

Setting Up Linux for CSE775 - DO

Jim Fawcett
CSE775 – Distributed Objects
Spring 2019

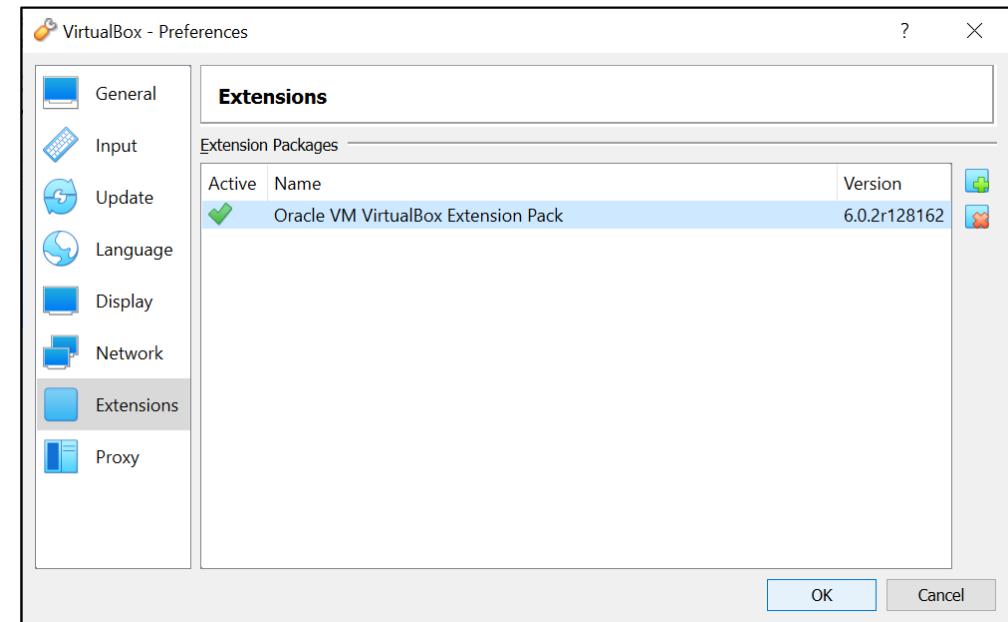
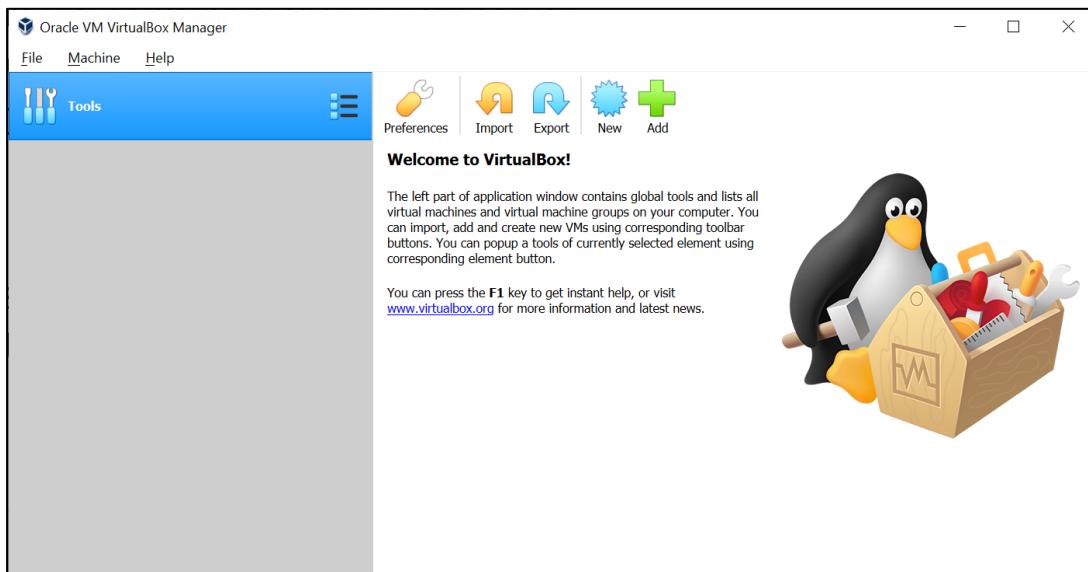
Tasks:

- Install VirtualBox
 - VMware Player would probably work as well
 - Download: <https://www.virtualbox.org/wiki/Downloads>
- Install a Debian Linux
 - I'll use Ubuntu 18.04
- Install “build essentials” tools
- Install Visual Studio Code (VS Code)
 - Setup build and launch tasks
- Optional Installs

Task #1 – Install Virtualbox

Installing Virtual Box

- I started by uninstalling Java and an old version of Virtualbox.
- Download and installation took about 2 minutes.
- <https://www.virtualbox.org/wiki/Downloads>
- Download Virtualbox extension pack and install



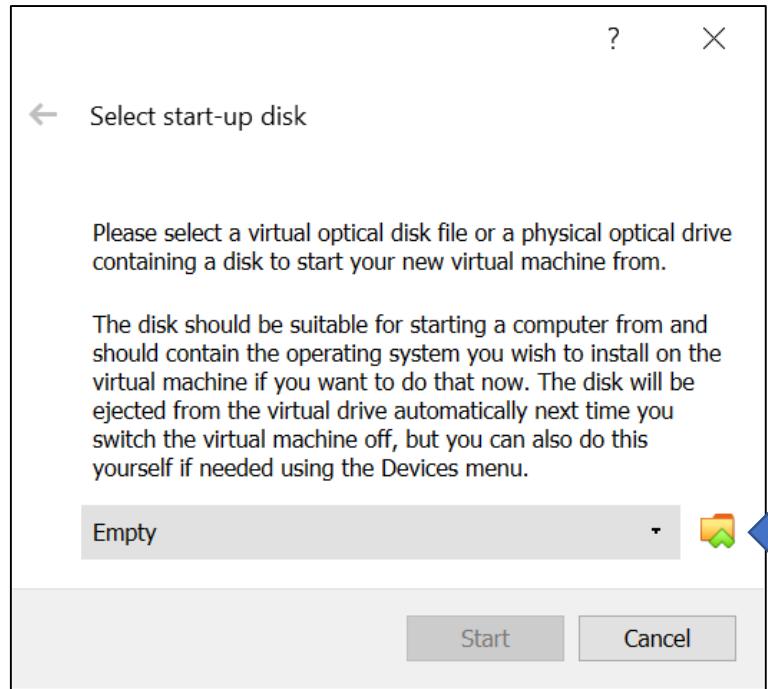
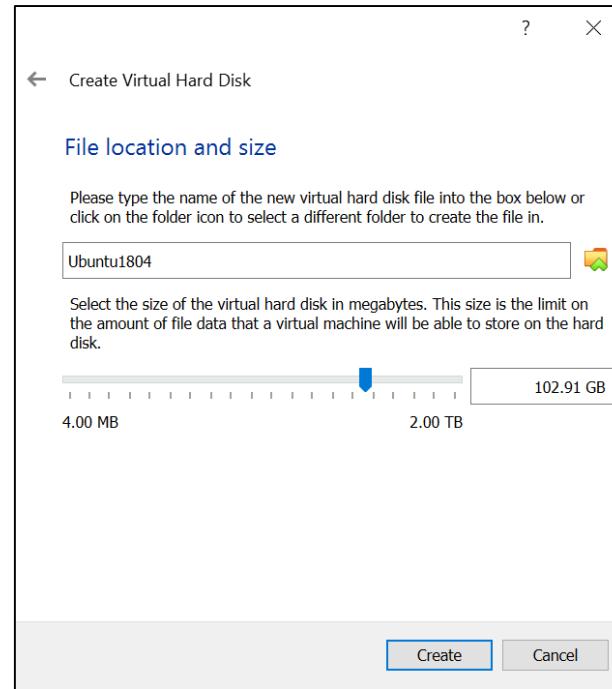
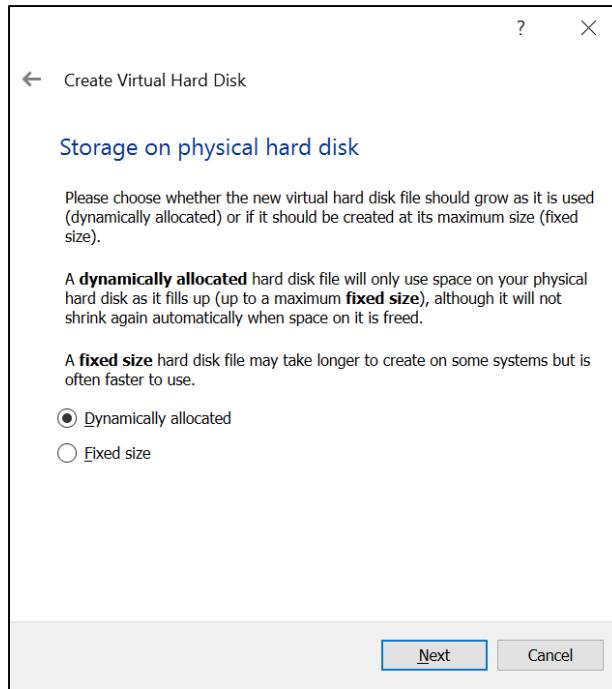
Setting VM Guest Properties

The image displays three sequential steps of a 'Create Virtual Machine' wizard:

- Memory size:** A slider and input field set at 4096 MB. A note says the recommended size is 1024 MB.
- Hard disk:** A note says you can add a virtual hard disk. Options include 'Do not add a virtual hard disk', 'Create a virtual hard disk now' (selected), and 'Use an existing virtual hard disk file'. A dropdown shows 'Empty'.
- Hard disk file type:** A note about choosing a file type. Options are 'VDI (VirtualBox Disk Image)' (selected), 'VHD (Virtual Hard Disk)', and 'VMDK (Virtual Machine Disk)'.

Need more RAM than recommended

Setup Virtual Disk and install Ubuntu



You will need a lot more disk space than the default setting

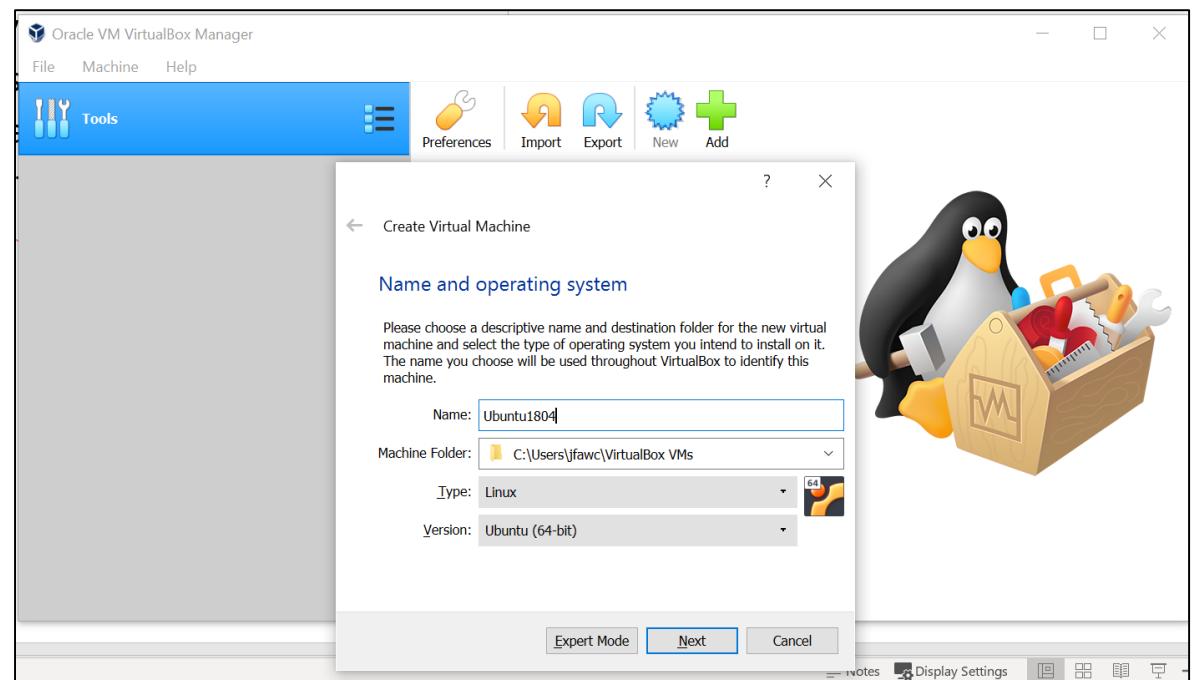
Here's where you select the Ubuntu iso file after downloading.

