaz de comunicación y transformación text-to-speech y viceversa, podemos decir que esta parte no tiene ninguna inteligencia y se limita a la transformación del mensaje y enviarlo a Dialogflow.

Dialogflow es el servicio NLP (Natural Language Processing) de Google, este servicio es entrenado en base a unas intenciones (Intents) y unas entidades (Entities) para comprender que es lo que quiere el usuario y reaccionar en consecuencia.

Lo veremos más fácil con un ejemplo:

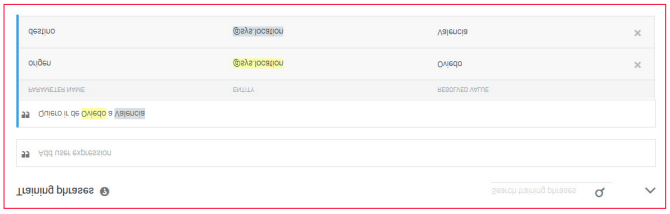


Imagen 2.- Ejemplo de entrenamiento del servicio NLP de Google.

Introduciremos oraciones que signifiquen lo mismo para entrenar el modelo, en este caso podríamos jugar con "Quiero ir", "Me gustaría ir", "Llévame" … indicando posteriormente que irán dos entidades distintas. A la primera entidad la llamaremos "Origen" y a la segunda "Destino" (las entidades de lugar son muy comunes y vienen entrenadas en todos los NLP). Estas son las unidades básicas de un NLP.

Dialogflow se encargará de recibir un mensaje desde Assistant, entender cuál es la intención del usuario en base al entrenamiento, extraer las entidades del mensaje y entonces llamará a la lógica de negocio que hayas indicado, desde responder con un mensaje predefinido a llamar a una función, llamar o no a una base de datos y preparar un mensaje dinámico.

Y ahora estarás diciendo "Vale, pero aún no he visto nada de .net" y tienes razón, pero es muy importante que primero tengas claras las partes de un asistente y luego te explico como sustituirlas por componentes en .net.

Dialogflow (y veremos que el resto igual) tienen un problema y es que Google quiere que uses su NLP, entonces ¿Dónde meto .net? Lo primero que puedes hacer es que Dialogflow consulte en tus APIs .net por una respuesta. ¿Nos vemos en el siguiente número?

Importante: En el siguiente número daré por hecho que sabes construir una API REST y alojarla en Azure.

DIEGO ZAPICO FERREIRO
@dzapic0

27

MS-Teams: Como autenticar nuestros desarrollos

Introducción

Desde el punto de vista del desarrollo, Teams no se había tenido muy en cuenta debido a que la plataforma de por sí ya contiene una gran cantidad de utilidades que se utilizan en el día a día. Dicho uso ha propiciado que las empresas se encuentren con la tesitura de incluir sus desarrollos dentro de Teams. Seguro que este planteamiento a los más viejos del lugar nos recuerda a lo que pasaba con SharePoint en su versión MOSS 2007. ¿Cuáles son las principales diferencias desde hace una década? Principalmente que los tiempos han cambiado, Microsoft Teams no está atado a un framework, permite mucha más libertad a la hora de elegir la tecnología a utilizar. Está pensado para el cloud. Si miramos lo que podíamos hacer antaño era un "simple" User Control y muchas dificultades para incluirlo con otras aplicaciones de la empresa. Por esta parte vamos viendo como Microsoft aprende en parte de los errores que se cometieron en el pasado.

Ahora bien, dentro de esta libertad, que opciones tenemos desde el punto de vista de autenticación:

- SPFx => Con la versión anterior se permite que nuestros desarrollos tal como se ejecutan en SharePoint, se puedan ejecutar dentro de un Teams. Aquí la autenticación va integrada con el propio modelo de desarrollo. La aplicación esta alojada en el catálogo de Aplicaciones de SharePoint Online y dentro de Teams se ejecuta en un iFrame de una página de SharePoint. Por lo que en este tipo de aplicaciones todo está incluido de serie. Ahora bien, en el contexto de una aplicación del mundo real, no todo está en el propio contexto, sino que es posible que se necesite consumir atrás apis/aplicaciones y para ello tendremos que seguir lo indicado por la documentación oficial de SPFx. https://docs.microsoft.com/en-us/sharepoint/dev/spfx/use-aadhttpclient-enterpriseapi
- Aplicación Provider-Hosted=> Recuperó el termino Provider Hosted que se estableció con los Add-in de SharePoint porque el planteamiento es el mismo. El desarrollador elige la ubicación donde va a estar alojada su aplicación (cloud, on-premise). Pudiendo elegir cualquier alternativa que quiera, lo único que necesitamos es la url publica y que se pueda acceder desde cualquier parte del mundo. En este tipo de aplicaciones son en las que vamos a centrar el artículo.

NOTA: Antes de empezar con el tema de la autenticación, tenemos que tener en cuenta una particularidad que tiene el desarrollo en Teams: no se puede hacer redirección a ninguna url que no esté permitida dentro del manifiesto de la aplicación de Teams. Esto quiere decir que si el sistema de autenticación que va a utilizar nuestra aplicación lanza un PopUp/Redirección en el que nos solicita las credenciales, esta url debe de estar permitida. Si no, la aplicación no funcionará, ya que Teams bloqueará dicha llamada y nuestra aplicación no arrancará. Este es un funcionamiento totalmente distinto al que se utiliza dentro de los Tabs de Microsoft Teams (por eso debemos saber muy bien cada una de las particularidades que tiene el desarrollo sobre esta nueva plataforma).

Tipos de Autenticación

Mucha gente piensa que al estar en el contexto de Teams, el desarrollo ya está autenticado. Pero si vamos al concepto que tenemos de una aplicación Provider-Hosted, debemos tener claro que la autenticación del desarrollo es responsable del "Provider". Partiendo de esto, como developer podemos hacer uso de Azure Active Directory (al igual que hace Teams) u optar por otro proveedor de autenticación, ya sea un proveedor Custom o alguno de los comunes (Google, Twitter, Facebook, Instagram, Microsoft, …).

> ***"Teams es la herramienta que está reinventando la forma en la que las empresas llevan a cabo su comunicación y colaboración"***

En el caso de que optemos por otro proveedor que no sea el Azure Active Directory está claro que el usuario deberá poner un usuario y contraseña ya que las cuentas son distintas y desde el punto de vista de la experiencia de usuario es algo que todo usuario acepta sin problemas. De hecho, si miramos la gran mayoría de aplicaciones que hay en la Store de Teams, utilizan su propia autenticación (JIRA, BitBucket, Adobe…) Para incluir esta autenticación en el proyecto basta con seguir las indicaciones de ese proveedor.

Ahora bien, para organizaciones donde ya utilizan Azure Active Directory y tienen aplicaciones que ya utilizan este para acceder, lo normal es que se decanten por hacer uso de esta también para las apps de Teams. Un primer pensamiento (equivocado) es que al estar autenticado ya en

Teams, para poder acceder a nuestra aplicación con que exista una librería en la plataforma que sea la que se encargue de esto puede ser más que suficiente. Pues nada más lejos de la realidad, el hecho de que este autenticado en Teams no implica que tu aplicación pueda coger el contexto de esa autenticación y de esta forma evitar que el usuario se autentique. Tampoco implica que obteniendo los mismos permisos que tiene esta App podamos tener los mismos permisos que tiene Teams.

"Teams no se había tenido muy en cuenta debido a que la plataforma de por sí ya contiene una gran cantidad de utilidades"

¿Entonces como lo podemos hacer? Según la documentación facilitada por Microsoft para poder autenticar nuestra aplicación haciendo uso del directorio activo tendremos que tener una página de autenticación en la misma que se levante en un PopUp. En esta página el usuario introduce su usuario y contraseña y con la devolución del token ya vuelve al contexto de tu página. El flujo de autenticación seria tal que de la siguiente forma:

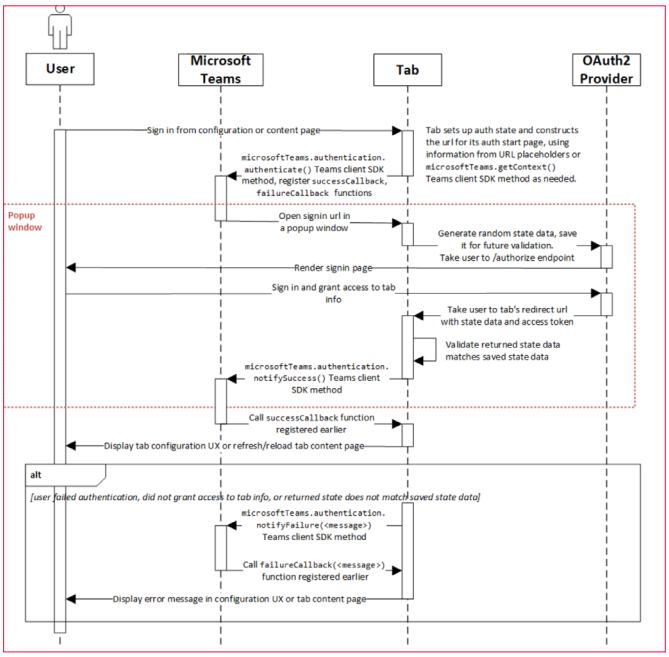


Imagen 1.- Flujo de autenticación en Teams (Fuente: Microsoft).

Ahora bien, una vez contada la teoría, vamos manos a la obra para ver los pasos que tendríamos que hacer:

1.– Para la autenticación vamos a utilizar la librería ADAL.JS para establecer la autenticación. Para ello en primer lugar tendremos una página silent-start.html en la que lo que tendremos la llamada a Adal para autenticar. El código sería el siguiente:

```
<html>
<head>
  <title>Silent Authentication Sample Login</title>
</head>
<body>
  <script src="https://statics.teams.microsoft.com/sdk/v1.0/js/MicrosoftTeams.min.js"  integrity="sha384-SNENyRfvDvybs-t1u0LawETYF6L5yMx5Ya1dIqWoG4UDTZ/5UAMB15h37ktdBby-Fh" crossorigin="anonymous"></script>
    <script src="https://secure.aadcdn.microsoftonline-p.com/lib/1.0.17/js/adal.min.js" crossorigin="anonymous"></script>

  <script type="text/javascript">
      microsoftTeams.initialize();
      microsoftTeams.getContext(function (context) {
        // ADAL.js configuration
        let config = {
          tenant: 'xxxx',
          clientId: "xxxx",
              redirectUri: window.location.origin + "/silent-end.html",
          endpoints: {
            api: 'xxxx',
            graph: 'https://graph.microsoft.com/'
          },
          cacheLocation: "localStorage",
          navigateToLoginRequestUrl: false,
        };

        config.displayCall = function (urlNavigate) {
          if (urlNavigate) {
            if (config.extraQueryParameters) {
              urlNavigate += "&" + config.extraQueryParameters;
            }
            window.location.replace(urlNavigate);
          }
        }

        // Navigate to the AzureAD login page
        let authContext = new AuthenticationContext(config);
        authContext.login();
      });
  </script>
</body>
</html>
```

2.– Una vez tenemos esta página nos vamos a crear otra página silent-end.html con el siguiente código:

```
<html>
<head>
  <title>Silent Authentication Sample Login</title>
</head>
<body>
  <script src="https://statics.teams.microsoft.com/sdk/v1.0/js/MicrosoftTeams.min.js"  integrity="sha384-SNENyRfvDvybs-t1u0LawETYF6L5yMx5Ya1dIqWoG4UDTZ/5UAMB15h37ktdBby-Fh" crossorigin="anonymous"></script>
    <script src="https://secure.aadcdn.microsoftonline-p.com/lib/1.0.17/js/adal.min.js" crossorigin="anonymous"></script>

  <script type="text/javascript">
      microsoftTeams.initialize();

      // ADAL.js configuration
      let config = {
        tenant: '**',
        clientId: "+++",
       redirectUri: window.location.origin + "/silent-end.html",
        endpoints: {
          api: 'xxx',
          graph: 'https://graph.microsoft.com/'
        },
        cacheLocation: "localStorage",
        navigateToLoginRequestUrl: false,
      };
      let authContext = new AuthenticationContext(config);

      if (authContext.isCallback(window.location.hash)) {
        authContext.handleWindowCallback(window.location.hash);
        // Only call notifySuccess or notifyFailure if this page is in the authentication popup
        if (window.opener) {
```

```
            if (authContext.getCachedUser()) {
              microsoftTeams.authentication.notifySuccess();
            } else {
              microsoftTeams.authentication.notifyFailure(auth-
Context.getLoginError());
            }
          }
        }
  </script>
</body>
</html>
```

"no se puede hacer redirección a ninguna url que no esté permitida dentro del manifiesto de la aplicación de Teams"

En esta parte la mayor novedad es el uso de la librería de Teams para JavaScript: para empezar a utilizarlo en primer lugar tenemos que inicializar la variable. A continuación, lo que se realiza es enviar notificación a Teams si la acción ha ido bien o mal. Si la acción ha ido bien se cerrará la página y en caso de error se mostrará una notificación.

3.– ¿Todas estas páginas están bien pero como podemos hacer que nuestra aplicación funcione? Partamos del caso que tenemos una aplicación ReactJS, en el caso tendremos que tener un componente que en primer lugar muestre un botón de Login. ¿Qué acción realizará este botón? Haciendo uso del sdk de Teams lo que haremos será ejecutar un contexto de login, cargando la página creada en el punto 1. Con un código similar al siguiente:

```
login() {
    microsoftTeams.authentication.authenticate({
      url: window.location.origin + "/silent-start.html",
      width: 600,
      height: 535,
      successCallback: (result) => success(result),
      failureCallback: (result) => failureCallback(result),
    });
```

Como veis, el funcionamiento es el siguiente:

- Cargamos un componente React.
- Cargamos el Popup que lo hace en la página silent-start.html.
- Cuando finaliza la autenticación redirige a la página silent-end.html.
- Una vez tengamos el resultado de la autenticación ya volvemos el control a nuestra página.

