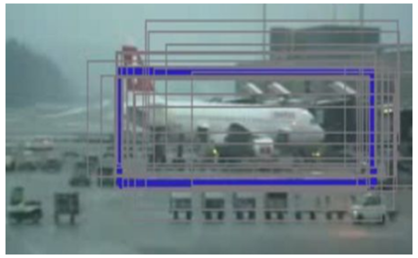
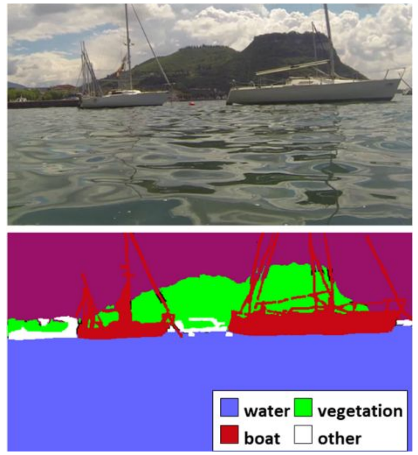
UNIVERSITÀ DEGLI STUDI DELLA BASILICATA

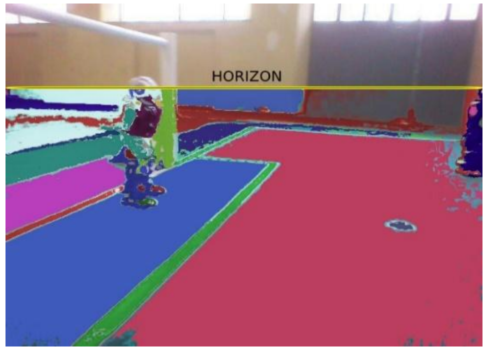
Corso di Sistemi Informativi
A.A. 2018/19

Docente
Domenico Daniele Bloisi

git + ROS (Python)

Maggio 2019

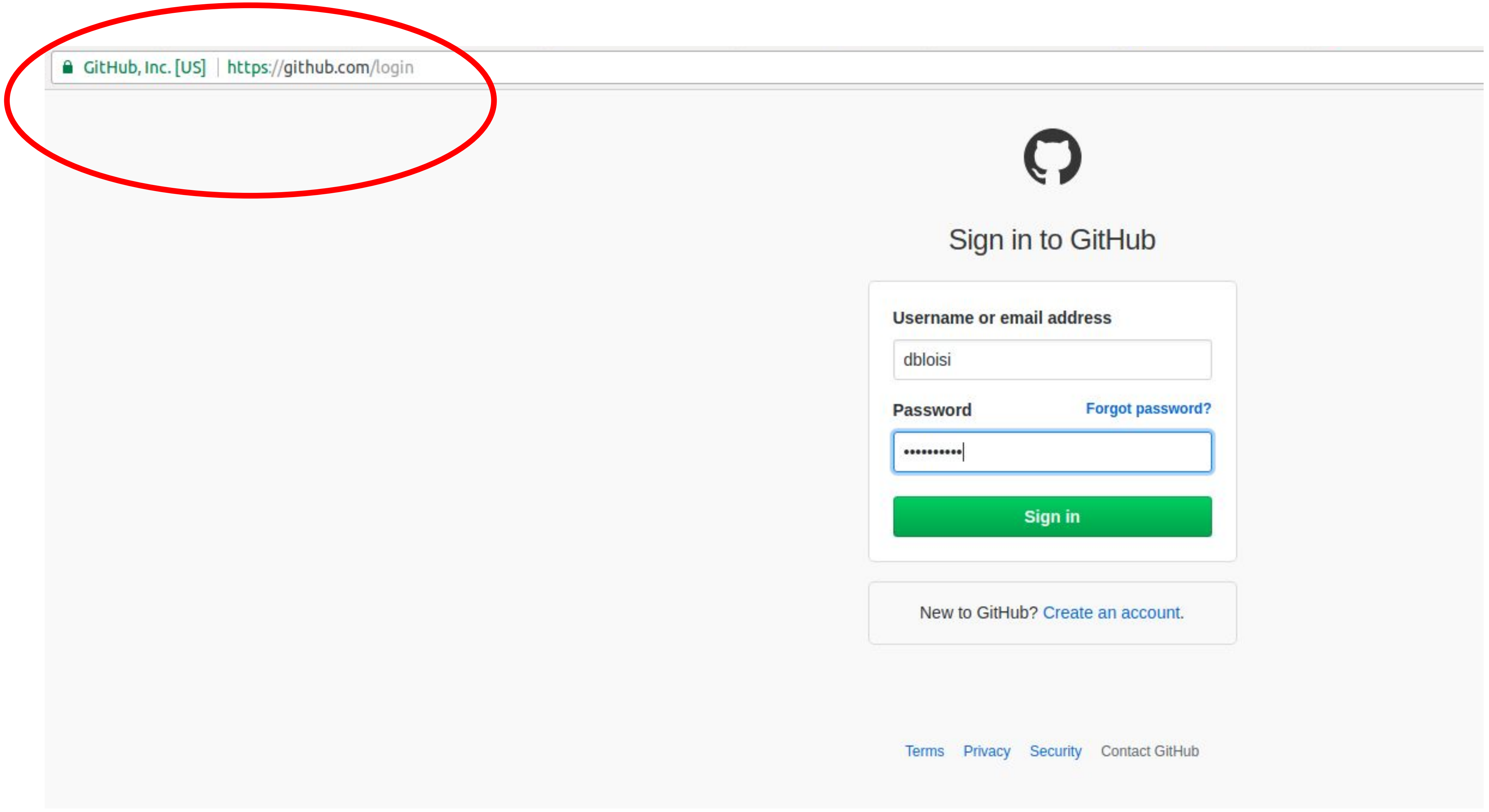


git + ROS

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



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

Sign in to GitHub

Username or email address

dbloisi

Password

Forgot password?

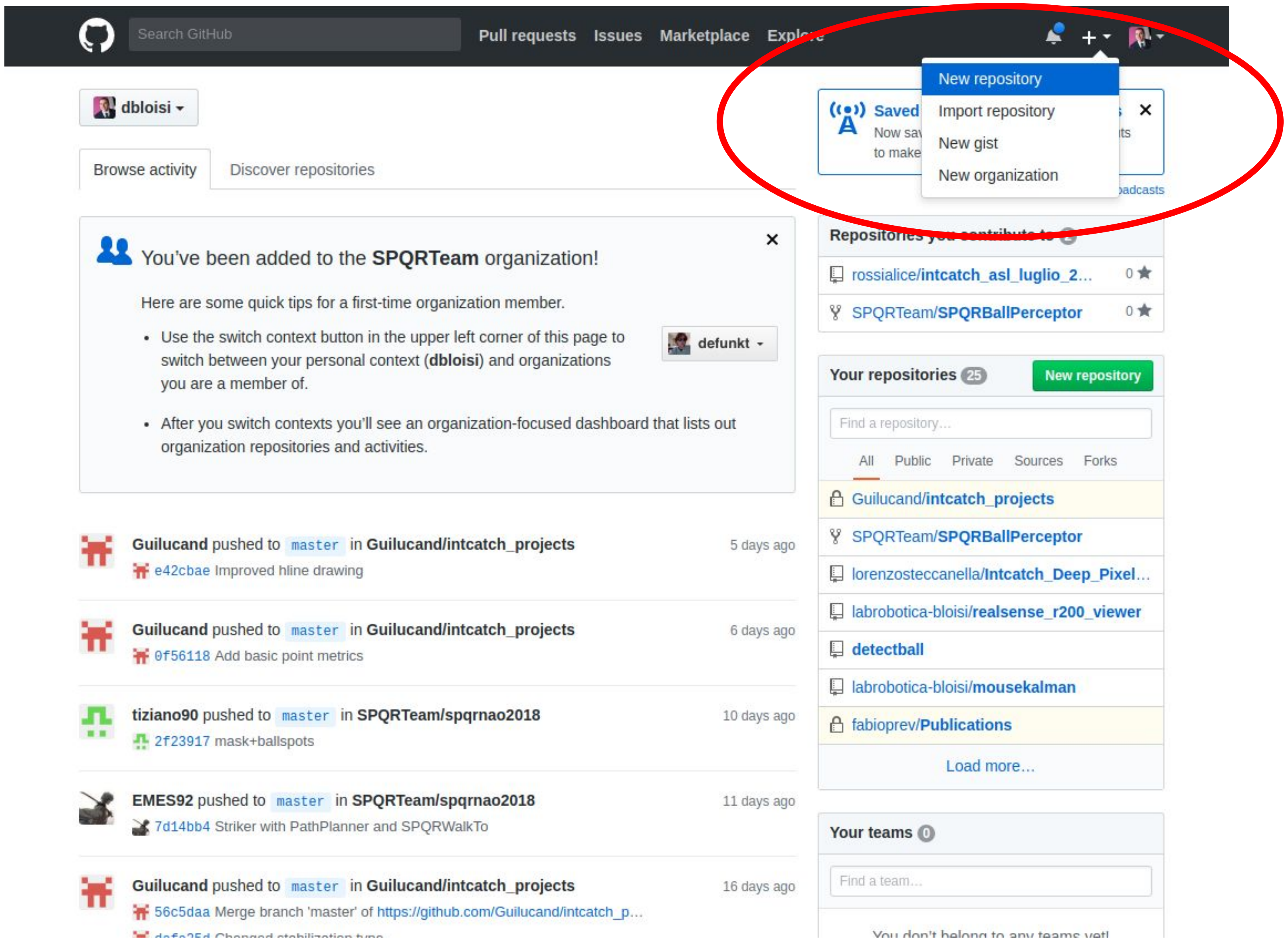
••••••••••

Sign in

New to GitHub? Create an account.

Terms Privacy Security Contact GitHub

Creare un repository git



You've been added to the **SPQRTeam** organization!

Here are some quick tips for a first-time organization member.

- Use the switch context button in the upper left corner of this page to switch between your personal context (**dbloisi**) and organizations you are a member of.
- After you switch contexts you'll see an organization-focused dashboard that lists out organization repositories and activities.

Guilucand pushed to `master` in **Guilucand/intcatch_projects** 5 days ago
`e42cbae` Improved hline drawing

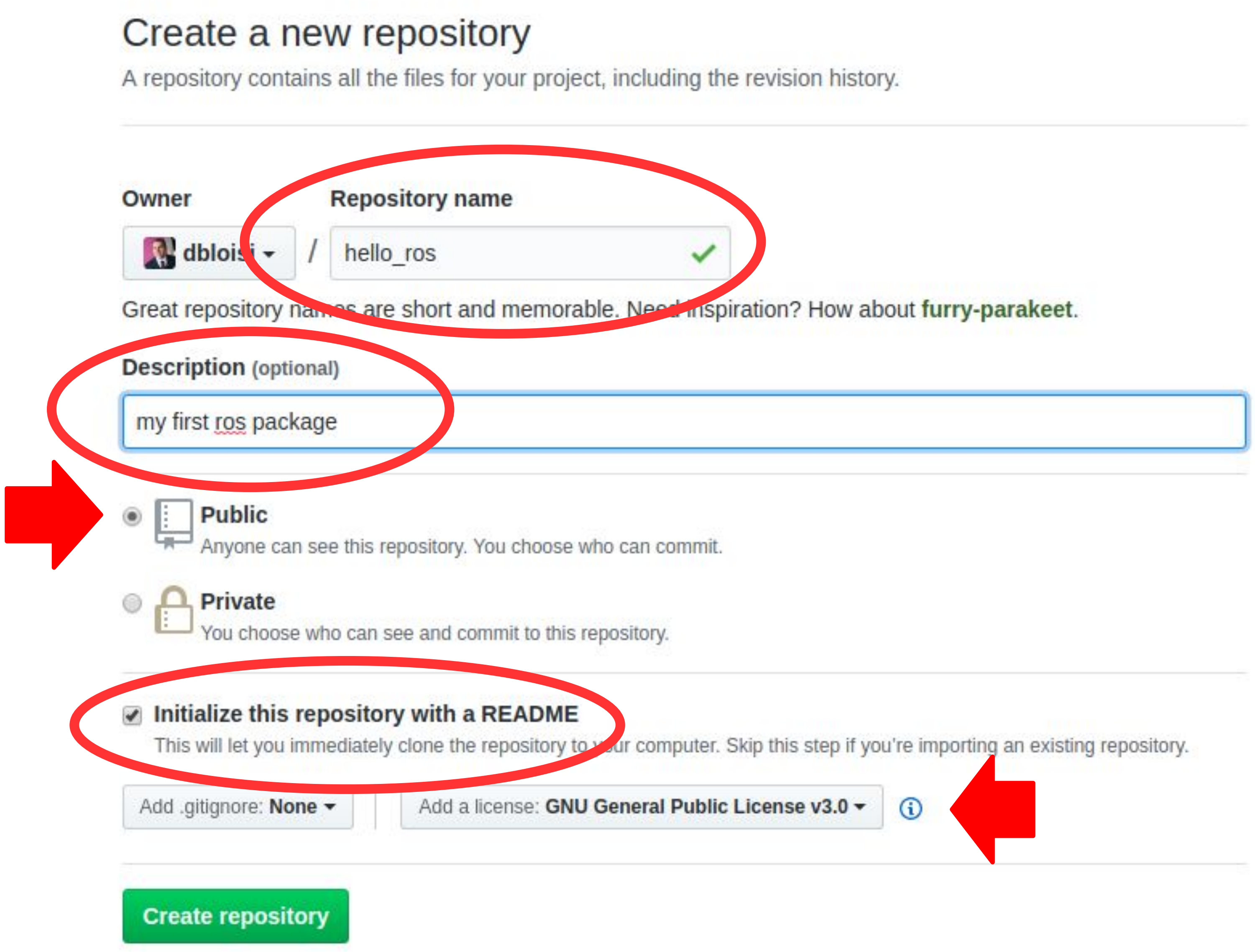
Guilucand pushed to `master` in **Guilucand/intcatch_projects** 6 days ago
`0f56118` Add basic point metrics

tiziano90 pushed to `master` in **SPQRTeam/spqrnao2018** 10 days ago
`2f23917` mask+ballspots

EMES92 pushed to `master` in **SPQRTeam/spqrnao2018** 11 days ago
`7d14bb4` Striker with PathPlanner and SPQRWalkTo

Guilucand pushed to `master` in **Guilucand/intcatch_projects** 16 days ago
`56c5daa` Merge branch 'master' of https://github.com/Guilucand/intcatch_p…

Repository name



Create a new repository

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

Owner: dbloisi / Repository name: hello_ros ✓

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

Description (optional)

my first ros package

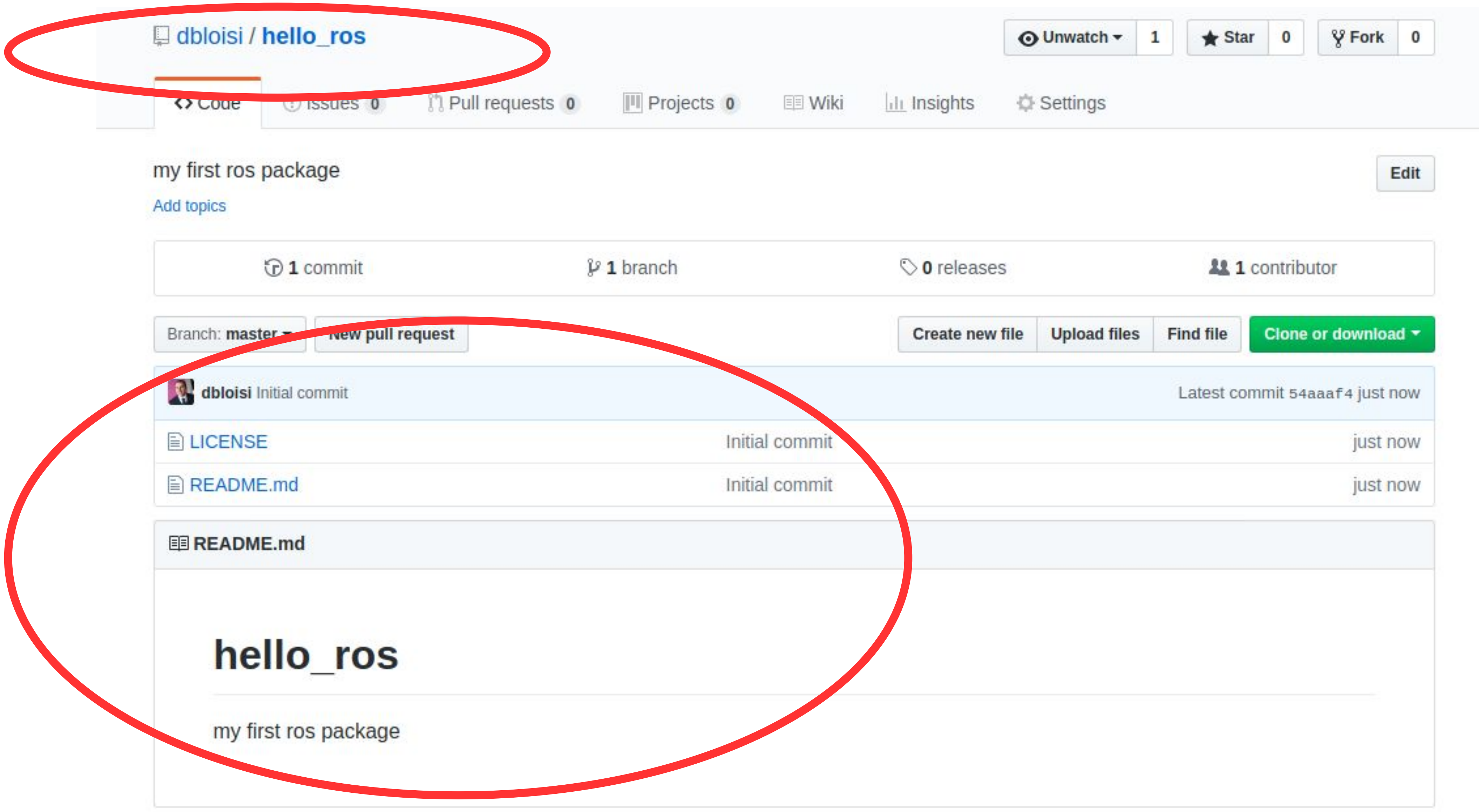
● **Public**
Anyone can see this repository. You choose who can commit.

