

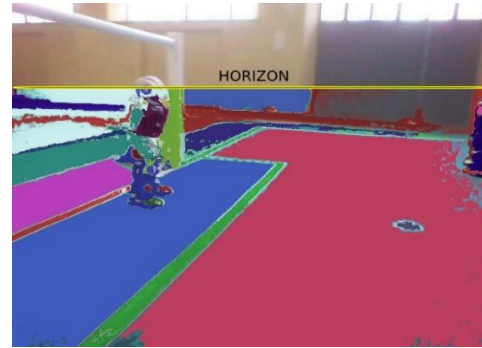
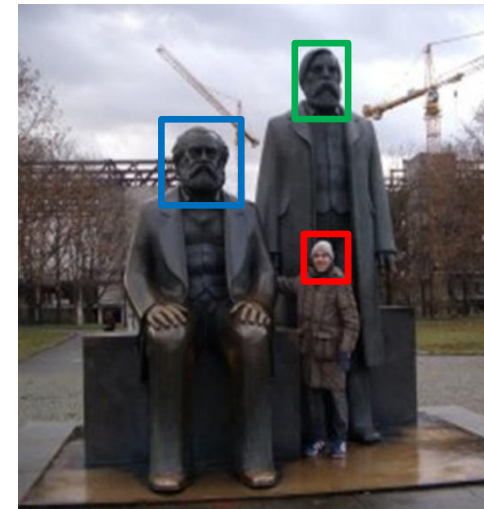
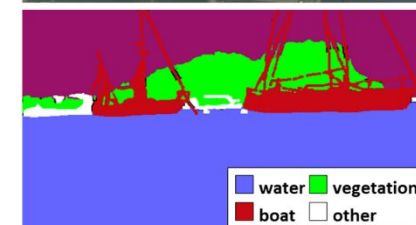


**UNIVERSITÀ DEGLI STUDI
DELLA BASILICATA**

Corso di Sistemi Informativi
A.A. 2018/19

Docente
Domenico Daniele Bloisi

git + **ROS (Python)**



Maggio 2019

git + ROS

Esempio pratico

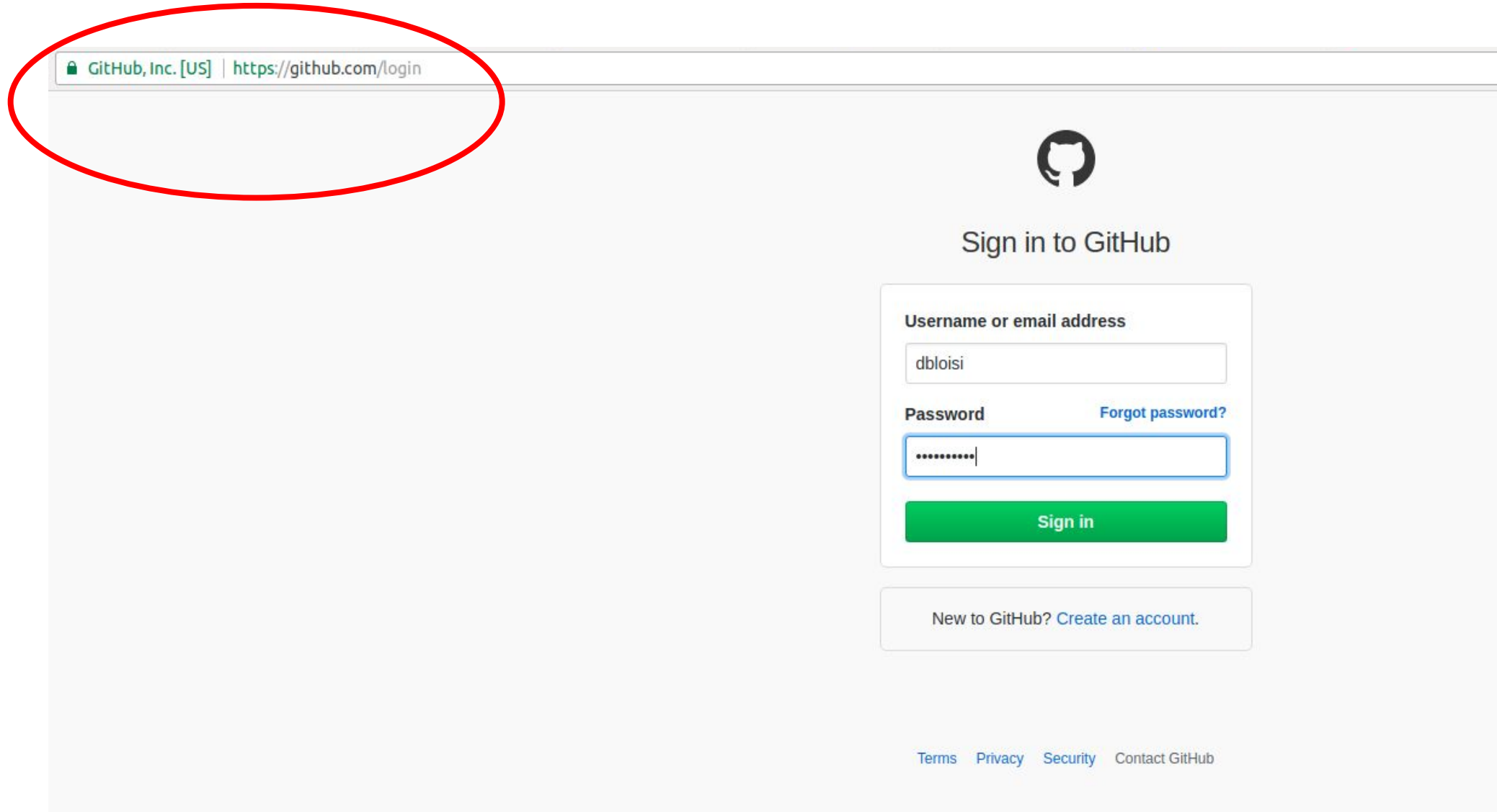
1. creare un repository git
2. creare un nodo ROS
3. condividere il nodo ROS
tramite il repository git
4. modificare il nodo ROS
usando git



+




Server git



The image shows a screenshot of the GitHub login page. A red circle highlights the browser's address bar, which contains the text "GitHub, Inc. [US] | https://github.com/login". The main content area features the GitHub logo at the top, followed by the heading "Sign in to GitHub". Below this is a login form with two input fields: "Username or email address" (containing "dbloisi") and "Password" (containing masked characters). A "Forgot password?" link is positioned to the right of the password field. A green "Sign in" button is located below the password field. At the bottom of the form, there is a link for "New to GitHub? Create an account.". The footer contains links for "Terms", "Privacy", "Security", and "Contact GitHub".

GitHub, Inc. [US] | https://github.com/login



Sign in to GitHub

Username or email address

dbloisi

Password [Forgot password?](#)

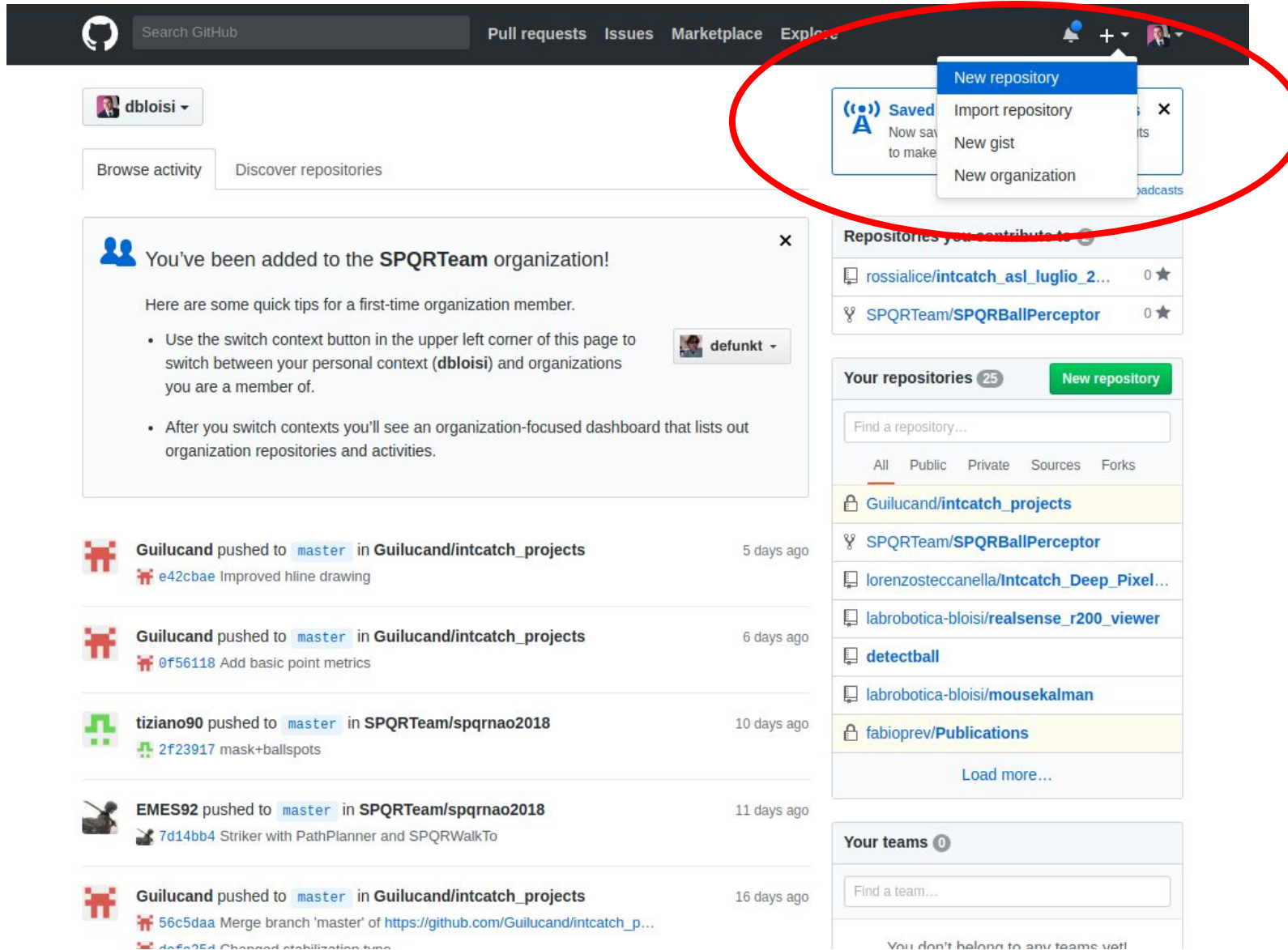
.....

Sign in

New to GitHub? [Create an account.](#)

[Terms](#) [Privacy](#) [Security](#) [Contact GitHub](#)

Creare un repository git



The screenshot shows the GitHub homepage for user **dbloisi**. A red circle highlights the top right navigation area, specifically the dropdown menu that appears when clicking the '+' icon. The menu options are:

- New repository
- Import repository
- New gist
- New organization

Below the navigation bar, there is a notification from the **SPQRTeam** organization stating: "You've been added to the SPQRTeam organization! Here are some quick tips for a first-time organization member."

