# 大米運維課堂

## 最前沿開源監控 Prometheus專題講座

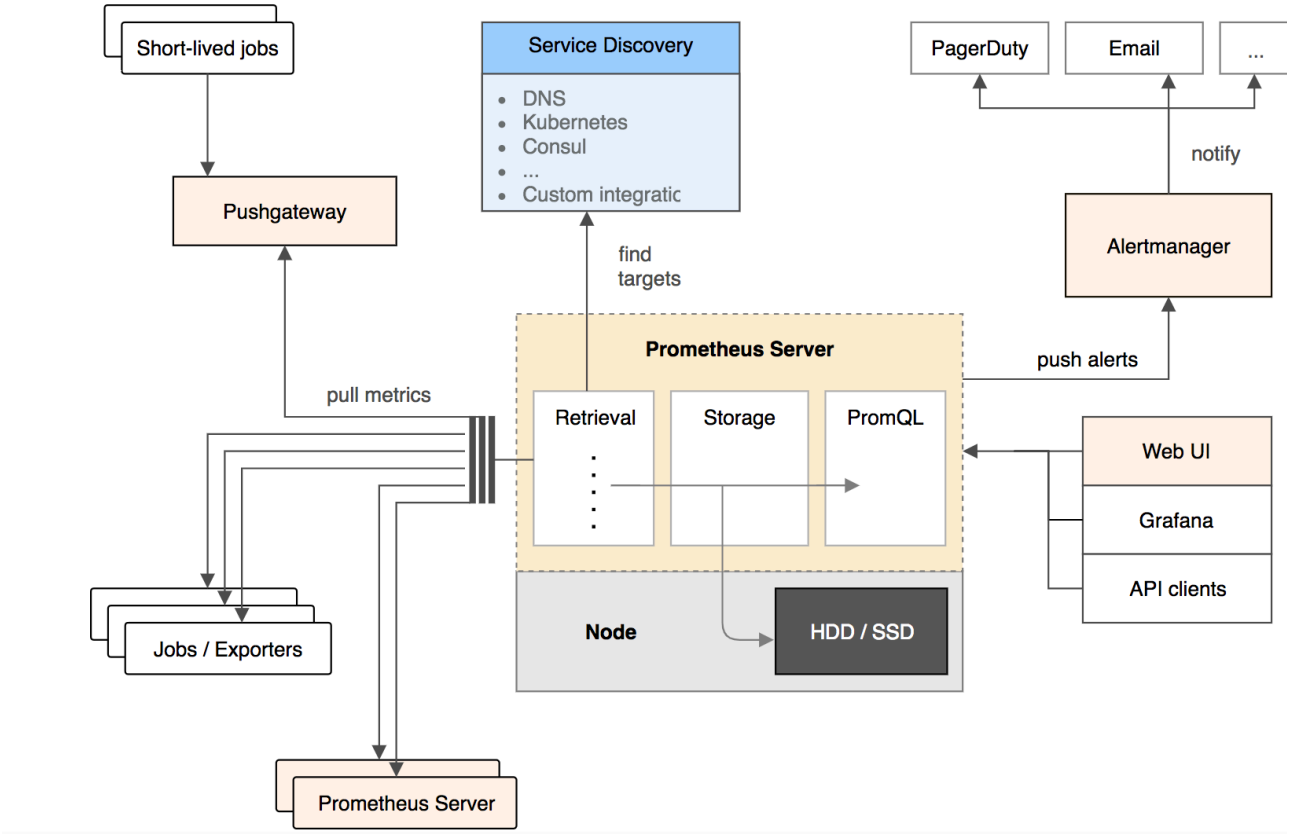
第四讲 Prometheus 运行框架介绍

第四讲内容

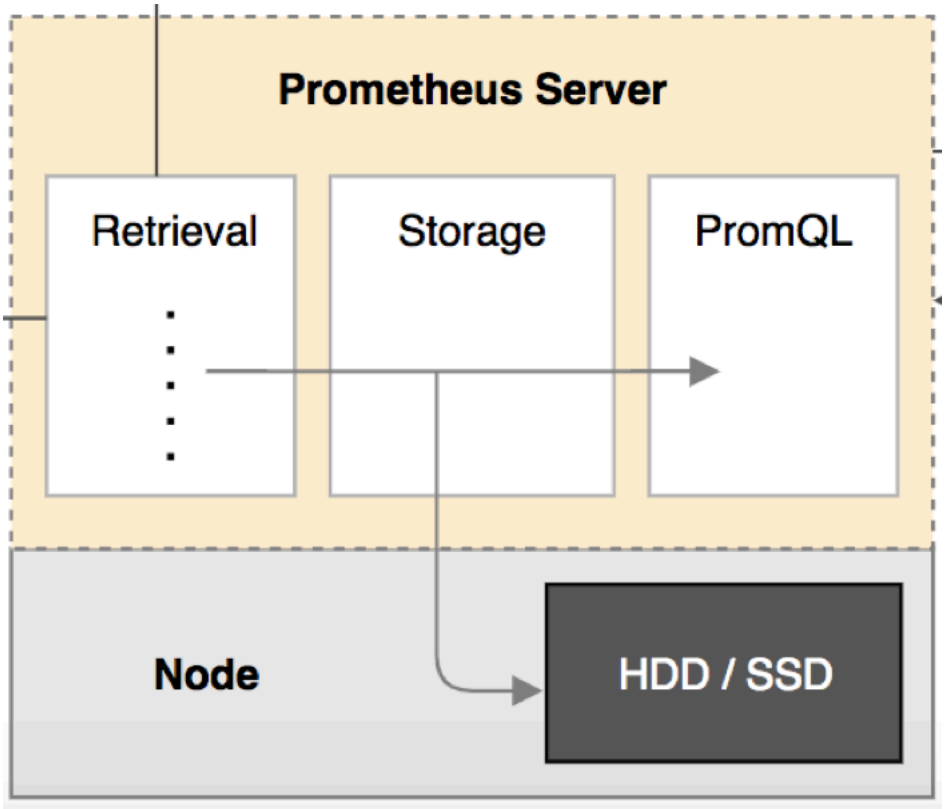
- 简单了解一下 promethues的框架结构
- 介绍一下 prometheus 的各种组件

**1） 框架结构的展示图**

This diagram illustrates the architecture of Prometheus and some of its ecosystem components:




- 我们先来看下这个部分




这里是 prometheus的服务端 也就是核心

prometheus本身是一个以进程方式启动，之后以多进程和多线程实现监控数据收集 计算 查询 更新 存储 的这样一个C/S模型运行模式

本身的启动很简单

```
root     14933  0.6  0.3 1351228 201956 ?      Sl   1月19 149:06 /data/prometheus/prometheus --web.listen-address=0.0.0.0:9090 --web.read-timeout=5m --web.max
-connections=10 --storage.tsdb.retention=15d --storage.tsdb.path=data/ --query.max-concurrency=20 --query.timeout=2m
```

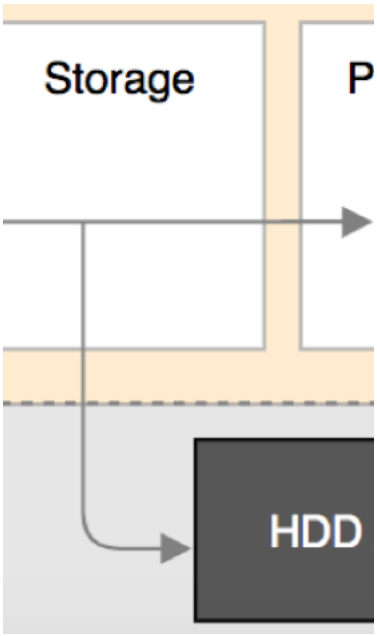
如果不带参数的 不考虑后台运行的问题

那么 prometheus 默认启动更简单

解压缩之后

./prometheus 即可 之后默认监听在9090端口 用来访问

- 接下来我们来看看prometheus的存储形式




这是一段来自官方的 prometheus存储数据的介绍

Prometheus includes a local on-disk time series database, but also optionally integrates with remote storage systems.

## Local storage

Prometheus's local time series database stores time series data in a custom format on disk.

Ingested samples are grouped into blocks of two hours. Each two-hour block consists of a directory containing one or more chunk files that contain all time series samples for that window of time, as well as a metadata file and index file (which indexes metric names and labels to series in the chunk files). The block for currently incoming samples is kept in memory and not fully persisted yet. It is secured against crashes by a write-ahead-log (WAL) that can be replayed when the Prometheus server restarts after a crash. When series are deleted via the API, deletion records are stored in separate tombstone files (instead of deleting the data immediately from the chunk files).

