

# EECS 367 & ROB 320 Lab ROS Tutorial

# Why is robotics so hard?





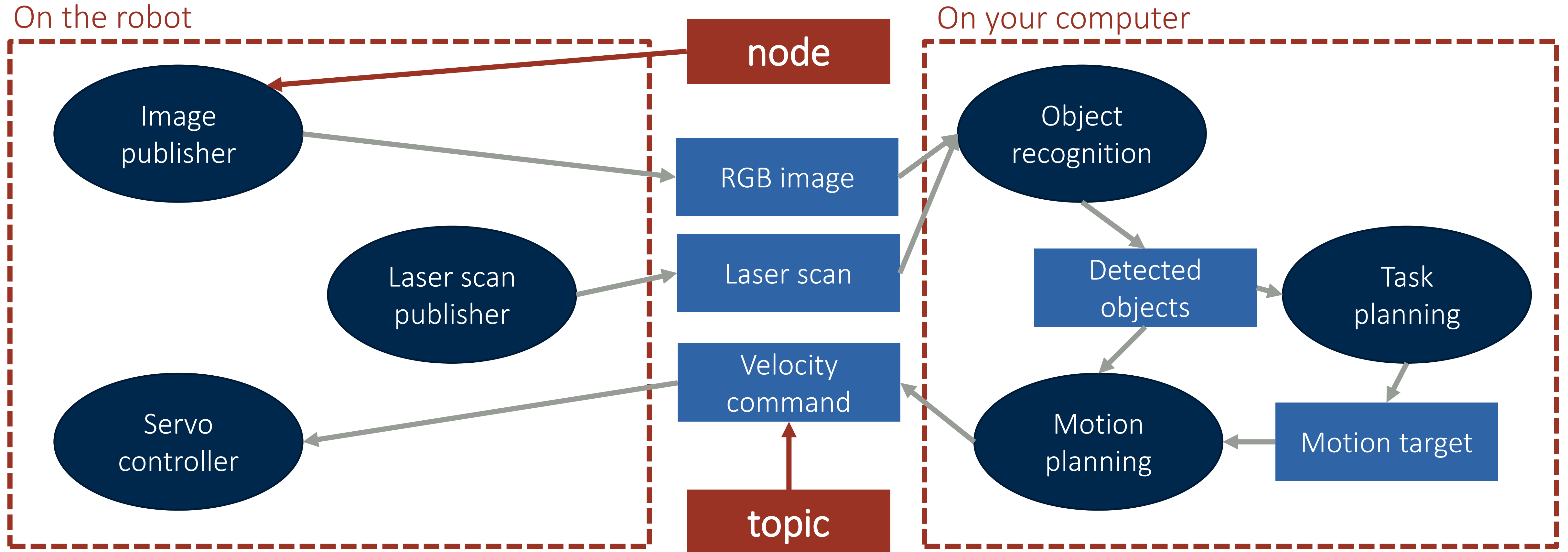
# What is ROS?

“The **Robot Operating System** (ROS) is a flexible framework for writing robot software. It is a collection of **tools, libraries, and conventions** that aim to simplify the task of creating complex and robust robot behavior across a wide variety of robotic platforms.” – [ros.org/about-ros](http://ros.org/about-ros)

ROS is a **software ecosystem** that gives you access to a lot of great tools and libraries **if you play by its rules.** - me

# Middleware

Core ROS feature is its **message passing interface** and related tools



# Tools, Libraries, Conventions

## Tools

Software and package management tools (roslaunch, rosdep, catkin)

Robotics tools (Gazebo, RViz, MoveIt!)

## Libraries

ROS libraries (roscpp)

Utilities (tf2)

## Conventions

Standard message types (std\_msgs, geometry\_msgs)

Robot description standard (URDF)

# Today: ROS Basics

1. Working with ROS nodes, topics, and messages
2. How to write and build your own ROS node
3. Publishers and subscribers
4. Demo of simple publisher/subscriber nodes
5. ROS tools to look into

# Today: ROS Basics

1. Working with ROS nodes, topics, and messages
2. How to write and build your own ROS node
3. Publishers and subscribers
4. Demo of simple publisher/subscriber nodes
5. ROS tools to look into



# Installing ROS

To work with the Fetch, install ROS Melodic

For best results, use Ubuntu 18.04  
Save yourself a lot of frustration!

