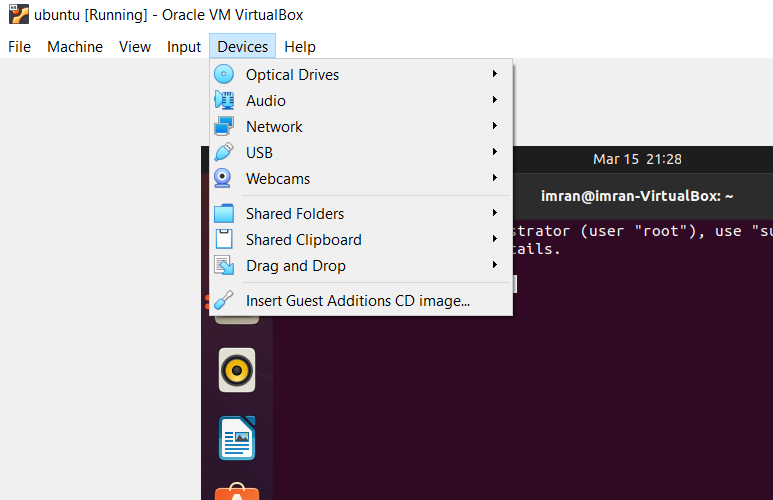
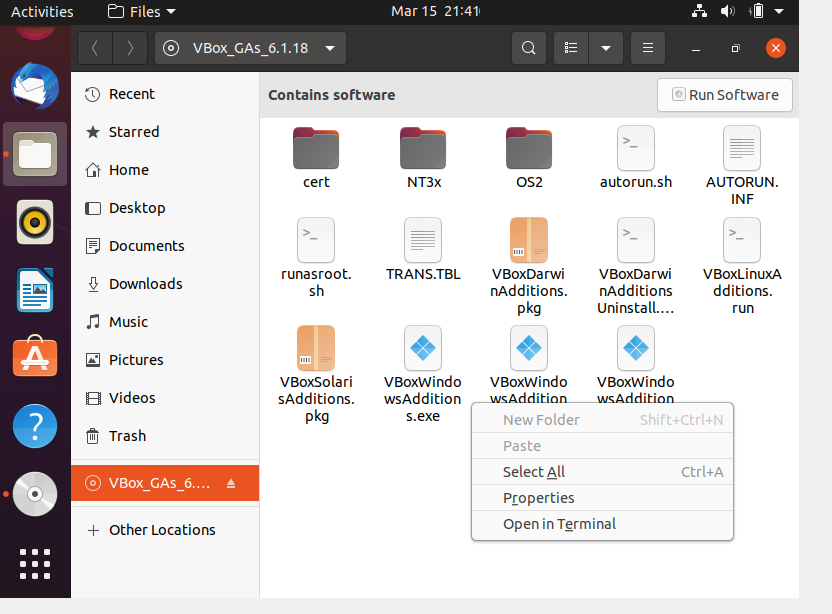
**LINUX COURSE**

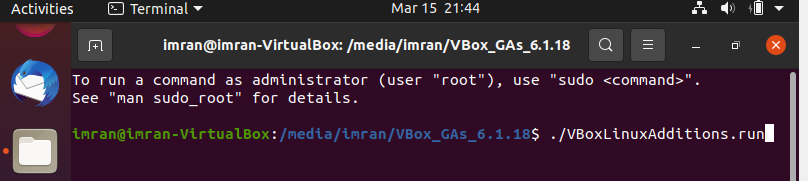
1. After installation of UBUNTU in virtual box dis select/remove the iso file from storage in settings.
2. After this there will be complete setup.
3. Then go to devices insert guest Additions CD image.



1. Now click on VBOX\_GAS or VBOX\_ADDITION which is disc shaped ( also can be found in home directory)’
2. In vbox\_GAS click on VBoxLinuxAdministrator and check permissions in properties if it is executable.
3. Now open terminal in vbox folder to select that location.



