



# SOFE 3950U / CSCI 3020U: Operating Systems

## Lab #1: Linux Setup Version 2.0

### Objectives

- To install a Linux-based operating system on your computer
- To install a C compiler and code editor in the Linux environment
- To create a GitHub account and learn how to publish projects to GitHub

### Important Notes

- Work in groups of **three** students
- All reports must be submitted as a PDF on blackboard, if source code is included submit everything as an archive (e.g. zip, tar.gz)
- Save the file as <lab\_number>\_<first student's id>.pdf (e.g. lab1\_100123456.pdf)
- If you cannot submit the document on Blackboard then please contact the TA (Pat Smuk) with your submission on slack at <http://sofe3950u.slack.com> or send him an email via [patrick.smuk@uoit.net](mailto:patrick.smuk@uoit.net).

# Deliverables

This lab is merely a setup lab to ensure that everyone is ready to begin work on future assignments. For full marks you must demonstrate that...

1. You have a working Linux environment.
2. You can edit and compile C code.
3. You can publish your C project and commits to GitHub.

**Please show the lab TA your setup before leaving. There is no report for this lab.**

**Note that you do not have to use the exact software recommended in this manual.**

You may use any Linux distribution you want, installed in any fashion you prefer, with your favorite text editor/IDE and compiler.

## Linux Setup

**OPTIONAL:** If you would prefer to dual boot Linux rather than use it within a virtual machine environment, then please download the following latest version of Xubuntu. It is recommended that you follow the instructions in the guide in the link below before attempting to install Xubuntu as the install process can **erase all of your existing data** if done incorrectly. If you are not comfortable with installing Linux then it is recommended that you **do not try to dual boot Linux**, and rather follow the instructions below using Linux in a virtual machine.

<http://mirror.us.leaseweb.net/ubuntu-cdimage/xubuntu/releases/16.04/release/xubuntu-16.04.1-desktop-amd64.iso>

<http://www.pcsteps.com/961-install-ubuntu-linux-windows/>

If you are able to install Linux as a dual boot then please see final step of the virtual machine instructions to install all of the necessary software (clang, lldb, etc.) required for the course.

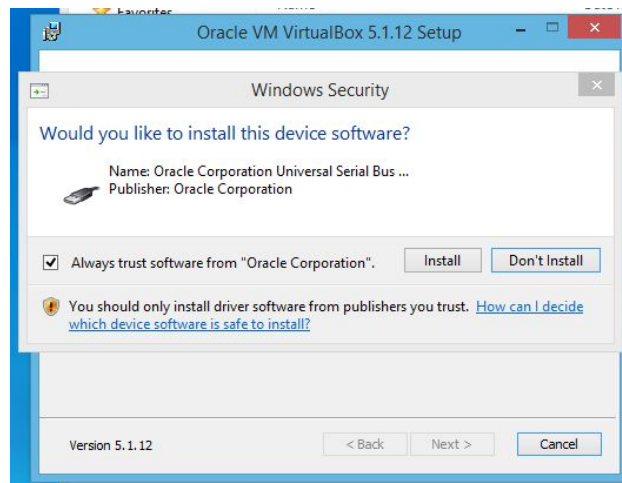
# Virtual Machine Instructions

1. Make sure that you download the following two files.

<http://download.virtualbox.org/virtualbox/5.1.12/VirtualBox-5.1.12-112440-Win.exe>

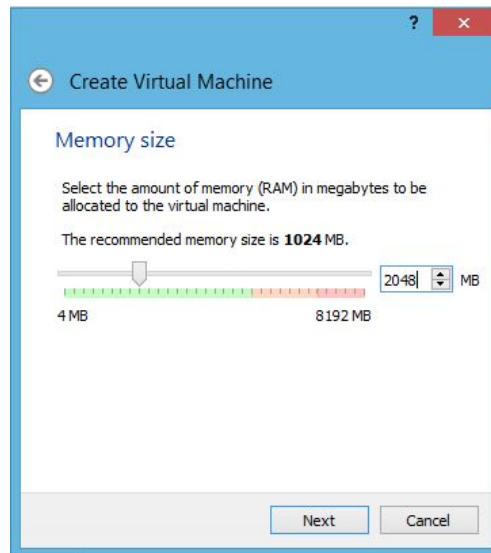
[https://docs.google.com/uc?authuser=0&id=0B\\_HAFnYs6Ur-d2ZNRWJvWTByTU0&export=download](https://docs.google.com/uc?authuser=0&id=0B_HAFnYs6Ur-d2ZNRWJvWTByTU0&export=download)

2. Install virtualbox just with the default installation options, make sure when prompted that you install the drivers.

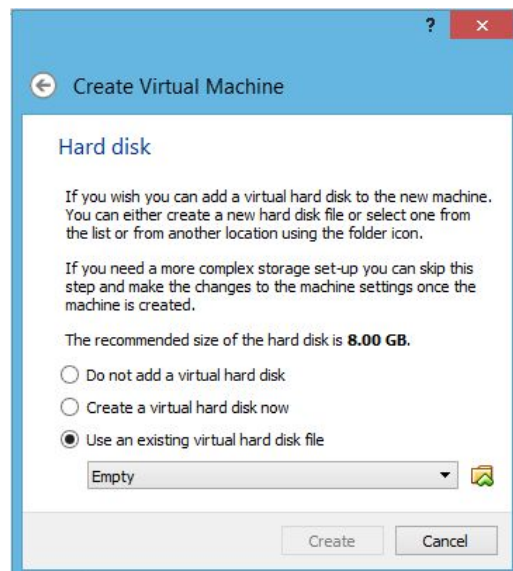


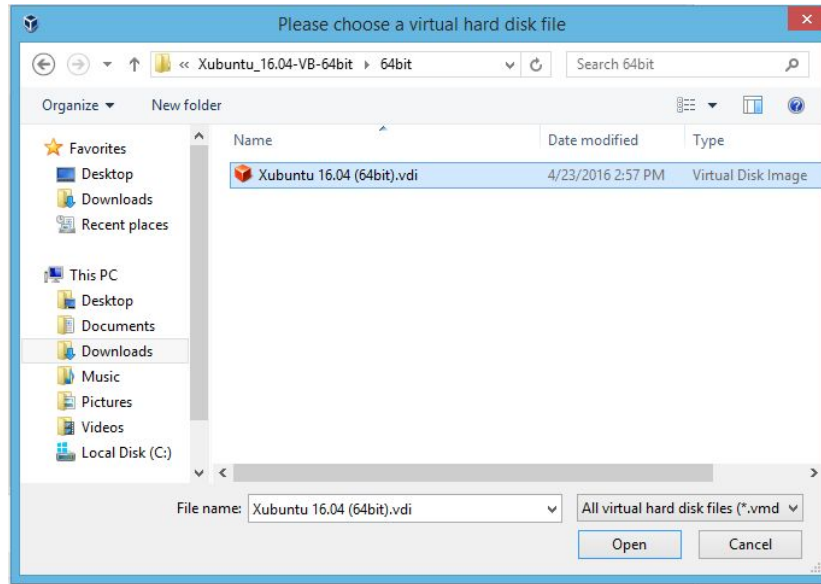
3. Extract the virtualbox Xubuntu image using 7zip, use the right click context menu in Windows on the file to bring up 7zip option to extract the file.
4. Follow the instructions below to configure and use the pre-configured Xubuntu image. Begin by pressing the "New" icon at the top menu in Virtualbox to start the process of creating a new machine. Make the new machine named Xubuntu and select for type **Linux** and for version **Ubuntu (64-bit)**.

- Next, for memory size make sure you set the memory size to **at least 2048 MB** (4096 MB recommended).

