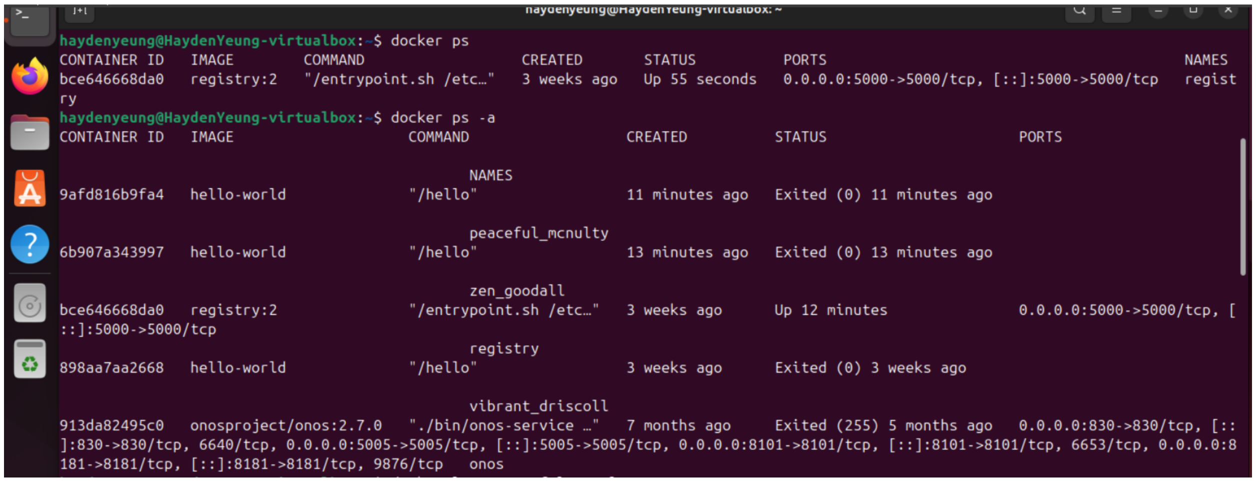
1.

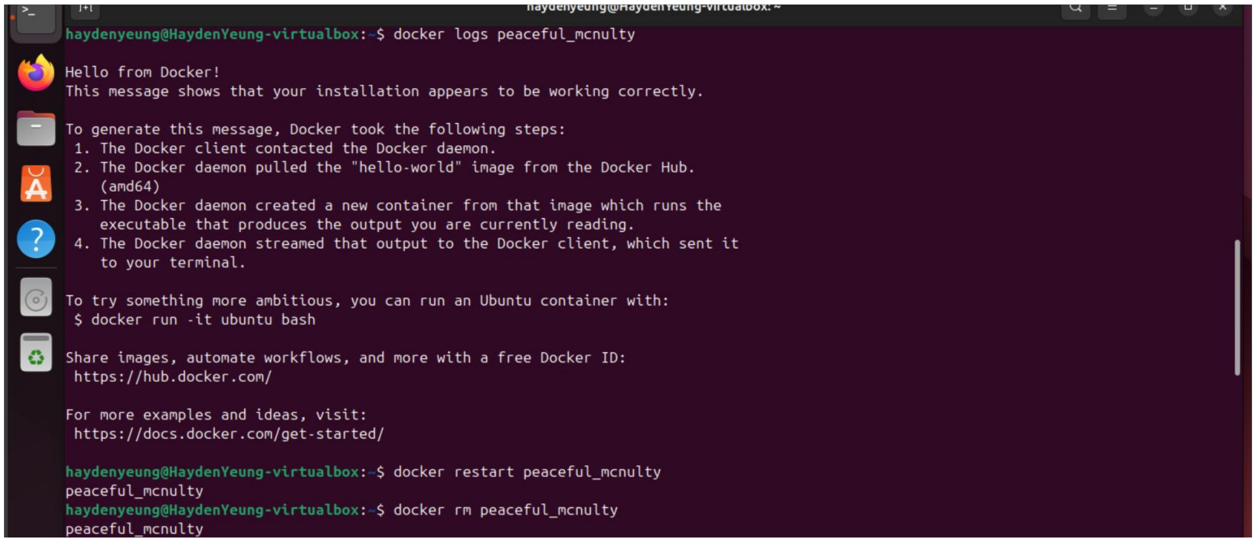
For this week, I learned the concept of monolithic, microservices, and how to work with Docker – these are equally important to me at the moment since I currently taking another cloud-related unit as knowledge gained from both help strengthen my foundation upon cloud technology much better through constant exposure.
By having a fresh understanding of both monolithic and microservices, I could start reading "Designing Data-Intensive Applications" book and grasp it at a faster rate than I used to be before. As for Docker, I only know it briefly not but detail like its mechanisms, how does it contribute to the overall process of an application / software, etc ... (or a.k.a why it is widely used globally).

