# The ROS build farm

## What it can do for me

### (And how it does it)

Oct. 9th 2016

Dirk Thomas

ROSCon 2016, Seoul, Korea

Open Source Robotics Foundation

# build.ros.org

Create easy-to-
install packages

Run tests before /
after code changes

Generate API doc
and meta information

# Central Service vs. Distributed

**You could**

- create Debian packages yourself
- run the test using e.g. Travis CI
- generate API documentation yourself

**But**

- setting up all these processes is quite some effort
  - having someone else do it is great for the users
- the artifacts should be hosted in a central place
- if you download "official" binaries you know what you are getting

# Central Service But Distributed Repositories

## A ROS distribution consists of numerous packages

- various vcs types
- various hosting services

**rosdistro/index.yaml**

```
distributions:
  ...
  kinetic:
    distribution: [kinetic/distribution.yaml]
    distribution_cache: <url to kinetic-cache.yaml.gz>
```

## github.com/ros/rosdistro

- for each ROS distribution
  - target platforms

    e.g. Kinetic: Ubuntu Xenial and Wily,

    Debian Jessie, (Fedora 23 and 24)
  - packages and versions
  - referencing all repositories
- format of yaml files specified in REP 143

**rosdistro/kinetic/distribution.yaml**

```
repositories:
  ...
  catkin:
    doc:
      ...
    release:
      ...
    source:
      ...

release_platforms:
  ...
  ubuntu:
  - wily
  - xenial
```

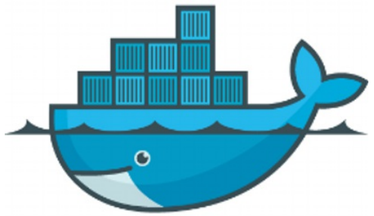
To use any of these services
you need to register your repositories

# The High-Level View

- Get the source code:
  e.g. `git clone`
- Install the required dependencies:
  using `rosdep`
- Build a package:
  `cmake`, `make`, `make install`
- Run the tests:
  `make test`
- Create a Debian package:
  `apt-src build`
- Generate documentation
  `rosdoc_lite`

# Some More Details

- Need to setup the environment for building
  - Install all dependencies (using binary packages)
  - Since each job has different dependencies each job needs to start with a "clean" state
  - Inside a Docker container
- Build the code from one repository
  - Might contain more than one ROS package
  - The repository is cloned into a catkin workspace
  - Invoke `catkin_make_isolated`
- Install the code
  - To check that the install step "works"
- Build and run the tests
  - Each test produces a xUnit-like result file



One container for multiple packages

Build vs. test dependencies

# `catkin_make` vs. `catkin_make_isolated`

While the Docker container contains the dependencies for all packages
Each package is still built "in isolation":

| | `catkin_make` | `catkin_make_isolated` |
|---|---|---|
| **CMake calls** | 1, for the whole ws | N, each pkg separately |
| **Side effects** | ⚠ Need to depend on other pkgs targets `catkin_EXPORTED_TARGETS` | - |
| **Location of artifacts** | Merged in a single folder<br>E.g. one include dir | Each pkg has its own folder |
| **Side effects** | ⚠ Package can implicitly access<br>e.g. the headers from other pkgs | - |
| **Pros** | Faster due to higher parallelization | Clean separation, better to identify problems |

