

Getting Started with ROS 2 Jazzy

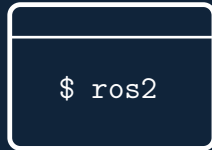
Installation, Development Tools, and First Examples

Install Configure Run Inspect

Dr. Kaveh Hooshmandi

Primary academic email: k.hooshmandi@arakut.ac.ir

Secondary email: Hooshmandi.kaveh@gmail.com



Today's route: install, prepare, run, inspect



By the end of the session, you can

Check the installation in a fresh shell; run two communicating programs; locate their nodes, topics and message type; and drive and inspect a turtle.

Key idea

Target: Ubuntu 24.04 (Noble) with ROS 2 Jazzy. The maintained installation guide supplies the repository setup; the slides explain what each step accomplishes.

Install 1/3: confirm the Ubuntu version

Target system

Ubuntu **24.04 (Noble)** with ROS 2 **Jazzy**. Use the matching guide if your Ubuntu release differs.

```
. /etc/os-release
echo "$VERSION_ID $VERSION_CODENAME"
dpkg --print-architecture
```

Before installing

Connect to the internet, refresh Ubuntu packages, and check that the official guide supports your architecture.

Key idea

Look for 24.04 noble. The architecture result, often amd64 or arm64, also guides your later VS Code download.

Open the official Jazzy installation guide

Install 2/3: repository, then packages

1. On the [official Ubuntu installation page](#), follow the current **locale**, **Universe**, and **ros2-apt-source** steps in order.
2. After the ROS repository is configured, refresh the package list and install the desktop edition:

```
sudo apt update  
sudo apt install ros-jazzy-desktop ros-dev-tools
```

Why the link matters

The repository setup commands and signing package can change. Copy that step from the maintained ROS documentation rather than an old slide.

`ros-jazzy-desktop` provides common desktop tools and demos; `ros-dev-tools` adds development utilities. docs.ros.org/en/jazzy/Installation/Ubuntu-Install-Debs.html

Install 3/3: activate and verify

```
source /opt/ros/jazzy/setup.bash
echo "$ROS_DISTRO"      # expected: jazzy
ros2 --help              # ROS commands appear
```

Every new terminal

Run `source /opt/ros/jazzy/setup.bash` again, or add that line once to `~/.bashrc`.

Full verification

The [official guide](#) includes a talker/listener test and troubleshooting steps.

Why source is needed in a new shell

Installed files

The packages live under `/opt/ros/jazzy`. Installing them does not automatically teach every open shell where to find ROS commands and packages.

```
source /opt/ros/jazzy/setup.bash
echo "$ROS_DISTRO" # jazzy
command -v ros2
```

What changes?

`source` updates environment variables in the *current shell*. It does not install ROS again. A second terminal pane has a separate shell and may need the same command.

Key idea

`ros2 -help` checks the command-line tool. The live publisher/subscriber exercise later checks communication as well.

Terminator 1/2: install and open

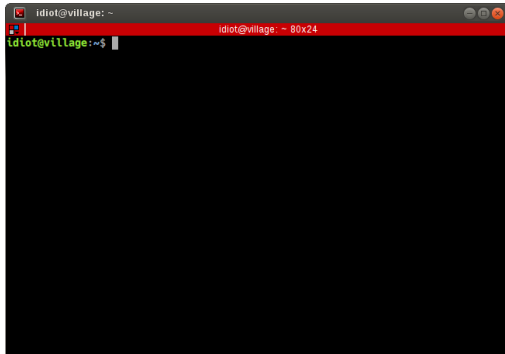
```
sudo apt install -y terminator  
terminator
```

Why use it?

Terminator keeps several terminal panes in one window. A running program can remain visible while you type commands in another pane.

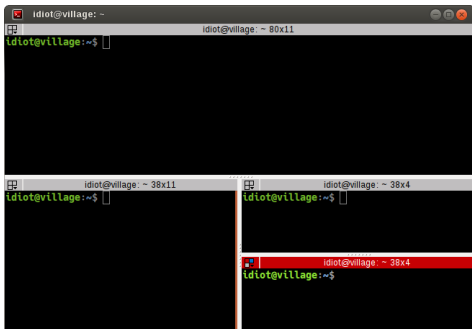
Key idea

The first command installs the application; the second opens it. You can also launch it from Ubuntu's applications menu.



