

# ROS Tutorial

|             |  |
|-------------|--|
| Created By  |  inbae Kang |
| Last Edited | @Oct 29, 2019 2:13 PM  |
| Property    |  |
| Tags        | Document ROS   |

## ROS\_Tutorial

### 1. Installation

- Set up your source.list

```
sudo sh -c 'echo "deb <http://packages.ros.org/ros/ubuntu> $(lsb_release -sc) main" > /etc/apt/sources.list.d/ros-latest.list'
```

- Set up Keys

```
sudo apt-key adv --keyserver 'hkp://keyserver.ubuntu.com:80' --recv-key C1CF6E31E6BADE8868B172B4F42ED6FBAB17C654
```

- Update debian package index

```
sudo apt-get update
```

- ROS Install

```
sudo apt-get install ros-kinetic-desktop-full
```

- Initialize ROSDEP

```
sudo rosdep init  
rosdep update
```

- Envirnment Setup

```
echo "source /opt/ros/kinetic/setup.bash" >> ~/.bashrc  
source ~/.bashrc
```

- Install dependencies Packages

```
sudo apt install python-rosinstall python-rosinstall-generator python-wstool build-essential
```

- Create ROS Workspace

```
mkdir -p ~/catkin_ws/src  
cd ~/catkin_ws/  
catkin_make  
source devel/setup.bash
```

### 2. ROS File System

- 진행하기 위해서는 ROS Tutorial 패키지를 설치해야 함. (distro는 버전에 따라 다름. 여기선 kinetic)

```
sudo apt-get install ros-<distro>-ros-tutorials
```

- File System Tools

- rospack : package에 대한 정보를 얻을 수 있음.
- rospack find : package 위치 정보 확인

```
rospack find roscpp
```

- roscl
- roscl : ros package의 위치로 이동.(서브디렉토리로도 이동 가능)

```
roscl roscpp  
roscl roscpp/cmake
```

- roscl log : ROS가 log를 저장하는 디렉토리로 이동.

```
roscl log
```

- roscl : ROS 디렉토리들의 파일 목록 바로 조회.

```
roscl roscpp_tutorial
```

- TAB 자동완성 가능.

### 3. Creating ROS Package

- Catkin package는 적어도 CMakeList.txt 와 package.xml 파일을 가지고 있어야 함.
- catkin package 만들기
- 디렉토리 이동 후 패키지 생성(dependencies 포함)

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

- 일반적인 Form

```
catkin_create_pkg <package_name> [depend1] [depend2] [depend3]
```

- Building a catkin workspace and sourcing the setup file
- 디렉토리 이동 후 catkin\_make

```
cd ~/catkin_ws  
catkin_make  
source ~/catkin_ws/devel/setup.bash
```

- Package dependencies
- First-order dependencies : First Order Dependencies를 확인

```
rospack depends1 beginner_tutorials
```

- Indirect dependencies : Recursive하게 모든 Dependencies를 확인

```
rospack depends beginner_tutorials
```

- Customizing Package
- package directory 안의 package.xml 파일을 수정하여 커스터마이징
- description Tag

```
<description>프로젝트에 대한 대략적인 설명</description>
```

- maintainer Tag : 최소 한명 이상의 package maintainer 있어야 함.

```
<maintainer email="cleverdevk@gmail.com>Inbae Kang</maintainer>
```

- license Tag

```
<license>BSD or MIT or GPLv2....</license>
```

- dependencies Tags : package의 dependency를 명시하는데 4가지로 나뉨. 자세한 내용은 [여기](#)를 참조.
- build\_depend : 빌드타임에 패키지 설치에 필요한 패키지들을 명시
- buildtool\_depend : 패키지를 빌드할 때 필요한 build system tool을 명시
- exec\_depend : 패키지를 실행하는데 필요한 패키지들을 명시.
- test\_depend : 오직 unit test를 위해서 필요한 추가적인 패키지를 명시.

```
<!-- Examples -->
<buildtool_depend>catkin</buildtool_depend>

<build_depend>message_generation</build_depend>
<build_depend>roscpp</build_depend>
<build_depend>std_msgs</build_depend>

<run_depend>message_runtime</run_depend>
<run_depend>roscpp</run_depend>
<run_depend>rospy</run_depend>
<run_depend>std_msgs</run_depend>

<test_depend>python-mock</test_depend>
```

#### 4. Building a ROS Package

- Building Packages
- catkin\_make 사용하기 : catkin\_make는 cmake와 make를 결합한 call이라고 생각하면 됨.

```
# catkin workspace에서..(catkin_ws)
catkin_make
catkin_make install # (optionally)
```

위의 명령어는 아래의 CMake 프로젝트의 Flow와 비슷한 흐름이라고 볼 수 있다.

```
# In a CMake project
$ mkdir build
$ cd build
$ cmake ..
$ make
$ make install # (optionally)
```

#### 5. Understanding ROS Nodes

- Overview of Graph Concepts

- Nodes : An executable that uses ROS to communicate with other nodes.
  - 계산을 수행하는 프로세스. 노드들은 그래프에 결합되고 다른 노드들과 통신한다.
  - 통신에는 Streaming Topics, RPC Services, Parameter Server를 사용한다.
  - 실행중인 모든 노드는 나머지 시스템 노드들과 고유하게 식별될 수 있는 Graph Resource Name이 있다.
  - 또한 노드는 Node Type을 가지는데, 노드의 패키지 이름과 노드의 실행파일 이름 그리고 Package Resource Name이다.
  - ROS는 이름을 통해서 모든 executable을 찾고 가장 먼저 찾아진 것을 고르기 때문에, 같은 이름의 다른 실행파일을 만들지 않도록 주의해야 한다.
  - Command-line Remapping Arguments  
ROS의 강력한 기능중 하나로, 같은 노드를 많은 환경설정으로 실행할 수 있다.

```
rosrun rospy_tutorials talker chatter:=wg/chatter
```

이렇게 실행을 하게 되면, talker에게 chatter대신 /wg/chatter에게 publish하도록 Remapping 할 수 있다.  
Remapping되는 예시는 다음과 같다.

| Remapping Argument      | Node Namespace | Matching Names | Final Resolved Name |
|-------------------------|----------------|----------------|---------------------|
| <u>foo:=bar</u>         | /              | foo, /foo      | /bar                |
| <u>foo:=bar</u>         | /baz           | foo, /baz/foo  | /baz/bar            |
| <u>/foo:=bar</u>        | /              | foo, /foo      | /bar                |
| <u>/foo:=bar</u>        | /baz           | /foo           | /baz/bar            |
| <u>/foo:=/a/b/c/bar</u> | /baz           | /foo           | /a/b/c/bar          |

- 그 외의 자세한 내용들은 [여기](#)를 참조.
- Messages : ROS data type used when subscribing or publishing to a topic.
  - Node들은 message를 topic에 publish해서 각 노드들끼리 통신한다.
  - message는 지정된 field들로 구성된 간단한 data structure이다.
  - Standard primitive types(integer, floating point, boolean, etc..)가 지원되며, 구조체와 같은 primitive나 배열도 포함될 수 있다.
  - 또한 ROS Service Call을 통해서 Request와 Response를 주고받을 수 있는데, 이러한 Request와 Response Message는 [srv file](#)로 정의됨.
  - msg files : message의 data structure를 명시하기 위한 simple text file. package의 msg라는 subdirectory에 저장된다. 또한 노드는 메시지 유형과 MD5 sum이 일치해야만 통신할 수 있다.
  - Message Types : Standard ROS [naming](#)을 사용하는데, "패키지이름/메시지파일이름" 이러한 형식이다.
  - Building : CMakeList.txt 파일 안에서 rosbuild\_genmsg()를 통해서 가능.
  - 그 외의 자세한 내용들은 [여기](#)를 참조.
- Topics : Nodes can publish messages to a topic as well as subscribe to a topic to receive messages.
  - 노드들이 메시지를 교환하는 이름이 있는 버스라고 생각하면 됨.
  - 정보의 production과 consumption을 분리하여 익명의 publish/subscribe를 가짐.
  - 따라서 노드들은 각각 누구와 통신하는지 모름. 대신에 관심이 있는 정보에 대해서 **Subscribe**하거나 **Publish**하는 것.
  - 이러한 특성에 따라서 토픽에 다수의 publisher/subscriber가 존재할 수 있음.
  - Topics은 그래서 undirectional하고, Streaming Communication이다.
  - Remote Procedure Call을 수행해야 하는 Node는(즉 요청에 응답해야하는 노드) 대신 Services를 사용해야 함.
  - 적은 양의 상태를 유지하는 Parameter Server가 존재함.

- 각각의 Topic은 해당 Topic에 publish하는데 사용되는 ROS Message Type에 의해서 type이 지정되며 각 노드는 일치하는 type의 메시지만 받을 수 있음. 이 과정에서, 모든 ROS Client는 MD5를 계산하고 일치하는지도 확인하여 일관된 코드 베이스에서 컴파일 되었는지도 체크함.
- Master 노드(아래 참조)는 type consistency에 제약을 받지 않는데, subscriber는 type이 매치되기 전까지는 message 전송을 하지 않음.
- TCP/IP기반으로 전송함. UDP는 현재 roscpp만 지원하니까 패스함.
- 그 외의 자세한 내용들은 [여기](#)를 참조.
- Master : Name service for ROS (i.e. helps nodes find each other)
  - ROS Master는 ROS system의 node들의 naming과 registration을 할 수 있도록 한다.
  - Topic과 Service에 대한 Publisher/Subscriber를 추적함.
  - Master의 역할은 ROS의 개별 노드들이 서로를 찾을 수 있도록 하는 것임.
  - 이러한 노드들이 서로를 찾으면 서로 peer-to-peer communication을 함.
  - Master가 Parameter Server를 제공함.
  - roscore 명령어로 다른 필수적인 components와 함께 실행됨.
  - naming 및 그외의 자세한 내용들은 [여기](#)를 참조.
- rosout : ROS equivalent of stdout/stderr
- roscore : Master + rosout + parameter server (parameter server will be introduced later)
- Node들은 Service를 사용하거나 제공할 수 있다.
- rosrun을 통해서 노드 생성하기

#### 1. roscore로 master 노드 생성 및 ROS 실행

```
roscore
```

```
roscore http://CLDVK-UBUNTU:11311/
cleverdevk@CLDVK-UBUNTU:~$ roscore
... logging to /home/cleverdevk/.ros/log/a8747d6a-da21-11e9-b9ca-68ecc58f6402/ro
slaunch-CLDVK-UBUNTU-4486.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://CLDVK-UBUNTU:35179/
ros_comm version 1.12.14

SUMMARY
=====

PARAMETERS
  * /rosdistro: kinetic
  * /rosversion: 1.12.14

NODES

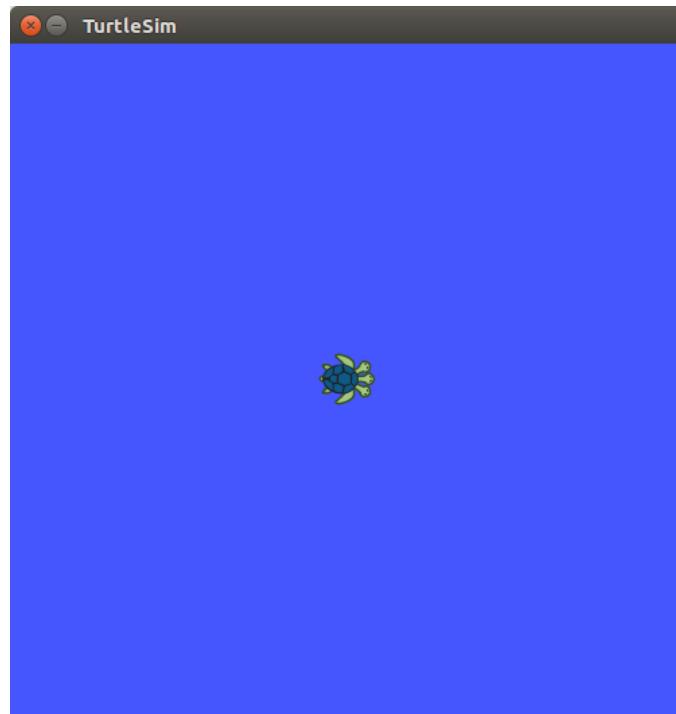
auto-starting new master
process[master]: started with pid [4496]
ROS_MASTER_URI=http://CLDVK-UBUNTU:11311/

setting /run_id to a8747d6a-da21-11e9-b9ca-68ecc58f6402
process[rosout-1]: started with pid [4509]
started core service [/rosout]
```

#### 2. 새로운 터미널을 열고 rosrun으로 거북이 실행하기

- rosrun '패키지이름' '노드이름' 형식.

```
rosrun turtlesim turtlesim_node
```



3. 새로운 터미널을 열고 노드이름을 다르게 거북이 실행하기

```
rosrun turtlesim turtlesim_node __name:=my_turtle
```

4. 새로운 터미널을 열고 현재 노드 확인하기.

```
rosvn node list
```

```
cleverdevk@CLDVK-UBUNTU:~$ rosvn node list
/myturtle
/rosout
/turtlesim
cleverdevk@CLDVK-UBUNTU:~$
```

- 두 개의 노드가 추가된 것을 알 수 있다.

## 6. Understanding ROS Topics

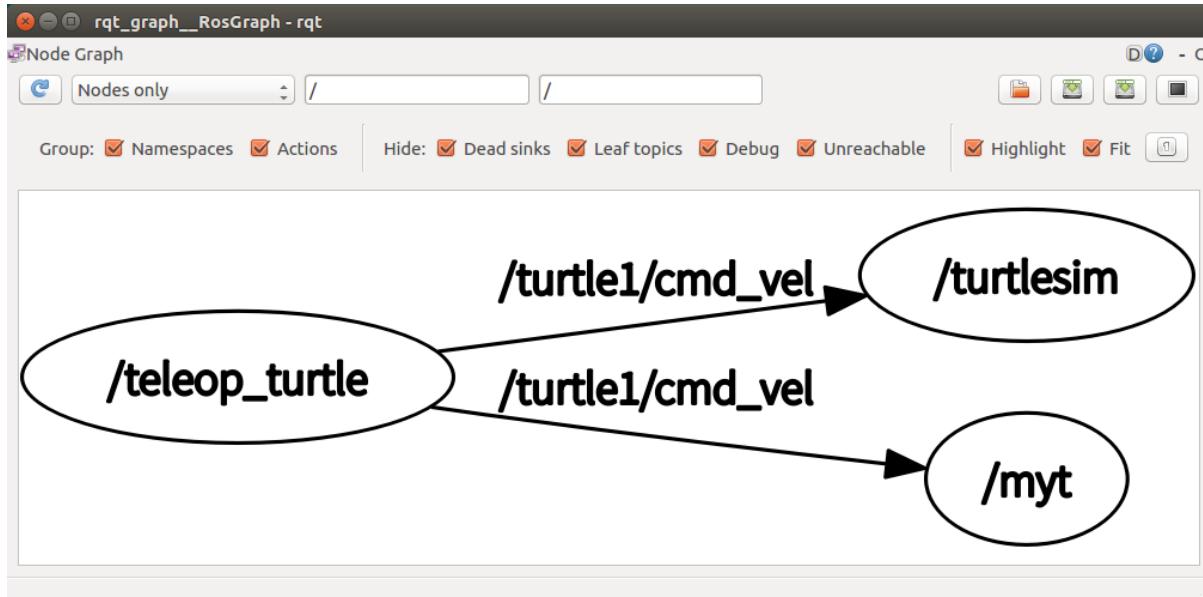
- Turtlesim Teleoperation : 거북이를 키보드 방향키로 움직이도록 할 수 있는 패키지

```
rosrun turtlesim turtle_teleop_key
```

를 실행하고 그 터미널에서 방향키를 누르면 거북이가 움직인다.

왜냐하면 turtle\_teleop\_key가 turtle1/cmd\_vel이라는 Topic에 Publish하고 turtlesim\_node가 Subscribe하기 때문에 가능하다.  
현재 Node 및 Topic에 관한 그래프는 rqt\_graph로 확인 가능하다.

```
#설치가 안되어 있다면  
sudo apt-get install ros-kinetic-rqt  
sudo apt-get install ros-kinetic-rqt-common-plugins  
  
#그래프 확인은 새로운 터미널을 열고  
rosrun rqt_graph rqt_graph
```



- rostopic 명령어 : rostopic tool은 ROS topic에 대한 정보들을 얻을 수 있는 명령어
  - rostopic -h : help option
  - rostopic echo : 해당 토픽에 publish되는 데이터를 볼 수 있다.

```
rostopic echo /turtle1/cmd_vel
```

rostopic echo를 켜 상태로 거북이를 움직이면, 메시지가 기록된다.

```
cleverdevk@CLDVK-UBUNTU: ~
/turtle1/cmd_vel      /turtle1/color_sensor
cleverdevk@CLDVK-UBUNTU:~$ rostopic echo /turtle1/cmd_vel
linear:
  x: 0.0
  y: 0.0
  z: 0.0
angular:
  x: 0.0
  y: 0.0
  z: -2.0
---
linear:
  x: -2.0
  y: 0.0
  z: 0.0
angular:
  x: 0.0
  y: 0.0
  z: 0.0
---
linear:
  x: 0.0
  y: 0.0
  z: 0.0
```

- ROS Messages
  - rostopic type [topic] : 토픽의 type 확인
  - rosmg show [type of topic] : message의 detail 확인

```
rostopic type /turtle1/cmd_vel
rosmg show geometry_msgs/Twist
```

```
cleverdevk@CLDVK-UBUNTU: ~
cleverdevk@CLDVK-UBUNTU:~$ rostopic type /turtle1/cmd_vel
geometry_msgs/Twist
cleverdevk@CLDVK-UBUNTU:~$ rosmg show geometry_msgs/Twist
geometry_msgs/Vector3 linear
  float64 x
  float64 y
  float64 z
geometry_msgs/Vector3 angular
  float64 x
  float64 y
  float64 z
```

- rostopic pub [topic] [msg\_type] [args] : currently advertised topic에 data를 publish할 수 있음.

```
rostopic pub -1 /turtle1/cmd_vel geometry_msgs/Twist -- '[2.0, 0.0, 0.0]' '[0.0, 0.0, 1.8]'
```

-1 option : 오직 한개의 message만 publish하고 exit

-- : 다음 인자가 옵션이 아님을 알려주는 것.

두개의 coordination은 linear와 angular에 대한 인자이다.

그 외의 자세한 내용은 [여기](#)를 참조.

## 7. Understanding ROS Services and Parameters

- ROS Service : Node들이 서로 통신할 수 있는 또 다른 방법. Services는 request를 보내고 response를 받을 수 있도록 해줌.

```
rosservice list      print information about active services
rosservice call      call the service with the provided args
rosservice type       print service type
rosservice find        find services by service type
rosservice uri        print service ROSRPC uri
```

- rosservice list : 현재 활성화된 노드들의 service list를 보여줌.  
해당 service call이 실행된다.

```
rosservice list
```

```
#output
/clear
/kill
/reset
/rosout/get_loggers
/rosout/set_logger_level
/spawn
/teleop_turtle/get_loggers
/teleop_turtle/set_logger_level
/turtle1/set_pen
/turtle1/teleport_absolute
/turtle1/teleport_relative
/turtlesim/get_loggers
/turtlesim/set_logger_level
```

