UNIVERSITÀ DEGLI STUDI DELLA BASILICATA

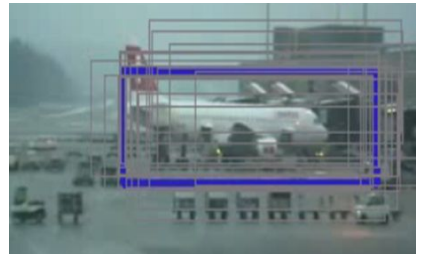


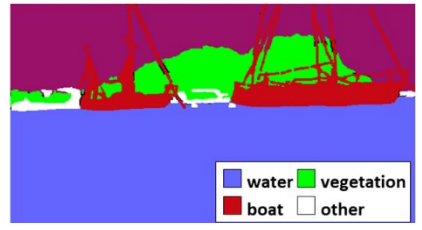
Corso di Visione e Percezione
A.A. 2019/2020

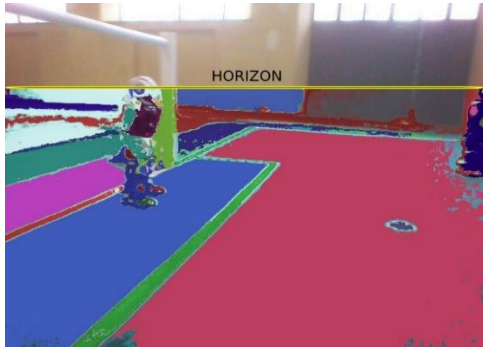
Docente
Domenico Daniele Bloisi

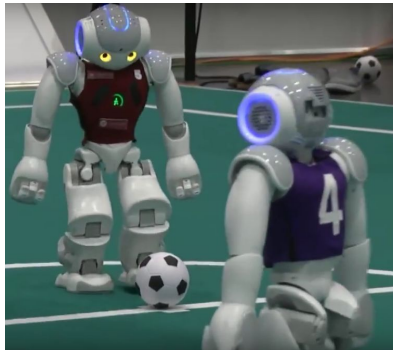
git +

ROS (Python)








Aprile 2020

git + ROS

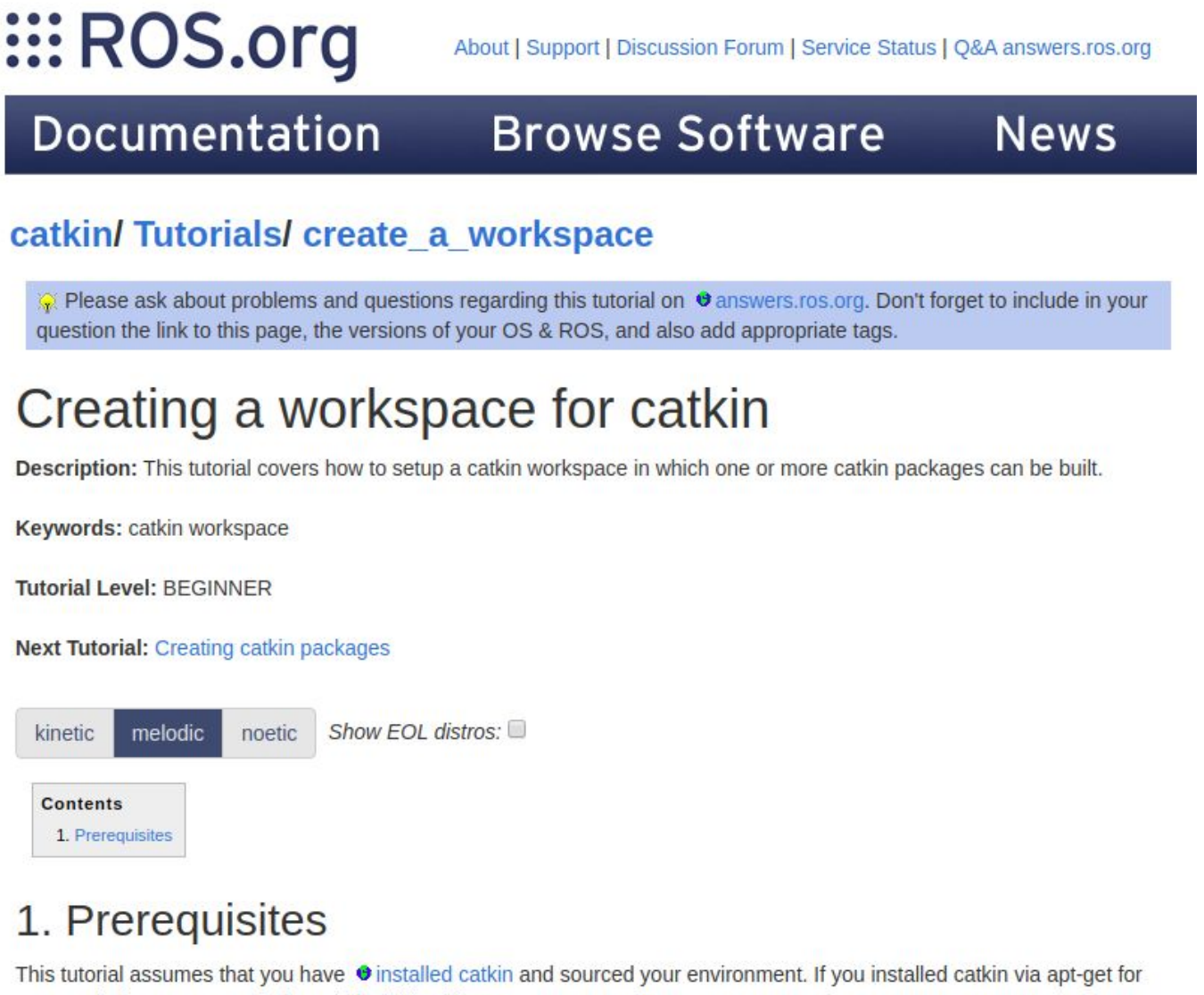
Esempio pratico

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

Prerequisiti

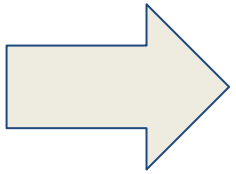
Avere un workspace per Catkin

http://wiki.ros.org/catkin/Tutorials/create_a_workspace



Prerequisiti

Python 3 compatibility



1. Prerequisites

This tutorial assumes that you have installed catkin and sourced your environment. If you installed catkin via apt-get for ROS melodic, your command would look like this:

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

Let's create and build a catkin workspace:

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

The catkin_make command is a convenience tool for working with catkin workspaces. Running it the first time in your workspace, it will create a `CMakeLists.txt` link in your 'src' folder.

Python 3 users: note, if you are building ROS from source to achieve Python 3 compatibility, and have setup your system appropriately (ie: have the Python 3 versions of all the required ROS Python packages installed, such as `catkin`) the first catkin_make command in a clean catkin workspace must be:

```
$ catkin_make -DPYTHON_EXECUTABLE=/usr/bin/python3
```

This will configure catkin_make with Python 3. You may then proceed to use just `catkin_make` for subsequent builds.

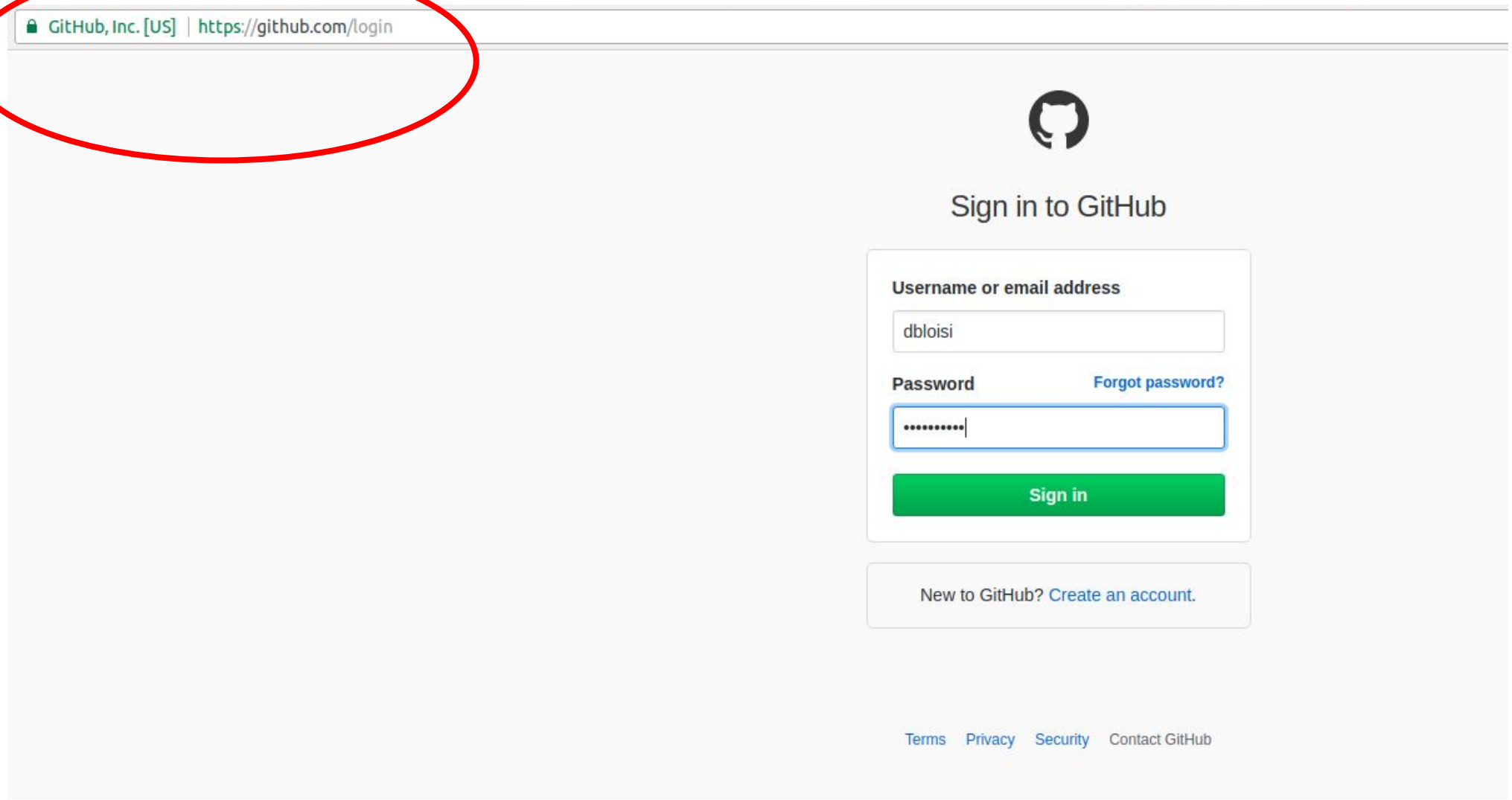
Additionally, if you look in your current directory you should now have a 'build' and 'devel' folder. Inside the 'devel' folder you can see that there are now several setup.*sh files. Sourcing any of these files will overlay this workspace on top of your environment. To understand more about this see the general catkin documentation: catkin. Before continuing source your new setup.*sh file:

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

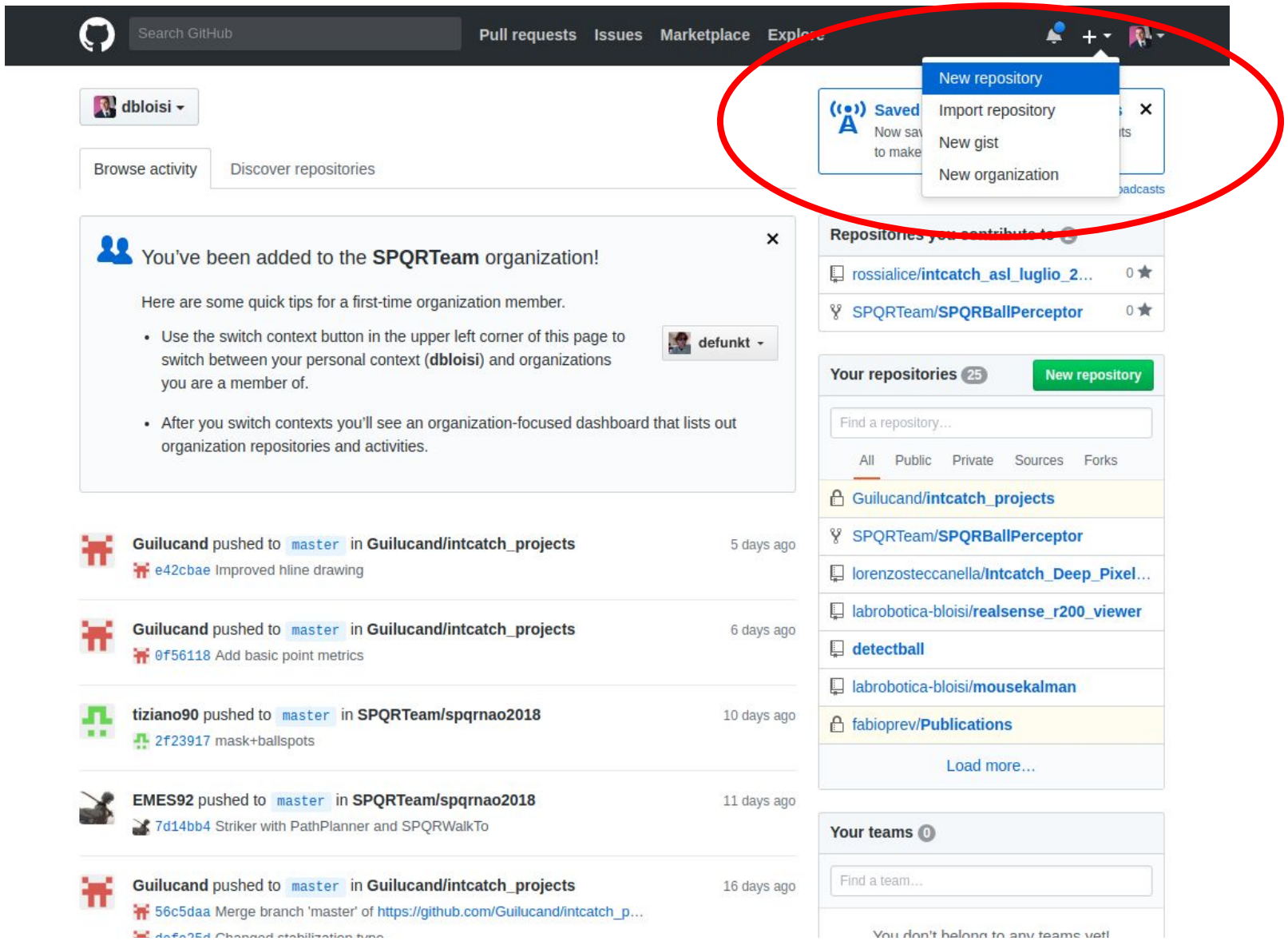
To make sure your workspace is properly overlayed by the setup script, make sure `ROS_PACKAGE_PATH` environment variable includes the directory you're in.

```
$ echo $ROS_PACKAGE_PATH
```

