

# Introduction à conda

...

*Loïc Gouarin, Alexis Jeandet, Vincent Rouvreau*

[https://github.com/gouarin/conda\\_tutorial](https://github.com/gouarin/conda_tutorial)

# Pourquoi utiliser conda ?

- Multi plateformes
- 32 et 64 bits
- Multi langages
- Environnements

*Pas besoin d'être administrateur  
de la machine !*

# Distributions

Anaconda

(<https://www.anaconda.com/>)

Miniconda

(<https://conda.io/miniconda.html>)

Anaconda Enterprise

*Nous vous encourageons à utiliser  
miniconda.*

# Après avoir installé miniconda

- Surcharge de votre PATH
- Commande conda pour gérer les paquets
- Une version Python et quelques packages de base



.../miniconda3

bin

lib

pkgs

envs

include

share

conda-  
meta



.../Miniconda3

Scripts

lib

pkgs

envs

include

share

conda-  
meta

**Gérer ses environnements de travail**

# Environnements

## Création

```
conda create -n myenv python=3.6
```

## Activer

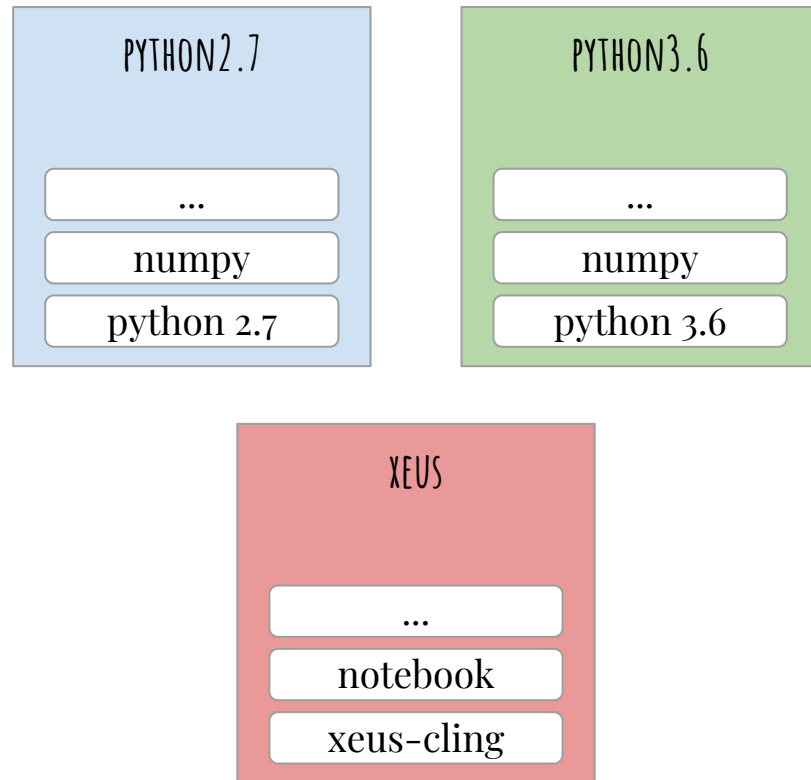
```
source activate myenv  
activate myenv (windows)
```

## Désactiver

```
source deactivate myenv  
deactivate myenv (windows)
```



# Environnements



# Environnements

## Lister

```
conda env list
```

## Enlever

```
conda env remove -n myenv
```

# Installation de packages

## Recherche

```
conda search numpy  
conda search 'numpy>1.10'  
conda search '*numpy*'
```

## Installation

```
conda install numpy  
conda install numpy=1.14.5
```

# Installation de packages

L'installation se fait dans  
l'environnement où vous vous  
trouvez.

Vous pouvez spécifier un autre  
environnement

```
conda install -n myenv numpy
```

# Installation de packages

## Enlever

```
conda remove numpy
```

## Mise à jour (tout, juste un)

```
conda update  
conda update numpy
```

## Lister

```
conda list
```

# Installation de packages

## Utilisation de pip

```
pip search numpy  
pip install numpy  
...
```

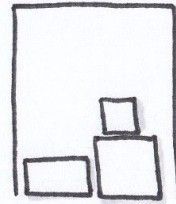
# Les dépôts

- Appelés *channels*
- Offre plus de choix
- Tout le monde peut ajouter ses packages



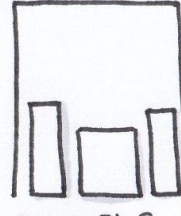
LOOKING FOR  
THIS  
PACKAGE

CHANNELS ARE LIKE STORAGES...



