# Install Ubuntu 16.04 in a virtual machine (or main machine)

The flight software you will be running for this class exercise has been configured and compiled to run under the Ubuntu 16.04 operating system. To more rapidly start using the software, we will create a virtual Ubuntu operating system environment on your machine.

We will use a virtual machine to make the running of this software easy in a known environment. In these examples, I am assuming that we will use VirtualBox, but any virtualization system can work.

1. Download VirtualBox
2. Download Ubuntu .iso
3. Install Ubuntu in VirtualBox
   1. In the VirtualBox GUI select new machine with the following parameters
      1. Name is CFS
      2. Type is Linux
      3. Version is Ubuntu (64 bit).
   2. Select defaults for memory (4GB recommended) and disk space (~50 GB recommended). Use defaults for everything else.
   3. Save the machine.
4. Update Settings
   1. elect the CFS machine you just created and then select “settings” from the main menu.
   2. Under “System”->Processor set the number of cores to run
   3. Under Display set the video memory (I use 32MB) and 1 monitor
   4. Under “Storage” load the Ubuntu .iso file in the virtual “CD-ROM drive”.
      1. In the ”Storage Tree” area under “Controller: IDE” select the icon of a disc.
      2. On the right “Attributes” pane, next to Optical Drive, select the little icon of a disc there, then “Choose Virtual optical Disk File” and select the Ubuntu .iso file.
      3. Hit OK.
5. Turn on the machine
   1. With the CFS machine selected, press the “start” button in Virtualbox.
   2. Select “Install Ubuntu” button.
   3. Select “Download updates while installing Ubuntu”
   4. Select “Erase disk and install Ubuntu” -> don’t worry, this is just the virtual machine. The hit “Install Now”, then Continue.
   5. Select timezone and keyboard.
   6. Select username: cfs
   7. Choose password: cfs\_password
   8. Hit “continue”
   9. Wait… eventually reboot. VirtualBox automatically removes the install medium so you can just hit enter when prompted.
6. Log in and
   1. **Install ruby sudo apt-get install ruby ruby-dev**

# Install Tools

1. Open a terminal in Ubuntu
2. Go to your home directory (~/)
3. Install GIT
   1. **sudo apt-get install git**
4. Install some build libraries
   1. **sudo apt-get install g++-multilib**
5. Install CURL (needed for COSMOS)
   1. **sudo apt-get install curl**

# Clone the Project from GitHub

1. Clone the project GIT repo.
   1. **git clone https://github.com/NasaDtn/spacesystems.git**

# Build and Configure the CFS Image

The NASA Goddard Space Flight Center GSFC) has released, open source, the Core Flight System (cFS) which comprises an Operating System Abstraction Layer (OSAL), an underlying execution infrastructure – the Core Flight Executive (cFE), and a series of applications that, together, make up a Command and Data Handling (C&DH) flight system. This step will compile the OSAL, cFE, and cFS in a docker container and run the software to make sure everything is working correctly.

1. Set up the CFS environment
   1. **cd cfs**
   2. **. ./setvars.sh (Note the first . is not a typo, it is . ./setvars.sh NOT ./setvars.sh)**
2. Build the CFS code
   1. Change to the build directory: **cd /build/cpu1**
   2. Configure the build: **make config**
   3. Build the OSAL, cFE, and CFS apps: **make**
   4. Wait while it all compiles. Look for “>>> DONE! <<<” at the end.
3. Make sure that CFS runs successfully on your platform.
   1. From the cpu1 directory (that you just built in) go to the exe folder: **cd exe**
   2. Run CFS: **sudo ./core-linux.bin –reset PO**
   3. Wait until you see the “Stop FLYWHEEL” message.
   4. Type CTRL-C to stop CFS and return to the command prompt.

You now have a docker container with a build CFS instance that is running an out-of-the-box spacecraft Command and Data Handling flight software system!

# Install and configure the Ground Command and Control Software

Now that we have a flight system, we need some way to communicate with that system by issuing commands and receiving back telemetry. For this project we will use the COSMOS open-source ground system.

1. Run the Ubuntu Linux COSMOS install script from <http://cosmosrb.com/docs/installation/)>
   1. It should look something like:
      1. bash <(\curl -sSL https://raw.githubusercontent.com/BallAerospace/COSMOS/master/vendor/installers/linux\_mac/INSTALL\_COSMOS.sh)
   2. When prompted, select S (for sudo) at the first prompt.
   3. Select Y for installing rbEnv (this will take a while to build things)
   4. Select Y for install and run cosmos demo
      1. Select OK when COSMOS window pops up
      2. Close the Launcher by clicking X on the launcher window.
2. Update your path
   1. **source ~/.bashrc**
3. Create a project demo (this will create a demo in ~/cfs\_demo)
   1. **cosmos demo cfs\_demo**
4. Replace the config directory from cfs\_demo with the config directory from the spacesystems area
   1. **rm -rf ~/cfs\_demo/config**
   2. **cp -R ~/spacesystems/config ~/cfs\_demo**
5. Start COSMOS
   1. **cd ~/cfs\_demo**
   2. **ruby Launcher**
   3. **Select “Update Project CRCs” from the splash screen then OK on the pop-up**
   4. **Select OK to close the splash screen**

# Commanding Flight Software

1. In a new terminal window start CFS
   1. Remember run . ./setvar.sh
2. Start COSMOS
   1. Remember, ruby Launcher from within your cfs\_demo directory
3. Open the Command And Telemetry Server
4. Open the Command Sender (select OK to pop-up)
5. Open the Telemetry Viewer
   1. Select show screen next to the WHE entry
6. Enable TO Output
   1. Select the TO\_OUTPUT\_ENABLE command from the Command Sender
   2. Set the last argument (current a long string) to the value “127.0.0.1”
   3. Send the command
   4. Watch telemetry stream in on the telemetry viewer screen