Maybe use another base image for docker and re-arrange the layers #3469

Unanswered **xmfcx** asked this question in **Ideas**

xmfcx on Apr 28, 2023 Maintainer edited

Right now, the base image for the docker containers in Autoware is `ubuntu:22.04`.

Order of dependencies:

- https://github.com/autowarefoundation/autoware/blob/main/amd64.env
  - ```
    rosdistro=humble
    rmw_implementation=rmw_cyclonedds_cpp
    base_image=ubuntu:22.04
    cuda_base_image=ubuntu:22.04
    prebuilt_base_image=ubuntu:22.04
    cuda_version=11.6
    cudnn_version=8.4.1.50-1+cuda11.6
    tensorrt_version=8.4.2-1+cuda11.6
    ```

- https://github.com/autowarefoundation/autoware/blob/main/.github/workflows/docker-build-and-push-main.yaml#L51
- https://github.com/autowarefoundation/autoware/blob/main/.github/actions/docker-build-and-push/action.yaml#L92
- https://github.com/autowarefoundation/autoware/blob/main/docker/autoware-universe/docker-bake.hcl#L11
- https://github.com/autowarefoundation/autoware/blob/main/docker/autoware-universe/Dockerfile#L2

Another image

Maybe we can use OSRF ROS2 humble desktop image to modularize the deployment of the current docker images?

https://hub.docker.com/layers/osrf/ros/humble-desktop/images/sha256-97f179f6bbcc60c6ffbef88486b3a29a3c79794c0a233e50a9f65130ac5533b5

Current image and how to improve

Right now we have a 15.7GB blob of a layer which we have to download the entirety of it if we do a docker pull.

But if we separate the `dockerfile` build steps into multiple layers, keeping less updated layers at beginning and more updated Autoware related layers at the end.

Category
Ideas

Labels
type:github-actions

7 participants
    ```

This way, our `dockerfile` will be able to make use of reusing layers better.

Here are our current layers and their sizes:

```
echo "| IMAGE ID | CREATED BY | CREATED AT | SIZE | COMMENT |"
echo "| --- | --- | --- | --- |" &&
docker history ce8af253c1a3 --format '| `{{.ID}}` |
`{{.CreatedBy}}` | `{{.CreatedAt}}` | `{{.Size}}` |
`{{.Comment}}` |'

# with useful columns, no trunc:
echo "| \`CREATED BY\` | \`SIZE\` |" && echo "| --- | --- |" &&
docker history ce8af253c1a3 --no-trunc --format '|
`{{.CreatedBy}}` | `{{.Size}}` |' | sed 's/|/\\|/g' | sed 's/\\|
`</| `</g' | sed 's/\\| /| /g' | sed 's/ \\|$/ |/g'
```

CREATED BY
`CMD ["/bin/bash"]`
`RUN |2 ROS_DISTRO=humble SETUP_ARGS=--no-cuda-drivers /bin/bash -o "source /opt/ros/${ROS_DISTRO}/setup.bash" > /etc/bash.bashrc # buil`
`RUN |2 ROS_DISTRO=humble SETUP_ARGS=--no-cuda-drivers /bin/bash -o update && DEBIAN_FRONTEND=noninteractive apt-get install --no-instal software-properties-common && apt-add-repository ppa:kisak/kisak-m DEBIAN_FRONTEND=noninteractive apt-get install --no-install-recommer mesa0 libegl1-mesa-dev libgbm-dev libgbm1 libgl1-mesa-dev libgl1-mes libglx-mesa0 && apt-get clean && rm -rf /var/lib/apt/lists/* # b`
`RUN |2 ROS_DISTRO=humble SETUP_ARGS=--no-cuda-drivers /bin/bash -o /etc/OpenCL/vendors && echo "libnvidia-opencl.so.1" > /etc/OpenCL/ && chmod 644 /etc/OpenCL/vendors/nvidia.icd # buildkit`
`RUN |2 ROS_DISTRO=humble SETUP_ARGS=--no-cuda-drivers /bin/bash -o https://gitlab.com/nvidia/container-images/opengl/raw/5191cf205d3e4bb1150091f9464499b076104354/glvnd/rur -o /etc/glvnd/egl_vendor.d/10_nvidia.json && chmod 644 /etc/glvnd/egl_vendor.d/10_nvidia.json # buildkit`
