首先clone项目代码

```
git clone https://github.com/prometheus-operator/kube-prometheus.git
```

```
cd kube-prometheus

# 进入到 manifests 目录下面，首先需要安装 setup 目录下面的资源对象
# 这会创建一个名为 monitoring 的命名空间，以及相关的 CRD 资源对象声明和 Prometheus-Operator 控制器
cd manifests
kubectl apply -f setup/
```

查看具体配置

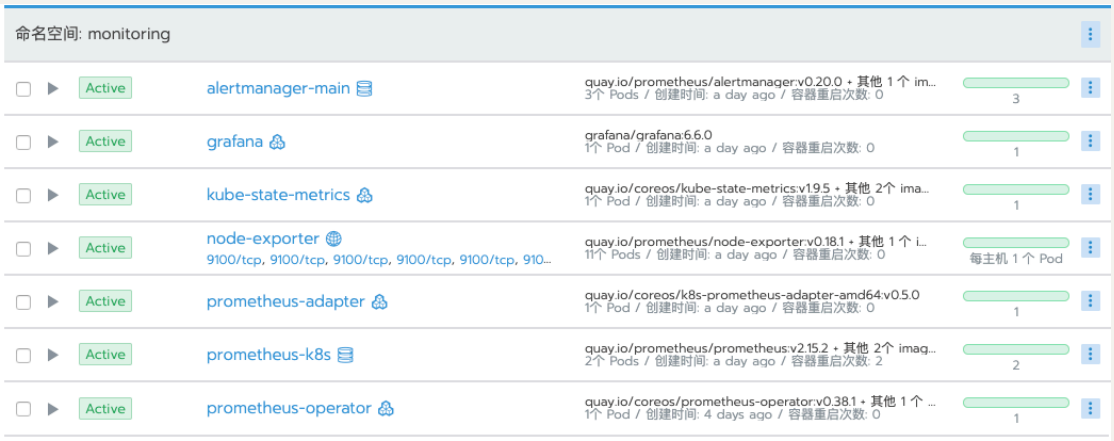
setup 下的配置文件解释

```
0namespace-namespace.yaml                                  ##命名空间
prometheus-operator-0alertmanagerCustomResourceDefinition.yaml   ##定义了crd 所部署的alertmanager部署信息
prometheus-operator-0podmonitorCustomResourceDefinition.yaml     ##pod层面的的自动生成的抓取配置
prometheus-operator-0prometheusCustomResourceDefinition.yaml     ##定义了crd 所部署的prometheus部署信息配置
prometheus-operator-0prometheusruleCustomResourceDefinition.yaml ##此crd可以定义报警配置信息
prometheus-operator-0servicemonitorCustomResourceDefinition.yaml ##service的自动生成的抓取配置
prometheus-operator-0thanosrulerCustomResourceDefinition.yaml    ##thanos 部署信息配置
prometheus-operator-clusterRole.yaml                       ##rbac权限配置
prometheus-operator-clusterRoleBinding.yaml                ##rbac绑定配置
prometheus-operator-deployment.yaml                        ##prome-operator部署配置
prometheus-operator-service.yaml                           ##prome-operator service配置
prometheus-operator-serviceAccount.yaml                    ## serviceaccount
```

部署后我们去去查看Prometheus-Operator容器的状态及日志，日志会显示CRD组件的状态信息，之前说到他是系统的控制中心，是一个无状态的容器。

然后部署kube-prometheus 查看容器状态

```
kubectl apply -f .
```



我们看到的就是原生部署的kube-prometheus

然后同学们我们可以看下各个容器的配置

首先我们看下prometheus容器，他启动了2个容器

一个prometheus系统容器

一个是[prometheus-config-reloader]：配置文件自动更新容器 默认3分钟

然后我们看下alertmanag容器配置

一个alertmanager容器

一个[config-reloader]：报警接受人配置自动更新容器

grafana

图形化展示

node-exporter

采集当前主机的资源 是一个DaemonSet类

kube-state-metrics

采集了k8s中各种资源对象的状态信息， 如：pod/deployment/service

prometheus-adapter

`prometheus`属于第三方的 解决方案，原生的k8s系统并不能对`Prometheus`的自定义指标进行解析，就需要借助于`k8s-prometheus-adapter`将这些指标数据查询接口转换为标准的`Kubernetes`自定义指标,咱们不光可以自己聚合自定义指标，还利用他可以完成hpa弹性伸缩，比如系统默认hpa弹性伸缩只能监控内存和cpu，利用prometheus-adapter咱们可以完成比如说是qps弹性伸缩，容器请求量达到10000以上就增加pods数量，低于10000就进行缩容。

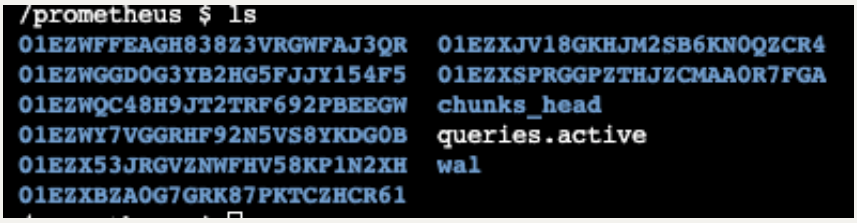
更改配置

我们现在看到kube-prometheus中除了alert和prometheus是有状态类型的，其他都是无状态类型的

alertmanager因为他就是个报警发送的容器所以我们可以看到他里面核心的就是挂载了configmap配置的报警接收者。

而prometheus容器中挂载了prometheus的配置文件，但是我们看到他的默认配置中数据目录是临时的存储：/prometheus 中存储的prometheus的数据块Prometheus将采集的样本放到内存中，默认每隔2小时将数据压缩成一个block，持久化到硬盘中，TSDB将存储的监控数据按照时间分成多个block存储,默认最小的block保存时间为2h,后台程序还会将小块合并成大块，减少内存中block的数量，便于索引查找数据，我们想获取时间较长的数据时，prometheus会把这些块进行一个解封操作进行读取,每一个块中包含该时间窗口内的所有样本数据(chunks)，元数据文件(meta.json)以及索引文件(index)。

我们可以看看里面都存在着哪些数据文件



01开头的这些目录就是块文件:

```
|   ├── chunks                # 保存压缩后的时序数据，每个chunks大小为512M，超过会生成新的chunks
|   |   └── 000001
|   ├── index                 # chunks中的偏移位置
|   ├── meta.json             # 记录block块元信息，比如 样本的起始时间、chunks数量和数据量大小等
|   └── tombstones            # 通过API方式对数据进行软删除,将删除记录存储在此处（API的删除方式，并不是立即将数据从chunks文件中移除）
```

外层的wal目录作用是:

```
└── wal                       #防止数据丢失(数据收集上来暂时是存放在内存中,wal记录了这些信息)
    ├── 00000366              #每个数据段最大为128M，存储默认存储两个小时的数据量。
```

还有一个是原生安装的没有增加容器反亲和性我们可以增加。

我们在prometheus-prometheus.yaml最底层增加下面红框内容容器反亲和性和pv挂载

```
    prometheus: k8s
    role: alert-rules
  securityContext:
    fsGroup: 2000
    runAsNonRoot: true
    runAsUser: 1000
  serviceAccountName: prometheus-k8s
  serviceMonitorNamespaceSelector: {}
  serviceMonitorSelector: {}
  version: 2.24.0
  affinity:
    podAntiAffinity:
      preferredDuringSchedulingIgnoredDuringExecution:
      - weight: 100
        podAffinityTerm:
          labelSelector:
            matchExpressions:
            - key: prometheus
              operator: In
              values:
              - k8s
          topologyKey: kubernetes.io/hostname
  storage:
    volumeClaimTemplate:
      spec:
        storageClassName: nfs
        resources:
          requests:
            storage: 50Gi
```

