Using Prometheus with InfluxDB for metrics storage

Roman Vynar

Senior Site Reliability Engineer, Quiq

September 26, 2017

OPEN SOURCE DATABASE CONFERENCE

PERCONA

LIVE EUROPE

DUBLIN

About Quiq

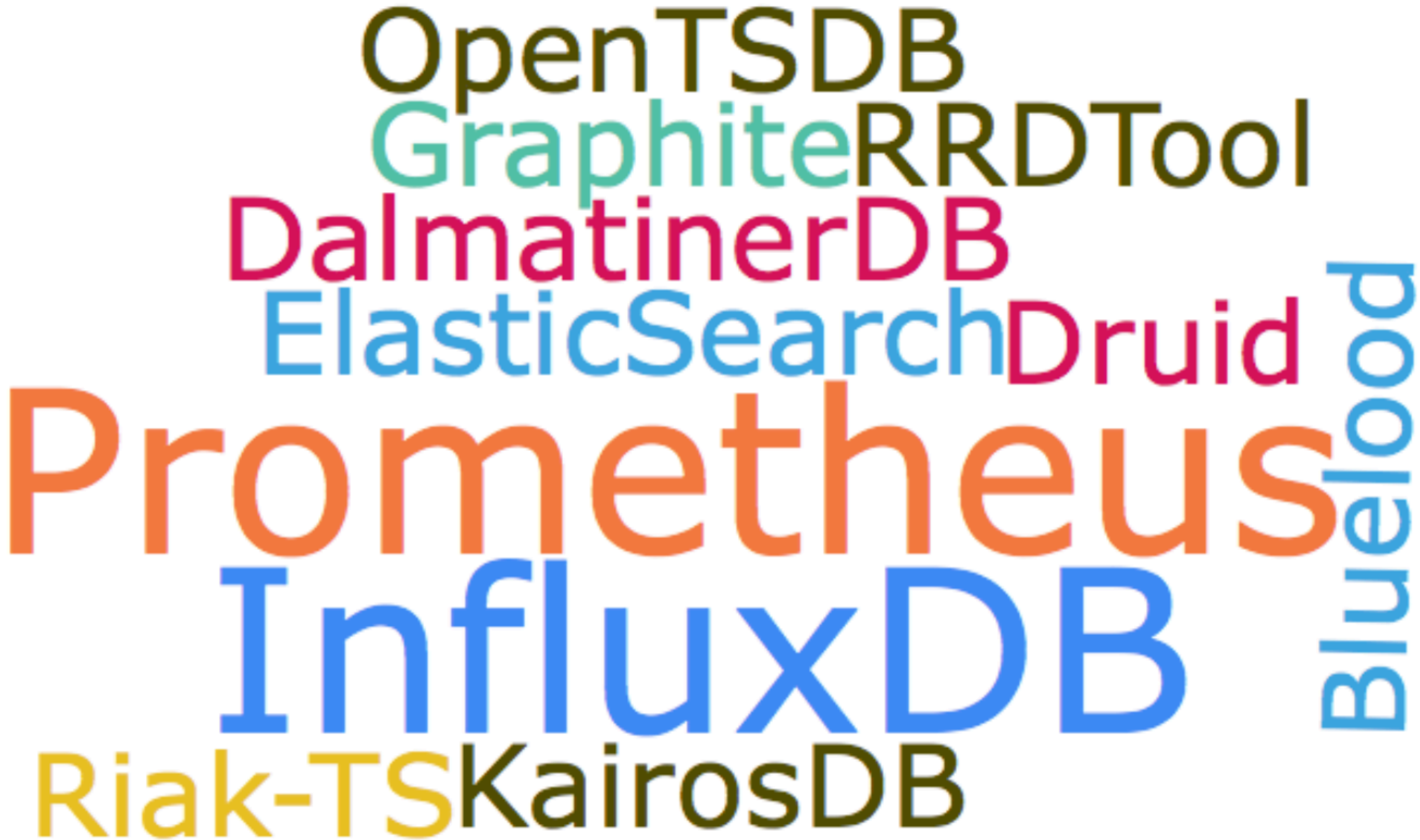
Quiq is a messaging platform for customer service.

https://goquiq.com

We monitor all our infrastructure with **1** Prometheus:
190 targets, **190K** time-series, **10K** samples/sec ingestion rate.