大米来给大家翻译和简化一下定义， 以上介绍主要包括如下几个点

- prometheus 采用的是 time-series （时间序列）的方式 以一种自定义的格式存储在 本地硬盘上
- prometheus的本地T-S（time-series）数据库以每两小时为间隔来分block（块）存储，每一个块中 又分为

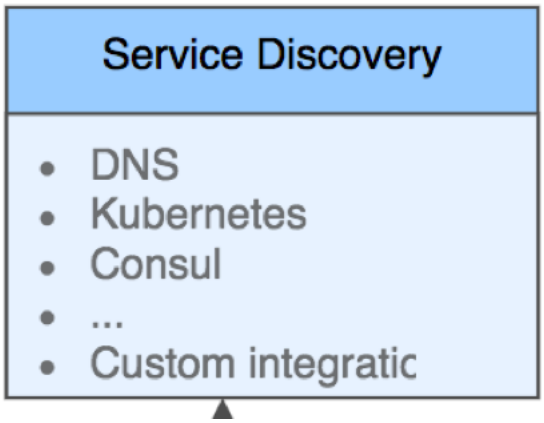
多个chunk文件（我们以后会介绍chunk的概念），chunk文件是用来存放 采集过来的数据的T-S数据，metadata 和 索引文件（index）

- index文件 是对metrics(prometheus中 一次K/V 采集数据 叫做一个metric) 和 labels(标签）进行索引 之后存储在 chunk中

chunk 是作为存储的基本单位，index and metadata是作为子集

- prometheus平时是将采集过来的数据 先都存放在内存之中（prometheus对内存的消耗还是不小的）以类似缓存的方式 用于加快搜索和访问
- 当出现当机时，prometheus有一种保护机制 叫做WAL 可以讲数据定期存入硬盘中 以chunk来表示，并在重新启动时 用以恢复进入内存