Installation tutorials on [wiki.ros.org](http://wiki.ros.org)

Install relevant Fetch packages

ros-melodic-fetch-ros

ros-melodic-fetch-gazebo

Distro	Release date	Poster	Tuturtle, turtle in tutorial	EOL date
ROS Noetic Ninjemys (Recommended)	May 23rd, 2020			May, 2025 (Focal EOL)
ROS Melodic Morenia	May 23rd, 2018			May, 2023 (Bionic EOL)
ROS Lunar Loggerhead	May 23rd, 2017			May, 2019
ROS Kinetic Kame	May 23rd, 2016			April, 2021 (Xenial EOL)
ROS Jade Turtle	May 23rd, 2015			May, 2017
ROS Indigo Igloo	July 22nd, 2014			April, 2019 (Trusty EOL)



# ROS Nodes

A **ROS node** is essentially a running instance of an executable from a ROS package

To see running nodes: `$ rosnode list`

ROS nodes use ROS client library to communicate with other nodes

In C++: `roscpp`

In Python: `rospy`

ROS nodes can...

Publish a ROS topic, subscribe to a ROS topic

Provide a ROS service, use a ROS service

# ROS Topics and Messages

A **ROS topic** is a data channel through which only one type of data can be sent

Data is contained in **messages**

Each topic has a single associated message type

A **publisher** node sends messages on a topic; a **subscriber** node receives them

Some useful commands for working with topics and messages include:

List current topics: `$ rostopic list`

Display messages on a topic: `$ rostopic echo /topic_name`

Display message type structure: `$ rosmmsg show msg_pkg/msg_name`

See publisher/subscriber graph: `$ rosrn rqt_graph rqt_graph`

# Today: ROS Basics

1. Working with ROS nodes, topics, and messages
2. **How to write and build your own ROS node**
3. Publishers and subscribers
4. Demo of simple publisher/subscriber nodes
5. ROS tools to look into



# ROS Build System

To easily write ROS nodes and compile them against the necessary dependencies, use **catkin**

- Build system for ROS

- Extension of cmake, works similarly

A catkin package must have package.xml and CMakeLists.txt and be located in its own folder within a **catkin workspace**

Recommended: Install catkin\_tools package for improved command-line interface...

- Create a catkin workspace in current folder: `$ catkin init`

- Build all packages in your workspace: `$ catkin build`

# Development Environment Setup

\* Assumes you have already installed ROS

```
$ source /opt/ros/melodic/setup.bash
```

Can put this line in your .bashrc or similar if frequently developing ROS packages

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

All packages need to be located in src subdirectory within workspace

```
$ cd ~/catkin_ws
```

```
$ catkin init
```

```
$ catkin build
```

Builds all packages in workspace by default

```
$ source ~/catkin_ws/devel/setup.bash
```

Can put this line in your .bashrc or similar if frequently using packages from this workspace

# Creating A Package

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

Must create package within src subdirectory

```
$ catkin create pkg autorob_tutorial
```

Should now have autorob\_tutorial directory with CMakeLists.txt and package.xml files

```
$ cd ~/autorob_tutorial
```

```
$ mkdir scripts
```

```
$ cd scripts
```

We will write example node in Python, so our code goes in the scripts subdirectory



# Demo Code

emgoeddel / **autorob\_tutorial**

Unwatch

1

Star

0

Fork

0

<> Code

Issues

Pull requests

Actions

Projects

Wiki

Security

Insights

Settings

master1 branch0 tags

Go to fileAdd fileCode

emgoeddel Merge branch 'master' of github.com:emgoeddel/autorob\_tutorial0827f1e 17 hours ago4 commits

scripts	Initial commit.	17 hours ago
CMakeLists.txt	Initial commit.	17 hours ago
README.md	Create README.md	17 hours ago
package.xml	Updates the package.xml	17 hours ago