The main content area displays a list of recent repository pushes:

- Guilucand** pushed to **master** in **Guilucand/intcatch_projects** 5 days ago. Commit: **e42cbae** Improved hline drawing.
- Guilucand** pushed to **master** in **Guilucand/intcatch_projects** 6 days ago. Commit: **0f56118** Add basic point metrics.
- tiziano90** pushed to **master** in **SPQRTeam/spqrnao2018** 10 days ago. Commit: **2f23917** mask+ballspots.
- EMES92** pushed to **master** in **SPQRTeam/spqrnao2018** 11 days ago. Commit: **7d14bb4** Striker with PathPlanner and SPQRWalkTo.
- Guilucand** pushed to **master** in **Guilucand/intcatch_projects** 16 days ago. Commit: **56c5daa** Merge branch 'master' of https://github.com/Guilucand/intcatch_projects.

On the right side, the "Your repositories" section shows 25 repositories, including **rossialice/intcatch_asl_luglio_2...** and **SPQRTeam/SPQRBallPerceptor**. The "Your teams" section shows 0 teams.

Repository name

Create a new repository

A repository contains all the files for your project, including the revision history.

Owner



Repository name

hello_ros ✓

Great repository names are short and memorable. Need inspiration? How about **furry-parakeet**.

Description (optional)

my first ros package



Public

Anyone can see this repository. You choose who can commit.



Private

You choose who can see and commit to this repository.



Initialize this repository with a README

This will let you immediately clone the repository to your computer. Skip this step if you're importing an existing repository.

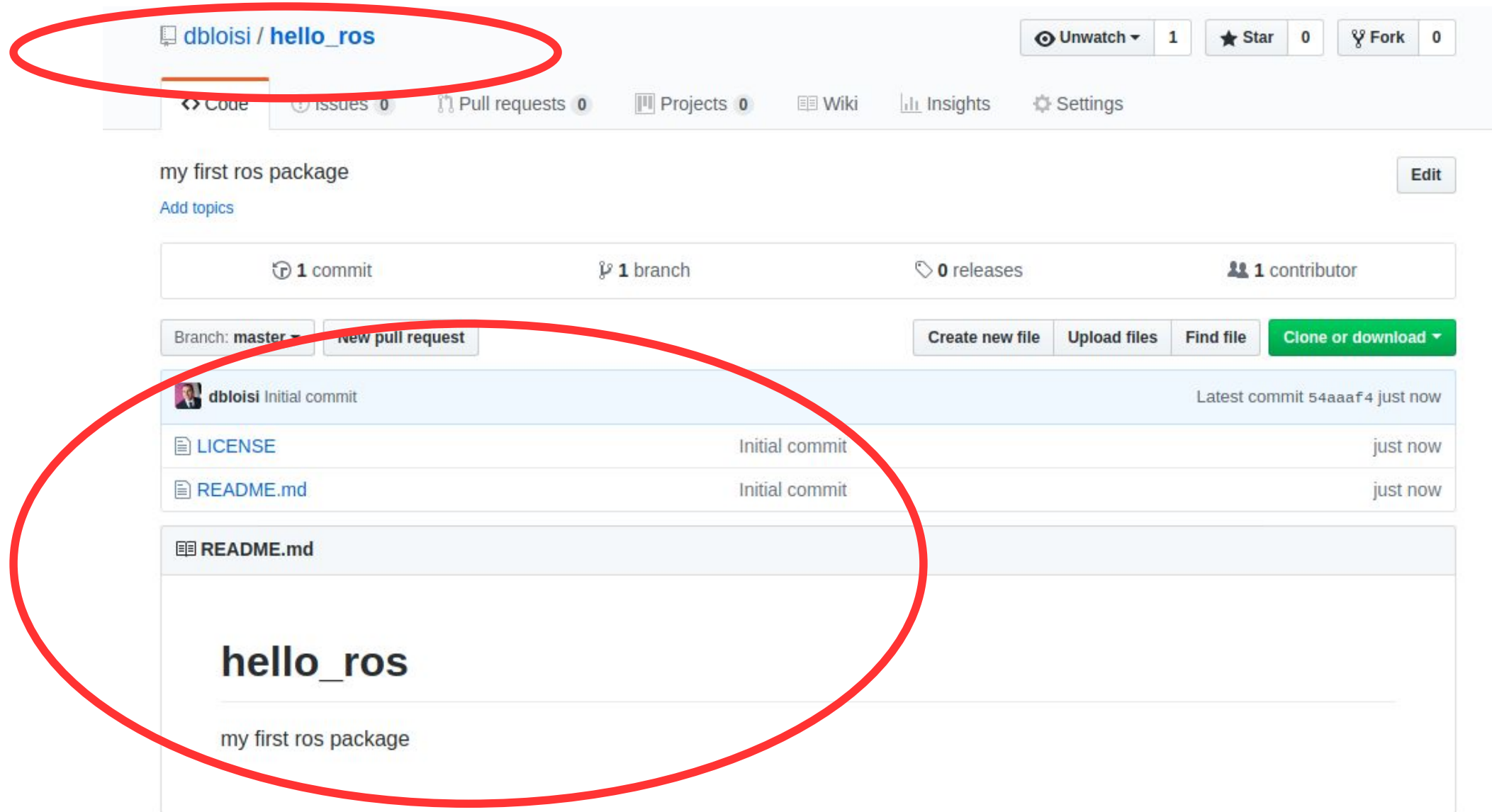
Add .gitignore: **None**

Add a license: **GNU General Public License v3.0**



Create repository

Repository creato



The screenshot shows a GitHub repository page for 'dbloisi / hello_ros'. The repository name is circled in red. Below the repository name, there are tabs for 'Code', 'Issues', 'Pull requests', 'Projects', 'Wiki', 'Insights', and 'Settings'. The repository description is 'my first ros package'. Below this, there are statistics: 1 commit, 1 branch, 0 releases, and 1 contributor. There are buttons for 'Branch: master', 'New pull request', 'Create new file', 'Upload files', 'Find file', and 'Clone or download'. The file list shows 'LICENSE' and 'README.md', both with 'Initial commit' and 'just now' timestamps. The 'README.md' file is expanded, showing the text 'hello_ros' and 'my first ros package'. The file list and the expanded README content are circled in red.

dbloisi / **hello_ros**

Unwatch 1 Star 0 Fork 0

Code Issues 0 Pull requests 0 Projects 0 Wiki Insights Settings

my first ros package [Add topics](#) [Edit](#)

1 commit 1 branch 0 releases 1 contributor

Branch: master [New pull request](#) [Create new file](#) [Upload files](#) [Find file](#) [Clone or download](#)

dbloisi Initial commit Latest commit 54aaaf4 just now

LICENSE	Initial commit	just now
README.md	Initial commit	just now

[README.md](#)

hello_ros

my first ros package

Indirizzo del repository remoto

The screenshot shows the GitHub interface for the repository 'dbloisi / hello_ros'. At the top, there are buttons for 'Unwatch' (1), 'Star' (0), and 'Fork' (0). Below this is a navigation bar with links for 'Code', 'Issues' (0), 'Pull requests' (0), 'Projects' (0), 'Wiki', 'Insights', and 'Settings'. The repository description is 'my first ros package' with an 'Edit' button. A summary bar shows '1 commit', '1 branch', '0 releases', and '1 contributor'. Below this is a section for the 'master' branch with a 'New pull request' button and a 'Clone or download' button. The 'Clone or download' button is highlighted with a red circle, and its dropdown menu is open, showing options for 'Clone with HTTPS' (with a question mark), 'Use SSH', and 'Download ZIP'. The 'Clone with HTTPS' option is selected, showing the URL 'https://github.com/dbloisi/hello_ros.' with a copy icon. Below the dropdown menu, the 'README.md' file is visible, showing the repository name 'hello_ros' and the description 'my first ros package'.

dbloisi / hello_ros

Unwatch 1 Star 0 Fork 0

Code Issues 0 Pull requests 0 Projects 0 Wiki Insights Settings

my first ros package Edit

Add topics

1 commit 1 branch 0 releases 1 contributor

Branch: master New pull request Create new file Upload files Find file Clone or download

dbloisi Initial commit

LICENSE Initial commit

README.md Initial commit

README.md

hello_ros

my first ros package

Clone with HTTPS ? Use SSH

Use Git or checkout with SVN using the web URL.

https://github.com/dbloisi/hello_ros. Copy

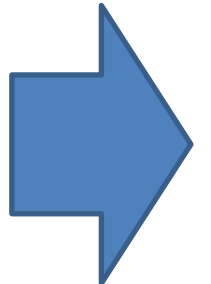
Download ZIP

Creazione del repository locale

Il repository remoto si trova in
https://github.com/dbloisi/hello_ros

Creiamo il repository locale nel nostro workspace ROS
`~/catkin_ws`

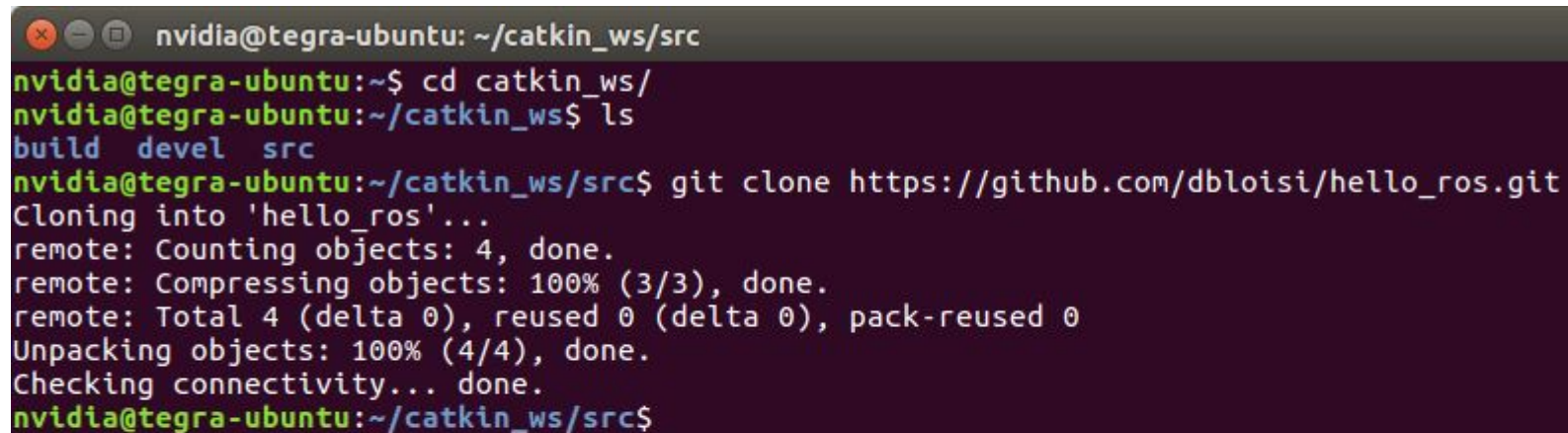
```
nvidia@tegra-ubuntu: ~/catkin_ws
nvidia@tegra-ubuntu:~$ cd catkin_ws/
nvidia@tegra-ubuntu:~/catkin_ws$ ls
build  devel  src
nvidia@tegra-ubuntu:~/catkin_ws$
```



Creazione del repository locale


Il repository remoto si trova in
https://github.com/dbloisi/hello_ros


Il repository locale sarà creato in
[~/catkin_ws/src/hello_ros](#)

A terminal window with a dark purple background and a title bar showing 'nvidia@tegra-ubuntu: ~/catkin_ws/src'. The terminal displays the following commands and output:

```
nvidia@tegra-ubuntu:~$ cd catkin_ws/  
nvidia@tegra-ubuntu:~/catkin_ws$ ls  
build  devel  src  
nvidia@tegra-ubuntu:~/catkin_ws/src$ git clone https://github.com/dbloisi/hello_ros.git  
Cloning into 'hello_ros'...  
remote: Counting objects: 4, done.  
remote: Compressing objects: 100% (3/3), done.  
remote: Total 4 (delta 0), reused 0 (delta 0), pack-reused 0  
Unpacking objects: 100% (4/4), done.  
Checking connectivity... done.  
nvidia@tegra-ubuntu:~/catkin_ws/src$
```

Creating a ROS package

 wiki.ros.org/ROS/Tutorials/CreatingPackage

 ROS.org


[About](#) | [Support](#) | [Discussion Forum](#) | [Service Status](#) | [Q&A answers.ros.org](#)

Search:

Documentation Browse Software News Download

ROS/ Tutorials/ CreatingPackage

Note: This tutorial assumes that you have completed the previous tutorials: [navigating the ROS filesystem](#).