Otras consideraciones y curiosidades

A la hora del desarrollo de una APP nuestra aplicación deberá poder ejecutarse tanto en Teams como fuera del contexto de Teams. Por este motivo es posible que la autenticación pueda ser diferente dentro de Teams como fuera. Pongamos por caso que nuestra aplicación se autentique haciendo uso de AdalJS cuando está fuera de Teams y cuando estemos en Teams tengamos que hacer el Flow que hemos incluido anteriormente.

Otro de los aspectos más curiosos a la hora del propio del funcionamiento que tiene Teams es que no es lo mismo que ejecutemos Teams, en el navegador, en la App o en un dispositivo móvil. El SDK tiene un método en el que nos indica si estamos dentro de Teams o no. Sin embargo, este método no funciona cuando estamos vía web. Por lo que para solucionarlo Wictor Willen publicó un workaround que es poner el siguiente método:

```
const inTeams = (): boolean => {
  try {
    return window.self !== window.top;
  } catch (e) {
    return true;
  }
}
```

En este articulo estamos centrando en la propia autenticación contra un sistema y dejamos para posteriores artículos como poder tener varios endpoint en los que vamos a autenticar. Ejemplo tengo una API y este api por un lado se autentica contra el Azure Active Directory, pero también accede a algunos endpoints de Graph. ¿Como lo podemos hacer? Todo esto lo abordaremos en futuros artículos.

Conclusiones

Microsoft Teams es un amplio abanico para el desarrollador, el que nos permite multitud de integración con diversos sistemas. Sin embargo, a este modelo de desarrollo le faltan aspectos, tener una forma de autenticar sencilla, poder utilizar el propio token que tiene Teams, poder incluir permisos para Graph, etc. Supongo que todo que se echa de menos es debido al rápido crecimiento y que en un futuro cercano podamos tenerlo y disfrutemos de la plataforma y como poder incluirle valor. Mientras tanto nos queda buscar alternativas para poder cumplir con los requerimientos solicitados.

ADRIÁN DIAZ CERVERA
Architect Software Lead at Encamina
MVP Office Development
http://blogs.encamina.com/desarrollandosobresharepoint
adiaz@encamina.com
@AdrianDiaz81

30 Entrevista con KWizCom

Since 2005, KWizCom has provided innovative solutions and services to make SharePoint even better for over 7,000 companies worldwide. KWizCom is a leading provider of SharePoint Forms, Workflows, Mobile, Wiki solutions, and over 70 other add-ons for SharePoint on-premises and apps for Office 365. KWizCom is a Gold Certified Microsoft Partner is headquartered in Toronto, Canada. To find out more about the company and its products, please visit www.kwizcom.com.

Official distributor in Spain:

Bittek Soluciones Tecnológicas
website: https://kwizcom.bittek.eu/
email: kwizcom@bittek.eu

When and why your company started to operate in the tech world?

KWizCom was founded in 2005 in Toronto, Canada. After encountering the numerous business needs unsolved by out-of-the-box SharePoint, we saw the need to develop pocket-friendly and easy to use add-ons that enhance SharePoint and are specifically designed for non-technical business users. Solutions that provide freedom to end-users to make SharePoint even better and customize it according to their business needs without having to rely on technical people or power users. KWizCom developed over 60 plug-ins that solve various SharePoint challenges and rapidly became one of the leading providers of SharePoint add-ins worldwide. Our portfolio of products is comprised of easy to use add-ons and offers a solution for almost any gap in SharePoint.

What are the main technical activities the company does Today?

KWizCom communicates with its customers on a regular basis, thoroughly learns its customers' needs and, accordingly, develops powerful yet easy to use SharePoint add-ins and apps for SharePoint Online to empower end users to address their business needs without having to rely on technical people. KWizCom's product portfolio is constantly updated with new version releases and new solutions for SharePoint and Office 365. Additionally, the company offers professional services where customization of certain features or a solution is done to meet the specific customer business requirements.

Last but not least, KWizCom's technical support team provides outstanding service and firmly stands behind each client to support and guide them at every step of their journey.

What are the main non-technical activities the company does Today?

KWizCom is an active supporter of various charity organizations and gladly sponsors some of their events, as well as provides free products for their company usage.

What are the activities the company does to support Microsoft Technical Communities?

KWizCom is an active supporter of numerous Microsoft technical communities and events. We support various Microsoft communities throughout the world by sponsoring their events and conferences.

Moreover, KWizCom frequently hosts technical web casts where the attendees learn valuable knowledge which they can implement in their daily work while working in SharePoint. It's worth noting that the company has a page dedicated to technical blog posts, and often participates in online discussion boards to share valuable experience, tips and their expertise. Importantly, KWizCom also has several open source projects that the company shares with the

community on GitHub.

KWizCom Forms is one of your most popular products. Why did you develop it and what does it do?

We have talked to myriads of SharePoint users and our existing customers to learn about their business needs and the challenges they face in their daily lives when working with SharePoint. Numerous times we heard similar stories that SharePoint end-users with no technical skills had to depend on forms experts to create the dynamic forms they need. Having to rely on someone else for simple forms customization created a bottleneck in many companies (which in turn, increased the cost of form creation and maintenance). The business users wanted one thing - to simply get the freedom to create smart and flexible forms depending on their specific business requirements. The existing Forms solutions are designed for technical power-users. Tools such as Microsoft Power Apps and Microsoft InfoPath, as well as other 3rd-party Forms tools are all similar in the way they all display a rich, “visual studio”-like designer which is far too technical for business-users that obviously cannot get near such a tool.

KWizCom has listened closely to the feedback and took the challenges the SharePoint users were facing to heart and has developed a powerful yet user friendly solution to address these issues.

KWizCom Forms was developed as a result, which empowers non-technical SharePoint users to effortlessly create sophisticated dynamic and smart forms from scratch in a matter of minutes. Designed for business end-users and not for power users, this cutting-edge Office 365 add-in dramatically reduces the cost of ownership as it removes the bottleneck of forms creation, because now many more users can easily create flexible forms.

Additionally, KWizCom Forms add-in is a SharePoint-hosted app which means it does not replace the SharePoint native forms, but rather enhances them, and allows customers to keep their SharePoint extensibility features (such as add custom columns to their forms).

32

Nuevas opciones de personalización en sitios modernos de SharePoint Online

Desde hace un par de años estamos asistiendo a la "modernización" de la plataforma SharePoint de la mano de una serie de innovaciones que Microsoft ha ido liberando en la versión cloud de la plataforma: páginas modernas, listas y bibliotecas modernas, WebParts de SharePoint Framework, etc. Sin embargo, esta "modernización" tenía (y sigue teniendo) un punto débil: las posibilidades de personalización de caja y sin recurrir a desarrollo son muy limitadas. Afortunadamente, uno de los puntos fuertes de SharePoint Online (SPO) es que continuamente tenemos mejoras y novedades en la plataforma como la reciente incorporación de opciones de personalización por defecto en sitios modernos. Como veremos en este artículo, las posibilidades de personalización siguen siendo reducidas, pero apuntan en la buena dirección.

Nuevas opciones de personalización liberadas

Las nuevas opciones de personalización de sitios modernos se encuentran disponibles en la opción "Change the look" ("Cambiar el aspecto") dentro del menú de configuración del sitio:

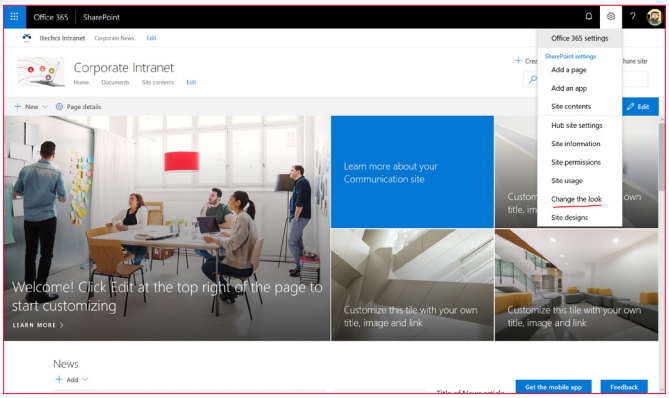


Imagen 1.- Acceso a la opción "Change the look".

Desde esta opción tendremos acceso a 4 configuraciones de cambio de aspecto del sitio:

- Theme (Tema), nos permite cambiar el tema del sitio por alguno de los temas disponible por defecto o bien por un tema personalizado que se haya desplegado en el tenant. Para crear nuevos temas personalizados, os recomiendo las siguientes referencias:
  - Referencia oficial en la documentación de Microsoft sobre temas en sitios de SPO: https://docs.microsoft.com/en-us/sharepoint/dev/declarative-customization/site-theming/sharepoint-site-theming-overview
  - Herramienta online para generar nuevos temas: https://developer.microsoft.com/en-us/fabric#/styles/themegenerator.
  - Como provisionar un tema personalizado en un tenant de SPO (Ejemplo de Joao Ferreira, colaborador habitual de CompartiMOSS): http://sharepoint.handsontek.net/2018/03/11/brand-modern-share-point-online-sites/

"continuamente tenemos mejoras y novedades en la plataforma como la reciente incorporación de opciones de personalización"

- Header (Cabecera), permite configurar la cabecera del Sitio en aspectos como el layout a utilizar, el logo del sitio o el color de fondo.
- Navigation (Navegación), permite elegir la experiencia de navegación del sitio.
- Footer (Pie de página), permite configurar el pie de página del sitio.

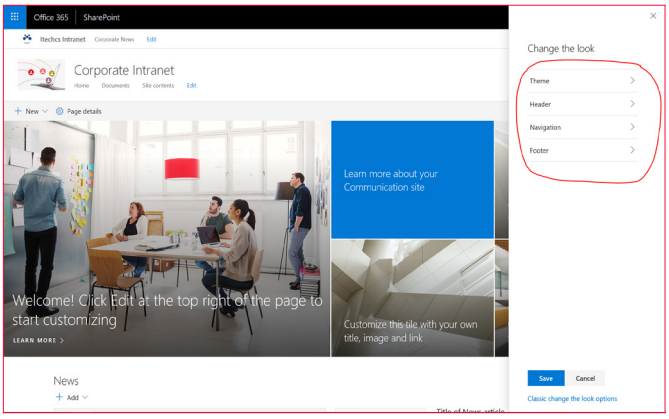


Imagen 2.- Opciones de configuración de aspecto de un sitio moderno de SPO.

Configuración de la cabecera

La opción de configuración de cabecera de un sitio moderno permite realizar los siguientes cambios:

- El diseño (layout), permitiendo elegir entre un diseño estándar (opción por defecto) o bien uno compacto.
- El logo del sitio.
- El color de fondo de la cabecera, permitiendo elegir entre cuatro combinaciones posibles.

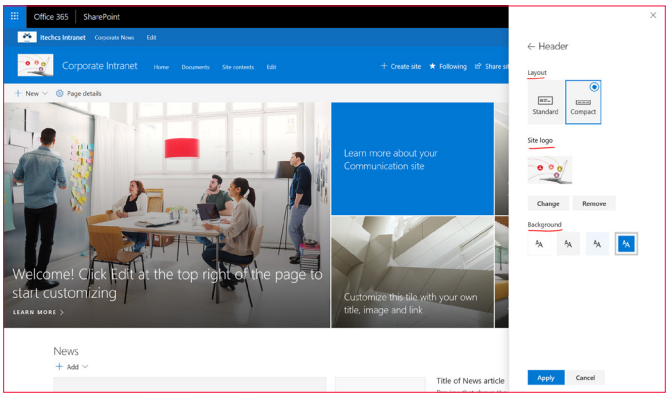


Imagen 3.- Opciones de personalización de la cabecera de un sitio.

Una vez hemos realizado los cambios a nivel de cabecera, es importante no solo aplicarlos, si no también guardarlos para que sean efectivos.

Configuración de la navegación

Esta opción permite elegir la experiencia de usuario en la navegación entre secciones del sitio de acuerdo con dos posibilidades:

- Cascada (Cascading), que es la opción por defecto.
- Megamenu.

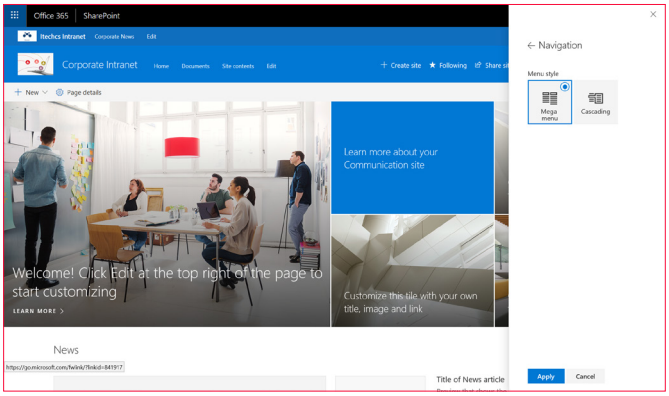


Imagen 4.- Opciones de configuración de la navegación del sitio.

