Mini Prometheus-Grafana

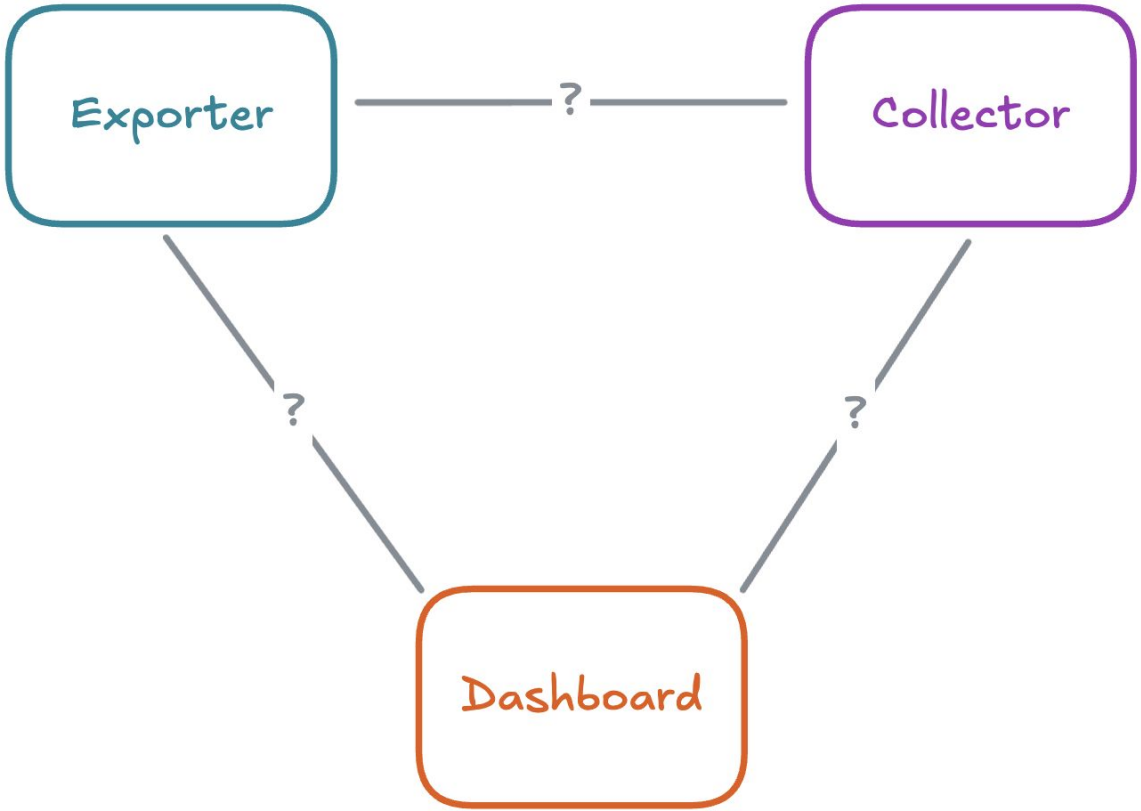
Hasan Kerem Şeker
Yunus Emre Özdemir

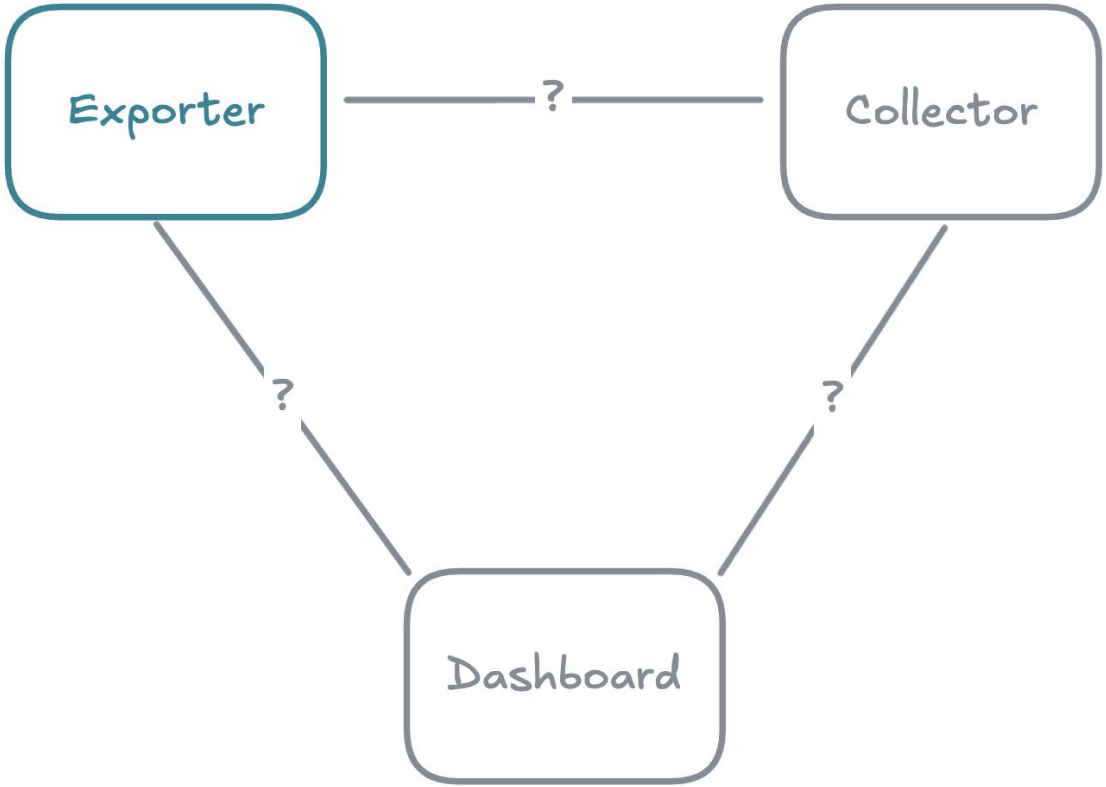
What is  Prometheus

What is Grafana

Aim

Lightweight, Real-time System Monitoring Framework


Exporter
?
Collector
?
?
Dashboard


Exporter
?
Collector
?
?
Dashboard

Metrics We Collect

CPU Usage

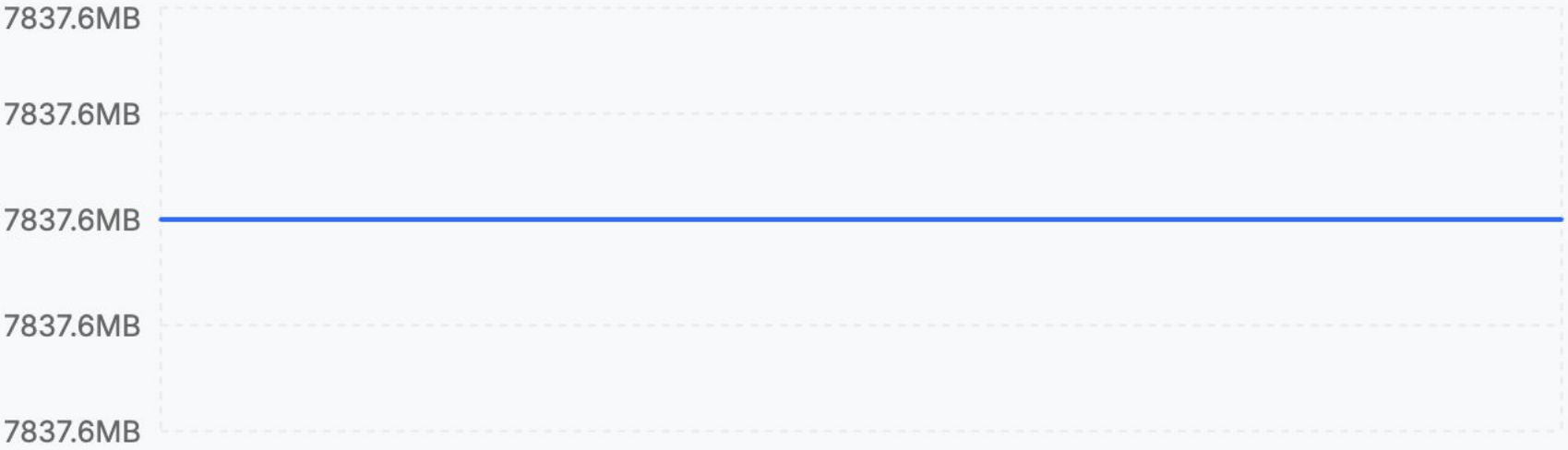
Current: 0.0%

Free Memory

Current: 6088.3MB

Total Memory

Current: 7837.6MB



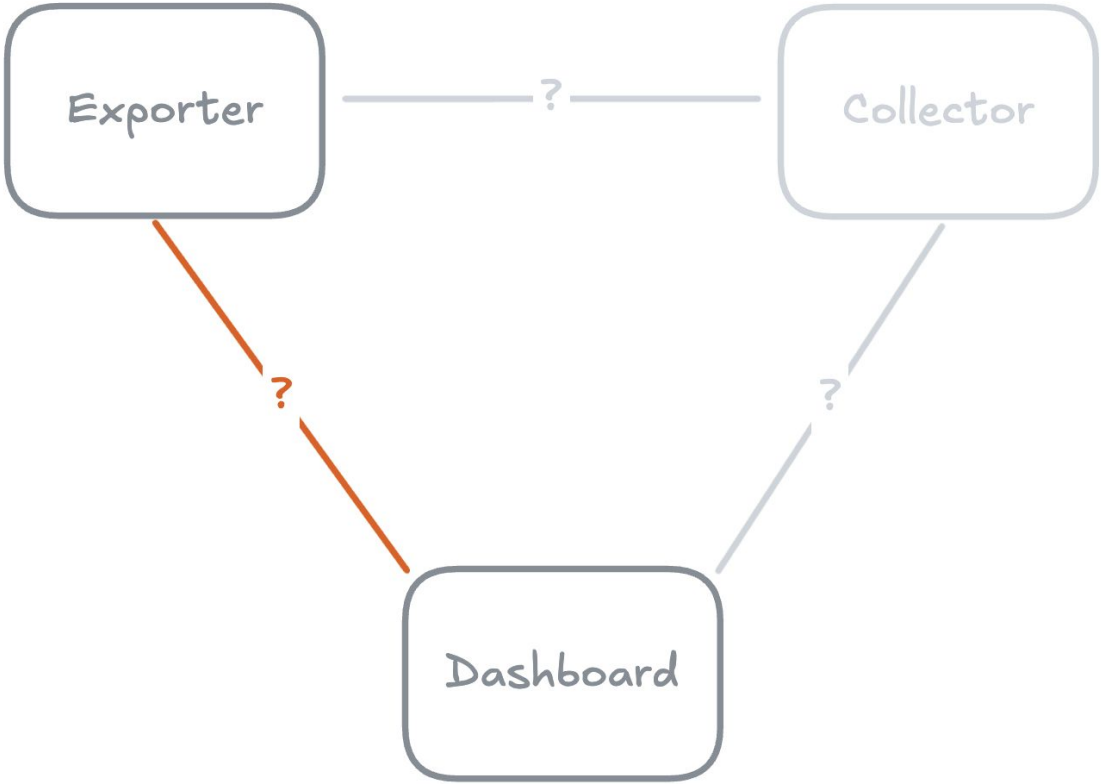
Thread Count

Current: 20

How Do We Collect the Metrics?

```
com.sun.management.OperatingSystemMXBean sunOsBean = (com.sun.management.OperatingSystemMXBean) osBean;

