Report

作业内容：

> 提交内容：
> 0、参考资料的学习笔记（阐述几个问题：什么是镜像？什么是容器？什么是NameSpace？什么是Cgroup？什么是Service？Service有什么用？Prometheus怎么工作的？）
> 1、自己编写的云应用代码github仓库
> 2、详细实验过程截图（证明自己的Service配置好了，证明自己的prometheus配置好了，能够拉到metrics了。）
> 3、README中简单阐述自己的代码逻辑
> 4、实验感想（可选）
>
> 上面的实验的评分逻辑：
> 0、问题回答是否正确且完整
> 1、代码注释是否清晰
> 2、metrics的设计是否合理且丰富
> 3、截图是否直观
> 4、README描述是否完整）
>
> 加分项（未完成加分项则满分90，加分项一个5分）：
> 0、截图证明配置了Grafana监控视图，并阐述如何配置
> 1、阐述Service API对象　对宿主机iptables的操作，截图查看宿主机iptables情况。

0.学习笔记

1.容器：docker容器就是用来运行镜像的小型的虚拟机，或者说是一个轻量级的沙箱，Docker利用容器来运行和隔离应用。。

特性	虚拟机	容器
隔离级别	操作系统级	进程级
隔离策略	Hypervisor	CGroups
系统资源	5~15%	0~5%
启动时间	分钟级	秒级
镜像存储	GB-TB	KB-MB
集群规模	上百	上万
高可用策略	备份、容灾、迁移	弹性、负载、动态

2.镜像：类似于虚拟机镜像，镜像是只读的，是创建docker容器的基础。运行中的一个镜像就构成了一格容器。

3.NameSpace:Linux Namespace是Linux内核提供的一种资源隔离方案。处于不同 namespace 的进程拥有独立的全局系统资源，改变一个 namespace 中的系统资源只会影响当前 namespace 里的进程，对其他 namespace 中的进程没有影响。

名称	宏定义	隔离的资源
IPC	CLONE_NEWIPC	System V IPC(信号量、消息队列和共享内存) 和 POSIX message queues
Network	CLONE_NEWNET	Network devices, stacks, ports, etc(网络设备、网络栈、端口等).
Mount	CLONE_NEWNS	Mount points(文件系统挂载点)
PID	CLONE_NEWPID	Process IDs(进程编号)
User	CLONE_NEWUSER	User and group IDs(用户和用户组)
UTS	CLONE_NEWUTS	Hostname and NIS domain name(主机名与 NIS 域名)
Cgroup	CLONE_NEWCGROUP	Cgroup root directory(cgroup 的根目录)

4.Cgroups:Control Groups 是Linux内核的一个功能，用来限制、控制与分离一个进程组的资源。实现cgroups的主要目的是为不同用户层面的资源管理，提供一个统一化的接口。从单个进程的资源控制到操作系统层面的虚拟化。Cgroups提供了以下四大功能:

- 资源限制（Resource Limitation）：cgroups可以对进程组使用的资源总额进行限制。如设定应用运行时使用内存的上限，一旦超过这个配额就发出OOM（Out of Memory）。
- 优先级分配（Prioritization）：通过分配的CPU时间片数量及硬盘IO带宽大小，实际上就相当于控制了进程运行的优先级。
- 资源统计（Accounting）： cgroups可以统计系统的资源使用量，如CPU使用时长、内存用量等等，这个功能非常适用于计费。
- 进程控制（Control）：cgroups可以对进程组执行挂起、恢复等操作。

5.Services: Service是将运行在一组 Pods 上的应用程序公开为网络服务的抽象方法。Kubernetes `Service` 定义了这样一种抽象：逻辑上的一组 `Pod`，一种可以访问它们的策略 —— 通常称为微服务。 如果需要对应用程序进行访问，实际上是说需要对在某个node，某个pod上的应用程序进行访问，但由于在应用程序的生命周期内，pod可以是不同的，因此需要一个机制对其进行追踪，这就是service的用处。

