POLYTECH SORBONNE

ROS

Robot Operating System

ROS

Historique

ROS 1.0	2010
Box Turtle	2010
C Turtle	2010
Diamondback	2011
Electric	2011
Fuerte	2012
Groovy	2012
Hydro	2013
Indigo	2014
Jade	2015
Kinetic	2016
Lunar	2017
Melodic	2018
Noetic	2020

ROS

Introduction

- ROS = Meta système d'exploitation pour la robotique (middleware)
- Caractéristiques principales :
  - Open source
  - Modulaire
  - Multi-langage (C++, Python, …)
- Permet d'uniformiser les architectures logiciels de robots
- Origine : Stanford Artificial Intelligent Lab

ROS

Open source

- www.ros.org
- Large communauté de développeurs
- Documentation, Wiki
- Tutoriels
- Accès au code source !

ROS

Principaux avantages

- Architecture de communication inter-processus et inter-machine
- Serveur de paramètres
- Système d'enregistrement et de rejeu
- Système de test
- Simulateur

ROS

Installation

- Sur linux, de préférence Ubuntu
- Kinetic→ Ubuntu 16 ou Melodic → Ubuntu 18
- http://wiki.ros.org/kinetic/Installation/Ubuntu

```
$ sudo apt-get install ros-kinetic-desktop-full
```

ROS

Démarrer ROS

```
$ roscore
```

- Cette commande lance ce qu'on appelle le master
- Il faut paramétrer les adresses IP des différentes machines s'il y'en a, afin qu'elles puissent communiquer entre elles :

```
$ export ROS_MASTER_URI=http://192.168.0.21:11311
```

```
$ export ROS_IP=192.168.0.31
```

- Enregistré dans le fichier `~/.bashrc`

ROS

Le master

- C'est le processus cœur
- Il comprend un serveur de noms
- Les différents processus qui seront lancés se présenteront à lui
- Il va mettre les processus en contact

→ Les processus en contact peuvent alors communiquer entre eux

```
$ roscore
```

ROS

Les nœuds (nodes)

- Les nœuds sont des processus indépendants
- Un nœud est une instance d'un exécutable
- Il peut être lié à un moteur, un capteur, ou purement logiciel...
- Un nœud peut envoyer et écouter des messages ROS à travers des topics

```
$ rosnode
```

ROS

Les topics

- Un topic est un système de transport de messages
- Périodique: il est publié à une fréquence définie
- Typé : il faut préciser quel type de messages on transporte
- Basé sur un système d'abonnement / publication (subscribe / publish)

```
$ rostopic
```

```
$ rosmsg
```

Exemple

Node 1 —publish→ **Topic** ←subscribe— **Node 2**

ROS

Les services

- Un service est aussi un système de transport de message
- Mais non périodique
- Il faut aussi préciser quel type de messages on transporte
- Basé sur un système de client/serveur
- Structure de donné : request/response

```
$ rosservice
```

```
$ rossrv
```

ROS

Les packages

- Un package est un dossier permettant d'organiser les fichiers
- Il contient :

- Package.xml
- CmakeLists.txt
- src, include
- script*
- srv*
- launch*

* optionnel

```
$ rospack
```

```
$ roscd
```

ROS

Catkin workspace

- Préparer un répertoire de travail (workspace)

```
$ mkdir -p ~/catkin_ws/src
```

```
$ cd ~/catkin_ws/src
```

```
$ catkin_init_workspace
```

- Créer un nouveau package

```
$ catkin_create_package nom_du_package liste_dependance
```

Ensuite il suffit d'éditer le fichier CMakeList.txt pour ajouter des exécutables/librairies/services/dépendances...

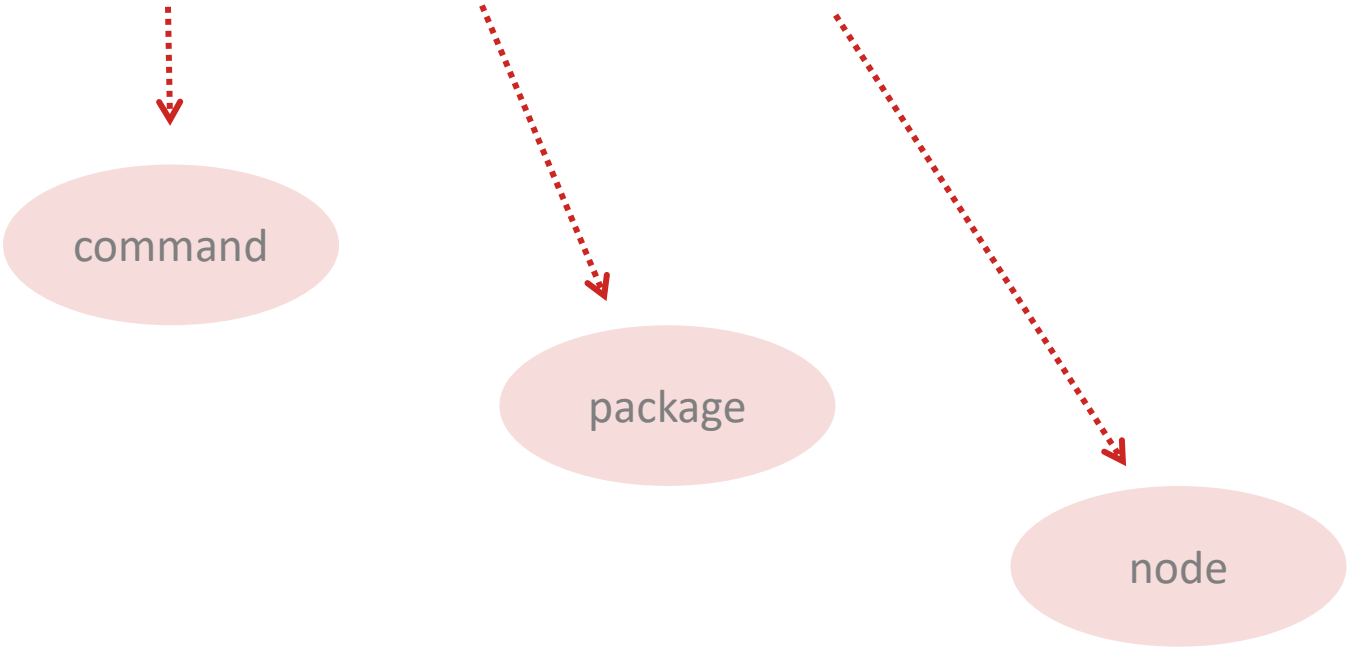
- Compiler, ATTENTION il faut obligatoirement se placer dans le répertoire ~/catkin_ws

```
$ catkin_make
```

ROS

Execution d’un node

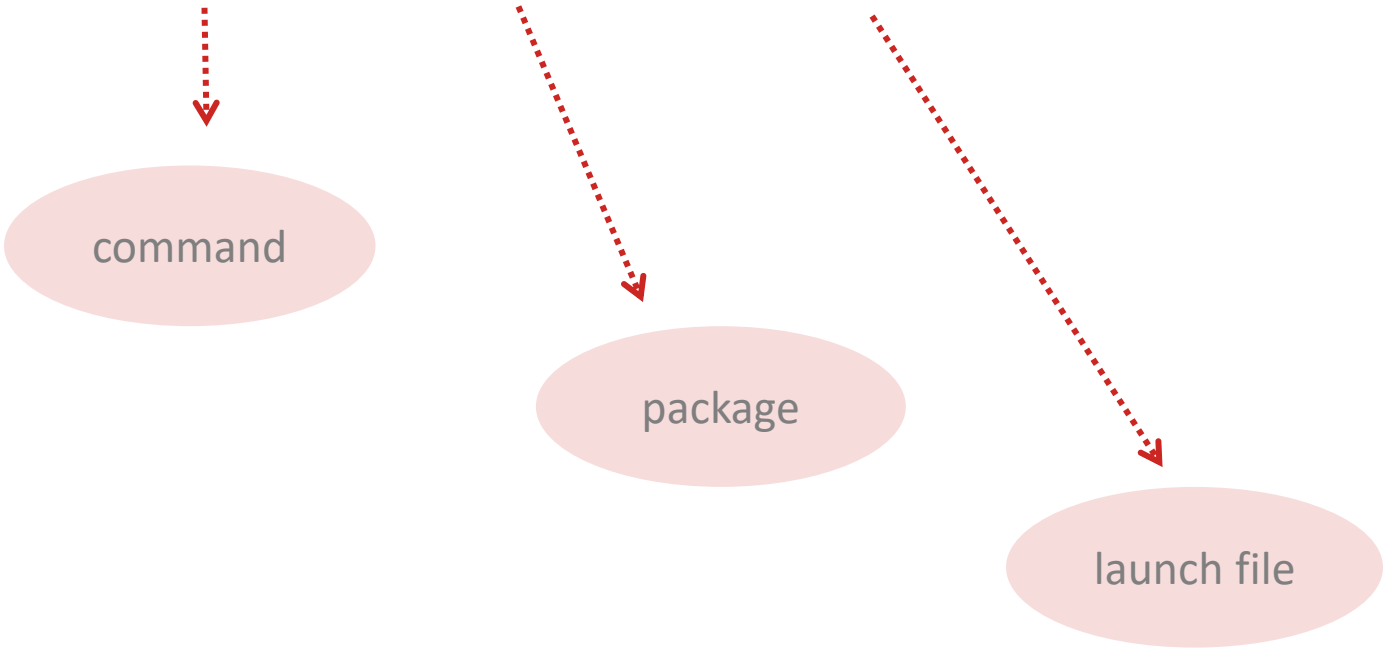
```
$ rosrun turtlesim turtlesim_node
```



Execution d’un launch file

```
<launch>
        <group ns="turtlesim1">
                <node pkg="turtlesim" name="sim" type="turtlesim_node"/>
        </group>
        <group ns="turtlesim2">
                <node pkg="turtlesim" name="sim" type="turtlesim_node"/>
        </group>
</launch>
```

```
$ roslaunch turtlesim multisim.launch
```



ROS

Les commandes basiques de ROS

- roscore
- roscd
- rospack
- rosrun
- roslaunch
- rosservice
- rosclean
- rosnode
- rostopic
- rosmsg
- rossrv
- rosdep
- rosbag
- (rqt et rviz)