Un ejemplo de Megamenu (Imagen extraída de contenidos de Microsoft) es el siguiente:

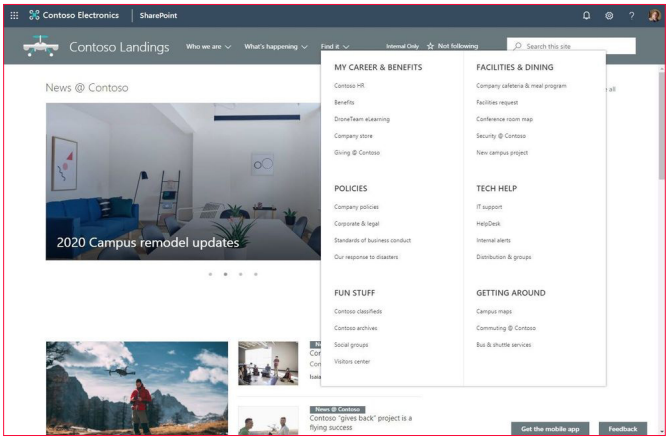


Imagen 5.- Ejemplo de Megamenu extraído de contenido de Microsoft.

Configuración del pie de página

Finalmente, el último elemento que podemos configurar con opciones de serie en sitios modernos de SPO es el pie de página. Para ello haremos uso de la opción del mismo nombre, de manera que podremos:

- Indicar si el pie de página va a estar visible a lo largo del sitio.
- El icono para mostrar en el pie de página en el caso en el que esté habilitado.
- El texto para mostrar en el pie de página en el caso en el que esté habilitado.
- Configurar los enlaces a mostrar en el pie de página en el caso en el que esté habilitado.

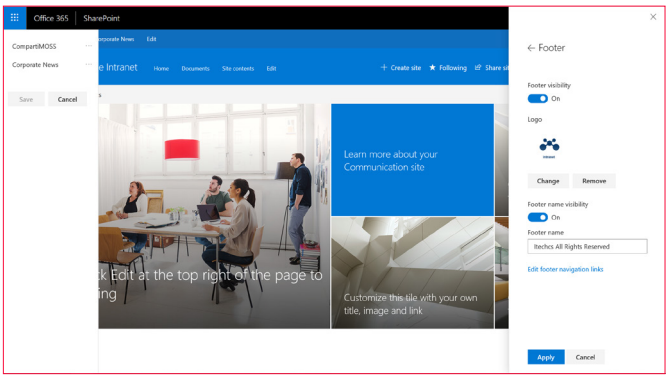


Imagen 6.- Configuración del pie de página de un sitio moderno.

> ***"el último elemento que podemos configurar con opciones de serie en sitios modernos de SPO es el pie"***

El resultado de realizar todas las configuraciones indicadas en el pie de página es el siguiente (*Nota: para visualizar el pie de página se necesitará hacer scroll en la página*).

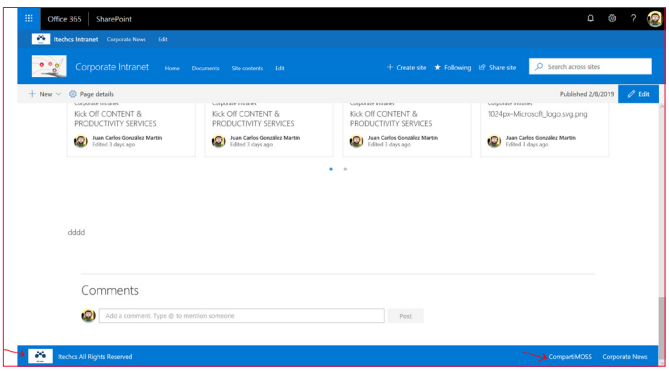


Imagen 7.- Pie de página configurado en el sitio.

Configuraciones a nivel de Hub Sites

Además de las opciones descritas de personalización para un sitio, si este es un Hub Site también podremos realizar dos personalizaciones básicas:

- Cambiar el logo del Hub Site.
- Cambiar el nombre del Hub Site.

Ambas opciones se muestran hacer clic en “Hub site settings” en el menú de configuración del sitio.

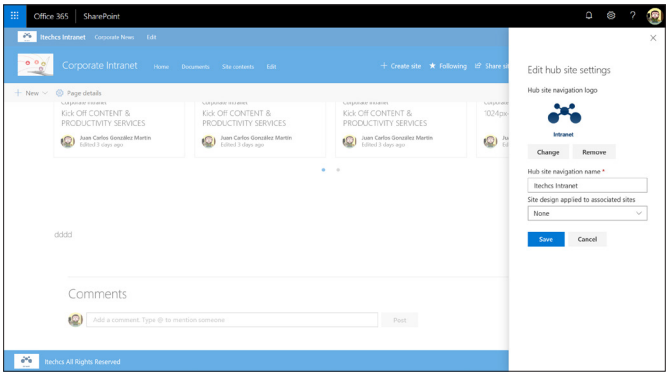


Imagen 8.- Opciones de configuración a nivel de Huib.

“Las posibilidades de personalización de sitios modernos de SPO han sido y siguen siendo bastante reducidas.”

Conclusiones

Las posibilidades de personalización de sitios modernos de SPO han sido y siguen siendo bastante reducidas. Sin embargo, las novedades recientes para personalizar mínimamente un sitio de SPO hacen pensar que poco a poco Microsoft nos irá proporcionando distintas posibilidades para personalizar sitios sin necesidad de recurrir al desarrollo.

JUAN CARLOS GONZÁLEZ
Office Apps & Services MVP | Office 365 SME
@jcgm1978

35

Azure AD, quién te ha visto y quién te ve

Al principio era solo “eso que gestiona las identidades de Office 365”, donde se asignan las licencias y poco más. Pero Azure AD es a día de hoy un robusto y completo servicio de Identity and Access Management (IdAM), con funcionalidades que le crecen como setas y características avanzadas de integración y seguridad.

Imagen 1.- Azure AD.

Azure AD Provisioning Service

El Servicio de aprovisionamiento de usuarios de Azure AD no es que sea totalmente nuevo, pero es una de las funcionalidades que más está evolucionando. Azure Active Directory (Azure AD) permite automatizar la creación, el mantenimiento y la eliminación de identidades en otras aplicaciones cloud como Dropbox, Salesforce, ServiceNow, etc. Esta funcionalidad incluye:

- Creación de cuentas automáticamente en otras aplicaciones cloud cuando se crean nuevos usuarios en Azure AD.
- Desactivación automática de cuentas en otras aplicaciones cloud cuando se dan de baja en Azure AD.
- Propagación automática de las actualizaciones de datos de cuentas de usuarios.
- Creación de otros tipos de objetos, como grupos, para las aplicaciones que lo admitan.

“una mera extensión de nuestro Directorio Activo en la nube, poco a poco se nos hace mayor”

El aprovisionamiento automático de usuarios también incluye:

- Coincidencia de las identidades entre Azure AD y las aplicaciones cloud de destino.
- Personalización del flujo de atributos entre Azure AD y la aplicación cloud de destino.
- Alertas de errores de aprovisionamiento.
- Informes y registros de actividades.

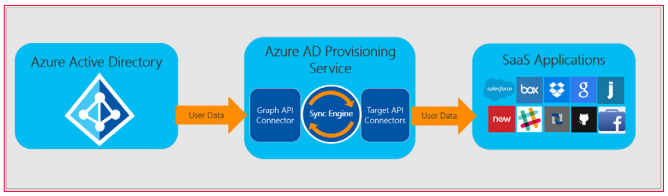


Imagen 2.- Azure AD Provisioning Service.

¿Cómo funciona?

Azure AD Provisioning Service utiliza la API de cada aplicación para crear, actualizar y deshabilitar usuarios por código. Así de fácil. Según cada aplicación, Azure AD Provisioning Service puede también crear, actualizar y deshabilitar otro tipo de objetos relacionados con la identidad, como grupos y roles.

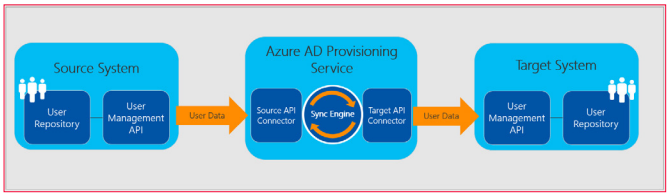


Imagen 3.- Cómo funciona.

“es una de las funcionalidades que más está evolucionando”

Pero eso no es todo. Además de ser capaz de aprovisionar usuarios a otras aplicaciones cloud, ahora también podemos aprovisionar usuarios en Azure AD que provengan de otras aplicaciones. Whatttt? Como lo oyes. ¿Alguien dijo MIM?

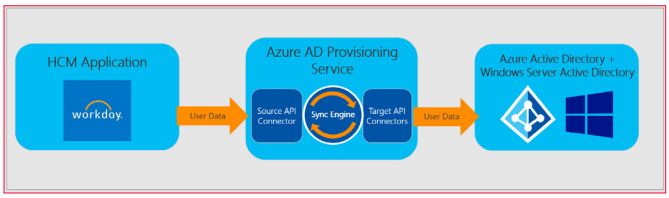


Imagen 4.- Azure AD Inbound User Provisioning.

De momento esta integración sólo está disponible con Workday, pero ya os podéis imaginar donde acabará todo

esto.

https://docs.microsoft.com/en-us/azure/active-directory/manage-apps/configure-automatic-user-provisioning-portal

Y si la aplicación cloud aún no tiene integración con Azure AD, utilizando SCIM (System for Cross-Domain Identity Management) podemos aprovisionar usuarios a cualquier aplicación que exponga un web Service.

https://docs.microsoft.com/en-us/azure/active-directory/manage-apps/use-scim-to-provision-users-and-groups

En la sección provisioning de las aplicaciones que tengamos integradas, podemos personalizar el mapeo de atributos entre Azure AD y esa aplicación.

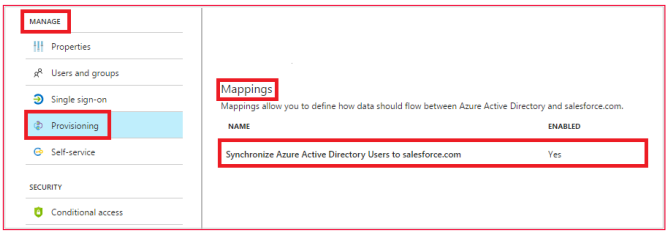


Imagen 5.- Mapeo de atributos.

"Poco a poco Azure AD se está convirtiendo en el gran motor de la gestión"

Conclusión

Poco a poco Azure AD se está convirtiendo en el gran motor de la gestión de identidades cloud. El salto a ser también un gestor de las identidades on-prem está al caer. En cualquier momento soportará ECMA2, y si nos fijamos bien, todos los desarrollos de Microsoft en el área de identidades se los lleva Azure AD. Ahí lo dejo.

PABLO ORTIZ BAIARDO
Infrastructure & Cloud Consultant
ortiz.pablo@gmail.com
@portiz2017
https://www.linkedin.com/in/portiz

37

Convert Classic root sites to Modern SharePoint

Recently I've got a few requests to transform classic root team sites with publishing enabled into modern SharePoint sites. The technic described in this article allows you to keep all your existent content, and at the same time allows you to take advantage of the out of the box responsive layouts and modern web parts.

Create a modern landing page

1.– On your root site collection navigate to a modern Library. If the library is not being displayed with the modern layout do the following:

- Open the browser developer tools by pressing F12.
- In the console type "GoToModern();" and press Enter.
- Wait for the page to refresh.

2.– Click on settings and then "Add a page":

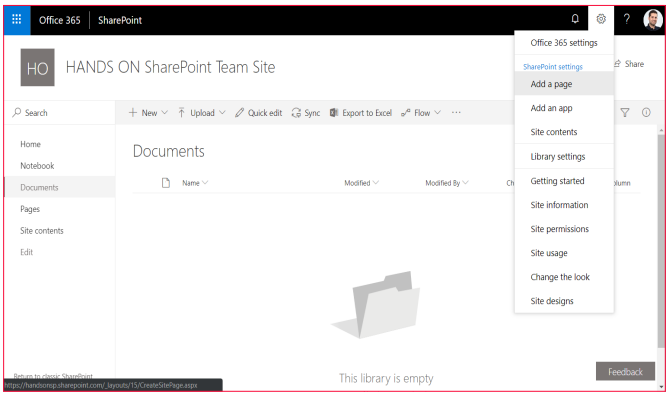


Image 1.- Adding a new modern page to the site.

3.– Once created give it a name and build your new landing page.

4.– Once you've built it according to your needs go to "Site Contents".

5.– Open the Site Pages library.

6.– Click on the "Show Actions (…)".

7.– Click on "Make Home Page".

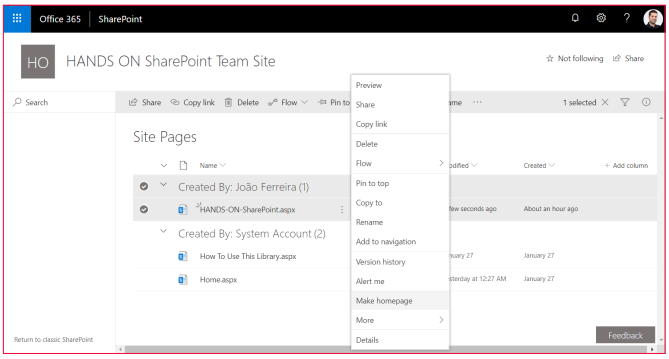


Image 2.- Making the page as site home page.

"few requests to transform classic root team sites with publishing enabled into modern SharePoint sites"

Manage the navigation

The new landing page is being displayed as a modern Team Site showing just the vertical navigation, to use the horizontal navigation follow the steps below.

1.– Go to Site Settings.

2.– Click on "Navigation Elements".

3.– Uncheck the "Enable Quick Launch".

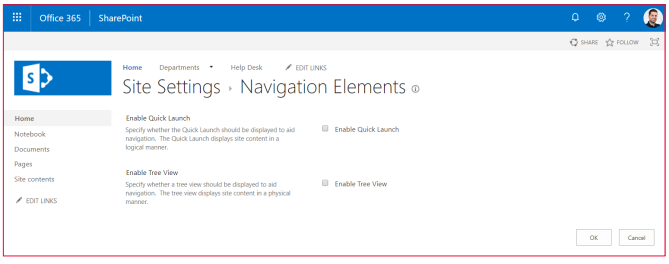


Image 3.- Modifying quick launch settings in the site.