 Please ask about problems and questions regarding this tutorial on [answers.ros.org](#). Don't forget to include in your question the link to this page, the versions of your OS & ROS, and also add appropriate tags.

Creating a ROS Package

Description: This tutorial covers using [roscat](#) or [catkin](#) to create a new package, and [roscat](#) to list package dependencies.

Tutorial Level: BEGINNER

Next Tutorial: [Building a ROS package](#)

catkin rosbld

Contents

1. What makes up a catkin Package?
2. Packages in a catkin Workspace
3. Creating a catkin Package
4. Building a catkin workspace and sourcing the setup file
5. package dependencies
 1. First-order dependencies
 2. Indirect dependencies
6. Customizing Your Package
 1. Customizing the package.xml
 1. description tag
 2. maintainer tags
 3. license tags
 4. dependencies tags
 5. Final package.xml
 2. Customizing the CMakeLists.txt

Wiki

- [Distributions](#)
- [ROS/Installation](#)
- [ROS/Tutorials](#)
- [RecentChanges](#)
- [CreatingPackage](#)

Page

- [Immutable Page](#)
- [Info](#)
- [Attachments](#)
- [More Actions:](#) ▼

User

- [Login](#)

<http://wiki.ros.org/ROS/Tutorials/CreatingPackage>

1. What makes up a catkin Package?

For a package to be considered a catkin package it must meet a few requirements:

- The package must contain a [catkin compliant package.xml](#) file.

catkin_create_pkg

```
nvidia@tegra-ubuntu: ~/catkin_ws/src
nvidia@tegra-ubuntu:~/catkin_ws/src$ catkin_create_pkg hello_ros std_msgs rospy roscpp
```

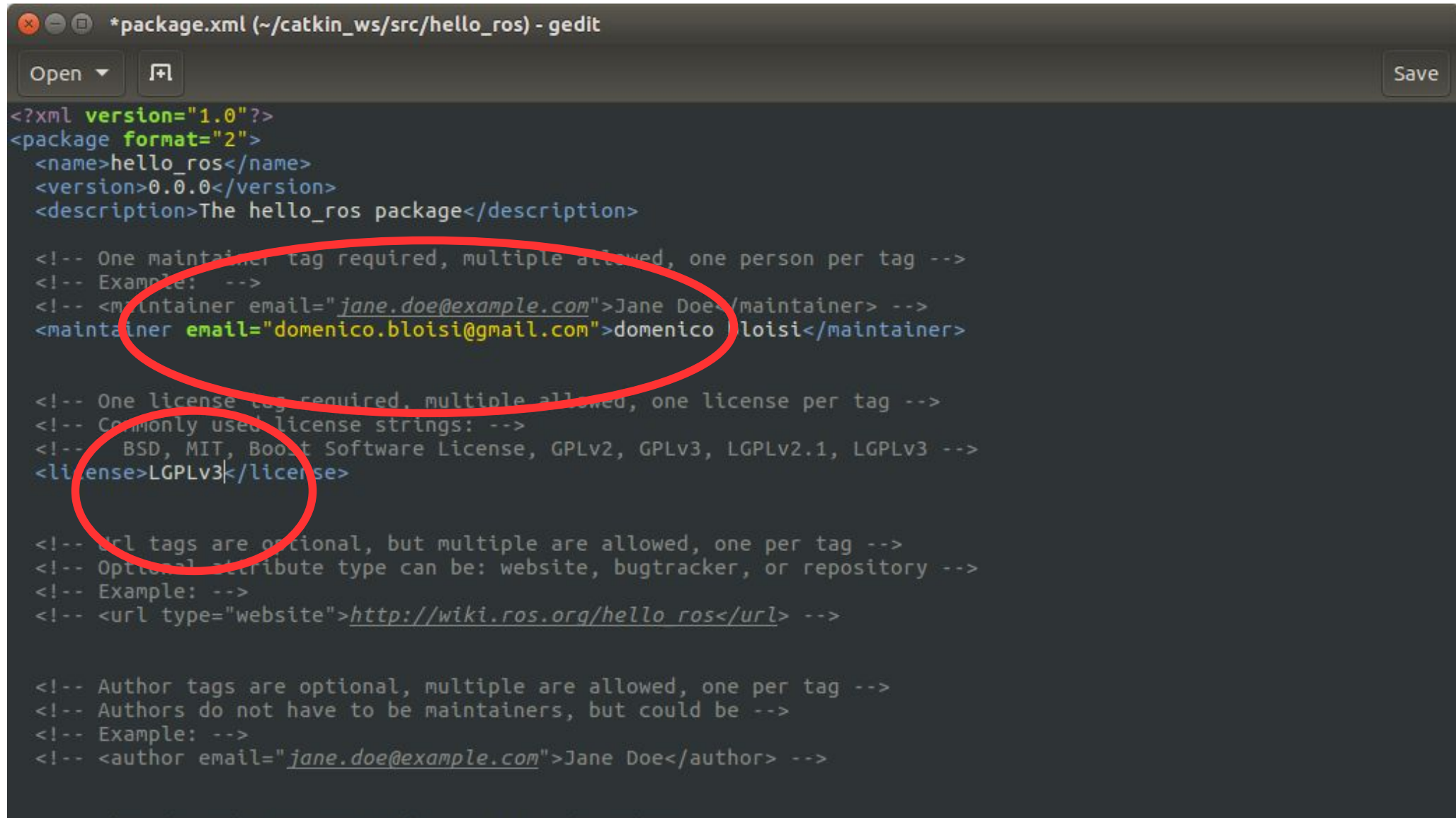


```
nvidia@tegra-ubuntu: ~/catkin_ws/src
nvidia@tegra-ubuntu:~/catkin_ws/src$ catkin_create_pkg hello_ros std_msgs rospy roscpp
Created file hello_ros/package.xml
Created file hello_ros/CMakeLists.txt
Created folder hello_ros/include/hello_ros
Created folder hello_ros/src
Successfully created files in /home/nvidia/catkin_ws/src/hello_ros. Please adjust the values in package.xml.
nvidia@tegra-ubuntu:~/catkin_ws/src$
```

package.xml

```
nvidia@tegra-ubuntu: ~/catkin_ws/src/hello_ros
nvidia@tegra-ubuntu:~/catkin_ws/src$ catkin_create_pkg hello_ros std_msgs rospy roscpp
Created file hello_ros/package.xml
Created file hello_ros/CMakeLists.txt
Created folder hello_ros/include/hello_ros
Created folder hello_ros/src
Successfully created files in /home/nvidia/catkin_ws/src/hello_ros. Please adjust the values in package.xml.
nvidia@tegra-ubuntu:~/catkin_ws/src$ cd hello_ros
nvidia@tegra-ubuntu:~/catkin_ws/src/hello_ros$ gedit package.xml
nvidia@tegra-ubuntu:~/catkin_ws/src/hello_ros$
```

Inserimento dati in package.xml



```
*package.xml (~/catkin_ws/src/hello_ros) - gedit
Open [icon] Save

<?xml version="1.0"?>
<package format="2">
  <name>hello_ros</name>
  <version>0.0.0</version>
  <description>The hello_ros package</description>

  <!-- One maintainer tag required, multiple allowed, one person per tag -->
  <!-- Example: -->
  <!-- <maintainer email="jane.doe@example.com">Jane Doe</maintainer> -->
  <maintainer email="domenico.bloisi@gmail.com">domenico bloisi</maintainer>

  <!-- One license tag required, multiple allowed, one license per tag -->
  <!-- Commonly used license strings: -->
  <!-- BSD, MIT, Boost Software License, GPLv2, GPLv3, LGPLv2.1, LGPLv3 -->
  <license>LGPLv3</license>

  <!-- Url tags are optional, but multiple are allowed, one per tag -->
  <!-- Optional attribute type can be: website, bugtracker, or repository -->
  <!-- Example: -->
  <!-- <url type="website">http://wiki.ros.org/hello_ros</url> -->

  <!-- Author tags are optional, multiple are allowed, one per tag -->
  <!-- Authors do not have to be maintainers, but could be -->
  <!-- Example: -->
  <!-- <author email="jane.doe@example.com">Jane Doe</author> -->
```


Dipendenze in package.xml

```
<!-- Examples: -->
<!-- Use depend as a shortcut for packages that are both build and exec dependencies -->
<!--   <depend>roscpp</depend> -->
<!--   Note that this is equivalent to the following: -->
<!--   <build_depend>roscpp</build_depend> -->
<!--   <exec_depend>roscpp</exec_depend> -->
<!-- Use build_depend for packages you need at compile time: -->
<!--   <build_depend>message_generation</build_depend> -->
<!-- Use build_export_depend for packages you need in order to build against this package: -->
<!--   <build_export_depend>message_generation</build_export_depend> -->
<!-- Use buildtool_depend for build tool packages: -->
<!--   <buildtool_depend>catkin</buildtool_depend> -->
<!-- Use exec_depend for packages you need at runtime: -->
<!--   <exec_depend>message_runtime</exec_depend> -->
<!-- Use test_depend for packages you need only for testing: -->
<!--   <test_depend>gtest</test_depend> -->
<!-- Use doc_depend for packages you need only for building documentation: -->
<!--   <doc_depend>doxygen</doc_depend> -->
<buildtool_depend>catkin</buildtool_depend>
<build_depend>roscpp</build_depend>
<build_depend>rospy</build_depend>
<build_depend>std_msgs</build_depend>
<build_export_depend>roscpp</build_export_depend>
<build_export_depend>rospy</build_export_depend>
<build_export_depend>std_msgs</build_export_depend>
<exec_depend>roscpp</exec_depend>
<exec_depend>rospy</exec_depend>
<exec_depend>std_msgs</exec_depend>

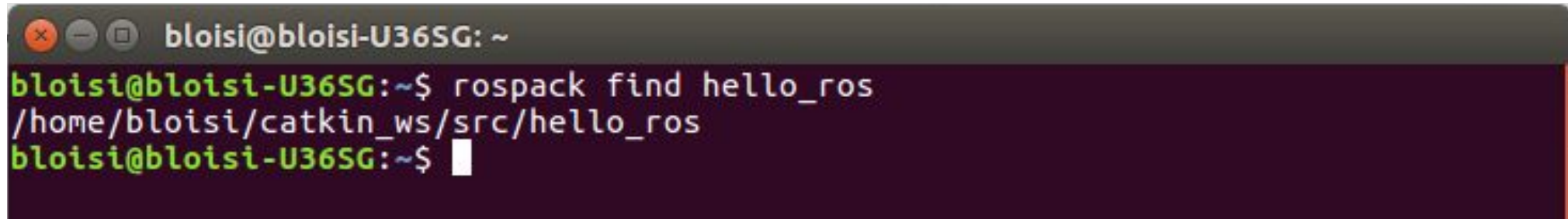
<!-- The export tag contains other, unspecified, tags -->
<export>
  <!-- Other tools can request additional information be placed here -->

</export>
</package>
```

Finding a ROS package

Now that your package has a manifest, ROS can find it. Try executing the command:

```
rospack find hello_ros
```

A terminal window with a dark background and light green text. The window title is 'bloisi@bloisi-U36SG: ~'. The prompt is 'bloisi@bloisi-U36SG:~\$'. The command 'rospack find hello_ros' has been entered, and the output is '/home/bloisi/catkin_ws/src/hello_ros'. The prompt is now 'bloisi@bloisi-U36SG:~\$' with a cursor.

```
bloisi@bloisi-U36SG: ~  
bloisi@bloisi-U36SG:~$ rospack find hello_ros  
/home/bloisi/catkin_ws/src/hello_ros  
bloisi@bloisi-U36SG:~$
```

if ROS is set up correctly you should see the physical location where your package is stored

<http://wiki.ros.org/ROS/Tutorials/Creating%20a%20Package%20by%20Hand>

Esempio Publisher/Subscriber Python

The screenshot shows the ROS.org website with the URL `wiki.ros.org/ROS/Tutorials/WritingPublisherSubscriber%28python%29` in the browser address bar. The page features the ROS.org logo, navigation links (About, Support, Discussion Forum, Service Status, Q&A answers.ros.org), and a search bar. A dark blue navigation bar contains links for Documentation, Browse Software, News, and Download. The main content area displays the breadcrumb `ROS/ Tutorials/ WritingPublisherSubscriber(python)` and a note stating that the tutorial assumes completion of previous tutorials. A yellow lightbulb icon prompts users to ask questions on `answers.ros.org`. The title `Writing a Simple Publisher and Subscriber (Python)` is prominently displayed, followed by a description: 'This tutorial covers how to write a publisher and subscriber node in python.' The tutorial level is marked as 'BEGINNER', and the next tutorial is 'Examining the simple publisher and subscriber'. Two tabs, 'catkin' and 'roscpp', are visible, with 'catkin' selected. A 'Contents' section lists the tutorial's structure: '1. Writing the Publisher Node' (with sub-items '1. The Code' and '2. The Code Explained') and '2. Writing the Subscriber Node' (with sub-items '1. The Code' and '2. The Code Explained'). On the right, a sidebar provides additional links under 'Wiki' (Distributions, ROS/Installation, ROS/Tutorials, RecentChanges, WritingPubl...ber(python)), 'Page' (Immutable Page, Info, Attachments, More Actions), and 'User' (Login).

← → ↻ ⓘ wiki.ros.org/ROS/Tutorials/WritingPublisherSubscriber%28python%29 ☆ ⋮

ROS.org About | Support | Discussion Forum | Service Status | Q&A answers.ros.org Search: Submit

Documentation Browse Software News Download

ROS/ Tutorials/ WritingPublisherSubscriber(python)

Note: This tutorial assumes that you have completed the previous tutorials: creating a ROS msg and srv.

💡 Please ask about problems and questions regarding this tutorial on answers.ros.org. Don't forget to include in your question the link to this page, the versions of your OS & ROS, and also add appropriate tags.

Writing a Simple Publisher and Subscriber (Python)

Description: This tutorial covers how to write a publisher and subscriber node in python.

Tutorial Level: BEGINNER

Next Tutorial: [Examining the simple publisher and subscriber](#)

catkin roscpp

Contents

- 1. Writing the Publisher Node
 - 1. The Code
 - 2. The Code Explained
- 2. Writing the Subscriber Node
 - 1. The Code
 - 2. The Code Explained

Wiki

- Distributions
- ROS/Installation
- ROS/Tutorials
- RecentChanges
- WritingPubl...ber(python)

Page

- Immutable Page
- Info
- Attachments
- More Actions: ▼

User

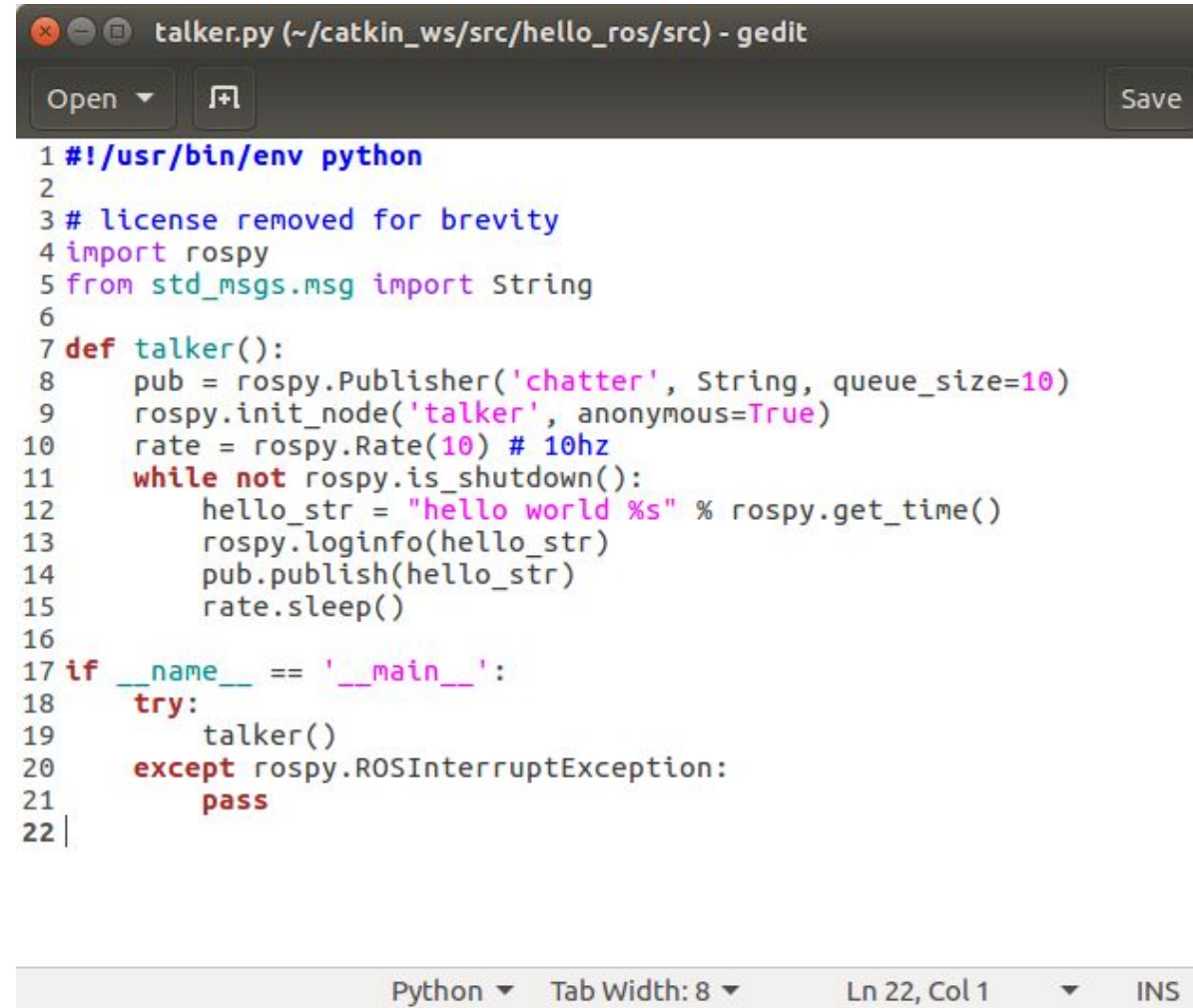
- Login

<http://wiki.ros.org/ROS/Tutorials/WritingPublisherSubscriber%28python%29>

Creiamo il publisher (talker.py)

```
bloisi@bloisi-U36SG: ~/catkin_ws/src/hello_ros/src
bloisi@bloisi-U36SG:~$ rospack find hello_ros
/home/bloisi/catkin_ws/src/hello_ros
bloisi@bloisi-U36SG:~$ cd /home/bloisi/catkin_ws/src/hello_ros
bloisi@bloisi-U36SG:~/catkin_ws/src/hello_ros$ ls
CMakeLists.txt  hello-ros.pdf  images  LICENSE  package.xml  README.md  src
bloisi@bloisi-U36SG:~/catkin_ws/src/hello_ros$ cd src/
bloisi@bloisi-U36SG:~/catkin_ws/src/hello_ros/src$ gedit talker.py
█
```

Codice del publisher (talker.py)



```
talker.py (~/catkin_ws/src/hello_ros/src) - gedit
Open ▾  [+1]  Save

1 #!/usr/bin/env python
2
3 # license removed for brevity
4 import rospy
5 from std_msgs.msg import String
6
7 def talker():
8     pub = rospy.Publisher('chatter', String, queue_size=10)
9     rospy.init_node('talker', anonymous=True)
10    rate = rospy.Rate(10) # 10hz
11    while not rospy.is_shutdown():
12        hello_str = "hello world %s" % rospy.get_time()
13        rospy.loginfo(hello_str)
14        pub.publish(hello_str)
15        rate.sleep()
16
17 if __name__ == '__main__':
18     try:
19         talker()
20     except rospy.ROSInterruptException:
21         pass
22 |
```

Python ▾ Tab Width: 8 ▾ Ln 22, Col 1 ▾ INS

https://raw.githubusercontent.com/ros/ros_tutorials/kinetic-devel/rospy_tutorials/001_talker_listener/talker.py

publisher (talker.py) eseguibile

```
bloisi@bloisi-U36SG: ~/catkin_ws/src/hello_ros/src
bloisi@bloisi-U36SG:~$ rospack find hello_ros
/home/bloisi/catkin_ws/src/hello_ros
bloisi@bloisi-U36SG:~$ cd /home/bloisi/catkin_ws/src/hello_ros
bloisi@bloisi-U36SG:~/catkin_ws/src/hello_ros$ ls
CMakeLists.txt  hello-ros.pdf  images  LICENSE  package.xml  README.md  src
bloisi@bloisi-U36SG:~/catkin_ws/src/hello_ros$ cd src/
bloisi@bloisi-U36SG:~/catkin_ws/src/hello_ros/src$ gedit talker.py
bloisi@bloisi-U36SG:~/catkin_ws/src/hello_ros/src$ chmod +x talker.py
bloisi@bloisi-U36SG:~/catkin_ws/src/hello_ros/src$
```

