



Univerza v Mariboru

Fakulteta za elektrotehniko,
računalništvo in informatiko

Projektna naloga 2

(ni imena?)

Študent:

Domen Hribnik

Denis Železnik

Merisa Mustajbašić

Datum in mesto:

21.05.2023, Maribor

Kazalo

| | |
|---|----------|
| 1. Docker Hub container registry..... | 1 |
| 1.1. Namestitev Docker Hub Registry | 1 |
| 2. GitHub Actions workflows..... | 4 |
| 2.1. GitHub Actions Workflow ideje za projekt ni_imena? | 5 |
| 3. Webhook | 6 |
| 3.1. Varnostne luknje pri uporabi Webhook | 8 |

1. Docker Hub container registry

1.1. Namestitev Docker Hub Registry

Najprej smo namestili uporabniški račun vodje skupine docker-ja na VM preko ukaza „docker login“.

```
ni-inena-admin@ni-inena:~$ docker login
WARNING: Your password will be stored unencrypted in /home/ni-inena-admin/.docker/config.json.
Configure a credential helper to remove this warning. See
https://docs.docker.com/engine/reference/commandline/login/#credentials-store

Login Succeeded
ni-inena-admin@ni-inena:~$ docker system info

Client:
Context:    default
Debug Mode: false
Plugins:
  app: Docker App (Docker Inc., v0.9.1-beta3)
  buildx: Build with Buildkit (Docker Inc., v0.6.1-docker)
  scan: Docker Scan (Docker Inc., v0.8.0)

Server:
Containers: 4
  Running: 0
  Paused: 0
  Stopped: 4
Images: 22
Server Version: 20.10.7
Storage Driver: overlay2
  Backing Filesystem: extfs
  Supports d_type: true
  Native Overlay Diff: false
userxattr: false
Logging Driver: json-file
Cgroup Driver: cgroupfs
Cgroup Version: 1
Plugins:
  Volume: local
  Network: bridge host ipvlan macvlan null overlay
  Log: awslogs fluentd gcplogs gelf journald json-file local logentries splunk syslog
Swarm: inactive
Runtimes: io.containerd.runc.v2 io.containerd.runtime.v1.linux runc
Default Runtime: runc
Init Binary: docker-init
  containerd version: 071fcd7d8303cbf684402823e425a9dd2e99286d
  runc version: d4c639434699f1b474674d1f856e4330db7
  init version: de44ad8
Security Options:
  apparmor
  seccomp
   Profile: default
Kernel Version: 4.15.0-1113-azure
Operating System: Ubuntu 16.04.7 LTS
OSType: linux
Architecture: x86_64
CPUs: 1
Total Memory: 921.7MiB
Name: ni-inena
ID: LAZC:H5VA:CM60:XSYS:YT63:SDV1:TARD:0038:2A07:1MZL:NG80:HV4L
Docker Root Dir: /var/lib/docker
Debug Mode: false
Username: mjorkk
Registry: https://index.docker.io/v1/
Labels:
Experimental: false
Insecure Registries:
  127.0.0.0/8
Live Restore Enabled: false

WARNING: No swap limit support
```

Nato ustvarimo repozitorij in nastavimo da je private.

Create repository

Namespace: mjorkk
Repository Name: ni_imena

Description: Private docker hub repository for the (ni_imena?) team

Visibility
Using 0 of 1 private repositories. [Get more](#)

