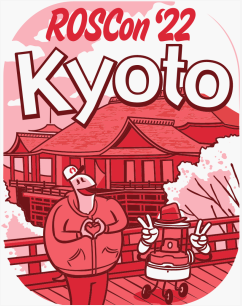
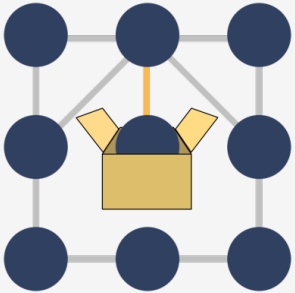
The ROS Build Farm and You!

How the packages that you release become binary packages in the ROS repositories





whoami

Steven! Ragnarök
ROS Infrastructure Team
discourse.ros.org/u/nuclearsandwich
github.com/nuclearsandwich
twitter.com/nuclearsandwich
steven@openrobotics.org

open robotics

Creating ROS 2 Packages

1. `ros2 pkg create …`
2. `catkin_generate_changelog`
3. `catkin_prepare_release`
4. Add a source repository entry in ros/rosdistro
5. Create a ros2-gbp release repository
6. `bloom-release …`
7. WAAAAAAAAAAAAAAAAAIT
8. `apt-get install ros-rolling-yourpackage`

Steps 1, 4, and 5 are only required for initial setup.

What is "the build farm"?

A cluster of machines which communicate with a central Jenkins build server to perform various build and automation tasks for the ROS project and community. There are actually two* of them.

build.ros.org	build.ros2.org
ROS Noetic	ROS 2 Rolling
ROS Melodic	ROS 2 Humble
~~ROS Lunar~~	~~ROS 2 Ga~~lactic
~~ROS Kinetic~~	ROS 2 Foxy

*ci.ros2.org is specialized and not a usual build farm

What is "the build farm"?

There are several different machine roles in the cluster:
Exactly one *Jenkins* host.
Exactly one *Repository (repo)* host.
At least one *Agent* host per architecture.
Optionally one or more *CI Agent* hosts.

What does the ROS build farm do?

Runs per repository CI

Tests pull requests and updates to the source branch.

Builds documentation

API and long form documentation for configured packages.

Builds packages for ROS repositories

Source packages and binary, packages.

Performs distribution management automation

Synchronize packages between repositories and update repository status pages.

...and it looks good doing it!

The entire build farm process is...

- publicly accessible to the entire ROS community
- run using open source software
- using public configuration information *
- running on hosts provisioned using open source infrastrucure as code

---

*excepting secrets like GPG private keys

ROS build farm open source projects

github.com /

ros-infrastructure /	ros_buildfarm
	ros_buildfarm_config
	reprepro-updater
	cookbook-ros-buildfarm
ros2 /	ros_buildfarm_config
ros /	rosdistro

What are the ROS Repositories?

Package repositories hosted on packages.ros.org and the ROS build farms.
APT repositories for Debian and Ubuntu
RPM repositories for Enterprise Linux, and perhaps someday Fedora and/or OpenSUSE
ROS and ROS 2 have separate repositories although there is no RPM repository for ROS 1.

What are the ROS Repositories?

Package repositories hosted on packages.ros.org and the ROS build farms.

	Public		Internal
ROS (1)	ros	ros-testing	ros-building
ROS 2	ros2	ros2-testing	ros2-building

building repositories are not publicly mirrored on packages.ros.org.

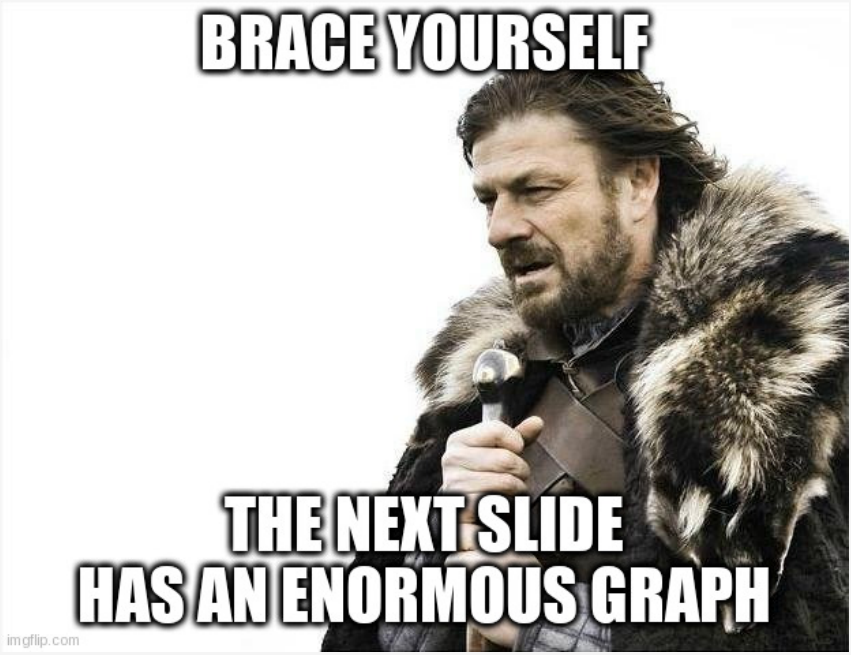
Builld farm repository "flow"