double cpuUsage = sunOsBean.getCpuLoad();
long totalMemory = sunOsBean.getTotalMemorySize();
long freeMemory = sunOsBean.getFreeMemorySize();
int threadCount = threadBean.getThreadCount();
```


Exporter
?
Collector
?
?
Dashboard

Load CPU

Load Memory

Load Threads

Free Resources

Clear Data

How do we load the system?

- Load Memory: http://localhost:8081/load/memory?duration=20
- Load Cpu: http://localhost:8081/load/cpu?duration=10
- Load Thread: http://localhost:8081/load/thread?duration=10
- Free the resources: http://localhost:8081/load/free

Load CPU

```
// Matrix multiplication to generate CPU load
double[][] m1 = new double[100][100];
double[][] m2 = new double[100][100];
double[][] result = new double[100][100];

for (int i = 0; i < 100; i++) {
    if (stopCpuLoad) break;
    for (int j = 0; j < 100; j++) {
        for (int k = 0; k < 100; k++) {
            result[i][j] += m1[i][k] * m2[k][j];
        }
    }
}
```

Load Memory

```
while (System.currentTimeMillis() < endTime) {
    memoryBlocks.add(new byte[1024 * 1024]); // Allocate 1MB
    Thread.sleep(millis:100);
}
```

Load Threads

```
for (int i = 0; i < 100; i++) {
    Thread t = new Thread(() -> {
        try {
            Thread.sleep(durationSeconds * 1000);
        } catch (InterruptedException e) {
            Thread.currentThread().interrupt();
        }
    });
    t.start();
    threads.add(t);
}
```

