Alerting in the **Prometheus** Universe

Fabian Reinartz, CoreOS

github.com/fabxc

@fabxc

A lot of traffic to monitor

Monitoring traffic should not be proportional to user traffic

A lot of targets to monitor

A single host can run hundreds of machines/procs/containers/...

Targets constantly change

Deployments, scaling up, scaling down, and rescheduling

Need a fleet-wide view

What’s my 99th percentile request latency across all frontends?

Drill-down for investigation

Which pod/node/... has turned unhealthy? How and why?

Monitor all levels, with the same system

Query and correlate metrics across the stack

Translate that to

Meaningful Alerting

Machine Learning

Anomaly Detection

Automated Alert Correlation

Self-Healing

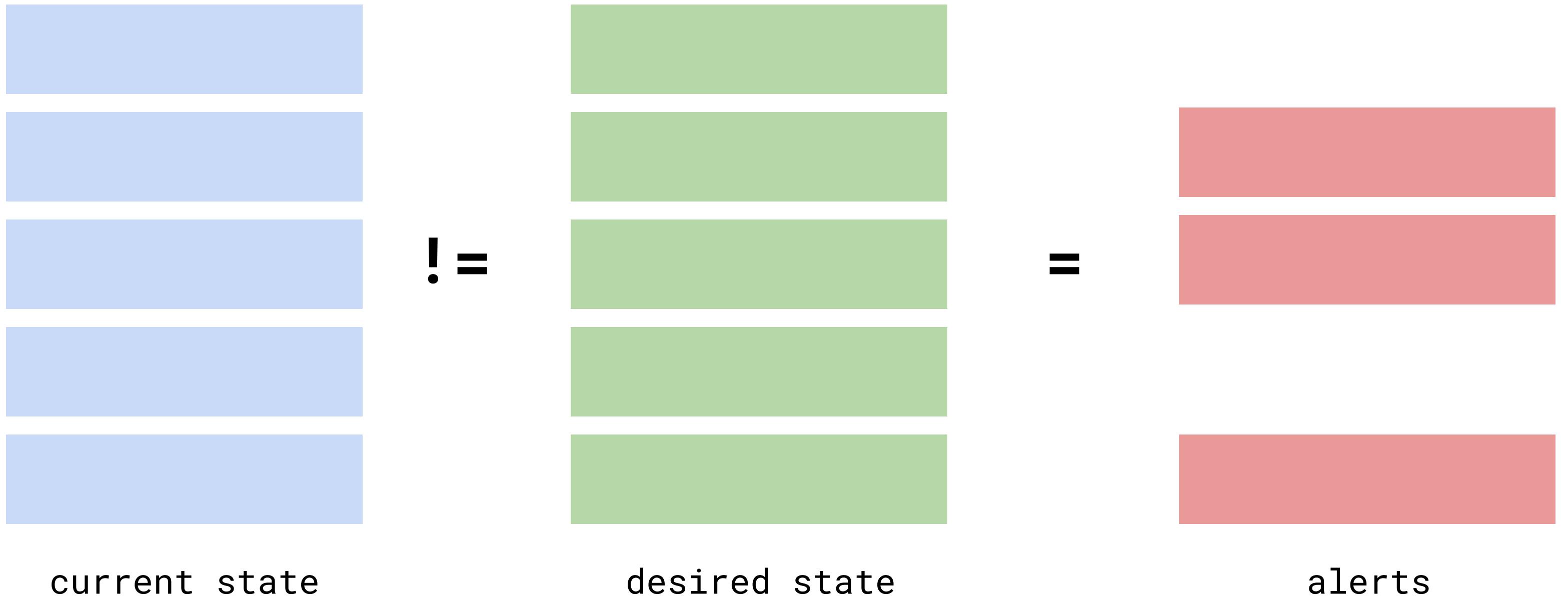
~~Anomaly Detection~~

If you are actually monitoring at scale, *something will always correlate*.

Huge efforts to eleminate huge number of false positives.

Huge chance to introduce false negatives.

Prometheus Alerts



current state != desired state = alerts

Prometheus Alerts

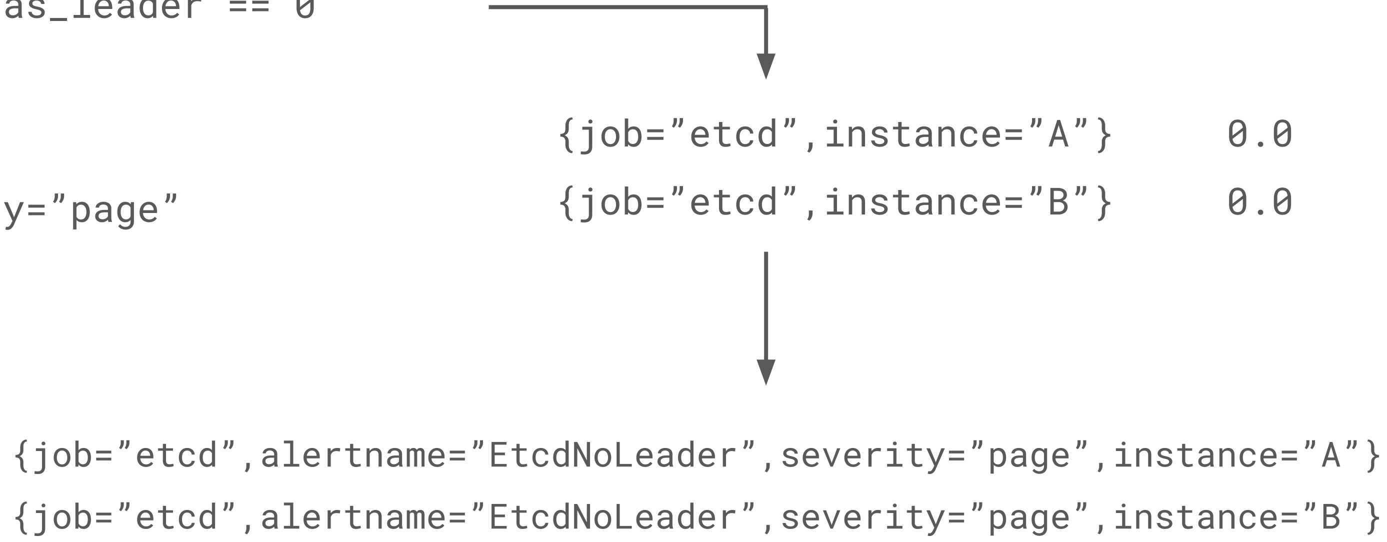
```
ALERT <alert name>
   IF <PromQL vector expression>
   FOR <duration>
   LABELS { ... }
   ANNOTATIONS { ... }
```

Each result entry is one alert:

```
<elem1>     <val1>
<elem2>     <val2>
<elem3>     <val3>
 ...
```

