Docker - 快速通关 （3h）

1. 安装

> 国内常见云平台：
>
> - 阿里云、腾讯云、华为云、青云.....

> 使用 CentOS 7.9
>
> WindTerm下载：
> https://github.com/kingToolbox/WindTerm/releases/download/2.6.0/WindTerm_2.6.1_Windows_Portable_x86_64.zip

```shell
# 移除旧版本docker
sudo yum remove docker \
                  docker-client \
                  docker-client-latest \
                  docker-common \
                  docker-latest \
                  docker-latest-logrotate \
                  docker-logrotate \
                  docker-engine

# 配置docker yum源。
sudo yum install -y yum-utils
sudo yum-config-manager \
--add-repo \
http://mirrors.aliyun.com/docker-ce/linux/centos/docker-ce.repo


# 安装 最新 docker
sudo yum install -y docker-ce docker-ce-cli containerd.io docker-buildx-plugin docker-compose-plugin

# 启动& 开机启动docker； enable + start 二合一
systemctl enable docker --now

# 配置加速
sudo mkdir -p /etc/docker
sudo tee /etc/docker/daemon.json <<-'EOF'
{
  "registry-mirrors": ["https://mirror.ccs.tencentyun.com"]
}
EOF
sudo systemctl daemon-reload
sudo systemctl restart docker
```

2. 命令

Shell

```shell
#查看运行中的容器
docker ps
#查看所有容器
docker ps -a
#搜索镜像
docker search nginx
#下载镜像
docker pull nginx
#下载指定版本镜像
docker pull nginx:1.26.0
#查看所有镜像
docker images
#删除指定id的镜像
docker rmi e784f4560448


#运行一个新容器
docker run nginx
#停止容器
docker stop keen_blackwell
#启动容器
docker start 592
#重启容器
docker restart 592
#查看容器资源占用情况
docker stats 592
#查看容器日志
docker logs 592
#删除指定容器
docker rm 592
#强制删除指定容器
docker rm -f 592
# 后台启动容器
docker run -d --name mynginx nginx
# 后台启动并暴露端口
docker run -d --name mynginx -p 80:80 nginx
# 进入容器内部
docker exec -it mynginx /bin/bash

# 提交容器变化打成一个新的镜像
docker commit -m "update index.html" mynginx mynginx:v1.0
# 保存镜像为指定文件
docker save -o mynginx.tar mynginx:v1.0
# 删除多个镜像
docker rmi bde7d154a67f 94543a6c1aef e784f4560448
```

```
# 加载镜像
docker load -i mynginx.tar


# 登录 docker hub
docker login
# 重新给镜像打标签
docker tag mynginx:v1.0 leifengyang/mynginx:v1.0
# 推送镜像
docker push leifengyang/mynginx:v1.0
```

3. 存储

两种方式，注意区分：

- 目录挂载： `-v /app/nghtml:/usr/share/nginx/html`
- 卷映射： `-v ngconf:/etc/nginx`

Shell

```
docker run -d -p 99:80 \
-v /app/nghtml:/usr/share/nginx/html \
-v ngconf:/etc/nginx \
--name app03 \
nginx
```

4. 网络

创建自定义网络，实现主机名作为稳定域名访问。

4.1. Redis主从同步集群

```shell
#自定义网络
docker network create mynet
#主节点
docker run -d -p 6379:6379 \
-v /app/rd1:/bitnami/redis/data \
-e REDIS_REPLICATION_MODE=master \
-e REDIS_PASSWORD=123456 \
--network mynet --name redis01 \
bitnami/redis

#从节点
docker run -d -p 6380:6379 \
-v /app/rd2:/bitnami/redis/data \
-e REDIS_REPLICATION_MODE=slave \
-e REDIS_MASTER_HOST=redis01 \
-e REDIS_MASTER_PORT_NUMBER=6379 \
-e REDIS_MASTER_PASSWORD=123456 \
-e REDIS_PASSWORD=123456 \
--network mynet --name redis02 \
bitnami/redis
```

4.2. 启动MySQL

```shell
docker run -d -p 3306:3306 \
-v /app/myconf:/etc/mysql/conf.d \
-v /app/mydata:/var/lib/mysql \
-e MYSQL_ROOT_PASSWORD=123456 \
mysql:8.0.37-debian
```

5. Docker Compose

5.1. 命令式安装

```shell
#创建网络
docker network create blog

#启动mysql
docker run -d -p 3306:3306 \
-e MYSQL_ROOT_PASSWORD=123456 \
-e MYSQL_DATABASE=wordpress \
-v mysql-data:/var/lib/mysql \
-v /app/myconf:/etc/mysql/conf.d \
--restart always --name mysql \
--network blog \
mysql:8.0

#启动wordpress
docker run -d -p 8080:80 \
-e WORDPRESS_DB_HOST=mysql \
-e WORDPRESS_DB_USER=root \
-e WORDPRESS_DB_PASSWORD=123456 \
-e WORDPRESS_DB_NAME=wordpress \
-v wordpress:/var/www/html \
--restart always --name wordpress-app \
--network blog \
wordpress:latest
```

