Manejo de Azure IoT desde aplicaciones y scripts usando PowerShell y Visual Studio C#

Luis Garreta

Índice

Introducción

El manejo de Azure IoT involucra la creación, configuración, y eliminación de recursos y servicios en la nube. Este manejo se puede realizar desde los siguientes medios:

- Desde el portal web, la más común
- Comandos Powershell, solo para Windows
- Comandos Azure CLI, para Linux y otros sistemas
- Comandos web, a través de de la interfaz REST API
- Aplicaciones (Lenguaje C# o node.js).

El manejo más completo se realiza a través del primer medio: el portal web (`https://portal.azure.com`), el cual es totalmente manual. Los tres siguientes medios: Powershell, AzureCLI, y REST API, corresponden a comandos que se ejecutan ya sea a través de shells como Powershell para Windows o Azure CLI para Linux, o que se envían sobre la Internet a modo de solicitudes y respuestas HTTPS [2]. Estos son medios más versátiles con los cuales es posible crear scripts o pequeños programas que ejecutan una secuencia de comandos que pueden crearde forma automática diferentes servicios o recursos en Azure. Finalmente el último medio: aplicaciones, corresponde a programación en un lenguaje de computación, ya sea C# o node.js, que dan la versatilidad de los comandos anteriores pero con la ventaja de crear aplicaciones que puedan utilizar otros recursos para resolver tareas más complejas.

Otras formas de manejar los recursos de Azure es a través de herramientas. Para el manejo de los dispositivos dentro de un Hub de IoT previamente creado, Microsoft provee la herramienta "Device Explorer" (`https://github.com/Azure/azure-iot-sdks/blob/master/doc/manage_iot_hub.md`) que permite realizar operaciones fundamentales sobre estos (e.g. crear, eliminar, actualizar, listar, envíar/-recibir mensajes). Otra herramienta para el manejo de recursos es "Cloud Explorer" (`https://docs.microsoft.com/en-us/azure/vs-azure-tools-resources-managing-with-cloud-explorer`) que

se integra en Visual Studio y permite manejar muchos de los recursos de Azure como el manejo de cuentas de usuario y subscripciones, y el manejo de cuentas de almacenamiento, entre otras.

Adicionalmente, el portal de Azure ha lanzado ha lanzado un nuevo servicio para generar el esquema de un recurso previamente creado y configurado (`https://docs.microsoft.com/en-us/azure/azure-resource-manager/resource-manager-export-template`), lo cual permite que ese esquema se pueda utilizar para crear de forma automática recursos del mismo tipo con modificaciones específicas de acuerdo a las necesidades del problema (Más adelante, en la creación automática de Hubs de IoT se mostrará el uso de estos esquemas).

Finalmente, se debe tener en cuanta que para el caso de las aplicaciones, que permiten automatizar la creación y uso de recursos de Azure de forma programada (e.g. PowerShell, C#, o node.js), cada una de estas debería tener sus propias credenciales de autenticación, las cuales se pueden crear ya sea a través de certificados o claves *(passwords)* [1]. En este documento vamos a mostrar primero la forma de crearlas usando contraseñas en vez de certificados, lo cual es más apropiado cuando se va a iniciar sesión en Azure desde una aplicación. Y segundo, vamos a mostrar una aplicación en C# bajo Visual Studio que contiene una clase con métodos completamente parametrizados para el manejo de recursos de forma automática y segura. La clase tiene cuatro métodos: el primero permite obtener el token de autorización para manejar recurso de forma programada. El segundo permite crear grupos de recursos, dado un nombre y una localización. El tercero permite crear cuentas de almacenamiento. Y el cuarto usa un esquema cargado en una cuenta de almacenamiento para crear un Hub de IoT en un grupo de recuros previamente creado.

