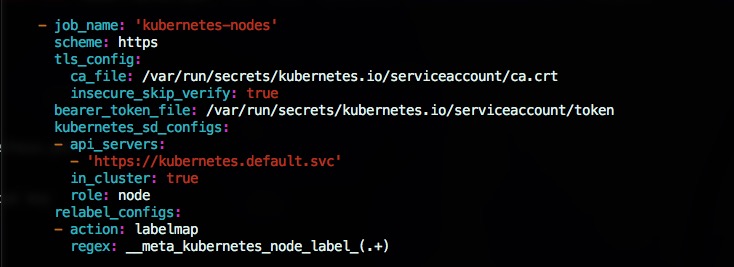
# Service discover

http://192.168.0.181:10255/metrics

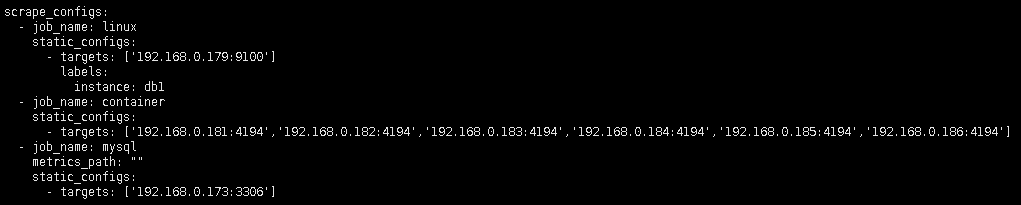
- job\_name: 'kubernetes-pods'  
  kubernetes\_sd\_configs:  
  - api\_server: 'C:\Users\dell\AppData\Local\Temp\%W@GJ$ACOF(TYDYECOKVDYB.pnghttp://x.x.x.x:8080'  
    role: pod

# Tls\_config+k8s\_sd\_config



# Static\_config

192.168.0.181:4194



# Kubernetes cluster monitoring (via Prometheus) dashboard for Grafana | Grafana Labs

https://grafana.com/dashboards/315

Initial idea was taken from [this dashboard](https://grafana.net/dashboards/162) and improved to exclude node-exporter dependency and to give more information about cluster state.

## Requirements

You only need to have running [Kubernetes](http://kubernetes.io/) cluster with deployed [Prometheus](https://prometheus.io/). Prometheus will use metrics provided by [cAdvisor](https://github.com/google/cadvisor) via [kubelet](http://kubernetes.io/docs/admin/kubelet)service (runs on each node of Kubernetes cluster by default) and via [kube-apiserver](http://kubernetes.io/docs/admin/kube-apiserver) service only.

Your Prometheus configuration has to contain following [scrape\_configs](https://prometheus.io/docs/operating/configuration/#scrape_config):

scrape\_configs:

- job\_name: kubernetes-nodes-cadvisor

scrape\_interval: 10s

scrape\_timeout: 10s

scheme: https # remove if you want to scrape metrics on insecure port

tls\_config:

ca\_file: /var/run/secrets/kubernetes.io/serviceaccount/ca.crt

bearer\_token\_file: /var/run/secrets/kubernetes.io/serviceaccount/token

kubernetes\_sd\_configs:

- role: node

relabel\_configs:

- action: labelmap

regex: \_\_meta\_kubernetes\_node\_label\_(.+)

metric\_relabel\_configs:

- action: replace

source\_labels: [id]

regex: '^/machine\.slice/machine-rkt\\x2d([^\\]+)\\.+/([^/]+)\.service$'

target\_label: rkt\_container\_name

replacement: '${2}-${1}'

