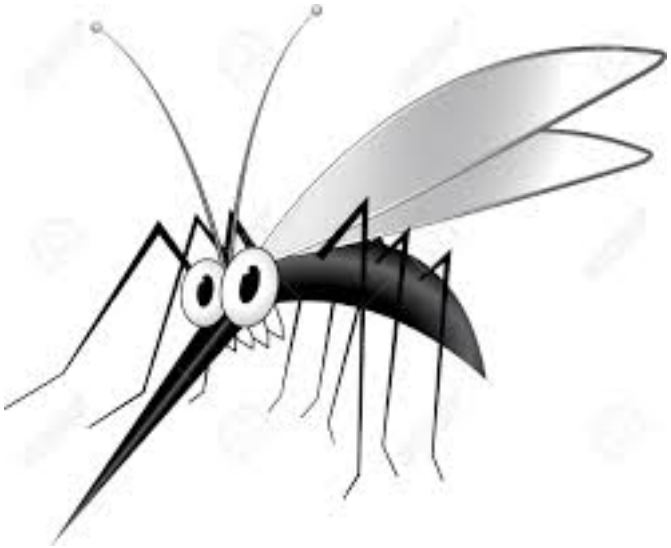

תרגיל כיתה\שיעורי בית- קורס דב-אופס

כי כל יתוש צריך להגיע ליעדו בזמן

נושאים מתורגלים: 2
מבוא 2
חקר מקרה 2
איך ניגשים לפיתרון התרגיל 3
מבוא: 3
ניתוח שחקנים, רכיבים,שלבי ביצוע, אילוצים ודרישות 3
ניתוח תהליך רצוי 4
איך לדעתינו יראה התוצר שלנו 4
עבודה 5
Appendixes: 6
Jenkins File Example: 6
SonarQube 10
Intro: 11
Main Components: 11
Work Flow: 11
Installations: 11

נושאים מתורגלים:

- Linux Operations (user management, services management)
- Jenkins:
  - Jenkins pipeline:
    - Simple Jenkins pipelines
    - scripting in pipeline pipeline
  - Jenkins plugins
  - Jenkins tool management
- Build Frameworks:
  - Maven
  - Docker
- Web server management:
  - Tomcat
- Central management\provisioning tools:
  - Vagrant
  - Ansible
- Secure Devops:
  - Dependency check (OWASP)
  - Source code check (Sonarqube)

מבוא

לתלמיד שלום,
הינך יועץ בתחום ה DevOps, בחברת "אלפי-בלי סודות", ונקראת לפתע ללקוח, אשר נתקל במספר רב של אתגרים. עליך לעמוד על קשייו, צרכיו, לנתחם ולתת מענה הולם. בעת מתן המענה או הפיתרון, עליך לנמק בפרוטרוט מדוע בחרת לממש את אשר בחרת, ומדוע החלטת לממש באופן שמימשת הפתרון.

טרם תיגש לתרגיל, אנא הקדש כ-15 דקות, לעיין בחוברת התרגול הזו, וודא כי הדרישות והתרגיל ברורים לך, וכמו כן ברורה לך צורת ההגשה.
שים לב, ייתכן ולא תסיים את כל תוכן התרגיל בזמן הכיתה, אל נא תרגש, זה תקין. השלם את התרגיל בבית, והכן אותו להגשה ניאותה לשיעור הקרוב.
ייתכן ובזמן התרגיל, תיתקל בנושאים ו\או כלים שלא הכרת אותם כלל, נסה להבין מתוך התרגיל, מה המידע הנחוץ לך על הכלי, על מנת שתוכל להשלים המשימה.
שים לב, אין בתרגילים הללו כל מוקשים, על כן תוודא שהינך עובר באופן מלא גם על הרפו המשוייך לכיתה, בטרם אתה נעזר במנוע חיפוש.

