CI / CD PIPELINE CU JENKINS

Eusebiu Rizescu[1]

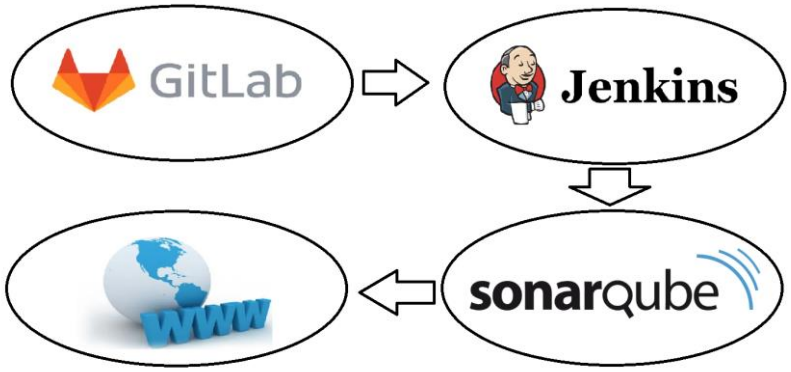
În prezent, idea de CI/CD este intalnita in majoritatea companiilor care dezvolda software. Atat CI cat și CD compun coloana vertebrală a mediului DevOps. Acest mod de lucru umple spatiul dintre echipele de dezvoltare și echipele de operations, automatizând impachetarea aplicatiilor, testarea și deployment-ul acestora. Scopul acestui document este continuarea activitatii din semestrul trecut prin proiectarea si construirea unui sistem de CI/CD propriu, consturit pe infrastructura proprie (on-premise)

Nowadays, the idea of CI/CD is seen in the majority of companies that makes software. Both CI and CD form the back bone of the modern DevOps environment. DevOps fills the gap between development and operations teams by automating the building of the applications, testing, and deployment of them. The purpose of this document is to continue the activity of the last semester by designing and building its own CI / CD system, built on its own infrastructure (on-premise)

Cuvinte cheie: CI/CD, GitLab, Jenkins, SonarQube

1. Introducere

Ca o scurta recapitulare a cercetarii din semestrul trecut, scopul final al acestui document este acela de a construi si folosi un sistem on-premise de CI/CD, cu ajutorul a diverse tool-uri gratuite. Semestrul trecut am descris termenii si conceptele folosite in cadrul CI/CD si anume: ce este CI, ce este CD, ce rol au acestea, descrierea unui pipeline, comparatie intre mai multe tool-uri de Source Code Management, comparatie intre mai multe tool-uri de build si deploy, descriere tool-uri de testare a codului sursa. In acest semestru am ales tool-urile care vor compune sistemul de pipeline care va fi creat, si testarea compatilitatii intre acestea. In prezent traseul pe care codul sursa il parcurge pana la deploy este acesta:



Figură 1 - Diagrama Pipeline

[1]Facultatea de Automatică şi Calculatoare, Universitatea POLITEHNICA din Bucureşti, România, e-mail: rizescueusebiu@gmail.com

2. Mediu de dezvoltare

In cadrul acestei etape, mediul de dezvoltare a fost o masina virtuala, virtualizata cu ajutorul VMWare. Ca sistem de operare aceasta a avut Debian 10 Buster, fara modificari majore. De ce Debian? Pentru ca sunt cel mai familiarizat cu acest OS, este foarte popular, si cel mai importnat, este mai lightwight decat Ubuntu. Din firewall (iptables), a trebuit sa accept trafic pe anumite porturi, si anume:

Aplicatie	Port (TCP)
Gitlab	10100
Jenkins	10200
SonarQube	10300
Aplicatie	10400

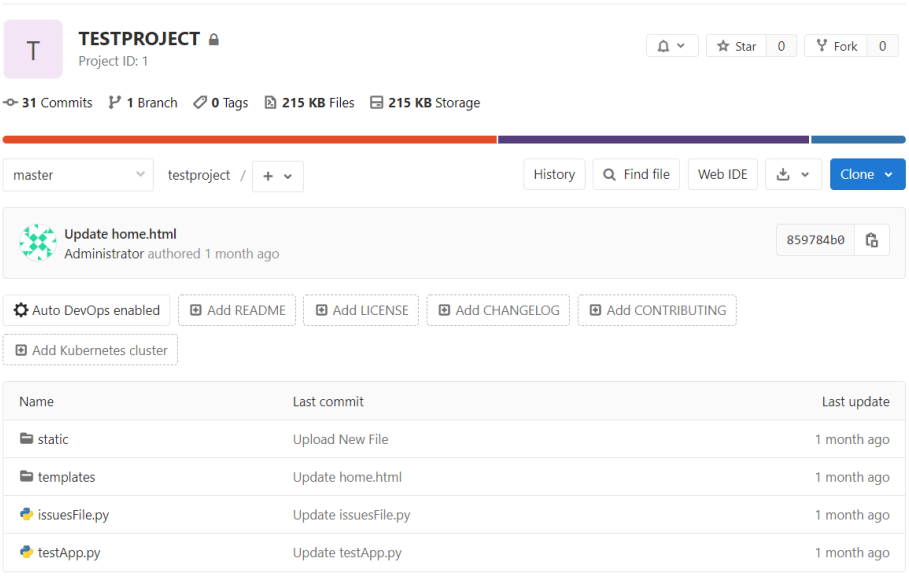
Figură 2 - Porturi folosite

Aplicatiile au fost instalate direct pe OS, nu in containere. Avand in vedere ca toate aplicatiile au fost instalate pe aceeasi masina, lucru care nu este de dorit in productie, s-au creat diverse servicii si aplicatii, dependinte are aplicatiilor principale (Postgres, Node Exporter, Prometheus, Grafana). In semestrele viitoare, voi rezolva acest neajuns, prin instalarea aplicatiilor in containere, in Docker, si totodata, voi configura reguli de healthcheck si automatic failover pentru ca aplicatiile sa fie highly-available. Mai multe detalii in sectiunea: „Semestrele urmatoare". Pentru a arata intreg procesul de pipeline, am facut un video care poate fi gasit la: (https://bit.ly/2zsulgX). Aici se poate observa ca traficul nu este criptat (se foloseste HTTP), deoarece acest pipeline a fost pentru a verifica si testa compatibilitatea intre tool-uri.

2. GitLab

GitLab este un instrument de DevOps, web-based care oferă gestionarea depozitelor Git, recenzii de coduri, urmărirea problemelor, fluxuri de activități și wikis-uri. Gitlab vine in doua versiuni Gitlab-EE (platita) si GitLab-CE (gratuita). GitLab se poate folisi atat in cloud cat si on-premise.

Versiunea de GitLab instalata este, Community Edition – 12.10.0. GitLab este o aplicatie foarte complexa, cu multe feature-uri, si astfel ca la pornire consuma foarte mult CPU.



Figură 3 - GitLab

3. Jenkins

Jenkins este un server de automatizare gratuit și open source. Acesta ajută la automatizarea părților de dezvoltare software legate de construire, testare și implementare, facilitând integrarea continuă și livrarea continuă. Este un sistem bazat pe server care rulează în containere servlet, cum ar fi Apache Tomcat.

Versiunea de Jenkins instalata este 2.222.1.

Cu ajutorul Jenkins, atunci cand un nou commit este observat in GitLab, se va porni in plan de build. In pipeline-ul exemlu, primul pas este pornirea scanarii codului sursa de catre Sonar, cu ajutorul agentului de Sonar numit sonar-scanner

```
echo "Running Sonar"
sonar-scanner -Dsonar.projectKey=TESTPRJ  -Dsonar.projectName=Test-Project
```

Figură 4 - Scanarea codului

Urmatorul pas este deploy-ul aplicatiei, care in exemxplul de fata, consta in oprirea serviciului web, copierea noului cod sursa si repornirea serviciului:

```
echo "Building App"
systemctl stop flaskapp
rm -rf /data/flaskApp/*
cp -R * /data/flaskApp
systemctl start flaskapp
```

Figură 5 - Deploy aplicatie

În cele mai multe cazuri, o executie a pipeline-ului este declanșată de o modificare cod sursă din repository. O modificare a codului declansaeza o notificare in Jenkins, care rulează un pipeline predefinit anterior. Concret, in exemplul nostru, cand este comis un nou cod, GitLab apaleaza un Webhook catre Jenkins, anuntandu-l pe acesta ca exista cod sursa nou. Apoi Jenkins ruleaza planul de build aferent.

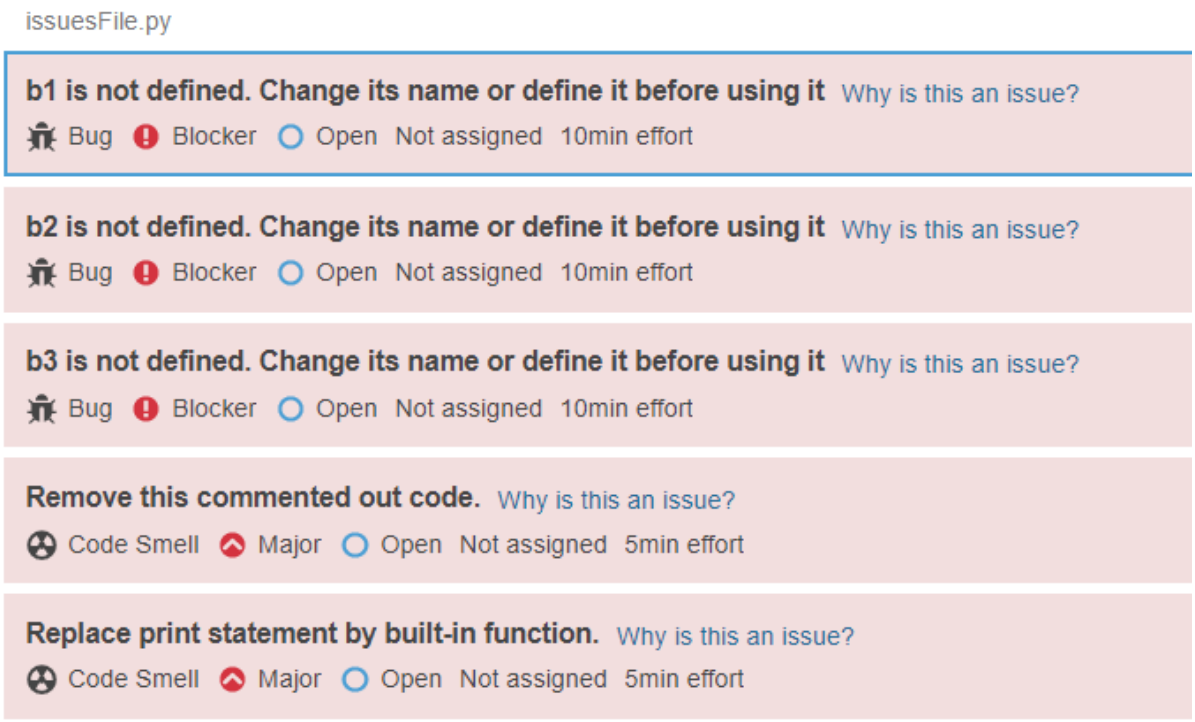
```
1)  Ambele: Creat project
2)  GitLab: Pus cheie ssh de read pe repository
3)  Jenkins: Pus pluginul de Gitlab
4)  Jenkins: In setari generale nu am facut nimic la rubrica GitLab.
5)  Jenkins: Creat Token: Profile (dreapta sus) / Configure / Add new token
6)  Jenkins: Manage Jenkins / Configure System / Git plugin pus un user si mail la misto
7)  Jenkins: Am pus cumva cheia privata echivalenta cheii publice din GitLab in Credentials
8)  Jenkins: Apoi in planul de build, am pus la Source Code Management / Git
        git@127.0.0.1:root/testproject.git  si am selectat cheia de acolo
9)  Jenkins: La Build Triggers am selectat "Build when a change is pushed to Gitlab"
10) In GitLab am pus Webhooks cand se da push la:
http://root:11f98056133324d959d305aa2ad84f5113@192.168.79.129:10200/project/TestProjectJenkins
```

