

Docker安裝

Centos

[Docker Engine on CentOS](#) | [Docker Documentation](#)

```
sudo yum remove docker \
    docker-client \
    docker-client-latest \
    docker-common \
    docker-latest \
    docker-latest-logrotate \
    docker-logrotate \
    docker-engine
```

```
sudo yum install -y yum-utils
sudo yum-config-manager \
    --add-repo \
    https://download.docker.com/linux/centos/docker-ce.repo
```

```
sudo yum-config-manager --enable docker-ce-nightly

sudo yum install docker-ce docker-ce-cli containerd.io

sudo systemctl start docker
sudo systemctl enable docker
```

~/docker/config.json

```
{
  "proxies":
  {
    "default":
    {
      "httpProxy": "http://192.168.1.12:3128",
      "httpsProxy": "http://192.168.1.12:3128",
      "noProxy": "*test.example.com,example2.com,127.0.0.0/8"
    }
  }
}
```

/etc/docker/daemon.json

```
{
  "registry-mirrors":["http://hub-mirror.c.163.com"],
}
```

```
registry-1.docker.io. 54 IN A 34.197.211.151
registry-1.docker.io. 54 IN A 34.192.145.113
registry-1.docker.io. 54 IN A 18.214.230.110
registry-1.docker.io. 54 IN A 52.55.168.20
registry-1.docker.io. 54 IN A 3.229.227.53
registry-1.docker.io. 54 IN A 3.209.182.229
registry-1.docker.io. 54 IN A 3.213.204.48
registry-1.docker.io. 54 IN A 54.161.109.204
```

```
mkdir /etc/systemd/system/docker.service.d
vim /etc/systemd/system/docker.service.d/http-proxy.conf
#####
[Service]
Environment="HTTP_PROXY=http://proxy.example.com:80/"
Environment="NO_PROXY=localhost,127.0.0.0/8,docker-registry.somecorporation.com"
```

```
#####  
sudo systemctl daemon-reload  
sudo systemctl show --property Environment docker  
sudo systemctl restart docker
```

使用get.docker.com 腳本

<https://get.docker.com>

```
curl -L get.docker.com -o get-docker.sh  
sh get-docker.sh
```

```
#!/bin/sh  
set -e  
# Docker CE for Linux installation script  
#  
# See https://docs.docker.com/engine/install/ for the installation steps.  
#  
# This script is meant for quick & easy install via:  
# $ curl -fsSL https://get.docker.com -o get-docker.sh  
# $ sh get-docker.sh  
#  
# For test builds (ie. release candidates):  
# $ curl -fsSL https://test.docker.com -o test-docker.sh  
# $ sh test-docker.sh  
#  
# NOTE: Make sure to verify the contents of the script  
#       you downloaded matches the contents of install.sh  
#       located at https://github.com/docker/docker-install  
#       before executing.  
#  
# Git commit from https://github.com/docker/docker-install when  
# the script was uploaded (Should only be modified by upload job):  
SCRIPT_COMMIT_SHA="93d2499759296ac1f9c510605fef85052a2c32be"  
  
# strip "v" prefix if present  
VERSION="${VERSION#v}"  
  
# The channel to install from:  
# * nightly  
# * test  
# * stable  
# * edge (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  
        --mirror)  
            mirror="$2"  
            shift  
            ;;
```

```

--dry-run)
  DRY_RUN=1
  ;;
--*)
  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"
    ;;
esac

command_exists() {
  command -v "$@" > /dev/null 2>&1
}

# version_gte checks if the version specified in $VERSION is at least
# the given CalVer (YY.MM) version. returns 0 (success) if $VERSION is either
# unset (=latest) or newer or equal than the specified version. Returns 1 (fail)
# otherwise.
#
# examples:
#
# VERSION=20.10
# version_gte 20.10 // 0 (success)
# version_gte 19.03 // 0 (success)
# version_gte 21.10 // 1 (fail)
version_gte() {
  if [ -z "$VERSION" ]; then
    return 0
  fi
  eval calver_compare "$VERSION" "$1"
}

# calver_compare compares two CalVer (YY.MM) version strings. 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:
#
# calver_compare 20.10 19.03 // 0 (success)
# calver_compare 20.10 20.10 // 0 (success)
# calver_compare 19.03 20.10 // 1 (fail)
calver_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 "$2" | cut -d'.' -f2)"
  if [ "${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
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 "$dist_version" ] && [ -r /etc/lsb-release ]; then
    dist_version="$(cat /etc/lsb-release && echo "$DISTRIB_CODENAME")"
fi
;;

debian|raspbian)
    dist_version="$(sed 's/\/.*//' /etc/debian_version | sed 's/\..*//')"
    case "$dist_version" in
        11)
            dist_version="bullseye"
        ;;
        10)
            dist_version="buster"
        ;;
        9)
            dist_version="stretch"
        ;;
        8)
            dist_version="jessie"
        ;;
    esac
;;

centos|rhel|sles)
    if [ -z "$dist_version" ] && [ -r /etc/os-release ]; then
        dist_version="$(cat /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="$(cat /etc/os-release && echo "$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"
    ;;
    fedora.*)
        if [ "$dist_version" -lt 33 ]; then
            deprecation_notice "$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=/usr/share/keyrings/docker-archive-keyring.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"
    $sh_c "curl -fsSL \"$DOWNLOAD_URL/linux/$lsb_dist/gpg\" | gpg --dearmor --yes -o /usr/share/keyrings/docker-archive-keyring.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").*-0~$lsb_dist"
        search_command="apt-cache madison 'docker-ce' | grep '$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="$pkg_version"
    fi
fi
(
    pkgs=""
    if version_gte "18.09"; then
        # older versions don't support a cli package
        pkgs="$pkgs docker-ce-cli${cli_pkg_version%}"
    fi
    if version_gte "20.10" && [ "$(uname -m)" = "x86_64" ]; then
        # also install the latest version of the "docker scan" cli-plugin (only supported on x86 currently)
        pkgs="$pkgs docker-scan-plugin"
    fi
    pkgs="$pkgs docker-ce${pkg_version%}"
    if ! is_dry_run; then
        set -x
    fi
    $sh_c "DEBIAN_FRONTEND=noninteractive apt-get install -y -qq --no-install-recommends $pkgs >/dev/null"
    if version_gte "20.10"; then
        # Install docker-ce-rootless-extras without "--no-install-recommends", so as to install slirp4netns when available
        $sh_c "DEBIAN_FRONTEND=noninteractive apt-get install -y -qq docker-ce-rootless-extras${pkg_version%} >/dev/null"
    fi
)
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
yum_repo="$DOWNLOAD_URL/linux/$lsb_dist/$REPO_FILE"
if ! curl -Ifs "$yum_repo" > /dev/null; then

