

Rockchip Linux Docker 部署指南

文件标识：RK-KF-YF-924

发布版本：V1.0.4

日期：2023-09-20

文件密级：绝密 秘密 内部资料 公开

免责声明

本文档按“现状”提供，瑞芯微电子股份有限公司（“本公司”，下同）不对本文档的任何陈述、信息和内容的准确性、可靠性、完整性、适销性、特定目的性和非侵权性提供任何明示或暗示的声明或保证。本文档仅作为使用指导的参考。

由于产品版本升级或其他原因，本文档将可能在未经任何通知的情况下，不定期进行更新或修改。

商标声明

“Rockchip”、“瑞芯微”、“瑞芯”均为本公司的注册商标，归本公司所有。

本文档可能提及的其他所有注册商标或商标，由其各自拥有者所有。

版权所有 © 2023 瑞芯微电子股份有限公司

超越合理使用范畴，非经本公司书面许可，任何单位和个人不得擅自摘抄、复制本文档内容的部分或全部，并不得以任何形式传播。

瑞芯微电子股份有限公司

Rockchip Electronics Co., Ltd.

地址：福建省福州市铜盘路软件园A区18号

网址：www.rock-chips.com

客户服务电话：+86-4007-700-590

客户服务传真：+86-591-83951833

客户服务邮箱：fae@rock-chips.com

前言

概述

本文主要介绍了 Docker 的基本使用方式，同时提供编译 SDK 的 Docker 镜像环境的构建方式，并针对使用过程中的常见问题进行总结，提供解决方法参考。

产品版本

芯片名称	内核版本
ALL	ALL

读者对象

本文档（本指南）主要适用于以下工程师：

技术支持工程师
软件开发工程师

修订记录

版本号	作者	修改日期	修改说明
V1.0.0	WJL	2022-04-12	初始版本
V1.0.1	WJL	2023-02-20	更新内容以及布局排版
V1.0.2	WJL	2023-03-29	更新 Dockerfile 和 FAQ
V1.0.3	WJL	2023-05-15	更新 Dockerfile
V1.0.4	WJL	2023-09-20	更新 Dockerfile

目录

Rockchip Linux Docker 部署指南

1. 介绍
2. 安装
 - 2.1 Debian 系发行版，如 Debian、Ubuntu
 - 2.2 Relhat 系发行版，如 Redhat、fedora、centos
3. 常用命令说明
 - 3.1 通过 Dockerfile 创建镜像
 - 3.2 删除镜像或者容器
 - 3.3 重命名镜像或者容器
 - 3.4 查镜像或者容器
 - 3.5 运行 Docker 环境
 - 3.6 镜像管理
4. 如何使用 Docker 编译 SDK
 - 4.1 构建 Docker 镜像
 - 4.1.1 使用 Dockerfile 构建
 - 4.1.2 使用提供的镜像
 - 4.2 使用 Docker 编译
 - 4.3 更新 Docker 镜像
5. 如何在 Rockchip 平台系统上运行 Docker
 - 5.1 Kernel 配置
 - 5.2 Buildroot 配置
 - 5.3 Debian 配置
6. Dockerfile 参考
7. FAQ
 - 7.1 获取镜像失败，error pulling image configuration: Get https:.....read: connection reset by peer
 - 7.2 代码所有权转移问题，error.GitError: rev-parse: fatal: detected dubious ownership in repository at

1. 介绍

Docker 是一个开源的应用容器引擎，开发者可以打包应用和依赖包到一个轻量级、可移植的容器中，然后发布到任何流行的 Linux 机器上，能够更高效的利用系统资源、保证一致的运行环境，实现持续交付和部署，以及后期更轻松的迁移、维护、扩展。

已验证的系统如下：

发行版本	Docker 版本	镜像加载	固件编译
ubuntu 22.04	24.0.5	pass	pass
ubuntu 21.10	20.10.12	pass	pass
ubuntu 21.04	20.10.7	pass	pass
ubuntu 18.04	20.10.7	pass	pass
fedora35	20.10.12	pass	NR (not run)