חקר מקרה

ברוכים הבאים למכון הטכנולוגי לחקר הזבובאים "תוש כל תוש". המכון נוסד ע"י כב' הפרופ' צוף-תוש, בשנת 2046, ומציע שירות time tracker, שמאפשר לכל יתוש להגיע בזמן לארוחת הצוף היומית שלו.
במהלך הפגישה באתר הלקוח, נפגשת עם גני מתגוננני, ראש תחום אבטחת המידע בארגון, שמספר לך על התרעת אבטחה, שגורם עויין (כנראה גורם ממשלתי מטעם מדינת סנו-בול), רוקם מתקפה שהוא מכנה "אל-תוש-בי", שמטרתה להחדיר לקוד המקור בצורה כזו או אחרת פרצת קוד הידועה בתור "DEET", אשר עלולה לגרום לסלידה של הלקוחות הנאמנים שלכם ממזון, ובכך לאיחורים משמעותיים.
המערכת נכתבה בשפת JAVA, ע"י צוות פיתוח חיצוני (המכון עוסק במחקר, לא בפיתוח).
בפגישה עם מנהל הפיתוח, הוא מעוניין לעבוד ב2 ענפים הקיימים לקוד המקור (master, develop).
את התוצרים של תהליך הבילד, הוא מעוניין לשמור באיזה שהוא מקום, שיקל עליו את האיתור של הגירסא, ואת הענף המקורי של קוד ממנו הוא נבנה. הוא אינו מכיר כלל את עולם ה devops, ואתה שם לב שכל הזמן הוא מנסה ללכוד זבובים סוררים. רק הוא מעיר לך כי היות שחלק מהזבובים הסוררים הם גם המפתחים שלו, ובגלל שהם זבובים, ייתכן שלא כל הבדיקות בענף הפיתוח Develop, יעברו בהצלחה ("מה לעשות" הוא מעיר במבט עייף, "הם כל הזמן עפים.."), אבל הוא כן מעוניין שאפילו שהבדיקות לא עובדות, הוא עדיין רוצה לראות את התוצרים.

כמו כן הוא מבקש, שכל בילד שרץ, מייצר תוצר חדש עם גרסא חדשה, כלומר לבילד מספר 10, הגרסא תהה x.x.x.10 . כמו כן, הוא מבקש ששם התוצר יהיה גם YATOSH, אבל שיהיה ניתן לשנות את שמו טרם התחלת הבילד.
כמו כן הוא תהה האם תוכל לייצר docker image, שמכיל את התוצר קוד, אבל מתי שמפעילים אותו במקום לגשת לURI הבא http://127.0.0.1:4444/time-tracker-1.3.40, יהיה ניתן לצרוך השירות בצורה הבאה: http://localhost:4444/yatosh, כי נמלות הבדיקות, שאמנם חרוצות הן מאד, הן מתקשות להבין איזה uri לגשת ולבדוק את התוצרים.
כמו כן הוא מבקש שבמה שלא תבנה, לא יהיה סיסמאות גלויות, בגלל שהיה להם פעם דליפת סיסמאות, וכרגע דב האבטחה, נמצא בשנת חורף עמוקה ואינו יכול לשנות הסיסמאות במידה וידלפו.
הוא מבקש שתנסה לתת מענה לבעיות של גני מתגוננני, אבל רק לאחר שהדרישות שלו בוא לסיפוקן.
כמו כן הוא מתלבט לגבי אופן ההטמעה, יש לו איזה שרת בצד, שהוא היה רוצה לבצע לו הטמעה מרחוק, הוא שמע על איזה כלי Ansible שמו, והיה מעוניין לבחון את האפשרות להשתמש בו להטמעה.
הוא גם שמע שיש כלי שנקרא Vagrant שיכול להרים מכונות בלחיצת כפתור. הוא מבקש שתיתן לו אפשרות כזו במה שלא תבנה לו. שהוא חושב על זה, שתיתן למי שמתחיל את תהליך הבילד, אפשרות בחירה, האם להטמיע באמצעות קונטיינר שרץ מקומית, או באמצעות הרמת מכונה מלאה מקומית, או הטמעה לתוך אותו שרת מרוחק.

