

Contents

Lab 1 – Introduction to ROS and Virtual Machines.....	2
Requirements:	2
Installing VMWare Workstation	2
Creating an Ubuntu 18.04 LTS Virtual Machine.....	2
Installing VMWare Tools on Ubuntu.....	11
Installing the Robot Operating System (ROS)	13
Setting Up the Jetson Nano VM Network.....	19
What is SSH?	19
Connecting	19
Working with the Robot Operating System (ROS).....	21
Navigating ROS Commands	21
Creating a Package.....	22
ROS Graph Concepts	25
TurtleSim.....	25
Introducing rqt_graph.....	27
Listing ROS Topics	28

Lab 1 – Introduction to ROS and Virtual Machines

Requirements:

-

Installing VMWare Workstation

<https://youtu.be/6ufEeFnLbxc>

Commented [A1]: Add procedure in case the video is removed

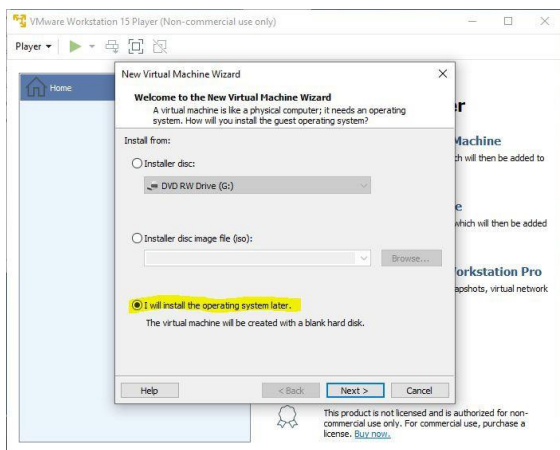
Creating an Ubuntu 18.04 LTS Virtual Machine

For these ROS (Robotic Operating System) labs, a Linux-based Operating System will be required. Since many students will be running Windows or MacOS, you will be instructed on how to create a virtual machine on your host operating system.

Download VMware Workstation (free with McMaster ID, <https://uts.mcmaster.ca/services/computers-printers-and-software/software-licensing/vmware-academic-program/>) and install it on your system. The operating system image (.ISO) of Ubuntu 18.04 LTS is also required and may be downloaded from Canonical's website (<https://releases.ubuntu.com/18.04/>). Although more recent versions of Ubuntu exist, it must be ensured that they are compatible with ROS melodic (<https://www.ros.org/repos/rep-0003.html>). This is because ROS Melodic is currently the latest supported ROS release for the Jetson Nano and both ROS instances must be able to communicate with each other.

Please select the downloaded **Desktop image**.

On VMware Workstation, please select **Create a New Virtual Machine**. Then select "I will install the operating system later."



After you select **Next**, choose the following settings below:

New Virtual Machine Wizard

Select a Guest Operating System
Which operating system will be installed on this virtual machine?

Guest operating system

☐ Microsoft Windows

☒ Linux

☐ Other

Version

Ubuntu 64-bit

Help < Back Next > Cancel

Once you have the options selected, press **Next**. You can be creative with the name of your virtual machine. Please leave the location as the default (do not change it).

New Virtual Machine Wizard

Name the Virtual Machine
What name would you like to use for this virtual machine?

Virtual machine name:

Ubuntu 64-bit 18.04 ROS

Location:

C:\Users\SEPTAdmin\Documents\Virtual Machines\Ubuntu 64-bit

Browse...

< Back Next > Cancel

40 GB of space will be allocated for the Virtual Machine. Anything around 30 GB should be fine. Select "Store virtual disk as a single file".

New Virtual Machine Wizard

Specify Disk Capacity

How large do you want this disk to be?

The virtual machine's hard disk is stored as one or more files on the host computer's physical disk. These file(s) start small and become larger as you add applications, files, and data to your virtual machine.

Maximum disk size (GB): 40

Recommended size for Ubuntu 64-bit: 20 GB

☒ Store virtual disk as a single file

☐ Split virtual disk into multiple files

Splitting the disk makes it easier to move the virtual machine to another computer but may reduce performance with very large disks.

Help

< Back

Next >

Cancel

Afterwards, a confirmation window will open. Click **Finish**.

New Virtual Machine Wizard

Ready to Create Virtual Machine

Click Finish to create the virtual machine. Then you can install Ubuntu 64-bit.

The virtual machine will be created with the following settings:

Name: Ubuntu 64-bit 18.04 ROS

Location: C:\Users\SEPTAdmin\Documents\Virtual Machines\Ub...

Version: Workstation 15.x

Operating System: Ubuntu 64-bit

Hard Disk: 40 GB

Memory: 4096 MB

Network Adapter: NAT

Other Devices: 2 CPU cores, CD/DVD, USB Controller, Printer, Sound...

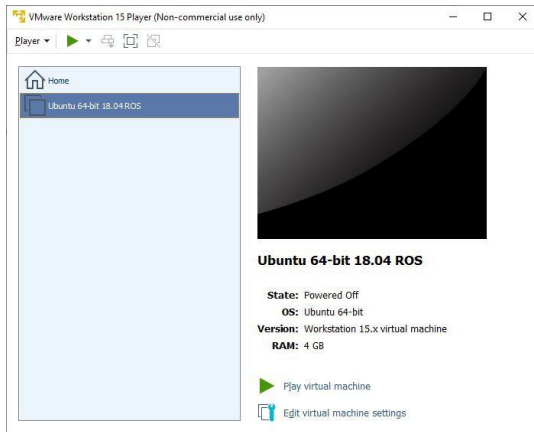
Customize Hardware...

