Sigurnost operacijskih sustava i aplikacija
Tin Plavec

Laboratorijska vježba 3
Izvještaj

Promjene u kodu

Metoda `perform_division` baca iznimku ukoliko je drugi element 0, umjesto da vraća NaN. Dodao sam jednostavno if grananje kako bi program vratio korektnu vrijednost.

```
        if self.b == 0:
            return float("nan")
        return self.a / self.b
```

Problem je i u provjeravanju lozinke, čija je vrijednost direktno zapisana u izvornom kodu aplikacije. To je sada riješeno pomoću varijable okruženja:

```
root_password = os.environ.get("ROOT_PASSWORD")
```

Ukoliko varijabla ne postoji ili je kraća od 6 znakova, program prekida s radom uz povratnu vrijednost 2.
Dodana je nova funkcija `eval_expression` koja zamjenjuje nesigurnu `eval` funkciju. Nova funkcija nešto je složenija, a koristi `ast.parse` što validira unos korisnika.

Unit testovi

Definirao sam 3 unit testa:

- Prvi provjerava da funkcija `perform_division` vraća NaN za argumente 7 i 0
- Drugi provjerava ispravnost funkcije `login_success`. Na ulaz stalno predaje 3.4, a očekuje da će funkcija prvo vratiti 1.0, a zatim 3.4
- Treći provjerava da funkcija `login_success` baca iznimku ValueError kada na ulaz dobije znak `a`

Bandit testovi

Bandit je pronašao 2 problema na originalnom kodu.
Problem *niskog* značaja je taj što je lozinka *hardcoded*, tj. napisana je direktno u izvornom kodu.
Problem *srednjeg* značaja je što je korištena funkcija `eval` koja je potencijalno nesigurna. Preporuka je koristiti funkciju `ast.literal_eval` koja je sigurnija.

Pylint testovi

Pylint (https://pypi.org/project/pylint/) je statički analizator Python koda koji ukazuje na razne greške, upozorenja te sintaksni stil. Za potrebe ove vježbe preporuke za sintaksni stil su onemogućene. U originalnom kodu postoji linija `import os`, no `os` se nigdje ne koristi, za što ovaj alat šalje upozorenje. Možemo dodati i `import re`, što će Pylint također detektirati.

Jenkins cjevovod

Jenkins je instaliran pomoću `apt` alata direktno na računalu, s preporučenim ekstenzijama. Cjevovod se pokreće manualno pritiskom na `Build Now`.

Definicija cjevovoda

Cjevovod je definiran na samom GitHub repozitoriju pomoću *Jenkinsfile*-a. Unutar Jenkins-a potrebno je namjestiti da se konguracija cjevovoda povlači s repozitorija. Nakon što stvorimo novi cjevovod, potrebno je otići na `Configure` pa `Pipeline` te za definiciju odabrati `Pipeline script from SCM`. Treba unijeti URL na repozitorij te ključ za pristup. Izgenerirao sam novi par privatnog i javnog ključa, javni dodao kao pristup za GitHub repozitorij, a privatni kao *credential* u Jenkins. *Credential* je vrste `SSH Username with private key`.

Povlačenje repozitorija

Za povlačenje repozitorija koristi se Git ekstenzija te *credential* koji je stvoren u prošlom koraku. Ova faza se u cjevovodu zove `Fetch Git repo`

Izgradnja Docker slike za testiranje i testiranje

Za testiranje koristi se Docker kontejner. Da bi Jenkins mogao graditi Docker kontejnere, Docker mora biti instaliran na mašini te korisnik Jenkins mora biti u Docker grupi kako bi imao pristup:

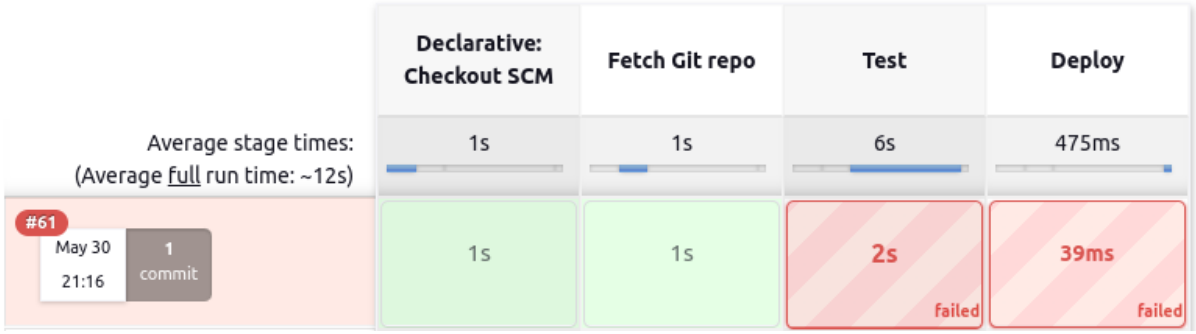
```
sudo usermod -aG docker jenkins
```

Dockerfile postavlja `ROOT_PASSWORD`, instalira potrebne pakete za testiranje, dodaje datoteke aplikacije, te pokreće skriptu `run.sh`. Ta skripta pak pokreće određen test s obzirom na varijablu `PERFORM_TEST`, ili pokreće samu aplikaciju ako ta varijabla nije definirana. Skripta je napravljena kako bi se isti *Dockerfile* mogao koristiti za testiranje i za produkciju.
U Jenkins cjevovodu faza `Test` provodi sva 3 testa nakon što izgradi kontejner. Valja primijetiti mapiranje *volume*-a za logove. Lokalni direktorij `/home/jenkins/cicd/logs` potrebno je prije kreirati ručno pomoću korisnika `jenkins` kako bi Jenkins mogao pristupiti tom direktoriju na mašini. Primijetimo da testovi, unutar skripte `run.sh` koji se izvršava u kontejneru, zapisuju svoj izlaz u datoteku u direktorij `logs` (koji je pak mapiran na lokalni direktorij na mašini). Ovdje se i dodaje vremenska oznaka na ime datoteke.
Alternativno, možemo izbrisati ovo preusmjeravanje izlaza testova kako bismo logove vidjeli na Jenkins sučelju. Potrebno je iz linije skripte maknuti oznaku preusmjeravanja `&>` te sve iza nje.

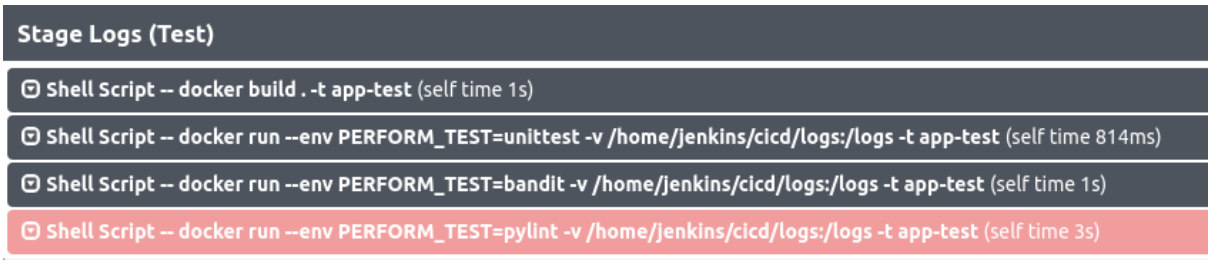
Izgradnja Docker slike za produkciju

Faza `Deploy` gradi produkcijsku sliku koja je identična onoj za testiranje. Ova slika ima Docker oznaku `app-production`. Ova faza izvodi se samo ako su sve prijašnje faze uspješno završile.

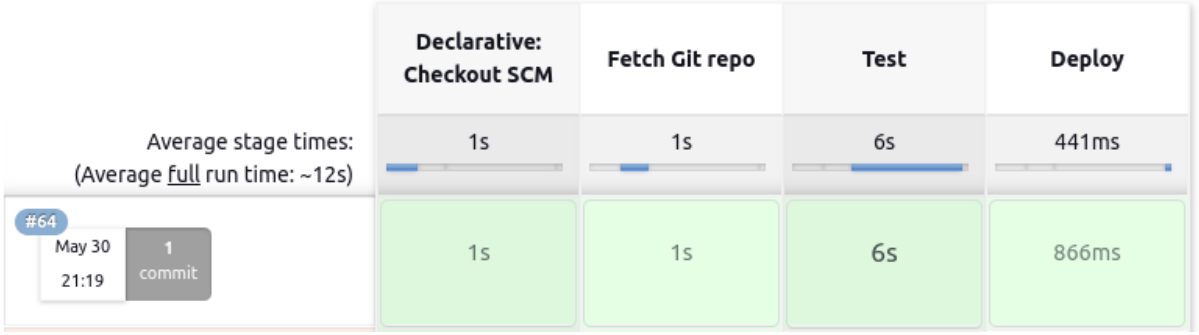
Slučaj kada jedan od testova ne uspije:



Slučaj kada jedino Pylint javi pogrešku:



Slučaj kada svi testovi prođu te produkcijski kontejner bude uspješno izgrađen:



Pokretanje produkcije

Aplikaciju možemo pokrenuti pomoću Docker-a, uz postavljanje lozinke za korisnika `root`:

```
docker run --env ROOT_PASSWORD=sosa23 -it -t app-production
```

Datoteke s logovima

Logovi su priloženi u direktoriju `primjeri_logova`. Naziv datoteke sastoji se od vremenske oznake te naziva testa.
Za unit testove jedan log pokazuje kako je test za `perform_division` izazvao iznimku, a drugi log pokazuje kako su sva 3 testa uspješno izvršena.
Za Bandit testove jedan log pokazuje kako test nije prošao jer u kodu koristimo potencijalno nesigurnu funkciju `eval` te nas savjetuje da koristimo paket `ast`. Drugi log pokazuje kako je Bandit uspješno izvršio testove.
Za Pylint testove jedan log upozorava nas da u kodu imamo nekorišteni *import*. Drugi log pokazuje kako test prolazi kada uklonimo taj *import*.