Univerza v Mariboru

Fakulteta za elektrotehniko,
računalništvo in informatiko

Projektna naloga 2

(ni imena?)

Študent:
Domen Hribernik
Denis Železnik
Merisa Mustajbašić

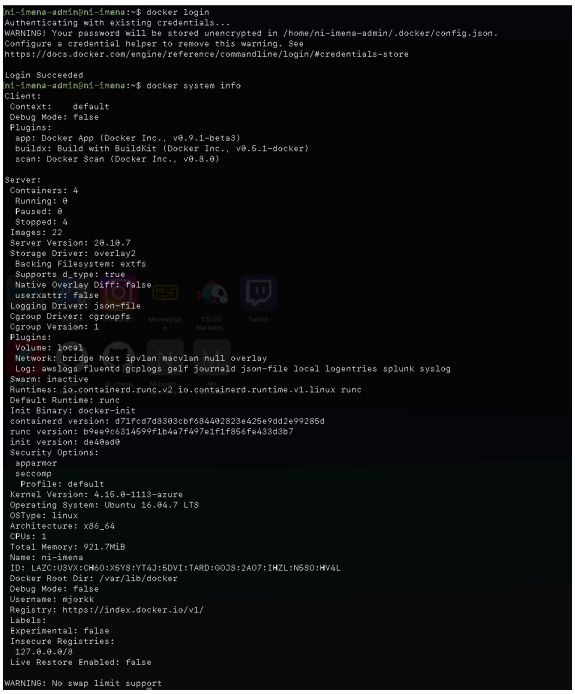
Datum in mesto:
21.05.2023, Maribor

Kazalo

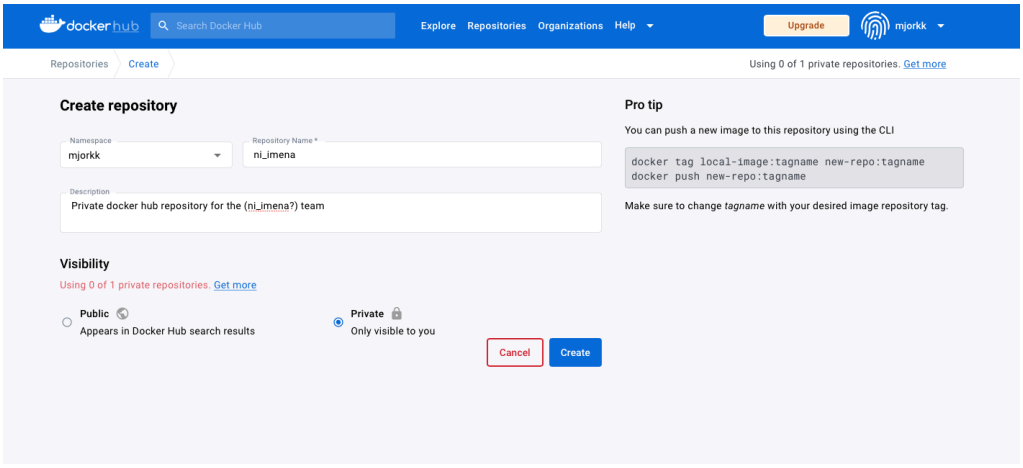
