PromCon 2019
Fun and profit with Alertmanager

Simon Pasquier (@SimonHiker), November 7, 2019

Who am I?

- Software engineer working at Red Hat
- Alertmanager & consul_exporter maintainer

Alerting craft

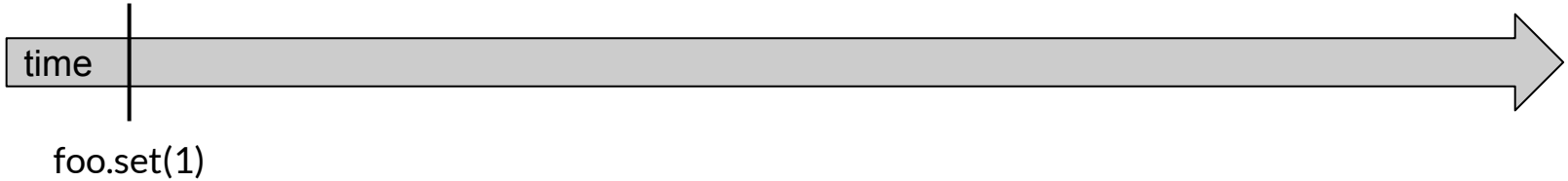
```
groups:
- name: example
  rules:
  - alert: HighRequestLatency
    expr: job:request_latency_seconds:mean5m{job="myjob"} > 0.5
    for: 10m
    labels:
      severity: page
    annotations:
      summary: High request latency
```

Guidelines

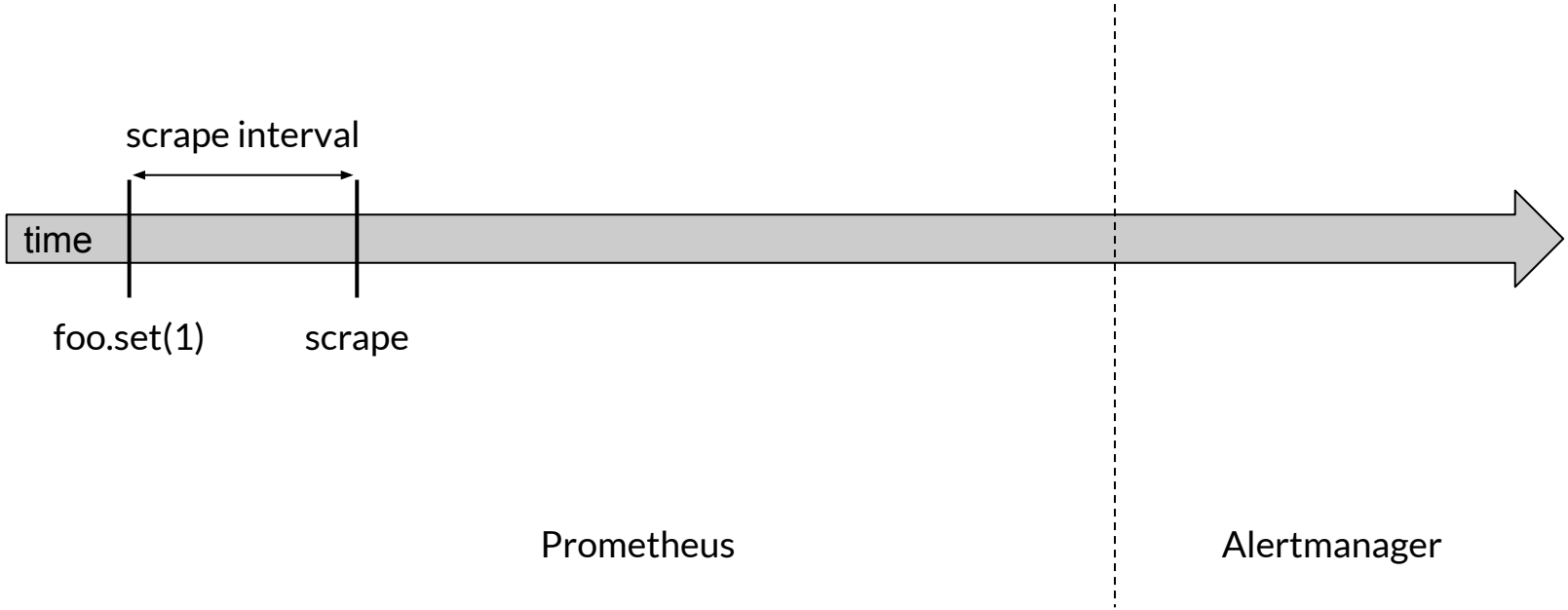
- Think about which labels to propagate.
- “Complex” alerts can be harmful.
- Spend some time to learn the template language.

