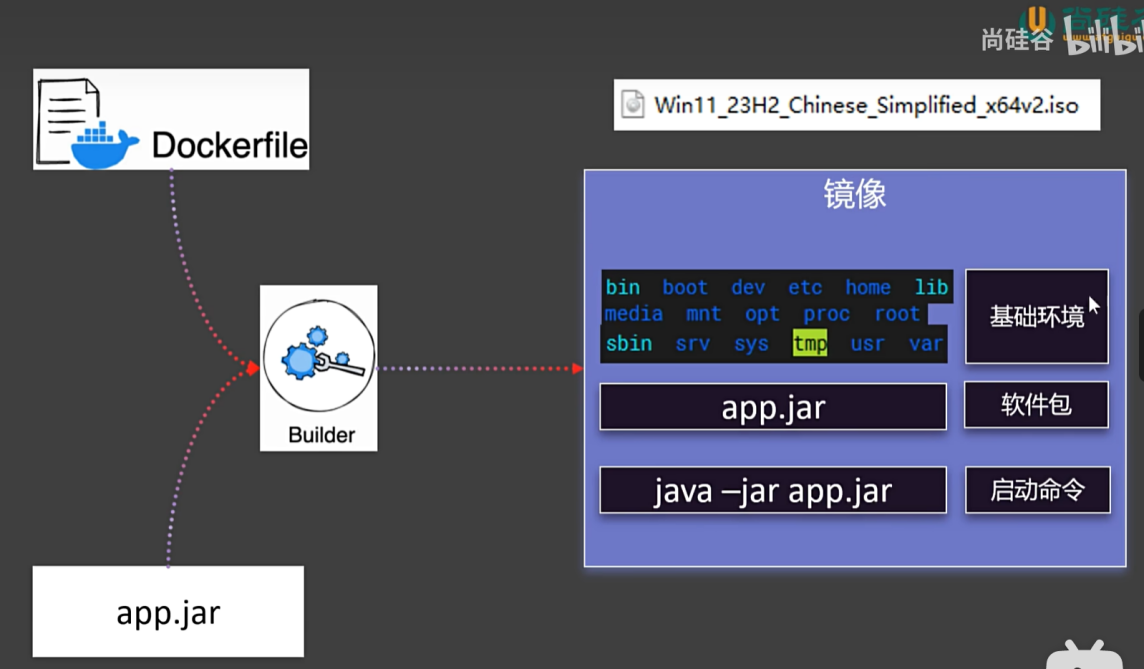
Dockerfile

图43



Dockerfile 是一个文本文件，其中包含了一系列指令（Instructions），这些指令定义了如何构建一个 Docker 镜像。通过 Dockerfile 可以自动化地创建自定义的 Docker 镜像。

图44



**Dockerfile 的基本结构**

以下是一个简单的 Dockerfile 示例及其说明：

# 选择基础镜像

FROM node:14

# 创建并设置工作目录

WORKDIR /app

# 将当前目录下的所有文件复制到工作目录

COPY . .

# 安装依赖

RUN npm install

# 暴露应用运行的端口

EXPOSE 3000

# 运行应用

CMD ["node", "app.js"]

**Dockerfile 指令说明**

FROM:指定构建镜像所基于的基础镜像。示例：FROM node:14 表示使用官方的 Node.js 14 版本镜像作为基础镜像。

WORKDIR:设置工作目录。所有后续的命令（如 RUN、CMD、COPY 等）都将相对于这个目录进行。示例：WORKDIR /app 表示设置 /app 目录为工作目录。

COPY: 将文件从主机复制到镜像中。示例：COPY . . 将当前目录下的所有文件复制到镜像的工作目录 /app 中。

RUN:在镜像构建过程中运行指定的命令。

示例：RUN npm install 在镜像中运行 npm install 命令来安装依赖。

EXPOSE: 声明容器在运行时监听的端口。示例：EXPOSE 3000 表示容器将监听 3000 端口。

CMD: 指定容器启动时要运行的命令。每个 Dockerfile 只能有一个 CMD 指令，如果指定了多个，则只有最后一个生效。示例：CMD ["node", "app.js"] 指定容器启动时运行 node app.js。

**创建和使用 Dockerfile**

1. 创建 Dockerfile: 在项目目录下创建一个名为 Dockerfile 的文件，并将上述内容粘贴进去。

2. 构建 Docker 镜像: 使用 docker build 命令构建 Docker 镜像。

docker build -t my-node-app .

-t 标志用于指定镜像的名称和标签。my-node-app 是镜像的名称，. 表示 Dockerfile 所在的当前目录。

3. 运行 Docker 容器

使用 docker run 命令运行构建的镜像。

docker run -p 3000:3000 my-node-app

-p 标志用于端口映射，将主机的 3000 端口映射到容器的 3000 端口。

**Dockerfile 的更多指令**

除了上述常用指令，Dockerfile 还支持其他多种指令，例如：

ENV: 设置环境变量。

示例：ENV NODE\_ENV production

ADD: 类似于 COPY，但可以自动解压 tar 文件，并支持从 URL 下载文件。

示例：ADD https://example.com/file.tar.gz /app/

ENTRYPOINT: 配置一个容器启动时执行的命令，可以与 CMD 配合使用。

示例：ENTRYPOINT ["npm", "start"]

VOLUME: 创建一个挂载点，用于挂载主机目录或其他容器的数据卷。

示例：VOLUME /app/data

USER: 指定运行后续指令的用户。

示例：USER node

通过 Dockerfile，你可以定义并自动化创建 Docker 镜像的过程，使得应用程序可以在任何支持 Docker 的环境中一致地构建和运行。

使用 rz 命令上传一个 jar 包(我们的软件)

