Accelerating Robotics Development with Embedded Linux

Rob Woolley, Wind River

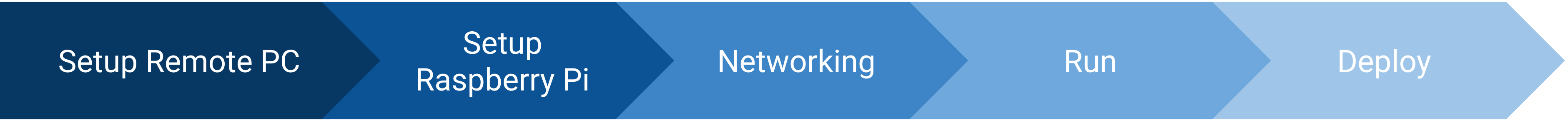
Who am I?

- 25 years experience with Linux and embedded
- Started using ROS in 2018
- Actively participating in a number of communities including OpenEmbedded (OE) and the Yocto Project (YP)
- Maintainer of the meta-ros since Nov 2022
  - A series of OpenEmbedded layers designed to add support for the Robot Operating System (ROS) for embedded Linux releases by the Yocto Project

Challenges for Robotics Development

- Setting up the development environment
- Getting support for a variety of embedded hardware
- Synchronizing custom changes with team mates and community
- Releasing product-quality software and supporting it over the long term
- Finding ready to use packages and filesystem images
- Enabling new developers to get started quickly and easily

Case Study: TurtleBot3 Quick Start Guide



Setup Remote PC

1. Install supported Ubuntu (or VM)
2. Install ROS packages
3. Add TurtleBot3 (TB3) packages

Setup Raspberry Pi

1. Write Raspbian to SD card
2. Attach keyboard and HDMI
3. Setup device configuration (hostname, WiFi SSID, etc.)
4. Download source code for TB3 applications
5. Build TB3 packages on RPi

Networking

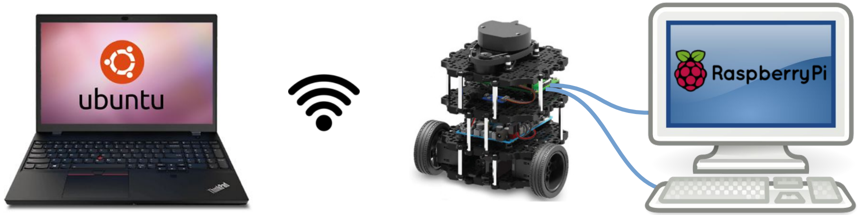
Configure remote PC and TB3 to communicate with each other

Run

Launch roscore and TB3 ROS applications on remote PC

Deploy

Launch TB3 ROS nodes on RPi. TB3 is operational.



https://emanual.robotis.com/docs/en/platform/turtlebot3/overview/

What has changed?

- New graduates do not typically have experience with embedded software
- Legacy tools are typically difficult to use, unfamiliar, and often insecure
- Building the Linux and ROS software stack is massive
  - compute time, filesystem size, and number of independent projects
- ROS is maturing and being used more and more in production systems
- New requirements for edge computing (eg. physical security, remote updates)
- COTS hardware with custom peripherals (sensors, actuators, accelerators)
- Convergence of software domains (eg. AI/ML, Analytics, IT, OT, Security, UX)

Ideal Quick Start for Beginners

Setup

Clone the git repository for a getting started tutorial. The workspace includes a DevContainer and Extension

Code

Developer makes some custom changes to the sample application.

Build

The DevContainer includes an SDK with the development tools and ROS environment. Developer runs the build to compile the app.

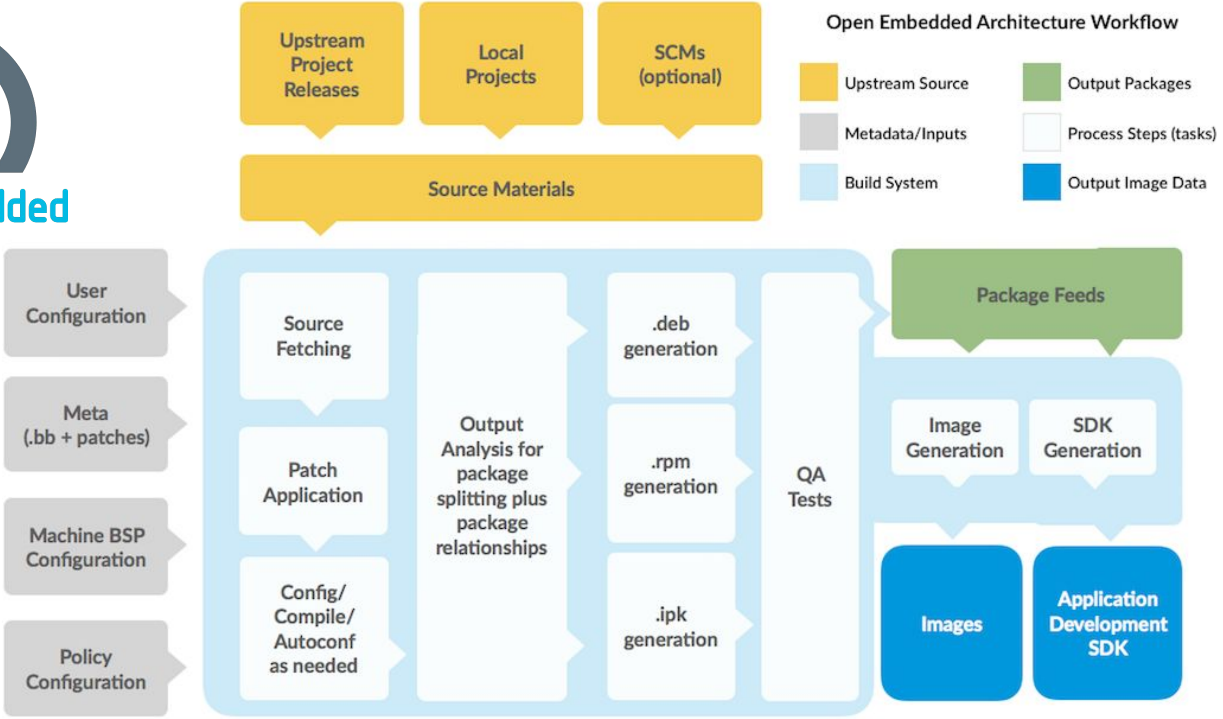
Deploy

A launch configuration is provided to allow the developer to test their application in a QEMU simulated environment with GDB

Why use ROS with OpenEmbedded?