Disponibilidad del Código:

Todo el código que se presenta en este documento lo puede descargar desde:
`https://github.com/lgarreta/AzureIoT-ManageResourcesCSharp`

1. Creación de los IDs para Autenticación

El proceso de autenticación se realiza a través de la creación de dos objetos: el directorio activo o *Azure Active Directory (AAD)* y la entidad de aplicación o *service principal*. Después se asigna el role *(owner)* al *service principal*. Al final los valores que nos interesa guardar son los siguientes cuatro:

ApplicationId: Este es el ID para sus aplicación en su directorio activo.

Password: Este es la clave para su entidad de aplicación

TenantId: Este es el ID de su directorio activo.

SubscriptionId: El ID de su suscripción.

Estos pasos se realizan en Powershell y se muestran en el siguiente listado (los cuadros corresponden a las salidas resultantes):

Algoritmo 1 Pasos para el proceso de autenticación con Azure Active Directoy.

```
// 1. Ingresar a su subscripcion con su usuario y clave y guardar el TenantId y el SubscriptionId:
Login-AzureRmAccount
```

```
Environment           : AzureCloud
Account               : luisgpujiot@outlook.com
TenantId              : 2707f6c8-6127-4a28-858a-1bad498d78b8
SubscriptionId        : 933a5bc9-18cb-4cd3-800a-2a2fdd692fb6
SubscriptionName      : Visual Studio Enterprise: BizSpark
CurrentStorageAccount :
```

```
// 2. Crear un nuevo Azure Active Directory y guardar el valor del ApplicationId
New-AzureRmADApplication -DisplayName iothubapp -HomePage http://iothubapp/home -IdentifierUris
    http://iothubapp -Password iothub2017A
```

```
DisplayName             : iothubapp
ObjectId                : b330059c-d3b3-4d32-bc03-b898922c23c5
IdentifierUris          : {http://iothubapp}
HomePage                : http://iothubapp/home
Type                    : Application
ApplicationId           : 4c1f4d33-e87b-4395-b758-dbc0d4afe25f
AvailableToOtherTenants : False
AppPermissions          :
ReplyUrls               : {}
```

```
// 3. Crear un nuevo service principal para la aplicación usando el ApplicationId
New-AzureRmADServicePrincipal -ApplicationId 4c1f4d33-e87b-4395-b758-dbc0d4afe25f
```

```
DisplayName  Type             ObjectId
-----------  ----             --------
iothubapp    ServicePrincipal 5e33ff06-7905-4925-997b-4cda7a895ac4
```

```
// 4. Asigne un rol al anterior service usando el ApplicationId
New-AzureRmRoleAssignment -RoleDefinitionName Owner -ServicePrincipalName 4c1f4d33-e87b-4395-b758-
    dbc0d4afe25f
```

```
RoleAssignmentId   : /subscriptions/933a5bc9-18cb-4cd3-800a-2a2fdd692fb6/providers/Microsoft.
    Authorization/roleAssignments/91899326-499e-4fd3-b209-b8a055efc892
Scope              : /subscriptions/933a5bc9-18cb-4cd3-800a-2a2fdd692fb6
DisplayName        : iothubapp
SignInName         :
RoleDefinitionName : Owner
RoleDefinitionId   : 8e3af657-a8ff-443c-a75c-2fe8c4bcb635
ObjectId           : 5e33ff06-7905-4925-997b-4cda7a895ac4
ObjectType         : ServicePrincipal
```

2. Manejo de recursos de Azure mediante aplicaciones en C#

A continuación vamos a describir una aplicación en C# bajo Visual Studio que permite de forma programática crear y usar recursos de Azure. Para esto hemos construido una clase llamada *AzureIoT-Management* que contiene métodos para cuatro tareas muy comúnes a la hora de trabajar con los recursos de Azure para IoT, estos son:

- Autenticarse de forma segura a través de un token SAS
- Crear grupos de recursos
- Crear Hubs de IoT
- Crear cuentas de almacenamiento

Todos los métodos son completamente parametrizados, es decir la información necesaria para crear el recurso viene dada explicitamente en el llamado de los métodos, lo que permite que se puedan utilizar para múltiples proyectos. Algunos parámetros más generales que manejan los recursos están implicitos en en los métodos (e.g. tipo de almacenamiento) y el desarrollador puede cambiarlos de acuerdo a sus necesidades.

