



Univerza v Mariboru

Fakulteta za elektrotehniko,
računalništvo in informatiko

Dokumentacija

(ni imena?)

Študent:
Domen Hribenik
Denis Železnik
Merisa Mustajbašić

Datum in mesto:
4.06.2023, Maribor

Kazalo

1.	Namestitev v Docker-ju	1
1.1.	Namestitev MongoDB	1
1.2.	Docker in GitHub	2
2.	Vzpoztavitev Azure VM	4
2.1.	Kreiranje VM	4
2.2.	Vprašanja o Azure portalu	7
	Kje in kako omogočite "port forwarding" ?	8
	Kakšen tip diska je bil dodan vaši navidezni napravi in kakšna je njegova kapaciteta ?	8
	Kje preverimo stanje trenutne porabe virov v naši naročnini ("Azure for students") ?	8
2.3.	Vzpoztavitev Docker aplikacije	9
	Uporaba git-a za prenos kode	9
	Namestite vse potrebne pakete za vašo aplikacijo	9
	Omogočite dostopnost vaše aplikacije iz javnega omrežja	10
3.	Docker Hub container registry.....	11
3.1.	Namestitev Docker Hub Registry	11
4.	GitHub Actions workflows.....	15
4.1.	GitHub Actions Workflow ideje za projekt ni_imena?	16
5.	Webhook	17
5.1.	Varnostne luknje pri uporabi Webhook	19

1. Namestitev v Docker-ju

1.1. Namestitev MongoDB

Najprej smo ustvarili račun na spletni strani MongoDB in nato ustvarili novo podatkovno bazo za naš projekt (Slika 1). V bazo smo dodali ustrezne kolekcije in dokumente, ki jih potrebujemo za spletno stran. Nato smo povabili še ostale člane skupine, da lahko vsi dostopamo do baze.

The screenshot shows the MongoDB Compass interface for the 'Ni-Imena' database. At the top, there are buttons for 'Connect', 'View Monitoring', 'Browse Collections', and '...'. Below this, there's a section titled 'Enhance Your Experience' with a green button labeled 'Upgrade'. On the right, there are monitoring metrics: R: 0, W: 0, Last 6 hours: 100.0/s. The main area lists users:

Display Name	Email Address	Project Role	Created	Last Login	Action
Denis Železnik	denis.zeleznik@student.um.si	Project Owner	04/23/23 - 07:18:40 PM	04/24/23 - 08:20:25 AM	
PENDING INVITE	domen.hribernik@student.um.si	--invite sent--	04/23/23 - 07:31:27 PM		
PENDING INVITE	merisa.mustajbasic@student.um.si	--invite sent--	04/23/23 - 07:31:27 PM		

Slika 1 - Ustvarjena baza projekta

User Management				
Display Name	Email Address	Project Role	Created	Last Login
Denis Železnik	denis.zeleznik@student.um.si	Project Owner	04/23/23 - 07:18:40 PM	04/24/23 - 08:20:25 AM
PENDING INVITE	domen.hribernik@student.um.si	--invite sent--	04/23/23 - 07:31:27 PM	
PENDING INVITE	merisa.mustajbasic@student.um.si	--invite sent--	04/23/23 - 07:31:27 PM	

Slika 2 - povabljeni člani skupine

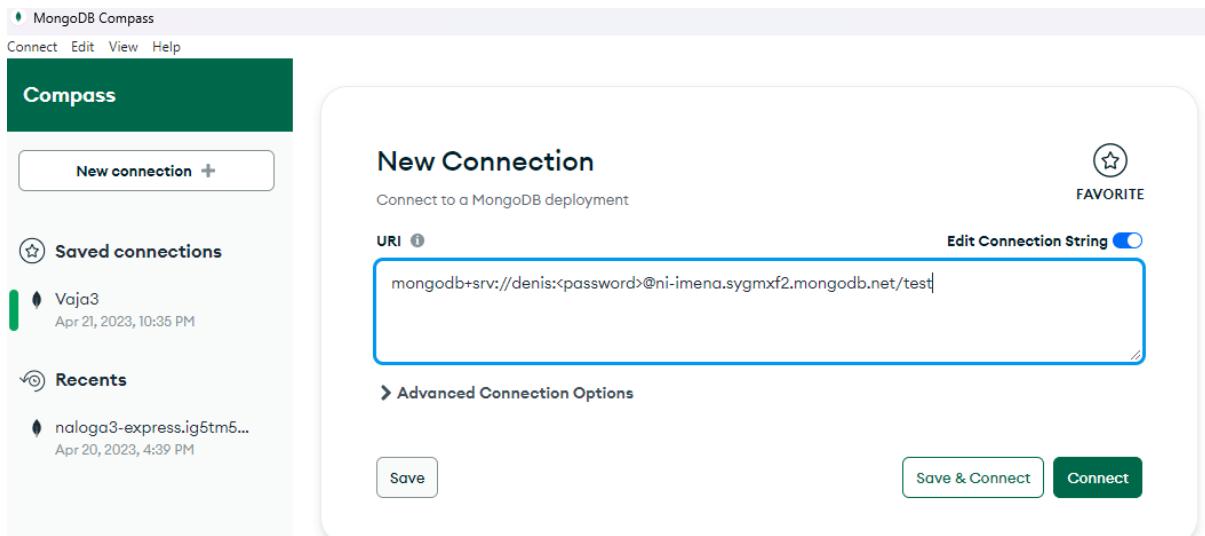
Naslednji korak je bil namestitev orodja MongoDBCompass, ki omogoča pregled in upravljanje z našo podatkovno bazo (Slika 3).

- Copy the connection string, then open MongoDB Compass.

```
mongodb+srv://denis:<password>@ni-imena.sygmxf2.mongodb.net/test
```

You will be prompted for the password for the **denis** user's (Database User) username. When entering your password, make sure that any special characters are **URL encoded**.

Slika 3 – link za povezavo na MongoDB compass



Slika 4 - povezava na MongoDB Compass

Po povezavi z bazo smo se prepričali, da je baza vidna in deluoča (Slika 5).