Free Resources

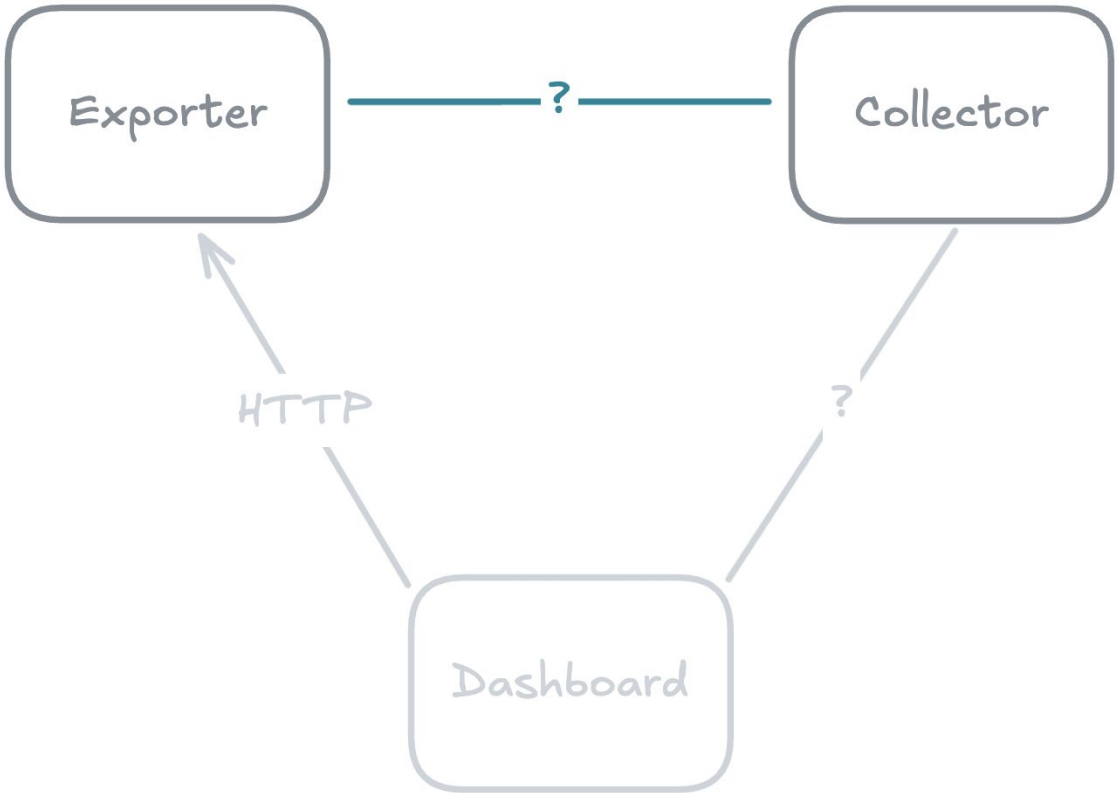
```java
private void freeResources() {
    // Signal CPU load tasks to stop
    stopCpuLoad = true;

    // Clear CPU load - force shutdown of executor service
    executorService.shutdownNow();
    try {
        // Wait for termination with timeout
        if (!executorService.awaitTermination(timeout:1, TimeUnit.SECONDS)) {
            System.err.println(x:"Executor service did not terminate in the specified time.");
        }
    } catch (InterruptedException e) {
        Thread.currentThread().interrupt();
    }
    executorService = Executors.newCachedThreadPool(); // Restart executor service

    // Clear memory load
    memoryBlocks.clear();
    System.gc(); // Request garbage collection to free memory

    // Clear thread load
    for (Thread thread : threads) {
        thread.interrupt(); // Interrupt thread
        try {
            thread.join(millis:1000); // Wait for thread to finish with timeout
        } catch (InterruptedException e) {
            Thread.currentThread().interrupt();
        }
    }
    threads.clear();

    System.out.println(x:"Resources freed successfully");
    stopCpuLoad = false; // Reset the flag for future CPU load tests
}
```


Exporter
?
Collector
HTTP
?
Dashboard

How do we send the data?

