Prometheus+Gafana 全方位立体式监控系统

讲师介绍

阿良

资深运维工程师，51CTO知名博主。曾就职在IDC，大数据，金融行业，现任职奇虎360公司负责PC浏览器业务。经重重磨炼，具有丰富的运维实战经验。

技术博客：http://blog.51cto.com/lizhenliang

阿良微信

Docker/K8s技术学员群：397834690

课程目录

一 聊聊监控

二 Prometheus概述

三 Prometheus部署

四 配置文件及核心功能

五 监控案例

六 告警神器 Alertmanager

七 全方位监控Kubernetes资源与应用

第 1 章 聊聊监控

1. 为什么要监控
2. 怎么来监控
3. 要监控什么
4. 准备工作

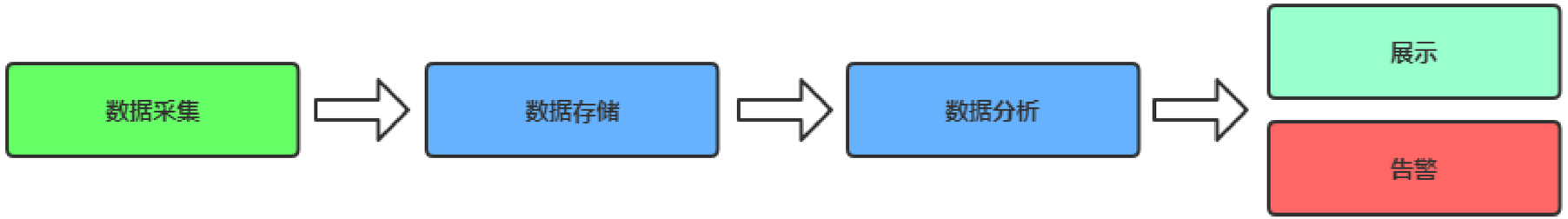
为什么要监控

- 对系统不间断实时监控
- 实时反馈系统当前状态
- 保证业务持续性运行

怎么来监控

- 监控工具
  - free
  - vmstat
  - df
  - top
  - ss
  - iftop
  - ...
- 监控系统
  - Zabbix
  - Open-Falcon
  - Prometheus

怎么来监控



准备工作

- 熟悉被监控对象
- 整理监控指标
- 告警阈值定义
- 故障处理流程

要监控什么

硬件监控	温度，硬件故障等
系统监控	CPU，内存，硬盘，网卡流量，TCP状态，进程数
应用监控	Nginx、Tomcat、PHP、MySQL、Redis等
日志监控	系统日志、服务日志、访问日志、错误日志
安全监控	WAF，敏感文件监控
API监控	可用性，接口请求，响应时间
业务监控	例如电商网站，每分钟产生多少订单、注册多少用户、多少活跃用户、推广活动效果
流量分析	根据流量获取用户相关信息，例如用户地理位置、某页面访问状况、页面停留时间等

第 2 章 Prometheus概述

1. Prometheus是什么
2. Prometheus组成及架构
3. 数据模型
4. 指标类型
5. 作业和实例

Prometheus 是什么

Prometheus（普罗米修斯）是一个最初在SoundCloud上构建的监控系统。自2012年成为社区开源项目，拥有非常活跃的开发人员和用户社区。为强调开源及独立维护，Prometheus于2016年加入云原生云计算基金会（CNCF），成为继Kubernetes之后的第二个托管项目。