איך ניגשים לפיתרון התרגיל

מבוא:

כנראה אתה התלמיד, אינך מבין מה מתחולל כאן, ישנם יתושים, אולי חזירי בר, אולי כותב התרגיל נטל סמים קשים טרם הכתיבה. אולם התרגיל, נועד לאתגר לא רק את היכלת שלנו לבוא ולהשתמש בכלים שנרכשו, אלה לעזור לנו להפריד בין עיקר וטפל, ולאפשר לנו להוציא את מרבית המידע הניתן לנו בחרק המקרה (מידע ישיר או עקיף). חשוב מאד כאן לא לרוץ להתחיל לקודד או\ו לבנות דברים, אלה לשבת, לזקק המידע, להבין מה נדרש מעימנו, איך אנו תופסים או "רואים" את הפתרון ואת השלבים הנדרשים, איזה מידע נדרש מאיתנו להשלים, ולשרטט.

ניתוח שחקנים, רכיבים,שלבי ביצוע, אילוצים ודרישות

מי הם השחקנים שלהם נדרש השירות?

_____
_____
_____

מה שלבי הביצוע הנדרשים מעימנו?		
שם שלב \ מספר	תיאור השלב (תיאור קצר ביותר)	מה ההישג הנדרש בשלב

מה הדרישות שלנו		
שם שלב \ מספר	דרישות בשלב	מה הרכיבים המשתתפים

ניתוח תהליך רצוי

איך לדעתינו יראה התוצר שלנו

JOB PROPERTIES	JOB NAME		
	JOB PARAMS		
	JOB ENV		
STAGES	STAGE 1		
	STAGE 1		
	STAGE 1		
	STAGE 1		
	STAGE 1		
END	ON SECCESS		
	ON FAILURE		

עבודה

Appendixes:

Jenkins File Example:

JenkinsFile 1 (Scripted)

```
#!/usr/bin/env groovy
pipeline {
    agent any
    parameters {
        choice(choices: ['Latest', 'Label','Changeset','DateTime'], description: 'how to pull code', name: 'code_pull_type')
        string(defaultValue: "", description: "for Changesets (\"C<ChangeSetID>\") b.Date/Time (\"D<DateTime_UTC>\") c.Label (\"L<Label>\") d.Latest Version (\"T\") ", name: 'code_pull_value')
        choice(choices: ['DEV', 'RELEASE'], description: 'what branch to build', name: 'branch')
    }
    environment {
        TFS_COLLECTION = "https://BilbiAndMe.visualstudio.com"
        TFS_REMOTE_WORKSPACE_PATH='$/'
        TFS_LOCAL_WORKSPACE_PATH="${WORKSPACE}\\TFS_WORKSPACE\\"
        TFS_LOCAL_OUTPUT_PATH="${WORKSPACE}\\PACKAGES\\"
        TFS_WORKSPACE_NAME="Hudson-${JOB_NAME}-${BUILD_NUMBER}"
        TFS_UTIL="tf"
    }
        def is_missingWorkspace = null
        def scm_versionGetCmd = null
        def build_projRef = null
    stages {
        stage('preperations'){
            steps {
                try {
                    dir("${TFS_LOCAL_WORKSPACE_PATH}"){
                            echo "workspace exists"
                            is_missingWorkspace = false
                        }
                    }
                catch (Exception e) {
                    is_missingWorkspace = true
                }

                switch(code_pull_type) {
                    case "Latest":
                        scm_versionGetCmd = "T"
                        break
                    case "Label":
                        scm_versionGetCmd = "L${code_pull_value}"
                        break
                    case "Changeset":
                        scm_versionGetCmd = "L${code_pull_value}"
                        break
                    case "Changeset":
                        scm_versionGetCmd = "C${code_pull_value}"
                        break
                }
                echo "${scm_versionGetCmd}"
                if (branch=="RELEASE"){
                    build_projRef="BuildMachine\\BuildMachine-dev.sln"
                } else {
                    build_projRef="BuildMachine\\BuildMachine.sln"
                }
            }
        }
        stage('pull SCM'){
            steps {
                // check if initial workspace exists, if not, create one, else get specific version
                withCredentials([usernamePassword(credentialsId: 'azure-jenkins-user', passwordVariable: 'pssd', usernameVariable: 'usrnme')]) {
                    if (is_missingWorkspace){
                        echo ("Checking out source code latest")

                            checkout([$class: 'TeamFoundationServerScm'
                                , credentialsConfigurer: [$class: 'AutomaticCredentialsConfigurer']
                                , projectPath: '$/'
                                , serverUrl: "${TFS_COLLECTION}"
                                , useOverwrite: true
                                , useUpdate: true
                                , workspaceName: "${TFS_WORKSPACE_NAME}"
                                , localdir: "${TFS_LOCAL_WORKSPACE_PATH}"
                                , password: "${pssd}"
                                , userName: "${usrname}"
                                ])
                    }


                    dir("${TFS_LOCAL_WORKSPACE_PATH}"){
                        /*bat "${TFS_UTIL} resolve /auto:TakeTheirs /noprompt /login:${TFS_USER},${TFS_PASS}"*/
                        bat '''
                            ${TFS_UTIL} get \$/ ^
                                /overwrite ^
                                /recursive ^
                                /force ^
                                /noprompt ^
                                /version:"${scm_versionGetCmd}"
                                /login:${usrnme},${pssd}

                        '''
                    }
                }

            }
        }
                dir("${TFS_LOCAL_WORKSPACE_PATH}"){
                    bat """
                        msbuild.exe ^
                            ${build_projRef} ^
                            /T:restore;build;package ^
                            /p:Configuration=Debug ^
                            /p:DeployOnBuild=true ^
                            /p:TypeScriptToolsVersion=latest ^
                            /p:PackageAsSingleFile=true ^
                            /p:DesktopBuildPackageLocation=${TFS_LOCAL_OUTPUT_PATH}\\ ^
                            /p:WebPublishMethod=WebPublishMethod
                    """
                }
            }
        }
        stage('publishing 2 nexus') {
            steps {
                echo "bla"
                // dir("${TFS_LOCAL_WORKSPACE_PATH}"){
                //      /*bat "${TFS_UTIL} resolve /auto:TakeTheirs /noprompt /login:${TFS_USER},${TFS_PASS}"*/
                //  nexusArtifactUploader artifacts: [[artifactId: 'obj_name', classifier: '', file: 'artifact.zip', type: 'zip']], credentialsId: 'nexus', groupId: '7-bilbi', nexusUrl: 'localhost:8081', nexusVersion: 'nexus3', protocol:
'http', repository: 'web-packages', version: 'd'
                // }
            }
```

JenkinsFile 2 (Last class practice)

```
node {

  stage('Preparation') {
            git 'https://github.com/zivkashtan/course.git'
  }
  stage('Build') {
            def mvnHome = tool name: '3.6', type: 'maven'
    sh "'${mvnHome}/bin/mvn' -Dmaven.test.failure.ignore clean package"

            writeFile file: "Dockerfile", text: """
                                FROM tomcat:8.0.20-jre8
                                COPY ./web/target/*.war /usr/local/tomcat/webapps/
                        """
                        sh "docker build -t time-tracker:${env.BUILD_ID} ."

                        sh """
                          docker login -u admin -p admin123 localhost:8123
                          docker tag time-tracker:${env.BUILD_ID} localhost:8123/time-tracker:latest
                          docker push localhost:8123/time-tracker:latest
                        """
  }

}
```

