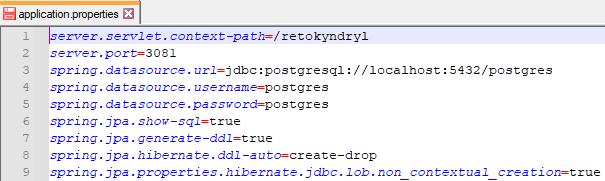
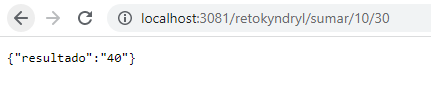
**Informe del reto técnico de Kyndryl**

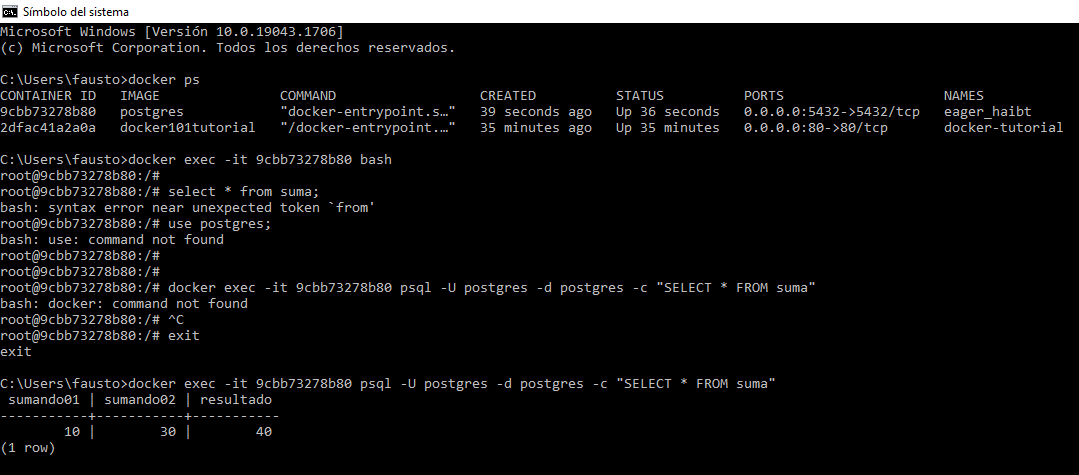
1. Almacenar los números y el resultado en una base de datos: Se utilizó Postgres como Base de Datos local y se registró su URL y credenciales en el archivo application.properties.



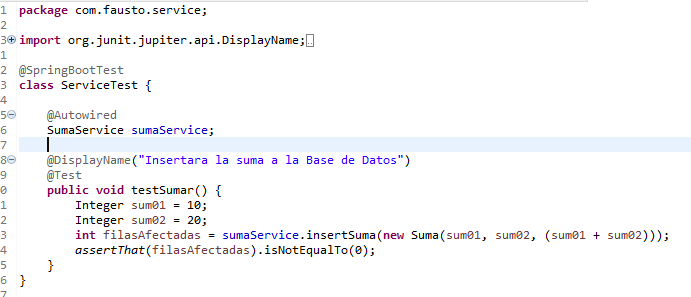
Se generó el backend usando Java y, a modo de prueba, se desplegó y probó el proyecto java en el servidor Nginx. A continuación se ve una ejecución del backendo usando el 10 y el 30:

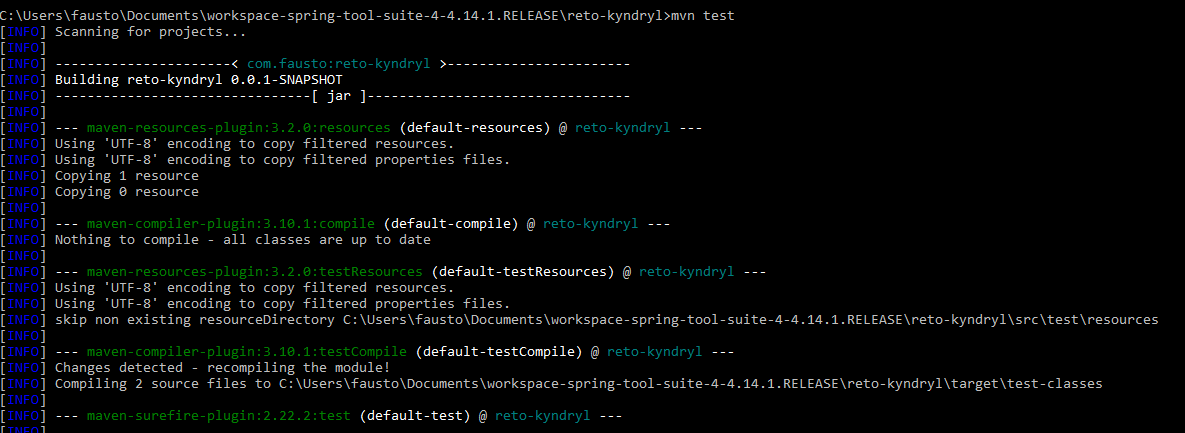


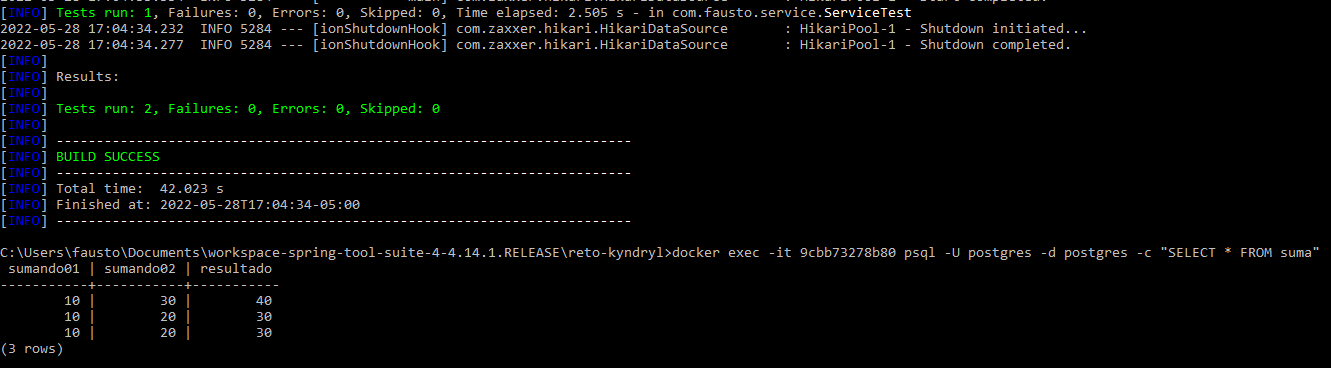
Se verificó que los sumandos y la suma si se almacenaron en la Base de Datos Postgres:



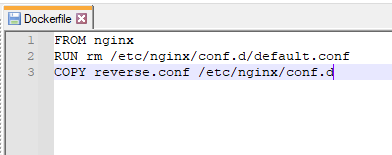
1. Generar un bashero/yml que realice la compilación del backend de forma automática, ejecutando los test unitarios: Se adjuntan las siguientes pantallas para evidenciar que se realizarom los test en java y que las pruebas de estas, en maven, se ejecutaron correctamente.

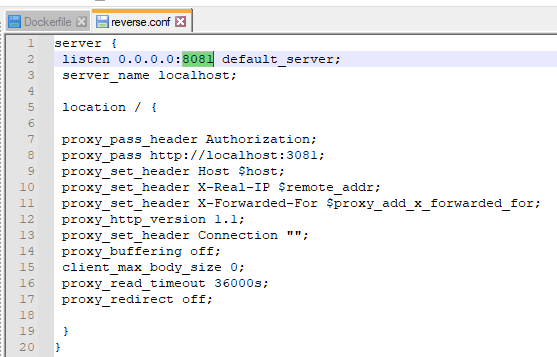


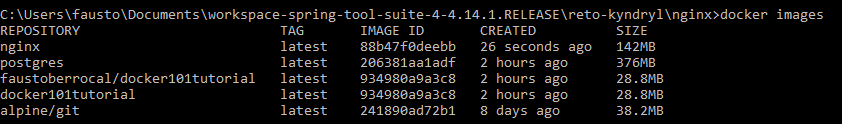




2. Generar un DockerFile, que permita construir el build de una imagen de Nginx como reverse-proxy: Se presenta el archivo Dockerfile solicitado, el archivo reverse.conf y la imagen creada tras su ejecución.