< Back

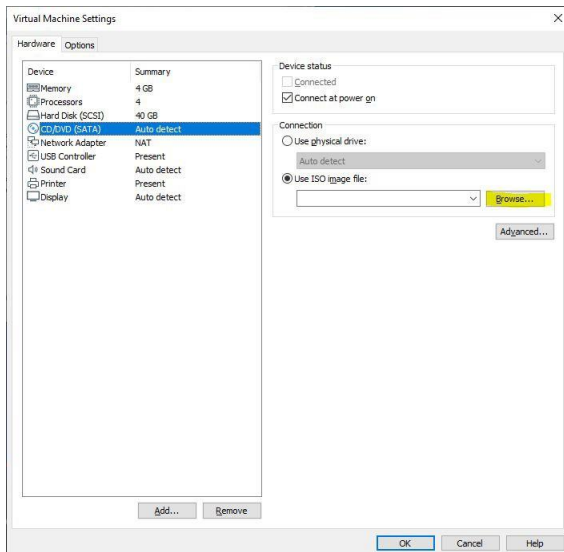
Finish

Cancel

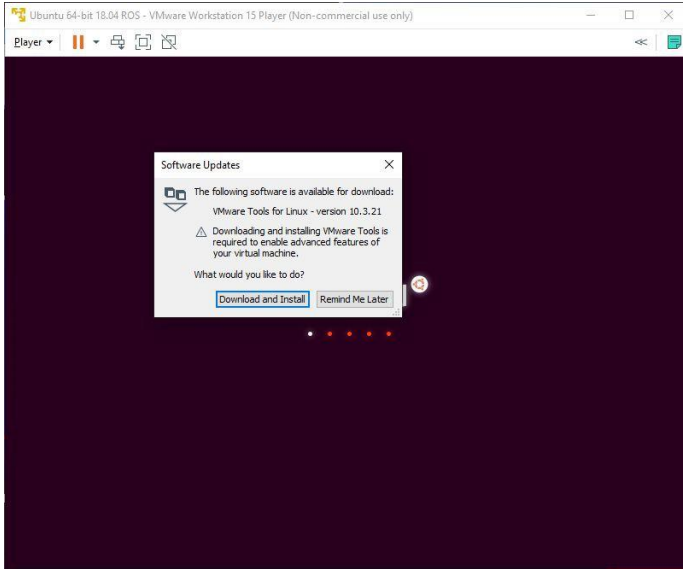
Next, notice that the virtual machine has been added.



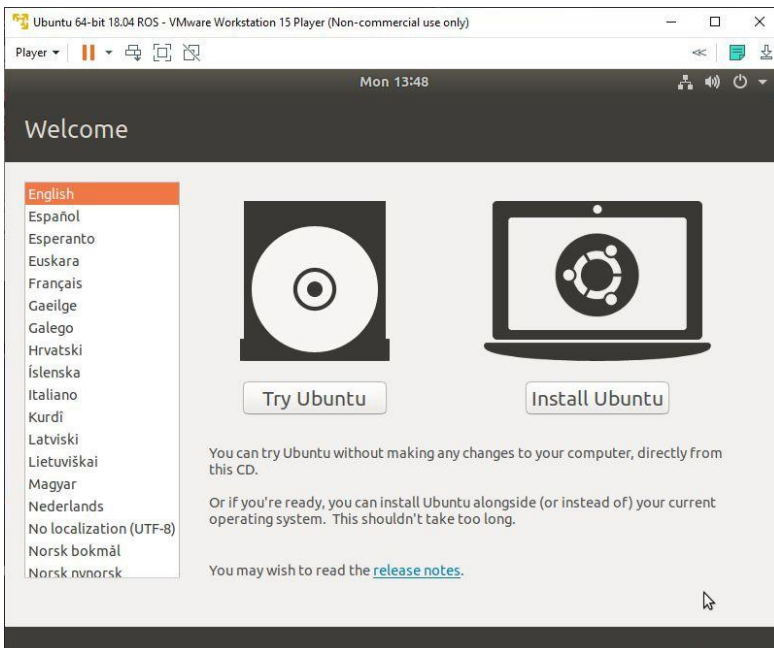
But before it can be run, it is important that you **Edit the virtual machine settings**.



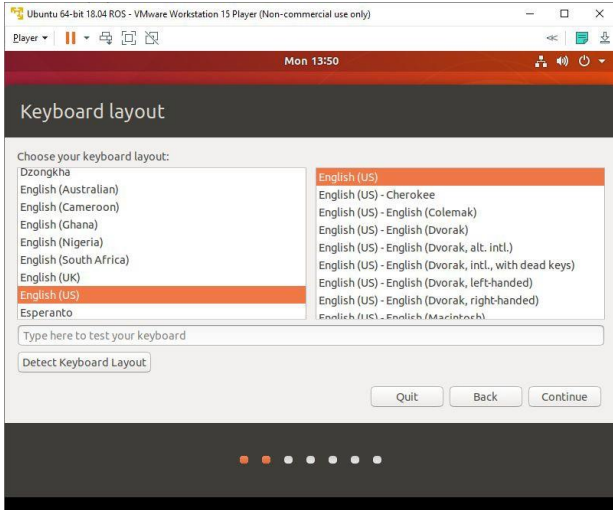
Modify the **Memory** and **Processors** allocated. Please choose reasonable amounts as your system will be different. A rule of thumb is to allocate less than half of your base system's CPU threads and RAM. In the **CD/DVD** setting, please **Browse** for the .ISO image we downloaded earlier. Now, begin booting the Virtual Machine. This is done by pressing **Play virtual machine**. The following window should pop up:



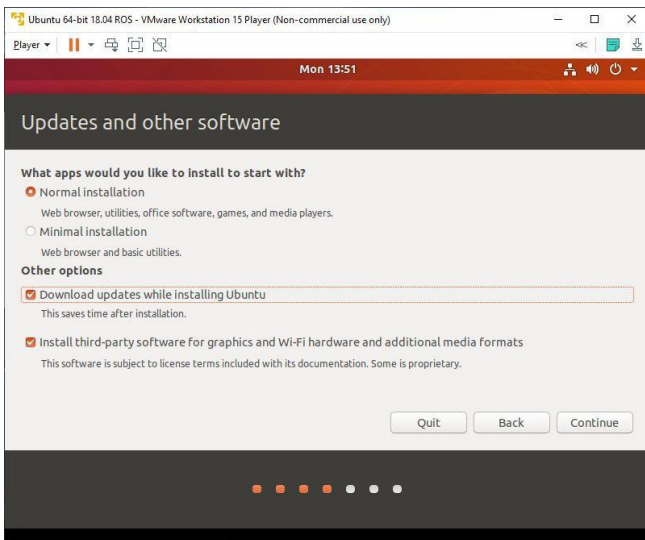
If prompted, install the latest version of VMWare Tools. Proceed by pressing **Download and Install**. Once updated, the boot process will continue and the following Welcome screen will be presented to you in VMWare Workstation Player.



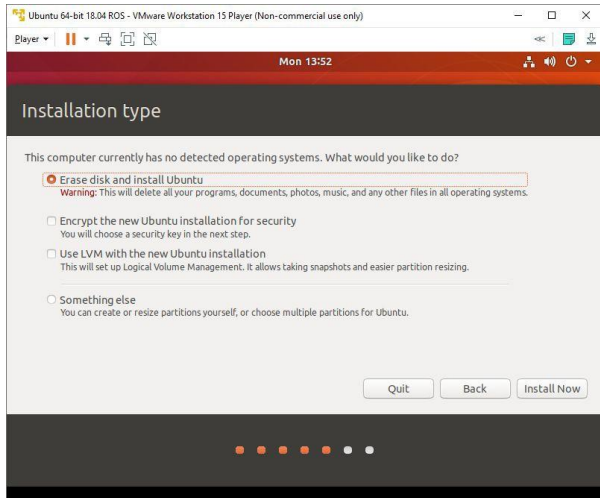
Select **Install Ubuntu** and follow along the screen options. The install process is very intuitive.