Task #2 – Install Ubuntu

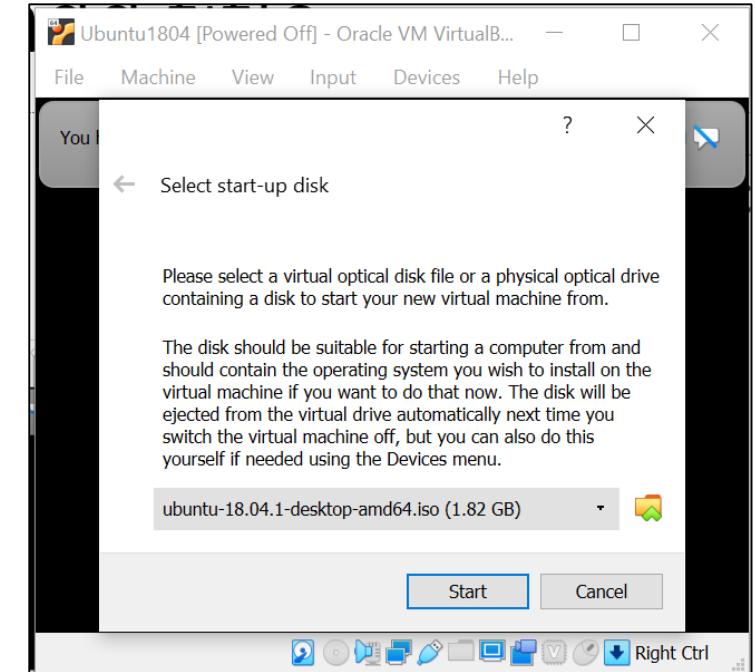
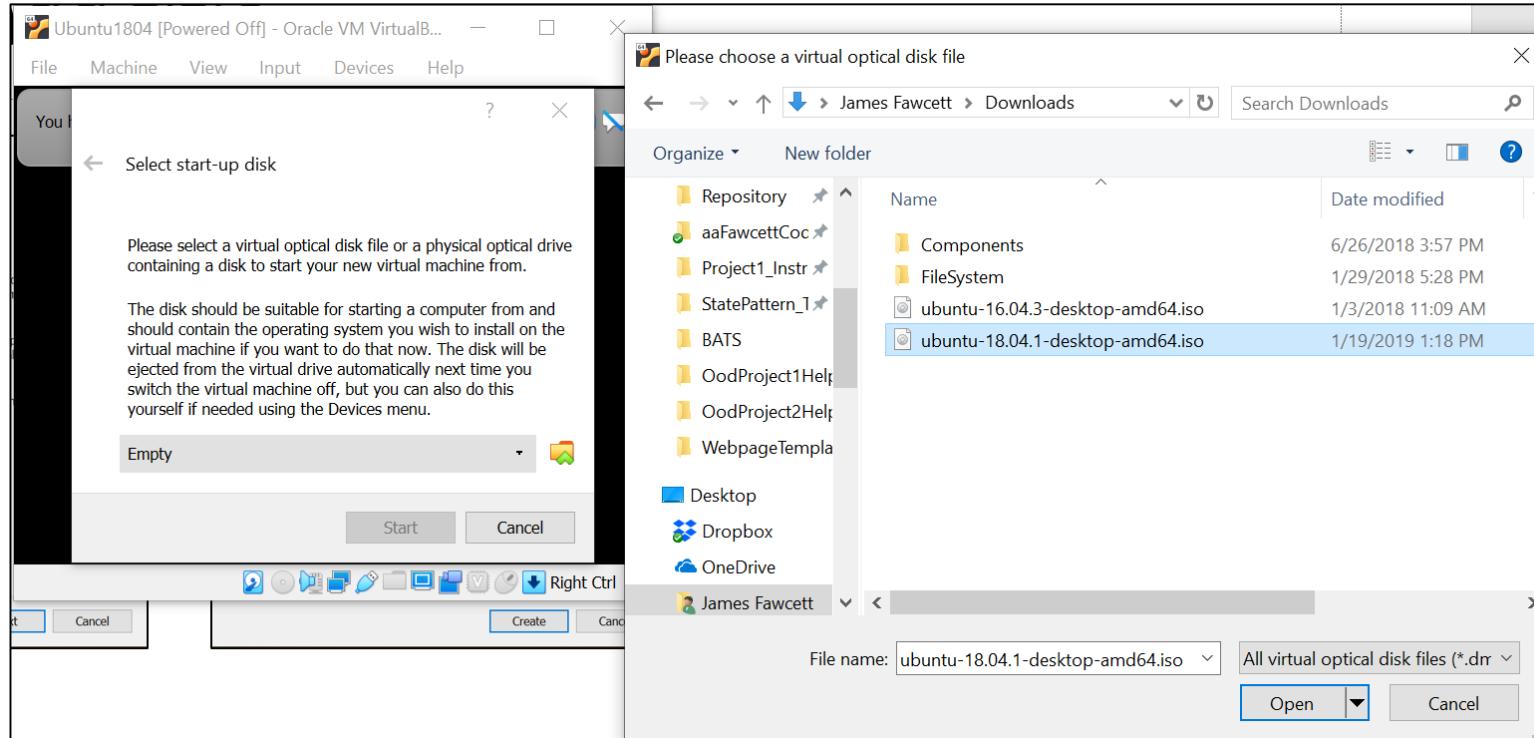
Download Ubuntu Desktop - 18.04.1 LTS

- Download Ubuntu (1.8GB)
 - <https://www.ubuntu.com/download/desktop>
 - Takes about 4 minutes
 - You get an iso image you install in Virtualbox.

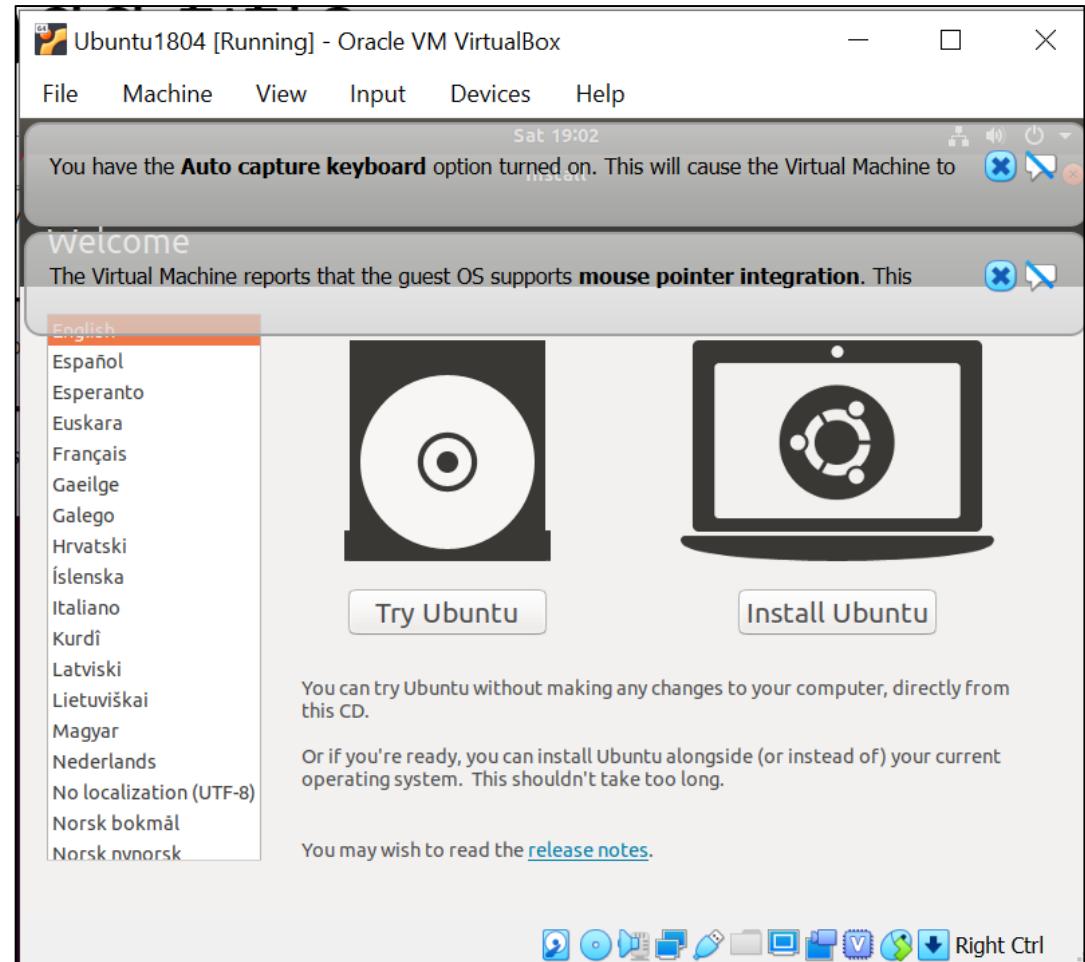
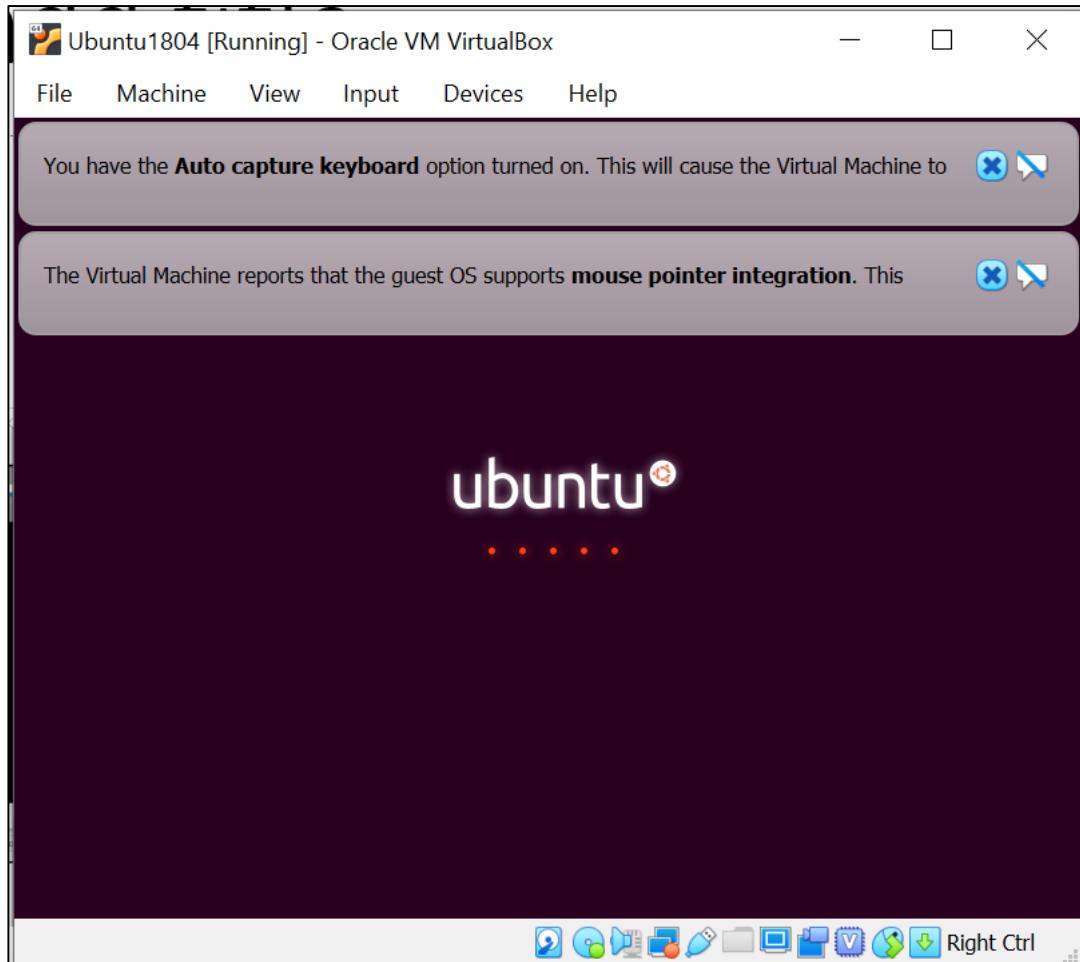
- Install in Virtualbox