— 然后我们来看下大图中的 这个部分




这里面讲的是 prometheus 可以集成的 服务发现功能

例如 Consul

prometheus本身跟其他的开源软件类似 也是通过定义配置文件 来给prometheus本身规定需要被监控的项目和被监控节点

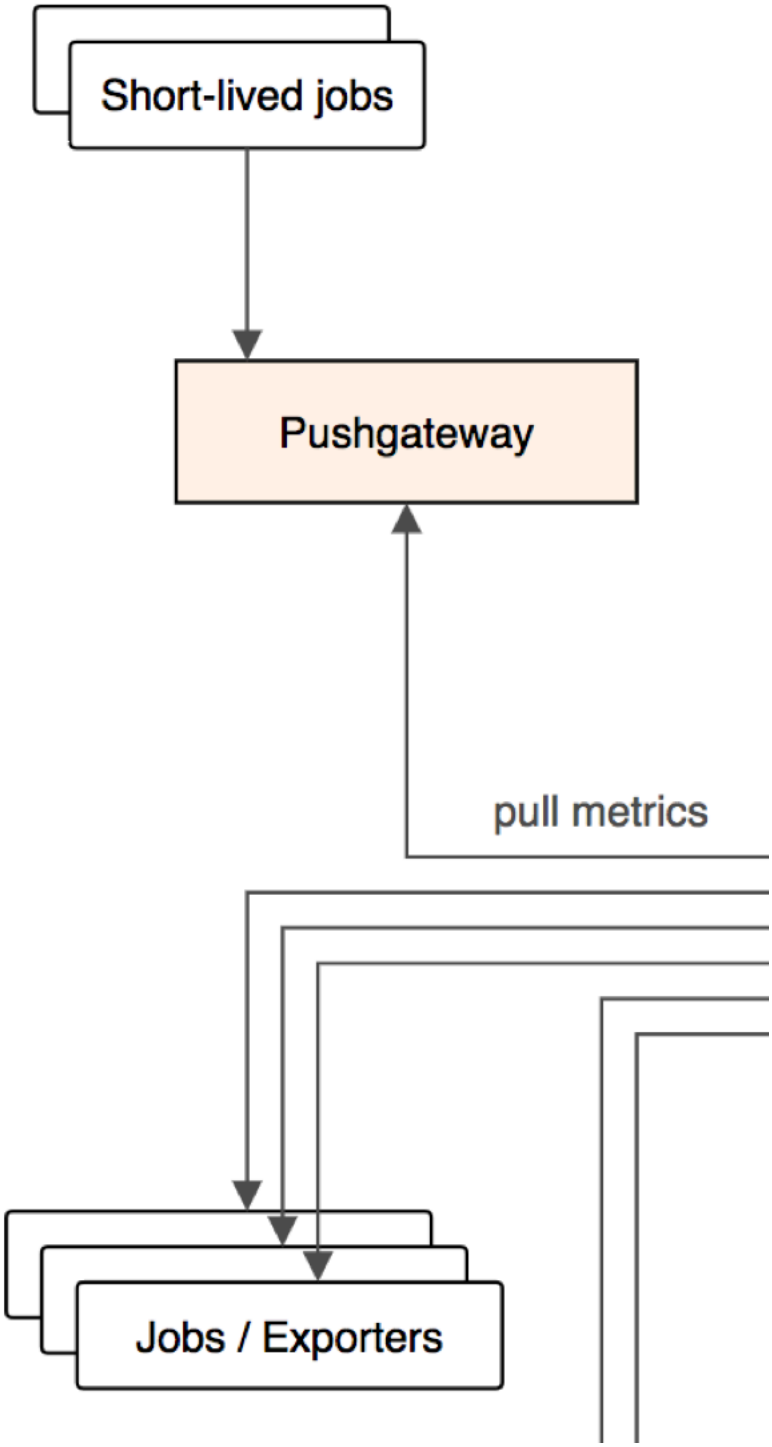
我们看下配置文件的模版

```
#
scrape_configs:
# The job name is added as a label `job=<job_name>` to any timeseries scraped from this confi
  - job_name: 'prometheus'

    # metrics_path defaults to '/metrics'
    # scheme defaults to 'http'.

    static_configs:
      - targets: ['prometheus.server:9090','prometheus.server:9100']

  - job_name: 'pushgateway'

    static_configs:
      - targets: ['localhost:9091','localhost:9092']
```

配置文件中 规定了一个 大的Job的名字
之后 在这个Jobs的名字下面 具体来定义 要被监控的节点 以及节点上具体的端口信息等等

那么如果prometheus 配合了 例如consul这种服务发现软件
prometheus的配置文件 就不再需要人工去 手工定义出来，而是能自动发现集群中 有哪些新机器 以及新机器上出现了哪些新服务 可以被监控

— 接下来我们来看看 采集客户端的部分




prometheus 的客户端 之前给大家介绍过 主要有两种 方式采集

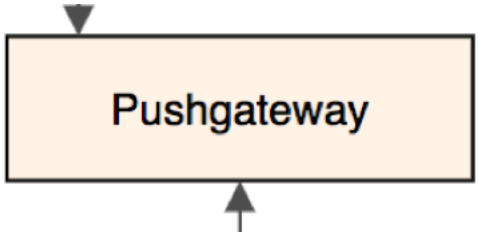
pull 主动拉取的形式

push 被动推送的形式

pull: 指的是 客户端（被监控机器）先安装各类已有exporters(由社区组织 或企业 开发的监控客户端插件)在系统上
之后，exporters 以守护进程的模式 运行 并 开始采集数据
exporter 本身也是一个http_server 可以对http请求作出响应 返回数据（K/V metrics）