1. Docker Hub container registry

1.1. Namestitev Docker Hub Registry

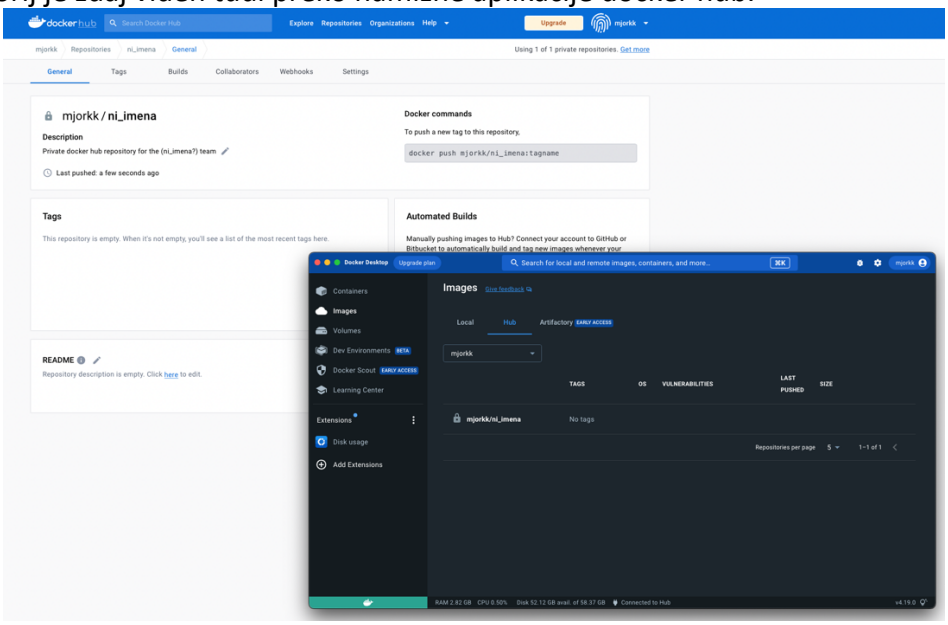
Najprej smo namestili uporabniski racun vodje skupine docker-ja na VM preko ukaza „docker login“.



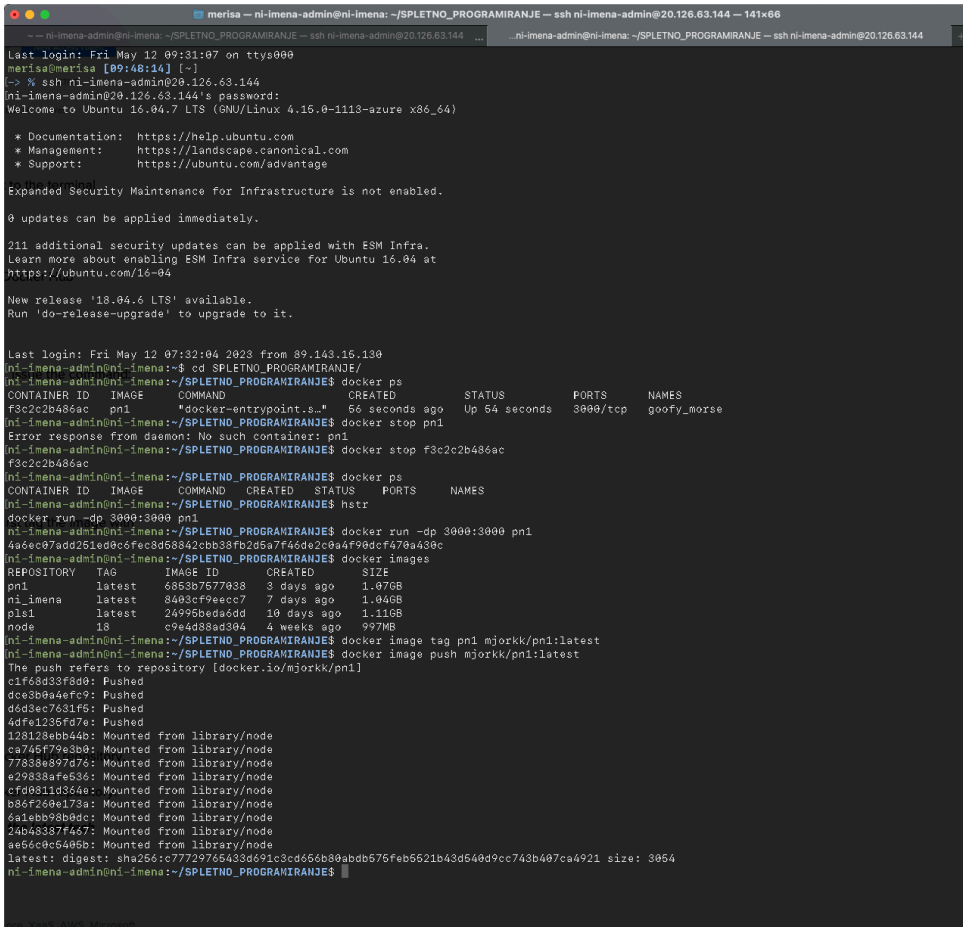
Nato ustvarimo repozitorij in nastavimo da je private.

