LE QUERNEC Loévan CDOF2

Containerization Technologies – TD 8

Step 0: GitLab

The connection is ok!

Step 1: Kics

Here is my Dockerfile:

```
Containerization Technologies > TD 8 > Dockerfile > ...
# use an official nginx image
FROM nginx:1.24

# copy the HTML file at /usr/share/nginx/html
COPY index.html /usr/share/nginx/html

# expose port 80
EXPOSE 80
```

And I run Kics on it (in PowerShell). First I defined a environment variable (for the path) and then I run the scan with kics.

The commands:

- **'$directory = "C:\Users\Loeva\OneDrive\Bureau\ESILV\A4 cycle ingé DIA\Semestre 8\~ programmation\Containerization Technologies\TD 8"'**
- '--rm' is removing the container when the scan is done;
  '-v' is creating the volume with the path;
  '-it' is for the interactive terminal;
  '-p' means '--path' for the scan;
  'docker run --rm -v ${directory}:/path -it checkmarx/kics scan -p /path'.

Here are the results of the scan (found 2 vulnerabilities):

```
Healthcheck Instruction Missing, Severity: LOW, Results: 1
Description: Ensure that HEALTHCHECK is being used. The HEALTHCHECK instruction tells Docker how t
o test a container to check that it is still working
Platform: Dockerfile
CWE: 710
Learn more about this vulnerability: https://docs.kics.io/latest/queries/dockerfile-queries/b03a74
8a-542d-44f4-bb86-9199ab4fd2d5

        [1]: ../../path/Dockerfile:2

                001: # use an official nginx image
                002: FROM nginx:1.24
                003:
```

```
Missing User Instruction, Severity: HIGH, Results: 1
Description: A user should be specified in the dockerfile, otherwise the image will run as root
Platform: Dockerfile
CWE: 250
Learn more about this vulnerability: https://docs.kics.io/latest/queries/dockerfile-queries/fd54f2
00-402c-4333-a5a4-36ef6709af2f

        [1]: ../../path/Dockerfile:2

                001: # use an official nginx image
                002: FROM nginx:1.24
                003:



Results Summary:
HIGH: 1
MEDIUM: 0
LOW: 1
INFO: 0
TOTAL: 2
```

Step 2: CI

Here‘s my ‘.gitlab-ci.yml’ file (all the variables have been created):

Containerization Technologies > TD 8 > .gitlab-ci.yml

```yaml
stages:
  - kics
  - build

variables:
  DOCKERFILE_PATH: Dockerfile

kics:
  stage: kics
  image:
    name: checkmarx/kics:latest
    entrypoint: [""]
  script:
    - |
      echo "Scanning Dockerfiles..."
      kics scan --path $DOCKERFILE_PATH
  only:
    - branches

build:
  stage: build
  image:
    name: gcr.io/kaniko-project/executor:v1.14.0-debug
    entrypoint: [""]
  script:
    - /kaniko/executor
      --context "${CI_PROJECT_DIR}"
      --dockerfile "${CI_PROJECT_DIR}/Dockerfile"
      --destination "${CI_REGISTRY_IMAGE}:${CI_COMMIT_SHORT_SHA}"
    - echo "{\"auths\":{\"${CI_REGISTRY}\":{\"auth\":\"$(printf "%s:%s" "${CI_REGISTRY_USE
```

There are 2 stages, one is running a scan of the Dockerfile with kics and the other one is building and pushing the image on my Docker hub with kaniko.