6.Prometheus:Prometheus 项目工作的核心，是使用 Pull （抓取）的方式去搜集被监控对象的 Metrics 数据（监控指标数据），然后，再把这些数据保存在一个 TSDB （时间序列数据库，比如 OpenTSDB、InfluxDB 等）当中，以便后续可以按照时间进行检索。 Prometheus 剩下的组件就是用来配合这套机制的运行。Metrics 数据指3种数据：

- 宿主机的监控数据
- 来自于 Kubernetes 的 API Server、kubelet 等组件的 /metrics API
- Kubernetes 相关的监控数据

使用pull的方式由于类似一种轮询的机制，相比push方式实际上能降低阻塞的几率，提高可用性。

Prometheus提供了一个集成的规范化显示数据的机制，不必直接查看metrics相关文件。

reference

https://zhuanlan.zhihu.com/p/53260098

https://www.kubernetes.org.cn/k8s

https://blog.csdn.net/ra681t58cjxsgckj31/article/details/104707642

https://en.wikipedia.org/wiki/Cgroups

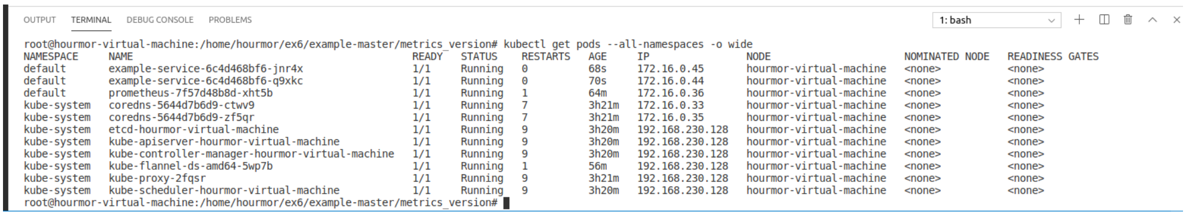
https://docs.docker.com/engine/swarm/how-swarm-mode-works/services/

