Simple Monitoring for Java Applications and Database

Roma Novikov
Percona

Introduction

Click to add text

About Myself

Roma Novikov

- Percona - Director of Platform, Engineering (2+ years)
- Since 2001: Web developer -> Lead/Architect -> Manager -> CTO

Interests:

- web, highload, monitoring, and observability

The Goal of This Presentation

Show the simple way to set up monitoring for Java application with database in one monitoring system and without changing the application

Focused for:

- Ops - to show how to get a general view of the application
- Devs - get to know what you are shipping in an easy way

Presentation Matters

- Visualization is a key - no language needed
- Get everyone on the same page
- Give managers meaningful information

Monitoring / Metrics

Click to add text

What to Use?

- “Common tools”
- top
- ps aux
- SHOW PROCESSLIST

- Why I need something new?
- What about data from “yesterday”?
- What's going on now with another part of the system? How is it affected?

What to Use?

- Saas/Paas + Vendor provided + DIY open source
- Challenges selecting the tools
  - Price
  - Support
  - Different environment coverage!
    - (Remember (hybrid) Clouds!)

What to Use?

Percona's choice: Prometheus and Grafana

- **Prometheus**
  - Simple but powerful architecture and data model
  - Exposition format
  - Targets

- **Grafana**
  - Data sources (30+)
  - Panel Types (50+)
  - Dashboards (X+)

PMM

Percona Monitoring and Management

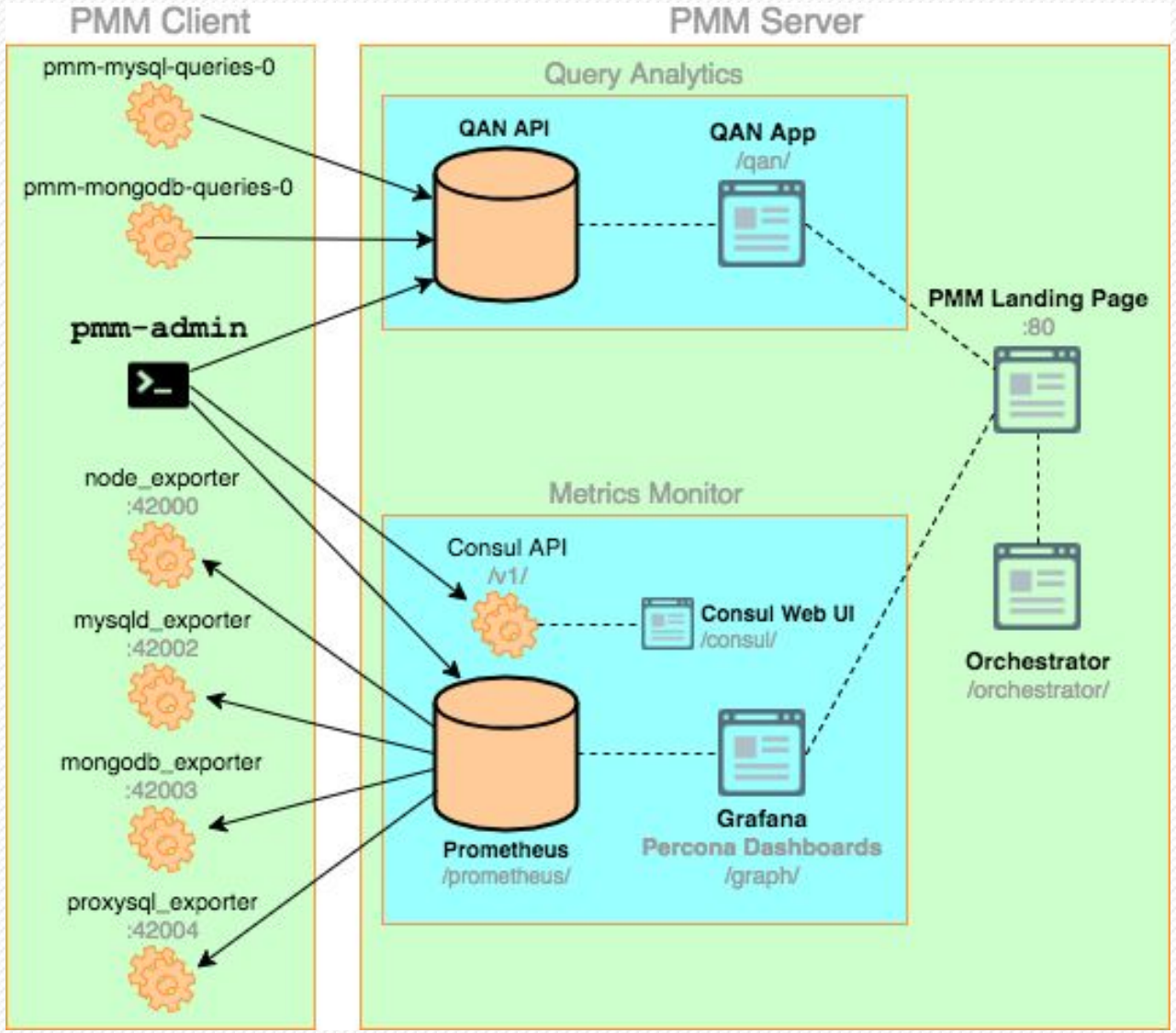
Why Did We Create PMM?