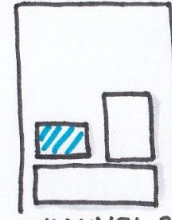
CHANNEL 1

FIRST, IT IS  
LOOKING FOR  
THE PACKAGE  
IN THIS CHANNEL  
(THIS HAS THE  
HIGHEST PRIORITY)



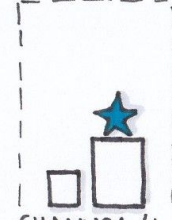
CHANNEL 2

IT WAS NOT  
IN THE FIRST  
CHANNEL, SO  
IT MOVES TO  
THE STORAGE  
WITH THE 2ND  
HIGHEST PRIORITY



CHANNEL 3

YAY! CONDA HAS  
FOUND THE PACKAGE!  
NOW IT IS ADDED  
TO YOUR ENVIRONMENT.



CHANNEL 4

WHAT HAPPENS IF CONDA DID NOT FIND THE PACKAGE?

BY DEFAULT, CONDA LOOKS FOR PACKAGES IN THE OFFICIAL STORAGES OF CONTINUUM.

WHY? BECAUSE BY DEFAULT THESE HAVE THE HIGHEST PRIORITIES.

HOWEVER, YOU HAVE THE POWER TO

**+** ADD NEW CHANNELS (STORAGES) THAT CONTAIN THE PACKAGES YOU NEED!

SO LET'S SAY YOU WANT TO INSTALL THIS PACKAGE: ★

→ CONDA DOES NOT FIND IT IN THE FIRST 3 CHANNELS

⇒ YOU NEED TO ADD A 4TH ONE!

THEY ARE  
CONDA'S  
DEVELOPERS.



# Les dépôts

## Utiliser un dépôt ponctuellement

```
conda install -c conda-forge numpy
```

## Ajouter un dépôt par défaut

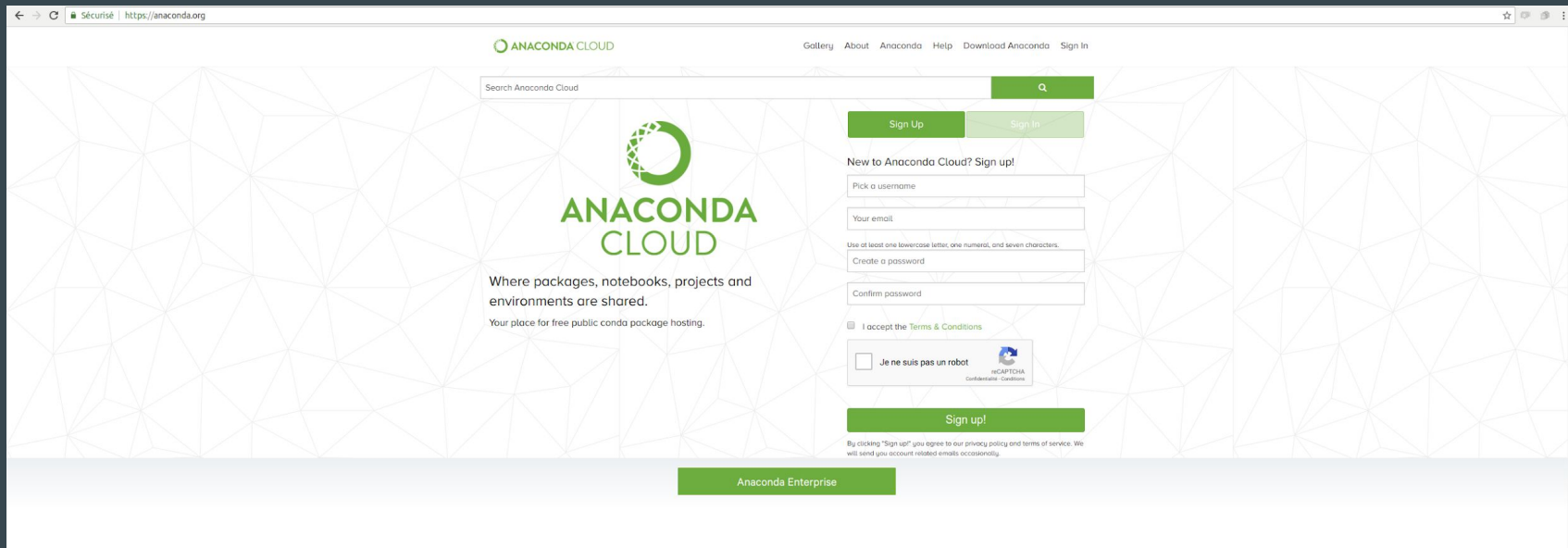
```
conda config --add channels \  
new_channel
```

## Le retirer

```
conda config --remove channels \  
new_channel
```

## Lister les dépôts par défaut


```
conda config --show channels
```



ou en ligne de commande

```
conda install anaconda-client  
anaconda search numpy
```

ONE MORE ASPECT:      PRIORITY > VERSION

LET'S SAY CHANNEL 1 (HIGHEST PRIORITY CHANNEL)  
CONTAINS THE VERSION 1.0 OF THIS PACKAGE:   
AND CHANNEL 4 (LOWEST PRIORITY)  
CONTAINS A MUCH NEWER, 2.0 VERSION OF IT.

WHICH ONE IS GOING TO BE INSTALLED?

VERSION 1.0, SINCE IT IS INSIDE A HIGHER PRIORITY CHANNEL!



# Distribuer son environnement

Le fichier  
environment.yml

```
name: myenv
```

```
channels:
```

- conda-forge
- gouarin

```
dependencies:
```

- python=3.6
- numpy
- scipy
- splinart
- pip:
  - pywavelets

# Distribuer son environnement

## L'export

```
conda list --explicit > myenv.txt
```

Démo !!

# Exercice

1. Récupérer le fichier [example.ipynb](#)
2. Construire un environnement permettant d'avoir le notebook fonctionnel
3. Exporter l'environnement
4. Créer un environnement vierge à partir de cet export et voir si tout fonctionne bien

**Construction de ses paquets**



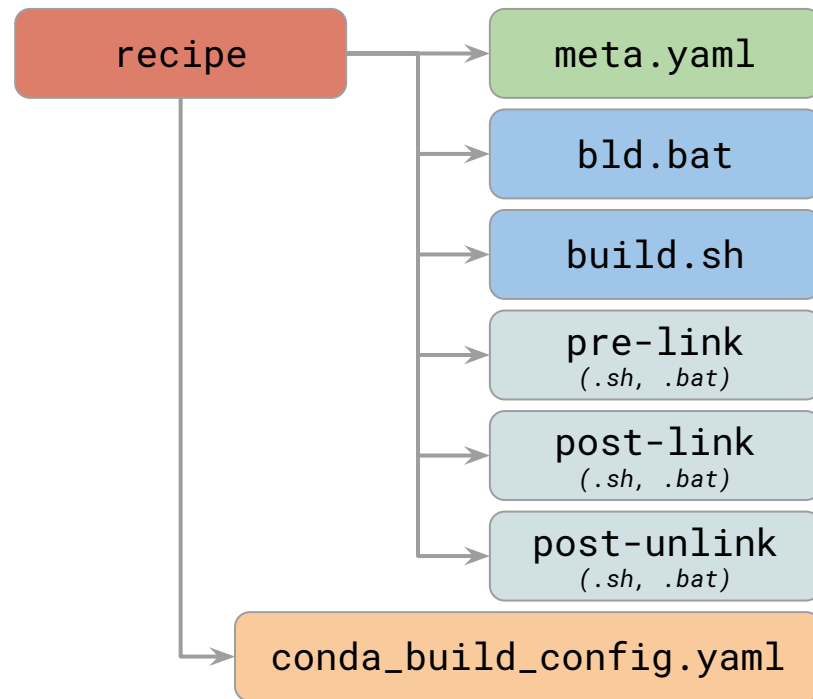
# Construire son package conda

Les paquets indispensables

```
conda install conda-build
```

```
conda install anaconda-client
```

# Structure du package



# Exemple de meta.yaml

## Informations du package

```
{% set name = "splinart" %}
```

```
{% set version = "0.1.9" %}
```

```
package:
```

```
  name: {{ name }}
```

```
  version: {{ version }}
```

```
source:
```

```
  git_rev: {{ version }}
```

```
  git_url: https://github.com/gouarin/{{ name }}.git
```

```
build:
```

```
  number: 0
```

```
  script: python setup.py install
```

# Exemple de meta.yaml

Construction, test et  
installation

```
requirements:
```

```
  build:
```

- python
- setuptools

```
  run:
```

- python
- numpy
- matplotlib
- six

```
test:
```

```
  imports:
```

- splinart

# Exemple de meta.yaml

Construction, test et  
installation

requirements:

*host: (depuis conda-build 3)*

- python
- setuptools

run:

- python
- numpy
- matplotlib
- six

test:

imports:

- splinart

# Exemple de meta.yaml

## Informations générales

```
about:  
  home: http://github.com/gouarin/splinar  
  license: BSD  
  description: 'spline art generator'  
  
extra:  
  recipe-maintainers:  
    - gouarin
```

# Exemple de build.sh

```
#!/bin/bash
```

```
python setup.py install
```

# Construction en local

conda build recipe

*Construction d'un environnement minimal*

*Installation des paquets nécessaire pour la construction*

*Exécution du build.sh*

*Installation du binaire dans un nouvel environnement vierge*

*Tests*



# Installation en local

```
conda install splinart --use-local
```

# meta.yaml

## Section source

### Source depuis une archive

```
source:  
  fn: xeus-cling-0.4.7.tar.gz  
  url: https://github.com/QuantStack/xeus-cling/archive/0.4.7.tar.gz  
  sha256: 713dcbed276882e4...
```