☐ Public
Appears in Docker Hub search results

☒ Private
Only visible to you

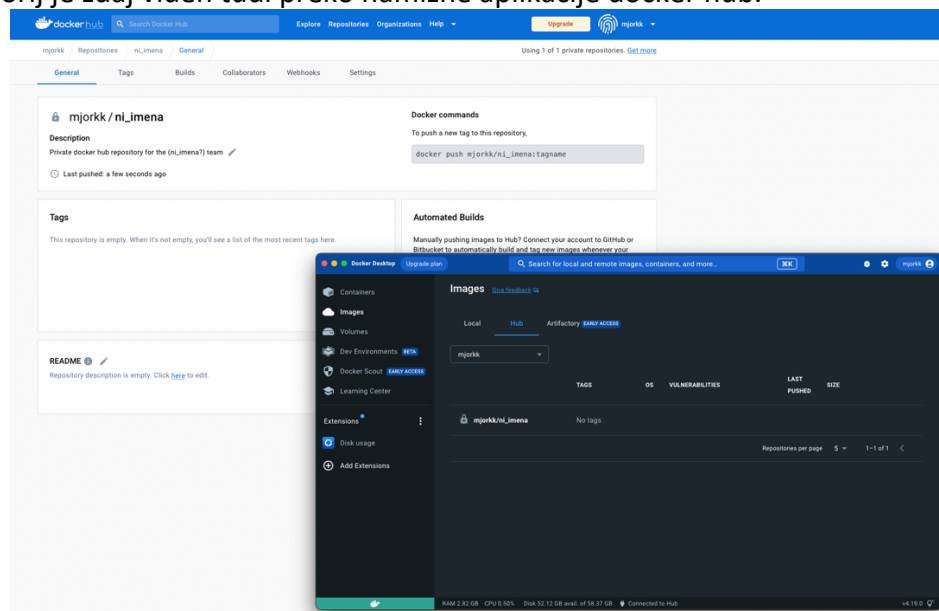
[Cancel](#) [Create](#)

Pro tip
You can push a new image to this repository using the CLI

```
docker tag local-image:tagname new-repo:tagname  
docker push new-repo:tagname
```

Make sure to change tagname with your desired image repository tag.

Repozitorij je zdaj viden tudi preko namizne aplikacije docker hub.



