Docker部署MySQL、Redis、Kafka、ES、Kibana

无奈何杨 2025-03-31 333 阅读5分钟 关注

告别排队，Trae订阅上线，首月Pro仅$3，开发无忧

Docker

`Docker` 的基础概念和安装就不多讲了，参考官网学习就行。

www.docker.com/

docs.docker.com/engine/inst...

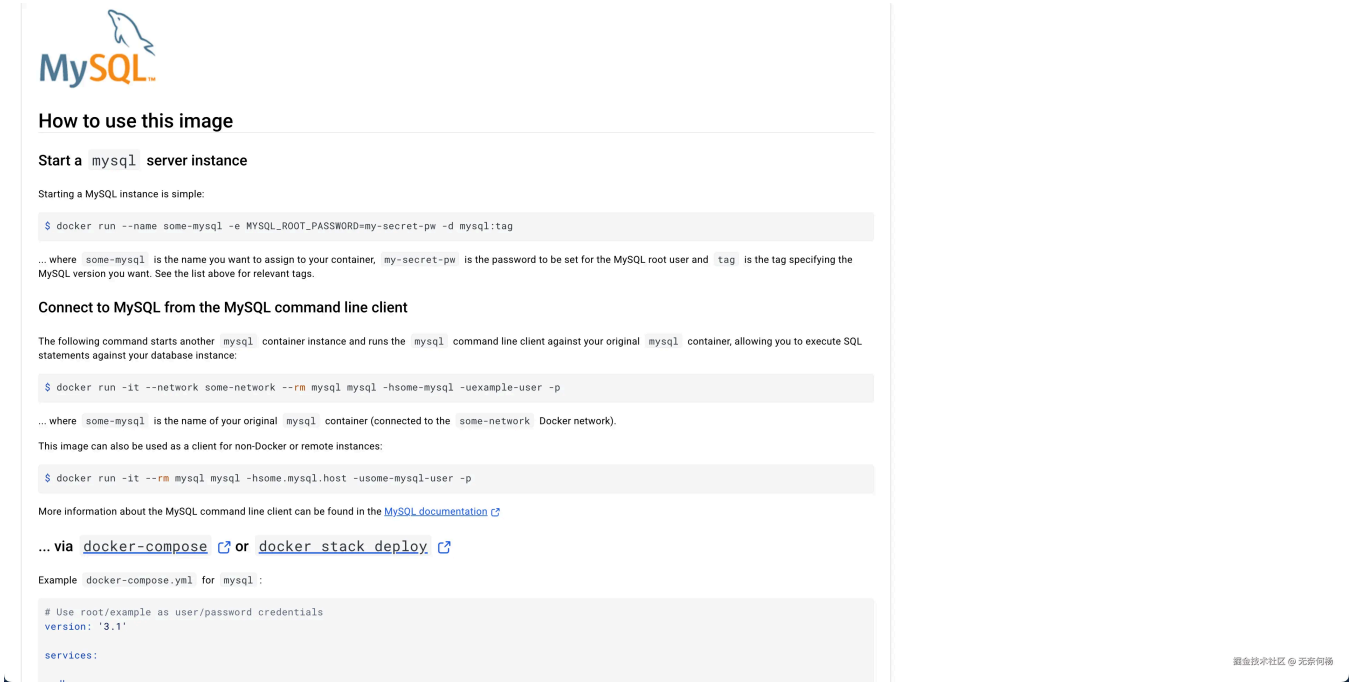
镜像仓库

hub.docker.com/explore

hub.docker.com/search?badg...

hub.docker已经提供了非常多的镜像，一般情况来讲使用官方镜像就好了。

如：hub.docker.com/_/mysql



MySQL

How to use this image

Start a `mysql` server instance

Starting a MySQL instance is simple:

```
$ docker run --name some-mysql -e MYSQL_ROOT_PASSWORD=my-secret-pw -d mysql:tag
```

... where `some-mysql` is the name you want to assign to your container, `my-secret-pw` is the password to be set for the MySQL root user and `tag` is the tag specifying the MySQL version you want. See the list above for relevant tags.

