

SISTEMSKA ADMINISTRACIJA,  
CI/CD

Rene Jaušovec

RIT 2 VS

14.4.2024

## KAZALO VSEBINE

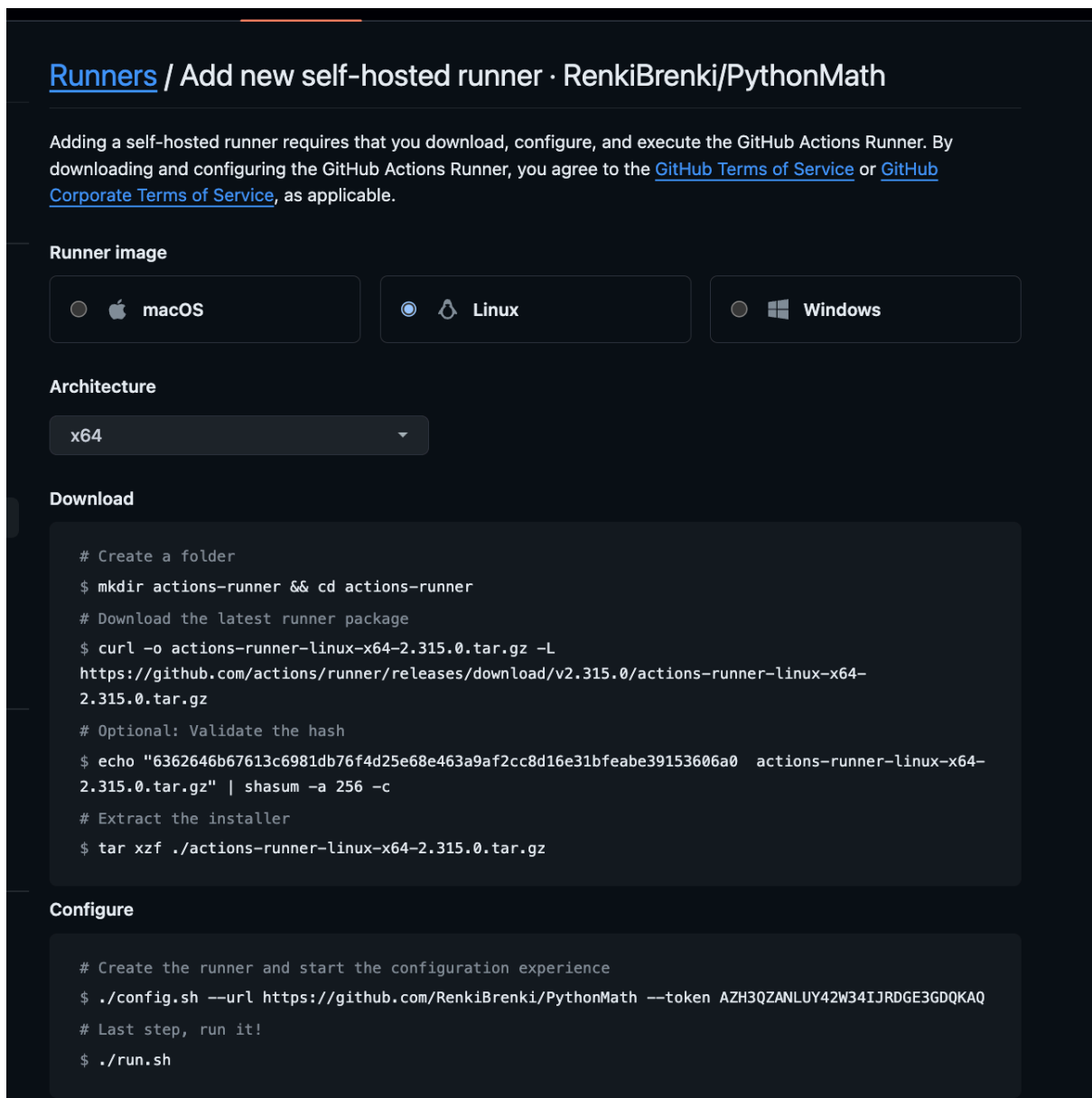
<b>1 Priprava runnerja .....</b>	<b>3</b>
<b>2 Priprava skripte za zagon testov.....</b>	<b>6</b>
<b>3 Stanje po spremembi na Github repozitoriju .....</b>	<b>12</b>

