

# Tutorial 1 - Ubuntu and ROS Introduction

MIE443- Aaron Hao Tan (Head Tutorial TA)

Developed by Daniel Dworakowski, Richard Hu,  
Christopher Thompson and Aaron Hao Tan



# Important Note for all Contests!

---

- You are allowed to subscribe to any topic from the contest launch files (minimal.launch, gmapping\_demo.launch etc.), however you are only allowed to publish to the topics specified in the *contest.cpp* files.
- Piazza Signup Link:

<https://piazza.com/class/lbp5mhalunf49h>

# Downloading packages from Quercus

- Access Quercus (<http://q.utoronto.ca/>) and log-in to your account.
- On the Dashboard, access MIE443SH1 course area.

UNIVERSITY OF TORONTO

weblogin idpz

UTORid / JOINid

Password

log in

Protect Your Account

Steps you can take to protect your account:

- Before you begin, make sure this page (URL) starts <https://idpz.utorauth.utoronto.ca/>...
- When using a public computer, [close all windows](#) and exit the browser.
- Keep your password a secret at all times. Tip: U of T will **never** ask for your password or other personal information by e-mail.

Log in Problems

Forgotten Password

How to Log Out

Finding Help

Warning: Your password may not be secure. Visit our [verify password](#) page.

Completely exit your web browser when you are finished.

[edit Thu 2018-Oct-04 12:52] | Site Feedback | Accessibility | © University of Toronto

Enter your UTORid and password

Dashboard

Access MIE443HS1

Account

Dashboard

Courses

Calendar

Inbox

Course Evals

Help

MIE443H1 S LEC0101 20191: M...

MIE443H1 S LEC0101 2019 Winter

QUERCUS UNIVERSITY OF TORONTO

Coming Up [View calendar](#)

Nothing for the next week

Recent Feedback

Nothing for now

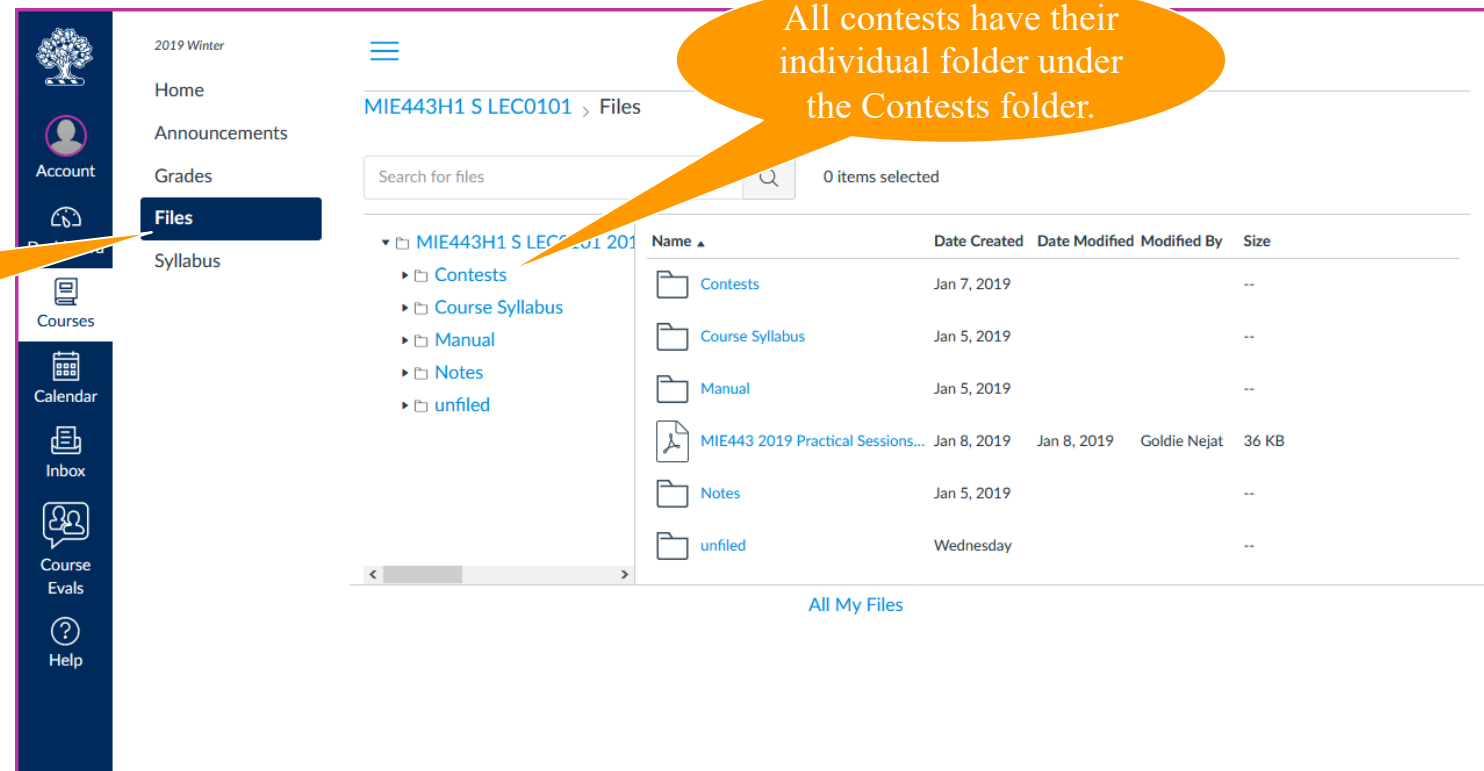
[View Grades](#)

# Accessing the Course Files

- On the course *Home* screen, click in *Files* to see all the documents and packages files uploaded to Quercus.
- Navigate to the *Contests* folder to access the source code and instructions for the contests.

Click here to see the course files.

All contests have their individual folder under the Contests folder.



The screenshot displays the Quercus course interface. On the left sidebar, the 'Files' button is highlighted. The main content area shows the 'Files' section for the course 'MIE443H1 S LEC0101'. A search bar is present at the top. Below it, a list of folders and files is shown. The 'Contests' folder is expanded, revealing its contents.

Name	Date Created	Date Modified	Modified By	Size
Contests	Jan 7, 2019			--
Course Syllabus	Jan 5, 2019			--
Manual	Jan 5, 2019			--
MIE443 2019 Practical Sessions...	Jan 8, 2019	Jan 8, 2019	Goldie Nejat	36 KB
Notes	Jan 5, 2019			--
unfiled	Wednesday			--

# Downloading Contest 1 Files

- On the *Files* section, select the *Contests* folder, then the *Contest 1* folder.
- Click on *mie443\_contest1.zip* and then in *Download mie443\_contest1.zip*.
- Also download turtlebot\_script.sh from the Ubuntu setup folder in tutorial folder.

MIE443H1 S LEC0101 > Files > Contests > Contest 1

Search for files

Q

0 items selected

▼ MIE443H1 S LEC0101 201

▼ Contests

▼ Contest 1

▼ Contest 2



▼ Contest 3

▼ Course Syllabus

▼ Manual

▼ Notes

▼ unfiled

Name ▲	Date Created	Date Modified	Modified By	Size
 Contest1.pdf	Yesterday	Yesterday	Silas Franco ...	244 KB
 mie443_contest1.zip	Jan 7, 2019	Jan 7, 2019	Silas Franco ...	6 KB

