HttpServer部署Docker

原始需求

- 构建本地镜像
- 编写 Dockerfile 将练习 2.2 编写的 httpserver 容器化
- 将镜像推送至 docker 官方镜像仓库
- 通过 docker 命令本地启动 httpserver
- 通过 nsenter 进入容器查看 IP 配置

作业需编写并提交 Dockerfile 及源代码。

HttpServer代码

```
package main

import (
    "fmt"
    "os"
    "strconv"

    //"github.com/thinkeridea/go-extend/exnet"
    "io"
    "log"
    "net/http"
    "strings"
)

/*
编写一个 HTTP 服务器，大家视个人不同情况决定完成到哪个环节，但尽量把 1 都做完：

    1，接收客户端 request，并将 request 中带的 header 写入 response header
    2，读取当前系统的环境变量中的 VERSION 配置，并写入 response header
    3，Server 端记录访问日志包括客户端 IP，HTTP 返回码，输出到 server 端的标准输出
    4，当访问 localhost/healthz 时，应返回 200
*/

// Main方法入口
func main() {
    println("环境正常")

    // 功能1
    http.HandleFunc("/requestAndResponse", requestAndResponse)

    // 功能2
    http.HandleFunc("/getVersion", getVersion)

    // 功能3
    http.HandleFunc("/ipAndStatus", ipAndStatus) //注册接口句柄

    // 功能4
    http.HandleFunc("/healthz", healthz) //注册接口句柄
```

```
    err := http.ListenAndServe(":81", nil) //监听空句柄，80端口被占用，使用81端口
    if nil != err {
        log.Fatal(err) //显示错误日志
    }
}

// 功能1，接收请求及响应
func requestAndResponse(response http.ResponseWriter, request *http.Request) {
    println("调用requestAndResponse接口")
    headers := request.Header //header是Map类型的数据
    println("传入的hander：")
    for header := range headers { //value是[]string
        //println("header的key：" + header)
        values := headers[header]
        for index, _ := range values {
            values[index] = strings.TrimSpace(values[index])
            //println("index=" + strconv.Itoa(index))
            //println("header的value：" + values[index])

        }
        //valueString := strings.Join(values, "")
        //println("header的value：" + valueString)
        println(header + "=" + strings.Join(values, ","))        //打印request的
header的k=v
        response.Header().Set(header, strings.Join(values, ",")) // 遍历写入
response的Header
        //println()

    }
    fmt.Fprintln(response, "Header全部数据:", headers)
    io.WriteString(response, "succeed")

}

// 功能2，获取环境变量的version
func getVersion(response http.ResponseWriter, request *http.Request) {
    println("调用getVersion接口")
    envStr := os.Getenv("VERSION")
    //envStr := os.Getenv("HADOOP_HOME")
    //println("系统环境变量：" + envStr) //可以看到 C:\soft\hadoop-3.3.1   Win10需要
重启电脑才能生效

    response.Header().Set("VERSION", envStr)
    io.WriteString(response, "succeed")

}

// 功能3，输出IP与返回码
func ipAndStatus(response http.ResponseWriter, request *http.Request) {
    println("调用ipAndStatus接口")

    form := request.RemoteAddr
    println("Client->ip:port=" + form) //虚拟机是桥接模式。使用postman返回的全部是
127.0.0.1  用手机打开网站192.168.1.139:81/ipAndStatus可以看到新IP
    ipStr := strings.Split(form, ":")
    println("Client->ip=" + ipStr[0]) //打印ip

    // 获取http响应码
```

```
    //response.WriteHeader(301) //手动设置响应码，默认200

    //response.WriteHeader(http.StatusOK)//由于默认是调用这个，∴返回码都是这个200【server.go有源码】
    println("Client->response code=" + strconv.Itoa(http.StatusOK))

    //println("response code->: " + code)

    io.WriteString(response, "succeed")
}

// 功能4，连通性测试接口
func healthz(response http.ResponseWriter, request *http.Request) {
    println("调用healthz接口")
    response.WriteHeader(200) //设置返回码200
    //response.WriteHeader(http.StatusOK)//默认会调用这个方法，默认就是200【server.go有源码】
    io.WriteString(response, "succeed")
}
```

