Proposal to change Docker image tag naming conventions #4995

youtalk started this conversation in **Design**

Category

Design

Labels

type:containers

component:openad...

5 participants

youtalk on Jul 16 Collaborator

edited by xmfcx

The current tag names for Autoware's Docker images have the following issues:

- The use of dates in tag names does not align with Autoware's version control.
- Running `docker pull ghcr.io/autowarefoundation/autoware:latest` pulls a very old image.
- There are tags prefixed with `humble-`, but currently, there are no other distribution options besides the Humble distribution.
- Since there are `devel` images for container development, pulling a `prebuilt` image has no practical use.
- CUDA drivers are installed on both the `prebuilt` and `runtime` images, which share the same parent `base` image.

Therefore, I would like to propose changing the naming conventions for Docker image tags as shown in the following table. I expected the release processes of `X.Y.Z` tags will be carried out by the owners of `autowarefoundation`.
I would like to discuss the release process in a separate post. This would involve running comprehensive scenario tests and real vehicle tests.

Stage	CUDA	The current tags	
		amd64	arm64
base	without CUDA drivers	autoware:latest-base autoware:20YYMMDD-base	autoware:latest-base-arm64 autoware:20YYMMDD-base-arm64
	with CUDA drivers	NA	NA
prebuilt	without CUDA drivers	autoware:latest-prebuilt autoware:20YYMMDD-prebuilt	autoware:latest-prebuilt-arm64 autoware:20YYMMDD-prebuilt-arm64

Stage	CUDA	The current tags	
	with CUDA drivers	autoware:latest-prebult-cuda autoware:20YYMMDD-prebuilt-cuda	autoware:latest-prebult-cuda-arm64 autoware:20YYMMDD-prebuilt-cuda-arm64
devel	without CUDA drivers	autoware:latest-prebuilt autoware:20YYMMDD-devel	autoware:latest-devel-arm64 autoware:20YYMMDD-devel-arm64
	with CUDA drivers	autoware:latest-prebult-cuda autoware:20YYMMDD-devel-cuda	autoware:latest-prebult-cuda-arm64 autoware:20YYMMDD-devel-cuda-arm64
runtime	without CUDA drivers	autoware:latest-runtime autoware:20YYMMDD-runtime	autoware:latest-runtime-arm64 autoware:20YYMMDD-runtime-arm64
	with CUDA drivers	autoware:latest-prebult-cuda autoware:20YYMMDD-runtime-cuda	autoware:latest-prebult-cuda-arm64 autoware:20YYMMDD-runtime-cuda-arm64

Finally, I aim to support multiple platforms by using the `docker manifest create` command, combining `amd64` and `arm64` images to a single tag image. Thanks to **@oguzkaganozt** 's contributions, the partial work for this has already been completed.
https://github.com/autowarefoundation/autoware/blob/main/.github/actions/combine-multi-arch-images/action.yaml

Related Issues

- **Remove the latest keyword from docker tags** #5175

↑ 1 👍 2

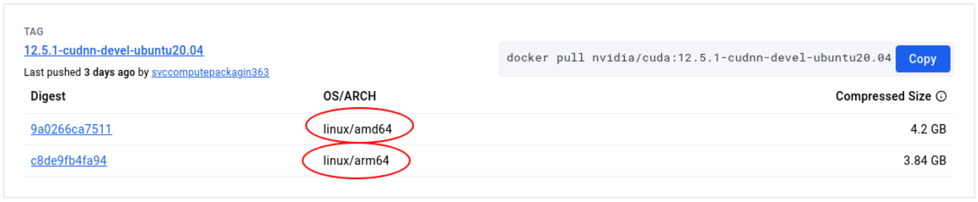
4 comments · 8 replies

Oldest Newest Top

xmfcx on Jul 16 Maintainer

Looking at NVIDIA docker image naming conventions, I see they don't add CPU architecture field to the tag name.
https://hub.docker.com/r/nvidia/cuda/tags

Maybe we can do the same, what do you think?


TAG
12.5.1-cudnn-devel-ubuntu20.04
Last pushed **3 days ago** by svccomputepackagin363

```
docker pull nvidia/cuda:12.5.1-cudnn-devel-ubuntu20.04
```

Digest	OS/ARCH	Compressed Size
9a0266ca7511	linux/amd64	4.2 GB
c8de9fb4fa94	linux/arm64	3.84 GB

They use the same tag for both `amd64` and `arm64`.

↑ 1

2 replies

youtalk on Jul 16 Collaborator Author

@xmfcx I think so. That is the multiple platforms build which I mentioned above.

> Finally, I aim to support multiple platforms by using the `docker manifest create` command, combining `amd64` and `arm64` images to a single tag image. Thanks to **@oguzkaganozt** 's contributions, the partial work for this has already been completed.
> https://github.com/autowarefoundation/autoware/blob/main/.github/actions/combine-multi-arch-images/action.yaml

I hope we will be able to support the multiple platforms build by the next step.

👍 1

xmfcx on Jul 16 Maintainer

I agree with the rest of proposed changes btw. 👍 Thanks!

❤️ 1

oguzkaganozt on Jul 16 Maintainer

Let me summarize what I understand from different flavour of docker images of Autoware:

- `base`: common base for all docker images
- `devel`: obviously for developers
- `prebuilt`: for CI-CD pipeline we seperated this from `devel` because of the insufficient disk space on github action runners.
- `runtime`: for end-user and direct deployment

So prebuilt image is only meant to be used in CI-CD pipeline if we remove this image then we need to update and re-check all workflows from `autoware` and `autoware-universe` repositories.

As you said `latest` is not active at the current state because we planned to make it active once openadkit official release is out but hence this was postponed so many times, I think we can enable it on each build by default.