## KAZALO SLIK

<b>Slika 1: Github navodila za self hosted runnerja .....</b>	<b>3</b>
<b>Slika 2: Postopek vzpostavitve runnerja na linux virtualnem okolju (Primer za arhitekturo arm64) .....</b>	<b>4</b>
<b>Slika 3: Napaka pri vzpostavitvi Python okolja, zaradi nepodprtih verzij na arm64 arhitekturi.....</b>	<b>5</b>
<b>Slika 4: Specifikacija dogodkov na vejo main .....</b>	<b>6</b>
<b>Slika 5: Priprava poslov.....</b>	<b>7</b>
<b>Slika 6: Build posel .....</b>	<b>8</b>
<b>Slika 7: Uspešen zagon skripte ob spremembi na glavni veji .....</b>	<b>9</b>
<b>Slika 8: Deploy skripta .....</b>	<b>10</b>
<b>Slika 9: Secrets vrednosti repozitorija.....</b>	<b>11</b>
<b>Slika 10: Skripta zapakiraj.sh .....</b>	<b>11</b>
<b>Slika 11: Docker repozitorij po izvedbi skripte .....</b>	<b>11</b>
<b>Slika 12: Izvedba skript po ukazu push.....</b>	<b>12</b>
<b>Slika 14: Uspešen zagon na linux x64 runnnerju .....</b>	<b>12</b>
<b>Slika 15: Neuspešna izvedba skripte na linux arm64 runnerju .....</b>	<b>13</b>

## 1 Priprava runnerja

Prvi korak pri uporabi CI/CD cevovoda je bil zagon self hosted runnerja. Sprva sem probaval uporabiti linux runnerja na MacOS virtualnem okolju, ki uporablja arm64 arhitekturo. Po številnih neuspešnih zagonih nastavitve python okolja sem ugotovil, da ni nobena verzija pythona podprta z arhitekturo arm64 z akcijo actions/setup-python@v4, zato sem preklopil na Windows računalnik z arhitekturo x64 in uspešno zagnal python okolje.



**Runners** / Add new self-hosted runner · RenkiBrenki/PythonMath

Adding a self-hosted runner requires that you download, configure, and execute the GitHub Actions Runner. By downloading and configuring the GitHub Actions Runner, you agree to the [GitHub Terms of Service](#) or [GitHub Corporate Terms of Service](#), as applicable.