We store customer-related and developer metrics of all the micro-services in InfluxDB using in-house InfluxDB HA implementation.

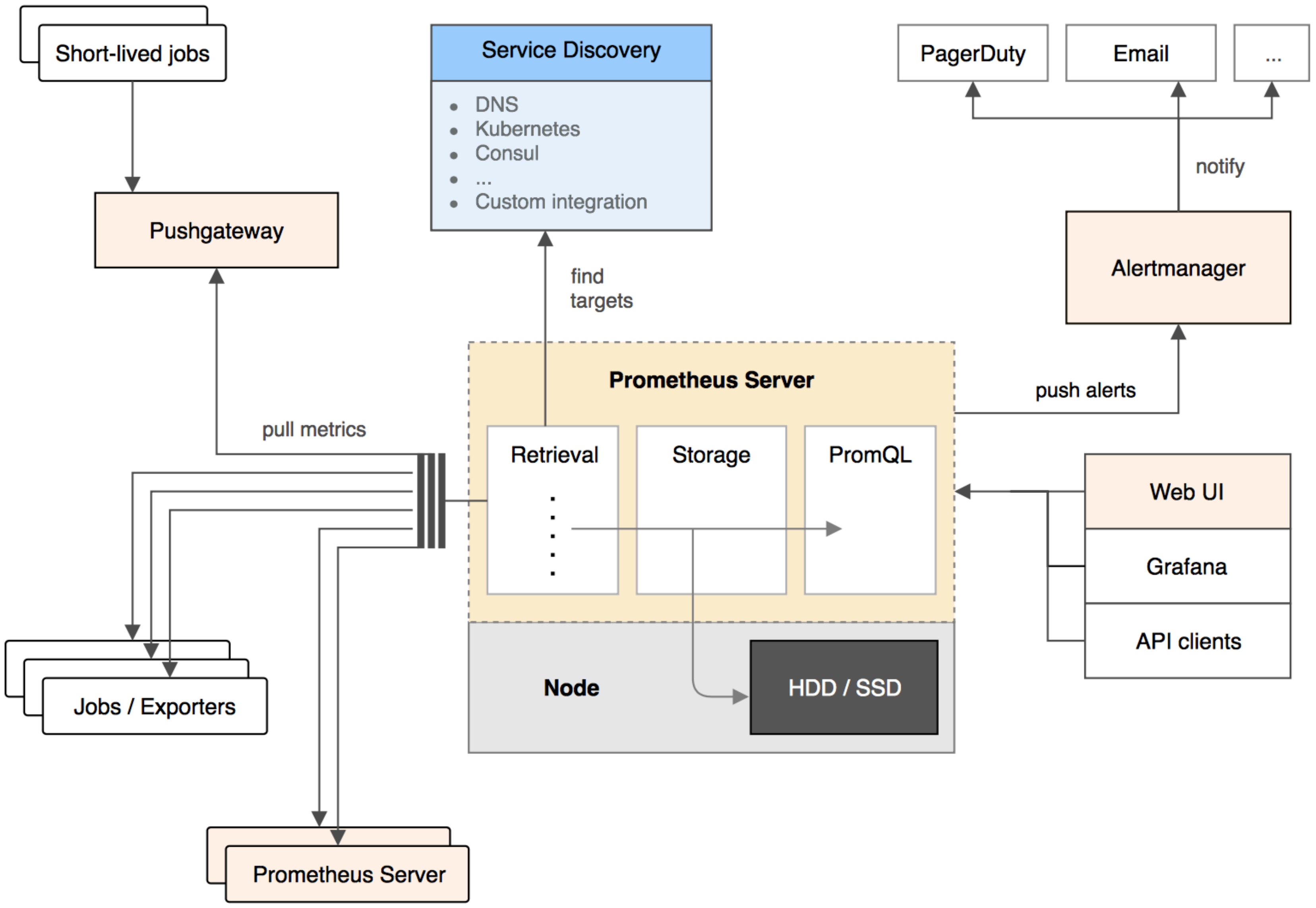
Time-series databases



Prometheus

- Prometheus is 100% open-source and community-driven
- Modern and efficient
- Multi-dimensional data model
- Collection via “pull” model
- Powerful query language and HTTP API
- Service discovery
- Alerting toolkit and integrations
- Federation of Prometheis