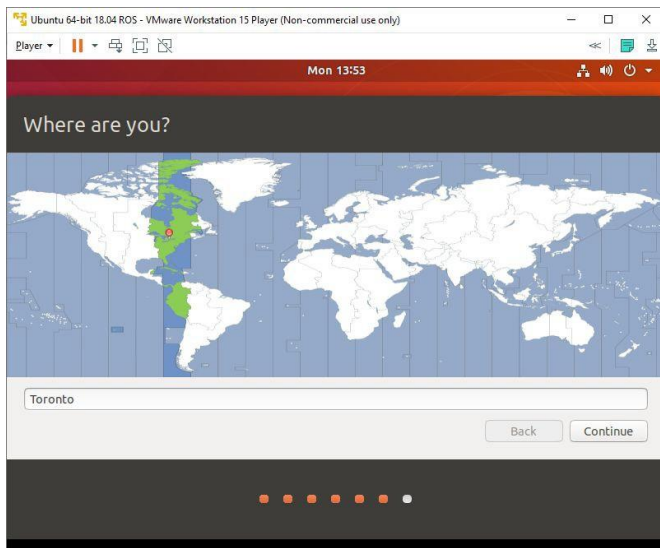
Make sure to tick the bottom option "**Install third-party software for graphics and Wi-Fi hardware and additional media formats**".



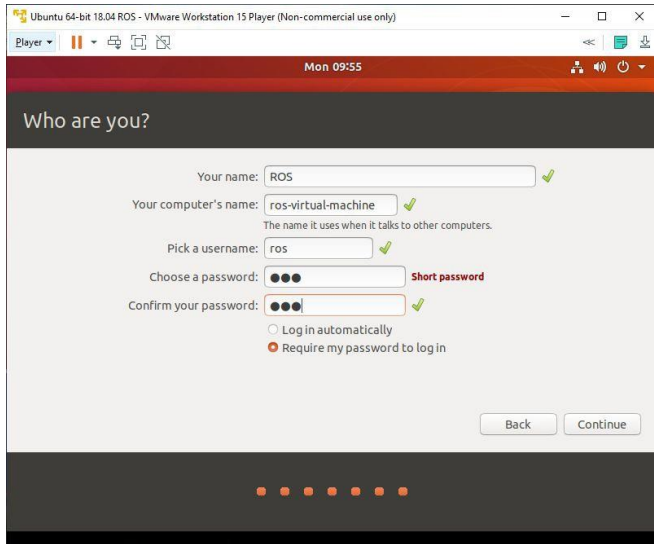
Since this is a virtual machine, it is isolated from your host operating system and the data on your computer. It is safe to 'Erase disk and install'. Leave as default and **Install Now**. Another Window will popup, press **Continue**.



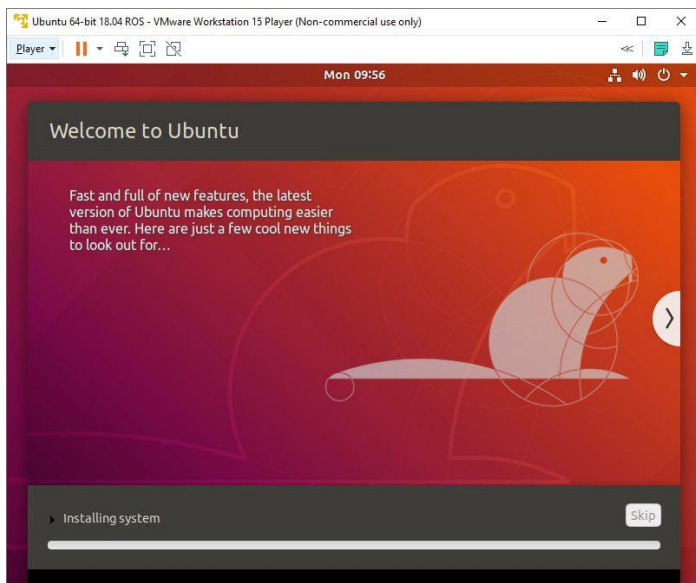
Choose your time zone. It would be your nearest major city. In our case, this would be Toronto, Ontario, Canada.



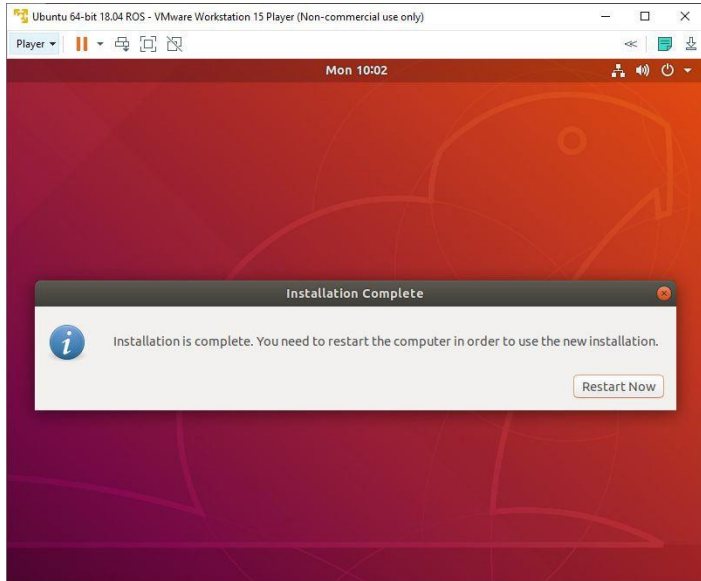
And finally, configure your Ubuntu credentials. A secure password is recommended. Press **Continue** when finished.



Finally, Ubuntu 18.04 LTS will begin installing on your virtual machine.

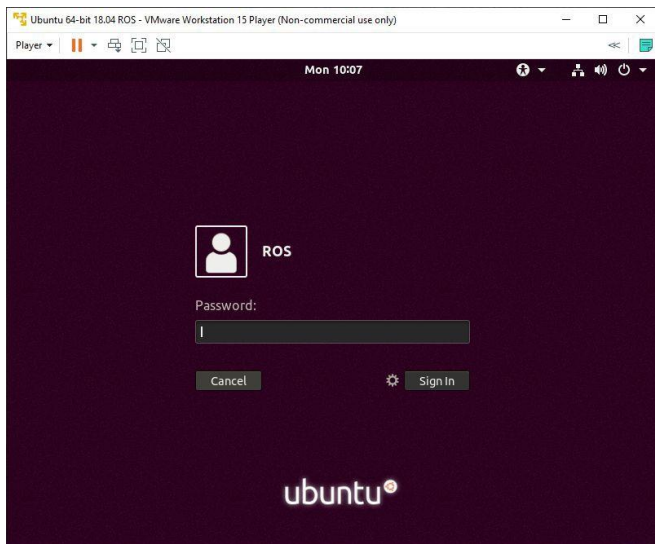


Once the installation is complete, please restart the VM.