**Runner image**

☐ macOS ☒ Linux ☐ Windows

**Architecture**

x64

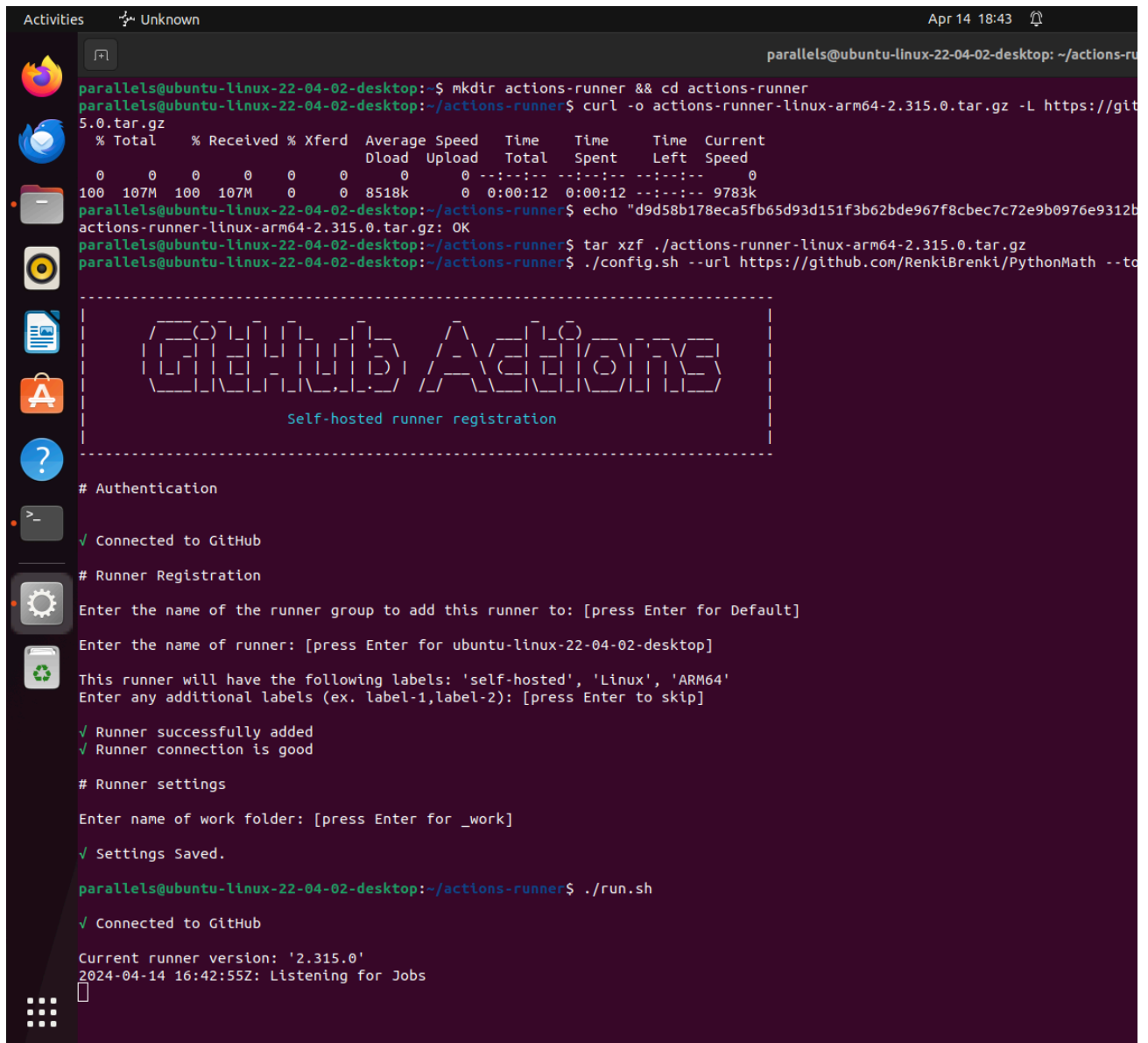
**Download**

```
# Create a folder
$ mkdir actions-runner && cd actions-runner
# Download the latest runner package
$ curl -o actions-runner-linux-x64-2.315.0.tar.gz -L
https://github.com/actions/runner/releases/download/v2.315.0/actions-runner-linux-x64-
2.315.0.tar.gz
# Optional: Validate the hash
$ echo "6362646b67613c6981db76f4d25e68e463a9af2cc8d16e31bfeabe39153606a0 actions-runner-linux-x64-
2.315.0.tar.gz" | shasum -a 256 -c
# Extract the installer
$ tar xzf ./actions-runner-linux-x64-2.315.0.tar.gz
```

**Configure**

```
# Create the runner and start the configuration experience
$ ./config.sh --url https://github.com/RenkiBrenki/PythonMath --token AZH3QZANLUY42W34IJDGE3GDQKAQ
# Last step, run it!
$ ./run.sh
```

