# **Explicación Detallada del Flujo CI/CD para Despliegue en WebLogic**

Este documento describe el flujo completo de Integración Continua (CI) y Despliegue Continuo (CD) que hemos diseñado, comenzando por la compilación y empaquetado de tu aplicación "Experto Contable" hasta su despliegue en un servidor WebLogic alojado en Azure.

## **Diagrama Simplificado del Flujo**

1. Disparador (Push/Manual)

|

v

2. Pipeline Orquestador (main-ci-cd-orchestrator.yml)

|

|--- Llama a --> 3. Pipeline de CI (deploy-container-app.yml - el de tu colega)

| | - Compila código Java

| | - Obtiene versión del pom.xml

| | - Crea un archivo .zip (ej. PR-123.zip) conteniendo el .ear

| | - Sube el .zip a JFrog Artifactory

| | - (Importante) Genera Salidas: nombre del .zip, ruta en JFrog, nombre original del .ear

|

|--- Job: Mapeo de Entradas CI a CD (map\_ci\_to\_cd\_inputs)

| | - Traduce el entorno (ej. "integration" de CI a "desarrollo" para CD)

| | - Pasa los nombres de artefactos de CI a CD

|

|--- Llama a --> 4. Pipeline "Llamador" de CD (deploy-experto-contable-cd.yml)

| - Recibe información del artefacto en JFrog del orquestador

| - Descarga el .zip desde JFrog

| - Extrae el archivo .ear del .zip

| - Deriva el nombre de la aplicación para WebLogic (ej. ExpertoContable\_v3.4)

| - Copia el .ear al servidor WebLogic (Azure VM) vía SCP

|

|--- Llama a --> 5. Pipeline Reutilizable de Despliegue WebLogic (weblogic-deploy-reusable.yml)

| - Se conecta al servidor WebLogic (Azure VM) vía SSH

| - Ejecuta `weblogic.Deployer -undeploy` (si aplica)

| - Ejecuta `weblogic.Deployer -deploy` con el .ear

| - Reporta el estado

## **1. Pipeline de CI (Integración Continua - deploy-container-app.yml)**

Este es el pipeline reutilizable que tu colega desarrolló. Su principal responsabilidad es tomar el código fuente, compilarlo y empaquetarlo, dejando un artefacto listo para el despliegue.

* **Propósito Principal**:
  + **Compilar el código Java**: Usando Maven y la versión de Java especificada.
  + **Gestionar Dependencias**: Opcionalmente, se conecta a JFrog Artifactory para descargar dependencias durante la compilación si use-artifactory es true.
  + **Obtener Versión y Nombre del Artefacto**: Lee el pom.xml para determinar el artifactId, version y packaging (ej. ear). Con esto, construye el nombre del artefacto original (ej. ExpertoContable-3.4.ear).
  + **Empaquetar para JFrog**: Comprime el artefacto compilado (ej. ExpertoContable-3.4.ear) dentro de un archivo .zip (ej. PR-123.zip). El nombre de este .zip se basa en el número de ejecución del pipeline o información del Pull Request.
  + **Subir a JFrog**: Sube este archivo .zip a un repositorio específico en JFrog Artifactory, versionándolo según el ambiente (ej. pipeline-int, pipeline-lab).
* **Entradas Clave que Recibe (de un orquestador)**:
  + environment: El entorno para el cual se está compilando (ej. integration, laboratory). Esto influye en el nombre del repositorio en JFrog.
  + build-command: El comando Maven para construir el proyecto (ej. mvn clean package -DskipTests).
  + compile\_folder: La carpeta donde se encuentra el artefacto compilado (ej. target/) después de ejecutar el build-command.
  + pom-version-file: Ruta al pom.xml principal para extraer la versión.
  + use-artifactory: Booleano para indicar si se usa Artifactory para dependencias y subida de artefactos.
* **Salidas (Outputs) Cruciales para el CD**:
  + Para que el CD funcione correctamente, este pipeline de CI **debe** generar salidas (outputs) que el pipeline orquestador pueda usar. Las más importantes son:
    - ci\_original\_artifact\_filename: El nombre del artefacto original que se compiló y está *dentro* del zip (ej. ExpertoContable-3.4.ear).
    - ci\_jfrog\_zip\_name: El nombre del archivo .zip que se subió a JFrog (ej. PR-123.zip).
    - ci\_jfrog\_repo\_path: El nombre/ruta del repositorio en JFrog donde se guardó el .zip (ej. ExpertoContable-int).