Docker sliko nasega projekta push-amo na docker hub s pomočjo ukazov : docker image tag in docker image push.

```
merisa — ni-imen-admin@ni-imen: ~/SPLETNO_PROGRAMIRANJE — ssh ni-imen-admin@20.126.63.144 — 141x66
-- ni-imen-admin@ni-imen: ~/SPLETNO_PROGRAMIRANJE — ssh ni-imen-admin@20.126.63.144 ... ..ni-imen-admin@ni-imen: ~/SPLETNO_PROGRAMIRANJE — ssh ni-imen-admin@20.126.63.144
Last login: Fri May 12 09:31:07 on ttys000
merisa@merisa [B9:46:14] [-]
-> % ssh ni-imen-admin@20.126.63.144
ni-imen-admin@20.126.63.144's password:
Welcome to Ubuntu 16.04.7 LTS (GNU/Linux 4.15.0-1113-azure x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:    https://landscape.canonical.com
 * Support:       https://ubuntu.com/advantage

Expanded Security Maintenance for Infrastructure is not enabled.

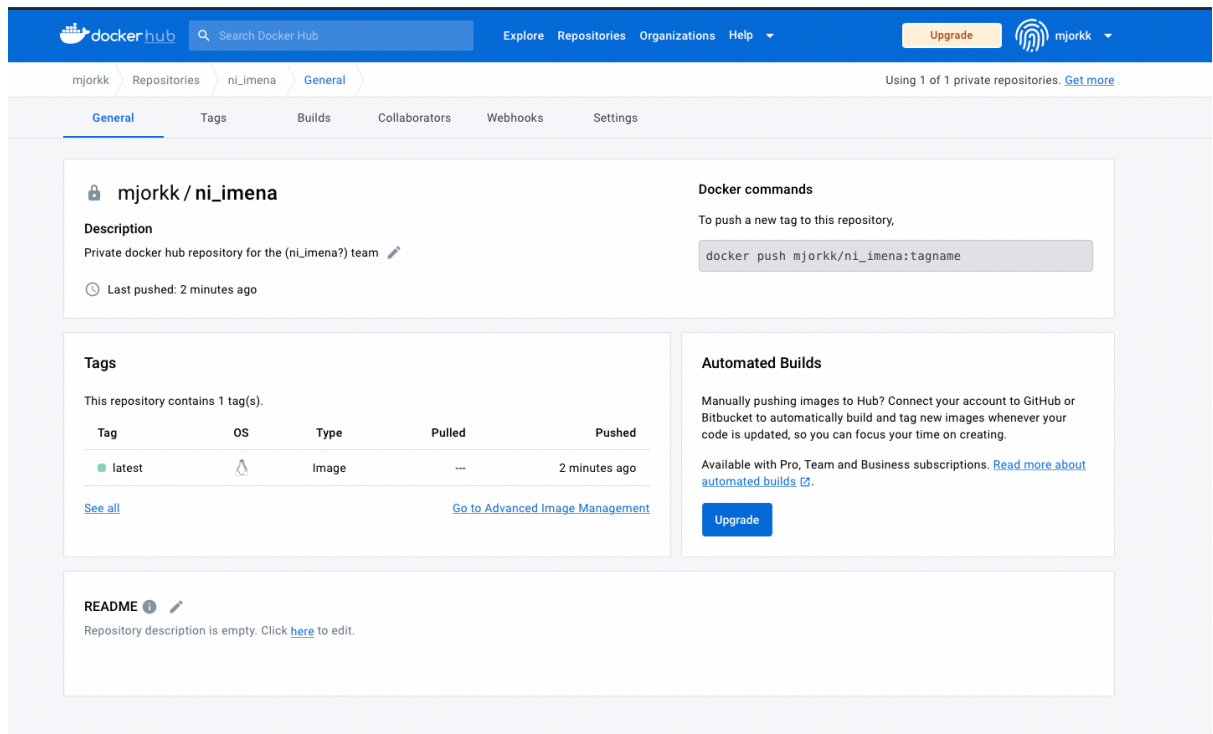
0 updates can be applied immediately.

211 additional security updates can be applied with ESM Infra.
Learn more about enabling ESM Infra service for Ubuntu 16.04 at
https://ubuntu.com/16-04

New release '18.04.6 LTS' available.
Run 'do-release-upgrade' to upgrade to it.

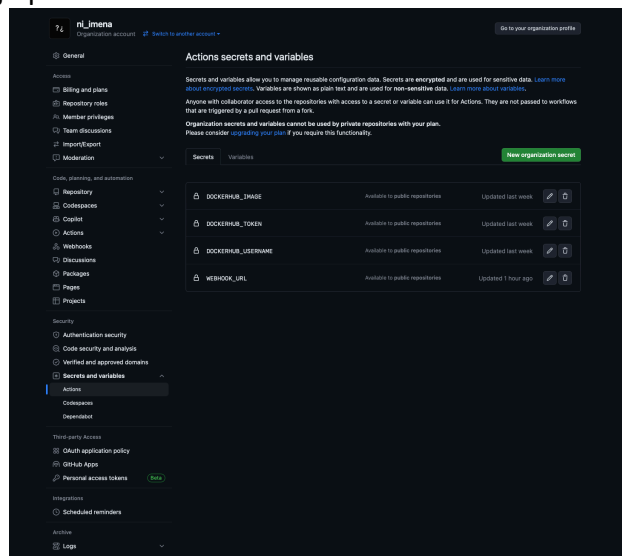
Last login: Fri May 12 07:32:04 2023 from 89.143.16.130
ni-imen-admin@ni-imen:~/$ cd SPLETNO_PROGRAMIRANJE/
ni-imen-admin@ni-imen:~/SPLETNO_PROGRAMIRANJE$ docker ps
CONTAINER ID   IMAGE      COMMAND                  CREATED        STATUS        PORTS        NAMES
f3c2c2b486ac   pn1       "docker-entrypoint.s..." 56 seconds ago Up 54 seconds 3000/tcp     goofy_morse
ni-imen-admin@ni-imen:~/SPLETNO_PROGRAMIRANJE$ docker stop pn1
Error response from daemon: No such container: pn1
ni-imen-admin@ni-imen:~/SPLETNO_PROGRAMIRANJE$ docker stop f3c2c2b486ac
f3c2c2b486ac
ni-imen-admin@ni-imen:~/SPLETNO_PROGRAMIRANJE$ docker ps
CONTAINER ID   IMAGE      COMMAND                  CREATED        STATUS        PORTS        NAMES
ni-imen-admin@ni-imen:~/SPLETNO_PROGRAMIRANJE$ hstr
docker run -dp 3000:3000 pn1