When will I be notified that something's broken?

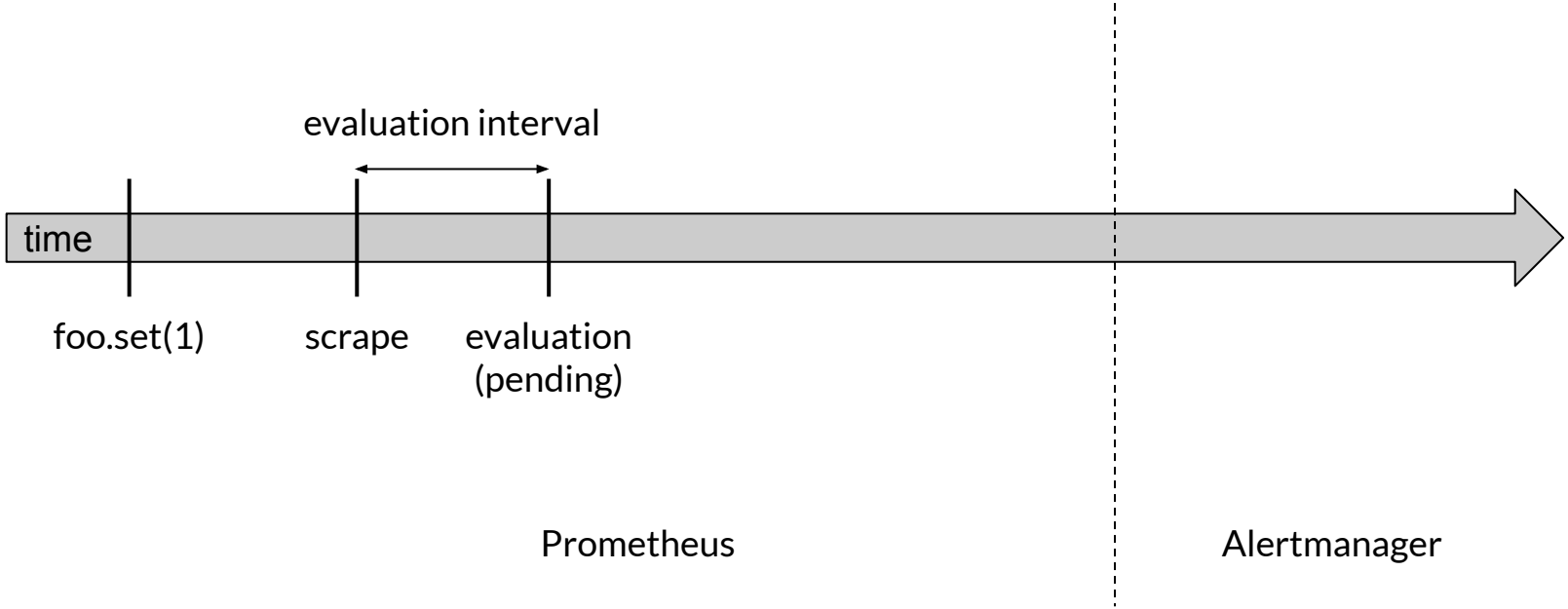
expr: foo > 0
for: 2m



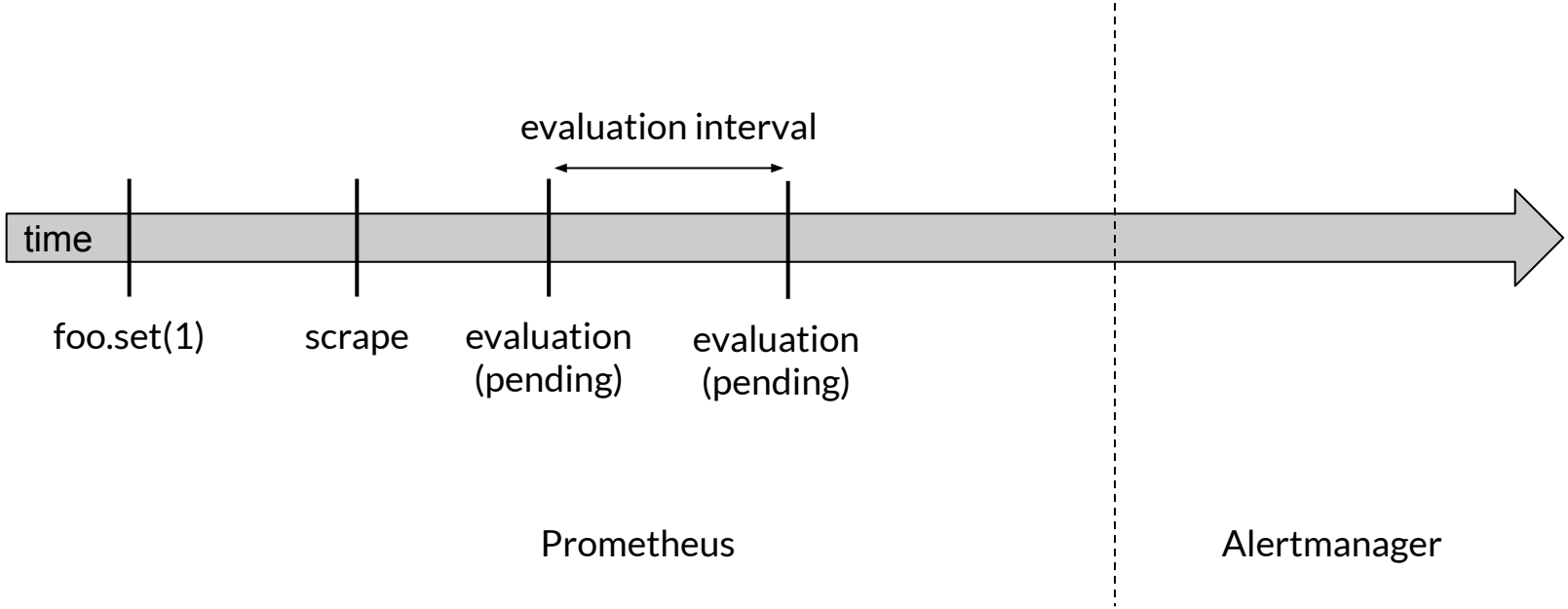
expr: foo > 0
for: 2m



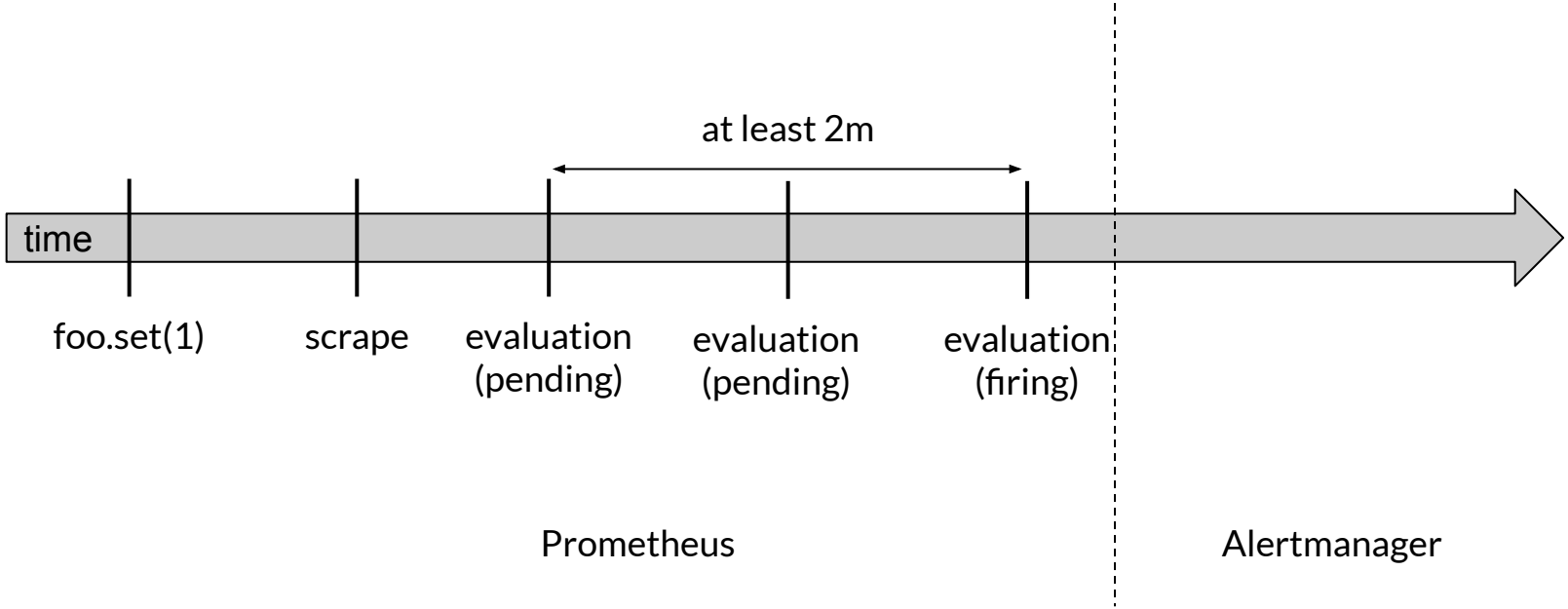
expr: foo > 0
for: 2m



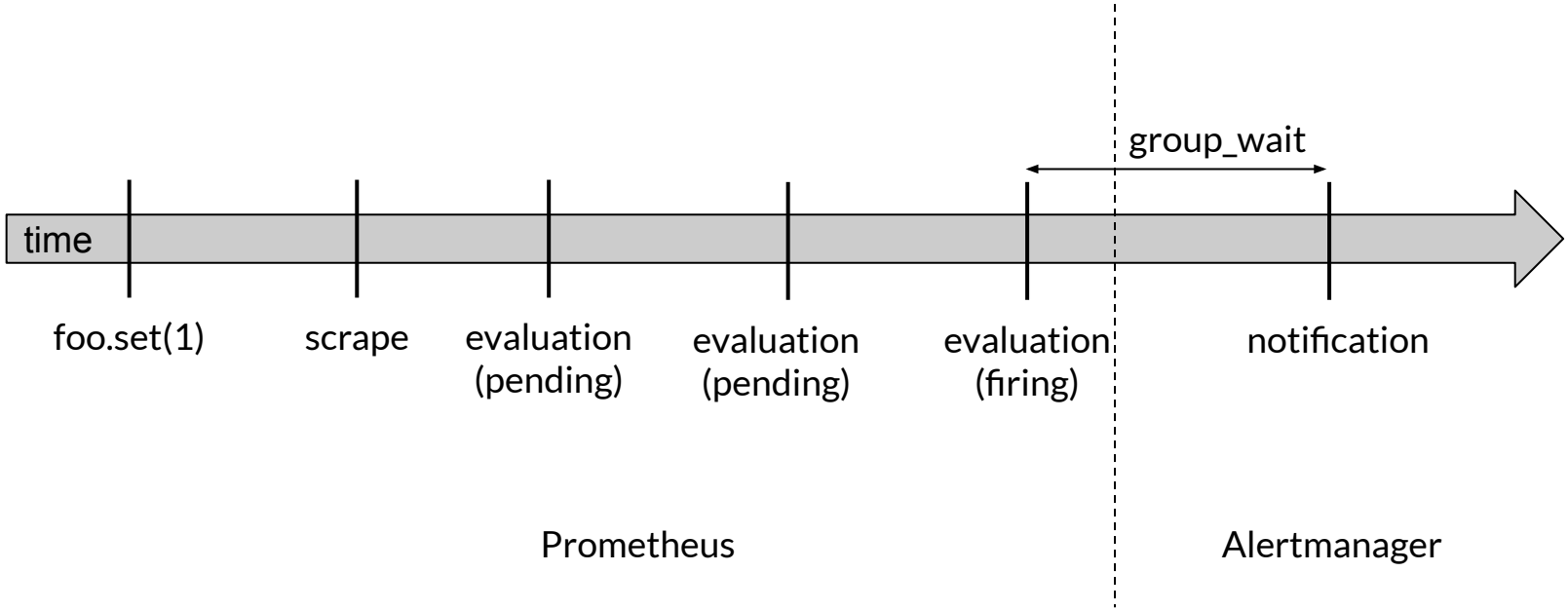
expr: foo > 0
for: 2m