`RUN |2 ROS_DISTRO=humble SETUP_ARGS=--no-cuda-drivers /bin/bash -o https://gitlab.com/nvidia/container-images/vulkan/raw/dc389b0445c788901fda1d85be96fd1cb9410164/nvidia_ic /etc/vulkan/icd.d/nvidia_icd.json && chmod 644 /etc/vulkan/icd.d/r buildkit`
`RUN |2 ROS_DISTRO=humble SETUP_ARGS=--no-cuda-drivers /bin/bash -o "$HOME"/.cache /etc/apt/sources.list.d/cuda*.list /etc/apt/sources.list.d/docker.list /etc/apt/sources.list.d/nvidia buildkit`
`RUN |2 ROS_DISTRO=humble SETUP_ARGS=--no-cuda-drivers /bin/bash -o dev-env.sh -y $SETUP_ARGS universe && pip uninstall -y ansible ans mkdir src && vcs import src < autoware.repos && rosdep update`

CREATED BY
DEBIAN_FRONTEND=noninteractive rosdep install -y --ignore-src --from rosdistro "$ROS_DISTRO"   && apt-get clean   && rm -rf /var/lib/apt/
RUN |2 ROS_DISTRO=humble SETUP_ARGS=--no-cuda-drivers /bin/bash -o ~/.ssh   && ssh-keyscan github.com >> ~/.ssh/known_hosts # buildkit
RUN |2 ROS_DISTRO=humble SETUP_ARGS=--no-cuda-drivers /bin/bash -o /autoware # buildkit
WORKDIR /autoware
COPY ansible/ /autoware/ansible/ # buildkit
COPY autoware.repos setup-dev-env.sh ansible-galaxy-requirements.ya arm64.env /autoware/ # buildkit
RUN |2 ROS_DISTRO=humble SETUP_ARGS=--no-cuda-drivers /bin/bash -o update && DEBIAN_FRONTEND=noninteractive apt-get -y install --no-ins git   ssh   && apt-get clean   && rm -rf /var/lib/apt/lists/* # buil
ARG SETUP_ARGS
ARG ROS_DISTRO
SHELL [/bin/bash -o pipefail -c]
/bin/sh -c #(nop) CMD ["/bin/bash"]
/bin/sh -c #(nop) ADD file:c8ef6447752cab2541ffca9e3cfa27d581f3491bc8f356f6eafd9512436093
/bin/sh -c #(nop) LABEL org.opencontainers.image.version=22.04
/bin/sh -c #(nop) LABEL org.opencontainers.image.ref.name=ubuntu
/bin/sh -c #(nop) ARG LAUNCHPAD_BUILD_ARCH
/bin/sh -c #(nop) ARG RELEASE

Related issue

#2840

cc. @oguzkaganozt @kenji-miyake @kaspermeck-arm @HamburgDave @armaganarsln

↑ 2 👍 3

4 comments · 28 replies Oldest | Newest | Top

oguzkaganozt on Apr 28, 2023 Maintainer

This layering is not that feasible and efficient right now as we are introducing too many diffs on one layer. There should be better organization when building the Autoware image. Also, there are too many development packages involved in runtime images. We have started putting some effort into addressing some of these issues in this repo: https://github.com/leo-drive/avte_autoware .

↑ 1 👍 2 0 replies

kaspermeck-arm on Apr 28, 2023 Collaborator edited

@xmfcx - start for starting this discussion!

I think that when we're going through this work of rethinking how we want to create and distribute Autoware containers, we should make the split between development environment (DE) and production environment (PE).

I would suggest the following:

- Single DE complete with all tools and dependencies
  - This container can compile and run all of Autoware
- Multi-stage build for PE, inspired by Athackst Dockerfiles
  - Starting with e.g., ROS Core
  - Then incrementally adding more dependencies e.g., math libraries and TVM
  - Finally adding NVIDIA dependencies

Other questions:

- Do we want a DE without NVIDIA?