ni-imen-admin@ni-imen:~/SPLETNO_PROGRAMIRANJE$ docker run -dp 3000:3000 pn1
40e6ec97add251ed0c6fec8d58842cbb38fb2d5a7f46de2c0a4f90dcf470a430c
ni-imen-admin@ni-imen:~/SPLETNO_PROGRAMIRANJE$ docker images
REPOSITORY    TAG       IMAGE ID       CREATED        SIZE
pn1            latest   6853b7677038   3 days ago    1.07GB
ni_imen       latest   8d03cf96cc7    7 days ago    1.04GB
pn1           latest   24926b0da6dd   10 days ago   1.11GB
node          18       c9e4d38ad304   4 weeks ago   997MB
ni-imen-admin@ni-imen:~/SPLETNO_PROGRAMIRANJE$ docker image tag pn1 mjorkk/pn1:latest
ni-imen-admin@ni-imen:~/SPLETNO_PROGRAMIRANJE$ docker image push mjorkk/pn1:latest
The push refers to repository [docker.io/mjorkk/pn1]
c1f68d33f8d0: Pushed
dce3b0a4efc9: Pushed
d6d3ec7631f5: Pushed
4dfe1235fd7e: Pushed
128128ebb44b: Mounted from library/node
ca745f79e3b0: Mounted from library/node
77838e897d76: Mounted from library/node
e29838afe536: Mounted from library/node
cfd0811d364e: Mounted from library/node
b86f260e173a: Mounted from library/node
6a1ebb98b0dc: Mounted from library/node
24b48387f467: Mounted from library/node
ae56c0c5405b: Mounted from library/node
latest: digest: sha256:c77729765433d691c3cd656b80abdb576feb5621b43d540d9cc743b407ca4921 size: 3054
ni-imen-admin@ni-imen:~/SPLETNO_PROGRAMIRANJE$
```

```
ni-imen-admin@ni-imen:~/SPLETNO_PROGRAMIRANJE$ docker tag pn1 mjorkk/ni_imen:latest
ni-imen-admin@ni-imen:~/SPLETNO_PROGRAMIRANJE$ docker push mjorkk/ni_imen:latest
The push refers to repository [docker.io/mjorkk/ni_imen]
c1f68d33f8d0: Pushed
dce3b0a4efc9: Pushed
d6d3ec7631f5: Pushed
4dfe1235fd7e: Pushed
128128ebb44b: Pushed
ca745f79e3b0: Pushed
77838e897d76: Pushed
e29838afe536: Pushed
cfd0811d364e: Pushed
b86f260e173a: Pushed
6a1ebb98b0dc: Pushed
24b48387f467: Pushed
ae56c0c5405b: Pushed
latest: digest: sha256:c77729765433d691c3cd656b80abdb576feb5621b43d540d9cc743b407ca4921 size: 3054
ni-imen-admin@ni-imen:~/SPLETNO_PROGRAMIRANJE$
```