4.– Click OK.

5.– Edit the navigation on the new landing page.

6.– Delete all the Items under "Current Navigation".

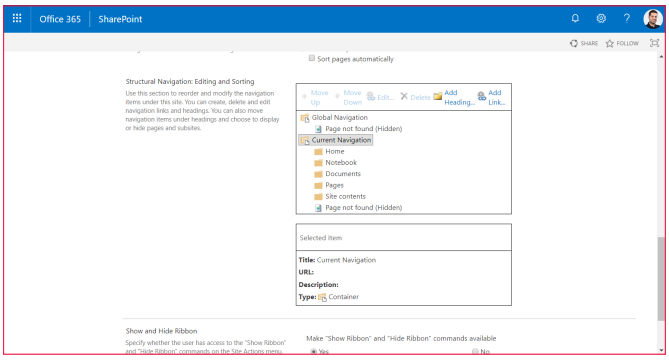


Image 4.- Modifying navigation settings in the site.

7.– For the global navigation I suggest you use "Metadata Navigation", it will increase the overall performance of your SharePoint site. After selecting the term for the global navigation, it will take a few minutes to be visible in the new modern page.

"new landing page is being displayed as a modern Team Site showing just the vertical"

Create a full width landing page

Even though you now have a modern landing page the content is still being formatted like a team site. To get it like a modern communication site with a full width top area and centered web part zones I've used a modified version of the Team Site Full Width Extension built by Luis Ribeiro.

This extension will make your site full width giving it a modern look and feel you are looking for.

To install the extension, follow the steps blow:

1.– Download and extract the solution.

2.– Open your App Catalog and upload it.

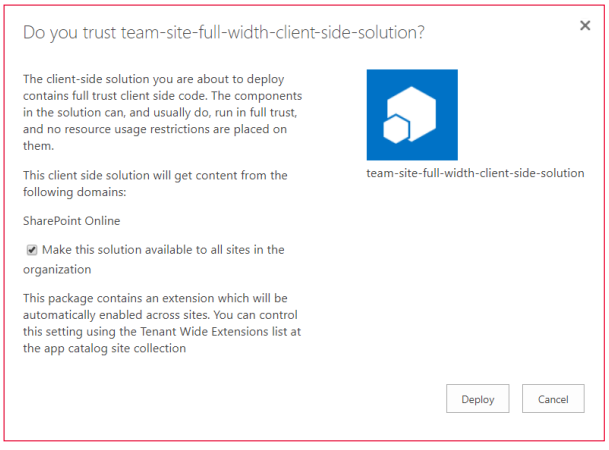


Image 5.- Installing the solution.

3.– Run the install.ps1 file.

4.– Provide the URL to the site collection where you want to apply the solution.

5.– Type Y to make the first web part zone full width.

6.– Type N to just center all the web part zones.

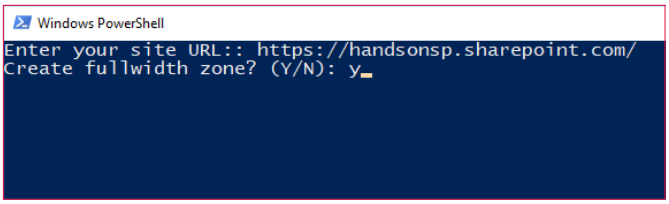


Image 6.- Script execution to configure the solution.

The result will look like the image below.

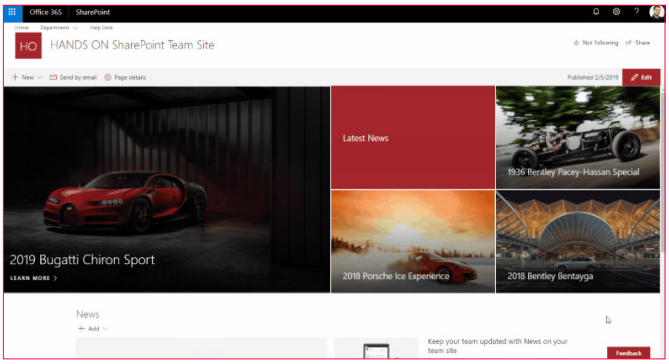


Image 7.- Final result.

"though you now have a modern landing page the content is still being formatted like"

Conclusion

This is not the only way to make your root site as a modern SharePoint site, this approach is suitable for scenarios where you need to keep your classic pages as they are but still want to take advantage of the modern SharePoint features.

If this approach is not exactly what you need have a look at this thread in the Microsoft Tech Community.

As always, the code for this solution on GitHub, feel free to use it and contribute to make it better.

JOAO FERRERIRA
Office Development MVP
SharePoint Team Lead at Bindtuning

Adiós Exchange UM Online

Así como estamos diciendo adiós a gran parte de las características de Skype for Business (podríamos decir que a todo el producto completo), es momento de irnos despidiendo también de los roles "hermanos" que residen en Exchange. Ahora es el turno de dar paso a la nueva generación de servicios basados en la nube, e irónicamente remplazando a una versión online, adiós Exchange Unified Messaging Online y bienvenido Cloud Voicemail.

Una despedida que se veía venir desde hace tiempo como varios de ustedes sabrán, y si no aquí estamos para informarlos. Exchange Server 2019, ya no cuenta con el rol de UM eso nos daba una pista que en su versión online dejaría de existir próximamente, igual nos pasa con el servicio de Skype for Business, tenemos nuestra versión on-premises pero en la nube las organizaciones nuevas con menos de 500 empleados ya no tendrán este servicio, sino que será ofrecido directamente con Microsoft Teams.

"es momento de irnos despidiendo también de los roles "hermanos" que residen en Exchange"

El sustituto Cloud Voicemail ya tiene un tiempo siendo utilizado con el servicio de Skype for Business Online y con Teams, así que podríamos decir que es un producto probado y con las necesidades de negocio cubiertas, y bien dije "podríamos decir" ya que tenemos que observar muy bien que es lo que nos quitaría esta nueva versión online, y que cosas nuevas nos trae.

Pero primero vamos a dar la fecha en la cual Microsoft dejara de ofrecer este servicio, ¿preparados? Febrero 2020 será la fecha en la cual Microsoft dirá adiós definitivo al servicio de ExUMO. Así que vamos a tener que prepararnos, y evaluar en qué fecha nos estarán migrando. Porque Microsoft conoce que hay ambientes con requerimientos particulares que no pueden migrarse de la noche a la mañana, sino que necesitan tiempo y analizar bien todas las opciones que se tienen y llevar una migración limpia (o al menos, lo mejor que se pueda).

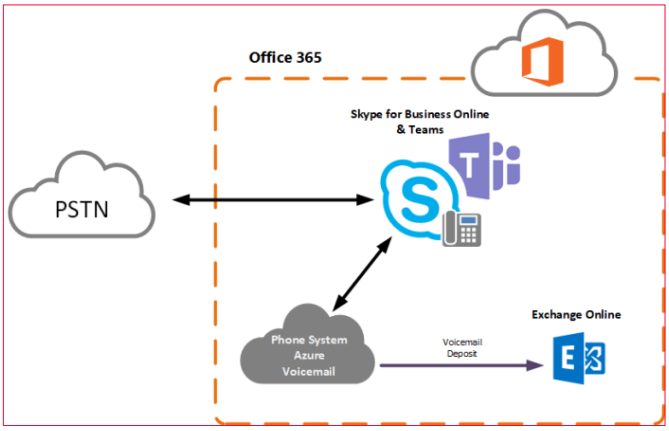


Imagen 1.- Ecosistema de servicios de UM de Microsoft.

Microsoft propone dividir la migración en 3 tipos diferentes de cliente:

Cliente 1

Compañías que se encuentran listas para la migración de acuerdo con lo siguiente:

- Clientes que usan el buzón de voz de una manera sencilla, es decir solo reciben las llamadas y las mandan a voicemail en caso de no contestar. No hay tratamiento de información más allá de esta simple tarea.

¿Qué migrarán para estos clientes? – Únicamente el correo de voz, ya que es lo único que utilizan.

¿Cuándo serán migrados? – Entre Marzo y Mayo de 2019.

Cliente 2

Compañías que tienen algún prerrequisito, es decir alguna interacción más allá de solo el uso de correo de voz.

- Compañías que tienen configurado AutoAttendant.
- Compañías con topología hibrida.
- Compañías que aun trabajan con extensiones en lugar de DIDs.

¿Qué migrarán para estos clientes? – Correo de voz, así

como los autoattendants. Y es un decir, ya que Microsoft se encargará de migrar los correos de voz, pero la parte de AutoAttendants será delegada a nuestros queridos administradores de sistemas, es decir que prepárense para migrar tantos AA como tengan configurados hoy en día, respetando los flujos y redireccionamientos que tienen hoy, sino tienen DID's y los mandan a extensiones, ahí empezaremos con algunos problemillas.

¿Cuándo serán migrados? – Entre Mayo y Diciembre de 2019

Cliente 3

Clientes que necesitan mucha actividad por parte de sus administradores, o que requieren alguna inversión adicional.

- Si tienen ExUMO integrado con alguna aplicación de terceros.
- Si necesitan Suscriber Access.
- Clientes con Lync 2010.
- Integración con Fax.

¿Qué migraran para estos clientes? Voicemail, AA y todos los demás pendientes, no será una batalla sencilla sobre todo para los pobres que tienen actualmente la integración con terceros, si bien Microsoft nos da tiempo necesario, pero va a requerir una inversión extra. (Dentro de las cuales esta, moverse a Skype for Business hybrid al menos).

¿Cuándo serán migrados? – Para febrero 2020

> ***"También si solo eres un usuario de Exchange y no de Skype la cosa esta peor, ya que no podrás continuar usando ExUMO para tu PBX"***

Como verán las cosas no pintan tan sencillo como parecen, si solo tienes que usar el buzón de voz estás del otro lado, pero si empezamos a meter más variables se pone color de hormiga (como decimos aquí en México).

También si solo eres un usuario de Exchange y no de Skype la cosa esta peor, ya que no podrás continuar usando ExUMO para tu PBX. Tendrás que seleccionar entre dos opciones. O te migras a la solución de telefonía de Microsoft (Skype o Teams) o vas a tener que comprar un appliance con algún vendor para que te pueda ofrecer el servicio ($$$).

Espero que tengan una buena planificación para llevar su servicio a cloud voicemail, o en caso de no ser posible, que encuentren la solución que mejor se adapte a sus necesidades.

Por último, les dejo una tabla comparativa entre Cloud Voicemail vs ExUMO.

ExchUMO and Azure cloud-based services feature matrix

SERVICE	FEATURE LEVEL	FEATURE	NOTES	CLOUD VM/AA	EXUMO
VM	Service Features	Support 3rd-party PBX		N	Y
VM	Service Features	Support Skype for Business Server		Q1CY19	Y
VM	Service Features	Support Microsoft Teams		Y	N
VM	Service Features	eDiscovery and Hold	For security and compliance	Y	Y
VM	Service Features	Exchange Rules support	For security and compliance	Y	Y
VM	User Features	PSTN Dial-in Access	Subscriber access	N	Y
VM	User Features	PSTN Outlook Voice Access	Subscriber access	N	Y
VM	User Features	Dial-in using an authenticated endpoint	Calling the voicemail service to listen to voice messages and change voicemail settings	Y	Y
VM	User Features	User setting to disable voicemail		Y	Y
VM	User Features	User setting to change the personal greeting		Y	Y
VM	User Features	User setting to create an OOF greeting		Y	Y
VM	User Features	User setting to change the default language		Y	Y
VM	User Features	User setting to overwrite default greeting with TTS		Y	N
VM	User Features	Record personal greetings (authenticated device)		Y	Y
VM	User Features	Record personal greetings (PSTN) - play on phone		N	Y
VM	User Features	User setting to disable transcription		N	Y
VM	User Features	Transcription		Y	Y
VM	User Features	Visual voicemail on all endpoints	With user control to play, delete, message waiting indicator, and status-toggle, on all supported endpoints	Y	Y
VM	User Features	MP3 audio file format in Outlook		Y	Y
VM	User Features	Variable speed play control		Y	Y
VM	User Features	Forward a voicemail	Forward a received voicemail to other users	Y	Y
VM	User Features	Sending a voice message to a group of users	Voicemail broadcast	N	Y
VM	User Features	Voicemail notification using SMS	Users can receive an SMS when they have a new voicemail	N	Y
VM	User Features	Supported greeting languages	Details here: https://docs.microsoft.com/en-us/microsoftteams/what-are-phone-system-auto-attendants	Y	Y
VM	User Features	Call answering rules		Q1CY19	Y
VM	User Features	Play on phone (PSTN)- to play message	Call me on my cell to listen to the voice message	N	Y
VM	User Features	Play on phone (Auth)- to play message	Call me on my authenticated device	Y	Y
VM	User Features	Shared mailbox between multiple users		Y	Y

