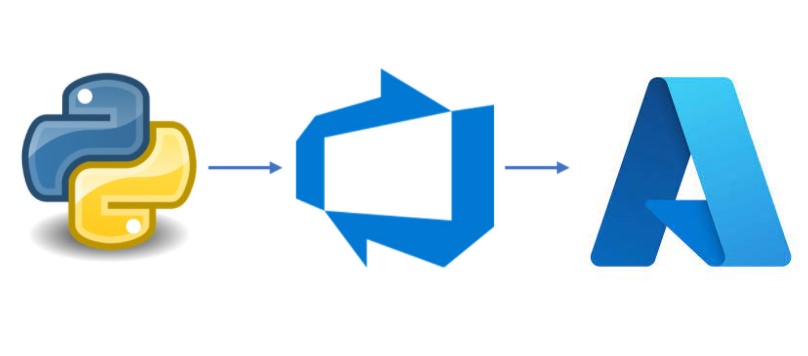
**Desplegar aplicaciónes Python de forma muy sencilla usando Azure**



# Facultad de Informática de Barcelona Grado en Ingeniería

Informática Especialidad de tecnología de la información

25-2-2022

Tutor: Omar Mohamed Mohamed Mahmoud Elkassar

Director: René Serral Gracià

Director GEP: Olga Pons Peregort

**INDEX**

1. **Introducción**
   1. **Contexto**
   2. **Actores**
   3. **Definiciones**
   4. **El problema**
2. **Justificación**
3. **Alcance**
   1. **Objetivos**
   2. **Requerimientos**
   3. **Riesgos**
4. **Metodología**
5. **Planificación temporal**
   1. **Descripción de las tareas**
   2. **Recursos**
   3. **Planificación de riesgos**
6. **Anàlisi de technologies**
   1. **Pipelines**
   2. **Azure** 
      1. **Function App**
      2. **Web App**
      3. **Azure kubernetes service**
      4. **Azure sql**
7. **Ejemplo**
8. **Guia de uso**
9. **Sistema de ficheros**
10. **Ejemplo**
11. **Gestión económica**
    1. **Presupuesto**
    2. **Control de gestión**
12. **Sostenibilidad**
13. **Bibliografía**

# 1. Introducción

Hoy en día cualquiera proyecto informático que sea grande o pequeño o que tenga la mida que tenga, casi seguro, que este proyecto se le ha hecho un proceso de despliegue antes para que obtenga su forma final en producción.

Este proceso de despliegue hay una gran probabilidad que se ha hecho en una plataforma en la nube, en alguna máquina virtual o en algún servicio que ofrece la nube, que está dedicado este servicio a la ejecución, como el servicio de Azure Web App o Function Apps que los ofrece Azure.

## 1.1. Contexto

Dentro del Grado de Ingeniería Informática en la facultad de informática de Barcelona (FIB), hay cuatro especialidades, especialidad de computación, especialidad de tecnología de la información, especialidad de sistema de la información y finalmente especialidad de software.

Este trabajo de fin de grado pertenece a la especialidad de la tecnología de la información.

El trabajo consistirá en ayudar los proyectos pequeños y medianos en cómo obtener una infraestructura dentro de la nube de Azure para tener en marcha sus aplicaciones que están corriendo en Python, sin tener que invertir mucho esfuerzo.

## 1.2. Actores

Aquí hablaremos sobre los principales actores que se van a aprovechar del proyecto, y principalmente son los desenrollados que tienen una aplicación en Python y quieren desplegar esta aplicación y hacer toda la faena necesaria para el proceso del despliegue.

Este proyecto los ayuda a hacer el despliegue de una forma mucho más fácil y rábida.

## 

## 1.4. El problema

En este trabajo se va a desarrollar un conjunto de codigos y también una guía para cómo usar los códigos y customizarlos, para ayudar a los usuarios que están interesados en probar o poner sus aplicaciones en modo de producción con integración continua, lo pueden hacer sin tener que preocuparse mucho por el proceso del despliegue y así concentran aún más en el proceso del desenrollo.

## 2. Justificación

El problema del despliegue del código hoy en día presenta uno de los problemas que están saliendo recientemente, y por esto ahora mismo existe el rol de devops, que antiguamente no existía, la responsabilidad de este rol principalmente consiste en como automatizar el procese para el despliegue del código.

A continuación, vamos a estudiar si es necesario el desarrollo de este proyecte de tal manera como se plantea o no, vamos a estudiar las alternativas y también la alternativa de usar otro servicio de nube.

### Template deployment

Es una solución de Azure que permite desplegar aplicaciones webs en Azure Web App de forma fácil, usando ejemplos predefinidos, pero no permite ni usar Function Apps, AKS, ni base de datos y en esto se diferencia este proyecto

### La forma manual

Siempre se puede hacer todo el proceso manualmente poniendo todas las configuraciones a medida, pero el problema de esto es que antes de empezar a hacerlo, tienes que hacer una formación previa y normalmente el proceso del desarrollo de la solución tarda mucho tiempo.

### Usar otro servicio de nube

En esta solución hemos decidido de tirar por Azure, culpa del ecosistema de Microsoft, que se considera un ecosistema muy completa, porque también incluye el Microsoft 365, que incluye soluciones por el tema de gestión de documentos como el Word y SharePoint y también soluciones por el tema de comunicación entre empleados como Teams y Yammer y mucho más, y si en un futuro la empresa se decide hacerse más grande se le ira super bien usar el Microsoft 365.

### Estudios teóricos

[https://medium.com/globant/how-to-create-and-deploy-a-python-azure-function-usingazure-devops-ci-cd-2aa8f8675716](https://medium.com/globant/how-to-create-and-deploy-a-python-azure-function-using-azure-devops-ci-cd-2aa8f8675716) [https://docs.microsoft.com/en-us/azure/devops/pipelines/ecosystems/pythonwebapp?view=azure-devops](https://docs.microsoft.com/en-us/azure/devops/pipelines/ecosystems/python-webapp?view=azure-devops)

<https://microsoft.github.io/PartsUnlimited/iac/200.2x-IaC-CDAzureSQLdbwithVSTSandVS.html>

[https://docs.microsoft.com/en-us/azure/devops/pipelines/ecosystems/kubernetes/akstemplate?view=azure-devops&pivots=pipelines-yaml](https://docs.microsoft.com/en-us/azure/devops/pipelines/ecosystems/kubernetes/aks-template?view=azure-devops&pivots=pipelines-yaml)

Estos tutoriales están super bien explicado y son muy funcionales para poder llegar al mismo resultado que puedes llegar usando este proyecto, pero la diferencia es que usando mi proyecto puedes encontrar todo el material que necesitas en uno solo lugar, la guía este mucho más optimizada para tus aplicaciones de Python.

## 3. Alcance

En este apartado vamos a hablar de los objetivos y subobjetivos y también de los

Requerimientos de este proyecto.

### 3.1. Objetivos

El objetivo principal de este trabajo es el desarrollo de una seria de códigos como a infraestructura y una guía para que el usuario pueda usar estos códigos, para poder crear la infraestructura necesaria para el despliegue de sus aplicaciones y después definir los pipelines para automatizar el proceso de la integración de los nuevos cambios de códigos, de tal manera que lo único que tiene que hacer el usuario es subir los cambios al Azure Repo y el proceso de poner los códigos en funcionamiento en la nube se hace de forma automática.

Aquí están explicados los subobjetivos

* El usuario puede crear un clúster de K8s, para que después publica sus microservicios en este mismo clúster.
* Crear un recurso de Function App y después usarlo para ejecutar los códigos allá.
* Crear un servicio de Web App y publicar las aplicaciones web allá.
* Crear un servicio de Azure Sql Databases, para poder usarlo para gestionar bases de datos allá.
* Configurar Azure Repos y Azure Pipelines para poder hacer los despliegues de forma automática.

### 3.2. Requerimientos

Ahora vamos a hablar sobre los requerimientos funcionales y n o funcionales

#### Requerimientos funcionales

Aquí vamos a describir los requerimientos funcionales para este proyecto

* Entender cómo funciona los diferentes recursos usadas para el despliegue en este proyecto (AKS, Function App, Azure Sql Database y Web App).
* Poder proporcionar un código como infraestructura que se puede usar para desplegar estos recursos y se puede reusar más de una vez.
* Entender como funcionen los Azure pipelines, para poder usarlos para desplegar los recursos y los códigos dentro de los recursos de forma automática.

#### Requerimientos no funcionales

Aquí vamos a describir los requerimientos no funcionales para este proyecto

* Tener una suscripción de Azure para poder hacer pruebas
* Saber cómo crear una guía fácil de leer y entender
* Preocuparse de que el código como a infraestructura se puede reusar más de una vez por diferentes usaurios.

### 3.3. Riesgos

Cualquier proyecto lleva riesgos, pero es muy importante poder clasificar los riesgos desde el principio, ya que así te ayuda de ir con más cuidado y plantear soluciones para ellos.

* Gastarse mucho dinero con Azure haciendo pruebas, ya que Azure no es gratis.
* Después de crear la infraestructura como a código, la configuración de los recursos se cambia y estos códigos ya dejen de funcionar.
* Borrar algún parte del código fuente de la aplicación antes de tenerlo en Azure repos.

## 4. Metodología

El desarrollo de este trabajo está ligado con la investigación de Azure, y no es un proyecto muy grande donde trabajan muchas personas (solo estoy yo), por esto es muy importante usar una metodología que no sea muy complicada y que se usa para proyectos pequeños y el equipo está formado por solo por una persona.

por esto hemos decidido de ir por la metodología scrum y modificarla para que sea solo

por una persona, que nos ayudara en planificar y organizar las diferentes tareas.

Cada 2 semanas (cada sprint) se hará reunión de máximo una hora conmigo mismo, en este reunión se decidirá las tareas que se tiene que hacer en los próximos 14 días y también se evaluar las tareas que se han hecho en los últimos 14 días

Cada día se hará una reunión conmigo mismo en esta reunión se decidirá la faena que se tiene que hacer en este mismo día

Para la organización de la metodología scrum voy a usar Azure bordas, que es un servicio gratuito de Azure para organizar faena usando la metodología scrum.

## 5. Planificación Temporal

## 

En este apartado vamos a explicar la planificación inicial de las tareas y también la planificación actual debido a la desviación del plan inicial

**5.1. Planificación Inicial**

*En este aparatado* vamos a explicar la planificación del plan inicial

## GP - Gestión del proyecto

Este apartado consiste en hacer toda la faena que va relacionada con la planificación y la gestión de cómo hacer el trabajo, esta fase se considera como la fase inicial.

### • *Objetivos (GP1)*

Antes de empezar con cualquier proyecto lo más importante es definir los objetivos del trabajo.

Esto se divide a 2 tipos de objetivos, los objetivos generales que se defienden al principio del proyecto, y los subobjetivos que se definen en el principio de cada sprint por lo que se quiere conseguir en este sprint, se estima que esta tarea llevara a capo unas 18 horas en total.

### • *Planificación (GP2)*

*En esta tarea vamos a definir como hacer el trabajo y se planificará como gestionar los previstos que, consisten en hacer una sesión diaria (yo solo), que en esta sesión voy a escoger las tareas que se tiene que hacer en este mismo día, y también dividir las tareas en subtareas si es necesario, se estima que esta tarea conllevará unas 50 horas.*

### • *Presupuesto (GP3)*

*Aquí vamos a detallar el presupuesto que va a acostar para hacer el proyecto, que incluye el material necesario para poder desarrollar el proyecto, tanto los personales como los gastos de Azure, se calcula que son unas 20 horas.*

### • *Sostenibilidad (GP4)*

En este apartado vamos a crear un informe para estudiar los efectos del proyecto sobre la sociedad y medioambiente, se estima que unas 17 hora para completar la tarea.

### • Documentación (GP5)

Es muy importante poder documentar la faena realizada para poder volver a modificar lo que ya está hecho o bien para valorar si otra persona tiene que ponerse con el código, también se incluye el tiempo invertido en desarrollar la guía de cómo usar el proyecto, se estima que se invertirán unas 75 horas.

o (GP5.1) Hacer la documentación del trabajo conllevará unas 40 horas

o (GP5.2) Hacer la guía de cómo usar los códigos para hacer el despliegue serán unas 35 horas

### • Presentación (GP6)

Una vez el trabajo ya finalizado, es muy importante poder preparar la presentación, esto se estima que son unas 10 horas.

## TP – Trabajo Previo

Aquí vamos a hablar sobre las tareas necesarias hacer, antes de empezar con el desarrollo.

### • Estudiar el entorno de Azure (TP1)

### El trabajo tratará de estudiar el entorno de Azure y como creamos los recursos y desplegamos las aplicaciones dentro de Azure, estudiando e indagando más sobre Azure y Azure pipeline, se calcula que son unas 50 horas.

### • Preparar el entorno del trabajo (TP2)

Esta tarea consiste en preparar la suscripción de Azure para poder trabajar con Azure, Azure no es un entorno gratis, sino que es de pago, pero, existen otras maneras para poder utilizarlo gratuitamente (como usar una suscripción de estudiantes), se calcula que serán unas 10 horas.

## DP – Desarrollo del Proyecto (DP)

En esta fase vamos a explicar las tareas que llevaran a cabo el proceso del desenrollo del proyecto.

### • Preparar la infraestructura de AKS(DP1)

Aquí vamos a preparar todo el material que necesitará el usuario para poder desplegar su aplicación de Python en AKS, calculamos que son unas 125 horas.

* (DP1.1) Crear el clúster del aks y también el pipeline del despligue del aks son 30 horas.
* (DP1.2) Preparar los workload donde se va a correr las aplicaciones en el clúster, se estima que son 30 horas.
* (DP1.3) Preparar los endpoints donde se va a llamar para acceder a las aplicaciones, serán unas 40 horas.
* (DP1.4) Preparar toda la infraestructura como a código para qué después, los pequeños y medianos proyectos la puedan usar para su despliegue de aplicaciones, se considera que son 25 horas.

### • Preparar la infraestructura de Azure web app(DP2)

Aquí vamos a hacer toda la faena necesaria para que el usuario pueda añadir sus web apps en las web apps de Azure de la manera más sencilla, se estima que serán unas 55 horas.

* (DP2.1) Crear el Azure web app y también el pipeline del despligue serán unas 30 horas.
* (DP2.2) Importar la infraestructura como a código para poder usarlo después en los proyectos, se calcula que son 25 horas.

### • Preparar la infraestructura de Function App (DP3)

Esta es la tarea que vamos a trabajar para que el usuario pueda poner su código en Function App, para que el código se quede corriendo allí, se ha planificado que serán unas 50 horas.

* (DP3.1) Crear el Function App y el pipeline del despligue del function app conllevará unas 35 horas.
* (DP3.2) Exportar la infraestructura como a código necesario, serán unas 15 horas.

