Install Docker Engine on Ubuntu

Prerequisites

OS requirements

To install Docker Engine, you need the 64-bit version of one of these Ubuntu versions:

- Ubuntu Hirsute 21.04
- Ubuntu Groovy 20.10
- Ubuntu Focal 20.04 (LTS)
- Ubuntu Bionic 18.04 (LTS)
- Ubuntu Xenial 16.04 (LTS)

Docker Engine is supported on `x86_64` (or `amd64`), `armhf`, and `arm64` architectures.

Install using the repository

Before you install Docker Engine for the first time on a new host machine, you need to set up the Docker repository. Afterward, you can install and update Docker from the repository.

Set up the repository

1. Update the `apt` package index and install packages to allow `apt` to use a repository over HTTPS:

```
sudo apt update && sudo apt upgrade
```

```
sudo apt install apt-transport-https ca-certificates curl gnupg lsb-release
```

2. Add Docker's official GPG key:

```
curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo gpg --dearmor -o /usr/share/keyrings/docker-archive-keyring.gpg
```

3. Use the following command to set up the **stable** repository. To add the **nightly** or **test** repository, add the word `nightly` or `test` (or both) after the word `stable` in the commands below.

```
echo "deb [arch=amd64 signed-by=/usr/share/keyrings/docker-archive-keyring.gpg] https://download.docker.com/linux/ubuntu $(lsb_release -cs) stable" | sudo tee /etc/apt/sources.list.d/docker.list > /dev/null
```

Install Docker Engine

1. Update the `apt` package index, and install the *latest version* of Docker Engine and containerd, or go to the next step to install a specific version:

```
sudo apt update
```

```
sudo apt install docker-ce docker-ce-cli containerd.io
```

2. Verify that Docker Engine is installed correctly by running the `hello-world` image.

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

This command downloads a test image and runs it in a container. When the container runs, it prints an informational message and exits.

Docker Engine is installed and running. The `docker` group is created but no users are added to it. You need to use `sudo` to run Docker commands.

Manage Docker as a non-root user

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 only access it using `sudo`. The Docker daemon always runs as the `root` user.

If you don't want to preface the `docker` command with `sudo`, create a Unix group called `docker` and add users to it. When the Docker daemon starts, it creates a Unix socket accessible by members of the `docker` group.

> Warning
>
> The `docker` group grants privileges equivalent to the `root` user.

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

1. Create the `docker` group.

```
sudo groupadd docker
```

2. Add your user to the `docker` group.

```
sudo usermod -aG docker $USER
```

3. Log out and log back in so that your group membership is re-evaluated.

   On Linux, you can also run the following command to activate the changes to groups:

   ```
   newgrp docker
   ```

4. Verify that you can run `docker` commands without `sudo`.

   ```
   docker run hello-world
   ```

Install cvmfs

Getting the software

To add the CVMFS repository and install CVMFS run :

```
wget https://ecsft.cern.ch/dist/cvmfs/cvmfs-release/cvmfs-release-
latest_all.deb
```

```
sudo dpkg -i cvmfs-release-latest_all.deb
```

```
rm -f cvmfs-release-latest_all.deb
```

```
sudo apt update
```

```
sudo apt install cvmfs
```

Setting up the Software

1. Create default.local

   Create `/etc/cvmfs/default.local` with `sudo nano /etc/cvmfs/default.local` and write in :

   ```

   ```

```
CVMFS_QUOTA=10000
CVMFS_REPOSITORIES=oasis.opensciencegrid.org
CVMFS_HTTP_PROXY=DIRECT
```

2. Configure AutoFS

```
sudo cvmfs_config setup
```

3. Verify the file system Check if CernVM-FS mounts the specified repositories by :

```
cvmfs_config probe
```

If the probe fails, try to restart autofs with `sudo systemctl restart autofs`

Download `clas12software` docker

Create folder : `mkdir ~/mywork` and `cd ~/mywork`

```
sudo docker run -it --rm -v /cvmfs:/cvmfs:shared -v
~/mywork:/jlab/work/mywork jeffersonlab/clas12software:production bash
```