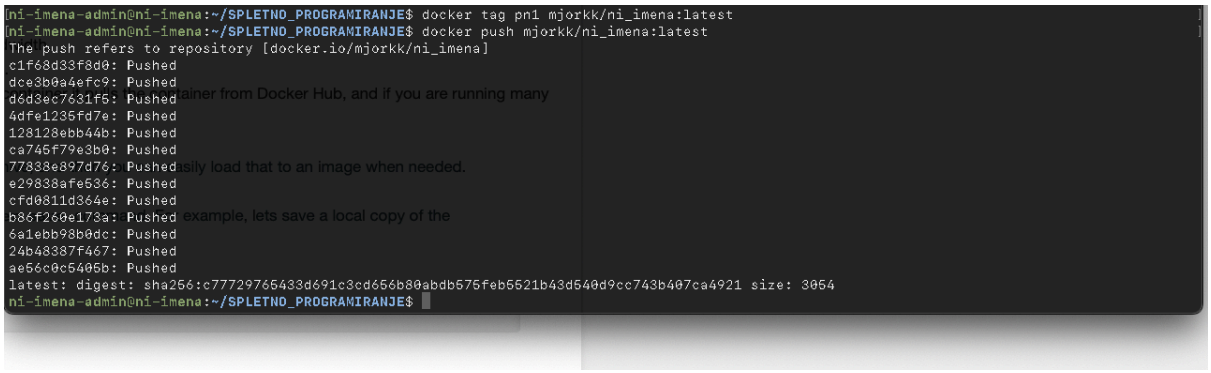


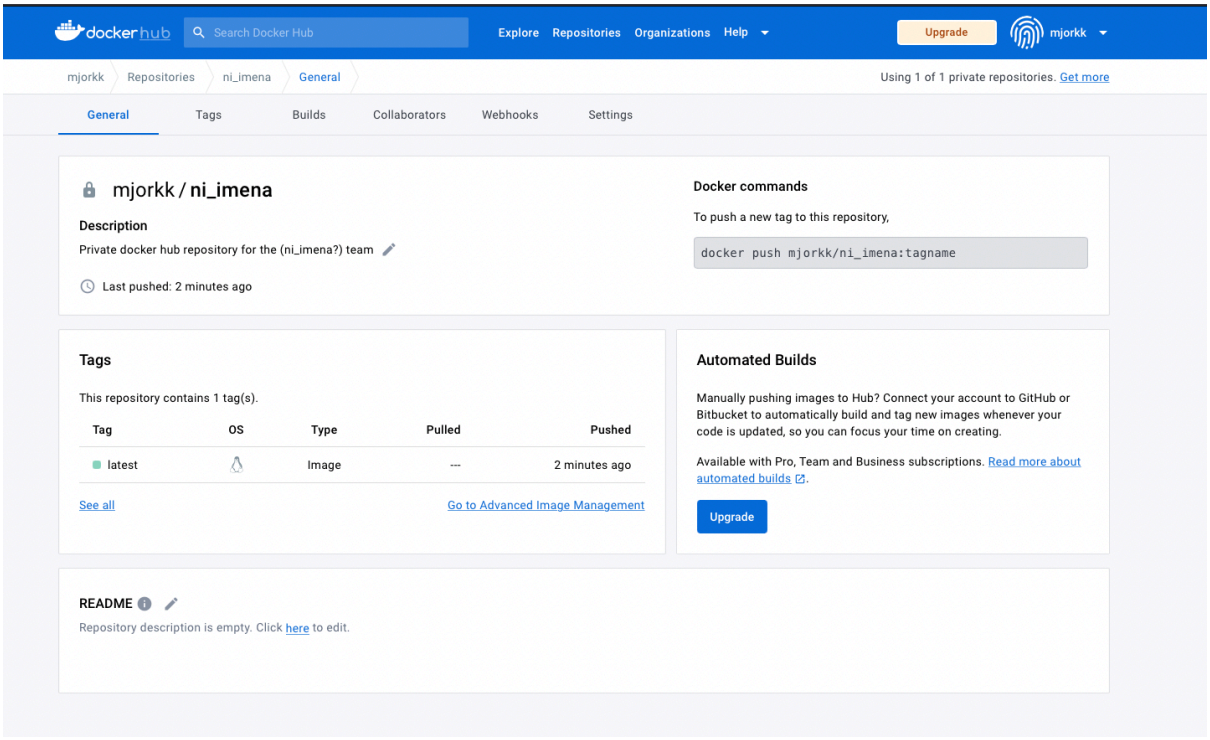
Repozitorij je zdaj viden tudi preko namizne aplikacije docker hub.



