ROS 2: Basics

Dawid Seredyński

Warsaw University of Technology

June 2026

Introduction
About ROS 2
What is ROS 2?
Distributions
Nodes
Communication
General properties
Topics
Services
Actions
Organization of software
Linux basics
Workspace
Package
rosdep
Building
Environment
Tools
ROS 2 core and stacks

© 2026 Dawid Seredyński, Warsaw University of Technology – CC BY-NC-ND 4.0, except where otherwise noted.

Aim of this lecture

Understand the concepts of *ROS 2*

- it is a huge ecosystem
- it is good to see it as a whole, before digging into details

Introduction
About ROS 2
What is ROS 2?
Distributions
Nodes
Communication
General properties
Topics
Services
Actions
Organization of software
Linux basics
Workspace
Package
rosdep
Building
Environment
Tools
ROS 2 core and stacks

© 2026 Dawid Seredyński, Warsaw University of Technology – CC BY-NC-ND 4.0, except where otherwise noted.

Learning goals

By the end of this module, you should be able to answer:

- what is *ROS 2* in general?
- what is a *ROS 2 distribution*?
- what is a *node*?
- what are communication methods in *ROS 2* and what are their purposes?
- what is Linux terminal and shell and what are the most basic commands?
- how is the software organized in *ROS 2*
- what are the most popular tools and stacks in *ROS 2*?

Introduction
About ROS 2
What is ROS 2?
Distributions
Nodes
Communication
General properties
Topics
Services
Actions
Organization of software
Linux basics
Workspace
Package
rosdep
Building
Environment
Tools
ROS 2 core and stacks

© 2026 Dawid Seredyński, Warsaw University of Technology – CC BY-NC-ND 4.0, except where otherwise noted.

The *Robot Operating System 2 (ROS 2)*★

- a set of software libraries and tools
- for building robot applications
- open source
- drivers, middleware, algorithms, developer tools

Key idea: A *ROS 2* system is a network of cooperating programs, connected through a common communication model

ROS 2 systems are:

- flexible
- heavily parametrized
- reusable

Introduction
About ROS 2
What is ROS 2?
Distributions
Nodes
Communication
General properties
Topics
Services
Actions
Organization of software
Linux basics
Workspace
Package
rosdep
Building
Environment
Tools
ROS 2 core and stacks

© 2026 Dawid Seredyński, Warsaw University of Technology – CC BY-NC-ND 4.0, except where otherwise noted.

ROS 2★ = plumbing + tools + capabilities + community

Over 2400 *packages*★ developed by:

- core team
- community

Robotics Enhancement Proposal (REP)★

Supported robots★

Introduction
About ROS 2
What is ROS 2?
Distributions
Nodes
Communication
General properties
Topics
Services
Actions
Organization of software
Linux basics
Workspace
Package
rosdep
Building
Environment
Tools
ROS 2 core and stacks

© 2026 Dawid Seredyński, Warsaw University of Technology – CC BY-NC-ND 4.0, except where otherwise noted.

ROS 2 distributions

A *distribution*★

- a versioned set of ROS *packages*
- provide a stable codebase
- changes limited to: bug fixes, non-breaking improvements

distribution	release	EOL
Rolling Ridley	–	–
Lyrical Luth	May 2026	May 2031
Kilted Kaiju	May 23, 2025	December 2026
Jazzy Jalisco	May 23, 2024	May 2029
Iron Irwini	May 23, 2023	December 4, 2024
Humble Hawksbill	May 23, 2022	May 2027
Galactic Geochelone	May 23, 2021	December 9, 2022
Foxy Fitzroy	June 5, 2020	June 20, 2023
	...	

Introduction
About ROS 2
What is ROS 2?
Distributions
Nodes
Communication
General properties
Topics
Services
Actions
Organization of software
Linux basics
Workspace
Package
rosdep
Building
Environment
Tools
ROS 2 core and stacks

© 2026 Dawid Seredyński, Warsaw University of Technology – CC BY-NC-ND 4.0, except where otherwise noted.

The ROS graph

A *ROS graph*

- a set of *ROS 2 nodes* processing data together at the same time
- connections between *nodes*

A *node*★

- responsible for a single, modular purpose (it's a design rule)
- can send and receive data from other *nodes*
- can be parametrized

A single executable (program) can contain one or more *nodes*

There are multiple communication *interfaces*

A robotic system – many *nodes* working in concert

Introduction
About ROS 2
What is ROS 2?
Distributions
Nodes
Communication
General properties
Topics
Services
Actions
Organization of software
Linux basics
Workspace
Package
rosdep
Building
Environment
Tools
ROS 2 core and stacks

© 2026 Dawid Seredyński, Warsaw University of Technology – CC BY-NC-ND 4.0, except where otherwise noted.