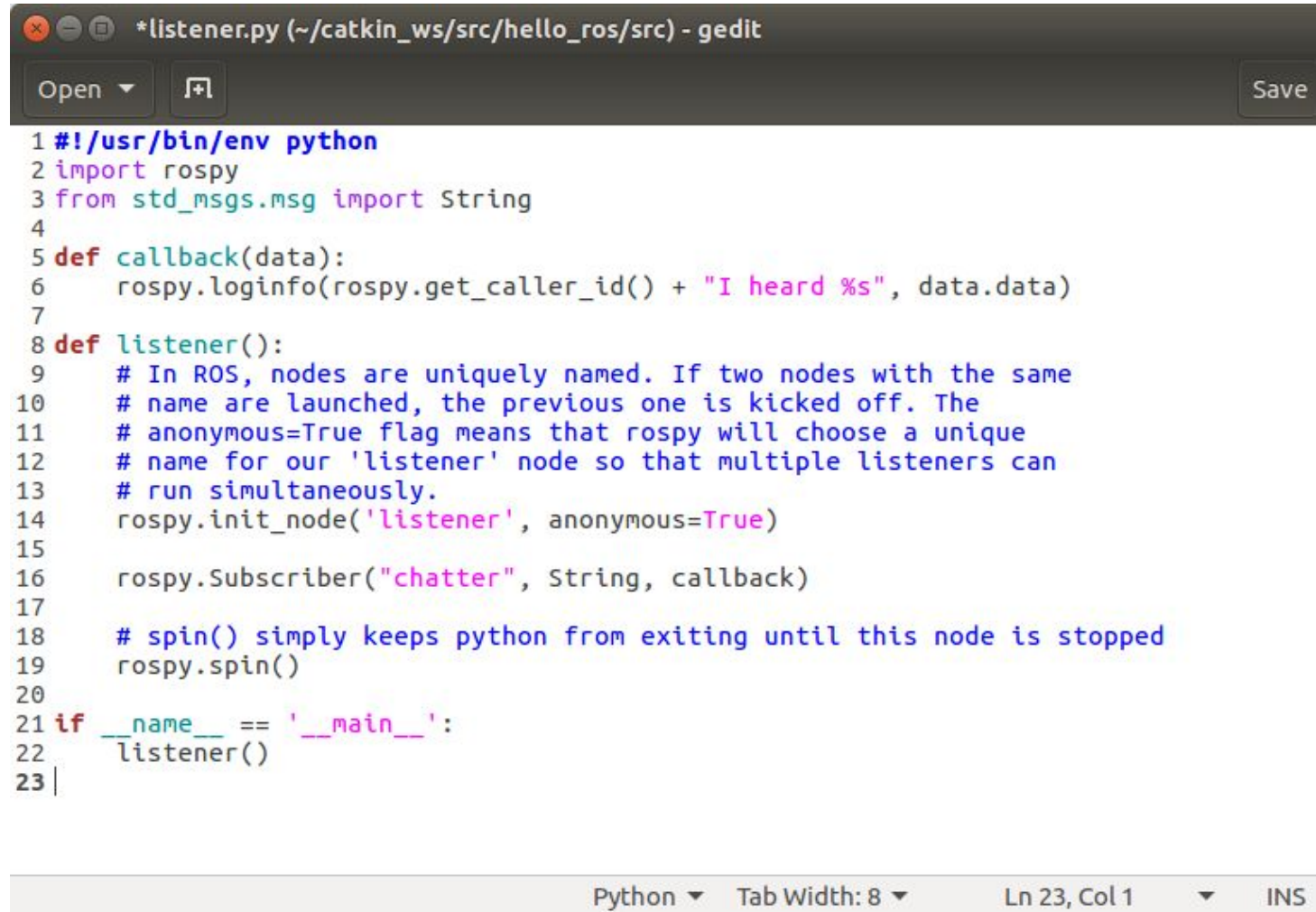
https://raw.githubusercontent.com/ros/ros_tutorials/kinetic-devel/rospy_tutorials/001_talker_listener/talker.py

Creiamo il subscriber (listener.py)

```
bloisi@bloisi-U36SG: ~/catkin_ws/src/hello_ros/src
bloisi@bloisi-U36SG:~$ rospack find hello_ros
/home/bloisi/catkin_ws/src/hello_ros
bloisi@bloisi-U36SG:~$ cd /home/bloisi/catkin_ws/src/hello_ros
bloisi@bloisi-U36SG:~/catkin_ws/src/hello_ros$ ls
CMakeLists.txt  hello-ros.pdf  images  LICENSE  package.xml  README.md  src
bloisi@bloisi-U36SG:~/catkin_ws/src/hello_ros$ cd src/
bloisi@bloisi-U36SG:~/catkin_ws/src/hello_ros/src$ gedit talker.py
bloisi@bloisi-U36SG:~/catkin_ws/src/hello_ros/src$ chmod +x talker.py
bloisi@bloisi-U36SG:~/catkin_ws/src/hello_ros/src$ gedit listener.py

```

Codice del subscriber (listener.py)



```
*listener.py (~/catkin_ws/src/hello_ros/src) - gedit
Open Save

1 #!/usr/bin/env python
2 import rospy
3 from std_msgs.msg import String
4
5 def callback(data):
6     rospy.loginfo(rospy.get_caller_id() + "I heard %s", data.data)
7
8 def listener():
9     # In ROS, nodes are uniquely named. If two nodes with the same
10    # name are launched, the previous one is kicked off. The
11    # anonymous=True flag means that rospy will choose a unique
12    # name for our 'listener' node so that multiple listeners can
13    # run simultaneously.
14    rospy.init_node('listener', anonymous=True)
15
16    rospy.Subscriber("chatter", String, callback)
17
18    # spin() simply keeps python from exiting until this node is stopped
19    rospy.spin()
20
21 if __name__ == '__main__':
22     listener()
23 |
```

Python Tab Width: 8 Ln 23, Col 1 INS

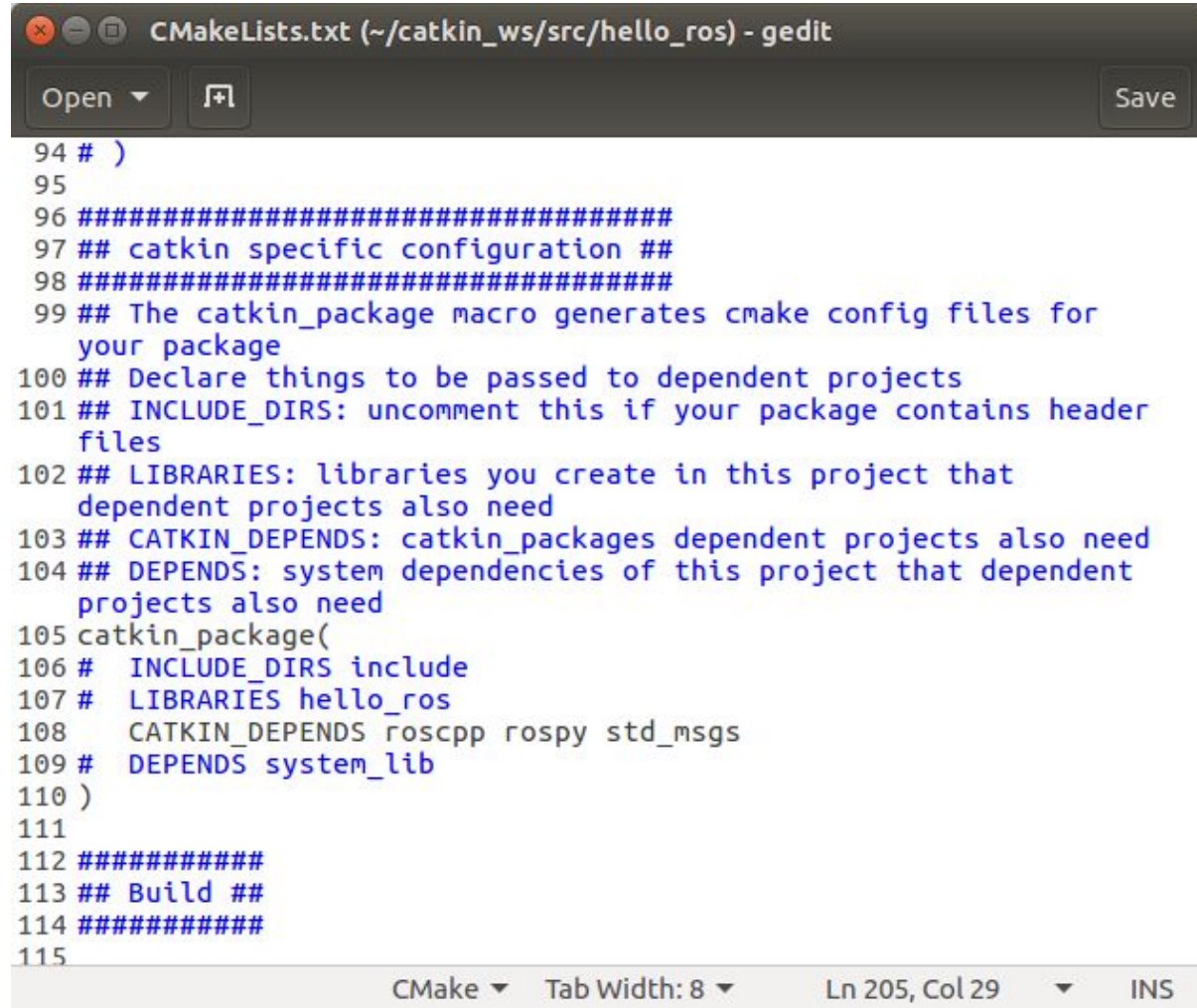
https://raw.githubusercontent.com/ros/ros_tutorials/kinetic-devel/rospy_tutorials/001_talker_listener/listener.py

Creiamo il subscriber (listener.py)

```
bloisi@bloisi-U36SG: ~/catkin_ws/src/hello_ros/src
bloisi@bloisi-U36SG:~$ rospack find hello_ros
/home/bloisi/catkin_ws/src/hello_ros
bloisi@bloisi-U36SG:~$ cd /home/bloisi/catkin_ws/src/hello_ros
bloisi@bloisi-U36SG:~/catkin_ws/src/hello_ros$ ls
CMakeLists.txt  hello-ros.pdf  images  LICENSE  package.xml  README.md  src
bloisi@bloisi-U36SG:~/catkin_ws/src/hello_ros$ cd src/
bloisi@bloisi-U36SG:~/catkin_ws/src/hello_ros/src$ gedit talker.py
bloisi@bloisi-U36SG:~/catkin_ws/src/hello_ros/src$ chmod +x talker.py
bloisi@bloisi-U36SG:~/catkin_ws/src/hello_ros/src$ gedit listener.py
bloisi@bloisi-U36SG:~/catkin_ws/src/hello_ros/src$ chmod +x listener.py
bloisi@bloisi-U36SG:~/catkin_ws/src/hello_ros/src$
```


Compiliamo il package hello_ros

Modifichiamo il file CMakeLists.txt in modo da poter compilare il package hello_ros contenente i due nodi talker e listener



```
CMakeLists.txt (~/.catkin_ws/src/hello_ros) - gedit
Open  Save
94 # )
95
96 #####
97 ## catkin specific configuration ##
98 #####
99 ## The catkin_package macro generates cmake config files for
your package
100 ## Declare things to be passed to dependent projects
101 ## INCLUDE_DIRS: uncomment this if your package contains header
files
102 ## LIBRARIES: libraries you create in this project that
dependent projects also need
103 ## CATKIN_DEPENDS: catkin_packages dependent projects also need
104 ## DEPENDS: system dependencies of this project that dependent
projects also need
105 catkin_package(
106 # INCLUDE_DIRS include
107 # LIBRARIES hello_ros
108 CATKIN_DEPENDS roscpp rospy std_msgs
109 # DEPENDS system_lib
110 )
111
112 #####
113 ## Build ##
114 #####
115
CMake  Tab Width: 8  Ln 205, Col 29  INS
```

