Veille PNPM

Avant de présenter qu’est-ce qu’NPM et c’est quoi un package manager?

https://developer.mozilla.org/en-US/docs/Learn/Tools_and_testing/Understanding_client-side_tools/Package_management

Un package manager c’est un software qui permet d’installer des packages, c’est-à-dire des bouts de code plus ou moins gros (un framework entier ou bien juste quelques fonctions), écrits et partagés par d’autres développeurs.

Pendant le développement d’un projet, un des buts est de ne pas avoir besoin de réinventer la roue car peu importe ce que vous développez, quelqu’un l’a fait, et mieux que vous.

Super! Maintenant comment est-ce que je peux récupérer ce code pour l’utiliser? Et bien à l’aide d’un gestionnaire de paquets (package manager).

Le package manager va vous permettre plusieurs choses:

- récupérer le code que d’autres partages sans vous soucier de comment paramétrer votre projet pour que le code fonctionne.
- installer les dépendances que les packages que vous installez utilisent.
- supprimer proprement les dépendances installés à votre place.
- vérifier la présence de vulnérabilité

NPM

https://www.npmjs.com/

NPM est le package manager le plus connu et utilisé au monde pour le code lié à Javascript.

Il s'appuie sur la communauté open source pour la publication et la maintenance des packages.

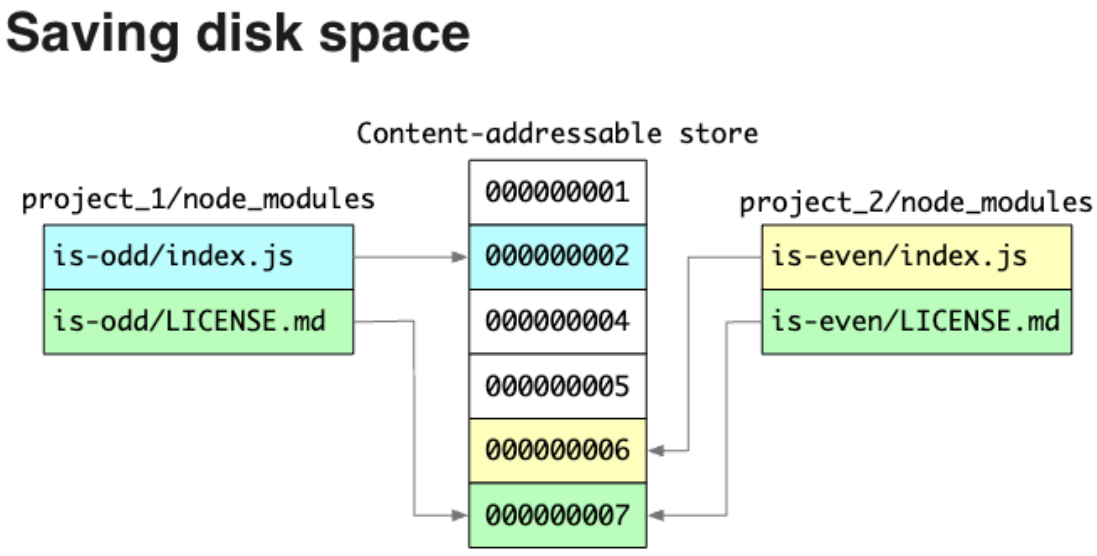
NPM est composé des trois ressources principales:

- Le site NPM pour découvrir des packages, mettre en place des organizations et autres
- Le CLI (command line interface) pour interagir avec NPM depuis votre terminal (ex: npm run dev, npm install <package>...)
- Le registery qui contient tous les packages disponibles

Pourquoi PNPM au lieu de NPM ou Yarn?

https://pnpm.io/fr/

- Lorsqu'on install un package sur un projet avec NPM, il le télécharge et l'installe, peu importe si on l'a déjà installé sur un autre projet.



- Avec PNPM, quand on veut installer un package, une vérification est faite pour voir si on l'a déjà installé et si c'est le cas, un lien symbolique est créé vers l'installation déjà faite. En fait, le dossier node_modules de votre projet est lié au dossier du package contenant dans le dossier PNPM qui contient l'ensemble des package.
    - Par ailleurs si une nouvelle version d'un dépendance apparaît, au lieu de mettre à jour les 100 projets qui utilisent la dépendance, vous allez juste mettre à jour l'unique dépendance installé sur votre machine.
    - Si la mise à jour ne touche qu'à 1 fichier, seul ce fichier va être ajouté au lieu de réinstaller toute la dépendance.

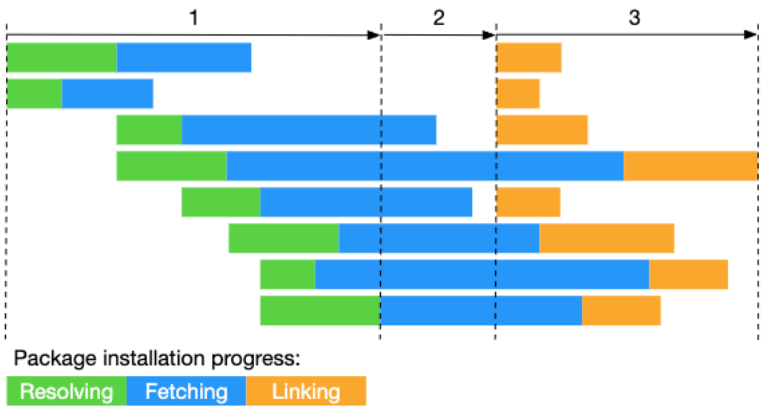
Différence entre le processus d'installation PNPM et NPM