Connect to MySQL from the MySQL command line client

The following command starts another `mysql` container instance and runs the `mysql` command line client against your original `mysql` container, allowing you to execute SQL statements against your database instance:

```
$ docker run -it --network some-network --rm mysql mysql -hsome-mysql -uexample-user -p
```

... where `some-mysql` is the name of your original `mysql` container (connected to the `some-network` Docker network).

This image can also be used as a client for non-Docker or remote instances:

```
$ docker run -it --rm mysql mysql -hsome.mysql.host -usome-mysql-user -p
```

More information about the MySQL command line client can be found in the MySQL documentation

... via `docker-compose` or `docker stack deploy`

Example `docker-compose.yml` for `mysql`:

```
# Use root/example as user/password credentials
version: '3.1'

services:
```

掘金技术社区 @ 无奈何杨

官方写的也相当明白，如何使用镜像，运行容器可以配置那些参数，日志等

还包括进入容器执行命名、查看文件等等

Docker Compose

Docker Compose 相较于单独使用 `docker run` 命令有很多优势，特别是在管理多容器应用时。

1. 简化复杂环境的配置

- **多容器管理**：当您的应用程序由多个服务组成（如前端、后端、数据库等），使用 `docker run` 需要为每个服务单独启动容器，并手动配置它们之间的网络和依赖关系。而 `Docker Compose` 可以通过一个简单的 `docker-compose.yml` 文件定义所有服务及其配置。
- **统一配置文件**：所有服务的配置都集中在一个 `YAML` 文件中，易于阅读和维护。

2. 自动化服务依赖

- **自动处理依赖**：在 `docker-compose.yml` 中可以指定服务之间的依赖关系。例如，您可以设置 MySQL 服务必须在应用服务之前启动。`Docker Compose` 会确保这些依赖关系得到满足。
- **重启策略**：可以定义重启策略来保证服务的高可用性，比如某个服务失败后自动重启。

3. 网络配置简化

- **内置网络管理**： `Docker Compose` 自动创建一个默认网络供所有服务使用，使得容器间通信变得非常简单。您不需要手动创建和管理网络。
- **自定义网络**：虽然可以手动创建网络并在 `docker run` 中指定，但 `Docker Compose` 提供了更简洁的方式来自定义网络。

4. 环境变量管理

- `.env` **文件支持**： `Docker Compose` 支持从 `.env` 文件加载环境变量，这使得管理和切换不同环境下的配置更加容易。
- **环境变量文件**：还可以通过 `env_file` 指令引用外部环境变量文件，进一步增强灵活性。

5. 卷和绑定挂载的便捷配置

- **持久化存储**：在 `Docker Compose` 文件中，可以通过简单的声明来配置卷或绑定挂载，从而轻松实现数据持久化。
- **一致的开发与生产环境**：可以在不同环境中复用相同的配置，只需调整少量参数即可适应不同的部署场景。

6. 命令行简化

- **单命令操作多个容器**：使用 `docker-compose up` 即可一次性启动所有定义的服务，而无需分别执行 `docker run` 命令。
- **便捷的生命周期管理**：除了启动服务外，还可以使用 `docker-compose down` 来停止并移除所有相关容器、网络和卷。

7. 版本控制友好

- **YAML 格式便于版本控制**：将 `docker-compose.yml` 文件纳入版本控制系统（如 `Git` ）中，便于团队协作和历史记录追踪。
- **回滚方便**：如果需要恢复到之前的配置版本，只需切换到相应的分支或标签即可。

8. 扩展性和伸缩性

- **轻松扩展服务实例数**：通过 `docker-compose scale` 或者在 `docker-compose.yml` 中定义 `deploy` 部分，可以轻松地增加或减少服务实例的数量。
- **负载均衡**：对于 `Web` 应用等场景， `Docker Compose` 能够结合 `Docker Swarm` 实现负载均衡。

示例对比

假设我们要部署一个包含 MySQL 和 Redis 的应用：

使用 `docker run`

bash 体验AI代码助手 复制代码

```bash
# 启动 MySQL 容器
docker run --name some-mysql -e MYSQL_ROOT_PASSWORD=my-secret-pw -d mysql:tag

# 启动 Redis 容器
docker run --name some-redis -d redis:alpine
```