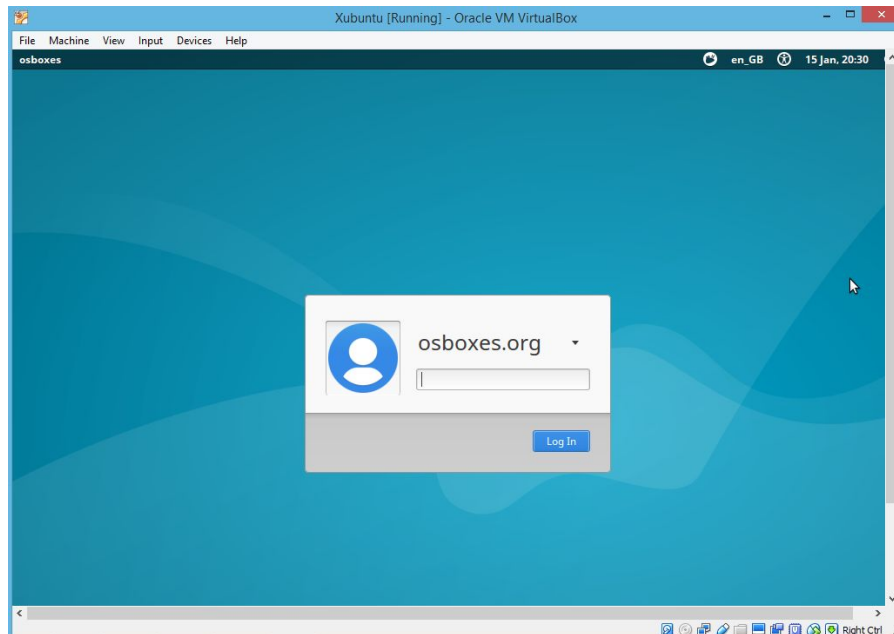


- Next, at the hard disk menu, select the final option "Use an existing virtual hard disk file" and then click the folder icon to load the virtual machine image that you extracted previously.

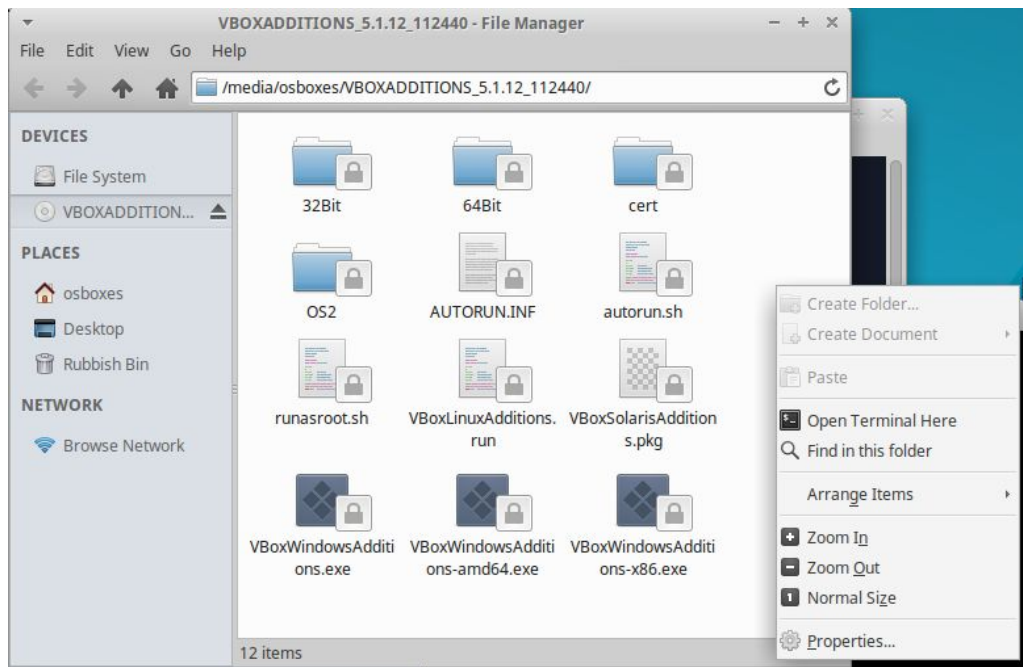




7. Finally, press start to launch your virtual machine, you should now be able to see the login screen. The password for the login is **osboxes.org**, this is also the password to use anytime you use the "sudo" command.



8. After logging into Xubuntu, next install the guest additions from the menu **“Devices”** then **“Insert Guest Additions CD Image”**. The Virtualbox guest additions make it much easier to work back and forth between the virtual machine and Windows by allowing you to share content that is copy and pasted.
9. You should now see that it has loaded a CD on the virtual machine, navigate to the contents of the CD directory and right click, from the menu select the option **“Open in terminal”** which will open a terminal window directly in the CD folder.



10. To install the Virtualbox guest additions run the following command from the terminal, when prompted for the password enter the same password as for login: **osboxes.org** After the installer has finished it will prompt you to restart your machine for the changes to take effect, after you restart the virtual machine the guest additions should now be installed and working.

**sudo ./VBoxLinuxAdditions.run**

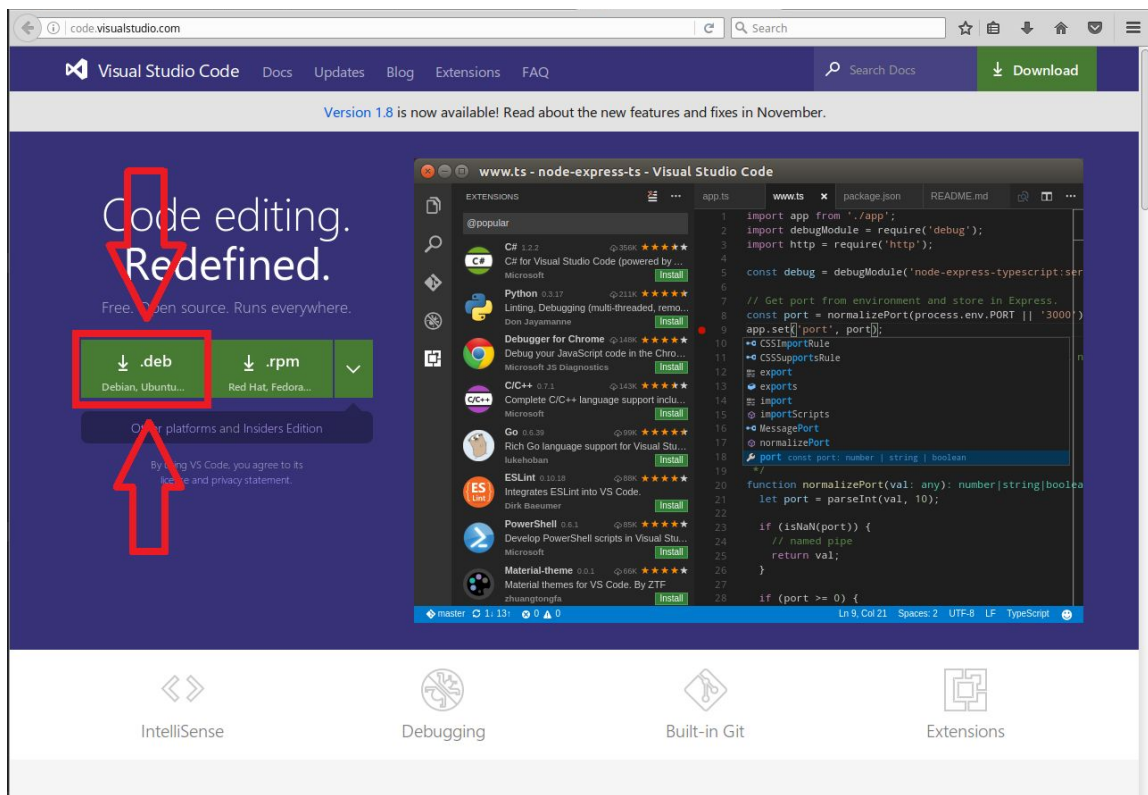
11. To enable the clipboard so you can use copy and paste back and forth between the virtual machine and your desktop from the main menu select **“Devices”** then for **“Shared Clipboard”** and **“Drag and Drop”** select the option **“Bidirectional”**.

