Docker installation on Centos:

#!/bin/sh

set -e

# Docker Engine for Linux installation script.

#

# This script is intended as a convenient way to configure docker's package

# repositories and to install Docker Engine, This script is not recommended

# for production environments. Before running this script, make yourself familiar

# with potential risks and limitations, and refer to the installation manual

# at https://docs.docker.com/engine/install/ for alternative installation methods.

#

# The script:

#

# - Requires `root` or `sudo` privileges to run.

# - Attempts to detect your Linux distribution and version and configure your

# package management system for you.

# - Doesn't allow you to customize most installation parameters.

# - Installs dependencies and recommendations without asking for confirmation.

# - Installs the latest stable release (by default) of Docker CLI, Docker Engine,

# Docker Buildx, Docker Compose, containerd, and runc. When using this script

# to provision a machine, this may result in unexpected major version upgrades

# of these packages. Always test upgrades in a test environment before

# deploying to your production systems.

# - Isn't designed to upgrade an existing Docker installation. When using the

# script to update an existing installation, dependencies may not be updated

# to the expected version, resulting in outdated versions.

#

# Source code is available at https://github.com/docker/docker-install/

#

# Usage

# ==============================================================================

#

# To install the latest stable versions of Docker CLI, Docker Engine, and their

# dependencies:

#

# 1. download the script

#

# $ curl -fsSL https://get.docker.com -o install-docker.sh

#

# 2. verify the script's content

#

# $ cat install-docker.sh

#

# 3. run the script with --dry-run to verify the steps it executes

#

# $ sh install-docker.sh --dry-run

#

# 4. run the script either as root, or using sudo to perform the installation.

#

# $ sudo sh install-docker.sh

#

# Command-line options

# ==============================================================================

#

# --version <VERSION>

# Use the --version option to install a specific version, for example:

#

# $ sudo sh install-docker.sh --version 23.0

#

# --channel <stable|test>

#

# Use the --channel option to install from an alternative installation channel.

# The following example installs the latest versions from the "test" channel,

# which includes pre-releases (alpha, beta, rc):

#

# $ sudo sh install-docker.sh --channel test

#

# Alternatively, use the script at https://test.docker.com, which uses the test

# channel as default.

#

# --mirror <Aliyun|AzureChinaCloud>

#

# Use the --mirror option to install from a mirror supported by this script.

# Available mirrors are "Aliyun" (https://mirrors.aliyun.com/docker-ce), and

# "AzureChinaCloud" (https://mirror.azure.cn/docker-ce), for example:

#

# $ sudo sh install-docker.sh --mirror AzureChinaCloud

#

# ==============================================================================

# Git commit from https://github.com/docker/docker-install when

# the script was uploaded (Should only be modified by upload job):

SCRIPT\_COMMIT\_SHA="e5543d473431b782227f8908005543bb4389b8de"

# strip "v" prefix if present

VERSION="${VERSION#v}"

# The channel to install from:

# \* stable

# \* test

# \* edge (deprecated)

# \* nightly (deprecated)

DEFAULT\_CHANNEL\_VALUE="stable"

if [ -z "$CHANNEL" ]; then

CHANNEL=$DEFAULT\_CHANNEL\_VALUE

fi

DEFAULT\_DOWNLOAD\_URL="https://download.docker.com"

if [ -z "$DOWNLOAD\_URL" ]; then

DOWNLOAD\_URL=$DEFAULT\_DOWNLOAD\_URL

fi

DEFAULT\_REPO\_FILE="docker-ce.repo"

if [ -z "$REPO\_FILE" ]; then

REPO\_FILE="$DEFAULT\_REPO\_FILE"

fi

mirror=''

DRY\_RUN=${DRY\_RUN:-}

while [ $# -gt 0 ]; do

case "$1" in

--channel)

CHANNEL="$2"

shift

;;

--dry-run)

DRY\_RUN=1

;;

--mirror)

mirror="$2"

shift

;;

--version)

VERSION="${2#v}"

shift

;;

--\*)

echo "Illegal option $1"

;;

esac

shift $(( $# > 0 ? 1 : 0 ))

done

case "$mirror" in

Aliyun)

DOWNLOAD\_URL="https://mirrors.aliyun.com/docker-ce"

;;

AzureChinaCloud)

