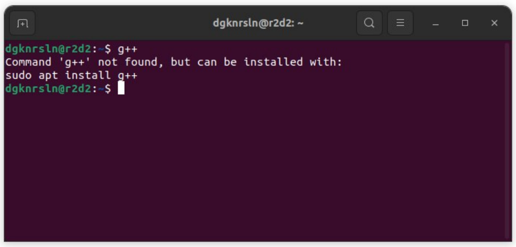
Development Environment Setup for Linux*

This document aims to explain how to set up the environment for assignments in the Linux operating system. Please make sure all steps are followed carefully. When a problem arises, please send an e-mail describing the problem, including what steps you took and what solutions you tried.

> Before starting to install docker, ensure that you enabled the virtualization technology. That setting generally is under the processor settings in the BIOS menu (search online for your BIOS key for your computer's manufacturer and model). The setting may be called VT-x, AMD-V, SVM, or Vanderpool. Enable Intel VT-d or AMD IOMMU if the options are available.

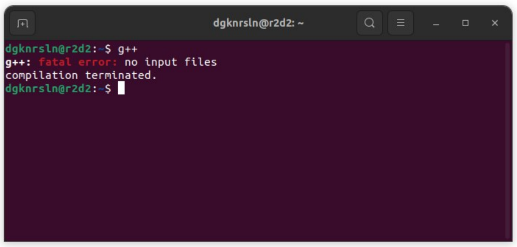
1 Installing Docker

1. Make sure that *GNU compiler collection* is installed in your system. Run g++ from terminal window to test. If it is not installed;



```
dgknrsln@r2d2:~$ g++
Command 'g++' not found, but can be installed with:
sudo apt install g++
dgknrsln@r2d2:~$
```

(a) GNU compiler collection is not installed. Command not found.



```
dgknrsln@r2d2:~$ g++
g++: fatal error: no input files
compilation terminated.
dgknrsln@r2d2:~$
```

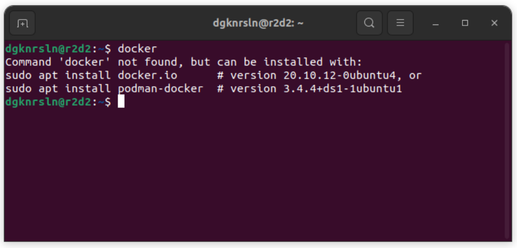
(b) GNU compiler collection is installed. g++ is looking for an input file.

1.1. Run the following commands as root or a user with sudo privileges.

```
sudo apt update
sudo apt install build-essential
```

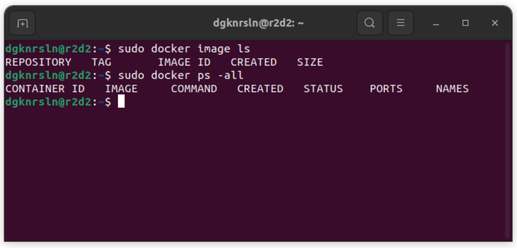
> In here, first command is used to download package information from all configured sources in order to get info on an updated version of existing packages or their dependencies. Latter command installs "*build-essential*" meta-package that includes the GNU compiler collection, GNU debugger, and other development libraries and tools required for compiling software.

> Note: In case you receive the g++ not found error, try removing gcc, g++, and build-essential packages and then reinstall build-essential.



```
dgknrsln@r2d2:~$ docker
Command 'docker' not found, but can be installed with:
sudo apt install docker.io      # version 20.10.12-0ubuntu4, or
sudo apt install podman-docker  # version 3.4.4+ds1-1ubuntu1
dgknrsln@r2d2:~$ 
```

(a) Docker is not installed. Command not found.



```
dgknrsln@r2d2:~$ sudo docker image ls
REPOSITORY   TAG       IMAGE ID   CREATED   SIZE
dgknrsln@r2d2:~$ sudo docker ps -all
CONTAINER ID   IMAGE     COMMAND   CREATED   STATUS    PORTS     NAMES
dgknrsln@r2d2:~$ 
```

(b) Docker is installed. There aren't any images and containers yet.

2. Make sure that *Docker* is installed and running in your system. Run `docker` from terminal window to test. If it is not installed;

   2.1. Run the following command as root or a user with sudo privileges.

   ```
   sudo apt install docker.io
   ```

   2.2. It may take a bit time according to the your download speed. After installation, it automatically starts, yet you can check whether it is running or not by following command

   ```
   sudo service docker status
   ```

   You may start the docker service if it is not active by following command:

   ```
   sudo service docker start
   ```

   2.3. To test the Docker installation, get the list of Docker images and containers. Both should be empty after the first installation.

   ```
   sudo docker image ls
   sudo docker ps -all
   ```

   Docker is a tool that uses OS-level virtualization to deliver software in packages which are called containers. The containers are basically a fully functional and portable computing environment surrounding the application and keeping it independent from other parallelly running environments. Individually each container simulates a different software application and runs isolated processes by bundling related configuration files, libraries and dependencies.

3. Create a folder in any directory (for example: `/home/Desktop/vm-docker`) with any name and place the Docker image called `dockerfile`, which is distributed over Ninova, in that folder. Open up a terminal and navigate to the directory where `dockerfile` is kept.

