ROS BUNDLING

MIKE PURVIS

CLEARPATH

THE HISTORY

ros-install-osx, the evolution of development and deployment at Clearpath

THE PROBLEM

stable releases for industrial customers in a rolling release world

THE BUNDLE

the tools which power our solution, and some unanticipated benefits

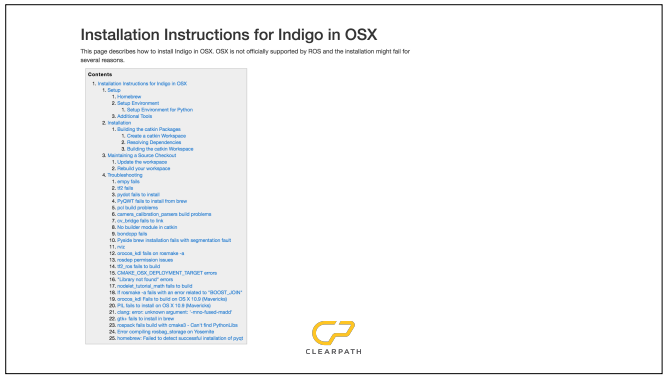
THE FUTURE

how the tooling can further evolve to better to support this deployment model

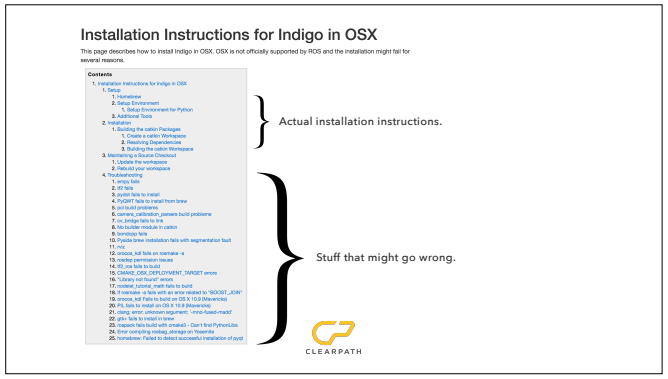
CLEARPATH

HISTORY

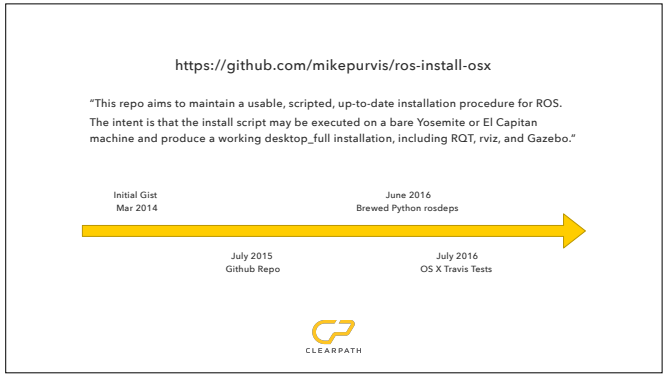
CLEARPATH



Building ROS Indigo on Mac OS X, two years ago.



The number of potential problems dwarfs the main body of the setup instructions.



New approach used brewed python/numpy/scipy instead of the system versions of those packages, and built in parallel with catkin_tools, rather than catkin_make_isolated. Started life as a gist, eventually evolved into runnable script. Positive side effect has been a centralized bug-tracker and place to review and collaborate on fixes with a growing community of ROS Mac users.

```
rosinstall_generator desktop_full --rosdistro indigo
  --deps --tar > desktop_full.rosinstall

wstool init -j8 src desktop_full.rosinstall

rosdep install --from-paths src --ignore-src --rosdistro indigo
  --as-root pip:no -y

catkin config --install

catkin build

source install/setup.sh
```

Process is:

1. Generate a list of all packages which will be in the workspace.
2. Download the tarball for each package.
3. Install system dependencies using rosdep (stuff like CMake, Boost, PCL).
4. Giant parallel build!

```
[build] Found '225' packages in 0.0 seconds.
Starting  >>> catkin
Finished  <<< catkin                                [ 0.8 seconds ]
Starting  >>> genmsg
Starting  >>> angles
Starting  >>> cmake_modules
Starting  >>> convex_decomposition
Finished  <<< genmsg                                [ 0.8 seconds ]
Starting  >>> cpp_common
Finished  <<< angles                                [ 0.7 seconds ]
Starting  >>> eigen_stl_containers
Finished  <<< cmake_modules                         [ 0.5 seconds ]
Starting  >>> ivcon
Finished  <<< convex_decomposition                  [ 0.9 seconds ]
Starting  >>> media_export
```

This is what the build looks like in catkin_tools. We're not going to dwell here except to say that a catkin_tools is equivalent in approach to catkin_make_isolated, meaning that each package is configured as its own project, rather than being combined together. This is critical because plain CMake packages like orocos_kdl and catkin itself can't be combined into one workspace the way catkin packages can.