○ **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**

Repository creato



dbloisi / hello_ros

my first ros package

1 commit | 1 branch | 0 releases | 1 contributor

dbloisi Initial commit — Latest commit 54aaaf4 just now

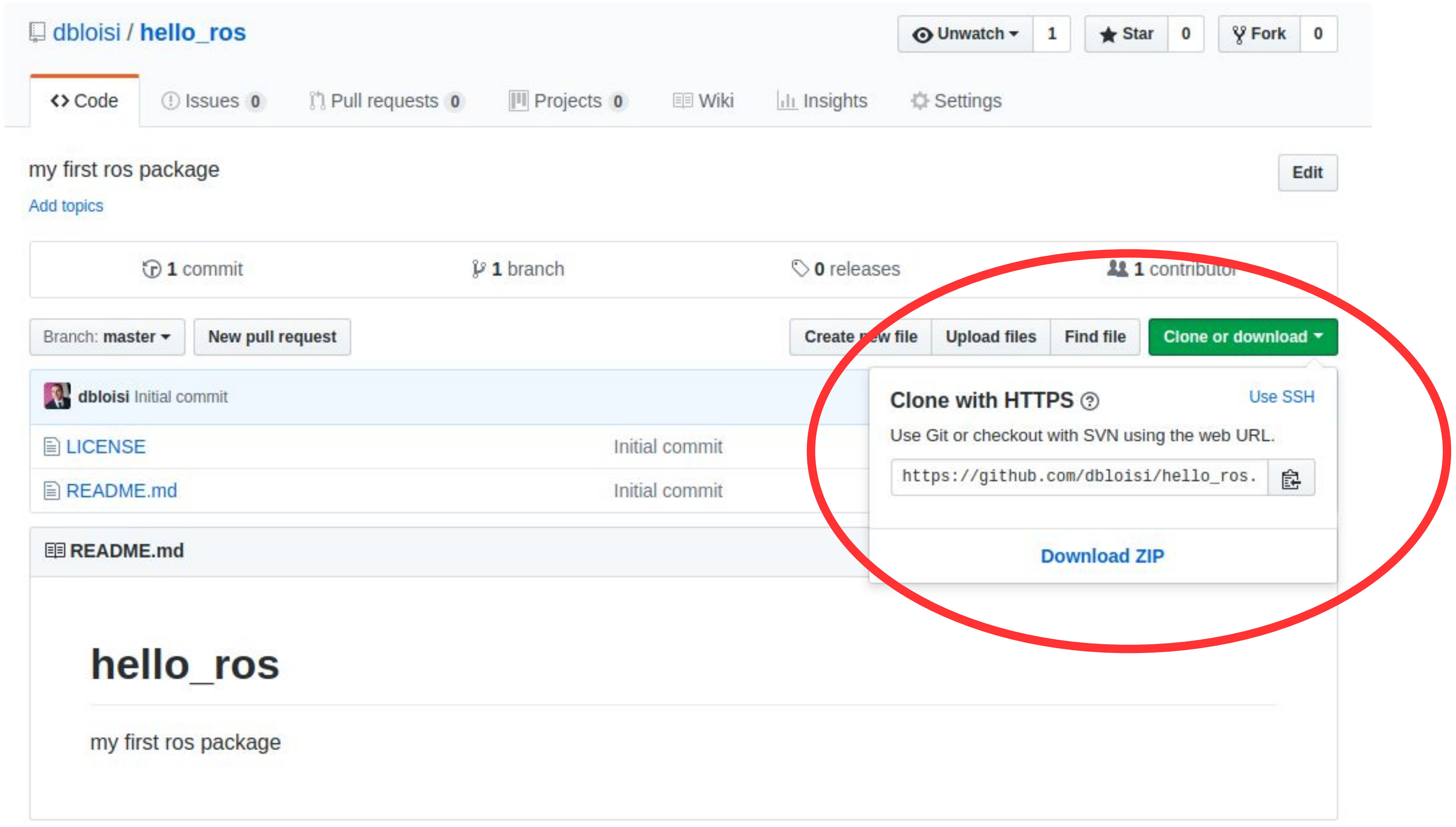
LICENSE	Initial commit	just now
README.md	Initial commit	just now

README.md

hello_ros

my first ros package

Indirizzo del repository remoto



my first ros package

dbloisi Initial commit

LICENSE	Initial commit
README.md	Initial commit

Clone with HTTPS

Use Git or checkout with SVN using the web URL.

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

README.md

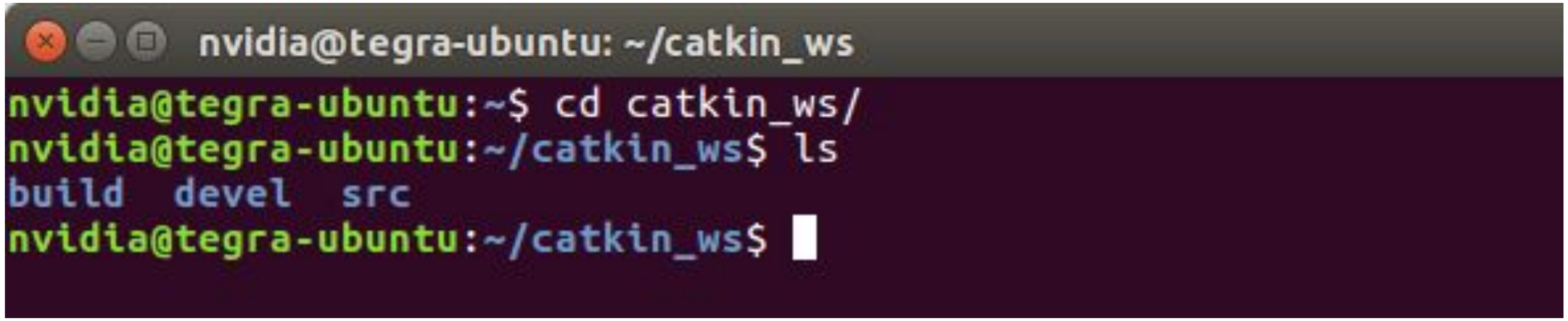
hello_ros

my first ros package

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

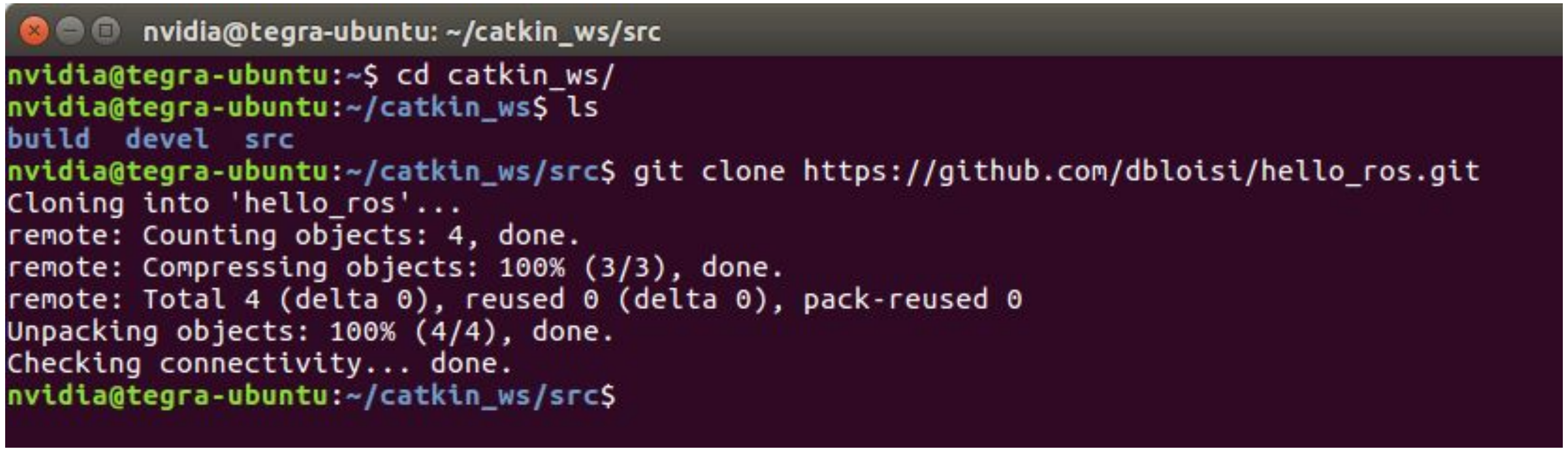


```
nvidia@tegra-ubuntu:~$ cd catkin_ws/
nvidia@tegra-ubuntu:~/catkin_ws$ ls
build  devel  src
nvidia@tegra-ubuntu:~/catkin_ws$ 
```

Creazione del repository locale

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

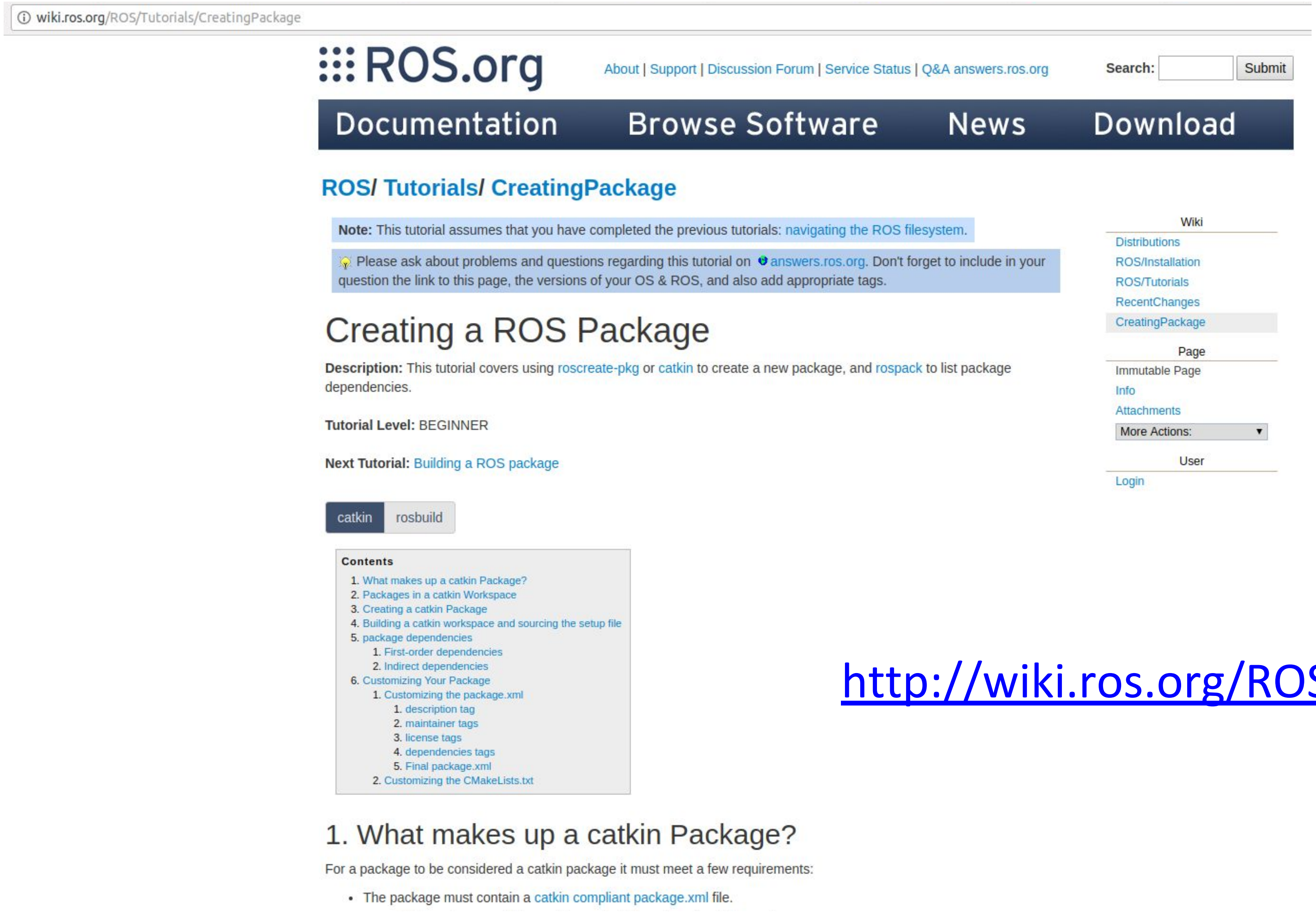
Il repository locale sarà creato in ~/catkin_ws/src/hello_ros



