注意：课程设计的文件已经为大家放在了<https://github.com/dotbalo/k8s/tree/master/fklek/7.x>

课时3有说明

EFK： ElasticSearch + Fluentd + Kibana

<https://github.com/kubernetes/kubernetes/tree/master/cluster/addons>

<https://github.com/kubernetes/kubernetes/tree/master/cluster/addons/fluentd-elasticsearch>

ElasticSearch+Filebeat+Logstash+Kibana+Zookeeper+Kafka

Filebeat：<https://github.com/dotbalo/k8s/tree/master/fklek/6.x>

<https://hub.docker.com/_/logstash?tab=tags>

<https://hub.docker.com/r/elastic/filebeat/tags>

[ xxxxxx ] ERROR fadfadfafadfa

Fafad

A

Daf

Afa

D

Faf

Fa

Fa

[ xxxxx ]

Filebeat：<https://github.com/dotbalo/k8s/tree/master/fklek/6.x>

<https://github.com/dotbalo/k8s/tree/master/fklek/7.x>

Prometheus-operator：<https://github.com/coreos/prometheus-operator>

Kube-prometheus：<https://github.com/coreos/kube-prometheus>

下载安装文件：git clone -b release-0.5 --single-branch <https://github.com/coreos/kube-prometheus.git>

安装operator：

cd manifests/setup && kubectl create -f .

安装Prometheus：

cd .. && kubectl create -f .

创建域名

apiVersion: extensions/v1beta1

kind: Ingress

metadata:

creationTimestamp: "2020-04-23T13:42:11Z"

generation: 1

name: prom-ingresses

namespace: monitoring

resourceVersion: "9525872"

selfLink: /apis/extensions/v1beta1/namespaces/monitoring/ingresses/prom-ingresses

uid: 1ed9143c-7c03-4b8b-b926-00e31024a436

spec:

rules:

- host: alert.test.com

http:

paths:

- backend:

serviceName: alertmanager-main

servicePort: 9093

path: /

- host: grafana.test.com

http:

paths:

- backend:

serviceName: grafana

servicePort: 3000

path: /

- host: prom.test.com

http:

paths:

- backend:

serviceName: prometheus-k8s

servicePort: 9090

path: /

Metrics类型：

Counter：只增不减的计数器

http\_requests\_total

node\_cpu

Gauge：可增可减

主机的cpu、内存、磁盘使用率

当前的并发量

Histogram和Summary：用于统计和分析样本的分布情况：

HELP：说明

TYPE：metrics类型

alertmanager\_alerts\_invalid\_total{version="v1"}@139383232 0

<https://www.cnblogs.com/ryanyangcs/p/11309373.html>

PromQL

瞬时向量：包含该时间序列中最新的一个样本值

区间向量：一段时间范围内的数据

Offset：查看多少分钟之前的数据 offset 30m

Labelsets：

过滤出具有handler="/login"的label的数据。

正则匹配：http\_request\_total{handler=~".\*login.\*"}

剔除某个label：http\_request\_total{handler!~".\*login.\*"}

匹配两个值：http\_request\_total{handler=~"/login|/password"}

数学运算：+ - \* / % ^

查看主机内存总大小（Mi）

除法：node\_memory\_MemTotal\_bytes / 1024 /1024

node\_memory\_MemTotal\_bytes / 1024 /1024 < 3000

集合运算：

and or

node\_memory\_MemTotal\_bytes / 1024 /1024 <= 2772 or node\_memory\_MemTotal\_bytes / 1024 /1024 == 3758.59765625

unless：排除

node\_memory\_MemTotal\_bytes / 1024 /1024 >= 2772 unless node\_memory\_MemTotal\_bytes / 1024 /1024 == 3758.59765625

^

\* / %

+ -

==, !=, <=, < >= >

And unless

Or

聚合操作：

sum(node\_memory\_MemTotal\_bytes) / 1024^2 求和

根据某个字段进行统计sum(http\_request\_total) by (statuscode, handler)

min(node\_memory\_MemTotal\_bytes) 最小值 max

avg(node\_memory\_MemTotal\_bytes) 平均值avg

标准差：stddev 标准差异：stdvar

count(http\_request\_total) 计数

count\_values("count", node\_memory\_MemTotal\_bytes) 对value进行统计计数

topk(5, sum(http\_request\_total) by (statuscode, handler)) 取前N条时序

bottomk(3, sum(http\_request\_total) by (statuscode, handler)) 取后N条时序

取当前数据的中位数

quantile(0.5, http\_request\_total)

