Informe Técnico: Infraestructura de Despliegue Automático Arquitectura de CI/CD (Integración y Despliegue Continuo) Personalizado con los requerimientos de larvosys.

1. Resumen General

Se ha implementado un entorno de desarrollo y despliegue basado en Docker, que permite a los desarrolladores subir su código mediante Git y verlo desplegado automáticamente en un servidor de aplicaciones WildFly 18, sin intervención manual del administrador,así también se se encuentra funcional postgres 17 con pgadmin 4 para el uso visual menos complicado y el final implementado a sido un servidor sftp para el traslado de archivos de manera segura.

2. Componentes del Ecosistema (Stack)

Componente	Función	Acceso / Puerto
Gitea	Servidor de Git privado para control de versiones.	http://194.5.152.156:3000
WildFly versión 18.0.1.Final	Servidor de aplicaciones Java (Jakarta EE).	http://194.5.152.156:8080
PostgreSQL	Base de Datos relacional para las aplicaciones. Y nombre del servidor postgres_db:5432	Puerto 5432
pgAdmin 4	Gestión visual de la base de datos.	http://194.5.152.156:5050

---

3. Auto-Deploy Hook

La pieza clave es un script de Git Hook (post-receive) configurado dentro de Gitea.

• ¿Cómo funciona?: Cada vez que el desarrollador hace un git push, el script detecta automáticamente si hay un archivo .war.

• Acción: Extrae el contenido binario del commit y lo deposita en la carpeta /home/deployments/ del VPS.

• Sincronización: Esta carpeta está vinculada ("mapeada") directamente con el contenedor de WildFly, logrando un despliegue inmediato.

4. Credenciales y Claves de Acceso

Servicio	URL de Acceso	Usuario	Contraseña
Gitea (Git)	http://194.5.152.156:3000	rodrigo15jul67@hotmail.com Rodrigo	Larvosys-2025GIT
pgAdmin (DB)	http://194.5.152.156:5050	admin@larvosys.com	Larvosys-2025PG
WildFly Console	http://194.5.152.156:9990	Admin richar	Larvosys-2025WF larvosysdev@
Postgres	postgres_db	postgres	Larvosys-2025DB
Sftp	194.5.152.156	richar	larvosysdev@

5. Rutas Críticas en el VPS

Configuración de WildFly: /home/wildfly_data/standalone/configuration/ (Aquí reside el archivo standalone.xml donde se gestionan los DataSources y el acceso al servidor).

Módulos y Drivers Externos: /home/wildfly_data/modules/ (Ruta crítica donde se instalan el driver de PostgreSQL y las librerías propias de la empresa, como ec.com.rsoft.*).

Carpeta de Despliegue Automático: /home/deployments/ (Punto de enlace donde el Git Hook deposita los archivos .war para su publicación inmediata).

Persistencia de Base de Datos: /root/entorno-dev/postgres_data/ (Asegura que toda la información de PostgreSQL se mantenga a salvo en el disco duro del VPS).

Repositorios de Código: /root/entorno-dev/gitea_data/ (Almacenamiento seguro de todos los proyectos y el historial de versiones de Gitea).

6. Instrucciones para el Desarrollador

Para que el sistema funcione, el desarrollador debe seguir estas reglas:

1. Crear Repositorio: Siempre usar la Plantilla (admin/plantilla-java) en Gitea para que el Hook se active.

2. Subir el archivo: El archivo .war debe ir en la raíz del repositorio o dentro del proyecto.

3. URL de la App: Si el archivo se llama sistema.war, la aplicación será accesible en http://194.5.152.156:8080/sistema/.

Ventajas de solución propuesta

• Automatización: El desarrollador no pierde tiempo subiendo archivos por FTP o SSH; solo hace git push.

• Aislamiento (Docker): Si algo falla en WildFly, no daña el resto del servidor. Es fácil de reiniciar y recuperar.

• Independencia: el servidor de Git propio (Gitea), ellos son dueños de su código y su infraestructura, sin depender de servicios externos costosos.

7. Interfaz Gráfica de Administración (Escritorio Remoto)

Se ha habilitado un entorno de escritorio (GUI) accesible vía RDP (Escritorio Remoto de Windows). Esto permite realizar tareas administrativas de archivos y navegación interna de forma visual, facilitando la curva de aprendizaje para el personal no técnico.

He optado por XFCE porque es un entorno de alto rendimiento que no sacrifica la potencia del servidor. Esto les da una interfaz amigable para gestionar sus archivos sin lentitud, manteniendo toda la capacidad del VPS para las aplicaciones Java.

Conclusión

El proyecto se entregó bajo una arquitectura de micro-servicios mediante contenedores. Esto significa que en un solo VPS robusto tienen la potencia de lo que antes requeriría 5 máquinas. He automatizado el flujo de trabajo para que el despliegue sea instantáneo vía Git, reduciendo el margen de error humano y eliminando la necesidad de un administrador de sistemas dedicado para subir archivos cada vez.

Recomendaciones

Actualmente el acceso es por IP y puertos. Se recomienda la implementación de un Proxy Inverso con SSL para que el acceso sea mediante un dominio (larvosys.com) y cuente con cifrado de seguridad HTTPS, lo cual protege las credenciales y el código fuente".

Echor por Ing.Adair Villarreal