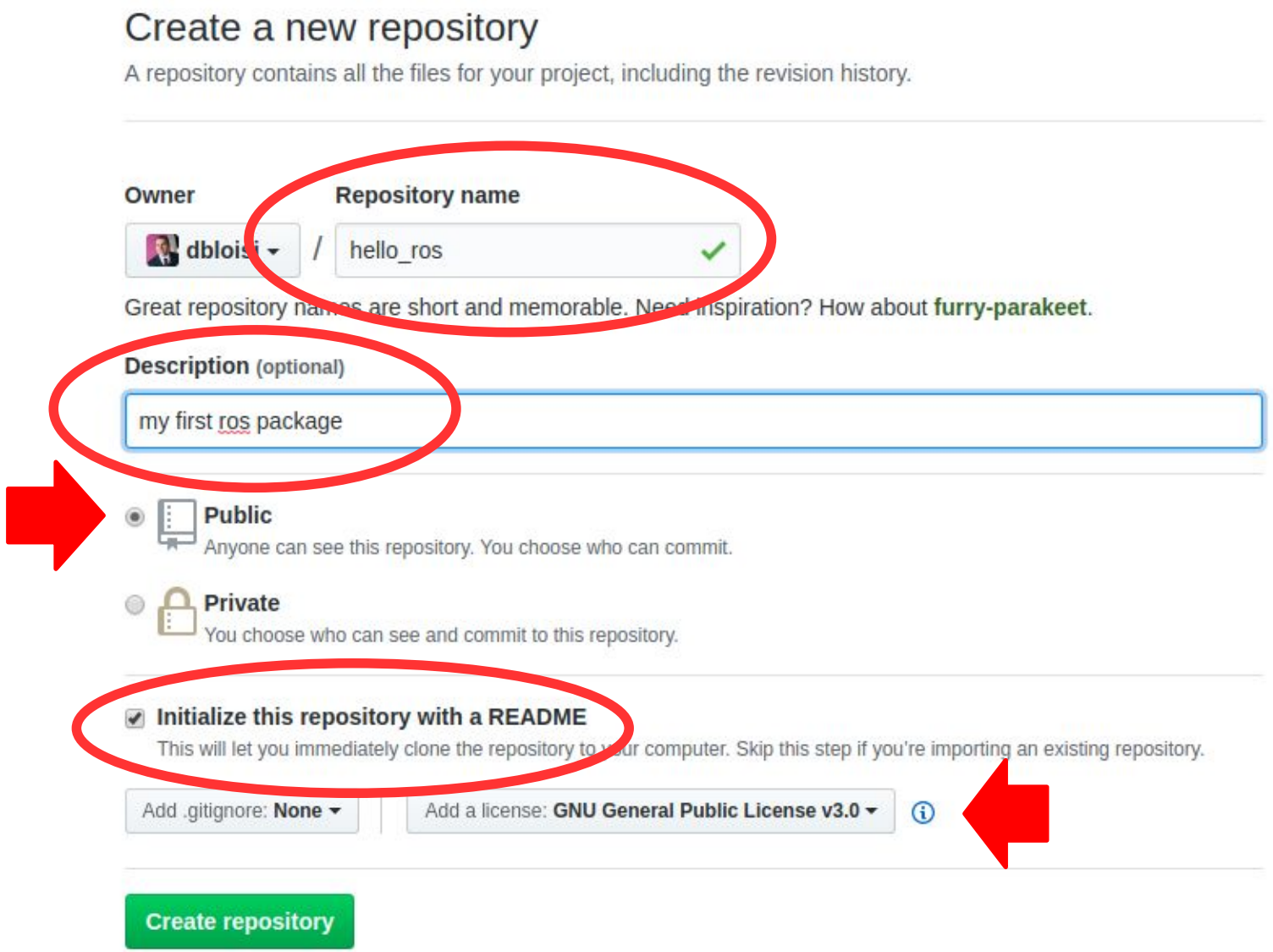
Server git


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

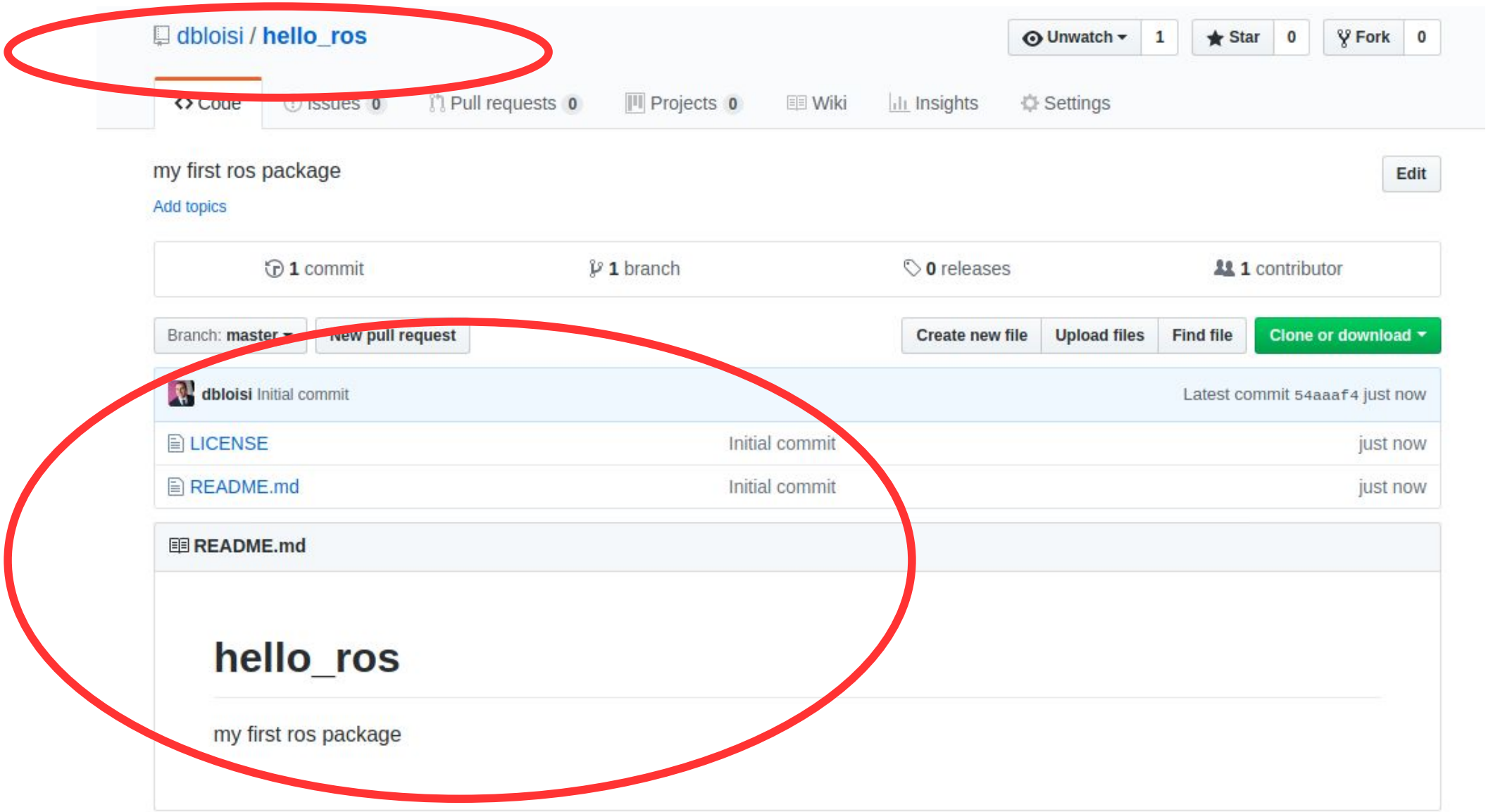
Creare un repository git



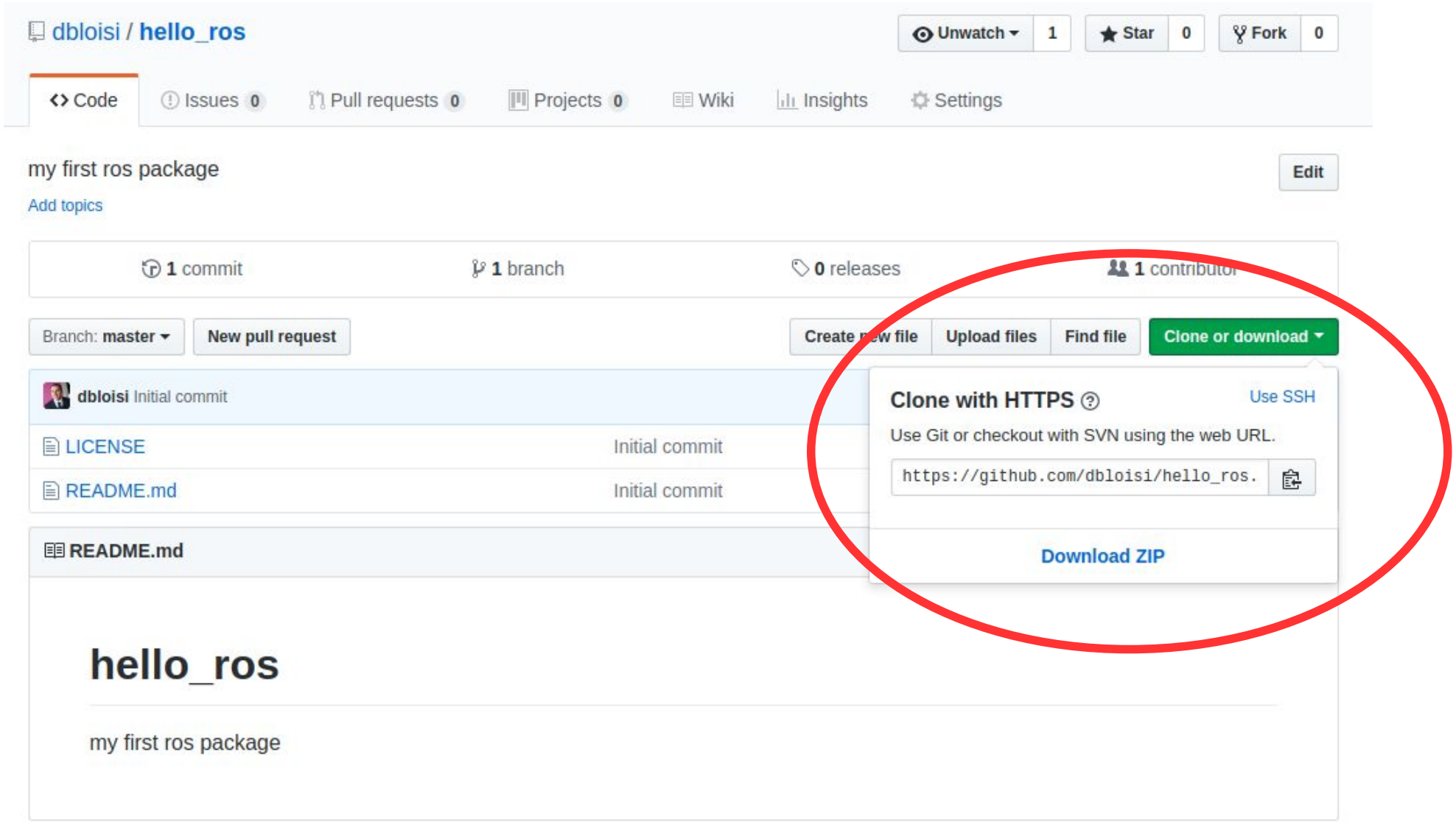
Repository name



Repository creato



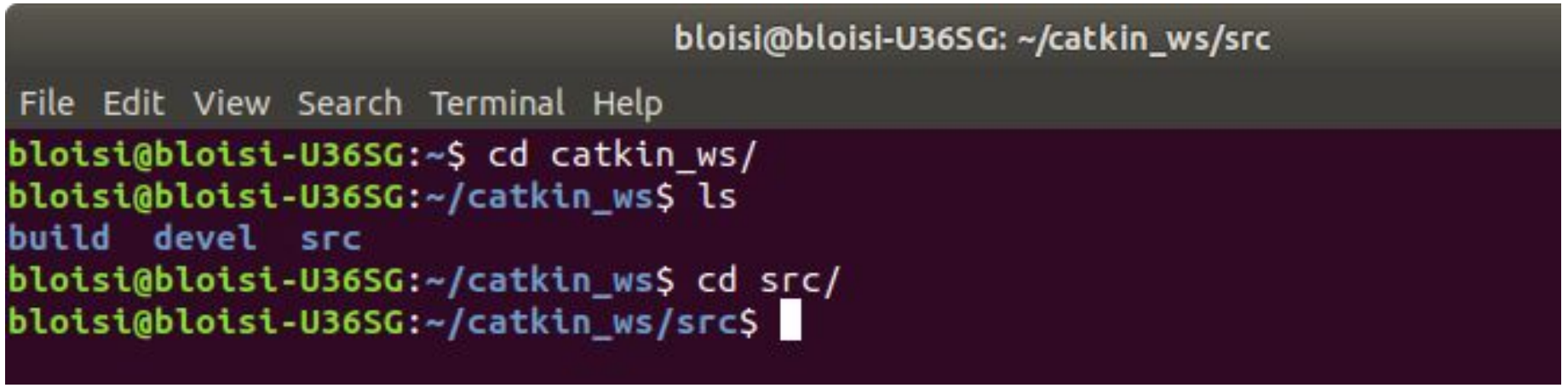
Indirizzo del repository remoto



Creazione del repository locale

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

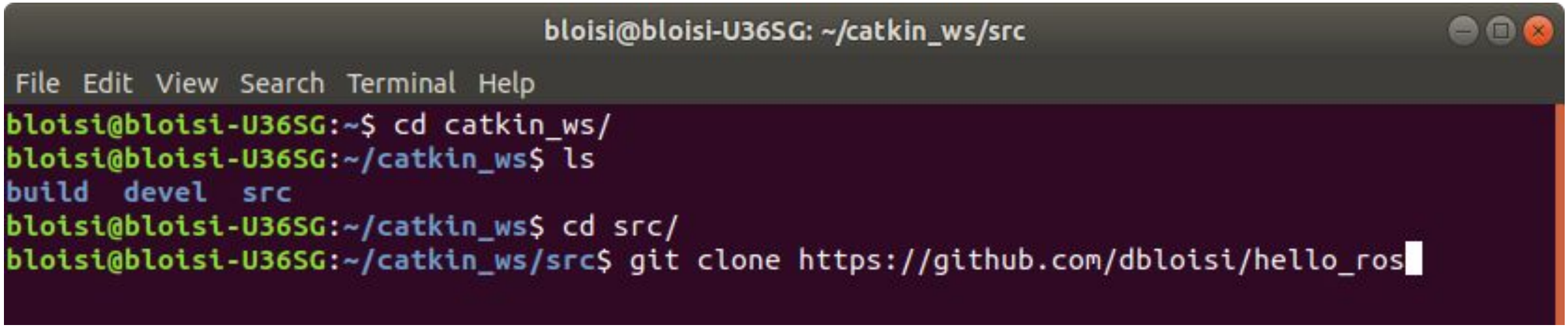
Creiamo il repository locale nella cartella src del nostro workspace ROS → ~/catkin_ws/src



