Note sur l'installation et utilisation d'un package avec `Pkg3` sous `Julia v1.0`

Xavier Gandibleux

15 août 2018

1 Introduction

Un package étend le langage de base avec de nouvelles fonctionnalités. Il doit être installé, à l'aide d'un gestionnaire de packages, avant d'être utilisé. L'installation est à faire qu'une seule fois. Le gestionnaire de packages de Julia a été revu à l'occasion de la sortie de la version 1.0 du langage de programmation ce 8 aout 2018 et devient `Pkg3` (voir la présentation dont le résumé est à `http://juliacon.org/2018/talks_workshops/64/` et la vidéo à `https://www.youtube.com/watch?v=GBi__3nF-rM`).

Complètement nouveau et reposant sur un principe différent de celui qui était en place jusqu'à la version 0.6.4 de Julia, `Pkg3` est conçu autour *d'environnements*, d'ensembles indépendants de *packages* qui peuvent être locaux à un *projet* individuel ou partagés et sélectionnés par nom. L'ensemble exact de packages et de versions dans un environnement est compris dans un *fichier manifeste* qui peut être vérifié dans un dépôt de projet et suivi par le contrôleur de version. `Pkg3` permet de garder les dépendances requises par différents projets dans des emplacements séparés, en créant des environnements. Il résout le dilemme "le projet X dépend de la version 1.x mais le projet Y nécessite la 3.x", et garde le répertoire site-packages global propre. Il est plus efficace mais aussi plus sophistiqué ; il distingue les notions de "Projet", "Packages", "Applications", "Library", "Environment", "Registry", "Depot", "Load path", et "Depot path".

Cette note se limite aux manipulations et commandes de base pour installer sous Julia version 1.0 et ultérieures un package référencé (registered package) dans la collection disponible à l'adresse `https://juliaobserver.com/`. Aussi, seuls les packages validés pour Julia v1.0 verront a priori leur installation aboutir sans lever d'erreur. Quand plusieurs actions sont possibles pour réaliser une même opération, seule une -la plus courante- est décrite. Respectant l'esprit d'une première prise en main, les détails techniques et fonctionnalités avancées ne sont pas abordés, le lecteur souhaitant acquérir une connaissance sur ces points se rapprochera de la documentation officielle dédiée au sujet sur le site Julia (`https://docs.julialang.org/en/latest/stdlib/Pkg/`).

Enfin, et à toute fin utile, l'annexe A reprend une présentation succincte des commandes pour installer un package sous Julia version 0.6.4 et antérieures.

Note : la production de ce texte ayant été réalisé dans la période de fermeture de l'université et donc sans possibilité de réaliser une relecture sur papier (l'achat d'une imprimante sur crédit recherche étant refusé par notre laboratoire), des fautes et imprécisions peuvent être présentes dans le texte ; je m'en excuse. Merci de me les communiquer par email (`Xavier.Gandibleux@univ-nantes.fr`) si vous vous en trouvez, ainsi que toute remarque que vous jugerez utile.

2 Mode Pkg REPL

Première nouveauté, le gestionnaire de package possède son propre mode REPL. Pour entrer dans le mode `Pkg REPL` depuis un REPL Julia, presser la touche "`]`". Le prompt devient :

```
(v1.0) pkg>
```

A partir de ce moment, les commandes du gestionnaire de packages peuvent être invoquées. Pour revenir au prompt `julia>`, appuyer sur `Ctrl+C`.

3 Commandes du gestionnaire de packages `Pkg3`

Il faut préalablement entrer dans le mode `Pkg REPL` avant d'invoquer les commandes suivantes.

Appel à l'aide en ligne :

```
help
```

affiche avec un commentaire succinct toutes les commandes disponibles sous le gestionnaire de packages.

Ajout d'un package référencé :

```
add   suivi du nom du package à installer
```

ajoute au projet courant le package qui est nommé. Plusieurs packages peuvent être ajoutés en une commande `add`. Il suffit de séparer les noms par au moins un espace, par exemple : `add vOptSpecific vOptGeneric`

Remarque : pour ajouter un package qui ne figure pas dans la collection des packages référencés, consultable à l'adresse `https://juliaobserver.com/`, il suffit de renseigner l'URL de son dépôt, par exemple : `add https://github.com/gsoleilhac/NSGAII.jl`

Donne le statut des packages ajoutés par l'utilisateur :

```
st
```

retourne les packages installés par l'utilisateur dans le projet.

Execute le test d'un package :

```
test   suivi du nom du package à tester
```

procède au test du package nommé.

Execute le script de construction des packages :

```
build   suivi du nom du package à construire
```

l'étape de construction d'un package est automatiquement effectuée quand un package est installé pour la première fois. Cette étape peut cependant être réalisée explicitement en invoquant la commande `build`.

Met à jour un package :

```
up   suivi du nom du package à mettre à jour
```

quand de nouvelles versions de packages utilisés par un projet sont disponibles, il est de bonne pratique de procéder à leur mise à jour en invoquant la commande `up`. Elle tentera la mise à jour de toutes les dépendances du projet vers la dernière version compatible.

Supprime un package :

```
rm   suivi du nom du package à supprimer
```

supprime dans le projet courant le package qui est nommé.

Effectue un ramasse-miettes (garbage collector) sur les packages :

```
gc
```

supprime les packages obsolètes et inutiles.

Sans plus de précision, les opérations présentées s'appliquent au projet "par défaut" (default project, sous `~/.julia/environments/v1.0`). Créer son propre projet ou créer un package se réalise à l'aide de manipulations et commandes du même type (voir la documentation officielle dédiée au sujet sur le site Julia à l'adresse `https://docs.julialang.org/en/latest/stdlib/Pkg/`).

4 Illustration avec le package `Example`

Le package `Example` est utilisé pour illustrer les effets des commandes.

```
(v1.0) pkg> add Example
 Resolving package versions...
 Installed Example - v0.5.1
  Updating `~/.julia/environments/v1.0/Project.toml`
  [7876af07] + Example v0.5.1
  Updating `~/.julia/environments/v1.0/Manifest.toml`
  [7876af07] + Example v0.5.1

(v1.0) pkg> st
    Status `~/.julia/environments/v1.0/Project.toml`
  [7876af07] Example v0.5.1

(v1.0) pkg> ^C

julia> using Example
julia> hello("Julia")
"Hello, Julia"
```

A Gestionnaire de packages sous Julia v0.6.4 et antérieures

Ajout d'un package référencé :

```
Pkg.add("  nom-du-package-à-installer  ")
```

ajoute dans l'environnement courant le package qui est nommé.

Statut des packages ajoutés :

```
Pkg.status()
```

liste les packages explicitement ajoutés à votre environnement Julia, ainsi que ceux qui ont été automatiquement ajoutés du fait de l'existence d'une dépendance.

Mettre à jour son environnement :

```
Pkg.update()
```

mise à jour de l'ensemble des packages de votre environnement.