1. Now in terminal type command.

****

1. If it asks for administrator privilages add sudo in beginning.
2. Now restart after successful installation

**Customisation**

1. **To change themes we need to download packages from internet.**
2. **Go to browser type GNOME-LOOK.ORG.**
3. **Use gtk 3.x**

**COMMAND PROMPT**

1. **To check which directory u r in “pwd” command is used.**
2. **cd = change directory.**
3. **/ = (absolute path) used to navigate from home folder.**
4. **In my home directory there is a directory called “usr” I need to get “share” directory from “usr”**

**cd /usr/share**

1. **Now if I want to get back to home directory I will use**

**$ cd /home**

1. **./ = current directory. If I am in usr directory I can access its inside directories.**

**cd /usr**

**cd ./**

**cd ./lib**

**(just like /usr/lib but from relative path)**

1. **cd ~ = tilda (used to go to home directory)**
2. **I can simultaneously go to home directory and choose another directory using.**

**Ex: cd ~/Documents**

1. **cd ../ is used to get back to parent directory.**

**$ cd ../**

**$ cd ~ (to get back to home directory).**

**LS COMMAND**

1. **ls = command is used list directories within current directory.**
2. **ls -l = lists directories and shows permissions of the directory or file.**
3. **ls -r =lists directories in reverse order.**
4. **Ls -p = define file type (color of file)**
5. **ls -s = lists based on file size.**

**GIVING PERMISSIONS**

1. **IF I want to open file then**

**nano ./filename**

1. **To give writable permissions to file**

**sudo(super user)**

**sudo nano ./filename**

1. **“ sudo !! ” can also be used to open file in writable format if file is open.**
2. **We can give root permission to user by using “sudo su” it will make user as switch user and has all permissions of super user.**
3. **To take diable super user permission**

**Use “ su imran” (imran is su user).**

**To install anything**

* **We need super permission to do it**
* **get-apt istall is the command**
* **to make it intall use sudo before it**
* **or sudo !! after it.**
* **If u want to install something u have half knowledge which of its product to install use**

**“apt-cache search (name of package)**

* **To upgrade use**

**Sudo apt-get upgrade**

**Change ownership of a file**

* **sudo chown root:imran hello.txt (**here hello is a file located in imran directory ,we have given root permission to it)
* **sudo chown 646 hello.txt** ( 6= read-write to owner, 4 =read to group, 6= read-write to public)
* taking back root previlige from root

**sudo chown imran:imran hello.txt**

* to remove file : **rm killo.txt**

**CREATING DIRECTOTY**

* **mkdir** = command used to make new directory.
* To create a file inside a directory

**Sudo nano ./mydir/file.txt**

* **ls -l ./mydir** =command will list all the files and directories inside mydir and also their permissions.
* -R = recursive command (used when we want to change file permission inside a directory)

**“sudo chown -R imran:imran ./mydir**

**(**root:root is permission which only owner has, so we change to user imran:imran so file can be changed to executable).

(until we do imran:imran from root:root …. We cannot change its state to drwxr -)

* **touch** = command creat new files

**cd mydir**

**touch file1.txt file2.txt file3.cpp**

* **rm ./\*.cpp** = this will remove .cpp files from current directory.
* **rm mydir/\*** =is used to delete everything inside directory.
* **rm -rf mydir** = will remove whole directory.
* **cp -r ./dir\_1 ./dir\_2** = used to copy dir \_1 and all its component to dir\_2.
* **mv killo.txt mydir/killo.txt** = will move killo file into another directory.
* **mv mydir mydrr =** it renames mydir to mydrr.
* **cp mydrr/hi mydir/hii** = copies hii file from mydrr to mydir
* **cp ./mydir/hi ./hi** = used to copy hi file from mydir to home directory
* mkdir ./newdirectory = can create new directory within current direct one.
* mkdir /usr/bin/newdirectory or mkdir ../../newdirectory (two levels up)
* **find . -type f -name “\*.txt”** = will list all txt files.
* **find . -type f -iname “\*.txt”**  = will list even case sensitive files.
* **GREP COMMAND**= used to search inside files.
* if u have a php file named xyz and want to search for function word in xyz.

