Hands On Lab DevOps para PowerBI

VM Name: **handsonlab4272891256000.westeurope.cloudapp.azure.com**

Usuarios

|  |  |  |
| --- | --- | --- |
| Usuario | Pass | Pto RDP |
| powerbiuser1@m365x628809.onmicrosoft.com | Paqu6320 | 54527 |
| [powerbiuser2@m365x628809.onmicrosoft.com](mailto:powerbiuser2@m365x628809.onmicrosoft.com) | Paqu6320 | 63085 |
| powerbiuser3@m365x628809.onmicrosoft.com | Paqu6320 | 57655 |
| powerbiuser4@m365x628809.onmicrosoft.com | Paqu6320 | 60999 |
| powerbiuser5@m365x628809.onmicrosoft.com | Paqu6320 | 61879 |
| powerbiuser6@m365x628809.onmicrosoft.com | Paqu6320 | 50454 |
| powerbiuser7@m365x628809.onmicrosoft.com | Paqu6320 | 64678 |
| powerbiuser8@m365x628809.onmicrosoft.com | Paqu6320 | 61709 |
| powerbiuser9@m365x628809.onmicrosoft.com | Paqu6320 | 56124 |
| powerbiuser10@m365x628809.onmicrosoft.com | Paqu6320 | 64516 |
| powerbiuser11@m365x628809.onmicrosoft.com | Paqu6320 | 57035 |
| powerbiuser12@m365x628809.onmicrosoft.com | Paqu6320 | 65002 |
| powerbiuser13@m365x628809.onmicrosoft.com | Paqu6320 | 55685 |
| powerbiuser14@m365x628809.onmicrosoft.com | Paqu6320 | 58031 |
| powerbiuser15@m365x628809.onmicrosoft.com | Paqu6320 | 55082 |
| powerbiuser16@m365x628809.onmicrosoft.com | Paqu6320 | 52229 |
| powerbiuser17@m365x628809.onmicrosoft.com | Paqu6320 | 53646 |
| powerbiuser18@m365x628809.onmicrosoft.com | Paqu6320 | 60039 |
| powerbiuser19@m365x628809.onmicrosoft.com | Paqu6320 | 53088 |
| powerbiuser20@m365x628809.onmicrosoft.com | Paqu6320 | 56631 |

Login VM:

Usuario: powerbi

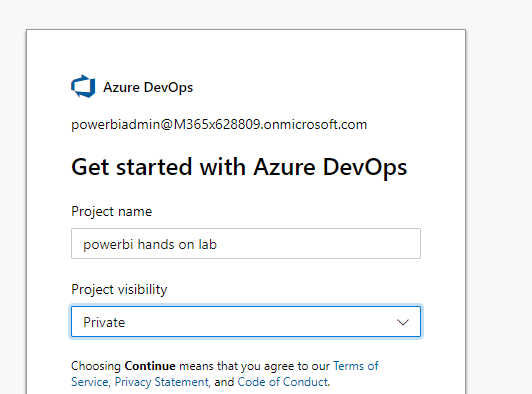
Password: Paqu6320

# Creación Organización Azure DevOps

Vamos a la dirección <https://dev.azure.com> y pulsamos sobre ‘sign in to Azure DevOps’

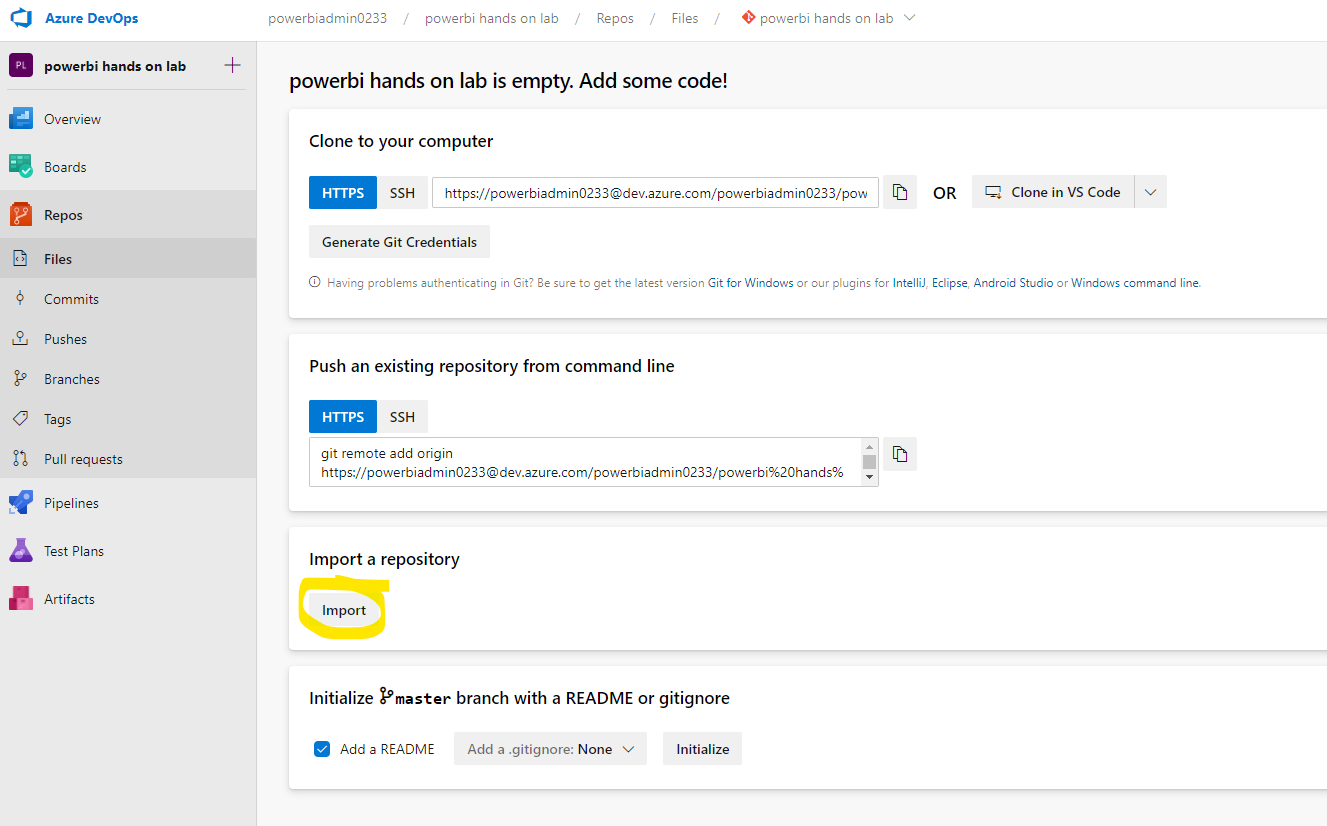


Nos creara una nueva organización y nos pedirá el nombre del proyecto inicial



Vamos a importar el repositorio padre con el que vamos a trabajar.

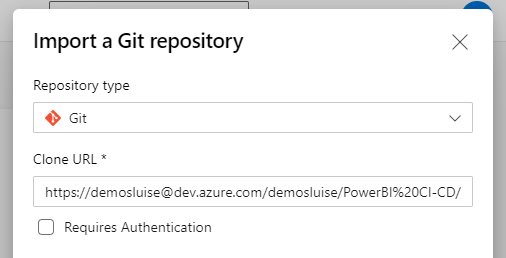
Vamos a la sección Repos/Files y pulsamos sobre import a repository



Introducimos la URL:

[*https://demosluise@dev.azure.com/demosluise/PowerBI%20CI-CD/\_git/PowerBI%20CI-CD*](https://demosluise@dev.azure.com/demosluise/PowerBI%20CI-CD/_git/PowerBI%20CI-CD)

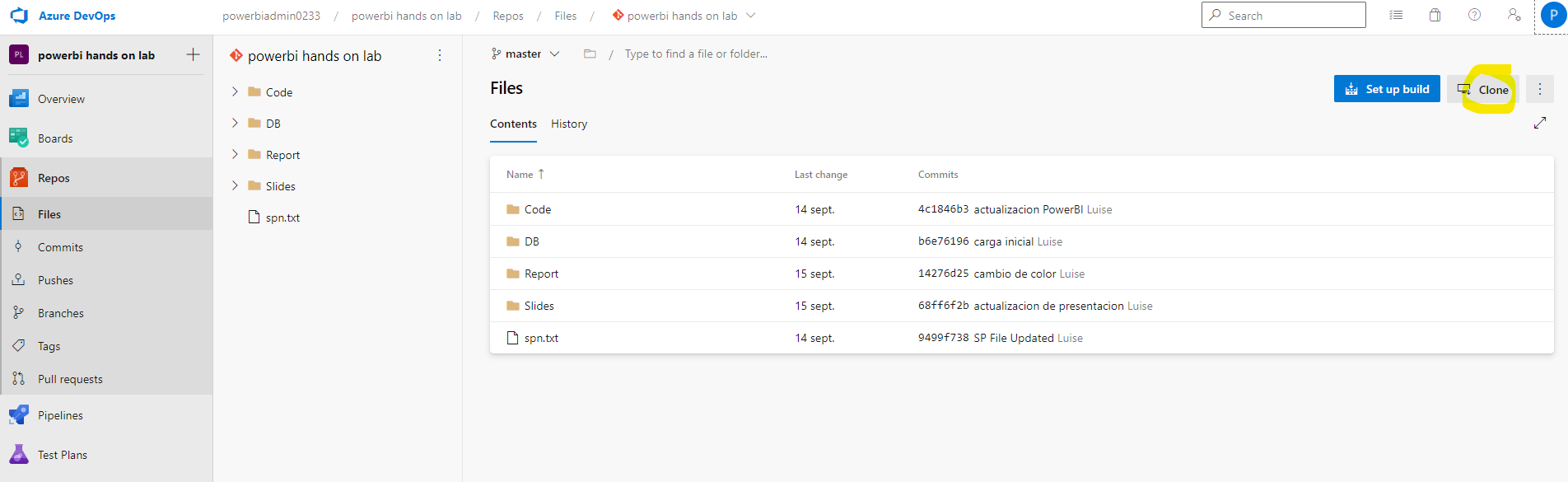
y le Pulsamos Import.



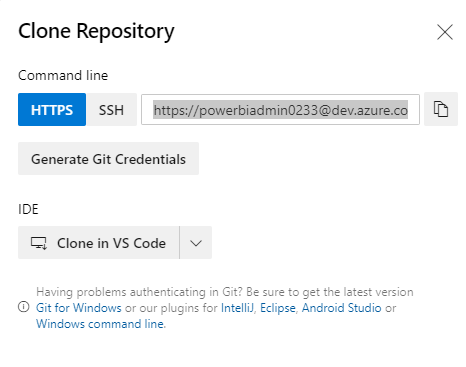
En unos momentos aparecerá un mensaje de Éxito en la importación.

## Clonación del repositorio a local

Pulsamos el botón clonar, en el repositorio de Azure Devops



Y copiamos la URL



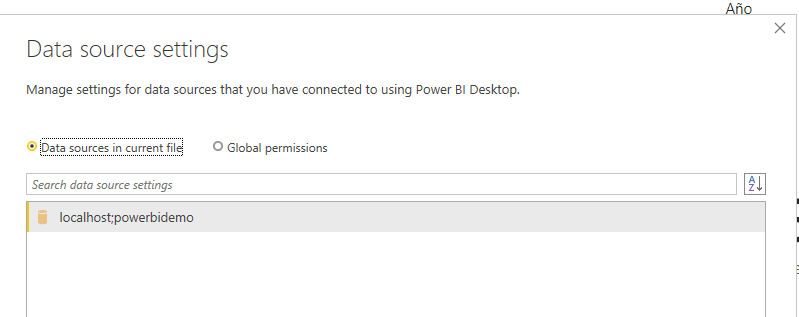
Desde PowerShell, vamos al directorio raíz C:\ y ejecutamos

*Git clone htps://powerbiadmin0233@dev.azure.com/powerbiadmin0233/powerbi%20hands%20on%20lab/\_git/powerbi%20hands%20on%20lab*

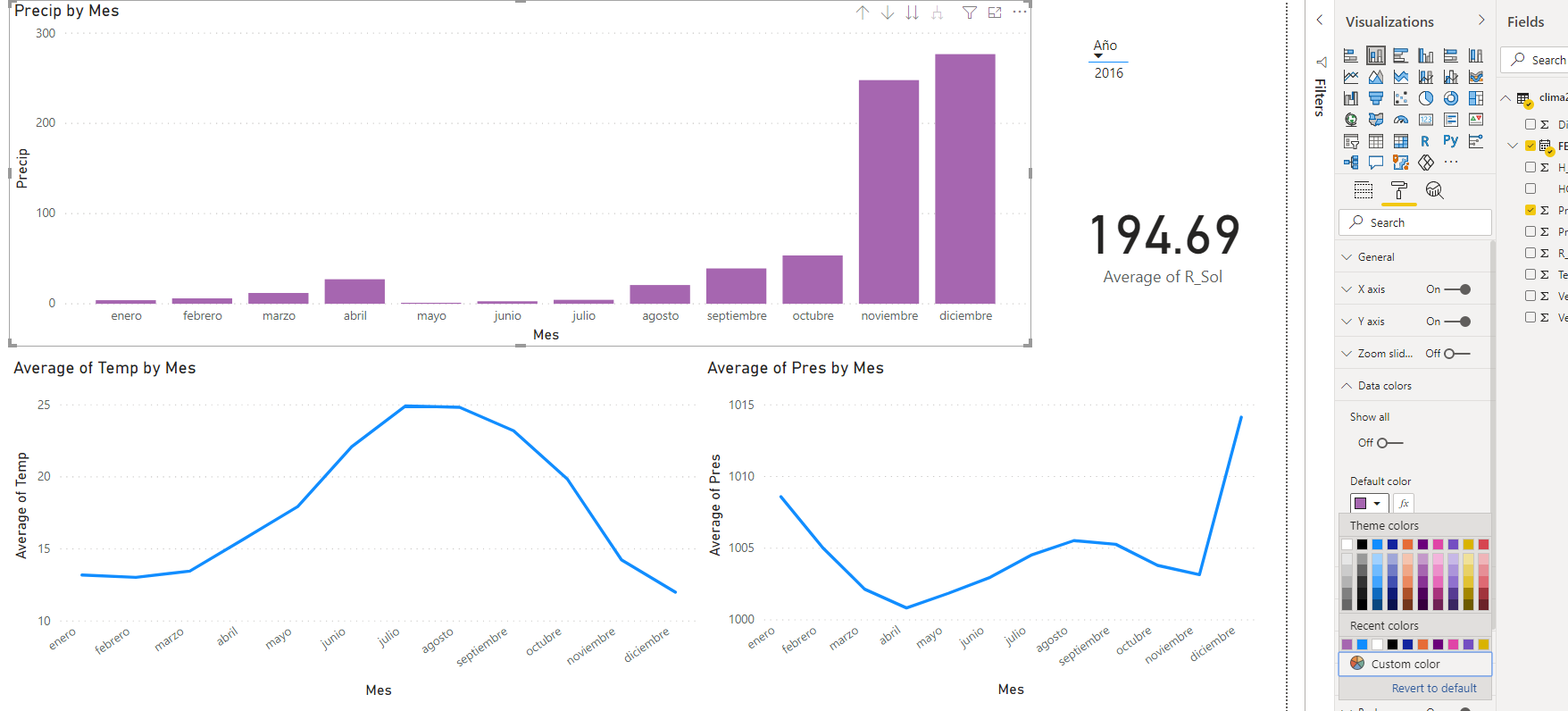
Ponemos las credenciales de Azure DevOps, cuando nos las pida y se sincronizara el repositorio.

