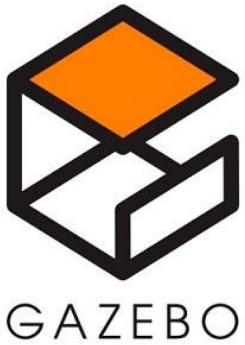
ROS and Gazebo
Iot Lab Course

What we’re going to use in this course



Installation Instructions for ROS:

Linux:

- `sudo apt install ros-humble-desktop`

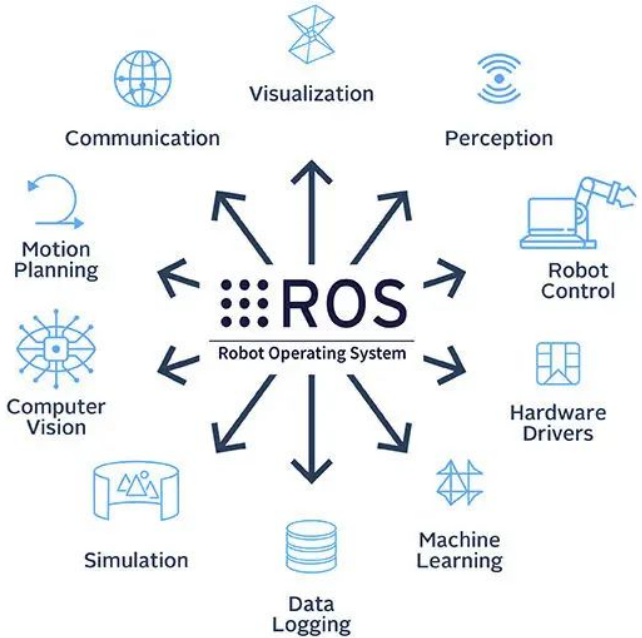
For **MacOS** and **Windows** refer to the website for the full installation instructions:

- https://docs.ros.org/en/humble/Installation.html

NOTE: While Windows is perfectly compatible with ROS, everything we will see for the project will use commands only compatible with Linux and Mac based OS, so it is highly recommended to install a VM if you only have Windows on your PC.

Robot Operating System - ROS

- The Robot Operating System (ROS) is a set of software **libraries** and **tools** that help you build **robot applications**.
- ROS offers a **standard** software platform to developers across industries that will carry them from research and prototyping all the way through to deployment and production.
- Allows the developer to implement more easily robot solutions without the hassle of handling multiple dependencies and different standards.