  - Is it possible to compile NVIDIA binaries without having an NVIDIA GPU?
- Do we want source code to be part of the DE container?
- Should built binaries be part of the PE containers?

Inspiration:

- ROS Official Dockerfiles
  - Specifically Humble Core (runtime only) and Humble Base
- Athackst ROS Dockerfiles
- SUSE Development Environment
- SUSE Production Environment

I would like to join and discuss this topic. Is this being discussed in the SW working group?

↑ 1 👍 2 6 replies

Show 1 previous reply

kaspermeck-arm on Apr 28, 2023 Collaborator

@xmfcx - thanks for your response!

> > I think that when we're going through this work of rethinking how we want to create and distribute Autoware containers, we should make the split between development environment (DE) and production environment (PE).
>
> Yes, I agree. Right now I'd like to focus on the **development environment first**, since that's what the majority of the Autoware users might benefit first. (And me, lol) And once we make it more efficient, we can focus on the PE image too.
>
> So I will focus my answers on single image development Dockerfile for now.

I agree, it makes sense to start with the DE. If we can try to label or categorize ROS packages with their specific dependencies, that'll be helpful later when creating the PEs. Is there a convenient way to get this information, e.g., from the `CMakeLists.txt` files for each ROS package?

> > Single DE complete with all tools and dependencies
>
> This is what I'd like to improve first. And have this Dockerfile structured with multi-stage builds.
>
> And also to take advantage of Docker's caching mechanism when building Docker images with multi-stage builds, we should structure the Dockerfile so that the layers that are likely to change frequently (like autoware) are in the final stage of the Dockerfile.
>
> This will help Docker reuse the cached layers from the earlier stages of the Dockerfile when building the final image, which can save time and resources during the build process.

Regarding the DE. Effectively, would we have a two-stage build; one without NVIDIA and one with NVIDIA? (with is based, i.e., `FROM`, on without)

> > Do we want source code to be part of the DE container?
>
> I also thought about this, and I think, it's unnecessary because we already mount the autoware folder into the docker and use it from there anyways. But even if we are to include it, we shouldn't include the `.git` folders in them since they are too large. (autoware.universe .git folder is about 600MB)
>
> But we can also add it as the last stage of the multi stage build.

I agree it's unnecessary.

> > I would like to join and discuss this topic. Is this being discussed in the SW working group?
>
> Right now, SW working group focuses on general issues since we have 3 new working groups:

```
* Perception & Sensing WG
* Planning & Control WG
* Localization & Mapping WG
```

> This is my first time bringing this issue up, so we can talk about this issue in the Software WG.

If you want, we can make this a topic in the Open AD Kit WG. This work fits nicely with some of the *dojos* we have in the pipeline. What do you think?

👍 1

xmfcx on Apr 28, 2023 Maintainer Author

> If we can try to label or categorize ROS packages with their specific dependencies, that'll be helpful later when creating the PEs. Is there a convenient way to get this information, e.g., from the CMakeLists.txt files for each ROS package?

For the scope of this discussion, it's not necessary, since I just want to reorganize the layers of the single image and if possible, reuse existing images.

But for exploring the dependency tree of a package/s, you can use https://colcon.readthedocs.io/en/released/reference/verb/graph.html
(But this will only give the ROS2 package dependencies)

> Regarding the DE. Effectively, would we have a two-stage build; one without NVIDIA and one with NVIDIA? (with is based, i.e., FROM, on without)

Right now, we have:

- https://github.com/orgs/autowarefoundation/packages/container/autoware-universe/85618306?tag=humble-latest-amd64
- https://github.com/autowarefoundation/autoware/pkgs/container/autoware-universe/85632914?tag=humble-latest-prebuilt-amd64
- https://github.com/autowarefoundation/autoware/pkgs/container/autoware-universe/85621839?tag=humble-latest-cuda-amd64
- https://github.com/autowarefoundation/autoware/pkgs/container/autoware-universe/70625594?tag=humble-latest-prebuilt-cuda-amd64

and their arm counterparts. Right now, I don't have a plan to change current images.

