https://meu-solutions.com/qa-automation-pipeline-with-ci-cd-cucumber-xray-jira-selenium-ruby-jenkins/

WHAT IS JENKINS

Jenkins automates building, testing, reporting, packaging, staging, deploying the application by using plugins.

Jenkins offers a simple way to set up a continuous integration or continuous delivery environment for almost any combination of languages and source code repositories using pipelines, as well as automating other routine development tasks.

Jenkins can be installed through native system packages, Docker or even run standalone by any machine with Java Runtime Environment installed. Jenkins is not an Application Lifecycle Management System rather it is just a continuous integration and continuous deployment system. It only builds and tests your project and then deploys to different environments.

WHAT IS CI/CD PIPELINE

Pipeline is a set of processes that take the code from version control and compile, build, test and deploy to production in automated fashion. The pipeline breaks down the software delivery process into stages. Each stage is made of different tasks which can be also carried out in parallel. When all tasks in a stage passes, next stage is triggered.

The CI/CD pipeline is one of the best practices for DevOps teams to implement, for delivering code changes more frequently and reliably.

Continuous Integration is a practice that helps to detect errors quickly and locate them easily. Once a culture of continuous integration has been set up, the next step is Continuous Delivery. Continuous Delivery is a set of practices that ensures that all the code that is produced is ready for deployment.

Continuous Integration is an automated build and execution of at unit and integration tests, performing code analysis. Continuous integration is a coding philosophy and set of practices that drive development teams to implement small changes and check in code to remote repositories frequently. Because most modern applications require developing code in different platforms and tools, the team needs a mechanism to integrate and validate its changes.

MAIN GOAL OF THE CI/CD is providing shorter release cycle. So, end users can get new version of application with new features of fixed bugs and better application.

INTEGRATING TESTING PROCESS INTO CI/CD PIPELINE

Continuous testing goes beyond test automation. Automated testing frameworks help quality assurance engineers define, execute, and automate various types of tests that can help development teams know whether a software build passes or fails. They include functionality tests that are developed at the end of every sprint and aggregated into a regression test for the entire application. These regression tests then inform the team whether a code change failed one or more of the tests developed across all functional areas of the application where there is test coverage.

A best practice is to enable and require developers to run all or a subset of regressions tests in their local environments. This step ensures that developers only commit code to version control after regression tests pass on the code changes.

Regression tests are just the start. Performance testing, API testing, static code analysis, security testing, and other testing forms can also be automated. The key is to be able to trigger these tests either through command line, webhook, or web service and that they respond with success or fail status codes.

Once testing is automated, continuous testing implies that the automation is integrated into the CI/CD pipeline. Some unit and functionality tests can be integrated into CI that flags issues before or during the integration process. Test that require a full delivery environment such as performance and security testing are often integrated into CD and performed after builds are delivered to target environments.

INSTALLING JENKINS

Pre-requisites:

1. Java (Latest is good)

2. Jenkins war file

Supporting tools:

3. Maven

4. Nunit

5. Nuget

6. Git

- Go to Jenkins.io web application

- Download the war file which is convenient for your computer

- Open terminal and change directory into jenkins.war file folder (cd + drag and drop the war file into terminal)

- "java -jar jenkins.war" + enter (if there any problem try to use home-brew way to install - https://formulae.brew.sh/formula/jenkins)

- At the bottom of terminal find the password and copy it

- Open chrome and type "localhost:8081" + enter or any EC2 IP from AWS or company's local server IP

- Paste the password

- accept recommended plugins

- create username password and other information in order to administration of jenkins

- Now installation is done

CREATING EC2 FROM AWS

- Download "Microsoft Remote Desktop 10" from app store for macOS

- go to aws.amazon.com

- open an AWS account ==>xxxxx@gmail.com / password \*\*\*\*\*\*\*\* (you are supposed to have an account what we have used for SQL)

- Go to account settings and choose region as "North Virginia" instead of "global"

- And then watch "JENKINS AMI INSTALLATION VIDEO" on VA-Automation/Modules.

- For class records watch "CI/CD with Jenkins | 08/25/2019" on the module again.

