

Containerization Technologies – TD 8

Step 0: GitLab

The connection is ok!

Step 1: Kics

Here is my Dockerfile:

```
Containerization Technologies > TD 8 > Dockerfile > ...
1  # use an official nginx image
2  FROM nginx:1.24
3
4  # copy the HTML file at /usr/share/nginx/html
5  COPY index.html /usr/share/nginx/html
6
7  # expose port 80
8  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/fd54f200-402c-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
1  stages:
2    - kics
3    - build
4
5  variables:
6    DOCKERFILE_PATH: Dockerfile
7
8  kics:
9    stage: kics
10   image:
11     name: checkmarx/kics:latest
12     entrypoint: [""]
13   script:
14     - |
15       echo "Scanning Dockerfiles..."
16       kics scan --path $DOCKERFILE_PATH
17   only:
18     - branches
19
20  build:
21    stage: build
22    image:
23     name: gcr.io/kaniko-project/executor:v1.14.0-debug
24     entrypoint: [""]
25    script:
26     - /kaniko/executor
27     --context "${CI_PROJECT_DIR}"
28     --dockerfile "${CI_PROJECT_DIR}/Dockerfile"
29     --destination "${CI_REGISTRY_IMAGE}:${CI_COMMIT_SHORT_SHA}"
30     - echo "{\"auths\":{\"${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:

```
18 Scanning Dockerfiles...
19 .0MO.
20 OMMX
21 ;NMX;
22 ...
23 WMMMd cWMMMO. KMMMO ;xKWMWMMMOc. ,xXMMWMMWkKc.
24 WMMMd .0MMMN: KMMMO :XMMWMMWMMWMMWL xMMWMMWMMWMMWL
25 WMMMd LWMMMO. KMMMO xMMWMMKc...'LXMK ,MMMX ;dXx
26 WMMMd.0MMNX; KMMMO cWMMMd 'MMWMMNL'
27 WMMWMMWMMWL KMMMO 0MMMN oMMWMMWMMkL.
28 WMMWMMWMMMO KMMMO 0MMNX .ckKWMWMMMO.
29 WMMWMMWokMMWk KMMMO oMMWMc .:0MMWMO
30 WMMWk. dWMMMO. KMMMO KMMWx' ,kNc :W0c. .NMWMMX
31 WMMMd cWMMMX. KMMMO kMMWMMWMMWMMMd .WMMWMMWk00MMWMMWL
32 WMMMd ,NMWMMN, KMMMO 'xNMWMMWMMWMMx, .l0WMMWMMWMMWk,
33 xkkk: ,kkkkx okkkL ;xKXKx; ;dOKKkc
34 Scanning with Keeping Infrastructure as Code Secure v1.7.13
35 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%
36 Healthcheck Instruction Missing, Severity: LOW, Results: 1
37 Description: Ensure that HEALTHCHECK is being used. The HEALTHCHECK instruction tells Docker how to test a container to check that it is still working
38 Platform: Dockerfile
39 CWE: 710
40 Learn more about this vulnerability: https://docs.kics.io/latest/queries/dockerfile-queries/b03a748a-542d-44f4-bb86-9199ab4fd2d5
41 [1]: Dockerfile:2
42 001: # use an official nginx image
43 002: FROM nginx:1.24
44 003:
45 Missing User Instruction, Severity: HIGH, Results: 1
46 Description: A user should be specified in the dockerfile, otherwise the image will run as root
47 Platform: Dockerfile
48 CWE: 250
49 Learn more about this vulnerability: https://docs.kics.io/latest/queries/dockerfile-queries/fd54f200-402c-4333-a5a4-36ef6709af2f
50 [1]: Dockerfile:2
51 001: # use an official nginx image
52 002: FROM nginx:1.24
53 003:
54 Results Summary:
55 HIGH: 1
56 MEDIUM: 0
57 LOW: 1
58 INFO: 0
59 TOTAL: 2
60 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 > ...
1  # use an official nginx image
2  FROM nginx:1.24
3
4  # copy the HTML file at /usr/share/nginx/html
5  COPY index.html /usr/share/nginx/html
6
7  # expose port 80
8  EXPOSE 80
9
10 # healthcheck instruction
11 HEALTHCHECK --interval=30s --timeout=30s --start-period=5s --retries=3 \
12     CMD curl -f http://localhost/ || exit 1
13
14 # specify an user
15 USER nginx
```

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

The pipeline in GitLab:

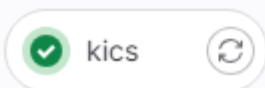
✓ Passed loevan le-quer nec created pipeline for commit 8fcfcc13

For main

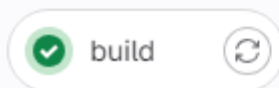
2 jobs 30 seconds, queued for 2 seconds

Pipeline Needs Jobs 2 Tests 0

kics



build



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).

```
7  variables:
8    DOCKERFILE_PATH: Dockerfile
9    KICS_RESULTS_PATH: kics-results
10
11  kics-scan:
12    stage: kics-scan
13    image:
14      name: checkmarx/kics:latest
15      entrypoint: [""]
16    script:
17      - |
18        echo "Scanning Dockerfiles..."
19        kics scan --path $DOCKERFILE_PATH --output-path $KICS_RESULTS_PATH
20    artifacts:
21      paths:
22        - $KICS_RESULTS_PATH/results.json
23    only:
24      - branches
25
26  kics-results:
27    stage: kics-results
28    image: alpine:latest
29    script:
30      - |
31        echo "Showing KICS scan results..."
32        cat $KICS_RESULTS_PATH/results.json
33    dependencies:
34      - kics-scan
35    only:
36      - branches
```

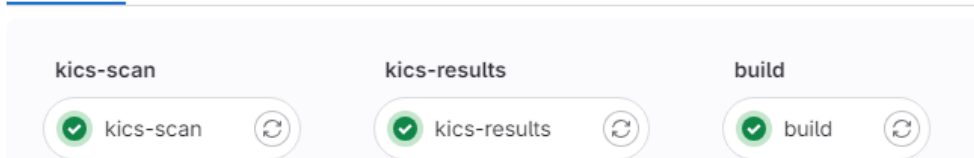
The pipeline in GitLab:

✓ Passed loevan le-quernec created pipeline for commit d9b6d090 finished 6 minutes ago

For main

latest 3 jobs 36 seconds, queued for 3 seconds

Pipeline Needs Jobs 3 Tests 0



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:

```
56  trivy-scan:
57    stage: trivy-scan
58    image:
59      name: aquasec/trivy:latest
60      entrypoint: [""]
61    script:
62      - |
63        echo "Scanning image..."
64        trivy image --input $CI_PROJECT_DIR/image.tar
65    dependencies:
66      - build
67    only:
68      - branches
```

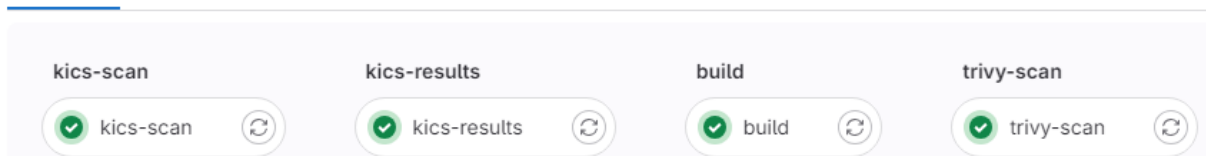
The pipeline in GitLab:

✓ Passed loevan le-quernec created pipeline for commit `ab938ab8` finished 3 minutes ago


For `main`

latest 4 jobs 1 minute 36 seconds, queued for 2 seconds


Pipeline Needs Jobs 4 Tests 0



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':





blxucreep/cont-tech-td8 

Updated 9 minutes ago

This repository does not have a description 

Tags

This repository contains 7 tag(s).

Tag	OS	Type	Pulled	Pushed
 ab930ab8		Image	---	9 minutes ago
 21de2899		Image	---	17 minutes ago

And it's actually ok!