<  >

All My Files

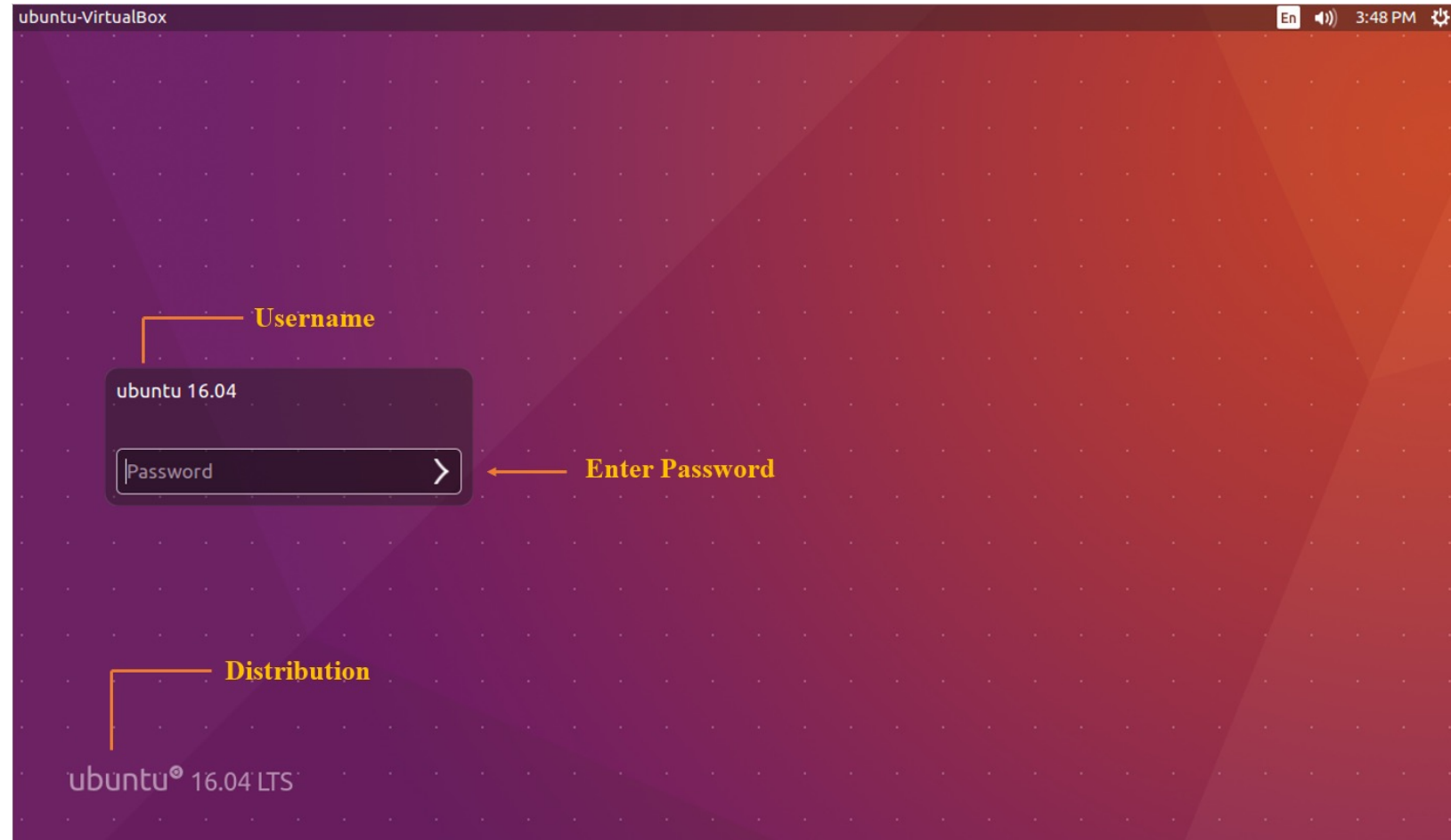


# Ubuntu Setup

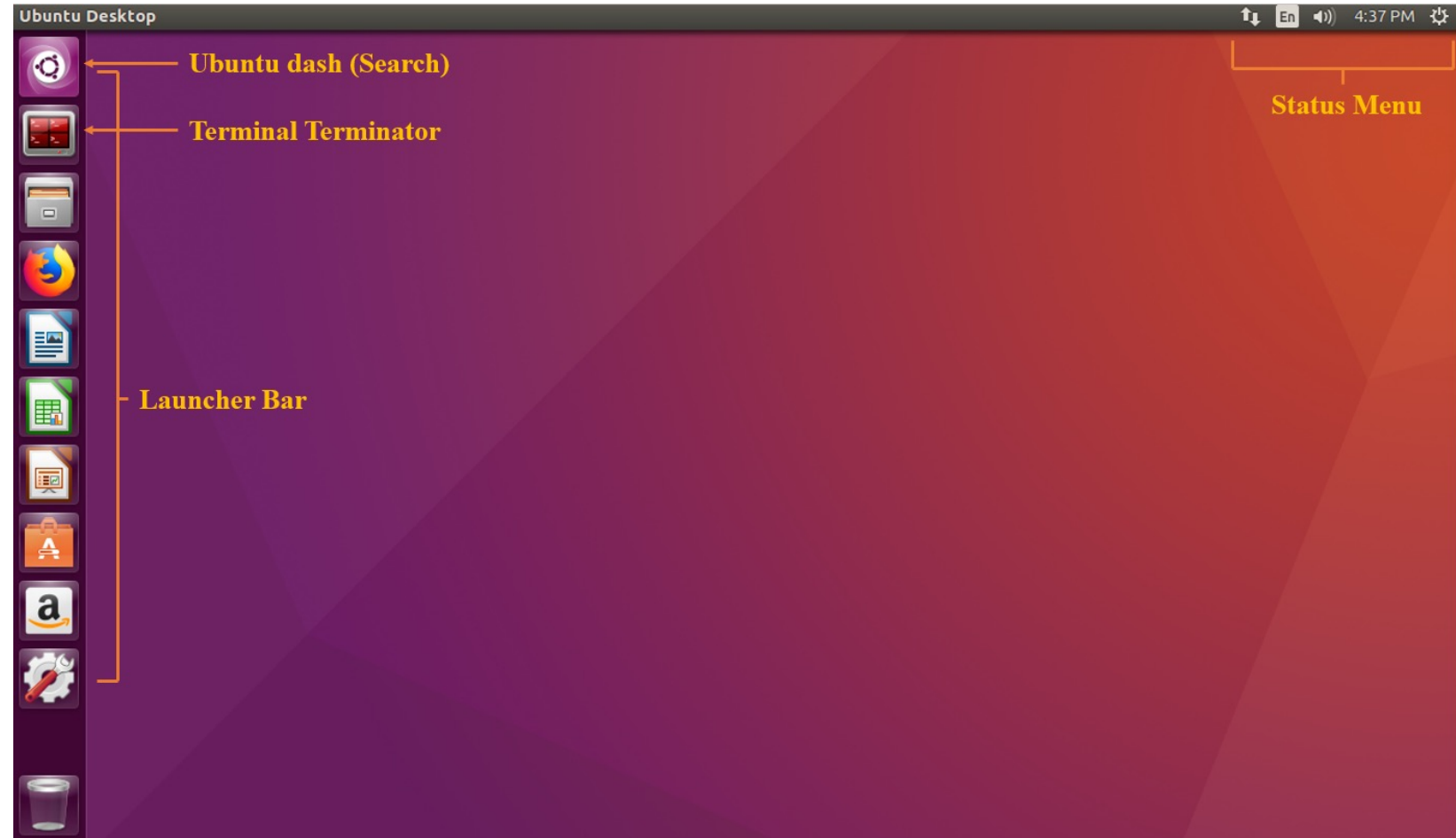
- The course requires Ubuntu 16.04 to be installed for ROS compatibility
- Multiple ways to install:
  - 1) Dual boot with windows
  - 2) Virtual box (see quercus for instructions “SettingUpUbuntuVirtualbox.pdf”)
  - 3) Docker (see quercus for instructions – for advanced users  
“SettingUpUbuntuDocker.pdf”)
    - In general, users with Macs or Microsoft surfaces must install virtual machines
- You must then execute the `turtlebot_script.sh` script from quercus to install all basic software and settings
  - After downloading the file on Ubuntu open a terminal (`alt+ctrl+t`) and enter:

```
bash ~/Downloads/turtlebot_script.sh #Press enter after copy-pasting this command and follow prompts
```

