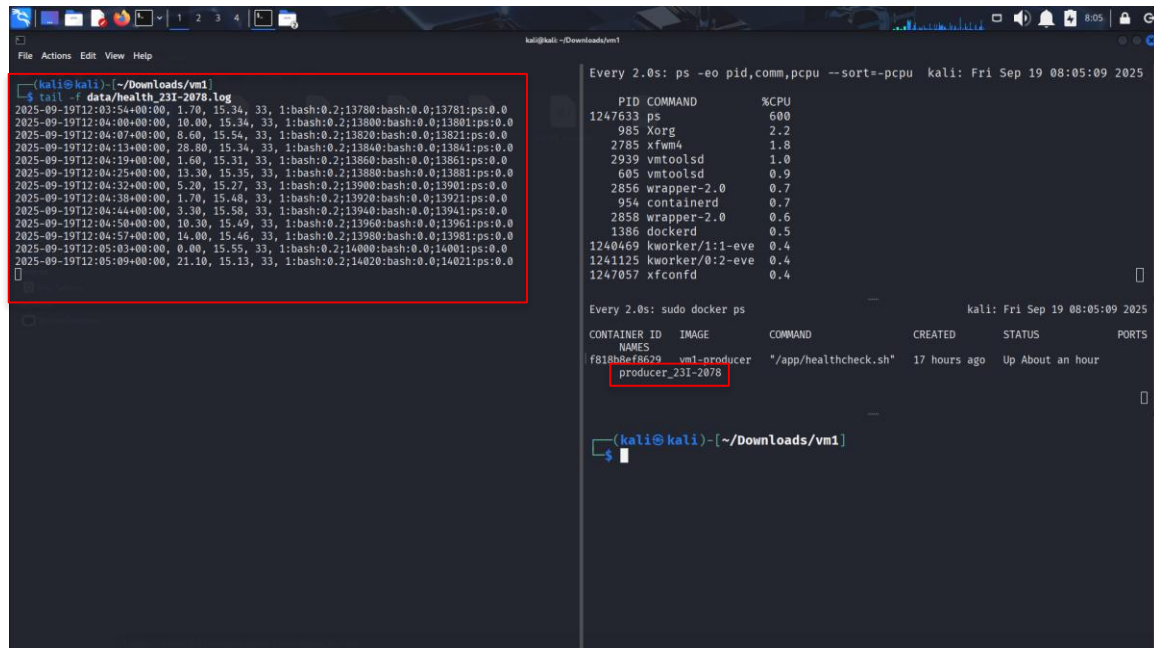


Name: Habib Ahmed

Roll No: 23I-2078

Section: A

## 1) System Logs:



The screenshot shows a Kali Linux terminal window with two panes. The left pane displays the output of the command `tail -f data/health_23I-2078.log`, showing a series of log entries with timestamps and system metrics. The right pane shows the output of the command `ps -eo pid,comm,pcpu --sort=-pcpu`, displaying a list of running processes sorted by CPU usage. Below this, the output of `sudo docker ps` is shown, listing the Docker containers. The container `producer_23I-2078` is highlighted with a red box.

```
(kali@kali) - [~/Downloads/vm1]
$ tail -f data/health_23I-2078.log
2025-09-19T12:03:54+00:00, 1.70, 15.34, 33, 1:bash:0.2;13780:bash:0.0;13781:ps:0.0
2025-09-19T12:04:00+00:00, 10.00, 15.34, 33, 1:bash:0.2;13800:bash:0.0;13801:ps:0.0
2025-09-19T12:04:07+00:00, 8.60, 15.54, 33, 1:bash:0.2;13820:bash:0.0;13821:ps:0.0
2025-09-19T12:04:13+00:00, 28.80, 15.34, 33, 1:bash:0.2;13840:bash:0.0;13841:ps:0.0
2025-09-19T12:04:19+00:00, 1.60, 15.31, 33, 1:bash:0.2;13860:bash:0.0;13861:ps:0.0
2025-09-19T12:04:25+00:00, 13.30, 15.35, 33, 1:bash:0.2;13880:bash:0.0;13881:ps:0.0
2025-09-19T12:04:32+00:00, 5.20, 15.27, 33, 1:bash:0.2;13900:bash:0.0;13901:ps:0.0
2025-09-19T12:04:38+00:00, 1.70, 15.48, 33, 1:bash:0.2;13920:bash:0.0;13921:ps:0.0
2025-09-19T12:04:44+00:00, 3.30, 15.58, 33, 1:bash:0.2;13940:bash:0.0;13941:ps:0.0
2025-09-19T12:04:50+00:00, 10.30, 15.49, 33, 1:bash:0.2;13960:bash:0.0;13961:ps:0.0
2025-09-19T12:04:57+00:00, 14.00, 15.46, 33, 1:bash:0.2;13980:bash:0.0;13981:ps:0.0
2025-09-19T12:05:03+00:00, 0.00, 15.55, 33, 1:bash:0.2;14000:bash:0.0;14001:ps:0.0
2025-09-19T12:05:09+00:00, 21.10, 15.13, 33, 1:bash:0.2;14020:bash:0.0;14021:ps:0.0

Every 2.0s: ps -eo pid,comm,pcpu --sort=-pcpu kali: Fri Sep 19 08:05:09 2025
PID COMMAND %CPU
1247633 ps 600
985 Xorg 2.2
2785 xfwm4 1.8
2939 vmtotlsd 1.0
605 vmtotlsd 0.9
2856 wrapper-2.0 0.7
954 containerd 0.7
2858 wrapper-2.0 0.6
1386 dockerd 0.5
1240469 kworker/1:1-eve 0.4
1241125 kworker/0:2-eve 0.4
1247057 xfconfd 0.4

Every 2.0s: sudo docker ps kali: Fri Sep 19 08:05:09 2025
CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS
f8188eef8629 vm1-producer "/app/healthcheck.sh" 17 hours ago Up About an hour
producer_23I-2078
```