prometheus配置文件解析

上面的配置现在我们已经完善了一部分了，现在我们来看下prometheus配置文件规则

一. alert配置

```
#我们现在知道的是prometheus中会有报警规则然后通过alert报警但是他们之间是怎么关联的呢？如下⬇️


global: #全局配置项
  scrape_interval: 30s #默认情况下抓取目标的频率.
  scrape_timeout: 10s    # 抓取超时时间.
  evaluation_interval: 30s ## 触发告警检测的时间
  external_labels: #用于外部系统标签的服务，外部服务接入进来必须存在external_labels，比如我们后面要说的thanos
    prometheus: monitoring/k8s
    prometheus_replica: prometheus-k8s-1
alerting: #alerting关联配置
  alert_relabel_configs: ## 在抓取之前对任何目标及其标签进行修改，这个不管。
  - separator: ;
    regex: prometheus_replica
    replacement: $1
    action: labeldrop  #将regex与所有标签名称匹配。匹配的任何标签都将从标签集中删除。
  alertmanagers:
  - scheme: http #连接的协议
    path_prefix: / #将推送HTTP路径警报前缀
    timeout: 10s  #超时时间
    api_version: v2 #api版本
    relabel_configs: #这块比较重要了这里写着怎么关联着alert
    - source_labels: [__meta_kubernetes_service_name] #通过service资源名称类找到alert
      separator: ; #固定的标签分割符号
      regex: alertmanager-main #service名称可以在serice找到alertmanager-main
      replacement: $1 #如果上面的regex匹配到了就替换当前值相当于$1=alertmanager-main
      action: keep #删除regex与连接的source_labels不匹配的目标
    - source_labels: [__meta_kubernetes_endpoint_port_name] #通过endpoint资源类找到端口
      separator: ;
      regex: web #找到alert要交互的端口名称，可以在service yaml文件中找到
      replacement: $1
      action: keep
    kubernetes_sd_configs: #这个是配置在哪个命名空间找到alert服务与上面配置结合
    - role: endpoints
      namespaces:
```

```
        names:
        - monitoring
rule_files:
- /etc/prometheus/rules/prometheus-k8s-rulefiles-0/*.yaml
# #从所有匹配的文件中读取规则和警报，可以在depoly yaml文件中找
到/etc/prometheus/rules/prometheus-k8s-rulefiles-0挂载为configmap名
称为prometheus-k8s-rulefiles-0里面有很多yaml文件
```

二. 抓取配置列表配置（抓取一小段）

```
scrape_configs: #抓取配置列表
- job_name: monitoring/alertmanager/0 #分配给已抓取指标的job名称，它是
唯一的
#Honor_timestamps控制Prometheus是否抓取数据中的时间戳。
#如果将honor_timestamps设置为“true”，则将使用目标公开的指标的时间戳。
#如果将honor_timestamps设置为“ false”，则目标忽略的度量标准的时间戳将被忽
略。
  honor_timestamps: true
  scrape_interval: 30s
  scrape_timeout: 10s
  metrics_path: /metrics
  scheme: http
  relabel_configs:
  - source_labels: [__meta_kubernetes_service_label_alertmanager]
    separator: ;
    regex: main
    replacement: $1
    action: keep
  - source_labels:
[__meta_kubernetes_service_label_app_kubernetes_io_component]
    separator: ;
    regex: alert-router
    replacement: $1
    action: keep
  - source_labels:
[__meta_kubernetes_service_label_app_kubernetes_io_name]
    separator: ;
    regex: alertmanager
    replacement: $1
    action: keep
  - source_labels:
[__meta_kubernetes_service_label_app_kubernetes_io_part_of]
    separator: ;
    regex: kube-prometheus
```

```
    replacement: $1
    action: keep
```

下面的内容大同小异，行数太多就不一一讲了，知道格式方法就可以了，这些都是Prometheus-Operator 原生自带的，不需要更改，一会我我们可以自定义一些配置。

接下来我们看下报警规则，这里面记载了，如果触发了指定规则，就会通知的配置