Software Tooling and Setup Guide

Contents

- Contents
- Software Tooling Choices
    - Software Tooling
    - Brief Justification of Tooling Choices
- Environment Setup Guide
    - Virtual Machine Setup
        - For Mac Users
    - Step 0
    - Step 1a Auto installation
    - Step 1b Manual Installation
    - Step 3
    - Step 4
    - Step 5
    - Step 6
    - Step 7
    - Step 8
    - (Optional) Step 9
    - Step 10
    - Troubleshooting
        - ROS Noetic Installation Isn't Working
        - You Don't Have sudo Privileges on your Virtual Machine
        - No AVX2 Support
        - Enabling Full-Screen for Virtual Box
        - Enabling Shared Clipboard Between Native OS and Virtual Machine

Software Tooling Choices

Software Tooling

The following tools are to be used in the project:

- ROS Noetic
- Python 3.8.10
- Ubuntu 20.04.6
- pip
- VirtualBox
- venv

Brief Justification of Tooling Choices

ROS Noetic: This is the latest and final version of ROS, as of May this year it is the only ROS distribution which will be actively maintained. The client, Wafa Johal, asked us to use ROS, so this distribution is the logical choice.

Ubuntu 20.04.6: Each ROS distribution is targeted at a specific Ubuntu platform, ROS Noetic is targeted at 20.04, so this is the sensible choice. The reason for 20.04.6 is one of practicality; this is the version currently available for download on ubuntu.com.

Python 3.8.10: This is the version of Python which comes default on Ubuntu 20.04, since this distro is targeted at ROS Noetic, we can be fairly certain there will be no compatibility issues. This makes it the best choice for a version of Python 3.x to use, and the client has requested that we use Python 3.x.

Pip: Several developers on the team have had the experience of being unable to install packages with Conda and Poetry, pip seems to be most reliable. Additionally, generating requirements.txt using conda list will not show dependencies installed with pip, so for the sake of reproducibility, it is easier to keep track of dependencies using pip.

VirtualBox: None of the developers are using Ubuntu 20.04 as their main OS, this is freely available and easy to use software which allows us to run Ubuntu 20.04 on both Mac and Windows.

Venv: as we are using a virtual machine for development, it should not be necessary to maintain Python virtual environments. If this becomes necessary for any developer, using venv is recommended as it is a lightweight tool which is separate from any package manager.

Environment Setup Guide

These instructions assume the user is not running Ubuntu 20.04 as their main operating system. Those who are can obviously skip the steps relating to the set-up of ubuntu on a virtual machine; however, these users should be conscious about setting up a fresh Python virtual environment for this project, as there may be dependency conflicts between this project and packages installed on their default system Python.

Virtual Machine Setup

First, download VirtualBox for your OS, we recommend using 7.0.8 as this is what the team is using. You can download it here: https://www.virtualbox.org/wiki/Downloads

Next, you need to download the ISO file for Ubuntu 20.04.6, which you can find here: https://releases.ubuntu.com/focal/

The next step is to create a Ubuntu 20.04 virtual machine on VirtualBox, please see here for instructions (steps 2 and 3 are most relevant here - note it is recommended to install with Guest Additions): https://ubuntu.com/tutorials/how-to-run-ubuntu-desktop-on-a-virtual-machine-using-virtualbox#2-create-a-new-virtual-machine

For Mac Users

You will need to install AVX2 support, run the following in your VM Terminal:

```
$ sudo apt update
$ sudo apt install libmkl-dev libmkl-avx2 -y
        # (option page: ok, no)
```

Step 0

Clone the repository:

```
sudo apt install git -y
git clone https://github.com/COMP90082-2023-SM1/NA-Boxjelly.git
```

Step 1a Auto installation

For simplicity's sake, we've created a setup.sh file that automatically setup and verify your development environment, build necessary dependencies, project packages, and optionally allows you to run test files.

```
cd <path to NA-Boxjelly>
./setup.sh
# follow on-screen options to select 1 (Yes) or 2 (No) when asked to run test cases
```

IMPORTANT: replace `<path-to-NA-Boxjelly>` with the path to the `/NA-Boxjelly` directory.

