Python + Prometheus = ?

Nikita Grishko

Flo Health Inc.

```
import prometheus
```

Monitoring?

Why Prometheus?

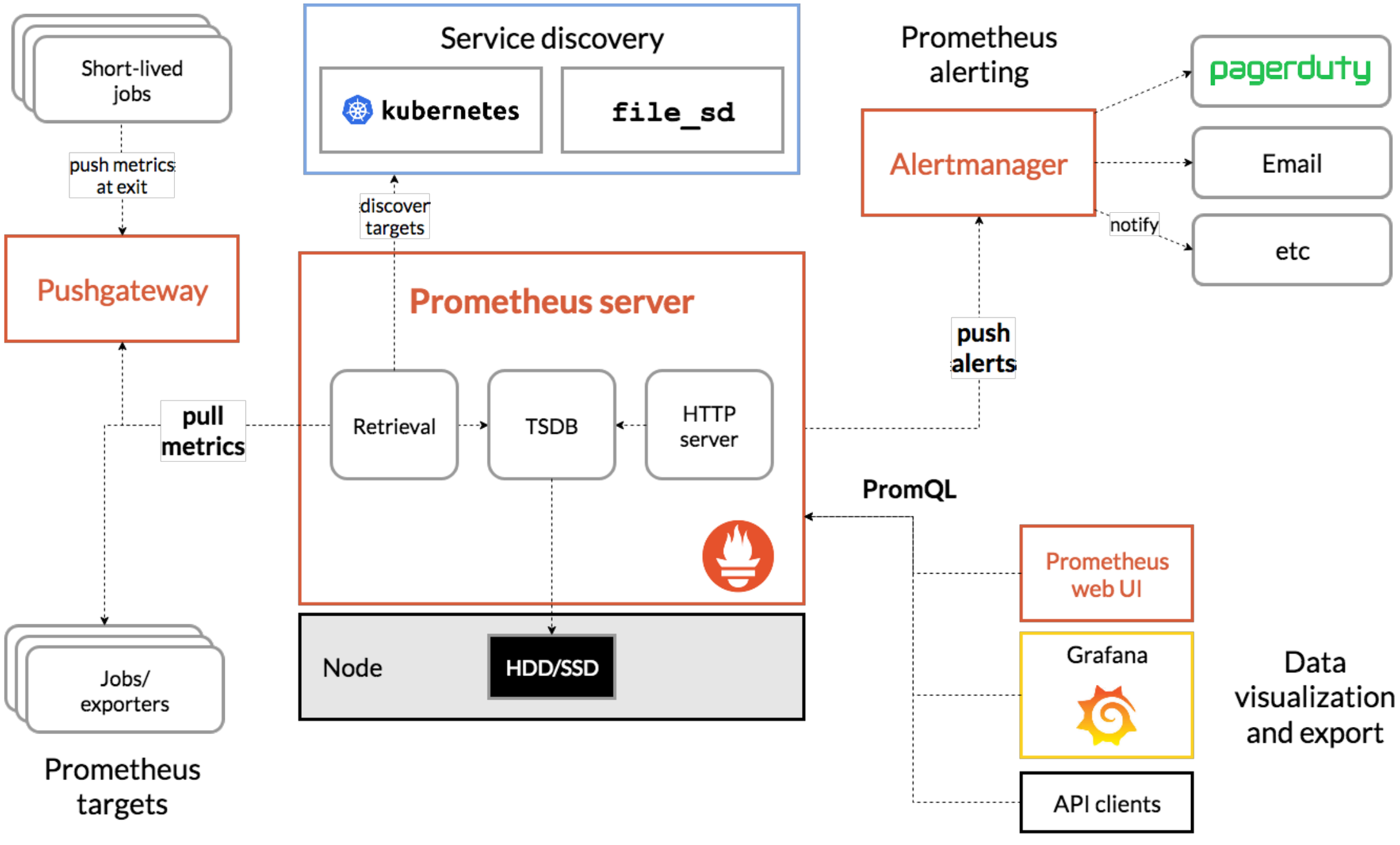
Why not XYZ?