# Login Screen



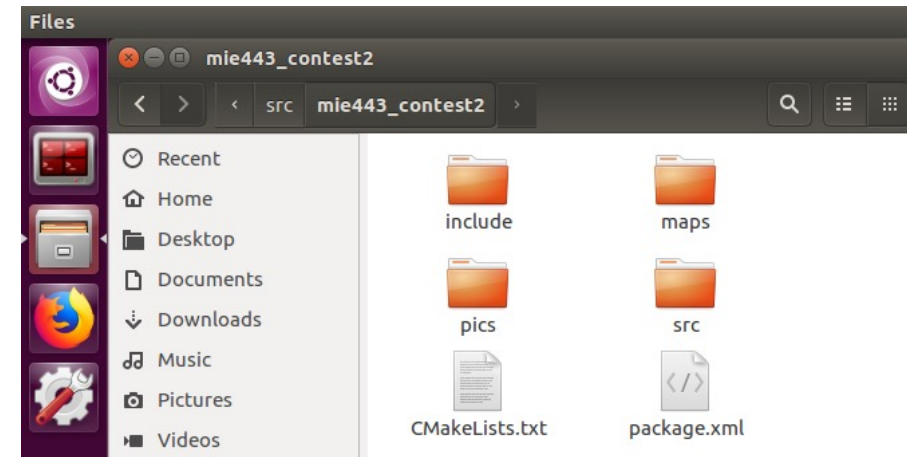
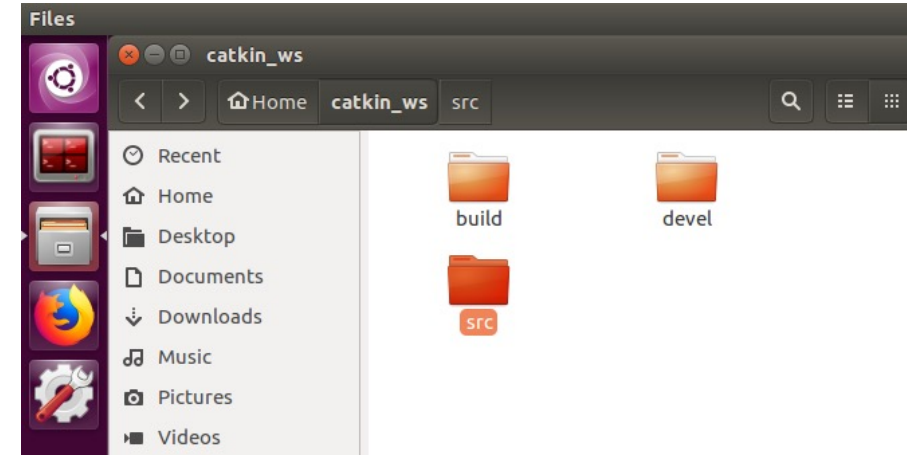
# Desktop Screen and Launcher Bar





# Workspace and Contest Packages

- Catkin Workspace
  - Directory containing all ROS files to be compiled
  - *src* folder contains all of the ROS packages
  - *build* and *devel* contain program dependencies and are automatically generated during compiling
- Package Folders
  - Directories within the workspace that contain the resources to compile the package
  - Packages can contain many folders depending on complexity of the project



# Terminal Interface

- Text-based/Non-Graphical interface
- Allows the user to navigate the computer system via text inputs and provides functionality that can be faster or is unavailable when using the graphical interface
- Command line input examples:

**cd** *file/path* # change directory  
**ls** # list directory contents  
**mkdir** *name* # create directory

```
turtlebot@turtlebot-ThinkPad-11e: ~/Desktop 162x42
turtlebot@turtlebot-ThinkPad-11e:~$ ls
catkin_ws  Desktop  Documents  Downloads  examples.desktop  Music  Pictures  Public  Templates  Videos
turtlebot@turtlebot-ThinkPad-11e:~$ cd Desktop/
turtlebot@turtlebot-ThinkPad-11e:~/Desktop$ mkdir temp
turtlebot@turtlebot-ThinkPad-11e:~/Desktop$ cd temp/
turtlebot@turtlebot-ThinkPad-11e:~/Desktop/temp$ ls
turtlebot@turtlebot-ThinkPad-11e:~/Desktop/temp$ cd ..
turtlebot@turtlebot-ThinkPad-11e:~/Desktop$ ls
screenshots  temp  turtlebot-doc.desktop
turtlebot@turtlebot-ThinkPad-11e:~/Desktop$ rm -r temp/
turtlebot@turtlebot-ThinkPad-11e:~/Desktop$ ls
screenshots  turtlebot-doc.desktop
turtlebot@turtlebot-ThinkPad-11e:~/Desktop$
```

# Terminal Emulator (Terminator)

- A wrapper program that extends the functionality of the basic terminal
  - Keyboard shortcut examples:
    - Ctrl-Shift-O – split screen up/down
    - Ctrl-Shift-E – split screen left/right
    - Ctrl-Shift-W – exit current window

```
ubuntu@ubuntu-VirtualBox: ~ 45x13
turtle_actionlib          turtlebot_foll
turtlebot_actions         turtlebot_gaze
turtlebot_bringup        turtlebot_inte
turtlebot_calibration     turtlebot_msgs
turtlebot_capabilities   turtlebot_navi
turtlebot_dashboard      turtlebot_rapp
turtlebot_description    turtlebot_rviz
ubuntu@ubuntu-VirtualBox:~$ clear
[3;J
ubuntu@ubuntu-VirtualBox:~$ roslaunch turtle
turtle_actionlib          turtlebot_foll
turtlebot_actions         turtlebot_gaze
turtlebot_bringup        turtlebot_inte

/opt/ros/kinetic/share/turtlebot_bringup/launch
File "/usr/lib/python2.7/threading.py"
, line 946, in join
<type 'exceptions.TypeError': 'NoneType
' object is not callable
[turtlebot_laptop_battery-7] process has
finished cleanly
log file: /home/ubuntu/.ros/log/d355258c
-e02e-11e7-b875-0800275efabb/turtlebot_l
aptop_battery-7*.log

ubuntu@ubuntu-VirtualBox: ~ 35x13
ubuntu@ubuntu-VirtualBox:~$ rostopi
c echo /particlecloud

/opt/ros/kinetic/share/turtlebot_rviz_launchers/launch/view_navigation.l
rviz (rviz/rviz)

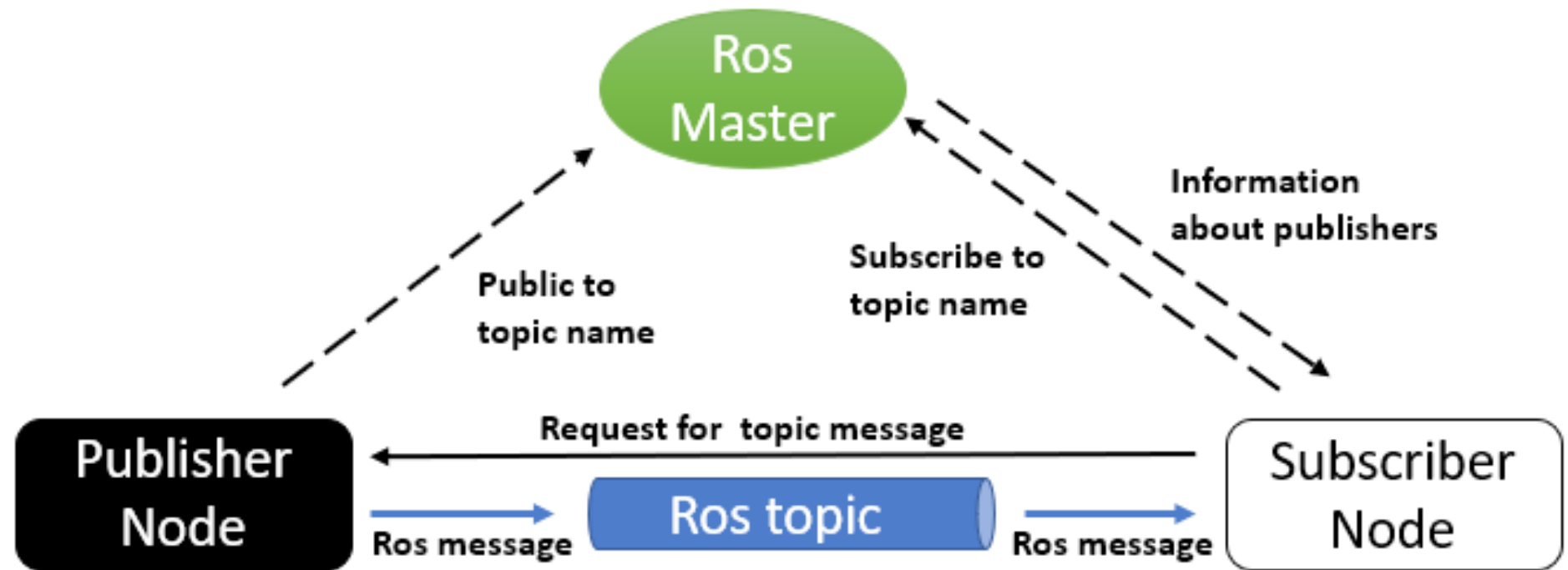
auto-starting new master
process[master]: started with pid [4679]
ROS_MASTER_URI=http://localhost:11311

setting /run_id to d355258c-e02e-11e7-b875-0800275efabb
process[rosout-1]: started with pid [4692]
started core service [/rosout]
process[rviz-2]: started with pid [4695]

ubuntu@ubuntu-VirtualBox: ~ 61x13
/move_base/NavfnROS/plan
/move_base/global_costmap/costmap
/move_base/global_costmap/costmap_updates
/move_base/local_costmap/costmap
/move_base/local_costmap/costmap_updates
/move_base/simple/goal
/particlecloud
/rosout
/rosout_agg
/scan
/tf
/tf_static
ubuntu@ubuntu-VirtualBox:~$
```