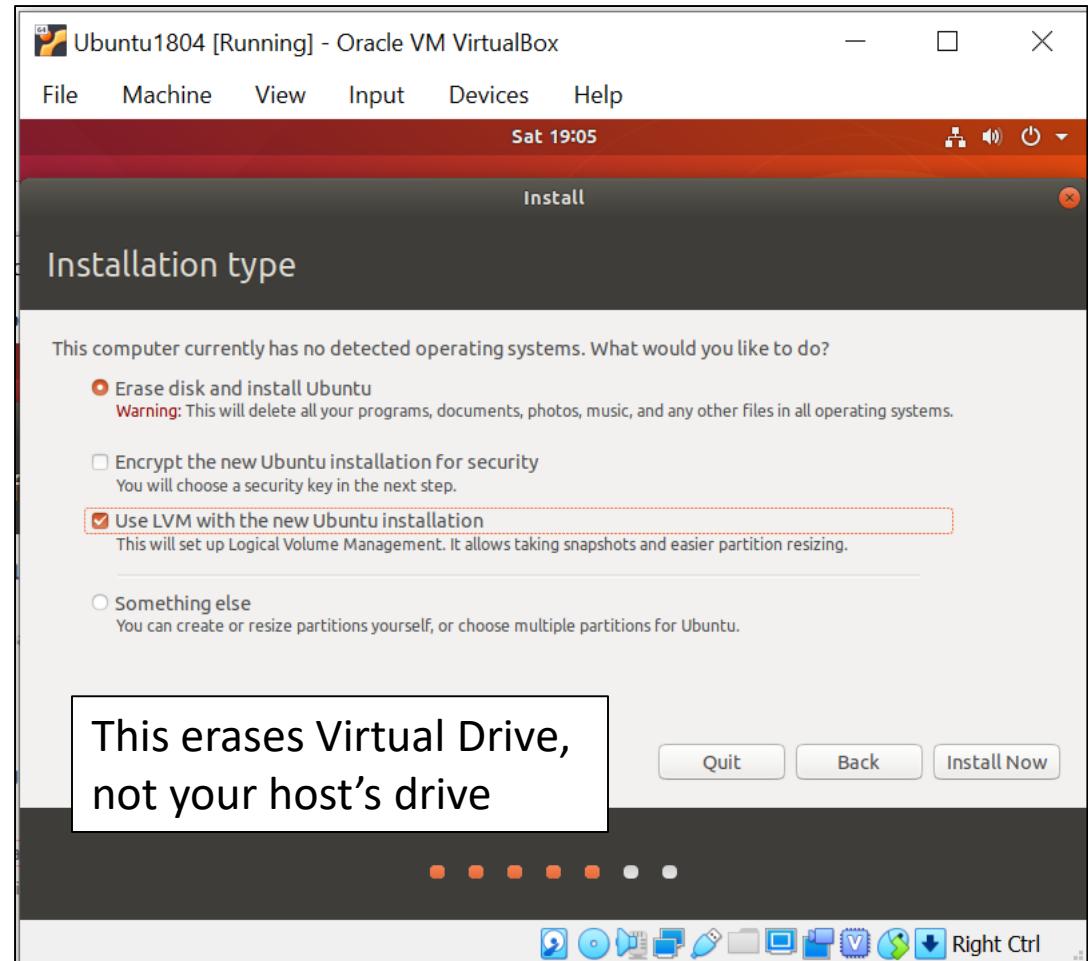
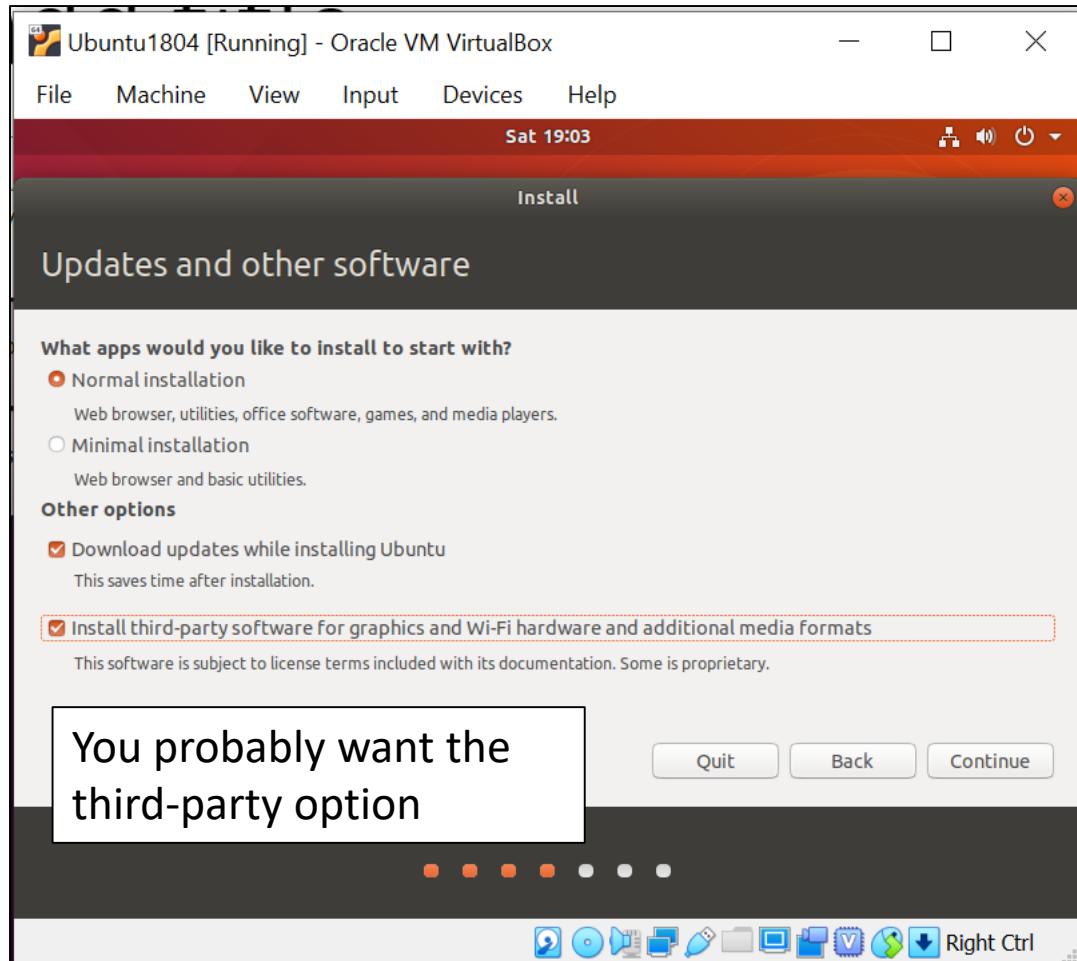
Select Ubuntu from Downloads folder



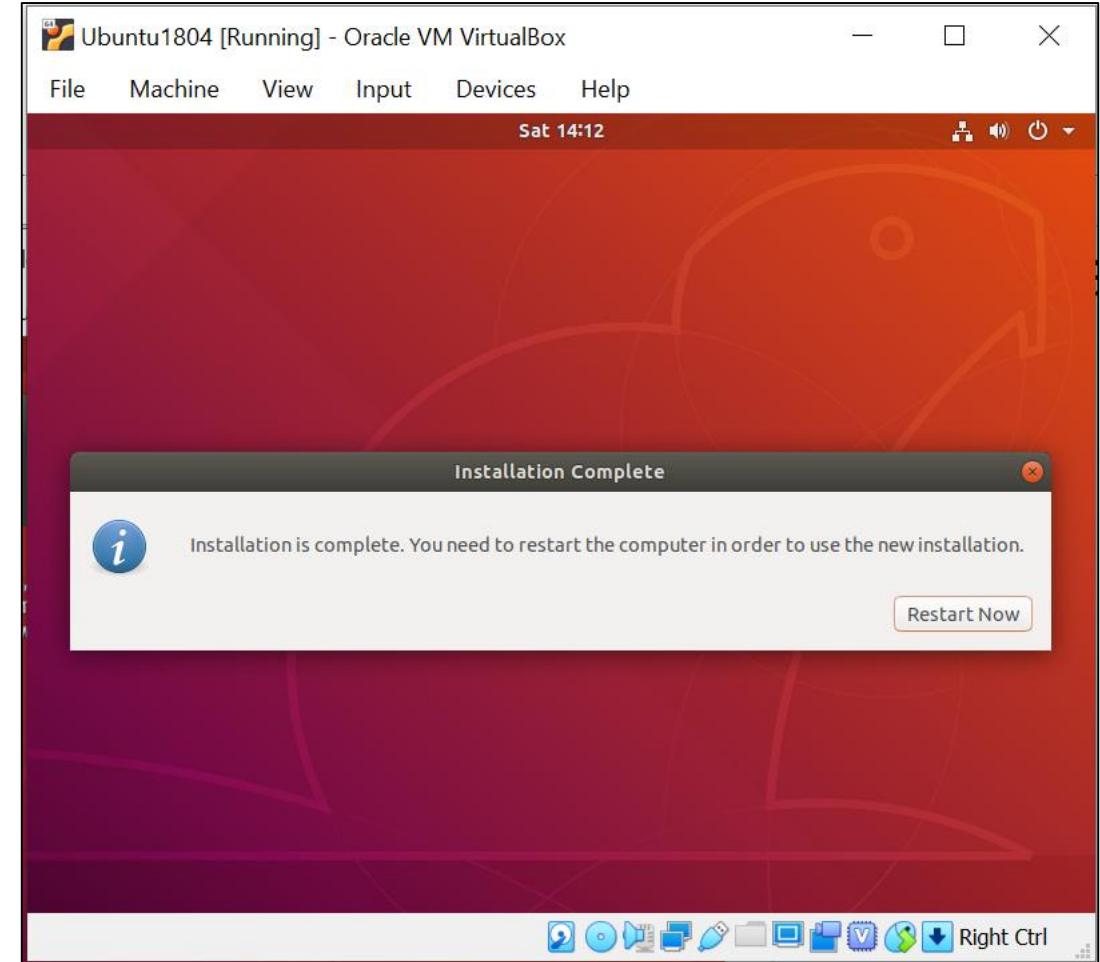
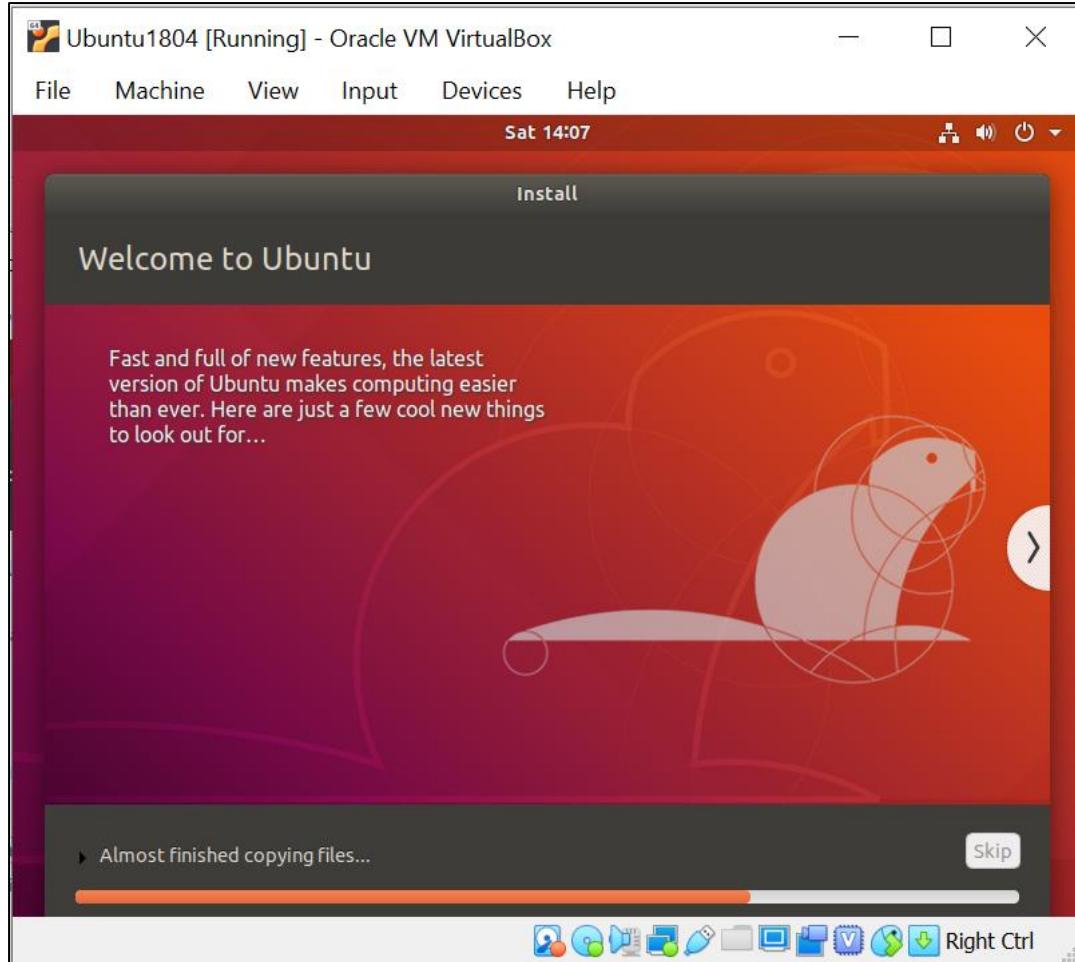
Installing Ubuntu