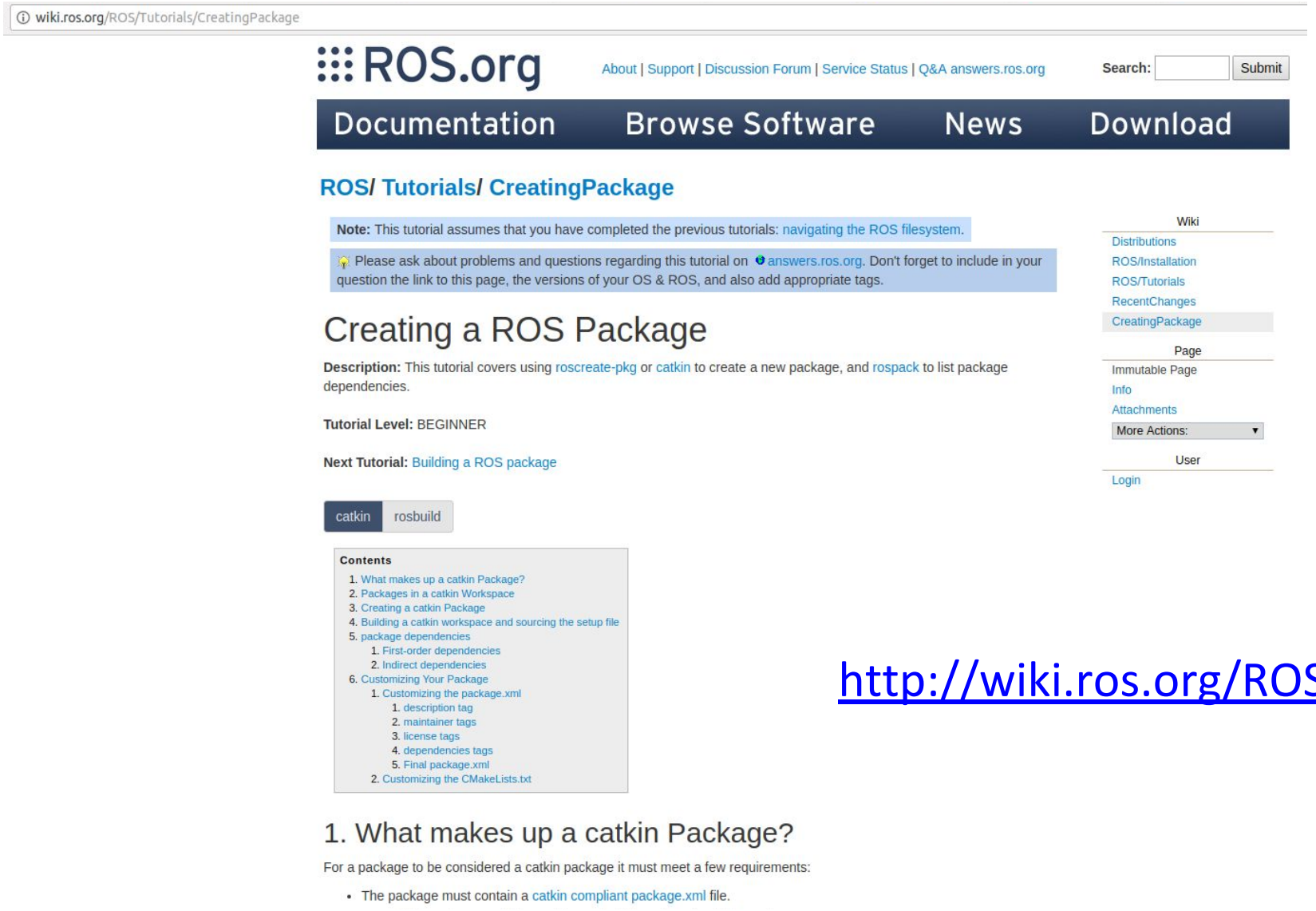
Creazione del repository locale

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

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

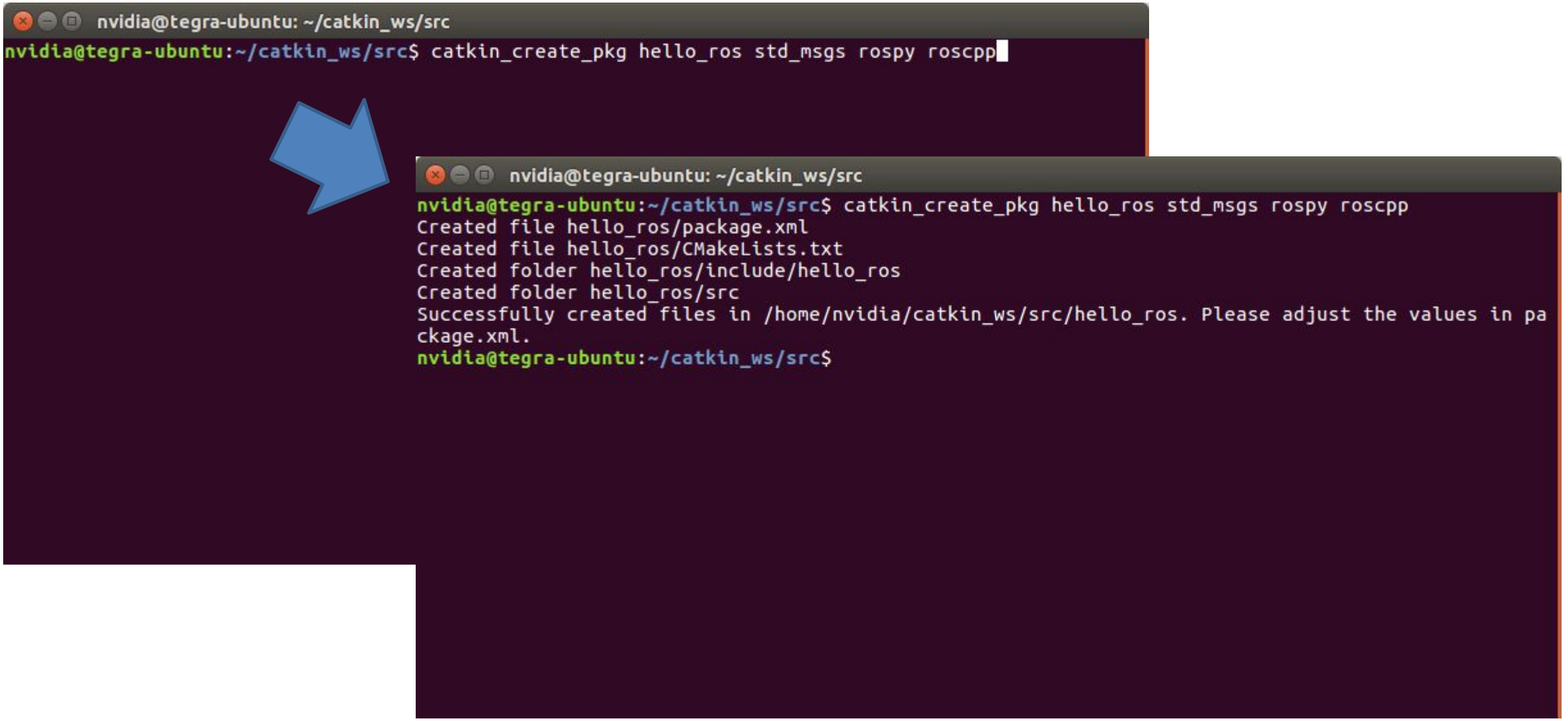


Creating a ROS package

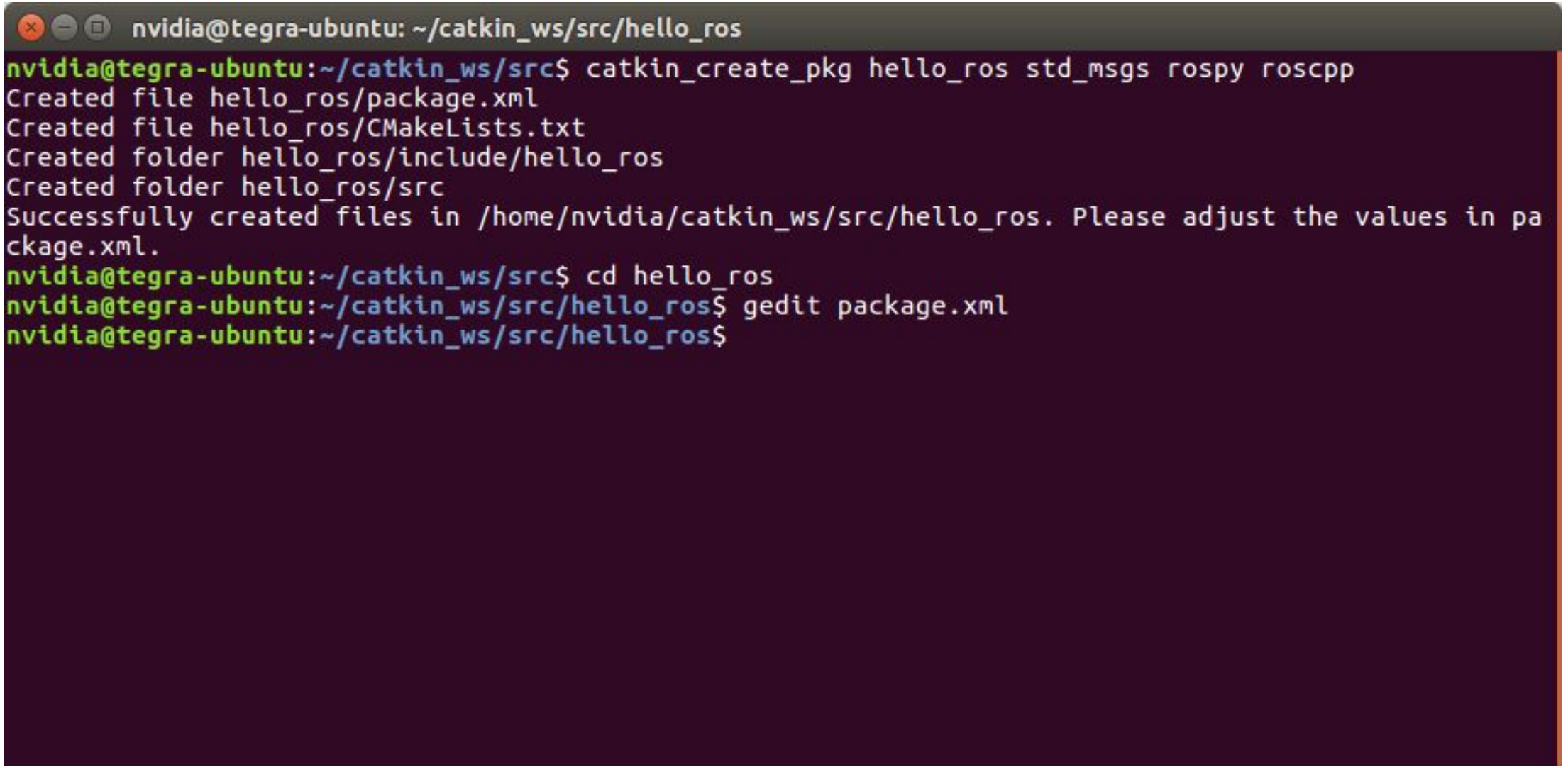


http://wiki.ros.org/ROS/Tutorials/CreatingPackage

catkin_create_pkg



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

*package.xml (~/catkin_ws/src/hello_ros) - gedit

Open 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

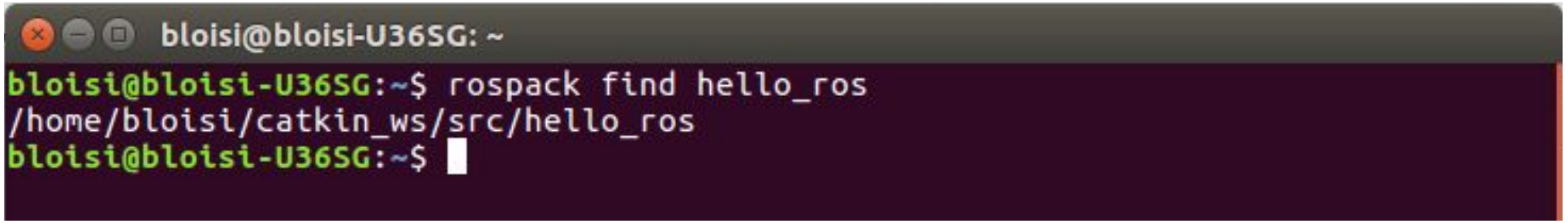
```
  <!-- Examples: -->
  <!-- Use depend as a shortcut for packages that are both build and exec dependencies -->
  <!--   <depend>roscpp</depend> -->
  <!--   Note that this is equivalent to the following: -->
  <!--   <build_depend>roscpp</build_depend> -->
  <!--   <exec_depend>roscpp</exec_depend>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

XML ▾ Tab Width: 8 ▾ Ln 16, Col 18 ▾ INS

