Catkin-Pip.

Release 0.2.3

AlexV

Aug 27, 2018

Contents

This ROS package allows using pure python package in ROS Systems, from catkin workspaces.

It was born from the need to interact with ROs from a pure python environment, along with pyros. Therefore the multiple workflows enabled by catkin_pip are workflows used by pyros and its dependencies.

CHAPTER 1

Overview

1.1 catkin_pip

Provides catkin extension (cmake hooks) to work with pure python packages in catkin workspaces.

Because state of the art python (ref. http://jeffknupp.com/blog/2013/08/16/open-sourcing-a-python-project-the-right-way/) should be allowed to work with catkin.

catkin_pip allows you to use your own package as a normal python package, with python workflow (example using virtualenvwrapper):

```
$ mkvirtualenv my_package_venv --system-site-packages
(my_package_venv)$ pip install -r requirements.txt
(my_package_venv)$ python -m my_package
(my_package_venv)$ nosetests my_package
(my_package_venv)$ deactivate
$
```

OR using the python workflow from inside a catkin workspace:

```
$ source /opt/ros/indigo/setup.bash
$ cd existing_catkin_ws
$ catkin_make
$ source devel/setup.bash
$ python -m my_package
$ nosetests my_package
```

TODO : improve this with real simple command line examples, copied verbatim.

It basically make use, through cmake, of the workspace as a virtual env would be used in a python flow.
Mostly it's just a few arguments added to pip to get it to install packages in the correct way in a workspace.

The provided cmake macros are:

- catkin_pip_setup()
- catkin_pip_requirements(requirements_file)
- catkin_pip_package()

they can be used like this :

```
...
cmake_minimum_required(VERSION 2.8.3)
project(my_project)

find_package(catkin REQUIRED COMPONENTS
    catkin_pip
)

# Getting pip requirements for catkin_pip itself
catkin_pip_setup()

# We need to install the project pip dependencies in the devel workspace being created
catkin_pip_requirements(${CMAKE_CURRENT_SOURCE_DIR}/requirements.txt)

# defining current package as a package that should be managed by pip (not catkin -␣
↪even though we make it usable with workspaces)
catkin_pip_package()

# Corresponding install rules are also setup by each of these macros.
...
```

As a result you can:

- Use pip/setup.py dependency mechanism with any python git repo, for devel workspace.
- Use pip dependency requirements mechanism with any pip dependency, for devel workspace.
- Directly work with python package sources in your usual catkin workspaces (devel only), after just adding a *CMakeLists.txt* and a *package.xml* files.
- Work with "hybrid" packages that can be released both via pip packages and ros packages, provided the proper dependencies exists for both package management systems (Note rosdep support of pip is not clear…).
- Do a Third Party release of an existing python package into a ROS package (no setup.py changes required).

1.2 Catkin-Pip Overview

This document describe the different areas where catkin-pip makes Python ROS development easier.

1.2.1 Pip package

Catkin-pip doesnt not change the behavior of a pip package. It just make its use easier in a catkin context.

Catkin-pip however suppose that the pip package needs to use latest python tools (setuptools, pip, pytest, nose) for manipulating it. This is in line with python expectation to use latest stable software (and not obsolete system dependencies).

1.2.2 ROS Package

Catkin-pip doesnt not change the behavior of a ROS package. It just make it easier to build one from a recent python package.

As ROS packages should only depend on other ros packages, one should be careful to have verified the package still behave as expected when not using pip dependencies before releasing.

1.2.3 Devel Space

When used to build a devel space, catkin-pip retrieves it s own dependencies (latest python tools) in the build folder. It also plug into the cmake flow to trigger pip download of the package dependencies (unless CATKIN_PIP_NO_DEPS is enabled)

After sourcing the devel space, the catkin-pip python tools are overlayed on top of the system ones, and the package dependencies also overlay the system version of it, if there are any. this makes it simple to use any version you would like and manage your dependencies with pip, as usual in python workflow.

With CATKIN_PIP_NO_DEPS enabled, only the catkin-pip python tools are retrieved by pip, and your package needs to rely on the system installed python dependencies. This is done in order for package developer to be able to slowly migrate from pip version to system version, so that they are eventually able to create a system package from their pip package.

1.2.4 Install Space

When used to create an install space, catkin-pip doesnt do anything special, except using the latest setuptools version to create the layout of that system package. No pip dependencies are ever loaded in that system, so using the install space requires your package to be able to run with system python dependencies, just as running from a system package would do.

When in that step, one should be careful that tools used to manipulate the package (test, docs) might be at a different version now (system) than in the devel space (catkin-pip version)

Creating a ROs package from the install space is done as usual, only using a recent version of setuptools.

1.2.5 Virtual environments

Since Catkin-pip allows normal python package to “merge” easily with a ROS package, into what we call a “hybrid” package, one can now choose to use either : - a ROS package, installed on the system, in the ros environment. - a pip package, installed on a virtual env, with access to the system packages.

Note that catkin-pip only solve the “build part” of the problem:

- To dynamically use the python packages from your catkin workspace when launching a python program outside a ROS environment , you should check pyros-setup.
- To dynamically discover/marshall/serialize messages format from your ROS processes, you should check pyros.

1.3 Catkin-Pip Notes

This documents gather various notes and detailed issues with Catkin-Pip

1.3.1 Python virtualenvs, imports /vs/ ROS workspaces

It is important to keep in mind that the way Python imports works, with or without virtualenv, is not compatible with the way ROS workspaces work.

Python Virtual envs allow only 2 layers, ie. you cannot overlay a venv on top of another venv :

- system python
- python in virtualenv

ROS workspaces allow many layers, by using CMAKE_PREFIX_PATH when building, and relying on the user to do *source setup.sh* before usage :

- system ROS
- ROS workspace1
- ROS workspace2
- etc.

Python imports behavior depends on sys.path. This is managed by the python site module. Python uses PYTHONPATH as a "last measure" of *temporarily* setting a site directory (usually called site-packages or dist-packages on debian), or any path containing python packages and modules. Python adds a directory to the sys.path only if it exists, and if it is not there yet. This behavior is relied upon to decide which packages to import and from where (1st PYTHONPATH paths, 2nd any dynamically discovered package (pth files, etc.), 3rd standard python directories on your system)

ROS uses PYTHONPATH as the "first and only way" to *permanently* (from ROS point of view) set a list of workspaces The PYTHONPATH is setup to contain the ROS workspaces paths in the appropriate order.

=> Using any python import feature might break this order that ROS relies on to have this workspace hierarchy working properly.

So the following behavior has been decided for Catkin-Pip : - Catkin-Pip adds extra site directories to the PYTHONPATH (to make things as obvious as possible for a ROS user) - Catkin-Pip does NOT do any dynamic discoveries (reading easy-install.pth file to discover editable packages). Better keep things simple and not reimplement a python behavior in a shell script… Because of this, **it can happen that a package PKG installed in an underlay will be imported before a catkin-pip'ed editable package PKG during development**. The expected workaround, is to **use pyros_setup in that package**. It will dynamically reset the desired path order in sys.path, following its user-modifiable configuration configuration file.

More work can be done here, to make things more obvious (like add a field to a catkin-pip'ed package and use pkg_resources on import to find the correct one for example)…

1.4 Changelog for package catkin_pip

1.4.1 0.2.3 (2017-08-11)

- Merge pull request #147 from pyros-dev/lunar adding lunar

- adding –ignore-src when calling rosdep on tests, to not attempt to download an old (or missing) version of catkin_pip.
- adding lunar
- Merge pull request #144 from pyros-dev/fix_destinations fixing catkin_destination not being called
- tests are now using the new catkin_pip_target and calling catkin_package directly.
- adding catkin_pip_target to API, to allow the user to call catkin_package how he wants.
- splitting catkin_pip_package in function and macro to expose the catkin_destinations variables set in the scope.
- adding install rules to verify catkin variables.
- Merge pull request #128 from pyros-dev/pyup-update-pytest-3.0.6-to-3.1.3 Update pytest to 3.1.3
- Update pytest from 3.0.6 to 3.1.3
- Contributors: AlexV, pyup-bot

1.4.2 0.2.2 (2017-05-30)

- Merge pull request #123 from yotabits/devel Added option NOSE_OPT to catkin_add_nosetests func
- Added option NOSE_OPT to catkin_add_nosetests func In order to use some specific option for nosetests we now have a NOSE_OPT parameter that allow to use some customs options for launching nosetests The same has been done for pytests with the param PYTEST_OPT
- Contributors: AlexV, Thomas

1.4.3 0.2.1 (2017-05-11)