Parameter

A *parameter*★ is a configuration value of a *node*

Each *node* maintains its own *parameters*

Various types: integers, floats, booleans, strings, and lists

Example usage:

- configuration
- tuning
- feature flags
- calibration values
- startup/runtime settings

Introduction
About ROS 2
What is ROS 2?
Distributions
Nodes
Communication
General properties
Topics
Services
Actions
Organization of software
Linux basics
Workspace
Package
rosdep
Building
Environment
Tools
ROS 2 core and stacks

© 2026 Dawid Seredyński, Warsaw University of Technology – CC BY-NC-ND 4.0, except where otherwise noted.

Names and namespaces

A *fully-qualified name* (*FQN*):

`/<namespace_1>/<namespace_2>/.../<name>`

A *node* should have an unique *FQN*

- uniqueness is not obligatory
- however, if not unique – undefined behavior

*communication channel*s are identified by *FQN*s

Motivation for namespaces:

- multiple robots in one system
- multiple nodes of the same type

Introduction
About ROS 2
What is ROS 2?
Distributions
Nodes
Communication
General properties
Topics
Services
Actions
Organization of software
Linux basics
Workspace
Package
rosdep
Building
Environment
Tools
ROS 2 core and stacks

© 2026 Dawid Seredyński, Warsaw University of Technology – CC BY-NC-ND 4.0, except where otherwise noted.

Communication

Communication: based on **anonymous publish/subscribe**

Messages: **strongly-typed**

Communication *interfaces* (types of *communication channels*):

topic★	– for continuous data streams	**publisher** → **subscriber**
service★	– for quick request/response	**client** ⇄ **server**
action★	– for long-running preemptible behavior	**client** ⇄ **server**

parameters – for configuration, not intended for communication

Introduction
About ROS 2
What is ROS 2?
Distributions
Nodes
Communication
General properties
Topics
Services
Actions
Organization of software
Linux basics
Workspace
Package
rosdep
Building
Environment
Tools
ROS 2 core and stacks

© 2026 Dawid Seredyński, Warsaw University of Technology – CC BY-NC-ND 4.0, except where otherwise noted.

topic★ – a communication channel based on bus-like publish/subscribe

message – one piece of data sent through a *topic*

Publishers and subscribers using the same *topic* can communicate directly by sending or receiving *messages*

Properties of a *topic*

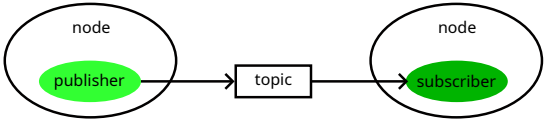
- name – identifies a *topic*
- *message* type (`.msg`) – must be consistent for all *nodes* using the same *topic*
- QoS (Quality of Service) – for each endpoint

One *node* can have multiple publishers and subscribers

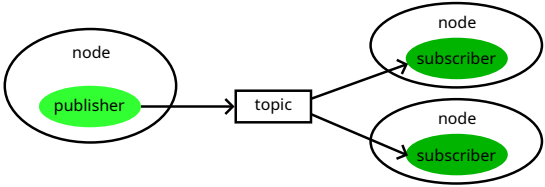
Introduction
About ROS 2
What is ROS 2?
Distributions
Nodes
Communication
General properties
Topics
Services
Actions
Organization of software
Linux basics
Workspace
Package
rosdep
Building
Environment
Tools
ROS 2 core and stacks

© 2026 Dawid Seredyński, Warsaw University of Technology – CC BY-NC-ND 4.0, except where otherwise noted.

One to one:



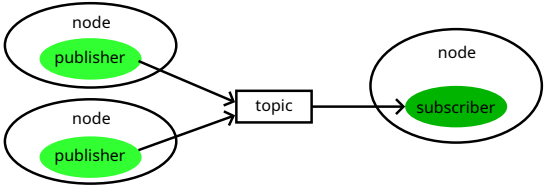
One to many:



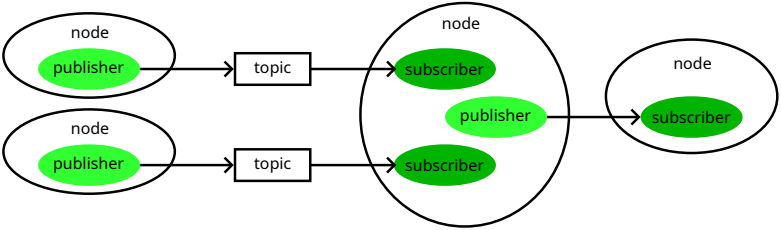
Introduction
About ROS 2
What is ROS 2?
Distributions
Nodes
Communication
General properties
Topics
Services
Actions
Organization of software
Linux basics
Workspace
Package
rosdep
Building
Environment
Tools
ROS 2 core and stacks

© 2026 Dawid Seredyński, Warsaw University of Technology – CC BY-NC-ND 4.0, except where otherwise noted.