Finding a ROS package

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

```
rospack find hello_ros
```



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

Finding a ROS package

Se il comando

`rospack find hello_ros`

non va a buon fine

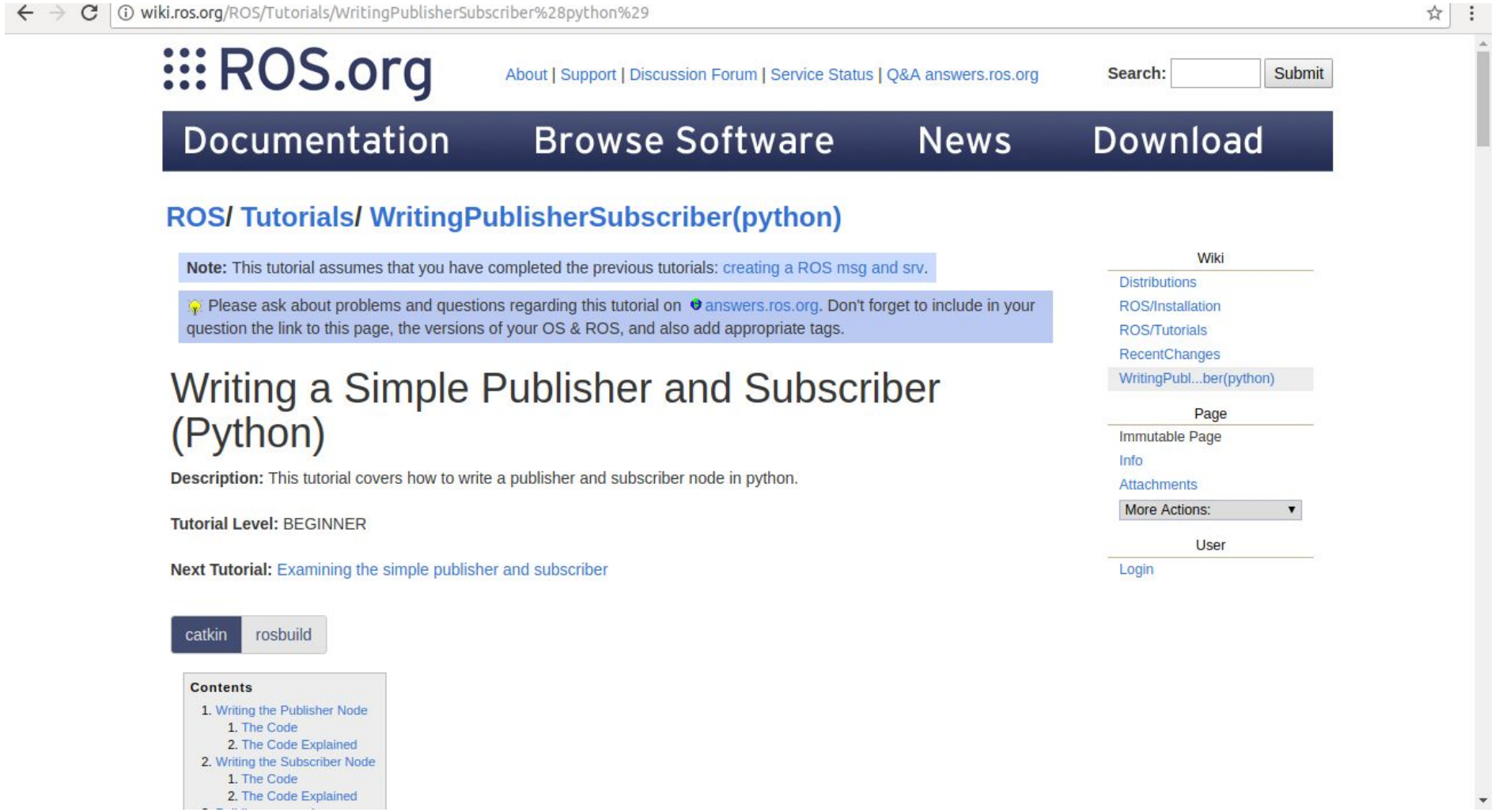
```
bloisi@bloisi-U36SG:~/catkin_ws$ rospack find hello_ros
[rospack] Error: package 'hello_ros' not found
```

provare con il comando

`source devel/setup.bash`

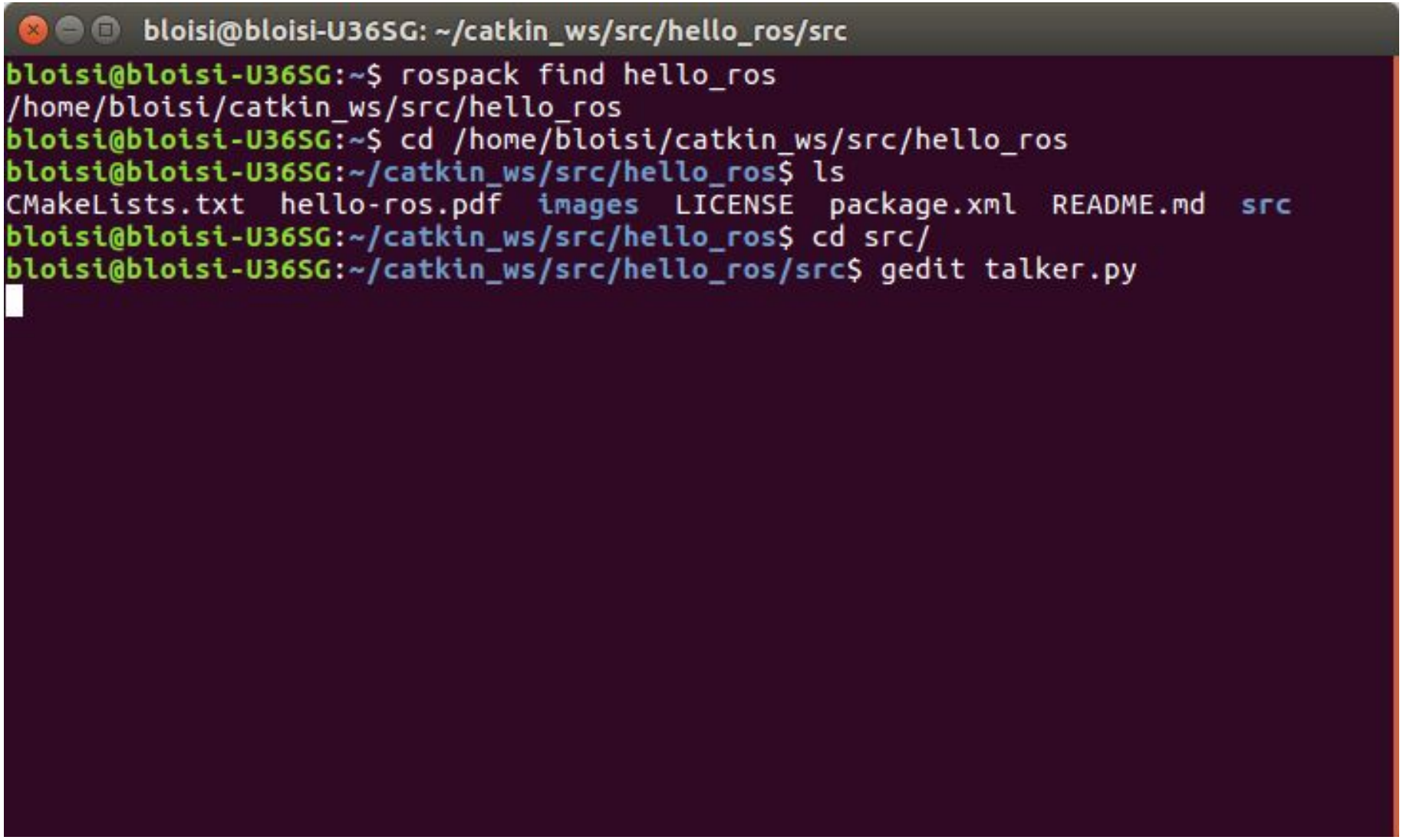
```
bloisi@bloisi-U36SG:~/catkin_ws$ source devel/setup.bash
bloisi@bloisi-U36SG:~/catkin_ws$ rospack find hello_ros
/home/bloisi/catkin_ws/src/hello_ros
bloisi@bloisi-U36SG:~/catkin_ws$
```

Esempio Publisher/Subscriber Python



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

Creiamo il publisher (talker.py)



Codice del publisher (talker.py)

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

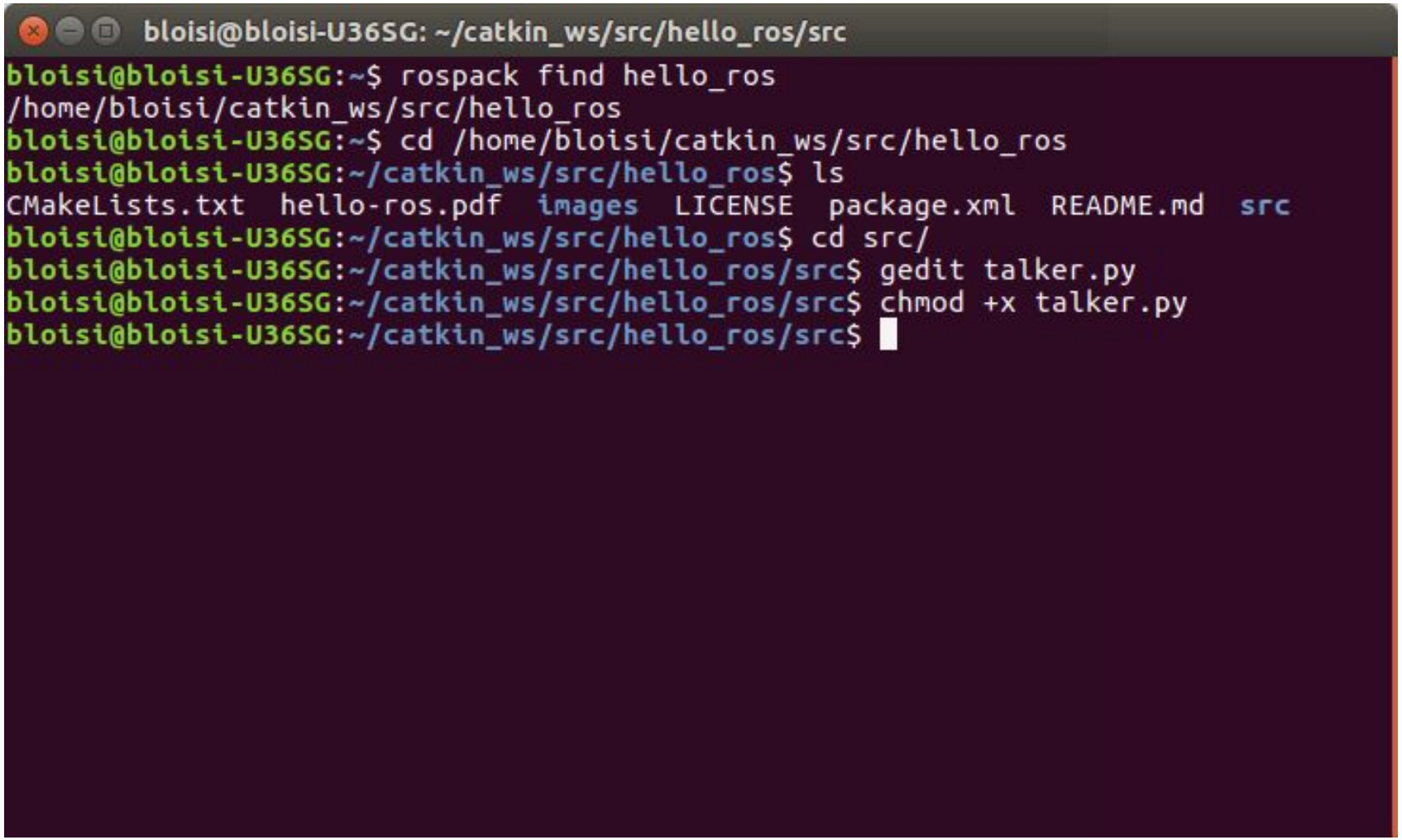
Open Save

