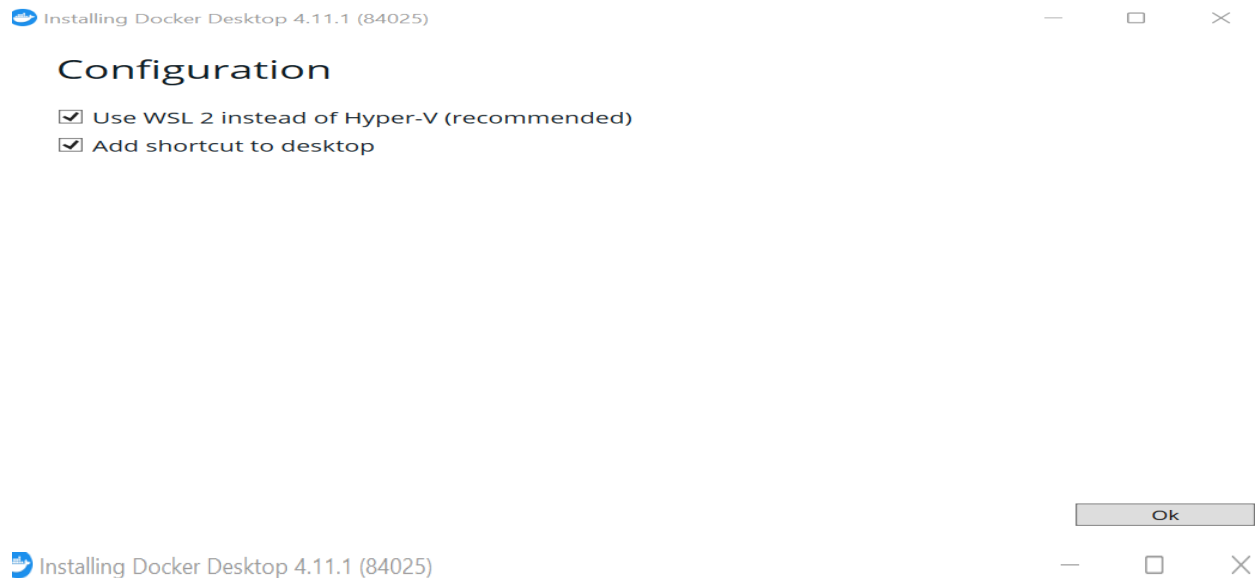


Codio Activity: Installing Docker Containers

Install Docker:



Docker Desktop 4.11.1

Unpacking files...

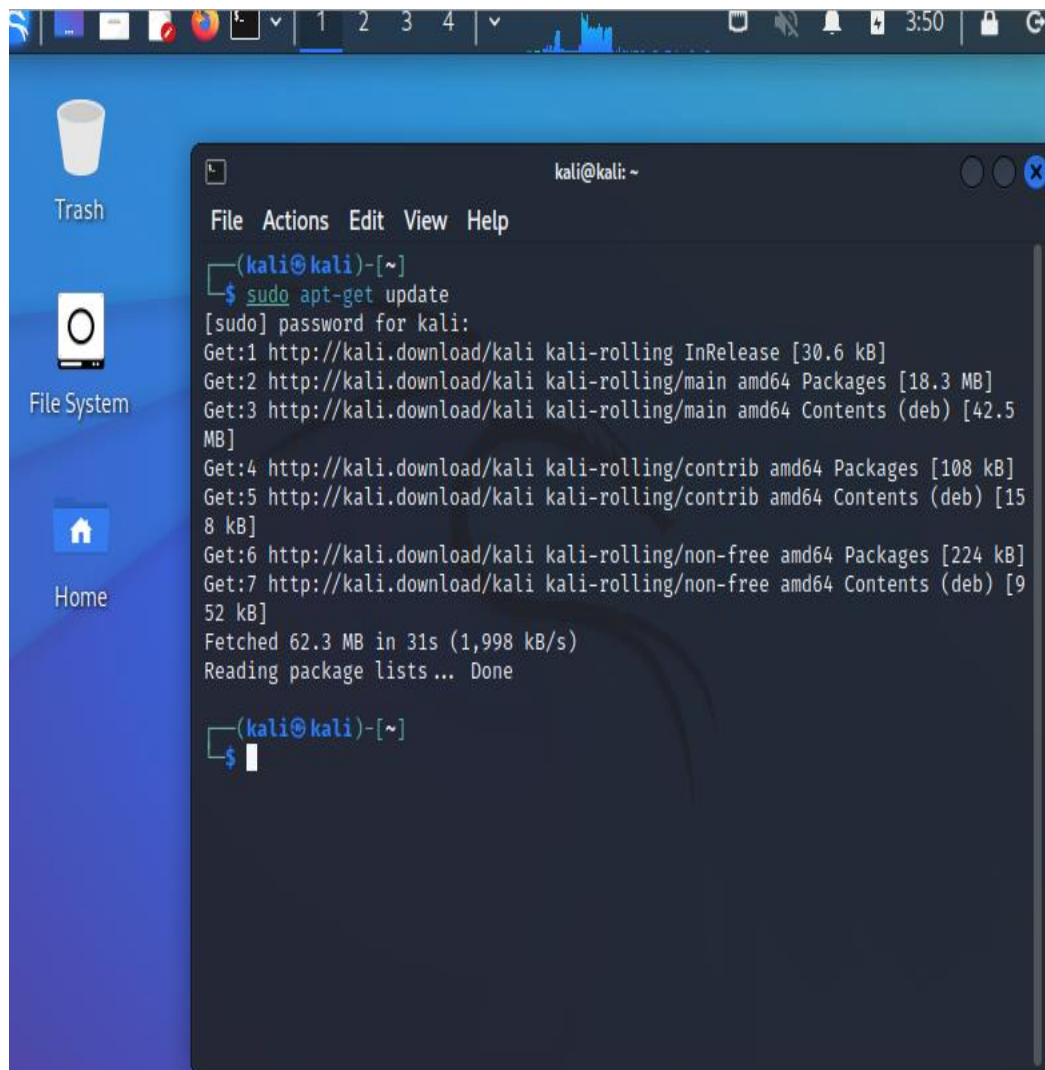
```
Unpacking file: resources/docker-desktop.iso
Unpacking file: resources/ddvp.ico
Unpacking file: resources/config-options.json
Unpacking file: resources/componentsVersion.json
Unpacking file: resources/bin/docker-compose
Unpacking file: resources/bin/docker
Unpacking file: resources/.gitignore
Unpacking file: InstallerCli.pdb
Unpacking file: InstallerCli.exe.config
Unpacking file: frontend/vk_swiftshader_icd.json
Unpacking file: frontend/v8_context_snapshot.bin
Unpacking file: frontend/snapshot_blob.bin
Unpacking file: frontend/resources/regedit/vbs/util.vbs
Unpacking file: frontend/resources/regedit/vbs/regUtil.vbs
```