Two days!

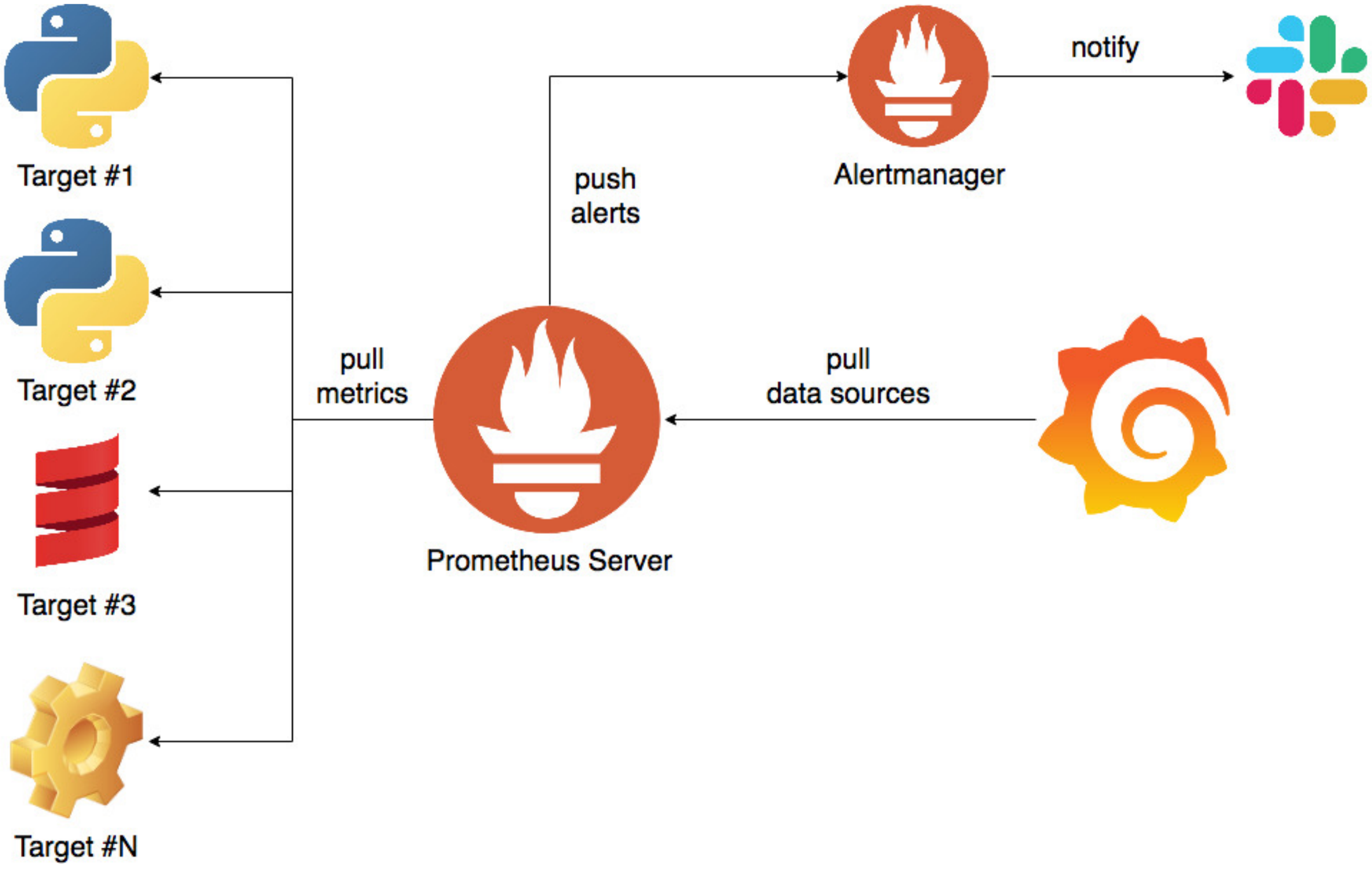
Prometheus?