Collection	Storage size:	Documents:
companies	6.44 MB	3 K
grades	7.62 MB	100 K
inspections	8.82 MB	80 K
routes		

Slika 5 - vidna in deluoča baza na MongoDB Compass

1.2. Docker in GitHub

spec	23/04/2023 22:28	File folder
src	23/04/2023 22:28	File folder
Dockerfile	23/04/2023 21:44	File
package.json	20/04/2023 15:58	JSON Source File
yarn.lock	20/04/2023 15:58	LOCK File
		1 KB
		1 KB
		148 KB

Slika 6 - kloniran projekt z GitHuba

Za nadaljevanje smo potrebovali Docker, ki smo ga namestili na naš računalnik. Po namestitvi smo klonirali naš projekt iz Githuba (Slika 6) in dodali potrebne datoteke ter Dockerfile (Slika 7 - vsebina Dockerfile datoteke) za node.js, ki je vseboval osnovna navodila za namestitev in konfiguracijo.

```

# syntax=docker/dockerfile:1

FROM node:18-alpine
WORKDIR /app
COPY . .
RUN yarn install --production
CMD [ "node", "src/index.js" ]
EXPOSE 3000

```

Slika 7 - vsebina Dockerfile datoteke

Nato smo izvedli ukaz za izgradnjo Docker slike (Slika 8 - izvedba ukaza za izgradnjo Docker image), ki je vsebovala našo spletno stran in vse potrebne odvisnosti. Ko je slika bila uspešno zgrajena (Slika 9 - izgrajen Docker image), smo jo uporabili za ustvarjanje Docker kontejnerja.

```

D:\Github Desktop\ni_imena\Docker\app>docker build -t ni_imena .
[+] Building 2.1s (13/13) FINISHED
=> [internal] load build definition from Dockerfile
=> => transferring dockerfile: 191B
=> [internal] load .dockerignore
=> => transferring context: 2B
=> resolve image config for docker.io/docker/dockerfile:1
=> CACHED docker-image://docker.io/docker/dockerfile:1@sha256:39b85bbfa7536a5feceb7372a0817649ecb2724562a38360f4
=> [internal] load .dockerignore
=> [internal] load build definition from Dockerfile
=> [internal] load metadata for docker.io/library/node:18-alpine
=> [1/4] FROM docker.io/library/node:18-alpine@sha256:ca5d399560a9d239cbfa28eec00417f1505e5e108f3ec6938d230767ea
=> [internal] load build context
=> => transferring context: 4.62MB
=> CACHED [2/4] WORKDIR /app
=> CACHED [3/4] COPY . .
=> CACHED [4/4] RUN yarn install --production
=> exporting to image
=> => exporting layers
=> => writing image sha256:60ab7b7d30f9851359cf90c9cb6da30f7f06d87c947a1b8b6f4037a8f2fd5b57
=> => naming to docker.io/library/ni_imena

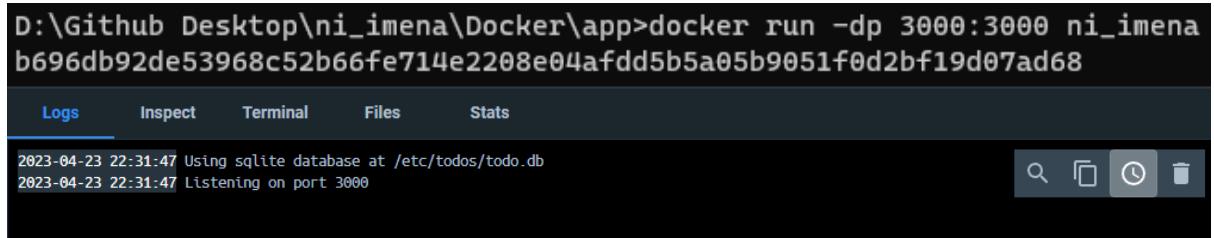
```

Slika 8 - izvedba ukaza za izgradnjo Docker image

D:\Github Desktop\ni_imena\Docker\app>docker images					
REPOSITORY	TAG	IMAGE ID	CREATED	SIZE	
ni_imena	latest	60ab7b7d30f9	44 minutes ago	265MB	
Name	Tag	Status	Created	Size	Actions
ni_imena 60ab7b7d30f9	latest	Unused	44 minutes ago	264.71 MB	

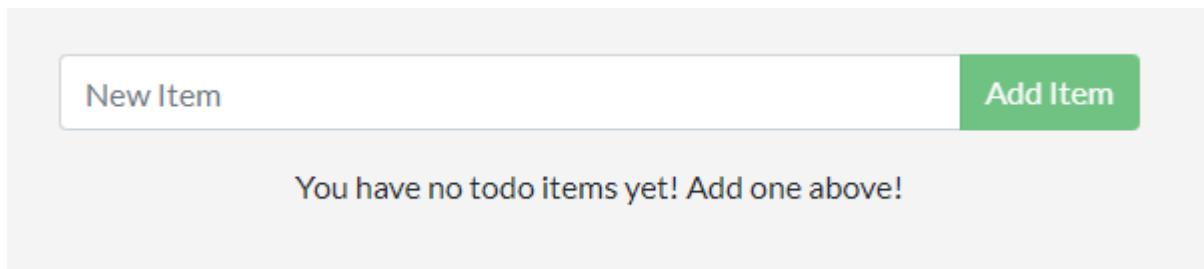
Slika 9 - izgrajen Docker image

Kontejner je bil uspešno ustvarjen in smo ga zagnali na lokalnem računalniku preko localhost:3000 (Slika 10 - izgrajen kontejner za spletno stran). Spletna stran je sedaj delujoča in dostopna preko spletnega brskalnika (Slika 11).



```
D:\Github\Desktop\ni_imena\Docker\app>docker run -dp 3000:3000 ni_imena
b696db92de53968c52b66fe714e2208e04afdd5b5a05b9051f0d2bf19d07ad68
Logs Inspect Terminal Files Stats
2023-04-23 22:31:47 Using sqlite database at /etc/todos/todo.db
2023-04-23 22:31:47 Listening on port 3000
```

Slika 10 - izgrajen kontejner za spletno stran



Slika 11 - delujoča spletna stran nameščena prek Docker-ja

Na koncu smo še pushali naše spremembe na Github, da lahko vsi člani naše skupine dostopajo do najnovejše različice kode.