   A Docker image is a read-only template that contains a set of instructions for creating a container that can run on the Docker platform. It provides a convenient way to package up applications and pre-configured server environments. You can double click on it to see our environment instructions.

4. After navigating to the folder where dockerfile is kept (`cd /home/Desktop/vm-docker`) and run the following command to build the docker image. Here, `dockeralgo` is the name of the image.

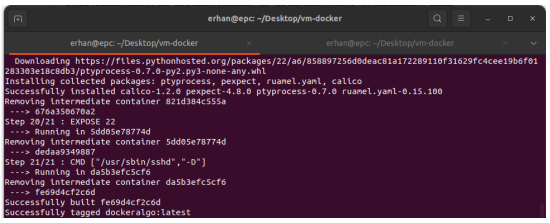
   ```
   sudo docker build -t dockeralgo .
   ```

*Ubuntu 22.04.2 LTS

Pay attention to the dot at the end of the command. It should take a while to build the image. If you get following warnings messages, it is safe to ignore them;

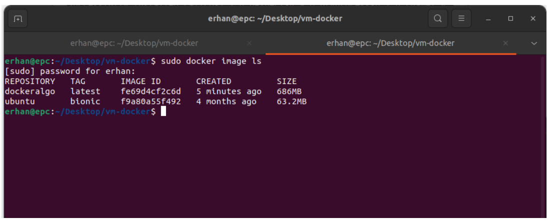
```
WARNING: apt does not have a stable CLI interface.  Use with caution in scripts.
debconf:  delaying package configuration, since apt-utils is not installed.
```

5. Check if the image has been built by getting the list of docker images.


erhan@epc: ~/Desktop/vm-docker
erhan@epc: ~/Desktop/vm-docker
erhan@epc: ~/Desktop/vm-docker

```
 Downloading https://files.pythonhosted.org/packages/22/a6/858897256d0deac81a172289110f31629fc4cee19b6f01
283303e18c8db3/ptyprocess-0.7.0-py2.py3-none-any.whl
Installing collected packages: ptyprocess, pexpect, ruamel.yaml, calico
Successfully installed calico-1.2.0 pexpect-4.8.0 ptyprocess-0.7.0 ruamel.yaml-0.15.100
Removing intermediate container 821d384c555a
 ---> 676a350670a2
Step 20/21 : EXPOSE 22
 ---> Running in 5dd05e78774d
Removing intermediate container 5dd05e78774d
 ---> dedaa9349887
Step 21/21 : CMD ["/usr/sbin/sshd","-D"]
 ---> Running in da5b3efc5cf6
Removing intermediate container da5b3efc5cf6
 ---> fe69d4cf2c6d
Successfully built fe69d4cf2c6d
Successfully tagged dockeralgo:latest
```

(a) Docker image is built.


erhan@epc: ~/Desktop/vm-docker
erhan@epc: ~/Desktop/vm-docker
erhan@epc: ~/Desktop/vm-docker

```
erhan@epc:~/Desktop/vm-docker$ sudo docker image ls
[sudo] password for erhan:
REPOSITORY   TAG       IMAGE ID       CREATED         SIZE
dockeralgo   latest    fe69d4cf2c6d   5 minutes ago   686MB
ubuntu       bionic    f9a80a55f492   4 months ago    63.2MB
erhan@epc:~/Desktop/vm-docker$ 
```

(b) Docker image can be seen in the image list.

6. Create a directory (`<dev_path>`) which will be visible to your container. You will be working on this directory (i.e reviewing recitation files, solving assignments). For example:

   ```
   /home/<username>/vm-docker/volume
   ```

7. Run the following command to boot a container instance from the image you have just built.

   ```
   docker run -p <local_port>:<container_port> -v <dev_path>:<container_path> -name
   <container_name> -hostname <container_host_name> -d <image_name>
   ```

`-p <local_port>:<container_port>`: Maps a port of the container to one of your PC ports. This is required to be able to connect into your container via SSH. <container_port> should be "22" as the image only exposes that port for the container, for different ports, dockerfile should be changed. Still, local port can be changed if there is another service operating on 2222 for example.

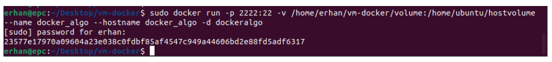
`-v <dev_path>:<container_path>`: Maps your local files in a directory to a directory inside the container.

`-name <container_name> -hostname <container_host_name>`: Provides name for the container and the host.

`-d <image_name>`: Provides the Docker image.

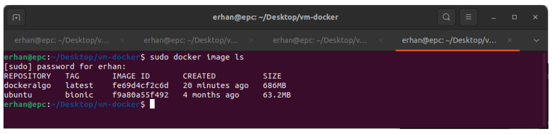
So, you should run something like this (you should change red part according to your <dev_path>);

```
sudo docker run -p 2222:22 -v /home/erhan/vm-docker/volume:/home/ubuntu/hostvolume
--name docker_algo --hostname docker_algo -d dockeralgo
```



```
erhan@epc:~/Desktop/vm-docker$ sudo docker run -p 2222:22 -v /home/erhan/vm-docker/volume:/home/ubuntu/hostvolume
--name docker_algo --hostname docker_algo -d dockeralgo
[sudo] password for erhan:
23577e17970a09604a23e038c0fdbf85af4547c949a44606bd2e88fd5adf6317
erhan@epc:~/Desktop/vm-docker$ 
```

(a) Docker container is created using the provided image.


erhan@epc: ~/Desktop/vm-docker
erhan@epc: ~/Desktop/v...
erhan@epc: ~/Desktop/v...
erhan@epc: ~/Desktop/v...
erhan@epc: ~/Desktop/v...

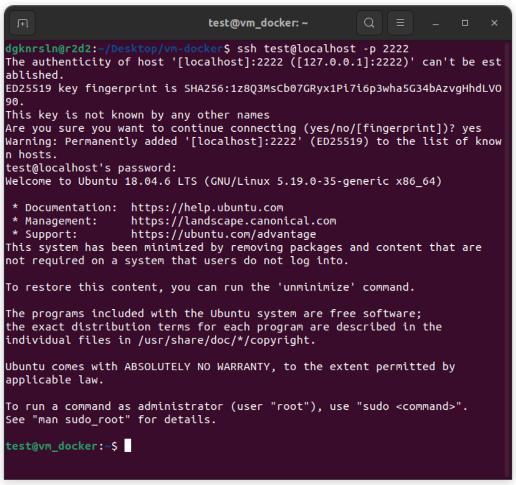
```
erhan@epc:~/Desktop/vm-docker$ sudo docker image ls
[sudo] password for erhan:
REPOSITORY   TAG       IMAGE ID       CREATED          SIZE
dockeralgo   latest    fe69d4cf2c6d   20 minutes ago   686MB
ubuntu       bionic    f9a80a55f492   4 months ago     63.2MB
erhan@epc:~/Desktop/vm-docker$ 
```

(b) Created container can be seen in the container list.

7.1. You may perform an additional check as by running;

```
sudo docker ps --all
```

8. Check if you can connect to your container via SSH, as follows;



```
dgknrsln@r2d2:~/Desktop/vm-docker$ ssh test@localhost -p 2222
The authenticity of host '[localhost]:2222 ([127.0.0.1]:2222)' can't be est
ablished.
ED25519 key fingerprint is SHA256:1z8Q3MsCb07GRyx1Pi7i6p3wha5G34bAzvgHhdLVO
90.
This key is not known by any other names
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '[localhost]:2222' (ED25519) to the list of know
n hosts.
test@localhost's password:
Welcome to Ubuntu 18.04.6 LTS (GNU/Linux 5.19.0-35-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage
This system has been minimized by removing packages and content that are
not required on a system that users do not log into.

To restore this content, you can run the 'unminimize' command.

The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.

To run a command as administrator (user "root"), use "sudo <command>".
See "man sudo_root" for details.

test@vm_docker:~$
```

(a) Connected to the container using SSH.

```
ssh test@localhost -p <local_port>
```

Your container's both username and password are set as `"test"`. You may change it by editing the `dockerfile`, rebuilding the image, and re-running the container. <local_port> is 2222 in given example. Don't forget to exit from the SSH session (`exit`).

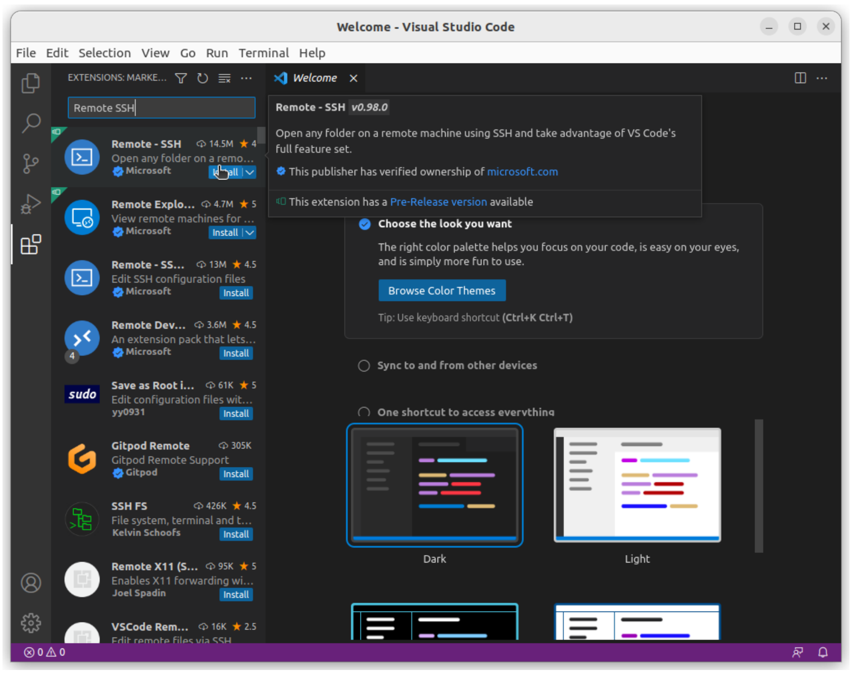
SSH (Secure Shell) is a network communication protocol that enables two computers to communicate via HTTP.

Note: In this step, if you encounter a warning stating "WARNING: REMOTE HOST IDENTIFICATION HAS CHANGED!", you need to set a different port number while booting up the docker instance after deleting it on Docker Desktop. This happens since you are trying to connect to a previous instance by using the same port, same address, and same username.

2 VS-Code Set-up

2.1 Installing Extensions

1. Install `Visual Studio Code` in your system. Visit here for installation instructions and FAQ.
2. Run `Visual Studio Code` and install the extension named `"Remote SSH"`.


Welcome - Visual Studio Code
File Edit Selection View Go Run Terminal Help
EXTENSIONS: MARKE...
Welcome

Remote SSH

Remote - SSH 14.5M ★ 4
Open any folder on a remo...
Microsoft Install

Remote Explo... 4.7M ★ 5
View remote machines for ...
Microsoft Install

Remote - SS... 13M ★ 4.5
Edit SSH configuration files
Microsoft Install

Remote Dev... 3.6M ★ 4.5
An extension pack that lets...
Microsoft Install

Save as Root i... 61K ★ 5
Edit configuration files wit...
yy0931 Install

Gitpod Remote 305K
Gitpod Remote Support
Gitpod Install

SSH FS 426K ★ 4.5
File system, terminal and t...
Kelvin Schoofs Install

Remote X11 (S... 95K ★ 5
Enables X11 forwarding wi...
Joel Spadin Install

VSCode Rem... 16K ★ 2.5
Edit remote files via SSH

Remote - SSH v0.98.0
Open any folder on a remote machine using SSH and take advantage of VS Code's full feature set.
This publisher has verified ownership of microsoft.com
This extension has a Pre-Release version available

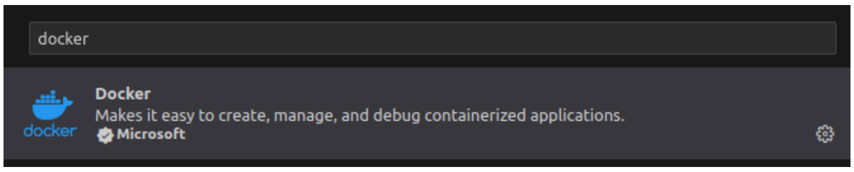
Choose the look you want
The right color palette helps you focus on your code, is easy on your eyes, and is simply more fun to use.
Browse Color Themes
Tip: Use keyboard shortcut (Ctrl+K Ctrl+T)

Sync to and from other devices

One shortcut to access everything

(a) Installing Remote SSH extension in VS Code.

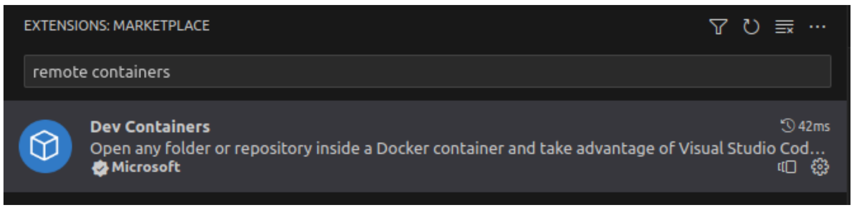
3. Install "Docker" extension on VS-Code.


docker

Docker
Makes it easy to create, manage, and debug containerized applications.
Microsoft

(a) Installing Docker extension in VS Code.

4. To attach Visual Studio Code on Docker containers, you need to install “Dev Containers” (formerly, remote containers extension):



remote containers

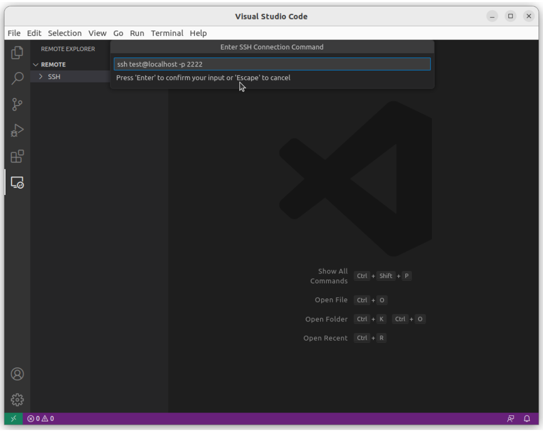
Dev Containers 42ms
Open any folder or repository inside a Docker container and take advantage of Visual Studio Cod...
Microsoft

(a) Installing Dev Containers extension in VS Code.

5. From now on, you can work with containers in VS-Code in 2 different ways: (1) Connecting with SSH within VS-Code, (2) Connecting with Docker extension. Choose which way to follow and navigate to the relevant instructions below. After that, follow instructions in 2.4 "In-container set-up" and so on.

2.2 Connecting with SSH within VS-Code

1. Click on the newly added "Remote Explorer" button and make a connection with `localhost`. If there is no SSH target you have to add a new (step a-b).

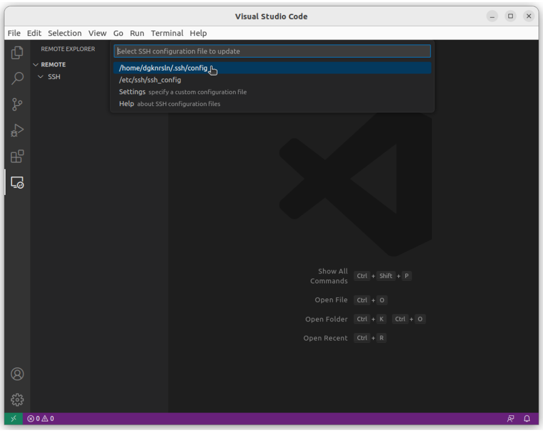


Enter SSH Connection Command

```
ssh test@localhost -p 2222
```

Press 'Enter' to confirm your input or 'Escape' to cancel

(a) Add new SSH target clicking "+" icon next to the SSH section. Write down `ssh test@localhost -p 2222`.



Select SSH configuration file to update

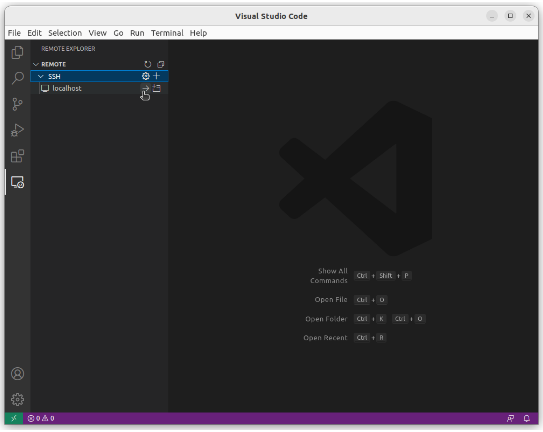
/home/dgknrsln/.ssh/config

/etc/ssh/ssh_config

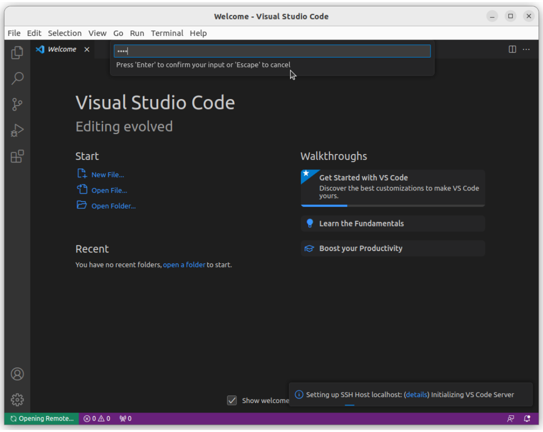
Settings specify a custom configuration file

Help about SSH configuration files

(b) Choose SSH configuration file to update as in the figure.



(c) Click → to open remote computer in the same window.



Press 'Enter' to confirm your input or 'Escape' to cancel

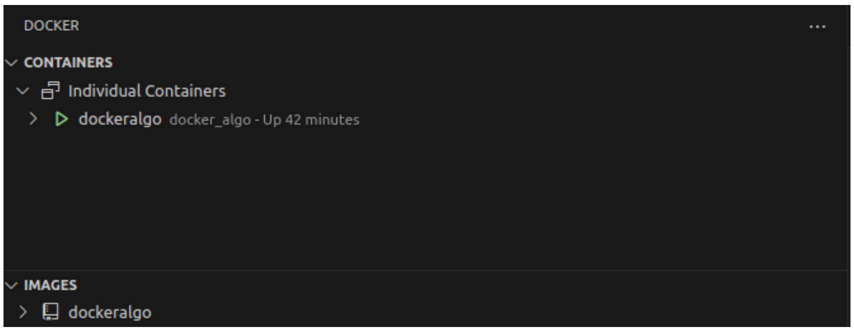
You have no recent folders, open a folder to start.

Setting up SSH Host localhost: (details) Initializing VS Code Server

(d) A new window should open and ask for password. If you didn't change anything, your default should be "test".

2.3 Connecting with Docker extension

1. Click on the docker icon on the left pane. Since the container is built from the image we prepared, we should see the container. Green arrow means that container is now active, red rectangle means container has stopped. Our docker_algo container is now up and ready to be used.

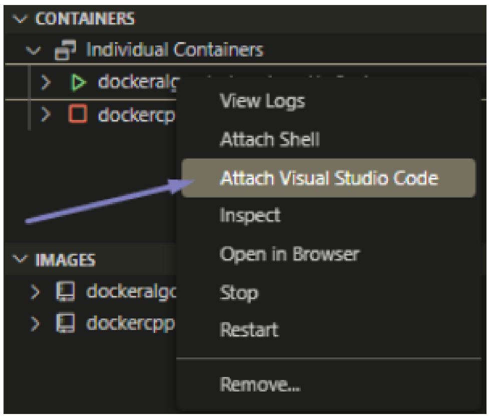


dockeralgo docker_algo - Up 42 minutes

(a) Container can be seen through docker extension.

> If containers and images are not showing up and error message is shown as "permission denied". Your user might not have the permission to access docker, you can change it by applying following commands from terminal:
>
> ```
> sudo usermod -aG docker $USER
> newgrp docker
> ```
>
> Then, re-start the VS-Code and try again.

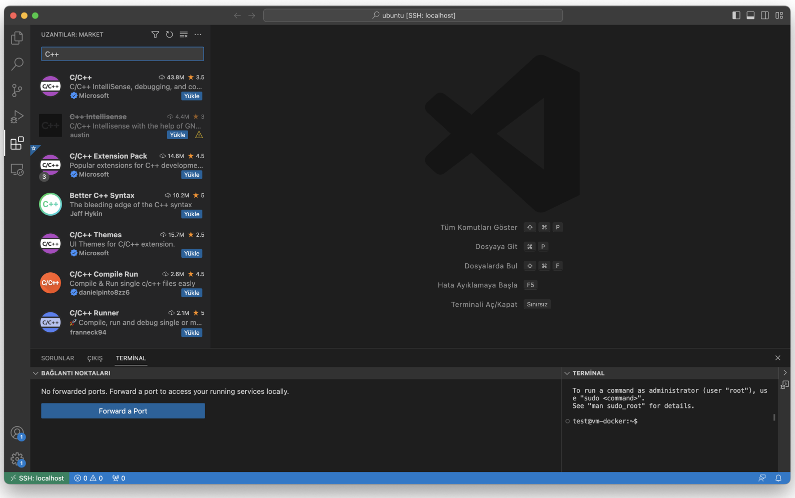
2. To work within the container, left-click on the docker_algo container and press "Attach Visual Studio Code". This will open a new VS-Code window for the container.



(a) A new window will be opened within the container.

2.4 In-container set-up

1. In the new window, we should install a few more extensions for C++ development. Search for "C++" among the extensions and install "C/C++ Extension Pack".

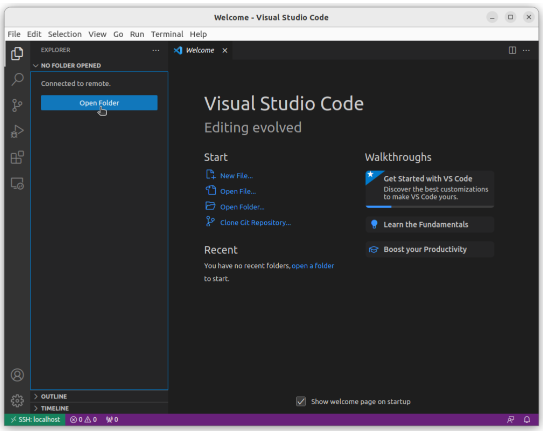


```
To run a command as administrator (user "root"), us
e "sudo <command>".
See "man sudo_root" for details.

test@vm-docker:~$
```

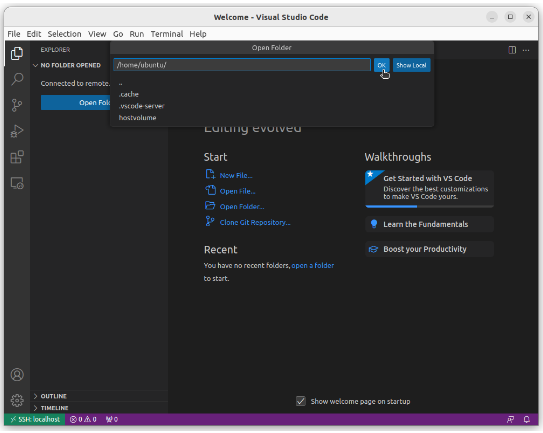
(a) Installing C/C++ extensions in VS Code.

2. After that, you can see folders in remote computer using "open folder" button as below. Every files you placed into your local `<dev_path>` will be visible in VS Code "`/home/ubuntu/hostvolume/`" directory.

3. If you encounter with permission error about files when using hostvolume on VS Code, you have to change ownership of the folder using following command;



(a) Click "open folder" button.

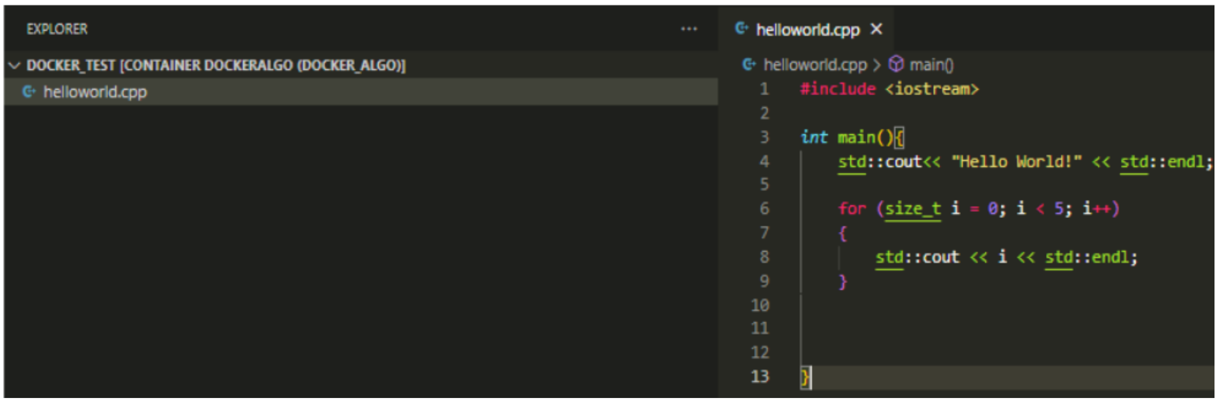


(b) Click "OK" button.

```
chown -R test /home/ubuntu/hostvolume
```

3 Developing in VS-Code

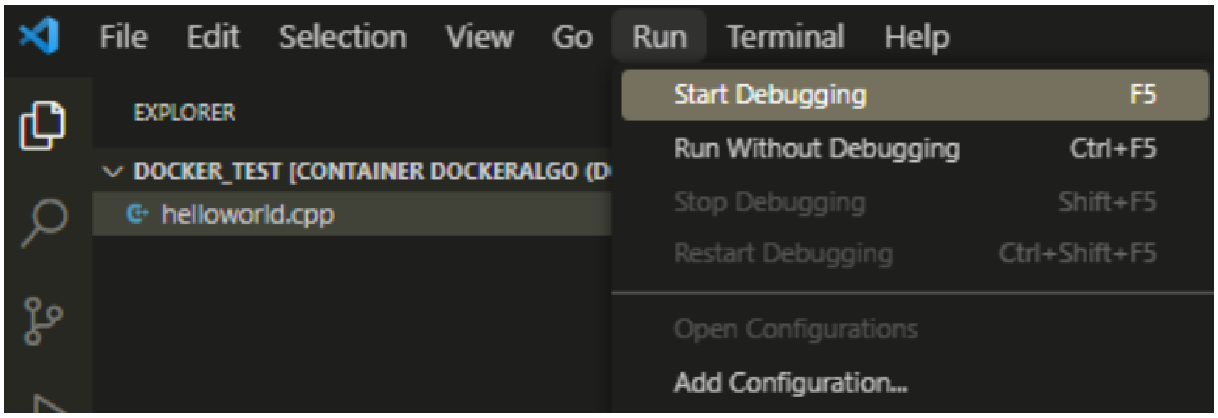
1. Press CTRL+K+O and navigate to home/Ubuntu/hostVolume. Create a folder named “docker_test” and open that folder with CTRL+K+O again. Create a “helloworld.cpp” within that folder like following:



```
#include <iostream>

int main(){
    std::cout<< "Hello World!" << std::endl;

    for (size_t i = 0; i < 5; i++)
    {
        std::cout << i << std::endl;
    }



}
```

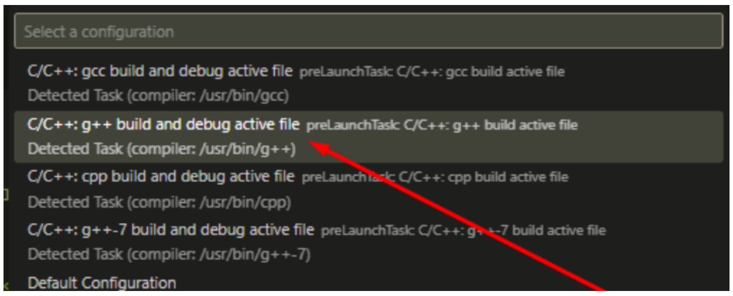
(a) helloworld.cpp

2. Press ”Start debugging” from “Run” menu.



(a) Run with debugging option

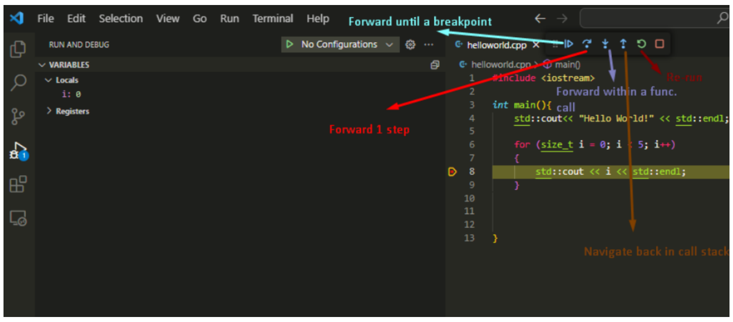
3. Choose g++ configuration in the pop-up window:



```
Select a configuration
C/C++: gcc build and debug active file  preLaunchTask: C/C++: gcc build active file
Detected Task (compiler: /usr/bin/gcc)
C/C++: g++ build and debug active file  preLaunchTask: C/C++: g++ build active file
Detected Task (compiler: /usr/bin/g++)
C/C++: cpp build and debug active file  preLaunchTask: C/C++: cpp build active file
Detected Task (compiler: /usr/bin/cpp)
C/C++: g++-7 build and debug active file  preLaunchTask: C/C++: g++-7 build active file
Detected Task (compiler: /usr/bin/g++-7)
Default Configuration
```

(a) g++ for building the .cpp files

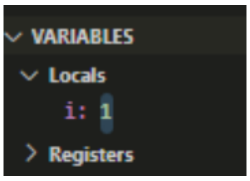
4. Place a breakpoint by clicking on the 8th line. Then start debugging again. When the breakpoint is reached, program stops at that state and shows the variables on the left pane. As can be seen, variable is 0 right now and it will be 1 in the next breakpoint encountering.


File Edit Selection View Go Run Terminal Help
Forward until a breakpoint
RUN AND DEBUG
No Configurations
VARIABLES
Locals
i: 0
Registers
Forward 1 step
helloworld.cpp
helloworld.cpp > main()
Forward within a func. call
Re-run
Navigate back in call stack

```
1  #include <iostream>
2
3  int main(){
4      std::cout<< "Hello World!" << std::endl;
5
6      for (size_t i = 0; i < 5; i++)
7      {
8          std::cout << i << std::endl;
9      }
10
11
12
13 }
```

(a) Debugging instructions

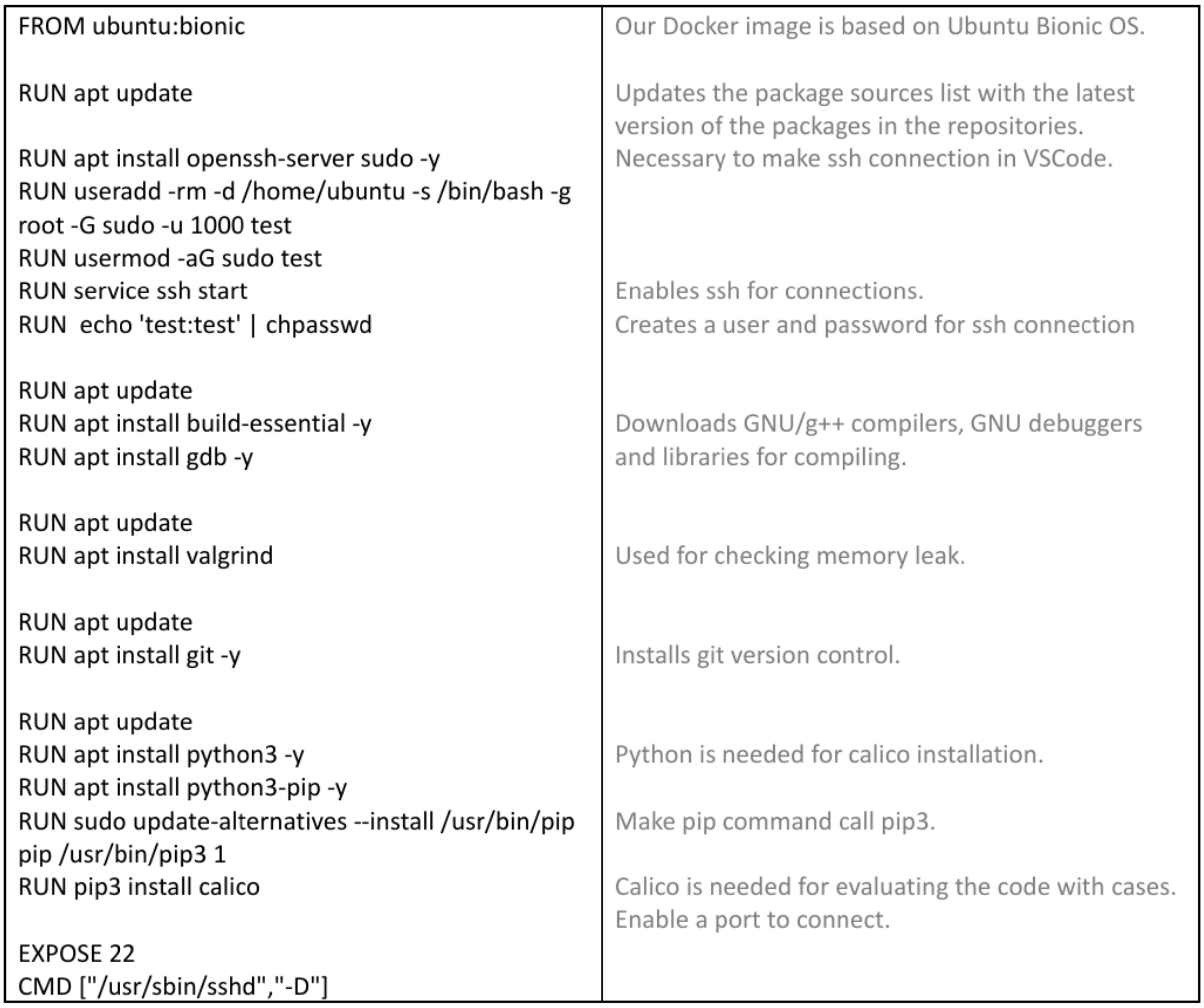
5. You can use several options when debugging and it has been shown with arrows on above image. When you forward the program until a new breakpoint is encountered, the variable would be 1



(a) change in value of variable can be seen through debugging pane

4 What is inside the Dockerfile?

You may ignore this part if you are not interested.



FROM ubuntu:bionic	Our Docker image is based on Ubuntu Bionic OS.
RUN apt update	Updates the package sources list with the latest version of the packages in the repositories.
RUN apt install openssh-server sudo -y	Necessary to make ssh connection in VSCode.
RUN useradd -rm -d /home/ubuntu -s /bin/bash -g root -G sudo -u 1000 test	
RUN usermod -aG sudo test	
RUN service ssh start	Enables ssh for connections.
RUN echo 'test:test' | chpasswd	Creates a user and password for ssh connection
RUN apt update	
RUN apt install build-essential -y	Downloads GNU/g++ compilers, GNU debuggers
RUN apt install gdb -y	and libraries for compiling.
RUN apt update	
RUN apt install valgrind	Used for checking memory leak.
RUN apt update	
RUN apt install git -y	Installs git version control.
RUN apt update	
RUN apt install python3 -y	Python is needed for calico installation.
RUN apt install python3-pip -y	
RUN sudo update-alternatives --install /usr/bin/pip pip /usr/bin/pip3 1	Make pip command call pip3.
RUN pip3 install calico	Calico is needed for evaluating the code with cases.
	Enable a port to connect.
EXPOSE 22	
CMD ["/usr/sbin/sshd","-D"]	

(a) Dockerfile