(Q1): From the terminal, type in the following commands and take snapshots after running each command. To validate the installation is correct, type $ docker run hello-world

(A1):

A screenshot of a cell phone

Description automatically generated

(Q2): Run a Busybox container. The pull command fetches the busybox image from the Docker registry and saves it to the VM. $ docker pull busybox

(A2):

A screenshot of a social media post

Description automatically generated

(Q3): Find available docker images in your VM.

$ docker images

Check that busybox container is [listed];

(A3):

A screenshot of a social media post

Description automatically generated

(Q4): Run the Docker container busybox. When you run the Docker image bosybox, Docker loads up the container and then runs a command in the container. When we run “docker run busybox”, there is no command, so Docker did nothing and exited.

$ docker run busybox

If you provide a command after “docker run busybox”, that command will be executed

$ docker run busybox echo “hello world from Sophia”

Replace ‘Sophia’ with your name. That way I knew that you are running Docker by yourself. Take a snapshot.

(A4):

A screenshot of a cell phone

Description automatically generated

(Q5): Run the following command to check what containers are currently running.

$ docker ps

It should be no docker running.

Run the following command to see a list of all comtainers that we ran. Notice the STATUS column shows that these containers exited a few minutes ago. Take a snapshot.

$ docker ps –a

Run the following local host command to see the difference with and without Docker.

$ ps -a

(A5):

1. Screenshot for $ docker ps and $ docker ps –aA screenshot of a cell phone

   Description automatically generated
2. Screenshot for $ ps -a – **Part 1**A screenshot of a cell phone

   Description automatically generated

**Part 2**A screenshot of a cell phone

Description automatically generated

(Q6): Compare Virtual Machine (VM), Container, and Sandbox with the following features in the table.

(A6):

|  |  |  |  |
| --- | --- | --- | --- |
| Features\Types | VM | Container | Sandbox |
| Same OS as host OS | No | Yes | Yes |
| Share data with host | No | No | No |
| Data remains when the VM / Container / Sandbox is closed | Yes | No | No |
| Additional OS overhead | Yes | No | No |
| Performance | Low performance | Best performance | Relatively lower performance than containers |
| Storage space | High | Lightweight | Lightweight |
| Communication among the applications and host | No communication with host software |  |  |
| Security level | Highly secured since the entire system is isolated from the host | Relatively less secure than VMs, however more secure than the sandboxes. Unlike sandboxes, they do not evaluate the application files before isolating them. | Less secured than VM and Containers |
| Examples of applications | VMWare, VirtualBox | Docker, Turbo | Google sandboxed API, Chrome |
| Advantages | 1-Highly secure  2-Very flexible with the OS. Any OS VM can be run on any host.  3-Multiple VMs with different OS can be hosted on a single platform. | 1-Lightweight  2-Very less resource intensive  3-Cost effective  4-Perfect for microservices | 1-Lightweight  2-Suitable for quarantining threats |
| Disadvantages | 1-Heavy weight, since every VM is completely separate from the host. They do not share operating systems.  2-Only a limited number of VMs can be run at the same time.  3-Takes time to start up, so not suitable for microservices. | 1-Has access to Operating system and has root access. So, any vulnerability within the kernel may affect other containers too.  2-Less flexibility with OS. Containers with another operating system, require the compatible host too. | 1-Not as secure as containers  2-Shares operating system with the host, which may become a hazard for the entire system if the sandbox is at risk. |