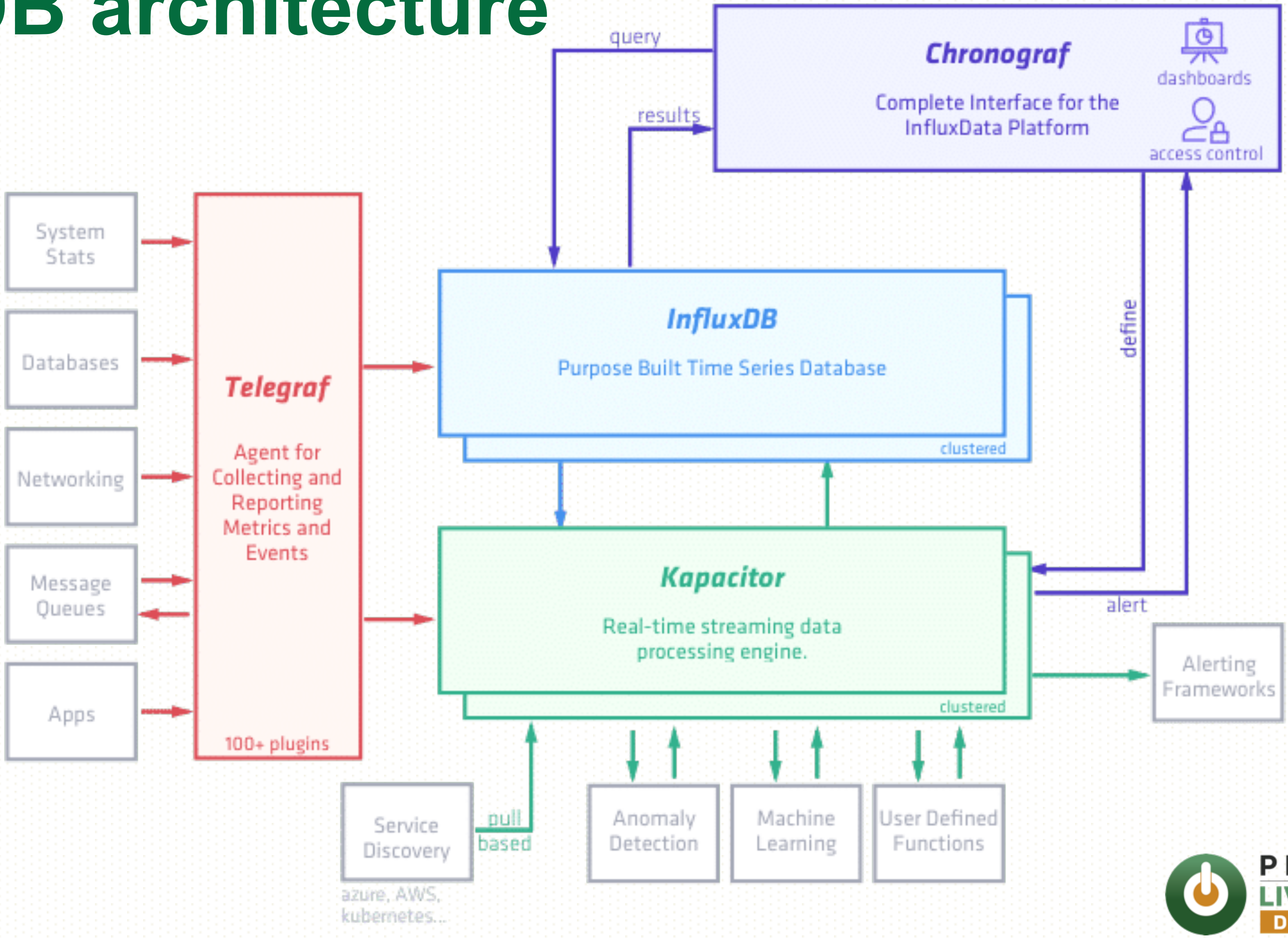
Prometheus architecture



InfluxDB

- Open-source and commercial offering
- Modern and efficient
- Multi-dimensional data model
- Collection via “push” model
- SQL and HTTP API
- A component of a full-stack platform
- Backup and restore
- Clustering (proprietary, commercial)

InfluxDB architecture



PERCONA LIVE EUROPE DUBLIN

Time-series structure

- **Prometheus**:

  metric{job="…", instance="…", label1="…", label2="…"} **float64** *timestamp (ms)*

  gauge | *counter* | *histogram* | *summary*

- **InfluxDB**:

  db.retention.measurement tag1="…",tag2=".." field1=**bool**,field2=**"string"**,field3=**int|**
  float64 *timestamp (ns)*

Prometheus 1.7.1 vs InfluxDB 1.3.5

Feature	Prometheus	InfluxDB
Metrics collection model	Pull	Push
Storage	Ephemeral	Long-lived
Data retention	A single, global	Multiple, per database
Service discovery	Built-in	N/A
Clustering	Federation	Commercial
Downsampling	Recording rules	Continuous queries
Query language	PromQL	InfluxSQL
Backup and restore	Another prom instance	Binary and raw formats
Integrations	Components, 3rd-party	TICK stack, 3rd-party

Prometheus and “pull”

- Prometheus scrapes metrics from remote exporters
- Configurable frequency of scraping
- Relabeling
- Simple protocol-buffer or text-based exposition format
- Custom on-demand metrics via textfile collector of node_exporter
- "Push" is also possible via pushgateway

Prometheus storage and retention

- A sophisticated local storage subsystem
- Chunks of constant size for the bulk sample data
- LevelDB for indexes
- Circular global retention
- Not really designed for long-term storage

Prometheus service discovery

- Service discovery out the box
- DNS
- Consul
- AWS
- GCP
- Azure
- Kubernetes
- Openstack
