Jankins pipeline codes:

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

1) git checkout:

```
pipeline{
agent any
stages{
stage('checkout'){
step{
git branch : 'master',
credentialID:'PAT'
url:github url which rep what to clone
}
}
}
}
```

2) git clone/ckeckout:

```
pipeline{
agent any
stages{

    stage('checkout'){
```

```
        steps{

            git credentialsId: 'newPAT', url: 'https://github.com/kanchana08/docker-compose-sample.git'

            }
}

      stage('docker build'){

         steps{

            sh 'docker build -t kanchana/new:v1 .'
}
 }
 }
}
```

3) to push the docker image to docker hub:

```
pipeline{

   agent any

   stages{

      stage('checkout'){

         steps{
```

```
            git credentialsId: 'newPAT', url: 'https://github.com/kanchana08/docker-compose-sample.git'

            }
}
stage('docker build'){

          steps{

            script{
sh 'docker build -t kanchana0812/projnew:v1 .'

            }
 }

      }

      stage('login') {

          steps{

            script{

                withCredentials([string(credentialsId: 'dockerhub_pwd', variable: 'dockerhub')]) {

                   sh 'docker login -u kanchana0812 -p ${dockerhub}'
}

                sh 'docker push kanchana0812/projnew:v1'
```

```
            }

          }

        }
    }

}
```

4)email notification:

```
pipeline{

  agent any

  stages{

    stage('checkout'){

      steps{

        git credentialsId: 'newPAT', url: 'https://github.com/kanchana08/docker-compose-sample.git'

        }
}

        stage('docker build'){
```

```
        steps{

            sh 'docker build -t kanchana/new:v1 .'



        }

    }



}



  post { ----------------------------------------------------> post is not  came to under stages

    success {

        mail to:"kanchanagm123@gmail.com", subject:"SUCCESS: ${currentBuild.fullDisplayName}",
body: "Yay, we passed."

    }

    failure {

        mail to:"kanchanagm123@gmail.com", subject:"FAILURE: ${currentBuild.fullDisplayName}",
body: "Boo, we failed."
```

```
        }

    }

}
```

For docker build:

```
pipeline {
    agent any
    environment {

        dockerImage = ''
    }

    stages{

     stage('git checkout'){

      steps {

            git credentialsId: 'newPAT', url: 'https://github.com/kanchana08/docker-compose-sample.git'

          }
```

```
    }

    stage('Building our image') {

        steps {


            sh 'docker system prune -af'
            sh 'docker build -t screenate:latest .'
            sh 'docker stop screenate || true && docker rm screenate || true'
        }
    }
    stage('Deploy Docker container') {

        steps {
            sh "docker run --rm -d --name screenate -p 4206:80 screenate:latest"

        }
    }
   }
    post{
      failure{
        mail to: "kanchanagm@stellaripl.com",
        subject: "jenkins build:${currentBuild.currentResult}: ${env.JOB_NAME}",
        body: "${currentBuild.currentResult}: Job ${env.JOB_NAME}\nMore Info can be found here: ${env.BUILD_URL}"

      }
```

```
    }
}
```

https://www.guru99.com/jenkins-github-integration.html#:~:text=How%20to%20Install%20Git%20Plugin%20in%20Jenkins%201,see%20your%20plugins%20listed%20among%20the%20rest.%20

Copy the file from one to anther folder

```
pipeline {

   agent {label 'windowsslave'}

  stages {

   stage('Git checkout') {

     steps {

      git branch: 'main', credentialsId: 'PAT', url: 'https://192.168.10.58/dev/dex-validation-wrapper.git'

     }

   }
   stage('Copy Files') {
    steps {
```

```
        bat 'xcopy F:\\Jenkins\\workspace\\Dex-validation-Wrapper\\*.* "E:\\Plant\\3-DE Batcher\\" /s /e
/h /y'
     }
   }

 }
 post{
      failure{
         mail to: "nadimpali.pradeepk@stellaripl.com",
         subject: "jenkins build:${currentBuild.currentResult}: ${env.JOB_NAME}",
         body: "${currentBuild.currentResult}: Job ${env.JOB_NAME}\nMore Info can be found here:
${env.BUILD_URL}"

      }
   }
}
```

Jenkins:

Jenkins is an open source, Java-based automation server that offers an easy way to

set up a continuous integration and continuous delivery (CI/CD) pipeline.

