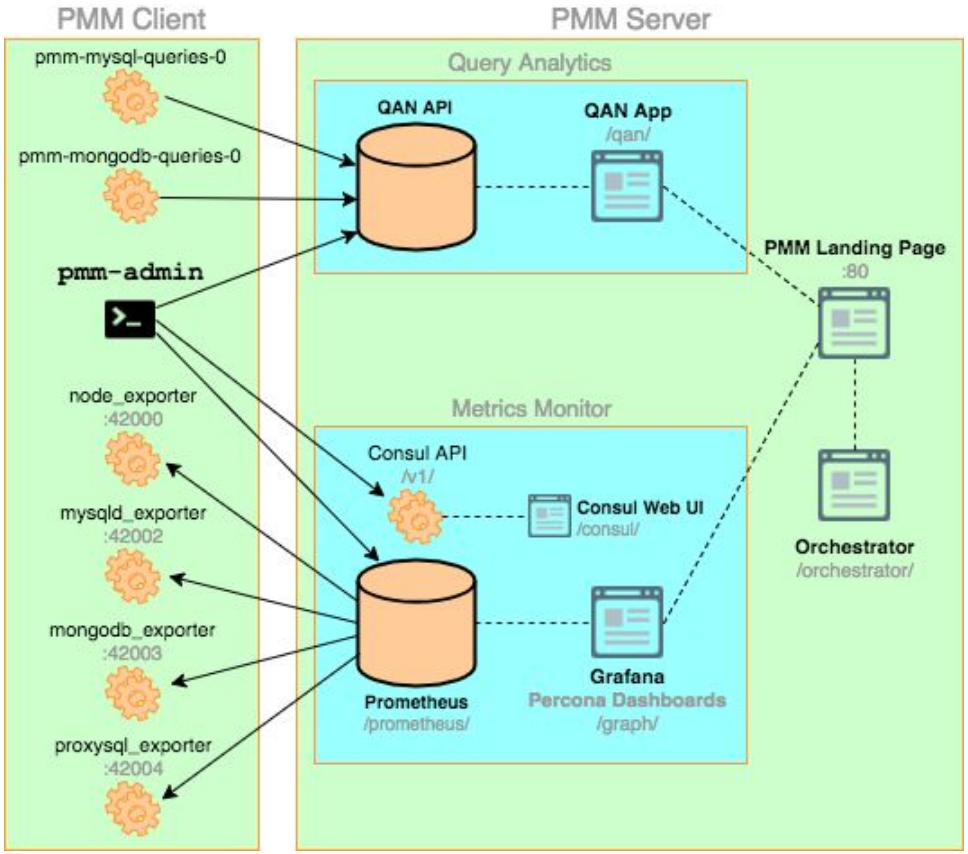

Taking PMM to the next level
Gabriel Ciciliani - Fernando Ipar
Pythian

AGENDA

- PMM in 2 minutes
- Production notes
- Beyond what's in the box
- QA

PMM in 2 minutes



https://www.percona.com/doc/percona-monitoring-and-management/architecture.html

PMM Server

Distributed as:

- A Docker image
- An Open Virtual Appliance (OVA) package
- An Amazon Machine Image (AMI)

PMM Client

Distributed as:

- An RPM, deb package
- Generic Linux binaries

Quick Start Reference

1. Create pmm-data container (lives "forever")
2. Create pmm-server container (destroyed on every upgrade)
3. Install at least one agent
4. Point agent to pmm-server
5. Deploy at least one service to monitor

https://www.percona.com/doc/percona-monitoring-and-management/deploy/server/docker.setting-up.html

https://www.percona.com/doc/percona-monitoring-and-management/deploy/index.html#installing-clients

Quick Start Reference: Step 1

```
# docker pull percona/pmm-server:latest
# docker create \
   -v /opt/prometheus/data \
   -v /opt/consul-data \
   -v /var/lib/mysql \
   -v /var/lib/grafana \
   --name pmm-data \
   percona/pmm-server:latest /bin/true
```

Quick Start Reference: Step 2

```
# docker run -d \
   -p 80:80 \
   --volumes-from pmm-data \
   --name pmm-server \
   --restart always \
   percona/pmm-server:latest
```

Quick start reference : Steps 3 through 5

```
# yum install pmm-client
# pmm-admin config --server <pmm-server-host>:<pmm-server-port>
# pmm-admin add mysql
```

Production notes

Running PMM in production

Production notes

- Store the pmm-data volumes on an dedicated volume
- Adjust total and Prometheus (if possible) memory on shared hosts
- Adjust metrics and queries retention
- Adjust metrics resolution if needed
- Enable authentication
- Backups
- Capacity planning

