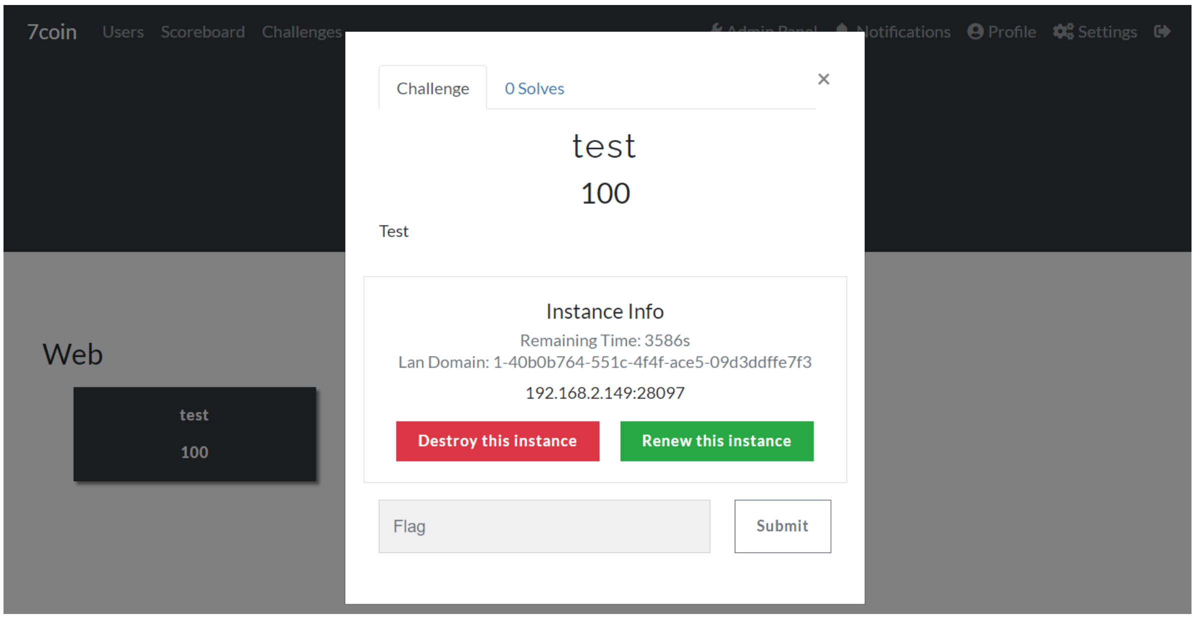
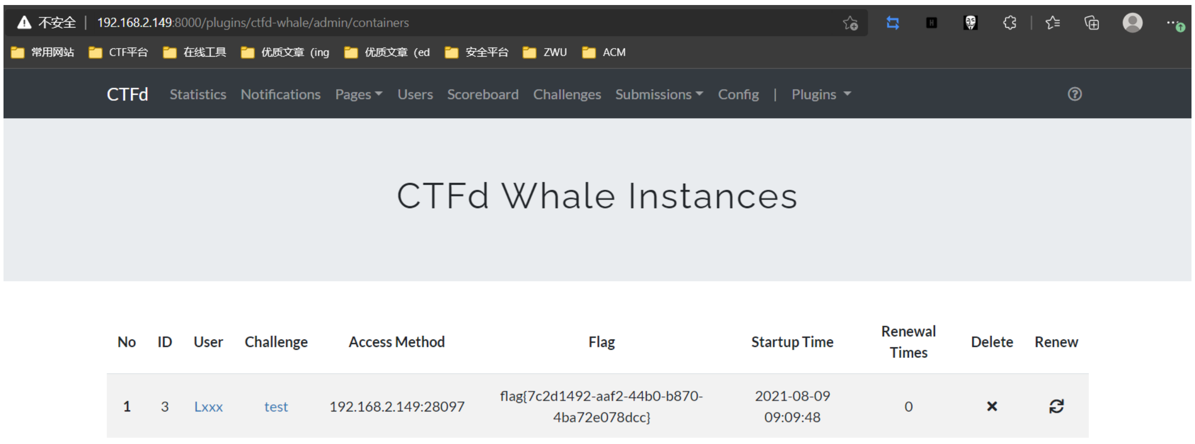
CTFd动态靶机搭建（详细总结）

成品展示：

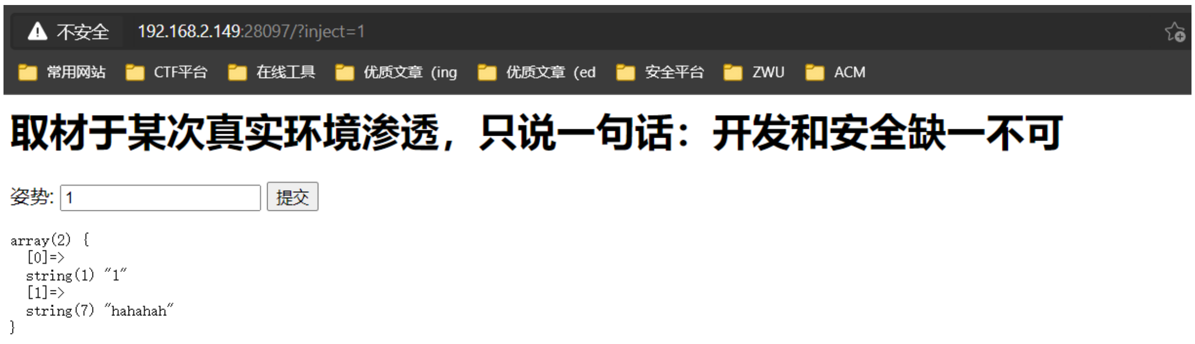


Test

Instance Info
Remaining Time: 3586s
Lan Domain: 1-40b0b764-551c-4f4f-ace5-09d3ddffe7f3
192.168.2.149:28097



No	ID	User	Challenge	Access Method	Flag	Startup Time	Renewal Times	Delete	Renew
1	3	Lxxx	test	192.168.2.149:28097	flag{7c2d1492-aaf2-44b0-b870-4ba72e078dcc}	2021-08-09 09:09:48	0	×	⟳



取材于某次真实环境渗透，只说一句话：开发和安全缺一不可

姿势: 1

```
array(2) {
  [0]=>
  string(1) "1"
  [1]=>
  string(7) "hahahah"
}
```

环境：

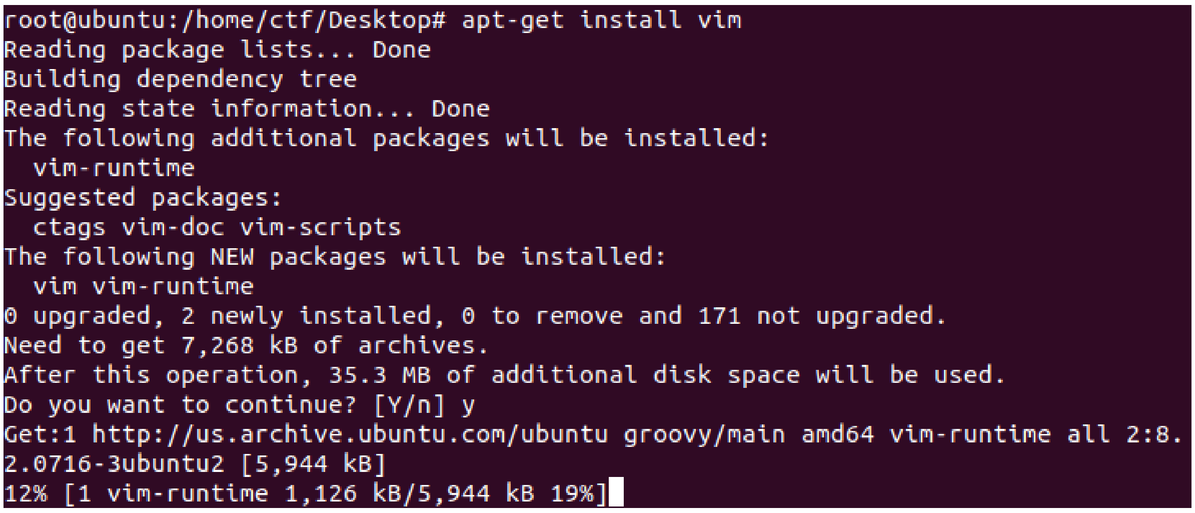
- 主机：Ubuntu 20.10
  - 本篇文章使用的机器的IP：192.168.2.151
- Docker版本：20.10.2
- Docker-compose版本：1.25.0

搭建步骤：

系统环境配置：

安装vim：

```
apt-get install vim
```



```
root@ubuntu:/home/ctf/Desktop# apt-get install vim
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following additional packages will be installed:
  vim-runtime
Suggested packages:
  ctags vim-doc vim-scripts
The following NEW packages will be installed:
  vim vim-runtime
0 upgraded, 2 newly installed, 0 to remove and 171 not upgraded.
Need to get 7,268 kB of archives.
After this operation, 35.3 MB of additional disk space will be used.
Do you want to continue? [Y/n] y
Get:1 http://us.archive.ubuntu.com/ubuntu groovy/main amd64 vim-runtime all 2:8.
2.0716-3ubuntu2 [5,944 kB]
12% [1 vim-runtime 1,126 kB/5,944 kB 19%]
```

因为是官方源，所以会比较慢，后面会进行换源。

切换镜像源：

这里以**Ubuntu20.10**为例：ubuntu | 镜像站使用帮助 | 清华大学开源软件镜像站 | Tsinghua Open Source Mirror

```
vim /etc/apt/sources.list
```

将原有的内容删除，往**source.list**里面加入以下内容，加完之后保存退出。

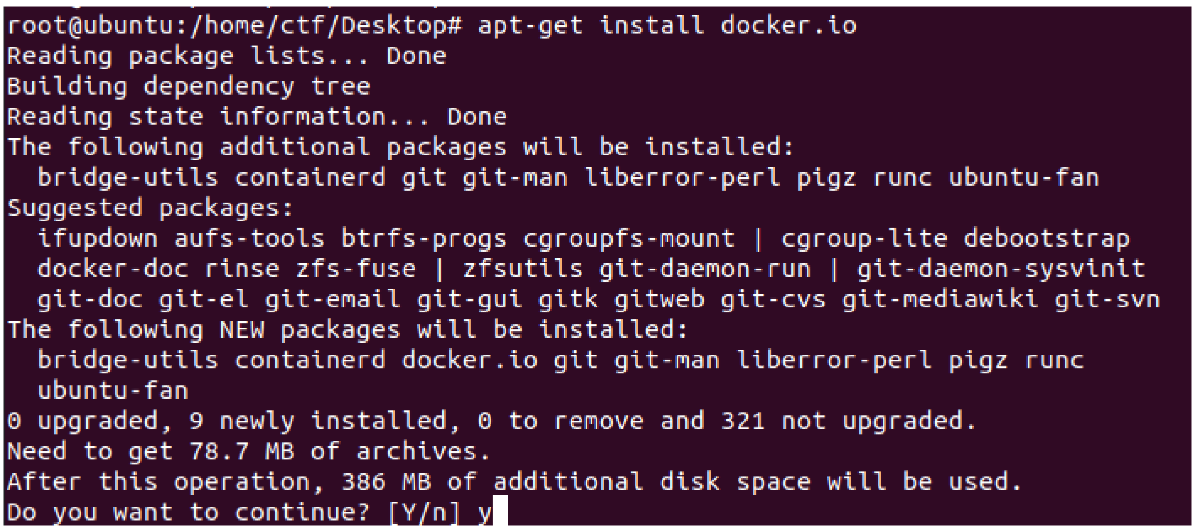
```
# 默认注释了源码镜像以提高 apt update 速度，如有需要可自行取消注释
deb https://mirrors.tuna.tsinghua.edu.cn/ubuntu/ groovy main restricted universe
multiverse
# deb-src https://mirrors.tuna.tsinghua.edu.cn/ubuntu/ groovy main restricted
universe multiverse
deb https://mirrors.tuna.tsinghua.edu.cn/ubuntu/ groovy-updates main restricted
universe multiverse
# deb-src https://mirrors.tuna.tsinghua.edu.cn/ubuntu/ groovy-updates main
restricted universe multiverse
deb https://mirrors.tuna.tsinghua.edu.cn/ubuntu/ groovy-backports main restricted
universe multiverse
# deb-src https://mirrors.tuna.tsinghua.edu.cn/ubuntu/ groovy-backports main
restricted universe multiverse
deb https://mirrors.tuna.tsinghua.edu.cn/ubuntu/ groovy-security main restricted
universe multiverse
# deb-src https://mirrors.tuna.tsinghua.edu.cn/ubuntu/ groovy-security main
restricted universe multiverse

# 预发布软件源，不建议启用
# deb https://mirrors.tuna.tsinghua.edu.cn/ubuntu/ groovy-proposed main
restricted universe multiverse
# deb-src https://mirrors.tuna.tsinghua.edu.cn/ubuntu/ groovy-proposed main
restricted universe multiverse
```