Continuous Integration (CI) is the process of automating the build and testing of

code every time a team member commits changes to version control. Continuous Delivery

(CD) is the process to build, test, configure and deploy from a build to a production environment.

Jenkins is a tool that is used for automation, and it is an open-source server that

allows all the developers to build, test and deploy software. It works or runs on java as

it is written in java. By using Jenkins, we can make a continuous integration of projects(jobs) or end-to-endpoint automation.

Steps to Install Jenkins:

Step 1: go to official jenkins website and click on downloads

https://jenkins.io

Step 2: Once we are on the Jenkins website, you will see the 'Download'

option available in the dashboard. There are 2 types of releases available.

Long-term support release

Weekly release

We will use the long term support link for the ubuntu platform

Follow provided command instructions to download and run jenkins.

Step 3: Bydefault jenkins runs with port 8080 . Search hostname/ip:8080

Step 4: Now the first thing to unlock Jenkins is ‘Password’; yes, we need to provide the initial admin password.

Directory: /var/lib/Jenkins/secrets/initialAdminPassword

Step 5: copy the following details and paste them into ubuntu server

[] Cat /var/lib/Jenkins/secrets/initialAdminPassword

[] 123jdsjfdsfmkf3

Step 6: Copy this password and paste it into the Jenkins window opened in the browser.

I have pasted my password and clicked Continue.

Step 7: Now, on the next screen, we will get the ‘Customize Jenkins’ screen.

Again, there are two options for the users to prefer.

Install suggested plugins. (I'll go with suggested plugins)

Select plugins to install.

We will choose the first option, as it is the most used list of plugins preferred

by the community members. Also, if we have any specific purpose, then only prefer the 2nd option.

With this, the recommended plugins will start to download.

Step 9: Fill the following fields:

Username

Password

Confirm Password

Full Name

E-mail address

Once the details are provided, click on the ‘Save and Finish’ button,

and we will get the screen. This means Jenkins is installed properly and it is ready to start.

to install the requied application-->we need to install the requied application to

clone the gitlab things this is the first procegure we need to do

[] jenkins

[] in bashboard

[] manage jenkins

[]manage plugins

[]available plugins

[] install without restart

for git clone using freestyle project

[] in dashboard

[] new item

[] enter an item name ---> type whatever name we want

[]selecte freestyle project

[] ok

[] in general----> description we can give whatever/optional

[] in source code manegement ----> git

[] in that [] repository URL ---> give github which ever project copy clone url and paste

[] credential ---> github username/token

[]save

[]build now

click on the project name select configure there we have build options

1)Trigger builds remotely :

If you want to trigger your project built from anywhere anytime then you should select

Trigger builds remotely option from the build triggers.

You'll need to provide an authorization token in the form of a string so that only those who

know it would be able to remotely trigger this project's builds.

This provides the predefined URL to invoke this trigger remotely.

[] trigger build remotely

[] Authentication token --> give whatever and copy the below the url in web past in my case i use local web for jenkins

localhost:8080/job/abcd/buildtoken= kanchana

[] enter

[] then check it aytomatically run

2)Build periodically:

If you want to schedule your project build periodically then you should select the Build periodically option from the build triggers.

You must specify the periodical duration of the project build in the scheduler field section

This field follows the syntax of cron (with minor differences). Specifically, each line consists of 5 fields separated by TAB or whitespace:

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.

To specify multiple values for one field, the following operators are available. In the order of precedence,

* specifies all valid values

M-N specifies a range of values

M-N/X or */X steps by intervals of X through the specified range or whole valid range

A,B,...,Z enumerates multiple values

Examples:

every fifteen minutes (perhaps at :07, :22, :37, :52)

H/15 * * * *

every ten minutes in the first half of every hour (three times, perhaps at :04, :14, :24)

H(0-29)/10 * * * *

once every two hours at 45 minutes past the hour starting at 9:45 AM and finishing at 3:45 PM every weekday.

