Make ROS & Gazebo Easy to Use

Using ROS and Gazebo can get a bit complicated from time to time. The following provides a list of commands and a Bash script containing a list of aliases for executing ROS-related commands. Before running the commands, please put `init.bash` to `/opt/ros` and then source it via `source /opt/ros/init.bash`

List of Commands

loadros	Initializes ROS environment
turtlebot_sim	Starts default TB simulation
turtlebot_rviz	Launches RVIZ in TB
turtlebot_teleop	Launches TB keyboard controls
turtlebot_willowgarage	Starts TB simulation with WG world
turtlebot_willowgarage_headless	Starts headless TB simulation (WG)
sawyer_sim	Starts Sawyer simulation

Contents of /opt/ros/init.bash

```
# ROS Aliases
alias loadros='source /opt/ros/kinetic/setup.bash'

# Turtlebot
alias turtlebot_sim='roslaunch turtlebot_gazebo turtlebot_world.launch'
alias turtlebot_rviz='roslaunch turtlebot_rviz_launchers view_robot.launch'
alias turtlebot_teleop='roslaunch turtlebot_teleop keyboard_teleop.launch'

# Turtlebot worlds
alias turtlebot_willowgarage='roslaunch turtlebot_gazebo turtlebot_world.launch
world_file:=worlds/willowgarage.world'
alias turtlebot_willowgarage_headless='roslaunch turtlebot_gazebo
turtlebot_world.launch world_file:=worlds/willowgarage.world gui:=false
headless:=true'

# Sawyer Robot
alias sawyer_sim='roslaunch sawyer_gazebo sawyer_world.launch'
```

Headless Gazebo Tutorial

Running a single Gazebo server

This operation will run `gzserver` without `gzclient`.

Add **`gui:=false`** and **`headless:=false`** to the roslaunch command. For example:

```
$ roslaunch turtlebot_gazebo turtlebot_world.launch
world_file:=worlds/willowgarage.world gui:=false headless:=true
```

Running multiple Gazebo servers

Open a new terminal window and type **`loadros`**. Then type the following:

```
$ export ROS_MASTER_URI=http://localhost:11311
$ export GAZEBO_MASTER_URI=http://localhost:11341
```

Then, run the headless Gazebo instance as described in the previous section.

For the same computer, after typing the above, open a new terminal window and type

```
$ export ROS_MASTER_URI=http://localhost:11312
$ export GAZEBO_MASTER_URI=http://localhost:11342
```

Then, launch the headless ROS instance as described in the previous section. As a result, you will have 2 separate Gazebo servers accessible via the IP and the port defined in ROS_MASTER_URI variable.

- For the same computer launching multiple Gazebo server instances, i.e. headless Gazebo, only change the port number.
- For cluster having multiple nodes, only change “localhost” with the IP address. Make sure that the ports are open in all cluster nodes.

Installing ROS on Windows

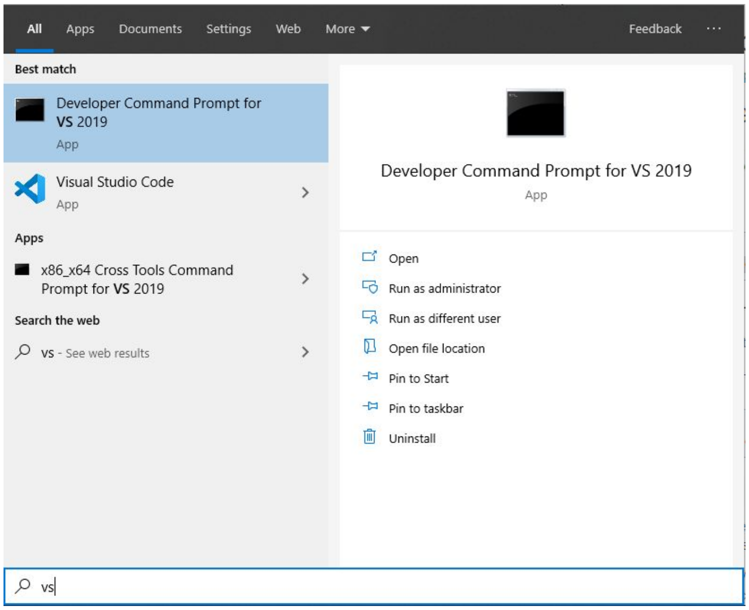
It is possible to install ROS Melodic on Windows via Chocolatey package manager.