2. Vzpoztavitev Azure VM

2.1. Kreiranje VM

Project details

Select the subscription to manage deployed resources and costs. Use resource groups like folders to organize and manage all your resources.

Subscription * (Disabled) Azure for Students
The selected subscription is disabled.

Resource group * (New) ni_imena
Create new

Virtual machine name * ni-imena-server

Region * (Europe) West Europe

Image * Ubuntu Server 16.04 LTS - x64 Gen1

Size * Standard_B1s - 1 vcpu, 1 GiB memory (US\$8.76/month)
See all sizes

Administrator account

Authentication type * SSH public key
Password

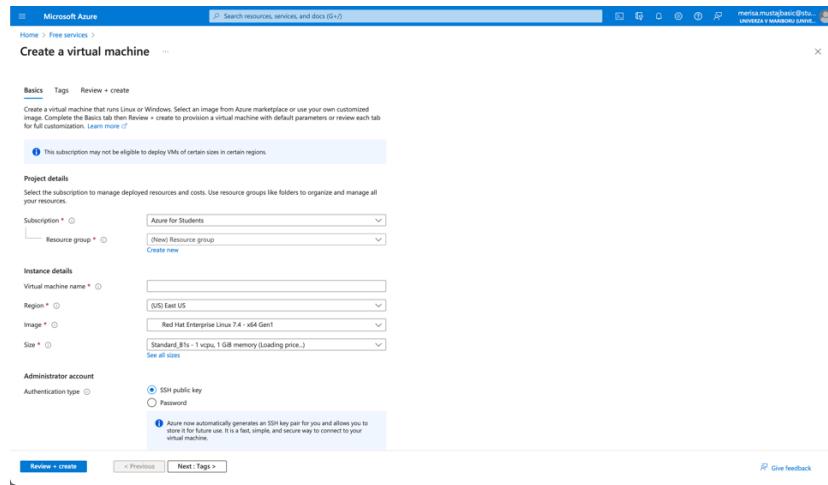
Azure now automatically generates an SSH key pair for you and allows you to store it for future use. It is a fast, simple, and secure way to connect to your virtual machine.

Review + create < Previous Next: Tags > **give feedback**

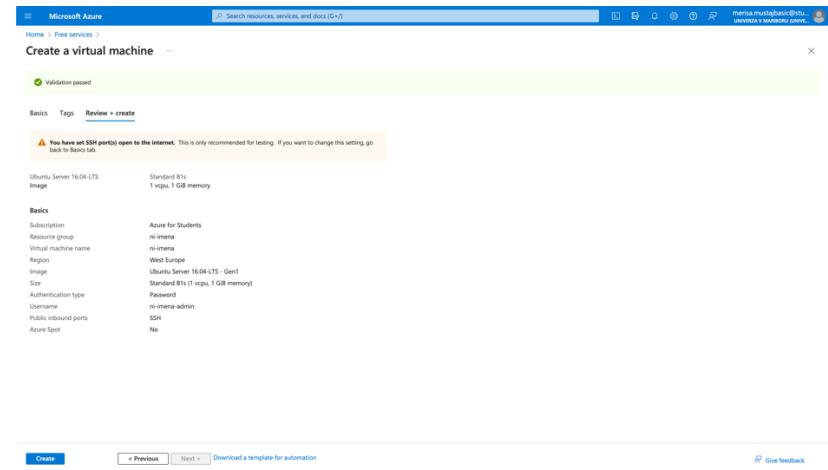
Slika 12 - napaka pri vzpostavitvi, račun ni aktiven

Po uspešni registraciji za brezplačnih 100 evrov storitev na Microsoft Azure (Slika 13), namestimo navidezni stroj kot je opisano v navodilima (Slika 14).

Slika 13 - unovčitev brezplačnih storitev na Microsoft Azure

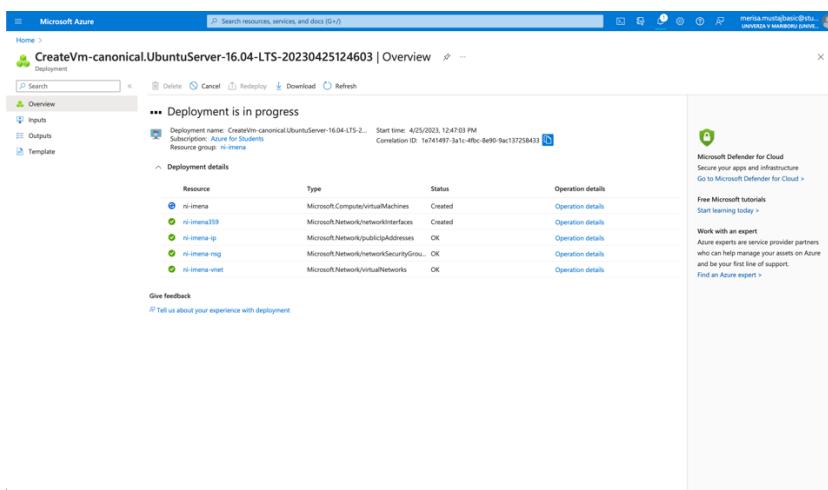


Slika 14 - specifikacije navideznega stroja



Slika 15 - overview VM-a

Ko smo preverili vse podane informacije in specifikacije, začnemo proces deploymenta našega stroja (Slika 16). Po končanem deployment-u smo uspešno naredili Linux navidezni stroj (Slika 17).