README.md

# autorob\_tutorial

This repo is for the AutoRob course ROS tutorial. It provides three basic ROS nodes:

1. simple\_subscriber demonstrates how to subscribe to a topic
2. simple\_publisher demonstrates how to publish to a topic
3. fetch\_controller puts the two together to move the fetch around

About

Basic ROS nodes in Python

Readme

Releases

No releases published  
Create a new release

Packages

No packages published  
Publish your first package

Languages

CMake70.7%Python29.3%

[https://github.com/emgoeddel/autorob\\_tutorial](https://github.com/emgoeddel/autorob_tutorial)

# Python Nodes

CMakeLists.txt

```
156  ## Mark executable scripts (Python etc.) for installation
157  ## in contrast to setup.py, you can choose the destination
158  catkin_install_python(PROGRAMS
159    scripts/simple_subscriber.py
160    scripts/simple_publisher.py
161    scripts/fetch_controller.py
162    DESTINATION ${CATKIN_PACKAGE_BIN_DESTINATION}
163  )
```



```
$ rosrun autorob_tutorial simple_subscriber.py
```

# Today: ROS Basics

1. Working with ROS nodes, topics, and messages
2. How to write and build your own ROS node
- 3. Publishers and subscribers**
4. Demo of simple publisher/subscriber nodes
5. ROS tools to look into



# Publish / Subscribe

A **publisher** sends messages on a topic

Does some processing and publishes messages as output

A **subscriber** receives messages from a topic

Takes the messages as input for processing

Requires a designated callback function to respond to messages

If a response or action is required, use a **service** instead

# Simple Subscriber

simple\_subscriber.py

```
1  #!/usr/bin/env python
2
3  import rospy
4  from sensor_msgs.msg import LaserScan
5
6  def subscriber_callback(data):
7      rospy.loginfo(rospy.get_caller_id() + " node received laser scan message!")
8      return
9
10 def node_init():
11     # Advertise a new node named 'simple_subscriber'
12     rospy.init_node('simple_subscriber')
13
14     # Register 'simple_subscriber' node as a subscriber node
15     # Parameters:
16     #   'base_scan'      specifies that this node will subscribe to the 'base_scan' topic
17     #   LaserScan        specifies that the topic's message type is sensor_msgs/LaserScan
18     #   listener_callback specifies the callback function used when a 'base_scan' message is received
19     rospy.Subscriber('base_scan', LaserScan, subscriber_callback)
20
21     # Keep node running until node is exited
22     rospy.spin()
23
24 if __name__ == '__main__':
25     node_init()
```