- action: replace

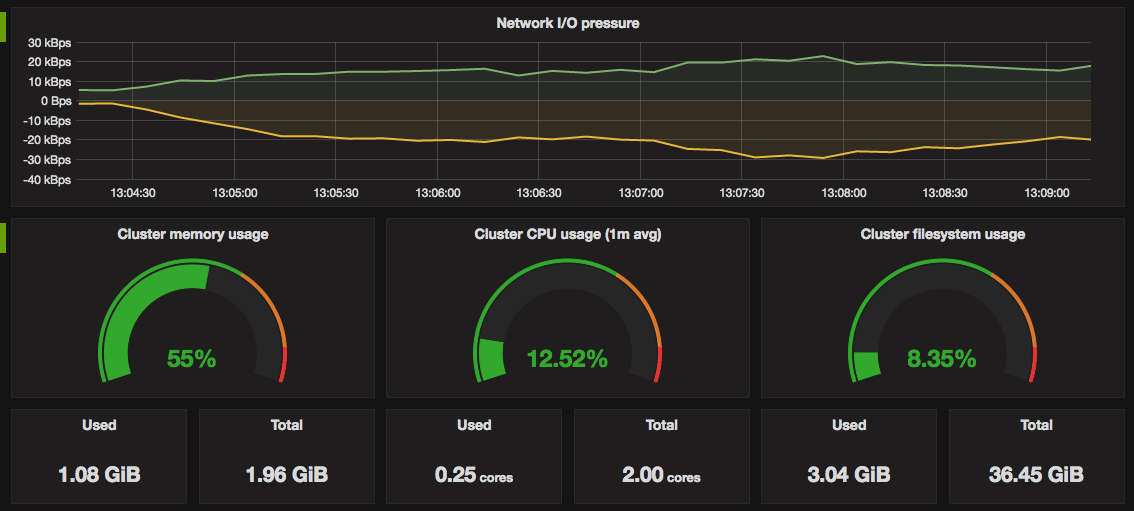
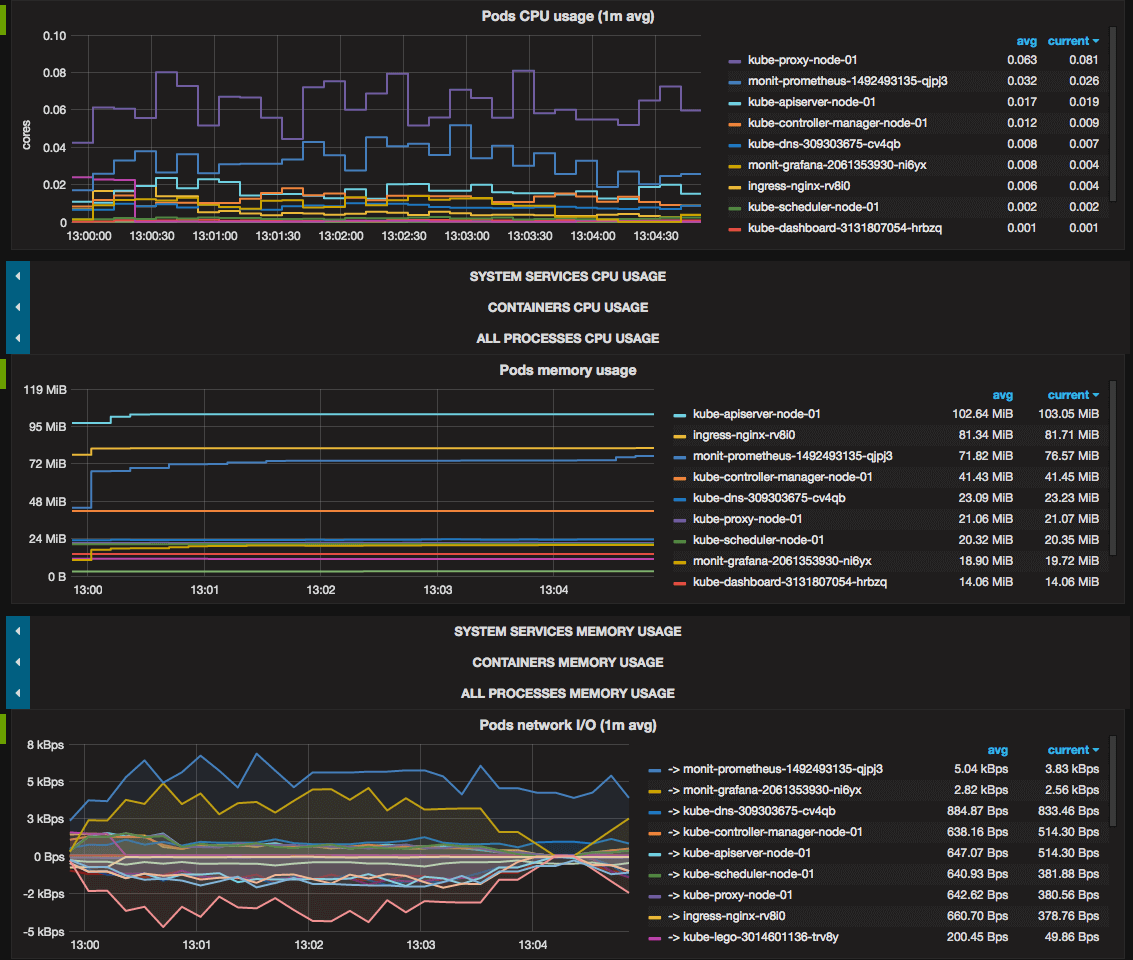