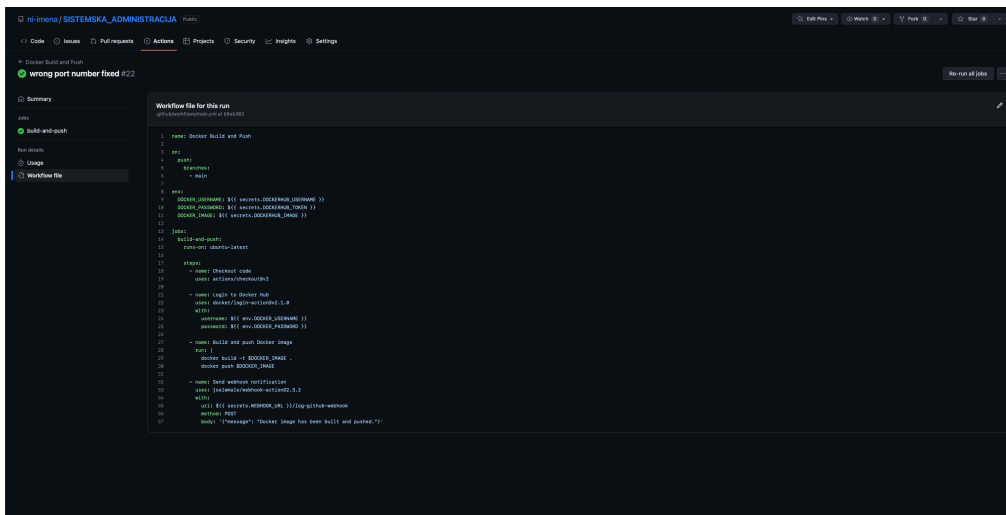
2. GitHub Actions workflows

Najprej nastavimo vdje potrebne GitHub secrets:

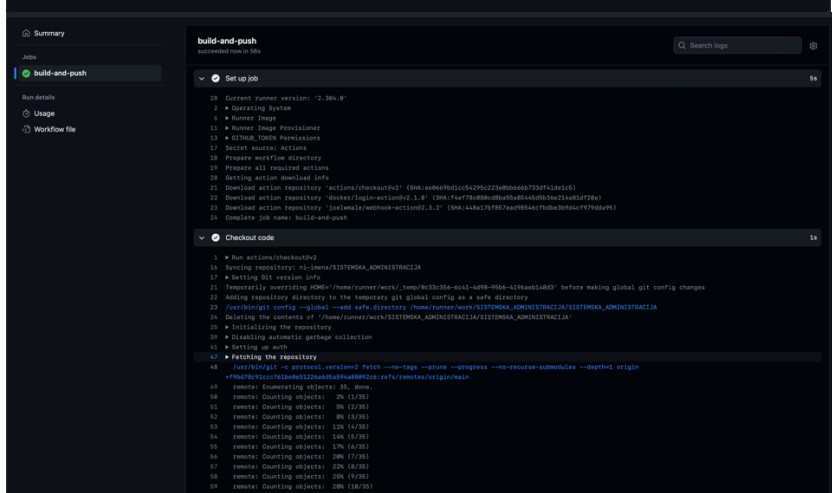
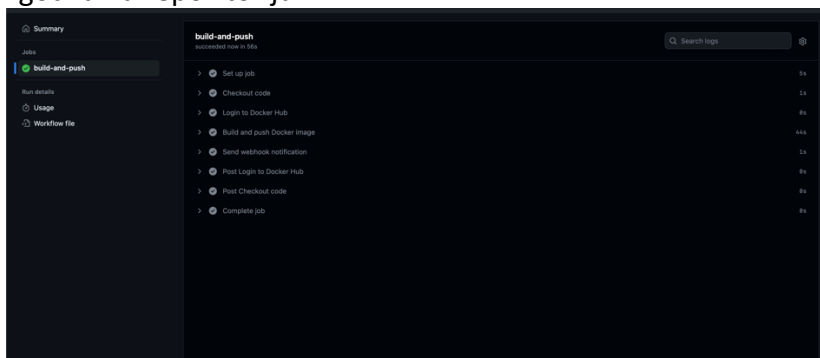


V zavihu za Workflows repozitorija napisemo skripto za podane naloge:

- ustvarjanje **Docker** slike in njeno nalaganje na **Docker Hub**,
- pošiljanje sporočila vašemu strežniku s pomočjo **webhooks**



Zdaj z vsakim pushom na repozitorij, se workflow zazene in naredi prejsnje omenjene naloge. To je tudi razvidno na docker hub, s katerim vidimo kdaj se je zadnja sprememba zgodila na repozitoriju.