使用 Docker Compose

yaml 体验AI代码助手 复制代码

```yaml
version: '3'
services:
  db:
    image: mysql:tag
    environment:
      MYSQL_ROOT_PASSWORD: my-secret-pw
  redis:
    image: redis:alpine
```

只需运行 `docker-compose up -d` 即可同时启动这两个服务，并且它们之间能够自动发现对方（如果在同一网络下）。

`Docker Compose` 提供了一种更为高效、灵活的方式来管理和部署复杂的多容器应用，特别适合开发、测试以及小规模生产环境中的快速部署需求。

示例Compose

github.com/wnhyang/coo...

也是此项目的部署方式

.env

shell 体验AI代码助手 复制代码

```shell
mysql_root_password=mysql_password
mysql_database=mysql_database
mysql_user=mysql_user
mysql_password=mysql_password
redis_password=redis_password
elasticsearch_password=elasticsearch_password
kibana_password=kibana_password
kibana_url=http://localhost:5601
coolguard_image=coolguard_image
```

docker-compose.yml

yaml 体验AI代码助手 复制代码

```yaml
services:
  mysql:
    image: mysql:8.0.36
    container_name: mysql
    volumes:
      - mysqldata:/var/lib/mysql
    ports:
      - "3306:3306"
    environment:
      MYSQL_ROOT_HOST: 'localhost'
      TZ: Asia/Shanghai
      MYSQL_ROOT_PASSWORD: ${mysql_root_password}
      MYSQL_DATABASE: ${mysql_database}
      MYSQL_USER: ${mysql_user}
      MYSQL_PASSWORD: ${mysql_password}
    networks:
      - custom_network

  redis:
    image: redis:7.2.7-alpine
    container_name: redis
    command: redis-server --requirepass ${redis_password}
    volumes:
      - redisdata:/data
    ports:
      - "6379:6379"
    environment:
      TZ: Asia/Shanghai
    networks:
      - custom_network

  kafka:
```

```yaml
    image: apache/kafka:3.7.0
    container_name: kafka
    volumes:
      - kafkadata:/var/lib/kafka/data
    ports:
      - "9092:9092"
      - "9093:9093"
    environment:
      KAFKA_NODE_ID: 1
      KAFKA_LOG_DIRS: /var/lib/kafka/data
      KAFKA_METADATA_LOG_REPLICATION_FACTOR: 1
      KAFKA_DEFAULT_REPLICATION_FACTOR: 1
      KAFKA_PROCESS_ROLES: broker,controller
      KAFKA_LISTENERS: PLAINTEXT://0.0.0.0:9092,CONTROLLER://0.0.0.0:9093
      KAFKA_ADVERTISED_LISTENERS: PLAINTEXT://kafka:9092
      KAFKA_CONTROLLER_LISTENER_NAMES: CONTROLLER
      KAFKA_LISTENER_SECURITY_PROTOCOL_MAP: CONTROLLER:PLAINTEXT,PLAINTEXT:PLAINTEXT
      KAFKA_CONTROLLER_QUORUM_VOTERS: 1@kafka:9093
      KAFKA_OFFSETS_TOPIC_REPLICATION_FACTOR: 1
      KAFKA_TRANSACTION_STATE_LOG_REPLICATION_FACTOR: 1
      KAFKA_TRANSACTION_STATE_LOG_MIN_ISR: 1
      KAFKA_GROUP_INITIAL_REBALANCE_DELAY_MS: 0
      KAFKA_NUM_PARTITIONS: 3
      TZ: Asia/Shanghai
    networks:
      - custom_network

  elasticsearch:
    image: docker.elastic.co/elasticsearch/elasticsearch:8.17.3
    container_name: elasticsearch
    volumes:
      - esdata:/usr/share/elasticsearch/data
    ports:
      - "9200:9200"
      - "9300:9300"
    environment:
      ES_JAVA_OPTS: "-Xms1g -Xmx1g"
      discovery.type: single-node
      ELASTIC_PASSWORD: ${elasticsearch_password}
      xpack.security.enabled: "true"
      xpack.security.http.ssl.enabled: "false"
      xpack.security.transport.ssl.enabled: "false"
      TZ: Asia/Shanghai
    networks:
      - custom_network

  kibana:
    depends_on:
      - elasticsearch
    image: docker.elastic.co/kibana/kibana:8.17.3
```

```
    container_name: kibana
    volumes:
      - kibanadata:/usr/share/kibana/data
    ports:
      - "5601:5601"
    environment:
      ELASTICSEARCH_HOSTS: http://elasticsearch:9200
      ELASTICSEARCH_USERNAME: kibana
      ELASTICSEARCH_PASSWORD: ${kibana_password}
      SERVER_PUBLICBASEURL: ${kibana_url}
      XPACK_SECURITY_ENABLED: "true"
      XPACK_SECURITY_HTTP_SSL_ENABLED: "false"
      TZ: Asia/Shanghai
    networks:
      - custom_network
  coolguard:
    image: ${coolguard_image}
    container_name: coolguard
    volumes:
      - /docker/coolguard/logs:/coolguard/logs
    ports:
      - "8081:8081"
    environment:
      - "SPRING_PROFILES_ACTIVE=demo"
    networks:
      - custom_network

volumes:
  mysqldata:
    driver: local
  redisdata:
    driver: local
  kafkadata:
    driver: local
  esdata:
    driver: local
  kibanadata:
    driver: local

networks:
  custom_network:
    driver: bridge
```