- A single tool to cover all supported databases
- Makes life easy with Prometheus and Grafana
- A common tool for internal use

Architecture

Main Components:

- Prometheus
- Grafana
- Percona dashboards
- Consul
- Query Analytics
- CLI tool
- Prometheus exporters



Distribution

- **Client**
  - Linux package
  - Binary
- **Server**
  - Docker image
  - AWS Marketplace
  - Virtual appliances - OVF

How to Extend?

- External services
- Write PR and add new technology

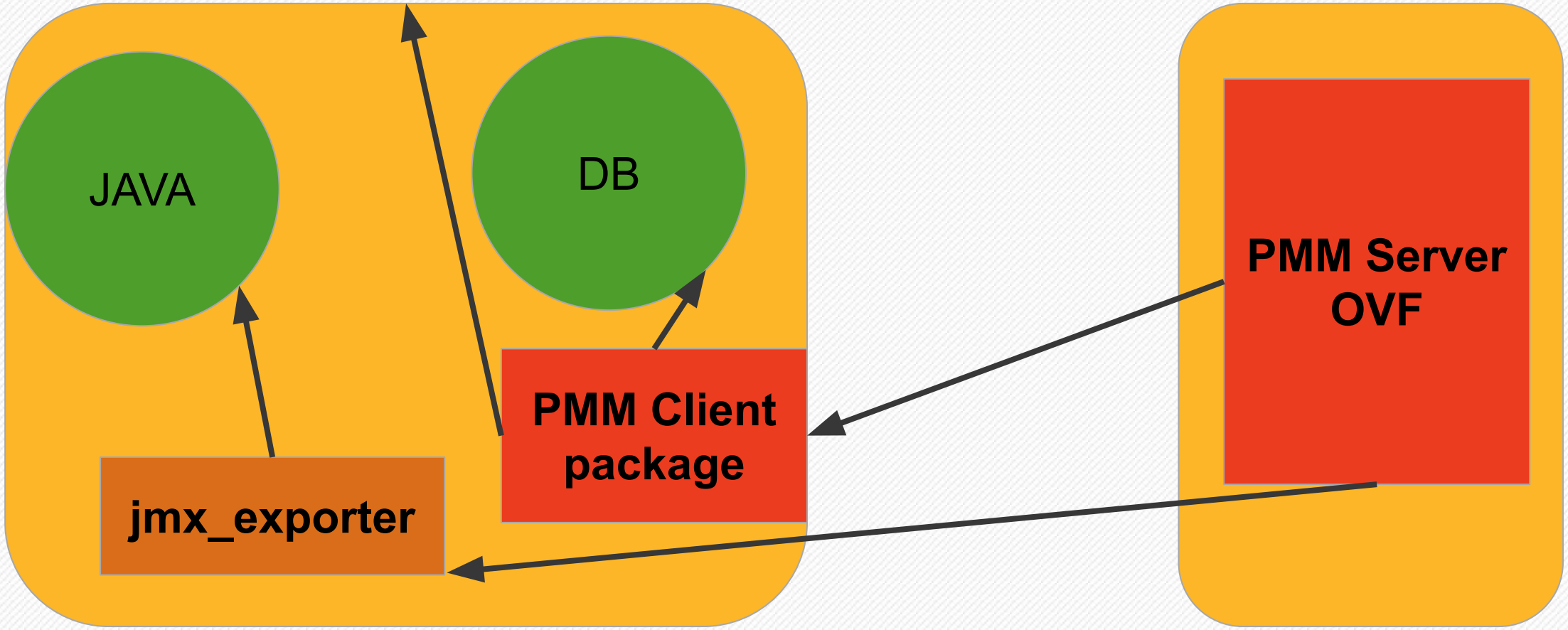
Setting up Monitoring

Click to add text

Introduction

- What will we monitor?
  - Java Application as .jar + Database (MySQL) in docker
- How?
  - Pmm = OS + Database monitoring
  - External services monitoring - JMX_exporter to add inside PMM
- Result
  - One app / Datasource (PMM / Prometheus) with data about OS, DB, JVM
  - Simple dashboard to see all at once

Big Picture



PMM Server

1. Setup and run PMM Server - https://www.percona.com/downloads/pmm/

PERCONA

Services Products Solutions Resources More Blog Communit

Subscribe to the
Percona Newsletter
Subscribe Now!

Buy Support Now!

NOTE: Percona Monitoring and Management (PMM) employs a client/server model. You must download and install both the c
and server applications. The directions for doing this are in the documentation.

Version:

Percona Monitoring and Management 1.17.1

Software:

Server - Virtual Appliance OVF

pmm-server-1.17.1.md5sum	56 bytes
pmm-server-1.17.1.ova	806.2 MB

Client - Description

- Java application as .jar file + DB (MySQL) in docker
- Used Vagrant for simplification

Client - Installation

- Get jmx_exporter for Prometheus
  - Jmx_exporter: https://github.com/prometheus/jmx_exporter
- Download **`https://repo1.maven.org/maven2/io/prometheus/jmx/jmx_prometheus_javaagent/0.11.0/jmx_prometheus_javaagent-0.11.0.jar`**
- Start Java APP and Java agent with JMX exporter
  - Create config file **`config.yaml`**
    - Empty file = track everything

Client - Run

- Run:
  ```
  #  java
  -javaagent:./jmx_prometheus_javaagent-0.11.0.jar=8181:
  config.yaml -jar myapp.jar
  ```