### • Definir el pipeline final del despligue de los recursos (DP4)

Este parte es muy importante, ya que trata de definir y crear el pipeline que creara todos los recursos unas 20 horas trabajando

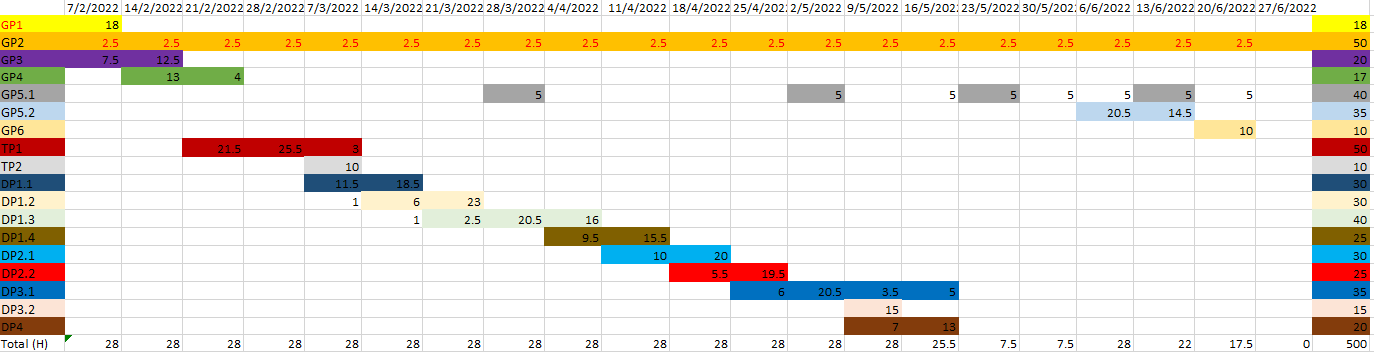


Figure 1: Taula de a planificación inicial del proyecto

* 1. **Planificación Actual**

La desviación respeto el plan inicial ha pasado debido a 4 razones.

* En el plan inicial no estaba planificado la opción de incluir una base de datos, la cual tuvimos que incluir 2 tareas nuevas
  + 1. (DP5.1) Crear la base de datos y el pipeline de despligue serán unas 10 horas.
    2. (DP5.2) Exportar la infraestructura como código necesario, serán unas 10 horas.
* Tuvimos un problema al intentar desplegar el recuso de Function App y por esto tuvimos que cambiar la tecnología de código como infraestructura de arm a bicep , y traducir todo el código de arm a bicep que también nos llevó a crear una tarea nueva (DP6) de 20 horas
* Al final se ha decidido de crear una prueba con código para demostrar como el usuario pueda adaptar su código, la cual se ha llevado a crear 4 tareas más
  + 1. (DP1.5) Crear el contenido para aks 20 horas.
    2. (DP2.3) Escribir el código para la web app 10 horas.
    3. (DP3.3) Escribir el código para la Function App 10 horas.
    4. (DP5.3) Crear la base de datos para el azure sql data base 10 horas.
* El proyecto empezó en 3-1 no en 2-1

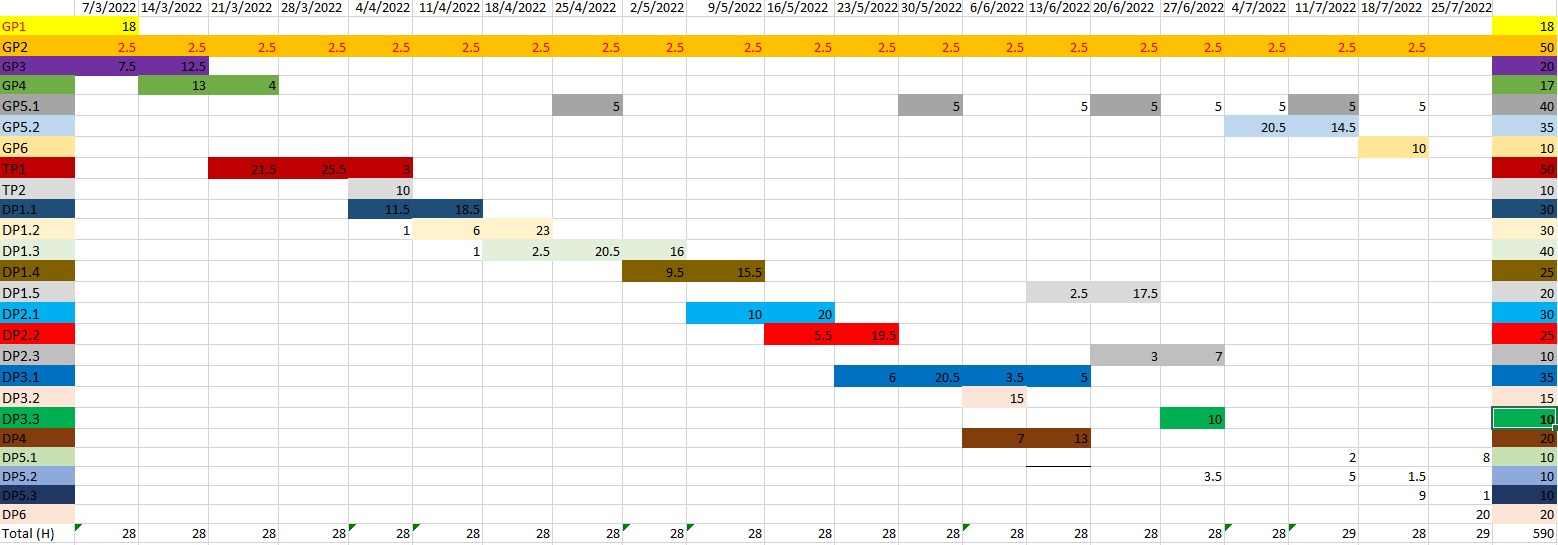


Figure 2: La planificacion real del proyecto

## 5.2. *Recursos*

### *Recursos humanos*

*Aquí vamos a estudiar los roles importantes para seguir desarrollando el proyecto*

* ***Jefe del proyecto (J)***

*Es la persona que toma les decisiones que van a nivel de negocio.*

* ***Director de tecnologías(T)***

*Es el responsable de decidir las tecnologías que se van a usar para el desarrollo del proyecto.*

* ***Desenrrollador (D)***

*Su tarea será seguir las decisiones del jefe del proyecto y el director de tecnologías para continuar con el proceso del desarrollo y también crear la guía de uso.*

## *5.3. Planificación de Riesgos*

Es muy importante poder tener una buena planificación de riesgos para dar paso a los improvistos que pueden pasar mientras se está desarrollando el proyecto. Ahora vamos a planificar los riesgos que pueden suceder y las medidas que podemos tomar.

### • Perder el portátil

Yo dispongo de dos portátiles (uno para el trabajo y otro personal) que son muy similares con las características, entonces si el principal que uso (el portátil del trabajo) le pasa algo puedo pasar al secundario, y todo mi código estará en Azure Repos, para poder recuperar el código en cualquier momento.

El tiempo del retraso para cambiar de un portátil a otro, será cuatro horas ya para poder configurar el otro portátil y establecer mi entrono de trabajo allá en el nuevo portátil.

### • El disco duro falla

Si el disco duro falla el final del día antes de subir el material (documentación + código) al Azure Repos, entonces en este caso la faena será para volver a escribir el trabajo perdido y también reemplazar el antiguo disco con uno nuevo.

El tiempo del retraso en este caso será 2 horas para volver a escribir el trabajo perdido y 3 horas para reemplazar el disco antiguo y también para comprar el nuevo disco y instalarlo.

### • Se cambia algo de los códigos de infraestructura de Azure o algo de las librerías que uso

Lo único que puedo hacer será revisar mi código y volver a adaptarlo a los nuevos cambios.

El tiempo del retraso en este caso será depende de lo que ha cambiado, pero será de 1 hora a 4 horas.

### • Mi suscripción gratuita de estudiante de Azure se termina

Cambiar mi suscripción gratuita a una normal de pago.

El tiempo del retraso en este caso será una hora como máximo para introducir los datos de la tarjeta del pago y también el tiempo que tarda Azure hasta que se renovar la suscripción.

### Base tecnológica

En este apartado vamos a estudiar las tecnologías importantes para poder seguir profundizando en el trabajo.

### Nube

La nube se considera uno de los conceptos más importantes hoy en dia en el mundo informático.

La nube es no tener el código de tus aplicaciones corriendo en tus servidores locales, sino que tú tienes un servicio alquilado con alguna empresa para alquiler la capacidad de computación y tener tus aplicaciones corriendo allá.

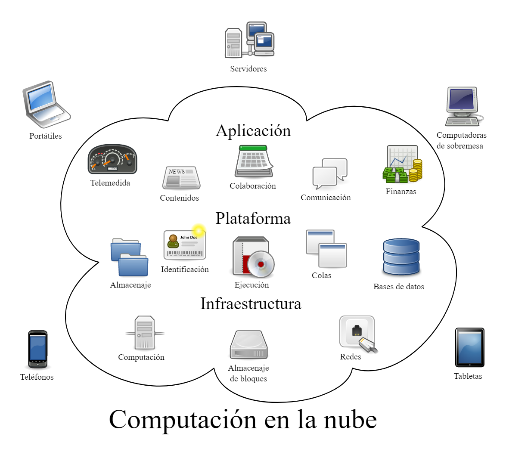


Figure 3: Presentación gráfica de la definición de la nube

### Prometheus

Prometheus es un software de monitorización y alertas, que es capaz de coleccionar las métricas de los diferentes servicios y recursos y las guarda en una base de datos en formato de series temporales



Figure 4: el logo de Prometheus

### Grafana

Grafana es una hermanita que consiste de una interfaz gráfica, que sirve para mostrar los datos de monitorización de forma gráfica, de manera muy intuitiva para que los usuarios pueden detectar los fallos en caso de problemas.



Figure 5: Logo de Grafana

### Azure

Es el servicio de nube que ofrece la compaña Microsoft, y se considera uno de los servicios de nube más grandes del mundo, Azure contiene el 20 porciento del mercado global de la computación en la nube.



Figure 6: Logo de la nube de Azure

### Function App

Es un servicio que ofrece Azure que permite a los desarrolladores ejecutar sus códigos sin tener que preocupar de nada, Azure gestiona todo lo necesario para que el código se ejecuta, y también Azure ofrece herramientas al usuario para que el usuario pueda determinar cuándo se tiene que ejecutar el codigo.



Figure 7: Logo de Function App

### Azure Web App

Es una hermanita de Azure que permita crear aplicaciones webs de forma muy rápida y sencilla, en que solo se tiene que proporcionar el código del cliente de la aplicación web, y Azure gestiono el resto.



Figure 8: Logo de Azure Web App

### Kubernetes (K8s)

Es una tecnología que permite crear microservicios y orquestarlos de forma que estos microservicios sean siempre disponibles y funcionales de la manera más optima posible.



Figure 9: Logo de Kubernetes

### Azure Kubernetes Service (AKS)

Es el servicio de Kuberntes que ofrece la nube de Azure, Azure tiene dos modalidades para usar este servicio, la primera Azure te deja la infraestructura fisica de Kubernetes ya creada y tú tienes que gestionarla y desplegar tus microservicios allá, o la segunda modalidad que tu solamente tienes que desplegar tus microservicios, y Azure se encarga del resto. 

Figure 10: Logo de Azure Kubernetes Service (AKS)

**App Service plan:**

Este recurso básicamente presenta la infraestructura física que se utiliza para correr el Function App y la Web App y el usuario normalmente puede decidir las características físicas de la infraestructura (de cuanto memoria y cpu dispone el recuso).



Figure 11: El logo deL App Service Plan

**Bicep**

Bicep es un lenguaje declarativo (que se declara lo que se quiere conseguir, no como se va a conseguir), que se usa para crear recursos en Azure, y de este forma si pasa cualquier problema con la infraestructura y se quiere volver a levantar toda la infraestructura de cero, se hace con solamente volver a ejecutar el código de bicep.

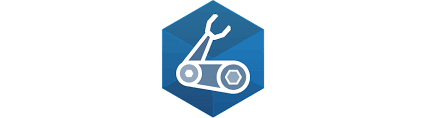


Figure 12: El logo de Bicep

### Azure Repos

Son los repositorios donde se guarden los códigos de las aplicaciones, para que después se usen en el despliegue.



Figure 13: Logo de Azure repos

**Pipelines**

Es una metodología para definir el ciclo de vida de las aplicaciones, de forma que todos los procesos que definen el ciclo de vida de una aplicación (compilar, test, desplegar, etc) se hacen de forma automática mediante código.

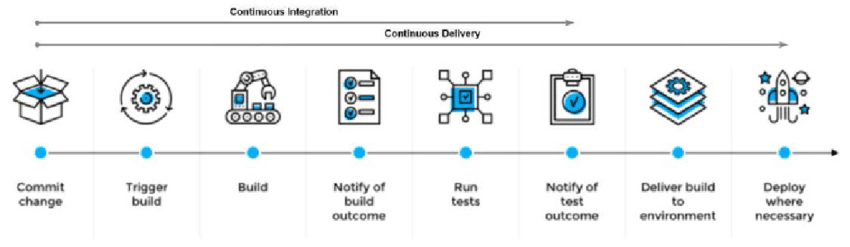


Figure 14: La arquitectura que normalmente forma un pipeline

### Azure Pipelines

Es la hermanita de pipeline que proporciona Azure.



Figure 15: Logo de Azure pipelines

**Azure Sql servers**

Son servidores sql que los proporciona Azure, que dentro de estos servidores se puede gestionar más de una base de datos, que todos van a están en la misma localización geográfica, solamente el usuario tiene que elegir la capacidad computacional del servidor y la localización.



Figure 16: Logo de Azure Sql Server

**Azure Sql databases**

Es el servicio de azure que permite a los usarios gestionar sus bases de datos sql dentro de un servidor sql de azure



Figure 17: Logo de Azure Sql Database

**Log Analytics Workspace**

Es un servicio de Azure, que se utiliza para guardar los logs de los diferentes recursos de Azure en diferentes tablas, y también permite consultar los logs usando un lenguaje que se llama Kusto, que es un lenguaje muy intuitiva, que parce bastante al lenguaje de Sql.

