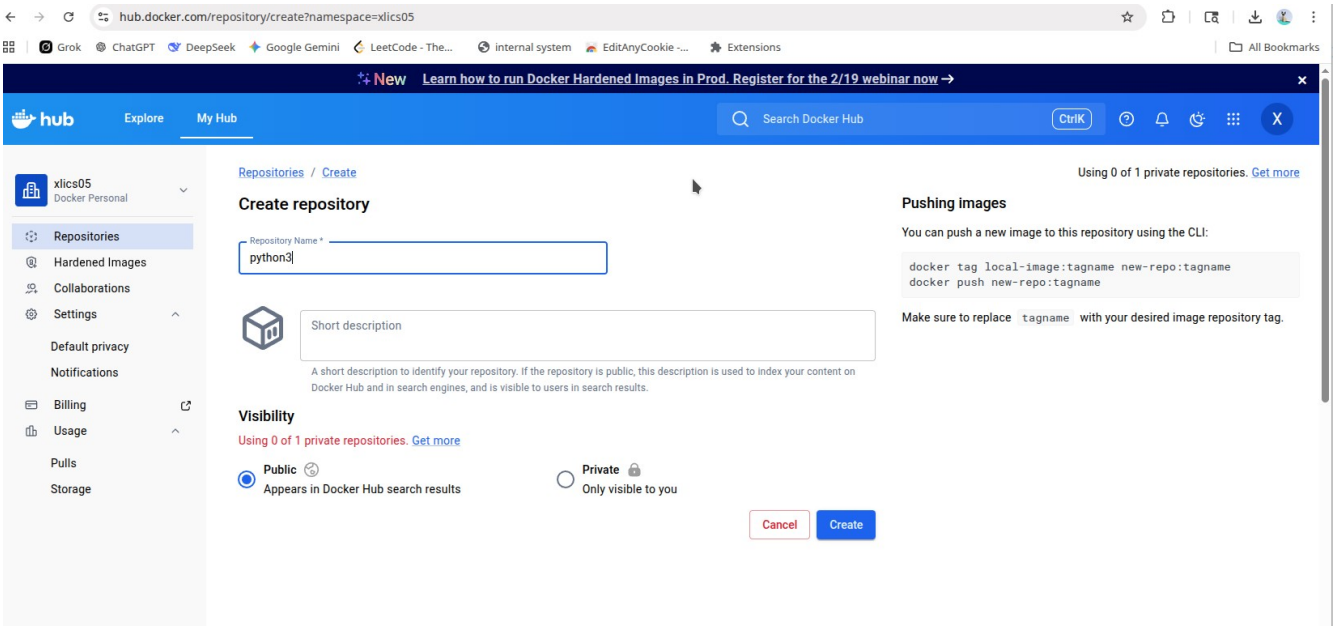
Build Docker Image with Linux Environmental Variables