- Support for robotics hardware developer kits
- Source-based build system (bitbake)
- Designed for preserving custom changes and configuration (“build to order”)
- Support for cross-platform builds (supporting building at cloud-scale)
- Maintenance
  - Software Bill of Material (SBOM)
  - Reproducible builds
  - Long-term Support (LTS) releases
  - Commercial support (from semiconductor vendors and others)

Open Embedded Architecture Workflow



Open Embedded Architecture Workflow

🟨	Upstream Source	🟩	Output Packages
▨	Metadata/Inputs	□	Process Steps (tasks)
▨	Build System	🟦	Output Image Data

Upstream Project Releases

Local Projects

SCMs (optional)

Source Materials

User Configuration

Meta (.bb + patches)

Machine BSP Configuration

Policy Configuration

Source Fetching

Patch Application

Config/ Compile/ Autoconf as needed

Output Analysis for package splitting plus package relationships

.deb generation

.rpm generation

.ipk generation

QA Tests

Package Feeds

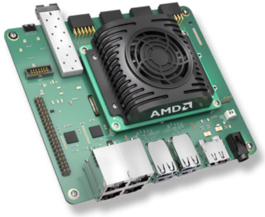
Image Generation

SDK Generation

Images

Application Development SDK

Supported Hardware (meta-ros)



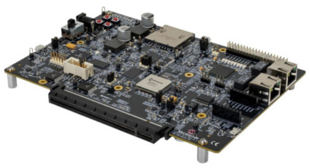
AMD Kria KR260

Nvidia Jetson
(AGX Orin, AGX Xavier,
Orin Nano)

Qualcomm RB3 Gen 2

Raspberry Pi 4 / 5

AAEON UP Squared
Developer Board with
Intel Processor



Microchip PolarFire
SoC FPGA

Development Scenarios

1. Standalone Developer
2. Cloud-based Builds
3. Self-hosted Build Farm
4. Application Developer

Standalone Developer

Yocto Quick Start (https://docs.yoctoproject.org/brief-yoctoprojectqs/)

- 90 GB disk space, 8GB RAM
- Supported OS (Ubuntu, Fedora, CentOS, Debian, OpenSUSE)
- https://docs.yoctoproject.org/ref-manual/system-requirements.html#supported-linux-distributions

Clone the reference distro with git

```
git clone git://git.yoctoproject.org/poky
```

Initialize the Build Environment

```
cd poky; source oe-init-build-env
```

Build basic console-only image

```
bitbake core-image-base
```

Boot the image in QEMU

```
runqemu qemux86-64
```

Using the kas tool

Install kas

```
python3 -m venv venv && source venv/bin/activate && pip3 install kas
```

Clone meta-ros

```
git clone -b build https://github.com/ros/meta-ros
```

Choose a configuration

e.g. oeros-<YOCTO RELEASE>-<ROS RELEASE>-<BOARD>.yml

Checkout

```
KAS_WORK_DIR=<PROJECT_DIR>
kas checkout meta-ros/kas/oeros-kirkstone-humble-raspberrypi4-64.yml
```

Deploying the filesystem

Retrieve the filesystem image

```
build/tmp-glibc/deploy/images/raspberrypi4-64/ros-image-core-humble-raspberrypi4-64.rootfs.wic.bz2
```

Writing the image

If using the Balena Etcher tool to write your SD Card, you may provide it with this file directly.

If using dd or bmaptool, you must first decompress the bzip2 file first.

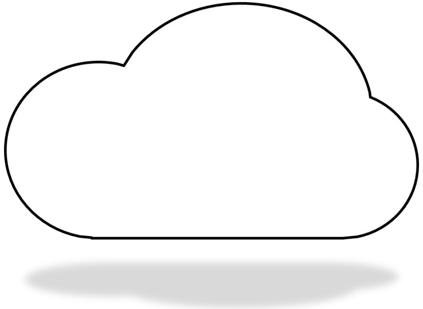
```
oe-run-native bmaptool-native bmaptool copy <build/tmp/deploy/images/machine/image.wic> </dev/sdX>
```

Cloud-based Builds

4 key reasons to use the cloud:

1. Building Linux and ROS is resource intensive
2. Open source moves fast, use continuous iteration to stay up-to-date
3. Share artifacts to save time and lower the bar for all developers
4. Long-term support and maintenance

