4.2.1.9 docker最佳实践

1. 安装及版本

建议安装版本：

Docker 版本：24.0.2

Docker-compose 版本：2.17.2

1.1 在线安装

1.1.1 安装docker

安装参考网址：

https://yeasy.gitbook.io/docker_practice/install/centos

- 卸载旧版本

```
$ sudo yum remove docker \
                  docker-client \
                  docker-client-latest \
                  docker-common \
                  docker-latest \
                  docker-latest-logrotate \
                  docker-logrotate \
                  docker-selinux \
                  docker-engine-selinux \
                  docker-engine
```

- 使用 yum 安装

```
sudo yum install -y yum-utils

#
sudo yum-config-manager \
    --add-repo \
    https://mirrors.aliyun.com/docker-ce/linux/centos/docker-ce.repo

sudo sed -i 's/download.docker.com/mirrors.aliyun.com\/docker-ce/g' /etc/yum.repos.d/docker-ce.repo
```

- 安装docker

```
sudo yum install docker-ce docker-ce-cli containerd.io
```

- 启动docker

```
sudo systemctl enable docker
sudo systemctl start docker
```

- 测试 Docker 是否安装正确

```
docker run --rm hello-world
```

1.1.2 安装docker-compse

安装参考网址：

https://yeasy.gitbook.io/docker_practice/compose/install

- 安装

```
sudo curl -L https://github.com/docker/compose/releases/download/v2.17.2/docker-compose-`uname -s`-`uname -m` > /usr/local/bin/docker-compose

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

- 测试

```
docker-compose --version
```

- 卸载

```
sudo rm /usr/local/bin/docker-compose
```

1.2 离线安装

1.2.1 安装docker

- 若服务器可连接外网，可从官网下载docker

https://download.docker.com/linux/static/stable/x86_64/docker-24.0.2.tgz

- 若服务器可连接广联达文档中心，可从文档中心下载

https://gly-prod-gdoc-cityha.oss-cn-beijing.aliyuncs.com/object/76/2ebd70eb8c456cb51d35b2f42ef76a?Expires=1695033774&OSSAccessKeyId=TfBNAWbCpI2jFI7t&Signature=JcY1z03mNwUZ9NhwXnmsh611pUI%3D&response-content-disposition=attachment%3Bfilename%3D%22docker-24.0.2.tgz%22

- 若服务器不可连接外网和文档中心，则本地下载通过传输工具传输到服务器上

软件安装（CentOS7）

```
  tar -zxvf docker-24.0.2.tgz
  docker /usr/bin/
  cp docker/* /usr/bin/
  /etc/systemd/system/,docker.service
  cd /etc/systemd/system/
  touch docker.service
  docker.service,
  vim docker.service
```

```
[Unit]
Description=Docker Application Container Engine
Documentation=https://docs.docker.com
After=network-online.target firewalld.service
Wants=network-online.target

[Service]
Type=notify
# the default is not to use systemd for cgroups because the delegate issues still
# exists and systemd currently does not support the cgroup feature set required
# for containers run by docker
ExecStart=/usr/bin/dockerd --selinux-enabled=false --insecure-registry=192.168.200.128
ExecReload=/bin/kill -s HUP $MAINPID
# Having non-zero Limit*s causes performance problems due to accounting overhead
# in the kernel. We recommend using cgroups to do container-local accounting.
LimitNOFILE=infinity
LimitNPROC=infinity
LimitCORE=infinity
# Uncomment TasksMax if your systemd version supports it.
# Only systemd 226 and above support this version.
#TasksMax=infinity
TimeoutStartSec=0
# set delegate yes so that systemd does not reset the cgroups of docker containers
Delegate=yes
# kill only the docker process, not all processes in the cgroup
KillMode=process
# restart the docker process if it exits prematurely
Restart=on-failure
StartLimitBurst=3
StartLimitInterval=60s

[Install]
WantedBy=multi-user.target
```

```
  docker.service
  chmod 777 /etc/systemd/system/docker.service
  docker.service
  systemctl daemon-reload

  systemctl start docker

  systemctl enable docker.service
  docker
  systemctl enable docker.service
```

软件安装（华为欧拉）