Then we can see the results of the scan:

```
Scanning Dockerfiles...
                    .0M0.
                    0MMMx
                    ;NMX;
                     ...          ...            ....
WMMMd     cWMMM0.  KMMM0      ;xKWMMMMN0c.     ,xXMMMMMWXkc.
WMMMd   .0MMMN:    KMMM0    :XMMMMMMMMMMMWl   xMMMMMWMMMMMl
WMMMd  lWMMM0.     KMMM0   xMMMMKc...'lXMk   ,MMMMx   .;dXx
WMMMd.0MMMX;       KMMM0  cMMMMd        '    'MMMMNl'
WMMMNWMMMMl        KMMM0  0MMMN                oMMMMMMMXkl.
WMMMMMMMMMMo       KMMM0  0MMMX                 .ckKWMMMMMM0.
WMMMMWokMMMMk      KMMM0  oMMMMc               .     .:0MMMM0
WMMMK.  dMMMM0.    KMMM0   KMMMMx'    ,kNc   :W0c.    .NMMMX
WMMMd    cWMMMX.   KMMM0    kMMMMMWXNMMMMMd .WMMMMWKO0NMMMMl
WMMMd     ,NMMMN,  KMMM0     'xNMMMMMMMNx,   .l0WMMMMMMMWk,
xkkk:      ,kkkkx  okkkl        ;xKXKx;          ;dOKKkc
Scanning with Keeping Infrastructure as Code Secure v1.7.13
Preparing Scan Assets:  \Preparing Scan Assets: DoneExecuting queries: [---------------------------------->_____________________________________] 48.81
_______] 57.14%Executing queries: [--------------------------------------------->__________________________] 64.29%Executing queries: [---------------
------------------------------------------------------>_______________] 78.57%Executing queries: [-----------------------------------------------------
------------------------->______] 91.67%Executing queries: [------------------------------------------------------------------->] 98.81%Executing
0.00%
Healthcheck Instruction Missing, Severity: LOW, Results: 1
Description: Ensure that HEALTHCHECK is being used. The HEALTHCHECK instruction tells Docker how to test a container to check that it is still working
Platform: Dockerfile
CWE: 710
Learn more about this vulnerability: https://docs.kics.io/latest/queries/dockerfile-queries/b03a748a-542d-44f4-bb86-9199ab4fd2d5
        [1]: Dockerfile:2
                001: # use an official nginx image
                002: FROM nginx:1.24
                003:
Missing User Instruction, Severity: HIGH, Results: 1
Description: A user should be specified in the dockerfile, otherwise the image will run as root
Platform: Dockerfile
CWE: 250
Learn more about this vulnerability: https://docs.kics.io/latest/queries/dockerfile-queries/fd54f200-402c-4333-a5a4-36ef6709af2f
        [1]: Dockerfile:2
                001: # use an official nginx image
                002: FROM nginx:1.24
                003:
Results Summary:
HIGH: 1
MEDIUM: 0
LOW: 1
INFO: 0
TOTAL: 2
Cleaning up project directory and file based variables
```

There are 2 problems:

- The HEALTHCHECK instruction is missing;
- There isn't any USER specified in the Dockerfile.

Step 3: Fixing the error

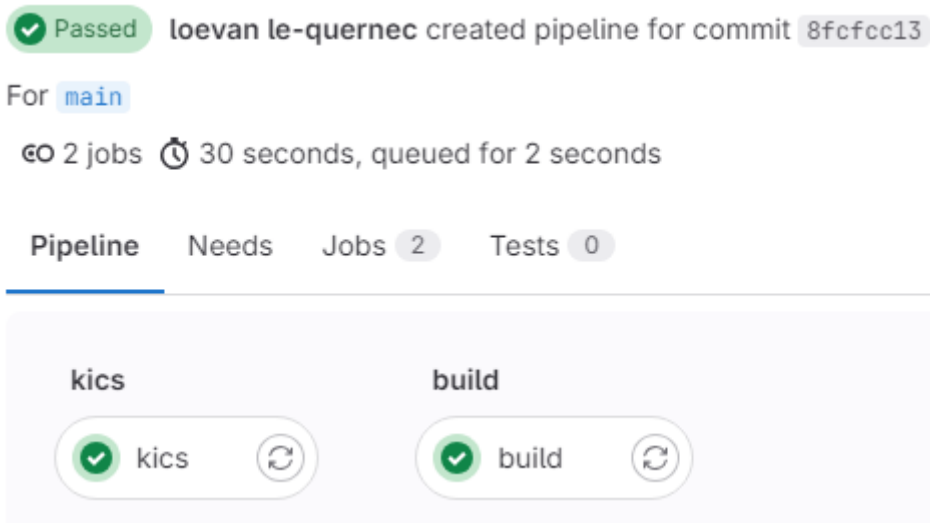
Let's fix this:

Containerization Technologies > TD 8 > Dockerfile > ...

```
# use an official nginx image
FROM nginx:1.24

# copy the HTML file at /usr/share/nginx/html
COPY index.html /usr/share/nginx/html

# expose port 80
EXPOSE 80

# healthcheck instruiction
HEALTHCHECK --interval=30s --timeout=30s --start-period=5s --retries=3 \
    CMD curl -f http://localhost/ || exit 1

# specify an user
USER nginx
```

I also added a simple .html file in order for the build job to succeed.