---

desktop_full.rosinstall

```
- tar:
    local-name: actionlib
    uri: https://github.com/ros-gbp/actionlib-release/archive/release/indigo/actionlib/1.11.6-0.tar.gz
    version: actionlib-release-release-indigo-actionlib-1.11.6-0
- tar:
    local-name: angles
    uri: https://github.com/ros-gbp/geometry_angles_utils-release/archive/release/indigo/angles/1.9.10-0.tar.gz
    version: geometry_angles_utils-release-release-indigo-angles-1.9.10-0
- tar:
    local-name: bond_core/bond
    uri: https://github.com/ros-gbp/bond_core-release/archive/release/indigo/bond/1.7.17-0.tar.gz
    version: bond_core-release-release-indigo-bond-1.7.17-0
...
```

This is a snippet of what's inside the rosinstall file that wstool consumes. It's a list of tarball links which will be downloaded and unpacked to populate the source workspace.

ros/rosdistro/indigo/distribution.yaml

```
actionlib:
  release:
    tags:
      release: release/indigo/{package}/{version}
    url: https://github.com/ros-gbp/actionlib-release.git
    version: 1.11.6-0
bond_core:
  release:
    packages:
    - bond
    - bond_core
    - bondcpp
    - bondpy
    - smclib
```

CLEARPATH

The rosinstall_generator tools uses as its source data the distribution.yaml which governs which version of every package is the current latest in a given distribution. So in this case, version 1.11.6-0 was the last bloomed release of actionlib into ROS Indigo. The build farm uses this data to generate Jenkins jobs to build debian packages, but rosinstall_generator uses it to generate source workspaces.

---

ros/rosdistro/indigo/distribution.yaml

```
actionlib:
  release:
    tags:
      release: release/indigo/{package}/{version}
    url: https://github.com/ros-gbp/actionlib-release.git
    version: 1.11.6-0
bond_core:
  release:
    packages:
    - bond
    - bond_core
    - bondcpp
    - bondpy
    - smclib
```

https://github.com/ros-gbp/actionlib-release/archive/release/indigo/actionlib/1.11.6-0.tar.gz

CLEARPATH

The generator is capable of generating a list of git URLs, but downloading tarballs is much faster than cloning git repos, so there is logic to recognize known git hosts and supply tarball URLs with the --tar flag.

```
rosinstall_generator desktop_full --rosdistro indigo --deps --tar
```

CLEARPATH

This used to support github only, but we patched it to also recognize bitbucket and gitlab. Now the other missing piece here is the --deps flag. How does rosinstall_generator know which packages to pull in when you call for dependencies?

http://repositories.ros.org/rosdistro_cache/<distro>-cache.yaml.gz


```
  action_relay: <?xml version="1.0" encoding="utf-8"?><package format="2"><name>action_relay</name><version>0.5.
1</version><description>Handles action server exceptions during loss of network</description><maintainer email="
msafri@clearpathrobotics.com">msafri</maintainer><license>Clearpath Proprietary</license><buildtool_depend>catki
n</buildtool_depend><depend>actionlib</depend><depend>actionlib_msgs</depend><depend>multimaster_msgs</depend><d
epend>roscpp</depend><test_depend>roslint</test_depend></package>
  actionlib: <?xml version="1.0" encoding="utf-8"?><package><name>actionlib</name><version>1.11.5</version><desc
ription>The actionlib stack provides a standardized interface for interfacing with preemptable tasks. Examples o
f this include moving the base to a target location, performing a laser scan and returning the resulting point c
loud, detecting the handle of a door, etc.</description><maintainer email="mikael@osrfoundation.org">Mikael Argu
edas</maintainer><license>BSD</license><url type="website">http://www.ros.org/wiki/actionlib</url><url type="bug
tracker">https://github.com/ros/actionlib/issues</url><url type="repository">https://github.com/ros/actionlib</u
rl><author>Eitan Marder-Eppstein</author><author>Vijay Pradeep</author><buildtool_depend version_gte="0.5.78">ca
tkin</buildtool_depend><build_depend>actionlib_msgs</build_depend><build_depend>boost</build_depend><build_depen
d>message_generation</build_depend><build_depend>roscpp</build_depend><build_depend>rospy</build_depend><build_d
epend>rostest</build_depend><build_depend>std_msgs</build_depend><run_depend>actionlib_msgs</run_depend><run_dep
end>boost</run_depend><run_depend>message_runtime</run_depend><run_depend>roscpp</run_depend><run_depend>rospy</
run_depend><run_depend>rostest</run_depend><run_depend>std_msgs</run_depend><test_depend>rosnode</test_depend></
package>
  actionlib_msgs: <?xml version="1.0" encoding="utf-8"?><package><name>actionlib_msgs</name><version>1.11.9</ver
sion><description>actionlib_msgs defines the common messages to interact with an action server and an action cli
ent. For full documentation of the actionlib API see the<a href="http://www.ros.org/wiki/actionlib">actionlib</a
>package.</description><maintainer email="tfoote@osrfoundation.org">Tully Foote</maintainer><license>BSD</licens
e><url>http://ros.org/wiki/actionlib_msgs</url><author>Vijay Pradeep</author><buildtool_depend version_gte="0.5.
78">catkin</buildtool_depend><build_depend>message_generation</build_depend><build_depend>std_msgs</build_depend
><run_depend>message_generation</run_depend><run_depend>message_runtime</run_depend><run_depend>std_msgs</run_de
pend><export><architecture_independent/></export></package>
```