TCP

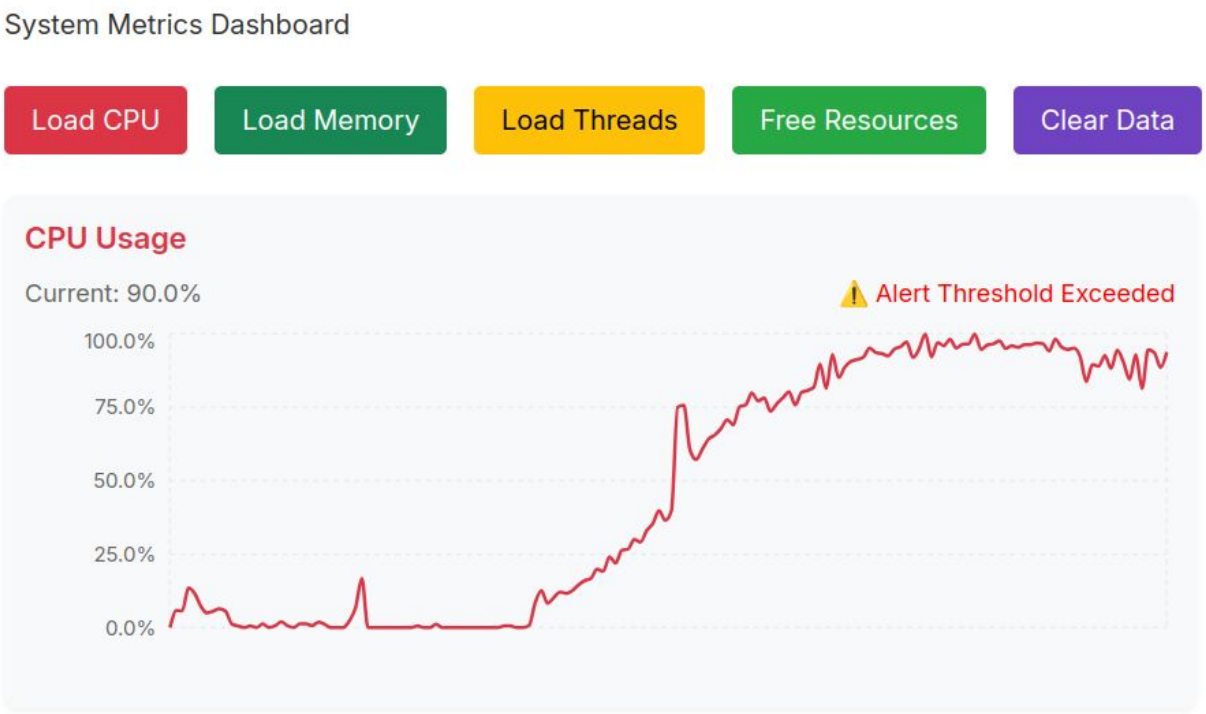
UDP

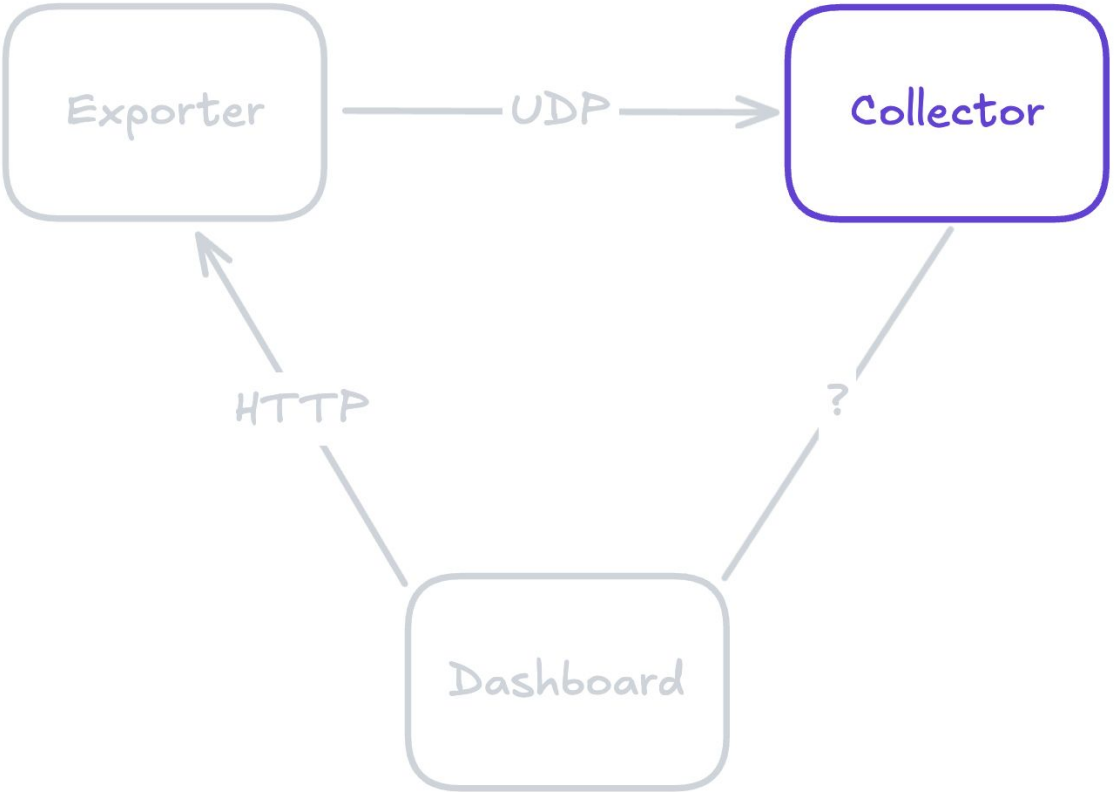
What does the UDP packet look like?