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

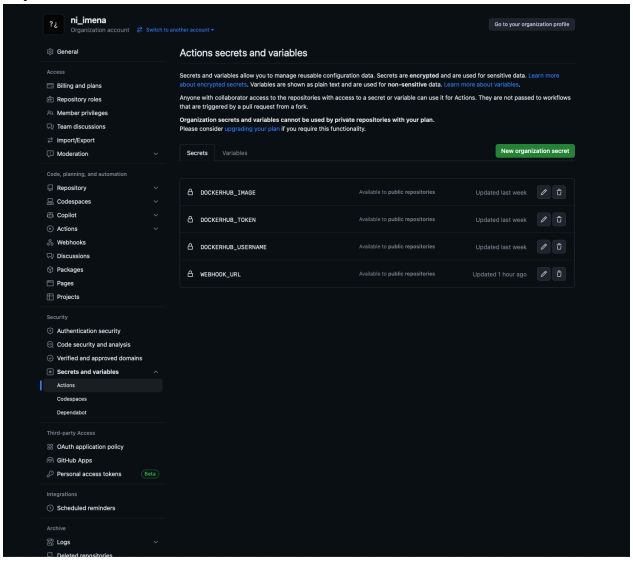

merisa — ni-imena-admin@ni-imena: ~/SPLETNO_PROGRAMIRANJE — ssh ni-imena-admin@20.126.63.144 — 141×66
Last login: Fri May 12 09:31:07 on ttys000
merisa@merisa [09:48:14] [~]
-> % ssh ni-imena-admin@20.126.63.144
ni-imena-admin@20.126.63.144's password:
Welcome to Ubuntu 16.04.7 LTS (GNU/Linux 4.15.0-1113-azure 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-imena-admin@ni-imena:~$ cd SPLETNO_PROGRAMIRANJE/
ni-imena-admin@ni-imena:~/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-imena-admin@ni-imena:~/SPLETNO_PROGRAMIRANJE$ docker stop pn1
Error response from daemon: No such container: pn1
ni-imena-admin@ni-imena:~/SPLETNO_PROGRAMIRANJE$ docker stop f3c2c2b486ac
f3c2c2b486ac
ni-imena-admin@ni-imena:~/SPLETNO_PROGRAMIRANJE$ docker ps
CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES
ni-imena-admin@ni-imena:~/SPLETNO_PROGRAMIRANJE$ hstr
docker run -dp 3000:3000 pn1
ni-imena-admin@ni-imena:~/SPLETNO_PROGRAMIRANJE$ docker run -dp 3000:3000 pn1
4a6ec07add251ed0c6fec8d58842cbb38fb2d5a7f46de2c0a4f90dcf470a430c
ni-imena-admin@ni-imena:~/SPLETNO_PROGRAMIRANJE$ docker images
REPOSITORY TAG IMAGE ID CREATED SIZE
pn1 latest 6853b7577038 3 days ago 1.07GB
ni_imena latest 8403cf9eecc7 7 days ago 1.04GB
pls1 latest 24995beda6dd 10 days ago 1.11GB
node 18 c9e4d88ad304 4 weeks ago 997MB
ni-imena-admin@ni-imena:~/SPLETNO_PROGRAMIRANJE$ docker image tag pn1 mjorkk/pn1:latest
ni-imena-admin@ni-imena:~/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:c77729765433d691c3cd656b80abdb575feb5521b43d540d9cc743b407ca4921 size: 3054
ni-imena-admin@ni-imena:~/SPLETNO_PROGRAMIRANJE$


ni-imena-admin@ni-imena:~/SPLETNO_PROGRAMIRANJE$ docker tag pn1 mjorkk/ni_imena:latest
ni-imena-admin@ni-imena:~/SPLETNO_PROGRAMIRANJE$ docker push mjorkk/ni_imena:latest
The push refers to repository [docker.io/mjorkk/ni_imena]
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:c77729765433d691c3cd656b80abdb575feb5521b43d540d9cc743b407ca4921 size: 3054
ni-imena-admin@ni-imena:~/SPLETNO_PROGRAMIRANJE$