Prometheus Alerts

```
ALERT EtcdNoLeader
   IF etcd_has_leader == 0
   FOR 1m
   LABELS {
     severity=”page”
   }
```

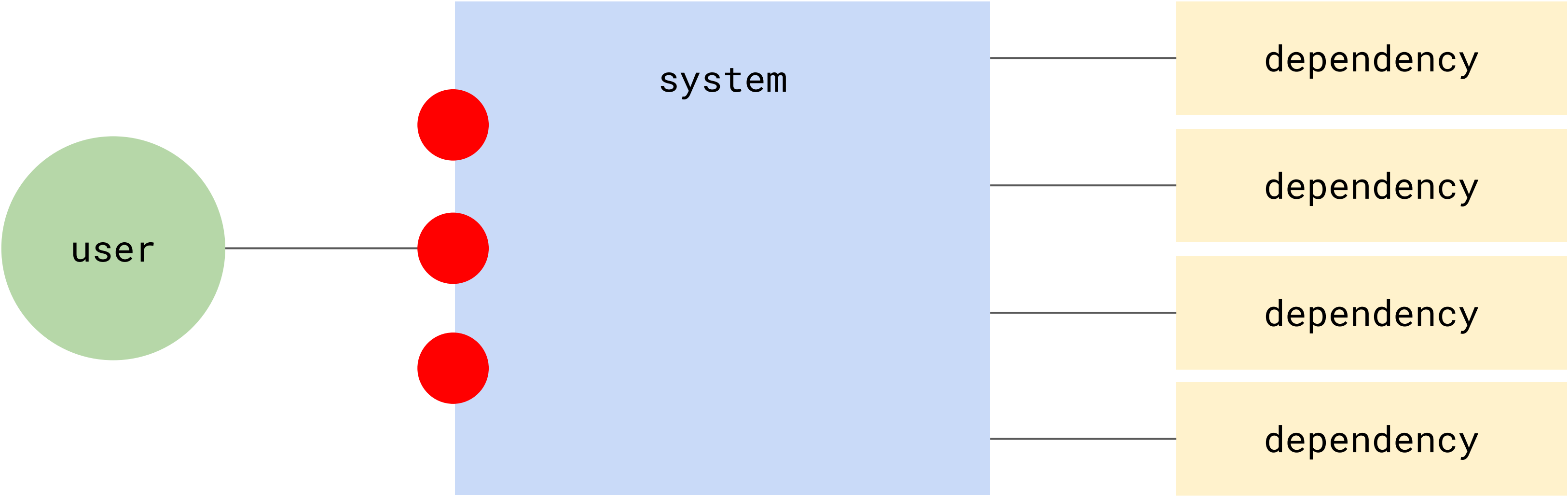


```
{job=”etcd”,instance=”A”}     0.0
{job=”etcd”,instance=”B”}     0.0
```

```
{job=”etcd”,alertname=”EtcdNoLeader”,severity=”page”,instance=”A”}
{job=”etcd”,alertname=”EtcdNoLeader”,severity=”page”,instance=”B”}
```

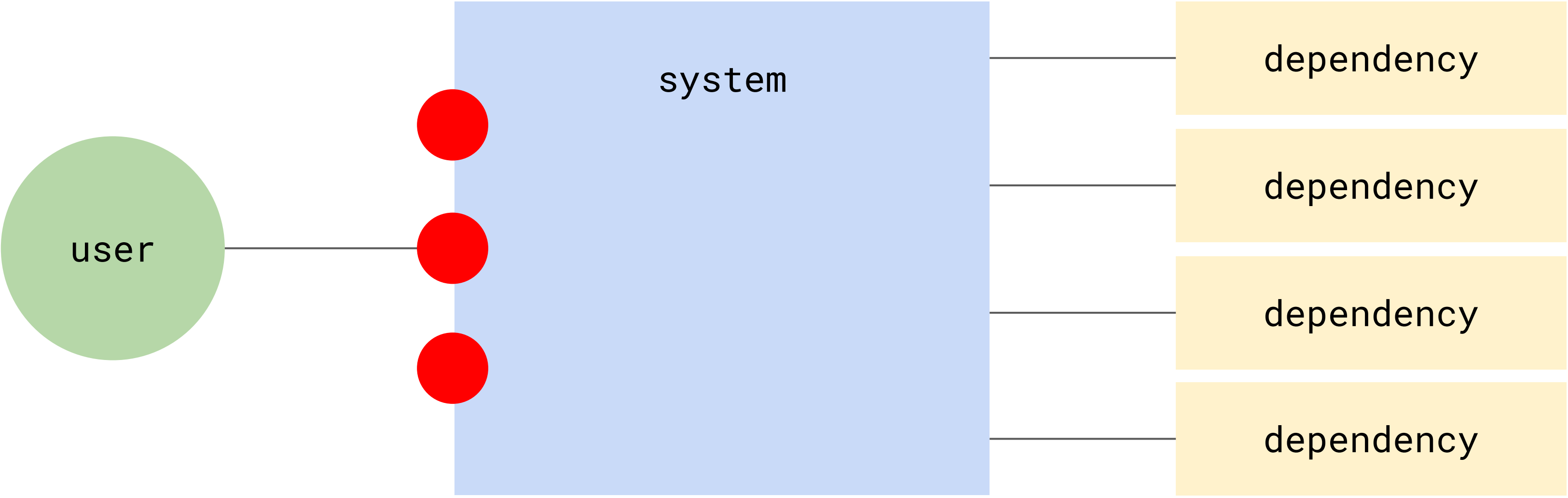
Symptom-based pages

Urgent issues – Does it hurt your user?