2.

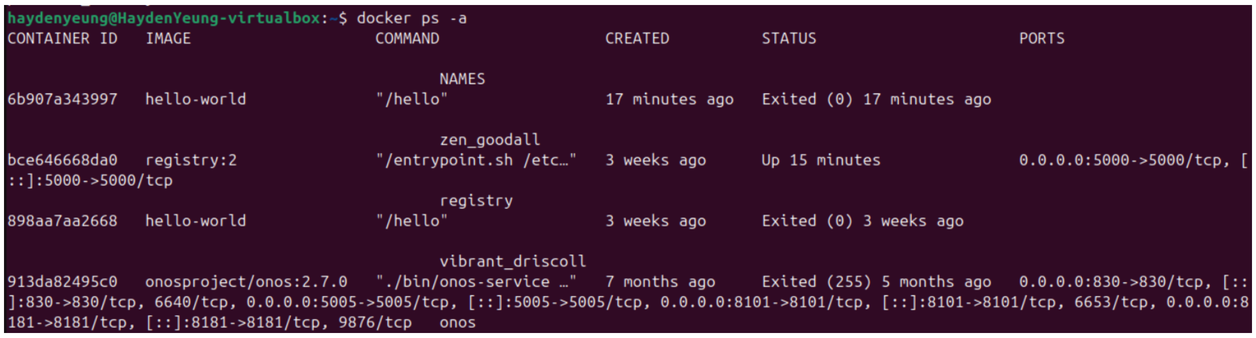
Task 1 – Learn by doing



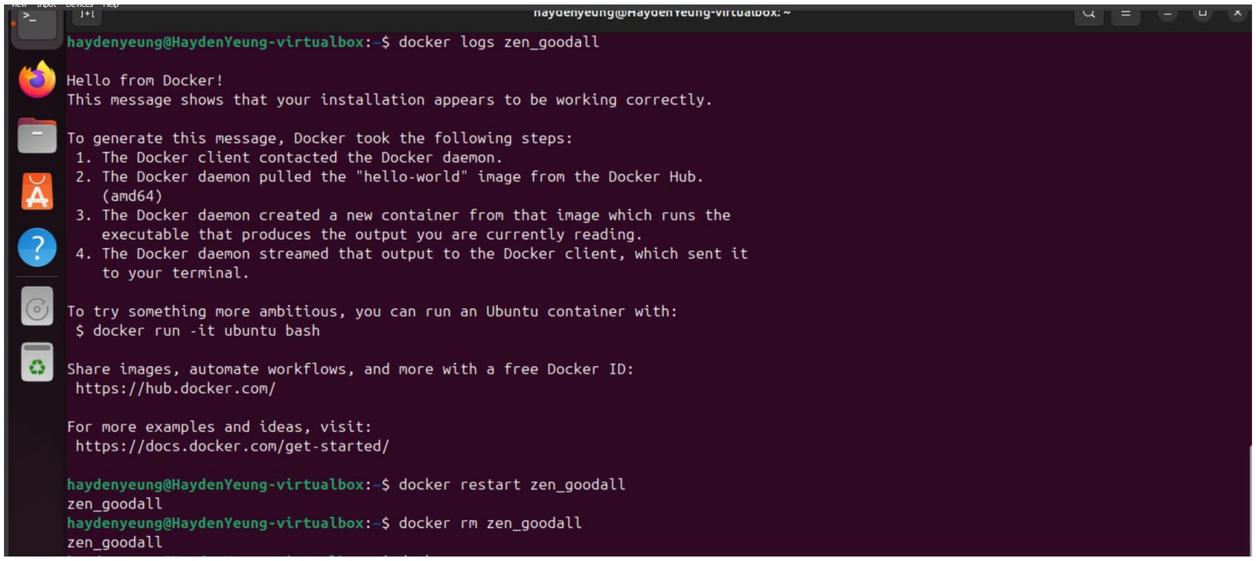
```
haydenyeung@HaydenYeung-virtualbox:~$ docker ps
CONTAINER ID   IMAGE        COMMAND                  CREATED       STATUS          PORTS                                       NAMES
bce646668da0   registry:2   "/entrypoint.sh /etc…"   3 weeks ago   Up 55 seconds   0.0.0.0:5000->5000/tcp, [::]:5000->5000/tcp   regist
ry
haydenyeung@HaydenYeung-virtualbox:~$ docker ps -a
CONTAINER ID   IMAGE                     COMMAND                  CREATED          STATUS                     PORTS

                                                   NAMES
9afd816b9fa4   hello-world               "/hello"                 11 minutes ago   Exited (0) 11 minutes ago

                                                   peaceful_mcnulty
6b907a343997   hello-world               "/hello"                 13 minutes ago   Exited (0) 13 minutes ago

                                                   zen_goodall
bce646668da0   registry:2                "/entrypoint.sh /etc…"   3 weeks ago      Up 12 minutes              0.0.0.0:5000->5000/tcp, [
::]:5000->5000/tcp
                                                   registry
898aa7aa2668   hello-world               "/hello"                 3 weeks ago      Exited (0) 3 weeks ago

                                                   vibrant_driscoll
913da82495c0   onosproject/onos:2.7.0    "./bin/onos-service …"   7 months ago     Exited (255) 5 months ago   0.0.0.0:830->830/tcp, [::
]:830->830/tcp, 6640/tcp, 0.0.0.0:5005->5005/tcp, [::]:5005->5005/tcp, 0.0.0.0:8101->8101/tcp, [::]:8101->8101/tcp, 6653/tcp, 0.0.0.0:8
181->8181/tcp, [::]:8181->8181/tcp, 9876/tcp   onos
```



