Groups:

Each of you will be split into groups of 3 for the project. On Courseworks you will need to sign up for a group of your own choosing by HW0. Make sure you have selected your teammates and if anyone is still looking for teammates please email me davidwatkins@cs.columbia.edu or post on the course Piazza here.

Setup:

Buy a flash drive

In order to do the homework for this class, as well as complete your projects, you will need a computer running Ubuntu 16.04 with access to OpenGL, in particular it needs to run applications with an Ogre runtime environment. This is for applications such as Gazebo and Rviz (which we will touch on later). Because the CLIC lab does not have publicly accessible computers and using a cloud based computing service can be very slow for streaming OpenGL applications we have opted to use Ubuntu 16.04 installed on a USB 3.0 64GB flash drive. I recommend every group invest in a drive with a high speed, such as this one here. Alternatively I have had success using Parallels on an Apple computer for this purpose however you will be responsible for debugging any issues with this. Each group will need to have their own flash drive and will need to show me that they can run Gazebo on their computer for HW0. If for any reason you cannot use a flash drive or you are unable to use some alternative such as Parallels please reach out to me and we can discuss alternatives davidwatkins@cs.columbia.edu.

Install Ubuntu 16.04 on the flash drive

Once you have obtained your drive you will need to install Ubuntu 16.04 on it. There are a few ways of doing this but here is the most straightforward way.

1. Install Virtualbox from here
2. Install VirtualBox 5.2.6 Oracle VM VirtualBox Extension Pack also from here
3. Download the Ubuntu 16.04.3 LTS ISO from here
4. Create a new Virtual Machine, call it Ubuntu, select type: Linux and Version: Ubuntu (64-bit)
5. Use the default memory (it doesn't matter)
6. Do not add a virtual hard disk
7. If you are running Linux or Mac OS X you will likely need to add yourself to the vboxusers group

$ sudo adduser $USER vboxusers

1. Attach your USB drive to your machine
2. Start your VM and choose install Ubuntu
3. Click on clean install and choose your USB drive
4. Wait until it finishes installing
5. After your installation, test your installation by rebooting into your drive through your BIOS to see if it worked. On an Apple machine this will mean holding the ⌘ option key down.
6. You should have successfully booted into an Ubuntu 16.04 instance!


Install ROS Kinetic

The installation of ROS is just setting up the correct apt repository on your fresh installation and making sure programs such as Gazebo run. If you have any trouble with this section please refer to the official ROS tutorial here. I also highly recommend you use `apt` instead of `apt-get` whenever possible as it is more informative and sometimes faster.

```
$ 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://ha.pool.sks-keyservers.net:80 --recv-key 421C365BD9FF1F717815A3895523BAEEB01FA116
$ sudo apt update
$ sudo apt install ros-kinetic-desktop-full
$ sudo rosdep init
$ sudo rosdep update
$ echo "$source /opt/ros/kinetic/setup.bash" >> ~/.bashrc
$ source ~/.bashrc
$ sudo apt-get install python-rosinstall python-rosinstall-generator python-wstool build-essential python-catkin-tools
```

To affirm that the installation went well open Gazebo on your machine and make sure you can add a robot. You can open Gazebo by clicking the Dash button on the top left and typing Gazebo. Once you have it open you can insert a robot into the scene by clicking the Insert tab on the top left, selecting the PR2, and dragging it into the scene. The PR2 is probably the most intensive robot to simulate and therefore is a good benchmark as to whether your computer will comfortably run ROS simulations.

Additional Installs

I also recommend you install node-manager. This is a very useful tool for debugging ROS nodes and will help in general launching complex launch scripts.

Build Environment

For our class, and for the sake of simplicity, each group should use the following build tools:

- **catkin** is the default ROS cmake replacement. This allows for easy compilation of projects which involve multiple programming languages and is very good at debugging potential issues with compilation in your code.
- **Kinetic** is the version of ROS that is compatible with Ubuntu 16.04. You may find while Googling around for help that other people may have used Indigo, Hydro, or Jade, all of which are older
versions of ROS. Kinetic has the most support and longevity of the current ROS distros and therefore will be what is taught in this class.