Abrimos el Dashboard PowerBI ClimaValencia.pbix que esta en la ruta C:\powerbi%20hands%20on%20lab\Report

Chequeamos que el origen de datos es el SQL local



Y cambiamos el color del KPI Precip by mes de Morado a Azul.



Guardamos los cambios en PowerBi Desktop.

Desde powershell, configuramos git para poder hacer subidas.

*Git config –global user.email “powerbiuserXX@m365x628809.onmicrosoft.com”*

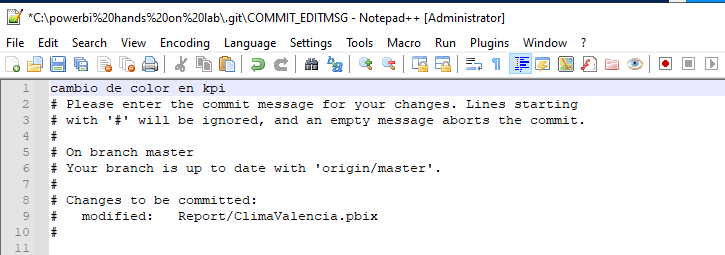
Añadimos las modificaciones a Git con

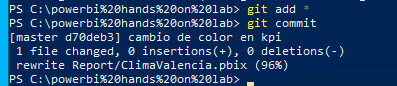
*Git add \**

Y actualizamos el repositorio local con la nueva version

*Git commit*

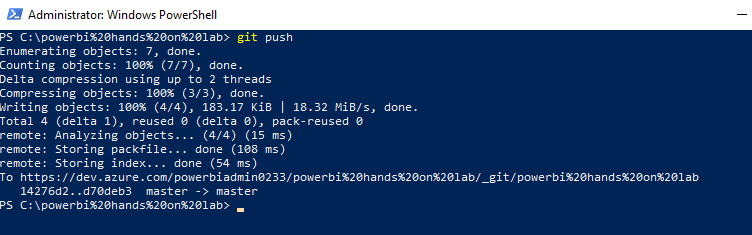
Se abrirá un Notepad++ para que añadamos el mensaje que acompañara a la nueva versión. Lo escribimos en la primera línea y guardamos y cerramos el Notepad++



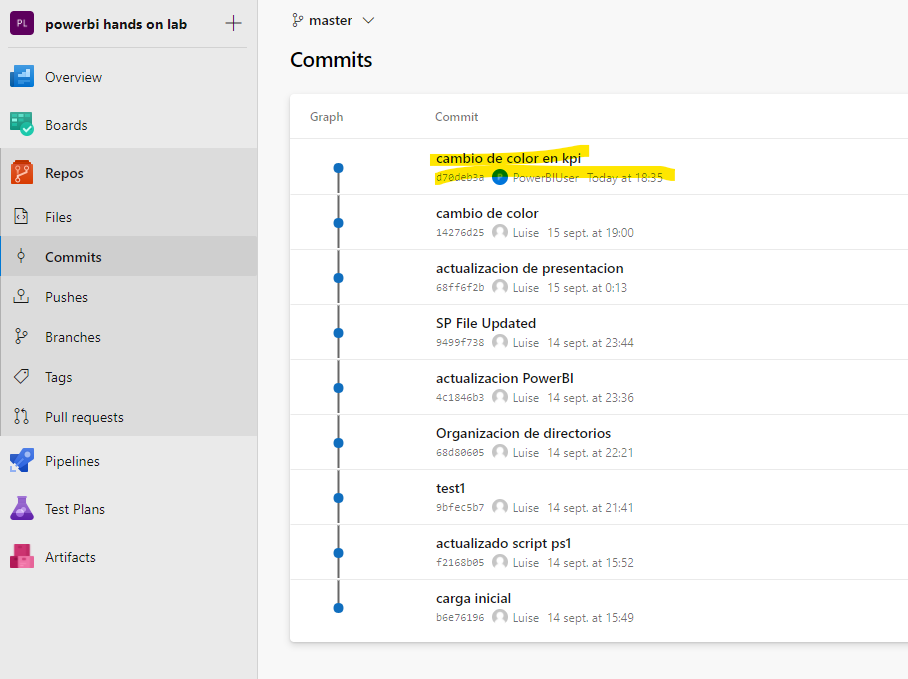


El archivo se ha actualizado en el Git local. Para sincronizarlo con el repositorio remoto ejecutamos:

*Git push*

**

Comprobamos en el repositorio de Azure DevOps, la historia del repositorio y comprobamos que ha subido la ultima versión desde la pestaña **‘commits’**

****

Desde la consola, recibimos la última versión del repositorio con

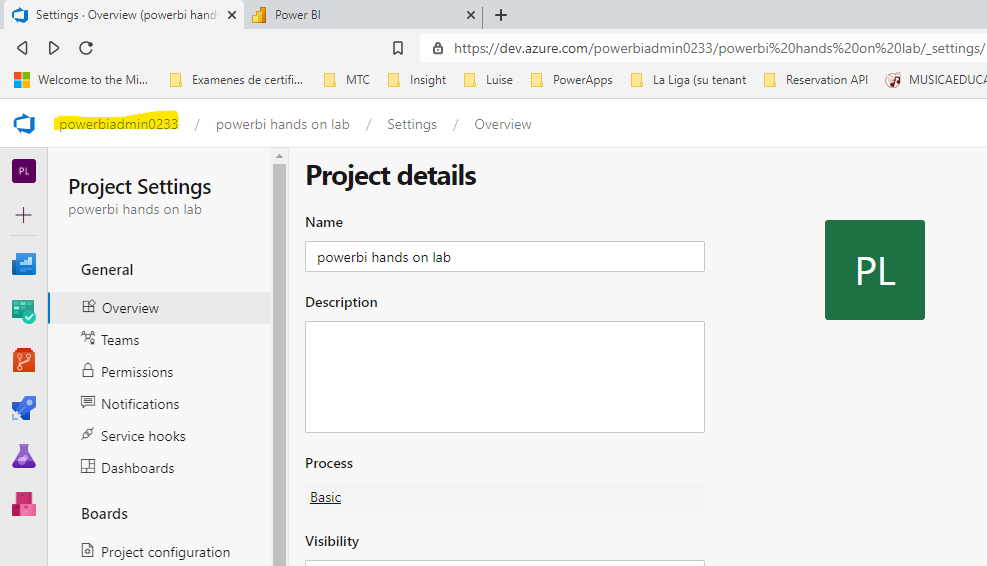
*Git pull*

# Part 2 - Publicacion automática de Dashboards.

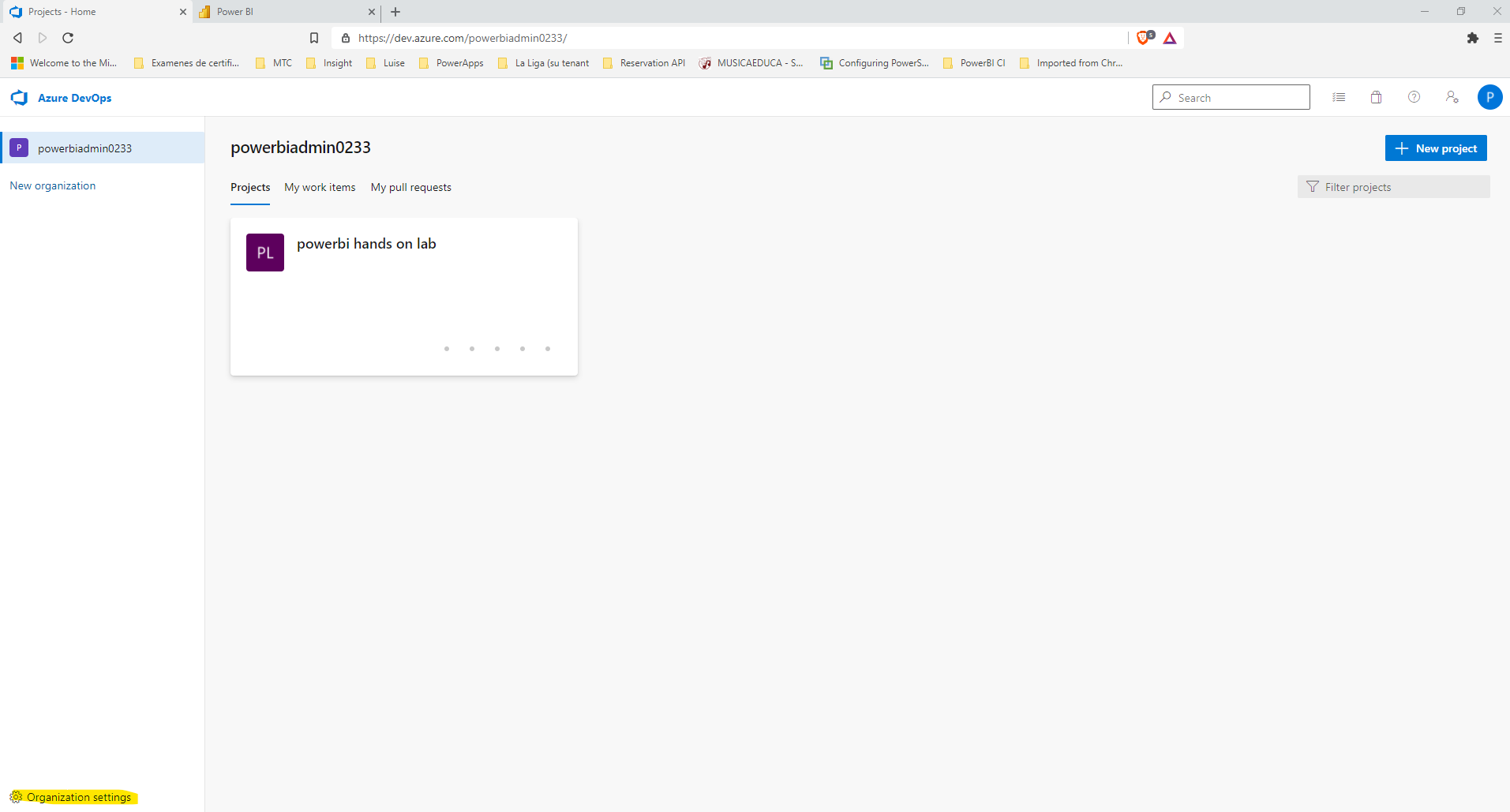
## Preparación del entorno

### Instalación de Extensiones en Azure DevOps

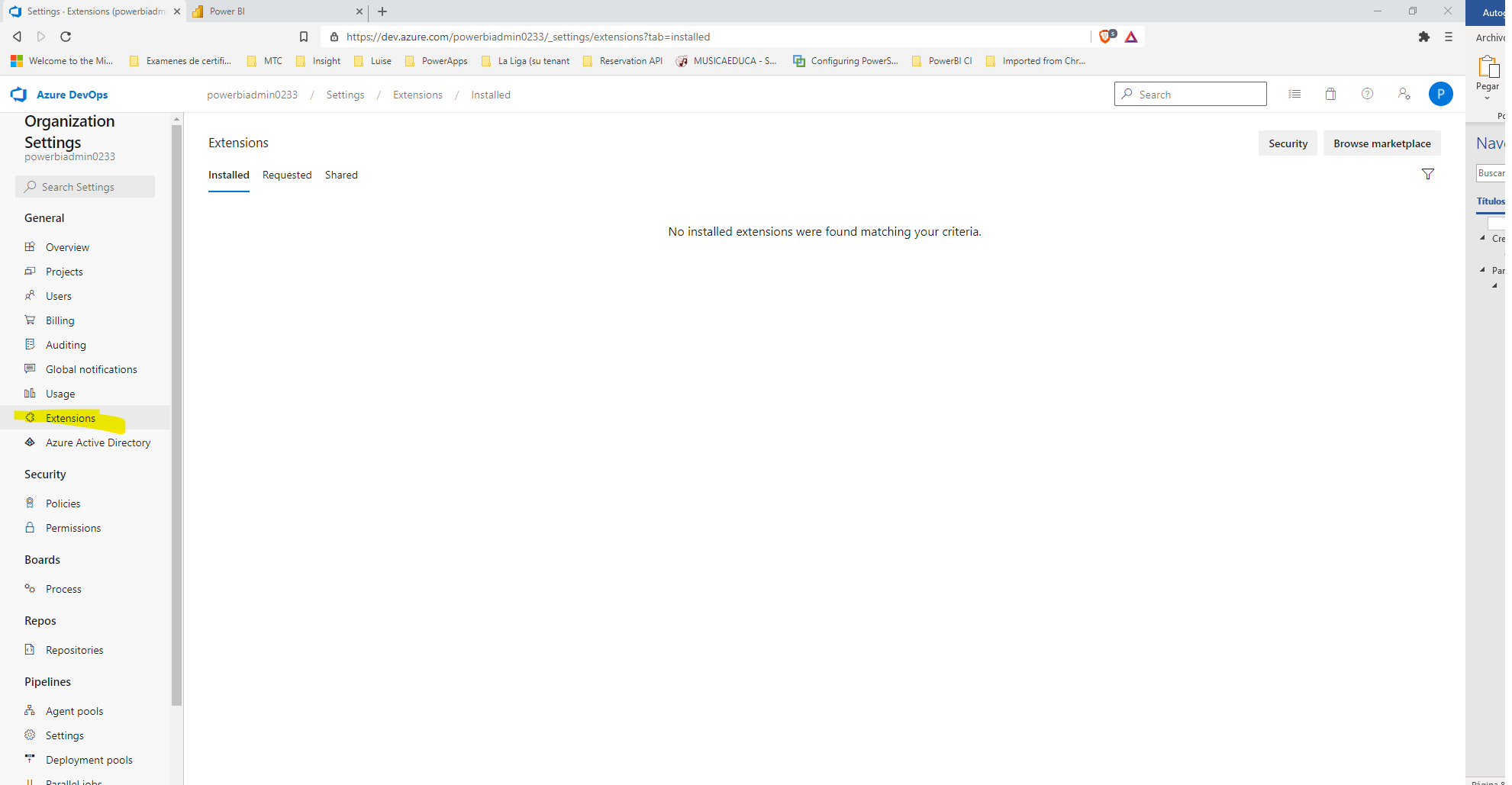
En Azure DevOps, nos vamos al raíz de la organización de Azure DevOps



