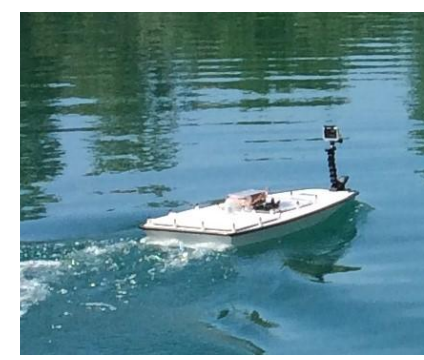




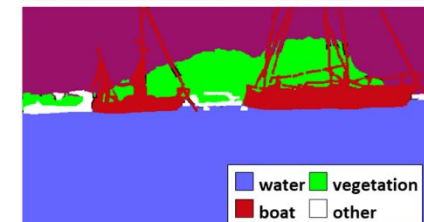
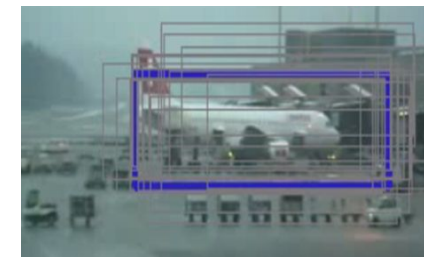
**UNIVERSITÀ DEGLI STUDI
DELLA BASILICATA**

Corso di Visione e Percezione



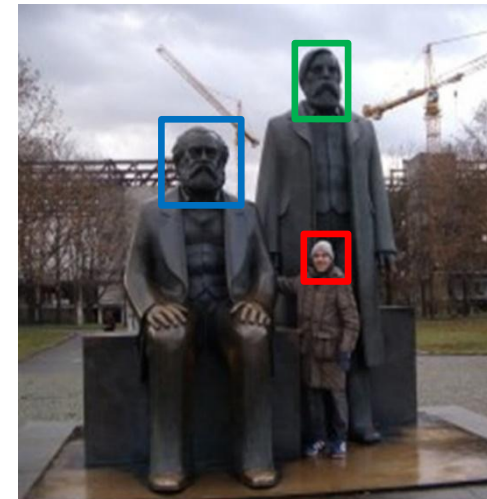
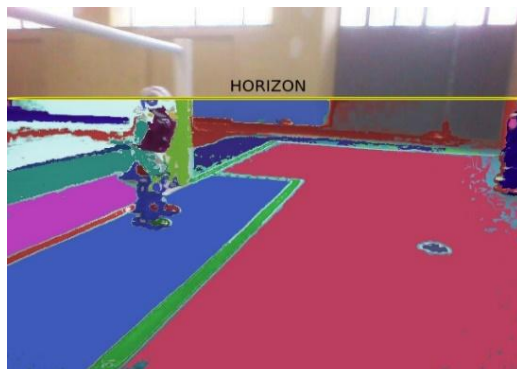
Docente

Domenico D. Bloisi



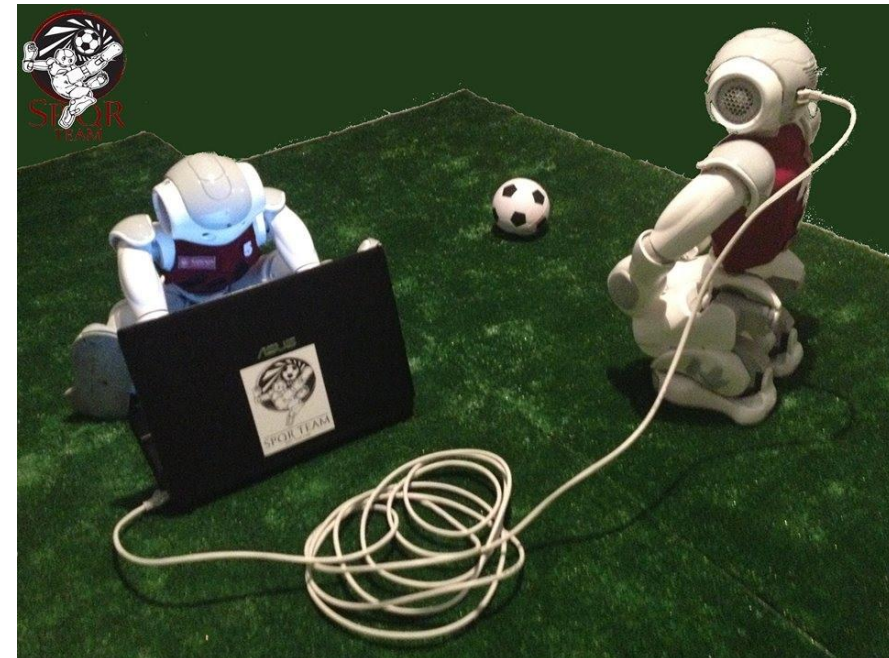
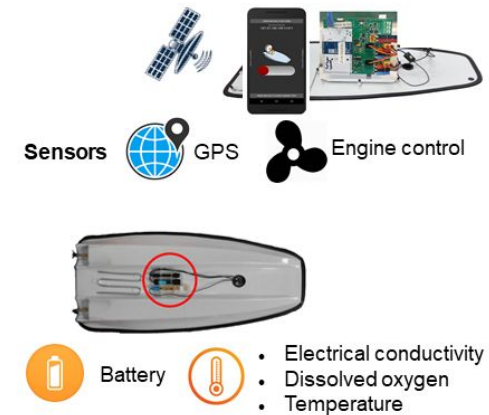
 **git** +

 **ROS (Python)**



Domenico Daniele Bloisi

- Ricercatore RTD B
Dipartimento di Matematica, Informatica ed Economia
Università degli studi della Basilicata
<http://web.unibas.it/bloisi>
- SPQR Robot Soccer Team
Dipartimento di Informatica, Automatica e Gestionale Università degli studi di Roma “La Sapienza”
<http://spqr.diag.uniroma1.it>



Informazioni sul corso

- Home page del corso
<http://web.unibas.it/bloisi/corsi/visione-e-percezione.html>
- Docente: Domenico Daniele Bloisi
- Periodo: **Il semestre** marzo 2021 – giugno 2021

Martedì 17:00-19:00 (Aula COPERNICO)

Mercoledì 8:30-10:30 (Aula COPERNICO)



Codice corso Google Classroom:
[https://classroom.google.com/c/
NjI2MjA4MzgzNDFa?cjc=xgolays](https://classroom.google.com/c/NjI2MjA4MzgzNDFa?cjc=xgolays)