JenkinsFile 2 (With DockerRegistry)

```
node {
   checkout scm

   docker.withRegistry('https://registry.example.com', 'credentials-id') {

      def customImage = docker.build("my-image:${env.BUILD_ID}")

      /* Push the container to the custom Registry */
      customImage.push()
   }
}
```

JenkinsFile 2 (sonarQube)

```
node {

  stage('Preparation') {
                git 'https://github.com/zivkashtan/course.git'
  }
  stage('Build') {
                def mvnHome = tool name: '3.6', type: 'maven'
    sh "'${mvnHome}/bin/mvn' -Dmaven.test.failure.ignore clean package"

                writeFile file: "Dockerfile", text: """
                                FROM tomcat:8.0.20-jre8
                                COPY ./web/target/*.war /usr/local/tomcat/webapps/
                            """
                            sh "docker build -t time-tracker:${env.BUILD_ID} ."

                            sh """
                              docker login -u admin -p admin123 localhost:8123
                              docker tag time-tracker:${env.BUILD_ID} localhost:8123/time-tracker:latest
                              docker push localhost:8123/time-tracker:latest
                            """
  }
  stage('Results') {
    junit '**/target/surefire-reports/TEST-*.xml'
                def scannerHome = tool name: 'scanner', type: 'hudson.plugins.sonar.SonarRunnerInstallation'
                writeFile file: "sonar-project.properties", text: """sonar.projectKey=tt
                sonar.projectName=time-tracker
                sonar.host.url=http://localhost:9000
                sonar.sources= /var/lib/jenkins/workspace/test/core/src
                """
               withSonarQubeEnv {
                            sh "${scannerHome}/bin/sonar-scanner"
                            }
                            sh 'cp web/target/time-tracker*.war web/target/time-tracker.war'
    nexusArtifactUploader artifacts: [[artifactId: 'time-tracker', classifier: 'pom', file: 'web/target/time-tracker.war', type: 'war']], credentialsId: 'afa29d1d-e9ec-45f9-909f-33a987f06069', groupId: 'time-tracker', nexusUrl: 'localhost:8081', nexusVersion: 'nexus3', protocol: 'http', repository: 'time-tracker', version: "${env.BUILD_ID}"
  }
}
```

JenkinsFile 2 (With Nexus Uploader)

```
pipeline {
    agent {
        label "master"
    }
    tools {
        // Note: this should match with the tool name configured in your jenkins instance (JENKINS_URL/configureTools/)
        maven "Maven 3.6.0"
    }
    environment {
        // This can be nexus3 or nexus2
        NEXUS_VERSION = "nexus3"
        // This can be http or https
        NEXUS_PROTOCOL = "http"
        // Where your Nexus is running
        NEXUS_URL = "172.17.0.3:8081"
        // Repository where we will upload the artifact
        NEXUS_REPOSITORY = "repository-example"
        // Jenkins credential id to authenticate to Nexus OSS
        NEXUS_CREDENTIAL_ID = "nexus-credentials"
    }
    stages {
        stage("clone code") {
            steps {
                script {
                    // Let's clone the source
                }
            }
        }
        stage("mvn build") {
            steps {
                script {
                    // If you are using Windows then you should use "bat" step
                    // Since unit testing is out of the scope we skip them
                    sh "mvn package -DskipTests=true"
                }
            }
        }

        stage("publish to nexus") {
            steps {
                script {
                    // Read POM xml file using 'readMavenPom' step , this step 'readMavenPom' is included in: https://plugins.jenkins.io/pipeline-utility-steps
                    pom = readMavenPom file: "pom.xml";
                    // Find built artifact under target folder
                    filesByGlob = findFiles(glob: "target/*.${pom.packaging}");
                    // Print some info from the artifact found
                    echo "${filesByGlob[0].name} ${filesByGlob[0].path} ${filesByGlob[0].directory} ${filesByGlob[0].length} ${filesByGlob[0].lastModified}"
                    // Extract the path from the File found
                    artifactPath = filesByGlob[0].path;
                    // Assign to a boolean response verifying If the artifact name exists
                    artifactExists = fileExists artifactPath;
                    if(artifactExists) {
                        echo "*** File: ${artifactPath}, group: ${pom.groupId}, packaging: ${pom.packaging}, version ${pom.version}";
                        nexusArtifactUploader(
                            nexusVersion: NEXUS_VERSION,
                            protocol: NEXUS_PROTOCOL,
                            nexusUrl: NEXUS_URL,
                            groupId: pom.groupId,
                            version: pom.version,
                            repository: NEXUS_REPOSITORY,
                            credentialsId: NEXUS_CREDENTIAL_ID,
                            artifacts: [
                                // Artifact generated such as .jar, .ear and .war files.
                                [artifactId: pom.artifactId,
                                classifier: '',
                                file: artifactPath,
                                type: pom.packaging],
                                // Lets upload the pom.xml file for additional information for Transitive dependencies
                                [artifactId: pom.artifactId,
                                classifier: '',
                                file: "pom.xml",
                                type: "pom"]
                            ]
                        );
                    } else {
                        error "*** File: ${artifactPath}, could not be found";
                    }
                }
            }
        }
    }
}
```

