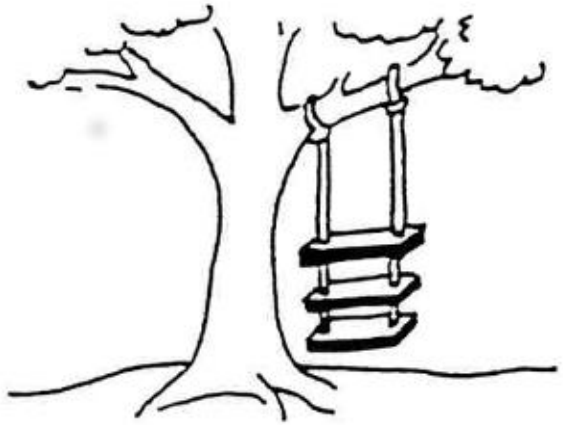
Observing Your Applications With Sentry And Prometheus

PyconDE 2017

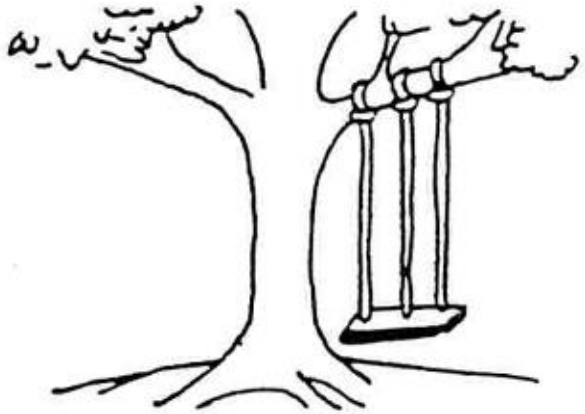
Patrick Muehlbauer @tmuxbee

BlueYonder

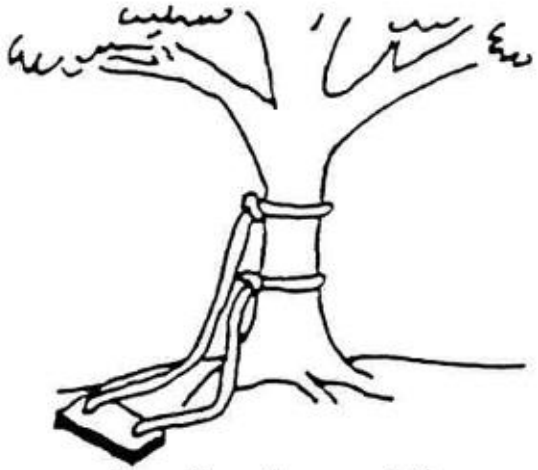
"Problem solving is an art form not fully appreciated by some"



