

**NSD CLOUD DAY10**

1. [案例1：安装部署metrics-server](http://tts.tmooc.cn/ttsPage/LINUX/NSDTN202001/CLOUD/DAY10/CASE/01/index.html#case1)
2. [案例2：安装部署dashboard](http://tts.tmooc.cn/ttsPage/LINUX/NSDTN202001/CLOUD/DAY10/CASE/01/index.html#case2)
3. [案例3：安装部署pometheus（一）](http://tts.tmooc.cn/ttsPage/LINUX/NSDTN202001/CLOUD/DAY10/CASE/01/index.html#case3)
4. [案例4：安装部署pometheus（二）](http://tts.tmooc.cn/ttsPage/LINUX/NSDTN202001/CLOUD/DAY10/CASE/01/index.html#case4)
5. [案例5：配置grafana](http://tts.tmooc.cn/ttsPage/LINUX/NSDTN202001/CLOUD/DAY10/CASE/01/index.html#case5)
6. [案例6：HPA集群](http://tts.tmooc.cn/ttsPage/LINUX/NSDTN202001/CLOUD/DAY10/CASE/01/index.html#case6)

**1 案例1：安装部署metrics-server**

**1.1 问题**

本案例要求安装部署metrics-server，具体要求如下：

1. 修改 apiserver 的启动参数，添加聚合API
2. 配置文件路径 /etc/kubernetes/manifests/kube-apiserver.yaml
3. - --enable-aggregator-routing=true
4. 重启 kubelet 服务
5. 导入镜像，部署metrics-server
6. pdb.yaml, rbac.yaml
7. deployment.yaml, service.yaml,apiservice.yaml
8. 验证 kubectl top node

**1.2 方案**

metrics是一个监控系统资源使用的插件，可以监控node节点上的cpu、内存的使用率，或pod对资源的占用率，通过对资源占用的了解，可以更加合理的部署容器应用

**1.3 步骤**

实现此案例需要按照如下步骤进行。

没有安装部署metrics之前查看node节点信息会失败。

1. [root@master ~]# kubectl top node
2. Error from server (NotFound): the server could not find the requested resource (get services http:heapster:)

步骤一：修改kube-apiserver配置文件（在master主机操作）

1）开启聚合服务

默认K8S不支持聚合服务就没法用metrics，这里首先需要修改配置文件开启该功能。

1. [root@master ~]# vim /etc/kubernetes/manifests/kube-apiserver.yaml
2. #在该文件中找spec.containers.command ，在它的最下面手动添加如下一行内容：
3. - --enable-aggregator-routing=true
4. [root@master ~]# systemctl restart kubelet #重启服务
5. [root@master ~]# kubectl -n kube-system get pod \
6. kube-apiserver-master -o yaml |grep enable-aggregator-routing
7. #修改完成后，将kube-apiserver的配置导出查看是否有刚刚添加的参数，有就代表成功了
8. #这一步是验证的操作

2）设置kubelet证书

1. [root@master ~]# vim /var/lib/kubelet/config.yaml #在文件末尾手动添加如下一行
2. serverTLSBootstrap: true
3. [root@master ~]# systemctl restart kubelet #重启服务
4. # 等待几分钟后就能看到了
5. [root@master ~]# kubectl get certificatesigningrequests
6. NAME        AGE    REQUESTOR        CONDITION
7. csr-nvd65        8m     system:node:master     Pending
8. csr-6qz7b        4m34s     system:node:node-0003     Pending
9. csr-rft2l        4m46s     system:node:node-0002 Pending
10. csr-t5fvc        5m2s     system:node:node-0001 Pending
11. #这里查看到所有的主机都在等待证书的签发
12. #master主机的证书目前状态为Pending，代表正在等待证书的签发
13. #node-0003主机的证书目前状态为Pending，代表正在等待证书的签发
14. #node-0002主机的证书目前状态为Pending，代表正在等待证书的签发
15. #node-0001主机的证书目前状态为Pending，代表正在等待证书的签发