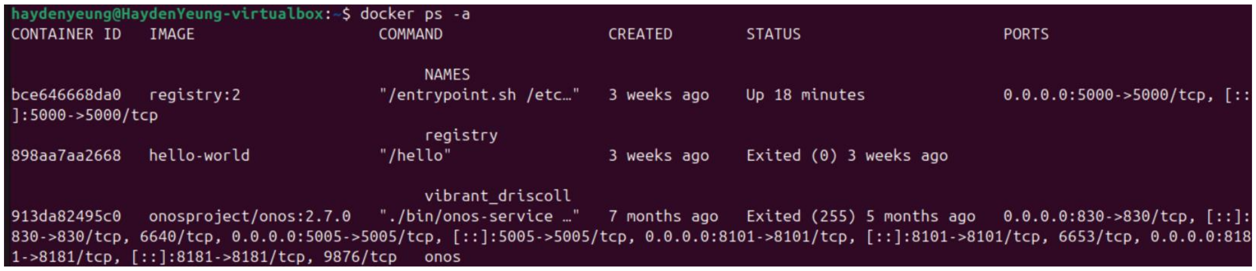
```
haydenyeung@HaydenYeung-virtualbox:~$ docker logs peaceful_mcnulty

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/

For more examples and ideas, visit:
 https://docs.docker.com/get-started/

haydenyeung@HaydenYeung-virtualbox:~$ docker restart peaceful_mcnulty
peaceful_mcnulty
haydenyeung@HaydenYeung-virtualbox:~$ docker rm peaceful_mcnulty
peaceful_mcnulty
```



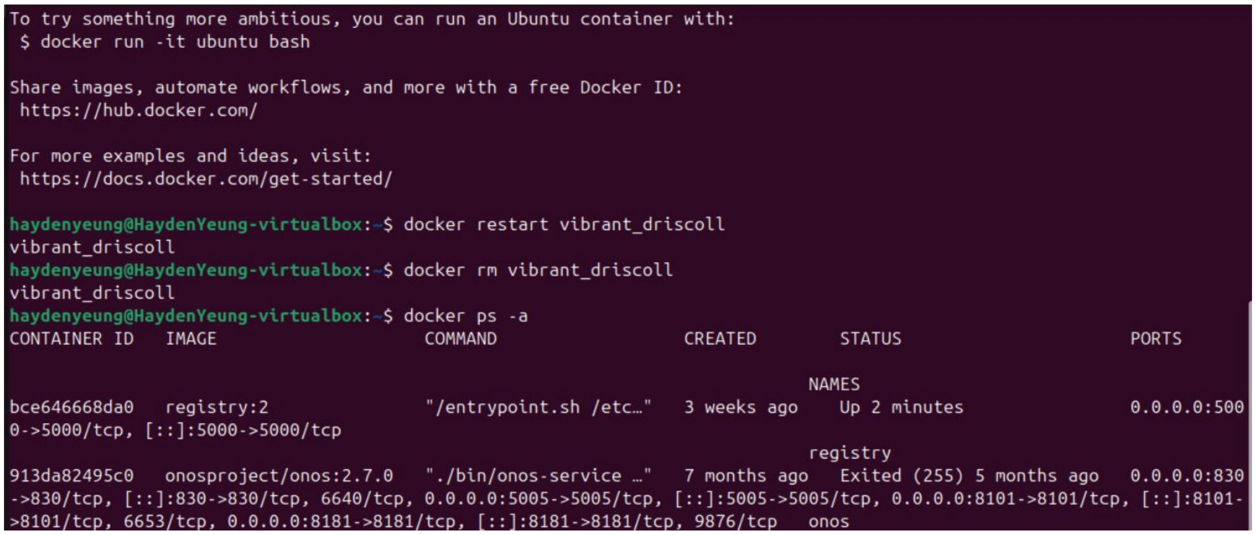
```
haydenyeung@HaydenYeung-virtualbox:~$ docker ps -a
CONTAINER ID   IMAGE                     COMMAND                  CREATED          STATUS                        PORTS
                                                 NAMES
6b907a343997   hello-world               "/hello"                 17 minutes ago   Exited (0) 17 minutes ago
                                                 zen_goodall
bce646668da0   registry:2                "/entrypoint.sh /etc…"   3 weeks ago      Up 15 minutes                 0.0.0.0:5000->5000/tcp, [
::]:5000->5000/tcp
                                                 registry
898aa7aa2668   hello-world               "/hello"                 3 weeks ago      Exited (0) 3 weeks ago
                                                 vibrant_driscoll
913da82495c0   onosproject/onos:2.7.0    "./bin/onos-service …"   7 months ago     Exited (255) 5 months ago   0.0.0.0:830->830/tcp, [::
]:830->830/tcp, 6640/tcp, 0.0.0.0:5005->5005/tcp, [::]:5005->5005/tcp, 0.0.0.0:8101->8101/tcp, [::]:8101->8101/tcp, 6653/tcp, 0.0.0.0:8
181->8181/tcp, [::]:8181->8181/tcp, 9876/tcp    onos
```