Ricevimento

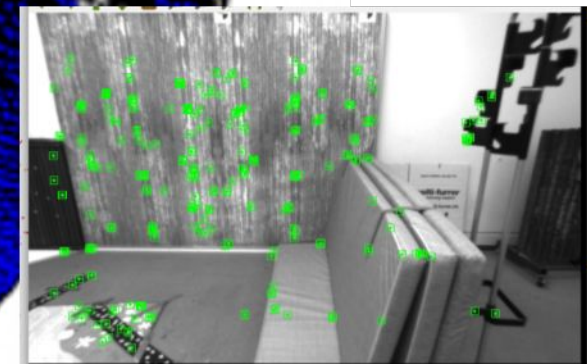
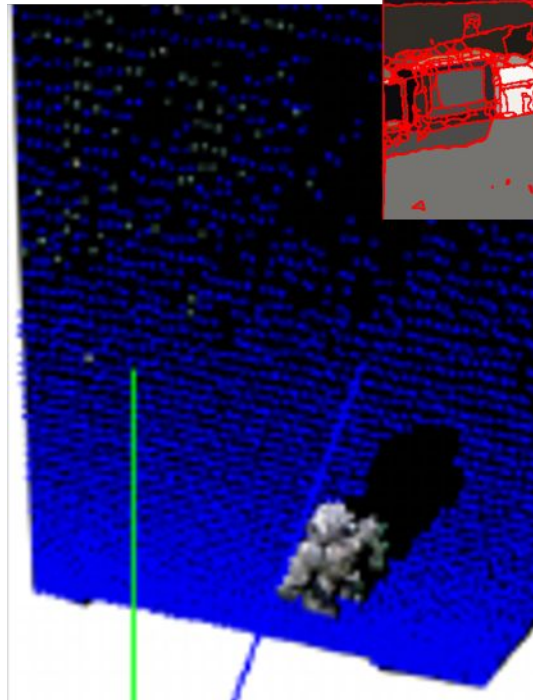
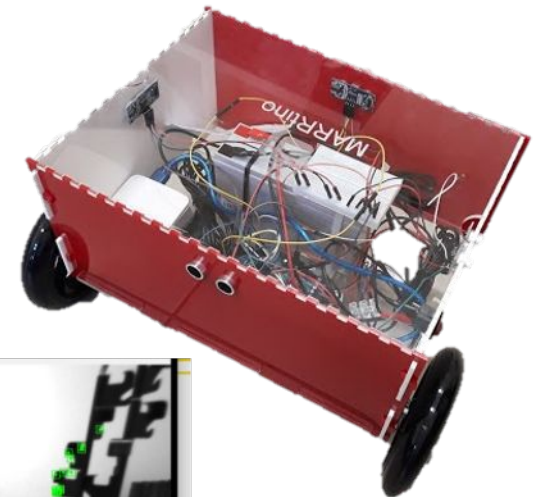
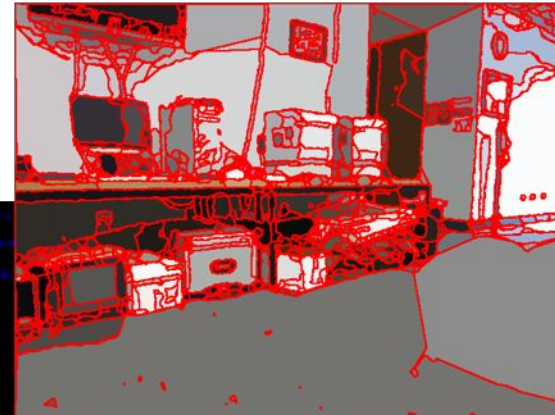
- Su appuntamento tramite Google Meet

Per prenotare un appuntamento inviare
una email a
domenico.bloisi@unibas.it



Programma – Visione e Percezione

- Introduzione al linguaggio Python
- Elaborazione delle immagini con Python
- Percezione 2D – OpenCV
- Introduzione al Deep Learning
- ROS
- Il paradigma publisher and subscriber
- Simulatori
- Percezione 3D - PCL



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

The screenshot shows the ROS.org website with the following elements:

- Browser address bar: `wiki.ros.org/catkin/Tutorials/create_a_workspace`
- ROS.org logo and navigation links: About | Support | Discussion Forum | Service Status | Q&A answers.ros.org
- Search bar with a "Submit" button.
- Dark navigation bar with links: Documentation, Browse Software, News, Download.
- Breadcrumbs: [catkin/](#) [Tutorials/](#) [create_a_workspace](#)
- Light blue callout box: "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."
- Section title:

Creating a workspace for catkin
- Description: "This tutorial covers how to setup a catkin workspace in which one or more catkin packages can be built."
- Keywords: catkin workspace
- Tutorial Level: BEGINNER
- Next Tutorial: [Creating catkin packages](#)
- OS distribution tabs: kinetic, melodic, **noetic** (selected), with a "Show EOL distros" checkbox.
- Index box: "Indice" with a link to "1. Prerequisites".
- Section title:

1. Prerequisites
- Text: "This tutorial assumes that you have [installed catkin](#) and sourced your environment. If you installed catkin via apt-get for

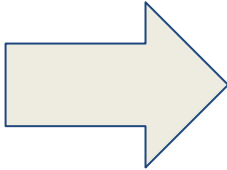
On the right side of the page, there is a sidebar with the following sections:

- ROS 2 Documentation: "The ROS Wiki is for ROS 1. Are you using ROS 2 (Dashing/Foxy/Rolling)? [Check out the ROS 2 Documentation](#)"
- Wiki navigation menu: Distributions, ROS/Installation, ROS/Tutorials, RecentChanges, **create_a_workspace** (highlighted).
- Pagina: "Pagina non alterabile", Informazioni, Allegati, Altre azioni: "Mostra il testo grezzo" (dropdown), "Esegui" button.
- Utente: "Accedi" button.

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

Compatibilità Python in Melodic

Nota per
Melodic
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 noetic, your command would look like this:

```
$ source /opt/ros/noetic/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 in ROS Melodic and earlier: 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
```

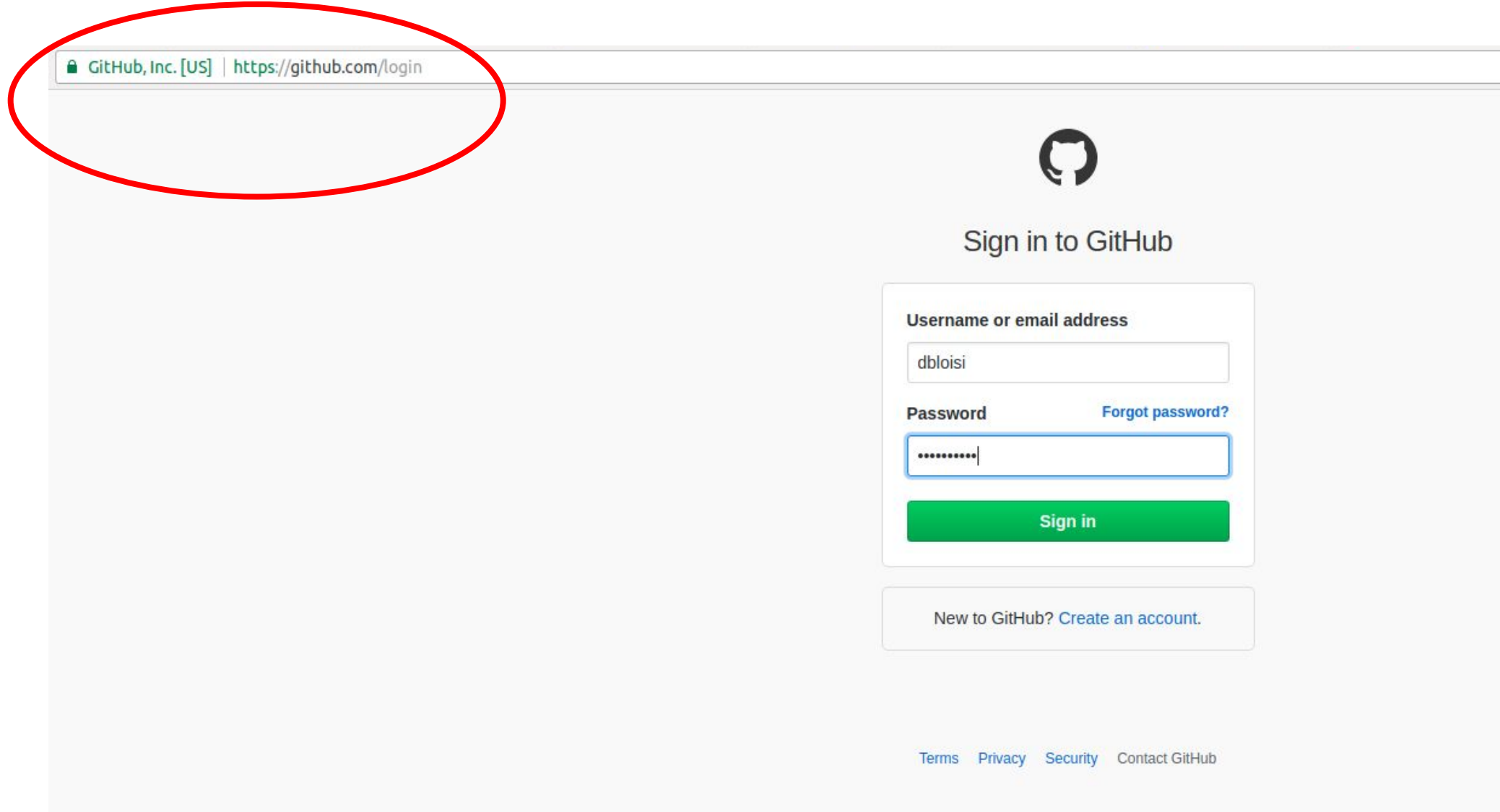
This will configure `catkin_make` with Python 3. You may then proceed to use just `catkin_make` for subsequent builds.