Production notes: pmm-data

```
# docker create \
-v /pmmdata/prometheus/data:/opt/prometheus/data \
-v /pmmdata/consul-data:/opt/consul-data \
-v /pmmdata/mysql:/var/lib/mysql \
-v /pmmdata/grafana:/var/lib/grafana \
--name pmm-data \
percona/pmm-server:latest /bin/true
```

There are gotchas:

- When using docker.io (ie. Centos < 7) bind mounts will hide any image existing files in the mount point.
- On Centos, file privileges and owners set from within the container are visible from the host OS and may overlap with existing

Production notes: adjusting memory

```
# docker run --memory=4G -e METRICS_MEMORY=209152 ...
```

- `METRICS_MEMORY` only works on Prometheus v1 (PMM < 1.13)
- Set it to 2/3 of the total memory assigned

Production notes: adjusting retention

```
# docker run -e METRICS_RETENTION=4400h -e QUERIES_RETENTION=4400h ...
```

- `METRICS_RETENTION` sets **storage.local.retention** for Prometheus 1.x (PMM < 1.13) or **storage.tsdb.retention** for Prometheus 2.x.

- `QUERIES_RETENTION` is used as argument for the **purge-qan-data** script.

Production notes: metrics resolution

```
# docker run -e METRICS_RESOLUTION=5 ...
```

- Can be used in cases where latency is above 1 second

Production notes: enabling authentication

```
# docker run -e SERVER_USER=<username> -e SERVER_PASSWORD=<password> ...
```

```
# pmm-admin config --server 127.0.0.1 --server-user <username>
--server-password <password>
```

```
OK, PMM server is alive.

PMM Server      | 127.0.0.1 (password-protected)

Client Name     | coltrane

Client Address  | 172.17.0.1
```

Production notes: backups

- pmm-server container is ephemeral by design
- Cold backup is the official method
- pmm-data should be backed up while pmm-server is not running

https://www.percona.com/doc/percona-monitoring-and-management/deploy/server/docker.backing-up.html

Production notes: capacity planning

As of v1.13:

- At least 2Gb of **memory** for a production environment (up 5 monitored target systems)
- From there, consider 8 target systems per Gb of **memory** available for pmm-server
- **CPU**: ~50 monitored systems per core (or 100K metrics/sec per CPU core)
- ~1 GB of **storage** for each monitored database node (data retention=1w)