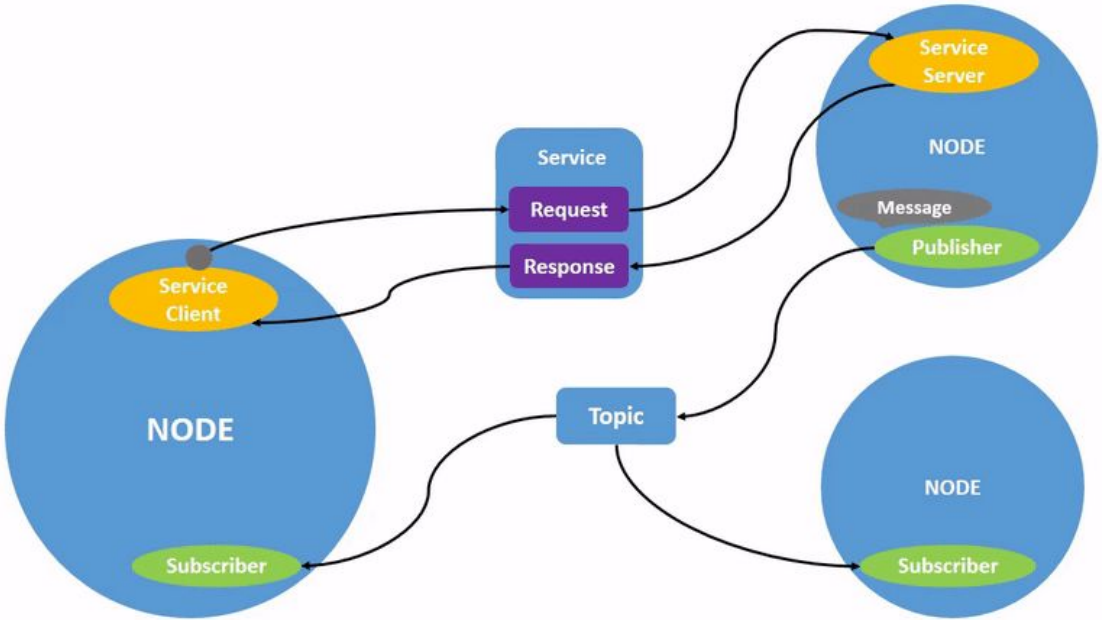


Why ROS?

- Global Community
- Proven in Use
- Multi-domain
- Multi-platform (sigh)
- Open Source
- Commercial Friendly
- Shortens time to deploy a new solution

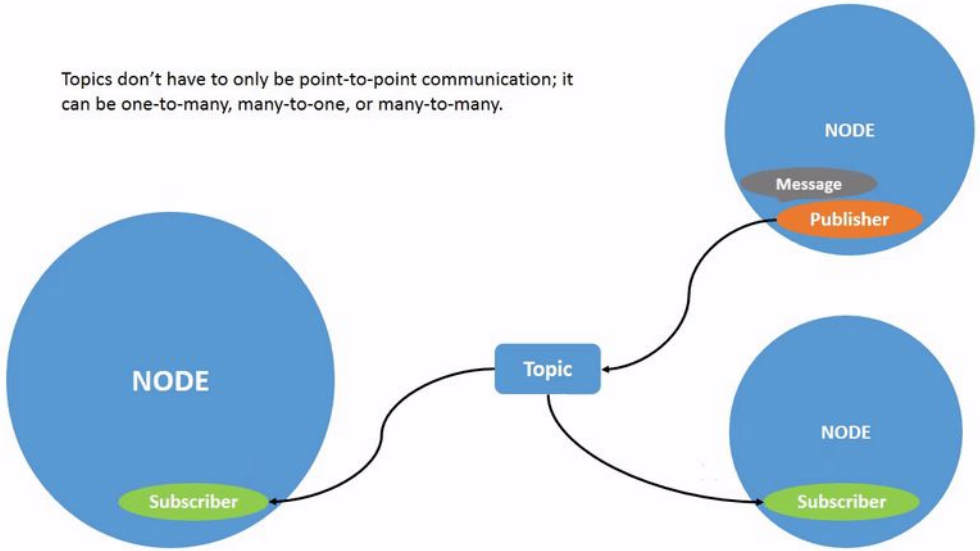
ROS Graph

- The ROS graph is a network of ROS 2 elements processing data together at one time
- It encompasses all executables and the connections between them if you were to map them all out and visualize them.
- Each **node** in ROS should be responsible for a **single**, **module** purpose (e.g. one node for controlling wheel motors, one node for controlling a laser range-finder, etc).
- Each node can send and receive data to other nodes via topics, services, actions, or parameters.