The pipeline in GitLab:



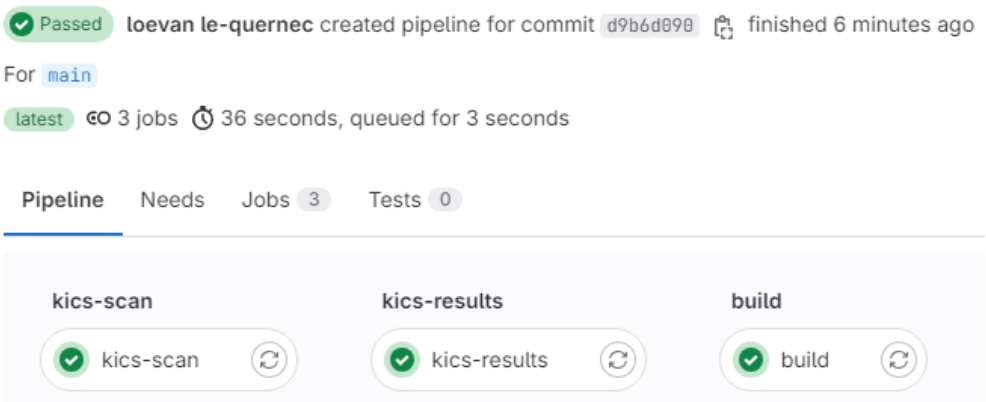
Bonus 1: Split the scan and the results

I splited the job into 2:

- ‘kics-scan’ scans the Dockerfile then saves the files stored in the directory I specified;
- ‘kics-results’ shows the results of the scan with the alpine image and a simple ‘cat’ command (the ‘dependencies’ is used to retrieve the artifacts from the previous job).

```yaml
variables:
  DOCKERFILE_PATH: Dockerfile
  KICS_RESULTS_PATH: kics-results

kics-scan:
  stage: kics-scan
  image:
    name: checkmarx/kics:latest
    entrypoint: [""]
  script:
    - |
      echo "Scanning Dockerfiles..."
      kics scan --path $DOCKERFILE_PATH --output-path $KICS_RESULTS_PATH
  artifacts:
    paths:
      - $KICS_RESULTS_PATH/results.json
  only:
    - branches

kics-results:
  stage: kics-results
  image: alpine:latest
  script:
    - |
      echo "Showing KICS scan results..."
      cat $KICS_RESULTS_PATH/results.json
  dependencies:
    - kics-scan
  only:
    - branches
```

The pipeline in GitLab:



Bonus 2: Split the build and the push

I didn't achieved to do this part, but I found a way to save my image in a .tar file, in order to do the Trivy bonus part. Here the line I added in the build job:

```
--tar-path "${CI_PROJECT_DIR}/image.tar"
```

(in the script)

Then I save my 'image.tar' file as an artifact:

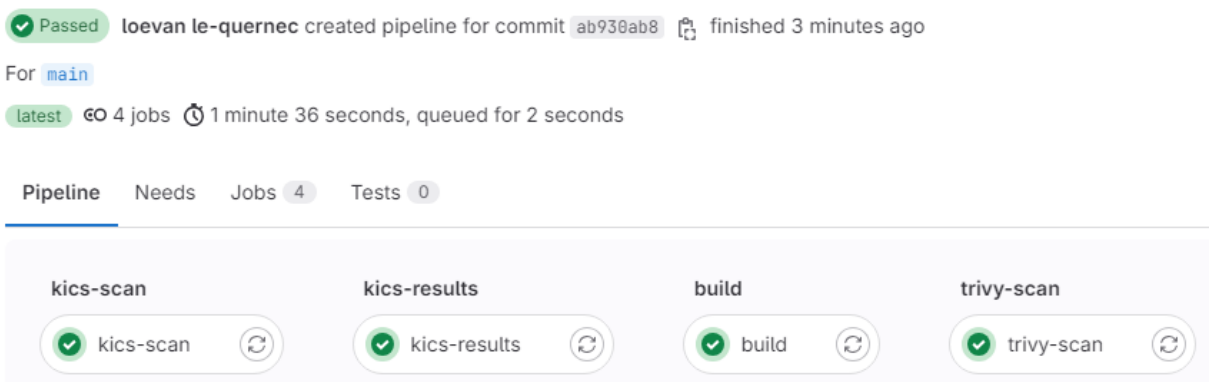
```
artifacts:
  paths:
    - $CI_PROJECT_DIR/image.tar
```

Bonus 3: Adding Trivy

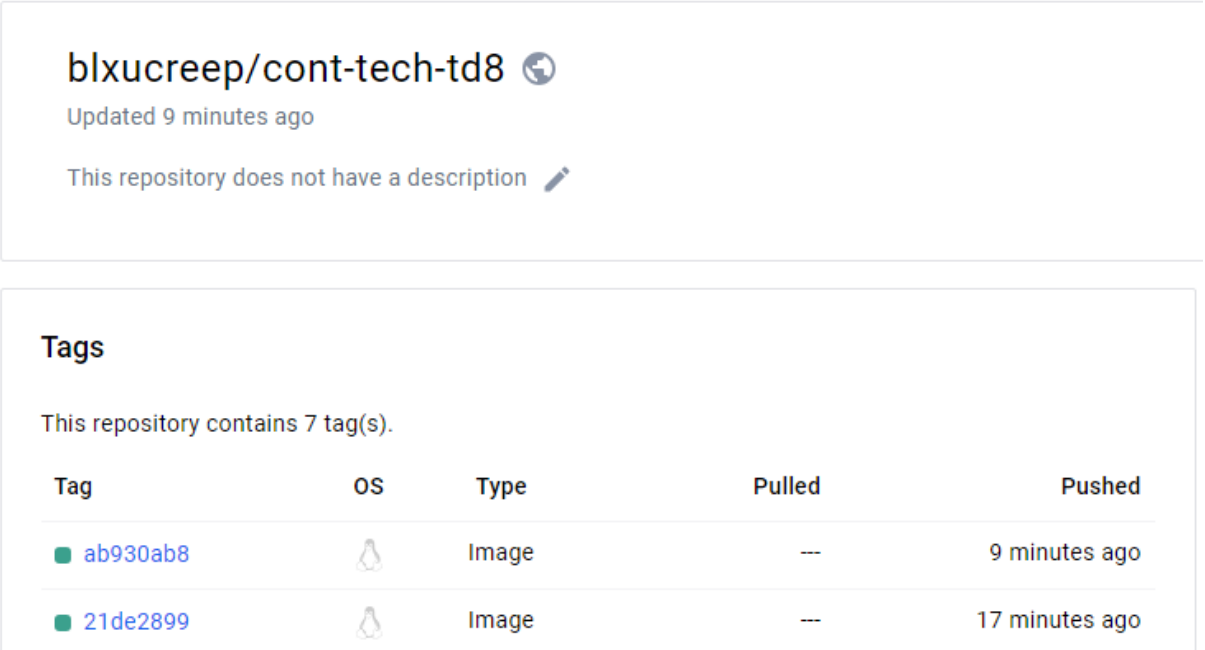
And now in the Trivy part, I added the trivy-scan stage:

```
trivy-scan:
  stage: trivy-scan
  image:
    name: aquasec/trivy:latest
    entrypoint: [""]
  script:
    - |
      echo "Scanning image..."
      trivy image --input $CI_PROJECT_DIR/image.tar
  dependencies:
    - build
  only:
    - branches
```

The pipeline in GitLab:



To see that everything is working fine at the end of the TD, we can see that our images are correctly pushed on Docker hub with the ‘CI_COMMIT_SHORT_SHA’:



And it’s actually ok!