基于Docker的三层Web应用容器化部署项目汇报

------解决环境一致性问题，实现多服务快速编排

汇报人:张挺

1. 项目背景与目标

为什么需要容器化部署？

环境一致性差

某电商企业部署时，开发环境运行正常的代码在生产环境频繁报错，排查发现是依赖库版本差异导致，修复耗时3天。

部署效率低下

某金融系统采用手动部署，单节点部署需执行20+步骤，全量更新耗时约4小时，期间业务中断风险高。

资源利用率低

某政务平台为保证峰值性能，为各应用单独配置服务器，平均CPU使用率仅15%-20%，硬件资源浪费严重。

- 容器化封装三层应用：前端（Nginx）、后端（Node.js）、数据库（MySQL）
- 使用 docker-compose 实现一键编排与服务依赖管理
- 确保服务间网络互通与数据库数据持久化

2.关键概念与原理

镜像分层存储机制

Docker采用UnionFS实现镜像分层，如Nginx镜像由基础层+应用层构成，修改仅增量更新，提升部署效率30%。

容器隔离与资源限制

通过Linux Namespaces和Cgroups实现隔离，阿里电商平台用此技术将单服务器容器密度提升至传统虚拟机的5倍。

多阶段构建优化

以Java项目为例，先用Maven镜像编译打包，再将JAR包复制到轻量Alpine镜像，最终镜像体积减少60%以上。

核心功能与配置文件

Docker Compose通过YAML文件定义多容器应用，如定义web、db、redis服务，指定镜像、端口映射和依赖关系，简化部署流程。

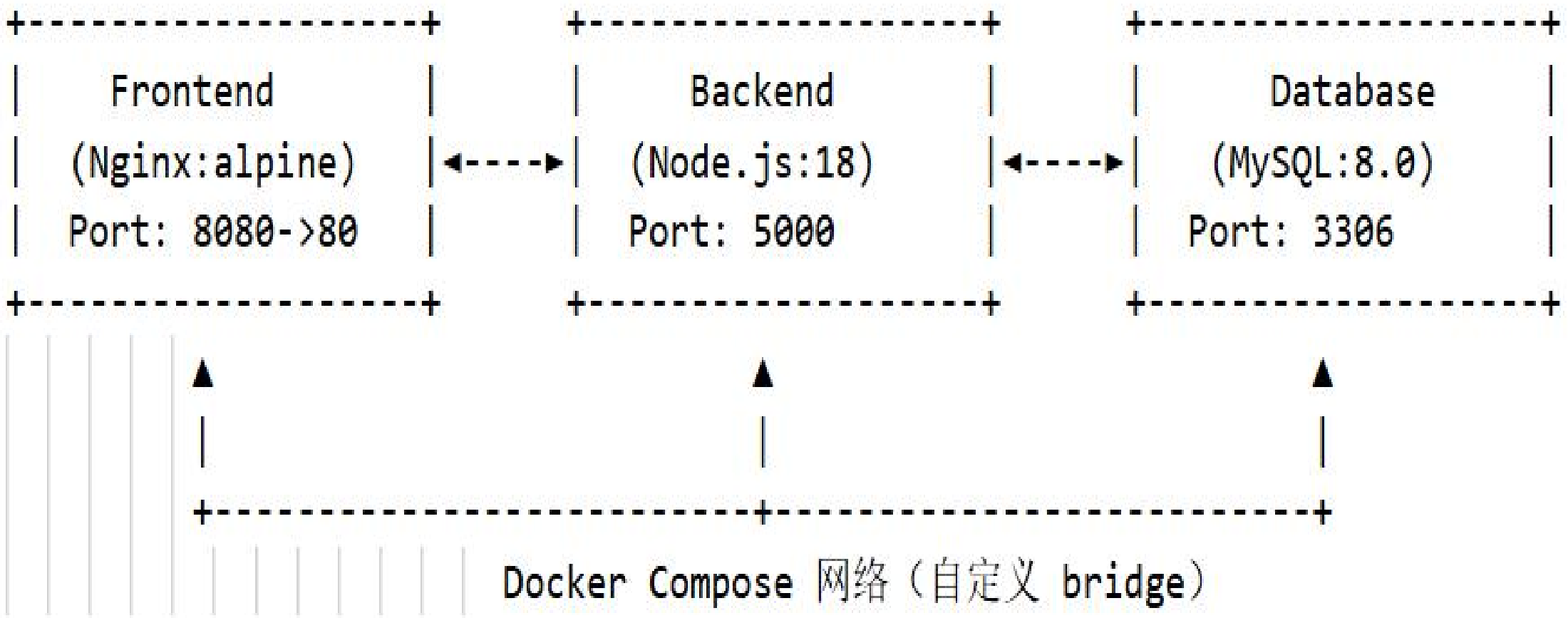
进程管理与资源隔离

课程中学习的Linux Namespaces技术，如PID、Mount命名空间，是Docker容器实现进程隔离的核心，对应Docker run时的--pid=host等参数配置。

数据持久化与存储管理

课程涉及的MySQL数据持久化方案，对应Docker通过-v /host/data:/var/lib/mysql挂载宿主机目录，确保容器重启数据不丢失。

3.架构拓扑


Frontend
(Nginx:alpine)
Port: 8080->80
Backend
(Node.js:18)
Port: 5000
Database
(MySQL:8.0)
Port: 3306
Docker Compose 网络（自定义 bridge）

04

关键实现

```
fb@fb-virtual-machine:~/my-app/frontend$ cat Dockerfile
FROM nginx:alpine

COPY index.html /usr/share/nginx/html/

EXPOSE 80
fb@fb-virtual-machine:~/my-app/frontend$
```

```
fb@fb-virtual-machine:~/my-app/backend$ cat Dockerfile
FROM python:3.9-slim

WORKDIR /app

RUN pip install --no-cache-dir flask==2.3.3 pymysql==1.1.0 python-dotenv==1.0.0

COPY requirements.txt .
COPY app.py .

EXPOSE 5000

CMD ["python", "app.py"]
fb@fb-virtual-machine:~/my-app/backend$
```

```
fb@fb-virtual-machine:~/my-app$ ./scripts/start.sh
🚀 启动三层应用...
[+] Running 3/4
 ⋮ Network my-app_default    Created                                    11.8s
 ✔ Container myapp-db        Healthy                                     5.9s
 ✔ Container myapp-backend   Healthy                                    11.5s
 ✔ Container myapp-frontend  Started                                    11.6s
⏳ 等待服务启动...