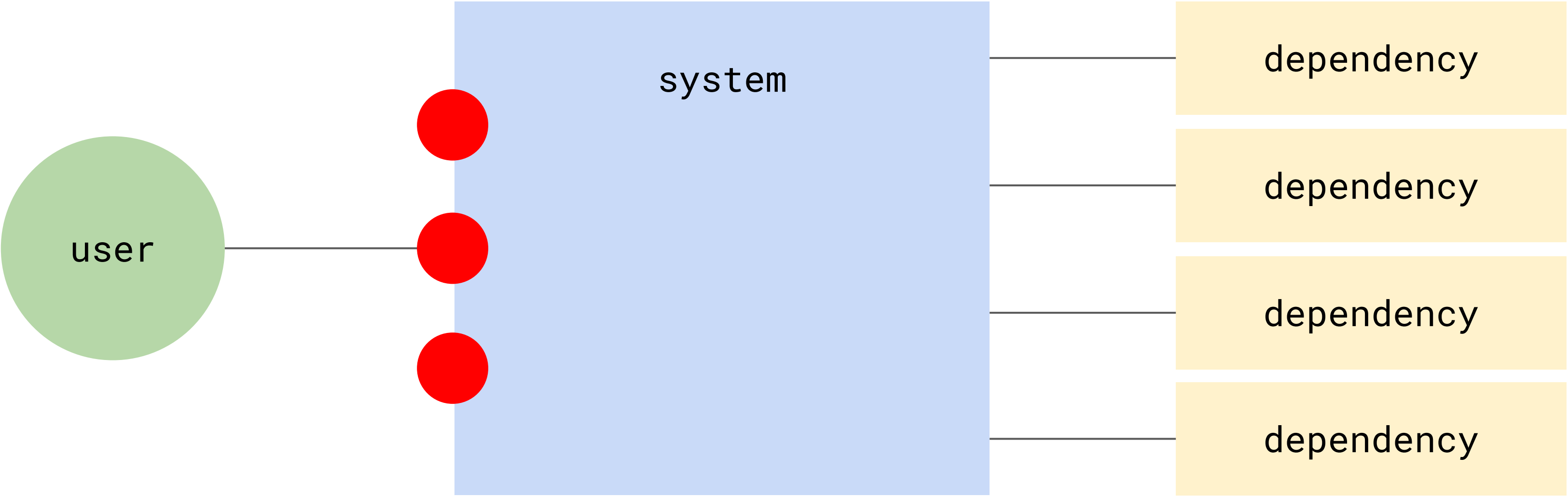
Latency

Four Golden Signals



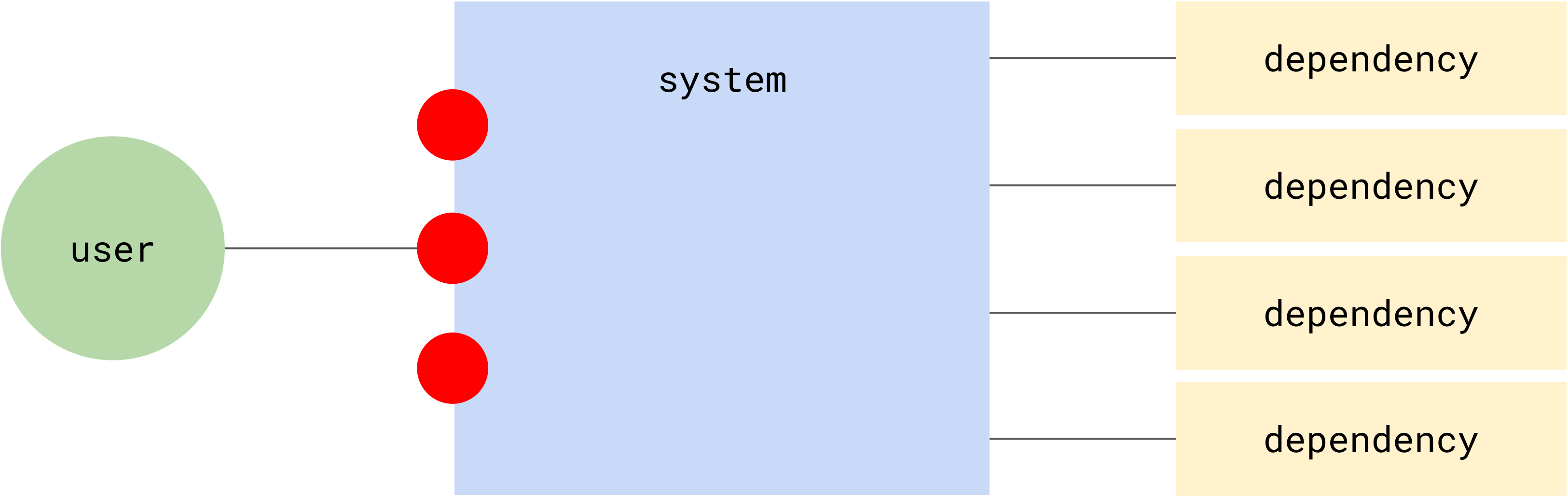
Traffic

Four Golden Signals



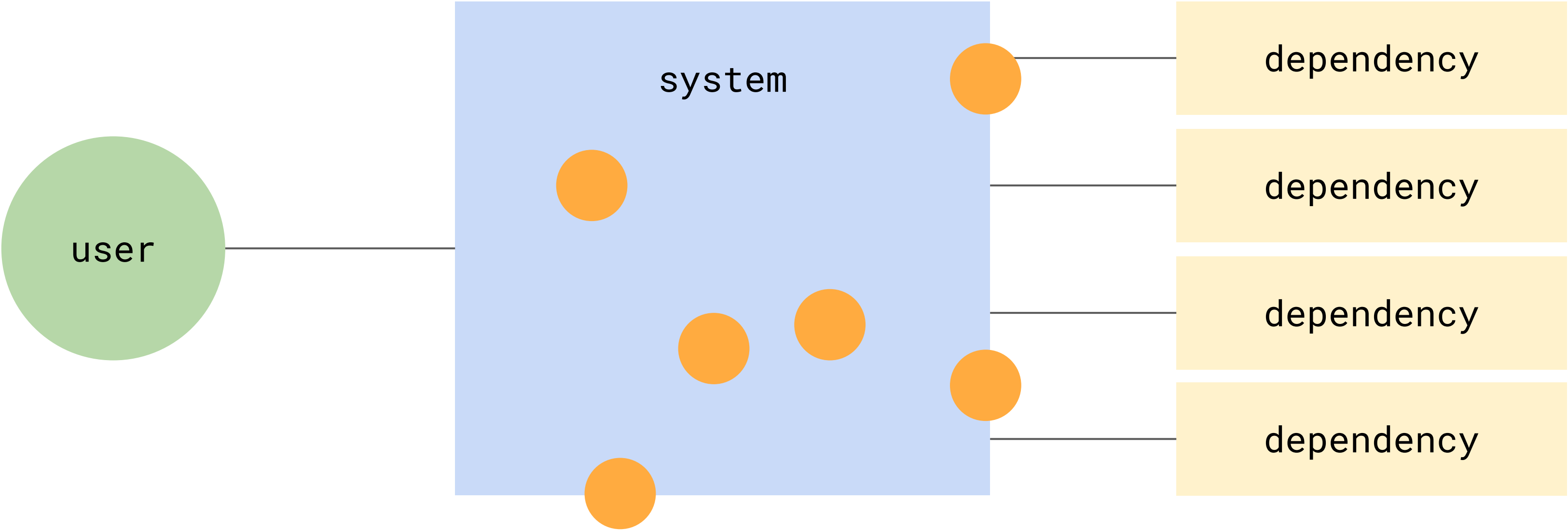
Errors

Four Golden Signals



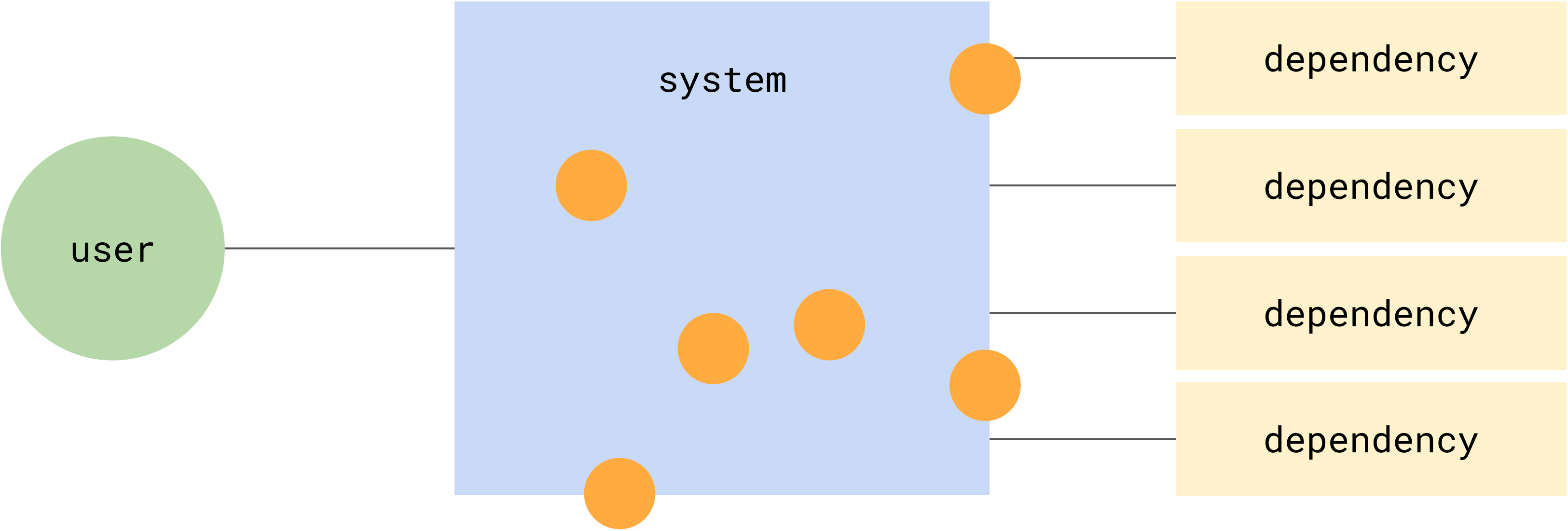
Cause-based warnings

Helpful context, non-urgent problems



Saturation / Capacity

Four Golden Signals



```
requests_total{instance="web-1", path="/index", method="GET"}
requests_total{instance="web-1", path="/index", method="POST"}
requests_total{instance="web-3", path="/api/profile", method="GET"}
requests_total{instance="web-2", path="/api/profile", method="GET"}
 …
 …

request_errors_total{instance="web-1", path="/index", method="GET"}
request_errors_total{instance="web-1", path="/index", method="POST"}