45 9-16/2 * * 1-5

once in every two hours slot between 9 AM and 5 PM every weekday (perhaps at 10:38 AM, 12:38 PM, 2:38 PM, 4:38 PM)

H H(9-16)/2 * * 1-5

once a day on the 1st and 15th of every month except December

H H 1,15 1-11 *

3) GitHub webhook trigger for GITScm polling:

A webhook is an HTTP callback, an HTTP POST that occurs when something happens through a simple event-notification via HTTP POST.

GitHub webhooks in Jenkins are used to trigger the build whenever a developer commits something to the branch.

Let's see how to add build a webhook in GitHub and then add this webhook in Jenkins.

[]Go to your project repository.

[]Go to "settings" in the right corner.

[]Click on "webhooks."

[]Click "Add webhooks."

[]payload URL --> copy the jenkins url

[] gave jenkinsURL/github-webhook

[]go to content type

[] select--> application/json

[] go to which events would you like to trigger this webhook

[] select--> send me everything

[]active

[] update webhook ---> it build automaticaly if we change code it build automaticaly and in jenkins it show 1 commits

[] using pipeline code

ther is 2 type of code

1) scripted : here we use node

```
node {
stage('build'){
}
stage ('test'){
}
stage('deploy'){
}
}
```

2)Declaration :

most of the time we use this only

```
pipeline{
agent any
stages{
stage('build'){
steps{
}
}
stage('test'){
step{
}
}
```

---

1) pipeline code for git checkout

```
pipeline{
agent any
stages{
stage('checkout'){
step{
git branch : 'master',
credentialID:'PAT'
url:github url which rep what to clone
}
}
}
}
```

```
pipeline{
agent any
stages{
stage('checkout'){
step{
git branch : 'master',
credentialID:'PAT'
url:github url which rep what to clone
}
}
}
}
```

---

pipeline code for git clone/ckeckout

```
pipeline{

   agent any

   stages{

      stage('checkout'){

         steps{

            git credentialsId: 'newPAT', url: 'https://github.com/kanchana08/docker-compose-sample.git'

            }




      }

      stage('docker build'){

         steps{

            sh 'docker build -t kanchana/new:v1 .'



         }
```

```
        }


    }


}
```

---

pipeline code for pull the docker to dockerhub:

```
pipeline{

  agent any

  stages{

     stage('checkout'){

        steps{

           git credentialsId: 'newPAT', url: 'https://github.com/kanchana08/docker-compose-sample.git'

           }
```

```
        }

        stage('docker build'){

           steps{

               sh 'docker build -t kanchana0812/pro:v1 .'



           }

        }

        stage('Login') {



steps {

sh 'echo Kanchu@08 | docker login -u kanchana0812 --password-stdin'



}
```

```
}

stage('Push') {




steps {

sh 'docker push kanchana0812/pro:v1 '

}

}




      stage('logout') {

         steps {

            sh 'docker logout'

      }

      }

   }
```

}

---

this is the best way for pipeline code for pull the docker to dockerhub

```
pipeline{

   agent any

   stages{

      stage('checkout'){

         steps{

            git credentialsId: 'newPAT', url: 'https://github.com/kanchana08/docker-compose-sample.git'

            }



      }

      stage('docker build'){
```

```
    steps{
        script{




        sh 'docker build -t kanchana0812/projnew:v1 .'
        }


    }
}
stage('login') {
    steps{
        script{
            withCredentials([string(credentialsId: 'dockerhub_pwd', variable: 'dockerhub')]) {
                sh 'docker login -u kanchana0812 -p ${dockerhub}'
```

```
                }

                sh 'docker push kanchana0812/projnew:v1'

               }

            }

         }
   }

}
```

for creating docker credential

in pipeline syntax

in []sample step

[] withCredentials bind credential to veriable

in []binding

[]secret text

in that []variable--->whatever we want

[] credential

[]add

[]in kind

[]select--> secret text

[] in secret--->dockerhub password

[]in ID --->whatever we want

[]add

[]credential ----> add ID

[]Generate pipeline secret

[]finally we get code

[] withCredentials([string(credentialsId: 'dockerhub_pwd', variable: 'dockerhub')]) {

