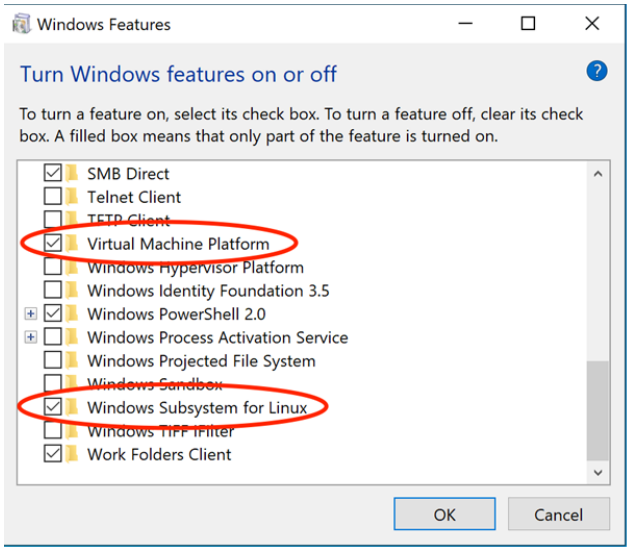
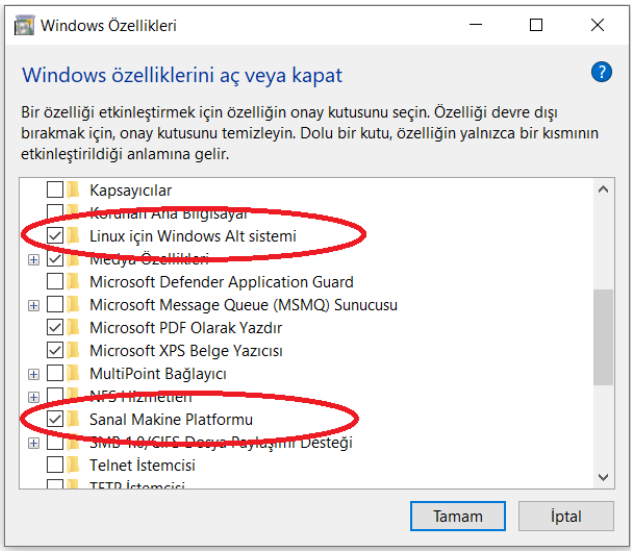
ITU Computer Engineering Department
BLG 223E Data Structures, Fall 2021-2022
Development Environment Setup – Windows 10

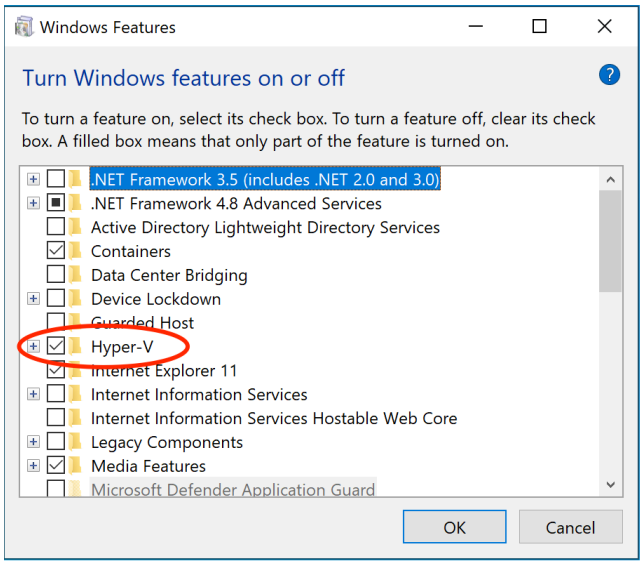
1. Enable BIOS-level hardware virtualization support in your computer's BIOS settings. If you have not done this before, this site may be helpful.

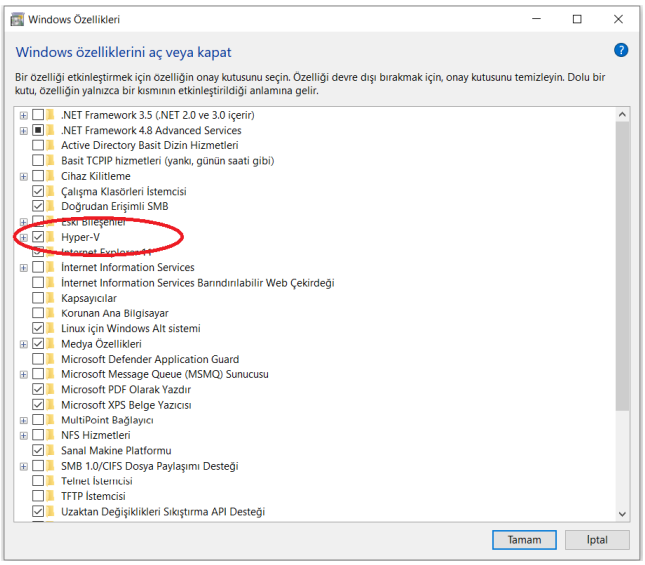
2. Enable Virtual Machine Platform and Windows Subsystem for Linux (WSL) Windows features:





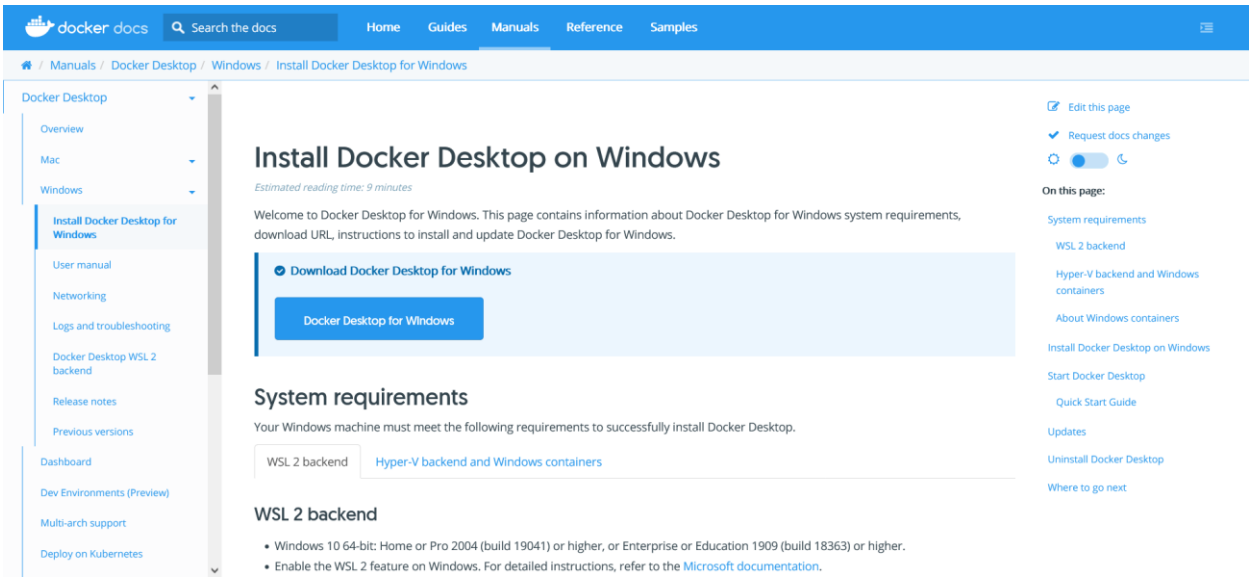
3. Enable Hyper-V Windows feature:



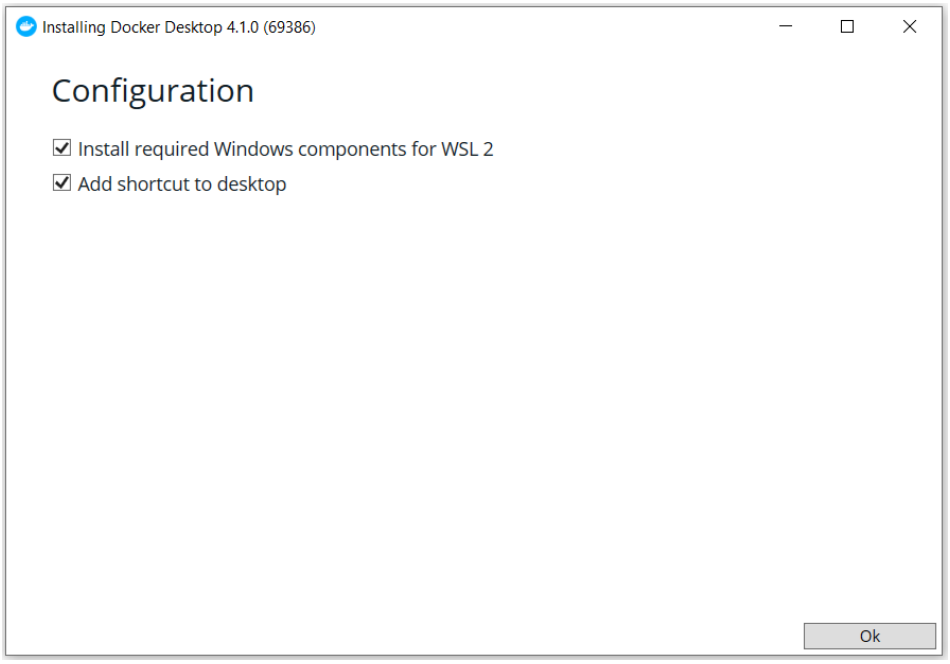


4. Update Windows 10 (Start->Settings->Update & Security) so that the changes are applied (e.g., installation of WSL). Then, restart your computer.

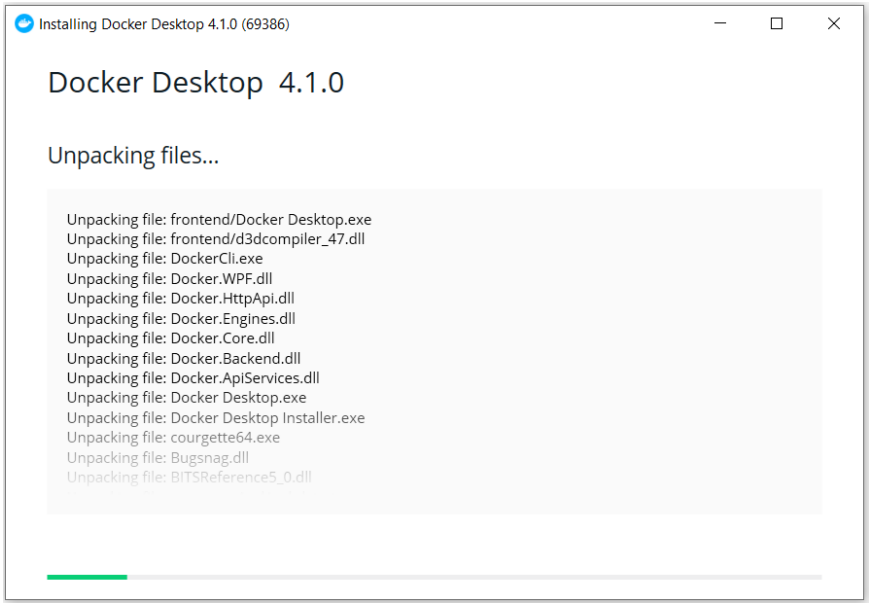
5. Open your web browser (e.g, Firefox or Chrome) and go to the Install Docker Desktop on Windows web page to download Docker Desktop Installer application:

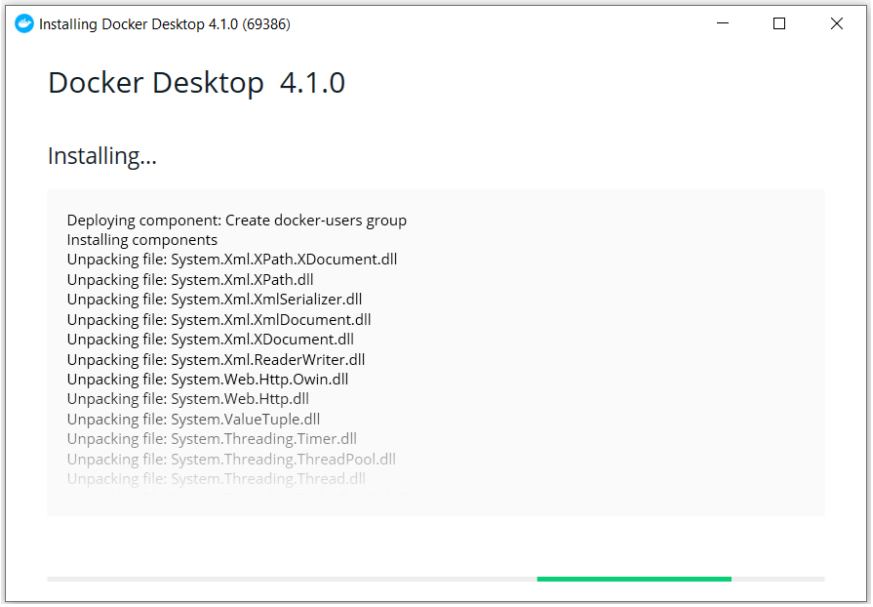


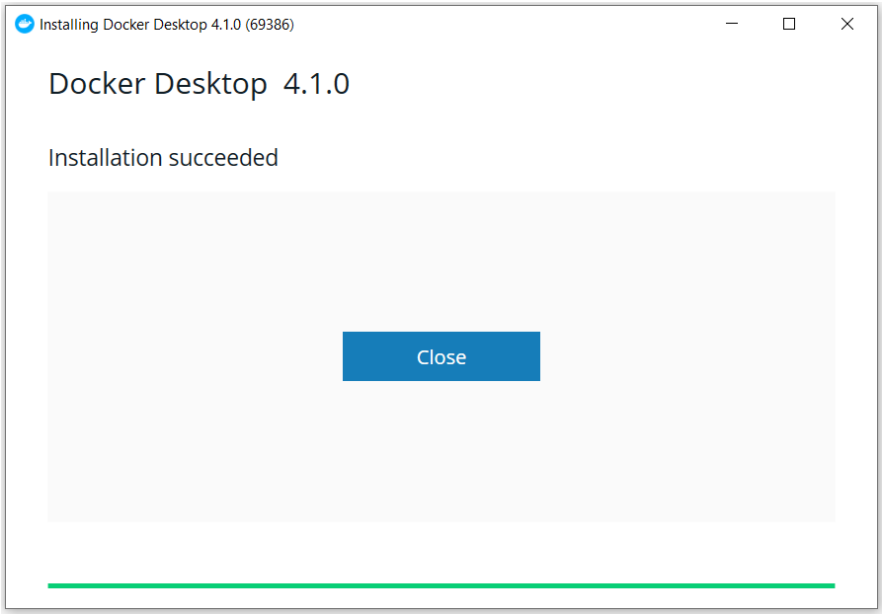
6. Run the Docker Desktop Installer and accept default Configuration settings:



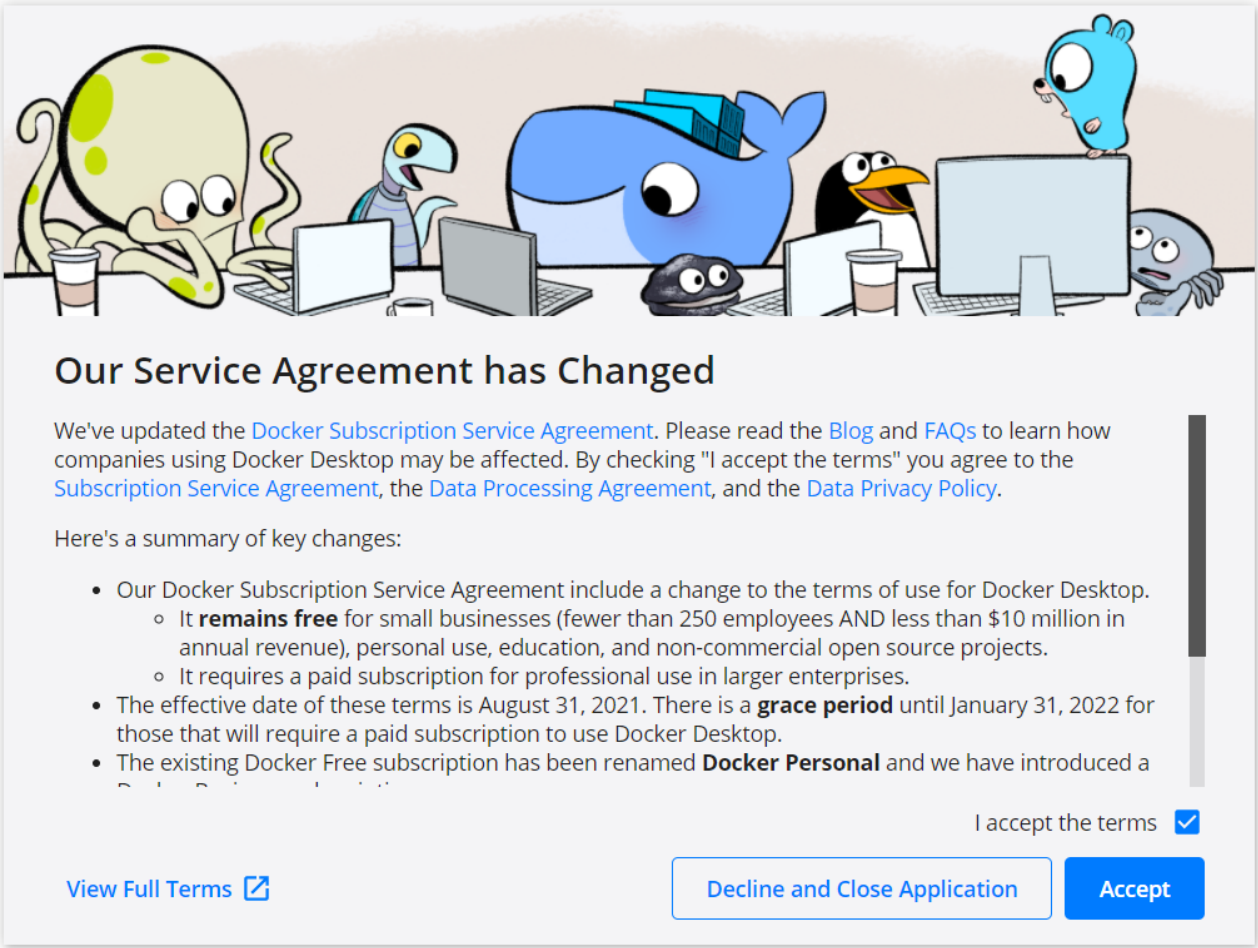
7. After Unpacking files... and Installing.. the Docker, you will see Installation Succeeded message. Close the window, now on you can start using the Docker:


Installing Docker Desktop 4.1.0 (69386)
Docker Desktop 4.1.0
Unpacking files...
Unpacking file: frontend/Docker Desktop.exe
Unpacking file: frontend/d3dcompiler_47.dll
Unpacking file: DockerCli.exe
Unpacking file: Docker.WPF.dll
Unpacking file: Docker.HttpApi.dll
Unpacking file: Docker.Engines.dll
Unpacking file: Docker.Core.dll
Unpacking file: Docker.Backend.dll
Unpacking file: Docker.ApiServices.dll
Unpacking file: Docker Desktop.exe
Unpacking file: Docker Desktop Installer.exe
Unpacking file: courgette64.exe
Unpacking file: Bugsnag.dll
Unpacking file: BITSReference5_0.dll