```
tar -zxvf docker-24.0.2.tgz
chmod +x docker/*
#/usr/bin/
cp docker/* /usr/bin/
#
vim /usr/lib/systemd/system/docker.service

#
[Unit]
Description=Docker Application Container Engine
Documentation=https://docs.docker.com
After=network-online.target firewalld.service
Wants=network-online.target

[Service]
Type=notify
#docker/MYAPP/docker/lib
ExecStart=/usr/bin/dockerd
ExecReload=/bin/kill -s HUP $MAINPID
LimitNOFILE=65535
LimitNPROC=65535
LimitCORE=65535
TasksMax=65535
TimeoutStartSec=0
Delegate=yes
KillMode=process
Restart=on-failure
StartLimitBurst=3
StartLimitInterval=60s

[Install]
WantedBy=multi-user.target


#
mkdir -p /data/docker/lib
systemctl daemon-reload
#
systemctl start docker && systemctl enable docker
#
systemctl status docker
```

1.2.2 安装docker-compsoe

docker-compose-linux-x86_64

将包上传至服务器，执行

mv docker-compose-linux-x86_64 /usr/local/bin/docker-compose

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

测试

docker-compose -v

1.3 用户名和密码

```
docker login --username=bimone-dev@bimcop jsf-image.glodon.com
 x5d1ajknpYsBNW3O
```

2. 默认配置

在项目中可以在系统根目录（即pom.xml所在目录）自定义Dockerfile文件。若没有自定义的Dockerfile文件，则会采用默认的Dockerfile文件。此处贴出默认的Dockerfile（以quality服务为例）。

默认Dockerfile

```
FROM jsf-image.glodon.com/bimone/migration:1.0.8
RUN mkdir -p /data/lib
WORKDIR /data
COPY **/target/estate-quality-service*.jar ./app.jar
RUN mkdir -p /migration/scripts/
COPY migration/scripts /migration/scripts/
 CMD java $JAVA_OPTS -jar app.jar
```

启动docker镜像时，入口为entrypoint.sh，此处贴出默认的entrypoint.sh

```
#!/bin/sh
cd /migration
mkdir log
./tools/bin/migrate status > ./log/status.log
./tools/bin/migrate pending --force > ./log/pending.log
cd /data
java $JAVA_OPTS -jar /data/app.jar
```

3. 自定义配置

推荐如果有启动依赖的服务，将被依赖的服务单独抽离出一个独立的docker-compose单独启动，其他服务建成批量启动。（depends_on不好用）

3.1 批量启动服务（以工程工作台为例）

进入目录/home/deploy/bin 目录，

执行命令./deployall.sh即可启动多个服务。

3.1.1 deployall.sh

```bash
#!/bin/bash
#
source ./config.sh

#
if grep -q "^Quality=" config.sh; then
    #
    quality=$(grep "^Quality=" config.sh | cut -d= -f2-)
    #
    if [[ -z "$quality" ]]; then
        echo "quality hasn't set version, please set it in the config.sh file."
        exit
    fi
else
    echo "quality hasn't set version, please set it in the config.sh file."
    exit
fi

if grep -q "^Safety=" config.sh; then
    #
    safety=$(grep "^Safety=" config.sh | cut -d= -f2-)
    #
    if [[ -z "$safety" ]]; then
        echo "safety hasn't set version, please set it in the config.sh file."
        exit
    fi
else
    echo "safety hasn't set version, please set it in the config.sh file."
    exit
fi

if grep -q "^Construction=" config.sh; then
    #
    construction=$(grep "^Construction=" config.sh | cut -d= -f2-)
    #
    if [[ -z "$construction" ]]; then
        echo "construction hasn't set version, please set it in the config.sh file."
        exit
    fi
else
    echo "construction hasn't set version, please set it in the config.sh file."
    exit
fi

if grep -q "^Plan=" config.sh; then
    #
    plan=$(grep "^Plan=" config.sh | cut -d= -f2-)
    #
    if [[ -z "$plan" ]]; then
        echo "plan hasn't set version, please set it in the config.sh file."
        exit
    fi
else
    echo "plan hasn't set version, please set it in the config.sh file."
    exit
fi