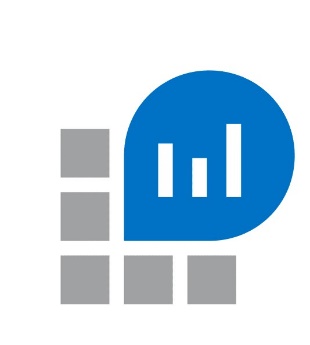


Figure 18: El logo del Log Analytics Workspace

**Application Insights**

Es un servicio de logs que se puede conecta con las aplicaciones, y con el recurso de Function App permite ver los logs en directo sin la necesidad de entrar al Log Analytics



Figure 19:El log de Application Insights

**Azure Container Registery**

Es un recurso de Azure que básicamente se usa para guardar imagines de contenedores, que se puede guardar más de una imagen y cada imagen se puede guardar con más de una versión, todos dentro del mismo recuso



Figure 20: Logo de Azure Container Registery

**Ngnix-Ingress**

Es un controlador que esta creado con Ngnix que tiene reglas de entrutamientos, para enrutar desde fuera del cluster a las diferentes apliaciones que están curriendo dentro del cluster de kubernetes

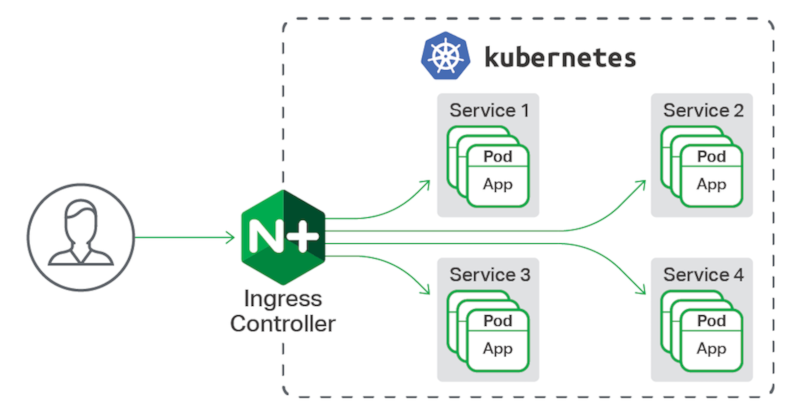


Figure 21: Presentacion grafica de como funciona el controlador de ngnix-ingress

**Helm**

Es una hermanita que nos ayuda a gestionar nuestros aplicaciones de Kubernetes, Helm se usa para administrar y también guardar y compartir las aplicaciones de Kubernetes, de una manera que nos ayuda a reutilizar estas aplicaciones varias veces.

Helm también tiene su propio lenguaje para crear plantillas de las aplicaciones de Kubernetes, de tal manera que nos ayuda a desplegar las mismas aplicaciones en diferentes entornos sin tener errores.



Figure 22:El logo de Helm

### Trabajo Hecho

En este apartado vamos a detallar el trabajo hecho por este proyecto, que básicamente se devide en dos partes, lo que se ha hecho en azure, y lo que se ha hecho a nivel de pipelines

**Azure**

Aquí vamos a discutir todo lo que se ha hecho con la nube de Azure

**Bicep**

En este trabajo usamos bicep para definir toda la infraestructura de los diferentes recusos.

Principalmente tenemos un fichero principal que se llama main.bicep que es el fichero que se va a ejecutar y después tenemos otros ficheros dentro de la carpeta bicep-templates, que allá están definidos las plantillas de todos los recursos que vamos a usar, y de este forma si un recuso lo vamos a usar más de una vez, entonces no hace falta definirlo más de una vez, solamente con llamarlo las veces que hace falta con pasar los parámetros ya lo tendremos

Los nombres de los ficheros como los recusos siguen las abreviaturas que definen Microsoft para azure :

App Service Plan: plan

Function app: func

Web app: app

Log Analytics Workspace: log

Application Insights: appi

Azure SQL database: sqldb

Azure SQL Database server: sql

Azure Kubernetes Service: aks

Azure Container Registry: acr

Storage Account: st

**Function App**

El Function App en este proyecto sirve como un entorno de ejecución para que el usuario pone sus códigos que quiere que se ejecuten en Python versión 3.9, el usuario solamente pone su código y también las librarías y las versiones que use, y Azure se encarga del resto.

La manera que el usuario tiene para poder lanzar su código dentro del Function App, es usando el URL del Function, y el código se ejecutará y responderá con el resultado esperado

El mismo Function App lo podemos usar para poner más de un código y que todos se ejecuten de forma separada (cada uno se ejecuta con su URL).

Para que el usuario pueda monetizar sus códigos, el Function App se le crea previamente un Application Insights y este Application Insights se conecta con un Log Analytics para que el usuario puede ver todos los logs (excepciones, errores, warning, etc) y las métricas(la memoria disponible, los procesos del CPU, las requestas, etc), en dos modos, en modo directo y también se puede ver las dadas de fechas anteriores que están guardadas.

Se necesita un storage account para poder crear el Azure App, previamente necesitamos crear un Storage Account, como un lugar para guardar los códigos que se tiene que ejecutar.

El Function app tiene que estar creado encima de un service app plan, que se considera como el servidor donde se corre el código y el usuario puede elegir las características físicas del este servidor

**Azure Kubernetes Service:**

Todas las aplicaciones que vienen con el proyecto están empaquetados usando Helm, para qué pueden tener la máxima portabilidad posible.

Las aplicaciones que vienen predefinidos con el aks son los siguientes:

* Ingress-Ngnix: Esta aplicación tiene todos los componentes necesarios, para hacer funcionar el controlado Ingress de Nginx, porque las aplicaciones en el clúster cuando quieren comunica con el exterior lo hacen utilizando este controlado, de esta manera ahorramos el dinero, ya que, si no para cada aplicación que se quiere comunicar con el exterior tendremos que crear un servicio de tipo loadbalancer que en el aks, este tipo de servicio es un servicio de pago, porque Azure te crea un loadbalancer por cada servicio de este tipo
* Grafana: Esta aplicación tiene todos los componentes de Grafana y Prometheus para que Prometheus coge las métricas desde el clúster de Kubernetes y también tiene una aplicación que se llama azure-exporter creada dentro de la aplicación de Grafana, esta aplicación lo que hace es usar unos permisos que se asignan al clúster de Kubernetes para que pueda leer los datos de monitorización en Azure y exporta las métricas de Azure en un puerto que posteriormente el Prometheus lea las métricas de Azure de este puerto, y exporta estas métricas a Grafana para que los usuarios pueden ver las métricas en la interfaz de Grafana. Para Prometheus y Grafana hay una regla en el controlador de Ingress para que los usuarios externos pueden entrar a la interfaz gráfica de Grafana como Prometheus
* La última aplicación es la aplicación creada para que el usuario pueda usarla para poner su proyecto aquí, y también tiene una regla en el controlador de Ingress para que los usuarios externos también pueden conectar a la aplicación

El aks dentro de un pipeline se le asigna los permisos necesarios para que el azure-exporte puede leer las métricas desde Azure.

El aks se le asigna un log analytics y así el usuario no le hace falta conectarse dentro del clúster para qué pueda ver los estados y los logs de los diferentes proyectos dentro del clúster, también desde el portal de Azure, Azure crear gráficas y reportes muy útiles y explicativos para todos los componentes del aks.

Con el aks también se crea un acr y se conecta con el aks, esto es para que después de crear la imagen del proyecto del cliente y la imagen del proyecto del azure-exporter, estas dos imágenes se suben al acr, porque posteriormente el clúster las puede usar para levantar los proyectos

**Azure SQL database**

El Sql Database es el servicio de Azure que utilizamos para manejar la base de datos del cliente

La base de datos se le asigna un log analytics para guardar los logs y las métricas de la base de datos

Con este servicio básicamente creamos dos recursos más:

* Un servidor de sql, básicamente es para mantener la base de datos (es la infraestructura donde corre la base de datos)
* Creamos una regla en el firewall del servidor de Sql para permitir que los servicios de Azure pueden interactuar con el servidor (creamos este regla básicamente para permitir que los pipelines pueden trabajar con el servidor)

**Web App**

La funcionalidad del web app es muy similar al Function App, la única diferencia es que uno es solamente es una función y la otra es una aplicación web

En la aplicación web el usuario tiene que poner todos los ficheros necesarios para la aplicación web y también las dependencias que usa la aplicación web y también sus versiones.

El web app también se conecta a un log analytics para guardar los logs y las métricas como los otros recursos, también como el Function App, tiene el modo de ver los logs en directo activado, pero esta vez sin la necesidad de un Application Insights

Para correr el web app solamente hace falta un Service App Plan, no hace falta un Storage Account

**Log analytics Worksapce**

Los cuatro recursos que hemos mencionado anteriormente usen Log Analytics Worksapce, pero los cuatro usen el mismo Log Analytics y esto es para ahorrar en los gastos

**Estructura de Pipelines**

Principalmente el sistema de los pipelines se divide en 5 pipelines. Un pipeline principal para crear los recursos y desplegar los contenidos y cuatro pipelines de contenido para los cuatro recursos (Aks, Azure Function, Azure web app, Azure sql), para sí hace falta modificar el contenido de un de los recursos, entonces de esta manera no hará falta volver a desplegar toda la infraestructura de cero.

**Principal-pipeline**

Este es el pipeline principal, este pipeline es el pipeline responsable de crear todos los recursos, i también desplegar todos los contenidos de cada recurso, este pipeline llama a todos los stages que están definidos dentro del proyecto, el fichero que contiene este pipeline es el fichero principal-pipeline.yml, y tiene un trigger programado para lanzarse si se cambia un de los ficheros principal-pipeline.yml, main.bicep o algún fichero dentro del el directorio bicep-templates.

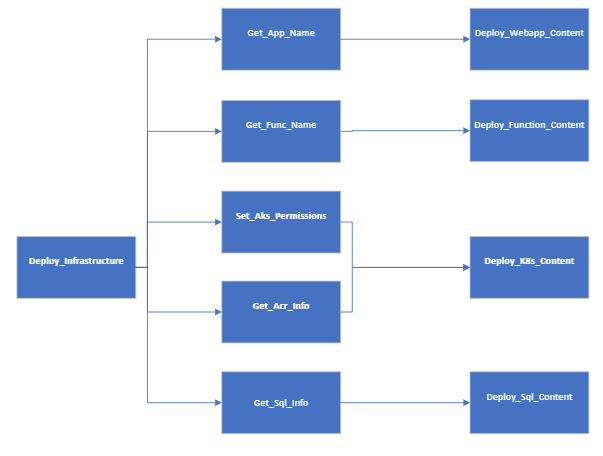


Figure 23: La estructura de los stages del Principal-pipeline

**Aks-pipeline**

Este pipeline es el pipeline responsable de desplegar los contenidos dentro del aks, este pipeline contiene el stage Set\_Aks\_Permissions, Get\_Acr\_Info y Deploy\_K8s\_Content y su trigger se lanza en función si se cambia el fichero aks-pipeline.yml o algún fichero del directorio content/aks-content/.

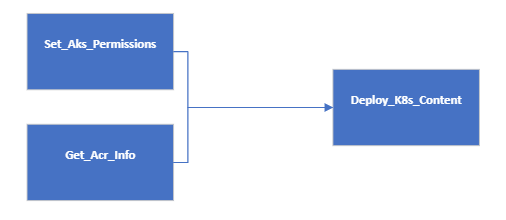


Figure 24: La estructura de los stages del Aks-pipeline

**Azure-function-pipeline**

La funcionalidad de este pipeline es desplegar el contenido dentro del azure function y este pipeline solamente contiene dos stages que se llamen Deploy\_Function\_Content y Get\_Func\_Name, y el trigger de este pipeline se lanza si se cambia el fichero function-pipeline.yml o algún fichero dentro del directorio content/function-content/.

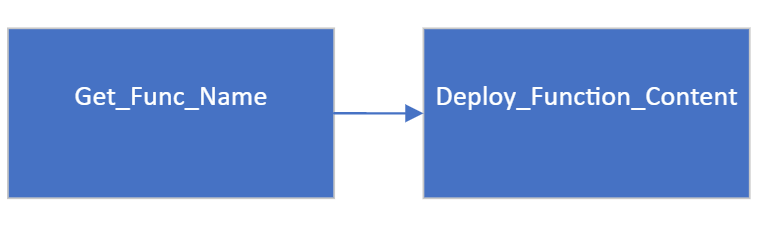


Figure 25: La estructura de los stages del Azure-function-pipeline

**Azure-webapp-pipeline**

La funcionalidad de este pipeline es el desligué del contenido del azure webapp, este pipeline contiene dos stages que son Get\_App\_Name y Deploy\_Webapp\_Content, el pipeline está definido dentro del fichero webapp-pipeline.yml y su trigger se lanza si se cambia el contenido del fichero webapp-pipeline.yml o los ficheros dentro del directorio content/webapp-content/.

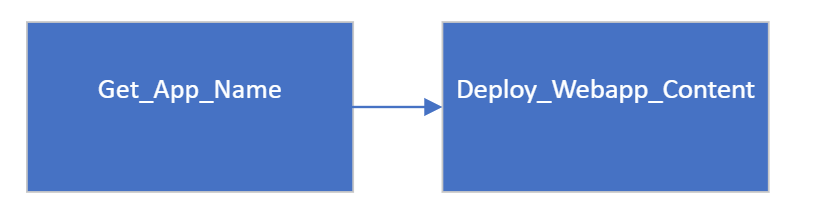


Figure 26: La estructura de los stages del Azure-webapp-pipeline

**Azure-sql-pipeline**

Finalmente tenemos este pipeline que es básicamente un pipeline para desplegar la base de datos que como los otros pipelines de contenidos tiene dos stages que se llamen Deploy\_Sql\_Content y Get\_Sql\_Info, el pipeline está definido dentro del fichero sql-stage.yml y se lanza su trigger si se cambia el fichero sql-pipeline.yml o los ficheros del directorio content/sql-content/.

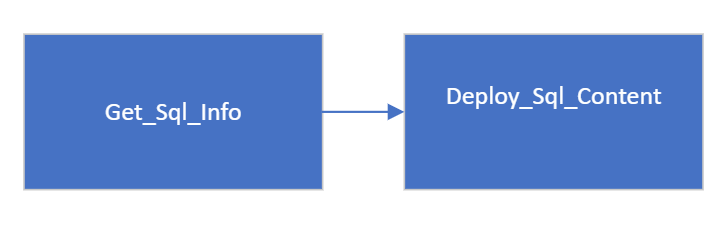


Figure 27::La estructura de los stages del Azure-sql-pipeline

**Estructura de Stages**

Los stages son la unidad de trabajo más grande que se puede definir en los pipelines de azure, también dentro de los stages podemos definir más de una tarea que se pueden compilar en diferentes entornos.

En nuestro proyecto los stages están definidos en ficheros independientes de los ficheros que están definidos los pipelines y de esta manera ganamos la aventaja de que en caso de que más de un pipeline necesiten el mismo stage, entonces no nos hará falta definir el mismo stage más de una vez, sino solamente con llamar el stage desde los diferentes pipelines ya lo tendremos.

A continuación vamos a detallar en cada uno de los stages en nuestro proyecto.