SonarQube

Intro:

SonarQube (formerly Sonar) is an open-source platform developed by SonarSource for continuous inspection of code quality to perform automatic reviews with static analysis of code to detect bugs, code smells, and security vulnerabilities on 20+ programming languages. SonarQube offers reports on duplicated code, coding standards, unit tests, code coverage, code complexity, comments, bugs, and security vulnerabilities.
SonarQube can record metrics history and provides evolution graphs. SonarQube provides fully automated analysis and integration with Maven, Ant, Gradle, MSBuild and continuous integration tools (Atlassian Bamboo, Jenkins, Hudson, etc.).

Main Components:

SonarQube server:
The server itself, used by InfoSec teams
SonarQube Runner:
The executable agent located usually on the CI build machine. Keep in mind there is need for connection between the runner and the sonarqube server
Jenkins Plugin:
This plugin enables the code scanner, and knows how to submit the results to sonarQube server
- Sonar: (https://plugins.jenkins.io/sonar)

Work Flow:

1. Install the SonarQube server
2. On CI build server:
    1. Install the SonarQube Runner
    2. Configure the SonarQube runner (o
3. Jenkins:
    1. Install the SonarQube Jenkins Plugin
    2. Configure the connections to SonarQube Server in global configs
    3. Configure the runner in the global tools configurations
4. Config in sonarqube new project

Installations:

SonarQube server:
On server:

```
//create user
groupadd sonar
useradd sonar -G sonar

//download zip
wget https://sonarsource.bintray.com/Distribution/sonarqube/sonarqube-6.4.zip
//Unzip it
sudo yum -y install unzip
sudo unzip sonarqube-6.4.zip -d /opt
//setup service
cat << EOF > /etc/systemd/system/sonar.service
[Unit]
Description=SonarQube service
After=syslog.target network.target

[Service]
Type=forking

ExecStart=/opt/sonarqube/bin/linux-x86-64/sonar.sh start
ExecStop=/opt/sonarqube/bin/linux-x86-64/sonar.sh stop

User=sonar
Group=sonar
Restart=always

[Install]
WantedBy=multi-user.target
EOF
sudo systemctl start sonar
sudo systemctl enable sonar
```

On Docker:

SonarQube Runner:

```
Wget http://repo1.maven.org/maven2/org/codehaus/sonar/runner/sonar-runner-dist/2.4/sonar-runner-dist-2.4.zip
sudo unzip sonar-runner-dist-2.4.zip -d /opt
```

Jenkins Plugin: