Test the ROS installation:

`printenv | grep ROS` → should output a list of the ROS enviroment variables.

Create a ROS workspace:

`mkdir -p ~/workspace_name/src` — The -p paramater creates as many parent directories as needed.
`catkin_make ~/workspace_name/` → sets up the catkin build sistem.
`source ~/workspace_name/devel/setup.bash` → overlay the workspace on top of the enviroment.
`echo $ROS_PACKAGE_PATH` → checks the workspace setup, should output its path in the enviroment. outputs all packages on the ROS enviroment.

ROS FileSystem

- Packages: sw organization unit of ROS code. (libraries, executables, scripts, artifacts)
- Manifests: describes a package content (dependencies between packages and meta info)

The ROS FileSystem tools will only work along those packages whose directories have been overlayed on top of the ROS environment. If not it'll be like if they doesn exist.

rospack : retrieves package information.
`rospack info package_name` the path of the package.

rosed : change direcarie
`rosed locationname/subdir`
`rosed roscpp` → a custom version of cpp for ROS
`rosed log` → the log for all current ROS programs.

rosls : list the files in a ROS package. (if the path provided is wrong it will ls your current direcarie)

catkin Packages

- package.xml

```
<package format="2">
  <name> package_name </name>
  <version> xx.xx.xx </version>
  <description>
    The package content and purpose
  </description>
  <maintainer email="juancasado@mrblissjulgrin.com"> Juan Casado </maintainer>
  <license> BSD </license>
  <url> http://www.mrblissjulgrin.com </url>
  <author> Juan Casado </author>
</package>
```

Dependencies on .xml (xml dedares dependencies but does not create them) //CMake contents

<depend> (buid, export and execution dependence)
<buildtool_depend> (they are needet to perform the building)
<build_depend> (they are needed at build time) Libraries and exported headers find_package()-ed
<build_export_depend> (they are needed to build libraries) catkin_package()
<exec_depend> (needed to run)
<test_depend> (needed to test)
<doc_depend> (needed to generate documentation)

e.g.

```
<buidtool_depend> catkin </buildtool_depend>
<depend> roscpp </depend>
<depend> std_msgs </depend>
```

<build_depend> message_generation </build_depend>
<exec_depend> message_runtime </exec_depend>
<exec_depend> raspy </exec_depend>
<test_depend> python-mock </test_depend>
<doc_depend> doxygen </doc_depend>

Metapackages: group multiple packages into a single logical one.
They must contain:

- In the xml file:

```
<export>
  <metapackage />
</export>
```

- In the CMakeLists.txt file

```
cmake_minimum_required (VERSION 2.8.3)
project (package_name)
find_package (catkin REQUIRED)
catkin_metapackage ()
```

The metapackages can only depend on execution with other packages of the group and have a buildtool_depend on catkin

- CMakeList.txt with user catkin

It describes how to build and where to install the package.

```
cmake_minimum_required (VERSION 2.8.3)
project (package_name)        → Later inside the CMakeList.txt you can use ${PROJECT_NAME}
find_package (catkin REQUIRED)
// if others e.g. find_package (nodelet REQUIRED)
```

for every find_package some variables are created:

```
PACKAGE_NAME_FOUND      → True if package is found
PACKAGE_NAME_INCLUDES   → include paths exported by the package
PACKAGE_NAME_LIBS       → libraries exported by the package
PACKAGE_NAME_DEFINITIONS
```

you can merge various packages and the content of their variables by:

```
find_package (catkin REQUIRED COMPONENTS nodelet)
find_package (Boost REQUIRED COMPONENTS thread)
catkin_python_setup()
```

you need to have an aditional setup.py file.

```
add_message_files()     .msg
add_service_files()     .srv
add_action_files()      .action
generate_messages()
catkin_package ()  declares build specific information
  INCLUDE_DIRS      exported include paths
  LIBRARIES         exported libraries from the project
  CATKIN_DEPENDS    other catkin project that the current depends on
  DEPENDS           we depend on them but not through catkin → the usually haven't be "find_packaged"
  CFG_EXTRAS
```

e.g.

```
catkin_package (
  INCLUDE_DIRS include              // There is an include folder where exported headers go
  LIBRARIES ${PROJECT_NAME}
  CATKIN_DEPENDS roscpp nodelet // Packages that we depend on
  DEPENDS eing opencv )             // System dependencies that we depend on
set_target_properties()   → used to rename targets internally  PROPERTIES OUTPUT_NAME
                          → used to change the output directory PROPERTIES LIBRARY_OUTPUT_DIRECTORY
include_directories()
```

you need to include all your *_INCLUDE_DIRS variables.

```
include_directories( include ${Boost_INCLUDE_DIRS} ${catkin_INCLUDE_DIRS})
link_directories()
    can add aditional library path but it is not recommended to use it.
add_library( ${PROJECT_NAME} ${${PROJECT_NAME}_SRCS})
add_executable ( executable_name  src/file1.cpp  src/file2.cpp)
target_link_libraries ( target  moo)  ---> The librarie sould be libmoo.so
if (CATKIN_ENABLE_TESTING)
  catkin_add_gtest( test  file.cpp)
endif()
install ( TARGETS ${PROJECT_NAME}
  ARCHIVE  DESTINATION  ${CATKIN_PACKAGE_LIB_DESTINATION}    DLL en windows stubs
  LIBRARY  DESTINATION  ${CATKIN_PACKAGE_LIB_DESTINATION}    non DLL
  RUNTIME  DESTINATION  ${CATKIN_PACKAGE_BIN_DESTINATION}    exe and DLL libraries
)
catkin_install_python ( PROGRAMS  scripts/myscript
  DESTINATION ${CATKIN_PACKAGE_BIN_DESTINATION} )
install (DIRECTORY  include/${PROJECT_NAME}/
    DESTINATION  ${CATKIN_PACKAGE_INCLUDE_DESTINATION}
    PATTERN  ".svn" EXCLUDE
)
install (DIRECTORY  include/
    DESTINATION  ${CATKIN_GLOBAL_INCLUDE_DESTINATION}
    PATTERN  ".svn" EXCLUDE
)
install (DIRECTORY  roslaunch
    DESTINATION  ${CATKIN_PACKAGE_SHARE_DESTINATION}/launch
    PATTERN  ".svn" EXCLUDE
)
```

- Package folder

  Each packeg needs to have its own folder and tree of folders

```
package/                    workspace/
  CMakeList.txt      =>       src/
  package.xml                   CMakeList.txt
                                package1/
                                  CMakeList.txt
                                  package.xml
                                package2/
                                  CMakeList.txt
                                  package.xml
                                  ...
                              build/   ---> the build when building came here
                              devel/   ---> default location for executables an libraries before
                                            installing packages.
```

Building a ROS package

To create a package inside a workspace:

```
catkin_create_package  name  <dependencies>...
```

To build packages inside a workspace:

```
catkin_make
```

Add the workspace to the ROS eviroment

```
. ~/workspace/devel/setup.bash
```

previously after istallation we should have done:

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

View package dependencies

rospack depends1 package_name

if you aply this same command to any element of the dependencies list a new list of its own dependencies will be printed. All recursive dependencies can be printed with:

rospack depends package_name