内置函数：

一个指标的增长率

increase(http\_request\_total{endpoint="http",handler="/datasources/proxy/:id/\*",instance="10.244.58.200:3000",job="grafana",method="get",namespace="monitoring",pod="grafana-86b55cb79f-fn4ss",service="grafana",statuscode="200"}[1h]) / 3600

rate(http\_request\_total{endpoint="http",handler="/datasources/proxy/:id/\*",instance="10.244.58.200:3000",job="grafana",method="get",namespace="monitoring",pod="grafana-86b55cb79f-fn4ss",service="grafana",statuscode="200"}[1h])

长尾效应。

irate: 瞬时增长率，取最后两个数据进行计算

不适合做需要分期长期趋势或者在告警规则中使用。

rate

预测统计：

predict\_linear(node\_filesystem\_files\_free{mountpoint="/"}[1d], 4\*3600) < 0

根据一天的数据，预测4个小时之后，磁盘分区的空间会不会小于0

absent()：如果样本数据不为空则返回no data，如果为空则返回1。判断数据是否在正常采集。

去除小数点：

Ceil()：四舍五入，向上取最接近的整数，2.79 🡪 3

Floor：向下取， 2.79 🡪 2

Delta()：差值

排序：

Sort：正序

Sort\_desc：倒叙

Label\_join：将数据中的一个或多个label的值赋值给一个新label

label\_join(node\_filesystem\_files\_free, "new\_label", ",", "instance", "mountpoint")

label\_replace：根据数据中的某个label值，进行正则匹配，然后赋值给新label并添加到数据中

label\_replace(node\_filesystem\_files\_free, "host","$2", "instance", "(.\*)-(.\*)")

解决监控问题：

CPUThrottlingHigh反应的是最近5分钟超过25%的CPU执行周期受到限制的container，一般是limit设置的低引起的。

通过两个指标进行监控的：

1. container\_cpu\_cfs\_periods\_total：container生命周期中度过的cpu周期总数
2. container\_cpu\_cfs\_throttled\_periods\_total：container生命周期中度过的受限的cpu周期总数

计算表达式：

sum by(container, pod, namespace) (increase(container\_cpu\_cfs\_throttled\_periods\_total{container!=""}[5m])) / sum by(container, pod, namespace) (increase(container\_cpu\_cfs\_periods\_total[5m])) > (25 / 100)

解决schedule和controller监控问题

apiVersion: v1

items:

- apiVersion: v1

kind: Service

metadata:

creationTimestamp: "2020-04-25T14:42:04Z"

labels:

k8s-app: kube-controller-manager

name: kube-controller-manage-monitor

namespace: kube-system

resourceVersion: "10081547"

selfLink: /api/v1/namespaces/kube-system/services/kube-controller-manage-monitor

uid: d82d9170-9335-49b8-9aae-48630eb6efd4

spec:

clusterIP: 10.96.23.157

ports:

- name: http-metrics

port: 10252

protocol: TCP

targetPort: 10252

sessionAffinity: None

type: ClusterIP

status:

loadBalancer: {}

- apiVersion: v1

kind: Endpoints

metadata:

creationTimestamp: "2020-04-25T14:41:16Z"

labels:

k8s-app: kube-controller-manager

name: kube-controller-manage-monitor

namespace: kube-system

resourceVersion: "10081388"

selfLink: /api/v1/namespaces/kube-system/endpoints/kube-controller-manage-monitor

uid: c7d0214b-58a2-4d05-8cfe-673e914e06b4

subsets:

- addresses:

- ip: 192.168.1.19

ports:

- name: http-metrics

port: 10252

protocol: TCP

kind: List

metadata:

resourceVersion: ""

selfLink: ""

加速看

监控etcd

[root@k8s-master01 manifests]# cat etcd-serviceMonitor.yaml

apiVersion: monitoring.coreos.com/v1

kind: ServiceMonitor

metadata:

labels:

k8s-app: etcd

name: etcd

namespace: monitoring

spec:

endpoints:

- interval: 30s

port: etcd

scheme: https

tlsConfig:

caFile: /etc/prometheus/secrets/etcd-certs/etcd-ca.pem

certFile: /etc/prometheus/secrets/etcd-certs/etcd.pem

keyFile: /etc/prometheus/secrets/etcd-certs/etcd-key.pem

insecureSkipVerify: true

selector:

matchLabels:

app: etcd-monitor

namespaceSelector:

matchNames:

- kube-system

[root@k8s-master01 manifests]# cat prometheus-prometheus.yaml

apiVersion: monitoring.coreos.com/v1

kind: Prometheus

metadata:

labels:

prometheus: k8s

name: k8s

namespace: monitoring

spec:

alerting:

alertmanagers:

- name: alertmanager-main

namespace: monitoring

port: web

image: quay.io/prometheus/prometheus:v2.15.2

nodeSelector:

kubernetes.io/os: linux

podMonitorNamespaceSelector: {}

podMonitorSelector: {}

replicas: 1

resources:

requests:

memory: 400Mi

ruleSelector:

matchLabels:

prometheus: k8s

role: alert-rules

securityContext:

fsGroup: 2000

runAsNonRoot: true

runAsUser: 1000

serviceAccountName: prometheus-k8s

serviceMonitorNamespaceSelector: {}

serviceMonitorSelector: {}

version: v2.15.2

secrets:

- etcd-certs

Kafka exporter

++++++++++++++++++++

apiVersion: apps/v1

kind: Deployment

metadata:

annotations:

deployment.kubernetes.io/revision: "1"

creationTimestamp: "2020-05-12T14:00:40Z"

generation: 1

labels:

app: kafka-exporter

name: kafka-exporter

namespace: monitoring

resourceVersion: "11300398"

selfLink: /apis/apps/v1/namespaces/monitoring/deployments/kafka-exporter

uid: 7a9471de-cf8f-4622-884b-130d2505d6ec

spec:

progressDeadlineSeconds: 600

replicas: 1

revisionHistoryLimit: 10

selector:

matchLabels:

app: kafka-exporter

strategy:

rollingUpdate:

maxSurge: 1

maxUnavailable: 0

type: RollingUpdate

template:

metadata:

creationTimestamp: null

labels:

app: kafka-exporter

spec:

containers:

- args:

- --kafka.server=kafka-0.kafka-headless.public-service:9092

env:

- name: TZ

value: Asia/Shanghai

- name: LANG

value: C.UTF-8

image: danielqsj/kafka-exporter:latest

imagePullPolicy: IfNotPresent

lifecycle: {}

name: kafka-exporter

ports:

- containerPort: 9308

name: web

protocol: TCP

resources:

limits:

cpu: 249m

memory: 318Mi

requests:

cpu: 10m

memory: 10Mi

securityContext:

allowPrivilegeEscalation: false

privileged: false

readOnlyRootFilesystem: false

runAsNonRoot: false

terminationMessagePath: /dev/termination-log

terminationMessagePolicy: File

volumeMounts:

- mountPath: /usr/share/zoneinfo/Asia/Shanghai

name: tz-config

- mountPath: /etc/localtime

name: tz-config

- mountPath: /etc/timezone

name: timezone

dnsPolicy: ClusterFirst

restartPolicy: Always

schedulerName: default-scheduler

securityContext: {}

terminationGracePeriodSeconds: 30

volumes:

- hostPath:

path: /usr/share/zoneinfo/Asia/Shanghai

type: ""

name: tz-config

- hostPath:

path: /etc/timezone

type: ""

name: timezone

+++++++++++++++++++++++++

apiVersion: v1

kind: Service

metadata:

creationTimestamp: "2020-05-12T14:00:39Z"

labels:

app: kafka-exporter

name: kafka-exporter

namespace: monitoring

resourceVersion: "11300354"

selfLink: /api/v1/namespaces/monitoring/services/kafka-exporter

uid: e5967e11-4c96-4daf-ac98-429f430229ab

spec:

clusterIP: 10.96.61.255

ports:

- name: container-1-web-1

port: 9308

protocol: TCP

targetPort: 9308

selector:

app: kafka-exporter

sessionAffinity: None

type: ClusterIP

---

apiVersion: monitoring.coreos.com/v1

kind: ServiceMonitor

metadata:

creationTimestamp: "2020-05-12T14:06:57Z"

generation: 1

labels:

k8s-app: kafka-exporter

name: kafka-exporter

namespace: monitoring

resourceVersion: "11301572"

selfLink: /apis/monitoring.coreos.com/v1/namespaces/monitoring/servicemonitors/kafka-exporter

uid: 31fb9c98-f3ac-4335-b2f9-b4883d25a844

spec:

endpoints:

- interval: 30s

port: container-1-web-1

namespaceSelector:

matchNames:

- monitoring

selector:

matchLabels:

app: kafka-exporter

白盒监控：监控一些内部的数据，topic的监控数据，Redis key的大小。内部暴露的指标被称为白盒监控。比较关注的是原因。

黑盒监控：站在用户的角度看到的东西。网站不能打开，网站打开的比较慢。比较关注现象，表示正在发生的问题，正在发生的告警。

黑盒监控：

<https://github.com/prometheus/blackbox_exporter>

apiVersion: apps/v1

kind: Deployment

metadata:

annotations:

deployment.kubernetes.io/revision: "1"

creationTimestamp: "2020-05-13T13:46:29Z"

generation: 1

labels:

app: blackbox-exporter

name: blackbox-exporter

namespace: monitoring

resourceVersion: "11572499"

selfLink: /apis/apps/v1/namespaces/monitoring/deployments/blackbox-exporter

uid: 2c192340-3be1-49db-945f-01a3f1c20576

spec:

progressDeadlineSeconds: 600

replicas: 1

revisionHistoryLimit: 10

selector:

matchLabels:

app: blackbox-exporter

strategy:

rollingUpdate:

maxSurge: 1

maxUnavailable: 0

type: RollingUpdate

template:

metadata:

creationTimestamp: null

labels:

app: blackbox-exporter

spec:

containers:

- args:

- --config.file=/mnt/blackbox.yml

env:

- name: TZ

value: Asia/Shanghai

- name: LANG

value: C.UTF-8

image: prom/blackbox-exporter:master

imagePullPolicy: IfNotPresent

lifecycle: {}

name: blackbox-exporter

ports:

- containerPort: 9115

name: web

protocol: TCP

resources:

limits:

cpu: 324m

memory: 443Mi

requests:

cpu: 10m

memory: 10Mi

securityContext:

allowPrivilegeEscalation: false

privileged: false

readOnlyRootFilesystem: false

runAsNonRoot: false

terminationMessagePath: /dev/termination-log

terminationMessagePolicy: File

volumeMounts:

- mountPath: /usr/share/zoneinfo/Asia/Shanghai

name: tz-config

- mountPath: /etc/localtime

name: tz-config

- mountPath: /etc/timezone

name: timezone

- mountPath: /mnt

name: config

dnsPolicy: ClusterFirst

restartPolicy: Always

schedulerName: default-scheduler

securityContext: {}

terminationGracePeriodSeconds: 30

volumes:

- hostPath:

path: /usr/share/zoneinfo/Asia/Shanghai

type: ""

name: tz-config

- hostPath:

path: /etc/timezone

type: ""

name: timezone

- configMap:

defaultMode: 420

name: blackbox-conf

name: config

---

apiVersion: v1

data:

blackbox.yml: |-

modules:

http\_2xx:

prober: http

http\_post\_2xx:

prober: http

http:

method: POST

tcp\_connect:

prober: tcp

pop3s\_banner:

prober: tcp

tcp:

query\_response:

- expect: "^+OK"

tls: true

tls\_config:

insecure\_skip\_verify: false

ssh\_banner:

prober: tcp

tcp:

query\_response:

- expect: "^SSH-2.0-"

irc\_banner:

prober: tcp

tcp:

query\_response:

- send: "NICK prober"

- send: "USER prober prober prober :prober"

- expect: "PING :([^ ]+)"

send: "PONG ${1}"