Press **Restart Now**. A terminal window may appear with text running down your screen. Do not fret, this is normal. If you appear to be stuck within the VM and you cannot find your mouse. Press **Ctrl + Alt** to bring back your mouse.

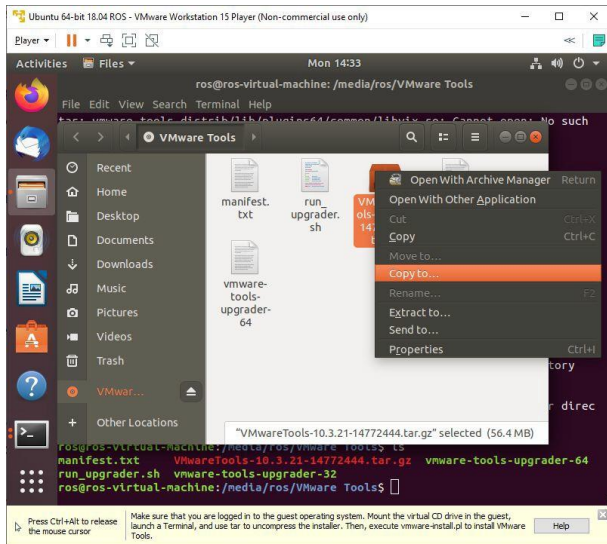
Power Off the VM by pressing the exit button. After it has successfully powered off, please reboot the VM.



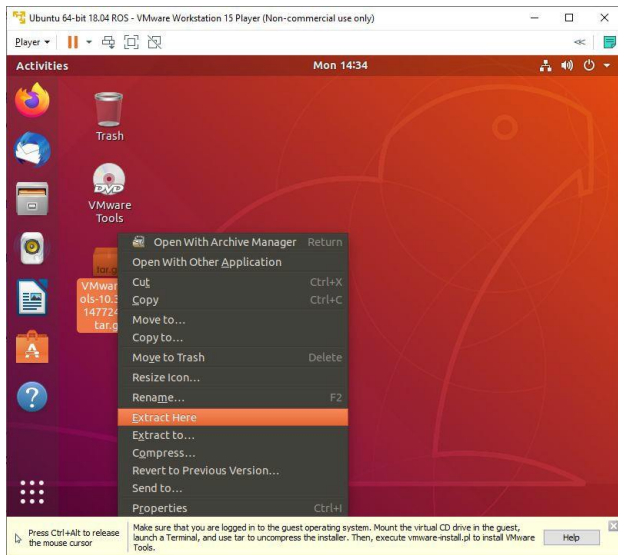
Log in and go through the first-time starting messages. Afterwards, you will be ready to work with Ubuntu!

Installing VMWare Tools on Ubuntu

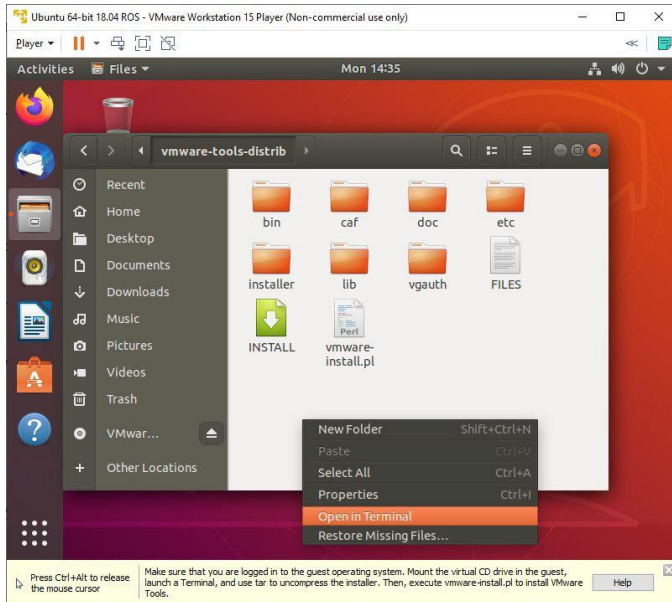
VMware Tools is an add-on that will improve the ease of control over basic functions such as copying and pasting. With VMware open, click the top left **Player** > **Manage** > **Install VMware Tools**. After a short while, a directory will appear. Please double click on the directory. Copy this directory to your desktop.



On your desktop, right click the **VMwareTools.x.x.x-xxxx.tar.gz** file and extract it.



Go into the extracted **vmware-tools-distrib** folder. Next, **open** the folder in **Terminal**.



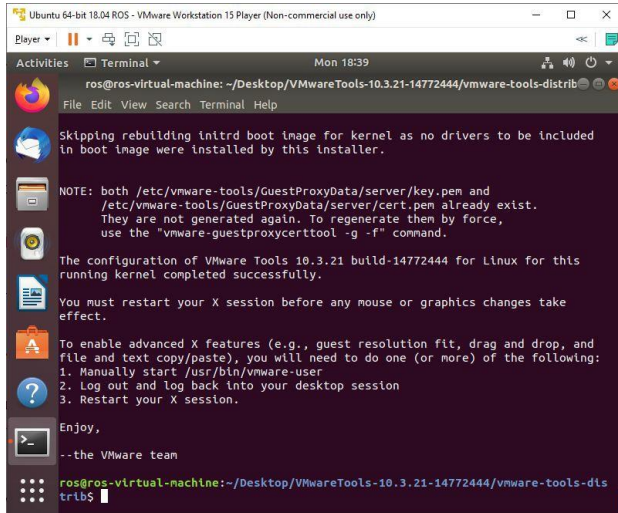
Once the terminal has appeared, run the following terminal commands:

```
sudo chmod u+x vmware-install.pl
```

chmod u+x will change the access permissions of the perl install script to be executable. The following command will execute that script.

```
sudo ./vmware-install.pl -d
```