Requirements

1. Visual Studio 2019
    a. Go to https://azureforeducation.microsoft.com/devtools
    b. Sign in with your IASTATE Net ID (you will be redirected to login.iastate.edu)
    c. Download Visual Studio 2019 Enterprise (VS2019)
    d. Make sure that you included the following workloads while installing VS2019
        i. Desktop Environment with C++
        ii. Universal Windows Application
    e. Follow on-screen instructions to complete installation
2. Chocolatey Package Manager
    a. Follow the instructions on https://chocolatey.org/

Install ROS

1) Run VS2019 Command Prompt as administrator



2) Add ROS repository to chocolatey

```
$ choco source add -n=ros-win -s="https://roswin.azurewebsites.net/api/v2"
--priority=1
```

3) Install ROS Melodic packages

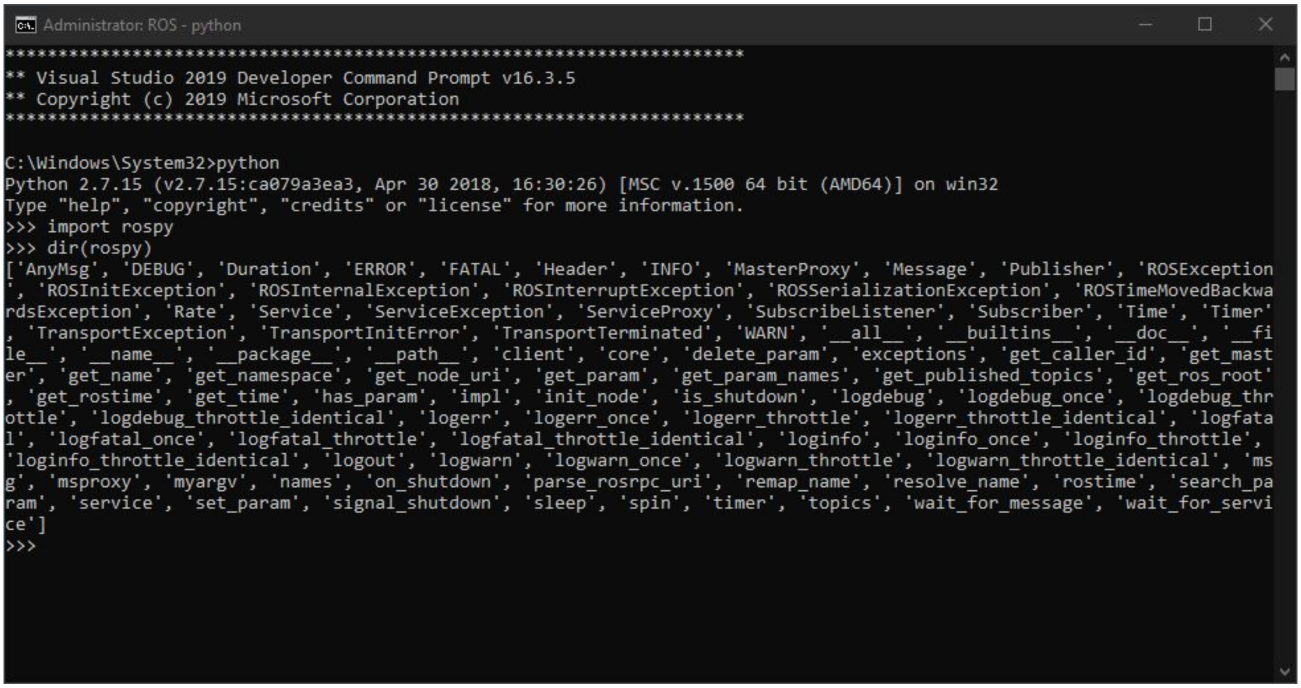
```
$ choco upgrade ros-melodic-desktop_full -y --execution-timeout=0
```