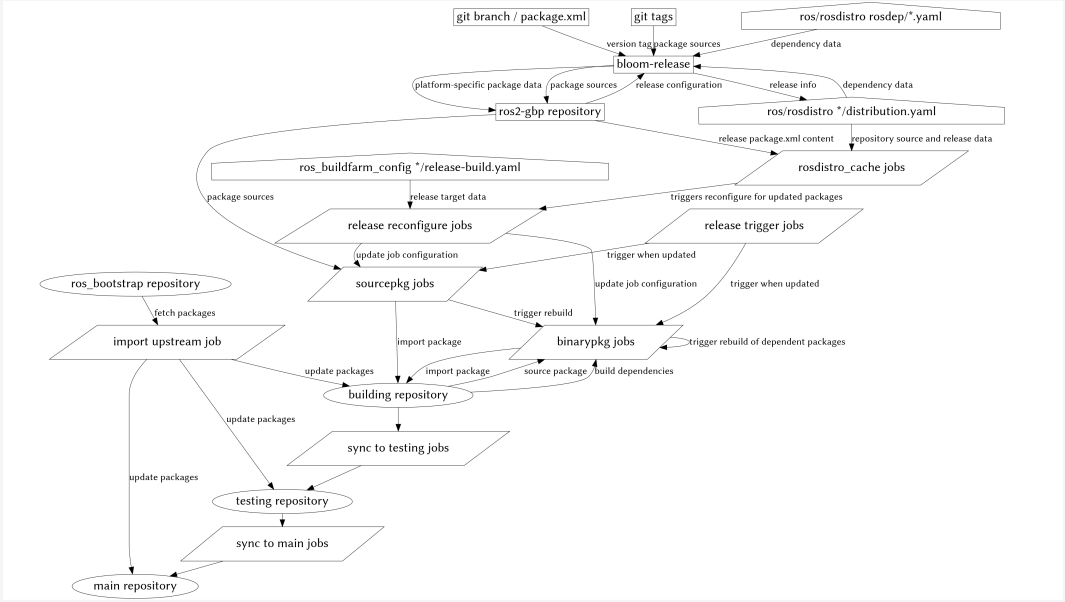
ros_bootstrap repository
: Entry point for infrastructure packages and third party packages

building repository
: Staging ground for packages being rebuilt

testing repository
: Repositories for use in CI and manual testing

main repository
: Default installation path for supported platforms


BRACE YOURSELF
THE NEXT SLIDE
HAS AN ENORMOUS GRAPH
imgflip.com


git branch / package.xml
git tags
ros/rosdistro rosdep/*.yaml
version tag
package sources
dependency data
bloom-release
platform-specific package data
package sources
release configuration
release info
dependency data
ros2-gbp repository
ros/rosdistro */distribution.yaml
release package.xml content
repository source and release data
rosdistro_cache jobs
ros_buildfarm_config */release-build.yaml
release target data
package sources
triggers reconfigure for updated packages
release reconfigure jobs
release trigger jobs
update job configuration
trigger when updated
ros_bootstrap repository
sourcepkg jobs
update job configuration
trigger when updated
fetch packages
trigger rebuild
import upstream job
import package
binarypkg jobs
trigger rebuild of dependent packages
update packages
import package
source package
build dependencies
building repository
update packages
sync to testing jobs
update packages
testing repository
sync to main jobs
main repository

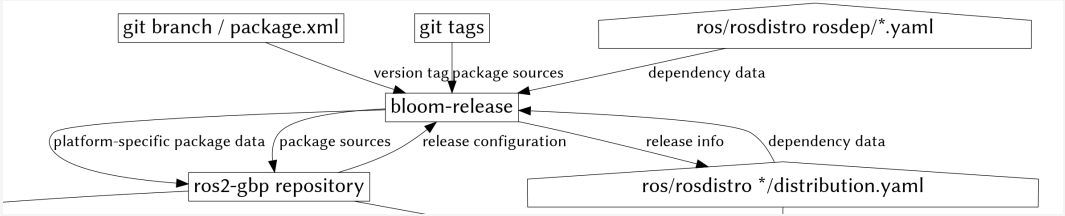
Don’t Panic

We’re going to get through this together

One step at a time...