Afterwards, you should see the following:



```
ros@ros-virtual-machine: ~/Desktop/VMwareTools-10.3.21-14772444/vmware-tools-distrib
File Edit View Search Terminal Help

Skipping rebuilding initrd boot image for kernel as no drivers to be included
in boot image were installed by this installer.

NOTE: both /etc/vmware-tools/GuestProxyData/server/key.pem and
/etc/vmware-tools/GuestProxyData/server/cert.pem already exist.
They are not generated again. To regenerate them by force,
use the "vmware-guestproxycerttool -g -f" command.

The configuration of VMware Tools 10.3.21 build-14772444 for Linux for this
running kernel completed successfully.

You must restart your X session before any mouse or graphics changes take
effect.

To enable advanced X features (e.g., guest resolution fit, drag and drop, and
file and text copy/paste), you will need to do one (or more) of the following:
1. Manually start /usr/bin/vmware-user
2. Log out and log back into your desktop session
3. Restart your X session.

Enjoy,

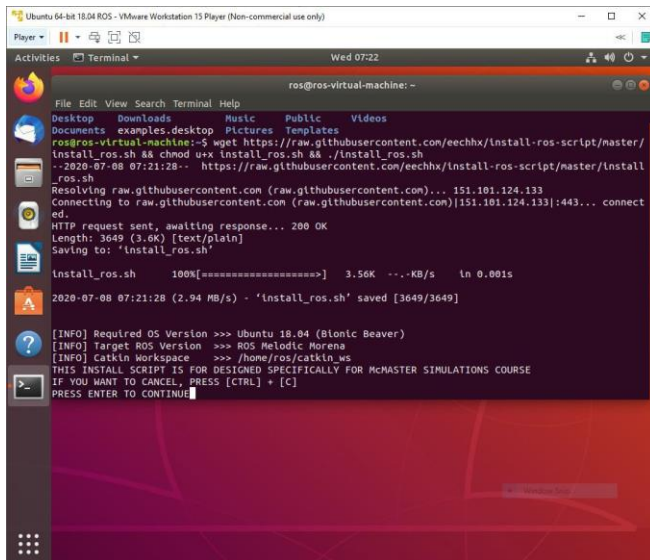
--the VMware team

ros@ros-virtual-machine:~/Desktop/VMwareTools-10.3.21-14772444/vmware-tools-dis
trib$
```

Installing the Robot Operating System (ROS)

To streamline the install process, an install script has been created for ROS that will download ROS and other dependencies that are required. It will also setup our ROS environment and catkin workspace.

Open **Terminal** by pressing the "Home" or "Windows" key and searching for it. You may alternatively use **CTRL + ALT + T**.



```
ros@ros-virtual-machine: ~
File Edit View Search Terminal Help
Desktop Downloads Basic Public Videos
Documents examples.desktop Pictures Templates
ros@ros-virtual-machine:~$ wget https://raw.githubusercontent.com/eechhx/install-ros-script/master/
install_ros.sh && chmod u+x install_ros.sh && ./install_ros.sh
--2020-07-08 07:21:28-- https://raw.githubusercontent.com/eechhx/install-ros-script/master/install
_ros.sh
Resolving raw.githubusercontent.com (raw.githubusercontent.com)... 151.101.124.133
Connecting to raw.githubusercontent.com (raw.githubusercontent.com)[151.101.124.133]:443... connect
ed.
HTTP request sent, awaiting response... 200 OK
Length: 3649 (3.6K) [text/plain]
Saving to: 'install_ros.sh'

install_ros.sh 100%[=====] 3.56K --.-KB/s in 0.001s

2020-07-08 07:21:28 (2.94 MB/s) - 'install_ros.sh' saved [3649/3649]

[INFO] Required OS Version >>> Ubuntu 18.04 (Bionic Beaver)
[INFO] Target ROS Version >>> ROS Melodic Morenia
[INFO] Catkin Workspace >>> /home/ros/catkin_ws
THIS INSTALL SCRIPT IS FOR DESIGNED SPECIFICALLY FOR McMASTER SIMULATIONS COURSE
IF YOU WANT TO CANCEL, PRESS [CTRL] + [C]
PRESS ENTER TO CONTINUE
```

To download the script, use the following command.

wget https://raw.githubusercontent.com/eechhx/install-ros-script/master/install_ros_jetson.sh

Once downloaded, permissions must be modified to allow for executing.

```
chmod u+x install_ros_jetson.sh
```

Next, run the script using the following command.

```
./install_ros_jetson.sh
```

Ensure that the script runs to completion without any errors. Errors may prevent the build process of your catkin workspace and dependencies. Below, is the script in case that the source is no longer accessible, or you would like to use it for troubleshooting purposes. This was pasted on June 25th, 2021.

```
#!/bin/bash

#Author: Eech Hsiao

echo ""

echo "[INFO] Required OS Version >>> Ubuntu 18.04 (Bionic Beaver)"

echo "[INFO] Target ROS Version >>> ROS Melodic Morena"

echo "[INFO] Catkin Workspace >>> $HOME/catkin_ws"

echo "THIS INSTALL SCRIPT IS FOR DESIGNED SPECIFICALLY FOR McMASTER MACBOT JETSON NANO"

echo "IF YOU WANT TO CANCEL, PRESS [CTRL] + [C]"

read -p "PRESS ENTER TO CONTINUE"

ROS_DISTRO=${ROS_DISTRO:="melodic"}

name_catkin_workspace=${name_catkin_workspace:="catkin_ws"}

echo "[INFO] INSTALLING ROS"

echo "[INFO] Written for MacBot // Ubuntu 18.04 // ROS Melodic Morena"

echo "[INFO] Adding ROS Repository"

if [ ! -e /etc/apt/sources.list.d/ros-latest.list ]; then

    sudo sh -c 'echo "deb http://packages.ros.org/ros/ubuntu $(lsb_release -sc) main" > /etc/apt/sources.list.d/ros-latest.list'

fi

echo "[INFO] Downloading ROS Keys"

sudo apt-key adv --keyserver 'hkp://keyserver.ubuntu.com:80' --recv-key
C1CF6E31E6BADE8868B172B4F42ED6FBAB17C654

echo "[INFO] Downloading Intel RealSense Keys"

sudo apt-key adv --keyserver keys.gnupg.net --recv-key F6E65AC044F831AC80A06380C8B3A55A6F3EFCDE || sudo apt-key
adv --keyserver hkp://keyserver.ubuntu.com:80 --recv-key F6E65AC044F831AC80A06380C8B3A55A6F3EFCDE

echo "[INFO] Adding Intel RealSense Repository"

if [ ! -e /etc/apt/sources.list.d/danielrichter2007-ubuntu-grub-customizer-bionic.list ]; then

    sudo add-apt-repository "deb http://realsense-hw-public.s3.amazonaws.com/Debian/apt-repo bionic main" -u

fi
```

```
echo "[INFO] Installing ROS and Other Packages"

sudo apt install -y \
    python-rosdep \
    apt-utils \
    librealsense2-utils \
    librealsense2-dev \
    ros-$ROS_DISTRO-realsense2-camera \
    ros-$ROS_DISTRO-teleop-twist-keyboard \
    ros-$ROS_DISTRO-ros-controllers \
    ros-$ROS_DISTRO-gmapping \
    ros-$ROS_DISTRO-navigation \
    ros-$ROS_DISTRO-urdf \
    ros-$ROS_DISTRO-gazebo-ros \
    ros-$ROS_DISTRO-xacro \
    ros-$ROS_DISTRO-geometry \
    ros-$ROS_DISTRO-geometry-msgs \
    ros-$ROS_DISTRO-sensor-msgs \
    ros-$ROS_DISTRO-image-transport \
    ros-$ROS_DISTRO-dynamic-reconfigure \
    ros-$ROS_DISTRO-diagnostic-updater \
    ros-$ROS_DISTRO-nodelet-core \
    ros-$ROS_DISTRO-camera-info-manager \
    ros-$ROS_DISTRO-perception-pcl \
    ros-$ROS_DISTRO-csm \
    ros-$ROS_DISTRO-ros-base

# Initialize and Update rosdep
if [ ! -e /etc/ros/rosdep/sources.list.d/20-default.list ]; then
    sudo rosdep init
fi
rosdep update

echo "[INFO] Environment Setup"
```



```
source /opt/ros/$ROS_DISTRO/setup.sh

printf "\nsource /opt/ros/$ROS_DISTRO/setup.bash" >> $HOME/.bashrc

echo "[INFO] Create Catkin Workspace"
mkdir -p $HOME/$name_catkin_workspace/src
cd $HOME/$name_catkin_workspace/src
catkin_init_workspace
cd $HOME/$name_catkin_workspace
catkin_make

echo "[INFO] Sourcing catkin_workspace"
printf "\nsource ~/ $name_catkin_workspace/devel/setup.bash" >> $HOME/.bashrc

echo "[INFO] Fixing Gazebo REST Request Error"
sed -i -e 's,https://api.ignitionfuel.org,https://api.ignitionrobotics.org,g' ~/.ignition/fuel/config.yaml

echo "[INFO] Git clone RealSense Gazebo / ROS"
cd $HOME/$name_catkin_workspace/src
wget https://raw.githubusercontent.com/eechhx/install-ros-script/master/_d435.gazebo.xacro
wget https://raw.githubusercontent.com/eechhx/install-ros-script/master/_d435.urdf.xacro
git clone https://github.com/pal-robotics/realsense_gazebo_plugin.git
git clone https://github.com/IntelRealSense/realsense-ros.git

echo "[INFO] Moving and Replacing d435 Files"
mv -f $HOME/$name_catkin_workspace/src/_d435.urdf.xacro $HOME/$name_catkin_workspace/src/realsense-ros/realsense2_description/urdf
mv -f $HOME/$name_catkin_workspace/src/_d435.gazebo.xacro $HOME/$name_catkin_workspace/src/realsense-ros/realsense2_description/urdf

echo "[INFO] Git clone diff_drive ROS package"
git clone https://github.com/merose/diff_drive.git

echo "[INFO] Update and Upgrade Packages"
sudo apt-get update -y
sudo apt-get dist-upgrade -y
```

```
echo "[INFO] Jetson Nano uses OpenCV4. Temp solution of downgrading to OpenCV3 for compatability with  
vision_opencv/cv_bridge [melodic-branch]"
```

```
sudo apt -y --allow-downgrades install libopencv-dev=3.2.0+dfsg-4ubuntu0.1
```

```
sudo apt-mark hold libopencv-dev
```

```
echo "[INFO] Finished Full Installation"
```

```
exit 0
```