Prometheus is an open-source systems monitoring and alerting toolkit

Prometheus Architecture



Prometheus Architecture



pull vs push

https://prometheus.io/docs/introduction/faq/#why-do-you-pull-rather-than-push

Let's code it!

2. micro app_prom.py (micro)

```python
from aiohttp import web
from prometheus_client import Counter, generate_latest


async def fire(request):
    request.app["counter"].labels("fire").inc()
    return web.Response(body=b"Fire!")


async def metrics(request):
    request.app["counter"].labels("metrics").inc()
    return web.Response(body=generate_latest())


async def get_application():
    app = web.Application()
    app.add_routes([
        web.get("/fire", fire),
        web.get("/metrics", metrics),
    ])
    app["counter"] = Counter("reqs", "Reqs counter", ["view"])
    return app

```

app_prom.py (1,1) python unix Alt-g: show bindings, CtrlG: open help

```
$ python app_prom.py
```

```
$ for i in `seq 30`; do http :8080/fire; done
```

```
$ http :8080/metrics | grep reqs_total
# HELP reqs_total Reqs counter
# TYPE reqs_total counter
reqs_total{view="fire"} 30.0
reqs_total{view="metrics"} 1.0
```

```
$ python app_prom.py
```

```
$ gunicorn app_prom:get_application \
    --workers=4 \
    --worker-class=aiohttp.GunicornWebWorker
```

```
$ for i in `seq 30`; do http :8000/fire; done
```

```
$ http :8000/metrics | grep reqs_total
# HELP reqs_total Reqs counter
# TYPE reqs_total counter
reqs_total{view="fire"} 13.0
reqs_total{view="metrics"} 1.0
```

```
$ http :8000/metrics | grep reqs_total
# HELP reqs_total Reqs counter
# TYPE reqs_total counter
reqs_total{view="fire"} 9.0
reqs_total{view="metrics"} 1.0
```

Gunicorn is based on the pre-fork worker model. This means that there is a central master process that manages a set of worker processes.


I'm going on an adventure!

Multiprocess Mode

- https://github.com/prometheus/client_python#multiprocess-mode-gunicorn

- The **prometheus_multiproc_dir** environment variable must be set to a directory that the client library can use for metrics.

- **MultiProcessCollector** must be used to collect files with metrics from file system **on each request** to /metrics endpoint.

Multiprocess Mode

- Complex deployment configuration especially in case of containers…

- Multiprocessing mode is **slow**!

Multiprocess Mode

- https://github.com/prometheus/client_python/issues/374
- https://github.com/prometheus/client_python/issues/367
- https://github.com/prometheus/client_python/issues/275
- https://github.com/prometheus/client_python/issues/127
- https://github.com/prometheus/client_python/issues/204
- https://github.com/korfuri/django-prometheus/issues/89
- …


4GIFs.com

pull vs push

Prometheus Pushgateway

- https://github.com/prometheus/pushgateway
- The Prometheus Pushgateway exists to allow ephemeral and batch jobs to expose their metrics to Prometheus.
- The Pushgateway is explicitly **not an aggregator or distributed counter** but rather a metrics cache.
- On each push, metrics **will be replaced**.

```
redis    ← redis_exporter    ← Prometheus
postgres ← postgres_exporter ← Prometheus
xyz      ← xyz_exporter      ← Prometheus
```

Exporters

- statsd + statsd_exporter
- collectd + collectd_exporter
- telegraf

What we want?

- Labels support.
- Easy to deploy and operate.
- Client libraries and community adaptation.

collectd

- https://github.com/collectd/collectd
- https://github.com/prometheus/collectd_exporter

- Limited tags (labels) support (https://github.com/collectd/collectd/pull/1655).
- Complex deployment (two services) and configuration.
- Abandoned client libraries.

telegraf

- https://github.com/influxdata/telegraf

- https://github.com/influxdata/telegraf/tree/master/plugins/outputs/prometheus_client

- Hard to configure aggregations and integration with prometheus.

statsd

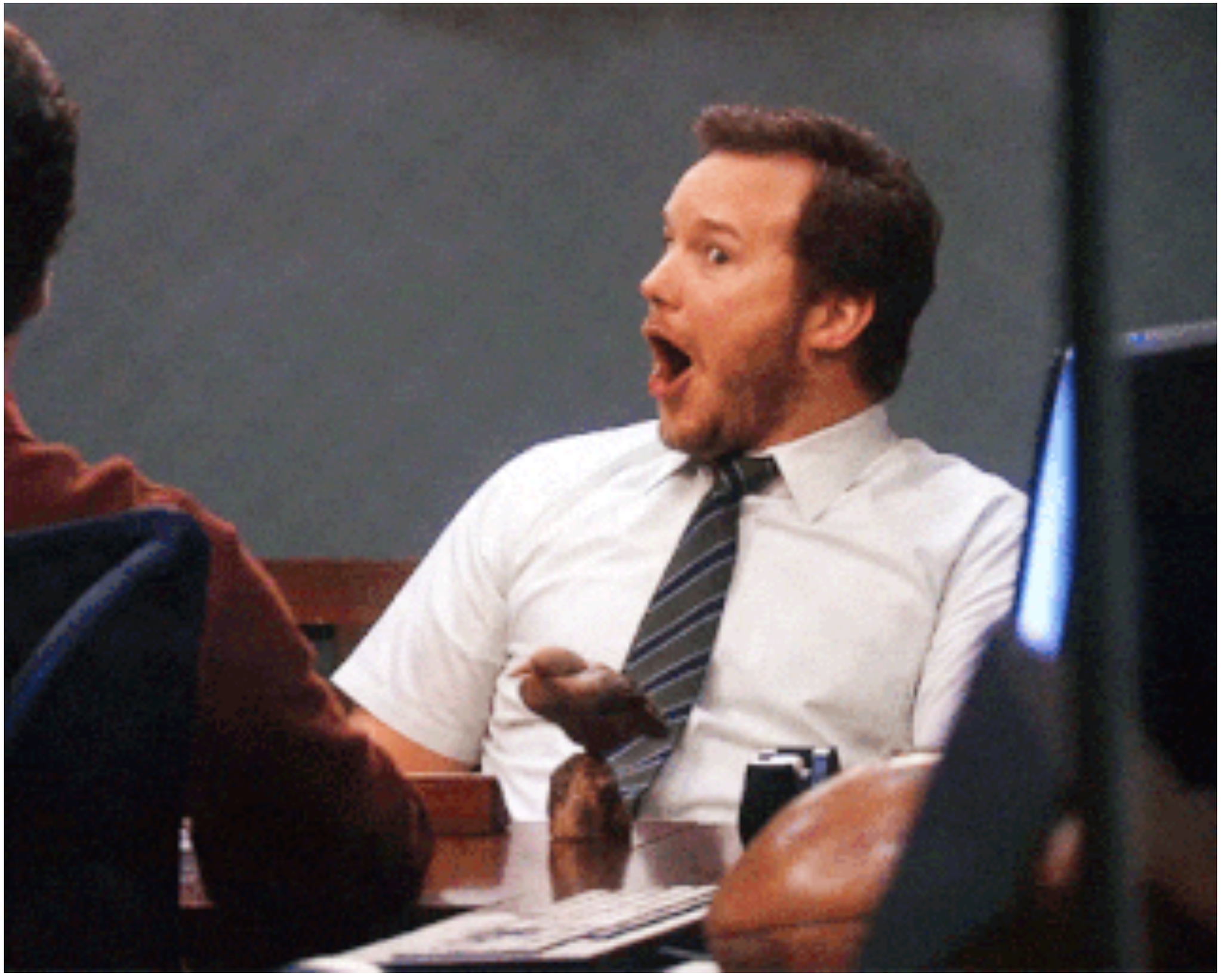
- https://github.com/statsd/statsd
- https://github.com/prometheus/statsd_exporter
- Labels not supported.
- Complex deployment (two services) and configuration.

statsd_exporter

- Since the StatsD exporter uses the same line protocol as StatsD itself, you can also configure your applications to send StatsD metrics directly to the exporter. In that case, **you don't need to run a StatsD server anymore**.

statsd_exporter

- DogStatsD extension (https://docs.datadoghq.com/developers/dogstatsd/).
- name:value|type|@sample_rate**|#tag:value**
- The exporter will convert DogStatsD-style tags to prometheus labels.
- https://pypi.org/project/datadog/



Let's try it!

```
$ docker run -d \
    -p 9102:9102 \
-p 9125:9125 \
-p 9125:9125/udp \
prom/statsd-exporter
```

2. micro app_statsd.py (micro)

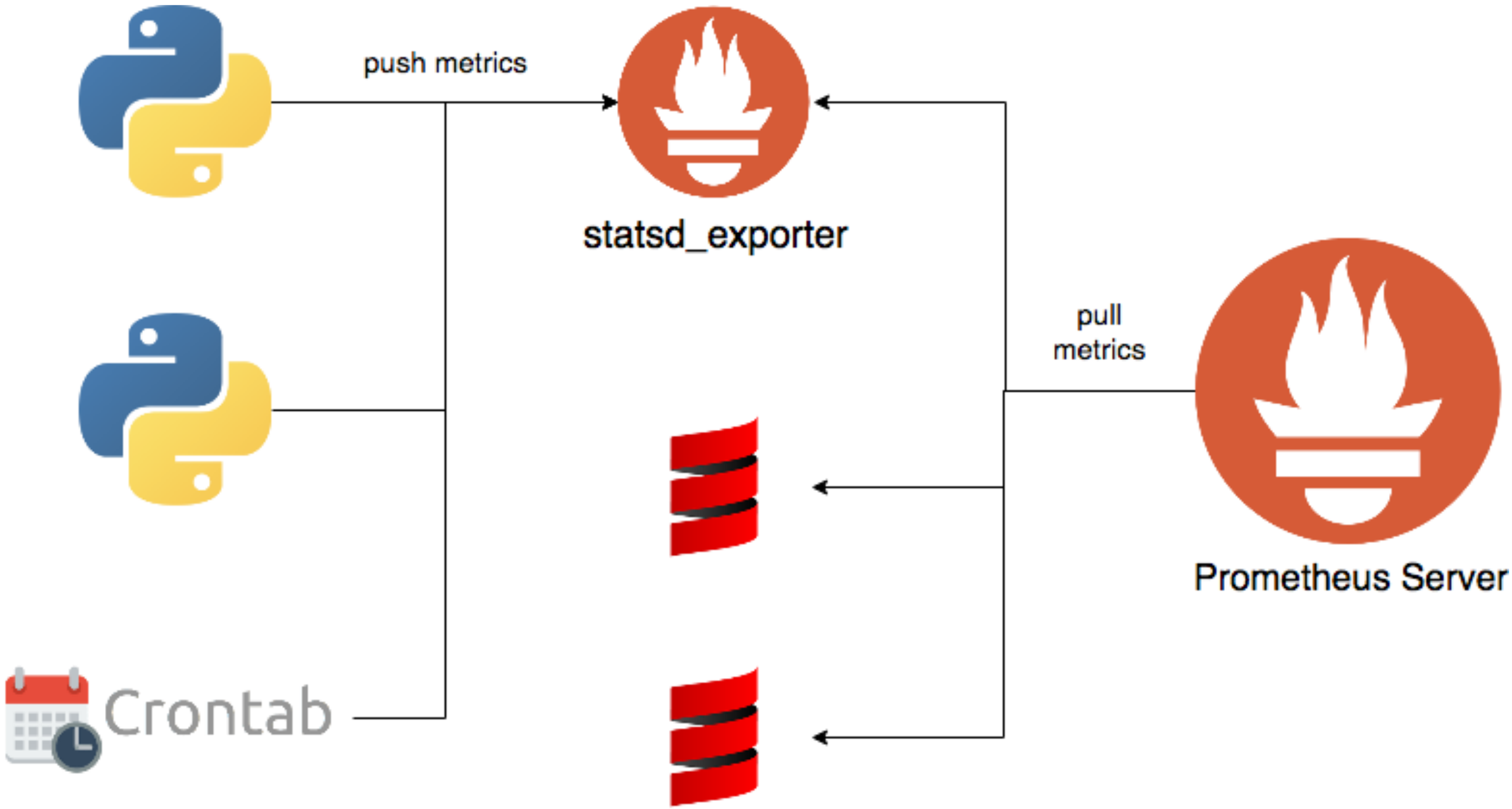
```python
import aiodogstatsd
from aiohttp import web


async def fire(request):
    request.app["statsd"].increment(
        "reqs_total", tags={"view": "fire"}
    )
    return web.Response(body=b"Fire!")


async def statsd_client(app):
    app["statsd"] = aiodogstatsd.Client(host="0.0.0.0", port=9125)
    await app["statsd"].connect()
    yield
    await app["statsd"].close()


async def get_application():
    app = web.Application()
    app.add_routes([web.get("/fire", fire)])
    app.cleanup_ctx.append(statsd_client)
    return app
```