📊 服务状态:
NAME             IMAGE             COMMAND                  SERVICE    CREATED
         STATUS                    PORTS
myapp-backend    my-app-backend    "python app.py"          backend    26 secon
ds ago   Up 20 seconds (healthy)   0.0.0.0:5000->5000/tcp, :::5000->5000/tcp
myapp-db         mariadb:10.6      "docker-entrypoint.s…"   db         26 second
s ago    Up 26 seconds (healthy)   0.0.0.0:3306->3306/tcp, :::3306->3306/tcp
myapp-frontend   my-app-frontend   "/docker-entrypoint.…"   frontend   26 second
s ago    Up 15 seconds             0.0.0.0:8080->80/tcp, :::8080->80/tcp

✅ 应用已启动!
🌐 前端访问: http://localhost:8080
🔧 后端 API: http://localhost:5000
🗄️数据库端口: 3306

📝 查看日志: docker-compose logs -f
🛑 停止服务: ./scripts/stop.sh
```

```
fb@fb-virtual-machine:~/my-app$ cat docker-compose.yml
version: '3.8'

services:
  db:
    image: mariadb:10.6
    container_name: myapp-db
    restart: no
    environment:
      MYSQL_ROOT_PASSWORD: password
      MYSQL_DATABASE: myapp
    volumes:
      - db_data:/var/lib/mysql
      - ./database/init.sql:/docker-entrypoint-initdb.d/init.sql
    ports:
      - "3306:3306"
    healthcheck:
      test: ["CMD", "mysqladmin", "ping", "-h", "localhost", "-u", "root", "-ppassword"]
      interval: 10s
      timeout: 5s
      retries: 5
      start_period: 60s

  backend:
    build:
      context: ./backend
      dockerfile: Dockerfile
    container_name: myapp-backend
    restart: no
    environment:
      DB_HOST: db
      DB_USER: root
      DB_PASSWORD: password
      DB_NAME: myapp
    ports:
      - "5000:5000"
    depends_on:
      db:
        condition: service_healthy
    healthcheck:
      test: ["CMD", "python", "-c", "import urllib.request; urllib.request.urlopen('http://localhost:5000/health', timeout=5).read()"]
      interval: 30s
      timeout: 10s
      retries: 3
      start_period: 50s

  frontend:
    build:
      context: ./frontend
      dockerfile: Dockerfile
```