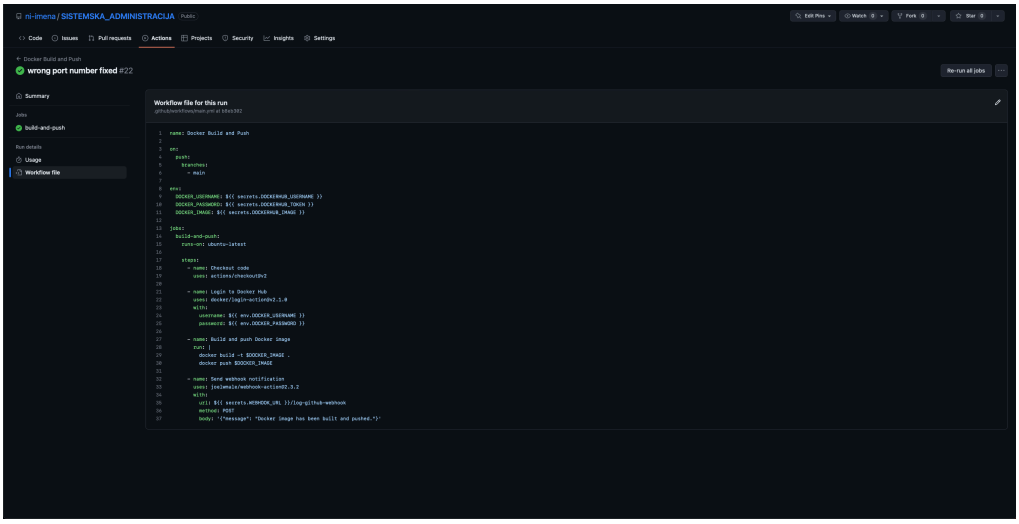
2. GitHub Actions workflows

Najprej nastavimo vdje potrebne GitHub secrets:

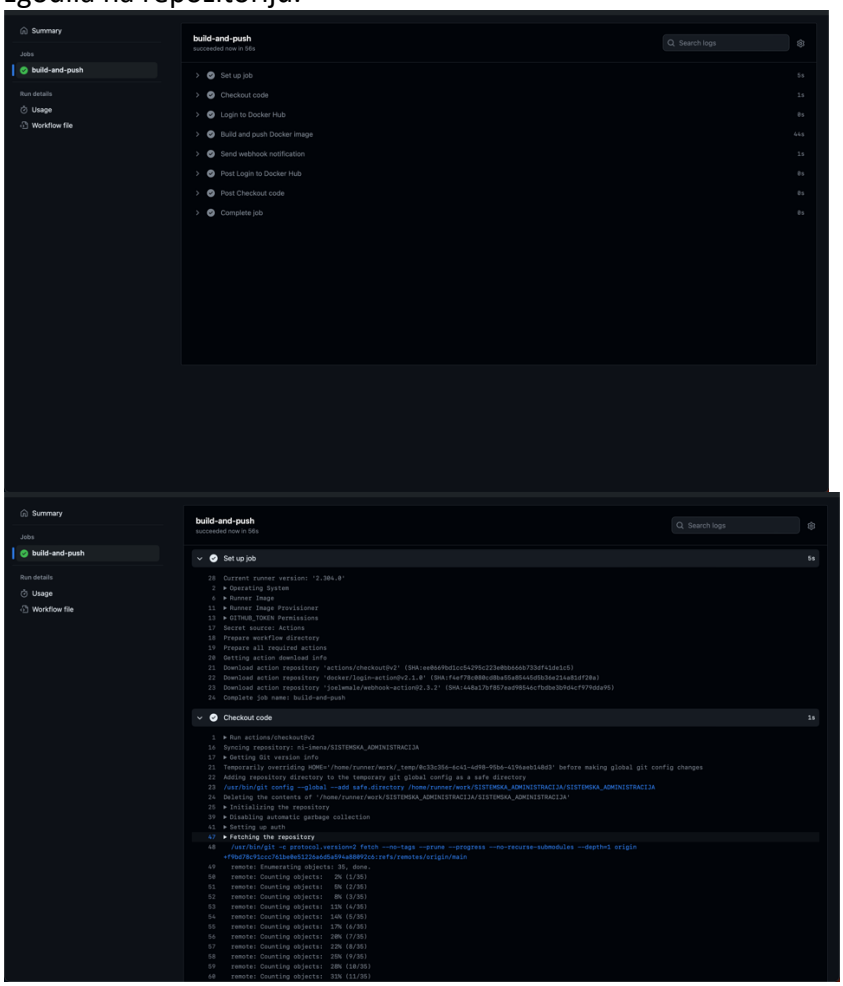


V zavihu za Workflows repozitorija napisemo skripto za podane naloge:

- 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.



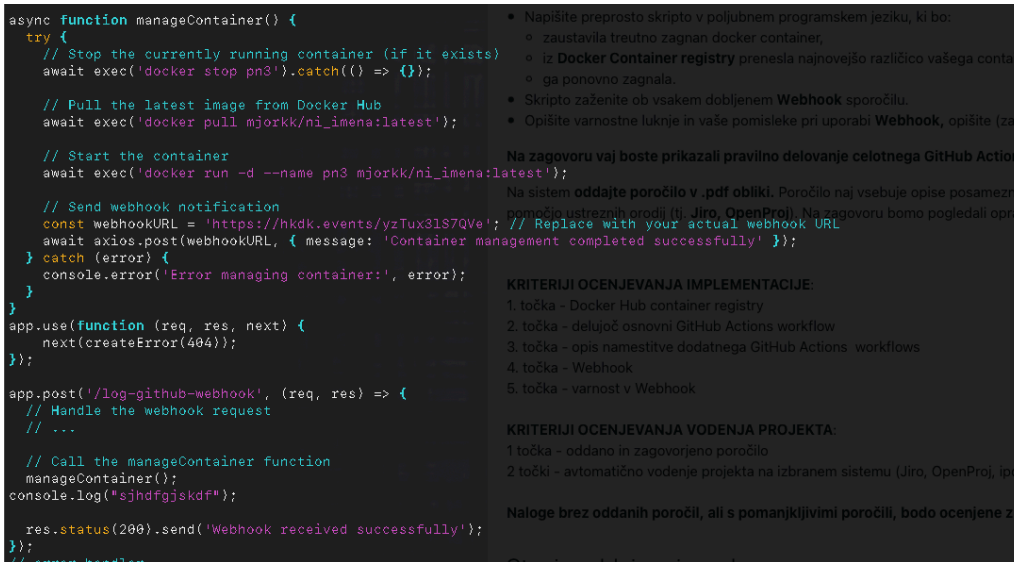
2.1. GitHub Actions Workflow ideje za projekt ni_imena?

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 potegnjenega zahtevka (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 treutno zagnan docker container,
  - iz **Docker Container registry** prenesla najnovejšo različico vašega conta
  - 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 pravilno delovanje celotnega GitHub Actio

Na sistem **oddajte poročilo v .pdf obliki.** Poročilo naj vsebuje opise posamezn
pomočjo ustreznih orodij (ti. **Jiro, OpenProj**). Na zagovoru bomo pogledali opr

KRITERIJI OCENJEVANJA IMPLEMENTACIJE:

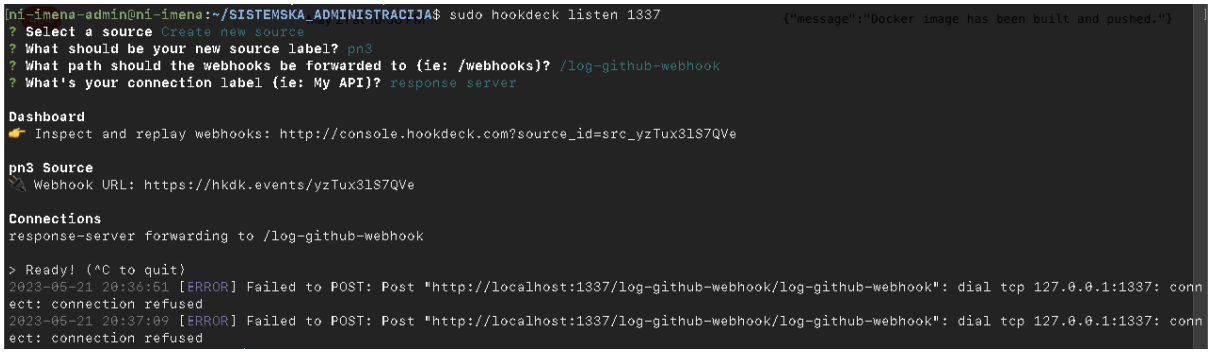
1. točka - Docker Hub container registry
2. točka - delujoč osnovni GitHub Actions workflow
3. točka - opis namestitve dodatnega GitHub Actions workflows
4. točka - Webhook
5. točka - varnost v Webhook

KRITERIJI OCENJEVANJA VODENJA PROJEKTA:

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

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

Instaliramo in zazenemo na portu 1337 hookdeck:

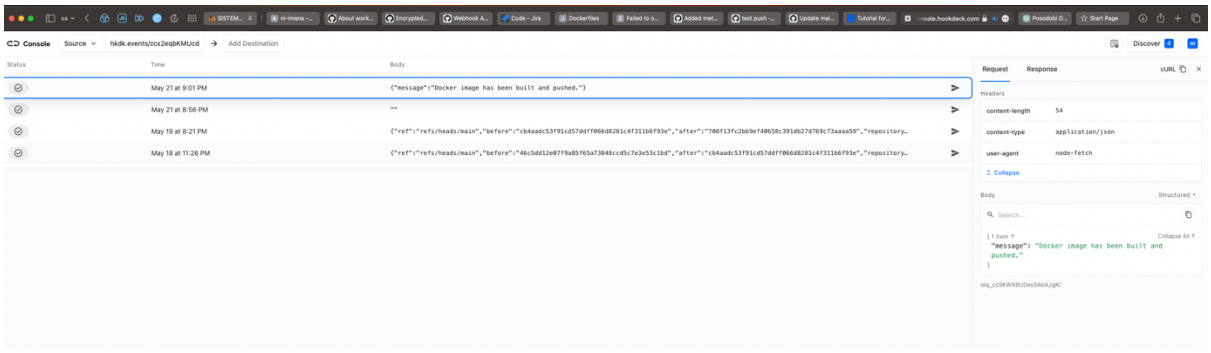


```
ni-imena-admin@ni-imena:~/SISTEMSKA_ADMINISTRACIJA$ sudo hookdeck listen 1337                {"message":"Docker image has been built and pushed."}
? Select a source Create new source
? What should be your new source label? pn3
? What path should the webhooks be forwarded to (ie: /webhooks)? /log-github-webhook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

Workflow file for this run

.github/workflows/main.yml at d3b8557

```yaml
name: Docker Build and Push

on:
  push:
    branches:
      - main

env:
  DOCKER_USERNAME: ${{ secrets.DOCKERHUB_USERNAME }}
  DOCKER_PASSWORD: ${{ secrets.DOCKERHUB_TOKEN }}
  DOCKER_IMAGE: ${{ secrets.DOCKERHUB_IMAGE }}

jobs:
  build-and-push:
    runs-on: ubuntu-latest

    steps:
      - name: Checkout code
        uses: actions/checkout@v2

      - name: Login to Docker Hub
        uses: docker/login-action@v2.1.0
        with:
          username: ${{ env.DOCKER_USERNAME }}
          password: ${{ env.DOCKER_PASSWORD }}

      - name: Build and push Docker image
        run: |
          docker build -t $DOCKER_IMAGE .
          docker push $DOCKER_IMAGE

      - name: Send webhook notification
        uses: joelwmale/webhook-action@2.3.2
        with:
          url: http://20.126.63.144:1337/log-github-webhook
          method: POST
          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. **Ponavljanje (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 pomislekov pri uporabi Webhook-ov lahko upoštevatmo 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 prestrezanje in nepooblaščeno 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.