12. Open a terminal by running the Terminal Emulator program from the programs menu, then enter the following commands to install all of the software you will need for the rest of the lab (you can copy and paste it all as one command):

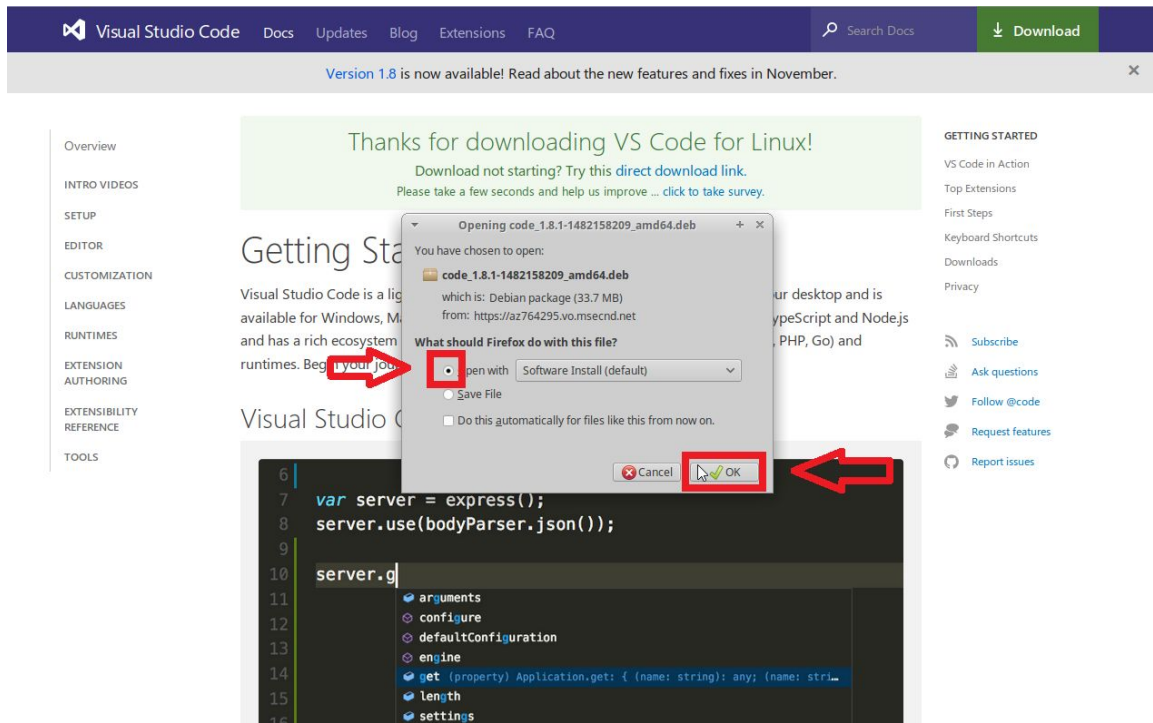
```
sudo apt-get update -y
sudo apt-get install -y build-essential clang lldb \
valgrind htop git
```

## IDE Installation (Visual Studio Code)

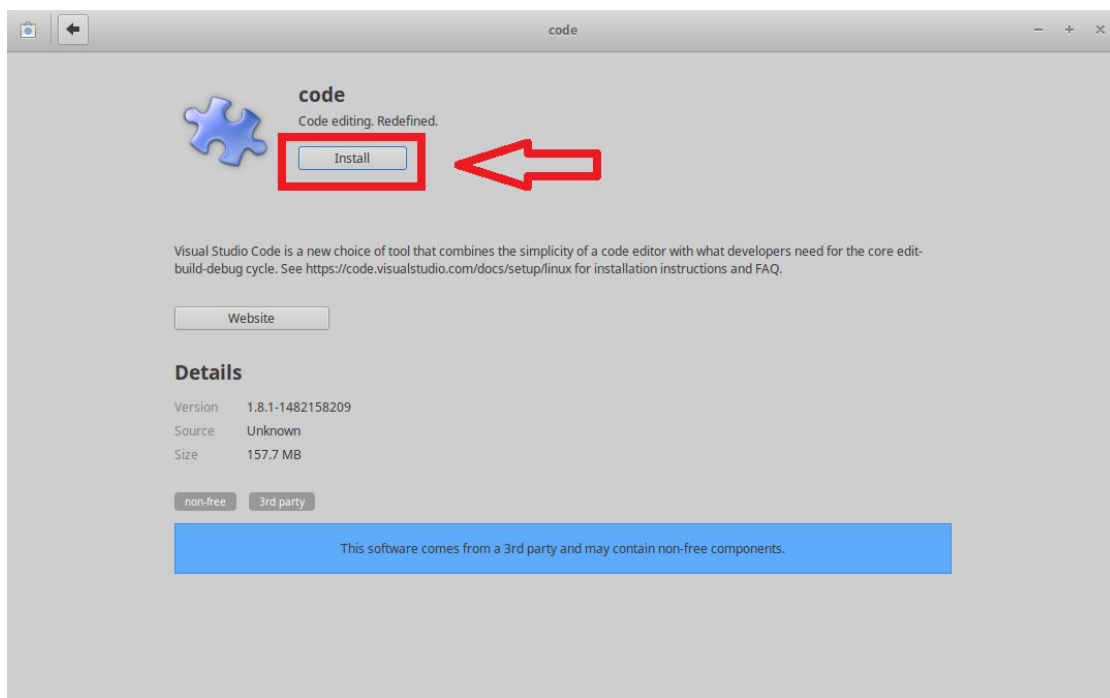
1. Download the .deb version of Visual Studio Code by going to <https://code.visualstudio.com/> in the browser of your Linux virtual machine and clicking the .deb download button.