DOWNLOAD\_URL="https://mirror.azure.cn/docker-ce"

;;

"")

;;

\*)

>&2 echo "unknown mirror '$mirror': use either 'Aliyun', or 'AzureChinaCloud'."

exit 1

;;

esac

case "$CHANNEL" in

stable|test)

;;

edge|nightly)

>&2 echo "DEPRECATED: the $CHANNEL channel has been deprecated and is no longer supported by this script."

exit 1

;;

\*)

>&2 echo "unknown CHANNEL '$CHANNEL': use either stable or test."

exit 1

;;

esac

command\_exists() {

command -v "$@" > /dev/null 2>&1

}

# version\_gte checks if the version specified in $VERSION is at least the given

# SemVer (Maj.Minor[.Patch]), or CalVer (YY.MM) version.It returns 0 (success)

# if $VERSION is either unset (=latest) or newer or equal than the specified

# version, or returns 1 (fail) otherwise.

#

# examples:

#

# VERSION=23.0

# version\_gte 23.0 // 0 (success)

# version\_gte 20.10 // 0 (success)

# version\_gte 19.03 // 0 (success)

# version\_gte 21.10 // 1 (fail)

version\_gte() {

if [ -z "$VERSION" ]; then

return 0

fi

eval version\_compare "$VERSION" "$1"

}

# version\_compare compares two version strings (either SemVer (Major.Minor.Path),

# or CalVer (YY.MM) version strings. It returns 0 (success) if version A is newer

# or equal than version B, or 1 (fail) otherwise. Patch releases and pre-release

# (-alpha/-beta) are not taken into account

#

# examples:

#

# version\_compare 23.0.0 20.10 // 0 (success)

# version\_compare 23.0 20.10 // 0 (success)

# version\_compare 20.10 19.03 // 0 (success)

# version\_compare 20.10 20.10 // 0 (success)

# version\_compare 19.03 20.10 // 1 (fail)

version\_compare() (

set +x

yy\_a="$(echo "$1" | cut -d'.' -f1)"

yy\_b="$(echo "$2" | cut -d'.' -f1)"

if [ "$yy\_a" -lt "$yy\_b" ]; then

return 1

fi

if [ "$yy\_a" -gt "$yy\_b" ]; then

return 0

fi

mm\_a="$(echo "$1" | cut -d'.' -f2)"

mm\_b="$(echo "$2" | cut -d'.' -f2)"

# trim leading zeros to accommodate CalVer

mm\_a="${mm\_a#0}"

mm\_b="${mm\_b#0}"

if [ "${mm\_a:-0}" -lt "${mm\_b:-0}" ]; then

return 1

fi

return 0

)

is\_dry\_run() {

if [ -z "$DRY\_RUN" ]; then

return 1

else

return 0

fi

}

is\_wsl() {

case "$(uname -r)" in

\*microsoft\* ) true ;; # WSL 2

\*Microsoft\* ) true ;; # WSL 1

\* ) false;;

esac

}

is\_darwin() {

case "$(uname -s)" in

\*darwin\* ) true ;;

\*Darwin\* ) true ;;

\* ) false;;

esac

}

deprecation\_notice() {

distro=$1

distro\_version=$2

echo

printf "\033[91;1mDEPRECATION WARNING\033[0m\n"

printf " This Linux distribution (\033[1m%s %s\033[0m) reached end-of-life and is no longer supported by this script.\n" "$distro" "$distro\_version"

echo " No updates or security fixes will be released for this distribution, and users are recommended"

echo " to upgrade to a currently maintained version of $distro."

echo

printf "Press \033[1mCtrl+C\033[0m now to abort this script, or wait for the installation to continue."

echo

sleep 10

}

get\_distribution() {

lsb\_dist=""

# Every system that we officially support has /etc/os-release

if [ -r /etc/os-release ]; then

lsb\_dist="$(. /etc/os-release && echo "$ID")"

fi

# Returning an empty string here should be alright since the

# case statements don't act unless you provide an actual value

echo "$lsb\_dist"

}

echo\_docker\_as\_nonroot() {

if is\_dry\_run; then

return

fi

if command\_exists docker && [ -e /var/run/docker.sock ]; then

(

set -x

$sh\_c 'docker version'