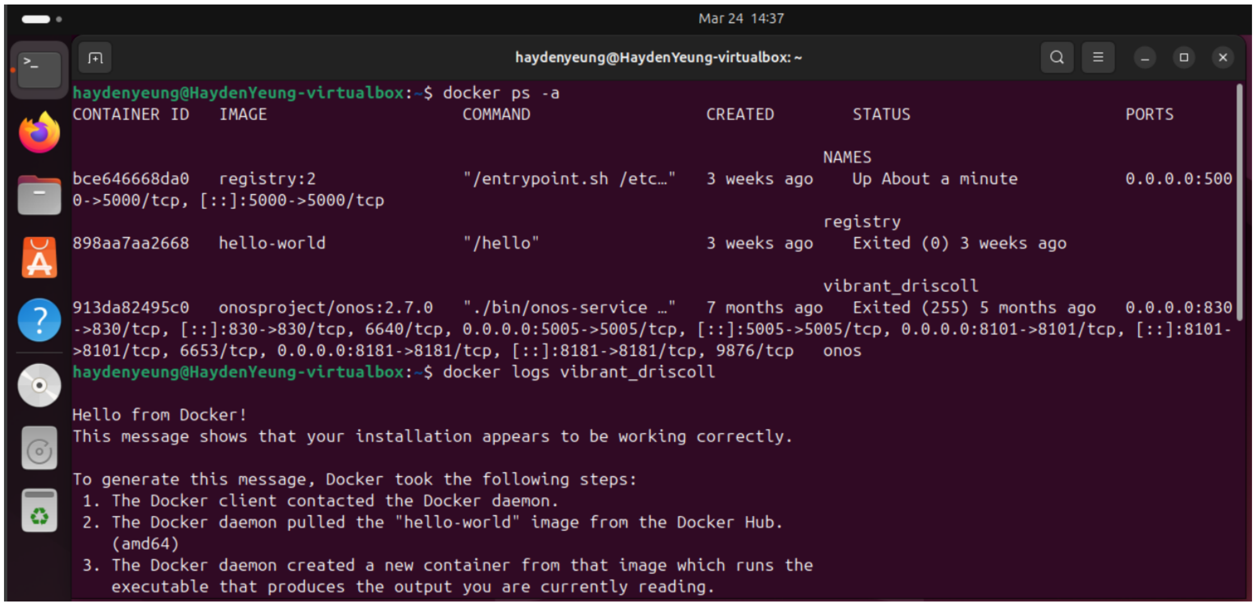


```
haydenyeung@HaydenYeung-virtualbox:~$ docker logs zen_goodall

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/

For more examples and ideas, visit:
 https://docs.docker.com/get-started/

haydenyeung@HaydenYeung-virtualbox:~$ docker restart zen_goodall
zen_goodall
haydenyeung@HaydenYeung-virtualbox:~$ docker rm zen_goodall
zen_goodall
```



```
haydenyeung@HaydenYeung-virtualbox:~$ docker ps -a
CONTAINER ID   IMAGE                     COMMAND                  CREATED          STATUS                        PORTS
                                                 NAMES
bce646668da0   registry:2                "/entrypoint.sh /etc…"   3 weeks ago      Up 18 minutes                 0.0.0.0:5000->5000/tcp, [::
]:5000->5000/tcp
                                                 registry
898aa7aa2668   hello-world               "/hello"                 3 weeks ago      Exited (0) 3 weeks ago
                                                 vibrant_driscoll
913da82495c0   onosproject/onos:2.7.0    "./bin/onos-service …"   7 months ago     Exited (255) 5 months ago   0.0.0.0:830->830/tcp, [::]:
830->830/tcp, 6640/tcp, 0.0.0.0:5005->5005/tcp, [::]:5005->5005/tcp, 0.0.0.0:8101->8101/tcp, [::]:8101->8101/tcp, 6653/tcp, 0.0.0.0:818
1->8181/tcp, [::]:8181->8181/tcp, 9876/tcp    onos
```


```
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/

For more examples and ideas, visit:
 https://docs.docker.com/get-started/

haydenyeung@HaydenYeung-virtualbox:~$ docker restart vibrant_driscoll
vibrant_driscoll
haydenyeung@HaydenYeung-virtualbox:~$ docker rm vibrant_driscoll
vibrant_driscoll
haydenyeung@HaydenYeung-virtualbox:~$ docker ps -a
CONTAINER ID   IMAGE                     COMMAND                  CREATED         STATUS                          PORTS

                                                                             NAMES
bce646668da0   registry:2               "/entrypoint.sh /etc…"   3 weeks ago     Up 2 minutes                    0.0.0.0:500
0->5000/tcp, [::]:5000->5000/tcp
                                                                             registry
913da82495c0   onosproject/onos:2.7.0   "./bin/onos-service …"   7 months ago    Exited (255) 5 months ago       0.0.0.0:830
->830/tcp, [::]:830->830/tcp, 6640/tcp, 0.0.0.0:5005->5005/tcp, [::]:5005->5005/tcp, 0.0.0.0:8101->8101/tcp, [::]:8101-
>8101/tcp, 6653/tcp, 0.0.0.0:8181->8181/tcp, [::]:8181->8181/tcp, 9876/tcp   onos
```

