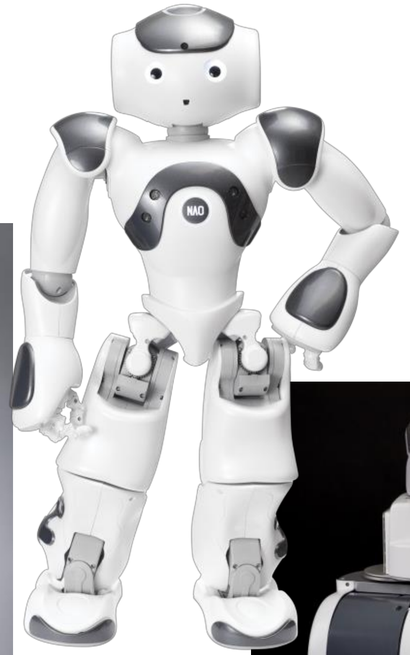
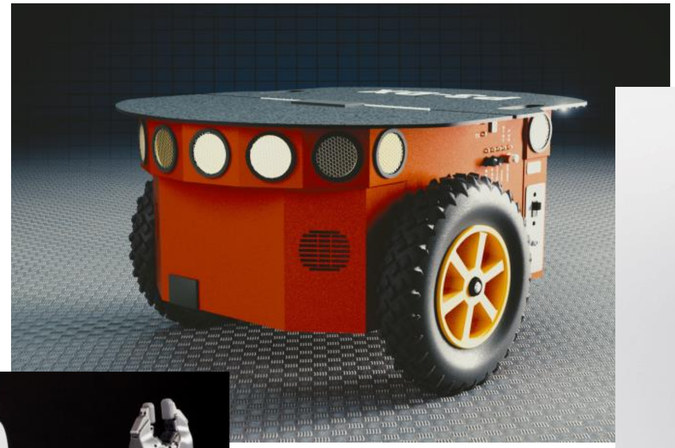
EECS 367 & ROB 320 Lab
ROS Tutorial

Michigan EECS 367 *Introduction to Autonomous Robotics* | ROB 320 *Robot Operating Systems*

Why is robotics so hard?





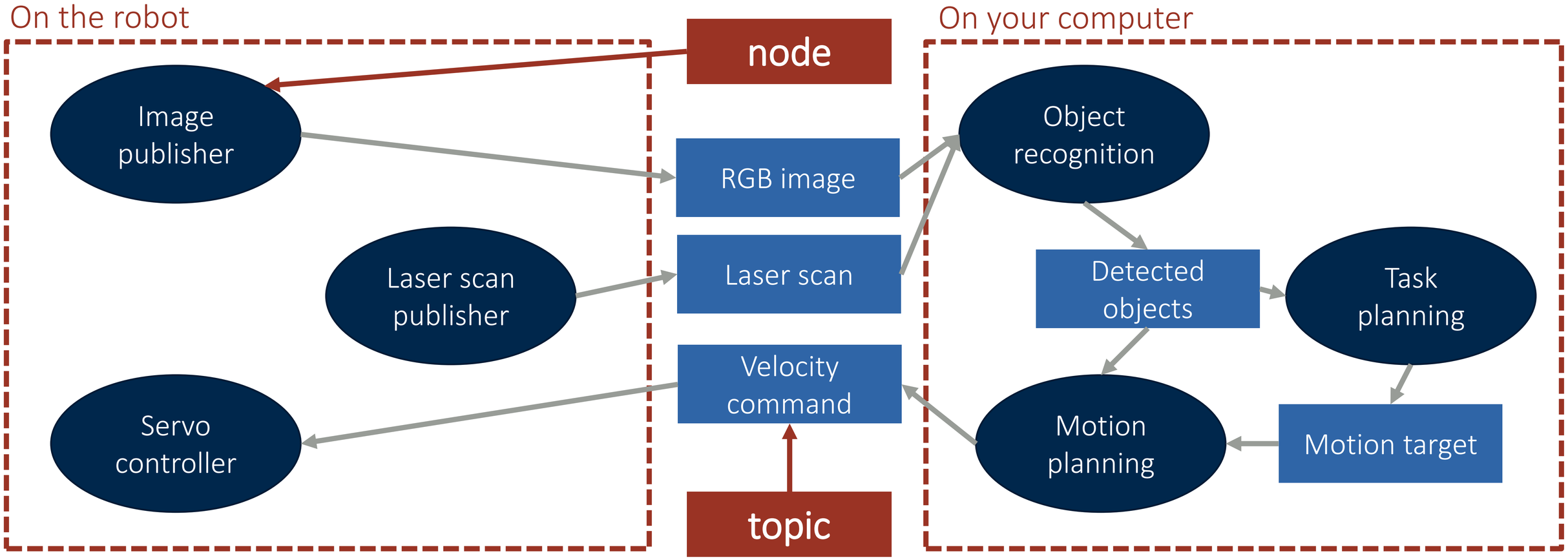
What is ROS?