Slika 1: Github navodila za self hosted runnerja



```
parallels@ubuntu-linux-22-04-02-desktop:~$ mkdir actions-runner && cd actions-runner
parallels@ubuntu-linux-22-04-02-desktop:~/actions-runner$ curl -o actions-runner-linux-arm64-2.315.0.tar.gz -L https://github.com/actions/runner/releases/download/v2.315.0/actions-runner-linux-arm64-2.315.0.tar.gz
% Total % Received % Xferd Average Speed Time Time Time Current
0 0 0 0 0 0 0 0 0:00:00 0:00:00 0:00:00 0
100 107M 100 107M 0 0 8518k 0 0:00:12 0:00:12 0:00:00 9783k
parallels@ubuntu-linux-22-04-02-desktop:~/actions-runner$ echo "d9d58b178eca5fb65d93d151f3b62bde967f8cbec7c72e9b0976e9312b"
actions-runner-linux-arm64-2.315.0.tar.gz: OK
parallels@ubuntu-linux-22-04-02-desktop:~/actions-runner$ tar xzf ./actions-runner-linux-arm64-2.315.0.tar.gz
parallels@ubuntu-linux-22-04-02-desktop:~/actions-runner$ ./config.sh --url https://github.com/RenkiBrenki/PythonMath --token

-----
GITHUB ACTIONS
Self-hosted runner registration
-----

# Authentication
✓ Connected to GitHub

# Runner Registration
Enter the name of the runner group to add this runner to: [press Enter for Default]
Enter the name of runner: [press Enter for ubuntu-linux-22-04-02-desktop]
This runner will have the following labels: 'self-hosted', 'Linux', 'ARM64'
Enter any additional labels (ex. label-1,label-2): [press Enter to skip]

✓ Runner successfully added
✓ Runner connection is good

# Runner settings
Enter name of work folder: [press Enter for _work]
✓ Settings Saved.

parallels@ubuntu-linux-22-04-02-desktop:~/actions-runner$ ./run.sh

✓ Connected to GitHub

Current runner version: '2.315.0'
2024-04-14 16:42:55Z: Listening for Jobs
```

Slika 2: Postopek vzpostavitve runnerja na linux virtualnem okolju (Primer za arhitekturo arm64)

Vzpostavitev runnerja je precej enostavno, saj samo uporabimo ukaze, ki nam jih pripravi github in runner se brez težav namesti. Za tem izvedemo run.sh skripto in runner čaka na posle v ozadju.

```
✓ ✖ setup Python 0s
1 ▶ Run actions/setup-python@v4
8 ▼ Installed versions
9   Version 3.10.12 was not found in the local cache
10  Error: The version '3.10.12' with architecture 'arm64' was not found for Ubuntu 22.04.
11  The list of all available versions can be found here: https://raw.githubusercontent.com/actions/python-versions/main/versions-
    manifest.json
```

*Slika 3: Napaka pri vzpostavitvi Python okolja, zaradi nepodprtih verzij na arm64 arhitekturi*

## 2 Priprava skripte za zagon testov

Po uspešni nastavitvi self hosted runnerja na linux virtualnem okolju sem začel z pripravo skripte za zagon testov v python okolju z preverjenimi verzijami, ki so podprte na linux x64 arhitekturi.

```
name: CI

on:
  push:
    branches:
      - main
  pull_request:
    branches:
      - main
  workflow_dispatch:
```

Prvi korak priprave skripte je določiti ime akcije. Poimenoval sem jo CI. Nato je potrebno skripti povedati ob katerem dogodku je potrebno zagnati skripto. V mojem primeru se skripta zažene ob push dogodku na glavno vejo main in na pull request na glavno vejo.

Slika 4: Specifikacija dogodkov na vejo main

```

13
14 jobs:
15   checkForTests:
16     runs-on: self-hosted
17     steps:
18       - uses: actions/checkout@v2
19
20       - name: Check if files exist
21         run: |
22           if [ -f $GITHUB_WORKSPACE/main_test.py ]; then
23             echo "File exists"
24             echo 0 > error.txt
25           else
26             echo "File does not exist"
27             exit 1 > error.txt
28           fi
29       - name: Load artifact
30         uses: actions/upload-artifact@v4
31         with:
32           name: file_exists
33           path: error.txt
34           retention-days: 1
35