Get\_Acr\_Info

Este stage se usa en los pipelines principal-pipeline y Aks-pipeline, se llama justamente antes de llamar el stage Deploy\_K8s\_Content y solamente tiene una dependencia al stage Deploy\_infrastructure cuando se lanza desde el pipeline principal-pipeline.

La funcionalidad de este stage es traer el nombre y la url del acr, porque posteriormente el stage Deploy\_K8s\_Content, puede usar esta información para subir las imágenes al acr y también conectarse el aks con el acr.

Este stage consigue su tarea partiendo del sistema de tags que se define previamente para todos los recursos en el momento del despliegue de este proyecto, en los ficheros de bicep.

Set\_Aks\_Permissions

Este stage se llama solamente desde el pipeline principal-pipeline, se llama antes del stage Deploy\_K8s\_Content y solamente tiene una dependencia al stage Deploy\_infrastructure

La funcionalidad de este stage es dar los permisos necesarios al aks para que el exportador de métricas de azure pueda tener los permisos suficientes para poder acceder a las métricas.

Los permisos que se dan en azure es a nivel de clúster, y el role que se asigna, es un role para leer las métricas desde azure, en este stage hay una variable que se llama useAksMonitoring que tiene que estar a true para que se ejecute este stage que si no, este stage se salta.

Deploy\_K8s\_Content

Este stage se llama desde los pipelines principal-pipeline y Aks-pipeline, lo que hace este stage es primero conectarse al acr y subir la imagen del proyecto del cliente y otra imagen para el exportador de métricas que utiliza el prometheus.

Finalmente despliega cinco proyectos usando helm que son el controlador de ingress- nginx, la aplicación del cliente, grafana y prometheus, el exportador de métricas de azure para prometheus.

Este stage tiene dos dependencia al stage de Set\_Aks\_Permissions y Get\_Acr\_Info.

Get Func Name

Este stage se usa en los pipelines Principal-pipeline y Function-pipeline y tiene una dependencia al stage Deploy\_Infrastructure cuando se lanza desde el pipeline Principal-pipeline

La funcionalidad de este stage, es conseguir el nombre del azure function, ya que este nombre se elige de forma aleatoria, y el nombre es importante para el stage Deploy\_Function\_Content, para poder subir el contenido dentro del azure function.

El stage localitza el recuso del azure function, usando el sistema de tags aplicados a todos los recusos dentro de este proyecto.

Deploy\_Function\_Content

En este stage hay más de una tarea definida que básicamente lo que hace el stage es conectarse al azure function y preparar todo el entorno y instalar los paquetes necesarios que define el usuario dentro del fichero requierments.txt y finalmente comprime todos los ficheros y los sube dentro del azure function.

Este stage tiene una única dependencia al stage Get\_Func\_Name y se lanza en los pipelines Principal-pipeline y Function-pipeline.

Deploy\_Infrastructure

Este es el stage principal y no hay ni una stage que se puede ejecutarse sin que previamente este stage se halla ejecutado y que no haya dado errores, todas las otras stages tienen dependencia a este stage aun que no sea dependencia directa.

Este stage es responsable de desplegar todos los recursos usando el fichero main.bicep y las plantillas dentro de la carpeta bicep-templates que también están definidos en bicep.

Este stage solamente corra dentro del pipeline principal-pipeline.

Get\_App\_Name

En este stage basicamente se consigue el nombre del recurso del webapp, para que el stage Deploy\_Webapp\_Content, pueda localizar el webapp y pueda subir el código.

Este stage localitza el recuso del webapp, usando el sistema de tags aplicados a todos los recusos que se depliguen dentro de este proyecto.

Este stage se usa dentro de dos pipelines principal-pipeline y webapp-pipeline y tiene una dependencia al stage Deploy\_Infrastructure cuando se lanza dentro del pipeline principal-pipeline.

Deploy\_Webapp\_Content

Aquí básicamente se prepara el entorno de la webapp y se hi instalar todos los paquestes necesarios, y finalmente se comprime el conjunto de fichero y se sube todo a la webapp.

Este stage se llama desde los pipelines Webapp-pipeline y principal-pipeline, la dependencia que tiene este stage es dependencia directa al stage Get\_App\_Name.

Get\_Sql\_Info

La funcionalidad de este stage es conseguir el nombre y el url del servidor sql, para que después el Deploy\_Sql\_Content, pueda subir la base de datos dentro del servicio de sql.

Este stage tiene una dependencia directa al stage Deploy\_Infrastructure cuando se executa dentro del pipeline principal-pipeline y se llama desde los pipelines sql-pipeline y principal-pipeline

Deploy\_Sql\_Content

Este stage es responsable de subir el contenido en la sql service.

Lo que hace este stage básicamente es, abrir el cortafuego para poder conectarse a la base de datos y después sube el contenido de la base de datos y finalmente vuelve a cerrar el cortafuego.

Este stage tiene una dependencia directa al stage Get\_Sql\_Info y se llama desde los pipelines sql-pipeline y principal-pipeline

**Azure Repository Structure**

En el repositorio de azure nosotros tenemos todos los ficheros para nuestro proyecto, básicamente allá hay todo, y nosotros hemos decidido de tirar por esta vía por dos razones principales:

1. Para proteger nuestros ficheros y datos para no perderlos.
2. Ya que usamos el cloud de azure, nos resultara mucho más fácil usar azure repositorio que otro repositorio, ya que también usamos azure pipelines.

En el directorio root de nuestro repositorio lo que encontramos son tres carpetas y seis ficheros, que a continuación vamos a explicar de que consta cada uno

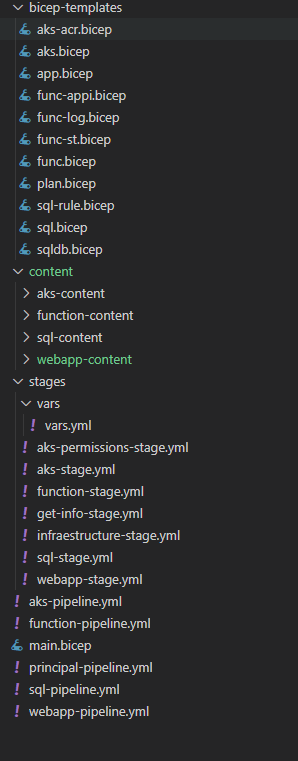


Figure 28: La estructura de ficheros dentro del proyecto

**Los seis ficheros son**

Primero, los cinco ficheros que definen los cinco pipelines que necesitamos para el diseño de nuestro sistema de integración continua (principal-pipeline.yml, aks-pipeline.yml, function-pipeline.yml, webapp-pipeline.yml, sql-pipeline.yml) y main.bicep que es el fichero principal que llama a las diferentes plantillas para desplegar la infraestructura de recursos dentro de azure.

**Los tres directories son los siguientes**:

bicep-template

En este directorio hay 11 ficheros, que contienen los diferentes plantillas de los diferentes recursos usados para desplegar la infraestructura, las plantillas están definidas en bicep, y todas se llaman desde el main.bicep.

  A continuación tenemos la lista de los ficheros y el contenido de cada fichero

* aks-acr.bicep: Este fichero contiene la plantilla que se usa para crear el registro de contenedores que usar el aks para obtener las imágenes para los pods.
* aks.bicep: Aquí está definida la planilla del aks para desplegarse el cluster.
* app.bicep: Aquí están el despliegue del webapp.
* func-appi.bicep: Este fichero se usa para el despliegue del application insights que se usa con el azure function.
* func-log.bicep: Este fichero se usa para desplegar el log analytics que usa el azure function.
* func-st.bicep: Aquí se crea el storage account que se usa para que el azure function puede guardar sus datos.
* func.bicep: Este es la plantilla que se usa para el azure function.
* plan.bicep: Este fichero se usa para crear dos app service plan uno para el azure function y el otro para el webapp.
* sql-rule.bicep: Este fichero es para crear los accesos en el firewall del servidor sql que por defecto tiene todos los accesos negados.
* sql.bicep: Aquí se crea la base de datos dentro del servidor sql.
* sqldb.bicep: Esta plantilla se usa para crear el servidor sql que contiene la base de datos.

content

Este directorio contiene 4 subdirectorios aks-content, function-content, sql-content, webapp-content, que cada uno de estos cuatro subdirectorios contiene el contenido que se usa en los recursos (azure kubernetes service, azure function, azure webapp, azure sql).

aks-content: Aquí hay 6 subdirectorios, cada uno es para un componente diferente dentro del aks.

* App: Este subdirectorio se usa para desplegar la aplicación del cliente dentro del aks.
* docker-images: Este subdirectorio contiene 2 imágenes de docker una para la aplicación del cliente y otra para el exportador de métricas que usar prometheus para traer las métricas de azure.
* ingress-ngnix: aquí está definido el contenido que se usa para desplegar el controlador de ingress-ngnix.
* kube-prometheus-stack: Aquí están las definiciones de los diferentes componentes que se usen para prometheus y grafana para correr dentro del cluster.
* prometheus-exporter: Este es el subdirectorio que define el despliegue del exportador de métricas que usa prometheus dentro de azure.

Pipelines

Dentro de este directorio están definidas todas las stages que se usen y también los variables que se usen en todo el proyecto

* vars: Es un subdirectorio que contiene un fichero que se llama vars.yml que allá están todas las variables de toda el sistema.
* aks-permissions-stage.yml: Aquí está el stage que actúa para dar al clúster de k8s los permisos necesarios para el exportador de métricas de azure.
* aks-stage.yml: Es un fichero donde está definido el stage responsable de subir el contenido de aks en el mismo.
* function-stage.yml: Este stage es el responsable de subir la función definida del usuario dentro del azure function.
* infraestructure-stage.yml: Aquí está definida el stage que desplegá el código de bicep para desplegar toda la infraestructura.
* sql-stage.yml: En este fichero está definida la stage responsable de subir la base de datos en el sql service de azure.
* webapp-stage.yml: Este stage es el responsable de subir el código de la aplicación web del cliente en el webapp.
* Get-info-stage: Este fichero contiene cuatro diferentes stages, que los cuatro están definidos con el mismo código y en el mismo fichero, los cuatro pipelines definidos en este fichero, son Get\_Acr\_Info, Get\_Func\_Name, Get\_App\_Name y Get\_Sql\_Info

### Ejemplo

El ejemplo creado para este proyecto es un ejemplo muy sencillo que usa todos los recursos creados en este proyecto (App Service Plan, Function app, Web app, Log Analytics Workspace, Application Insights, Azure SQL Database, Azure SQL Database Server, Azure Kubernetes Service, Azure Container Registry, Storage Account) y también enseña como conectar entre ellos.

El ejemplo consiste de un pequeño calculador online que calcular el cuadrado y la raíz cuadrada de un número y en función de si se tiene que calcular el cuadrado o la raíz cuadrada, se llama al Azure Function o al Azure Kubernetes Service.

La parte frontal de esta aplicación esta subida al Web App y consiste del código del la web app hecho con Python usando Flask y las tres plantillas que se usen (la plantilla principal, la plantilla del cuadrado y la plantilla de la raíz cuadrada)

El funcionamiento de la web app es el siguiente el usuario entra a la plantilla principal y allá hay dos botones uno es el botón de calcular el cuadrado y el otro es el botón de la raíz cuadrada, en función si el usuario clica a un botón o a otro se mandara a una plantilla o a otra.

Si el usuario clica al botón de calcular el cuadrado, el usuario se mandara a la plantilla de exponential.html, que consiste de un formulario de un campo de input para que el usuario escribe el número que quiere conseguir su cuadrado y un botón, cuando el usuario clica al botón el número se mandara al Azure Kubernetes Servicey y el Azure Kubernetes Service lo que hará es buscar si este número está en la tabla de exponentiation y en caso de que sí. Lo devolverá al usuario y en caso de que no esta, el Azure Kubernetes Service se lo volverá a calcular y se lo mandara para guardar a la tabla de base de datos de exponentiation, y también se lo volverá al web app para mostrarlo al usuario.

El mismo proceso pasa con la raíz cuadrada, pero en lugar de mandar el número al Azure Kubernetes Service, se mandara al Function App, y la plantilla que se llamara esta vez es la squad.html

Para la web app las plantillas están dentro de la carpeta /content/webapp-content/templates y el código de la aplicación web que está hecho con Python estará en el /content/webapp-content/app.py y finalmente las dependencias que usa el web app está dentro del /content/webapp-content/requirements.txt

Para la base de datos Sql, la base de datos está definida dentro del fichero content\sql- content\database.sql

Para el Function App, el código está dentro del fichero content\function-content\HttpTrigger1\\_\_init\_\_.py y las dependencias están dentro de content\function- content\requirements.txt

Para el aks la definición de la imagen de Docker que usa la Aplicación está dentro del fichero content\aks-content\Dockerfile, la Aplicación que corre dentro de la imagen de Docker está dentro del fichero content\aks-content\app\app.py, y las dependencias que usa esta imagen está en el fichero content\aks-content\app\requirements.txt

### Guía de uso

aquí nos vamos a detallar en los pasos que tenemos que seguir para poder poner el proyecto en funcionamiento.