1. Prepare the source release
2. Let the build farm do its thing
3. Wait for a sync
4. Install your package

1. Preparing the source release



1. Preparing the source release

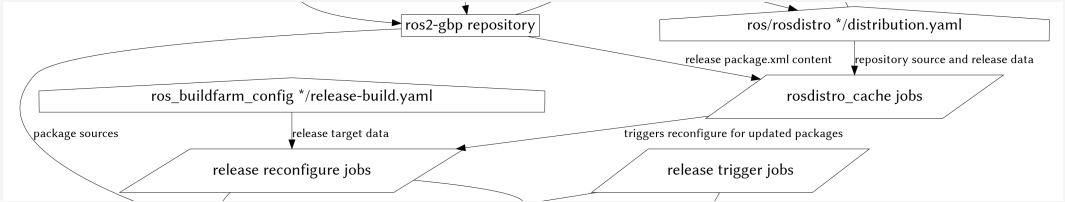
There are many infrastructure tools to help with this:

1. `catkin_generate_changelog` to help create and update changelogs.
2. `catkin_prepare_release` to update version info, tag, and push the release
3. `bloom-release` to update the release repository data for new releases.

1. Preparing the source release

ros/rosdistro release PRs opened automatically by Bloom.

2. The build farm's thing



2. The build farm's thing

Jenkins release management jobs

1. rosdistro_cache jobs run every five minutes polling for changes in ros/rosdistro
2. reconfigure jobs for changed distributions are triggered by rosdistro cache
3. every 15 minutes trigger jobs start packaging jobs for changed packages

rosdistro cache

gzip compressed yaml document containing

`distribution_file` a copy of the distribution file from ros/rosdistro

`release_repo_package_xmls` package.xml contents from the currently released package version

`source_repo_package_xmls` empty (fetching source package.xml is tricky)

`name` the short name of this rosdistro

`type`, `version` REP-141 format information

rosdistro-cache jobs

Examples `rolling_rosdistro-cache`, `humble_rosdistro-cache`

- Generates a new rosdistro cache file and uploads it to the repository host.
- Runs on five minute intervals.
- Polls for distribution.yaml changes via HTTPS. Changes sometimes lag due to CDN caching.
- Triggers reconfigure jobs for updated packages.

release reconfigure jobs

Examples `Rrel_reconfigure-jobs`, `Rrel_rhel_reconfigure-jobs`, `Rrel_ujv8_reconfigure-jobs`

- One job per release build file in ros_buildfarm_config
- Uses release build config and ros_buildfarm templates to generate a sourcepkg and binarypkg job for each package
- Runs when triggered by rosdistro cache to update changed packages
- Runs every 24 hours to keep job configurations up to date

release trigger jobs

Examples `Rrel_trigger-jobs`, `Rrel_rhel_trigger-jobs`, `Rrel_ujv8_trigger-jobs`

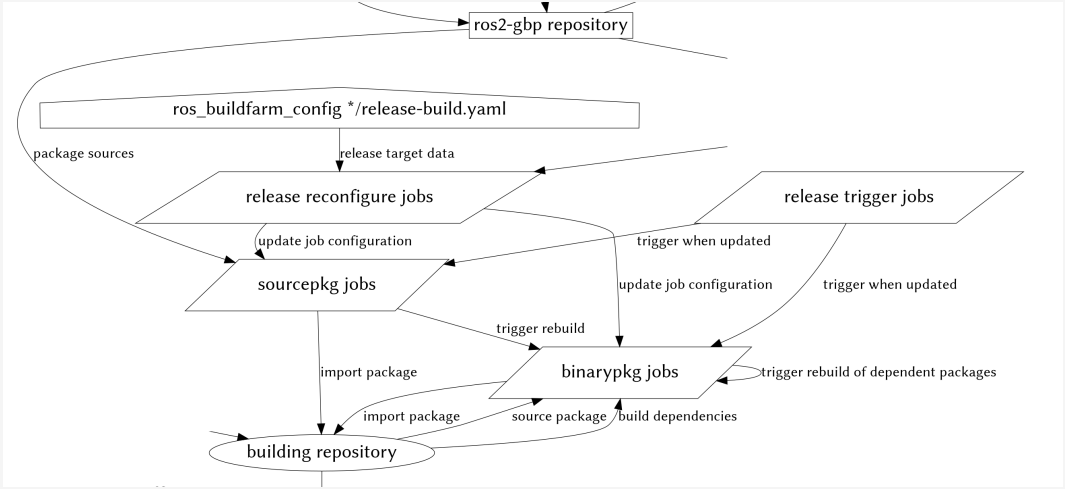
- One job per release build file in ros_buildfarm_config
- Compares packages in the building repository with expected packages based on rosdistro cache and triggers sourcepkg and/or binarypkg jobs to implement changes