) || true

fi

# intentionally mixed spaces and tabs here -- tabs are stripped by "<<-EOF", spaces are kept in the output

echo

echo "================================================================================"

echo

if version\_gte "20.10"; then

echo "To run Docker as a non-privileged user, consider setting up the"

echo "Docker daemon in rootless mode for your user:"

echo

echo " dockerd-rootless-setuptool.sh install"

echo

echo "Visit https://docs.docker.com/go/rootless/ to learn about rootless mode."

echo

fi

echo

echo "To run the Docker daemon as a fully privileged service, but granting non-root"

echo "users access, refer to https://docs.docker.com/go/daemon-access/"

echo

echo "WARNING: Access to the remote API on a privileged Docker daemon is equivalent"

echo " to root access on the host. Refer to the 'Docker daemon attack surface'"

echo " documentation for details: https://docs.docker.com/go/attack-surface/"

echo

echo "================================================================================"

echo

}

# Check if this is a forked Linux distro

check\_forked() {

# Check for lsb\_release command existence, it usually exists in forked distros

if command\_exists lsb\_release; then

# Check if the `-u` option is supported

set +e

lsb\_release -a -u > /dev/null 2>&1

lsb\_release\_exit\_code=$?

set -e

# Check if the command has exited successfully, it means we're in a forked distro

if [ "$lsb\_release\_exit\_code" = "0" ]; then

# Print info about current distro

cat <<-EOF

You're using '$lsb\_dist' version '$dist\_version'.

EOF

# Get the upstream release info

lsb\_dist=$(lsb\_release -a -u 2>&1 | tr '[:upper:]' '[:lower:]' | grep -E 'id' | cut -d ':' -f 2 | tr -d '[:space:]')

dist\_version=$(lsb\_release -a -u 2>&1 | tr '[:upper:]' '[:lower:]' | grep -E 'codename' | cut -d ':' -f 2 | tr -d '[:space:]')

# Print info about upstream distro

cat <<-EOF

Upstream release is '$lsb\_dist' version '$dist\_version'.

EOF

else

if [ -r /etc/debian\_version ] && [ "$lsb\_dist" != "ubuntu" ] && [ "$lsb\_dist" != "raspbian" ]; then

if [ "$lsb\_dist" = "osmc" ]; then

# OSMC runs Raspbian

lsb\_dist=raspbian

else

# We're Debian and don't even know it!

lsb\_dist=debian

fi

dist\_version="$(sed 's/\/.\*//' /etc/debian\_version | sed 's/\..\*//')"

case "$dist\_version" in

12)

dist\_version="bookworm"

;;

11)

dist\_version="bullseye"

;;

10)

dist\_version="buster"

;;

9)

dist\_version="stretch"

;;

8)

dist\_version="jessie"

;;

esac

fi

fi

fi

}

do\_install() {

echo "# Executing docker install script, commit: $SCRIPT\_COMMIT\_SHA"

if command\_exists docker; then

cat >&2 <<-'EOF'

Warning: the "docker" command appears to already exist on this system.

If you already have Docker installed, this script can cause trouble, which is

why we're displaying this warning and provide the opportunity to cancel the

installation.

If you installed the current Docker package using this script and are using it

again to update Docker, you can safely ignore this message.

You may press Ctrl+C now to abort this script.

EOF

( set -x; sleep 20 )

fi

user="$(id -un 2>/dev/null || true)"

sh\_c='sh -c'

if [ "$user" != 'root' ]; then

if command\_exists sudo; then

sh\_c='sudo -E sh -c'

elif command\_exists su; then

sh\_c='su -c'

else

cat >&2 <<-'EOF'

Error: this installer needs the ability to run commands as root.

We are unable to find either "sudo" or "su" available to make this happen.

EOF

exit 1

fi

fi

if is\_dry\_run; then

sh\_c="echo"

fi

# perform some very rudimentary platform detection

lsb\_dist=$( get\_distribution )

lsb\_dist="$(echo "$lsb\_dist" | tr '[:upper:]' '[:lower:]')"

if is\_wsl; then

echo

echo "WSL DETECTED: We recommend using Docker Desktop for Windows."

echo "Please get Docker Desktop from https://www.docker.com/products/docker-desktop/"

echo

cat >&2 <<-'EOF'

You may press Ctrl+C now to abort this script.