nvidia@tegra-ubuntu: ~/catkin_ws/src

```
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: Submit

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 roscreate-pkg or catkin to create a new package, and rospack to list package dependencies.

Tutorial Level: BEGINNER

Next Tutorial: Building a ROS package

catkin rosbuild

Contents

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.

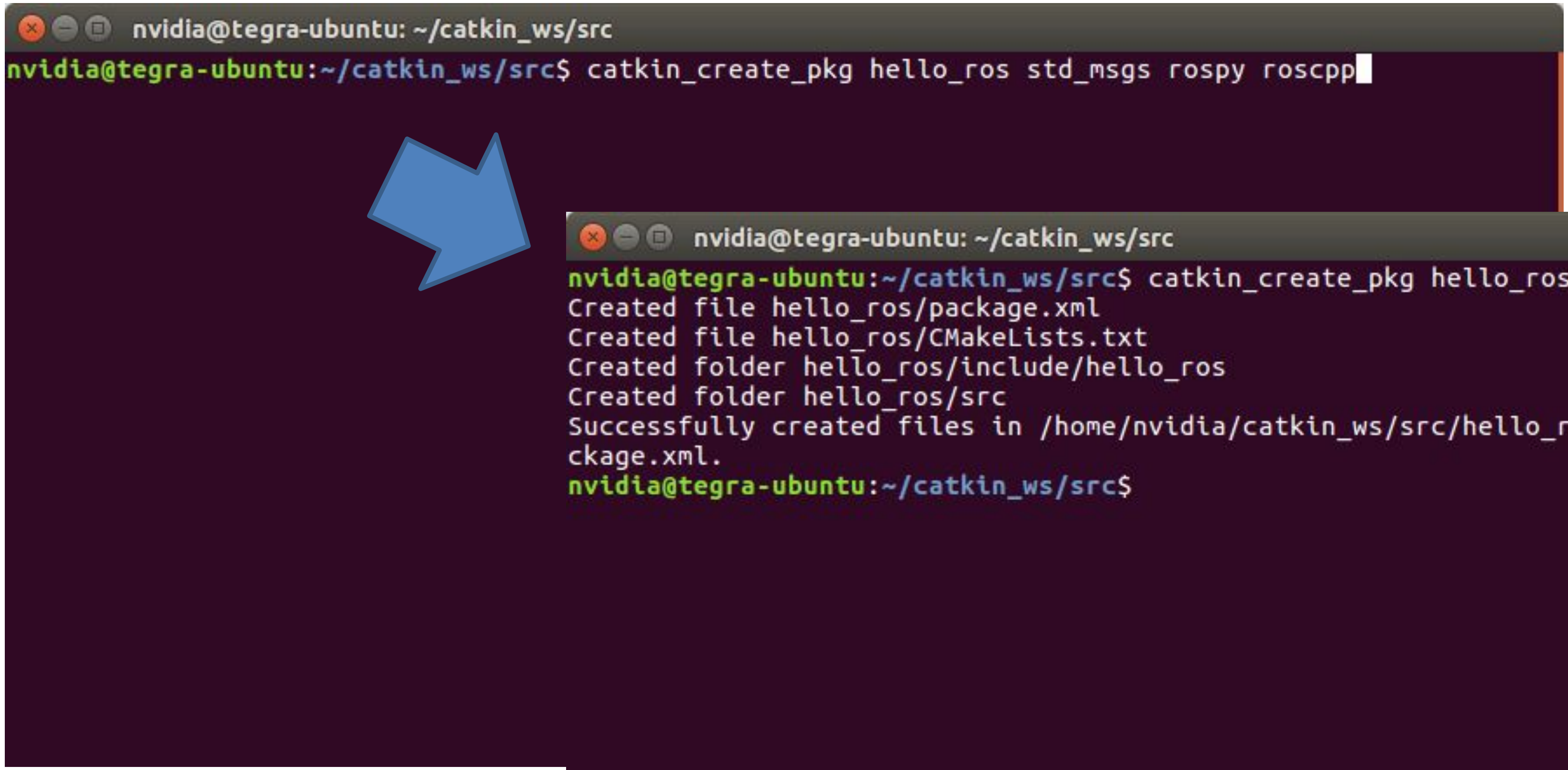
Wiki
- Distributions
- ROS/Installation
- ROS/Tutorials
- RecentChanges
- CreatingPackage

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

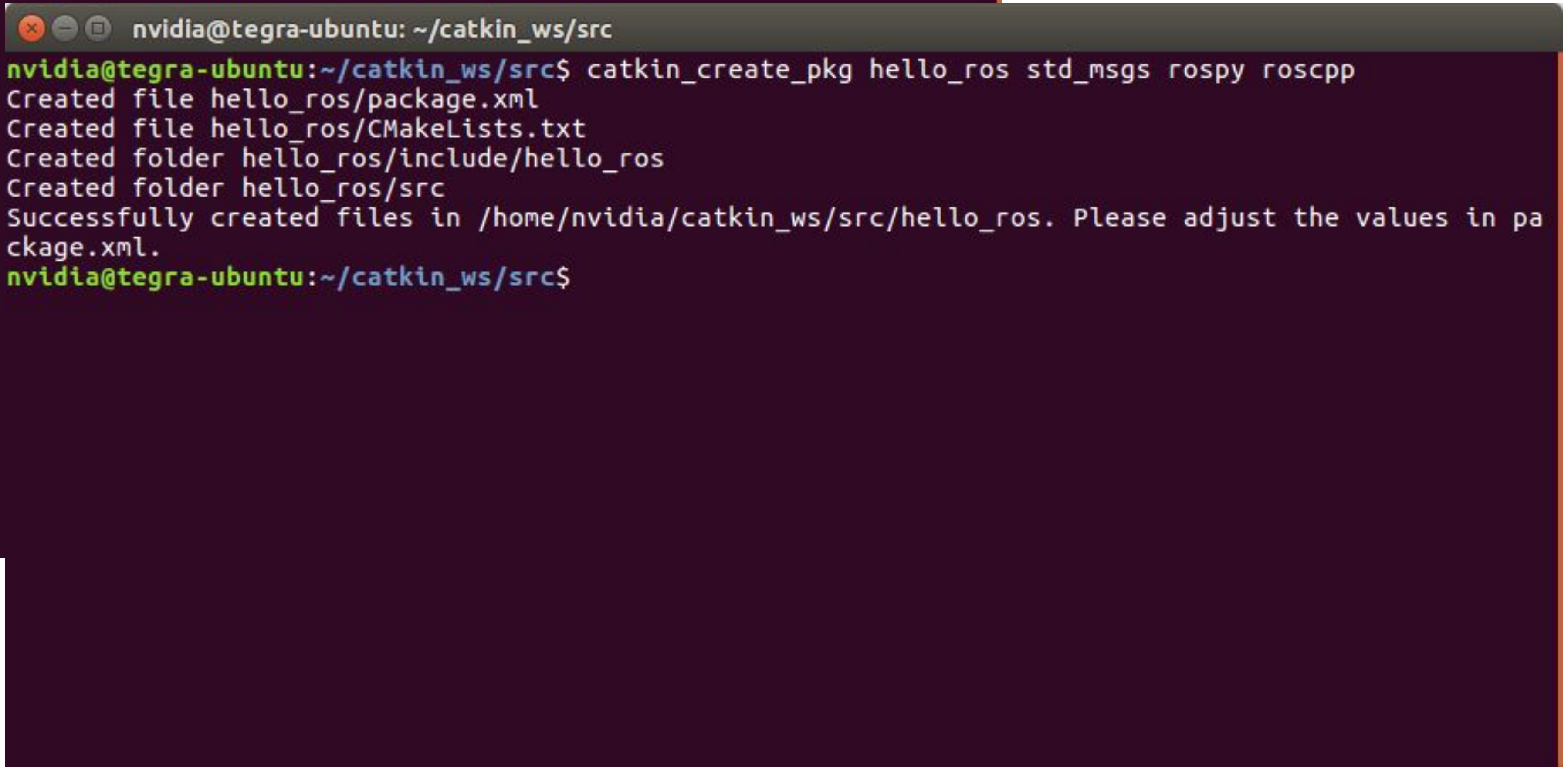
User
- Login

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

catkin_create_pkg

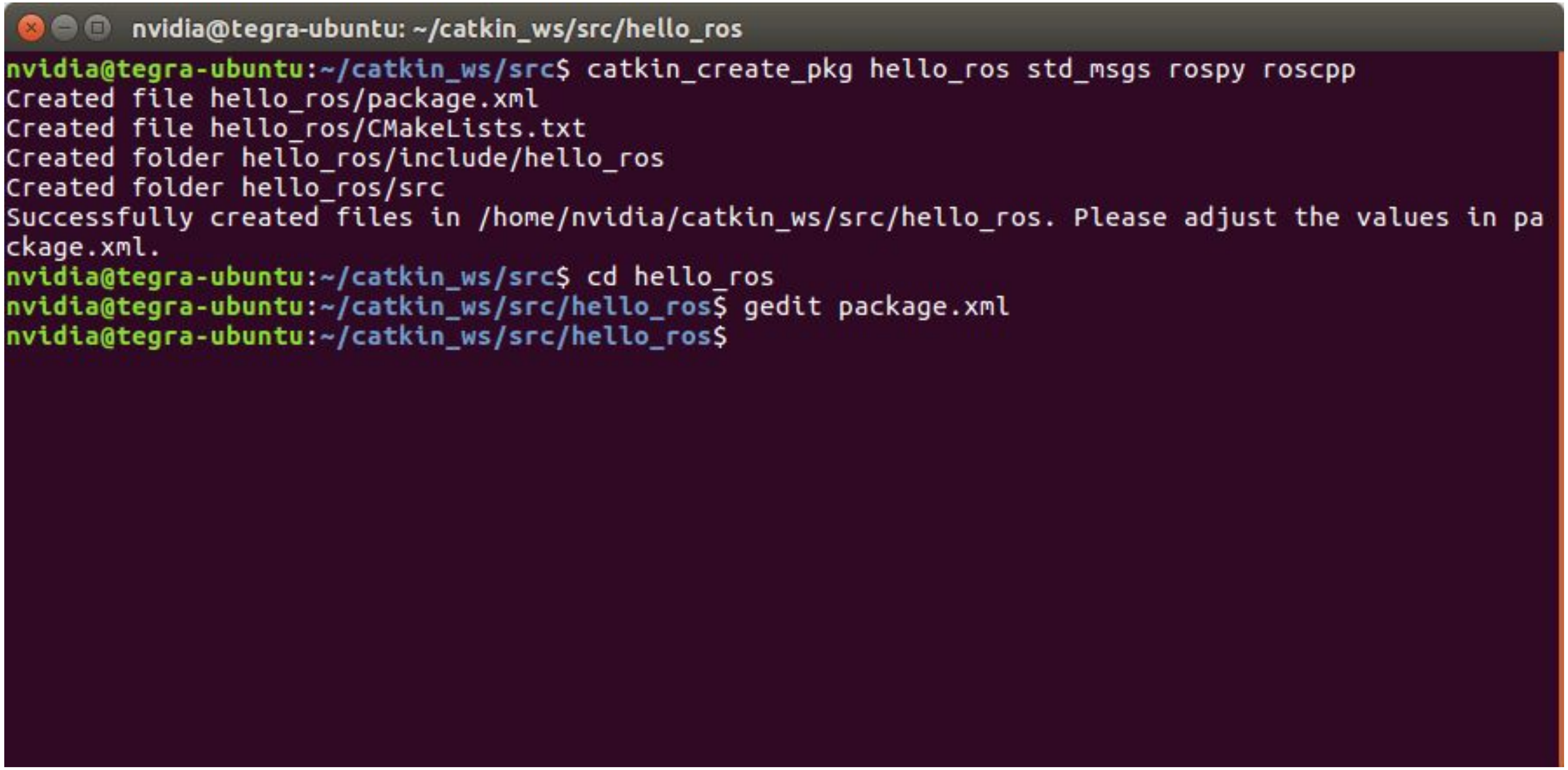


```
nvidia@tegra-ubuntu:~/catkin_ws/src$ catkin_create_pkg hello_ros std_msgs rospy roscpp
```



```
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 pa
ckage.xml.
nvidia@tegra-ubuntu:~/catkin_ws/src$
```

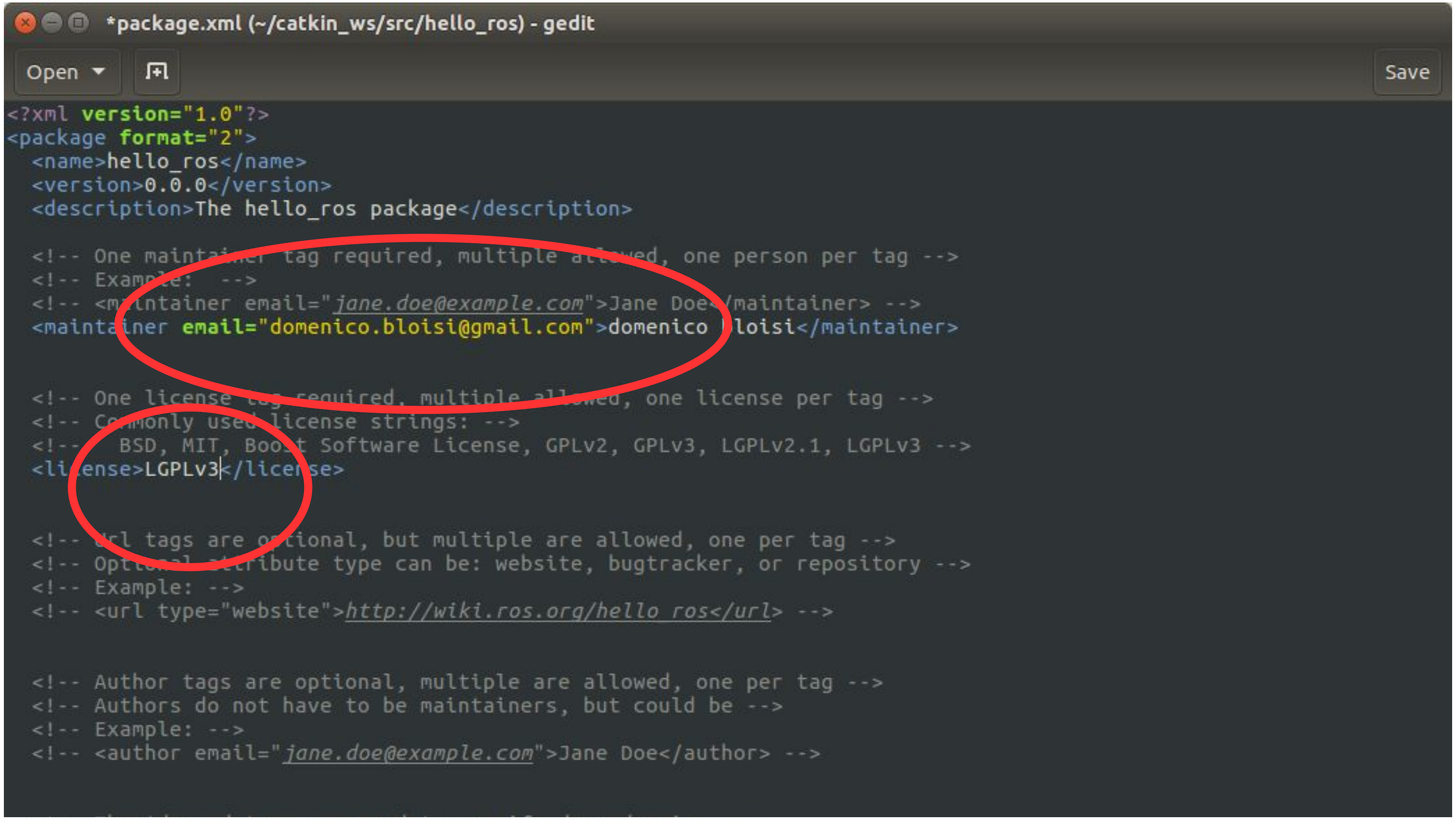
package.xml



nvidia@tegra-ubuntu: ~/catkin_ws/src/hello_ros

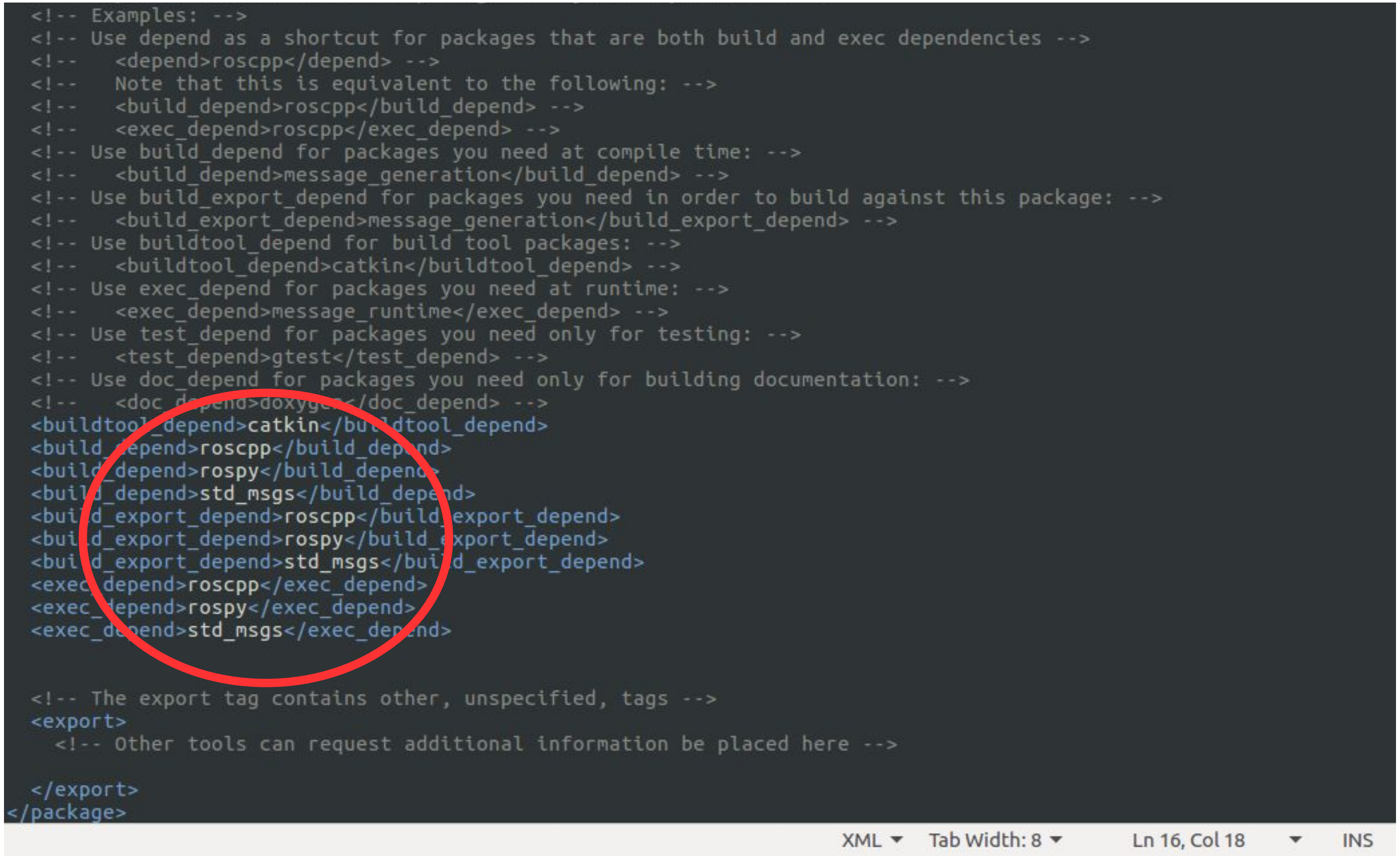
```
nvidia@tegra-ubuntu:~/catkin_ws/src$ catkin_create_pkg hello_ros std_msgs rospy roscpp
Created file hello_ros/package.xml
Created file hello_ros/CMakeLists.txt
Created folder hello_ros/include/hello_ros
Created folder hello_ros/src
Successfully created files in /home/nvidia/catkin_ws/src/hello_ros.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



```
<?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

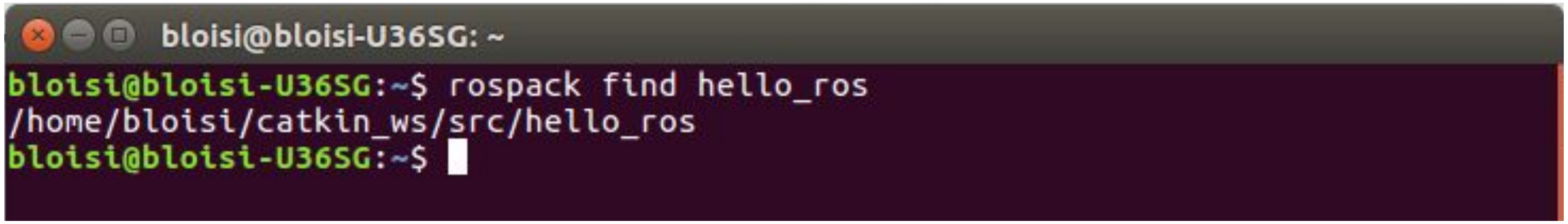


