

WEEK 5: Kafka Producer & Consumer with CPU Load Simulation

PRODUCER (Running on Google Colab)

Install Kafka Python client:

```
!pip install kafka-python
```

Python Script:

```
import time
from kafka import KafkaProducer
import requests
import json

# Kafka Producer Configuration
producer = KafkaProducer(
    bootstrap_servers='3.108.236.121:9092', # Replace with your Kafka
    broker address
    value_serializer=lambda v: json.dumps(v).encode('utf-8') # Serialize
    dictionary to JSON
)

while True:
    try:
        # Fetch Node Exporter Metrics
        response = requests.get('http://3.108.236.121:9100/metrics') #
        Replace with your URL
        metrics = response.text

        # Send subset of metrics to Kafka
        for line in metrics.splitlines():
            if line.startswith('node_cpu_seconds_total'):
                data = {"metric": line}
                print("Sending metric to Kafka:", data)
                producer.send('test-topic', value=data)

        print("Metrics sent to Kafka topic 'test-topic'")
        producer.flush()
        time.sleep(10) # Send every 10 seconds

    except Exception as e:
        print(f"Error fetching or sending metrics: {e}")
        time.sleep(10)
```

CONSUMER (Running on Google Colab)

Install Kafka Python client:

```
!pip install kafka-python
```

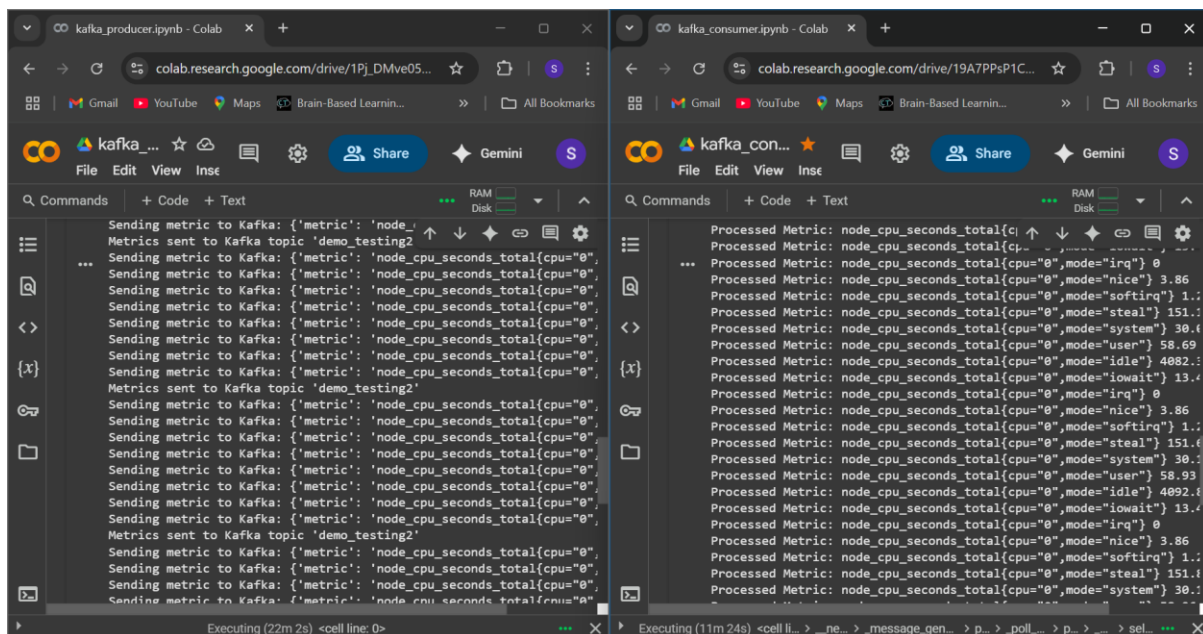
Python Script:

```
from kafka import KafkaConsumer
import json

# Kafka consumer
consumer = KafkaConsumer(
    'test-topic',
    bootstrap_servers='3.108.236.121:9092',
    auto_offset_reset='earliest',
    enable_auto_commit=True,
    group_id='metrics-consumer-group',
    value_deserializer=lambda v: json.loads(v.decode('utf-8'))
)

print("Listening for messages on topic 'test-topic'...")

try:
    for message in consumer:
        metric_name = message.value.get("metric", "N/A")
        print(f"Processed Metric: {metric_name}")
except KeyboardInterrupt:
    print("Consumer stopped.")
    consumer.close()
```