if grep -q "^Statistic=" config.sh; then
    #
    statistic=$(grep "^Statistic=" config.sh | cut -d= -f2-)
    #
    if [[ -z "$statistic" ]]; then
        echo "statistic hasn't set version, please set it in the config.sh file."
        exit
    fi
else
    echo "statistic hasn't set version, please set it in the config.sh file."
    exit
fi

echo "quality= $quality"
echo "safety= $safety"
echo "construction= $construction"
echo "plan= $plan"
echo "statistic= $statistic"

docker-compose --env-file ./config.sh pull
docker-compose --env-file ./config.sh up -d
```

大部分为校验，如果认为没有必要，则只要以下部分即可：

```bash
docker-compose --env-file ./config.sh pull
docker-compose --env-file ./config.sh up -d
```

3.1.2 config.sh

```
Quality=v1.28.51
Construction=v1.0.15
Safety=v1.20.66
Plan=v1.17.41
Statistic=v1.7.44
```

该配置文件记录各服务tag号。

3.1.3 docker-compose.yml

给出工作台服务配置文件示例（此配置文件基于自定义的Dockerfile和entrypoint.sh）：

```
version: "3.3"
services:
  estate-quality-service:
    image: jsf-image.glodon.com/bimone/estate-quality-service:$Quality
    ports:
      - "7099:7099"
    restart: always
    network_mode: host
    environment:
      spring.application.name: quality
      spring.cloud.config.label: lc
      spring.cloud.config.uri: http://192.168.6.48:7001/
      spring.profiles.active: prod
      JAVA_OPTS: -server -Xms2048m -Xmx2048m -XX:+HeapDumpOnOutOfMemoryError -XX:HeapDumpPath=/data/log
/java_heapdump.hprof
      TZ: Asia/Shanghai
    volumes:
      - /data/deploy/bin/estate-quality-service/environments:/migration/environments
      - /data/deploy/bin/estate-quality-service/log:/data/log
      - /data/deploy/bin/estate-quality-service/migrationlog:/migration/log
    healthcheck:
      test: ["CMD", "curl", "-f", "http://127.0.0.1:7099/actuator/health"]
      interval: 45s
      timeout: 30s
      retries: 3

  estate-safety-service:
    image: jsf-image.glodon.com/bimone/estate-safety-service:$Safety
    ports:
      - "7091:7091"
    restart: always
    network_mode: host
    environment:
      spring.application.name: safety
      spring.cloud.config.label: lc
      spring.cloud.config.uri: http://192.168.6.48:7001/
      spring.profiles.active: prod
      JAVA_OPTS: -server -Xms2048m -Xmx2048m -XX:+HeapDumpOnOutOfMemoryError -XX:HeapDumpPath=/data/log
/java_heapdump.hprof
      TZ: Asia/Shanghai
    volumes:
      - /data/deploy/bin/estate-safety-service/environments:/migration/environments
      - /data/deploy/bin/estate-safety-service/log:/data/log
      - /data/deploy/bin/estate-safety-service/migrationlog:/migration/log
    healthcheck:
      test: ["CMD", "curl", "-f", "http://127.0.0.1:7091/actuator/health"]
      interval: 45s
      timeout: 30s
      retries: 3

  estate-construction-basic-service:
    image: jsf-image.glodon.com/bimone/estate-construction-basic-service:$Construction
    ports:
      - "7094:7094"
    restart: always
    network_mode: host
    environment:
      spring.application.name: construction-basic
      spring.cloud.config.label: lc
      spring.cloud.config.uri: http://192.168.6.48:7001/
      spring.profiles.active: prod
      JAVA_OPTS: -server -Xms2048m -Xmx2048m -XX:+HeapDumpOnOutOfMemoryError -XX:HeapDumpPath=/data/log
/java_heapdump.hprof
      TZ: Asia/Shanghai
    volumes:
      - /data/deploy/bin/estate-construction-basic-service/environments:/migration/environments
      - /data/deploy/bin/estate-construction-basic-service/log:/data/log
      - /data/deploy/bin/estate-construction-basic-service/migrationlog:/migration/log
    healthcheck:
      test: ["CMD", "curl", "-f", "http://127.0.0.1:7094/actuator/health"]
      interval: 45s
      timeout: 30s
      retries: 3

  estate-plan-service:
    image: jsf-image.glodon.com/bimone/estate-plan-service:$Plan
    ports:
      - "7090:7090"
    restart: always
    network_mode: host
    environment:
      spring.application.name: plan
      spring.cloud.config.label: lc
      spring.cloud.config.uri: http://192.168.6.48:7001/
      spring.profiles.active: prod
      JAVA_OPTS: -server -Xms2048m -Xmx2048m -XX:+HeapDumpOnOutOfMemoryError -XX:HeapDumpPath=/data/log
/java_heapdump.hprof
      TZ: Asia/Shanghai
    volumes:
      - /data/deploy/bin/estate-plan-service/environments:/migration/environments
      - /data/deploy/bin/estate-plan-service/log:/data/log
      - /data/deploy/bin/estate-plan-service/migrationlog:/migration/log
    healthcheck:
      test: ["CMD", "curl", "-f", "http://127.0.0.1:7090/actuator/health"]
      interval: 45s
      timeout: 30s
      retries: 3

  estate-statistic-service:
    image: jsf-image.glodon.com/bimone/estate-statistic-service:$Statistic
    ports:
      - "7036:7036"
    restart: always
    network_mode: host
    environment:
      spring.application.name: statistic
      spring.cloud.config.label: lc
      spring.cloud.config.uri: http://192.168.6.48:7001/
      spring.profiles.active: prod
      JAVA_OPTS: -server -Xms2048m -Xmx2048m -XX:+HeapDumpOnOutOfMemoryError -XX:HeapDumpPath=/data/log
/java_heapdump.hprof
      TZ: Asia/Shanghai
    volumes:
      - /data/deploy/bin/estate-statistic-service/environments:/migration/environments
      - /data/deploy/bin/estate-statistic-service/log:/data/log
      - /data/deploy/bin/estate-statistic-service/migrationlog:/migration/log
    healthcheck:
      test: ["CMD", "curl", "-f", "http://127.0.0.1:7036/actuator/health"]
      interval: 45s
      timeout: 30s
      retries: 3
```

其中需要修改的地方：

- estate-quality-service、estate-safety-service等改成相应服务；
- image：镜像地址；
- ports：端口；
- environment：环境变量
- volumes：

1. /data/deploy/dev/estate-statistic-service/environments:/migration/environments
/data/deploy/dev/estate-statistic-serviceenvironmentsmigrationenvironmentsdevelopment.propertiesmigration
2. /data/deploy/dev/estate-statistic-service/log:/migration/log

- healthcheck：服务健康校验，详见3.3

3.2 启动单个服务（以quality为例）

进入/data/deploy/bin/estate-quality-service 目录，

执行命令./deploy $VERSION即可启动单个服务。

3.2.1 deploy.sh

```
#!/bin/bash
input_ver=
if [ ! -n "$1" ] ;then
    input_ver="latest"
else
    echo "the word you input is $1"
    input_ver="$1"
fi

echo "input_ver $input_ver"

cd "$(dirname $0)"

docker-compose down
VERSION=$input_ver docker-compose pull
VERSION=$input_ver docker-compose up -d
```

3.2.2 docker-compose.yml

```
version: "3.3"
services:
  quality:
    image: jsf-image.glodon.com/bimone/estate-quality-service:$VERSION
    ports:
      - "7099:7099"
    restart: always
    network_mode: host
    environment:
      spring.application.name: quality
      spring.cloud.config.label: estate
      spring.cloud.config.uri: http://10.20.1.55:7001/
      spring.profiles.active: dev
      JAVA_OPTS: -server -Xms2048m -Xmx2048m -XX:+HeapDumpOnOutOfMemoryError -XX:HeapDumpPath=/data/log
/java_heapdump.hprof
      TZ: Asia/Shanghai
    volumes:
      - /data/deploy/bin/estate-quality-service/environments:/migration/environments
      - /data/deploy/bin/estate-quality-service/log:/data/log
      - /data/deploy/bin/estate-quality-service/migrationlog:/migration/log
    healthcheck:
      test: ["CMD", "curl", "-f", "http://127.0.0.1:7099/actuator/health"]
      interval: 45s
      timeout: 30s
      retries: 3
```

3.3 健康检查

健康检查需要进行两部分操作:

3.3.1 安装autoheal

拉取Autoheal镜像，可以从公司拉取