https://prometheus.io

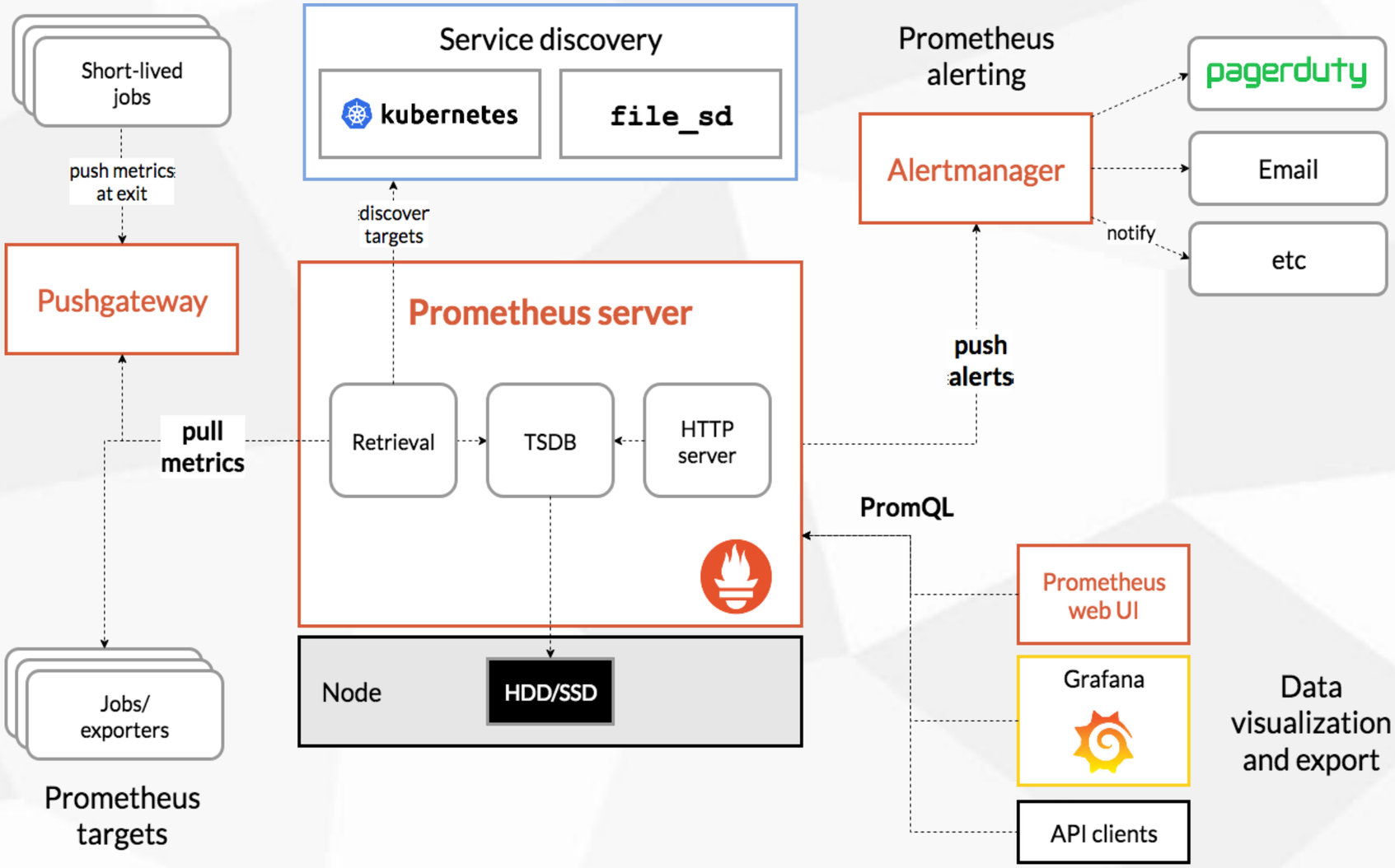
https://github.com/prometheus

Prometheus 是什么

Prometheus 特点：

- 多维数据模型：由度量名称和键值对标识的时间序列数据
- PromSQL：一种灵活的查询语言，可以利用多维数据完成复杂的查询
- 不依赖分布式存储，单个服务器节点可直接工作
- 基于HTTP的pull方式采集时间序列数据
- 推送时间序列数据通过PushGateway组件支持
- 通过服务发现或静态配置发现目标
- 多种图形模式及仪表盘支持（grafana）

Prometheus 组成及架构



Prometheus 组成及架构

- Prometheus Server：收集指标和存储时间序列数据，并提供查询接口
- ClientLibrary：客户端库
- Push Gateway：短期存储指标数据。主要用于临时性的任务
- Exporters：采集已有的第三方服务监控指标并暴露metrics
- Alertmanager：告警
- Web UI：简单的Web控制台

数据模型

Prometheus将所有数据存储为时间序列；具有相同度量名称以及标签属于同一个指标。

每个时间序列都由度量标准名称和一组键值对（也成为标签）唯一标识。

时间序列格式：

<metric name>{<label name>=<label value>, ...}

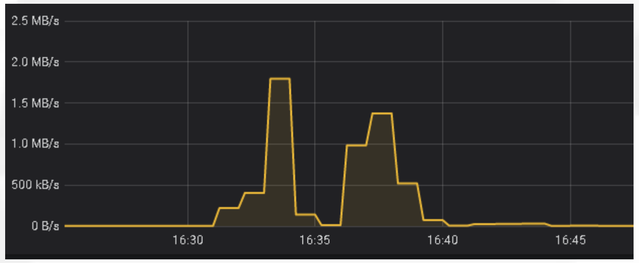
示例：api_http_requests_total{method="POST", handler="/messages"}

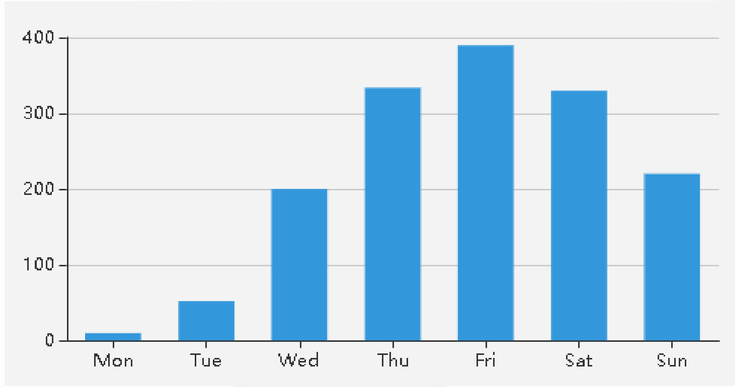
指标类型

- Counter：递增的计数器
- Gauge：可以任意变化的数值
- Histogram：对一段时间范围内数据进行采样，并对所有数值求和与统计数量
- Summary：与Histogram类似

指标类型


22:31:00
22:31:10
22:31:20
22:31:30
22:31:40
22:31:50
22:32:00
22:32:10
22:32:20
22:32:30


2.5 MB/s
2.0 MB/s
1.5 MB/s
1.0 MB/s
500 kB/s
0 B/s
16:30
16:35
16:40
16:45


400
300
200
100
0
Mon
Tue
Wed
Thu
Fri
Sat
Sun

作业和实例

实例：可以抓取的目标称为实例（Instances）

作业：具有相同目标的实例集合称为作业（Job）

```
scrape_configs:
  - job_name: 'prometheus'
    static_configs:
    - targets: ['localhost:9090']
  - job_name: 'node'
    static_configs:
    - targets: ['192.168.1.10:9090']
```