- Merge pull request #115 from asmodehn/distutils Implementing distutils support
- refining tests on install flow, to confirm which test framework is used. upgrading catkin package format to advised format 2.
- fixes to support distutils as well as setuptools.
- adding setuptools_setup test project and tests to validate package structure after installation.
- adding test to make sure “make install” does not trigger errors.
- adding a basic package to test distutils based setup.py
- Contributors: AlexV, yotabits

1.4.4 0.2.0 (2017-03-17)

- removing dependency on daemontools (not available on fedora)
- setlock -> flock
- Merge pull request #95 from k-okada/install_data data should not install right under the CMAKE_INSTALL_PREFIX
- fixing syntax for ros distro autodetection when building this package.
- Merge pull request #97 from asmodehn/gopher-devel Preparing next release 0.2
- install data

- Contributors: AlexV, Kei Okada, alexv
- Merge branch 'devel' into gopher-devel
- now using catkin_pip_runcmd to retrieve cookiecutter package samples.
- removing install envhook. seems to break things. we dont want the install envhook to be created for catkin_pip since the target build env does not exist.
- Merge pull request #61 from asmodehn/refactor_envs Refactor envs
- adding docs to make decision clear regarding catkin / pyros_setup behavior.
- fixing up install devel command since we do not pass editable pkg path into pythonpath anymore.
- Merge branch 'gopher-devel' into refactor_envs
- Merge pull request #76 from asmodehn/easyinstall_fix Easyinstall fix
- commenting install tests… we dont have any.
- always adding the catkin_pip_env path to pythonpath to be able to find basic tools from the first time around.
- envhooks now not adding to pythonpath from pth files. too confusing. envhooks now not adding non existent paths to pythonpath. maybe good idea to match site module behavior. fixing tests.
- improved tests. added test for check with pyros_setup and interractive debugging
- fixed broken tests. now needs pyros-setup to get .pth paths before workspaces. cosmetics.
- improved tests and fix all. added actual test for pytest to pipproject.
- cleaning up command args when running pip install –editable to clean python path and workaround pip bug…
- replacing CI_ROS_DISTRO var by ROS_DISTRO, to have it set for calls to rosdep.
- more exhaustive test for sys_path
- commenting echoes from script since it cannot echo without breaking things anyway.
- reviewing shell script to make them a tiny more robust…
- removed useless bash envhook, fixed easy-install parse to include last line even if no empty line at end of file.
- Merge pull request #77 from asmodehn/pyup-update-pytest-3.0.5-to-3.0.6 Update pytest to 3.0.6
- Merge pull request #86 from asmodehn/pyup-update-cookiecutter-1.5.0-to-1.5.1 Update cookiecutter to 1.5.1
- Update cookiecutter from 1.5.0 to 1.5.1
- moved rosdep install out of catkin_pip, only used for tests. temporarily skipping broken test because of pip behavior…
- improved concurrency handling for pip by using setlock from daemontools. improved status message output.
- adding libssl-dev as dependency forpypackage
- adding catkin_pip function to call rosdep install. attempting fix for pypackage tests.
- adding libffi-dev as build_depend for python_boilerplate
- API changed ! redesigned workflow by doing "pip install -e package" during make stage instead of configure. All tests passing.
- Update pytest from 3.0.5 to 3.0.6
- reverted to pip 8.1.2 changes added in prevision of switching to pip 9.X when –ignore-installed working…
- added comment about sourcing install/setup.bash

- improved test cases to check the content of easy-install.pth
- adding cookiecutter pypackage-minimal for tests
- adding cookiecutter pypackage for tests
- adding cookiecutter pylibrary for testing
- adding result of investigation for unexpected dist-packages in sys.path tail… still WIP
- now handling environment setup only via catkin_env from env-hooks. made pure sh envhook to allow implementing other shells.
- WIP refactoring how we setup configure / build / devel/ install enironments
- improved path_prepend shell script
- Contributors: AlexV, alexv, pyup-bot
- removing install envhook. seems to break things. we dont want the install envhook to be created for catkin_pip since the target build env does not exist.
- Merge pull request #61 from asmodehn/refactor_envs Refactor envs
- adding docs to make decision clear regarding catkin / pyros_setup behavior.
- fixing up install devel command since we do not pass editable pkg path into pythonpath anymore.
- Merge branch 'gopher-devel' into refactor_envs
- Merge pull request #76 from asmodehn/easyinstall_fix Easyinstall fix
- commenting install tests… we dont have any.
- always adding the catkin_pip_env path to pythonpath to be able to find basic tools from the first time around.
- envhooks now not adding to pythonpath from pth files. too confusing. envhooks now not adding non existent paths to pythonpath. maybe good idea to match site module behavior. fixing tests.
- improved tests. added test for check with pyros_setup and interractive debugging
- fixed broken tests. now needs pyros-setup to get .pth paths before workspaces. cosmetics.
- improved tests and fix all. added actual test for pytest to pipproject.
- cleaning up command args when running pip install –editable to clean python path and workaround pip bug…
- replacing CI_ROS_DISTRO var by ROS_DISTRO, to have it set for calls to rosdep.
- more exhaustive test for sys_path
- commenting echoes from script since it cannot echo without breaking things anyway.
- reviewing shell script to make them a tiny more robust…
- removed useless bash envhook, fixed easy-install parse to include last line even if no empty line at end of file.
- moved rosdep install out of catkin_pip, only used for tests. temporarily skipping broken test because of pip behavior…
- improved concurrency handling for pip by using setlock from daemontools. improved status message output.
- adding libssl-dev as dependency forpypackage
- adding catkin_pip function to call rosdep install. attempting fix for pypackage tests.
- adding libffi-dev as build_depend for python_boilerplate
- API changed ! redesigned workflow by doing "pip install -e package" during make stage instead of configure. All tests passing.

- reverted to pip 8.1.2 changes added in prevision of switching to pip 9.X when –ignore-installed working…
- added comment about sourcing install/setup.bash
- improved test cases to check the content of easy-install.pth
- adding cookiecutter pypackage-minimal for tests
- adding cookiecutter pypackage for tests
- adding cookiecutter pylibrary for testing
- adding result of investigation for unexpected dist-packages in sys.path tail… still WIP
- now handling environment setup only via catkin_env from env-hooks. made pure sh envhook to allow implementing other shells.
- WIP refactoring how we setup configure / build / devel/ install enironments
- improved path_prepend shell script
- Contributors: AlexV, alexv

1.4.5 0.1.18 (2017-03-04)

- Pin pytest to latest version 3.0.5
- Pin pytest-timeout to latest version 1.2.0
- Pin nose to latest version 1.3.7
- Pin pytest-cov to latest version 2.4.0
- Pin cookiecutter to latest version 1.5.0
- adding pyup checks for dependencies
- Contributors: AlexV, alexv, pyup-bot

1.4.6 0.1.17 (2017-01-13)

- now always ignore-installed when installing requirements.
- pinned pip to 8.1.2 because of https://github.com/asmodehn/catkin_pip/issues/58
- Merge pull request #57 from asmodehn/devel upgrading gopher_devel
- Merge pull request #56 from asmodehn/gopher-devel drop some echoing
- drop some echoing
- Contributors: AlexV, Daniel Stonier, alexv

1.4.7 0.1.16 (2016-09-05)

- now also checking for –system for pip > 6.0.0.
- small improvements for travis checks
- Contributors: AlexV, alexv

1.4.8 0.1.15 (2016-09-01)

- now transferring paths from pth in devel site-packages to pythonpath shell env, to handle egg-link and workspace overlaying together…
- adding current devel space dist-packages via envhook to get it even if env not sourced… is it a good idea ?
- officially not supporting broken old pip on EOL saucy.
- Contributors: AlexV, alexv

1.4.9 0.1.14 (2016-08-30)

- Merge pull request #44 from asmodehn/pip_system Now checking for pip –system option before using.
- Now checking for pip –system option before using. cleanup some cmake status messages.
- improving pip detection
- Contributors: AlexV, alexv

1.4.10 0.1.13 (2016-08-28)

- fixing install rule for moved script.
- getting rid of rospack dependency. didnt always work. moved pythonpath_prepend shell script to use it via cmake variable.
- now checking system pip version to choose command line arguments for setup
- Contributors: AlexV

1.4.11 0.1.12 (2016-08-27)

- Merge pull request #40 from asmodehn/env_hooks Env hooks
- Merge pull request #39 from asmodehn/include_seq preventing multiple includes, reviewing variable scope.
- preventing multiple includes, reviewing variable scope.
- Merge branch 'devel' of https://github.com/asmodehn/catkin_pip into env_hooks # Conflicts: # CMakeLists.txt # cmake/catkin-pip.cmake.in # cmake/env-hooks/42.site_packages.bash.develspace.in
- Updated README
- Merge pull request #33 from asmodehn/install_no_deps first implementation of –no-deps to no install a package dependencie…
- Merge pull request #35 from asmodehn/kinetic-devel fixing pip upgrade for kinetic, based on ROS_DISTRO env var.
- requirements now correctly loading catkin-pip build/catkin_pip_env. now avoiding to load catkin-pip-requirements by itself.
- fixing check of envvar ROS_DISTRO from cmake configure to decide which pip command to run
- fixing rospack call. passing travis matrix env vars via shell command since docker run vars break on exec call.
- now passing travis matrix env vars to container.

