Verdaccio

Alvin Berthelot

Version 1.0.0

Ce(tte) œuvre est mise à disposition selon les termes de la Licence Creative Commons.

Attribution - Partage dans les Mêmes Conditions 3.0 non transposé.

La licence, ses explications ainsi que les moyens de contribution et réappropriation sont détaillés à la fin.

Petit rappel npm

npm le compagnon associé de Node.js

Une fois Node.js installé, celui-ci ne vient pas seul, il inclut par défaut npm qui est son **gestionnaire de modules**.

```
npm -v
```

Ce gestionnaire de modules permet d'installer et de publier facilement des modules (signifiant "package" en JavaScript) pour Node.js : serveurs HTTP, frameworks, drivers, processeurs, etc.

npm un gestionnaire ET un dépôt de modules

npm est à la fois lui même un module dédié pour Node.js (servant de gestionnaire) et le registre de modules accessible via Internet.

Sa popularité et son fonctionnement avec Node.js en ont fait le registre pour tout l'écosystème JavaScript désormais.

Installer un module via npm

Dans une invite de commandes, l'instruction `npm install` permet d'installer un module. Il existe 2 manières d'installer un module, soit de manière locale au projet (option par défaut), soit de manière globale (avec l'option `-g`).

```
// local
npm install <module_name>

// global
npm install -g <module_name>
```

Dans le premier cas, le module sera installé depuis le répertoire courant où l'instruction a été lancée. Il ne sera donc accessible que depuis ce répertoire courant.

Dans le deuxième cas, le module sera installé dans un répertoire commun lié à l'installation de Node. Il sera donc accessible depuis n'importe quel répertoire de l'environnement.

Installer tous les modules

Lors d'installations locales de modules, npm va rapatrier ceux-ci dans un répertoire `node_modules`.

Pour installer au pré-requis tous les modules nécessaires au bon fonctionnement d'un projet, npm va s'appuyer sur le fichier `package.json` qui liste les dépendances (et leurs versions associées) avec `dependencies`, `devDependencies`, `optionnalDependencies` et `peerDependencies`.

Il suffit ensuite d'executer `npm install` dans le répertoire où se situe le fichier `package.json` pour installer toutes les dépendances listées dans le répertoire `node_modules`.

```
npm install
```

Un exemple de `package.json`

```
{
  "name": "colourfull-api",
  "version": "1.0.5",
  "description": "Web API to deliver palettes of colours.",
  "dependencies": {
    "body-parser": "1.5.2",
    "cors": "2.8.1",
    "express": "4.2.0"
  },
  "devDependencies": {
    "expect.js": "0.3.1",
    "mocha": "1.21.0",
    "superagent": "0.18.2"
  },
  "scripts": {
    "start": "node index.js",
    "test": "mocha --timeout 15000 test/"
  },
  "repository": {
    "type": "git",
    "url": "https://github.com/my-company/colourfull-api"
  },
  "license": "MIT license"
}
```

npm oui mais ...

Limites du fonctionnement

Avec npm la récupération des dépendances se fait selon un **fonctionnement standard, reproductible, maintenable et évolutif**.

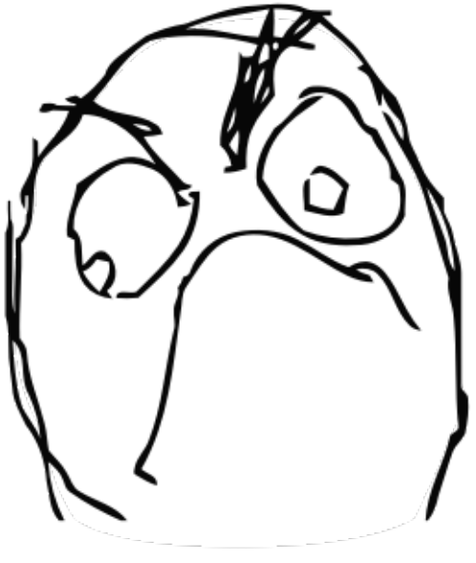
Il a toutefois quelques limites sur le dépôt npm :

- Il n'est reproductible que **si le dépôt npm est opérationnel** pour fournir les dépendances.
- Ce fonctionnement ne permet pas l'ajout de dépendances JavaScript à titre privé (excepté avec un **compte payant**).

En bref, vous êtes dépendant du dépot de dépendances.