4) Create a shortcut for ROS console

1. Right-click on desktop
2. Select *New - Shortcut*
3. Paste `C:\Windows\System32\cmd.exe /k "C:\Program Files (x86)\Microsoft Visual Studio\2019\Enterprise\Common7\Tools\VsDevCmd.bat" -arch=amd64 -host_arch=amd64 && c:\opt\ros\melodic\x64\setup.bat` in the text box.
4. Name the shortcut “ROS Console”
5. Make sure that the ROS console runs as administrator
    a. Right click on the shortcut
    b. Select *Properties*
    c. Select *Shortcut* tab on the new window
    d. Click *Advanced* tab
    e. Select *Run as administrator*
    f. Close the window

5) Double-click on the shortcut to run ROS console.

6) The following screenshot shows opening the Python prompt, importing `rospy` package and executing `dir(rospy)` on the ROS console.



```
**********************************************************************
** Visual Studio 2019 Developer Command Prompt v16.3.5
** Copyright (c) 2019 Microsoft Corporation
**********************************************************************

C:\Windows\System32>python
Python 2.7.15 (v2.7.15:ca079a3ea3, Apr 30 2018, 16:30:26) [MSC v.1500 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> import rospy
>>> dir(rospy)
['AnyMsg', 'DEBUG', 'Duration', 'ERROR', 'FATAL', 'Header', 'INFO', 'MasterProxy', 'Message', 'Publisher', 'ROSException
', 'ROSInitException', 'ROSInternalException', 'ROSInterruptException', 'ROSSerializationException', 'ROSTimeMovedBackwa
rdsException', 'Rate', 'Service', 'ServiceException', 'ServiceProxy', 'SubscribeListener', 'Subscriber', 'Time', 'Timer'
, 'TransportException', 'TransportInitError', 'TransportTerminated', 'WARN', '__all__', '__builtins__', '__doc__', '__fi
le__', '__name__', '__package__', '__path__', 'client', 'core', 'delete_param', 'exceptions', 'get_caller_id', 'get_mast
er', 'get_name', 'get_namespace', 'get_node_uri', 'get_param', 'get_param_names', 'get_published_topics', 'get_ros_root'
, 'get_rostime', 'get_time', 'has_param', 'impl', 'init_node', 'is_shutdown', 'logdebug', 'logdebug_once', 'logdebug_thr
ottle', 'logdebug_throttle_identical', 'logerr', 'logerr_once', 'logerr_throttle', 'logerr_throttle_identical', 'logfata
l', 'logfatal_once', 'logfatal_throttle', 'logfatal_throttle_identical', 'loginfo', 'loginfo_once', 'loginfo_throttle',
'loginfo_throttle_identical', 'logout', 'logwarn', 'logwarn_once', 'logwarn_throttle', 'logwarn_throttle_identical', 'ms
g', 'msproxy', 'myargv', 'names', 'on_shutdown', 'parse_rosrpc_uri', 'remap_name', 'resolve_name', 'rostime', 'search_pa
ram', 'service', 'set_param', 'signal_shutdown', 'sleep', 'spin', 'timer', 'topics', 'wait_for_message', 'wait_for_servi
ce']
>>>
```

Upgrade ROS

Run the following command:

```
$ choco upgrade ros-melodic-desktop_full -y --execution-timeout=0
```