app_statsd.py (1,1) python unix Alt-g: show bindings, CtrlG: open help

```
$ gunicorn app_statsd:get_application \
    --workers=4 \
    --worker-class=aiohttp.GunicornWebWorker
```

```
$ for i in `seq 30`; do http :8000/fire; done
```

```
$ http :9102/metrics | grep reqs_total
# HELP reqs_total Metric autogenerated by statsd_exporter.
# TYPE reqs_total counter
reqs_total{view="fire"} 30
```


push metrics
statsd_exporter
pull
metrics
Prometheus Server
Crontab

StatsD gauge → Prometheus gauge
StatsD counter → Prometheus counter
StatsD timer → Prometheus summary

StatsD gauge → Prometheus gauge
StatsD counter → Prometheus counter
StatsD timer → Prometheus histogram

histogram vs summary

- https://prometheus.io/docs/practices/histograms/
- https://povilasv.me/prometheus-tracking-request-duration/

```
defaults:
timer_type: histogram
buckets: [.05, .1, .25, .5, 1, 2.5]
```

Bonus part!

Gunicorn

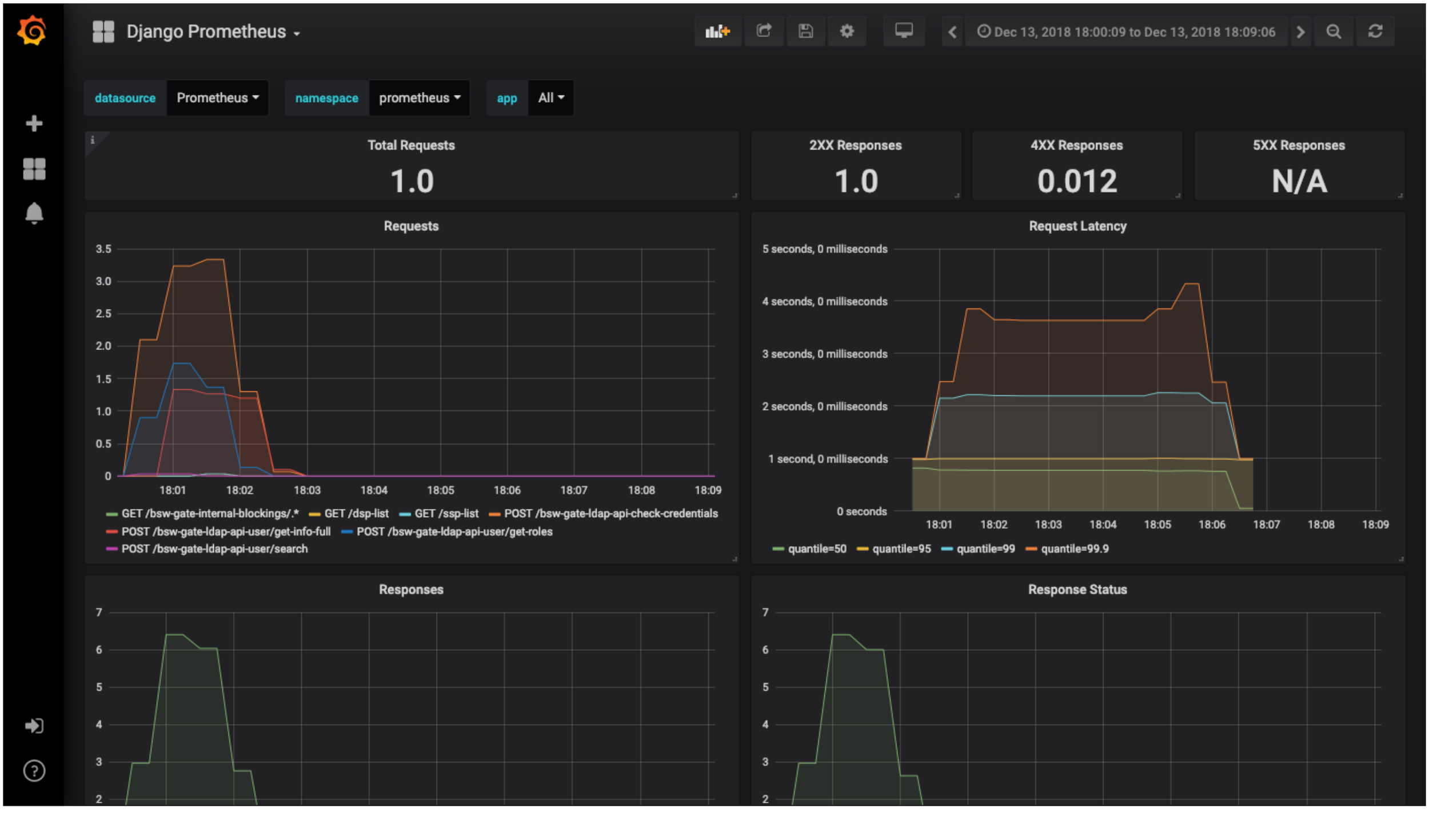
- https://medium.com/@damianmyerscough/monitoring-gunicorn-with-prometheus-78995415006 9
- https://docs.gunicorn.org/en/stable/instrumentation.html
- gunicorn --statsd-host=…
- AIOHTTP not supported…
- Labels not supported…

Sentry

- https://docs.sentry.io/server/internal-metrics/