Required Virtual Box and Ubuntu



1. Perform a standard update to ensure you are using the latest libraries:

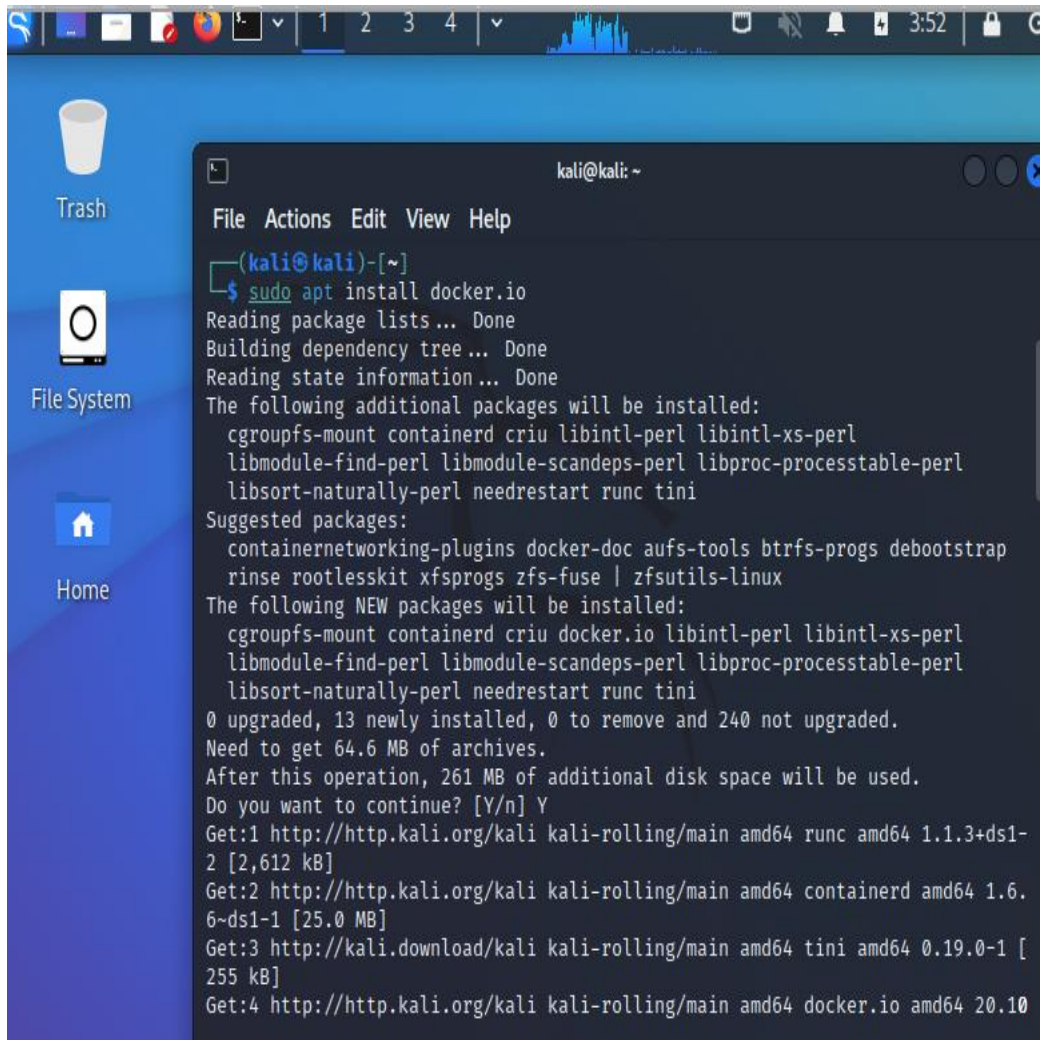
sudo apt-get update

A screenshot of a Kali Linux desktop environment. The desktop background is blue. On the left side, there are icons for 'Trash', 'File System', and 'Home'. A terminal window is open in the center, displaying the output of the command 'sudo apt-get update'. The terminal window has a title bar that says 'kali@kali: ~'. The output of the command shows the progress of updating the package lists from various repositories, including kali-rolling, main amd64, contrib amd64, and non-free amd64. The total size of the fetched data is 62.3 MB, and it took 31 seconds to fetch at a rate of 1,998 kB/s. The terminal window also shows the prompt '(kali@kali)-[~]' and a dollar sign '\$' indicating the root user's shell.

```
kali@kali: ~  
File Actions Edit View Help  
(kali@kali)-[~]  
$ sudo apt-get update  
[sudo] password for kali:  
Get:1 http://kali.download/kali kali-rolling InRelease [30.6 kB]  
Get:2 http://kali.download/kali kali-rolling/main amd64 Packages [18.3 MB]  
Get:3 http://kali.download/kali kali-rolling/main amd64 Contents (deb) [42.5 MB]  
Get:4 http://kali.download/kali kali-rolling/contrib amd64 Packages [108 kB]  
Get:5 http://kali.download/kali kali-rolling/contrib amd64 Contents (deb) [15 8 kB]  
Get:6 http://kali.download/kali kali-rolling/non-free amd64 Packages [224 kB]  
Get:7 http://kali.download/kali kali-rolling/non-free amd64 Contents (deb) [9 52 kB]  
Fetched 62.3 MB in 31s (1,998 kB/s)  
Reading package lists... Done  
(kali@kali)-[~]  
$
```

