

Name: Habib Ahmed

Roll No: 23I-2078

Section: A

1) System Logs:

The screenshot shows a terminal window with two main sections of output. The top section is a log file named 'health_23I-2078.log' located at '/data/health_23I-2078.log'. It contains several lines of timestamped log entries, each consisting of a timestamp, three numerical values, and the string '33, 1:bash:0.2;13780: bash:0.0;13781:ps:0.0'. The bottom section shows the command 'ps -eo pid,comm,pcpu --sort=-pcpu' running every 2.0s, followed by the command 'sudo docker ps' running every 2.0s. The Docker output lists a container named 'producer_23I-2078' with ID 'f818bbef629', which has been running for 17 hours and is currently up.

```
(kali㉿kali)-[~/Downloads/vm1]
└─$ cat /data/health_23I-2078.log
2025-09-19T12:03:54+00:00, 16.70, 15.94, 33, 1:bash:0.2;13780: bash:0.0;13781:ps:0.0
2025-09-19T12:04:00+00:00, 10.00, 15.36, 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, 10.00, 15.49, 33, 1:bash:0.2;13960: bash:0.0;13961:ps:0.0
2025-09-19T12:04:56+00:00, 0.00, 15.57, 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, 15.15, 33, 1:bash:0.2;14020: bash:0.0;14021:ps:0.0
└─$ ps -eo pid,comm,pcpu --sort=-pcpu
kali: Fri Sep 19 08:05:09 2025
PID COMMAND %CPU
1247633 ps 6.00
985 Xorg 2.2
2785 xfwm4 1.8
2939 vntoolsd 1.0
605 vntoolsd 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
f818bbef629 _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 terminal window on a Kali Linux desktop. The terminal has two tabs open. The left tab displays the command `tail -f data/received_23I-2078.log`, which is monitoring a log file for sensor data. The right tab shows the command `sudo docker ps`, listing a running container named `consumer_23I-2078`.

```

kali㉿kali:[ ~/Downloads/vm2 ]
└$ tail -f data/received_23I-2078.log
2025-09-19T12:05:41+00:00, 16.40, 15.53, 33, 1:bash:0.2;14120:bas
h:0.0;14121:ps:0.0
2025-09-19T12:05:47+00:00, 5.20, 15.41, 33, 1:bash:0.2;14140:bas
h:0.0;14141:ps:0.0
2025-09-19T12:05:53+00:00, 10.30, 15.56, 33, 1:bash:0.2;14160:bas
h:0.0;14161:ps:0.0
2025-09-19T12:05:59+00:00, 1.70, 15.63, 33, 1:bash:0.2;14180:bas
h:0.0;14181:ps:0.0
2025-09-19T12:06:06+00:00, 8.60, 15.60, 33, 1:bash:0.2;14200:bas
h:0.0;14201:ps:0.0
2025-09-19T12:06:12+00:00, 1.70, 15.77, 33, 1:bash:0.2;14220:bas
h:0.0;14221:ps:0.0
2025-09-19T12:06:18+00:00, 1.70, 16.02, 33, 1:bash:0.2;14240:bas
h:0.0;14241:ps:0.0
2025-09-19T12:06:24+00:00, 1.60, 15.64, 33, 1:bash:0.2;14260:bas
h:0.0;14261:ps:0.0
2025-09-19T12:06:31+00:00, 48.20, 15.94, 33, 1:bash:0.2;14280:bas
h:0.0;14281:ps:0.0
2025-09-19T12:06:37+00:00, 0.00, 15.82, 33, 1:bash:0.2;14300:bas
h:0.0;14301:ps:0.0
2025-09-19T12:06:43+00:00, 1.60, 15.74, 33, 1:bash:0.2;14320:bas
h:0.0;14321:ps:0.0
└$ 

```

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS
856b743cc645	vm2-consumer	"/app/receiver.sh"	18 hours ago	Up About an hour
				consumer_23I-2078

1_received.png

- From “vm2/data/received_23I-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:

The screenshot shows a terminal window with two panes. The left pane displays the command `tail -f data/summary_23I-2078.txt` and its output, which includes several lines of performance metrics followed by three lines of alerts. The right pane shows the command `watch 'sudo docker exec consumer_23I-2078 /app/analyze.sh'` and its output, which includes a Docker ps command showing a running container named `consumer_23I-2078`.

```
( kali㉿kali )-[ ~/Downloads/vm2 ]$ tail -f data/summary_23I-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_23I-2078 /app/analyze.sh'
Every 2.0s: sudo docker ps
CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES
856b743cce45 vm2-consumer "/app/receiver.sh" 18 hours ago Up 2 hours 0.0.0:5000→5000/tcp, :::5000→5000/tcp consumer_23I-2078
```

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

A screenshot of a terminal window titled "kali@kali: ~/Downloads/vm2". The window contains several command-line sessions:

- The first session shows the output of the command `tail -f data/summary_23I-2078.txt`, which monitors system statistics (avg_cpu, avg_mem, max_disk) over time. A red box highlights the output of this command.
- The second session shows the output of `watch 'sudo docker exec consumer_23I-2078 /app/analyze.sh'`. It displays a table of Docker container statistics every 2.0s. A red box highlights the container ID, image, command, created time, status, and port mapping for the consumer_23I-2078 container.
- The third session shows the output of `crontab -l`, which lists a cron job: `* * * * * docker exec consumer_23I-2078 /app/analyze.sh`. A red box highlights this cron entry.
- The fourth session shows the command `ps`.

3_crontab.png

A screenshot of a terminal window titled "kali@kali: ~/Downloads/vm2". The window contains several command-line sessions:

- The first session shows the output of `tail -f data/summary_23I-2078.txt`, monitoring system statistics over time.
- The second session shows the output of `watch 'sudo docker exec consumer_23I-2078 /app/analyze.sh'`. It displays a table of Docker container statistics every 2.0s. A red box highlights the container ID, image, command, created time, status, and port mapping for the consumer_23I-2078 container.
- The third session shows the output of `ps`.
- The fourth session shows the command `crontab -l`, which lists a cron job: `* * * * * docker exec consumer_23I-2078 /app/analyze.sh`. A red box highlights this cron entry.
- The fifth session shows the command `crontab -l > cronjob_23I-2078.txt`.
- The sixth session shows the command `ps`.

3_crontab_optional.png

`*/5 * * * * docker exec consumer_23I-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”