❓ The build farm only builds single threaded

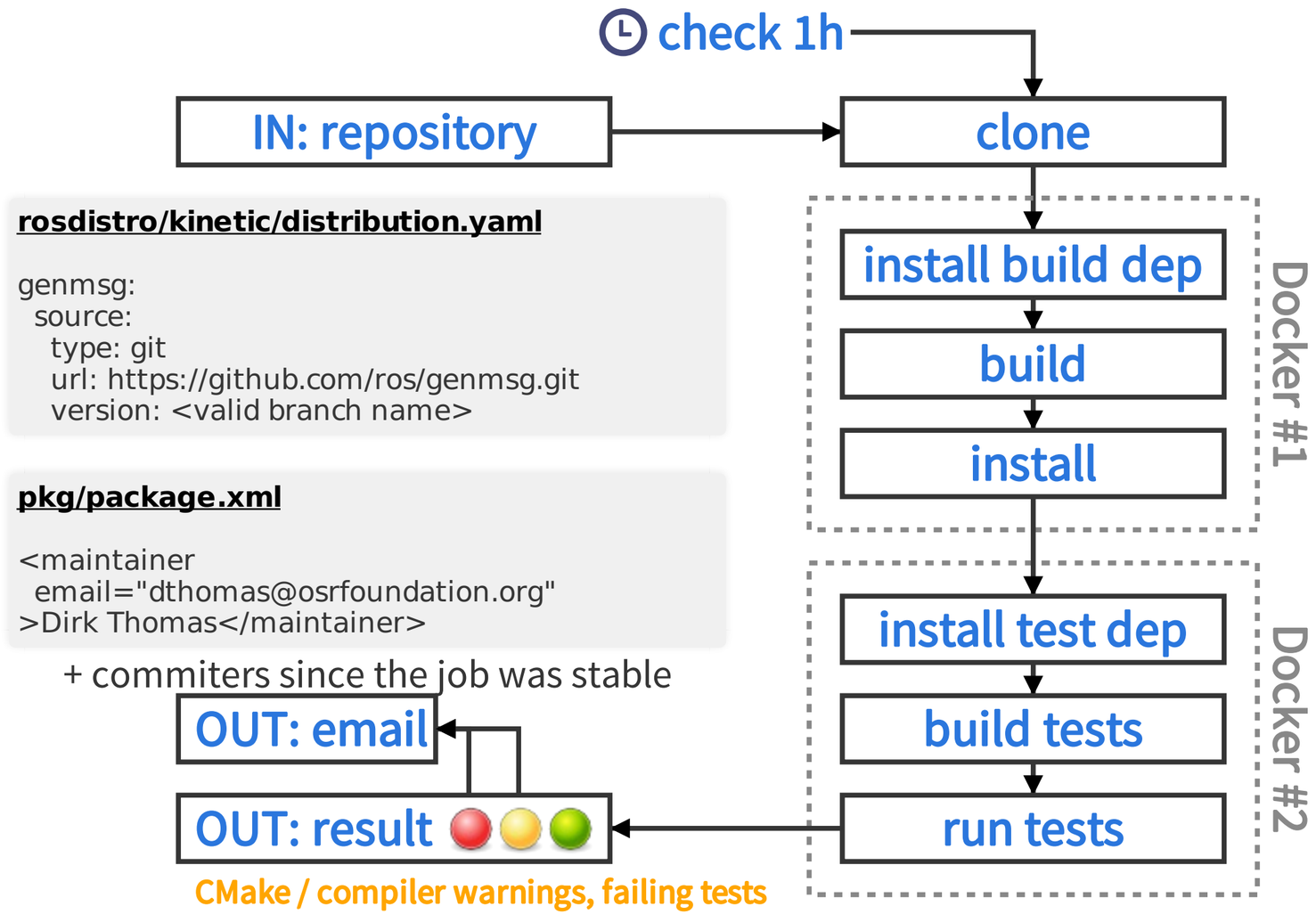
Create easy-to-
install packages

Run tests before /
after code changes

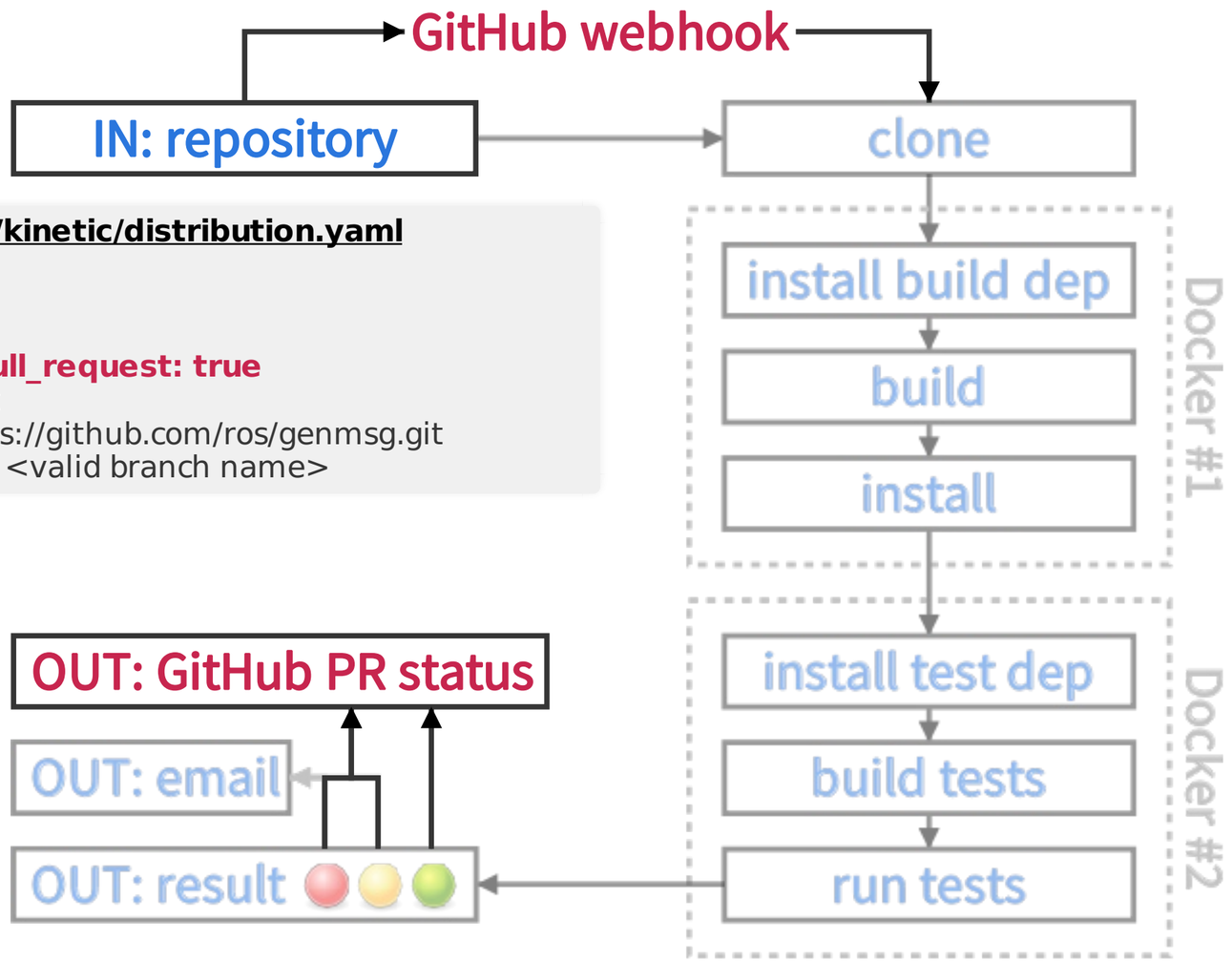


Generate API doc
and meta information

# Devel Jobs


check 1h
IN: repository
clone
rosdistro/kinetic/distribution.yaml
genmsg:
source:
type: git
url: https://github.com/ros/genmsg.git
version: <valid branch name>
pkg/package.xml
<maintainer
email="dthomas@osrfoundation.org"
>Dirk Thomas</maintainer>
+ commiters since the job was stable
install build dep
build
install
Docker #1
install test dep
build tests
run tests
Docker #2
OUT: email
OUT: result
CMake / compiler warnings, failing tests

# Pull Request Jobs



# Pull Request Jobs

**Limitations**

- `build.ros.org` only supports GitHub at the moment
- Jenkins GitHub user `ros-pull-request-builder` needs access to the org unit

  wiki.ros.org/buildfarm/Pull request testing

**Alternatives**

- Run the same process using any other CI provider
- E.g. Travis CI, see example `.travis.yml` file:

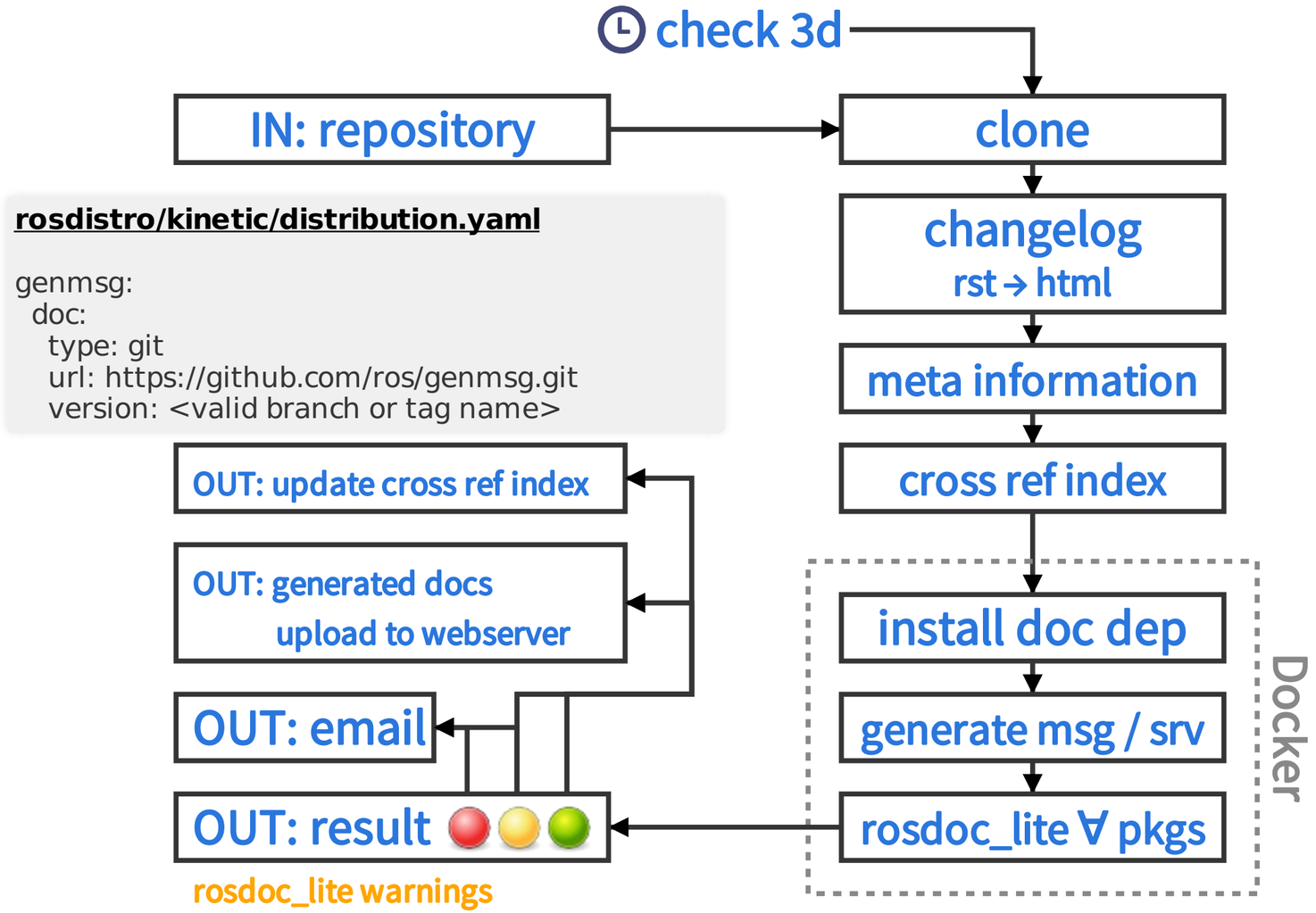
  docs in github.com/ros-infrastructure/ros_buildfarm

Create easy-to-
install packages

Run tests before /
after code changes

Generate API doc
and meta information

# Doc Jobs


check 3d
IN: repository
clone
changelog
rst → html
meta information
cross ref index
install doc dep
generate msg / srv
rosdoc_lite ∀ pkgs
Docker
rosdistro/kinetic/distribution.yaml
genmsg:
doc:
type: git
url: https://github.com/ros/genmsg.git
version: <valid branch or tag name>
OUT: update cross ref index
OUT: generated docs
upload to webserver
OUT: email
OUT: result
rosdoc_lite warnings

# Wiki Integration

## roscpp

electric | fuerte | groovy | hydro | indigo | jade | **kinetic** Documentation Status

***ros_comm***: *message_filters* | *ros* | *rosbag* | *rosconsole* | *roscpp* | *rosgraph* | *rosgraph_msgs* | *roslaunch* | *roslisp* | *rosmaster* | *rosmsg* | *rosnode* | *rosout* | *rosparam* | *rospy* | *rosservice* | *rostest* | *rostopic* | *roswtf* | *std_srvs* | *topic_tools* | *xmlrpcpp*

# Package Summary

✔ Released ✔ Continuous integration ✔ Documented

roscpp is a C++ implementation of ROS. It provides a client library that enables C++ programmers to quickly interface with ROS Topics, Services, and Parameters. roscpp is the most widely used ROS client library and is designed to be the high-performance library for ROS.

- Maintainer status: maintained
- Maintainer: Dirk Thomas <dthomas AT osrfoundation DOT org>
- Author: Morgan Quigley, Josh Faust, Brian Gerkey, Troy Straszheim
- License: BSD
- Source: git https://github.com/ros/ros_comm.git (branch: kinetic-devel)