第 3 章 Prometheus 部署

1. 二进制部署
2. Docker部署
3. Web控制台
4. 配置Prometheus监控本身

Prometheus 部署

二进制部署：https://prometheus.io/docs/prometheus/latest/getting_started/

Docker部署：https://prometheus.io/docs/prometheus/latest/installation/

访问Web：http://localhost:9090

配置Prometheus监控本身：

```
scrape_configs:
  - job_name: 'prometheus'
    scrape_interval: 5s
    static_configs:
      - targets: ['localhost:9090']
```

第 4 章 配置文件与核心功能

1. 全局配置文件
2. scrape_configs
3. relabel_configs
4. 基于文件的服务发现

全局配置文件

```
global:
  [ scrape_interval: <duration> | default = 1m ]
  [ scrape_timeout: <duration> | default = 10s ]
  [ evaluation_interval: <duration> | default = 1m ]
  external_labels:
    [ <labelname>: <labelvalue> ... ]

rule_files:
  [ - <filepath_glob> ... ]

scrape_configs:
  [ - <scrape_config> ... ]

alerting:
  alert_relabel_configs:
    [ - <relabel_config> ... ]
  alertmanagers:
    [ - <alertmanager_config> ... ]

remote_write:
  [ - <remote_write> ... ]
remote_read:
  [ - <remote_read> ... ]
```

https://prometheus.io/docs/prometheus/latest/configuration/configuration/

scrape_configs


```
job_name: <job_name>

[ scrape_interval: <duration> | default = <global_config.scrape_interval> ]
[ scrape_timeout: <duration> | default = <global_config.scrape_timeout> ]
[ metrics_path: <path> | default = /metrics ]
[ honor_labels: <boolean> | default = false ]

[ scheme: <scheme> | default = http ]
params:
  [ <string>: [<string>, ...] ]
basic_auth:
  [ username: <string> ]
  [ password: <secret> ]
  [ password_file: <string> ]
[ bearer_token: <secret> ]
[ bearer_token_file: /path/to/bearer/token/file ]
tls_config:
  [ <tls_config> ]
[ proxy_url: <string> ]

consul_sd_configs:
  [ - <consul_sd_config> ... ]
dns_sd_configs:
  [ - <dns_sd_config> ... ]
file_sd_configs:
  [ - <file_sd_config> ... ]
kubernetes_sd_configs:
  [ - <kubernetes_sd_config> ... ]
...

static_configs:
  [ - <static_config> ... ]
relabel_configs:
  [ - <relabel_config> ... ]
metric_relabel_configs:
  [ - <relabel_config> ... ]

[ sample_limit: <int> | default = 0 ]
```

relabel_configs

relabel_configs ： 允许在采集之前对任何目标及其标签进行修改

重新标签的意义？

- 重命名标签名
- 删除标签
- 过滤目标

relabel_configs

```
relabel_configs:
  # 源标签
  [ source_labels: '[' <labelname> [, ...] ']' ]
  # 多个源标签时连接的分隔符
  [ separator: <string> | default = ; ]
  # 重新标记的标签
  [ target_label: <labelname> ]
  # 正则表达式匹配源标签的值
  [ regex: <regex> | default = (.*) ]
  #
  [ modulus: <uint64> ]
  # 替换正则表达式匹配到的分组，分组引用 $1,$2,$3,...
  [ replacement: <string> | default = $1 ]
  # 基于正则表达式匹配执行的操作
  [ action: <relabel_action> | default = replace ]
```

relabel_configs

action：重新标签动作

- replace：默认，通过regex匹配source_label的值，使用replacement来引用表达式匹配的分组
- keep：删除regex与连接不匹配的目标 source_labels
- drop：删除regex与连接匹配的目标 source_labels
- labeldrop：删除regex匹配的标签
- labelkeep：删除regex不匹配的标签
- hashmod：设置target_label为modulus连接的哈希值source_labels
- labelmap：匹配regex所有标签名称。然后复制匹配标签的值进行分组，replacement分组引用（${1},${2},...）替代

基于文件的服务发现

支持服务发现的来源：

- azure_sd_configs
- consul_sd_configs
- dns_sd_configs
- ec2_sd_configs
- openstack_sd_configs
- file_sd_configs
- gce_sd_configs
- kubernetes_sd_configs
- marathon_sd_configs
- nerve_sd_configs
- serverset_sd_configs
- triton_sd_configs

第 5 章 监控案例

1. 监控Linux服务器
2. 监控CPU，内存，硬盘
3. 监控服务状态
4. 使用Grafana炫图展示监控数据
5. 监控Docker服务器
6. 监控MySQL服务器

监控Linux服务器

node_exporter：用于*NIX系统监控，使用Go语言编写的收集器。

使用文档：https://prometheus.io/docs/guides/node-exporter/
GitHub：https://github.com/prometheus/node_exporter
exporter列表：https://prometheus.io/docs/instrumenting/exporters/

监控CPU，内存，硬盘

CPU使用率：

100 - (avg(irate(node_cpu_seconds_total{mode="idle"}[5m])) by (instance) * 100)

内存使用率：

100 - (node_memory_MemFree_bytes+node_memory_Cached_bytes+node_memory_Buffers_bytes) / node_memory_MemTotal_bytes * 100