注意：一定要更新一下镜像源！！（不然后面没法安装包）

```
apt-get update
```

安装docker：

```
apt-get install docker.io
```

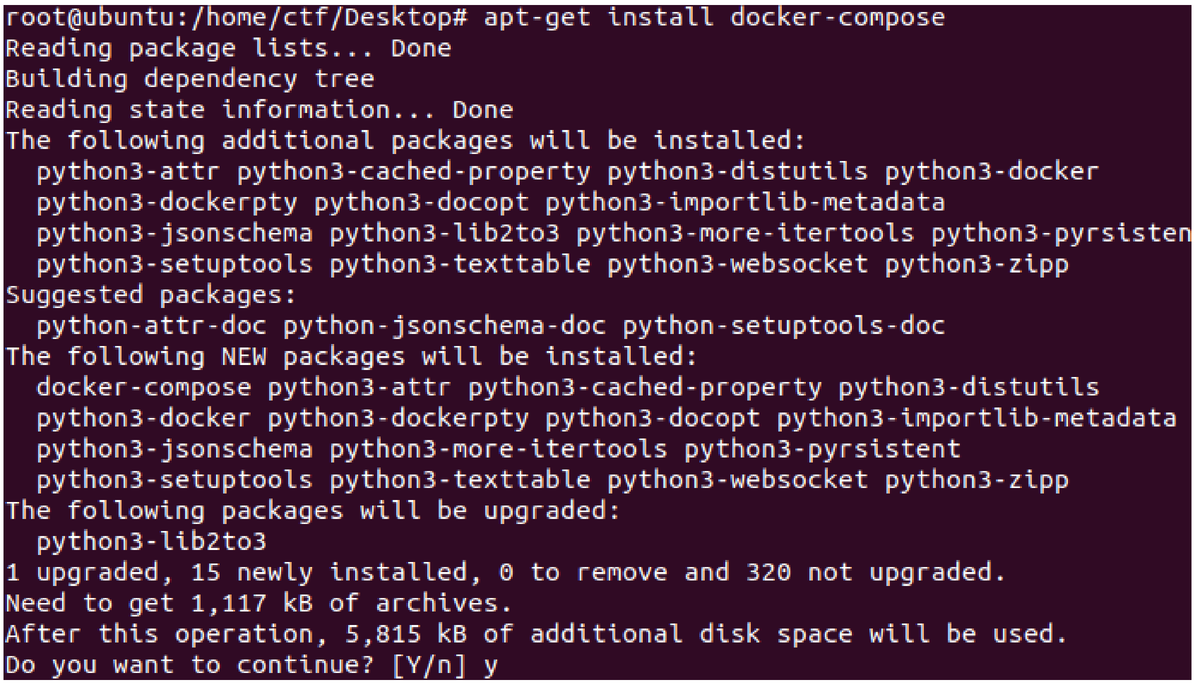


```
root@ubuntu:/home/ctf/Desktop# apt-get install docker.io
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following additional packages will be installed:
  bridge-utils containerd git git-man liberror-perl pigz runc ubuntu-fan
Suggested packages:
  ifupdown aufs-tools btrfs-progs cgroupfs-mount | cgroup-lite debootstrap
  docker-doc rinse zfs-fuse | zfsutils git-daemon-run | git-daemon-sysvinit
  git-doc git-el git-email git-gui gitk gitweb git-cvs git-mediawiki git-svn
The following NEW packages will be installed:
  bridge-utils containerd docker.io git git-man liberror-perl pigz runc
  ubuntu-fan
0 upgraded, 9 newly installed, 0 to remove and 321 not upgraded.
Need to get 78.7 MB of archives.
After this operation, 386 MB of additional disk space will be used.
Do you want to continue? [Y/n] y
```

输入**y**后回车

安装docker-compose：

```
apt-get install docker-compose
```

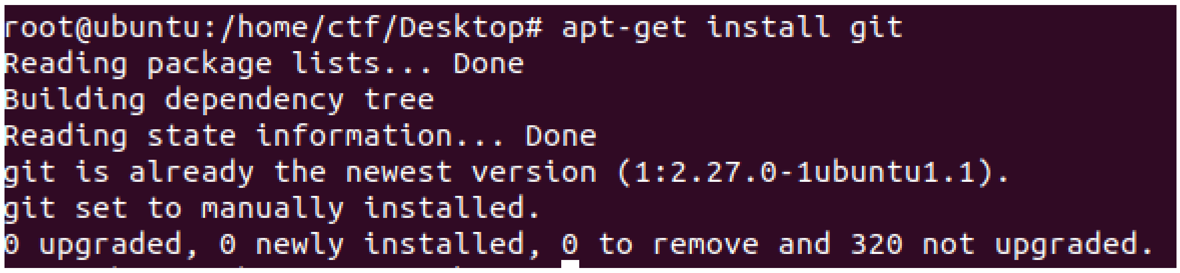


```
root@ubuntu:/home/ctf/Desktop# apt-get install docker-compose
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following additional packages will be installed:
  python3-attr python3-cached-property python3-distutils python3-docker
  python3-dockerpty python3-docopt python3-importlib-metadata
  python3-jsonschema python3-lib2to3 python3-more-itertools python3-pyrsisten
  python3-setuptools python3-texttable python3-websocket python3-zipp
Suggested packages:
  python-attr-doc python-jsonschema-doc python-setuptools-doc
The following NEW packages will be installed:
  docker-compose python3-attr python3-cached-property python3-distutils
  python3-docker python3-dockerpty python3-docopt python3-importlib-metadata
  python3-jsonschema python3-more-itertools python3-pyrsistent
  python3-setuptools python3-texttable python3-websocket python3-zipp
The following packages will be upgraded:
  python3-lib2to3
1 upgraded, 15 newly installed, 0 to remove and 320 not upgraded.
Need to get 1,117 kB of archives.
After this operation, 5,815 kB of additional disk space will be used.
Do you want to continue? [Y/n] y
```

同样按**y**后回车

安装git：

```
apt-get intsall git
```



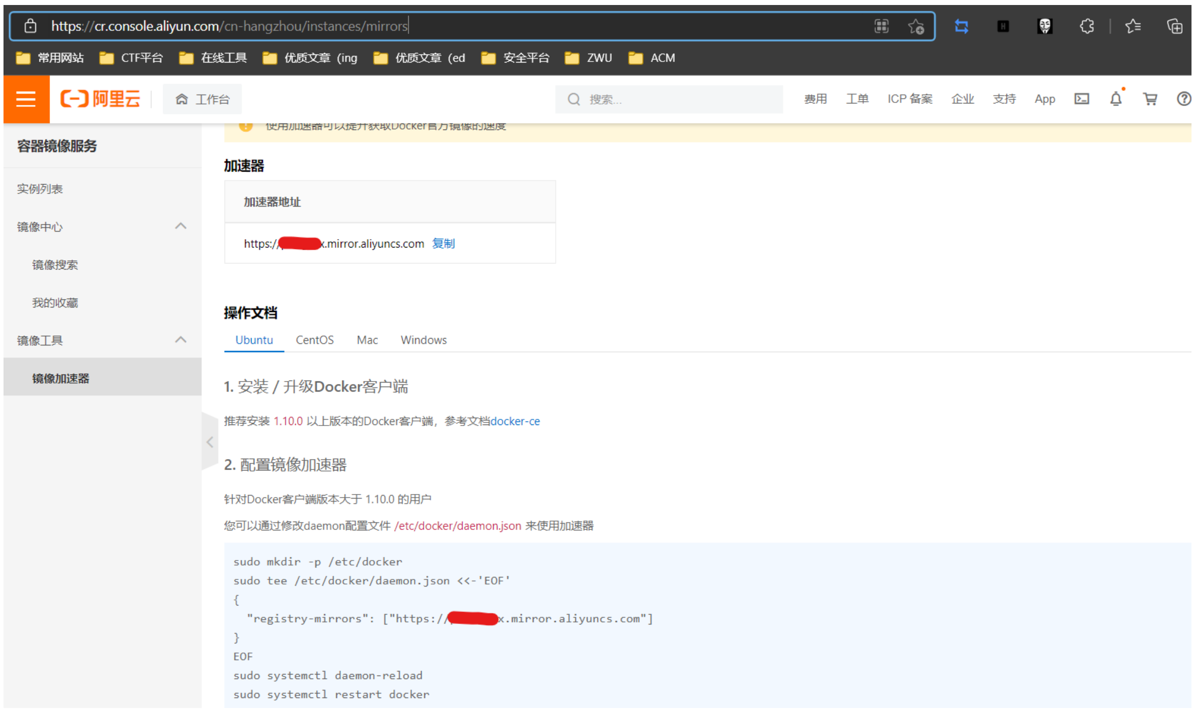
```
root@ubuntu:/home/ctf/Desktop# apt-get install git
Reading package lists... Done
Building dependency tree
Reading state information... Done
git is already the newest version (1:2.27.0-1ubuntu1.1).
git set to manually installed.
0 upgraded, 0 newly installed, 0 to remove and 320 not upgraded.
```

一般来说，安装过docker之后，git也已经安装好了，不过我们还是再确认一下。

docker镜像加速：

进入阿里云镜像服务官网：[容器镜像服务 (aliyun.com)]

根据下方教程配置自己的Docker静态加速器。


https://cr.console.aliyun.com/cn-hangzhou/instances/mirrors
常用网站 CTF平台 在线工具 优质文章（ing 优质文章（ed 安全平台 ZWU ACM
阿里云 工作台 搜索... 费用 工单 ICP 备案 企业 支持 App
容器镜像服务
实例列表
镜像中心
镜像搜索
我的收藏
镜像工具
镜像加速器
Ubuntu CentOS Mac Windows
复制

使用加速器可以提升获取Docker官方镜像的速度

加速器

加速器地址

https://[redacted]x.mirror.aliyuncs.com

操作文档

1. 安装 / 升级Docker客户端

推荐安装 1.10.0 以上版本的Docker客户端，参考文档docker-ce

2. 配置镜像加速器

针对Docker客户端版本大于 1.10.0 的用户

您可以通过修改daemon配置文件 /etc/docker/daemon.json 来使用加速器

```
sudo mkdir -p /etc/docker
sudo tee /etc/docker/daemon.json <<-'EOF'
{
  "registry-mirrors": ["https://[redacted]x.mirror.aliyuncs.com"]
}
EOF
sudo systemctl daemon-reload
sudo systemctl restart docker
```

镜像地址每个人都不一样，根据每个人的情况设置。

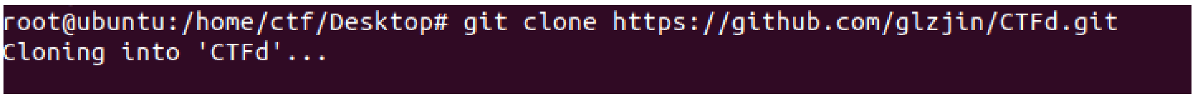
```
sudo mkdir -p /etc/docker
sudo tee /etc/docker/daemon.json <<-'EOF'
{
  "registry-mirrors": ["https://xxxxxxxx.mirror.aliyuncs.com"]
}
EOF
sudo systemctl daemon-reload
sudo systemctl restart docker
```

下载CTFd：

这一个CTFd是赵师傅改写的，链接在下方。之所以要使用赵师傅改写的CTFd，是因为官方CTFd可能会与动态靶机插件ctf-whale冲突。

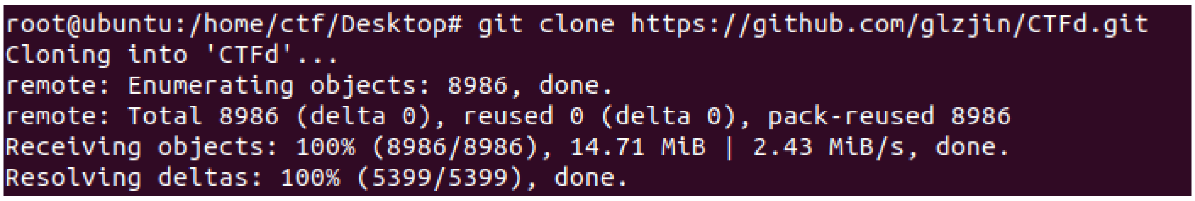
- https://github.com/glzjin/CTFd.git

```
git clone https://github.com/glzjin/CTFd.git
```



```
root@ubuntu:/home/ctf/Desktop# git clone https://github.com/glzjin/CTFd.git
Cloning into 'CTFd'...
```

如果clone的过程中卡住了，可以尝试Ctrl+C将命令掐断，然后重新执行，实在不行的话，可以挂代理。



```
root@ubuntu:/home/ctf/Desktop# git clone https://github.com/glzjin/CTFd.git
Cloning into 'CTFd'...
remote: Enumerating objects: 8986, done.
remote: Total 8986 (delta 0), reused 0 (delta 0), pack-reused 8986
Receiving objects: 100% (8986/8986), 14.71 MiB | 2.43 MiB/s, done.
Resolving deltas: 100% (5399/5399), done.
```

下载frpc：

下载frpc：

```
wget
https://github.com/fatedier/frp/releases/download/v0.29.0/frp_0.29.0_linux_amd64
.tar.gz
```

同样的，如果clone的过程中卡住了，可以尝试Ctrl+C将命令掐断，然后重新执行，实在不行的话，可以挂代理。

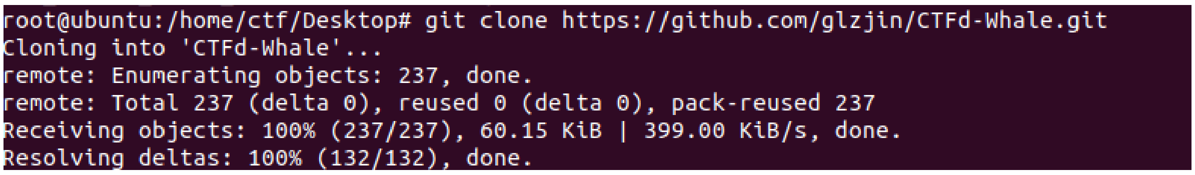
将下载下来的frpc解压：

```
tar -zxvf frp_0.29.0_linux_amd64.tar.gz
```

下载ctfd-whale：

地址：https://github.com/glzjin/CTFd-Whale

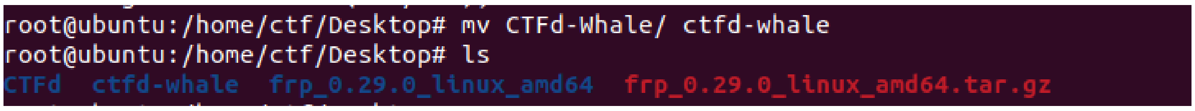
```
git clone https://github.com/glzjin/CTFd-Whale.git
```



```
root@ubuntu:/home/ctf/Desktop# git clone https://github.com/glzjin/CTFd-Whale.git
Cloning into 'CTFd-Whale'...
remote: Enumerating objects: 237, done.
remote: Total 237 (delta 0), reused 0 (delta 0), pack-reused 237
Receiving objects: 100% (237/237), 60.15 KiB | 399.00 KiB/s, done.
Resolving deltas: 100% (132/132), done.
```

