WEEK-5 (Stress testing)

JMX exporter setup:

Step 1: create jmx directory

```
sudo mkdir -p /opt/jmx-exporter

sudo chown $USER:$USER /opt/jmx-exporter

cd /opt/jmx-exporter
```

Step 2: download jmx

```
wget
https://repo1.maven.org/maven2/io/prometheus/jmx/jmx_prometheus_javaag
ent/1.0.1/jmx_prometheus_javaagent-1.0.1.jar
```

Step 3: update kafka start command:

```
KAFKA_HEAP_OPTS="-Xmx512M -Xms512M" export KAFKA_OPTS="$KAFKA_OPTS -
javaagent:/opt/jmx-exporter/jmx_prometheus_javaagent-
1.0.1.jar=7071:/opt/jmx-exporter/kafka-jmx-config.yaml" bin/kafka-
server-start.sh config/server.properties
```

Core Kafka JMX Metrics (via JMX Exporter)

1. Log End Offset

Tracks the latest offset in a partition (i.e., how large each partition is getting):

```
kafka.log:type=Log,name=LogEndOffset
```

Why it's important:

More log segments = more disk usage. You can use this to estimate growth rate.

2. Messages In Per Second

Rate of incoming messages:

```
kafka.server:type=BrokerTopicMetrics,name=MessagesInPerSec
```

Why it's important:

High message throughput = faster disk consumption.

3. Bytes In/Out Per Second

Useful for understanding I/O and topic traffic:

```
kafka.server:type=BrokerTopicMetrics,name=BytesInPerSec

kafka.server:type=BrokerTopicMetrics,name=BytesOutPerSec
```

4. Under-Replicated Partitions

If disk fills up, Kafka might fail to replicate data:

```
kafka.server:type=ReplicaManager,name=UnderReplicatedPartitions
```

5. Log Flush Latency

Log flushing delays can mean I/O issues due to disk pressure:

```
kafka.log:type=LogFlushStats,name=LogFlushRateAndTimeMs
```

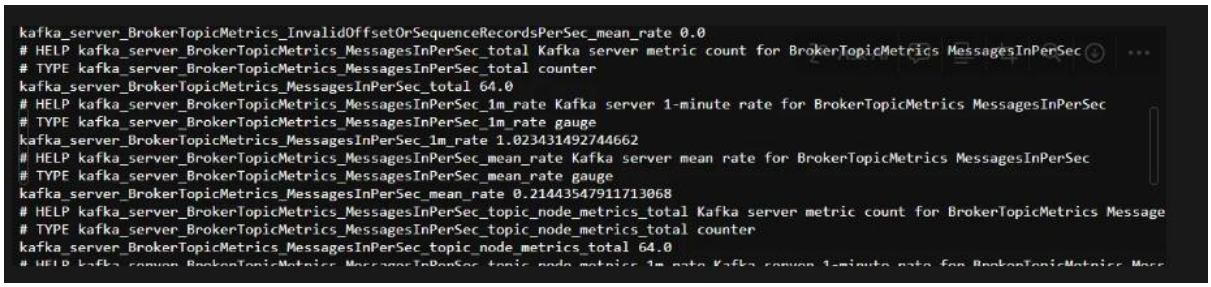
6. Controller State / Offline Partitions

```
To detect if Kafka is starting to fail due to low disk:
```

Step 4: To export these metrics out of kafka , create kafka-jmx-config.yaml file:

```
startDelaySeconds: 0

rules:

  - pattern: 'kafka.server<type=(.+), name=(.+)><>Count' name:
    kafka_server_$1_$2_total type: COUNTER

  - pattern: 'kafka.server<type=(.+), name=(.+)><>OneMinuteRate'
    name: kafka_server_$1_$2_1m_rate type: GAUGE

  - pattern: 'kafka.server<type=(.+), name=(.+)><>MeanRate' name:
    kafka_server_$1_$2_mean_rate type: GAUGE

  - pattern: 'kafka.server<type=(.+), name=(.+)><>Value' name:
    kafka_server_$1_$2 type: GAUGE labels: kafka_server: "$1"

  - pattern: 'kafka.log<type=Log, name=(.+), topic=(.+),
    partition=(.+)><>Value' name: kafka_log_$1 type: GAUGE labels:
    topic: "$2" partition: "$3"
```

Step 5: Start the kafka broker now.
The jmx exporter should ship the metrics in port 7071.



1) You can configure Prometheus to scrape this metrics from port 7071:

In prometheus.yml:

```
scrape_configs:
 - job_name: 'kafka'
   static_configs:
    - targets: ['<kafka-host>:7071']
```

2) Reload Prometheus after editing.

Test:

Visit http://<prometheus-host>:9090 and check for metric like kafka_log_remaining_bytes.

3) Make sure kafka broker and zookeeper is running and a producer is producing messages.