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.

Server git



The image shows a screenshot of the GitHub login page. The browser's address bar at the top left is circled in red and contains the text "GitHub, Inc. [US] | https://github.com/login". The main content area features the GitHub logo (a black octocat) centered at the top. Below the logo, the text "Sign in to GitHub" is displayed. The login form consists of two input fields: "Username or email address" with the value "dbloisi" and "Password" with masked characters ".....". A blue link "Forgot password?" 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 "New to GitHub? Create an account.". The footer of the page 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

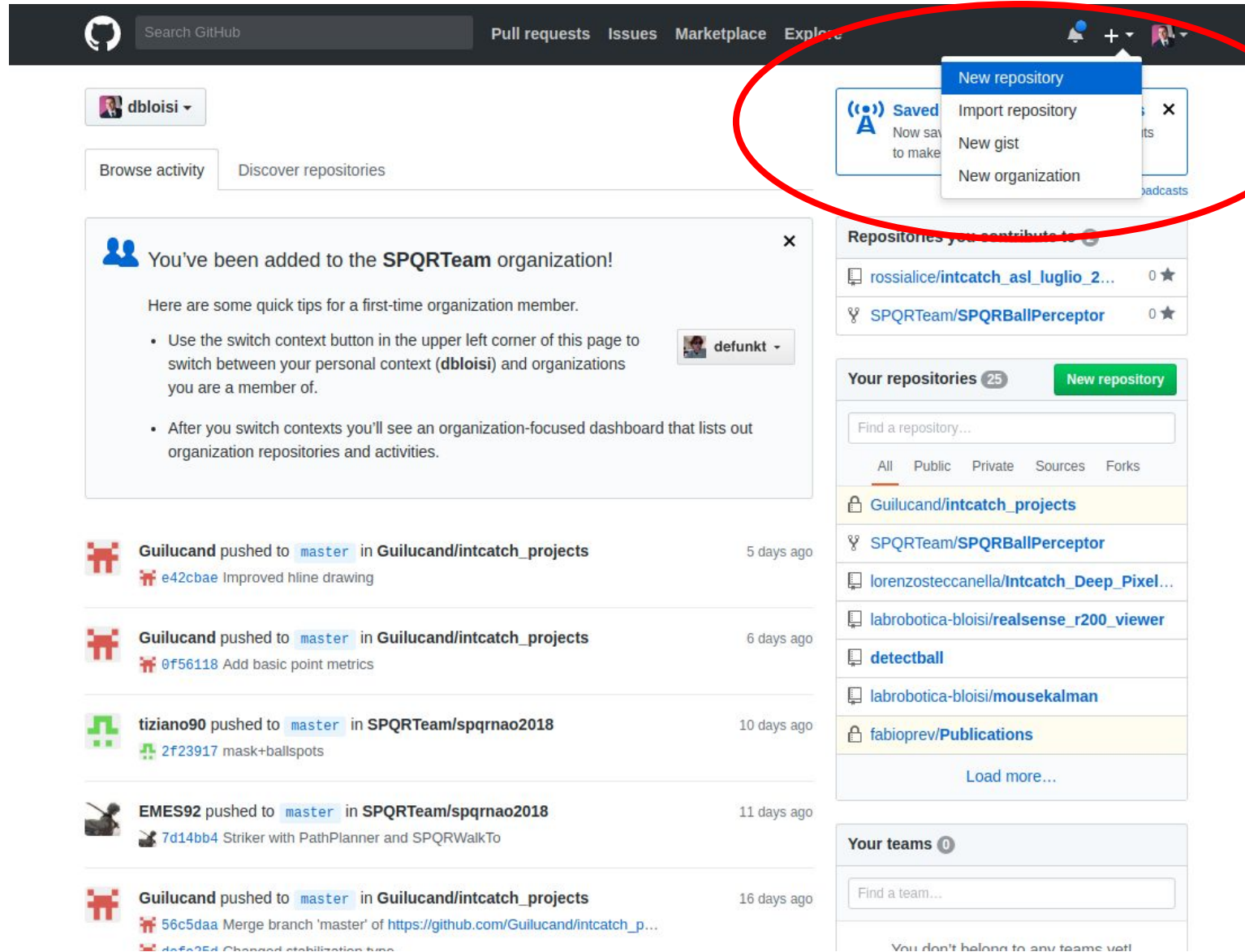
Password [Forgot password?](#)
.....

[Sign in](#)

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

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

Creare un repository git



The screenshot shows the GitHub user interface for user 'dbloisi'. The top navigation bar includes 'Search GitHub', 'Pull requests', 'Issues', 'Marketplace', and 'Explore'. A dropdown menu is open, showing options: 'New repository', 'Import repository', 'New gist', and 'New organization'. This menu is circled in red. Below the navigation bar, there are tabs for 'Browse activity' and 'Discover repositories'. A notification box states: '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.' Below the notification, a list of repository activity is shown, including pushes to 'master' in 'Guilucand/intcatch_projects' and 'SPQRTeam/spqrnao2018'. On the right side, there are sections for 'Repositories you contribute to', 'Your repositories' (25), and 'Your teams' (0).

dbloisi

Search GitHub

Pull requests Issues Marketplace Explore

Browse activity Discover repositories

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...

4a625d4 Changed stabilization func...

Repositories you contribute to

rossialice/intcatch_asl_luglio_2... 0 stars

SPQRTeam/SPQRBallPerceptor 0 stars

Your repositories 25 **New repository**

Find a repository...