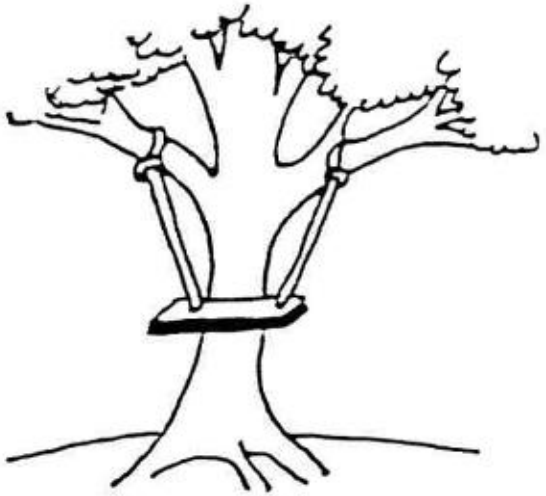
As proposed by the project sponsors



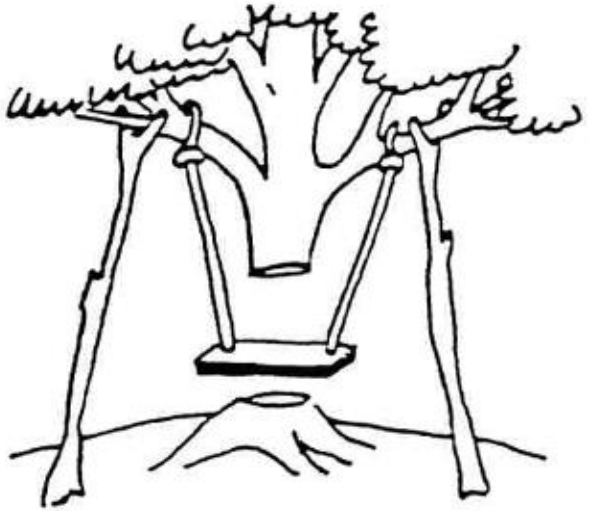
As specified in the project request



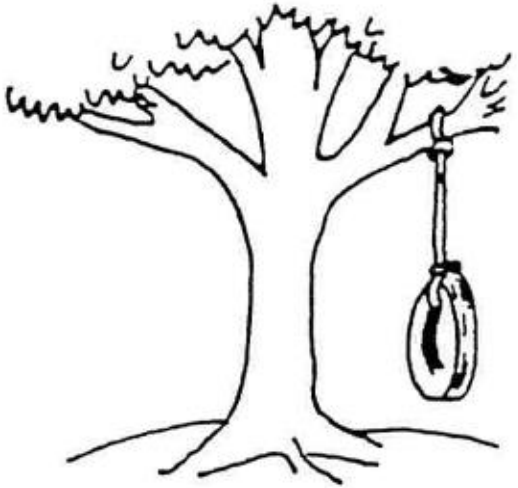
As designed by the senior analyst



As produced by the programmers



As installed at the user's site

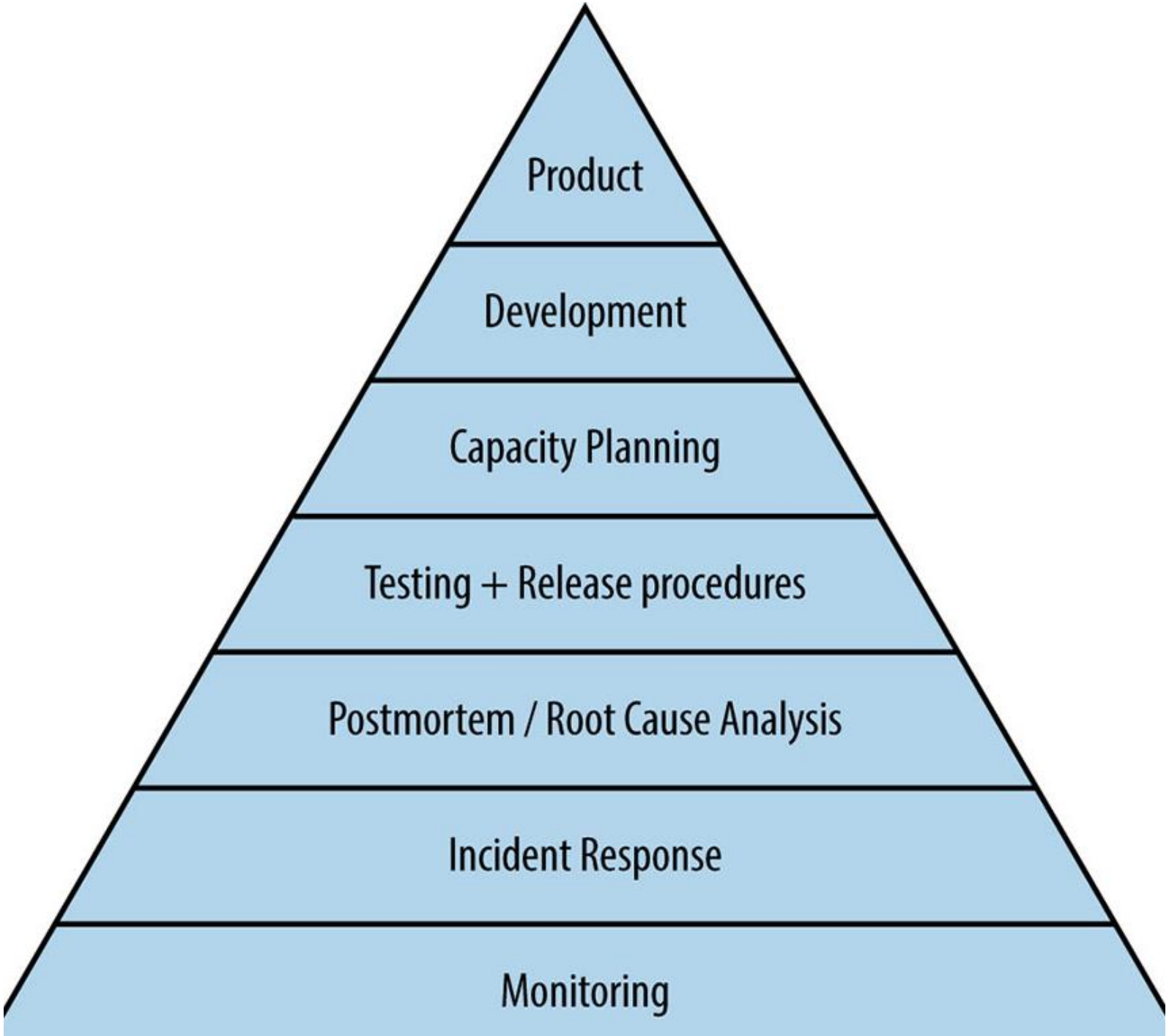


What the user wanted