2.1. Proceso de Autenticación

Se siguen los mismos pasos descritos en la sección 1, los cuales se resumen en el siguiente script de PowerShell:

Algoritmo 2 Script PowerShell Proceso de Autenticación.

```
$login = Login-AzureRmAccount
$tenantId = $login.Context.Tenant.TenantId
$suscriptionId = $login.Context.Tenant.TenantId
$nombreAplicacion = "progiothubapp"
$clave = "iothub2017A"

$app = New-AzureRmADApplication -DisplayName $nombreAplicacion -HomePage "http://$nombreAplicacion/
    home" -IdentifierUris "http://$nombreAplicacion" -Password $clave
$serv = New-AzureRmADServicePrincipal -ApplicationId $app.ApplicationId
$role = New-AzureRmRoleAssignment -RoleDefinitionName Owner -ServicePrincipalName $app.
    ApplicationId

echo "TenantId: $tenantId"
echo "SubscriptionId: $suscriptionId"
echo "ApplicationId: $app.ApplicationId"
echo "Password: $clave"
```

Al final, deben anotarse los cuatro valores (TenantId, SubscriptionId, ApplicationId, and Password) que imprime al final ya que serán usados en la aplicación de C#.

2.2. Preparación del proyecto en Visual Studio

Para el proyecto se necesita configurar a Visual Studio para trabajar con Azure y adicionamente instalar (si no están ya instalados) los siguientes paquetes usando el manejador de paquetes de Visual Studio (Nuget Package Manager):

- Microsoft.Azure
- Microsoft.Azure.Management.ResourceManager
- Microsoft.IdentityModel.Clients.ActiveDirectory
- Microsoft.Azure.Management.Storage

Además, para cargar archivos a cuentas de almacenamiento en Azure necesita la herramienta de Visual Studio "cloud explorer".

2.3. Llamado de los métodos de la aplicación en C#

En el siguiente listado se presenta el método principal con el llamado de los métodos para manejar los distintos recursos de Azure. Lo primero es obtener el token de autorización, después se puede crear otros recursos como un grupo de recursos, una cuenta de almacenamiento, o un hub de IoT.

En este último caso, creación del Hub, se deben cargar previamente a una cuenta de almacenamiento dos archivos: uno para el esquema de Hub y el otro con los parámetros de creación (entre estos el nombre del Hub). Los detalles de cómo cargarlos se mostrarán en la sección que muestra el método de creación del Hub.