All Public Private Sources Forks

Guilucand/intcatch_projects

SPQRTeam/SPQRBallPerceptor

lorenzosteccanella/Intcatch_Deep_Pixel...

labrobotica-bloisi/realsense_r200_viewer

detectball

labrobotica-bloisi/mousekalman

fabioprev/Publications

Load more...

Your teams 0

Find a team...

You don't belong to any teams yet!

Repository name

Create a new repository

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

Owner



Repository name

hello_ros ✓

1

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

2

Description (optional)

my first ros package

3



Public

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

Private

You choose who can see and commit to this repository.

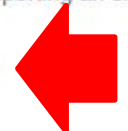
4

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



5

Create repository

Repository create

The screenshot shows a GitHub repository page for 'dbloisi / hello_ros'. The repository name is circled in red. Below the repository name, there are navigation tabs 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, there are buttons for 'Branch: master', 'new pull request', 'Create new file', 'Upload files', 'Find file', and 'Clone or download'. The commit history table is circled in red, showing an initial commit by 'dbloisi' with files 'LICENSE' and 'README.md'. Below the table, the 'README.md' content 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	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 navigation tabs for 'Code', 'Issues', 'Pull requests', 'Projects', 'Wiki', 'Insights', and 'Settings'. Below these, the repository name 'my first ros package' is displayed with an 'Edit' button. A summary bar shows '1 commit', '1 branch', '0 releases', and '1 contributor'. A row of buttons includes 'Branch: master', 'New pull request', 'Create new file', 'Upload files', 'Find file', and 'Clone or download'. The 'Clone or download' button is highlighted with a red circle, and its dropdown menu is open, showing options for cloning with HTTPS, SSH, and downloading a ZIP file. The HTTPS URL 'https://github.com/dbloisi/hello_ros.' is visible in the dropdown. Below the dropdown, the repository's file structure is shown, including 'LICENSE' and 'README.md' files, and a 'README.md' section with the repository name 'hello_ros' and 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

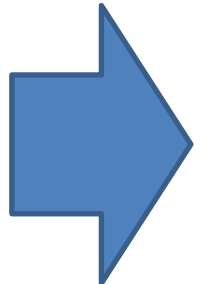
Creazione del repository locale

Il repository remoto si trova in

https://github.com/dbloisi/hello_ros

Spostiamoci nella cartella src del nostro workspace ROS che si trova in `~/catkin_ws/src`

```
bloisi@bloisi-U36SG: ~/catkin_ws/src
File Edit View Search Terminal Help
bloisi@bloisi-U36SG:~$ cd catkin_ws/
bloisi@bloisi-U36SG:~/catkin_ws$ ls
build  devel  src
bloisi@bloisi-U36SG:~/catkin_ws$ cd src/
bloisi@bloisi-U36SG:~/catkin_ws/src$
```



Clone del repository remoto

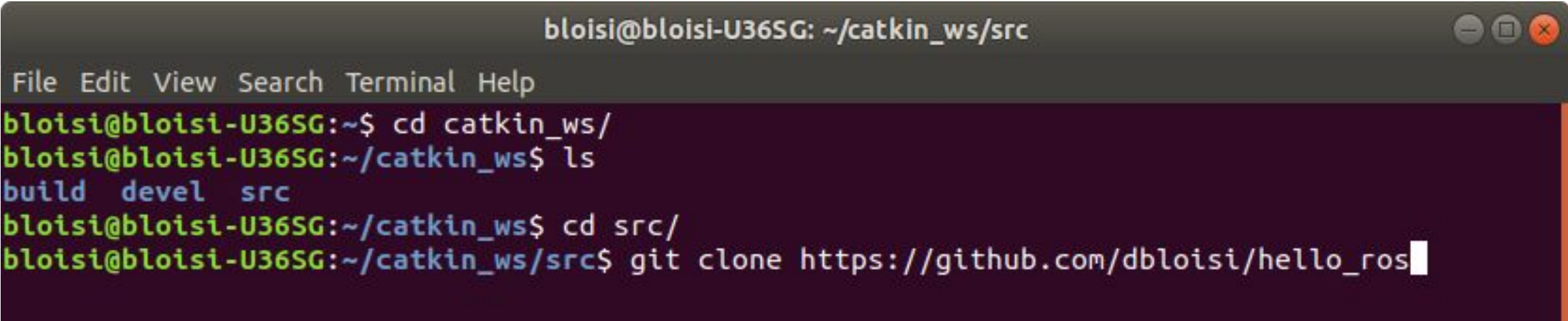
1. Cloniamo il repository remoto che si trova in

https://github.com/dbloisi/hello_ros

con il comando

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

2. Verrà creato un repository locale in [~/catkin_ws/src/hello_ros](#)



```
bloisi@bloisi-U36SG: ~/catkin_ws/src
File Edit View Search Terminal Help
bloisi@bloisi-U36SG:~$ cd catkin_ws/
bloisi@bloisi-U36SG:~/catkin_ws$ ls
build  devel  src
bloisi@bloisi-U36SG:~/catkin_ws$ cd src/
bloisi@bloisi-U36SG:~/catkin_ws/src$ git clone https://github.com/dbloisi/hello_ros
```

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 [roscatkin](#) or [catkin](#) to create a new package, and [roscatkin](#) to list package dependencies.

Tutorial Level: BEGINNER

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

catkin rosbuilt

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 [ ] 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>

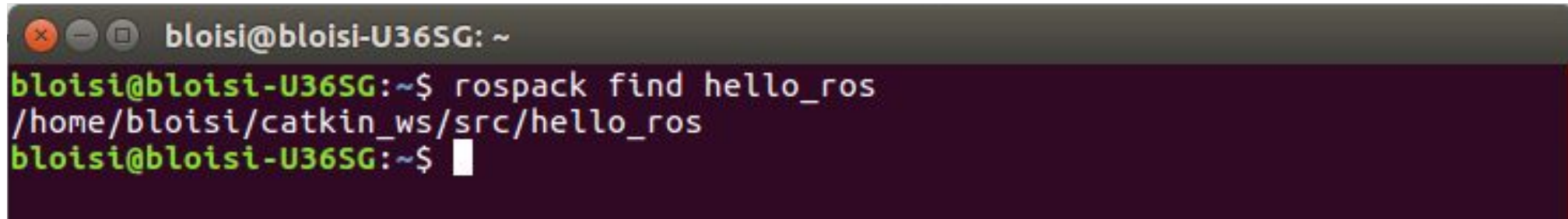
<!-- The export tag contains other, unspecified, tags -->
<export>
  <!-- Other tools can request additional information be placed here -->

</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 text. The window title is 'bloisi@bloisi-U36SG: ~'. The prompt is 'bloisi@bloisi-U36SG:~\$'. The command 'rospack find hello_ros' is entered. 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>

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

The screenshot shows the ROS.org website. The browser address bar displays the URL: <http://wiki.ros.org/ROS/Tutorials/WritingPublisherSubscriber%28python%29>. The page header features the ROS.org logo and navigation links: About | Support | Discussion Forum | Service Status | Q&A answers.ros.org. A search box with a 'Submit' button is also present. Below the header is a dark blue navigation bar with links for Documentation, Browse Software, News, and Download. The main content area is titled 'ROS/ Tutorials/ WritingPublisherSubscriber(python)'. A note states: 'Note: This tutorial assumes that you have completed the previous tutorials: creating a ROS msg and srv.' Below this, a message asks users to post questions on answers.ros.org. The main heading is 'Writing a Simple Publisher and Subscriber (Python)'. A description follows: 'Description: This tutorial covers how to write a publisher and subscriber node in python.' The tutorial level is 'BEGINNER'. The next tutorial is 'Examining the simple publisher and subscriber'. There are two tabs: 'catkin' (selected) and 'roscpp'. A 'Contents' box lists the following sections: 1. Writing the Publisher Node (1. The Code, 2. The Code Explained), 2. Writing the Subscriber Node (1. The Code, 2. The Code Explained). The right sidebar contains a 'Wiki' section with links to Distributions, ROS/Installation, ROS/Tutorials, RecentChanges, and WritingPubl...ber(python) (highlighted). Below this is a 'Page' section with links to Immutable Page, Info, Attachments, and a 'More Actions' dropdown. At the bottom of the sidebar is a 'User' section with a 'Login' link.