- adding apt-get update call. also install sudo as not installed by default on xenial and required by rosdep. cosmetics
- using docker cp instead of volume to workaround docker/travis bug.
- removing volume to $HOME in case it is the cause of docker breaks.
- travis_checks script now change to its directory as first step. fixed some docker commands.
- fixing ros image name, container_name. added rosdep comand to get dependencies.
- changing travis to use docker to test multiple distro.
- fixing pip upgrade for kinetic, based on ROS_DISTRO env var.
- Restructured documentation
- started new doc structure
- documentation improvements
- adding doc as reference for basic catkin build release flow
- first implementation of –no-deps to no install a package dependencies via pip. helps confirm rosdep dependencies
- now using simplified sh env_hook
- Contributors: AlexV, alexv

1.4.12 0.1.11 (2016-08-11)

- added description of the catkin_pip build flow
- we might not need the install envhook after all. correct setuptools is found via path in install script. correct tools for test or other should be found via path in generated scripts, and used via catkin/make commands.
- added warning in pycharm setup doc.
- added first draft of pycharm setup doc
- improved workflow doc with pointer to example package repos.
- adding documentation for 3 ros-python workflows enabled by catkin_pip
- improving documentation
- disabling tests check from travis on install since mypippkg doesnt have any yet.
- fixing travis_checks to run our pytest version from catkin_pip_env
- cleaning up doc, installing ros-base in travis install step.
- adding specific script for travis checks. added basic doc structure.
- new travis build flow to split devel and install flow and avoid one unwanted interferences.
- Contributors: alexv

1.4.13 0.1.10 (2016-08-09)

- added rospack dependency
- (Re)adding site-packages folder creation in devel workspace.

- setup of catkin_pip environment also adds the workspace site-packages to the python path to get it ready for use, even if envhook was not used before.
- Merge pull request #28 from asmodehn/separate_catkin_pip_env separating catkin_pip environment with workspace environment.
- making sure env-hooks have all variables setup before adding.
- separating catkin_pip environment with workspace environment. added envhook for loading caktin_pip env on installspace. removing install script for python on windows for now (outdated).
- Contributors: AlexV, alexv

1.4.14 0.1.9 (2016-06-24)

- fixed site_packages env-hook. bash script seems to work fine after all, the problem was somewhere else. simplified the envhook flow between catkin, package, overlay.
- changed site-packages env-hook to have .sh extension. moving prepend function into catkin-pip package itself.
- Contributors: alexv

1.4.15 0.1.8 (2016-06-06)

- fix nose and pytest test runners to launch from pip latest install by catkin-pip.
- fix PYTHONPATH manipulation to prepend a path. not adding /opt/ros/<distro> to the path since original catkin will take care of that.
- Contributors: AlexV

1.4.16 0.1.7 (2016-06-05)

- fixed site-packages env-hook to install with catkin-pip and not built project, and to be activated only in devel space.
- Contributors: AlexV

1.4.17 0.1.6 (2016-06-05)

- improving python_install templates to match original version more…
- improving python install script to pass only one –root option
- Contributors: AlexV, alexv

1.4.18 0.1.5 (2016-06-03)

- removing subdir in cfg_extra because of https://github.com/ros/catkin/issues/805
- Contributors: alexv

1.4.19 0.1.4 (2016-06-02)

- adding pytest as a test runner. now using our nose in nosetests (instead of sytem one) small fixes.
- now travis building on jade as well
- Contributors: AlexV, alexv

1.4.20 0.1.3 (2016-06-01)

- renaming catkin_pure_python to catkin_pip for clarity
- Contributors: alexv

1.4.21 0.1.2 (2016-05-30)

- fixing python_setuptools_install templates location and permissions
- Contributors: alexv

1.4.22 0.1.1 (2016-05-30)

- fixing catkin_pip_runcmd for package, hopefully.
- Contributors: AlexV

1.4.23 0.1.0 (2016-05-29)

- separating catkin_pip_setup and catkin_package macros.
- now ignoring installed pip packaging when fetching requirements for pipproject.
- removing debug output for shell envhook
- fixing install procedure to get same structure as the distutils version.
- now catkin-pip package is using normal catkin_package(), and installs fine, although with setuptools, which might break packaging…
- refactoring cmake include and configure. test project devel space ok. the rest is still broken…
- small improvement to do less configuration
- now using an envhook to modify pythonpath instead of hacking catkin's _setup_util.py
- _setup_util.py hack now done in cmake binary dir instead of final workspace.
- Contributors: AlexV, alexv

1.4.24 0.0.8 (2016-05-10)

- not writing cmake files into workspace anymore. instead in build directory of each package.
- added doc about pip/ros dependency handling.
- Contributors: alexv

1.4.25 0.0.7 (2016-05-09)

- removing –ignore-installed for editable package, to allow requirements to satisfy setup.py dependencies.
- changing package to format v2
- Contributors: alexv

1.4.26 0.0.6 (2016-04-29)

- adding –ignore-installed to avoid pip picking up local editable package when installing.
- informative comments
- better fix for catkin-pip requirements not found in workspace path
- fixing travis to run tests for catkin-pip
- Contributors: alexv

1.4.27 0.0.5 (2016-04-26)

- fix catkin-pip requirements not found in workspace path
- typo
- Contributors: alexv

1.4.28 0.0.4 (2016-04-08)

- Merge remote-tracking branch 'origin/indigo' into indigo
- now prepending site-packages path. also for install space.
- Contributors: alexv

1.4.29 0.0.3 (2016-04-07)

- small refactor to improve cmake messages
- now specifying source director and exists-action backup when installing requirements. restored previous behavior to check for installed packages before installing current package. this avoid reinstalling dependencies satisfied by requirements.
- always cleaning cache for catkin_pip for safety.
- added –ignore-installed so pip doesnt try to remove old packages from system. quick Readme Roadmap
- Contributors: alexv

1.4.30 0.0.2 (2016-04-04)

- cleaning up cmake ouput. fixing install sys pip path and pippkg path.
- Merge pull request #2 from asmodehn/install_rules Install rules
- improve pip finding. fixed install.

- restructuring to get install running same code as devel
- adding git ignore and cmake file for building mypippkg test
- removed ROS dependency on cookiecutter since we need to get it from pip.
- added travis build status
- fixing default argument for catkin_pip_package fixing catkin_pip test build.
- attempting to fix nose and tests…
- improved environment detection and setup.
- improved readme
- fixed changelog
- Contributors: AlexV

1.4.31 0.0.1 (2016-03-31)

- fixing install rules. improving pip download by using cache for catkin-pip requirements.
- now devel workspace populated with latest pip.
- first version of package. still trying stuff out…
- Contributors: AlexV

CHAPTER 2

Packages : Working with Catkin-pip

Where we describe the different kind of package we can work with, and how catkin-pip makes the transition easier.

2.1 Catkin Package

Ref : http://wiki.ros.org/ROS/Tutorials/catkin/CreatingPackage

We are focusing here only in python packages. We will use the “catkin way of doing things” described here, as a reference when discussing improvements brought with catkin_pip.