https://kubernetes.io/zh/docs/concepts/services-networking/service/

https://time.geekbang.org/column/article/72281

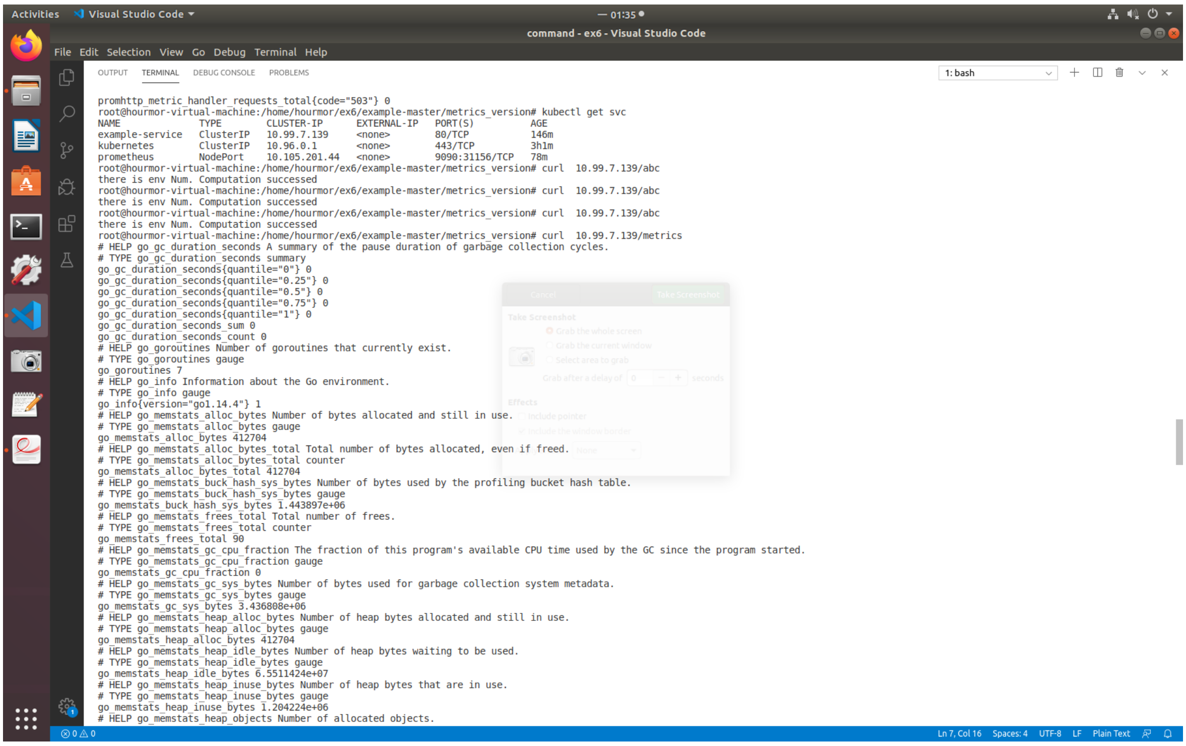
2.实验过程截图

所有pod正常运行



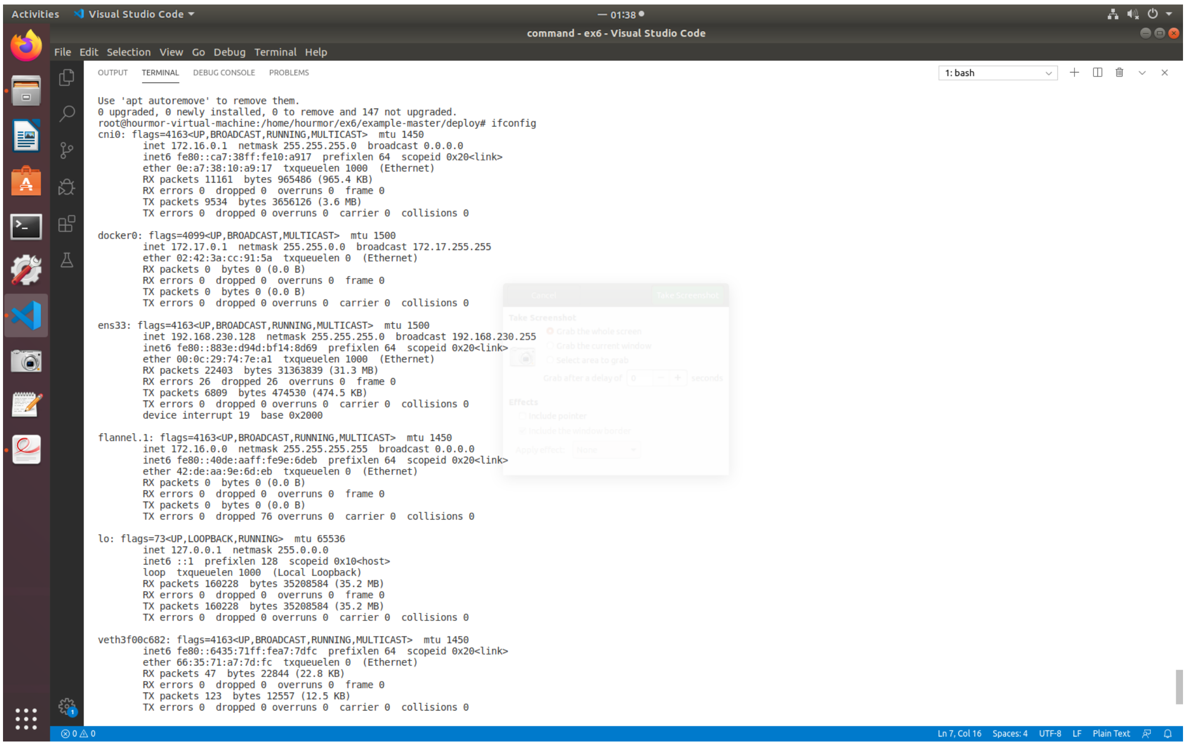
```
root@hourmor-virtual-machine:/home/hourmor/ex6/example-master/metrics_version# kubectl get pods --all-namespaces -o wide
NAMESPACE     NAME                                                 READY   STATUS    RESTARTS   AGE    IP                NODE                       NOMINATED NODE   READINESS GATES
default       example-service-6c4d468bf6-jnr4x                     1/1     Running   0          68s    172.16.0.45       hourmor-virtual-machine    <none>           <none>
default       example-service-6c4d468bf6-q9xkc                     1/1     Running   0          70s    172.16.0.44       hourmor-virtual-machine    <none>           <none>
default       prometheus-7f57d48b8d-xht5b                          1/1     Running   1          64m    172.16.0.36       hourmor-virtual-machine    <none>           <none>
kube-system   coredns-5644d7b6d9-ctwv9                             1/1     Running   7          3h21m  172.16.0.33       hourmor-virtual-machine    <none>           <none>
kube-system   coredns-5644d7b6d9-zf5qr                             1/1     Running   7          3h21m  172.16.0.35       hourmor-virtual-machine    <none>           <none>
kube-system   etcd-hourmor-virtual-machine                         1/1     Running   9          3h20m  192.168.230.128   hourmor-virtual-machine    <none>           <none>
kube-system   kube-apiserver-hourmor-virtual-machine               1/1     Running   9          3h20m  192.168.230.128   hourmor-virtual-machine    <none>           <none>
kube-system   kube-controller-manager-hourmor-virtual-machine      1/1     Running   9          3h20m  192.168.230.128   hourmor-virtual-machine    <none>           <none>
kube-system   kube-flannel-ds-amd64-5wp7b                          1/1     Running   1          56m    192.168.230.128   hourmor-virtual-machine    <none>           <none>
kube-system   kube-proxy-2fqsr                                     1/1     Running   9          3h21m  192.168.230.128   hourmor-virtual-machine    <none>           <none>
kube-system   kube-scheduler-hourmor-virtual-machine               1/1     Running   9          3h20m  192.168.230.128   hourmor-virtual-machine    <none>           <none>
root@hourmor-virtual-machine:/home/hourmor/ex6/example-master/metrics_version#
```