2. Choose to open the .deb with Software Install and click OK to start the download.



3. Click the Install button once the download is complete and the Software Install program opens.

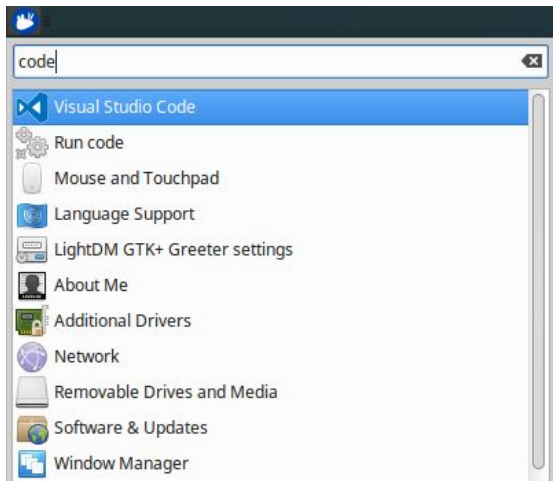




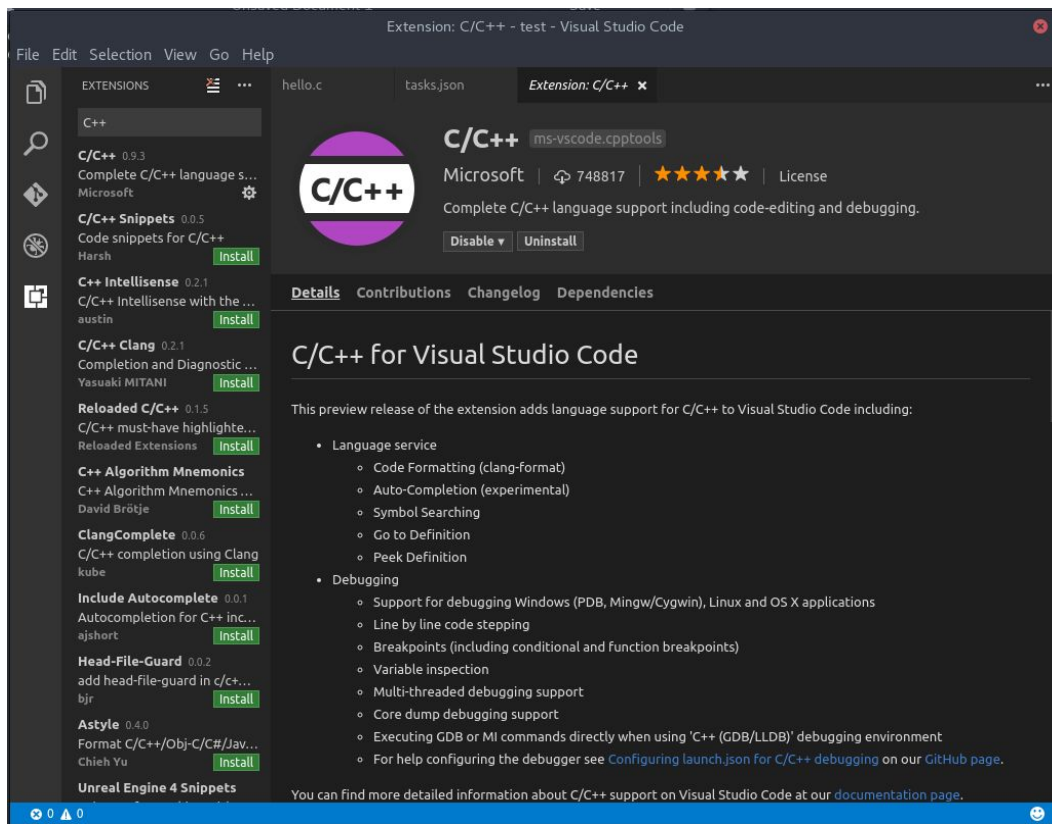
4. **If the previous step failed:** Download the Visual Studio Code .deb file again but instead of opening it save it to your desktop. Right-click your desktop and click the "Open Terminal" menu option. In your terminal enter:

```
sudo dpkg -i ./code_*.deb  
sudo apt-get install -f
```

5. **Visual Studio Code is now installed.** It can be launched from the program launcher via the search tool or within the Development category.



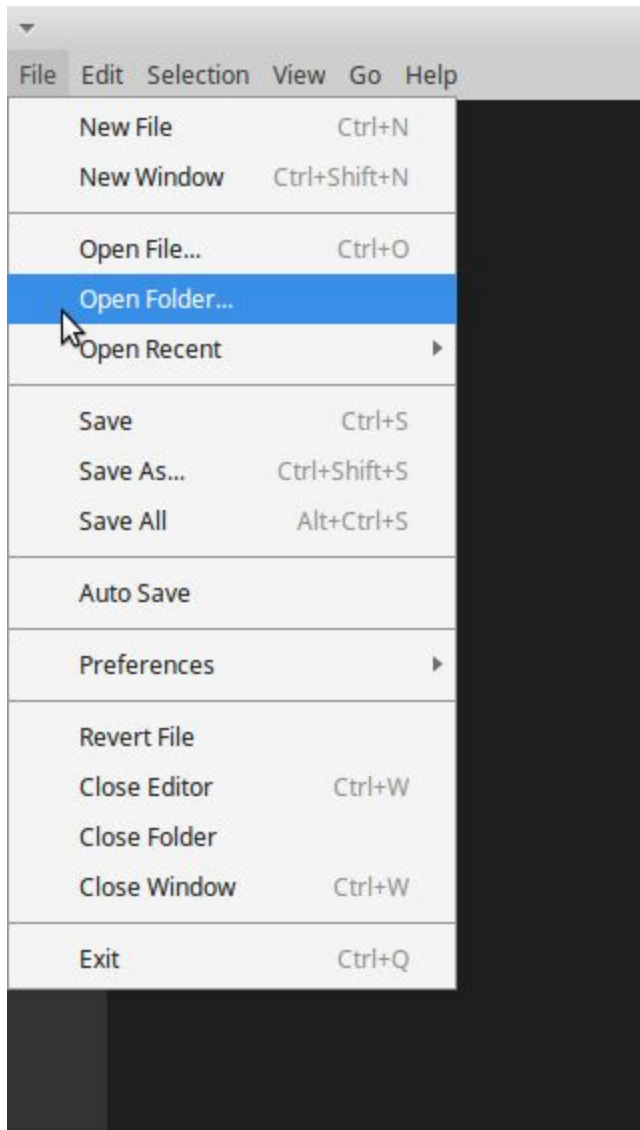
6. Lastly, install support for C/C++ in Visual Studio Code by selecting the final left navigation option (CTRL+SHIFT+X) to install extensions. In the search enter **C++** and install the C/C++ language extension published by Microsoft.



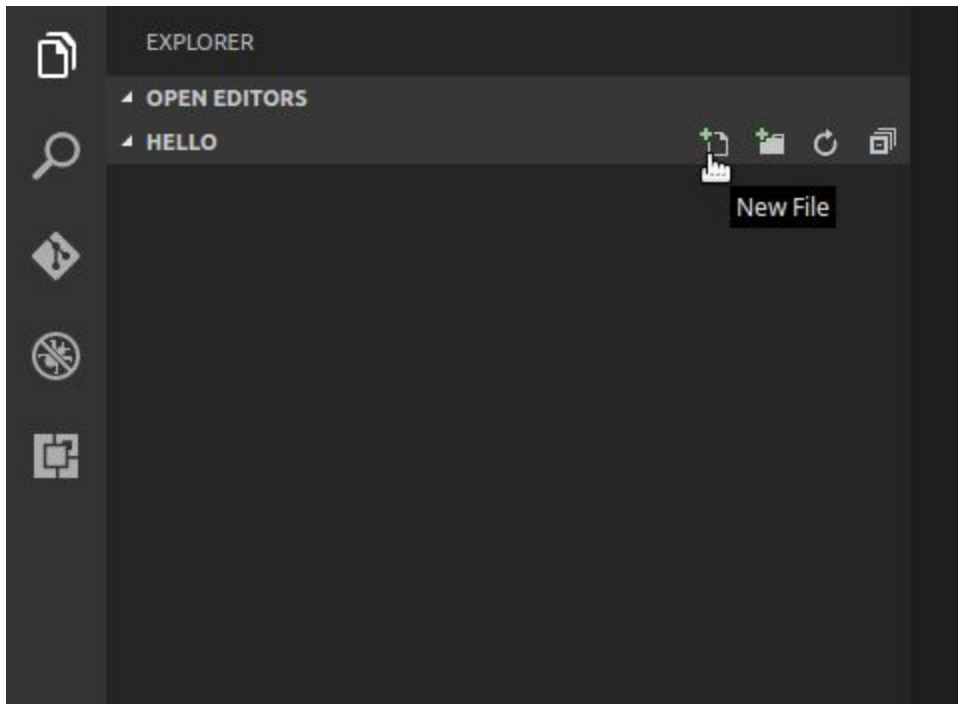
## C Project Setup

To setup a new C language project with Visual Studio Code to allow for building and debugging within the IDE please perform the following steps:

1. Open the folder that will contain your project's files in VS Code by choosing the "Open Folder..." option from the "File" menu. (If you haven't created a directory for your project yet you can create one with the "Create Folder" button in the "Open Folder..." dialog box.)