文本编辑器

```
      dockerfile: Dockerfile
    container_name: myapp-frontend
    restart: no
    ports:
      - "8080:80"
    depends_on:
      backend:
        condition: service_healthy

volumes:
  db_data:
    driver: local
fb@fb-virtual-machine:~/my-app$
```

05

验证结果

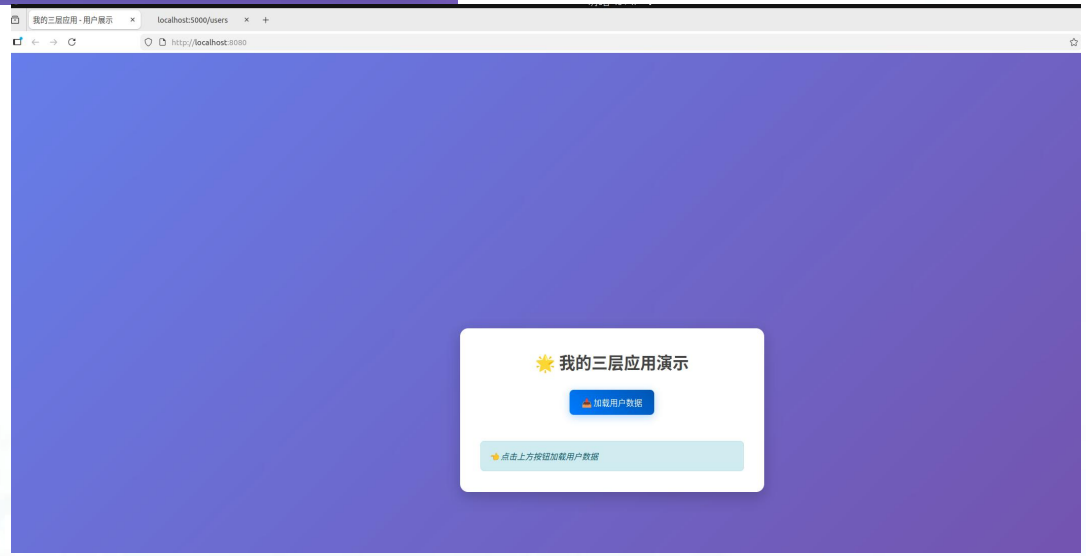
插入数据

```
fb@fb-virtual-machine:~/my-app$ curl -X POST http://localhost:5000/add_user \
  -H "Content-Type: application/json" \
  -d '{"name": "小夏", "email": "xiaoxia@example.com"}' > response.json
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100   166  100   116  100    50  17459   7525 --:--:-- --:--:-- --:--:-- 27666
```

查询数据

```
fb@fb-virtual-machine:~/my-app$ cat response.json
{
  "email": "xiaoxia@example.com",
  "message": "\u7528\u6237\u6dfb\u52a0\u6210\u529f",
  "name": "\u5c0f\u590f"
}
fb@fb-virtual-machine:~/my-app$
```

```
fb@fb-virtual-machine:~/my-app$ ./monitoring/log-collector.sh
📋 开始收集日志...
✅ 日志已保存到 ./logs 目录
📊 查看摘要: cat ./logs/summary.log
fb@fb-virtual-machine:~/my-app$ cat logs/summary.log
=== 日志摘要 2026年 01月 03日 星期六 18:51:21 CST ===
数据库日志行数: 26
后端日志行数: 19
前端日志行数: 24
fb@fb-virtual-machine:~/my-app$
```