```
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

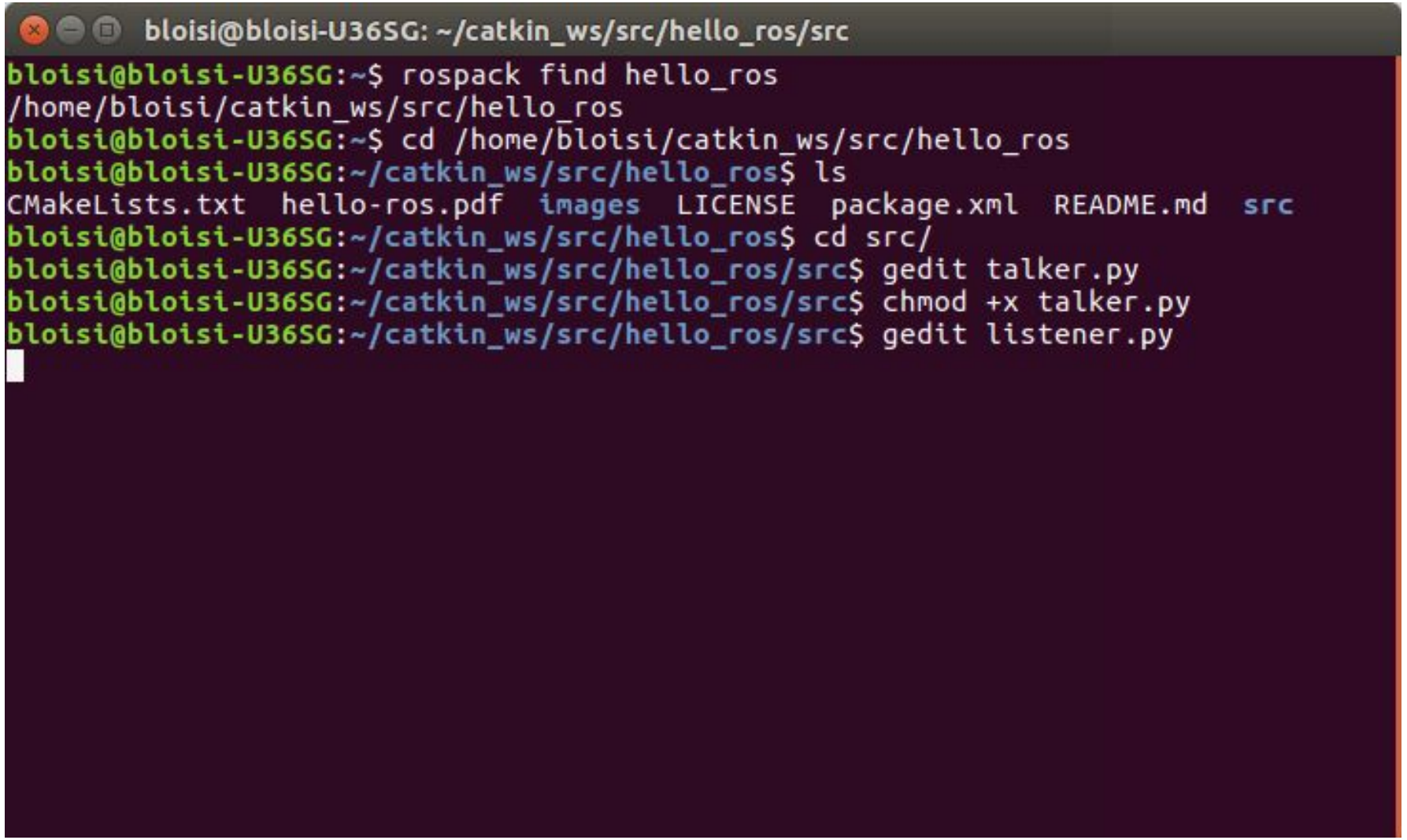
https://raw.github.com/ros/ros_tutorials/kinetic-devel/rospy_tutorials/001_talker_listener/talker.py

publisher (talker.py) eseguibile



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

Creiamo il subscriber (listener.py)



Codice del subscriber (listener.py)

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

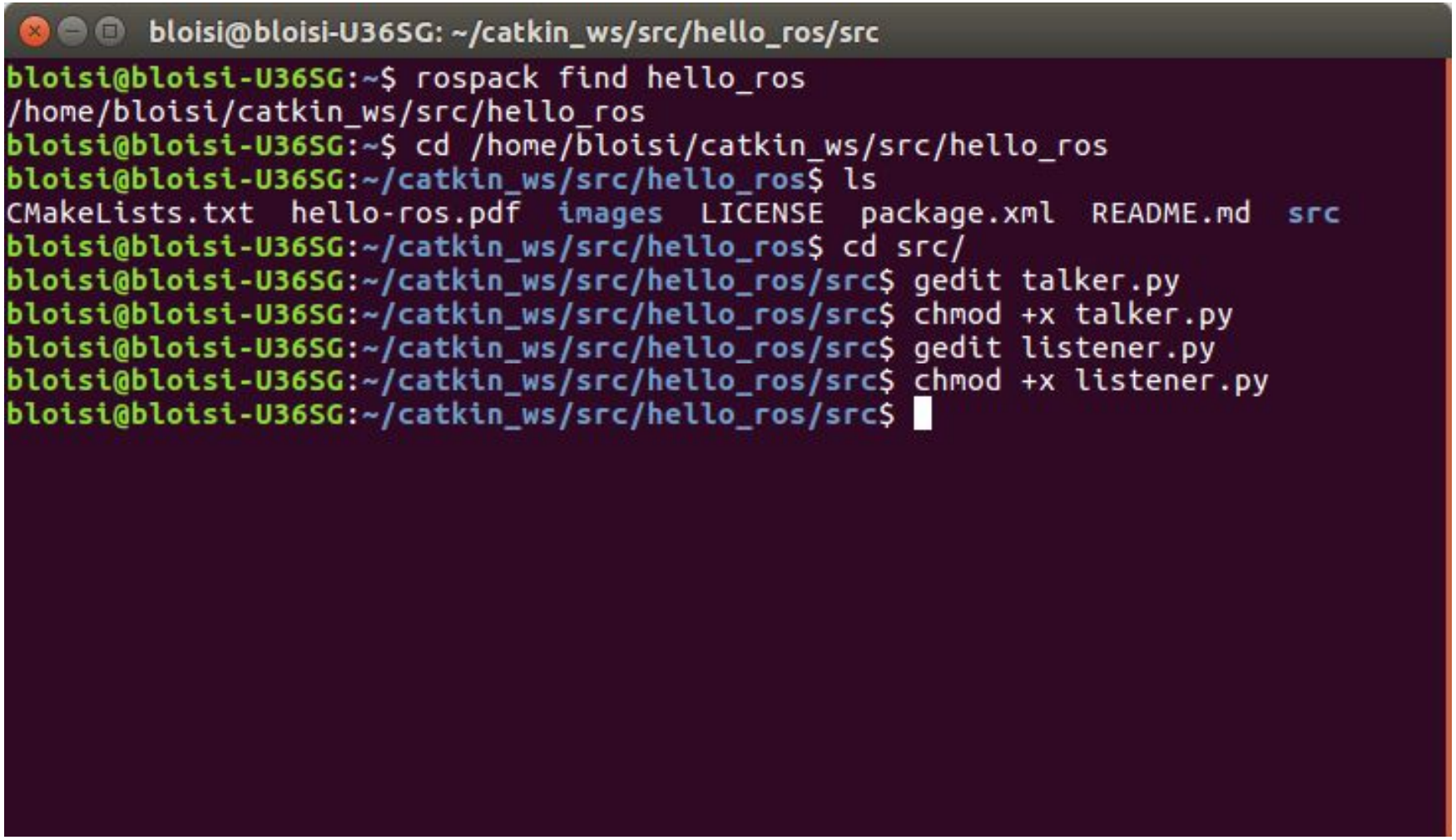
Open Save

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

Creiamo il subscriber (listener.py)



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

```
# )

##################################
## 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 ##
###########