Created by: Kevin Li

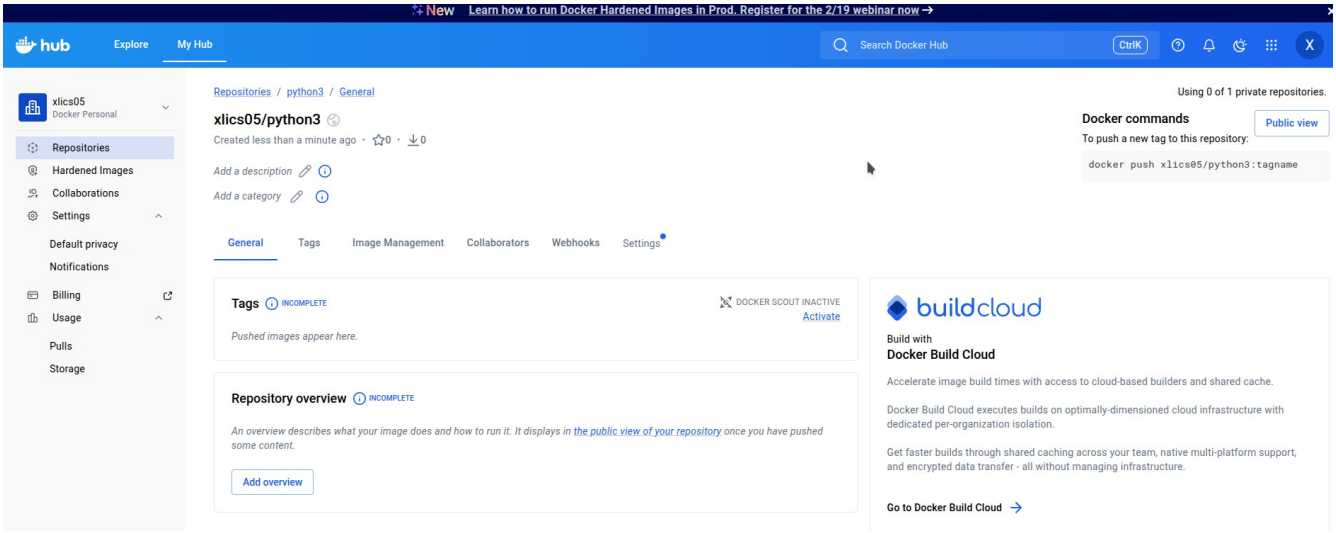
Created at: 02/18/2026

1/ Create a https://hub.docker.com/ account

2/ Create a repo called “Python3”



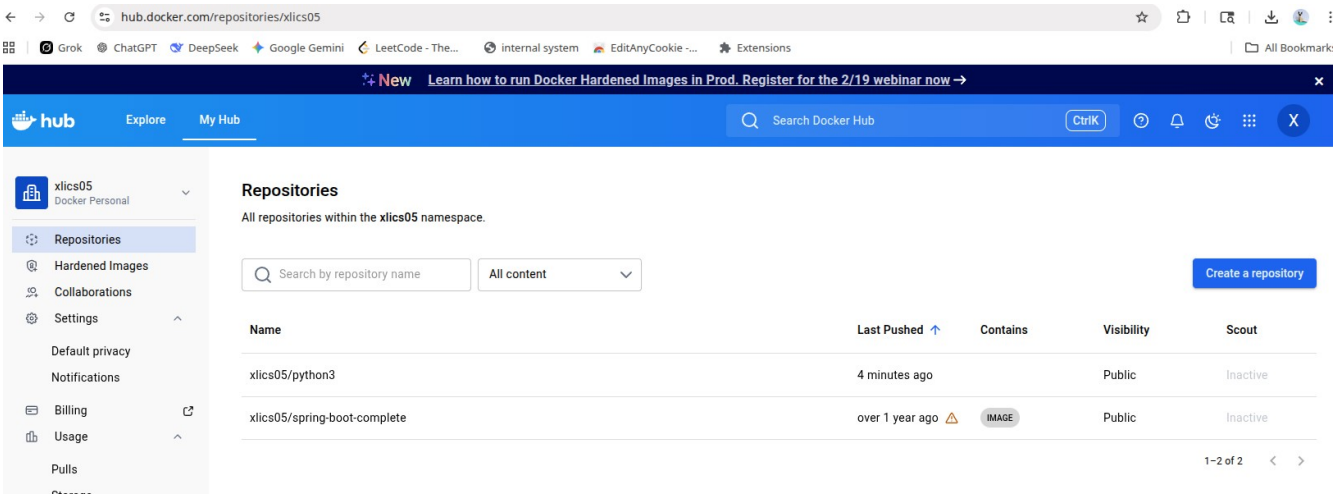
Next, you will see the following screen,



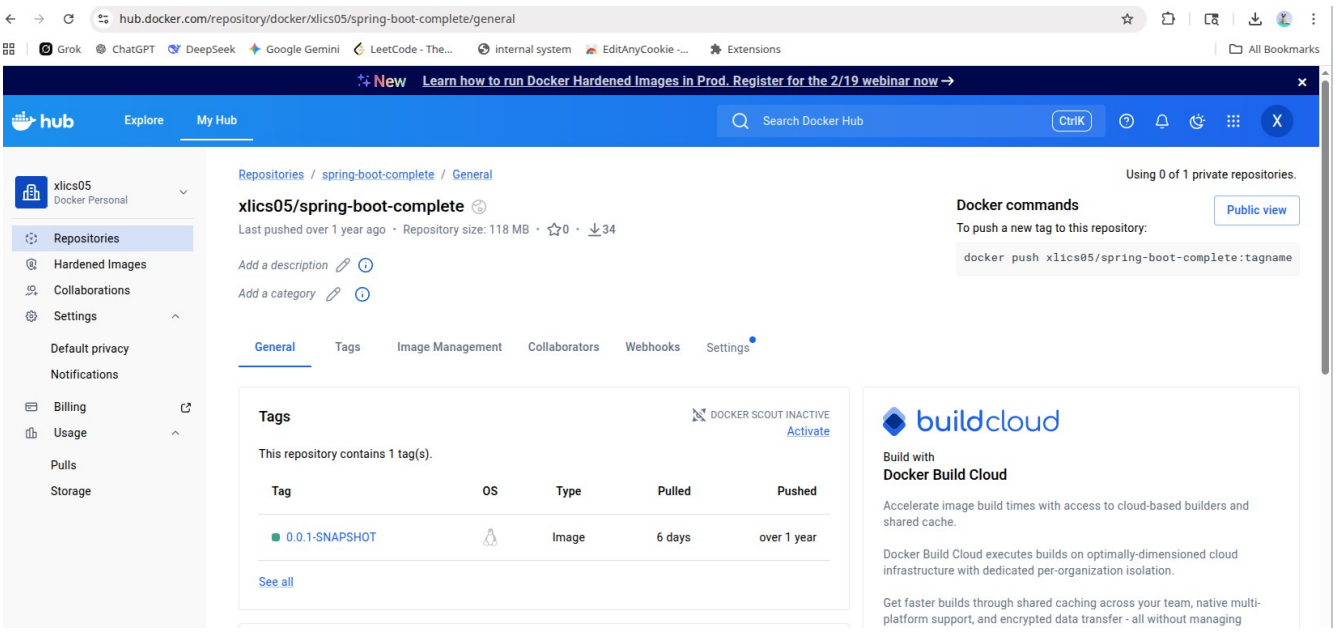
On the top right corner, it shows you how to push your built image to the docker.io. But we will have to do a `docker login` first before we could push to the above repository.