Alternative #1

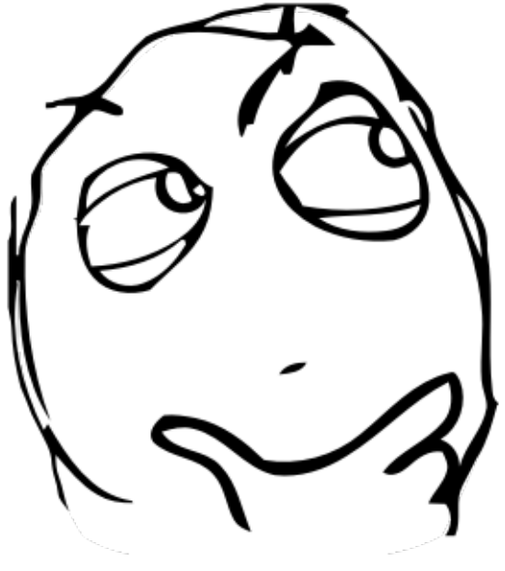
Une première approche pourraît être d'incorporer les dépendances comme des sources dans le gestionnaire de version du projet.



La distinction entre sources et dépendances est primordiale. Elles n'ont pas les mêmes cycles de vie et les mêmes portées.

Alternative #2

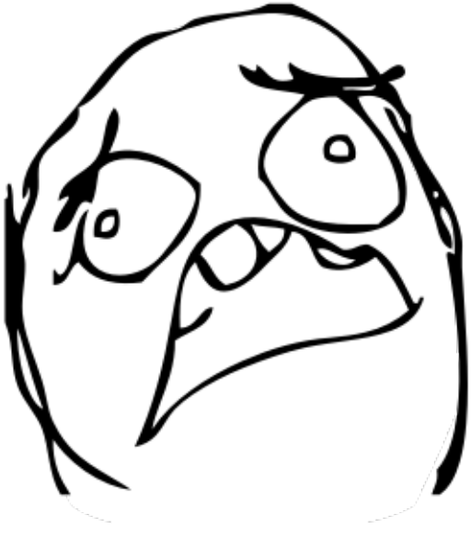
Une deuxième approche pourraît être d'incorporer chaque dépendance comme un projet à part entière dans un gestionnaire de version.



Toujours un problème concernant les dépendances publiques, dans le sens où elles peuvent ne plus être disponibles à un instant T.

Alternative #3

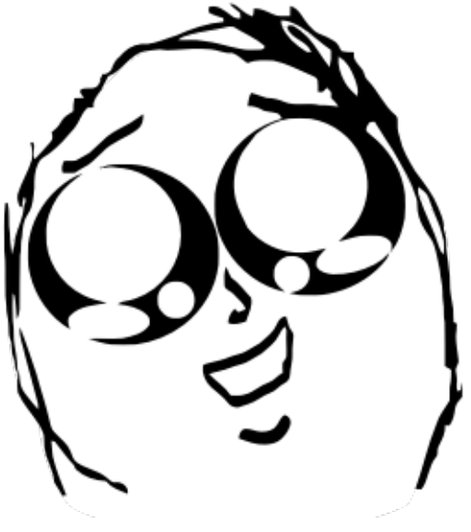
Une troisième approche pourraît être de créer sa propre instance de npm sur un serveur dédié.



C'est trop complexe et couteux pour notre besoin.

