# CI/CD

Poročilo – Naloga 3

Sistemska administracija

Andraž Krajnc
R-IT 2

# Kazalo



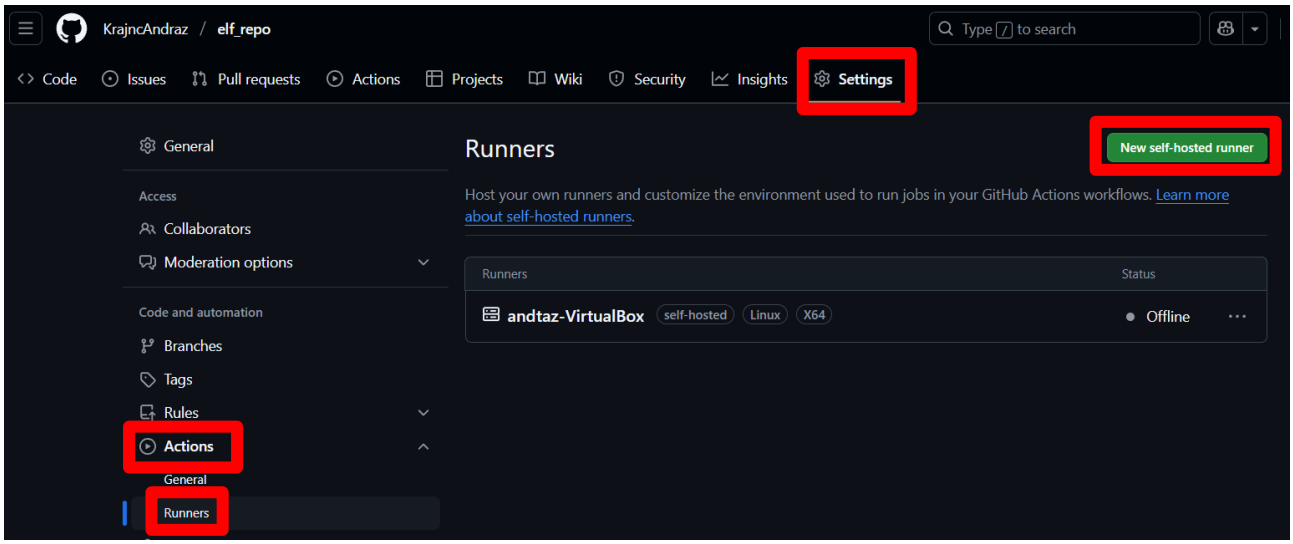
# Kazalo slik

# 1. Viri

Repozitorij naloge je dostopen na https://github.com/KrajncAndraz/elf_repo.
DockerHub repozitorij je dostopen na https://hub.docker.com/r/ak1feri/feri_sa.

# 2. Priprava sistema

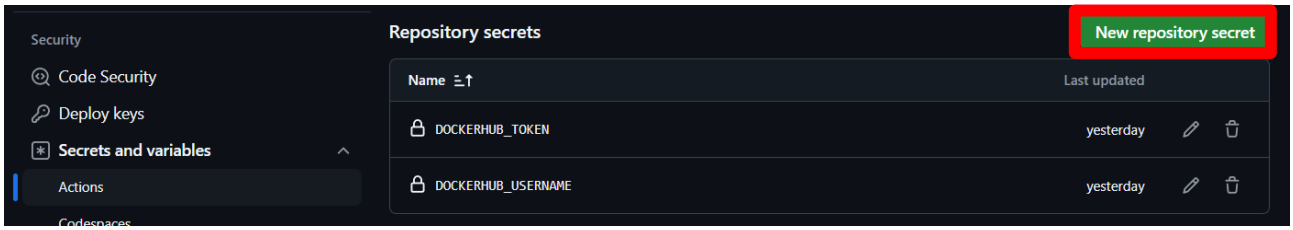
Odpremo git repozitorij in se pomaknemo skozi Settings -> Actions -> Runners -> New self-hosted runner in ustvarimo nov runner, da se nam ne zaračuna stroškov uporabe git-ovega runner-ja.