- 8181 - port for the exporter

Client - Verify Exporter

- Open http://192.168.0.105:8181/metrics

```
# HELP process_cpu_seconds_total Total user and system CPU time spent in seconds.
# TYPE process_cpu_seconds_total counter
process_cpu_seconds_total 5.96
# 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 1.557658214176E9
# HELP process_open_fds Number of open file descriptors.
# TYPE process_open_fds gauge
process_open_fds 26.0
# HELP process_max_fds Maximum number of open file descriptors.
# TYPE process_max_fds gauge
process_max_fds 1048576.0
# HELP process_virtual_memory_bytes Virtual memory size in bytes.
# TYPE process_virtual_memory_bytes gauge
process_virtual_memory_bytes 3.056689152E9
# HELP process_resident_memory_bytes Resident memory size in bytes.
# TYPE process_resident_memory_bytes gauge
process_resident_memory_bytes 6.4110592E7
# HELP jvm_gc_collection_seconds Time spent in a given JVM garbage collector in seconds.
# TYPE jvm_gc_collection_seconds summary
jvm_gc_collection_seconds_count{gc="Copy",} 4.0
jvm_gc_collection_seconds_sum{gc="Copy",} 0.115
jvm_gc_collection_seconds_count{gc="MarkSweepCompact",} 0.0
jvm_gc_collection_seconds_sum{gc="MarkSweepCompact",} 0.0
# HELP jvm_info JVM version info
# TYPE jvm_info gauge
jvm_info{version="1.8.0_191-8u191-b12-0ubuntu0.18.04.1-b12",vendor="Oracle Corporation",runtime="OpenJDK Runtime Environment",} 1.0
# HELP jmx_exporter_build_info A metric with a constant '1' value labeled with the version of the JMX exporter.
# TYPE jmx_exporter_build_info gauge
jmx_exporter_build_info{version="0.11.0",name="jmx_prometheus_javaagent",} 1.0
# HELP jvm_threads_current Current thread count of a JVM
# TYPE jvm_threads_current gauge
```

Client - Install pmm-client

- Configuring Percona Repositories with percona-release
  https://www.percona.com/doc/percona-repo-config/percona-release.html
- install pmm-client
  `sudo apt-get install pmm-client`
- Configure Client
  `sudo pmm-admin config --server=192.168.0.104 --server-insecure-ssl --server-password=admin --server-user=admin`

```
OK, PMM server is alive.

PMM Server      | 192.168.0.104 (insecure SSL, password-protected)
Client Name     | vagrant
Client Address  | 192.168.0.105
```

Client - Configure Monitoring

- Add MySQL monitoring

```
sudo pmm-admin add mysql
[linux:metrics] OK, now monitoring this system.
[mysql:metrics] OK, now monitoring MySQL metrics using DSN
root:***@tcp(localhost:3306)
[mysql:queries] OK, now monitoring MySQL queries from perfschema using
DSN root:***@tcp(localhost:3306)
```

- Add External service for monitoring

```
sudo pmm-admin add external:service JMX
--service-port=8181
External service added.
```

Client - Verify Installation

- Verification command
  `sudo pmm-admin list`
  `pmm-admin 1.17.1`

```
PMM Server      | 192.168.0.104 (insecure SSL, password-protected)
Client Name     | vagrant
Client Address  | 192.168.0.105
Service Manager | linux-systemd

...
```

Client - Verify Installation

```
…
-------------- -------- ----------- --------
-----------------------------
-----------------------------------------------
SERVICE TYPE   NAME     LOCAL PORT  RUNNING  DATA SOURCE
OPTIONS
-------------- -------- ----------- --------
-----------------------------
-----------------------------------------------
mysql:queries  vagrant  -           YES
root:***@tcp(localhost:3306)  query_source=perfschema,
query_examples=true
linux:metrics  vagrant  42000       YES      -
mysql:metrics  vagrant  42002       YES
root:***@tcp(localhost:3306)
..
```

Client - Verify Installation

```
...
Job name  Scrape interval  Scrape timeout  Metrics path  Scheme  Target
Labels                Health
JMX       1m0s             10s             /metrics      http
192.168.0.105:8181  instance="vagrant"  UP
```

Visualisation

Click to add text

MySQL in Grafana

MySQL Overview

Last 1 hour Refresh every 1m

Interval auto Host vagrant PMM Annotations

Query Analytics OS MySQL MongoDB PostgreSQL HA Cloud Insight PMM

MySQL Uptime
16.8 hours

Current QPS
8.31

InnoDB Buffer Pool Size
128 MiB

Buffer Pool Size of Total RAM
3%

Connections

MySQL Connections

200 150 100 50 0

17:08 17:09 17:10 17:11 17:12

	min	max	avg
Max Connections	151	151	151
Max Used Connections	15	15	15
Connections	13	14	14