docker push xlics05/python3:tagname

After you create it, you will see I have two repos now,



By clicking my previous repo for springboot, you will see something like the following,



and here you will see this,

```
docker push xlics05/spring-boot-complete:tagname
```

`xlics05` → namespace

`spring-boot-complete:tagnam` → image

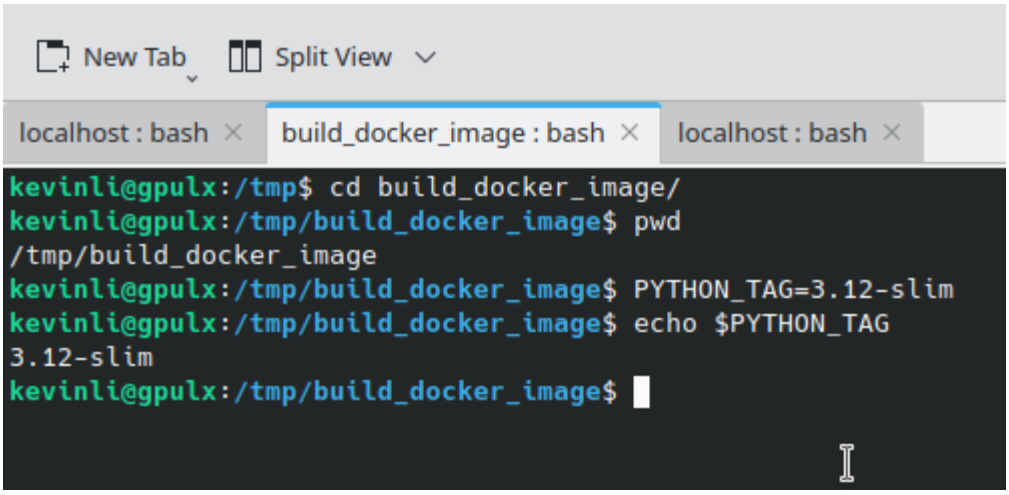
3/ Let us prepare a Dockerfile with ARG passing values from Linux CLI

```
ARG PYTHON_TAG
FROM python:$PYTHON_TAG
ENTRYPOINT ["/bin/bash"]
```

4/ Build the image

The following is the `docker build` command I would like to type in the Linux CLI.

Let us open linux terminal and set a variable with of the following,


New Tab
Split View
localhost : bash
build_docker_image : bash
localhost : bash
kevinli@gpulx:/tmp$ cd build_docker_image/
kevinli@gpulx:/tmp/build_docker_image$ pwd
/tmp/build_docker_image
kevinli@gpulx:/tmp/build_docker_image$ PYTHON_TAG=3.12-slim
kevinli@gpulx:/tmp/build_docker_image$ echo $PYTHON_TAG
3.12-slim
kevinli@gpulx:/tmp/build_docker_image$