**grep “function” xyz.php**

* **grep -i “function” xyz.php (case unsensitive)**

or

**grep “function” ./\*** (which means it will not only search in xyz but also in other files of the current directory)

* **grep -n -i “function” ./\*** (-n used to show line number of the result).
* **find . -type f -iname “\*.php” -exec grep -n -i “function” {} +**

here it finds function irrespective of case and also execute the search to show which in which line specific code exist of search result.

* We can save the output of the command from previous into a file by = ls > outfile.txt

**find . -type f -iname “\*.php” -exec grep -n -i “function” {} + > f.txt**

( here in command itself we added output file to save in.)

ELASTIC SEARCH

What is Elasticsearch ?

ElasticSearch is an Open-**source** Enterprise REST based Real-time Search and Analytics Engine. It's core Search Functionality is built using **Apache** Lucene, but supports many other features.

How to install Elastic search?

* First check java is installed in system or not by typing java -version.
* If not installed linux will give hint how to install jre through commands.
* After installation of openjdk completes.
* Use this commands to install elastic search.
* wget -qO - https://artifacts.elastic.co/GPG-KEY-elasticsearch | sudo apt-key add –
* sudo apt-get install apt-transport-https
* echo "deb https://artifacts.elastic.co/packages/6.x/apt stable main" | sudo tee -a /etc/apt/sources.list.d/elastic-6.x.list

if not works use below commands

* sudo apt-get update && sudo apt-get install elasticsearch
* echo "deb https://artifacts.elastic.co/packages/oss-6.x/apt stable main" | sudo tee -a /etc/apt/sources.list.d/elastic-6.x.list

how to check whether elastic is installed or not?

apt-cache policy elasticsearch = this command will show you which version of eastic search is installed.

How to start elastic search ?

sudo services elasticsearch start = it will give result ok.

Go to browse and type localhost:9200 it will give a script which confirms the it started.

How to stop elasticsearch?

sudo services elasticsearch stop = it will stop service.

How to change localhost address of elastic search?

sudo nano /etc/elasticsearch/elasticsearch.yml

this file will open and go to the bottom where you will find #https:9200 change it to ex: https:1150 and remove # ..

now save and close the file.

Now type

sudo service elasticsearch restart it will restart the service to specified address.

Alternate and professional way to using

sudo service elasticsearch start and stop is

sudo systemctl start elasticsearch (ctl-control).

**GIT AND GITHUB**

**How to install git in linux?**

**sudo apt-get install git git-extras**

**git remote add github** [**https://github.com/narmiahsap/java.git**](https://github.com/narmiahsap/java.git)

**(here if errors shows the use origin in place of github .**

**This command is used to connect our github repository to our terminal)**

**git config –global user.name “narmiahsap”**

**git config –global user.email** [**pashaimran53@gmail.com**](mailto:pashaimran53@gmail.com)

**(this commands are used to connect our account)**

**GITHUB INITIALS SHURU**

Installing git

**sudo apt-git install git git-extras**

initiating git with user profile

**git config --global user.name**

**git config –global email** [**imran.pasha78911@gmail.com**](mailto:imran.pasha78911@gmail.com)

to initiate new repository

**git init**

to check what and all can we push to githb from the directory we use .

**ls lart**

now to add what we want to add from the list

**git add <file name>**

if want to select all then

**git add .**

**or git add -A**

To check status of github type

**git status**

after adding everything we need to , then we should commit it

**git commit -m “initial commit”**

**after everything we do we should type**

**git status**

**if a file get modified which is committed then we can recover it .**

**git checkout <filename>**

(it will recover your file how it was before modification)

Again likewise type

**git status**

if multiple files get modified by mistake then we have a command to recover it

**git checkout -f**

(it will match old commit and recover)

Git log is used to show when and what committed at what time

**git log**

if u want to filter commit log of last 5 then

**git log -p -5**

if you want to see in which line and where the modification took place in your code type

**git diff**

(if you added that modified filed by git add -A , then git diff will show nothing and in git status it will show modified.)

If u still want to get your file back from modified then

**git diff - - staged**

(it will show modified line with code)

**git checkout -f**

(it will restore old code to that of modified)

to remove anything which is selected but not delete from folder then type

**git rm --cached <file name>**

to remove file completely both from folder and staging we use

**git rm <filename>**

we should remove that file from commit if committed before removing.