Slika 16 - deployment VM-a

Slika 17 - ustvarjen VM

SSH dostop za vse člane je zagotovljen preko javnega IP naslova:

```
C:\Users\Denis>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

UA Infra: Extended Security Maintenance (ESM) is not enabled.

1 update can be applied immediately.
To see these additional updates run: apt list --upgradable

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

Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.

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

Last login: Wed Apr 26 20:31:45 2023 from 109.182.226.177
ni-imena-admin@ni-imena:~$
```

```
C:\Users\Denis>ssh ni-imena-admin@20.126.63.144
The authenticity of host '20.126.63.144 (20.126.63.144)' can't be established.
ED25519 key fingerprint is SHA256:HC1qkf2Anpur4h7BqI9uv7s36ghHNHxhTDA9pvegl.
This key is not known by any other names
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '20.126.63.144' (ED25519) to the list of known hosts.
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

UA Infra: Extended Security Maintenance (ESM) is not enabled.

1 update can be applied immediately.
To see these additional updates run: apt list --upgradable

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

Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.

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

Last login: Wed Apr 26 19:48:38 2023 from 46.123.250.45
See "man sudo_root" for details.
ni-imena-admin@ni-imena:~$
```

Slika 18 - ssh dostop vseh članov

2.2. Azure portal osnove

Kje in kako omogočite "port forwarding" ?

V Azure portalu lahko omogočimo "port forwarding" za navidezno napravo (virtual machine) tako, da v meniju navidezne naprave izberemo možnost "Networking". Nato izberemo "Add inbound port rule" in nastavimo pravila za "port forwarding" glede na naše potrebe. Na primer, mi smo rabili omogočiti promet na portu 3000 kako bi lahko dostopali do njega preko javnega IP naslova, zato smo dodali pravilo ki nam to omogoči.

Priority	Name	Port	Protocol	Source	Destination	Action
300	SSH	22	TCP	Any	Any	Allow
310	AllowAnyCustom3000Inbound	3000	Any	Any	Any	Allow
65000	AllowVnetInBound	Any	Any	VirtualNetwork	VirtualNetwork	Allow
65001	AllowAzureLoadBalancerInBound	Any	Any	AzureLoadBalancer	Any	Allow
65500	DenyAllInBound	Any	Any	Any	Any	Deny

Slika 19 - port forwarding

Kakšen tip diska je bil dodan vaši navidezni napravi in kakšna je njegova kapaciteta ?

V meniju navidezne naprave v Azure portalu lahko preverimo tip diska, ki je bil dodan naši navidezni napravi, tako da kliknemo na "Disks". Tam vidimo seznam diskov, ki so na voljo za navidezno napravo, vključno z njihovo kapaciteto in tipom diska (npr. HDD ali SSD). V našem primeru vrsta diska je Premium SSD LRS in kapaciteta 64 GB.

Kje preverimo stanje trenutne porabe virov v naši naročnini ("Azure for students") ?

V Azure portalu lahko preverimo trenutno porabo virov v naši naročnini tako, da izberemo možnost "Cost Management + Billing" v levem meniju.

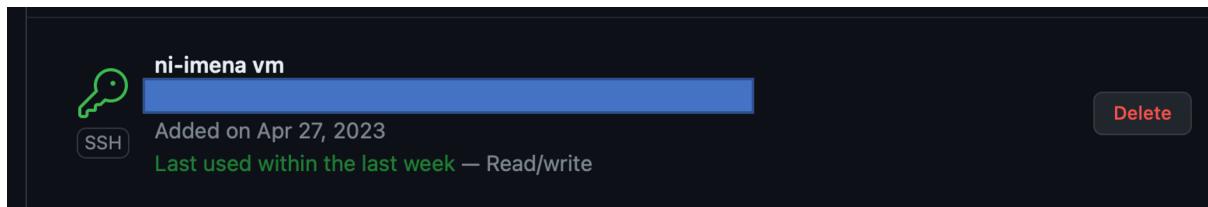
The screenshot shows the Microsoft Azure portal with the 'Subscriptions' view selected. A specific subscription named 'Univerza v Mariboru' is highlighted. The main content area displays the 'Azure for Students' dashboard. Key sections include:

- Essentials:** Shows Subscription ID, Directory, Status, My role (Owner), Plan (Azure Plan), and Secure Score (Not available).
- Spending rate and forecast:** A chart showing current usage at €0.09 and a forecasted total of €2.19.
- Costs by resource:** A donut chart showing costs for 'ri-imena-ip' at €0.09.
- Top free services by usage:** A table listing services and their usage status. Services listed include Storage Premium Page Blobs, Virtual Machines, Networking, Azure Cosmos DB, and Storage Files.
- Azure Defender coverage:** A section indicating that Azure Defender is not enabled for this subscription, with a 'Upgrade coverage' button.

Slika 20 - poraba virov

2.3. Vzpostavitev Docker aplikacije

Uporaba git-a za prenos kode



Slika 21 - git za prenos kode projekta

Omogočimo najprej ssh povezavo torej naredimo keygen ki ga dodamo na github ki gostuje organizaciju za projekt. Z tem lahko kloniramo repozitorije prek SSH.

Namestite vse potrebne pakete za vašo aplikacijo

Nameščen je docker, mongocli, node itd.

Struktura projekta je prikazana

```

[ni-imena-admin@ni-imena:~$ cd SPLETNO_PROGRAMIRANJE/
[ni-imena-admin@ni-imena:~/SPLETNO_PROGRAMIRANJE$ ls
Dockerfile bin node_modules public views
README.md controllers package-lock.json routes
app.js models package.json verifyJWT.js
[ni-imena-admin@ni-imena:~/SPLETNO_PROGRAMIRANJE$ docker build -t pn1 .
Sending build context to Docker daemon 40.18MB
Step 1/7 : FROM node:18
--> c9e4d88ad304
Step 2/7 : WORKDIR /SPLETNO_PROGRAMIRANJE
--> Using cache
--> 5d0696a8c396
Step 3/7 : COPY package*.json .
--> Using cache
--> 55a6069ce298
Step 4/7 : RUN npm install
--> Using cache
--> 3ed2a90c56b8
Step 5/7 : COPY . /SPLETNO_PROGRAMIRANJE
--> 94368cfaf614
Step 6/7 : EXPOSE 3000
--> Running in 34d79d77665a
Removing intermediate container 34d79d77665a
--> 8dcc2cbec7b8
Step 7/7 : CMD ["npm", "run", "dev"]
--> Running in aa3192eb65c4
Removing intermediate container aa3192eb65c4
--> 6853b7577038
Successfully built 6853b7577038
Successfully tagged pn1:latest
[ni-imena-admin@ni-imena:~/SPLETNO_PROGRAMIRANJE$ hstr

```

Slika 22 - struktura folderja, Dockerfile ipd

Omogočite dostopnost vaše aplikacije iz javnega omrežja

Z ukazom „docker run“ zaženemo naš container in je aplikacija potem dostopna preko javnega naslova VM-a, s dodanom številko vrat (v našem primeru je to 3000).