2. Click the “New File” button to create a new C file. Name this file “hello.c” for this lab session.

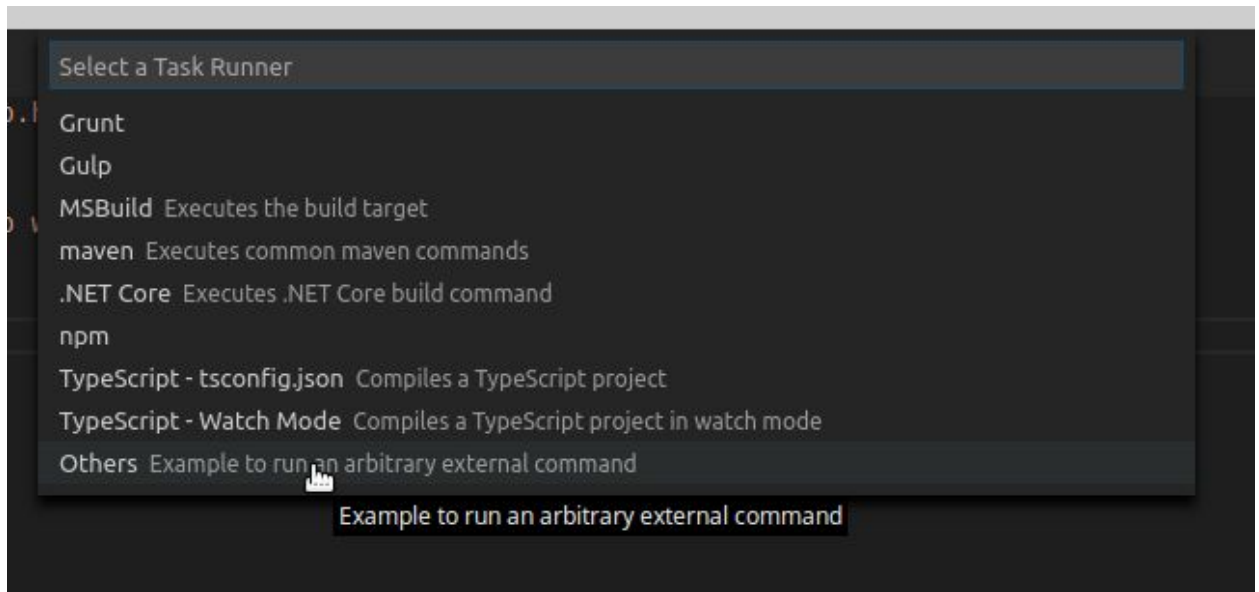


3. Enter the following code into the editor window for hello.c:

```
hello.c x
1  #include <stdio.h>
2
3  int main() {
4      puts("Hello world");
5      return 0;
6  }
7
```

4. Press the “Run Build Task” key combo (Ctrl-Shift-B) to build the code. Since no build command is configured a pop-up will appear. Press the “Configure Task Runner” button and choose “Others” from the menu that appears.





5. Change the command to “make” and remove any arguments from the “args” array, then save tasks.json.

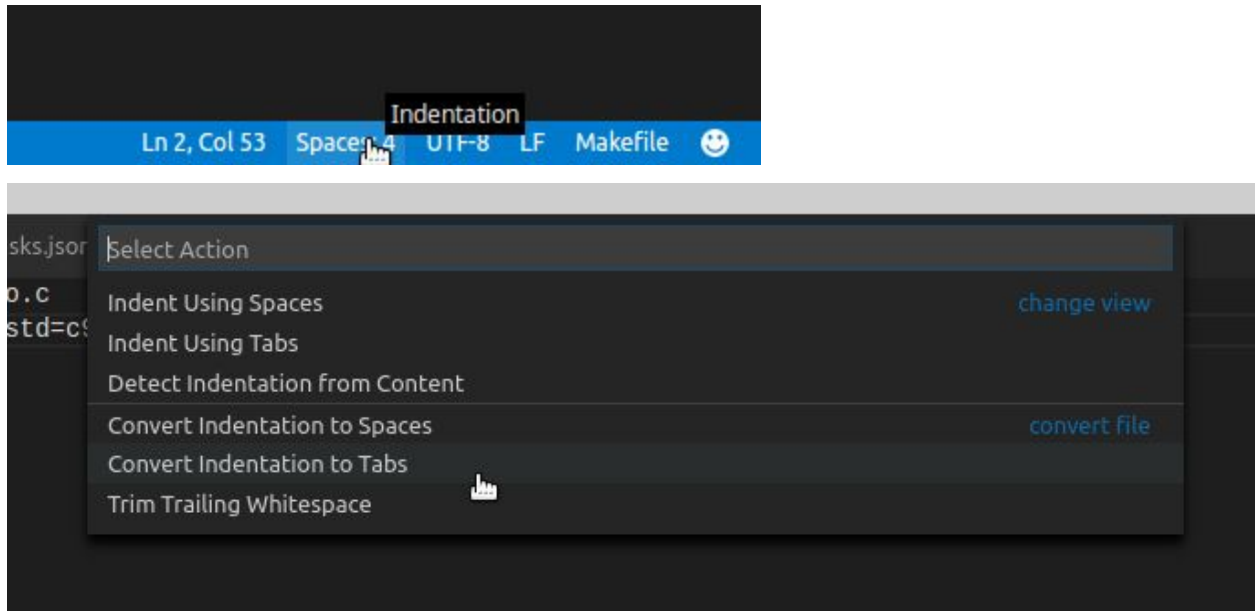
```
1 {  
2     // See https://go.microsoft.com/fwlink/?LinkId=733558  
3     // for the documentation about the tasks.json format  
4     "version": "0.1.0",  
5     "command": "make",  
6     "isShellCommand": true,  
7     "args": [],  
8     "showOutput": "always"  
9 }
```

6. Create a new file called “Makefile” in the root of the project folder and enter the following into the file:

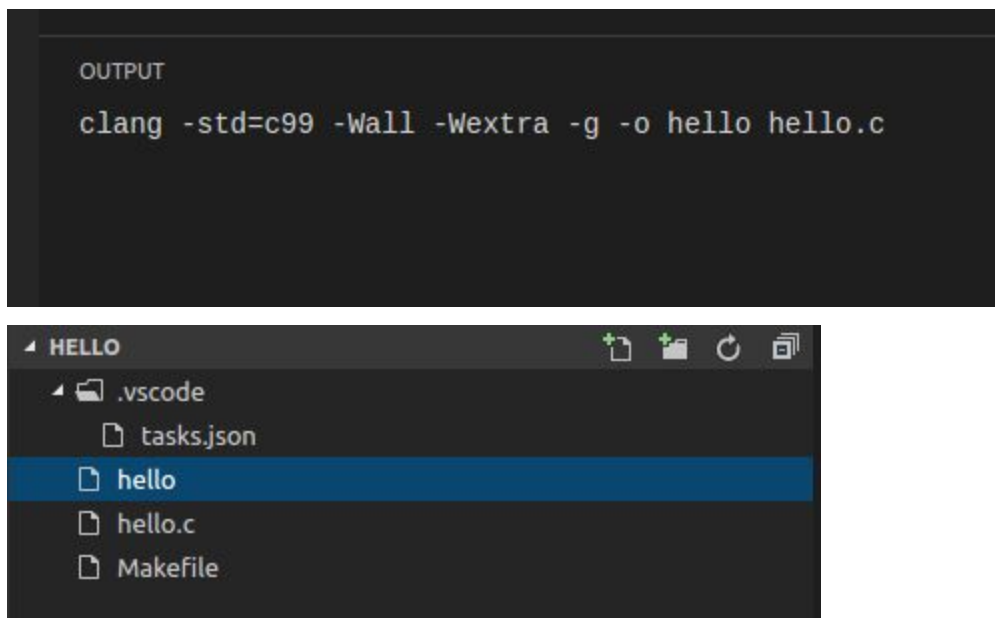
```
1 hello: hello.c  
2     clang -std=c99 -Wall -Wextra -g -o hello hello.c
```

7. **Make is very picky about indentation!** The second line should be indented by exactly one tab character. Make sure that there are only tabs in your Makefile by clicking the indentation settings button in the bottom status bar and choosing the “Convert Indentation to Tabs” option from the drop-down menu that appears.

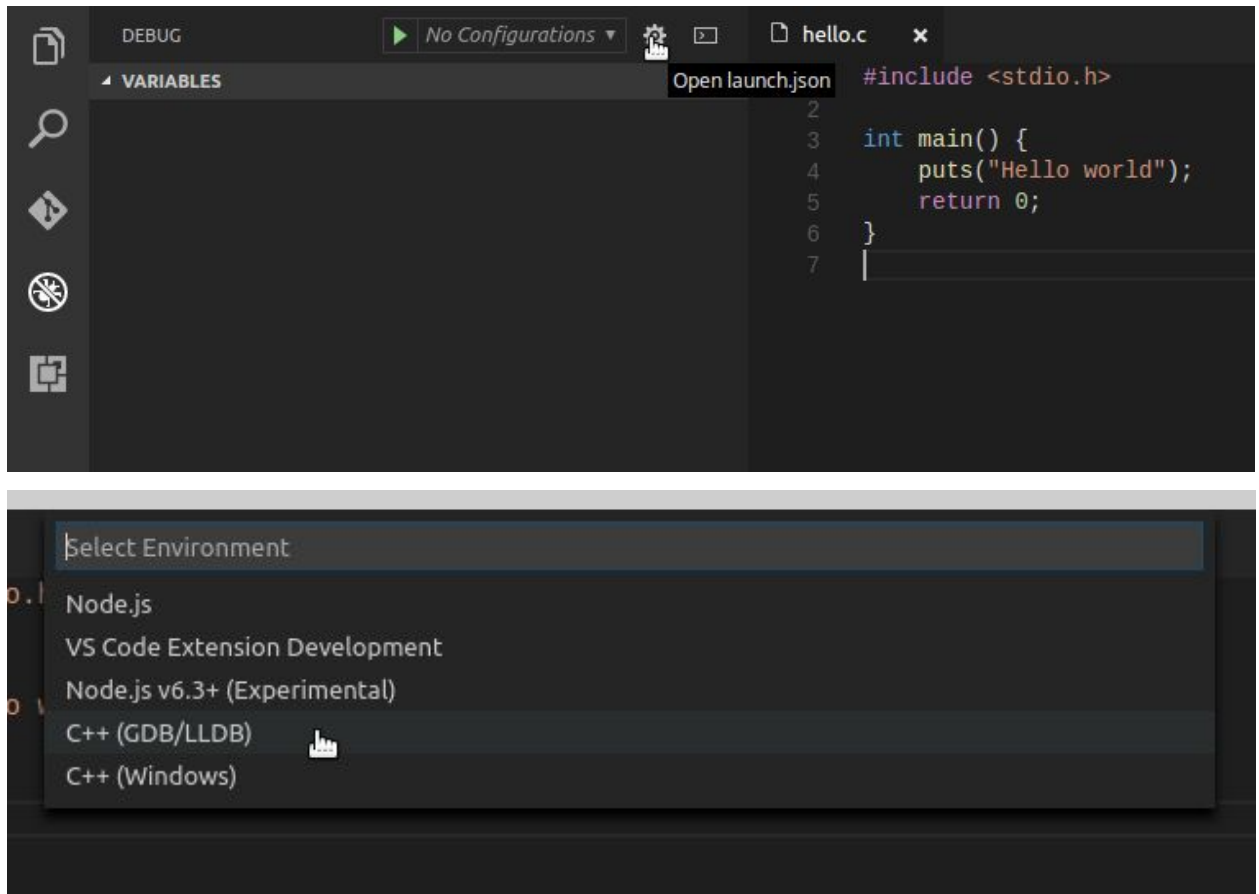
*Remember to save the changes to Makefile after you do this.*



8. Press the “Run Build Task” key command (Ctrl-Shift-B) again to build the program. The “Output” window should show the command that was executed by Make and a file named “hello” should appear in your project.



9. Open the “Debug” panel by clicking the fourth button in the toolbar on the left side of VS Code. Press the “Open launch.json” button and choose the “C++ (GDB/LLDB)” option from the drop-down menu to generate a launch.json file.



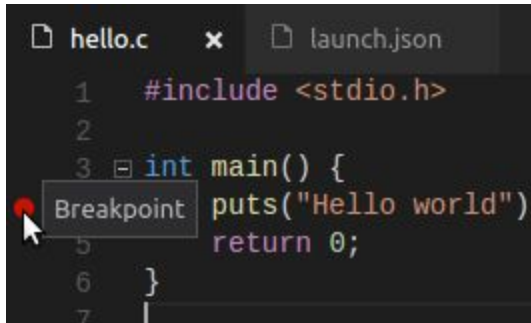
10. Modify the "MIMode" option in the "linux" section of the "C++ Launch" and "C++ Attach" configurations to be "lldb" and remove the "setupCommands" part. Change the "program" option of each configuration to "\${workspaceRoot}/hello".



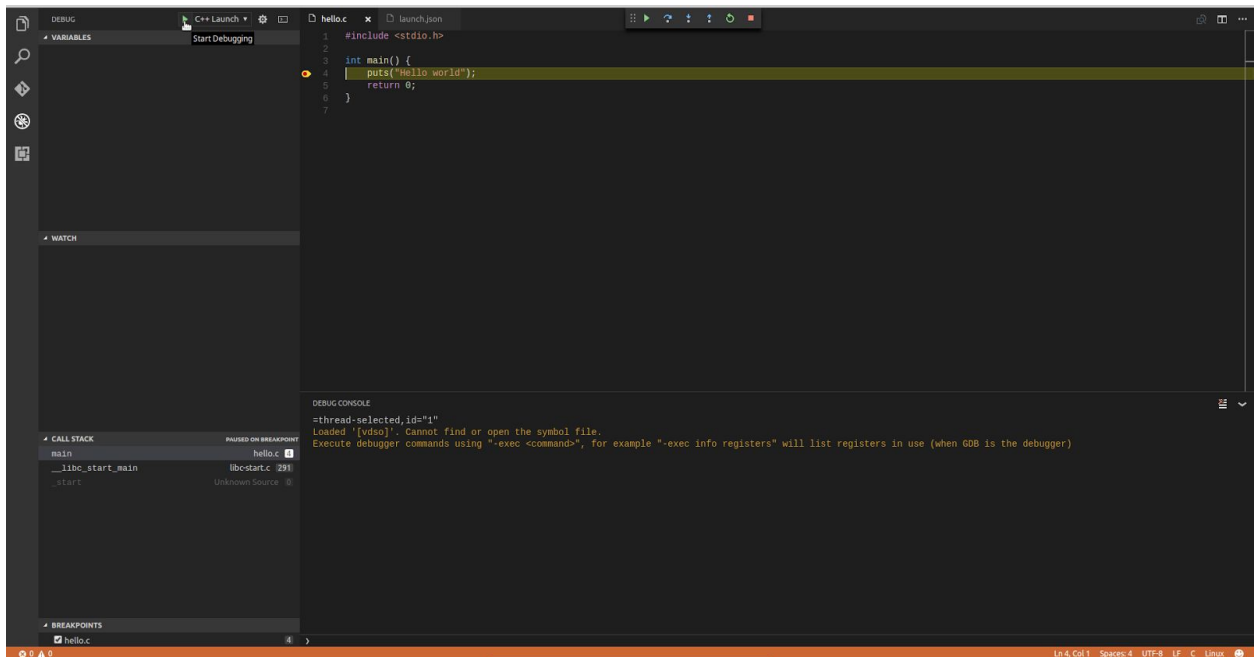
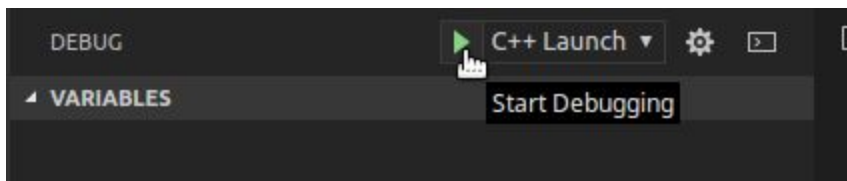
```
1  {
2      "version": "0.2.0",
3      "configurations": [
4          {
5              "name": "C++ Launch",
6              "type": "cppdbg",
7              "request": "launch",
8              "program": "${workspaceRoot}/hello",
9              "args": [],
10             "stopAtEntry": false,
11             "cwd": "${workspaceRoot}",
12             "environment": [],
13             "externalConsole": true,
14             "linux": {
15                 "MIMode": "lldb"
16             },
17             "osx": {
18                 "MIMode": "lldb"
19             },
20             "windows": {
21                 "MIMode": "gdb",
22                 "setupCommands": [
23                     {
24                         "description": "Enable pretty-printing for gdb",
25                         "text": "-enable-pretty-printing",
26                         "ignoreFailures": true
27                     }
28                 ]
29             }
30         },
31         {
32             "name": "C++ Attach",
33             "type": "cppdbg",
34             "request": "attach",
35             "program": "${workspaceRoot}/hello",
36             "processId": "${command.pickProcess}",
37             "linux": {
38                 "MIMode": "lldb"
39             },
40             "osx": {
41                 "MIMode": "lldb"
42             },
43             "windows": {
44                 "MIMode": "gdb"
```

11. Set a breakpoint in the first line of the main function in hello.c by clicking the empty space in the margin to the left of the line number. A red circle will appear.





12. Run the program in lldb by clicking the green play button in the “Debug” panel. The program will stop at the breakpoint. You can now use the controls at the top of the editor to step through the program line by line.



# GitHub Setup

1. For the labs and assignments in this course it is recommended that you become familiar with version control systems, in particular Git as it will be very beneficial for your collaborative work in this course and other courses.
2. If you haven't already create an account at <https://github.com> **make sure to use your uoit.net account**. After creating your account you will receive an email from GitHub to verify your email to complete the signup.
3. **Optional:** After you've created your GitHub account and verified your email apply for the student pack, which gives you access to a **plethora of software and services all for free!** (github premium account, domain names, \$100 in free hosting on digital ocean, etc.).

[https://education.github.com/discount\\_requests/new](https://education.github.com/discount_requests/new)

4. Next, go through the following tutorial in order to familiarize yourself with GitHub and git.

<https://try.github.io>

5. **Optional:** If you would like to learn more about git the following are all excellent resources, git immersion is great if you wish to become more familiar with the commands.

<http://git-scm.com/docs/gittutorial>

<http://gitref.org/>

<http://gitimmersion.com/>

<https://help.github.com/articles/set-up-git/#platform-linux>

# Git Setup / Publishing Your Project

1. Before you are able to commit anything with Git, you will need to tell Git the name and email address you would like it to sign your commits with. Run the following commands in a terminal, replacing "John Doe" with your name:

```
git config --global user.name "John Doe"  
git config --global user.email "john.doe@uoit.net"
```

2. Next you will need to set up a key-pair to be able to push commits to GitHub. Open a terminal and enter the following command:

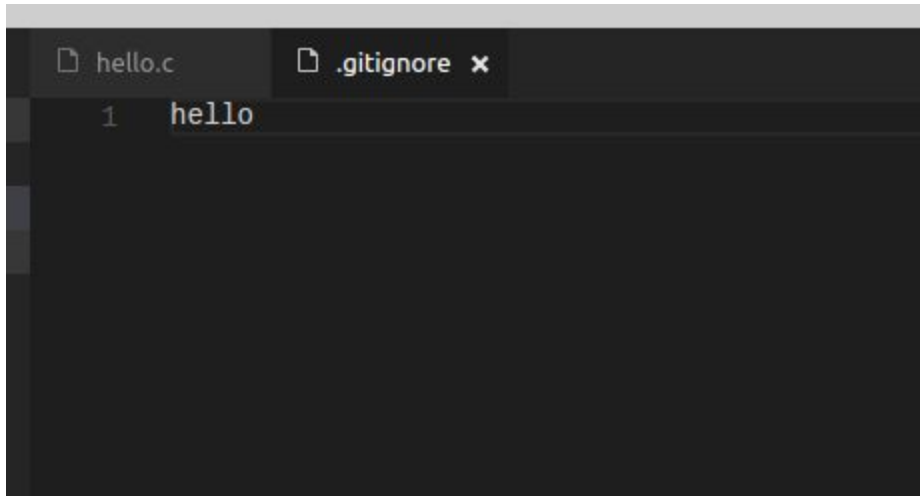
```
ssh-keygen -t rsa -b 4096
```

Press enter for each prompt **without entering anything** (i.e. **do not** enter a path or a password). After this dump your public key to the terminal by entering:

```
cat ~/.ssh/id_rsa.pub
```

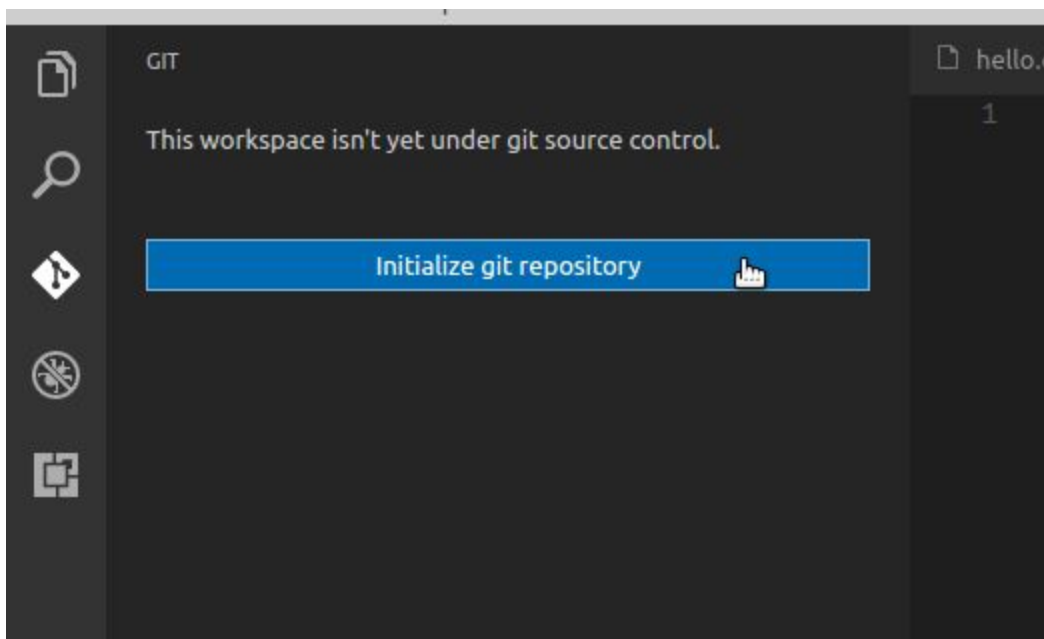
Copy the key from the terminal. Go to <https://github.com/settings/keys> (GitHub login required) and click the green "New SSH key" button in the top-right. Enter a title for the key (e.g. "Xubuntu VM") and paste the key in the "Key" box. Then click the green "Add SSH key" button to add the key to your GitHub account.

3. Add a file named .gitignore to the project. This file tells Git which files of your project it should not bother tracking for changes or including in commits (e.g. files generated by the compiler). Put the following in your .gitignore:

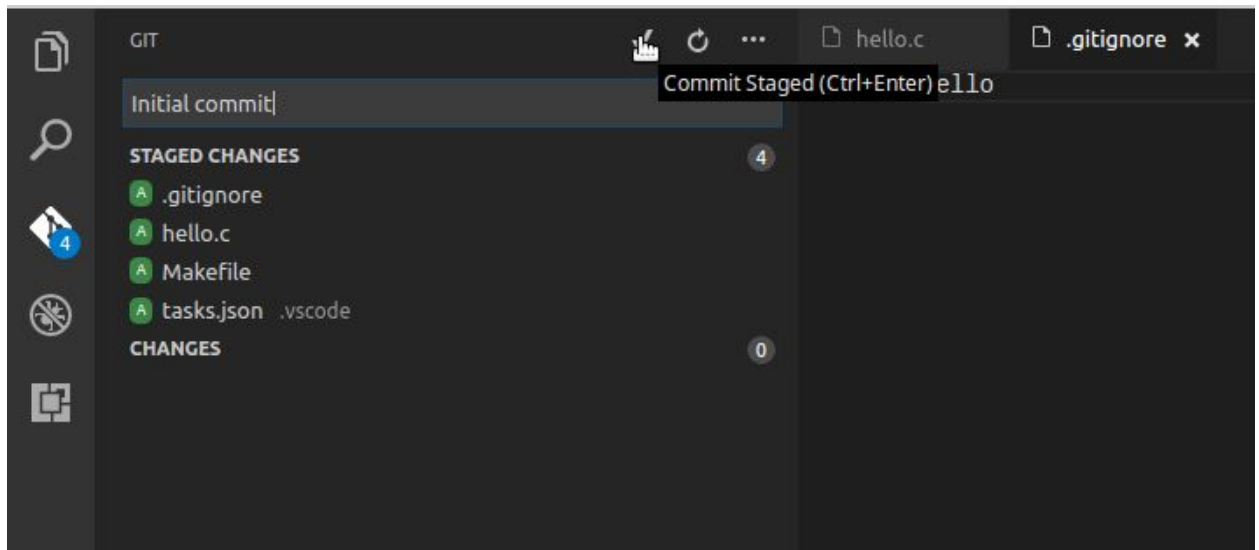
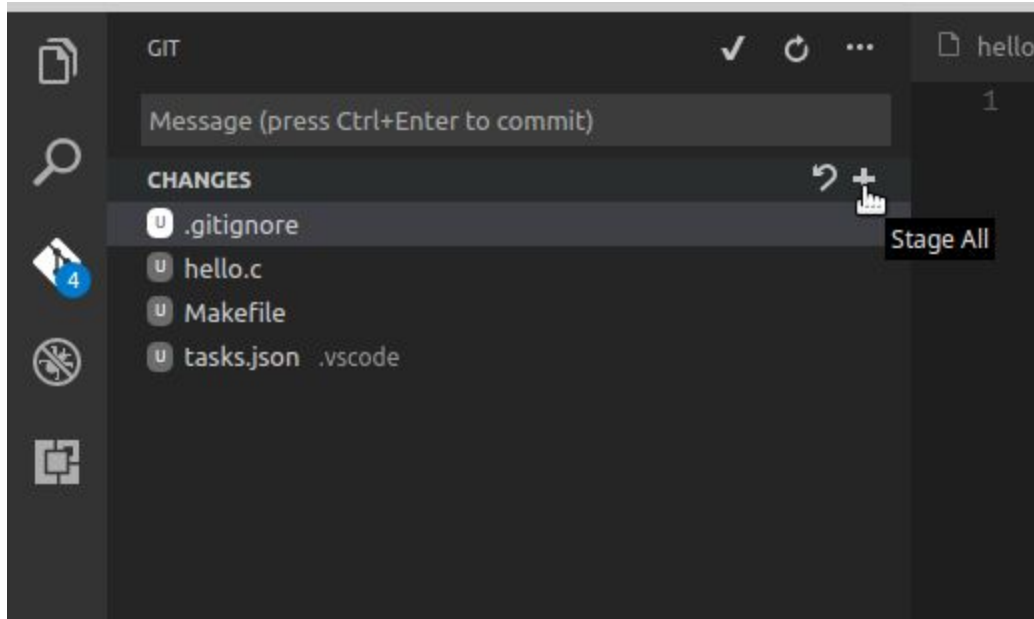


This will prevent Git from tracking the hello program generated by the C compiler.

4. Open the Git panel in VS Code (the third icon in the toolbar on the left) and click the “Initialize git repository” button to turn your project into a Git repository.

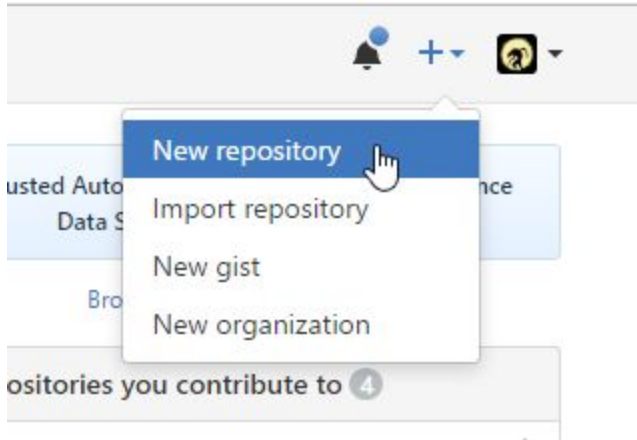


5. Press the “Stage all” button to stage all of the files in your project, then type a commit message in the message box (e.g. “Initial commit”) and press the “Commit” button to make your first commit.



6. You have now committed your changes to the Git repository on your local machine. To publish your repo on GitHub, first open a terminal with your project as the current directory by pressing Ctrl-Shift-C in Visual Studio Code. Set this aside for now.

Next create a new repo by opening <https://github.com> and clicking the + button in the upper-right corner of the page, and then selecting the “New repository” option. Give your repository a name and ensure that the “Initialize this repository with a README” option is unchecked.



Your repo will be initialized to a placeholder page. **Leave this open** as you will need it a few steps later.

A screenshot of the GitHub repository placeholder page for 'Patman64 / hello'. The page has a header with the repository name and icons for 'Unwatch', 'Star' (1), and 'Fork' (0). Below the header is a navigation bar with links for 'Code', 'Issues' (0), 'Pull requests' (0), 'Projects' (0), 'Wiki', 'Pulse', 'Graphs', and 'Settings'. The main content area is divided into four sections: 1. 'Quick setup — if you've done this kind of thing before' with buttons for 'Set up in Desktop', 'HTTPS', and 'SSH' (selected), and a text input field containing 'git@github.com:Patman64/hello.git'. 2. '...or create a new repository on the command line' with a code block containing the following commands: 

```
echo "# hello" >> README.md
git init
git add README.md
git commit -m "first commit"
git remote add origin git@github.com:Patman64/hello.git
git push -u origin master
```

 3. '...or push an existing repository from the command line' with a code block containing: 

```
git remote add origin git@github.com:Patman64/hello.git
git push -u origin master
```

 4. '...or import code from another repository' with a note 'You can initialize this repository with code from a Subversion, Mercurial, or TFS project.' and an 'Import code' button.

💡 ProTip! Use the URL for this page when adding GitHub as a remote.

7. You can publish it to your GitHub repo by running the commands specified in the repo's placeholder page inside the terminal you set aside in the previous step.

Patman64 / hello

Unwatch 1 Star 0 Fork 0

Code Issues 0 Pull requests 0 Projects 0 Wiki Pulse Graphs Settings

**Quick setup — if you've done this kind of thing before**

Set up in Desktop or **HTTPS** **SSH** git@github.com:Patman64/hello.git

We recommend every repository include a README, LICENSE, and .gitignore.

**...or create a new repository on the command line**

```
echo "# hello" >> README.md
git init
git add README.md
git commit -m "first commit"
git remote add origin git@github.com:Patman64/hello.git
git push -u origin master
```

**...or push an existing repository from the command line**

```
git remote add origin git@github.com:Patman64/hello.git
git push -u origin master
```

**...or import code from another repository**

You can initialize this repository with code from a Subversion, Mercurial, or TFS project.

Import code

ProTip! Use the URL for this page when adding GitHub as a remote.

**These commands will be slightly different for your repo!** Do not type in what you see in the image above. **NOTE:** Make sure that the "SSH" button is clicked in the "Quick setup" area so that the URL is of the form "[git@github.com](https://github.com:Patman64/hello.git):<your GitHub name>/<project name>.git".

8. After typing in the commands you can refresh your GitHub repo's page to see the contents of the repo.

Patman64 / hello

Unwatch 1 Star 0 Fork 0

Code Issues 0 Pull requests 0 Projects 0 Wiki Pulse Graphs Settings

No description or website provided. [Edit](#)

1 commit 1 branch 0 releases 1 contributor

Branch: master New pull request Create new file Upload files Find file Clone or download

Patman64 Initial commit Latest commit a769ef8 21 minutes ago

.vscode	Initial commit	21 minutes ago
.gitignore	Initial commit	21 minutes ago
Makefile	Initial commit	21 minutes ago
hello.c	Initial commit	21 minutes ago

Help people interested in this repository understand your project by adding a README. [Add a README](#)

You have now published your project on GitHub.

**If you are reading this and have done all of the above, you have completed the lab. Please show the T.A. your progress before you leave.**