//some block}

[]in //some block ---> sh 'docker login -u kanchana0812(dockerhub repo ID) -p ${dockerhub}(variable:)

---

pipeline email notification

```
pipeline{

   agent any

   stages{

      stage('checkout'){

         steps{

            git credentialsId: 'newPAT', url: 'https://github.com/kanchana08/docker-compose-sample.git'
```

```
            }


        }

        stage('docker build'){

           steps{

              sh 'docker build -t kanchana/new:v1 .'


           }

        }


    }
```

post { ----------> **post is not came to under stages**

```
        success {
```

```
        mail to:"kanchanagm123@gmail.com", subject:"SUCCESS: ${currentBuild.fullDisplayName}", body: "Yay, we passed."

    }

    failure {

        mail to:"kanchanagm123@gmail.com", subject:"FAILURE: ${currentBuild.fullDisplayName}", body: "Boo, we failed."

    }

  }

}
```

E-mail notification :-

[]in jenkins dashboard

[]manage jenkins

[]manage plugins

[] available or installed

[] email ext plugins [] email ext plugins

[] in jenkins dashboard

[] manage jenkins

[] configure system

[] email notification

[]SMTP server

[] smtp.gamil.com

[]default user email suffix

[]@gamil.com

[]advance

[]user name

[]kanchana@gamil.com

[]password

[]select use SSL

we need set our email account

[]google account

[]security

[]input password as asked

[]turn Once(you could use SMS to get gmail code to activate 2-step)

[]google account

[]security

[]app password

[]input password as asked

[]select the app and device

[]example-->other (custom name)

[]input app name

[]jenkins

[]generate

[]copy a 16 -character password

[] use a 16 -char password with gmail

continu with jenkins

[] select [] use SSL

SMTP port

[]465

[]replay to address

[]kanchanagm123@gmail

[]test configuration by sending test email

[]test email recipient

[] kanchanagm123@gmail

[]test configuration ----> we get mail

---

extended email notification:

[]SMTP server

[] smtp.gamil.com

[]465

[]advance

[]credential

[]username

[]password

[]description

[]use SSL

[]in end default trigger

[]select always,failure_any , success

and email notification also important we need to check

[]in dashboard

[]job/project

[]configure

[]push build action

[]select add post build action

[]email notification

[]project recipient

[]give user email using comma we can add the user

---

creating user and permission

[]manage jenkins

[]manage user

[]create user

[]provide users name,password

[]manage jenkins

[]manage plugins

[]install role based authorization

[]after installing we get manage and assign roles under manage jenkins

[]manage jenkins

[]configure globle security

[]authentication

[]security realm

[]select jenkins own user datebase

[]authorization

[]role based stategy

[]saved

[]manage jenkins

[]manage and assign roles

[]manage roles

[]role to add

[]we can create like developer/frontend

[]save

[]assign roles

[]select the permission

---

email notification using outlook :

[] in jenkins dashboard

[] manage jenkins

[] configure system

[]under jenkins location

[]under system admine email address

[]jenkins<kanchana@stellaripl.com>

[]under extended email notification

[]SMTP server

[] smtp.office365.com

[]SMtp port

[]587

[]under advance

[]credential

[]add

[]username

[]password

[]add

[]select use TLS

[]default user email suffix

[]@Stellaripl.com

[]default content type

[]select plan text(text/plan)

[]under email notification

[]SMTP server

[]smtp.office365.com

[]default user email suffix

[]@stellaripl.com

[]under advanced

[]use SMTP authentication

[]username

[]password

[]select use TLS

[]SMTP port

[]587

[]test configuration by sending test email

[]test email recipient ---->kanchana@stellaripl

[]test configuration--->we get mail

then

[]GO to job/project

[]configuration

[]post build actions

[]in add

[]select editable email notification

[]project recipient list

[]kanchana@stellaripl ----> using comma we can add

[]save