Setting Up the Jetson Nano VM Network

What is SSH?

SSH (**Secure SHell**) is a Network Protocol that allows one device (virtual machine) to connect to another device (MacBot) over a securely encrypted TCP/IP connection. This will be extremely useful for this set of labs where there is a need to remotely connect and interface with the MacBot. As long as the MacBot is connected to the same network as your PC, you will be able to access and modify the files through SSH.

Before SSHing into the Jetson Nano on the MacBot, the IP address of the device needs to be known. Connect a monitor to the Jetson Nano on the MacBot using HDMI and proceed to powering on and booting the MacBot's Jetson Nano. Open the **Terminal** and run the following command:

sudo ifconfig

sudo stands for 'superuser do' and temporarily elevates the privileges of that command. It is the equivalent to 'run as administrator' on Windows.

ifconfig is a UNIX command that returns information regarding network interfaces that are being used. There are three interfaces that you are likely to see:

- eth0 - First ethernet interface. Any other would be designated eth1, eth2, etc.
- lo - Loopback interface. A special network interface that the OS uses to communicate with itself.
- wlan0 - First wireless network on the system. Subsequent interfaces designated as wlan1, wlan2, etc.

It is important that both your computer and your MacBot are connected to the **same** network. To elaborate, if your computer is using an ethernet connection, your MacBot must also be connected to the same ethernet network. The same applies when using WiFi.

This lab will proceed with using Ethernet for now. To do so, the eth0 address must be noted down:

inet addr:192.168.1.9

This is the IP address of the MacBot's Jetson Nano and will be the address that is used to connect remotely via SSH.

Note: Your address will likely differ from this one.

Note: IP addresses are typically assigned dynamically. This means that there is no guarantee that if the MacBot disconnects and reconnects that it will keep the same IP address. If you have admin login credentials to the network, you may be able to assign a static IP address that does not change. Otherwise, just keep this in mind.

Connecting

Open the **Terminal** on the Ubuntu 18.04 LTS virtual machine.

To test if your VM is able to communicate to the MacBot's Jetson Nano over the Ethernet network, use the ping command. The response time is echoed to the terminal window.

ping 192.168.1.9

```
ros@ros-virtual-machine: ~  
File Edit View Search Terminal Help  
ros@ros-virtual-machine:~$ ping 192.168.1.9  
PING 192.168.1.9 (192.168.1.9) 56(84) bytes of data.  
64 bytes from 192.168.1.9: icmp_seq=1 ttl=64 time=1.98 ms  
64 bytes from 192.168.1.9: icmp_seq=2 ttl=64 time=1.03 ms  
64 bytes from 192.168.1.9: icmp_seq=3 ttl=64 time=1.15 ms  
64 bytes from 192.168.1.9: icmp_seq=4 ttl=64 time=0.688 ms  
64 bytes from 192.168.1.9: icmp_seq=5 ttl=64 time=1.18 ms  
64 bytes from 192.168.1.9: icmp_seq=6 ttl=64 time=0.579 ms  
^C  
--- 192.168.1.9 ping statistics ---  
6 packets transmitted, 6 received, 0% packet loss, time 5014ms  
rtt min/avg/max/mdev = 0.579/1.104/1.984/0.454 ms  
ros@ros-virtual-machine:~$
```

As you can see, the VM is receiving a response of 64 bytes from the MacBot with the IP 192.168.1.9. Now, begin to SSH into the MacBot.