Skip the following steps and proceed to **Step 10** if using the auto installation method.

Step 1b Manual Installation

If something is wrong with the auto installation, the following steps will show how to do the installation manually.

First setup the ROS Noetic package by running:

```
sudo sh -c 'echo "deb http://packages.ros.org/ros/ubuntu $(lsb_release -sc) main" > /etc/apt/sources.list.d/ros-latest.list'
sudo apt install curl -y
curl -s https://raw.githubusercontent.com/ros/rosdistro/master/ros.asc | sudo apt-key add -
sudo apt update
sudo apt install git python3-pip ros-noetic-desktop-full -y
echo "export PATH="$PATH:$HOME/.local/bin"" >> ~/.profile
```

You can also checkout the more detailed installation guide

Once ROS Noetic is set up, you must run the following source command in every bash terminal you are going to use ROS in:

```
source /opt/ros/noetic/setup.bash
```

If you want to have multiple terminals open when using ROS Noetic, it is useful to add this source command to your .bashrc file as follows:

```
echo "source /opt/ros/noetic/setup.bash" >> ~/.bashrc
source ~/.bashrc
```

Step 3

Install dependencies for building ROS packages:

```
sudo apt install python3-rosdep python3-rosinstall python3-rosinstall-generator python3-wstool build-essential
sudo rosdep init
rosdep update
```

Step 4

Required Python packages installation:

```
cd <path-to-NA-Boxjelly>
pip install -r requirements.txt
```

Step 5

Install the packages needed to build boost from source:

```
pip3 install qibuild scikit-build toml
sudo apt install libgmock-dev patchelf -y
echo "PATH=$PATH:$HOME/.local/bin" >> ~/.bashrc
source ~/.bashrc
```

Step 6

Boost 1.77.0 is needed for us to build libqi-python from source.
Download boost from this link, and extract it by running the following:

```
curl -L https://boostorg.jfrog.io/artifactory/main/release/1.77.0/source/boost_1_77_0.tar.bz2 -o boost_1_77_0.
tar.bz2
tar -xjf boost_1_77_0.tar.bz2
cd boost_1_77_0
./bootstrap.sh
./b2 --with=all -j 4 install --prefix=$HOME/boost_1_77_0
echo "export BOOST_ROOT=$HOME/boost_1_77_0" >> ~/.bashrc
echo "export Boost_INCLUDE_DIR=$HOME/boost_1_77_0/include" >> ~/.bashrc
echo "export Boost_LIBRARY_DIR=$HOME/boost_1_77_0/lib" >> ~/.bashrc
```

Step 7

Build and install libqi-python from source by following the code below:

```
git clone https://github.com/aldebaran/libqi-python.git
cd libqi-python
sed -i "s+'-DQIPYTHON_STANDALONE:BOOL=ON'+'-DQIPYTHON_STANDALONE:BOOL=ON', '-DBOOST_ROOT:PATH=$HOME
/boost_1_77_0', '-DBoost_INCLUDE_DIR:PATH=$HOME/boost_1_77_0/include', '-DBoost_LIBRARY_DIR:PATH=$HOME
/boost_1_77_0/lib'+g" setup.py
python3 ./setup.py bdist_wheel -j 4
cd dist
pip3 install <qi.whl file>
```

IMPORTANT: replace `<username>` with your ubuntu username.
IMPORTANT: replace `<qi.whl file>` with the qi.whl file found in `/dist`.