磁盘使用率：

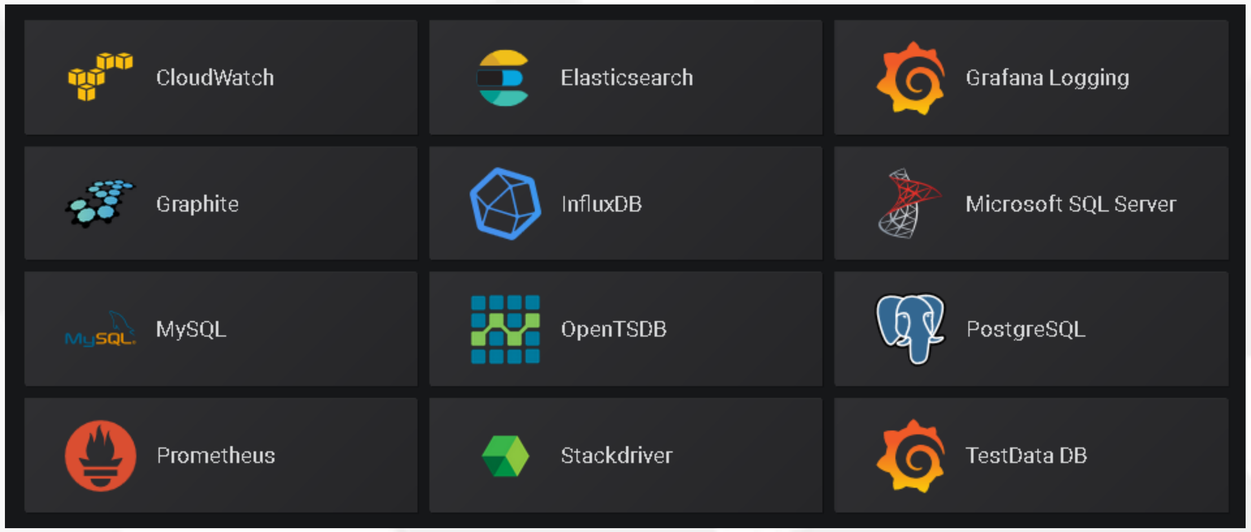
100 - (node_filesystem_free_bytes{mountpoint="/",fstype=~"ext4|xfs"} / node_filesystem_size_bytes{mountpoint="/",fstype=~"ext4|xfs"} * 100)

监控服务运行状态

node_exporter --collector.systemd --collector.systemd.unit-whitelist=(docker|sshd|nginx).service

Grafana炫图展示监控数据

Grafana是一个开源的度量分析和可视化系统。



https://grafana.com/grafana/download

https://grafana.com/dashboards/9276

监控Docker服务器

cAdvisor（Container Advisor）用于收集正在运行的容器资源使用和性能信息。

https://github.com/google/cadvisor

https://grafana.com/dashboards/193

监控MySQL服务器

mysql_exporter：用于收集MySQL性能信息。

https://github.com/prometheus/mysqld_exporter

https://grafana.com/dashboards/7362

第 6 章 告警

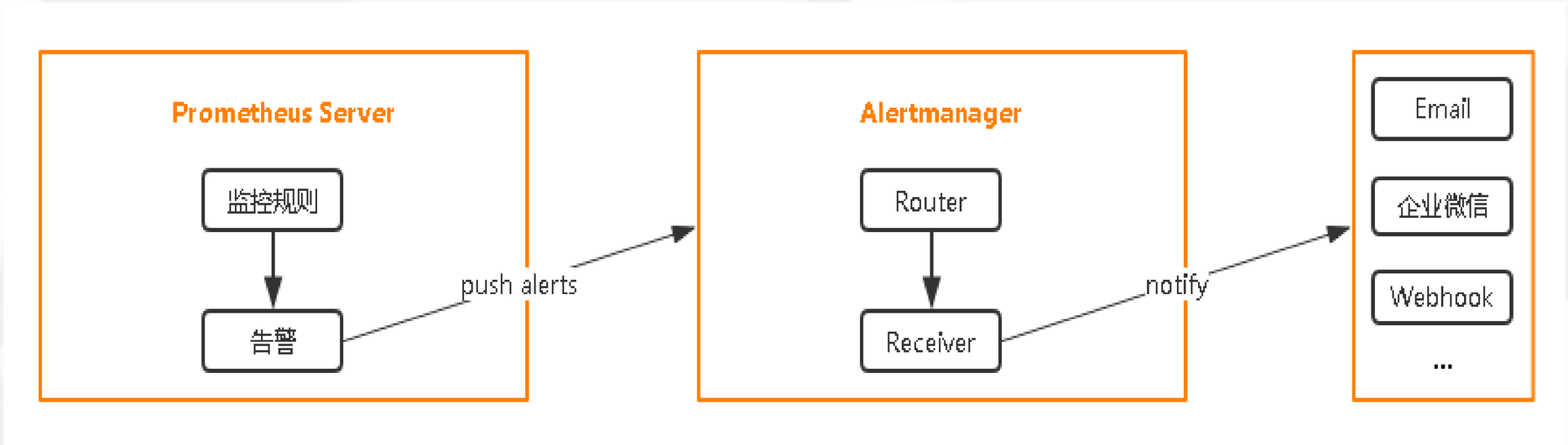
1. 部署Alertmanager
2. 配置Prometheus与Alertmanager通信
3. 在Prometheus中创建告警规则
4. 告警状态
5. 告警分配
6. 告警收敛（分组，抑制，静默）
7. Prometheus一条告警怎么触发的?
8. 编写告警规则案例