# ROS Overview

- ROS = Robot Operating System



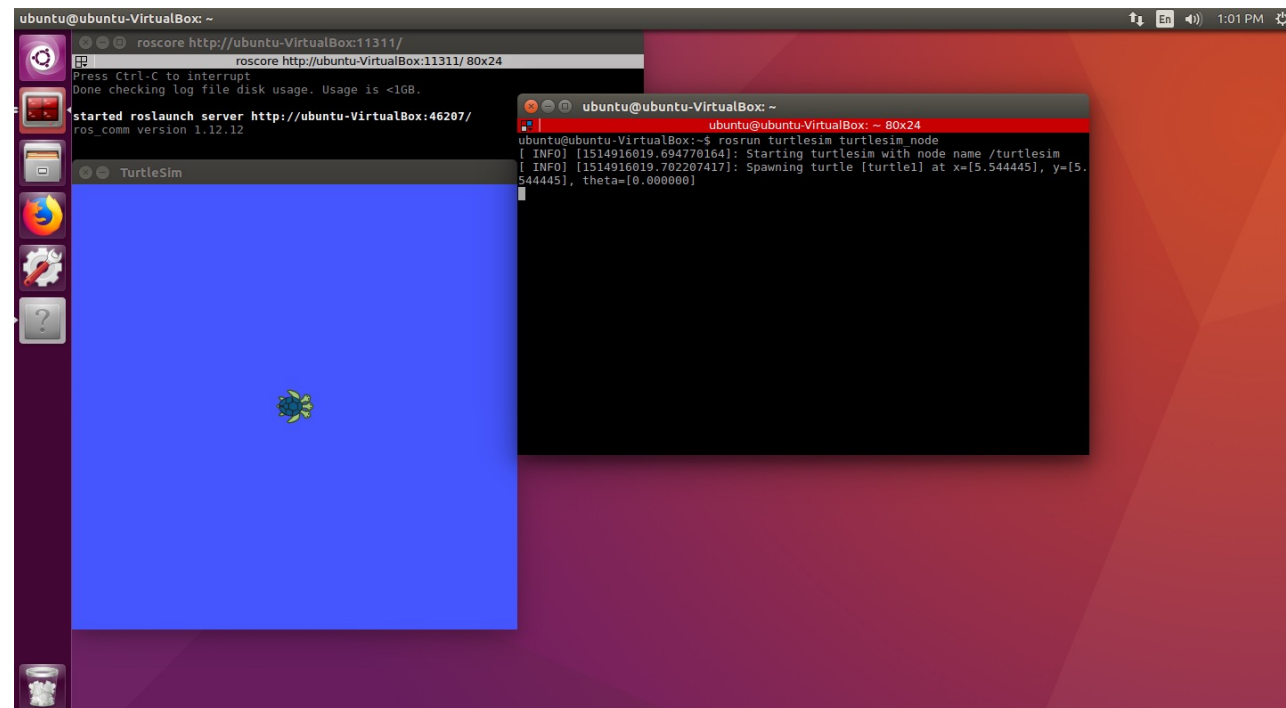
# ROS TurtleSim Example

- To run the TurtleSim example, the following commands have to be entered in their own respective terminals:

**roscore** #Initializes the ROS master

**roslaunch turtlesim turtlesim\_node** #Launches the turtlesim GUI

**roslaunch turtlesim turtle\_teleop\_key** #Sends teleop commands to the turtlesim



# Demonstrating Topics

- The following commands illustrate how you can use the rostopic command to see the topic data passing between programs:

```
roslaunch mie443_contest1 turtlebot_world.launch world:=1 #Launch gazebo
roslaunch turtlebot_teleop keyboard_teleop.launch #Launch keyboard teleop
rostopic echo /mobile_base/events/bumper #listen to a published topic
```

```
turtlebot@turtlebot-ThinkPad-11e: ~
turtlebot@turtlebot-ThinkPad-11e: ~ 80x24

turtlebot@turtlebot-ThinkPad-11e:~$ rostopic echo /mobile_base/events/bumper
bumper: 0
state: 1
---
bumper: 1
state: 1
---
bumper: 0
state: 0
---
bumper: 1
state: 0
---
bumper: 1
state: 1
---
bumper: 0
state: 1
---
bumper: 0
state: 0
---
bumper: 1
```

```
/opt/ros/kinetic/share/turtlebot_teleop/launch/keyboard_teleop.launch http://lo
/home/turtlebot/catkin_ws/src/mie443_contest1
/opt/ros/kinetic/share/turtlebot_teleop/launch/keyboard_teleop.launch http://localhost:11311 85x21
ROS_MASTER_URI=http://localhost:11311

process[turtlebot_teleop_keyboard-1]: started with pid [23476]

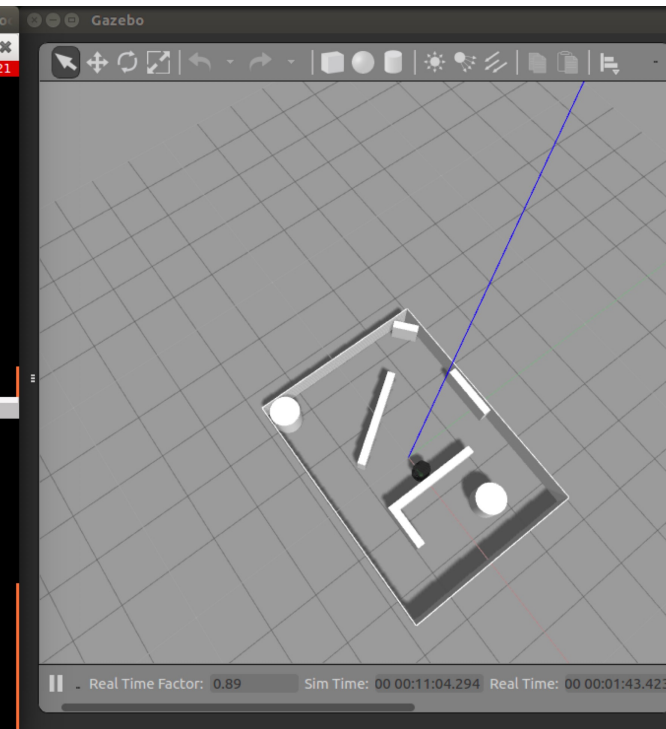
Control Your Turtlebot!
-----
Moving around:
  u   i   o
  j   k   l
  m   ,   .

q/z : increase/decrease max speeds by 10%
w/x : increase/decrease only linear speed by 10%
e/c : increase/decrease only angular speed by 10%
space key, k : force stop
anything else : stop smoothly

CTRL-C to quit

currently:    speed 0.2    turn 1

turtlebot@turtlebot-Latitude-E7250: ~/catkin_ws/src/mie443_contest1 85x21
state: 0
---
bumper: 1
state: 1
---
bumper: 1
state: 0
---
bumper: 1
state: 1
---
bumper: 1
state: 0
---
bumper: 1
state: 1
---
bumper: 1
state: 0
---
```