example-service 正常执行&&获取metrics



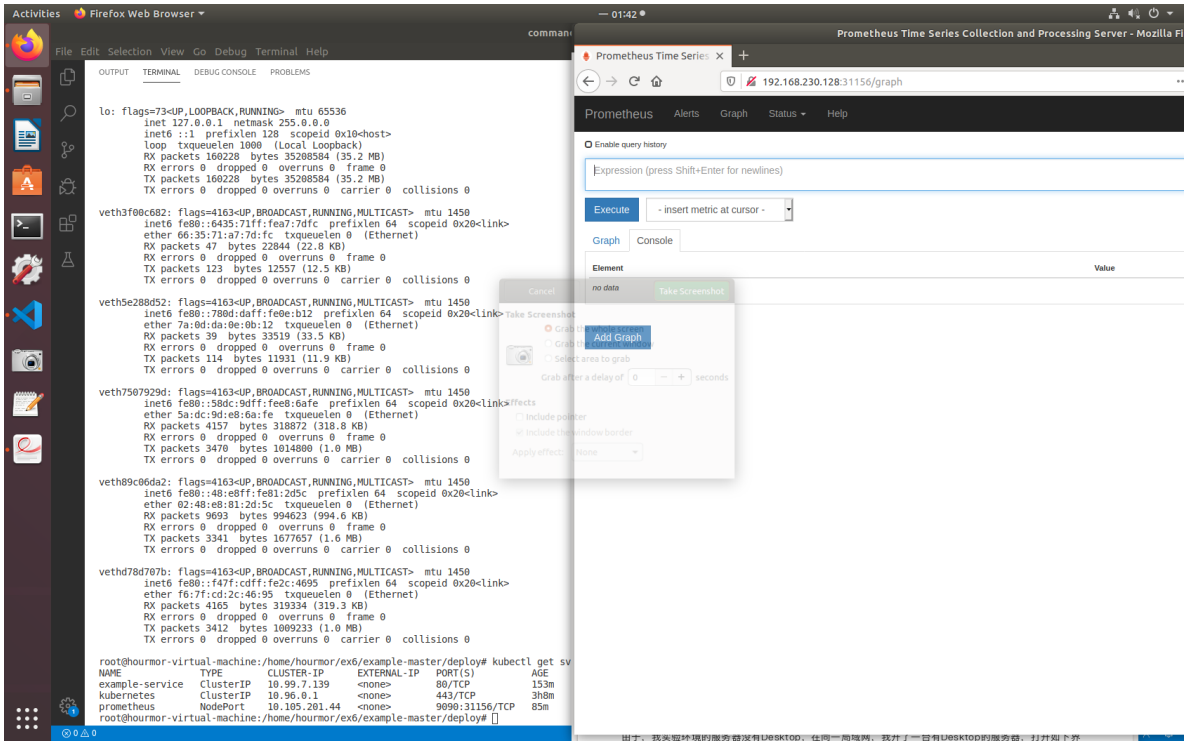
```
promhttp_metric_handler_requests_total{code="503"} 0
root@hourmor-virtual-machine:/home/hourmor/ex6/example-master/metrics_version# kubectl get svc
NAME              TYPE        CLUSTER-IP      EXTERNAL-IP   PORT(S)          AGE
example-service   ClusterIP   10.99.7.139     <none>        80/TCP           146m
kubernetes        ClusterIP   10.96.0.1       <none>        443/TCP          3h1m
prometheus        NodePort    10.105.201.44   <none>        9090:31156/TCP   78m
root@hourmor-virtual-machine:/home/hourmor/ex6/example-master/metrics_version# curl  10.99.7.139/abc
there is env Num. Computation successed
root@hourmor-virtual-machine:/home/hourmor/ex6/example-master/metrics_version# curl  10.99.7.139/abc
there is env Num. Computation successed
root@hourmor-virtual-machine:/home/hourmor/ex6/example-master/metrics_version# curl  10.99.7.139/abc
there is env Num. Computation successed
root@hourmor-virtual-machine:/home/hourmor/ex6/example-master/metrics_version# curl  10.99.7.139/metrics
# HELP go_gc_duration_seconds A summary of the pause duration of garbage collection cycles.
# TYPE go_gc_duration_seconds summary
go_gc_duration_seconds{quantile="0"} 0
go_gc_duration_seconds{quantile="0.25"} 0
go_gc_duration_seconds{quantile="0.5"} 0
go_gc_duration_seconds{quantile="0.75"} 0
go_gc_duration_seconds{quantile="1"} 0
go_gc_duration_seconds_sum 0
go_gc_duration_seconds_count 0
# HELP go_goroutines Number of goroutines that currently exist.
# TYPE go_goroutines gauge
go_goroutines 7
# HELP go_info Information about the Go environment.
# TYPE go_info gauge
go_info{version="go1.14.4"} 1
# HELP go_memstats_alloc_bytes Number of bytes allocated and still in use.
# TYPE go_memstats_alloc_bytes gauge
go_memstats_alloc_bytes 412704
# HELP go_memstats_alloc_bytes_total Total number of bytes allocated, even if freed.
# TYPE go_memstats_alloc_bytes_total counter
go_memstats_alloc_bytes_total 412704
# HELP go_memstats_buck_hash_sys_bytes Number of bytes used by the profiling bucket hash table.
# TYPE go_memstats_buck_hash_sys_bytes gauge
go_memstats_buck_hash_sys_bytes 1.443897e+06
# HELP go_memstats_frees_total Total number of frees.
# TYPE go_memstats_frees_total counter
go_memstats_frees_total 90
# HELP go_memstats_gc_cpu_fraction The fraction of this program's available CPU time used by the GC since the program started.
# TYPE go_memstats_gc_cpu_fraction gauge
go_memstats_gc_cpu_fraction 0
# HELP go_memstats_gc_sys_bytes Number of bytes used for garbage collection system metadata.
# TYPE go_memstats_gc_sys_bytes gauge
go_memstats_gc_sys_bytes 3.436808e+06
# HELP go_memstats_heap_alloc_bytes Number of heap bytes allocated and still in use.
# TYPE go_memstats_heap_alloc_bytes gauge
go_memstats_heap_alloc_bytes 412704
# HELP go_memstats_heap_idle_bytes Number of heap bytes waiting to be used.
# TYPE go_memstats_heap_idle_bytes gauge
go_memstats_heap_idle_bytes 6.5511424e+07
# HELP go_memstats_heap_inuse_bytes Number of heap bytes that are in use.
# TYPE go_memstats_heap_inuse_bytes gauge
go_memstats_heap_inuse_bytes 1.204224e+06
# HELP go_memstats_heap_objects Number of allocated objects.
```