2. Install Docker from the standard repository:

sudo apt install docker.io



```
kali@kali: ~  
File Actions Edit View Help  
(kali@kali)-[~]  
$ sudo apt install docker.io  
Reading package lists... Done  
Building dependency tree... Done  
Reading state information... Done  
The following additional packages will be installed:  
  cgroupfs-mount containerd criu libintl-perl libintl-xs-perl  
  libmodule-find-perl libmodule-scandeps-perl libproc-processtable-perl  
  libsort-naturally-perl needrestart runc tini  
Suggested packages:  
  containernetworking-plugins docker-doc aufs-tools btrfs-progs debootstrap  
  rinse rootlesskit xfsprogs zfs-fuse | zfsutils-linux  
The following NEW packages will be installed:  
  cgroupfs-mount containerd criu docker.io libintl-perl libintl-xs-perl  
  libmodule-find-perl libmodule-scandeps-perl libproc-processtable-perl  
  libsort-naturally-perl needrestart runc tini  
0 upgraded, 13 newly installed, 0 to remove and 240 not upgraded.  
Need to get 64.6 MB of archives.  
After this operation, 261 MB of additional disk space will be used.  
Do you want to continue? [Y/n] Y  
Get:1 http://http.kali.org/kali kali-rolling/main amd64 runc amd64 1.1.3+ds1-  
2 [2,612 kB]  
Get:2 http://http.kali.org/kali kali-rolling/main amd64 containerd amd64 1.6.  
6~ds1-1 [25.0 MB]  
Get:3 http://kali.download/kali kali-rolling/main amd64 tini amd64 0.19.0-1 [  
255 kB]  
Get:4 http://http.kali.org/kali kali-rolling/main amd64 docker.io amd64 20.10
```

3. As part of the installation process, Docker will create a new group (called docker). You need to add your username to that group:

```
sudo usermod -aG docker [ali]
```