- localhost:8081 ==> my port in my company is port 8081

- localhost:8081 means ==> access port 8081 in this computer

- I cannot use this port address on MAC because it is in Windows

If you want to connect AWS and work about Jenkins:

1. Go to awsamazon.com

2. Click my account

3. Enter email address and password as above

4. Click console home

5. Click Launch a virtual machine

6. Search "cybertek" specifically on search box

7. Click "Community AMIs" tab

8. Click Cybertek-Jenkins-2019 ==> Select link

9. Click Next again Next again Next again Next until reaching "Configure Security Group" page

10. When you reach "Configure Security Group" page choose "All Traffic" as Type and "Anywhere" as Source and click next.

11. And then click "Launch" on "Step 7: Review Instance Launch" page.

12. Choose any key pair on pop up page (It is not important, you can type "newkey") and then click "Launch Instances".

13. Then click "View Instances" on Launch Status page.

14. You will see all your created Instances on this page. Choose what you want and then click "Connect" on top of page.

15. You will see a pop up page. Click only "Download Remote Desktop File" button.

16. You will be asked for username and password. Enter as below. You can also find those password on

Canvas video which tells "JENKINS AMI INSTALLATION VIDEO" on VA-Automation/Modules.

17. Once you click OK "Microsoft Remote Desktop 10" application will be launch.

18. Launch chrome browser and type "localhost:8081" as url and enter.

19. You will be asked for username and password again. This time use "admin" for both.

20. Once you click enter that is all.... Good luck with =Jenkins...

\*\*\* What we have done till now. We have created a server(EC2 machine). But dependency applications are missing now.

Normally DevOps team is responsible for installing those dependency applications...

- We need at least Java, Maven, browser and git on that remote machine. We have to tell DevOps to install those applications.

- In my company DevOps or Operations Team (depends on company who is in charge for Jenkins) maintains Jenkins issues.

- As a tester my Jenkins account is not authorized to create new jobs on Jenkins. I can only see the jobs and results...

- To create a job for my smoke test on Jenkins, I create a Task Ticket on Jira for DevOps team.

For Exp: Create Smoke test on Jenkins for team Alpha Project Vytrack. I need maven, browsers, git, java 8 minimum, etc.

\*\*\*Enter the following user name and password to access EC2 instance:

username: Administrator

password: ?VCwbM5NjcQ

CONFIGURING GLOBAL SECURITY

- You can set your database from here (local or any dedicated server for that job)

- You can set git issues

- You can set the jenkins url

- you can set docker

- you can set email notification for test results

GLOBAL TOOL CONFIGURATION

- you can set git and maven settings from here

MANAGE USERS

- you can manage the user with specific authorization from here

ADDING PLUGINS

- Click Manage Jenkins on main page of the Jenkins

- Click Manage Plugins

- Click Available

- Search for cucumber report and git plugins

CREATING A NEW FREE STYLE PROJECT

1. Click "New Item"

2. Give a name like "Smoke\_Test"

3. Choose "Free Style" type and click OK on bottom side

4. Now we are on configuration page

CONFIGURING A FREE STYLE JOB

GENERAL

1. Do not change anything here

SOURCE CODE MANAGEMENT

1. Click Git

2. Provide repository url information (I used Vasya's CucumberOnline repository)

3. If your repo is private you must provide your credential

4. Provide which branch we want to run (\*/master, develop, etc.)

5. Click save

BUILD TRIGGERS

In this section we will set up how often and what the job is going to be run...

Build Periodically ==> Runs based on a schedule which we use mostly

Build after other projects are built ==> For example we want to run smoke test after each deployment by developers.

Once developers deploy something, our smoke test is going to be run

Type "H 6 \* \* 1-5" exactly this statement to the ==> Build Periodically window. It means that it will be run around 6 am from Monday to Friday.

Whole Syntax explanation is below:

MINUTE HOUR DOM MONTH DOW

MINUTE Minutes within the hour (0–59)

HOUR The hour of the day (0–23)

DOM The day of the month (1–31)

MONTH The month (1–12)

DOW The day of the week (0–7) where 0 and 7 are Sunday, 1 is Monday

0 \* \* \* \* ==> Build every hour at exactly on the hour (6.00 am, 8.00pm)

0 8 \* \* \* ==> Build every day exactly 8.00 am

H/5 \* \* \* \* ==> Build every 5 minutes

H 6 \* \* 1-5 ==> Build around 6 am from first day of the week (Monday) to 5th day (Friday)

H 1 \* 6,20 \* \* ==> Build on 6th and 20th day of the month around 1 am for regression

BUILD ENVIRONMENT

We can manage the environment variables.