Tree Swing graphic by S Høgh 1993 - from Businessballs.com/treeswing.htm 2013

How it runs in production

Hierarchy Of Reliability



https://landing.google.com/sre/book/chapters/part3.html

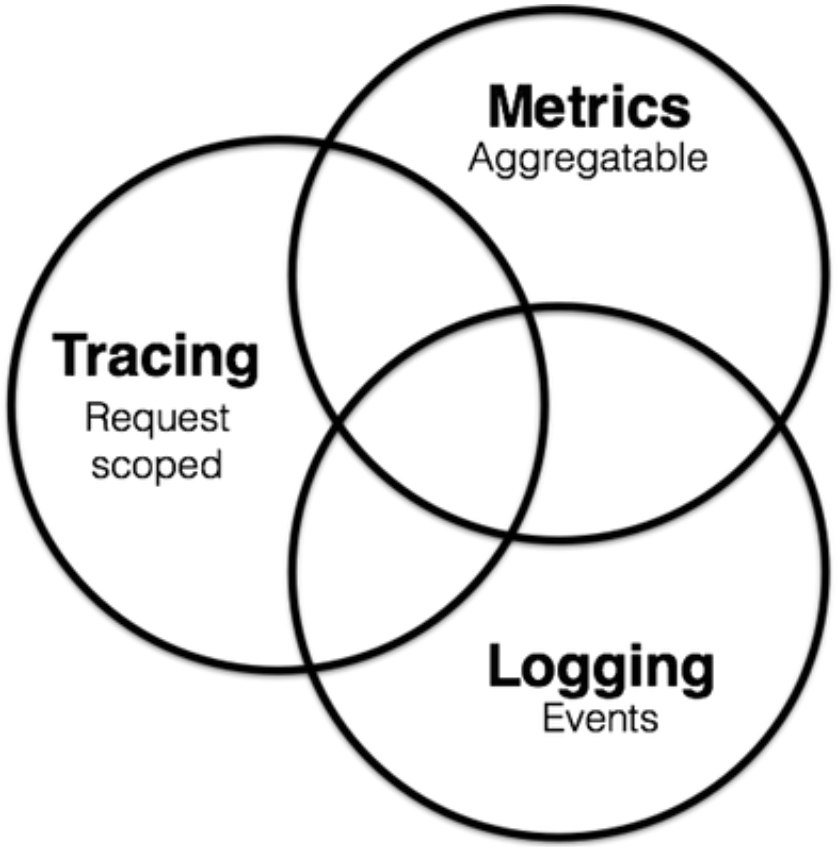
Goals

- **Fix problems** before they happen
- **Quickly pinpoint and fix problems** when they happen
- **Analyze and understand issues** after they happened

