Manual de configuraciones

Hugo Jiménez Chévez

DevOps

Versión: 1.0

Octubre, 2023

**Contenido**

[**1.** **Introducción** 3](#_Toc148967602)

[**2.** **Herramientas para utilizar.** 3](#_Toc148967603)

[**2.1.** **Sonarqube** 3](#_Toc148967604)

[**2.1.1.** **Dependencias java** 5](#_Toc148967605)

[**2.2.** **Docker** 6](#_Toc148967606)

[**2.3.** **Azure Agent Pool SelfHosted** 9](#_Toc148967607)

[**2.4.** **Container Registry** 12](#_Toc148967608)

[**2.5.** **Kubernetes** 16](#_Toc148967609)

[**3.** **Azure DevOps** 22](#_Toc148967610)

[**3.1.** **Pipeline** 23](#_Toc148967611)

[**3.1.1. Pipeline con tareas de azure** 23](#_Toc148967612)

[**3.1.2. Pipeline yaml** 42](#_Toc148967613)

# **Introducción**

El uso de despliegues en Kubernetes representa un pilar fundamental en la gestión de contenedores y la orquestación de aplicaciones en entornos de infraestructura escalable. A través de este mecanismo, las organizaciones pueden lograr una mayor eficiencia en la implementación y actualización de sus aplicaciones, garantizando la disponibilidad, escalabilidad y fiabilidad deseada. En esta introducción, exploraremos en detalle los conceptos clave, ventajas y prácticas recomendadas para aprovechar al máximo los despliegues de Kubernetes.

# **Herramientas para utilizar.**

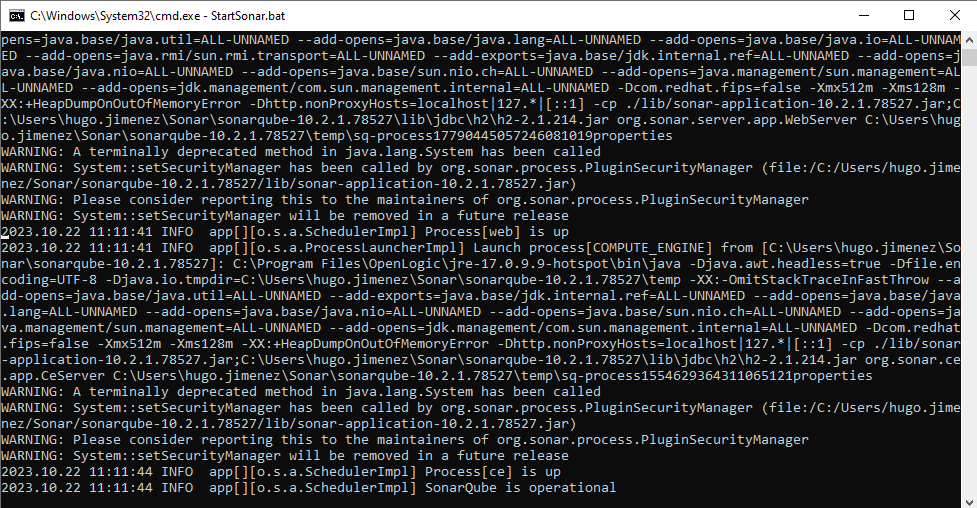
## **Sonarqube**

Para utilizar SonarQube, es imprescindible descargar los complementos necesarios que permitirán la puesta en marcha de SonarQube. Estos complementos se encuentran disponibles para su descarga en la página oficial de SonarQube.

<https://www.sonarsource.com/products/sonarqube/downloads/success-download-community-edition/>

En esta ocasión, es necesario descargar la versión "Community" de SonarQube. Una vez descargada, se obtiene un archivo ZIP que debemos ubicar en la carpeta de destino donde deseamos descomprimirlo y nos ubicamos dentro de la carpeta /bin/Windows-86-64. Luego, procedemos a abrir una ventana de comandos (CMD) para ejecutar el siguiente comando.

Ejecutar en cmdshell > StartSonar.bat



El cual nos ejecutaran los procesos y unas vez que indique que el sonar está en ejecución procedemos a levantarlo

Interfaz de usuario gráfica, Aplicación

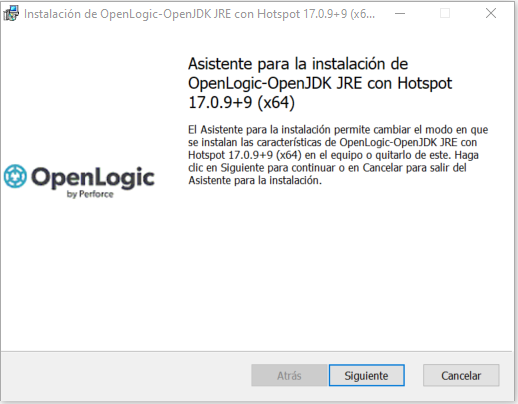
Descripción generada automáticamente

### **Dependencias java**

Es importante mencionar que para que la aplicación de Java funcione es necesario instalar el complemento de java el cual se obtiene del siguiente enlace de open logic.

<https://www.openlogic.com/openjdk-downloads?field_java_parent_version_target_id=807&field_operating_system_target_id=436&field_architecture_target_id=All&field_java_package_target_id=401>

El cual nos descargara un ejecutable y solo debemos dar clic sobre el y nos levantara la siguiente pantalla en el cual la instalación es muy sencilla, ya que solo es necesario darle en siguiente en todos los pasos y se instalara sin problemas.



## **Docker**

Para aprovechar Docker al máximo, es esencial obtener los componentes necesarios que habilitarán la implementación de SonarQube. Estos componentes se encuentran disponibles para su descarga en el sitio web oficial de Docker.