```
  <!-- Examples: -->
  <!-- Use depend as a shortcut for packages that are both build and exec dependencies -->
  <!--   <depend>roscpp</depend> -->
  <!--   Note that this is equivalent to the following: -->
  <!--   <build_depend>roscpp</build_depend> -->
  <!--   <exec_depend>roscpp</exec_depend> -->
  <!-- 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
```

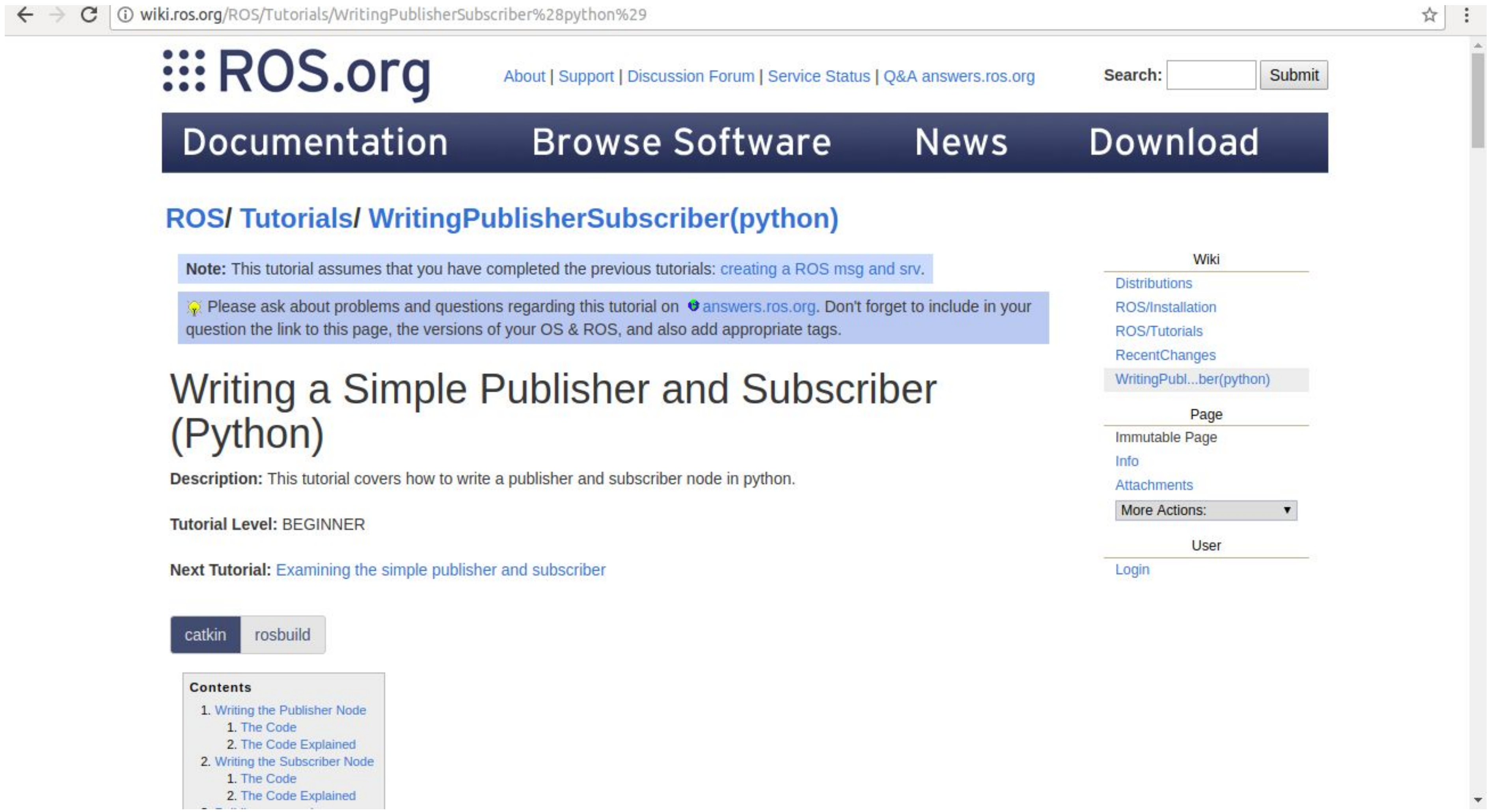


```
bloisi@bloisi-U36SG:~$ rospack find hello_ros
/home/bloisi/catkin_ws/src/hello_ros
bloisi@bloisi-U36SG:~$
```

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



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 rosbuild

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

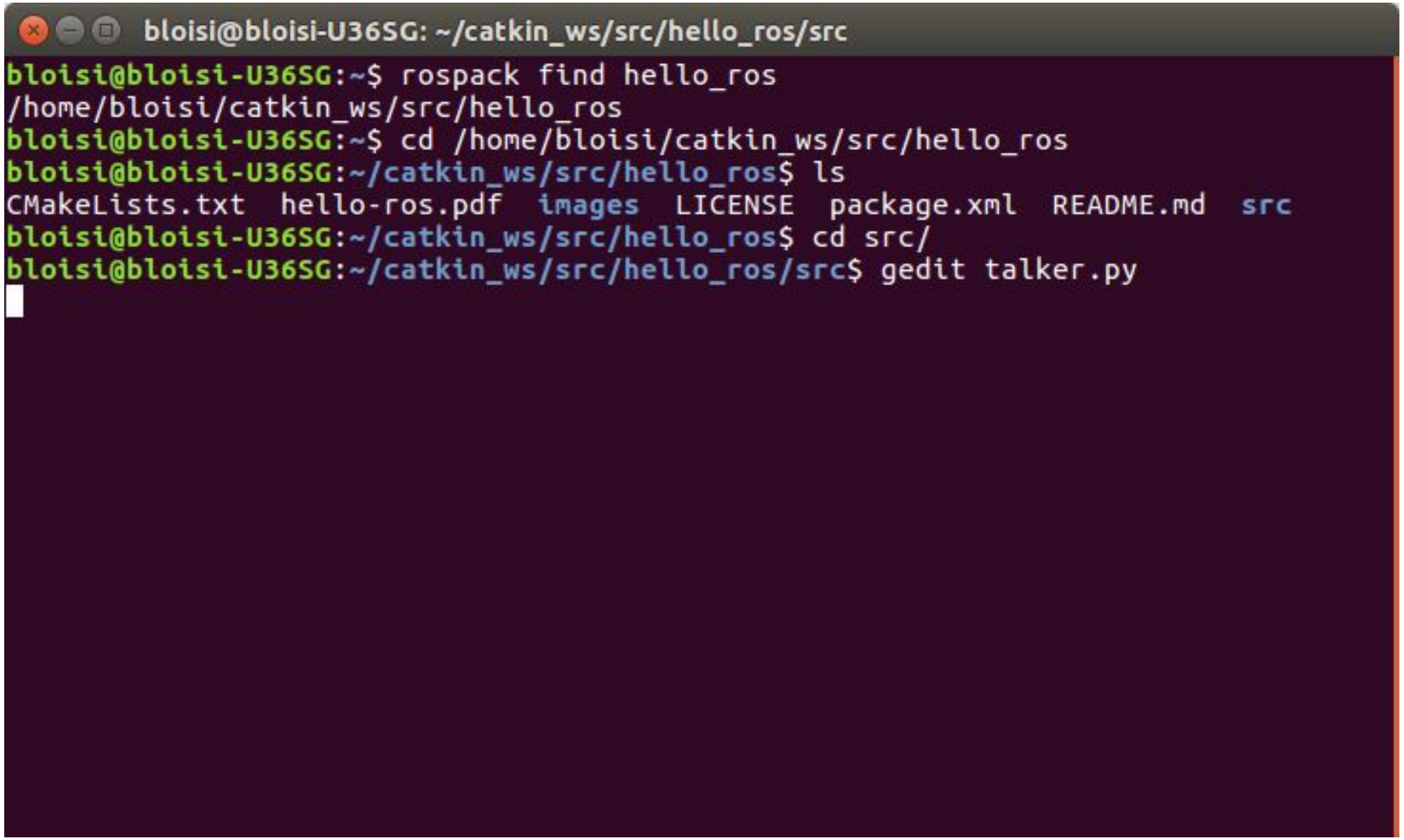
Immutable Page
Info
Attachments
More Actions:

User

Login

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

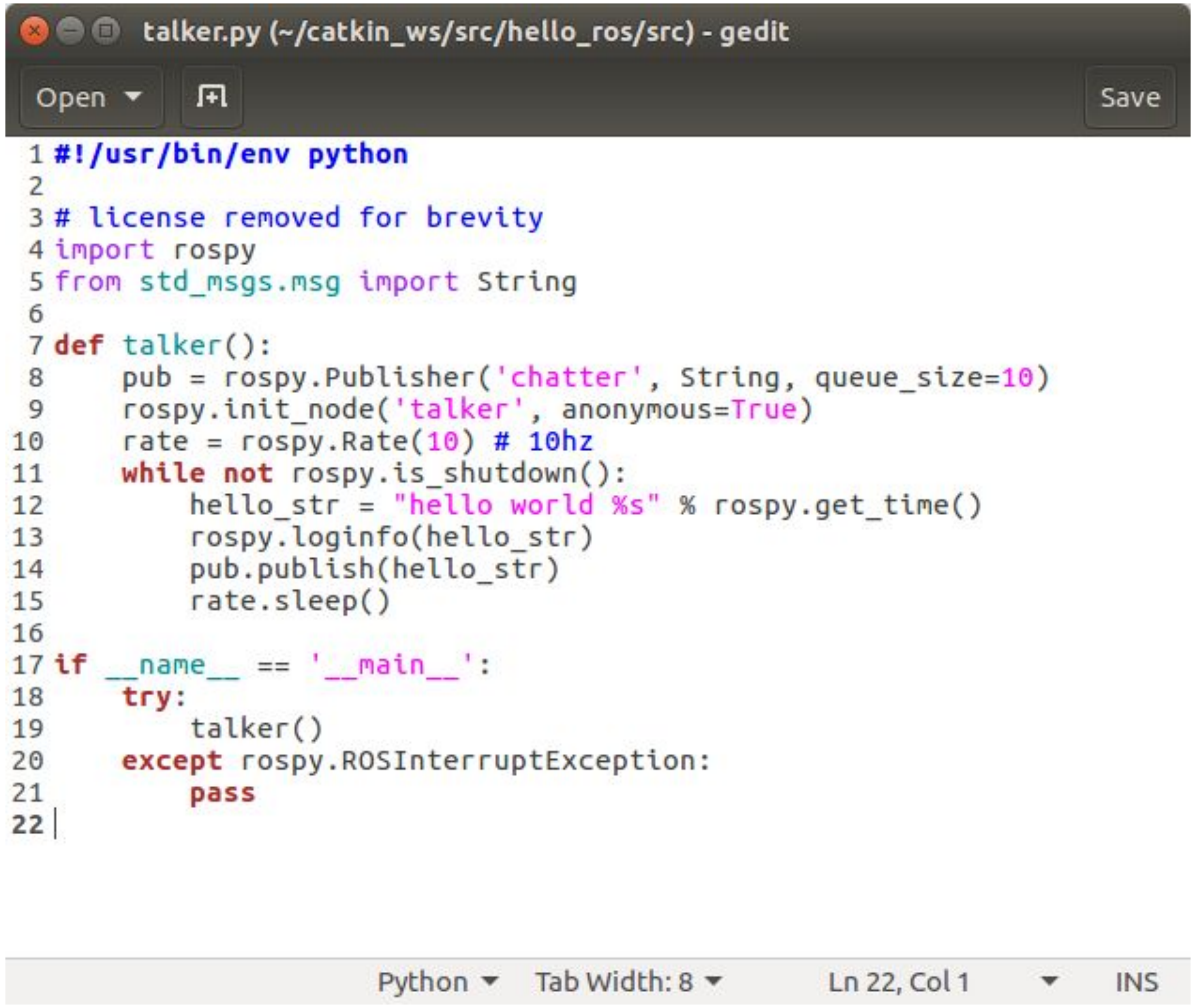
Creiamo il publisher (talker.py)



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)

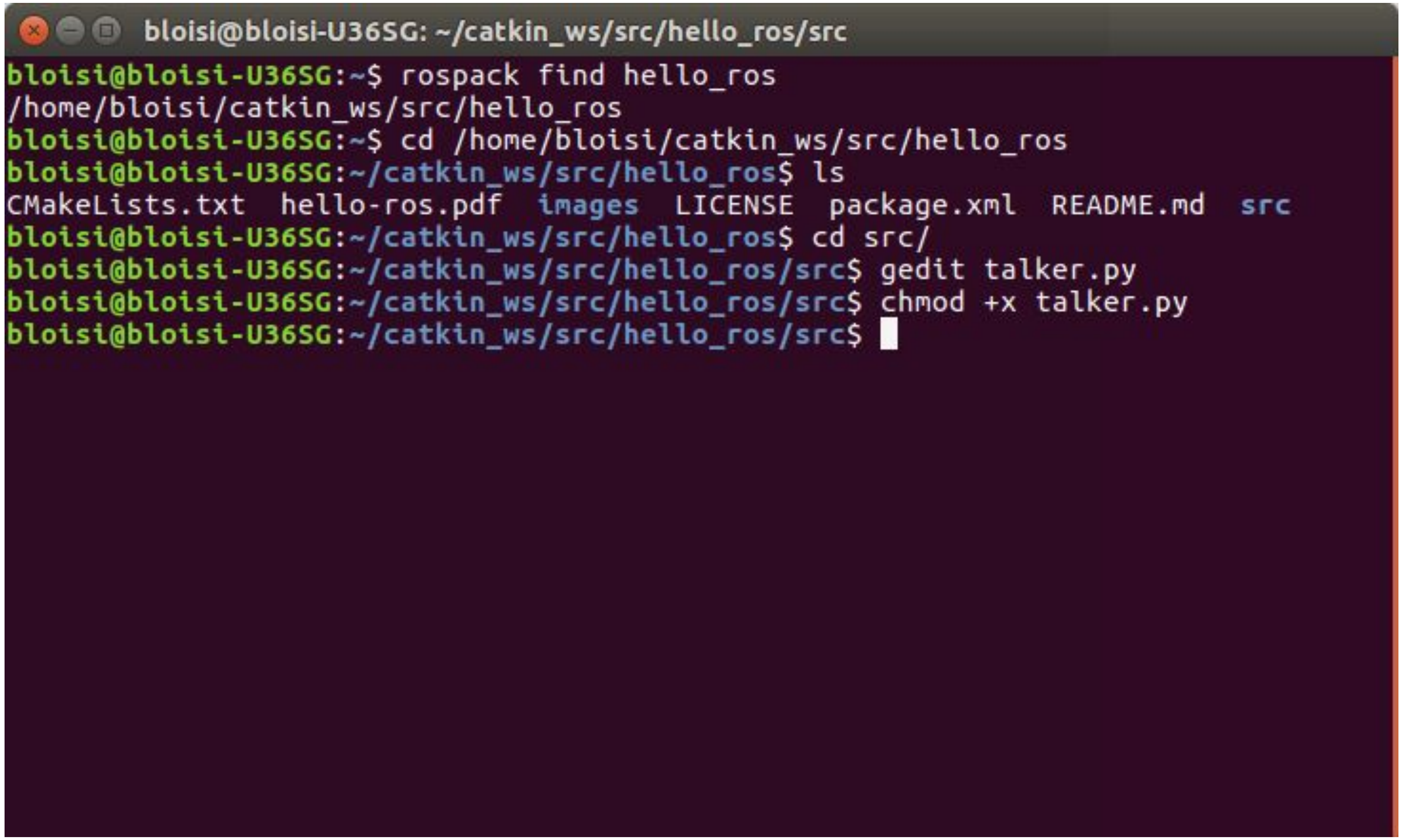


talker.py (~/catkin_ws/src/hello_ros/src) - gedit

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

# license removed for brevity
import rospy
from std_msgs.msg import String

def talker():
    pub = rospy.Publisher('chatter', String, queue_size=10)
    rospy.init_node('talker', anonymous=True)
    rate = rospy.Rate(10) # 10hz
    while not rospy.is_shutdown():
        hello_str = "hello world %s" % rospy.get_time()
        rospy.loginfo(hello_str)
        pub.publish(hello_str)
        rate.sleep()

if __name__ == '__main__':
    try:
        talker()
    except rospy.ROSInterruptException:
        pass

```

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

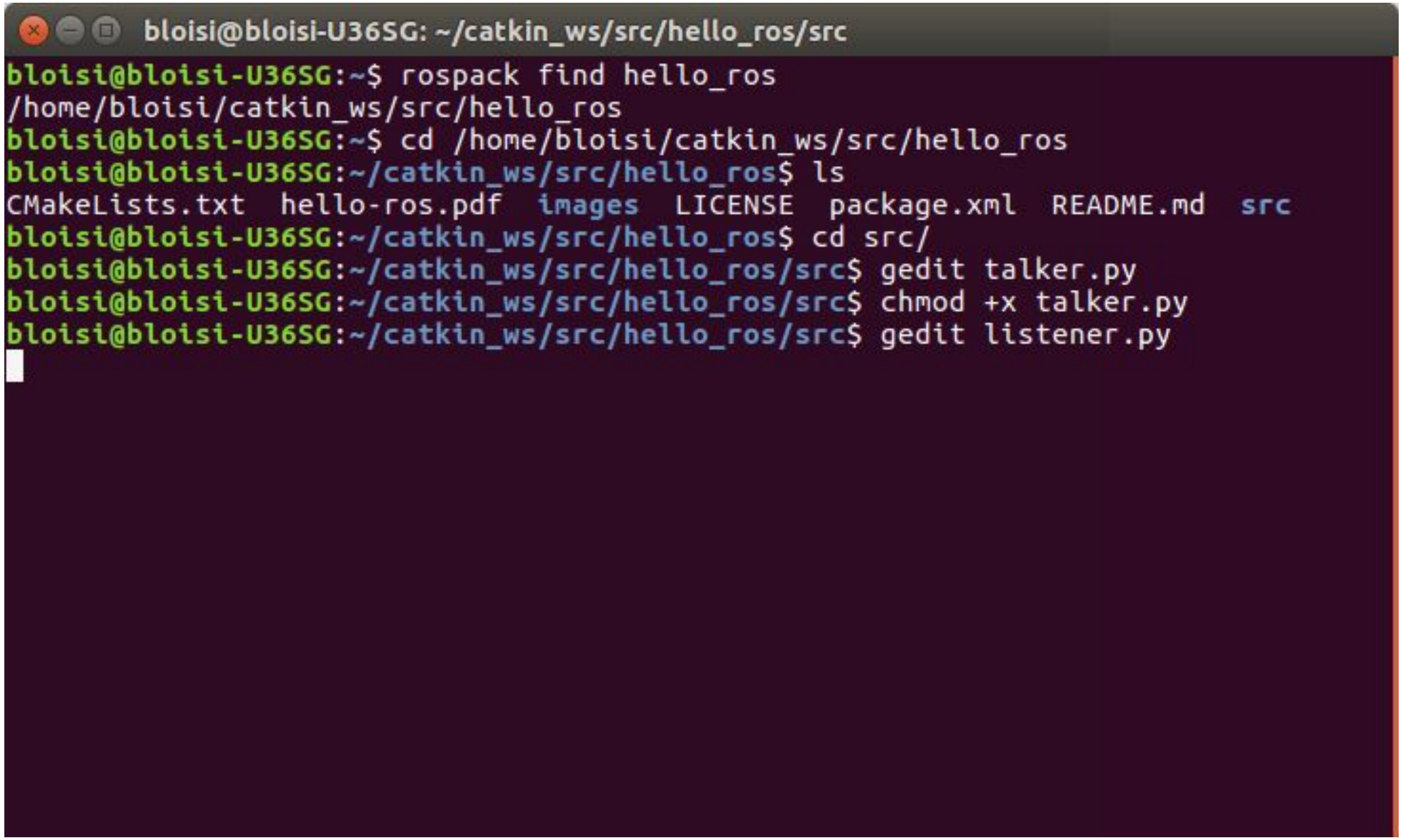
publisher (talker.py) eseguibile



```
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.github.com/ros/ros_tutorials/kinetic-devel/rospy_tutorials/001_talker_listener/talker.py

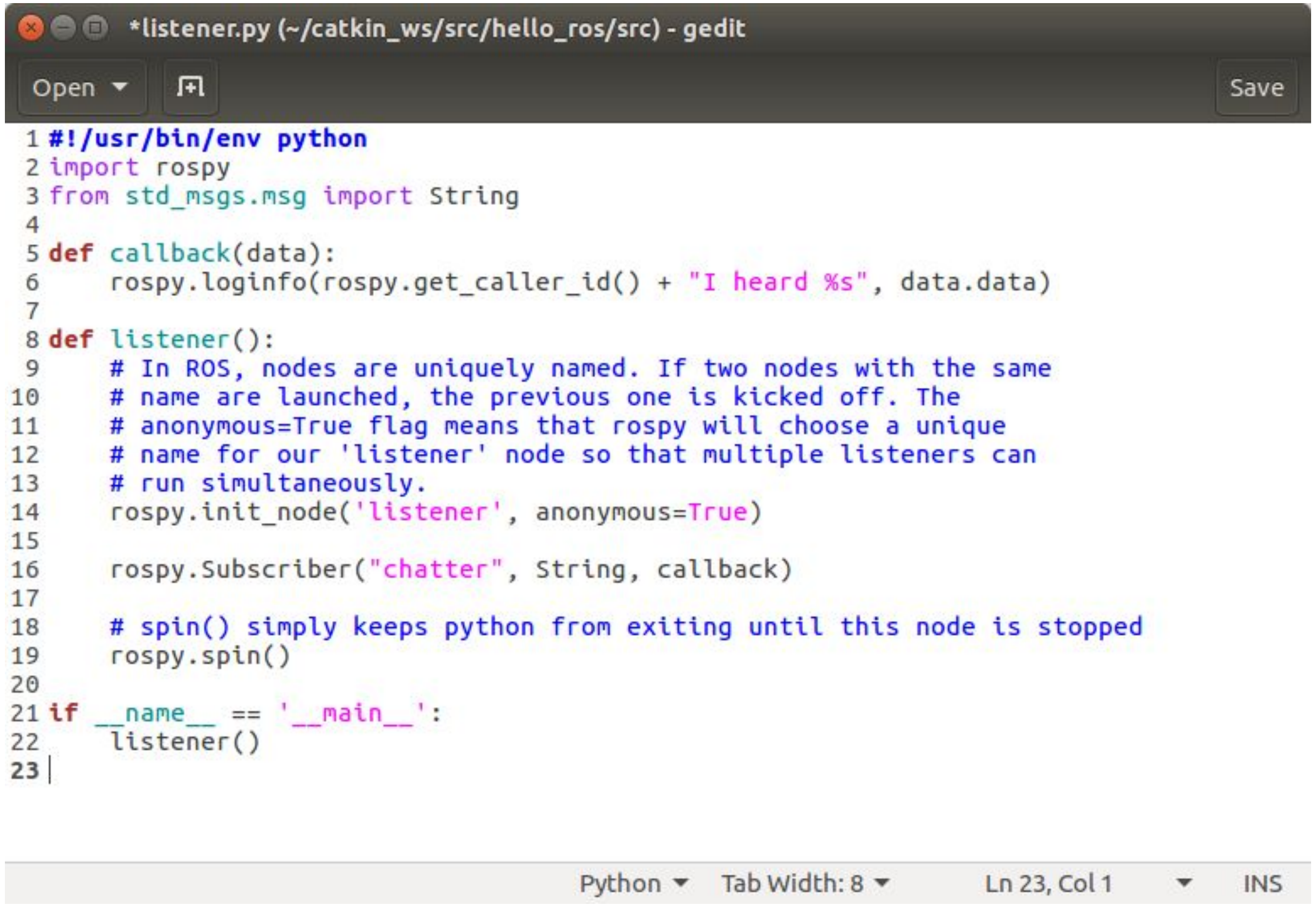
Creiamo il subscriber (listener.py)



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
```

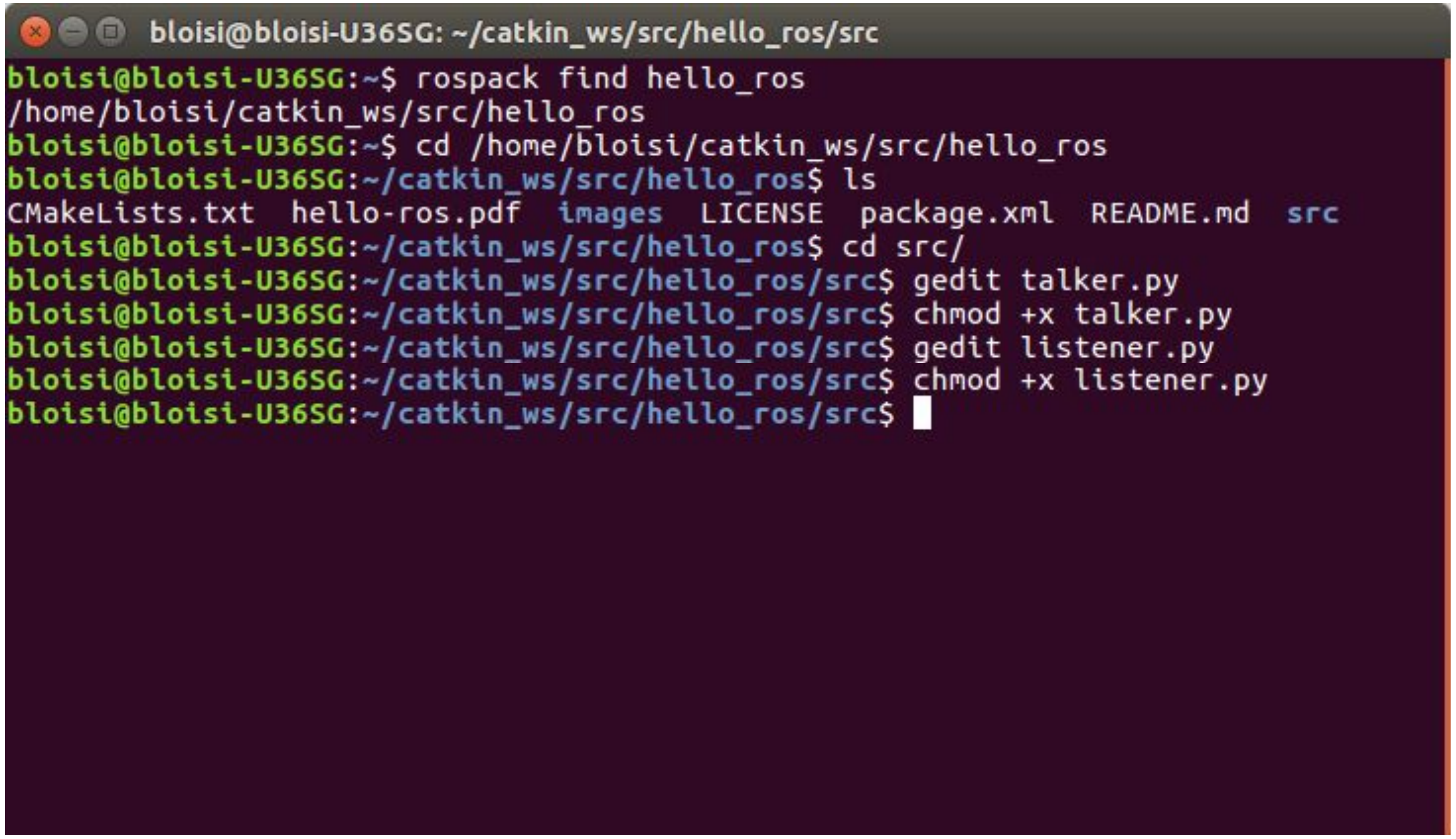
Codice del subscriber (listener.py)



```python
#!/usr/bin/env python
import rospy
from std_msgs.msg import String

def callback(data):
    rospy.loginfo(rospy.get_caller_id() + "I heard %s", data.data)

def listener():
    # In ROS, nodes are uniquely named. If two nodes with the same
    # name are launched, the previous one is kicked off. The
    # anonymous=True flag means that rospy will choose a unique
    # name for our 'listener' node so that multiple listeners can
    # run simultaneously.
    rospy.init_node('listener', anonymous=True)

    rospy.Subscriber("chatter", String, callback)

    # spin() simply keeps python from exiting until this node is stopped
    rospy.spin()

if __name__ == '__main__':
    listener()

```

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

Creiamo il subscriber (listener.py)