- Dynamic and flexible configuration

Prometheus federation

- Federation allows a Prometheus server to scrape selected time series from another Prometheus server.
- Hierarchical federation
- Cross-service federation

Prometheus recording rules

- Recording rules allow you to precompute frequently needed or expensive expressions and save their result as a new set of time series
- Can be used for downsampling

PromQL

- Prometheus provides a functional expression language that lets the user select and aggregate time series data in real time.
- Cross-metric queries
- Grouping and joins
- Functions over functions

Prometheus and backups

- No backup mechanism
- However, you can run multiple Prometheus instances to do exactly the same job to keep a standby copy.

Prometheus integrations

- Grafana
- Alertmanager
- Dropwizard, Gitlab, Docker, etc.

- InfluxDB: read, write
- OpenTSDB: write
- Chronix: write
- Graphite: write
- PostgreSQL/TimescaleDB: read, write

Prometheus vs InfluxDB

Feature	Prometheus	InfluxDB
Metrics collection model	Pull	Push
Storage	Ephemeral	Long-lived
Data retention	A single, global	Multiple, per database
Service discovery	Built-in	N/A
Clustering	Federation	Commercial
Downsampling	Recording rules	Continuous queries
Query language	PromQL	InfluxSQL
Backup and restore	Another prom instance	Binary and raw formats
Integrations	Components, 3rd-party	TICK stack, 3rd-party

InfluxDB and “push”

- Telegraph pushes samples to InfluxDB
- There are 100+ plugins for Telegraphs
- "Push" on demand

InfluxDB storage and retention

- Compressed and encoded data are organized in shards with duration
- Shards are grouped into shard groups by time and duration
- Multiple databases
- Multiple retentions per database
- Each database has its own set of WAL and TSM files