```

CMake Tab Width: 8 Ln 205, Col 29 INS

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

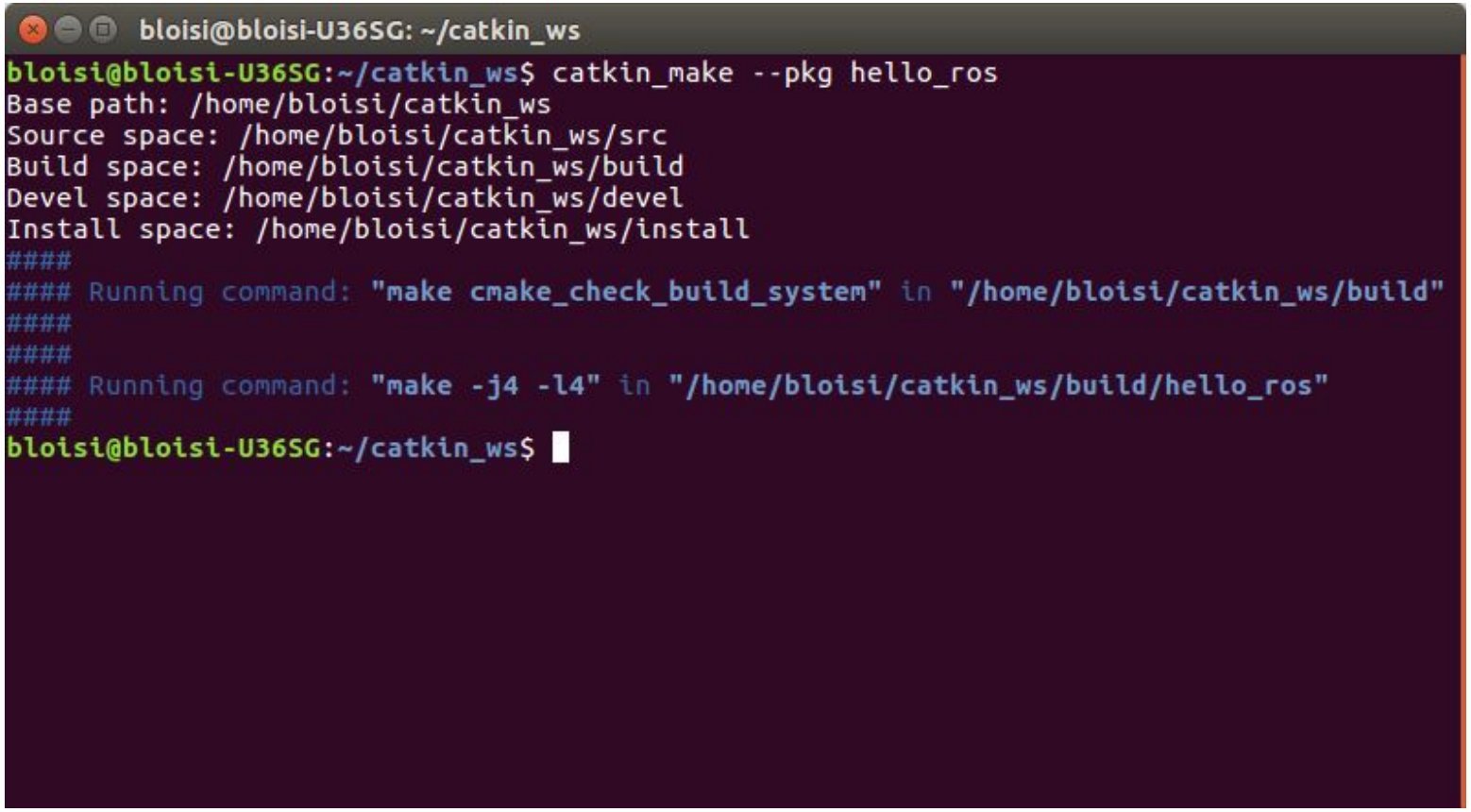
CMakeLists.txt

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

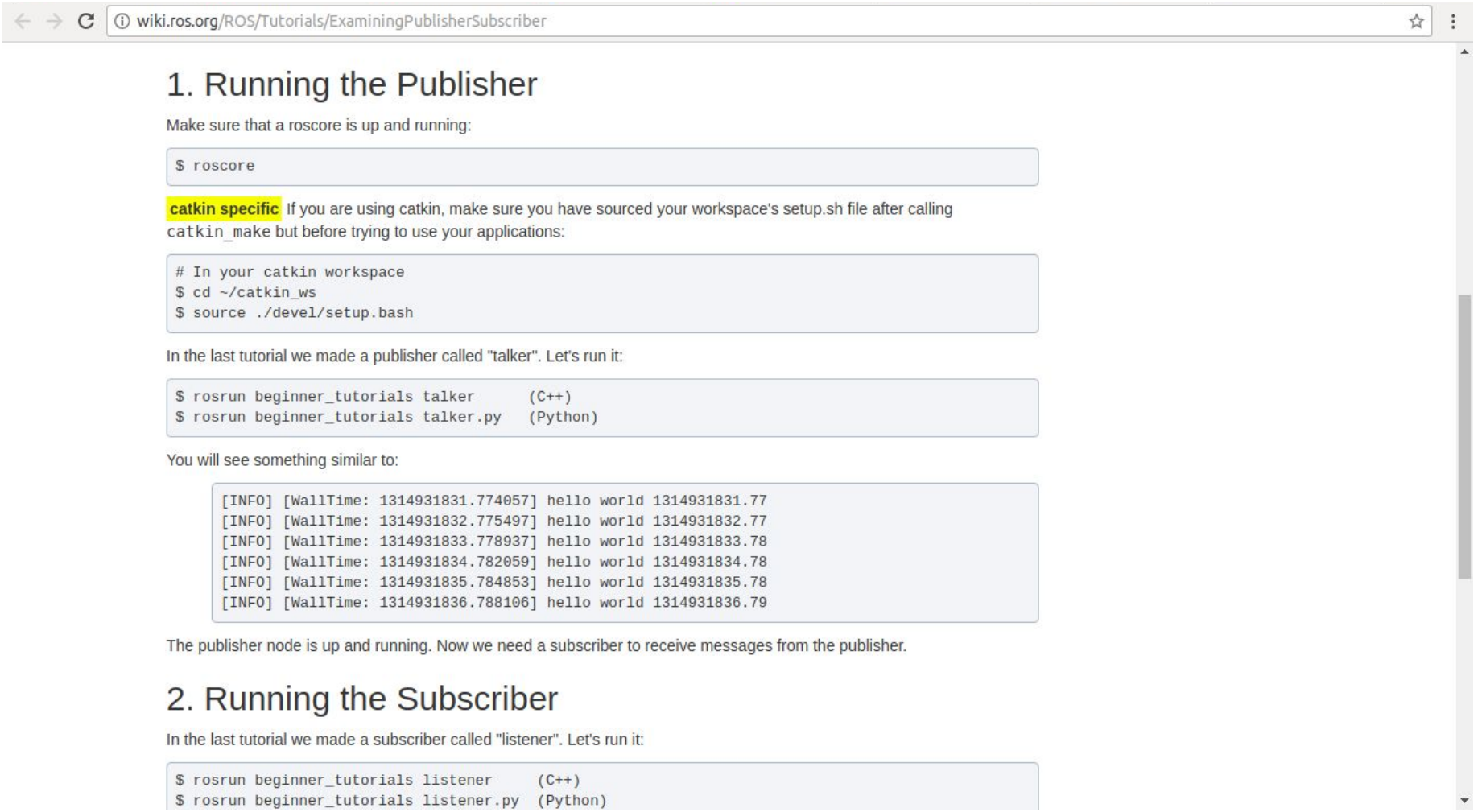
http://wiki.ros.org/ROS/Tutorials/Creating%20a%20Package%20by%20Hand

Compilazione con catkin_make

```
catkin_make --pkg hello_ros
```



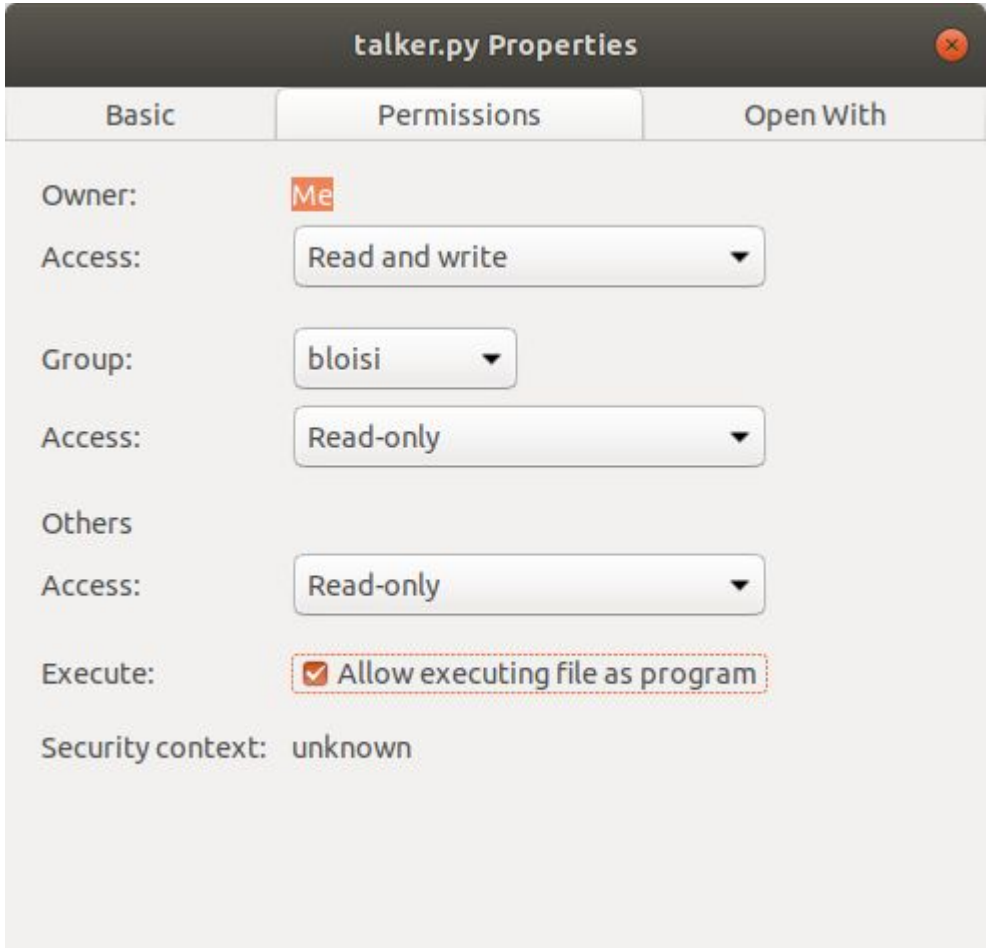
Esecuzione del nodo talker



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

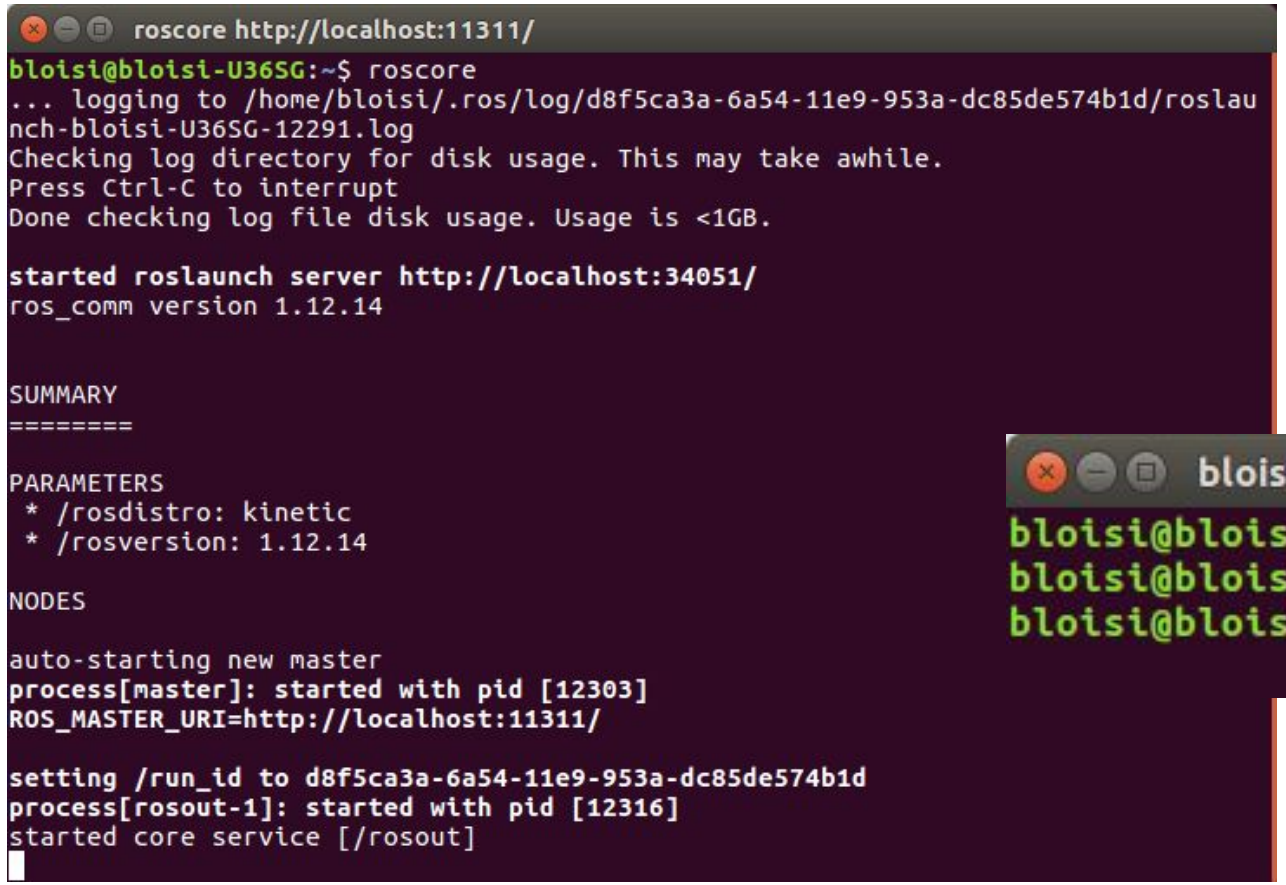
Esecuzione del nodo talker

Per mandare in esecuzione il file `talker.py` assicuriamoci di aver assegnato ad esso i permessi per poter essere eseguito dal sistema operativo