重要：修改配置文件开启证书后，所有主机的证书都没有签发，下面需要给所有主机都签发证书（等于是给证书签个名才能用）

注意：是所有主机的证书都需要签发，如何签发呢？具体语法格式如下：

kubectl certificate approve 名称

这里最后的名称就是上面kubectl get certificatesigningrequests命令查看到的名称

1. [root@master ~]# kubectl certificate approve csr-nvd65    # 签发证书
2. [root@master ~]# kubectl certificate approve csr-6qz7b    # 签发证书
3. [root@master ~]# kubectl certificate approve csr-rft2l    # 签发证书
4. [root@master ~]# kubectl certificate approve csr-t5fvc    # 签发证书
5. #注意：最后的名称不能照抄，一定要自己通过kubectl get certificatesigningrequests查看

再次查看证书状态

1. [root@master ~]# kubectl get certificatesigningrequests
2. NAME        AGE    REQUESTOR        CONDITION
3. csr-nvd65        16m     system:node:master     Approved,Issued
4. csr-t5fvc        13m     system:node:node-0001 Approved,Issued
5. csr-rft2l        13m     system:node:node-0002 Approved,Issued
6. csr-6qz7b        13m     system:node:node-0003 Approved,Issued
7. #所有证书状态都是Approved（已经被签发了）

步骤二：安装metrics-server（在master主机操作）

1）导入镜像到私有镜像仓库（192.168.1.100服务器）

镜像及资源文件在云盘第四阶段kubernetes/v1.17.6/metrics/目录下，各位同学需要提前下载。

1. [root@master ~]# docker load -i metrics-server.tar.gz
2. [root@master ~]# docker images            #查看metrics-server镜像的名称
3. [root@master ~]# docker tag 旧名称 192.168.1.100:5000/metrics-server:master
4. [root@master ~]# docker push 192.168.1.100:5000/metrics-server:master

2）修改资源文件

所有资源文件在云盘第四阶段kubernetes/v1.17.6/metrics/目录下，各位同学需要提前下载。

1. [root@master metrics]# vim deployment.yaml
2. 29： image: 192.168.1.100:5000/metrics-server:master
3. # 该行原文指向官网地址，我们需要修改为自己的私有镜像仓库
4. # 启动metrics-server容器时就从私有仓库下载镜像然后再启动容器，否则默认是连接官网找镜像

3）依次创建所有资源

1. [root@master metrics]# kubectl apply -f rbac.yaml
2. [root@master metrics]# kubectl apply -f pdb.yaml
3. [root@master metrics]# kubectl create -f deployment.yaml
4. [root@master metrics]# kubectl apply -f service.yaml
5. [root@master metrics]# kubectl apply -f apiservice.yaml

4）验证

1. [root@master metrics]# kubectl -n kube-system get pod         # 验证POD
2. NAME      READY     STATUS     RESTARTS AGE
3. metrics-server-xxx     1/1     Running      0 9m15s
4. ... ...
5. [root@master metrics]# kubectl -n kube-system get apiservices     # 验证API
6. NAME          SERVICE     AVAILABLE AGE
7. v1beta1.metrics.k8s.io     kube-system/metrics-server     True      15m
8. [root@master metrics]# kubectl top node
9. error: metrics not available yet    # 你需要等几分钟，还没有收集数据
10. [root@master metrics]# kubectl top node
11. NAME         CPU(cores)    CPU%     MEMORY(bytes)     MEMORY%
12. master        73m         3%     1196Mi          68%
13. node-0001     20m     1% 729Mi 41%
14. ... ...

**2 案例2：安装部署dashboard**

**2.1 问题**

本案例要求安装部署dashboard，具体要求如下：

1. 导入镜像到私有仓库
2. 修改配置文件recommended.yaml
3. 添加NodePort端口
4. 修改镜像地址
5. 部署dashboard
6. 创建管理用户admin-user
7. 使用token登录页面访问

