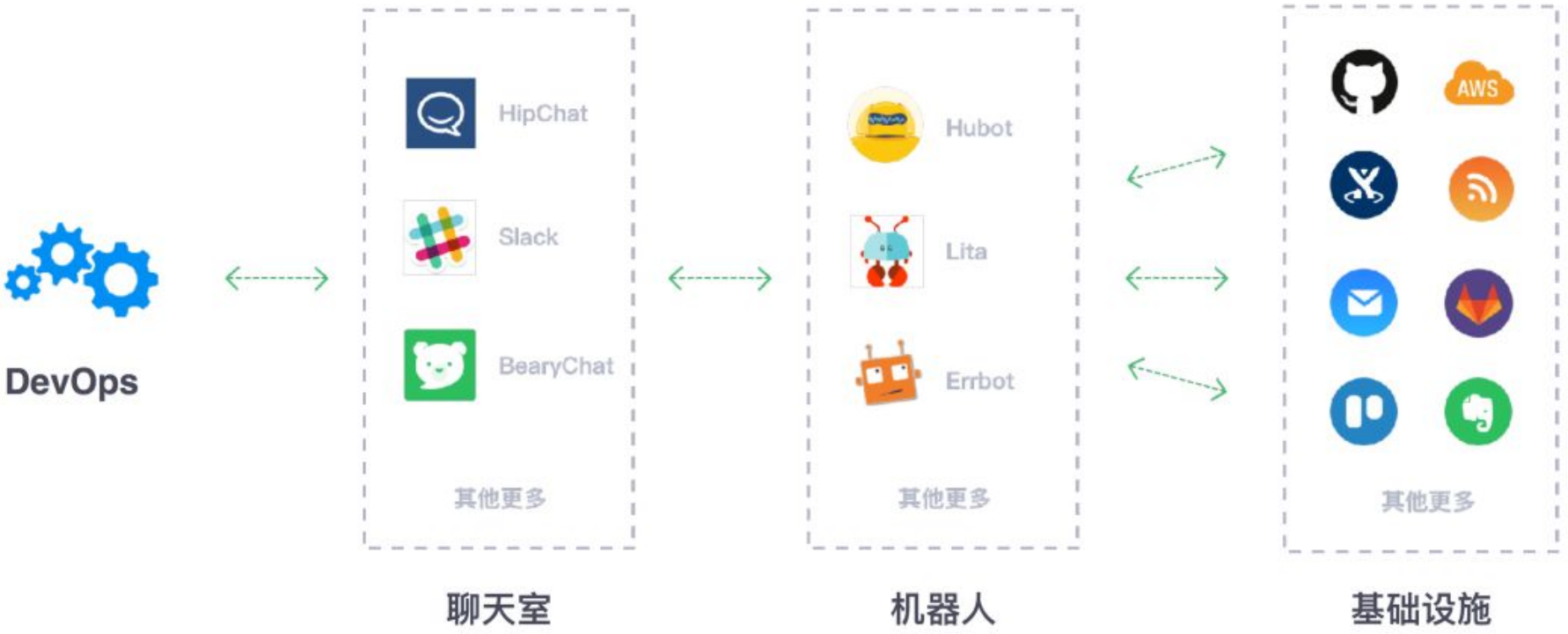
