Installation of CRUMB (GAZEBO) - ROS kinetic on Ubuntu 16.04

Autor: Benjamín Vega Herrera

February 11, 2020

Contents

1 Prerequisites

The installation has been installed on Ubuntu 16.04.6 LTS (Xenial Xerus) with the last updates at *January, 20 2020*. Installing ROS kinetic needs the kernel 4.4.0-x-generic, in order to know the kernel version that Ubuntu has installed we can use the command *uname -a*. If the kernel version is greater than 4.4.0.x we have to downgrade the kernel, for this, we use the follow commands to install the new kernel:

- *sudo apt install linux-image-generic-4.4.0-x-generic*
- *sudo apt install linux-headers-4.4.0-x-generic*. In this case, it has been installed the version 4.4.0-171.

To install Grub Customizer in Ubuntu: The software has an official PPA repository contains the packages for all current Ubuntu releases.

1. Open terminal either via Ctrl+Alt+T or by searching for 'terminal' from app launcher. When it opens, run command to add the PPA:
   - *sudo add-apt-repository ppa:danielrichter2007/grub-customizer*
2. After added the PPA, run commands one by one to refresh package cache and install Grub Customizer:
   - *sudo apt-get update*
   - *sudo apt-get install grub-customizer*

Now, we can choose to run ubuntu with kernel-4.4.0-x or another kernels.

2 ROS Installation

ROS Kinetic ONLY supports Wily (Ubuntu 15.10), Xenial (Ubuntu 16.04) and Jessie (Debian 8) for debian packages.

2.1 Configure your Ubuntu repositories

Configure your Ubuntu repositories to allow "restricted," "universe," and "multiverse." You can follow the Ubuntu guide for instructions on doing this.

2.2 Setup source.list

Setup your computer to accept software from packages.ros.org.

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

2.3 Setup keys

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

If you experience issues connecting to the keyserver, you can try substituting hkp://pgp.mit.edu:80 or hkp://keyserver.ubuntu.com:80 in the previous command.

Alternatively, you can use curl instead of the apt-key command, which can be helpful if you are behind a proxy server:

```
curl -sSL 'http://keyserver.ubuntu.com/pks/lookup?op=get&search=0xC1
CF6E31E6BADE8868B172B4F42ED6FBAB17C654' | sudo apt-key add -
```

2.4 Installation

First, make sure your Debian package index is up-to-date:

```
sudo apt-get update
```

There are many different libraries and tools in ROS. ROS.org provided four default configurations to get started: **Desktop-Full Install: (Recommended)**, **Desktop Install**, **ROS-Base: (Bare Bones)** and **Individual Package**. You can also install ROS packages individually.

In case of problems with the next step, you can use following repositories instead of the ones mentioned above ros-shadow-fixed.

We install **Desktop-Full Install**.

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

Note: if you must start from the scratch

```
sudo apt-get remove --purge ros-*
```

To find available packages, use:

```
apt-cache search ros-kinetic
```

2.5 Initialize rosdep

Before you can use ROS, you will need to initialize rosdep. rosdep enables you to easily install system dependencies for source you want to compile and is required to run some core components in ROS.

```
sudo rosdep init
rosdep update
```

2.6 Environment setup

It's convenient if the ROS environment variables are automatically added to your bash session every time a new shell is launched:

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

If you have more than one ROS distribution installed, /.bashrc must only source the setup.bash for the version you are currently using.

If you just want to change the environment of your current shell, instead of the above you can type:

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

If you use zsh instead of bash you need to run the following commands to set up your shell:

```
echo "source /opt/ros/kinetic/setup.zsh" >> ~/.zshrc
source ~/.zshrc
```

2.7 Dependencies for building packages

Up to now you have installed what you need to run the core ROS packages. To create and manage your own ROS workspaces, there are various tools and requirements that are distributed separately. For example, rosinstall is a frequently used command-line tool that enables you to easily download many source trees for ROS packages with one command.

To install this tool and other dependencies for building ROS packages, run:

```
sudo apt install python-rosinstall python-rosinstall-generator
python-wstool build-essential
```

3 Installation of the robot CRUMB (GAZEBO)

CRUMB robot is based on turtlebot2-gazebo, so it the first step we need accomplish is the installation of such package. Nevertheless, we need install some dependencies. Once the dependencies are installed, we will proceed to download and install the CRUMB robot.

3.1 turtlebot2 / CRUMB dependencies

3.1.1 Register the server's public key:

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

3.2 Add the server to the list of repositories:

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

3.2.1 Install the libraries:

```
sudo apt-get install librealsense2-dkms librealsense2-utils
```

Note: the above two lines will deploy librealsense2 udev rules, build and activate kernel modules, runtime library and executable demos and tools.

