I-VICS R4 Installation Document

Introduction

> The supported environments are specified in System Dependencies and Target Environments.

The recommended method for installation is through the use of ADE, a Docker-based tool to ensure that all developers in a project have a common, consistent development environment. It comes with a pre-built version of Autoware.Auto, so that you will not need to compile it yourself if you do not want to.

Pre-Installation Requirements

- Hardware Requirements

`amd64` / `x86_64` (Intel/AMD)

`arm64` / `aarch64` / `arm64v8` (ARM v8, 64-bit)

- Software Prerequisites

Target Software Platforms

ROS Version	Operating System	System Dependencies
ROS2 Foxy (**active development**)	Ubuntu 20.04 LTS	REP-2000 section
ROS2 Dashing (**maintenance only**)	Ubuntu 18.04 LTS	REP-2000 section

Install ADE

ADE is a modular Docker-based tool to ensure that all developers in a project have a common, consistent development environment.

Follow the install instructions, which are reproduced here for convenience:

1. Verify that the requirements listed here are fulfilled. In particular, if docker was not used before, one may need to go through the docker post-install steps.
2. Download the latest statically-linked binary for your platform from the Releases page of the `ade-cli` project
3. Name the binary `ade` and install it in `PATH`. On Ubuntu, `/usr/local/bin` is recommended for system-wide installation, otherwise choose e.g. `~/.local/bin` for a local installation that doesn't require `sudo` rights.
4. Make the binary executable: `chmod +x ade`
5. Check that it is installed:

```
$ which ade
/path/to/ade
$ ade --version
<version>
```

Setup ADE home and project checkout

ADE needs a directory on the host machine which is mounted as the user's home directory within the container. The directory is populated with dotfiles, and must be different than the user's home directory *outside* of the container. In the event ADE is used for multiple, projects it is recommended to use dedicated `adehome` directories for each project.

ADE looks for a directory containing a file named `.adehome` starting with the current working directory and continuing with the parent directories to identify the ADE home directory to be mounted.
$ mkdir -p ~/adehome
$ cd ~/adehome
$ touch .adehome

For ADE to function, it must be properly configured. Autoware.Auto provides an .aderc file which is expected to exist in the current working directory, or in any parent directory. Additionally, default configuration values can be overridden by setting environment variables. See the `ade --help` output for more information about using environment variables to define the configuration.
$ cd ~/adehome
$ git clone https://gitlab.com/autowarefoundation/autoware.auto/AutowareAuto.git

Sharing files between the host system and ADE

It might come in handy to share files such as dotfiles or utility programs from your host machine with ADE. If you only have a single `adehome` directory, there is a way to do that without duplicating them: move them inside the `adehome` directory, then create a symlink in the host system to their regular location. For instance,
$ cd ~
$ mv ~/.bashrc ~/ade-home/.bashrc
$ ln -s ~/ade-home/.bashrc

It will then appear as `~/.bashrc` to the host system and to ADE.

Another option is to put utility programs into `~/adehome/.local/bin` and symlink. The opposite direction will not work, files in a Docker container can not be symlinks to the outside.

NoteThe programs have to be self-contained! They should not depend on loading libraries from e.g. /usr/lib.

Entering the development environment

$ cd AutowareAuto

To start the default environment:
$ ade start --update --enter

There are several preconfigured environments to choose from by specifying an ADE rc file. To see what is available, run
ls -l .aderc*

Choose one, then launch with:
ade --rc .aderc-amd64-foxy start --update --enter

Congratulations! Now you should have a terminal inside ADE:
$ade:~$

The next steps are to proceed to Usage, or to work on the Autoware.Auto code itself as described in Contributor's guide.

What is where inside ADE?

Upon entering, ADE outputs the images used to create the environment; e.g.
$ ade enter
Entering ade with following images:
ade-foxy | 8b1e0efdde07 | master | registry.gitlab.com/autowarefoundation/autoware.auto/autowareauto/amd64/ade-foxy:master
binary-foxy | 0e582f863d4c | master | registry.gitlab.com/autowarefoundation/autoware.auto/autowareauto/amd64/binary-foxy:master
foxy | 2020.06 | 2020.06 | registry.gitlab.com/autowarefoundation/autoware.auto/ade-lgsvl/foxy:2020.06

The images are mounted under `/opt`:
@ade:~$ ls /opt
AutowareAuto # image: binary-foxy:master
lgsvl # image: ade-lgsvl/foxy:2020.06
ros # image: ade-foxy:master

The code in `/opt/AutowareAuto` is built from a particular version of the master branch of Autoware.Auto. The master branch is built multiple times a day in CI; see the container registry. With `ade ... --update`, the latest available version of each image is downloaded.

Cleanup

ADE uses Docker, and over time unused images, containers, and volumes begin to clutter the hard drive. Follow the steps below to clean the Docker file system of stale images.

Start relevant Docker resources

First, verify that ADE is running:
$ cd ~/adehome/AutowareAuto
$ ade start

If ADE is used for more than one project, verify all ADE instances are running; the same rule applies for any other non-ADE Docker containers that should be preserved.

NoteDocker resources that are not started/running will be removed!

Docker disk usage

To assess the disk usage situation, run the following command:
$ docker system df
TYPE TOTAL ACTIVE SIZE RECLAIMABLE
Images 13 11 14.03GB 916.9MB (6%)
Containers 11 0 2.311MB 2.311MB (100%)
Local Volumes 17 15 5.411GB 17.8MB (0%)
Build Cache 0 0 0B 0B

Remove unused docker items

Use `docker system prune` to remove any Docker items not used for currently running containers:
$ docker system prune -a --volumes

Troubleshooting

Here are solutions for a few specific errors:

Error - "forward compatibility was attempted on non supported hw" when starting ADE

When starting `ade` with GPU support enabled for NVIDIA graphics, you may sometimes receive the following error:
docker: Error response from daemon: OCI runtime create failed: container_linux.go:349: starting container process caused "process_linux.go:449: container init caused \"process_linux.go:432: running prestart hook 0 caused \\\"error running hook: exit status 1, stdout: , stderr: nvidia-container-cli: initialization error: cuda error: forward compatibility was attempted on non supported hw\\\\n\\\"\"": unknown.
ERROR: Command return non-zero exit code (see above): 125

This usually indicates that a new NVIDIA graphics driver has been installed (usually via `apt`) but the system has not yet been restarted. A similar message may appear if the graphics driver is not available, for example because of resuming after suspend.

Solution

Restart your system after installing the new NVIDIA driver.

Error - "Unable to create the rendering window after 100 tries" when launching GUI application

If you have an NVIDIA GPU and are using the proprietary NVIDIA GPU driver, you may encounter this error when using the default `.aderc` or `.aderc-arm64` files. This is due to a decision that was made regarding support for users with and without NVIDIA GPUs and those with and without the proprietary NVIDIA driver. For more information you can review the discussion that lead to this decision in this issue.

To resolve this issue, simply remove the line `export ADE_DISABLE_NVIDIA_DOCKER=true` from the `.aderc` file that you are using and restart `ade` wit h:
ade$ exit
$ ade stop
$ ade start --update --enter