“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

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

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 singe 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: `$ rosmsg show msg_pkg/msg_name`
- See publisher/subscriber graph: `$ rosrun 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

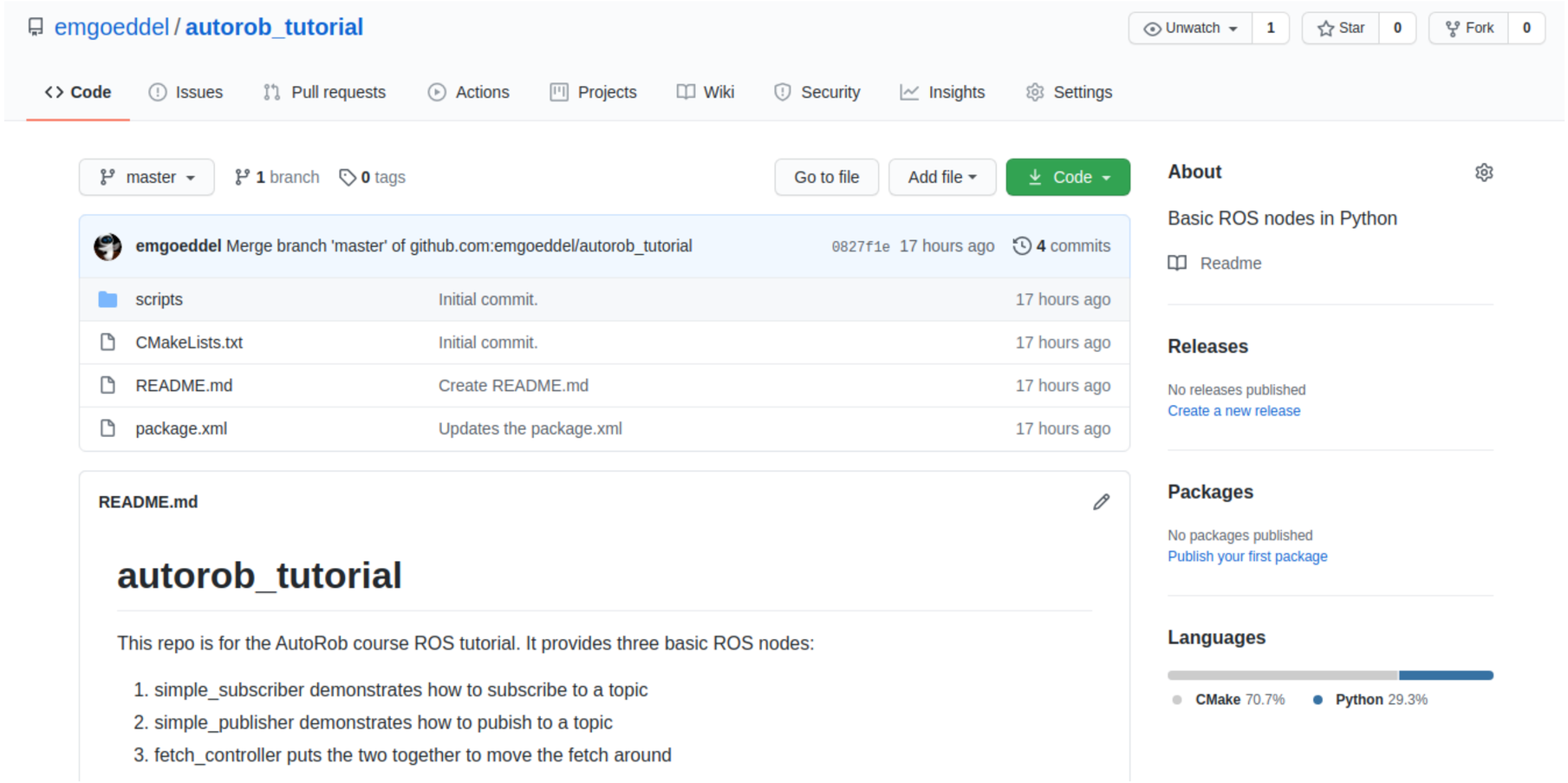
```
$ cd ~/autorob_tutorial
```

```
$ mkdir scripts
```

```
$ cd scripts
```