Production notes: capacity planning

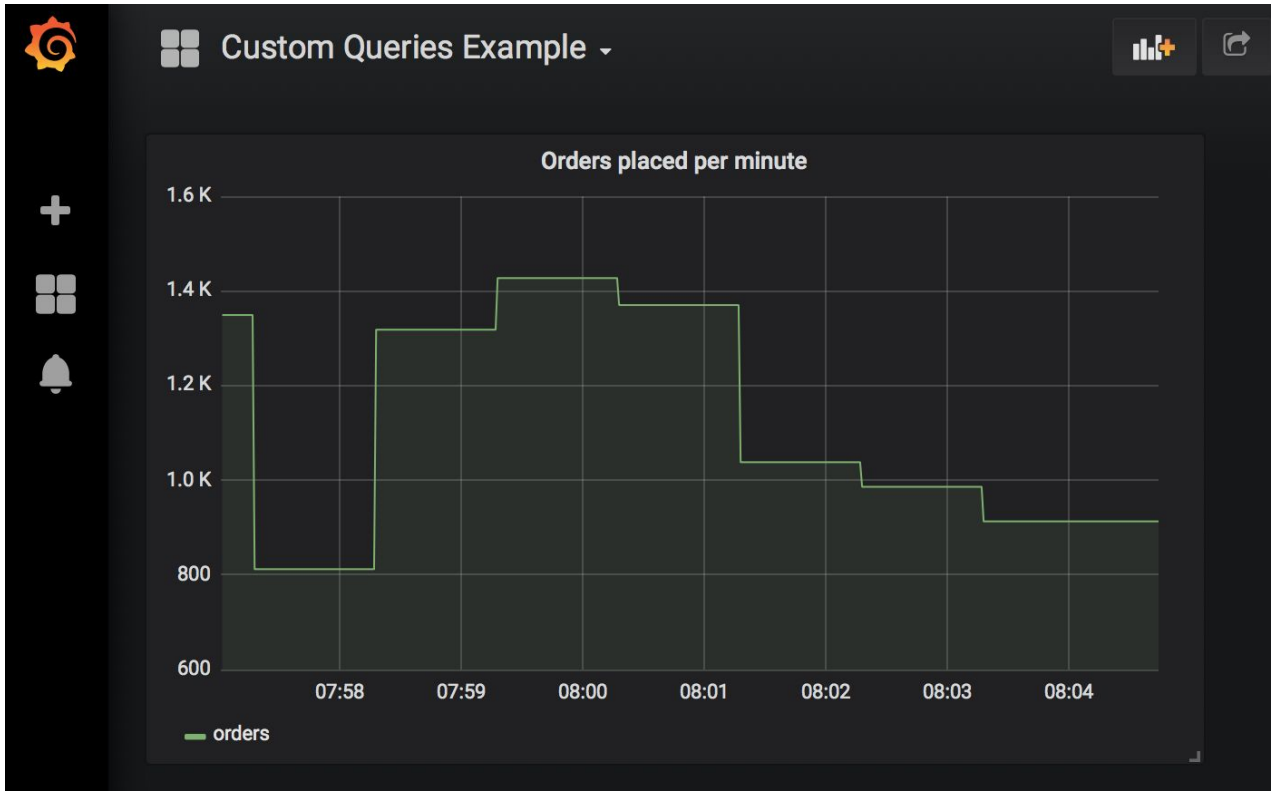
Monitoring PMM resources

- System Overview dashboard, choosing container host or container name
- If container name chosen, memory limits **not reflected** (full host memory will be displayed)
  - Visible with `docker stats pmm-server`
- Keep an eye on:
  - CPU Saturation and Max Core Usage
  - Memory Utilization
  - Disk IO Load

https://www.percona.com/blog/2018/09/28/scaling-percona-monitoring-management-pmm/

Beyond what's in the box

Beyond what's in the box: Application Instrumentation 1



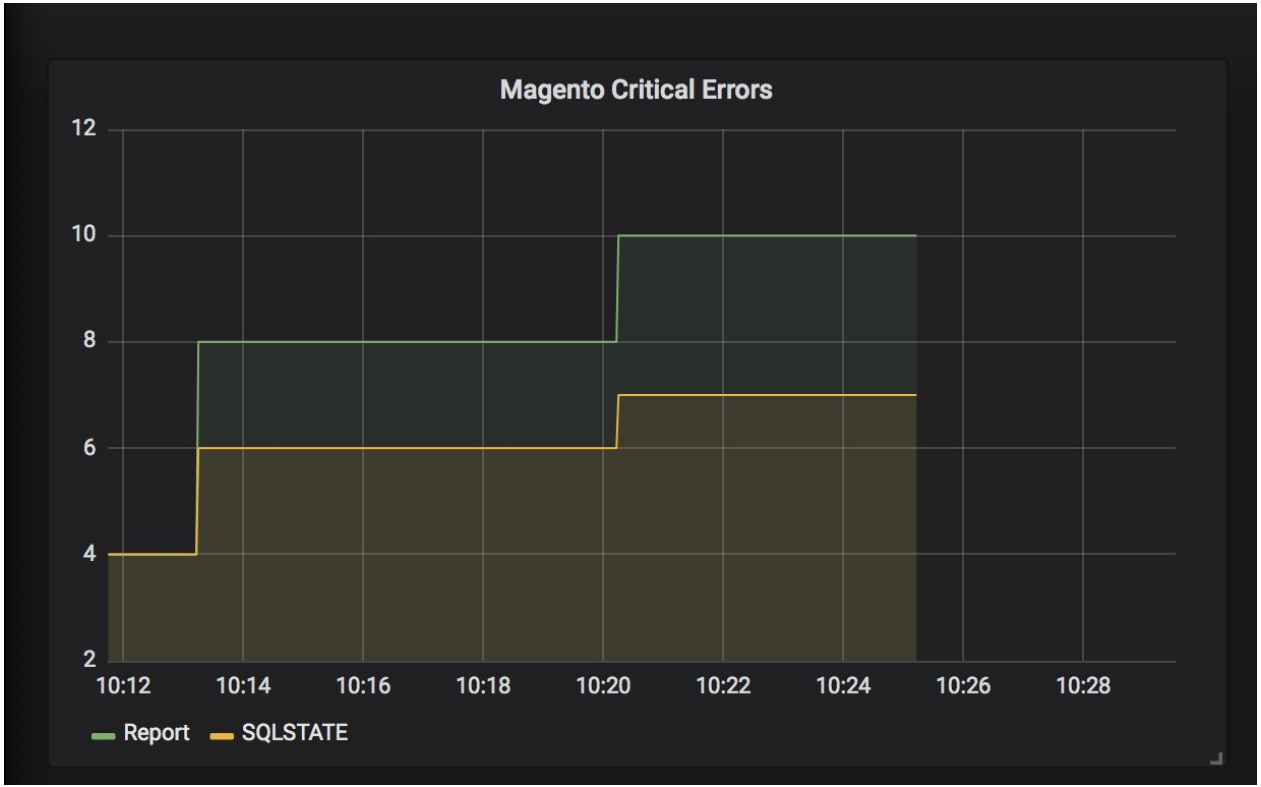
Beyond what's in the box: Application Instrumentation 1

```
$ cat /usr/local/percona/pmm-client/queries-mysqld.yml

---

mysql_orders_placed_last_minute:

   query: "select count(id) as orders from pleu18.orders where created_at between now() -
interval 1 minute and now()"

   metrics:

   - orders:

      usage: "GAUGE"

      description: "orders placed per minute"
```