```
docker pull jsf-image.glodon.com/bimone/autoheal:1.0.0
```

也可以从官方仓库拉取

```
docker pull willfarrell/autoheal
```

安装Autoheal容器

```
docker run -d \
  --name=autoheal \
  --restart=always \
  -e AUTOHEAL_CONTAINER_LABEL=all \
  -v /var/run/docker.sock:/var/run/docker.sock \
  jsf-image.glodon.com/bimone/autoheal:1.0.0 # willfarrell/autoheal
```

请注意，Autoheal只会监控在其之后启动的容器。如果您已经有一些正在运行的容器，您需要重新启动它们，以便Autoheal可以开始监控它们。

3.3.2 docker-compose.yml 添加配置

在docker-compose.yml中添加配置:

```
healthcheck:
  test: ["CMD", "curl", "-f", "http://127.0.0.1:7036/actuator/health"]
  interval: 45s
  timeout: 30s
  retries: 3
```

其中 test 后面需要添加心跳接口，如 http://127.0.0.1:7036/actuator/health，需要根据需要修改。

另外一些旧的容器中没有安装curl命令（1.0.5及之前没有，1.0.7包含），若没有，可以进入容器执行以下命令添加。

```
RUN apk add curl
```

3.4 docker日志配置

1. 确定要配置日志的Docker守护进程的位置。根据操作系统，Docker守护进程的配置文件位置可能会有所不同。以下是一些常见操作系统的示例位置：

- Ubuntu / Debian:

  ```
  /etc/docker/daemon.json
  ```

- CentOS / Fedora:

  ```
  /etc/docker/daemon.json
  ```

- macOS:

  ```
  /etc/docker/daemon.json
  ```

如果找不到该文件，创建一个新文件并将其命名为 daemon.json。

1. 使用文本编辑器打开 daemon.json文件，并添加以下内容：

```
{
  "exec-opts": ["native.cgroupdriver=systemd"],
  "log-driver": "json-file",
  "log-opts": {
    "max-size": "100m",
    "max-file": "10"
  },
  "registry-mirrors": [
    "https://jsf-image.glodon.com"
  ],
  "data-root": "/data/docker/lib"
}
```

上述配置示例将Docker日志驱动程序设置为json-file，并指定每个日志文件的最大大小为100兆字节（"max-size": "100m"），最多保留10个日志文件（"max-file": "10"）。可以根据需要自定义其他日志选项。

1. 保存并关闭 daemon.json 文件。
2. 重新启动Docker守护进程以应用新的日志配置。根据不同的操作系统，执行适当的命令来重新启动Docker守护进程。以下是一些常见操作系统的示例命令：

- Ubuntu / Debian:

  ```
  sudo systemctl restart docker
  ```

- CentOS / Fedora:

  ```
  sudo systemctl restart docker
  ```

- macOS:

  ```
  sudo killall Docker && open -a Docker
  ```

请注意，重新启动Docker守护进程将会中断正在运行的容器，因此请确保在适当的时间执行此操作。

现在，Docker日志已成功配置。可以使用docker logs命令来查看容器的日志。

3.5 docker 清理镜像

清理未使用的镜像以及相关的无用镜像层（dangling layers）：

```
docker image prune -a
```

3.6 docker 可视化

可以安装Portainer可视化工具，docker-compose.yml配置：

```
version: '3'

services:
  portainer:
    image: jsf-image.glodon.com/bimone/portainer:1
    ports:
      - 9000:9000
    volumes:
      - /var/run/docker.sock:/var/run/docker.sock
      - portainer_data:/data
    restart: always

volumes:
  portainer_data:
```

启动命令：

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

然后访问ip:9000进入界面

4. docker常用命令

```
# service
$ sudo service docker start

# systemctl
$ sudo systemctl start docker

#  image
$ docker image ls

#  image
$ docker image rm [imageName]

#  image
docker image pull library/hello-world

#
$ docker container ls

#
$ docker container ls --all

#  SIGKILL
docker container kill [containerID]

#
docker container rm [containerID]

# docker container run image
$ docker container run -p 8000:3000 -it koa-demo /bin/bash
#
$ docker container run -p 8000:3000 -it koa-demo:0.0.1 /bin/bash
-- -p 3000  8000
-- -it Shell  Shell
-- koa-demo:0.0.1image  latest
-- /bin/bash Bash Shell

#
$ docker container start [containerID]

#  SIGTERM  SIGKILL
$ docker container stop [containerID]

#  docker
$ docker container logs [containerID]
-f :  # docker logs -f mynginx
--since :
-t :
--tail :N
# docker logs --since="2016-07-01" --tail=10 mynginx

#  docker
$ docker exec -it [containerID] sh

#  Docker
$ docker container cp [containID]:[/path/to/file] .

# registry.cn-beijing.aliyuncs.com/bimone/migration:1.0.8
$ docker build -t registry.cn-beijing.aliyuncs.com/bimone/migration:1.0.8 .

#
$ docker push registry.cn-beijing.aliyuncs.com/bimone/migration:1.0.8
```

附件

development.properties

```
#
#    Copyright 2010-2016 the original author or authors.
#
#    Licensed under the Apache License, Version 2.0 (the "License");
#    you may not use this file except in compliance with the License.
#    You may obtain a copy of the License at
#
#       http://www.apache.org/licenses/LICENSE-2.0
#
#    Unless required by applicable law or agreed to in writing, software
#    distributed under the License is distributed on an "AS IS" BASIS,
#    WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
#    See the License for the specific language governing permissions and
#    limitations under the License.
#

## Base time zone to ensure times are consistent across machines
time_zone=GMT+8:00

## The character set that scripts are encoded with
script_char_set=UTF-8

## JDBC connection properties.
driver=com.mysql.jdbc.Driver
url=jdbc:mysql://10.0.164.57:30175/construct-product?useUnicode=true&characterEncoding=UTF-8&allowMultiQueries=tr
ue&autoReconnect=true&useSSL=false
username=newuser
password=newuser

#
# A NOTE ON STORED PROCEDURES AND DELIMITERS
#
# Stored procedures and functions commonly have nested delimiters
# that conflict with the schema migration parsing.  If you tend
# to use procs, functions, triggers or anything that could create
# this situation, then you may want to experiment with
# send_full_script=true (preferred), or if you can't use
# send_full_script, then you may have to resort to a full
# line delimiter such as "GO" or "/" or "!RUN!".
#
# Also play with the autocommit settings, as some drivers
# or databases don't support creating procs, functions or
# even tables in a transaction, and others require it.
#

# This ignores the line delimiters and
# simply sends the entire script at once.
# Use with JDBC drivers that can accept large
# blocks of delimited text at once.
send_full_script=true

# This controls how statements are delimited.
# By default statements are delimited by an
# end of line semicolon.  Some databases may
# (e.g. MS SQL Server) may require a full line
# delimiter such as GO.
# These are ignored if send_full_script is true.
delimiter=;
full_line_delimiter=false

# If set to true, each statement is isolated
# in its own transaction.  Otherwise the entire
# script is executed in one transaction.
# Few databases should need this set to true,
# but some do.
auto_commit=false

# Custom driver path to allow you to centralize your driver files
# Default requires the drivers to be in the drivers directory of your
# initialized migration directory (created with "migrate init")
# driver_path=

# Name of the table that tracks changes to the database
changelog=changelog

# Migrations support variable substitutions in the form of ${variable}
# in the migration scripts.  All of the above properties will be ignored though,
# with the exception of changelog.
# Example: The following would be referenced in a migration file as ${ip_address}
# ip_address=192.168.0.1
```

常见问题汇总

报错：/bin/bash^M: bad interpreter

见 https://blog.csdn.net/violet_echo_0908/article/details/52042137