6. 排错与复盘

问题1：容器网络通信异常

现象：docker pull nginx:alpine报错 context deadline exceeded。

定位过程：检查网络连通性（`ping hub.docker.com`正常），发现 DNS 解析失败（`nslookup registry-1.docker.io`超时）。

根因：系统 DNS 配置错误（systemd-resolved 服务异常）。

解决方法：禁用 systemd-resolved，配置静态 DNS（`/etc/resolv.conf`添加 `nameserver 8.8.8.8`）。

复盘与预防：网络问题优先排查 DNS 和防火墙，避免盲目更换镜像源。

问题2：构建 backend 时找不到 Dockerfile

现象：docker compose up报错 open Dockerfile: no such file or directory。

定位过程：检查 `docker-compose.yml`中 `build: ./backend`路径，发现 backend 目录下无 Dockerfile。

根因：手动创建 Dockerfile 时因权限问题失败（`frontend/Dockerfile`属主为 root）。

解决方法：修复文件属主（`sudo chown -R fb:fb ~/my-app`），重新创建 Dockerfile。

复盘与预防：文件操作时注意权限，避免使用 `sudo`导致属主混乱。

问题3：前端无法访问后端 API

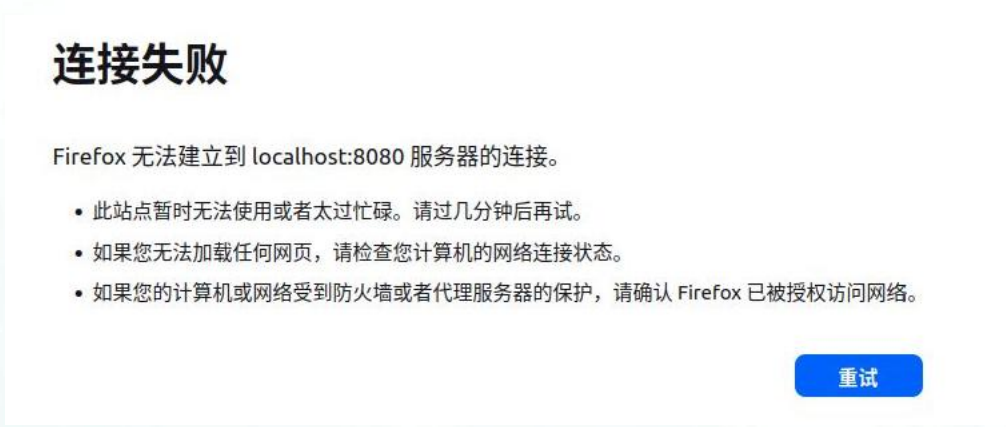
现象：浏览器控制台报错 `Failed to fetch /api/users`。

定位过程：检查容器网络（`docker network inspect app-network`），发现 backend 服务未正常启动

根因：后端依赖数据库，但数据库启动较慢导致 backend 连接失败。

解决方法：在 backend 服务中添加健康检查，或通过 `depends_on`配合 `condition: service_healthy`延迟启动（需 MySQL 配置健康检查）。