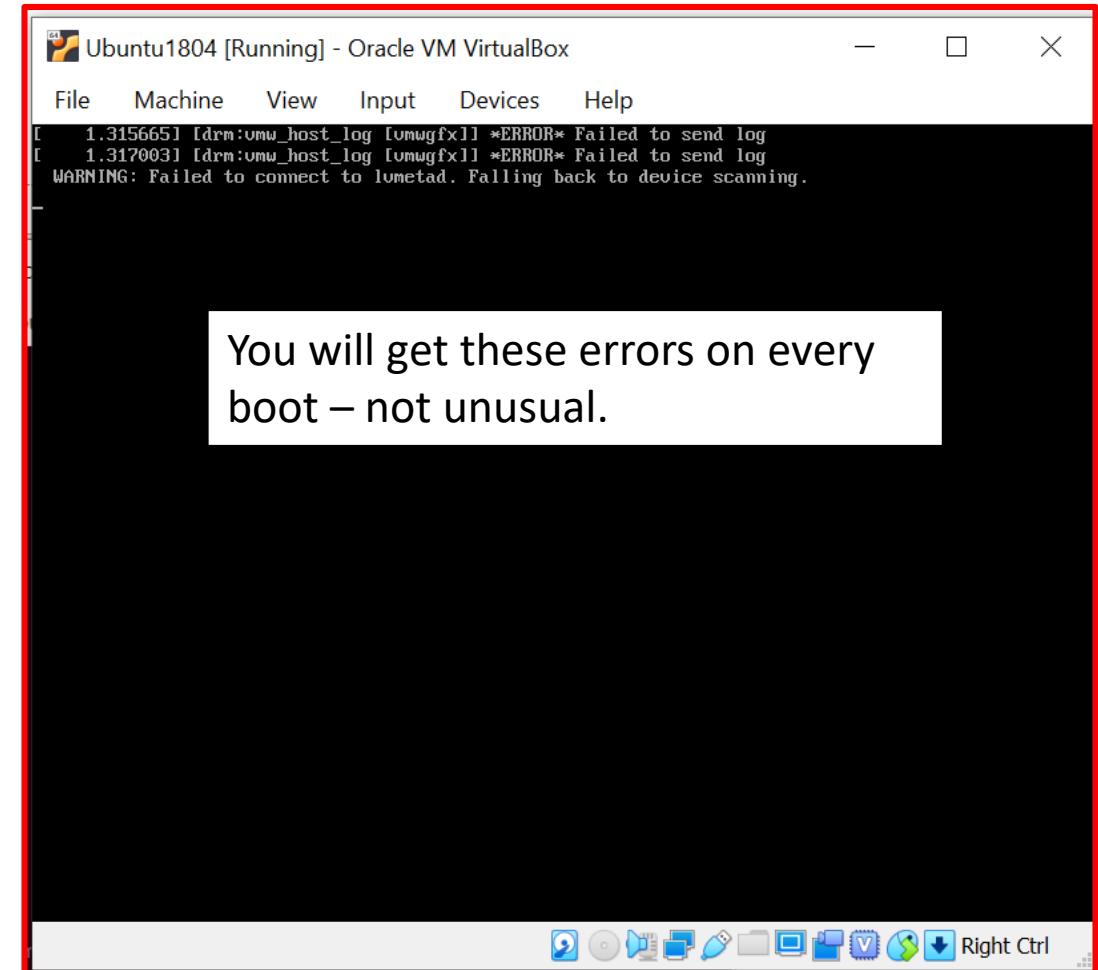
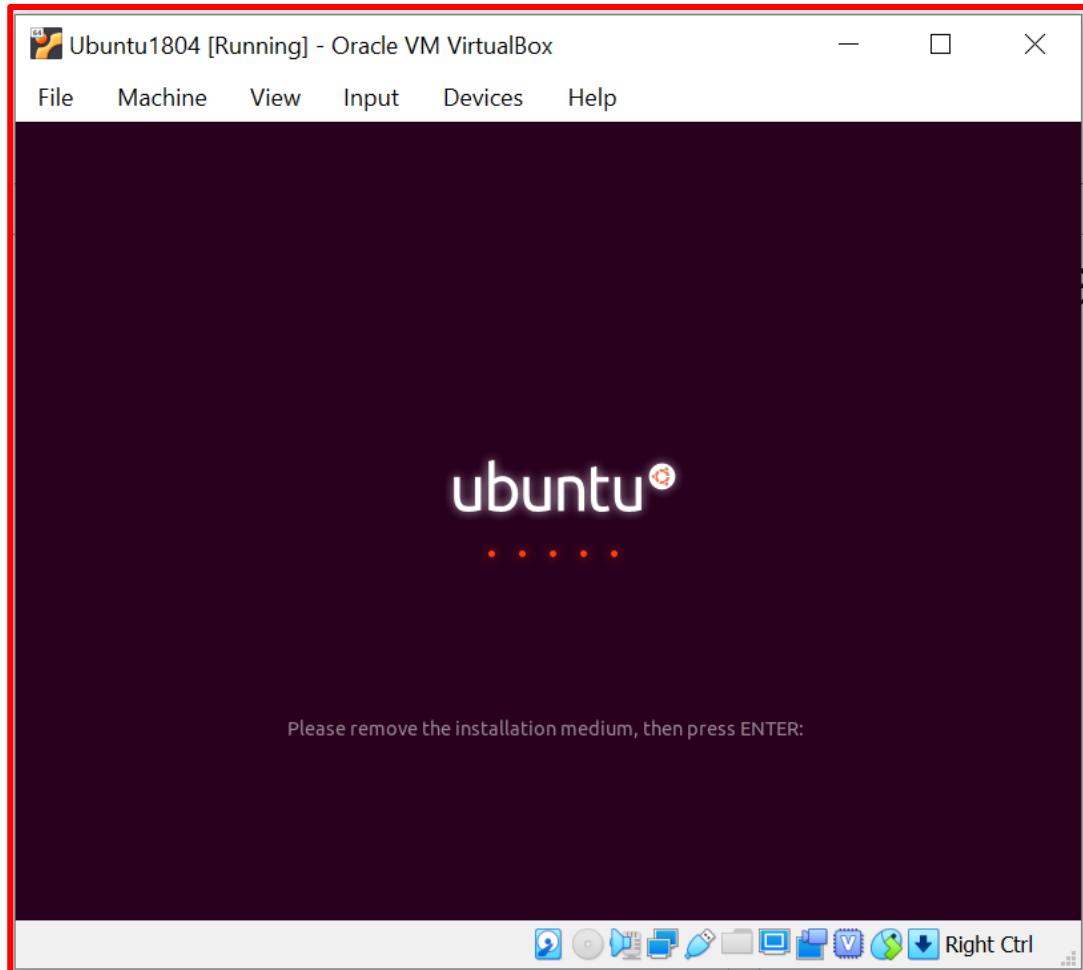
Installing Ubuntu (continued)



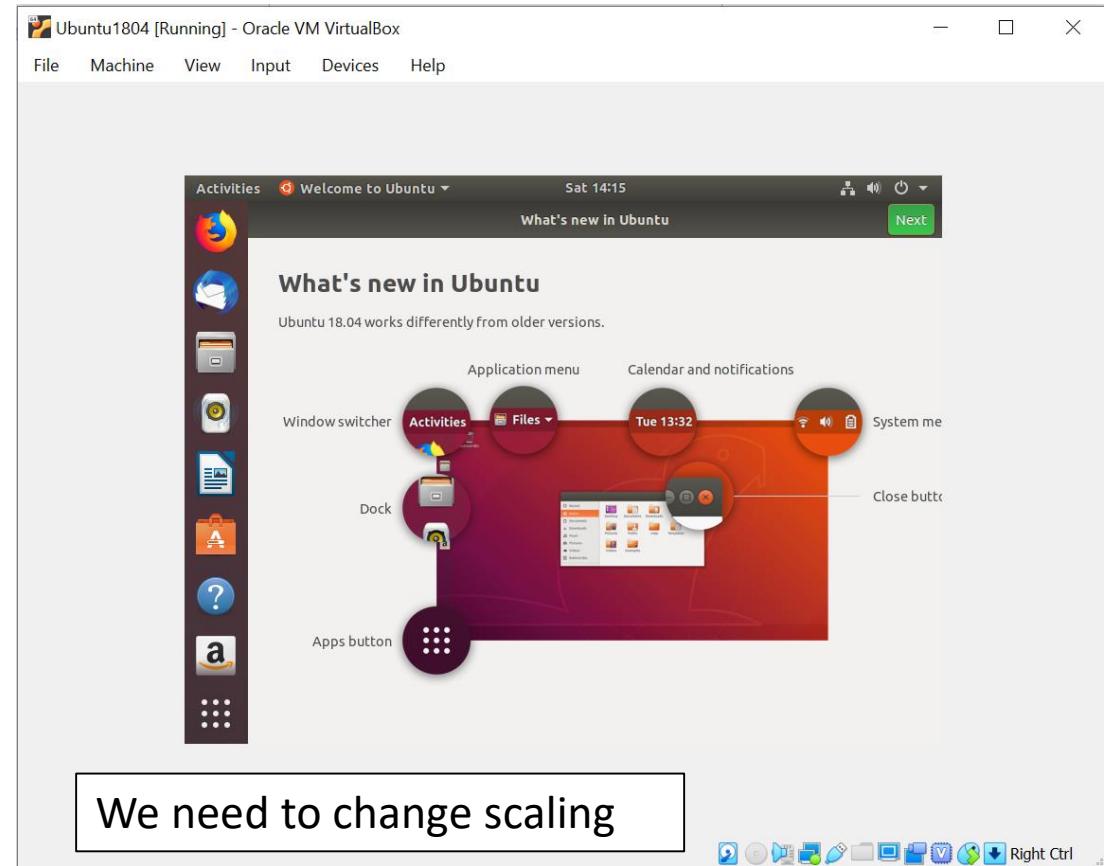
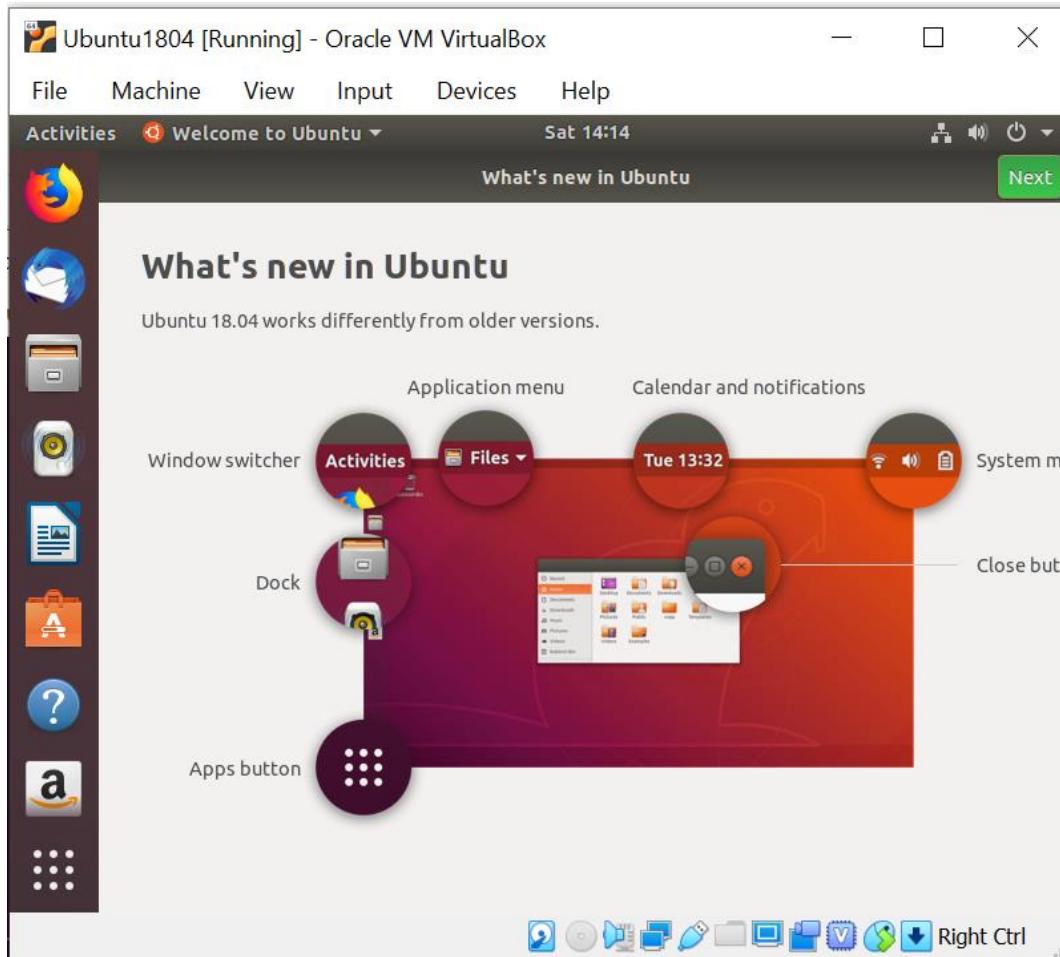
A few minutes later



