# CICD PIPELINE USING JENKINS WITH MANUAL JOBS CREATION

For continuous integration and continuous deployment the popular tool used by the Devops engineer is Jenkins.

To install Jenkins the specifications were dual core processor and 8 GB Ram.

And for installation of Jenkins need to have java installed

So installed java in azure server

Using the below commands.

**sudo apt update -y – to update the server if anything is in repositories.**

**sudo apt install openjdk-17\* -y -- to install java version 17.**

To install Jenkins we use below script which is available in Jenkins website.

A screenshot of a computer

Description automatically generated

Once the Jenkins is installed in the server it can be accessed through port 8080.

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

Every application such as terraform, docker, github, email, azure, etc… most of the tools has plugins so Jenkins install those plugins to able to perform the specified jobs where the commands given by user.

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

In Jenkins to perform any task we need to build a job which has all the setting can be configured such as need to write the code /script to perform the task so once if we perform build now the job is performed and we can also check the output.

We can schedule the job using corn format which can be generated by corntab guru.

For printing particular column in linux the command used is awk ‘{print $4}’

To get ram in mega bytes free -m

To get ram in Giga bytes free -g

We can build triggers remotely while configuring using the jenkins token.

The link is [**http://20.173.96.114:8080/job/job1//build?token=qdqedqedqedqds**](http://20.173.96.114:8080/job/job1//build?token=qdqedqedqedqds)for triggering the job remotely.

A screenshot of a chat

Description automatically generated

We can perform sonar analysis using Jenkins so whatever the software we want to work it should be installed in the system.

To generate the build when the code is pushed in the github then plugin is used and to update the status of the **build mailer plugin** is installed.

Once plugin is installed we were able to fetch the code from githun and send emails of build status.

To setup the email configuration we need to create apppassword for gmail so that it can be used I the configuration part.

A screenshot of a computer

Description automatically generated

Once the email is configured then we would get the emails when the job is done a=or when the job is failed.

A screenshot of a computer

Description automatically generated

We can fetch the code from github as mentioned below.\

A screenshot of a computer

Description automatically generated

The files which were fetched in the github were stored in folder workplace and now we can execute the sonarqube command so that the code analysis can be done using Jenkins.

Command used to run the sonarqube analysis:  
sudo docker run --rm -e SONAR\_HOST\_URL="http://20.173.96.114:9000" -v "$(pwd):/usr/src" sonarsource/sonar-scanner-cli -Dsonar.projectKey=jen3 -Dsonar.login=sqp\_ed85765f3a20c5b67a8a1059616c4bed374c4b0f -X

Which has authentication key, port ip address/url and configuration data.

A screenshot of a computer

Description automatically generated

Corresponding build output.

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

There were many options available to configure the job.

Allows whether to run the job in same server or another node/server.

A screenshot of a computer

Description automatically generated

To trigger the job when there is change in github repository.

A screenshot of a computer

Description automatically generated

Will build the other job once the above job runs successfully.

A close-up of a line

Description automatically generated

Another node is added so that we can deploy the job in that server which is connected using ssh.

A screenshot of a computer

Description automatically generated

A white lined paper with blue lines

Description automatically generated

Job used to build artifacts in nexus repository.

A close-up of a computer screen

Description automatically generated

Job used to download the artifacts into the server.

A screenshot of a computer

Description automatically generated

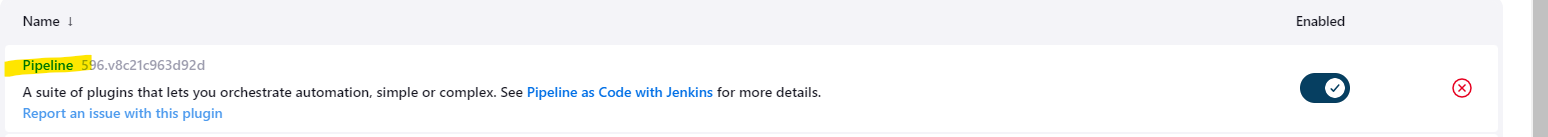
# Build CICD Pipeline using pipeline plugin.

Jenkins pipeline is used to automate the process of deployment and store the code in GitHub.

Whenever there is change in code automatically pipeline would be started and perform stages mentioned in pipeline.

Here I have practiced concept of pipeline by creating stages for SonarQube code analysis, building the artifacts for application, storing the artifacts into nexus, deploying the application into webserver where the artifacts are downloaded from nexus repositories.

Pipeline plugin is needed to create a pipeline job in Jenkins.



After installing the plugin then a pipeline is created as shown below by giving github repo and pipeline path.

A screenshot of a computer

Description automatically generated

So once it is configured started creating the jobs in the Jenkins file.

The Jenkins file is also stored in github where it has history of detailed commits and every step.

<https://github.com/saitejat1907/lms/tree/dev> --- GitHub repository.

A screenshot of a computer

Description automatically generated

Code which is used to perform sonar analysis and build the application.

A screen shot of a computer program

Description automatically generated

Code which is used to to store and deploy the artifacts including versioning which is mentioned in package.json file.

A computer screen shot of a program code

Description automatically generated

Once the code is committed everytime pipeline would be runn automatically and at last by solving all the errors which occurred able to deploy the application successfully.

A screenshot of a computer

Description automatically generated

SonarQube analysis:  
A screenshot of a computer

Description automatically generated

Nexus artifacts:  
A screenshot of a computer

Description automatically generated

Deploying the application:  
A blue rectangle with white border

Description automatically generated

Tested by changing the index file background colour so application deployed automatically without any human intervention all the jobs were doen automatically which were mentioned in the pipeline.

A screenshot of a computer

Description automatically generated

A red square with white text

Description automatically generated