Monitoring Practical Task

> ## Practical task

Monitoring part

1. Rebuild your spring-petclinic Docker image to run with JMX Prometheus exporter.
2. Validate that exporter endpoint is accessible at <ip_or_hostname>:<exporter_port>/metrics
3. Run Prometheus Docker container scrapping JMX metrics.
4. Validate that Prometheus gathers data by running the following queries:
    a. java_lang_OperatingSystem_CpuLoad
    b. java_lang_OperatingSystem_FreeMemorySize
    c. java_lang_Memory_HeapMemoryUsage_used
    d. java_lang_Runtime_Uptime
5. Play around with different views and explore the user interface
6. Run Grafana container, log in, explore browser and:
    a. Create Prometheus data source in the user interface
    b. Import dashboard with its id - 10519
7. OPTIONAL: Automate the creation of data_source and dashboard resources
    a. It may be: [
       Pure API requests,
       Grafonnet and API requests,
       Ansible,
       Terraform
       ]
    b. Choose whatever fits you

Logging part

1. Install Loki and Promtail
2. Run your spring-boot binary locally in the following way:
   java <app>.jar > <app>.log
8. Run Loki with the default configuration
9. Prepare your Promtail config to read logs from <app>.log and send logs to Loki. Run it
10. Log in Grafana
    a. Add Loki data source
    b. Follow along to Explore tab and check out for logs

Monitoring

1. Rebuild your spring-petclinic Docker image to run with JMX Prometheus exporter.

I modified previous Dockerfile to install JMX exporter and run it as Java agent.

```
Dockerfile > ...
FROM maven:3.8.7-openjdk-18-slim AS build

RUN mkdir /app
COPY . /app
WORKDIR /app
RUN mvn package

FROM alpine:3.14 AS jmx_exporter
RUN wget https://github.com/prometheus/jmx_exporter/releases/download/1.1.0/
jmx_prometheus_javaagent-1.1.0.jar -O /jmx_prometheus_javaagent.jar


# Minimal rintime image - only JRE
FROM gcr.io/distroless/java21-debian12 AS runtime
COPY --from=build /app/target/*.jar /app.jar
COPY --from=jmx_exporter /jmx_prometheus_javaagent.jar /jmx_prometheus_javaagent.jar
COPY --from=build /app/jmx-exporter.conf.yaml /config.yaml
ENTRYPOINT [ "java" ]
CMD [ "-javaagent:/jmx_prometheus_javaagent.jar=8081:/config.yaml", "-jar", "/app.jar" ]
```

Below configuration file for the exporter gives all the metrics

```
jmx-exporter.conf.yaml > ...
startDelaySeconds: 0
ssl: false
lowercaseOutputName: false
lowercaseOutputLabelNames: false
```

2. Validate that exporter endpoint is accessible at <ip_or_hostname>:<exporter_port>/metrics

```
08:15:27 adrwal@olek-desktop-pc monitoring-task ±|master ✗|→ docker run -it --rm --name petclinic-jmx -p 8081:8081 petclinic-jmx-exporter
2025-01-07 19:16:52.671 | main | INFO | io.prometheus.jmx.JavaAgent | Starting ...
2025-01-07 19:16:52.850 | main | INFO | io.prometheus.jmx.JavaAgent | HTTP enabled [true]
2025-01-07 19:16:52.851 | main | INFO | io.prometheus.jmx.JavaAgent | HTTP host:port [0.0.0.0:8081]
2025-01-07 19:16:52.851 | main | INFO | io.prometheus.jmx.JavaAgent | OpenTelemetry enabled [false]
2025-01-07 19:16:52.882 | main | INFO | io.prometheus.jmx.JavaAgent | Running ...

:: Built with Spring Boot :: 3.3.5
```

```
localhost:8081/metrics
Robota | ACloudGuru In... | ACloudGuru In... | ACloudGuru In... | Docker

# HELP java_lang_ClassLoading_LoadedClassCount java.lang:name=null,type=ClassLoading,attribute=LoadedClassCount
# TYPE java_lang_ClassLoading_LoadedClassCount untyped
java_lang_ClassLoading_LoadedClassCount 17348.0
# HELP java_lang_ClassLoading_TotalLoadedClassCount java.lang:name=null,type=ClassLoading,attribute=TotalLoadedClassCount
# TYPE java_lang_ClassLoading_TotalLoadedClassCount untyped
java_lang_ClassLoading_TotalLoadedClassCount 17348.0
# HELP java_lang_ClassLoading_UnloadedClassCount java.lang:name=null,type=ClassLoading,attribute=UnloadedClassCount
# TYPE java_lang_ClassLoading_UnloadedClassCount untyped
java_lang_ClassLoading_UnloadedClassCount 0.0
# HELP java_lang_ClassLoading_Verbose java.lang:name=null,type=ClassLoading,attribute=Verbose
# TYPE java_lang_ClassLoading_Verbose untyped
java_lang_ClassLoading_Verbose 0.0
# HELP java_lang_CodeCacheManager_Valid java.lang:name=CodeCacheManager,type=MemoryManager,attribute=Valid
# TYPE java_lang_CodeCacheManager_Valid untyped
java_lang_CodeCacheManager_Valid{type="MemoryManager"} 1.0
# HELP java_lang_CodeHeap_non_nmethods_CollectionUsageThresholdSupported java.lang:name=CodeHeap 'non-nmethods',type=MemoryPool,att
# TYPE java_lang_CodeHeap_non_nmethods_CollectionUsageThresholdSupported untyped
java_lang_CodeHeap_non_nmethods_CollectionUsageThresholdSupported{type="MemoryPool"} 0.0
# HELP java_lang_CodeHeap_non_nmethods_PeakUsage_committed java.lang.management.MemoryUsage java.lang:name=CodeHeap 'non-nmethods',
# TYPE java_lang_CodeHeap_non_nmethods_PeakUsage_committed untyped
java_lang_CodeHeap_non_nmethods_PeakUsage_committed{type="MemoryPool"} 2555904.0
# HELP java_lang_CodeHeap_non_nmethods_PeakUsage_init java.lang.management.MemoryUsage java.lang:name=CodeHeap 'non-nmethods',type=
# TYPE java_lang_CodeHeap_non_nmethods_PeakUsage_init untyped
java_lang_CodeHeap_non_nmethods_PeakUsage_init{type="MemoryPool"} 2555904.0
# HELP java_lang_CodeHeap_non_nmethods_PeakUsage_max java.lang.management.MemoryUsage java.lang:name=CodeHeap 'non-nmethods',type=M
# TYPE java_lang_CodeHeap_non_nmethods_PeakUsage_max untyped
java_lang_CodeHeap_non_nmethods_PeakUsage_max{type="MemoryPool"} 5840896.0
# HELP java_lang_CodeHeap_non_nmethods_PeakUsage_used java.lang.management.MemoryUsage java.lang:name=CodeHeap 'non-nmethods',type=
# TYPE java_lang_CodeHeap_non_nmethods_PeakUsage_used untyped
java_lang_CodeHeap_non_nmethods_PeakUsage_used{type="MemoryPool"} 1801344.0
# HELP java_lang_CodeHeap_non_nmethods_UsageThreshold java.lang:name=CodeHeap 'non-nmethods',type=MemoryPool,attribute=UsageThresho
# TYPE java_lang_CodeHeap_non_nmethods_UsageThreshold untyped
java_lang_CodeHeap_non_nmethods_UsageThreshold{type="MemoryPool"} 0.0
# HELP java_lang_CodeHeap_non_nmethods_UsageThresholdCount java.lang:name=CodeHeap 'non-nmethods',type=MemoryPool,attribute=UsageTh
# TYPE java_lang_CodeHeap_non_nmethods_UsageThresholdCount untyped
java_lang_CodeHeap_non_nmethods_UsageThresholdCount{type="MemoryPool"} 0.0
# HELP java_lang_CodeHeap_non_nmethods_UsageThresholdExceeded java.lang:name=CodeHeap 'non-nmethods',type=MemoryPool,attribute=Usag
# TYPE java_lang_CodeHeap_non_nmethods_UsageThresholdExceeded untyped
java_lang_CodeHeap_non_nmethods_UsageThresholdExceeded{type="MemoryPool"} 0.0
```