request_errors_total{instance="web-3", path="/api/profile", method="GET"}
request_errors_total{instance="web-2", path="/api/profile", method="GET"}
 …
 …
```

```
ALERT HighErrorRate
  IF sum rate(request_errors_total[5m])) > 500
```

```
{}      534
```

```
ALERT HighErrorRate
  IF sum rate(request_errors_total[5m])) > 500
```

```
{}      534
```

Ehhh

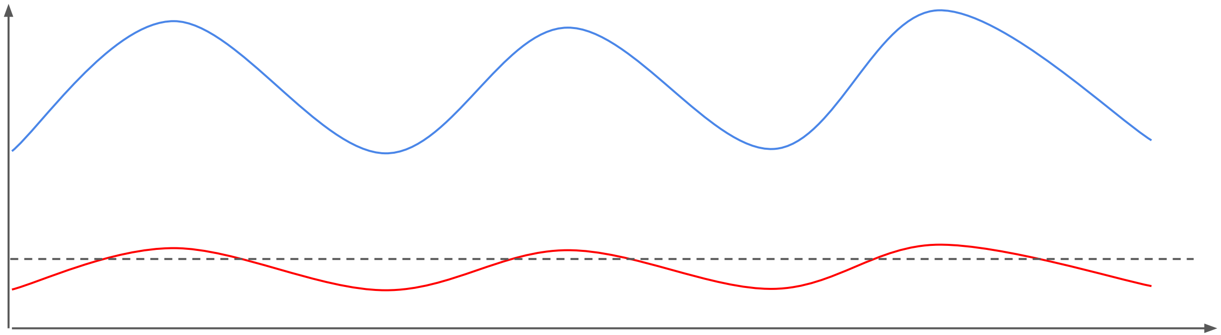
Absolute threshold

alerting rule needs constant tuning as traffic changes

```
ALERT HighErrorRate
  IF sum rate(request_errors_total[5m])) > 500
```

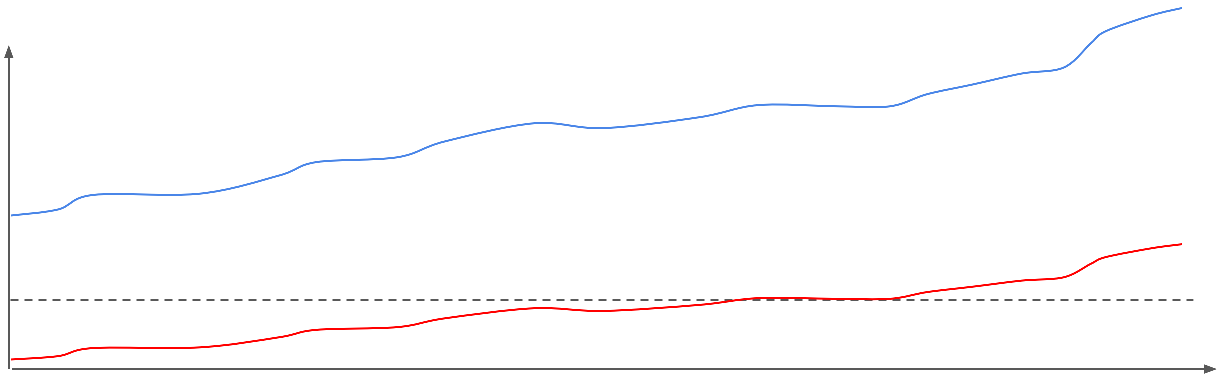
```
{}      534
```

traffic changes over days



```
ALERT HighErrorRate
  IF sum rate(request_errors_total[5m])) > 500
```