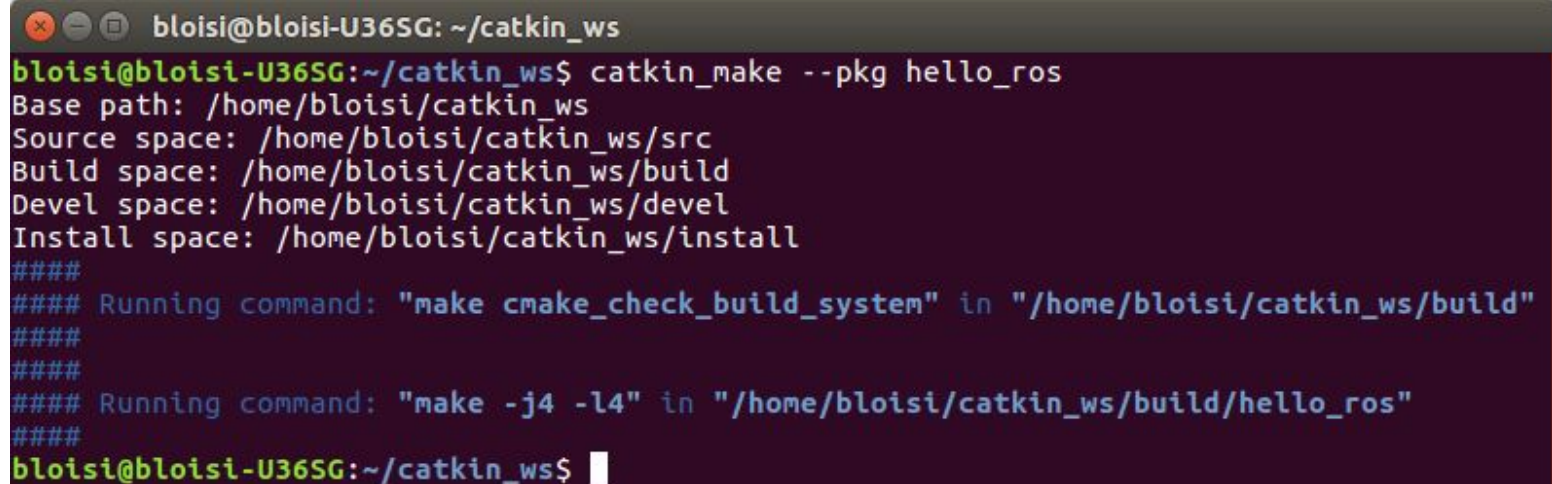

CMakeLists.txt

We need the CMakeLists.txt file so that catkin_make, which uses CMake for its more powerful flexibility when building across multiple platforms, builds the package



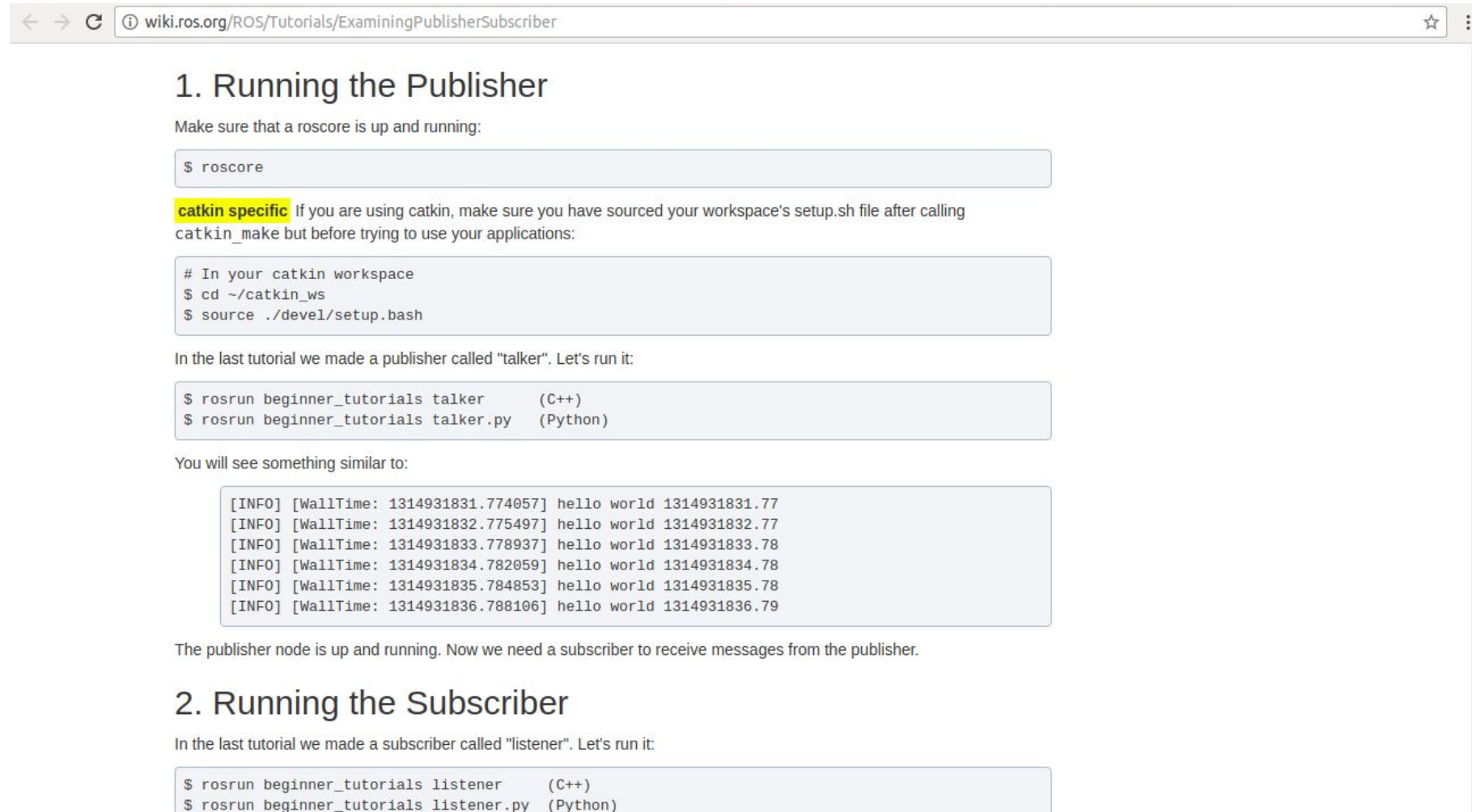
Compilazione con catkin_make

```
catkin_make --pkg hello_ros
```

A terminal window with a dark purple background and light green text. The window title is 'bloisi@bloisi-U36SG: ~/catkin_ws'. The user has entered the command 'catkin_make --pkg hello_ros'. The terminal output shows the base path, source space, build space, devel space, and install space. It then shows two 'Running command' lines: 'make cmake_check_build_system' in the build directory and 'make -j4 -l4' in the build directory for the 'hello_ros' package. The prompt returns to the user.

```
bloisi@bloisi-U36SG: ~/catkin_ws
bloisi@bloisi-U36SG:~/catkin_ws$ catkin_make --pkg hello_ros
Base path: /home/bloisi/catkin_ws
Source space: /home/bloisi/catkin_ws/src
Build space: /home/bloisi/catkin_ws/build
Devel space: /home/bloisi/catkin_ws/devel
Install space: /home/bloisi/catkin_ws/install
####
#### Running command: "make cmake_check_build_system" in "/home/bloisi/catkin_ws/build"
####
####
#### Running command: "make -j4 -l4" in "/home/bloisi/catkin_ws/build/hello_ros"
####
bloisi@bloisi-U36SG:~/catkin_ws$
```

Esecuzione del nodo talker



The screenshot shows a web browser window with the address bar displaying `wiki.ros.org/ROS/Tutorials/ExaminingPublisherSubscriber`. The page content is as follows:

1. Running the Publisher

Make sure that a roscore is up and running:

```
$ roscore
```

catkin specific If you are using catkin, make sure you have sourced your workspace's setup.sh file after calling `catkin_make` but before trying to use your applications:

```
# In your catkin workspace
$ cd ~/catkin_ws
$ source ./devel/setup.bash
```

In the last tutorial we made a publisher called "talker". Let's run it:

```
$ rosrun beginner_tutorials talker      (C++)
$ rosrun beginner_tutorials talker.py  (Python)
```

You will see something similar to:

```
[INFO] [WallTime: 1314931831.774057] hello world 1314931831.77
[INFO] [WallTime: 1314931832.775497] hello world 1314931832.77
[INFO] [WallTime: 1314931833.778937] hello world 1314931833.78
[INFO] [WallTime: 1314931834.782059] hello world 1314931834.78
[INFO] [WallTime: 1314931835.784853] hello world 1314931835.78
[INFO] [WallTime: 1314931836.788106] hello world 1314931836.79
```

The publisher node is up and running. Now we need a subscriber to receive messages from the publisher.

2. Running the Subscriber

In the last tutorial we made a subscriber called "listener". Let's run it:

```
$ rosrun beginner_tutorials listener    (C++)
$ rosrun beginner_tutorials listener.py (Python)
```

roscore + rosrun

Apriamo un terminale e lanciamo `roscore`

```
roscore http://localhost:11311/
bloisi@bloisi-U36SG:~$ roscore
... logging to /home/bloisi/.ros/log/d8f5ca3a-6a54-11e9-953a-dc85de574b1d/roslau
nch-bloisi-U36SG-12291.log
Checking log directory for disk usage. This may take awhile.
Press Ctrl-C to interrupt
Done checking log file disk usage. Usage is <1GB.

started roslaunch server http://localhost:34051/
ros_comm version 1.12.14

SUMMARY
=====

PARAMETERS
* /rostdistro: kinetic
* /rosversion: 1.12.14

NODES

auto-starting new master
process[master]: started with pid [12303]
ROS_MASTER_URI=http://localhost:11311/

setting /run_id to d8f5ca3a-6a54-11e9-953a-dc85de574b1d
process[roscout-1]: started with pid [12316]
started core service [/roscout]
```

Apriamo un secondo terminale e
lanciamo
`roslaunch hello_ros talker.py`

```
bloisi@bloisi-U36SG: ~/catkin_ws
bloisi@bloisi-U36SG:~$ cd ~/catkin_ws
bloisi@bloisi-U36SG:~/catkin_ws$ source ./devel/setup.bash
bloisi@bloisi-U36SG:~/catkin_ws$ roslaunch hello_ros talker.py
```

Cosa accade?