How often do we send the data?

100ms - 10 data packets per second

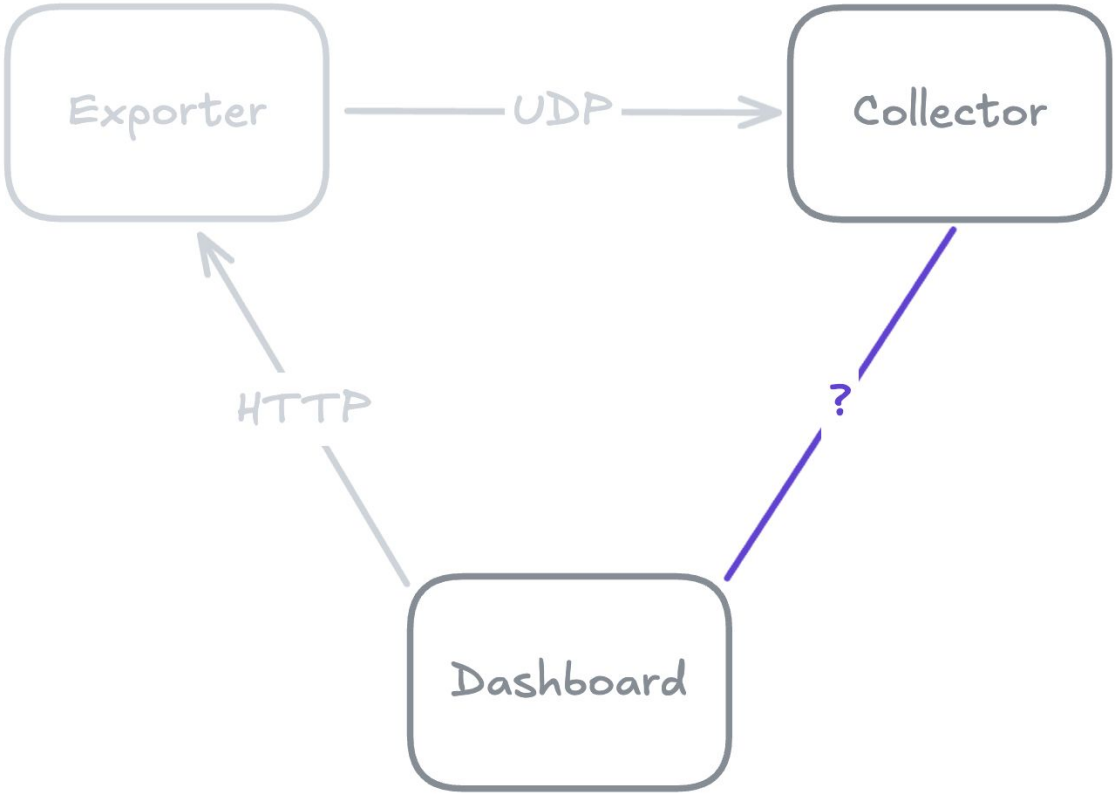
What are alerts?