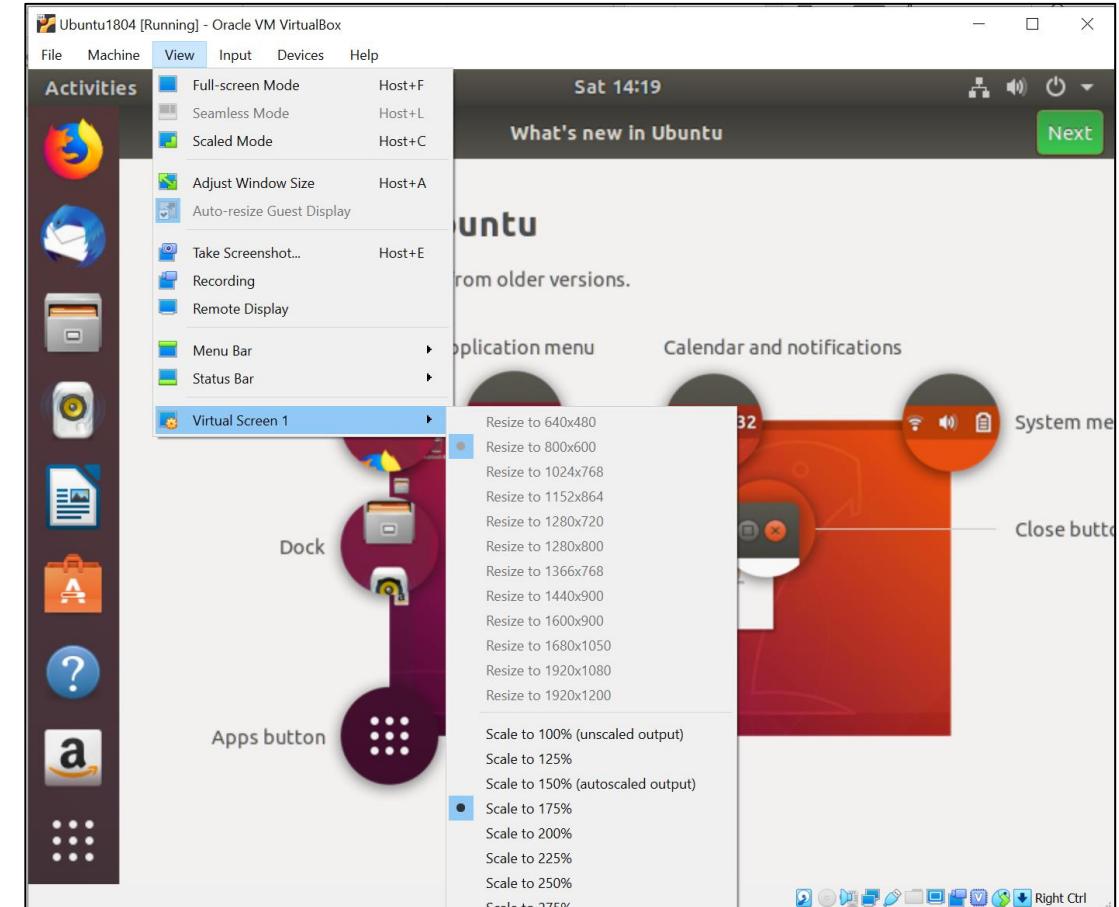
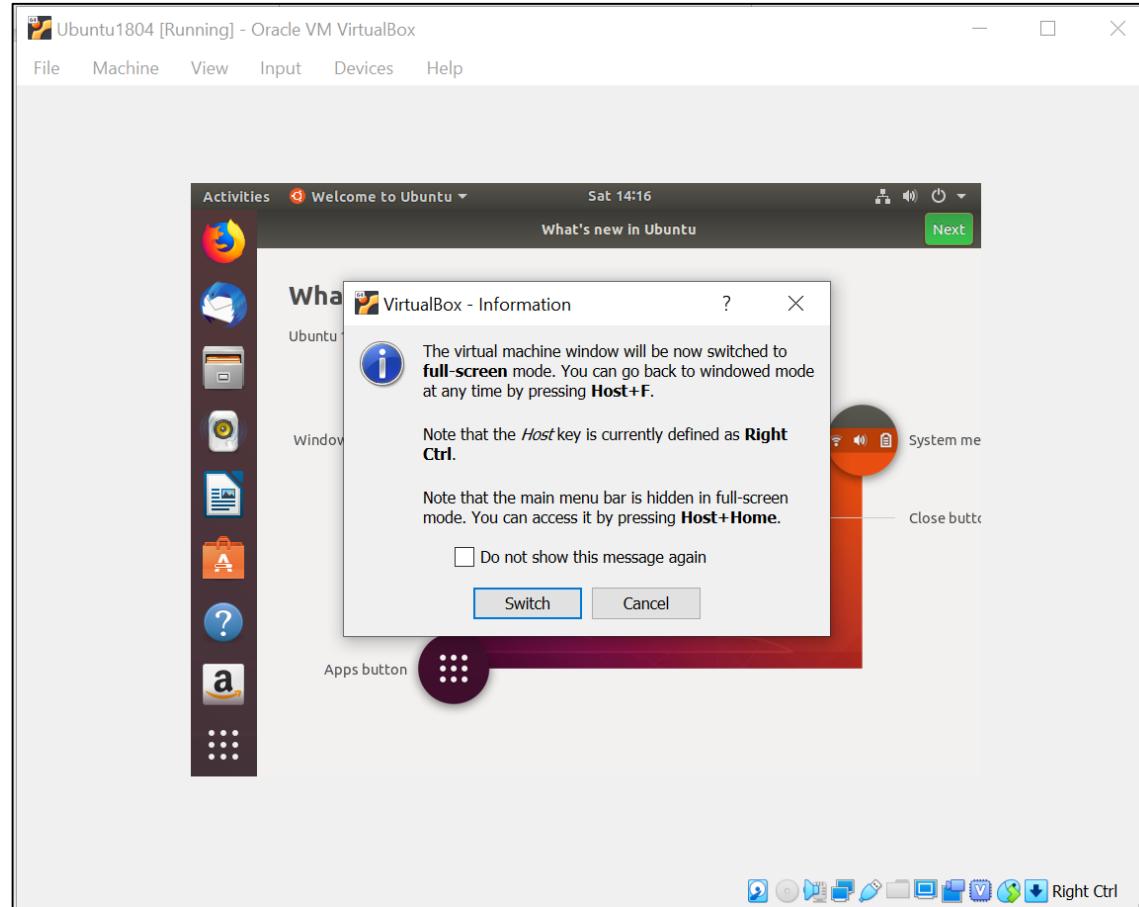
Restarting



Now it's time to configure

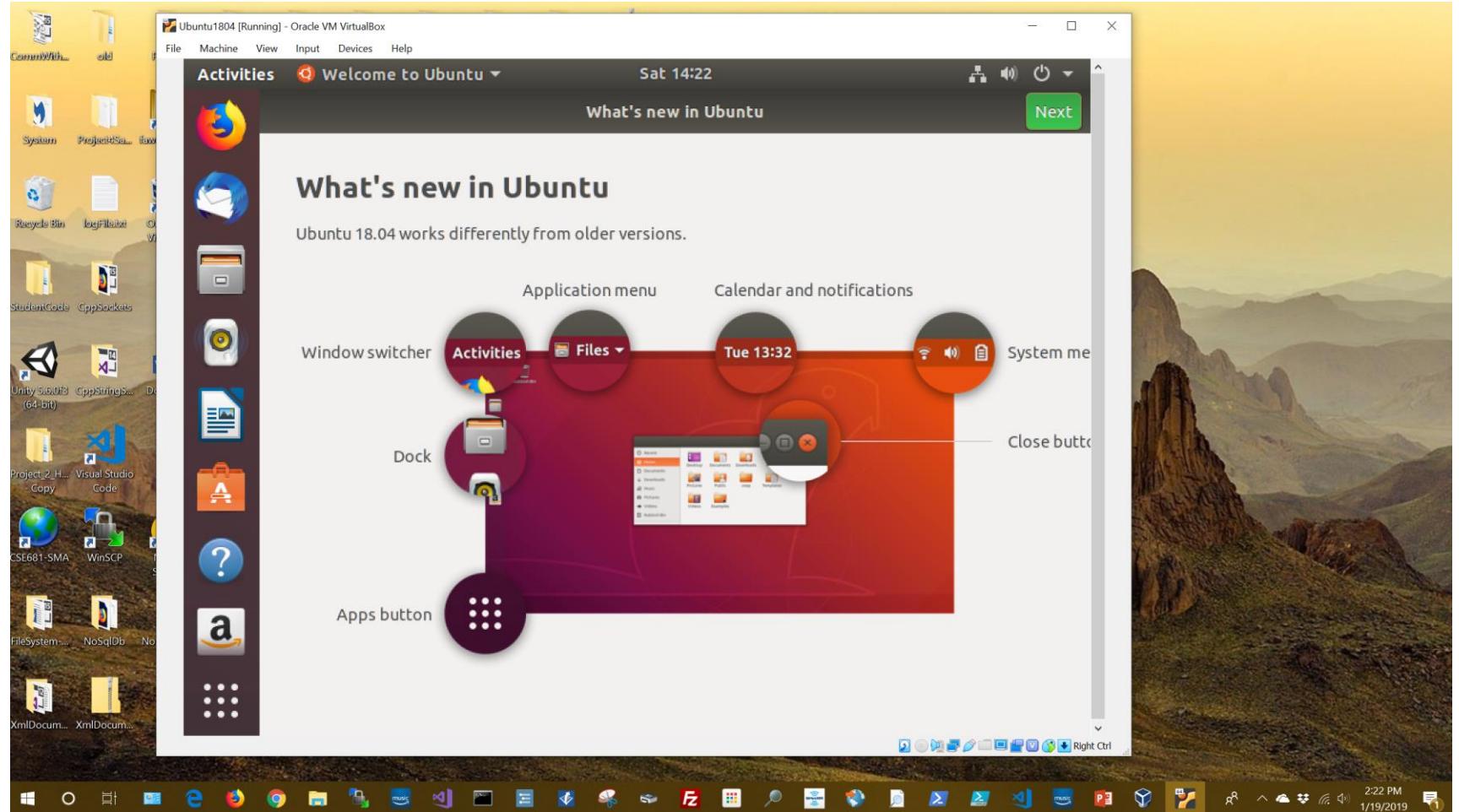


Setting Scaling

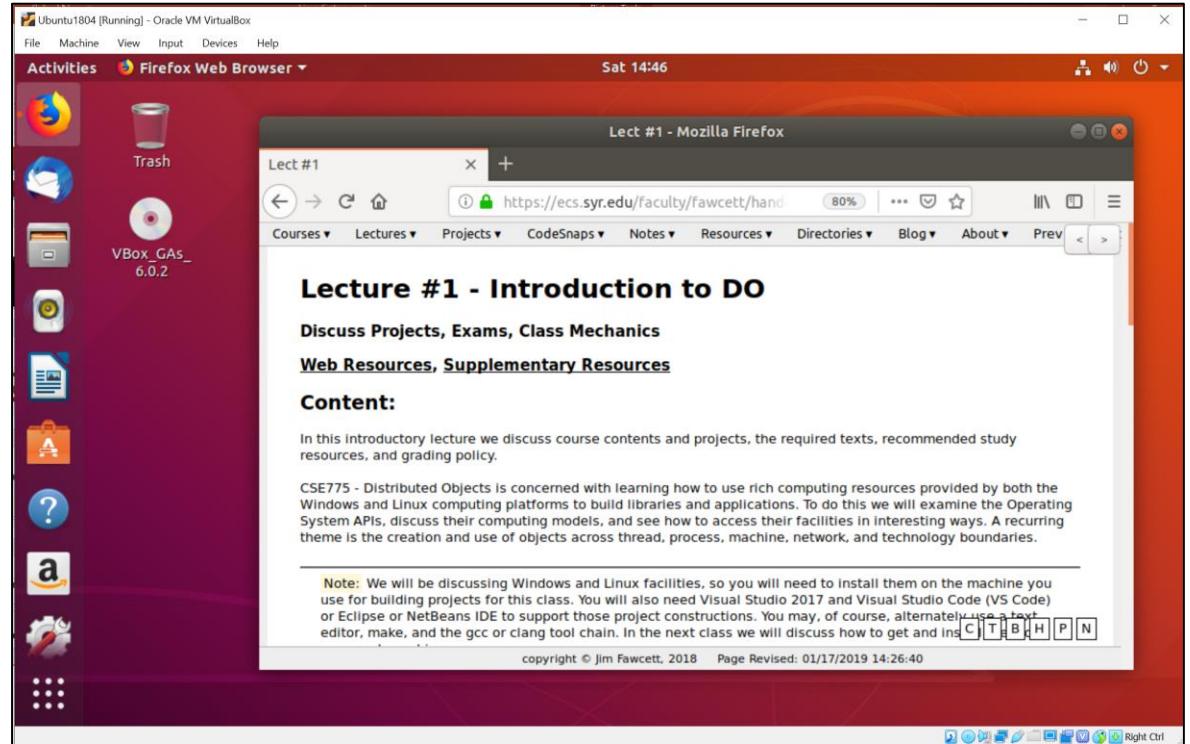
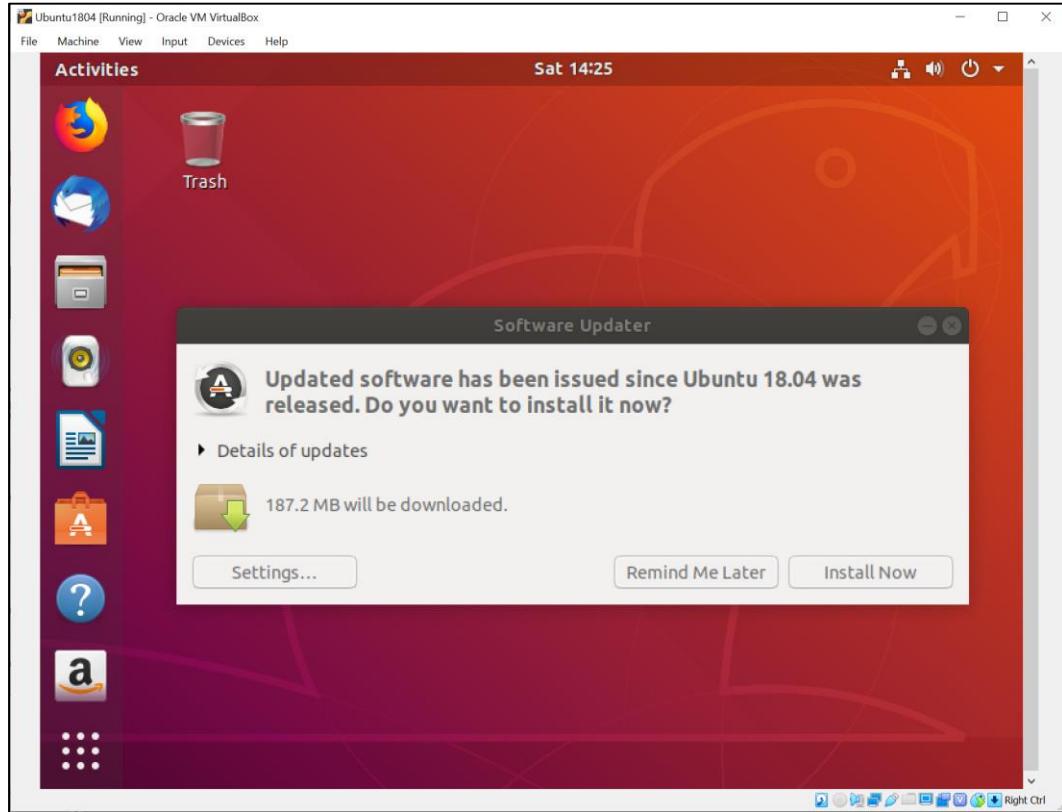


Here's Scaled View

- The view is configured with the VM host, VirtualBox.
- Set menu bar icon sizes with settings > dock in Ubuntu



What do you know - It Works!