3. Run Prometheus Docker container scrapping JMX metrics.

Create Docker network to put both containers on

```
Agent pid 813612
11:52:05 adrwal@olek-desktop-pc monitoring-task ±|master ✗|→ docker network create -d bridge prom-net
b0b71a54a02c1a118b6b93cc010c01162f2f99e988bf48f029a9d0e09c260d47
12:01:17 adrwal@olek-desktop-pc monitoring-task ±|master ✗|→
```

```
12:01:47 adrwal@olek-desktop-pc monitoring-task ±|master ✗|→ docker run -it --rm --name petclinic-jmx --network prom-net -p 8081:8081 petclinic-jmx-exporter
2025-01-08 11:02:02.174 | main | INFO | io.prometheus.jmx.JavaAgent | Starting ...
2025-01-08 11:02:02.380 | main | INFO | io.prometheus.jmx.JavaAgent | HTTP enabled [true]
2025-01-08 11:02:02.380 | main | INFO | io.prometheus.jmx.JavaAgent | HTTP host:port [0.0.0.0:8081]
2025-01-08 11:02:02.381 | main | INFO | io.prometheus.jmx.JavaAgent | OpenTelemetry enabled [false]
2025-01-08 11:02:02.407 | main | INFO | io.prometheus.jmx.JavaAgent | Running ...
```

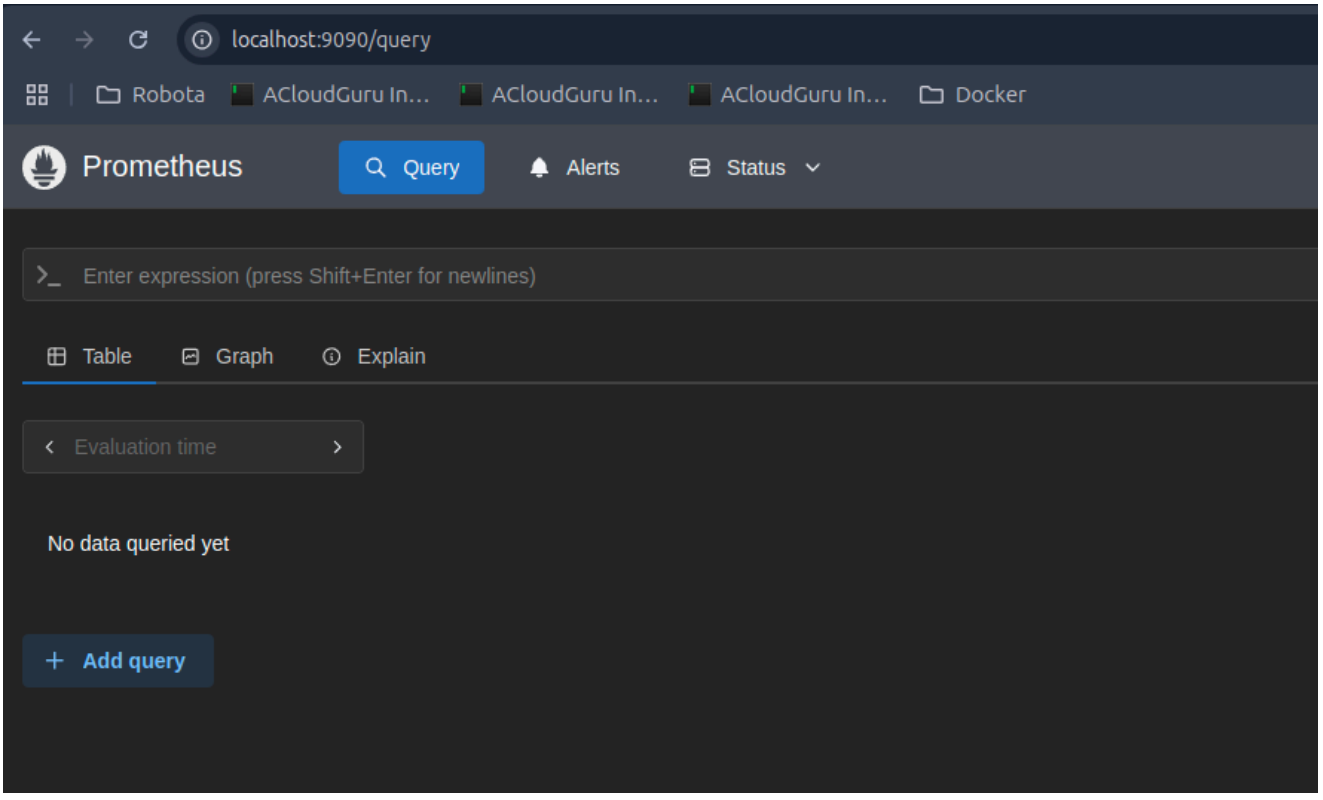
```
12:17:09 adrwal@olek-desktop-pc monitoring-task ±|master ✗|→ docker run     -p 9090:9090     -v ./prometheus.yml:/etc/prometheus/prometheus.yml     -v prometheus-data:/prometheus --network prom-net --name prom   --rm  prom/prometheus
time=2025-01-08T11:17:17.761Z level=INFO source=main.go:636 msg="No time or size retention was set so using the default time retention" duration=15d
```