Figură 6 - Integrare GitLab cu Jenkins

4. SonarQube

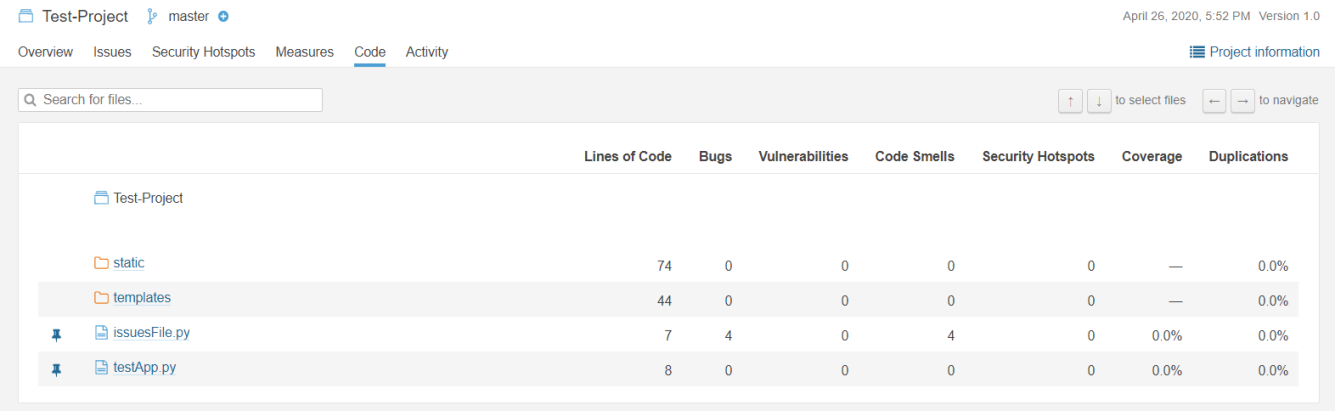
SonarQube este instrumentul principal pentru inspecția continuă a calității codului și securității bazelor de cod și îndrumarea echipelor de dezvoltare în timpul evaluărilor codului. SonarQube oferă rapoarte despre secvente de cod reduplicat, complexitate cod, standarde de coding, teste unitare, acoperire cod, comentarii, bug-uri și vulnerabilități de securitate.

Versiunea de SonarQube instalata este: SonarQube Comunity Edition – 8.22



Figură 7 - Sonar exemplu erori

Pentru a face analiza pe cod sursa, SonarQube foloseste un agent, numit sonar-scanner, care parseaza codul, si produce erori. Astfel, load-ul nu este pe server, ci pe agentul unde se ruleaza scanarea. Serverul tine istoria problemelor din cod, modificarea severitatiloe eventualor problemelor, permisiuni si multe altele.



Figură 8 - Alte informatii oferite de SonarQube

5. Aplicatie Exemplu

Aplicatia pe care am folosit-o pentru a testa pipeline-ul este o simpla aplicatie in Flask (Python framework). Este un template nemodificat, nimic spectaculos. Contributia mea a fost modelul de deploy al acestei aplicatii, cu ajutorul unui serviciu (/etc/init.d/flaskapp). Mai jos este doar o secventa din acest serviciu, care este pornirea serviciului:

```
case $1 in
  start)
        echo -n "$(date) -- Starting flask app: "
        flaskProcess=$(ps -eopid,cmd | grep "python3 testApp.py" | \
              grep -v "grep" | sed "s/^ //g" | awk '{$1=$1;print}' | cut -d" " -f1)
        if [ ! -z "$flaskProcess" ]
        then
                echo "$(date) -- There are other flask processes running"
                exit 1
        fi
        cd /data/flaskApp
        nohup python3 testApp.py >> /data/flaskApp/flaskapp.log 2>&1 &
        echo "Flask started OK"
  ;;
```

Figură 9 - Serviciu pornire aplicatie

6. Semestrele urmatoare

Avand in vedere ca tool-urile sunt suportate unul de celalalt, urmatorul pas este deploy-ul acestora in Docker. Se vor urmari aspecte de monitorizare, healthcheck, high availibilty, automatic failover, scalare automata a numarului de agenti de Jenkins. De asemenea se va urmari si trecerea la HTTPS pentru toate cele trei servicii. Mediul in care aceste servicii vor fi deployate trebuie investigat: masini virtuale locale, cloud sau cluster de Raspberry Pi

7. Concluzii

Pipeline-ul este complet, la fiecare modificare de cod se va trigger-ui scanarea de cod si apoi dploy-ul. Totusi, toate aceste servicii ruleaza pe o singura masina, lucru care nu este de preferat intr-un mediu de productie. Astfel ca se va urmari configurarea tool-urilor in containere de Docker, pe mai multe masini.

BIBLIOGRAFIE

[1] https://about.gitlab.com/company/
[2] https://www.vultr.com/docs/how-to-install-gitlab-community-edition-ce-11-x-on-debian-9
[3] https://www.jenkins.io/doc/
[4] https://docs.sonarqube.org/latest/
[5] https://markdotto.com/projects/