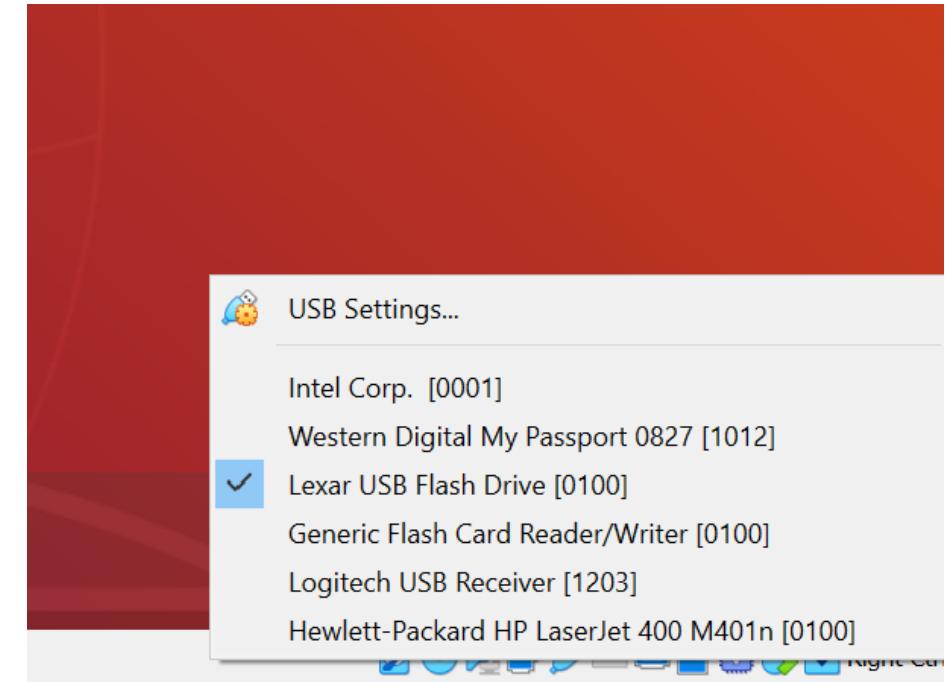
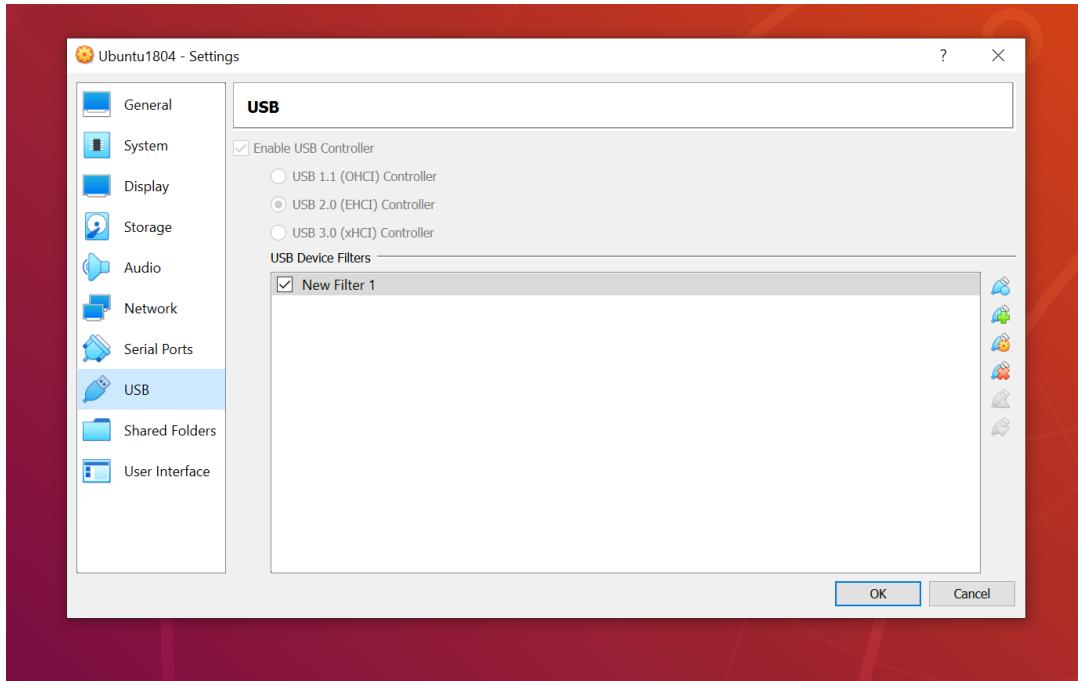
Update Ubuntu Installation using apt pkgmgr

```
jim@jim-VirtualBox: ~
File Edit View Search Terminal Help
To run a command as administrator (user "root"), use "sudo <command>".
See "man sudo_root" for details.

jim@jim-VirtualBox:~$ sudo apt-get update
[sudo] password for jim:
Hit:1 http://us.archive.ubuntu.com/ubuntu bionic InRelease
Get:2 http://security.ubuntu.com/ubuntu bionic-security InRelease [83.2 kB]
Get:3 http://us.archive.ubuntu.com/ubuntu bionic-updates InRelease [88.7 kB]
Get:4 http://us.archive.ubuntu.com/ubuntu bionic-backports InRelease [74.6 kB]
Get:5 http://us.archive.ubuntu.com/ubuntu bionic-updates/main i386 Packages [423 kB]
Get:6 http://us.archive.ubuntu.com/ubuntu bionic-updates/main amd64 Packages [489 kB]
Get:7 http://us.archive.ubuntu.com/ubuntu bionic-updates/universe amd64 Packages [711 kB]
Get:8 http://us.archive.ubuntu.com/ubuntu bionic-updates/universe i386 Packages [702 kB]
Fetched 2,572 kB in 1s (1,965 kB/s)
Reading package lists... Done
jim@jim-VirtualBox:~$
```

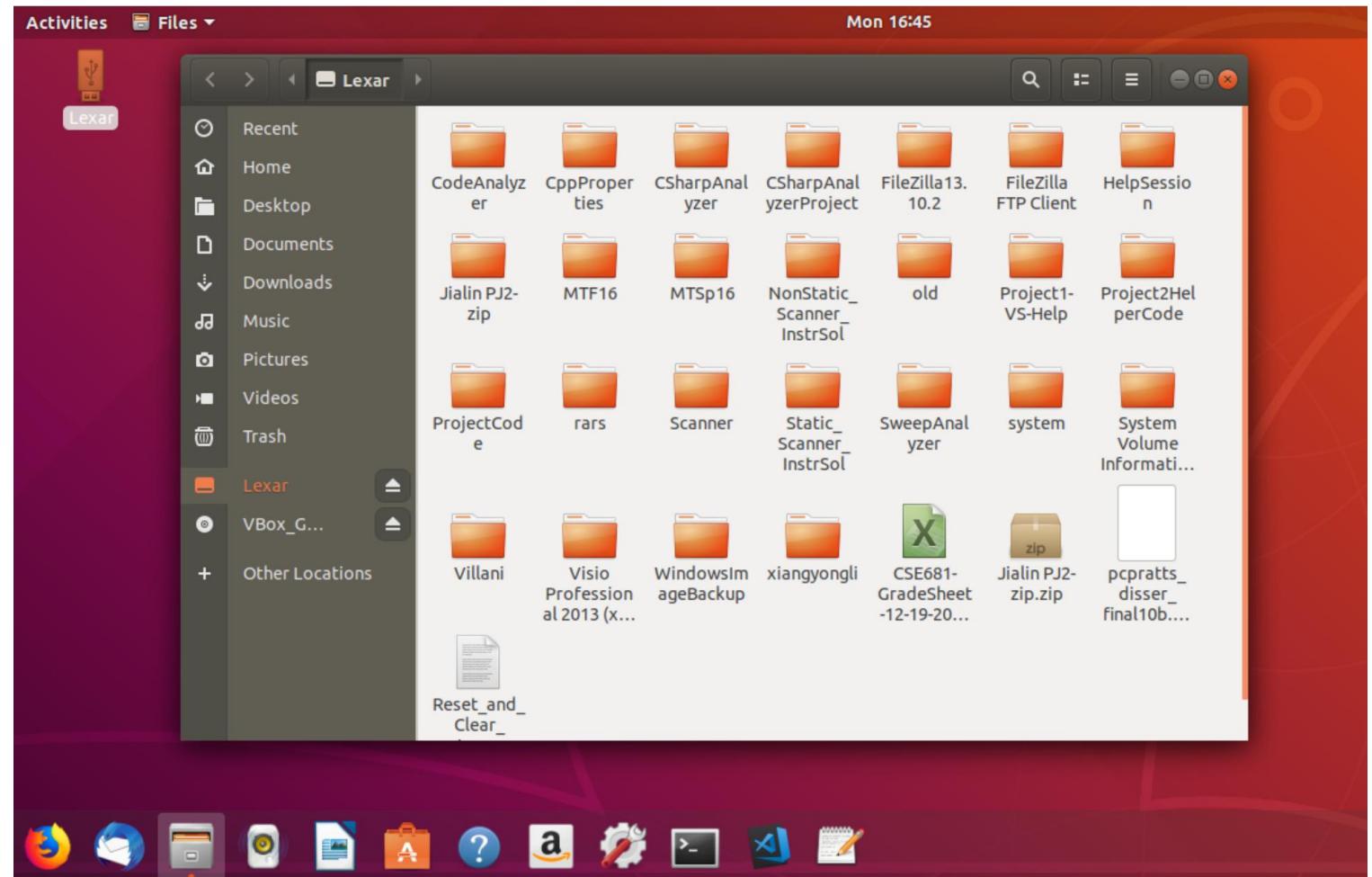
```
jim@jim-VirtualBox: ~
File Edit View Search Terminal Help
jim@jim-VirtualBox:~$ sudo apt-get upgrade
Reading package lists... Done
Building dependency tree
Reading state information... Done
Calculating upgrade... Done
The following packages will be upgraded:
  gnome-software gnome-software-common gnome-software-plugin-snap gvfs-
  gvfs-backends gvfs-bin gvfs-common gvfs-daemons gvfs-fuse gvfs-libs
  ubuntu-software
11 upgraded, 0 newly installed, 0 to remove and 0 not upgraded.
Need to get 0 B/3,711 kB of archives.
After this operation, 7,168 B of additional disk space will be used.
Do you want to continue? [Y/n] █
```

Accessing usb Devices



Using usb devices

- A bug in Virtualbox causes other devices, e.g., Bluetooth mouse, to fail.
- Restarting the VM resolves that problem.
- Large drives will probably fail to connect.



Task #3 – Install build-essential

gcc toolchain configured by Ubuntu team to build Ubuntu

- gcc (c/c++ compiler and linker)
- Make
- Many other tools

Install “build essential” gcc tool chain

```
jim@jim-VirtualBox:~$ sudo apt-get install build-essential
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following additional packages will be installed:
  dpkg-dev fakeroot g++ g++-7 gcc gcc-7 libalgorithm-diff-perl
  libalgorithm-diff-xs-perl libalgorithm-merge-perl libasan4 libatomic1
  libc-dev-bin libc6-dev libcilkrt5 libfakeroot libgcc-7-dev libitm1 liblsan0
  libmpx2 libquadmath0 libstdc++-7-dev libtsan0 libubsan0 linux-libc-dev make
  manpages-dev
Suggested packages:
  debian-keyring g++-multilib g++-7-multilib gcc-7-doc libstdc++-6-7-dbg
  gcc-multilib autoconf automake libtool flex bison gcc-doc gcc-7-multilib
  gcc-7-locales libgcc1-dbg libgomp1-dbg libitm1-dbg libatomic1-dbg
  libasan4-dbg liblsan0-dbg libtsan0-dbg libubsan0-dbg libcilkrt5-dbg
  libmpx2-dbg libquadmath0-dbg glibc-doc libstdc++-7-doc make-doc
The following NEW packages will be installed:
  build-essential dpkg-dev fakeroot g++ g++-7 gcc gcc-7 libalgorithm-diff-perl
  libalgorithm-diff-xs-perl libalgorithm-merge-perl libasan4 libatomic1
  libc-dev-bin libc6-dev libcilkrt5 libfakeroot libgcc-7-dev libitm1 liblsan0
  libmpx2 libquadmath0 libstdc++-7-dev libtsan0 libubsan0 linux-libc-dev make
  manpages-dev
0 upgraded, 27 newly installed, 0 to remove and 0 not upgraded.
```