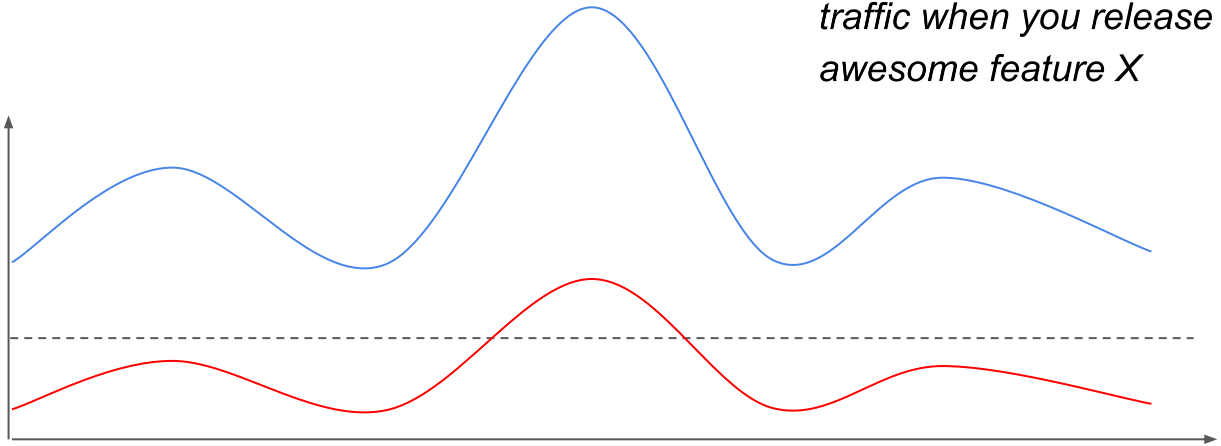
```
{}      534
```



traffic changes over months

```
ALERT HighErrorRate
  IF sum rate(request_errors_total[5m])) > 500
```

```
{}      534
```



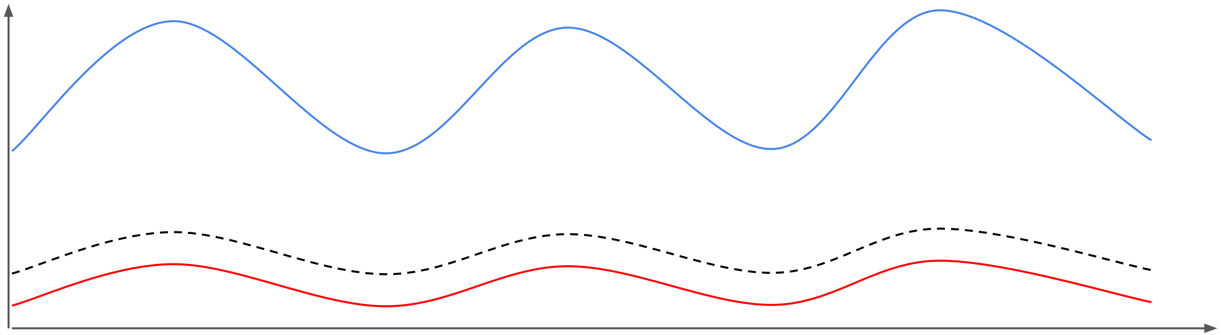
traffic when you release awesome feature X

```
ALERT HighErrorRate
   IF sum rate(request_errors_total[5m]) /
      sum rate(requests_total[5m]) * 100 > 1
```

```
{}      1.8354
```

```
ALERT HighErrorRate
  IF sum rate(request_errors_total[5m]) /
     sum rate(requests_total[5m]) > 0.01
```

```
{}      1.8354
```



```
ALERT HighErrorRate
   IF sum rate(request_errors_total[5m]) /
      sum rate(requests_total[5m]) * 100 > 1
```

```
{}      1.8354
```

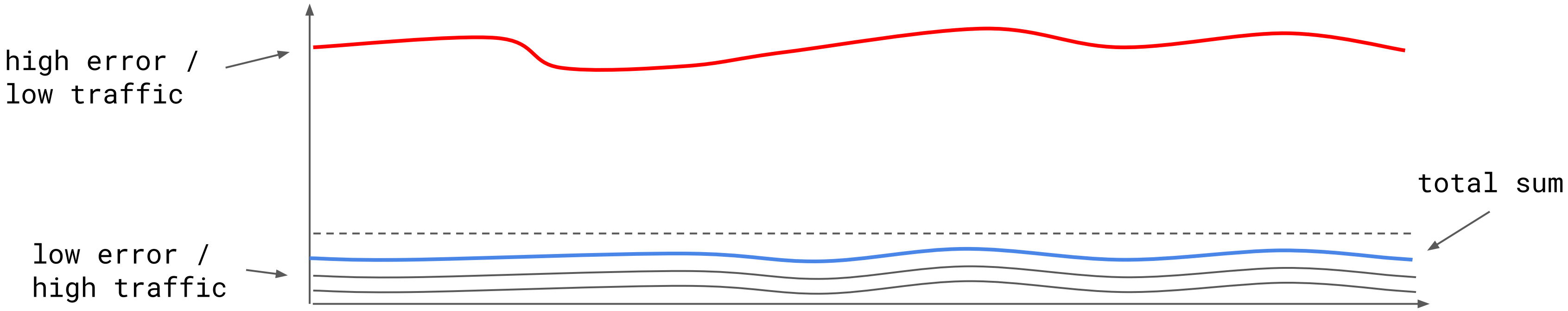
Meehh

No dimensionality in result

loss of detail, signal cancelation

```
ALERT HighErrorRate
  IF sum rate(request_errors_total[5m]) /
     sum rate(requests_total[5m]) * 100 > 1
```

```
{}       1.8354
```



```
ALERT HighErrorRate
  IF sum by(instance, path) rate(request_errors_total[5m]) /
     sum by(instance, path) rate(requests_total[5m]) * 100 > 0.01
```

```
{instance=”web-2”, path=”/api/comments”}      2.435
{instance=”web-1”, path=”/api/comments”}      1.0055
{instance=”web-2”, path=”/api/profile”}       34.124
```

```
ALERT HighErrorRate
   IF sum by(instance, path) rate(request_errors_total[5m]) /
      sum by(instance, path) rate(requests_total[5m]) * 100 > 1
```

```
{instance=”web-2”, path=”/api/v1/comments”}      2.435
 ...
```