<https://docs.docker.com/desktop/install/windows-install/>

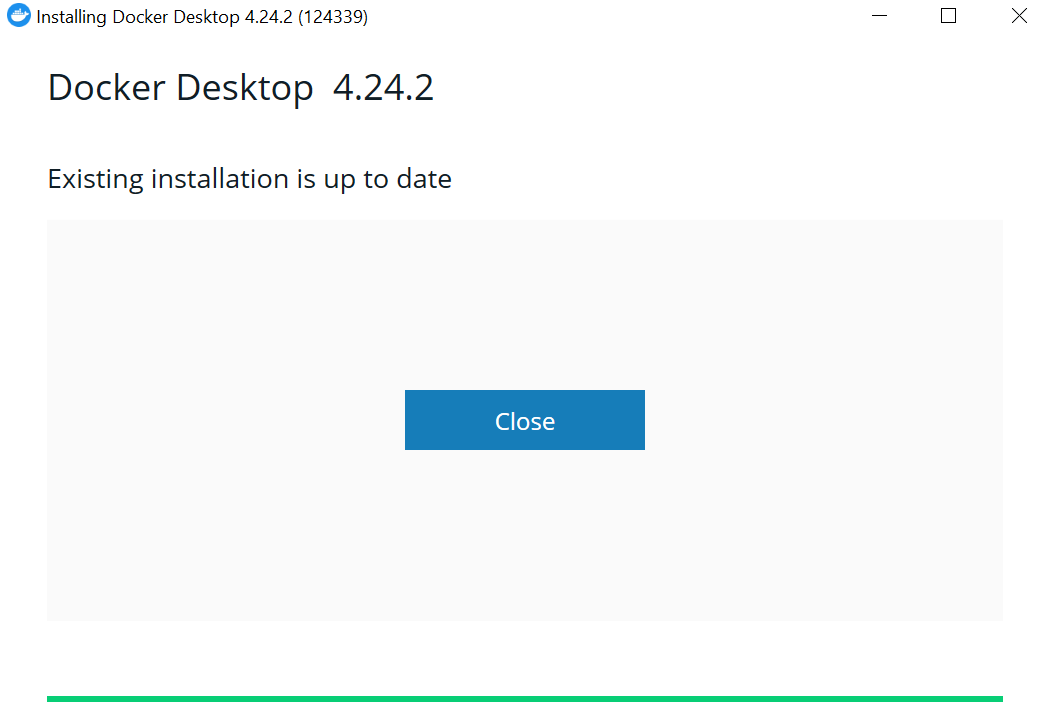
Es importante mencionar que primero debemos tener una cuenta en Docker hub para instalar la versión de Docker de escritorio, ya que es esta es necesaria para las compilaciones de más adelante.

Dicho lo anterior, ubicamos el ejecutable que descargamos del enlace anterior y solo debemos dar clic y nos saldrá la siguiente pantalla, la cual debemos mantener ambas opciones que nos aparecen marcadas y le damos “ok”.

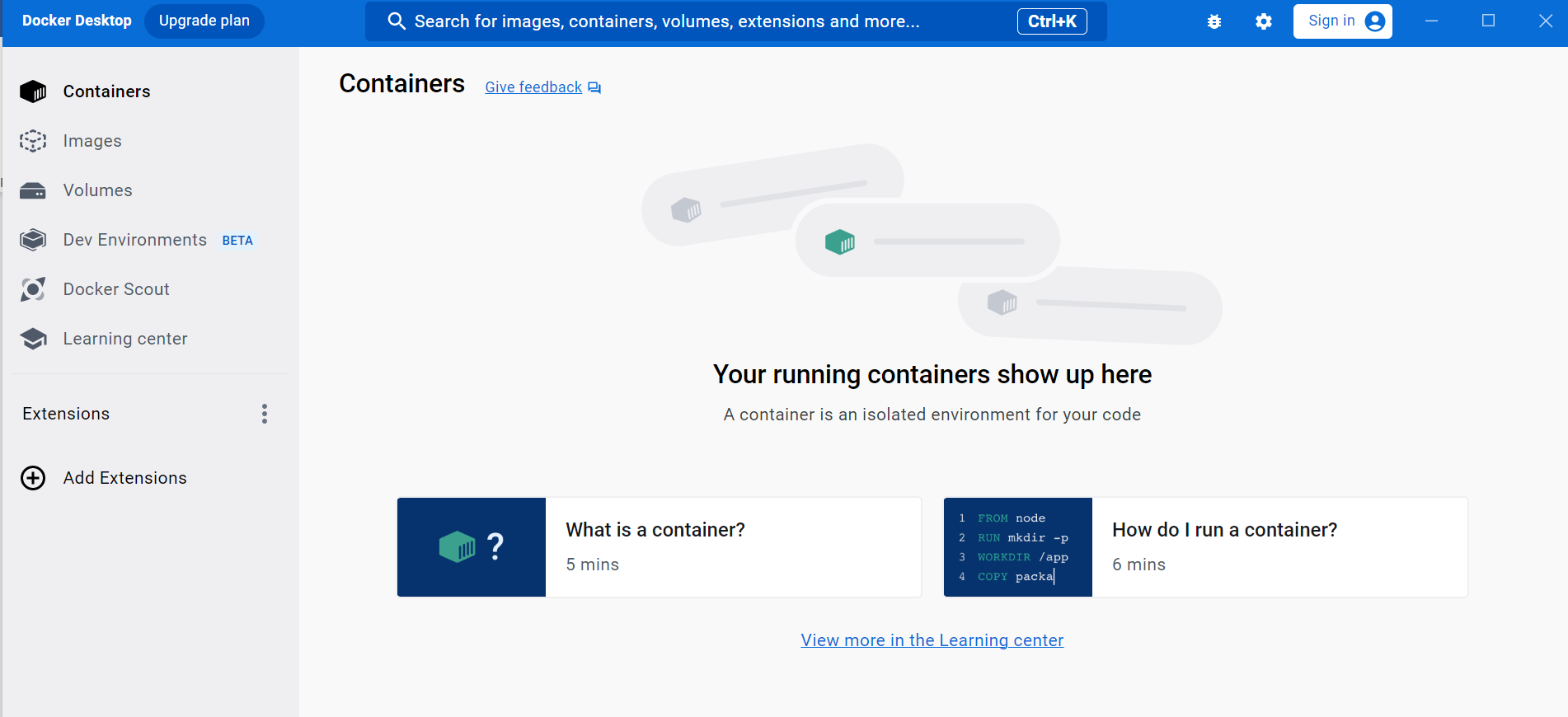
Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

Una vez que le hayamos dado en “ok”, comenzara la instalación y solo debemos esperar que termine y le damos en “close”.