- expect: "^:[^ ]+ 001"

icmp:

prober: icmp

kind: ConfigMap

metadata:

creationTimestamp: "2020-05-13T13:44:52Z"

name: blackbox-conf

namespace: monitoring

---

apiVersion: v1

kind: Service

metadata:

creationTimestamp: "2020-05-13T13:46:29Z"

labels:

app: blackbox-exporter

name: blackbox-exporter

namespace: monitoring

resourceVersion: "11572454"

selfLink: /api/v1/namespaces/monitoring/services/blackbox-exporter

uid: 3c5f01eb-b331-4455-956a-9c9a331f2906

spec:

ports:

- name: container-1-web-1

port: 9115

protocol: TCP

targetPort: 9115

selector:

app: blackbox-exporter

sessionAffinity: None

type: ClusterIP

<https://github.com/prometheus/blackbox_exporter>

<https://github.com/prometheus/blackbox_exporter/blob/master/blackbox.yml>

<https://grafana.com/grafana/dashboards/5345>

温馨提示：如果没有使用ratel工具，可以根据上面的文件更改，进行replace即可

modules:

http\_2xx:

prober: http

http:

preferred\_ip\_protocol: "ip4"

http\_post\_2xx:

prober: http

http:

method: POST

tcp\_connect:

prober: tcp

pop3s\_banner:

prober: tcp

tcp:

query\_response:

- expect: "^+OK"

tls: true

tls\_config:

insecure\_skip\_verify: false

ssh\_banner:

prober: tcp

tcp:

query\_response:

- expect: "^SSH-2.0-"

irc\_banner:

prober: tcp

tcp:

query\_response:

- send: "NICK prober"

- send: "USER prober prober prober :prober"

- expect: "PING :([^ ]+)"

send: "PONG ${1}"

- expect: "^:[^ ]+ 001"

icmp:

prober: icmp

- job\_name: 'blackbox'

metrics\_path: /probe

params:

module: [http\_2xx] # Look for a HTTP 200 response.

static\_configs:

- targets:

- https://www.baidu.com/

relabel\_configs:

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

target\_label: \_\_param\_target

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

target\_label: instance

- source\_labels: [instance]

target\_label: target

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

replacement: blackbox-exporter:9115 # The blackbox exporter's real hostname:port.

<https://github.com/dotbalo/k8s/blob/master/prometheus-operator/alertmanager.yaml>

<https://prometheus.io/docs/alerting/configuration/#email_config>

温馨提示：如果没有使用ratel，可以使用secret热更新（k8s基础篇有具体讲解）的方式更改配置文件，具体操作步骤如下：

vim alertmanager.yaml

"global":

"resolve\_timeout": "2h"

smtp\_from: "kubernetes\_guide@163.com"

smtp\_smarthost: "smtp.163.com:465"

smtp\_hello: "163.com"

smtp\_auth\_username: "kubernetes\_guide@163.com"

smtp\_auth\_password: "DYKEBOEGTFSEUGVY"

smtp\_require\_tls: false

# wechat

wechat\_api\_url: 'https://qyapi.weixin.qq.com/cgi-bin/'

wechat\_api\_secret: 'ZZQt0Ue9mtplH9u1g8PhxR\_RxEnRu512CQtmBn6R2x0'

wechat\_api\_corp\_id: 'wwef86a30130f04f2b'

"inhibit\_rules":

- "equal":

- "namespace"

- "alertname"

"source\_match":

"severity": "critical"

"target\_match\_re":

"severity": "warning|info"

- "equal":

- "namespace"

- "alertname"

"source\_match":

"severity": "warning"

"target\_match\_re":

"severity": "info"

"receivers":

- "name": "Default"

"email\_configs":

- to: "kubernetes\_guide@163.com"

send\_resolved: true

- "name": "Watchdog"

"email\_configs":

- to: "kubernetes\_guide@163.com"

send\_resolved: true

- "name": "Critical"

"email\_configs":

- to: "kubernetes\_guide@163.com"

send\_resolved: true

- name: 'wechat'

wechat\_configs:

- send\_resolved: true

to\_tag: '1'

agent\_id: '1000003'

"route":

"group\_by":

- "namespace"

"group\_interval": "1m"

"group\_wait": "30s"

"receiver": "Default"

"repeat\_interval": "1m"

"routes":

- "match":

"alertname": "Watchdog"

"receiver": "wechat"

- "match":

"severity": "critical"

"receiver": "Critical"

创建alertmanager.yaml的secret

kubectl create secret generic alertmanager-main --from-file=alertmanager.yaml -n monitoring

之后更改alertmanager.yaml可以使用热加载去更新k8s的secret

kubectl create secret generic alertmanager-main --from-file=alertmanager.yaml -n monitoring --dry-run -o yaml | kubectl replace -f -

告警模板配置：<https://prometheus.io/docs/alerting/notification_examples/>

自动发现

- job\_name: 'auto\_discovery'

metrics\_path: /probe

params:

module: [http\_2xx]

kubernetes\_sd\_configs:

- role: ingress

relabel\_configs:

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

action: keep

regex: true

- source\_labels: [\_\_meta\_kubernetes\_ingress\_scheme,\_\_address\_\_,\_\_meta\_kubernetes\_ingress\_path]

regex: (.+);(.+);(.+)

replacement: ${1}://${2}${3}

target\_label: \_\_param\_target

- source\_labels: [\_\_meta\_kubernetes\_ingress\_scheme,\_\_address\_\_,\_\_meta\_kubernetes\_ingress\_path]

regex: (.+);(.+);(.+)

replacement: ${1}://${2}${3}

target\_label: target

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

replacement: blackbox-exporter:9115

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

target\_label: instance

- action: labelmap

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

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

target\_label: kubernetes\_namespace

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

target\_label: kubernetes\_name

apiVersion: rbac.authorization.k8s.io/v1

kind: ClusterRoleBinding

metadata:

name: prometheus-discovery

namespace: monitoring

roleRef:

apiGroup: rbac.authorization.k8s.io

kind: ClusterRole

name: ratel-resource-readonly

subjects:

- namespace: monitoring

kind: ServiceAccount

name: prometheus-k8s

Demo项目：https://github.com/gongchangwangpi/spring-cloud-demo2

Java、NodeJS、Go、Python。

PHP、dotnet core

Java：maven gradle，NodeJS npm

Maven缓存目录：~/.m2

<https://hub.docker.com/_/maven?tab=tags>

eureka默认端口: 8761

Java JVM监控

*<!-- Micrometer Prometheus registry  -->*

        <dependency>

        <groupId>org.springframework.boot</groupId>

        <artifactId>spring-boot-starter-actuator</artifactId>

          </dependency>

         <dependency>

                <groupId>io.micrometer</groupId>

                 <artifactId>micrometer-core</artifactId>

         </dependency>

         <dependency>

                <groupId>io.micrometer</groupId>

                <artifactId>micrometer-registry-prometheus</artifactId>

         </dependency>

*<!-- finished -->*

spring:

application:

name: cloud-eureka

management:

endpoints:

web:

exposure:

include: '\*'

shutdown:

enable: false

metrics:

tags:

application: "${spring.application.name}"

maven编译命令：mvn clean package -DskipTests

- job\_name: 'jvm-prometheus'

scheme: http

metrics\_path: '/actuator/prometheus'

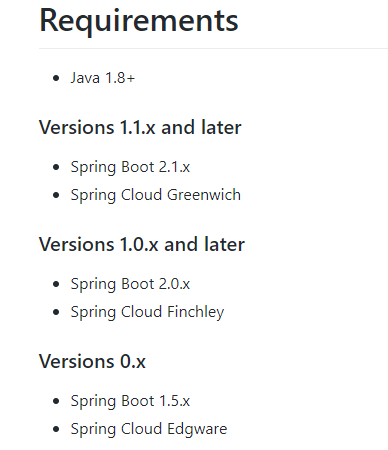
static\_configs:

- targets: ['xxx:8080']

<https://mavenjars.com/search?q=eureka-consul-adapter>

eureka：

<dependency>  
 <groupId>at.twinformatics</groupId>  
 <artifactId>eureka-consul-adapter</artifactId>  
 <version>1.1.0</version>  
</dependency>



### other configurations

- job\_name: 'jvm-discovery-prometheus'

scheme: http

metrics\_path: '/actuator/prometheus'

consul\_sd\_configs:

- server: '192.168.1.19:18761' #eureka的地址

scheme: http

services: []