*Slika 2.1: Pot do kreacije novega runner-ja.*

Ob kliku na gumb se odprejo navodila, za vzpostavitev runnerja glede na operacijski sistem. Samo kopiramo in prilepimo komande v ukazno vrstico.

Nato ustvarimo DockerHub profil in repozitorij, če ga še nimamo. Za tem se pomaknemo malo nižje v stolpčnem meniju nastavitev na sliki 2.1, v zavihek Secrets and variables -> Actions. Tu zapišemo vpisne podatke za DockerHub, ki jih bomo kasneje potrebovali za nalaganje Docker slik v repozitorij.




*Slika 2.2: Pot do skrivnosti (vpisni podatki)*

# 2. Implementacija

V svojem projektu ustvarimo mape .github/workflows in test. Pozorni moramo biti na to, da so skrite mape prikazane saj mape, ki se začnejo s ».« se vsaj v Linux-u skrijejo.
V mapo »workflows« implementiramo akcije oz. YAML datoteke, ki se bodo izvršile ob npr. push na main. V mapo »test« pa implementiramo teste, ki jih želimo avtomatsko zagnati ob določenih scenarijih.

## 2.1 Implementacija testa

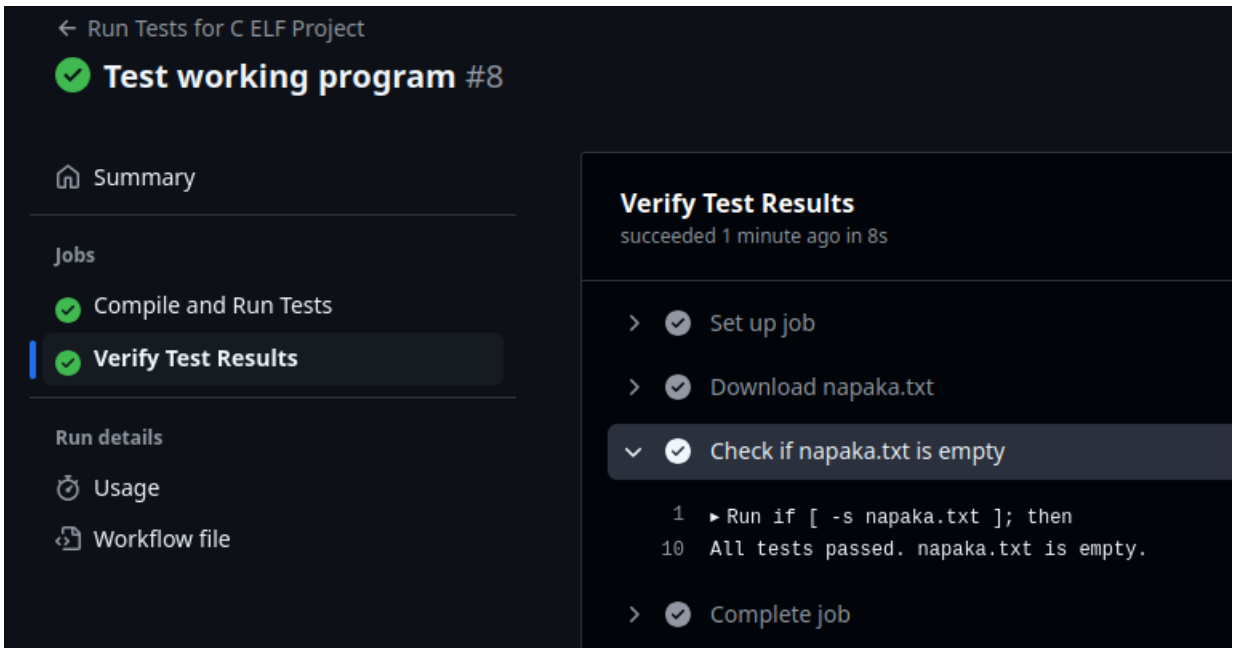
V našem primeru uporabljamo Linux kot izvajalno okolje in nalogo ELF iz predmeta Namenska programska oprema. Zato je ta skripta napisana v bash (.sh) in imenovana test_elf_changer.sh.
V programu elf_changer so bile 3 možnosti oz. 3 glavne funkcionalnosti programa, ki že imajo implementirano rokovanje z napakami. V skripti za testiranje zato uporabljamo samo preverjanje izhoda in iščemo znane izpise napak z ukazom grep. Če je napaka najdena, se jo zapiše v novo datoteko imenovano napaka.txt.

