Simple Monitoring for Java Applications and Database

Roma Novikov
Percona
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
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+)
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
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

JAVA

DB

PMM Client package

jmx_exporter

PMM Server OVF
PMM Server

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

- 8181 - port for the exporter
Client - Verify Exporter

Client - Install pmm-client

- Configuring Percona Repositories with percona-release
  

- 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
  
  
  ```bash
  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
  
  ```bash
  sudo pmm-admin add external:service JMX
  --service-port=8181
  ```

  External service added.
Client - Verify Installation

- Verification command
  ```bash
  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

...  

<table>
<thead>
<tr>
<th>SERVICE TYPE</th>
<th>NAME</th>
<th>LOCAL PORT</th>
<th>RUNNING</th>
<th>DATA SOURCE</th>
<th>OPTIONS</th>
</tr>
</thead>
<tbody>
<tr>
<td>mysql:queries</td>
<td>vagrant</td>
<td>-</td>
<td>YES</td>
<td></td>
<td>query_source=perfschema, query_examples=true</td>
</tr>
<tr>
<td>root:***@tcp(localhost:3306)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>linux:metrics</td>
<td>vagrant</td>
<td>42000</td>
<td>YES</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td>mysql:metrics</td>
<td>vagrant</td>
<td>42002</td>
<td>YES</td>
<td></td>
<td></td>
</tr>
<tr>
<td>root:***@tcp(localhost:3306)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

..
## Client - Verify Installation

...  
<table>
<thead>
<tr>
<th>Job name</th>
<th>Scrape interval</th>
<th>Scrape timeout</th>
<th>Metrics path</th>
<th>Scheme</th>
<th>Target</th>
</tr>
</thead>
<tbody>
<tr>
<td>JMX</td>
<td>1m0s</td>
<td>10s</td>
<td>/metrics</td>
<td>http</td>
<td></td>
</tr>
<tr>
<td>192.168.0.105:8181</td>
<td>instance=&quot;vagrant&quot;</td>
<td></td>
<td></td>
<td>UP</td>
<td></td>
</tr>
</tbody>
</table>
MySQL in Grafana
OS in Grafana
Java in Grafana
JVM in Grafana
Get Grafana Dashboard

- Go to [https://grafana.com/dashboards/](https://grafana.com/dashboards/)
- Find good dashboard
- Example:
  - [https://grafana.com/dashboards/3066/revisions](https://grafana.com/dashboards/3066/revisions)
Outcome
DIY Dashboard if Required
Any Questions?
Thank You!
Thank You to Our Sponsors
Rate My Session

Introducing gh-ost: triggerless, painless, trusted online schema migrations
○ 11:20 → 12:10
● Matterhorn 2

Rate & Review

Tap a star to rate

Feedback (optional)