Metric name

Legend

Beyond what's in the box: Application Instrumentation 2



Beyond what's in the box: Application Instrumentation 2

- Using **Grok exporter** https://github.com/fstab/grok_exporter
- Config:

```
global:
    config_version: 2
input:
    type: file
    path: ./magento_exception.log
    readall: true
```

Beyond what's in the box: Application Instrumentation 2

Config (continued):

```
grok:

    patterns_dir: ./patterns

    additional_patterns:

    - 'MAGENTO_MESSAGE .*'

    - 'NUMBER [0-9]*'
```

Beyond what's in the box: Application Instrumentation 2

Config (continued):

```
metrics:

    - type: counter

      name: magento_critical_errors

      help: Total number of main.CRITICAL entries in the magento log

      match: '.* main.CRITICAL: %{MAGENTO_MESSAGE:message}'

      labels:

          error_message: '{{.message}}'
server:

    host: localhost

    port: 9144
```

Tweak MAGENTO_MESSAGE pattern to get more info

Beyond what's in the box: Application Instrumentation 2

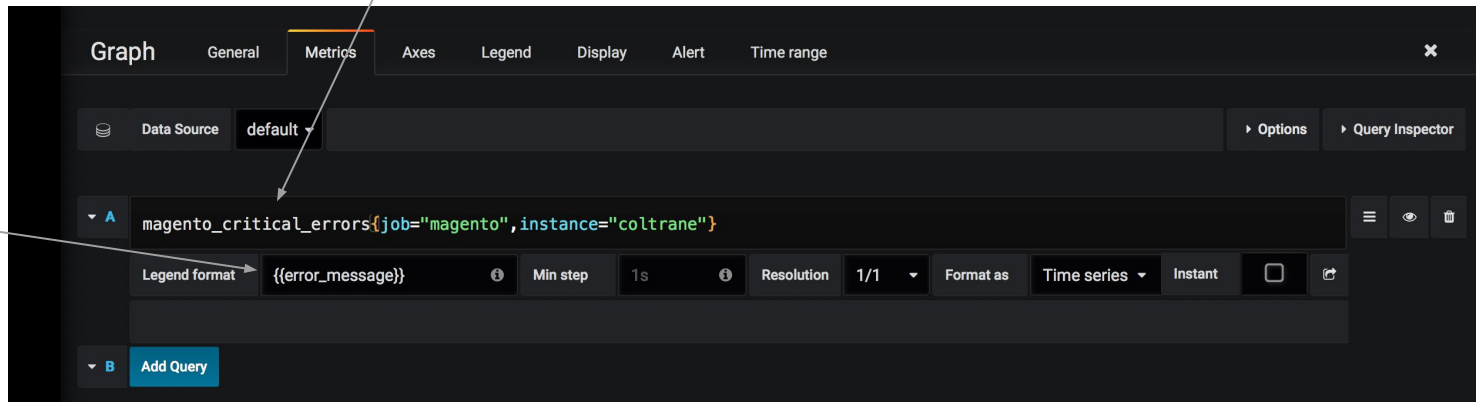
Start and add it:

```
# ./grok_exporter --config ./config.yml

# pmm-admin add external:service magento --service-port=9144 --path /metrics
```