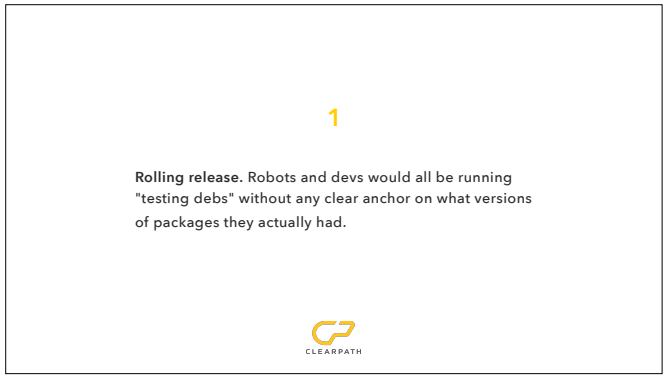

Every few minutes the buildfarm examines the distribution.yaml and updates a *secondary* yaml which is the contents of the distribution plus an additional dictionary which contains the complete package.xml string for every package. By accessing this cache, rosinstall_generator is able to quickly determine recursive dependencies.

So. That's ros-install-osx. The second piece of history is that Clearpath has executed a pivot in the past two years, transitioning from a research products company to focusing primarily on a full-stack solution for autonomous mobile robots in the industrial logistics space. This has meant a bunch of changes in how we deploy which are what have primarily driven the development of our bundle pipeline.

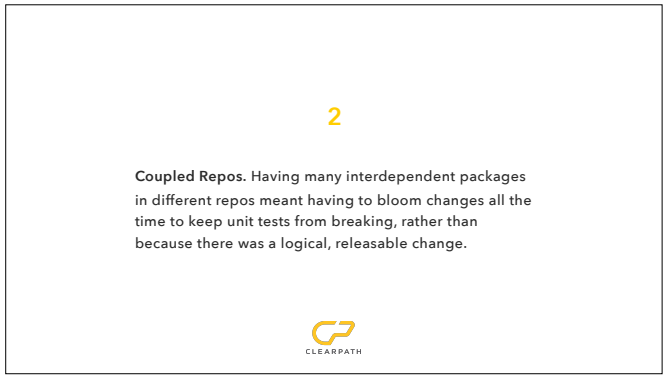
THE PROBLEMS

CLEARPATH

About a year ago, I officially joined the industrial team, as Firmware and Integration manager. Deployments to robots, developers, simulation environments were all based on bloomed deb packages as generated by buildbot-ros and then later ros_buildfarm. We had the following issues with this approach.



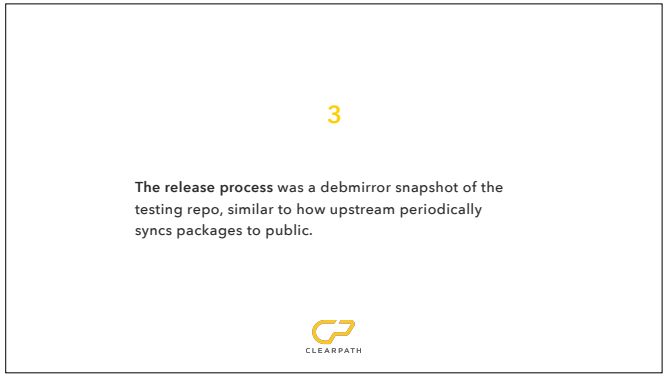
We had scripts which would use rospack list or rosversion to at least *report* the versions on a machine at a given time (for example, when a problem was observed, or a crash occurred), but knowing is only half the battle—it doesn't help you *reproduce* a bad state when trying to debug it later.



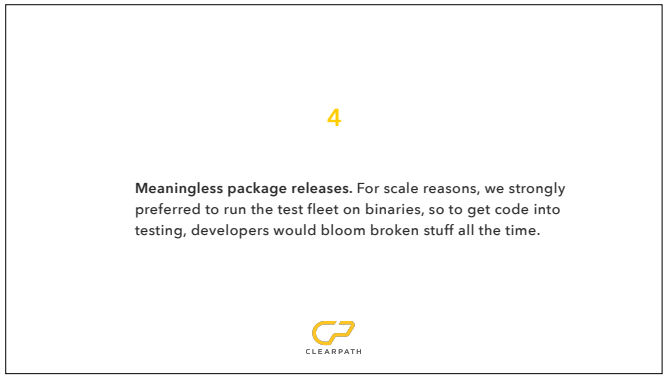
Because ros_buildfarm sets up the test environment by installing the latest *released* version of all test dependencies.

