prometheus-operator架构

在容器诞生之前我们的监控体系早起可能会用到Nagios zabbix cacti Ganglia 来监控云主机

在云计算崛起的时代我们利用上面所列出的监控软件对k8s集群来说就不太友好了，因为容器的生命周期可能会很短，上面的监控程序会无法定位到当前程序是否是真正意义上的挂掉或者存活，当前我们也可以利用golang 或者python调用api接口来判断容器真实的状态，但是这样做的过程肯定是复杂化的，而且我们不仅仅是只监控单方面容器的状态，他的资源使用，延迟，每秒的请求数，每秒的错误数等都是咱们的重点，而且微服务架构底下对监控的要求：既能知道服务整体的运行情况，也能够保持足够的粒度，知道某个组件的运行情况，如果有这么多的维度的监控项，咱们的代码也会成倍的增加。

Prometheus 是一套开源的监控 & 报警 & 时间序列数据库的组合,起始是由 SoundCloud 公司开发的。成立于 2012 年，之后许多公司和组织接受和采用 prometheus,他们便将它独立成开源项目，目前是独立的开源项目，任何公司都可以使用它，2016 年，Prometheus 加入了云计算基金会，成为 kubernetes 之后的第二个托管项目，现在最常见的 Kubernetes 容器管理系统中，通常会搭配 Prometheus 进行监控。

Prometheus采取了一种新的模型，将采集时序数据作为整个系统的核心，无论是告警还是构建监控图表，都是通过操纵时序数据来实现的。Prometheus通过指标的名称以及label（key/value)的组合来识别时序数据，每个label代表一个维度，可以增加或者减少label来控制所选择的时序数据

我们可以简单说下什么是时序

时间序列的格式类似于`（key，value）`这种格式，只不过是把key变成了时间戳（timestamp，value）形式，形成了即一个时间点拥有一个对应值，例如生活中很常见的天气预报，如：[(14:00, 27°C),(15:00,28°C),(16:00,26°C)]，就是一个单维的时间序列，这种按照时间戳和值存放的序列也被称之为向量（vector）。

	数据	
...	...	
now1周	*	
now1小时	*	
now30分钟	*	
now15分钟	*	
now1分钟	*	
now	*	

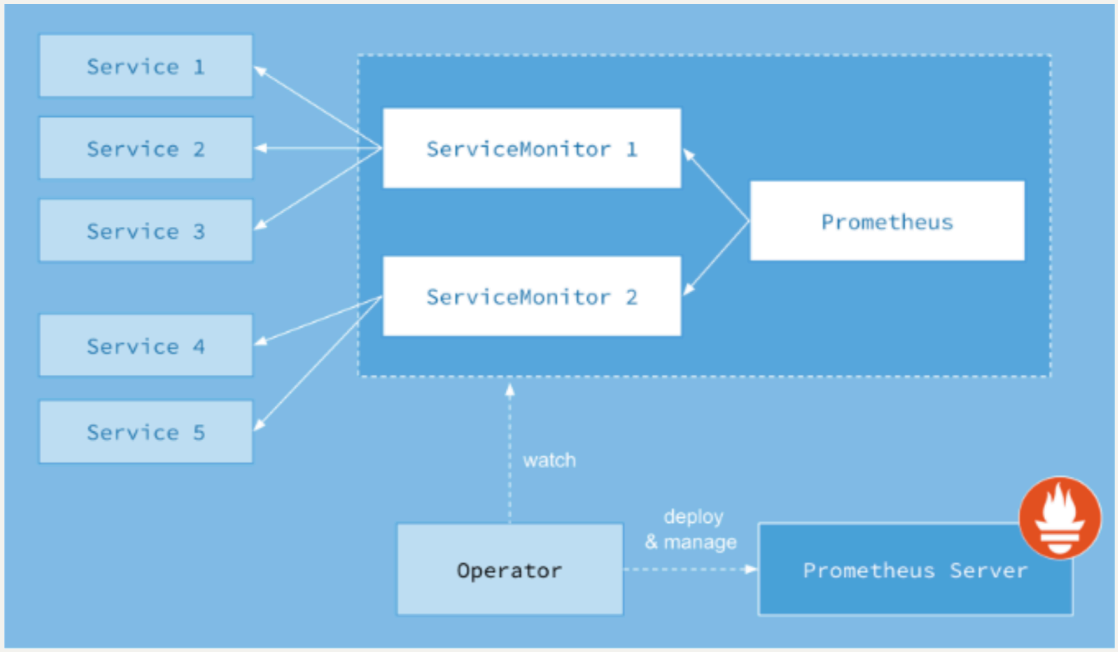
现在我们看到的只是一个独立维度的矩阵，如果我们的数据里面增加一个维度的指标呢，比如说各个省份的每个时间段的天气情况