3.2.2 Install de the package ros-kinetic-arbotix

```
sudo apt install ros-kinetic-arbotix
```

3.2.3 setup source.list (optional)

```
sudo sh -c 'echo "deb-src http://us.archive.ubuntu.com/ubuntu/ xenial main
restricted deb-src http://us.archive.ubuntu.com/ubuntu/xenial-updates main
restricted deb-src http://us.archive.ubuntu.com/ubuntu/ xenial-backports main
restricted universe multiverse deb-src http://security.ubuntu.com/ubuntu/
xenial-security main restricted" >
\/etc/apt/sources.list.d/official-source-repositories.list'
```

3.2.4 Install linux header generic:

```
sudo apt-get install -y linux-headers-generic
```

3.3 install turtlebot-gazebo

```
sudo apt-get install ros-kinetic-turtlebot-gazebo
```

3.3.1 The following package is used to check the installation of turtlebot.

Download test source from github

```
cd ~
git clone -b base https://github.com/richardw05/mybot_ws.git
catkin_init_workspace
catkin_make # build
echo "source ~/mybot_ws/devel/setup.bash" >> ~/.bashrc # Adds workspace to search path
source ~/.bashrc
```

```
roslaunch turtlebot_gazebo turtlebot_world.launch
```

From same machine

```
roslaunch  turtlebot_teleop  keyboard_teleop.launch
```

3.4 Installation of the CRUMB software

Prepare crumb workspace.

```
cd ~ && mkdir crumb_ws && mkdir crumb_ws/src
```

Download CRUMB sources from github repository.

```
git clone https://github.com/CRUMBproject/ROS.git
```

Note: the current sources of CRUMB can not be built on 16.04 + kinetic. Please, refer to Vicente for more details.

Copy the content of the crumb directory to crumb_ws/src and remove the folders.

```
~/crumb_ws/src/crumb/crumb_listener and ~/crumb_ws/src/roboticsgroup_gazebo_plugins
```

And Doing:

```
catkin_make
```

TODO: it is necessary to check the functionality of both deleted nodes and repair them (adapt them to the current version of gazebo -7.12-).

3.5 Possibles Errors

- ERROR: cannot launch node of type [controller_manager/spawner]: controller_manager.

We have to install "ros-kinetic-ros-control" package.

```
sudo apt install ros-kinetic-ros-control
```

4 VPN configuration (needed for running nodes in multiple machines/platforms)

Install openvpn in both PCs

```
sudo apt install openvpn
```

Download the latest version of EasyRSA (needed for generating the RSA keys)

```
sudo -i

cd /opt
git clone https://github.com/OpenVPN/easy-rsa.git
cd easy-rsa/easyrsa3
```

other way

```
cd /opt
wget https://github.com/OpenVPN/easy-rsa/releases/download/v3.0.4/EasyRSA-3.0.4.tgz
# (If it is not downloaded, check the latest version available in the repository,
it may change)
tar -xvf EasyRSA-3.0.4.tgz
cd EasyRSA-3.0.4/
```

Setup RSA info

```
cp vars.example vars
```

Open vars file and find the following commented vars

```
#set_var EASYRSA_REQ_COUNTRY "US"
#set_var EASYRSA_REQ_PROVINCE "California"
#set_var EASYRSA_REQ_CITY "San Francisco"
#set_var EASYRSA_REQ_ORG "Copyleft Certificate Co"
#set_var EASYRSA_REQ_EMAIL "me@example.net"
#set_var EASYRSA_REQ_OU "My Organizational Unit"
```

Uncomment them, and change their values according yours preferences

```
set_var EASYRSA_REQ_COUNTRY "ES"
set_var EASYRSA_REQ_PROVINCE "Malaga"
set_var EASYRSA_REQ_CITY "Malaga"
set_var EASYRSA_REQ_ORG "UMA"
set_var EASYRSA_REQ_EMAIL "garthim@uma.es"
set_var EASYRSA_REQ_OU "ISA"
```

Initialize the environment

```
./easyrsa init-pki
```

Generate the Diffie-Hellman parameters

```
./easyrsa gen-dh
```

Generate RSA key and CA certificate

```
./easyrsa build-ca nopass # with no password protection, use garthim as an entity
```

Generate and sign the station RSA key and certificate

```
./easyrsa gen-req station nopass
./easyrsa sign-req server station
```