4. Log out and then back in to update your user status.

5. Check that you are a member of the docker group:

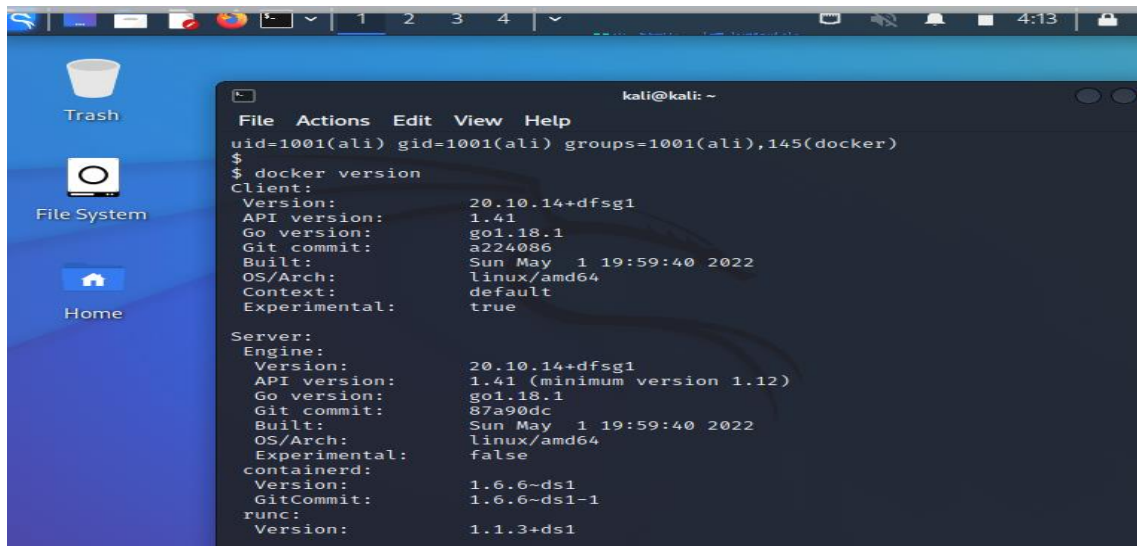
id (produces the following output:

```
kali@kali: ~  
File Actions Edit View Help  
(kali@kali)-[~]  
$ ls -al /home  
total 12  
drwxr-xr-x 3 root root 4096 Aug 8 06:10 .  
drwxr-xr-x 18 root root 4096 Aug 8 06:28 ..  
drwxr-xr-x 15 kali kali 4096 Aug 28 03:59 kali  
  
(kali@kali)-[~]  
$ sudo useradd -m ali  
  
(kali@kali)-[~]  
$ sudo usermod -aG docker ali  
  
(kali@kali)-[~]  
$ su ali  
Password:  
su: Authentication failure  
  
(kali@kali)-[~]  
$ su ali  
Password:  
su: Authentication failure  
  
(kali@kali)-[~]  
$ sudo passwd ali  
New password:  
Retype new password:
```

```
kali@kali: ~  
File Actions Edit View Help  
(kali@kali)-[~]  
$ sudo usermod -aG docker ali  
  
(kali@kali)-[~]  
$ su ali  
Password:  
su: Authentication failure  
  
(kali@kali)-[~]  
$ su ali  
Password:  
su: Authentication failure  
  
(kali@kali)-[~]  
$ sudo passwd ali  
New password:  
Retype new password:  
passwd: password updated successfully  
  
(kali@kali)-[~]  
$ su ali  
Password:  
$ whoami  
ali  
$ id  
uid=1001(ali) gid=1001(ali) groups=1001(ali),145(docker)  
$
```

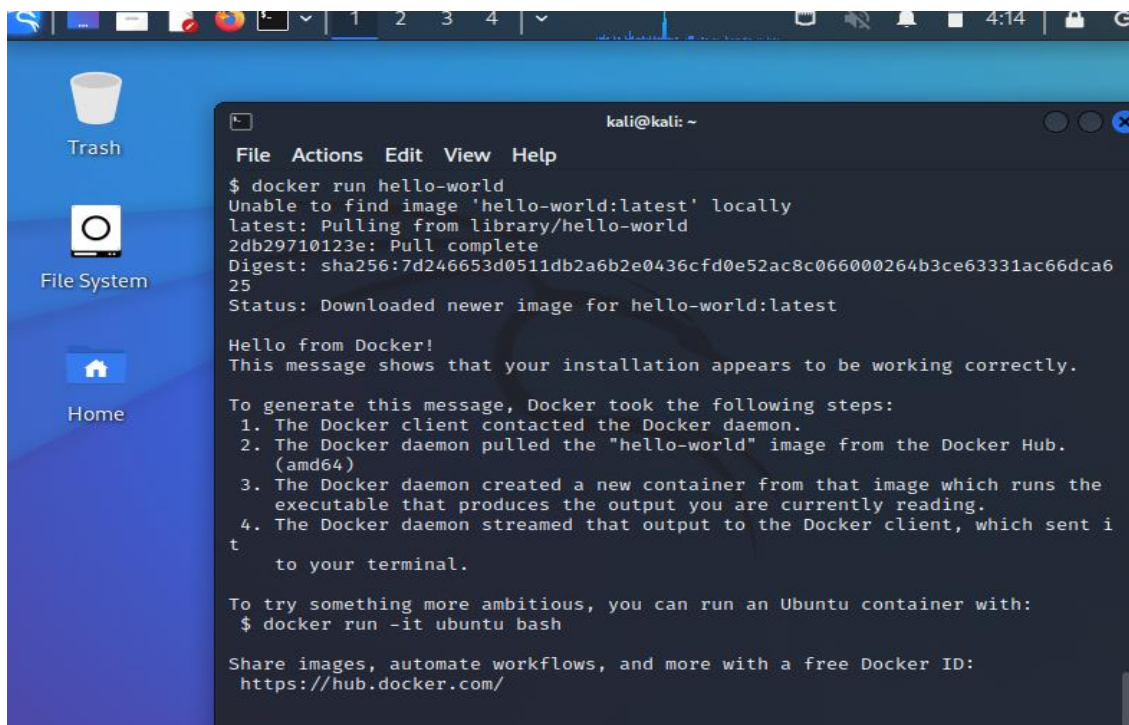

6. Issue the version command to check that docker is working properly:

docker version (produces the output shown below:



```
kali@kali: ~  
File Actions Edit View Help  
uid=1001(ali) gid=1001(ali) groups=1001(ali),145(docker)  
$ docker version  
Client:  
Version:          20.10.14+dfsg1  
API version:      1.41  
Go version:       go1.18.1  
Git commit:       a224086  
Built:            Sun May 1 19:59:40 2022  
OS/Arch:          linux/amd64  
Context:          default  
Experimental:     true  
  
Server:  
Engine:  
Version:          20.10.14+dfsg1  
API version:      1.41 (minimum version 1.12)  
Go version:       go1.18.1  
Git commit:       87a90dc  
Built:            Sun May 1 19:59:40 2022  
OS/Arch:          linux/amd64  
Experimental:     false  
containerd:  
Version:          1.6.6-ds1  
GitCommit:        1.6.6-ds1-1  
runc:  
Version:          1.1.3+ds1
```

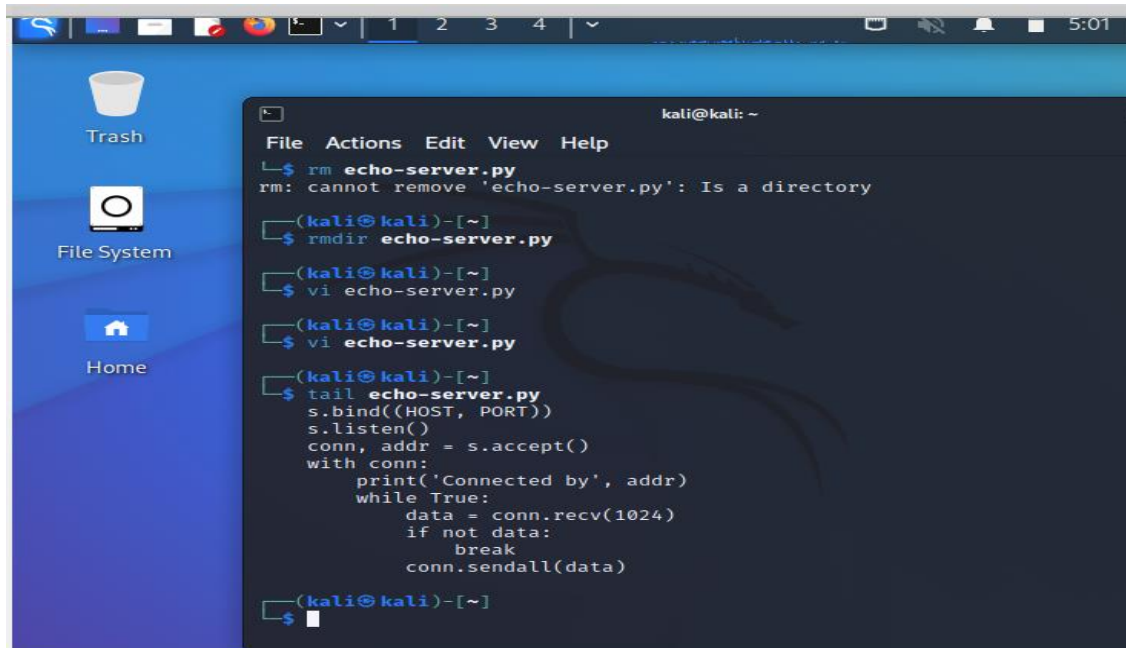
7. You can also run the "docker run hello-world" command to test a (very simple) container.



```
kali@kali: ~  
File Actions Edit View Help  
$ docker run hello-world  
Unable to find image 'hello-world:latest' locally  
latest: Pulling from library/hello-world  
2db29710123e: Pull complete  
Digest: sha256:7d246653d0511db2a6b2e0436cfd0e52ac8c066000264b3ce63331ac66dca625  
Status: Downloaded newer image for hello-world:latest  
  
Hello from Docker!  
This message shows that your installation appears to be working correctly.  
  
To generate this message, Docker took the following steps:  
1. The Docker client contacted the Docker daemon.  
2. The Docker daemon pulled the "hello-world" image from the Docker Hub.  
   (amd64)  
3. The Docker daemon created a new container from that image which runs the  
   executable that produces the output you are currently reading.  
4. The Docker daemon streamed that output to the Docker client, which sent it  
   to your terminal.  
  
To try something more ambitious, you can run an Ubuntu container with:  
$ docker run -it ubuntu bash  
  
Share images, automate workflows, and more with a free Docker ID:  
https://hub.docker.com/
```

Codio Activity: Socket Programming

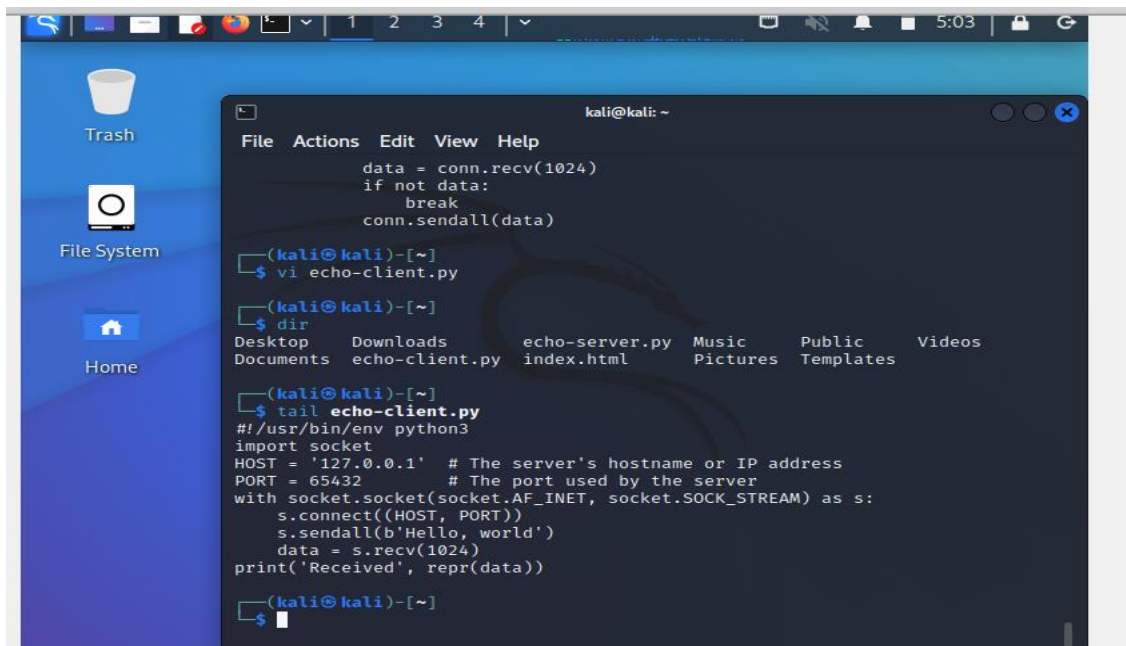
Copy the following code into a file named echo-server.py



The screenshot shows a Kali Linux desktop environment with a terminal window open. The terminal displays the following commands and output:

```
kali@kali: ~  
File Actions Edit View Help  
$ rm echo-server.py  
rm: cannot remove 'echo-server.py': Is a directory  
(kali@kali)-[~]  
$ rmdir echo-server.py  
(kali@kali)-[~]  
$ vi echo-server.py  
(kali@kali)-[~]  
$ vi echo-server.py  
(kali@kali)-[~]  
$ tail echo-server.py  
s.bind((HOST, PORT))  
s.listen()  
conn, addr = s.accept()  
with conn:  
    print('Connected by', addr)  
    while True:  
        data = conn.recv(1024)  
        if not data:  
            break  
        conn.sendall(data)  
(kali@kali)-[~]  
$
```

Copy the following code into a file named echo-client.py



The screenshot shows a Kali Linux desktop environment with a terminal window open. The terminal displays the following commands and output:

```
kali@kali: ~  
File Actions Edit View Help  
data = conn.recv(1024)  
if not data:  
    break  
conn.sendall(data)  
(kali@kali)-[~]  
$ vi echo-client.py  
(kali@kali)-[~]  
$ dir  
Desktop Downloads echo-server.py Music Public Videos  
Documents echo-client.py index.html Pictures Templates  
(kali@kali)-[~]  
$ tail echo-client.py  
#!/usr/bin/env python3  
import socket  
HOST = '127.0.0.1' # The server's hostname or IP address  
PORT = 65432 # The port used by the server  
with socket.socket(socket.AF_INET, socket.SOCK_STREAM) as s:  
    s.connect((HOST, PORT))  
    s.sendall(b'Hello, world')  
    data = s.recv(1024)  
    print('Received', repr(data))  
(kali@kali)-[~]  
$
```

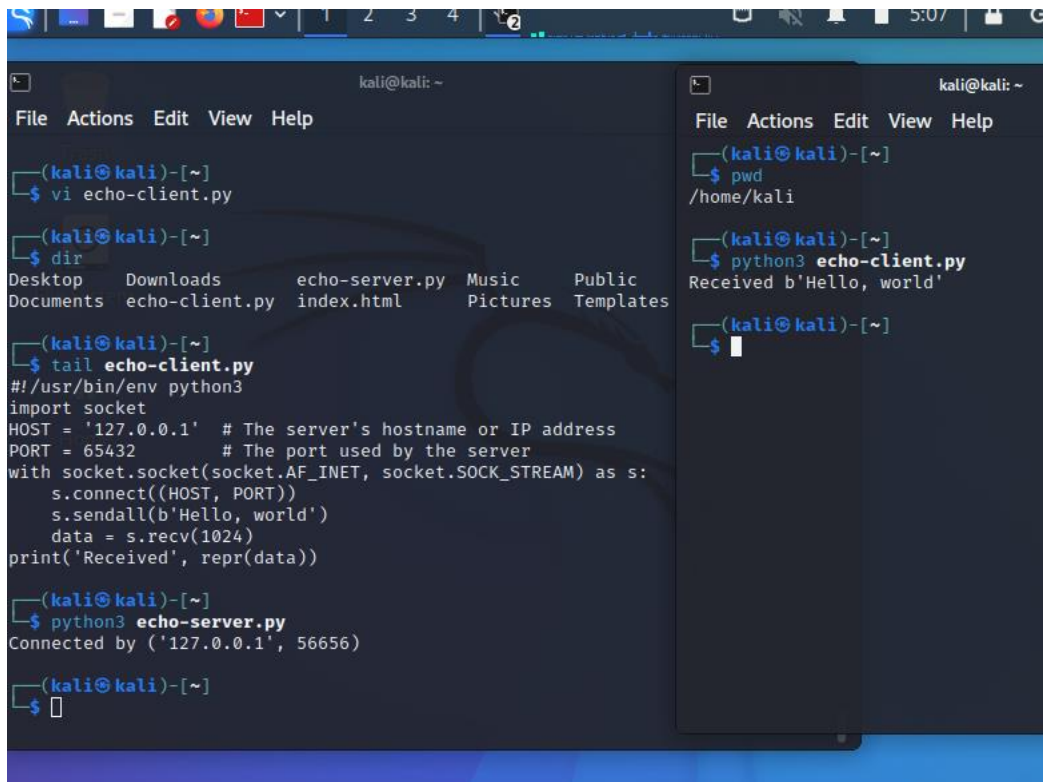
Open a terminal. Start the server by running the following command in a terminal:

```
python3 echo-server.py
```

Open a second terminal. Start the client by running the following command in the terminal:

```
python3 echo-client.py
```

The client and server will now talk with one another.



```
kali@kali: ~  
File Actions Edit View Help  
(kali@kali)-[~]  
$ vi echo-client.py  
(kali@kali)-[~]  
$ dir  
Desktop Downloads echo-server.py Music Public  
Documents echo-client.py index.html Pictures Templates  
(kali@kali)-[~]  
$ tail echo-client.py  
#!/usr/bin/env python3  
import socket  
HOST = '127.0.0.1' # The server's hostname or IP address  
PORT = 65432 # The port used by the server  
with socket.socket(socket.AF_INET, socket.SOCK_STREAM) as s:  
    s.connect((HOST, PORT))  
    s.sendall(b'Hello, world')  
    data = s.recv(1024)  
    print('Received', repr(data))  
(kali@kali)-[~]  
$ python3 echo-server.py  
Connected by ('127.0.0.1', 56656)  
(kali@kali)-[~]  
$  
kali@kali: ~  
File Actions Edit View Help  
(kali@kali)-[~]  
$ pwd  
/home/kali  
(kali@kali)-[~]  
$ python3 echo-client.py  
Received b'Hello, world'  
(kali@kali)-[~]  
$
```

Question 1

In relation to echo-server.py, what is achieved using the command:

```
s.bind((HOST, PORT))?
```

making a Reverse Shell. and in the server.py file i got this error. i has trying in de socket_bind() s.bind((host, port))

Question 2

In relation to echo-server.py, what is achieved using the command:

`s.connect((HOST, PORT))`?

Used to connect to the remote socket with connect.

Codio Activity: Producer-Consumer Mechanism

Producer/Consumer Problem (also known as the 'bounded buffer' problem):

- A 'producer' is producing items at a particular (unknown and sometimes unpredictable) rate.
- A 'consumer' is consuming the items – again, at some rate.

For example, a producer-consumer scenario models an application producing a listing that must be consumed by a printer process, as well as a keyboard handler producing a line of data that will be consumed by an application program. This is shown in the picture below (Shene, 2014).

Items are placed in a buffer when produced, so:

- Consumer should wait if there isn't an item to consume
- Producer shouldn't 'overwrite' an item in the buffer

Synchronisation is necessary because:

- If the consumer has not taken out the current value in the buffer, then the producer should not replace it with another.
- Similarly, the consumer should not consume the same value twice.

Task

Run producer-consumer.py in the provided Codio workspace (**Producer-Consumer Mechanism**), where the queue data structure is used.

A copy of the code is available here for you.

code source: <https://techmonger.github.io/55/producer-consumer-python/>

```
from threading import Thread
```

```

from queue import Queue

q = Queue()
final_results = []

def producer():
    for i in range(100):
        q.put(i)

def consumer():
    while True:
        number = q.get()
        result = (number, number**2)
        final_results.append(result)
        q.task_done()

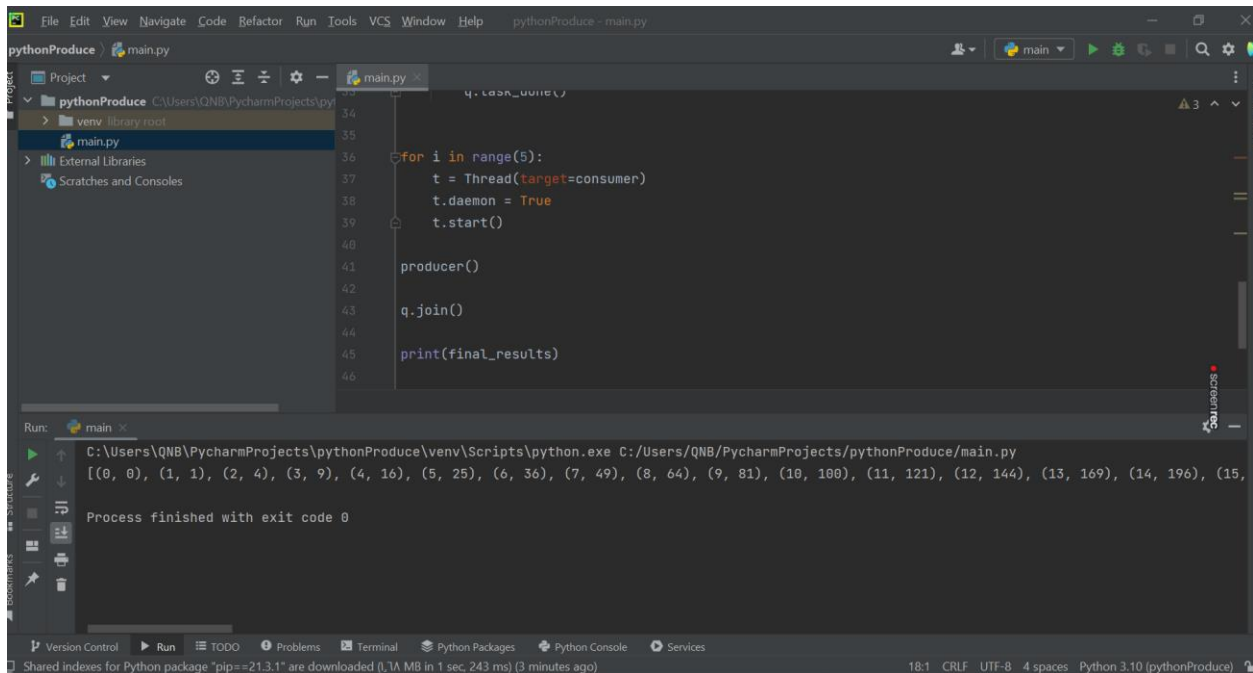
for i in range(5):
    t = Thread(target=consumer)
    t.daemon = True
    t.start()

producer()

q.join()

print (final_results)

```



The screenshot shows the PyCharm IDE with a project named 'pythonProduce'. The file 'main.py' is open, displaying the following code:

```

from queue import Queue

q = Queue()
final_results = []

def producer():
    for i in range(100):
        q.put(i)

def consumer():
    while True:
        number = q.get()
        result = (number, number**2)
        final_results.append(result)
        q.task_done()

for i in range(5):
    t = Thread(target=consumer)
    t.daemon = True
    t.start()

producer()

q.join()

print (final_results)

```

The Run window at the bottom shows the command executed: `C:\Users\QNB\PycharmProjects\pythonProduce\venv\Scripts\python.exe C:\Users\QNB\PycharmProjects\pythonProduce/main.py`. The output is a list of 100 pairs of (number, number squared), starting from (0, 0) and ending with (15, 225). The process finished with exit code 0.

Answer the following questions:

1. How is the queue data structure used to achieve the purpose of the code?

It's used to fill the produce data queue – in this case the numbers from 0 to 99. The program can then generate multiple consumer (threads) to process them.

2. What is the purpose of `q.put()`?

This is to fill the queue with the generated produce (this is from 0 to 99).

3. What is achieved by `q.get()`?

To get the contents of the queue, this is first in first out, thus the multiple instantiated consumer threads will process them from 0 to 99.

4. What functionality is provided by `q.join()`?

`q.join` prevents the python program to exit until all instantiated threads are finished (e.g. 0 to 99 produce has been consumed/calculated - ^2).

5. Extend this producer-consumer code to make the producer-consumer scenario available in a secure way. What technique(s) would be appropriate to apply? I don't get this question, by secure other threads must not know what others are working on? In this case we can randomize the producer generated number and/or implement encryptions (public/private keys).