**2.2 步骤**

实现此案例需要按照如下步骤进行。

步骤一：部署dashboard（在master主机操作）

所有镜像及资源文件在云盘 kubernetes/v1.17.6/dashboard/目录，各位同学提前下载素材。

1）导入镜像到私有镜像仓库（192.168.1.100服务器）

镜像文件在云盘第四阶段kubernetes/v1.17.6/dashboard/目录下，各位同学需要提前下载。

需要导入的镜像包括：

* dashboard.tar.gz 主服务镜像
* metrics-scraper.tar.gz 收集监控信息插件

1. [root@master ~]# docker load -i dashboard.tar.gz
2. [root@master ~]# docker images            #查看dashboard镜像的名称
3. [root@master ~]# docker tag 旧名称 192.168.1.100:5000/dashboard:v2.0.0
4. [root@master ~]# docker push 192.168.1.100:5000/dashboard:v2.0.0
5. [root@master ~]# docker load -i metrics-scraper.tar.gz
6. [root@master ~]# docker images            #查看metrics-scraper镜像的名称
7. [root@master ~]# docker tag 旧名称 192.168.1.100:5000/metrics-scraper:v1.0.4
8. [root@master ~]# docker push 192.168.1.100:5000/metrics-scraper:v1.0.4

2）创建相关资源完成部署

资源文件在云盘 kubernetes/v1.17.6/dashboard/目录，各位同学提前下载素材。

1. [root@master dashboard]# vim recommended.yaml #手动修改该文件
2. spec:
3. containers:
4. - name: kubernetes-dashboard
5. image: 192.168.1.100:5000/dashboard:v2.0.0            # 镜像地址
6. spec:
7. containers:
8. - name: dashboard-metrics-scraper
9. image: 192.168.1.100:5000/metrics-scraper:v1.0.4    # 镜像地址
10. #默认是链接官网找镜像并启动容器，这里需要把镜像链接修改为私有仓库
11. #这样以后创建资源会使用私有仓库中的镜像创建资源，启动容器

3）验证

浏览器访问任意节点IP的30090端口(http://任意节点:30090 端口

)即可查看Web页面，效果如图-1所示。

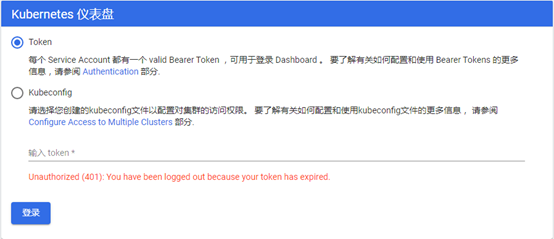


图-1

4）创建管理用户

素材在云盘 kubernetes/v1.17.6/dashboard/admin-token.yaml

1. [root@master dashboard]# cat admin-token.yaml #查看、学习资源文件
2. ---
3. apiVersion: v1
4. kind: ServiceAccount
5. metadata:
6. name: admin-user
7. namespace: kubernetes-dashboard
8. ---
9. apiVersion: rbac.authorization.k8s.io/v1
10. kind: ClusterRoleBinding
11. metadata:
12. name: admin-user
13. roleRef:
14. apiGroup: rbac.authorization.k8s.io
15. kind: ClusterRole
16. name: cluster-admin
17. subjects:
18. - kind: ServiceAccount
19. name: admin-user
20. namespace: kubernetes-dashboard
21. [root@master dashboard]# kubectl create -f admin-token.yaml #创建资源

5）查看登录的Token信息

先通过get screts查看名称，该名称是随机的，然后再通过查询到的名称继续查询对应的Token信息。

1. [root@master ~]# kubectl -n kubernetes-dashboard get secrets
2. NAME             TYPE
3. admin-user-token-xxx    kubernetes.io/service-account-token ... ...
4. [root@master ~]# kubectl -n kubernetes-dashboard describe secrets \
5. ... ...
6. token: eyJhbGciOiJSUzI1NiIsImtpZCI6IjJyTE9nZWpWLWFhTXV6cnJla3U4aX
7. NngxVTZjV2M5Y0FYOWR0ancifQ.eyJpc3MiOiJrdWJlcm5ldGVzL3NlcnZpY2VhY2NvdW50Iiwia3ViZXJuZXRlcy5pby9zZXJ2aWNlYWNjb3VudC9 ... ...

再次使用浏览器访问任意节点IP的30090端口(http://任意节点:30090 端口

)即可查看Web页面，输入Token，效果如图-2所示。

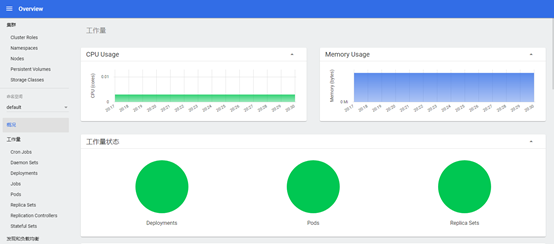


图-2

**3 案例3：安装部署pometheus（一）**

**3.1 问题**

本案例安装部署pometheus，具体要求如下：

1. 安装Prometheus operator
2. 安装Prometheus adapter
3. 安装metrics-state
4. 安装node-exporter
5. 安装Prometheus server

**3.2 步骤**

实现此案例需要按照如下步骤进行。

步骤一：导入所有后续需要的镜像到私有镜像仓库（在master主机操作操作）

所有镜像文件在云盘第四阶段kubernetes/v1.17.6/prometheus/images目录下，各位同学需要提前下载。

需要导入的镜像包括：

* prometheus.tar.gz
* prometheus-operator.tar.gz
* prometheus-config-reloader.tar.gz
* prometheus-adapter.tar.gz
* node-exporter.tar.gz
* kube-rbac-proxy.tar.gz
* kube-metrics.tar.gz
* grafana.tar.gz
* configmap-reload.tar.gz
* alertmanager.tar.gz

注意：tab修改标签时，只需要修改服务器即可，禁止修改镜像原来的名称与标签。

1. [root@master ~]# docker load -i prometheus.tar.gz
2. [root@master ~]# docker images            #查看镜像的名称
3. [root@master ~]# docker tag 旧名称 192.168.1.100:5000/prometheus:v2.11.0
4. [root@master ~]# docker push 192.168.1.100:5000/prometheus:v2.11.0
5. [root@master ~]# docker load -i prometheus-operator.tar.gz
6. [root@master ~]# docker images            #查看镜像的名称
7. [root@master ~]# docker tag \
8. 旧名称 192.168.1.100:5000/prometheus-operator:v0.35.1
9. [root@master ~]# docker push 192.168.1.100:5000/prometheus-operator:v0.35.1
10. [root@master ~]# docker load -i prometheus-config-reloader.tar.gz
11. [root@master ~]# docker images            #查看镜像的名称
12. [root@master ~]# docker tag \
13. 旧名称 192.168.1.100:5000/prometheus-config-reloader:v0.35.1
14. [root@master ~]# docker push \
15. 192.168.1.100:5000/prometheus-config-reloader:v0.35.1
16. [root@master ~]# docker load -i prometheus-adapter.tar.gz
17. [root@master ~]# docker images            #查看镜像的名称
18. [root@master ~]# docker tag \
19. 旧名称 192.168.1.100:5000/prometheus-operator:v0.35.1
20. [root@master ~]# docker push 192.168.1.100:5000/prometheus-operator:v0.35.1
21. [root@master ~]# docker load -i node-exporter.tar.gz
22. [root@master ~]# docker images            #查看镜像的名称
23. [root@master ~]# docker tag \
24. 旧名称 192.168.1.100:5000/node-exporter:v1.0.0
25. [root@master ~]# docker push 192.168.1.100:5000/node-exporter:v1.0.0
26. [root@master ~]# docker load -i kube-rbac-proxy.tar.gz
27. [root@master ~]# docker images            #查看镜像的名称
28. [root@master ~]# docker tag \
29. 旧名称 192.168.1.100:5000/kube-rbac-proxy:v0.4.1
30. [root@master ~]# docker push 192.168.1.100:5000/kube-rbac-proxy:v0.4.1
31. [root@master ~]# docker load -i kube-metrics.tar.gz
32. [root@master ~]# docker images            #查看镜像的名称
33. [root@master ~]# docker tag \
34. 旧名称 192.168.1.100:5000/kube-state-metrics:v1.9.2
35. [root@master ~]# docker push 192.168.1.100:5000/kube-state-metrics:v1.9.2
36. [root@master ~]# docker load -i alertmanager.tar.gz
37. [root@master ~]# docker images            #查看镜像的名称
38. [root@master ~]# docker tag \
39. 旧名称 192.168.1.100:5000/grafana:6.4.3
40. [root@master ~]# docker push 192.168.1.100:5000/grafana:6.4.3
41. [root@master ~]# docker load -i    configmap-reload.tar.gz
42. [root@master ~]# docker images            #查看镜像的名称
43. [root@master ~]# docker tag \
44. 旧名称 192.168.1.100:5000/configmap-reload:v0.3.0
45. [root@master ~]# docker push 192.168.1.100:5000/configmap-reload:v0.3.0
46. [root@master ~]# docker load -i    alertmanager.tar.gz
47. [root@master ~]# docker images            #查看镜像的名称
48. [root@master ~]# docker tag \
49. 旧名称 192.168.1.100:5000/alertmanager:v0.18.0
50. [root@master ~]# docker push 192.168.1.100:5000/alertmanager:v0.18.0

步骤二：修改资源文件部署各种容器服务（在master主机操作操作）

所有资源的素材文件都在云盘第四阶段kubernetes/v1.17.6/Prometheus/目录下。各位同学需要提前下载整个目录到master主机。

1）安装operator

需要修改资源文件，默认资源文件制定的是从官网下载镜像启动容器，我们需要修改为自己的192.168.1.100私有镜像仓库的路径。

1. [root@master prometheus]# vim setup/prometheus-operator-deployment.yaml
2. 27 - --config-reloader-image=192.168.1.100:5000/configmap-reload:v0.3.0
3. 28 - --prometheus-config-reloader=192.168.1.100:5000/prometheus-config-reloader:v0.35.1
4. 29 image: 192.168.1.100:5000/prometheus-operator:v0.35.1
5. #提示：上面这些镜像的链接路径如果不知道，可以使用docker images查看。
6. [root@master prometheus]# kubectl apply -f setup/

2）安装prom-adapter

需要修改资源文件，默认资源文件制定的是从官网下载镜像启动容器，我们需要修改为自己的192.168.1.100私有镜像仓库的路径。

1. [root@master prometheus]# vim prom-adapter/prometheus-adapter-deployment.yaml
2. 28 image: 192.168.1.100:5000/k8s-prometheus-adapter-amd64:v0.5.0
3. #提示：上面这些镜像的链接路径如果不知道，可以使用docker images查看。
4. [root@master prometheus]# kubectl apply -f prom-adapter/

3）安装metrics-state

需要修改资源文件，默认资源文件制定的是从官网下载镜像启动容器，我们需要修改为自己的192.168.1.100私有镜像仓库的路径。

1. [root@master prometheus]# vim metrics-state/kube-state-metrics-deployment.yaml
2. 24     image: 192.168.1.100:5000/kube-rbac-proxy:v0.4.1
3. 41     image: 192.168.1.100:5000/kube-rbac-proxy:v0.4.1
4. 58     image: 192.168.1.100:5000/kube-state-metrics:v1.9.2
5. #提示：上面这些镜像的链接路径如果不知道，可以使用docker images查看。
6. [root@master prometheus]# kubectl apply -f metrics-state/

4）安装node-exporter

需要修改资源文件，默认资源文件制定的是从官网下载镜像启动容器，我们需要修改为自己的192.168.1.100私有镜像仓库的路径。

1. [root@master prometheus]# vim node-exporter/node-exporter-daemonset.yaml
2. 27     image: 192.168.1.100:5000/node-exporter:v1.0.0
3. 57     image: 192.168.1.100:5000/kube-rbac-proxy:v0.4.1
4. #提示：上面这些镜像的链接路径如果不知道，可以使用docker images查看。
5. [root@master prometheus]# kubectl apply -f node-exporter/

5）安装Prometheus server

需要修改资源文件，默认资源文件制定的是从官网下载镜像启动容器，我们需要修改为自己的192.168.1.100私有镜像仓库的路径。

1. [root@master prometheus]# vim prom-server/prometheus-prometheus.yaml
2. 14     baseImage: 192.168.1.100:5000/prometheus
3. 34     version: v2.11.0
4. #提示：上面这些镜像的链接路径如果不知道，可以使用docker images查看。
5. [root@master prometheus]# kubectl apply -f prom-server/

步骤三：对外发布Prometheus容器服务（在master主机操作操作）

相关资源文件共享在云盘第四阶段kubernetes/v1.17.6/prometheus/prom-server/prometheus-service.yaml。

下面使用使用nodePort发布服务将容器的9090端口映射到真机节点的30001端口。

1. [root@master prometheus]# vim prom-server/prometheus-service.yaml
2. ... ...
3. spec:
4. type: NodePort
5. ports:
6. - name: web
7. port: 9090
8. nodePort: 30001
9. targetPort: web
10. ... ...
11. [root@master prometheus]# kubectl apply -f prom-server/prometheus-service.yaml

使用浏览器访问任意节点的30001端口，即可访问到Web网页，效果如图-3所示。

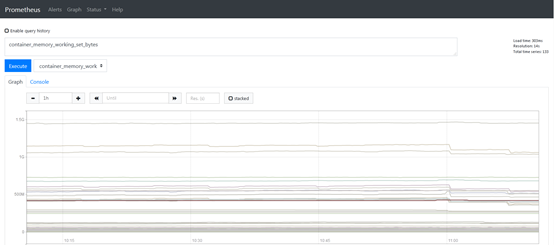


图-3

**4 案例4：安装部署pometheus（二）**

**4.1 问题**

本案例延续前面的案例3，继续部署pometheus，具体要求如下：

1. 安装alertmanager
2. 安装Prometheus grafana

**4.2 步骤**

实现此案例需要按照如下步骤进行。

步骤一：安装alertmanager（在master主机操作）

需要修改资源文件，默认资源文件制定的是从官网下载镜像启动容器，我们需要修改为自己的192.168.1.100私有镜像仓库的路径。

1. [root@master prometheus]# vim alertmanager/alertmanager-alertmanager.yaml
2. 09     baseImage: 192.168.1.100:5000/alertmanager
3. 18     version: v0.18.0
4. #提示：上面这些镜像的链接路径如果不知道，可以使用docker images查看。
5. [root@master prometheus]# kubectl apply -f alertmanager/

步骤二：安装grafana（在master主机操作）

需要修改资源文件，默认资源文件制定的是从官网下载镜像启动容器，我们需要修改为自己的192.168.1.100私有镜像仓库的路径。

1. [root@master prometheus]# vim grafana/grafana-deployment.yaml
2. 19    - image: 192.168.1.100:5000/grafana:6.4.3
3. #提示：上面这些镜像的链接路径如果不知道，可以使用docker images查看。
4. [root@master prometheus]# kubectl apply -f grafana/

步骤三：对外发布grafana服务（在master主机操作）

相关资源文件共享在云盘第四阶段kubernetes/v1.17.6/prometheus/ grafana/grafana-service.yaml。

下面使用使用nodePort发布服务将容器的3000端口映射到真机节点的30002端口。

1. [root@master prometheus]# vim grafana/grafana-service.yaml
2. ... ...
3. spec:
4. type: NodePort
5. ports:
6. - name: http
7. port: 3000
8. nodePort: 30002
9. targetPort: http
10. ... ...
11. [root@master prometheus]# kubectl apply -f grafana/grafana-service.yaml

使用浏览器访问任意节点的30002端口，即可访问到Web网页，效果如图-4所示。



图-4

**5 案例5：配置grafana**

**5.1 问题**

本案例要求配置grafana，具体要求如下。

1. 配置数据源
2. 展示监控页面

**5.2 步骤**

实现此案例需要按照如下步骤进行。

步骤一：配置grafana（在任意主机操作）

1）登录

登录的默认用户和密码：admin/admin

第一次登录需要修改密码，效果如图-5所示。

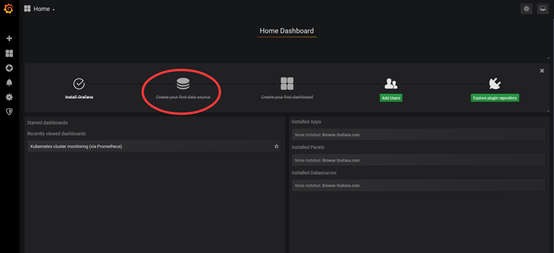


图-5

2）修改数据源地址

我们添加的数据为prometheus

数据源就是 Prometheus service 的地址

可以填写prometheus的Service内部的DNS名称

http://prometheus-k8s.monitoring.svc.cluster.local:9090

如何查看到这个DNS的域名呢？可以执行如下的命令即可。

1. [root@master ~]# kubectl -n monitoring get service
2. NAME         TYPE     CLUSTER-IP     EXTERNAL-IP    PORT(S)
3. grafana NodePort     10.254.169.248     <none>     3000:30002/TCP
4. prometheus-k8s NodePort     10.254.44.72 <none>     9090:30001/TCP
5. ... ...
6. # 找到这里的prometheus-k8s名称后
7. # 在它的后面附加一个固定的monitoring.svc.cluster.local:9090
8. # 连接在一起就是：http://prometheus-k8s.monitoring.svc.cluster.local:9090