<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 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
```

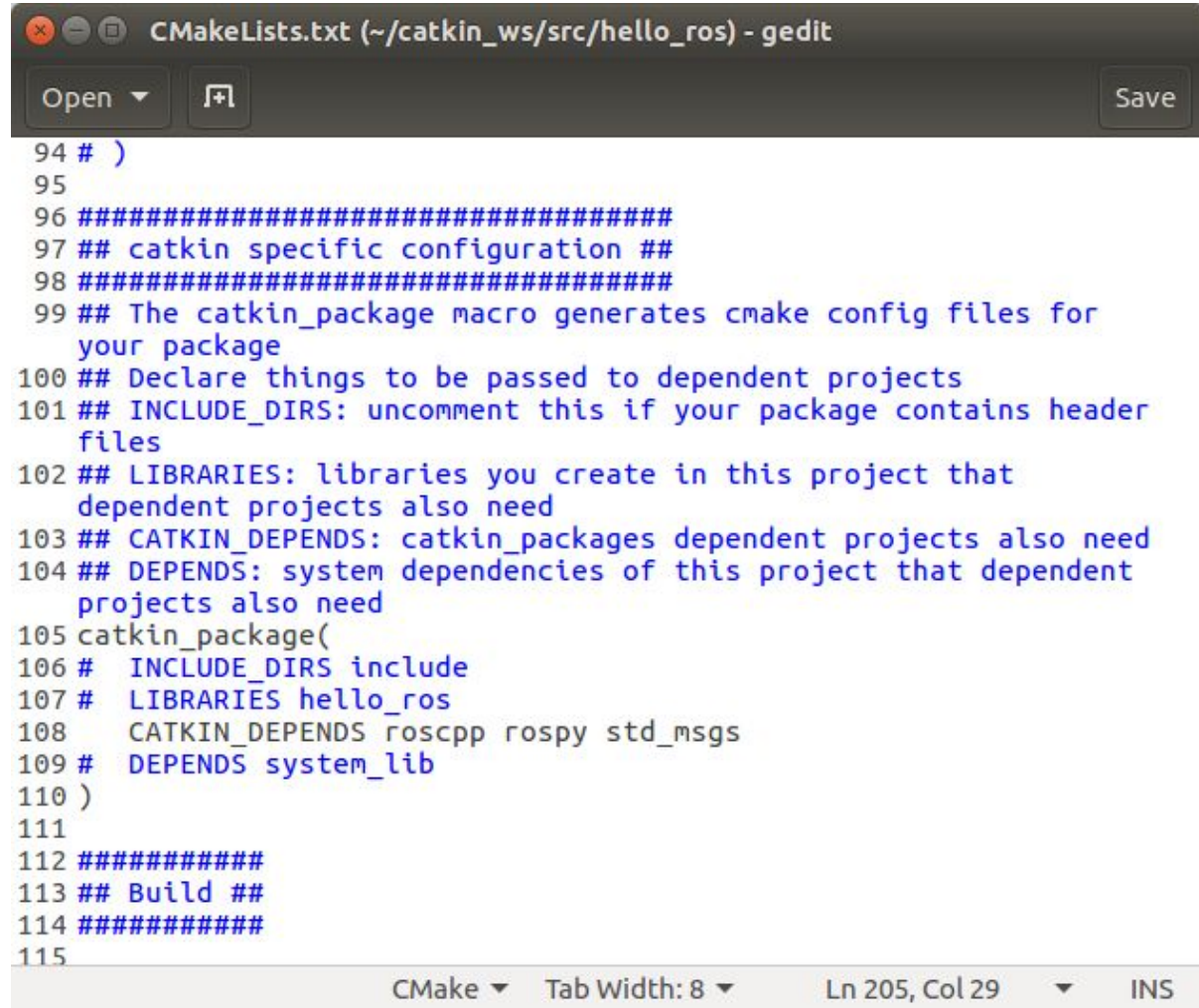
https://raw.githubusercontent.com/ros/ros_tutorials/kinetic-devel/rospy_tutorials/001_talker_listener/listener.py

Creiamo il subscriber (listener.py)