复盘与预防：多服务依赖需考虑服务启动顺序和健康状态，避免“假启动”。



```
fb@fb-virtual-machine:~/my-app$ ./scripts/start.sh
🚀 启动三层应用...
Traceback (most recent call last):
  File "/usr/local/bin/docker-compose", line 8, in <module>
    sys.exit(main())
  File "/usr/local/lib/python3.10/dist-packages/compose/cli/main.py", line 81, in main
    command_func()
  File "/usr/local/lib/python3.10/dist-packages/compose/cli/main.py", line 200, in perform_command
    project = project_from_options('.', options)
  File "/usr/local/lib/python3.10/dist-packages/compose/cli/command.py", line 60, in project_from_options
    return get_project(
  File "/usr/local/lib/python3.10/dist-packages/compose/cli/command.py", line 152, in get_project
    client = get_client(
  File "/usr/local/lib/python3.10/dist-packages/compose/cli/docker_client.py", line 41, in get_client
    client = docker_client(
  File "/usr/local/lib/python3.10/dist-packages/compose/cli/docker_client.py", line 124, in docker_client
    kwargs = kwargs_from_env(environment=environment, ssl_version=tls_version)
TypeError: kwargs_from_env() got an unexpected keyword argument 'ssl_version'
⏳ 等待服务启动...
📊 服务状态:
Traceback (most recent call last):
  File "/usr/local/bin/docker-compose", line 8, in <module>
    sys.exit(main())
  File "/usr/local/lib/python3.10/dist-packages/compose/cli/main.py", line 81, in main
    command_func()
  File "/usr/local/lib/python3.10/dist-packages/compose/cli/main.py", line 200, in perform_command
    project = project_from_options('.', options)
  File "/usr/local/lib/python3.10/dist-packages/compose/cli/command.py", line 60, in project_from_options
    return get_project(
  File "/usr/local/lib/python3.10/dist-packages/compose/cli/command.py", line 152, in get_project
    client = get_client(
  File "/usr/local/lib/python3.10/dist-packages/compose/cli/docker_client.py", line 41, in get_client
    client = docker_client(
  File "/usr/local/lib/python3.10/dist-packages/compose/cli/docker_client.py", line 124, in docker_client
    kwargs = kwargs_from_env(environment=environment, ssl_version=tls_version)
TypeError: kwargs_from_env() got an unexpected keyword argument 'ssl_version'
```

```
fb@fb-virtual-machine:~/my-app$ ./scripts/stop.sh
⏹止三层应用...
eback (most recent call last):
le "/usr/local/bin/docker-compose", line 8, in <module>
sys.exit(main())
le "/usr/local/lib/python3.10/dist-packages/compose/cli/main.py", line 81, in main
command_func()
le "/usr/local/lib/python3.10/dist-packages/compose/cli/main.py", line 200, in perform_command
project = project_from_options('.', options)
le "/usr/local/lib/python3.10/dist-packages/compose/cli/command.py", line 60, in project_from_options
return get_project(
le "/usr/local/lib/python3.10/dist-packages/compose/cli/command.py", line 152, in get_project
client = get_client(
le "/usr/local/lib/python3.10/dist-packages/compose/cli/docker_client.py", line 41, in get_client
client = docker_client(
le "/usr/local/lib/python3.10/dist-packages/compose/cli/docker_client.py", line 124, in docker_client
kwargs = kwargs_from_env(environment=environment, ssl_version=tls_version)
Error: kwargs_from_env() got an unexpected keyword argument 'ssl_version'
⏹用已停止!
数据已持久化保存在 Docker volume 中
```

连接失败

Firefox 无法建立到 localhost:5000 服务器的连接。

- 此站点暂时无法使用或者太过忙碌。请过几分钟后再试。
- 如果您无法加载任何网页，请检查您计算机的网络连接状态。
- 如果您的计算机或网络受到防火墙或者代理服务器的保护，请确认 Firefox 已被授权访问网络。

重试

7.分工与贡献

模块/任务	负责人
环境搭建和核心实现	付波
PPT和文档制作	夏晨博
答辩和验证	张挺
验证	叶子衡
纠错	陈炎浩

8. 总结与展望

当前不足

01

安全性薄弱

数据库密码明文存储在 `docker-compose.yml`中，生产环境下存在泄露风险；未配置容器健康检查，服务异常后无法自动恢复。

02

缺乏自动化能力

镜像构建、服务部署依赖手动执行命令，无 CI/CD 流程支持，迭代效率低。

03

监控与日志不完善

仅通过 `docker compose logs`查看日志，无可视化监控工具（如 Prometheus + Grafana），难以实时掌握服务状态。

未来改进

01

性能优化

通过 Docker 资源限制（`mem_limit`、`cpus`）避免单个服务占用过多资源；使用 Nginx 反向代理实现负载均衡（多实例部署）。

02

可靠性提升

为所有服务配置健康检查（`healthcheck`），结合 `restart: on-failure`实现异常自动重启；定期备份数据库数据卷。

03

自动化与 DevOps

编写 Shell 脚本或 Makefile 实现“一键构建-部署-测试”；接入 GitHub Actions 实现代码提交后自动构建镜像并部署。

04

安全加固

使用 Docker Secrets 管理敏感信息（如数据库密码）；限制容器权限（如 `--read-only`挂载只读文件系统）；配置防火墙规则仅开放必要端口。

THE END

谢谢