Used configuration file for Prometheus:

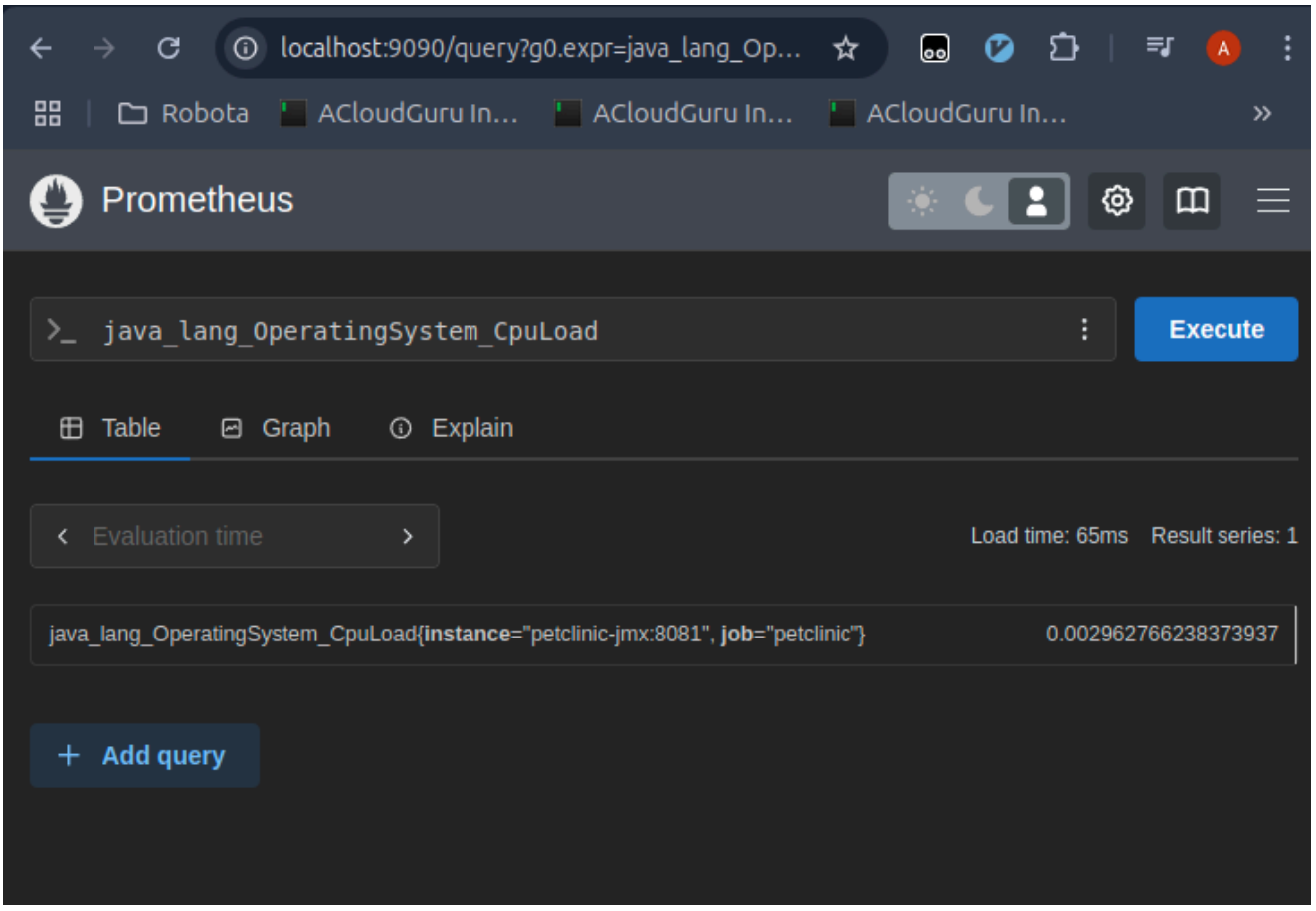
prometheus.yml > [ ]scrape_configs > {}0 > [ ]static_configs > {}0