运行

在 `docker-compose.yml` 同目录执行

shell 体验AI代码助手 复制代码

```shell
# 单独运行
docker compose up -d <service_name>
# 一起运行
docker compose up -d
```

或指定

- **-f 参数** 指定文件的路径
- **-p 参数** 指定项目名，默认不指定时使用当前目录名作为项目的名称

shell 体验AI代码助手 复制代码

```shell
docker-compose -f /home/user/projects/myapp/docker-compose.yml -p myapp up -d mysql redi
```

标签： 后端 话题： 每天一个知识点

本文收录于以下专栏

1 / 2

软件&工具 专栏目录

分享软件工具

6 订阅 · 9 篇文章

订阅

上一篇 免费使用满血版DeepSeek-R1...

下一篇 开源项目更新到个人仓库并保...

评论 0

登录 / 注册 即可发布评论！

暂无评论数据

目录

收起

相关推荐

为你推荐

使用Windows电脑快速入门Docker

蒸汽蘑菇 1年前 1.4k 5 评论 Docker 前端 容器

Docker入门系列——DockerFile的使用

叶知秋水 6月前 294 6 评论 前端 Docker

docker容器由浅入深解析

lcomedy喜剧 3年前 365 2 评论 Docker

Docker快速入门

王延领 3年前 853 3 评论 Docker

docker入门学习一

5大大大大雄 3年前 400 1 评论 Docker

Docker 系列 - 02 - 入门 & Nginx 服务 & Docker 概念【合集】

jsliang 3年前 1.7k 9 2 Docker Nginx

Docker基础使用教程

热心市民余生 3年前 392 2 1 Docker 后端

Docker入门之docker基本命令

讷言丶 4月前 166 1 评论 后端 运维 Docker

docker技术的安装与应用

chensi2113 3年前 512 点赞 评论 Docker 后端

linux服务器使用docker部署Vue+Egg.js项目

curtain 3月前 71 1 评论 Docker 容器

Docker入门讲解

暗余 4年前 1.3k 15 1 Docker

深入浅出Docker应用-Docker Compose实战

Lunaticskytql 1年前 318 点赞 评论 Docker

「Docker 系列」- Docker 的安装与常见命令的使用

你算哪块小蛋糕 3年前 297 1 评论 Docker 后端

如何使用docker + nginx来部署前端项目

sensFeng 6月前 1.5k 26 2 前端 Linux Nginx

Docker实战应用

Jason学长 4年前 937 5 评论 Docker