BlueYonder

The Three Pillars Of Observability

- **Logging:** Logs are immutable records of discrete events that happened over time
- **Metrics:** A set of numbers describing a particular process or activity
- **Tracing:** capture the lifetime of requests as they flow through the various components of a distributed system

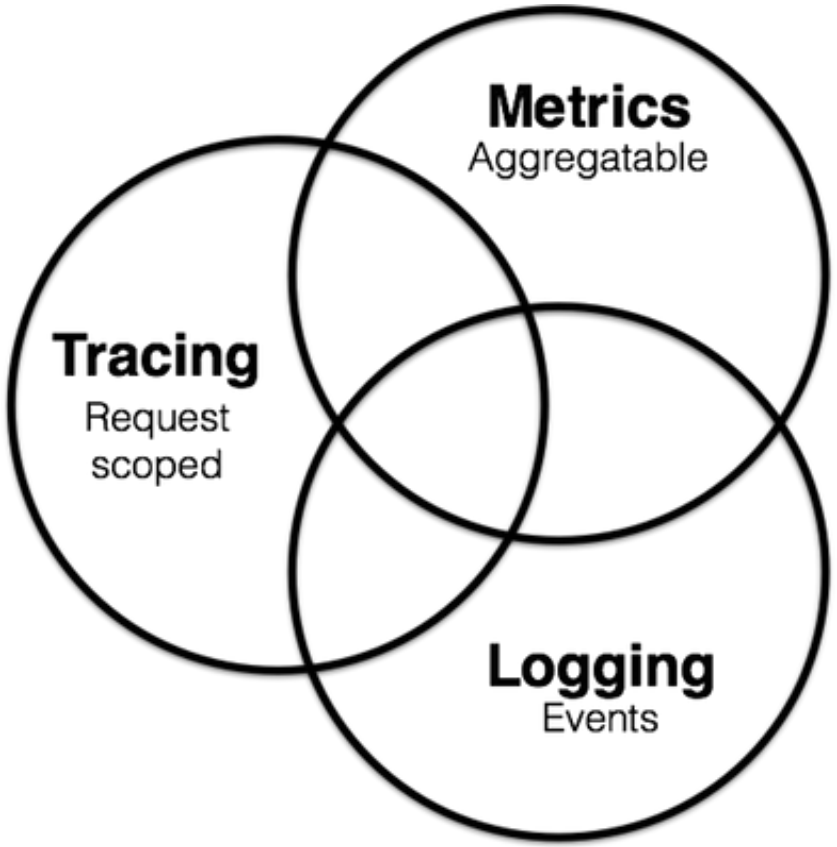


https://peter.bourgon.org/blog/2017/02/21/metrics-tracing-and-logging.html

BlueYonder

The Three Pillars Of Observability

- **Logging:** Logs are immutable records of discrete events that happened over time
- **Metrics:** A set of numbers describing a particular process or activity
- **Tracing:** capture the lifetime of requests as they flow through the various components of a distributed system



https://peter.bourgon.org/blog/2017/02/21/metrics-tracing-and-logging.html

BlueYonder

Error Logging

```
def process_data(data):
    return data[1] / data[0]
```

BlueYonder

Error Logging

```
def process_data(data):
    return data[1] / data[0]

try:
    process_data(data)
except Exception:
    logger.exception('Fatal error while processing data!')
```

BlueYonder

Error Logging

- How to know that an exception occurred?
- How to find the important log messages?