5.2. compose.yaml

YAML

```yaml
name: myblog
services:
  mysql:
    container_name: mysql
    image: mysql:8.0
    ports:
      - "3306:3306"
    environment:
      - MYSQL_ROOT_PASSWORD=123456
      - MYSQL_DATABASE=wordpress
    volumes:
      - mysql-data:/var/lib/mysql
      - /app/myconf:/etc/mysql/conf.d
    restart: always
    networks:
      - blog

  wordpress:
    image: wordpress
    ports:
      - "8080:80"
    environment:
      WORDPRESS_DB_HOST: mysql
      WORDPRESS_DB_USER: root
      WORDPRESS_DB_PASSWORD: 123456
      WORDPRESS_DB_NAME: wordpress
    volumes:
      - wordpress:/var/www/html
    restart: always
    networks:
      - blog
    depends_on:
      - mysql

volumes:
  mysql-data:
  wordpress:

networks:
  blog:
```

5.3. 特性

- 增量更新
  - 修改 Docker Compose 文件。重新启动应用。只会触发修改项的重新启动。
- 数据不删
  - 默认就算down了容器，所有挂载的卷不会被移除。比较安全

6. Dockerfile

app.jar

```Shell
FROM openjdk:17

LABEL author=leifengyang

COPY app.jar /app.jar

EXPOSE 8080

ENTRYPOINT ["java","-jar","/app.jar"]
```

7. 附录 – 一键安装超多中间件

```shell
#Disable memory paging and swapping performance
sudo swapoff -a

# Edit the sysctl config file
sudo vi /etc/sysctl.conf

# Add a line to define the desired value
# or change the value if the key exists,
# and then save your changes.
vm.max_map_count=262144

# Reload the kernel parameters using sysctl
sudo sysctl -p

# Verify that the change was applied by checking the value
cat /proc/sys/vm/max_map_count
```

7.1. yaml

> 注意：将下面文件中 `kafka` 的 `119.45.147.122` 改为你自己的服务器IP。
>
> 准备一个 `compose.yaml` 文件，内容如下：