2.1. GitHub Actions Workflow ideje za projekt ni_imena?

Za projekt "Digitalni dvojček tekača" bi lahko dodali naslednje dodatne GitHub Actions workflows:

1. **Continuous Integration (CI) Workflow:** Ta workflow bi bil odgovoren za avtomatizirano preverjanje kode in zagotavljanje, da so vse spremembe v kodi združljive in delujejo pravilno. Workflow bi se sprožil ob vsakem potisku v glavno vejo (npr. **main**) ali pri pošiljanju sprememb v odvisnih vejah. Vključeval bi naslednje korake:
 - Preverjanje kode s statično analizo, stilskimi smernicami in preverjanjem kakovosti kode.
 - Izvajanje enotnih testov, da se zagotovi delovanje osnovnih funkcionalnosti.
 - Gradnja in pakiranje aplikacije v ustrezno obliko (npr. Docker sliko).
2. **Deployment Workflow:** Ta workflow bi bil odgovoren za avtomatizirano distribucijo aplikacije na izbrani ciljni okolje po uspešnem preverjanju kode. Workflow bi se sprožil po združitvi sprememb v glavno vejo (npr. **main**). Vključeval bi naslednje korake:
 - Preverjanje kode (kot v CI workflow).
 - Gradnja in pakiranje aplikacije v ustrezno obliko (npr. Docker sliko).
 - Distribucija aplikacije na ciljni strežnik ali oblak (npr. AWS, Azure, GCP).
 - Zagon aplikacije na ciljnem okolju in preverjanje, ali je pravilno delujoča.
3. **Scheduled Workflow:** Ta workflow bi bil nastavljen za periodično izvajanje določenih nalog, na primer za izvajanje rednih nalog vzdrževanja, arhiviranja podatkov ali preverjanja delovanja sistema. Workflow bi se sprožil vnaprej določen časovni interval ali na določen dan v tednu/mesecu. Naloge, ki bi se izvajale v tem workflowu, bi bile specifične za potrebe projekta.
4. **Pull Request Workflow:** Ta workflow bi se sprožil ob odprtju ali posodobitvi potegnjene zahteve (pull requesta) in bi pomagal pri preverjanju kode in zagotavljanju kakovosti pred združitvijo sprememb v glavno vejo. Vključeval bi podobne korake kot CI workflow, vendar bi se izvajal samo na spremembah, vključenih v potegnjeni zahtevek.
5. **Issue/Feature Workflow:** Ta workflow bi bil namenjen za upravljanje s prijavami napak (issues) ali predlogi funkcionalnosti (features). Sprožil bi se ob odprtju ali posodobitvi novega zapisa in bi lahko vključeval korake, kot so preverjanje in validacija zapisa, dodelitev nalog

3. Webhook

V kodu servera dodamo del za sprejemanje webhooka.


```

async function manageContainer() {
  try {
    // Stop the currently running container (if it exists)
    await exec('docker stop pn3').catch(() => {});

    // Pull the latest image from Docker Hub
    await exec('docker pull mjorkk/ni_imena:latest');

    // Start the container
    await exec('docker run -d --name pn3 mjorkk/ni_imena:latest');

    // Send webhook notification
    const webhookURL = 'https://hkdk.events/yzTux3lS7QVe'; // Replace with your actual webhook URL
    await axios.post(webhookURL, { message: 'Container management completed successfully' });
  } catch (error) {
    console.error('Error managing container:', error);
  }
}

app.use(function (req, res, next) {
  next(createError(404));
});

app.post('/log-github-webhook', (req, res) => {
  // Handle the webhook request
  // ...

  // Call the manageContainer function
  manageContainer();
  console.log("sjhdfgjskdf");

  res.status(200).send('Webhook received successfully');
});

```

- Napišite preprosto skripto v poljubnem programskem jeziku, ki bo:
 - zaustavila trenutno zagnan docker container,
 - iz Docker Container registry prenesla najnovejšo različico vašega containerja,
 - ga ponovno zagnala.
- Skripto zaženite ob vsakem dobljenem Webhook sporočilu.
- Opišite varnostne luknje in vaše pomisleke pri uporabi Webhook, opišite (za

Na zagovoru vaj boste prikazali pravično delovanje celotnega GitHub Actions workflowa.