```

*Slika 5: Priprava poslov*

Nato sem pripravil posle, ki se bodo zagnali ob push dogodku na glavno vejo. Prvi dogodek je checkForTest, ki preveri ali obstajajo testi v repozitoriju. V prvem koraku povemo skripti, da naj se izvede na self hosted runnerju, katerega smo ustvarili v prvem koraku cevovoda. Prvi korak posla je akcija checkout, ki pridobi zadnji commit iz repozitorija in zagotovi, da lahko uporabljamo datoteke v repozitoriju, nato preverimo če datoteka z testi obstaja in zapišemo stanje v datoteko error.txt v katero se ob uspehu zapiše 0 in ob neuspešnosti vrednost 1. Datoteko smo shranili v artefakt file\_exists, zaradi komunikacije med posli.

```

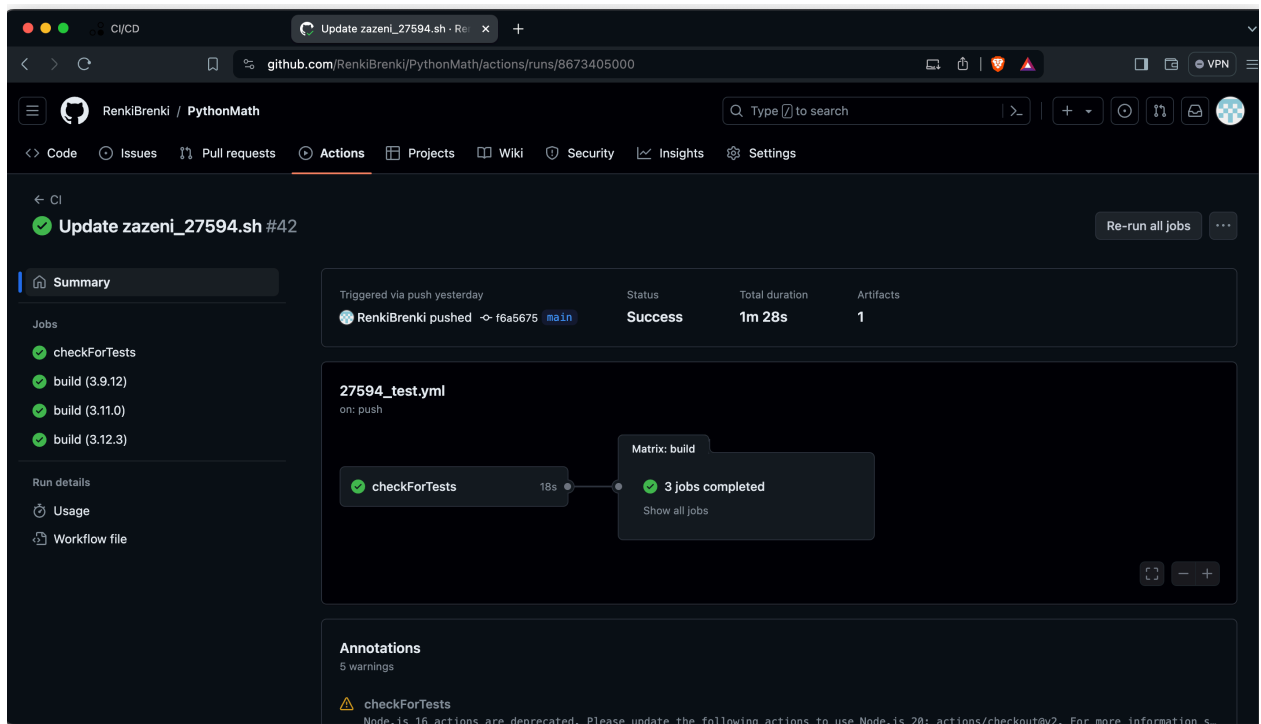
36   build:
37     needs: checkForTests
38     runs-on: self-hosted
39     strategy:
40       matrix:
41         python-version: [3.9.12, 3.11.0, 3.12.3] #3.10.4
42     steps:
43       - uses: actions/checkout@v4
44
45       - name: Download artifact
46         uses: actions/download-artifact@v4
47         with:
48           name: file_exists
49
50       - name: Check artifact
51         shell: bash
52         run: |
53           value=`cat error.txt`
54           rm error.txt
55           if [ "$value" -gt 0 ]; then
56             echo "no file"
57             exit 1
58           fi
59
60       - name: setup Python
61         uses: actions/setup-python@v4
62         with:
63           python-version: ${ matrix.python-version }
64
65       - name: install dependencies
66         run: |
67           python -m pip install --upgrade pip
68           pip install pytest
69       - name: Test
70         run: pytest

```

Slika 6: Build posel

Naslednji posel je build, ki se začne izvajati ko se checkForTests zaključi. Vsebuje matriko python verzij. Posel za vsako python verzijo, ki jo navedemo v matriko izvede teste v main\_test.py. To izvedemo tako, da prenesemo artefakt, katerega smo shranili v prejšnjem poslu, nato preverimo vsebino in če obstaja datoteka pripravimo python okolje in zaženemo teste z pytest ukazom. Pytest izvede datoteki, ki se začne z test\_ ali konča z \_test.