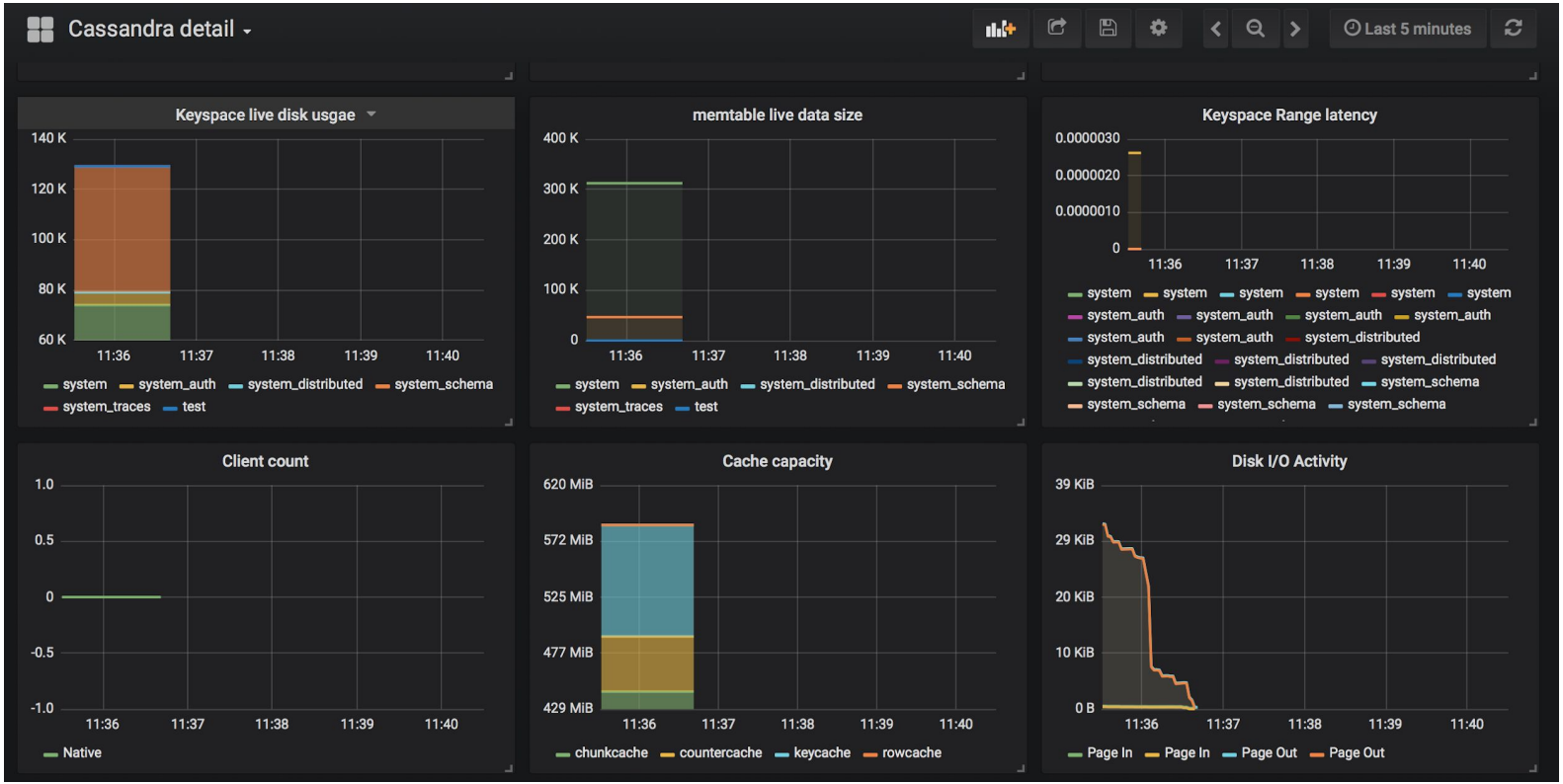
Add the graph:

name from config.yml

labels from config. yml



Beyond what's in the box: Cassandra



Beyond what's in the box: Cassandra

- Using https://github.com/nabto/cassandra-prometheus
- Add jar to cassandra-env.sh `JVM_OPTS`:
- Start Cassandra, then add target to PMM:

```
# pmm-admin add external:service cassandra --service-port=7400
```

- Import dashboard:

  https://grafana.com/dashboards/371

  https://gist.github.com/fipar/87fe346de88aa71b9f7656dc5938c23d

Beyond what's in the box: Monitoring X

- Metrics from X must be available to PMM's Grafana.
- Available data sources:
  - Prometheus (exporter for X needed)
  - QAN-API (if you want to create custom graphs from Query Analytics Data)
- `pmm-admin add external:service`.
- Create graphs or create/import dashboard

Q&A