- **Python, C/C++, and Javascript** all have compatible ROS software packages. You are welcome to use any of them in your projects although I advise you to use Python. You can use C/C++ if you are comfortable with them. Only use the Javascript variety if you are building a Node.js application, otherwise it is not completely finished and is missing some crucial features and documents. There are other languages that are compatible with ROS but your TA has not used them and therefore is unable to vouch for or assist with the use of these alternatives.

- **Jetbrains CLion** is by far the best editor for ROS as it will allow you to auto complete your code, set up new projects quickly, and offer an easy to use debugging solution. If there is enough demand I will post a CLion set up tutorial. I recommend you all get the student license which is valid as long as you have a .edu email.

- **Gitman** is an excellent tool for managing dependencies of projects which are varying in language. Unfortunately ROS dependencies are installed mostly through apt, which can makes projects for this class which depend primarily on git based dependencies particularly challenging. I recommend you get familiar with gitman and look [here](#) for an example of a gitman.yml for a ROS project.

### Tutorials

For this class I recommend you look through the official ROS tutorials for Kinetic to get comfortable with the software. Some notes to consider before going through the tutorials:

- Replace `catkin_make` with `catkin build`
- Replace `catkin_init_workspace` in the `workspace/src` directory with `catkin init` in the `workspace` directory
- If for some reason you cannot access specific commands in ROS you will need to run `source /opt/ros/kinetic/setup.bash` after following the installation instructions above. The command `echo "source /opt/ros/kinetic/setup.bash" >> ~/.bashrc` puts this command into your `~/.bashrc` file which is executed every time you open a new terminal window. The `setup.bash` script inserts specific environment variables for your build environment to link against ROS dependencies.
- To easily open a new Terminal window in ROS use `ctrl+alt+T`

I recommend you finish the following tutorials to get comfortable with ROS:

1. **Installing and Configuring Your ROS Environment** - This tutorial walks you through installing ROS and setting up the ROS environment on your computer.
2. **Navigating the ROS Filesystem** - This tutorial introduces ROS filesystem concepts, and covers using the roscd, rosls, and rospack commandline tools.
3. **Creating a ROS Package** - This tutorial covers using roscreate-pkg or catkin to create a new package, and rospack to list package dependencies.
4. **Building a ROS Package** - This tutorial covers the toolchain to build a package.
5. **Understanding ROS Nodes** - This tutorial introduces ROS graph concepts and discusses the use of roscore, rosnode, and rosrerun commandline tools.
6. **Understanding ROS Topics** - This tutorial introduces ROS topics as well as using the rostopic
and rqt_plot commandline tools.

7. **Understanding ROS Services and Parameters** - This tutorial introduces ROS services, and parameters as well as using the rosservice and rosparam commandline tools.

8. **Using rqt_console and roslaunch** - This tutorial introduces ROS using rqt_console and rqt_logger_level for debugging and roslaunch for starting many nodes at once. If you use ROS fuerte or earlier distros where rqt isn't fully available, please see this page with this page that uses old rx based tools.

9. **Using rosed to edit files in ROS** - This tutorial shows how to use rosed to make editing easier.

10. **Creating a ROS msg and srv** - This tutorial covers how to create and build msg and srv files as well as the rosmsg, rossrv and roscp commandline tools.

11. **Writing a Simple Publisher and Subscriber (C++)** - This tutorial covers how to write a publisher and subscriber node in C++.

12. **Writing a Simple Publisher and Subscriber (Python)** - This tutorial covers how to write a publisher and subscriber node in python.

13. **Examining the Simple Publisher and Subscriber** - This tutorial examines running the simple publisher and subscriber.

14. **Writing a Simple Service and Client (C++)** - This tutorial covers how to write a service and client node in C++.

15. **Writing a Simple Service and Client (Python)** - This tutorial covers how to write a service and client node in python.

16. **Examining the Simple Service and Client** - This tutorial examines running the simple service and client.

A full list of tutorials for ROS can be found at: [http://wiki.ros.org/ROS/Tutorials](http://wiki.ros.org/ROS/Tutorials).