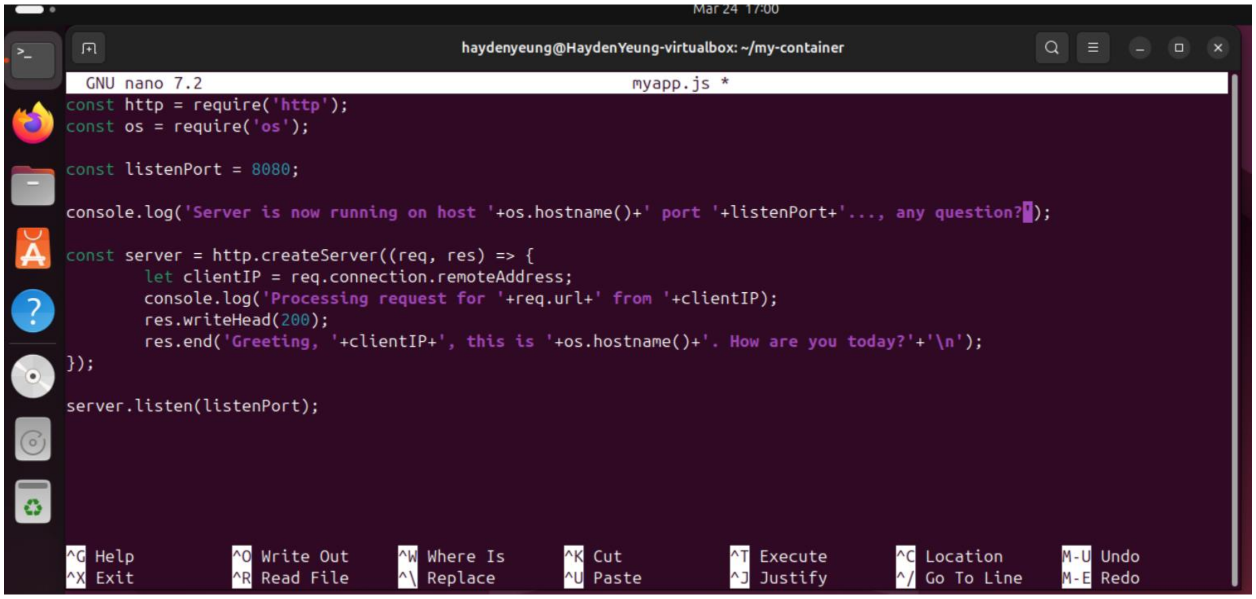

Mar 24 14:37

haydenyeung@HaydenYeung-virtualbox: ~

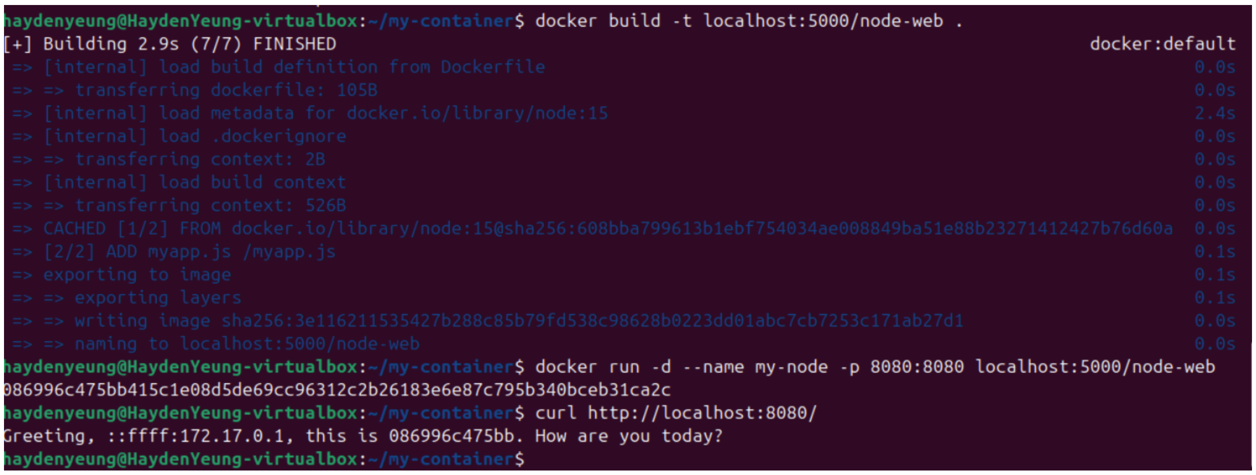
```
haydenyeung@HaydenYeung-virtualbox:~$ docker ps -a
CONTAINER ID   IMAGE                     COMMAND                  CREATED         STATUS                          PORTS

                                                                             NAMES
bce646668da0   registry:2               "/entrypoint.sh /etc…"   3 weeks ago     Up About a minute               0.0.0.0:500
0->5000/tcp, [::]:5000->5000/tcp
                                                                             registry
898aa7aa2668   hello-world              "/hello"                 3 weeks ago     Exited (0) 3 weeks ago

                                                                             vibrant_driscoll
913da82495c0   onosproject/onos:2.7.0   "./bin/onos-service …"   7 months ago    Exited (255) 5 months ago       0.0.0.0:830
->830/tcp, [::]:830->830/tcp, 6640/tcp, 0.0.0.0:5005->5005/tcp, [::]:5005->5005/tcp, 0.0.0.0:8101->8101/tcp, [::]:8101-
>8101/tcp, 6653/tcp, 0.0.0.0:8181->8181/tcp, [::]:8181->8181/tcp, 9876/tcp   onos
haydenyeung@HaydenYeung-virtualbox:~$ docker logs vibrant_driscoll

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.
```

I found that I have 3 images of hello-world (peaceful_mcnulty, zen_goodall, vibrant_driscoll) so I wen ahead logs, restart then remove them from the list displayed from "docker ps -a".