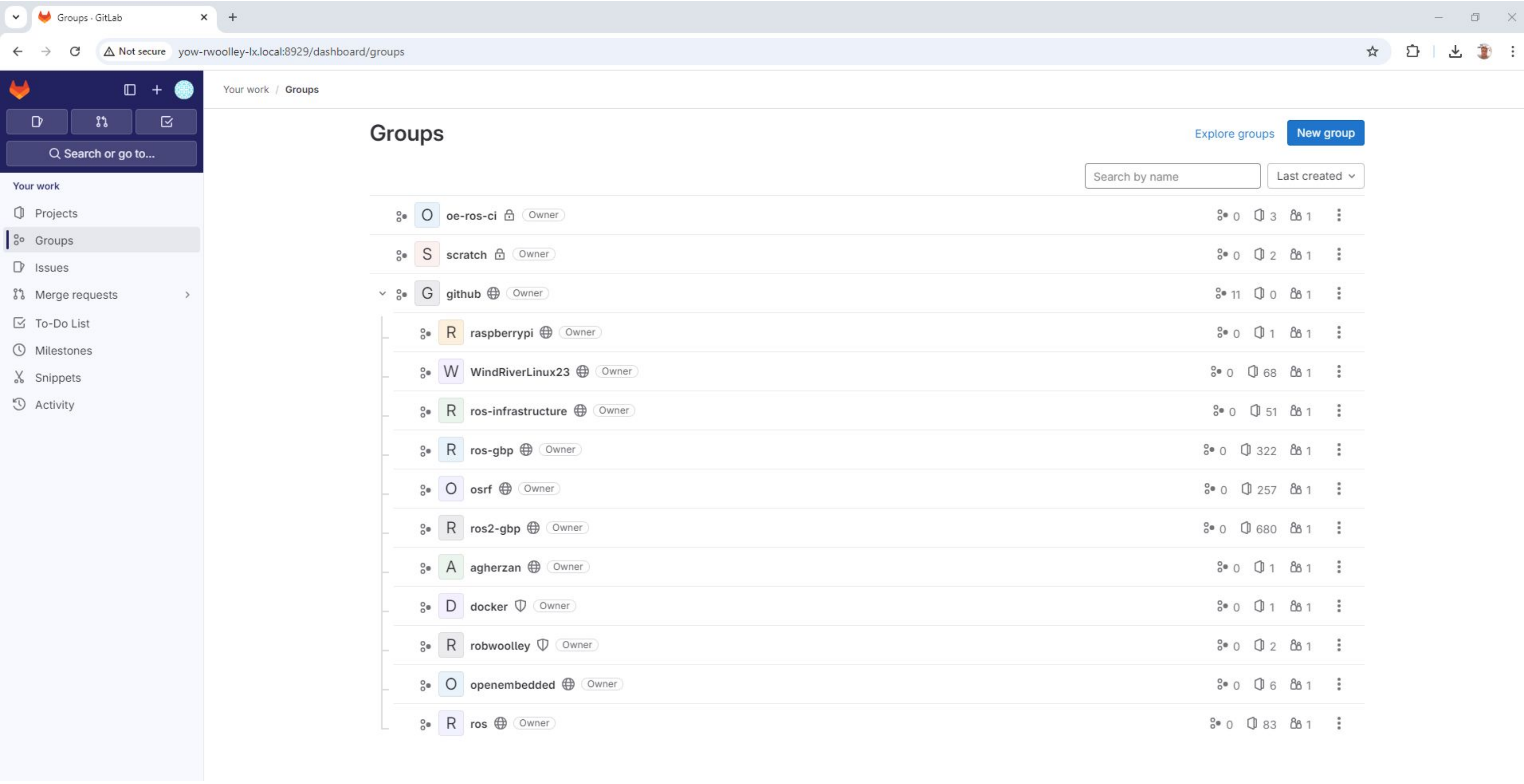
Cloud-based Builds



- Leverage GitLab CI with runners for compute resources
- Use CROPS containers for supported host environment
- Use kas to set up the build environment
- Publish logs and artifacts back to GitLab or AWS S3 bucket
- Registration of runners is currently manual; experimenting with dynamically creating instances with runners on demand
- Caching of shared state cache and fetched git repositories in cloud storage reduces build times
- Well suited for nightly builds networking is fast and free, store only the artifacts you need and shut down compute instances

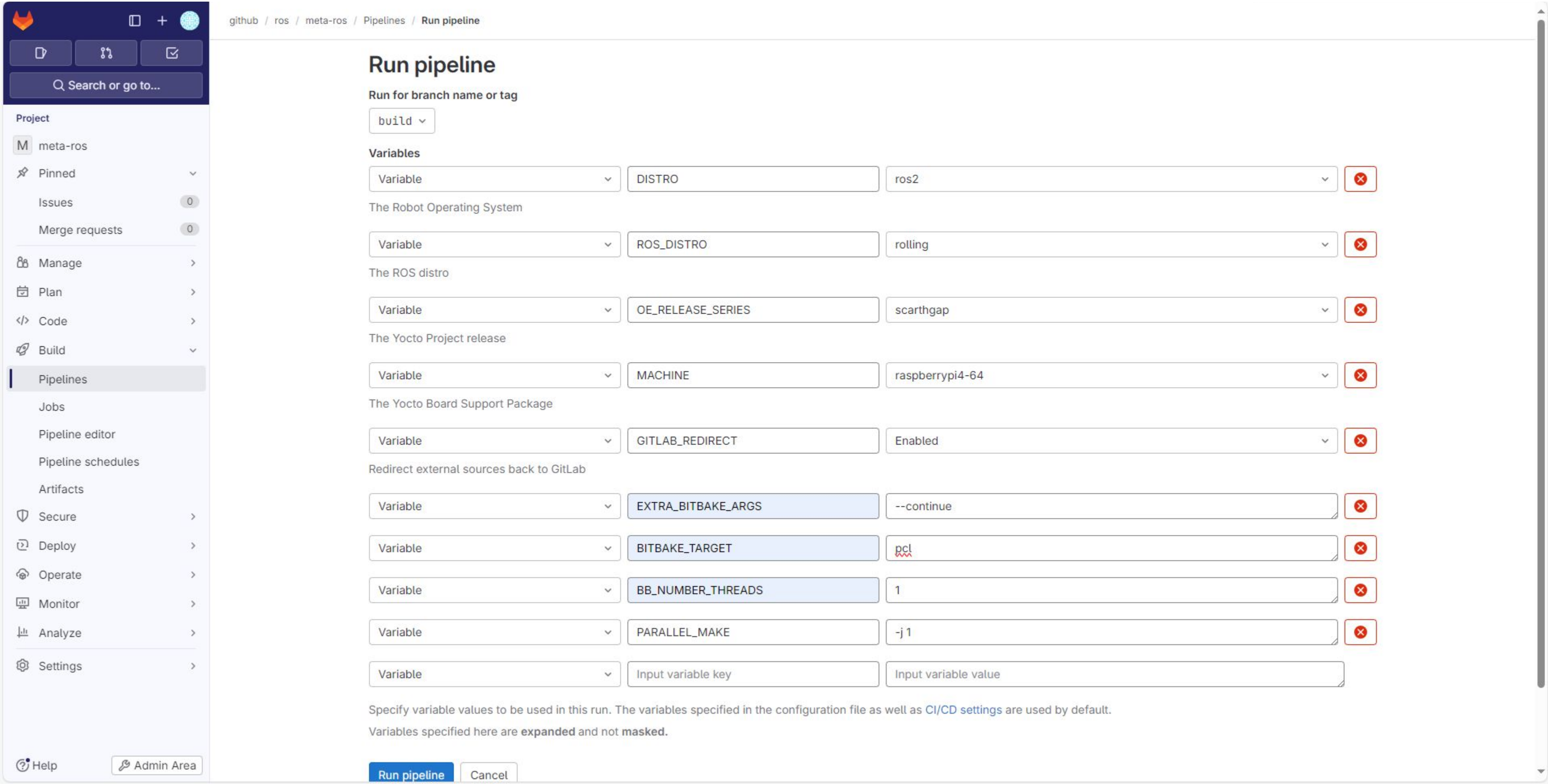
Self-hosted Build Farm

- Run the GitLab Container locally on your own hardware
- Benefits include up-front costs, privacy, and security
- Can use the same GitLab CI setup as in the cloud
- Using static runners, caching can now happen locally on the runner itself
- Mirror git repositories locally to reduce Internet bandwidth (or support air-gapped environment)