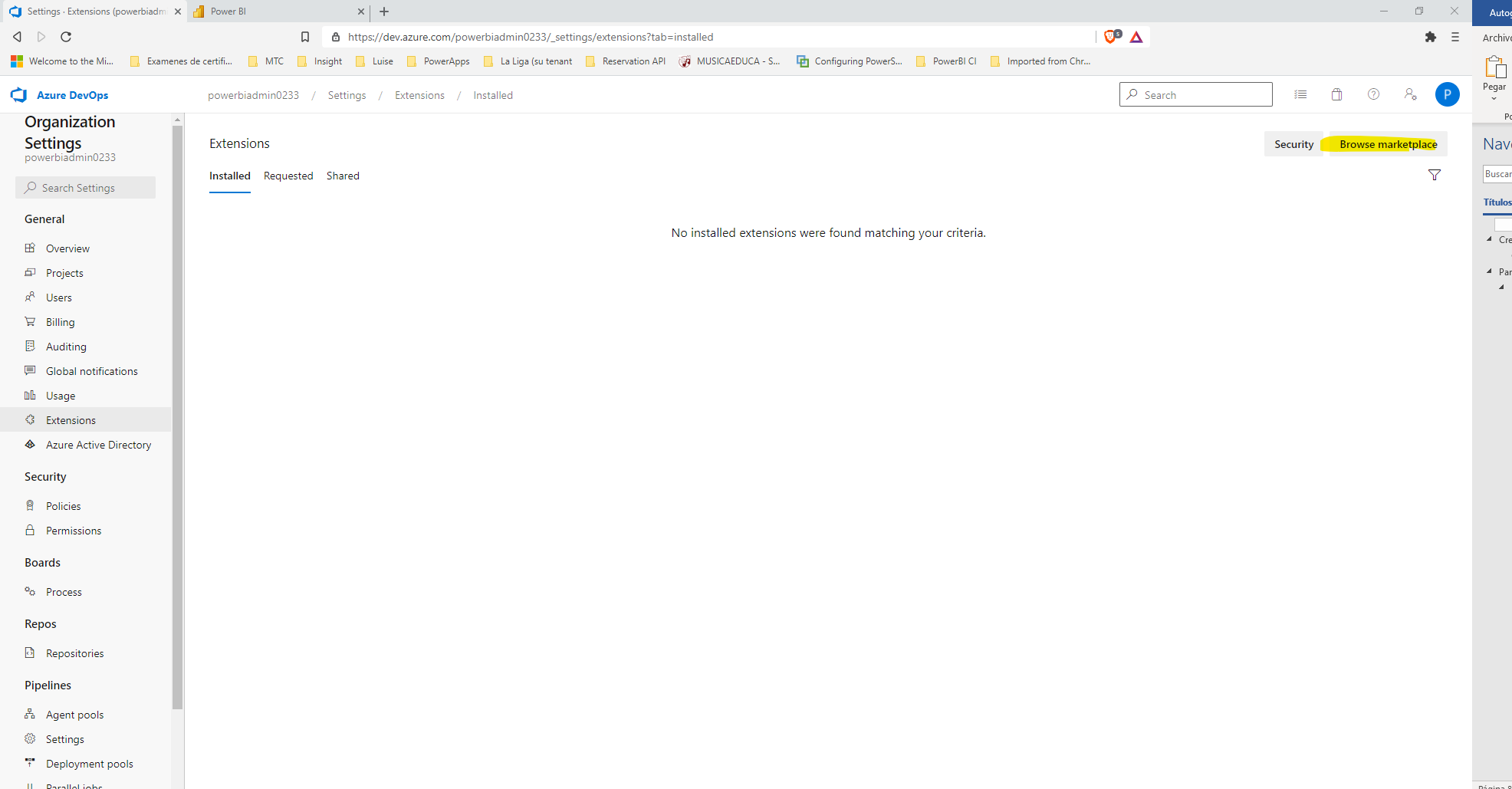
Pulsamos en ‘Organization Settings’



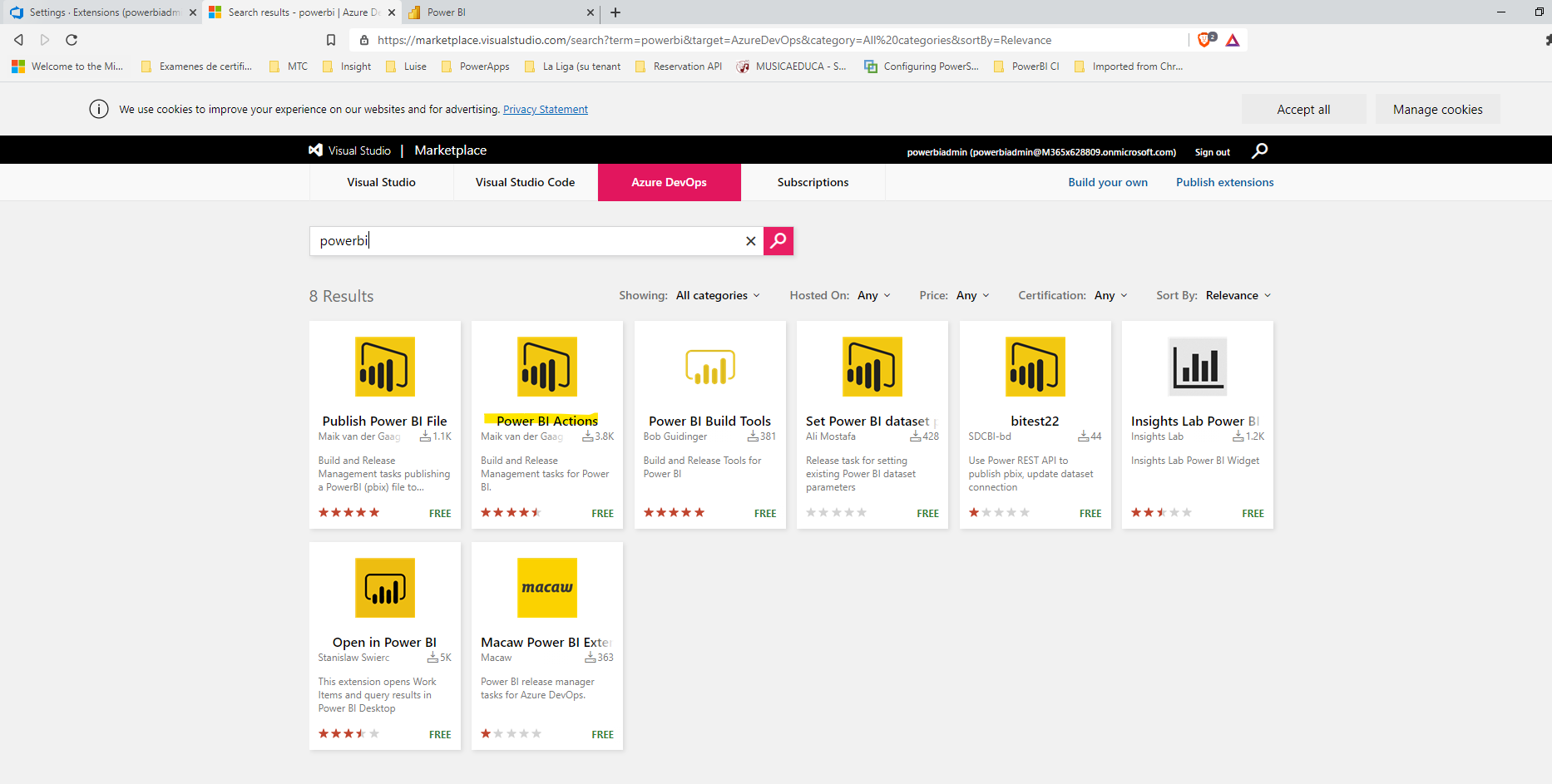
Seleccionamos ‘Extensions’



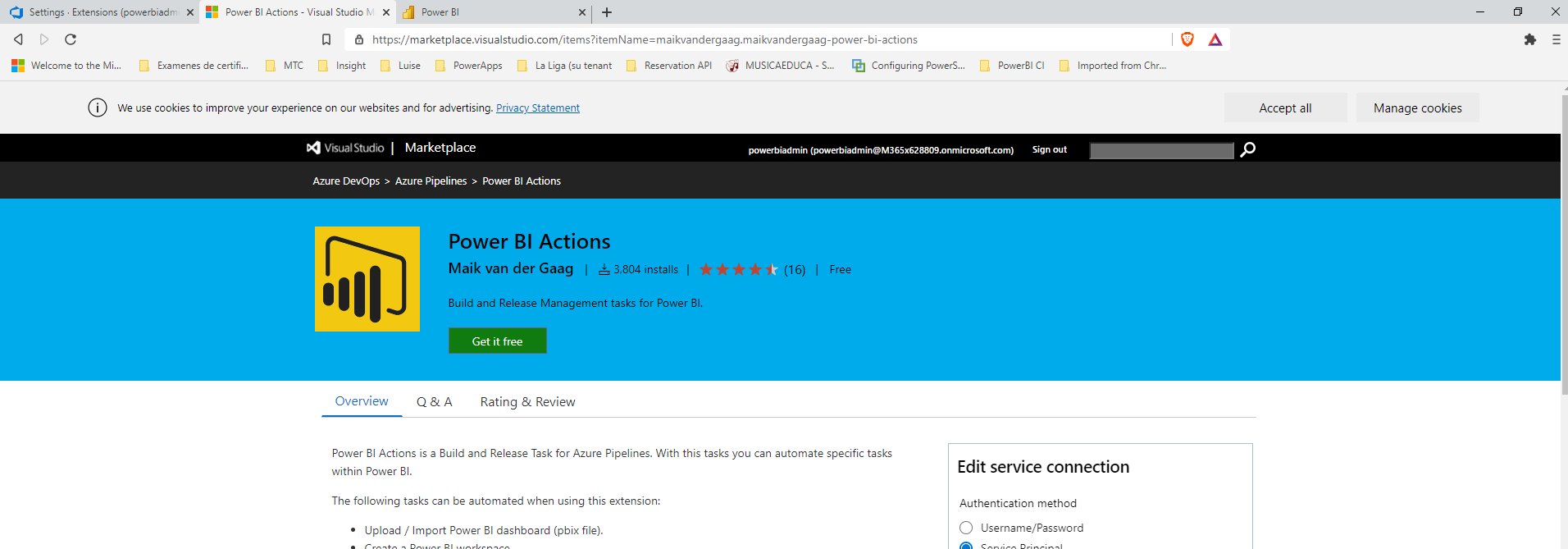
Browse MarketPlace



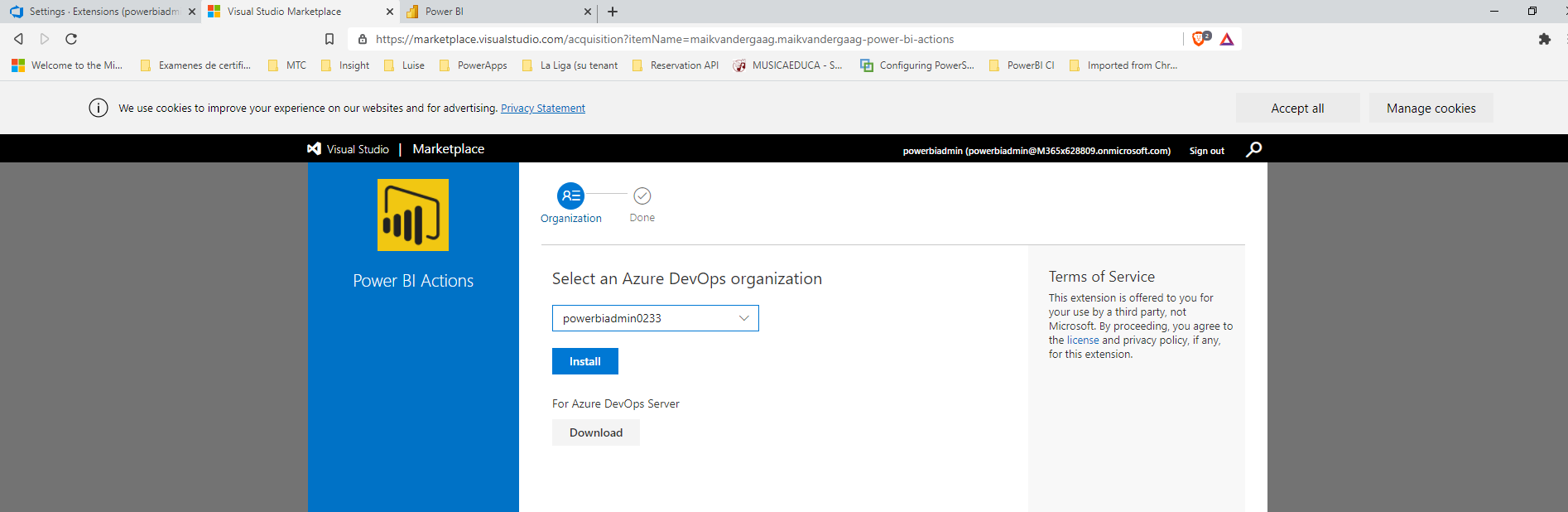
Buscamos ‘PowerBI’ y seleccionamos la extensión ‘Power BI Actions’.



LA añadimos con ‘Get It Free’



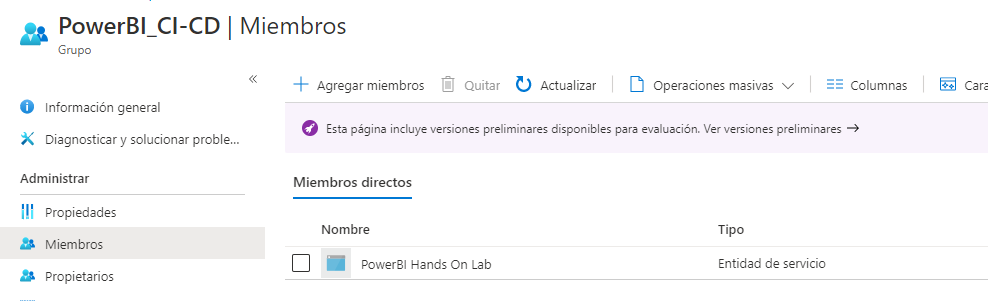
Y la añadimos a nuestra organización con ‘Install’



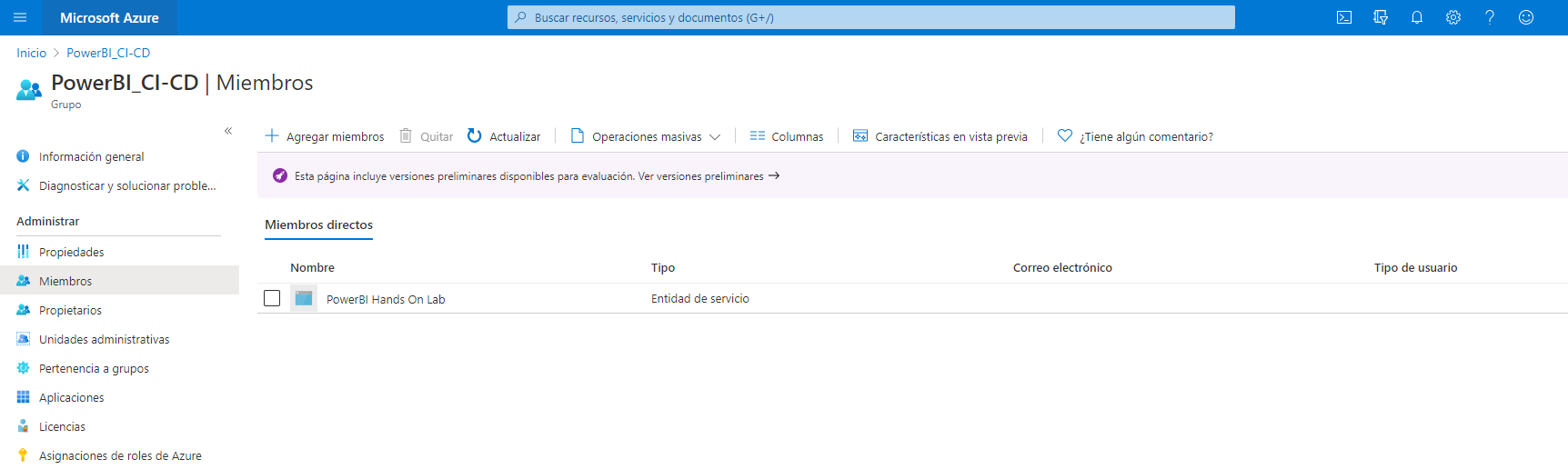
**Estas tareas que vienen a continuación ya están realizadas. Como compartimos Tenant las ha realizado el administrador. Pero hay que hacerlas si se trata de un Tenant nuevo**