Format: #ssh username@ip_address

ssh macbot@192.168.1.9

```
macbot@macbot-desktop: ~  
File Edit View Search Terminal Help  
ros@ros-virtual-machine:~$ ssh macbot@192.168.1.9  
macbot@192.168.1.9's password:  
Welcome to Ubuntu 18.04.4 LTS (GNU/Linux 4.9.140-tegra aarch64)  
  
* Documentation:  https://help.ubuntu.com  
* Management:    https://landscape.canonical.com  
* Support:       https://ubuntu.com/advantage  
  
* Are you ready for Kubernetes 1.19? It's nearly here! Try RC3 with  
  sudo snap install microk8s --channel=1.19/candidate --classic  
  https://microk8s.io/ has docs and details.  
This system has been minimized by removing packages and content that are  
not required on a system that users do not log into.  
  
To restore this content, you can run the 'unminimize' command.  
  
310 packages can be updated.  
28 updates are security updates.  
  
Last login: Fri Aug 14 12:30:35 2020 from 192.168.1.12  
'RANDR Query Version' returned error -1  
Initialization of randr failed.  
Trying next method...  
X request failed: XOpenDisplay  
Initialization of vidmode failed.  
Trying next method...  
No more methods to try.  
macbot@macbot-desktop:~$
```

After entering your login password to the MacBot, you will notice that the terminal window is now connected to the Jetson. You can now access, navigate and modify the Jetson's filesystem. But more importantly, you are able to run certain nodes without taxing the Jetson. Visualizing large amounts of data on the Jetson will produce significant performance issues. Doing this allows for the streaming of sensor data from the Jetson Nano to the VM. This data can further be visualized in the VM. This will be explored further in the upcoming labs.

Working with the Robot Operating System (ROS)

<https://youtu.be/FCsMMIYI9VA>

ROS(Robot Operating System) will be used in these labs. The liberty has already been taken in the previous section to install ROS and its required packages. There are many different distributions of ROS available, but this series of labs will be using Melodic Morena on Ubuntu version 18.04 LTS. Melodic Morena has an End-Of-Line (EOL) support date in 2023. This means that after 2023, transitioning to a newer version of ROS must be strongly considered since Melodic will no longer be receiving updates and security patches.

The “Robot Operating System” is not actually an operating system, but rather middleware that helps integrate the operating system and commonly used dependencies for the development of robotic systems. Note that there are many other different kinds of robotic platforms, but ROS is a popular open-source option. In being open-source, ROS it is constantly being improved and tested by the community and industry alike. It allows users establish an interface between hardware and software platforms, allowing for seamless integration of many distributed systems. A popular application of ROS in industry is in the rapid prototyping of complex systems.

ROS is packaged with many development tools such as rqt, RViz, and Gazebo. These tools allow for the visualization of the flow of information within ROS and help bring your code to life. We will be learning more about these tools as we progress through this series of labs.

Throughout this series of labs, the focus will not be about programming. Instead, we will focus in integrating and familiarizing ourselves with the ROS framework. We will assume that you have basic knowledge about Linux and the terminal.

Navigating ROS Commands

It is important to first learn how to navigate the ROS filesystem to access packages. What is a package? Packages are units of ROS code which have their own libraries, executables, etc. Being able to traverse in and out of these packages to access the files within is important for working with ROS. The focus of this section will be on covering the commonly used commands that allow the user to interact with ROS and ROS Packages.

- `roscd` : Allows the user to change directory to a package. Note that this also works for any subdirectories within the package!
 - `roscd macbot_gazebo`
 - `roscd macbot_navigation/launch`
- `rosls` : Lists the currently installed package. Also lists the subdirectories within an installed folder. This is useful for checking directories and files within a package without actually navigating to the package.

- **rosls < TAB Twice>**
- **rosls gazebo**

Creating a Package

The ROS Workspace has already been configured through the previous ROS install script. This workspace is the directory where ROS packages will be compiled and executed from. It is actually possible have multiple catkin workspaces on one system, but this gets very complicated if you do not configure your workspaces correctly!

To create a package, **catkin** build tool will be used. It is the official build system for recent distributions of ROS. You may run into forum posts that speak about `roscpp` or `ament_cmake`. Please stay away from these and exclusively use `catkin` for the time being. Because various ROS packages may use different programming languages, tools, etc., ROS has their own custom build system. To understand the full scope of the `catkin` build system is beyond the scope of this series of labs. Instead, the focus will be put on learning how to utilize it.

First, before proceeding, the topic of sourcing the ROS environment should be touched upon. In order for ROS to work inside of your terminal, the terminal needs to know where ROS is installed, which installed version you are using, and where your `catkin` workspace is located in the Ubuntu filesystem. This is achieved through 2 commands that need to be entered every single time a new terminal window opens that you intend to use for ROS.

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

```
source ~/catkin_ws/devel/setup.bash
```

This task can be automated by adding the set of commands into the `~/.bashrc` file. This is the terminal window configuration file, and it runs whenever a terminal window opens. Be careful making changes. It must be edited with `sudo` privileges.

```
sudo nano ~/.bashrc
```

Please scroll all the way down to an open line at the bottom.

Type the following commands:

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

```
source ~/catkin_ws/devel/setup.bash
```

```
echo "ROS Sourced!"
```

Save the modified configuration file by pressing `ctrl-s` then `ctrl-x`.

Now, attempt to open a new terminal window and see that it prints `"ROS Sourced!"` on the first line. If there is an error, double-check the ROS installation paths, and the name and location of your `catkin` workspace.

Catkin packages are structured like this:

- Catkin_package
 - CMakeLists.txt - Describes how to build the code and where to install
 - package.xml - Provides information about the package

Navigate to the source directory within the catkin workspace:

```
cd ~/catkin_ws/src
```

The `cd` stands for “Change Directory”. The `~` represents the home directory. The forward slashes represent navigating through the hierarchy of the system directory.

The goal now is to create a custom ROS package. Verify that you are still positioned in the `~/catkin_ws/src` directory.

Please execute the command below. Dependencies are external libraries and tools that the package may sometimes need in order to function properly. In this case, 3 dependencies are required for this package to function correctly.

Format: `catkin_create_pkg <package_name> [dependency 1] [dependency 2] [dependency 3]`

```
catkin_create_pkg mypackage std_msgs roscpp rospy
```

Now that the ROS package has been created, navigate back to the root of the catkin workspace. This is at `~/catkin_ws`. Please run `catkin_make` to rebuild the workspace.

```
catkin_make
```

Alternatively, the `'cd ..'` command could also be used to quickly navigate up a directory.