- Sentry provides an abstraction called ‘metrics’ which is used for internal monitoring, generally timings and various counters.

Grafana


Django Prometheus
Dec 13, 2018 18:00:09 to Dec 13, 2018 18:09:06
datasource
Prometheus
namespace
prometheus
app
All
Total Requests
1.0
2XX Responses
1.0
4XX Responses
0.012
5XX Responses
N/A
Requests
3.5
3.0
2.5
2.0
1.5
1.0
0.5
0
18:01
18:02
18:03
18:04
18:05
18:06
18:07
18:08
18:09
GET /bsw-gate-internal-blockings/.*
GET /dsp-list
GET /ssp-list
POST /bsw-gate-ldap-api-check-credentials
POST /bsw-gate-ldap-api-user/get-info-full
POST /bsw-gate-ldap-api-user/get-roles
POST /bsw-gate-ldap-api-user/search
Request Latency
5 seconds, 0 milliseconds
4 seconds, 0 milliseconds
3 seconds, 0 milliseconds
2 seconds, 0 milliseconds
1 second, 0 milliseconds
0 seconds
quantile=50
quantile=95
quantile=99
quantile=99.9
Responses
Response Status
7
6
5
4
3
2

Grafana

- https://grafana.com/dashboards?dataSource=prometheus

- ~5-10 minutes!

Without StatsD?

Prometheus + Consul

- http://blog.archer.onl/article/collecting-prometheus-metrics-for-python-services-behind-gunicorn-using-consul/
- https://www.consul.io/
- https://github.com/hynek/prometheus_async

https://github.com/Gr1N/talks

Questions?

https://github.com/Gr1N