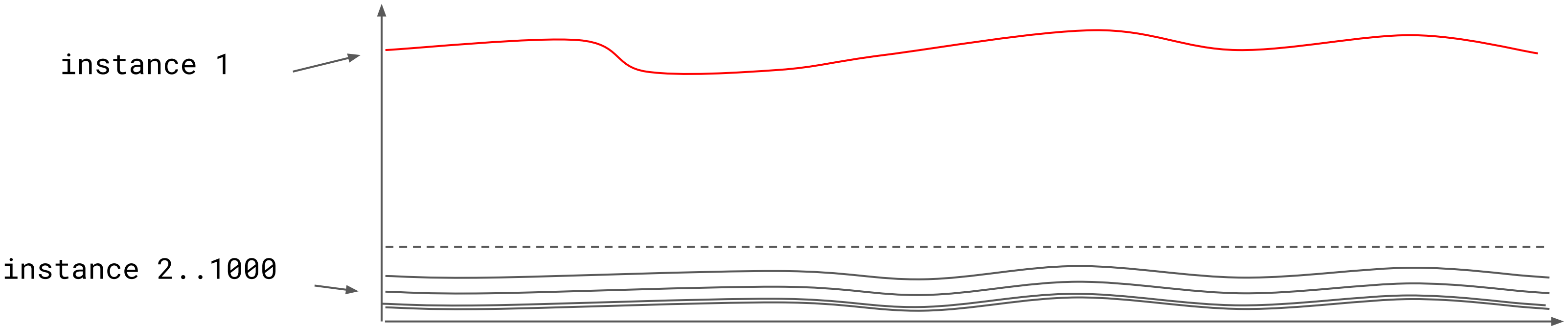
Booo

Wrong dimensions

aggregates away dimensions of fault-tolerance

```
ALERT HighErrorRate
   IF sum by(instance, path) rate(request_errors_total[5m]) /
      sum by(instance, path) rate(requests_total[5m]) * 100 > 1
```

```
{instance=”web-2”, path=”/api/v1/comments”}     2.435
 ...
```

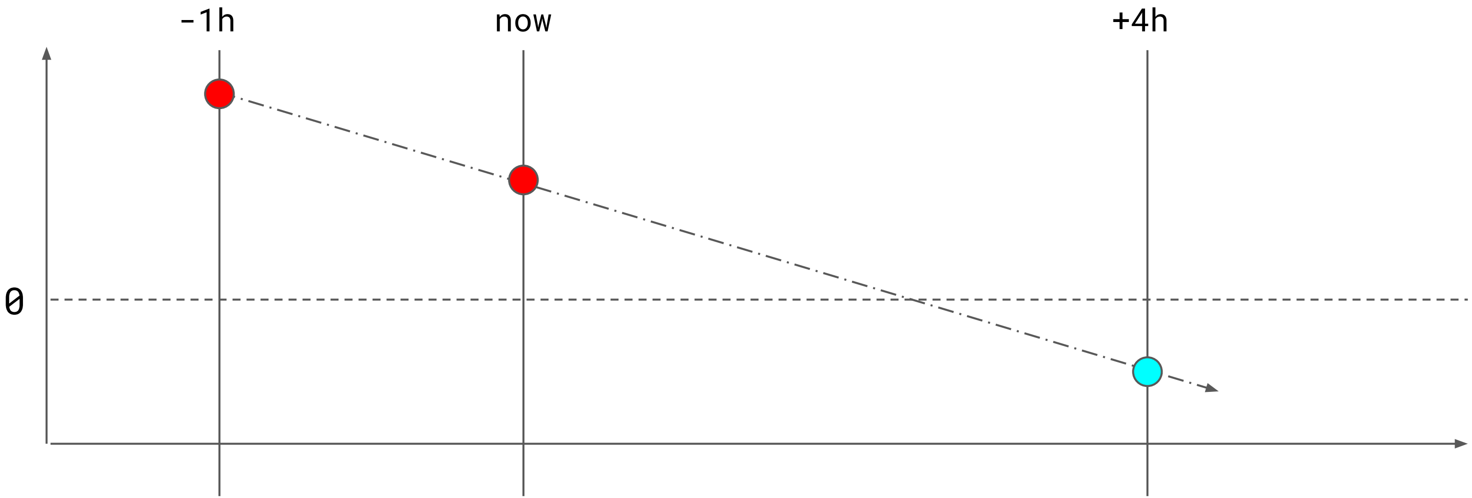


```
ALERT HighErrorRate
   IF sum without(instance) rate(request_errors_total[5m]) /
      sum without(instance) rate(requests_total[5m]) * 100 > 1
```

```
{method=”GET”, path=”/api/v1/comments”}       2.435
{method=”POST”, path=”/api/v1/comments”}      1.0055
{method=”POST”, path=”/api/v1/profile”}       34.124
```

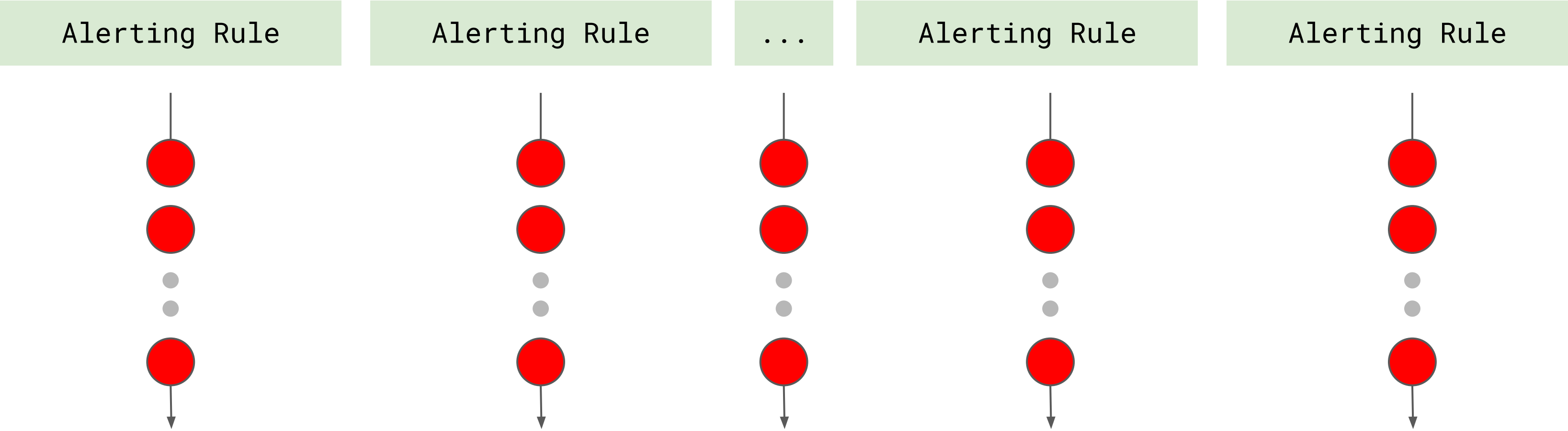
```
ALERT DiskWillFillIn4Hours
    IF predict_linear(node_filesystem_free{job='node'}[1h], 4*3600) < 0
    FOR 5m
    ANNOTATIONS {
       summary     = “device filling up”,
       description = “{{$labels.device}} mounted on {{$labels.mountpoint}} on
                      {{$labels.instance}} will fill up within 4 hours.”
    }
```

```
ALERT DiskWillFillIn4Hours
    IF predict_linear(node_filesystem_free{job='node'}[1h], 4*3600) < 0
    FOR 5m
    ...
```