1\_health.png

- From “vm1/data/health\_23I-2078.log”:

```
2025-09-19T12:27:56+00:00, 16.90, 15.56, 33,
1:bash:0.2;18380:bash:0.0;18381:ps:0.0
2025-09-19T12:28:02+00:00, 19.60, 15.58, 33,
1:bash:0.2;18400:bash:0.0;18401:ps:0.0
2025-09-19T12:28:08+00:00, 11.90, 15.63, 33,
1:bash:0.2;18420:bash:0.0;18421:ps:0.0
```

The screenshot shows a Kali Linux terminal window with the title bar 'kali@kali: ~/Downloads/vm2'. The terminal is divided into two panes. The left pane shows the output of the command `tail -f data/received_231-2078.log`, displaying a series of log entries with timestamps and IP addresses. The right pane shows the output of the command `sudo docker exec consumer_231-2078 ...`, displaying system metrics and a table of Docker containers. The container table has columns for CONTAINER ID, IMAGE, COMMAND, NAMES, CREATED, and STATUS. The container 'consumer\_231-2078' is highlighted with a red box.

```
kali@kali:~/Downloads/vm2
$ tail -f data/received_231-2078.log
2025-09-19T12:05:41+00:00, 16.40, 15.53, 33, 1:bash:0.2;14120:bash:0.0;14121:ps:0.0
2025-09-19T12:05:47+00:00, 5.20, 15.41, 33, 1:bash:0.2;14140:bash:0.0;14141:ps:0.0
2025-09-19T12:05:53+00:00, 10.30, 15.56, 33, 1:bash:0.2;14160:bash:0.0;14161:ps:0.0
2025-09-19T12:05:59+00:00, 1.70, 15.63, 33, 1:bash:0.2;14180:bash:0.0;14181:ps:0.0