**git commit -a -m “ comment”**

we can also check which file is modified using status command

**git status -s**

**CREATING BRANCH**

To create new branch in github alongside of main branch/ master branch

**git branch second**

if u wrote

**git branch**

it will show list of branches.

To switch from main branch to second one type

**git checkout second**

to confirm u r in second branch use

**git status**

to merge second into master

**git merge second (second is name given to new branch)**

To create new branch and directly switch to it use

**git checkout -b third**

**GITHUB PULL AND PUSH**

* First create a repository in github with a private (if u want private)
* Then establish connection from repository to terminal
* git remote add origin https:// …… .git
* now go back to git hub to generate a ssh key if your repo is private .
* go-to-repo …..

[ click on right side cornor profile icon

choose settings

select ssh and gps keys

click on generate a new ssh key and adding it to ssh-agent

Paste the text below, substituting in your GitHub email address.

ssh-keygen -t ed25519 -C "*your\_email@example.com*"

now keep clicking enter until u get keys randomart image

to start the ssh-agent in the background

eval `ssh-agent -s`

**you will get Agent pid**

Add your SSH private key to the ssh-agent. If you created your key with a different name, or if you are adding an existing key that has a different name, replace id\_ed25519 in the command with the name of your private key file.

ssh-add ~/.ssh/id\_ed25519

**identity will be added.**

# **Adding a new SSH key to your GitHub account**

Copy the SSH public key to your clipboard.

If your SSH public key file has a different name than the example code, modify the filename to match your current setup. When copying your key, don't add any newlines or whitespace.

$ clip < ~/.ssh/id\_ed25519.pub

# Copies the contents of the id\_ed25519.pub file to your clipboard

**Tip:** If clip isn't working, you can locate the hidden .ssh folder, open the file in your favorite text editor, and copy it to your clipboard.

Note : This .ssh folder contains id\_ed25519.pub key copy it and paste in key generator ..connection will be established.

**PUSH AND PULL REPOSITORY**

How to push repository to github

git init

git add .

git commit -m “commit”

git pull –-rebase origin main

git push origin main

then the folder gets reposited to github.

If u have created a new repository in github and want to upload new files then,

**push an existing repository from the command line**

many times the git don’t let u upload your folders to repository , then

you should copy your file to main/home by creating new folder

git init

git remote add origin https://github.com/pashaimran1/lio.git

git add .

git commit -m “ commit statement”

git branch -M main

git push -u origin main

If want to upload same files or directory in different branch then

**git push origin <branch name>.**

**GITHUB ACTIONS**

**Generating tokens**

**-) github -> settings -> Developer settings -> personal access tokens.**

**-) generate a new token -> tick all and generate.**

**-) copy the token**

**-) go to repo -> settings -> secrets -> create new -> paste**

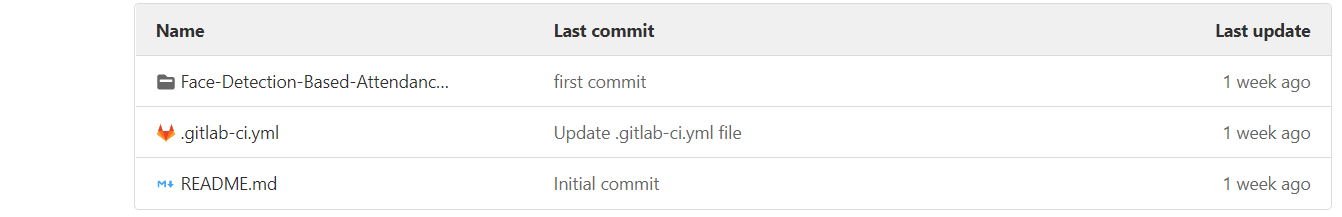
**-) now click on Actions if it wont run then pull the code slightly edit and push ..now action will run.**

**GITLAB**

Before you start, make sure you have:

* A project in GitLab that you would like to use CI/CD for.
* Maintainer or owner access for the project.