For having an interactive windows :

```
sudo docker run -it --rm -p 6080:6080 -v /cvmfs:/cvmfs:shared -v
~/mywork:/jlab/work/mywork jeffersonlab/clas12software:production bash
```

For quit interactive docker : `crtl p + crtl q`

Generate ALERT geometry

Create `script_install.sh` in `mywork` with inside :

```
echo "remove java-1.8.0"
dnf remove java-1.8.0-openjdk-headless.x86_64 -y

echo "install java-11"
dnf install java-11-openjdk-devel -y

echo "install maven"
```

```
wget https://www-us.apache.org/dist/maven/maven-3/3.6.3/binaries/apache-
maven-3.6.3-bin.tar.gz -P /tmp
tar xf /tmp/apache-maven-3.6.3-bin.tar.gz -C /opt
ln -s /opt/apache-maven-3.6.3 /opt/maven

export JAVA_HOME=/usr/lib/jvm/jre-openjdk
export M2_HOME=/opt/maven
export MAVEN_HOME=/opt/maven
export PATH=${M2_HOME}/bin:${PATH}

echo "Set python as alternative for python3"
alternatives --set python /usr/bin/python3

echo "groovy install"
curl -s get.sdkman.io | bash
source "$HOME/.sdkman/bin/sdkman-init.sh"
sdk install groovy
```

Run it :

```
. script_install.sh
```

Clone the `clas12-offline-software` repository :

```
git clone https://github.com/JeffersonLab/clas12-offline-software
```

Change to the Alert branch :

```
cd clas12-offline-software
```

```
git checkout Alert
```

And build it :

```
./build-coatjava.sh
```

Go to `mywork` folder :

```
cd /jlab/work/mywork
```

Clone the detectors repository :

```
git clone https://github.com/gemc/detectors
```

```
cd detectors/clas12
```

```
./../../clas12-offline-software/coatjava/bin/run-groovy
alert/AHDC_geom/factory_ahdc.groovy --variation rga_fall2018 --runnumber 11
```

```
cp ahdc__* alert/AHDC_geom/
```

```
./../../clas12-offline-software/coatjava/bin/run-groovy
alert/ATOF_geom/factory_atof.groovy --variation rga_fall2018 --runnumber 11
```

```
cp atof__* alert/ATOF_geom/
```

build the detectors :

```
cd alert/AHDC_geom
```

```
./ahdc.pl config.dat
```

```
cd ../ATOF_geom
```

Change line `detector_name: myatof` to `detector_name: atof` in `config.dat`.

```
./atof.pl config.dat
```

Go to `mywork` folder :

```
cd /jlab/work/mywork
```

Clone `clas12Tags` repository :

```
git clone https://github.com/gemc/clas12Tags
```

```
cd clas12Tags/4.4.0/source
```

```
scons -j4 OPT=1
```

Create a `alert.gcard` :

```
<gcard>

    <!-- Implementation ahdc, example -->
    <detector
name="/jlab/work/mywork/detectors/clas12/alert/AHDC_geom/ahdc"
factory="TEXT" variation="default"/>
    <detector
name="/jlab/work/mywork/detectors/clas12/alert/ATOF_geom/myatof"
factory="TEXT" variation="default"/>

    <option name="BEAM_P" value="e-, 10.0*GeV, 20*deg, 20*deg"/>
    <option name="SPREAD_P" value="2.0*GeV, 20*deg, 180*deg, flat"/>

    <option name="BEAM_V" value="(0, 0, -2.0)cm"/>
    <option name="SPREAD_V" value="(0.0, 0.0)cm"/>

    <option name="OUTPUT" value="txt, out.txt"/>
    <option name="N" value="1"/>


</gcard>
```

Run gemc :

```
./gemc -USE_GUI=0 alert.gcard
```