SERVICE	FEATURE LEVEL	FEATURE	NOTES	CLOUD VM/AA	EXUMO
VM	Caller Features	Caller experience - protected voicemail	The caller can choose an option to mark a recorded message as protected	N	Y
VM	Caller Features	Caller experience - private voicemail	The caller can choose an option to mark a recorded message as private	N	Y
VM	Caller Features	Silence detection		N	Y
VM	Tenant-Admin Features	Server-level protected voicemail	Tenant-admin can configure a service-level rule to mark incoming voicemail as protected	Y	Y
VM	Tenant-Admin Features	Change recording duration time limit	CVM hard coded to 5 minutes	N	Y
VM	Tenant-Admin Features	Change silence detection timeout		N/A	Y
VM	Tenant-Admin Features	Change number of input failure	CVM: hard coded to 3	N	Y
VM	Tenant-Admin Features	Change the default language		Y	Y
VM	Tenant-Admin Features	Disable/enable transcription		Y	Y
VM	Tenant-Admin Features	Disable/enable missed call notification		N	Y
VM	Tenant-Admin Features	Help Microsoft improve voice mail preview		Y	Y
VM	Tenant-Admin Features	Customize text message for enabled users		N/A	Y
VM	Tenant-Admin Features	Transcription profanity masking		Y	N
VM	Tenant-Admin Features	Voicemail policy		Y	Y
VM	Tenant-Admin Features	Web portal administration		CY19	Y
VM	Tenant-Admin Features	PowerShell		Y	Y
AA	Service Features	AA support 3rd-party PBX		N	Y
AA	Service Features	Support Skype for Business Server		Y	Y
AA	Service Features	Support Microsoft Teams		Y	N
AA	Service Features	Dial by name, DTMF input		Y	Y
AA	Service Features	Dial by name, speech input		Y	Y
AA	Service Features	Multi-language support	Language details here: https://docs.microsoft.com/en-us/microsoftteams/what-are-phone-system-auto-attendants	Y	Y

SERVICE	FEATURE LEVEL	FEATURE	NOTES	CLOUD VM/AA	EXUMO
AA	Service Features	Transfer to operator, CQ, or a user		Y	Y
AA	Service Features	Transfer to PSTN number internally (DID RNL)		Y	Y
AA	Service Features	Transfer to PSTN number externally		Q2CY19	Y
AA	Service Features	Business hours		Y	Y
AA	Service Features	Menu options	IVR menu options	Y	Y
AA	Service Features	Assigning an cloud PSTN number to AA		Y	N
AA	Service Features	Assigning an on-prem PSTN number to AA		Y	Y
AA	Service Features	Custom user selection	Enabling callers to reach customized list of organization users	Y	Y
AA	Service Features	After-hours and holidays treatment		Y	Y
AA	Service Features	Custom greeting using text to speech		Y	Y
AA	Service Features	Extension dialing	Reaching a user by dialing their extension	CY19	Y
AA	Service Features	Mailbox for AA callers to leave a message		CY19	Y
AA	Service Features	Multiple PSTN number assignment to an AA		Y	Y
AA	Tenant-Admin Features	Web portal administration		Y	N
AA	Tenant-Admin Features	PowerShell cmdlets		Y	Y
Fax	Service Features	Fax integration		N	Y

RODOLFO CASTRO AGUILAR
MVP Office Apps and Services
Twitter : @ucblogmx
http://ucblogmx.com

Mentoring

Comparti*MOSS*

Un servicio experto alrededor de su SharePoint

CompartiMOSS le puede ayudar a través de su programa de Mentoring!

Contacte con nosotros y le enviaremos los planes de mentoring que tenemos disponibles para SharePoint.

43

Trabajando con diseños de sitio en SharePoint Online

¿Qué son los diseños y scripts del sitio de SharePoint?

Un diseño de sitio SharePoint es un conjunto de acciones que se ejecutan en un sitio después de que éste ha sido creado. Con los diseños de sitio, por ejemplo, podemos crear listas, tipos de contenido, aplicar temas, etc.

Un script de sitio es un texto en formato JSON que contiene las acciones que vamos a ejecutar. Podemos pensar que un diseño de sitio es un contenedor de uno o varios scripts de sitio.

Como hemos visto, los diseños de sitio contienen acciones que se ejecutan una vez se ha creado el sitio, es decir, si tenemos un sitio en el que se aplicó un diseño y actualizamos ese diseño, el sitio no se actualizará automáticamente, sino que tendremos que volver a aplicar el diseño para que se apliquen los cambios.

Tampoco podemos añadir acciones antes de la creación del sitio, por ejemplo, incorporar un flujo de aprobación antes de la creación del sitio.

Existen diseños de sitio predeterminados dentro de SharePoint que podemos ver al crear un nuevo sitio moderno o también podremos crear nuestros diseños de sitio personalizados.

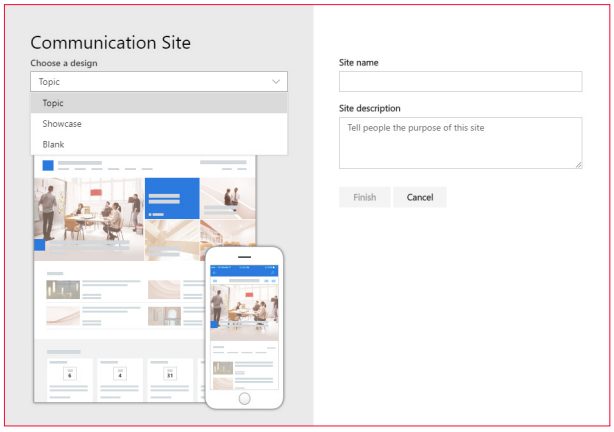


Imagen 1.- Diseños de sitio por defecto en SharePoint Online.

Creando un script de sitio

Los scripts de sitio son archivos JSON que especifican una lista ordenada de acciones que se ejecutarán al crear el sitio, es decir, las acciones se ejecutarán en el orden que indiquemos en el fichero JSON.

Un script de sitio puede contener varias acciones, que a su vez pueden contener subacciones, y cada acción viene definida por un valor de verb. Únicamente se pueden utilizar valores de verb que estén contenidas dentro del schema y si pones un valor que no esté disponible, se mostrará un bonito mensaje de error advirtiendo que esa acción no está disponible. Se irán añadiendo nuevos valores por lo que te aconsejo que revises que acciones están disponibles antes de crear el script de sitio. Puedes ver las acciones disponibles en https://docs.microsoft.com/en-us/sharepoint/dev/declarative-customization/site-design-json-schema

"Un diseño de sitio SharePoint es un conjunto de acciones que se ejecutan en un sitio después de que éste ha sido creado"

La estructura general de este fichero JSON es la siguiente:

```
{
  "$schema": "schema.json",
    "actions": [
      ...
      <one or more verb actions>
      ...
    ],
    "bindata": {},
    "version": 1
};
```

Dentro de cada acción indicaremos el valor de verb que queremos ejecutar y los parámetros necesarios para esa acción. Por ejemplo, podemos tener este script sencillo que aplica un tema y crea una lista:

```
{
 "$schema": "schema.json",
 "actions": [
  {
   "verb": "applyTheme",
   "themeName": "Tema Compartimoss"
  },
  {
   "verb": "createSPList",
   "listName": "Revistas",
   "templateType": 100,
```

```
    "subactions": [
     {
      "fieldType": "Number",
      "displayName": "Numero",
      "internalName": "NumeroRevista",
      "addToDefaultView": true,
      "isRequired": true,
      "verb": "addSPField"
     },
     {
      "verb": "setTitle",
      "title": "Revistas"
     }
    ]
   }
  ],
  "bindata": {},
  "version": 1
}
```

Puedes crear el fichero JSON directamente a mano, pero existen varias herramientas que con una interfaz gráfica que pueden ayudarte a crear estos ficheros. Una de estas herramientas es sitedesigner.io. Desde esta herramienta podrás ir añadiendo las acciones que necesites a tu script de sitio de una forma visual y luego podrás exportar ese script a un fichero JSON o incluso da la posibilidad de generar el código PowerShell necesario para desplegar el script a tu tenant.

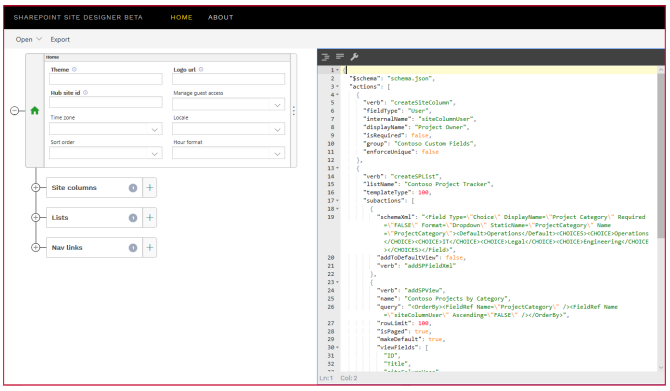


Imagen 2.- Herramienta SharePoint Site Designer.

"Una vez que hayamos creado el sitio se nos muestra un mensaje diciendo que se está aplicando el diseño"

Exportar una lista existente a un script de sitio

Recientemente, Microsoft liberó una nueva funcionalidad para poder generar un script para una lista que tengamos en un sitio. Para ello utilizaremos el comando de PowerShell Get-SPOSiteScriptFromList. El siguiente script de PowerShell mostrará en pantalla el script de la lista que indiquemos:

```
$adminSiteUrl = "https://tenant-admin.sharepoint.com"
$listUrl = "https:// tenant.sharepoint.com/sites/site/Lists/list-name"

$cred = Get-Credential

Connect-SPOService $adminSiteUrl -Credential $cred

Get-SPOSiteScriptFromList -ListUrl $listUrl
```

Desplegando diseños y scripts de sitios

Una vez que hayamos creado nuestro script tendremos que desplegarlo a nuestro tenant. Tenemos varias opciones para hacer el despliegue: se pueden desplegar con PowerShell o a través de REST API o CSOM. Con el siguiente script desplegamos el script que hemos creado de ejemplo a nuestro tenant y creamos un diseño de sitio que contiene el script:

```
$adminSiteUrl = "https://tenant-admin.sharepoint.com"

$script = @'
{
 "$schema": "schema.json",
 "actions": [
  {
   "verb": "applyTheme",
   "themeName": "Tema Compartimoss"
  },
  {
   "verb": "createSPList",
   "listName": "Revistas",
   "templateType": 100,
   "subactions": [
    {
     "fieldType": "Number",
     "displayName": "Numero",
     "internalName": "NumeroRevista",
     "addToDefaultView": true,
     "isRequired": true,
     "verb": "addSPField"
    },
    {
     "verb": "setTitle",
     "title": "Revistas"
    }
   ]
  }
 ],
 "bindata": {},
 "version": 1
}
'@

$webTemplate = "64"
$siteScriptTitle = "Custom Team Site Script"
$siteDesignTitle = "Custom Team Site Design"

$cred = Get-Credential

Connect-SPOService $adminSiteUrl -Credential $cred

$siteScript = Add-SPOSiteScript -Title $siteScriptTitle -Content $script

Add-SPOSiteDesign -Title $siteDesignTitle -WebTemplate $webTemplate -SiteScripts $siteScript
```

Dentro del script vemos que tenemos un parámetro webTemplate. Este parámetro sirve para indicar en qué tipo de plantilla vamos a mostrar nuestro diseño cuando creamos una colección de sitios desde la interfaz de usuario. Este parámetro puede tener los siguientes valores:

- 64: Sitios de equipo.
- 68: Sitios de comunicación.

Lo que realmente hace el parámetro webTemplate es asociar el diseño de sitio en el desplegable de la interfaz de usuario, pero se puede asignar cualquier diseño de sitio a cualquier tipo de colección de sitios programáticamente. Es más, si en ese parámetro ponemos un valor distinto a los dos valores que he mencionado, el diseño de sitio estará oculto en la interfaz de usuario, pero aun así se podrá asignar a una colección de sitios vía script de PowerShell o REST.

Administrando los diseños de sitio con PowerShell

Existen varios comandos de PowerShell que nos pueden resultar útiles a la hora de administrar los diseños y scripts de sitio que tengamos desplegados en nuestro tenant. Los diseños y scripts de sitio se despliegan a nivel de tenant por lo que para poder ejecutar los comandos tendremos que estar conectados al sitio de administración de SharePoint:

- Obtener información de diseños y script de sitio: Podemos obtener información de los diseños y script de sitio desplegados en nuestro tenant con los comandos:

```
Get-SPOSiteDesign
Get-SPOSiteScript
```

- Modificar un diseño de sitio:

```
Set-SPOSiteDesign -Identity $siteDesignId -SiteScripts $siteScriptId
```

- Modificar un script de sitio:

```
Set-SPOSiteScript -Identity $siteScriptId -Content $scriptContent
```

- Eliminar un diseño de sitio:

```
Remove-SPOSiteDesign -Identity $siteDesignId
```

- Eliminar un script de sitio:

```
Remove-SPOSiteScript -Identity $siteScriptId
```

Aplicando diseños de sitios

Tenemos varias formas para aplicar un diseño de sitio, una de ellas es mediante la interfaz de usuario. Al crear una nueva colección de sitios y seleccionar el tipo de sitio que hemos incluido en el parámetro webTemplate de nuestro script, veremos que nuestro diseño de sitio está disponible en un desplegable:

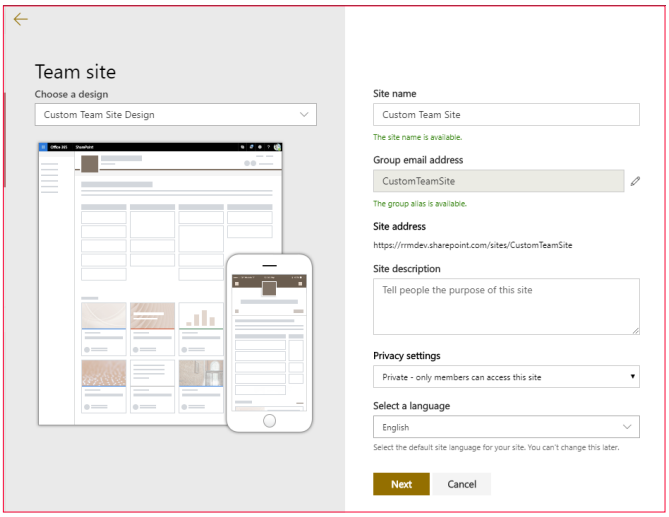


Imagen 3.- Diseño de sitio personalizado.

Una vez que hayamos creado el sitio se nos muestra un mensaje diciendo que se está aplicando el diseño, incluso podremos ver el progreso de la ejecución.

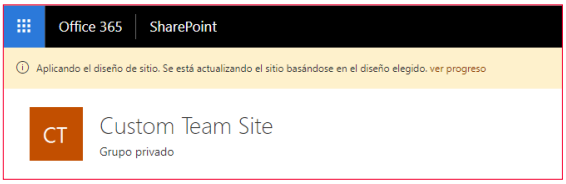


Imagen 4.- Mensaje de aplicación del diseño.

En el caso que cualquiera de las acciones que hayamos incluido en nuestro diseño diese un error, el script sigue ejecutándose hasta completar su ejecución.

Desde la interfaz de usuario también podemos aplicar un diseño de sitio a un sitio que hayamos creado anteriormente. Para ello vamos a la rueda de configuración del sitio y pulsamos sobre Diseños de sitios

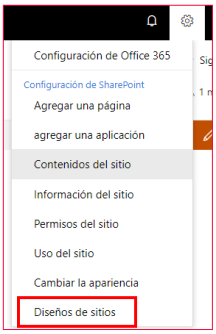


Imagen 5.- Acceso a los diseños de sitios desde el icono de configuración del sitio.

Se nos mostrará un panel lateral con todas las ejecuciones de diseños de sitios que hayamos hecho, junto al resultado de cada una de las ejecuciones:

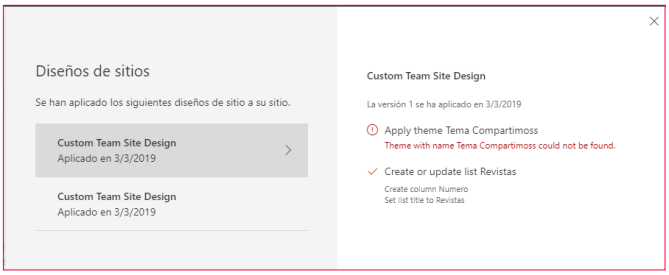


Imagen 6.- Panel con el detalle de los diseños de sitios que se hayan aplicado.

También veremos un enlace para ver todos los diseños de sitio disponibles donde podremos seleccionar otro diseño de sitio para aplicar.

Igual que aplicamos el diseño de sitio vía interfaz de usuario, podemos hacerlo mediante PowerShell, para ello utilizamos el comando Invoke-SPOSiteDesign de la siguiente manera:

```
Invoke-SPOSiteDesign -Identity <siteDesignId> -WebUrl https://tenant.sharepoint.com/sites/CustomTeamSite
```

Una vez ejecutado mostrará un resumen de las acciones ejecutadas y su resultado:

```
Title                          OutcomeText                                          Outcome
-----                          -----------                                          -------
Apply theme Tema Compartimoss  Theme with name Tema Compartimoss could not be found. Failure
Create or update list Revistas List with name Revistas already exists.               NoOp
Create column Numero                                                                 Success
Set list title to Revistas                                                           Success
```

Imagen 7.- Resultado de la aplicación de un diseño de sitio con PowerShell.

Permisos en diseños de sitios

Todos los diseños de sitios son públicos y visibles para todos por defecto, pero puede ocurrir que queramos limitar el uso de nuestro diseño a ciertos usuarios o grupos de seguridad.

Para poder hacer esto tendremos que hacer uso del comando de PowerShell Grant-SPOSiteDesignRights de la siguiente manera:

```
$adminSiteUrl = "https://tenant-admin.sharepoint.com"
$siteDesignId = "<siteDesignId>"
$principals = "<securityGroup>", "<user>"

$cred = Get-Credential
Connect-SPOService $adminSiteUrl -Credential $cred

Grant-SPOSiteDesignRights -Identity $siteDesignId -Principals $principals -Rights View
```

Podemos dar permiso tanto a grupos de seguridad como a usuarios concretos y podremos asignar a varios separándolos por una coma.

El valor del parámetro Rights siempre debe ser View.

En el caso que queramos quitar los permisos a ciertos usuarios para usar nuestro diseño de sitio utilizaremos el comando Revoke-SPOSiteDesignRights:

```
Revoke-SPOSiteDesignRights -Identity $siteDesignId -Principals $principals
```

Siempre podremos ver los permisos que tiene asignado un diseño de sitio ejecutando el comando:

```
Get-SPOSiteDesignRights -Identity <siteDesignId>
```

Diseños de sitio y Hub sites

Los diseños de sitio son un gran complemento a los Hub Sites ya que podemos hacer que en todos los sitios que creemos dentro de un Hub se ejecute nuestro diseño de sitio. Para poder asociar un diseño de sitio a un Hub Site, podemos hacerlo desde la interfaz de usuario o vía PowerShell.

Para asociar nuestro diseño de sitio a un Hub desde la interfaz de usuario, únicamente tendremos que ir a nuestro Hub y acceder a la Configuración del sitio concentrador:

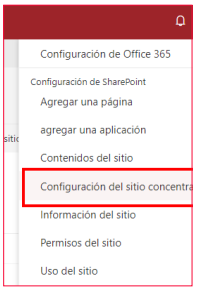


Imagen 8.- Acceso a la configuración del Hub.

Se nos abrirá un panel lateral donde podremos seleccionar el diseño de sitio asociado a ese Hub:

Imagen 9.- Selección del diseño de sitio a aplicar en el Hub.

También podemos asociar el diseño de sitio a través de PowerShell, para ello lo haríamos con el siguiente script:

```
$hubSiteUrl = "https://<tenant>.sharepoint.com/sites/<hubSite>"
$siteDesignId = "<siteDesignId>"

$adminSiteUrl = "https://<tenant>-admin.sharepoint.com"
Connect-SPOService $adminSiteUrl

Set-SPOHubSite $hubSiteUrl -SiteDesignId $siteDesignId
```

Una vez hayamos asignado un diseño de sitio a un Hub, todos los sitios que creemos dentro de ese hub se le aplicará nuestro diseño de sitio.

Otra de las opciones que tenemos para integrar diseños de sitios y Hub Sites, es la posibilidad de unir un sitio que estemos creando a un Hub desde el propio diseño de sitio. Dentro del script de sitio incluido en nuestro diseño podemos añadir una acción joinHubSite donde le indicamos el Hub donde vamos a asociar nuestro sitio. De esta forma todos los sitios que generemos con este diseño se asociarán

directamente al Hub que hayamos indicado.

```
{
 "$schema": "schema.json",
 "actions": [
  {
   "verb": "joinHubSite",
   "hubSiteId": "<hubSiteId>"
  }
 ],
 "bindata": {},
 "version": 1
}
```

Diseños de sitio y motor de aprovisionamiento remoto de PnP

Con lo que hemos visto podemos pensar que los diseños y scripts de sitio pueden ser un sustituto del motor de aprovisionamiento de PnP, pero más bien yo lo veo como un complemento. Por ejemplo, aún hay cosas que no podemos hacer con diseños de sitio y sí se puede con plantillas de PnP o incluso si queremos ejecutar una plantilla PnP que ya tengamos creada dentro de la experiencia estándar de creación de sitios en SharePoint Online.

Una de las acciones que podemos incluir dentro de nuestro script de sitio es la posibilidad de llamar a Flow, de esta forma podemos ejecutar una plantilla de PnP dentro de la ejecución de nuestro script de sitio

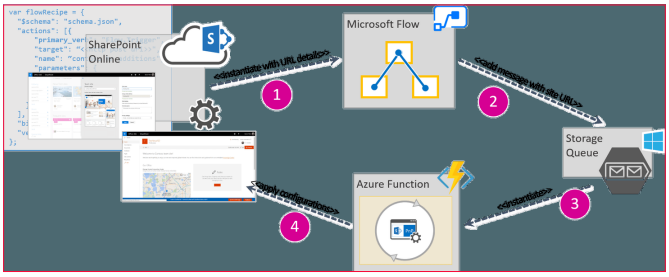


Imagen 10.- Patrón de llamada a un Flow en un Script de Sitio para aplicar una plantilla de PnP.

Para llamar a Flow desde nuestro script de sitio debemos incluir una acción de esta forma:

```
{
  "$schema": "schema.json",
  "actions": [
  {
      "verb": "triggerFlow",
      "url": "[paste the workflow trigger URL here]",
      "name": "Call Flow",
      "parameters": {
        "event":"",
        "product":""
      }
  }
  ],
  "bindata": {},
  "version": 1
}
```

Si quieres ver un tutorial de todos los pasos para realizarlo puedes verlo en https://github.com/SharePoint/sp-dev-docs/blob/master/docs/declarative-customization/site-design-pnp-provisioning.md

Otra posibilidad de integración entre los diseños de sitio y el aprovisionamiento con plantillas de PnP, es la posibilidad de desplegar diseños y scripts de sitio dentro de nuestra plantilla, incluso la de crear colecciones de sitios con esas plantillas.

Dentro del nodo Tenant de nuestra plantilla podemos declarar nuestro script de sitio siguiendo este esquema:

```
<pnp:SiteScript
   Title="xsd:string"
   Description="xsd:string"
   JsonFilePath="xsd:string"
   Overwrite="xsd:boolean">
</pnp:SiteScript>
```

También podemos definir nuestros diseños de sitio asociándoles los scripts de sitio que necesitemos, o incluso limitar los usuarios que queremos que puedan utilizar nuestro diseño de la misma forma que hemos visto anteriormente.

```
<pnp:SiteDesign
   Title="xsd:string"
   Description="xsd:string"
   IsDefault="xsd:boolean"
   WebTemplate=""
   PreviewImageUrl="xsd:string"
   PreviewImageAltText="xsd:string"
   Overwrite="xsd:boolean">
 <pnp:SiteScripts />
 <pnp:Grants />
</pnp:SiteDesign>
```

Conclusión

Los diseños de sitio nos proporcionan otra posibilidad a la hora de personalizar la creación de nuestras colecciones de sitio. A los usuarios les permite crear colecciones de sitio con una plantilla que hayamos definido dentro del proceso de creación de sitios Out Of the Box de SharePoint y a los administradores les permite administrar que diseños están disponible en el tenant y que grupo de usuarios pueden utilizar esos diseños. Todo ello con menos desarrollo que con las opciones que teníamos anteriormente.

Microsoft sigue trabajando para incorporar nuevas acciones que podamos ejecutar dentro de un script de sitio. También tenemos limitado el número de acciones que podemos incluir en un diseño de sitio, actualmente 300. Por todo esto la combinación de diseños y scripts de sitio junto con el motor de aprovisionamiento remoto de PnP me parece una muy buena opción a la hora de aprovisionar nuestro sitio dentro de SharePoint Online.

RUBÉN RAMOS MATEO
Technical Architect en Ricoh
ruben_rm@outlook.com
@rubenr79
https://rubenrm.com/

48

Azure Dev Spaces, desarrollando con Kubernetes y en equipo

Bajando a nuestro local la arquitectura de microservicios en AKS

¿Microservicios o monolítico? ¿Desplegamos en contenedores nuestras aplicaciones?¿Como los orquestamos?, etc. Si este tipo de preguntas no te dicen mucho no te preocupes, la intención de este artículo es conocer las herramientas que nos van a poner un pelín más fácil ir a una arquitectura de microservicios desplegados en contenedores en Azure.

Concepto contenedor, Docker

El primer concepto que nos debería ir sonando es el concepto "Contendor". Vamos a entender por contendor el empaquetado de un software y sus dependencias, que nos va a permitir aislar unas aplicaciones de otras.

> ***"los servicios que ejecutamos en Kubernetes, se agrupan en PODS, que se entiende como la unidad mínima de computo en Kubernetes"***

Docker es un proyecto Open Source, que nos va a permitir construir, almacenar y ejecutar aplicaciones distribuidas en contenedores. Se van a ejecutar en la mayoría de las distribuciones de Linux, Windows y Mac Os, usando "Docker Engine".

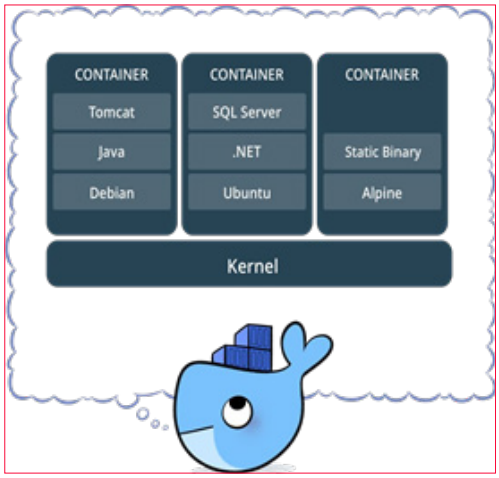


Imagen 1.- Elementos de Docker.

Para terminar, hay que aclarar que todos los contenedores comparten el mismo Kernel, y que no debemos confundir una maquina virtual con un contenedor de Docker, ya que mientras que en una maquina virtual necesitamos empaquetar el sistema operativo, en este caso el contenedor es independiente del mismo.

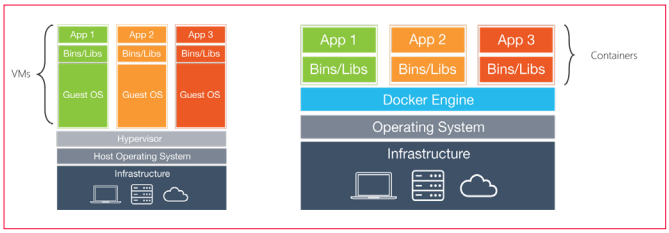


Imagen 2.- VMs vs Docker.

Para definir nuestro contenedor necesitaremos un fichero .Dockerfile, que nos permita empaquetar las aplicaciones necesarias en nuestra imagen, tal y como veremos un poco más adelante en el artículo.

