FCIM, UTM, TIDPP

Laboratorul 3

Dependency Management, Build automation, Unit Test, Continuous Integration

Înainte de a începe elaborarea laboratorului fiecare student trebuie să își găsească o aplicație WEB de pe Github sau oricare alt hosting de repozitorii. Aplicația trebuie să posede un funcțional bine definit, nu se acceptă aplicații foarte simple, aplicații ce nu utilizează baza de date și aplicații fără backend/frontend.

Fiecare student trebuie să cloneze aplicația găsită, să creeze un repozitoriu pe Github și să încarce codul sursă aplicației pe repozitoriul creat.

Pentru aplicația aleasă să se scrie **3 teste unitare** pentru backend. Testele unitare nu trebuie să fie complexe. Pentru teste creați un branch "**staging**" aparte din master și odată ce toate testele vor trece cu succes să actualizați branch-ul master cu staging.

Jenkins

În Jenkins să se creeze un pipeline care să conțină următoarele:

1. Să se extragă ultima versiune a proiectului din repozitoriul aflat pe Github branch-ul master

2. Să se instaleze dependențele în mod automat și să se execute compilarea codului

3. Să se pornească testele unitare pentru backend, rezultatele să fie salvate într-un format Junit XML. Fișierele generate să fie procesate în pipeline de către JUnit indiferent de starea build-ului.

4. Să se seteze un parametru boolean **CLEAN_WORKSPACE** unde în cazul că este adevărat va șterge mapa creată de Jenkins pentru build-ul curent.

5. Să se seteze două variabile de mediu **ON_SUCCESS_SEND_EMAIL** și **ON_FAILURE_SEND_EMAIL** cu conținutul "true".

6. Sa se declare o etapă "Testing frontend" după etapa de testare backend și să se seteze un parametru boolean **TESTING_FRONTEND** implicit false. Pașii acestei etape să fie executați doar dacă parametrul anterior este adevărat. În steps nu trebuie să se execute testele pentru frontend, să aveți doar un singur echo ce afișează valoarea parametrului TESTING_FRONTEND.

Să fie setată declanșarea execuției pipeline-ului din Jenkins în mod automat în fiecare zi odată la două ore.

În dependență de variabilele de mediu declarate în punctul 5 să se expedieze un mesaj pe mail în mod automat cu un text dacă totul a avut loc cu succes sau nu. În mesaj să se includă datele variabilelor de mediu **JOB_NAME**, **BUILD_NUMBER** și **BUILD_URL.**

Important:

- La susținerea laboratorului sunt admiși doar stundeții la care toate etapele din pipeline s-au executat cu succes.
- Manager de pachete(dependențe) pentru aplicații **Python**, **Java**, **JavaScript**, **PHP**: https://blog.idrsolutions.com/2018/07/what-is-a-package-manager-and-why-should-you-use-one/
- Manager de pachete(dependențe) pentru aplicații **.NET**: https://visualstudiomagazine.com/articles/2014/03/01/6-top-net-package-and-dependency-management-tools.aspx
- Exemplu de pipeline Jenkins pentru o aplicație web **Python/Django**: https://github.com/bdmoleg/TIDPP-FCIM-CI-CD/blob/master/Jenkinsfile
- Integrarea continuă și limbajele de programare interpretabile: https://softwareengineering.stackexchange.com/questions/310559/how-can-ci-be-used-for-interpreted-languages
- Cum se integrează MSBuild și NuGet cu Jenkins: https://www.codeproject.com/Articles/878203/Integrate-Jenkins-with-MSBuild-and-NuGet

Întrebări la apărarea laboratorului:

- De ce avem nevoie de Jenkins ?
- Ce este un Jenkins pipeline ?
- Care sunt tipurile de pipeline Jenkins ?
- Care sunt modalitățile de a declanșa un pipeline Jenkins ?
- Cum se realizează integrarea continuă cu Jenkins?
- Pentru ce este nevoie de variabile de mediu în Jenkins ?
- Numiți 5 variabile de mediu implicit disponibile în Jenkins
- Ce se înțelege prin integrare continuă (Continuous Integration) ?
- Ce se înțelege prin livrare continuă (Continuous Delivery) ?
- Ce se înțelege prin plasare continuă (Continuous Deployment) ?
- Care sunt diferențele dintre integrarea continuă, livrarea continuă și plasare continuă ?
- Care sunt factorii de succes pentru integrarea continuă ?
- Care este rolul serverului de integrare continuă ?
- Cum DevOps influențează dezvoltarea de software ?
- Numiți 5 instrumente CI/CD
- Care este ciclul de viață al unui produs program ?
- Ce reprezintă și care este rolul testării unitare ?
- Numiți 5 cadre de testare bazate pe xUnit
- Care este rolul aserțiunilor în testarea automată ?
- Cum Jenkins identifică schimbarea versiunii în repozitoriu ?

Linkuri utile pentru efectuarea laboratorului:

- https://www.fullstack.cafe/blog/15-continuous-integration-interview-questions-in-2018
- https://www.interviewbit.com/jenkins-interview-questions/

- https://e.printstacktrace.blog/jenkins-pipeline-environment-variables-the-definitive-guide/
- https://e.printstacktrace.blog/5-common-jenkins-pipeline-mistakes/
- https://www.jenkins.io/doc/book/pipeline/
- https://www.oreilly.com/library/view/jenkins-the-definitive/9781449311155/ch04.html
- https://medium.com/southworks/creating-a-jenkins-pipeline-for-a-net-core-application-937a2165b073
- https://www.devopsuniversity.org/jenkins-setup-guide/
- https://www.fosstechnix.com/build-java-project-using-maven-in-jenkins-pipeline/
- https://en.wikipedia.org/wiki/List_of_build_automation_software
- https://www.todaysoftmag.ro/article/2268/continuous-delivery
- https://dzone.com/articles/build-tools-java-survey-results
- https://docs.microsoft.com/en-us/dotnet/core/testing/unit-testing-best-practices
- https://www.todaysoftmag.ro/article/377/din-uneltele-artizanului-software-unit-testing
- https://en.wikipedia.org/wiki/List_of_unit_testing_frameworks
- https://martinfowler.com/articles/mocksArentStubs.html
- https://martinfowler.com/bliki/TestPyramid.html
- https://martinfowler.com/bliki/Xunit.html
- https://www.simform.com/blog/test-coverage/
- https://junit.org/junit5/docs/5.0.1/api/org/junit/jupiter/api/Assertions.html
- https://docs.python.org/3/library/unittest.html
- https://plugins.jenkins.io/junit/
- https://harness.io/blog/software-development-life-cycle/
- https://en.wikipedia.org/wiki/Program_lifecycle_phase