BlueYonder

SENTRY

Error Logging With Sentry

- Sends notifications for events (mail, slack, …)
- Sends notifications *once*
- Aggregates events (collect statistics, identify regressions)
- Clients for multiple languages and platforms
- Opensource / SAAS (with free plans for small projects)

BlueYonder

PyconDE 2017 / pyconde2017

Issues Overview User Feedback Releases

Star Project | Project Settings

ZeroDivisionError /division/<int:numerator>/<int:denominator>

division by zero

ISSUE #	ASSIGNED	EVENTS	USERS
PYCONDE2017-8		4	1

Resolve | Ignore | Link Issue Tracker

Details Comments 0 User Feedback 0 Tags Events

Share this event

Event fc20d04081894574a614f959abe60fa9
Oct 26, 2017 2:25:36 PM UTC | JSON (8.7 KB)

Older Newer

TAGS

browser	Chrome 62.0
device	Other
level	error
os	Linux
server_name	pt-pmuehlbauer
transaction	/division/<int:numerator>/<int:denominator>
url	http://localhost:8000/division/6/0
user	ip:127.0.0.1

MESSAGE

```
ZeroDivisionError: division by zero
```

(Default Environment)

LAST 24 HOURS

LAST 30 DAYS

FIRST SEEN

When: 2 minutes ago
Oct 26, 2017 2:24:48 PM UTC

Release: *not configured*

LAST SEEN

PyconDE 2017 / pyconde2017

Issues Overview User Feedback Releases

Star Project | Project Settings

ZeroDivisionError /division/<int:numerator>/<int:denominator>

division by zero

ISSUE #	ASSIGNED	EVENTS	USERS
PYCONDE2017-8		4	1

Resolve | Ignore | Link Issue Tracker

Details Comments 0 User Feedback 0 Tags Events

Share this event

Event fc20d04081894574a614f959abe60fa9
Oct 26, 2017 2:25:36 PM UTC | JSON (8.7 KB)

Older Newer

(Default Environment)

LAST 24 HOURS

LAST 30 DAYS

FIRST SEEN

When: 2 minutes ago
Oct 26, 2017 2:24:48 PM UTC

Release: *not configured*

LAST SEEN

TAGS

browser	Chrome 62.0
device	Other
level	error
os	Linux
server_name	pt-pmuehlbauer
transaction	/division/<int:numerator>/<int:denominator>
url	http://localhost:8000/division/6/0
user	ip:127.0.0.1

MESSAGE

```
ZeroDivisionError: division by zero
```

EXCEPTION (most recent call first)

App Only | Full | Raw

ZeroDivisionError

division by zero

pyconde2017/app.py in **division** at line **11**

```
6. sentry = Sentry(app,
dsn='http://bc211d6ae07645b0aacc9a532e2f952b:3969d142a1f34d8faa4be82ea174cd3b@localhost:9000/2')
7.
8.
9. @app.route('/division/<int:numerator>/<int:denominator>', methods=['GET'])
10. def division(numerator, denominator):
11.     result = numerator / denominator
12.     return f"{numerator} / {denominator} = {result}"
13.
14.
15. def process_data(data):
16.     new_data = []
```

denominator	0
numerator	6

Called from: **flask/app.py** in **dispatch_request**

When: 2 minutes ago

Oct 26, 2017 2:25:36 PM UTC

Release: *not configured*

Tags

browser	50%	Chrome 62.0
device	100%	Other
level	100%	error
os	50%	Linux
server_name	100%	pt-pmuehlbauer
transaction	100%	/division/<int:numera...
url	50%	http://localhost:8000...
user	100%	127.0.0.1

Notifications

```python
from flask import Flask
from raven.contrib.flask import Sentry

app = Flask(__name__)
sentry = Sentry(app, dsn='http://a532e2f952b:82ea174cd3b@localhost:9000/2')


@app.route('/division/<int:numerator>/<int:denominator>', methods=['GET'])
def division(numerator, denominator):
    result = numerator / denominator
    return f"{numerator} / {denominator} = {result}"
```