# Gmapping Demo

- The following commands illustrate how to use the simulated Turtlebot gmapping code to begin creating 2D maps of the environment:

```
roslaunch mie443_contest1 turtlebot_world.launch world:=practice #Launch gazebo
```

```
roslaunch mie443_contest1 gmapping.launch #Begins the mapping software
```

```
roslaunch turtlebot_teleop keyboard_teleop.launch #begin Turtlebot base teleop software
```

```
/opt/ros/kinetic/share/turtlebot_gazebo/launch/gmapping_demo.launch http://localhost:11311 8
* /slam_gmapping/minimumScore: 200
* /slam_gmapping/odom_frame: odom
* /slam_gmapping/ogain: 3.0
* /slam_gmapping/particles: 80
* /slam_gmapping/resampleThreshold: 0.5
* /slam_gmapping/sigma: 0.05
* /slam_gmapping/srr: 0.01
* /slam_gmapping/srt: 0.02
* /slam_gmapping/str: 0.01
* /slam_gmapping/stt: 0.02
* /slam_gmapping/temporalUpdate: -1.0
* /slam_gmapping/xmax: 1.0
* /slam_gmapping/xmin: -1.0
* /slam_gmapping/ymax: 1.0
* /slam_gmapping/ymin: -1.0

NODES
/
  slam_gmapping (gmapping/slam_gmapping)

ROS_MASTER_URI=http://localhost:11311

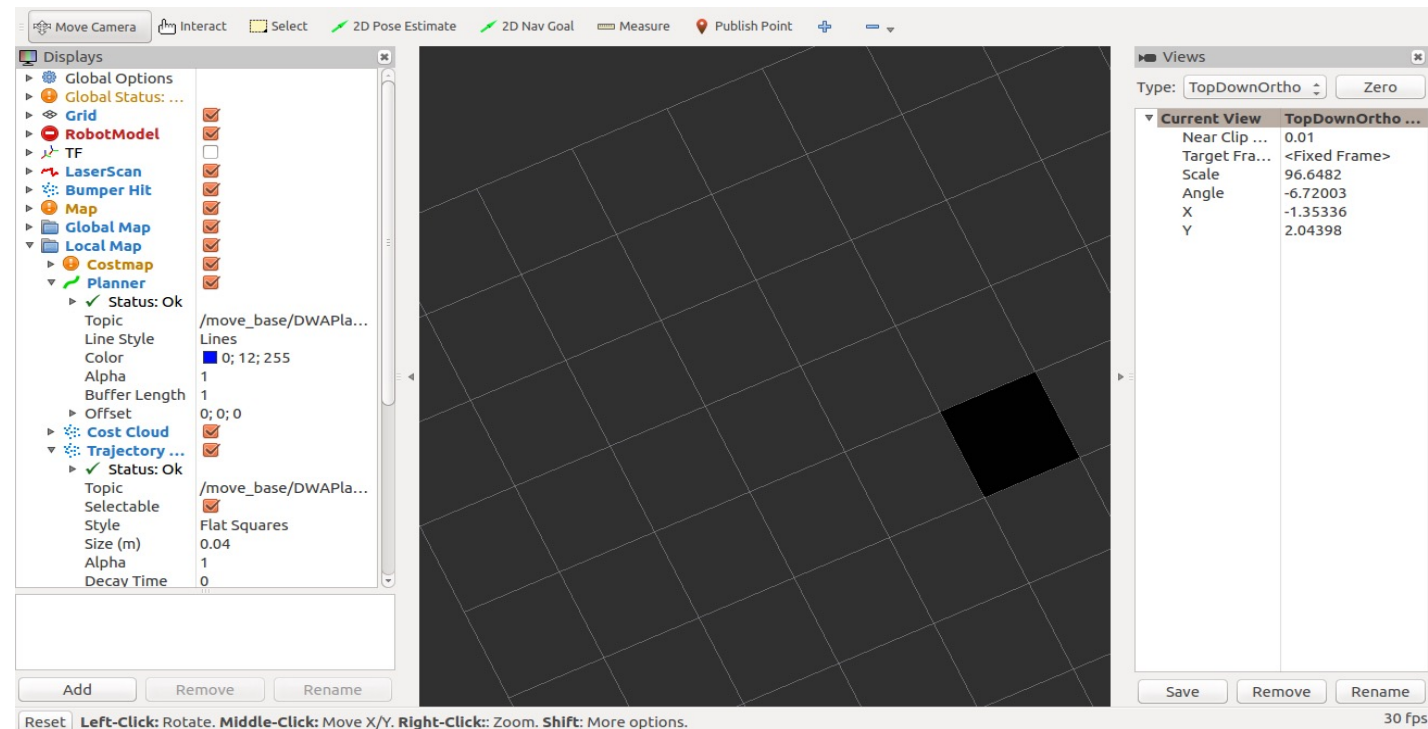
process[slam_gmapping-1]: started with pid [7173]
```



# Visualize Map

- To visualize various topics in ROS a program called Rviz is used. The following command is used to open Rviz and visualize the progress of the map being created by the simulated TurtleBot:

**`roslaunch turtlebot_rviz_launchers view_navigation.launch` #Opens RVIZ that has turtlebot settings for visualizing a map**

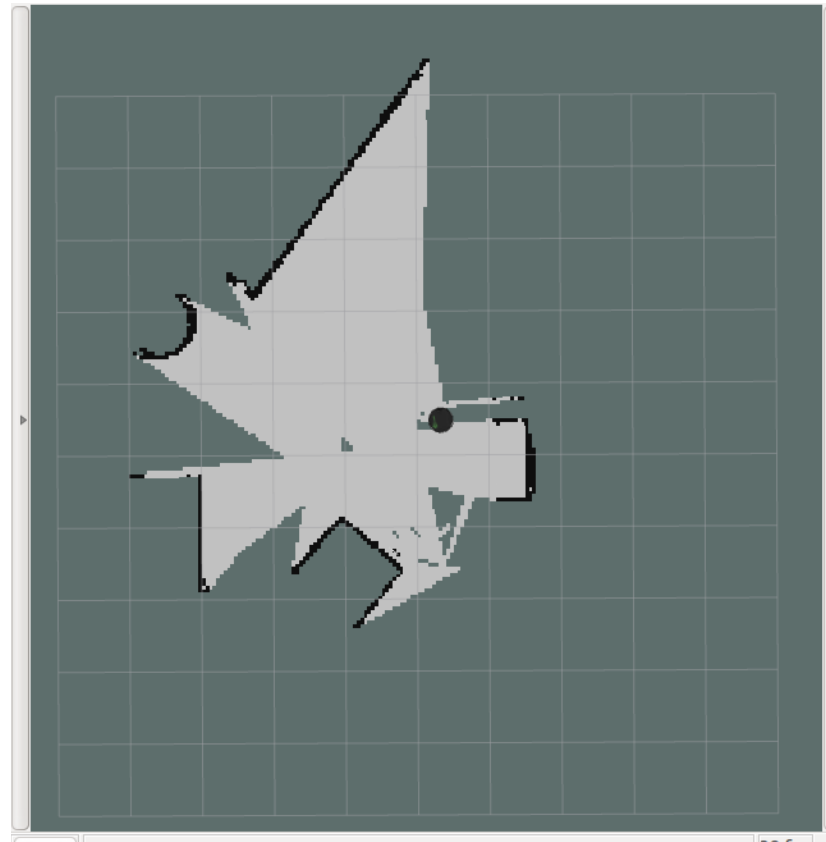




# Saving a Map

- The following command is used to save a map being created with gmapping on the simulated Turtlebot for future use:

```
roslaunch map_server map_saver -f your_map_name #Saves the gmapping map to the current directory with your own file name
```



# Questions?



# Talker & Listener Live Demo

1. Open a terminal: *ctrl + alt + t*
2. Make a catkin workspace folder and a subfolder called src

```
mkdir -p ~/talker_ws/src
```