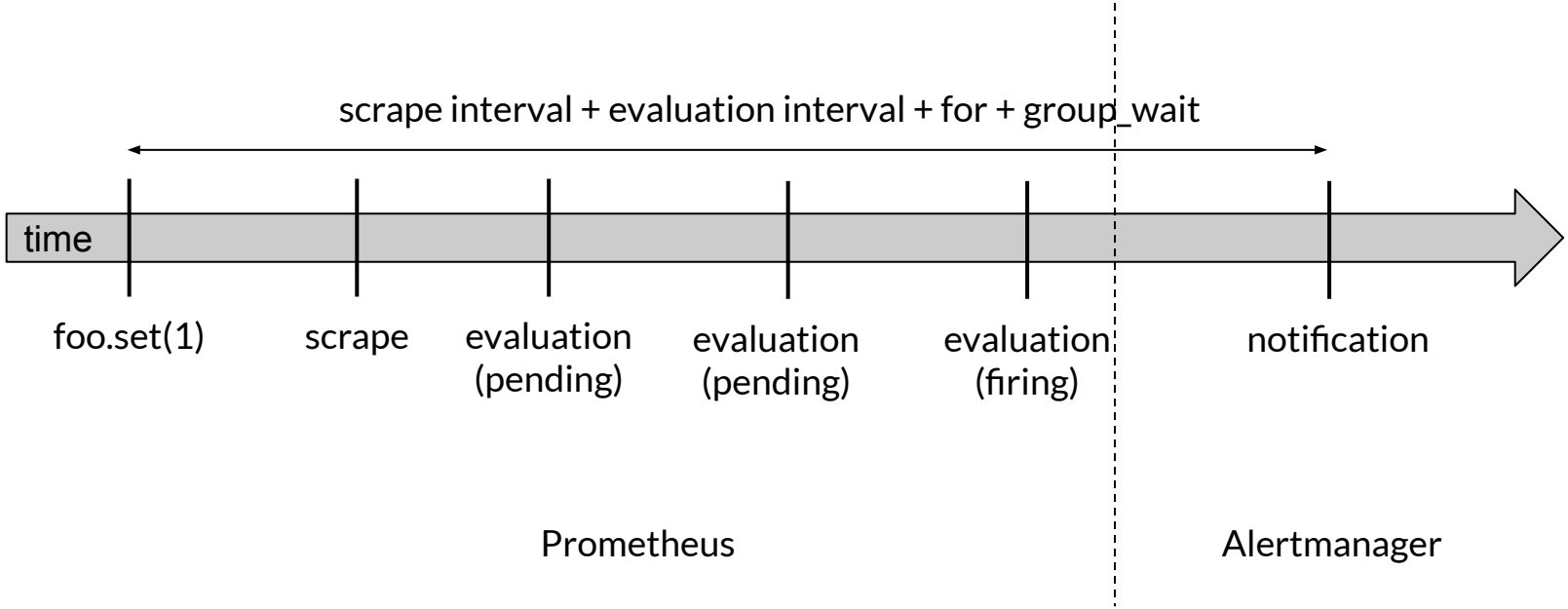
expr: foo > 0
for: 2m



expr: foo > 0
for: 2m



expr: foo > 0
for: 2m



Things to know

- Use “for” to avoid flapping alerts.
- group_interval for subsequent updates (including resolution).
- repeat_interval for reminders.
  - `--data.retention` flag (#1806).

Routing


Tim Simmons
@timsimlol
