\chapter{ Alerting System}

Today the most trending subject in monitoring area is a prediction and automated detection. It makes people free from 24/7 managing, maintaining and monitoring system.

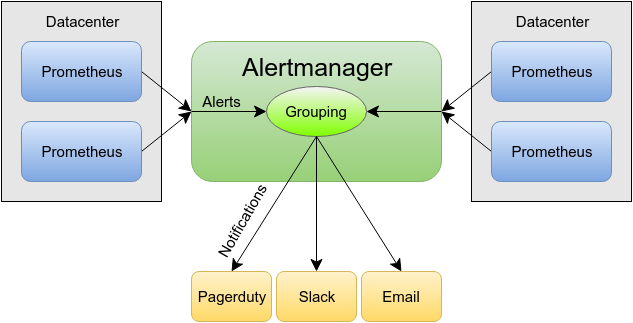
For monitoring we are using Prometheus tool. Since this tool saving constantly log data about system it is possible to reuse this logs to build an alert system.

Prometheus tool provides an Alert Manager module. Natively this tool supports different notification methods like email notification or some request on Slack.

\chapter{ Alerting System Architecture}

Since the Alert Manager is a part of Prometheus Tool it has its own binary. The idea behind is to have only one Alert Manager and have monitoring tool on multiple machines. If machine goes down or even Prometheus itself, the Alert Manager can catch and deliver this event.

To understand how Alert Manager works it is essential to understand the architecture of the whole system:

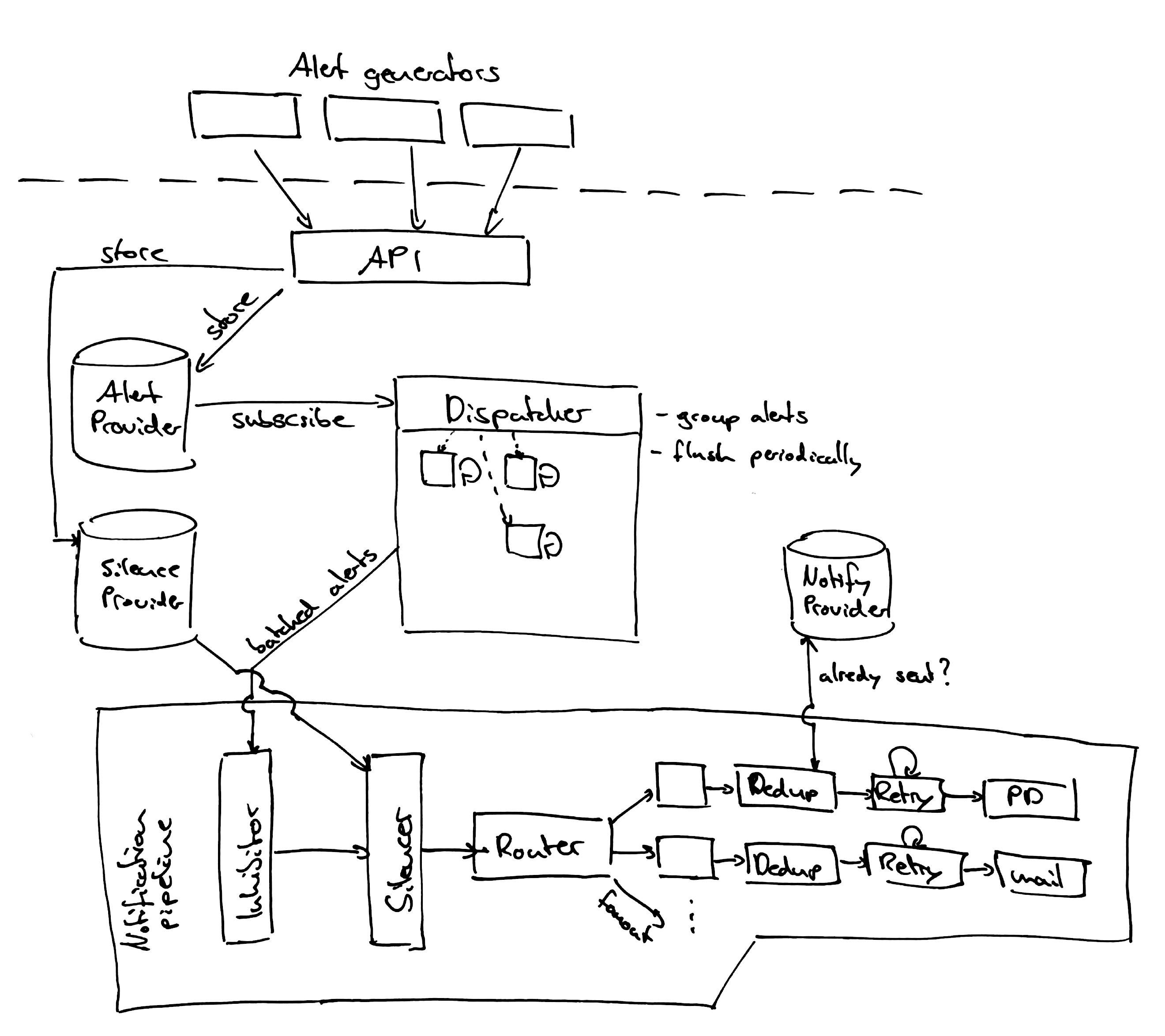


This architecture is a typical messaging platform.

Messaging Service send messages between multiple clients. It is implement Producer-Consumer Pattern. In the Alert Manager Architecture the role of producer is taken by Prometheus Datacenter. Alert Manager consumes the messages. Consumer knows nothing about producer and just subscribe on event. With this approach it is possible to attach multiple producers. [1]

Alerts can be collected in groups by datacenter. It means if an event occurs on multiple machines it can be packed into one notification and fired accordingly.

In Prometheus configuration need to be setup only two things: reference on Alert Manager and Alert Rules. When Alert Manager consume an event it just dispatch it via notification.



\chapter{ Alert Rules}

As it was mention one of the main changes in Prometheus setup is to configure the alert rules. Every Prometheus Monitoring Tool can have it own alerting rules, which can be defined. There is also a possibility to reference on some common alerting rule for every monitoring system on every machine.

Alerting rules are instructions to the monitoring system it can be user as well for alerting as for recording.

Recording rules allows to pre-compute frequently needed expressions or expressions, which are resource or time consuming. These rules are saving result in a new set of time series. It is like indexing this data, so that prevent expansive I/O methods.

The rules are being executed sequentially with a predefined interval.

With a alerting rules it is possible to define alert namely deviation by particular expression from Prometheus Tool and its exports (modules). It allows building an alert even on combined query.

In case if Alert Manager is not available all alerts are saving into buffer. As soon Alert Manager online all events will be fired sequentially.

\chapter{N2Sky Integration}

Alerting System is represented as a module in N2Sky.

Alerting Client is the additional configuration upon Prometheus Monitoring System. When Prometheus will be executed it should have reference on Alerting System and its rules.

Since the client should be installed on each OpenStack instance it was integrated into image snapshot. When the new instance will be spawned with a OpenStack snapshot, the client will be automatically executed there.

Alerting Client fire alerts depending on configured rules. The rules can be created via user interface. Every alert has its severity level. Depending on it the fired events will be represented differently:

* Saverity

[1] Strategies for Integrating Messaging and Distributed Object Transactions https://link.springer.com/chapter/10.1007/3-540-45559-0\_16