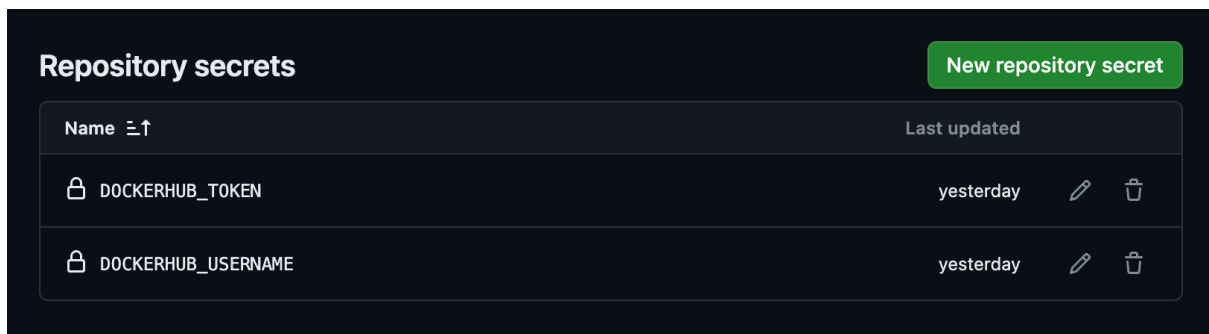
Slika 7: Uspešen zagon skripte ob spremembi na glavni veji

### 3 Priprava skripte za push na DockerHub repozitorij

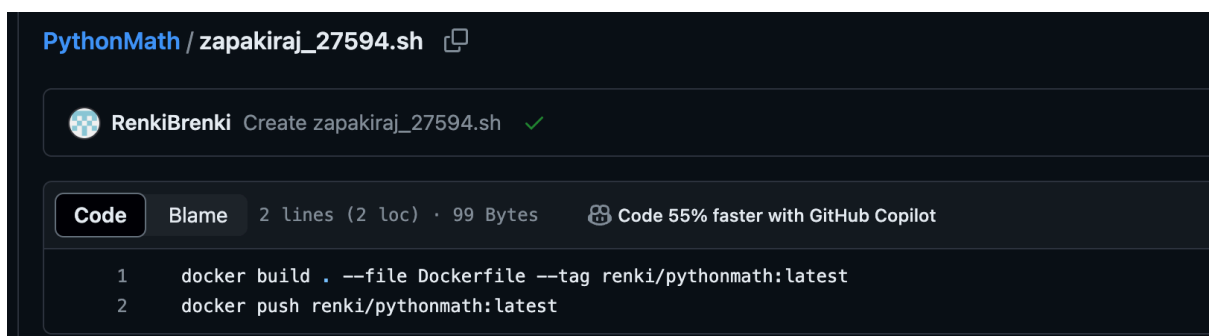
```
1  name: CD
2
3  on:
4    workflow_run:
5      workflows:
6        - "CI"
7      types:
8        - completed
9
10   workflow_dispatch:
11
12   jobs:
13     deploy:
14       runs-on: self-hosted
15       steps:
16         - uses: actions/checkout@v4
17
18         - name: Docker login
19           uses: docker/login-action@v3
20           with:
21             username: ${ secrets.DOCKERHUB_USERNAME }
22             password: ${ secrets.DOCKERHUB_TOKEN }
23
24         - name: Docker push
25           run: |
26             sh zapakiraj_27594.sh
27
```

Slika 8: Deploy skripta

Podobno kot pri prejšnji skripti sem jo poimenoval, tokrat CD. Ta skripta se izvede kadar se skripta CI do konca izvede. Skripta je sestavljena iz posla deploy, ki izvede prijavo v DockerHub z akcijo login-action z uporabniškim imenom in tokenom, katere sem navedel v secrets sekcijo repozitorija. Nato izvede skripto zapakiraj.sh, ki izvede push na DockerHub repozitorij.