1. Entramos a este url https://dev.azure.com/ y nos logeamos
2. Después tenemos que crear un proyecto dentro de nuestro azure devops:
   1. Tenemos que entrar a nuestra organitzacion y clicar a New project

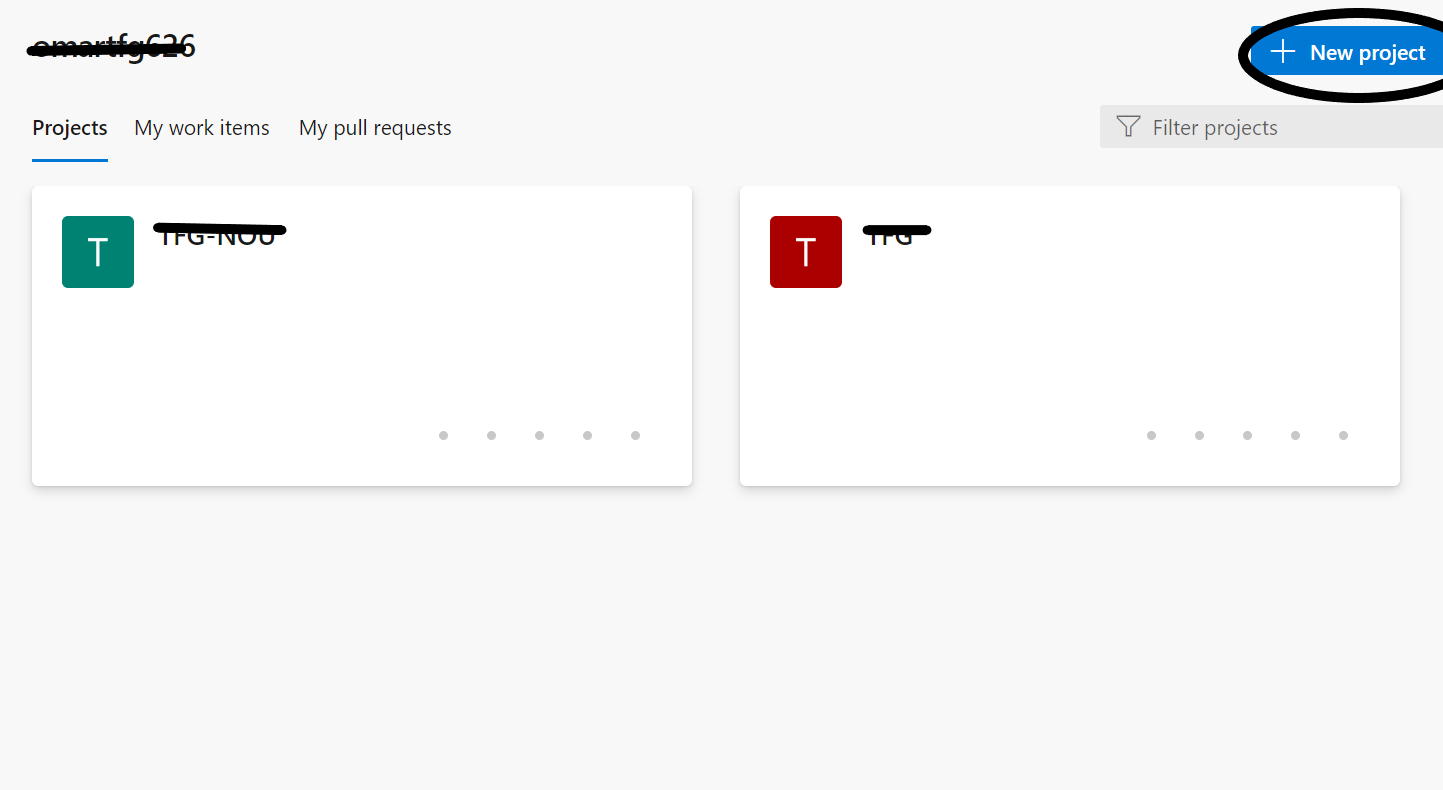


Figure 29: El portal de organización dentro de Azure Devops

* 1. Y después entrar los datos de nuestro proyecto y finalmente le damos a create

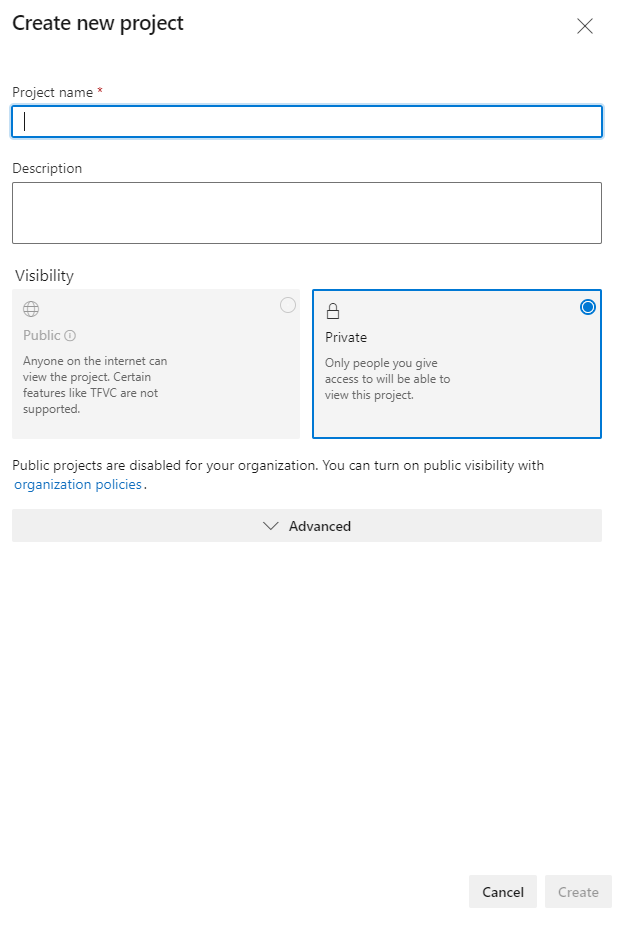


Figure 30: La plantilla para crear un proyecto dentro de Azure Devops

1. Subir el proyecto a Azure Repos
   1. Primero tenemos que entrar al proyecto
   2. Y clicar a Azure Repos y copiamos el código de git (hace falta que tengamos aceso a un terminal y también tener git instalado en nuestra computadora)

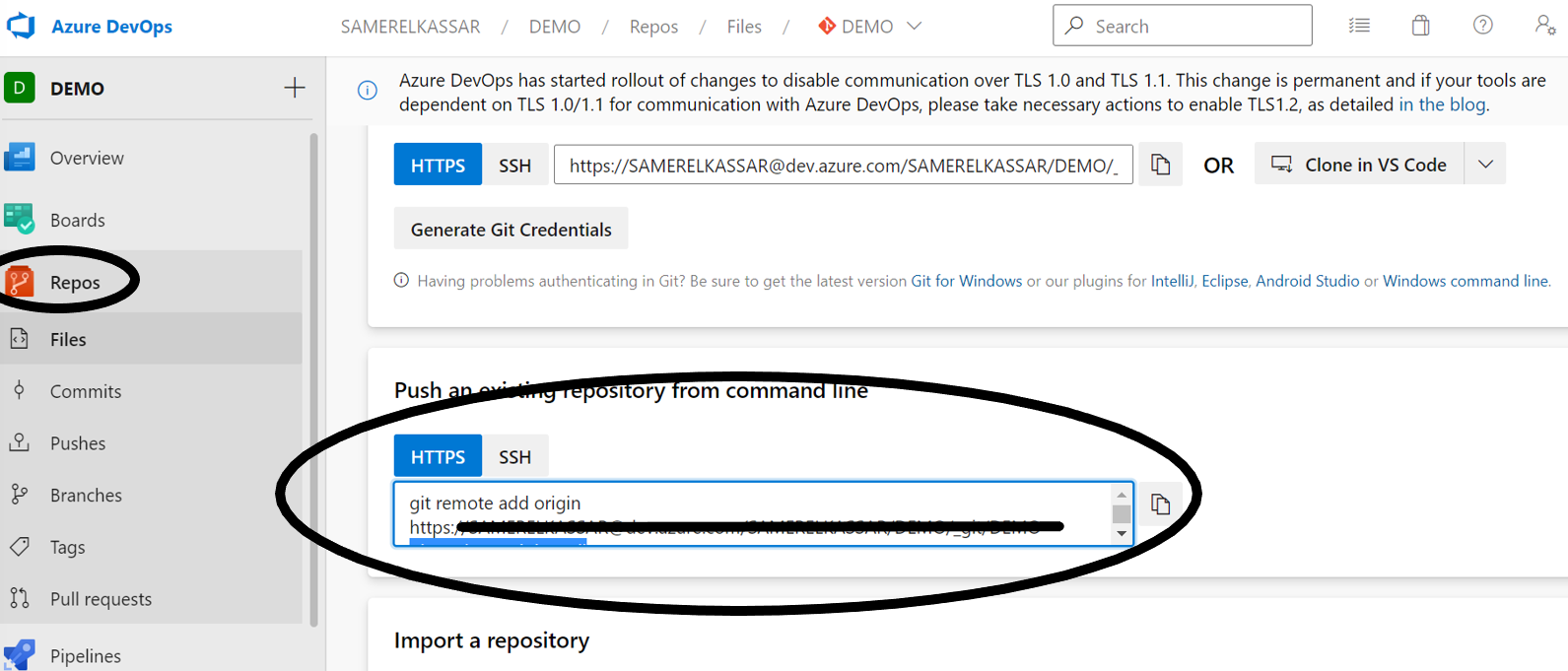


Figure 31: La plantilla de Azure Repos dentro de un proyecto en Azure Devops

* 1. Y finalmente nos saldrá una pestaña y tendremos que autenticarnos y el código se subirá solo

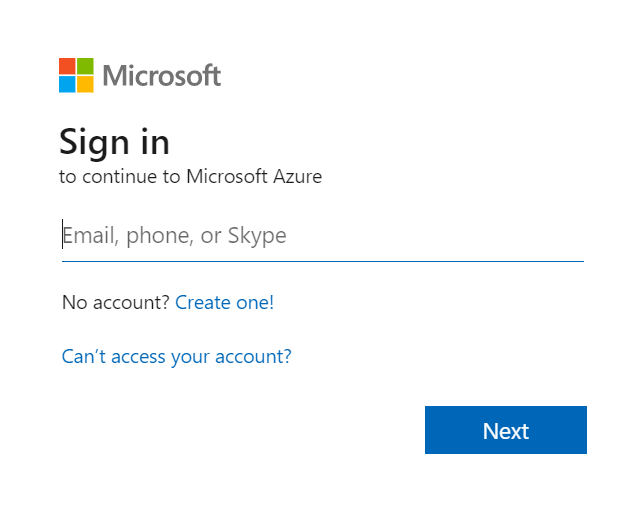


Figure 32: La pestaña para autenticarse con el objectivo de subir un codigo en Azure Repos

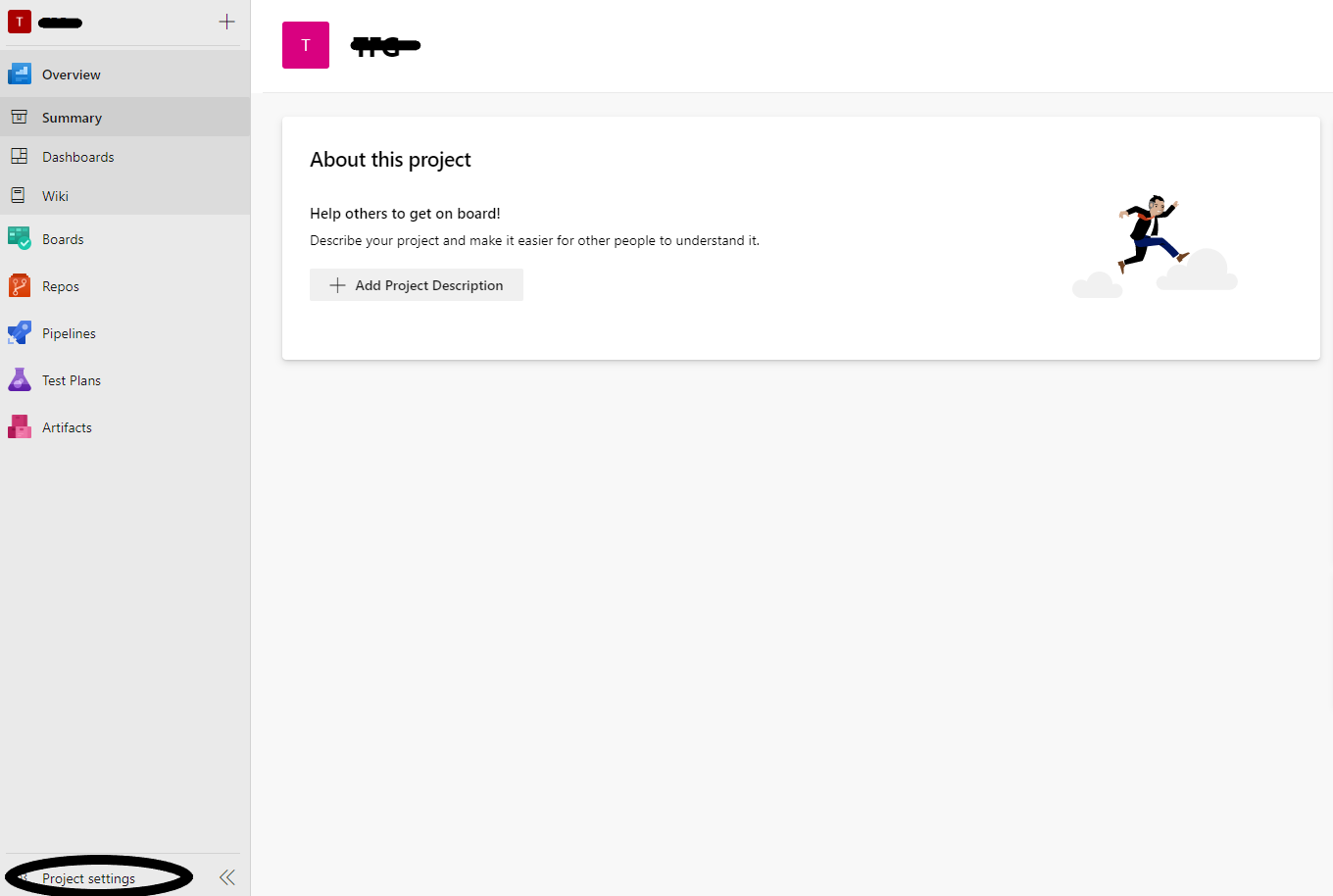
1. Ahora que ya tenemos el código tenemos que crear un service connection, para que azure devops tenga los permisos suficientes para trabajar con azure
   1. Primero tenemos que entrar en el menu de los settings del proyecto

Figure 33: El portal de proyecto dentro de Azure Devops

* 1. Y después ir a service connections y finalmente clicar a Create service connection

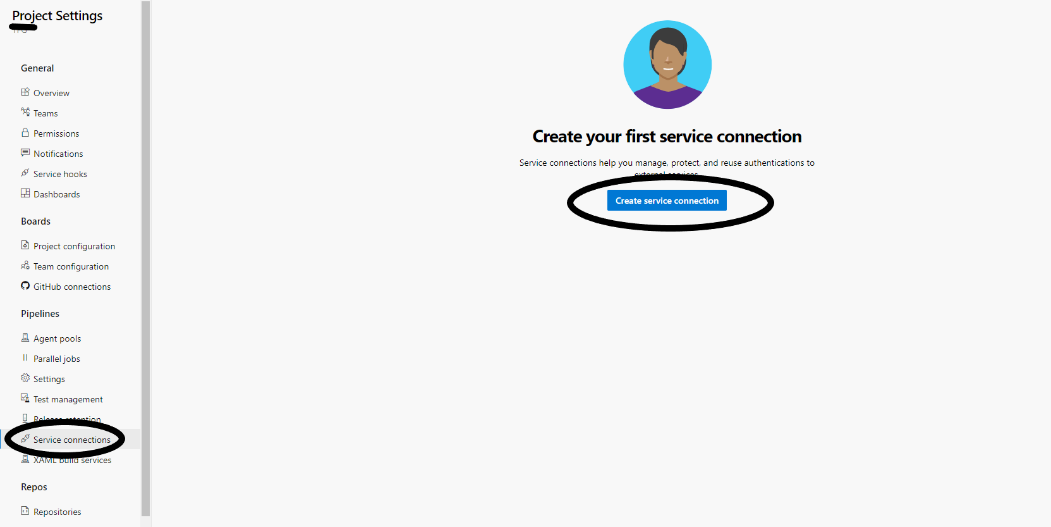


Figure 34: La plantilla para crear un Service Connection dentro de un proyecto

* 1. y de la lista que nos sale de los tipos del service connection tenemos que elegir Azure Resource Manager y después clicar next

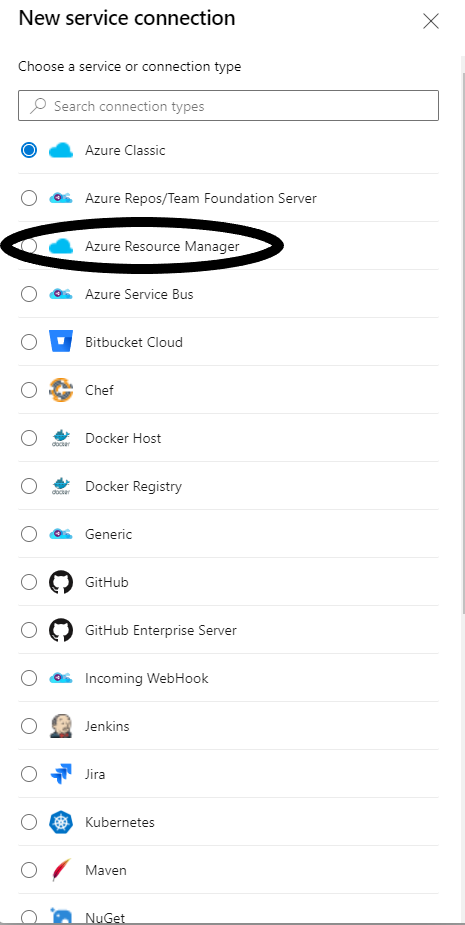


Figure 35: Los tipos de los Service Connection dentro de Azure Devops

* 1. y después elegir Service Principal (automatic) de la authentication method

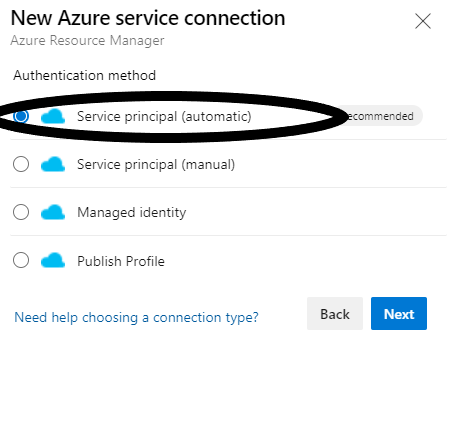


Figure 36: Los tipos de los Azure Service Connection dentro de Azure Devops

* 1. y en los caractaristiques del service connection le damos el nombre de service\_connection y le ponemos al scope de suscripción y finalmente elegimos la suscripción y clicamos a save

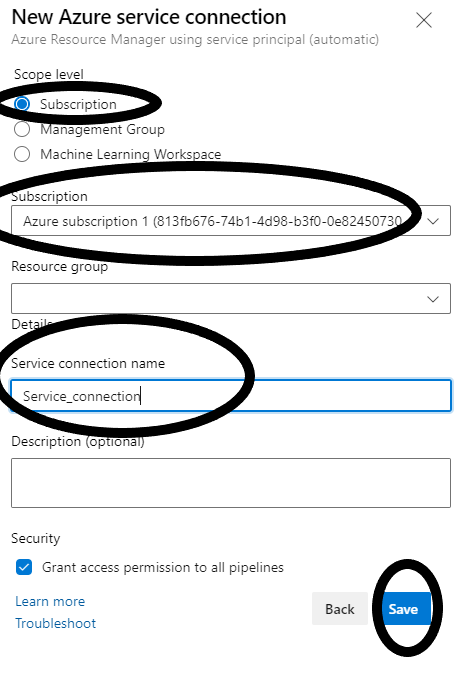


Figure 37: La pestaña para crear un Azure Service Principal

* 1. Ahora entramos en el portal de Azure y tenemos que entrar en subscriptions.

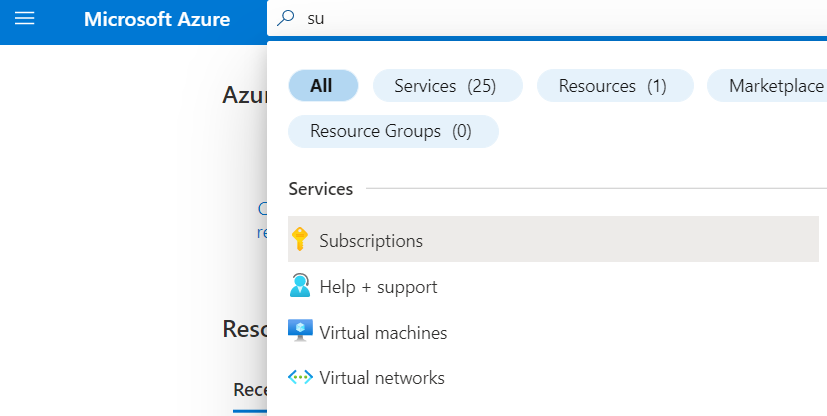


Figure 38: El buscador del Portal de Azure

* 1. Y ahora tenemos que apuntar el id de la suscripcion que usamos en algún lado

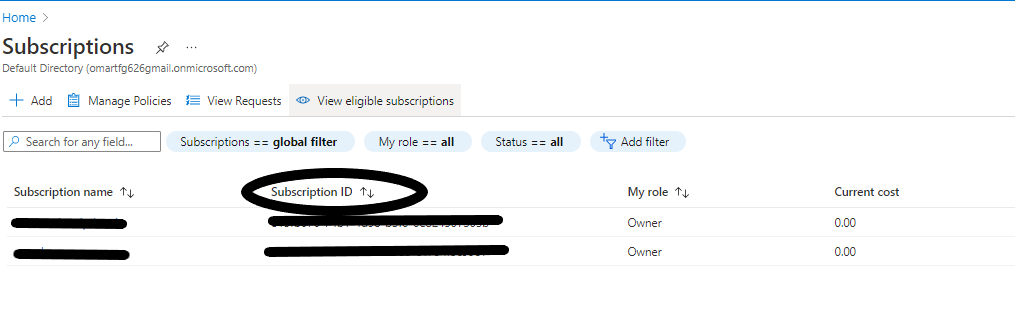


Figure 39: La plantilla de suscripcions dentro de Azure

* 1. Y ahora tenemos que abrir el cloud Shell clicando en la icona del cloud shell

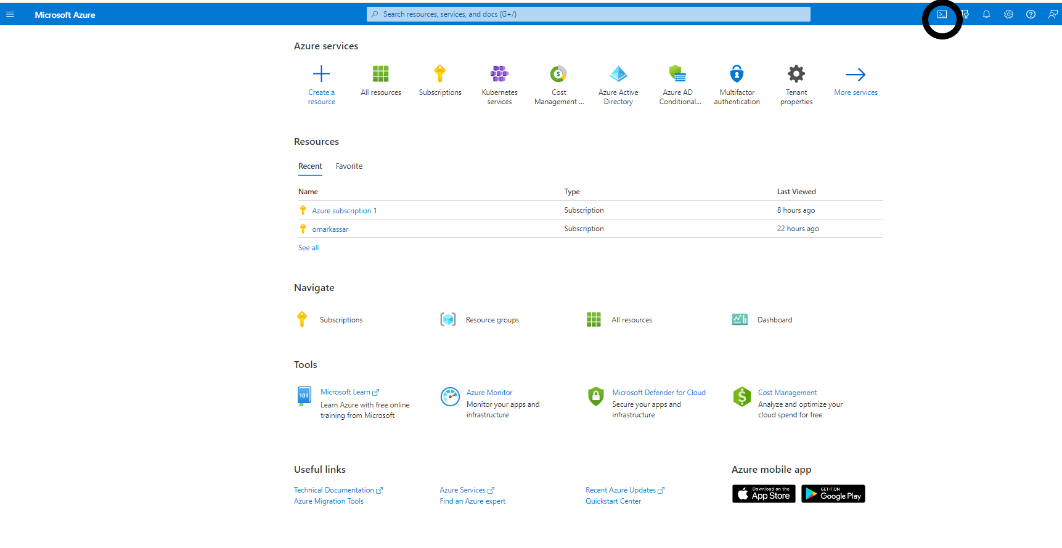


Figure 40: La icona del Cloud Shell

* 1. Si es la primera vez que usamos el cloud Shell entonces tendremos que crear un storage account

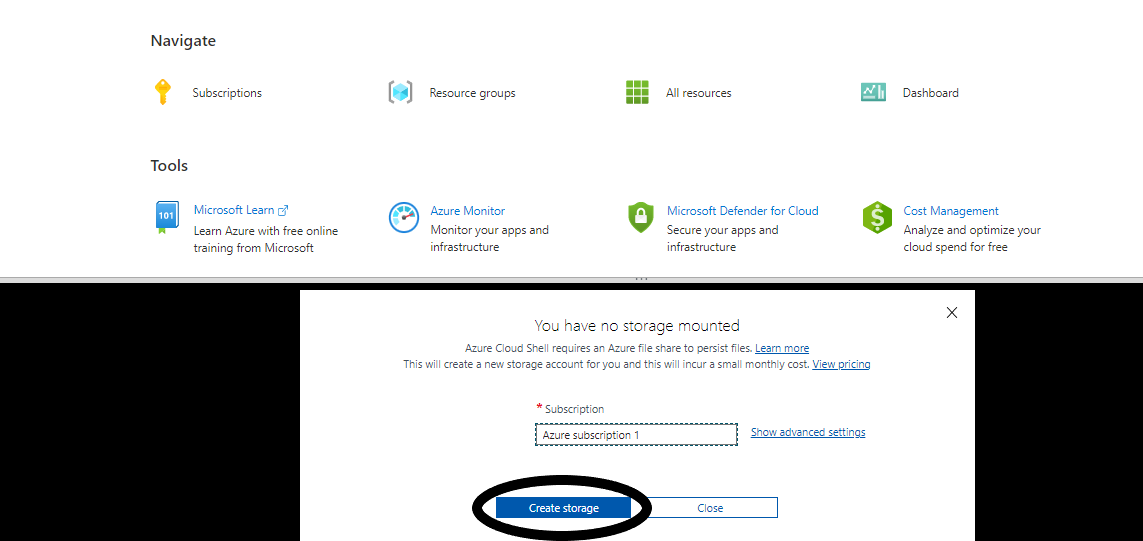


Figure 41: La pestaña del Cloud Shell de Azure

* 1. En el cloud Shell tenemos que executar az ad app list y de la lista que sale tenemos que elegir el objecto que tenga el campo de createdDateTime igual a la fecha en que hemos creado el Service Connection en el azure devops y también que tenga un displayName simialr a: ‘el nombre de usuario que creó el service connection’ + ‘-‘ + ‘el nombre del proyecto en el azure devops’+’un string random’, tenemos que copiar el displayName de este objecto para usarlo luego

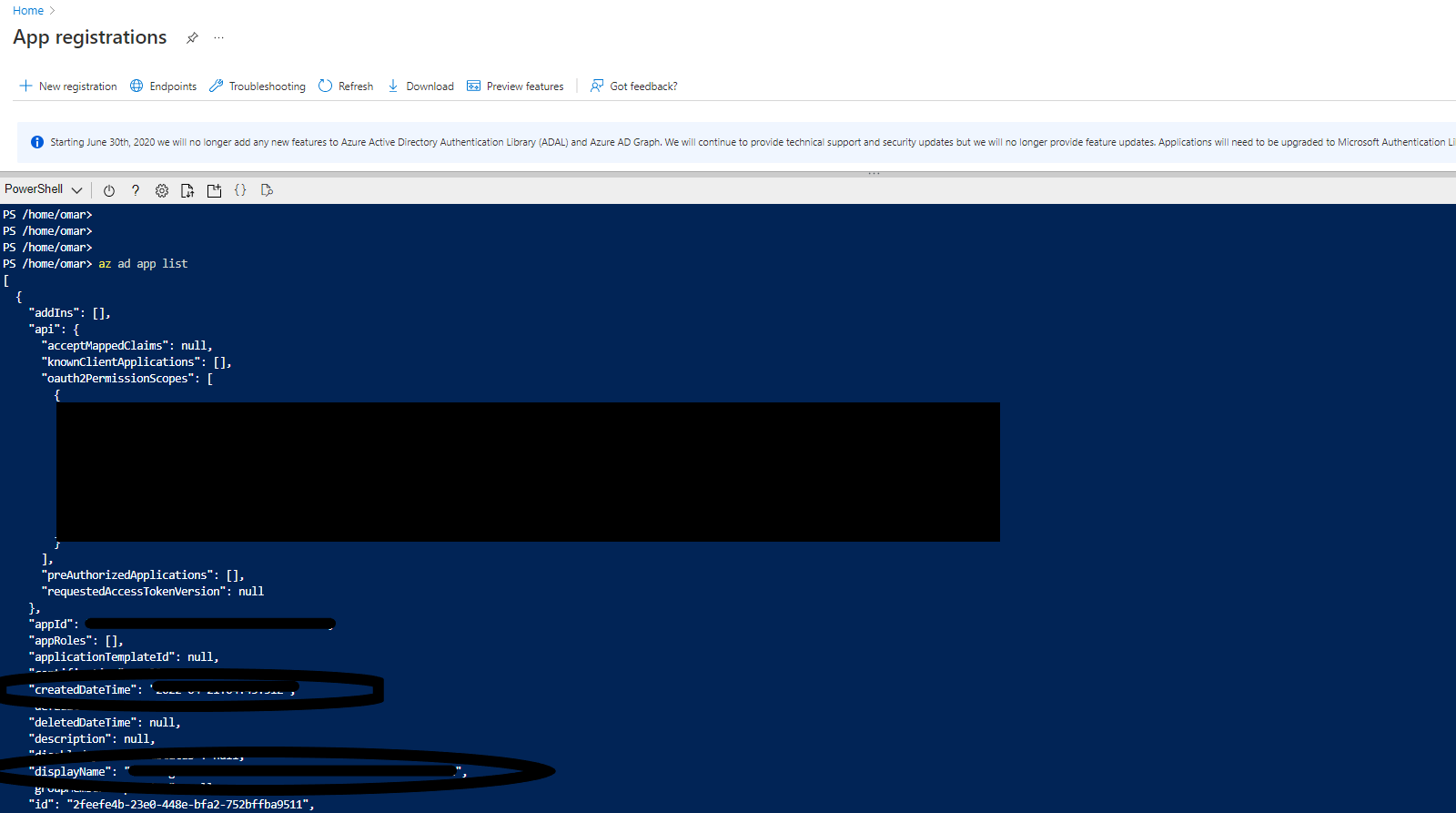


Figure 42: La salida del comando az ad app list

* 1. Tenemos que executar este comando en el cloud Shell:

az ad sp create-for-rbac --name ‘el display name que copiamos en el paso anterior’ --role owner --scopes /subscriptions/’el id de la suscripción que copiamos anteriormente’ (este paso lo hacemos para dar a los pipelines el permiso de owner)