Terminator window; official Terminator documentation

Terminator 2/2: split panes and prepare each shell



Split layout; official Terminator documentation

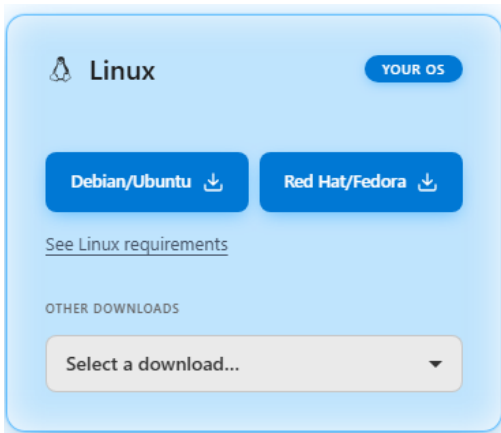
Ctrl+Shift+O	Top/bottom split
Ctrl+Shift+E	Side-by-side split
Ctrl+Tab	Next pane
Ctrl+Shift+W	Close pane

Each pane is a new shell

Right-click for split options. Load the ROS environment in each new pane if it is not already in `.bashrc`.

```
source /opt/ros/jazzy/setup.bash
echo "$ROS_DISTRO" # expected: jazzy
```

VS Code 1/4: download and install



Official VS Code download page

Choose the Ubuntu package

Open `code.visualstudio.com/download`;
choose the Linux .deb file matching your
CPU architecture and save it in Downloads.

```
cd ~/Downloads
ls code_*.deb
sudo apt install ./code_*.deb
code --version
```

If several `code_*.deb` files exist, install the exact downloaded filename after `./`.

VS Code 2/4: open your workspace

```
mkdir -p ~/ros2_ws/src  
cd ~/ros2_ws  
code .
```

Meaning

`code .` opens the current folder. The dot means *this directory*.

Use the built-in terminal

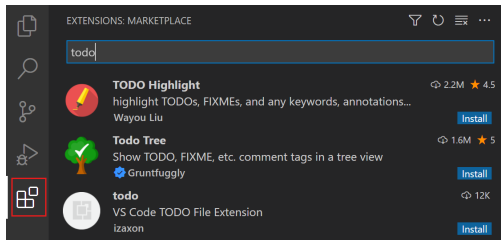
Choose **Terminal** → **New Terminal**. It opens a shell in the project folder, so you can edit files and run commands in one window.

```
source /opt/ros/jazzy/setup.bash  
echo "$ROS_DISTRO" # jazzy
```

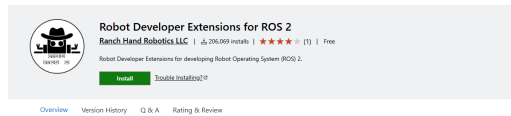
Key idea

A newly opened integrated terminal needs the setup line unless your `.bashrc` already loads it. The `ros2_ws` folder is a practice workspace.

VS Code 3/4: find and install extensions



Sample Extensions view: `Ctrl+Shift+X`



ROS 2 listing; Visual Studio Marketplace

Replace the sample search with these exact IDs; check the publisher, then click **Install**.

Role	Extension ID	Publisher
ROS 2	Ranch-Hand-Robotics.rde-ros-2	Ranch Hand Robotics
Python	ms-python.python	Microsoft
C++	ms-vscode.cpptools	Microsoft (optional)

VS Code 4/4: verify the integrated setup

In **Terminal** → **New Terminal**, run:

```
source /opt/ros/jazzy/setup.bash
echo "$ROS_DISTRO"      # jazzy
ros2 --help             # ROS commands
code --version          # VS Code version
code --list-extensions  # installed extension IDs
```

If an extension is missing

In the Extensions view, search its ID from the previous slide and click **Install**. You may also use `code -install-extension ID` with that exact ID.

Key idea

You are ready when jazzy appears, `ros2 -help` works, and the ROS 2 and Python extensions appear in the installed list. Add C/C++ when needed.