EOF

( set -x; sleep 20 )

fi

case "$lsb\_dist" in

ubuntu)

if command\_exists lsb\_release; then

dist\_version="$(lsb\_release --codename | cut -f2)"

fi

if [ -z "$dist\_version" ] && [ -r /etc/lsb-release ]; then

dist\_version="$(. /etc/lsb-release && echo "$DISTRIB\_CODENAME")"

fi

;;

debian|raspbian)

dist\_version="$(sed 's/\/.\*//' /etc/debian\_version | sed 's/\..\*//')"

case "$dist\_version" in

12)

dist\_version="bookworm"

;;

11)

dist\_version="bullseye"

;;

10)

dist\_version="buster"

;;

9)

dist\_version="stretch"

;;

8)

dist\_version="jessie"

;;

esac

;;

centos|rhel|sles)

if [ -z "$dist\_version" ] && [ -r /etc/os-release ]; then

dist\_version="$(. /etc/os-release && echo "$VERSION\_ID")"

fi

;;

\*)

if command\_exists lsb\_release; then

dist\_version="$(lsb\_release --release | cut -f2)"

fi

if [ -z "$dist\_version" ] && [ -r /etc/os-release ]; then

dist\_version="$(. /etc/os-release && echo "$VERSION\_ID")"

fi

;;

esac

# Check if this is a forked Linux distro

check\_forked

# Print deprecation warnings for distro versions that recently reached EOL,

# but may still be commonly used (especially LTS versions).

case "$lsb\_dist.$dist\_version" in

debian.stretch|debian.jessie)

deprecation\_notice "$lsb\_dist" "$dist\_version"

;;

raspbian.stretch|raspbian.jessie)

deprecation\_notice "$lsb\_dist" "$dist\_version"

;;

ubuntu.xenial|ubuntu.trusty)

deprecation\_notice "$lsb\_dist" "$dist\_version"

;;

ubuntu.impish|ubuntu.hirsute|ubuntu.groovy|ubuntu.eoan|ubuntu.disco|ubuntu.cosmic)

deprecation\_notice "$lsb\_dist" "$dist\_version"

;;

fedora.\*)

if [ "$dist\_version" -lt 36 ]; then

deprecation\_notice "$lsb\_dist" "$dist\_version"

fi

;;

esac

# Run setup for each distro accordingly

case "$lsb\_dist" in

ubuntu|debian|raspbian)

pre\_reqs="apt-transport-https ca-certificates curl"

if ! command -v gpg > /dev/null; then

pre\_reqs="$pre\_reqs gnupg"

fi

apt\_repo="deb [arch=$(dpkg --print-architecture) signed-by=/etc/apt/keyrings/docker.gpg] $DOWNLOAD\_URL/linux/$lsb\_dist $dist\_version $CHANNEL"

(

if ! is\_dry\_run; then

set -x

fi

$sh\_c 'apt-get update -qq >/dev/null'

$sh\_c "DEBIAN\_FRONTEND=noninteractive apt-get install -y -qq $pre\_reqs >/dev/null"

$sh\_c 'install -m 0755 -d /etc/apt/keyrings'

$sh\_c "curl -fsSL \"$DOWNLOAD\_URL/linux/$lsb\_dist/gpg\" | gpg --dearmor --yes -o /etc/apt/keyrings/docker.gpg"

$sh\_c "chmod a+r /etc/apt/keyrings/docker.gpg"

$sh\_c "echo \"$apt\_repo\" > /etc/apt/sources.list.d/docker.list"

$sh\_c 'apt-get update -qq >/dev/null'

)

pkg\_version=""

if [ -n "$VERSION" ]; then

if is\_dry\_run; then

echo "# WARNING: VERSION pinning is not supported in DRY\_RUN"

else

# Will work for incomplete versions IE (17.12), but may not actually grab the "latest" if in the test channel

pkg\_pattern="$(echo "$VERSION" | sed 's/-ce-/~ce~.\*/g' | sed 's/-/.\*/g')"

search\_command="apt-cache madison docker-ce | grep '$pkg\_pattern' | head -1 | awk '{\$1=\$1};1' | cut -d' ' -f 3"

pkg\_version="$($sh\_c "$search\_command")"

echo "INFO: Searching repository for VERSION '$VERSION'"

echo "INFO: $search\_command"