1. Asegurar que todos los providoors que usamos estén registrados dentro de azure
   1. Tenemos que ir a las suscripcions y entrar a la suscripción que usamos

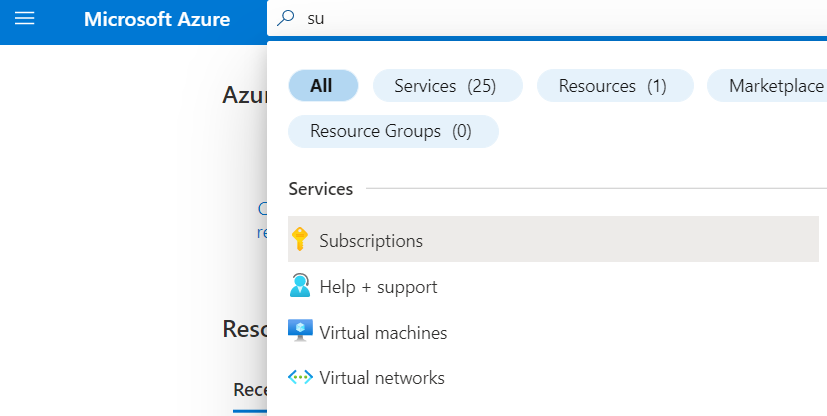


Figure 43: El buscador del portal de Azure

* 1. Dentro de la suscripción tenemos que ir a Resource Providers y asegurar que los providoors Microsoft.OperationsManagemen y Microsoft.ContainerRegistry están registrados (en el status tiene que salir Registered), sino tenemos que clicar al providoor y después clicamos a Register

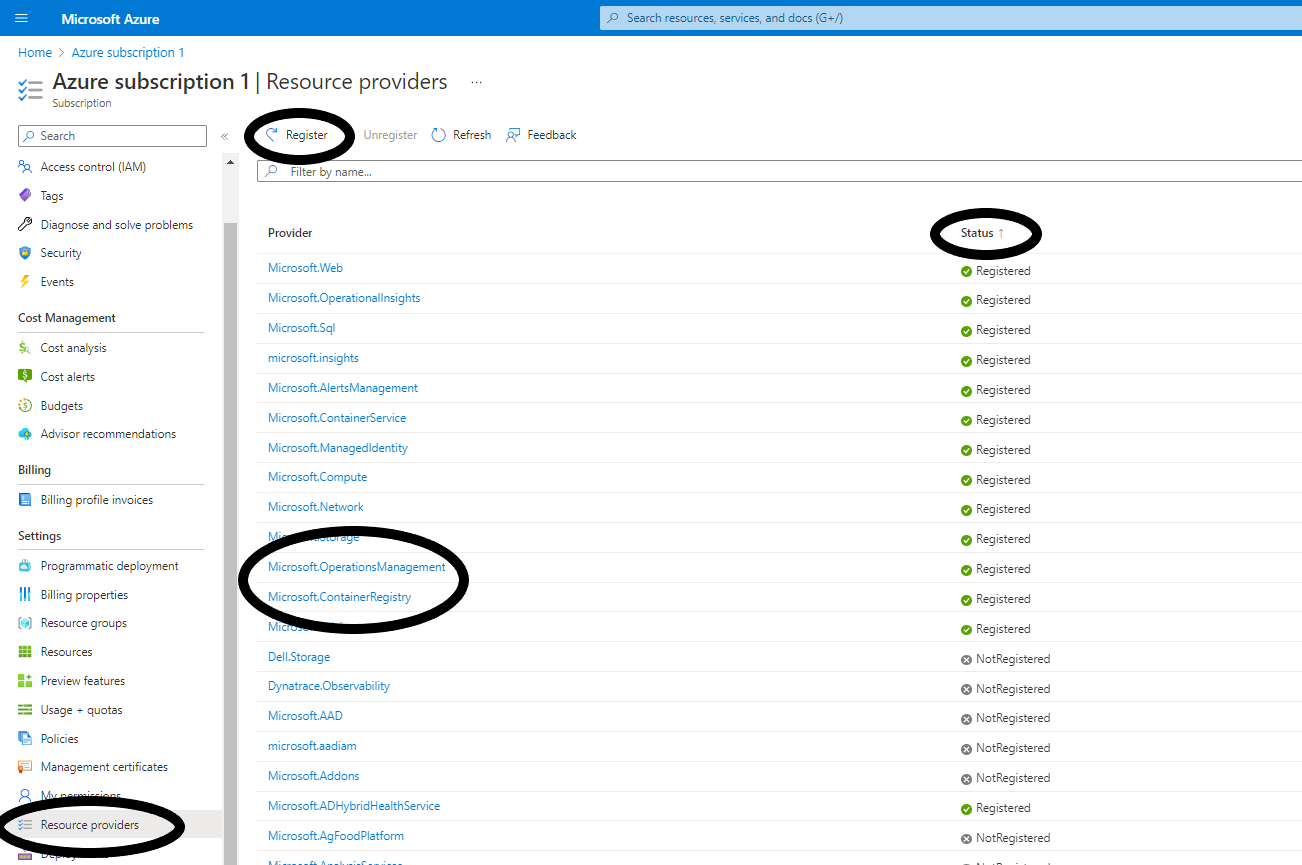


Figure 44: La pestaña de los Resource Providers en Azure

1. Finalmente crear los pipelines tenemos que volver a entrar al proyecto dentro del azure devops
   1. Dentro del proyecto tenemos que ir a pipelines y después clicar a New pipeline

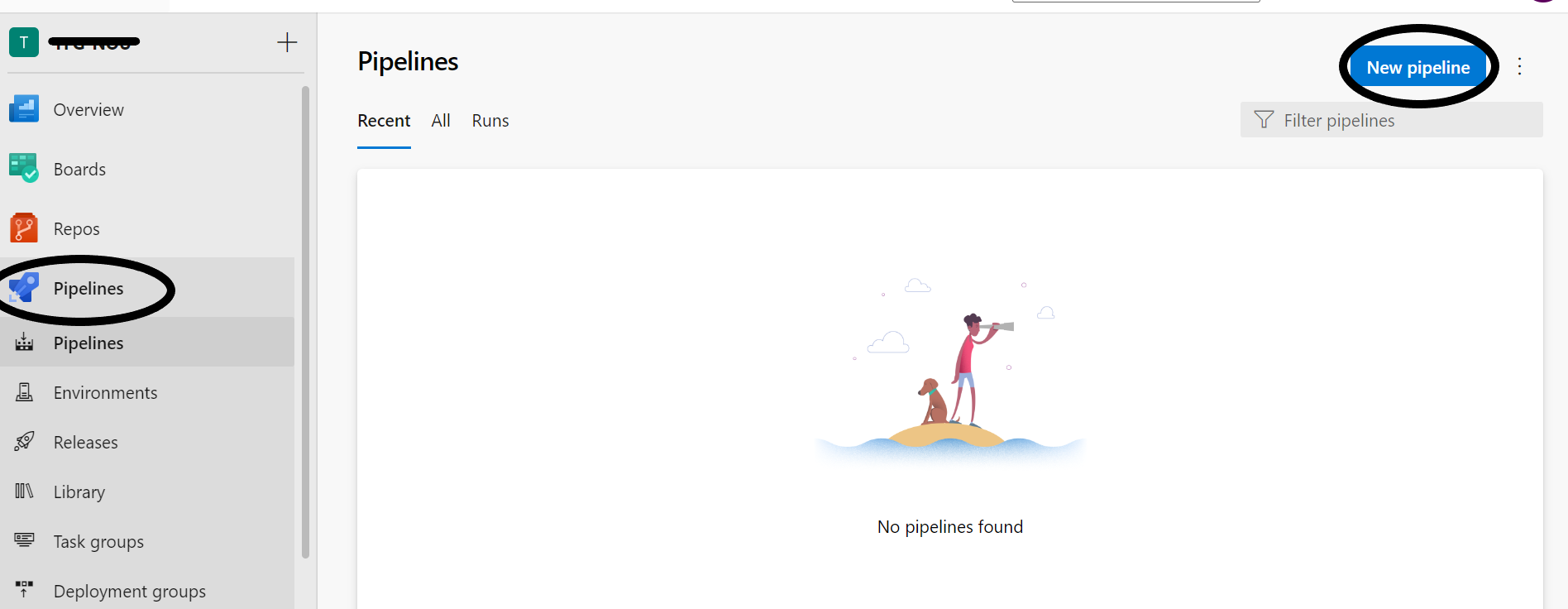


Figure 45: La pestaña de Azure Pipelines dentro de un proyecto en Azure Devops

* 1. Después tenemos que elegir la opción de Azure Repos Git

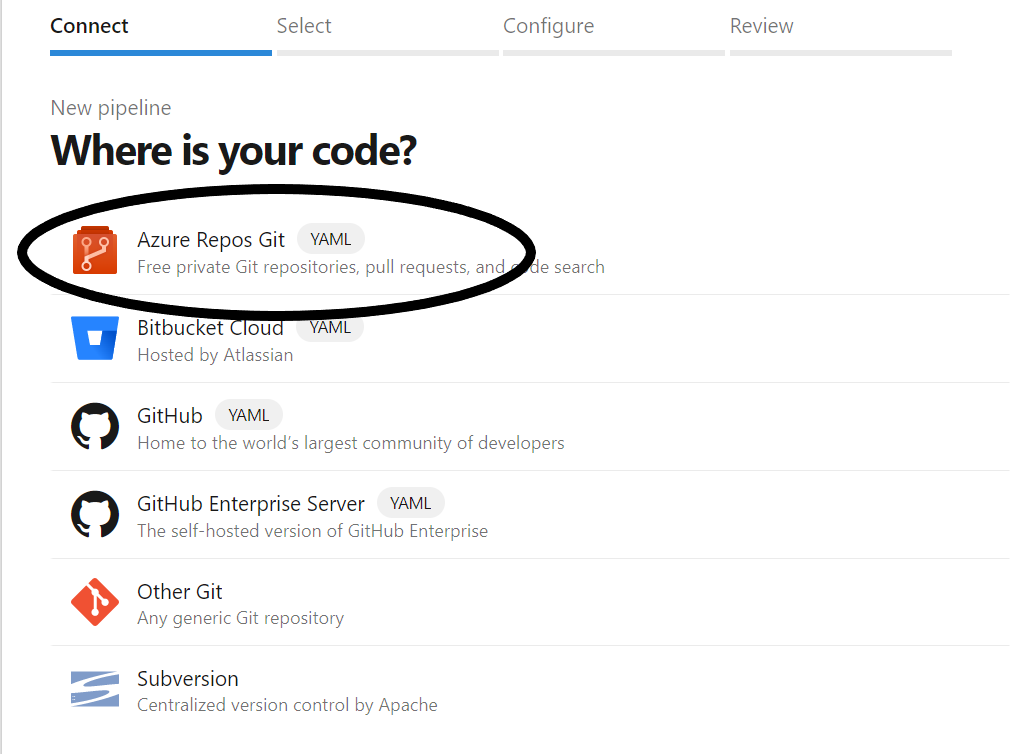


Figure 46: Los recursos desde el cual puedes crear un Azure Pipeline

* 1. Y después elegimos la opción de Existing Azure Pipelines y ponemos el path del pipeline que queremos y finalmente le damos a continue

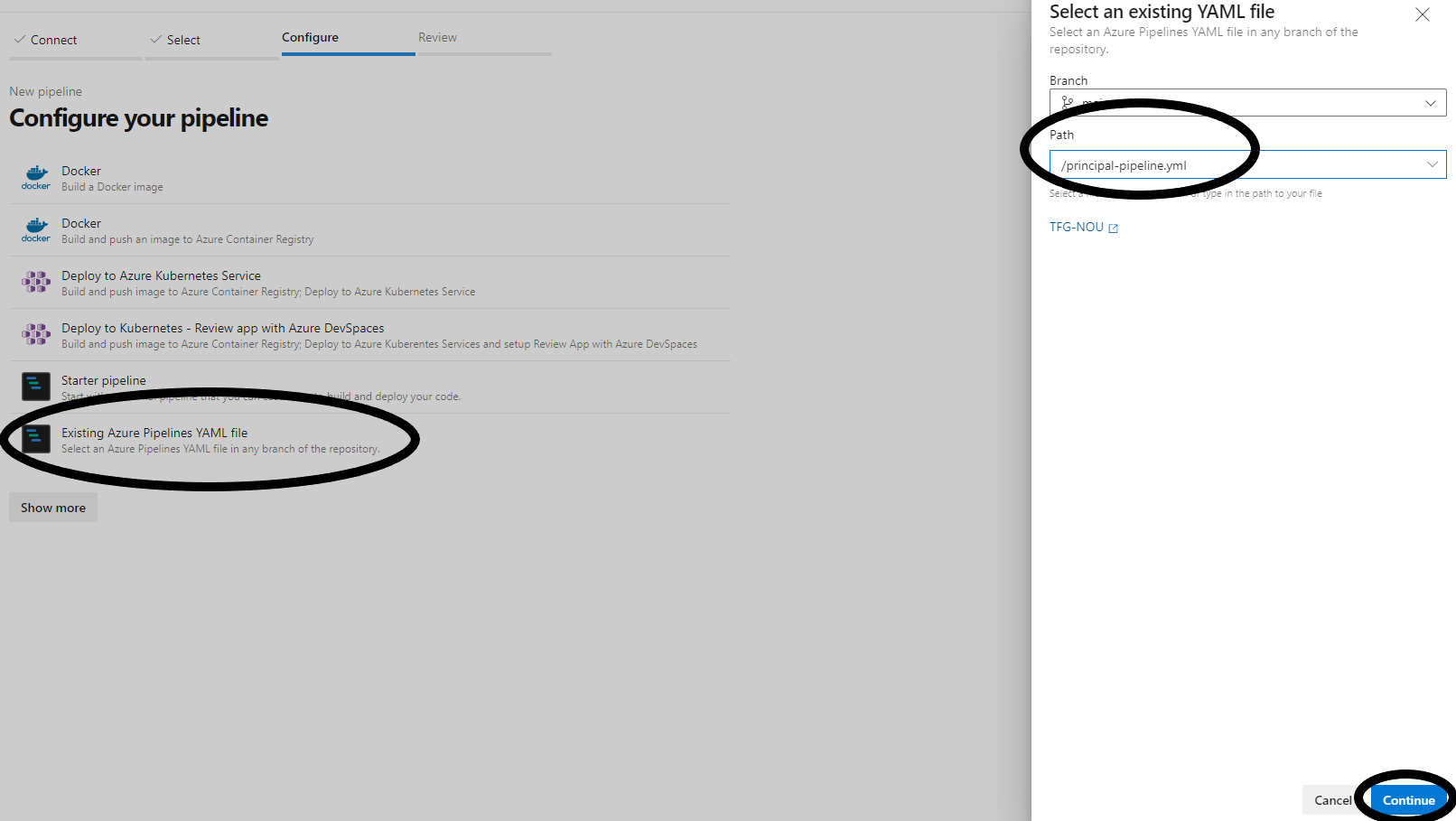


Figure 47: Las opciones que tienes para configurar Azure Pipeline teniendo en cuenta que el recurso que uses es el Azure Repos Git.