InfluxDB downsampling

- Configurable retentions per database
- Continuous queries across retentions and databases
- Flexible time grouping, resampling intervals and offsets
- Commercial clustering ensures the data is copied to X replicas

InfluxQL

- SQL-like language
- Schema exploration
- Flexible grouping by time intervals
- No joins
- No functions over functions

InfluxDB backup and restore

- Built-in backup/restore tool
- Backup/restore a specific database/retention/shard
- Backup since a specific date
- Separate backup of datastore and metastore
- HTTP API allows for a plain-text backup/restore too

InfluxDB integrations

- Kapacitor
- Chronograf
- Grafana
- Remote read/write by Prometheus

Prometheus vs InfluxDB

Feature	Prometheus	InfluxDB
Metrics collection model	Pull	Push
Storage	Ephemeral	Long-lived
Data retention	A single, global	Multiple, per database
Service discovery	Built-in	N/A
Clustering	Federation	Commercial
Downsampling	Recording rules	Continuous queries
Query language	PromQL	InfluxSQL
Backup and restore	Another prom instance	Binary and raw formats
Integrations	Components, 3rd-party	TICK stack, 3rd-party

PERCONA LIVE EUROPE DUBLIN

Prometheus + InfluxDB

Feature	Prometheus	InfluxDB
Metrics collection model	**Pull**	Push
Storage	Ephemeral	**Long-lived**
Data retention	A single, global	**Multiple, per database**
Service discovery	**Built-in**	N/A
Clustering	Federation	Commercial
Downsampling	Recording rules	**Continuous queries**
Query language	**PromQL**	SQL
Backup and restore	Another prom instance	**Binary and raw formats**
Integrations	Components, 3rd-party	TICK stack, 3rd-party

PERCONA LIVE EUROPE DUBLIN

What is better?

InfluxDB:
- For event logging.
- Commercial option offers clustering for InfluxDB, which is also better for long term data storage.
- Eventually consistent view of data between replicas.

Prometheus:
- Primarily for metrics.
- More powerful query language, alerting, and notification functionality.
- Higher availability and uptime for graphing and alerting.

PERCONA LIVE EUROPE DUBLIN

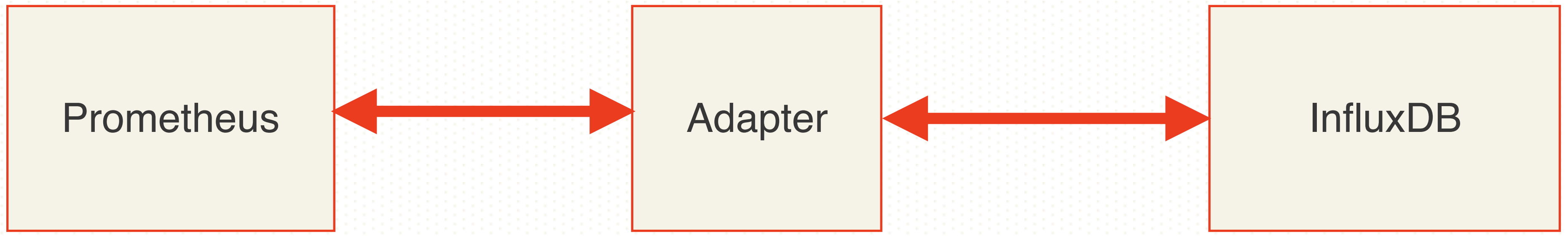
Prometheus and InfluxDB integration

Currently, there are 2 options:

1. Using remote_storage_adapter:
   https://github.com/prometheus/prometheus/tree/master/documentation/examples/remote_storage/remote_storage_adapter

2. Writing to InfluxDB directly (nightly builds of not yet released v1.4):
   https://www.influxdata.com/blog/influxdb-now-supports-prometheus-remote-read-write-natively/ (posted on Sep 14, 2017)