2025-09-19T12:06:06+00:00, 8.60, 15.60, 33, 1:bash:0.2;14200:bash:0.0;14201:ps:0.0
2025-09-19T12:06:12+00:00, 1.70, 15.77, 33, 1:bash:0.2;14220:bash:0.0;14221:ps:0.0
2025-09-19T12:06:18+00:00, 1.70, 16.02, 33, 1:bash:0.2;14240:bash:0.0;14241:ps:0.0
2025-09-19T12:06:24+00:00, 1.60, 15.64, 33, 1:bash:0.2;14260:bash:0.0;14261:ps:0.0
2025-09-19T12:06:31+00:00, 48.20, 15.94, 33, 1:bash:0.2;14280:bash:0.0;14281:ps:0.0
2025-09-19T12:06:37+00:00, 0.00, 15.82, 33, 1:bash:0.2;14300:bash:0.0;14301:ps:0.0
2025-09-19T12:06:43+00:00, 1.60, 15.74, 33, 1:bash:0.2;14320:bash:0.0;14321:ps:0.0

Every 2.0s: sudo docker exec consumer_231-2078 ... kali: Fri Sep 19 08:06:45 2025
avg_cpu=8.69 avg_mem=15.40 max_disk=33.00 ALERT

Every 2.0s: sudo docker ps kali: Fri Sep 19 08:06:45 2025
CONTAINER ID   IMAGE          COMMAND                  CREATED        STATUS
PORTS
856b743cce45   vm2-consumer  "/app/receiver.sh"      18 hours ago  Up About an hou
r   0.0.0.0:5000->5000/tcp, :::5000->5000/tcp  consumer_231-2078

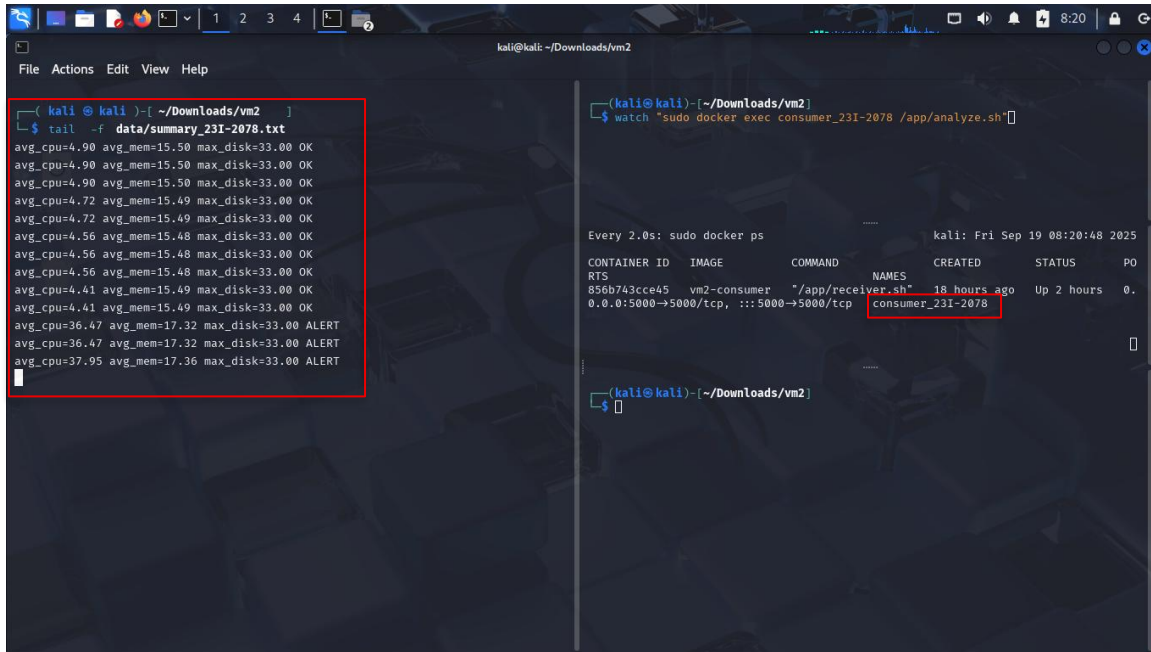
(kali@kali)~[~/Downloads/vm2]
```

1\_received.png

- From “vm2/data/received\_231-2078.log”:

```
2025-09-19T12:28:27+00:00, 8.30, 15.59, 33,
1:bash:0.2;18480:bash:0.0;18481:ps:0.0
2025-09-19T12:28:33+00:00, 1.60, 15.58, 33,
1:bash:0.2;18500:bash:0.0;18501:ps:0.0
2025-09-19T12:28:40+00:00, 1.70, 15.62, 33,
1:bash:0.2;18520:bash:0.0;18521:ps:0.0
```

## 2) Analysis Output:



```
(kali@kali)~[~/Downloads/vm2]
$ tail -f data/summary_231-2078.txt
avg_cpu=4.90 avg_mem=15.50 max_disk=33.00 OK
avg_cpu=4.90 avg_mem=15.50 max_disk=33.00 OK
avg_cpu=4.90 avg_mem=15.50 max_disk=33.00 OK
avg_cpu=4.72 avg_mem=15.49 max_disk=33.00 OK
avg_cpu=4.72 avg_mem=15.49 max_disk=33.00 OK
avg_cpu=4.56 avg_mem=15.48 max_disk=33.00 OK
avg_cpu=4.56 avg_mem=15.48 max_disk=33.00 OK
avg_cpu=4.56 avg_mem=15.48 max_disk=33.00 OK
avg_cpu=4.41 avg_mem=15.49 max_disk=33.00 OK
avg_cpu=4.41 avg_mem=15.49 max_disk=33.00 OK
avg_cpu=36.47 avg_mem=17.32 max_disk=33.00 ALERT
avg_cpu=36.47 avg_mem=17.32 max_disk=33.00 ALERT
avg_cpu=37.95 avg_mem=17.36 max_disk=33.00 ALERT

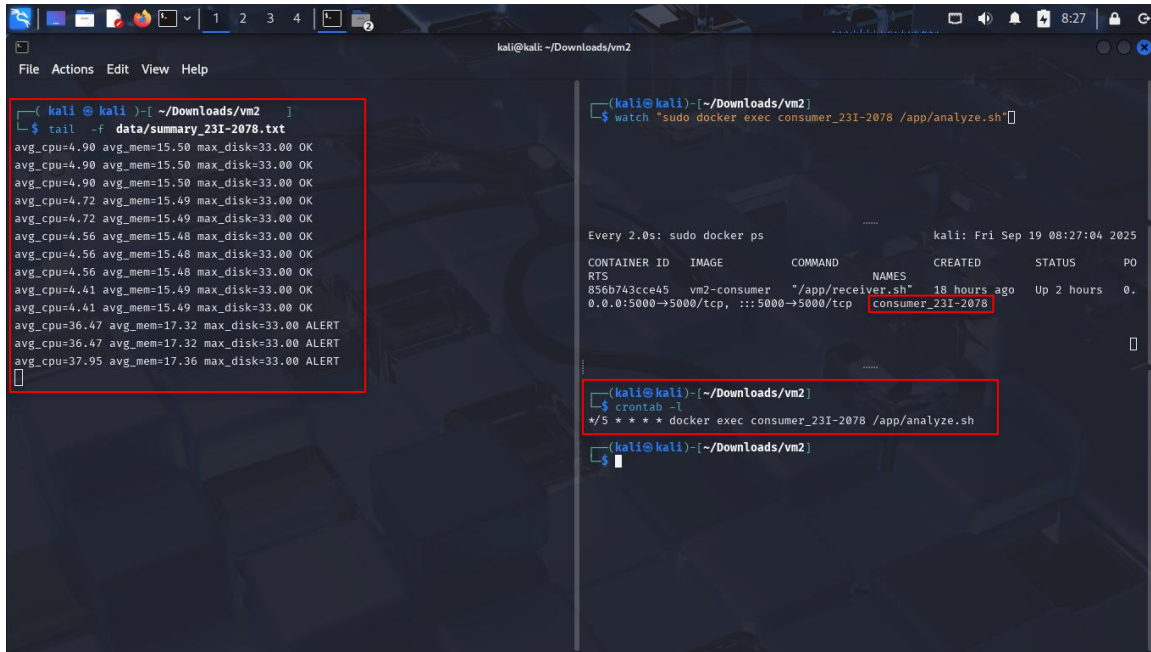
(kali@kali)~[~/Downloads/vm2]
$ watch 'sudo docker exec consumer_231-2078 /app/analyze.sh'

Every 2.0s: sudo docker ps                                kali: Fri Sep 19 08:20:48 2025
CONTAINER ID   IMAGE          COMMAND                  NAMES                  CREATED          STATUS          PO
RTS
856b743cce45   vm2-consumer  "/app/receiver.sh"      consumer_231-2078     18 hours ago    Up 2 hours      0.
0.0:5000->5000/tcp, :::5000->5000/tcp
```

2\_summary.png

```
avg_cpu=4.41 avg_mem=15.49 max_disk=33.00 OK
avg_cpu=36.47 avg_mem=17.32 max_disk=33.00 ALERT
avg_cpu=36.47 avg_mem=17.32 max_disk=33.00 ALERT
```