部署Alertmanager

地址1：https://prometheus.io/download/
地址2：https://github.com/prometheus/alertmanager/releases

1. 部署Alertmanager
2. 配置Prometheus与Alertmanager通信
3. 在Prometheus中创建告警规则

配置Prometheus与Alertmanager通信



在Prometheus中创建告警规则

示例：

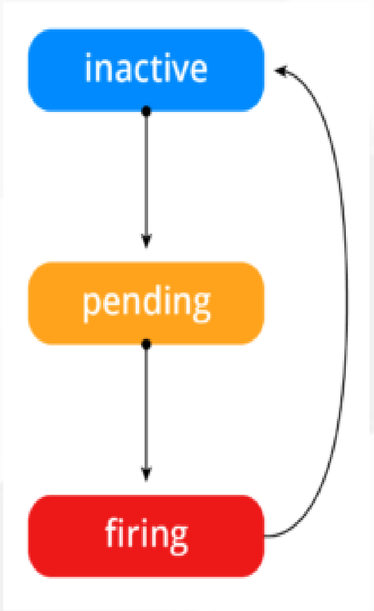
```
groups:
- name: example
  rules:

  # Alert for any instance that is unreachable for >5 minutes.
  - alert: InstanceDown
    expr: up == 0
    for: 5m
    labels:
      severity: page
    annotations:
      summary: "Instance {{ $labels.instance }} down"
      description: "{{ $labels.instance }} of job {{ $labels.job }} has been down for more than 5 minutes."
```

https://prometheus.io/docs/prometheus/latest/configuration/alerting_rules/

告警状态

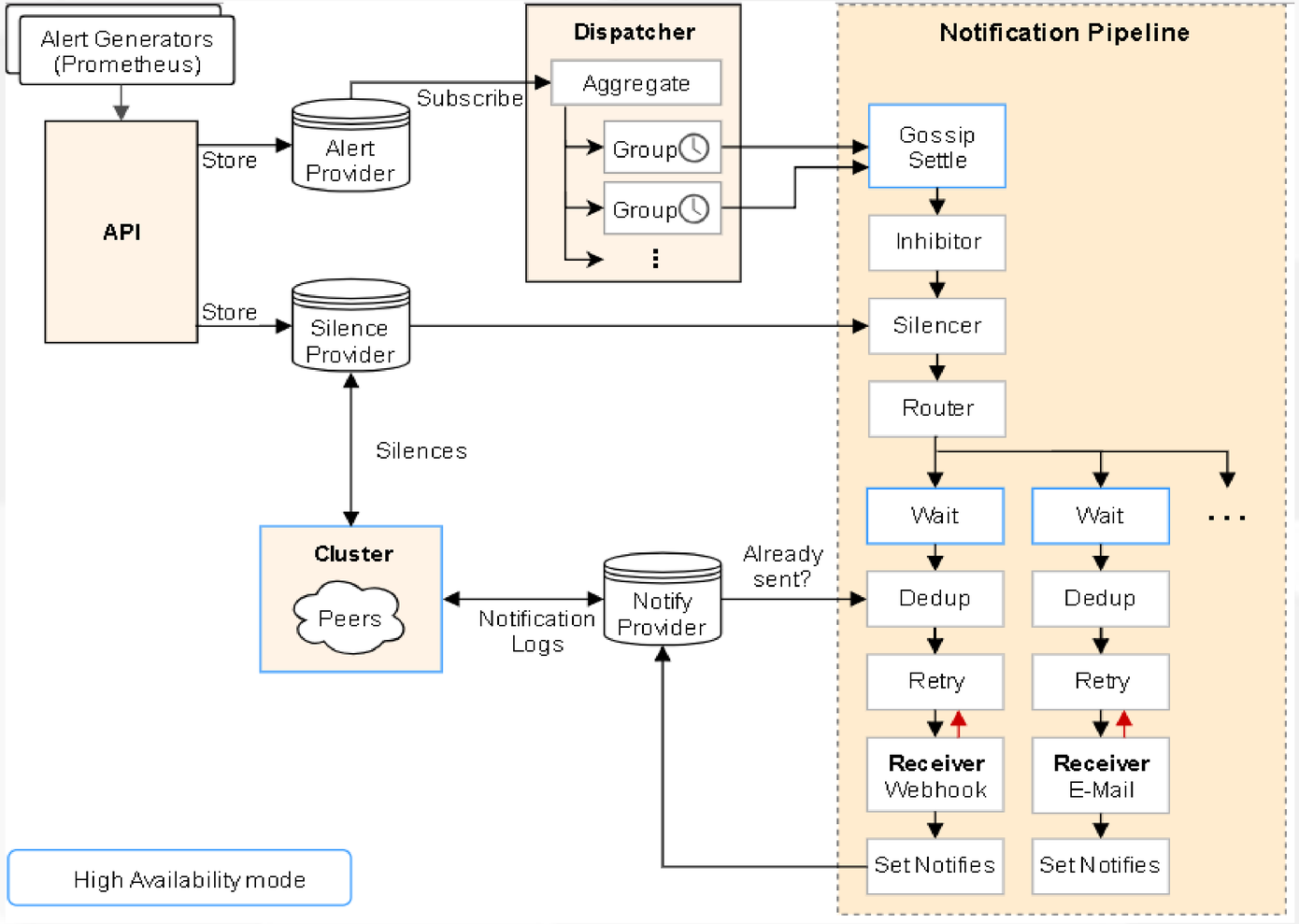
- Inactive：这里什么都没有发生。
- Pending：已触发阈值，但未满足告警持续时间
- Firing：已触发阈值且满足告警持续时间。警报发送给接受者。