En Azure AD, creamos un SPN con un secret y un grupo de seguridad. Añadimos el SPN como miembro de ese grupo de seguridad

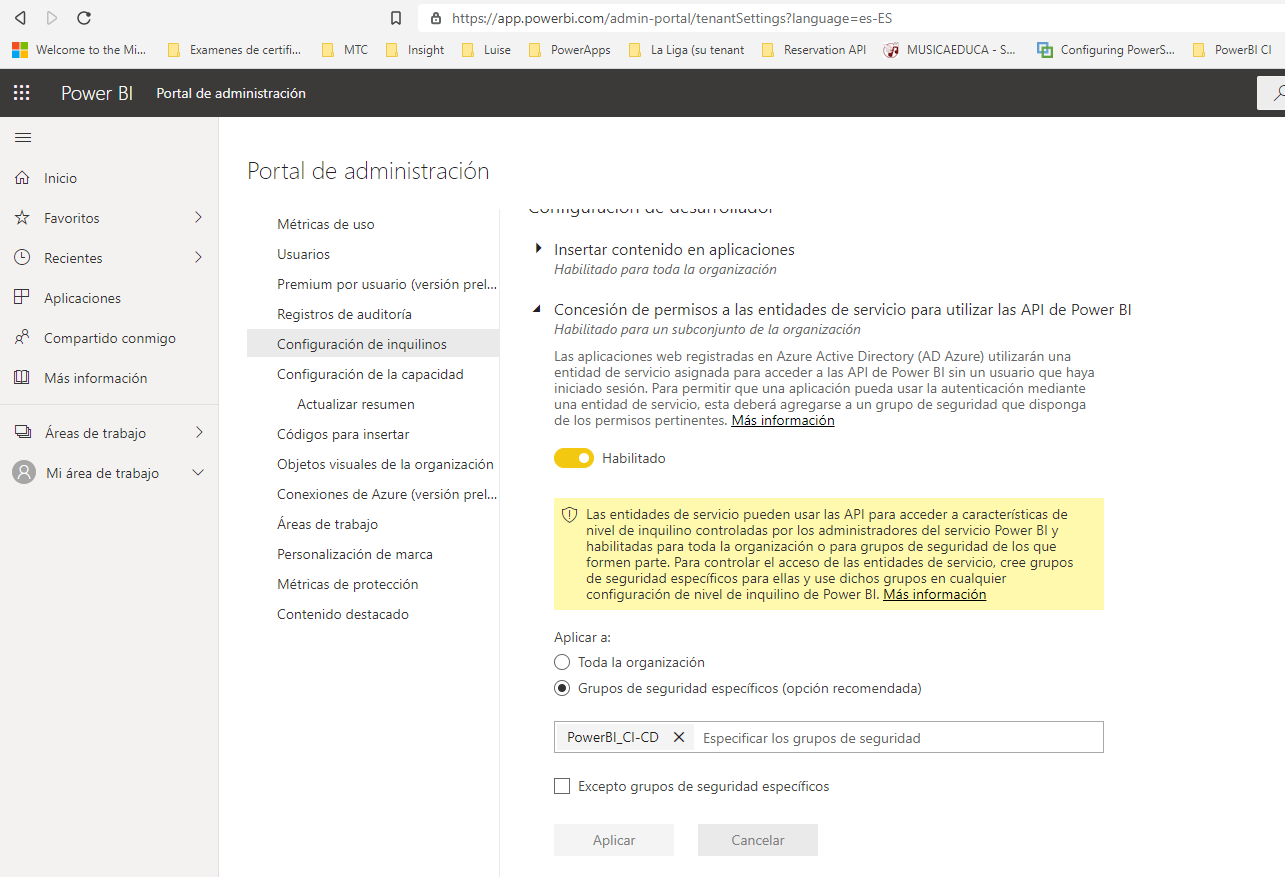
<https://portal.azure.com/#blade/Microsoft_AAD_IAM/ActiveDirectoryMenuBlade/Overview>



En Azure AD, creamos un grupo de seguridad y metemos dentro el SPN.

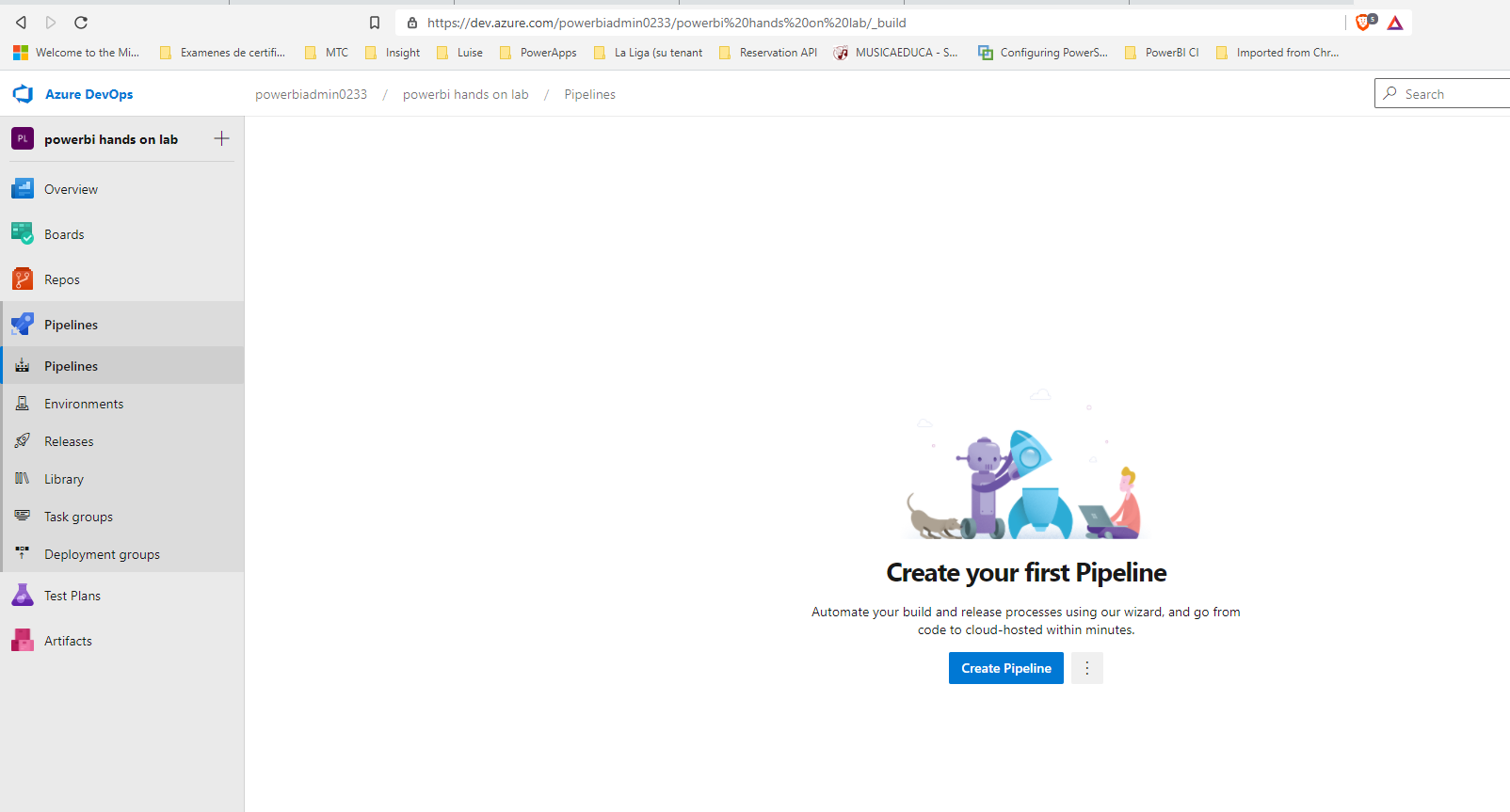


En el centro de administración de PowerBI, damos permisos al SPN

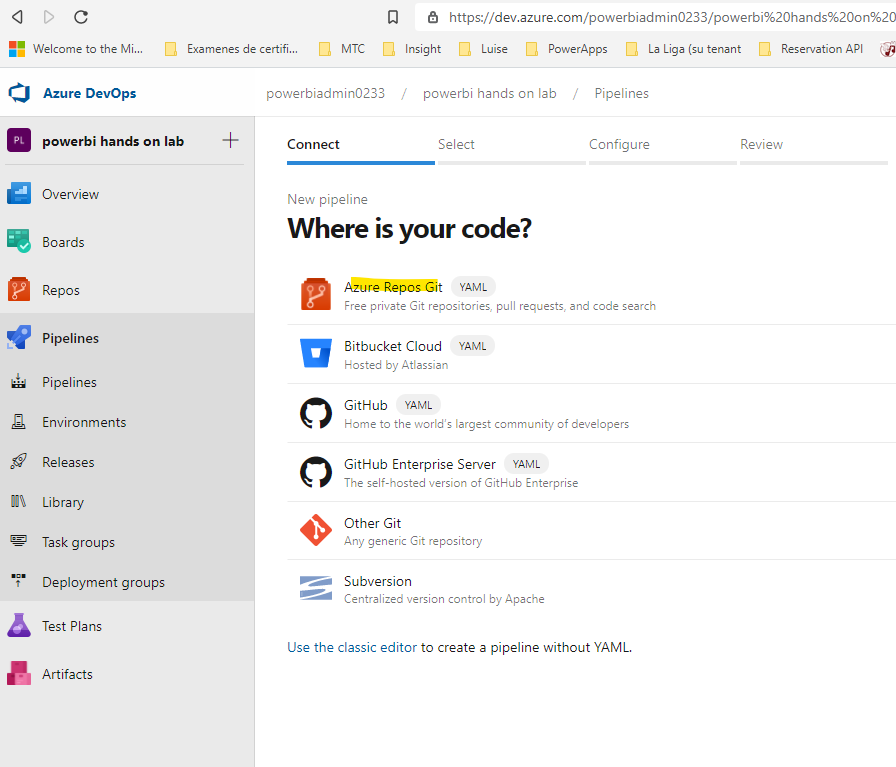


## Creación del pipeline de CI

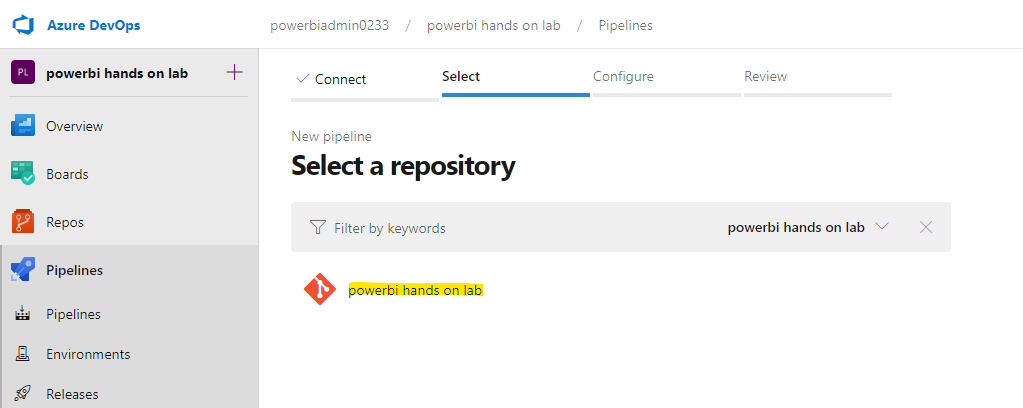
En Azure DevOps, vamos a las sección de Pipelines y le damos a crear Pipeline



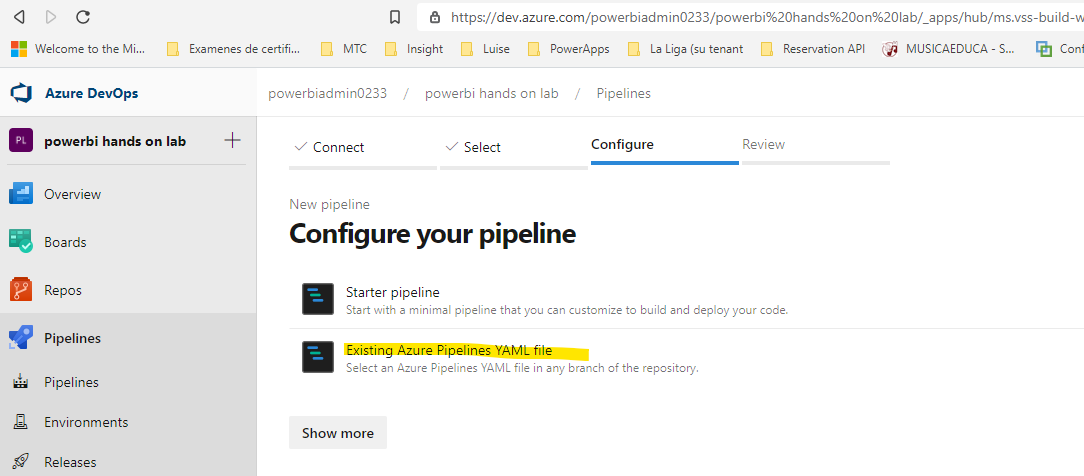
Seleccionamos que nuestro código esta en Azure Repos Git



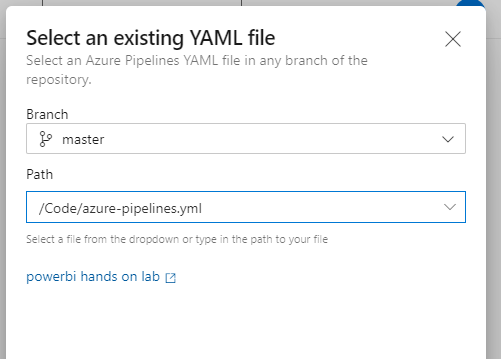
Seleccionamos nuestro repositorio.