BlueYonder

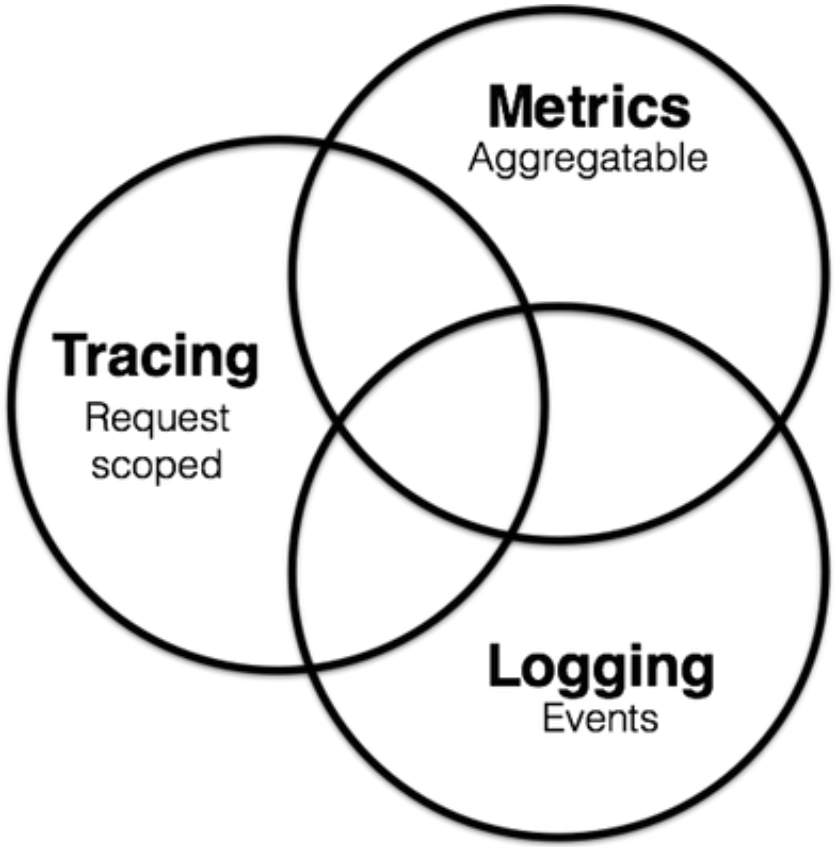
```python
from flask import Flask
from raven.contrib.flask import Sentry

app = Flask(__name__)
sentry = Sentry(app, dsn='http://a532e2f952b:82ea174cd3b@localhost:9000/2')


@app.route('/division/<int:numerator>/<int:denominator>', methods=['GET'])
def division(numerator, denominator):
    result = numerator / denominator
    return f"{numerator} / {denominator} = {result}"
```

BlueYonder

```python
# django
INSTALLED_APPS = (
    'raven.contrib.django.raven_compat',
)

# plain python
from raven import Client

client = Client('http://a532e2f952b:82ea174cd3b@localhost:9000/2')

try:
    result = division(numerator, denominator)
except Exception:
    client.captureException()
```

The Three Pillars Of Observability

- **Logging:** Logs are immutable records of discrete events that happened over time
- **Metrics:** A set of numbers describing a particular process or activity
- **Tracing:** capture the lifetime of requests as they flow through the various components of a distributed system



https://peter.bourgon.org/blog/2017/02/21/metrics-tracing-and-logging.html

BlueYonder

Metrics

A set of numbers describing a particular process or activity

# of requests	3	17	22	38
CPU load	0.3	0.5	0.2	0.8
time	03:20	03:21	03:22	03:23

BlueYonder


2.25
2.00
1.75
1.50
1.25
1.00
0.75
0.50
0.25
0
18:00
18:30
19:00
19:30
20:00
20:30
21:00
21:30
22:00
22:30
23:00
23:30