## 2.2 Implementacija akcij

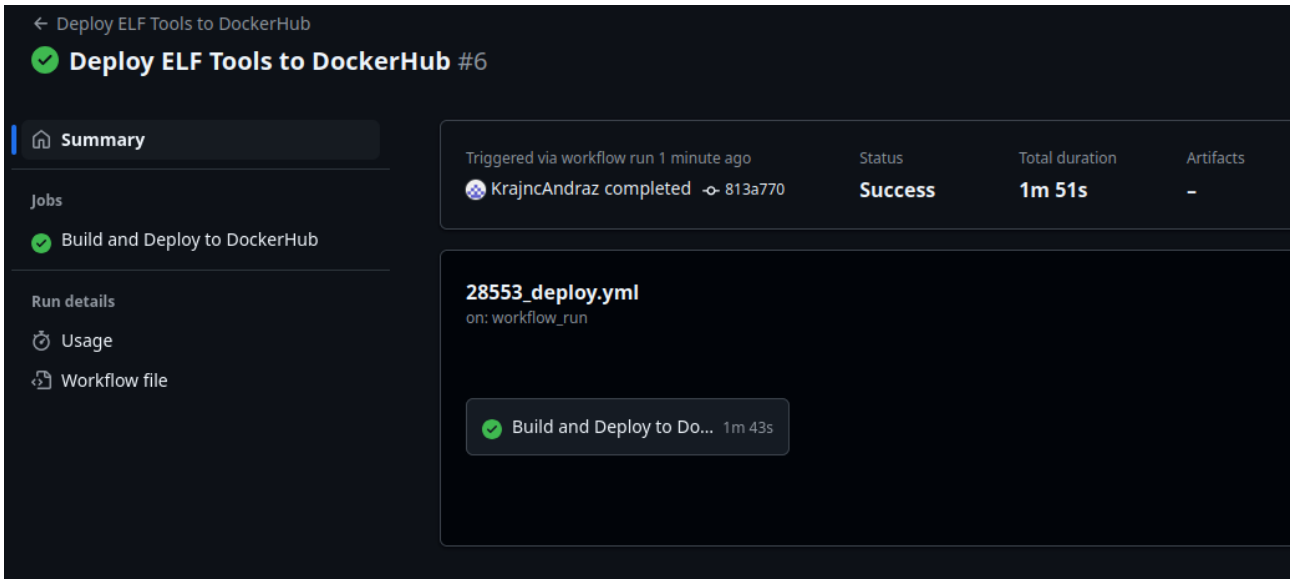
Implementacija testne akcije je sestavljena iz pogoja, da se zažene ob potisku na main vejo in vsebuja 2 posla. Prvi poišče in zažene teste ter shrani datoteko napaka.txt kot artefakt. Drugi pa preveri, če so bili uspešni tako da preveri, če je napaka.txt prazna.
Implementacija publikacijske akcije vsebuje pogoj, da se zažene le po uspešni izvršitvi prejšnje akcije. Če je pogoj zagotovljen, projekt spakira v docker sliko in jo s pomočjo skrivnosti, ki smo jih prej nastavili, naloži na naš DockerHub repozitorij.

# 3. Izvrševanje akcij

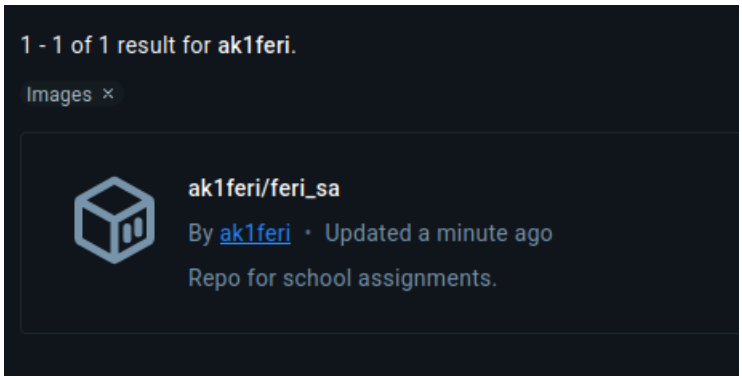
Ko se koda pravilno izvrši, je to razvidno iz slik 3.1, 3.2 in 3.3. Ko pa se ne pa je to prikazano, na sliki 3.4.