> If you want, we can make this a topic in the Open AD Kit WG. This work fits nicely with some of the dojos we have in the pipeline. What do you think?

Right now, I am just collecting thoughts of developers who might be interested.
I prefer discussing it offline for now. And if anyone is interested to tackle the issues, I am open for it.

👍 1

doganulus on Apr 29, 2023 Collaborator edited

> Do we want the source code to be part of the DE container?

I would say no. The source code must be mounted and built in the DE. Here it may be also a good idea to split a headless Build Environment (BE), which can build Autoware and is usable in CI/CD, and add the developer layer (with graphical tools) on top of the BE.

> Should built binaries be part of the PE containers?

I would say yes. Binaries and their runtime dependencies only. I tried but found it hard to extract runtime dependencies though. It needs a discussion or dojo, I think.

👍 1

kaspermeck-arm on May 1, 2023 Collaborator edited

@doganulus, thanks for your comment, I agree with the source code not being included in the DE and the binaries included in the PE.

> > Do we want the source code to be part of the DE container?
>
> I would say no. The source code must be mounted and built in the DE. Here it may be also a good idea to split a headless Build Environment (BE), which can build Autoware and is usable in CI/CD, and add the developer layer (with graphical tools) on top of the BE.

Would the DE expand the BE? (i.e., DE using `from` BE)

Edit:
I've been reading a bit more about the different environment and "staging" seems to be a term that frequently comes up. Not sure how a stage and a build environment would differ. **@doganulus** , are you familiar with this term?

doganulus on May 2, 2023 Collaborator edited

@kaspermeck-arm By a build environment, I mean a container where all build dependencies are installed and ready to build Autoware. This includes the compiler, required system libraries, and third-party dependencies. I also like to put testing tools in my build environments.

Autoware developers would also need extra tools, especially graphical tools, then I would love to add them as a separate layer. And, yes, this extends the build environment using the `FROM` statement. There might be other ways though if we use Ansible as discussed below.

Not sure if I understand `staging` in this context. But maybe some elaborate workflows need to distinguish multiple build environments, say, to check newer versions of dependencies for example. Then this can make sense and might help for easier migrations.

> The other thought: A staging environment would mean Autoware runtime environment + simulation environment where the scenario-based tests run. I think this is closer to the meaning of the term in web development.

kenji-miyake on May 1, 2023

@xmfcx Thank you for raising this. I'll give you some related information.

> Right now we have a 15.7GB blob of a layer which we have to download the entirety of it if we do a docker pull.

The largest part (maybe over 10GB) is CUDA, which will be improved by #3084.

> Maybe we can use OSRF ROS2 humble desktop image to modularize the deployment of the current docker images?

It could be, but please note that OSRF doesn't provide CUDA images. Maybe we need to create some base images containing ROS and CUDA by ourselves. (For example, in `autowarefoundation/autoware-docker-base-images` or something?)

> But if we separate the dockerfile build steps into multiple layers, keeping less updated layers at beginning and more updated Autoware related layers at the end.

It's a good practice, but please note that (although you already know that) splitting it into too many layers may increase maintenance costs.

↑ 1

9 replies

Show 4 previous replies

kaspermeck-arm on May 1, 2023 Collaborator

@kenji-miyake, thanks for the explanation, I think I understand.

Why are we using ansible?

kenji-miyake on May 1, 2023

> Why are we using ansible?

@kaspermeck-arm Just because we don't have enough resource to maintain an elaborated Dockerfile. If someone can maintain it, it's okay for me to change Dockerfile drastically.

👍 1

kenji-miyake on May 1, 2023

> NVIDIA support is a trickier one... Are there legal implications redistributing built containers including NVIDIA support?

@kaspermeck-arm I believe there is no problem if we don't redistribute CUDA Driver Libraries.