Ubuntu安装Go

```
Last login: Tue May  3 20:22:13 2022 from 192.168.88.1
zhiyong@zhiyong-vm-dev:~$ ifconfig
ens33: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.88.50  netmask 255.255.255.0  broadcast 192.168.88.255
        inet6 fe80::e89d:a89:8f8f:9304  prefixlen 64  scopeid 0x20<link>
        ether 00:0c:29:56:16:b5  txqueuelen 1000  (以太网)
        RX packets 892  bytes 721182 (721.1 KB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 694  bytes 97876 (97.8 KB)
        TX errors 0  dropped 0 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  (本地环回)
        RX packets 218  bytes 20407 (20.4 KB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 218  bytes 20407 (20.4 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

zhiyong@zhiyong-vm-dev:~$ go
Command 'go' not found, but can be installed with:
sudo snap install go         # version 1.17.6, or
sudo apt  install golang-go  # version 2:1.16~0ubuntu1
sudo apt  install gccgo-go   # version 2:1.16~0ubuntu1
See 'snap info go' for additional versions.
zhiyong@zhiyong-vm-dev:~$ sudo apt  install golang-go
升级了 0 个软件包，新安装了 33 个软件包，要卸载 0 个软件包，有 0 个软件包未被升级。
需要下载 212 MB 的归档。
解压缩后会消耗 844 MB 的额外空间。
```

```
您希望继续执行吗？ [Y/n] y
zhiyong@zhiyong-vm-dev:~$ go version
go version go1.17 linux/amd64
```

Ubuntu编译Go代码

先检查81端口占用情况：

win中使用：

```
C:\Users\zhiyong>netstat -ano | findstr "81"
C:\Users\zhiyong>taskkill -f -pid 这里写PID号
```

杀进程。Goland做Debug就很方便。

Ubuntu使用：

```
zhiyong@zhiyong-vm-dev:~$ mkdir -p dockerstudy/httpserver/go
zhiyong@zhiyong-vm-dev:~$ cd /home/zhiyong/dockerstudy/httpserver/go/
zhiyong@zhiyong-vm-dev:~/dockerstudy/httpserver/go$ ll
总用量 8
drwxrwxr-x 2 zhiyong zhiyong 4096  5月  5 00:02 ./
drwxrwxr-x 3 zhiyong zhiyong 4096  5月  5 00:02 ../
zhiyong@zhiyong-vm-dev:~/dockerstudy/httpserver/go$ sudo apt install vim
zhiyong@zhiyong-vm-dev:~/dockerstudy/httpserver/go$ vim main.go
# 将Go代码写进去

zhiyong@zhiyong-vm-dev:~/dockerstudy/httpserver/go$ ll
总用量 12
drwxrwxr-x 2 zhiyong zhiyong 4096  5月  5 00:07 ./
drwxrwxr-x 3 zhiyong zhiyong 4096  5月  5 00:02 ../
-rw-rw-r-- 1 zhiyong zhiyong 3559  5月  5 00:07 main.go
zhiyong@zhiyong-vm-dev:~/dockerstudy/httpserver/go$ netstat -atunlp | grep 81
（并非所有进程都能被检测到，所有非本用户的进程信息将不会显示，如果想看到所有信息，则必须切换到
root 用户）
udp        0      0 0.0.0.0:52981           0.0.0.0:*
-
zhiyong@zhiyong-vm-dev:~/dockerstudy/httpserver/go$ sudo go build -o
gohttpserver main.go
zhiyong@zhiyong-vm-dev:~/dockerstudy/httpserver/go$ ll
总用量 6000
drwxrwxr-x 2 zhiyong zhiyong    4096  5月  5 00:09 ./
drwxrwxr-x 3 zhiyong zhiyong    4096  5月  5 00:02 ../
-rwxr-xr-x 1 root    root    6128735  5月  5 00:09 gohttpserver*
-rw-rw-r-- 1 zhiyong zhiyong    3559  5月  5 00:07 main.go
zhiyong@zhiyong-vm-dev:~/dockerstudy/httpserver/go$ ./gohttpserver
环境正常
2022/05/05 00:09:19 listen tcp :81: bind: permission denied
zhiyong@zhiyong-vm-dev:~/dockerstudy/httpserver/go$ sudo ./gohttpserver
环境正常
```