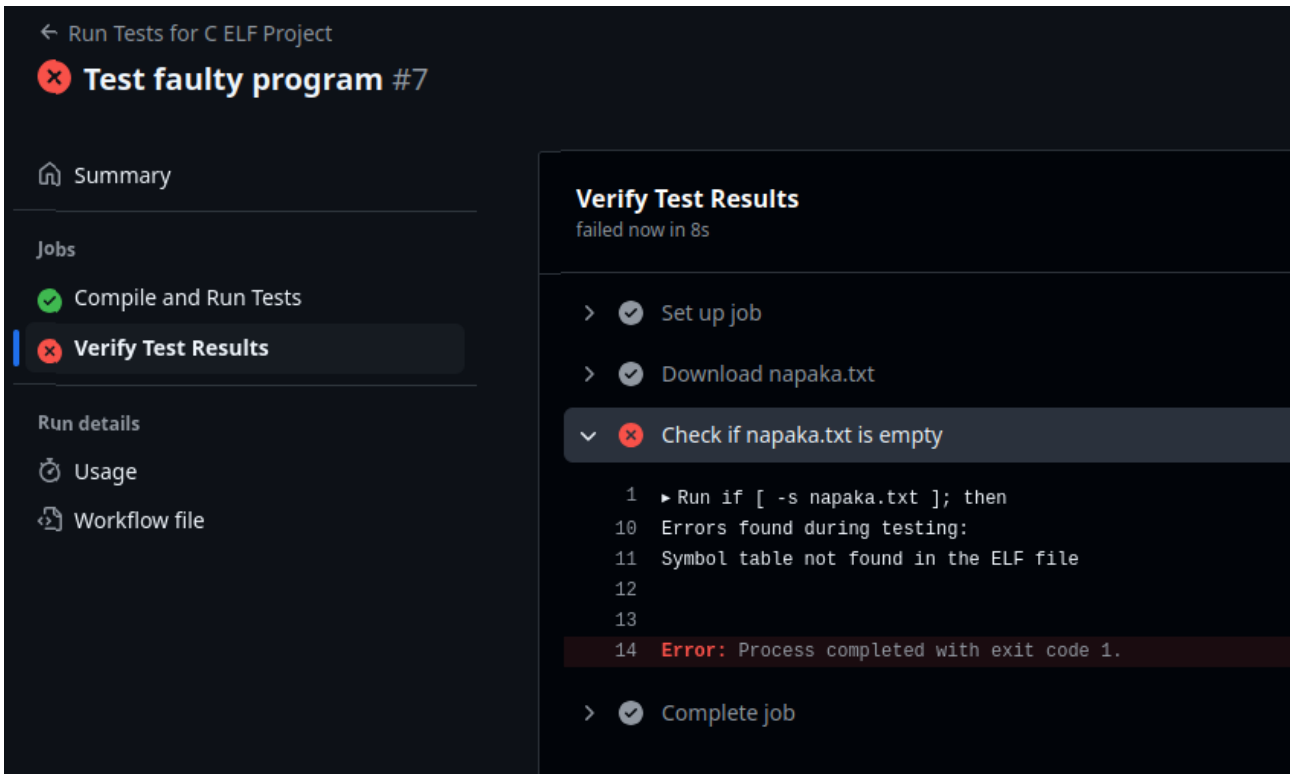

*Slika 3.1: Uspešna prva akcija*




*Slika 3.2: Uspešna druga akcija*




*Slika 3.3: Uspešno naložena slika na DockerHub*




*Slika 3.4: Neuspešno izvršena akcija*