# Simple Publisher

simple\_publisher.py

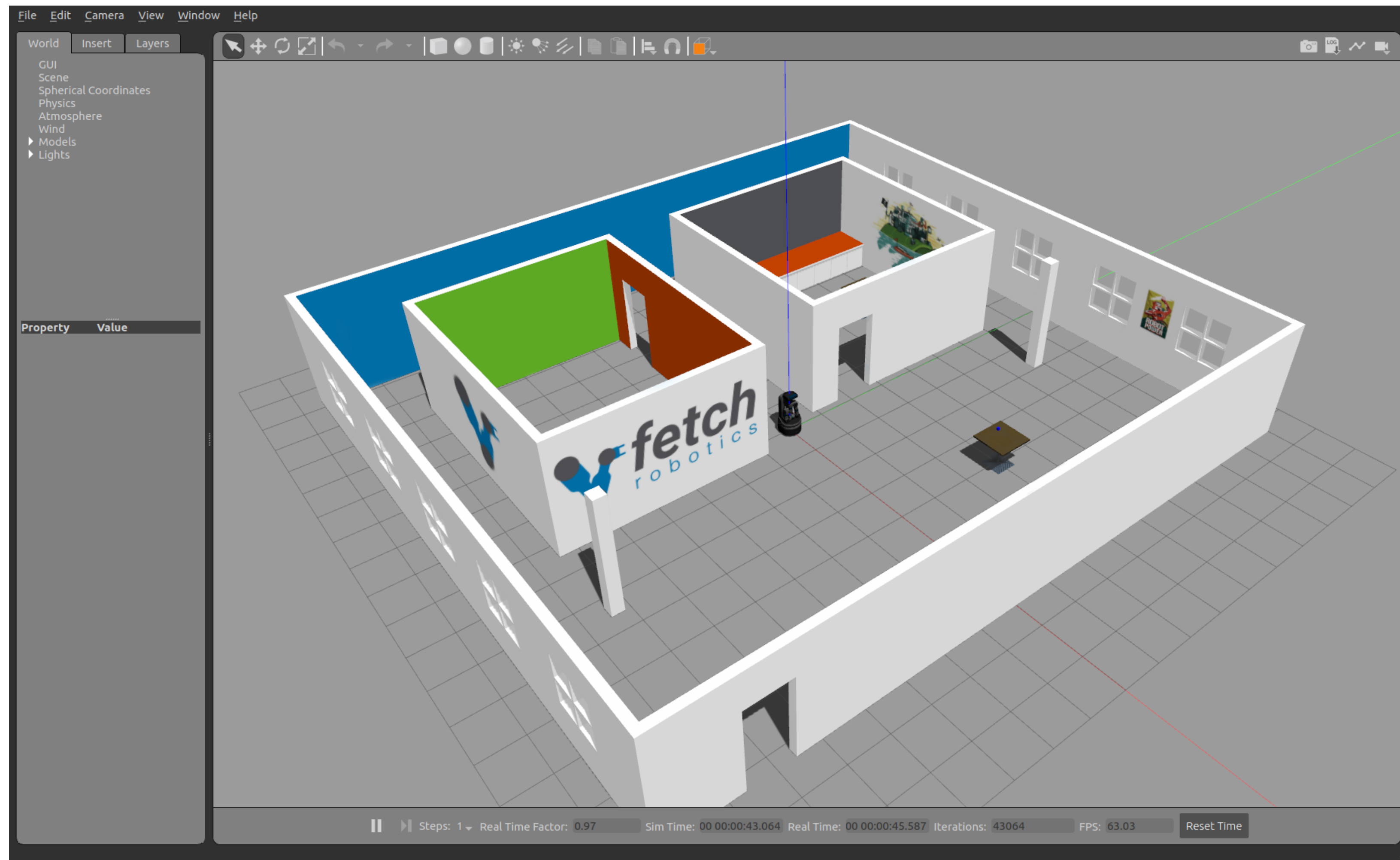
```
1  #!/usr/bin/env python
2
3  import rospy
4  from std_msgs.msg import String
5
6  def node_init():
7      # Advertise a new node named 'simple_publisher'
8      rospy.init_node('simple_publisher')
9
10     # Register 'simple_publisher' node as a publisher node
11     # Parameters:
12     #   'time_update' specifies that this node will publish to the 'time_update' topic
13     #   String           specifies that the topic's message type is std_msgs/String
14     #   queue_size=10 specifies maximum queue size before messages are dropped
15     pub = rospy.Publisher('time_update', String, queue_size=10)
16
17     # Publish once per second
18     rate = rospy.Rate(1)
19     while not rospy.is_shutdown():
20         message_string = "Uptime is %s" % rospy.get_time()
21         pub.publish(message_string);
22         rate.sleep()
23
24 if __name__ == '__main__':
25     node_init()
```