Algoritmo 3 Llamado de los métodos para manejo de recursos de Azure.

```
using System;
using Microsoft.Azure.Management.ResourceManager;
using Microsoft.Azure.Management.ResourceManager.Models;
using Microsoft.IdentityModel.Clients.ActiveDirectory;
using Microsoft.Rest;
using Microsoft.Azure.Management.Storage;
using Microsoft.Azure;
using Microsoft.Azure.Management.Storage.Models;

namespace CreateIoTHub {
  class AzureIoTManagement {
    //--------------------------------------------------------------------
    // Replace de corresponding values taken from the autentication process
    //--------------------------------------------------------------------
    static string tmpApplicationId = "4c1f4d33-e87b-4395-b758-dbc0d4afe25f";
    static string tmpSubscriptionId = "933a5bc9-18cb-4cd3-800a-2a2fdd692fb6";
    static string tmpTenantId = "2707f6c8-6127-4a28-858a-1bad498d78b8";
    static string tmpAppPassword = "iothub2017A";

    //--------------------------------------------------------------------
    // Temporal names for resources
    //--------------------------------------------------------------------
    static string tmpResourceGroupName = "progrg1";
    static string tmpResourceGroupLocation = "East US";
    static string tmpIoTHubName = "progiothub1";
    static string tmpDeployName = "progdeploy1";
    static string tmpStorageAccountName = "progstrgacc1";
    static string tmpStorageAddress = "https://progstrgacc1.blob.core.windows.net/";

    //--------------------------------------------------------------------
    // Main method: retrieve the access token and call the other methods
    //--------------------------------------------------------------------
    static void Main(string[] args) {
      // Retrieve a token from Azure AD using the application id and password
      var accessToken = GetAuthorizationToken (tmpApplicationId, tmpSubscriptionId, tmpTenantId,
          tmpAppPassword);

      // Create, or obtain a reference to, the resource group you are using
      var rgResponse = CreateUpdateResourceGroup (accessToken, tmpSubscriptionId,
          tmpResourceGroupName, tmpResourceGroupLocation);

      // Create a storage account
      CreateStorageAccount(accessToken, tmpSubscriptionId, tmpStorageAccountName,
          tmpResourceGroupName, tmpResourceGroupLocation);

      // Create the IoTHub
      //CreateIoTHub(accessToken, tmpSubscriptionId, tmpIoTHubName, tmpDeployName);

      Console.WriteLine("End, press some key");
      Console.ReadLine();
    }

    //------------------------------------------------------------
    // Methods to manage resources and services in Azure
    //------------------------------------------------------------
    ...
}
```

2.4. Obtención del token de autorización

Se necesitan como entrada los valores resultantes del proceso de autenticación: applicationId, subscriptionId, tenantId, y password. Y se retorna el token de autorización del directorio activo como una cadena de texto.

Algoritmo 4 Método para obtener el token de autorización.

```
private static string GetAuthorizationToken (string applicationId, string subscriptionId,
    string tenantId, string appPassword)  {
  Console.WriteLine("Getting authorization token credentials" + "...");
  var credential = new ClientCredential(applicationId, appPassword);
  var authContext = new AuthenticationContext(string.Format("https://login.windows.net/{0}",
      tenantId));
  AuthenticationResult token = authContext.AcquireTokenAsync("https://management.core.windows.
      net/", credential).Result;
  if (token == null) {
    throw new InvalidOperationException("Failed to obtain the token");
  }
  return token.AccessToken;
}
```

2.5. Creación de un grupo de recursos

Crea o obtiene una referencia al grupo de recursos dado como parámetro de entrada. Se necesita como entrada el token de acceso, el subscriptionId, el nombre del grupo de recursos, y la localización donde se va a crear ese grupo (si no existe). Retorna una referencia de tipo ResourceGroup que representa al grupo de recursos creado o existente.

Algoritmo 5 Método para crear un grupo de recursos.

```
private static ResourceGroup CreateUpdateResourceGroup(string accessToken, string
    subscriptionId, string rgName, string rgLocalization) {
  Console.WriteLine("Creating/Updating resource group: " + rgName + "...");

  var tokenCredentials = new TokenCredentials(accessToken);
  var rmClient = new ResourceManagementClient(tokenCredentials) {SubscriptionId =
      subscriptionId};

  var rgResponse = rmClient.ResourceGroups.CreateOrUpdate(rgName, new ResourceGroup(
      rgLocalization));
  if (rgResponse.Properties.ProvisioningState != "Succeeded") {
    throw new InvalidOperationException("Failed to creating/updating resource group");
  }
  Console.WriteLine("Resource Group:\n" + rgResponse.ToString());
  return rgResponse;
}
```

2.6. Creación de una cuenta de almacenamiento

Este método crea una cuenta de almacenamiento dado como entrada el token de acceso, el subscriptionId, el nombre a asignar a la cuenta, el grupo de recursos donde se creará la cuenta, y la localización donde ser va a crear. El tipo de cuenta que se crea es "StandarLRS".