Na sistem oddajte poročilo v .pdf obliki. Poročilo naj vsebuje opise posameznih korakov, ki jih boste izvedli, da boste zagotovili, da je sistem deluje pravilno. Na zagovoru bomo pogledali oporočilo in ga ocenili.

KRITERIJI OCENJEVANJA IMPLEMENTACIJE:

1. točka - Docker Hub container registry
2. točka - delujoč osnovni GitHub Actions workflow
3. točka - opis namestitve dodatnega GitHub Actions workflows
4. točka - Webhook
5. točka - varnost v Webhook

KRITERIJI OCENJEVANJA VODENJA PROJEKTA:

- 1 točka - oddano in zagovorjeno poročilo
- 2 točki - avtomatično vodenje projekta na izbranem sistemu (Jiro, OpenProj, iz

Naloge brez oddanih poročil, ali s pomanjkljivimi poročili, bodo ocenjene z 0 točkami.

Instaliramo in zazenemo na portu 1337 hookdeck:

```

ni-imena-admin@ni-imena:~/SISTEMSKA_ADMINISTRACIJA$ sudo hookdeck listen 1337
{"message":"Docker image has been built and pushed."}
? Select a source Create new source
? What should be your new source label? pn3
? What path should the webhooks be forwarded to (ie: /webhooks)? /log-github-webhook
? What's your connection label (ie: My API)? response server

Dashboard
🔍 Inspect and replay webhooks: http://console.hookdeck.com?source_id=src_yzTux3lS7QVe

pn3 Source
🔗 Webhook URL: https://hkdk.events/yzTux3lS7QVe

Connections
response-server forwarding to /log-github-webhook

> Ready! (^C to quit)
2023-05-21 20:36:51 [ERROR] Failed to POST: Post "http://localhost:1337/log-github-webhook/log-github-webhook": dial tcp 127.0.0.1:1337: connect: connection refused
2023-05-21 20:37:09 [ERROR] Failed to POST: Post "http://localhost:1337/log-github-webhook/log-github-webhook": dial tcp 127.0.0.1:1337: connect: connection refused

```

Workflow file for this run

.github/workflows/main.yml at d3b8557

```
1 name: Docker Build and Push
2
3 on:
4   push:
5     branches:
6       - main
7
8 env:
9   DOCKER_USERNAME: ${ secrets.DOCKERHUB_USERNAME }
10  DOCKER_PASSWORD: ${ secrets.DOCKERHUB_TOKEN }
11  DOCKER_IMAGE: ${ secrets.DOCKERHUB_IMAGE }
12
13 jobs:
14   build-and-push:
15     runs-on: ubuntu-latest
16
17     steps:
18     - name: Checkout code
19       uses: actions/checkout@v2
20
21     - name: Login to Docker Hub
22       uses: docker/login-action@v2.1.0
23       with:
24         username: ${ env.DOCKER_USERNAME }
25         password: ${ env.DOCKER_PASSWORD }
26
27     - name: Build and push Docker image
28       run: |
29         docker build -t $DOCKER_IMAGE .
30         docker push $DOCKER_IMAGE
31
32     - name: Send webhook notification
33       uses: joelwmaile/webhook-action@2.3.2
34       with:
35         url: http://20.126.63.144:1337/log-github-webhook
36         method: POST
37         body: '{"message": "Docker image has been built and pushed."}'
```

The screenshot shows a web browser window with a console log and a request/response inspector. The console log shows a message 'Docker image has been built and pushed.' at 9:01 PM. The request/response inspector shows a POST request to http://20.126.63.144:1337/log-github-webhook with a JSON body containing the message.