2.1.1 Package Structure

This is a generic example inspired by the pure catkin package from the Pyros dependencies : pyros-test

Files hierarchy

The source file hierarchy is:

```
.
├── CHANGELOG.rst              # A changelog, usually generated from git commit with␣
↪catkin_generate_changelog
├── CMakeLists.txt             # A CMakeLists, using Catkin macros
├── doc
│   ├── changelog_link.rst     # Documentation link to existing changelog
│   ├── conf.py                # Sphinx doc configuration
│   ├── mydocs.rst             # Documentation for Sphinx
│   └── readme_link.rst        # Documentation link to existing readme
├── nodes
│   └── node1.py               # python script launching a ROS node
├── scripts
│   └── script1.py             # python utility script
```

(continues on next page)

(continued from previous page)

```
├── package.xml               # package.xml required by catkin
├── README.rst
├── setup.py                  # setup.py using distutils and catkin_pkg extension
├── src
│   └── mypkg                 # mypkg python package
│       ├── module1.py
│       └── __init__.py
├── msg
│   └── mymessage.msg         # ROS message definition
└── srv
    └── myservice.srv         # ROS service definition
```

- package.xml looks like

```
<?xml version="1.0"?>
<package format="2">
  <name>mypkg</name>
  <version>0.0.4</version>
  <description>
    MyPkg description
  </description>

  <license>BSD</license>

  <url type="repository">https://github.com/author/mypkg</url>
  <url type="bugtracker">https://github.com/author/mypkg/issues</url>

  <author email="author@gmail.com">Author</author>
  <maintainer email="maintainer@gmail.com">Maintainer</maintainer>

  <buildtool_depend version_gte="0.6.18">catkin</buildtool_depend>

  <depend version_gte="1.11.19">rospy</depend>
  <depend version_gte="0.5.10">std_msgs</depend>

  <build_depend version_gte="0.2.10">message_generation</build_depend>
  <build_depend version_gte="0.10.0">roslint</build_depend>

  <exec_depend version_gte="0.4.12">message_runtime</exec_depend>

  <!-- these dependencies are only for testing -->
  <test_depend version_gte="1.11.19">rostest</test_depend>
  <test_depend version_gte="1.11.12">rosunit</test_depend>

  <!-- documentation dependencies -->
  <doc_depend version_gte="0.2.10">python-catkin-pkg</doc_depend>

</package>
```

- CMakeLists.txt looks like

```
cmake_minimum_required(VERSION 2.8.3)
project(mypkg)

# Minimal Python module setup
find_package(catkin REQUIRED COMPONENTS
    roslint
```

(continues on next page)

(continued from previous page)

```
    rospy
    std_msgs
    message_generation
)

catkin_python_setup()

add_message_files(DIRECTORY msg
    FILES
    mymessage.msg
)
add_service_files(DIRECTORY srv
    FILES
    myservice.srv
)
generate_messages(DEPENDENCIES std_msgs)

catkin_package( CATKIN_DEPENDS message_runtime std_msgs)

install(
    PROGRAMS
        nodes/node1.py
    DESTINATION ${CATKIN_PACKAGE_BIN_DESTINATION}
    )

install(
    PROGRAMS
        scripts/script1.py
    DESTINATION ${CATKIN_PACKAGE_BIN_DESTINATION}
    )

# Lint Python modules
file(GLOB_RECURSE ${PROJECT_NAME}_PY_SRCS
     RELATIVE ${PROJECT_SOURCE_DIR} src/pyros_test/*.py)
roslint_python(${${PROJECT_NAME}_PY_SRCS})
```

- setup.py looks like:

```
from distutils.core import setup
from catkin_pkg.python_setup import generate_distutils_setup

# ROS PACKAGING
# using distutils : https://docs.python.org/2/distutils
# fetch values from package.xml
setup_args = generate_distutils_setup(
    packages=[
        'mypkg',
    ],
    package_dir={
        'mypkg': 'src/mypkg',
    }
)
setup(**setup_args)
```

Dependencies

Dependencies are expressed via rosdep keys in the package.xml file.

Note these rosdep keys can refer to pip packages (check the rosdistro repository, you will find pip keys) but AFAIK:

- there are no guarantees that these pip packages will play nice along the rest of your ROS distro.
- there are little information on how rosdep handle pip dependencies (version requirements ?)
- there is no clear visibility on rosdep pip support in the long term.

If you think I am mistaken please open an issue on catkin_pip repository, and share the information you have regarding these topics.

2.1.2 Package Development Workflow

This is a description of the generic ROS catkin workflow to retrieve, develop, build, test and release a catkin-based package. We will use pyros-test project as an example. **TODO : travis check these with doctest + running these in isolation in container**

- Retrieve the project:

```
$ mkdir -p catkin_ws/src
$ cd catkin_ws/src/
$ wstool init
Writing /home/alexv/doctest/catkin_ws/src/.rosinstall

update complete.

$ wstool set pyros-test https://github.com/asmodehn/pyros-test.git --git

     Add new elements:
  pyros-test        git  https://github.com/asmodehn/pyros-test.git

Continue: (y)es, (n)o: y
Overwriting /home/alexv/doctest/catkin_ws/src/.rosinstall
Config changed, remember to run 'wstool update pyros-test' to update the folder␣
↪from git

$ wstool update pyros-test
[pyros-test] Fetching https://github.com/asmodehn/pyros-test.git (version None)␣
↪to /home/alexv/doctest/catkin_ws/src/pyros-test
Cloning into '/home/alexv/doctest/catkin_ws/src/pyros-test'...
remote: Counting objects: 87, done.
remote: Total 87 (delta 0), reused 0 (delta 0), pack-reused 87
Unpacking objects: 100% (87/87), done.
Checking connectivity... done.
[pyros-test] Done.
```

- Source your ROS environment:

```
$ source /opt/ros/indigo/setup.bash
```

- Build with catkin_make:

```
$ catkin_make
Base path: /home/alexv/doctest/catkin_ws
```

(continues on next page)

(continued from previous page)

```
Source space: /home/alexv/doctest/catkin_ws/src
Build space: /home/alexv/doctest/catkin_ws/build
Devel space: /home/alexv/doctest/catkin_ws/devel
Install space: /home/alexv/doctest/catkin_ws/install
Creating symlink "/home/alexv/doctest/catkin_ws/src/CMakeLists.txt" pointing to "/
↪opt/ros/indigo/share/catkin/cmake/toplevel.cmake"
####
#### Running command: "cmake /home/alexv/doctest/catkin_ws/src -DCATKIN_DEVEL_
↪PREFIX=/home/alexv/doctest/catkin_ws/devel -DCMAKE_INSTALL_PREFIX=/home/alexv/
↪doctest/catkin_ws/install -G Unix Makefiles" in "/home/alexv/doctest/catkin_ws/
↪build"
####
-- The C compiler identification is GNU 4.8.4
-- The CXX compiler identification is GNU 4.8.4
-- Check for working C compiler: /usr/bin/cc
-- Check for working C compiler: /usr/bin/cc -- 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: /home/alexv/doctest/catkin_ws/devel
-- Using CMAKE_PREFIX_PATH: /opt/ros/indigo
-- This workspace overlays: /opt/ros/indigo
-- Found PythonInterp: /usr/bin/python (found version "2.7.6")
-- Using PYTHON_EXECUTABLE: /usr/bin/python
-- Using Debian Python package layout
-- Using empy: /usr/bin/empy
-- Using CATKIN_ENABLE_TESTING: ON
-- Call enable_testing()
-- Using CATKIN_TEST_RESULTS_DIR: /home/alexv/doctest/catkin_ws/build/test_results
-- Looking for include file pthread.h
-- Looking for include file pthread.h - found
-- Looking for pthread_create
-- Looking for pthread_create - not found
-- Looking for pthread_create in pthreads
-- Looking for pthread_create in pthreads - not found
-- Looking for pthread_create in pthread
-- Looking for pthread_create in pthread - found
-- Found Threads: TRUE
-- Found gtest sources under '/usr/src/gtest': gtests will be built
-- Using Python nosetests: /usr/bin/nosetests-2.7
-- catkin 0.6.18
-- BUILD_SHARED_LIBS is on
-- ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
-- ~~  traversing 1 packages in topological order:
-- ~~  - pyros_test
-- ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
-- +++ processing catkin package: 'pyros_test'
-- ==> add_subdirectory(pyros-test)
-- Using these message generators: gencpp;genlisp;genpy
-- pyros_test: 0 messages, 1 services
-- Configuring done
-- Generating done
-- Build files have been written to: /home/alexv/doctest/catkin_ws/build
####
```

(continues on next page)

(continued from previous page)

```
#### Running command: "make -j8 -l8" in "/home/alexv/doctest/catkin_ws/build"
####
Scanning dependencies of target std_msgs_generate_messages_cpp
Scanning dependencies of target std_msgs_generate_messages_py
Scanning dependencies of target _pyros_test_generate_messages_check_deps_
↪StringEchoService
Scanning dependencies of target std_msgs_generate_messages_lisp
[  0%] [  0%] Built target std_msgs_generate_messages_cpp
[  0%] Built target std_msgs_generate_messages_py
Built target std_msgs_generate_messages_lisp
[  0%] Built target _pyros_test_generate_messages_check_deps_StringEchoService
Scanning dependencies of target pyros_test_generate_messages_cpp
Scanning dependencies of target pyros_test_generate_messages_lisp
Scanning dependencies of target pyros_test_generate_messages_py
[ 75%] [ 75%] [ 75%] Generating Lisp code from pyros_test/StringEchoService.srv
Generating C++ code from pyros_test/StringEchoService.srv
Generating Python code from SRV pyros_test/StringEchoService
[100%] Generating Python srv __init__.py for pyros_test
[100%] Built target pyros_test_generate_messages_lisp
[100%] Built target pyros_test_generate_messages_py
[100%] Built target pyros_test_generate_messages_cpp
Scanning dependencies of target pyros_test_generate_messages
[100%] Built target pyros_test_generate_messages
```