The easy argument here is "don't have so many repos," but this is not so simple, particularly when you consider our relationship to upstream. Take our controls code, which is spread across three repos: navigation, the planner, and the tracker, where navigation is a fork of upstream, and the planner and the tracker are proprietary. At this point our navigation fork is permanent and permanently private, but a year ago there was still a vague ambition to try to merge our

---



Meaning that there was no sane way to put incremental bugfixes on a build — you were either on the bleeding edge or a stale snapshot. We experimented with injecting one-off bug fix debs into a release, but it was a very manual, error-prone process.



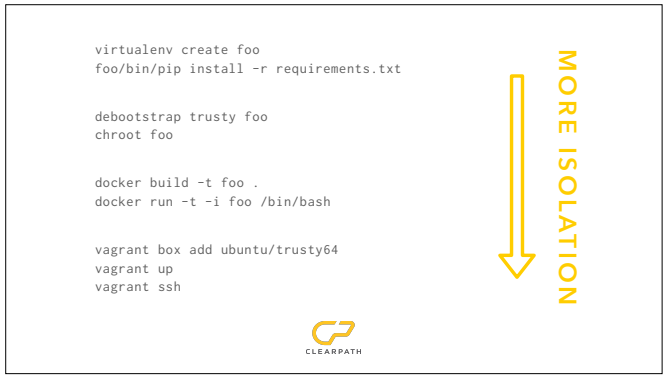
Part of how I justify getting sucked into this project despite being the “firmware” manager is that we increasingly view the entire robot as an embedded target to be flashed, rather than a Linux PC on wheels with an ordinary computer lifecycle.

This was the proposal that my colleagues and I discussed as a starting point.

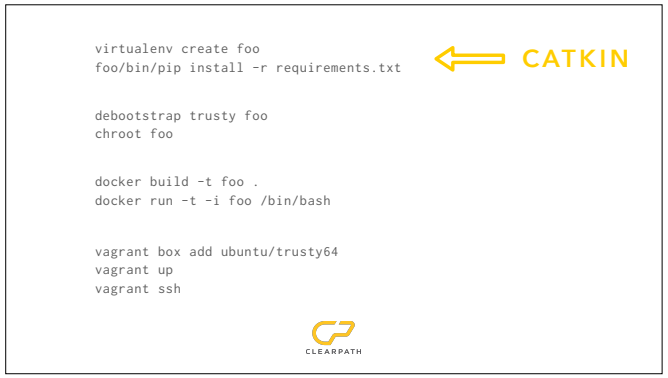
Based on ros-install-osx— just build everything! Why bother separating out an overlay?

We need versioning and a distribution scheme; let's just use a debs and an apt repo; those are already a thing.

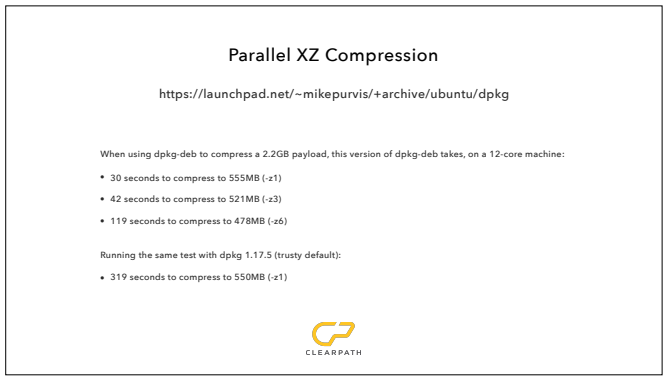
So your first question might be "seriously, a deb package, in this age of sexy things like containers?"



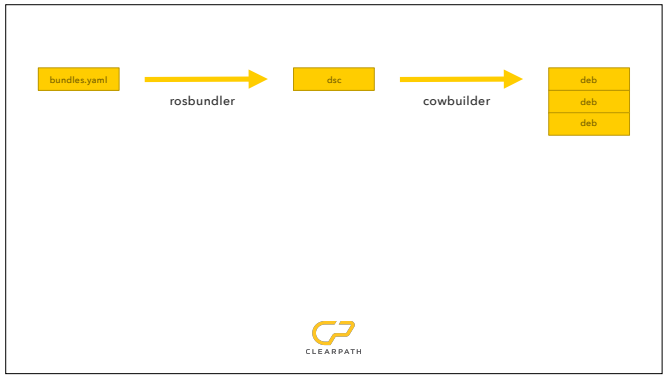
And it's true, isolation is a big thing right now. Type and degree of isolation varies.



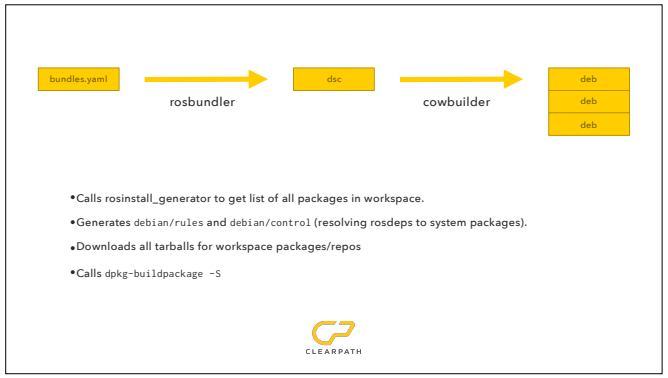
Base catkin is isolated in terms of filesystem only. Sort of a virtualenv for C++ CMake projects. We're not doing multiple apps, and port or CPU isolation would just get in our way, so we don't need a docker container— we can ship this thing as a deb, and then there's a lot of additional tools we can take advantage of in the debian ecosystem which don't exist for docker (and we don't have to run the daemon).