## **2. Pipeline Orquestador (main-ci-cd-orchestrator.yml)**

Este es el pipeline principal que inicia todo el proceso. Decide cuándo y cómo ejecutar el CI y luego el CD.

* **Propósito Principal**:
  1. Ser el punto de entrada para el flujo completo (ej. se activa por un push a una rama o manualmente).
  2. Llamar al pipeline de CI, pasándole los parámetros necesarios.
  3. Tomar las salidas del pipeline de CI.
  4. Mapear/transformar estas salidas y el entorno para que sean compatibles con las entradas del pipeline de CD.
  5. Llamar al pipeline de CD.
* **Disparadores (on)**:
  1. Puede ser un push a una rama específica (ej. main).
  2. Puede ser un workflow\_dispatch para ejecución manual, donde se puede especificar el entorno.
* **Jobs (Trabajos)**:
  1. **call\_ci\_build**:
     + Llama al pipeline de CI reutilizable (deploy-container-app.yml).
     + Le pasa las entradas necesarias (entorno, comando de compilación, ruta al pom, etc.).
     + Espera a que el CI termine.
  2. **map\_ci\_to\_cd\_inputs**:
     + **Este job es vital para la conexión entre CI y CD.**
     + Depende de que call\_ci\_build haya terminado (needs: call\_ci\_build).
     + **Toma las outputs generadas por el CI.** Si el CI no las genera explícitamente, este job intentaría re-derivarlas, lo cual es más complejo y propenso a errores. Es mucho mejor que el CI las provea.
     + **Mapea el entorno**: El CI podría usar un entorno llamado integration, mientras que el CD para WebLogic podría llamarlo desarrollo. Este job hace esa "traducción".
     + Genera sus propias outputs que son exactamente las que el pipeline de CD espera.
  3. **call\_cd\_deploy**:
     + Depende de map\_ci\_to\_cd\_inputs.
     + Llama al pipeline "Llamador" de CD (deploy-experto-contable-cd.yml).
     + Le pasa las entradas mapeadas por el job anterior (nombre del ZIP en JFrog, nombre original del .ear, entorno de CD, etc.).
     + Utiliza secrets: inherit para pasar de forma segura las credenciales necesarias.

## **3. Pipeline "Llamador" de CD (deploy-experto-contable-cd.yml)**

Este pipeline es el primer paso del Despliegue Continuo. Se encarga de obtener el artefacto correcto de JFrog y prepararlo para el despliegue en WebLogic.

* **Propósito Principal**:
  1. Ser llamado por el orquestador (workflow\_call).
  2. Descargar el archivo .zip (que contiene el .ear) desde JFrog Artifactory.
  3. Extraer el archivo .ear del .zip.
  4. Derivar el nombre de la aplicación como WebLogic lo espera (ej. de ExpertoContable-3.4.ear a ExpertoContable\_v3.4).
  5. Copiar el archivo .ear extraído al servidor WebLogic (que está en una VM de Azure) usando SCP (Secure Copy Protocol).
  6. Llamar al pipeline reutilizable final que ejecuta los comandos de WebLogic.
* **Entradas Clave que Recibe (del orquestador)**:
  1. jfrog\_zip\_artifact\_name: Nombre del archivo .zip en JFrog (ej. PR-123.zip).
  2. jfrog\_zip\_repository\_path: Repositorio en JFrog donde está el .zip (ej. ExpertoContable-int).
  3. jfrog\_base\_url: URL base de JFrog.
  4. original\_artifact\_filename\_inside\_zip: Nombre del .ear dentro del .zip (ej. ExpertoContable-3.4.ear).
  5. environment: El entorno de despliegue para CD (ej. desarrollo, laboratorio).
* **Secretos que Utiliza**:
  1. Credenciales de JFrog para descargar (JFROG\_USER, JFROG\_PASSWORD).
  2. Credenciales SSH para acceder a la VM de Azure donde corre WebLogic (AZURE\_VM\_IP\_<ENTORNO>, AZURE\_VM\_SSH\_USER, AZURE\_VM\_SSH\_PRIVATE\_KEY).
  3. Contraseñas de WebLogic para el despliegue (WEBLOGIC\_PASSWORD\_<ENTORNO>).
* **Job Principal (prepare\_and\_deploy\_artifact)**:
  1. **Validación de Entorno**: Verifica que las configuraciones para el entorno dado (IPs, contraseñas) existan.
  2. **Descarga del ZIP de JFrog**: Usa curl para bajar el .zip.
  3. **Extracción del Artefacto**: Usa unzip para obtener el .ear del .zip. El CI lo empaqueta con -j (junk paths), así que el .ear está en la raíz del zip.
  4. **Derivación del Nombre para WebLogic**: Un script transforma ExpertoContable-3.4.ear a ExpertoContable\_v3.4. Esto es importante porque el comando weblogic.Deployer -name usa este formato.
  5. **Copia del Artefacto al Servidor WebLogic**: Usa la action appleboy/scp-action para transferir el .ear extraído al directorio /u01/WLS14.1.1/middleware/domains/instaladores/ en la VM de Azure.
  6. **Llamada al Pipeline Reutilizable de Despliegue**: Invoca a weblogic-deploy-reusable.yml, pasándole todos los detalles necesarios (datos de conexión SSH, detalles de WebLogic, nombre de la aplicación derivado, nombre del archivo .ear en el servidor).