# Today: ROS Basics

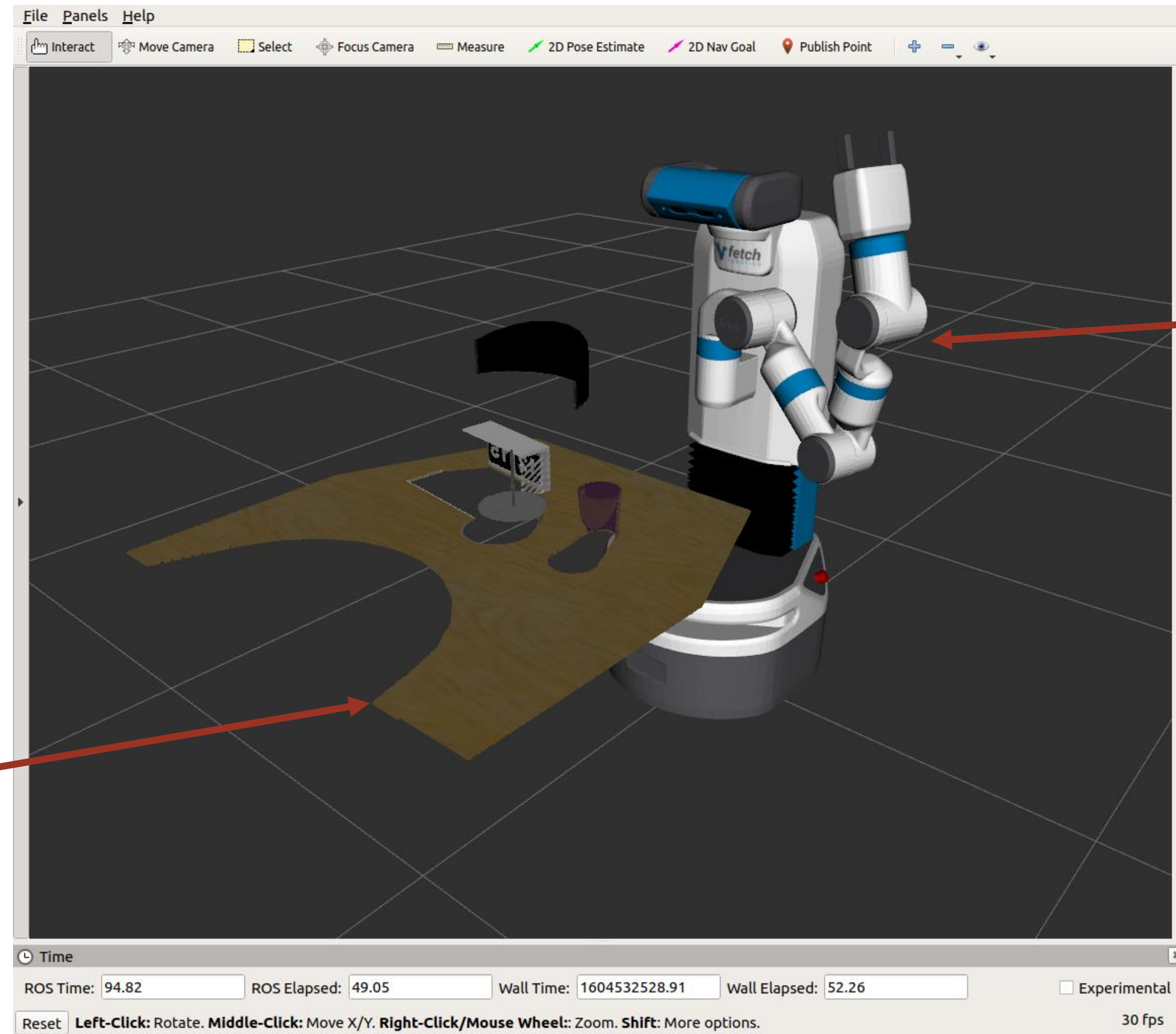
1. Working with ROS nodes, topics, and messages
2. How to write and build your own ROS node
3. Publishers and subscribers
- 4. Demo of simple publisher/subscriber nodes**
5. ROS tools to look into