Alertmanager

Aggregate, deduplicate, and route alerts



Alerting Rule

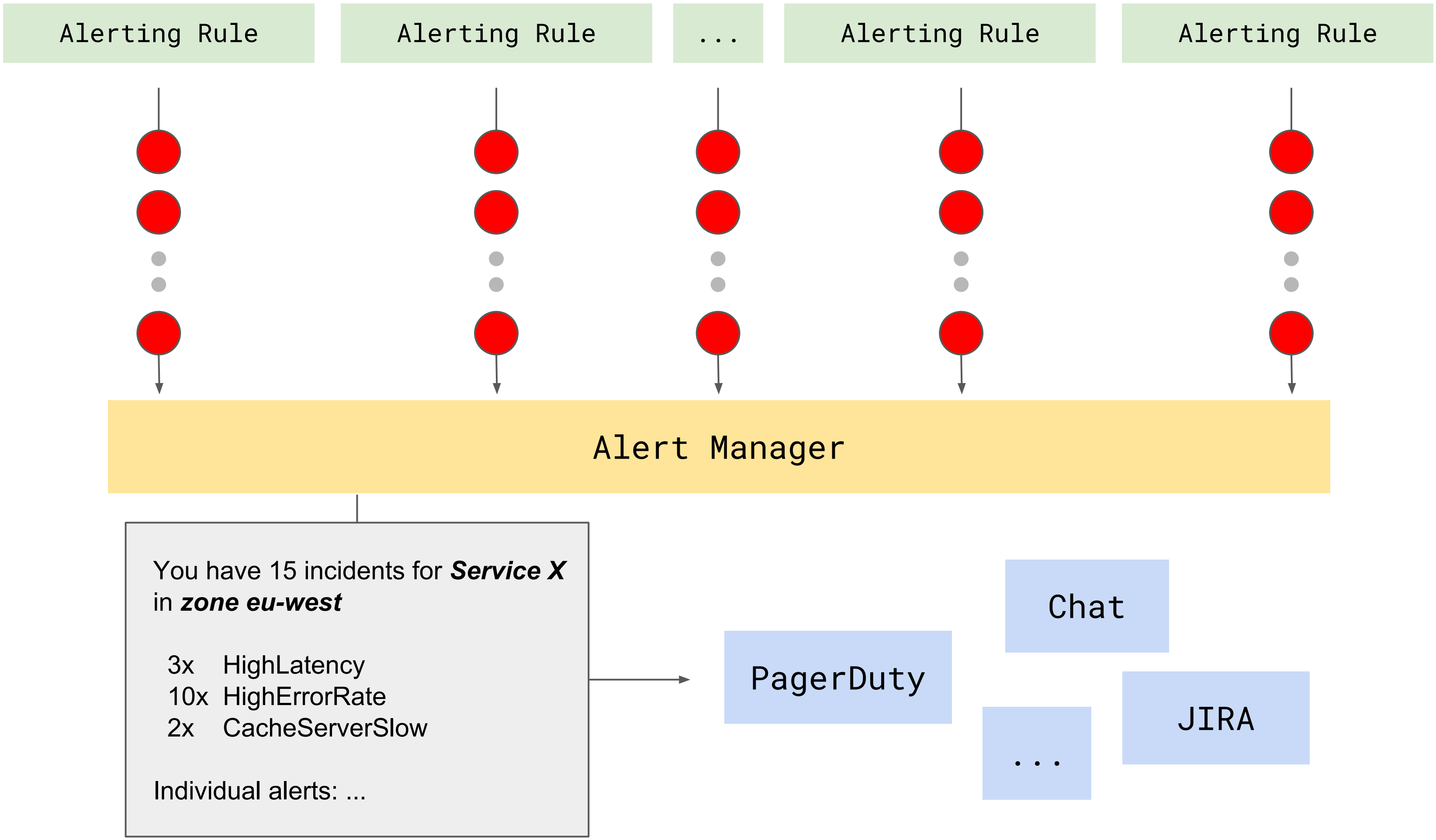
Alerting Rule

...

Alerting Rule

Alerting Rule

```
04:11   hey, HighLatency,   service=”X”, zone=”eu-west”, path=/user/profile, method=GET
04:11   hey, HighLatency,   service=”X”, zone=”eu-west”, path=/user/settings, method=GET
04:11   hey, HighLatency,   service=”X”, zone=”eu-west”, path=/user/settings, method=GET
04:11   hey, HighErrorRate, service=”X”, zone=”eu-west”, path=/user/settings, method=POST
04:12   hey, HighErrorRate, service=”X”, zone=”eu-west”, path=/user/profile, method=GET
04:13   hey, HighLatency,   service=”X”, zone=”eu-west”, path=/index, method=POST
04:13   hey, CacheServerSlow, service=”X”, zone=”eu-west”, path=/user/profile, method=POST
        . . .
04:15   hey, HighErrorRate, service=”X”, zone=”eu-west”, path=/comments, method=GET
04:15   hey, HighErrorRate, service=”X”, zone=”eu-west”, path=/user/profile, method=POST
```



Alerting Rule

Alerting Rule

...

Alerting Rule

Alerting Rule

Alert Manager

You have 15 incidents for ***Service X*** in ***zone eu-west***

3x HighLatency
10x HighErrorRate
2x CacheServerSlow

Individual alerts: ...

PagerDuty

Chat

...

JIRA

Inhibition

● **{alertname="DatacenterOnFire", severity="huge-page", zone="eu-west"}**

↓

```
if active,
mute everything else in same zone
```

● {alertname="LatencyHigh", severity="page", ..., zone="eu-west"}

...

● {alertname="LatencyHigh", severity="page", ..., zone="eu-west"}

● {alertname="ErrorsHigh", severity="page", ..., zone="eu-west"}

...