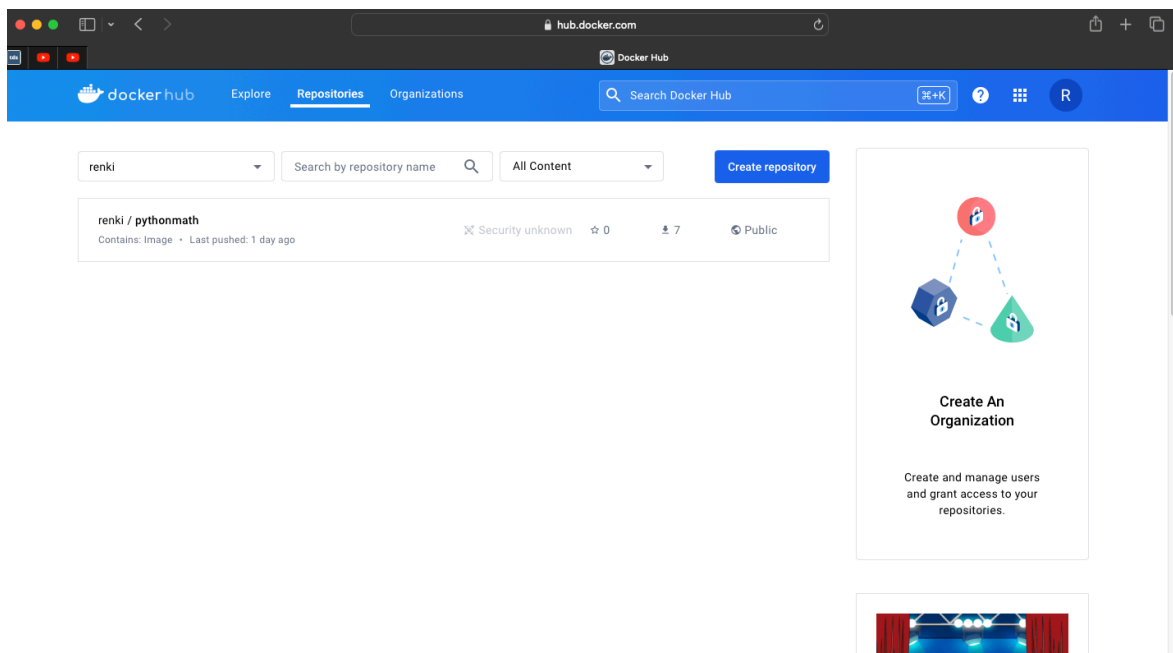


Slika 9: Secrets vrednosti repozitorija



Slika 10: Skripta zapakiraj.sh

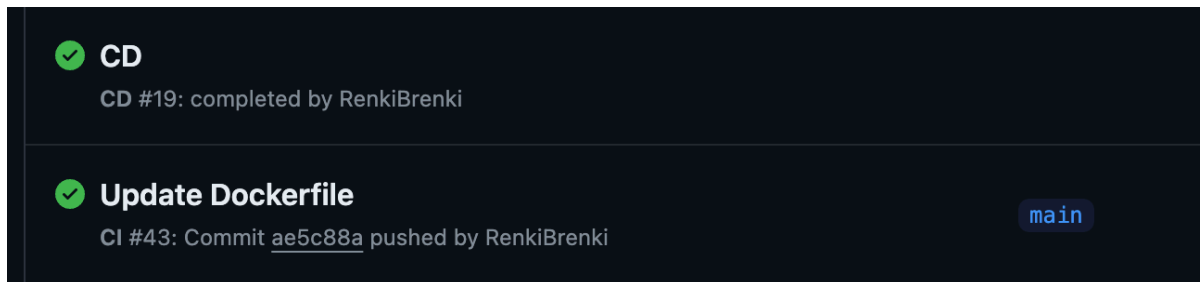
Skripta zapakiraj.sh ustvari Docker image in ga objavi na DockerHub repozitorij z imenom renki/pythonmath z verzijo latest.