```
bloisi@bloisi-U365G: ~/catkin_ws/src/hello_ros/src
bloisi@bloisi-U365G:~$ rospack find hello_ros
/home/bloisi/catkin_ws/src/hello_ros
bloisi@bloisi-U365G:~$ cd /home/bloisi/catkin_ws/src/hello_ros
bloisi@bloisi-U365G:~/catkin_ws/src/hello_ros$ ls
CMakeLists.txt  hello-ros.pdf  images  LICENSE  package.xml  README.md  src
bloisi@bloisi-U365G:~/catkin_ws/src/hello_ros$ cd src/
bloisi@bloisi-U365G:~/catkin_ws/src/hello_ros/src$ gedit talker.py
bloisi@bloisi-U365G:~/catkin_ws/src/hello_ros/src$ chmod +x talker.py
bloisi@bloisi-U365G:~/catkin_ws/src/hello_ros/src$ gedit listener.py
bloisi@bloisi-U365G:~/catkin_ws/src/hello_ros/src$ chmod +x listener.py
bloisi@bloisi-U365G:~/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 [icon] 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
```

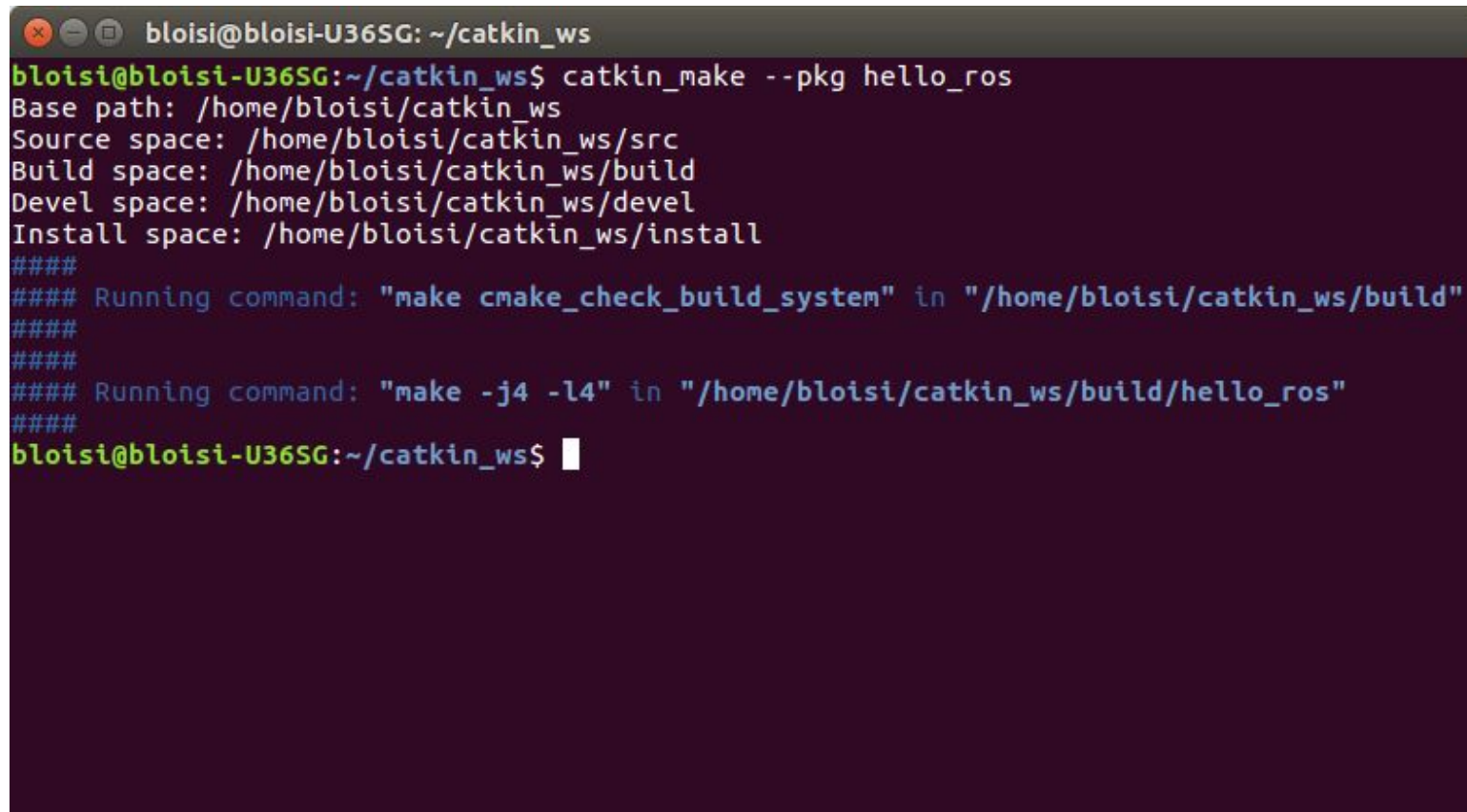
CMakeLists.txt

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



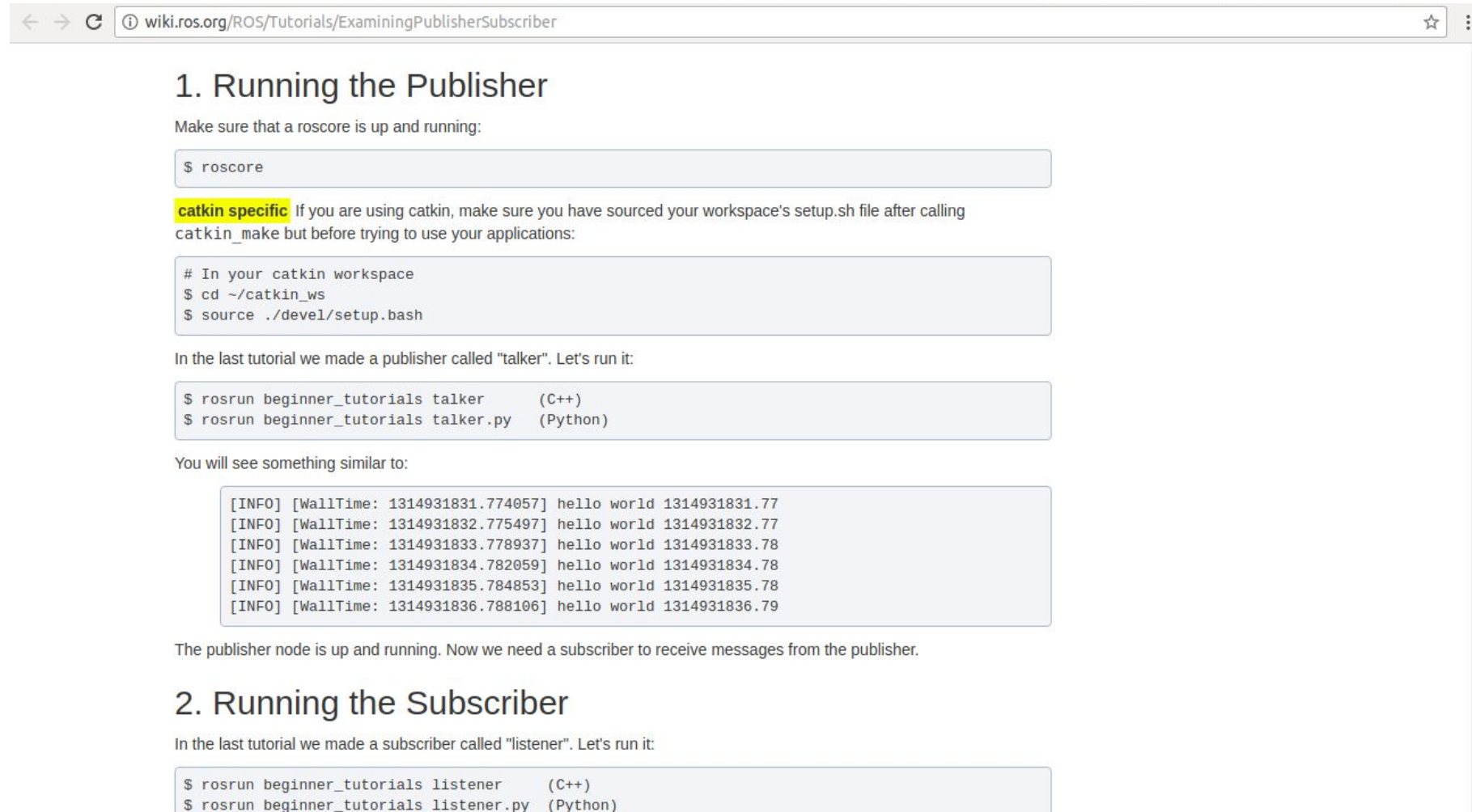
Compilazione con catkin_make

```
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 workspace configuration: "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", and "Install space: /home/bloisi/catkin_ws/install". It then shows two "####" separator lines, followed by "#### Running command: 'make cmake_check_build_system' in '/home/bloisi/catkin_ws/build'", another "####" separator line, and "#### Running command: 'make -j4 -l4' in '/home/bloisi/catkin_ws/build/hello_ros'", and a final "####" separator line. The prompt "bloisi@bloisi-U36SG:~/catkin_ws\$" is visible at the bottom with a white cursor.

```
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 containing '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:

```
$ rosruntime beginner_tutorials talker      (C++)
$ rosruntime 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.