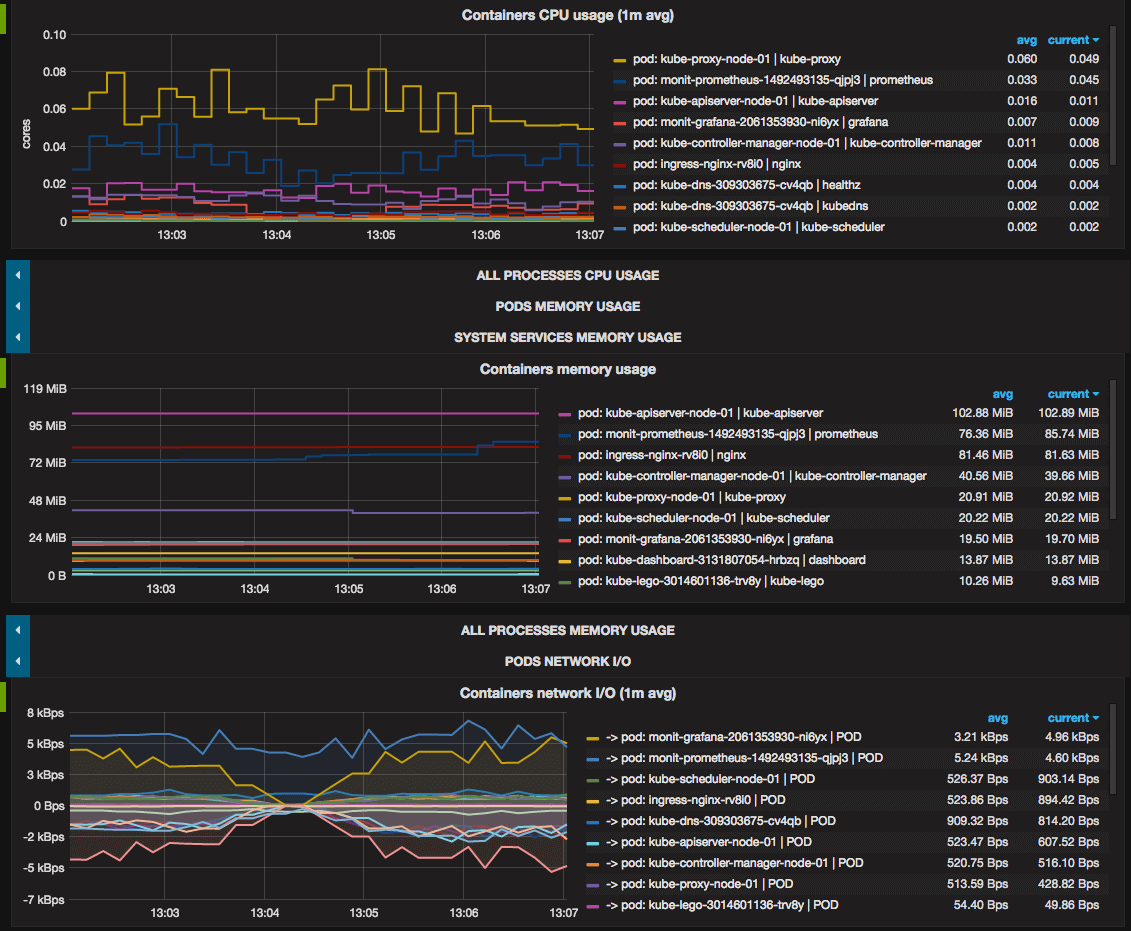
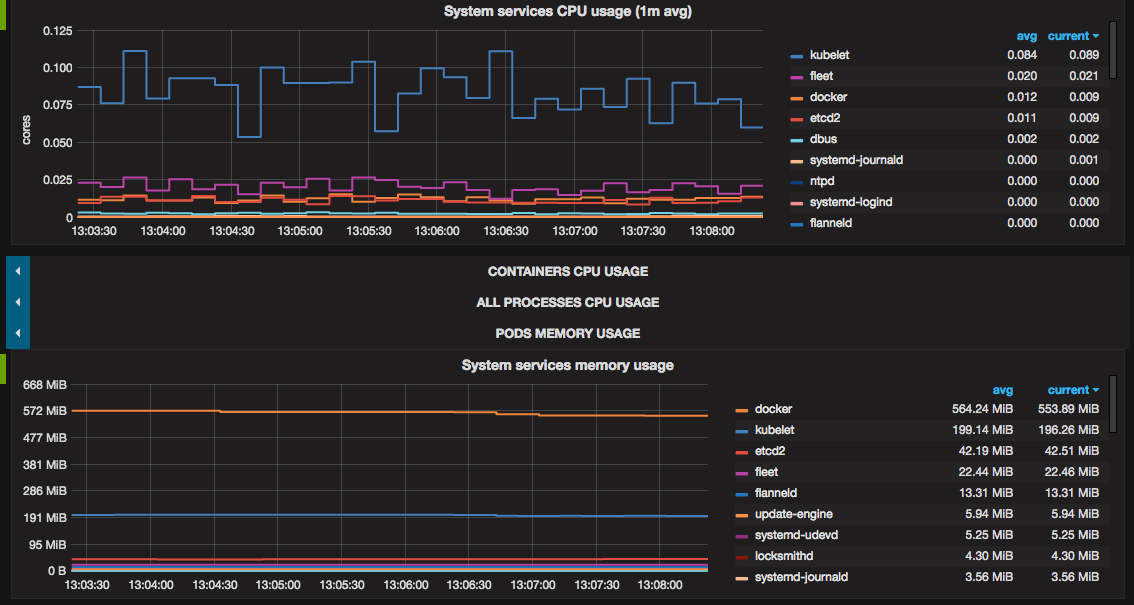