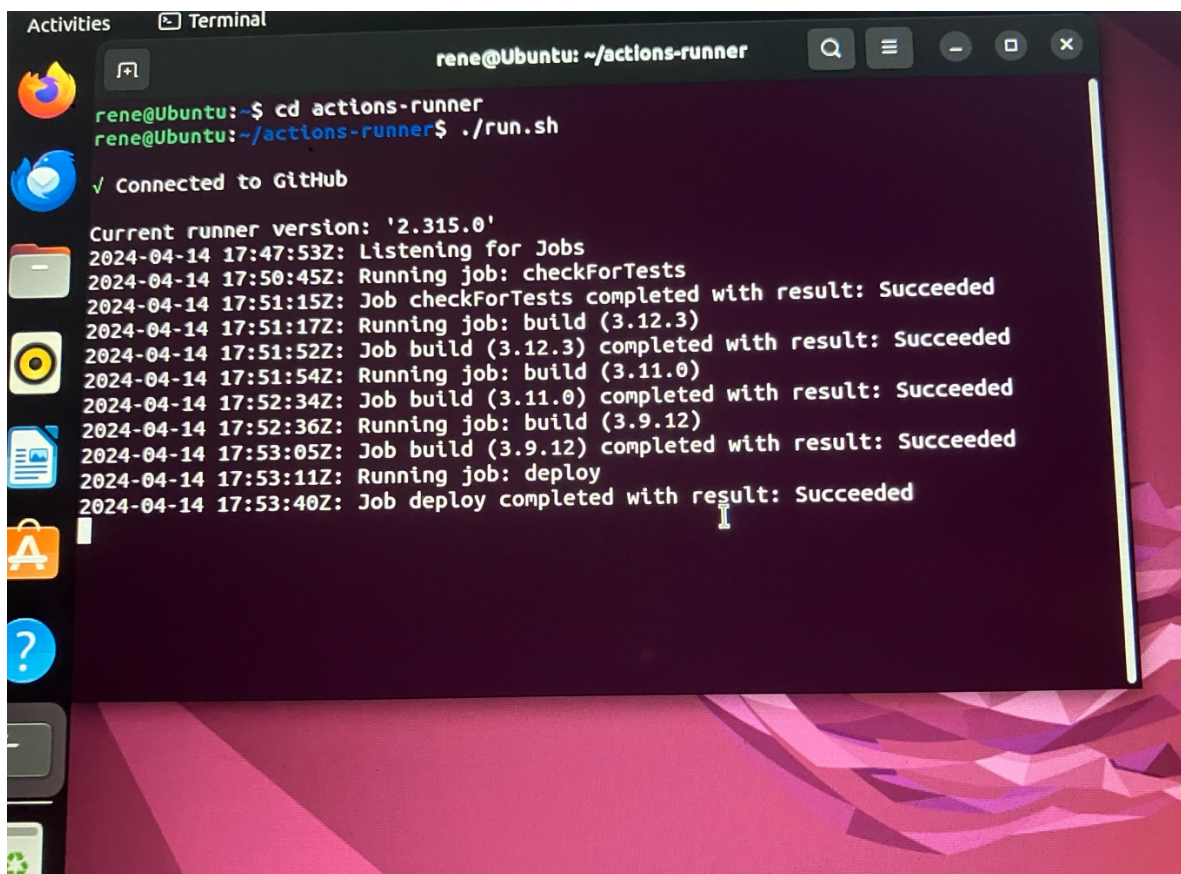
Slika 11: Docker repozitorij po izvedbi skripte

### 3 Stanje po spremembi na Github repozitoriju

Po izvedbi push ukaza na glavno vejo se izvedeta obe skripti. Prvo se izvede CI in nato CD.



Slika 12: Izvedba skript po ukazu push



Slika 14: Uspešen zagon na linux x64 runnnerju

```
✓ Connected to GitHub

Current runner version: '2.315.0'
2024-04-14 16:42:55Z: Listening for Jobs
2024-04-14 17:31:41Z: Running job: checkForTests
2024-04-14 17:31:54Z: Job checkForTests completed with result: Succeeded
2024-04-14 17:31:55Z: Running job: build (3.9.12)
2024-04-14 17:32:08Z: Job build (3.9.12) completed with result: Failed
2024-04-14 17:32:10Z: Running job: build (3.12.3)
2024-04-14 17:32:16Z: Job build (3.12.3) completed with result: Canceled
2024-04-14 17:32:21Z: Running job: deploy
2024-04-14 17:32:33Z: Job deploy completed with result: Failed
2024-04-14 17:42:14Z: Runner connect error: The HTTP request timed out after 00:01:00. Retrying until reconnecte
```

*Slika 15: Neuspešna izvedba skripte na linux arm64 runnerju*