So all in all I agree with your changes. 👍

↑ 1 ❤️ 1 2 replies

evshary on Jul 16 Collaborator

Could I ask what is the difference between `prebuilt` and `runtime`? I know one is for CI usage while the other is for deployment, but I'm still not sure what is different inside their content.

About the tag, perhaps we can also add the release one. In previous images, there is something like `humble-2024.03-prebuilt-arm64` for fixed version `2024.03` in the Autoware repository. It would be great if we can keep this policy.

Another thing might be out of the topic: is it possible to move the legacy image (in `autoware-universe` before) to the `ghcr.io/autowarefoundation/autoware`. It would be helpful since some projects rely on these legacy images.

Thank you all for your efforts!

doganulus on Jul 16 Collaborator

@evshary The best thing is we can ask that `autoware` repo can be docker-buildable via the repo URL.

https://docs.docker.com/reference/cli/docker/image/build/#git-repositories

Then, we can build legacy images at any time in history with zero maintenance. This is why I often argue against the existing custom script (`setup.sh`, `build.sh`, `run.sh`, etc.) in the `autoware` repo because they prevent such nice features.

doganulus on Jul 16 Collaborator edited ▾

You probably do not need the `prebuilt` tag as you would not want to package prebuilt binaries again, now distributed by the `runtime` image.

You need a `build` or `builder` image that can build Autoware (core, universe, or both) in the CI and else. And it should not include the source code and the prebuilt binaries. For cache purposes, you can document the ability to use the build cache from the registry (`--cache-from`).

Runtime images should be leaner as much as possible. Headless, and without any build tools and devel libraries. These images are the end product that goes to customers.

I prefer `cuda` images as the default if this is what you prefer, which I assume is the case. Hence, you can consider `nocuda` tag to differentiate cuda-less version. Anyway, how much of Autoware can be usable without Cuda acceleration, and how serious is the performance degradation?

BONUS: What would be nice: Optimize `devel` images for `distrobox` use.
That gives the best developer experience of both local and container worlds.

↑ 2 👍 2 4 replies

xmfcx on Jul 18 Maintainer

CUDA dependent parts of Autoware

> Anyway, how much of Autoware can be usable without Cuda acceleration, and how serious is the performance degradation?

Here is the list of packages that depend on CUDA:

1. bytetrack
2. cuda_utils
3. image_projection_based_fusion
4. lidar_apollo_instance_segmentation
5. lidar_centerpoint
6. lidar_transfusion
7. tensorrt_classifier
8. tensorrt_common
9. tensorrt_yolox
10. traffic_light_classifier
11. traffic_light_fine_detector

Their short descriptions

Module Name	Description
bytetrack	Object tracking algorithm for dynamic objects
cuda_utils	Utility functions for CUDA-based computations
image_projection_based_fusion	Fuses sensor data based on image projections
lidar_apollo_instance_segmentation	Segments objects in lidar data using Apollo model
lidar_centerpoint	Lidar-based object detection using CenterPoint
lidar_transfusion	Fuses lidar data for enhanced perception
tensorrt_classifier	Object classification using TensorRT
tensorrt_common	Common utilities for TensorRT integration

Module Name	Description
tensorrt_yolox	YOLOX object detection optimized with TensorRT
traffic_light_classifier	Classifies traffic light labels using cropped images with `cnn_classifier` and `hsv_classifier`
traffic_light_fine_detector	Detects traffic lights using YoloX-s and CNN-based methods

Without CUDA, these won't even compile and many perception tasks won't function at all. So for Autoware perception component heavily depends on CUDA. The rest of the components function without CUDA.

About usage of `nocuda` against `cuda`

> I prefer cuda images as the default if this is what you prefer, which I assume is the case. Hence, you can consider nocuda tag to differentiate cuda-less version.

I agree we that we recommend and use CUDA version by default. But still, I think it's better to use the `cuda` tag since it is an addition and can change if enough effort is exerted (e.g. transitioning to TVM from TensorRT).

doganulus on Jul 18 Collaborator

The regular user would pull `ghcr.io/autowarefoundation/autoware:latest`. Do you want them to use the version with CUDA or not? This is the question. You can still keep the `-cuda` suffix as another alias, of course.

Also, you could test `lean` or `dispatch` runtimes for TensorRT, which are very lightweight, but I don't know how much performance degradation they could cause. If they are not so bad, that can be the default.

👍 1

xmfcx on Jul 18 Maintainer

> The regular user would pull `ghcr.io/autowarefoundation/autoware:latest`. Do you want them to use the version with CUDA or not? This is the question.

I think `autoware:latest` should point to cuda version with runtime with lightest setup that works out of box without user needing to download anything else. This wouldn't be for dev usage, optimized for just running the autoware. not even the git histories would be present nor the src folder.

doganulus on Jul 18 Collaborator

Runtime images should include only binaries (build artifacts) and their runtime dependencies. Nvidia is doing it right for example.

The problem is that even Quality-1 ROS packages do not differentiate `runtime` and `devel` packages. So, the runtime image would be bloated with `gcc`, `python3-dev`, `javascript`, `numpy`, and many devel libraries if a package declares dependency to `rclcpp`.

👍 1

xmfcx on Jul 18 Maintainer edited

@VRichardJP has also some requests:

- **Lightweight devel build for docker** #4991

Related discussion:

- Slimming down the container image size #5007
  - Relevant comment under the discussion #5007

Summary:

I'd like an image with:

- all devel tools installed for CI purposes
- no artifacts
- no autoware folder

An additional image that derives from that image can be created that contains the artifacts and the autoware folder (if necessary)

↑ 1 0 replies