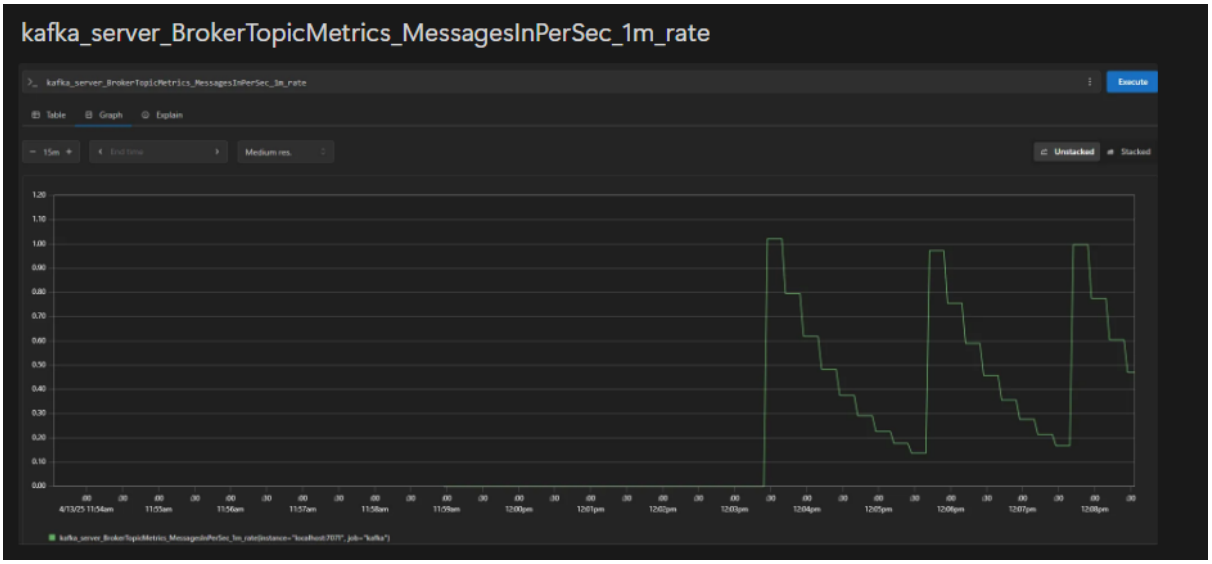
<u>Stress-testing (High CPU Load):</u>

1) We will be using the stress-ng package.

   Download stress-ng package by:

```
Sudo apt-install stress-ng
```

2) First analyse metrics over Prometheus before running the stress test:



3) Run stress testing (let’s flood the cpu load upto 70%-80%)

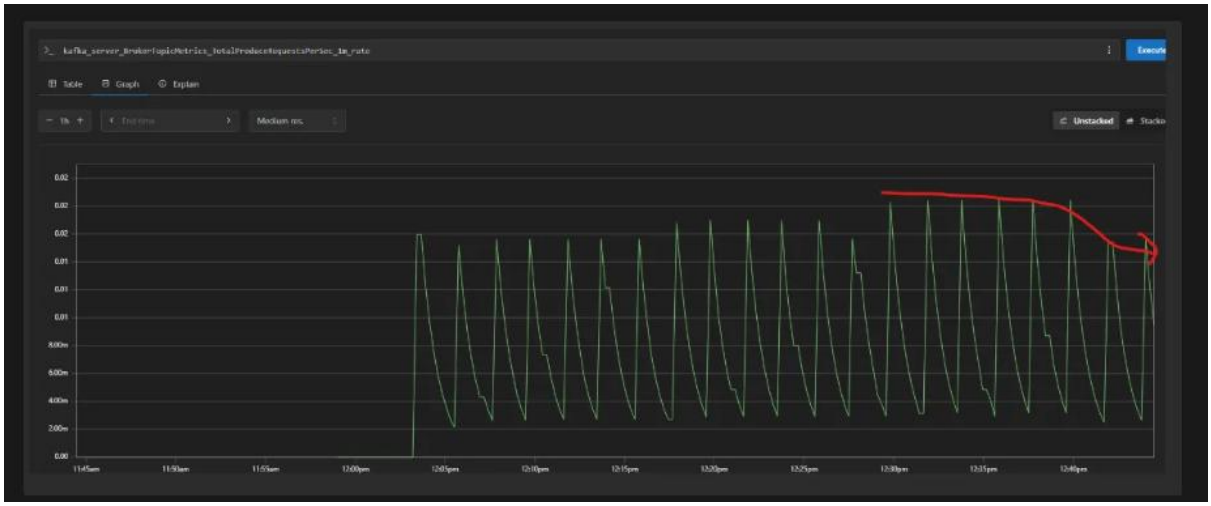
```
Command: stress --cpu 4 --timeout 600
```

This spawns **4 CPU workers** on a **2-core system**, creating **fierce contention** and pushing Kafka’s JVM threads to the edge.

JVM threads will fight for CPU time, causing request handler timeouts, GC delays, lag in producers/consumers, or even ZooKeeper timeouts.

Results:

Decreased rate of message requests produced persec.