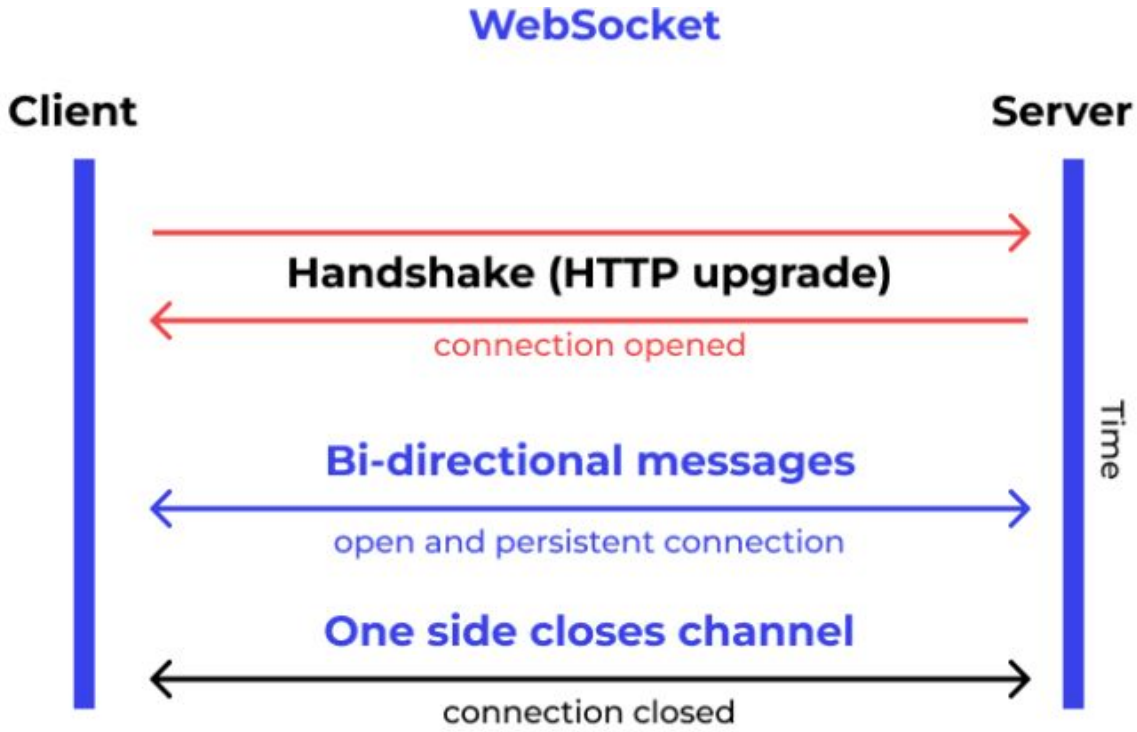

Exporter
UDP
Collector
HTTP
?
Dashboard

How do we store the data in the Collector?

```
private final Map<String, Map<String, Deque<MetricPoint>>> store;
```

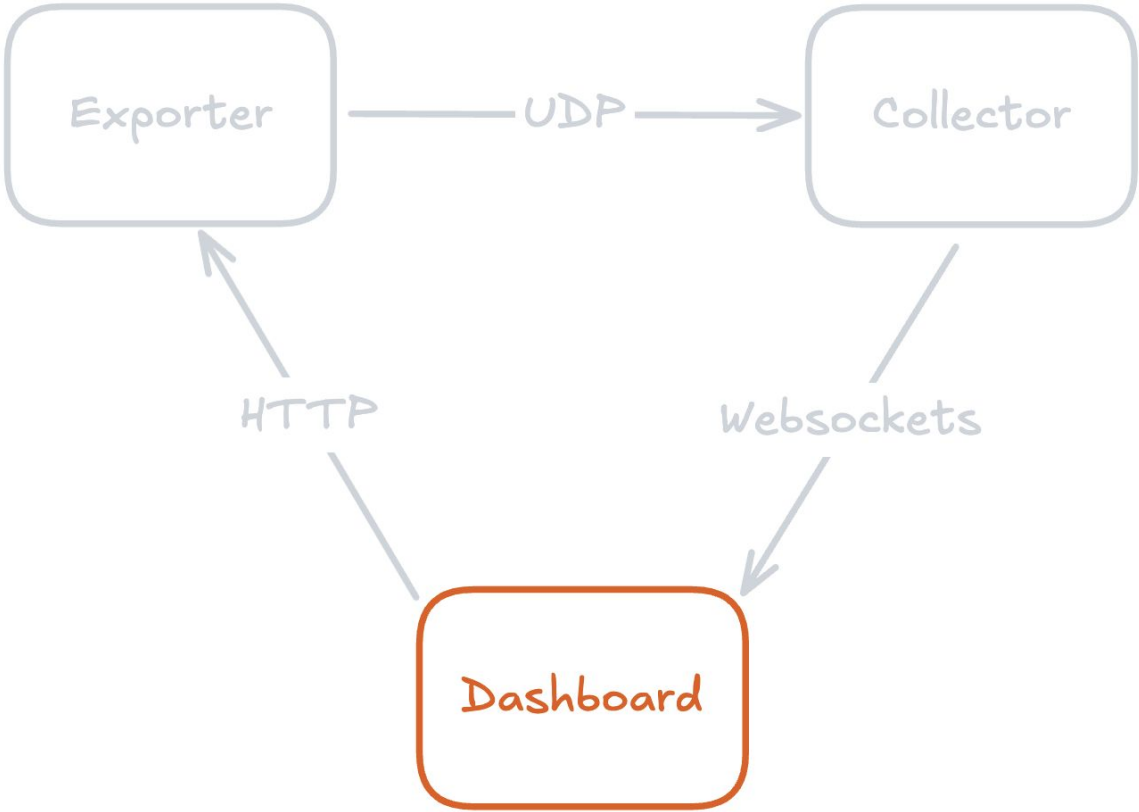

Exporter
UDP
Collector
HTTP
?
Dashboard

How do we stream the new data in real time?



