

## Containerization Technologies – TD 7

### Step 0: GitLab

The connection is ok!

### Step 1: Simple CI

Here's my '.gitlab-ci.yml' file I created from the gitlab CI quickstart:

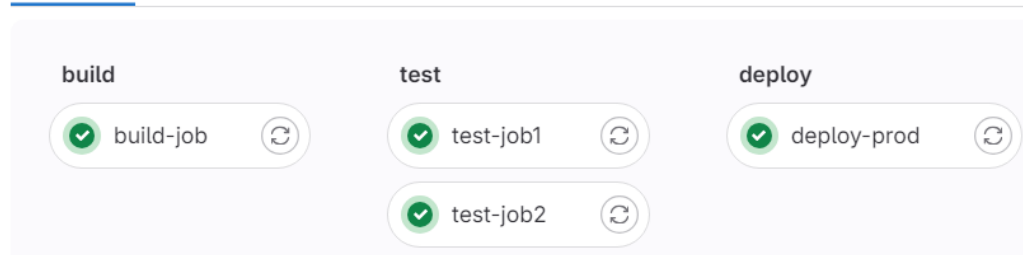
```
Containerization Technologies > TD 7 > 📄 .gitlab-ci.yml
1  build-job:
2    stage: build
3    script:
4      - echo "Hello, $GITLAB_USER_LOGIN!"
5
6  test-job1:
7    stage: test
8    script:
9      - echo "This job tests something"
10
11 test-job2:
12   stage: test
13   script:
14     - echo "This job tests something, but takes more time than test-job1."
15     - echo "After the echo commands complete, it runs the sleep command for 20 seconds"
16     - echo "which simulates a test that runs 20 seconds longer than test-job1"
17     - sleep 20
18
19 deploy-prod:
20   stage: deploy
21   script:
22     - echo "This job deploys something from the $CI_COMMIT_BRANCH branch."
23   environment: production
```

Then, in GitLab my pipeline look like as in the TD:

#### first '.gitlab-ci.yml' file

✓ Passed loevan le-querneec created pipeline for commit 0b878176 📄 finished 1 minute ago  
For main  
latest 🕒 4 Jobs ⌚ 35 seconds, queued for 2 seconds

Pipeline Needs Jobs 4 Tests 0



## Step 2: Build your application

I can see the lines about Docker too:

```
✓ 3 Preparing the "docker" executor
  4 Using Docker executor with image ubuntu:22.04 ...
  5 Pulling docker image ubuntu:22.04 ...
```

Here's the code I used from the gitlab documentation:

```
Containerization Technologies > TD 7 > 🏠 .gitlab-ci.yml
1  build:
2    stage: build
3    image:
4      name: gcr.io/kaniko-project/executor:v1.14.0-debug
5      entrypoint: [""]
6    script:
7      - /kaniko/executor
8        --context "${CI_PROJECT_DIR}"
9        --dockerfile "${CI_PROJECT_DIR}/Dockerfile.app"
10       --destination "${CI_REGISTRY_IMAGE}"
```

(I removed the rules part for now)

But wait, let's add the line to create the 'config.json' file:

```
script:
- /kaniko/executor
  --context "${CI_PROJECT_DIR}"
  --dockerfile "${CI_PROJECT_DIR}/Dockerfile"
  --destination "${CI_REGISTRY_IMAGE}:${CI_COMMIT_TAG}"
- echo "{\"auths\":{\"${CI_REGISTRY}\":{\"auth\":\"$(printf \"%s:%s\" \"${CI_REGISTRY_USER}\" \"${CI_REGISTRY_PASSWORD}\")\"}}}"
```

(I can't show the entire line but everything is here)

Now, let's set up the variables. First, the variables asked in the TD (the second script):

- CI\_REGISTRY is the link provided (<https://index.docker.io/v1/>);
- CI\_REGISTRY\_USER is my Docker username (blxucreep);
- CI\_REGISTRY\_PASSWORD is an access token I created on Docker.


























I also need to set up two other variables from the first script I used:

- CI\_PROJECT\_DIR is the directory of the project (for me, it's ./app);
- CI\_REGISTRY\_IMAGE is the name of the image (I named it 'blxucreep/app').