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

Demo Code



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

```python
#!/usr/bin/env python

import rospy
from sensor_msgs.msg import LaserScan

def subscriber_callback(data):
    rospy.loginfo(rospy.get_caller_id() + " node received laser scan message!")
    return

def node_init():
    # Advertise a new node named 'simple_subscriber'
    rospy.init_node('simple_subscriber')

    # Register 'simple_subscriber' node as a subscriber node
    # Parameters:
    #    'base_scan'       specifies that this node will subscribe to the 'base_scan' topic
    #    LaserScan         specifies that the topic's message type is sensor_msgs/LaserScan
    #    listener_callback specifies the callback function used when a 'base_scan' message is received
    rospy.Subscriber('base_scan', LaserScan, subscriber_callback)

    # Keep node running until node is exited
    rospy.spin()

if __name__=='__main__':
    node_init()
```

Simple Publisher

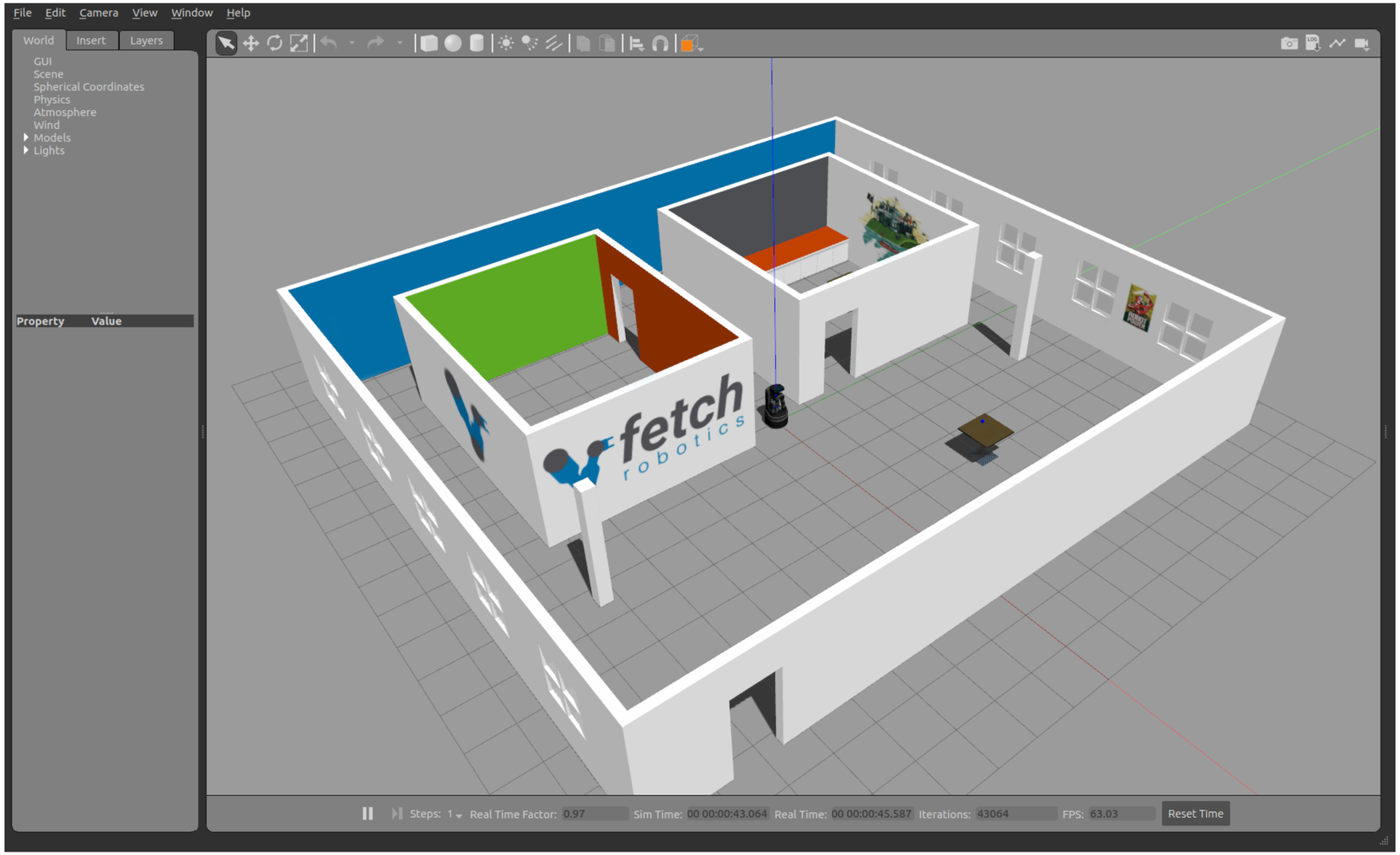
simple_publisher.py

```python
#!/usr/bin/env python

import rospy
from std_msgs.msg import String

def node_init():
    # Advertise a new node named 'simple_publisher'
    rospy.init_node('simple_publisher')

    # Register 'simple_publisher' node as a publisher node
    # Parameters:
    #    'time_update' specifies that this node will publish to the 'time_update' topic
    #    String        specifies that the topic's message type is std_msgs/String
    #    queue_size=10 specifies maximum queue size before messages are dropped
    pub = rospy.Publisher('time_update', String, queue_size=10)

    # Publish once per second
    rate = rospy.Rate(1)
    while not rospy.is_shutdown():
        message_string = "Uptime is %s" % rospy.get_time()
        pub.publish(message_string);
        rate.sleep()

if __name__=='__main__':
    node_init()
```

Today: ROS Basics

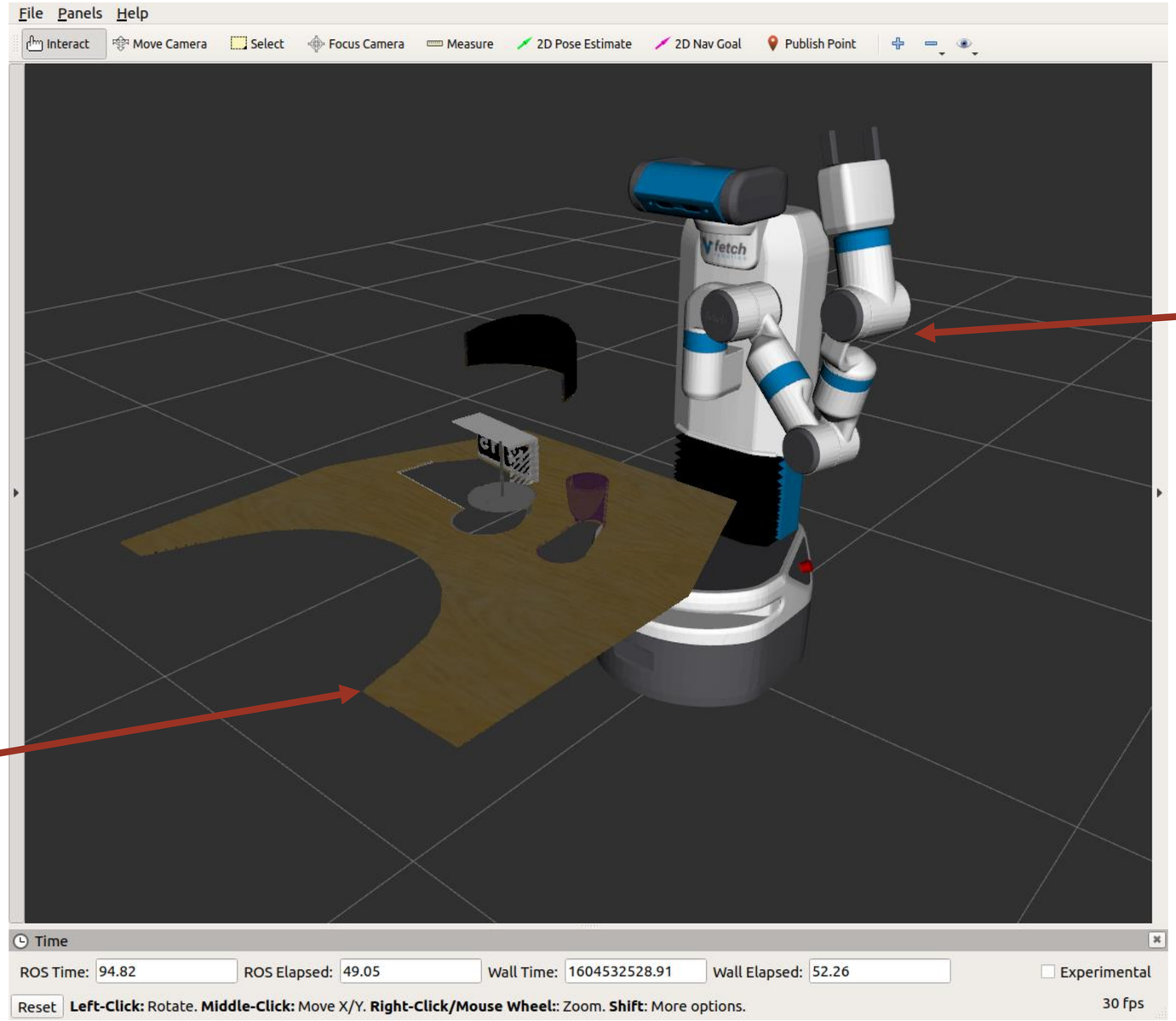
1. Working with ROS nodes, topics, and messages
2. How to write and build your own ROS node
3. Publishers and subscribers
4. **Demo of simple publisher/subscriber nodes**
5. ROS tools to look into

Gazebo Simulation




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

RViz

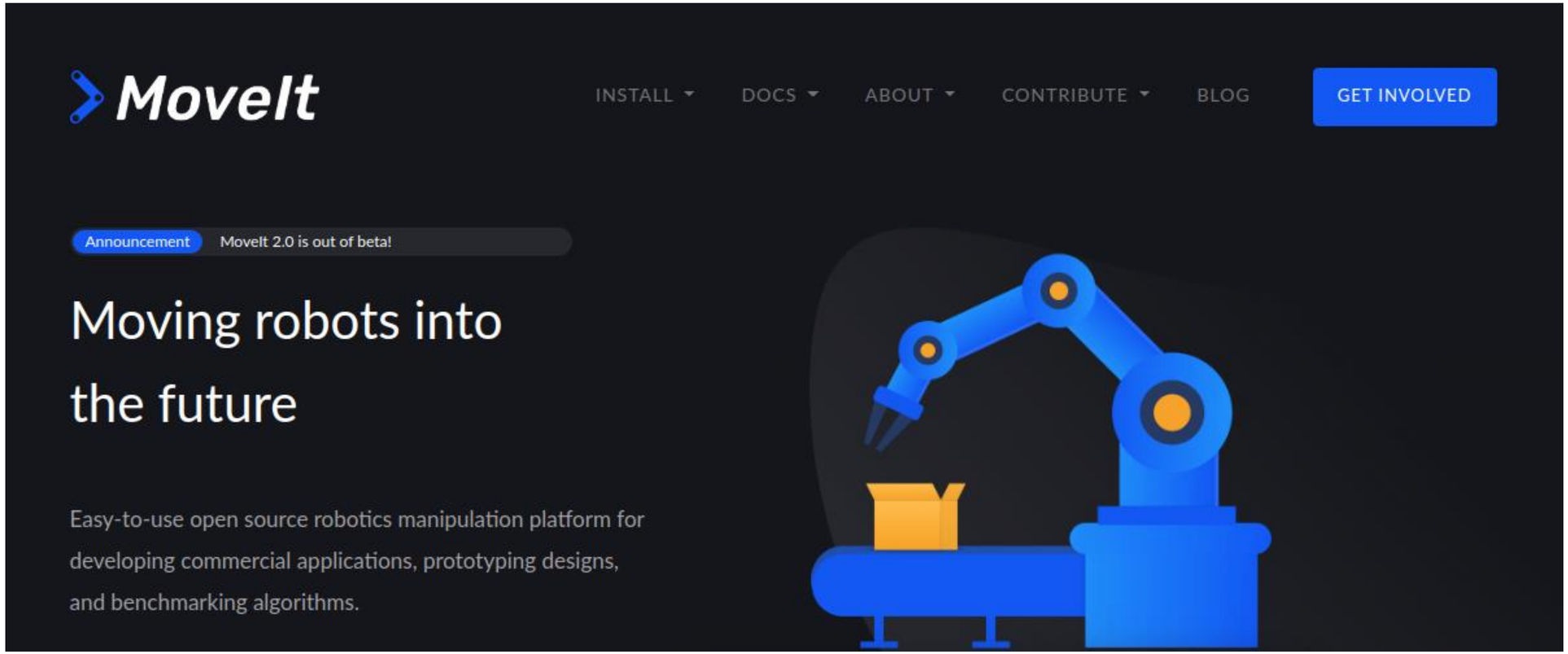


robot model

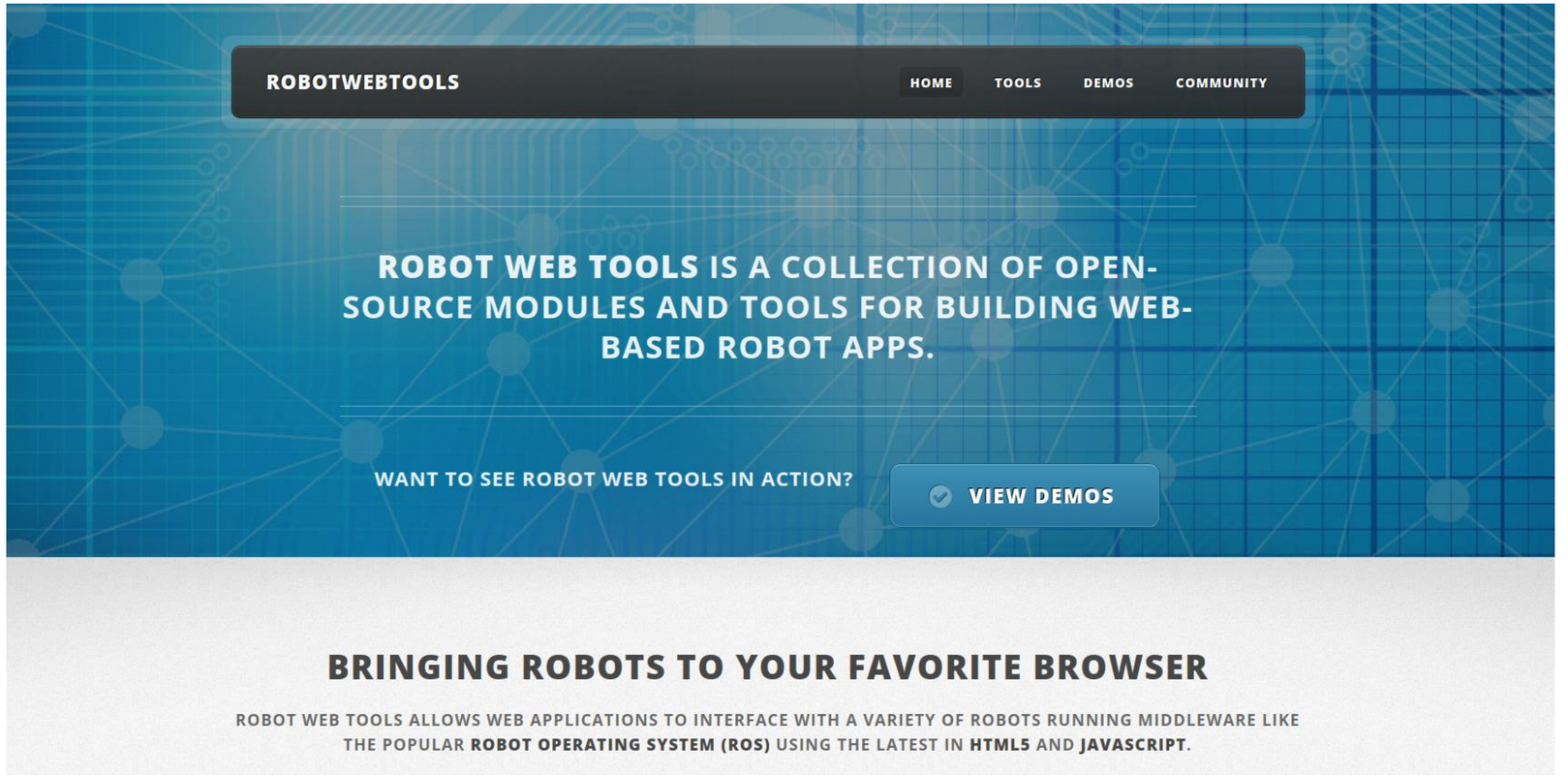
point cloud

```
$ rosrun rviz rviz
```

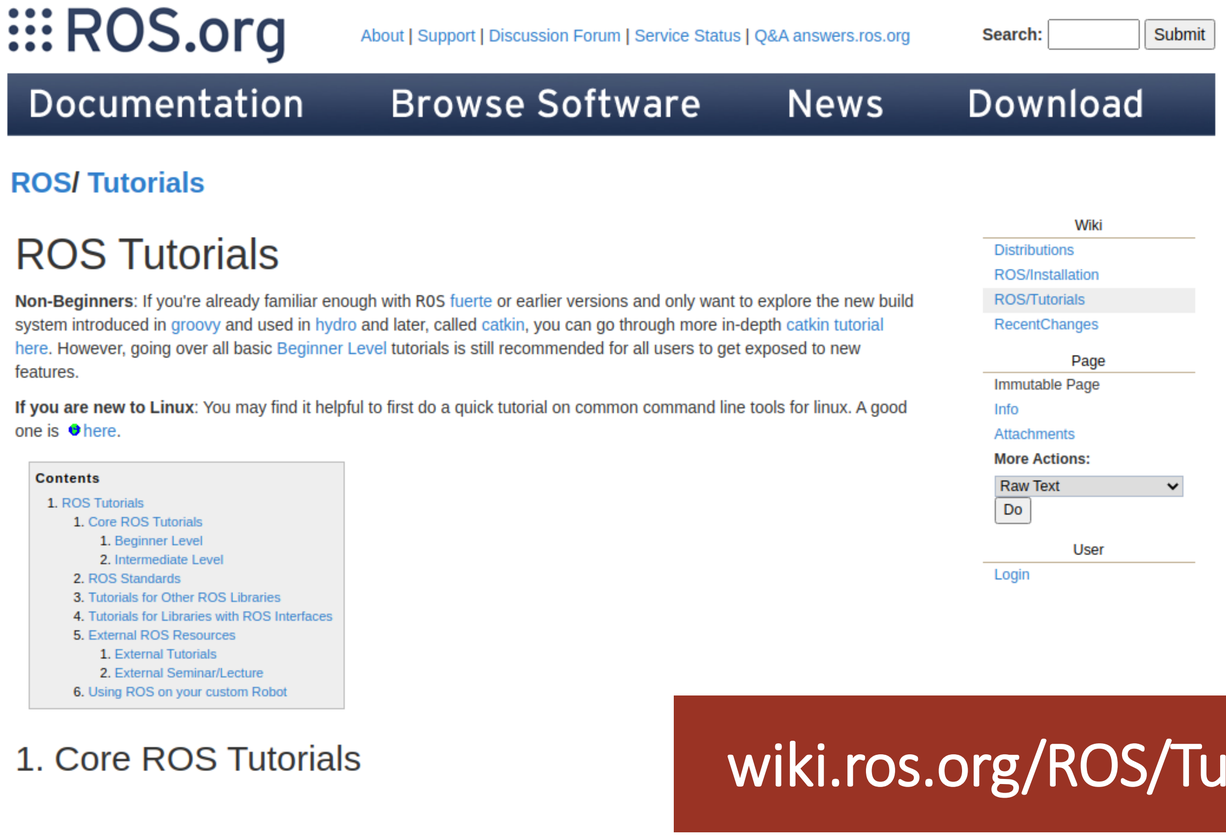
MoveIt!


MoveIt
INSTALL
DOCS
ABOUT
CONTRIBUTE
BLOG
GET INVOLVED
Announcement
MoveIt 2.0 is out of beta!
Moving robots into the future
Easy-to-use open source robotics manipulation platform for developing commercial applications, prototyping designs, and benchmarking algorithms.

Robot Web Tools



How to Learn More



wiki.ros.org/ROS/Tutorials