prometheus 用 pull 这种主动拉的方式（HTTP get） 去访问每个节点上exporter 并采样回需要的数据

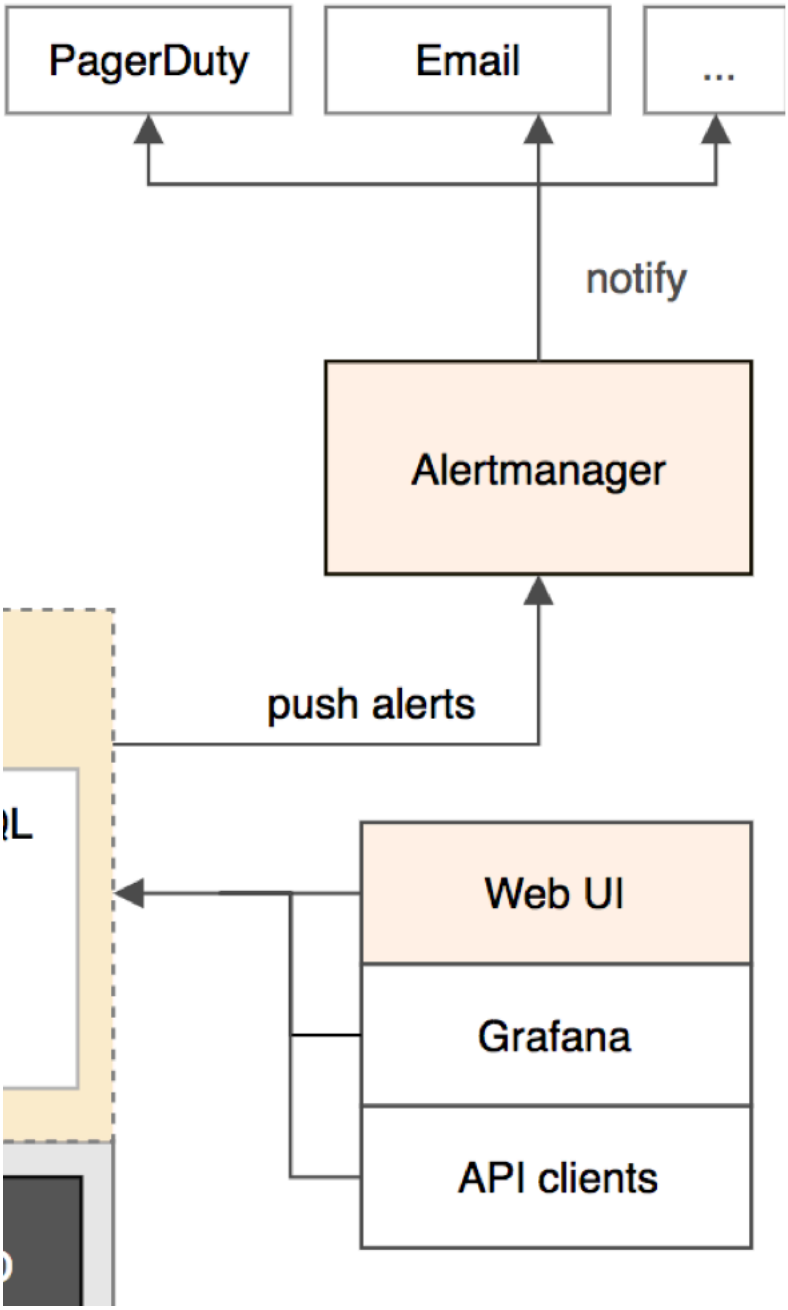
**push**：




指的是 在客户端（或者服务端）安装这个 官方提供的pushgateway插件
然后，使用我们运维自行开发的各种脚本 把监控数据组织成k/v的形式 metrics形式 发送给pushgateway
之后 pushgateway会再推送给prometheus

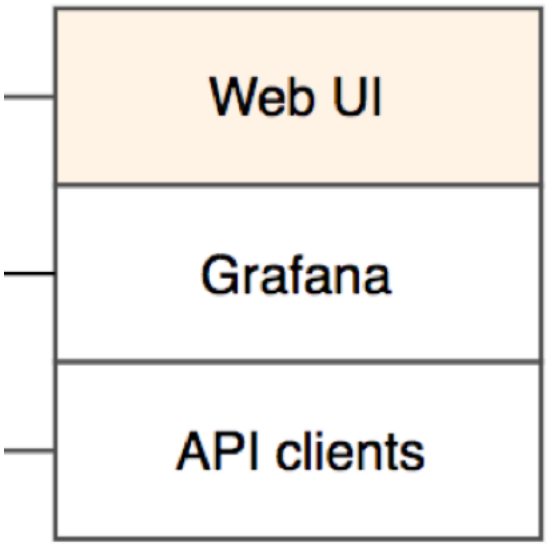
这种是一种被动的数据采集模式

**— 最后 咱们来看下这个部分 报警的部分**




prometheus 本身并不具备报警的功能

只能通过第三方 开源或者商业软件实现报警




另外
这里指的是 prometheus 可以依赖很多 方式二次绘制监控图形
本课程中 首推Grafana