Containers and Autoware #3230

YossyTaka started this conversation in **Design**

YossyTaka on Jan 25, 2023

(Repost of the question I posted in Open AD Kit Discord channel for the record)

Howdy. I have a question regarding the usage of containers in the context of Autoware in general.

Could someone please shed a light on the motivation behind splitting Autoware into multiple containers? I ask because it hinders our ability to perform certain optimizations (e.g. "zero-copy" messages where we'd send messages across nodes only with pointers as a payload instead of sending full data.)
Wouldn't it suffice to simply split them into different processes instead of containers? It still satisfies our desire to make it micro-service arch.

Note that I can see the benefit of using containers in a broader context though. For instance, if we are developing against variable combinations of ECUs, then it does make sense to use containers to absorb the differences. But in that case, it suffices to have the entire Autoware in a single container.

Thoughts?

↑ 1 👍 2

Category

Design

Labels

None yet

3 participants

1 comment · 5 replies

Oldest Newest Top

doganulus on Jan 26, 2023 Collaborator edited

Indispensable is modularization and having well-defined modules in Autoware (in any complex system indeed). Among other things, I like the idea of splitting Autoware into containers because it would also enforce strict modularization, which is a good thing for system design and safety. But rightfully, @pukachupenn brings up performance and optimization concerns for this idea. Supporting zero-copy among containers may not be feasible currently.

Yet I would love to see well-defined Autoware components (`perception`, `planning`, `control`, etc.) as if they were to be split into separate containers. One way to achieve this without dedicating ourselves to the idea of multiple containers might be to split build and installation scripts for each component. Then one could make one container or multiple containers out of them easily and actually test both approaches and the granularity in various settings. Thus delaying the original discussion until we have more information on runtime performance implications.

Question:
Do you see any drawback in splitting build and installation scripts for each Autoware component?

↑ 1 5 replies

esteve on Jan 26, 2023 Maintainer

> Question: Do you see any drawback in splitting build and installation scripts for each Autoware component?

This can be easily done by creating metapackages for each component/module that depend on the individual packages, this wouldn't require a priori any extra scripts. In practice, they'd be empty packages that have a package.xml file that point to all its dependencies. So in order to build a component we'd only need to do `colcon build --packages-up-to COMPONENT-METAPACKAGE`

doganulus on Jan 26, 2023 Collaborator edited

Thank you, Esteve. Do you know also a way to install runtime dependencies only from ROS packages?

esteve on Jan 26, 2023 Maintainer edited

If a package is not part of the rosdistro index, you'd need to write what's called a "vendor" package (see https://github.com/ros2/libyaml_vendor/ for an example)

Unfortunately, this is one of many of `rosdep`'s shortcomings, a better solution in my opinion would be to integrate [conan](https://conan.io/ with ROS and get rid of `rosdep`, or add it as a backend for rosdep (see https://discourse.ros.org/t/ros-2-conan-integration/20846?u=esteve for a discussion about it), but for now `rosdep` is all we have.

doganulus on Jan 26, 2023 Collaborator

I have said build and installation scripts above but actually, a single Ansible playbook for each component is what we need. The playbook to install dependencies and call the build tool for the configuration given... Maybe no need for meta-packages as well if dependencies are manually managed in the playbook. I think Ansible-first installation has certain benefits including remote deployment of Autoware.

YossyTaka on Jan 26, 2023 Author

> I am not intimately familiar with each Autoware ROS node, but my guess is that the communications between some nodes would be more verbose and heavier weight than some others. Identifying them first may be a key to decide what could be split across containers without large perf impact vs what would be better off left together.