Orquestando contenedores con AKS, Kubernetes en Azure

Kubernetes es el orquestador de contenedores, líder en la industria, y en Azure lo encontramos como el servicio Azure Kubernetes Service(AKS). Antes de empezar a trabajar con AKS, debemos tener algún que otro concepto de Kubernetes claro.

Kubernetes nos va a permitir escalar en un sistema de nodos de trabajo las ejecuciones de nuestros contenedores de Docker, aunque tenemos que tener en cuenta que no se mantiene la persistencia entre nodos de forma nativa.

Si profundizamos un poco más en Kubernetes vemos que los servicios que ejecutamos en Kubernetes, se agrupan en PODS, que se entiende como la unidad mínima de computo en Kubernetes, y no es más que la agrupación de contenedores que comparten una misma IP, almacenamiento y variables de entorno.

Si desplegamos dentro de un servicio o POD, varios contenedores, estos se podrán llamar entre ellos por IP interna, esto nos puede dar una pista de intentar juntar en un mismo Pods aquellas aplicaciones o sistemas que tengan relación.

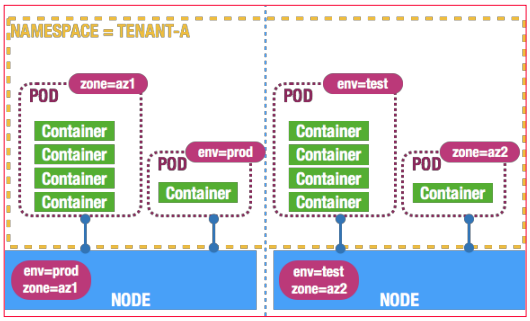


Imagen 3.- PODS.

Volviendo a AKS, no deja de ser un clúster de Kubernetes administrado en Azure que nos asegura alta disponibilidad como el resto de los servicios PaaS, es configurable el escalado y nos aporta una serie de herramientas a nivel de monitorización. En resumen, es una forma fácil de crear un clúster de Kubernetes, y abstraernos del proceso de configuración de un clúster de este tipo.

Imagen 4.- Kubernetes / AKS

¿Qué nos aporta Azure Dev Space?

Hasta aquí una brevísima introducción a conceptos necesarios, para que de verdad entendamos que es Azure Dev Space.

Uno de los mayores problemas del desarrollo Software es coordinar a todos los miembros de un equipo, y que podamos evolucionar de forma correcta una aplicación y su código fuente. A pesar de tener el código muy controlado gracias a herramientas como Azure Dev Ops y a GitHub, no es más cierto que seguimos con claros problemas por ejemplo con el versionado de una aplicación, o con equipos que tengan muy diferenciadas las competencias de Front-end y Backend, en ocasiones obligando a implementar entornos de integración bastante costosos para que todo el equipo pueda probar sus desarrollos.

Azure Dev Spaces, nace precisamente para cubrir esta necesidad, y no es otra que facilitar el desarrollo de aplicaciones en equipo, con la "particularidad" de que por debajo lo que hace este servicio es implementar un clúster de Kubernetes de AKS, y nos permite depurar en local este mismo. De esta forma podemos trabajar en equipo, y trabajar con una arquitectura basada en contendores y Kubernetes, pero desde un entorno local.

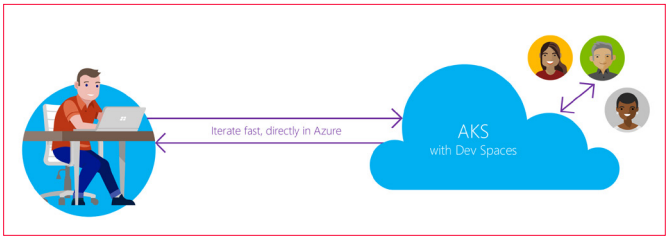


Imagen 5.- Azure Dev Space.

Cuando depuramos nuestra aplicación en Azure Dev Space, estamos realmente generando un contenedor de nuestra aplicación, y desplegándola en un AKS de nuestra suscripción. Además, se nos crea una conexión SSH que nos permite depurar en tiempo real nuestra aplicación desde nuestro Visual Studio local.

Cada miembro del equipo puede crear un espacio de trabajo que herede de una rama principal tal y cual haríamos en un repositorio de código fuente. Si creamos un espacio que herede de DEV podremos versionar nuestro Backend o nuestro Front, teniendo disponibles ambos servicios en un contenedor propio para el usuario, permitiendo hacer pruebas de forma aislada con una versión nueva, y si procede integrarlas en el entorno principal de DEV.

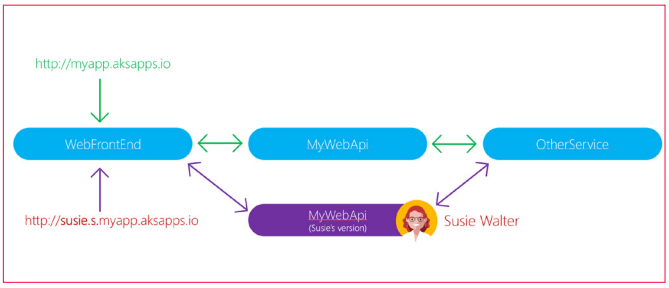


Imagen 6.- Diagrama con espacios de trabajo

Como vemos en la imagen anterior, siempre que creemos un espacio de nombres que herede de un principal como DEV, tendremos en tiempo real dos URL'S. Una la de nuestro espacio de trabajo personal como el del ejemplo de Susie, y otro principal.

- Url entorno principal: myapp.aksapp.io
- Url entorno de trabajo personal: mydevspace.myapp.aksapp.io

> ***"Azure Dev Spaces nace para facilitar el desarrollo de aplicaciones en equipo"***

La ventaja de este modelo es que por un lado una persona que solo trabaje con el proyecto de front, puede hacer uso de las APIS que ya estaban desplegadas en DEV, y versionar su versión de front, y viceversa. Además, podemos invocar ambas url's, y visualizar los cambios y poder hacer un QA mucho más preciso, sin necesidad de hacer un despliegue por cada cambio.

Creando nuestra primera aplicación .Net Core en Azure Dev Space

Vamos a hacer un ejemplo muy sencillo de como implementar una aplicación .Net Core con Visual Studio en Dev Space.Tenemos que tener claro que antes de empezar necesitamos crear un clúster de Kubernetes administrador en Azure con AKS. Existen algunas limitaciones para usar Azure Dev Space:

- El clúster de Kubernetes debe ejecutar la versión de Kubernetes 1.9.6 o superior.
- Esta disponible únicamente en las regiones EastUS, EastUS2, CentralUS, WebUS2, WestEurope, SoutheastAsia, CanadaCentral o Canada East.
- Necesitamos activar el Enrutamiento por HTTP.

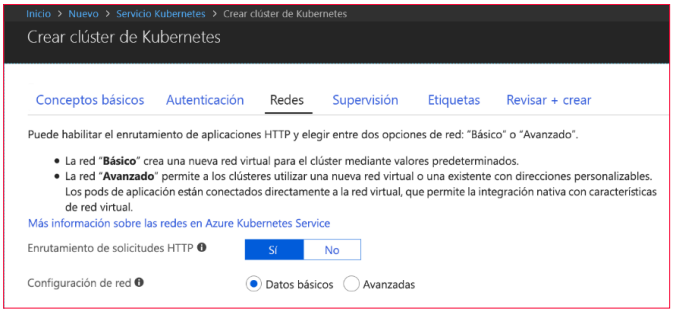


Imagen 7.- Portal Azure AKS.

Además, necesitamos instalar las Tool de Kubernetes para Visual Studio, para ello desde el propio Visual Studio 2017, podemos seleccionar Tools->Extensions & Updates, y buscamos "Visual Studio Tools for Kubernetes", y si no lo tenemos lo instalamos.

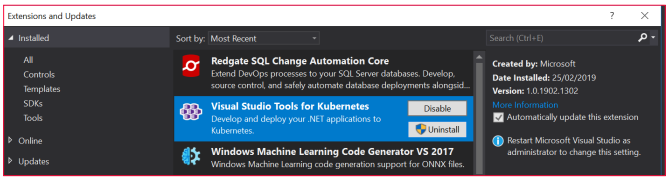


Imagen 8.- Tools Kubernetes VS.

Una vez tenemos nuestro cluster y las Tools de Kubernetes, nos creamos una aplicación .Net Core 2.0, nos vale una aplicación MVC básica para dar forma al ejemplo.

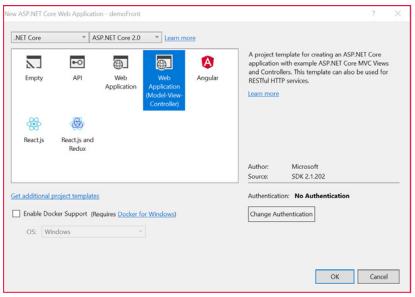


Imagen 9.- Aplicación MVC.

Una vez creado el proyecto y la solución, vamos a desplegarla en nuestro clúster de AKS con Azure Dev Space. En el desplegable de depuración, seleccionamos Azure Dev Space, y se lanzará una ventana de conexión a nuestra suscripción de Azure.

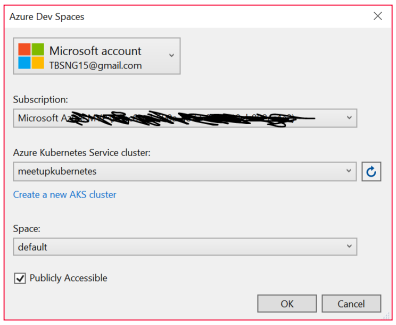


Imagen 10.- Configurando Azure Dev Space.

Deberemos seleccionar la suscripción donde desplegamos el servicio de AKS, seleccionar el propio clúster de AKS, y elegir un espacio de desarrollo. De momento solo tenemos el "default" que se crea de base, y es el que elegiremos.

Una vez aceptemos, se generará en nuestra solución varios ficheros, pero nos vamos a fijar en dos:

- DockerFile: contiene la información para generar nuestro contenedor Docker, y que desplegaremos en AKS.
- Azds.yaml: Contiene la configuración necesaria para implementar el espacio de desarrollo de Azure Dev Space.

Si lanzamos la depuración de la aplicación, veremos que se genera una url del tipo demoFront.aksapp.io que apunta a nuestro despliegue en AKS, y si ponemos un punto de depuración en por ejemplo en el HomeController, la ejecución se detiene en local.

Esta aplicación ya está desplegada en AKS, en un contenedor de Docker que Azure Dev Space ha definido por nosotros, y accesible por IP Pública por cualquier miembro de nuestro equipo de desarrollo.

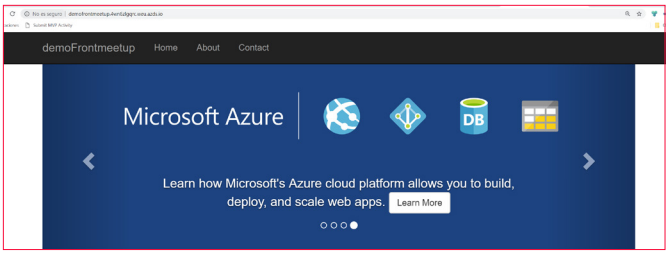


Imagen 11.- App desplegada en AKS.

Llamando de un contenedor a otro, Front y Backend

Ya tenemos nuestro Front desplegado en AKS, pero nunca tenemos un escenario tan sencillo de ejecución, normalmente también necesitamos implementar un catalogo de servicios en Backend.

Para ello vamos a crear una aplicación también .Net Core del tipo Web API que llamaremos demoBack, y la vamos a configurar en el mismo espacio de trabajo default, tal y como hicimos en el ejemplo anterior.

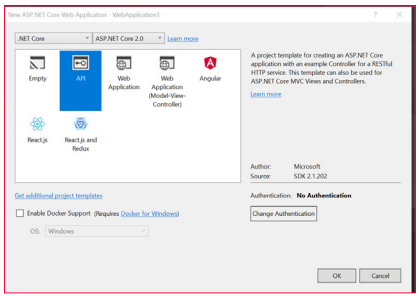


Imagen 12.- Creando un API Net Core.

Si lanzamos esta aplicación tendremos una url del tipo demoback.aksapp.io, accesible del mismo modo que nuestra app de front.

¿Cómo comunicamos nuestro contenedor de front con el de back?. Para ello vamos a usar las "cabeceras de enrutamiento" que nos proporciona Azure Dev Space.

Si vamos a nuestro HomeController, en el método que invoca la vista "About" básica que nos crea el ejemplo, vamos a añadir una petición HTTP con el siguiente código C#:

```
using (var client = new System.Net.Http.HttpClient())
        {
          // Call *mywebapi*, and display its response in the page
            var request = new System.Net.Http.HttpRequestMes-
sage();
             request.RequestUri = new Uri("http://demoback/api/
values/1");
          if (this.Request.Headers.ContainsKey("azds-route-as"))
           {
             // Propagate the dev space routing header
             request.Headers.Add("azds-route-as", this.Request.
Headers["azds-route-as"] as IEnumerable<string>);
           }
          var response = await client.SendAsync(request);
          ViewData["Message"] += " and " + await response.Con-
tent.ReadAsStringAsync();
        }
```

Es importante entender dos líneas, ya que el código no tiene misterio simplemente hace una petición Get al API Controller Values que hemos creado con el proyecto de API Backend.

Lo particular del ejemplo es:

- Añadimos una cabecera a la petición del tipo "azds-rout-as" que nos va a permitir conectar y enrutar las peticiones entre nuestros contenedores.
- La url que indicamos con "new Uri("http://demoback/api/values/1");" es del tipo http://myappbackend/api/values/1", es decir este código nos va a permitir como veremos en el siguiente ejemplo, que si creamos un nuevo espacio de trabajo que herede de este default, que siga funcionando esta petición. Es decir, el enrutamiento de Azure Dev Space nos permite que apuntando a la aplicación base de AKS, en este caso de nuestro backend, se vaya enrutando en función de si estamos en un espacio de trabajo u otro, sin necesidad de modificar este uri endpoint.