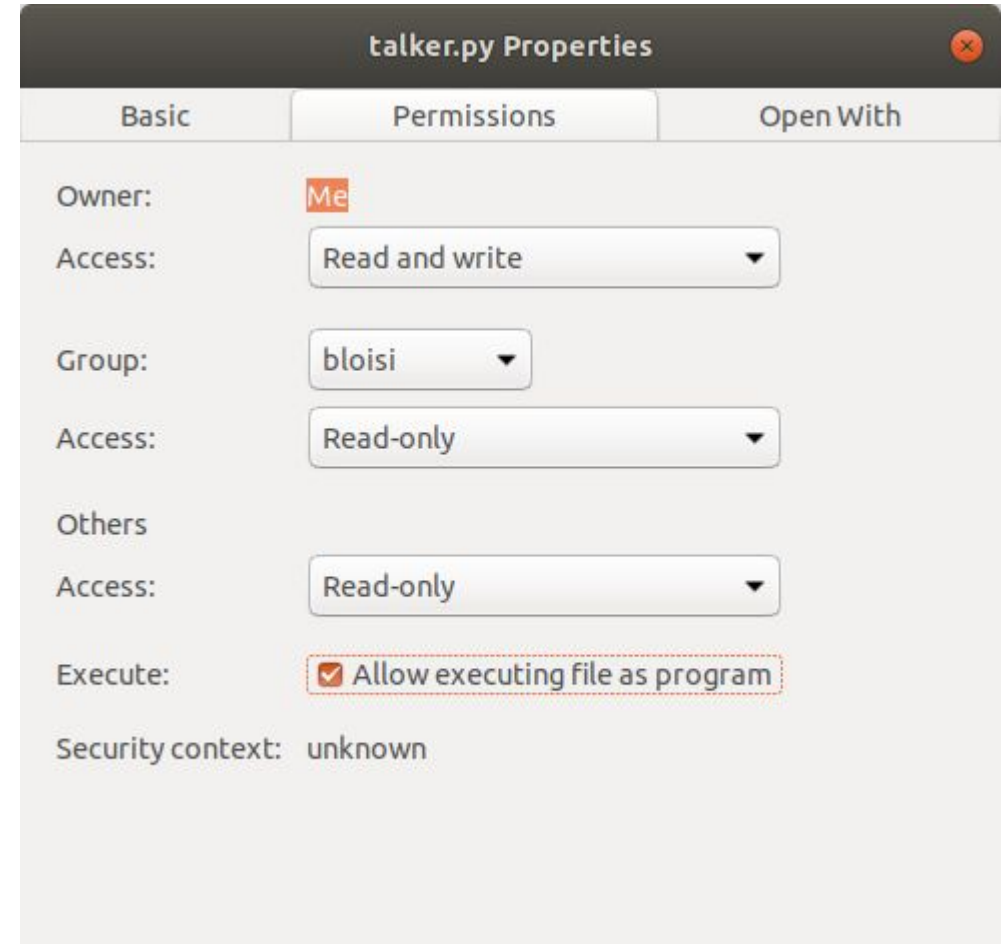
2. Running the Subscriber

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

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

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`

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

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?

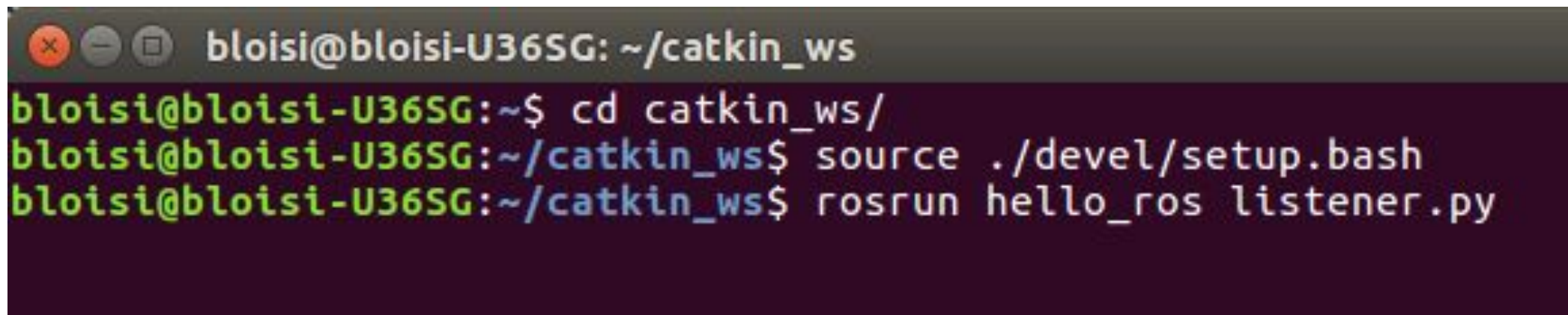
Esecuzione del nodo talker

```
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 terminal window with a dark background and light text. The window title is 'bloisi@bloisi-U365G: ~/catkin_ws'. The terminal shows three lines of commands: 'cd catkin_ws/', 'source ./devel/setup.bash', and 'roslaunch hello_ros listener.py'. The prompt is 'bloisi@bloisi-U365G:~/catkin_ws\$'.

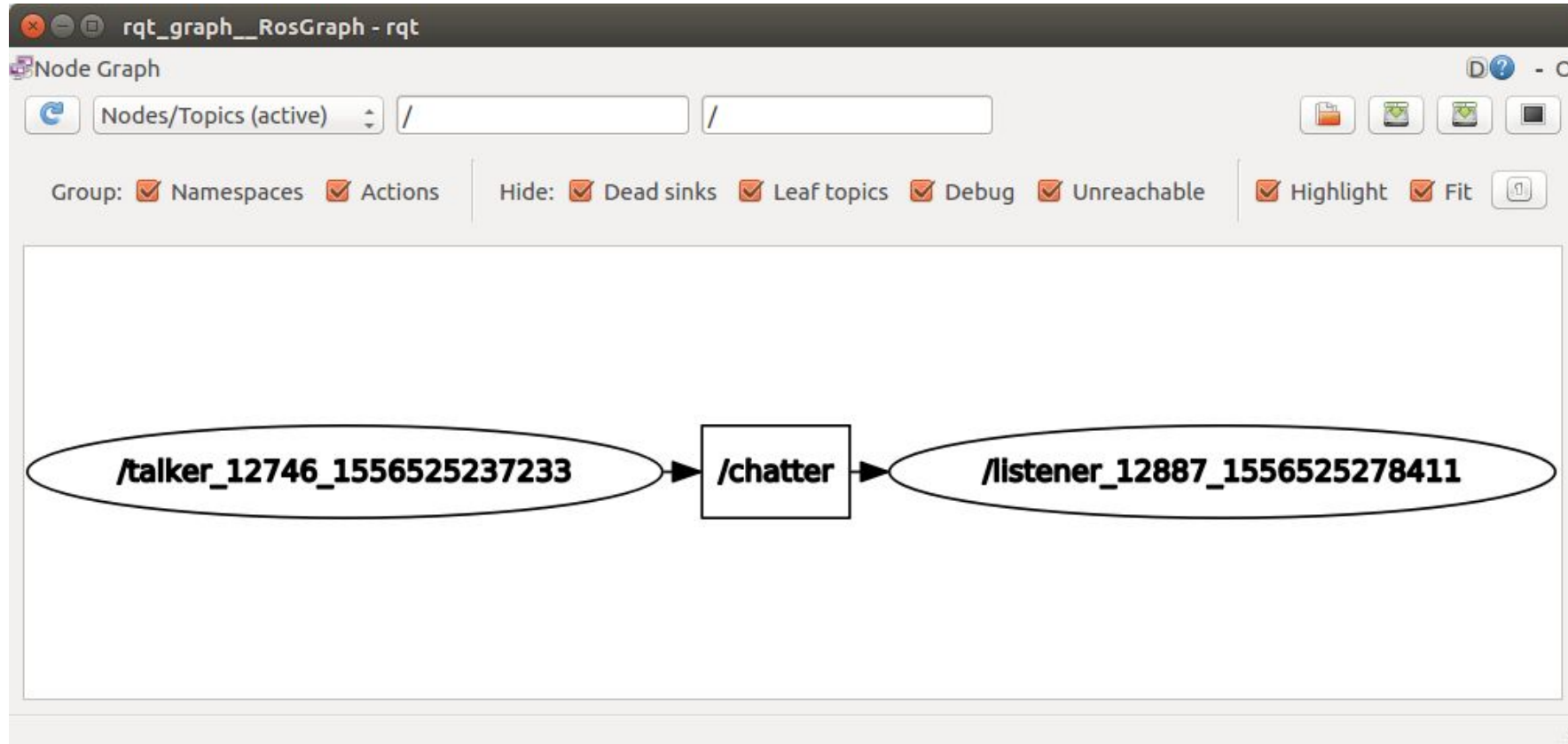
anche per `listener.py` assicuriamoci di
aver assegnato i permessi per poter essere
eseguito

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$ 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
```

