GARNET | Ghanaian Academic and Research Network

DAY 1 · SESSION 3

Prometheus Core Concepts

The data model, metric types, and prometheus.yml

RECAP

What we have so far

Prometheus is running

Container is up. Web UI on :9090. /targets shows one green target — Prometheus scraping itself.

Grafana is running

Container is up. Web UI on :3000. We will add Prometheus as a data source after this session.

Mental model is in place

Pull-based, dimensional, single-binary, time-series. Three ideas from the first session.

Now: what is actually inside Prometheus, and how do we tell it what to scrape?

DATA MODEL

Anatomy of a sample

```
node_network_receive_bytes_total{device="eth0", instance="server-01:9100"}  847219338  @1714502400
```

METRIC NAME	LABELS	VALUE	TIMESTAMP
what is being measured	*dimensions / context*	*the number*	*when (unix epoch)*

Every sample is exactly these four things — nothing more, nothing less.

VOCABULARY

Four metric types

Counter

Only goes up. Resets to zero on restart.

e.g. Total requests served

Gauge

Goes up and down. Snapshot of a current value.

e.g. Memory in use right now

Histogram

Distributions in buckets — for latency / size.

e.g. Request duration buckets

Summary

Like histogram, but quantiles computed client-side.

e.g. p99 latency at scrape time

METRIC TYPE 1

Counters: the running total

http_requests_total

t=0s	1,247
t=15s	1,289
t=30s	1,331
t=45s	1,373
t=60s	1,415

Rules of counters

- Always increases (or resets to 0)
- Raw value is rarely useful
- Use `rate()` to get per-second change
- Convention: name ends in `_total`
- Reset on restart is normal — Prometheus handles it

"What was the request rate?" is meaningful. "What is the request count?" is not — the answer just keeps growing.

METRIC TYPE 2

Gauges: the snapshot

node_memory_MemAvailable_bytes

t=0s	**4.2 GB**
t=15s	**3.9 GB**
t=30s	**4.0 GB**
t=45s	**3.7 GB**
t=60s	**4.1 GB**

Rules of gauges

- Goes up AND down freely
- Raw value IS meaningful — it's what you graph
- Common functions: `avg_over_time`, `max_over_time`
- No naming convention — just the unit suffix
- Examples: temperature, queue length, link up/down

> *Many SNMP "status" values are gauges with two states: 1 = up, 0 = down. We will see this on Day 2.*

METRIC TYPE 3

Histograms: distribution in buckets

A histogram is several time series at once — one per bucket boundary, plus _count and _sum.

```
http_request_duration_seconds_bucket{le="0.1"} 1247
http_request_duration_seconds_bucket{le="0.5"} 1820
http_request_duration_seconds_bucket{le="1.0"} 1955
http_request_duration_seconds_bucket{le="+Inf"} 2003
http_request_duration_seconds_count 2003
http_request_duration_seconds_sum 412.7
```

Read: *1,247 requests took ≤ 0.1s · 573 took 0.1-0.5s · 135 took 0.5-1.0s · 48 took >1s*

CONFIGURATION

Anatomy of prometheus.yml

```yaml
global:
  scrape_interval: 15s
  evaluation_interval: 15s

scrape_configs:
  - job_name: 'prometheus'
    static_configs:
      - targets: ['localhost:9090']

  - job_name: 'node'
    static_configs:
      - targets:
          - 'host-01:9100'
          - 'host-02:9100'
```

Three things to know

`global`

Defaults that apply to every scrape job below.

`job_name`

Becomes a label on every metric scraped from this job.

`targets`

List of host:port endpoints to scrape.

THE /metrics ENDPOINT

What scraping actually fetches

Plain text. One metric per line. Comments optional. That's all.

```
# HELP node_load1 1m load average.
# TYPE node_load1 gauge
node_load1 0.42

# HELP node_network_receive_bytes_total Network receive in bytes.
# TYPE node_network_receive_bytes_total counter
node_network_receive_bytes_total{device="eth0"} 8.471e+08
node_network_receive_bytes_total{device="lo"} 21337

# HELP node_filesystem_avail_bytes Filesystem space available in bytes.
# TYPE node_filesystem_avail_bytes gauge
node_filesystem_avail_bytes{mountpoint="/"} 4.2e+10
```

Try it: curl http://localhost:9090/metrics — it's the same simple format, all the way down.

TSDB

Storage basics

~1-2

bytes/sample

After compression. TSDB is space-efficient.

15 days

default retention

Configurable via --storage.tsdb.retention.time

Local

disk only

No replication built-in. Use Thanos/Mimir for HA & long-term.

Disk planning rule of thumb: series × samples_per_second × bytes_per_sample × seconds

```
100,000 series × 1/15s × 1.5 bytes × 15 days ≈ 13 GB
```

CARDINALITY

The one trap to avoid

High-cardinality labels will crush your Prometheus.

Each unique combination of label values is a separate time series.

Good labels

- `device="eth0"` (a few values)
- `instance="router-1"` (bounded)
- `method="GET"` (small set)
- `status="200"` (small set)

Dangerous labels

- `user_id` (millions of values)
- `request_id` (unique each request)
- `url` with query string
- `ip_address` (high cardinality)

TO THE KEYBOARD

Now let's add a real target

Hands-on: edit prometheus.yml, install Node Exporter, see metrics flow end-to-end.