注意：这个时候要将CTFd-Whale文件夹重命名为小写。

```
mv CTFd-Whale/ ctfd-whale
```



```
root@ubuntu:/home/ctf/Desktop# mv CTFd-Whale/ ctfd-whale
root@ubuntu:/home/ctf/Desktop# ls
CTFd  ctfd-whale  frp_0.29.0_linux_amd64  frp_0.29.0_linux_amd64.tar.gz
```

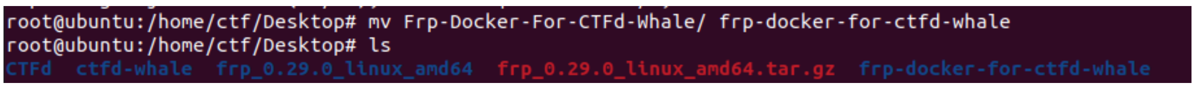
下载docker版本的frps：

地址：https://github.com/glzjin/Frp-Docker-For-CTFd-Whale

```
git clone https://github.com/glzjin/Frp-Docker-For-CTFd-Whale
```

注意：将Frp-Docker-For-CTFd-Whale也重命名为小写

```
mv Frp-Docker-For-CTFd-Whale/ frp-docker-for-ctfd-whale
```



```
root@ubuntu:/home/ctf/Desktop# mv Frp-Docker-For-CTFd-Whale/ frp-docker-for-ctfd-whale
root@ubuntu:/home/ctf/Desktop# ls
CTFd  ctfd-whale  frp_0.29.0_linux_amd64  frp_0.29.0_linux_amd64.tar.gz  frp-docker-for-ctfd-whale
```

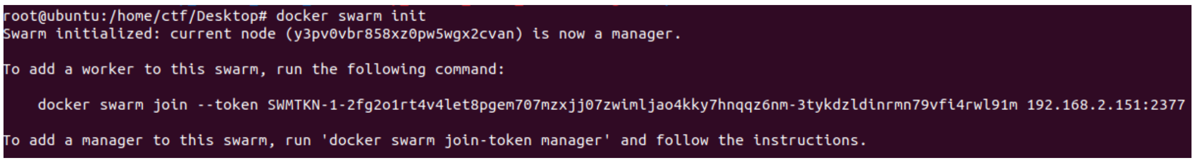
CTFd环境配置：

接下来就开始配置CTFd的一些文件了！

Docker集群设置：

先初始化：

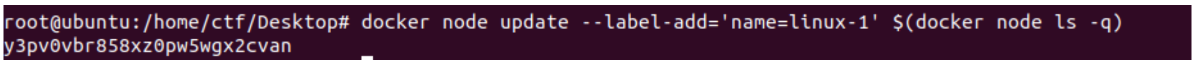
```
docker swarm init
```



```
root@ubuntu:/home/ctf/Desktop# docker swarm init
Swarm initialized: current node (y3pv0vbr858xz0pw5wgx2cvan) is now a manager.

To add a worker to this swarm, run the following command:

    docker swarm join --token SWMTKN-1-2fg2o1rt4v4let8pgem707mzxjj07zwimljao4kky7hnqqz6nm-3tykdzldinrmn79vfi4rwl91m 192.168.2.151:2377

To add a manager to this swarm, run 'docker swarm join-token manager' and follow the instructions.
```

将刚刚初始化的这个集群加入到节点当中，命令执行后，返回的就是节点ID了，暂时不用管这个节点ID。

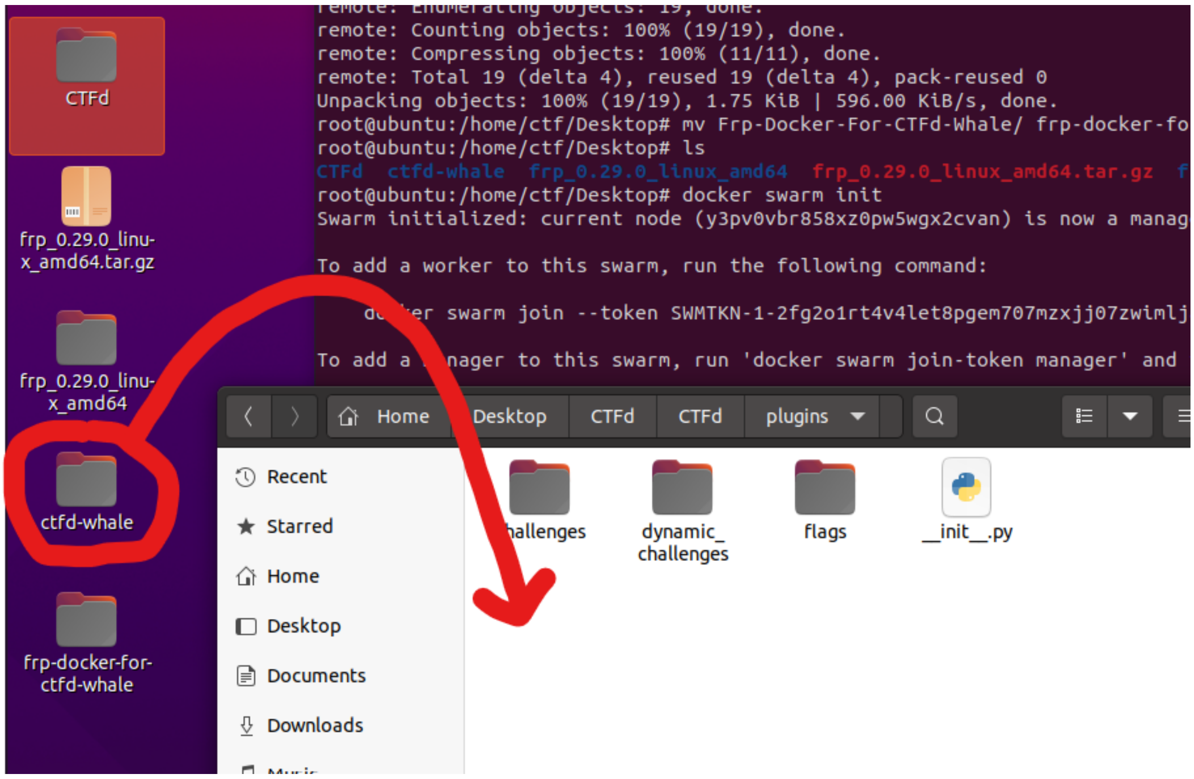
```
docker node update --label-add='name=linux-1' $(docker node ls -q)
```



```
root@ubuntu:/home/ctf/Desktop# docker node update --label-add='name=linux-1' $(docker node ls -q)
y3pv0vbr858xz0pw5wgx2cvan
```

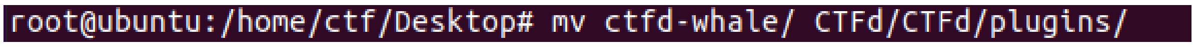
将ctfd-whale放入CTFd的插件目录中：

如下图所示：

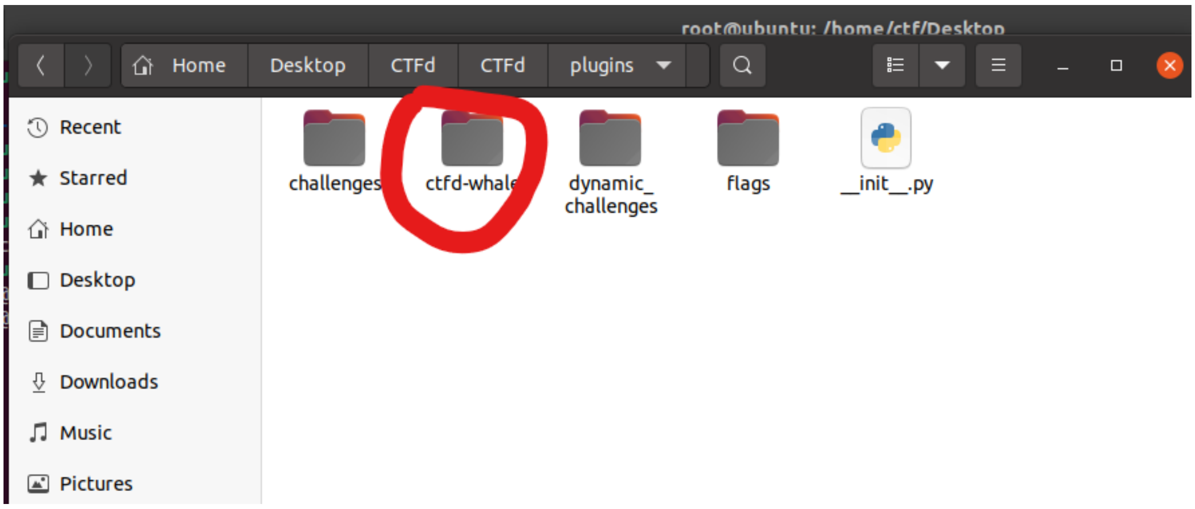


```
remote: Enumerating objects: 19, done.
remote: Counting objects: 100% (19/19), done.
remote: Compressing objects: 100% (11/11), done.
remote: Total 19 (delta 4), reused 19 (delta 4), pack-reused 0
Unpacking objects: 100% (19/19), 1.75 KiB | 596.00 KiB/s, done.
root@ubuntu:/home/ctf/Desktop# mv Frp-Docker-For-CTFd-Whale/ frp-docker-fo
root@ubuntu:/home/ctf/Desktop# ls
CTFd  ctfd-whale  frp_0.29.0_linux_amd64  frp_0.29.0_linux_amd64.tar.gz  f
root@ubuntu:/home/ctf/Desktop# docker swarm init
Swarm initialized: current node (y3pv0vbr858xz0pw5wgx2cvan) is now a manag

To add a worker to this swarm, run the following command:

    docker swarm join --token SWMTKN-1-2fg2o1rt4v4let8pgem707mzxjj07zwimlj

To add a manager to this swarm, run 'docker swarm join-token manager' and
```

```
mv ctfd-whale/ CTFd/CTFd/plugins/
```



```
root@ubuntu:/home/ctf/Desktop# mv ctfd-whale/ CTFd/CTFd/plugins/
```



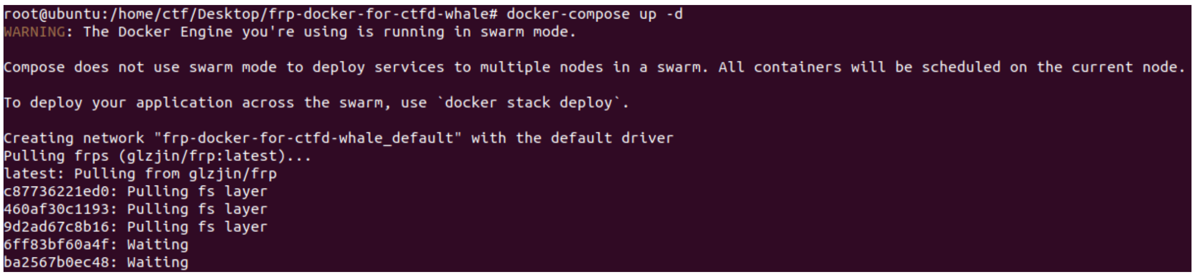
启动docker版本的frps及frps配置：

进入该目录：

```
cd frp-docker-for-ctfd-whale/
```

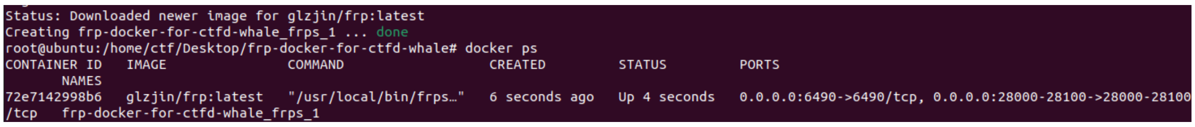
启动该docker：

```
docker-compose up -d
```



```
root@ubuntu:/home/ctf/Desktop/frp-docker-for-ctfd-whale# docker-compose up -d
WARNING: The Docker Engine you're using is running in swarm mode.

Compose does not use swarm mode to deploy services to multiple nodes in a swarm. All containers will be scheduled on the current node.

To deploy your application across the swarm, use `docker stack deploy`.

Creating network "frp-docker-for-ctfd-whale_default" with the default driver
Pulling frps (glzjin/frp:latest)...
latest: Pulling from glzjin/frp
c87736221ed0: Pulling fs layer
460af30c1193: Pulling fs layer
9d2ad67c8b16: Pulling fs layer
6ff83bf60a4f: Waiting
ba2567b0ec48: Waiting
```

耐心等待镜像构建。



```
Status: Downloaded newer image for glzjin/frp:latest
Creating frp-docker-for-ctfd-whale_frps_1 ... done
root@ubuntu:/home/ctf/Desktop/frp-docker-for-ctfd-whale# docker ps
CONTAINER ID   IMAGE               COMMAND                  CREATED          STATUS          PORTS
       NAMES
72e7142998b6   glzjin/frp:latest   "/usr/local/bin/frps…"   6 seconds ago    Up 4 seconds    0.0.0.0:6490->6490/tcp, 0.0.0.0:28000-28100->28000-28100
/tcp   frp-docker-for-ctfd-whale_frps_1
```