Prometheus and InfluxDB integration



docker-compose.yml

```
$ cat PL17-Dublin/docker-compose.yml
version: '2'

services:

  prom:
    image: prom/prometheus:v1.7.1
    command: -storage.local.path="/promdata"
    ports:
      - "9090:9090"
    volumes:
      - ./prometheus.yml:/prometheus/prometheus.yml:ro
      - ./promdata:/promdata

  influxdb:
    image: influxdb:1.3.5
    command: -config /etc/influxdb/influxdb.conf
    ports:
      - "8086:8086"
    volumes:
      - ./influxdata:/var/lib/influxdb
```

Running InfluxDB

```
docker-compose up -d influxdb
docker exec -ti pl17dublin_influxdb_1 influx
> CREATE USER "admin" WITH PASSWORD 'admin' WITH ALL PRIVILEGES;

docker exec -ti pl17dublin_influxdb_1 bash
> influx
>> auth
>> CREATE DATABASE prometheus;
>> CREATE USER "prom" with password 'prom';
>> GRANT ALL ON prometheus TO prom;
>> ALTER RETENTION POLICY "autogen" ON "prometheus" DURATION 1d
REPLICATION 1 SHARD DURATION 1d DEFAULT;
>> SHOW RETENTION POLICIES ON prometheus;
```

Running remote_storage_adapter

```
go get github.com/prometheus/prometheus/documentation/examples/
remote_storage/remote_storage_adapter
```

```
INFLUXDB_PW=prom $GOPATH/bin/remote_storage_adapter
  -influxdb-url=http://localhost:8086
  -influxdb.username=prom
  -influxdb.database=prometheus
  -influxdb.retention-policy=autogen
```

Prometheus config file

```
global:
  scrape_interval: 1s
  scrape_timeout: 1s

scrape_configs:
  - job_name: prometheus
    static_configs:
      - targets: ['localhost:9090']
        labels:
          instance: prom

remote_write:
  - url: http://docker.for.mac.localhost:9201/write
```

Running Prometheus and verification

```
docker-compose up -d prom

docker logs pl17dublin_prom_1
docker logs -f --tail 10 pl17dublin_influxdb_1

docker exec -ti pl17dublin_influxdb_1 bash
> influx
>> auth
>> USE prometheus;
>> SHOW MEASUREMENTS;
```

Downsampling with InfluxDB

```
CREATE DATABASE trending;
CREATE RETENTION POLICY "1m" ON trending DURATION 0s REPLICATION 1 SHARD DURATION 1w DEFAULT;
CREATE RETENTION POLICY "5m" ON trending DURATION 0s REPLICATION 1 SHARD DURATION 1w DEFAULT;
SHOW RETENTION POLICIES ON trending;

USE prometheus;
CREATE CONTINUOUS QUERY scrape_samples_scraped_1m ON prometheus BEGIN SELECT LAST(value) as "value"
INTO trending."1m".scrape_samples_scraped FROM scrape_samples_scraped GROUP BY time(1m) END;
CREATE CONTINUOUS QUERY scrape_samples_scraped_5m ON prometheus BEGIN SELECT LAST(value) as "value"
INTO trending."5m".scrape_samples_scraped FROM scrape_samples_scraped GROUP BY time(5m) END;
SHOW CONTINUOUS QUERIES;
USE trending;
SHOW MEASUREMENTS;
SHOW SHARDS;
SELECT * FROM trending."1m".scrape_samples_scraped;
```

PERCONA LIVE EUROPE DUBLIN

Prometheus remote read (proxy to InfluxDB)

```
$ cat PL17-Dublin/docker-compose.yml
version: '2'

services:

  promread:
    image: prom/prometheus:v1.7.1
    command: -storage.local.engine=none
    ports:
      - "9091:9090"
    volumes:
      - ./promread.yml:/prometheus/prometheus.yml:ro
```

Prometheus remote read

Prometheus configuration:

```
remote_read:
  - url: http://docker.for.mac.localhost:9201/read
```

Start Prometheus with the above config:

```
docker-compose up -d prom read
```

Questions?

Thank you!