Many to one (be careful):



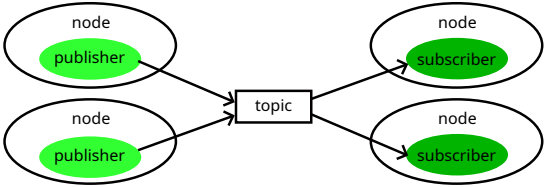
Many to one (design with a "mux"):



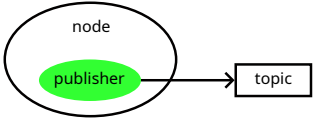
Introduction
About ROS 2
What is ROS 2?
Distributions
Nodes
Communication
General properties
Topics
Services
Actions
Organization of software
Linux basics
Workspace
Package
rosdep
Building
Environment
Tools
ROS 2 core and stacks

© 2026 Dawid Seredyński, Warsaw University of Technology – CC BY-NC-ND 4.0, except where otherwise noted.

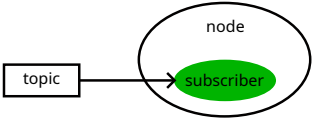
Many to many (be careful):



No subscribers:



No publishers:



Introduction
About ROS 2
What is ROS 2?
Distributions
Nodes
Communication
General properties
Topics
Services
Actions
Organization of software
Linux basics
Workspace
Package
rosdep
Building
Environment
Tools
ROS 2 core and stacks

© 2026 Dawid Seredyński, Warsaw University of Technology – CC BY-NC-ND 4.0, except where otherwise noted.

Typical use of *topics*:

- distribution of:
  - sensor data
  - robot state
  - simulation time
  - transforms, frames
  - visualization / debug data
- command streams
- pipelines
- logging and replay (`ros2 bag`)

Not recommended for:

- request-response – use a *ROS 2 service* instead
- a long-running request with feedback – use a *ROS 2 action* instead

Introduction
About ROS 2
What is ROS 2?
Distributions
Nodes
Communication
General properties
Topics
Services
Actions
Organization of software
Linux basics
Workspace
Package
rosdep
Building
Environment
Tools
ROS 2 core and stacks

© 2026 Dawid Seredyński, Warsaw University of Technology – CC BY-NC-ND 4.0, except where otherwise noted.

service★ – a remote procedure call

- request-response interactions between *nodes*
- a service has two sides:
  - client: sends request
  - server: processes request and sends back response
- a service provides data or performs an operation only when explicitly called by a client
- one to one connections:
  - one client connects to one server
  - one server can handle multiple clients independently
- one *node* can have multiple clients and servers

Recommendations:

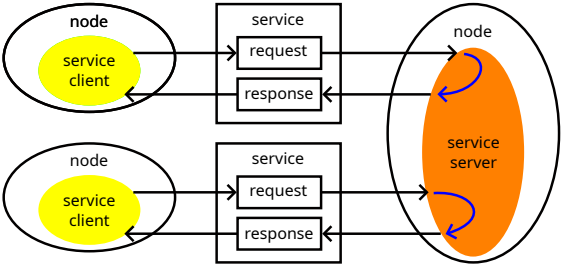
- yes: calculations, short procedures
- no: long-running tasks

Introduction
About ROS 2
What is ROS 2?
Distributions
Nodes
Communication
General properties
Topics
Services
Actions
Organization of software
Linux basics
Workspace
Package
rosdep
Building
Environment
Tools
ROS 2 core and stacks

© 2026 Dawid Seredyński, Warsaw University of Technology – CC BY-NC-ND 4.0, except where otherwise noted.

Properties of a *service*

- name – identifies a *service*; only one service server of a given name can exist
- *service* type (`.srv`) – must be consistent for all servers and clients using the same *service*
  - request type
  - response type
- QoS (Quality of Service) – for each endpoint



© 2026 Dawid Seredyński, Warsaw University of Technology – CC BY-NC-ND 4.0, except where otherwise noted.

services: blocking, waiting

A client can wait for a *service* to become available

- synchronously – blocking
- asynchronously
  - by polling

A client can wait for a *service* call to complete

- synchronously – blocking
- asynchronously
  - by polling
  - by callback

Introduction
About ROS 2
What is ROS 2?
Distributions
Nodes
Communication
General properties
Topics
Services
Actions
Organization of software
Linux basics
Workspace
Package
rosdep
Building
Environment
Tools
ROS 2 core and stacks

© 2026 Dawid Seredyński, Warsaw University of Technology – CC BY-NC-ND 4.0, except where otherwise noted.

action★ – a long-running remote procedure call with feedback and the ability to cancel or preempt (replace) a goal

- client: sends a goal (a request)
- server: realizes the goal until it is completed or canceled or preempted

Typical scenario:

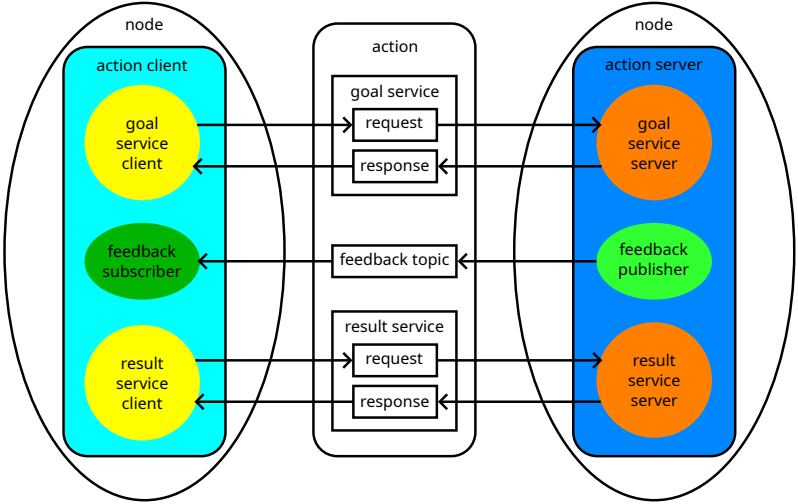
- client sends a goal
- server can accept or reject the goal
- if the goal is accepted, server:
  - realizes the goal
  - regularly sends feedback to the client
  - informs the client when finished, returns result
- client can cancel or preempt the goal

Multiple clients are allowed

- server decides how to handle multiple goal requests (reject new, preempt current, queue, execute in parallel)

Introduction
About ROS 2
What is ROS 2?
Distributions
Nodes
Communication
General properties
Topics
Services
Actions
Organization of software
Linux basics
Workspace
Package
rosdep
Building
Environment
Tools
ROS 2 core and stacks

© 2026 Dawid Seredyński, Warsaw University of Technology – CC BY-NC-ND 4.0, except where otherwise noted.



Introduction
About ROS 2
What is ROS 2?
Distributions
Nodes
Communication
General properties
Topics
Services
Actions
Organization of software
Linux basics
Workspace
Package
rosdep
Building
Environment
Tools
ROS 2 core and stacks

© 2026 Dawid Seredyński, Warsaw University of Technology – CC BY-NC-ND 4.0, except where otherwise noted.

Linux basics: the filesystem

Lot of things in Linux are represented as files

- the root (in the filesystem):
  `/`
- the home directory:
  `/home/<user_name>` or shortly: `~/`

Users can modify files in their home directories.

The rest of the tree (most of): read-only

Paths, e.g.:

- `/<dir1>/<dir2>/<file>` – an absolute path
- relative paths:
  - `<dir1>/<dir2>/<file>`
  - `./<dir1>/<dir2>/<file>`
  - `../<dir0>/<dir1>/<dir2>/<file>`

Hidden files: file name starts with . (a dot)

Introduction
About ROS 2
What is ROS 2?
Distributions
Nodes
Communication
General properties
Topics
Services
Actions
Organization of software
Linux basics
Workspace
Package
rosdep
Building
Environment
Tools
ROS 2 core and stacks

© 2026 Dawid Seredyński, Warsaw University of Technology – CC BY-NC-ND 4.0, except where otherwise noted.

Linux basics: terminals and shells

A terminal – a window/application where you type commands

A shell – a program running inside the terminal that interprets commands

- interactive command interpreter
- a scripting language

Common shell types:

- bash
- sh
- zsh

Different shells – different options, syntax, features

Introduction
About ROS 2
What is ROS 2?
Distributions
Nodes
Communication
General properties
Topics
Services
Actions
Organization of software
Linux basics
Workspace
Package
rosdep
Building
Environment
Tools
ROS 2 core and stacks

© 2026 Dawid Seredyński, Warsaw University of Technology – CC BY-NC-ND 4.0, except where otherwise noted.

A shell:

- allows to execute commands
- has its own state:
  - the current working directory (*CWD*)
  - environment variables

CDW-related commands:

- `ls` – list
- `pwd` – print working directory
- `cd` – change directory

Environment variables-related commands:

- `export <var_name>=<var_value>`
- `echo $<var_name>`
- `env`

Introduction
About ROS 2
What is ROS 2?
Distributions
Nodes
Communication
General properties
Topics
Services
Actions
Organization of software
Linux basics
Workspace
Package
rosdep
Building
Environment
Tools
ROS 2 core and stacks

© 2026 Dawid Seredyński, Warsaw University of Technology – CC BY-NC-ND 4.0, except where otherwise noted.