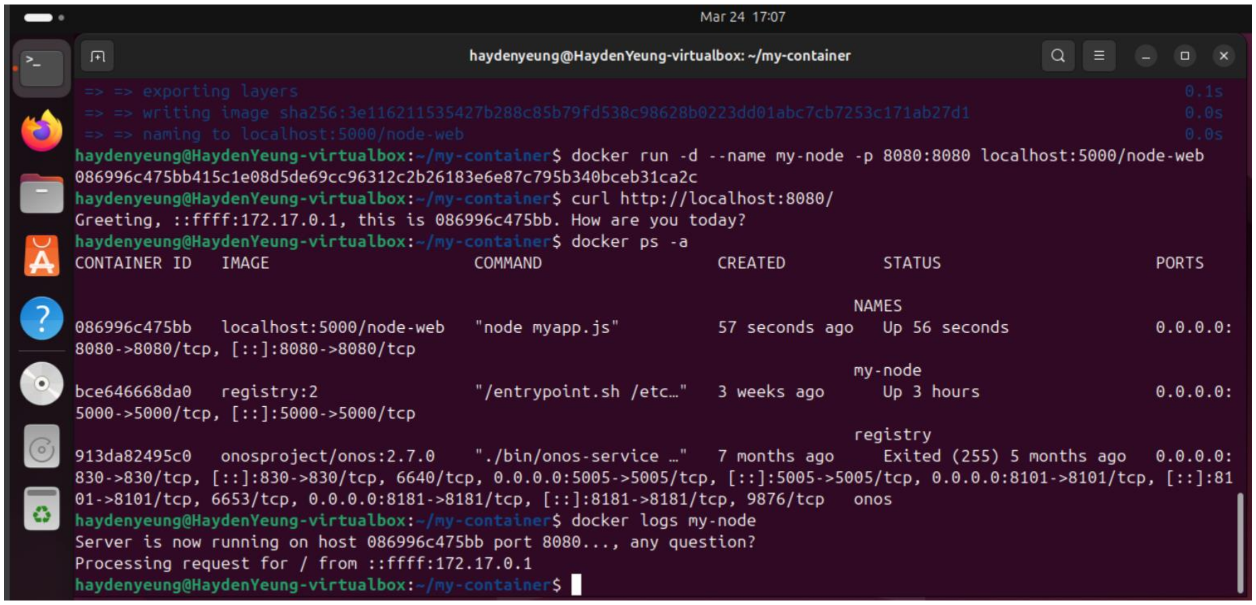
Task 2 - Update the container image



```
haydenyeung@HaydenYeung-virtualbox: ~/my-container
  GNU nano 7.2                                        myapp.js *
const http = require('http');
const os = require('os');

const listenPort = 8080;

console.log('Server is now running on host '+os.hostname()+' port '+listenPort+'..., any question?');

const server = http.createServer((req, res) => {
        let clientIP = req.connection.remoteAddress;
        console.log('Processing request for '+req.url+' from '+clientIP);
        res.writeHead(200);
        res.end('Greeting, '+clientIP+', this is '+os.hostname()+'. How are you today?'+'\n');
});

server.listen(listenPort);

^G Help        ^O Write Out   ^W Where Is    ^K Cut         ^T Execute     ^C Location    M-U Undo
^X Exit        ^R Read File   ^\ Replace     ^U Paste       ^J Justify     ^/ Go To Line  M-E Redo
```



```
haydenyeung@HaydenYeung-virtualbox:~/my-container$ docker build -t localhost:5000/node-web .
[+] Building 2.9s (7/7) FINISHED                                                          docker:default
 => [internal] load build definition from Dockerfile                                                0.0s
 => => transferring dockerfile: 105B                                                                0.0s
 => [internal] load metadata for docker.io/library/node:15                                          2.4s
 => [internal] load .dockerignore                                                                   0.0s
 => => transferring context: 2B                                                                     0.0s
 => [internal] load build context                                                                   0.0s
 => => transferring context: 526B                                                                   0.0s
 => CACHED [1/2] FROM docker.io/library/node:15@sha256:608bba799613b1ebf754034ae008849ba51e88b23271412427b76d60a  0.0s
 => [2/2] ADD myapp.js /myapp.js                                                                    0.1s
 => exporting to image                                                                              0.1s
 => => exporting layers                                                                             0.1s
 => => writing image sha256:3e116211535427b288c85b79fd538c98628b0223dd01abc7cb7253c171ab27d1        0.0s
 => => naming to localhost:5000/node-web                                                            0.0s