Y al levantar la aplicación nos debería salir de la siguiente forma.



## **Azure Agent Pool SelfHosted**

Para Instalar el azure agent pool para las compilaciones, es crucial adquirir los componentes necesarios que permitirán la implementación de este agente. Estos componentes están disponibles para su descarga en Azure DevOps.

Por lo que debemos seguir los siguientes pasos.

Primeramente, se debe ingresar en azure DevOps en la sección de “Organization Setting”.

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

Una vez ingresado, procedemos a ingresar en la sección de “Agent pool”

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

Luego, nos dirigimos a agregar nuevo pool “Add pool”

Tabla

Descripción generada automáticamente con confianza media

En la pantalla que nos saldrá después de dar clic en agregar nuevo pool, debemos seleccionar la opción de “Self hosted”

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

Una vez hayamos elegido la opción anterior debemos llenar los siguientes datos.

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

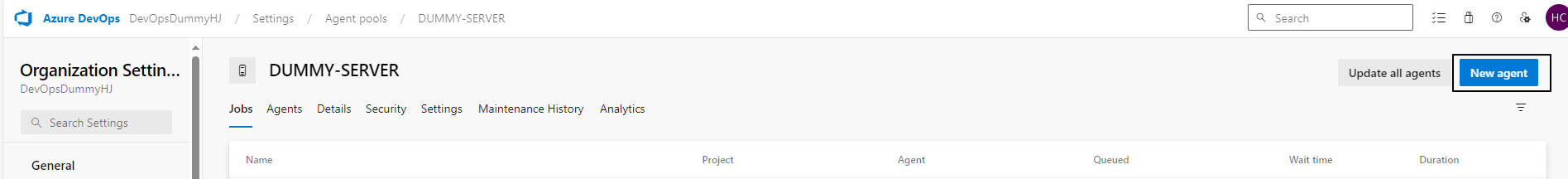
Descripción generada automáticamente

Posteriormente, seleccionamos el agente que creamos

Patrón de fondo

Descripción generada automáticamente con confianza media

Y luego seleccionamos la opción “New agent”



El cual nos desplegara la siguiente pantalla con toda la información necesaria.

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

Una vez que hayamos hecho todos los pasos anteriores, se creara un nuevo servicio windows en el cual debemos asignar un usuario con permisos administradores para acceder a todos los complementos necesarios para las compilaciones.



## **Container Registry**

Para configurar nuestro contenedor donde almacenaremos nuestras imágenes Docker, debemos hacer los siguientes pasos, primeramente, ingresamos al portal del azure.

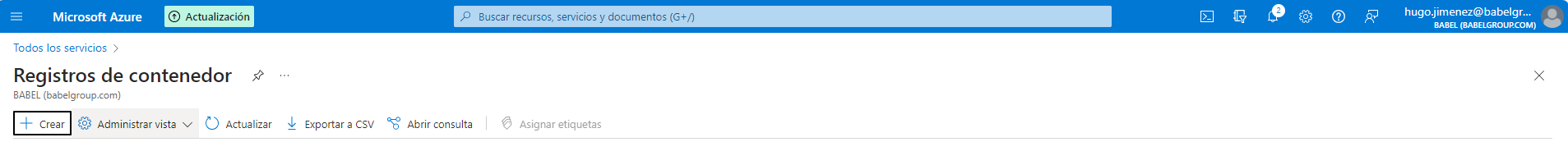
[**https://portal.azure.com/**](https://portal.azure.com/)

Posteriormente, buscamos todos los servicios y nos dirigimos en la sección de contenedores y elegimos la opción de “Registros de contenedores”.

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

Una vez hayamos ingresado en los contenedores, procedemos a crear uno nuevo.



Una vez que haga clic en "Crear", se abrirá la siguiente pantalla, donde se le solicitará la información del contenedor. Debe proporcionar el nombre del grupo de recursos (es fundamental utilizar el mismo para esta configuración), asignar un nombre al contenedor, seleccionar una ubicación geográfica y elegir el plan de precios estándar.

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

En la siguiente sección dejamos los datos por defectos y continuamos.

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

En esta pantalla de cifrado, dejamos todo por defecto.

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

Con respecto a las etiquetas, pueden omitir esta parte, pero es importante asignarle algunos para referencias a que proyecto pertenece, quien lo creo o la descripción del recurso creado.

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

Como paso final, solo validamos que los datos estén correctos y procedemos a crear el “registro de contenedor”

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

## **Kubernetes**

Para configurar nuestro Kubernetes donde desplegaremos nuestras imágenes almacenadas en el contenedor, debemos hacer los siguientes pasos, primeramente, ingresamos al portal del azure.

[**https://portal.azure.com/**](https://portal.azure.com/)

Posteriormente, buscamos todos los servicios y nos dirigimos en la sección de contenedores y elegimos la opción de “Servicios de Kubernetes”.

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

Una vez hayamos ingresado en los servicios de Kubernetes, procedemos a crear uno nuevo y elegimos la opción para esta prueba “Crear un clúster de Kubernetes”.

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

Una vez que haga clic en "Crear", se abrirá la siguiente pantalla, donde se le solicitará la información del Kubernetes. Debe proporcionar el nombre del grupo de recursos (es fundamental utilizar el mismo para esta configuración), asignar la configuración preestablecida del clúster, asignar un nombre al Kubernetes, seleccionar una ubicación geográfica, zonas de disponibilidad, plan de tarifa (en este caso se usó el estándar) y por último la autenticación.

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

En la siguiente sección se deja los valores por defecto.

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

En la siguiente pantalla en redes, se utilizo la conexión establecida por defecto.

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

En la siguiente sección de integraciones debemos asignar el contenedor que creamos anteriormente, de esta forma nos será más fácil realizar los despliegues de las imágenes.

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

En opciones avanzadas dejamos por defecto el grupo de recursos que se crear.