The whole transcript of building the docker image is provided below,

```
kevinli@gpulx:/tmp/build_docker_image$ touch Dockerfile
kevinli@gpulx:/tmp/build_docker_image$ vi Dockerfile
kevinli@gpulx:/tmp/build_docker_image$ docker build -t python3:test --build-arg
PYTHON_TAG=$PYTHON_TAG .
DEPRECATED: The legacy builder is deprecated and will be removed in a future
release.
            Install the buildx component to build images with BuildKit:
            https://docs.docker.com/go/buildx/

Sending build context to Docker daemon  2.048kB
Step 1/3 : ARG PYTHON_TAG
Step 2/3 : FROM python:$PYTHON_TAG
3.12-slim: Pulling from library/python
0c8d55a45c0d: Already exists
690eaffcf0e9: Pulling fs layer
9395e1d7be50: Pulling fs layer
4948ee383266: Pulling fs layer
4948ee383266: Verifying Checksum
4948ee383266: Download complete
690eaffcf0e9: Verifying Checksum
690eaffcf0e9: Download complete
9395e1d7be50: Verifying Checksum
9395e1d7be50: Download complete
690eaffcf0e9: Pull complete
9395e1d7be50: Pull complete
4948ee383266: Pull complete
Digest: sha256:9e01bf1ae5db7649a236da7be1e94ffbbbdd7a93f867dd0d8d5720d9e1f89fab
Status: Downloaded newer image for python:3.12-slim
---> b3b92273ebb4
Step 3/3 : ENTRYPOINT ["/bin/bash"]
---> Running in 61520ee925b2
---> Removed intermediate container 61520ee925b2
---> cbf941185644
Successfully built cbf941185644
Successfully tagged python3:test
```

```
docker build -t python3:test --build-arg PYTHON_TAG=$PYTHON_TAG .
```

5/ Check your image is proper

```
kevinli@gpulx:/tmp/build_docker_image$ docker image ls |grep python3
python3                             test              cbf941185644   About a minute
ago   119MB
```

Now image exists…

Next, we need to check the layer of image we grab or base on which is the pre-built python:3.12-slim is working as it is.

Let us run the image,

```
kevinli@gpulx:/tmp/build_docker_image$ docker run -it python3:test
root@cecb91a62544:/# ls
bin  boot  dev  etc  home  lib  lib64  media  mnt  opt  proc  root  run  sbin  srv
sys  tmp  usr  var
```

```
root@cecb91a62544:/# which python
/usr/local/bin/python
```

```
root@cecb91a62544:/# python --version
Python 3.12.12
```

Now check with `docker ps` in another tab, and you will see the python3:test container is running.

```
kevinli@gpulx:~/git/localhost$ docker ps
CONTAINER ID   IMAGE           COMMAND           CREATED         STATUS
PORTS                                       NAMES
a2c9a5ef8eb6   python3:test    "/bin/bash"       8 seconds ago   Up 6 seconds
romantic_visvesvaraya
b086042ce985   my-python-app   "python app.py"   47 hours ago    Up 37 minutes
0.0.0.0:5002->5000/tcp, [::]:5002->5000/tcp   my-running-app
kevinli@gpulx:~/git/localhost$
```

If you press ctrl+c to terminate it, and you will not see it, because you must keep the container running.

```
kevinli@gpulx:/tmp/build_docker_image$ docker run -dit --name python3-test-
container  python3:test
ee9947819b051d9ec6a8f716c7ccfc434e675ebaba30e0198e80b515e7392219
```

It exists out, but it is running in the background,

Let us use the same terminal, and check `docker ps`

```
kevinli@gpulx:/tmp/build_docker_image$ docker ps
CONTAINER ID   IMAGE           COMMAND           CREATED          STATUS
PORTS                                     NAMES
ee9947819b05   python3:test    "/bin/bash"       34 seconds ago   Up 33 seconds
python3-test-container
```

Since it is not a one-shot mode, and it continues to run, because of the option/flag `-d`.

Now, what if we want to exec into the container, we can still use the same terminal or a different one to access the container with its id.

```
kevinli@gpulx:/tmp/build_docker_image$ docker exec -it ee9947819b05 /bin/bash
root@ee9947819b05:/# ls
bin  boot  dev  etc  home  lib  lib64  media  mnt  opt  proc  root  run  sbin  srv
sys  tmp  usr  var
root@ee9947819b05:/# which python
/usr/local/bin/python
root@ee9947819b05:/# python --version
Python 3.12.12
```

6/ Push the image to your repo

If you are unfamiliar with the usage, remember, everything today is self-contained. Do you need a tutorial or instruction? For new comers, yes, for senior, no.

```
kevinli@gpulx:/tmp/build_docker_image$ docker login --help
Usage:  docker login [OPTIONS] [SERVER]

Authenticate to a registry.
Defaults to Docker Hub if no server is specified.

Options:
 -p, --password string   Password or Personal Access Token (PAT)
     --password-stdin    Take the Password or Personal Access Token (PAT) from
stdin
 -u, --username string   Username
```