It will look something like this:

```
administrator@ROS: ~/catkin_ws
File Edit View Search Terminal Help
-- Using CATKIN_ENABLE_TESTING: ON
-- Call enable_testing()
-- Using CATKIN_TEST_RESULTS_DIR: /home/administrator/catkin_ws/build/test_results
-- Found gtest sources under '/usr/src/gtest': gtests will be built
-- Found gmock sources under '/usr/src/gtest': gmock will be built
-- Found PythonInterp: /usr/bin/python2 (found version "2.7.17")
-- Using Python nosetests: /usr/bin/nosetests-2.7
-- catkin 0.7.23
-- BUILD_SHARED_LIBS is on
-- BUILD_SHARED_LIBS is on
-- ~~~ traversing 1 packages in topological order:
-- ~~~ - mypackage
-- ~~~ ~~~~~
-- +++ processing catkin package: 'mypackage'
-- ==> add_subdirectory(mypackage)
-- Configuring done
-- Generating done
-- Build files have been written to: /home/administrator/catkin_ws/build
####
#### Running command: "make -j12 -l12" in "/home/administrator/catkin_ws/build"
####
administrator@ROS:~/catkin_ws$
```


The **rospack** command can be used to see what dependencies are associated with a particular package.

rospack depends mypackage

This will show the following packages: roscpp, rospy, std_msgs. All of which were included as a dependencies when creating the custom ROS package. Dependencies themselves will have their own dependencies, so feel free to check what those are and familiarize yourselves.

Navigate inside the mypackage directory and locate package.xml. Open this file in nano text editor.

nano package.xml

This series of labs will not cover the specifics so for further reading please visit the following website:

<http://wiki.ros.org/catkin/package.xml>

ROS Graph Concepts

This section will cover ROS nodes, topics, and messages.

Nodes can be thought of as processes. They communicate with each other through topics which can be messaged to and read from to exchange information.

Topics can be thought of as messaging channels that are used to exchange information between ROS nodes. The user interacts with topics through publish and subscribe routines.

Messages can be expressed through a wide variety of types. One example is std_msgs::Twist which publishes velocity in JSON format. These messages are published to topics and accessed by the nodes that are subscribed to that particular topic.

An example of these three concepts working together could be a turtlebot. A package called teleop_twist_keyboard may publish std_msgs::Twist messages over the topic /cmd_vel. The differential drive ROS node will be subscribed to this topic and asynchronously actuate the motors based off of this data.

TurtleSim

TurtleSim is a ROS package that comes preinstalled with ROS. It is a great introductory learning tool that explores packages, nodes, and the communication between nodes.

ROS core will be ran frequently. This command starts the ROS master node, which tracks all of the ROS communication such as publishers and subscribers to the various active topics. These nodes will notify the master that it will publish or subscribe to various topics. Since it oversees the whole communication structure of ROS, it must always be running in order for us to be able to test or use ROS functionalities. Also, note that only one ROS core node should be active at one time. Multi-master networking is not covered in this series of labs.

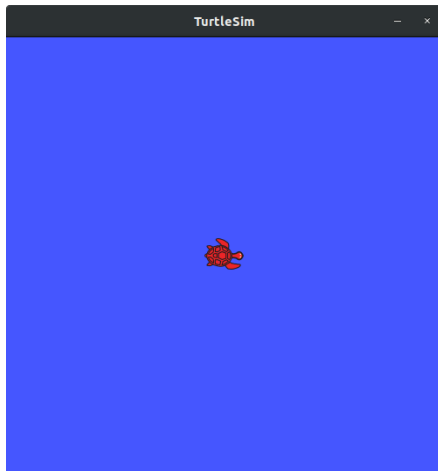
roscore

The command `roslaunch` allows one to run nodes from packages directly. **Open a new tab** (Ctrl + Shift + T) by pressing the top right new tab button and run the following commands to run the TurtleSim node:

Format: `roslaunch [package] [node]`

`roslaunch turtlesim turtlesim_node`

Notice that the following window will open. The turtle will randomly generate, so do not fret if your turtle does not look like the one below.



Run the following command in a new sourced terminal window.

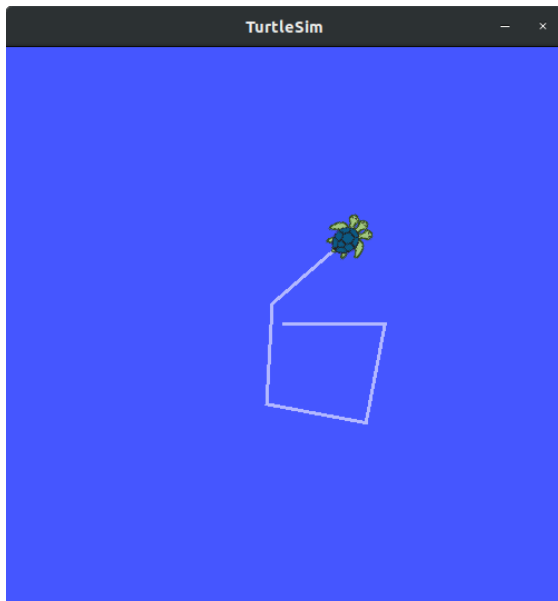
`roslaunch` list

This command prints a list of the currently running ROS nodes. It is used frequently when troubleshooting or developing ROS systems. Next, the `turtle_teleop_key` ROS node must be launched.

`roslaunch turtlesim turtle_teleop_key`

This node allows for the user to control the position and movement of the turtle in the window. This is done in the terminal emulator window that the teleop node is launched from, using the arrow keys on the keyboard.

To exit the program, use **Ctrl + C** to interrupt cancel `roscore` or any of the ROS nodes that are active.

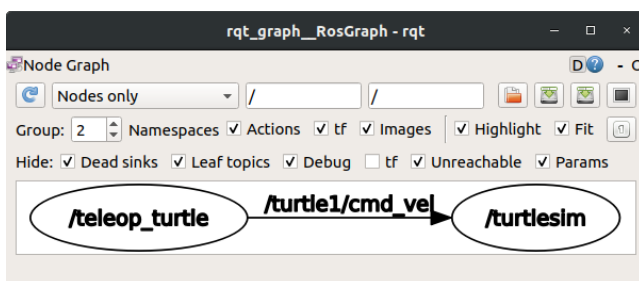


Introducing rqt_graph

There are many tools within ROS that allow for the visualization of data and communication between nodes. Rqt_graph is one of these tools. It breaks what is happening in the current ROS environment down into an easily readable chart.

roslaunch rqt_graph rqt_graph

The following window should appear:



This visualizes both the teleop_turtle and turtlesim ROS nodes. Turtlesim is subscribing to the '/turtle1/cmd_vel' topic which the teleop_turtle node is publishing to.

Rqt_graph is one of those tools that ROS developers find themselves using frequently in the troubleshooting process.

[Listing ROS Topics](#)

The rostopic tool allows the ROS developer to gain more insight into active topics.

rostopic bw # Displays bandwidth used by a topic

rostopic echo # Print messages to screen

rostopic hz # Display publishing rate of topic

rostopic list # Prints information about active topics

rostopic pub # Publish data to topics

rostopic type # Prints topic type

Once one begins implementing and working with a real physical platform with sensors and actuators, then these tools will become very helpful. However, in particular, the commands **rostopic list** and **rostopic echo** are the two most frequently used ones.