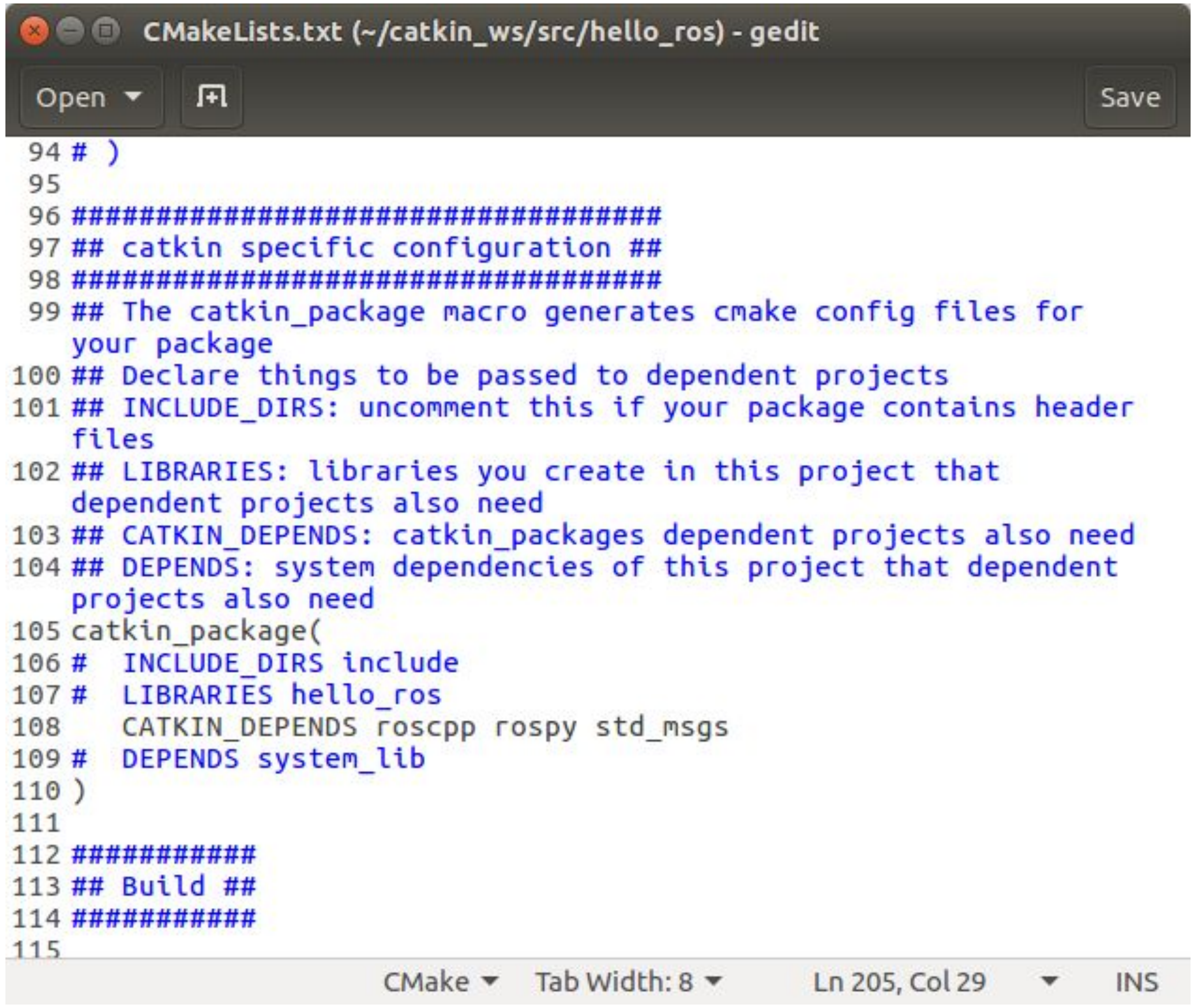


bloisi@bloisi-U36SG: ~/catkin_ws/src/hello_ros/src

```
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



```
# )

###################################
## catkin specific configuration ##
###################################
## The catkin_package macro generates cmake config files for your package
## Declare things to be passed to dependent projects
## INCLUDE_DIRS: uncomment this if your package contains header files
## LIBRARIES: libraries you create in this project that dependent projects also need
## CATKIN_DEPENDS: catkin_packages dependent projects also need
## DEPENDS: system dependencies of this project that dependent projects also need
catkin_package(
#  INCLUDE_DIRS include
#  LIBRARIES hello_ros
   CATKIN_DEPENDS roscpp rospy std_msgs
#  DEPENDS system_lib
)

###########
## Build ##
###########
```

http://wiki.ros.org/rospy_tutorials/Tutorials/Makefile

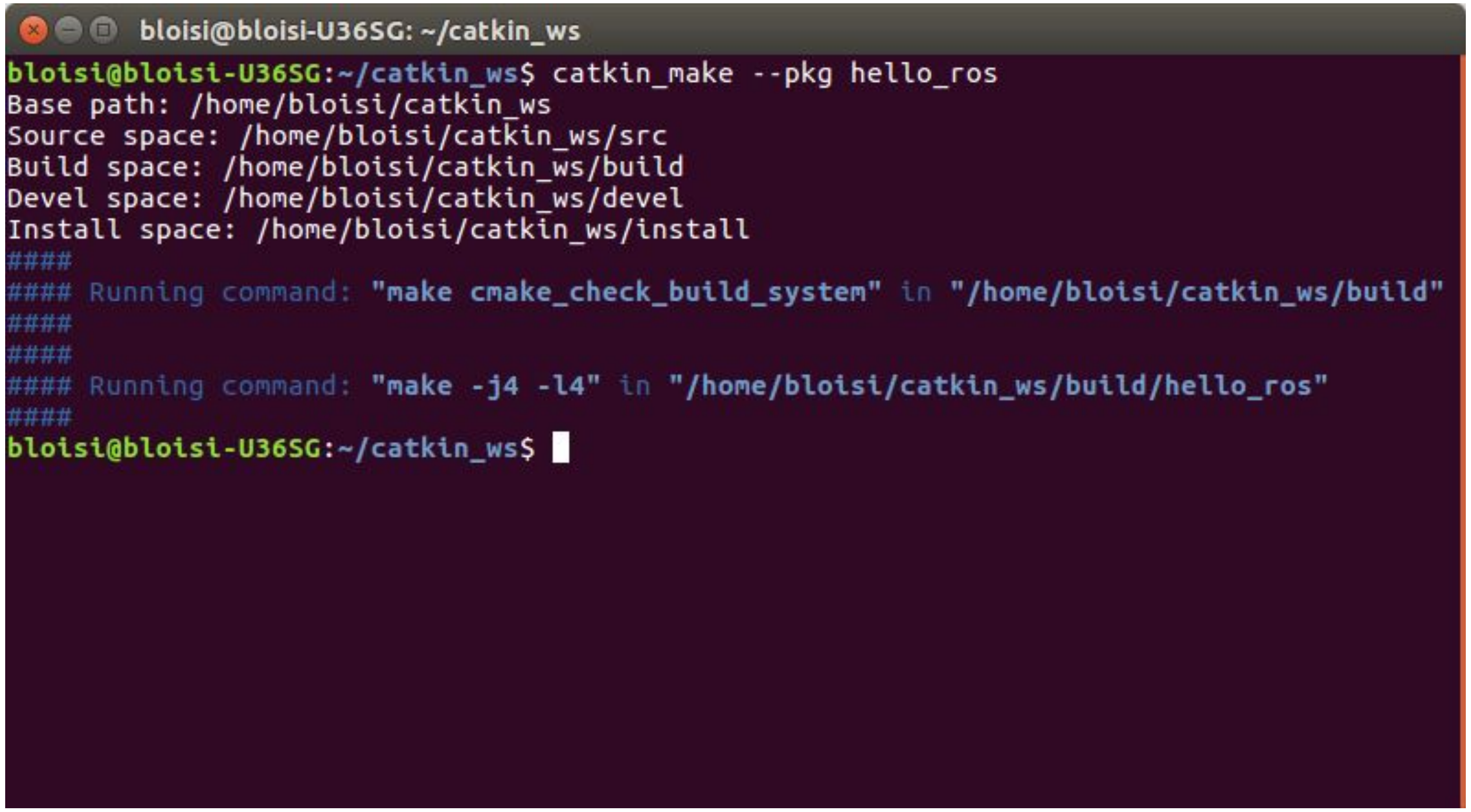
CMakeLists.txt

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

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

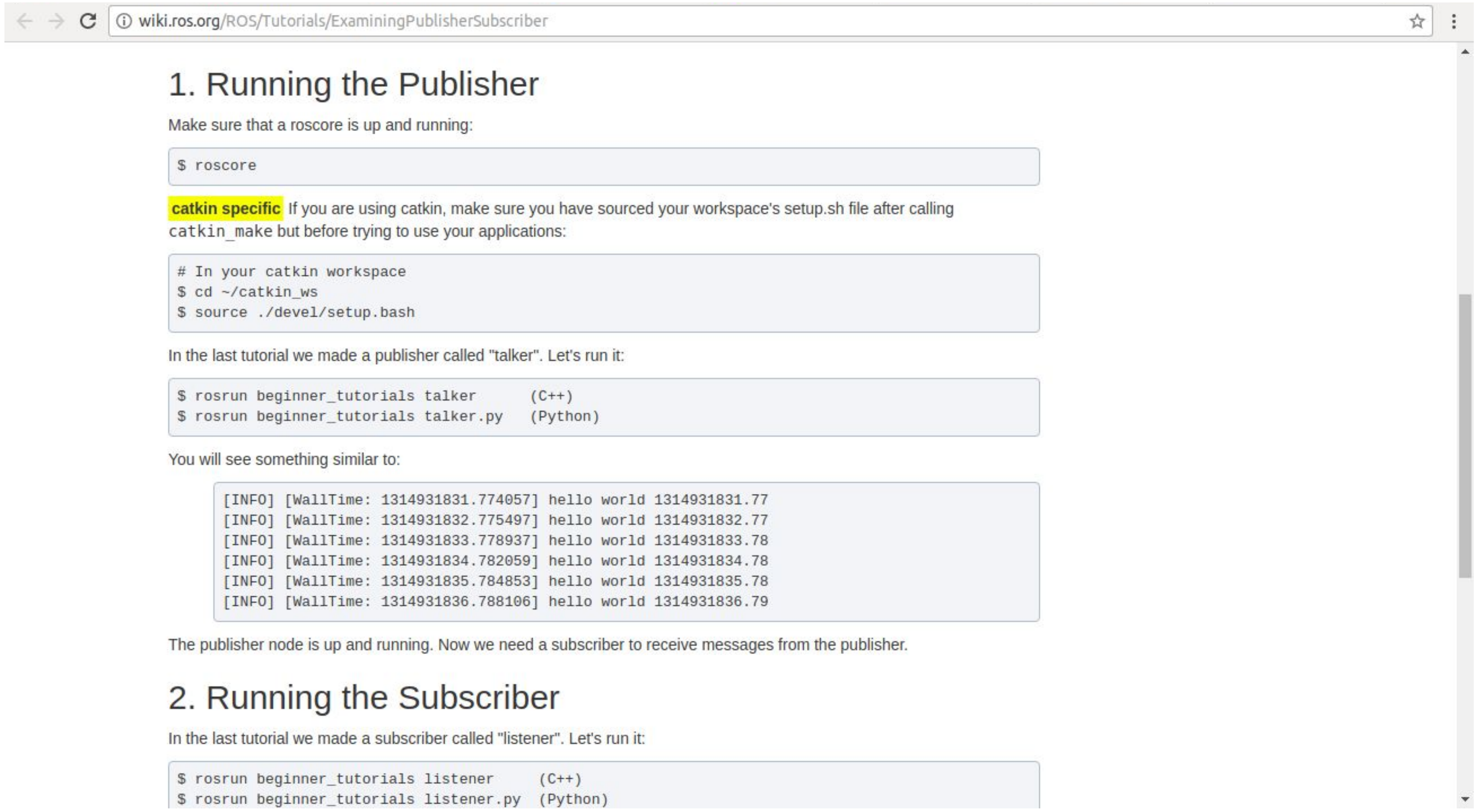
Compilazione con catkin_make

```
catkin_make --pkg hello_ros
```



```
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



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

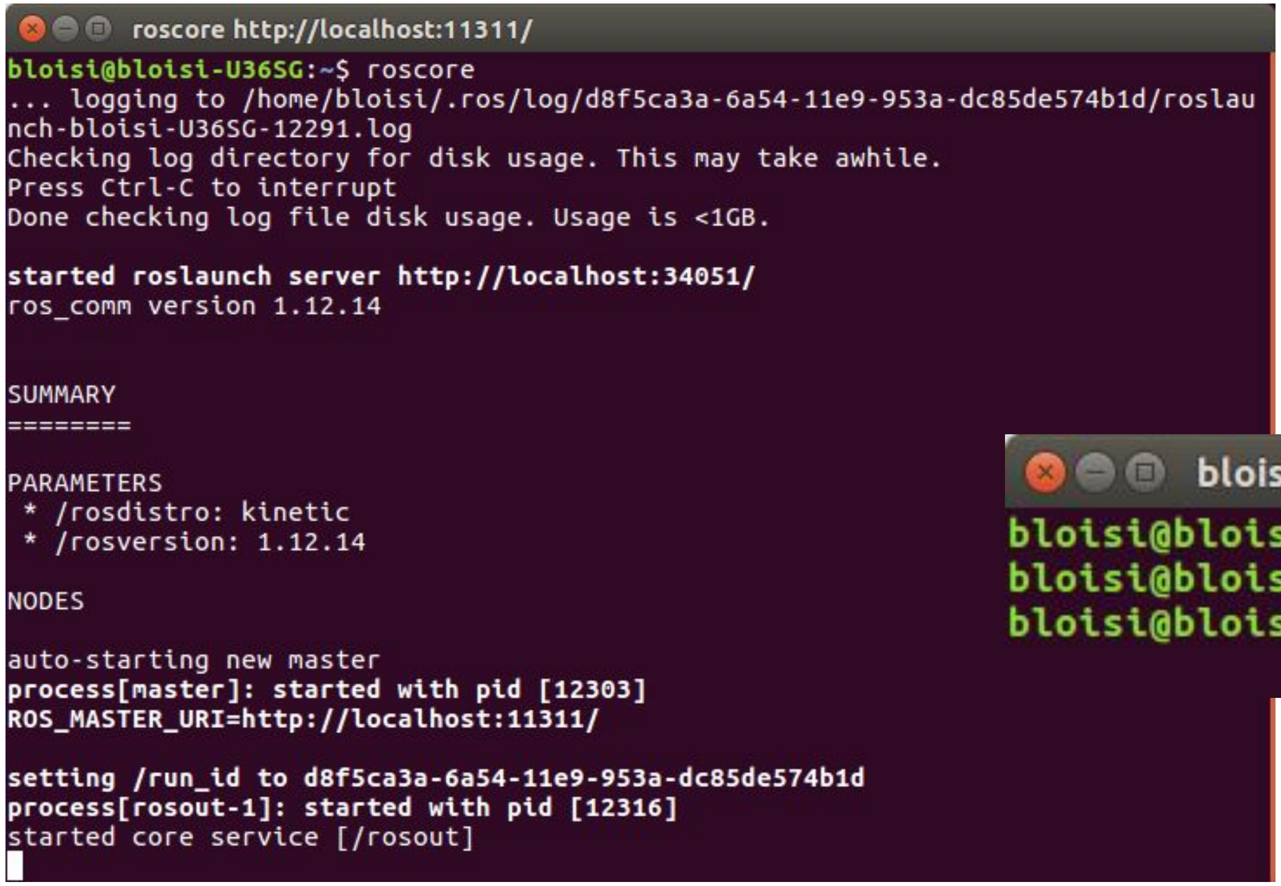
In the last tutorial we made a subscriber called "listener". Let's run it:

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

http://wiki.ros.org/ROS/Tutorials/ExaminingPublisherSubscriber

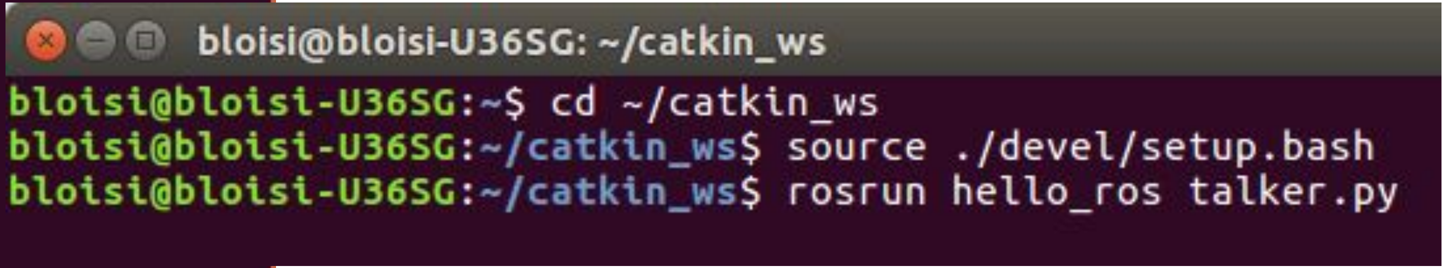
roscore + rosrun

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
 * /rosdistro: 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[rosout-1]: started with pid [12316]
started core service [/rosout]
```

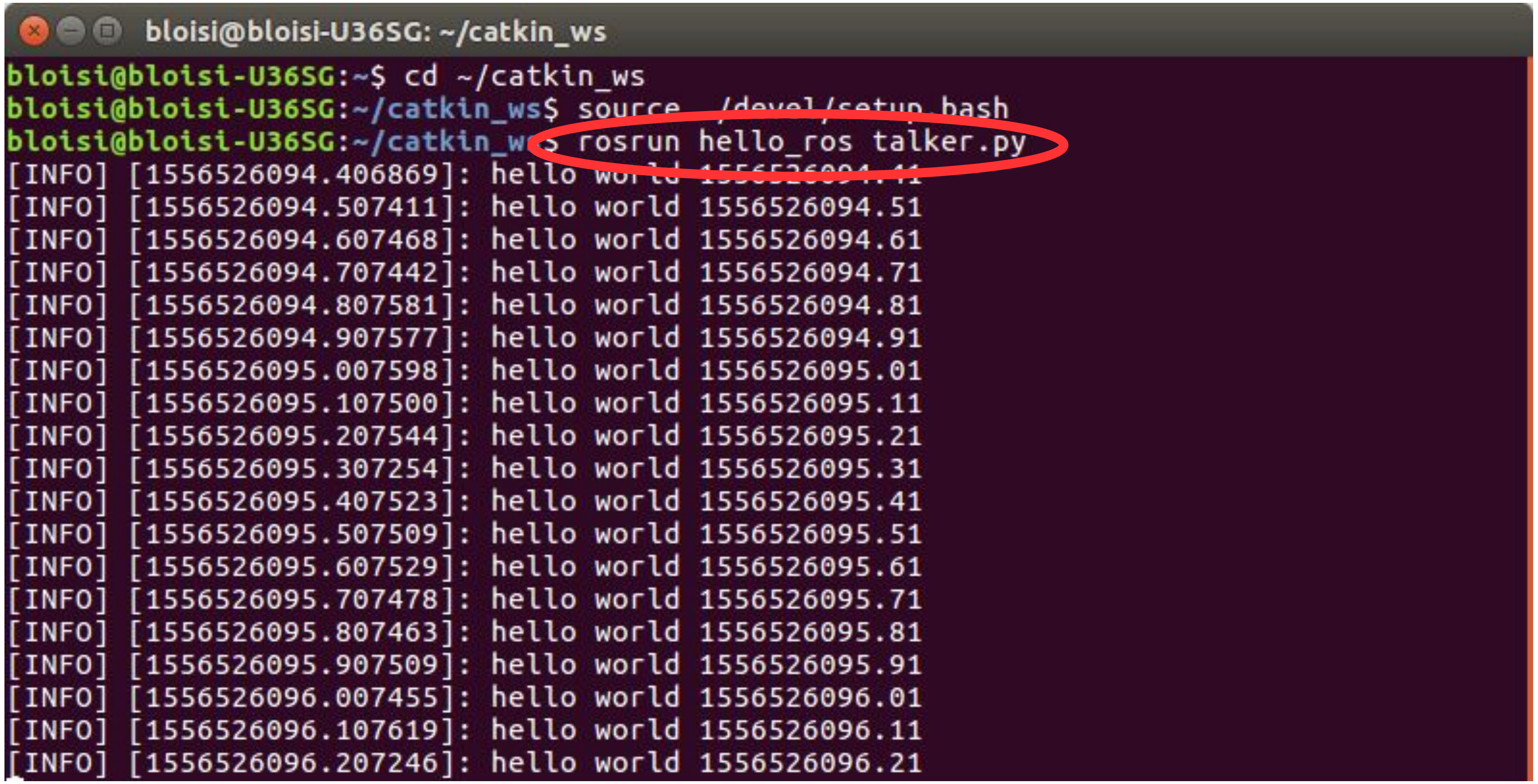
Apriamo un secondo terminale e lanciamo
`rosrun hello_ros talker.py`



```
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$ rosrun hello_ros talker.py
```

Cosa accade?

Esecuzione del nodo talker

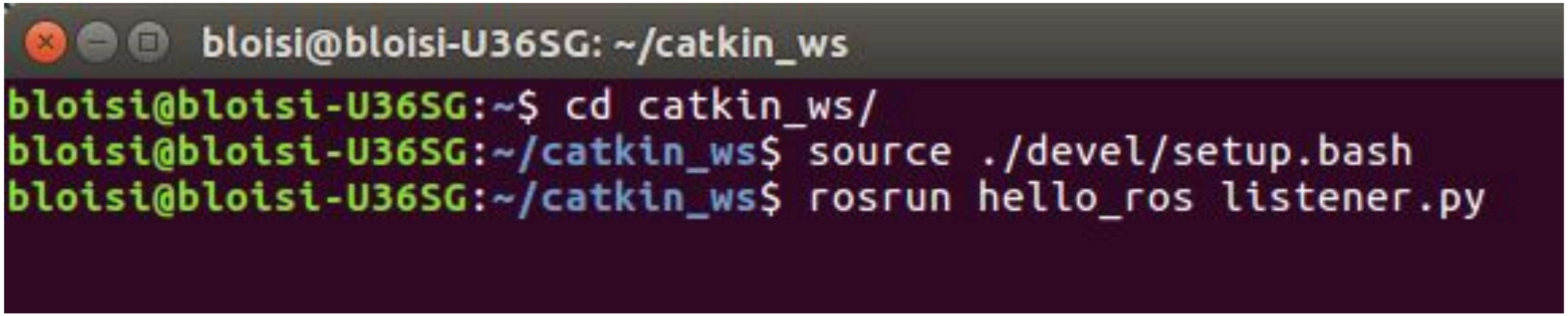


```
bloisi@bloisi-U36SG: ~/catkin_ws
bloisi@bloisi-U36SG:~$ cd ~/catkin_ws
bloisi@bloisi-U36SG:~/catkin_ws$ source ./devel/setup.bash
bloisi@bloisi-U36SG:~/catkin_ws$ rosrun hello_ros talker.py
[INFO] [1556526094.406869]: hello world 1556526094.41
[INFO] [1556526094.507411]: hello world 1556526094.51
[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

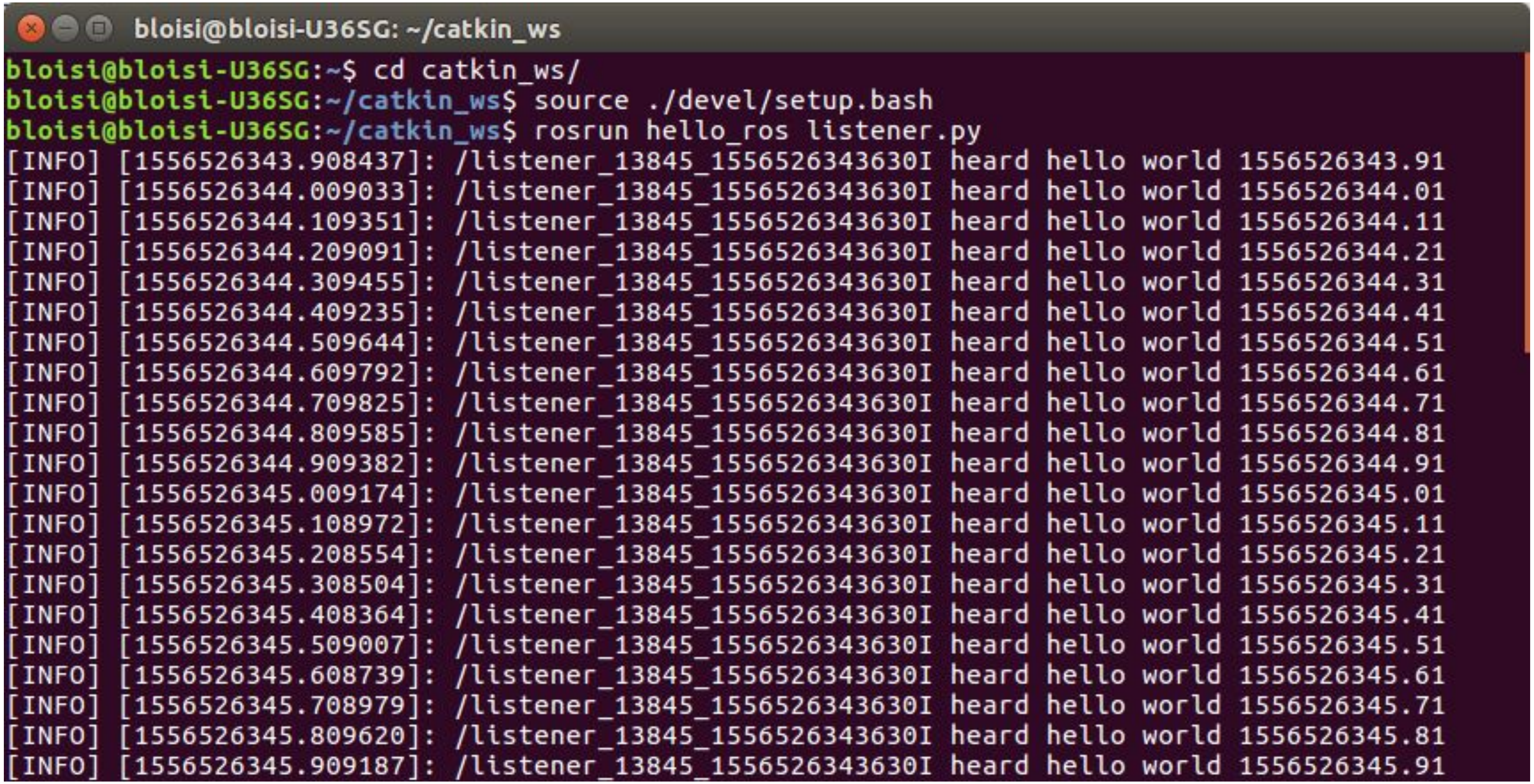
Apriamo un terzo terminale e
lanciamo

```
rosrun hello_ros listener.py
```



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

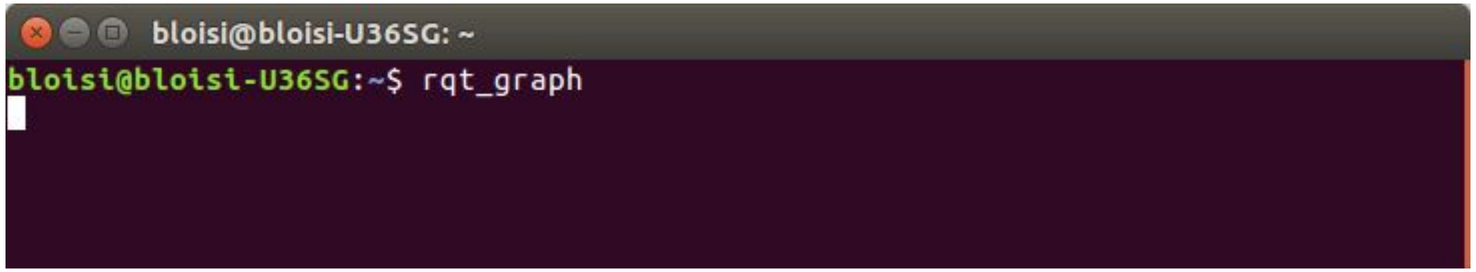
Esecuzione del nodo listener



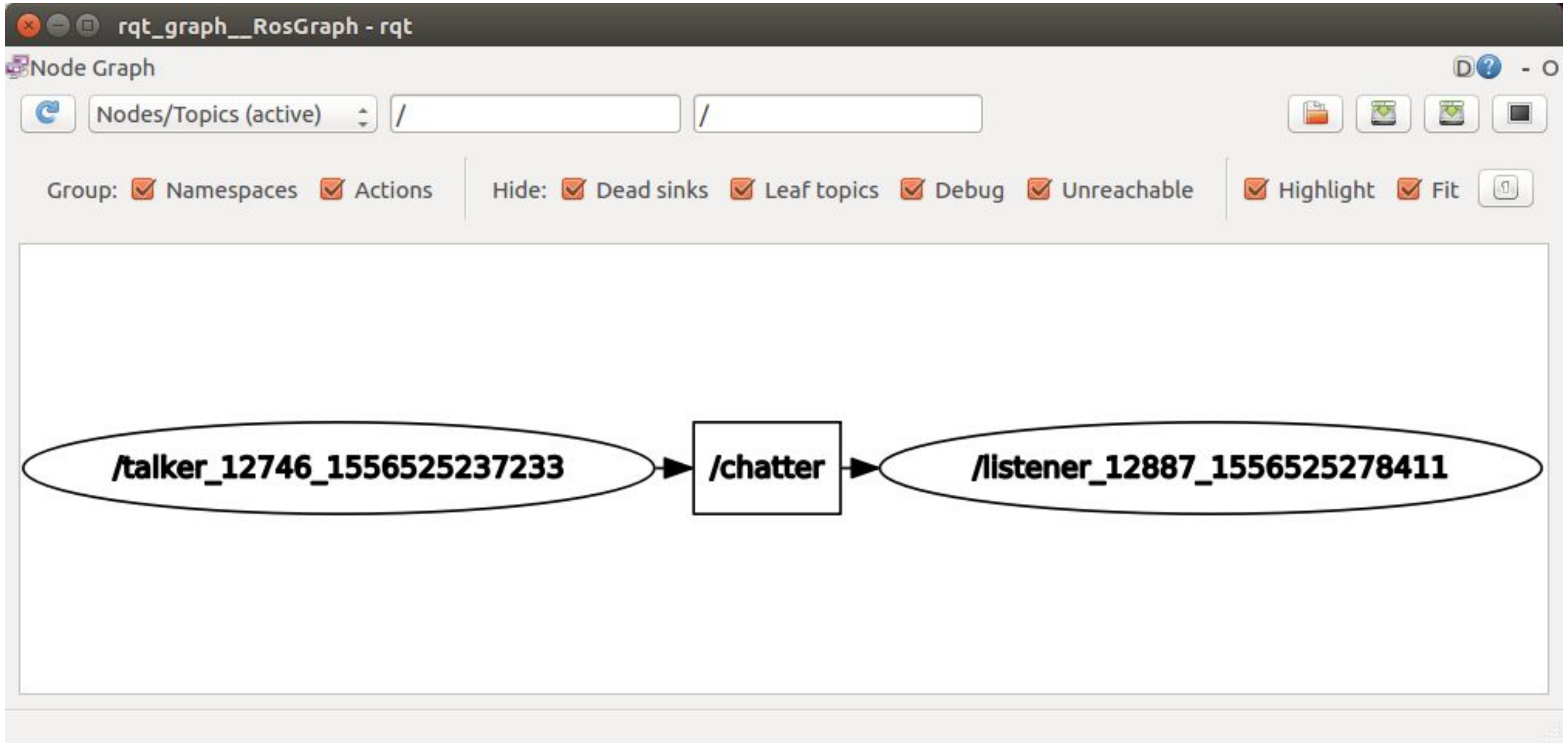
bloisi@bloisi-U36SG: ~/catkin_ws

```
bloisi@bloisi-U36SG:~$ cd catkin_ws/
bloisi@bloisi-U36SG:~/catkin_ws$ source ./devel/setup.bash
bloisi@bloisi-U36SG:~/catkin_ws$ rosrun 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

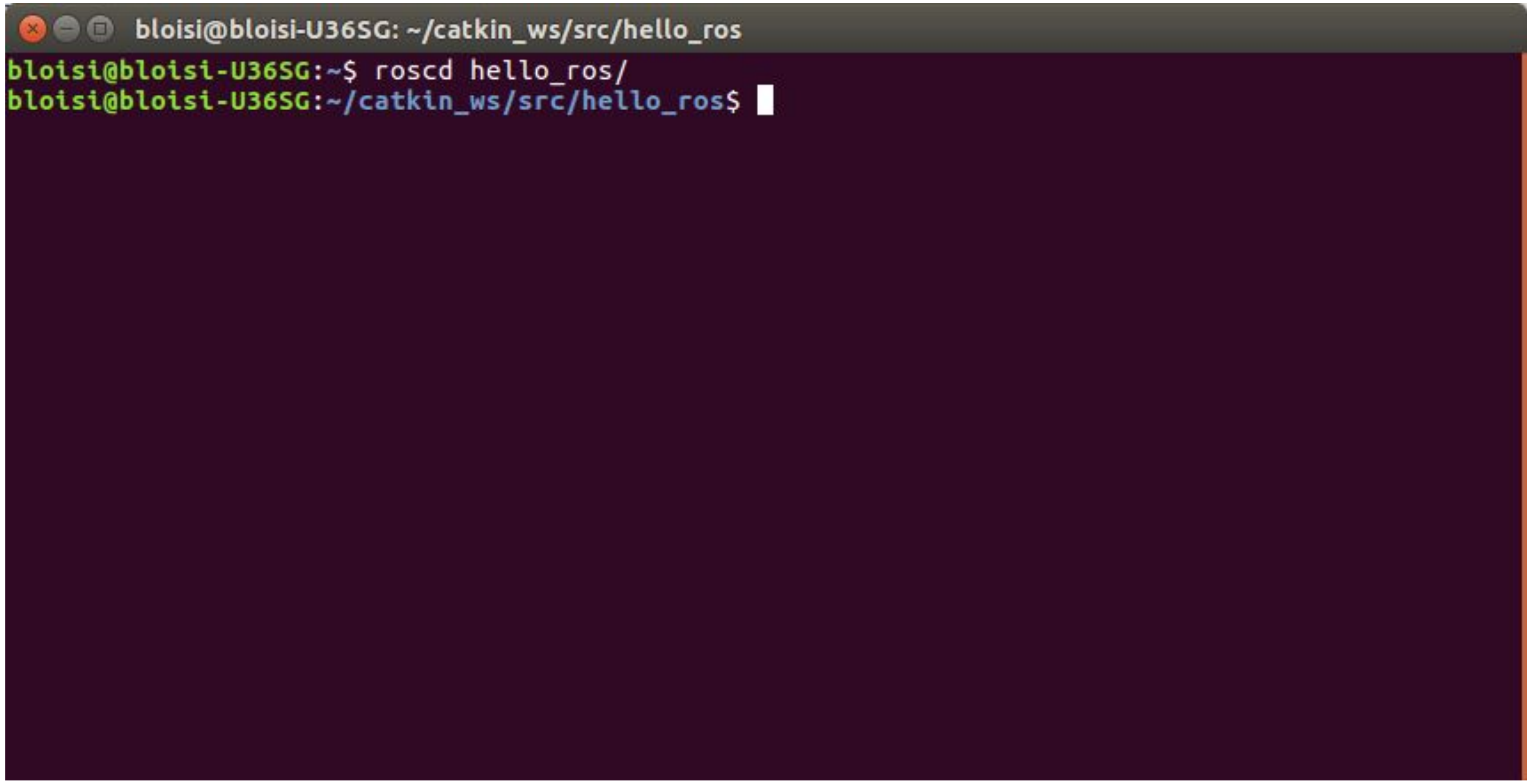


