

The Construct

Learn and develop for robots with ROS

PRESENTS

ROS Developers Live Class n74

theconstruct.ai

How to create your own
ROSject

ROS
Developers
LIVE CLASS
#74

The Construct

How to create a Rosject

In this class, you will learn how to create a ROSject and all the parts that compose it. The ROSject allows you to create full ROS projects without mixing all its elements: ROS code, Gazebo simulation, documentation, datasets.

You will learn how to organise all that material into a single ROS project (a ROSject), and how to share it with your colleagues, students or teachers, so they can reproduce your results.

What is a ROSject?

Here at the ROS Development Studio, ROSjects are the way we create our ROS projects in an orderly manner. Establishing a separate order between **simulations, ROS code, datasets, and documentation** is very important when reviewing, sharing or even debugging a project.

The ability to work in a simple and orderly way and all the benefits that ROS offers us in its different branches allows us to better understand and optimize projects in various fields of robotics, such as mobile, industrial robotics, etc. And in turn, work in different fields, such as artificial intelligence, navigation, etc.

If you are interested in becoming a **Robotics Developer** you will need to know how to represent the robot structure in the proper way so you can program it with ROS.

(To know more about becoming a robotics developer, read this guide about [How To Become a Robotics Developer \(http://www.theconstructsim.com/become-robotics-developer/\)](http://www.theconstructsim.com/become-robotics-developer/))

This rosject has been created by **Christian Chavez** and **Ricardo Tellez** from **The Construct**. You can use this rosject freely as long as you keep this notice.

REQUIREMENTS :

- **Basics of Linux.** If you don't have that knowledge, [check this FREE online course \(https://www.robotigniteacademy.com/en/course/linux-robotics/details/\)](https://www.robotigniteacademy.com/en/course/linux-robotics/details/)



- **Ros Basics.** If you don't have that knowledge, [check this online course \(https://www.robotigniteacademy.com/en/course/ros-in-5-days/details/\)](https://www.robotigniteacademy.com/en/course/ros-in-5-days/details/)



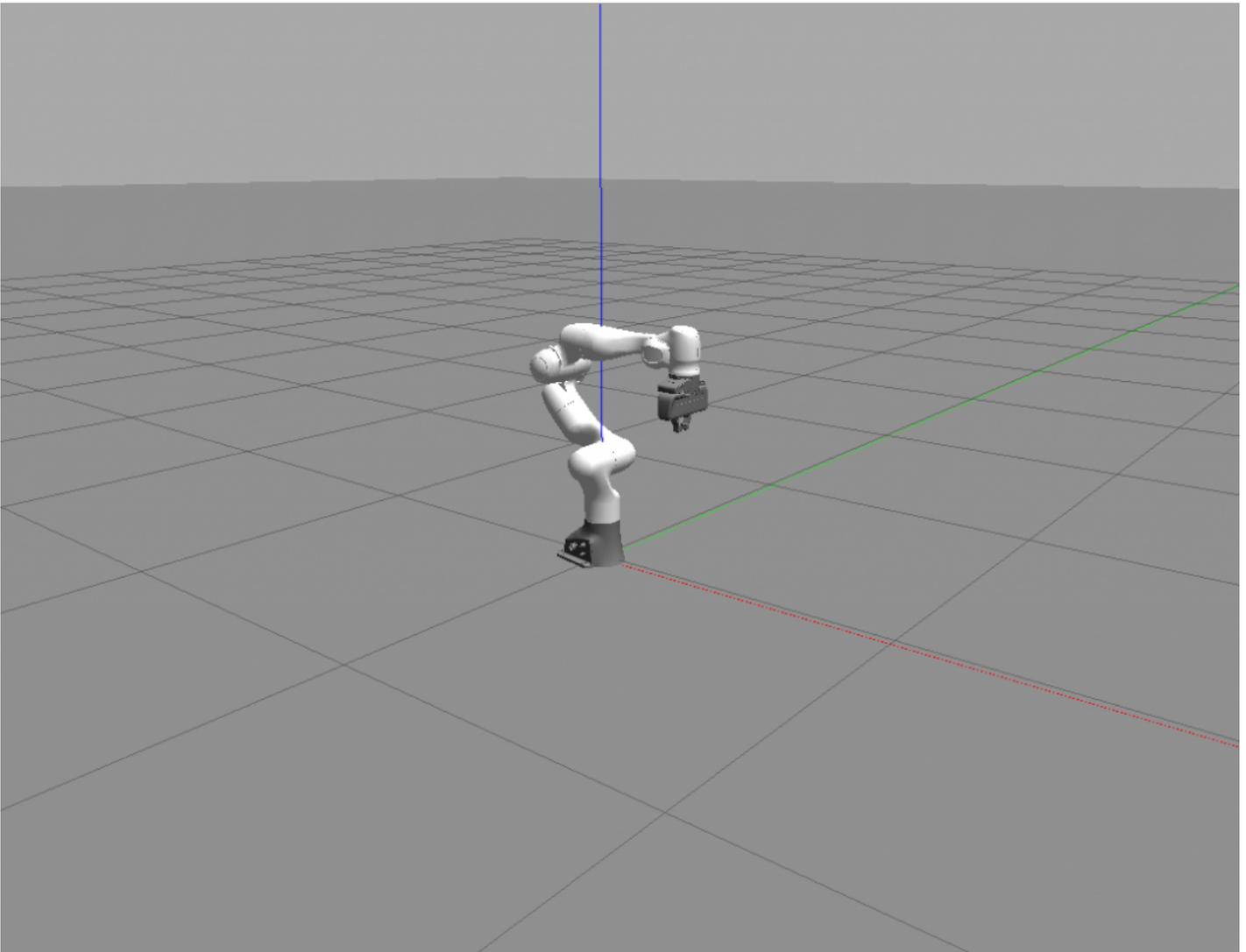
- **Love for Robotics and patience**
- ... That's it! Let's go!

In this class, we'll learn:

- How to create a rosject.

Robot for today's Live Class

Today you're going to use the Panda, from FRANKA EMIKA.



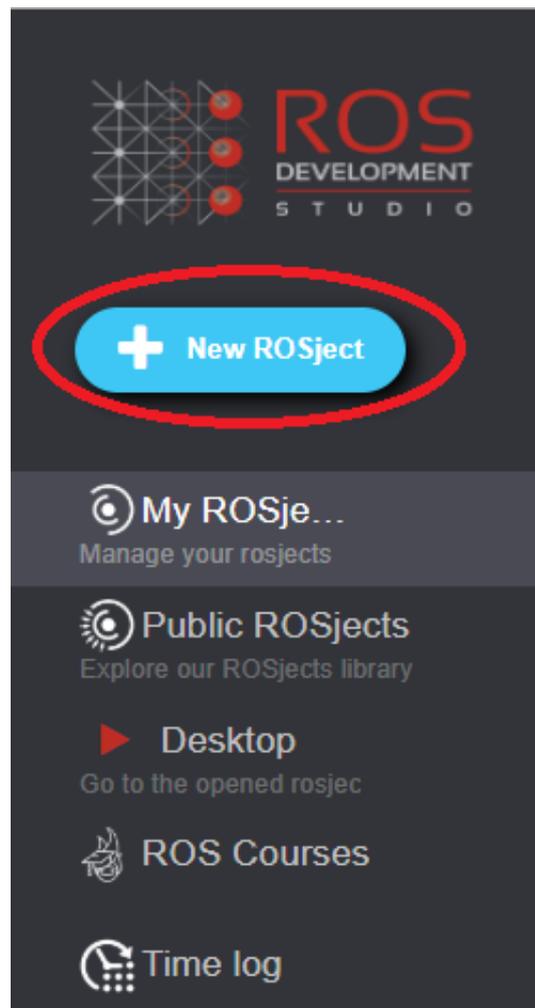
How to create a ROSject

Let's see all the steps required to create your own rosject

1. Create an empty ROSject

First we are going to create an empty ROSject that will contain all our code, simulations, documentations and datasets

When you go to our **ROS Development Studio** you will find a menu in the left side of the window similar to the next image. You have to click in the button that said **+ New ROSject**



Once you have clicked, you have to fill the necessary data in order to create a new rosject.