If you're like me and you suddenly find yourself generating a bunch of large-ish debs, you'll quickly discover that the default compression for dpkg-deb is single-threaded xz, which can be extremely slow. Add this PPA to your 8 or 16 core buildslave and watch it scream through that end-of-build compression step.



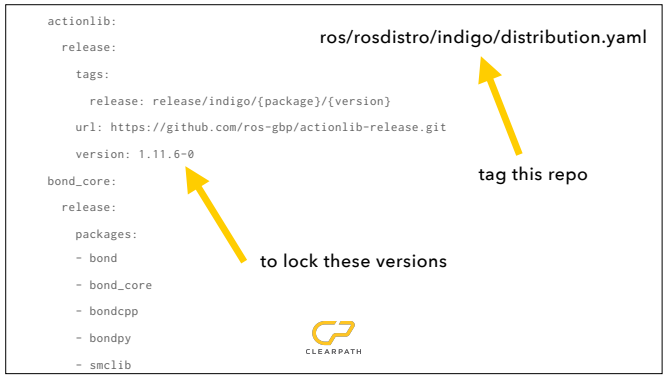
So, here's basically our process. A bundle config file defines what our bundles are, where the rosdistro for each one is, and what packages should be used to "seed" rosinstall_generator.



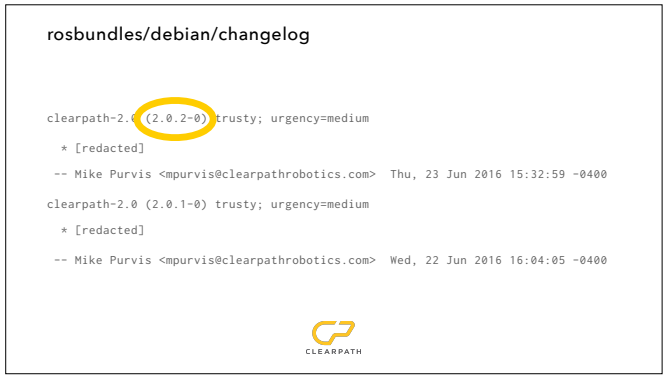
Our rosbundler tool generates the required debian package metadata files, including instructions for how to build everything (the debian/rules file), and then downloads the tarballs, naming them according to debian conventions.

Keen observers will note that this doesn't actually address a lot of the problems listed above. By default, rosinstall_generator is always grabbing the latest released version of every package, so this bundle is equivalent to whatever the testing debs are at the moment that the bundle is built. How do we actually lock in the versions of the packages?

Well, here's the distribution file which actually defines the versions of everything. This lives in a git repo, and a git repo can be tagged.

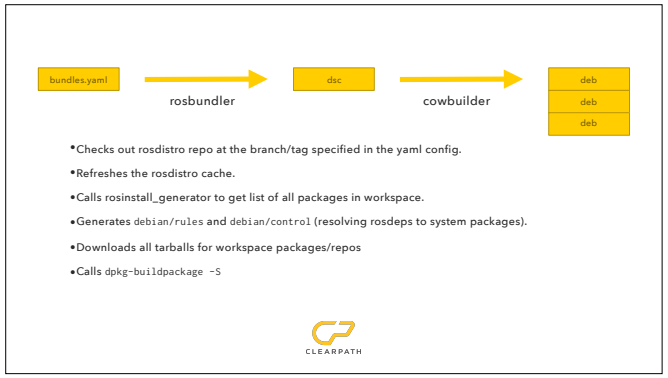


So we'll tag the repo with the number of our major release, and that'll forever give us a reference point to regenerate this configuration.



In fact, we use a regular Debian changelog file which lives alongside the bundles.yaml file, and the version specified in each changelog entry refers directly to a rosdistro tag.

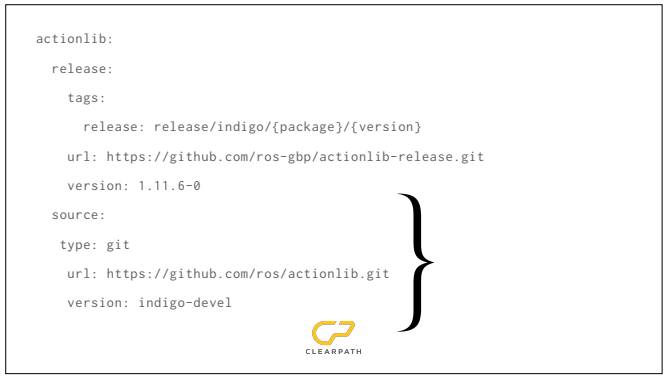
Big and bold so it can't be missed.