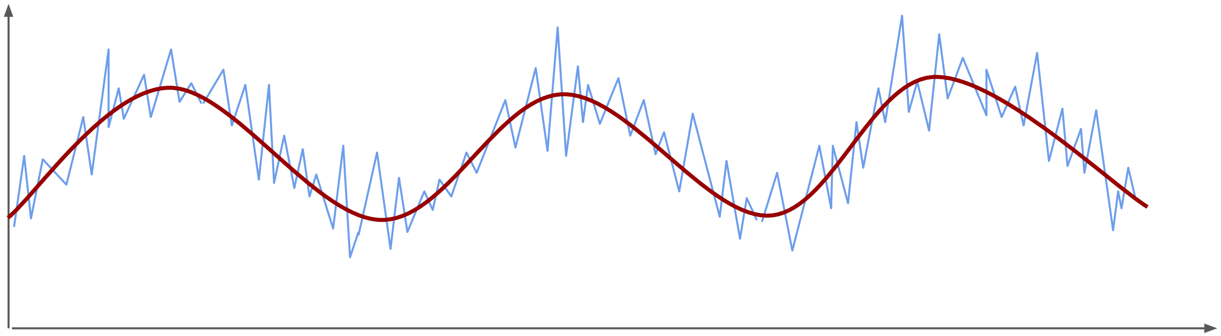
● {alertname="ServiceDown", severity="page", ..., zone="eu-west"}

Anomaly Detection

Practical Example

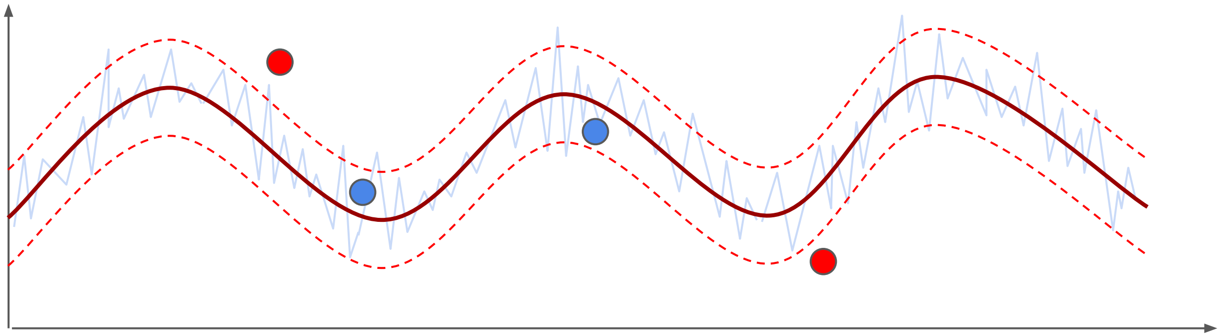
```
job:requests:rate5m = sum by(job) (rate(requests_total[5m]))

job:requests:holt_winters_rate1h = holt_winters(
    job:requests:rate5m[1h], 0.6, 0.4
)
```



Practical Example

```
ALERT AbnormalTraffic
IF abs(
     job:requests:rate5m - job:requests:holt_winters_rate1h offset 7d
   )
  >
   0.2 * job:request_rate:holt_winters_rate1h offset 7d
FOR 10m
...
```



Practical Example

```
  instance:latency_seconds:mean5m
> on (job) group_left()
  (
      avg by (job)(instance:latency_seconds:mean5m)
    + on (job)
      2 * stddev by (job)(instance:latency_seconds:mean5m)
  )
```

Practical Example

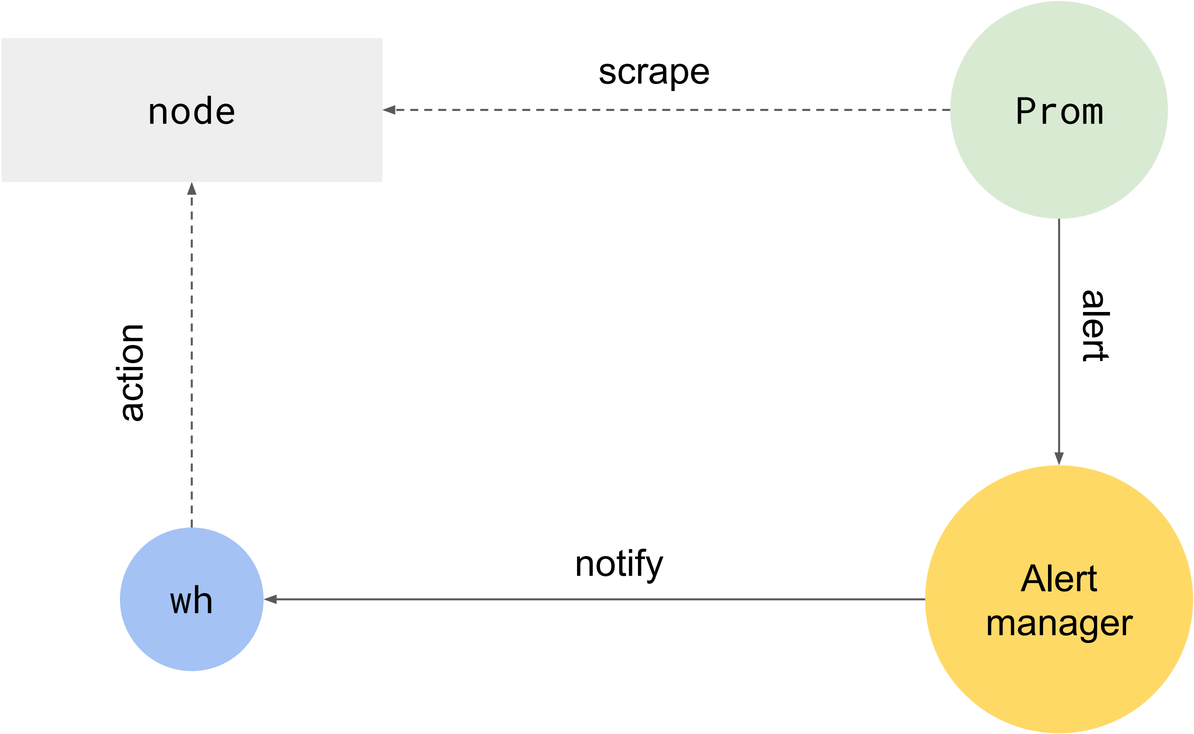
```
(
    instance:latency_seconds:mean5m
  > on (job) group_left()
    (
        avg by (job)(instance:latency_seconds:mean5m)
      + on (job)
        2 * stddev by (job)(instance:latency_seconds:mean5m)
    )
)
> on (job) group_left()
  1.2 * avg by (job)(instance:latency_seconds:mean5m)
```

Practical Example

```
(
    instance:latency_seconds:mean5m
  > on (job) group_left()
    (
        avg by (job)(instance:latency_seconds:mean5m)
      + on (job)
        2 * stddev by (job)(instance:latency_seconds:mean5m)
    )
)
> on (job) group_left()
  1.2 * avg by (job)(instance:latency_seconds:mean5m)
```

```
and on (job)
  avg by (job)(instance:latency_seconds_count:rate5m) > 1
```

Self Healing


node
scrape
Prom
action
alert
notify
wh
Alert
manager

Conclusion

- Symptom-based pages + cause based warnings provide good coverage and insight into service availability
  - Design alerts that are adaptive to change, preserve as many dimensions as possible, aggregate away dimensions of fault tolerance
  - Use linear prediction for capacity planning and saturation detection
- Advanced alerting expressions allow for well-scoped and practical anomaly detection
- Raw alerts are not meant for human consumption
- The Alertmanager aggregates, silences, and routes groups of alerts as meaningful notifications