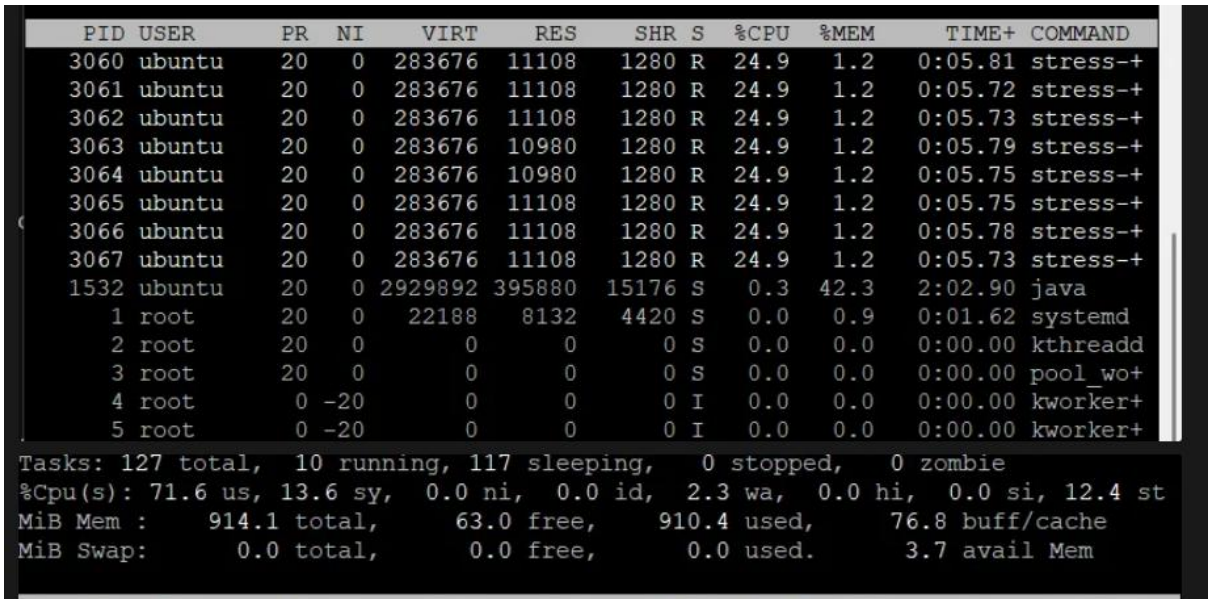


4)Extreme Stress testing

Using stress-ng

```
stress-ng --cpu 6 --vm 2 --vm-bytes 90% --timeout 600s

stress-ng \
--cpu 6 \
--cpu-method matrixprod \
--vm 2 \
--vm-bytes 90% \
--hdd 2 \
--io 4 \
--timeout 600s \
--metrics-brief
```

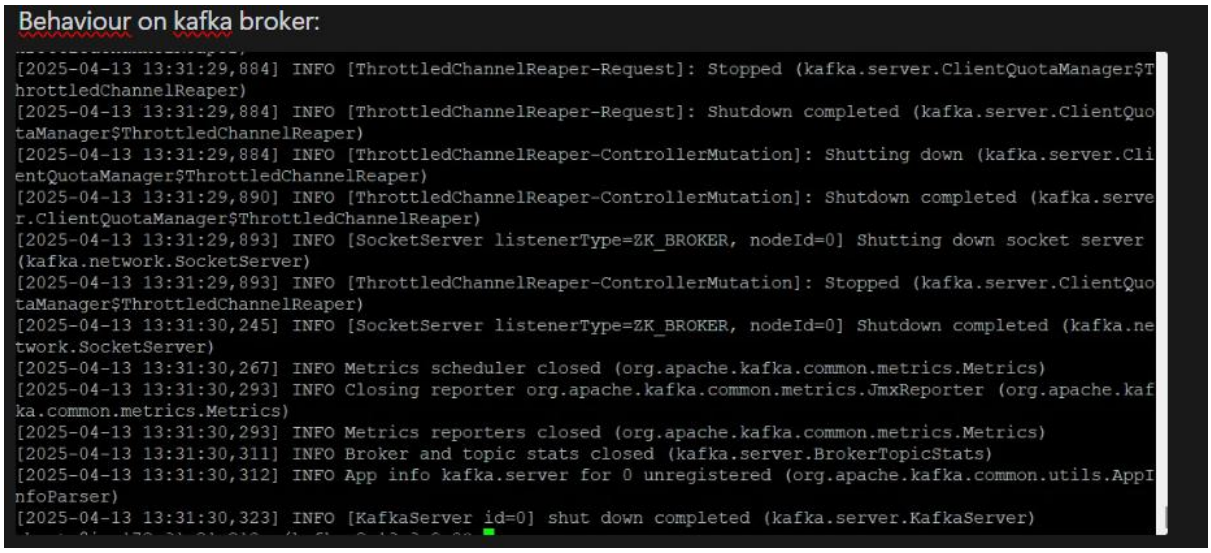
Check memory usage using command : top



As you can see, we are almost at 98-99% of memory usage, 910 MiB used out of 914 Mib

Results:

Prometheus starts breaking, unable to capture metrics of kafka.



Observation: Controlled shutdown initiated by kafka broker after few seconds.