211. What is Continuous Integration & Delivery. Understand the flow

how to include test automation in the CI/CD pipeline.

 in terms of Google, Microsoft, all this, they have continuous delivery every 2mins.

So Jenkins is a place where you will have multiple jobs either to run your test cases

or to push the code into production.

So the moment Webhook tells to Jenkins somebody pushed into my

Push the latest code into GitHub. If you configure GitHub Webhook Trigger Jenkins, GitHub will notify Jenkins when new code is added. Jenkins will execute all quality jobs, and if all TC’s passed, it will also execute a job that moves on into production. So all this automated process.

Interview : can you just create a high level CI/CD setup and make sure you are Jenkins test automation job runs when someone pushes the code into GitHub. ?

need to configure Webhook GitHub triggers, and you need to make sure once you receive

that notification, Jenkins job of your automation test run automatically when someone pushes the code.

A computer screen with a diagram

Description automatically generated

212. Configure GitHub acc and push the Selenium Framework into repository

let's use this framework repository itself to push our code and we make sure we run our test cases.

In command prompt

mkdir cicd – if u want to create any new subdirectory with the name cicd

cd cicd – if u want to inside the directory

whatever the framework is there in our local just copy it and paste it in the cicd folder.

Now, this project, let's push into GitHub first.

cd projectname – go inside the project

gitinit - So this command make your existing folder Git compatible.

What do you mean by Git compatible? So that means you can interact with the remote Git repositories in the internet and you can push your local code to them, So the communication between remote GitHub repositories and whatever you have in your local system can be enabled by simply providing this command(git init)

git branch – no branch

git checkout -b main – created one new branch ( branchname-main)

So for any Git repository, by default, the branch name would be main.

Git add .

Git commit -m “Initial files commit” – we have committed all your files in the folder.

Ready to push this code to the central repository.

Create new repo(Automation CICD) in github

Git remote add origin link ( repo link )

Git push origin main

if you're pushing the code to this repository, then this account needs to be authorized here.

Git eco system git credential manager -  It aims to provide a consistent and secure authentication experience, including multi-factor auth, to every major source control hosting service and platform.

Now, the code, whatever we have in our local, push it to the repository.

whoever pushes the code, this webhook has to go and tell Jenkins, hey, you had a push, go and run Jenkins job, automation job.

213. Install Fresh Jenkins war & configure necessary plugins & create Selenium Job

Google – Jenkins download – check ur java version and download the generic java package(.war) LTS file.

Clean up Jenkins if u have downloaded before – c/user/Hari.sankar/.jenkins folder delete.

How to clean up throw cmd.

C:\Users\Hari.Sankar>cd .jenkins

Open .

.Jenkins folder should be empty. So that way you can freshly install new Jenkins

on new Java version without getting any headache of compatibility issues.

 I'm asking to start Jenkins server at this port

cd downloads

java -jar Jenkins.war --httpPort=9090

browser -> localhost:9090 -> give the password after run the above command they will give the password -> continue -> Installed suggested plugins.

Once u landed on the Jenkins.  I want you to get another plugin(maven), because you know, we have a selenium Maven project with us in eclipse.

Manage Jenkins -> plugins -> Available plugins -> maven Integration select -> Install.

Whenever you install something, it's better to stop the server and restart again

with the same command. So that way it loads again freshly with the updated plugins.

New Item -> seleniumAuotamtionTests(maven project select) – k

Source code management -> git -> give the repo url(github repo) – optional ( add-> Jenkins-> kind(secrete text) - > pass the secret text

Go to github -> right side on account level click on settings -> developer settings -> personal access tokens -> token classic -> create new token -> generate new token(classic) -> select all the permissions(I/m giving all the permissions to the secrete key.) -> give note -> click on generate token -> copy token -> paste it in Jenkins -> add it.

now Jenkins can talk to the GitHub account with this secret text, and ask whatever it required.

We have given information from where it has to pull the code

Give the branch name (\*/branchname)

Build trigger -> GitHub hook trigger for GITScm polling(select) – save

Manage Jenkins -> tools -> Add Maven -> unselect Install automatically

Name : apache maven-3.9.6 -> paste the path where the maven is present in maven\_HOME – save.

Open the created job -> configure -> Build – goal and options (test -PRegression ) - save

it will only choose that regression profile in the Maven. – build now

When I trigger this job, it takes the code, downloads the code from the GitHub repository

and it triggers on that GitHub repository. It triggers this Maven command called test-Pregression, where a test will start execution.

214. Understand Github webhook trigger & configure it to activate Selenium Jenkin job

when someone pushes the code to GitHub, how will Jenkins know to trigger this quality job

and deployment job. That is the core point of CI/CD ?

what GitHub guys did is they introduced a plugin called GitHub Webhook to trigger.

So that means when somebody pushes the code, there will be one hook activated

and it'll go and tell the Jenkins that, "Hey, there is a push.

Github -> then from that repository level, go to Settings -> webhook -> Add a new webhook ->

So you need to give the Payload URL. So this payload URL is nothing but the URL where Jenkins is listening

go to your Jenkins and Manage Jenkins -> system -> github servers -> select the specify another hook url for github configuration -> copy the link ( This is the place where Jenkins can put its ear to listen about all webhooks coming from GitHub. ) - after you uncheck and don't save anything.

When pushing the code to this repository, this webhook will immediately inform to this link.

This is the place where our Jenkins is listening for all the webhooks from GitHub. So that's how both will communicate that, "Hey, something happened and they will go and trigger."

So now, if you think that this job should be automatically triggered after webhook event is received, then go to that job,

go to Configuration and Build Triggers-> select Githubhook trigger( whenever you receive a hook, trigger it,)

when you work in real time in your companies, instead of local host, you will have one server URL, A public IP address where you will provide it.

So one drawback with this webhook is you cannot practice in your local

So there is a software called ngrok. What the software does is, temporarily, it will make your private URL exposed to the public URL by providing a unique link. I mean, it redirects to one public URL, so that public URL, you can place it here

google -> ngrok -> signin -> download windows 64 bit -> unzip it-> Extract -> run the file

it will open one cmd then copy the command in ngrok and paste it here and run Our token, save to configuration file. Our ultimate goal here is to point our local IP, We want to point this to a public IP where I can put link here.

So again, copy this third command, run the command ( Please give me public IP address,

redirecting this one so that way, I can make that CI/CD.

And ngrok will do that for you.)

It is saying it is your request of your local is being forwarded to one public IP address. Copy the link paste it in browser.  Jenkins, what you have in your local system now can be accessed from the internet.

Replace the payload url. you know why I'm doing this? Because webhook does not accept local IP address.

3steps

Step one, went to our Jenkin job and we told to our job that make sure that when webhook trigger received, run this job.

Step two, we went to do our GitHub repository and created one webhook.

For that it ask us URL, that URL, we don't have public URL.

So that's why step three, we went downloaded ngrok software, used our local host,

got that public IP and put it in our webhook and nicely created and it gave us a ping.

Modify the code and in cmd -> go to the project path -> git status -> git add file path -> git commit -m “ hi” -> git push origin main -> Automatically job will trigger.