Rockchip Linux Docker Deploy Developer Guide

ID: RK-KF-YF-924

Release Version: V1.0.2

Release Date: 2023-03-29

Security Level: ☐Top-Secret ☐Secret ☐Internal ■Public

Rockchip Electronics Co., Ltd.

No.18 Building, A District, No.89, software Boulevard Fuzhou, Fujian,PRC

Website: www.rock-chips.com

Customer service Tel: +86-4007-700-590

Customer service Fax: +86-591-83951833

Customer service e-Mail: fae@rock-chips.com

Preface

Overview

This document is going to introduce the basic usage of Docker, and provides the way to build the image environment for building the docker of the SDK, and summarizes some frequently ask questions during usage process and provides solutions for reference.

Product Version

Chipset	Kernel Version
ALL	ALL

Intended Audience

This document (this guide) is mainly intended for:

Technical support engineers

Software development engineers

Revision History

Version	Author	Date	Change Description
V1.0.0	WJL	2022-04-12	Initial version
V1.0.1	WJL	2023-02-20	Update content and layout
V1.0.2	WJL	2023-03-29	Update Dockerfile and FAQ

Contents

1. Overview

Docker is an open source application container engine. Developers can package applications and dependencies into a lightweight and portable container, and then publish them to any popular Linux device, which can use system resources more efficiently and ensure a consistent operating environment, to realize continuous release and deployment, and facilitate porting, maintenance, and expansion in the future.

Validated systems are as follows:

Release Version	Docker Version	Image Load	Firmware Build
ubuntu 22.04	20.10.12	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. Installation

2.1 Debian-based Distributions, Such As Debian, Ubuntu

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

2.2 Relhat-based Distributions, Such As Redhat, fedora, centos

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

3. Usage

3.1 Create an Image with a Dockerfile

```
# $name:$tag Image name: tag name
# Dockfile is built based on the context, the built directory must contain
Dockerfile
# $dockerfile specifies the Dockerfile name, it is PATH/Dockerfile by default
# $dockerfile_dir Dockerfile is located at PATH
sudo docker build -f $dockerfile -t $name:$tag $dockerfile_dir
```

3.2 Delete an Image or Container

```
# $imageID
sudo docker rmi $imageID

# $containerID
sudo docker rm $containerID
```

3.3 Rename Image or Container

```
# $imageID: Image ID
# $name:$tag: image name:tag name
sudo docker tag $imageID $name:$tag

# $containerID container ID
# $name container name
sudo docker rename $containerID $name
```

3.4 To Check the Image or Container

```
sudo docker image ls

sudo docker container ls
```

3.5 Run Docker Environment

```
# Run Docker environment
# --privileged: privileged mode
# -it: means enabling interactive mode, /bin/bash: means interactive mode
# -v $host_dir:$docker_dir: maps the host directory into Docker
# -p $host_port:$docker_port: maps the host port into Docker
# -u $docker_user: specifies to use the user in Docker to log in
# -w $cwd_dir: switches to the path inside the container
# -d --detach: set background running mode

# Run the specified image
sudo docker run --privileged -it -u $docker_user -v $host_dir:$docker_dir
$imageID /bin/bash
```

```
# Run the specified container
sudo docker exec -it -w $cwd_dir $containerID /bin/bash
```

3.6 Image Management

```
# Login to dockerhub account
sudo docker login -u $username -p $password

# Pull the dockerhub image
sudo docker image pull $imageID

# Push image to dockerhub
sudo docker image push $username/$imagename

# Export local image (tar archive file)
sudo docker image save $name:$tag -o ${dockerimage.tar}

# Import local image (tar archive file)
sudo docker image load -i ${dockerimage.tar}

# Change the local immage and commit
# -m commit information
# -a the author of the commit
# $containerID: the container ID which is changed
# $new_name:$new_tag: commited image name tag
sudo docker commit -m $commit_message -a $author $containerID $new_name:$new_tag
```

4. How to build SDK with Docker

4.1 Build Image by Using Dockerfile

```
# Refer to the Dockerfile provided by this document
# Suppose Dockerfile is in /home/docker/Dockerfile
cd /home/docker
sudo docker build -t docker_rk:latest .
```