| Status | Time | Body |
|--------|--------------------|--|
| ✓ | May 21 at 9:01 PM | ["message": "Docker image has been built and pushed."] |
| ✓ | May 21 at 8:56 PM | --- |
| ✓ | May 19 at 8:21 PM | ["ref": "refs/heads/main", "before": "64aadc3791cd576ff7866d281c473116f93a", "after": "798f137c2609ef48638c3938b276769c73aaa59", "repository": "https://github.com/JoelWmaile/webhook-action.git", "sender": "joelwmaile", "event": "push"}] |
| ✓ | May 18 at 11:26 PM | ["ref": "refs/heads/main", "before": "64aadc3791cd576ff7866d281c473116f93a", "after": "798f137c2609ef48638c3938b276769c73aaa59", "repository": "https://github.com/JoelWmaile/webhook-action.git", "sender": "joelwmaile", "event": "push"}] |

Request: POST http://20.126.63.144:1337/log-github-webhook

Response: 200 OK

Body: {"message": "Docker image has been built and pushed."}

3.1. Varnostne luknje pri uporabi Webhook

Pri uporabi Webhook-ov obstaja nekaj varnostnih lukenj in pomislekov, ki jih je treba upoštevati. Nekatere od teh lukenj so:

1. **Nepristnost (Spoofing):** Napadalec lahko poskuša poslati lažne Webhook zahteve, pretvarjajoč se, da prihajajo od zaupanja vrednega vira. To bi lahko privedlo do izvedbe neželenih dejanj ali vnosa škodljive kode v vaš sistem.
2. **Ponavljjanje (Replay Attack):** Napadalec lahko prestreže veljavne Webhook zahteve in jih nato ponovi kasneje, da povzroči podvojene ali neželene operacije.
3. **Vdiranje (Injection):** Napadalec lahko poskuša vstaviti zlonamerno vsebino v Webhook zahtevo, kar bi lahko povzročilo izvajanje škodljive kode ali zlorabo sistema.
4. **Nepopolno preverjanje podatkov (Incomplete Data Validation):** Če ne izvajate zadostne preverbe podatkov v prejetih Webhook zahtevah, obstaja tveganje za vnos neveljavnih ali zlonamernih podatkov v vaš sistem.
5. **Dostopna pravica (Access Control):** Neustrezno upravljanje dostopnih pravic do Webhook-ov lahko pomeni, da nepooblaščene osebe pridobijo dostop do občutljivih podatkov ali funkcij.

Za odpravo teh varnostnih lukenj in pomislov pri uporabi Webhook-ov lahko upoštevamo naslednje prakse:

1. **Overjanje (Authentication):** Zagotovimo, da preverite pristnost prejetih Webhook zahtev, na primer z uporabo avtentikacije z žetonom (token-based authentication). Vsak vir Webhook-a naj ima svoj edinstven žeton, ki ga uporabljamo za preverjanje pristnosti.
2. **Šifriranje (Encryption):** Če prenašamo občutljive podatke prek Webhook-ov, jih šifriramo, da se prepreči prestrežanje in nepooblaščno branje.
3. **Preverjanje integritete (Integrity Checking):** Za preprečevanje ponavljanja napadov uporabite mehanizme preverjanja integritete, na primer s pomočjo časovnih oznak ali digitalnih podpisov. S tem zagotovimo, da so prejete Webhook zahteve unikatne in niso bile spremenjene.
4. **Preverjanje vhodnih podatkov (Input Data Validation):** Temeljito preverimo vhodne podatke, ki jih prejmemo prek Webhook-ov. Preverimo, ali so podatki v skladu z določenimi pravili in omejitvami.
5. **Upravljanje dostopa (Access Management):** Omogočimo ustrezno upravljanje dostopnih pravic do Webhook-ov.
6. **Dnevniško beleženje (Logging):** Zagotovimo ustrezno dnevniško beleženje dogodkov, povezanih z Webhook-i. Zabeležimo prejete zahteve, njihove rezultate in morebitne napake. S tem boste lahko sledili dogodkom, analizirali morebitne varnostne incidente in izvedli potrebne ukrepe.