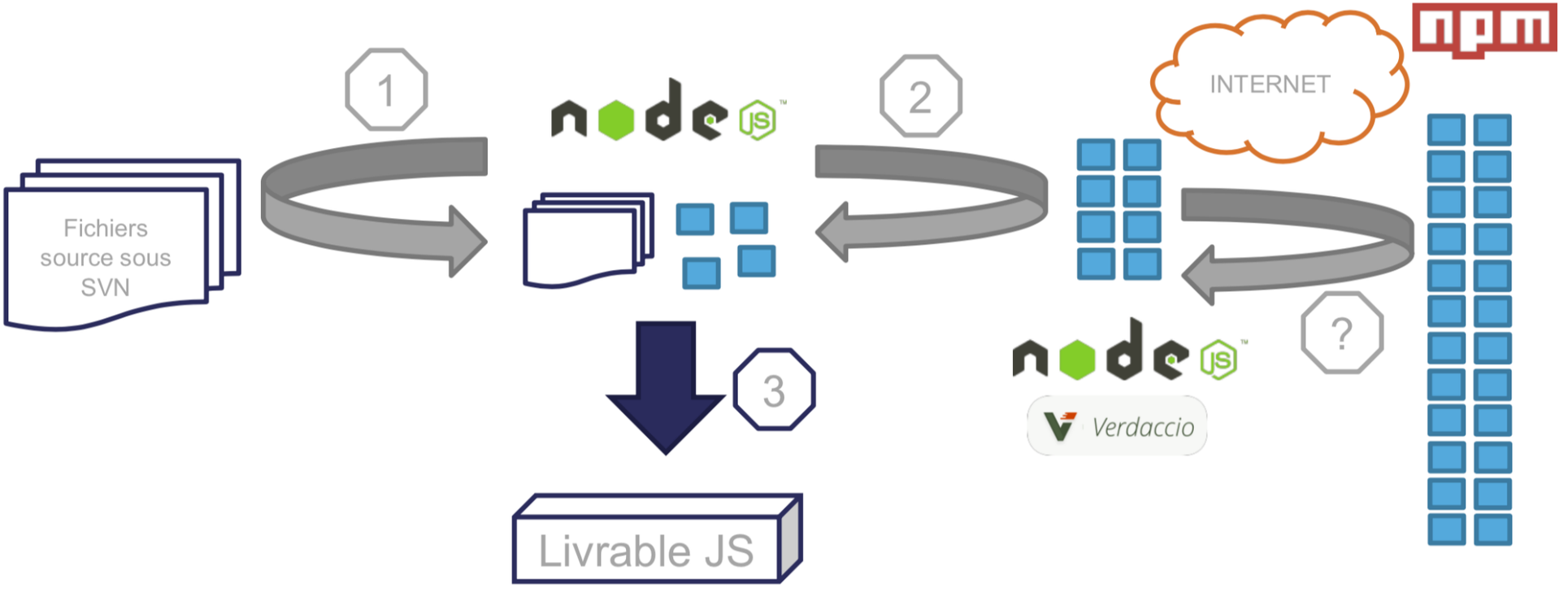
Alternative #4

Une quatrième approche serait d'utiliser un proxy / cache qui interroge lui même le dépôt public npm.



Verdaccio

Fonctionnement de Verdaccio



Installation de Verdaccio

Solution 1 : Installer Verdaccio en globale sur un serveur où Node.js est installé.

```
npm install -g verdaccio
```

Solution 2 : Utiliser Docker avec une image Verdaccio

```
docker pull verdaccio/verdaccio
```

Les histoires de proxy finissent toujours bien en général …

Configuration du dépôt de dépendances

Monter un dépôt de dépendances c'est bien, encore faut-il que les ressources qui en ont besoin pointent sur ce dépôt.

```
npm config ls

npm set registry http://localhost:4873

npm config ls
```

Par ressources on entend les **développeurs** mais il faut aussi penser aux **environnements d'intégration continue**.

Récupération des dépendances publiques

C'est là qu'une partie de la magie s'opère, c'est absolument transparent !

```
npm install
```

Que les dépendances soient **publiques** ou **privées**.

Bon pour les privées, encore faut-il **les avoir publiées** (sur le bon dépôt tant qu'à faire).

```
npm publish --registry http://localhost:4873
```

Cela nécessite l'enregistrement d'un utilisateur au préalable.

Les minutes Bonaldi

Bilan

L'intérêt d'utiliser Verdaccio peut-être multiple :

- Se couper (partiellement) de la dépendance du dépôt npm
- Publier facilement des dépendances privées
- Corriger rapidement une dépendance publique
- Servir de cache++ lorsqu'on développe en offline

Ce que j'aime, c'est que c'est **simple** à comprendre, à mettre en place et à utiliser.

L'adhérence à Verdaccio est quasi inexistante, sur les dépendances c'est une autre histoire ;)

Licence

Pour toutes questions, réclamations ou remarques, merci d'envoyer un message à alvin.berthelot@webyousoon.com.

Explications licence CC BY-SA 3.0

Cette licence permet aux autres de remixer, arranger, et adapter votre œuvre, même à des fins commerciales, tant qu'on vous accorde le mérite en citant votre nom et qu'on diffuse les nouvelles créations selon des conditions identiques.

Cette licence est souvent comparée aux licences de logiciels libres, "open source" ou "copyleft".

Toutes les nouvelles œuvres basées sur les vôtres auront la même licence, et toute œuvre dérivée pourra être utilisée même à des fins commerciales.

C'est la licence utilisée par Wikipédia ; elle est recommandée pour des œuvres qui pourraient bénéficier de l'incorporation de contenu depuis Wikipédia et d'autres projets sous licence similaire.

Contribution et réappropriation

Ce fichier PDF est généré avec Asciidoctor à partir d'un dépôt Git se trouvant sous GitHub.

https://github.com/alvinberthelot/slides-node-js

Cela signifie que vous n'avez pas besoin de vous battre avec un fichier binaire (le PDF) pour **contribuer**, **vous réapproprier le contenu** ou **modifier le thème** de présentation.

Contribution

Vous voulez **contribuer au contenu** car :

- Il y a une erreur (ça arrive à tout le monde), de typographie, de compréhension, ou tout autre chose.
- Vous souhaitez apporter une précision.

Il vous suffit de contribuer au projet via Git par le moyen d'une "pull request" sur le dépôt Git.

Réappropriation

N'oubliez pas les conditions de la licence.

Vous voulez vous **réapproprier le contenu** car :

- Vous souhaitez donner un style différent.
- Vous souhaitez enlever/ajouter/modifier des sections dans votre contexte.

Il vous suffit de "forker" le dépôt Git et d'y apporter vos propres modifications, puis de générer par vous même le nouveau PDF.