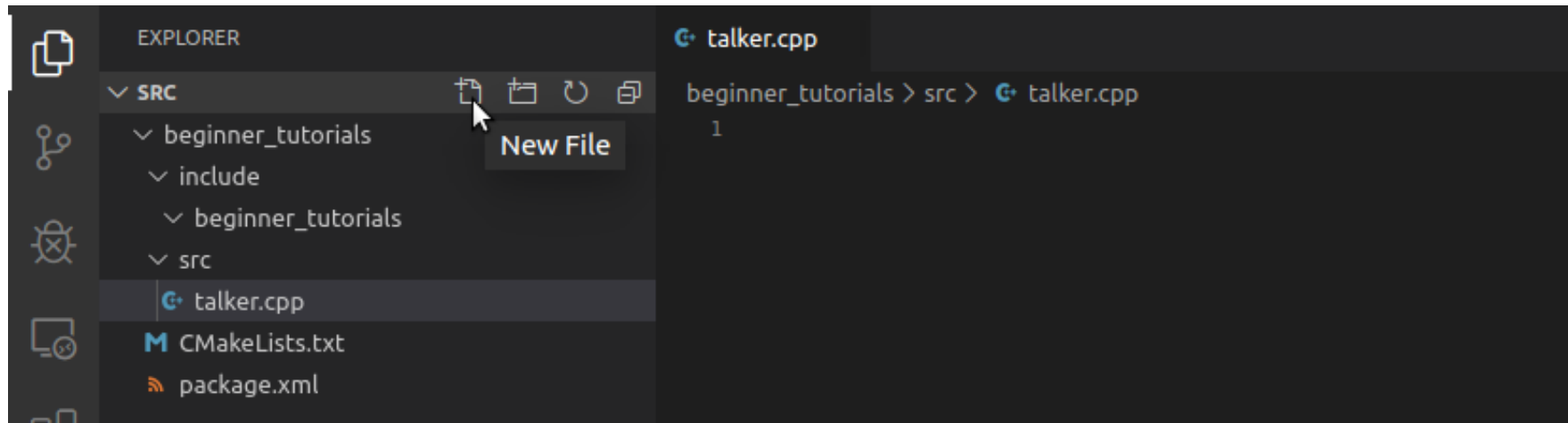
3. Create a ros package called beginner\_tutorials, that uses the libraries: std\_msgs, rospy, and roscpp

```
cd ~/talker_ws/src  
catkin_create_pkg beginner_tutorials std_msgs rospy roscpp
```

4. Open VScode at this location, make sure to include the dot in the command  
code .

# Talker & Listener Live Demo

1. Create a *talker.cpp* file under *beginner\_tutorials/src* folder, **alternatively**, this could also be done with the terminal command below:



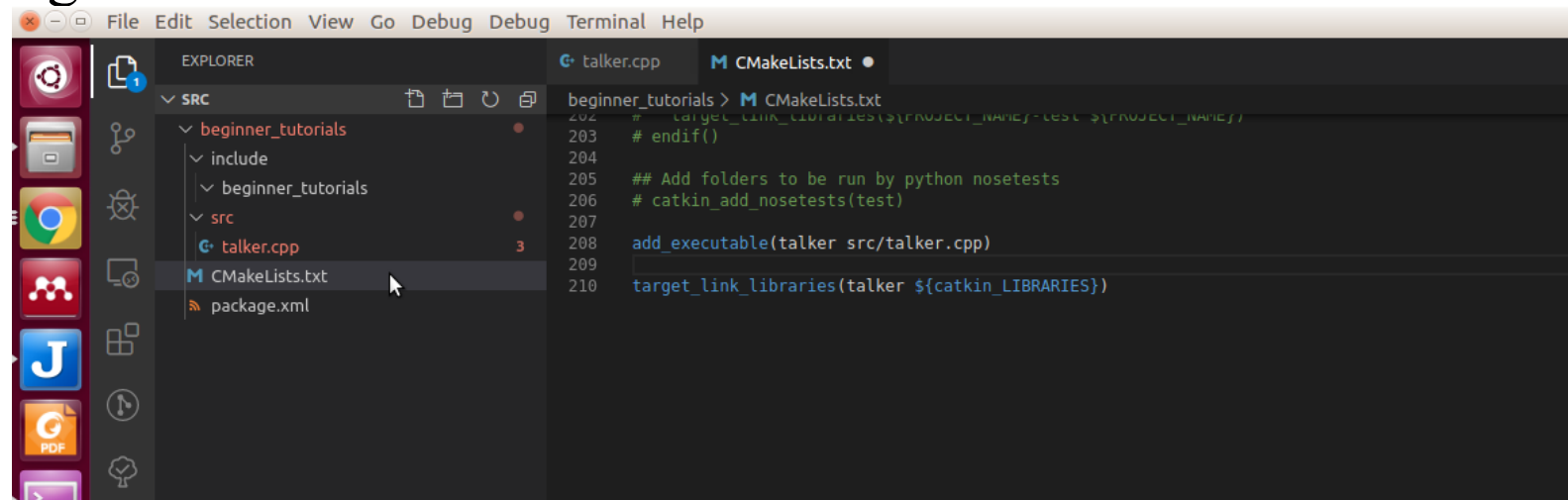
```
touch ~/talker_ws/src/beginner_tutorials/src/talker.cpp
```

2. Copy and paste the code from this link into the *talker.cpp* file:

[https://raw.githubusercontent.com/ros/ros\\_tutorials/kinetic-devel/roscpp\\_tutorials/talker/talker.cpp](https://raw.githubusercontent.com/ros/ros_tutorials/kinetic-devel/roscpp_tutorials/talker/talker.cpp)

# Talker & Listener Live Demo

1. Open the *CMakeLists.txt* of your *beginner\_tutorials* package and append the compiling instructions to the bottom of the file:



```
add_executable(talker src/talker.cpp)  
target_link_libraries(talker ${catkin_LIBRARIES})
```

2. Back in your terminal, build your workspace:

```
cd ~/talker_ws/  
catkin_make
```

# Talker & Listener Live Demo

1. Run *roscore* in a terminal.
2. In a **separate** terminal, follow the instruction to source the workspace you just built and run the talker node:

**roscore #run this in terminal 1**

**#run these in terminal 2**

**cd talker\_ws**

**source devel/setup.bash**

**roslaunch beginner\_tutorials talker**

```
roscore http://thousandsunny:11311/ 57x66
richard@thousandsunny:~
$ roscore
... logging to /home/richard/.ros/log/3f2bee3a-202e-11ea-
alff-d46d6dec383a/roslaunch-thousandsunny-24516.log
Checking log directory for disk usage. This may take awhi
le.
Press Ctrl-C to interrupt
Done checking log file disk usage. Usage is <1GB.

started roslaunch server http://thousandsunny:45319/
ros_comm version 1.12.14

SUMMARY
=====
PARAMETERS
* /rostdistro: kinetic
* /rosversion: 1.12.14

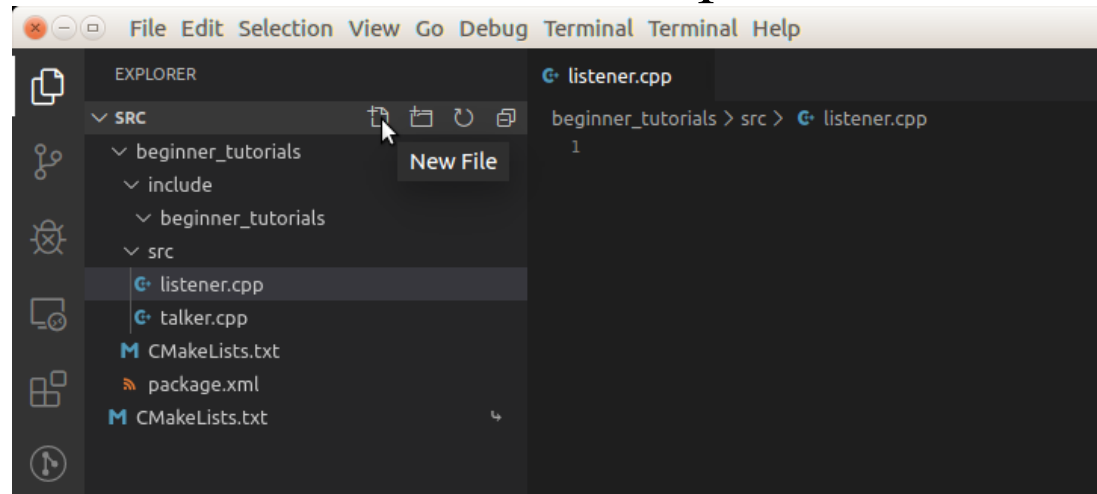
NODES
auto-starting new master
process[master]: started with pid [24526]
ROS_MASTER_URI=http://thousandsunny:11311/

setting /run_id to 3f2bee3a-202e-11ea-alff-d46d6dec383a
process[roslaunch-1]: started with pid [24539]
started core service [/roslaunch]

richard@thousandsunny:~
$ cd talker_ws/
/home/richard/talker_ws
richard@thousandsunny:~/talker_ws
$ source devel/setup.bash
richard@thousandsunny:~/talker_ws
$ roslaunch beginner_tutorials talker
[ INFO] [1576519445.619909334]: hello world 0
[ INFO] [1576519445.720018301]: hello world 1
[ INFO] [1576519445.820021916]: hello world 2
[ INFO] [1576519445.920052429]: hello world 3
[ INFO] [1576519446.020153409]: hello world 4
[ INFO] [1576519446.120131831]: hello world 5
[ INFO] [1576519446.220137756]: hello world 6
[ INFO] [1576519446.320149264]: hello world 7
[ INFO] [1576519446.420070209]: hello world 8
[ INFO] [1576519446.520137963]: hello world 9
[ INFO] [1576519446.620139347]: hello world 10
[ INFO] [1576519446.720074643]: hello world 11
[ INFO] [1576519446.820135067]: hello world 12
[ INFO] [1576519446.920069841]: hello world 13
[ INFO] [1576519447.020025811]: hello world 14
[ INFO] [1576519447.120058458]: hello world 15
[ INFO] [1576519447.220064903]: hello world 16
[ INFO] [1576519447.320056200]: hello world 17
[ INFO] [1576519447.420136652]: hello world 18
[ INFO] [1576519447.520134132]: hello world 19
[ INFO] [1576519447.620058877]: hello world 20
[ INFO] [1576519447.720076392]: hello world 21
```