点击如图-6和图-7所示的按钮，修改数据源。

添加数据源Prometheus，名字随意，URL需要填写Prometheus内部的DNS名称，

默认端口号 9090。

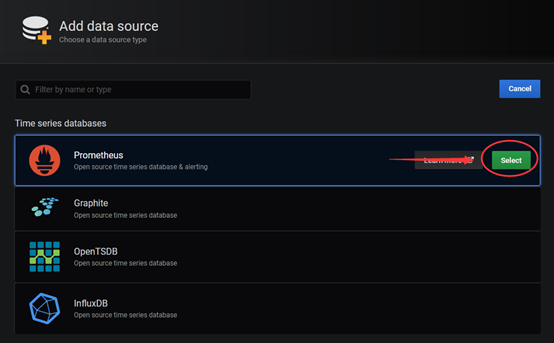


图-6

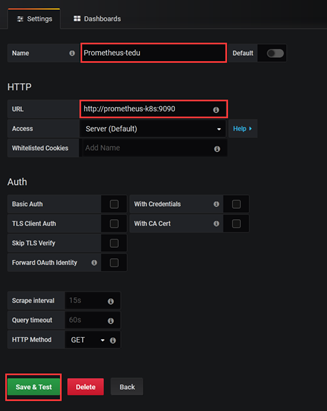


图-7

3）导入模板

导入模板，需要正确配置后点击保存和测试后开始添加仪表盘

点开import，输入模板ID，效果如图-8所示。

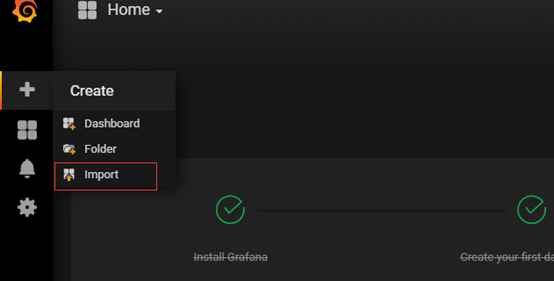


图-8

常用的模块为315，如图-9所示。

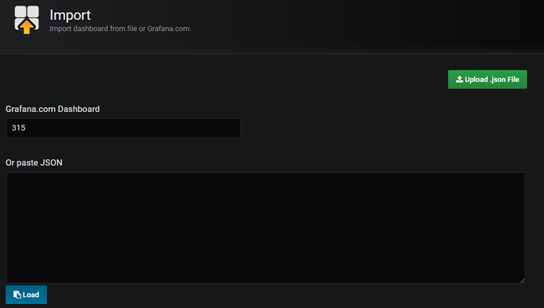


图-9

数据源就是刚刚定义的 Prometheus，如图-10所示。

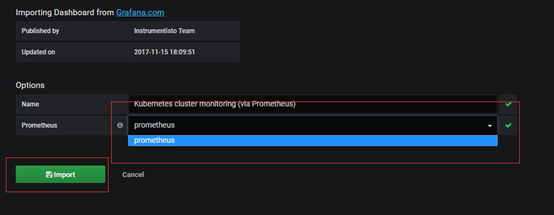


图-10

最后查看监控效果，如图-11和图-12所示。



图-11

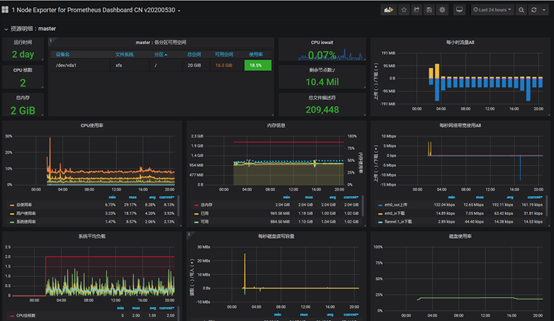


图-12

**6 案例6：HPA集群**

**6.1 问题**

本案例利用监控指标实现HPA集群，具体要求如下。

1. 完成一个弹性集群

**6.2 步骤**

实现此案例需要按照如下步骤进行。

步骤一：部署一个弹性集群（在master主机操作）

1）查看、学习资源文件

资源在云盘共享第四阶段kubernetes/v1.17.6/config/hpa-example.yaml。

1. [root@master ~]# cat hpa-example.yaml #查看、学习资源文件
2. ---
3. apiVersion: apps/v1
4. kind: Deployment
5. metadata:
6. name: myweb
7. spec:
8. selector:
9. matchLabels:
10. app: apache
11. replicas: 1
12. template:
13. metadata:
14. labels:
15. app: apache
16. spec:
17. containers:
18. - name: apache
19. image: 192.168.1.100:5000/myos:httpd
20. ports:
21. - containerPort: 80
22. resources:
23. requests:
24. cpu: 200m
25. restartPolicy: Always
26. ---
27. apiVersion: v1
28. kind: Service
29. metadata:
30. name: web-service
31. spec:
32. ports:
33. - protocol: TCP
34. port: 80
35. targetPort: 80
36. selector:
37. app: apache
38. type: ClusterIP
39. ---
40. apiVersion: autoscaling/v1
41. kind: HorizontalPodAutoscaler
42. metadata:
43. name: myweb
44. spec:
45. minReplicas: 1
46. maxReplicas: 3
47. scaleTargetRef:
48. apiVersion: apps/v1
49. kind: Deployment
50. name: myweb
51. targetCPUUtilizationPercentage: 50
52. [root@master ~]# kubectl apply -f hpa-example.yaml
53. deployment.apps/myweb created
54. service/web-service created

2）创建相关资源

1. [root@master ~]# kubectl apply -f hpa-example.yaml #创建资源
2. deployment.apps/myweb created
3. service/web-service created
4. horizontalpodautoscaler.autoscaling/myweb created
5. [root@master ~]# kubectl get hpa
6. NAME REFERENCE TARGETS MINPODS MAXPODS REPLICAS AGE
7. myweb Deploy/myweb <unknown>/50% 1 3 0 10s
8. [root@master ~]# kubectl get hpa         # 等待几分钟后执行
9. NAME REFERENCE TARGETS MINPODS MAXPODS REPLICAS AGE
10. myweb Deploy/myweb 3%/50% 1 3 0 10s

步骤二：压力测试

想办法对前面创建的容器进行压力测试，消耗容器CPU资源！！！

步骤三：查看动态扩容效果

1. [root@master ~]# kubectl get hpa
2. NAME REFERENCE TARGETS MINPODS MAXPODS REPLICAS AGE
3. myweb Deployment/myweb 287%/50% 1 3 3 16m
4. [root@master ~]# kubectl get pod
5. NAME READY STATUS RESTARTS AGE
6. myweb-7f89fc7b66-kzhj5 1/1 Running 0 16m
7. myweb-7f89fc7b66-nh4kn 1/1 Running 0 21s
8. myweb-7f89fc7b66-q2jnf 1/1 Running 0 21s
9. #当容器CPU占用过高时，集群可以自动扩容
10. -----------------------------------------------------------------------------
11. [root@master ~]# kubectl get hpa
12. NAME REFERENCE TARGETS MINPODS MAXPODS REPLICAS AGE
13. myweb Deployment/myweb 1%/50% 1 3 3 20m
14. [root@master ~]# kubectl get pod
15. NAME READY STATUS RESTARTS AGE
16. myweb-7f89fc7b66-kzhj5 1/1 Running 0 22m
17. # 当容器CPU占用率恢复正常时，容器可以自动缩减