另一个黑窗口：

```
zhiyong@zhiyong-vm-dev:~$ netstat -atunlp | grep 81
（并非所有进程都能被检测到，所有非本用户的进程信息将不会显示，如果想看到所有信息，则必须切换到
root 用户）
tcp6       0      0 :::81                   :::*                    LISTEN
 -
udp        0      0 0.0.0.0:52981           0.0.0.0:*
-
zhiyong@zhiyong-vm-dev:~$ sudo netstat -atunlp | grep 81
[sudo] zhiyong 的密码：
tcp6       0      0 :::81                   :::*                    LISTEN
 12367/./gohttpserve
udp        0      0 0.0.0.0:52981           0.0.0.0:*
834/avahi-daemon: r
```

可以看到进程启动。

浏览器访问：

```
http://192.168.88.50:81/healthz
```

显示文字succeed，说明编译成功，且该编译后的可执行程序在Ubuntu可以正常使用。

结束程序：

```
zhiyong@zhiyong-vm-dev:~/dockerstudy/httpserver/go$ sudo ./gohttpserver
环境正常
调用healthz接口
^C
zhiyong@zhiyong-vm-dev:~$ sudo netstat -atunlp | grep 81
tcp6       0      0 192.168.88.50:81        192.168.88.1:49547      TIME_WAIT
-
udp        0      0 0.0.0.0:52981           0.0.0.0:*
834/avahi-daemon: r
```

Ubuntu安装Docker

```
zhiyong@zhiyong-vm-dev:~$ docker
Command 'docker' not found, but can be installed with:
sudo snap install docker     # version 20.10.11, or
sudo apt  install docker.io  # version 20.10.12-0ubuntu2~21.10.1
See 'snap info docker' for additional versions.
zhiyong@zhiyong-vm-dev:~$ sudo apt  install docker.io
zhiyong@zhiyong-vm-dev:~$ sudo apt  install docker.io
[sudo] zhiyong 的密码：
正在读取软件包列表... 完成
正在分析软件包的依赖关系树... 完成
正在读取状态信息... 完成
下列软件包是自动安装的并且现在不需要了：
```

```
  libfwupdplugin1 linux-headers-5.13.0-30 linux-headers-5.13.0-30-generic linux-
image-5.13.0-30-generic linux-modules-5.13.0-30-generic
  linux-modules-extra-5.13.0-30-generic
使用'sudo apt autoremove'来卸载它(它们)。
将会同时安装下列软件:
  bridge-utils containerd git git-man liberror-perl pigz runc ubuntu-fan
建议安装:
  ifupdown aufs-tools btrfs-progs cgroupfs-mount | cgroup-lite debootstrap
docker-doc rinse zfs-fuse | zfsutils git-daemon-run | git-daemon-sysvinit git-
doc
  git-email git-gui gitk gitweb git-cvs git-mediawiki git-svn
下列【新】软件包将被安装:
  bridge-utils containerd docker.io git git-man liberror-perl pigz runc ubuntu-
fan
升级了 0 个软件包，新安装了 9 个软件包，要卸载 0 个软件包，有 0 个软件包未被升级。
需要下载 70.1 MB 的归档。
解压缩后会消耗 315 MB 的额外空间。
您希望继续执行吗？ [Y/n] y
```

注册Docker Hub

```
https://hub.docker.com/
```

构建镜像

拉取Ubuntu镜像:

```
zhiyong@zhiyong-vm-dev:~$ docker pull ubuntu
Using default tag: latest
Got permission denied while trying to connect to the Docker daemon socket at
unix:///var/run/docker.sock: Post
"http://%2Fvar%2Frun%2Fdocker.sock/v1.24/images/create?
fromImage=ubuntu&tag=latest": dial unix /var/run/docker.sock: connect:
permission denied
zhiyong@zhiyong-vm-dev:~$ sudo docker pull ubuntu
[sudo] zhiyong 的密码:
Using default tag: latest
latest: Pulling from library/ubuntu
125a6e411906: Pull complete
Digest: sha256:26c68657ccce2cb0a31b330cb0be2b5e108d467f641c62e13ab40cbec258c68d
Status: Downloaded newer image for ubuntu:latest
docker.io/library/ubuntu:latest
zhiyong@zhiyong-vm-dev:~$ docker images
Got permission denied while trying to connect to the Docker daemon socket at
unix:///var/run/docker.sock: Get
"http://%2Fvar%2Frun%2Fdocker.sock/v1.24/images/json": dial unix
/var/run/docker.sock: connect: permission denied
zhiyong@zhiyong-vm-dev:~$ sudo docker images
REPOSITORY   TAG       IMAGE ID       CREATED      SIZE
ubuntu       latest    d2e4e1f51132   4 days ago   77.8MB
```