**Package Links**

- **Code API**
- **Msg/Srv API**
- Tutorials
- FAQ
- Changelog
- Reviews

**Dependencies (12)**

**Used by (354)**

**Jenkins jobs (12)**

source ubuntu xenial
build passing

binary ubuntu xenial amd64
build passing

devel ubuntu xenial amd64
build passing

doc
build passing

**wiki markup**

```
<<PackageHeader(roscpp)>>
```

Create easy-to-
install packages

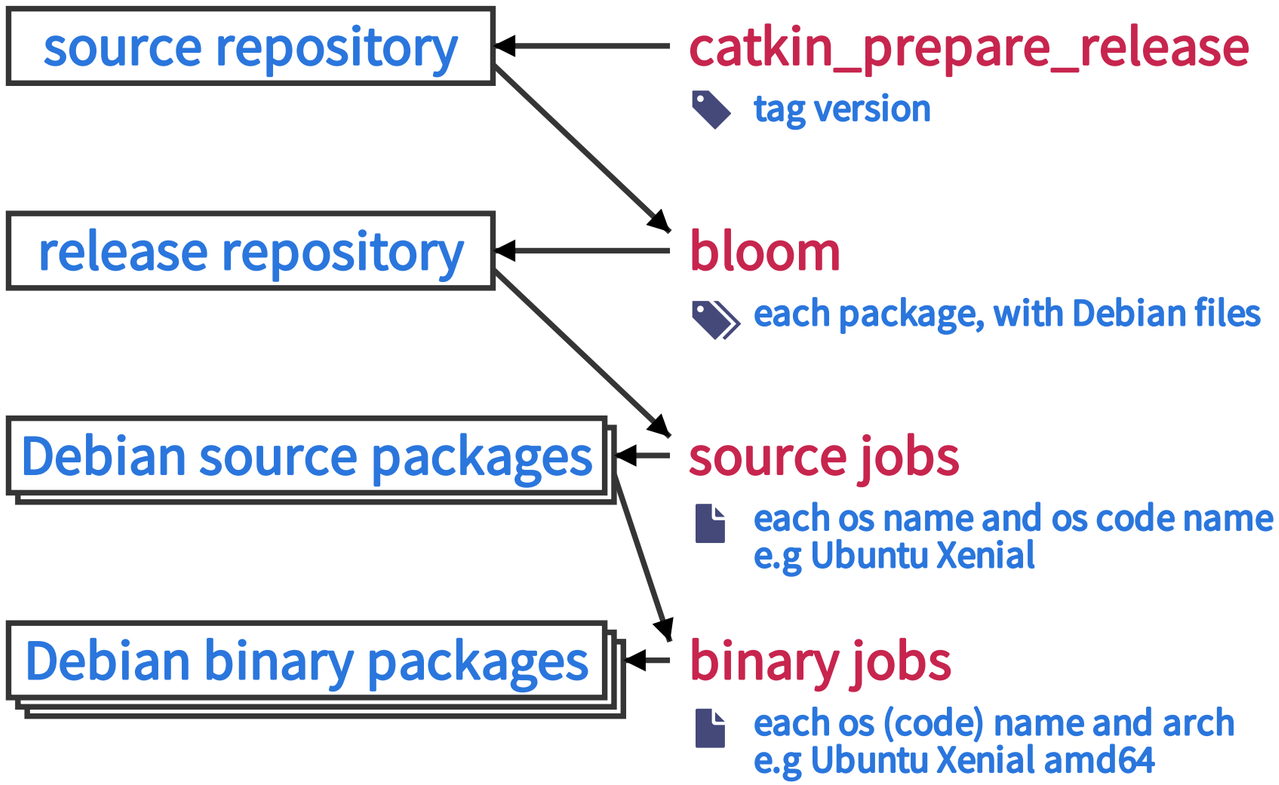


Run tests before /
after code changes

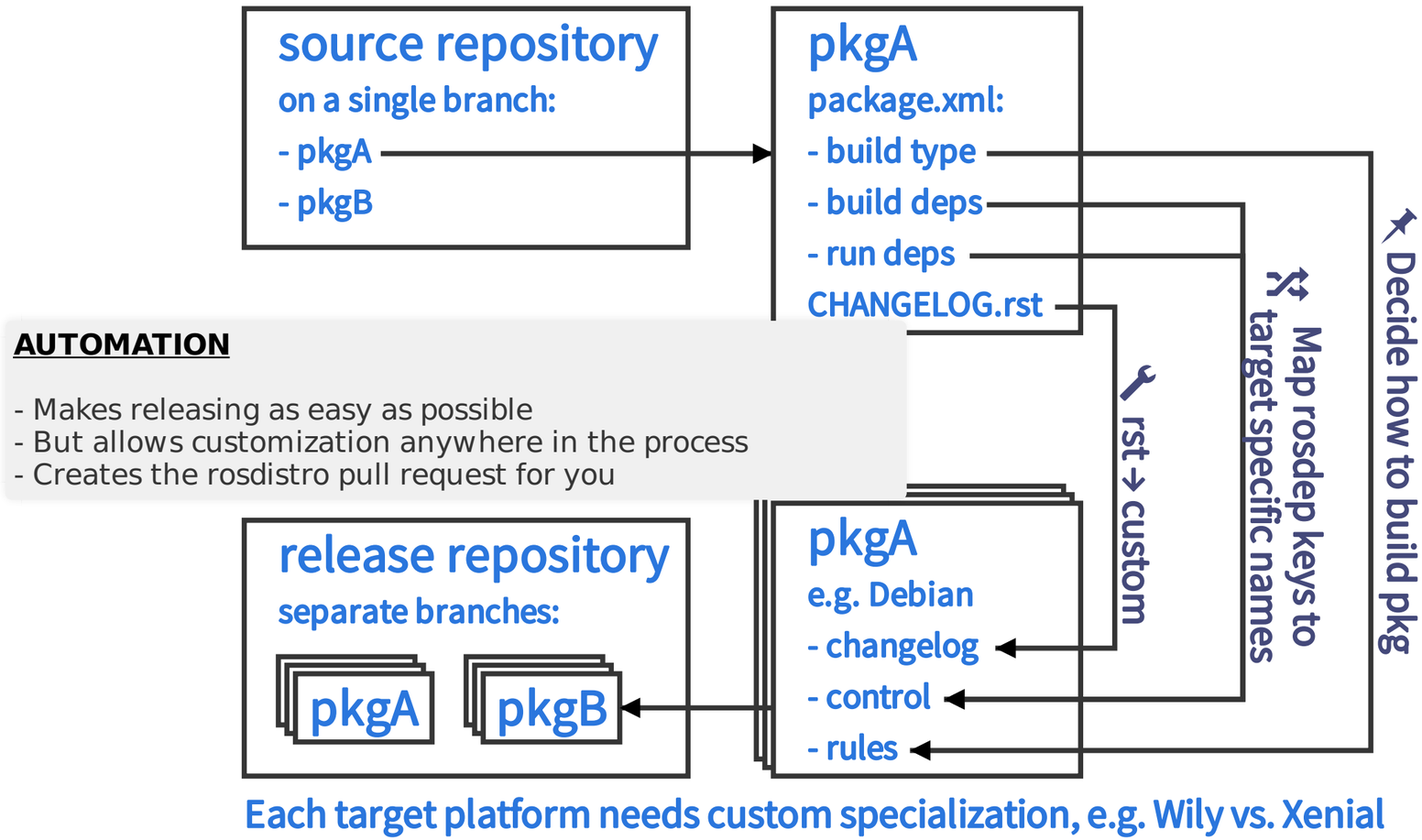


Generate API doc
and meta information

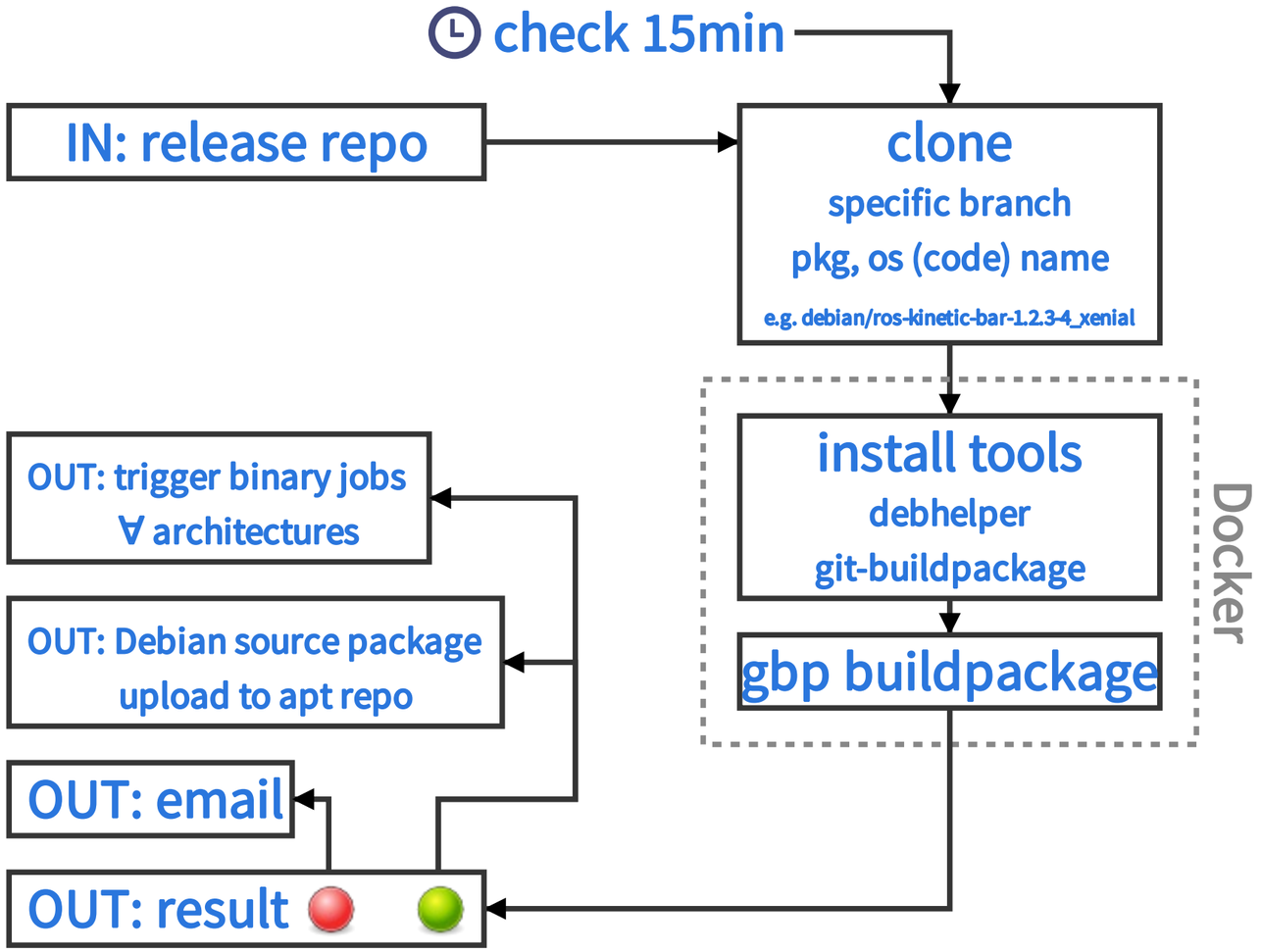