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

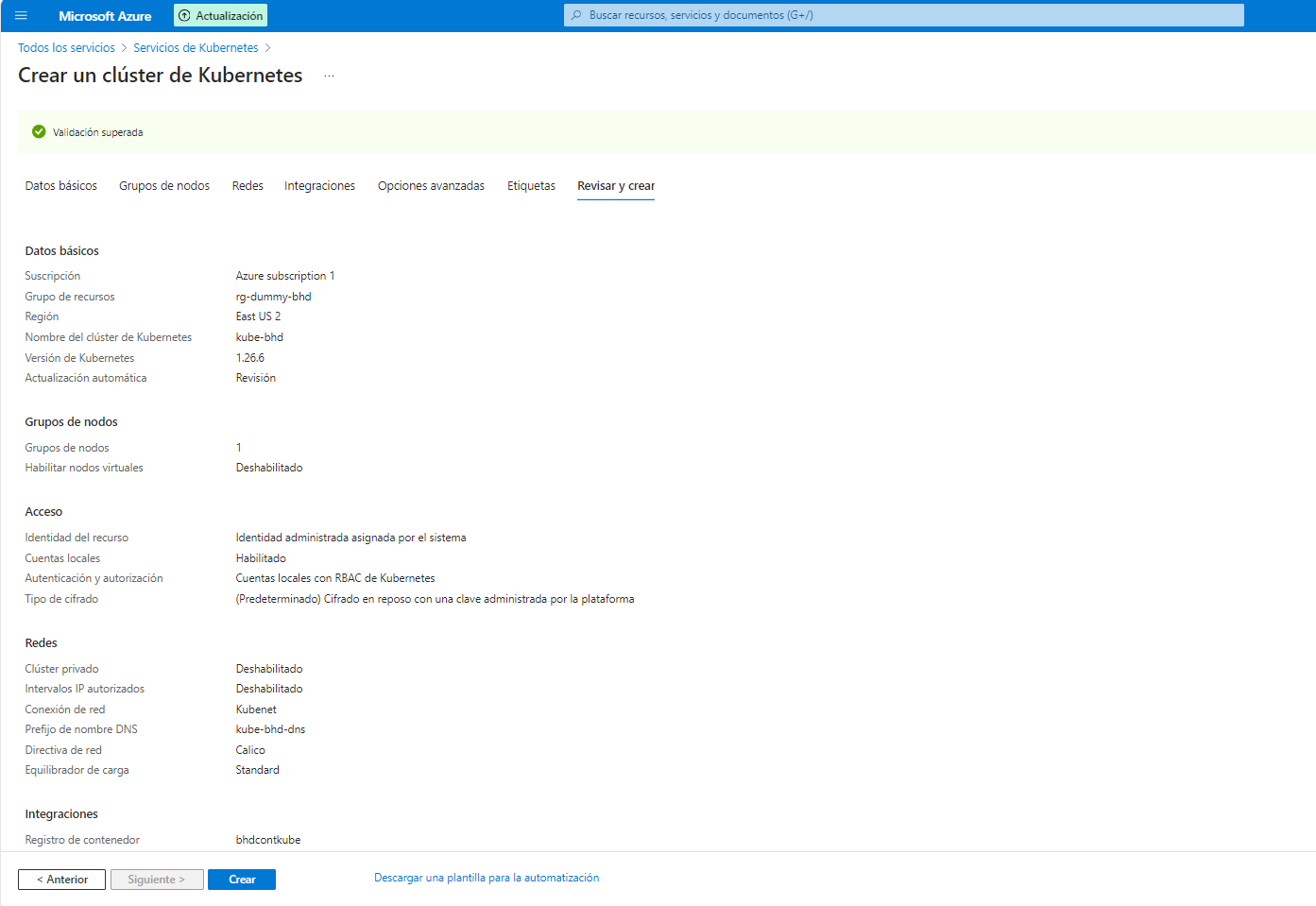
Descripción generada automáticamente

En cuanto a las etiquetas, es posible pasar por alto esta sección, sin embargo, resulta crucial asignar algunas para tener referencias sobre a qué proyecto pertenece, quién lo creó o la descripción del recurso creado.

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

Y por último solo debemos revisar que la aplicación este correcta y se procede a crear el Kubernetes.



Una vez creado el Kubernetes debemos asignar la opcion que permita el balanceo, esto es muy importante para poder levantar la aplicaición.

En este caso procedemos a dirigirnos a redes

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

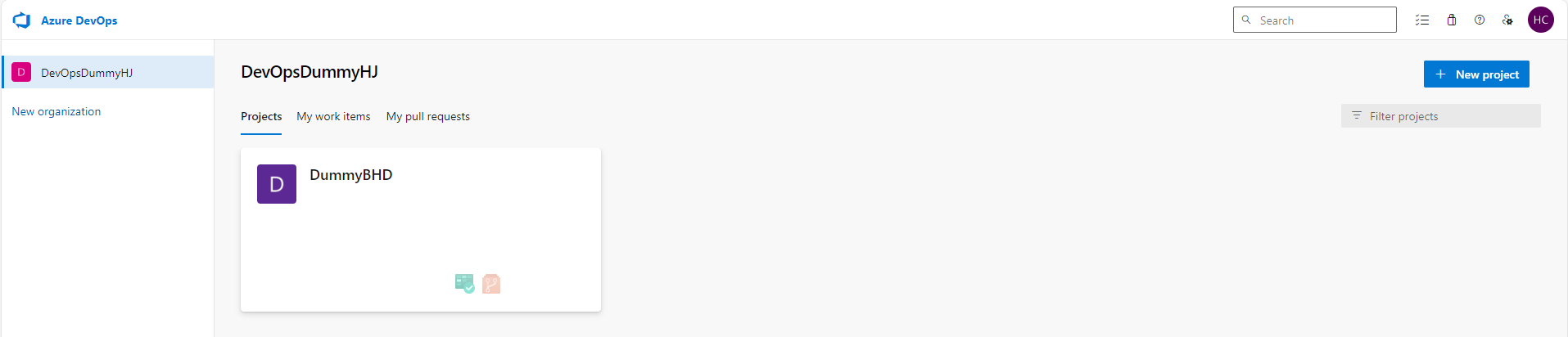
Una vez en la sección de redes, debemos habilitar la opción Habilitar enrutamiento de solicitudes HTTP y aplicar el cambio, y con esto queda listo nuestro Kubernetes para los despliegues.