YAML

```yaml
name: devsoft
services:
  redis:
    image: bitnami/redis:latest
    restart: always
    container_name: redis
    environment:
      - REDIS_PASSWORD=123456
    ports:
      - '6379:6379'
    volumes:
      - redis-data:/bitnami/redis/data
      - redis-conf:/opt/bitnami/redis/mounted-etc

  mysql:
    image: mysql:8.0.31
    restart: always
    container_name: mysql
    environment:
      - MYSQL_ROOT_PASSWORD=123456
    ports:
      - '3306:3306'
      - '33060:33060'
    volumes:
      - mysql-conf:/etc/mysql/conf.d
      - mysql-data:/var/lib/mysql

  rabbit:
    image: rabbitmq:3-management
    restart: always
    container_name: rabbitmq
    ports:
      - "5672:5672"
      - "15672:15672"
    environment:
      - RABBITMQ_DEFAULT_USER=rabbit
      - RABBITMQ_DEFAULT_PASS=rabbit
      - RABBITMQ_DEFAULT_VHOST=dev
    volumes:
      - rabbit-data:/var/lib/rabbitmq
      - rabbit-app:/etc/rabbitmq
  opensearch-node1:
    image: opensearchproject/opensearch:2.13.0
    container_name: opensearch-node1
    environment:
```

```
      - cluster.name=opensearch-cluster # Name the cluster
      - node.name=opensearch-node1 # Name the node that will run in this container
      - discovery.seed_hosts=opensearch-node1,opensearch-node2 # Nodes to look for when discovering the cluster
      - cluster.initial_cluster_manager_nodes=opensearch-node1,opensearch-node2 # Nodes eligibile to serve as cluster manager
      - bootstrap.memory_lock=true # Disable JVM heap memory swapping
      - "OPENSEARCH_JAVA_OPTS=-Xms512m -Xmx512m" # Set min and max JVM heap sizes to at least 50% of system RAM
      - "DISABLE_INSTALL_DEMO_CONFIG=true" # Prevents execution of bundled demo script which installs demo certificates and security configurations to OpenSearch
      - "DISABLE_SECURITY_PLUGIN=true" # Disables Security plugin
    ulimits:
      memlock:
        soft: -1 # Set memlock to unlimited (no soft or hard limit)
        hard: -1
      nofile:
        soft: 65536 # Maximum number of open files for the opensearch user - set to at least 65536
        hard: 65536
    volumes:
      - opensearch-data1:/usr/share/opensearch/data # Creates volume called opensearch-data1 and mounts it to the container
    ports:
      - 9200:9200 # REST API
      - 9600:9600 # Performance Analyzer

  opensearch-node2:
    image: opensearchproject/opensearch:2.13.0
    container_name: opensearch-node2
    environment:
      - cluster.name=opensearch-cluster # Name the cluster
      - node.name=opensearch-node2 # Name the node that will run in this container
      - discovery.seed_hosts=opensearch-node1,opensearch-node2 # Nodes to look for when discovering the cluster
      - cluster.initial_cluster_manager_nodes=opensearch-node1,opensearch-node2 # Nodes eligibile to serve as cluster manager
      - bootstrap.memory_lock=true # Disable JVM heap memory swapping
      - "OPENSEARCH_JAVA_OPTS=-Xms512m -Xmx512m" # Set min and max JVM heap sizes to at least 50% of system RAM
      - "DISABLE_INSTALL_DEMO_CONFIG=true" # Prevents execution of bundled demo script which installs demo certificates and security configurations to OpenSearch
      - "DISABLE_SECURITY_PLUGIN=true" # Disables Security plugin
    ulimits:
```

```
      memlock:
        soft: -1 # Set memlock to unlimited (no soft or hard limit)
        hard: -1
      nofile:
        soft: 65536 # Maximum number of open files for the opensearch user - set to at least 65536
        hard: 65536
    volumes:
      - opensearch-data2:/usr/share/opensearch/data # Creates volume called opensearch-data2 and mounts it to the container

  opensearch-dashboards:
    image: opensearchproject/opensearch-dashboards:2.13.0
    container_name: opensearch-dashboards
    ports:
      - 5601:5601 # Map host port 5601 to container port 5601
    expose:
      - "5601" # Expose port 5601 for web access to OpenSearch Dashboards
    environment:
      - 'OPENSEARCH_HOSTS=["http://opensearch-node1:9200","http://opensearch-node2:9200"]'
      - "DISABLE_SECURITY_DASHBOARDS_PLUGIN=true" # disables security dashboards plugin in OpenSearch Dashboards

  zookeeper:
    image: bitnami/zookeeper:3.9
    container_name: zookeeper
    restart: always
    ports:
      - "2181:2181"
    volumes:
      - "zookeeper_data:/bitnami"
    environment:
      - ALLOW_ANONYMOUS_LOGIN=yes

  kafka:
    image: 'bitnami/kafka:3.4'
    container_name: kafka
    restart: always
    hostname: kafka
    ports:
      - '9092:9092'
      - '9094:9094'
    environment:
      - KAFKA_CFG_NODE_ID=0
      - KAFKA_CFG_PROCESS_ROLES=controller,broker
      - KAFKA_CFG_LISTENERS=PLAINTEXT://:9092,CONTROLLER://:9093,EXTERNAL://0.0.0.0:9094
```

```
      - KAFKA_CFG_ADVERTISED_LISTENERS=PLAINTEXT://kafka:9092,EXTERNAL://119.45.147.122:9094
      - KAFKA_CFG_LISTENER_SECURITY_PROTOCOL_MAP=CONTROLLER:PLAINTEXT,EXTERNAL:PLAINTEXT,PLAINTEXT:PLAINTEXT
      - KAFKA_CFG_CONTROLLER_QUORUM_VOTERS=0@kafka:9093
      - KAFKA_CFG_CONTROLLER_LISTENER_NAMES=CONTROLLER
      - ALLOW_PLAINTEXT_LISTENER=yes
      - "KAFKA_HEAP_OPTS=-Xmx512m -Xms512m"
    volumes:
      - kafka-conf:/bitnami/kafka/config
      - kafka-data:/bitnami/kafka/data
  kafka-ui:
    container_name: kafka-ui
    image: provectuslabs/kafka-ui:latest
    restart: always
    ports:
      - 8080:8080
    environment:
      DYNAMIC_CONFIG_ENABLED: true
      KAFKA_CLUSTERS_0_NAME: kafka-dev
      KAFKA_CLUSTERS_0_BOOTSTRAPSERVERS: kafka:9092
    volumes:
      - kafkaui-app:/etc/kafkaui

  nacos:
    image: nacos/nacos-server:v2.3.1
    container_name: nacos
    ports:
      - 8848:8848
      - 9848:9848
    environment:
      - PREFER_HOST_MODE=hostname
      - MODE=standalone
      - JVM_XMX=512m
      - JVM_XMS=512m
      - SPRING_DATASOURCE_PLATFORM=mysql
      - MYSQL_SERVICE_HOST=nacos-mysql
      - MYSQL_SERVICE_DB_NAME=nacos_devtest
      - MYSQL_SERVICE_PORT=3306
      - MYSQL_SERVICE_USER=nacos
      - MYSQL_SERVICE_PASSWORD=nacos
      - MYSQL_SERVICE_DB_PARAM=characterEncoding=utf8&connectTimeout=1000&socketTimeout=3000&autoReconnect=true&useUnicode=true&useSSL=false&serverTimezone=Asia/Shanghai&allowPublicKeyRetrieval=true
      - NACOS_AUTH_IDENTITY_KEY=2222
      - NACOS_AUTH_IDENTITY_VALUE=2xxx
      - NACOS_AUTH_TOKEN=SecretKey012345678901234567890123456789012345678901234567890123456789
```

```
      - NACOS_AUTH_ENABLE=true
    volumes:
      - /app/nacos/standalone-logs/:/home/nacos/logs
    depends_on:
      nacos-mysql:
        condition: service_healthy
  nacos-mysql:
    container_name: nacos-mysql
    build:
      context: .
      dockerfile_inline: |
        FROM mysql:8.0.31
        ADD https://raw.githubusercontent.com/alibaba/nacos/2.3.2/distribution/conf/mysql-schema.sql /docker-entrypoint-initdb.d/nacos-mysql.sql
        RUN chown -R mysql:mysql /docker-entrypoint-initdb.d/nacos-mysql.sql
        EXPOSE 3306
        CMD ["mysqld", "--character-set-server=utf8mb4", "--collation-server=utf8mb4_unicode_ci"]
    image: nacos/mysql:8.0.30
    environment:
      - MYSQL_ROOT_PASSWORD=root
      - MYSQL_DATABASE=nacos_devtest
      - MYSQL_USER=nacos
      - MYSQL_PASSWORD=nacos
      - LANG=C.UTF-8
    volumes:
      - nacos-mysqldata:/var/lib/mysql
    ports:
      - "13306:3306"
    healthcheck:
      test: [ "CMD", "mysqladmin" ,"ping", "-h", "localhost" ]
      interval: 5s
      timeout: 10s
      retries: 10
  prometheus:
    image: prom/prometheus:v2.52.0
    container_name: prometheus
    restart: always
    ports:
      - 9090:9090
    volumes:
      - prometheus-data:/prometheus
      - prometheus-conf:/etc/prometheus

  grafana:
    image: grafana/grafana:10.4.2
    container_name: grafana
```

```
    restart: always
    ports:
      - 3000:3000
    volumes:
      - grafana-data:/var/lib/grafana

volumes:
  redis-data:
  redis-conf:
  mysql-conf:
  mysql-data:
  rabbit-data:
  rabbit-app:
  opensearch-data1:
  opensearch-data2:
  nacos-mysqldata:
  zookeeper_data:
  kafka-conf:
  kafka-data:
  kafkaui-app:
  prometheus-data:
  prometheus-conf:
  grafana-data:
```

7.2. 启动

Shell

```
# 在 compose.yaml 文件所在的目录下执行
docker compose up -d
# 等待启动所有容器
```

> tip：如果重启了服务器，可能有些容器会启动失败。再执行一遍 `docker compose up -d` 即可。所有程序都可运行成功，并且不会丢失数据。请放心使用。

7.3. 访问

- zookeeper可视化工具下载：
  - https://github.com/vran-dev/PrettyZoo/releases/download/v2.1.1/prettyZoo-win.zip

- redis可视化工具下载：
  - https://github.com/qishibo/AnotherRedisDesktopManager/releases/download/v1.6.4/Another-Redis-Desktop-Manager.1.6.4.exe

组件（容器名）	介绍	访问地址	账号/密码	特性
Redis(redis)	k-v 库	你的ip:6379	单密码模式：123456	已开启AOF
MySQL(mysql)	数据库	你的ip:3306	root/123456	默认utf8mb4字符集
Rabbit(rabbit)	消息队列	你的ip:15672	rabbit/rabbit	暴露5672和15672端口
OpenSearch(opensearch-node1/2)	检索引擎	你的ip:9200		内存512mb；两个节点
opensearch-dashboards	search可视化	你的ip:5601		
Zookeeper(zookeeper)	分布式协调	你的ip:2181		允许匿名登录
kafka(kafka)	消息队列	你的ip:9092 外部访问:9094		占用内存512mb
kafka-ui(kafka-ui)	kafka可视化	你的ip:8080		
nacos(nacos)	注册/配置中心	你的ip:8848	nacos/nacos	持久化数据到MySQL
nacos-mysql(nacos-mysql)	nacos配套数据库	你的ip:13306	root/root	
prometheus(prometheus)	时序数据库	你的ip:9090		
grafana(grafana)		你的ip:3000	admin/admin	