# DA5402: Assignment 6

Let's now learn to build an AI application with instrumentation using Prometheus. We learned how to use exporters to monitor standard application and about building an exporter for our application via Promotheus instrumentation library. Let's try creating an exporter for exporting the system (node) level metrics without using **`node_exporter`** or **`windows_exporter`**.

## Task 1 [20 points]

In the Linux environment, we have a command name **`iostat`** which lists the disk level IO statistics like the snapshot below.

```
sudarsun@kappa ~> iostat
Linux 5.15.0-134-generic (kappa)        23/03/25        _x86_64_        (16 CPU)

avg-cpu:  %user   %nice %system %iowait  %steal   %idle
           2.61    0.00    1.19    0.04    0.00   96.16

Device             tps    kB_read/s    kB_wrtn/s    kB_dscd/s    kB_read    kB_wrtn    kB_dscd
nvme0n1          30.17       431.22       992.42         0.00    4276880    9842997          0
nvme1n1          14.10       224.51        24.42         0.00    2226744     242228          0
sda               0.06         0.93         0.00         0.00       9223          0          0
sdb               0.09         1.12         0.00         0.00      11109         12          0
```

We can run this command repeatedly using **`watch -n1 iostat`** to have a live monitoring from the command line. You may also redirect the output of **`iostat`** into a file for further processing using the command **`iostat > /tmp/io`**, which will appear like below.

```
sudarsun@kappa ~> cat /tmp/io
Linux 5.15.0-134-generic (kappa)        23/03/25        _x86_64_        (16 CPU)

avg-cpu:  %user   %nice %system %iowait  %steal   %idle
           2.56    0.00    1.19    0.04    0.00   96.21

Device             tps    kB_read/s    kB_wrtn/s    kB_dscd/s    kB_read    kB_wrtn    kB_dscd
nvme0n1          29.84       417.85       988.41         0.00    4278452   10120369          0
nvme1n1          13.66       217.47        23.66         0.00    2226744     242228          0
sda               0.06         0.90         0.00         0.00       9223          0          0
sdb               0.09         1.08         0.00         0.00      11109         12          0
```

You can process the text dump to collect the statistics. The objective here is to convert the collected statistics into suitable Prometheus metric types with appropriate names. And you repeat the data collection every second (using **`crontab`** or via scripting), so Prometheus service can pick the metrics at the **`scrape_interval >= 1s`**. The metrics to be exported are:

- io_read_rate{device=``xxx"}
- io_write_rate{device=``xxx"}
- io_tps{device=``xxx"}
- io_read_bytes{device=``xxx"}
- io_write_bytes{device=``xxx"}
- cpu_avg_percent{mode=``xxx"}  xxx \in {user, nice, system, iowait, idle}

## Task 2 [20 points]

Likewise, we can read the current memory information from **`/proc/meminfo`**, which appears like below:

```
sudarsun@kappa ~ [1]> cat /proc/meminfo
MemTotal:       32797544 kB
MemFree:        14501236 kB
MemAvailable:   23688392 kB
Buffers:            9824 kB
Cached:          9610412 kB
SwapCached:            0 kB
Active:          3574364 kB
Inactive:       13267216 kB
Active(anon):      47116 kB
Inactive(anon):  7651520 kB
Active(file):    3527248 kB
Inactive(file):  5615696 kB
Unevictable:          48 kB
Mlocked:              48 kB
SwapTotal:       6850556 kB
SwapFree:        6850556 kB
Dirty:              2048 kB
Writeback:             0 kB
AnonPages:       7221596 kB
Mapped:          1825976 kB
Shmem:            477288 kB
KReclaimable:     509016 kB
Slab:             829992 kB
SReclaimable:     509016 kB
SUnreclaim:       320976 kB
KernelStack:       27648 kB
PageTables:        81604 kB
NFS_Unstable:          0 kB
Bounce:                0 kB
WritebackTmp:          0 kB
CommitLimit:    23249328 kB
Committed_AS:   40505704 kB
VmallocTotal:   34359738367 kB
VmallocUsed:      133612 kB
VmallocChunk:          0 kB
Percpu:            30976 kB
HardwareCorrupted:     0 kB
AnonHugePages:         0 kB
ShmemHugePages:        0 kB
ShmemPmdMapped:        0 kB
FileHugePages:         0 kB
FilePmdMapped:         0 kB
HugePages_Total:       0
HugePages_Free:        0
HugePages_Rsvd:        0
HugePages_Surp:        0
Hugepagesize:       2048 kB
Hugetlb:               0 kB
DirectMap4k:     1860452 kB
DirectMap2M:    18003968 kB
DirectMap1G:    13631488 kB
```

The file updates live, which can be observed using **watch -n1 cat /proc/meminfo** command. Similar to Task 1, you should repeatedly read from **/proc/meminfo** every second to monitor the live memory statistics, followed by exporting them as Prometheus metrics. Every line item from the meminfo listing should be exported as a metric. Every metric from meminfo should have **meminfo_** as the prefix. For instance the line item **MemFree** should become **meminfo_free_memory** as the metric.

Note the, both tasks should be implemented as individual functions in a single Python script, which would expose the metrics at port 18000. So, http://localhost:18000/metrics should report all the exported metrics from both functions. As usual, use of logging mechanism is mandatory. Comments around the source code block is crucial.

## Task 3 [10 points]

Setup Prometheus server and configure it to scrape from your instrumented application by setting the scrape interval to 2 seconds.  Ensure that the metrics your exposed from the app are queryable from Prometheus UI console.