# Talker & Listener Live Demo

1. Create a *listener.cpp* under *beginner\_tutorials/src* folder, similar procedure as the previous slides. Terminal command is provided below:

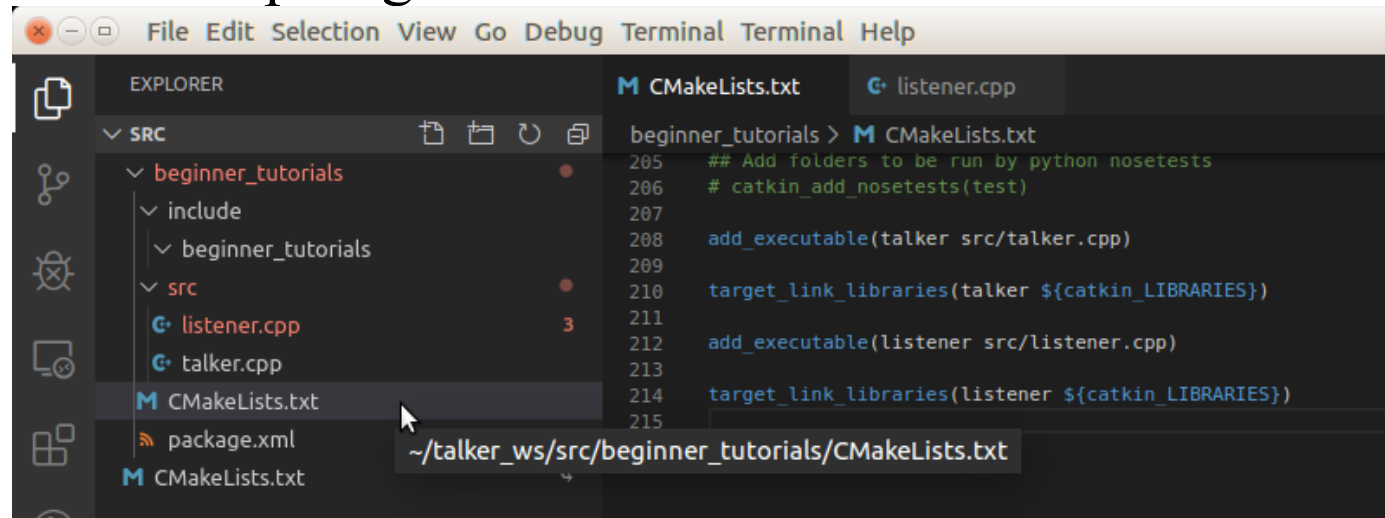


```
touch ~/talker_ws/src/beginner_tutorials/src/listener.cpp
```

2. Copy and paste the code from this link into the *listener.cpp* file: [https://raw.githubusercontent.com/ros/ros\\_tutorials/kinetic-devel/roscpp\\_tutorials/listener/listener.cpp](https://raw.githubusercontent.com/ros/ros_tutorials/kinetic-devel/roscpp_tutorials/listener/listener.cpp)

# Talker & Listener Live Demo

1. Similar to before, open the *CMakeLists.txt* of your *beginner\_tutorials* package and append the compiling instructions to the bottom of the file:



```
add_executable(listener src/listener .cpp)
target_link_libraries(listener ${catkin_LIBRARIES})
```

2. Back in your terminal, build your workspace:

```
cd ~/talker_ws/
catkin_make
```



# Talker & Listener Live Demo

1. Run *roscore* in a terminal.
2. In a **separate** terminal, follow the instruction to source the workspace you just built and run the talker node:

**roscore #run this in terminal 1**

**#run these in terminal 2**

**cd talker\_ws**

**source devel/setup.bash**

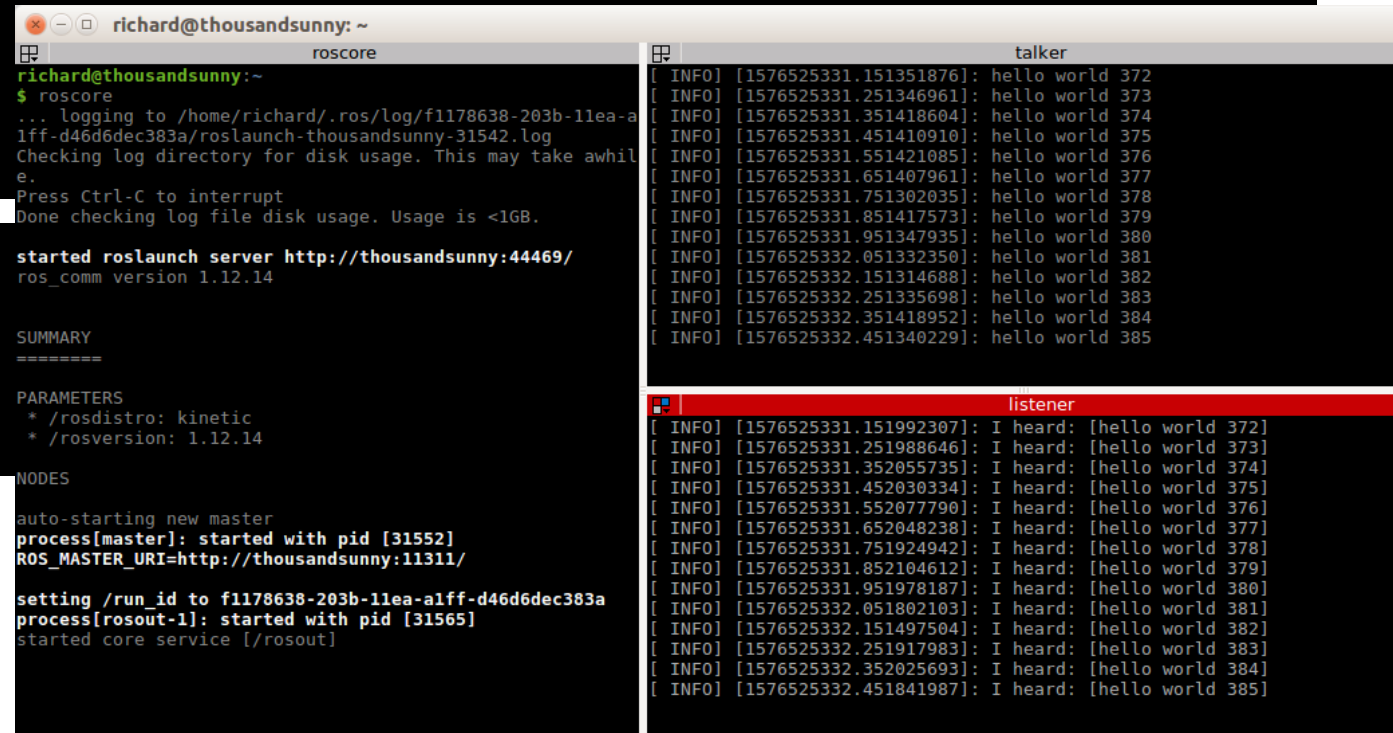
**roslaunch beginner\_tutorials talker**

