1.

This week, I learned different types of storage being used in Kubernetes and how to work / interact with them through Lab Manual. For instance, I learned how create additional initContainer along side with the main container that used to house the nodeJS application as well as mounting those two containers to a same volume where they can access the common file like datestamp.txt. In addition, I learned to use PV, PVC & hostPath to create a Pod. I also learned about configMap but still not yet fully undestand its functionality so I will spend more time to work on it.

These are all the important things that I learned from this week lecture.

2.

Task 1 – Remembering Kubernetes

```
haydenyeung@HaydenYeung-virtualbox:~/my-container$ docker images
REPOSITORY                  TAG       IMAGE ID       CREATED          SIZE
localhost:5000/node-date    latest    3b214b27b30a   31 seconds ago   936MB
registry                    2         26b2eb03618e   18 months ago    25.4MB
haydenyeung@HaydenYeung-virtualbox:~/my-container$ docker push localhost:5000/no
de-date
Using default tag: latest
The push refers to repository [localhost:5000/node-date]
a03beb1490b5: Mounted from node-web
f92723793659: Mounted from node-web
f0d8cfcdba81: Mounted from node-web
4a06816805a3: Mounted from node-web
b257e69d416f: Mounted from node-web
1e9c28d06610: Mounted from node-web
cddb98d77163: Mounted from node-web
ed0a3d9cbcc7: Mounted from node-web
8c8e652ecd8f: Mounted from node-web
2f4ee6a2e1b5: Mounted from node-web
latest: digest: sha256:260a035a109e9f37213b489ee6d7e30c52f7ab6c9064587d7025b8ffb
784f088 size: 2422
```

Created an image named localhost:5000/node-date and pushed it to the local repository of Docker.

```
haydenyeung@HaydenYeung-virtualbox: ~/my-container
 GNU nano 7.2                                myapp.yaml *
apiVersion: v1
kind: Pod
metadata:
  name: node-date
spec:
  containers:
  - name: my-custom-pod-51p
    image: localhost:5000/node-date
    ports:
    - containerPort: 8080
```

Wrote 'myapp.yaml' to deploy a custom pod using the pushed image above.

```
haydenyeung@HaydenYeung-virtualbox:~/my-container$ nano myapp.yaml
haydenyeung@HaydenYeung-virtualbox:~/my-container$ kubectl apply -f myapp.yaml
pod/node-date created
haydenyeung@HaydenYeung-virtualbox:~/my-container$ kubectl get pods
NAME                                READY   STATUS    RESTARTS   AGE
my-website-t41p-857867cbc5-qxrh5    1/1     Running   0          18m
node-date                           1/1     Running   0          7s
```

Pod was successfully created from 'myapp.yaml' file

```
haydenyeung@HaydenYeung-virtualbox: ~/node-date    haydenyeung@HaydenYeung-virtualbox: ~/node-date
haydenyeung@HaydenYeung-virtualbox:~/node-date$ curl localhost:8080
, I was created on Mon Apr 14 00:29:33 UTC 2025

haydenyeung@HaydenYeung-virtualbox:~/node-date$
```

Result obtained from "Automation with an emptyDir and an init Container"- because the instruction on creating myapp.js in Lab Week 5 was 'res.write(', I was created on '+data);' so this is expected and different from the result shown in the Lab manual.

```
haydenyeung@HaydenYeung-virtualbox:~/node-date$ cd ~/
haydenyeung@HaydenYeung-virtualbox:~$ mkdir ~/node-data
haydenyeung@HaydenYeung-virtualbox:~$ nano node-data-pv.yaml
haydenyeung@HaydenYeung-virtualbox:~$ nano node-data-pvc.yaml
haydenyeung@HaydenYeung-virtualbox:~$ kubectl apply -f node-data-pv.yaml
persistentvolume/node-data-pv created
haydenyeung@HaydenYeung-virtualbox:~$ kubectl apply -f node-data-pvc.yaml
persistentvolumeclaim/node-data-pvc created
haydenyeung@HaydenYeung-virtualbox:~$ kubectl get pvc
NAME            STATUS   VOLUME         CAPACITY   ACCESS MODES   STORAGECLASS   VOLUMEATTRIBUTESCLASS
 AGE
node-data-pvc   Bound    node-data-pv   1Gi        ROX                           <unset>
 67s
```

Successfully created both Persistent Volume & Persistent Volume Claim based on their respective .yaml file & checked with 'kubectl get pvc' command.

```
haydenyeung@HaydenYeung-virtualbox:~$ nano node-date-hostpath.yaml
haydenyeung@HaydenYeung-virtualbox:~$ kubectl apply -f node-date-hostpath.yaml
pod/node-data-hostpath created
haydenyeung@HaydenYeung-virtualbox:~$ kubectl port-forward node-date-auto 8080
Error from server (NotFound): pods "node-date-auto" not found
haydenyeung@HaydenYeung-virtualbox:~$ kubectl get pods
NAME                 READY   STATUS    RESTARTS   AGE
node-data-hostpath   1/1     Running   0          71s
haydenyeung@HaydenYeung-virtualbox:~$ kubectl port-forward node-data-hostpath
error: TYPE/NAME and list of ports are required for port-forward
See 'kubectl port-forward -h' for help and examples
haydenyeung@HaydenYeung-virtualbox:~$ kubectl port-forward node-data-hostpath 8080
Forwarding from 127.0.0.1:8080 -> 8080
Forwarding from [::1]:8080 -> 8080
Handling connection for 8080
```

```
haydenyeung@HaydenYeung-virtualbox:~$ curl localhost:8080

haydenyeung@HaydenYeung-virtualbox:~$ date > node-data/datestamp.txt
haydenyeung@HaydenYeung-virtualbox:~$ curl localhost:8080
, I was created on Mon Apr 14 11:08:32 AM AEST 2025

haydenyeung@HaydenYeung-virtualbox:~$
```

This was expected as datestamp.txt was not yet created in “Decoupled storage” activities. Only when this files was added that resulted in the display in the “time-of-creation” prompt appear.

```
haydenyeung@HaydenYeung-virtualbox: ~
bash-5.2# ls /realRoot/home/haydenyeung
Bookstore                Templates                node-data-pvc.yaml
Desktop                  Videos                   node-date
Documents                assessing-mininet        node-date-hostpath.yaml
Downloads                dangerHostPath.yaml      packages-microsoft-prod.deb
LOIC                     jperf.zip                snap
Music                    my-container             www
Pictures                 node-data
Public                   node-data-pv.yaml
```

The created pod in activity “The Danger of hostPath” can really access host /home folder because no “denied access” was observed

```
root:*:19837:0:99999:7:::
daemon:*:19837:0:99999:7:::
bin:*:19837:0:99999:7:::
sys:*:19837:0:99999:7:::
sync:*:19837:0:99999:7:::
games:*:19837:0:99999:7:::
man:*:19837:0:99999:7:::
lp:*:19837:0:99999:7:::
mail:*:19837:0:99999:7:::
news:*:19837:0:99999:7:::
uucp:*:19837:0:99999:7:::
proxy:*:19837:0:99999:7:::
www-data:*:19837:0:99999:7:::
backup:*:19837:0:99999:7:::
list:*:19837:0:99999:7:::
irc:*:19837:0:99999:7:::
_apt:*:19837:0:99999:7:::
nobody:*:19837:0:99999:7:::
systemd-network:!*:19837::::::
systemd-timesync:!*:19837::::::
dhcpcd:!:19837::::::
messagebus:!:19837::::::
syslog:!:19837::::::
systemd-resolve:!*:19837::::::
/realRoot/etc/shadow
```

/shadow was also be able to be accessed.

```
haydenyeung@HaydenYeung-virtualbox:~/node-data$ kubectl create configmap node-co
nfigmap --from-file datestamp.txt
configmap/node-configmap created
haydenyeung@HaydenYeung-virtualbox:~/node-data$ kubectl get configmap node-confi
gmap -o yaml
apiVersion: v1
data:
  datestamp.txt: |
    Mon Apr 14 11:08:32 AM AEST 2025
kind: ConfigMap
metadata:
  creationTimestamp: "2025-04-14T01:21:27Z"
  name: node-configmap
  namespace: default
  resourceVersion: "276778"
  uid: de92197a-4087-422a-bb0a-1dc3de843a65
```

Created Configmap was inspected by ‘kubectl get configmap ….’

```
haydenyeung@HaydenYeung-virtualbox:~/node-data$ kubectl apply -f node-date-confi
gmap.yaml
pod/node-date-configmap created
haydenyeung@HaydenYeung-virtualbox:~/node-data$ kubectl port-forward node-date-c
onfigmap 8080
error: unable to forward port because pod is not running. Current status=Pending
haydenyeung@HaydenYeung-virtualbox:~/node-data$ kubectl get pods
NAME                  READY   STATUS    RESTARTS   AGE
bashroot              1/1     Running   0          15m
node-date-configmap   1/1     Running   0          34s
haydenyeung@HaydenYeung-virtualbox:~/node-data$ kubectl port-forward node-date-c
onfigmap 8080
Forwarding from 127.0.0.1:8080 -> 8080
Forwarding from [::1]:8080 -> 8080
```

Successfully created “node-date-configmap” from its respective .yaml file

```
haydenyeung@HaydenYeung-virtualbox: ~/node-data
haydenyeung@HaydenYeung-virtualbox: ~/no...   haydenyeung@HaydenYeung-virtualbox: ~/no...
haydenyeung@HaydenYeung-virtualbox:~/node-data$ curl localhost:8080
, I was created on Mon Apr 14 11:08:32 AM AEST 2025
```

Successfully generated result with datetime value after using ‘curl localhost:8080’

Challenge Task – Change the ConfigMap

1/ Proceed with command: kubectl edit configmap <configmap-name>, which is node-configmap in this case.

2/ Change the value stored in datestamp.txt to ‘Hello Kubernetes ConfigMap Update’

```
# Please edit the object below. Lines beginning with a '#' will be ignored,
# and an empty file will abort the edit. If an error occurs while saving this fi
le will be
# reopened with the relevant failures.
#
apiVersion: v1
data:
  datestamp.txt: |
   Hello Kubernetes ConfigMap Update
kind: ConfigMap
metadata:
  creationTimestamp: "2025-04-14T01:21:27Z"
  name: node-configmap
  namespace: default
  resourceVersion: "276778"
  uid: de92197a-4087-422a-bb0a-1dc3de843a65
~
```

3/ Updated the current configmap node-configmap with new value from datestamp.txt

```
haydenyeung@HaydenYeung-virtualbox:~/node-data$ kubectl create configmap node-co
nfigmap --from-file datestamp.txt
error: failed to create configmap: configmaps "node-configmap" already exists
haydenyeung@HaydenYeung-virtualbox:~/node-data$ kubectl create configmap node-co
nfigmap --from-file datestamp.txt --dry-run=client -o yaml | kubectl apply -f -
Warning: resource configmaps/node-configmap is missing the kubectl.kubernetes.io
/last-applied-configuration annotation which is required by kubectl apply. kubec
tl apply should only be used on resources created declaratively by either kubect
l create --save-config or kubectl apply. The missing annotation will be patched
automatically.
configmap/node-configmap configured
```

Had to used to below command, suggested from Gemini 2.5 Pro, to ‘updated’ the current node-configMap

4/ Check again with ‘curl localhost:8080’

```
haydenyeung@HaydenYeung-virtualbox: ~/no...   haydenyeung@HaydenYeung-virtualbox: ~/no...
haydenyeung@HaydenYeung-virtualbox:~/node-data$ curl localhost:8080
, I was created on Hello Kubernetes ConfigMap!

haydenyeung@HaydenYeung-virtualbox:~/node-data$
```

5/ I decided to retry again: because the first time I did exit the “port-forward” and this time I did the changing value while still “port-forward” and it shown as below:

```
haydenyeung@HaydenYeung-virtualbox: ~/no...   haydenyeung@HaydenYeung-virtualbox: ~/no...
haydenyeung@HaydenYeung-virtualbox:~/node-data$ curl localhost:8080
, I was created on Hello Kubernetes ConfigMap!

haydenyeung@HaydenYeung-virtualbox:~/node-data$ kubectl edit configmap node-conf
igmap
configmap/node-configmap edited
haydenyeung@HaydenYeung-virtualbox:~/node-data$ curl localhost:8080
, I was created on Hello Kubernetes ConfigMap!

haydenyeung@HaydenYeung-virtualbox:~/node-data$ curl localhost:8080
, I was created on Hello Kubernetes ConfigMap! It is me, Mario!!!
```

It had to took a bit of time to reflected the change.

3.

An example of a Kubernetes storage class providing temporary local storage is **emptyDir**. This storage class creates an empty volume that exists only for the lifecycle of a pod, making it ideal for ephemeral data (Kubernetes, 2025). Key features include its simplicity, as it requires no external provisioning, and its tight integration with the pod's lifecycle—data is automatically deleted when the pod terminates. However, it lacks persistence, meaning data is lost if the pod restarts or is rescheduled. Applications like caching layers or temporary scratch space in web servers often use emptyDir, as they require fast, local storage for short-lived data during processing, such as intermediate computation results in machine learning workloads (Kubernetes, 2025).

4.

A Kubernetes storage class that provides block storage is **aws-ebs**, which leverages Amazon Elastic Block Store (EBS) for persistent, high-performance storage (Kubernetes, 2025). Features of aws-ebs include low-latency access, support for dynamic provisioning, and the ability to attach volumes to a single pod, ensuring dedicated storage with consistent performance. It also offers snapshot capabilities for backups and encryption for security. Block storage like aws-ebs is well-suited for applications requiring high I/O performance, such as databases (e.g., MySQL or PostgreSQL), where direct, raw access to storage is critical for handling transactional workloads and ensuring data durability (Amazon Web Services, 2025).

References

Amazon Web Services. (2025). *Amazon Elastic Block Store (EBS)*. https://aws.amazon.com/ebs/

Kubernetes. (2025). *Storage classes*. https://kubernetes.io/docs/concepts/storage/storage-classes/