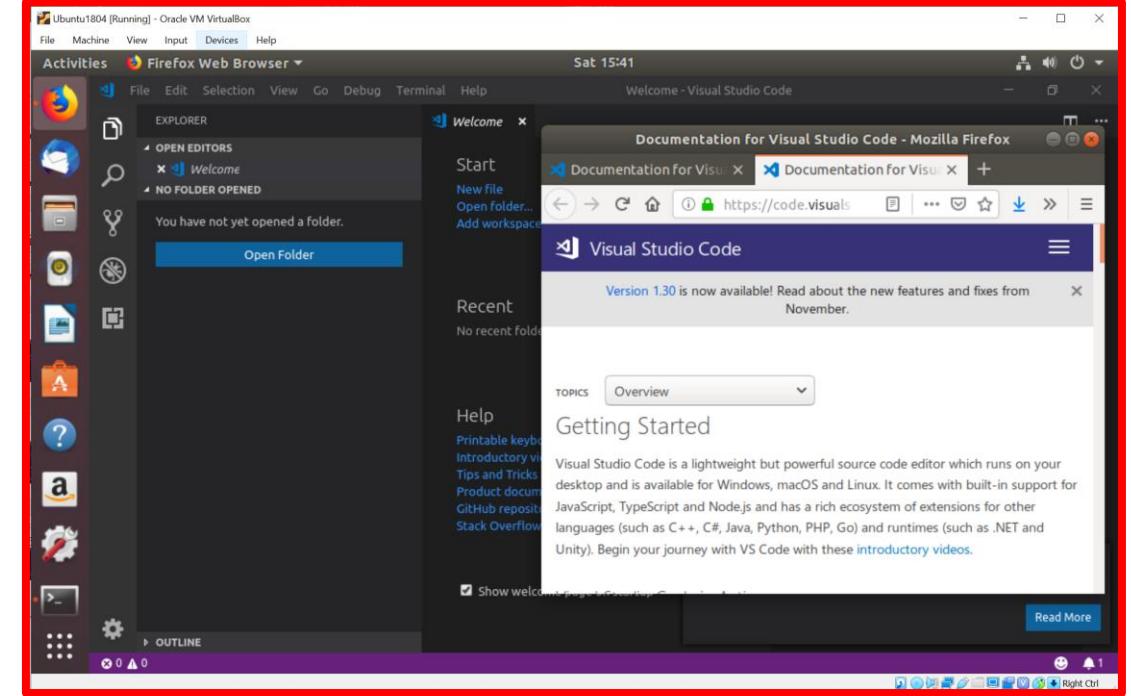
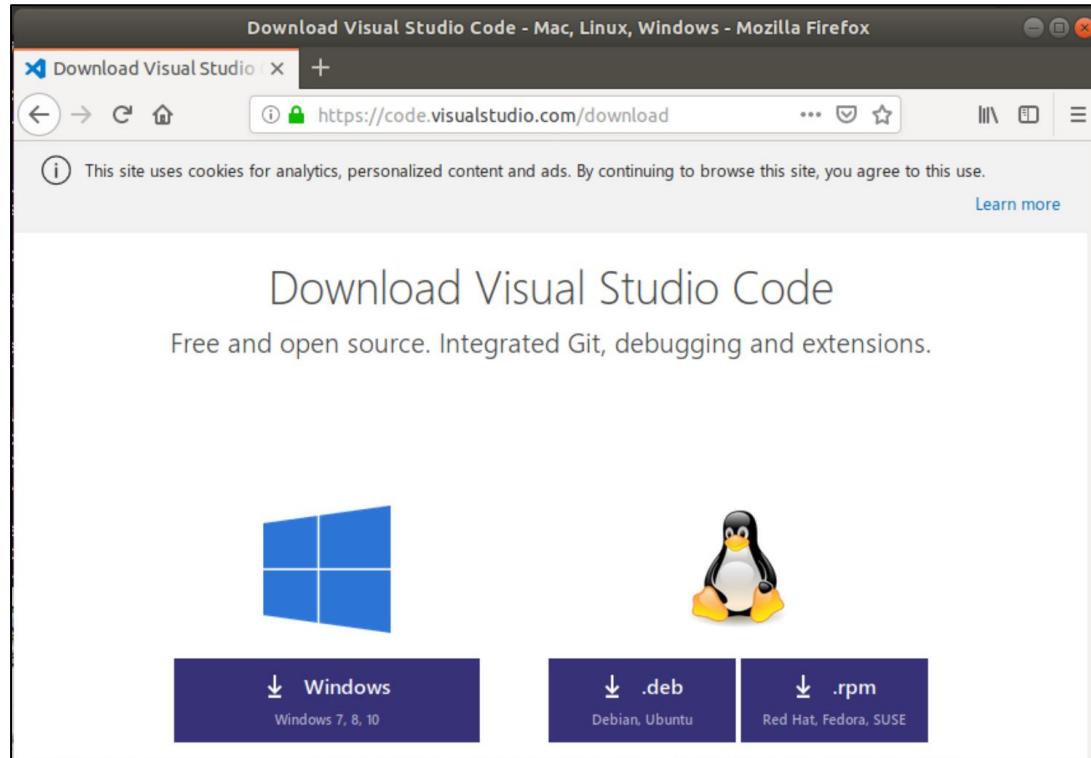
← Press tab

```
jim@jim-VirtualBox:~$ sudo apt-get install build-essential
Setting up libc6-dev:amd64 (2.27-3ubuntu1) ...
Setting up libitm1:amd64 (8.2.0-1ubuntu2~18.04) ...
Setting up fakeroot (1.22-2ubuntu1) ...
update-alternatives: using /usr/bin/fakeroot-sysv to provide /usr/bin/fakeroot (fakeroot) in auto mode
Setting up libgcc-7-dev:amd64 (7.3.0-27ubuntu1~18.04) ...
Setting up libstdc++-7-dev:amd64 (7.3.0-27ubuntu1~18.04) ...
Setting up libalgorithm-merge-perl (0.08-3) ...
Setting up libalgorithm-diff-xs-perl (0.04-5) ...
Setting up gcc-7 (7.3.0-27ubuntu1~18.04) ...
Setting up g++-7 (7.3.0-27ubuntu1~18.04) ...
Setting up gcc (4:7.3.0-3ubuntu2.1) ...
Setting up g++ (4:7.3.0-3ubuntu2.1) ...
update-alternatives: using /usr/bin/g++ to provide /usr/bin/c++ (c++) in auto mode
Setting up build-essential (12.4ubuntu1) ...
Processing triggers for libc-bin (2.27-3ubuntu1) ...
jim@jim-VirtualBox:~$ gcc --version
gcc (Ubuntu 7.3.0-27ubuntu1~18.04) 7.3.0
Copyright (C) 2017 Free Software Foundation, Inc.
This is free software; see the source for copying conditions. There is NO
warranty; not even for MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.

jim@jim-VirtualBox:~$
```

Task #4 – Install VS Code

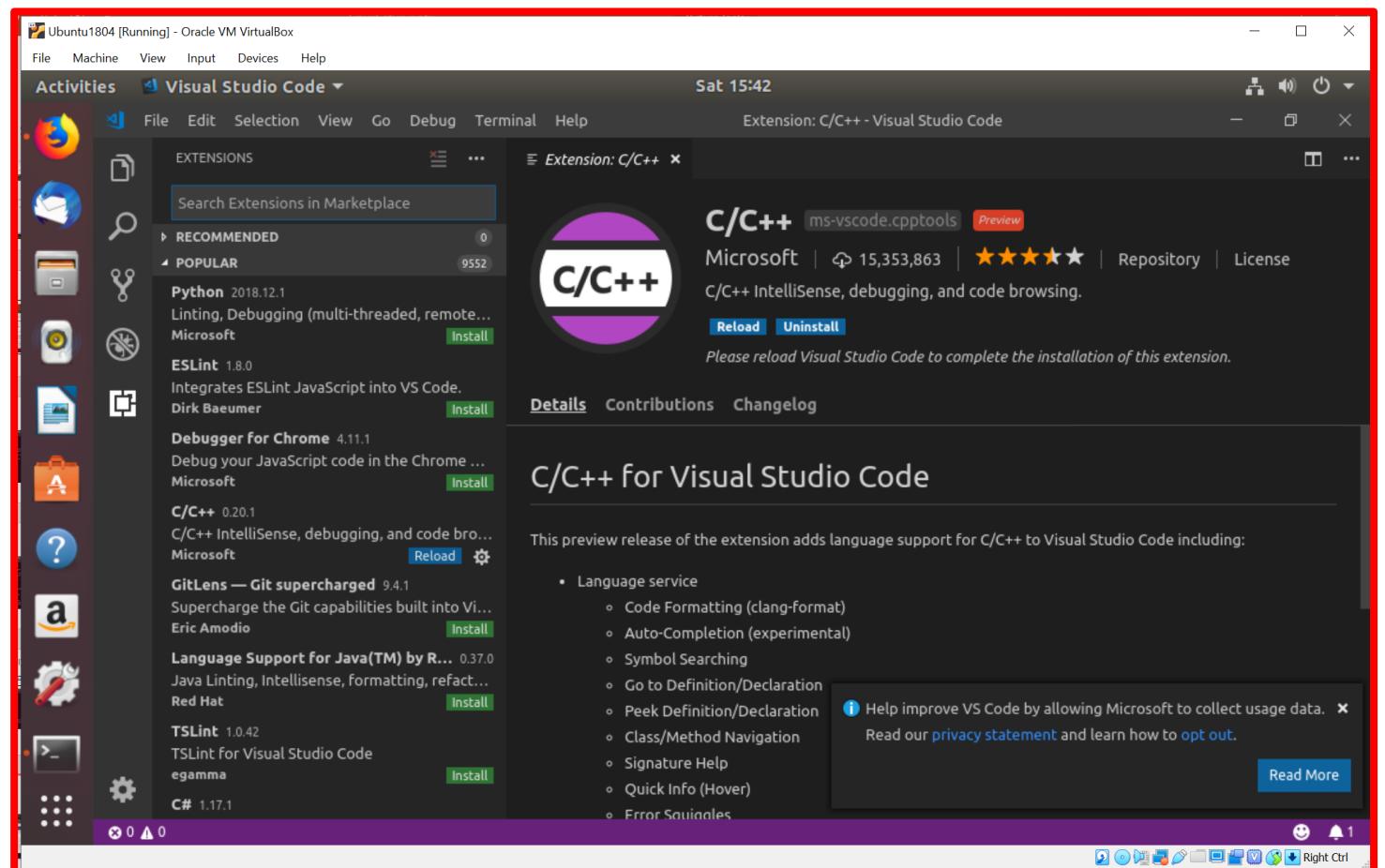
Download and Install Visual Studio Code



Follow directions on this page.
You can't just apt-get install.

Adding C++ Plugin

- Click on plugin icon
 - Bottom of left pane
- Select C/C++
- This configures json setting files (partially)
- You must already have a C++ tool chain installed



VS Code – Building C++

This screenshot shows the Visual Studio Code interface for a C++ project named "Hello". The Explorer sidebar shows files like "launch.json", "tasks.json", and "Hello.cpp". The "tasks.json" editor is open, displaying the following JSON configuration:

```
// See https://go.microsoft.com/fwlink/?LinkId=733558
// for the documentation about the tasks.json format
"version": "2.0.0",
"tasks": [
    {
        "label": "gcc build",
        "type": "shell",
        "command": "g++",
        "args": [
            "-g", "Hello.cpp"
        ],
        "group": {
            "kind": "build",
            "isDefault": true
        },
        "problemMatcher": [
            "$gcc"
        ]
    }
]
```

The terminal window shows the command being executed: `> Executing task: g++ -g Hello.cpp <`. A tooltip "gcc build" is visible over the terminal tab.

This screenshot shows the Visual Studio Code interface for the same "Hello" project. The Explorer sidebar shows files like "launch.json", "tasks.json", and "Hello.cpp". The "launch.json" editor is open, displaying the following JSON configuration:

```
"type": "cppdbg",
"request": "launch",
"program": "${workspaceFolder}/a.out",
"args": [],
"stopAtEntry": false,
"cwd": "${workspaceFolder}",
"environment": [],
"externalConsole": false,
"miMode": "gdb",
"miDebuggerPath": "/usr/bin/gdb",
"setupCommands": [
    {
        "description": "Enable pretty-printing 1",
        "text": "enable-pretty-printing",
        "ignoreFailures": true
    }
]
```

The terminal window shows the command being executed: `Hello World!` and `[1] + Done /usr/bin/gdb --interpreter=mi --tty=${DbgT erm} 0</tmp/Microsoft-MIEngine-In-yhwpf0pd.gpe 1>/tmp/Microsoft-MIEngine-Ou t-yingchvhc.s5c`. A tooltip "gdb execution" is visible over the terminal tab.

Tasks.json and Launch.json

```
{  
    // See https://go.microsoft.com/fwlink/?LinkId=733558  
    // for the documentation about the tasks.json format  
    "version": "2.0.0",  
    "tasks": [  
        {  
            "label": "gcc build",  
            "type": "shell",  
            "command": "g++",  
            "args": [  
                "-g", "Hello.cpp"  
            ],  
            "group": {  
                "kind": "build",  
                "isDefault": true  
            },  
            "problemMatcher": [  
                "$gcc"  
            ]  
        }  
    ]  
}
```

Build Task

```
{  
    // Use IntelliSense to learn about possible attributes.  
    // Hover to view descriptions of existing attributes.  
    // For more information, visit: https://go.microsoft.com/fwlink/?LinkId=733558  
    "version": "0.2.0",  
    "configurations": [  
        {  
            "name": "(gdb) Launch",  
            "type": "cppdbg",  
            "request": "launch",  
            "program": "${workspaceFolder}/a.out",  
            "args": [],  
            "stopAtEntry": false,  
            "cwd": "${workspaceFolder}",  
            "environment": [],  
            "externalConsole": false,  
            "MIMode": "gdb",  
            "miDebuggerPath": "/usr/bin/gdb",  
            "setupCommands": [  
                {  
                    "description": "Enable pretty-printing for gdb",  
                    "text": "-enable-pretty-printing",  
                    "ignoreFailures": true  
                }  
            ]  
        }  
    ]  
}
```

Debugger Launch

Add Configuration...

