Escuela de Ingeniería Informática
Universidad de Oviedo

Package Management

Autores:
Donato A. Martín – UO288787
Manuel de la Uz González – UO288991
Alejandro García – UO283076

¿Qué es un paquete?

Un paquete es un conjunto organizado de archivos y directorios que agrupa código fuente, metadatos, documentación y recursos de toda clase relacionados entre sí con una funcionalidad específica a resolver en cualquier lenguaje o entorno.

¿Por qué usar paquetes?

Los paquetes nos aportan múltiples beneficios a la hora de realizar código de calidad eficiente y mantenible. Los principales beneficios que nos aporta la utilización de paquetes son los siguientes:

Reutilización de código: nos permiten compartir funcionalidad entre diferentes partes del proyecto que estamos desarrollando

Identificación clara: Cada paquete tiene un nombre y una versión específica. Esto permite identificar y referenciar de manera precisa a un paquete en particular.

Control de versiones: Puedes especificar la versión de un paquete que deseas utilizar. Por ejemplo, si un paquete tiene las versiones 2.1, 2.2 y 2.3, pero no estás listo para actualizar a la 2.3, puedes seguir utilizando la 2.1.

Gestión de dependencias: simplifica la manera con la que tenemos que lidiar con las dependencias y garantiza la retrocompatibilidad

Gestión de paquetes

Gestores de paquetes

Los gestores de paquetes, en esencia, son herramientas de automatización de paquetes. Evitan que el desarrollador tenga que lidiar con la gestión de todos los paquetes que usa su proyecto.

Una de las principales ventajas de los gestores de paquetes es la uniformidad en su funcionamiento, dada una configuración siempre producirán e instalarán la misma lista de paquetes, dándonos un ambiente estandarizado.

Se suele buscar que el gestor este escrito en el mismo idioma que el lenguaje al que apoya, ya que la idea es que donde funcione el lenguaje debería funcionar su gestor de paquetes. También es interesante que el gestor no dependa de sí mismo, es decir, que no necesite paquetes ya que si necesitas un paquete para compilar el gestor de paquetes tienes un problema (Hacer ***BootStrapping*** de un gestor de paquetes no es una experiencia recomendable)

BootStrapping: *proceso de iniciar o arrancar un sistema o programa con la ayuda de un conjunto mínimo de instrucciones o recursos.*

Repositorios de paquetes

Vale, tenemos un gestor de paquetes, pero ¿De dónde saca los paquetes? La respuesta es que los va a buscar a repositorios de paquetes y según donde busca podemos clasificar a los gestores de paquetes en cuatro tipos

Monorepo

El gestor de paquetes va a buscar los paquetes a un repositorio central el cual indexa todos los paquetes a los que se puede acceder desde el gestor, un ejemplo de esto es NPM

Multirepo

Similar al Mono Repo, pero en alguna parte de la configuración puedes especificar una lista de repositorios en los cuales buscar los paquetes. Esto conlleva una dificultar añadida ¿Qué pasa si el mismo paquete está en varios repos?

Blockchain

También conocidos como gestores distribuidos, se basan en tecnologías como blockchain para evitar tener uno o varios repositorios centrales. Hoy en día son más bien una demo que algo listo para producción.

URL

Se basan en dar una URL a cada paquete y que el DNS sea el que haga el trabajo de búsqueda, el gestor se limita a descargar el paquete que haya en la URL dada. Un ejemplo es el plugin de JitPack para Maven el cual permite usar URLs de github y gitlab como paquetes

También está la cuestión de donde están estos repositorios, ya que mientras que para la mayoría de los desarrolladores los repositorios públicos son perfectamente válidos, para empresas grandes estos repositorios públicos pueden ser peligrosos o incompletos, por lo que se lanzan a la opción de hostear su propio repositorio de paquetes

Estructura

- **Sistema de nombres:** Permite identificar de forma única a los paquetes y sus versiones dentro de un contexto más amplio.
- **Algoritmo de selección de versiones:** Determina la versión que cada paquete debe utilizar para satisfacer las restricciones de dependencia de un proyecto.
- **Mecanismo de resolución de nombres:** Localiza y descarga los paquetes desde algún repositorio o fuente externa.
- **Mecanismo de instalación:** Coloca los paquetes en el sistema de archivos o en memoria para que puedan ser usados en el proyecto

Funcionamiento

En cuanto al funcionamiento de los gestores de paquetes, primero tenemos los paquetes, un conjunto de código que puedes utilizar en tu programa para no escribir todo desde cero. Cada paquete tiene un nombre y una versión específica, lo que nos permite identificar y referenciar de manera precisa a un paquete en particular. Además de esto los paquetes contienen metadatos que incluyen el nombre del paquete, la versión y una lista de otros paquetes requeridos por ese paquete (dependencias). Por último, podemos especificar la versión de un paquete que deseamos utilizar. Por ejemplo, si un paquete tiene las versiones 2.1, 2.2 y 2.3, pero no estás listo para actualizar a la 2.3, podeos seguir utilizando la 2.1.

Dependency Hell

¿Como se representan las dependencias?

Normalmente se representan las dependencias como un grafo, en el cual hay ciertas reglas para asegurarnos de que el gestor de paquetes funciona correctamente (Estas reglas dependen del gestor específico).
Una nota impórtate es que, aunque las dependencias circulares sean un problema grave, muchos gestores las permiten dado que un desarrollador de paquetes no puede saber a priori si su paquete está creando una dependencia circular o no.

El infierno

Imaginemos que tenemos dos paquetes A y B y que ambos dependen del paquete C. Pero A quiere la versión 1 y B la versión 2, esto obviamente es un problema.
La solución más rápida seria instalar ambas versiones de C y darle a A y B lo que quieren (Opción usada por algunos gestores), pero ahora tenemos variables y espacios de nombres repetidos y el código probablemente ni compile.
Otra solución, es buscar una combinación de versiones de A B y C que sean compatibles (Este proceso es computacionalmente caro y no se nos garantiza que funcione), pero no se nos garantiza un paquete C compatible, además, aunque lo halla es seguro es que al actualizar C rompas D y luego al arreglar D descubras que A y E tenían dependencias con D y se rompan, y todo esto sin considerar dependencias circulares, parece que hay una versión de A y B si A depende de C que depende de B que depende de A.