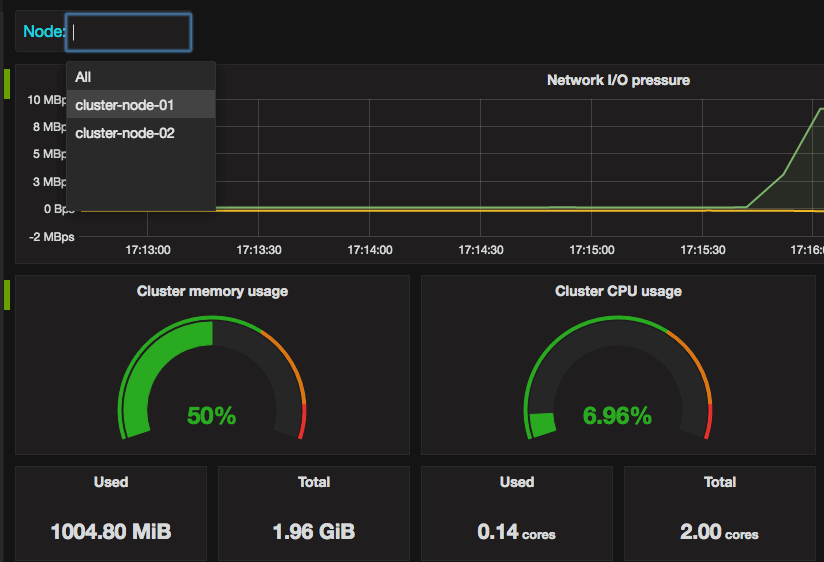
source\_labels: [id]