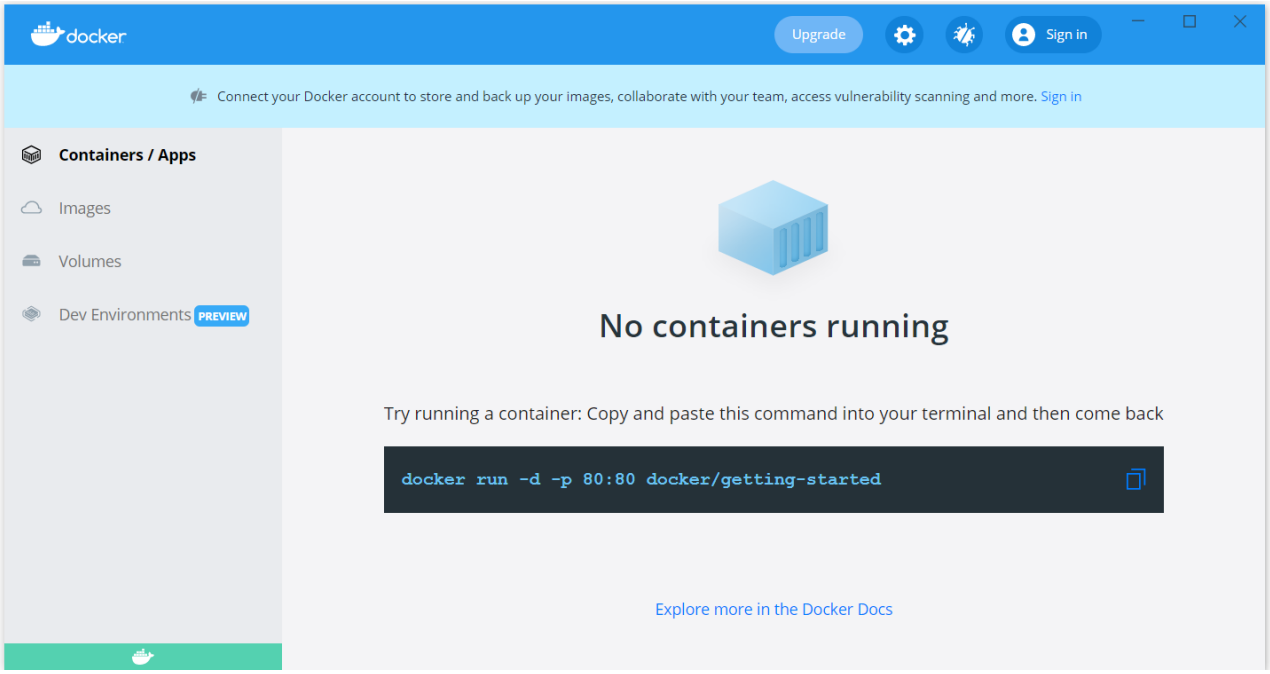

Installing Docker Desktop 4.1.0 (69386)
Docker Desktop 4.1.0
Installing...
Deploying component: Create docker-users group
Installing components
Unpacking file: System.Xml.XPath.XDocument.dll
Unpacking file: System.Xml.XPath.dll
Unpacking file: System.Xml.XmlSerializer.dll
Unpacking file: System.Xml.XmlDocument.dll
Unpacking file: System.Xml.XDocument.dll
Unpacking file: System.Xml.ReaderWriter.dll
Unpacking file: System.Web.Http.Owin.dll
Unpacking file: System.Web.Http.dll
Unpacking file: System.ValueTuple.dll
Unpacking file: System.Threading.Timer.dll
Unpacking file: System.Threading.ThreadPool.dll
Unpacking file: System.Threading.Thread.dll


Installing Docker Desktop 4.1.0 (69386)
Docker Desktop 4.1.0
Installation succeeded
Close

8. Open Docker Desktop application and accept the terms:



9. You end up with the following screen which states that there are no running containers:



10. Open a terminal window and create a directory (e.g., named as DockerTest) with mkdir DockerTest command. Change your working directory as DockerTest with the cd DockerTest command. You can get a list of docker images (docker images ls) and containers (docker ps --all):

```
Komut İstemi

D:\>mkdir DockerTest

D:\>cd DockerTest

D:\DockerTest>docker images ls
REPOSITORY    TAG        IMAGE ID    CREATED    SIZE

D:\DockerTest>docker ps --all
CONTAINER ID    IMAGE      COMMAND    CREATED    STATUS     PORTS      NAMES

D:\DockerTest>
```

11. Put the dockerfile, distributed over ninova to the DockerTest directory.

12. Run the following command to build the docker image.

```
docker build -t <image_name> .
```

- Pay attention to the dot at the end of the command
- You may use an arbitrary name for your image. The name of the image in our example is ayar/blg223e (i.e., we used the docker build -t ayar/blg223e . command)
- It should take a while to build the image, a screen similar to below should appear at the end:

```
Komut İstemi
[+] Building 951.1s (20/20) FINISHED
 => [internal] load build definition from Dockerfile                                                    0.0s
 => => transferring dockerfile: 539B                                                                    0.0s
 => [internal] load .dockerignore                                                                       0.0s
 => => transferring context: 2B                                                                         0.0s
 => [internal] load metadata for docker.io/library/ubuntu:latest                                       22.7s
 => [ 1/16] FROM docker.io/library/ubuntu:latest@sha256:44ab2c3b26363823dcb965498ab06abf74a1e6af20a732902250743df0d4172d   144.5s
 => => resolve docker.io/library/ubuntu:latest@sha256:44ab2c3b26363823dcb965498ab06abf74a1e6af20a732902250743df0d4172d     0.0s
 => => sha256:44ab2c3b26363823dcb965498ab06abf74a1e6af20a732902250743df0d4172d 1.42kB / 1.42kB                0.0s
 => => sha256:3555f4996aea6be945ae1532fa377c88f4b3b9e6d93531f47af5d78a7d5e3761 529B / 529B                    0.0s
 => => sha256:597ce1600cf4ac5f449b66e75e840657bb53864434d6bd82f00b172544c32ee2 1.46kB / 1.46kB                0.0s
 => => sha256:f3ef4ff62e0da0ef761ec1c8a578f3035bef51043e53ae1b13a20b3e03726d17 28.57MB / 28.57MB             142.9s
 => => extracting sha256:f3ef4ff62e0da0ef761ec1c8a578f3035bef51043e53ae1b13a20b3e03726d17                      1.4s
 => [ 2/16] RUN apt update                                                                             109.9s
 => [ 3/16] RUN apt install openssh-server sudo -y                                                     235.4s
 => [ 4/16] RUN useradd -rm -d /home/ubuntu -s /bin/bash -g root -G sudo -u 1000 test                    0.7s
 => [ 5/16] RUN usermod -aG sudo test                                                                    0.7s
 => [ 6/16] RUN service ssh start                                                                        0.7s
 => [ 7/16] RUN  echo 'test:test' | chpasswd                                                             0.7s
 => [ 8/16] RUN apt update                                                                               2.5s
 => [ 9/16] RUN apt install build-essential -y                                                         216.8s
 => [10/16] RUN apt install gdb -y                                                                      66.4s
 => [11/16] RUN apt update                                                                              13.5s
 => [12/16] RUN apt install git -y                                                                      58.0s
 => [13/16] RUN apt update                                                                              12.6s
 => [14/16] RUN apt install python3 -y                                                                   2.0s
 => [15/16] RUN apt install python3-pip -y                                                              41.2s
 => [16/16] RUN pip install calico                                                                      21.0s
 => exporting to image                                                                                   1.7s
 => => exporting layers                                                                                  1.7s
 => => writing image sha256:07797bf555e346678f594f5df1321acf2c24117596fc7b0f6586a60940d37287              0.0s
 => => naming to docker.io/ayar/blg223e                                                                  0.0s

Use 'docker scan' to run Snyk tests against images to find vulnerabilities and learn how to fix them

D:\DockerTest>
```

13. Check if the image has been built with the docker image ls command:

```
Komut İstemi

D:\DockerTest>docker image ls
REPOSITORY      TAG       IMAGE ID       CREATED          SIZE
ayar/blg223e    latest    07797bf555e3   22 minutes ago   632MB

D:\DockerTest>
```

14. Generate a directory to put the files that would be visible to your container. Let's call this directory's path as <dev_path>

15. 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 `container_port` of the container to one of your pc `local_port`s. This is required to be able to ssh into your container
- `-v <dev_path>:<container_path>`: Maps your local files to a directory inside the container
- The rest is self explanatory.
- It should almost instantly boot the container, a screen similar to below should appear

```
Komut İstemi

D:\DockerTest>docker run -p 2222:22 -v D:\DockerTest\Volume:/home/ubuntu/hostvolume --name blg223e --hostname blg223e -d ayar/blg223e
d9006a6795d89d9965b7e30027a69e86a66cc3e8ebaa7a9d376468edba4d91cc

D:\DockerTest>_
```

- You may perform an additional check by running the `docker ps --all` command

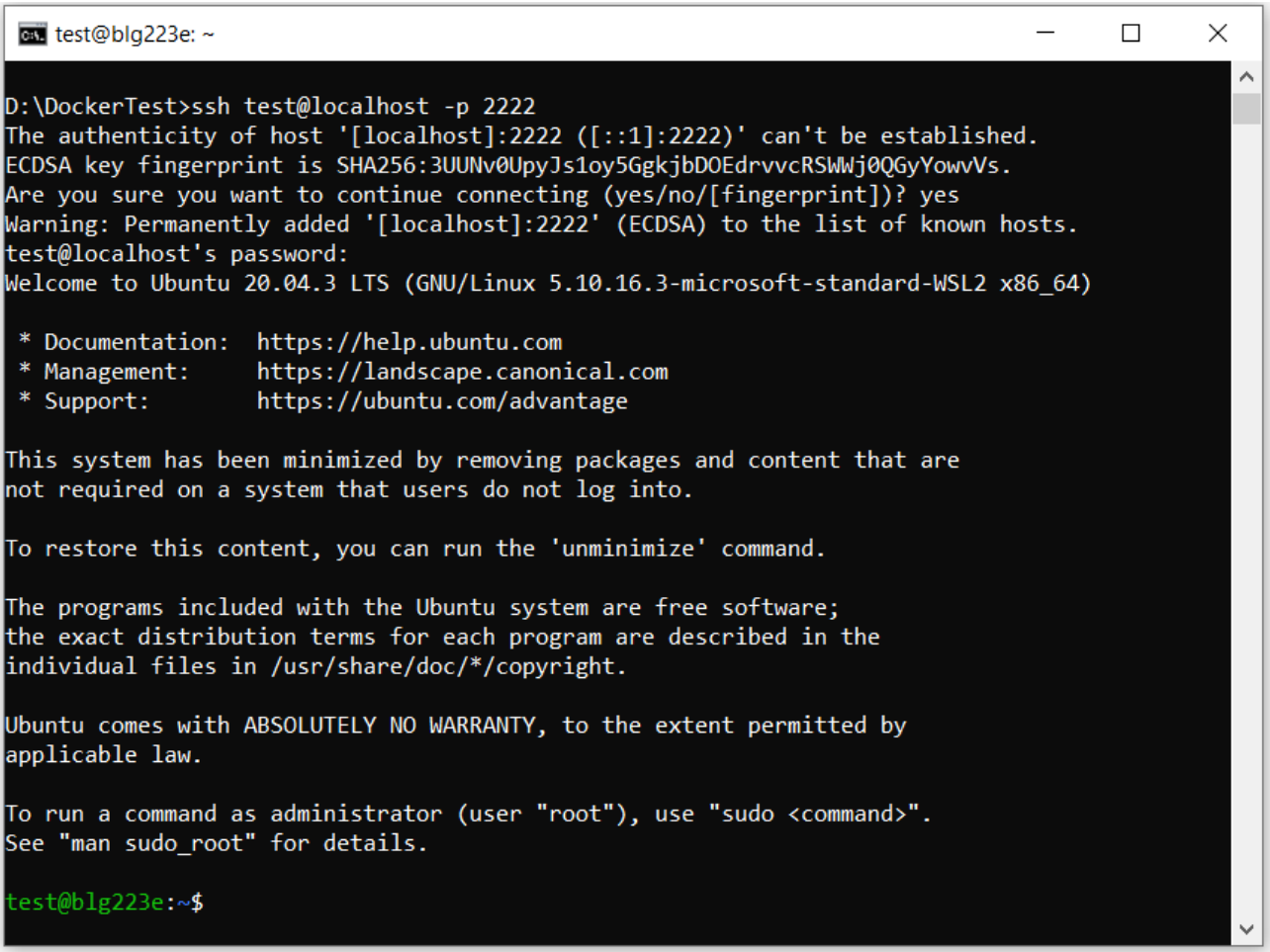
```
Komut İstemi

D:\DockerTest>docker ps -all
CONTAINER ID   IMAGE          COMMAND                  CREATED          STATUS          PORTS                    NAMES
d9006a6795d8   ayar/blg223e   "/usr/sbin/sshd -D"      4 minutes ago    Up 4 minutes    0.0.0.0:2222->22/tcp     blg223e

D:\DockerTest>_
```

