Technical documentation

version 1.0

Obsah

Introduction **3**

Main features **3**

Technologies **4**

Installation and preparations 4

Launching the application **5**

Dockerfile 5

Launcher 5

Entrypoint 5

Line clean service 6

Python application 6

Script wait-for-it.sh 6

Introduction

The main goal of this application is to demonstrate knowledge of new technologies such as docker, ROS, Python, and some of its libraries. In docker container runs turtlesim node, own clear service, and rosbrigde. On the host computer there run a python application that uses library roslibpy to communicate with ros in a docker container and publishing on its topics. Communication is shown on the GUI application with the turtle where the python application moves with the turtle via arrow keys.

Main features

- own image base on image ros:melodic defined in dockerfile
- turtlesim node and clear service run in a container
- turtlesim GUI shows communications with python application on the host computer
- python application publishing on turtlesim topic and move with turtle via arrows keys
- the python application uses roslibpy to communicate with ros in a container
- publishing to docker container via rosbridge_websocket and tf2_web_republisher
- cleaning trajectory line behind the turtle with clear service

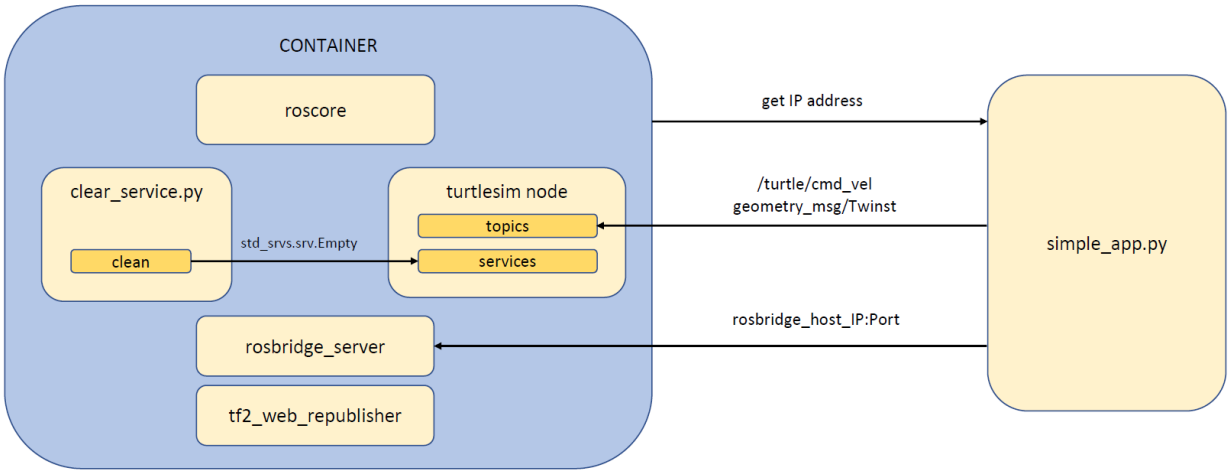

Figure 1 application schematic

Technologies

Following technologies were used to develop this application:

- Operating system Ubuntu 18.04
- Docker – version 20.10.2
- Python – version 2.7.17, with these libraries
    - Roslibpy
    - Docker
    - Pynput
- Ros:melodic docker image
- Script wait-for-it.sh

To run the application or continue with developing you have to have installed all mentioned technologies.

Installation and preparations

Follow these commands to install necessary software packages:

- Python
    - *sudo apt-get install python*
- Python pip
    - *sudo apt-get install python-pip*
- To install all python dependencies run this command:
    - *bash install_python_dependencies.sh*

When you want to run the docker command you have to be logged as the root user or the currently logged user has to be a member of the docker group. Here is the command that adds the currently logged user to the docker group.

- *bash setup_user_groups.sh*

Launching the application

Use this command to launch the application:

- *bash launcher.sh*

After any modification, you can also use this command. It builds docker and runs the python application. Description of launcher file is in chapter launcher.

Description of files in the project

In this section, a short description of source files can be found.

Dockerfile

In the dockerfile, is written the description of the new image. A new image is based on ros:melodic image. First of all the required software packages are installed. Then the catkin workspace is built. Some files are copied to a container and set as executable. In the end, the entrypoint is defined.

Launcher

In this file, all processes are started. There is also a command which builds a new image. You don't need to build a new image particularly after any changes in dockerfile. After running a container the python application starts. When the application is closed, the running container is killed and removed.

Entrypoint

Inside the entrypoint file are defined processes that have to be executed. These are the processes:

- *roscore*
- *turtlesim node*
- *rosbridge_server and rosbridge_websocket*
- *tf2_web_republisher*
- *clear_service.py*

Each process is running in the background. To ensure that all processes will launch correctly, script wait-for-it.sh is used.

Line clean service

This is a very simple service that cleans the trajectory behind the turtle in the GUI application. It runs periodically ten times per second.

Python application

This application connects to the ros inside the created container. After a successful connection, it is publishing on the turtle topic using arrow keys. This causes movement of the turtle.

Script wait-for-it.sh

This script is used inside the entypoint. It waits for the process defined in its argument. When that process is running it will run the desired command.