I changed a line because my application Dockerfile is named 'Dockerfile.app':

```
9  --dockerfile "${CI_PROJECT_DIR}/Dockerfile.app"
```

Finally, I have these variables:

CI/CD Variables </> 5				Reveal values	Add variable
↑ Key	Value	Environments	Actions		
CL_PROJECT_DIR  <span>Expanded</span>	***** 	All (default) 	 		
CL_REGISTRY  <span>Expanded</span>	***** 	All (default) 	 		
CL_REGISTRY_IMAGE  <span>Expanded</span>	***** 	All (default) 	 		
CL_REGISTRY_PASSWORD  <span>Masked</span> <span>Expanded</span>	***** 	All (default) 	 		
CL_REGISTRY_USER  <span>Expanded</span>	***** 	All (default) 	 		

Let's commit & push, and see how our build goes:

```
1 Running with gitlab-runner 16.8.0 (c72a09b6)
2   on runner-01 -H-vAzzu-, system ID: s_648b9ca6d4fe
3 Preparing the "docker" executor
4 Using Docker executor with image gcr.io/kaniko-project/executor:v1.14.0-debug ...
5 Pulling docker image gcr.io/kaniko-project/executor:v1.14.0-debug ...
6 Using docker image sha256:a4404d1ebd45121579c48012979e8e8d85ffffba872a8b1e85e21a95bd67ffb86475128a41def9 ...
7 Preparing environment
```


.


.


.


```
68 INFO[0011] Taking snapshot of full filesystem...
69 INFO[0012] CMD ["python", "app.py"]
70 INFO[0012] Pushing image to blxucreepp/app
71 INFO[0016] Pushed index.docker.io/blxucreepp/app@sha256:919b512ffa10788c414a02dd65...
72 $ echo "{\"auths\":{\"${CI_REGISTRY}\":{\"auth\":\"${printf \"%s:%s\" \"${CI_REGISTRY_USER}\" \"${CI_REGISTRY_PASSWORD}\"}}}}\" | docker login --username ${CI_REGISTRY_USER} --password-stdin ${CI_REGISTRY}
73 Cleaning up project directory and file based variables
74 Job succeeded
```

And it's a success! Let's see if our image is available in Docker hub:

 **dockerhub** Explore Repositories Organizations

blxucreepp 




All Content 

Create repository


blxucreepp / app

Contains: Image | Last pushed: 5 minutes ago

 Security unknown

☆ 0

↓ 0

 Public

## Step 3: Expected behaviour

Two things to say:

- First, let's change the variable `CI_REGISTRY_IMAGE` by `'blxucreep/cont-tech-app'` (something more meaningful than simply `'blxucreep/app'`);
- Now we can add the rules part to deploy our image with the last commit tag.


See the updated `'.gitlab-ci.yml'` file:


```
6  script:
7    - /kaniko/executor
8      --context "${CI_PROJECT_DIR}"
9      --dockerfile "${CI_PROJECT_DIR}/Dockerfile.app"
10     --destination "${CI_REGISTRY_IMAGE}:${CI_COMMIT_TAG}"
11     - echo "{\"auths\":{\"${CI_REGISTRY}\":{\"auth\":\"$(pri
12  rules:
13     - if: $CI_COMMIT_TAG
```


I pushed with `'my-custom-tag'`:


```
PS C:\Users\Loeva\OneDrive\Bureau\ESILV\4 cycle Ingé DIA\Semestre 8\~ programmation\Containerization Technologies\TD 7> git push --set-upstream origin main my-custom-tag
Enter passphrase for key '/c/Users/Loeva/.ssh/id_rsa':
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 16 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 889 bytes | 889.00 KiB/s, done.
Total 3 (delta 1), reused 0 (delta 0), pack-reused 0 (from 0)
To gitlab.sre.paris:td7-td7-loevan-le-quernec.git
 85d33ee..23ffb2a  main -> main
* [new tag]         my-custom-tag -> my-custom-tag
```

Let's see the build:



 Passed  
00:00:22  
14 minutes ago

commit tag part for the image + 'image.txt' file  
#3985 my-custom-tag 85d33ee4   
latest