A command in a shell is executed as a separate child process

However, the command:
`source <script>`

- executes the `<script>` in the current terminal
- it can change the state of the terminal, e.g. set environment variables
- `.` is an alias for `source`, e.g.:
  `. <script>` is equivalent to: `source <script>`

Bash sources `~/.bashrc` file by default

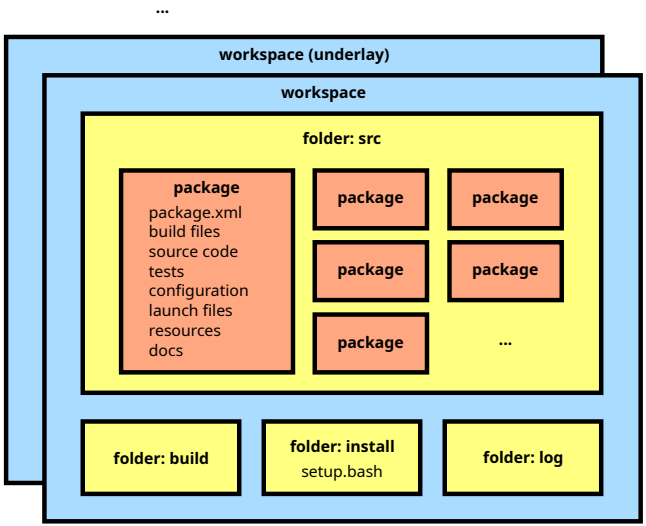
- you can add some useful `source` commands or variables there

Pro tip: to source a hidden file "`...`" in the parent directory:
`. ./../...`

Introduction
About ROS 2
What is ROS 2?
Distributions
Nodes
Communication
General properties
Topics
Services
Actions
Organization of software
Linux basics
Workspace
Package
rosdep
Building
Environment
Tools
ROS 2 core and stacks

© 2026 Dawid Seredyński, Warsaw University of Technology – CC BY-NC-ND 4.0, except where otherwise noted.

Organization of software: *workspace*



Introduction
About ROS 2
What is ROS 2?
Distributions
Nodes
Communication
General properties
Topics
Services
Actions
Organization of software
Linux basics
Workspace
Package
rosdep
Building
Environment
Tools
ROS 2 core and stacks

© 2026 Dawid Seredyński, Warsaw University of Technology – CC BY-NC-ND 4.0, except where otherwise noted.

Organization of software: *package*

A *package*★ is an organizational unit for *ROS 2* code

package.xml

- package manifest
- marks the root of a *package*
- contains meta information:
  - name
  - version
  - description
  - maintainer
  - license
  - dependencies – required to:
    - check if all required packages are installed
    - determine order of building packages
    - install missing packages with `rosdep`

Introduction
About ROS 2
What is ROS 2?
Distributions
Nodes
Communication
General properties
Topics
Services
Actions
Organization of software
Linux basics
Workspace
Package
rosdep
Building
Environment
Tools
ROS 2 core and stacks

© 2026 Dawid Seredyński, Warsaw University of Technology – CC BY-NC-ND 4.0, except where otherwise noted.

Organization of software: *package*

C++ code can be compiled into:

- executables
- *component*s (e.g. *node*s)
- libraries (linked)
- *plugin*s (dynamically loaded libraries)

Build types:

- `ament_cmake` – for C++ ROS 2 *package*s
- `ament_python` – for pure Python ROS 2 *package*s
- `cmake` – for a plain CMake project (non-ROS, not recommended)

An `ament_cmake` *package* can contain Python scripts and modules

- if so, it should depend also on `ament_cmake_python`

Introduction
About ROS 2
What is ROS 2?
Distributions
Nodes
Communication
General properties
Topics
Services
Actions
Organization of software
Linux basics
Workspace
Package
rosdep
Building
Environment
Tools
ROS 2 core and stacks

© 2026 Dawid Seredyński, Warsaw University of Technology – CC BY-NC-ND 4.0, except where otherwise noted.

rosdep

`rosdep` connects *ROS 2 package* dependency names from `package.xml` to system packages' names

- it installs the system dependencies needed by *ROS 2 packages*
- it maps dependency names from `package.xml` to the correct package names for the operating system

Introduction
About ROS 2
What is ROS 2?
Distributions
Nodes
Communication
General properties
Topics
Services
Actions
Organization of software
Linux basics
Workspace
Package
rosdep
Building
Environment
Tools
ROS 2 core and stacks

© 2026 Dawid Seredyński, Warsaw University of Technology – CC BY-NC-ND 4.0, except where otherwise noted.

Building a *workspace*

Available build tools★ – defined for each *package* individually:

- **ament_cmake** – CMake for C++ (build file: CMakeLists.txt)
- **ament_python** – setuptools for Python (build file: setup.py)
- other

Meta-build tool: **colcon** (*workspace* level)

- "collective construction"
- building in correct dependency order
- cheat sheet★ – very useful!

The command:

```
colcon build --symlink-install
```


Introduction
About ROS 2
What is ROS 2?
Distributions
Nodes
Communication
General properties
Topics
Services
Actions
Organization of software
Linux basics
Workspace
Package
rosdep
Building
Environment
Tools
ROS 2 core and stacks

© 2026 Dawid Seredyński, Warsaw University of Technology – CC BY-NC-ND 4.0, except where otherwise noted.