# Release Process



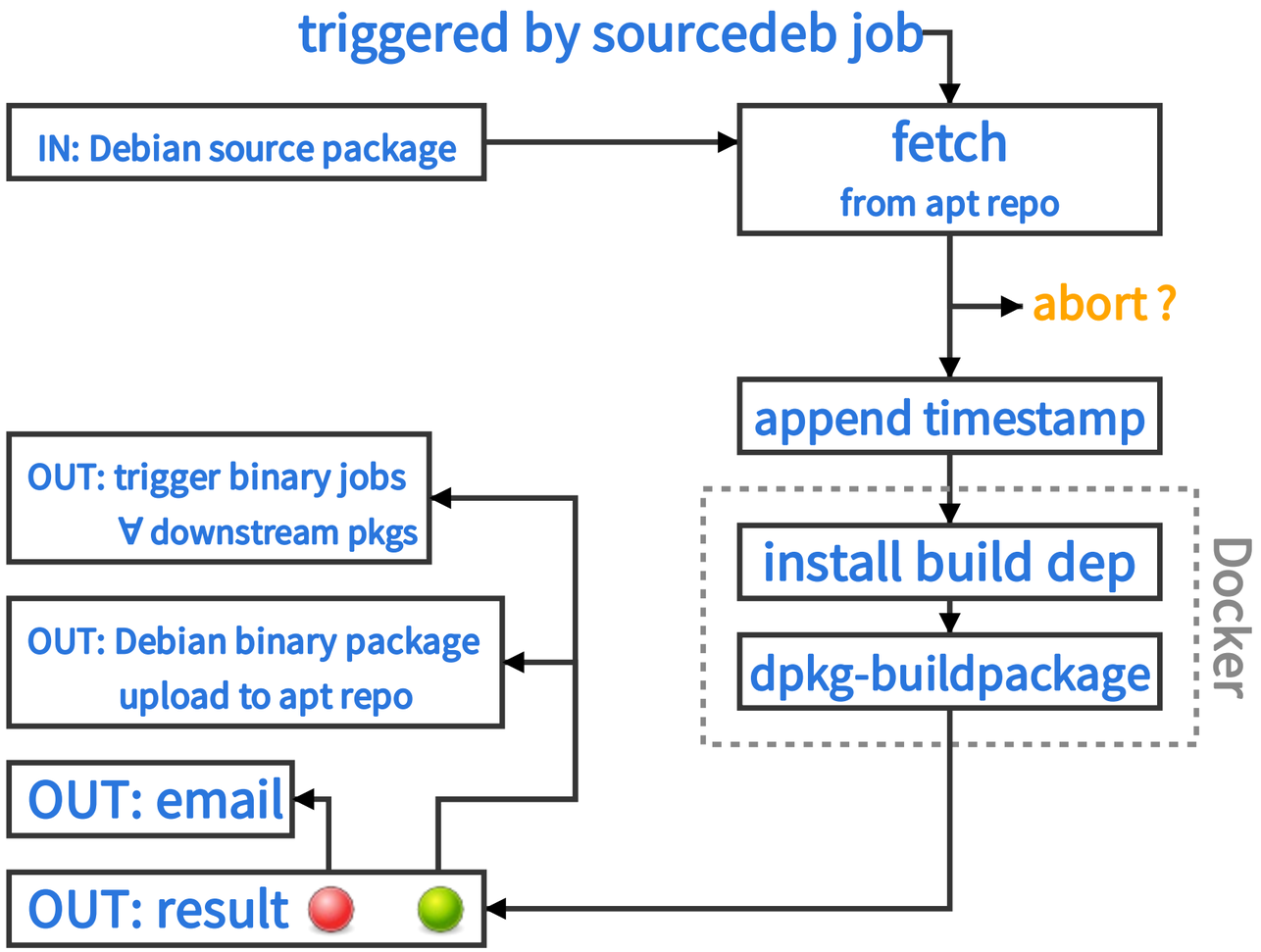
# Bloom


source repository
on a single branch:
- pkgA
- pkgB
pkgA
package.xml:
- build type
- build deps
- run deps
CHANGELOG.rst
AUTOMATION
- Makes releasing as easy as possible
- But allows customization anywhere in the process
- Creates the rosdistro pull request for you
release repository
separate branches:
pkgA
pkgB
pkgA
e.g. Debian
- changelog
- control
- rules
rst → custom
Map rosdep keys to target specific names
Decide how to build pkg
Each target platform needs custom specialization, e.g. Wily vs. Xenial

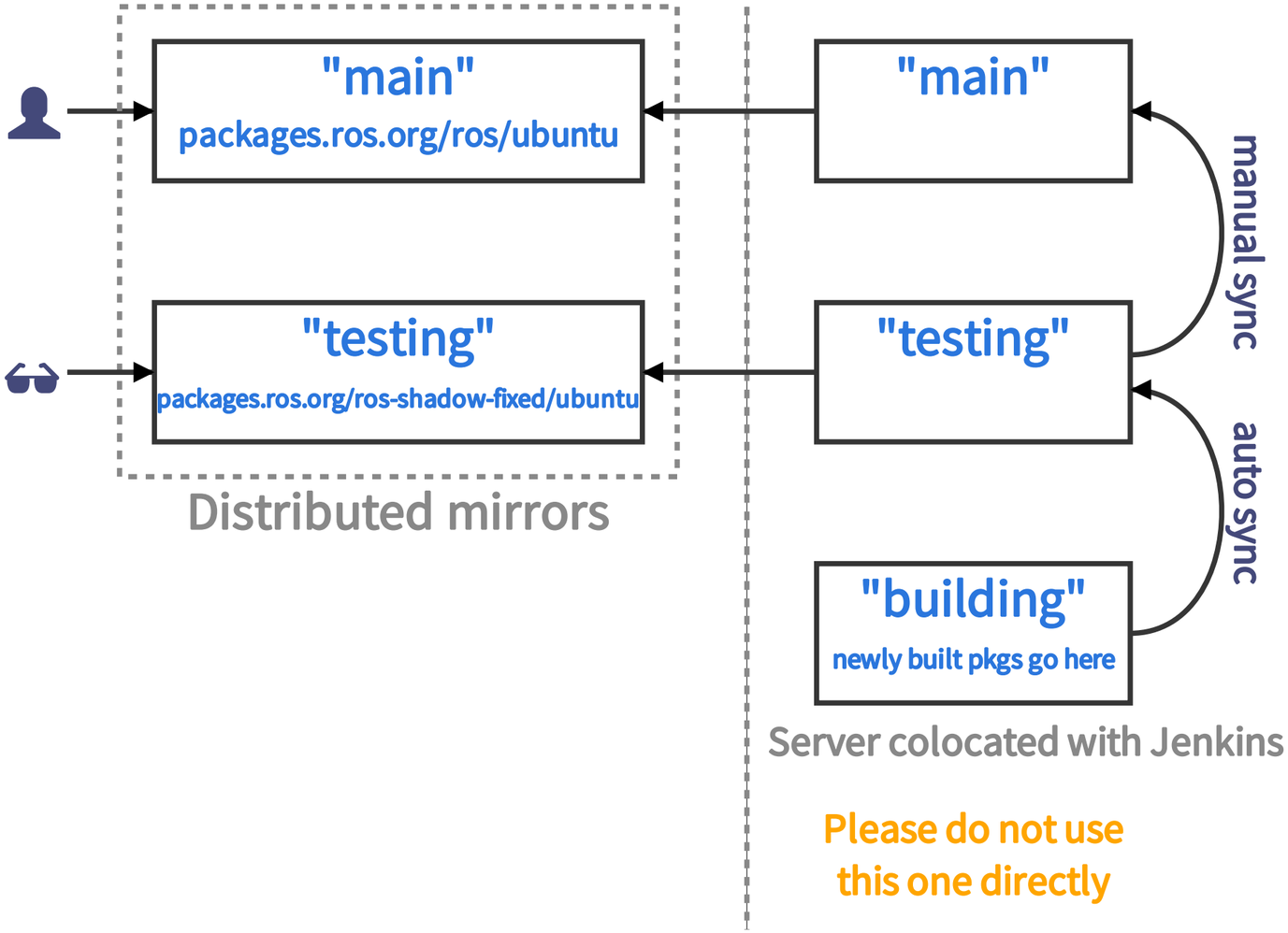
# Sourcedeb Jobs



# Binarydeb Jobs



# Apt Repositories



# Status Pages

ROS.org

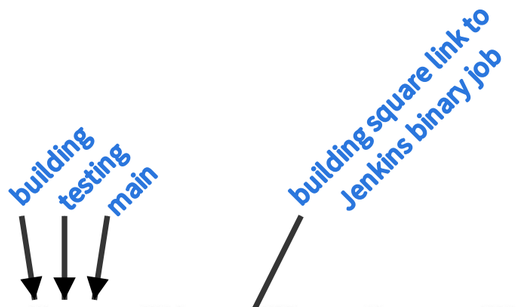
**ROS Kinetic - release status**

Quick filter: *, SYNC, REGRESSION, DIFF, BLUE, RED, ORANGE, YELLOW, GRAY

*showing 941 of 941 total*

the repositories
same version
lower version
higher version
missing
obsolete
intentionally missing

Page was generated:
8 minutes ago




| Name | Repo | Version | Status | Maintainer | Wsource 941 941 850 | W64 903 903 835 | W32 903 903 835 | Xsource 941 941 850 | X64 910 910 836 | X32 910 910 836 |
|---|---|---|---|---|---|---|---|---|---|---|
| ackermann_msgs | ackermann_msgs | 1.0.1-0 | maintained | Jack O'Quin | | | | | | |
| actionlib | actionlib | 1.11.6-0 | maintained | Mikael Arguedas | | | | | | |
| actionlib_lisp | roslisp_common | 0.2.8-0 | developed | Lorenz Moesenlechner<br>Georg Bartels | | | | | | |
| actionlib_msgs | common_msgs | 1.12.4-0 | maintained | Tully Foote | | | | | | |
| actionlib_tutorials | common_tutorials | 0.1.8-0 | maintained | Daniel Stonier | | | | | | |
| agvs_common | agvs_common | 0.1.3-1 | maintained | Roberto Guzmán<br>Román Navarro | | | | | | |
| agvs_control | agvs_sim | 0.1.3-0 | maintained | Román Navarro<br>Roberto Guzmán | | | | | | |
| agvs_description | agvs_common | 0.1.3-1 | maintained | Roberto Guzmán<br>Román Navarro | | | | | | |
| agvs_gazebo | agvs_sim | 0.1.3-0 | maintained | Roberto Guzmán<br>Román Navarro | | | | | | |
| agvs_pad | agvs_common | 0.1.3-1 | maintained | Román Navarro | | | | | | |
| agvs_robot_control | agvs_sim | 0.1.3-0 | maintained | Roberto Guzmán<br>Román Navarro | | | | | | |
| agvs_sim | agvs_sim | 0.1.3-0 | maintained | Roberto Guzmán<br>Román Navarro | | | | | | |
| agvs_sim_bringup | agvs_sim | 0.1.3-0 | maintained | Roberto Guzmán<br>Román Navarro | | | | | | |



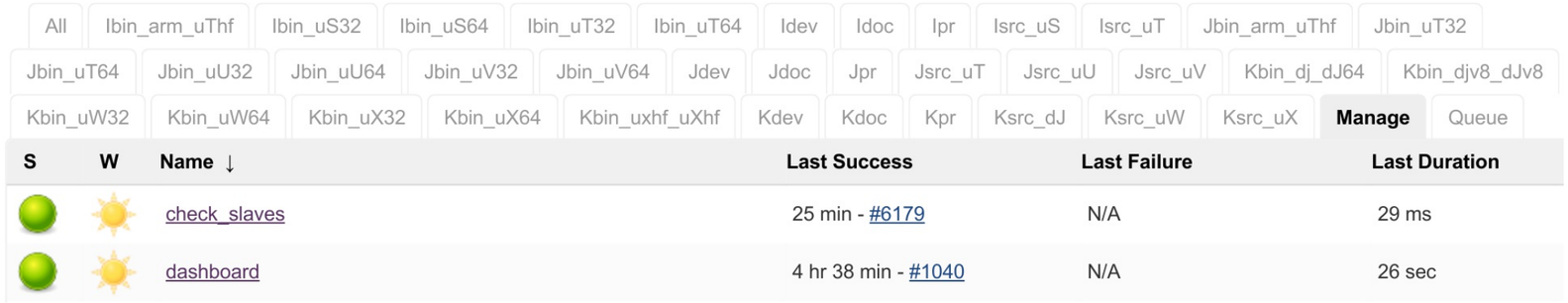
repositories.ros.org/status_page/

# "Decipher" Jenkins Job Names

**Some example job names**

- `lsrc_uS__roscpp__ubuntu_saucy__source`
- `Jdev__ros_comm__ubuntu_trusty_amd64`
- `Kbin_uX64__rospack__ubuntu_xenial_amd64__binary`

- Until first `__`: prefix for group of jobs




- Until next `__`: the package or repository name
- Then *os name*, *os code name*, *arch* (except for source jobs), *source* / *binary* (for release jobs)

# Jenkins Statistics

| | count | success | unstable | failure | aborted | not_built | disabled |
|---|---|---|---|---|---|---|---|
| All | 39785 | 38504 (96.78 %) | 148 (0.37 %) | 361 (0.90 %) | 250 (0.62 %) | 14 (0.03 %) | 508 (1.27 %) |
| Ibin_arm_uThf | 2348 | 1997 (85.05 %) | - | 16 (0.68 %) | 8 (0.34 %) | 0 (0.00 %) | 327 (13.92 %) |
| Ibin_uS32 | 2348 | 2316 (98.63 %) | - | 22 (0.93 %) | 9 (0.38 %) | 1 (0.04 %) | 0 (0.00 %) |
| Ibin_uS64 | 2348 | 2314 (98.55 %) | - | 22 (0.93 %) | 11 (0.46 %) | 1 (0.04 %) | 0 (0.00 %) |
| Ibin_uT32 | 2348 | 2333 (99.36 %) | - | 12 (0.51 %) | 3 (0.12 %) | 0 (0.00 %) | 0 (0.00 %) |
| Ibin_uT64 | 2348 | 2334 (99.40 %) | - | 11 (0.46 %) | 3 (0.12 %) | 0 (0.00 %) | 0 (0.00 %) |
| Idev | 769 | 645 (83.87 %) | 26 (3.38 %) | 98 (12.74 %) | 0 (0.00 %) | 0 (0.00 %) | 0 (0.00 %) |
| Idoc | 807 | 756 (93.68 %) | 31 (3.84 %) | 16 (1.98 %) | 1 (0.12 %) | 3 (0.37 %) | 0 (0.00 %) |
| Ipr | 29 | 20 (68.96 %) | 3 (10.34 %) | 2 (6.89 %) | 0 (0.00 %) | 4 (13.79 %) | 0 (0.00 %) |
| Isrc_uS | 2348 | 2348 (100.00 %) | - | 0 (0.00 %) | 0 (0.00 %) | 0 (0.00 %) | 0 (0.00 %) |
| Isrc_uT | 2348 | 2348 (100.00 %) | - | 0 (0.00 %) | 0 (0.00 %) | 0 (0.00 %) | 0 (0.00 %) |
| Kbin_dj_dJ64 | 941 | 894 (95.00 %) | - | 12 (1.27 %) | 33 (3.50 %) | 0 (0.00 %) | 2 (0.21 %) |
| Kbin_djv8_dJv8 | 941 | 875 (92.98 %) | - | 14 (1.48 %) | 33 (3.50 %) | 0 (0.00 %) | 19 (2.01 %) |
| Kbin_uW32 | 941 | 903 (95.96 %) | - | 12 (1.27 %) | 26 (2.76 %) | 0 (0.00 %) | 0 (0.00 %) |
| Kbin_uW64 | 941 | 903 (95.96 %) | - | 12 (1.27 %) | 26 (2.76 %) | 0 (0.00 %) | 0 (0.00 %) |
| Kbin_uX32 | 941 | 910 (96.70 %) | - | 10 (1.06 %) | 21 (2.23 %) | 0 (0.00 %) | 0 (0.00 %) |
| Kbin_uX64 | 941 | 910 (96.70 %) | - | 10 (1.06 %) | 21 (2.23 %) | 0 (0.00 %) | 0 (0.00 %) |
| Kbin_uxhf_uXhf | 941 | 817 (86.82 %) | - | 11 (1.16 %) | 30 (3.18 %) | 0 (0.00 %) | 83 (8.82 %) |
| Kdev | 291 | 215 (73.88 %) | 47 (16.15 %) | 29 (9.96 %) | 0 (0.00 %) | 0 (0.00 %) | 0 (0.00 %) |
| Kdoc | 304 | 288 (94.73 %) | 11 (3.61 %) | 4 (1.31 %) | 1 (0.32 %) | 0 (0.00 %) | 0 (0.00 %) |
| Kpr | 30 | 23 (76.66 %) | 3 (10.00 %) | 1 (3.33 %) | 0 (0.00 %) | 3 (10.00 %) | 0 (0.00 %) |
| Ksrc_dJ | 941 | 933 (99.14 %) | - | 0 (0.00 %) | 0 (0.00 %) | 0 (0.00 %) | 8 (0.85 %) |
| Ksrc_uW | 941 | 941 (100.00 %) | - | 0 (0.00 %) | 0 (0.00 %) | 0 (0.00 %) | 0 (0.00 %) |
| Ksrc_uX | 941 | 941 (100.00 %) | - | 0 (0.00 %) | 0 (0.00 %) | 0 (0.00 %) | 0 (0.00 %) |
| Manage | 86 | 86 (100.00 %) | 0 (0.00 %) | 0 (0.00 %) | 0 (0.00 %) | 0 (0.00 %) | 0 (0.00 %) |

# Run "Jobs" Locally

The build farm is powered by the Python package ros_buildfarm
After installing it:

- Debian package `python-ros-buildfarm`
- PyPI `ros_buildfarm`

You can run:

- generate_devel_script.py <many args>
- generate_doc_script.py <many args>
- generate_release_script.py <many args>

- to easily reproduce results of the build farm
- run "jobs" for any target platform (within Docker) locally

# Test Across Multiple Repos

① You make related changes in different repos, e.g.

- repo A: add a new API
- repo B: use the new API

  → The devel / PR jobs won't work for this case ☹

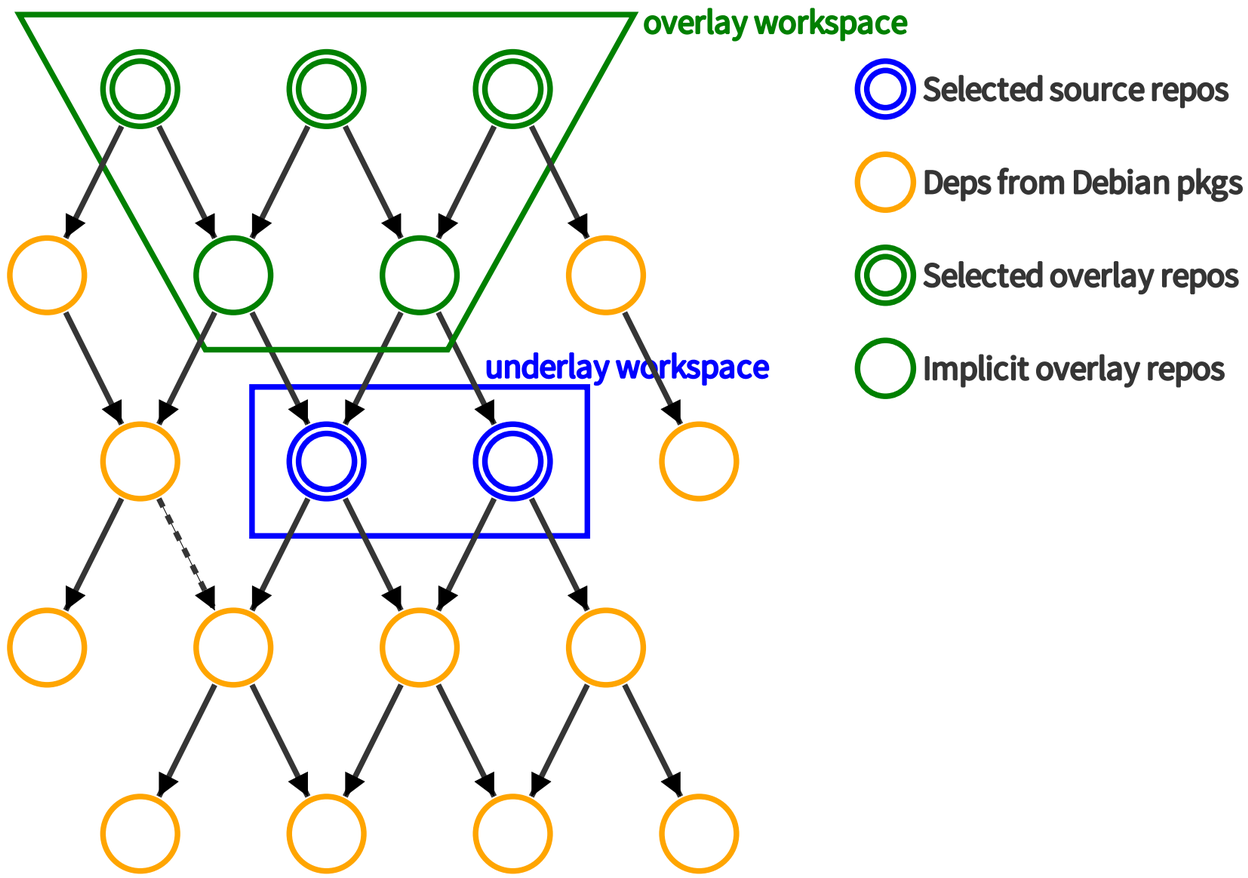
  - repo A: passes but doesn't really check the new API
  - repo B: fails because it's using the **last released version** of A

② You want to check if your changes break any downstream packages

- repo A: make some changes
- repo B: check that the tests for the last released version still pass

  → The devel / PR jobs don't cover for this case either ☹

# Prerelease "Job"




prerelease.ros.org

# Supported Targets

The build farm **can** produce binary packages for

- Ubuntu, Debian
- amd64, i386, arm64, armhf

We can't cover all combinations ( **$**), for **Kinetic** we cover

- Ubuntu: Xenial & Wily amd64 & i386, Xenial arm64 & amrhf
- Debian: Jessie amd64 & arm64

Bloom also generates meta information for rpm (Fedora)

- But we don't build binary packages (yet?)

If you want to create binaries for a not supported target

- Talk to us (<hint> donations can cover the cost for the needed resources 😉)
- Contribute code to automate building binaries for new targets
- or [see next slide]

# Your Own Build Farm

Use cases

- Build a different OS name / OS code name / architecture combination
- Process private repositories
  - You will need your own rosdistro database
- Try modified version of the build farm

Two step process [wiki.ros.org/buildfarm]

1. Provision machines:
   - Jenkins master, Apache webserver, Build slaves
2. Generate the Jenkins jobs
   - Many configuration option

Expect continuous effort

# What Can You Do

## Register your repositories

- add `doc` entry to generate documentation
  - create a wiki page for your packages
- add `source` entry for devel jobs and easy cloning for users
  - if the devel job fails, don't remove the entry, only disable the job:

```
test_commits: false
```

  - e.g. your repo depends on other packages which are not released
- consider enabling `pull request testing`
- `release` your packages, also into newer ROS distros

## Watch out for notification emails from Jenkins

- read it, search for the problem, try to fix it, ask for help
- as a last resort: disable the job, blacklist target, remove the release

# Troubleshooting

"It works locally but the job fails" → something must be different

- Have you tried using `catkin_make_isolated`?
- Most commonly you have a dependency installed but the build farm doesn't. Check the declared dependencies.

```
CMake Error at /opt/ros/kinetic/share/catkin/cmake/catkinConfig.cmake:83 (find_package):
  Could not find a package configuration file provided by "<pkg_name>"
```

"The devel job passed but the release failed"

- In a devel job all packages share the same container. One package might declare a dependency another package uses without declaring it itself.

Expect (very rarely) all kinds of "hicups"

- apt update failing since the repository is being modified concurrently
- network problems, GitHub resetting the connection, an http request not being handled, ...
- If your problem persists it's probably not such a hicup

# Questions...

For more information go to:
github.com/ros-infrastructure/ros_buildfarm