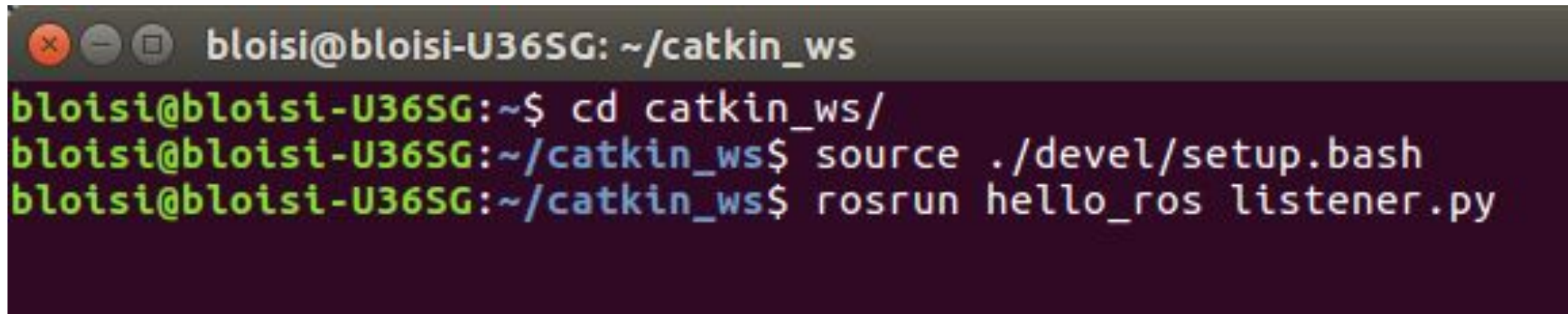
Esecuzione del nodo talker

```
bloisi@bloisi-U36SG: ~/catkin_ws
bloisi@bloisi-U36SG:~$ cd ~/catkin_ws
bloisi@bloisi-U36SG:~/catkin_ws$ source /devel/setup.bash
bloisi@bloisi-U36SG:~/catkin_ws$ rosrun hello_ros talker.py
[INFO] [1556526094.406869]: hello world 1556526094.41
[INFO] [1556526094.507411]: hello world 1556526094.51
[INFO] [1556526094.607468]: hello world 1556526094.61
[INFO] [1556526094.707442]: hello world 1556526094.71
[INFO] [1556526094.807581]: hello world 1556526094.81
[INFO] [1556526094.907577]: hello world 1556526094.91
[INFO] [1556526095.007598]: hello world 1556526095.01
[INFO] [1556526095.107500]: hello world 1556526095.11
[INFO] [1556526095.207544]: hello world 1556526095.21
[INFO] [1556526095.307254]: hello world 1556526095.31
[INFO] [1556526095.407523]: hello world 1556526095.41
[INFO] [1556526095.507509]: hello world 1556526095.51
[INFO] [1556526095.607529]: hello world 1556526095.61
[INFO] [1556526095.707478]: hello world 1556526095.71
[INFO] [1556526095.807463]: hello world 1556526095.81
[INFO] [1556526095.907509]: hello world 1556526095.91
[INFO] [1556526096.007455]: hello world 1556526096.01
[INFO] [1556526096.107619]: hello world 1556526096.11
[INFO] [1556526096.207246]: hello world 1556526096.21
```


Esecuzione del nodo listener

Apriamo un terzo terminale e
lanciamo

```
roslaunch hello_ros listener.py
```

A screenshot of a Linux terminal window. The title bar shows the user 'bloisi' on machine 'bloisi-U36SG' in the directory '~/catkin_ws'. The terminal has a dark purple background with green and white text. It shows three commands being executed in sequence: 'cd catkin_ws/', 'source ./devel/setup.bash', and 'roslaunch hello_ros listener.py'.

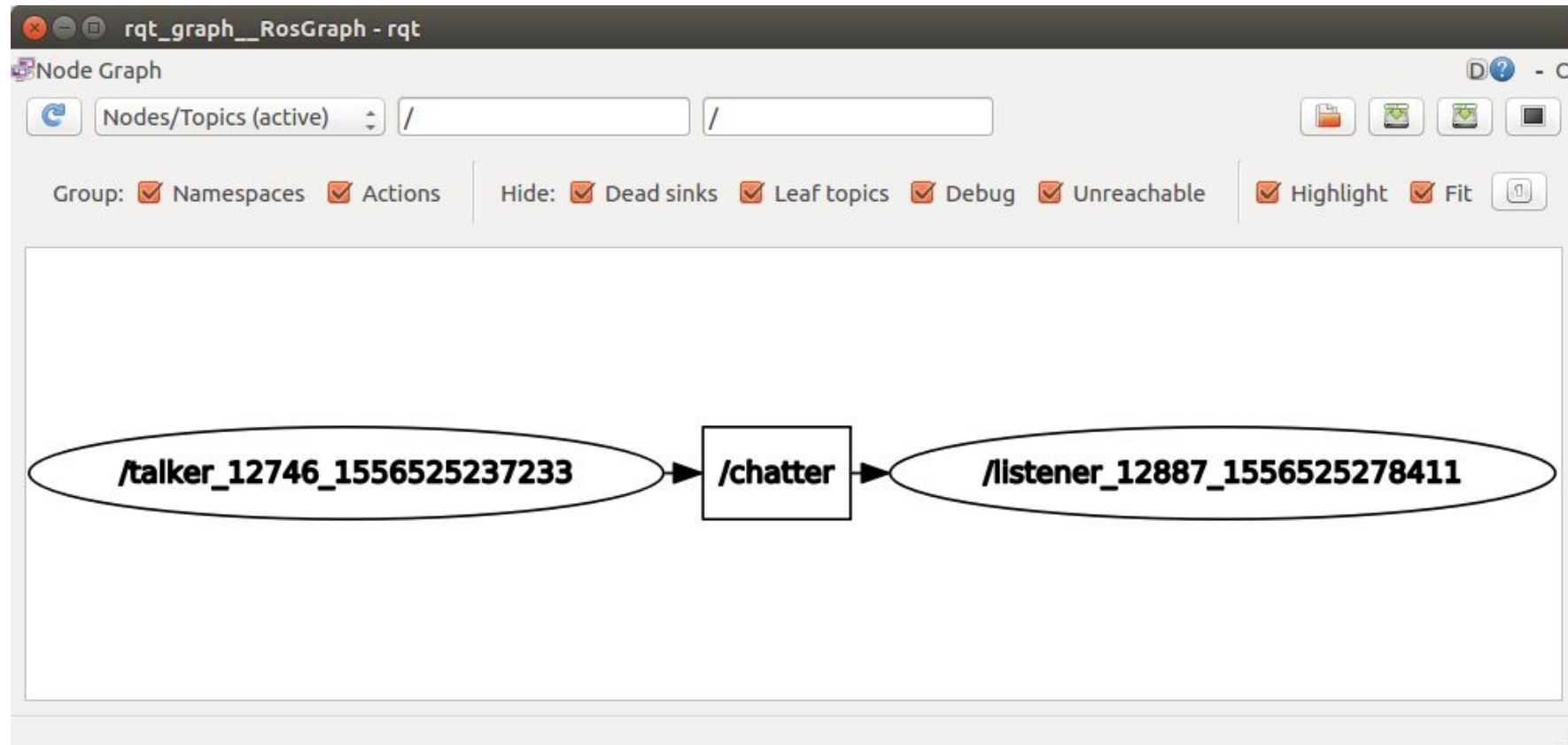
```
bloisi@bloisi-U36SG: ~/catkin_ws  
bloisi@bloisi-U36SG:~$ cd catkin_ws/  
bloisi@bloisi-U36SG:~/catkin_ws$ source ./devel/setup.bash  
bloisi@bloisi-U36SG:~/catkin_ws$ roslaunch hello_ros listener.py
```

Esecuzione del nodo listener

```
bloisi@bloisi-U36SG: ~/catkin_ws
bloisi@bloisi-U36SG:~$ cd catkin_ws/
bloisi@bloisi-U36SG:~/catkin_ws$ source ./devel/setup.bash
bloisi@bloisi-U36SG:~/catkin_ws$ rosrn hello_ros listener.py
[INFO] [1556526343.908437]: /listener_13845_1556526343630I heard hello world 1556526343.91
[INFO] [1556526344.009033]: /listener_13845_1556526343630I heard hello world 1556526344.01
[INFO] [1556526344.109351]: /listener_13845_1556526343630I heard hello world 1556526344.11
[INFO] [1556526344.209091]: /listener_13845_1556526343630I heard hello world 1556526344.21
[INFO] [1556526344.309455]: /listener_13845_1556526343630I heard hello world 1556526344.31
[INFO] [1556526344.409235]: /listener_13845_1556526343630I heard hello world 1556526344.41
[INFO] [1556526344.509644]: /listener_13845_1556526343630I heard hello world 1556526344.51
[INFO] [1556526344.609792]: /listener_13845_1556526343630I heard hello world 1556526344.61
[INFO] [1556526344.709825]: /listener_13845_1556526343630I heard hello world 1556526344.71
[INFO] [1556526344.809585]: /listener_13845_1556526343630I heard hello world 1556526344.81
[INFO] [1556526344.909382]: /listener_13845_1556526343630I heard hello world 1556526344.91
[INFO] [1556526345.009174]: /listener_13845_1556526343630I heard hello world 1556526345.01
[INFO] [1556526345.108972]: /listener_13845_1556526343630I heard hello world 1556526345.11
[INFO] [1556526345.208554]: /listener_13845_1556526343630I heard hello world 1556526345.21
[INFO] [1556526345.308504]: /listener_13845_1556526343630I heard hello world 1556526345.31
[INFO] [1556526345.408364]: /listener_13845_1556526343630I heard hello world 1556526345.41
[INFO] [1556526345.509007]: /listener_13845_1556526343630I heard hello world 1556526345.51
[INFO] [1556526345.608739]: /listener_13845_1556526343630I heard hello world 1556526345.61
[INFO] [1556526345.708979]: /listener_13845_1556526343630I heard hello world 1556526345.71
[INFO] [1556526345.809620]: /listener_13845_1556526343630I heard hello world 1556526345.81
[INFO] [1556526345.909187]: /listener_13845_1556526343630I heard hello world 1556526345.91
```

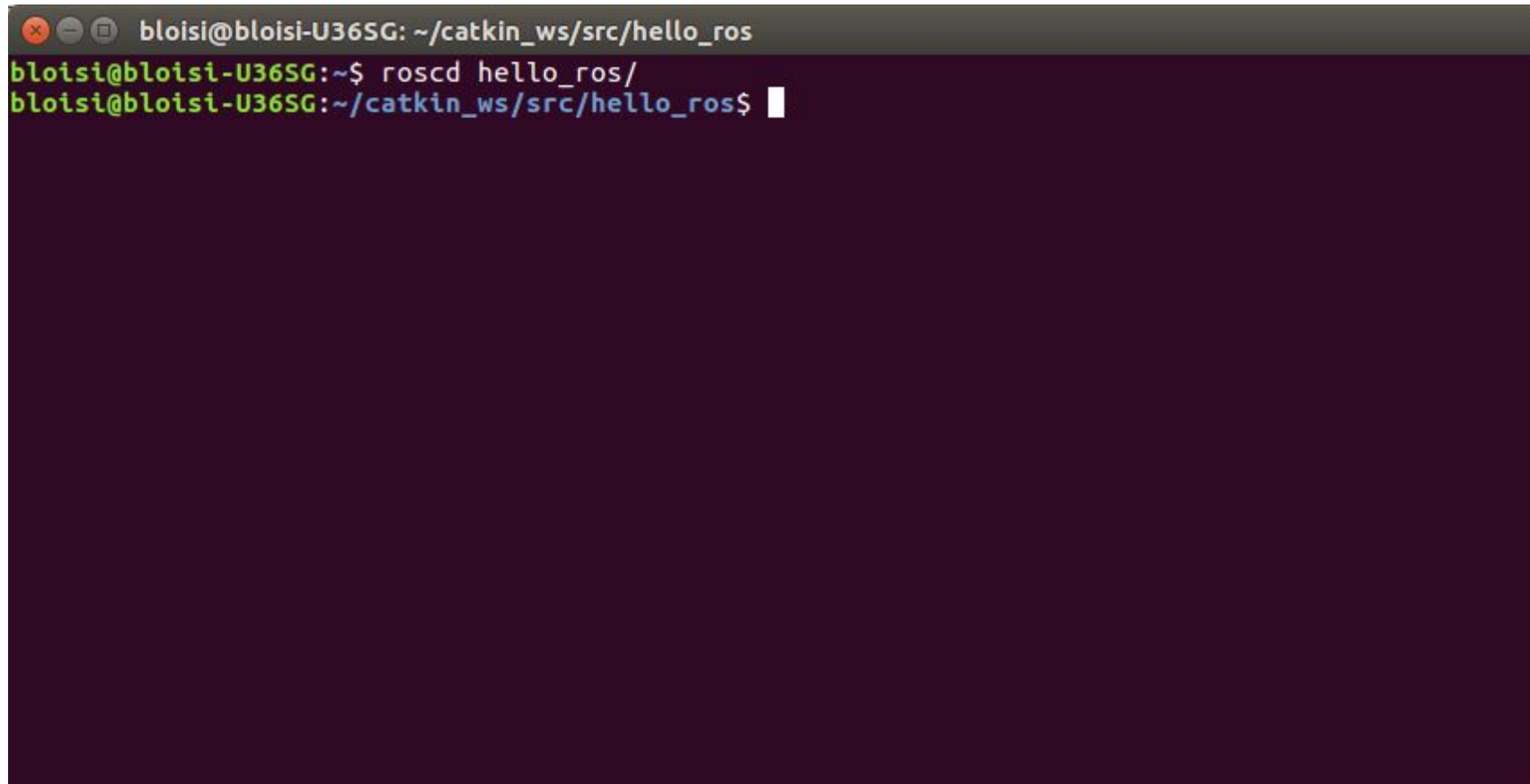

rqt_graph

```
bloisi@bloisi-U36SG: ~  
bloisi@bloisi-U36SG:~$ rqt_graph  
[redacted]
```



roscd

Con roscd possiamo navigare nel filesystem per portarci nella directory del nostro package