- Source devel space:

```
$ source devel/setup.bash
```

- Run tests with catkin_make test:

```
$ catkin_make test
Base path: /home/alexv/doctest/catkin_ws
Source space: /home/alexv/doctest/catkin_ws/src
Build space: /home/alexv/doctest/catkin_ws/build
Devel space: /home/alexv/doctest/catkin_ws/devel
Install space: /home/alexv/doctest/catkin_ws/install
####
#### Running command: "make cmake_check_build_system" in "/home/alexv/doctest/
↪catkin_ws/build"
####
####
#### Running command: "make test -j8 -l8" in "/home/alexv/doctest/catkin_ws/build"
####
Running tests...
Test project /home/alexv/doctest/catkin_ws/build
No tests were found!!!
```

- Debug tests with catkin_make run_tests:

```
$ catkin_make run_tests
Base path: /home/alexv/doctest/catkin_ws
Source space: /home/alexv/doctest/catkin_ws/src
Build space: /home/alexv/doctest/catkin_ws/build
Devel space: /home/alexv/doctest/catkin_ws/devel
Install space: /home/alexv/doctest/catkin_ws/install
####
#### Running command: "make cmake_check_build_system" in "/home/alexv/doctest/
↪catkin_ws/build"
```

(continues on next page)

(continued from previous page)

```
####
####
#### Running command: "make run_tests -j8 -l8" in "/home/alexv/doctest/catkin_ws/
↪build"
####
Scanning dependencies of target run_tests
Built target run_tests
```

From a different shell to not have your environment polluted with devel space:

- Source your ROS environment:

```
$ source /opt/ros/indigo/setup.bash
```

- Install with catkin_make install:

```
$ catkin_make install
Base path: /home/alexv/doctest/catkin_ws
Source space: /home/alexv/doctest/catkin_ws/src
Build space: /home/alexv/doctest/catkin_ws/build
Devel space: /home/alexv/doctest/catkin_ws/devel
Install space: /home/alexv/doctest/catkin_ws/install
####
#### Running command: "make cmake_check_build_system" in "/home/alexv/doctest/
↪catkin_ws/build"
####
####
#### Running command: "make install -j8 -l8" in "/home/alexv/doctest/catkin_ws/
↪build"
####
[  0%] [  0%] [  0%] Built target std_msgs_generate_messages_lisp
Built target std_msgs_generate_messages_py
Built target std_msgs_generate_messages_cpp
[  0%] Built target _pyros_test_generate_messages_check_deps_StringEchoService
[100%] [100%] [100%] Built target pyros_test_generate_messages_lisp
Built target pyros_test_generate_messages_cpp
Built target pyros_test_generate_messages_py
[100%] Built target pyros_test_generate_messages
Install the project...
-- Install configuration: ""
-- Installing: /home/alexv/doctest/catkin_ws/install/_setup_util.py
-- Installing: /home/alexv/doctest/catkin_ws/install/env.sh
-- Installing: /home/alexv/doctest/catkin_ws/install/setup.bash
-- Installing: /home/alexv/doctest/catkin_ws/install/setup.sh
-- Installing: /home/alexv/doctest/catkin_ws/install/setup.zsh
-- Installing: /home/alexv/doctest/catkin_ws/install/.rosinstall
+ cd /home/alexv/doctest/catkin_ws/src/pyros-test
+ mkdir -p /home/alexv/doctest/catkin_ws/install/lib/python2.7/dist-packages
+ /usr/bin/env PYTHONPATH=/home/alexv/doctest/catkin_ws/install/lib/python2.7/
↪dist-packages:/home/alexv/doctest/catkin_ws/build/lib/python2.7/dist-packages:/
↪opt/ros/indigo/lib/python2.7/dist-packages CATKIN_BINARY_DIR=/home/alexv/
↪doctest/catkin_ws/build /usr/bin/python /home/alexv/doctest/catkin_ws/src/pyros-
↪test/setup.py build --build-base /home/alexv/doctest/catkin_ws/build/pyros-test␣
↪install --install-layout=deb --prefix=/home/alexv/doctest/catkin_ws/install --
↪install-scripts=/home/alexv/doctest/catkin_ws/install/bin
running build
running build_py
```

(continues on next page)

(continued from previous page)

```
creating /home/alexv/doctest/catkin_ws/build/pyros-test/lib.linux-x86_64-2.7
creating /home/alexv/doctest/catkin_ws/build/pyros-test/lib.linux-x86_64-2.7/
↪pyros_test
copying src/pyros_test/echo_node.py -> /home/alexv/doctest/catkin_ws/build/pyros-
↪test/lib.linux-x86_64-2.7/pyros_test
copying src/pyros_test/__init__.py -> /home/alexv/doctest/catkin_ws/build/pyros-
↪test/lib.linux-x86_64-2.7/pyros_test
running install
running install_lib
creating /home/alexv/doctest/catkin_ws/install/lib/python2.7/dist-packages/pyros_
↪test
copying /home/alexv/doctest/catkin_ws/build/pyros-test/lib.linux-x86_64-2.7/pyros_
↪test/echo_node.py -> /home/alexv/doctest/catkin_ws/install/lib/python2.7/dist-
↪packages/pyros_test
copying /home/alexv/doctest/catkin_ws/build/pyros-test/lib.linux-x86_64-2.7/pyros_
↪test/__init__.py -> /home/alexv/doctest/catkin_ws/install/lib/python2.7/dist-
↪packages/pyros_test
byte-compiling /home/alexv/doctest/catkin_ws/install/lib/python2.7/dist-packages/
↪pyros_test/echo_node.py to echo_node.pyc
byte-compiling /home/alexv/doctest/catkin_ws/install/lib/python2.7/dist-packages/
↪pyros_test/__init__.py to __init__.pyc
running install_egg_info
Writing /home/alexv/doctest/catkin_ws/install/lib/python2.7/dist-packages/pyros_
↪test-0.0.4.egg-info
-- Installing: /home/alexv/doctest/catkin_ws/install/share/pyros_test/srv/
↪StringEchoService.srv
-- Installing: /home/alexv/doctest/catkin_ws/install/share/pyros_test/cmake/pyros_
↪test-msg-paths.cmake
-- Installing: /home/alexv/doctest/catkin_ws/install/include/pyros_test
-- Installing: /home/alexv/doctest/catkin_ws/install/include/pyros_test/
↪StringEchoServiceResponse.h
-- Installing: /home/alexv/doctest/catkin_ws/install/include/pyros_test/
↪StringEchoServiceRequest.h
-- Installing: /home/alexv/doctest/catkin_ws/install/include/pyros_test/
↪StringEchoService.h
-- Installing: /home/alexv/doctest/catkin_ws/install/share/common-lisp/ros/pyros_
↪test
-- Installing: /home/alexv/doctest/catkin_ws/install/share/common-lisp/ros/pyros_
↪test/srv
-- Installing: /home/alexv/doctest/catkin_ws/install/share/common-lisp/ros/pyros_
↪test/srv/_package.lisp
-- Installing: /home/alexv/doctest/catkin_ws/install/share/common-lisp/ros/pyros_
↪test/srv/pyros_test-srv.asd
-- Installing: /home/alexv/doctest/catkin_ws/install/share/common-lisp/ros/pyros_
↪test/srv/StringEchoService.lisp
-- Installing: /home/alexv/doctest/catkin_ws/install/share/common-lisp/ros/pyros_
↪test/srv/_package_StringEchoService.lisp
Listing /home/alexv/doctest/catkin_ws/devel/lib/python2.7/dist-packages/pyros_
↪test ...
Compiling /home/alexv/doctest/catkin_ws/devel/lib/python2.7/dist-packages/pyros_
↪test/__init__.py ...
Listing /home/alexv/doctest/catkin_ws/devel/lib/python2.7/dist-packages/pyros_
↪test/srv ...
Compiling /home/alexv/doctest/catkin_ws/devel/lib/python2.7/dist-packages/pyros_
↪test/srv/_StringEchoService.py ...
Compiling /home/alexv/doctest/catkin_ws/devel/lib/python2.7/dist-packages/pyros_
↪test/srv/__init__.py ...
```