现在就变成了一个多维度的矩阵，成为了一个·多列的向量，这个时候我们就可以知道各个省份每个时间天气情况了。

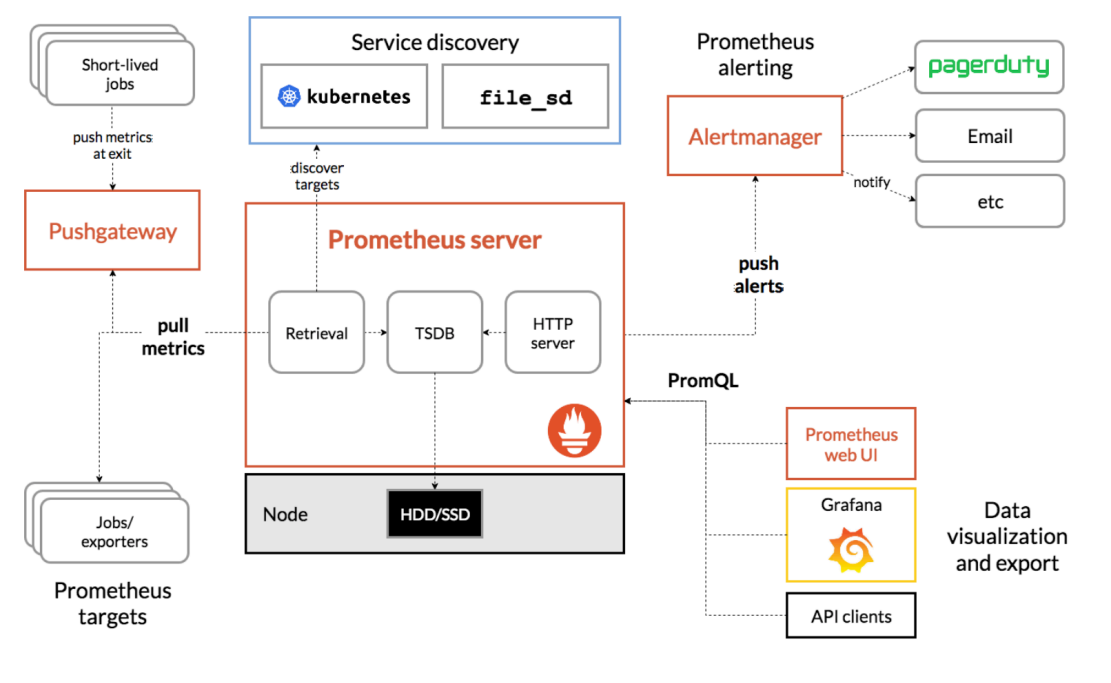
还可以基于多维度矩阵查询我们想要的多种数据，比如哪些省份1周内温度高于30度。

这些维度概念我们讲到promeQL使用查询的时候就可以一一应验，到时候可以更清晰直观的看到。

Prometheus-operator 已经改名为 Kube-promethues



- Operator： Operator 资源会根据自定义资源（Custom Resource Definition / CRDs）来部署和管理 Prometheus Server，同时监控这些自定义资源事件的变化来做相应的处理，是整个系统的控制中心。
- Prometheus： Prometheus 资源是声明性地描述 Prometheus 部署的期望状态。
- Prometheus Server： Operator 根据自定义资源 Prometheus 类型中定义的内容而部署的 Prometheus Server 集群，这些自定义资源可以看作是用来管理 Prometheus Server 集群的 StatefulSets 资源。
- ServiceMonitor： ServiceMonitor 也是一个自定义资源，它描述了一组被 Prometheus 监控的 targets 列表。该资源通过 Labels 来选取对应的 Service Endpoint，让 Prometheus Server 通过选取的 Service 来获取 Metrics 信息。
- Service： Service 资源主要用来对应 Kubernetes 集群中的 Metrics Server Pod，来提供给 ServiceMonitor 选取让 Prometheus Server 来获取信息。简单的说就是 Prometheus 监控的对象，例如 Node Exporter Service、Mysql Exporter Service 等等。
- Alertmanager： Alertmanager 也是一个自定义资源类型，由 Operator 根据资源描述内容来部署 Alertmanager 集群。