启动Ubuntu镜像：

```
zhiyong@zhiyong-vm-dev:~$ sudo docker run -ti ubuntu /bin/bash
root@c515a662d46c:/# pwd
/
root@c515a662d46c:/# vim test.txt
bash: vim: command not found    #需要安装vim
root@c515a662d46c:/# apt-get update
root@c515a662d46c:/# apt-get -y install vim
root@c515a662d46c:/# exit
exit
zhiyong@zhiyong-vm-dev:~$ sudo docker ps -a
CONTAINER ID   IMAGE     COMMAND       CREATED          STATUS
   PORTS     NAMES
c515a662d46c   ubuntu    "/bin/bash"   11 minutes ago   Exited (0) 39 seconds
ago             goofy_heisenberg
zhiyong@zhiyong-vm-dev:~$ sudo docker images
REPOSITORY   TAG       IMAGE ID       CREATED      SIZE
ubuntu       latest    d2e4e1f51132   4 days ago   77.8MB
zhiyong@zhiyong-vm-dev:~$ sudo docker commit -m="add vim" -a="zhiyong"
c515a662d46c ubuntuwithvm
sha256:428e030662bc465ce126cf3a9e726b1be34219583c1200865ad11e89a16b961a
zhiyong@zhiyong-vm-dev:~$ sudo docker images
REPOSITORY     TAG       IMAGE ID       CREATED          SIZE
ubuntuwithvm   latest    428e030662bc   14 seconds ago   170MB
ubuntu         latest    d2e4e1f51132   4 days ago       77.8MB
```

DockerFile内容：

```
FROM ubuntu
RUN echo '这是一个基于ubuntu且应用为httpserver的镜像' && mkdir -p
/dockerstudy/httpserver
COPY go/gohttpserver /dockerstudy/httpserver/
#此处source路径必须是相对路径，使用绝对路径会报错COPY failed: file not found in build
context or excluded by .dockerignore: stat
/home/zhiyong/dockerstudy/httpserver/go/gohttpserver: file does not exist

ENTRYPOINT ["/dockerstudy/httpserver/gohttpserver"]
```

构建命令：

```
zhiyong@zhiyong-vm-dev:~/dockerstudy/httpserver$ pwd
/home/zhiyong/dockerstudy/httpserver
zhiyong@zhiyong-vm-dev:~/dockerstudy/httpserver$ vim Dockerfile
zhiyong@zhiyong-vm-dev:~/dockerstudy/httpserver$ sudo docker build -t
gohttpserver:v1 .
Sending build context to Docker daemon  6.136MB
Step 1/4 : FROM ubuntu
 ---> d2e4e1f51132
Step 2/4 : RUN echo '这是一个基于ubuntu且应用为httpserver的镜像' && mkdir -p
/dockerstudy/httpserver
 ---> Using cache
 ---> 78301d6519e6
```

```
Step 3/4 : COPY go/gohttpserver /dockerstudy/httpserver
 ---> a71b9bf69a19
Step 4/4 : ENTRYPOINT ["/dockerstudy/httpserver/gohttpserver"]
 ---> Running in f5ac6f75f356
Removing intermediate container f5ac6f75f356
 ---> ffdef51c9a42
Successfully built ffdef51c9a42
Successfully tagged gohttpserver:v1
zhiyong@zhiyong-vm-dev:~/dockerstudy/httpserver$ sudo docker images
REPOSITORY     TAG       IMAGE ID       CREATED          SIZE
gohttpserver   v1        ffdef51c9a42   2 minutes ago    83.9MB
ubuntuwithvm   latest    428e030662bc   56 minutes ago   170MB
ubuntu         latest    d2e4e1f51132   4 days ago       77.8MB
zhiyong@zhiyong-vm-dev:~/dockerstudy/httpserver$ sudo docker run -p 82:81
gohttpserver:v1
环境正常
```