CPU LOAD SIMULATION – M1

1. Simulating CPU Load via Math Computation

Create the directory and script:

```
mkdir cpu_simulation
cd cpu_simulation
nano cpu_load_simulator.py
```

Python Script:

```
import math
import time

# Simulate CPU load by continuously performing computations
def simulate_cpu_load():
    while True:
        _ = math.factorial(100000) # Simulating CPU-bound operation
        time.sleep(0.1)

if __name__ == "__main__":
    simulate_cpu_load()
```

Run the script in the background:

```
python3 cpu_load_simulator.py &
```

2. View CPU Load Using `top`

```
top
```

This command provides a dynamic real-time view of CPU, memory usage, and running processes.

3. Using `psutil` to Export CPU Usage via Prometheus

Install and edit script:

```
nano cpu_load_simulator1.py
```

Python Script:

```
from prometheus_client import start_http_server, Gauge
import time
import psutil

# Create a metric to track CPU usage
cpu_usage_gauge = Gauge('python_cpu_seconds_total', 'Total CPU usage by Python')
```

```
# Start Prometheus server to expose metrics
start_http_server(8000)

while True:
    cpu_usage = psutil.cpu_percent(interval=1)
    cpu_usage_gauge.set(cpu_usage)
    time.sleep(1)
```

Run the script in background:

```
python3 cpu_load_simulator1.py &
```

Test with curl:

```
curl http://localhost:8000/
```

OUTPUT SCREENSHOT:

```
# HELP python_cpu_seconds_total Total CPU usage by Python
# TYPE python_cpu_seconds_total gauge
python_cpu_seconds_total 88.0
[ec2-user@ip-172-31-0-104 cpu_simulation]$ curl http://localhost:8000/
# HELP python_gc_objects_collected_total Objects collected during gc
# TYPE python_gc_objects_collected_total counter
python_gc_objects_collected_total(generation="0") 69.0
python_gc_objects_collected_total(generation="1") 279.0
python_gc_objects_collected_total(generation="2") 0.0
# HELP python_gc_objects_uncollectable_total Uncollectable objects found during GC
# TYPE python_gc_objects_uncollectable_total counter
python_gc_objects_uncollectable_total(generation="0") 0.0
python_gc_objects_uncollectable_total(generation="1") 0.0
python_gc_objects_uncollectable_total(generation="2") 0.0
# HELP python_gc_collections_total Number of times this generation was collected
# TYPE python_gc_collections_total counter
python_gc_collections_total(generation="0") 44.0
python_gc_collections_total(generation="1") 4.0
python_gc_collections_total(generation="2") 0.0
# HELP python_info Python platform information
# TYPE python_info gauge
python_info(implementation="CPython",major="3",minor="7",patchlevel="16",version="3.7.16") 1.0
# HELP process_virtual_memory_bytes Virtual memory size in bytes.
# TYPE process_virtual_memory_bytes gauge
process_virtual_memory_bytes 3.93080832e+08
# HELP process_resident_memory_bytes Resident memory size in bytes.
# TYPE process_resident_memory_bytes gauge
process_resident_memory_bytes 2.3360064e+07
# HELP process_start_time_seconds Start time of the process since unix epoch in seconds.
# TYPE process_start_time_seconds gauge
process_start_time_seconds 1.7446452213e+09
# HELP process_cpu_seconds_total Total user and system CPU time spent in seconds.
# TYPE process_cpu_seconds_total counter
process_cpu_seconds_total 0.1
# HELP process_open_fds Number of open file descriptors.
# TYPE process_open_fds gauge
process_open_fds 6.0
# HELP process_max_fds Maximum number of open file descriptors.
# TYPE process_max_fds gauge
process_max_fds 65535.0
# HELP python_cpu_seconds_total Total CPU usage by Python
# TYPE python_cpu_seconds_total gauge
python_cpu_seconds_total 91.3
[ec2-user@ip-172-31-0-104 ~]$ curl http://localhost:8000/
# HELP python_gc_objects_collected_total Objects collected during gc
# TYPE python_gc_objects_collected_total counter
python_gc_objects_collected_total(generation="0") 69.0
python_gc_objects_collected_total(generation="1") 279.0
python_gc_objects_collected_total(generation="2") 0.0
# HELP python_gc_objects_uncollectable_total Uncollectable objects found during GC
# TYPE python_gc_objects_uncollectable_total counter
python_gc_objects_uncollectable_total(generation="0") 0.0
python_gc_objects_uncollectable_total(generation="1") 0.0
python_gc_objects_uncollectable_total(generation="2") 0.0
# HELP python_gc_collections_total Number of times this generation was collected
# TYPE python_gc_collections_total counter
python_gc_collections_total(generation="0") 44.0
python_gc_collections_total(generation="1") 4.0
python_gc_collections_total(generation="2") 0.0
# HELP python_info Python platform information
# TYPE python_info gauge
python_info(implementation="CPython",major="3",minor="7",patchlevel="16",version="3.7.16") 1.0
# HELP process_virtual_memory_bytes Virtual memory size in bytes.
# TYPE process_virtual_memory_bytes gauge
process_virtual_memory_bytes 3.93080832e+08
# HELP process_resident_memory_bytes Resident memory size in bytes.
# TYPE process_resident_memory_bytes gauge
process_resident_memory_bytes 2.2249472e+07
# HELP process_start_time_seconds Start time of the process since unix epoch in seconds.
# TYPE process_start_time_seconds gauge
process_start_time_seconds 1.7446444272e+09
# HELP process_cpu_seconds_total Total user and system CPU time spent in seconds.
# TYPE process_cpu_seconds_total counter
process_cpu_seconds_total 0.2
# HELP process_open_fds Number of open file descriptors.
# TYPE process_open_fds gauge
process_open_fds 6.0
# HELP process_max_fds Maximum number of open file descriptors.
# TYPE process_max_fds gauge
process_max_fds 65535.0
# HELP python_cpu_seconds_total Total CPU usage by Python
# TYPE python_cpu_seconds_total gauge
python_cpu_seconds_total 68.5
```