2. 安装

2.1 Debian 系发行版，如 Debian、Ubuntu

```
sudo apt-get install docker.io
```

2.2 Relhat 系发行版，如 Redhat、fedora、centos

```
sudo yum install docker  
sudo dnf install docker
```

3. 常用命令说明

3.1 通过 Dockerfile 创建镜像

```
# $name:$tag 镜像名称：标签名称  
# Dockerfile 基于上下文构建，构建目录必须包含Dockerfile  
# $dockerfile 指定 Dockerfile 名称，默认是 PATH/Dockerfile  
# $dockerfile_dir Dockerfile 所在路径是 PATH  
sudo docker build -f $dockerfile -t $name:$tag $dockerfile_dir
```

3.2 删除镜像或者容器

```
# $imageID 镜像ID  
sudo docker rmi $imageID  
  
# $containerID 容器ID  
sudo docker rm $containerID
```

3.3 重命名镜像或者容器

```
# $imageID 镜像ID  
# $name:$tag 镜像名称:标签名称  
sudo docker tag $imageID $name:$tag  
  
# $containerID 容器ID  
# $name 容器名称  
sudo docker rename $containerID $name
```

3.4 查镜像或者容器

```
sudo docker image ls  
  
sudo docker container ls
```

3.5 运行 Docker 环境

```
# 运行 Docker 环境  
# --privileged 特权模式  
# -it 表示开启交互模式, /bin/bash 表示交互方式  
# -v $host_dir:$docker_dir 将主机目录映射到 Docker 内  
# -p $host_port:$docker_port 将主机端口映射到 Docker 内  
# -u $docker_user 指定使用 Docker 内用户登陆  
# -w $cwd_dir 切换到容器内的路径  
# -d --detach 设置后台运行模式  
  
# 运行指定的镜像  
sudo docker run --privileged -it -u $docker_user -v $host_dir:$docker_dir  
$imageID /bin/bash  
  
# 运行指定的容器  
sudo docker exec -it -w $cwd_dir $containerID /bin/bash
```

3.6 镜像管理

```
# 登录 dockerhub 账号
sudo docker login -u $username -p $password

# 拉取 dockerhub 镜像
sudo docker image pull $imageID

# 推送镜像到 dockerhub
sudo docker image push $username/$imagename

# 导出本地镜像 (tar archive file)
sudo docker image save $name:$tag -o ${dockerimage.tar}

# 导入本地镜像 (tar archive file)
sudo docker image load -i ${dockerimage.tar}

# 本地镜像提交修改
# -m 提交的描述信息
# -a 提交的作者
# $containerID 发生修改的容器ID
# $new_name:$new_tag 提交后的镜像名称标签
sudo docker commit -m $commit_message -a $author $containerID
$new_name:$new_tag
```

4. 如何使用 Docker 编译 SDK

4.1 构建 Docker 镜像

4.1.1 使用 Dockerfile 构建

```
# 参考该文档提供的 Dockerfile
# 假设 Dockerfile 位于 /home/docker/Dockerfile
cd /home/docker
sudo docker build -t docker_rk:latest .
```

4.1.2 使用提供的镜像

Docker 镜像可从网址 [docker](#) 获取。

4.2 使用 Docker 编译

```
# 假设 SDK 位于 /home/user/sdk
# 将 SDK 映射到 Docker 镜像内，并进入镜像内
sudo docker run --privileged -it -u rk -v /home/user/sdk:/home/rk/sdk
docker_rk:latest /bin/bash

# 切换到 Docker 内的路径，编译方法可以通过 build.sh -h 查看
cd /home/rk/sdk
./build.sh -h
```

4.3 更新 Docker 镜像

```
# 退出 Docker 镜像后，除映射目录外的所有修改都不会保留，要保留相应修改需要更新 Docker 镜像
# 假定本次镜像实例化的容器为 rk@ecbbcdc7e5ca:/$
# 按照以下命令，更新 Docker 镜像
sudo docker commit -m "update" ecbbcdc7e5ca docker_rk:latest
```

5. 如何在 Rockchip 平台系统上运行 Docker

5.1 Kernel 配置

Docker 运行需要 kernel 开启 cgroups、namespace、netfilter、overlayfs 等功能的支持，请确保你所使用的配置已经满足 docker 运行的要求。宿主机上可以通过脚本 `/usr/share/docker.io/contrib/check-config.sh` 进行检查，如果系统上没有该脚本，可以通过[check-config.sh](#)获取。

同时，我们提供了通用的 docker 配置，可以通过以下命令进行配置：

```
make ARCH=arm64 rockchip_linux_defconfig rockchip_linux_docker.config
```

