

Creación de un proyecto DevOps con Azure PowerShell

10/10/2023 · [Fernando de la Torre](#) · Arquitecto de Software

Contenido

[Variables y métodos que debemos definir en primer lugar](#)

[Pasos a seguir para crear un proyecto en Azure DevOps](#)

[Creación de un repositorio en Azure DevOps](#)

[Conclusión](#)

En este post veremos **cómo crear un proyecto en Azure DevOps, asignar a los usuarios a un grupo y vincular ese grupo al proyecto**, además de la **creación de un repositorio y librerías**, todo ello utilizando un *script* de **PowerShell**. La finalidad es poder **crear un script de automatización** y para ahorrar tiempo al generar la estructura base de nuestros proyectos.

Lo primero que debemos tener en cuenta es que deberemos tener instalado [Visual Studio](#) u otra herramienta con la que podamos crear y ejecutar nuestro *script*. Una vez llegados a este punto, lo primero que **definiremos en nuestro script** será una **serie de variables y funciones** que nos servirán a lo largo de todo el desarrollo.

Variables y métodos que debemos definir en primer lugar

En primer lugar, debemos **definir las siguientes variables**.

```
$Organization="Profile"
$ProjectName="Proyecto"
$ProjectDescription="Proyecto creado desde powershell"
$ProcessTemplateName = "Scrum"
$repositoryName= "mirepositorio"
```

También crearemos **dos métodos** para nuestras **peticiones Get y Post**, para no tener que repetir código.

```
function Headers-Get(){
    $encodedBytes = [System.Text.Encoding]::UTF8.GetBytes("profile:$env:AZURE_DEVOPS_EXT_PAT")
    $base64PAT = [convert]::ToBase64String($encodedBytes)
    $headers = New-Object "System.Collections.Generic.Dictionary[[String],[String]]"
    $headers.Add("Authorization", "Basic $base64PAT")

    $headers
}

function Headers-Post(){
    $encodedBytes = [System.Text.Encoding]::UTF8.GetBytes("profile:$env:AZURE_DEVOPS_EXT_PAT")
    $base64PAT = [convert]::ToBase64String($encodedBytes)
```

```
$headers = New-Object "System.Collections.Generic.Dictionary[[String],[String]]"
$headers.Add("Authorization", "Basic $base64PAT")
$headers.Add("Content-Type", "application/json")

$headers

}
```

Pasos a seguir para crear un proyecto en Azure DevOps

Lo primero que tenemos que realizar es la **recuperación del ID de la plantilla Scrum** (*template* que usaremos para nuestro proyecto) según nuestra organización, para ello ejecutaremos el siguiente código. Obteniendo como resultado el objeto de la plantilla.

```
$headers= Headers-Get

$response = Invoke-RestMethod "https://dev.azure.com/${$Organization}/_apis/process/processes?api-version=7.1-preview.1" -Method 'GET' -Headers $headers

$template = $response.value | Where-Object { $_.name -eq $ProcessTemplateName }
```

Para crear el proyecto debemos tener en cuenta la petición que debemos realizar.

```
POST https://dev.azure.com/{organization}/_apis/projects?api-version=7.0
```

Una vez que tenemos dicha información, pasaremos a **crear el cuerpo de la petición** y realizaremos la llamada a la API, para que cree el proyecto.

```
$headers= Headers-Post

$body = "{
  n  "name": "${$ProjectName}",
    "description": "${$ProjectDescription}",
  n  "capabilities": {
    "versioncontrol": {
  n    "sourceControlType": "Git"
    },
  n  "processTemplate": {
    "templateTypeId": "${$template.id}"
  n  }
  }
n}"

Invoke-RestMethod "https://dev.azure.com/${$Organization}/_apis/projects?api-version=7.1-preview.4" -
Method 'POST' -Headers $headers -Body $body

</code></pre>
<!-- /wp:code -->

<!-- wp:paragraph -->
<p>Con esto ya tendremos creado nuestro proyecto.</p>
<!-- /wp:paragraph -->

<!-- wp:heading -->
<h2>Creación de grupos, asignación de usuarios y roles</h2>
<!-- /wp:heading -->

<!-- wp:paragraph -->
<p>Ahora vamos a crear un grupo donde estarán, por ejemplo, los <em>developers</em> del proyecto. Podemos <strong>crear diferentes grupos</strong>, según nuestras necesidades, para gestionar nuestro proyecto.</p>
<!-- /wp:paragraph -->
```

```

<!-- wp:code -->
<pre class="wp-block-code"><code>$GroupName="Developers"</code></pre>
<!-- /wp:code -->

<!-- wp:paragraph -->
<p>En primer lugar, <strong>recuperaremos el ID</strong> del proyecto que hemos generado.</p>
<!-- /wp:paragraph -->

<!-- wp:code -->
<pre class="wp-block-code"><code>$headers= Headers-Get

$response = Invoke-RestMethod
"https://dev.azure.com/$($Organization)/_apis/projects/$($ProjectName)?api-version=7.0" -Method 'GET' -
Headers $headers

$ProjectId =$response.id
</code></pre>
<!-- /wp:code -->

<!-- wp:paragraph -->
<p>Una vez que hayamos obtenido el identificador de proyecto, <strong>buscaremos el
<em>ScopeDescriptor</em></strong><em> </em>del mismo. Para ello, realizaremos la siguiente
petición.</p>
<!-- /wp:paragraph -->

<!-- wp:code -->
<pre class="wp-block-code"><code>$headers= Headers-Get

$response = Invoke-RestMethod "https://vssps.dev.azure.com/$($Organization)/_apis/graph/descriptors/$($
ProjectId)?api-version=6.0-preview.1" -Method 'GET' -Headers $headers

$projectDescriptor = $response.value
</code></pre>
<!-- /wp:code -->

<!-- wp:paragraph -->
<p>Ahora ya tenemos todos los datos necesarios para crear el grupo.</p>
<!-- /wp:paragraph -->

<!-- wp:code -->
<pre class="wp-block-code"><code>$headers= Headers-Post

$body = "{
    "displayName": "$($GroupName)",
    n "description": "$($GroupName)"
}"

$grupo = Invoke-RestMethod
"https://vssps.dev.azure.com/$($Organization)/_apis/graph/groups?scopeDescriptor=$($projectDescriptor)&
api-version=5.0-preview.1" -Method 'POST' -Headers $headers -Body $body