```
bloisi@bloisi-U36SG:~$ rqt_graph
```


rqt_graph__RosGraph - rqt
Node Graph
Nodes/Topics (active)
/
/
Group: ✓ Namespaces ✓ Actions
Hide: ✓ Dead sinks ✓ Leaf topics ✓ Debug ✓ Unreachable
✓ Highlight ✓ Fit
/talker_12746_1556525237233
/chatter
/listener_12887_1556525278411

roscd

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

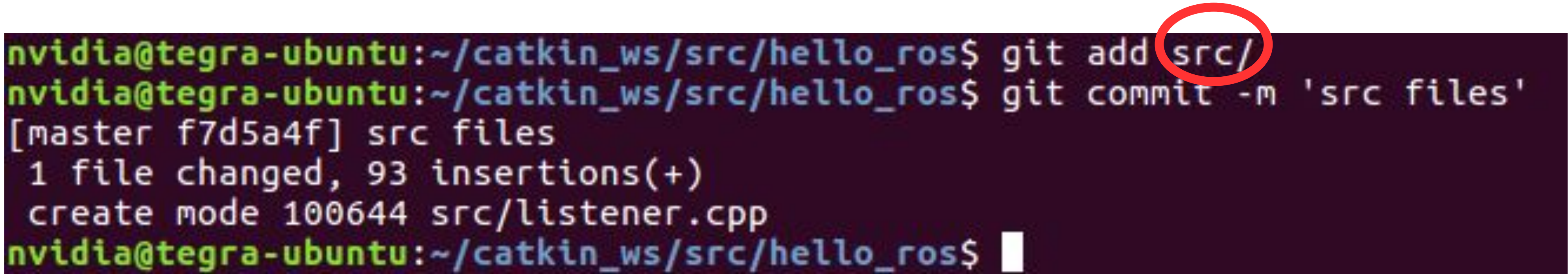


```
bloisi@bloisi-U36SG: ~/catkin_ws/src/hello_ros
bloisi@bloisi-U36SG:~$ roscd hello_ros/
bloisi@bloisi-U36SG:~/catkin_ws/src/hello_ros$
```

Aggiorniamo il repository locale

Aggiorniamo il repository locale con la cartella src

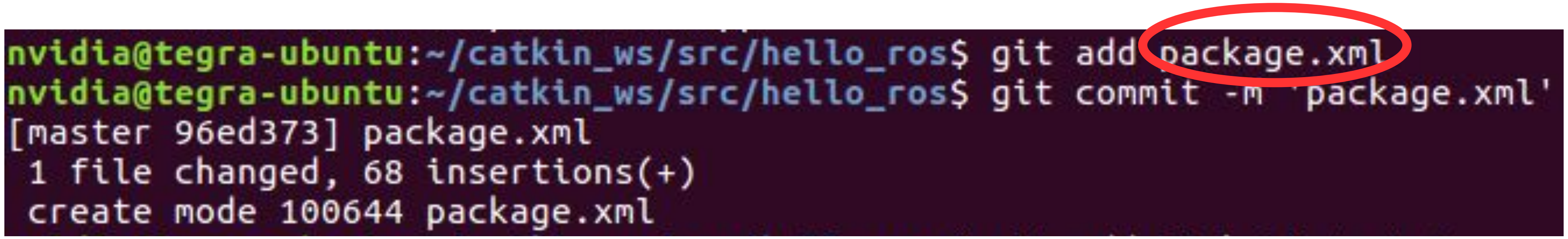
```
git add
git commit
```



```
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)

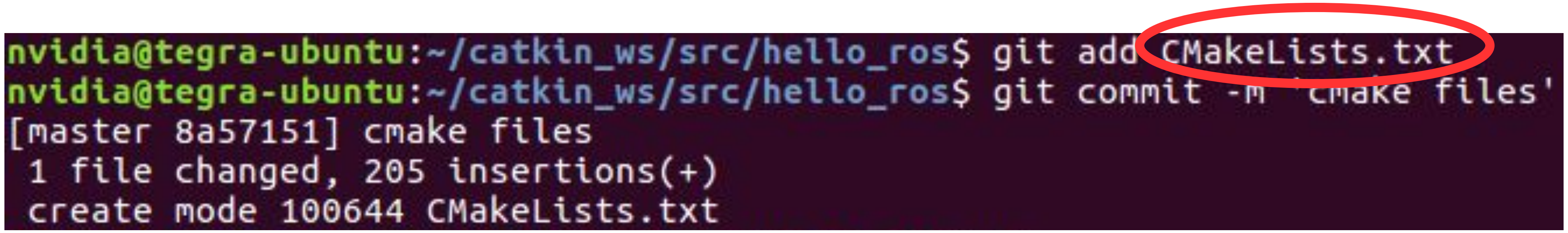
```
git add
git commit
```



```
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)

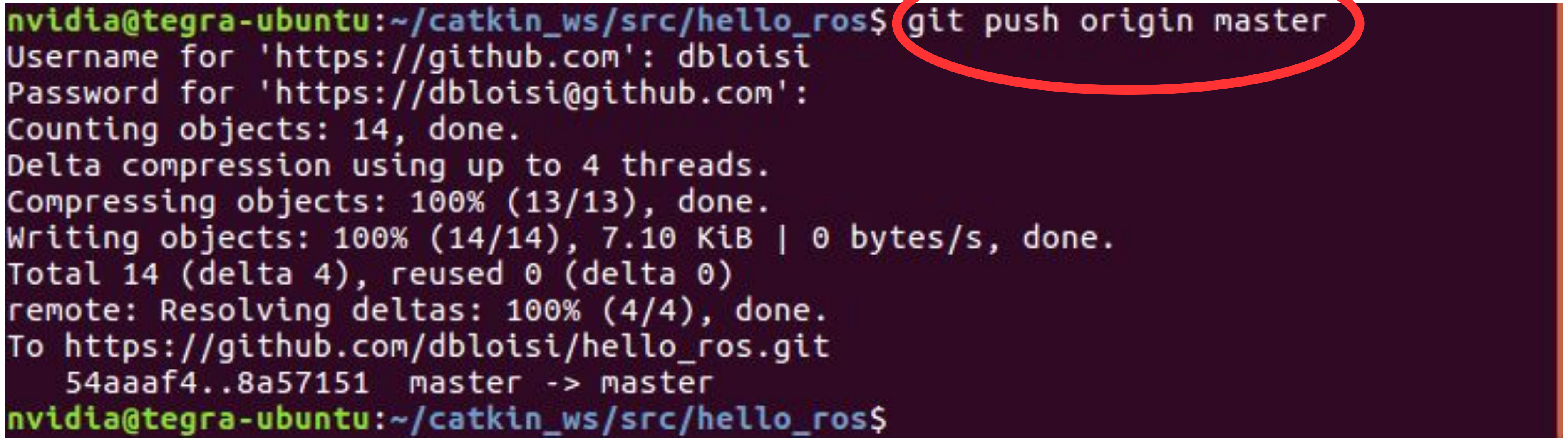
```
git add
git commit
```



```
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

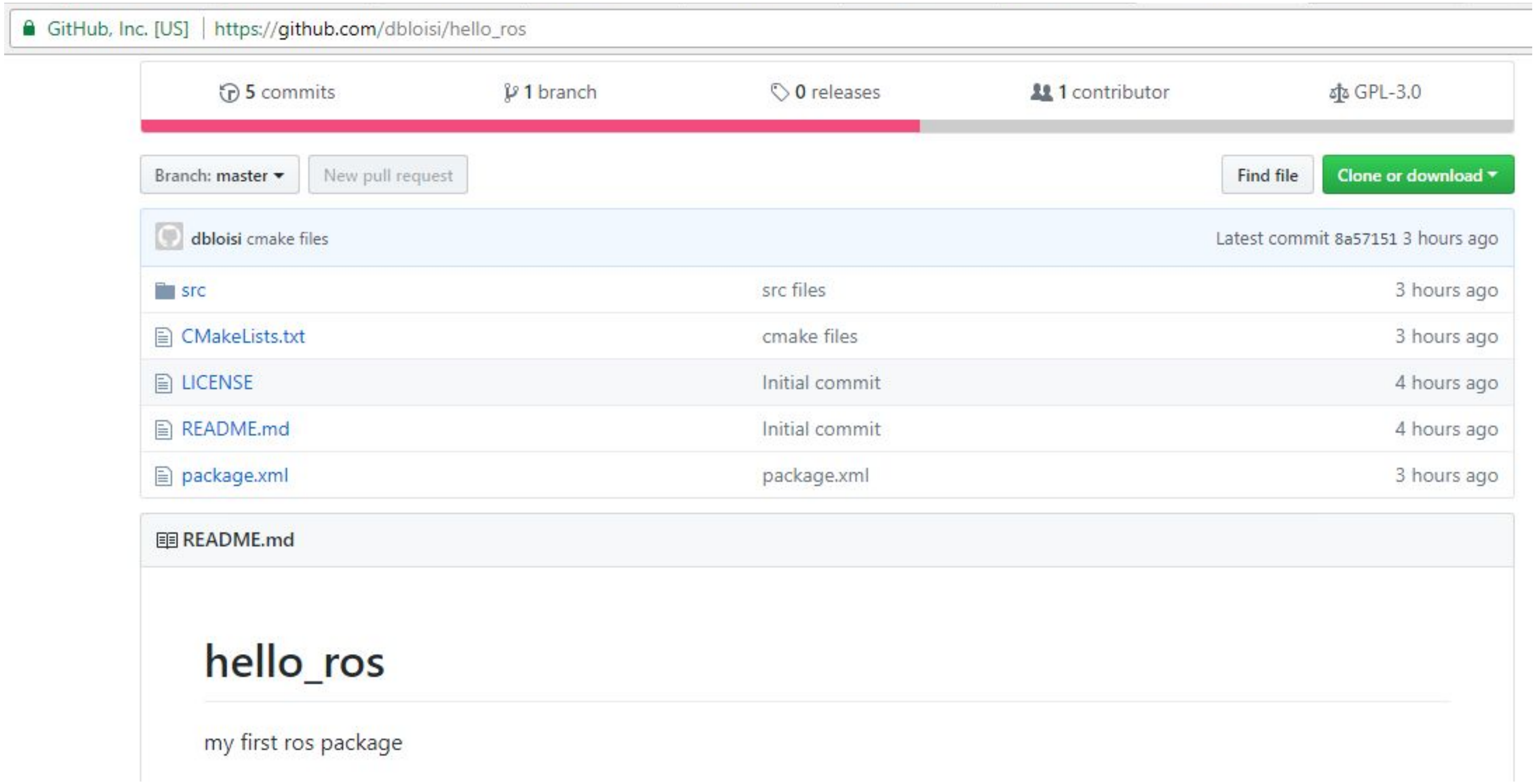
`git push`



```
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

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

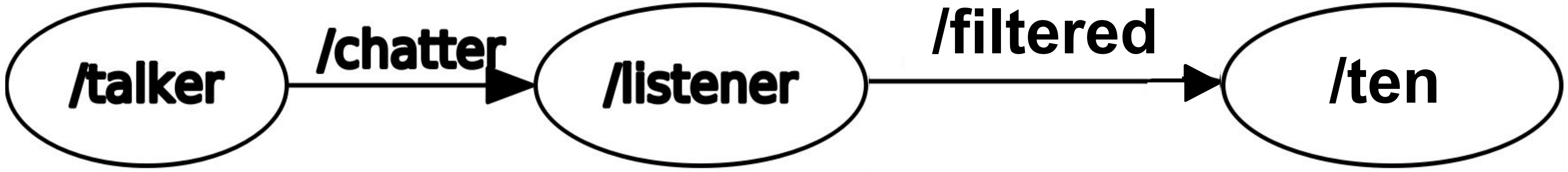
my first ros package

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

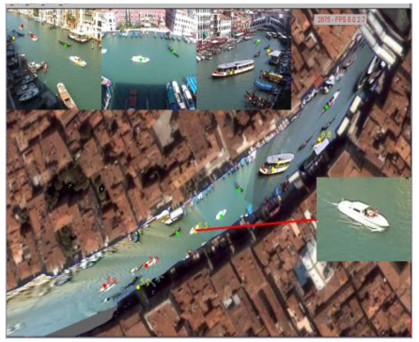
5. Modificare il codice del listener in modo che pubblichi a sua volta un messaggio dopo aver ascoltato 10 messaggi provenienti dal talker
6. Creare un nuovo nodo ten che ascolti i messaggi del listener e li stampi a video
7. Aggiornare il repository remoto



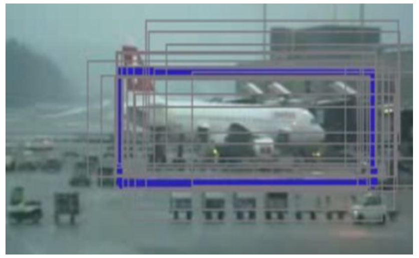
UNIVERSITÀ DEGLI STUDI DELLA BASILICATA

Corso di Sistemi Informativi
A.A. 2018/19

Docente
Domenico Daniele Bloisi



git + ROS (Python)



Maggio 2019