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

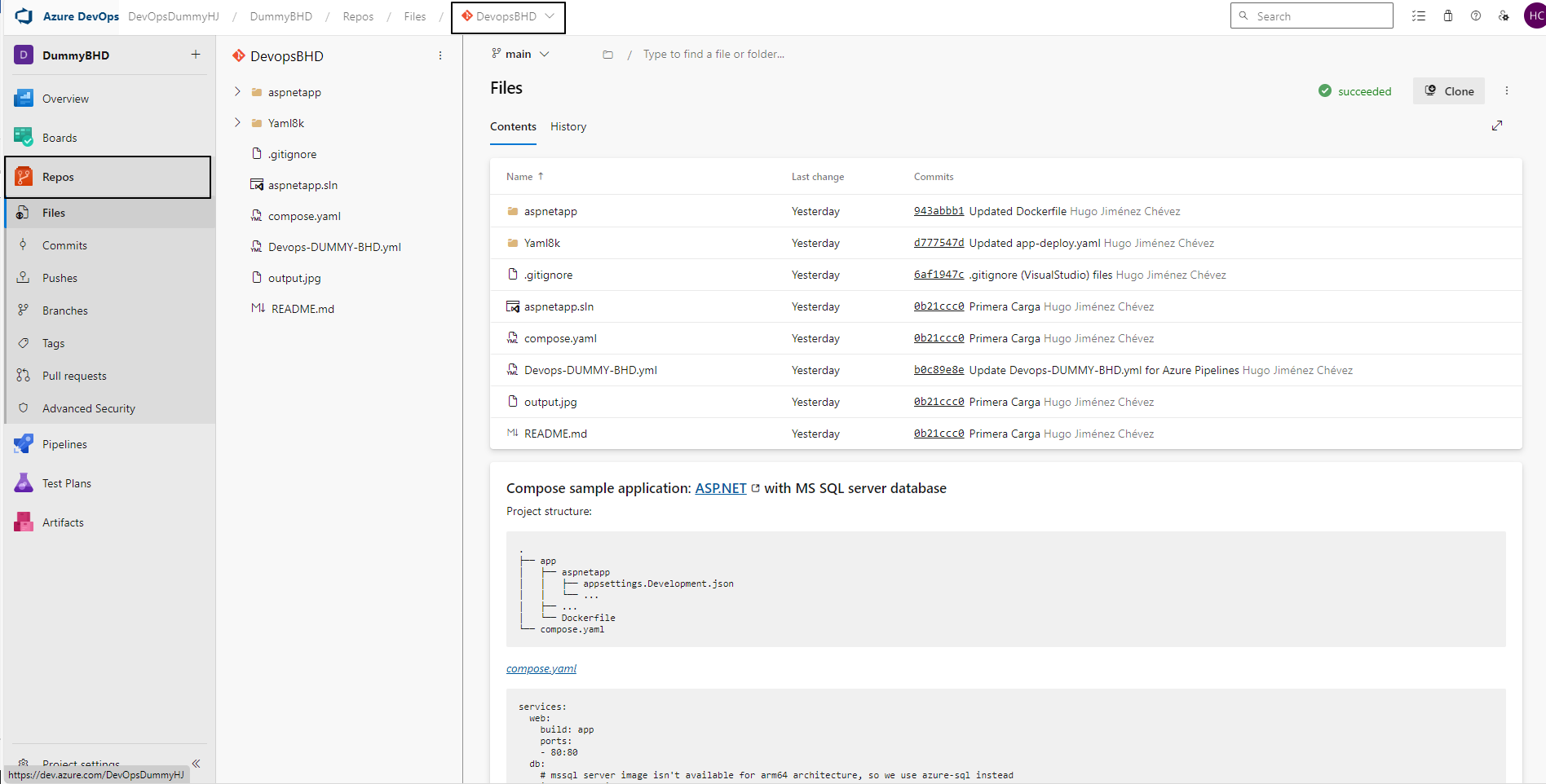
Descripción generada automáticamente

# **Azure DevOps**

Primeramente, debemos crear un team Project donde realizaremos todas las configuraciones de los despliegues.



Con anterioridad se debió versionar el código fuente proporcionado para la prueba, ya que este viene con las configuraciones necesarias de Docker.



Posteriormente, se procede a configurar los pipelines.

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

## **Pipeline**

### **3.1.1. Pipeline con tareas de azure**

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

Como primer paso se asignó el Azure Agent Pool SelfHosted, ya que el agente posee todas los complementos para compilar las tareas.

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

**SonarQube**

Se optó por comenzar utilizando la tarea de SonarQube, la cual se configuró inicialmente para establecer la comunicación con SonarQube, donde se publicarán los resultados del análisis.

Una captura de pantalla de una computadora

Descripción generada automáticamente

Para realizar la comunicación, se tuvo que agregar la siguiente configuración de la cual nos solicitaba.

**Server Url:** Se debe agregar la url que levanta la página de Sonarqube

**Token:** En Sonarqube podemos generar el token que nos solicitan

**Services Connection name:** Nombre de la conexión que le queremos asignar.

**Grant access:** Se debe marcar la opción para tener permisos desde azure**.**

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

Para generar el token, debemos realizar lo siguiente:

Debemos crear un nuevo proyecto en Sonarqube, en este caso que sea para Azure DevOps.

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

Si ya lo tenemos creado, solo le damos en “import” para crear el nuevo proyecto.