```

Para **añadir un usuario a nuestro grupo** realizaremos los siguientes pasos:

Lo primero, es recuperar el *SubjectDescriptor* del usuario y luego asignarlos al grupo.

```

$listMembersDevelopment = @("mail1@mail.com","mail2@mail.com",...)

# recorremos todos los miembros
foreach ($member in $listMembersDevelopment){

#recuperamos el subjectdescriptor del miembro
    $headers= Headers-Get

$response = Invoke-RestMethod "https://vssps.dev.azure.com/$($Organization)/_apis/identities?api-

```

```
version=6.0&searchFilter=General&filterValue=${($member)}" -Method 'GET' -Headers $headers

$userDescriptors = $response.value[0].subjectDescriptor

#petición para asignar al grupo
foreach ($userDescriptor in $userDescriptors) {
    $headers= Headers-Get

Invoke-RestMethod
"https://vssps.dev.azure.com/${($Organization)}/_apis/graph/memberships/${($userDescriptor)}/${($grupo.descripcion)}?api-version=5.0-preview.1" -Method 'PUT' -Headers $headers
}

}
```

Creación de un repositorio en Azure DevOps

Lo primero que haremos será **recuperar el ID del proyecto** al cual queremos añadir un nuevo repositorio. Para ello, realizaremos **una llamada GET** a nuestro Azure con la siguiente estructura de URL:

```
GET https://dev.azure.com/{organization}/_apis/projects/{projectId}?api-version=7.0
```

Por lo tanto, nuestro código resultante sería de la siguiente manera:

```
$headers= Headers-Get

$Project = Invoke-RestMethod
"https://dev.azure.com/${($Organization)}/_apis/projects/${($ProjectName)}?api-version=7.1-preview.4" -
Method 'GET' -Headers $headers
```

Para recuperar el ID de nuestro proyecto, **utilizaremos la variable** en la que se ha guardado la **respuesta del Invoke**.

```
$projectId = $Project.id
```

Ahora que ya tenemos el ID de nuestro proyecto, procederemos a la **creación de un nuevo repositorio** asignado al proyecto.

Lo primero que debemos tener en cuenta es la **URL a la que se realizará la llamada**, la cual es:

```
POST https://dev.azure.com/{organization}/{project}/_apis/git/repositories?api-version=7.0
```

Y debemos **crear el cuerpo de la petición**, en el cual indicaremos el ID del proyecto recuperado y el nombre de nuestro repositorio, siguiendo la siguiente estructura:

```
$body = "{
  n  "name": "${($repositoryName)",
    "project": {
      n    "id": "${($projectId)"
    }
  n}"
</code></pre>
<!-- /wp:code -->

<!-- wp:paragraph -->
<p>Como consecuencia, nuestro código resultante sería el siguiente:</p>
<!-- /wp:paragraph -->

<!-- wp:code -->
<pre class="wp-block-code"><code>$headers= Headers-Post
```

```

$url = "https://dev.azure.com/$(Organization)/_apis/git/repositories?api-version=7.0"

$body = "{
  "name": "$(RepositoryName)",
  "project": {
    "id": "$(ProjectId)"
  }
}"

Invoke-WebRequest -Uri $url -Method POST -Headers $headers -Body $body -ContentType "application/json"

```

Si todo ha ido correctamente, **obtendremos una respuesta 200**, que indica que se ha creado el repositorio.