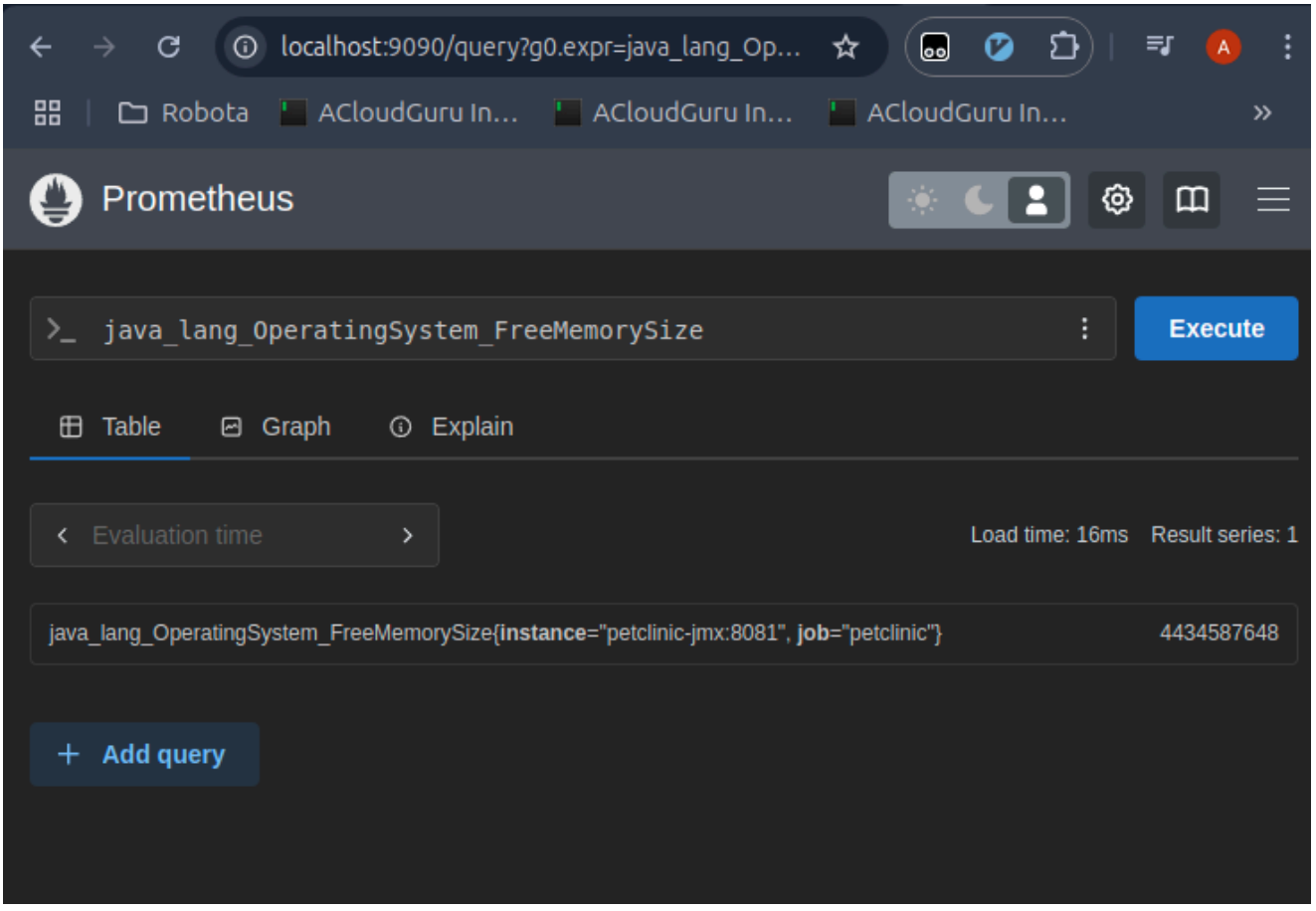
prometheus.json - Prometheus configuration file (prometheus.jso

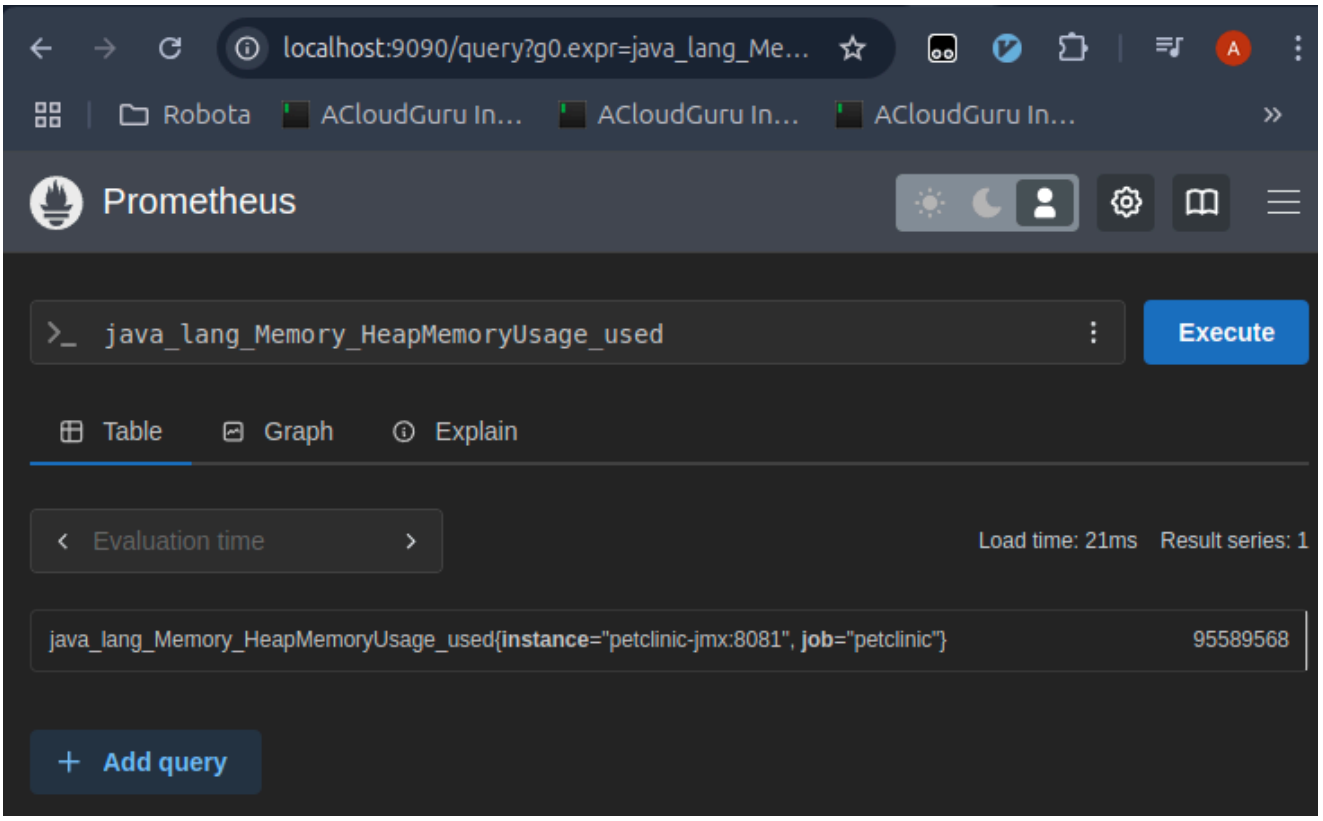
```
scrape_configs:
  - job_name: 'petclinic'
    static_configs:
      - targets: ['petclinic-jmx:8081']
```