References

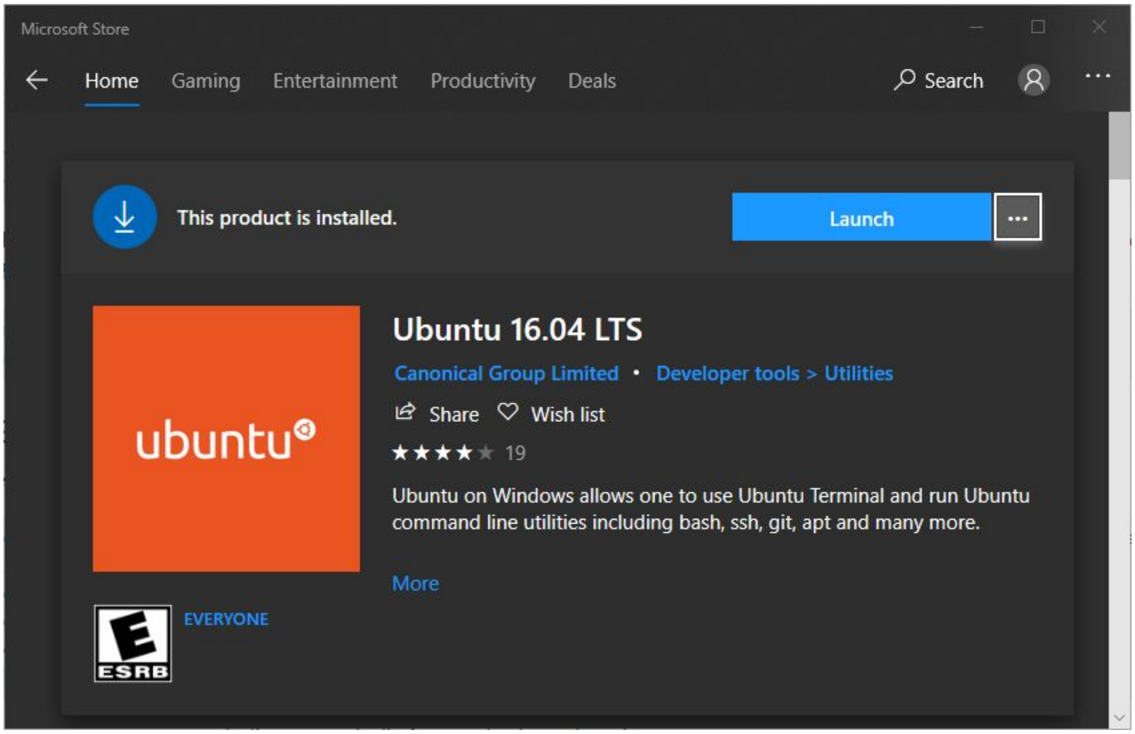
- http://wiki.ros.org/Installation/Windows

Installing ROS on Windows Subsystem for Linux

We can install ROS Kinetic on Windows Subsystem for Linux (WSL). Microsoft Store has Ubuntu 16.04 LTS and ROS Kinetic supports Ubuntu 16.04.

Install Ubuntu 16.04 LTS

1) Open Microsoft Store and install Ubuntu 16.04 LTS



This product is installed.

Ubuntu on Windows allows one to use Ubuntu Terminal and run Ubuntu command line utilities including bash, ssh, git, apt and many more.

2) Second, select Ubuntu 16.04 LTS from Start menu and complete installation. It will ask for a new username and password. Please note that username-password combination for WSL can be different from your Windows login credentials.
3) Use the following commands to update Ubuntu.

```
$ sudo su -
$ apt update
$ apt upgrade
```

Install ROS Kinetic

Run the following commands to install ROS Kinetic

Install repositories

```
$ 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-key adv --keyserver 'hkp://keyserver.ubuntu.com:80' --recv-key
C1CF6E31E6BADE8868B172B4F42ED6FBAB17C654
```

Update package database

```
$ sudo apt update
```

Install ROS packages

```
$ sudo apt install ros-kinetic-desktop-full
```

Initialize rosdep repositories

```
$ sudo rosdep init
$ rosdep update
```

Post-installation

The following command will add ROS initialization script to WSL Ubuntu startup:

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

Also initialize DISPLAY variable to enable VcXsrv on WSL Ubuntu:

```
$ echo "export DISPLAY=:0" >> ~/.bashrc
```

Close WSL window and open it again to initialize ROS Kinetic.

Install Turtlebot on ROS

Run the following command to install Turtlebot packages on WSL:

```
$ apt install ros-kinetic-turtlebot ros-kinetic-turtlebot-apps
ros-kinetic-turtlebot-interactions ros-kinetic-turtlebot-simulator
```

Install X Server for Windows

Download and install **VcXsrv** from https://sourceforge.net/projects/vcxsrv/. Launch VcXsrv from the start menu and **uncheck “Native OpenGL” option** under Extra Settings. You may keep the other options as they are.

The X server will run on the background and accept connections from WSL Ubuntu. As a result, we will be able to run applications that opens a graphical user interface.