Job parameters:

missing only – triggers jobs for packages not present in repositories.

source only – triggers jobs for source packages only; useful when source packages need to be updated before binaries are rebuilt

Packaging jobs



source package jobs

Examples `Rsrc_uJ64__rclcpp__ubuntu_jammy_source`

- Fetch package sources with platform-specific metadata from ros2-gbp repository and create platform source package (*.dsc* for Debian/Ubuntu, (*.srpm* for Fedora/RHEL) and upload it to the building repository.
- Does not use ROS tools like colcon, rosdep, bloom. By this time all package information has been rendered into distribution-specific formats.
- A successful build will trigger binary package builds for all binary packages generated from the created source package.
- Old binary packages are removed when a new source package is imported.

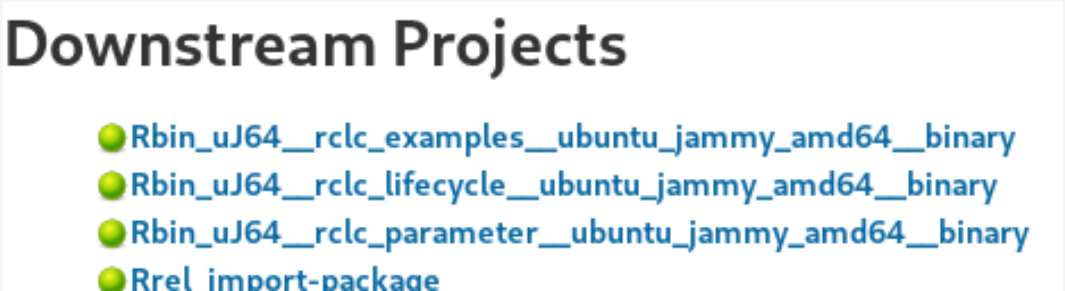
binary package jobs

Examples `Rbin_uJ64__rclcpp__ubuntu_jammy_amd64__binary`

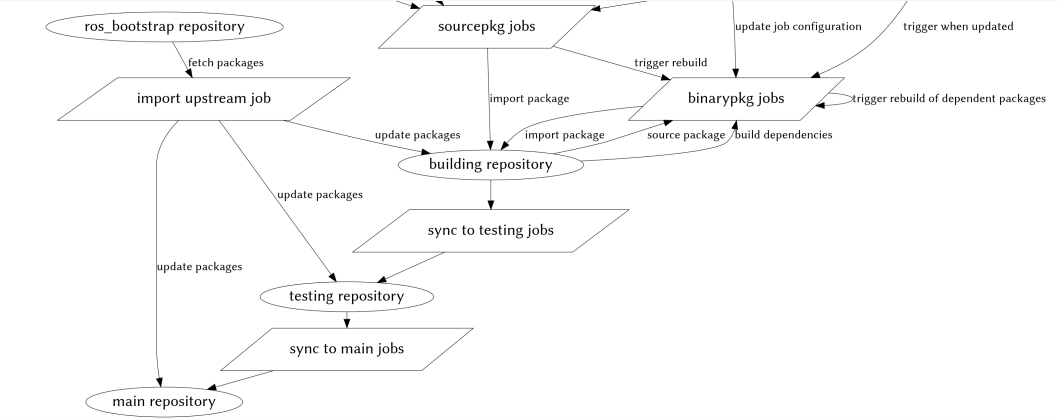
- Fetch the source from the building repository.
- Installs build dependencies.
- Uploads the binary package created to the building repository.
- Does not use ROS tools like colcon, rosdep, bloom. By this time all package information has been rendered into distribution-specific formats.
- A successful build will complete by importing the newly built binary package into the building repository and *invalidating* all package's downstream of it.

Package invalidation

- Remove all packages which depend recursively on the newly rebuilt package.
- Prevents ABI breaks in production.
- Instant feedback for downstream packages when APIs change.