Interfaz de usuario gráfica, Aplicación

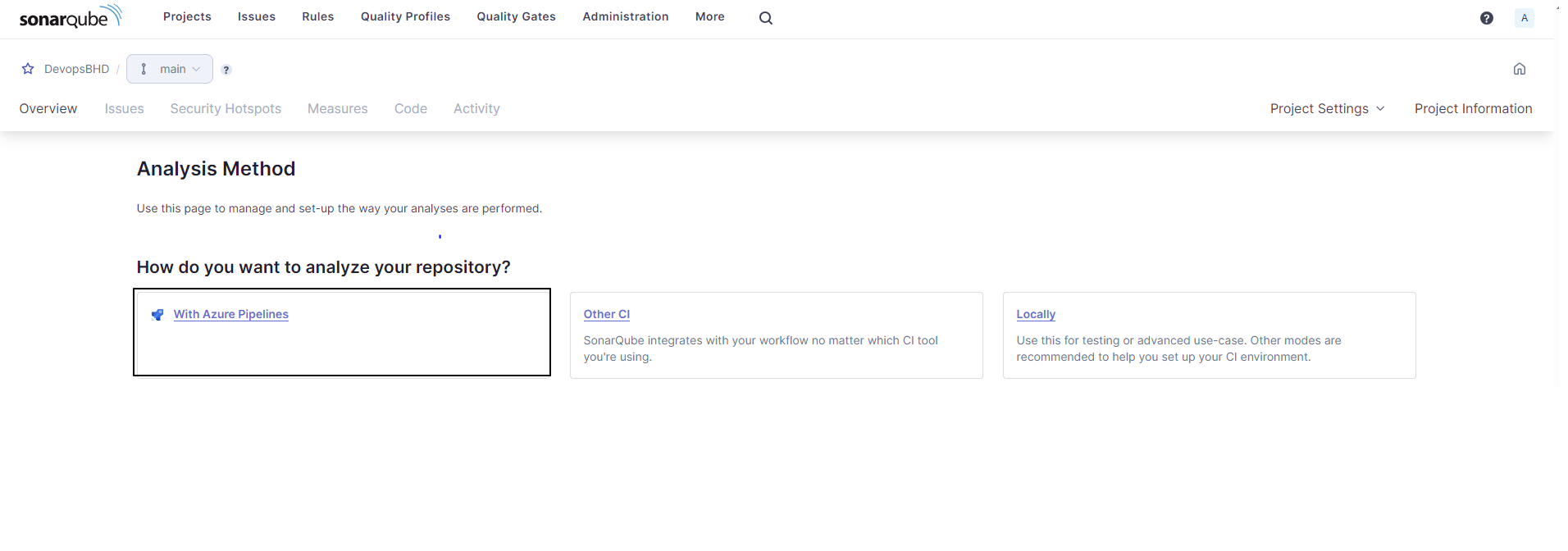
Descripción generada automáticamente

En esta sección elegiremos la opción que decidamos y le damos en crear proyecto.

Interfaz de usuario gráfica, Texto, Aplicación, Teams

Descripción generada automáticamente

Posteriormente, en la siguiente pestaña seleccionamos azure pipelines para hacer la comunicación.



Una vez realizadas los pasos anteriores, nos saldrá la información necesaria para crear el token.

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

Para generar el token, agregamos la siguiente información.

**Token Name:** Nombre del token.

**Expires In:** Cuanto tiempo expirara el token.

Y le damos en generar y copiamos el token y una vez realizado estos pasos le damos en continuar.

Interfaz de usuario gráfica, Texto, Aplicación, Chat o mensaje de texto

Descripción generada automáticamente

Luego, para terminar de configurar Sonarqube, se agregan las siguientes tareas: “Run Code Analysis”, que nos permitirá escanear el código compilado.

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

Y además, utilizaremos la tarea de “Publish Quality gate Result” que es la que se encargara de publicar los resultados en Sonarqube.

Captura de pantalla de computadora

Descripción generada automáticamente

Una vez terminado las tareas de Sonarqube, al ejecutar deberíamos poder observar lo siguiente en la herramienta de sonar.

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

**Compilador de solución.**

Para compilar la solución se utilizó las tareas de Netcore (Ya que el proyecto utilizado es .Net 5)

Captura de pantalla de computadora

Descripción generada automáticamente

Las tareas son “donet restore” que es la que permite restaurar los paquetes NuGet, “donet build” que es la tarea que se encarga de compilar las solución y “donet publish” que se encarga del empaquetado.

**Docker**

**Container Registry.**

Una vez que hayamos compilado la solución, procederemos con la publicación de la imagen Docker, por lo que usaremos la siguiente tarea para convertir la aplicación en imagen y publicar en los contenedores.

Captura de pantalla de computadora

Descripción generada automáticamente

En la tarea de Docker nos pide la siguiente información:

Container Registry: Es la conexión que debemos configurar para la comunicación entre azure DevOps y el contenedor.

Container repository: Es el nombre que le asignamos al contenedor en Azure.

Commands: La acción a ejecutar.

Dockerfile: El archivo Docker que se encarga de transformar la solución en una imagen.

Para la conexión debemos rellenar con los siguiente datos.

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

Una vez que hayamos configurado la conexión y al ejecutar el pipeline, en nuestro contenedor deberían aparecer las imágenes correspondientes.

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

**Docker Hub**

Una vez que hayamos compilado la solución, procederemos con la publicación de la imagen Docker, por lo que usaremos la siguiente tarea para convertir la aplicación en imagen y publicar en los contenedores. En este caso publicaremos en Docker Hub.