The screenshot shows a terminal window with the following content:

```

merisa - ni-imena-admin@ni-imena: ~/SPLETNO_PROGRAMIRANJE -- ssh ni-imena-admin@20.12...
Last login: Thu May 4 20:39:26 2023 from 109.182.226.177
[ni-imena-admin@ni-imena:~$ ls
SISTEMSKA_ADMINISTRACIJA SPLETNO_PROGRAMIRANJE app anni_imena cpu,1 GiB memory)
[ni-imena-admin@ni-imena:~$ cd SPLETNO_PROGRAMIRANJE/
[ni-imena-admin@ni-imena:~/SPLETNO_PROGRAMIRANJE$ ls
Dockerfile bin node_modules public ni-imena_views_ault
README.md controllers package-lock.json routes
app.js models package.json name verifyJWT.js
[ni-imena-admin@ni-imena:~/SPLETNO_PROGRAMIRANJE$ docker build -t pn1 .
Sending build context to Docker daemon 40.18MB
Step 1/7 : FROM node:18
--> c9e4d88ad304
Step 2/7 : WORKDIR /SPLETNO_PROGRAMIRANJE
--> Using cache
--> 5d0696a8c396
Step 3/7 : COPY package*.json .
--> Using cache
--> 55a6069ce298
Step 4/7 : RUN npm install
--> Using cache
--> 3ed2a90c56b8
Step 5/7 : COPY . /SPLETNO_PROGRAMIRANJE
--> 94368cfaf614
Step 6/7 : EXPOSE 3000
--> Running in 34d79d77665a
Removing intermediate container 34d79d77665a
--> 8dcc2cbec7b8
Step 7/7 : CMD ["npm", "run", "dev"]
--> Running in aa3192eb65c4
Removing intermediate container aa3192eb65c4
--> 6853b7577038
Successfully built 6853b7577038
Successfully tagged pn1:latest
[ni-imena-admin@ni-imena:~/SPLETNO_PROGRAMIRANJE$ hstr

```

Below the terminal, there is a summary table of network settings:

	Networking	
Public IP address	20.126.63.144 (Network interface ni-imena359)	
Public IP address (IPv6)	-	
Private IP address	10.1.1.4	
Private IP address (IPv6)	-	
Virtual network/subnet	ni-imena-vnet/default	
DNS name	Configure	

```

docker run -dp 3000:3000 pn1
ni-imena-admin@ni-imena:~/SPLETNO_PROGRAMIRANJE$ docker run -dp 3000:3000 pn1
ced29961584cdc13c8abbede22fd68b464d65e830ee52acf4729896362da5934
ni-imena-admin@ni-imena:~/SPLETNO_PROGRAMIRANJE$
```

The screenshot shows a Docker container running a web application. At the top, a terminal window shows the command to run the Docker container and its ID. Below it, two screenshots of the web application are displayed side-by-side. The left screenshot shows a 'Registracija' (Registration) page with fields for E-mail, Uporabniško ime (Username), and Gost (Guest). The right screenshot shows the 'Ni_Imena' homepage with the message 'Welcome to Ni_Imena'. At the bottom, another terminal window shows the Docker logs, detailing the start of a nodemon process and several HTTP requests being handled.

```

[ni-imena-admin@ni-imena:~/SPLETNO_PROGRAMIRANJE$ docker logs ced29961584cdc13c8abbede22fd68b464d65e830ee52acf4729896362da5934
> nalogal1@1.0.0 dev
> nodemon ./bin/www

[nodemon] 2.0.22
[nodemon] to restart at any time, enter `rs`
[nodemon] watching path(s): *.*-imena-admin@2.0.12...
[nodemon] watching extensions: js,mjs,json
[nodemon] starting `node ./bin/www`
GET / 200 69.379 ms - 1238
GET /users/login 200 5.831 ms - 1956
GET / 304 11.880 ms - -
GET /users/register 200 4.309 ms - 2111
GET /map 404 6.496 ms - 2546
GET / 304 5.008 ms -
```