浏览器：

```
http://192.168.88.50:82/healthz
```

可以看到succeed。并且Linux黑窗口会显示：

```
zhiyong@zhiyong-vm-dev:~/dockerstudy/httpserver$ sudo docker run -p 82:81
gohttpserver:v1
环境正常
调用healthz接口
```

说明容器构建成功。

```
zhiyong@zhiyong-vm-dev:~/dockerstudy/httpserver$ sudo docker ps -a
CONTAINER ID   IMAGE             COMMAND                  CREATED
STATUS                         PORTS     NAMES
93d3eac39a12   gohttpserver:v1   "/dockerstudy/httpse…"   4 minutes ago
Exited (2) 27 seconds ago                awesome_johnson
c515a662d46c   ubuntu            "/bin/bash"              About an hour ago
Exited (0) About an hour ago             goofy_heisenberg
```

ctrl+C退出即可。确保容器已经关闭。

推送至hub.docker.com

先创建新的仓库：zhiyongdocker/dockerstudy

官方给出了推送的命令：

```
docker tag local-image:tagname new-repo:tagname
docker push new-repo:tagname
```

先登录：

```
zhiyong@zhiyong-vm-dev:~/dockerstudy/httpserver$ sudo docker login
Login with your Docker ID to push and pull images from Docker Hub. If you don't
have a Docker ID, head over to https://hub.docker.com to create one.
Username: zhiyongdocker
Password:
WARNING! Your password will be stored unencrypted in /root/.docker/config.json.
Configure a credential helper to remove this warning. See
https://docs.docker.com/engine/reference/commandline/login/#credentials-store

Login Succeeded
```

修改镜像名称并推送：

```
zhiyong@zhiyong-vm-dev:~/dockerstudy/httpserver$ sudo docker images
REPOSITORY     TAG       IMAGE ID       CREATED             SIZE
gohttpserver   v1        ffdef51c9a42   29 minutes ago      83.9MB
ubuntuwithvm   latest    428e030662bc   About an hour ago   170MB
ubuntu         latest    d2e4e1f51132   4 days ago          77.8MB
zhiyong@zhiyong-vm-dev:~/dockerstudy/httpserver$ sudo docker tag gohttpserver:v1
zhiyongdocker/dockerstudy/httpserver:v1
zhiyong@zhiyong-vm-dev:~/dockerstudy/httpserver$ sudo docker images
REPOSITORY                              TAG       IMAGE ID       CREATED
  SIZE
gohttpserver                            v1        ffdef51c9a42   30 minutes ago
   83.9MB
zhiyongdocker/dockerstudy/httpserver    v1        ffdef51c9a42   30 minutes ago
   83.9MB
ubuntuwithvm                            latest    428e030662bc   About an hour ago
  170MB
ubuntu                                  latest    d2e4e1f51132   4 days ago
   77.8MB
zhiyong@zhiyong-vm-dev:~/dockerstudy/httpserver$ sudo docker push
zhiyongdocker/dockerstudy/httpserver:v1
#名称不对会失败
The push refers to repository [docker.io/zhiyongdocker/dockerstudy/httpserver]
b45d2bb1a7ec: Preparing
8a1c3d1d2aba: Preparing
e59fc9495612: Preparing
denied: requested access to the resource is denied
zhiyong@zhiyong-vm-dev:~/dockerstudy/httpserver$ sudo docker images
REPOSITORY                              TAG       IMAGE ID       CREATED
  SIZE
gohttpserver                            v1        ffdef51c9a42   34 minutes ago
   83.9MB
zhiyongdocker/dockerstudy/httpserver    v1        ffdef51c9a42   34 minutes ago
   83.9MB
ubuntuwithvm                            latest    428e030662bc   About an hour ago
  170MB
ubuntu                                  latest    d2e4e1f51132   4 days ago
   77.8MB
zhiyong@zhiyong-vm-dev:~/dockerstudy/httpserver$ sudo docker tag gohttpserver:v1
zhiyongdocker/dockerstudy:v1
zhiyong@zhiyong-vm-dev:~/dockerstudy/httpserver$ sudo docker images
REPOSITORY                              TAG       IMAGE ID       CREATED
  SIZE
```

```
gohttpserver                            v1        ffdef51c9a42   34 minutes ago
   83.9MB
zhiyongdocker/dockerstudy/httpserver    v1        ffdef51c9a42   34 minutes ago
   83.9MB
zhiyongdocker/dockerstudy               v1        ffdef51c9a42   34 minutes ago
   83.9MB
ubuntuwithvm                            latest    428e030662bc   About an hour ago
  170MB
ubuntu                                  latest    d2e4e1f51132   4 days ago
   77.8MB
zhiyong@zhiyong-vm-dev:~/dockerstudy/httpserver$ sudo docker push
zhiyongdocker/dockerstudy:v1
The push refers to repository [docker.io/zhiyongdocker/dockerstudy]
b45d2bb1a7ec: Pushed
8a1c3d1d2aba: Pushed
e59fc9495612: Pushed
v1: digest:
sha256:94edccac916aeccf113bfdcbd7ce2e35b774839790bdecdab417d5c049021068 size:
947
```

