see point 9 for switching between non-root and root user

(1) ssh to your server via terminal.

ssh [ZC13@EMIMDGXA100GPU4.ccds.io](mailto:ZC13@EMIMDGXA100GPU4.ccds.io) -L 8100:localhost:8100

note 8100 is the port number, it can be any 4-digit number larger than 8100

(2) Download your docker files.

In GitHub, start a new repository (or fork) and clone to local path

Download <https://github.com/zhennongchen/docker_example>, copy to local path

Then push to GitHub

Go back to terminal, git clone this repo, enter your personal access token (<https://docs.github.com/en/authentication/keeping-your-account-and-data-secure/creating-a-personal-access-token>) as your password

(3) ssh on VScode

Open VScode, make sure you have downloaded the extension “remote ssh” and “dev container” from the market.

Click the green button at the left bottom cornerShape, icon, arrow

Description automatically generated

Select “connect to host”

If you are the new user: select “configure SSH hosts”, then select “/Users/zhennongchen/.ssh.config”, add the following in the config file, the first line is the nickname of your host

Graphical user interface, text

Description automatically generated

Then click the green button again, connect to host, select “zhennongchen\_CCDS” which is the nickname.

Now we are in the server from VScode!

(4) Modify docker files

Click File -> open folder -> your repo

Go to subfolder “docker”

- modify dockerfile:

If you are using tensorflow, google “tensorflow docker image”, select one in “tags”. Or you can directly use the exemplar one: FROM tensorflow/tensorflow:2.4.1-gpu

If you are using pytorch, google “pytorch docker image”, select one.

- modify requirements.txt

Add the python package you want to install.

If you want to install a Github repo, then go to dockerfile and add “RUN python3 -m pip install git+http://....”

- modify docker\_build.sh

Re-name your docker image, must be lowercase:

sudo docker build -t docker\_ctp:1.0.0 .

(5) Build the docker image

Open the VScode terminal, type: ./docker\_build.sh

Go to terminal, type: “sudo docker image ls” to check whether the image has been installed.

(6) Build the docker container

- Modify docker\_run.sh

Define your HOST\_WORK\_DIR (your repo folder, lowercase) and HOST\_DATA\_DIR, change the CONTAINER\_WORKDIR as the same name as your HOST\_WORK\_DIR

Also change the port number

Also change the last line as your docker image name

Save

Then go to terminal, run ./docker\_run.sh

You should see a red Tensorflow (only if you use FROM tensorflow...)

A picture containing text

Description automatically generated

If you are the new user, you need to add docker user, see details in <https://docs.docker.com/engine/install/linux-postinstall/>, type these commands in terminal

Go back to VScode, click green button, select “attach to running container”,

Click green button again, select “Open container configuration file”.

Then go to subfolder “vscode\_config/docker\_config.json”, change the workspaceFolder and remoteUser (all lowercase for this one), then copy everything and paste to the container configuration file.

Then close the VScode window. Open VScode window again.

Click green button, select “attach to running container”

(If you failed to attach to running container, then you did something wrong in the previous step. To correct, click F1, then type remote-containers open attached container configure file, then correct the typo in the container configuration file)

Now if everything is correct, you should be in the container. Check the extension -> container, it should have many extensions including Jupyter notebook installed.

(7) Work in VScode (for script editing and Jupyter notebook)

Now you can use VScode with the Jupyter notebook feature.

Open any script, you should see there are cells. Click Run cells will give you an interactive UI showing the results.

(8) Work in Jupyter Lab (for DL training, long-time running program)

Go to terminal, make sure you are now in the docker container (you run the ./docker\_run.sh before in the terminal), then type: jupyter lab --port=8120 --no-browser --ip=0.0.0.0

(GPU4 8100, GPU7 8110, GPU8 8120, GPU5 8150, GPU6 8160, GPU2 8200, GPU3 8300, GPU1 8000)

Copy token, go to browser (Safari), type localhost:8100/, paste the token.

Pick Other->terminal, cd to workspace to see your code, then you can run the code here

You can open as many terminals as you want, just click the + button on the left top corner and open another terminal.

(9) Work in container as root or non-root user:

As non-root user: it is recommended to work in container (i.e., run codes) as the non-root user otherwise it may cause lots of troubles.

Our entrypoint.sh last line**: su - ${CONTAINER\_UNAME}** ensures we enter the container as a non-root user. In the docker container, type “whoami”, it should return something not “root”

As a root user: open a new terminal and log onto the server from it. Use “docker container ls” to get the NAME of current docker. Type command **“docker exec -it -u 0 NAME bash”.** -u ensures the entry as root. Now we are in the docker as the root. Type whoami should give us “root”. Here we can do apt install/update. When it’s done, use control+A+D to exit and close the terminal.

Last, don’t forget to push the updates to GitHub

Go back to terminal, go to the repo folder

git add .

git status

git commit -m ‘message’

(git remote add origin ‘https:XXXXX’)

git push -u origin master

type personal access token as the password (personal access token: ghp\_5hyRhYvnZo2oHfrOD03aygqqbpkycP1U9Q8H)

Check this repo in GitHub