How do we get the old data from the collector?

- GET http://localhost:8081/api/metrics
- DELETE http://localhost:8081/api/metrics


Exporter
UDP
Collector
HTTP
Websockets
Dashboard

Tech Stack of Dashboard

- React Router
- Recharts

System Metrics Dashboard

✅ Connected

Load CPU | Load Memory | Load Threads | Free Resources | Clear Data

CPU Usage

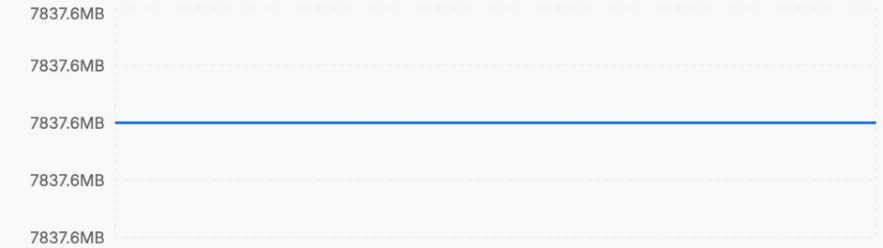
Current: 0.0%

Free Memory

Current: 6068.8MB

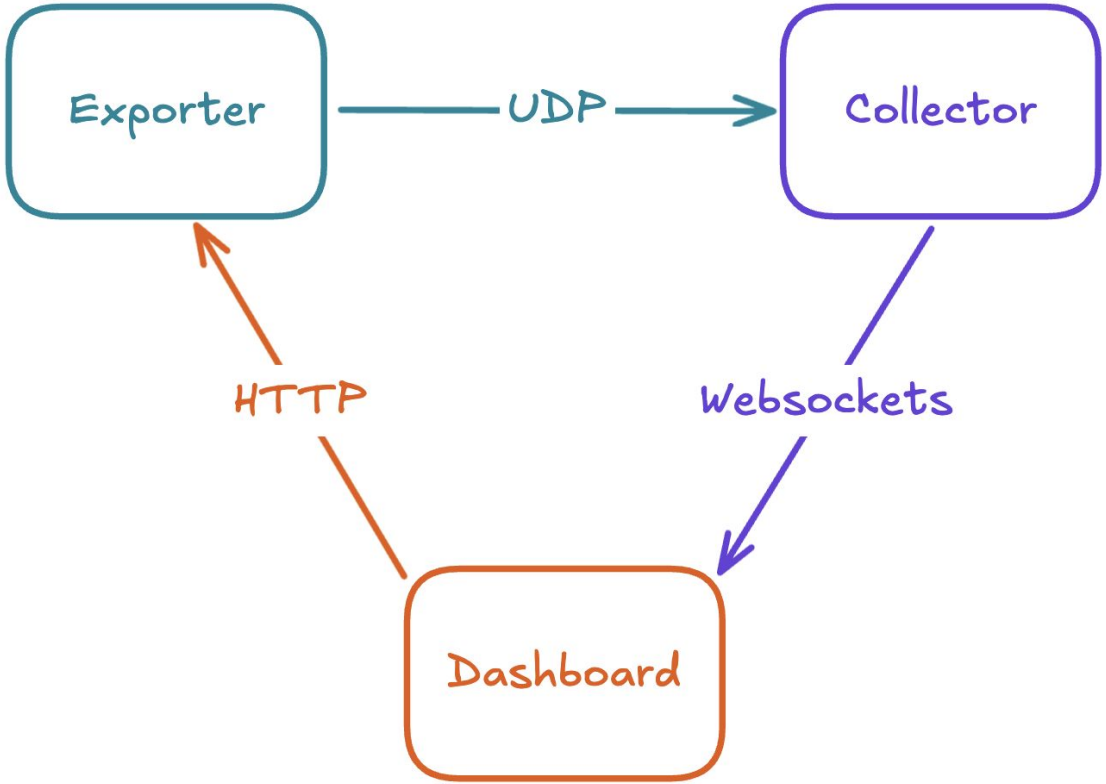
Total Memory

Current: 7837.6MB



Thread Count

Current: 20


Exporter
UDP
Collector
HTTP
Websockets
Dashboard

Demo