regex: '^/system\.slice/(.+)\.service$'

target\_label: systemd\_service\_name

replacement: '${1}'

## Features

* Total and used cluster resources: CPU, memory, filesystem.  
  And total cluster network I/O pressure.  
  
* [Kubernetes pods](http://kubernetes.io/docs/user-guide/pods) usage: CPU, memory, network I/O.  
  
* Containers usage: CPU, memory, network I/O.  
  [Docker](https://www.docker.com/) and [rkt](https://coreos.com/rkt) containers which runs on cluster nodes but outside Kubernetes are also monitored.  
  
* [systemd](https://freedesktop.org/wiki/Software/systemd) system services usage: CPU, memory.  
  
* Showing all above metrics both for all cluster and each node separately.  
  

## Troubleshooting

If filesystem usage panels display N/A, you should correct device=~"^/dev/[vs]da9$" filter parameter in metrics query with devices your system actually has.

# prometheus/prometheus-kubernetes.yml

at master · prometheus/prometheus

https://github.com/prometheus/prometheus/blob/master/documentation/examples/prometheus-kubernetes.yml

# A scrape configuration for running Prometheus on a Kubernetes cluster.

# This uses separate scrape configs for cluster components (i.e. API server, node)

# and services to allow each to use different authentication configs.

#

# Kubernetes labels will be added as Prometheus labels on metrics via the

# `labelmap` relabeling action.

# Scrape config for API servers.

#

# Kubernetes exposes API servers as endpoints to the default/kubernetes

# service so this uses `endpoints` role and uses relabelling to only keep

# the endpoints associated with the default/kubernetes service using the

# default named port `https`. This works for single API server deployments as

# well as HA API server deployments.

scrape\_configs:

- job\_name: 'kubernetes-apiservers'

kubernetes\_sd\_configs:

- role: endpoints

# Default to scraping over https. If required, just disable this or change to

# `http`.

scheme: https

# This TLS & bearer token file config is used to connect to the actual scrape

# endpoints for cluster components. This is separate to discovery auth

# configuration because discovery & scraping are two separate concerns in

# Prometheus. The discovery auth config is automatic if Prometheus runs inside

# the cluster. Otherwise, more config options have to be provided within the

# <kubernetes\_sd\_config>.

tls\_config:

ca\_file: /var/run/secrets/kubernetes.io/serviceaccount/ca.crt

# If your node certificates are self-signed or use a different CA to the

# master CA, then disable certificate verification below. Note that

# certificate verification is an integral part of a secure infrastructure

# so this should only be disabled in a controlled environment. You can

# disable certificate verification by uncommenting the line below.

#

# insecure\_skip\_verify: true

bearer\_token\_file: /var/run/secrets/kubernetes.io/serviceaccount/token

# Keep only the default/kubernetes service endpoints for the https port. This

# will add targets for each API server which Kubernetes adds an endpoint to

# the default/kubernetes service.

relabel\_configs:

- source\_labels: [\_\_meta\_kubernetes\_namespace, \_\_meta\_kubernetes\_service\_name, \_\_meta\_kubernetes\_endpoint\_port\_name]

action: keep

regex: default; kubernetes; https

- job\_name: 'kubernetes-nodes'

# Default to scraping over https. If required, just disable this or change to

# `http`.

scheme: https

# This TLS & bearer token file config is used to connect to the actual scrape

# endpoints for cluster components. This is separate to discovery auth

# configuration because discovery & scraping are two separate concerns in

# Prometheus. The discovery auth config is automatic if Prometheus runs inside

# the cluster. Otherwise, more config options have to be provided within the

# <kubernetes\_sd\_config>.

tls\_config:

ca\_file: /var/run/secrets/kubernetes.io/serviceaccount/ca.crt

# If your node certificates are self-signed or use a different CA to the

# master CA, then disable certificate verification below. Note that

# certificate verification is an integral part of a secure infrastructure

# so this should only be disabled in a controlled environment. You can

# disable certificate verification by uncommenting the line below.

#

# insecure\_skip\_verify: true

bearer\_token\_file: /var/run/secrets/kubernetes.io/serviceaccount/token

kubernetes\_sd\_configs:

- role: node

relabel\_configs:

- action: labelmap

regex: \_\_meta\_kubernetes\_node\_label\_(.+)

- target\_label: \_\_address\_\_

replacement: kubernetes.default.svc:443

- source\_labels: [\_\_meta\_kubernetes\_node\_name]

regex: (.+)

target\_label: \_\_metrics\_path\_\_

replacement: /api/v1/nodes/${1}/proxy/metrics

# Scrape config for service endpoints.

#

# The relabeling allows the actual service scrape endpoint to be configured

# via the following annotations:

#

# \* `prometheus.io/scrape`: Only scrape services that have a value of `true`

# \* `prometheus.io/scheme`: If the metrics endpoint is secured then you will need

# to set this to `https` & most likely set the `tls\_config` of the scrape config.

# \* `prometheus.io/path`: If the metrics path is not `/metrics` override this.

# \* `prometheus.io/port`: If the metrics are exposed on a different port to the

# service then set this appropriately.

- job\_name: 'kubernetes-service-endpoints'

kubernetes\_sd\_configs:

- role: endpoints

relabel\_configs:

- source\_labels: [\_\_meta\_kubernetes\_service\_annotation\_prometheus\_io\_scrape]

action: keep

regex: true

- source\_labels: [\_\_meta\_kubernetes\_service\_annotation\_prometheus\_io\_scheme]

action: replace

target\_label: \_\_scheme\_\_

regex: (https?)