In my case, I did set a username and password for docker.io repo, although I use google Oauth authentication which is my gmail authentication.

```
kevinli@gpulx:/tmp/build_docker_image$ docker login docker.io -u xlics05

i Info → A Personal Access Token (PAT) can be used instead.
         To create a PAT, visit https://app.docker.com/settings


Password:

WARNING! Your credentials are stored unencrypted in
'/home/kevinli/.docker/config.json'.
Configure a credential helper to remove this warning. See
https://docs.docker.com/go/credential-store/

Login Succeeded
```

Once we login, we will need to execute the following two commands,

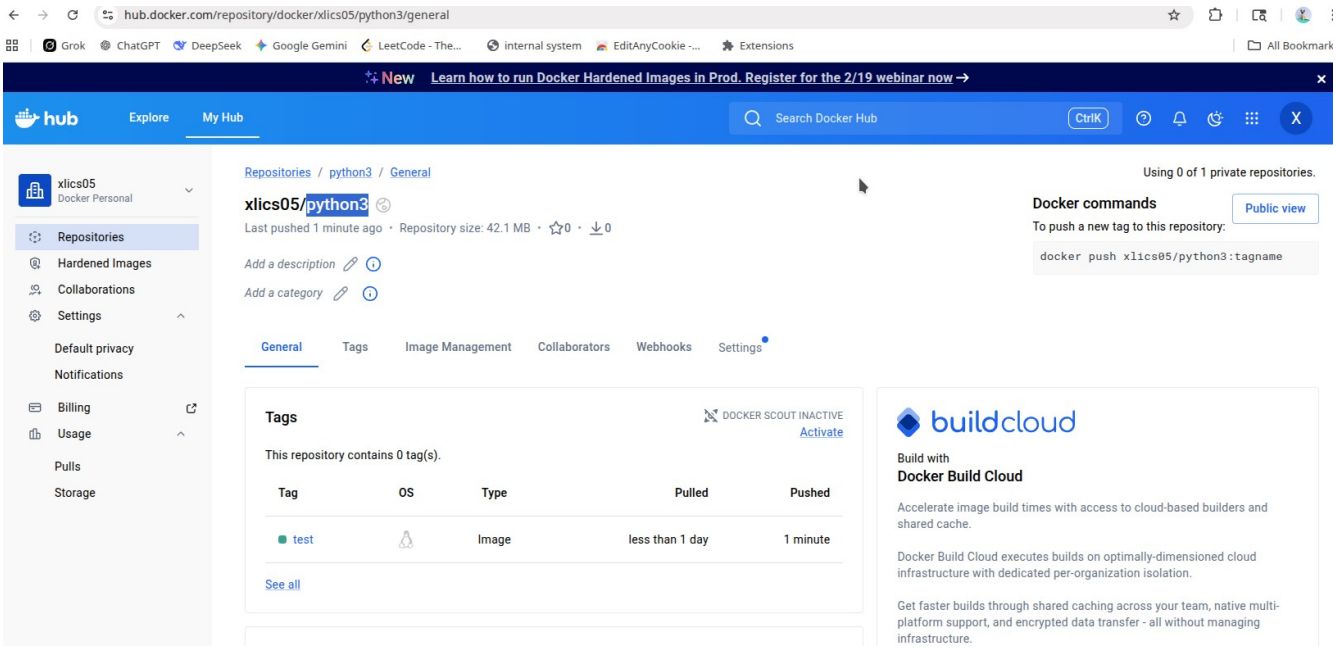
`docker tag python3:test docker.io/xlics05/python3/python3:test``

`docker push xlics05/python3/python3:test`

In my terminal, it looks like this,

```
kevinli@gpulx:/tmp/build_docker_image$ docker tag python3:test
docker.io/xlics05/python3:test
kevinli@gpulx:/tmp/build_docker_image$ docker push xlics05/python3:test
The push refers to repository [docker.io/xlics05/python3]
e606afe81a9a: Pushed
50b7356375f2: Pushed
2cb59db770d1: Pushed
a8ff6f8cbdfd: Pushed
test: digest:
sha256:3bca0dc0a1d32afbc4d2cb02ba2bb058a758d8f4972301669b391daaa64e2bdc size: 1159
```

7/ Check the image was pushed to the docker.io



Now we are all good.