BUILD

- This is one of the most important part.

- We describe what Jenkins Job will do specifically.

- Since we create Maven Projects and we will run the test through Maven.

- Choose "Invoke top level Maven targets" from dropdown menu in Build.

- Choose "MAVEN\_HOME" option under as Maven version under Build. ==> NOTE: IOT to choose MAVEN\_HOME from here DevOps team is supposed to configure Jenkins with Maven in "Global Configuration Tool" menu in Jenkins main setups.

- Type "clean verify -Dcucumber.options="--tags @regression"" as goal. It runs this specific @regression tags.

- Look below lines for more information about this "type" window...

\*\*\* We can also specify by using the tags instead of separate runner class for smoke and regression suite.

So;

verify -Dcucumber.options="--tags @smoke" ==> for smoke test

verify -Dcucumber.options="--tags @regression" ==> for regression tets

verify -Dcucumber.options="--tags @smoke -dry-run" or ==> Run as dry-run

verify -Dcucumber.options="--dry-run" ==> Run as dry-run

POST-BUILD ACTIONS

- We can add steps to report the result. In my project we use Cucumber plain to generate reports for my tests.

- Click "Add post-build actions" on Post-Build Actions Window.

- Then click "Cucumber Report" on dropdown menu

- In our CukesRunner(In codes which comes from GitHub), we have to have "json:target/cucumber.json" options. Without it We cannot get cucumber reports on Jenkins. Because all HTML reports converted from json format.

- When we click "Cucumber reports" on left navigation bar on Jenkins, it will show us the reports by using some graphics.

HOW TO SEND EMAILS AFTER EACH BUILD

- Click "Add post-build actions" on Post-Build Actions Window.

- Then "Editable Email Notification" on dropdown menu

- Add your team members' email addresses to "Project Recipient List" window

- Content type ==> type only "HTML"

- Default Subject==> xxx

- Click "Advance Settings" then find "Triggers".

- Then delete default content (delete developers).

- Click "Add Trigger" again and choose "Always". If we do not choose "Always", then developers will not get any notification email if tests pass successfully. If there is any failure, developers will get a notification email. If you want to add more people from your team you have to add their email address from "Project Recipient List" window on top of the same page.

- IOT get email reports from Jenkins, we have to set our EC2 machine's IP address into Jenkins configuration. For this; click "Jenkins" on top left, hover over "Manage Jenkins", then click "Change System Configuration". Then insert the Jenkins web page URL into "Jenkins URL" under "Jenkins Location". ==> URL: http://ec2-3-91-24-15.compute-1.amazonaws.com:8081 (It is different for each AWS EC2 machine. Put your IP)

- We can also publish the test results if company has Jira Xray.

RUNNING TEST

1. Click "Build Now" button to start run.

2. There will appear #1 under Build History Title at the same page

SEEING THE CONSOLE OUTPUT OF THE TEST

1. Click the small downward arrow just next to #1 or #X (#X: number of the test).

2. Click Console Output ==> You see all previous test console outputs.

3. You can see also the place where the jerkins job is stored on your computer from the console output.

\* Project can also be built from Maven Command Line by using "chocolatey". I don't have any idea about this for now.

CONFIGURING JENKINS TO RUN A BUILD AUTOMATICALLY ON CODE PUSH TO GITHUB

Running builds as soon as code is pushed to the central repository is important, as it enables you to detect problems quickly and locate them more easily.

1. Get the Hook URL of Jenkins:

- Open the Jenkins Dashboard.

- Go to: Manage Jenkins > Configure System

- Under GitHub Plugin Configuration, Click on ‘Advanced…’

- Check ‘Specify another hook url for GitHub configuration’ checkbox to see hook URL

- A text box will appear with a hook URL. This is the Hook URL at which Jenkins will listen for POST requests. Copy this URL and uncheck the ‘Specify another hook url for GitHub configuration’ checkbox again.