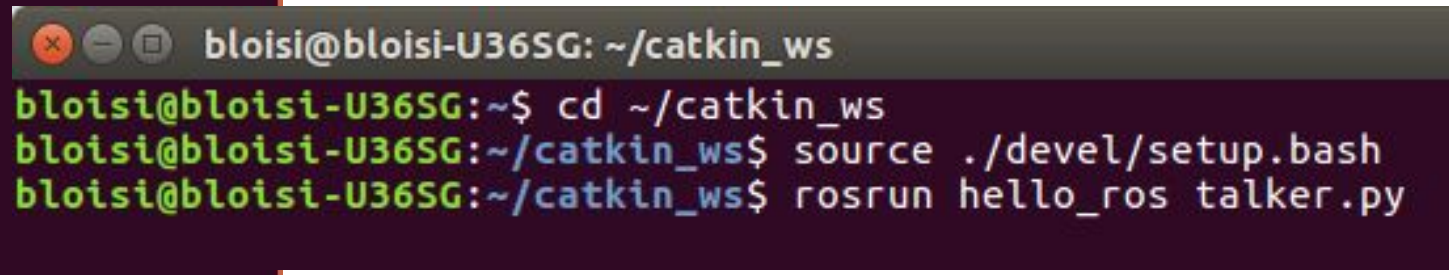
roscore + rosrun

Apriamo un terminale e lanciamo `roscore`



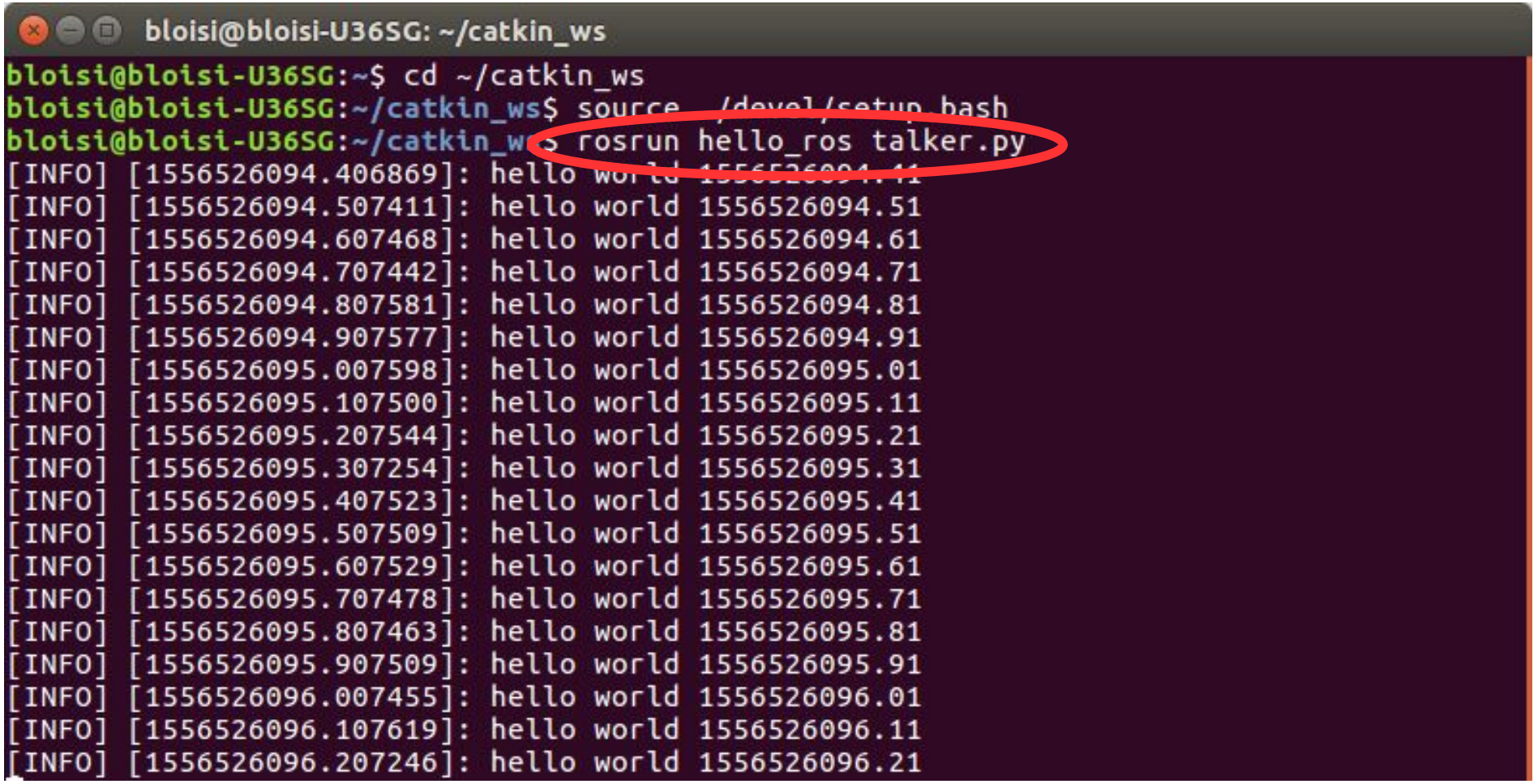
Apriamo un secondo terminale e lanciamo

`rosrun hello_ros talker.py`



Cosa accade?

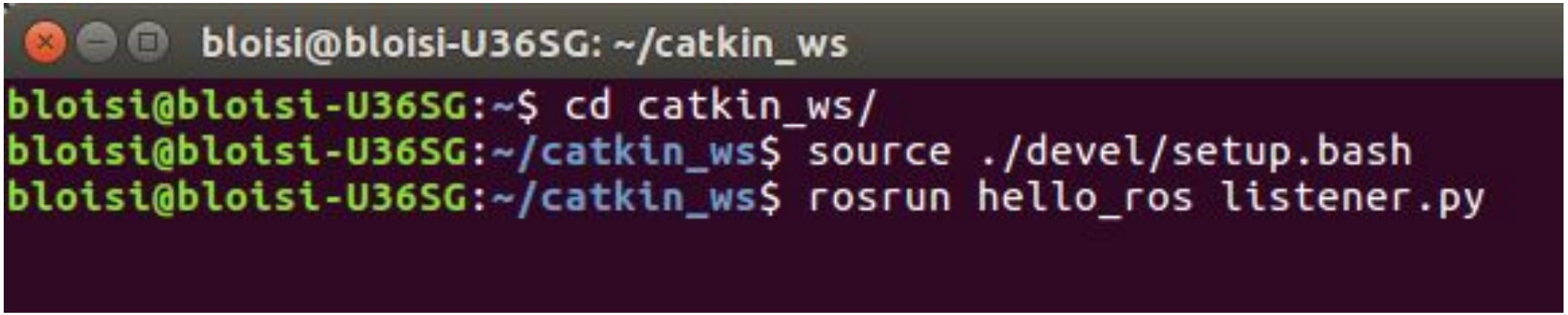
Esecuzione del nodo talker



Esecuzione del nodo listener

Apriamo un terzo terminale e lanciamo

`rosrun hello_ros listener.py`

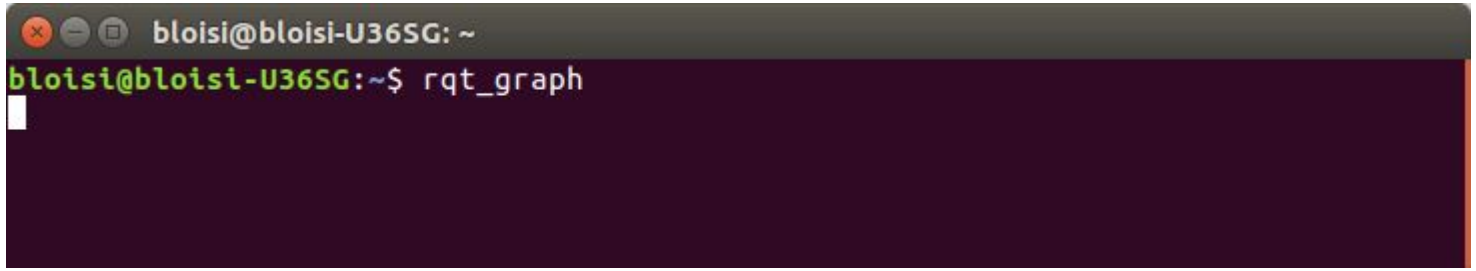


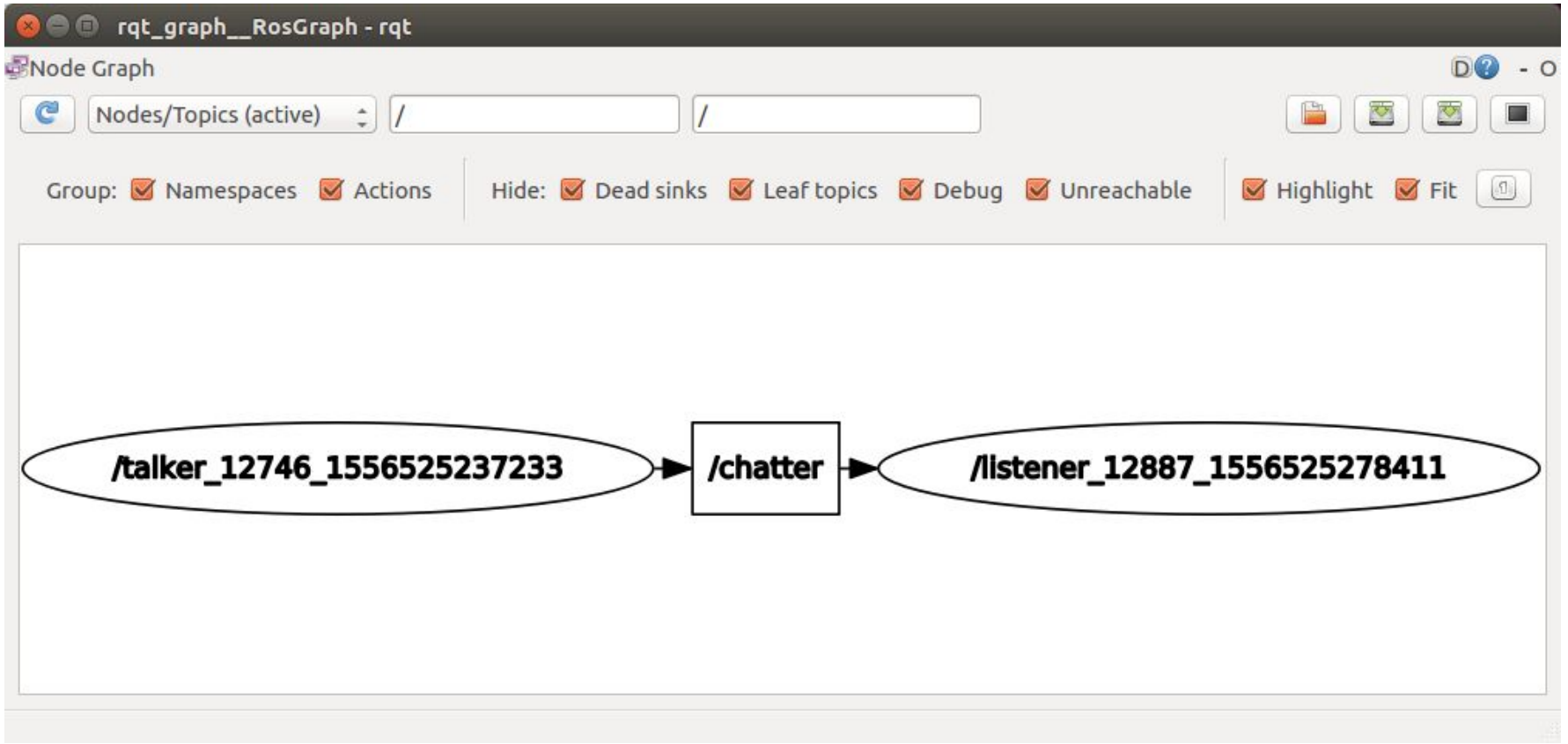
anche per `listener.py` assicuriamoci di aver assegnato i permessi per poter essere eseguito