if [ -z "$pkg\_version" ]; then

echo

echo "ERROR: '$VERSION' not found amongst apt-cache madison results"

echo

exit 1

fi

if version\_gte "18.09"; then

search\_command="apt-cache madison docker-ce-cli | grep '$pkg\_pattern' | head -1 | awk '{\$1=\$1};1' | cut -d' ' -f 3"

echo "INFO: $search\_command"

cli\_pkg\_version="=$($sh\_c "$search\_command")"

fi

pkg\_version="=$pkg\_version"

fi

fi

(

pkgs="docker-ce${pkg\_version%=}"

if version\_gte "18.09"; then

# older versions didn't ship the cli and containerd as separate packages

pkgs="$pkgs docker-ce-cli${cli\_pkg\_version%=} containerd.io"

fi

if version\_gte "20.10"; then

pkgs="$pkgs docker-compose-plugin docker-ce-rootless-extras$pkg\_version"

fi

if version\_gte "23.0"; then

pkgs="$pkgs docker-buildx-plugin"

fi

if ! is\_dry\_run; then

set -x

fi

$sh\_c "DEBIAN\_FRONTEND=noninteractive apt-get install -y -qq $pkgs >/dev/null"

)

echo\_docker\_as\_nonroot

exit 0

;;

centos|fedora|rhel)

if [ "$(uname -m)" != "s390x" ] && [ "$lsb\_dist" = "rhel" ]; then

echo "Packages for RHEL are currently only available for s390x."

exit 1

fi

if [ "$lsb\_dist" = "fedora" ]; then

pkg\_manager="dnf"

config\_manager="dnf config-manager"

enable\_channel\_flag="--set-enabled"

disable\_channel\_flag="--set-disabled"

pre\_reqs="dnf-plugins-core"

pkg\_suffix="fc$dist\_version"

else

pkg\_manager="yum"

config\_manager="yum-config-manager"

enable\_channel\_flag="--enable"

disable\_channel\_flag="--disable"

pre\_reqs="yum-utils"

pkg\_suffix="el"

fi

repo\_file\_url="$DOWNLOAD\_URL/linux/$lsb\_dist/$REPO\_FILE"

(

if ! is\_dry\_run; then

set -x

fi

$sh\_c "$pkg\_manager install -y -q $pre\_reqs"

$sh\_c "$config\_manager --add-repo $repo\_file\_url"

if [ "$CHANNEL" != "stable" ]; then

$sh\_c "$config\_manager $disable\_channel\_flag 'docker-ce-\*'"

$sh\_c "$config\_manager $enable\_channel\_flag 'docker-ce-$CHANNEL'"

fi

$sh\_c "$pkg\_manager makecache"

)

pkg\_version=""

if [ -n "$VERSION" ]; then

if is\_dry\_run; then

echo "# WARNING: VERSION pinning is not supported in DRY\_RUN"

else

pkg\_pattern="$(echo "$VERSION" | sed 's/-ce-/\\\\.ce.\*/g' | sed 's/-/.\*/g').\*$pkg\_suffix"

search\_command="$pkg\_manager list --showduplicates docker-ce | grep '$pkg\_pattern' | tail -1 | awk '{print \$2}'"

pkg\_version="$($sh\_c "$search\_command")"

echo "INFO: Searching repository for VERSION '$VERSION'"

echo "INFO: $search\_command"

if [ -z "$pkg\_version" ]; then

echo

echo "ERROR: '$VERSION' not found amongst $pkg\_manager list results"

echo

exit 1

fi

if version\_gte "18.09"; then

# older versions don't support a cli package

search\_command="$pkg\_manager list --showduplicates docker-ce-cli | grep '$pkg\_pattern' | tail -1 | awk '{print \$2}'"

cli\_pkg\_version="$($sh\_c "$search\_command" | cut -d':' -f 2)"

fi

# Cut out the epoch and prefix with a '-'

pkg\_version="-$(echo "$pkg\_version" | cut -d':' -f 2)"

fi

fi

(

pkgs="docker-ce$pkg\_version"

if version\_gte "18.09"; then

# older versions didn't ship the cli and containerd as separate packages

if [ -n "$cli\_pkg\_version" ]; then

pkgs="$pkgs docker-ce-cli-$cli\_pkg\_version containerd.io"

else

pkgs="$pkgs docker-ce-cli containerd.io"

fi

fi

if version\_gte "20.10"; then

pkgs="$pkgs docker-compose-plugin docker-ce-rootless-extras$pkg\_version"

fi

if version\_gte "23.0"; then

pkgs="$pkgs docker-buildx-plugin"

fi

if ! is\_dry\_run; then

set -x

fi

$sh\_c "$pkg\_manager install -y -q $pkgs"