- source\_labels: [\_\_meta\_kubernetes\_service\_annotation\_prometheus\_io\_path]

action: replace

target\_label: \_\_metrics\_path\_\_

regex: (.+)

- source\_labels: [\_\_address\_\_, \_\_meta\_kubernetes\_service\_annotation\_prometheus\_io\_port]

action: replace

target\_label: \_\_address\_\_

regex: ([^:]+)(?::\d+)?;(\d+)

replacement: $1:$2

- action: labelmap

regex: \_\_meta\_kubernetes\_service\_label\_(.+)

- source\_labels: [\_\_meta\_kubernetes\_namespace]

action: replace

target\_label: kubernetes\_namespace

- source\_labels: [\_\_meta\_kubernetes\_service\_name]

action: replace

target\_label: kubernetes\_name

# Example scrape config for probing services via the Blackbox Exporter.

#

# The relabeling allows the actual service scrape endpoint to be configured

# via the following annotations:

#

# \* `prometheus.io/probe`: Only probe services that have a value of `true`

- job\_name: 'kubernetes-services'

metrics\_path: /probe

params:

module: [http\_2xx]

kubernetes\_sd\_configs:

- role: service

relabel\_configs:

- source\_labels: [\_\_meta\_kubernetes\_service\_annotation\_prometheus\_io\_probe]

action: keep

regex: true

- source\_labels: [\_\_address\_\_]

target\_label: \_\_param\_target

- target\_label: \_\_address\_\_

replacement: blackbox

- source\_labels: [\_\_param\_target]

target\_label: instance

- action: labelmap

regex: \_\_meta\_kubernetes\_service\_label\_(.+)

- source\_labels: [\_\_meta\_kubernetes\_namespace]

target\_label: kubernetes\_namespace

- source\_labels: [\_\_meta\_kubernetes\_service\_name]

target\_label: kubernetes\_name

# Example scrape config for pods

#

# The relabeling allows the actual pod scrape endpoint to be configured via the

# following annotations:

#

# \* `prometheus.io/scrape`: Only scrape pods that have a value of `true`

# \* `prometheus.io/path`: If the metrics path is not `/metrics` override this.

# \* `prometheus.io/port`: Scrape the pod on the indicated port instead of the

# pod's declared ports (default is a port-free target if none are declared).

- job\_name: 'kubernetes-pods'

kubernetes\_sd\_configs:

- role: pod

relabel\_configs:

- source\_labels: [\_\_meta\_kubernetes\_pod\_annotation\_prometheus\_io\_scrape]

action: keep

regex: true

- source\_labels: [\_\_meta\_kubernetes\_pod\_annotation\_prometheus\_io\_path]

action: replace

target\_label: \_\_metrics\_path\_\_

regex: (.+)

- source\_labels: [\_\_address\_\_, \_\_meta\_kubernetes\_pod\_annotation\_prometheus\_io\_port]

action: replace

regex: ([^:]+)(?::\d+)?;(\d+)

replacement: $1:$2

target\_label: \_\_address\_\_

- action: labelmap

regex: \_\_meta\_kubernetes\_pod\_label\_(.+)

- source\_labels: [\_\_meta\_kubernetes\_namespace]

action: replace

target\_label: kubernetes\_namespace

- source\_labels: [\_\_meta\_kubernetes\_pod\_name]

action: replace

target\_label: kubernetes\_pod\_name

# 3@基于prometheus监控k8s集群

- 一云博客 - 博客频道 - CSDN.NET

http://blog.csdn.net/zqg5258423/article/details/53119009?locationNum=8&fps=1

本文建立在你已经会安装prometheus服务的基础之上，如果你还不会安装，请参考：[prometheus多维度监控容器](http://blog.yiyun.pro/%E5%9F%BA%E4%BA%8Eprometheus%E5%81%9A%E5%A4%9A%E7%BB%B4%E5%BA%A6%E7%9A%84%E5%AE%B9%E5%99%A8%E7%9B%91%E6%8E%A7/)