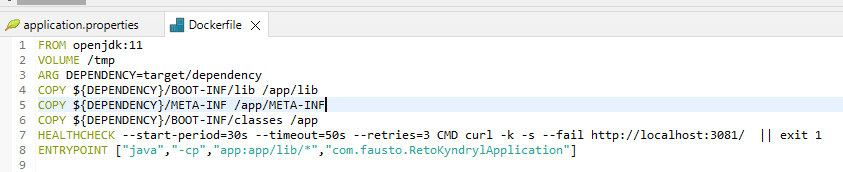


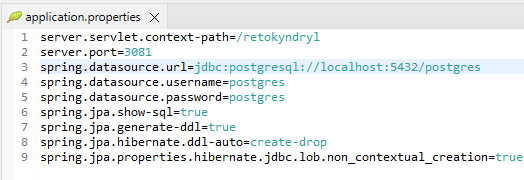


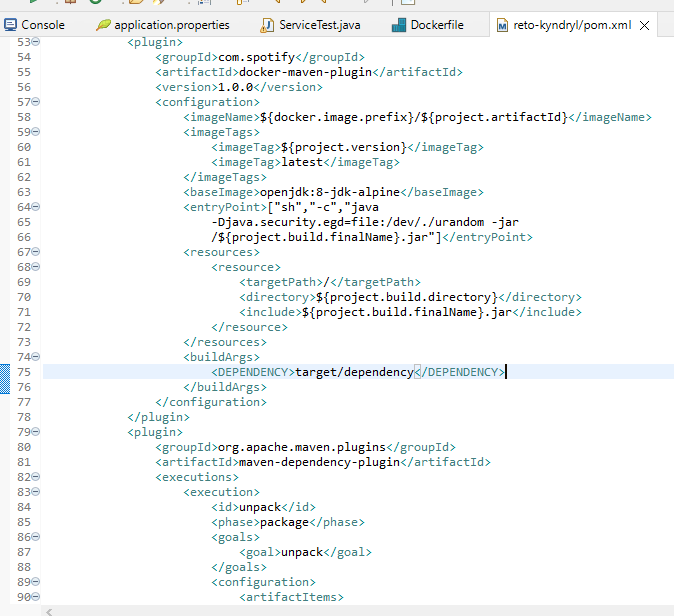


3. Generar un DockerFile, que permita construir el build de una imagen con en backend:

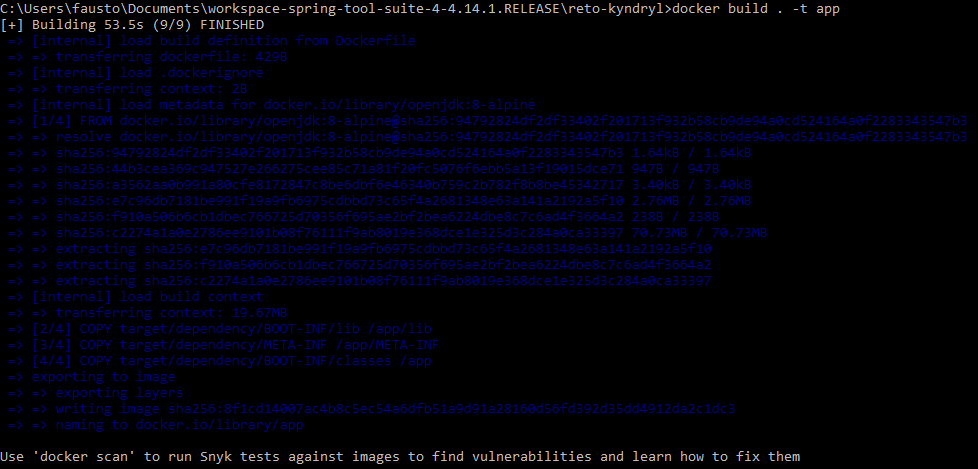
Se presenta el archivo Dockerfile, asi como el applicaction.properties actualizado y el pom.xml.