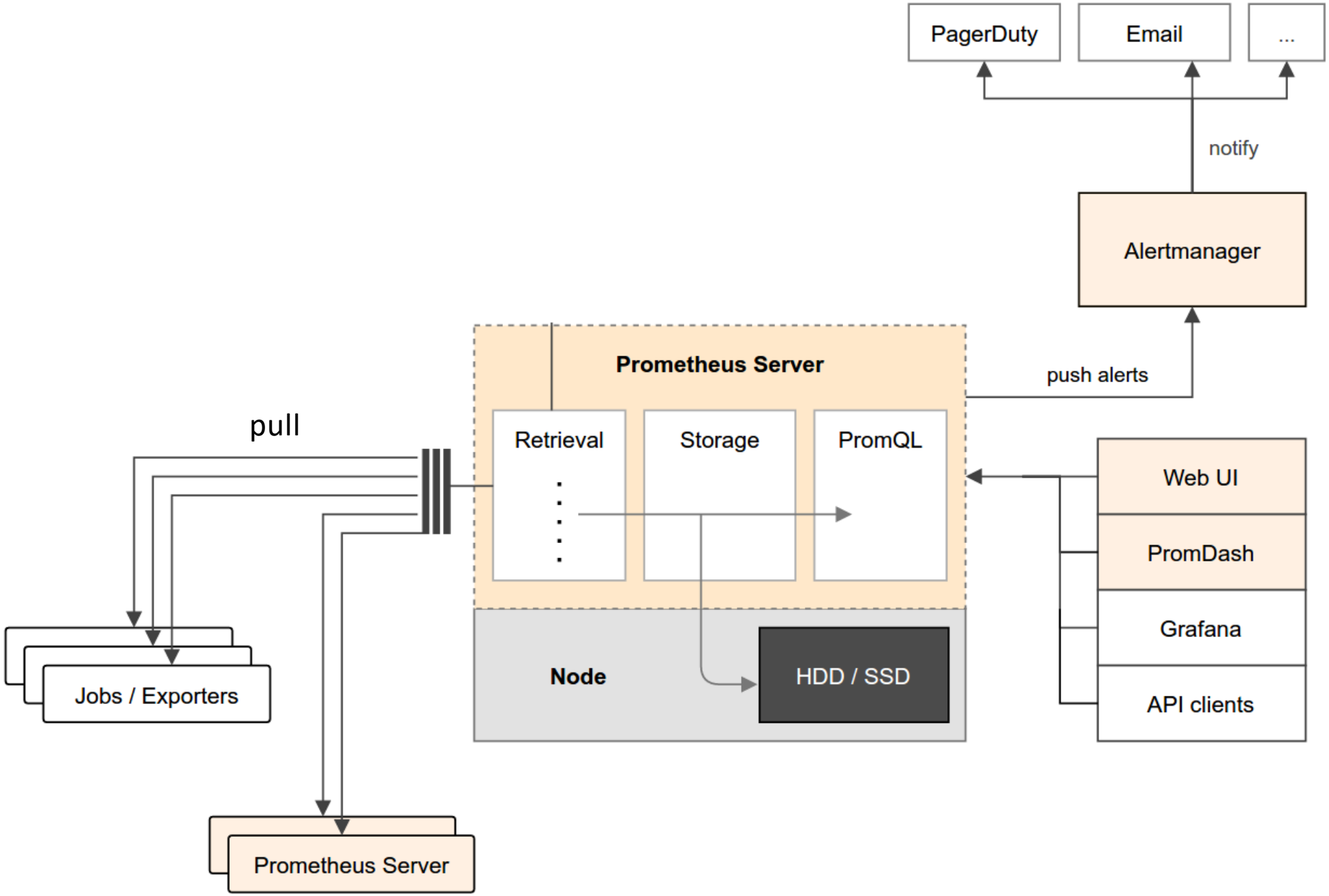
Prometheus

- time series database for system and application metrics
- multi-dimensional data model
  - http_request_count{job='pyconde2017'}
- expressive query language
  - aggregation rules
  - graphing
  - alerting