### 3) Process Scheduling Evidence



```
(kali@kali)~[~/Downloads/vm2]
$ tail -f data/summary_231-2078.txt
avg_cpu=4.90 avg_mem=15.50 max_disk=33.00 OK
avg_cpu=4.90 avg_mem=15.50 max_disk=33.00 OK
avg_cpu=4.72 avg_mem=15.49 max_disk=33.00 OK
avg_cpu=4.72 avg_mem=15.49 max_disk=33.00 OK
avg_cpu=4.56 avg_mem=15.48 max_disk=33.00 OK
avg_cpu=4.56 avg_mem=15.48 max_disk=33.00 OK
avg_cpu=4.56 avg_mem=15.48 max_disk=33.00 OK
avg_cpu=4.41 avg_mem=15.49 max_disk=33.00 OK
avg_cpu=4.41 avg_mem=15.49 max_disk=33.00 OK
avg_cpu=36.47 avg_mem=17.32 max_disk=33.00 ALERT
avg_cpu=36.47 avg_mem=17.32 max_disk=33.00 ALERT
avg_cpu=37.95 avg_mem=17.36 max_disk=33.00 ALERT

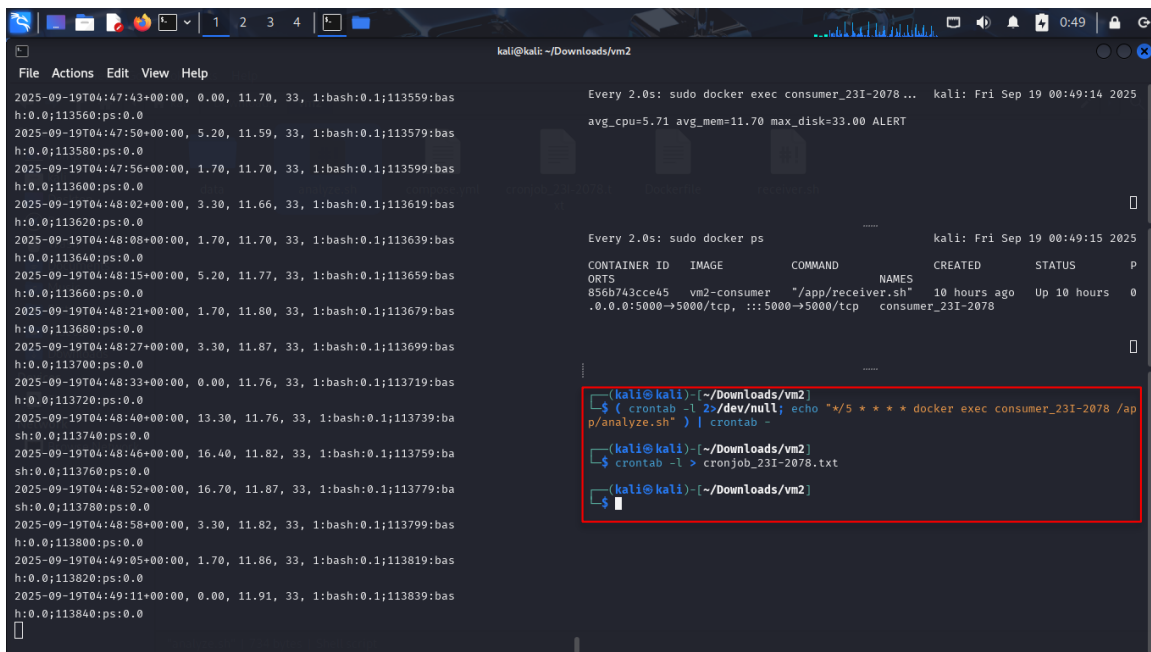
(kali@kali)~[~/Downloads/vm2]
$ watch 'sudo docker exec consumer_231-2078 /app/analyze.sh'

Every 2.0s: sudo docker ps                                kali: Fri Sep 19 08:27:04 2025
CONTAINER ID   IMAGE          COMMAND                  NAMES              CREATED          STATUS          PO
RTS
856b743cce45   vm2-consumer  "/app/receiver.sh"      consumer_231-2078  18 hours ago    Up 2 hours      0.
0.0.0:5000->5000/tcp, :::5000->5000/tcp

(kali@kali)~[~/Downloads/vm2]
$ crontab -l
*/5 * * * * docker exec consumer_231-2078 /app/analyze.sh

(kali@kali)~[~/Downloads/vm2]
$
```