`colcon` searches the current directory for *packages*

- every folder with a properly formatted `package.xml` is a *package*
- it searches the whole directory tree below *CWD*
- it creates `build`, `install` and `log` directories in the *CWD*
- always check where you are (`pwd` command)

Symlink install:

- non-compiled installed files are symlinked where possible
- Python files, *launch files*, configs, and resources are easier to edit
- rebuild, if:
  - added or removed directories or files
  - changed C++ sources
  - changed `CMakeLists.txt` or `package.xml`

Introduction
About ROS 2
What is ROS 2?
Distributions
Nodes
Communication
General properties
Topics
Services
Actions
Organization of software
Linux basics
Workspace
Package
rosdep
Building
Environment
Tools
ROS 2 core and stacks

© 2026 Dawid Seredyński, Warsaw University of Technology – CC BY-NC-ND 4.0, except where otherwise noted.

Programming languages

Maintained by the core *ROS 2* team:

- C++
- Python

Maintained by the *ROS 2* community:

- Ada
- C (micro-ROS)
- JVM and Android Java and Android bindings
- .NET Core, UWP and C#
- Node.js (JavaScript API)
- Rust
- Flutter and Dart Flutter and Dart bindings

Niche languages: usually in an `ament_cmake` *package*

Introduction
About ROS 2
What is ROS 2?
Distributions
Nodes
Communication
General properties
Topics
Services
Actions
Organization of software
Linux basics
Workspace
Package
rosdep
Building
Environment
Tools
ROS 2 core and stacks

© 2026 Dawid Seredyński, Warsaw University of Technology – CC BY-NC-ND 4.0, except where otherwise noted.

Environment

To use ROS you need to load ROS *workspace*-related environment variables★

- `source /opt/ros/jazzy/setup.bash` (the lowest *workspace*)
- `. ./install/setup.bash` (your *workspace*)

`setup.<shell_type>` – various supported shells; use the compatible one

It sets the following variables:

- `ROS_DISTRO`, `ROS_VERSION`, `ROS_PYTHON_VERSION` – versions
- `ROS_PACKAGE_PATH` – for finding *packages*
- `ROS_AUTOMATIC_DISCOVERY_RANGE` – for discovery
- `AMENT_PREFIX_PATH`, `CMAKE_PREFIX_PATH` – for building
- `PYTHONPATH`, `LD_LIBRARY_PATH` – for finding Python modules and libraries

You need to source ROS in every terminal where you want to use ROS

Introduction
About ROS 2
What is ROS 2?
Distributions
Nodes
Communication
General properties
Topics
Services
Actions
Organization of software
Linux basics
Workspace
Package
rosdep
Building
Environment
Tools
ROS 2 core and stacks

© 2026 Dawid Seredyński, Warsaw University of Technology – CC BY-NC-ND 4.0, except where otherwise noted.

Beware of LOCALE

If you are:

Polish, German, French, Spanish, Latin American, Italian, Portuguese, Brasilian, Dutch, Czech, Slovak, Slovenian, Croatian, Hungarian, Romanian, Bulgarian, Greek, Danish, Finnish, Swedish, Norwegian, Russian, Ukrainian, Lithuanian, Latvian, Estonian, Turkish, Indonesian, Vietnamese,

your digital separator may be , (coma) instead of . (dot), so:

- this is a number: 1,234 (it's a double)
- and this is not: 1.234 (it's a string)

You will get a YAML/URDF/SDF/parameter parsing surprise! (very unexpected)

ROS 2 will throw an error, e.g.:

```
[rviz2-14] [ERROR] [...] [moveit...]: Exception caught while processing action
'loadRobotModel': Wrong parameter type, parameter 'max_velocity' is of type 'double',
setting it to 'string' is not allowed.
```

Solution: `export LC_NUMERIC=C` (or any other locale that uses "." like en_US.UTF-8)

Introduction
About ROS 2
What is ROS 2?
Distributions
Nodes
Communication
General properties
Topics
Services
Actions
Organization of software
Linux basics
Workspace
Package
rosdep
Building
Environment
Tools
ROS 2 core and stacks

© 2026 Dawid Seredyński, Warsaw University of Technology – CC BY-NC-ND 4.0, except where otherwise noted.

rviz2

A 3D visualization tool

rviz2 can:

- visualize *topic* of various message types
- publish commands

rviz2 is *plugin*-based

- powerful *plugin*s for various frameworks (e.g. for *MoveIt*, Nav2)
- you can write your own *plugin*s

It is <u>not</u> a simulator

Command:

```
ros2 run rviz2 rviz2
```


Introduction
About ROS 2
What is ROS 2?
Distributions
Nodes
Communication
General properties
Topics
Services
Actions
Organization of software
Linux basics
Workspace
Package
rosdep
Building
Environment
Tools
ROS 2 core and stacks

© 2026 Dawid Seredyński, Warsaw University of Technology – CC BY-NC-ND 4.0, except where otherwise noted.

rqt_graph

Displays the *ROS 2* graph

Not all connections are visible:

- *node – plugin*
- *ros2_control*
- *services*
- *actions* that were not called

For big, complex systems – limited utility

- good for inspecting topic connections: mismatched names, unconnected, multiple publishers, etc.