Metric Types

- Counter - tracks counts of events or running totals
- Gauge - to report instantaneous values
- Histogram - tracks the size and number of events in buckets
- Summary - tracks the size and number of events

BlueYonder


PagerDuty
Email
...
notify
Alertmanager
push alerts
Prometheus Server
Retrieval
Storage
PromQL
pull
Node
HDD / SSD
Web UI
PromDash
Grafana
API clients
Jobs / Exporters
Prometheus Server

Prometheus Config

```
scrape_configs:
  - job_name: 'pyconde'
    static_configs:
    - targets: ['localhost:8000']
```

BlueYonder

Instrumenting the Python

pip install prometheus_client

```
from prometheus_client import start_http_server


if __name__ == '__main__':
    start_http_server(8000)
    app.run()
```

BlueYonder

Instrumenting the Python

```
from prometheus_client import (generate_latest, CollectorRegistry,
                               CONTENT_TYPE_LATEST, Gauge, Histogram)

IN_PROGRESS = Gauge("pyconde_inprogress_requests",
                    "Number of HTTP requests in progess")
REQUEST_TIME = Histogram("pyconde_request_duration_seconds",
                         "Time spent in HTTP requests.")
DIVISION_TIME = Histogram("pyconde_division_duration_seconds",
                          "Time spent doing divisions.")
```

BlueYonder

Instrumenting the Python

```
@app.route('/division/<int:numerator>/<int:denominator>', methods=['GET'])
@REQUEST_TIME.time()
@IN_PROGRESS.track_inprogress()
def division(numerator, denominator):
    time.sleep(random.random() + 0.1)
    with DIVISION_TIME.time():
        time.sleep(random.random() * 2)
        result = numerator / denominator
    return f"{numerator} / {denominator} = {result}"
```

BlueYonder

Instrumenting the Python

```
@app.route('/division/<int:numerator>/<int:denominator>', methods=['GET'])
@REQUEST_TIME.time()
@IN_PROGRESS.track_inprogress()
def division(numerator, denominator):
    time.sleep(random.random() + 0.1)
    with DIVISION_TIME.time():
        time.sleep(random.random() * 2)
        result = numerator / denominator
    return f"{numerator} / {denominator} = {result}"
```

BlueYonder

Instrumenting the Python

```
@app.route('/division/<int:numerator>/<int:denominator>', methods=['GET'])
@REQUEST_TIME.time()
@IN_PROGRESS.track_inprogress()
def division(numerator, denominator):
    time.sleep(random.random() + 0.1)
    with DIVISION_TIME.time():
        time.sleep(random.random() * 2)
        result = numerator / denominator
    return f"{numerator} / {denominator} = {result}"
```

BlueYonder

```
# HELP pyconde_request_duration_seconds Time spent in HTTP requests.
# TYPE pyconde_request_duration_seconds histogram
pyconde_request_duration_seconds_bucket{le="0.005"} 6.0
pyconde_request_duration_seconds_bucket{le="0.01"} 6.0
pyconde_request_duration_seconds_bucket{le="0.1"} 6.0
pyconde_request_duration_seconds_bucket{le="+Inf"} 6.0
pyconde_request_duration_seconds_count 6.0
pyconde_request_duration_seconds_sum 0.0006361549999382987
```

BlueYonder

Alerts

```
ALERT InstanceDiskFillingUp
  IF (predict_linear(node_filesystem_free[1h], 2 * 3600) < 0)
  FOR 5m
  LABELS {severity="page"}
  ANNOTATIONS {description="Disk will run out of disk in 2 hours."}
```

BlueYonder

More Prometheus

https://hynek.me/talks/prometheus/

BlueYonder

Recap

- We cannot avoid all failures
- Sentry (sentry.io) for error logging
- Prometheus (prometheus.io) for metrics collecting

BlueYonder

Thanks!

Patrick Muehlbauer

@tmuxbee

BlueYonder

https://github.com/blue-yonder/

https://tech.blue-yonder.com/