haydenyeung@HaydenYeung-virtualbox:~/my-container$ docker run -d --name my-node -p 8080:8080 localhost:5000/node-web
086996c475bb415c1e08d5de69cc96312c2b26183e6e87c795b340bceb31ca2c
haydenyeung@HaydenYeung-virtualbox:~/my-container$ curl http://localhost:8080/
Greeting, ::ffff:172.17.0.1, this is 086996c475bb. How are you today?
haydenyeung@HaydenYeung-virtualbox:~/my-container$
```



```
 => => exporting layers                                                                                  0.1s
 => => writing image sha256:3e116211535427b288c85b79fd538c98628b0223dd01abc7cb7253c171ab27d1              0.0s
 => => naming to localhost:5000/node-web                                                                 0.0s
haydenyeung@HaydenYeung-virtualbox:~/my-container$ docker run -d --name my-node -p 8080:8080 localhost:5000/node-web
086996c475bb415c1e08d5de69cc96312c2b26183e6e87c795b340bceb31ca2c
haydenyeung@HaydenYeung-virtualbox:~/my-container$ curl http://localhost:8080/
Greeting, ::ffff:172.17.0.1, this is 086996c475bb. How are you today?
haydenyeung@HaydenYeung-virtualbox:~/my-container$ docker ps -a
CONTAINER ID   IMAGE                     COMMAND                  CREATED          STATUS                      PORTS

                                                                                  NAMES
086996c475bb   localhost:5000/node-web   "node myapp.js"          57 seconds ago   Up 56 seconds               0.0.0.0:
8080->8080/tcp, [::]:8080->8080/tcp
                                                                                  my-node
bce646668da0   registry:2                "/entrypoint.sh /etc…"   3 weeks ago      Up 3 hours                  0.0.0.0:
5000->5000/tcp, [::]:5000->5000/tcp
                                                                                  registry
913da82495c0   onosproject/onos:2.7.0    "./bin/onos-service …"   7 months ago     Exited (255) 5 months ago   0.0.0.0:
830->830/tcp, [::]:830->830/tcp, 6640/tcp, 0.0.0.0:5005->5005/tcp, [::]:5005->5005/tcp, 0.0.0.0:8101->8101/tcp, [::]:81
01->8101/tcp, 6653/tcp, 0.0.0.0:8181->8181/tcp, [::]:8181->8181/tcp, 9876/tcp   onos
haydenyeung@HaydenYeung-virtualbox:~/my-container$ docker logs my-node
Server is now running on host 086996c475bb port 8080..., any question?
Processing request for / from ::ffff:172.17.0.1
haydenyeung@HaydenYeung-virtualbox:~/my-container$
```

I simply change the text display of console.log() and res.end. Since I still fresh with backend (node.js) so I would stop here.

3.

Capital Expenditure (CapEx) and Operational Expenditure (OpEx) are two primary categories of business expenses, each with distinct financial implications. CapEx refers to the funds a company uses to acquire, upgrade, or maintain physical assets such as property, industrial buildings, or equipment. These are substantial, upfront investments that provide value over time and are typically capitalized on the balance sheet and depreciated over the asset's useful life. For example, purchasing servers for an on-premises data center is considered a CapEx investment (Watts, 2025; Wikipedia, n.d.).

OpEx, on the other hand, encompasses the ongoing costs associated with the daily operations of a business. These expenses are fully deducted in the accounting period they are incurred. Examples include salaries, utilities, and rent. In the context of IT infrastructure, subscribing to cloud services falls under OpEx, as businesses pay regular fees for the services they consume without significant upfront investments (GeeksforGeeks, 2023; Watts, 2025).

When comparing locally hosted infrastructure to cloud computing, the distinction between CapEx and OpEx becomes particularly significant. Deploying and maintaining on-premises infrastructure requires substantial CapEx, as organizations must invest in hardware, software licenses, and facilities. Additionally, they incur ongoing OpEx for utilities, staffing, and maintenance. This model can tie up capital and may lead to underutilized resources if capacity planning is not optimal (Watts, 2025).

Conversely, cloud computing operates on a pay-as-you-go model, aligning with OpEx. Companies can scale resources up or down based on demand, paying only for what they use. This approach offers financial flexibility, reduces the need for large upfront investments, and allows organizations to respond swiftly to changing business needs. However, it is essential to monitor usage to prevent unexpected costs (Lovett, 2023).

In summary, CapEx involves significant upfront investments in physical assets, typical of locally hosted infrastructures, while OpEx pertains to ongoing operational costs, characteristic of cloud computing services. The choice between these models depends on an organization's financial strategy, scalability requirements, and capacity to manage IT resources.

References

GeeksforGeeks. (2023). *CapEx vs OpEx in cloud computing*. Retrieved March 22, 2025, from https://www.geeksforgeeks.org/capex-vs-opex-in-cloud-computing/

Watts, S. (2025, January 29). *CapEx vs. OpEx for cloud, IT spending, and business operations*. Retrieved March 21, 2025, from https://www.splunk.com/en_us/blog/learn/capex-vs-opex.html

Lovett, C. (2023, January 18) *CapEx vs OpEx: IT spending and the cloud*. Retrieved March 22, 2025, from https://www.tierpoint.com/blog/capex-vs-opex-cloud-whats-the-difference/

Wikipedia. (n.d.). *Capital expenditure*. Retrieved March 22, 2025, from https://en.wikipedia.org/wiki/Capital_expenditure

4.

Many organizations prefer Operational Expenditure (OpEx) over Capital Expenditure (CapEx) because it offers financial flexibility and reduces the burden of large upfront costs. Instead of making a significant investment in hardware and infrastructure, businesses can opt for a pay-as-you-go model, which allows them to scale resources as needed and keep cash flow more predictable (BCS365, 2022). This approach is particularly beneficial for companies looking to remain agile in a fast-changing market, as it frees up capital for innovation and other strategic initiatives (Ewoldt B., 2024).

However, shifting from locally hosted infrastructure to cloud services introduces new risks. One key challenge is the loss of direct control over IT assets and security, as cloud providers manage most of the infrastructure (Flexential, 2023). This can make it harder for businesses to monitor performance and ensure compliance with internal security policies. Additionally, cloud migration may expose organizations to data breaches or service disruptions if not handled properly, making risk assessment and strategic planning essential (Morrow, T. 2018).

References

BCS365. (2022). *Benefits of shifting your CapEx to OpEx with cloud computing*. Retrieved March 21, 2025, from https://bcs365.com/insights/benefits-of-shifting-your-capex-to-opex-with-cloud-computing

Flexential. (2023, June 27). *Cloud migration risks & mitigation strategies*. Retrieved March 21, 2025, from https://www.flexential.com/resources/blog/how-mitigate-risk-cloud-migration-strategies

Ewoldt, E. (2024, February 13). *Cloud migration: How innovative companies are moving to the cloud*. Retrieved March 21, 2025, from https://www.ntiva.com/blog/cloud-migration-why-companies-are-moving-to-the-cloud

Morrow, T. (2018, March 5). *12 Risks, Threats, & Vulnerabilities in Moving to the Cloud*. Retrieved March 21, 2025, from https://insights.sei.cmu.edu/blog/12-risks-threats-vulnerabilities-in-moving-to-the-cloud/.