* 1. Finalmente le damos a Run

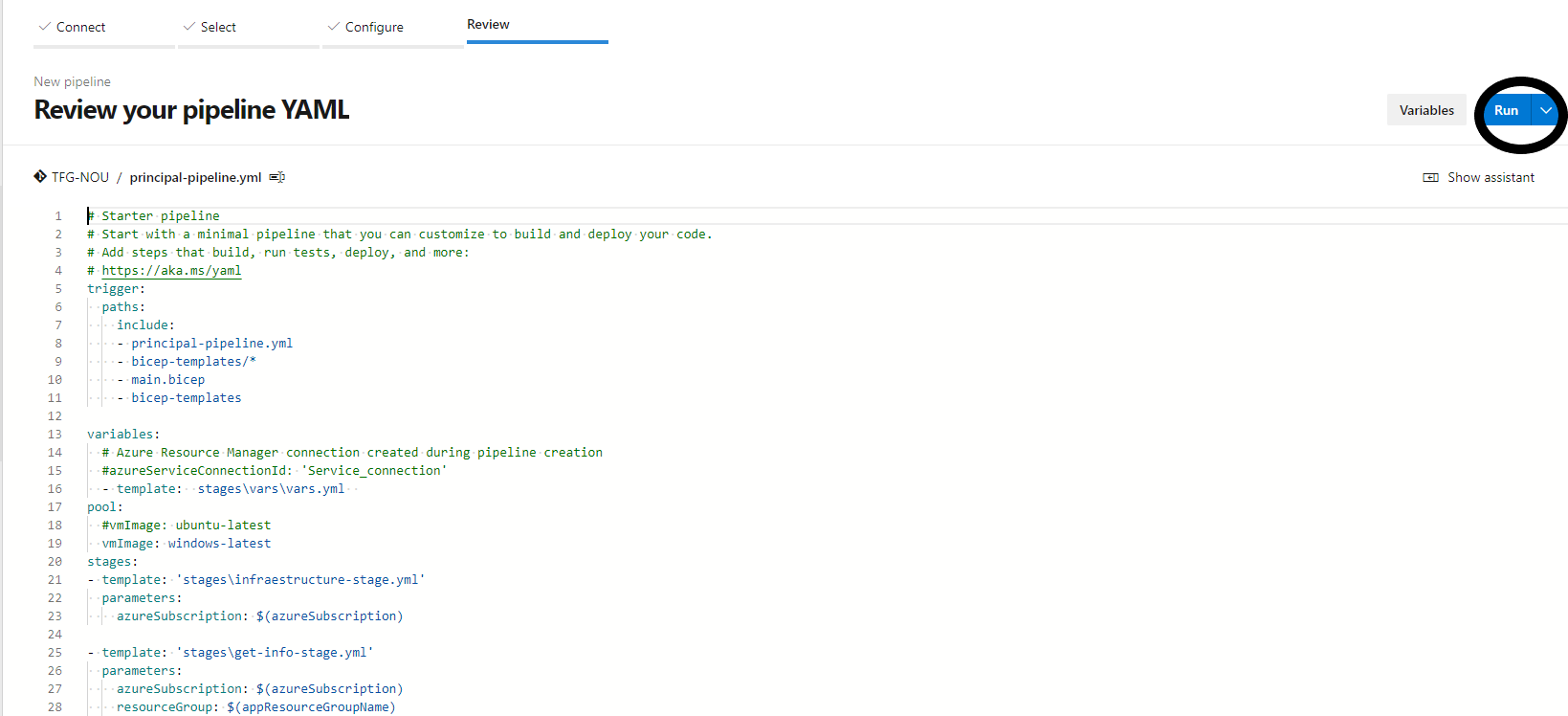


Figure 48: La pestaña que te demuestra los pipelines para comprobarlos antes de mandarlos para ejecutar.

(estos pasos lo tenemos que repetir por cada pipeline cambiando el path en el paso numero c, para el Principal-pipeline el path es /principal-pipeline.yml, para el Aks-pipeline el path es /aks-pipeline.yml, para el Azure-function-pipeline el path es /function-pipeline.yml, para el Azure-webapp-pipeline el path es /webapp-pipeline.yml y finalmente para el Azure-sql-pipeline el path es el /sql-pipeline.yml )

### Sostenibilidad

En este apartado vamos a hablar sobre la sostenibilidad del proyecto de fin de grado, primero vamos a hacer una autoevaluación y después discutir el proyecto en los ámbitos sociales y ambiental.

#### • Autoevaluación

En la carrera de informática llevo 5 años, la verdad durante estos 5 años en la carrera he encontrado muchas maneras para poder ver el lado de la sostenibilidad en la informática.

hay una asignatura optativa que habla sobre el lado social de la

informática también en algunas asignaturas hay algunos proyectos que se hacen al final del curso, que se hacen para subir la nota transversal, pero la verdad yo he echado de menos algo que sea obligatoria para que los alumnos tengan más conciencia por el lado de la sostenibilidad de la informática.

Yo me considero una persona bastante consiente del medioambiente y también de la influencia de la informática sobre nuestra sociedad y pienso que es algo muy importante porque nos puede ayudar mejorarnos como sociedad.

Lo que yo creo que me faltaría, es formaciones de cómo hacer propuestos económicas de proyecto informáticos, este tema nunca he tocado.

#### • Dimensión Ambiental

Para mí el medio ambiente es una de las cosas más importantes en un proyecto, el medio ambiente es la tierra donde vivimos es todo lo que tenemos que cuidar lo máximo posible.

Microsoft es una de las empresas que más que se importen por el medio ambiente, Microsoft han abierto el año pasado un Datacenter en Suecia que la energía que usa es 100% energía renovable.

Microsoft por 2025 tiene un plan que toda la energía que consume la empresas sea energías de recursos renovables.

Esto implica que la energía que se consume desde los Azure Datacenters, sea energía renovable, lo que implicar reducir los polución sobre el medio ambiente ,cuando la gente usan la nube de Azure.



Figure 49: El Datacenter de azure en Suecia

#### • Dimensión social

Este trabajo no tiene muchos afectos sociales ya que no es una red social ni tampoco un video juego, sino es una hermanita que ayudar a desplegar aplicaciones en Azure.

Pero así de efectos Social lo k podemos considerar, es animarse a más gente a desplegar sus aplicaciones en la nube y dejar de tener sus aplicaciones en Onpremises, lo que permite a la gente poder gestionarse mejor tema de espacios y también tener menos preocupación por sus proyectos, ya que la parte física no la gestionen ellos.

### 7. Gestión económica

Ahora vamos a hablar sobre la gestión económica del proyecto final de carrera y vamos a dividir la gestión económica en dos partes: presupuestos y control de gestión.

#### 7.1. Presupuestos

Aquí en presupuestos vamos a discutir los presupuestos previstos para pagar

##### Coste personal

Como ya hemos discutido anteriormente los tres roles que necesitamos para el desarrollo del proyecto son jefe del proyecto, director de tecnologías y desenrollador.

En la siguiente tabla vamos a detallar el coste de cada rol por hora y también mensualmente vamos a calcular lo que nos va a costar.

|  |  |  |
| --- | --- | --- |
| **Rol** | **Coste por hora** | **Coste mensual** |
| *Jefe del proyecto* | 35 euro/Hora | 5600 euro |
| *Director de tecnologías* | 30 euro/Hora | 4800 euro |
| desenrollador | 20 euro/Hora | 3200 euro |

Figure 50: Tabla de salarios

##### Costes generales

Ahora vamos a hablar sobre lo que nos va a costar los recursos necesarios por el proyecto que básicamente son dos recursos (el Pc y la suscripción de Azure).

|  |  |
| --- | --- |
| Recurso | Coste |
| Portátil | 800 euro |
| Suscripción de Azure | 100 euro |

Figure 51: Tabla de los costos de los recursos

11

Aquí vamos a hacer la gestión del presupuesto, a nivel de tareas, también vamos a calcular los costes incluyendo el coste de la seguridad social (lo vamos a calcular multiplicando el coste por 1,3).

Así que lo que vamos a contar aquí, va a ser el coste personal por cada tarea por separado.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Tarea | Tiempo | Dependencia | Roles | Coste (euro) | CosteSS(euro) |
| GP  GP1  GP2  GP3  GP4  GP5  GP5.1  GP5.2  GP6 | 190  18  50  20  17  75  40  35  10 | -  -  GP1  GP2  -  DP  DP  DP  DP | J  J  J  J  J  J  J  J  J | 6,825  630  1,750  700  595  2,625  1,400  1,225  525 | 8,872.5  819  2,275  910  773.5  3,412.5  1,820  1,592.5  682.5 |
| TP  TP1  TP2 | 60  50  10 | GP1  GP1  GP1 | T  T  T | 1,800  1,500  300 | 2,340  1,950  390 |
| DP  DP1  DP1.1  DP 1.2  DP 1.3  DP 1.4  DP2  DP2.1  DP 2.2  DP3  DP3.1  DP3.2  DP4 | 250  125  30  30  40  25  55  30  25  50  35  15  20 | TP  TP  -  DP1.1  DP1.2  DP1.3  TP  -  DP2.1  TP  -  DP3.1  DP1 Y DP2 Y DP3 | D  D  D  D  D  D  D  D  D  D  D  D  D | 5,000  2,500  600  600  800  500  1100  600  500  1000  700  300  400 | 6500  3250  780  780  1,040  650  1,430  780  650  1300  910  390  520 |

Figure 52: Tabla de costes por tarea

##### Contingencia

Ahora vamos a calcula las Contingencia, las Contingencia la calculamos multiplicando el valor inicial por 15 porciento .

|  |  |  |
| --- | --- | --- |
| Tipo | Coste | Contingencia |
| Portátil | 800 euro | 120 euro |
| Azure suscripción | 100 euro | 15 euro |
| personal | 17,715.5 euro | 2,657.325 euro |
| total | 18,615.5 euro | 2,792.325 euro |

Figure 53: Tabla de Contingencia

### 7.2.Control de gestión

Una vez ya tenemos definido el presupuesto, vamos a definir los mecanismos para evitar desviaciones, por ejemplo, indicadores numéricos que nos van a ayudar con el control. En las reuniones del final de cada sprint, cada vez que se acabe una tarea, actualizamos el presupuesto con las horas reales y lo vamos a comprar con las horas estimadas desde el inicio.

Para poder controlar los imprevistos, al finalizar de cada tarea vamos a comparar los gastos extra que hemos gastado con los imprevistos y contingencia. De esta manera, podemos detectar las desviaciones y predecir si se necesita recortar alguna tarea o subir el presupuesto, ahora vamos a explicar los indicadores que vamos a usar para predecir las desviaciones.

* Desviación coste personal por tarea: (coste estimado − coste real)  horas reales
* Desviación realización tareas: (horas estimadas − horas reales) \* coste real
* Desviación coste recursos por tarea: (coste estimado − coste real)  horas reales
* Desviación total en la realización de tareas: coste estimado total − coste real total
* Desviación de coste de imprevistos: coste estimado imprevistos − coste real

#### 8. Bibliografía

1. (4 Companies Control 67% of the World's Cloud Infrastructure, 2022)PCMAG. 2022. *4 Companies Control 67% of the World's Cloud Infrastructure*. [online] Available at: <https://www.pcmag.com/news/four-companies-control-67-of-the-worlds-cloudinfrastructure> [Accessed 28 February 2022].
2. *Supporting our customers on the path to net zero: The microsoft cloud and decarbonization*. The Official Microsoft Blog. (2022, February 11). Retrieved March 21, 2022, from https://blogs.microsoft.com/blog/2021/10/27/supportingour-customers-on-the-path-to-net-zero-the-microsoft-cloud-and-decarbonization/ 3. 14, W. W. | D. (2022, February 9). *The Tech behind Microsoft's Sustainable Data Center Plan in Sweden*. Data Center Knowledge | News and analysis for the data center industry. Retrieved March 21, 2022, from

https://www.datacenterknowledge.com/energy/tech-behind-microsoft-ssustainable-data-center-plan-sweden

1. 14, W. W. | D. (2022, February 9). *The Tech behind Microsoft's Sustainable Data Center Plan in Sweden*. Data Center Knowledge | News and analysis for the data center industry. Retrieved March 21, 2022, from

https://www.datacenterknowledge.com/energy/tech-behind-microsoft-ssustainable-data-center-plan-sweden

1. *Microsoft sources 24/7 renewable energy for data centers*. ESG Today. (2021, November 4). Retrieved March 21, 2022, from

https://www.esgtoday.com/microsoft-sources-24-7-renewable-energy-for-datacenters/#:~:text=Unveiled%20in%20July%202021%2C%20the,and%20datacenter s%20globally%20by%202025.

1. Pimpalnerkar, A. (2021, December 22). *How to create and deploy a python Function App using Azure devops CI/CD*. Medium. Retrieved March 21, 2022, from https://medium.com/globant/how-to-create-and-deploy-a-python-azure-functionusing-azure-devops-ci-cd-2aa8f8675716
2. Juliakm. (n.d.). *Build and deploy Python Web Apps - Azure Pipelines*. Build and deploy Python web apps - Azure Pipelines | Microsoft Docs. Retrieved March 21,

2022, from https://docs.microsoft.com/en-

us/azure/devops/pipelines/ecosystems/python-webapp?view=azure-devops

1. Azooinmyluggage. (n.d.). *Deploy to azure kubernetes service with azure pipelines*

*- azure pipelines*. Deploy to Azure Kubernetes Service with Azure Pipelines -

Azure Pipelines | Microsoft Docs. Retrieved March 21, 2022, from https://docs.microsoft.com/en-

us/azure/devops/pipelines/ecosystems/kubernetes/aks-template?view=azuredevops&pivots=pipelines-yaml