Groups

O	oe-ros-ci 🔒	Owner	0	3	1
S	scratch 🔒	Owner	0	2	1
G	github 🌐	Owner	11	0	1
R	raspberrypi 🌐	Owner	0	1	1
W	WindRiverLinux23 🌐	Owner	0	68	1
R	ros-infrastructure 🌐	Owner	0	51	1
R	ros-gbp 🌐	Owner	0	322	1
O	osrf 🌐	Owner	0	257	1
R	ros2-gbp 🌐	Owner	0	680	1
A	agherzan 🌐	Owner	0	1	1
D	docker 🛡	Owner	0	1	1
R	robwoolley 🛡	Owner	0	2	1
O	openembedded 🌐	Owner	0	6	1
R	ros 🌐	Owner	0	83	1



Run pipeline

Run for branch name or tag

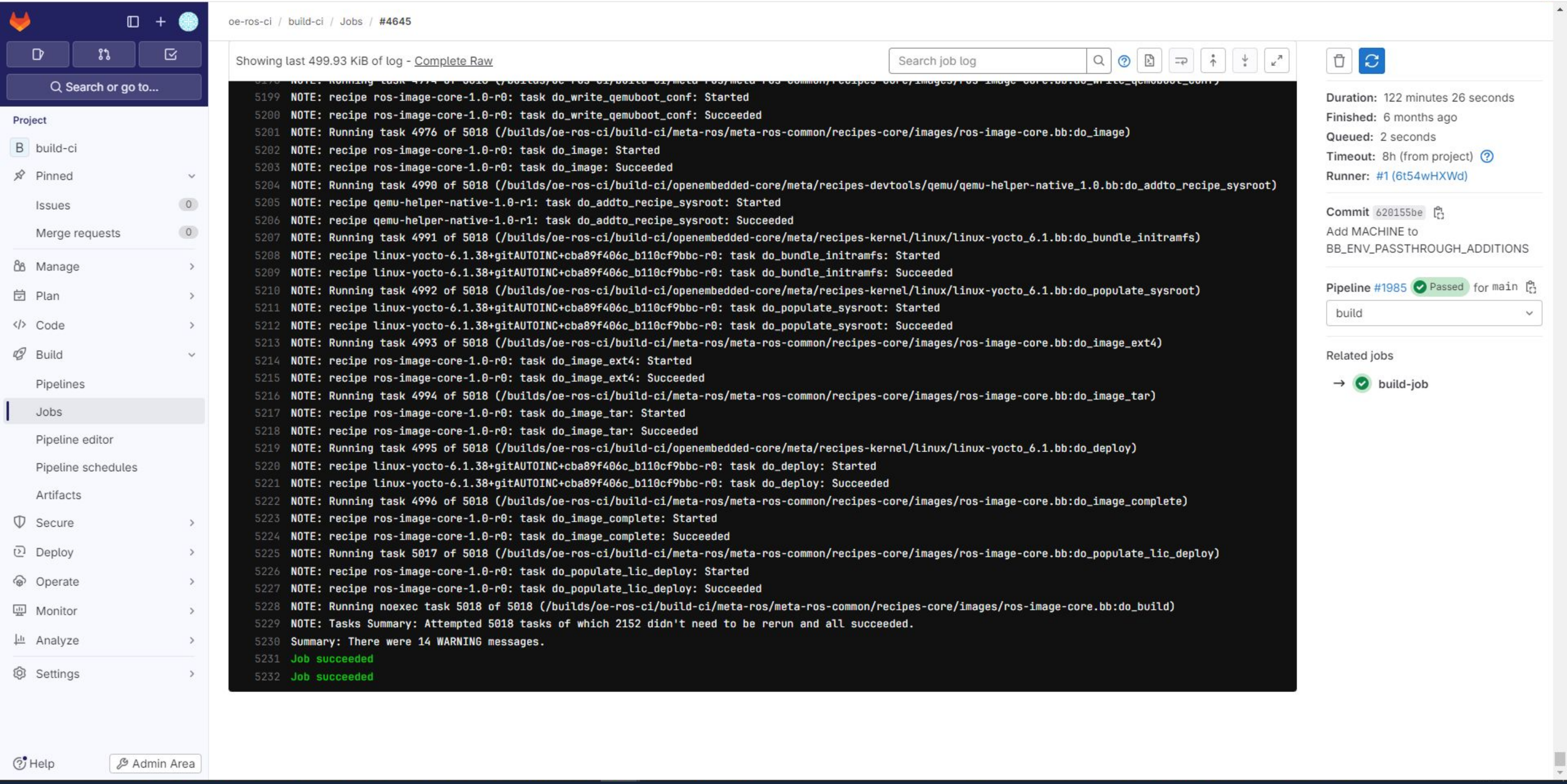
build

Variables

Variable	DISTRO	ros2
The Robot Operating System		
Variable	ROS_DISTRO	rolling
The ROS distro		
Variable	OE_RELEASE_SERIES	scarthgap
The Yocto Project release		
Variable	MACHINE	raspberrypi4-64
The Yocto Board Support Package		
Variable	GITLAB_REDIRECT	Enabled
Redirect external sources back to GitLab		
Variable	EXTRA_BITBAKE_ARGS	--continue
Variable	BITBAKE_TARGET	pcl
Variable	BB_NUMBER_THREADS	1
Variable	PARALLEL_MAKE	-j 1
Variable	Input variable key	Input variable value

Specify variable values to be used in this run. The variables specified in the configuration file as well as CI/CD settings are used by default.