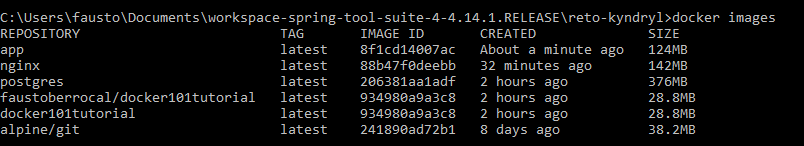


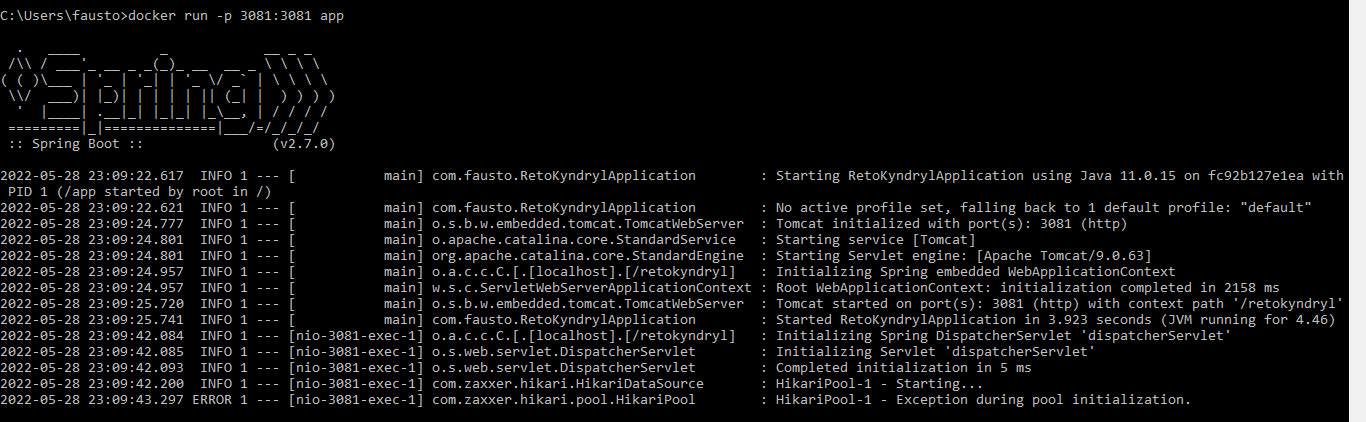




También se presenta la ejecución del built del Dockerfile y la lista de las imagenes creadas y finalmente se presenta la ejecución de la imagen app.

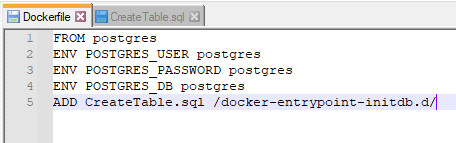


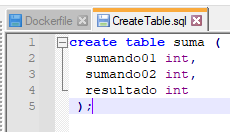




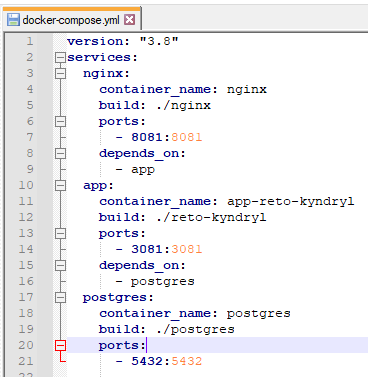
4. Generar DockerFile que permita construir el build de una imagen personalizada con la base de datos:

Se presenta el archivo Dockerfile, asi como el archivo CreateTable.sql usado dentro de este file

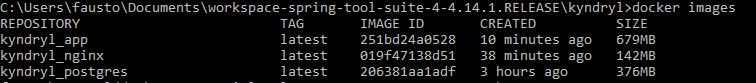




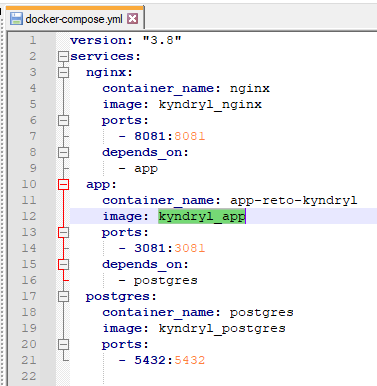
5. Generar un bashero/yml que realice la construcción automática de las imágenes mediante los archivos DockerFile respectivos:Se presenta el archivo docker-compose.yml asi como la ejecución del .YML y finalmente se muestra las imágenes generadas.