(continues on next page)

(continued from previous page)

```
-- Installing: /home/alexv/doctest/catkin_ws/install/lib/python2.7/dist-packages/
↪pyros_test
-- Installing: /home/alexv/doctest/catkin_ws/install/lib/python2.7/dist-packages/
↪pyros_test/srv
-- Installing: /home/alexv/doctest/catkin_ws/install/lib/python2.7/dist-packages/
↪pyros_test/srv/_StringEchoService.pyc
-- Installing: /home/alexv/doctest/catkin_ws/install/lib/python2.7/dist-packages/
↪pyros_test/srv/_StringEchoService.py
-- Installing: /home/alexv/doctest/catkin_ws/install/lib/python2.7/dist-packages/
↪pyros_test
-- Installing: /home/alexv/doctest/catkin_ws/install/lib/python2.7/dist-packages/
↪pyros_test/srv
-- Installing: /home/alexv/doctest/catkin_ws/install/lib/python2.7/dist-packages/
↪pyros_test/srv/__init__.pyc
-- Installing: /home/alexv/doctest/catkin_ws/install/lib/python2.7/dist-packages/
↪pyros_test/srv/__init__.py
-- Installing: /home/alexv/doctest/catkin_ws/install/lib/pkgconfig/pyros_test.pc
-- Installing: /home/alexv/doctest/catkin_ws/install/share/pyros_test/cmake/pyros_
↪test-msg-extras.cmake
-- Installing: /home/alexv/doctest/catkin_ws/install/share/pyros_test/cmake/pyros_
↪testConfig.cmake
-- Installing: /home/alexv/doctest/catkin_ws/install/share/pyros_test/cmake/pyros_
↪testConfig-version.cmake
-- Installing: /home/alexv/doctest/catkin_ws/install/share/pyros_test/package.xml
-- Installing: /home/alexv/doctest/catkin_ws/install/lib/pyros_test/echo.py
-- Installing: /home/alexv/doctest/catkin_ws/install/lib/pyros_test/emptyService.
↪py
-- Installing: /home/alexv/doctest/catkin_ws/install/lib/pyros_test/slowService.py
-- Installing: /home/alexv/doctest/catkin_ws/install/lib/pyros_test/
↪triggerService.py
-- Installing: /home/alexv/doctest/catkin_ws/install/lib/pyros_test/common.py
-- Installing: /home/alexv/doctest/catkin_ws/install/lib/pyros_test/string_pub_
↪node.py
-- Installing: /home/alexv/doctest/catkin_ws/install/lib/pyros_test/string_pubnot_
↪node.py
-- Installing: /home/alexv/doctest/catkin_ws/install/lib/pyros_test/string_slow_
↪node.py
-- Installing: /home/alexv/doctest/catkin_ws/install/lib/pyros_test/string_sub_
↪node.py
```

- Source the install space:

```
$ source install/setup.bash
```

- Run tests as a final user would:

```
**TODO**
```

Release

- Change to the project directory you want to release:

```
$ cd src/pyros-test/
```

- Generate the changelog:

```
$ catkin_generate_changelog
Found packages: pyros_test
Querying commit information since latest tag...
Updating forthcoming section of changelog files...
- updating './CHANGELOG.rst'
Done.
Please review the extracted commit messages and consolidate the changelog entries␣
↪before committing the files!
```

- Prepare the release:

```
$ catkin_prepare_release
```

- bloom-release –rosdistro indigo –track indigo pyros_test

If this is not accurate, if I missed a step, or if there is a better way to do the same, please open a PullRequest or an issue on the catkin_pip repository with the related information so this doc can be corrected

TODO : review this with the new catkin tools coming up (ie catkin build)

2.1.3 Continuous Testing Workflow

Because no software works until it has been tested, you should configure travis on your repository to run test with each commit and pull request.

Catkin testing can be done with a simple *.travis.yml* file and a small shell script.

2.2 Pip Package

Ref : https://packaging.python.org/

There have been historically many ways to package and distribute python code. While many of these are still usable, we are focusing here on the python packaging authority way to package python code.

2.2.1 Package Structure

This is a generic example inspired by the pure pip package from Pyros dependencies: pyros-setup

Files hierarchy

The source file hierarchy is:

```
.
├── CHANGELOG.rst              # A changelog, usually generated from git commit with␣
↪gitchangelog
├── doc
│   ├── changelog_link.rst     # Documentation link to existing changelog
│   ├── conf.py                # Sphinx doc configuration
│   ├── mydocs.rst             # Documentation for Sphinx
│   └── readme_link.rst        # Documentation link to existing readme
├── requirements.txt
├── MANIFEST.in
```

(continues on next page)

(continued from previous page)

```
├── pyros_setup
│   ├── __init__.py
│   └── module1.py
├── README.rst
├── setup.cfg
├── setup.py
└── tox.ini
```

Dependencies

Dependencies are manage with pip, usually installed in a virtual environment.

There are two kinds of dependencies : - The dependencies needed for the installed pacakge to work (in setuptools.setup install_requires parameter) - The dependencies needed for the developer to work with the package (in requirements.txt)

For more detail about these, you should refer to : https://packaging.python.org/requirements/

2.2.2 Package Development Workflow

This is a description of the generic python workflow to develop, build, test and release a python based package. We will use pyros-setup project as an example. **TODO : travis check these with doctest + running these in isolation in container**

- Get the Source ie. clone a Repo:

```
$ git clone https://github.com/asmodehn/pyros-setup.git
Cloning into 'pyros-setup'...
remote: Counting objects: 718, done.
remote: Compressing objects: 100% (64/64), done.
remote: Total 718 (delta 29), reused 0 (delta 0), pack-reused 654
Receiving objects: 100% (718/718), 134.89 KiB | 0 bytes/s, done.
Resolving deltas: 100% (433/433), done.
Checking connectivity... done.
```

- Create a virtual environment and activate it (here using virtualenvwrapper). We also want a virtualenvironment that can use the ROS & system packages:

```
$ mkvirtualenv pyros-setup --system-site-packages
New python executable in /home/alexv/.virtualenvs/pyros-setup/bin/python
Installing setuptools, pip, wheel...done.
```

- Install with pip:

```
$ pip install .
Processing /home/alexv/doctest/pyros-setup
Collecting six (from pyros-setup==0.1.2)
/home/alexv/.virtualenvs/pyros-setup/local/lib/python2.7/site-packages/pip/_
↪vendor/requests/packages/urllib3/util/ssl_.py:318: SNIMissingWarning: An HTTPS␣
↪request has been made, but the SNI (Subject Name Indication) extension to TLS␣
↪is not available on this platform. This may cause the server to present an␣
↪incorrect TLS certificate, which can cause validation failures. You can upgrade␣
↪to a newer version of Python to solve this. For more information, see https://
↪urllib3.readthedocs.org/en/latest/security.html#snimissingwarning.
  SNIMissingWarning
```

(continues on next page)

(continued from previous page)

```
/home/alexv/.virtualenvs/pyros-setup/local/lib/python2.7/site-packages/pip/_
→vendor/requests/packages/urllib3/util/ssl_.py:122: InsecurePlatformWarning: A␣
→true SSLContext object is not available. This prevents urllib3 from configuring␣
→SSL appropriately and may cause certain SSL connections to fail. You can␣
→upgrade to a newer version of Python to solve this. For more information, see␣
→https://urllib3.readthedocs.org/en/latest/security.html#insecureplatformwarning.
  InsecurePlatformWarning
  Using cached six-1.10.0-py2.py3-none-any.whl
Collecting pyros_config>=0.1.2 (from pyros-setup==0.1.2)
  Using cached pyros_config-0.1.3-py2-none-any.whl
Collecting pytest>=2.5.1 (from pyros-setup==0.1.2)
  Using cached pytest-2.9.2-py2.py3-none-any.whl
Collecting py>=1.4.29 (from pytest>=2.5.1->pyros-setup==0.1.2)
  Using cached py-1.4.31-py2.py3-none-any.whl
Building wheels for collected packages: pyros-setup
  Running setup.py bdist_wheel for pyros-setup ... done
  Stored in directory: /home/alexv/.cache/pip/wheels/39/50/33/
→ab49df5cef0ef2ce4e23dabd0c9ea5d81f9af131c80d4b2523
Successfully built pyros-setup
Installing collected packages: six, py, pytest, pyros-config, pyros-setup
Successfully installed py-1.4.31 pyros-config-0.1.3 pyros-setup-0.1.2 pytest-2.9.
→2 six-1.10.0
```