完成以后，可以使用`docker ps`查看是否正在运行。

这个时候我们查看一下frps的配置。

frps/frps.ini文件如下：

```
[common]
bind_port = 6490
token = randomme
```

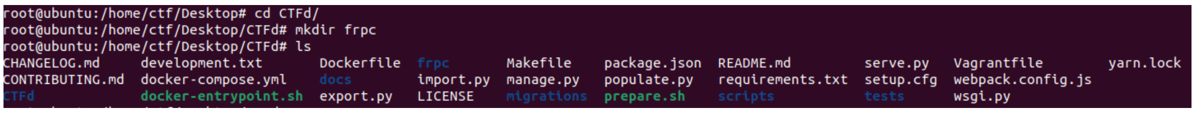
这里的token可以改也可以不改，一般也就不改了。

将frpc传入CTFd中：

这一步骤实际上就是：将frpc，frpc.ini，frpc_full.ini，LICENSE这四个文件放在CTFd/frpc文件夹中。

进入CTFd目录中，新建一个frpc文件夹

```
cd CTFd/
mkdir frpc
```



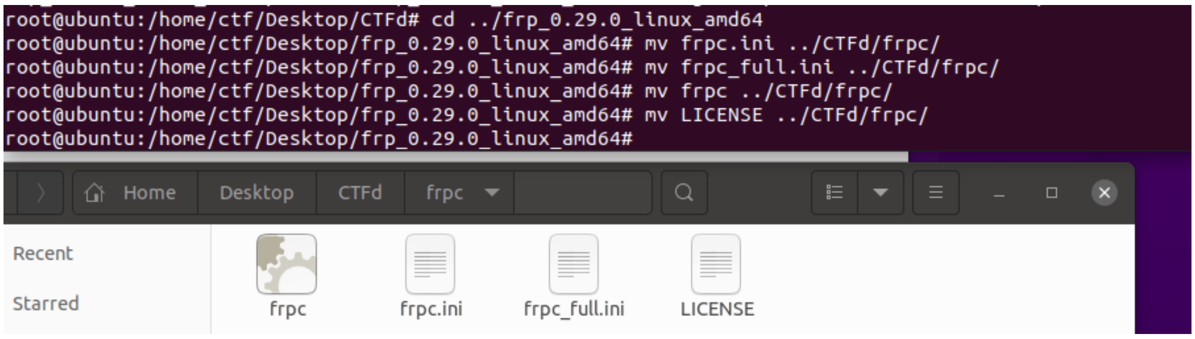
```
root@ubuntu:/home/ctf/Desktop# cd CTFd/
root@ubuntu:/home/ctf/Desktop/CTFd# mkdir frpc
root@ubuntu:/home/ctf/Desktop/CTFd# ls
CHANGELOG.md     development.txt       Dockerfile  frpc       Makefile    package.json  README.md         serve.py   Vagrantfile        yarn.lock
CONTRIBUTING.md  docker-compose.yml    docs        import.py  manage.py   populate.py   requirements.txt  setup.cfg  webpack.config.js
CTFd             docker-entrypoint.sh  export.py   LICENSE    migrations  prepare.sh    scripts           tests      wsgi.py
```

进入frpc的目录（frp_0.29.0_linux_amd64）

```
cd ../frp_0.29.0_linux_amd64
```

将上述四个文件移动到frpc文件夹中

```
mv frpc.ini ../CTFd/frpc/
mv frpc_full.ini ../CTFd/frpc/
mv frpc ../CTFd/frpc/
mv LICENSE ../CTFd/frpc/
```



```
root@ubuntu:/home/ctf/Desktop/CTFd# cd ../frp_0.29.0_linux_amd64
root@ubuntu:/home/ctf/Desktop/frp_0.29.0_linux_amd64# mv frpc.ini ../CTFd/frpc/
root@ubuntu:/home/ctf/Desktop/frp_0.29.0_linux_amd64# mv frpc_full.ini ../CTFd/frpc/
root@ubuntu:/home/ctf/Desktop/frp_0.29.0_linux_amd64# mv frpc ../CTFd/frpc/
root@ubuntu:/home/ctf/Desktop/frp_0.29.0_linux_amd64# mv LICENSE ../CTFd/frpc/
root@ubuntu:/home/ctf/Desktop/frp_0.29.0_linux_amd64#
```

修改frpc.ini文件：

进入frpc目录中，修改frpc.ini文件

```
[common]
token = randomme
server_addr = 172.1.0.4
server_port = 6490
pool_count = 200
tls_enable = true

admin_addr = 172.1.0.3
admin_port = 7400
```

配置Dockerfile：

进入ctfd目录，将下方内容复制到Dockerfile中

```
FROM python:3.7-alpine
RUN sed -i 's/dl-cdn.alpinelinux.org/mirrors.ustc.edu.cn/g' /etc/apk/repositories
&& \
    apk update && \
    apk add linux-headers libffi-dev gcc make musl-dev py-pip mysql-client git
openssl-dev   #这里注意1
RUN adduser -D -u 1001 -s /bin/bash ctfd

WORKDIR /opt/CTFd
RUN mkdir -p /opt/CTFd /var/log/CTFd /var/uploads

COPY requirements.txt .

RUN apk add gcc
RUN apk add musl-dev
RUN apk add libxslt-dev
RUN apk add g++
RUN apk add make
RUN apk add libffi-dev
RUN apk add openssl-dev
RUN apk add libtool

RUN pip install -r requirements.txt -i
https://mirrors.tuna.tsinghua.edu.cn/pypi/web/simple/   #这里注意2

COPY . /opt/CTFd

RUN for d in CTFd/plugins/*; do \
      if [ -f "$d/requirements.txt" ]; then \
        pip install -r $d/requirements.txt -i
https://mirrors.tuna.tsinghua.edu.cn/pypi/web/simple/ ; \
      fi; \
    done; #同样注意2

RUN chmod +x /opt/CTFd/docker-entrypoint.sh
RUN chown -R 1001:1001 /opt/CTFd
RUN chown -R 1001:1001 /var/log/CTFd /var/uploads

USER 1001
EXPOSE 8000
ENTRYPOINT ["/opt/CTFd/docker-entrypoint.sh"]
```

配置docker-compose.yml文件：

```
version: '2.2'

services:
  ctfd-nginx:
    image: nginx:1.17
    volumes:
      - ./nginx/http.conf:/etc/nginx/nginx.conf   #这里注意
    user: root
    restart: always
    ports:
      #- "85:80"     #我将这里注释掉了，这里通过nginx转发感觉速度访问速度会变慢，可能因为我
的配置问题，多次尝试之后直接开8000端口访问不会对服务造成影响
      - "443:443"
    networks:
        default:
        internal:
    depends_on:
      - ctfd
    cpus: '1.00'  #可改
    mem_limit: 150M     #可改
  ctfd:
    build: .
    user: root
    restart: always
    ports:
      - "8000:8000"     #这里原本没开端口，直接打开访问网站速度会加快
    environment:
      - UPLOAD_FOLDER=/var/uploads
      - DATABASE_URL=mysql+pymysql://root:ctfd@db/ctfd
      - REDIS_URL=redis://cache:6379
      - WORKERS=1
      - LOG_FOLDER=/var/log/CTFd
      - ACCESS_LOG=-
      - ERROR_LOG=-
      - REVERSE_PROXY=true
    volumes:
      - .data/CTFd/logs:/var/log/CTFd
      - .data/CTFd/uploads:/var/uploads
      - .:/opt/CTFd:ro
      - /var/run/docker.sock:/var/run/docker.sock     #这里是添加的
    depends_on:
      - db
    networks:
        default:
        internal:
        frp:
            ipv4_address: 172.1.0.2
    cpus: '1.00'     #可改
    mem_limit: 450M     #可改

  db:
    image: mariadb:10.4
    restart: always
    environment:
      - MYSQL_ROOT_PASSWORD=ctfd
      - MYSQL_USER=ctfd
      - MYSQL_PASSWORD=ctfd
    volumes:
      - .data/mysql:/var/lib/mysql
    networks:
        internal:
    # This command is required to set important mariadb defaults
    command: [mysqld, --character-set-server=utf8mb4, --collation-
server=utf8mb4_unicode_ci, --wait_timeout=28800, --log-warnings=0]
    cpus: '1.00'     #可改
    mem_limit: 750M     #可改

  cache:
    image: redis:4
    restart: always
    volumes:
      - .data/redis:/data
    networks:
        internal:
    cpus: '1.00'     #可改
    mem_limit: 450M     #可改

  frpc:
    image: glzjin/frp:latest     #赵师傅tql
    restart: always
    volumes:
      - ./frpc:/conf/     #这里注意
    entrypoint:
        - /usr/local/bin/frpc
        - -c
        - /conf/frpc.ini
    networks:
        frp:
            ipv4_address: 172.1.0.3  #记住此处
        frp-containers:
    cpus: '1.00'     #可改
    mem_limit: 250M     #可改

networks:
    default:
    internal:
        internal: true
    frp:
        driver: bridge
        ipam:
            config:
                - subnet: 172.1.0.0/16
    frp-containers:
        driver: overlay
        internal: true
        ipam:
            config:
                - subnet: 172.2.0.0/16
```

配置requirements.txt

这里主要是修改gevent的版本号，原本是1.4.0的，我这边修改成了20.9.0

```
Flask==1.1.1
Werkzeug==0.16.0
Flask-SQLAlchemy==2.4.1
Flask-Caching==1.4.0
Flask-Migrate==2.5.2
Flask-Script==2.0.6
SQLAlchemy==1.3.11
SQLAlchemy-Utils==0.36.0
passlib==1.7.2
bcrypt==3.1.7
six==1.13.0
itsdangerous==1.1.0
requests>=2.20.0
PyMySQL==0.9.3
gunicorn==19.9.0
normality==2.0.0
dataset==1.1.2
mistune==0.8.4
netaddr==0.7.19
redis==3.3.11
datafreeze==0.1.0
python-dotenv==0.10.3
flask-restplus==0.13.0
pathlib2==2.3.5
flask-marshmallow==0.10.1
marshmallow-sqlalchemy==0.17.0
boto3==1.10.39
marshmallow==2.20.2
gevent==20.9.0
```

配置nginx：

在docker-compose.yml的目录下，新建一个nginx文件夹

```
mkdir nginx
```

进入nginx文件夹

```
cd nginx
```

创建一个文件http.conf，输入以下内容：

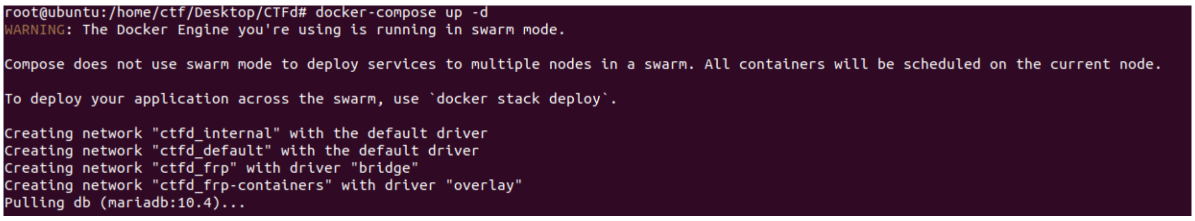
```
worker_processes 4;
events {
  worker_connections 1024;
}
http {
  # Configuration containing list of application servers
  upstream app_servers {
    server ctfd:8000;
  }
  server {
    listen 80;
    client_max_body_size 4G;
    # Handle Server Sent Events for Notifications
    location /events {
      proxy_pass http://app_servers;
      proxy_set_header Connection '';
      proxy_http_version 1.1;
      chunked_transfer_encoding off;
      proxy_buffering off;
      proxy_cache off;
      proxy_redirect off;
      proxy_set_header Host $host;
      proxy_set_header X-Real-IP $remote_addr;
      proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
      proxy_set_header X-Forwarded-Host $server_name;
    }
    # Proxy connections to the application servers
    location / {
      proxy_pass http://app_servers;
      proxy_redirect off;
      proxy_set_header Host $host;
      proxy_set_header X-Real-IP $remote_addr;
      proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
      proxy_set_header X-Forwarded-Host $server_name;
    }
  }
}
```

保存后退出。

开始构建：

进入docker-compose.yml所在的文件目录下

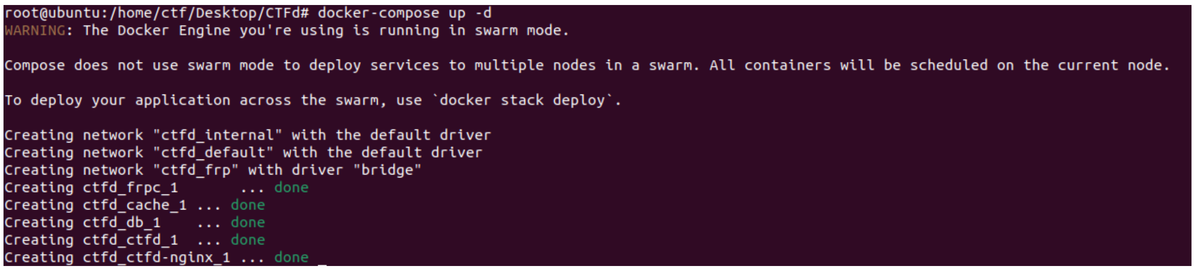
```
docker-compose up -d
```



```
root@ubuntu:/home/ctf/Desktop/CTFd# docker-compose up -d
WARNING: The Docker Engine you're using is running in swarm mode.

Compose does not use swarm mode to deploy services to multiple nodes in a swarm. All containers will be scheduled on the current node.

To deploy your application across the swarm, use `docker stack deploy`.

Creating network "ctfd_internal" with the default driver
Creating network "ctfd_default" with the default driver
Creating network "ctfd_frp" with driver "bridge"
Creating network "ctfd_frp-containers" with driver "overlay"
Pulling db (mariadb:10.4)...
```

耐心等待。。。

完成后应该是这个样子

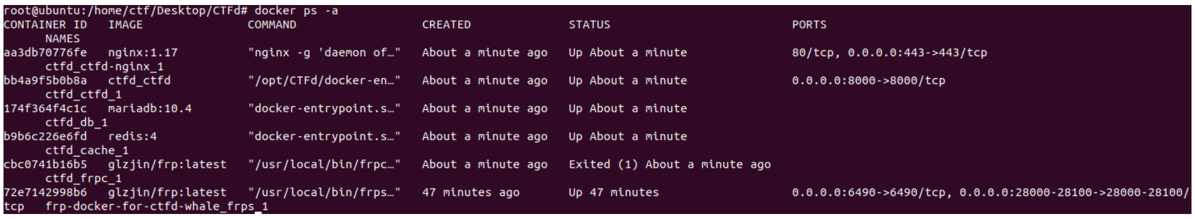


```
root@ubuntu:/home/ctf/Desktop/CTFd# docker-compose up -d
WARNING: The Docker Engine you're using is running in swarm mode.

Compose does not use swarm mode to deploy services to multiple nodes in a swarm. All containers will be scheduled on the current node.

To deploy your application across the swarm, use `docker stack deploy`.

Creating network "ctfd_internal" with the default driver
Creating network "ctfd_default" with the default driver
Creating network "ctfd_frp" with driver "bridge"
Creating ctfd_frpc_1       ... done
Creating ctfd_cache_1 ... done
Creating ctfd_db_1    ... done
Creating ctfd_ctfd_1  ... done
Creating ctfd_ctfd-nginx_1 ... done
```

配置网络：

我们先看一下现在的容器状态：

```
docker ps -a
```

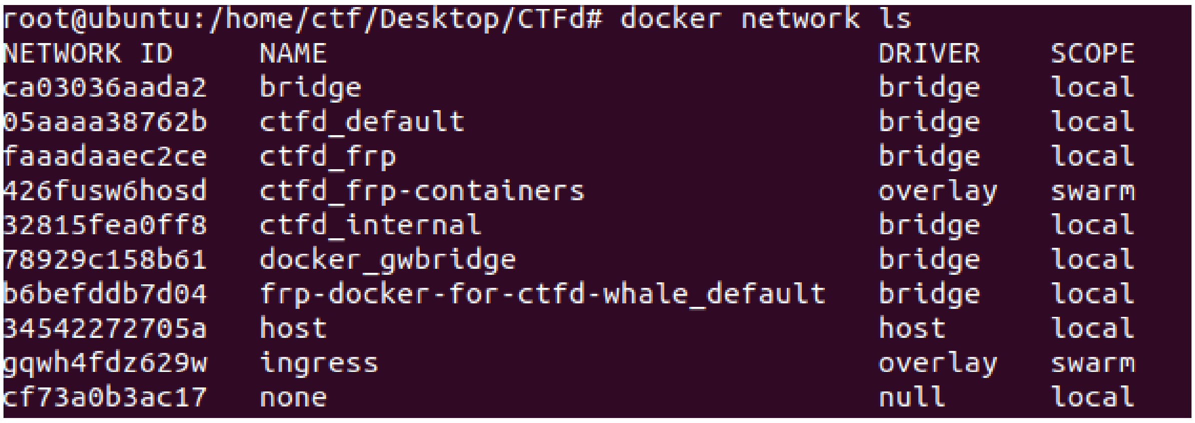


```
root@ubuntu:/home/ctf/Desktop/CTFd# docker ps -a
CONTAINER ID   IMAGE               COMMAND                  CREATED              STATUS                         PORTS
      NAMES
aa3db70776fe   nginx:1.17          "nginx -g 'daemon of…"   About a minute ago   Up About a minute              80/tcp, 0.0.0.0:443->443/tcp
      ctfd_ctfd-nginx_1
bb4a9f5b0b8a   ctfd_ctfd           "/opt/CTFd/docker-en…"   About a minute ago   Up About a minute              0.0.0.0:8000->8000/tcp
      ctfd_ctfd_1
174f364f4c1c   mariadb:10.4        "docker-entrypoint.s…"   About a minute ago   Up About a minute
      ctfd_db_1
b9b6c226e6fd   redis:4             "docker-entrypoint.s…"   About a minute ago   Up About a minute
      ctfd_cache_1
cbc0741b16b5   glzjin/frp:latest   "/usr/local/bin/frpc…"   About a minute ago   Exited (1) About a minute ago
      ctfd_frpc_1
72e7142998b6   glzjin/frp:latest   "/usr/local/bin/frps…"   47 minutes ago       Up 47 minutes                  0.0.0.0:6490->6490/tcp, 0.0.0.0:28000-28100->28000-28100/
tcp    frp-docker-for-ctfd-whale_frps_1
```

可以看到**ctfd_frpc_1**这个容器的状态是退出状态。

我们再看一下现在docker的网络：

```
docker network ls
```



```
root@ubuntu:/home/ctf/Desktop/CTFd# docker network ls
NETWORK ID     NAME                               DRIVER    SCOPE
ca03036aada2   bridge                             bridge    local
05aaaa38762b   ctfd_default                       bridge    local
faaadaaec2ce   ctfd_frp                           bridge    local
426fusw6hosd   ctfd_frp-containers                overlay   swarm
32815fea0ff8   ctfd_internal                      bridge    local
78929c158b61   docker_gwbridge                    bridge    local
b6befddb7d04   frp-docker-for-ctfd-whale_default  bridge    local
34542272705a   host                               host      local
gqwh4fdz629w   ingress                            overlay   swarm
cf73a0b3ac17   none                               null      local
```

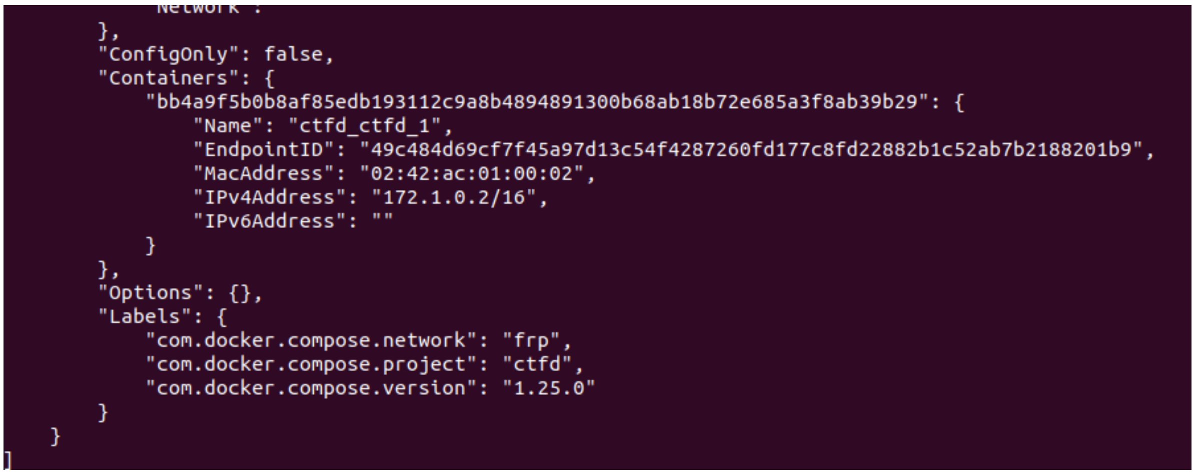
一会我们需要将ctfd_frpc_1，frp-docker-for-ctfd-whale_frps_1，ctfd_ctfd_1这三个容器加入到ctfd_frp网络中

并且这三个容器的IP如下：

- ctfd_ctfd_1：172.1.0.2
- ctfd_frpc_1：172.1.0.3
- frp-docker-for-ctfd-whale_frps_1：172.1.0.4

查看一下ctfd_frp网络

```
docker network inspect ctfd_frp
```



```
        },
        "ConfigOnly": false,
        "Containers": {
            "bb4a9f5b0b8af85edb193112c9a8b4894891300b68ab18b72e685a3f8ab39b29": {
                "Name": "ctfd_ctfd_1",
                "EndpointID": "49c484d69cf7f45a97d13c54f4287260fd177c8fd22882b1c52ab7b2188201b9",
                "MacAddress": "02:42:ac:01:00:02",
                "IPv4Address": "172.1.0.2/16",
                "IPv6Address": ""
            }
        },
        "Options": {},
        "Labels": {
            "com.docker.compose.network": "frp",
            "com.docker.compose.project": "ctfd",
            "com.docker.compose.version": "1.25.0"
        }
    }
]
```

这个时候只有ctfd_ctfd_1这个容器是在ctfd_frp网络里的。

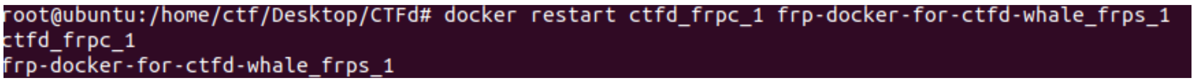
注意要指定IP：

```
docker network connect --ip 172.1.0.3 ctfd_frp ctfd_frpc_1
```

```
docker network connect --ip 172.1.0.4 ctfd_frp frp-docker-for-ctfd-whale_frps_1
```

然后重启一下这两个容器：

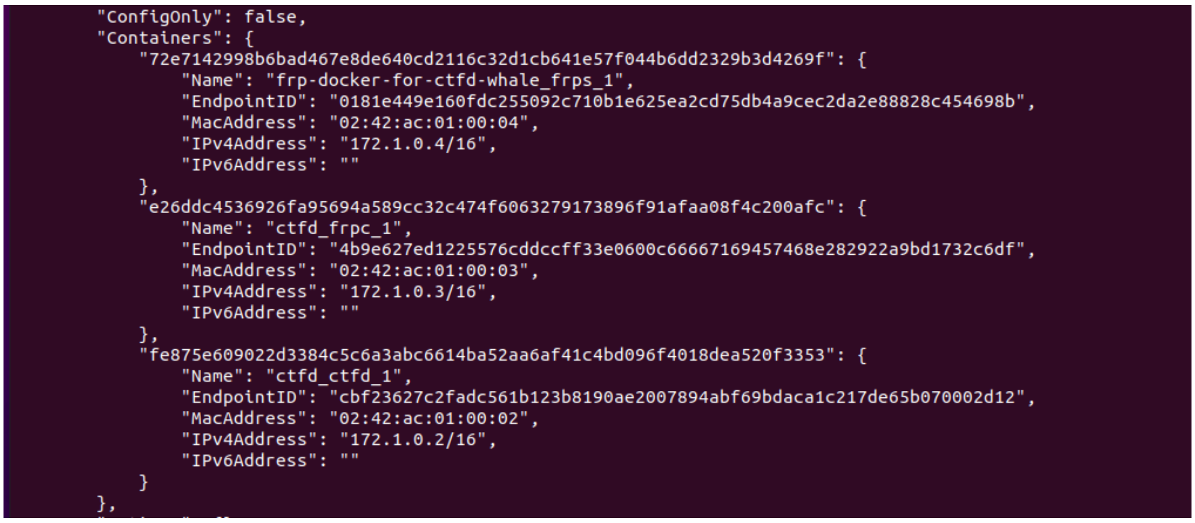
```
docker restart ctfd_frpc_1 frp-docker-for-ctfd-whale_frps_1
```



```
root@ubuntu:/home/ctf/Desktop/CTFd# docker restart ctfd_frpc_1 frp-docker-for-ctfd-whale_frps_1
ctfd_frpc_1
frp-docker-for-ctfd-whale_frps_1
```

重启完成之后再查看一下ctfd_frp网络

```
docker network inspect ctfd_frp
```

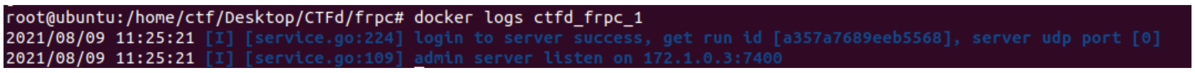


```
        "ConfigOnly": false,
        "Containers": {
            "72e7142998b6bad467e8de640cd2116c32d1cb641e57f044b6dd2329b3d4269f": {
                "Name": "frp-docker-for-ctfd-whale_frps_1",
                "EndpointID": "0181e449e160fdc255092c710b1e625ea2cd75db4a9cec2da2e88828c454698b",
                "MacAddress": "02:42:ac:01:00:04",
                "IPv4Address": "172.1.0.4/16",
                "IPv6Address": ""
            },
            "e26ddc4536926fa95694a589cc32c474f6063279173896f91afaa08f4c200afc": {
                "Name": "ctfd_frpc_1",
                "EndpointID": "4b9e627ed1225576cddccff33e0600c66667169457468e282922a9bd1732c6df",
                "MacAddress": "02:42:ac:01:00:03",
                "IPv4Address": "172.1.0.3/16",
                "IPv6Address": ""
            },
            "fe875e609022d3384c5c6a3abc6614ba52aa6af41c4bd096f4018dea520f3353": {
                "Name": "ctfd_ctfd_1",
                "EndpointID": "cbf23627c2fadc561b123b8190ae2007894abf69bdaca1c217de65b070002d12",
                "MacAddress": "02:42:ac:01:00:02",
                "IPv4Address": "172.1.0.2/16",
                "IPv6Address": ""
            }
        },
```

可以看到，这个时候，ctfd_frpc_1，frp-docker-for-ctfd-whale_frps_1，ctfd_ctfd_1这三个容器已经加入到ctfd_frp网络中

查看ctfd_frpc_1容器日志：

```
docker logs ctfd_frpc_1
```



```
root@ubuntu:/home/ctf/Desktop/CTFd/frpc# docker logs ctfd_frpc_1
2021/08/09 11:25:21 [I] [service.go:224] login to server success, get run id [a357a7689eeb5568], server udp port [0]
2021/08/09 11:25:21 [I] [service.go:109] admin server listen on 172.1.0.3:7400
```