On voit bien que le processus d'NPM est bloquant:

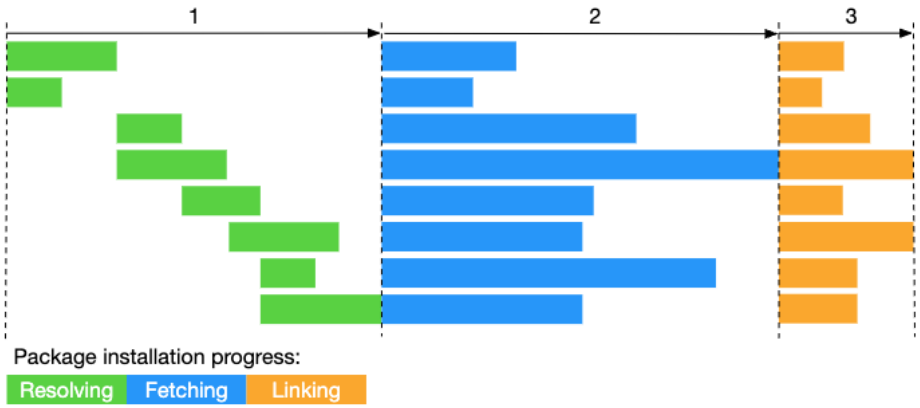
- Avec PNPM quand une dépendance est trouvé, elle est directement récupérée
- Avec NPM, on doit attendre que tous les dépendances soient résolues puis toutes récupérées avant de les écrire dans le dossier node_modules.

C'est cette structure bloquante qui rend NPM plus lent.

1. Dependency resolution. All required dependencies are identified and fetched to the store.
2. Directory structure calculation. The `node_modules` directory structure is calculated based on the dependencies.
3. Linking dependencies. All remaining dependencies are fetched and hard linked from the store to `node_modules`.



This approach is significantly faster than the traditional three-stage installation process of resolving, fetching, and writing all dependencies to `node_modules`.



- Puisqu'on ne télécharge et installe pas à nouveau le package, un gain de temps est réalisé, ce qui rend PNPM plus rapide qu'NPM.
- Et puisqu'on installe pas à nouveau le package, on sauvegarde de l'espace disque sur notre machine.
- PNPM prend en charge les mono repos. Un mono repo est un base de code regroupant plusieurs sous projets au lieu de diviser chaque projet dans un repository séparé. PNPM crée un référentiel des packages qui est partagé entre les différents projets du monorepo

- PNPM utilise un mécanisme de résolution de dépendances qui évite les conflits de versions en installant les versions requises des paquets au niveau de la racine du système de fichiers. Cela permet de résoudre les conflits potentiels et de garantir la cohérence des dépendances entre les projets.
- Défaut: PNPM n’est pas compatible avec tous les environnements de travail qui nécessiteraient par exemple des scripts avec des commandes seulement possible avec NPM.
- La communauté autour de PNPM plus petite que celle autour de NPM, ce qui rend sa maintenance des packages plus longue.

- Beaucoup de devs adoptent PNPM car il est efficace et certaines grands boîtes sponsorisent PNPM comme:
  - Figma
  - Discord
  - Jet brains
  - Prisma
  - Vercel

  La comparaison des fonctionnalités entre PNPM, Yarn (package manager de Facebook) et NPM.

Fonctionnalité	pnpm	Yarn	npm
Support des workspaces	✓	✓	✓
Isolation des `node_modules`	✓ - Par défault	✓	✓
Mode traditionnel pour les `node_modules`	✓	✓	✓ - Par défault
Installation automatique des `peer dependencies`	✓	✗	✓
Plug'n'Play	✓	✓ - Par défault	✗
Zero-Installs	✗	✓	✗
Correction des dépendances	✓	✓	✗
Gestion des versions de Node.js	✓	✗	✗
Possède un lockfile	✓ - `pnpm-lock.yaml`	✓ - `yarn.lock`	✓ - `package-lock.json`
Support des surcharges	✓	✓ - Via les résolutions	✓
Mémoire d'adressage globale	✓	✗	✗
Exécution dynamique des paquets	✓ - Via `pnpm dlx`	✓ - Via `yarn dlx`	✓ - Via `npx`
Cache d'effets secondaires	✓	✗	✗
Affichage des licenses	✓ - Via `pnpm licenses list`	✓ - Via un plugin	✗

Est-ce que le flag -g d'NPM utilise une adresse mémoire globale comme PNPM?

Non, la commande `npm install -g` n'utilise pas une adresse mémoire globale comme PNPM. Lorsque vous exécutez `npm install -g`, cela installe un package de manière globale sur votre système, mais il stocke généralement ces packages dans un répertoire spécifique dédié aux packages globaux. Ces

packages ne sont pas partagés entre les projets de la même manière que PNPM le fait avec son approche de stockage de paquets partagés.

Lorsque vous installez un package globalement avec `npm install -g`, les exécutables fournis par ce package sont souvent ajoutés au chemin d'accès de votre système, ce qui vous permet de les exécuter depuis n'importe quel répertoire sur votre machine. Cependant, chaque projet utilise encore ses propres dépendances installées localement dans son répertoire de travail.

En revanche, PNPM utilise des liens symboliques pour partager les packages entre les projets, ce qui permet d'économiser de l'espace disque et de réduire le temps d'installation en évitant de télécharger et de stocker les paquets plusieurs fois.