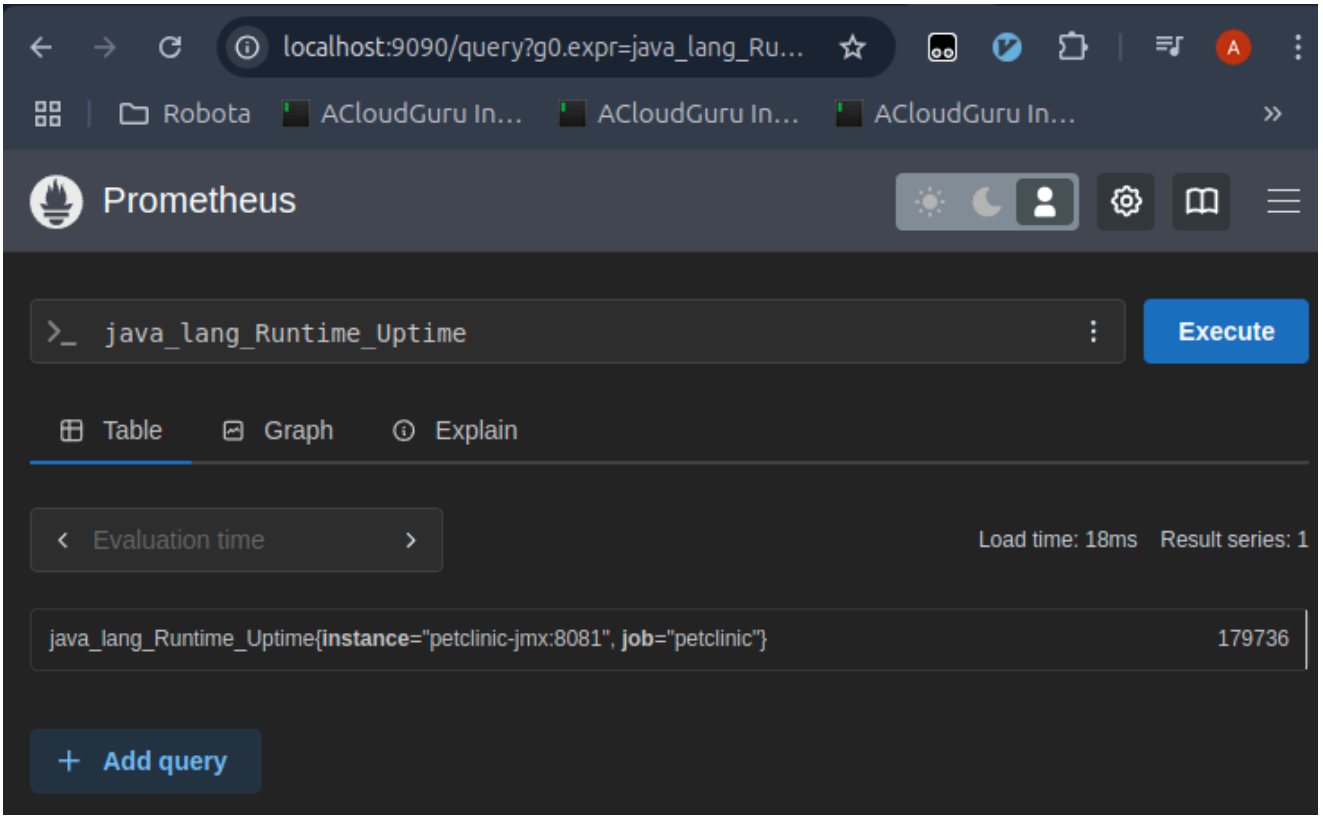


4. Validate that Prometheus gathers data

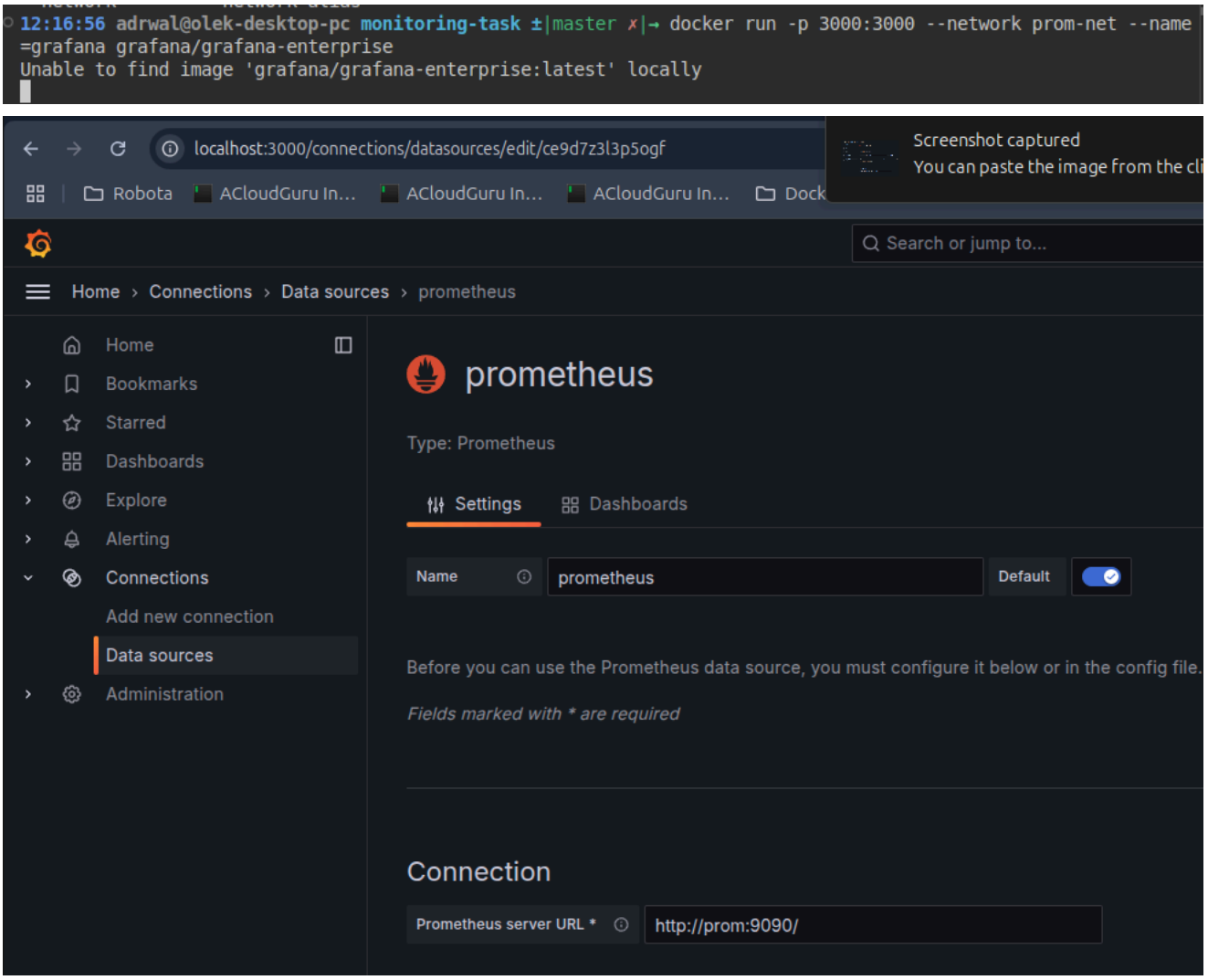

localhost:9090/query?g0.expr=java_lang_Op...
Robota
ACloudGuru In...
ACloudGuru In...
ACloudGuru In...
Prometheus
java_lang_OperatingSystem_CpuLoad
Execute
Table
Graph
Explain
Evaluation time
Load time: 65ms
Result series: 1
java_lang_OperatingSystem_CpuLoad{instance="petclinic-jmx:8081", job="petclinic"}
0.002962766238373937
Add query


localhost:9090/query?g0.expr=java_lang_Op...
Robota
ACloudGuru In...
ACloudGuru In...
ACloudGuru In...
Prometheus
java_lang_OperatingSystem_FreeMemorySize
Execute
Table
Graph
Explain
Evaluation time
Load time: 16ms
Result series: 1
java_lang_OperatingSystem_FreeMemorySize{instance="petclinic-jmx:8081", job="petclinic"}
4434587648
Add query