- rosservice type

```
rosservice type /clear
```

- rosservice call : service call하는데,

```
#형식
rosservice call [service] [args]

#/clear는 args가 없기 때문에
rosservice call /clear

#파이프를 사용하여 메시지 타입을 받아서 rossrv show에 인자로 전달하면 바로 argument를 확인할 수 있다.
rosservice type /spawn | rossrv show
```

```

cleverdevk@CLDVK-UBUNTU: ~
/turtle1/set_pen
/turtle1/teleport_absolute
/turtle1/teleport_relative
/turtlesim/get_loggers
/turtlesim/set_logger_level
cleverdevk@CLDVK-UBUNTU:~$ rosservice type /spawn
turtlesim/Spawn
cleverdevk@CLDVK-UBUNTU:~$ rosservice type /spawn | rossrv show
float32 x
float32 y
float32 theta
string name
---
string name

cleverdevk@CLDVK-UBUNTU:~$ rosservice type /spawn | rossrv show
float32 x
float32 y
float32 theta
string name
---
string name

cleverdevk@CLDVK-UBUNTU:~$ 

```

해당 argument에 맞춰 call과 같이 전달하면,

```

rosservice call /spawn 2 2 0.2 "helloturtle"
#terminal return
name : helloturtle

```



- ROS Param(rosparam) : ROS Parameter Server에 데이터를 저장 수정할 수 있게 해 주는 명령어. Parameter Server에는 integer, float, boolean, dictionary, list들을 저장할 수 있으며, YAML markup language를 문법으로 사용한다.

```

rosparam set      set parameter
rosparam get      get parameter
rosparam load    load parameters from file
rosparam dump    dump parameters to file
rosparam delete  delete parameter
rosparam list    list parameter names

```

- `rosparam list` : 현재 parameter리스트를 보여줌.

```
rosparam list

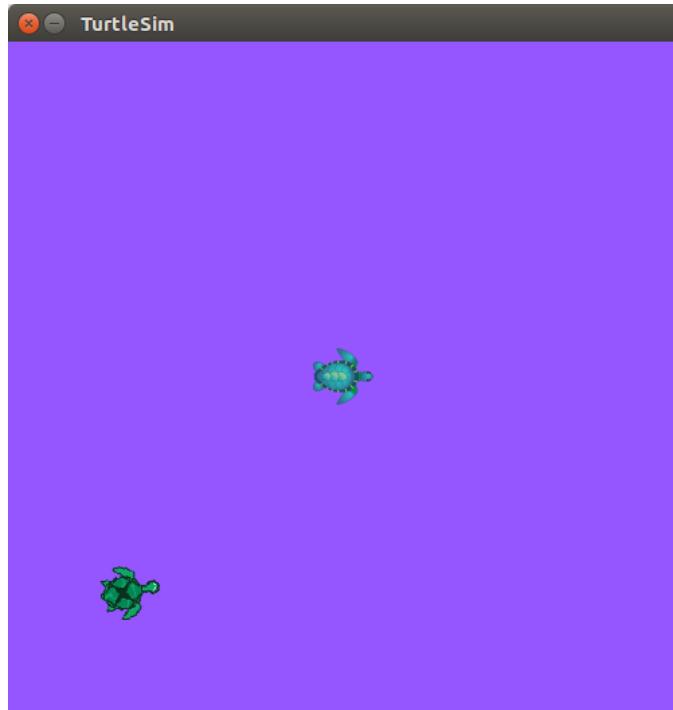
#output
/background_b
/background_g
/background_r
/rosdistro
/roslaunch_uris/host_57aea0986fef__34309
/rosversion
/run_id
```

- `rosparam set / get` : 해당 파라미터 값을 조정 / 확인.

```
#형식
rosparam set [param_name] [value]
rosparam get [param_name]

#/로 get을 하면 모든 parameter를 확인 할 수 있다.
rosparam get /

#값 변경
rosparam set /background_r 150
#새로고침을 위해
rosservice call /clear
```



- `rosparam dump / load` : 파라미터 값을 저장 / 로드

```
#형식
rosparam dump [file_name] [namespace]
rosparam load [file_name] [namespace]

#param.yaml파일에 현재 파라미터 저장
rosparam dump params.yaml

#params.yaml에 있는 값을 copy namespace에 저장.
rosparam load params.yaml copy
rosparam get /copy/background_b
```