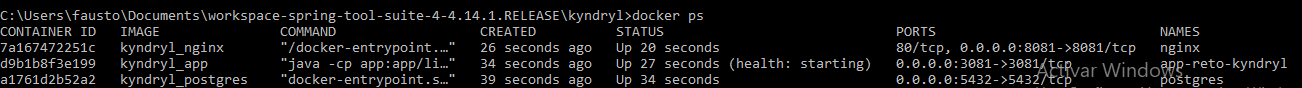
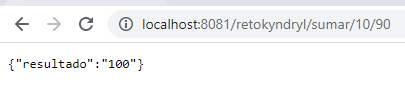


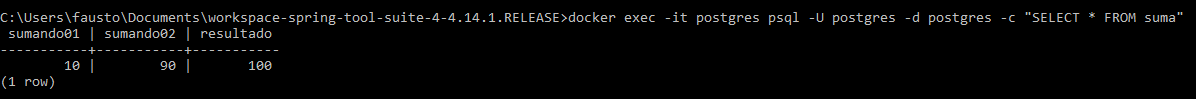




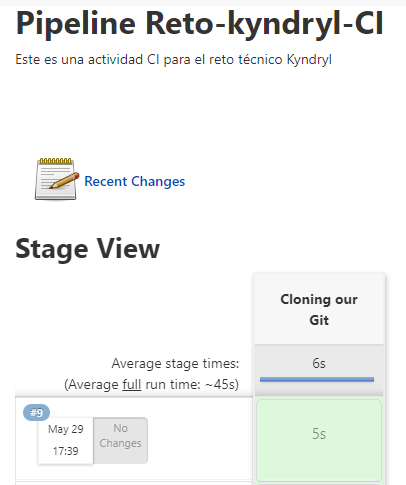
6. Generar un bashero/yml que permita ejecutar las imágenes de Nginx y el Backend conectando a) Nginx ---> Backend ----> Base Datos:Se presenta el archivo docker/compose.yml y el resultadode la ejecución del .YML. Finalmente se muestra una ejecución del back-end.