Variables specified here are **expanded** and not **masked.**



```
NOTE: recipe ros-image-core-1.0-r0: task do_write_qemuboot_conf: Started
NOTE: recipe ros-image-core-1.0-r0: task do_write_qemuboot_conf: Succeeded
NOTE: Running task 4976 of 5018 (/builds/oe-ros-ci/build-ci/meta-ros/meta-ros-common/recipes-core/images/ros-image-core.bb:do_image)
NOTE: recipe ros-image-core-1.0-r0: task do_image: Started
NOTE: recipe ros-image-core-1.0-r0: task do_image: Succeeded
NOTE: Running task 4990 of 5018 (/builds/oe-ros-ci/build-ci/openembedded-core/meta/recipes-devtools/qemu/qemu-helper-native_1.0.bb:do_addto_recipe_sysroot)
NOTE: recipe qemu-helper-native-1.0-r1: task do_addto_recipe_sysroot: Started
NOTE: recipe qemu-helper-native-1.0-r1: task do_addto_recipe_sysroot: Succeeded
NOTE: Running task 4991 of 5018 (/builds/oe-ros-ci/build-ci/openembedded-core/meta/recipes-kernel/linux/linux-yocto_6.1.bb:do_bundle_initramfs)
NOTE: recipe linux-yocto-6.1.38+gitAUTOINC+cba89f406c_b110cf9bbc-r0: task do_bundle_initramfs: Started
NOTE: recipe linux-yocto-6.1.38+gitAUTOINC+cba89f406c_b110cf9bbc-r0: task do_bundle_initramfs: Succeeded
NOTE: Running task 4992 of 5018 (/builds/oe-ros-ci/build-ci/openembedded-core/meta/recipes-kernel/linux/linux-yocto_6.1.bb:do_populate_sysroot)
NOTE: recipe linux-yocto-6.1.38+gitAUTOINC+cba89f406c_b110cf9bbc-r0: task do_populate_sysroot: Started
NOTE: recipe linux-yocto-6.1.38+gitAUTOINC+cba89f406c_b110cf9bbc-r0: task do_populate_sysroot: Succeeded
NOTE: Running task 4993 of 5018 (/builds/oe-ros-ci/build-ci/meta-ros/meta-ros-common/recipes-core/images/ros-image-core.bb:do_image_ext4)
NOTE: recipe ros-image-core-1.0-r0: task do_image_ext4: Started
NOTE: recipe ros-image-core-1.0-r0: task do_image_ext4: Succeeded
NOTE: Running task 4994 of 5018 (/builds/oe-ros-ci/build-ci/meta-ros/meta-ros-common/recipes-core/images/ros-image-core.bb:do_image_tar)
NOTE: recipe ros-image-core-1.0-r0: task do_image_tar: Started
NOTE: recipe ros-image-core-1.0-r0: task do_image_tar: Succeeded
NOTE: Running task 4995 of 5018 (/builds/oe-ros-ci/build-ci/openembedded-core/meta/recipes-kernel/linux/linux-yocto_6.1.bb:do_deploy)
NOTE: recipe linux-yocto-6.1.38+gitAUTOINC+cba89f406c_b110cf9bbc-r0: task do_deploy: Started
NOTE: recipe linux-yocto-6.1.38+gitAUTOINC+cba89f406c_b110cf9bbc-r0: task do_deploy: Succeeded
NOTE: Running task 4996 of 5018 (/builds/oe-ros-ci/build-ci/meta-ros/meta-ros-common/recipes-core/images/ros-image-core.bb:do_image_complete)
NOTE: recipe ros-image-core-1.0-r0: task do_image_complete: Started
NOTE: recipe ros-image-core-1.0-r0: task do_image_complete: Succeeded
NOTE: Running task 5017 of 5018 (/builds/oe-ros-ci/build-ci/meta-ros/meta-ros-common/recipes-core/images/ros-image-core.bb:do_populate_lic_deploy)
NOTE: recipe ros-image-core-1.0-r0: task do_populate_lic_deploy: Started
NOTE: recipe ros-image-core-1.0-r0: task do_populate_lic_deploy: Succeeded
NOTE: Running noexec task 5018 of 5018 (/builds/oe-ros-ci/build-ci/meta-ros/meta-ros-common/recipes-core/images/ros-image-core.bb:do_build)
NOTE: Tasks Summary: Attempted 5018 tasks of which 2152 didn't need to be rerun and all succeeded.
Summary: There were 14 WARNING messages.
Job succeeded
Job succeeded
```

Duration: 122 minutes 26 seconds
Finished: 6 months ago
Queued: 2 seconds
Timeout: 8h (from project)
Runner: #1 (6t54wHXWd)

Commit 620155be
Add MACHINE to BB_ENV_PASSTHROUGH_ADDITIONS

Pipeline #1985 Passed for main

build

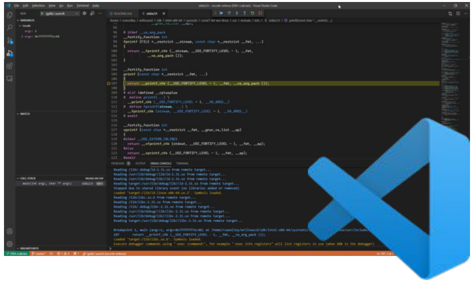
Related jobs

→ build-job

Application Developer

- Bitbake can produce SDKs that include the tools, libraries, and headers needed to do development
- Supports development on the command-line as well as IDE across supported Linux distros
- Community working group has fresh PRs to support building with colcon and Python
- Using CROPS as a DevContainer supports a quick development environment across Linux, Windows, and MacOS