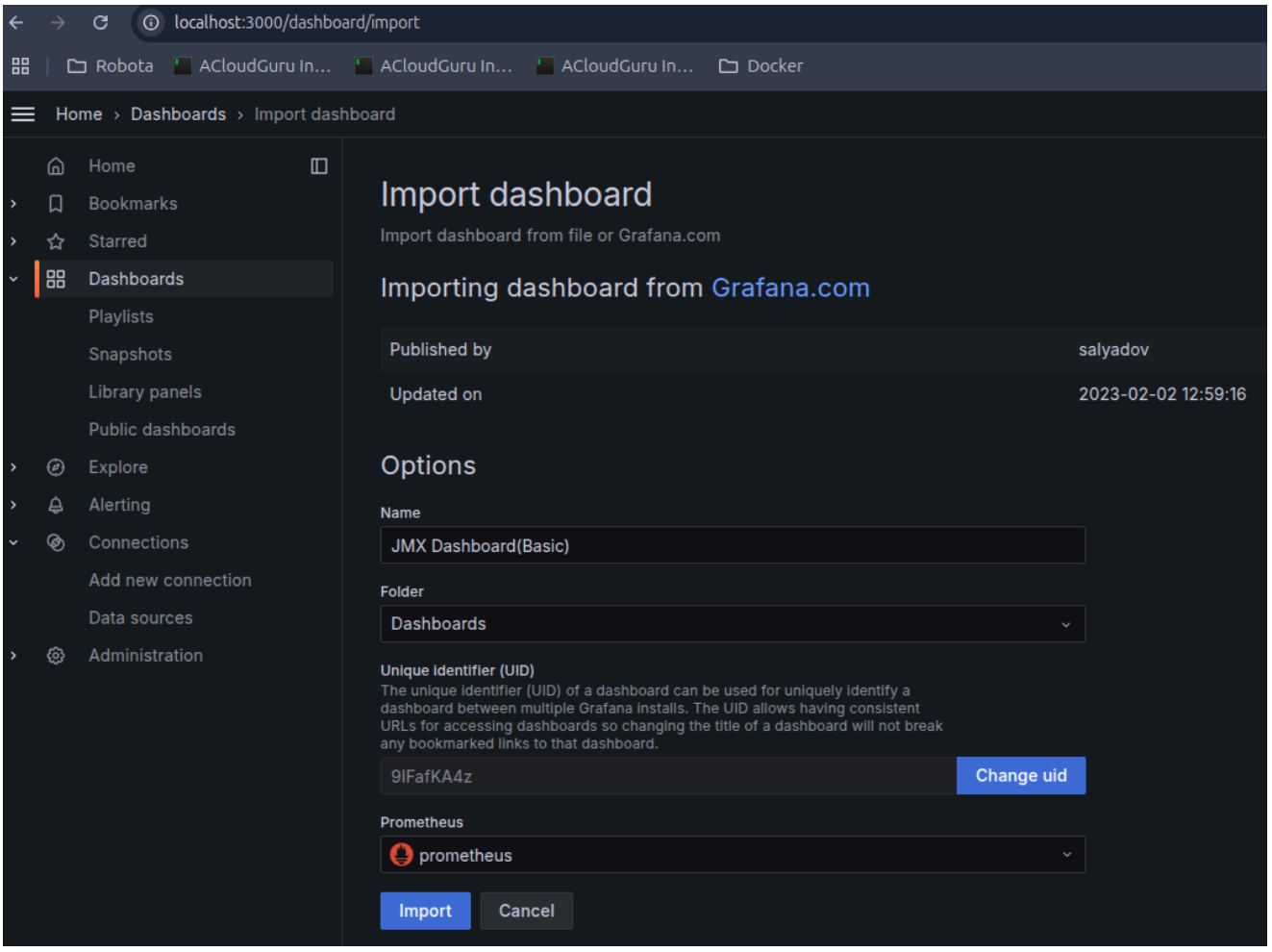


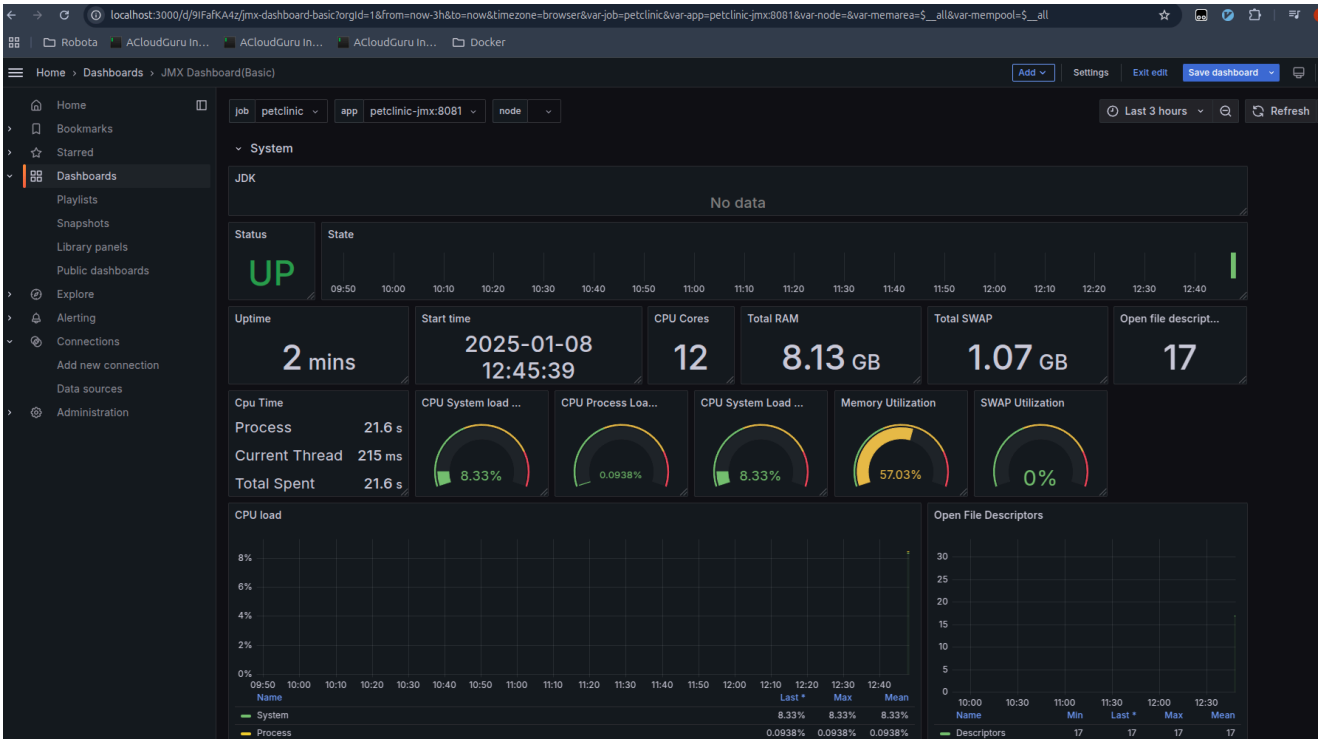


6. Run Grafana container, log in, explore browser and
a. Create Prometheus data source in the user interface.
b. Import dashboard with its id - 10519



Dashboard with ID 10519 seemed to be deprecated and most of the queries for the metrics weren't working due to using incorrect names, so I used newer dasboard for JMX (2023 vs 2017).





7. OPTIONAL: Automate the creation of data_source and dashboard resources

To automate Grafana deployment I created terraform configuration file to deploy Grafana in Docker container and configure data source with dashboard. Dashboard config file is stored locally.

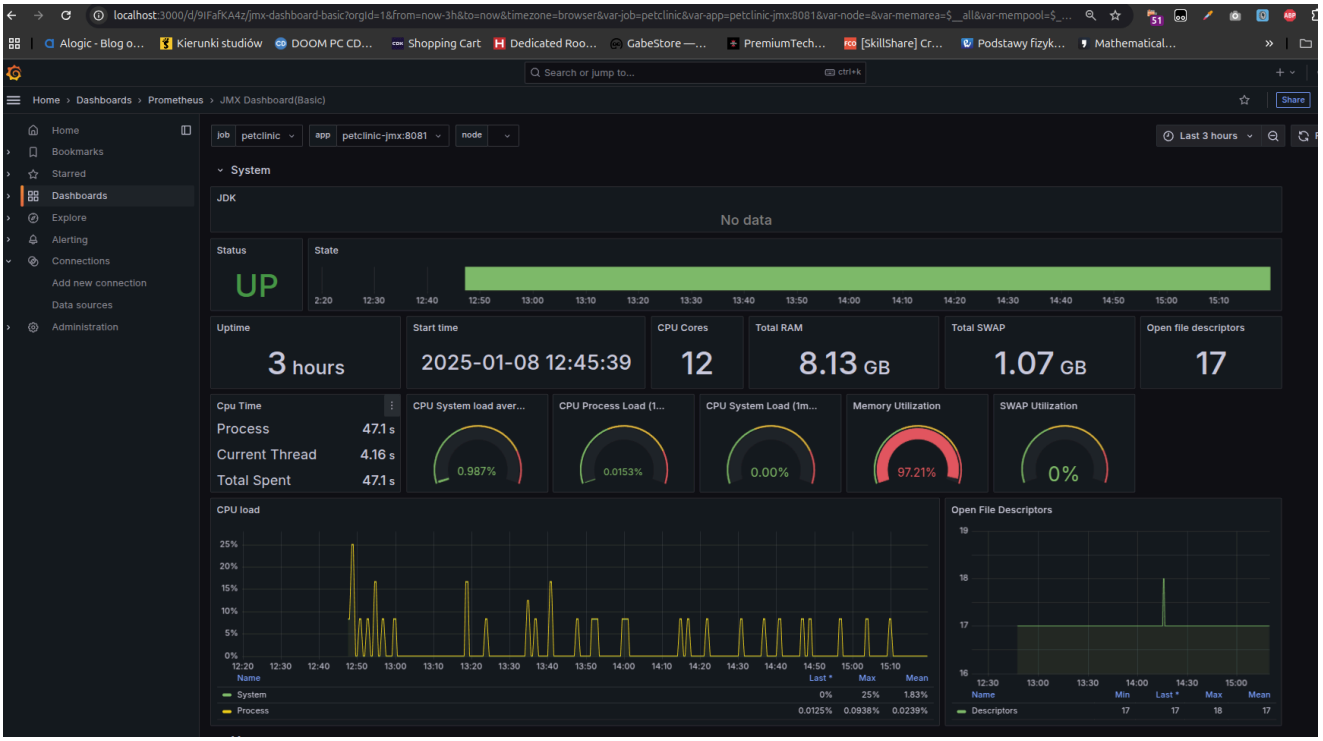
```
# main.tf

terraform {
  required_version = ">= 1.3.0"
  required_providers {
    docker = {
      source  = "kreuzwerker/docker"
      version = "3.0.2"
    }
    grafana = {
      source  = "grafana/grafana"
      version = "3.15.3"
    }
  }
}

provider "docker" {
}

provider "grafana" {
  url  = var.grafana_url
  auth = "${var.grafana_login}:${var.grafana_password}"
}


resource "docker_image" "grafana_image" {
  name         = "grafana/grafana-oss:latest"
  keep_locally = false
}

resource "docker_container" "grafana_container" {
  name         = "grafana"
  image        = docker_image.grafana_image.name
  network_mode = "host"
  ports {
    internal = 3000
    external = 3000
  }

  env = [
    "GF_SECURITY_ADMIN_USER=${var.grafana_login}",
    "GF_SECURITY_ADMIN_PASSWORD=${var.grafana_password}",
  ]
}

resource "grafana_data_source" "prometheus_ds" {