Step 8

Build the Cowriter package by running the following:

```
cd <path-to-NA-Boxjelly>

rm -rf build/*
rm -rf devel/*

catkin_make --only-pkg-with-deps chilitags_catkin
catkin_make --only-pkg-with-deps ros_markers
catkin_make -DCATKIN_WHITELIST_PACKAGES=""

source <path-to-NA-Boxjelly>/devel/setup.bash

# run this once after catkin_make is run to ensure the
# build files can be sourced when launching a new terminal
echo "source <path-to-NA-Boxjelly>/devel/setup.bash" >> ~/.bashrc

# setup the database path in src/catkin_ws/src/choose_adaptive_words/nodes/parameters.py
# PATHDB='<path-to-NA-Boxjelly>/ui_database'"
```

(Optional) Step 9

Run test cases:

```
cd <path-to-NA-Boxjelly>
catkin_make run_tests
```

Step 10

Install Choregraphe, the NAO robot simulator:

Download setup.run file in https://www.aldebaran.com/en/support/nao-6/downloads-softwares

```
cd Downloads
chmod +x choregraphe-suite-2.8.6.23-linux64-setup.run
sudo ./choregraphe-suite-2.8.6.23-linux64-setup.run
```

Run the simulator:

```
/opt/'Softbank Robotics'/'Choregraphe Suite 2.8'/bin/choregraphe_launcher
```

In case of error:

```
/opt/Softbank Robotics/Choregraphe Suite 2.8/bin/../lib/../lib/../lib/libz.so.1:
version `ZLIB_1.2.9' not found (required by /usr/lib/x86_64-linux-gnu/libpng16.so.16)
```

run the following:

```
cd /opt/'Softbank Robotics'/'Choregraphe Suite 2.8'/lib/
sudo mv libz.so.1 libz.so.1.old
sudo ln -s /lib/x86_64-linux-gnu/libz.so.1
```

Troubleshooting

ROS Noetic Installation Isn't Working

You may need to delete the key and add a different one before it will install, try running the following which should resolve the issue.

```
$ sudo apt-key del 421C365BD9FF1F717815A3895523BAEEB01FA116
$ sudo -E apt-key adv --keyserver 'hkp://keyserver.ubuntu.com:80' --recv-key
C1CF6E31E6BADE8868B172B4F42ED6FBAB17C654
$ sudo apt clean && sudo apt update
$ sudo apt install git pip ros-noetic-desktop-full
```

You Don't Have sudo Privileges on your Virtual Machine

Please follow these steps:

```
$ su
$ visudo
```

- Under the heading '# User privilege specification' add the line: `<username> ALL=(ALL:ALL) ALL`
- Save and exit the permissions text file

```
$ exit
$ sudo -l
# <user password>
```

This should confirm that you can run sudo commands. required sudo privileges.

No AVX2 Support

Due to the need for AVX2 support to run the VM on Mac, if the host machine has HyperV enabled, we will need to turn it off. (if using VirtualBox, should see a *V on a chip* (AMD-V), instead of a *V on a turtle* (HyperV) on button right of the window)

If you see "V on a turtle", in host machine terminal:

```
$ bcdedit /set hypervisorlaunchtype off
$ DISM /Online /Disable-Feature:Microsoft-Hyper-V
# (restart host machine)
```

Enabling Full-Screen for Virtual Box

```
$ sudo apt install build-essential gcc make perl dkms
$ y
```

1. Reboot the Ubuntu VM (power down then log back on)
2. In the top left of the VirtualBox window, click the 'Devices' button
3. From the drop-down menu, click 'Insert Guest Additions CD Image..'
4. A prompt should come up on the Ubuntu screen asking if you want to run BVox_GAs_7.0.6, select 'Run'
5. Power down the VM then start it again and log in
6. Try minimise then maximise the screen, with some luck it will go full screen
7. If this has not worked, right click the CD-ROM icon on the left toolbar and click 'Eject'
8. Repeat steps 2-6, if it didn't work the first time, it should work the second

Enabling Shared Clipboard Between Native OS and Virtual Machine

Note: It is useful, but not strictly necessary, to share a clipboard between your VM and normal OS. To enable this, click 'Devices' in the top left of the VirtualBox widow, hover your cursor over 'Shared Clipboard' then select 'Bidirectional'.