BASH
THE BOURNE-AGAIN SHELL



```
# ifdef __va_arg_pack
__fortify_function int
fprintf (FILE *__restrict __stream, const char *__restrict __fmt, ...)
{
  return __fprintf_chk (__stream, __USE_FORTIFY_LEVEL - 1, __fmt,
            __va_arg_pack ());
}

__fortify_function int
printf (const char *__restrict __fmt, ...)
{
  return __printf_chk (__USE_FORTIFY_LEVEL - 1, __fmt, __va_arg_pack ());
}
# elif !defined __cplusplus
#  define printf(...) \
  __printf_chk (__USE_FORTIFY_LEVEL - 1, __VA_ARGS__)
#  define fprintf(stream, ...) \
  __fprintf_chk (stream, __USE_FORTIFY_LEVEL - 1, __VA_ARGS__)
# endif

__fortify_function int
vprintf (const char *__restrict __fmt, __gnuc_va_list __ap)
{
#ifdef __USE_EXTERN_INLINES
  return __vfprintf_chk (stdout, __USE_FORTIFY_LEVEL - 1, __fmt, __ap);
#else
  return __vprintf_chk (__USE_FORTIFY_LEVEL - 1, __fmt, __ap);
#endif
```

```
Reading /lib/.debug/ld-2.31.so from remote target...
Reading /usr/lib/debug//lib/ld-2.31.so from remote target...
Reading /usr/lib/debug/lib//ld-2.31.so from remote target...
Reading target:/usr/lib/debug/lib//ld-2.31.so from remote target...
Stopped due to shared library event (no libraries added or removed)
Loaded 'target:/lib/ld-linux-x86-64.so.2'. Symbols loaded.
Reading /lib/libc.so.6 from remote target...
Reading /lib/libc-2.31.so from remote target...
Reading /lib/.debug/libc-2.31.so from remote target...
Reading /usr/lib/debug//lib/libc-2.31.so from remote target...
Reading /usr/lib/debug/lib//libc-2.31.so from remote target...
Reading target:/usr/lib/debug/lib//libc-2.31.so from remote target...

Breakpoint 1, main (argc=1, argv=0x7fffffffec48) at /home/rcwoolley/wrlinuxcd/sdk/intel-x86-64/sysroots/
107         return __printf_chk (__USE_FORTIFY_LEVEL - 1, __fmt, __va_arg_pack ());
Loaded 'target:/lib/libc.so.6'. Symbols loaded.
Execute debugger commands using "-exec <command>", for example "-exec info registers" will list registers in use (when GDB is the debugger)
```

Current Status of meta-ros

☑ Support for all maintained Yocto Project and ROS releases

☑ Community support for robotics hardware development kit

☑ Automation for maintenance and CI/CD builds

□ Merge fixes for SDK

□ Complete integration of visualization tools (Gazebo and RViz)

□ Stand-up public GitLab infrastructure

□ Publish artifacts (image and SDK) for download

□ Enhance superflore tool for SBOM / SPDX support

Next Steps

- To get started follow https://github.com/ros/meta-ros/blob/build/kas/README.md
- Help with ROS OpenEmbedded (meta-ros) may be found on the ROS Discourse OpenEmbedded category (https://discourse.ros.org/c/openembedded/26) or OSRF Discord #cwg-openembedded
- Issues and pull requests for meta-ros may be submitted on GitHub: https://github.com/ros/meta-ros/
- Join us at the ROS OpenEmbedded Community Group
    - Bi-weekly meetings on Monday at 3pm UTC (5pm CEST / 11am EDT / 8am PDT)
    - Google Meet: https://meet.google.com/ncq-atrn-wyk
    - Minutes: https://docs.google.com/document/d/1LqUjcu6vdlqVJO62SreCyjzddNDZhfO2n-7qYghY_cQ/edit?usp=sharing

Backup

Supported Combinations

Yocto Release		ROS1 Distros	ROS 2 Distros		
	(Rolling)	Noetic	Humble (LTS)	Iron	Jazzy
		May 2025	May 2027	Nov 2024	May 2029
Walnascar (Dev)	(Future)	May 2025	May 2027	Nov 2024	(Future)
Styhead	(Apr 2025)	May 2025	(April 2025)	Nov 2024	(Apr 2025)
Scarthgap (LTS)	Apr 2028	May 2025	May 2027	Nov 2024	Apr 2028
Nanbield	~~Apr 2024~~	~~Apr 2024~~	~~Apr 2024~~	~~Apr 2024~~	~~Apr 2024~~
Mickledore	~~Nov 2023~~	~~Nov 2023~~	~~Nov 2023~~	~~Nov 2023~~	~~Nov 2023~~
Langdale	~~May 2023~~	~~May 2023~~	~~May 2023~~	~~May 2023~~	~~May 2023~~
Kirkstone (LTS)	Apr 2026	May 2025	Apr 2026	Nov 2024	Apri 2026

Why choose embedded Linux?

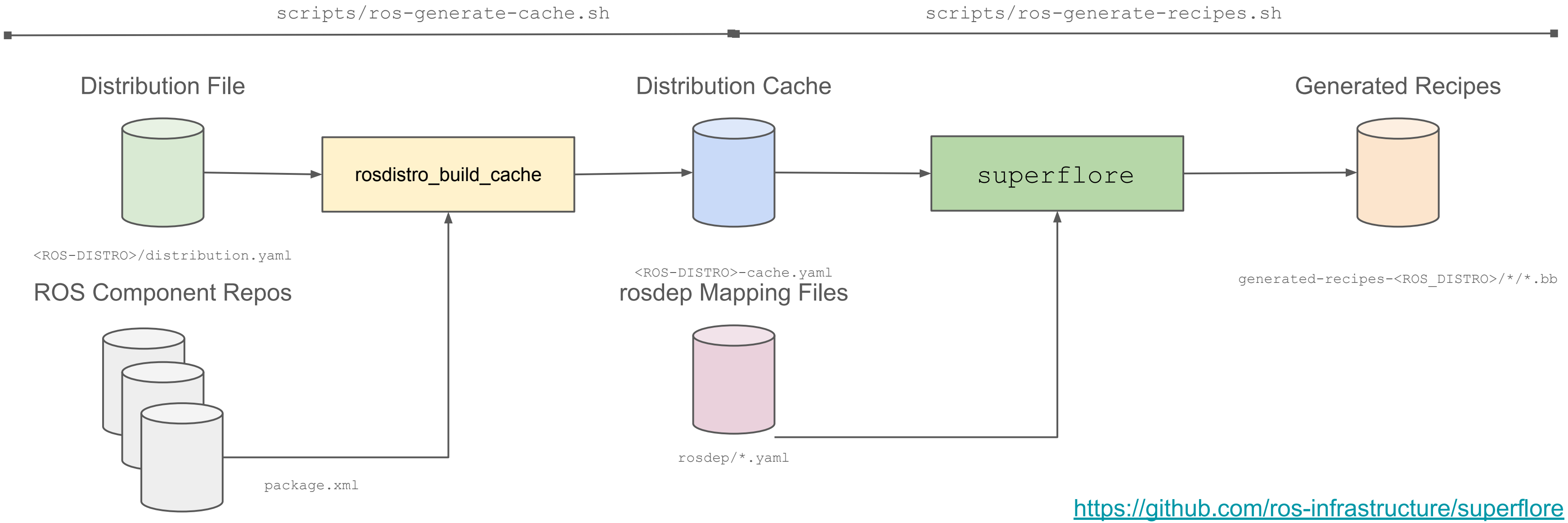
	debian GNU/Linux	openembedded + yocto PROJECT
Hardware	Laptops, PCs, and servers*	SBCs & embedded devices
Primary Distribution Method	Binary images & packages	Source-based build system
Delivery	Ready / Out-of-the-box	Custom / Built to order
Build environment	Self-hosted build	Cross-platform build
Support	Long-Term Support (LTS), Reproducible Builds, Software Bill of Material (SBOM) generation, Commercial Support	
Community	Centralized Community Infrastructure	Decentralized Community with Vendors