Running Turtlebot on WSL

Make sure that VcXsrv is running (check the tray icon next to the clock) and run the following commands:

```
$ export GAZEBO_IP=127.0.0.1
$ LIBGL_ALWAYS_INDIRECT=0 && gazebo
```

If this works, replace `gazebo` with `roslaunch turtlebot_gazebo turtlebot_world.launch` and launch the default Turtlebot simulator.

References

- https://janbernloehr.de/2017/06/10/ros-windows
- https://github.com/microsoft/WSL/issues/3368#issuecomment-414717437

ROS Docker Containers

The following examples assume that you are using Docker Toolbox (VirtualBox) but not Docker for Windows (Hyper-V). The IP addresses might be different for Docker for Windows but the commands will be the same.

Turtlebot Simulation

This Docker image contains ROS Kinetic, Gazebo 7 and Turtlebot packages. The ROS environment will be automatically loaded for all commands run inside the container.

Pull the Docker image

```
$ docker pull scslabisu/turtlebot:sim-default
```

Run the container

```
$ docker run --name gzserver --hostname gzserver -p 11345:11345 -p 11311:11311
-e “ROS_IP=172.17.0.2” -d scslabisu/turtlebot:sim-default
```

The container name will be `tb1` and it will be visible via `docker ps` command. We are also forwarding the ROS port (11311) and the Gazebo port (11345) to the localhost.

Gazebo Web

This Docker image contains ROS Kinetic, Gazebo 7 and GzWeb packages. The ROS environment will be automatically loaded for all commands run inside the container.

Pull Docker image from SCSLab repository

```
$ docker pull scslabisu/gzweb:latest
```

Find the IP address of the Docker container that runs the simulation

Run the following command to list the environmental variables for the Docker container.

```
$ docker inspect gzserver
```

You will see the internal IP address in the output. Let’s assume that the IP address is `172.17.0.2`. We will use this IP address to set `GAZEBO_MASTER_URI` variable in the next step.

Run the container

We will use the IP address that we have found in the previous step to pass the environment variables `GAZEBO_MASTER_URI` to the Docker container.

```
$ docker run --name webclient --hostname webclient -p 8080:8080 -e
“ROS_IP=172.17.0.3” -e “ROS_MASTER_URI=http://172.17.0.2:11311” -e
“GAZEBO_MASTER_URI=http://172.17.0.2:11345” -d scslabisu/gzweb:latest
```

Similarly, you can run the Turtlebot using the following command:

```
$ docker run --name webclient --hostname webclient -p 8080:8080 -e
“ROS_IP=172.17.0.3” -e “ROS_MASTER_URI=http://172.17.0.2:11311” -e
“GAZEBO_MASTER_URI=http://172.17.0.2:11345” -d scslabisu/gzweb:turtlebot
```

Access to Gazebo Web

Docker Toolbox for Windows

Docker Toolbox for Windows comes with VirtualBox. In this setup, you will be able to access Gazebo Web via Docker’s main virtual server (docker-machine) IP address. To find the IP address, run the following command:

```
$ docker-machine env
```

You will see the IP address of the virtual server next to `DOCKER_HOST` variable. For instance, let’s assume that the IP address is `192.168.99.100`. Then, open Google Chrome and go to the address http://192.168.99.100:8080 to access Gazebo Web UI.

Docker Desktop for Windows

Please refer to the official Docker documentation.

Docker Engine for Linux

Docker Engine creates a network device, e.g. docker0. Therefore, it is possible to access the container via its IP address.

Run the following command to list the environmental variables for the Docker container.

```
$ docker inspect webclient
```

You will see the internal IP address in the output. Let’s assume that the IP address is `172.17.0.3`. Then, open Google Chrome and go to the address http://172.17.0.3:8080 to access Gazebo Web UI.

Keyboard Teleoperation

It is possible to control the Turtlebot using keyboard input.

Pull the Docker image

```
$ docker pull scslabisu/turtlebot:teleop
```

Run the container

```
$ docker run --name teleop --hostname teleop -e “ROS_IP=172.17.0.4” -e
“ROS_MASTER_URI=http://172.17.0.2:11311” -it scslabisu/turtlebot:teleop
```

The container name will be `teleop` and it will be visible via `docker ps` command. It will be an interactive container.