Algoritmo 6 Método para creación de una cuenta de almacenamiento.

```
private static void CreateStorageAccount(string accessToken, string subscriptionId, string
    accountName, string rgName, string rgLocalization) {
  Console.WriteLine("Creating an Storage Account: " + "...");
  var tokenCredentials = new TokenCredentials(accessToken);
  var rmClient = new ResourceManagementClient(tokenCredentials) { SubscriptionId =
      subscriptionId };
  // Add a Microsoft.Storage provider to the subscription.
  var storageProvider = rmClient.Providers.Register("Microsoft.Storage");

  // Auth..
  var cloudTokenCredencials = new TokenCloudCredentials(subscriptionId, accessToken);
  var smClient = new StorageManagementClient(cloudTokenCredencials);

  // Your new storage account.
  var storageAccount = smClient.StorageAccounts.Create(rgName, accountName,
      new StorageAccountCreateParameters() {
        Location = rgLocalization,
        AccountType = AccountType.StandardLRS
      }
  );
  Console.WriteLine(">>> Storage Account: " + storageAccount.ToString());
}
```

2.7. Creación de un Hub de IoT

La creación de un Hub de IoT se realiza usando un esquema o template de *Azure Resource Manager*, el cual se debe cargar previamente a una cuenta de almacenamiento y que consite en dos archivos: uno para el esquema de Hub y el otro con los parámetros de creación (que contiene el nombre del Hub).

Para cargar los archivos se debe utilizar las herramientas de Visual Studio *Solution Explorer* o *Cloud Explorer*. En cualquiera de estas herramientas se ubica la subscripción en la que se está trabajando, después se ubica la cuenta de almacenamiento, y dentro del item de "blobs" se crea un nuevo contenedor llamado "templates". Una vez en el contenedor se procede a cargar *(upload)* los dos archivos seleccionándolos desde la ubicación local de su equipo donde los haya creado. El contenido de los archivos se muestra a continuación.

Algoritmo 7 Contenido archivo "esquema_iothub.json"

```
{ "$schema": "https://schema.management.azure.com/schemas/2015-01-01/deploymentTemplate.json#",
  "contentVersion": "1.0.0.0",
  "parameters": {"hubName": {"type": "string" }},
  "resources": [
  { "apiVersion": "2016-02-03",
    "type": "Microsoft.Devices/IotHubs",
    "name": "[parameters('hubName')]",
    "location": "East US",
    "sku": { "name": "S1", "tier": "Standard", "capacity": 1 },
    "properties": { "location": "East US" }
  }
  ],
  "outputs": {
    "hubKeys": {"value": "[listKeys(resourceId('Microsoft.Devices/IotHubs', parameters('hubName'))
        , '2016-02-03')]", "type": "object" }
  }
}
```

Algoritmo 8 Contenido archivo "paramtetros_iothub.json"

```
{ "$schema": "https://schema.management.azure.com/schemas/2015-01-01/deploymentParameters.json#",
  "contentVersion": "1.0.0.0",
  "parameters": {"hubName": { "value": "progiothub1" }}
}
```

Con los archivos ya cargados se puede ejecutar la aplicación con el método de creación del Hub de IoT, el cual se muestra a continuación. Como entrada necesita el token de acceso, el subscriptionId, el nombre del grupo de recursos donde se va a crear, y un nombre para identificar el despliegue que va

a realizar. Al final, el metodo imprime los valores de las claves de los recursos que se crean con el Hub de IoT.

Algoritmo 9 Método para creación de una cuenta de almacenamiento.

```
static void CreateIoTHub (string accessToken, string subscriptionId, string rgName, string
    deploymentName) {
  Console.WriteLine("Creating IoT Hub " + "...");
  var tokenCredentials = new TokenCredentials(accessToken);
  var resourceManagementClient = new ResourceManagementClient(tokenCredentials) {
       SubscriptionId = subscriptionId };

  // Submit the template and parameter files to the Azure Resource Manager
  var createResponse = resourceManagementClient.Deployments.CreateOrUpdate(
    rgName, deploymentName, new Deployment() {
      Properties = new DeploymentProperties {
        Mode = DeploymentMode.Incremental,
        TemplateLink = new TemplateLink { Uri = tmpStorageAddress + "templates/esquema_iothub.
            json" },
        ParametersLink = new ParametersLink { Uri = tmpStorageAddress + "templates/
            parametros_iothub.json" },
      } });

  // Displays the status and the keys for the new IoT hub:
  string state = createResponse.Properties.ProvisioningState;
  Console.WriteLine("Deployment state: {0}", state);
  if (state != "Succeeded")   {
    throw new InvalidOperationException("Failed to create the IoT Hub");
  }
  Console.WriteLine(createResponse.Properties.Outputs);
}
```