告警分配

```
route:
  receiver: 'default-receiver'
  group_wait: 30s
  group_interval: 5m
  repeat_interval: 4h
  group_by: [cluster, alertname]
routes:
- receiver: 'database-pager'
    group_wait: 10s
    match_re:
      service: mysql|cassandra
- receiver: 'frontend-pager'
    group_by: [product, environment]
    match:
      team: frontend
receivers:
- name:  ‘database-pager’
  email_configs:
  - to: 'zhenliang369@163.com'
- name:  ‘frontend-pager’
  email_configs:
  - to: 'zhenliang369@163.com'
```

告警收敛（分组，抑制，静默）



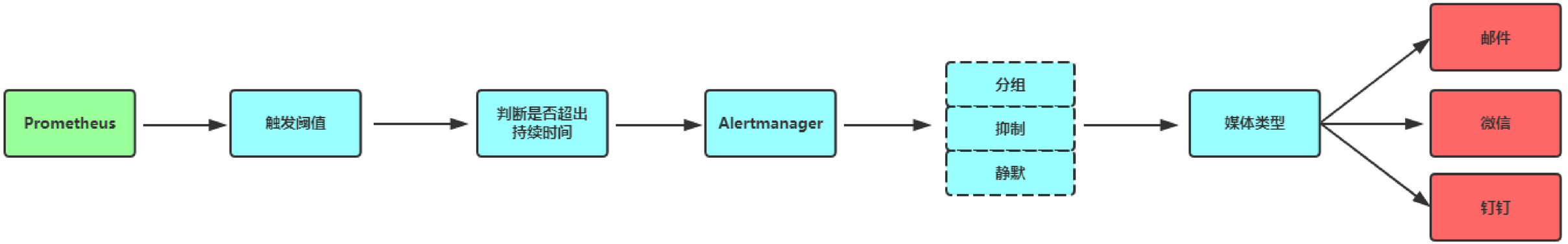
告警收敛（分组，抑制，静默）

分组（group）： 将类似性质的警报分类为单个通知

抑制（Inhibition）： 当警报发出后，停止重复发送由此警报引发的其他警报

静默（Silences）： 是一种简单的特定时间静音提醒的机制

Prometheus一条告警怎么触发的?



编写告警规则案例

示例：

```
groups:
- name: example
  rules:

  # Alert for any instance that is unreachable for >5 minutes.
  - alert: InstanceDown
    expr: up == 0
    for: 5m
    labels:
      severity: page
    annotations:
      summary: "Instance {{ $labels.instance }} down"
      description: "{{ $labels.instance }} of job {{ $labels.job }} has been down for more than 5 minutes."
```