获取网络配置信息

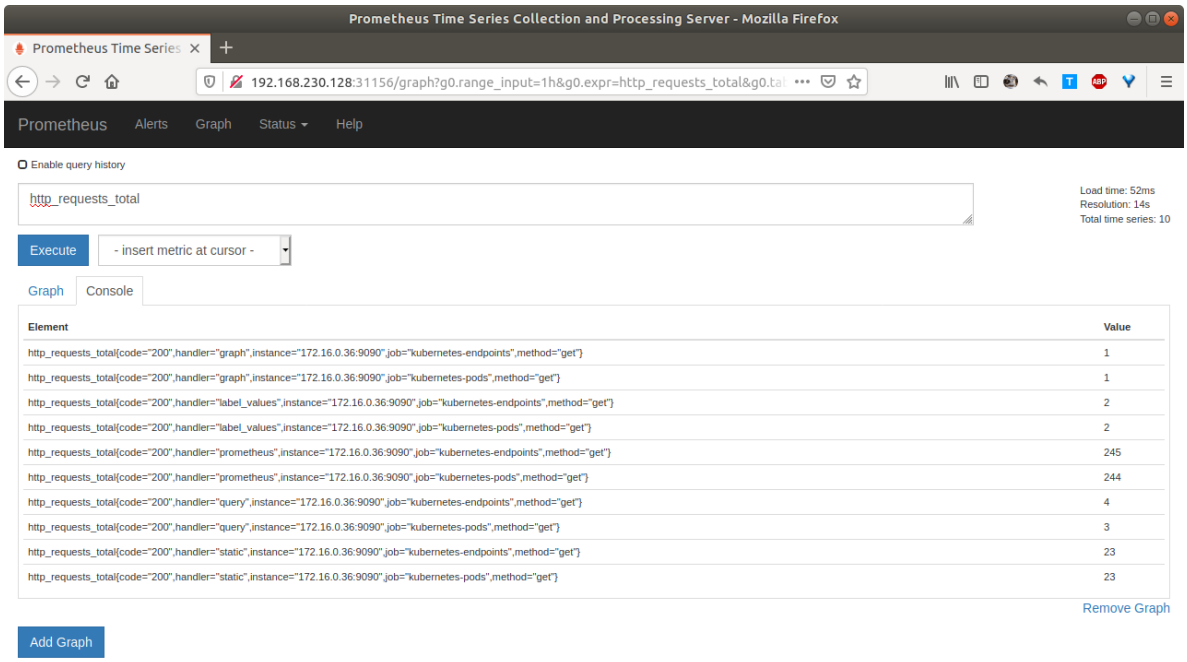


```
Use 'apt autoremove' to remove them.
0 upgraded, 0 newly installed, 0 to remove and 147 not upgraded.
root@hourmor-virtual-machine:/home/hourmor/ex6/example-master/deploy# ifconfig
cni0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1450
        inet 172.16.0.1  netmask 255.255.255.0  broadcast 0.0.0.0
        inet6 fe80::ca7:38ff:fe10:a917  prefixlen 64  scopeid 0x20<link>
        ether 0e:a7:38:10:a9:17  txqueuelen 1000  (Ethernet)
        RX packets 11161  bytes 965486 (965.4 KB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 9534  bytes 3656126 (3.6 MB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

docker0: flags=4099<UP,BROADCAST,MULTICAST>  mtu 1500
        inet 172.17.0.1  netmask 255.255.0.0  broadcast 172.17.255.255
        ether 02:42:3a:cc:91:5a  txqueuelen 0  (Ethernet)
        RX packets 0  bytes 0 (0.0 B)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 0  bytes 0 (0.0 B)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

ens33: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.230.128  netmask 255.255.255.0  broadcast 192.168.230.255
        inet6 fe80::883e:d94d:bf14:8d69  prefixlen 64  scopeid 0x20<link>
        ether 00:0c:29:74:7e:a1  txqueuelen 1000  (Ethernet)
        RX packets 22403  bytes 31363839 (31.3 MB)
        RX errors 26  dropped 26  overruns 0  frame 0
        TX packets 6809  bytes 474530 (474.5 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
        device interrupt 19  base 0x2000

flannel.1: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1450
        inet 172.16.0.0  netmask 255.255.255.255  broadcast 0.0.0.0
        inet6 fe80::40de:aaff:fe9e:6deb  prefixlen 64  scopeid 0x20<link>
        ether 42:de:aa:9e:6d:eb  txqueuelen 0  (Ethernet)
        RX packets 0  bytes 0 (0.0 B)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 0  bytes 0 (0.0 B)
        TX errors 0  dropped 76 overruns 0  carrier 0  collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
        inet 127.0.0.1  netmask 255.0.0.0
        inet6 ::1  prefixlen 128  scopeid 0x10<host>
        loop  txqueuelen 1000  (Local Loopback)
        RX packets 160228  bytes 35208584 (35.2 MB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 160228  bytes 35208584 (35.2 MB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

veth3f00c682: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1450
        inet6 fe80::6435:71ff:fea7:7dfc  prefixlen 64  scopeid 0x20<link>
        ether 66:35:71:a7:7d:fc  txqueuelen 0  (Ethernet)
        RX packets 47  bytes 22844 (22.8 KB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 123  bytes 12557 (12.5 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

获知Prometheus开放的端口


Activities
Firefox Web Browser
Prometheus Time Series Collection and Processing Server - Mozilla Fi
Prometheus Time Series
192.168.230.128:31156/graph
Prometheus
Alerts
Graph
Status
Help
Enable query history
Expression (press Shift+Enter for newlines)
Execute
- insert metric at cursor -
Graph
Console
Element
Value
no data
Add Graph
root@hourmor-virtual-machine:/home/hourmor/ex6/example-master/deploy# kubectl get sv
NAME TYPE CLUSTER-IP EXTERNAL-IP PORT(S) AGE
example-service ClusterIP 10.99.7.139 <none> 80/TCP 153m
kubernetes ClusterIP 10.96.0.1 <none> 443/TCP 3h8m
prometheus NodePort 10.105.201.44 <none> 9090:31156/TCP 85m

打开网页进行访问


Prometheus Time Series Collection and Processing Server - Mozilla Firefox
Prometheus Time Series
192.168.230.128:31156/graph?g0.range_input=1h&g0.expr=http_requests_total&g0.ta
Prometheus
Alerts
Graph
Status
Help
Enable query history
http_requests_total
Load time: 52ms
Resolution: 14s
Total time series: 10
Execute
- insert metric at cursor -
Graph
Console
Element
Value
http_requests_total{code="200",handler="graph",instance="172.16.0.36:9090",job="kubernetes-endpoints",method="get"} 1
http_requests_total{code="200",handler="graph",instance="172.16.0.36:9090",job="kubernetes-pods",method="get"} 1
http_requests_total{code="200",handler="label_values",instance="172.16.0.36:9090",job="kubernetes-endpoints",method="get"} 2
http_requests_total{code="200",handler="label_values",instance="172.16.0.36:9090",job="kubernetes-pods",method="get"} 2
http_requests_total{code="200",handler="prometheus",instance="172.16.0.36:9090",job="kubernetes-endpoints",method="get"} 245
http_requests_total{code="200",handler="prometheus",instance="172.16.0.36:9090",job="kubernetes-pods",method="get"} 244
http_requests_total{code="200",handler="query",instance="172.16.0.36:9090",job="kubernetes-endpoints",method="get"} 4
http_requests_total{code="200",handler="query",instance="172.16.0.36:9090",job="kubernetes-pods",method="get"} 3
http_requests_total{code="200",handler="static",instance="172.16.0.36:9090",job="kubernetes-endpoints",method="get"} 23
http_requests_total{code="200",handler="static",instance="172.16.0.36:9090",job="kubernetes-pods",method="get"} 23
Remove Graph
Add Graph