5.2 Buildroot 配置

Buildroot 默认不开启 docker 相关配置，如需要 docker 相关功能，可以开启以下配置：

```
BR2_PACKAGE_CGROUPFS_MOUNT=y
BR2_PACKAGE_DOCKER_ENGINE=y
BR2_PACKAGE_DOCKER_ENGINE_EXPERIMENTAL=y
BR2_PACKAGE_DOCKER_ENGINE_STATIC_CLIENT=y
BR2_PACKAGE_DOCKER_ENGINE_DRIVER_BTRFS=y
BR2_PACKAGE_DOCKER_ENGINE_DRIVER_DEVICEMAPPER=y
BR2_PACKAGE_DOCKER_ENGINE_DRIVER_VFS=y
```

5.3 Debian 配置

Debian 上直接安装 docker 即可，需要注意 Debian 默认使用 iptables-nft，而 docker 默认使用 iptables-legacy，故需要配置 iptables 使用 legacy 版本，可以通过以下命令进行切换：

```
# 使用 iptables-legacy
update-alternatives --set iptables /usr/sbin/iptables-legacy
update-alternatives --set ip6tables /usr/sbin/ip6tables-legacy

# 使用 iptables-nft
update-alternatives --set iptables /usr/sbin/iptables-nft
update-alternatives --set ip6tables /usr/sbin/ip6tables-nft
```

6. Dockerfile 参考

```
FROM ubuntu:22.04
RUN \
# use mirror sources
echo "deb http://mirrors.ustc.edu.cn/ubuntu/ jammy main restricted universe
multiverse\n\
deb http://mirrors.ustc.edu.cn/ubuntu/ jammy-security main restricted universe
multiverse\n\
deb http://mirrors.ustc.edu.cn/ubuntu/ jammy-updates main restricted universe
multiverse\n\
deb http://mirrors.ustc.edu.cn/ubuntu/ jammy-backports main restricted universe
multiverse" \
> /etc/apt/sources.list \
# install packages
&& DEBIAN_FRONTEND=noninteractive apt-get update -y \
&& DEBIAN_FRONTEND=noninteractive apt-get upgrade -y \
&& DEBIAN_FRONTEND=noninteractive apt-get install -y \
bc binfmt-support bison bsdmainutils build-essential chrpath cmake cpio curl
device-tree-compiler diffstat \
expat expect fdisk file flex gawk git iputils-ping libegl1-mesa libelf-dev
libgmp-dev libgucharmap-2-90-dev \
liblz4-tool libmpc-dev libncurses-dev libssl1.2-dev libssl-dev locales live-
build mesa-common-dev net-tools \
patchelf python2 python2-dev python3 python-is-python3 python3-git python3-
jinja2 python3-pexpect python3-pip \
python3-pyelftools python3-subunit qemu-user-static repo rsync ssh strace swig
vim socat sudo texinfo time tree \
unzip wget \
# add user in docker
&& useradd -c 'rk user' -m -d /home/rk -s /bin/bash rk && usermod -a -G sudo rk \
&& sed -i -e '/%sudo tALL=(ALL:ALL) ALL/ c %sudo ALL=(ALL) NOPASSWD: ALL'
/etc/sudoers \
&& echo "docker image build complete"
# delete useless package and cache if need
#&& apt-get autoclean && apt-get autoremove && rm -rf /var/lib/apt/lists/*
```

7. FAQ

7.1 获取镜像失败，error pulling image configuration: Get https:.....read: connection reset by peer

当前网络无法访问 Docker 官网镜像仓库，可以通过配置 Docker 国内镜像仓库，解决无法访问的问题，可参考以下修改：

```
# 在 /etc/docker/daemon.json 添加以下内容
{
  "registry-mirrors": ["https://docker.mirrors.ustc.edu.cn/"]
}
```

7.2 代码所有权转移问题，error.GitError: rev-parse: fatal: detected dubious ownership in repository at

通常代码的所有权属于当前下载代码的 Host 主机用户，在 Docker 中编译时，因为 Docker 环境中的用户没有代码的所有权，所以会触发编译报错，可以通过以下命令修改所有仓库的转移权。当 Docker 内用户获得代码所有权时，相对的，Host 主机用户也会因为权限转移而丢失代码的权限。

```
git config --global --add safe.directory "*"
```