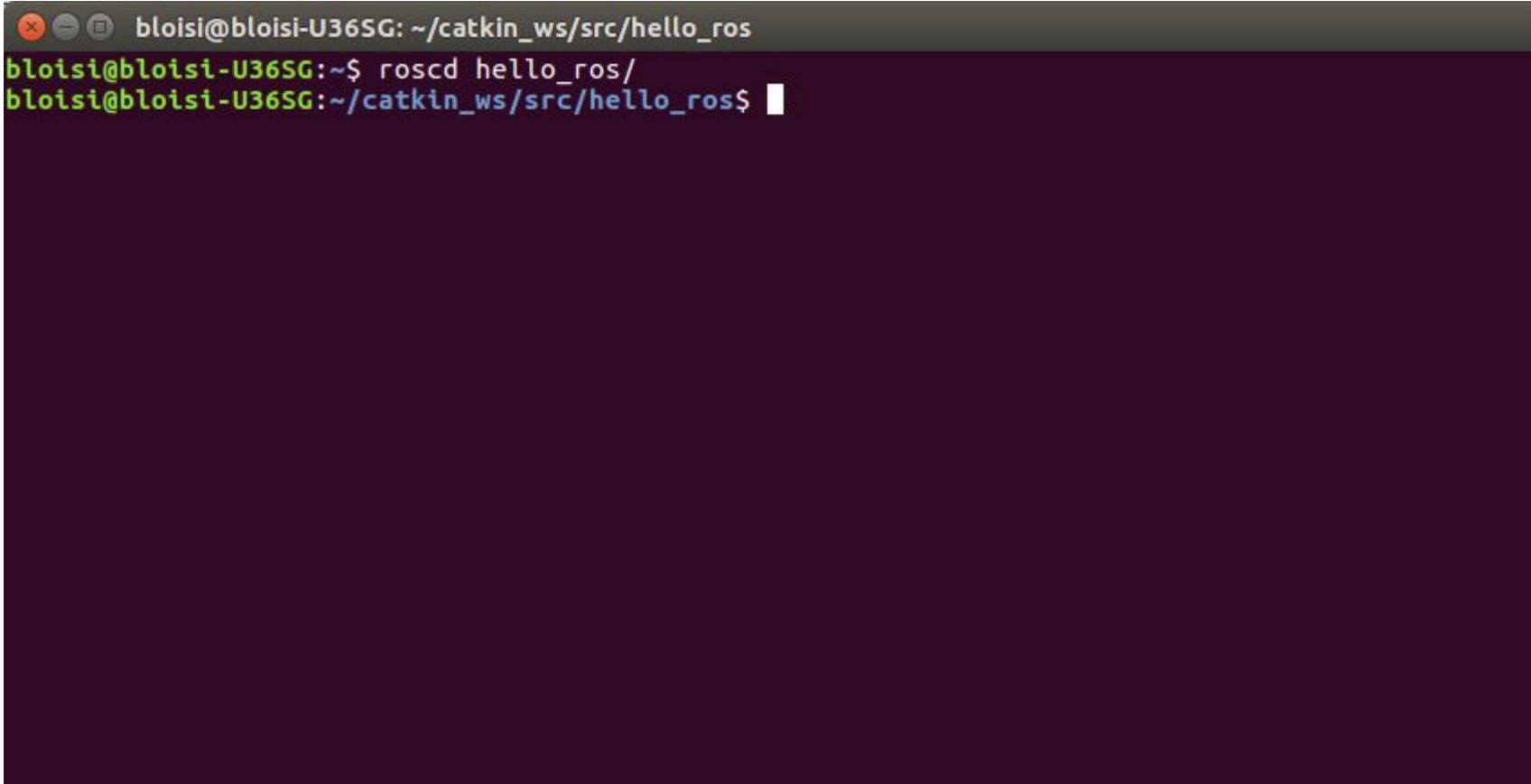
rqt_graph

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



roscd

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

A terminal window with a dark purple background. The title bar shows the user 'bloisi' on machine 'bloisi-U36SG' in the directory '~/catkin_ws/src/hello_ros'. The terminal shows the command 'roscd hello_ros/' being executed, which changes the current directory to '~/catkin_ws/src/hello_ros'.

```
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
```

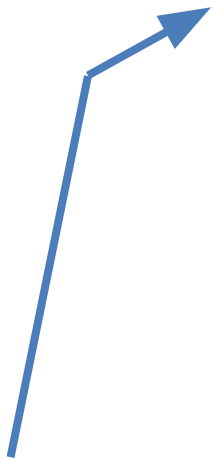
Aggiorniamo il repository locale (CMakeLists.txt)

```
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

Repository remoto aggiornato

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

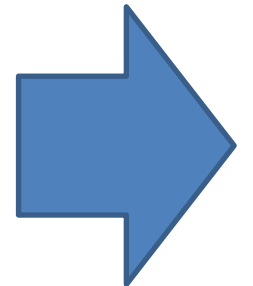
README.md

hello_ros

my first ros package

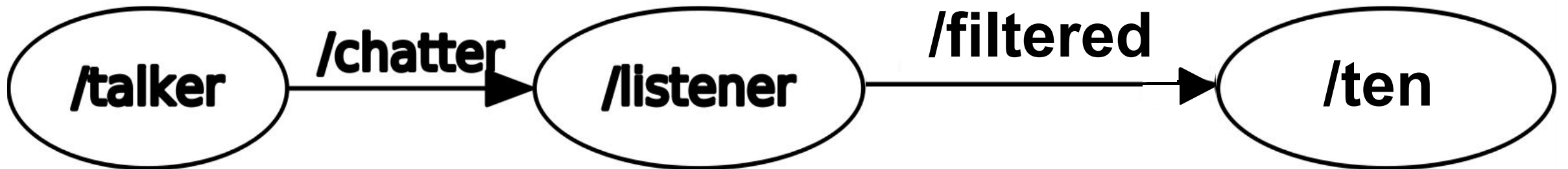
Esercitazione

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 Visione e Percezione



Docente

Domenico D. Bloisi

 **git** +

 **ROS (Python)**