Captura de pantalla de computadora

Descripción generada automáticamente

En la tarea de Docker nos pide la siguiente información:

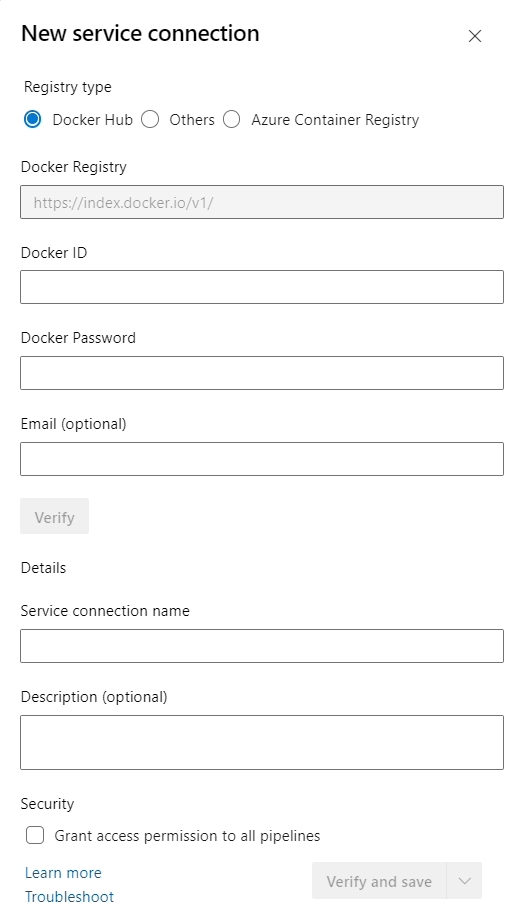
**Container Registry:** Es la conexión que debemos configurar para la comunicación entre azure DevOps y Docker hub.

**Container repository:** Es el nombre que le asignamos al contenedor en Docker hub.

**Commands:** La acción a ejecutar.

**Dockerfile:** El archivo Docker que se encarga de transformar la solución en una imagen.

Para la conexión debemos rellenar con los siguiente datos



Una vez que hayamos configurado la conexión y al ejecutar el pipeline, en nuestro contenedor deberían aparecer las imágenes correspondientes.

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

Si lo que necesitamos es validar que la imagen haya quedado funcional, debemos hacer lo siguiente.

Debemos mantener en ejecución la aplicación de escritorio de Docker, luego abrimos un cmd y ejecutamos los siguientes comandos.

docker push hugojch/dummybhd:tagname

docker run -d -p 8080:80 --name appnet hugojch/dummybhd:74.

Con esto lo que hacemos es levantar el Docker por localhost y el puerto asignado.



**Kubernetes**

Una vez que hayamos generado la imagen y subida al container Registry, procederemos con la publicación de la imagen Docker a Kubernetes por usaremos la siguiente tarea (kubectl).

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

En la tarea de Kubernetes nos pide la siguiente información:

**Services Connction type:** Es la conexión que necesitaremos crear para la comunicación entre Azure Devops y Kubernetes.

**Azure Subscription:** Es la suscripción donde está alojado el Kubernetes.

Resource group: Grupo de recursos asignado del Kubernetes.

**Kubernetes cluster:** Kubernetes creado para la publicación de las imágenes.

**Use cluster admin credentials:** Se debe marcar esta opción.

**Namespace:** En este caso usaremos el default.

**Command:** Acción que deseamos que se realice.

**Use configuration:** Se debe marcar esta opción para utilizar el yaml que contiene ciertos valores a utilizar.

**File path:** Carpeta donde está alojado el yaml.

**Secret:** Configuración del contenedor.

**Type of Secret:** En este caso usaremos Container Registry.

**Container Registry name:** Nombre del contenedor donde se encuentran las imágenes

**Force update secret:** Se utiliza para forzar la actualización de las imágenes.

Con respecto al yaml utilizado, esto fue lo que se configuro con el fin de crear ciertas características para el funcionamiento del Kubernetes.

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

Además, este yaml nos crear un load balancer que nos permitirá sacar la url pública.



Una vez que hayamos configurado la tarea y al ejecutar el pipeline, en nuestro Kubernetes deberían aparecer las imágenes correspondientes

Interfaz de usuario gráfica, Aplicación, Tabla

Descripción generada automáticamente

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

Para validar que la aplicación quedo publicada y funcional, procedemos a ingresar por medio del load balancer configurado en el yaml anterior.

http://20.81.72.155/

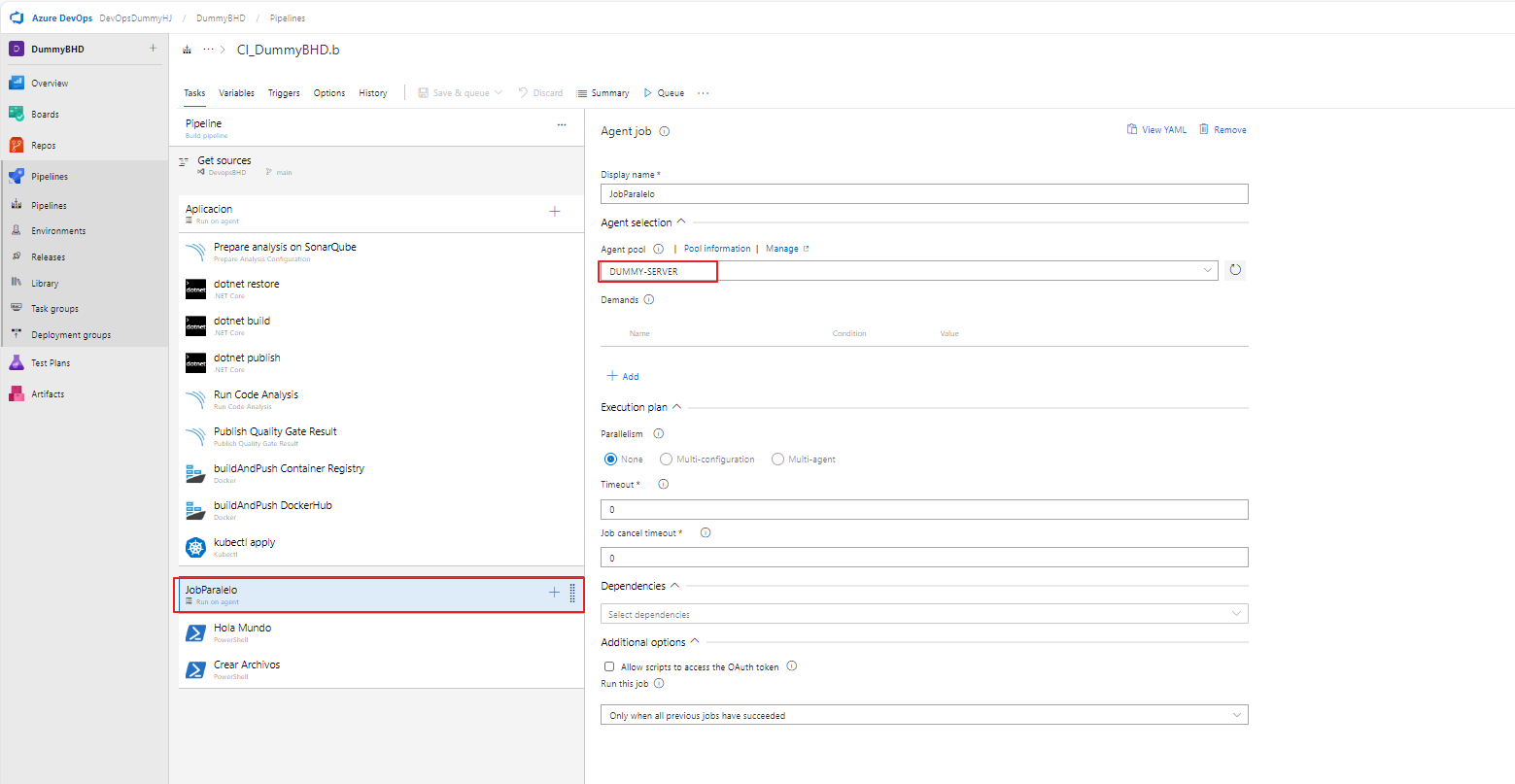
Y de esta forma podemos observar que la publicación fue exitosa.