References

- https://gist.github.com/ruffsl/4a24c26a1aa2cc733c64

Using Docker Compose

Docker Compose is a tool for defining and running multi-container Docker applications. It uses a YAML file to configure services. Following is an example YAML file to run the Turtlebot stack.

```
version: "3"

services:
    gzserver:
        image: "scslabisu/turtlebot:sim-default"
        ports:
            - "11311:11311"
            - "11345:11345"
        environment:
            - "ROS_IP=172.18.0.2"
    webclient:
        image: "scslabisu/gzweb:turtlebot"
        ports:
            - "8080:8080"
        environment:
            - "ROS_IP=172.18.0.3"
            - "ROS_MASTER_URI=http://gzserver:11311"
            - "GAZEBO_MASTER_URI=http://gzserver:11345"
        depends_on:
            - "gzserver"
    webteleop:
        image: "scslabisu/turtlebot:webteleop"
        environment:
            - "ROS_IP=172.18.0.4"
            - "ROS_MASTER_URI=http://gzserver:11311"
        depends_on:
            - "gzserver"
```

Save this file as `docker-compose.yml` and run it via `docker-compose up` command.

Processing 3-D Models in ROS

Introduction

Install `rospack` first before proceeding with the model processing:

```
$ apt install rospack-tools
```

Then, run the following command to find the directory for Turtlebot 3-D models:

```
$ rospack find turtlebot_description
```

The following examples can be applied to any model, unless some special conditions are specified. We will use Turtlebot for the sake of simplicity.

Converting XACRO to URDF

For Turtlebot, the models are stored in the `turtlebot_description` package. Please refer to the following ROS wiki for more details: http://wiki.ros.org/turtlebot_description

If you have followed the package installation tutorial (i.e. using apt or apt-get), you will find the turtlebot_description package under `/opt/ros/kinetic/share/turtlebot_description`. The recent versions of the converter application `xacro` can be directly called from the command line as follows:

```
$ xacro --inorder {TBD_PATH}/robots/kobuki_hexagons_kinect.urdf.xacro >
kobuki_hexagons_kinect.urdf
```

where `{TBD_PATH}` is `/opt/ros/kinetic/share/turtlebot_description` for ROS Kinetic.

Creating Docker Containers

Building Containers

Run the following command in the directory that contains `Dockerfile`:

```
$ docker build -t {DOCKER_IMAGE_NAME} .
```

where `{DOCKER_IMAGE_NAME}` is the name of the docker image you will be building.

Pushing Containers

Please refer to this Medium.com article for details.

SSH into a Container

Use `docker ps` to find the name of the existing container and then, use the following command:

```
$ docker exec -it {CONTAINER_NAME} /bin/bash
```

`/bin/bash` can be replaced by another command.

Turtlebot

Turtlebot setup runs a WiFi router which allows access under limited connectivity conditions, i.e. no internet. It is recommended that you connect to the WiFi router and then SSH to the Turtlebot's computer.

WiFi and SSH Connection

SSID: TP_Link_3432
Key: 05882878
Turtlebot IP Address: 192.168.0.10

Remote Access

The following usernames and passwords can be used to SSH into the ROS Indigo server (nVidia Jetson TK1).

Non-privileged user: tbuser
Non-privileged pass: tbuser

Privileged user: ubuntu
Privileged pass: ubuntu

Privileged user can run sudo command. Non-privileged cannot.

Running ROS and Components

After the log in, the ROS environment is automatically loaded. There is no need to source the ROS bash script. Instead of the roslaunch commands, you can use the provided shortcuts.

Definition	ROS Command	Shortcut
Load ROS	roslaunch turtlebot_bringup minimal.launch	tb_bringup
Activate teleop	roslaunch turtlebot_teleop keyboard_teleop.launch	tb_kbteleop
Activate camera	roslaunch openni2_launch openni2.launch	tb_camera

Basic ROS Commands

- rostopic list
- rostopic bw `/camera/rgb/image_raw`
- rostopic echo `/camera/depth/points`
- rosrun image_view image_view image:=`/camera/rgb/image_raw`
- ….