Si debugamos este código podremos poner un brakepoint en el api, y ver que desde la vista de About del front se invoca correctamente al backend de nuestro espacio de trabajo default.

"Podremos ver en la url generada para mi nuevo espacio de trabajo shernandez.s.demofront.aksapp.io, que me devuelve mi nueva versión del Api"

Creando espacios de trabajo

Vamos a probar que nuestro código de enrutado por cabecera sigue funcionando cuando creamos un nuevo espacio de desarrollo. Para ello debemos seleccionar en las propiedades del proyecto tanto de front, como de backend la pestaña debug.

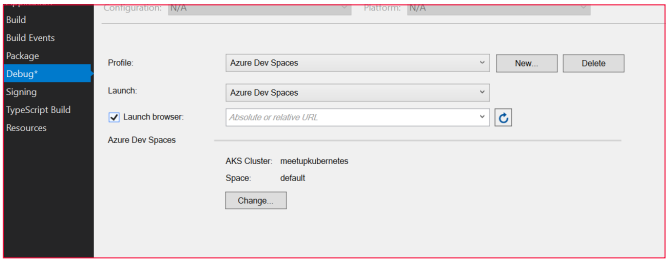


Imagen 13.- Creando un espacio de desarrollo.

Si seleccionamos en Change, se nos abre la misma ventana de conexión a nuestro clúster que vimos anteriormente, y en el desplegable de Space, seleccionamos "create new space". En la siguiente ventana, seleccionamos el espacio default (va a actuar como nuestro MASTER), y creamos un sitio de prueba para nosotros, en mi caso shernandez.

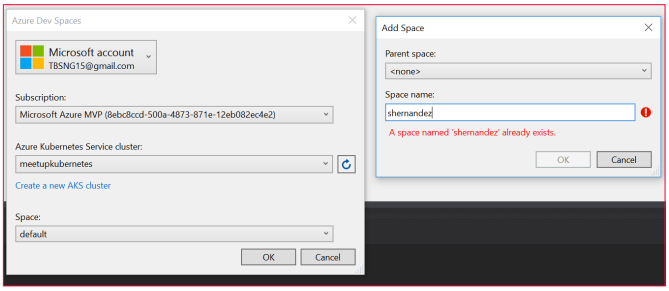


Imagen 14.- Mi espacio personal.

Si esto ha funcionado de forma correcta, se habrá creado un nuevo contenedor, que no es más que una versión del original de default, y desde este podemos evolucionarlo a nuestra demanda.

Si volvemos a depurar el api y el front, lanzándolo contra nuestro AKS con F5, podremos comprobar que el enrutado sigue funcionando, o lo que es mejor aun, si desplegamos con F5 y previamente hacemos un cambio en nuestro controller del tipo de la siguiente imagen, veremos que el api se ha modificado solo para el nuevo dev space creado "shernandez".

```
[HttpGet("{id}")]
0 references | 0 changes | 0 authors, 0 changes | 0 requests | 0 exceptions
public string Get(int id)
{
    return "value mi espacio mi sitio shernandez";
}
```

Imagen 15.- Cambio en el api.

Podremos ver en la url generada para mi nuevo espacio de trabajo shernandez.s.demofront.aksapp.io, que me devuelve mi nueva versión del api, en cambio en demofront.aksapp.io que corresponde a la url de front en el espacio default (espacio base), se devuelve el valor anterior.

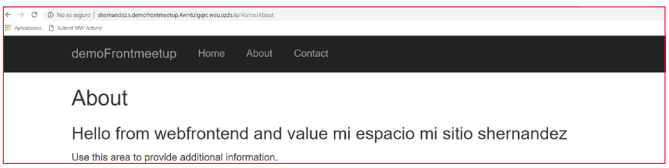


Imagen 16.- Resultado del nuevo api.

Por el contario en el espacio original default, tenemos el resultado original del API en la pantalla contacto.

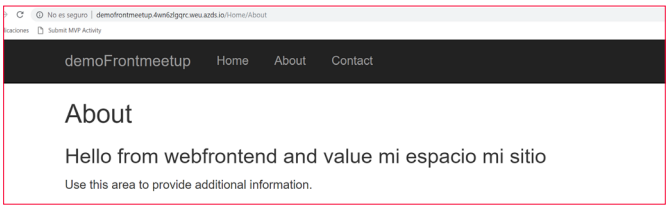


Imagen 17.- Api original.

Visual Studio es cómodo, pero es recomendable usar Azure CLI

Siguiendo los ejemplos anteriores, perfectamente podemos usar Visual Studio para crear y desplegar nuestros espacios de trabajo. Pero también es cierto, que yo al menos en mi experiencia personal, cuando depuramos las aplicaciones desde el Visual Studio, al detenerlas también se detienen los espacios, y esto si lo hacemos contra un espacio de trabajo base como DEV puede dar problemas a los miembros del equipo.

Una forma de poder manejar Azure Dev Space es por línea de comandos con Azure CLI.

Para ejecutarlo, simplemente abriendo una aplicación de consola una vez descargado el paquete de Azure CLI, podemos ejecutar los siguientes comandos:

Logarse contra suscripción de Azure

```
az login
```

Conectarse al clustes AKS vía Azure Dev Space

```
az aks use-dev-spaces -g AzureDevops -n Meetupkubernetes
```

Subir una ap a Azds (debemos situarnos el directorio con el yaml y el dockerfile a desplegar)

```
azds up
```

Seleccionar un espacio de trabajo

```
azds space select --name scott
```

Listas urls de los espacios de trabajo

```
azds list-uris
```

```
C:\Users\shernandez>azds list-uris
Uri                                                 Status
--------------------------------------------------  ---------
http://demobackmeetup.4wn6zlgqrc.weu.azds.io/       Available
http://demofrontmeetup.4wn6zlgqrc.weu.azds.io/      Available
http://webapplication2.4wn6zlgqrc.weu.azds.io/      Available
```

"necesitamos entender el concepto contenedor, el concepto clúster y orquestador como es Kubernetes, para poder ver todo el potencial de Azure Dev Space"

Conclusiones del uso de Azure Dev Spaces

Sin duda estamos ante un mundo apasionante este el de los microservicios, pero desde mi franqueza también debo avisar que el primer contacto es más que duro. Como hemos podido ver en este pequeño artículo, y desde un punto de vista muy superficial, necesitamos entender el concepto contenedor, el concepto clúster y orquestador como es Kubernetes, para poder ver todo el potencial de Azure Dev Space.

Para equipos maduros, que bien trabajen con soluciones en microservicios, o que tengan un escenario de despliegue por contenedores, este es un servicio más que recomendable. La principal virtud, es que, con los espacios de desarrollo, vamos a poder versionar y probar tanto nuestro backend como nuestro front sin necesidad de hacer infinidad de pull request y despliegues en integración.

Además, permite por ejemplo a un compañero de front, tener plataformado su entorno de trabajo, debugando su lado front, y teniendo una versión de las apis estables que sabemos no van a cambiar al menos hasta conectarse al espacio principal.

En resumen, muy interesante esta primera aproximación a un servicio que espero siga evolucionando.

SERGIO HERNÁNDEZ MANCEBO
Principal Team Leader en Encamina
Microsoft Azure MVP

53

Nosotros

Alberto Diaz

Alberto Díaz es SharePoint Team Lead en ENCAMINA, liderando el desarrollo de software con tecnología Microsoft.

Para la comunidad, ha fundado TenerifeDev (www.tenerifedev.com) con otros colaboradores, un grupo de usuarios de .NET en Tenerife, y coordinador de SUGES (Grupo de Usuarios de SharePoint de España, www.suges.es) y colaborador con otras comunidades de usuarios. Microsoft MVP de SharePoint Server desde el año 2011 y asiduo conferenciante en webcast y conferencias de tecnología de habla hispana.

Sitio Web: http://blogs.encamina.com/negocios-sharepoint/
Email: adiazcan@hotmail.com
Blogs: http://geeks.ms/blogs/adiazmartin
Twitter: @adiazcan

Fabián Imaz

Fabián Imaz, MVP de SharePoint Server trabaja en el mundo del desarrollo de software desde hace más de 10 años, teniendo la suerte de trabajar en distintas arquitecturas y tecnologías Microsoft. Pertenece a la firma Siderys, http://www.siderys.com empresa de desarrollo de Software especializada en SharePoint 2007/2010/2013 y en desarrollo de soluciones inteligentes.

Desde los comienzos Fabián ha trabajado en distintitas comunidades donde organiza y promueve eventos locales para la difusión de tecnología dentro de los miembros de las mismas. Es director de la carrera SharePoint 2010 y SharePoint 2013 en Microsoft Virtual Academy, http://www.mslatam.com/latam/technet/mva2/ Home.aspx y cuenta con un sitio en CodePlex con varios desarrollos http://siderys.codeplex.com.

Sitio Web: http://www.siderys.com
Email: fabiani@siderys.com.uy
Blogs: http://blog.siderys.com
Twitter: @fabianimaz

Gustavo Velez

Gustavo Velez es Ingeniero Mecánico y Electrónico; trabaja en la arquitectura, diseño e implementación de sistemas de IT basados en tecnologías de Microsoft, especialmente SharePoint, Office 365 y Azure.

Propietario del sitio especializado en información sobre SharePoint en español http://www. gavd.net, autor de ocho libros sobre SharePoint y sus tecnologías y numerosos articulos y conferencias sobre el tema.

Sitio Web: http://www.gavd.net
Email: gustavo@gavd.net
Blogs: http://geeks.ms/blogs/gvelez/

Juan Carlos González Martín

Ingeniero de Telecomunicaciones por la Universidad de Valladolid y Diplomado en Ciencias Empresariales por la Universidad Oberta de Catalunya (UOC). Cuenta con más de 14 años de experiencia en tecnologías y plataformas de Microsoft diversas (SQL Server, Visual Studio, .NET Framework, etc.), aunque su trabajo diario gira en torno a las plataformas SharePoint & Office 365. Juan Carlos es MVP de Office Apps & Services y co-fundador del Grupo de Usuarios de SharePoint de España (SUGES, www.suges.es), delGrupo de Usuarios de Cloud Computing de España (CLOUDES) y de la Comunidad de Office 365. Hasta la fecha, ha publicado 11 libros sobre SharePoint & Office 365, así como varios artículos encastellano y en inglés sobre ambas plataformas.

Email: jcgonzalezmartin1978@hotmail.com
Blogs: http://geeks.ms/blogs/jcgonzalez &
http://jcgonzalezmartin.wordpress.com/

Santiago Porras

Innovation Team Leader en ENCAMINA, lidera el desarrollo de productos mediante tecnologías Microsoft. Se declara un apasionado de la tecnología, destacando el desarrollo para dispositivos móviles y web, donde ya cuenta con 16 años de experiencia.

Microsoft MVP in Developer Technologies, colabora con las comunidades de desarrolladores desde su blog personal **http://geeks.ms/santypr** y ocasionalmente en **CompartiMOSS.com**. Además, es uno de los coordinadores de TenerifeDev, grupo de usuarios de .NET en Tenerife (**http://www.tenerifedev.com**)

Sitio Web: http://www.santiagoporras.es
Email: santiagoporras@outlook.com
Blogs: http://geeks.ms/santypr
Twitter: @saintwukong

Coordinadores de sección

GASTÓN CRUZ
Coordinador de PowerBi
gastoncruz@gmail.com

XAVIER MORERA
Coordinador de .Net
xavier@familiamorera.com

PABLO PERALTA
Coordinador de Dynamics CRM
pablop2006@gmail.com

¿Desea colaborar con CompartiMOSS?

La subsistencia del magazine depende de los aportes en contenido de todos. Por ser una revista dedicada a información sobre tecnologías de Microsoft en español, todo el contenido deberá ser directamente relacionado con Microsoft y escrito en castellano. No hay limitaciones sobre el tipo de articulo o contenido, lo mismo que sobre el tipo de tecnología.
Si desea publicar algo, por favor, utilice uno de los siguientes formatos:

- Artículos de fondo: tratan sobre un tema en profundidad. Normalmente entre 2000 y 3000 palabras y alrededor de 4 o 5 figuras. El tema puede ser puramente técnico, tanto de programación como sobre infraestructura, o sobre implementación o utilización.

- Artículos cortos: Artículos cortos: Máximo 1000 palabras y 1 o 2 figuras. Describen rápidamente una aplicación especial de alguna tecnología de Microsoft, o explica algún punto poco conocido o tratado. Experiencias de aplicación en empresas o instituciones puede ser un tipo de artículo ideal en esta categoría.

- Ideas, tips y trucos: Algunos cientos de palabras máximo. Experiencias sobre la utilización de tecnologías de Microsoft, problemas encontrados y como solucionarlos, ideas y trucos de utilización, etc. Los formatos son para darle una idea sobre cómo organizar su información, y son una manera para que los editores le den forma al magazine, pero no son obligatorios. Los artículos deben ser enviados en formato Word (.doc o .docx) con el nombre del autor y del artículo.

Si desea escribir un artículo de fondo o corto, preferiblemente envíe una proposición antes de escribirlo, indicando el tema, aproximada longitud y número de figuras. De esta manera evitaremos temas repetidos y permitirá planear el contenido de una forma efectiva.

Envíe sus proposiciones, artículos, ideas y comentarios a la siguiente dirección:
revista@compartimoss.com adiazcan@hotmail.com
fabiani@siderys.com.uy jcgonzalezmartin1978@hotmail.com
gustavo@gavd.net


Comparti
MOSS