A terminal window with a dark purple background and a grey title bar. The title bar contains the text 'bloisi@bloisi-U36SG: ~/catkin_ws/src/hello_ros'. The terminal shows two lines of text: the first line is 'bloisi@bloisi-U36SG:~\$ roscd hello_ros/' and the second line is 'bloisi@bloisi-U36SG:~/catkin_ws/src/hello_ros\$' followed by a white cursor. The text is in a monospaced font with green for the prompt and blue for the path.

```
bloisi@bloisi-U36SG: ~/catkin_ws/src/hello_ros
bloisi@bloisi-U36SG:~$ roscd hello_ros/
bloisi@bloisi-U36SG:~/catkin_ws/src/hello_ros$
```


Aggiorniamo il repository locale

Aggiorniamo il repository locale con la cartella src

```
git add  
git commit
```

```
nvidia@tegra-ubuntu:~/catkin_ws/src/hello_ros$ git add src/  
nvidia@tegra-ubuntu:~/catkin_ws/src/hello_ros$ git commit -m 'src files'  
[master f7d5a4f] src files  
1 file changed, 93 insertions(+)  
create mode 100644 src/listener.cpp  
nvidia@tegra-ubuntu:~/catkin_ws/src/hello_ros$
```

Aggiorniamo il repository locale (package.xml)

```
git add  
git commit
```

```
nvidia@tegra-ubuntu:~/catkin_ws/src/hello_ros$ git add package.xml  
nvidia@tegra-ubuntu:~/catkin_ws/src/hello_ros$ git commit -m 'package.xml'  
[master 96ed373] package.xml  
1 file changed, 68 insertions(+)  
create mode 100644 package.xml
```

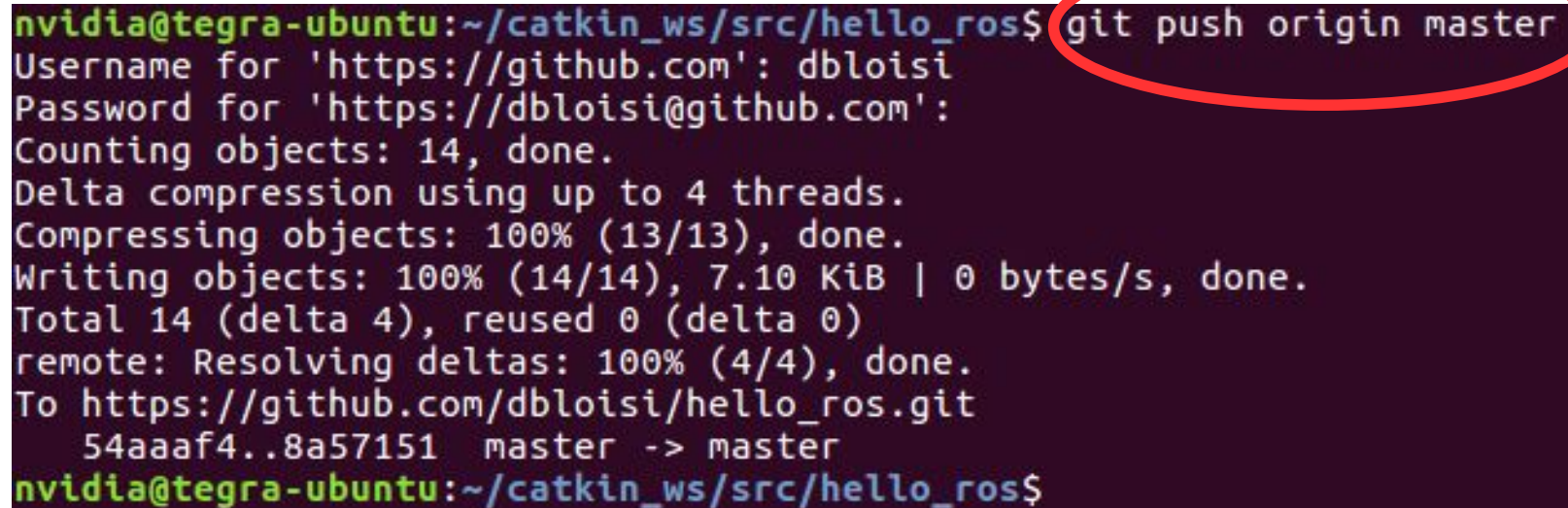
Aggiorniamo il repository locale (CMakeLists.txt)

```
git add  
git commit
```

```
nvidia@tegra-ubuntu:~/catkin_ws/src/hello_ros$ git add CMakeLists.txt  
nvidia@tegra-ubuntu:~/catkin_ws/src/hello_ros$ git commit -m 'cmake files'  
[master 8a57151] cmake files  
1 file changed, 205 insertions(+)  
create mode 100644 CMakeLists.txt
```

Aggiorniamo il repository remoto


git push





```
nvidia@tegra-ubuntu:~/catkin_ws/src/hello_ros$ git push origin master
Username for 'https://github.com': dbloisi
Password for 'https://dbloisi@github.com':
Counting objects: 14, done.
Delta compression using up to 4 threads.
Compressing objects: 100% (13/13), done.
Writing objects: 100% (14/14), 7.10 KiB | 0 bytes/s, done.
Total 14 (delta 4), reused 0 (delta 0)
remote: Resolving deltas: 100% (4/4), done.
To https://github.com/dbloisi/hello_ros.git
  54aaaf4..8a57151  master -> master
nvidia@tegra-ubuntu:~/catkin_ws/src/hello_ros$
```


Verranno richieste le credenziali di accesso (username e password) per il server git


Aggiorniamo il repository remoto


 GitHub, Inc. [US] | https://github.com/dbloisi/hello_ros

 5 commits

 1 branch

 0 releases

 1 contributor


 GPL-3.0






Branch: master ▾


New pull request

Find file

Clone or download ▾

 dbloisi cmake files Latest commit 8a57151 3 hours ago

 src	src files	3 hours ago
 CMakeLists.txt	cmake files	3 hours ago
 LICENSE	Initial commit	4 hours ago
 README.md	Initial commit	4 hours ago
 package.xml	package.xml	3 hours ago

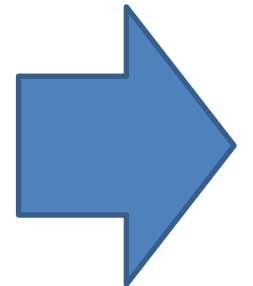
 README.md

hello_ros

my first ros package

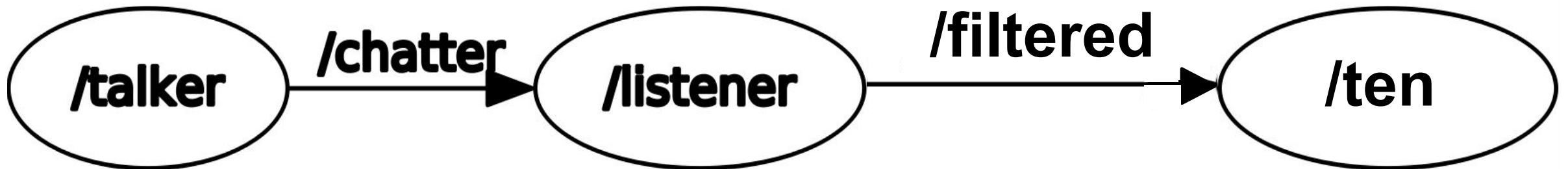
Esercitazione

1. Creare un account su un server git (es. GitHub, BitBucket, GitLab)
2. Creare un repository denominato my_hello_ros
3. Creare un package my_hello_ros contenente i nodi talker e listener
4. Caricare il codice sul proprio repository



Esercitazione

5. Modificare il codice del listener in modo che pubblichi a sua volta un messaggio dopo aver ascoltato 10 messaggi provenienti dal talker
6. Creare un nuovo nodo **ten** che ascolti i messaggi del listener e li stampi a video
7. Aggiornare il repository remoto



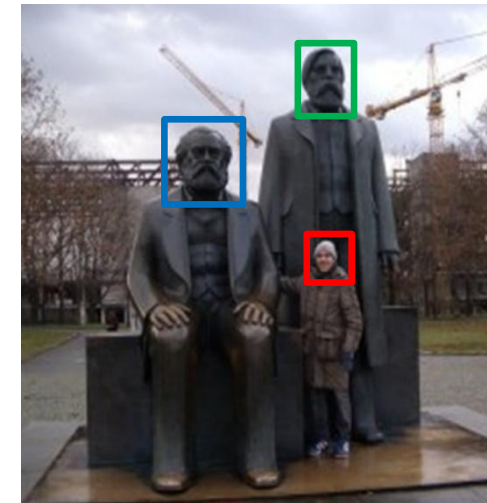
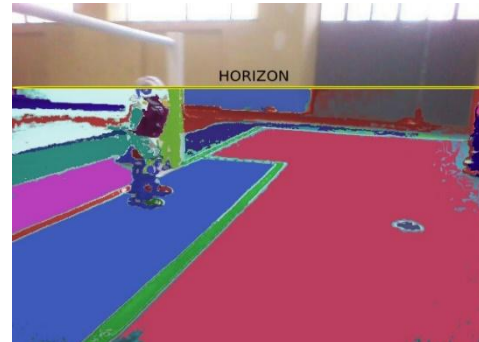
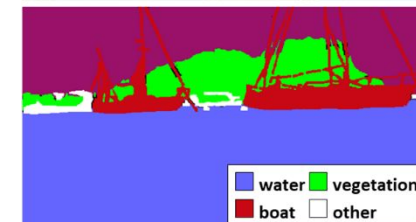


**UNIVERSITÀ DEGLI STUDI
DELLA BASILICATA**

Corso di Sistemi Informativi
A.A. 2018/19

Docente
Domenico Daniele Bloisi

git + **ROS (Python)**



Maggio 2019