So to finish this out, when we're building a "release" bundle, we use the package versions from a tag of our rosdistro, rather than the master branch.

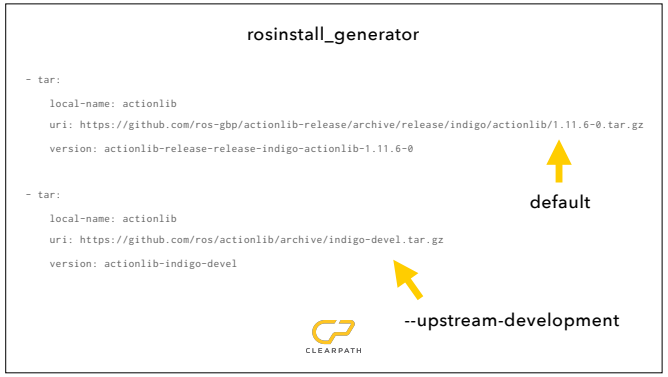
And prior to tagging a legit "release", we can still build a "prerelease", which is each package at its latest version. Eventually we may tag those as well— it would be nice to understand which repos have mutated when we're looking at a regression.

Now, we've talked a lot about the release stanza of the distribution.yaml entry— this is the entry whose git URL references a GBP repo, one managed by bloom.



But the source stanza is actually where the bundle really comes into its full potential. Let's look at a new rosinstall_generator flag.

Adding —upstream-development means that instead of getting the source from the bloom GBP repo, we get it directly from the source repo.



Above is what we get by default— this is the latest released source, pulled from the GBP repo. Underneath is with the —upstream-development flag, this is the development branch from the upstream repo.

Can you see what's happening here? There's a new type of bundle that's possible, where we don't need to bloom at all.

DEVEL	PRERELEASE	RELEASE
version is a timestamp	version is a timestamp	version from changelog
rosdistro from branch tip	rosdistro from branch tip	rosdistro from tag
upstream devel sources		

CLEARPATH

The devel bundle is built entirely of devel sources. Seems scary right? This has actually worked out phenomenally well for us, for a few reasons.

DEVEL

version is a timestamp

rosdistro from branch tip

upstream devel sources

- test fleet is always running the latest devel code.
- when something breaks upstream (eg, ros_comm), we catch it immediately.
- we're motivated to push patches, since we get the fix right away.
- we bloom for logical functional releases releases only.
- devel bundle can feed other testing workflows, including multi-robot simulation testing and unit test jobs.

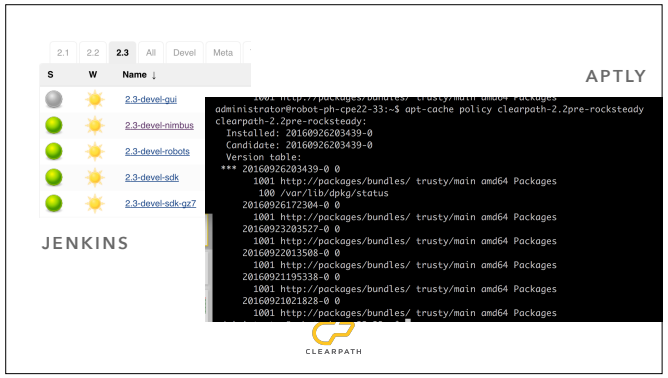
CLEARPATH

BUNDLE TYPE

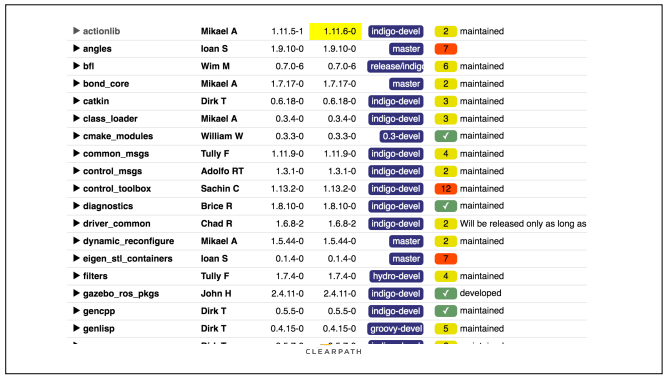
MAJOR VERSION

	devel	prerelease	release
2.0	clearpath-2.0devel-robots clearpath-2.0devel-nimbus clearpath-2.0devel-sdk	clearpath-2.0pre-robots clearpath-2.0pre-nimbus clearpath-2.0pre-sdk	clearpath-2.0-robots clearpath-2.0-nimbus clearpath-2.0-sdk
2.1	clearpath-2.1devel-robots clearpath-2.1devel-nimbus clearpath-2.1devel-sdk	clearpath-2.1pre-robots clearpath-2.1pre-nimbus clearpath-2.1pre-sdk	clearpath-2.1-robots clearpath-2.1-nimbus clearpath-2.1-sdk
2.2	clearpath-2.2devel-robots clearpath-2.2devel-nimbus clearpath-2.2devel-sdk	clearpath-2.2pre-robots clearpath-2.2pre-nimbus clearpath-2.2pre-sdk	clearpath-2.2-robots clearpath-2.2-nimbus clearpath-2.2-sdk

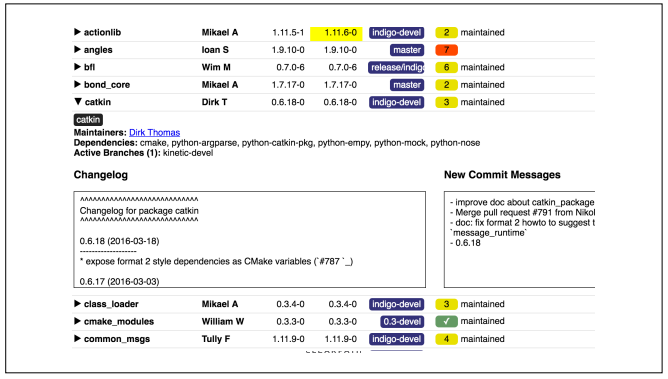
CLEARPATH



In terms of our actual infrastructure, we build the bundles on Jenkins, and then send them to Aptly. Aptly is super cool for many reasons— two of them are that it can run as a REST service, so you don't have to deal with SSH or SCP, you can just POST packages to it with curl if you want.



This is our build status page, which is source repo oriented as opposed to the build farm one, which is deb package oriented.



Each row pops open to reveal the changelog and commits which have occurred between the last release and the current level branch.

```
$ rosdistro_freeze_source -h

Freeze a rosdistro's source branch versions to hashes or tags. If
neither

--release-version nor --release-tag are specified, the hashes of the
current

devel branches are used.


positional arguments:

  index                 Path to a local index.yaml file.

  dist_names            The names of the distributions (default: all)


optional arguments:

  -h, --help            show this help message and exit

  -j JOBS, --jobs JOBS  How many worker threads to use.

  -q, --quiet           Suppress updating status bar (for script/CI
usage).

  --release-version     Freeze to the hash of current release tag.

  --release-tag         Freeze to name of current release tag.
```

---

rosdistro_freeze_source

```
actionlib:
  source:
   type: git
    url: https://github.com/ros/actionlib.git
    version: indigo-devel
```

CLEARPATH

---

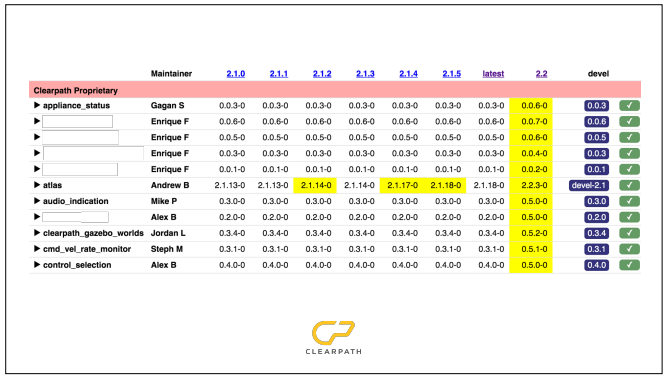
rosdistro_freeze_source

```
actionlib:
  source:
   type: git
    url: https://github.com/ros/actionlib.git
    version: indigo-devel
```

```
actionlib:
  source:
   type: git
    url: https://github.com/ros/actionlib.git
    version: 32ce9fdb3194228795a2f31faee8259dd57ae452
```

```
actionlib:
  source:
   type: git
    url: https://github.com/ros/actionlib.git
    version: 1.11.6
```