4. Stop Any Running Python CPU Scripts

Find and kill process:

```
ps aux | grep python
kill <PID> # Replace <PID> with actual process ID
```

CPU LOAD SIMULATION – M2

Using `stress` to Simulate CPU Load on EC2

Install `stress`:

```
sudo yum install -y epel-release
sudo amazon-linux-extras install epel -y
sudo yum install stress -y
```

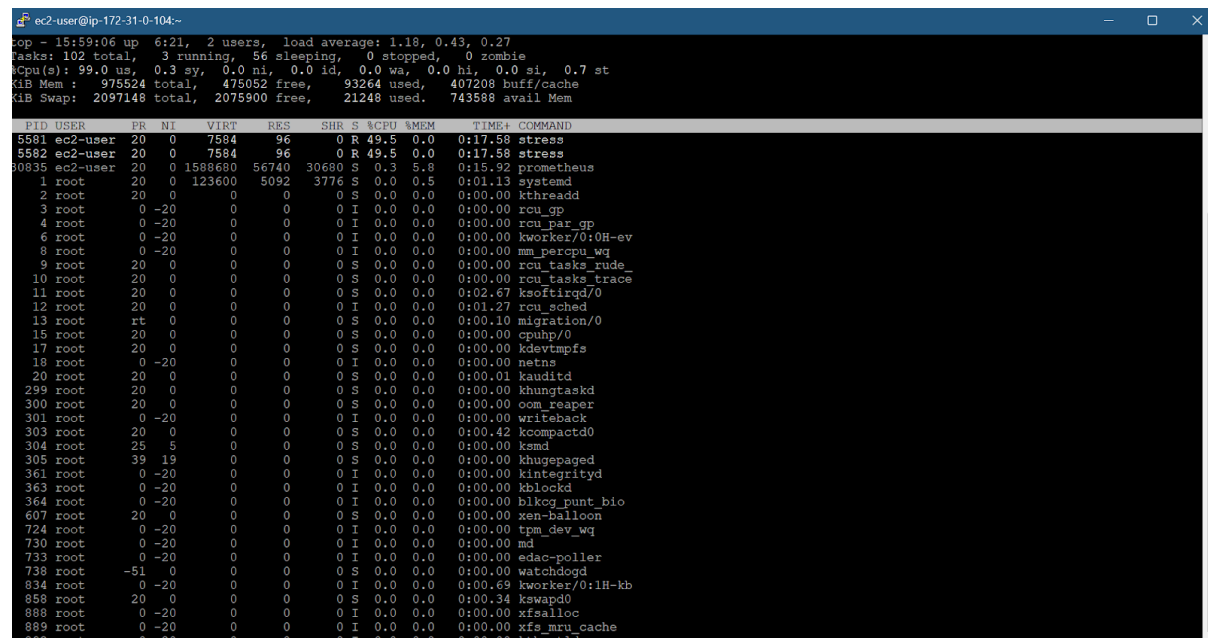
Run `stress`:

```
stress --cpu 2 --timeout 60
```