)

echo\_docker\_as\_nonroot

exit 0

;;

sles)

if [ "$(uname -m)" != "s390x" ]; then

echo "Packages for SLES are currently only available for s390x"

exit 1

fi

if [ "$dist\_version" = "15.3" ]; then

sles\_version="SLE\_15\_SP3"

else

sles\_minor\_version="${dist\_version##\*.}"

sles\_version="15.$sles\_minor\_version"

fi

repo\_file\_url="$DOWNLOAD\_URL/linux/$lsb\_dist/$REPO\_FILE"

pre\_reqs="ca-certificates curl libseccomp2 awk"

(

if ! is\_dry\_run; then

set -x

fi

$sh\_c "zypper install -y $pre\_reqs"

$sh\_c "zypper addrepo $repo\_file\_url"

if ! is\_dry\_run; then

cat >&2 <<-'EOF'

WARNING!!

openSUSE repository (https://download.opensuse.org/repositories/security:SELinux) will be enabled now.

Do you wish to continue?

You may press Ctrl+C now to abort this script.

EOF

( set -x; sleep 30 )

fi

opensuse\_repo="https://download.opensuse.org/repositories/security:SELinux/$sles\_version/security:SELinux.repo"

$sh\_c "zypper addrepo $opensuse\_repo"

$sh\_c "zypper --gpg-auto-import-keys refresh"

$sh\_c "zypper lr -d"

)

pkg\_version=""

if [ -n "$VERSION" ]; then

if is\_dry\_run; then

echo "# WARNING: VERSION pinning is not supported in DRY\_RUN"

else

pkg\_pattern="$(echo "$VERSION" | sed 's/-ce-/\\\\.ce.\*/g' | sed 's/-/.\*/g')"

search\_command="zypper search -s --match-exact 'docker-ce' | grep '$pkg\_pattern' | tail -1 | awk '{print \$6}'"

pkg\_version="$($sh\_c "$search\_command")"

echo "INFO: Searching repository for VERSION '$VERSION'"

echo "INFO: $search\_command"

if [ -z "$pkg\_version" ]; then

echo

echo "ERROR: '$VERSION' not found amongst zypper list results"

echo

exit 1

fi

search\_command="zypper search -s --match-exact 'docker-ce-cli' | grep '$pkg\_pattern' | tail -1 | awk '{print \$6}'"

# It's okay for cli\_pkg\_version to be blank, since older versions don't support a cli package

cli\_pkg\_version="$($sh\_c "$search\_command")"

pkg\_version="-$pkg\_version"

fi

fi

(

pkgs="docker-ce$pkg\_version"

if version\_gte "18.09"; then

if [ -n "$cli\_pkg\_version" ]; then

# older versions didn't ship the cli and containerd as separate packages

pkgs="$pkgs docker-ce-cli-$cli\_pkg\_version containerd.io"

else

pkgs="$pkgs docker-ce-cli containerd.io"

fi

fi

if version\_gte "20.10"; then

pkgs="$pkgs docker-compose-plugin docker-ce-rootless-extras$pkg\_version"

fi

if version\_gte "23.0"; then

pkgs="$pkgs docker-buildx-plugin"

fi

if ! is\_dry\_run; then

set -x

fi

$sh\_c "zypper -q install -y $pkgs"

)

echo\_docker\_as\_nonroot

exit 0

;;

\*)

if [ -z "$lsb\_dist" ]; then

if is\_darwin; then

echo

echo "ERROR: Unsupported operating system 'macOS'"

echo "Please get Docker Desktop from https://www.docker.com/products/docker-desktop"

echo

exit 1

fi

fi

echo

echo "ERROR: Unsupported distribution '$lsb\_dist'"

echo

exit 1

;;

esac

exit 1

}

# wrapped up in a function so that we have some protection against only getting

# half the file during "curl | sh"

do\_install