Docker

Docker LXC VM VM , LXC -
LXC cgroups - snapshot image docker( ) AUFS
kernel patch host , dotcloud( ) kernel grsec patch .

- YAML
- Docker Compose
- Dockerfile
- [shell] docker deploy service script
- [shell] Docker 2
- docker
- Docker Container

YAML

YAML "YAML Ain't a Markup Language" YAML YAML "Yet Another M

YAML

YAML `.yml` `wzhz.xyz.yml`

- tab
- '#'

YAML

- mapping / hashes / dictionary
- sequence / list
- scalars

YAML

key: value *

key:{key1: value1, key2: value2, ...}

```
key:
    child-key: value
    child-key2: value2
```

key value

```
?
    - complexkey1
    - complexkey2
:
    - complexvalue1
    - complexvalue2
```

[complexkey1,complexkey2] [complexvalue1,complexvalue2]

YAML

-

```
- A
- B
- C
```

YAML

```
key: [value1, value2, ...]
```

```
-
 - A
 - B
 - C
```

```
companies:
    -
        id: 1
        name: company1
        price: 200W
    -
        id: 2
        name: company2
        price: 500W
```

companies id name price

(flow)

```
companies: [{id: 1,name: company1,price: 200W},{id: 2,name: company2,price: 500W}]
```

```
languages:
  - Ruby
  - Perl
  - Python
websites:
```

```
  YAML: yaml.org
  Ruby: ruby-lang.org
  Python: python.org
  Perl: use.perl.org
```

json

```
{
  languages: [ 'Ruby', 'Perl', 'Python'],
  websites: {
    YAML: 'yaml.org',
    Ruby: 'ruby-lang.org',
    Python: 'python.org',
    Perl: 'use.perl.org'
  }
}
```

- Null
-

```
boolean:
    - TRUE  #true,True
    - FALSE  #false False
float:
    - 3.14
    - 6.8523015e+5  #
int:
    - 123
    - 0b1010_0111_0100_1010_1110    #
null:
    nodeName: 'node'
    parent: ~  # ~ null
string:
    -
    - 'Hello world'  #
    - newline
      newline2    #
date:
    - 2018-02-17    #     ISO 8601    yyyy-MM-dd
datetime:
```

```
    -  2018-02-17T15:02:31+08:00    #   ISO 8601          T      +
```

`&` `*` :

```
defaults: &defaults
  adapter:  postgres
  host:     localhost

development:
  database: myapp_development
  <<: *defaults

test:
  database: myapp_test
  <<: *defaults
```

 :

```
defaults:
  adapter:  postgres
  host:     localhost

development:
  database: myapp_development
  adapter:  postgres
  host:     localhost

test:
  database: myapp_test
  adapter:  postgres
  host:     localhost
```

& defaults **<<** *

 :

```
- &showell Steve
- Clark
- Brian
- Oren
```

```
- *showell
```

JavaScript :

```
[ 'Steve', 'Clark', 'Brian', 'Oren', 'Steve' ]
```

YAML

YAML

yaml

Docker Compose

Compose

Compose Docker Compose YML YML

`YAML`

Compose

- Dockerfile
- docker-compose.yml
- docker-compose up

`docker-compose.yml`

```
# yaml
version: '3'
services:
  web:
    build: .
    ports:
   - "5000:5000"
    volumes:
   - .:/code
    - logvolume01:/var/log
    links:
   - redis
  redis:
    image: redis
volumes:
  logvolume01: {}
```

Compose

Linux

Linux Github compose-releases

`Docker Compose`

```
$ sudo curl -L "https://github.com/docker/compose/releases/download/1.24.1/docker-compose-$(uname -s)-$(uname -m)" -o /usr/local/bin/docker-compose
```

Compose 1.24.1

```
$ sudo chmod +x /usr/local/bin/docker-compose
```

```
$ sudo ln -s /usr/local/bin/docker-compose /usr/bin/docker-compose
```

```
$ docker-compose --version
cker-compose version 1.24.1, build 4667896b
```

`alpine` `py-pip` `python-dev` `libffi-dev` `openssl-dev` gc `libc-dev` `make`

macOS

Mac `Docker` `Docker Toolbox` `Compose` `Docker` `Compose` `Docker` MacOS Docker

windows PC

Windows `Docker` `Docker Toolbox` `Compose` `Docker` Wi `Compose` `Docker` Windows Docker

1

```
$ mkdir composetest
$ cd composetest
```

app.py

`composetest/app.py`

```
import time

import redis
from flask import Flask
```

```
app = Flask(__name__)
cache = redis.Redis(host='redis', port=6379)



def get_hit_count():
    retries = 5
    while True:
        try:
            return cache.incr('hits')
        except redis.exceptions.ConnectionError as exc:
            if retries == 0:
                raise exc
            retries -= 1
            time.sleep(0.5)



@app.route('/')
def hello():
    count = get_hit_count()
    return 'Hello World! I have been seen {} times.\n'.format(count)
```

redis redis 6379

composetest requirements.txt

```
flask
redis
```

2 Dockerfile

composetest Dockerfile

```
FROM python:3.7-alpine
WORKDIR /code
ENV FLASK_APP app.py
ENV FLASK_RUN_HOST 0.0.0.0
RUN apk add --no-cache gcc musl-dev linux-headers
COPY requirements.txt requirements.txt
RUN pip install -r requirements.txt
COPY . .
```

```
CMD ["flask", "run"]
```

Dockerfile

- FROM python:3.7-alpine: Python 3.7
- WORKDIR /code: /code
- ```
  ENV FLASK_APP app.py
  ENV FLASK_RUN_HOST 0.0.0.0
  ```
  flask
- RUN apk add --no-cache gcc musl-dev linux-headers: gcc MarkupSafe SQLAlchemy P
- ```
  COPY requirements.txt requirements.txt
  RUN pip install -r requirements.txt
  ```
  requirements.txt Python
- COPY . .: .
- CMD ["flask", "run"]: flask run

3 docker-compose.yml

docker-compose.yml

docker-compose.yml

```
# yaml
version: '3'
services:
  web:
    build: .
    ports:
     - "5000:5000"
  redis:
    image: "redis:alpine"
```

Compose web redis

- web web Dockerfile 5000 Flask Web 5000
- redis redis Docker Hub Redis

4 Compose

```
```

```
docker-compose up
```

-d

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

yml

version

yml compose

build

webapp ./dir/Dockerfile

```
version: "3.7"
services:
  webapp:
    build: ./dir
```

Dockerfile args

```
version: "3.7"
services:
  webapp:
    build:
      context: ./dir
      dockerfile: Dockerfile-alternate
      args:
        buildno: 1
      labels:
        - "com.example.description=Accounting webapp"
        - "com.example.department=Finance"
        - "com.example.label-with-empty-value"
      target: prod
```

- context
- dockerfile Dockerfile
- args
- labels

- target

cap_add cap_drop

```
cap_add:
  - ALL #

cap_drop:
  - SYS_PTRACE #    ptrace
```

cgroup_parent

    cgroup

```
cgroup_parent: m-executor-abcd
```

command

```
command: ["bundle", "exec", "thin", "-p", "3000"]
```

container_name

```
container_name: my-web-container
```

depends_on

- docker-compose up db redis web
- docker-compose up SERVICE SERVICE docker-compose up web db redi
- docker-compose stop web db redis

```
version: "3.7"
services:
  web:
    build: .
    depends_on:
      - db
      - redis
  redis:
```

```
    image: redis
  db:
    image: postgres
```

web redis db

deploy

swarm

```
version: "3.7"
services:
  redis:
    image: redis:alpine
    deploy:
      mode replicated
      replicas: 6
      endpoint_mode: dnsrr
      labels:
        description: "This redis service label"
      resources:
        limits:
          cpus: '0.50'
          memory: 50M
        reservations:
          cpus: '0.25'
          memory: 20M
      restart_policy:
        condition: on-failure
        delay: 5s
        max_attempts: 3
        window: 120s
```

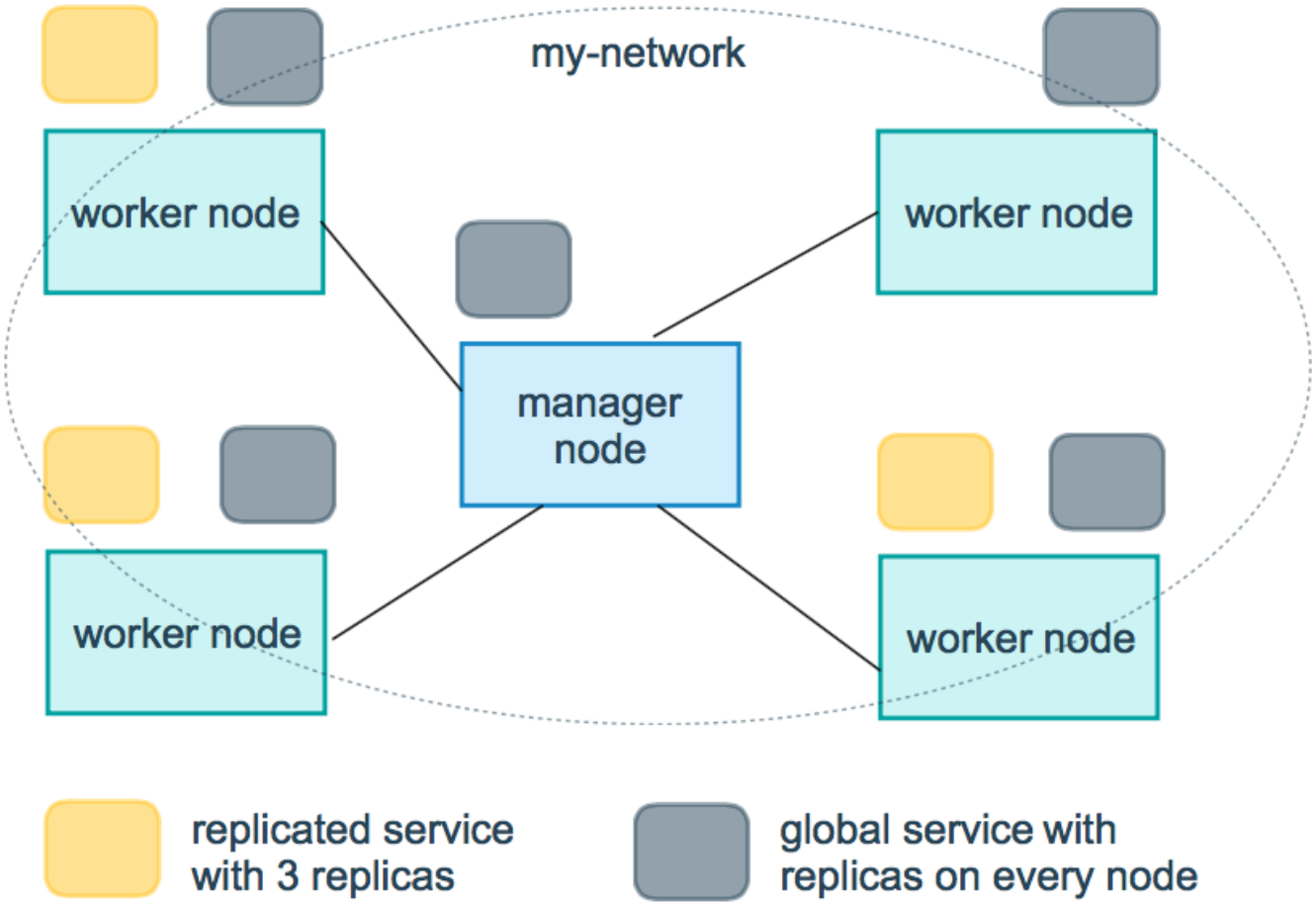
endpoint_mode

```
endpoint_mode: vip
# Docker           ip            ip
endpoint_mode: dnsrr
# DNS    DNSRR              ip       ip
```

labels labels deploy deploy labels

mode

- replicated
- global
- replicated global



replicas mode replicated

resources redis cpu

restart_policy

- condition none on-failure any any
- delay 0
- max_attempts
- window 0
- rollback_config
- parallelism 0
- delay 0s

- failure_action continue pause pause
- monitor (ns|us|ms|s|m|h) 0s
- max_failure_ratio 0
- order stop-first start-first stop-first
- update_config
- parallelism
- delay
- failure_action continue rollback pause pause
- monitor (ns|us|ms|s|m|h) 0s
- max_failure_ratio
- order stop-first start-first stop-first

V3.4

devices

```
devices:
  - "/dev/ttyUSB0:/dev/ttyUSB0"
```

dns

DNS

```
dns: 8.8.8.8


dns:
  - 8.8.8.8
  - 9.9.9.9
```

dns_search

DNS

```
dns_search: example.com


dns_search:
  - dc1.example.com
  - dc2.example.com
```

entrypoint

entrypoint

```
entrypoint: /code/entrypoint.sh
```

```
entrypoint:
    - php
    - -d
    - zend_extension=/usr/local/lib/php/extensions/no-debug-non-zts-20100525/xdebug.so
    - -d
    - memory_limit=-1
    - vendor/bin/phpunit
```

env_file

```
env_file: .env
```

```
env_file:
  - ./common.env
  - ./apps/web.env
  - /opt/secrets.env
```

environment

YML True False

```
environment:
  RACK_ENV: development
  SHOW: 'true'
```

expose

```
expose:
 - "3000"
 - "8000"
```

extra_hosts

docker client --add-host

```
extra_hosts:
 - "somehost: 162. 242. 195. 82"
 - "otherhost: 50. 31. 209. 229"
```

/etc/hosts ip

```
162. 242. 195. 82  somehost
50. 31. 209. 229   otherhost
```

healthcheck

docker

```
healthcheck:
  test: ["CMD", "curl", "-f", "http: //localhost"] #
  interval: 1m30s #
  timeout: 10s #
  retries: 3 #
  start_period: 40s #
```

image

```
image: redis
image: ubuntu: 14. 04
image: tutum/influxdb
image: example-registry. com: 4000/postgresql
image: a4bc65fd #   id
```

logging

driver json-file

```
driver: "json-file"
driver: "syslog"
```

```
driver: "none"
```

json-file

```
logging:
  driver: json-file
  options:
    max-size: "200k" #       200k
    max-file: "10" #   10
```

syslog syslog-address

```
logging:
  driver: syslog
  options:
    syslog-address: "tcp://192.168.0.42:123"
```

network_mode

```
network_mode: "bridge"
network_mode: "host"
network_mode: "none"
network_mode: "service:[service name]"
network_mode: "container:[container name/id]"
```

networks

networks

```
services:
  some-service:
    networks:
      some-network:
        aliases:
         - alias1
      other-network:
        aliases:
         - alias2
networks:
  some-network:
```

```
    # Use a custom driver
    driver: custom-driver-1
  other-network:
    # Use a custom driver which takes special options
    driver: custom-driver-2
```

aliases

restart

- no
- always
- on-failure 0
- unless-stopped Docker

```
restart: "no"
restart: always
restart: on-failure
restart: unless-stopped
```

swarm restart_policy

secrets

```
version: "3.1"
services:

mysql:
  image: mysql
  environment:
    MYSQL_ROOT_PASSWORD_FILE: /run/secrets/my_secret
  secrets:
    - my_secret

secrets:
  my_secret:
    file: ./my_secret.txt
```

security_opt

schema

```
security-opt
  - label: user: USER   #
  - label: role: ROLE   #
  - label: type: TYPE   #
  - label: level: LEVEL  #
```

stop_grace_period

SIGTERM (stop_signal) SIGKILL

```
stop_grace_period: 1s #   1
stop_grace_period: 1m30s #   1   30
```

10

stop_signal

SIGTERM

SIGUSR1 SIGTERM

```
stop_signal: SIGUSR1
```

sysctls

```
sysctls:
  net.core.somaxconn: 1024
  net.ipv4.tcp_syncookies: 0

sysctls:
  - net.core.somaxconn=1024
  - net.ipv4.tcp_syncookies=0
```

tmpfs

```
tmpfs: /run
```

```
tmpfs:
  - /run
  - /tmp
```

ulimits

ulimit

```
ulimits:
  nproc: 65535
  nofile:
    soft: 20000
    hard: 40000
```

volumes

```
version: "3.7"
services:
  db:
    image: postgres:latest
    volumes:
      - "/localhost/postgres.sock:/var/run/postgres/postgres.sock"
      - "/localhost/data:/var/lib/postgresql/data"
```

Dockerfile

```
FROM openjdk:8
VOLUME /tmp
# EXPOSE 80
ENV TZ=Asia/Shanghai JAVA_OPTS="-Xmx512m"
RUN ln -snf /usr/share/zoneinfo/$TZ /etc/localtime && echo $TZ > /etc/timezone
ADD *.jar app.jar
ENTRYPOINT java $JAVA_OPTS -Djava.security.egd=file:/dev/./urandom -jar /app.jar
# ENTRYPOINT exec java $JAVA_OPTS -Djava.security.egd=file:/dev/./urandom -jar /app.jar --
Dspring.config.location=/config/*
#     Tomcat        java.security.egd      /dev/urandom   ENTRYPOINT
ENTRYPOINT ["java", $JAVA_OPTS, "-Djava.security.egd=file:/dev/./urandom","-jar", "/app.jar",
"--Dspring.config.location=/config/*"]

FROM java:8
VOLUME /tmp
EXPOSE 8080
ENV TZ=Asia/Shanghai JAVA_OPTS="-Xmx512m"
RUN ln -snf /usr/share/zoneinfo/$TZ /etc/localtime && echo $TZ > /etc/timezone
ADD *.jar app.jar
ENTRYPOINT java $JAVA_OPTS -Djava.security.egd=file:/dev/./urandom -jar /app.jar

#
FROM openjdk:11
VOLUME /tmp
ENV TZ=Asia/Shanghai JAVA_OPTS="-Xmx512m"
RUN ln -snf /usr/share/zoneinfo/$TZ /etc/localtime && echo $TZ > /etc/timezone
ADD *.jar app.jar
# CMD echo "10.254.7.7 bzdts.chinaetc.org" >> /etc/hosts; java $JAVA_OPTS -
Djava.security.egd=file:/dev/./urandom -jar /app.jar --Dspring.config.location=/config/*
ENTRYPOINT echo "10.254.7.7 bzdts.chinaetc.org" >> /etc/hosts && java $JAVA_OPTS -
Djava.security.egd=file:/dev/./urandom -jar /app.jar --Dspring.config.location=/config/*



# tomcat_8.5.47-jdk11
FROM tomcat:8.5.47-jdk11
```

```
# MAINTAINER itqmdx@gmail.com
# VOLUME /usr/local/tomcat
EXPOSE 8080
# RUN rm -rf /usr/local/tomcat/webapps/*
ENV TZ=Asia/Shanghai JAVA_OPTS=-Xmx512m
RUN ln -snf /usr/share/zoneinfo/$TZ /etc/localtime && echo $TZ > /etc/timezone
# ADD ./target/* /usr/local/tomcat/webapps/
ENTRYPOINT ['/usr/local/tomcat/bin/catalina.sh', 'run']
# ENTRYPOINT echo '10.254.7.7 bzdts.chinaetc.org' >> /etc/hosts &&
/usr/local/tomcat/bin/catalina.sh run
```

[shell] docker deploy service script

```
#! /bin/bash

# service name
SERVICE_NAME=service-name
# service port (--net=host invalid)
OPEN_PORT=7000
#
INSTANCES=1

# log path
LOG_PATH=/logs

# author: wzhz
# email: itqmdx@gmail.com
# version: v0.4
version=v0.4

# local ip (--net=host invalid)
IP=$(ip a | grep inet | grep -v 127.0.0.1 | grep -v inet6 | grep -v docker | awk '{print $2}'
| tr -d 'addr:' | awk -F '/' '{print $1}' | head -1)
# get ip
# ip addr | grep 'state UP' -A2 | tail -n1 | awk '{print $2}' | awk -F"/" '{print $1}'
DATEVERSION=$(date +'%Y.%m.%d.%H')

# script_dir
script_dir=$(readlink -f $0)
bootpath=$(dirname $script_dir)

# logspath=$bootpath/logs
# configpath=$bootpath/config
# jarpath=$bootpath/jar

RED='\e[1;31m'
```

```
GREEN='\e[1;32m'
YELLOW='\033[1;33m'
BLUE='\E[1;34m'
PINK='\E[1;35m'
RES='\033[0m'

# get all filename in specified path
getFileName() {
  path=$1
  files=$(ls $bootpath/jar)
  for filename in $files
  do
    echo $filename # >> filename.txt
  done

  for file in `find  $1 -name "*.jar"`
  do
    echo $file
  done
}

# touch Dockerfile
createDockerfile() {
#  --Dspring.config.location=/config/*
cat > ./Dockerfile << EOF
FROM openjdk:8
VOLUME /logs
EXPOSE $OPEN_PORT
ENV TZ=Asia/Shanghai JAVA_OPTS="-server -Xms512m -Xmx512m -XX:PermSize=64M -
XX:MaxNewSize=256m -XX:MaxPermSize=128m -Djava.awt.headless=true "
RUN ln -snf /usr/share/zoneinfo/\$TZ /etc/localtime && echo \$TZ > /etc/timezone
ADD *.jar app.jar
ENTRYPOINT exec java \$JAVA_OPTS -Djava.security.egd=file:/dev/./urandom -jar /app.jar
EOF
}

# delete old and images
deleteOldImage() {
  docker image rm -f $SERVICE_NAME:$DATEVERSION >> /dev/null 2>&1;
  docker image ls
```

```
}
# delete old containers
deleteOldContainer() {
  OLD_INSTANCES=$(docker container ps -a | grep -i $SERVICE_NAME | wc -l);
  for((i=0;i<$OLD_INSTANCES;i++));
  do
    docker container stop $SERVICE_NAME-$i >> /dev/null 2>&1;
    docker container rm -f $SERVICE_NAME-$i >> /dev/null 2>&1;
  done
  # rm -rf $bootpath/logs;
  if docker container ps -a | grep -i $SERVICE_NAME;then
    echo -e $RED hase $OLD_INSTANCES instances. $RES
  fi
  docker container ps
}

# build docker image
buildImage() {
  docker build -t $SERVICE_NAME:$DATEVERSION . ;
  docker image ls
}

# run docker container
runImage() {
  for((i=0;i < $INSTANCES;i++));
  do
    name=$SERVICE_NAME-$i
    port=$(($OPEN_PORT+$i))
    docker container rm -f $name >> /dev/null 2>&1
    echo create container is $name:$IP:$port;

    # docker run \
    # --net=host \
    # -v $bootpath/logs:$LOG_PATH \
    # -v $bootpath/config:/config \
    # --name $name \
    # --restart=on-failure:10 \
    # -d $SERVICE_NAME >> /dev/null 2>&1
    docker run \
    --expose=$port \
```

```
    -p $port:$port \
    -v $bootpath/logs:$LOG_PATH \
    -e server.port=$port \
    -e spring.application.name=$SERVICE_NAME \
    -e spring.cloud.client.ip-address=$IP \
    -e EUREKA_INSTANCE_INSTANCE-ID=$IP:$SERVICE_NAME:$port \
    -e EUREKA_INSTANCE_IP-ADDRESS=$IP \
    -e SERVER_PORT=$port \
    -e JAVA_OPTS=-Xmx512m \
    --name $name \
    --restart=on-failure:10 \
    -d $SERVICE_NAME:$DATEVERSION # >> /dev/null 2>&1
    CONTAINERID_NEW=`docker container ps -a | grep ${name}| awk '{print $NF}'`
    echo new container created successfully is $CONTAINERID_NEW
    if [ $i -lt $INSTANCES ];then
      sleep 1;
    fi
  done
  docker container ps
}
startContainer() {
  for((i=0;i < $INSTANCES;i++));
  do
    name=$SERVICE_NAME-$i
    port=$(($OPEN_PORT+$i))

    docker container start $name >> /dev/null 2>&1
    echo start container is $name:$IP:$port
    if [ $i -lt $INSTANCES ];then
      sleep 1;
    fi
  done
  docker container ps
}
restartContainer() {
  for((i=0;i < $INSTANCES;i++));
  do
    name=$SERVICE_NAME-$i
    port=$(($OPEN_PORT+$i))
```

```
    docker container restart $name >> /dev/null 2>&1
    echo restart container is $name:$IP:$port
    if [ $i -lt $INSTANCES ];then
      sleep 1;
    fi
  done
  docker container ps
}
stopContainer() {
  for((i=0;i < $INSTANCES;i++));
  do
    name=$SERVICE_NAME-$i
    port=$(($OPEN_PORT+$i))

    docker container stop $name >> /dev/null 2>&1
    echo stop container is $name:$IP:$port
    if [ $i -lt $INSTANCES ];then
      sleep 1;
    fi
  done
  docker container ps
}
viewContainerLog() {
  if [ $INSTANCES -eq 1 ];then
    showLog $SERVICE_NAME-0
  else
    # more
    echo -e $GREEN show logs for containers: $RES
    docker ps -a | grep ${SERVICE_NAME}| awk '{print $1, $2, $(NF-1), $NF}'
    read -p 'please input a container id or name: ' input
    showLog $input
  fi
}
showLog() {
  docker container logs -f --tail=100 $1
}
readme() {
  echo -e $GREEN ---- deploy service script $RES
  echo -e $YELLOW ---- author: wzhz $RES
  echo -e $YELLOW ---- email: itqmdx@gmail.com $RES
```

```
  echo -e $YELLOW ---- version: $version $RES
}

current() {
  echo
  echo -e $PINK current time is $(date +'%Y-%m-%d %T') $RES
  echo
}

var() {
  echo
  echo IP $IP
  echo SERVICE_NAME $SERVICE_NAME
  echo OPEN_PORT $OPEN_PORT
  echo INSTANCES $INSTANCES
  echo DATEVERSION $DATEVERSION
  echo
}

# setting env var
setEnvironmentVariable() {
  ARRT=$1
  ARRT_NAME=`echo ${ARRT} | awk -F '=' '{print $1}'`
  ARRT_VALUE=`echo ${ARRT} | awk -F '=' '{print $2}'`
  # echo $ARRT_NAME is $ARRT_VALUE
  if [ $ARRT_NAME == 'name' ]; then
    SERVICE_NAME=$ARRT_VALUE
  elif [ $ARRT_NAME == 'port' ]; then
    OPEN_PORT=$ARRT_VALUE
  elif [ $ARRT_NAME == 'ip' ]; then
    IP=$ARRT_VALUE
  elif [ $ARRT_NAME == 'i' ]; then
    INSTANCES=$ARRT_VALUE
  else
    echo
    echo -e $RED $ARRT no matches found. $RES
    echo
  fi
}
```

```
functionItems() {
  echo
  echo -e $GREEN = 0. perform steps 7-8 and 1-3 automatically $RES
  echo -e $BLUE = 1. create current environment\'s Dockerfile $RES
  echo -e $BLUE = 2. build image $SERVICE_NAME $RES
  echo -e $BLUE = 3. run image $SERVICE_NAME $RES
  echo -e $BLUE = 4. start $SERVICE_NAME\'s containers $RES
  echo -e $BLUE = 5. restart $SERVICE_NAME\'s containers $RES
  echo -e $BLUE = 6. stop $SERVICE_NAME\'s containers $RES
  echo -e $BLUE = 7. delete $SERVICE_NAME\'s containers  $RES
  echo -e $BLUE = 8. delete image $SERVICE_NAME $RES
  echo -e $BLUE = 9. view $SERVICE_NAME\'s container log $RES
  echo -e $RED = 99. configure global information $RES
  echo
}

main() {
  functionItems
  read -p 'please input a function item no: ' input
  echo your input is $input
  case $input in
    0)
    deleteOldContainer
    echo -e $GREEN delete containers successfully. $RES
    deleteOldImage
    echo -e $GREEN delete image successfully. $RES
    createDockerfile
    echo -e $GREEN Dockerfile created successfully, default is based on openjdk:8. $RES
    buildImage
    echo -e $GREEN created successfully, image is $SERVICE_NAME. $RES
    runImage
    echo -e $GREEN runs successfully. $RES
    ;;
    1)
    createDockerfile
    echo -e $GREEN Dockerfile created successfully, default is based on openjdk:8. $RES
    cat Dockerfile
    ;;
    2)
    buildImage
```

```
    echo -e $GREEN created successfully, image is $SERVICE_NAME. $RES
    ;;
    3)
    runImage
    echo -e $GREEN runs successfully. $RES
    ;;
    4)
    startContainer
    echo -e $GREEN start container successfully. $RES
    ;;
    5)
    restartContainer
    echo -e $GREEN restart container successfully. $RES
    ;;
    6)
    stopContainer
    echo -e $GREEN stop container successfully. $RES
    ;;
    7)
    deleteOldContainer
    echo -e $GREEN delete containers successfully. $RES
    ;;
    8)
    deleteOldImage
    echo -e $GREEN delete image successfully. $RES
    ;;
    9)
    viewContainerLog
    echo -e $GREEN view container log complete. $RES
    ;;
    99)
    echo -e $YELLOW developing... $RES
    ;;
    *)
    echo -e $RED wrong input, exit 0. $RES
    exit 0
    ;;
  esac
}
```

```
echo -e $YELLOW working directory is $bootpath $RES
cd $bootpath;ls -all;

for arg in $@
do
  setEnvironmentVariable $arg
done

readme

current

var

while true
do
  main
  sleep 1
done
```

[shell] Docker 2

```
#! /bin/bash

if [ $1 == "" ];then
     echo "plsese input the image name of the jar api"
     exit 0
fi

IMAGE=$1
echo "input image is ${IMAGE}"









######################### delete the container of this jar api  ########################

CONTAINERID=`docker ps -a| grep ${IMAGE} |awk '{print $1}'`
echo "contained id is ${CONTAINERID}"

docker stop ${CONTAINERID}
docker rm ${CONTAINERID}

########################  get image name and image tag ....sed -n          ######################


IMAGENAME=`echo ${IMAGE} |  awk -F ":" '{print $1}'`
echo "imagename is ${IMAGENAME}"

IMAGETAG=`echo ${IMAGE}|  awk -F ":" '{print $2}'`
echo "imagetag is ${IMAGETAG}"

IMAGEROWNUM=`docker images |  grep ${IMAGENAME}|  wc -l`
echo "imagerownum is ${IMAGEROWNUM}"

########################                                tag       ########################
```

```
for ((i=1;i<=$IMAGEROWNUM;i++))
  do
    echo "i is ${i}"
    IMAGENAME2=`docker images|grep ${IMAGENAME} | sed -n "${i}p" |awk '{print $1}'`
    echo "imagename2 is ${IMAGENAME2}"
    if [ "${IMAGENAME2}" == "${IMAGENAME}" ];then
    IMAGETAG2=`docker images|grep ${IMAGENAME} | sed -n "${i}p" |awk '{print $2}'`
    echo "imagetag2 is ${IMAGETAG2}"
       if [ "${IMAGETAG2}" == "${IMAGETAG}" ];then
       IMAGEID=`docker images|grep ${IMAGENAME} | sed -n "${i}p" |awk '{print $3}'`
       echo "imageid is ${IMAGEID}"

       docker rmi ${IMAGEID}
       fi
    fi
done


########################      ########################

docker build -t ${IMAGE} .

########################                 dockerfile   java -jar     ########################

docker run -d --net=host -v /home/xjjtuser/dataAnalysis-logs/:/data-analysis/  -v
/home/xjjtuser/docker-program/config/:/config/ --name data-analysis  ${IMAGE}

#
containerid_new=`docker ps -a | grep ${IMAGE}| awk '{print $1}'`
echo "containerid_new is ${containerid_new}"

docker logs "${containerid_new}"
```

docker

docker

`root`

`vi /etc/docker/daemon.json`

```
{
"log-driver":"json-file",
"log-opts":{"max-size":"500M","max-file":"3"}
}
```

`docker`

docke`systemctl restart docker`

Docker Container

Docker Container

0x00

Docker `namespace` `Docker` `Container` `docker run --link` `docker-compose` `docker image` `CTF` `Image`

0x01

- upstart

```
# Dockerfile
From ubuntu:14.04
RUN apt-get update && apt-get upgrade -y && apt-get install mysql apache2 php7.0
ADD web /var/www/html
RUN service mysql start && /var/www/html/init_sql.sh && service mysql stop
CMD service mysql start && service apache2 start && while true; do sleep 10;done
```

- systemd

```
# Dockerfile
From ubuntu:16.04
RUN apt-get update && apt-get upgrade -y && apt-get install mysql apache2 php7.0
ADD web /var/www/html
RUN systemctl start mysql && /var/www/html/init_sql.sh && systemctl stop mysql
CMD systemctl start mysql && systemctl start apache2 && while true; do sleep 10;done
```

```
# Dockerfile
From ubuntu:16.04
RUN apt-get update && apt-get upgrade -y && apt-get install mysql apache2 php7.0
ADD web /var/www/html
ADD entrypoint.sh /sbin/
RUN chmod +x /sbin/entrypoint.sh /var/www/html/init_sql.sh&&
     /etc/init.d/mysql start && /var/www/html/init_sql.sh && /etc/init.d/mysql stop
```

```
```

```
CMD /sbin/entrypoint.sh
```

```
#!/bin/bash

# entrypoint.sh
/usr/bin/mysqld start &
/usr/bin/httpd &
while true
do
sleep 100
done
```

1 2 container 3

```
SIGTERM
```

1 2 docker service xxx start systemctl start xxx upstart container init 3 init PID=1

```
root@vpscn:/var/lib/docker# docker exec -it hackmd sh
/hackmd # ps -ef
PID   USER     TIME   COMMAND
    1 hackmd     1:03 node app.js
   42 hackmd     0:00 /usr/local/bin/node ./lib/workers/dmpWorker.js
   62 root       0:00 sh
   69 root       0:00 ps -ef
```

3 container --init tini init

0x02

phusion/baseimage SUCTF2018 0.10.1 ubuntu 16.04 a Container Entrypoint runit ~~WebIDE~~ Coding Web Terminal NUAACTF/SUCTF xinetd

SUCTF2018 Dockerfile

```
#Dockerfile
FROM phusion/baseimage:0.10.1
MAINTAINER Yibai Zhang
```

```
RUN sed -i 's/archive.ubuntu.com/mirrors.aliyun.com/g' /etc/apt/sources.list &&
    sed -i 's/security.ubuntu.com/mirrors.aliyun.com/g' /etc/apt/sources.list &&
    apt-get update && apt-get install -y apache2 libapache2-mod-php php-mysql mariadb-server
&&
    apt-get clean && rm -rf /var/lib/apt/lists/* /tmp/* /var/tmp/* /var/www/html/*

RUN mkdir -p /etc/service/apache2/ &&
    printf "#!/bin/shnntrap "apachectl -k graceful-stop" 1 2 3 6 15nnexec /usr/sbin/apachectl
-D FOREGROUNDn" > /etc/service/apache2/run &&
    chmod +x /etc/service/apache2/run && mkdir -p /etc/service/mysql/ &&
    printf "#!/bin/shnntrap "mysqladmin -uroot -psuCTF_P1us_1s shutdown" 1 2 3 6 15nnexec
/usr/bin/mysqld_safe" > /etc/service/mysql/run &&
    mkdir -p /var/run/mysqld/ && chown mysql:mysql /var/run/mysqld &&
    chmod 700 /etc/service/mysql/run /etc/service/apache2/run

COPY web /var/www/html
COPY flag /flag
RUN echo "secure-file-priv=/var/www/" >>/etc/mysql/mariadb.conf.d/50-server.cnf && chmod -R
777 /var/www/html/favicon
COPY init_sql.sh /tmp/init_sql.sh
RUN chmod +x /tmp/init_sql.sh && bash -c "/tmp/init_sql.sh" && rm /tmp/init_sql.sh
EXPOSE 80
```

```
#!/usr/bin/env bash
#init_sql.sh

mysqld_safe &
echo -n "Waiting for mysql startup"
while ! mysqladmin --host="localhost" --silent ping ; do
    echo -n "."
    sleep 1
done
echo

mysql -uroot <<EOF
UPDATE mysql.user SET Password=PASSWORD('XXXXXX'), plugin = '' WHERE User='root';
create database calc;
use calc;
create table user(
id INT NOT NULL AUTO_INCREMENT primary key,
```

```
username varchar(32) NOT NULL,
password varchar(32) NOT NULL
)ENGINE=InnoDB DEFAULT CHARSET=utf8;
insert into user values(1,'admin','aa67095d8e65d624548cb6b50bd4778e');
create table file(
id INT NOT NULL AUTO_INCREMENT primary key,
filename varchar(32) NOT NULL,
filehash varchar(32) NOT NULL,
sig varchar(120) NOT NULL
)ENGINE=InnoDB DEFAULT CHARSET=utf8;
create table flag(
flag varchar(120) primary key
)ENGINE=InnoDB DEFAULT CHARSET=utf8;
insert into flag values('SUCTF{a_very_long_long_long_long_long_fake_flag_d}');
grant SELECT, INSERT on calc.user to 'suctf'@localhost identified by 'suctf';
grant SELECT, INSERT, UPDATE on calc.file to 'suctf'@localhost ;
grant SELECT on calc.flag to 'suctf'@localhost ;
FLUSH PRIVILEGES;
EOF

mysqladmin -uroot -pXXXXXX shutdown
```

printf "#!/bin/shnntrap "apachectl -k graceful-stop" 1 2 3 6 15nnexec /usr/sbin/apachectl -D FOREGROUNDn" > /etc/service/apache2/run runit phusion/baseimage run /etc/service/apache2/run

```
#! /bin/sh

trap "apachectl -k graceful-stop" 1 2 3 6 15

exec /usr/sbin/apachectl -D FOREGROUND
```

runit Apache2 1 2 3 6 15 (graceful) Apache2 Apac
apache2 Container ~~ -