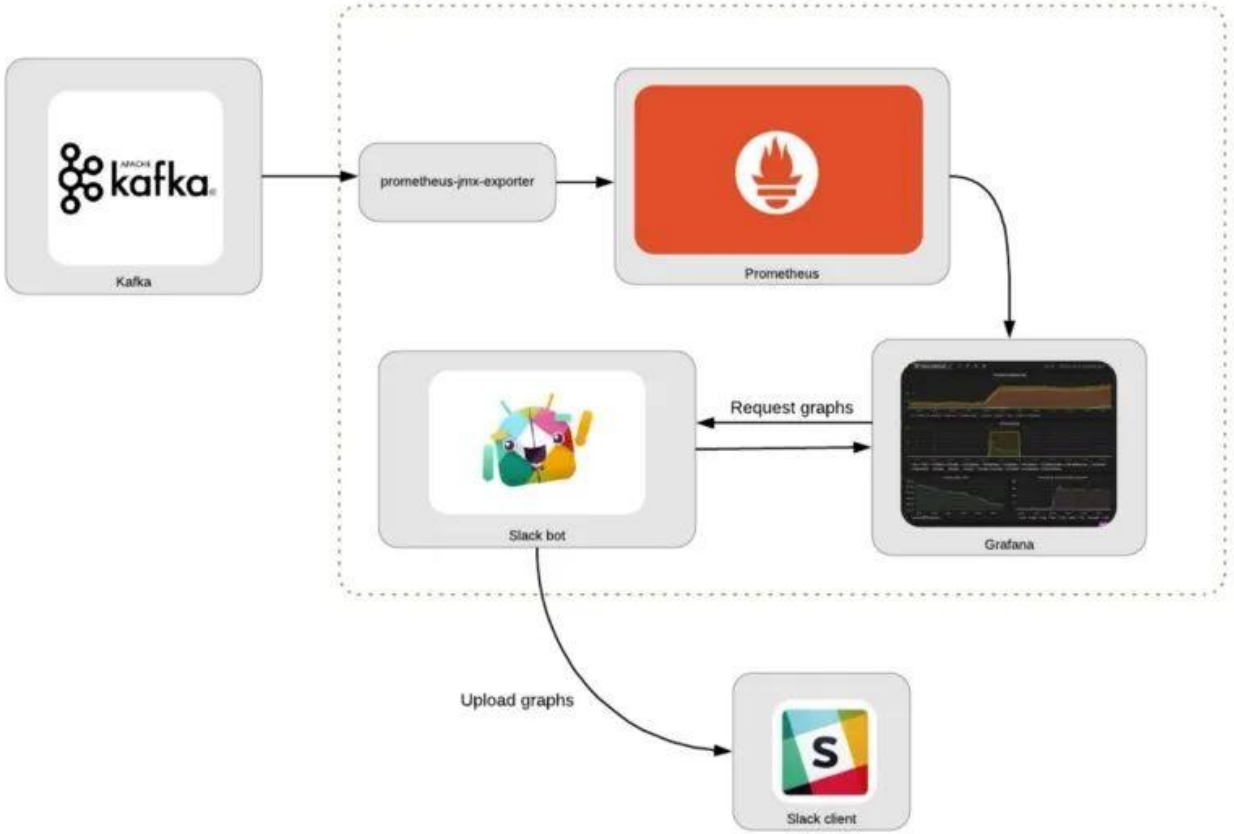
Prometheus告警