2. Go to your remote repo on GitHub:

- Click ‘Settings’ on the navigation bar

- Click ‘Webhooks & services’ on the navigation bar on the left-hand side of the screen.

- Paste the hook URL that you copied from Jenkins into the ‘Payload URL’.

- You can select the events for which you want the Jenkins build to be triggered. We will select ‘Just the push event’ because we want to run the build when we push our code to the repository.

- Click ‘Add webhook’ to add the webhook.

- You should now see the webhook you just added in the list of Webhooks for that repository like this.

3. Go to Jenkins Project:

- We now have Jenkins configured to run builds automatically when code is pushed to central repositories. However, Jenkins doesn’t run all builds for all projects. To specify which project builds need to run, we have to modify the project configuration.

- In Jenkins, go to the project configuration of the project for which you want to run an automated build.

- In the ‘Build Triggers’ section, select ‘Build when a change is pushed to GitHub’.

- Save your project.

Now, Jenkins will run the build automatically once you push your code to the GitHub repository.

CREATING A PIPELINE PROJECT

1. Click "New Item"

2. Give a name like "Regression\_Test"

3. Choose "Pipeline" type and click OK on bottom side

4. Now we are on configuration page

CONFIGURING PIPELINE PROJECT

\* It is mostly similar to freestyle project. Only change is Pipeline Tab on bottom of the page.

- click configure on project page

- choose "Pipeline script" from Definition

- choose "Use Groovy Sandbox" checkbox

- click "Pipeline Syntax" link and open in new browser tab

- in new tab choose "node: Allocate node" from Sample Step dropdown ==> it will generate a node to run the tests

- Type "master" as Label

- click "Generate Pipeline Script" ==> it will give us groovy syntax to copy it and paste to main jenkins page

\* it will a syntax like that ==> "node{ }" --> you will put the next syntaxes into that curly braces

- in new tab choose "git:Git" from Sample Step dropdown

- copy and paste your repository url

- choose your branch and credentials specification

- click "Generate Pipeline Script" ==> it will give us groovy syntax to copy it and paste to main jenkins page

- paste it into "Script" window on main jenkins page

- go back to Pipeline Syntax page again

- choose "bat: Windows Batch Script" from Sample Step dropdown menu

- Type "mvn verify" into Batch script window, click "Generate Pipeline Script" then copy and paste as previous

REPORTING FOR PIPELINE

https://jenkins.io/DOC/BOOK/PIPELINE/ ==> read from there that how to create jenkins pipeline and its features.

- First we have to install plugin for reporting. Click "Manage Jenkins"

- Click "Manage Plugins"

- Click "Available" tab

- Search for "cucumber report" from right top of the page

- Select "Cucumber report" and click "install without restart"

- Logout and login jenkins IOT refresh the plugins

- click again "configure" on pipeline project page

- click "Pipeline Syntax" link and open in new browser tab

- in new tab choose "step: General Build Step" from Sample Step dropdown

- This time you will "Cucumber reports" on "Build Step" dropdown menu after installing cucumber report

- You can also set up more option if you click "Advanced" button

- click Generate Pipeline Script

- Copy the syntax and paste into node{///} again on main page.

- Then save it again

- After clicking "Build Now" button one more time; it will execute the tests and generate report on Jenkins as well.

- Go to main menu of the pipeline. Click the number of the execution (example: #3) just below the Build History, now you will see "Cucumber reports" button on left side of the page.

EXAMPLE FOR GROOVY CODES FOR PIPELINE

node {

// Select SCM

stage("Pulling code from github") {

git 'https://github.com/CybertekSchool/summer-online2019-web-orders.git'

}

// build

stage("Running tests") {

if (isUnix()) {

sh label: '', script: 'mvn clean verify'

} else {

bat label: '', script: 'mvn clean verify'

}

}

// generate report

stage("Generate cucumber report"){

cucumber fileIncludePattern: '\*\*/\*.json', sortingMethod: 'ALPHABETICAL'

}

//send email

stage("Send email"){

emailext body:

'''${JELLY\_SCRIPT, template="html"}

$PROJECT\_NAME - Build # $BUILD\_NUMBER - $BUILD\_STATUS

<br>

<strong>Check Cucumber HTML report: $PROJECT\_URL/$BUILD\_NUMBER/cucumber-html-reports/overview-features.html</strong>''',

subject: 'Smoke test status: $BUILD\_STATUS',

to: '$DEFAULT\_RECIPIENTS'

}

}

CREATING STAGES FOR PIPELINE

- If you use the syntax below inside the node{ /// }, it will generate some fancy stage view during and after the build of the pipeline on main page.

stage('Pulling code from github') {

git 'https://github.com/CybertekSchool/summer-online2019-web-orders.git'

}

stage('Build and test') {

bat label: '', script: 'mvn clean test'

}

JENKINS PIPELINE SCRIPT

https://jenkins.io/DOC/BOOK/PIPELINE/ ==> read from there that how to create jenkins pipeline and its features.

- Go to your Jenkins file by using your EC2 machine

- Click Manage Jenkins

- Click configuration system

- Make sure that Jenkins URL window is corresponding with your EC2 machine (same device)

- Create new job on jenkins

- Name it web\_orders\_smoke\_test\_with\_pipeline\_

- Click pipeline ==> instead of free style we are creating pipeline job

- Insert the codes below into Pipeline Script window

- Insert your email into that codes (last line)

- Click apply and then save

- Go to main page of Jenkins and click Build Now button

node {

​

// Select SCM

​

stage("Pulling code from github") {

git 'https://github.com/CybertekSchool/summer-online2019-web-orders.git'

}

​

// build

stage("Running tests") {

if (isUnix()) {

sh label: '', script: 'mvn clean test'

} else {

bat label: '', script: 'mvn clean test'

}

}

​

// generate report

stage("Generate cucumber report"){

cucumber fileIncludePattern: '\*/.json', sortingMethod: 'ALPHABETICAL'

}

​

//send email

stage("Send email"){

emailext body:

'''${JELLY\_SCRIPT, template="html"}

$PROJECT\_NAME - Build # $BUILD\_NUMBER - $BUILD\_STATUS

<br>

<strong>Check Cucumber HTML report: $PROJECT\_URL/$BUILD\_NUMBER/cucumber-html-reports/overview-features.html</strong>''',

subject: 'Smoke test status: $BUILD\_STATUS',

to: 'you@email.com'

}

}

CREATING REGRESSION ON ALL BROWSERS

- Click new item on jenkins

- Name it as "web\_orders\_regression\_on\_all\_browsers"

- Choose pipeline as job style again

- Click "pipeline syntax" on the bottom of the page. It will show us some syntax templates

- Choose ==> parallel : Execute in parallel then click question mark now it will show you how the syntax is for parallel execution (Vasya has already arranged the syntax and put into github).

- Then go to Pipeline section in configuration page again. Click "Pipeline script from SCM" instead of "Pipeline script"

- Pass your remote repository url into "Repository URL" window

- Pass "Jenkinsfile" into Script path window (Because your code script is written in this Jenkins file in github)

- Click Apply and Save

- Click "Build now" and it will send the regression test report into your email address

LINKING SELENIUM GRID WITH JENKINS

1. Create a new EC2 on AWS for Jenkins as we did before. Follow same way.

2. Click "connect" on top of page.

3. Click "Download Remote Desktop File"

4. Double click to this page in your downloads folder

5. Windows Remote Desktop Application will be launched automatically for Mac.

6. Enter username and password as above.

7. Copy selenium\_grid.zip file into windows machine desktop and extract all files and double click to "windows\_start\_hub\_and\_node" file. It will launch two different terminal, one for hub and one for nodes. Instead of typing all codes to launch and set hub and nodes, it helps to set up quickly. We can also use this zip file even working in company. It is easy and useful, keep it in somewhere to you later on.

8. Go to AWS console --> EC2 --> Click on Selenium Grid -->

9. Go to Driver class in your project and insert EC2 URL as we did before above.

URL: ==> http://ec2-3-91-241-154.compute-1.amazonaws.com:8081 (It is different for each AWS EC2 machine. Put your IP)

10. Come to Jenkins web site again. You create a new job. All steps are the same as what we did before. There is no special thing. The most important thing is, you will use the git url which contains the setup according to Selenium Grid. Because we prepared everything on AWS EC2 machine and in our Selenium Driver Class. Jenkins is only going to use that git url, the rest is the same.

