Certified Tech Developer
The Ultimate Degree

Infraestructura II

Artefactos

El artefacto es un objeto o serie de objetos que se produce a partir de estas compilaciones. Estos resultantes pueden, a su vez, utilizarse para una segunda y/o última compilación. Tenemos como ejemplo de ello archivos binarios: dll, jar, war, ear, msi, archivos exe, etc.

¡Naveguemos entre artefactos para comprender qué son y cómo se administran!

Navegando entre artefactos

Las aplicaciones suelen estar fragmentadas en distintos componentes individuales que luego se pueden ensamblar hacia un producto completo. Esto generalmente sucede durante la fase de compilación, cuando son creados fragmentos más pequeños de la misma. Esto suele hacerse para tener un uso más eficiente de los recursos, reducir los tiempos de compilación, hacer un mejor seguimiento para la depuración binaria, etc.

El tipo de artefacto o artifact va a depender del producto con el que se programe la aplicación.

Veamos el tipo de artifact por producto:

- Java: jar, ear, war, etc.
- .Net: dll, exe, etc.
- Python: .py, sdist.

También pueden existir tipos de artefactos no relacionados con el producto específico, pero necesarios para el funcionamiento. Por ejemplo archivos, YAML, XML, TXT, MD, lib, so, bin, etc.

¡Un momento! Para complejizar un poco más el panorama, un artefacto también puede ser un script, un diagrama, un modelo de datos, etc.

Pero entonces, ¿un artifact es cualquier tipo de archivo? En términos estrictos, **es todo aquello que resulta de un proceso de build.**

Administración

Un producto de repositorios de artefactos nos permite básicamente efectuar las siguientes operaciones:

- mover
- copiar
- eliminar

... artefactos para mantener la consistencia en cada repo.

Cuando un artefacto se mueve, copia o elimina, el producto debe actualizar automáticamente los llamados “descriptores de metadatos”.

Ejemplo de ello pueden ser: maven-metadata.xml, RubyGems, Npm, etc.

Metadata

¡Los metadatos son datos acerca de los datos! Cada artefacto contendrá metadatos que son esenciales para la reutilización de código.

Una característica existente es la de permitir a los desarrolladores compartir código y utilizar componentes de terceros. En este sentido, los metadatos cumplen un rol clave en la colaboración. **¿Por qué?**

Por ejemplo, al chequear los datos asociados a cada artefacto podemos ver si fueron alterados. Esto será de utilidad para validar su descripción, por ejemplo, la versión de un producto. Podremos saber -acerca de ese cambio- quien lo hizo, cuándo, a qué hora exactamente, qué dependencias tiene, etc.

Sin embargo, la combinación de muchos tipos de metadatos puede llegar a complejizar el proceso de colaboración. ¡Una buena estrategia inicial es vital para evitar caer en este tipo de situaciones!

Almacenamiento

¿Dónde se almacena todo este código y sus respectivos artefactos?

En este punto está claro que el código suele almacenarse en sistemas de control de versiones como GitHub, GitLab o BitBucket.

Pero este no es el caso para los artefactos que pueden -por ejemplo- ser archivos binarios. Y quizás no tiene mucho sentido utilizar un repositorio de proyectos para almacenar archivos binarios que son ilegibles para el ser humano y pueden ser bastante grandes en tamaño.

Es aquí donde los **repositorios de binarios** son una parte tan vital del proceso de integración continua.

El repositorio binario puede permitir alojar todo esto en un solo lugar haciendo que su administración sea más simple. Ejemplo de productos típicos que podemos encontrar en el mercado hoy día son: Artifactory, Nexus, Harbor, etc.

También vamos a encontrar aquellos que son basados en Cloud: Azure Artifact, Artifact Registry, etc.

Accediendo a nuestros artefactos

Como todo producto de software, su sintaxis variará pero su propiedad será la misma: ofrecer una forma sencilla de realizar consultas que especifique un criterio de búsqueda, filtros, opciones de clasificación y parámetros de salida.

Esto se logra mediante la exposición de una API RESTful, ya que siendo objetos que deban utilizarse de inmediato para proporcionar datos de salida, el tiempo de acceso y respuesta debe ser extremadamente rápido y con bajo consumo de memoria.

Todo alrededor de una filosofía

La importancia de un repo de artefactos se puede entender en relación con la filosofía DevOps: "desarrollar mejores aplicaciones, más rápido y con constantes entregas de funciones o productos de software".

Una práctica común en integración continua es la de construir el binario una sola vez, subirlo a un repositorio de binarios y luego llamarlo desde allí para implementarlo en los diferentes entornos. De esa manera nos aseguramos de que el código base que ya funciona en el ambiente de desarrollo es la misma base que se introdujo en producción.

Conclusión

Los artefactos son subproductos de un proceso de desarrollo de software. Son los **componentes** con los que un software está hecho.

Son sumamente necesarios durante el desarrollo, las operaciones diarias, el mantenimiento y la actualización del software.

Su correcta administración es fundamental para el correcto funcionamiento de un producto gestionado mediante una filosofía de DevOps.