MySQL Client Thread Activity

Threads

15 10 5 0

17:08 17:09 17:10 17:11 17:12

	min	max	avg
Peak Threads Connected	13.00	14.00	13.89
Peak Threads Running	1.00	3.00	1.07
Avg Threads Running	1.00	2.00	1.04

Table Locks

MySQL Questions

25 20 15 10

MySQL Thread Cache

10.0 7.5 5.0 2.5

OS in Grafana

System Overview

Last 1 hour Refresh every 1m

Data for vagrant with 20s resolution

System Uptime	Virtual CPUs	Load Average	RAM	Memory Available	Virtual Memory	Disk Space	Min Space Available
14.5 hours	1	0.03	3.85 GiB	12%	4.81 GiB	61.80 GiB	30%

CPU

CPU Usage

	min	max	avg
user	1.11%	68.10%	8.49%
system	0.42%	8.86%	1.12%
softirq	0%	2.48%	0.12%
iowait	0%	4.21%	0.08%
nice	0%	0.05%	0.00%

CPU Saturation and Max Core Usage

	min	max	avg
Normalized CPU Load	0.20	6.45	0.99
Max CPU Core Utilization (right-y)	1.57%	74.15%	9.73%

Interrupts and Context Switches

Processes

Java in Grafana

Advanced Data Exploration

Last 1 hour Refresh every 1m

metric java Interval auto Host All PMM Annotations

Insight PMM

java_lang_ClassLoading_LoadedClassCount
java_lang_ClassLoading_TotalLoadedClassCount
java_lang_ClassLoading_UnloadedClassCount
java_lang_ClassLoading_Verbose
java_lang_Compilation_CompilationTimeMonitoringSupported
java_lang_Compilation_TotalCompilationTime
java_lang_GarbageCollector_CollectionCount
java_lang_GarbageCollector_CollectionTime
java_lang_GarbageCollector_LastGcInfo_GcThreadCount
java_lang_GarbageCollector_LastGcInfo_duration
java_lang_GarbageCollector_LastGcInfo_endTime
java_lang_GarbageCollector_LastGcInfo_id
java_lang_GarbageCollector_LastGcInfo_memoryUsageAfterGc_committed
java_lang_GarbageCollector_LastGcInfo_memoryUsageAfterGc_init

Metric Resolution

1.00 min

View Actual Metric Values (Gauge)

	min	max	avg	current
java	109.00	133.00	123.50	133.00
java	0	0	0	0

View Metric Rate of Change (Counter)

JVM in Grafana

Advanced Data Exploration

metric jvm | Interval auto | Host All | PMM Annotations

Quer | goDB | PostgreSQL | HA | Cloud | Insight | PMM

jvm_buffer_pool_capacity_bytes
jvm_buffer_pool_used_buffers
jvm_buffer_pool_used_bytes
jvm_classes_loaded
jvm_classes_loaded_total
jvm_classes_unloaded_total
jvm_gc_collection_seconds_count
jvm_gc_collection_seconds_sum
jvm_info
jvm_memory_bytes_committed
jvm_memory_bytes_init
jvm_memory_bytes_max
jvm_memory_bytes_used
jvm_memory_pool_bytes_committed

etric Name

ds_started_total

Metric Resolution

1.00 min

View Actual Metric Values (Gauge)