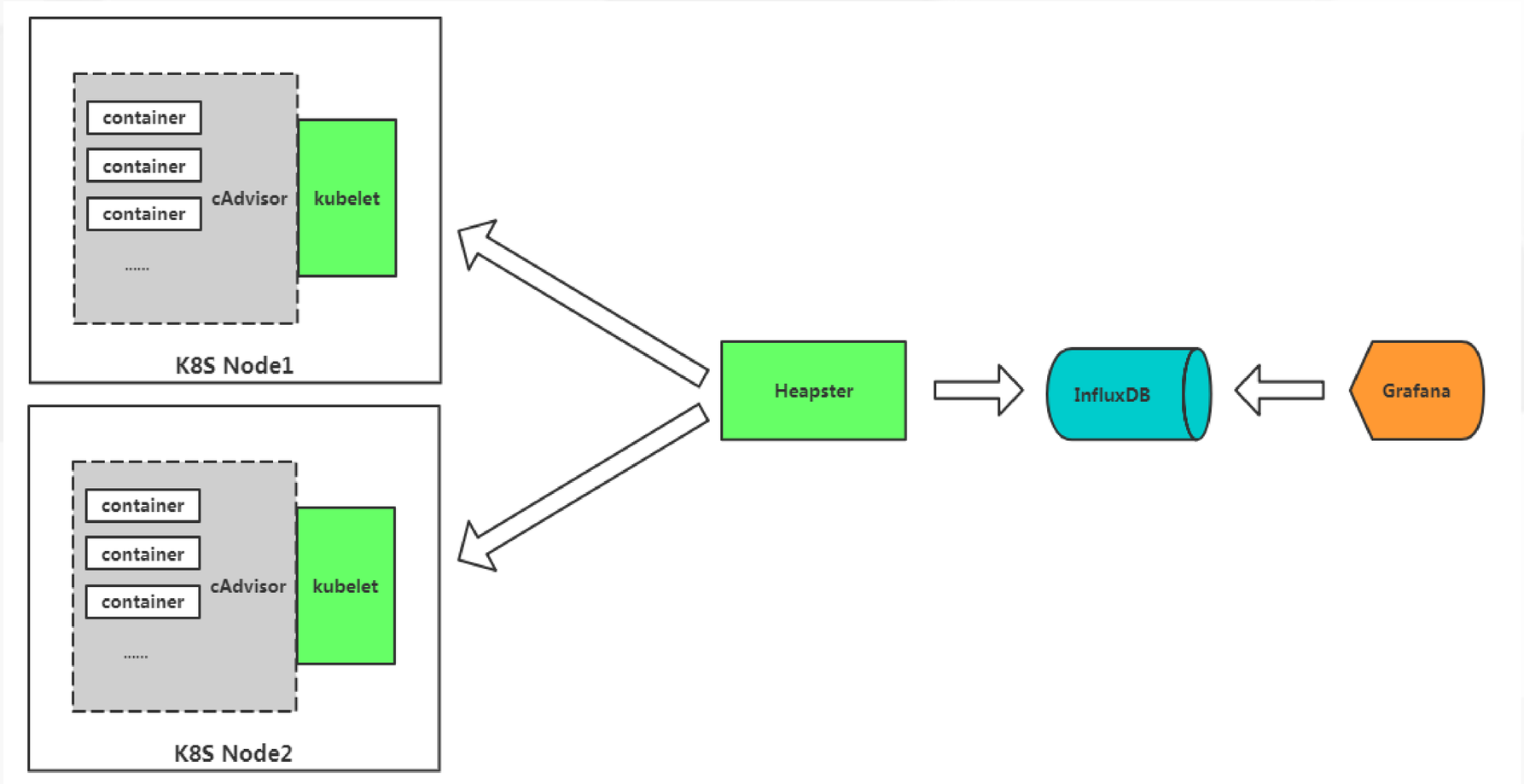
第 7 章 全方位监控Kubernetes

1. K8S监控方案
2. K8S监控指标
3. 实现思路
4. 在K8S中部署Prometheus
5. 监控K8S集群Node
6. 在K8S中部署Grafana与可视化
7. 监控K8S集群中Pod
8. 监控K8S资源对象
9. 在K8S中部署Alertmanager
10. 告警规则与告警通知

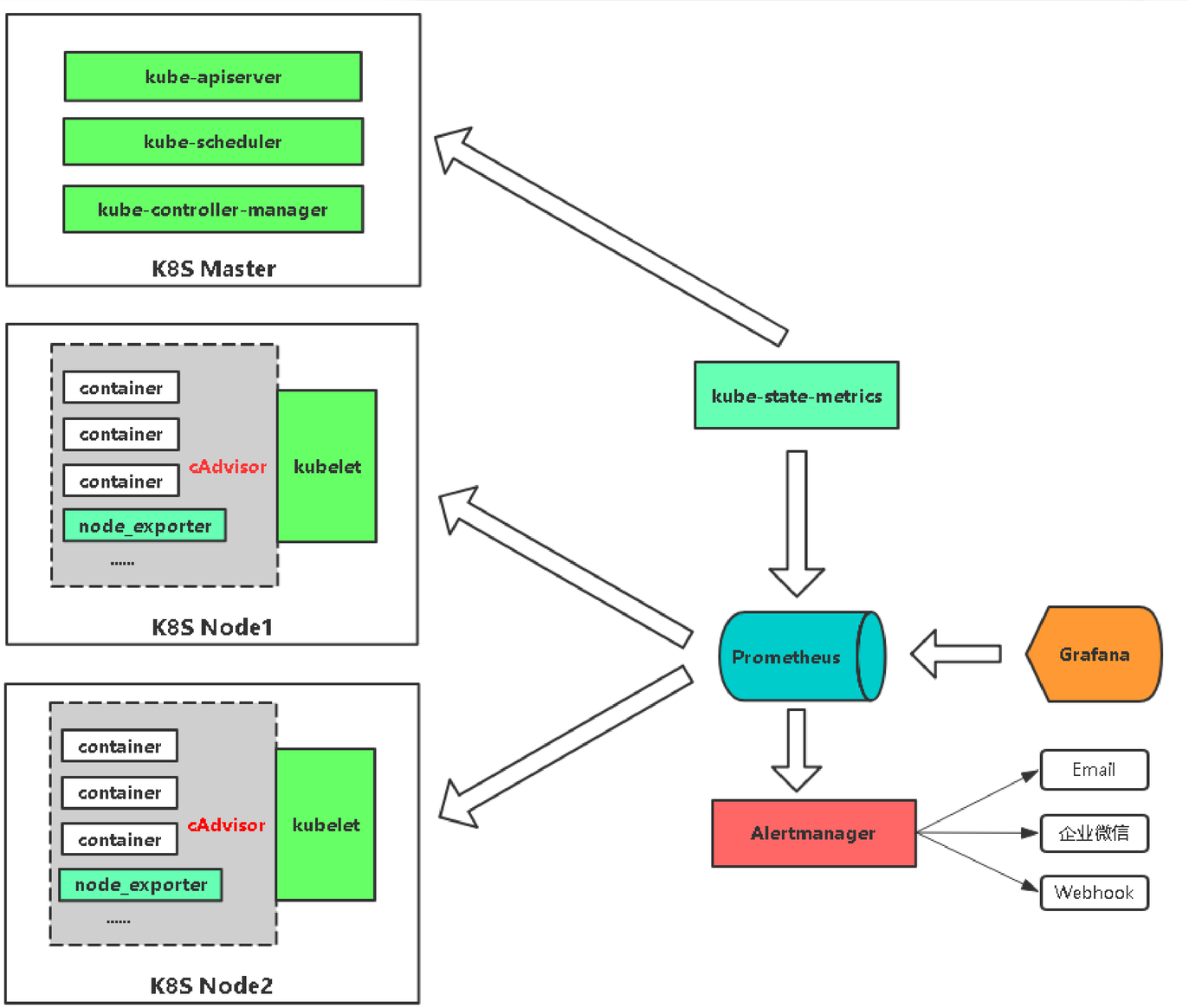
K8S监控方案

- cAdvisor+Heapster+InfluxDB+Grafana
- cAdvisor/exporter+Prometheus+Grafana

K8S监控方案


container
container
container
.....
cAdvisor
kubelet
K8S Node1
container
container
container
.....
cAdvisor
kubelet
K8S Node2
Heapster
InfluxDB
Grafana