It's seems ok, and my image is in Docker hub with the right tag:

**blxucreep/cont-tech-app**   
Updated less than a minute ago  
This repository does not have a description 

**Tags**  
This repository contains 2 tag(s).

Tag	OS	Type	Pulled	Pushed
 latest		Image	---	a minute ago
 my-custom-tag		Image	---	13 minutes ago

[See all](#)

My file 'image.txt':

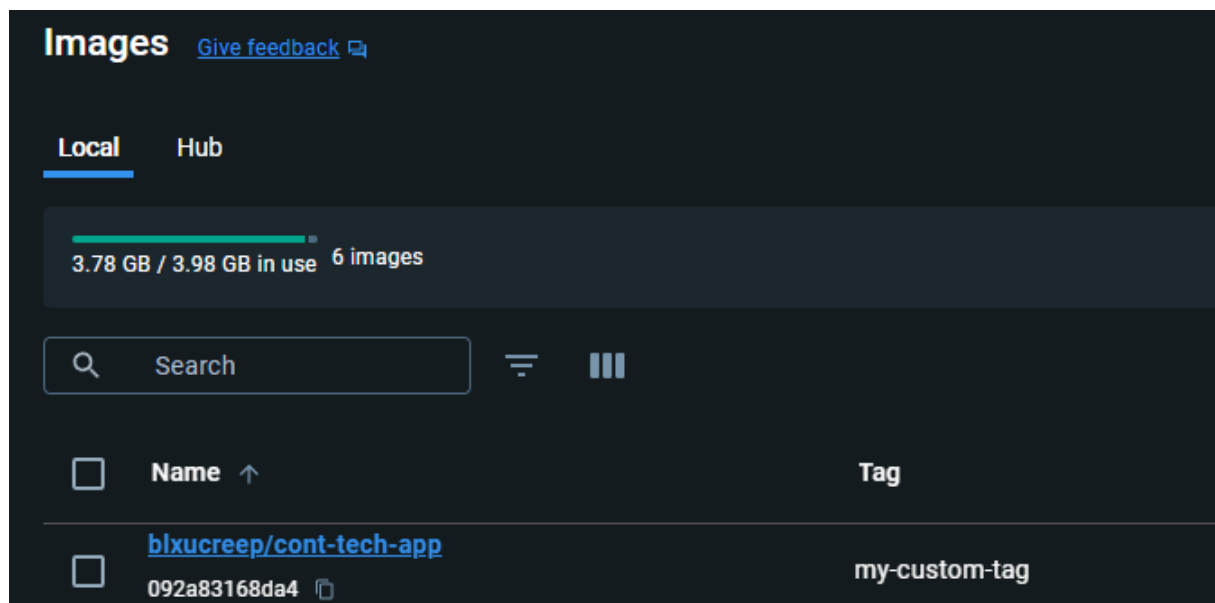
```
Containerization Technologies > TD 7 > image.txt
1 blxucreep/cont-tech-app
```

And I can pull my image from Docker hub:

```
blxucreep@Blxucreep:/mnt/c/Users/Loeva/OneDrive/Bureau/ESILV/A4 cycle ingé DIA/Semestre 8/~ programmation/
Containerization Technologies/TD 7$ docker pull $(cat image.txt):my-custom-tag
my-custom-tag: Pulling from blxucreep/cont-tech-app
e1caac4eb9d2: Already exists
51d1f07906b7: Already exists
07b545b886b2: Already exists
f86bc27bff61: Already exists
1a56bca2cd81: Already exists
0f9f427eaa9f: Pull complete
53229dda639c: Pull complete
4735bbe9e918: Pull complete
Digest: sha256:91ff4b9e347a5c40f82da3731bda7253d265196f9f28255c90c71ec586b31f42
Status: Downloaded newer image for blxucreep/cont-tech-app:my-custom-tag
docker.io/blxucreep/cont-tech-app:my-custom-tag

What's Next?
View a summary of image vulnerabilities and recommendations → docker scout quickview blxucreep/cont-tech
-app:my-custom-tag
```

In Docker desktop:



My image is ready to use!