本地启动http server容器

```
zhiyong@zhiyong-vm-dev:~$ sudo docker run -p 83:81 zhiyongdocker/dockerstudy:v1
环境正常
调用healthz接口
^Czhiyong@zhiyong-vm-dev:~$ sudo dockeps -a
CONTAINER ID   IMAGE                           COMMAND                  CREATED
          STATUS                          PORTS     NAMES
8377df17b1e1   zhiyongdocker/dockerstudy:v1   "/dockerstudy/httpse…"   20
seconds ago       Exited (2) 5 seconds ago                 zen_ritchie
8e97bbad0d3b   zhiyongdocker/dockerstudy:v1   "/dockerstudy/httpse…"   About a
minute ago   Exited (2) About a minute ago             blissful_black
93d3eac39a12   gohttpserver:v1                "/dockerstudy/httpse…"   35
minutes ago       Exited (2) 31 minutes ago                awesome_johnson
c515a662d46c   ubuntu                         "/bin/bash"              2 hours
ago          Exited (0) 2 hours ago                   goofy_heisenberg
```

和之前的状态相同。浏览器：

```
http://192.168.88.50:83/healthz
```

可以看到succeed，说明容器正常。

进入容器查看 IP 配置

重启容器：

```
zhiyong@zhiyong-vm-dev:~$ sudo docker ps -a
```

```
CONTAINER ID   IMAGE                              COMMAND                  CREATED
       STATUS                       PORTS                                     NAMES
8377df17b1e1   zhiyongdocker/dockerstudy:v1   "/dockerstudy/httpse…"   10
minutes ago   Up 37 seconds               0.0.0.0:83->81/tcp, :::83->81/tcp
zen_ritchie
8e97bbad0d3b   zhiyongdocker/dockerstudy:v1   "/dockerstudy/httpse…"   11
minutes ago   Exited (2) 11 minutes ago
blissful_black
93d3eac39a12   gohttpserver:v1                "/dockerstudy/httpse…"   45
minutes ago   Exited (2) 41 minutes ago
awesome_johnson
c515a662d46c   ubuntu                             "/bin/bash"              2 hours
ago      Exited (0) 2 hours ago
goofy_heisenberg
zhiyong@zhiyong-vm-dev:~$ sudo docker stop 8377df17b1e1
8377df17b1e1
zhiyong@zhiyong-vm-dev:~$ sudo docker ps -a
CONTAINER ID   IMAGE                              COMMAND                  CREATED
       STATUS                       PORTS     NAMES
8377df17b1e1   zhiyongdocker/dockerstudy:v1   "/dockerstudy/httpse…"   10
minutes ago   Exited (2) 20 seconds ago             zen_ritchie
8e97bbad0d3b   zhiyongdocker/dockerstudy:v1   "/dockerstudy/httpse…"   12
minutes ago   Exited (2) 11 minutes ago             blissful_black
93d3eac39a12   gohttpserver:v1                "/dockerstudy/httpse…"   46
minutes ago   Exited (2) 41 minutes ago             awesome_johnson
c515a662d46c   ubuntu                             "/bin/bash"              2 hours
ago      Exited (0) 2 hours ago                goofy_heisenberg
zhiyong@zhiyong-vm-dev:~$ sudo docker restart 8377df17b1e1
8377df17b1e1
zhiyong@zhiyong-vm-dev:~$ sudo docker ps -a
CONTAINER ID   IMAGE                              COMMAND                  CREATED
       STATUS                       PORTS                                     NAMES
8377df17b1e1   zhiyongdocker/dockerstudy:v1   "/dockerstudy/httpse…"   10
minutes ago   Up 5 seconds                0.0.0.0:83->81/tcp, :::83->81/tcp
zen_ritchie
8e97bbad0d3b   zhiyongdocker/dockerstudy:v1   "/dockerstudy/httpse…"   12
minutes ago   Exited (2) 11 minutes ago
blissful_black
93d3eac39a12   gohttpserver:v1                "/dockerstudy/httpse…"   46
minutes ago   Exited (2) 41 minutes ago
awesome_johnson
c515a662d46c   ubuntu                             "/bin/bash"              2 hours
ago      Exited (0) 2 hours ago
goofy_heisenberg
```

查看Docker镜像的PID：

```
zhiyong@zhiyong-vm-dev:~$ sudo docker container top 8377df17b1e1
UID                 PID                 PPID                C
STIME               TTY                 TIME                CMD
root                17780               17760               0
01:39               ?                   00:00:00
/dockerstudy/httpserver/gohttpserver

zhiyong@zhiyong-vm-dev:~$ sudo docker inspect -f {{.State.Pid}} 8377df17b1e1
17780
```

查看帮助：

```
zhiyong@zhiyong-vm-dev:~$ nsenter --help

用法：
 nsenter [选项] [<程序> [<参数>...]]

以其他程序的名字空间运行某个程序。

选项：
 -a, --all              enter all namespaces
 -t, --target <pid>     要获取名字空间的目标进程
 -m, --mount[=<文件>]   进入 mount 名字空间
 -u, --uts[=<文件>]     进入 UTS 名字空间(主机名等)
 -i, --ipc[=<文件>]     进入 System V IPC 名字空间
 -n, --net[=<文件>]     进入网络名字空间
 -p, --pid[=<文件>]     进入 pid 名字空间
 -C, --cgroup[=<文件>]  进入 cgroup 名字空间
 -U, --user[=<文件>]    进入用户名字空间
 -T, --time[=<file>]    enter time namespace
 -S, --setuid <uid>     设置进入空间中的 uid
 -G, --setgid <gid>     设置进入名字空间中的 gid
     --preserve-credentials 不干涉 uid 或 gid
 -r, --root[=<目录>]     设置根目录
 -w, --wd[=<dir>]       设置工作目录
 -F, --no-fork          执行 <程序> 前不 fork
 -Z, --follow-context  根据 --target PID 设置 SELinux 环境

 -h, --help             display this help
 -V, --version          display version

更多信息请参阅 nsenter(1)。
```

根据PID进入Docker容器的网络命名空间：

```
zhiyong@zhiyong-vm-dev:~$ sudo nsenter -n -t17780
root@zhiyong-vm-dev:/home/zhiyong# ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group
default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
18: eth0@if19: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP
group default
    link/ether 02:42:ac:11:00:02 brd ff:ff:ff:ff:ff:ff link-netnsid 0
    inet 172.17.0.2/16 brd 172.17.255.255 scope global eth0
       valid_lft forever preferred_lft forever
root@zhiyong-vm-dev:/home/zhiyong# exit
注销
```