Generate and sign the robot (1) RSA key and certificate

```
./easyrsa gen-req robot1 nopass
./easyrsa sign-req client robot1
```

Create the certificate revocation list:

```
sudo ./easyrsa gen-crl
sudo cp pki/crl.pem /etc/openvpn/keys/
```

[Optional] Improving server robustness against DDoS attacks

```
openvpn --genkey --secret ta.key
```

List of files generated during the previous process

```
pki/dh.pem (station)
pki/crl.pem (station)
pki/ca.crt (station and robots)
ta.key (station and robots)
pki/private/ca.key (to sign in the CA machine)

pki/private/station.key (station)
pki/issued/station.crt (station)

pki/private/robot1.key (robot1)
pki/issued/robot1.crt (robot1)
```

Copying keys and certificates to the station

```
sudo mkdir /etc/openvpn/keys
sudo cp pki/dh.pem /etc/openvpn/keys
sudo cp pki/ca.crt /etc/openvpn/keys
sudo cp pki/private/station.key /etc/openvpn/keys
sudo cp pki/issued/station.crt /etc/openvpn/keys
sudo cp ta.key /etc/openvpn/keys
sudo cp pki/crl.pem /etc/openvpn/keys
```

Copying keys and certificates to each robot1

```
scp pki/private/robot1.key varevalo@192.168.1.100:/tmp
scp pki/issued/robot1.crt varevalo@192.168.1.100:/tmp
scp pki/ca.crt varevalo@192.168.1.100:/tmp
scp ta.key varevalo@192.168.1.100:/tmp
```

And moving them to the right folders

```
sudo mkdir /etc/openvpn/keys
sudo mv /tmp/robot1.key /etc/openvpn/keys
sudo mv /tmp/robot1.crt /etc/openvpn/keys
sudo mv /tmp/ca.crt /etc/openvpn/keys
sudo mv /tmp/ta.key /etc/openvpn/keys
```

Create the station configuration file

```
sudo nano /etc/openvpn/station.conf
```

And copy the following in it.

```
port 1194
proto udp
server 192.168.10.0 255.255.255.0
client-to-client
persist-key
persist-tun
ca /etc/openvpn/keys/ca.crt
cert /etc/openvpn/keys/station.crt
dh /etc/openvpn/keys/dh.pem
key /etc/openvpn/keys/station.key
tls-auth /etc/openvpn/keys/ta.key 0
crl-verify /etc/openvpn/keys/crl.pem
comp-lzo adaptive
dev tun
ifconfig-pool-persist server-ipp.txt 0
keepalive 10 120
cipher AES-256-CBC
auth SHA512
tls-version-min 1.2
tls-cipher TLS-DHE-RSA-WITH-AES-256-GCM-SHA384
log /var/log/openvpn/server.log
verb 3
```

Check configuration and launch server as service if it workspace

```
sudo openvpn --config /etc/openvpn/station.conf
```

If it works

```
sudo systemctl start openvpn.service
```

Create the robot1 configuration file

```
sudo nano /etc/openvpn/robot1.conf
```

And copy the following in it:

```
client
dev tun
proto udp
port 1194
remote 192.168.1.200 1194
remote-cert-tls server
resolv-retry infinite
nobind
persist-key
persist-tun
comp-lzo
verb 3
cipher AES-256-CBC
auth SHA512
tls-version-min 1.2
tls-cipher TLS-DHE-RSA-WITH-AES-256-GCM-SHA384
ca /etc/openvpn/keys/ca.crt
key /etc/openvpn/keys/robot1.key
cert /etc/openvpn/keys/robot1.crt
key-direction 1
tls-auth /etc/openvpn/keys/ta.key 1
```

Check the configuration from command line

```
sudo openvpn --config /etc/openvpn/robot1.conf
```

And, if it properly works, launchs server as service

```
sudo systemctl start openvpn.service
```

5 Setup ROS for working over a VPN

Finally, a couple of changes must be added to the /.bashrc files.

```
# ROS files
source /opt/ros/kinetic/setup.bash
source ~/catkin_ws/devel/setup.bash
...

# Name of the machine in the VPN network. The usage of this
# var implies to add the pairs <ip,hostname> of all VPN
# network to the hosts files.
export ROS_HOSTNAME=station # station or robot1

# it is possible to use the host ip (inside of the vpn) instead of hostname.
export ROS_IP=192.168.10.1

# finally, the URI of the host that runs roscore must to added.
export ROS_MASTER_URI=http://localhost:11311 # if roscore is executed locally
#export ROS_MASTER_URI=http://192.168.10.1:11311  # if roscore is executed
# remotely
```