



# INTRODUCTION À ROS2

# Déroulé de la semaine

Lundi	Mardi	Mercredi	Jeudi
Introduction	Robotique Mobile	Robotique de manipulation	Quadrucoptère

## Ressources disponibles

Ce cours: [https://github.com/Lexoa/ros2\\_intro.git](https://github.com/Lexoa/ros2_intro.git)

Tutoriel officiel pour ros2 humble: <https://docs.ros.org/en/humble/Tutorials.html>

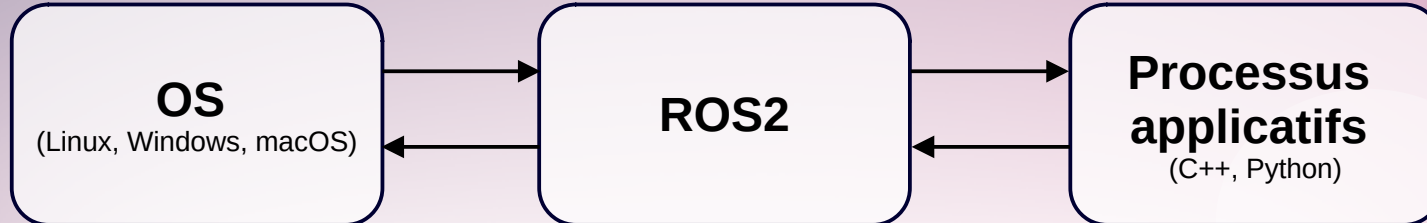
Forum d'entraide : <https://robotics.stackexchange.com/>

Version de ce cours sous ros1 : <https://learn.ros4.pro/fr/>

# Le framework



- **ROS** = Robot Operating System
- Middleware robotique
- Langages principaux : C++ et Python
- Mais aussi... Ada, Go, Java, Node.js, Obj. C, C, Rust, .Net



# Le framework



- **ROS =**
  - Bibliothèques logicielles
  - Outils logiciels : RViz2, Gazebo, rqt...
  - Implémentations de l'état de l'art : navigation, manipulation,...
  - Ressources documentaires et communautaires

# Le framework



- **ROS =**

- ros                      Core : rclpy, rclcpp
- ros-planning              MoveIt2 / OMPL / Task Planning
- ros2-control              Generic control interface
- ros-perception              tf2, cv\_bridge, image\_transport,...
- ros-vizualisation              RViz2, rqt
- ros-simulation              Gazebo/Ignition
- ros2-drivers              ros2\_serial\_example, openni2\_camera,...

# Le framework



- **Basés sur ROS**

- Ros-industrial      UR, Fanuc, ABB, KUKA, PAL ...
- Autoware.Auto      Autonomous driving framework
- Rapyuta.io      Cloud robotics framework
- AWS Robomaker      Cloud robotics framework









- **Interfacés avec ROS**

- CoppeliaSim
- Webots

# Distributions

# ROS 2

## ROS1 (2010-2025)

ROS Noetic Ninjemys (Recommended)	May 23rd, 2020			May, 2025 (Focal EOL)
ROS Melodic Morenia	May 23rd, 2018			June 27, 2023 (Bionic EOL)
ROS Lunar Loggerhead	May 23rd, 2017			May, 2019
ROS Kinetic Kame	May 23rd, 2016			April, 2021 (Xenial EOL)

## ROS2 (2017+)

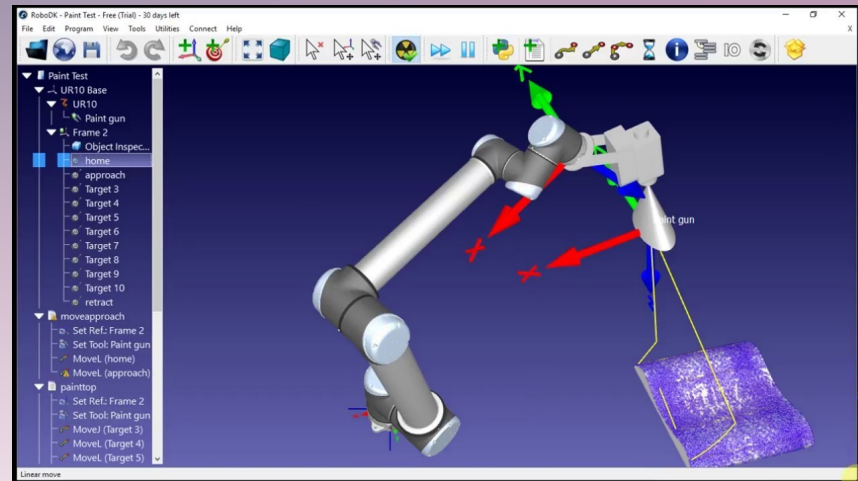
Distro	Release date	Logo	EOL date	ROS Boss
Jazzy Jalisco	May 23, 2024		May 2029	Marco A. Gutiérrez
Iron Irwini	May 23, 2023		December 4, 2024	Yadunund Vijay
Humble Hawksbill	May 23, 2022		May 2027	Audrow Nash
Galactic Geochelone	May 23, 2021		December 9, 2022	Scott Logan
Foxy Fitzroy	June 5, 2020		June 20, 2023	Jacob Perron / Dharin
Eloquent Elusor	November 22, 2019		November 2020	Michael Carroll
Dashing Diademata	May 31, 2019		May 2021	Steven! Ragnarök

# Le framework

# ROS 2

ROS vs RobotDK ...

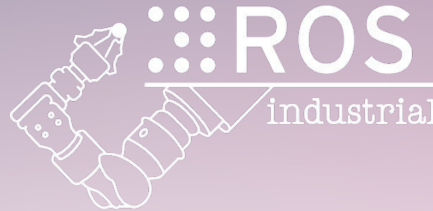
- Framework logiciel  $\neq$  logiciel
- On écrit du code C++/Python
- Avec des outils graphiques en plus
- Programmation  $\neq$  paramétrage
- Le mot d'ordre : Interopérable
- + de code logiciel = besoin de + de maîtrise de la qualité





# ROS 2

- Framework opensource foisonnant
- Origine académique où chacun rajoute sa brique
- Développement peu testés et non forcément maintenus

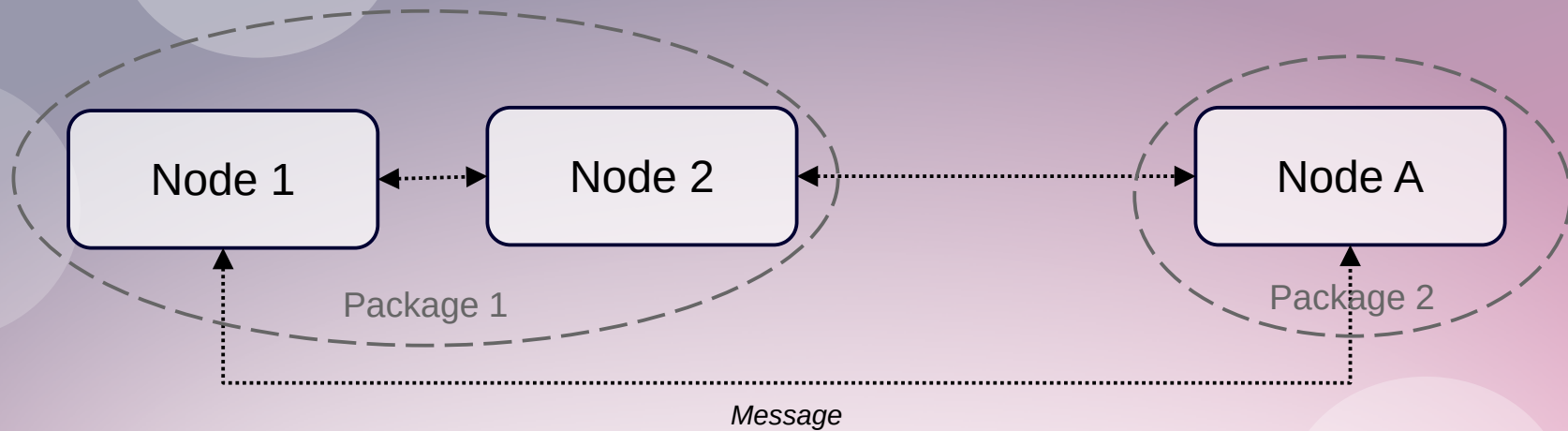


- Sous-ensemble de ROS qualitatif « industry-grade »
- Label attribués aux packages ROS respectant des bonnes pratiques et des tests
- Objectif : Faciliter le respect des normes et la certification des machines



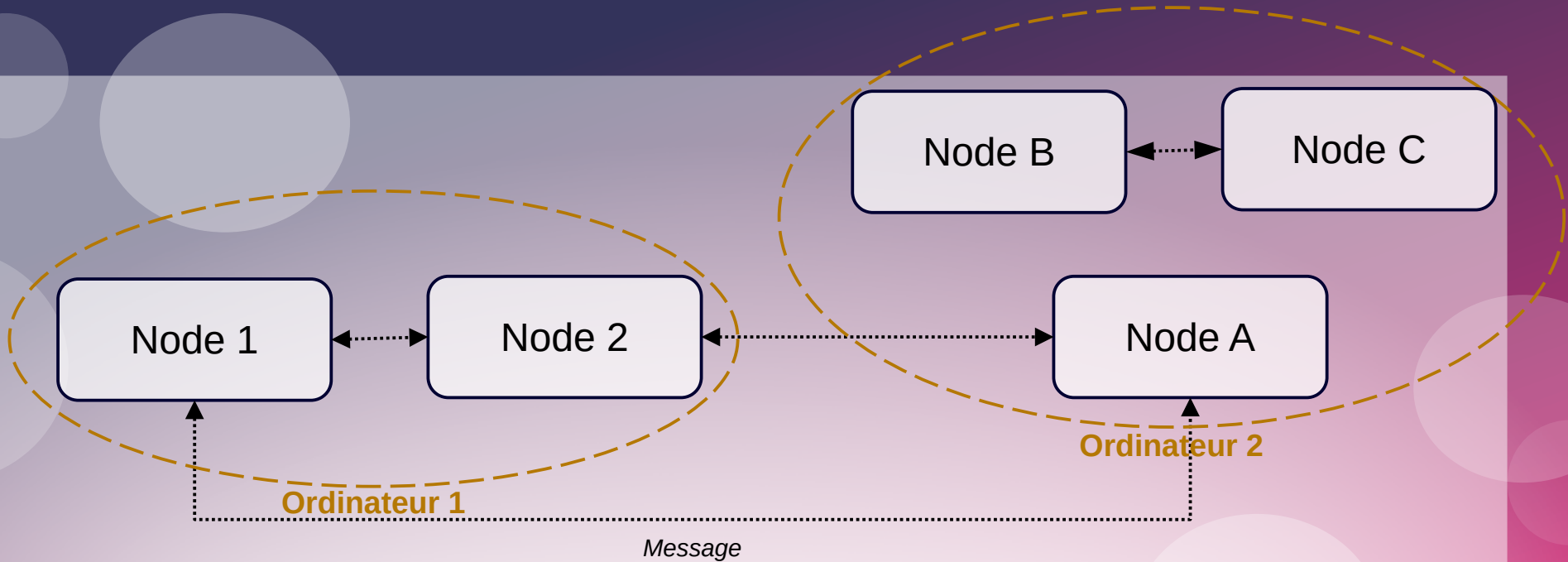
## INTRODUCTION À ROS2

# Nœuds



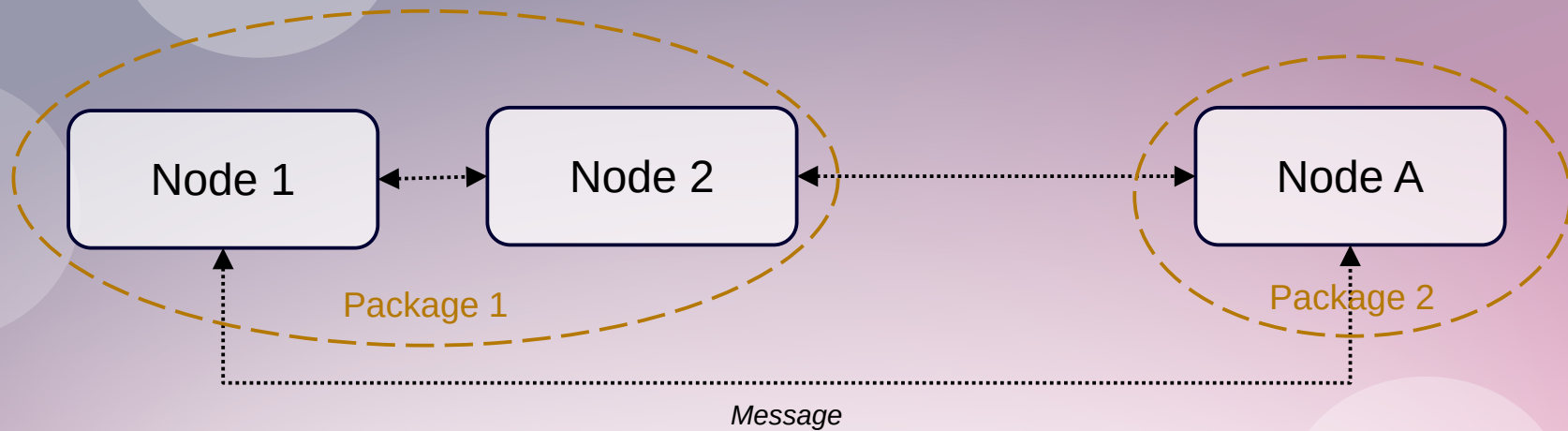
Les nœuds représentent la plus petite unité de processuer de ROS. C'est un programme exécutable qui ne fait qu'une chose à la fois.

# Domain ID



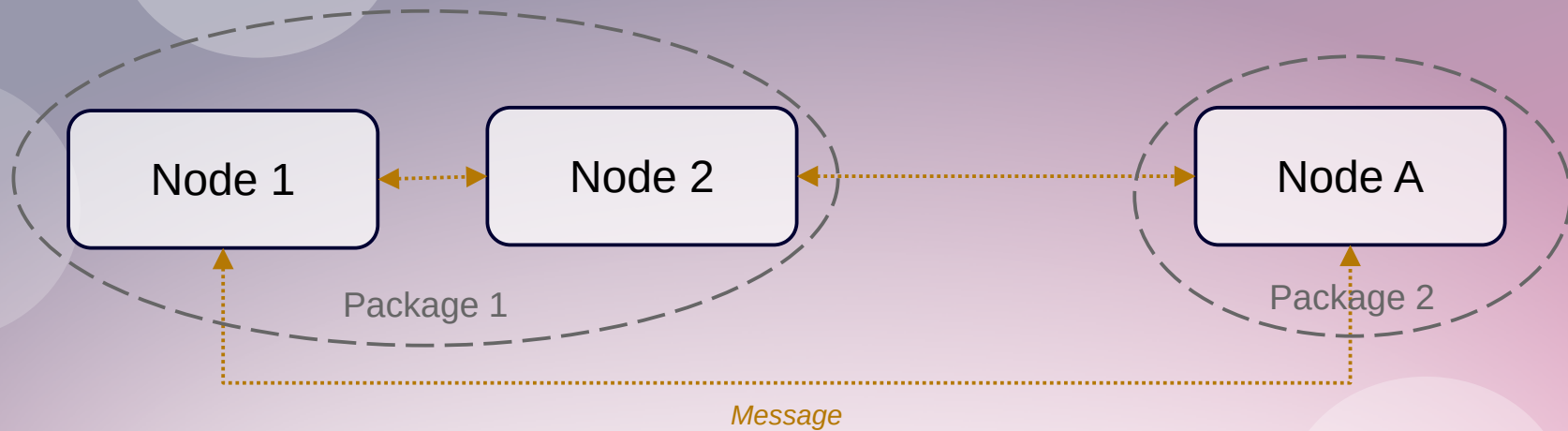
En ROS2 toutes les machines d'un réseau ont accès au node des autres, il est important de définir un Domain ID (entre 0 et 101) propre à chaque machine.

# Les packages



Les packages sont utilisés afin d'organiser les programmes sous ROS.

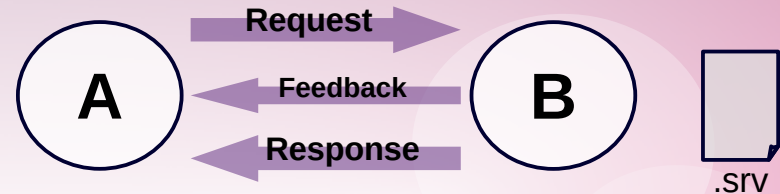
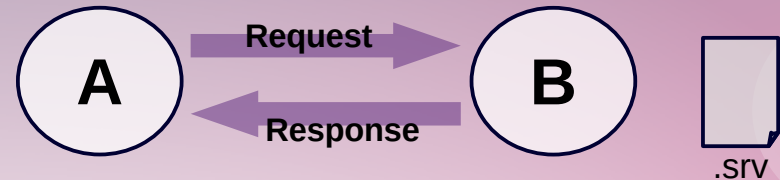
# Topics et messages



Les nœuds communiquent entre eux en se transmettant des infos sur des canaux de communication appelés Topics

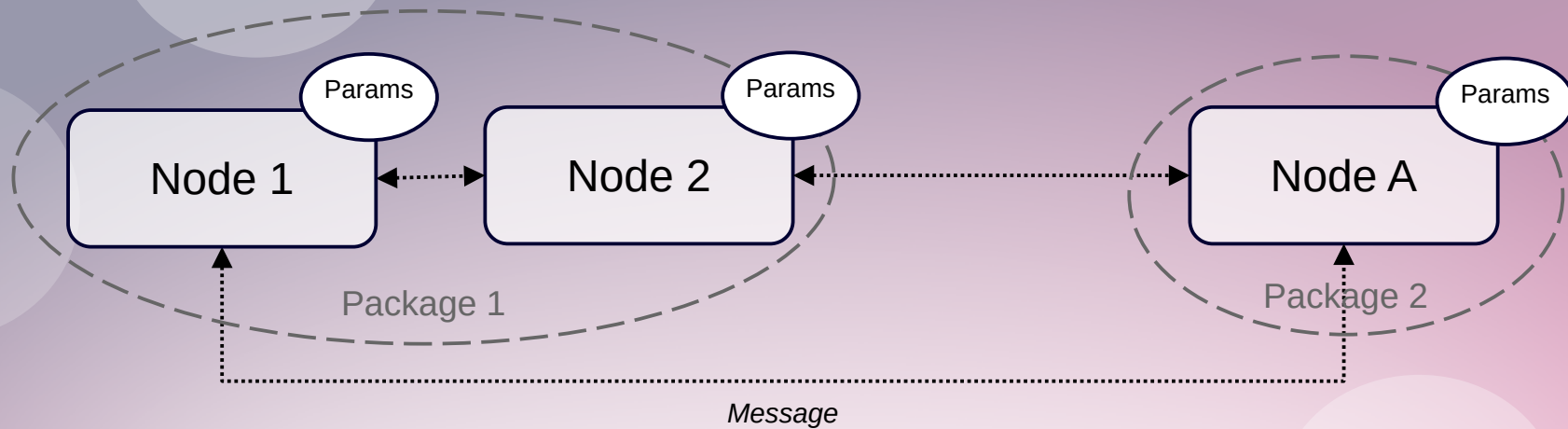
# Messages / Services / Action

- Messages
  - Publisher/Subscriber
- Services
  - Blocking Request/Response
- Action services
  - Async Request/Response





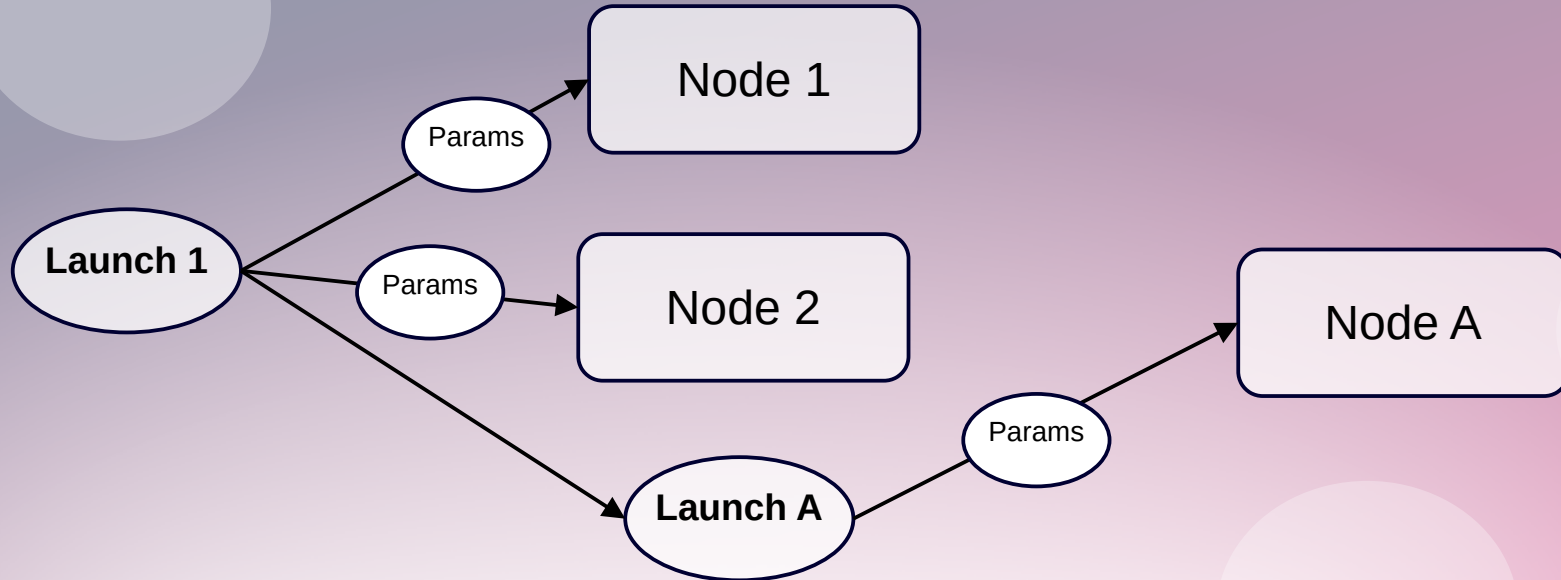
# Serveur de paramètres



Les serveurs de paramètres permettent de sauvegarder la configuration de chaque node sous forme de *.yaml*



# Launching nodes



Les fichiers launch permettent de lancer plusieurs node en même temps avec les paramètres associés



**PREMIER PAS SUR ROS2**

# Workspace & package ROS2

**Colcon** est le build system utilisé par ROS2, il permet de prendre en compte les dépendances entre les packages lors de la compilation.

- Créer un package :

```
~/ros2_ws/src$ ros2 pkg create --build-type ament_cmake --license Apache-2.0 <package_name>
```

- Compiler le workspace :

```
~/ros2_ws/$ colcon build
```

- Clean le workspace

```
~/ros2_ws/$ colcon build --cmake-target clean
```

- Chargé un workspace compilé :

- ~\$ `source ~/ros2_ws/install/setup.bash`
- ~\$ `source /opt/ros/humble/setup.bash`



À ajouter au `.bashrc`

# Commande ROS2 de base

- Démarrer un nœud :
  - ~\$ `ros2 run <package_name> <executable_name>`
  - Changer le nom de la node :
    - ~\$ `ros2 run <package_name> <executable_name> --ros-args --remap __node:=<my_name>`
  - Charger un fichier de parametre de node :
    - ~\$ `ros2 run <package_name> <executable_name> --ros-args --params-file <file_name>`
- Utiliser un launch file :
  - ~\$ `ros2 launch <package_name> <launchfile_name>`
  - Changer un parametre
    - ~\$ `ros2 launch <package_name> <launchfile_name> <param_name>:=<my_param_value>`

# Commande ROS2 de base

- Commandes liées au node:

~\$ `ros2 node list`      ~\$ `ros2 node info <node_name>`

- Commandes liées au topic:

~\$ `ros2 topic list`      ~\$ `ros2 topic info <topic_name>`

~\$ `ros2 topic echo <topic_name>` ~\$ `ros2 topic pub <topic_name> <msg_type> '<args>'`

- Commande liées au service :

~\$ `ros2 service list -t` ~\$ `ros2 service type <service_name>`

~\$ `ros2 service call <service_name> <service_type> <args>`

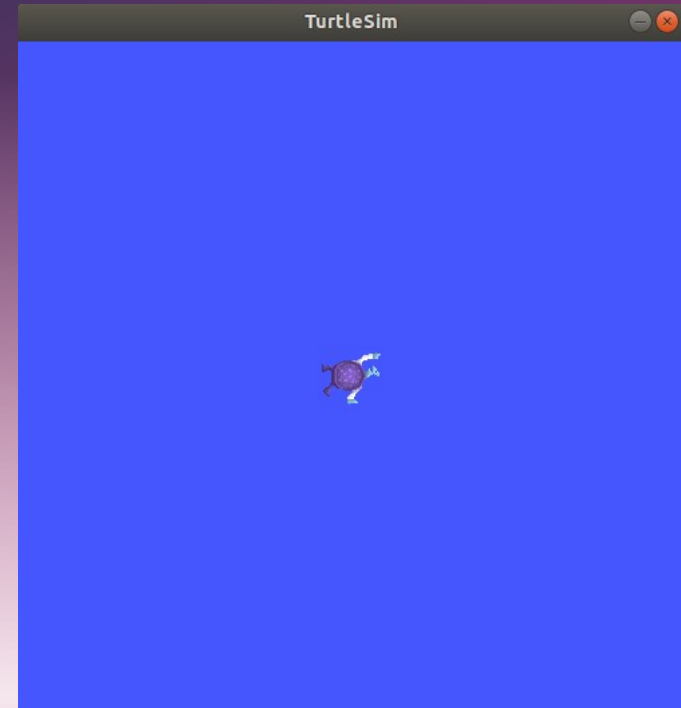
- Commande liées au param :

~\$ `ros2 param list -t`    ~\$ `ros2 param get <node_name> <param_name>`

~\$ `ros2 param set <node_name> <param_name> <value>`

# Démo turtlesim

- `ros2 run turtlesim turtlesim_node`
- `rqt`
- `ros2 run turtlesim turtle_teleop_key`
- `ros2 node info /turtlesim`
- `rqt_graph`
- `ros2 topic list`
- `ros2 topic pub /turtle1/cmd_vel geometry_msgs/msg/Twist "{linear: {x: 2.0, y: 0.0, z: 0.0}, angular: {x: 0.0, y: 0.0, z: 1.8}}"`
- `ros2 service call /spawn turtlesim/srv/Spawn "{x: 2, y: 2, theta: 0.2, name: ''}"`
- `ros2 param set /turtlesim background_r 150`
- `ros2 launch turtlesim multisim.launch.py`





**PLACE À LA PRATIQUE**

# Place à la pratique

1. Écrire un publisher qui envoie un string : le nom de votre ordinateur et un int : votre domain\_id + afficher le message via une commande et via un subscriber.
2. Avec un autre node, écrire un service qui permet d'envoyer le nom de votre ordinateur et votre domain\_id. La réponse devra indiquer si il s'agit du bon nom et du bon numéro.
3. Utiliser un launch pour lancer le serveur du service et le publisher, utiliser un paramètre pour modifier le nom de l'ordinateur et le domain\_id,

[https://github.com/Lexoa/ros2\\_intro.git](https://github.com/Lexoa/ros2_intro.git)