3\_crontab.png



```
(kali@kali)~[~/Downloads/vm2]
$ tail -f data/summary_231-2078.txt
2025-09-19T04:47:43+00:00, 0.00, 11.70, 33, 1:bash:0.1;113559:bas
h:0.0;113560:ps:0.0
2025-09-19T04:47:50+00:00, 5.20, 11.59, 33, 1:bash:0.1;113579:bas
h:0.0;113580:ps:0.0
2025-09-19T04:47:56+00:00, 1.70, 11.70, 33, 1:bash:0.1;113599:bas
h:0.0;113600:ps:0.0
2025-09-19T04:48:02+00:00, 3.30, 11.66, 33, 1:bash:0.1;113619:bas
h:0.0;113620:ps:0.0
2025-09-19T04:48:08+00:00, 1.70, 11.70, 33, 1:bash:0.1;113639:bas
h:0.0;113640:ps:0.0
2025-09-19T04:48:15+00:00, 5.20, 11.77, 33, 1:bash:0.1;113659:bas
h:0.0;113660:ps:0.0
2025-09-19T04:48:21+00:00, 1.70, 11.80, 33, 1:bash:0.1;113679:bas
h:0.0;113680:ps:0.0
2025-09-19T04:48:27+00:00, 3.30, 11.87, 33, 1:bash:0.1;113699:bas
h:0.0;113700:ps:0.0
2025-09-19T04:48:33+00:00, 0.00, 11.76, 33, 1:bash:0.1;113719:bas
h:0.0;113720:ps:0.0
2025-09-19T04:48:40+00:00, 13.30, 11.76, 33, 1:bash:0.1;113739:ba
sh:0.0;113740:ps:0.0
2025-09-19T04:48:46+00:00, 16.40, 11.82, 33, 1:bash:0.1;113759:ba
sh:0.0;113760:ps:0.0
2025-09-19T04:48:52+00:00, 16.70, 11.87, 33, 1:bash:0.1;113779:ba
sh:0.0;113780:ps:0.0
2025-09-19T04:48:58+00:00, 3.30, 11.82, 33, 1:bash:0.1;113799:bas
h:0.0;113800:ps:0.0
2025-09-19T04:49:05+00:00, 1.70, 11.86, 33, 1:bash:0.1;113819:bas
h:0.0;113820:ps:0.0
2025-09-19T04:49:11+00:00, 0.00, 11.91, 33, 1:bash:0.1;113839:bas
h:0.0;113840:ps:0.0

(kali@kali)~[~/Downloads/vm2]
$ watch 'sudo docker exec consumer_231-2078 /app/analyze.sh'

Every 2.0s: sudo docker exec consumer_231-2078 ...      kali: Fri Sep 19 00:49:14 2025
avg_cpu=5.71 avg_mem=11.70 max_disk=33.00 ALERT

(kali@kali)~[~/Downloads/vm2]
$ watch 'sudo docker ps'
Every 2.0s: sudo docker ps                                kali: Fri Sep 19 00:49:15 2025
CONTAINER ID   IMAGE          COMMAND                  NAMES              CREATED          STATUS          P
ORTS
856b743cce45   vm2-consumer  "/app/receiver.sh"      consumer_231-2078  10 hours ago    Up 10 hours      0
.0.0.0:5000->5000/tcp, :::5000->5000/tcp

(kali@kali)~[~/Downloads/vm2]
$ ( crontab -l 2>/dev/null; echo "*/5 * * * * docker exec consumer_231-2078 /ap
p/analyze.sh" ) | crontab -
(kali@kali)~[~/Downloads/vm2]
$ crontab -l > cronjob_231-2078.txt
(kali@kali)~[~/Downloads/vm2]
$
```

3\_crontab\_optional.png

`*/5 * * * * docker exec consumer_231-2078 /app/analyze.sh`

## 4) Problems Faced and Solutions

- **Problem 1:** Learning and using new Linux/Kali utilities (e.g., awk, ps, nc)
- **Fix/Workaround:** Explored different commands step by step, tested outputs, and wrote small scripts (CPU/memory/disk checkers, log monitors) to understand how each utility works in real scenarios. This hands-on approach made the tools easier to remember and apply.
  
- **Problem 2:** Understanding Dockerfile commands and docker-compose.yml configuration
- **Fix/Workaround:** Practiced building Docker images with common instructions (FROM, RUN, COPY, VOLUME, ENTRYPOINT) and experimented with docker compose syntax for container orchestration. Fixed issues by comparing old vs. new syntax and learning how volumes, and services are defined in YAML.

## Appendix

- VM1: Container name "producer\_23I-2078"
- VM2: Container name "consumer\_23I-2078"
- Listener port: "5000"