Use a second terminal to monitor with:

```
top
```

OUTPUT SCREENSHOT:



The screenshot shows a terminal window titled 'ec2-user@ip-172-31-0-104:~'. The top of the terminal displays system status: 'top - 15:59:06 up 6:21, 2 users, load average: 1.18, 0.43, 0.27'. Below this, it shows 'Tasks: 102 total, 3 running, 56 sleeping, 0 stopped, 0 zombie' and 'Cpu(s): 99.0 us, 0.3 sy, 0.0 ni, 0.0 id, 0.0 wa, 0.0 hi, 0.0 si, 0.7 st'. The memory section shows 'Mem: 975524 total, 475052 free, 93264 used, 407208 buff/cache' and 'Swap: 2097148 total, 2075900 free, 21248 used, 743588 avail Mem'. The main part of the terminal is a 'top' command output showing a list of processes. The first two processes are 'stress' (PID 5581 and 5582) running as 'ec2-user' with 20% CPU and 0% memory usage. The rest of the processes are system daemons and services running as 'root' with varying CPU and memory usage.

PID	USER	PR	NI	VIRT	RES	SHR	S	%CPU	%MEM	TIME+	COMMAND
5581	ec2-user	20	0	7584	96	0	R	49.5	0.0	0:17.58	stress
5582	ec2-user	20	0	7584	96	0	R	49.5	0.0	0:17.58	stress
30835	ec2-user	20	0	1588680	56740	30680	S	0.3	5.8	0:15.92	prometheus
1	root	20	0	123600	5092	3776	S	0.0	0.5	0:01.13	systemd
2	root	20	0	0	0	0	S	0.0	0.0	0:00.00	kthreadd
3	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	rcu_gp
4	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	rcu_par_gp
6	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	kworker/0:0H-ev
8	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	mm_percpu_wq
9	root	20	0	0	0	0	S	0.0	0.0	0:00.00	rcu_tasks_rude
10	root	20	0	0	0	0	S	0.0	0.0	0:00.00	rcu_tasks_trace
11	root	20	0	0	0	0	S	0.0	0.0	0:02.67	ksoftirqd/0
12	root	20	0	0	0	0	I	0.0	0.0	0:01.27	rcu_sched
13	root	rt	0	0	0	0	S	0.0	0.0	0:00.10	migration/0
15	root	20	0	0	0	0	S	0.0	0.0	0:00.00	cpuhp/0
17	root	20	0	0	0	0	S	0.0	0.0	0:00.00	kdevtmpfs
18	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	netns
20	root	20	0	0	0	0	S	0.0	0.0	0:00.01	kauditd
299	root	20	0	0	0	0	S	0.0	0.0	0:00.00	khungtaskd
300	root	20	0	0	0	0	S	0.0	0.0	0:00.00	oom_reaper
301	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	writeback
303	root	20	0	0	0	0	S	0.0	0.0	0:00.42	kcompactd0
304	root	25	5	0	0	0	S	0.0	0.0	0:00.00	kamd
305	root	39	19	0	0	0	S	0.0	0.0	0:00.00	khugepaged
361	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	kintegrityd
363	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	kblockd
364	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	blkcg_punt_bio
607	root	20	0	0	0	0	S	0.0	0.0	0:00.00	xen-balloon
724	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	tpm_dev_wq
730	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	md
733	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	edac-poller
738	root	-51	0	0	0	0	S	0.0	0.0	0:00.00	watchdogd
834	root	0	-20	0	0	0	I	0.0	0.0	0:00.69	kworker/0:1H-kb
858	root	20	0	0	0	0	S	0.0	0.0	0:00.34	xswpd0
888	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	xfsalloc
889	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	xfs_mru_cache
892	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	kthrotl