\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Configuracion de Taller de DEVOPS**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Elaborado por:**

Pedro Abel

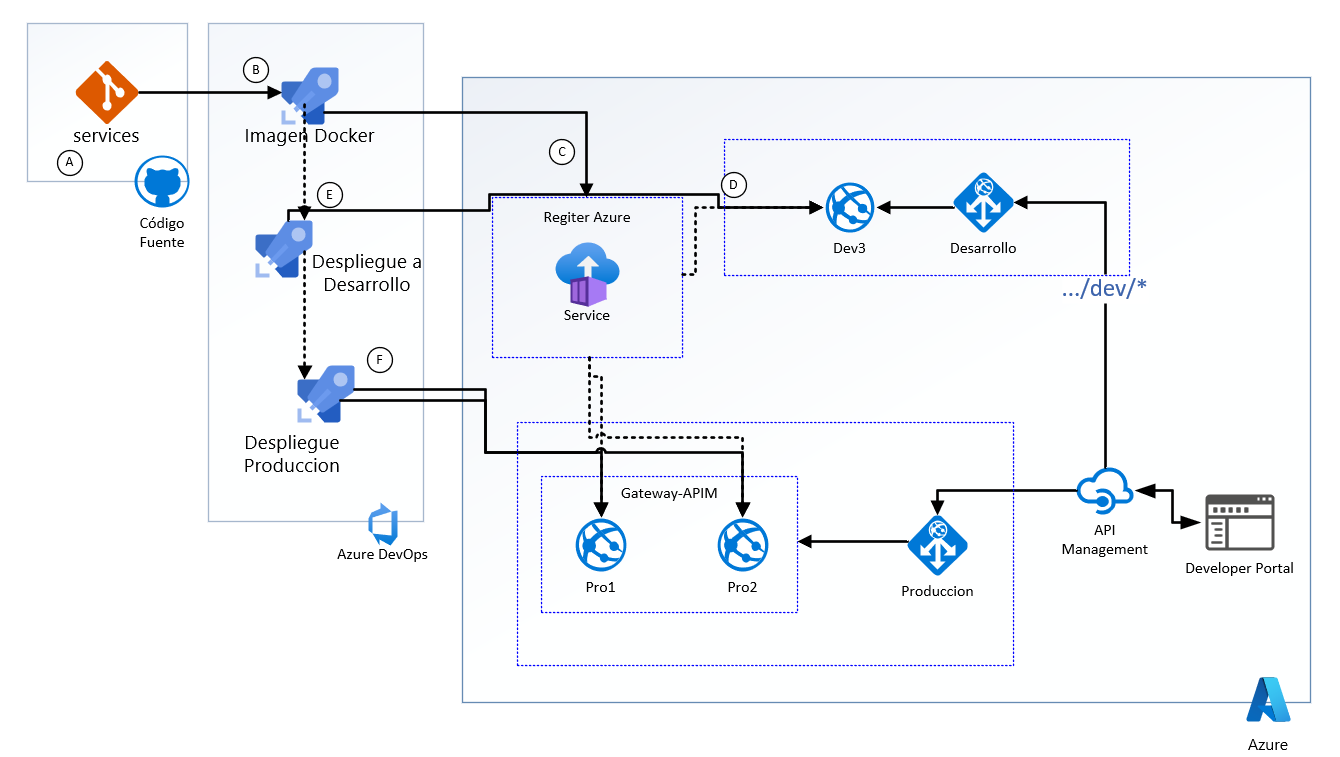
# Objetivo del documento

Este documento tiene como alcance detallar la infraestructura del taller de Devops, su alcance y las configuraciones necesarias para la replicación del escenario.

# Alcance

Tiene como alcance el poder replicar el escenario inicial y detallar las diferentes estructuras utilizadas durante el taller.

# Arquitectura de la implementación



El despliegue de la aplicación consta de 5 fases antes de ponerla en producción:

1. Cuando el equipo de desarrollador sube cambios en los repositorios empresariales, según estándares propios de la empresa se mantienen 2 troncales de trabajo: Development y Main.
2. Cuando se sensa cambios en el código fuente, se procede a generar la imagen Docker dependiente de la rama principal en la que se este trabajando se guardan con sus etiquetas.
3. Cuando una incorporacion de cambios se detecta de la rama:
   1. Desarroll: se procede a generar una imagen en el registro de repositorio en Azure con el postfijo: -DEV y como tag el numero de la versión de compilación.
   2. Main: no se agrega ningún postfijo y la etiqueta que se coloca es “LATEST”.
4. Una vez que la imagen en desarrollo se coloco en el registro de repositorio, se procede al despliegue continuo en desarrollo.
5. Se tiene filtrado que los cambios realizados en cualquier rama que no sea main no genera despliegue automático.