ADDING SLACK PLUGIN

- Click Manage Jenkins on main page of the Jenkins

- Click Manage Plugins

- Click Available

- Filter slack from the right top corner (type "slack" and it will show only slack plugins)

- Choose all of them and download and restart Jenkins

- After restarting choose again Manage Jenkins

click configuration

com to bottom and find Slack section

insert summer2019... into Workspace window

Click Add just next to Credentials and choose Jenkins

As Kind ==> choose "Secret Text"

As Secret ==> Go to Slack Web Page. Click "App" on top left. Search for "Jenkins". Then click Jenkins from results. Then click View In App Directory button. Then click "Add to Slack". Choose any slack channel. Click "Add Jenkins CI Integration".

As ID type ==> "slack-token"

How do you handle test reports? ==> Jenkins is linked with Jira and Slack as well. We have notification from jenkins in multiple ways--> email, Slack, Jira X-Ray

Groovy syntax:

#!/usr/bin/env groovy

pipeline {

agent {

label 'master'

}

triggers{

cron env.TYPE == "testngRegression.xml" ? "13 0 \* \* \*" : ""

}

options {

ansiColor('xterm')

}

stages {

stage('checkout Mobile 2.0 Test Automation repository'){

steps {

git branch: 'master',

credentialsId: '592384e6-1979-45cf-9709-049915ee6a16',

url: 'git@bitbucket.org:arterramodesto/mobile2.0\_testautomation.git'

}

}

stage('Running Mobile2.0 Regression'){

when { environment name: 'TYPE', value: 'testngRegression.xml' }

steps {

sh 'mvn clean test -DtestngFile=testngRegression.xml -Dmaven.test.failure.ignore=true'

}

post {

failure {

emailext(

attachmentsPattern: 'output.zip',

subject: "Check output.txt file for results. Use Below link to see HTML reports from the Jenkins Dashboard",

body: "$JOB\_NAME - Build # $BUILD\_NUMBER - Failure:\n\nCheck console output at $JOB\_DISPLAY\_URL to view the results.",

recipientProviders: [[$class: 'CulpritsRecipientProvider']],

to: 'ykucukvatan@kajeet.com csura@kajeet.com'

)

}

success {

emailext(

attachmentsPattern: 'output.zip',

body: "$JOB\_NAME - Build # $BUILD\_NUMBER - Success:\n\nCheck console output at $JOB\_DISPLAY\_URL to view the results.",

subject: "Check output.txt file for results. Use Below link to see HTML reports from the Jenkins Dashboard",

recipientProviders: [[$class: 'CulpritsRecipientProvider']],

to: 'ykucukvatan@kajeet.com csura@kajeet.com'

)

}

}

}

stage('Running Mobile2.0 Smoke'){

when { environment name: 'TYPE', value: 'testngSmoke.xml' }

steps {

sh 'mvn clean test -DtestngFile=testngSmoke.xml -Dmaven.test.failure.ignore=true'

}

post {

failure {

emailext(

subject: "Check output.txt file for results. Use Below link to see HTML reports from the Jenkins Dashboard",

body: "$JOB\_NAME - Build # $BUILD\_NUMBER - Failure:\n\nCheck console output at $JOB\_DISPLAY\_URL to view the results.",

recipientProviders: [[$class: 'CulpritsRecipientProvider']],

to: 'ykucukvatan@kajeet.com csura@arterra.com'

)

}

success {

emailext(

body: "$JOB\_NAME - Build # $BUILD\_NUMBER - Success:\n\nCheck console output at $JOB\_DISPLAY\_URL to view the results.",

subject: "Check output.txt file for results. Use Below link to see HTML reports from the Jenkins Dashboard",

recipientProviders: [[$class: 'CulpritsRecipientProvider']],

to: 'ykucukvatan@kajeet.com csura@arterra.com'

)

}

}

}

stage('Publishing html reports') {

steps {

publishHTML([allowMissing: false, alwaysLinkToLastBuild: false, keepAll: true, reportDir: '\\reports\\', reportFiles: 'ExtentReport.html', reportName: 'HTML Report', reportTitles: ''])

}

}

}

}