CentOS

https://docs.docker.com/engine/installation/linux/centos/

Docker runs on CentOS 7.X. An installation on other binary compatible EL7 distributions such as Scientific Linux might succeed, but Docker does not test or support Docker on these distributions.

This page instructs you to install using Docker-managed release packages and installation mechanisms. Using these packages ensures you get the latest release of Docker. If you wish to install using CentOS-managed packages, consult your CentOS documentation.

Prerequisites

Docker requires a 64-bit installation regardless of your CentOS version. Also, your kernel must be 3.10 at minimum, which CentOS 7 runs.

To check your current kernel version, open a terminal and use `uname -r` to display your kernel version:

```
$ uname -r
3.10.0-229.el7.x86_64
```

Finally, it is recommended that you fully update your system. Please keep in mind that your system should be fully patched to fix any potential kernel bugs. Any reported kernel bugs may have already been fixed on the latest kernel packages.

Install

There are two ways to install Docker Engine. You can install using the `yum` package manager. Or you can use `curl` with the `get.docker.com` site. This second method runs an installation script which also installs via the `yum` package manager.

Install with yum

1. Log into your machine as a user with `sudo` or `root` privileges.

2. Make sure your existing yum packages are up-to-date.

   ```
   $ sudo yum update
   ```

3. Add the yum repo.

   ```
   $ sudo tee /etc/yum.repos.d/docker.repo <<-'EOF'
   [dockerrepo]
   name=Docker Repository
   baseurl=https://yum.dockerproject.org/repo/main/centos/$releasever/
   ```

```
enabled=1
gpgcheck=1
gpgkey=https://yum.dockerproject.org/gpg
EOF
```

4. Install the Docker package.

   ```
   $ sudo yum install docker-engine
   ```

5. Start the Docker daemon.

   ```
   $ sudo service docker start
   ```

6. Verify `docker` is installed correctly by running a test image in a container.

   ```
   $ sudo docker run hello-world
   Unable to find image 'hello-world:latest' locally
       latest: Pulling from hello-world
       a8219747be10: Pull complete
       91c95931e552: Already exists
       hello-world:latest: The image you are pulling has been verified.
   Important: image verification is a tech preview feature and should not be
   relied on to provide security.
       Digest:
   sha256:aa03e5d0d5553b4c3473e89c8619cf79df368babd1.7.1cf5daeb82aab55838d
       Status: Downloaded newer image for hello-world:latest
       Hello from Docker.
       This message shows that your installation appears to be working
   correctly.

       To generate this message, Docker took the following steps:
        1. The Docker client contacted the Docker daemon.
        2. The Docker daemon pulled the "hello-world" image from the Docker Hub.
               (Assuming it was not already locally available.)
        3. The Docker daemon created a new container from that image which runs
   the
               executable that produces the output you are currently reading.
        4. The Docker daemon streamed that output to the Docker client, which
   sent it
               to your terminal.

       To try something more ambitious, you can run an Ubuntu container with:
        $ docker run -it ubuntu bash

       For more examples and ideas, visit:
        http://docs.docker.com/userguide/
   ```

Install with the script

1. Log into your machine as a user with `sudo` or `root` privileges.

2. Make sure your existing yum packages are up-to-date.

   ```
   $ sudo yum update
   ```

3. Run the Docker installation script.

   ```
   $ curl -fsSL https://get.docker.com/ | sh
   ```

   This script adds the `docker.repo` repository and installs Docker.

4. Start the Docker daemon.

   ```
   $ sudo service docker start
   ```

5. Verify `docker` is installed correctly by running a test image in a container.

   ```
   $ sudo docker run hello-world
   ```

Create a docker group

The `docker` daemon binds to a Unix socket instead of a TCP port. By default that Unix socket is owned by the user `root` and other users can access it with `sudo`. For this reason, `docker` daemon always runs as the `root` user.

To avoid having to use `sudo` when you use the `docker` command, create a Unix group called `docker` and add users to it. When the `docker` daemon starts, it makes the ownership of the Unix socket read/writable by the `docker` group.

> **Warning**: The `docker` group is equivalent to the `root` user; For details on how this impacts security in your system, see *Docker Daemon Attack Surface* for details.

To create the `docker` group and add your user:

1. Log into Centos as a user with `sudo` privileges.

2. Create the `docker` group.

   ```
   sudo groupadd docker
   ```

3. Add your user to `docker` group.

   ```
   sudo usermod -aG docker your_username
   ```

4. Log out and log back in.

   This ensures your user is running with the correct permissions.

5. Verify your work by running `docker` without `sudo`.

   ```
   $ docker run hello-world
   ```

Start the docker daemon at boot

To ensure Docker starts when you boot your system, do the following:

```
$ sudo chkconfig docker on
```

If you need to add an HTTP Proxy, set a different directory or partition for the Docker runtime files, or make other customizations, read our Systemd article to learn how to customize your Systemd Docker daemon options.

Uninstall

You can uninstall the Docker software with `yum`.

1. List the package you have installed.

   ```
   $ yum list installed | grep docker
   yum list installed | grep docker
   docker-engine.x86_64   1.7.1-1.el7 @/docker-engine-1.7.1-1.el7.x86_64.rpm
   ```

2. Remove the package.

   ```
   $ sudo yum -y remove docker-engine.x86_64
   ```

   This command does not remove images, containers, volumes, or user-created configuration files on your host.

3. To delete all images, containers, and volumes, run the following command:

   ```
   $ rm -rf /var/lib/docker
   ```

4. Locate and delete any user-created configuration files.