这个图刚接触的同学可能看的比较乱，但是实际是比较好理解的，我带同学们来拆开来说下就知道了。

```
prometheus server
 主要负责数据的采集和存储，提供PromQL查询语言支持
```

```
Retrieval

采样模块，采用http协议，默认pull模式拉取数据，也可以通过中间网关push数据，
prometheus的服务器在哪里拉取数据，检索拉取到的数据分发给 TSDB进行存储
```

```
TSDB

存储模块默认本地存储为TSDB
```

```
HTTP server

提供http接口查询和面板，默认端口为9090
```

short-lived jobs

存在时间不足以被删除的短暂或批量业务，无法通过pull的方式 拉取，需要使用push的方式，与pushgeteway结合使用

例如每隔一天做一次数据库备份，我们想要知道每次备份执行了多长时间，备份是否成功，我们备份任务只会执行一段时间，如果备份任务结束了，Prometheus Server该如何拉取备份指标的数据呢？解决这种问题，可以通过Prometheus的pushgateway组件来做，每个备份任务将指标推送pushgateway组件，pushgateway将推送来的指标缓存起来，Prometheus Server从Pushgateway中拉取指标。

Service Discovery
服务发现，prometheus支持多种服务发现机制：文件，DNS，k8s，openstack,等，基于服务发现的过程，通过第三方接口，prometheus查询到需要监控的target列表，然后轮询这些target获取监控数据

pushgateway
支持临时性的job主动推送指标的中间网关，prometheus默认通过pull方式从exporters拉取，但有些情况我们是不允许promethes与exporters直接进行通信的，这时候我们可以使用pushgateway由客户端主动push数据到pushgateway，在由prometheus拉取。很多时候我们需要自定义一些组件来采集

!!!!
pushgateway 适用于短暂任务节点的监控数据收集，因为他设计为一个监控指标的缓存，这意味着它不会主动过期服务上报的指标，这种情况在服务一直运行的时候不会有问题，但当服务被重新调度或销毁时，Pushgateway依然会保留着之前节点上报的指标。如果有多个实例的话，会造成数据重复

而且在拉模式下，Prometheus可以更容易的查看监控目标实例的健康状态，并且可以快速定位故障，但在推模式下，由于不会对客户端进行主动探测，因此对目标实例的健康状态也变得一无所知

exporters
支持其他数据源的指标导入到prometheus,支持数据库，硬件，消息中间件，存储系统，http服务器，jmx等

负责收集目标对象的性能数据，并通过http接口供prometheus server获取，只要符合接口格式，就可以被采集

alertmanager
用来进行报警发送到各个平台，可以进行消息分组消息抑制等功能

简单总体解释：

首先：prometheus根据配置定时去拉取各个节点的数据，默认使用的拉取方式是pull，也可以使用pushgateway提供的push方式获取各个监控节点的数据。

然后：将获取到的数据存入TSDB，TSDB最终会把内存中的时间序列压缩落到硬盘， 此时prometheus已经获取到了监控数据，可以使用内置的PromQL进行查询。它的报警功能使用Alertmanager提供，Alertmanager是prometheus的告警管理和发送报警的一个组件，prometheus Server会定时地执行告警规则，告警规则是PromQL表达式，表达式的值是true或false，如果是true，就将产生的告警数据推送给alertmanger。告警通知的聚合、分组、发送、禁用、恢复等功能，并不是Prometheus Server来做的，而是Alertmanager来做的，Prometheus Server只是将触发的告警数据推送给Alertmanager，然后Alertmanger根据配置将告警聚合到一块，发送给对应的接收人。

最后prometheus数据接入grafana，由grafana进行图形化管理。