16. Check if you can ssh into your container with the ssh username@host_name -p <local_port> command as follows:

ssh test@localhost -p 2222

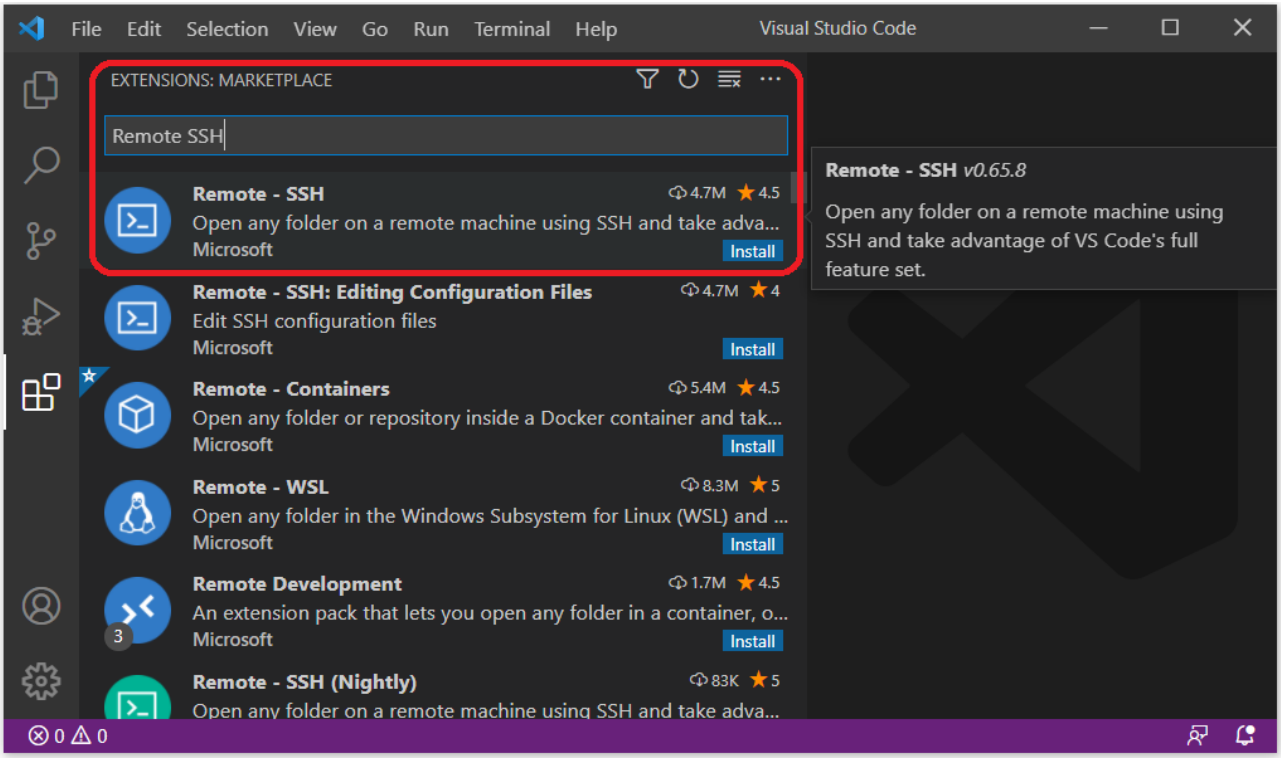
- Your container's username and password both are set as test. You may change it by editing the dockerfile and rebuilding the image and re-running the container.

- Don't forget to exit from the ssh session.

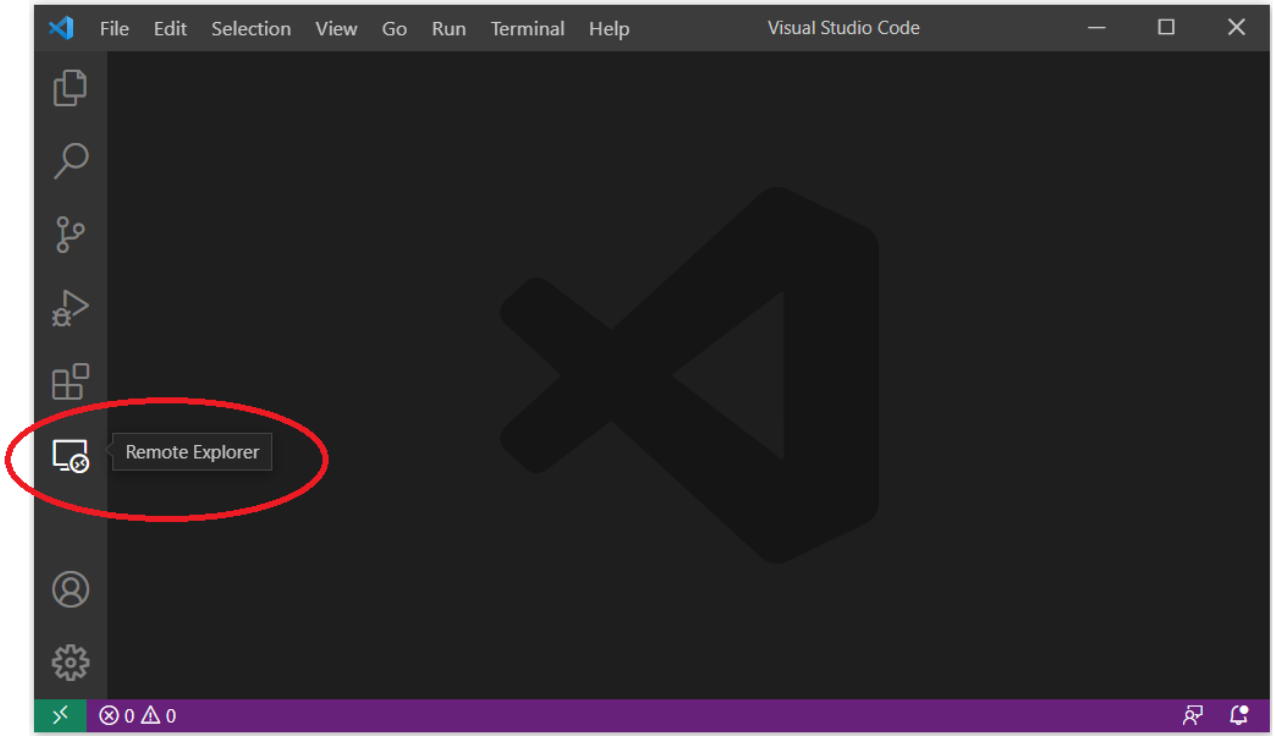


17. Install Visual Studio Code in your system. (!!!Be careful: Visual Studio and Visual Studio Code are different.)

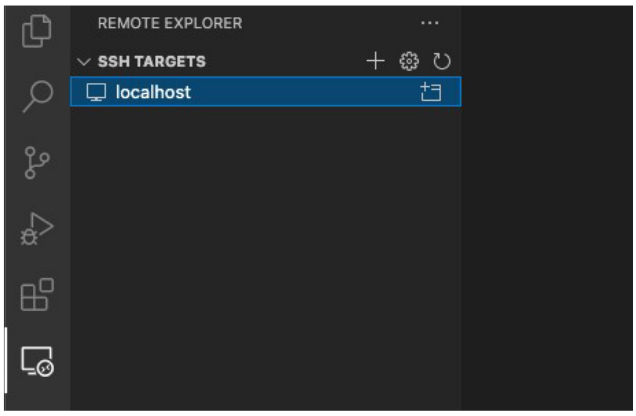
18. Run Visual Studio Code and open EXTENSIONS MARKETPLACE by pressing CTRL+SHIFT+X. Then, search for the extension named Remote SSH and install it:



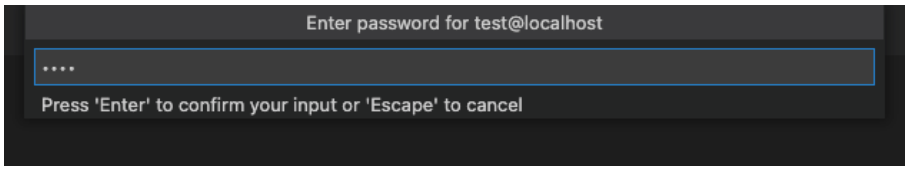
19. After installing the extension, you will see a newly added Remote Eplorer button.



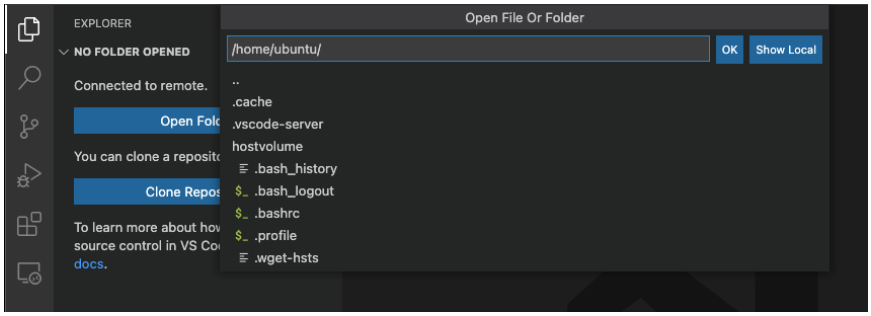
20. Click on the newly added “Remote Explorer” button and make a connection with localhost, If there is no ssh target you have to add new.



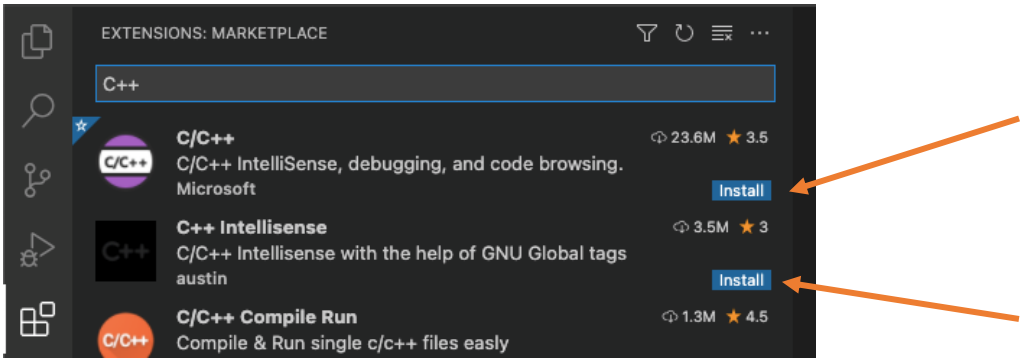
21. A new window should open and ask for password. If you didn’t change anything, your default should be test



22. When ready, check if you can open up your local folder from the container as follows



23. In the new window, we should install a few more extensions for C++ development. Search for C++ among the extensions and install C/C++ and C++ Intellisense



24. Afterwards you can copy your development files in your local <dev_path> and continue developing.