查看网络配置:

```
zhiyong@zhiyong-vm-dev:~$ su - root
密码:
su: 认证失败
zhiyong@zhiyong-vm-dev:~$ sudo su - root
root@zhiyong-vm-dev:~# cd /run/d
dbus/   docker/
root@zhiyong-vm-dev:~# cd /run/docker/
containerd/   libnetwork/   netns/        plugins/      runtime-runc/ swarm/
root@zhiyong-vm-dev:~# cd /run/docker/netns/
root@zhiyong-vm-dev:/run/docker/netns# ll
总用量 0
drwxr-xr-x 2 root root  60  5月  5 01:39 ./
drwx------ 8 root root 180  5月  4 23:38 ../
-r--r--r-- 1 root root   0  5月  5 01:39 2822acc443f5
root@zhiyong-vm-dev:/run/docker/netns# docker ps
CONTAINER ID   IMAGE                          COMMAND                  CREATED
       STATUS          PORTS                                  NAMES
8377df17b1e1   zhiyongdocker/dockerstudy:v1   "/dockerstudy/httpse…"   31
minutes ago   Up 20 minutes   0.0.0.0:83->81/tcp, :::83->81/tcp   zen_ritchie
root@zhiyong-vm-dev:/run/docker/netns# docker inspect 8377df17b1e1 | grep -i
sandbox
            "SandboxID":
"2822acc443f5b4f718b5a4c5d8cdf704868af013b37c257dcc6a69233d29d042",
            "SandboxKey": "/var/run/docker/netns/2822acc443f5",
root@zhiyong-vm-dev:/run/docker/netns# nsenter --
net=/var/run/docker/netns/2822acc443f5 sh
# iptables -nvL -t mangle
Chain PREROUTING (policy ACCEPT 0 packets, 0 bytes)
 pkts bytes target     prot opt in     out     source               destination

Chain INPUT (policy ACCEPT 0 packets, 0 bytes)
 pkts bytes target     prot opt in     out     source               destination

Chain FORWARD (policy ACCEPT 0 packets, 0 bytes)
 pkts bytes target     prot opt in     out     source               destination

Chain OUTPUT (policy ACCEPT 0 packets, 0 bytes)
 pkts bytes target     prot opt in     out     source               destination
```

```
Chain POSTROUTING (policy ACCEPT 0 packets, 0 bytes)
 pkts bytes target     prot opt in     out     source               destination
# ipvsadm -ln
sh: 2: ipvsadm: not found
# exit
root@zhiyong-vm-dev:/run/docker/netns#
```

相关的namespace配置如下：

```
root@zhiyong-vm-dev:/proc# ll /proc/17780/ns
总用量 0
dr-x--x--x 2 root root 0  5月  5 01:39 ./
dr-xr-xr-x 9 root root 0  5月  5 01:39 ../
lrwxrwxrwx 1 root root 0  5月  5 02:05 cgroup -> 'cgroup:[4026532761]'
lrwxrwxrwx 1 root root 0  5月  5 02:05 ipc -> 'ipc:[4026532633]'
lrwxrwxrwx 1 root root 0  5月  5 02:05 mnt -> 'mnt:[4026532631]'
lrwxrwxrwx 1 root root 0  5月  5 01:39 net -> 'net:[4026532636]'
lrwxrwxrwx 1 root root 0  5月  5 02:05 pid -> 'pid:[4026532634]'
lrwxrwxrwx 1 root root 0  5月  5 02:05 pid_for_children -> 'pid:[4026532634]'
lrwxrwxrwx 1 root root 0  5月  5 02:05 time -> 'time:[4026531834]'
lrwxrwxrwx 1 root root 0  5月  5 02:05 time_for_children -> 'time:[4026531834]'
lrwxrwxrwx 1 root root 0  5月  5 02:05 user -> 'user:[4026531837]'
lrwxrwxrwx 1 root root 0  5月  5 02:05 uts -> 'uts:[4026532632]'
```

再次重启容器：

```
zhiyong@zhiyong-vm-dev:~$ sudo docker restart 8377df17b1e1
8377df17b1e1
zhiyong@zhiyong-vm-dev:~$ sudo docker inspect -f {{.State.Pid}} 8377df17b1e1
18210
zhiyong@zhiyong-vm-dev:~$ sudo nsenter -n -t18210
root@zhiyong-vm-dev:/home/zhiyong# ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group
default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
20: eth0@if21: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP
group default
    link/ether 02:42:ac:11:00:02 brd ff:ff:ff:ff:ff:ff link-netnsid 0
    inet 172.17.0.2/16 brd 172.17.255.255 scope global eth0
       valid_lft forever preferred_lft forever
root@zhiyong-vm-dev:/home/zhiyong# exit
注销
```

可以看到PID变化，但是容器IP并没有变化。