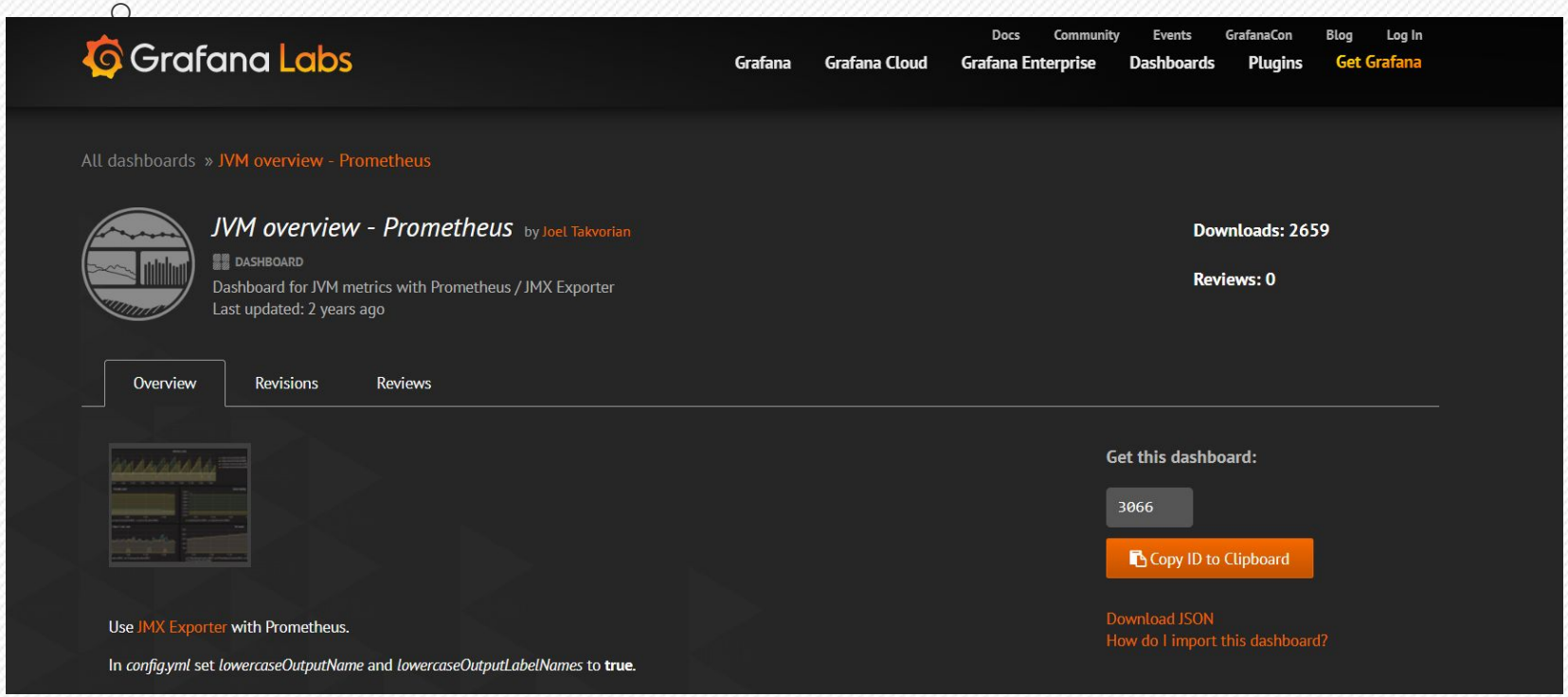
15 14 13 12 11 10 9

16:25 16:30 16:35 16:40 16:45 16:50 16:55

jvm

Get Grafana Dashboard

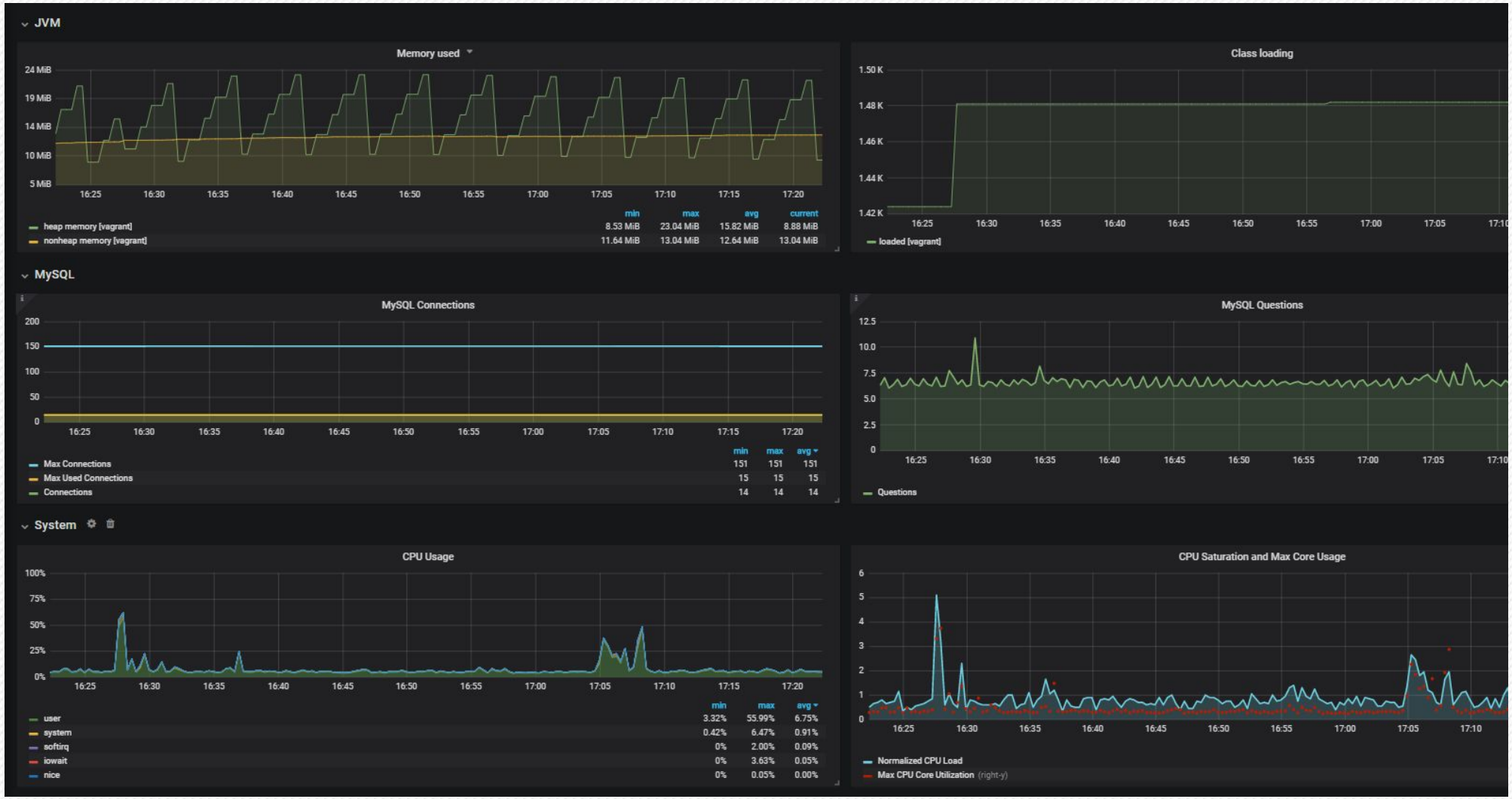
- Go to https://grafana.com/dashboards/
- Find good dashboard
- Example:
  - https://grafana.com/dashboards/3066/revisions
  -



Outcome

Click to add text

DIY Dashboard if Required



Any Questions?

Click to add text

Thank You!

Click to add text

Thank You to Our Sponsors

continuent
VividCortex
VERITAS The truth in information.
aws
EDB ENTERPRISEDB
MySQL
Altinity
PingCAP
SmartStyle
facebook
ScaleGrid
Vendita
intel OPTANE
GALERA CLUSTER
ProxySQL
aiven
Pythian love your data
MariaDB FOUNDATION
FREE SOFTWARE FOUNDATION
PlanetScale
Bloomberg Engineering
yelp
AUSTIN TECHNOLOGY COUNCIL

Rate My Session