Le decimos que lo cree desde un YAML existente

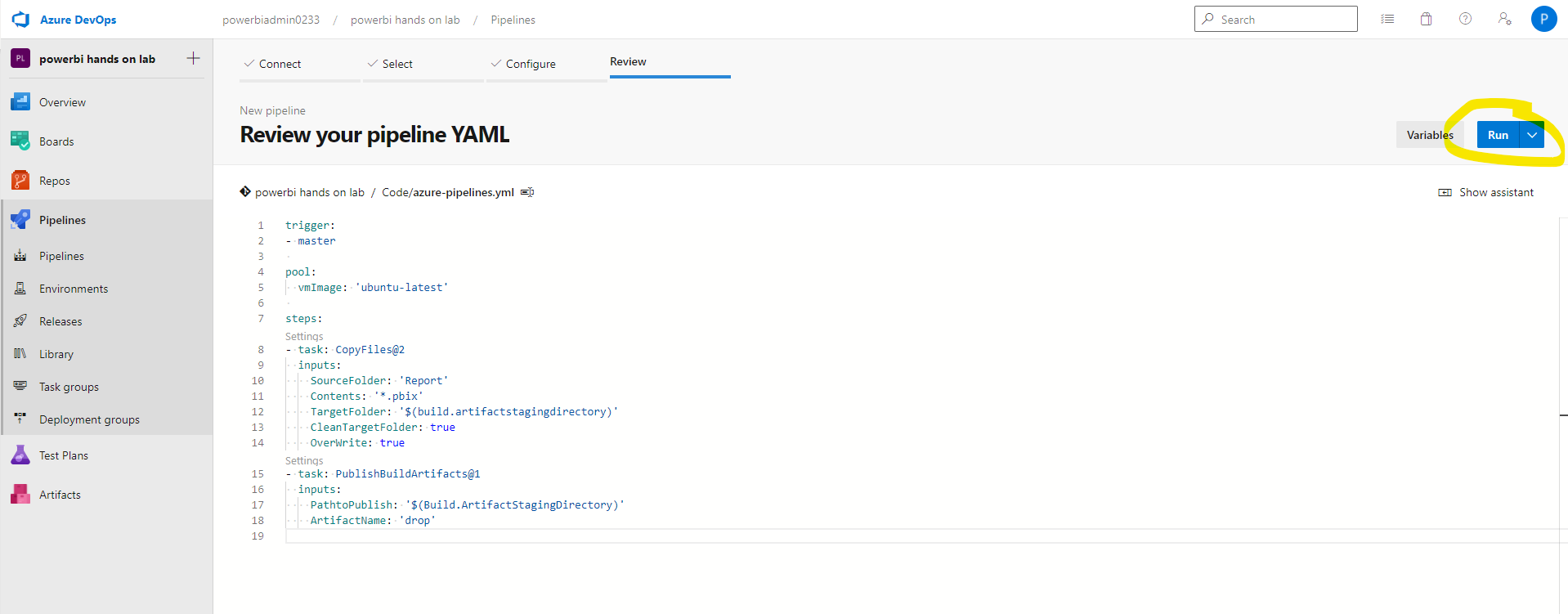


Seleccionamos el YAML que hay dentro de la carpeta /Code/



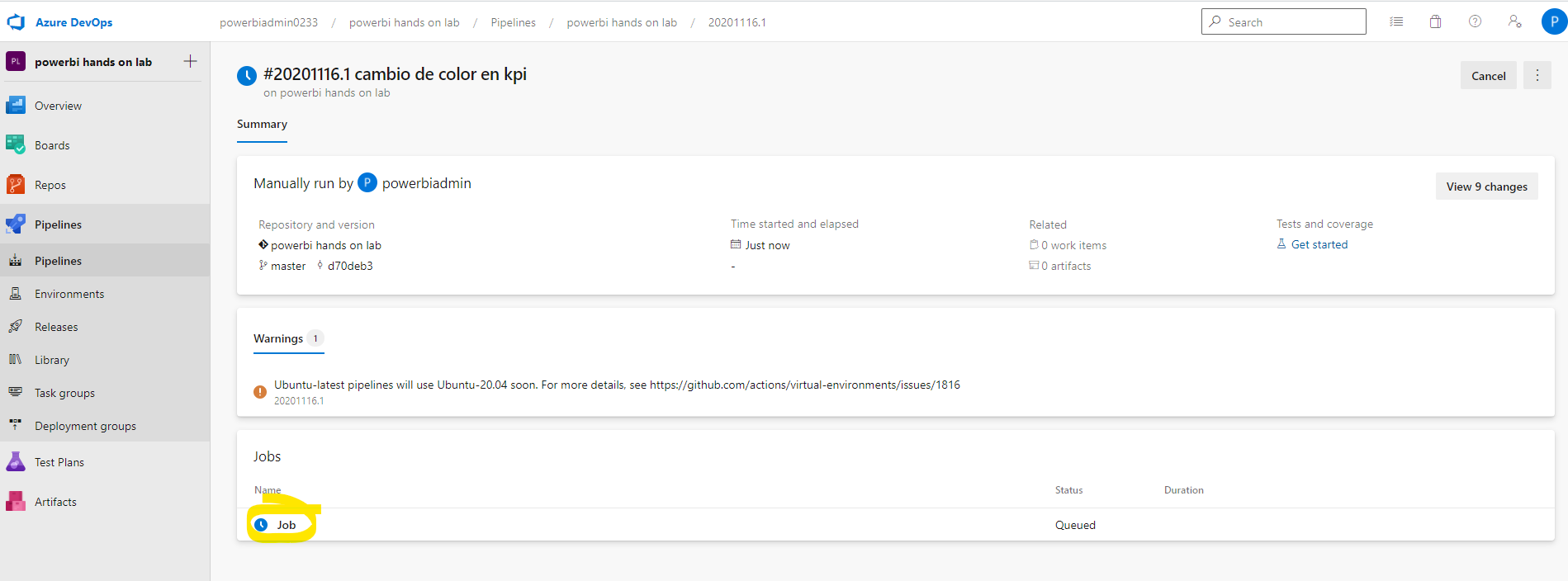
Se cargara el archivo YAML y nos lo muestra en modo edición por si queremos hacer alguna modificación.

Si leemos el YAML, veremos que lo único que hace es copiar los archivo PBIX que existen dentro de la carpeta /Report/ del repositorio y luego publicarlo en la zona de Staging de Azure DevOps. De esta zona es de donde se alimentan los Pipelines de Publicacion.



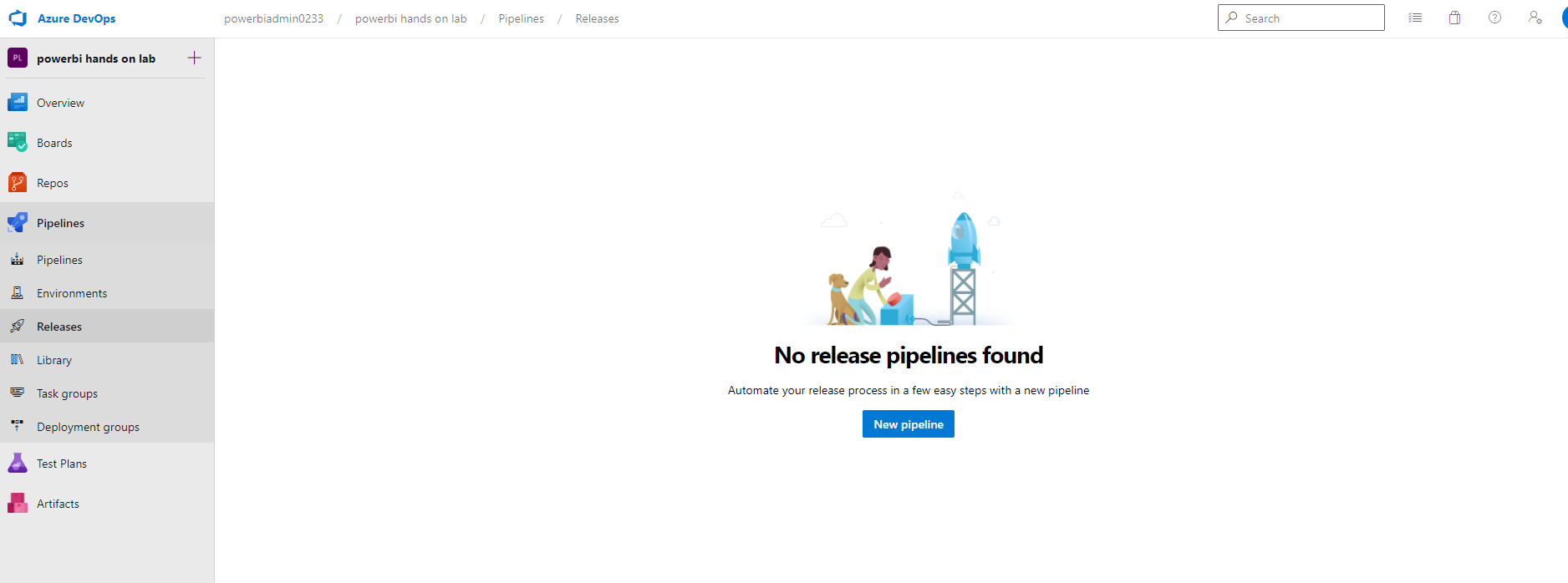
Pulsamos sobre RUN para comprobar que el Pipeline funciona. Esta acción, también lo graba.

Si pulsamos sobre job, veremos las tareas que realiza el agente de compilación

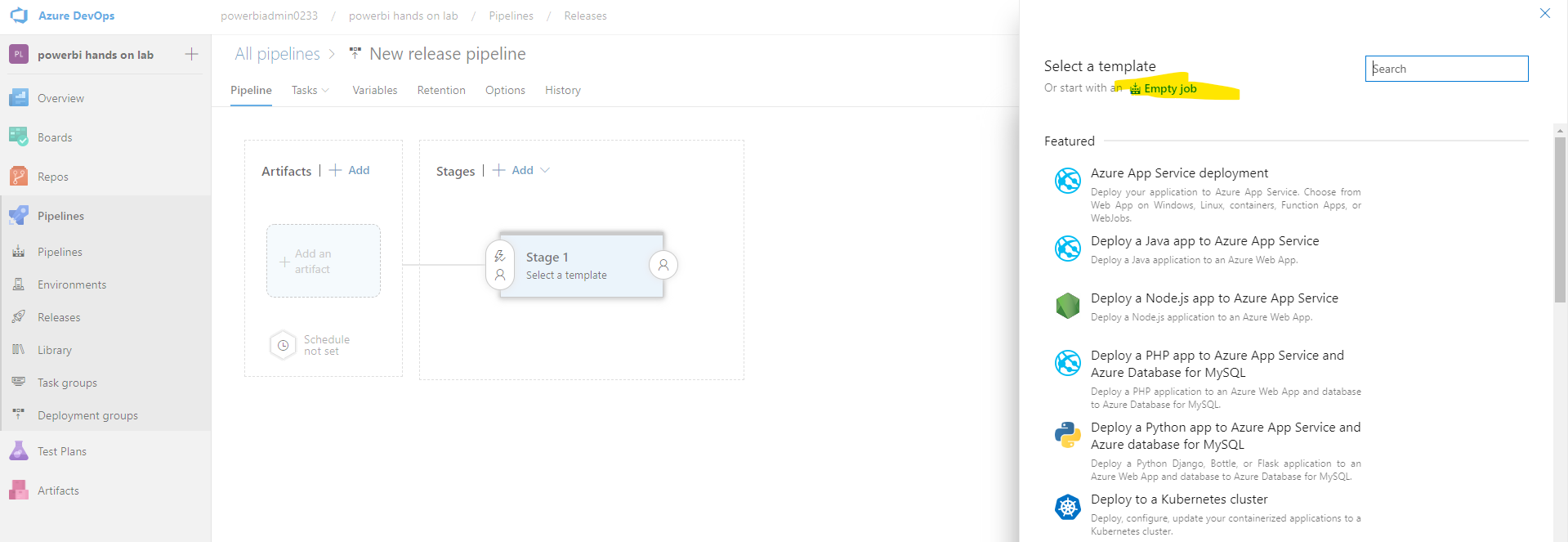


## Pipeline de CD

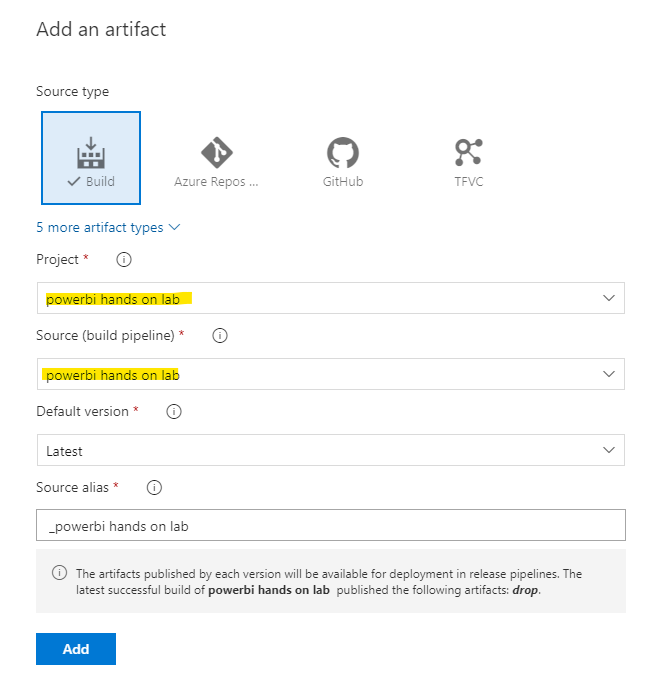
Para crear el pipeline de despliegue, vamos a la pestaña ‘Releases’ y pulsamos sobre ‘New Pipeline’



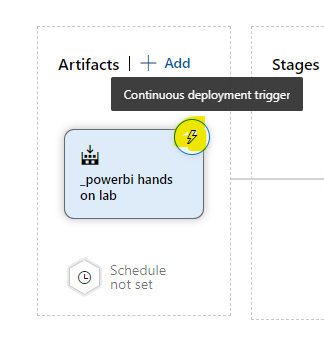
Seleccionamos ‘Empty Job’

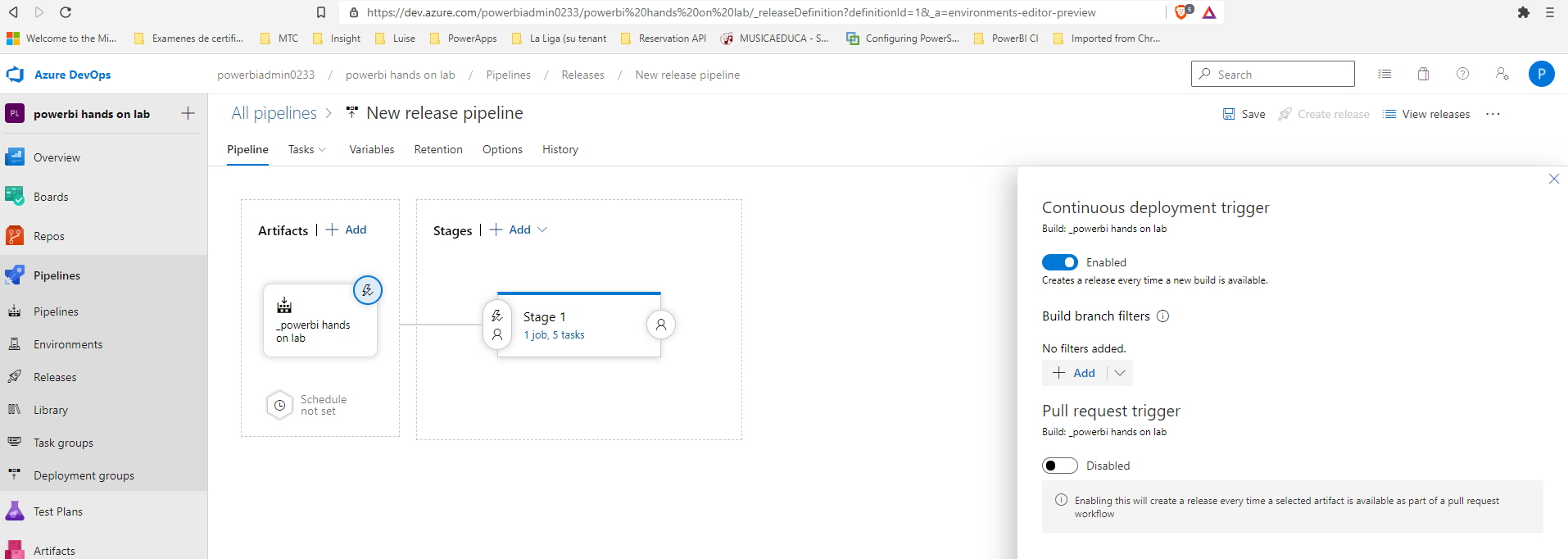


Añadimos nuestro report como artefacto. Seleccionamos **build** como source Type, Nuestro proyecto, y la pipeline creada en el paso anterior.

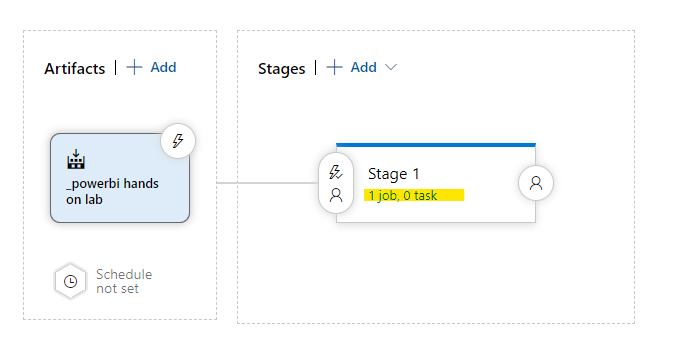


Una vez seleccionado el artefacto, marcamos el rayo de la esquina superior derecha y seleccionamos ‘Continuous Deployment Trigger’, para habilitar el despliegue continuo, es decir, que cuando aparezca un nuevo artefacto, se lancen las tareas de este pipeline.

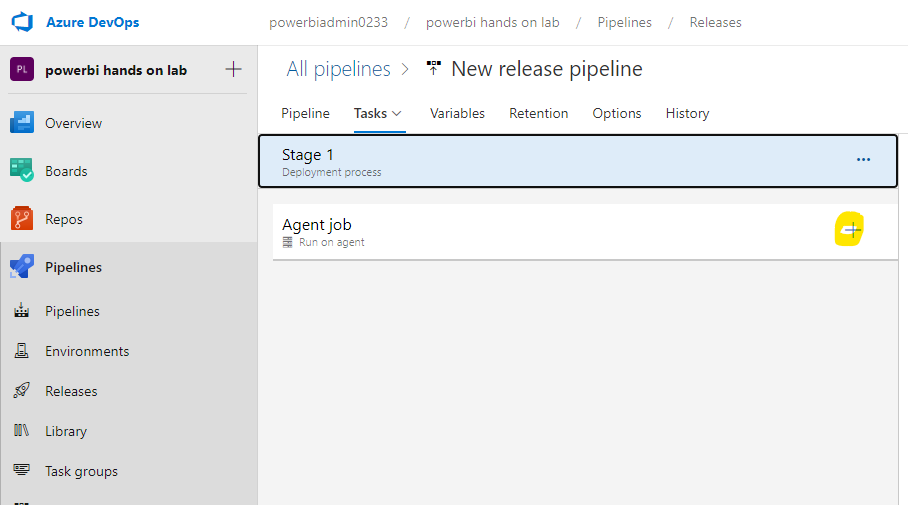




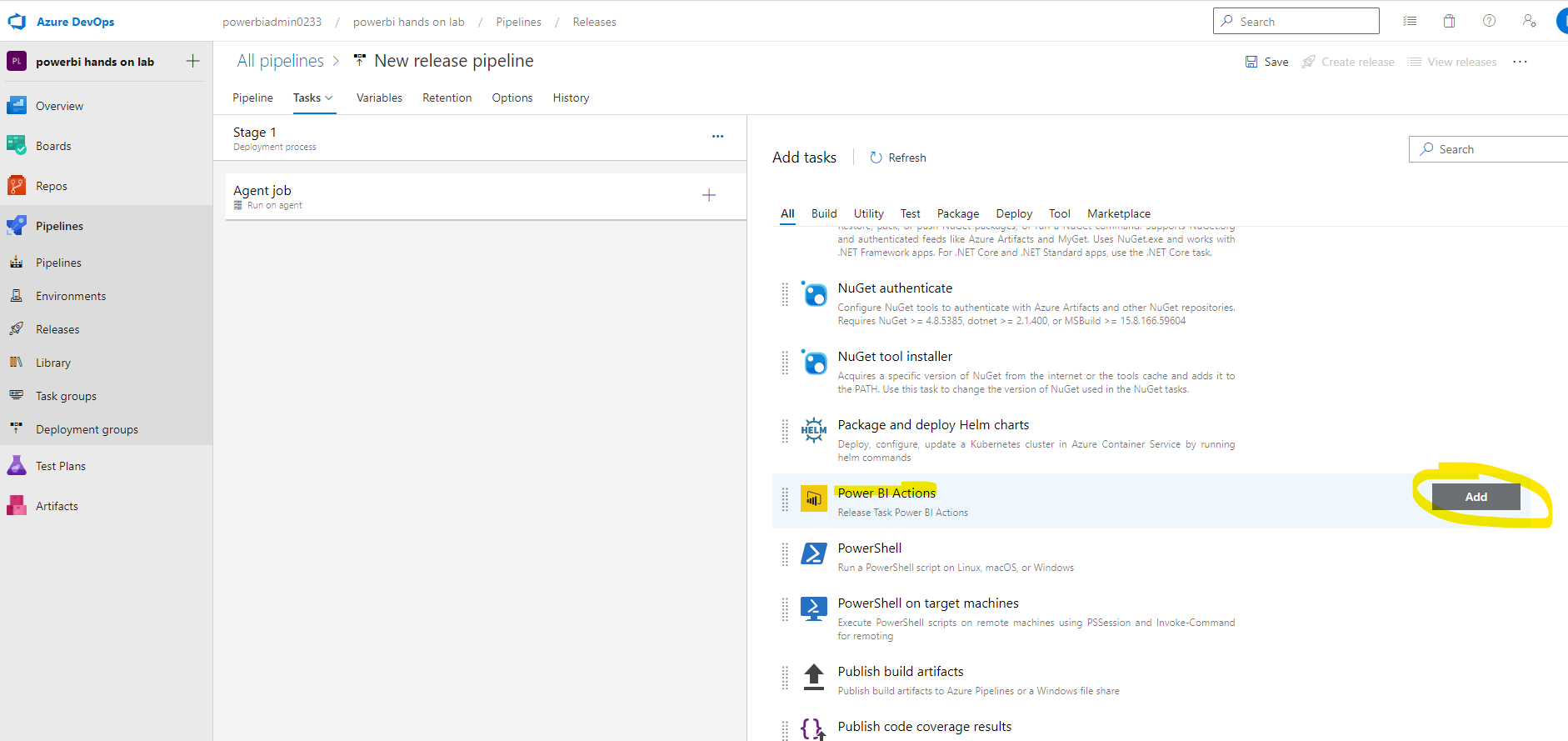
Pulsamos sobre ‘Task’ en la sección de Stage 1



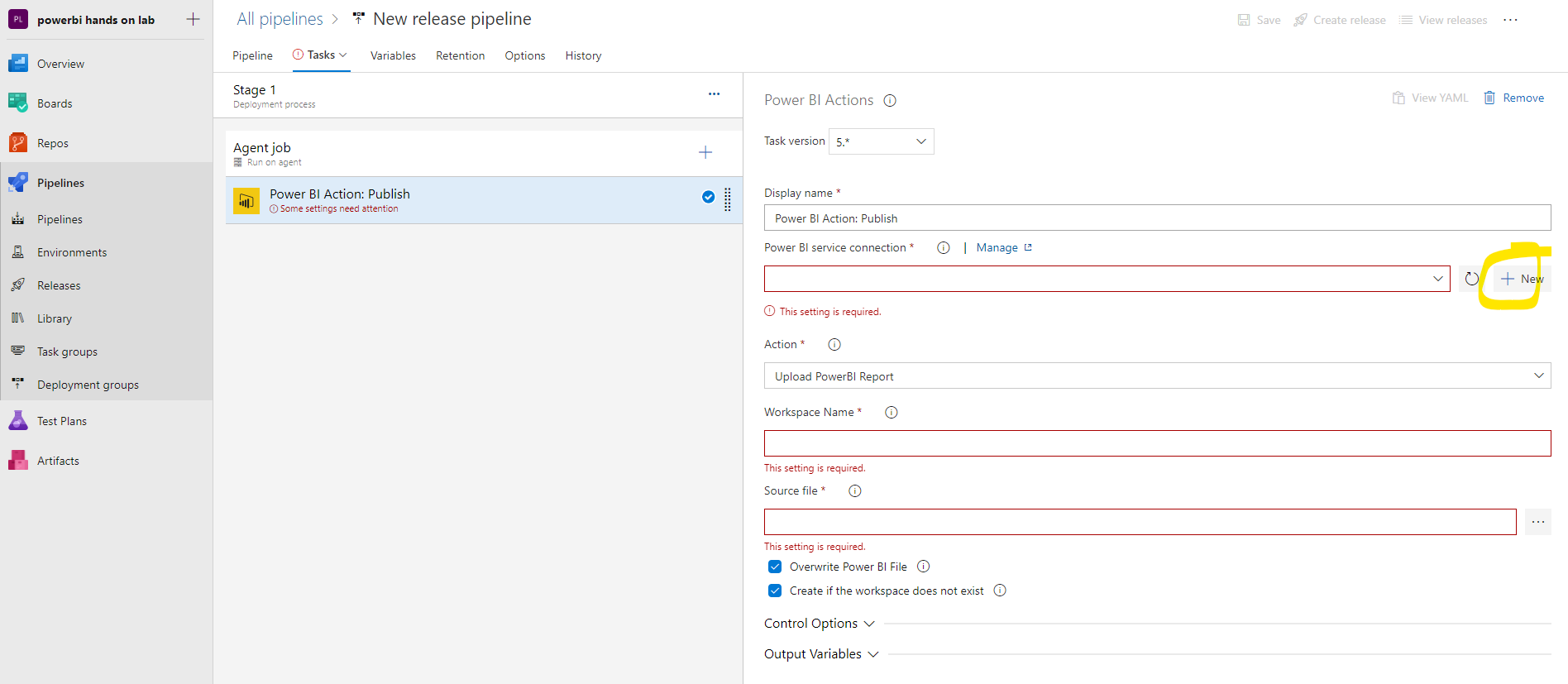
Y pulsamos sobre el signo + del Agent Job para añadir tareas.



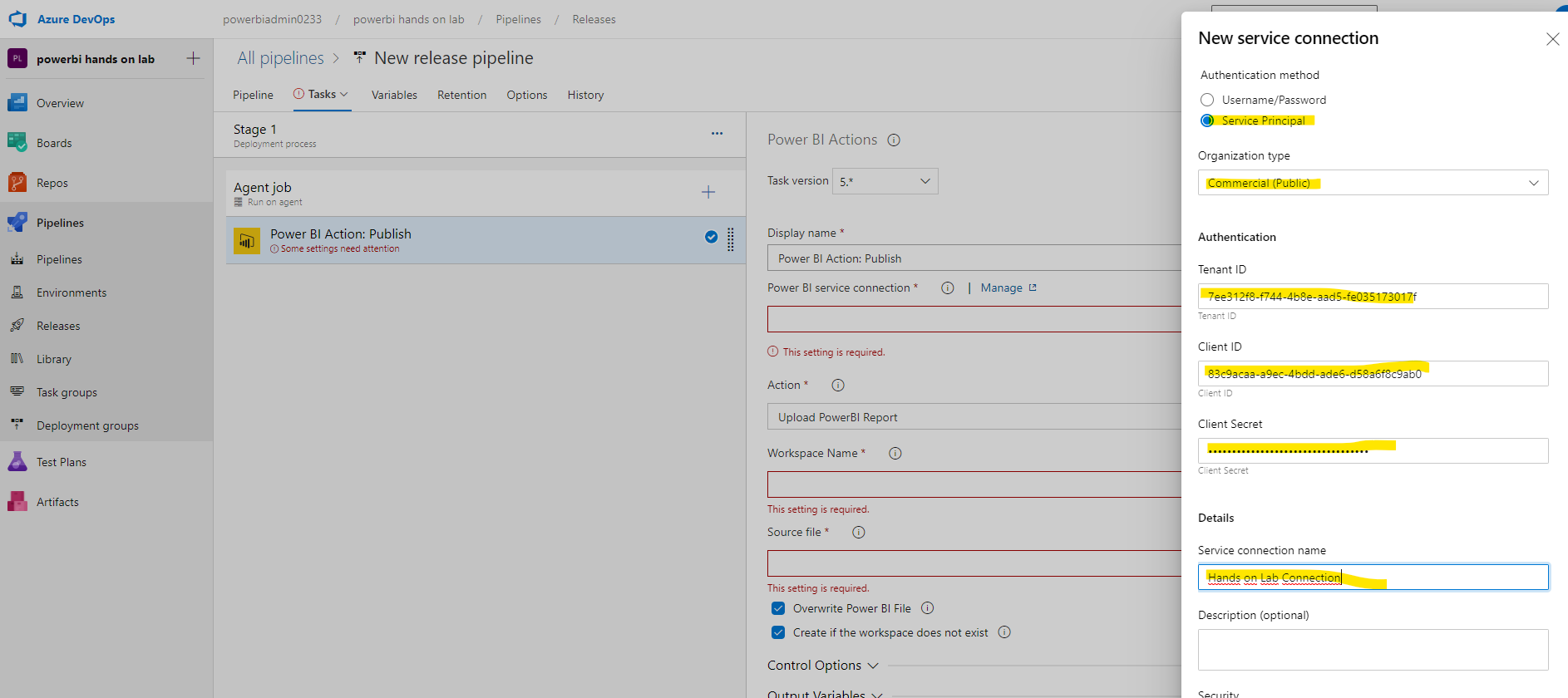
Buscamos la tarea ‘**Power BI Actions’** y la añadimos



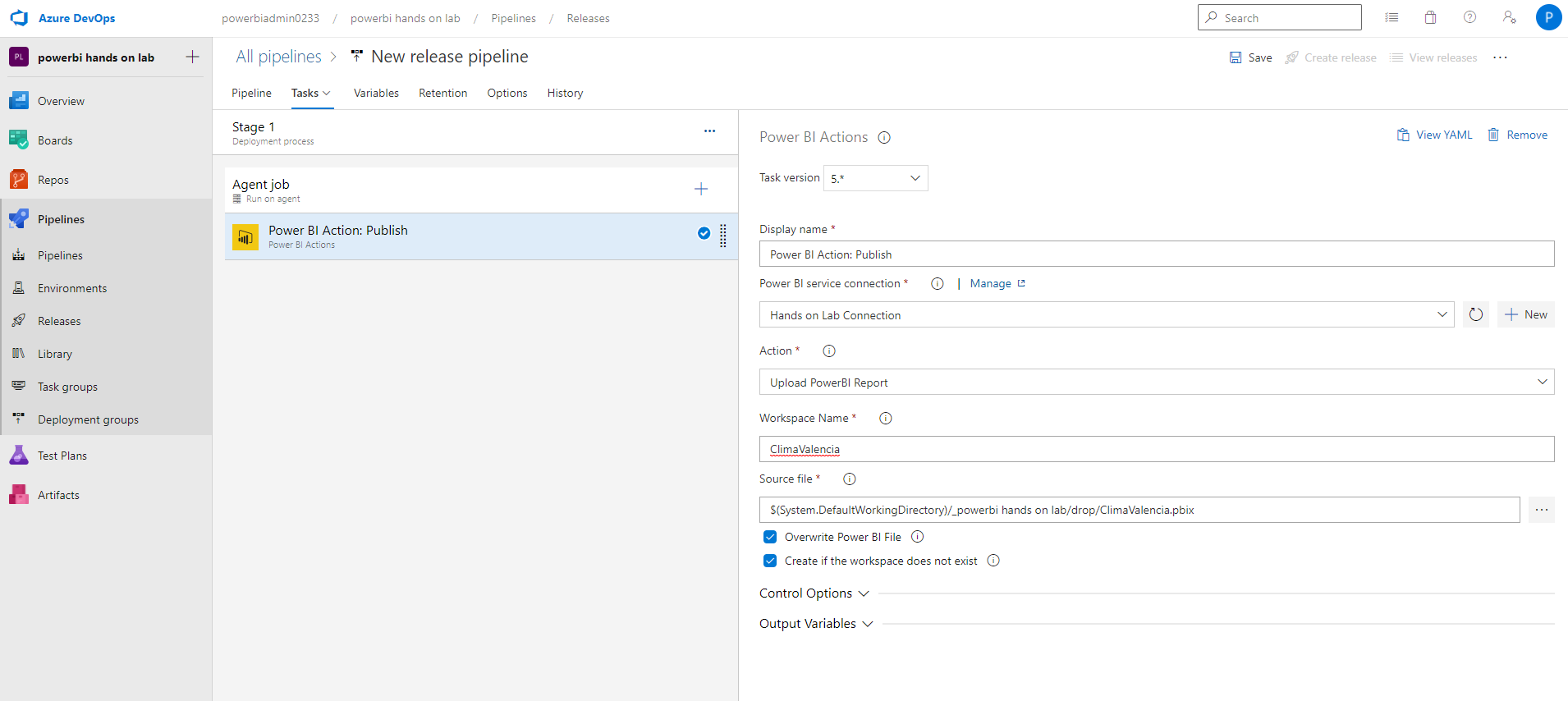
Una vez añadida la tarea, pulsamos sobre ella para configurarla. Al ser la primera vez que usamos PowerBI Actions, tendremos que configurar la conexión a nuestro entorno PowerBI. Pulsando en New



Rellenamos el formulario con los datos que tenemos en el archivo SPN.txt del repositorio y salvamos.

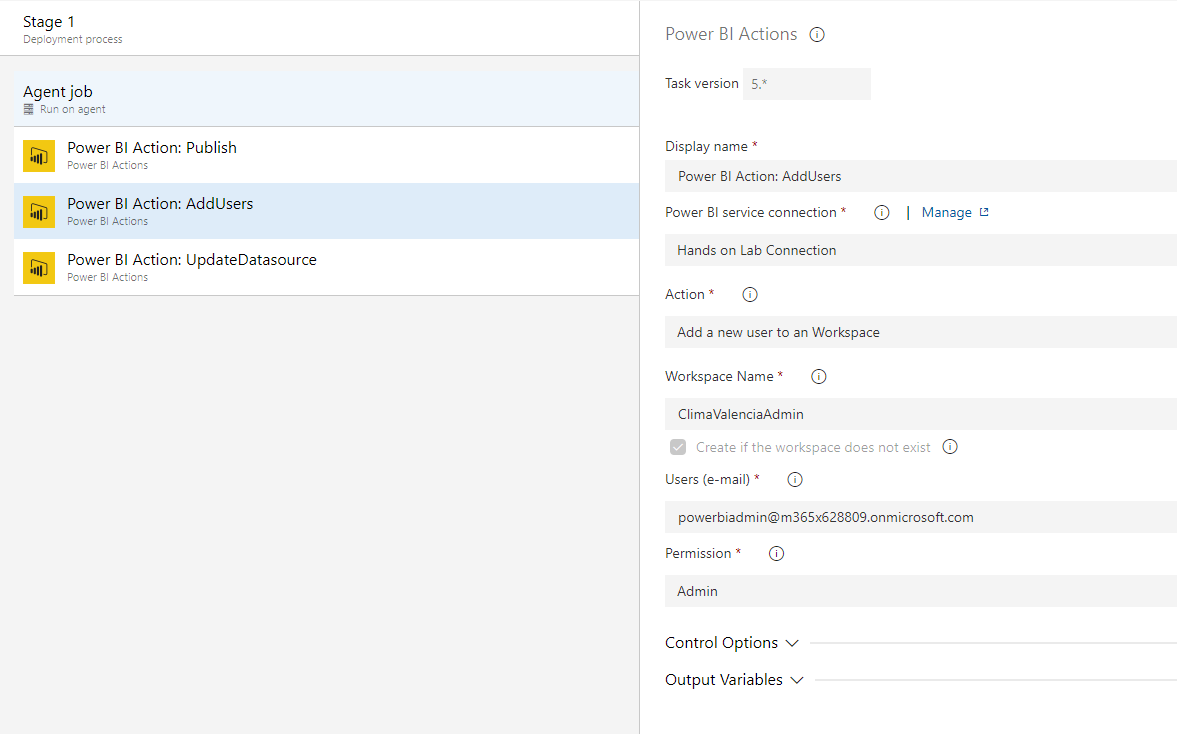


Una vez creada la conexión, configuramos la tarea con los siguientes parámetros. En el WorkspaceName, ponemos ClimaValencia<userid>

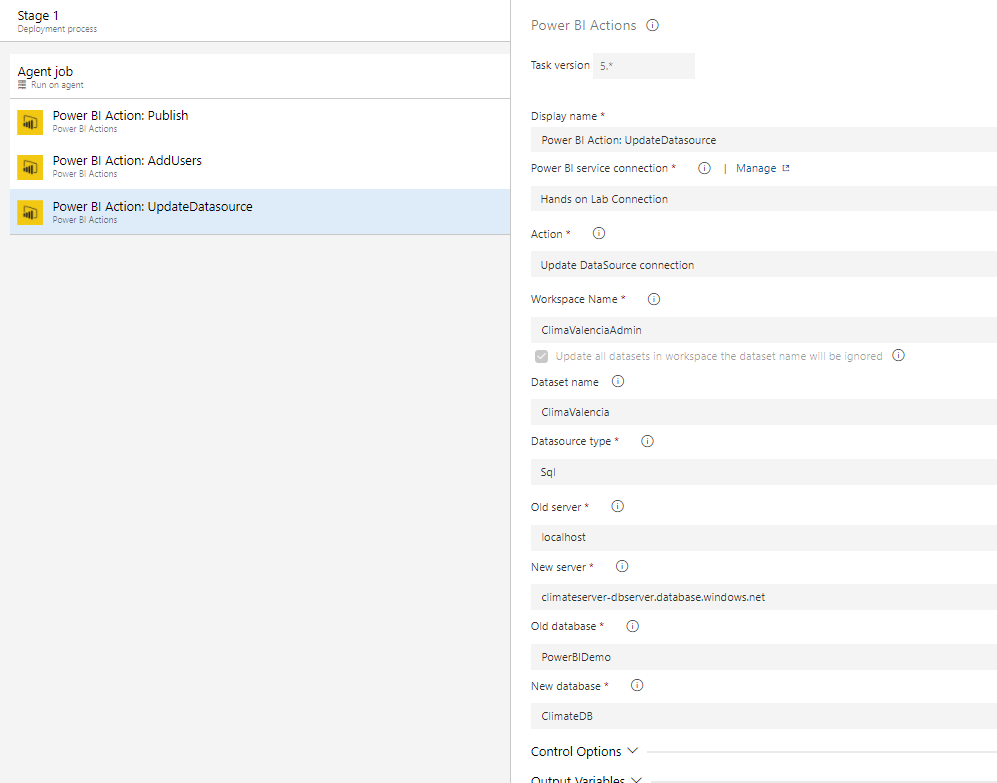


Una vez creada esta tarea, añadimos dos tareas mas de tipo Power BI Actions con las siguientes configuraciones.

La primera nos añade como administradores al Workspace que se va a crear.



La segunda actualiza el dataset para que apunte a la base de datos de producción.



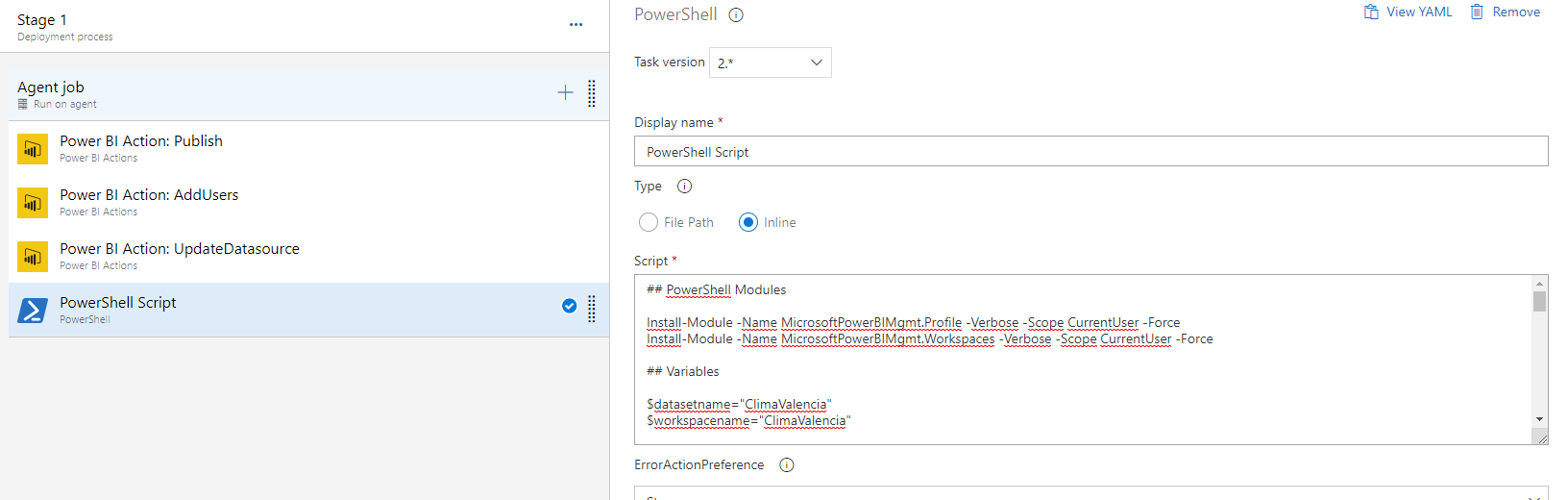
Por ultimo vamos a añadir una tarea de tipo powershell, que actualizara las credenciales del dataset, ya que esto no se puede hacer aun desde la extensión powerbi actions.

Editamos el script powershell que esta en la carpeta Code del repositorio.

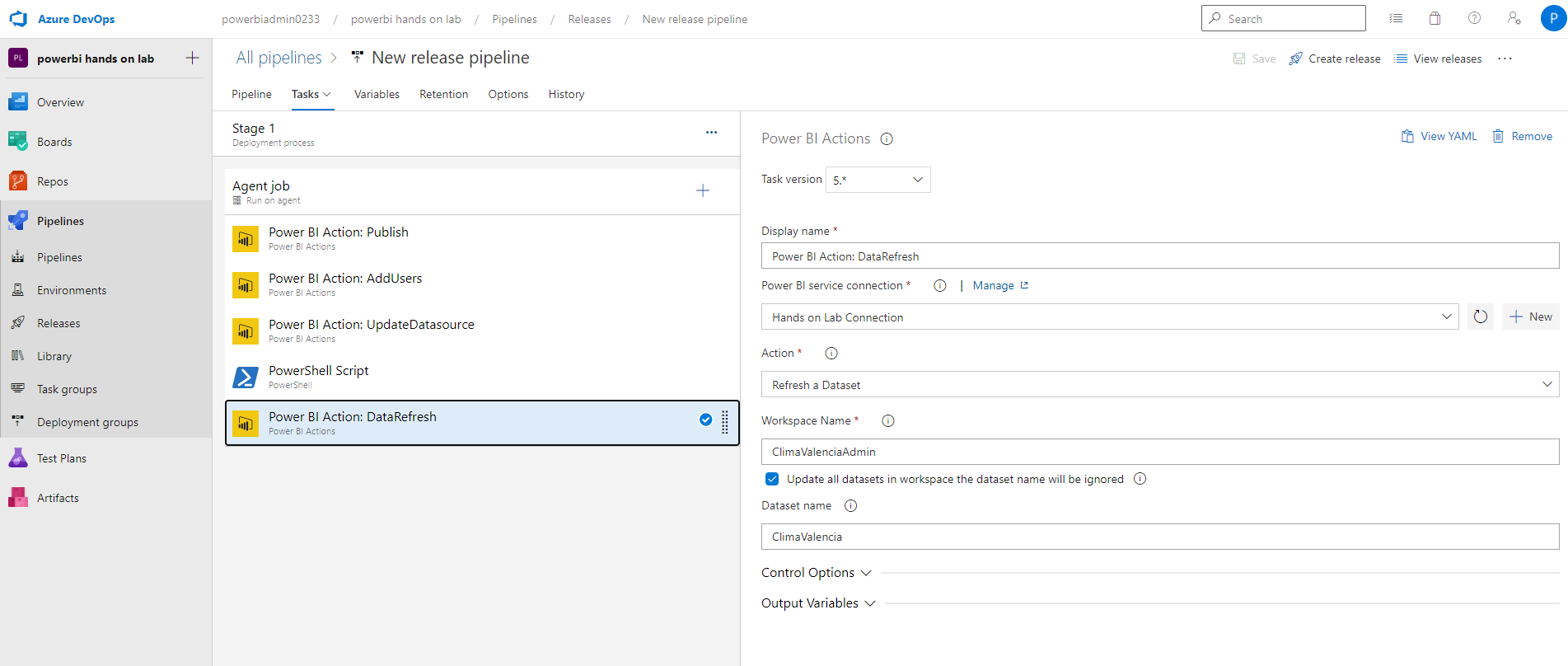
Y editamos las variables WorkspaceName, client\_ID, Tenant\_ID, Client\_sec para que se adapten a nuestro entorno.

Este script powershell primero realiza un TakeOver del Workspace y posteriormente utiliza API de PowerBI para actualizar las credenciales del dataset con el usuario y la contraseña del servidor SQL.

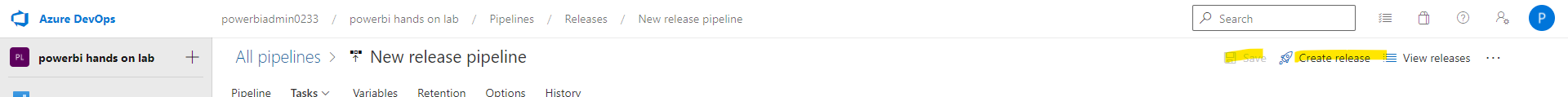
En la tarea, seleccionamos ‘Inline’ y copiamos nuestro script ahí.

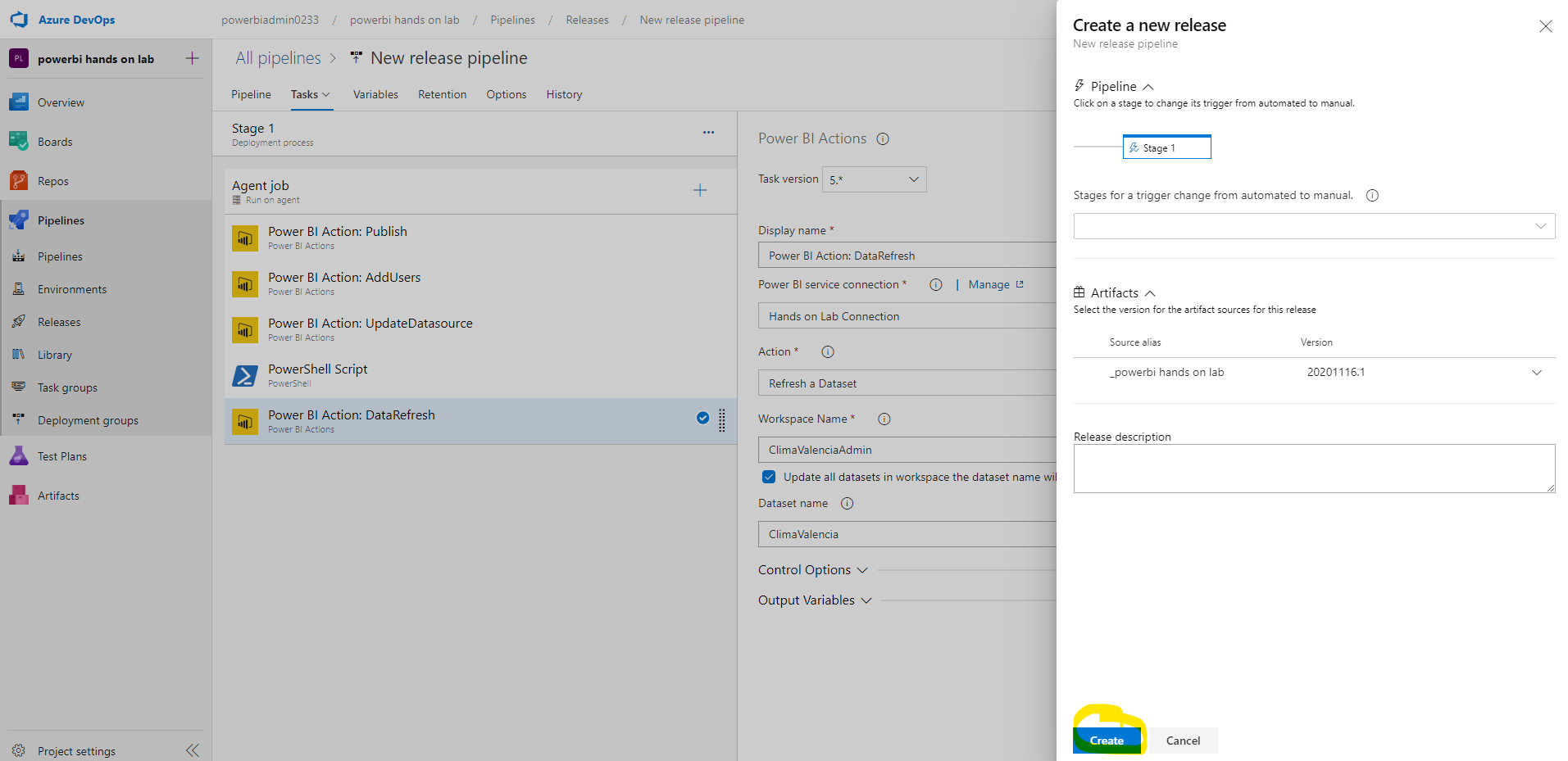
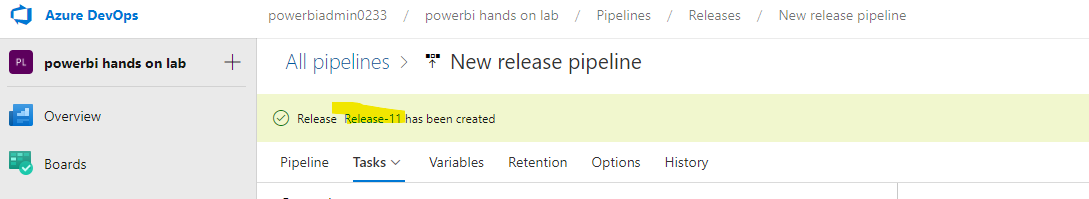


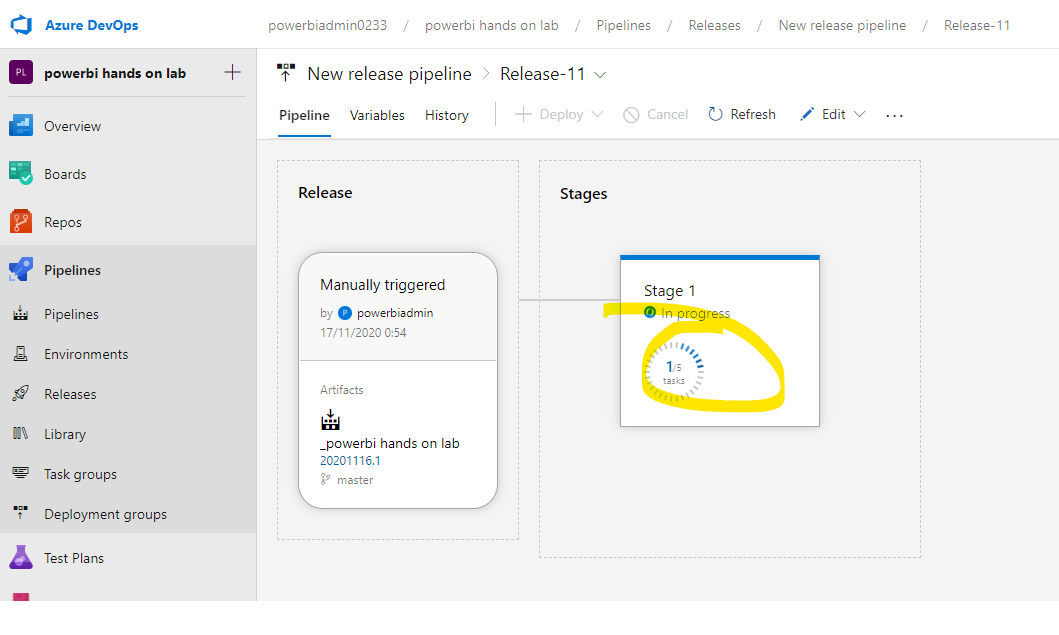
Por ultimo vamos a añadir otra tarea de tipo PowerBI Actions que ejecute el refresco de los datos desde la nueva fuente tras la actualización del data source y sus credenciales.



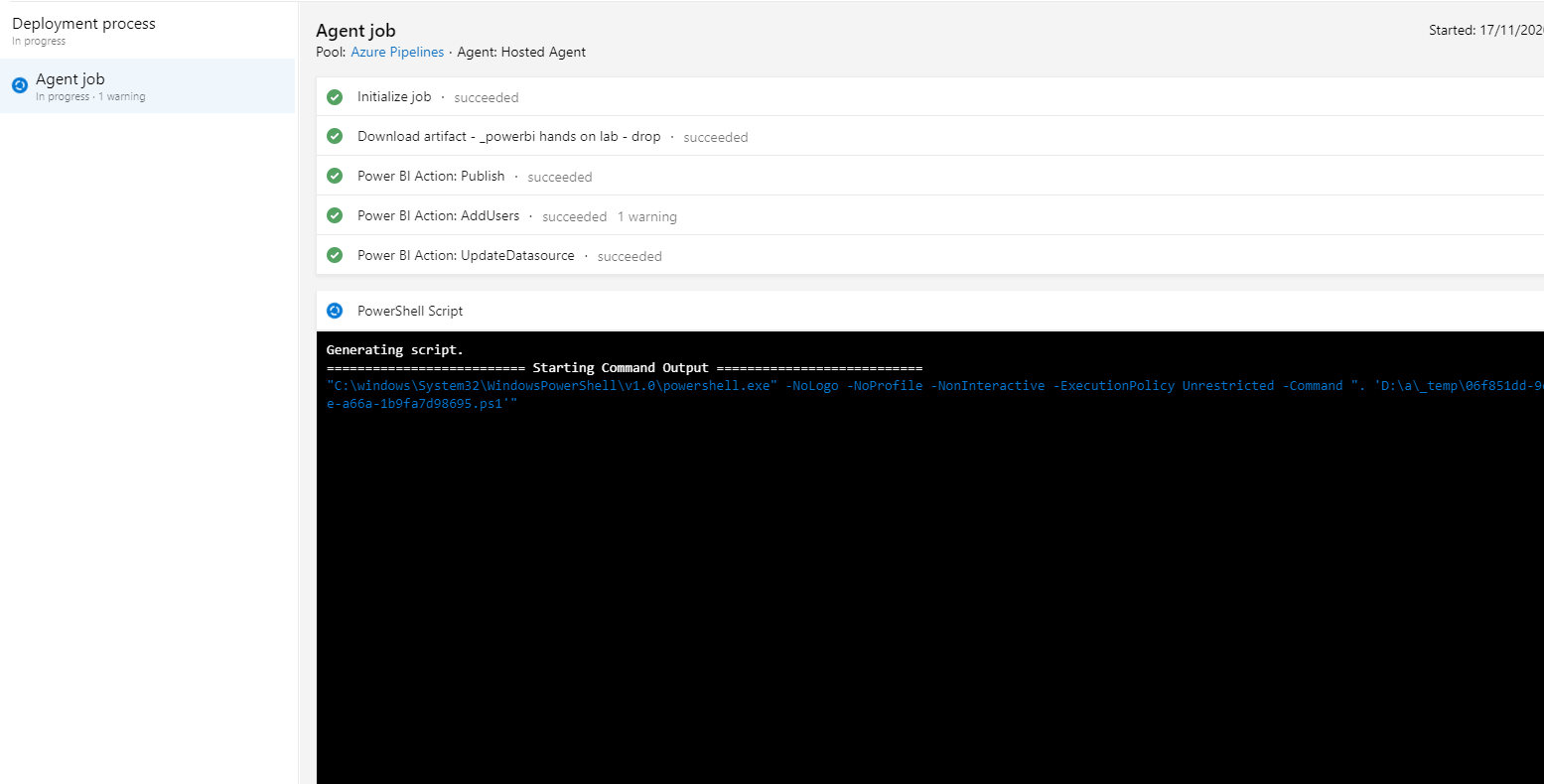
Una vez creadas las tareas, le daremos a Save y posteriormente a créate reléase para ver como funciona.





Si pulsamos sobre la tarea en curso, veremos el log de ejecución en tiempo real.

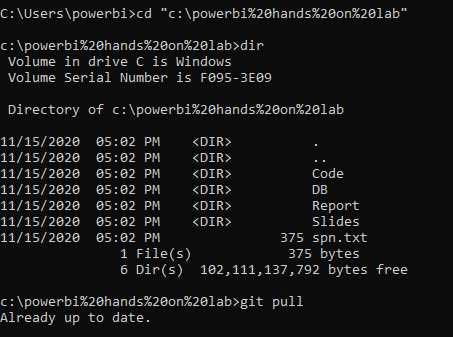


## Todo junto 😉

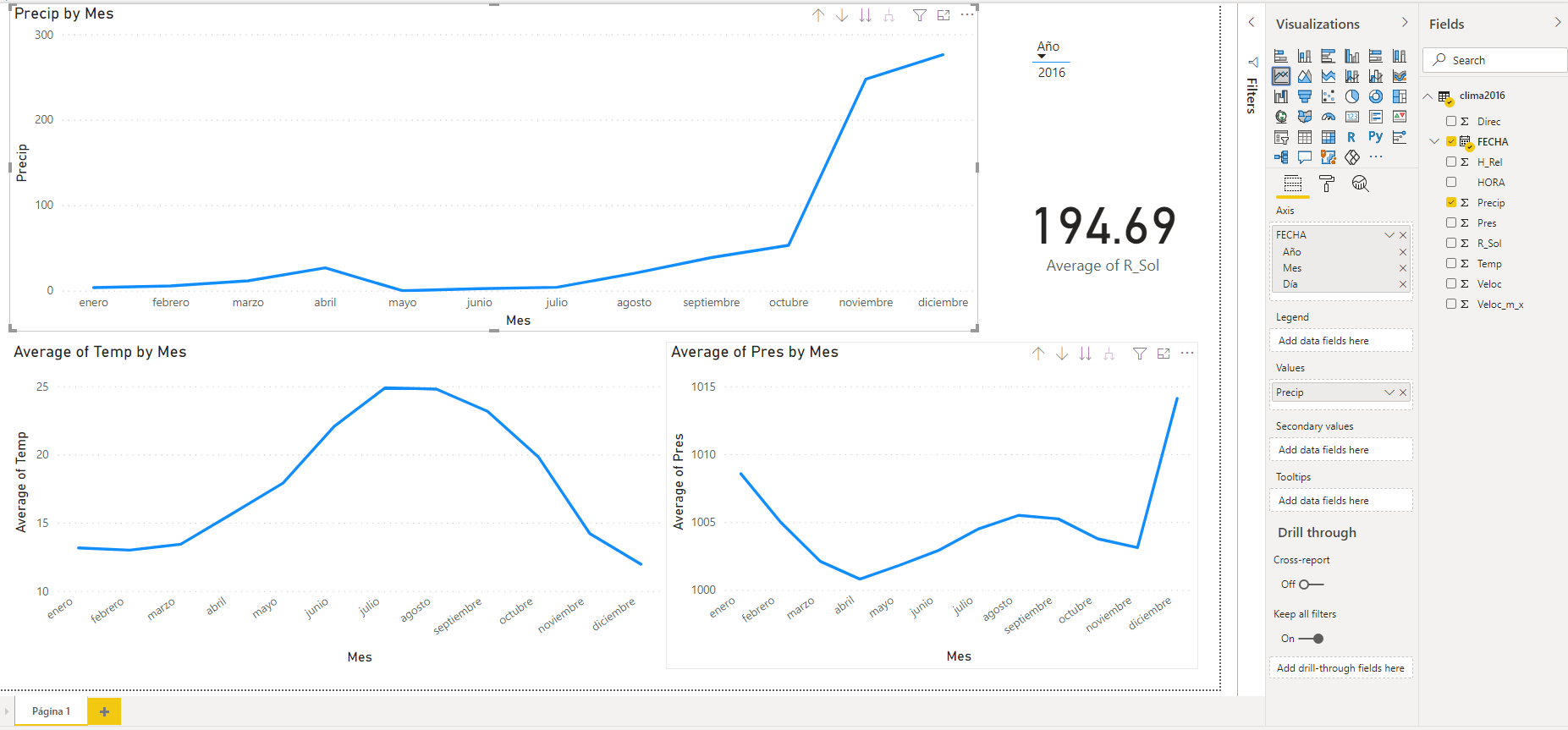
En este ultimo apartado, vamos a actualizar en local el dashboard desde PowerBI Desktop, subirlo a GIT y ver como se desencadena el flujo de despliegue automáticamente.

1.- Realizamos un *git pull* desde el directorio del repositorio para verificar que estamos en la

última versión.



2.- Editamos el report en PowerBI Desktop y cambiamos algo. Por ejemplo cambiamos el KPI ‘Precip by mes’ a grafico de líneas.



Salvamos los cambios.

3.- Actualizamos el repositorio

* Git add \*
* Git commit
* Git push

4.- Vamos a la sección de **releases** de Azure DevOps y veremos que se ha disparado una nueva reléase. Una vez finalizada, actualizamos en PowerBI y tendremos la nueva versión del dashboard.