Una vez que sabemos cómo recuperar el ID del proyecto y cómo crear un repositorio en ese proyecto, intentaremos **unificar todo el código** asignando valores a nuestras variables.

```

$headers= Headers-Get

$Project = Invoke-RestMethod
"https://dev.azure.com/$(Organization)/_apis/projects/$(ProjectName)?api-version=7.1-preview.4" -
Method 'GET' -Headers $headers

$projectId = $Project.id

$headers= Headers-Post

$url = "https://dev.azure.com/$(Organization)/_apis/git/repositories?api-version=7.0"

$body = "{
  "name": "$(RepositoryName)",
  "project": {
    "id": "$(ProjectId)"
  }
}"

Invoke-WebRequest -Uri $url -Method POST -Headers $headers -Body $body -ContentType "application/json"

</code></pre>
<!-- /wp:code -->

<!-- wp:paragraph -->
<p>Si, por el contrario, deseas <strong>crear más de un repositorio</strong> para nuestro proyecto, solo habría que <strong>duplicar el <em>body</em></strong>, modificando el nombre del repositorio, y duplicar la línea de <em>Invoke</em>. Aunque otra forma de realizar esta operación es realizar una<strong> pequeña modificación a la hora de llamar al <em>invoke</em> </strong>de creación.</p>
<!-- /wp:paragraph -->

<!-- wp:code -->
<pre class="wp-block-code"><code>@( "repo001", "repo002" ) | ForEach-Object {
  $name = $_
  $body = @"
{
  "name": "$name",
  "project": {
    "id": "$projectId"
  }
}
"@
  Invoke-WebRequest -Uri $url -Method POST -Headers $headers -Body $body -ContentType
"application/json"
}
</code></pre>
<!-- /wp:code -->

<!-- wp:heading -->

```

```

<h2>Creación de variables Vsts</h2>
<!-- /wp:heading -->

<!-- wp:paragraph -->
<p>Para crear <strong>variables de entorno</strong> en nuestro apartado <em>Pipelines > Library,</em>debemos tener en cuenta la petición que debemos realizar, la cual es la siguiente:</p>
<!-- /wp:paragraph -->

<!-- wp:code -->
<pre class="wp-block-code"><code>POST
https://dev.azure.com/{organization}/_apis/serviceendpoint/endpoints?api-version=7.0</code></pre>
<!-- /wp:code -->

<!-- wp:paragraph -->
<p>Lo primero que rellenaremos será, como siempre, nuestra cabecera que enviaremos en la solicitud, la URL con los parámetros necesarios y e<strong>l <em>projectId</em></strong><em> </em>previamente<strong>recuperado</strong>. A continuación, configuraremos el cuerpo del mensaje con toda la información necesaria.</p>
<!-- /wp:paragraph -->

<!-- wp:list -->
<ul><!-- wp:list-item -->
<li>Por ejemplo, vamos a crear un grupo de variables llamado "GRUPO_VARIABLES" que incluirá las siguientes variables: ENTORNO: PRE y PERTENECE_SISTEMA: SI.</li>
<!-- /wp:list-item --></ul>
<!-- /wp:list -->

<!-- wp:code -->
<pre class="wp-block-code"><code>$body = "{
    "name":"GRUPO_VARIABLES",
    n "type":"Vsts",
    "variables":{
        n "ENTORNO":{ "isSecret":false, "value": "PRE)" },
</code></pre>
<!-- /wp:code -->

<!-- wp:code -->
<pre class="wp-block-code"><code>                                " PERTENECE_SISTEMA ":{ "isSecret":false, "value":
"SI" }
        n },
        "variableGroupProjectReferences":[{
            n "name":"GRUPO_VARIABLES",
              "projectReference":{
                n "id":"${$ProjectId}"
                  }
            }
        ]
    }"
</code></pre>
<!-- /wp:code -->

<!-- wp:paragraph -->
<p>Como resultado de todos los pasos, el fragmento de código debería quedar de la siguiente manera:</p>
<!-- /wp:paragraph -->

<!-- wp:code -->
<pre class="wp-block-code"><code>$headers= Headers-Post

$url = "https://dev.azure.com/${$Organization}/_apis/distributedtask/variablegroups?api-version=7.0"
$body = "{
    "name":"${$GroupName}",
    n "type":"Vsts",
    "variables":{
        n "ENTORNO":{ "isSecret":false, "value": "${$Entorno)" },
          "GRUPOS":{ "isSecret":false, "value": "${$VarGrupos)" }
        n },
        "variableGroupProjectReferences":[{
            n "name":"${$GroupName)",
              "projectReference":{

```

```
        n        "id": "$($ProjectId)`"
        }
    }
}]"
Invoke-WebRequest -Uri $url -Method POST -Headers $headers -Body $body -ContentType
"application/json"
```

Conclusión

Aunque en este artículo solo se ha visto una pequeña pincelada de lo que podemos hacer con la API de Azure DevOps mediante llamadas desde PowerShell, hay que tener en cuenta que automatizar la creación de proyectos es algo complejo, pero que ahorrará una gran cantidad de tiempo. La creación manual es un proceso que puede llevar varias horas, mientras que generando un *script* que se encargue de ello, reduciríamos el tiempo a unos minutos. Espero que esta breve introducción a la automatización de proyectos en Azure DevOps mediante la ejecución de un *script* les haya resultado gratificante.