```



```

echo "Error: Unable to curl repository file $yum_repo, is it valid?"
exit 1
fi
if [ "$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
(
if ! is_dry_run; then
set -x
fi
$sh_c "$pkg_manager install -y -q $pre_reqs"
$sh_c "$config_manager --add-repo $yum_repo"

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
(
if ! is_dry_run; then
set -x
fi
# install the correct cli version first
if [ -n "$cli_pkg_version" ]; then
$sh_c "$pkg_manager install -y -q docker-ce-cli-$cli_pkg_version"
fi
$sh_c "$pkg_manager install -y -q docker-ce-$pkg_version"
if version_gte "20.10"; then
$sh_c "$pkg_manager install -y -q docker-ce-rootless-extras$pkg_version"
fi

```

```

)
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

sles_version="${dist_version##*}"
sles_repo="$DOWNLOAD_URL/linux/$lsb_dist/$REPO_FILE"
opensuse_repo="https://download.opensuse.org/repositories/security:SELinux/SLE_15_SP$sles_version/security:SELinux.repo"
if ! curl -Ifs "$sles_repo" > /dev/null; then
    echo "Error: Unable to curl repository file $sles_repo, is it valid?"
    exit 1
fi
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 $sles_repo"
    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
    $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"

        search_command="zypper search -s --match-exact 'docker-ce-rootless-extras' | grep '$pkg_pattern' | tail -1 | awk '{print \$6}'"
        rootless_pkg_version="$($sh_c "$search_command")"
        rootless_pkg_version="-$rootless_pkg_version"
    fi
fi
(
    if ! is_dry_run; then
        set -x
    fi
    # install the correct cli version first
    if [ -n "$cli_pkg_version" ]; then

```

```

    $sh_c "zypper install -y docker-ce-cli-$cli_pkg_version"
  fi
  $sh_c "zypper install -y docker-ce-$pkg_version"
  if version_gte "20.10"; then
    $sh_c "zypper install -y docker-ce-rootless-extras-$rootless_pkg_version"
  fi
)
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

```

🕒 修訂版本 #10

★ 由 treeman 建立於 12 🕒 @🕒🕒🕒 2022 03:06:06

🔧 由 treeman 更新於 5 🕒🕒🕒🕒 2023 10:14:58