# Gazebo Simulation



```
$ roslaunch fetch_gazebo playground.launch
```

# RViz



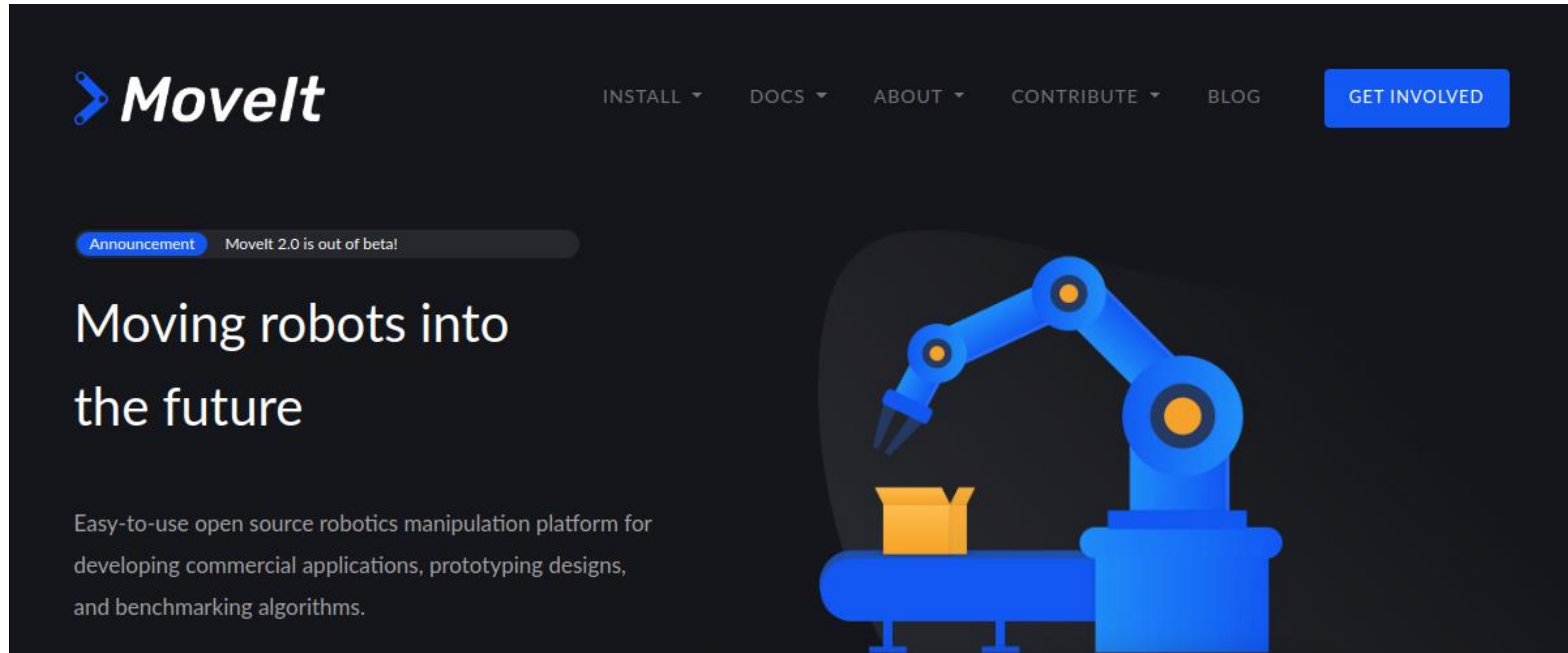
point cloud

robot model

```
$ rosrun rviz rviz
```

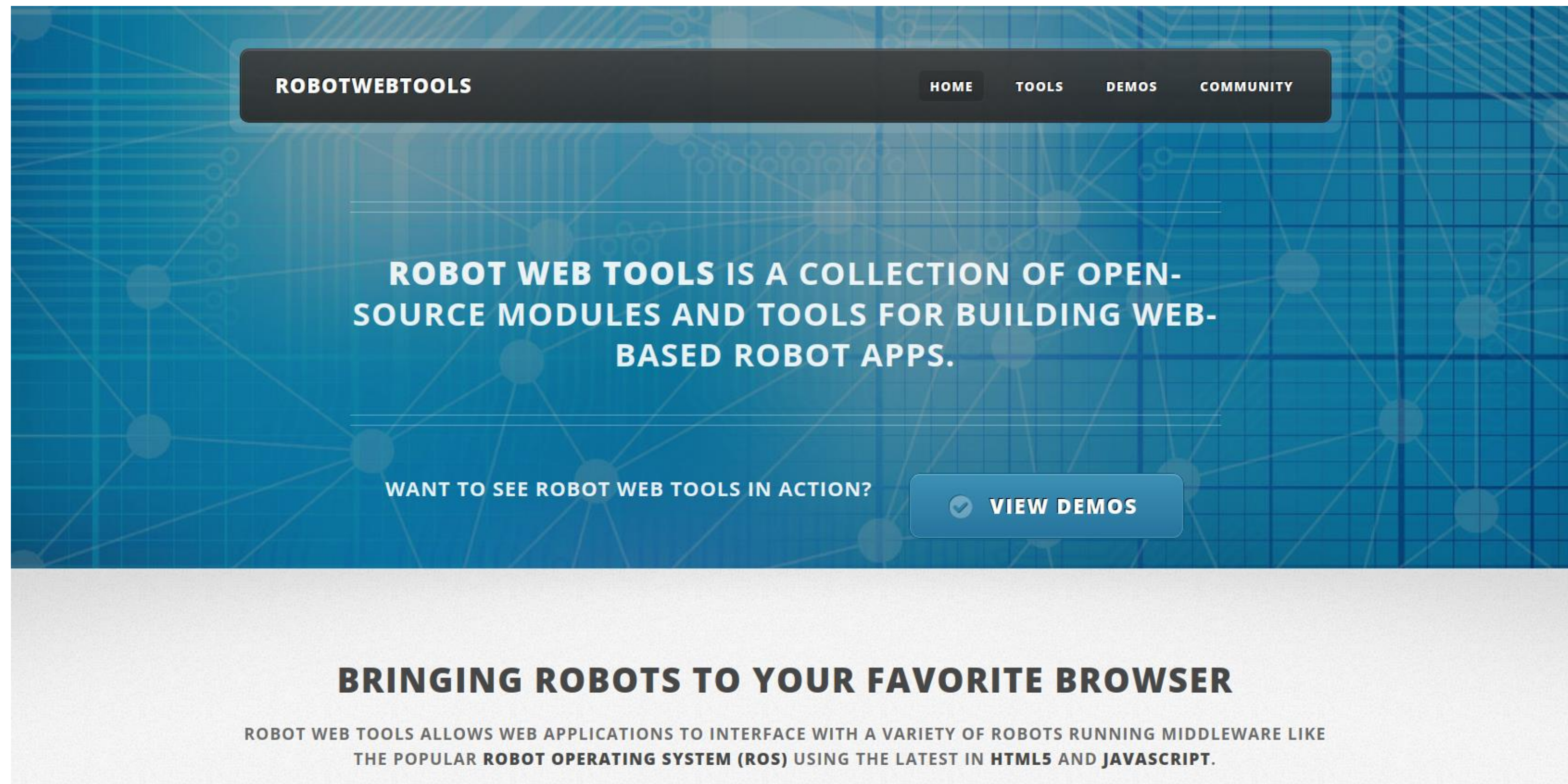


# Movelt!





# Robot Web Tools





# How to Learn More

ROS.org

About

Support

Discussion Forum

Service Status

Q&A answers.ros.org

Search:

Submit

Documentation

Browse Software

News

Download

ROS/ Tutorials

ROS Tutorials

Non-Beginners:

If you're already familiar enough with ROS [fuerte](#) or earlier versions and only want to explore the new build system introduced in [groovy](#) and used in [hydro](#) and later, called [catkin](#), you can go through more in-depth [catkin tutorial here](#). However, going over all basic [Beginner Level](#) tutorials is still recommended for all users to get exposed to new features.

If you are new to Linux:

You may find it helpful to first do a quick tutorial on common command line tools for linux. A good one is [here](#).

Contents

1. ROS Tutorials

1. Core ROS Tutorials

1. Beginner Level

2. Intermediate Level

2. ROS Standards

3. Tutorials for Other ROS Libraries

4. Tutorials for Libraries with ROS Interfaces

5. External ROS Resources

1. External Tutorials

2. External Seminar/Lecture

6. Using ROS on your custom Robot

Wiki

Distributions

ROS/Installation

ROS/Tutorials

RecentChanges

Page

Immutable Page

Info

Attachments

More Actions:

Raw Text

Do

User

Login

## 1. Core ROS Tutorials

wiki.ros.org/ROS/Tutorials