Related: autowarefoundation/autoware_core_universe_prototype#338
(But I'm not so confident.)

👍 1

doganulus on May 2, 2023 Collaborator edited

We can do it only via Ansible roles and playbooks. In Dockerfile, we would call the playbook(s) and build the development container. The same playbooks can be used locally for those preferring local development or remotely for bare metal deployments. And I think we can refactor the current setup around Ansible and remove shell scripts, `.env` files, and `.repos` files. Ansible would easily cover the functionality of these tools and more. This would be especially helpful for newcomers and students who are often lost in the initial setup/build process.

👍 1

xmfcx on May 2, 2023 Maintainer Author

@doganulus I was thinking the same, I think it's the cleanest solution.

I think @kenji-miyake is raising the issue of maintenance of manually keeping the role names up to date in case of future refactorings:
@kenji-miyake said:

> - If you split this into multiple steps, at least you need to write some role names in Dockerfile.
> - That means, when you modify roles, you need to edit both playbooks and Dockerfile, which is a little increase of maintenance costs.

But as long as someone does this work and documents it well to help future maintainers, I think it's a worth investment.

As @kenji-miyake said,

> Just because we don't have enough resource to maintain an elaborated Dockerfile. If someone can maintain it, it's okay for me to change Dockerfile drastically.

It's a matter of who will take the responsibility and work on the task.

@doganulus said:

> And I think we can refactor the current setup around Ansible and remove shell scripts, .env files, and .repos files. Ansible would easily cover their functionality of these tools and more.

- `.env` files are useful for also making the manual installation steps easier too. And they are used across CI in general.
- `.repos` file is used wor `vcs` and I like using it and have no problems with it.

But I think these 2 topics should be discussed in another discussion topic.

👍 2

xmfcx on May 8, 2023 Maintainer Author

Here is an updated version of my idea of reconstructed images.

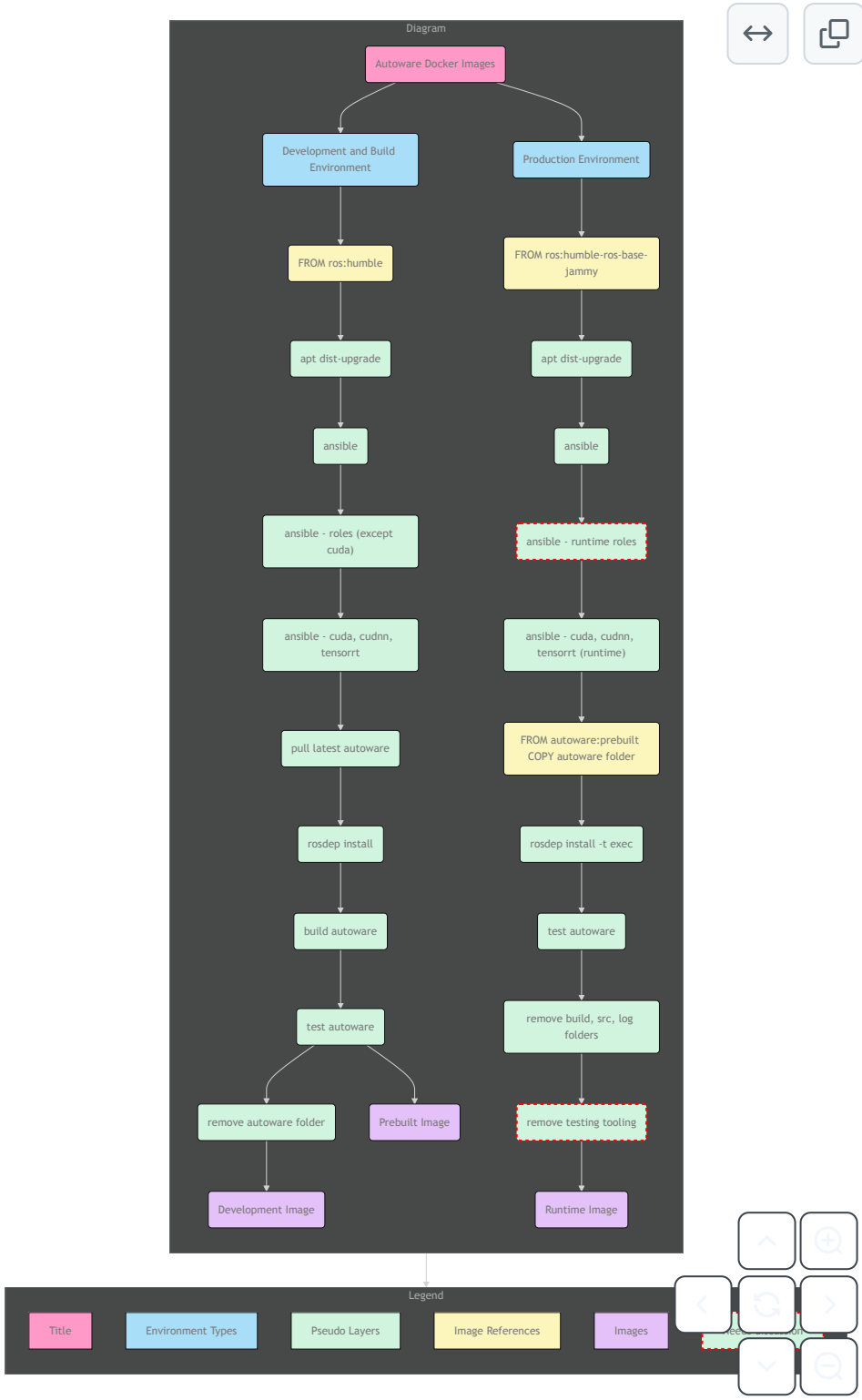
Changes

- Instead of osrf/ros:humble-desktop
  - Use ros:rolling-ros-base-jammy and ros:humble
  - Because it has arm64 versions and also light and heavy versions are divided.