3. Manejo de dispositivos de Azure IoT mediante la herramienta Device Explorer

En el caso del manejo de los dispositivos, una de las formas más convenientes es realizarlo a través de la herramienta provista por Microsoft llamada Device Explorer (Figura 1). La herramienta permite realizar la mayoría de operaciones relacionadas con los dispositivos pertenecientes a un Hub de IoT (`https://github.com/Azure/azure-iot-sdks/blob/master/doc/manage_iot_hub.md`):

- Crear nuevos dispositivos
- Eliminar ya existentes
- Obtener información de los dispositivos (e.g. claves de acceso)
- Listar los existentes
- Envíar y recibir mensajes

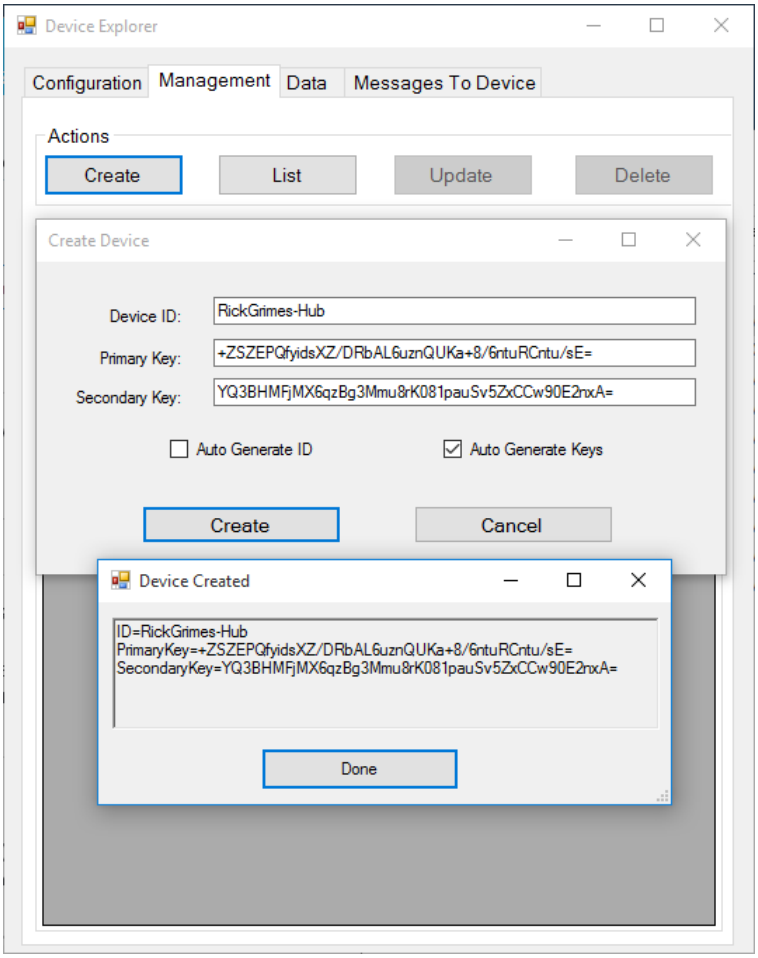

Figura 1: Manejo de dispositivos con Device Explorer.

Recursos

[1] Microsoft Docs. Create Azure service principal with PowerShell.

[2] Microsoft Docs. IoT Hub REST.