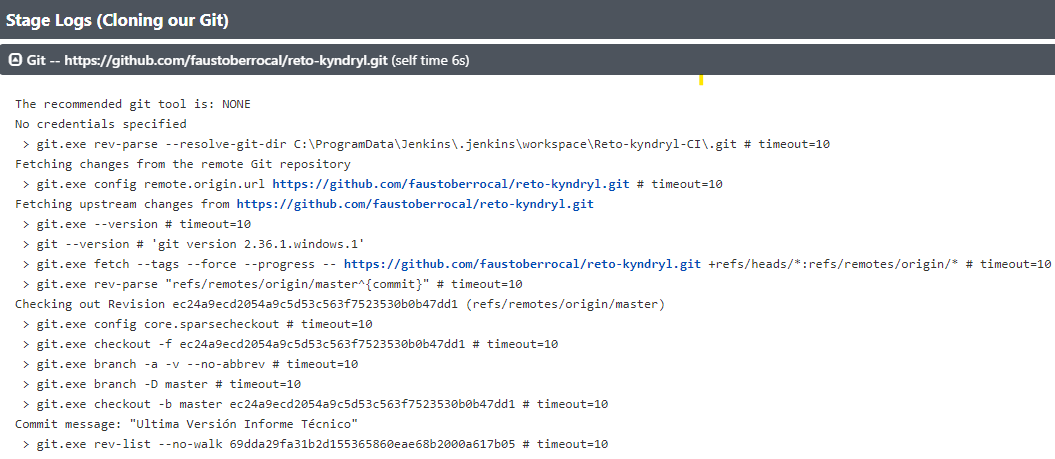




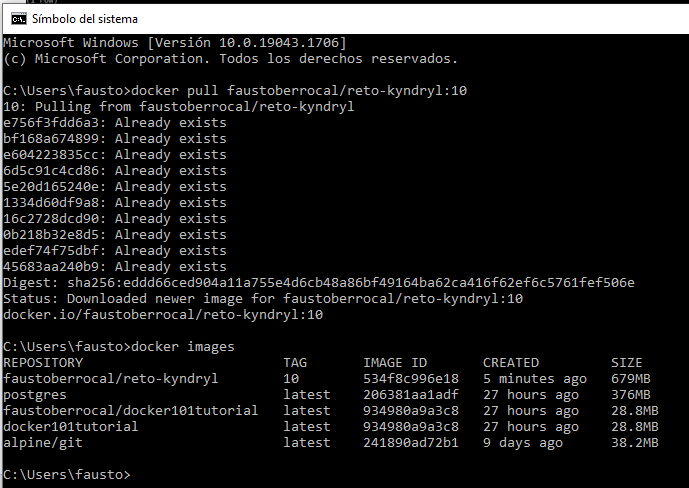


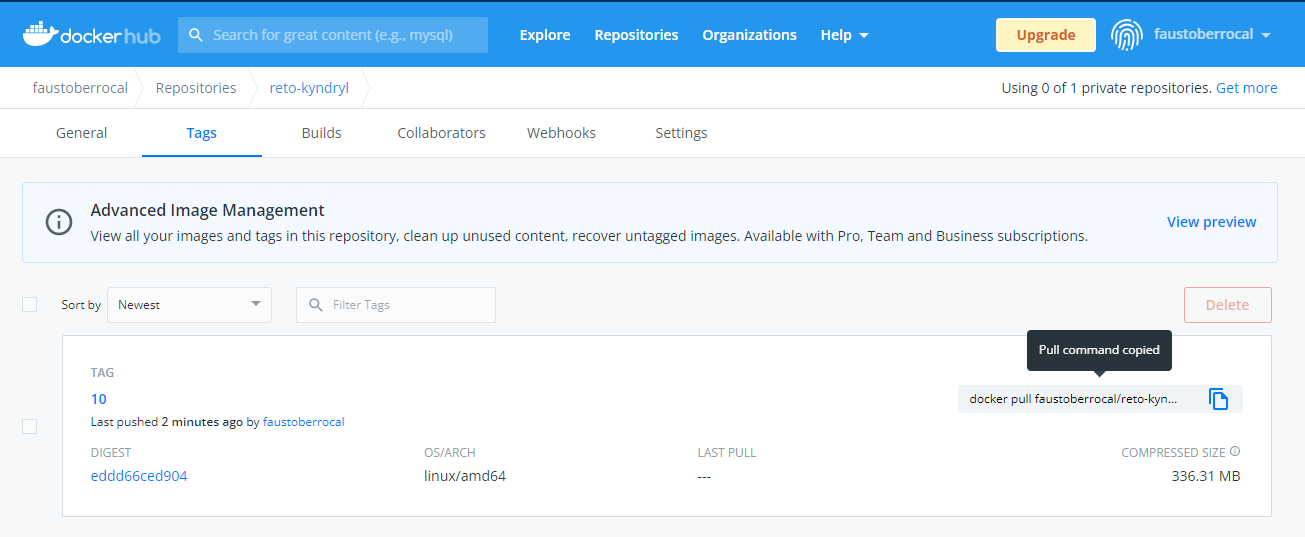
7. Continuous Integration: Realizar el despliegue mediante un toolchain de ALM por ejemplo Github/Gitlab/Bitbucket + Jenkins/Bamboo + Sonarqube: Se generó el Stage solicitado, se muestra el Stage log actualizado:

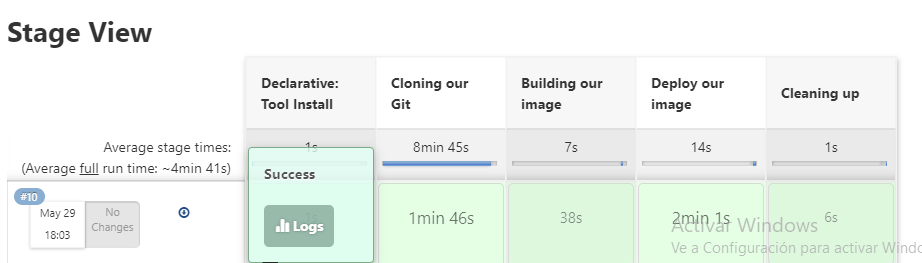




8. Continuous Delivery: Implementar Docker Registry en local o Cloud Provider (AWS, Azure, GCP, Ibm Cloud, Ali Cloud) para las imágenes de docker, Artifactory, Nexus Sonatype: Se pudo realizar el CD logrando realizar el pull de la imagen con el tag: 10.







9. Continuous Deployment : Realizar un Pipeline (Jenkins, vía código) / Plan (vía código \*Bamboo Specs\*) para el deploy automático: Automáticamente se consulta el repositorio de Github cada minuto y lanza un job de Jenkins con la nueva versión.