Related ideas with debian packaging of Autoware

- Runtime image can also be created by the resulting debian installation method of Generate Debian packages for the ROS packages in Autoware.core and Autoware.universe

Rough Dockerfiles Diagram



I'd like to hear your opinions about this way of restructuring the layers and images.

cc. @oguzkaganozt @kenji-miyake @kaspermeck-arm @HamburgDave @armaganarsln @doganulus @esteve

↑ 1 👍 4 13 replies

Show 8 previous replies

oguzkaganozt on May 11, 2023 Maintainer

Some comments and findings about Autoware containers:

- Right now we have monolithic image which includes all dependencies and executables in one docker image. I think we should have a **minimal custom base image** (on top of **osrf:ros-base**) to have a good starting point for specific runtime images. Also having a custom base image can be more maintainable then we can focus more on specific needs in other images.

- Also our image contains unnecessary dependencies for runtime this puts significant overhead to the size of our images; to fix this we should change our ansible playbook to be modular as possible so that we can install only necessary dependencies for specific needs.

- Regarding the need to have source code to run the rosdep in production environment, maybe we can dump the rosdep keys into a file inside prebuilt image then just copy the file and install the keys inside the production image. For dumping the keys we can use something like:
  `rosdep install -y --ignore-src --from-paths src --simulate --reinstall --rosdistro "$ROS_DISTRO"`

👍 1

kaspermeck-arm on May 16, 2023 Collaborator

@oguzkaganozt, thanks for your comments!

> ```
> * Right now we have monolithic image which includes dependencies and executables in one docker image. I think we should have a **minimal custom base image** (on top of **osrf:ros-base**) to have a good starting point for specific runtime images. Also having a custom base image can be more maintainable then we can focus more on specific needs in other images.
> ```

How many dependencies are there for all of Autoware? E.g.:

- ROS core (this is the baseline)
  - This is the lowest level of dependency
- Other ROS dependencies
- Math libraries
  - E.g., BLAS implementations such as ArmPL
- Simulators
- CUDA
  - TensorRT, CuDNN, etc...

How do we find these dependencies? I assume on a per-node basis?