  depends_on = [
    docker_container.grafana_container
  ]
```

```
  uid        = "prometheus"
  name       = "Prometheus"
  type       = "prometheus"
  url        = var.prometheus_url
  is_default = true
}

resource "grafana_folder" "prometheus_folder" {
  title      = "Prometheus"
  depends_on = [docker_container.grafana_container,
grafana_data_source.prometheus_ds]
}

resource "grafana_dashboard" "example_dashboard" {
  depends_on = [
    grafana_data_source.prometheus_ds
  ]

  folder      = grafana_folder.prometheus_folder.id
  config_json = file("${path.module}/dashboard.json")
}
```

```
# variables.tf

variable "grafana_login" {
  description = "Grafana login"
  default     = "admin"
  sensitive   = true
}

variable "grafana_password" {
  description = "Grafana password"
  default     = "admin"
  sensitive   = true
}

variable "grafana_url" {
  description = "Grafana URL"
  default     = "http://localhost:3000"
}

variable "prometheus_url" {
  description = "Prometheus URL"
  default     = "http://localhost:9090"
}
```

```
⊗ 03:31:04 adrwal@olek-desktop-pc tf ±|master ✗|→ sudo terraform apply
```

```
d8b9067d68a675a924175]
grafana_data_source.prometheus_ds: Creating...
grafana_data_source.prometheus_ds: Creation complete after 7s [id=1:prometheus]
grafana_folder.prometheus_folder: Creating...
grafana_folder.prometheus_folder: Creation complete after 0s [id=1:be9dnx5i7glj4b]
grafana_dashboard.example_dashboard: Creating...
grafana_dashboard.example_dashboard: Creation complete after 0s [id=1:9IFafKA4z]

Apply complete! Resources: 5 added, 0 changed, 0 destroyed.
```