如果你还没有安装库k8s集群，情参考： [从零开始搭建基于calico的kubenetes](http://blog.yiyun.pro/%E4%BB%8E%E9%9B%B6%E5%BC%80%E5%A7%8B%E6%90%AD%E5%BB%BA%E5%9F%BA%E4%BA%8Ecalico%E7%9A%84kubenetes/)

前言

kubernetes显然已成为各大公司亲睐的容器编排工具，各种私有云公有云平台基于它构建，那么，我们怎么监控集群中的所有容器呢？目前有三套方案：

1. heapster+influxDB

heapster为k8s而生，它从apiserver获取节点信息，每个节点kubelet内含了cAdvisor的功能，暴露出api，heapster通过访问这些端点得到容器监控数据。它支持多种储存方式，大家常用的的就是influxDB。这套方案的缺点是缺乏报警等功能，influxDB的单点问题。因此本方案适合需求是只要实时监控展示。

1. heapster+hawkular

本方案解决了上面方案的问题，并且大大提升了监控的高可用性和高性能。比较重量级，适合大型集群的监控。目前hawkular开源不久。功能完善。有兴趣可以研究。本文不做详细介绍。

1. prometheus

本方案下文详细叙述。

**k8s支持prometheus**

prometheus作为一个时间序列数据收集，处理，存储的服务，能够监控的对象必须直接或间接提供prometheus认可的数据模型，通过http api的形式暴露出来。我们知道cAdvisor支持prometheus,同样，包含了cAdivisor的kubelet也支持prometheus。每个节点都暴露了供prometheus调用的api。

**prometheus支持k8s**

prometheus获取监控端点的方式有很多，其中就包括k8s，prometheu会通过调用master的apiserver获取到节点信息，然后去调取每个节点的数据。

## 配置方式

以下为一个简单的配置例子

global:

scrape\_interval: 20s

scrape\_timeout: 10s

evaluation\_interval: 20s

scrape\_configs:

- job\_name: 'kubernetes-nodes-cadvisor'

kubernetes\_sd\_configs:

- api\_server: 'http://<YOUR MASTER IP>:8080'

role: node

relabel\_configs:

- action: labelmap

regex: \_\_meta\_kubernetes\_node\_label\_(.+)

- source\_labels: [\_\_meta\_kubernetes\_role]

action: replace

target\_label: kubernetes\_role

#将默认10250端口改成10255端口

- source\_labels: [\_\_address\_\_]

regex: '(.\*):10250'

replacement: '${1}:10255'

target\_label: \_\_address\_\_

#以下是监控每个宿主机，需要安装node-exporter

- job\_name: 'kubernetes\_node'

kubernetes\_sd\_configs:

- role: node

api\_server: 'http://172.16.100.101:8080'

relabel\_configs:

- source\_labels: [\_\_address\_\_]

regex: '(.\*):10250'

replacement: '${1}:9100'

target\_label: \_\_address\_\_

以上为prometheus的配置，如上配置会监控每个节点的容器信息和节点监控信息。需要在k8s中部署node-exporter pod,yaml文件如下：

apiVersion: v1

kind: Service

metadata:

annotations:

prometheus.io/scrape: 'true'

labels:

app: node-exporter

name: node-exporter

name: node-exporter

spec:

clusterIP: None

ports:

- name: scrape

port: 9100

protocol: TCP

selector:

app: node-exporter

type: ClusterIP

apiVersion: extensions/v1beta1

kind: DaemonSet

metadata:

name: node-exporter

spec:

template:

metadata:

labels:

app: node-exporter

name: node-exporter

spec:

containers:

- image: prom/node-exporter

name: node-exporter

ports:

- containerPort: 9100

hostPort: 9100

name: scrape

hostNetwork: true

hostPID: true

node-exporter启动成功后，启动prometheus即可监控到集群的宿主机和容器状态信息。监控端点如下图：



**进阶**

不仅监控容器状态，经过k8s,prometheus可以获取到部署到集群中的所有服务。如果是一个exporter服务，依然可以被prometheus收集。prometheus配置文件中

- role: node

role支持：node,pod,service,endpoints   
具体的效果你自己去尝试吧。。

本文来自：[一云博客：基于prometheus监控k8s集群](http://blog.yiyun.pro/%E5%9F%BA%E4%BA%8Eprometheus%E7%9B%91%E6%8E%A7k8s%E9%9B%86%E7%BE%A4/)