**Assignment 1**

# CSCI E-118 Introduction to Blockchain and Bitcoin

## Part 1: Installations

### Installing Ubuntu

If you have a **LTS** version of Ubuntu, **14.04** through **20.04**, you're all set. That includes code names: **Trusty Tahr**, **Xenial Xerus**, **Bionic Beaver**, or **Focal Fossa**. Otherwise...

1) Go to https://releases.ubuntu.com/bionic/

Download the appropriate ISO desktop image file.

Unless you know what you're doing, and have another preference, you should click on and download **"64-bit PC (AMD64) desktop image"**. Don't worry if your machine has an Intel processor instead of an AMD processor. They both use the same architecture (**x86_64**).

What we are installing is Ubuntu version **18.04 (LTS)** , codename **Bionic Beaver**, which has standard support until April 2023, and extended support until April 2028. See here if you are interested in support for other versions: https://wiki.ubuntu.com/Releases. See here if you are interested in how versioning works for Ubuntu: https://ubuntu.com/about/release-cycle.

2) Go to https://www.virtualbox.org/wiki/Downloads

Select the appropriate Host OS (the type of operating system your machine runs). - If you're on Windows -> "Windows hosts". - If you're on Mac -> "OS X hosts".

VirtualBox is a free application which provides virtualization for x86 and AMD64/Intel64. In other words, it's a virtual machine (VM). When using the VM, your base operating system (OS) is called the "Host". The operating system that you run in the VM is called the "Guest". In this case, we want to run Ubuntu as the "Guest OS".

3) Install VirtualBox.

When setting up, **SELECT "Fixed Size" for storage on Physical Disk (!)**. There are two

reasons for this. 1) So you don't accidently consume too much space; 2) The VM is supposedly a little faster when you used Fixed Size. If you'd rather choose Dynammically Allocated, that's fine.

You will need to allocate several GB's of space (especially with the Fixed Size option). Ubuntu takes up 15GB. On top of that you will need space for various tools, software, etc. which we will be installing within Ubuntu. 25GB should be sufficient. You can allocate more or less depending on how much space you have capacity for.

You will want to allocate at least 4GB of RAM. If you can afford to allocate more, it would be beneficial.

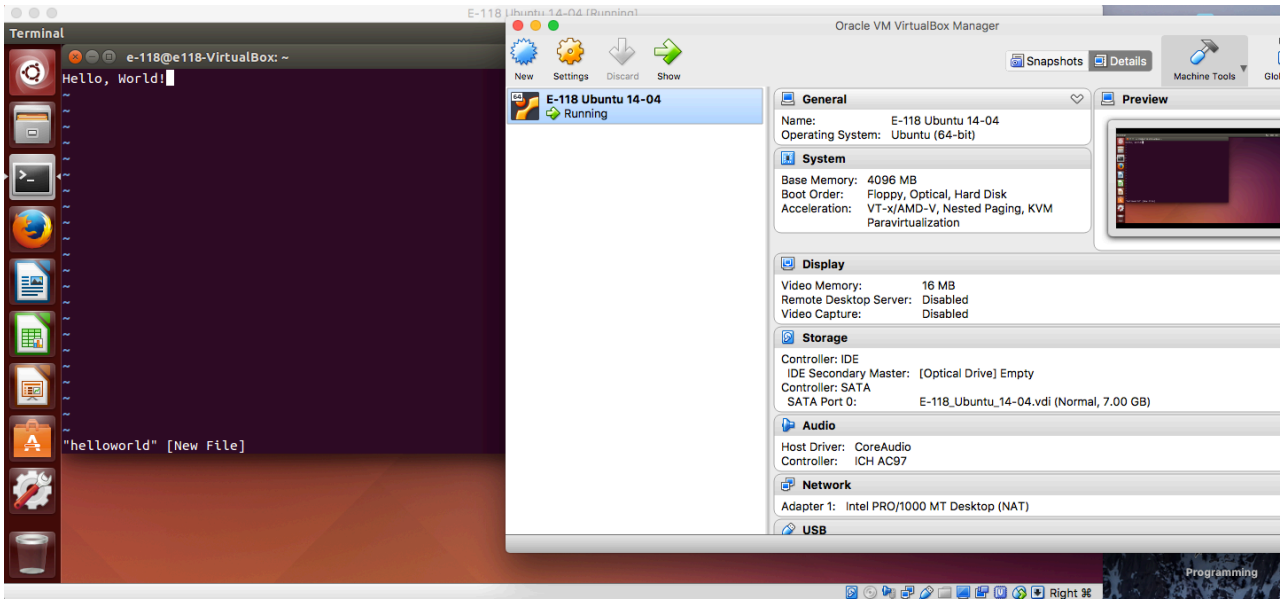
If you are running Windows, following this video may help: https://www.youtube.com/watch?v=5sa0acU4pmY. At **10:37** in the video, select "Fixed Size" (unless you really want Dynamically Allocated).

If you are running MacOS, follwoing this video may help: https://www.youtube.com/watch?v=ThsxqznrgCw. At **1:54** in the video, select "Fixed Size" (unless you really want Dynamically Allocated).

4) You might encounter a situation where you cannot select Ubuntu-64-bit within VirtualBox. If so try: https://askubuntu.com/questions/985356/i-cant-run-linux-64-bit-on-virtualbox -> go into the BIOS and look for virtualization options.

OR, try: https://docs.fedoraproject.org/en-US/Fedora/13/html/Virtualization*Guide/sect-Virtualization-Troubleshooting-Enabling*Intel*VT*and*AMDV virtualization*hardware*extensions*in_BIOS.html.

5) Once you have VirtualBox installed, and running Ubuntu, take a screenshot and submit it as part of Homework 1. It should look something like this:




6) You will probably want to install **Guest Additions**. This is a set of device drivers and system applications which will enable several useful features, e.g. better mouse support, shared folders (very useful), better video support, shared clipboard (copy-paste between Host and Guest), and more. See here for details: https://www.virtualbox.org/manual/ch04.html.

**Guest Additions** is included in your installation of VirtualBox, as a virtual CD-ROM. Basically, how this works is by putting the virtual CD inside an imaginary CD drive on your VM. To do this, first make sure VirtualBox is running and that Ubuntu is launched. On the window running the Guest (Ubuntu), look at the top. You should see a menu bar. Click **"Devices"** and then **"Insert Guest Additions CD Image..."**.

7) If you have installed **Guest Additions**, you can now set up a shared folder. This is a folder that both your Host and Guest have access to. That means you can easily transfer files between the two.

On your Host, create a folder somewhere on your file system, for example on your desktop, or in your documents folder. This will be the designated shared folder. Then navigate to the window running the Ubuntu VM. On the top menu bar of the window, click **"Devices"** and then **"Shared Folders > Shared Folder Settings"**. In this settings menu, click the blue folder icon with the green plus sign on it, in order to add a new shared folder. Select the folder path dropdown and choose **other**. Then choose the folder you designated as the shared folder on your Host in the previous step. Select **auto-mount** and then **OK** and **OK** again.

## Installing Software for Ubuntu

These are the main installations for the course. There may be others later, but these will give

you the facility in programming and testing Decentralized Applications.

Download **installations.sh** from canvas, in Assignment 1, and place somewhere within your Ubuntu VM.

Within the Ubuntu VM:

Open the terminal. You can click on the terminal icon, or press `Ctrl-Alt-T`.

1) Install Anaconda.

**Anaconda** includes **Conda** which is an open source package management system and environment management system for Python that is very useful. It's sort of like `pip` meets `venv`, if you know about those. **Anaconda** also includes a collection of open-source scientific packages. Examples are `numpy` and `pandas`. If you don't know about those, don't worry.

The **Anaconda** distribution takes up 3 GB of disk space. That includes all of the many packages it installs.

If you would rather install just the package and environment manage system, **Conda**, and then intall packages as you go, you can instead intall **Miniconda**, which takes up 400 MB of space.

**NOTE:** We will **not** use the vast majority of packages that come with **Anaconda**.

To install **Anaconda** or **Miniconda**, follow the instructions here https://docs.anaconda.com/anaconda/ (make sure you are intalling within the VM, not your Host).

2) Create a new Conda environment.

Once you have **Anaconda** or **Miniconda** installed, you will be able to use **Conda** (the package and environment manager).

We will create a new environment to use for the rest of this class.

In your terminal, type `conda create -n block_env python=3.6`.

`block_env` is the name of the environment. You can choose whatever name you want, just make sure to remember it (or you can type `conda env list` to see the environments you've created). Notice we've explicitly set `python=3.6`. This is to make sure we are all using the same version of Python. If you choose another version, you're responsible for handling and issues that may come up.

Now that we've created a new environment, type `conda activate block_env`. This will

activate the environment. Now any new packages we install will be under this environment. If we deactivate this environment, or switch to another one, those packages will not be available to us. In the future, if you want to keep using your Ubuntu VM, but you don't care about all of the tools we installed, you can just delete the entire environment using `conda env remove -n block_env`.

NOTE: You will need to run `conda activate block_env` everytime you launch terminal. If you know what you're doing, you can do this automatically by modifying your **.bashrc** file.

3) Run the installation script.

Once you've activated the conda environment, navigate to where you've stored **installations.sh** using the `cd` command.

Then run, `chmod u+x installations.sh`. That will change the permissions and make the **installations.sh** file executable. Next run the script by typing `bash installations.sh block_env`. Notice we've passed the environment name as an argument. The script expects this, so make sure to include it.

After the installation script finishes, close the terminal and reopen it.

SIDELINE: If you look in the script, you'll notice it's using **pip** to install packages. What about **conda**? In general you should use `conda install` when you can (when the package is available in the Anaconda repository). In general though, it's okay to use **pip**. When you do a `pip install` from within a conda environment, the package is still installed within the conda environment. See here and here for more details.

4) Launch Jupyter.

Among other things, the installation script installed **Jupyter**. If you aren't aware of **Jupyter Notebooks**, it's a programming envrionment. At its base is IPython, a Read-Evaluate-Print-Loop (REPL) for Python. Instead of running a Python script, the REPL lets you run Python interactively in the terminal. You type in a command, it evaluates that command, and prints the output, if any. **Jupyter Notebooks** extend this. A notebook runs in your web browser. It consists of cells, where you can input code. The output of the code is printed below the cell. You can also include text cells, which have support for text, markdown, and LaTeX. That means you can include math, images, videos, and more.

For a quick intro to **Jupyter Notebooks**, see here https://www.youtube.com/watch?v=jZ952vChhuI.

More specifically, we'll be using **Jupyter Lab** which is like an IDE for notebooks. For an intro to

**Jupyter Lab**, see here: https://www.youtube.com/watch?v=ctOM-Gza04Y.

To launch **Jupyter Lab**, go to your terminal and type `jupyter lab`. Your web browser should automatically launch with the full Jupyter environment.

Create a new notebook: File -> New -> Notebook.

The installation script set the **block_env** conda environment as one of the kernels. A kernel lets you run a given programming runtime. A Python kernel lets you run Python. **IPython** is the standard Python kernel. The **block_env** kernel that was set up for you lets you run **IPython** alongside the other Python packages installed in the conda environment.

Make sure the **block_env** kernel is running. In the menu bar, select: Kernel -> Change Kernel... -> select "Python(block_env)" in the dropdown menu.