System installation and your workspace

Installed by Ubuntu packages

`/opt/ros/jazzy` contains the ROS tools and installed examples. The setup script exposes them to the current shell.

```
source /opt/ros/jazzy/setup.bash
ros2 pkg executables demo_nodes_cpp
```

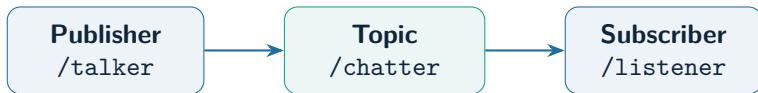
Owned by you

`~/ros2_ws/src` is where your future source packages will go. Today it may be empty: the demo packages are already installed system-wide.

Key idea

After building a workspace in a later lesson, its `install/setup.bash` will also be sourced. That file does not exist before the first build.

The small model behind the examples



Four words to read the next slides

A **package** groups software; an **executable** starts a program; a running **node** participates in ROS; a **topic** carries messages with a declared **type**.

Read this command

`ros2 run demo_nodes_cpp talker`: run the talker executable from the `demo_nodes_cpp` package. Its node publishes text messages.

Talker and listener: the first live example

Terminal A: C++ talker

```
source /opt/ros/jazzy/setup.bash  
ros2 run demo_nodes_cpp talker
```

Terminal B: Python listener

```
source /opt/ros/jazzy/setup.bash  
ros2 run demo_nodes_py listener
```

Key idea

Start Pane A first, then Pane B. The talker prints `Publishing`; the listener prints `I heard`. Keep both running, and open a third pane for inspection. This checks the C++ and Python APIs together.

Official ROS 2 Jazzy installation examples

Find the two running nodes

```
source /opt/ros/jazzy/setup.bash
ros2 node list
ros2 node info /listener
```

Read the results

ros2 node list should include /talker and /listener. ros2 node info /listener lists its subscriptions; look for /chatter.

Key idea

A node is a running participant. This list describes *what is running now*; close the listener and its name disappears. Keep both examples running for the next checks.

Inspect the topic, type and actual data

```
ros2 topic list -t  
ros2 topic info /chatter  
ros2 interface show std_msgs/msg/String  
ros2 topic echo /chatter
```

Press Ctrl+C to stop only the echo command.

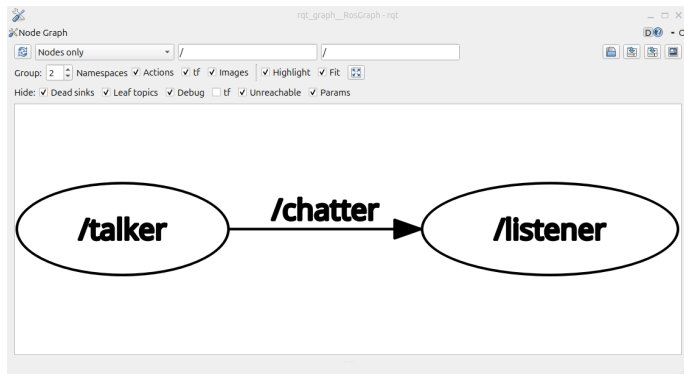
Read the results

/chatter has type `std_msgs/msg/String`. The type contains one field, string data. `topic info` should report one publisher and one subscription.

What echo adds

The graph shows connections. `echo` prints the actual data: field; watch its counter change.

See the connections with `rqt_graph`



MOGI-ROS example screenshot

Leave the talker and listener running; the graph shows their live connection.

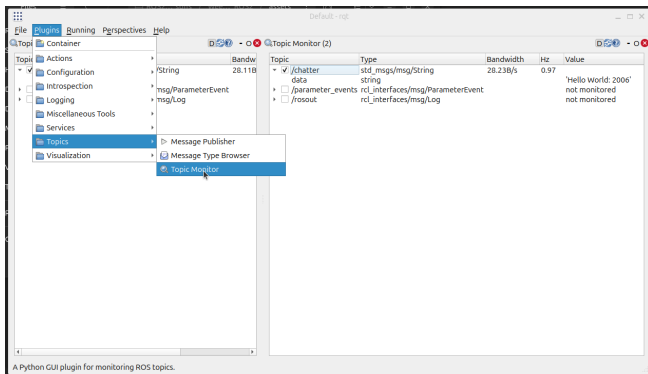
Launch

`rqt_graph`

Read the arrow

`/talker` publishes on `/chatter`; `/listener` subscribes.

Monitor a message in rqt



MOGI-ROS example screenshot

Terminal alternative: `ros2 topic echo /chatter`. Stop the demo nodes with `Ctrl+C`.

