Building a ROS Package

Toolchain to build a package

## **1.Building Packages**

As long as all of the system dependencies of your package are installed, we can now build your new package.

**Note:** If you installed ROS using apt or some other package manager, you should already have all of your dependencies.

Before continuing remember to source your environment setup file if you have not already. On Ubuntu it would be something like this:

# source /opt/ros/%YOUR\_ROS\_DISTRO%/setup.bash

$ source /opt/ros/kinetic/setup.bash # For Kinetic for instance

### **1.1 Using catkin\_make**

[catkin\_make](http://wiki.ros.org/catkin/commands/catkin_make) is a command line tool which adds some convenience to the standard catkin workflow. You can imagine that [catkin\_make](http://wiki.ros.org/catkin/commands/catkin_make) combines the calls to cmake and make in the standard CMake workflow.

Usage:

# In a catkin workspace

$ catkin\_make [make\_targets] [-DCMAKE\_VARIABLES=...]

For people who are unfamiliar with the standard CMake workflow, it breaks down as follows:

**Note:** If you run the below commands it will not work, as this is just an example of how CMake generally works.

# In a CMake project

$ mkdir build

$ cd build

$ cmake ..

$ make

$ make install # (optionally)

This process is run for each CMake project. In contrast catkin projects can be built together in workspaces. Building zero to many catkin packages in a workspace follows this work flow:

# In a catkin workspace

$ catkin\_make

$ catkin\_make install # (optionally)

The above commands will build any catkin projects found in the src folder. This follows the recommendations set by [REP128](http://ros.org/reps/rep-0128.html). If your source code is in a different place, say my\_src then you would call catkin\_make like this:

**Note:** If you run the below commands it will not work, as the directory my\_src does not exist.

# In a catkin workspace

$ catkin\_make --source my\_src

$ catkin\_make install --source my\_src # (optionally)

For more advanced uses of [catkin\_make](http://wiki.ros.org/catkin/commands/catkin_make) see the documentation: [catkin/commands/catkin\_make](http://wiki.ros.org/catkin/commands/catkin_make)

### **Building Your Package**

If you are using this page to build your own code, please also take a look at the later tutorials [(C++)](http://wiki.ros.org/ROS/Tutorials/WritingPublisherSubscriber%28c%2B%2B%29)/[(Python)](http://wiki.ros.org/ROS/Tutorials/WritingPublisherSubscriber%28python%29) since you may need to modify CMakeLists.txt.

You should already have a [catkin workspace](http://wiki.ros.org/catkin/Tutorials/create_a_workspace) and a new catkin package called beginner\_tutorials from the previous tutorial, [Creating a Package](http://wiki.ros.org/ROS/Tutorials/CreatingPackage). Go into the catkin workspace if you are not already there and look in the src folder:

$ cd ~/catkin\_ws/

$ ls src

* beginner\_tutorials/ CMakeLists.txt@

You should see that there is a folder called beginner\_tutorials which you created with [catkin\_create\_pkg](http://wiki.ros.org/catkin/commands/catkin_create_pkg) in the previous tutorial. We can now build that package using [catkin\_make](http://wiki.ros.org/catkin/commands/catkin_make):

$ catkin\_make

You should see a lot of output from cmake and then make, which should be similar to this:

* Base path: /home/user/catkin\_ws
* Source space: /home/user/catkin\_ws/src
* Build space: /home/user/catkin\_ws/build
* Devel space: /home/user/catkin\_ws/devel
* Install space: /home/user/catkin\_ws/install
* ####
* #### Running command: "cmake /home/user/catkin\_ws/src
* -DCATKIN\_DEVEL\_PREFIX=/home/user/catkin\_ws/devel
* -DCMAKE\_INSTALL\_PREFIX=/home/user/catkin\_ws/install" in "/home/user/catkin\_ws/build"
* ####
* -- The C compiler identification is GNU 4.2.1
* -- The CXX compiler identification is Clang 4.0.0
* -- Checking whether C compiler has -isysroot
* -- Checking whether C compiler has -isysroot - yes
* -- Checking whether C compiler supports OSX deployment target flag
* -- Checking whether C compiler supports OSX deployment target flag - yes
* -- Check for working C compiler: /usr/bin/gcc
* -- Check for working C compiler: /usr/bin/gcc -- works
* -- Detecting C compiler ABI info
* -- Detecting C compiler ABI info - done
* -- Check for working CXX compiler: /usr/bin/c++
* -- Check for working CXX compiler: /usr/bin/c++ -- works
* -- Detecting CXX compiler ABI info
* -- Detecting CXX compiler ABI info - done
* -- Using CATKIN\_DEVEL\_PREFIX: /tmp/catkin\_ws/devel
* -- Using CMAKE\_PREFIX\_PATH: /opt/ros/kinetic
* -- This workspace overlays: /opt/ros/kinetic
* -- Found PythonInterp: /usr/bin/python (found version "2.7.1")
* -- Found PY\_em: /usr/lib/python2.7/dist-packages/em.pyc
* -- Found gtest: gtests will be built
* -- catkin 0.5.51
* -- BUILD\_SHARED\_LIBS is on
* -- ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
* -- ~~ traversing packages in topological order:
* -- ~~ - beginner\_tutorials
* -- ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
* -- +++ add\_subdirectory(beginner\_tutorials)
* -- Configuring done
* -- Generating done
* -- Build files have been written to: /home/user/catkin\_ws/build
* ####
* #### Running command: "make -j4" in "/home/user/catkin\_ws/build"
* ####

Note that [catkin\_make](http://wiki.ros.org/catkin/commands/catkin_make) first displays what paths it is using for each of the 'spaces'. The spaces are described in the [REP128](http://ros.org/reps/rep-0128.html) and by documentation about catkin workspaces on the wiki: [catkin/workspaces](http://wiki.ros.org/catkin/workspaces). The important thing to notice is that because of these default values several folders have been created in your catkin workspace. Take a look with ls:

$ ls

* build
* devel
* src

The build folder is the default location of the [build space](http://wiki.ros.org/catkin/workspaces#Build_Space) and is where cmake and make are called to configure and build your packages. The devel folder is the default location of the [devel space](http://wiki.ros.org/catkin/workspaces#Development_.28Devel.29_Space), which is where your executables and libraries go before you install your packages.