## **4. Pipeline Reutilizable de Despliegue WebLogic (weblogic-deploy-reusable.yml)**

Este es el último eslabón de la cadena, encargado de interactuar directamente con WebLogic para realizar el despliegue.

* **Propósito Principal**:
  1. Conectarse de forma segura al servidor WebLogic (en la VM de Azure) vía SSH.
  2. Ejecutar los comandos weblogic.Deployer para desinstalar la versión anterior de la aplicación (si skip\_undeploy no es true).
  3. Ejecutar el comando weblogic.Deployer para instalar (desplegar) la nueva versión del archivo .ear que ya fue copiado al servidor.
* **Entradas Clave que Recibe (del "Llamador" de CD)**:
  1. ssh\_host, ssh\_username: Para la conexión SSH.
  2. weblogic\_admin\_url, weblogic\_username, weblogic\_targets: Para los comandos weblogic.Deployer.
  3. application\_name\_in\_weblogic: El nombre que la aplicación tendrá en WebLogic (ej. ExpertoContable\_v3.4).
  4. artifact\_filename\_on\_server: El nombre del archivo .ear en el servidor (ej. ExpertoContable-3.4.ear).
  5. Otras rutas y opciones como weblogic\_home\_lib\_path, artifact\_source\_directory\_on\_server.
* **Secretos que Utiliza**:
  1. SSH\_PRIVATE\_KEY: Para la conexión SSH.
  2. WEBLOGIC\_PASSWORD: Para los comandos weblogic.Deployer.
* **Mecanismo de Acción**:
  1. Construye los comandos weblogic.Deployer para undeploy y deploy como un script.
  2. Usa la action appleboy/ssh-action para ejecutar este script remotamente en el servidor WebLogic.

## **Flujo de Datos y Entornos**

* El **entorno** se define inicialmente en el orquestador (ej. integration).
* El CI usa este entorno para determinar a qué repositorio de JFrog subir el .zip (ej. ExpertoContable-int).
* El job map\_ci\_to\_cd\_inputs del orquestador traduce este entorno de CI a un entorno de CD (ej. integration -> desarrollo).
* El CD usa el entorno de CD para:
  + Saber de qué repositorio de JFrog descargar (aunque el nombre del ZIP y el repo vienen del CI).
  + Seleccionar las IPs correctas de las VMs de Azure.
  + Seleccionar las contraseñas correctas para WebLogic.
  + Determinar los targets correctos en WebLogic (ej. CLUSTER\_HWEC-D).

Este flujo modular y reutilizable te permite tener un proceso de CI/CD robusto y adaptable para tus aplicaciones WebLogic. La clave está en el correcto paso de información (especialmente nombres de artefactos y rutas) entre los diferentes pipelines a través de las outputs y las inputs.