A. Screenshots in task1

```
s108062213@cc-course-host:~/task1$ sudo docker images --filter "reference=108062213:2"
REPOSITORY    TAG       IMAGE ID       CREATED          SIZE
108062213     2         dedf07092189   22 minutes ago   917kB
```

```
s108062213@cc-course-host:~/task1$ sudo docker ps
CONTAINER ID   IMAGE          COMMAND      CREATED         STATUS         PORTS                                                  NAMES
45c8edf481b2   108062213:2    "./server"   4 seconds ago   Up 4 seconds   0.0.0.0:30057->8080/tcp, :::30057->8080/tcp   c1
```

```
s108062213@cc-course-host:~/task1$ ./client
Hello message sent
Hello from server
```

B. Screenshots in task3

```
s108062213@cc-course-host:~/task1$ kubectl get all
NAME                                     READY   STATUS    RESTARTS      AGE
pod/socket-deployment-6fcc677f79-68ndd   1/1     Running   0             106m
pod/socket-deployment-6fcc677f79-tdmtr   1/1     Running   0             106m
pod/socket-deployment-6fcc677f79-vnv72   1/1     Running   1 (24s ago)   106m
pod/socket-deployment-6fcc677f79-wxvp9   1/1     Running   1 (34s ago)   106m
pod/socket-deployment-6fcc677f79-wz4jp   1/1     Running   0             106m

NAME                     TYPE        CLUSTER-IP     EXTERNAL-IP   PORT(S)          AGE
service/kubernetes       ClusterIP   10.96.0.1      <none>        443/TCP          4h26m
service/socket-service   NodePort    10.96.112.38   <none>        8080:30057/TCP   105m

NAME                                READY   UP-TO-DATE   AVAILABLE   AGE
deployment.apps/socket-deployment   5/5     5            5           106m

NAME                                           DESIRED   CURRENT   READY   AGE
replicaset.apps/socket-deployment-6fcc677f79   5         5         5       106m
s108062213@cc-course-host:~/task1$ ./client
Hello message sent
Hello from server
```

C. Performance in task2 between VM and Container

Container CPU

```
Total operations: 66361936 (6635623.01 per second)

64806.58 MiB transferred (6480.10 MiB/sec)


General statistics:
    total time:                          10.0001s
    total number of events:              66361936

Latency (ms):
         min:                                    0.00
         avg:                                    0.00
         max:                                    0.50
         95th percentile:                        0.00
         sum:                                 3915.10

Threads fairness:
    events (avg/stddev):           66361936.0000/0.00
    execution time (avg/stddev):   3.9151/0.00
```

Container Memory

```
CPU speed:
    events per second:  4385.84

General statistics:
    total time:                          10.0003s
    total number of events:              43863

Latency (ms):
         min:                                    0.23
         avg:                                    0.23
         max:                                    1.23
         95th percentile:                        0.23
         sum:                                 9993.96

Threads fairness:
    events (avg/stddev):           43863.0000/0.00
    execution time (avg/stddev):   9.9940/0.00
```

Container FileIO (filesize=1G, testmode=rndrw)

```
File operations:
    reads/s:                      3661.95
    writes/s:                     2441.30
    fsyncs/s:                     7815.86

Throughput:
    read, MiB/s:                  57.22
    written, MiB/s:               38.15

General statistics:
    total time:                          10.0101s
    total number of events:              139217

Latency (ms):
         min:                                    0.00
         avg:                                    0.07
         max:                                   12.92
         95th percentile:                        0.34
         sum:                                 9965.33

Threads fairness:
    events (avg/stddev):           139217.0000/0.00
    execution time (avg/stddev):   9.9653/0.00
```

VM CPU

```
Prime numbers limit: 10000
Initializing worker threads...

Threads started!

CPU speed:
    events per second:  4340.65

General statistics:
    total time:                          10.0001s
    total number of events:              43410

Latency (ms):
         min:                                    0.22
         avg:                                    0.23
         max:                                    4.04
         95th percentile:                        0.23
         sum:                                 9993.80

Threads fairness:
    events (avg/stddev):           43410.0000/0.00
    execution time (avg/stddev):   9.9938/0.00
```

VM Memory

```
Total operations: 65705488 (6570023.23 per second)

64165.52 MiB transferred (6416.04 MiB/sec)

General statistics:
    total time:                          10.0000s
    total number of events:              65705488

Latency (ms):
         min:                                    0.00
         avg:                                    0.00
         max:                                    1.08
         95th percentile:                        0.00
         sum:                                 3917.98

Threads fairness:
    events (avg/stddev):           65705488.0000/0.00
    execution time (avg/stddev):   3.9180/0.00
```

VM FileIO (filesize=1G, testmode=rndrw)

```
File operations:
    reads/s:                      3524.41
    writes/s:                     2349.60
    fsyncs/s:                     7525.43

Throughput:
    read, MiB/s:                  55.07
    written, MiB/s:               36.71

General statistics:
    total time:                          10.0094s
    total number of events:              134003

Latency (ms):
         min:                                    0.00
         avg:                                    0.07
         max:                                   12.18
         95th percentile:                        0.34
         sum:                                 9959.44

Threads fairness:
    events (avg/stddev):           134003.0000/0.00
    execution time (avg/stddev):   9.9594/0.00
```

In all task, the performance of the container is better than VM(under 1CPU and 2G memory limit).

D. Difference between docker container and VM.

1. Docker container running share with host OS kernel. VM are made up of plus kernel space of an OS system.
2. Docker Container is occupy resource(i.e CPU, RAM) only using it. VM will occupy resource no matter it would be used.
3. Docker container use smaller image. VM's image usually be larger.

E. Explain "Deployment", "Service", and "Pod"

Pod: Pod is small deployable unit. It can wrapped one or more containers inside a single pod. All containers inside the same pod would share same resource.
Deployment: Deployment manages Pod's life cycle. Anything which goes to Pods goes through Deployment. When we use deployment to run apps, it creates and destroys Pods dynamically.
Service: If we want to have connectivity with Pods, we need to use service.

F. What is kubernetes? Why we need it?

kubernetes(k8s) is an open source system help us manage microservices and automatically deploy and manage containers on several pods. The reason we use k8s is everything on k8s are container that means we can create, destroy, and deploy them easily.

G. Why container technology is widely used in cloud computing environment?

Containerized applications are easier to migrate to the cloud. Containers also make it easier to leverage the extensive automation capabilities of the cloud. They can easily be deployed, cloned or modified using APIs provided by the container engine.

H. Why will the data disappear when container exits? How to keep them?

Create data in a container not record in the image, so if we exit the container and recreate one, the data would disappear.
There are multiple way to solve this problem and I find a way on the internet. That is using bind mount that the host mounted the file inside the container.

I. Why k8s use pod to manipulate the application instead of the container?

Running pods instead of containers to ensure containers within them can share same resource and local network, which let containers communicate between each other as they share physical hardware but isolated to some degree.

J. Explain the difference between k8s and docker-swarm? Why the k8s is more popular than the tools like docker-swarm?

Docker Swarm emphasizes ease of use, making it most suitable for simple applications that are quick to deploy and easy to manage.

k8s focuses on open-source and modular orchestration, offering an efficient container orchestration solution for high-demand applications with complex configuration.

Although k8s is more complex, it provides built-in monitor, self-healing, fault-tolerant, automatic scaling, multiple security protocols. I think it the main reason why k8s is more popular.

K. Docker is unsafe than virtual machine, please explain why and what is the vulnerability.

1. Kernel exploit: The kernel are sharing with host and all containers. Any one container cause kernel panic will shut down the whole host. In VMs, an attacker need to attack VM kernel and the hypervisor before touch the kernel of host.
2. DoS attack: All container share same resource. If one container monopolize access to certain resource, it would result into DoS.
3. Container breakout: User should only get certain containers access right. They shouldn't access host or other containers. The root(owner) need worry existing bugs let users adapt their access authorization.
4. Poison image: Using poison image cause the host and data in containers in danger. However, we are difficult checking the image have been tampered, be safe, and come from where they claim come from.
5. Compromising secret: Attackers get secret(i.e private key, password) can access the service.

L. Explain the difference between windows container and Hyper-V.

Windows server containers share the underlying OS kernel, which makes them smaller than VMs. The problem is security should be concerned. Hyper-V containers and their dependencies reside in Hyper-V VMs and provide an additional layer of isolation. Containers are typically used for microservices and stateless applications because they are deposable by design and, as such, don't store persistent data.