Interfaz de usuario gráfica

Descripción generada automáticamente

**Configuración Job paralelo.**

Como primer paso se asignó el Azure Agent Pool SelfHosted, ya que el agente posee todas los complementos para compilar las tareas.



Seguidamente, se configuro un script por medio de PowerShell que imprime 10 veces el “hola mundo”

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

A continuación, en los logs podemos observar que efectivamente se imprimen los hola mundo.

Texto

Descripción generada automáticamente

Posteriormente, se procede a crear un script que crea 10 archivos con la fecha y los imprime

Captura de pantalla de computadora

Descripción generada automáticamente

Y como podemos observar a continuación los archivos con las fehas.

Texto

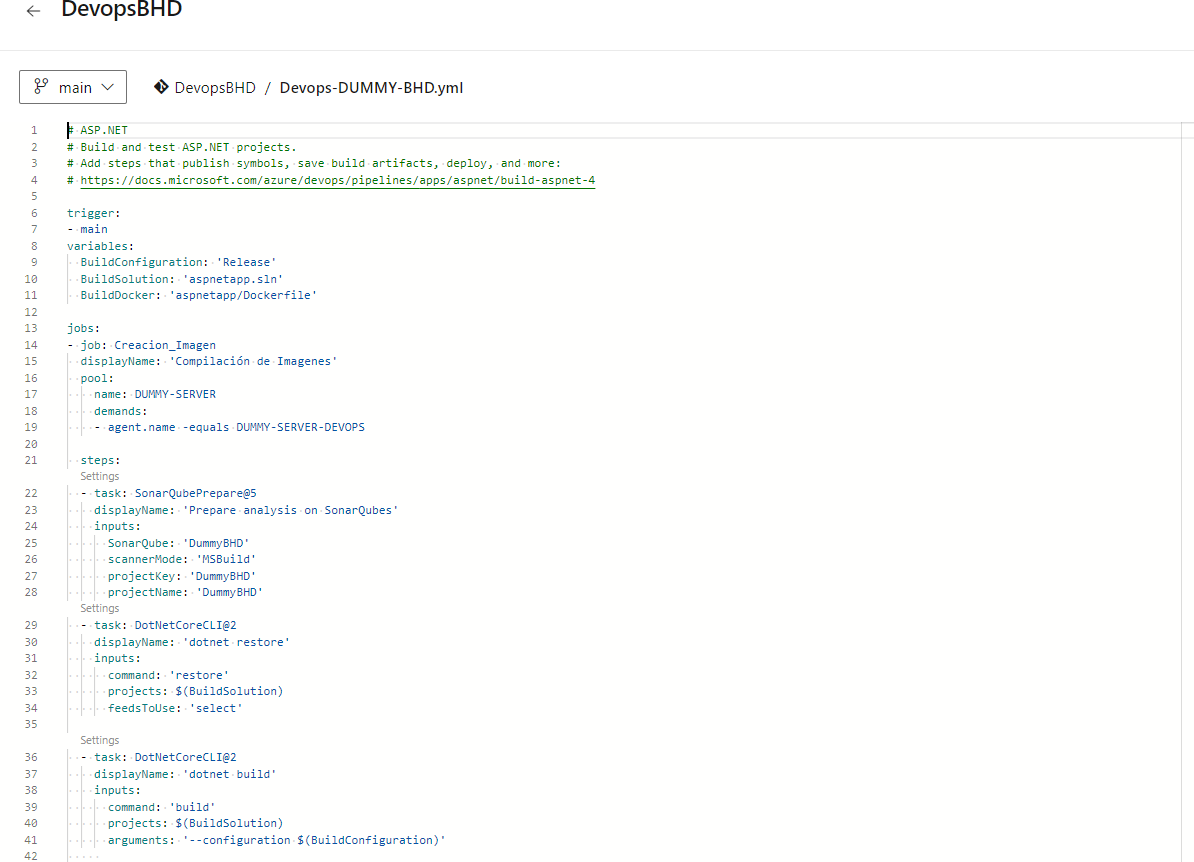
Descripción generada automáticamente

### **3.1.2. Pipeline yaml**

Aparte de las tareas de azure, se creo otro pipeline, solo que esta vez utilizando yaml para configuración del proceso devops, las configuraciones son las mismas que lo explicado anteriormente.

**Jobs 1**

Se configuro las tareas en yaml para la construcción de la imagen Docker y su despliegue hacia Kubernetes.



Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

**Jobs 2**

En el segundo jobs del Yaml se configuro las pruebas de imprimir 10 hola mundos y la creación de los archivos con su fecha usando PowerShell.

Interfaz de usuario gráfica, Texto

Descripción generada automáticamente