图45

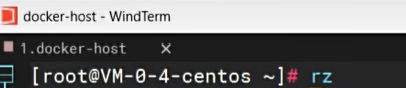
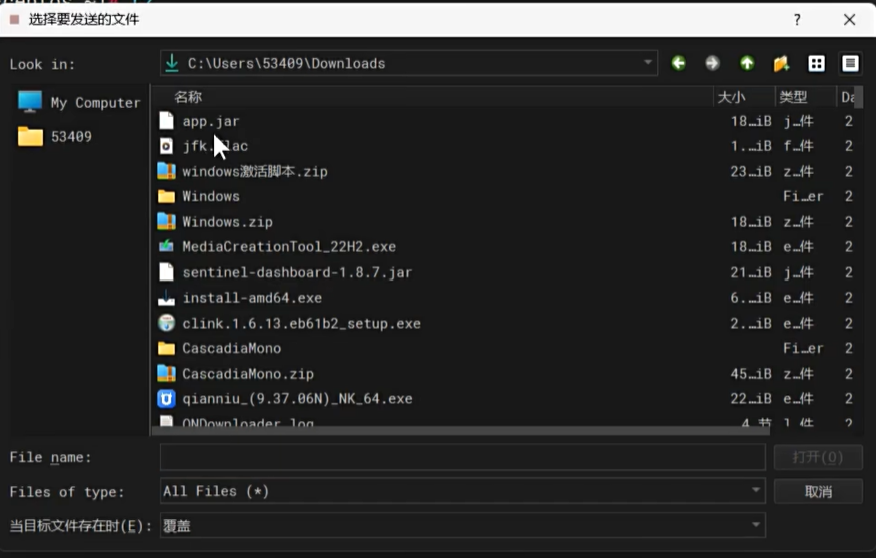


图46

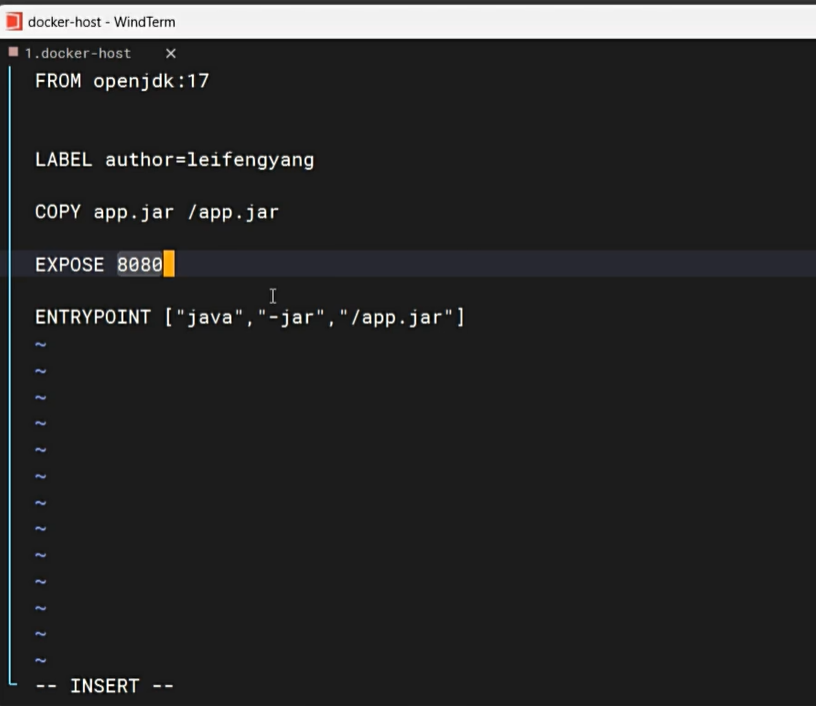


创建dockerfile，输入i进入插入模式

图47

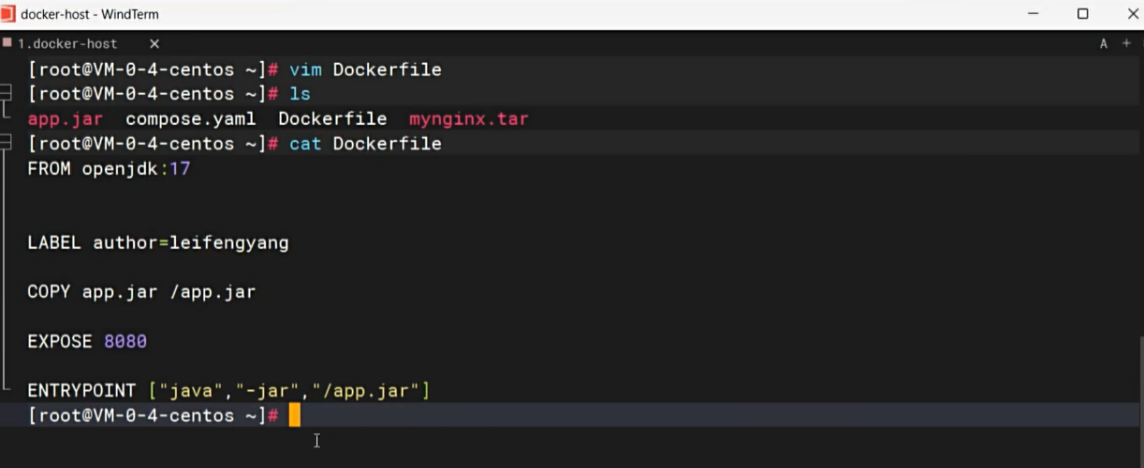


插入模式下编写dockerfile，完成后再INSERT位置输入:wq 完成保存

图48  


检查dockerfile

图49



构建镜像，复制 jar 包到容器内部

图50



图51