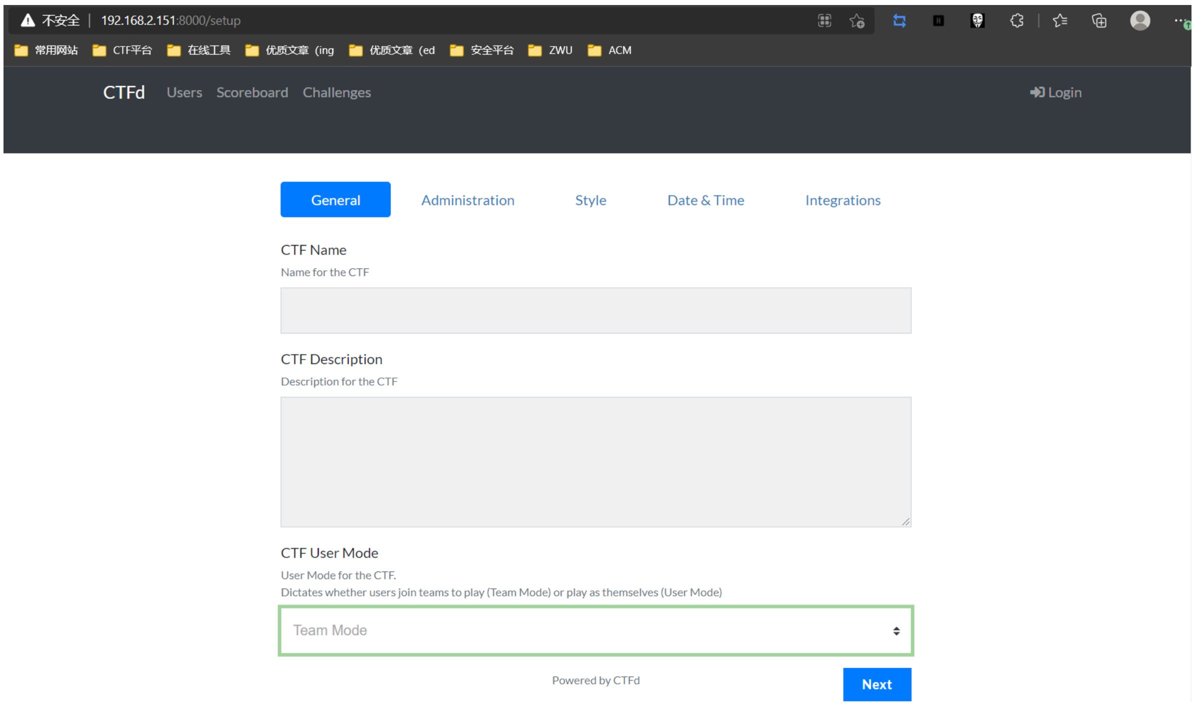
这样就算已经配置好了。

接下来就是ctfd-whale配置。

ctfd-whale配置:

在浏览器中访问**IP:8000**

我这台机器就是http://192.168.2.151:8000



192.168.2.151:8000/setup

CTFd Users Scoreboard Challenges Login

General Administration Style Date & Time Integrations

CTF Name
Name for the CTF

CTF Description
Description for the CTF

CTF User Mode
User Mode for the CTF.
Dictates whether users join teams to play (Team Mode) or play as themselves (User Mode)

Team Mode

Powered by CTFd

Next

这一块就是ctfd的前置配置，按照自己的需求配置一下就好了

网站初步配置完成之后，开始配置ctfd-whale插件。

设置方面，就按照赵总的配置就好了，我这边参考error师傅的。

属性	配置
Docker API URL	unix://var/run/docker.sock
Frp API IP	frpc的ip配置
Frp API Port	frpc的端口配置
Frp Http Domain Suffix	Docker API URL to connect（可填None）
Frp Http Port	80
Frp Direct IP Address	你的公网ip，本机即为127.0.0.1
Frp Direct Minimum Port	与之前frps最小端口呼应
Frp Direct Minimum Port	与之前frps最大端口呼应
Max Container Count	不超过最大-最小
Max Renewal Times	最大实例延时次数
Frp config template	填入frps的配置，只需填[common]
Docker Auto Connect Containers	ctfd_frpc_1
Docker Dns Setting	可填机器内DNS，没有可填个外网DNS
Docker Swarm Nodes	linux-1 与前面swarm集群呼应
Docker Multi-Container Network Subnet	内网题大子网ip配置/CIDR
Docker Multi-Container Network Subnet New Prefix	每个内网题实例的CIDR
Docker Container Timeout	单位为秒

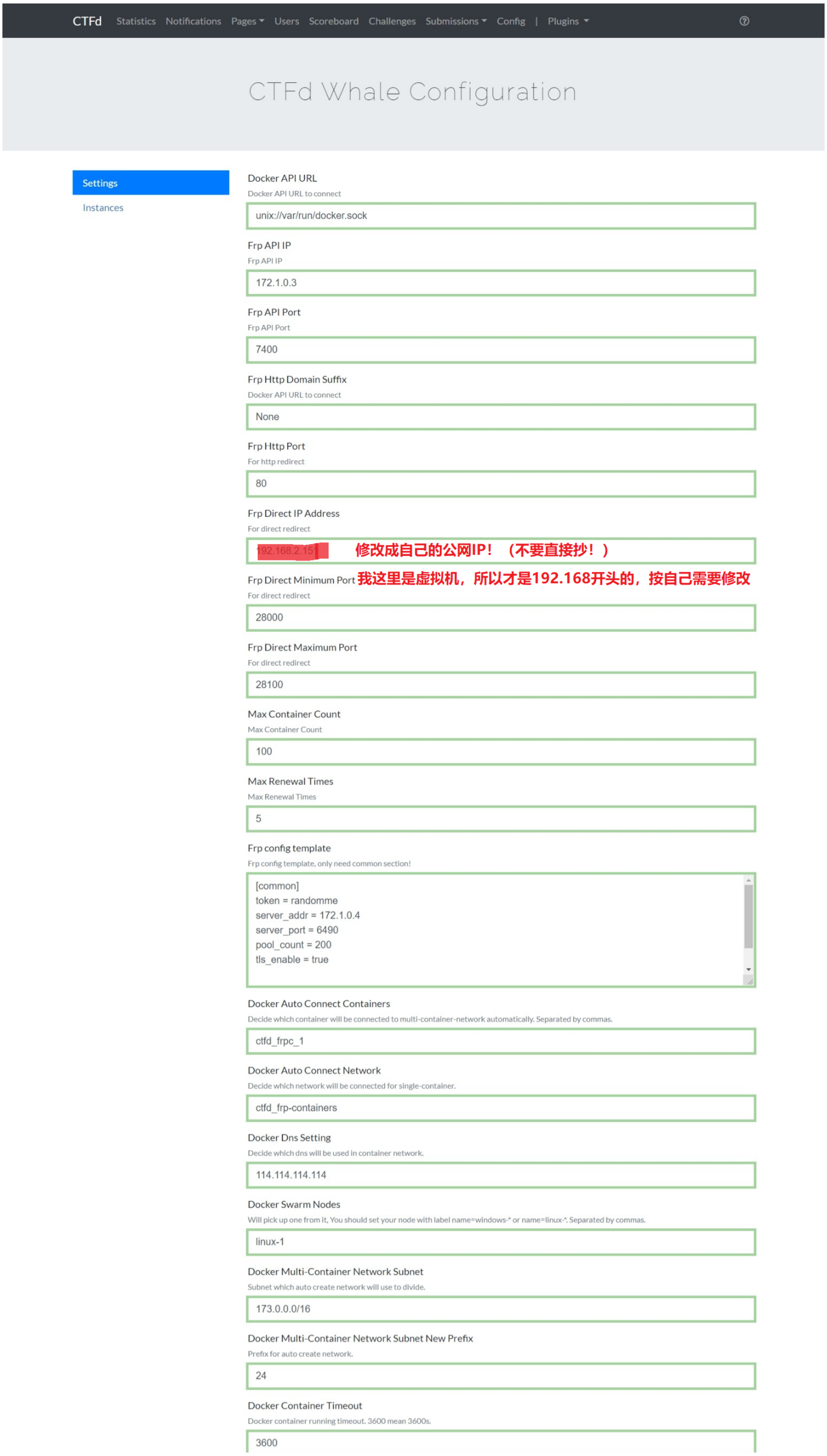
其中Frp config template配置内容如下：

```
[common]
token = randomme
server_addr = 172.1.0.4
server_port = 6490
pool_count = 200
tls_enable = true

admin_addr = 172.1.0.3
admin_port = 7400
```

其他的按照下面这张图片配置就好

注意：

Frp Direct IP Address这个一定要修改成自己的IP，如果是云服务器就输入公网IP，如果是虚拟机那就输入虚拟机的IP。



CTFd Whale Configuration

Docker API URL
Docker API URL to connect
unix://var/run/docker.sock

Frp API IP
Frp API IP
172.1.0.3

Frp API Port
Frp API Port
7400

Frp Http Domain Suffix
Docker API URL to connect
None

Frp Http Port
For http redirect
80

Frp Direct IP Address
For direct redirect
[redacted] 修改成自己的公网IP！（不要直接抄！）

Frp Direct Minimum Port 我这里是虚拟机，所以才是192.168开头的，按自己需要修改
For direct redirect
28000

Frp Direct Maximum Port
For direct redirect
28100

Max Container Count
Max Container Count
100

Max Renewal Times
Max Renewal Times
5

Frp config template
Frp config template, only need common section!
```
[common]
token = randomme
server_addr = 172.1.0.4
server_port = 6490
pool_count = 200
tls_enable = true
```

Docker Auto Connect Containers
Decide which container will be connected to multi-container-network automatically. Separated by commas.
ctfd_frpc_1

Docker Auto Connect Network
Decide which network will be connected for single-container.
ctfd_frp-containers

Docker Dns Setting
Decide which dns will be used in container network.
114.114.114.114

Docker Swarm Nodes
Will pick up one from it, You should set your node with label name=windows-* or name=linux-*. Separated by commas.
linux-1

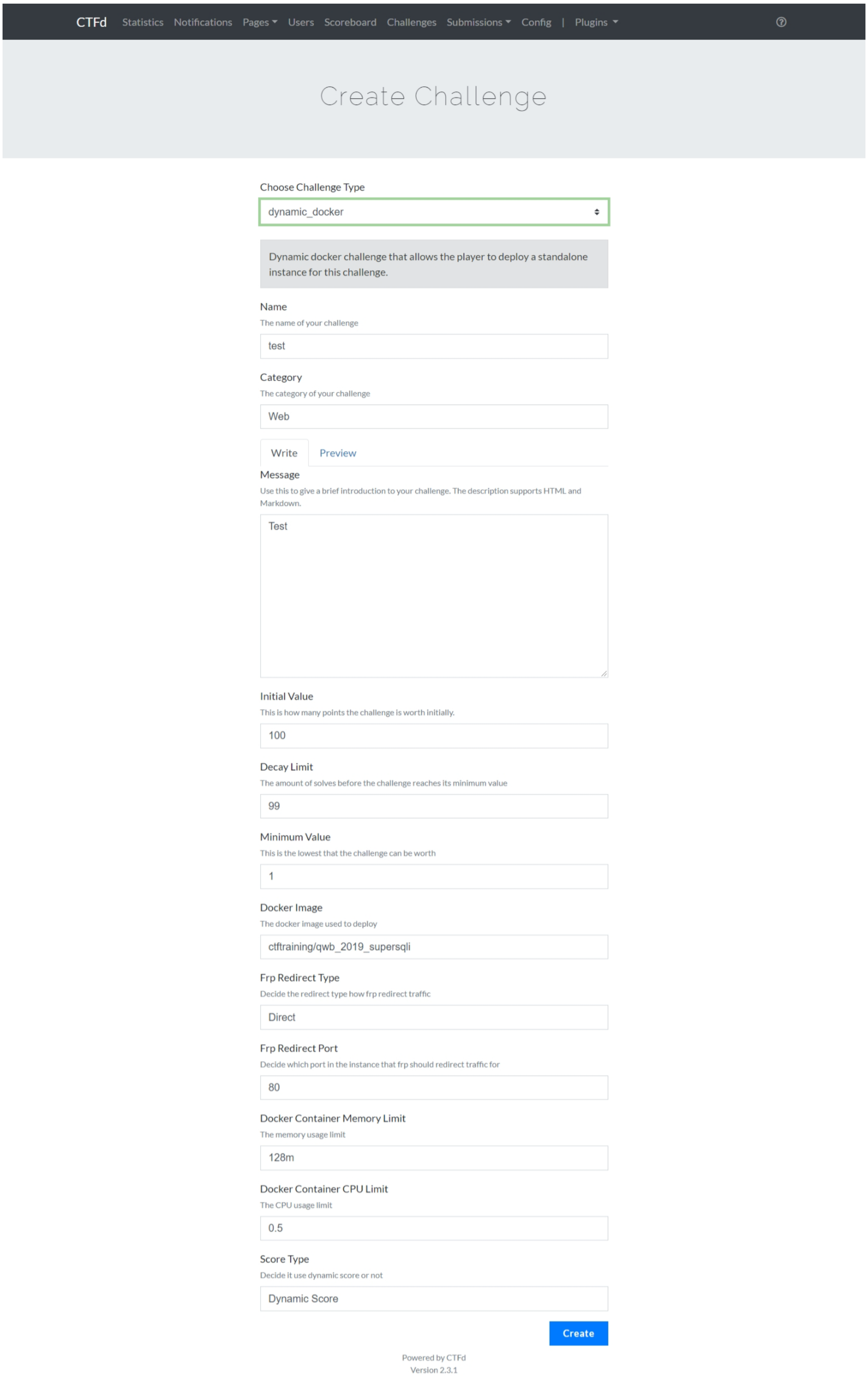
Docker Multi-Container Network Subnet
Subnet which auto create network will use to divide.
173.0.0.0/16

Docker Multi-Container Network Subnet New Prefix
Prefix for auto create network.
24

Docker Container Timeout
Docker container running timeout. 3600 mean 3600s.
3600

添加一道题目：

添加的题目如下：



Create Challenge

Choose Challenge Type

dynamic_docker

Dynamic docker challenge that allows the player to deploy a standalone instance for this challenge.

Name

The name of your challenge

test

Category

The category of your challenge

Web

Message

Use this to give a brief introduction to your challenge. The description supports HTML and Markdown.

Test

Initial Value

This is how many points the challenge is worth initially.

100

Decay Limit

The amount of solves before the challenge reaches its minimum value

99

Minimum Value

This is the lowest that the challenge can be worth

1

Docker Image

The docker image used to deploy

ctftraining/qwb_2019_supersqli

Frp Redirect Type

Decide the redirect type how frp redirect traffic

Direct

Frp Redirect Port

Decide which port in the instance that frp should redirect traffic for

80

Docker Container Memory Limit

The memory usage limit

128m

Docker Container CPU Limit

The CPU usage limit

0.5

Score Type

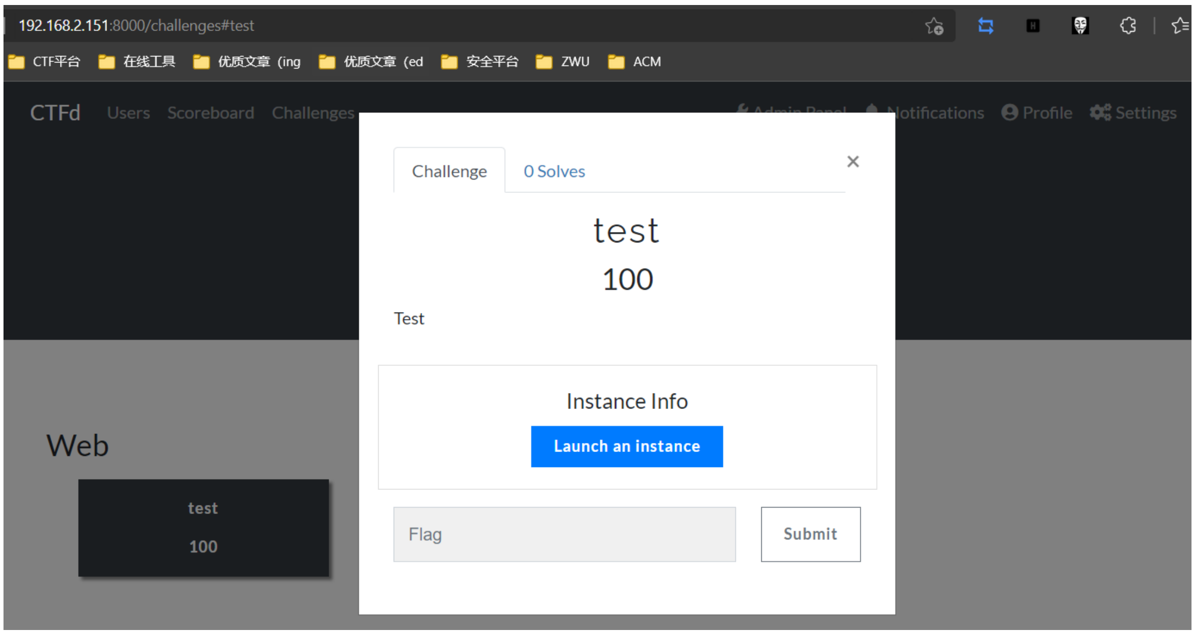
Decide it use dynamic score or not

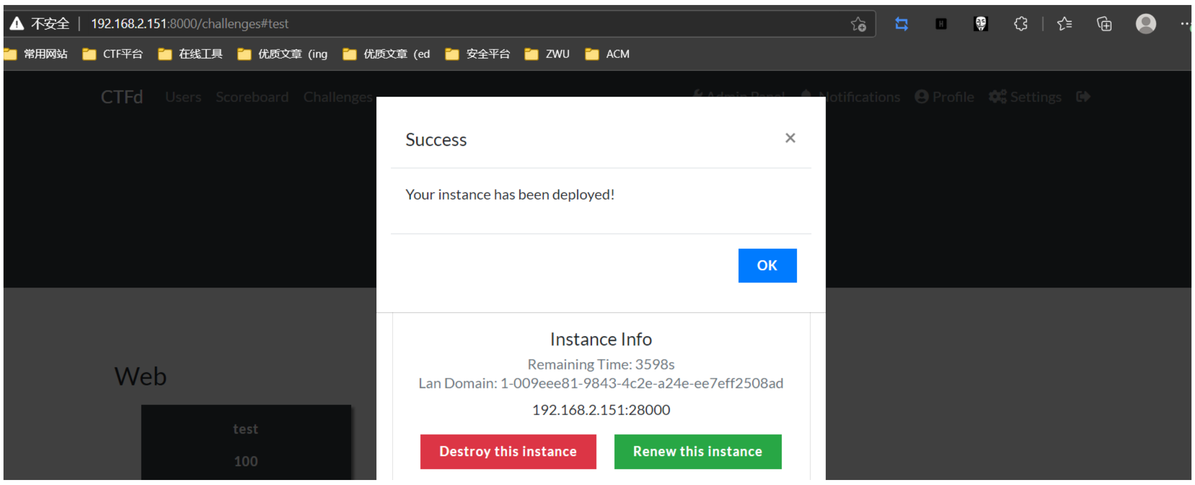
Dynamic Score

Powered by CTFd
Version 2.3.1

其中Docker image里就输入题目的tag就好了

测试开启题目：



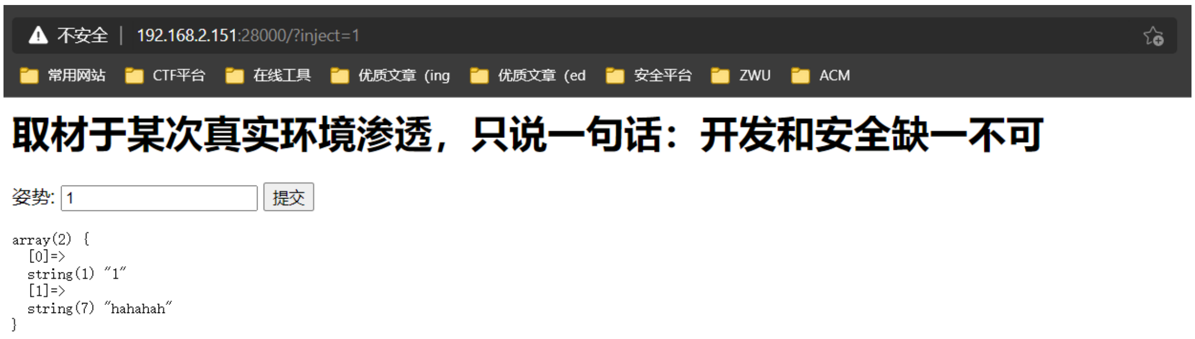


Your instance has been deployed!

Remaining Time: 3598s
Lan Domain: 1-009eee81-9843-4c2e-a24e-ee7eff2508ad
192.168.2.151:28000

可以看到，已经正确分配IP了

第一次获取题目要等一会，因为要拉题目镜像，等一会后就可以访问题目了。

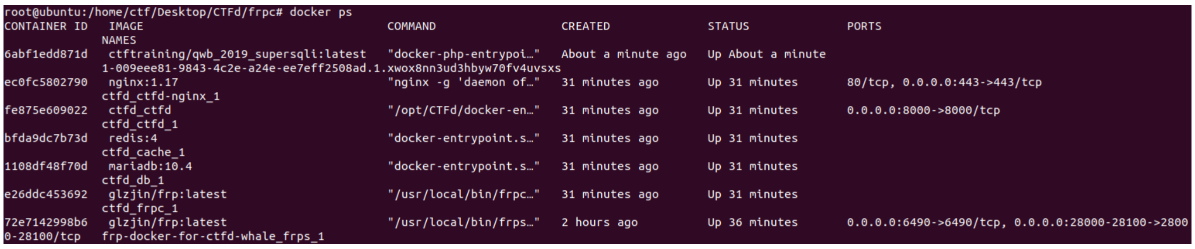


取材于某次真实环境渗透，只说一句话：开发和安全缺一不可

姿势: 1

```
array(2) {
  [0]=>
  string(1) "1"
  [1]=>
  string(7) "hahahah"
}
```

在Docker里也可以看到相应的题目容器。



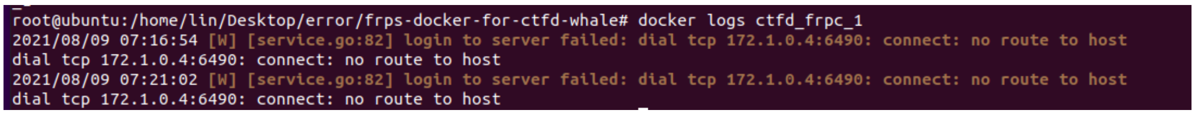
```
root@ubuntu:/home/ctf/Desktop/CTFd/frpc# docker ps
CONTAINER ID   IMAGE                                COMMAND                  CREATED              STATUS              PORTS
               NAMES
6abf1edd871d   ctftraining/qwb_2019_supersqli:latest   "docker-php-entrypoi…"   About a minute ago   Up About a minute
               1-009eee81-9843-4c2e-a24e-ee7eff2508ad.1.xwox8nn3ud3hbyw70fv4uvsxs
ec0fc5802790   nginx:1.17                           "nginx -g 'daemon of…"   31 minutes ago       Up 31 minutes       80/tcp, 0.0.0.0:443->443/tcp
               ctfd_ctfd-nginx_1
fe875e609022   ctfd_ctfd                            "/opt/CTFd/docker-en…"   31 minutes ago       Up 31 minutes       0.0.0.0:8000->8000/tcp
               ctfd_ctfd_1
bfda9dc7b73d   redis:4                              "docker-entrypoint.s…"   31 minutes ago       Up 31 minutes
               ctfd_cache_1
1108df48f70d   mariadb:10.4                         "docker-entrypoint.s…"   31 minutes ago       Up 31 minutes
               ctfd_db_1
e26ddc453692   glzjin/frp:latest                    "/usr/local/bin/frpc…"   31 minutes ago       Up 31 minutes
               ctfd_frpc_1
72e7142998b6   glzjin/frp:latest                    "/usr/local/bin/frps…"   2 hours ago          Up 36 minutes       0.0.0.0:6490->6490/tcp, 0.0.0.0:28000-28100->2800
0-28100/tcp    frp-docker-for-ctfd-whale_frps_1
```

问题报错以及踩坑：

frpc日志报错：

报错：

如果日志像下方这样，那么就是网络没有配置好



```
root@ubuntu:/home/lin/Desktop/error/frps-docker-for-ctfd-whale# docker logs ctfd_frpc_1
2021/08/09 07:16:54 [W] [service.go:82] login to server failed: dial tcp 172.1.0.4:6490: connect: no route to host
dial tcp 172.1.0.4:6490: connect: no route to host
2021/08/09 07:21:02 [W] [service.go:82] login to server failed: dial tcp 172.1.0.4:6490: connect: no route to host
dial tcp 172.1.0.4:6490: connect: no route to host
```

解决方案：

仔细配置网络，网络配置的要求如下：

- 将ctfd_frpc_1，frp-docker-for-ctfd-whale_frps_1，ctfd_ctfd_1这三个容器加入到ctfd_frp网络中
- 这三个容器对应的IP如下：
  - ctfd_ctfd_1：172.1.0.2
  - ctfd_frpc_1：172.1.0.3
  - frp-docker-for-ctfd-whale_frps_1：172.1.0.4

活用docker的日志功能：

```
docker logs <容器ID或名称>
```

活用docker的网络配置功能：

查看网络

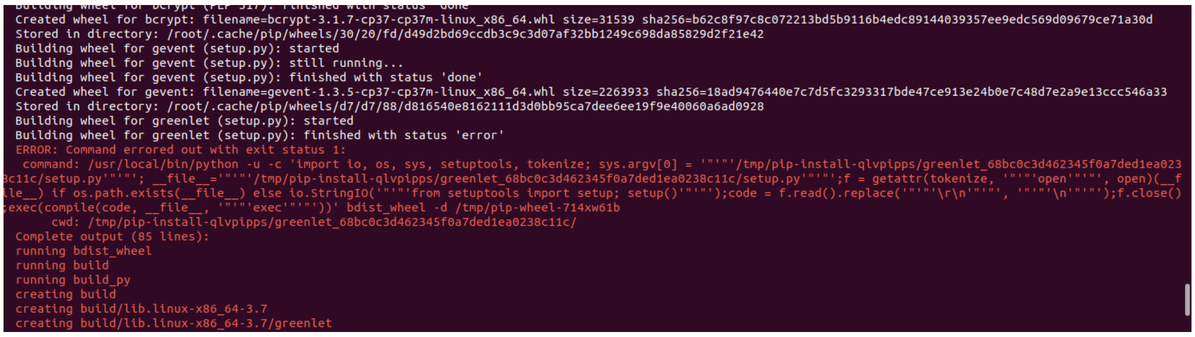
```
docker network ls
```

给容器分配指定IP

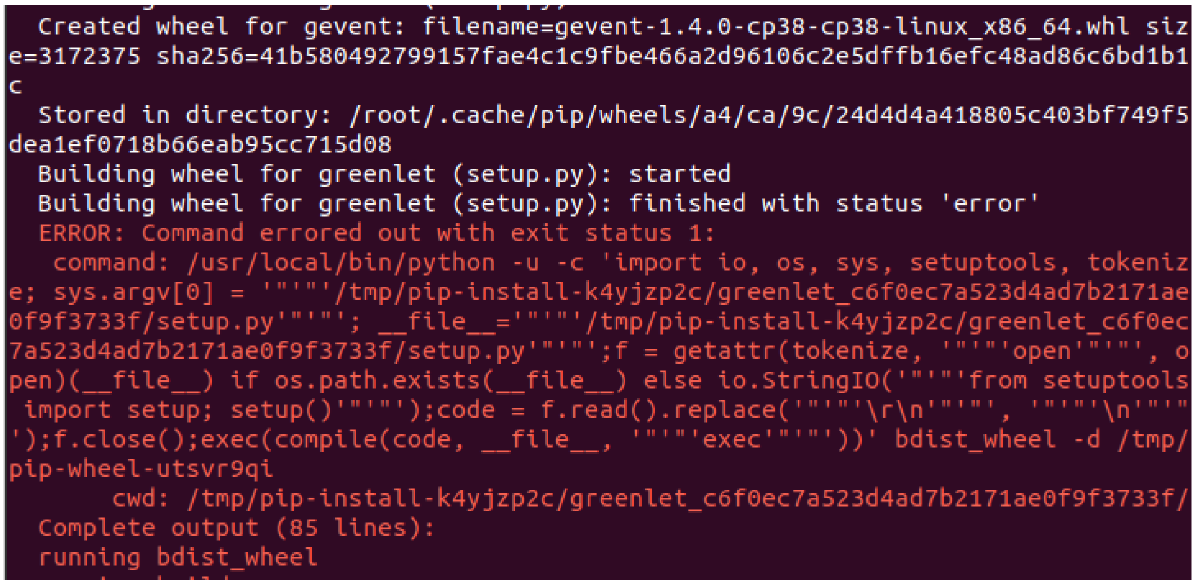
```
docker network connect --ip 172.1.0.4 <网络名称> <容器ID或名称>
```

Dockerfile的gevent报错：

报错：



```
 Building wheel for bcrypt (PEP 517): finished with status 'done'
 Created wheel for bcrypt: filename=bcrypt-3.1.7-cp37-cp37m-linux_x86_64.whl size=31539 sha256=b62c8f97c8c072213bd5b9116b4edc89144039357ee9edc569d09679ce71a30d
 Stored in directory: /root/.cache/pip/wheels/30/20/fd/d49d2bd69ccdb3c9c3d07af32bb1249c698da85829d2f21e42
 Building wheel for gevent (setup.py): started
 Building wheel for gevent (setup.py): still running...
 Building wheel for gevent (setup.py): finished with status 'done'
 Created wheel for gevent: filename=gevent-1.3.5-cp37-cp37m-linux_x86_64.whl size=2263933 sha256=18ad9476440e7c7d5fc3293317bde47ce913e24b0e7c48d7e2a9e13ccc546a33
 Stored in directory: /root/.cache/pip/wheels/d7/d7/88/d816540e8162111d3d0bb95ca7dee6ee19f9e40060a6ad0928
 Building wheel for greenlet (setup.py): started
 Building wheel for greenlet (setup.py): finished with status 'error'
 ERROR: Command errored out with exit status 1:
  command: /usr/local/bin/python -u -c 'import io, os, sys, setuptools, tokenize; sys.argv[0] = '"'"'/tmp/pip-install-qlvpipps/greenlet_68bc0c3d462345f0a7ded1ea023
3c11c/setup.py'"'"'; __file__='"'"'/tmp/pip-install-qlvpipps/greenlet_68bc0c3d462345f0a7ded1ea0238c11c/setup.py'"'"';f = getattr(tokenize, '"'"'open'"'"', open)(__f
ile__) if os.path.exists(__file__) else io.StringIO('"'"'from setuptools import setup; setup()'"'"');code = f.read().replace('"'"'\r\n'"'"', '"'"'\n'"'"');f.close()
;exec(compile(code, __file__, '"'"'exec'"'"'))' bdist_wheel -d /tmp/pip-wheel-714xw61b
      cwd: /tmp/pip-install-qlvpipps/greenlet_68bc0c3d462345f0a7ded1ea0238c11c/
 Complete output (85 lines):
 running bdist_wheel
 running build
 running build_py
 creating build
 creating build/lib.linux-x86_64-3.7
 creating build/lib.linux-x86_64-3.7/greenlet
```

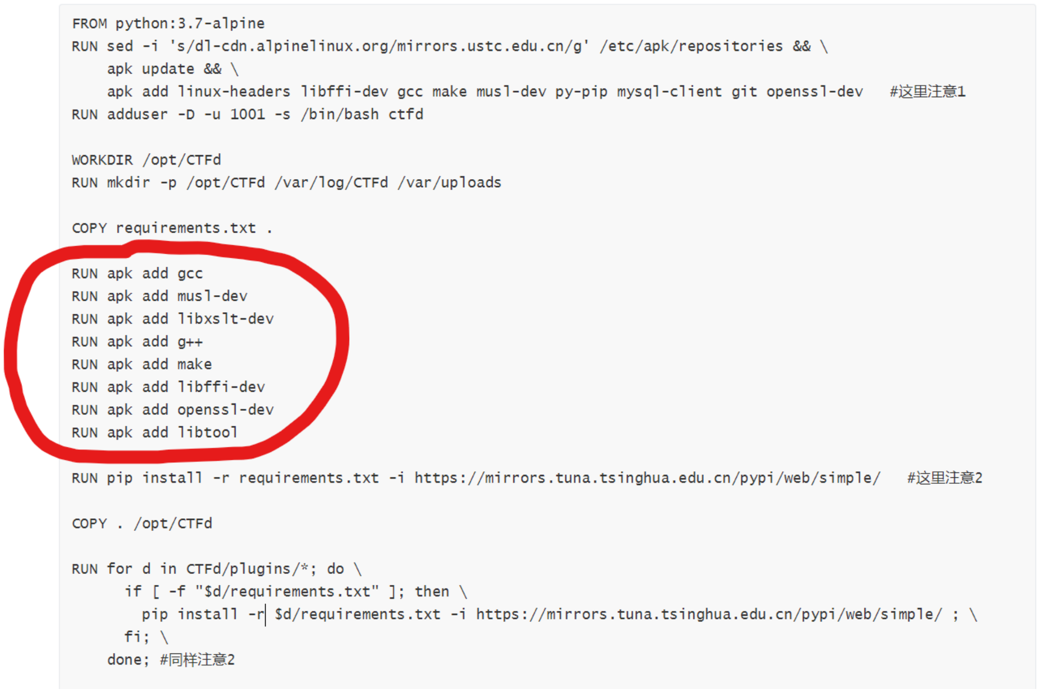


```
  Created wheel for gevent: filename=gevent-1.4.0-cp38-cp38-linux_x86_64.whl siz
e=3172375 sha256=41b580492799157fae4c1c9fbe466a2d96106c2e5dffb16efc48ad86c6bd1b1
c
  Stored in directory: /root/.cache/pip/wheels/a4/ca/9c/24d4d4a418805c403bf749f5
dea1ef0718b66eab95cc715d08
  Building wheel for greenlet (setup.py): started
  Building wheel for greenlet (setup.py): finished with status 'error'
  ERROR: Command errored out with exit status 1:
   command: /usr/local/bin/python -u -c 'import io, os, sys, setuptools, tokeniz
e; sys.argv[0] = '"'"'/tmp/pip-install-k4yjzp2c/greenlet_c6f0ec7a523d4ad7b2171ae
0f9f3733f/setup.py'"'"'; __file__='"'"'/tmp/pip-install-k4yjzp2c/greenlet_c6f0ec
7a523d4ad7b2171ae0f9f3733f/setup.py'"'"';f = getattr(tokenize, '"'"'open'"'"', o
pen)(__file__) if os.path.exists(__file__) else io.StringIO('"'"'from setuptools
 import setup; setup()'"'"');code = f.read().replace('"'"'\r\n'"'"', '"'"'\n'"'"
');f.close();exec(compile(code, __file__, '"'"'exec'"'"'))' bdist_wheel -d /tmp/
pip-wheel-utsvr9qi
       cwd: /tmp/pip-install-k4yjzp2c/greenlet_c6f0ec7a523d4ad7b2171ae0f9f3733f/
  Complete output (85 lines):
  running bdist_wheel
```

解决方案：

其他Dockerfile在构建的时候可能会出现gevent构建不成功的问题，有以下几种解决办法：

1. 将镜像源替换为清华源
2. 替换gevent版本
3. 将Dockerfile文件中添加以下内容

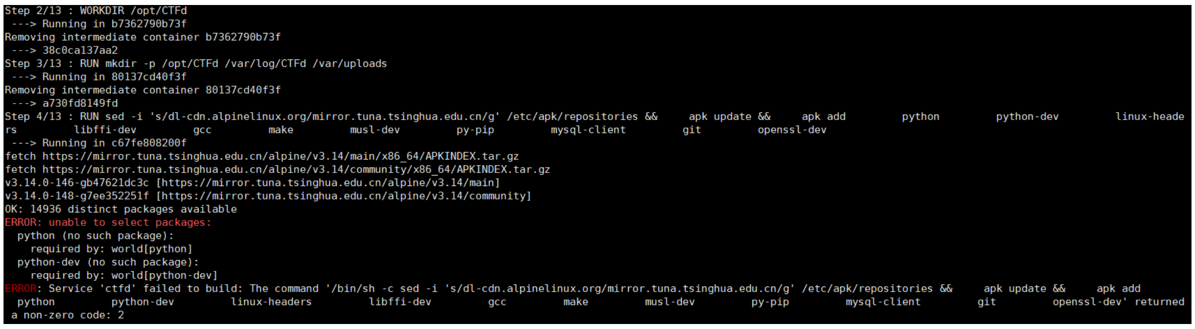


```
FROM python:3.7-alpine
RUN sed -i 's/dl-cdn.alpinelinux.org/mirrors.ustc.edu.cn/g' /etc/apk/repositories && \
    apk update && \
    apk add linux-headers libffi-dev gcc make musl-dev py-pip mysql-client git openssl-dev   #这里注意1
RUN adduser -D -u 1001 -s /bin/bash ctfd

WORKDIR /opt/CTFd
RUN mkdir -p /opt/CTFd /var/log/CTFd /var/uploads

COPY requirements.txt .

RUN apk add gcc
RUN apk add musl-dev
RUN apk add libxslt-dev
RUN apk add g++
RUN apk add make
RUN apk add libffi-dev
RUN apk add openssl-dev
RUN apk add libtool

RUN pip install -r requirements.txt -i https://mirrors.tuna.tsinghua.edu.cn/pypi/web/simple/   #这里注意2

COPY . /opt/CTFd

RUN for d in CTFd/plugins/*; do \
      if [ -f "$d/requirements.txt" ]; then \
        pip install -r $d/requirements.txt -i https://mirrors.tuna.tsinghua.edu.cn/pypi/web/simple/ ; \
      fi; \
    done; #同样注意2
```

（这篇文章的搭建方法，已经把坑都踩了，所以前面的教程是没有问题的）

Dockerfile的world模块报错：

报错：



```
Step 2/13 : WORKDIR /opt/CTFd
 ---> Running in b7362790b73f
Removing intermediate container b7362790b73f
 ---> 38c0ca137aa2
Step 3/13 : RUN mkdir -p /opt/CTFd /var/log/CTFd /var/uploads
 ---> Running in 80137cd40f3f
Removing intermediate container 80137cd40f3f
 ---> a730fd8149fd
Step 4/13 : RUN sed -i 's/dl-cdn.alpinelinux.org/mirror.tuna.tsinghua.edu.cn/g' /etc/apk/repositories &&     apk update &&     apk add         python         python-dev         linux-heade
rs         libffi-dev         gcc         make         musl-dev         py-pip         mysql-client         git         openssl-dev
 ---> Running in c67fe808200f
fetch https://mirror.tuna.tsinghua.edu.cn/alpine/v3.14/main/x86_64/APKINDEX.tar.gz
fetch https://mirror.tuna.tsinghua.edu.cn/alpine/v3.14/community/x86_64/APKINDEX.tar.gz
v3.14.0-146-gb47621dc3c [https://mirror.tuna.tsinghua.edu.cn/alpine/v3.14/main]
v3.14.0-148-g7ee352251f [https://mirror.tuna.tsinghua.edu.cn/alpine/v3.14/community]
OK: 14936 distinct packages available
ERROR: unable to select packages:
  python (no such package):
    required by: world[python]
  python-dev (no such package):
    required by: world[python-dev]
ERROR: Service 'ctfd' failed to build: The command '/bin/sh -c sed -i 's/dl-cdn.alpinelinux.org/mirror.tuna.tsinghua.edu.cn/g' /etc/apk/repositories &&     apk update &&     apk add
  python         python-dev         linux-headers         libffi-dev         gcc         make         musl-dev         py-pip         mysql-client         git         openssl-dev' returned
 a non-zero code: 2
```

解决方案：

将Dockerfile中的python和python-dev删掉，或者修改成`python2`或`python3`，`python2-dev`或`python3-dev`都可以。

参考资料：

- CTFd/CTFd: CTFs as you need them (github.com)
- CTFD支持动态靶机的搭建笔记（docker：ctfd+ctf-whale）2020.10.17 - 灰信网（软件开发博客聚合）(freesion.com)
- glzjin/CTFd-Whale: A plugin for CTFd which allow your users to deploy a standalone instance for challenges. (github.com)
- CTFD支持动态靶机的搭建笔记（docker：ctfd+ctf-whale） | Err0r
- 手把手教你如何建立一个支持ctf动态独立靶机的靶场（ctfd+ctfd-whale）_fjh1997的博客-CSDN博客
- ctfd使用ctfd-whale动态靶机插件搭建靶场指南 | VaalaCat