Slika 23 - dostop preko javnega IP naslova in številke vrat

3. Docker Hub container registry

3.1. Namestitev Docker Hub Registry

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

```

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

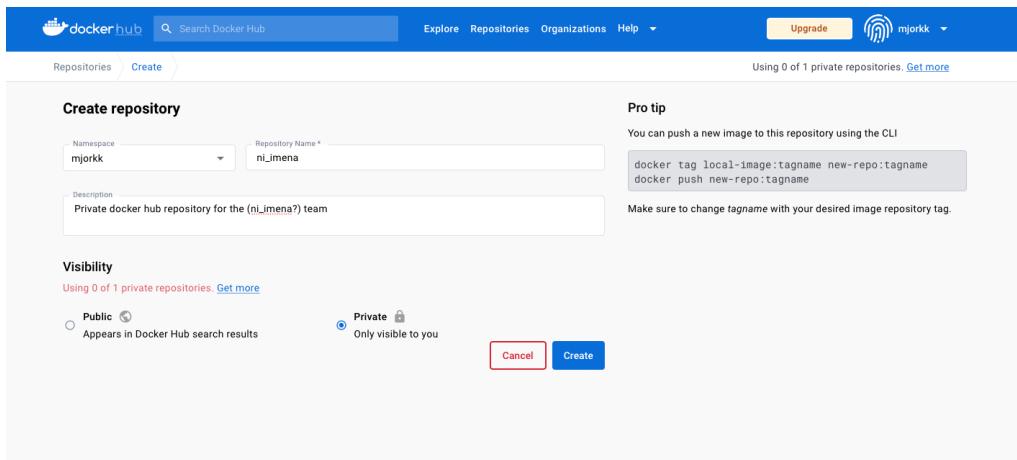
Login Succeeded
ni-imena-admin@ni-imena:~$ docker system info
Client:
  Context: default
  Debug Mode: false
  Plugins:
    app: Docker App (Docker Inc., v0.9.1-beta)
    buildx: Build with BuildKit (Docker Inc., v0.6.1-docker)
    scan: Docker Scan (Docker Inc., v0.8.0)

Server:
  Containers: 4
  Running: 0
  Paused: 0
  Stopped: 4
  Images: 22
  Server Version: 20.10.7
  Storage Driver: overlay2
    Backing Filesystem: ext4
    Supports d_type: true
    Native Overlay Diff: false
  Log Driver: json-file
  Cgroup Driver: cgroups
  Cgroup Version: 1
  Plugins:
    Volume: logs
    Network: bridge host ipvlan macvlan null overlay
    Log: awslogs fluentd gelf journald json-file local logentries splunk syslog
  Swap: inactive
  Kernel Version: 4.15.0-113-azure
  Operating System: Ubuntu 16.04.7 LTS
  OSType: linux
  Architecture: x86_64
  CPUs: 1
  Total Memory: 921.7MiB
  Name: ni_imena
  ID: LAZC:U3VX:CMH0:KVS:YT4J:SDVI:TARD:G03S:2A07:HZL:NSS0:H4l
  Docker Root Dir: /var/lib/docker
  Debug Mode: false
  User: root
  Registry: https://index.docker.io/v1/
  Labels:
    experimental: false
    image registries:
      10.7.0.6/0/3
    Live Restore Enabled: false
  WARNING: No swap limit support

```

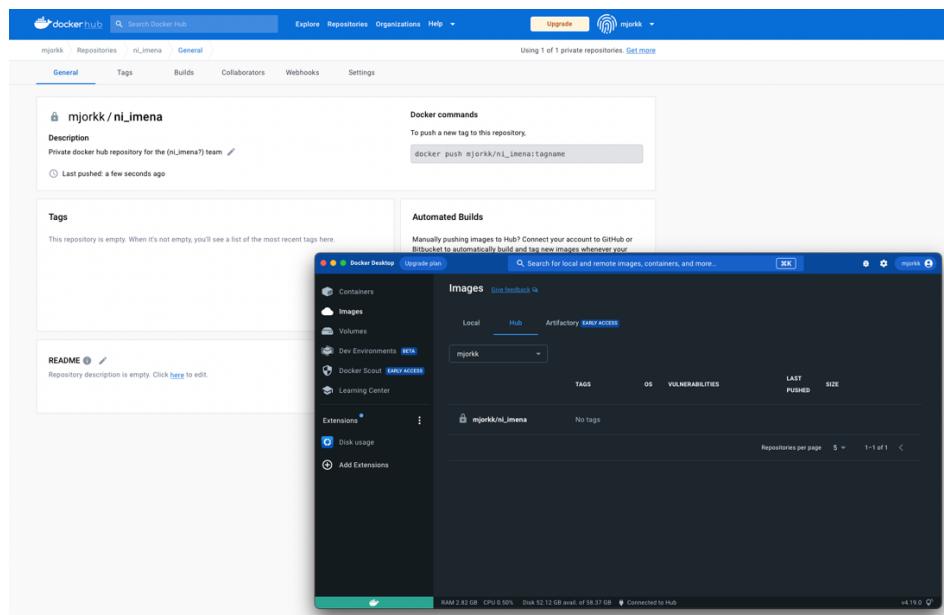
Slika 24 - namestitev Docker Hub Registry

Nato ustvarimo repozitorij in nastavimo da je private.



Slika 25 - nastavitev Docker Hub repozitorija

Repozitorij je zdaj viden tudi preko namizne aplikacije docker hub.



Slika 26 - namiyina aplikacija 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 - 141x66
-- ni-imena-admin@ni-imena: ~/SPLETNO_PROGRAMIRANJE - ssh ni-imena-admin@20.126.63.144 ...
ni-imena-admin@20.126.63.144: ~
Last login: Fri May 12 09:31:07 on ttys000
reboot   May 12 09:48:14
reboot   May 12 09:48:14
-- % ssh ni-imena-admin@20.126.63.144
ni-imena-admin@20.126.63.144: ~ 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/2e-04