Create new ROSJect

NAME	<input type="text" value="ROS developers live class n74 : How to create a rosject"/>
THUMBNAIL IMAGE (*.PNG, *.JPEG)	<input type="button" value="Choose File"/> No file chosen
ROS CONFIGURATION	<input type="text" value="Ubuntu 16.04 + ROS Kinetic + Gazebo 7"/>
SELECT A ROBOT TO PROGRAM FOR	<input type="text" value="- No robot selected -"/>
PRIVATE OR PUBLIC?	<input type="text" value="Private"/>
DESCRIPTION	<input type="text"/> Description of the simulation

Here you will have the name of your new ROSject, an image that will have your rosject, the ROS Configuration, some configurations already established for certain robots that you could use them, if you want to create a public or private rosject, and a description of your rosject. Once you have already fill all the data you will have something similar to this.

In order to make the rosject easier to identify, assign to it an image. For that, I usually go to Google Images and download one. Let's do it!



Image obtained from IF World Design Guide

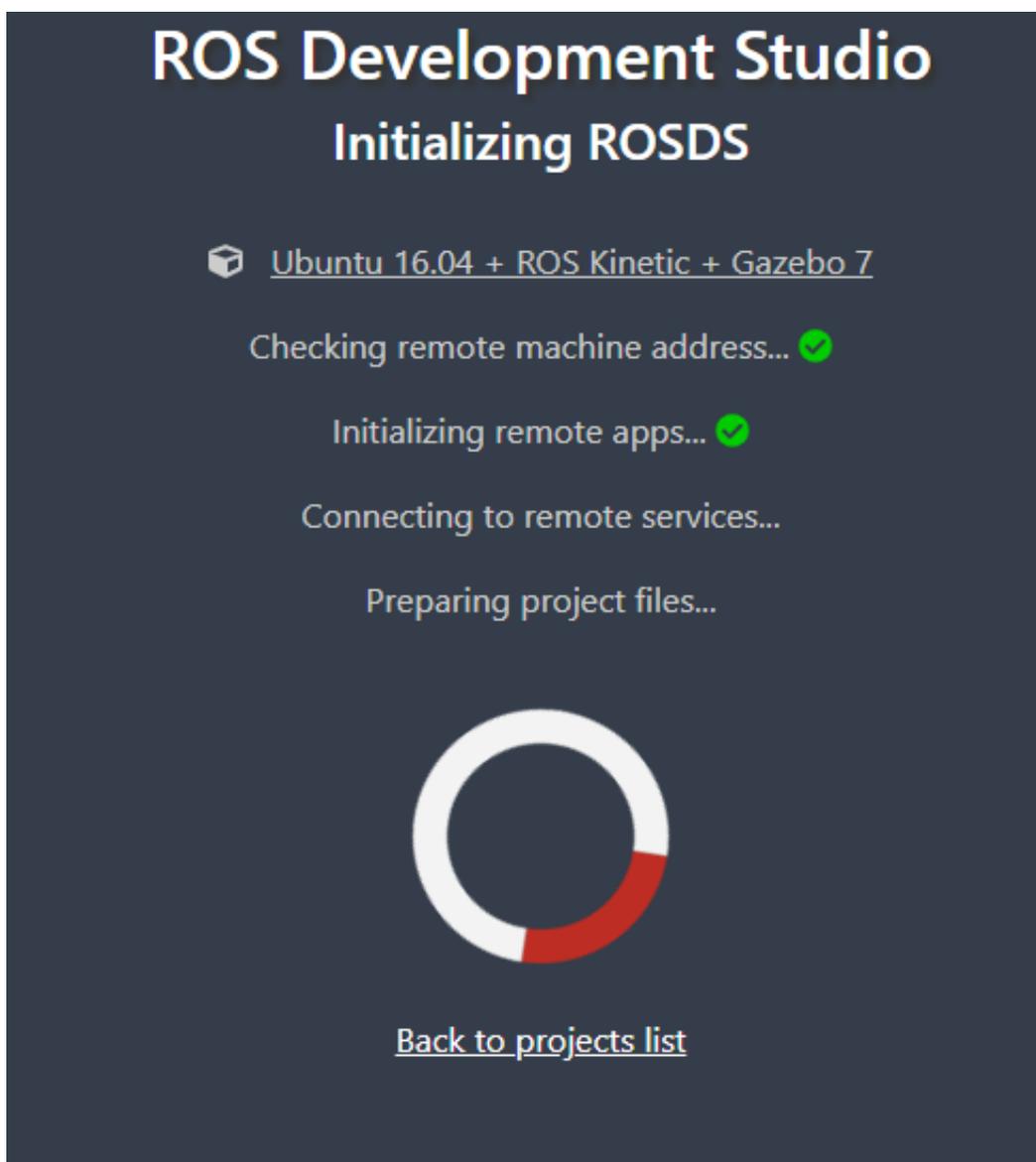
The screenshot shows the ROS Studio interface. On the left, there is a 'Select hardware' section with three options: 'Basic' (181:35:50), 'Pro I' (06:52:49), and 'Pro II' (00:00:00). Below these is a red 'Open ROSJect' button. The system configuration is listed as 'Ubuntu 16.04 + ROS Kinetic + Gazebo 7' with 'Datasets (0)' and 'Updated on 2019-11-18'. On the right, a 'README' section contains the text: 'This is your ROSJect documentation!', 'A place to present your ROS Project to the world!', and 'How to customize it:' followed by a list of instructions. Below the list is a code editor showing a Python print statement: 'In [1]: print 'Hello from my ROSJect!'' and the output 'Hello from my ROSJect!'. At the bottom, there are 'Share & Contribute' buttons (Share, Fork), 'ROSJect activity' (Issues), and 'Manage your ROSJect' (Edit, Remove) buttons.

To the right of the window you will have some view of the notebook or documentation of your ROSject, as it's new you will have something similar to the previous image . Then to the left you will have more options. You can choose the hardware that you will use, some rosjects requires powerful cpu, you can use them for a very fair price, but most of the rosjects needs only the basic cpu.

Then here you can open the ROSject or maybe Share it, edit, remove, etc.

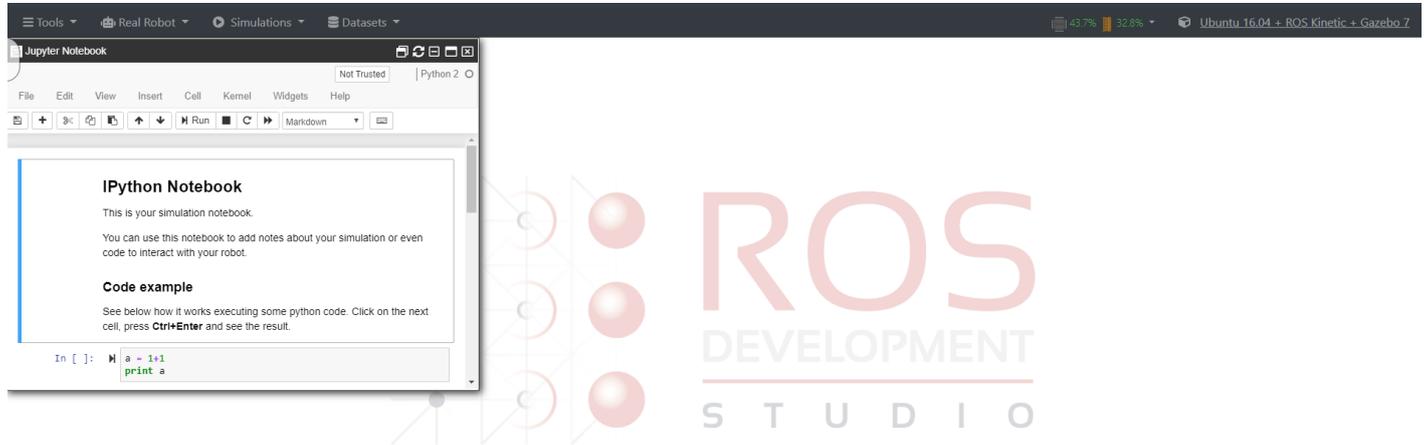
1b. Opening the empty ROSject

Now open the ROSject.



Now the ROSject is opening, wait a moment, this will load soon.

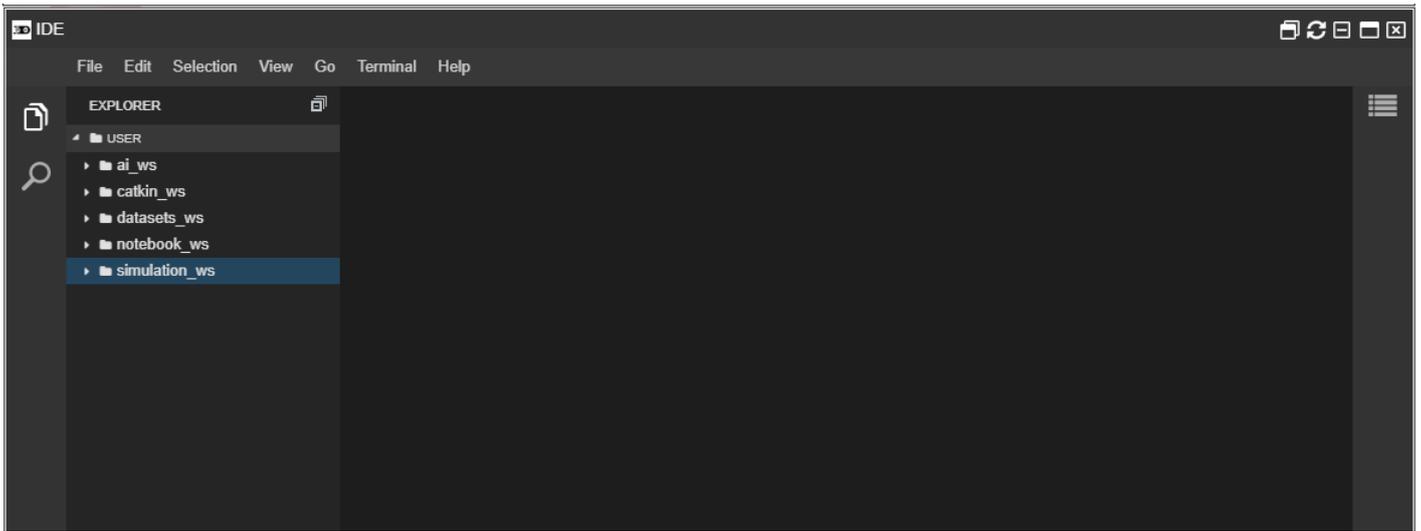
Now you should have something like the next image.



Where you have already opened the jupyter notebook, where you will write all the documentation of your ROSject.

We are going to see later how to create the documentation.

But let's see the workspaces that we have. Go to the **Tools** menu at the top of the windows and select IDE.



As you can see you have five workspace:

- ai_ws
- catkin_ws
- datasets_ws
- notebook_ws
- simulation_ws

ai_ws

The *ai_ws* is used for all related to artificial intelligence, we know that artificial intelligent could manage large amounts of information for proper processing, that's why we recommend to manage all of this in a dedicated workspace.

catkin_ws

The *catkin_ws* is the most common use in ROS, this workspace is used for all the scripts that is not use in other workspaces, move_groups, artificial vision, mapping, exploring, etc. **This time, in order not to spend much time, we won't use this workspace.**

datasets_ws

The *datasets_ws* is used only as its name says, store and manage datasets, here is where we gonna make datasets of the info we need, as rosbag for example, here we can store images, or just data of a sensor or a topic.

notebook_ws

The *notebook_ws* as we mentioned before is to keep all the information of the documentation, images and notebooks that the jupyter notebook will use.

simulation_ws

The *simulation_ws* is the second most common workspace used. This workspace is used to keep all the configuration needed in order to create a simulation of a robot, as the robot description, control, launches files of gazebo, rviz, etc.

2. Adding the simulation code

Well in order to prepare the simulation, go to the **Shell** inside the **Tools** menu and open it.

A terminal window titled "Shell" with a dark background. The prompt "user:~\$" is visible at the top left.

Here run the following command in order to go inside the **simulation_ws/src**

```
In [ ]: $ cd /home/user/simulation_ws/src
```

Here we will follow the instructions of the git made by **Erdal Pekel**

https://github.com/erdalpekel/panda_simulation (https://github.com/erdalpekel/panda_simulation), This will bring us the simulation of the PANDA from FRANKA EMIKA

Run the followings commands in a Shell to get the code of the simulation:

```
In [ ]: $ git clone https://github.com/erdalpekel/panda_simulation.git
```

3. Adding the ROS code

Then, on the `catkin_ws` you have to include the code that would run in the robot:

```
In [ ]: $ cd /home/user/catkin_ws/src
```

```
In [ ]: $ git clone https://github.com/erdalpekel/panda_moveit_config.git
```

```
In [ ]: $ git clone --branch simulation https://github.com/erdalpekel/frank_a_ros.git
```

Compiling the code requires some packages to be installed in the system. They have been already installed in the ROSDS for other projects, so we don't need to install them here.

Now, let's compile the robot code:

```
In [ ]: $ cd ..
```

```
In [ ]: $ catkin_make -j4 -DCMAKE_BUILD_TYPE=Release
```

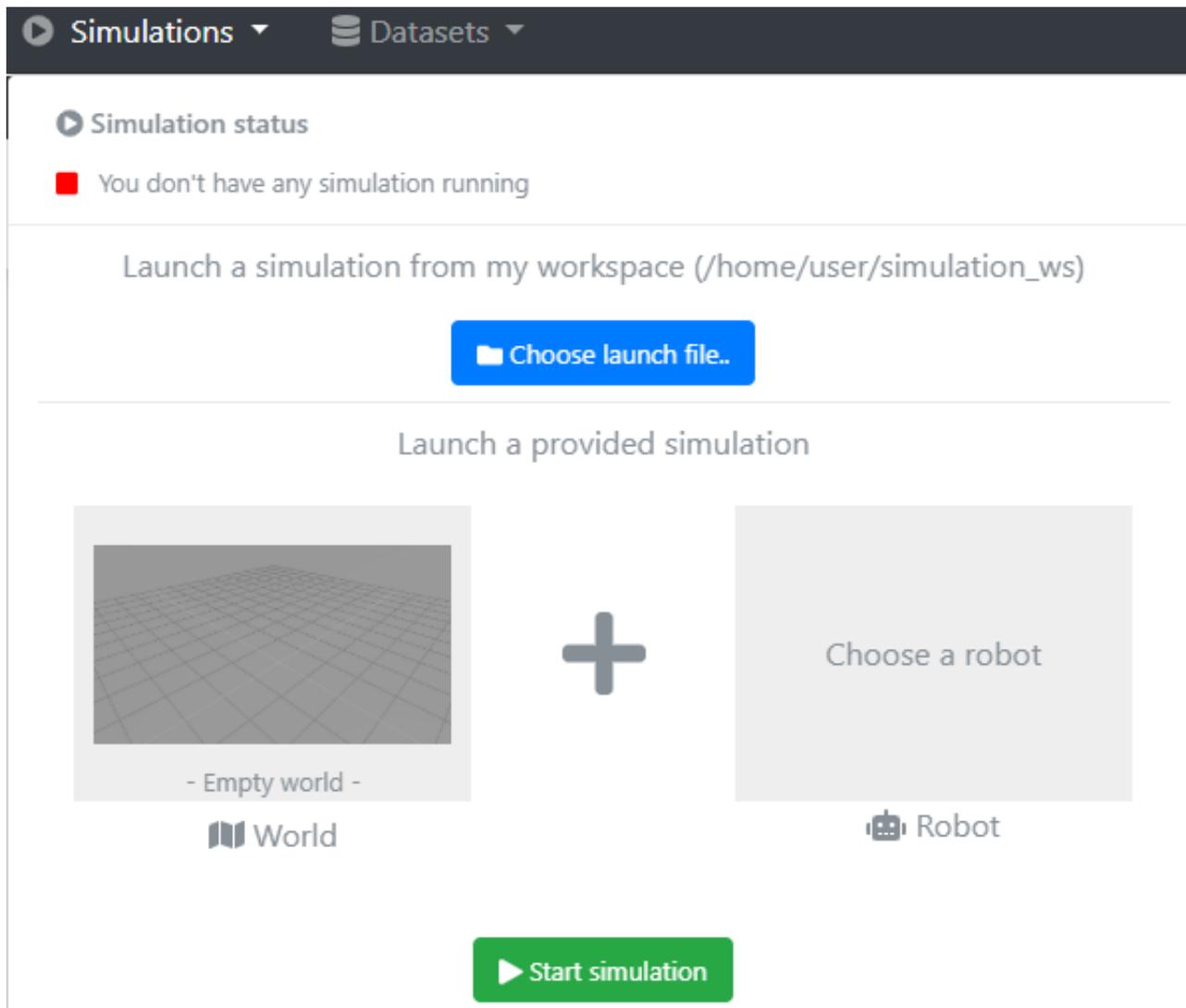
```
In [ ]: $ source devel/setup.bash
```

Then let's compile the simulation. We need to follow that specific order because the simulation depends on the code of the robot .

```
In [ ]: $ cd /home/user/simulation_ws  
$ catkin_make  
$ source devel/setup.bash
```

3b. launch the simulation

Great! Now it's time to see our simulation. Go to the **Simulations** menu at the top of the window.



Simulations ▾ Datasets ▾

▶ Simulation status

■ You don't have any simulation running

Launch a simulation from my workspace (/home/user/simulation_ws)

Choose launch file..

Launch a provided simulation

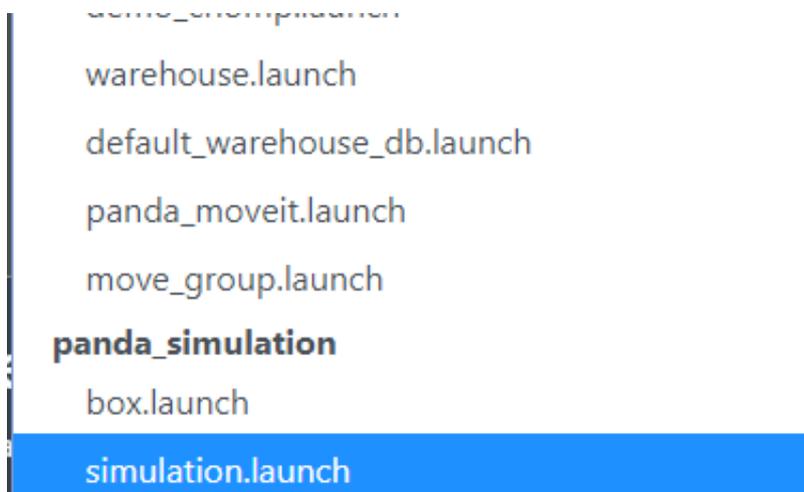
- Empty world -
World

+

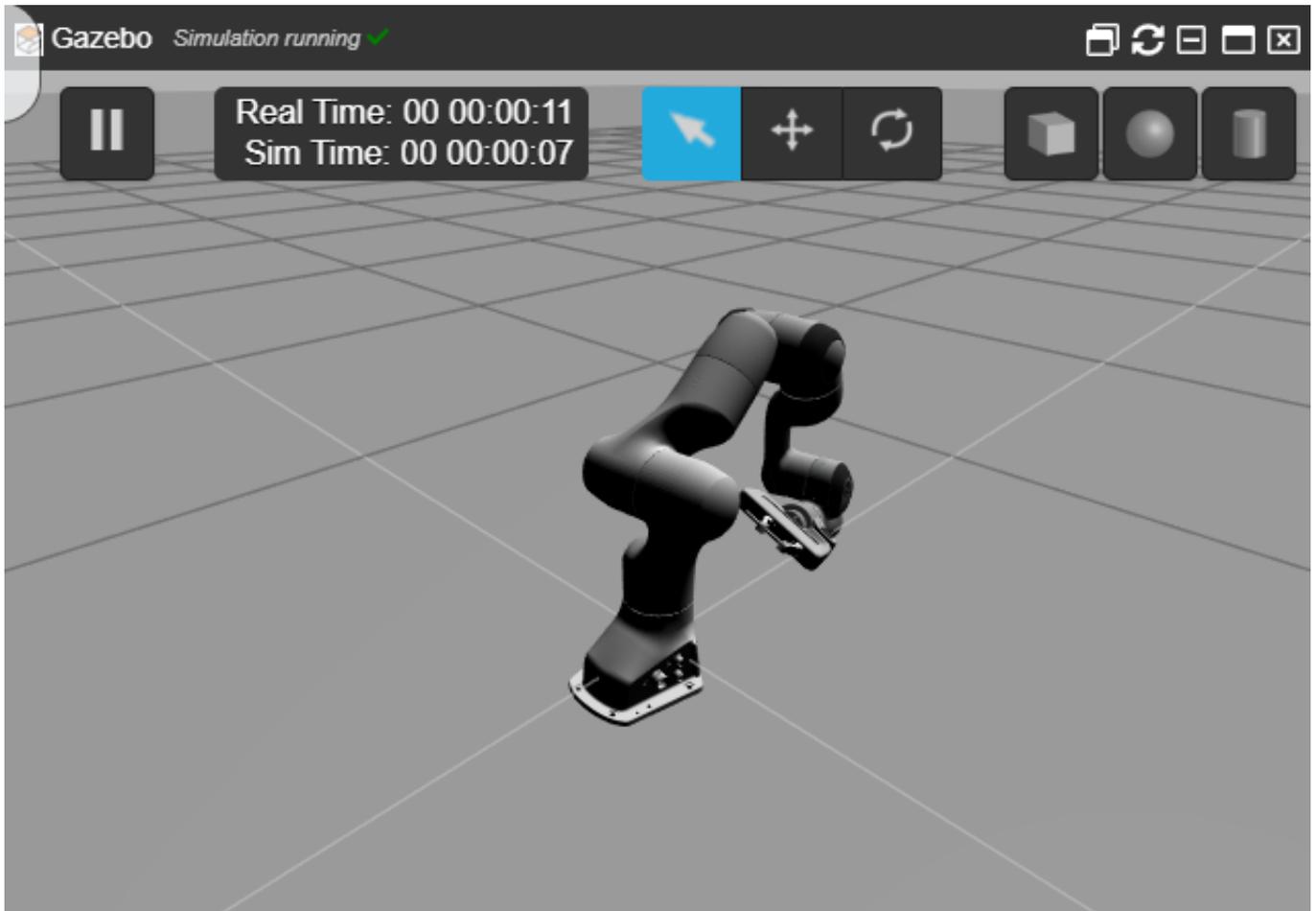
Choose a robot
Robot

▶ Start simulation

Then select Choose launch file.. and select **panda_simulation simulation.launch**



You will see the simulation running like the following image.

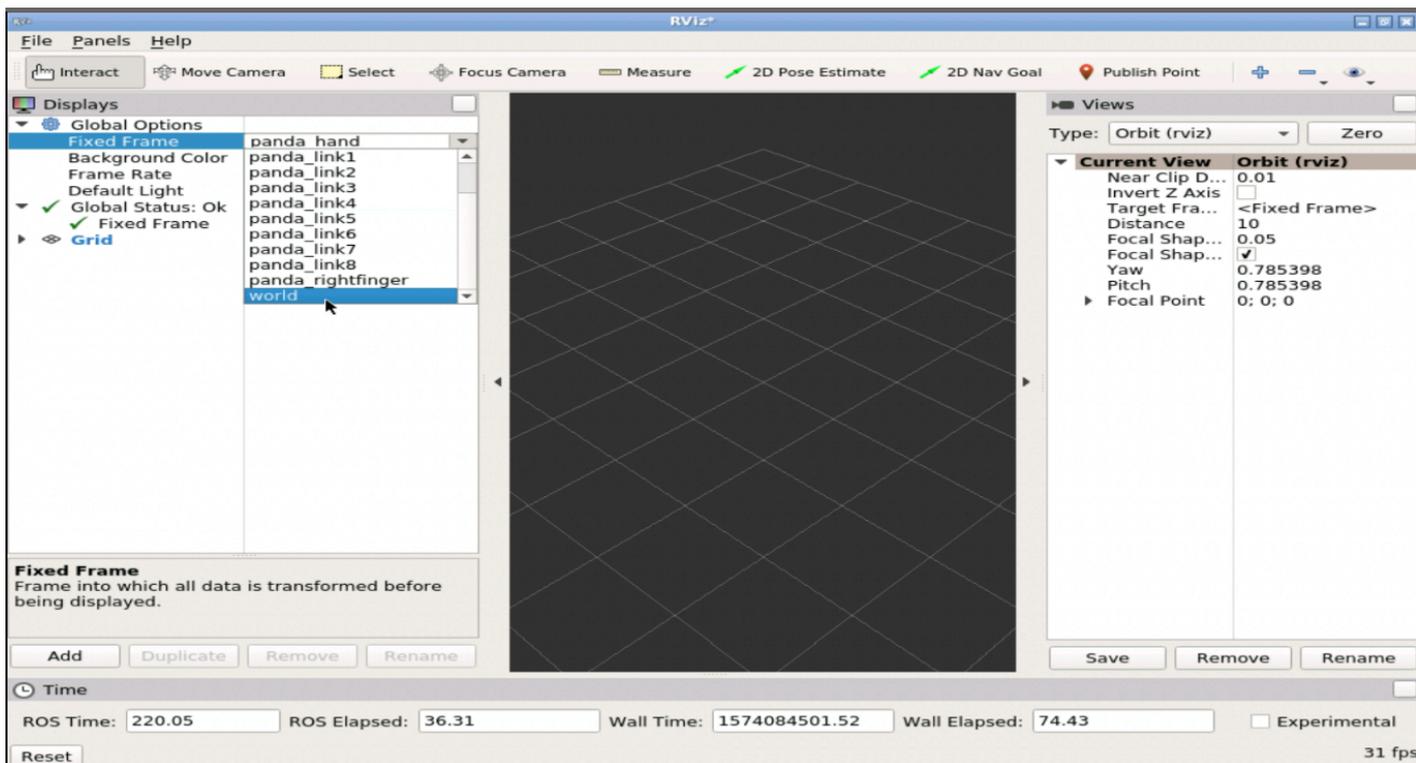


Now see how the simulation is integrated with **RVIZ** and **MOVEIT** in order to move the robot!

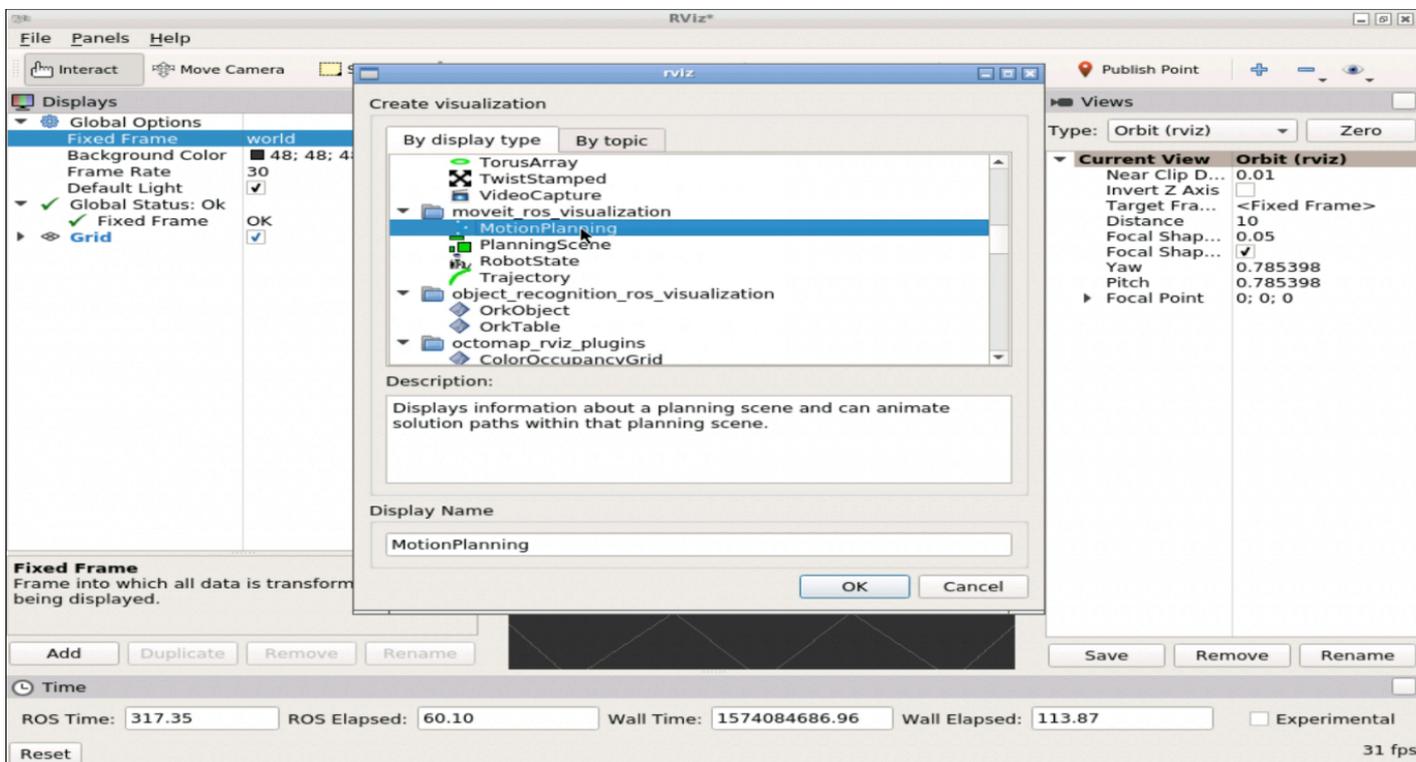
In the Shell run the next command.

```
In [ ]: $ rviz
```

After that, go to **Tools** menu and open the **Graphical Tools**



In the rviz change the **fixed frame** map to **world**, as the previous image, then Add a **MotionPlanning**



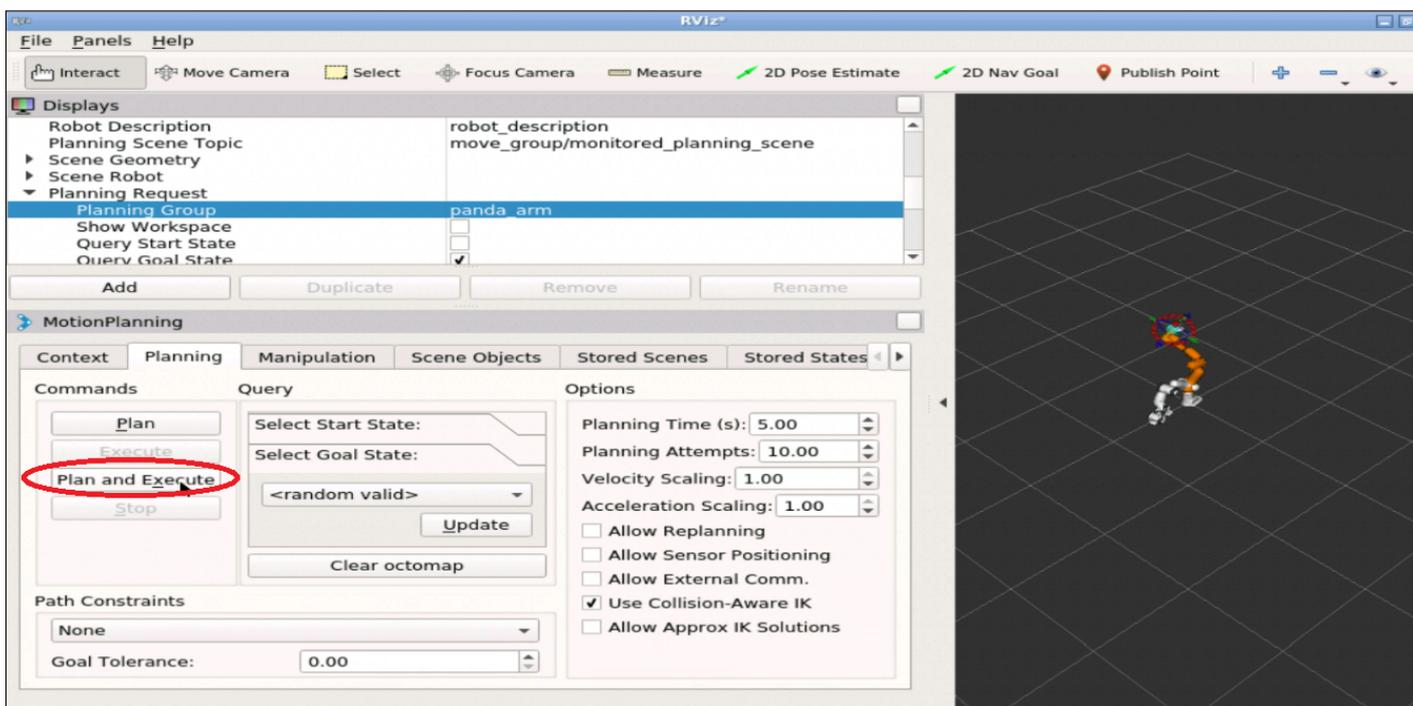
In order to configure the **MotionPlanning** open it



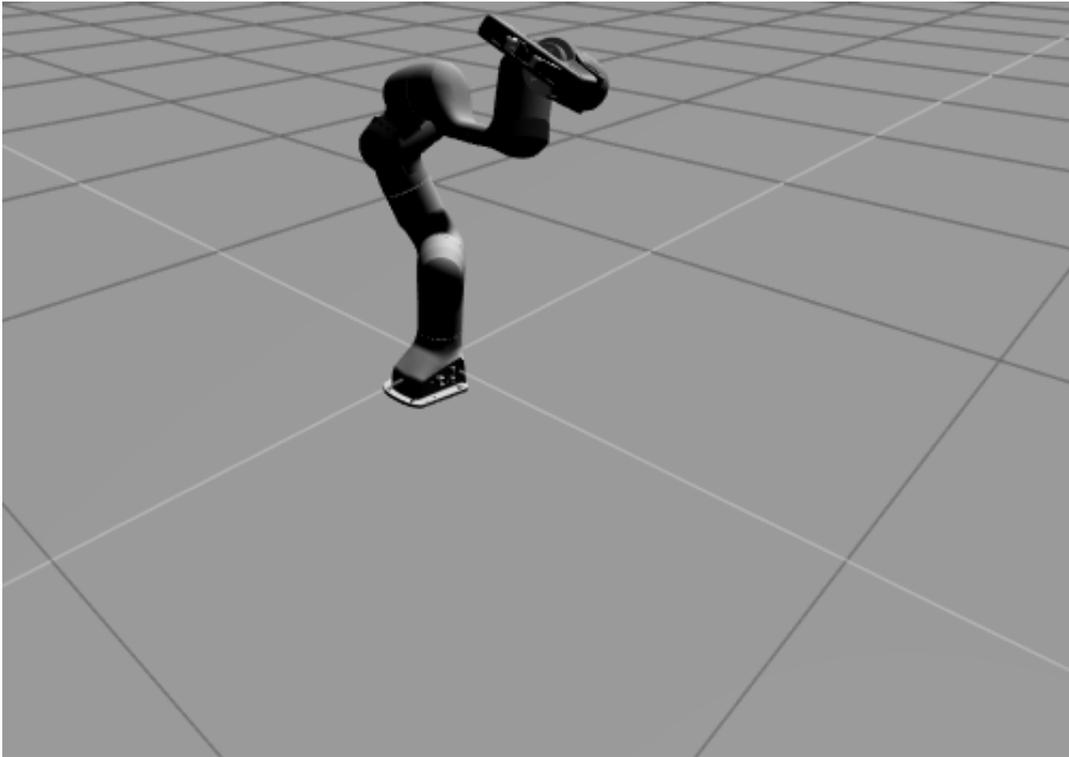
And go to **Planning Request** and change the **Planning group** hand to **panda_arm**



Now go to the **Planning** section, move the arm to a position you like and click on the **plan and execute** button.



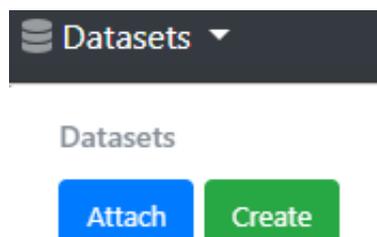
You will see how the simulation moves too.



4. How to create a dataset

Now let's see how to create a Dataset

go to the **Datasets** menu at the top of the window and select **create**



Fill all the data in order to create your dataset, the name of the dataset, the folder that will contain this dataset, some description , and select if you want that your dataset will be public or not

Create dataset

Create new dataset to save data files

Name

Folder to attach

Description

Is it public

Once you have already created, you will see that your new dataset is ready .

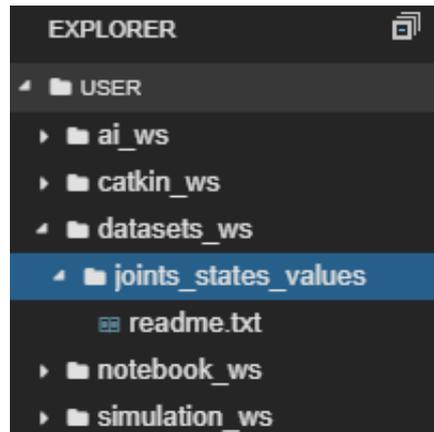
Datasets

joints_states_values (5.00 B)

(/home/user/datasets_ws/joints_states_values)

Ready! ✓

If you go to the IDE, you will see your new dataset in the **datasets_ws** ready to store all your data



Ok, now should be interesting to add some data to your new dataset, we will use **rosvbag** to do it, so open a new Shell and navigate to the dataset first.

```
In [ ]: $ cd datasets_ws/joints_states_values/
```

Then we will store all the information of a topic, in this case the topic called **/joint_states**, in order to do that run the following command in the Shell

```
In [ ]: $ rosvbag record /joint_states
```

This will record all the info of that topic, even if you move the robot again.

you can move the robot again while the rosvbag is recording

Once you want to stop the recording just press Ctrl +c in the shell to stop it

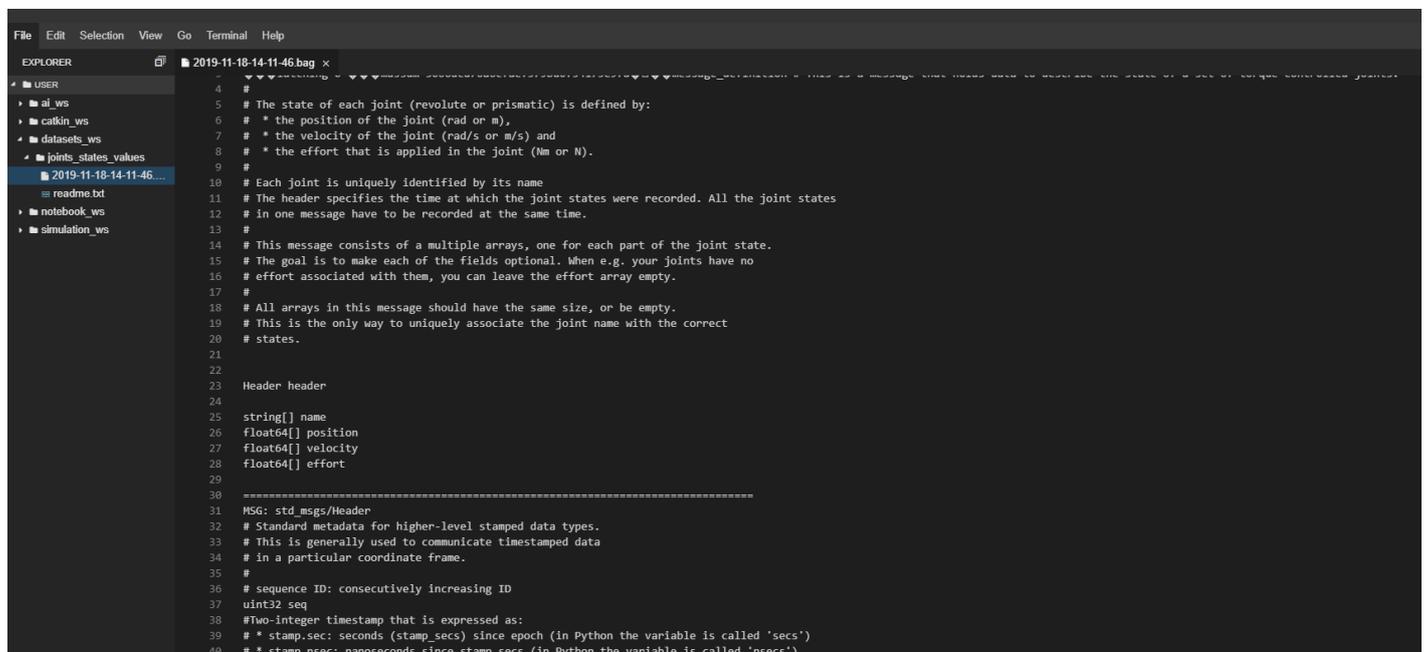
You can work with this information, for example let's see the info of the bag you have created, go to the Shell and run **\$ rosvbag info (the name of your bag)** you will have something similar to the next image.

```

path:          2019-11-18-14-11-46.bag
version:       2.0
duration:      32.6s
start:         Jan 01 1970 00:17:47.85 (1067.85)
end:           Jan 01 1970 00:18:20.48 (1100.48)
size:          747.9 KB
messages:      1633
compression:   none [1/1 chunks]
types:         sensor_msgs/JointState [3066dcd76a6cfaef579bd0f3417
3e9fd]
topics:        /joint_states 1633 msgs : sensor_msgs/JointSta
te

```

But, let's see how it looks like in the IDE



```

4 #
5 # The state of each joint (revolute or prismatic) is defined by:
6 # * the position of the joint (rad or m),
7 # * the velocity of the joint (rad/s or m/s) and
8 # * the effort that is applied in the joint (Nm or N).
9 #
10 # Each joint is uniquely identified by its name
11 # The header specifies the time at which the joint states were recorded. All the joint states
12 # in one message have to be recorded at the same time.
13 #
14 # This message consists of a multiple arrays, one for each part of the joint state.
15 # The goal is to make each of the fields optional. When e.g. your joints have no
16 # effort associated with them, you can leave the effort array empty.
17 #
18 # All arrays in this message should have the same size, or be empty.
19 # This is the only way to uniquely associate the joint name with the correct
20 # states.
21
22
23 Header header
24
25 string[] name
26 float64[] position
27 float64[] velocity
28 float64[] effort
29
30 =====
31 MSG: std_msgs/Header
32 # Standard metadata for higher-level stamped data types.
33 # This is generally used to communicate timestamped data
34 # in a particular coordinate frame.
35 #
36 # sequence ID: consecutively increasing ID
37 uint32 seq
38 #Two-integer timestamp that is expressed as:
39 # * stamp.sec: seconds (stamp_secs) since epoch (in Python the variable is called 'secs')
40 # * stamp.nsec: nanoseconds since stamp_secs (in Python the variable is called 'nsecs')

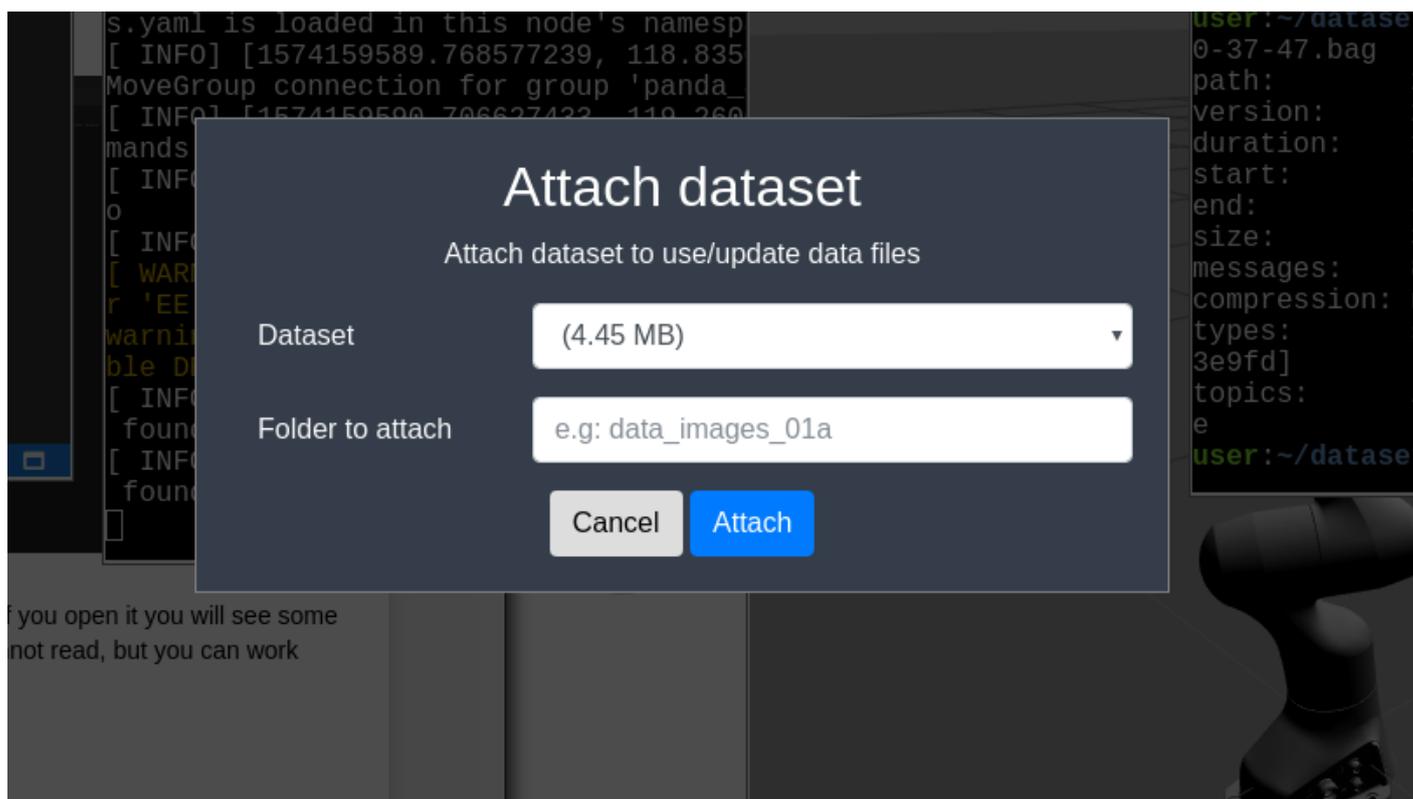
```

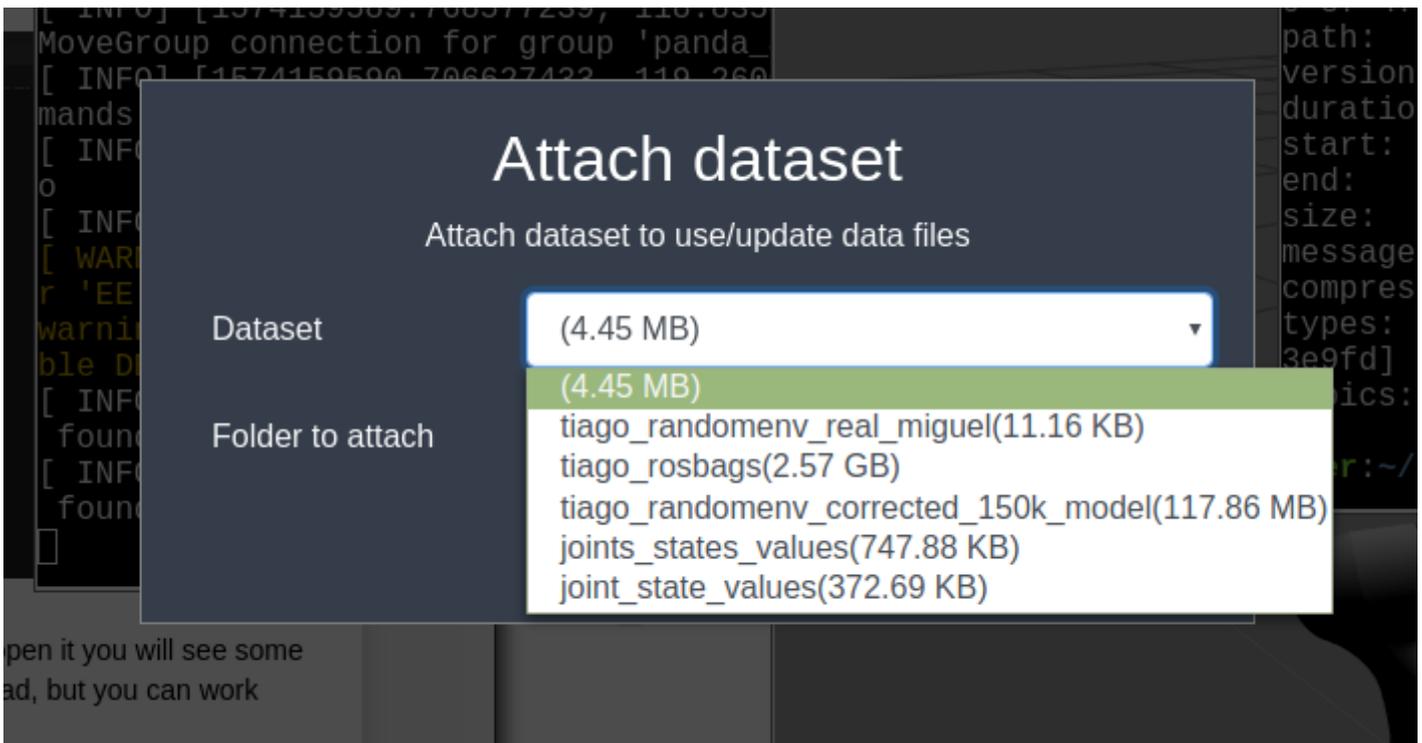
as you can see there is a new file, with the date of creation, and if you open it you will see some information, but most of the information is in other format that cannot read, but you can work with it.

You can attach other datasets to this rosject

If you created **datasets** in other rosjects, you can associate (attach) them with this rosject.

1. Go to the Datasets and press `Attach` .
2. Select the dataset you want to attach, on the `Dataset` field
3. Indicate the folder inside the `datasets_ws` that you want to put the new dataset
4. Press `Attach` . A new dataset will be created with that content, and the directory in the `datasets_ws` will be created.





5. Creating the documentation

In order of being organized all related to the documentation goes in the **notebook_ws**, here the notebooks that you will make in the jupyter notebook will be saved, also here the images that we will use in the documentation should be saved.

You should edit the **default notebook** or **create a new one**. Check Jupyter documentation to learn how.

In this Live Class, we are going to upload the notebook of this Live Class. You can find the notebook files in [this tar file \(http://www.theconstructsim.com/wp-content/uploads/2019/11/notebook_ws_Live_Class_74.tar\)](http://www.theconstructsim.com/wp-content/uploads/2019/11/notebook_ws_Live_Class_74.tar).

6. Save everything

Now that you have your whole rosject ready, you must save it.

If you press the `Save` icon, everything will be saved, including the datasets attached.

This means that you will be able to share this rosject with anybody, and it will contain all the things you already used there, including, **documentation, simulation, code and datasets**.

Now you are ready to **SHARE** your ROSject with anyone!

Mission completed!!

If you liked this video, please support us!

Really... we need your support!!!!

How can you support us?

1. Subscribe to our ROS online academy and become a Master of ROS Development

Go to our online academy. There is no faster way and funnier to learn ROS because we use the same method we did here.

We call the 30/70 method

- **30% of the time learning theory**
- **70% of the time practicing with simulated robots**

Search Paths, Courses and Live Classes ... Search

robotignite ACADEMY

Your learning
Continue your learning

Paths
Follow a guided path to ROS learning

Courses
Explore our Courses library

Live Classes
Attend a live/recorded ROS class

ROS Development Studio
Practise and share your knowledge

Accomplishments
View your certificates of completion

Most Popular

Basic ROS 5 Day(s) Python

ROS Basics in 5 Days
Learn the fundamentals of ROS to understand and be able to program robots.

Start Course Details

Basic ROS 5 Day(s) C++

ROS Basics in 5 Days (C++)
Learn the fundamentals of ROS to understand and be able to program robots.

Start Course Details

Basic ROS 5 Day(s) C++

ROS2 Basics for C++ in 5 days
Learn ROS2 basics now. It doesn't matter if you are new to ROS or a veteran, ROS2 is ...

Start Course Details

Basic ROS 5 Day(s) Python

Python 3 for Robotics
Master the basics of Python 3 for robot programming

Free Start Course Details

Basic ROS 5 Day(s) Python

Linux for Robotics
Learn the Linux fundamentals you'll need for robotics development

Free Start Course Details

Robot Creation 5 Day(s)

Your First Robot with ROS
Creating your first ROS based Robot from Scratch.

Artificial Intelligence 5 Day(s)

Deep Learning with Domain
Learn how to train any robot to recognize an object and pinpoint its 3D location with

Navigation 5 Day(s)

Fuse Sensor Data to Improve Localization
Learn how to fuse GPS, IMU, odometry and other sources of localization.

ROS Debug 5 Day(s)

Unit Testing with ROS
Learn how to perform Unit Tests with ROS on the 3 main levels of testing: Python tests.

Navigation 5 Day(s)

ROS Navigation in 5 Days
Learn how to make your robot navigate autonomously by using the ROS Navigation

GET HELP

<https://www.robotigniteacademy.com/en/course/unit-testing-ros/details/>

Check it out at <http://robotignite.academy> (<http://robotignite.academy>)

2. Buy one ROS Developers T-shirt!

	I'm a ROS developer €17.98 + additional fees Teespring		I'm a ROS developer €17.98 + additional fees Teespring		I'm a ROS developer €12.99 + additional fees Teespring		I'm a ROS developer €23.98 + additional fees Teespring
--	---	--	---	--	---	--	---

You can buy them at [our Teespring area \(https://teespring.com/stores/ros-developers\)](https://teespring.com/stores/ros-developers)

3. Give us a like in Youtube and subscribe to the channel

- Go to [our Youtube Channel \(https://www.youtube.com/channel/UCt6Lag-vv25fTX3e11mVY1Q\)](https://www.youtube.com/channel/UCt6Lag-vv25fTX3e11mVY1Q) and subscribe (it is free!!!)
- Give us a like to this video

KEEP PUSHING YOUR ROS LEARNING WITH PATIENCE AND GOOD HUMOUR!

Build the future, Become a ROS DEVELOPER