Command:
`rqt_graph`

Introduction
About ROS 2
What is ROS 2?
Distributions
Nodes
Communication
General properties
Topics
Services
Actions
Organization of software
Linux basics
Workspace
Package
rosdep
Building
Environment
Tools
ROS 2 core and stacks

© 2026 Dawid Seredyński, Warsaw University of Technology – CC BY-NC-ND 4.0, except where otherwise noted.

CLI – command line interface

Some of (many) useful commands:

- `ros2 -h`
- `ros2 <command> -h`
- `ros2 doctor --report`
- `ros2 run <package_name> <executable_name>`
- `ros2 launch <package_name> <launch_file>`
- `ros2 node list`
- `ros2 node info <package_name>`
- `ros2 topic list -t`
- `ros2 topic info <topic_name>`
- `ros2 topic echo <topic_name>`
- `ros2 topic pub <topic_name> <msg_type> <yaml_msg_data>`
- `ros2 interface show <msg_type>`

Auto-completion: [tab][tab]

cheat sheet★ – very useful!

Introduction
About ROS 2
What is ROS 2?
Distributions
Nodes
Communication
General properties
Topics
Services
Actions
Organization of software
Linux basics
Workspace
Package
rosdep
Building
Environment
Tools
ROS 2 core and stacks

© 2026 Dawid Seredyński, Warsaw University of Technology – CC BY-NC-ND 4.0, except where otherwise noted.

ROS 2 core and stacks

ROS 2 core

- API, *ROS 2* client library for C++ (rclcpp), Python (rclpy): *node*s, *topic*s, *service*s, *action*s, *parameter*s, launch, executors, lifecycle
- tf2, CLI

ROS 2 ecosystem stacks

- Nav2
- *MoveIt*
- *ros2_control*
- support for *Gazebo Sim* (ros_gz)
- SLAM / perception / add-ons

Introduction
About ROS 2
What is ROS 2?
Distributions
Nodes
Communication
General properties
Topics
Services
Actions
Organization of software
Linux basics
Workspace
Package
rosdep
Building
Environment
Tools
ROS 2 core and stacks

© 2026 Dawid Seredyński, Warsaw University of Technology – CC BY-NC-ND 4.0, except where otherwise noted.

Nav2★ – mobile robot navigation

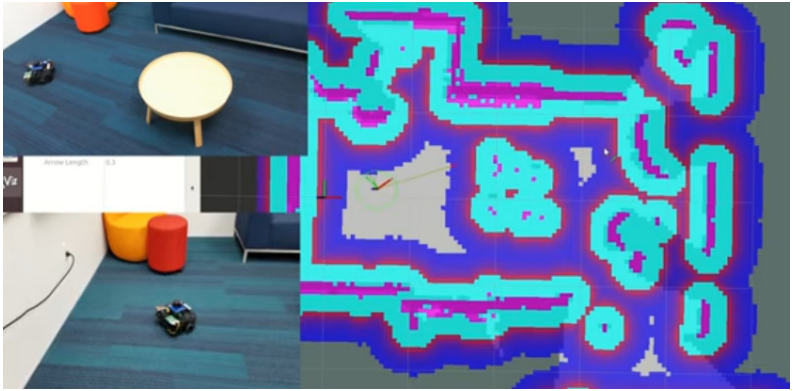


image source: https://docs.nav2.org/tutorials/docs/navigation2_on_real_turtlebot3.html

maps, localization/planning interfaces, controllers, recovery/behavior logic, integration patterns

Introduction
About ROS 2
What is ROS 2?
Distributions
Nodes
Communication
General properties
Topics
Services
Actions
Organization of software
Linux basics
Workspace
Package
rosdep
Building
Environment
Tools
ROS 2 core and stacks

© 2026 Dawid Seredyński, Warsaw University of Technology – Images excluded from license; © respective owner.