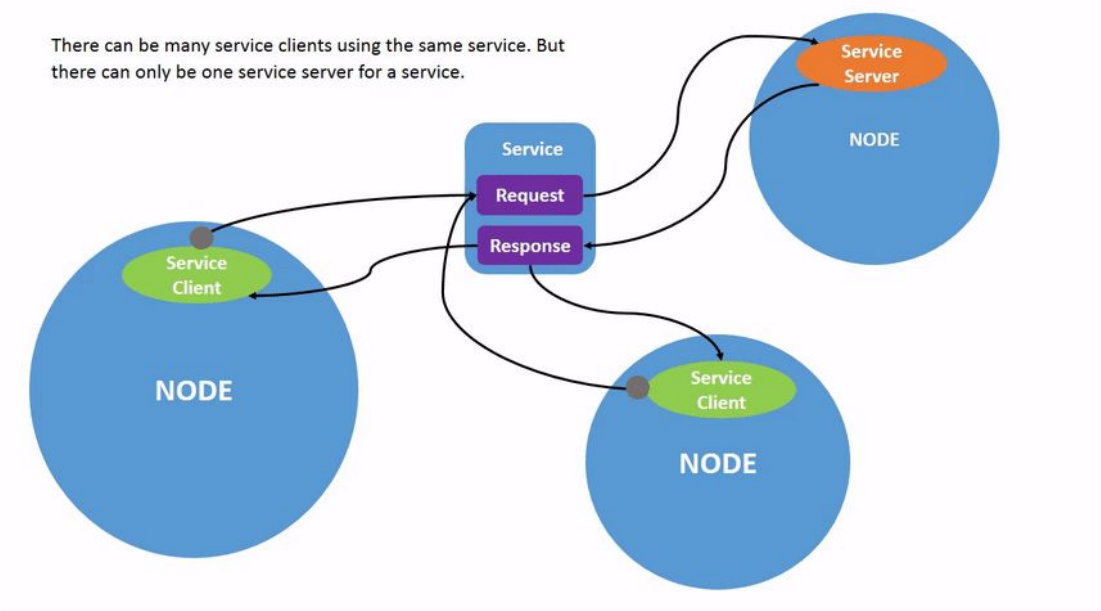
Topics

- Topics are a vital element of the ROS graph that act as a bus for nodes to exchange messages.
- Topics are one of the main ways in which data is moved between nodes and therefore between different parts of the system.
- A node may publish data to any number of topics and simultaneously have subscriptions to any number of topics.



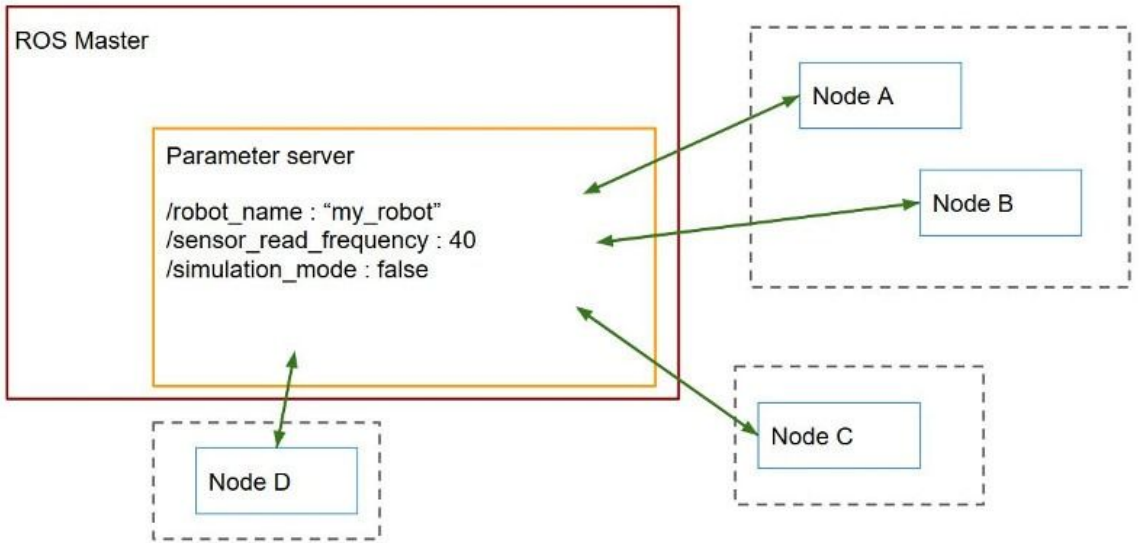
Services

- Services are another method of communication for nodes in the ROS graph.
- Services are based on a call-and-response model, versus topics' publisher-subscriber model.
- While topics allow nodes to subscribe to data streams and get continual updates, services only provide data when they are specifically called by a client.