### Source locale

```
source:  
  path: ../src
```

### Ajout d'un patch

```
source:  
  ...  
  patches:  
    - my.patch1  
    - my.patch2
```

*Possibilité d'utiliser mercurial et svn  
comme pour git.*

# meta.yaml

## Section requirements

### Build

Tous les outils nécessaires pour construire le package qui sont indépendants de l'architecture cible

### Host *(depuis conda-build 3)*

Tous les outils nécessaires pour construire le package sur une architecture cible

### Run

Tous les outils nécessaires pour utiliser le package sur une architecture cible

# meta.yaml

## Section test

### Requirements

```
test:  
  requires:  
    - nose
```

### Commands

```
test:  
  commands:  
    - test -f ${PREFIX}/bin/xeus-cling # [unix]  
    - if exist %LIBRARY_PREFIX%\bin\xeus-cling.exe (exit 0) else  
(exit 1) # [win]
```

### Imports

```
test:  
  imports:  
    - splinart
```

# Variables d'environnement

Conda build offre plusieurs variables d'environnement facilitant la construction du package.

- PREFIX
- PY\_VER
- SRC\_DIR
- ...

[Pour plus d'informations](#)

# Les sélecteurs

```
package:
```

```
  name: mypackage
```

```
  version: 0.1
```

```
build:
```

```
  number: 0
```

```
  skip:
```

```
    - true    # [ win and py<35 ]
```

```
requirements:
```

```
  host:
```

```
    - mpich    # [linux]
```

```
    - openmpi  # [osx]
```

```
    - msmapi   # [win]
```

```
...
```

Voir la [liste complète](#)

# Les variantes

- Permet de fixer les versions au build et au runtime
- Permet de faire plusieurs versions d'un package avec la même recette

# conda\_build\_config.yaml

```
python:
```

- 2.7
- 3.5
- 3.6

*conda\_build\_config.yaml*

```
package:
```

```
  name: mypackage  
  version: 0.1
```

```
...
```

```
requirements:
```

```
  host:
```

- python {{ python }}

```
  run:
```

- python

*meta.yaml*



# conda\_build\_config.yaml

```
boost:  
  - 1.61  
  - 1.63  
pin_run_as_build:  
  boost: x.x
```

*conda\_build\_config.yaml*

```
package:  
  name: mypackage  
  version: 0.1  
  
...  
  
requirements:  
  host:  
    - boost {{ boost }}  
  run:  
    - boost
```

*meta.yaml*

# Les compilateurs

```
requirements:
```

```
build:
```

- {{ compiler('c') }}
- {{ compiler('cxx') }}
- {{ compiler('fortran') }}

Démo !!

# Exercice

Ecrire la recette pour construire le package conda de l'un des projets suivants (ou un autre de votre choix)

- [https://github.com/bclmary/eccw\\_gui](https://github.com/bclmary/eccw_gui)
- <https://github.com/vle-forge/vle>
- <https://bitbucket.org/bixente/3photons>
- <https://github.com/pierrepo/autoclasswrapper>
- <https://github.com/GenomicParisCentre/eoulsan>
- [https://github.com/sagemath/sage\\_sample](https://github.com/sagemath/sage_sample)
- <https://github.com/mirebeau/hamiltonfastmarching/>
- <https://github.com/ToFuProject/tofu>

**Distribution de ses paquets**

# Son dépôt

Se créer un compte

<https://anaconda.org/>

Une fois le package créé

```
anaconda upload .../miniconda3/conda-bld/...tar.bz2
```

Pour utiliser le package

```
conda install -c mychannel mypackage
```



# CONDA-FORGE

A community led collection of recipes, build  
infrastructure and distributions for the conda package  
manager.



# conda-smithy

Facilite la construction d'un  
package sur différents OS

- Enregistre votre recette sur github
- Utilise les CI disponibles sur github
- Upload les packages directement sur anaconda cloud



# A vos tokens !

Pour que conda-smithy  
fonctionne correctement

Il est nécessaire de mettre les  
tokens dans le répertoire  
`$HOME/.conda-smithy`

- `github.token`  
<https://github.com/settings/tokens/new>
- `anaconda.token`
- `circle.token`  
<https://circleci.com/account/api>
- `travis.token`
- `appveyor.token`  
<https://ci.appveyor.com/api-token>

Démo !!

# Exercice

Déployer le package fait  
précédemment sur votre channel à  
l'aide de `conda-smithy`.