@PrometheusIO do you like my alertmanager
Tweeter
Traduire le Tweet
capacity-alerting-noir
pandora-alerts
alerts

Guidelines

- Keep it simple.
- First level routes to match services/teams.
- Use amtool or routing tree editor to test/validate.

To continue or not?

Fictional scenario:

- All notifications should go to Slack.
- Alerts with job=app should email the app team.
  - severity=critical should page the app team too.
- Alerts with severity=critical should page the ops team.

Silences, inhibitions, oh my!

Inhibition rule

```
route:
  group_by: [job]
  receiver: 'chat'
  routes:
  - match:
      alertname: "NodeDown"
    receiver: page

inhibit_rules:
- source_match:
    alertname: NodeDown
  target_match:
    alertname: TargetDown
  equal: [instance]
```

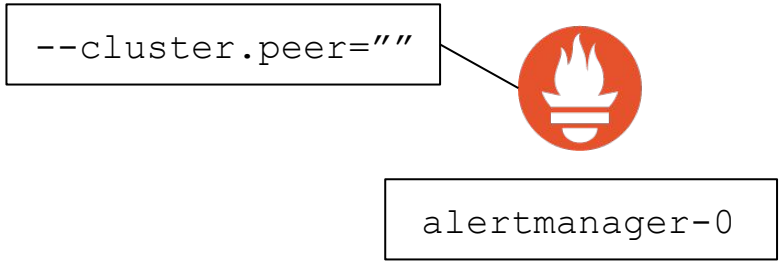
Gotchas

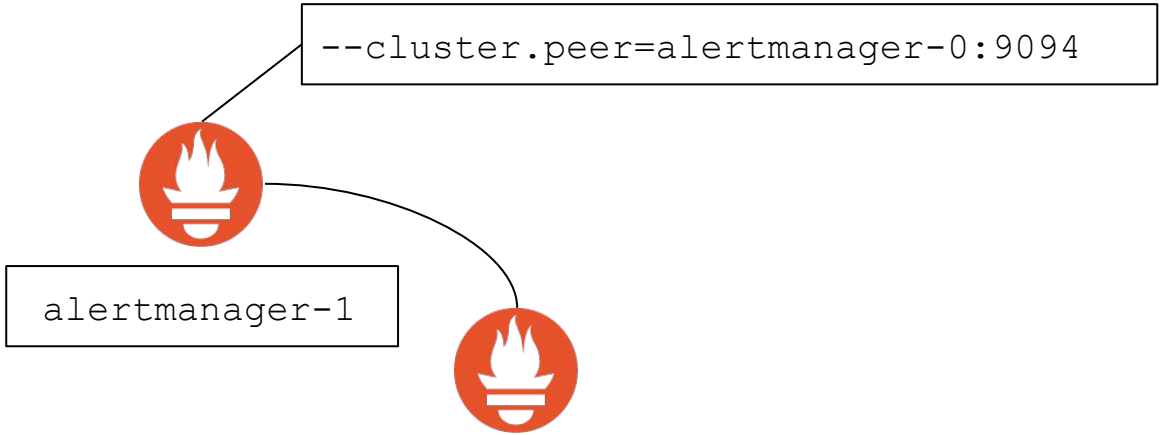
- Pick the appropriate silence duration (#1639).
- Corner cases with incident management systems.
- Inhibiting alerts can’t inhibit themselves (#666).