Open the view

Run rqt. Select **Plugins**
→ **Topics** → **Topic Monitor**.

What to watch

Tick `/chatter`; expand `data` to see the latest Hello World message.

Turtlesim 1/3: drive the turtle

Install if needed

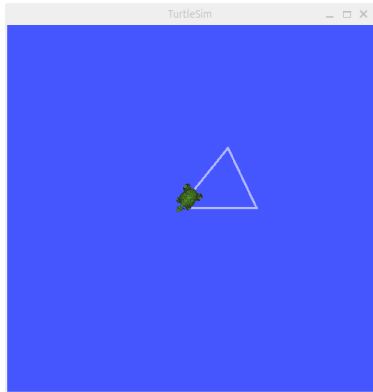
```
sudo apt install ros-jazzy-turtlesim
```

Pane A: simulator

```
source /opt/ros/jazzy/setup.bash  
ros2 run turtlesim turtlesim_node
```

Pane B: keyboard control

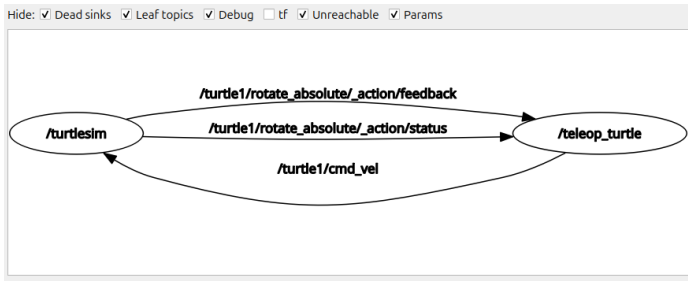
```
source /opt/ros/jazzy/setup.bash  
ros2 run turtlesim turtle_teleop_key
```



MOGI-ROS tutorial screenshot

Click Pane B and use the arrow keys. Stop either node with **Ctrl+C**.

Turtlesim 2/3: inspect the live connection



MOGI-ROS graph screenshot

```
ros2 node list
ros2 topic list
ros2 topic info /turtle1/
    cmd_vel
rqt_graph
```

Read the arrow

/teleop_turtle sends velocity commands on /turtle1/cmd_vel; /turtlesim receives them.

Focus on the /turtle1/cmd_vel arrow. The other lines show action status and feedback; those mechanisms will be studied later.

Turtlesim 3/3: inspect a velocity message

```
ros2 topic info /turtle1/cmd_vel  
ros2 interface show geometry_msgs/msg/Twist  
ros2 topic echo /turtle1/cmd_vel
```

Key idea

Keep the simulator and keyboard node running. In a third pane, start echo; focus the keyboard pane and press an arrow. Return to the third pane to read the values.

What the type means

Twist contains two 3D vectors: linear and angular. In this 2D example, linear.x controls forward motion and angular.z controls turning.

Predict, then observe

Press Up and then Left. Which field changes? Compare your prediction with the messages printed by echo.

When a step fails: inspect the smallest thing first

Symptom	First check	Next action
CLI cannot find <code>ros2</code>	<code>echo "\$ROS_DISTRO"</code> in this pane	Source Jazzy in this pane (slide 6).
No <code>/talker</code> in node list	Is the talker still running?	Restart its <code>ros2 run</code> command.
No messages on <code>/chatter</code>	Is the talker still running?	Restart the talker; inspect the topic again.
Graph looks empty	Are nodes running?	Refresh <code>rqt_graph</code> ; check filters.
Turtle does not move	Which pane has keyboard focus?	Click the teleop pane; press an arrow.

A blank `ROS_DISTRO` in one pane means its shell has not loaded the environment; it does not mean the installation has vanished.

Your independent five-minute check

1. Open fresh Terminator panes. Make `echo "$ROS_DISTRO"` print jazzy in each pane you use.
2. Run the talker and listener. Identify the node that publishes and the one that subscribes.
3. Find `/chatter`, its message type and one live value with the CLI.
4. Run the turtle and teleop nodes. Use `rqt_graph` to locate `/turtle1/cmd_vel`.

Expected evidence

The listener prints `I heard`; `topic info` shows a publisher and subscription; the turtle moves when the teleop pane receives an arrow key.

End each running program with `Ctrl+C`. The source workspace `~/ros2_ws/src` is ready for the next lesson on creating packages.