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_15565263436301 heard hello world 1556526343.91
[INFO] [1556526344.009033]: /listener_13845_15565263436301 heard hello world 1556526344.01
[INFO] [1556526344.109351]: /listener_13845_15565263436301 heard hello world 1556526344.11
[INFO] [1556526344.209091]: /listener_13845_15565263436301 heard hello world 1556526344.21
[INFO] [1556526344.309455]: /listener_13845_15565263436301 heard hello world 1556526344.31
[INFO] [1556526344.409235]: /listener_13845_15565263436301 heard hello world 1556526344.41
[INFO] [1556526344.509644]: /listener_13845_15565263436301 heard hello world 1556526344.51
[INFO] [1556526344.609792]: /listener_13845_15565263436301 heard hello world 1556526344.61
[INFO] [1556526344.709825]: /listener_13845_15565263436301 heard hello world 1556526344.71
[INFO] [1556526344.809585]: /listener_13845_15565263436301 heard hello world 1556526344.81
[INFO] [1556526344.909382]: /listener_13845_15565263436301 heard hello world 1556526344.91
[INFO] [1556526345.009174]: /listener_13845_15565263436301 heard hello world 1556526345.01
[INFO] [1556526345.108972]: /listener_13845_15565263436301 heard hello world 1556526345.11
[INFO] [1556526345.208554]: /listener_13845_15565263436301 heard hello world 1556526345.21
[INFO] [1556526345.308504]: /listener_13845_15565263436301 heard hello world 1556526345.31
[INFO] [1556526345.408364]: /listener_13845_15565263436301 heard hello world 1556526345.41
[INFO] [1556526345.509007]: /listener_13845_15565263436301 heard hello world 1556526345.51
[INFO] [1556526345.608739]: /listener_13845_15565263436301 heard hello world 1556526345.61
[INFO] [1556526345.708979]: /listener_13845_15565263436301 heard hello world 1556526345.71
[INFO] [1556526345.809620]: /listener_13845_15565263436301 heard hello world 1556526345.81
[INFO] [1556526345.909187]: /listener_13845_15565263436301 heard hello world 1556526345.91
```

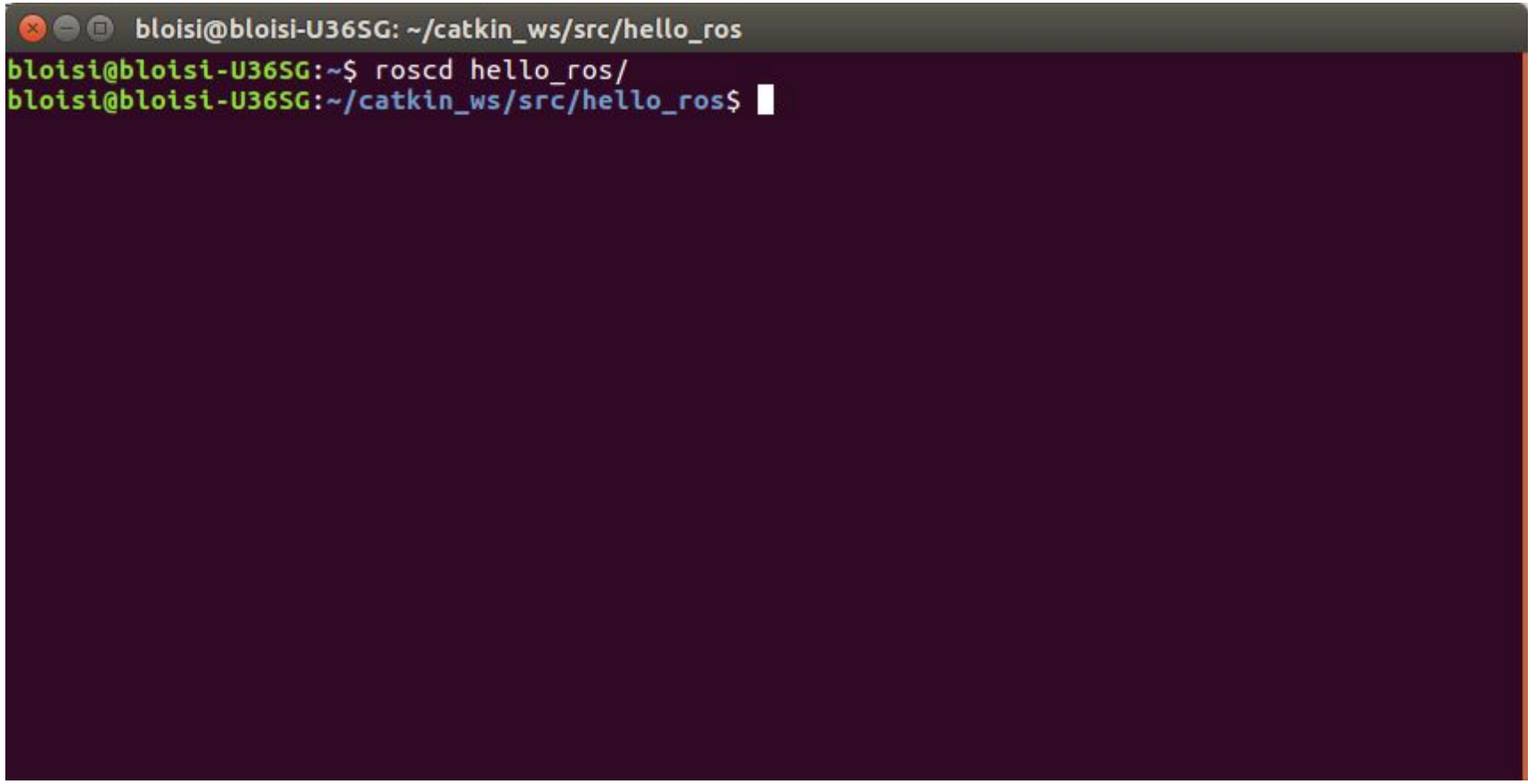
rqt_graph


bloisi@bloisi-U36SG: ~
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



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
```

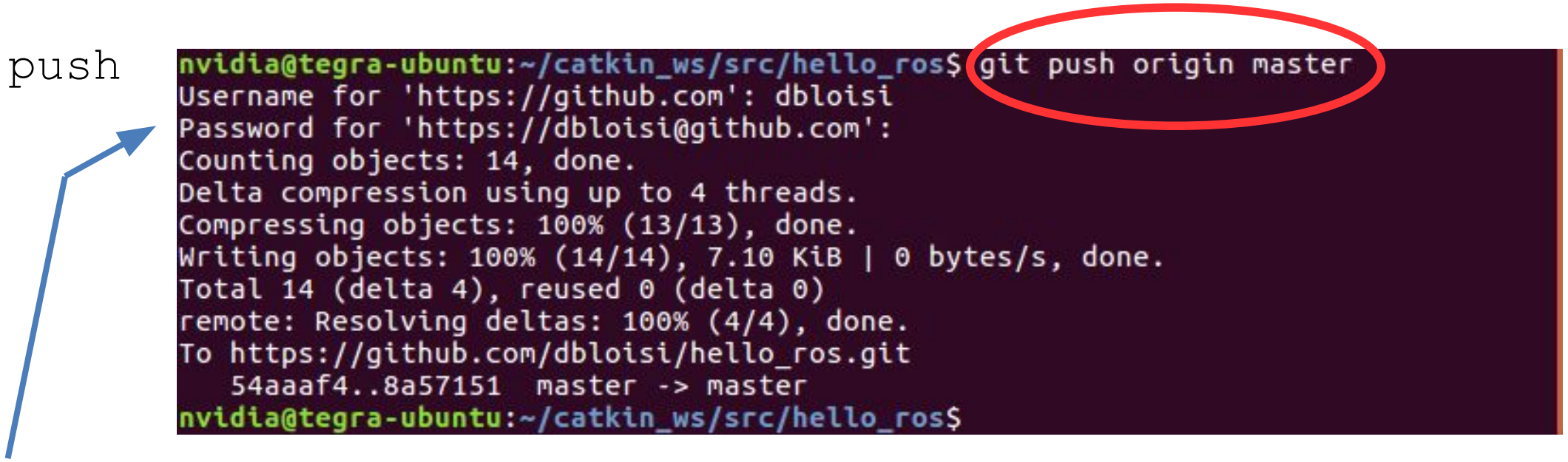
Aggiorniamo il repository locale (CMakeLists.txt)

`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
```

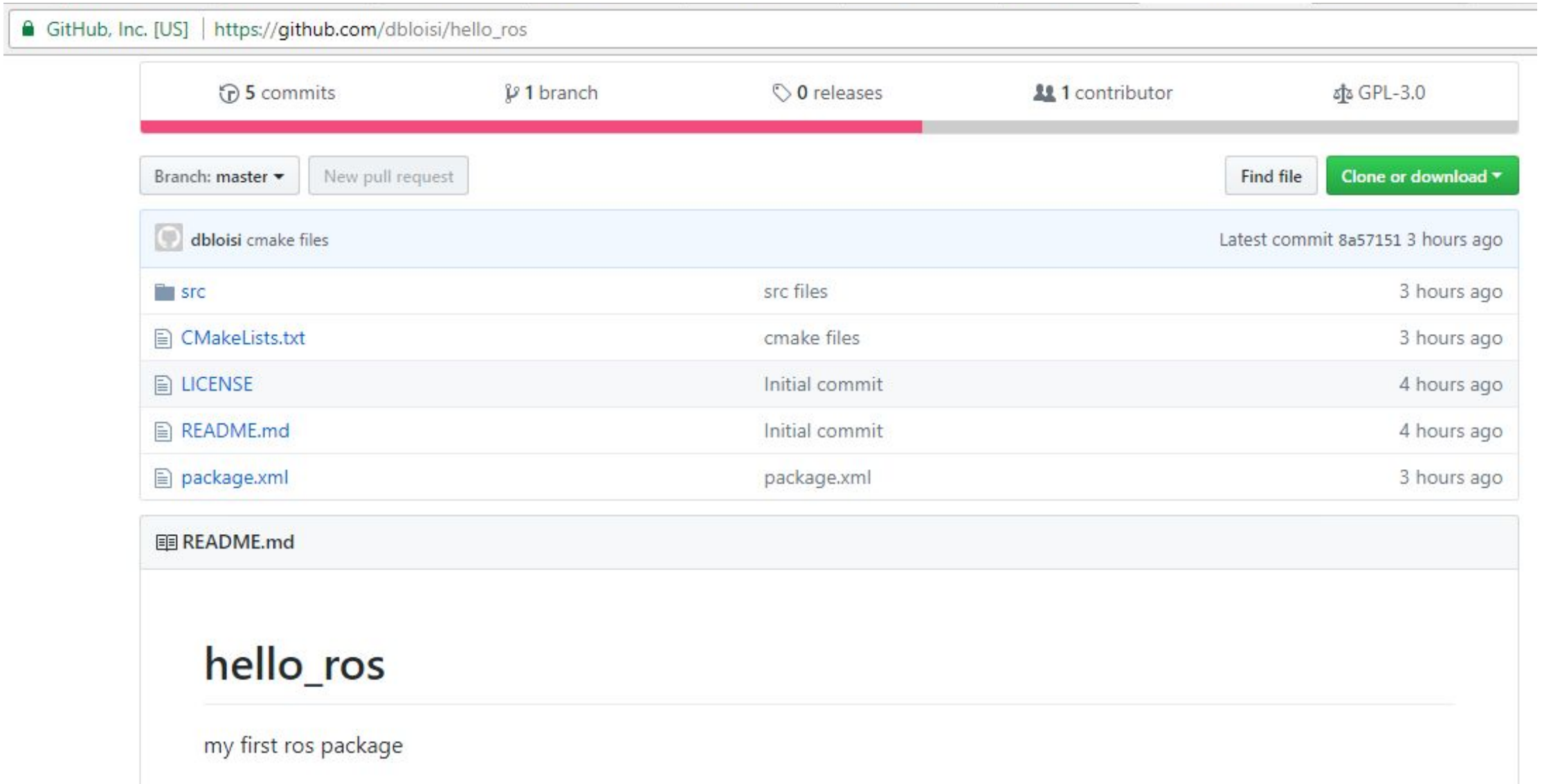
Aggiorniamo il repository remoto

`git push`



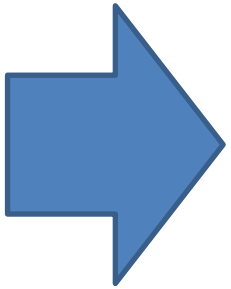
Verranno richieste le credenziali di accesso (username e password) per il server git

Aggiorniamo il repository remoto



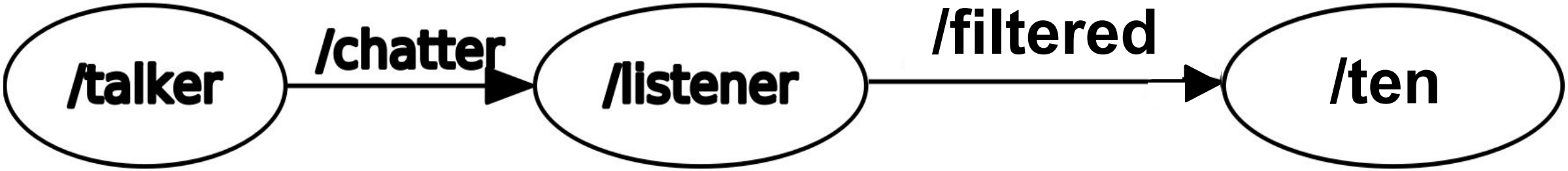
Esercitazione

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

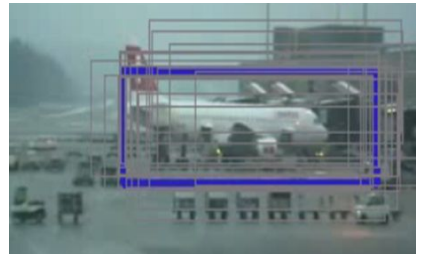


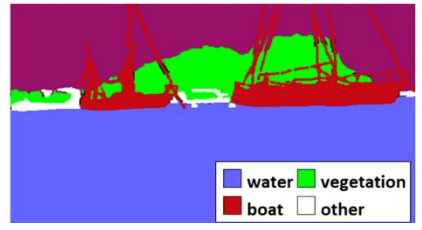
Corso di Visione e Percezione
A.A. 2019/2020

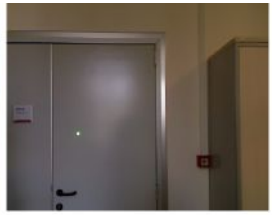
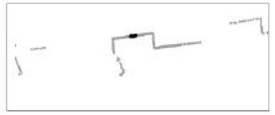
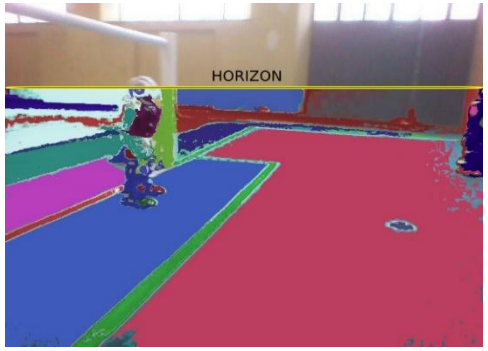
Docente
Domenico Daniele Bloisi

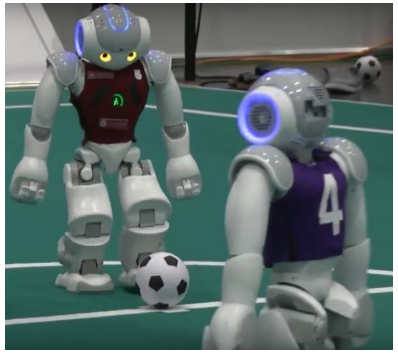
git +

ROS (Python)








Aprile 2020