slam_toolbox★

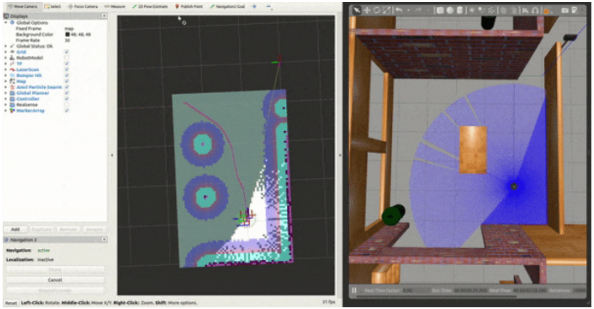


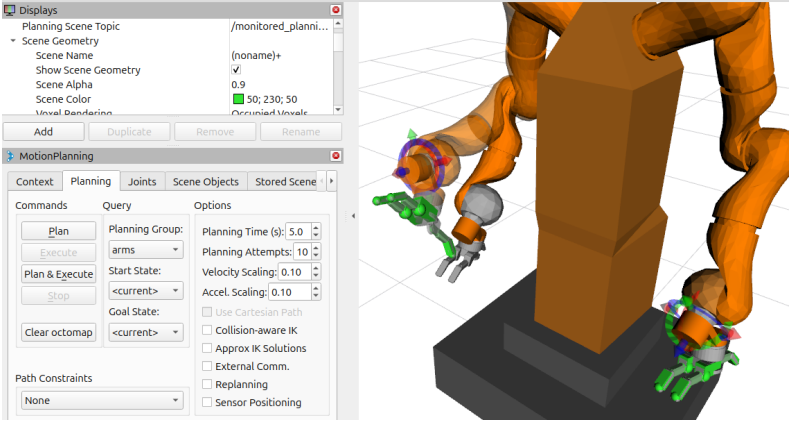
image source: https://docs.nav2.org/tutorials/docs/navigation2_with_slam.html

One of the most common *ROS 2* stacks for 2D SLAM (Simultaneous Localization and Mapping)

Introduction
About ROS 2
What is ROS 2?
Distributions
Nodes
Communication
General properties
Topics
Services
Actions
Organization of software
Linux basics
Workspace
Package
rosdep
Building
Environment
Tools
ROS 2 core and stacks

© 2026 Dawid Seredyński, Warsaw University of Technology – Images excluded from license; © respective owner.

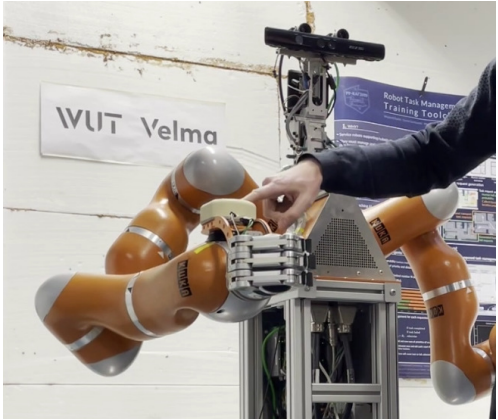
MoveIt★ – manipulation



kinematics, motion planning, planning scene / collision environment, trajectory processing, hybrid planning, multi-stage task planning

© 2026 Dawid Seredyński, Warsaw University of Technology – CC BY-NC-ND 4.0, except where otherwise noted.

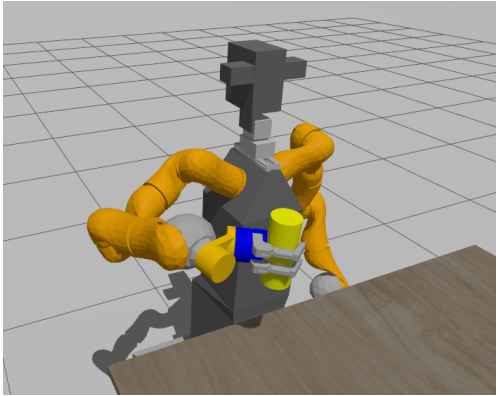
ros2_control★ – the control layer



hardware abstraction, controller management, reusable controllers, real-time operation

© 2026 Dawid Seredyński, Warsaw University of Technology – CC BY-NC-ND 4.0, except where otherwise noted.

Gazebo Sim + ros_gz



realistic physics (dynamics, contacts, forces), sensors (cameras, lidars, F/T), integration (ROS 2, ros2_control)

Introduction
About ROS 2
What is ROS 2?
Distributions
Nodes
Communication
General properties
Topics
Services
Actions
Organization of software
Linux basics
Workspace
Package
rosdep
Building
Environment
Tools
ROS 2 core and stacks

© 2026 Dawid Seredyński, Warsaw University of Technology – CC BY-NC-ND 4.0, except where otherwise noted.

image_pipeline★

image source: https://docs.ros.org/en/rolling/p/camera_calibration/doc/tutorial_stereo.html

Camera pre-processing: callibration, image rectification, stereo disparity, stereo point cloud generation

It is a very common foundation for camera-based robotics pipelines

Introduction
About ROS 2
What is ROS 2?
Distributions
Nodes
Communication
General properties
Topics
Services
Actions
Organization of software
Linux basics
Workspace
Package
rosdep
Building
Environment
Tools
ROS 2 core and stacks

© 2026 Dawid Seredyński, Warsaw University of Technology – Images excluded from license; © respective owner.

Questions?

Introduction
About ROS 2
What is ROS 2?
Distributions
Nodes
Communication
General properties
Topics
Services
Actions
Organization of software
Linux basics
Workspace
Package
rosdep
Building
Environment
Tools
ROS 2 core and stacks

© 2026 Dawid Seredyński, Warsaw University of Technology – CC BY-NC-ND 4.0, except where otherwise noted.