11/13/22

Alerting demo架构



Alerting demo架构 - bot



Debug - Slack Bot Token

- Create a slack bot: https://api.slack.com/bot-users

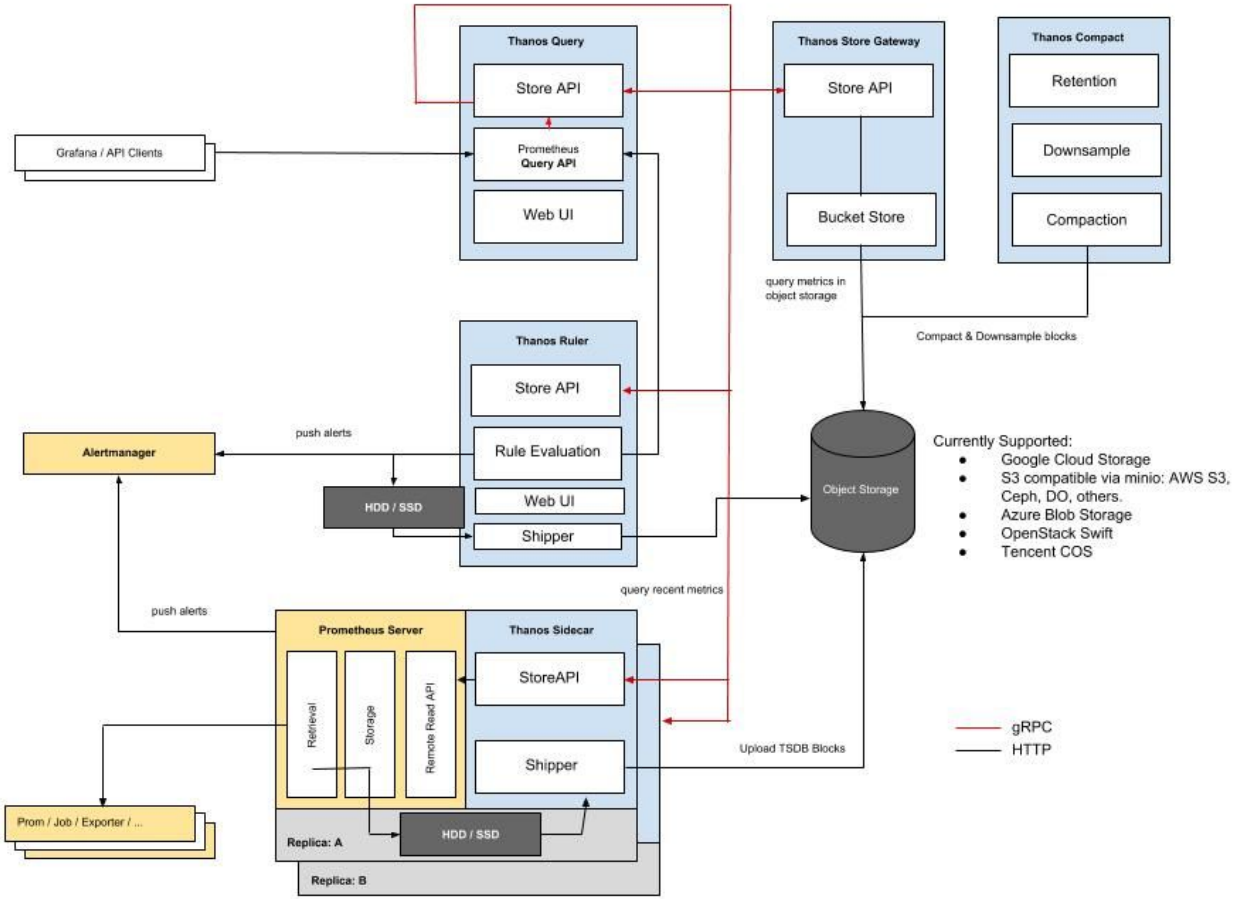
Prometheus单个节点的问题

数据量增加，业务量变大 -> 水平的业务切割 -> 问题：

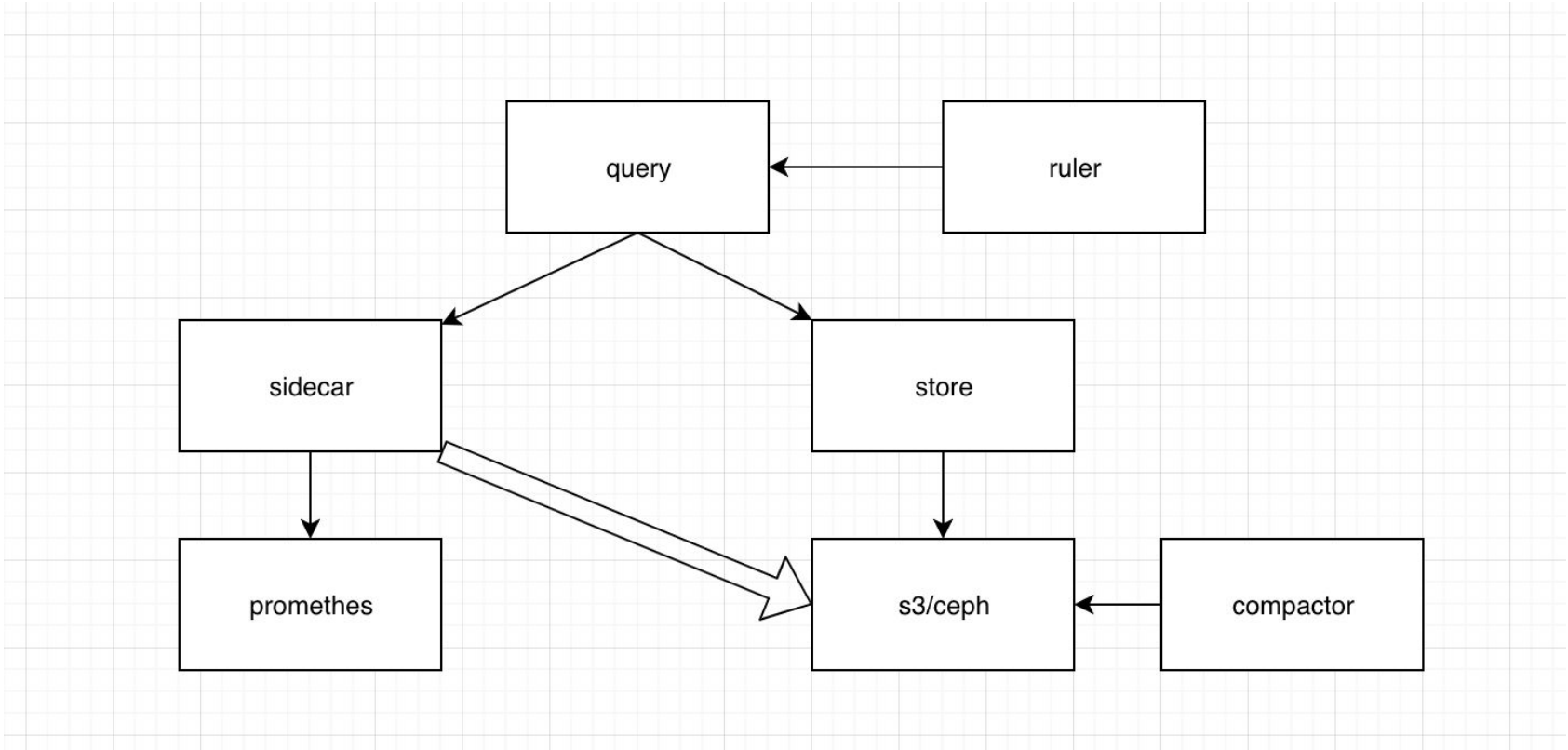
1. 查询时间有限。prometheus数据保存在本地节点是有限的空间，不可能查询特别久的时间，主要是本地结点的磁盘有限。
2. 查询范围有限。prometheus只能查询自己抓取的数据，当数据是由两个prometheus nodes分别抓取的时候，这样就无法选出两个集群的综合指标，例如top k这种，也无法跨越prometheus进行聚合操作。
3. 告警范围有限。告警这个查询是一个问题，prometheus的告警是通过定时round robin做的，查不到（不同节点间）的情况无法进行告警。

Thanos架构图

CNCF



Thanos架构图 - 简化



sidecar

抓取依旧是prometheus来做，多了一个sidecar组件，作用有两个，一个是代理prometheus提供查询，一个是把prometheus已经合成块的数据搬入其他高可用文件存储系统，例如上面的s3,ceph等。

sidecar把数据存入ceph等文件存储系统，就解决了文件存储的问题，数据可以保存的更久。

Store

文件已经存入了其他文件存储系统，那么需要提供查询的功能，这里有store组件来做，可以设置时间切面，store组件用来查询高可用文件系统中的数据。这样解决了存储和存储查询的问题了。

Query

query是查询的分发和数据的合并组件。负责和查询的请求发给所有的store和sidecar，然后各个组件会把查询结果返回给query。最终query提供了查询结果出去。这里需要注意的是发送给所有的store和sidecar。这里就解决了多个prometheus抓取后查询不到的问题。现在query会把多个prometheus的数据进行合并。

Ruler

ruler组件是用来做全局告警，解决的就是上面提出的prometheus的告警问题。ruler基于query进行查询。

Compactor

prometheus本身是有压缩的功能的，在和thanos配合的时候必须关闭。由compactor组件进行压缩和下采样。以此节省一部分存储，并且通过下采样提供更快捷的查询能力。