**#run these in terminal 3**

**cd talker\_ws**

**source devel/setup.bash**

**roslaunch beginner\_tutorials listener**

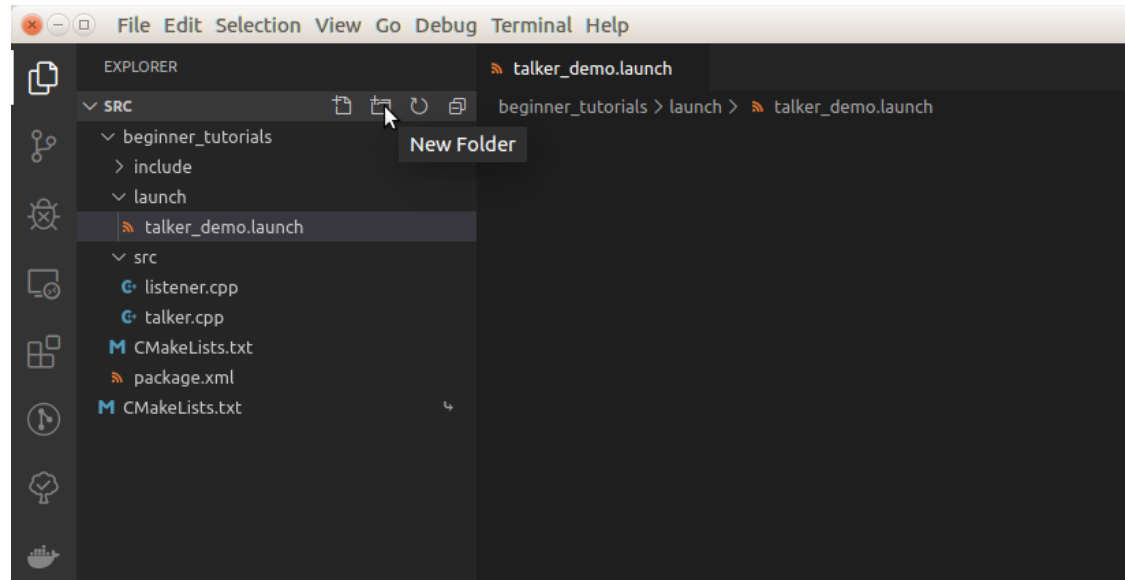


The screenshot displays three terminal windows from a user named 'richard' on a machine named 'thousandsunny'.

- Terminal 1 (roscore):** Shows the command 'roscore' being executed. It logs the user's location and checks disk usage. It then starts the 'roslaunch server' at 'http://thousandsunny:44469/' and reports 'ros\_comm version 1.12.14'. A summary and parameters section follows, indicating the ROS distribution is 'kinetic' and the version is '1.12.14'. The nodes section shows 'auto-starting new master', 'process[master]: started with pid [31552]', and 'ROS\_MASTER\_URI=http://thousandsunny:11311/'. It also shows 'setting /run\_id to f1178638-203b-11ea-af1f-d46d6dec383a', 'process[roslaunch-1]: started with pid [31565]', and 'started core service [/roslaunch]'.
- Terminal 2 (talker):** Displays a series of log messages from the 'talker' node. Each message is an INFO log with a timestamp and the text 'hello world' followed by a sequence number (e.g., 372, 373, 374, etc.).
- Terminal 3 (listener):** Displays a series of log messages from the 'listener' node. Each message is an INFO log with a timestamp and the text 'I heard: [hello world]' followed by a sequence number (e.g., 372, 373, 374, etc.).

# Talker & Listener Live Demo

1. Create a folder called *launch* under *beginner\_tutorials*, and a file called *talker\_demo.launch* in the *launch* folder:



2. Type in the following lines into the *talker\_demo.launch* file:

```
<launch>
  <node name="talker" pkg="beginner_tutorials" type="talker" output="screen"/>
  <node name="listener" pkg="beginner_tutorials" type="listener" output="screen"/>
</launch>
```

# Talker & Listener Live Demo

1. You can launch several nodes and a roscore at the same time using a launch file. Follow the instruction to run the launch file, note that you do not need to rebuild your workspace nor to run a *roscore* for these steps:

```
cd talker_ws
source devel/setup.bash
roslaunch beginner_tutorials talker_demo.launch
```

```
richard@thousandsunny: ~
/home/richard/talker_ws/src/beginner_tutorials/launch/talker_demo.launch http://localhost:11311 116x32
* /rostdistro: kinetic
* /rosversion: 1.12.14

NODES
/
  listener (beginner_tutorials/listener)
  talker (beginner_tutorials/talker)

auto-starting new master
process[master]: started with pid [1736]
ROS_MASTER_URI=http://localhost:11311

setting /run_id to 5abbc9da-203e-11ea-af1f-d46d6dec383a
process[rosout-1]: started with pid [1749]
started core service [/rosout]
process[talker-2]: started with pid [1752]
process[listener-3]: started with pid [1753]
[ INFO] [1576526277.834846221]: hello world 0
[ INFO] [1576526277.934986615]: hello world 1
[ INFO] [1576526278.034937614]: hello world 2
[ INFO] [1576526278.134899074]: hello world 3
[ INFO] [1576526278.234961084]: hello world 4
[ INFO] [1576526278.235719717]: I heard: [hello world 4]
[ INFO] [1576526278.334969407]: hello world 5
[ INFO] [1576526278.335541721]: I heard: [hello world 5]
[ INFO] [1576526278.434970544]: hello world 6
[ INFO] [1576526278.435543459]: I heard: [hello world 6]
[ INFO] [1576526278.534974723]: hello world 7
[ INFO] [1576526278.535519213]: I heard: [hello world 7]
[ INFO] [1576526278.634918917]: hello world 8
[ INFO] [1576526278.635491419]: I heard: [hello world 8]
```

# Questions?



# Visual Studio Code C++ Linter

- C++ Linter will show you the errors in your code.

```
14 ros::NodeHandle n;
15
16 ros::Publisher chatter_pub = n.advertise<std_msgs::String>("chatter", 1000);
17
18 ros::Rate loop_rate(10);
19
20 int count = 0;
21 while (ros::ok())
22 {
23
24     std_msgs::String msg;
25
26     std::stringstream ss;
27     ss << "hello world " << count;
28     msg.data = ss.str();
29
30     ROS_INFO("%s", msg.data.c_str());

```

PROBLEMS 3 SEARCH OUTPUT DEBUG CONSOLE Filter: E.g.: text, \*\*/\*.ts, !\*\*/node...

No problems have been detected in the workspace so far.

```
16 ros::Publisher chatter_pub = n.advertise<std_msgs::String>("chatter", 1000);
17
18 ros::Rate loop_rate(10);
19
20 int count = 0;
21 while (ros::ok())
22 {
23     making an error here
24
25     std_msgs::String msg;
26
27     std::stringstream ss;
28     ss << "hello world " << count;
29     msg.data = ss.str();
30

```

PROBLEMS 3 SEARCH OUTPUT DEBUG CONSOLE Filter: E.g.: text, \*\*/\*.ts, !\*\*/no...

talker.cpp beginner\_tutorials/src 3

- ✗ identifier "making" is undefined [23, 5]
- ✗ expected a ';' [23, 15]
- ✗ identifier "msg" is undefined [29, 5]

# Visual Studio Code C++ Linter

- Setting up a Linter

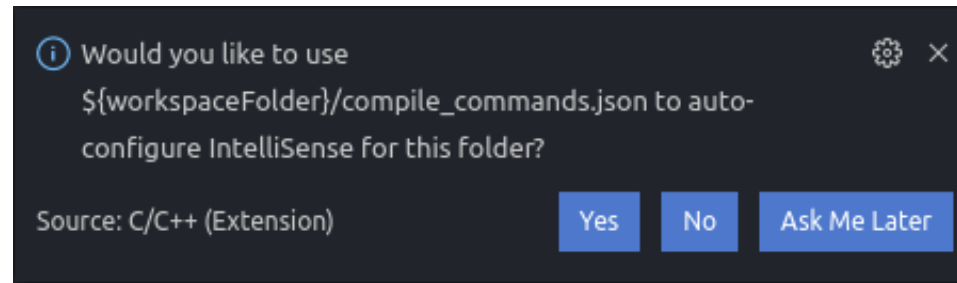
1. run the following command to generate compile\_commands.json, as well as moving it under your catkin\_ws/src folder:

```
catkin_make -DCMAKE_EXPORT_COMPILE_COMMANDS=ON && mv build/compile_commands.json src/
```

2. open your workspace in vscode, the easiest way to do it is in terminal:

```
code ~/catkin_ws/src/
```

3. Click yes when this window pops up:



## Notes:

1. You only need to do step 1 once, unless you add new libraries or new packages to your code.
2. First time using the linter will take a minute to parse your files, it will be faster after.
3. The linter detects changes when you save your file.