1. [Ensure you have runners available](https://docs.gitlab.com/ee/ci/quick_start/#ensure-you-have-runners-available) to run your jobs. If you don’t have a runner, [install GitLab Runner](https://docs.gitlab.com/runner/install/) and [register a runner](https://docs.gitlab.com/runner/register/) for your instance, project, or group.
2. [Create a .gitlab-ci.yml file](https://docs.gitlab.com/ee/ci/quick_start/#create-a-gitlab-ciyml-file) at the root of your repository. This file is where you define your CI/CD jobs.

****

When you commit the file to your repository, the runner runs your jobs. The job results [are displayed in a pipeline](https://docs.gitlab.com/ee/ci/quick_start/#view-the-status-of-your-pipeline-and-jobs).

# Install GitLab Runner manually on GNU/Linux

**sudo apt-get update**

**sudo apt-get curl**

# Download the binary for your system

sudo curl -L --output /usr/local/bin/gitlab-runner https://gitlab-runner-downloads.s3.amazonaws.com/latest/binaries/gitlab-runner-linux-amd64

# Give it permissions to execute

sudo chmod +x /usr/local/bin/gitlab-runner

# Create a GitLab CI user

sudo useradd --comment 'GitLab Runner' --create-home gitlab-runner --shell /bin/bash

# Install and run as service

sudo gitlab-runner install --user=gitlab-runner --working-directory=/home/gitlab-runner

sudo gitlab-runner start

##### **Register Runner**

##### Method

sudo gitlab-runner register --url https://gitlab.com/ --registration-token ($REGISTRATION\_TOKEN) ( copy from runner in gitlab)

sudo gitlab-runner register

Runtime platform arch=amd64 os=linux pid=4840 revision=54944146 version=13.10.0

Running in system-mode.

Enter the GitLab instance URL (for example, https://gitlab.com/):

[https://gitlab.com/]: https://gitlab.com/

Enter the registration token:

[A41xdJC3z2drv7gezYum]: A41xdJC3z2drv7gezYum

Enter a description for the runner:

[imran-VirtualBox]: ci, ssh, shell

Enter tags for the runner (comma-separated):

ci, ssh, dhell

Registering runner... succeeded runner=A41xdJC3

Enter an executor: shell, docker+machine, kubernetes, custom, docker, docker-ssh, parallels, ssh, virtualbox, docker-ssh+machine:

shell

### Create a .gitlab-ci.yml file

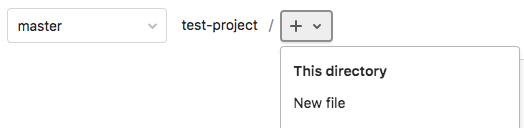
The .gitlab-ci.yml file is a [YAML](https://en.wikipedia.org/wiki/YAML) file where you configure specific instructions for GitLab CI/CD.

In this file, you define:

* The structure and order of jobs that the runner should execute.
* The decisions the runner should make when specific conditions are encountered.

To create a .gitlab-ci.yml file:

1. Go to **Project overview > Details**.
2. Above the file list, select the branch you want to commit to, click the plus icon, then select **New file**:

[](https://docs.gitlab.com/ee/ci/quick_start/img/new_file_v13_6.png)

1. For the **Filename**, type .gitlab-ci.yml and in the larger window, paste this sample code:
2. build-job:
3. stage: build
4. script:
5. - echo "Hello, $GITLAB\_USER\_LOGIN!"
6. test-job1:
7. stage: test
8. script:
9. - echo "This job tests something"
10. test-job2:
11. stage: test
12. script:
13. - echo "This job tests something, but takes more time than test-job1."
14. - echo "After the echo commands complete, it runs the sleep command for 20 seconds"
15. - echo "which simulates a test that runs 20 seconds longer than test-job1"
16. - sleep 20
17. deploy-prod:
18. stage: deploy
19. script:
20. - echo "This job deploys something from the $CI\_COMMIT\_BRANCH branch."

$GITLAB\_USER\_LOGIN and $CI\_COMMIT\_BRANCH are [predefined variables](https://docs.gitlab.com/ee/ci/variables/predefined_variables.html) that populate when the job runs.

1. Click **Commit changes**.

The pipeline starts when the commit is committed

#### .gitlab-ci.yml tips

* If you want the runner to [use a Docker container to run the jobs](https://docs.gitlab.com/ee/ci/docker/using_docker_images.html), edit the .gitlab-ci.yml file to include an image name:
* default:
* image: ruby:2.7.2

This command tells the runner to use a Ruby image from Docker Hub and to run the jobs in a container that’s generated from the image.

This process is different than [building an application as a Docker container](https://docs.gitlab.com/ee/ci/docker/using_docker_build.html). Your application does not need to be built as a Docker container to run CI/CD jobs in Docker containers.

* To validate your .gitlab-ci.yml file, use the [CI Lint tool](https://docs.gitlab.com/ee/ci/lint.html), which is available in every project.
* You can also use [CI/CD configuration visualization](https://docs.gitlab.com/ee/ci/pipeline_editor/index.html#visualize-ci-configuration) to view a graphical representation of your .gitlab-ci.yml file.

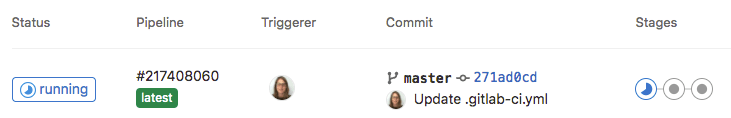
### View the status of your pipeline and jobs

When you committed your changes, a pipeline started.

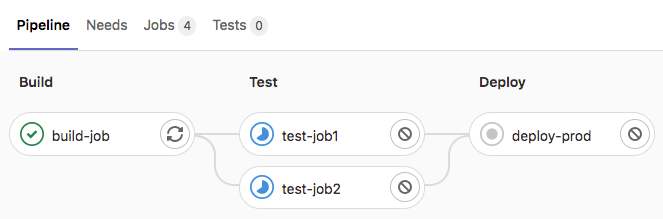
To view your pipeline:

* Go **CI/CD > Pipelines**.

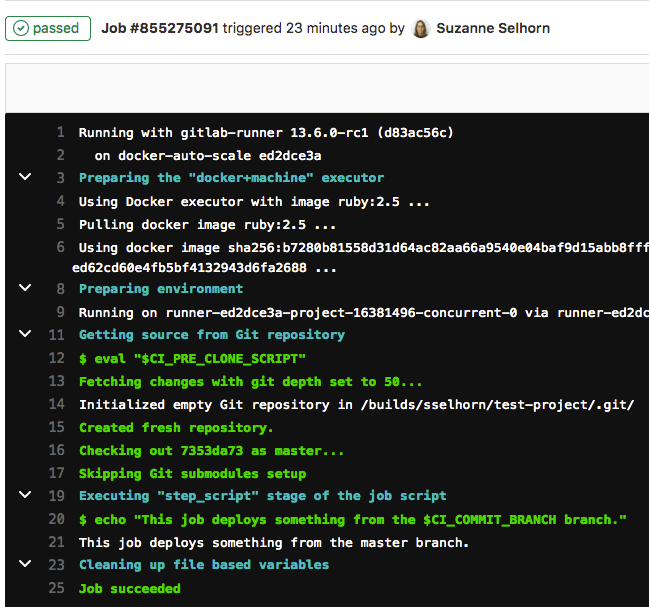
A pipeline with three stages should be displayed:

[](https://docs.gitlab.com/ee/ci/quick_start/img/three_stages_v13_6.png)

* To view a visual representation of your pipeline, click the pipeline ID.

[](https://docs.gitlab.com/ee/ci/quick_start/img/pipeline_graph_v13_6.png)

* To view details of a job, click the job name, for example, deploy-prod.

[](https://docs.gitlab.com/ee/ci/quick_start/img/job_details_v13_6.png)

**DOCKER**

# **Install Docker Engine on Ubuntu**

### **Uninstall old versions**

Older versions of Docker were called docker, docker.io, or docker-engine. If these are installed, uninstall them:

$ sudo apt-get remove docker docker-engine docker.io containerd runc

### **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**

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

$ sudo apt-get update

$ sudo apt-get install \

apt-transport-https \

ca-certificates \

curl \

gnupg \

lsb-release

1. Add Docker’s official GPG key:
2. $ 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**

sudo apt-get update

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

apt-cache madison docker-ce