* Debian derivatives like Armbian, Raspbian, and eLxr do provide support for select SBCs

INFO FOR MAINTAINERS

meta-ros v2: OpenEmbedded Layers for ROS 1 & ROS 2

Super Flore: An extended platform release manager for ROS



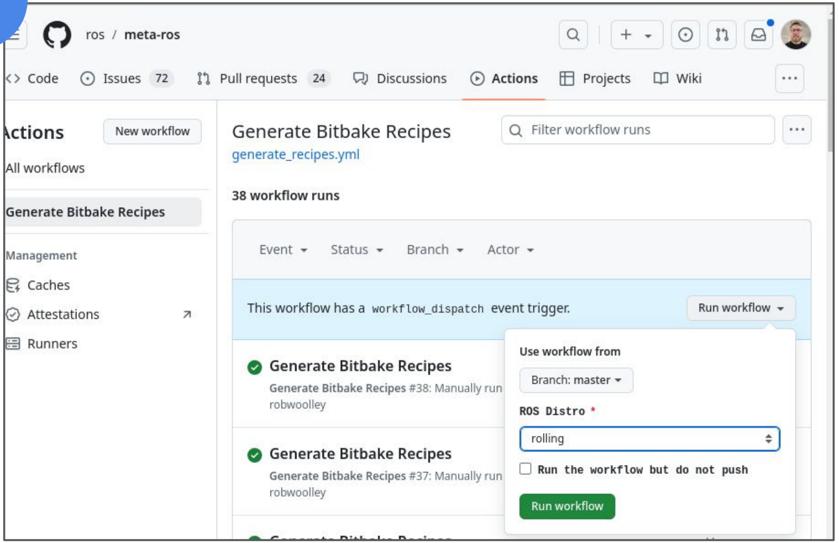
https://github.com/ros-infrastructure/superflore

Credit: ROS on OpenEmbedded Simpler Robotics Development, LG Electronics USA, LG ROSCon 2019

INFO FOR MAINTAINERS

GitHub Actions: Generate Bitbake Recipes

1



Generate Bitbake Recipes
generate_recipes.yml

38 workflow runs

This workflow has a `workflow_dispatch` event trigger. Run workflow

Use workflow from

Branch: master

`ROS Distro` *

rolling

☐ `Run the workflow but do not push`

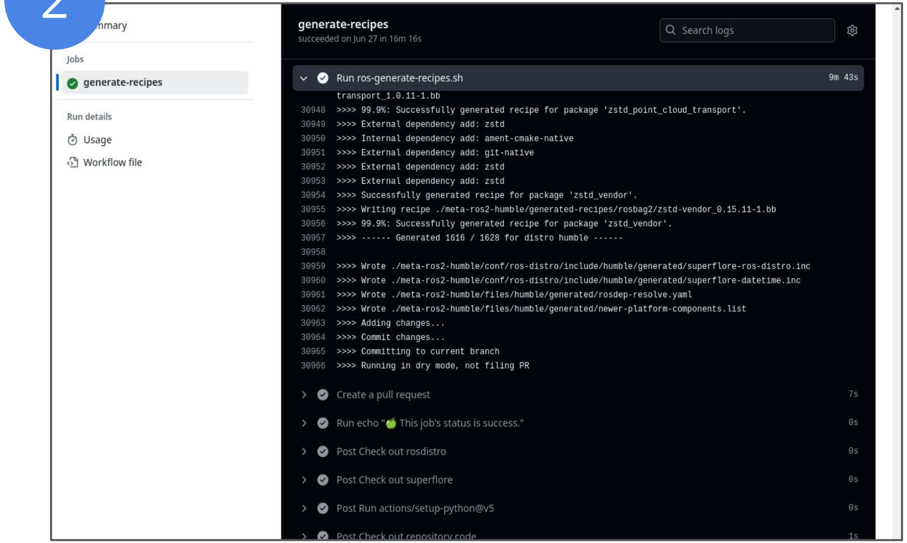
Run workflow

Generate Bitbake Recipes
Generate Bitbake Recipes #38: Manually run
robwoolley

Generate Bitbake Recipes
Generate Bitbake Recipes #37: Manually run
robwoolley

Run the action manually by choosing a Yocto release branch and a ROS Distro

2



generate-recipes
succeeded on Jun 27 in 16m 16s

Run ros-generate-recipes.sh 9m 43s

```
transport_1.0.11-1.bb
>>>> 99.9%: Successfully generated recipe for package 'zstd_point_cloud_transport'.
>>>> External dependency add: zstd
>>>> Internal dependency add: ament-cmake-native
>>>> External dependency add: git-native
>>>> External dependency add: zstd
>>>> External dependency add: zstd
>>>> Successfully generated recipe for package 'zstd_vendor'.
>>>> Writing recipe ./meta-ros2-humble/generated-recipes/rosbag2/zstd-vendor_0.15.11-1.bb
>>>> 99.9%: Successfully generated recipe for package 'zstd_vendor'.
>>>> ------ Generated 1616 / 1628 for distro humble ------

>>>> Wrote ./meta-ros2-humble/conf/ros-distro/include/humble/generated/superflore-ros-distro.inc
>>>> Wrote ./meta-ros2-humble/conf/ros-distro/include/humble/generated/superflore-datetime.inc
>>>> Wrote ./meta-ros2-humble/files/humble/generated/rosdep-resolve.yaml
>>>> Wrote ./meta-ros2-humble/files/humble/generated/newer-platform-components.list
>>>> Adding changes...
>>>> Commit changes...
>>>> Committing to current branch
>>>> Running in dry mode, not filing PR
```