- Run tests in current environment with nose or pytest. Note that to avoid import conflict, and for tests to pass, you might need to move to a directory where the module is not accessible from current working directory (https://github.com/asmodehn/pyros-setup/issues/27)

```
$ pyros_setup --pytest
============================= test session starts ==============================
platform linux2 -- Python 2.7.6, pytest-2.9.2, py-1.4.31, pluggy-0.3.1
rootdir: /home/alexv/doctest/pyros-setup, inifile:
collected 3 items

.. ...

=========================== 3 passed in 0.06 seconds ===========================
```

- Run tests on multiple python environments with tox:

```
$ tox
GLOB sdist-make: /home/alexv/doctest/pyros-setup/setup.py
py27 create: /home/alexv/doctest/pyros-setup/.tox/py27
py27 inst: /home/alexv/doctest/pyros-setup/.tox/dist/pyros_setup-0.1.2.zip
py27 installed: alembic==0.6.2,amqp==1.3.3,ansible==2.1.0.0,anyjson==0.3.3,apt-
→xapian-index==0.45,argh==0.26.1,args==0.1.0,autopep8==0.9.1,Babel==1.3,
→backports.ssl-match-hostname==3.5.0.1,beautifulsoup4==4.2.1,billiard==3.3.0.15,
→binaryornot==0.2.0,blinker==1.3,bloom==0.5.21,bzr==2.1.4,catkin-pkg==0.2.10,
→catkin-sphinx==0.2.2,catkin-tools==0.4.2,celery==3.1.6,chardet==2.0.1,
→Cheetah==2.4.4,cl==0.0.3,clint==0.5.1,cobbler==2.4.1,colorama==0.2.5,command-
→not-found==0.3,configobj==4.7.2,cookiecutter==0.6.4,coverage==3.7.1,
→debtagshw==0.1,defer==1.0.6,dirspec==13.10,distro-info==0.12,Django==1.6.1,
→docker-py==1.8.1,dockerpty==0.3.4,docopt==0.6.2,docutils==0.11,dulwich==0.9.4,
→empy==3.1,enum34==0.9.23,epydoc==3.0.1,fastimport==0.9.2,fig==1.0.1,Flask==0.10.
→1,futures==2.1.6,gbp==0.6.9,git-remote-helpers==0.1.0,gitchangelog==2.3.0,
→gunicorn==17.5,html5lib==0.999,httplib2==0.8,importlib==1.0.3,iotop==0.6,
→ipaddress==1.0.16,itsdangerous==0.22,Jinja2==2.7.2,jsonpickle==0.9.2,keyring==3.
→5,kitchen==1.1.1,kombu==3.0.7,launchpadlib==1.10.2,lazr.restfulclient==0.13.3,
→lazr.uri==1.0.3,libvirt-python==1.2.2,lxml==3.3.3,mailer==0.7,Mako==0.9.1,
→MarkupSafe==0.18,matplotlib==1.3.1,meld3==0.6.10,mercurial==1.4.2,mock==1.0.1,
→mod-python==3.3.1,MySQL-python==1.2.3,netaddr==0.7.10,netifaces==0.8,nose==1.3.
→1,numpy==1.8.2,oauth==1.0.1,oauthlib==0.6.1,oneconf==0.3.7.14.4.1,osrf-
→pycommon==0.1.2,PAM==0.4.2,paramiko==1.10.1,passlib==1.5.3,pathtools==0.1.2,
→pep8==1.4.6,pexpect==3.1,Pillow==2.3.0,piston-mini-client==0.7.5,pkginfo==1.3.2,
→pluggy==0.3.1,progressbar==2.3,protobuf==2.5.0,psutil==1.2.1,py==1.4.31,
```

(continues on next page)

(continued from previous page)

```
py27 runtests: PYTHONHASHSEED='473323988'
py27 runtests: commands[0] | py.test --pyargs pyros_setup
WARNING:test command found but not installed in testenv
  cmd: /home/alexv/.virtualenvs/pyros-setup/bin/py.test
  env: /home/alexv/doctest/pyros-setup/.tox/py27
Maybe you forgot to specify a dependency? See also the whitelist_externals␣
↪envconfig setting.
=============================================================== test session␣
↪starts ===============================================================
platform linux2 -- Python 2.7.6, pytest-2.9.2, py-1.4.31, pluggy-0.3.1
rootdir: /home/alexv/doctest/pyros-setup, inifile:
collected 3 items

pyros_setup/tests/test_setup.py ...

============================================================ 3 passed in 0.06␣
↪seconds ============================================================
___________________________________ summary ____________________________________
  py27: commands succeeded
  congratulations :)
```

- Build a distribution
- Release on Pypi with twine (you can also code a specific detailed workflow in your setup.py)

2.2.3 Continuous Testing Workflow

Because no software works until it has been tested, you should configure travis on your repository to run test with each commit and pull request.

Catkin testing can be done with a simple *.travis.yml* file and a small shell script.

A matrix build can be setup to test behavior in multiple python virtualenvs. Using tox for this is generally a good idea, an example is there <TODO>.

2.3 Hybrid Catkin-pip Package

Ref : http://wiki.ros.org/ROS/Tutorials/catkin/CreatingPackage

We are introducing here a hybrid way of working with python packages in catkin environment.

These Packages are structured to support both catkin workflow and python workflow. They can be built either from an existing python package, or from an existing catkin package.

Because of the dynamic nature of python, we do not need to care about message type definition (pyros deals with it dynamically). So we will not consider these in this document.

The dependencies of the package can be handled by:

- pip (using setuptools.setup ‘install_requires’ parameter and requirements.txt files)
- rosdep (using depend keys in packages.xml).

This makes it easier to :

- write ROS python code using recent python habits and dependencies.
- test your package with different version of packages available on pypi.

- specialize an existing python repository to integrate in your ROS development environment.
- generalize a ROS package to a more "high level" "system independent" python package.

An important limitation here is that this package will only work from source (inside a catkin devel workspace), and cannot be made into a ROS package, until all dependencies are resolvable via rosdep. This is due to the complexity of managing multiple package managers with different strategies and policies in place.

However it is perfectly releasable on pypi, just like any python package, provided that all your ROS dependencies are optional.

An example of the ros/python hybrid workflow is the pyros package : https://github.com/asmodehn/pyros Having a way to retrieve ros packages from the python install is a long term goal.

This workflow is usually a transition workflow during the conversion from a python package (usable in source with catkin pip) to a ROS package (deployable via debs). As such it is important to test both the ROS devel workflow, and the ROS install workflow, separately. pyros has such a setup.

The catkin devel and install workflow rely on rosdep for depencencies and the multiple pure python workflow rely on pip packages for dependencies. Therefore great care is needed to support interoparable version of all dependencies in all the different platforms.

This documents describes a hybrid package. For the step by step process to convert a python package into a ROS package, follow this documentation <TODO>.

2.3.1 Package Structure

This is a generic example inspired by the hybrid catkin-pip package from Pyros : pyros-test

Files hierarchy

Dependencies

2.3.2 Package Development Workflow

This is a description of the generic ROS catkin workflow to develop, build, test and release a catkin based package. We will use this as the reference when implementing catkin_pip improvements

- Build with catkin_make. this will use pip.
- Source devel space. this will source extra python environments
- Run tests with catkin_make test. This will use latest tests frameworks
- Debug tests with catkin_make run_tests. This will use latest tests frameworks

From a different shell to not have your environment polluted with devel space:

- Install with catkin_make install. This will use latest setuptools
- Source the install space. This will
- Run tests as a final user would.

Release

- catkin_generate_changelog OR gitchangelog
- catkin_prepare_release OR manual changes (or your own way in setup.py)
- pypi upload with twine

- bloom-release –rosdistro indigo –track indigo <package_name>

2.3.3 Continuous Testing Workflow

Because no software works until it has been tested, you should configure travis on your repository to run test with each commit and pull request.

Catkin testing can be done with a simple *.travis.yml* file and a small shell script. One important part is to separate the devel and the install flow to make sure dependencies used in the devel workspace wont affect the install test (which should use hte system dependencies)

A matrix build can be setup to test behavior in ROS as well as in python virtualenvs. An example is there https://travis-ci.org/asmodehn/pyros.

CHAPTER 3

Conversion : From python to catkin to python

Where we follow a step by step process to convert your existing code to be used in both catkin and python at the same time.

3.1 From Python to Catkin

This document should mirror the from_catkin_to_python document.

3.1.1 Step 1 : Hybrid Package for devel

The goal is to integrate an existing python package into a catkin environment. This is available right away for a devel workspace, you just need to create a CMakeLists.txt using catkin_pip, and a matching package.xml for catkin to detect it. you should compare the python package structure and the hybrid package structure to become familiar with the differences.

Running *catkin_make* will trigger a workflow very similar to the usual python workflow.

This makes it easier to :

- write a setup.py without requiring catkin to parse it.
- fork an existing python code to integrate into a ROS package, provided that all its dependencies are available via rosdep.

An example of the normal python package is the pyros_setup package : https://github.com/asmodehn/pyros-setup

An example of the basic catkin_pip workflow is the pyros_utils package : https://github.com/asmodehn/pyros-utils

For more details, you should check our reference python workflow there <TODO> and compare it with the hybrid workflow there <TODO>.

Your dependencies are still handled by pip in the devel workspace, so you can get started with using your package in catkin quickly.

3.1.2 Step 2 : Dependencies adjustments

Once your existing python package works fine in source, you can adjust your dependencies version to make sure all dependencies can be found on the system, or in ROS packages. All these dependencies should be resolvable from packages.xml, and installable via *rosdep*.

This does mean there will be duplicated dependencies in package.xml and in setup.py. It might be unsuitable in some cases (same versions everywhere), but necessary in some others (different version on different systems), so this is currently the best option.

Finally you should be able to create a ROS package. Be careful to test it, in order to verify all system dependencies are working.

For more details about setting up travis tests for a hybrid package, check <TODO>.

3.1.3 Step 3 : Third Party Package (optional)

You can work with your current hybrid package, if :

- you have write access to the original python package repository, you don't mind adding a few files, and you can make the hybrid package structure there.
- you have forked the original python package repository, and you are happy to do the maintenance of the versions in python versus the versions in ROS.

However if you don't have write access to the original repository, or you don't want the hassle of maintaining all different versions between the ROS distributions and the python environments, you should consider building a Third Party package. Ref : http://wiki.ros.org/bloom/Tutorials/ReleaseThirdParty

It is useful especially for all the python dependencies you might use in your project, but on which you have no control, except choosing the version you d like to use in you ROS distro.

Build a Third Party package from Hybrid Package

This is achieved by extracting the files useful only for catkin (CMakeLists.txt, package.xml, rosdoc.yml) from the hybrid package made before, and putting them in the rosrelease repository created by bloom when releasing a ROS package.

Be aware that now, your source package cannot be built in a catkin workspace, and you should use:

- The ROS package as a dependency in you parent catkin project, via rosdep (using depend keys in packages.xml)
- The pip package as a dependency in you parent python project, via pip (using setup.py install_requires and requirements.txt)

This makes it easier to :

- write a ROS python package using recent python habits and dependencies
- test your package with different version of packages available on pypi in your development environment.
- test the package on multiple python environment from the source repo (using travis or other build providers)
- test the package on multiple ROS distros from the release repo (using travis or other build providers)
- synchronize release on pypi for python users and on ros buildfarm for ros users.

An example of the ros/python hybrid workflow is the pyzmp package : https://github.com/asmodehn/pyzmp This is a pure python package, imported into ROS ecosystem as is, in a deb package so other ROS packages can depend on it.

To check how the release repository itself can be checked with travis to ensure the package tests are still passing when in a ROS environment. A few examples from the pyros dependencies:

- https://travis-ci.org/asmodehn/pyros-config-rosrelease
- https://travis-ci.org/asmodehn/pyzmp-rosrelease

Again, the third party package release is sometime not necessary. If your package doesnt have to be distributed via ROS package system, having a simple pip package is enough. https://github.com/asmodehn/pyros-setup is an example of this : it is a simple pip package, that is retrieved and used only via pip. Your package can use it as a dependency to access ROS modules from a python virtual env. For more user interaction with ROS from pure python, you want to have a look at https://github.com/asmodehn/pyros itself (provided as a pip package and a ROS package)

3.2 From Catkin to Python

This document should mirror the from_python_to_catkin document.

3.2.1 Step 1 : Hybrid Package

The goal is to generalize an existing catkin package into a python environment. However this project is only about build environments. For :

- a solution to find dependencies at runtime, check pyros_setup
- a solution to discover message type and convert them dynamically, check pyros

In a catkin devel workspace, you should :

- edit your CMakeLists.txt to use catkin_pip
- edit your setup.py, in order to use latest setuptools, and not the obsolete distutils required by catkin. Note you can leave the dependencies in package.xml to be resolved by rosdep in this first step. We can deal with that in Step 2.

You should compare the python package structure and the hybrid package structure to become familiar with the differences.

Running *catkin_make* will trigger a workflow very similar to the usual catkin workflow.

This makes it easier to :

- write a generic setup.py without requiring catkin to parse it.
- integrate recent python habits into catkin workflow.

An example of the normal catkin package is the pyros_test package : https://github.com/asmodehn/pyros-test

An example of the basic catkin_pip workflow is the pyros_utils package : https://github.com/asmodehn/pyros-utils

For more details, you should check our reference python workflow there <TODO> and compare it with the hybrid workflow there <TODO>.

After you successfully build your package with catkin_pip, your dependencies can be handled by pip in the devel workspace, so you can get started with adjusting your dependencies for python.

3.2.2 Step 2 : Dependencies adjustments

You should refer to the python way of handling dependencies in package_catkin_pip, and start adding your dependencies in requirements.txt or in setup.py.

This does mean there will be duplicated dependencies in package.xml and in setup.py. It might be unsuitable in some cases (same versions everywhere), but necessary in some others (different version on different systems), so this is currently the best option.

Finally you should already be able to create a python package, and even publish it on pypi. Be careful to test it, in order to verify all python dependencies are working.

For more details about setting up travis tests for a hybrid package, check <TODO>.

3.2.3 Step 3 : Python Enhancements (optional)

There are multiple ways to improve your package, using recent python tools. In random order:

- bump up your dependencies to use more recent version than the ones provided with your system or with ROS.
- use a recent test framework like pytest.
- use tox to test multiple python environment at once.
- use read-the-docs to publish your documentation for all to see.
- upgrade your workflow by using setup.py for preparing a release.
- publish your package on Pypi for all to use.

3.2.4 Step 4 : Third Party Package (optional)

You can work with your current hybrid packge, if :

- you have write access to the original catkin package repository, you don't mind adding a few files, and you can make the hybrid package structure there.
- you have forked the original catkin package repository, and you are happy to do the maintenance of the versions in ROS versus the versions in pypi.

However if you don't have write access to the original repository, or you don't want the hassle of maintaining all different versions between the ROS distributions and the python environments, you should consider building a Third Party package. Ref : http://wiki.ros.org/bloom/Tutorials/ReleaseThirdParty

It does mean that the source repository will be the python code repository, as the python code should work cross-platform. What needs to be adjusted is the catkin build to make a ROS system package out of the python package. This is doable via small patches in the release repository, along with a more "generic code" handling all differences in the environment.

For more details about this step, you should consult *Build a Third Party package from Hybrid Package*.

CHAPTER 4

IDE setup

4.1 PyCharm Setup

This describe how to setup pycharm to work on your ROS workspace(s)

4.1.1 Basic ROS workflow

The environment for your project is not a specific environment per Project/Repository/Package, but rather one environment per Workspace. => The project for Pycharm is at the workspace level.

- Basic ROS commands and python commands will work from the terminal, after sourcing the devel space as usual with ROS *source devel/setup.bash*
- Pycharm integrated python tools will work, but only if you have sourced the devel space before launching PyCharm, which might not be convenient in some cases…

BEWARE : PyCharm (and Python imports ?) have some issues with symlinks, so you should avoid them if possible when setting up your workspace.

4.1.2 Hybrid workflow

This assumes you are developing ROS packages at the same time as Python packages, in the same workspace. Both workflow need to be combined in order to develop and test with ROS and simultaneously develop and test with Python.

- ROS still finds all modules after a *source devel/setup.bash*
- Python (system python) configured in PyCharm will find all modules (from system or from workspaces) using pyros_setup. This is useful to use all PyCharm tools for python, even without sourcing the environment yourself, pyros_setup will do that for you. You will need to configure pyros_setup appropriately to describe your environment.

TODO : check pip requirements to retrieve git clones. TODO : check working on ROS workspace + independent python projects in same Pycharm window.

4.1.3 Python workflow (but still using ROS)

You can work as usual with Python. Pyros_setup will use the configuration provided to setup your ROS environment and be able to import it.

CHAPTER 5

Indices and tables

- genindex
- modindex