Docker tutorial

Goal

In this tutorial, you will learn how to effectively work with a Docker container to simulate the TIAGo robot and connect VSCode to the running container.

Setting up the Docker image

If you did not do before, pull the docker image from Agimus using one of the following methods:

```
docker pull reg.saurel.me/aws
docker pull reg-w.saurel.me/aws
docker pull gitlab.laas.fr:4567/gsaurel/aws-docker
```

Please take in mind the following remark of Guilhem:

- **reg.saurel.me/aws** if you are on the Eduroam network, and you'll get fast local wifi download
- **reg-w.saurel.me/aws** if your are on the wired ethernet network, and you'll get super fast local wired download
- **gitlab.laas.fr:4567/gsaurel/aws-docker** is now available for other situations, but you'll download over internet, so it's going to take a while

Note: If for some reason the docker image of Agimus is not working for you can you can use the following steps to pull the original image from PAL Robotics
Begin by logging in to the GitLab container registry using the following command. Use your GitLab credentials for login:

```
docker login -u agimus-user -p frpTR_--SSsbKWRJkK5V registry.gitlab.com
```

Pull the Docker image from the container registry:

```
docker pull
registry.gitlab.com/pal-robotics/agimus_winter_school/dockers/ros2:latest
```

Configuring pal_docker_utils

The pal_docker_utils package is a utility package that helps create your docker container with extra functionalities. The container will have access to ssh and the GPU. Furthermore, the container is accessed as a non-root user. These functions are essential when developing software for TIAGo. Clone the package in your home directory:

```
cd ~
git clone https://github.com/pal-robotics/pal_docker_utils.git
```

Create the following folders. These folders will both be available from your laptop and from the docker container. The *exchange* folder is where the ROS workspaces can be created. The *docker_mounts/alum* folder may contain any home folder configurations such as a *.bashrc* file.

```
mkdir -p ~/exchange
mkdir -p ~/docker_mounts/alum
```

Note: If these folders are not created beforehand, the docker will create them automatically. In some cases the folders will be created with root access only. When this happens, the access to these folders has to be manually set to *user:host_group* in order for the container to function properly.

Creating the Docker Container

Use the script from pal_docker_utils to start the Agimus Docker container. This will create and launch a container with the name agimus:

```
./pal_docker_utils/scripts/pal_docker.sh -it --name agimus -v
~/docker_mounts/alum:/home/user/
registry.gitlab.com/pal-robotics/agimus_winter_school/dockers/ros2:latest bash
```

Setup environment

After starting the container, open a new Terminator session from within Docker. This allows you to have multiple independent terminal windows inside the container:

```
terminator -u
```

Add the following two lines to the *~/.bashrc* file. More information about the ROS_DOMAIN_ID can be found here. In the course each student will be assigned a unique ROS_DOMAIN_ID to avoid interference of communication.

```
echo 'export ROS_DOMAIN_ID=XX' >> ~/.bashrc
echo 'source /opt/pal/alum/setup.bash' >> ~/.bashrc
```

Test simulation

To test if everything is working correctly, run the following command:

```
ros2 launch tiago_gazebo tiago_gazebo.launch.py
```

Create ROS 2 workspace

Navigate to the *exchange* folder in the docker and create a new folder, *agimus_ws*, that contains the folder *src*:

```
cd ~/exchange
mkdir -p agimus_ws/src
```

In the next step the package play_motion2_msgs is added to the workspace.

Note: If you are using the Agimus docker you can use the following command:

```
cp -r ~/play_motion2/play_motion2_msgs ~/exchange/agimus_ws/src/
```

Note: If you are using the original pal docker image you can clone the following repository in the agimus_ws folder:

```
git clone https://github.com/pal-robotics/play_motion2.git
rm -r play_motion2/play_motion2
```

Once packages are created in this workspace they can be build with the command:

```
colcon build
```

To use the run packages in the workspace the workspace has to be sourced. Open another terminal and source:

```
source ~/exchange/agimus_ws/install/setup.bash
```

Your environment should now be ready to use.

Connect VSCode to Docker container

Once the container is running it is possible to attach and develop in the container by using VSCode. This is done by opening the extension panel (Shortcut Ctrl+Shift+X) and installing the extension *dev containers*. More information on this option can be found here.

With the extension installed you can open the command palette (Shortcut Ctrl+Shift+P) and type the command and then select the desired container.

```
Dev Containers: Attach to Running container
```

Once the container is opened in VSCode use the command palette again to run the command below, this will open the configuration file for the container.

```
Dev Containers: Open Container Configuration File
```

In the configuration file some preferences can be set up for the specific container. One important setting is to set the remote user correctly. In the configuration file add the following:

```
"remoteUser": "user",
```

After setting *remoteUser*, restart VSCode and attach to the container again. It should now be using the *user* instead of *root.*