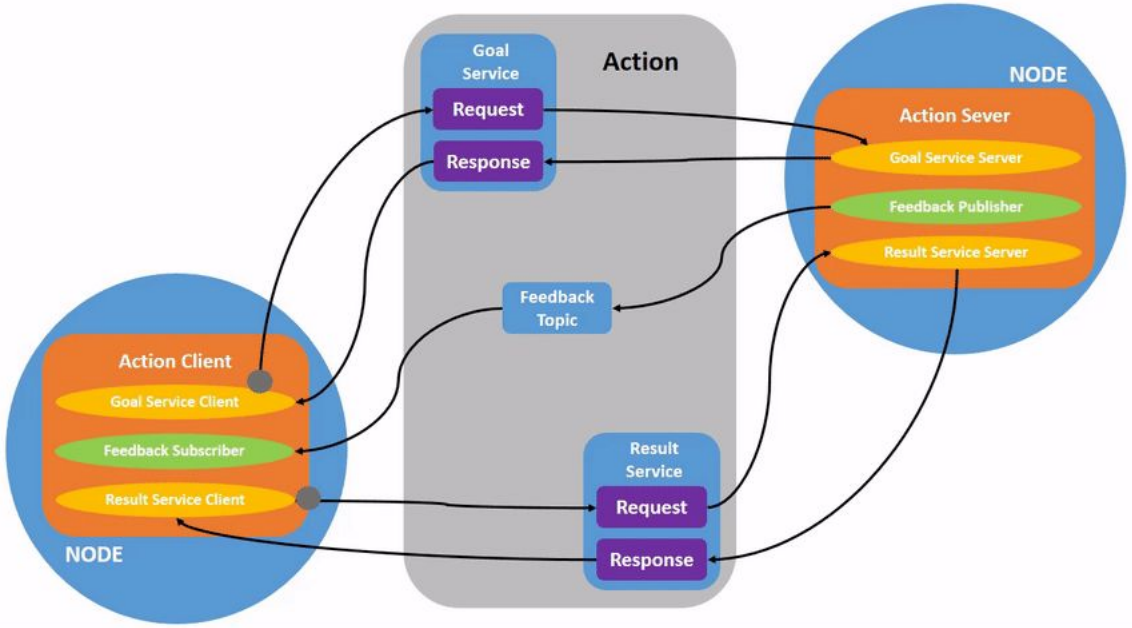
Parameters

- A parameter is a configuration value of a node. You can think of parameters as node settings.
- A node can store parameters as integers, floats, booleans, strings, and lists.
- In ROS 2, each node maintains its own parameters.



Actions

- Actions are one of the communication types in ROS 2 and are intended for long running tasks.
- They consist of three parts: a goal, feedback, and a result.
- Actions are built on topics and services.
- Their functionality is similar to services, except actions can be canceled.
- They also provide steady feedback, as opposed to services which return a single response.
- Actions use a client-server model, similar to the publisher-subscriber model.
- An “action client” node sends a goal to an “action server” node that acknowledges the goal and returns a stream of feedback and a result.



How would you structure a ROS drone like this?

Nodes:

- A node for the main body.
- One node for each motor (for a total of four).
- A node for the camera.

Topics:

- One topic where motors are subscribed, where messages to move them can be published.
- One topic for the camera to publish the image feed.
- Battery-related topic: where the status of the battery is published (the battery itself may be another node).

Services:

- Take-off service. Which allows the drone to lift from the ground.
- Land service. Which does the opposite.

More?
There is no “right” solution, it’s all about implementation.