K8S监控方案


kube-apiserver
kube-scheduler
kube-controller-manager
K8S Master
container
container
container
cAdvisor
node_exporter
.....
kubelet
K8S Node1
container
container
container
cAdvisor
node_exporter
.....
kubelet
K8S Node2
kube-state-metrics
Prometheus
Grafana
Alertmanager
Email
企业微信
Webhook

K8S监控指标

Kubernetes本身监控

- Node资源利用率
- Node数量
- Pods数量（Node）
- 资源对象状态

Pod监控

- Pod数量（项目）
- 容器资源利用率
- 应用程序

实现思路

监控指标	具体实现	举例
Pod性能	cAdvisor	容器CPU，内存利用率
Node性能	node-exporter	节点CPU，内存利用率
K8S资源对象	kube-state-metrics	Pod/Deployment/Service

服务发现：
https://prometheus.io/docs/prometheus/latest/configuration/configuration/#kubernetes_sd_config

在K8S中部署Prometheus

https://github.com/kubernetes/kubernetes/tree/master/cluster/addons/prometheus

监控K8S集群中Pod

kubelet的节点使用cAdvisor提供的metrics接口获取该节点所有容器相关的性能指标数据。

暴露接口地址：
https://NodeIP:10255/metrics/cadvisor
https://NodeIP:10250/metrics/cadvisor

在K8S中部署Grafana与可视化

Grafana是一个开源的度量分析和可视化系统。

https://grafana.com/grafana/download

推荐模板：

- 集群资源监控：3119
- 资源状态监控： 6417
- Node监控： 9276

监控K8S集群Node

node_exporter：用于*NIX系统监控，使用Go语言编写的收集器。

使用文档：https://prometheus.io/docs/guides/node-exporter/
GitHub：https://github.com/prometheus/node_exporter
exporter列表：https://prometheus.io/docs/instrumenting/exporters/

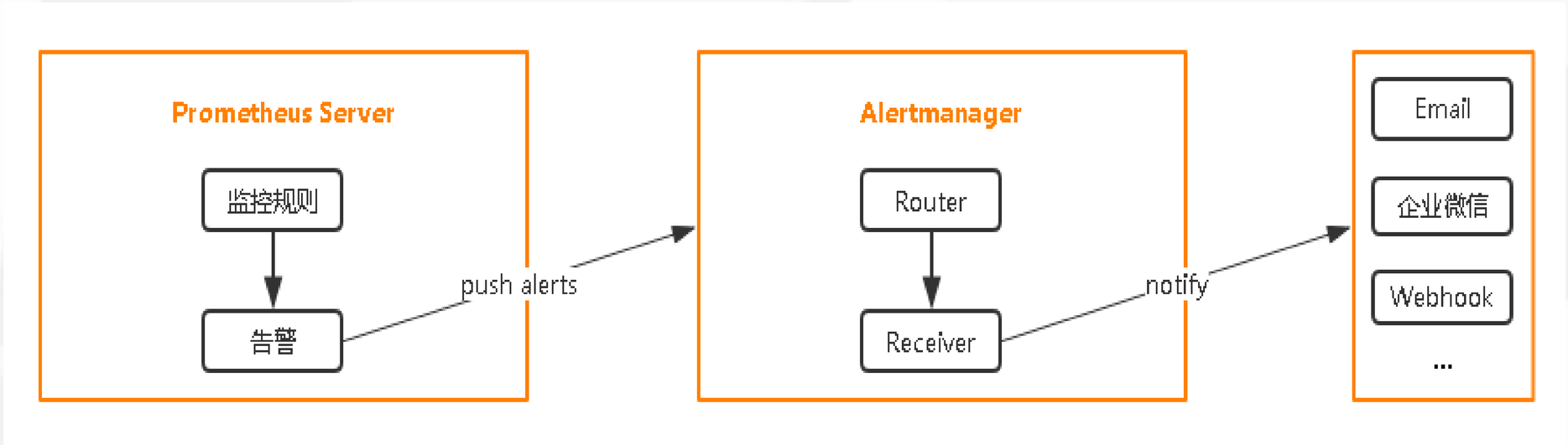
监控K8S资源对象

kube-state-metrics采集了k8s中各种资源对象的状态信息：

kube_daemonset_*
kube_deployment_*
kube_job_*
kube_namespace_*
kube_node_*
kube_persistentvolumeclaim_*
kube_pod_container_*
kube_pod_*
kube_replicaset_*
kube_service_*
kube_statefulset_*

https://github.com/kubernetes/kube-state-metrics

在K8S中部署Alertmanager



在K8S中部署Alertmanager

1. 部署Alertmanager
2. 配置Prometheus与Alertmanager通信
3. 配置告警
    1. prometheus指定rules目录
    2. configmap存储告警规则
    3. configmap挂载到容器rules目录
    4. 增加alertmanager告警配置

小结

1. 标签重要性（环境，部门，项目，管理者）
2. Grafana灵活
3. PromSQL
4. 利用服务发现动态加入目标

下一步计划：Prometheus集群， PromSQL， Grafana，对业务监控

谢谢