> ```
> * Also our image contains unnecessary dependencies f
> runtime this puts significant overhead to the size o
> our images; to fix this we should change our ansible
> playbook to be modular as possible so that we can
> install only necessary dependencies for specific needs.
> ```

I agree that having a modular approach will make it easier to build custom images. Basically, each module is an independent dependency install.

> ```
> * Regarding the need to have source code to run the rosdep in production environment, maybe we can dump the rosdep keys into a file inside prebuilt image then just copy the file and install the keys inside the production image. For dumping the keys we can use something like:
>   `rosdep install -y --ignore-src --from-paths src --simulate --reinstall --rosdistro "$ROS_DISTRO"`
> ```

I like this approach, then the method which TUM uses would work to create rosdep keys for the different features.

VRichardJP on May 22, 2023 Collaborator edited

@xmfcx Sorry I am late at the party:

I am not sure I get why Autoware should be built, then deleted in the dev image. The dev image can stop right after the rosdep install step. If you want to validate the dockerfile, then creating the derived images prebuilt/production should be good enough, no?

As @oguzkaganozt said, it would not be absurd to maintain a list of dependencies in a file (I am pretty sure it could be updated automatically). Actually, I wish it was possible to provide a custom list of dependencies to install when building the docker images locally, because I don't have only autoware packages in my workspace. Then it would not be necessary to clone the whole autoware project.

@xmfcx

> Right now we have a 15.7GB blob of a layer which we have to download the entirety of it if we do a docker pull.
> But if we separate the dockerfile build steps into multiple layers, keeping less updated layers at beginning and more updated Autoware related layers at the end.
> This way, our dockerfile will be able to make use of reusing layers better.

One "annoying" issue I think no one is mentionning is this:

https://github.com/autowarefoundation/autoware/blob/main/docker/autoware-universe/Dockerfile#L21

```
COPY ansible/ /autoware/ansible/
```

Any change to any of the playbook would invalidate all the layers after. So if the point is to split the ansible build phase into multiple layers, then the ansible files must be split and copied separately as well, right?

👍 1

xmfcx on May 22, 2023 Maintainer Author edited

@VRichardJP

> I am not sure I get why Autoware should be built, then deleted in the dev image. The dev image can stop right after the rosdep install step. If you want to validate the dockerfile, then creating the derived images prebuilt/production should be good enough, no?

You are right, I was fixated on the testing of the Autoware containers, I didn't think that through, I've updated the diagram.

> As **@oguzkaganozt** said, it would not be absurd to maintain a list of dependencies in a file (I am pretty sure it could be updated automatically). Actually, I wish it was possible to provide a custom list of dependencies to install when building the docker images locally, because I don't have only autoware packages in my workspace. Then it would not be necessary to clone the whole autoware project.

I've added the `Exec Dependencies` artifact/image thing to the diagram, this should solve both of the problems.

> Any change to any of the playbook would invalidate all the layers after. So if the point is to split the ansible build phase into multiple layers, then the ansible files must be split and copied separately as well, right?

You are right, we should split and copy the necessary parts at necessary layers.

@oguzkaganozt I've updated the diagram with your suggestions too.

> Right now we have monolithic image which includes all dependencies and executables in one docker image. I think we should have a minimal custom base image (on top of osrf:ros-base) to have a good starting point for specific runtime images. Also having a custom base image can be more maintainable then we can focus more on specific needs in other images.

In the diagram I've shared, I'm using ros:humble-ros-base-jammy for the runtime image because it has both ARM64 and AMD64 versions.

And for the dev environment, using ros:humble.

Is this in line with your suggestion?

> how are Development and Production Environments validated?

@kaspermeck-arm I've moved the testing/validation out of the scope of the dockerfiles.
We should do the testing in the CI which also builds these images.

.

👍 1

kaspermeck-arm on May 23, 2023 Collaborator

@xmfcx

This new diagram looks good! In the PE we can have a multi-stage build making, e.g., the CUDA dependencies optional.

👍 1