3. Waiting for a sync



sync to testing jobs

Examples `Rrel_sync-packages-to-testing_jammy_amd64`, `Rrel_sync-packages-to-testing_rhel_8_x86_64`

- Removes *all* ROS packages from the testing repository which match the job's rosdistro, platform version, and architecture, as well as associated source packages.
- Imports all ROS packages from the building repository which match the same criteria.

sync to testing jobs

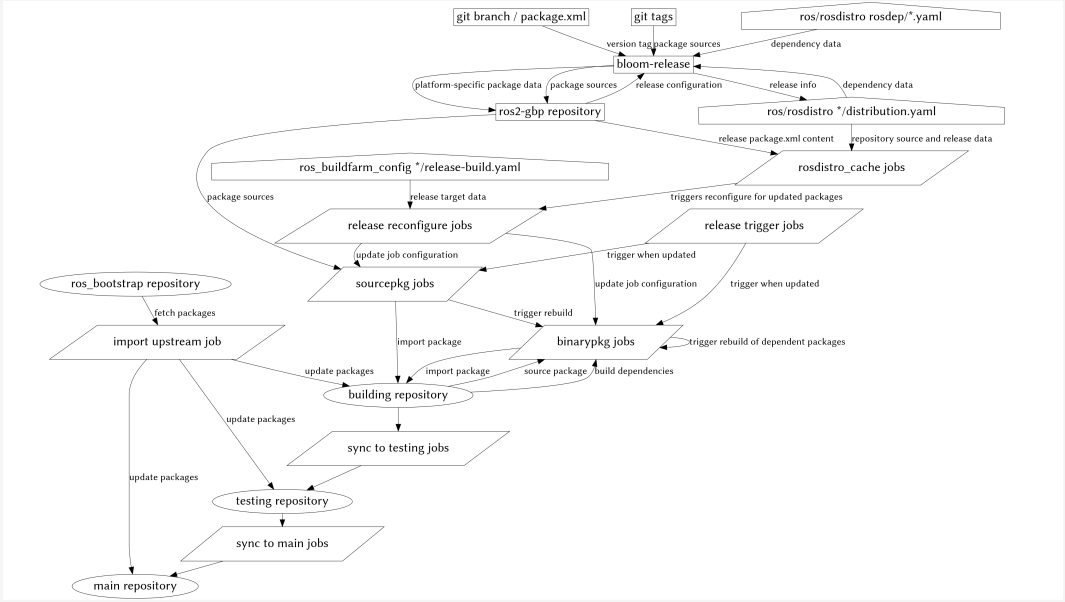
- This way, packages are deleted when there is no longer a binary job for them.
- All binary and source jobs block the testing jobs so these run only once during large rebuilds.
- Automated thresholds are set to prevent a bad release from purging the testing repository.

sync to main jobs

Examples `Frel_sync-packages-main`, `Hrel_sync-packages-to-main`

- Removes *all* ROS packages from the main repository which match the job's rosdistro for each platform version and architecture combination.
- Only ever run manually by the ROS core team.

There is a separate subordinate Jenkins job for handling RPM repositories


git branch / package.xml
git tags
ros/rosdistro rosdep/*.yaml
version tag
package sources
dependency data
bloom-release
platform-specific package data
package sources
release configuration
release info
dependency data
ros2-gbp repository
ros/rosdistro */distribution.yaml
release package.xml content
repository source and release data
rosdistro_cache jobs
ros_buildfarm_config */release-build.yaml
release target data
package sources
triggers reconfigure for updated packages
release reconfigure jobs
release trigger jobs
update job configuration
trigger when updated
ros_bootstrap repository
sourcepkg jobs
update job configuration
trigger when updated
fetch packages
trigger rebuild
import upstream job
import package
binarypkg jobs
trigger rebuild of dependent packages
update packages
import package
source package
build dependencies
building repository
update packages
sync to testing jobs
update packages
testing repository
sync to main jobs
main repository

Thanks!

Modes of integration

Distribution model integration

Software collections are curated and integrated through community effort. The software available is limited but using all of it together is much more likely to be mutually compatible.

Examples: Most Linux distributions (Ubuntu 22.04, Fedora 36, etc), ROS distributions (Noetic, Humble), Gazebo collections (Citadel, Fortress, Garden), Homebrew.

Modes of integration

Application model integration

Software is aggregated in service of a specific application from multiple available sources. Applications may choose to integrate arbitrary desired versions of software based on need but each application team must curate their own (often much smaller) set of dependencies.

Examples: Conda-forge, npm, PyPI, Crates.io/Cargo

Binary packages

Why do we need them?

Alternatives:

Everyone builds from source

’sup Gentoo

Docker / OCI

Currently these use the binary packages under the hood.

Conda / Spack

These are great on platforms without a distribution packaging system but...