4.2 Build SDK by using Docker Image

```
# Suppose the SDK is located at /home/user/SDK
# Maps the SDK to the Docker image and enter the image
sudo docker run --privileged -it -u rk -v /home/user/SDK:/home/rk
docker_rk:latest /bin/bash

# Switch to the path in Docker, the compilation method can be check through
build.sh -h
cd /home/rk
./build.sh -h
```

4.3 Update Local Docker Image

```
# After exiting the Docker image, all modifications except the mapping directory
will not be saved. If you want to save the corresponding modifications, the
Docker image needs to be updated
# Supports the container instantiated by this image is rk@ecbbcdc7e5ca:/$
# Follow the command below to update the Docker image
sudo docker commit -m "update" ecbbcdc7e5ca docker_rk:latest
```

5. How to run Docker in Rockchip platform

5.1 Kernel config

To run Docker, you need the kernel to enable the support of cgroups, namespace, netfilter, overlayfs and other functions, please make sure that the configuration you use has met the requirements of docker running. The host can be checked by the script `/usr/share/docker.io/contrib/check-config.sh`, or obtained by check-config.sh if it is not already available on the system.

At the same time, we provide a general docker configuration, which can be configured with the following command:

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

5.2 Buildroot config

Buildroot does not enable docker-related configurations by default, if you need docker-related functions, you can enable the following configurations:

```
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 config

Docker can be installed directly on Debian. It should be noted that Debian uses iptables-nft by default, and docker uses iptables-legacy by default, so you need to configure iptables to use the legacy version, which can be switched by the following command:

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

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

6. Dockerfile Reference

```
FROM ubuntu:20.04
ENV DEBIAN_FRONTEND=noninteractive
RUN \
# use mirror sources
echo "deb http://mirrors.ustc.edu.cn/ubuntu/ focal main restricted universe
multiverse\n\
deb http://mirrors.ustc.edu.cn/ubuntu/ focal-security main restricted universe
multiverse\n\
deb http://mirrors.ustc.edu.cn/ubuntu/ focal-updates main restricted universe
multiverse\n\
deb http://mirrors.ustc.edu.cn/ubuntu/ focal-backports main restricted universe
multiverse" \
> /etc/apt/sources.list \
# install packages
&& apt-get update -y && apt-get upgrade -y \
&& apt-get install curl -y \
&& curl https://mirrors.tuna.tsinghua.edu.cn/git/git-repo > /usr/bin/repo &&
chmod 755 /usr/bin/repo \
&& apt-get install -y bc binfmt-support bison bsdmainutils build-essential bzip2
chrpath cmake cpio cpp-aarch64-linux-gnu \
debianutils device-tree-compiler diffstat expat expect fakeroot fdisk flex g++
gawk gcc gcc-multilib git git-core \
g++-multilib gpgv2 iputils-ping libegl1-mesa libelf-dev libgmp-dev libgucharmap-
2-90-dev liblz4-tool libmpc-dev \
libsdl1.2-dev libssl-dev live-build make ncurses-dev net-tools patchelf python
python3 python3-crypto python3-git \
python3-jinja2 python3-pexpect python3-pip qemu-user-static rsync socat ssh
strace sudo texinfo time tree unzip vim \
wget xterm xz-utils zstd \
&& pip3 install pyelftools -i http://pypi.mirrors.ustc.edu.cn/simple/ --trusted-
host pypi.mirrors.ustc.edu.cn \
# add user in docker
&& useradd -c 'rk user' -m -d /home/rk -s /bin/bash rk && sed -i -e '/\%sudo/ c
\%sudo ALL=(ALL) NOPASSWD: ALL' /etc/sudoers && usermod -a -G sudo rk \
&& 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 Error Pulling Image Configuration: Get Https:......Read: Connection Reset by Peer

The current network cannot access the Docker official website image repository. You can solve this problem by configuring the Docker domestic image repository. Please refer to the following modifications:

```
# Add the following content to /etc/docker/daemon.json
{
   "registry-mirrors": ["https://docker.mirrors.ustc.edu.cn/"]
}
```

7.2 Error.GitError: ...... rev-parse: fatal: detected dubious ownership in repository at

Usually the ownership of the repository belongs to the host user who currently downloads the repository, when compiling in Docker, because the user in the Docker environment does not have the ownership of the repository, it will trigger a compilation error, you can modify the transfer right of all repositories through the following command. When a user in Docker takes ownership of the repository, the Host host user also loses the permission of the repository due to permission transfer.

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