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

Last Login: Fri May 12 07:37:04 2018 from 89.143.18.159
ni-imena-admin@ni-imena:~/SPLETNO_PROGRAMIRANJE/
ni-imena-admin@ni-imena:~/SPLETNO_PROGRAMIRANJE docker ps
CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES
f3e2c2b848ec "node:latest -e PORT=8080 -e entrypoint.sh" 2 seconds ago Up 64 seconds 0.0.0.0:8080/tcp goofy_morse
ni-imena-admin@ni-imena:~/SPLETNO_PROGRAMIRANJE docker stop p01
Error response from daemon: No such container: p01
ni-imena-admin@ni-imena:~/SPLETNO_PROGRAMIRANJE docker stop f3e2c2b848ec
f3e2c2b848ec
ni-imena-admin@ni-imena:~/SPLETNO_PROGRAMIRANJE docker ps
CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES
f3e2c2b848ec "node:latest -e PORT=8080 -e entrypoint.sh" 2 seconds ago Up 64 seconds 0.0.0.0:8080/tcp goofy_morse
ni-imena-admin@ni-imena:~/SPLETNO_PROGRAMIRANJE docker run -d -p 3000:3000 p01
ni-imena-admin@ni-imena:~/SPLETNO_PROGRAMIRANJE docker run -d -p 3000:3000 p01
4a6ed9ad261ed9ef8e80d958f205a7 ... 1 second ago 0.0.0.0:3000/tcp 7979439c
ni-imena-admin@ni-imena:~/SPLETNO_PROGRAMIRANJE docker images
REPOSITORY TAG IMAGE ID CREATED SIZE
p01 latest 28535779383 3 days ago 1.97GB
p01 latest 24959bedadd 10 days ago 1.11GB
node 18 <e4dd8a3d38a 4 weeks ago 937MB
ni-imena-admin@ni-imena:~/SPLETNO_PROGRAMIRANJE docker image tag p01 mjorkk/p01:latest
ni-imena-admin@ni-imena:~/SPLETNO_PROGRAMIRANJE docker image push mjorkk/p01:latest
The push refers to repository [docker.io/mjorkk/p01]
4cf68d3f3fdd: Pushed
4dfe1235fd7e: Pushed
4d86f264a173: Pushed
4d86f264a173: Pushed
4d86f264a173: Pushed
4dfe1235fd7e: Pushed
128128ebb44b: Mounted from library/node
247467fb9b4b: Mounted from library/node
77838e97d74: Mounted from library/node
e29838aef63d: Mounted from library/node
4cf68d3f3fdd: Mounted from library/node
b86f264a173a: Mounted from library/node
6a1ebbb78bdc: Mounted from library/node
24b48387f467: Mounted from library/node
24b48387f467: Mounted from library/node
latest: digest: sha256:c77729765433d491c3cd656b80abdb575feb5621b43d540d9cc743b407ca4921 size: 3064
ni-imena-admin@ni-imena:~/SPLETNO_PROGRAMIRANJE |
```

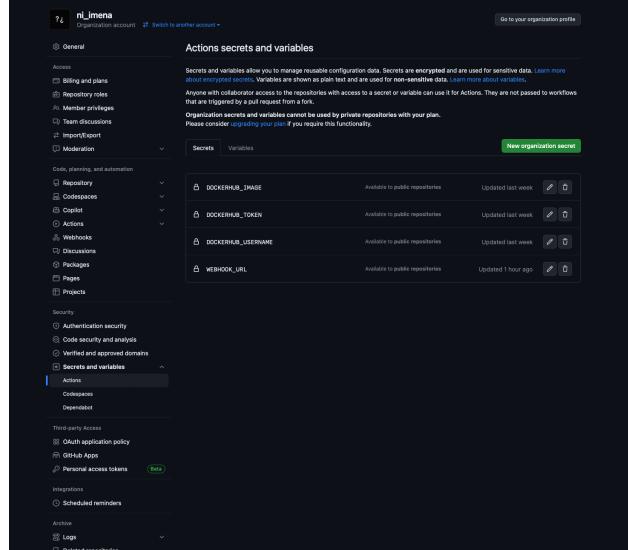
```
ni-imena-admin@ni-imena:~/SPLETNO_PROGRAMIRANJE$ docker tag p01 mjorkk/ni_imena:latest
ni-imena-admin@ni-imena:~/SPLETNO_PROGRAMIRANJE$ docker push mjorkk/ni_imena:latest
The push refers to repository [docker.io/mjorkk/ni_imena]
c1f68d3f3f8d0: Pushed
dce3b044efc9: Pushed
d6d3ee7c31f6: Pushed
128128ebb44b: Pushed
ca745ff79e3b0: Pushed
77838e97d76: Pushed
e29838aef636: Pushed
cf00811d364e: Pushed
b86f264a173a: Pushed example, lets save a local copy of the
6a1ebbb78bdc: Pushed
24b48387f467: Pushed
ae65c0c5405b: Pushed
latest: digest: sha256:c77729765433d491c3cd656b80abdb575feb5621b43d540d9cc743b407ca4921 size: 3054
ni-imena-admin@ni-imena:~/SPLETNO_PROGRAMIRANJE $
```

The screenshot shows the Docker Hub interface for the repository `mjorkk/ni_imena`. The repository has one tag, `latest`, pushed 2 minutes ago. There is a note about pushing a new tag to the repository. The `Docker commands` section includes the command `docker push mjorkk/ni_imena:tagname`. The `Automated Builds` section indicates that manually pushing images to Hub or connecting to GitHub or Bitbucket for automatic builds is available with Pro, Team, and Business subscriptions.

Slika 27 - prvi push na Docker Hub repozitorij

4. GitHub Actions workflows

Najprej nastavimo vdje potrebne GitHub secrets:



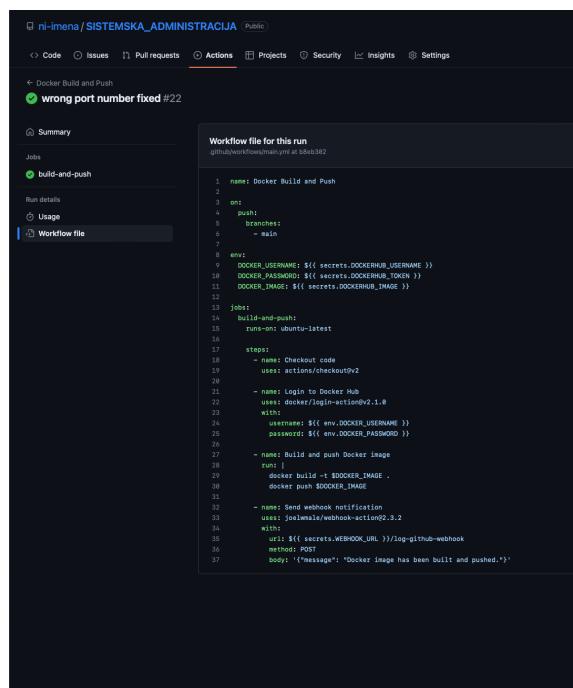
The screenshot shows the 'Actions secrets and variables' section in the GitHub settings. It lists four secrets:

- DOCKERHUB_IMAGE: Available to public repositories, updated last week.
- DOCKERHUB_TOKEN: Available to public repositories, updated last week.
- DOCKERHUB_USERNAME: Available to public repositories, updated last week.
- WEBHOOK_URL: Available to public repositories, updated 1 hour ago.

Slika 28 - 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**

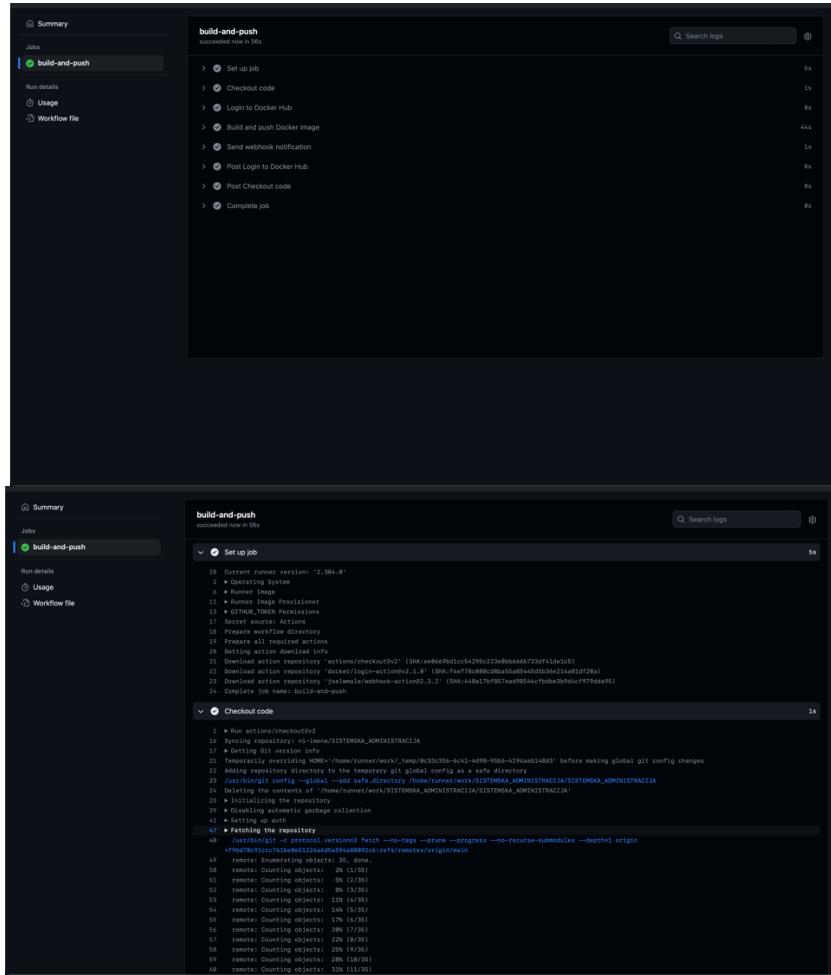


The screenshot shows the GitHub Actions workflow file for the 'build-and-push' job. The workflow defines a job named 'Docker Build and Push' that runs on pushes to the 'main' branch. It uses secrets for Docker credentials and performs a build and push to Docker Hub, followed by sending a webhook notification.