Makefile Example

```
// Hello.cpp

#include <iostream>
#include <string>

std::string makeString()
{
    std::string str = "I am a string";
    str += " with some appended text";
    return str;
}
int main()
{
    std::cout << "\n  Hello World!\n\n";
    std::cout << "\n  " << makeString();
    std::cout << "\n\n";
}
```

```
-----
# makefile - demo for Project #1
#
# Jim Fawcett, CSE775 - Distributed Objects, Spring 2019
#
# Notes:
#   - Indentations must be a single Tab (not spaces)
#   - Dependencies are not indented
#   - commands are indented with single tab
#

all: hello clean

# link hello.o to create executable hello
# you may add additional object files as needed

hello: hello.o
        g++ hello.o -o hello

# compile Hello.cpp to create object file hello.o
# you may add additional source files as needed

hello.o: Hello.cpp
        g++ -c -g -Wall Hello.cpp -o hello.o

# remove object files
# only called if cited in all: directive

clean:
        rm *.o
```

Running make

- Terminal >
Run Task >
make
- Builds application
as specified by
makefile

The screenshot shows a Visual Studio Code interface running on a Linux desktop. The title bar indicates the window is titled "Ubuntu1804 (ToolInstallationFinished) [Running] - Oracle VM VirtualBox". The main area displays a C++ file named "Hello.cpp" with the following code:

```
#include <iostream>
#include <string>

std::string makeString()
{
    std::string str = "I am a string";
    str += " with some appended text";
    return str;
}

int main()
{
    std::cout << "\n Hello World!\n\n";
    std::cout << "\n " << makeString();
    std::cout << "\n\n";
}
```

The terminal panel at the bottom shows the output of the "make" command:

```
> Executing task: make <
g++ -c -g -Wall Hello.cpp -o hello.o
g++ hello.o -o hello
rm *.o

Terminal will be reused by tasks, press any key to close it.
```

The status bar at the bottom right shows the current file is "makeString()", has 7 lines and 1 space, and is in C++ mode for Linux.

Debugging hello: Debug > Start Debugging

- Note breakpoint
- F10 => single step
- F5 => go to next breakpoint
- Note call stack
- Note Terminal

The screenshot shows a Visual Studio Code interface with a red border, running on an Ubuntu 18.04 VM. The code editor displays a file named Hello.cpp with the following content:

```
#include <iostream>
#include <string>

std::string makeString()
{
    std::string str = "I am a string";
    str += " with some appended text";
    return str;
}

int main()
{
    std::cout << "\n Hello World!\n\n";
    std::cout << "\n " << makeString();
    std::cout << "\n\n";
}
```

The debugger sidebar on the left shows the following status:

- VARIABLES**: Shows a local variable `str` with the error message: `<error: Cannot access memory...`.
- WATCH**: No items listed.
- BREAKPOINTS**: A single breakpoint is set at line 14 of Hello.cpp.
- CALL STACK**: Shows the current stack trace:

```
makeString[abi:cxx11]()() Hello.cpp
main() Hello.cpp [15:1]
```

The terminal pane at the bottom right shows the output: `Hello World!`. The status bar at the bottom indicates the current file is `makeString()` at line 7, column 1.

```
// See https://go.microsoft.com/fwlink/?LinkId=733558
// for the documentation about the tasks.json format
"version": "2.0.0",
"tasks": [
    {
        "label": "gcc build",
        "type": "shell",
        "command": "g++",
        "args": [
            "-g", "Hello.cpp"
        ],
        "group": {
            "kind": "build",
            "isDefault": true
        },
        "problemMatcher": [
            "$gcc"
        ]
    },
    {
        "label": "make",
        "type": "shell",
        "command": "make",
        "args": [
        ],
        "group": {
            "kind": "build",
            "isDefault": true
        },
        "problemMatcher": [
            "$gcc"
        ]
    }
]
```

gcc build

make build

Build and Launch JSON

```
// Use IntelliSense to learn about possible attributes.
// Hover to view descriptions of existing attributes.
// For more information, visit: https://go.microsoft.com/fwlink/?LinkId=830387
"version": "0.2.0",
"configurations": [
    {
        "name": "(gdb) Launch",
        "type": "cppdbg",
        "request": "launch",
        "program": "${workspaceFolder}/hello",
        "args": [],
        "stopAtEntry": false,
        "cwd": "${workspaceFolder}",
        "environment": [],
        "externalConsole": false,
        "MIMode": "gdb",
        "miDebuggerPath": "/usr/bin/gdb",
        "setupCommands": [
            {
                "outfiles": true,
                "description": "Enable pretty-printing for gdb",
                "text": "-enable-pretty-printing",
                "ignoreFailures": true
            }
        ]
    }
]
```

Launch gdb
debugger

Task #5 – Optional Installs

Download Asp.Net Core 2.2 (optional)

<https://dotnet.microsoft.com/download/linux-package-manager/ubuntu18-04/sdk-2.2.102>

Several detailed command line invocations are needed.

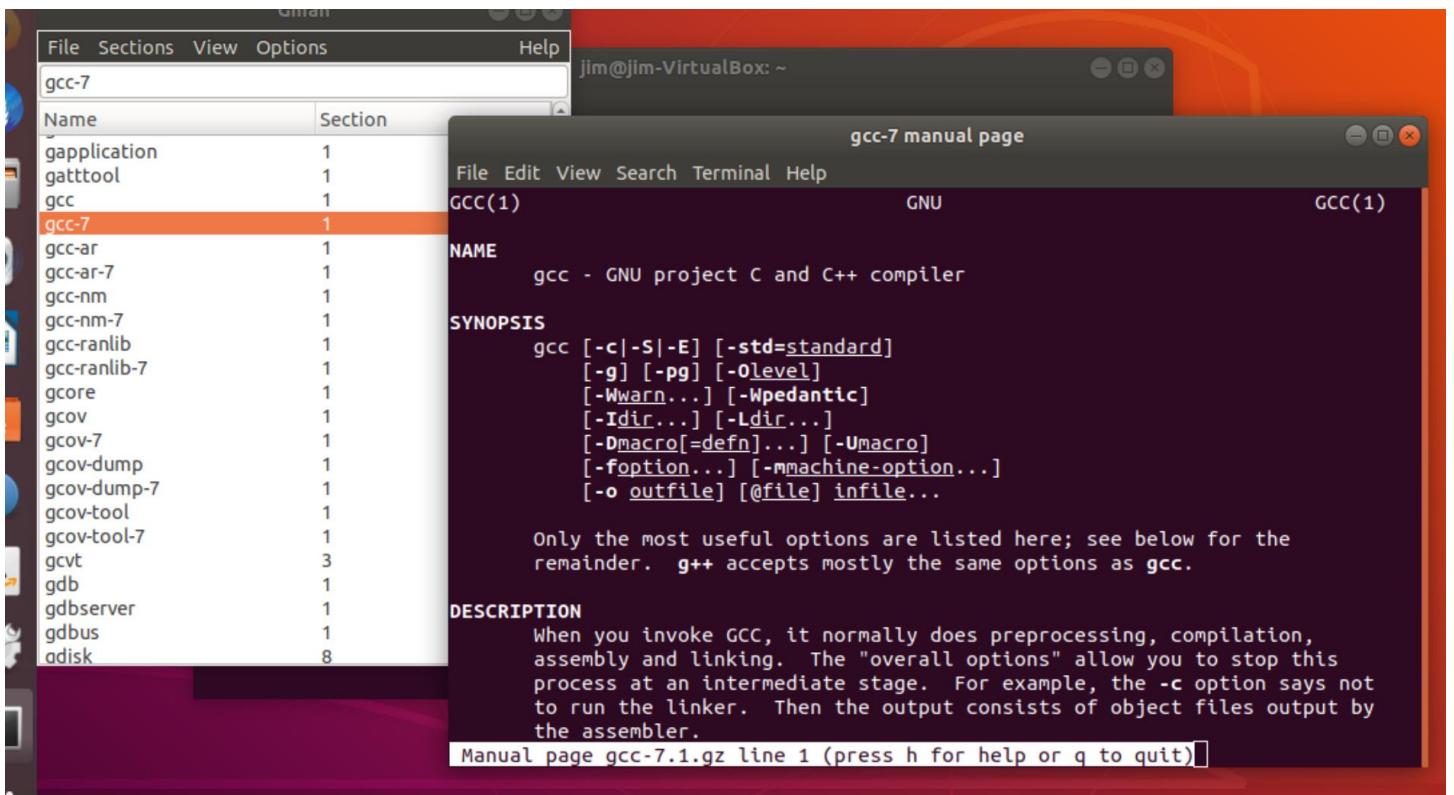
The screenshot shows the .NET Core 2.2 downloads page for Linux, macOS, and Windows. The URL in the address bar is https://dotnet.microsoft.com/download/dotnet-core/2.2.103. The page features a Microsoft logo and a purple header with the text ".NET Core 2.2 downloads". Below the header, there are two promotional banners: one for ML.NET and another for .NET Core Preview 27122-01. A "Feedback" button is visible at the bottom right of the page.

The screenshot shows a terminal window titled "jim@jim-VirtualBox: ~". The terminal displays the following information:

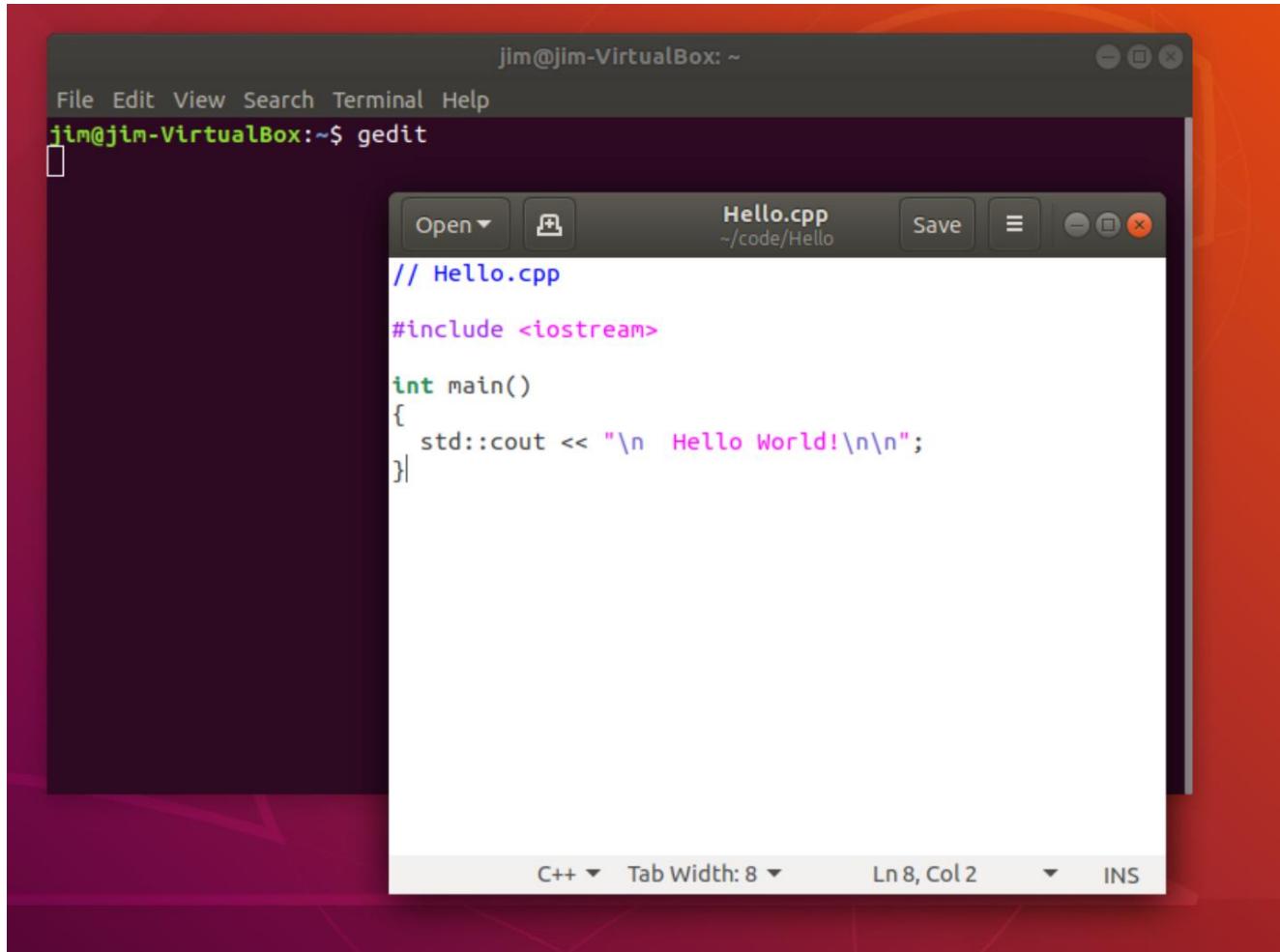
```
Commit: 8edbc2570a
Runtime Environment:
OS Name: ubuntu
OS Version: 18.04
OS Platform: Linux
RID: ubuntu.18.04-x64
Base Path: /usr/share/dotnet/sdk/2.2.103/
Host (useful for support):
Version: 2.2.1
Commit: 878dd11e62
.NET Core SDKs installed:
2.2.103 [/usr/share/dotnet/sdk]
.NET Core runtimes installed:
Microsoft.AspNetCore.All 2.2.1 [/usr/share/dotnet/shared/Microsoft.AspNetCore.All]
Microsoft.AspNetCore.App 2.2.1 [/usr/share/dotnet/shared/Microsoft.AspNetCore.App]
Microsoft.NETCore.App 2.2.1 [/usr/share/dotnet/shared/Microsoft.NETCore.App]
To install additional .NET Core runtimes or SDKs:
https://aka.ms/dotnet-download
jim@jim-VirtualBox:~$
```

Gman – man page helper

- Install gman
 - Sudo apt install gman



gedit – installed with Ubuntu



That's All Folks!