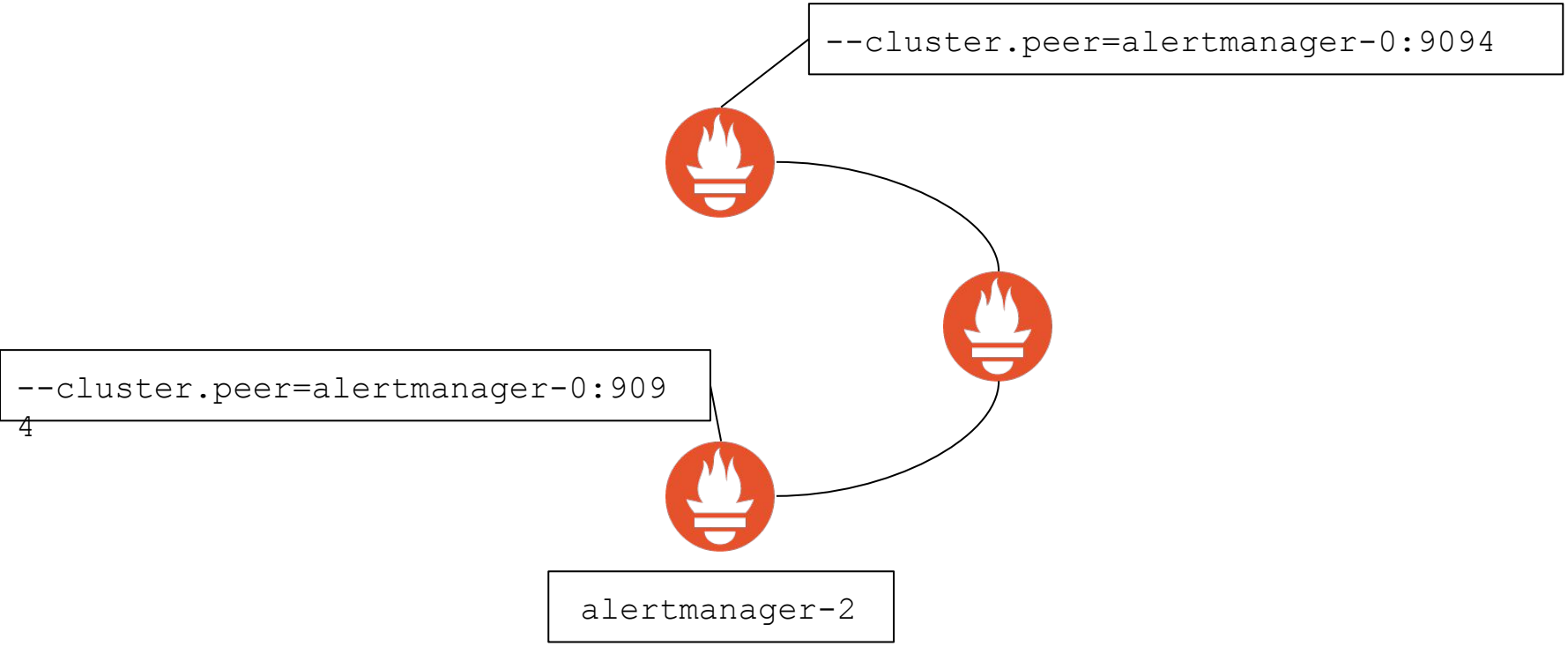
High availability

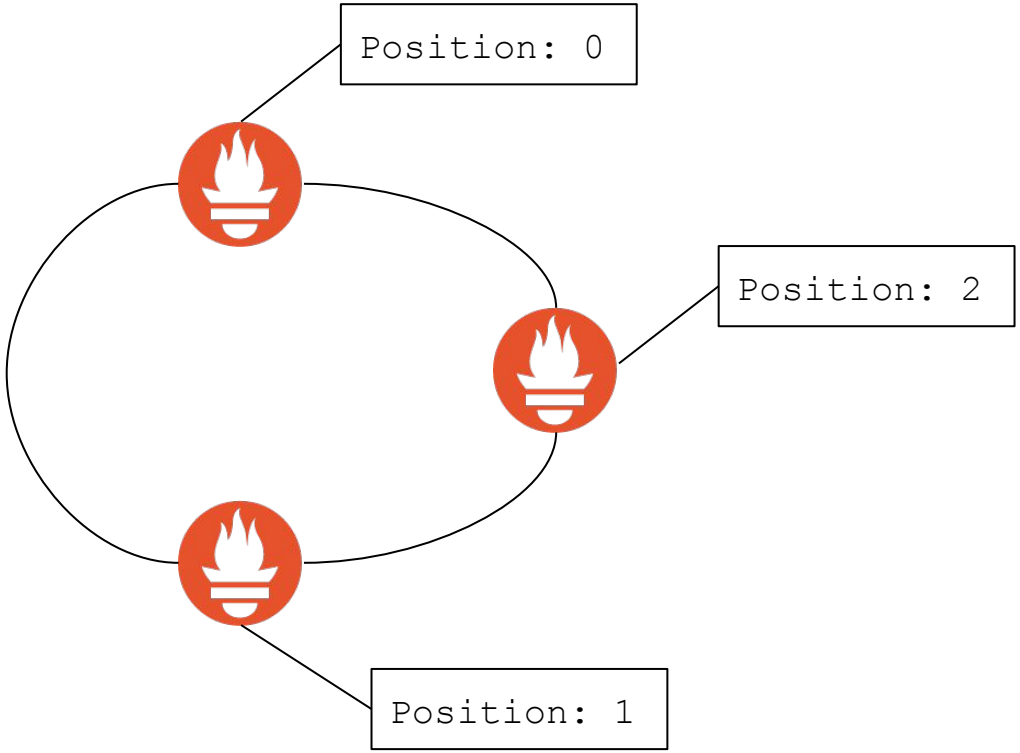
High availability

- Broadcast silences and notification logs.
- Based on the hashicorp/memberlist library.
- Requires a dedicated TCP/UDP port.
  - UDP for small messages (⩽ 700 bytes)
  - TCP otherwise


--cluster.peer=""
alertmanager-0


--cluster.peer=alertmanager-0:9094
alertmanager-1


--cluster.peer=alertmanager-0:9094
--cluster.peer=alertmanager-0:909
4
alertmanager-2


Position: 0
Position: 2
Position: 1

High availability flags

- Server
  --cluster.listen-address
  --cluster.advertise-address
- Peering
  --cluster.peer
  --cluster.peer-timeout (15s)
  --cluster.settle-timeout (1m)

High availability flags (continued)

- Data exchange
  --cluster.gossip-interval (250ms)
  --cluster.pushpull-interval (1m)
  --cluster.tcp-timeout (10s)
- Probes
  --cluster.probe-timeout (500ms)
  --cluster.probe-interval (1s)
- Reconnection
  --cluster.reconnect-interval (10s)
  --cluster.reconnect-timeout (6h)

Hidden stuff

- Peer names refreshed every 15 seconds.
- Messages gossiped to half of the nodes (min. 3).
- Gossip queue size of 4096 messages.
- Settle phase stops after 3 “stable” iterations.

Future work

- Encryption & authentication using mTLS (#1819).
- Better support for advertised address (#1909).

Conclusion

- Test all the things.
- Keep it simple.
- We ❤️ contributions!

Thanks!

Simon Pasquier
pasquier.simon@gmail.com
@SimonHiker

Psst, we're hiring!