Create a pull request 7s

Run echo "🍏 This job's status is success." 0s

Post Check out rosdistro 0s

Post Check out superflore 0s

Post Run actions/setup-python@v5 0s

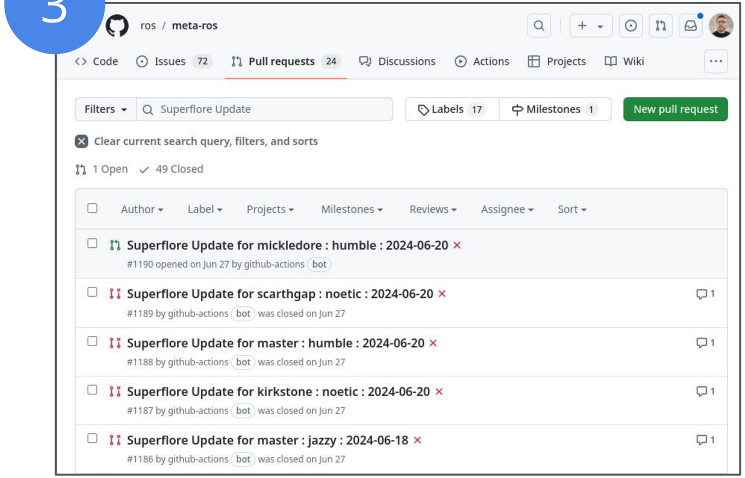
Post Check out repository code 1s

Pulling the ROS packages.xml files from the ROS GitHub repos is much faster (15 to 30 min)

INFO FOR MAINTAINERS

GitHub Actions (part 2)

3



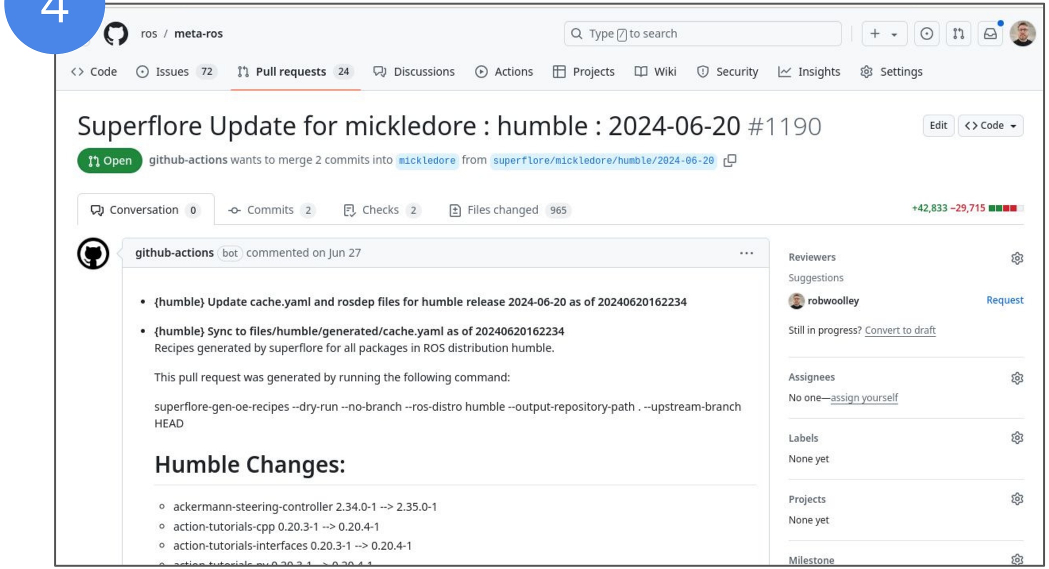
Superflore Update

☒ Clear current search query, filters, and sorts

1 Open ✓ 49 Closed

- □ Superflore Update for mickledore : humble : 2024-06-20 ✕
  #1190 opened on Jun 27 by github-actions bot
- □ Superflore Update for scarthgap : noetic : 2024-06-20 ✕ 1
  #1189 by github-actions bot was closed on Jun 27
- □ Superflore Update for master : humble : 2024-06-20 ✕ 1
  #1188 by github-actions bot was closed on Jun 27
- □ Superflore Update for kirkstone : noetic : 2024-06-20 ✕ 1
  #1187 by github-actions bot was closed on Jun 27
- □ Superflore Update for master : jazzy : 2024-06-18 ✕ 1
  #1186 by github-actions bot was closed on Jun 27

4



Superflore Update for mickledore : humble : 2024-06-20 #1190

Open github-actions wants to merge 2 commits into `mickledore` from `superflore/mickledore/humble/2024-06-20`

+42,833 −29,715

github-actions bot commented on Jun 27

- **{humble} Update cache.yaml and rosdep files for humble release 2024-06-20 as of 20240620162234**
- **{humble} Sync to files/humble/generated/cache.yaml as of 20240620162234**
  Recipes generated by superflore for all packages in ROS distribution humble.

  This pull request was generated by running the following command:

  superflore-gen-oe-recipes --dry-run --no-branch --ros-distro humble --output-repository-path . --upstream-branch HEAD

Humble Changes:

- ackermann-steering-controller 2.34.0-1 --> 2.35.0-1
- action-tutorials-cpp 0.20.3-1 --> 0.20.4-1
- action-tutorials-interfaces 0.20.3-1 --> 0.20.4-1
- action-tutorials-py 0.20.3-1 --> 0.20.4-1

robwoolley

Still in progress? Convert to draft

No one—assign yourself

None yet

None yet

A pull request is automatically created with details on recipes that changed.
From a security perspective, automating the process reduces the need for a in-depth code review.

Pipeline Runs

The pipeline can be run with optional parameters to change the behaviour:

- GITLAB_REDIRECT
  - To redirect git back to locally-hosted repositories
- META_ROS_GIT
  - Set the URL to use for pulling meta-ros
- META_ROS_BRANCH
  - Set the branch to use for the build
- BITBAKE_TARGET
  - Set the target for bitbake to run
- BB_NUMBER_THREADS
  - Set the number of concurrent bitbake tasks to run
- PARALLEL_MAKE
  - Set the number of jobs for Make to use for building a recipe

What is “Embedded” in 2024?

- Devices are connected to the network and remotely accessible
- Physical and remote access introduces new challenges to security
- New requirements for edge computing (eg. physical security, remote updates)
- COTS hardware with custom peripherals (sensors, actuators, accelerators)
- Tuned performance and system engineering tradeoffs (soft RT and SWaP)
- Mixed-Criticality with a Safety-Certified Hypervisor and RTOS with Linux
- Convergence of software domains (eg. AI/ML, Analytics, IT, OT, Security, UX)