CLEARPATH



	Maintainer	2.1.0	2.1.1	2.1.2	2.1.3	2.1.4	2.1.5	latest	2.2	devel	
Clearpath Proprietary											
▶ appliance_status	Gagan S	0.0.3-0	0.0.3-0	0.0.3-0	0.0.3-0	0.0.3-0	0.0.3-0	0.0.3-0	0.0.6-0	0.0.3	✓
▶ [redacted]	Enrique F	0.0.6-0	0.0.6-0	0.0.6-0	0.0.6-0	0.0.6-0	0.0.6-0	0.0.6-0	0.0.7-0	0.0.6	✓
▶ [redacted]	Enrique F	0.0.5-0	0.0.5-0	0.0.5-0	0.0.5-0	0.0.5-0	0.0.5-0	0.0.5-0	0.0.6-0	0.0.5	✓
▶ [redacted]	Enrique F	0.0.3-0	0.0.3-0	0.0.3-0	0.0.3-0	0.0.3-0	0.0.3-0	0.0.3-0	0.0.4-0	0.0.3	✓
▶ [redacted]	Enrique F	0.0.1-0	0.0.1-0	0.0.1-0	0.0.1-0	0.0.1-0	0.0.1-0	0.0.1-0	0.0.2-0	0.0.1	✓
▶ atlas	Andrew B	2.1.13-0	2.1.13-0	2.1.14-0	2.1.14-0	2.1.17-0	2.1.18-0	2.1.18-0	2.2.3-0	devel-2.1	✓
▶ audio_indication	Mike P	0.3.0-0	0.3.0-0	0.3.0-0	0.3.0-0	0.3.0-0	0.3.0-0	0.3.0-0	0.5.0-0	0.3.0	✓
▶ [redacted]	Alex B	0.2.0-0	0.2.0-0	0.2.0-0	0.2.0-0	0.2.0-0	0.2.0-0	0.2.0-0	0.5.0-0	0.2.0	✓
▶ clearpath_gazebo_worlds	Jordan L	0.3.4-0	0.3.4-0	0.3.4-0	0.3.4-0	0.3.4-0	0.3.4-0	0.3.4-0	0.5.2-0	0.3.4	✓
▶ cmd_vel_rate_monitor	Steph M	0.3.1-0	0.3.1-0	0.3.1-0	0.3.1-0	0.3.1-0	0.3.1-0	0.3.1-0	0.5.1-0	0.3.1	✓
▶ control_selection	Alex B	0.4.0-0	0.4.0-0	0.4.0-0	0.4.0-0	0.4.0-0	0.4.0-0	0.4.0-0	0.5.0-0	0.4.0	✓

CLEARPATH

Once a major release for us goes into maintenance, we “freeze” the devel branches to point to the tag of the last release. This allows development to move forward, while still giving us a stable level build upon which to put hot fixes.

DEMONSTRATION

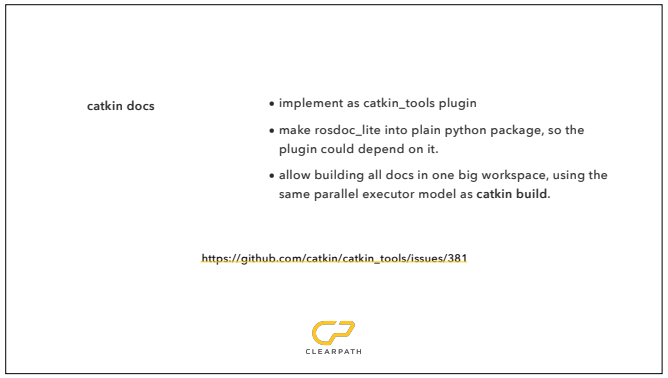
https://github.com/mikepurvis/ros-bundling

CLEARPATH

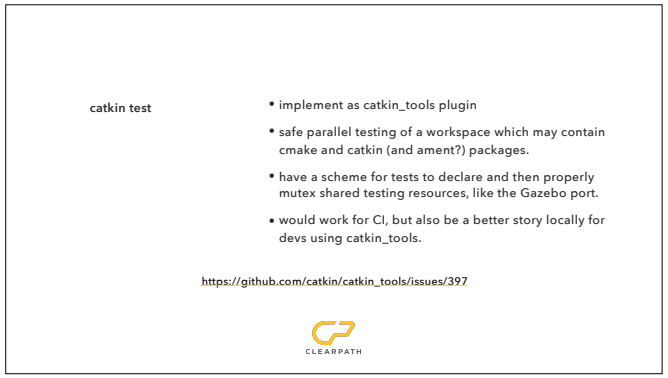
THE FUTURE

CLEARPATH

catkin test, catkin docs.
dropping release stanzas and blooming, consi.
once the ubuntu store is open source, investigate packaging as a snap.



One of the big wins with this approach would be potentially giving more opportunities to supply global overrides. So, same as how building the bundle lets you do fun things like set compiler flags that correspond to the architecture of your robot PC, or archive debug symbols, a future catkin docs might allow injecting a Doxygen theme, etc.



One of the other major functional gaps in catkin_tools right now is not having a good story for running tests.

source only

- drop release stanzas from internal distribution yaml.
- only bloom packages which are going for public release, otherwise do *only* catkin_prepare_release to uprev package.xml and create tag.

```
actionlib:
  source:
    type: git
    url: https://github.com/ros/actionlib.git
    version: indigo-devel
    release_version: 1.11.6
```

CLEARPATH

Snaps

- Packaging as a squashfs is compelling.
- Delta updates are compelling.
- Multiple release streams (edge, RC, etc) is *very* compelling.
- Blocked on the repository code being open sourced.

CLEARPATH

THANKS

William and Dirk

Jon Bohren

Nikolaus Demmel

Mike Ferguson

Contributors to ros-install-osx

Clearpath

CLEARPATH https://ottomotors.com/company/careers

OSRF staff for maintaining the tools and entertaining my crazy ideas and PRs.
Jon for re-architecting the catkin_tools backend.
Nikolaus for helping with OS X issues, and contributing the no-sudo option to rosdep.
Mike Ferguson for buildbot-ros and getting me into debian packaging and cowbuilder.
Clearpath for being an awesome place to work, and for allowing me time to work on this, and especially my colleagues on the platform and tools/ infrastructure teams. Obviously, we are hiring.