```
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:
    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
        runs-on: ubuntu-latest
        steps:
          - name: Build
            uses: docker/build-action@v2.0.0
            with:
              dockerfile: Dockerfile
              tag: ${{ env.DOCKER_IMAGE }}.latest
          - name: Push
            uses: docker/push-action@v2.0.0
            with:
              repository: ${{ env.DOCKER_IMAGE }}.latest
      - name: Send webhook notification
        uses: tekemla/webhook-action@2.3.2
        with:
          url: ${secrets.WEBHOOK_URL}
          method: POST
          body: {"message": "Docker image has been built and pushed."}
```

Slika 29 - Workflow

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.



Slika 30 - uspešno izveden workflow

4.1. 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 zahtevke (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

5. 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("sjhdfgjiskdf");

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

```

- Napišite preprosto skripto v poljubnem programskem jeziku, ki bo:
 - zustavila trenutno zagnan docker container,
 - iz **Docker Container registry** prenesla najnovejšo različico vašega kontejnera,
 - ga ponovno zagnala.
- Skripto zaženite ob vsakem dobljenem **Webhook** sporočilu.
- Opisite varnostne ukrepe in vaše pomislike pri uporabi **Webhook**, opišite (za zagovor) vaj boste prikazali pravilno delovanje celotnega GitHub Actions.

Na sistem oddajte poročilo v .pdf obliki. Poročilo naj vsebuje opise posameznih nujnih ustreznih podaj (ti, Jiro, OpenProj). Na zagovoru bomo pogledali opis KRITERIJI OCENJEVANJA IMPLEMENTACIJE:

1. točka - Docker Hub container registry
2. točka - deluječ osnovni GitHub Actions workflow
3. točka - opis namestitev 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, ipd.)

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

Slika 31 - Webhook

Instaliramo in zazenemo na portu 1337 hookdeck:

```

inf-1400s-admin@inf-1400s:~/SISTEMSKA_ADMINISTRACIJA$ sudo hookdeck listen 1337
Select a source ...
? What should be your new source label? xyz
? 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=arc_yzTux3ls7QVe
pn3 Source
Webhook URL: https://hkdk.events/yzTux3ls7QVe
Connections
response-server forwarding to /log-github-webhook
> Ready! (^C to quit)
2023-06-21 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
2023-06-21 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: joelwmae/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."}'

```

CD Console Source hdkd.events/zcx2egbKMuod → Add Destination

Status	Time	Body
OK	May 21 at 9:01 PM	{"message": "Docker image has been built and pushed."}
OK	May 21 at 8:59 PM	--
OK	May 19 at 8:21 PM	{"ref": "refs/heads/main", "before": "cb4aa5c53f91cd57ddff866d2814f311bd6f93e", "after": "728f13fc2bb9ef48658c3916b27d769c73aaa59", "repository": "repo_zcx2egbKMuod", "commit": "728f13fc2bb9ef48658c3916b27d769c73aaa59", "author": "node-fetch", "url": "http://20.126.63.144:1337/log-github-webhook"}
OK	May 18 at 11:26 PM	{"ref": "refs/heads/main", "before": "46c5d12e9779ab5f65a7384cc5c7e3e53c1b4", "after": "cb4aa5c53f91cd57ddff866d2814f311bd6f93e", "repository": "repo_zcx2egbKMuod", "commit": "cb4aa5c53f91cd57ddff866d2814f311bd6f93e", "author": "node-fetch", "url": "http://20.126.63.144:1337/log-github-webhook"}

Request Response

Headers

- content-length: 54
- content-type: application/json
- user-agent: node-fetch

Body

```

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

```

Slika 32 - webhook

5.1. 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.