Logging part

1. Install Loki and Promtail

I installed Loki and Promtail as Docker containers with default configuration taken from docs. Promtail will send all files with `*log` suffix to Loki instance avalible at `localhost:3100`:

loki-config.yaml > {} query_range > {} results_cache > {} cache > {} embedded_cache > enabled

```yaml
auth_enabled: false

server:
  http_listen_port: 3100
  grpc_listen_port: 9096

common:
  instance_addr: 127.0.0.1
  path_prefix: /tmp/loki
  storage:
    filesystem:
      chunks_directory: /tmp/loki/chunks
      rules_directory: /tmp/loki/rules
  replication_factor: 1
  ring:
    kvstore:
      store: inmemory

query_range:
  results_cache:
    cache:
      embedded_cache:
        enabled: true
        max_size_mb: 100

schema_config:
  configs:
    - from: 2020-10-24
      store: tsdb
      object_store: filesystem
      schema: v13
      index:
        prefix: index_
        period: 24h

ruler:
  alertmanager_url: http://localhost:9093
```

promtail-config.yaml > [ ] scrape_configs > {} 0 > [ ] static_configs > {} 0 > {} labels

```yaml
server:
  http_listen_port: 9080
  grpc_listen_port: 0

positions:
  filename: /tmp/positions.yaml

clients:
  - url: http://localhost:3100/loki/api/v1/push

scrape_configs:
- job_name: system
  static_configs:
  - targets:
      - localhost
    labels:
      job: varlogs
      __path__: /var/log/*log
```

```
05:04:52 adrwal@olek-desktop-pc monitoring-task ±|master ✗|→ docker run --name promtail -v ./promtail-config.y
aml:/mnt/config/promtail-config.yaml --net=host -v ./:/var/log grafana/promtail:3.2.1 -config.file=/mnt/config
/promtail-config.yaml
level=info ts=2025-01-08T16:05:00.344747825Z caller=promtail.go:133 msg="Reloading configuration file" md5sum=
edc10853de8ca3588211a02437261ab3
level=info ts=2025-01-08T16:05:00.345761766Z caller=server.go:351 msg="server listening on addresses" http=[::
```

```
04:39:34 adrwal@olek-desktop-pc monitoring-task ±|master ✗|→ docker run --name loki -v ./loki-config.yaml:/mnt
/config/loki-config.yaml -p 3100:3100 grafana/loki:3.2.1 -config.file=/mnt/config/loki-config.yaml
Unable to find image 'grafana/loki:3.2.1' locally
3.2.1: Pulling from grafana/loki
c6b97f964990: Download complete
9aee425378d2: Already exists
4aa0ea1413d3: Already exists
7c12895b777b: Already exists
```

Now Loki is available at `localhost:3100`:

localhost:3100/metrics

Robota | ACloudGuru In... | ACloudGuru In... | ACloudGuru In... | Docker

```
# HELP deprecated_flags_inuse_total The number of deprecated flags currently set.
# TYPE deprecated_flags_inuse_total counter
deprecated_flags_inuse_total 0
# HELP go_cgo_go_to_c_calls_calls_total Count of calls made from Go to C by the current process.
# TYPE go_cgo_go_to_c_calls_calls_total counter
go_cgo_go_to_c_calls_calls_total 0
# HELP go_cpu_classes_gc_mark_assist_cpu_seconds_total Estimated total CPU time goroutines spent
directly comparable to system CPU time measurements. Compare only with other /cpu/classes metric
# TYPE go_cpu_classes_gc_mark_assist_cpu_seconds_total counter
go_cpu_classes_gc_mark_assist_cpu_seconds_total 0.003265341
# HELP go_cpu_classes_gc_mark_dedicated_cpu_seconds_total Estimated total CPU time spent perform
comparable to system CPU time measurements. Compare only with other /cpu/classes metrics.
# TYPE go_cpu_classes_gc_mark_dedicated_cpu_seconds_total counter
go_cpu_classes_gc_mark_dedicated_cpu_seconds_total 0.276001187
# HELP go_cpu_classes_gc_mark_idle_cpu_seconds_total Estimated total CPU time spent performing G
```

Now what's left is to add Loki as Data source in Grafana: