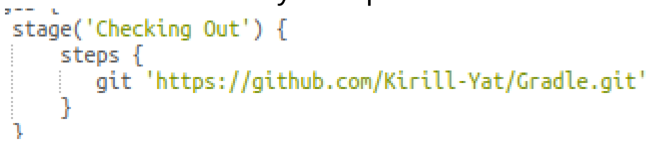
Task 2. Pipeline Plugin

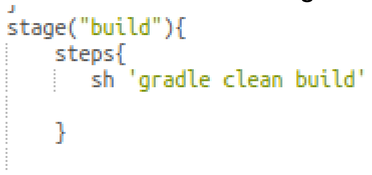
1) Install Pipeline plugin on the local Jenkins.
2) Create Jenkinsfile
3) The Scripted **Jenkinsfile** are stored in your github repo (root location in repo), Each student commits all step-by-step changes in own branch. Initial source = 'master' branch
4) Requirements for the Pipeline:
   1) 'Preparation (Checking out)'

      Checkout code from your repo



```
stage('Checking Out') {
    steps {
        git 'https://github.com/Kirill-Yat/Gradle.git'
    }
}
```

   2) 'Building code'. Should contain building code stage



```
stage("build"){
    steps{
        sh 'gradle clean build'

    }
```

   gradle *build*

   3) 'Testing code'

   Should contain **3 parallel execution** of tests

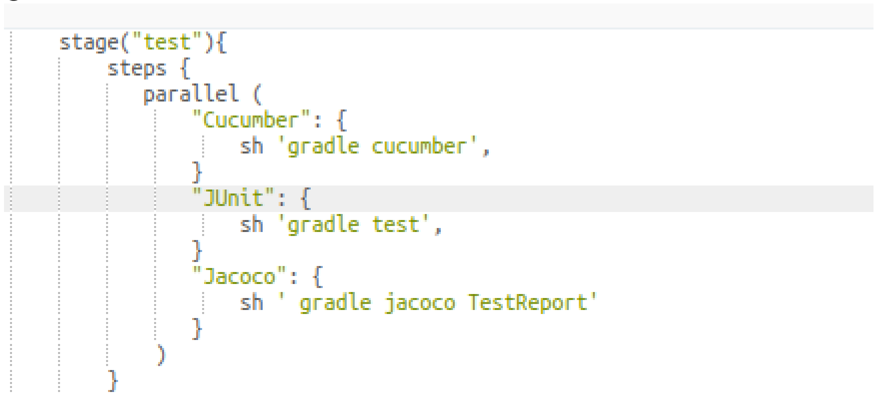
   Stages: 'Unit Tests', 'Jacoco Tests', 'Cucumber Tests'

   Additional commands:

   gradle *cucumber* – performs Cucumber tests

   gradle *jacocoTestReport* - Generates code coverage report for the test task.

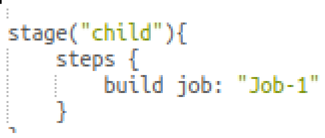
   gradle *test* - Runs the unit tests.



```
stage("test"){
    steps {
        parallel (
            "Cucumber": {
                sh 'gradle cucumber',
            }
            "JUnit": {
                sh 'gradle test',
            }
            "Jacoco": {
                sh ' gradle jacoco TestReport'
            }
        )
    }
```

   4) 'Triggering child job'

   Pipeline should wait for finishing triggered job.



```
stage("child"){
    steps {
        build job: "Job-1"
    }
```

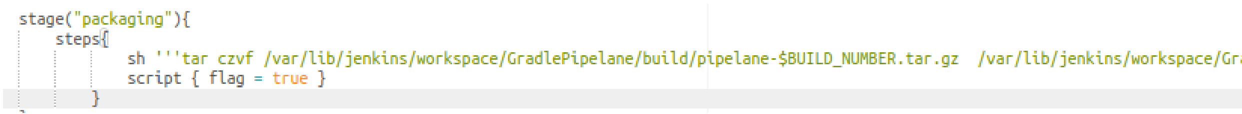
   5) 'Packaging and Publishing results'

   On this stage the job should:

   a) take:

   - '*Jenkinsfile*'

   b) Should create new artefact '**pipeline-{buildNumber}.tar.gz**' (where *buildNumber* - number of the current build)

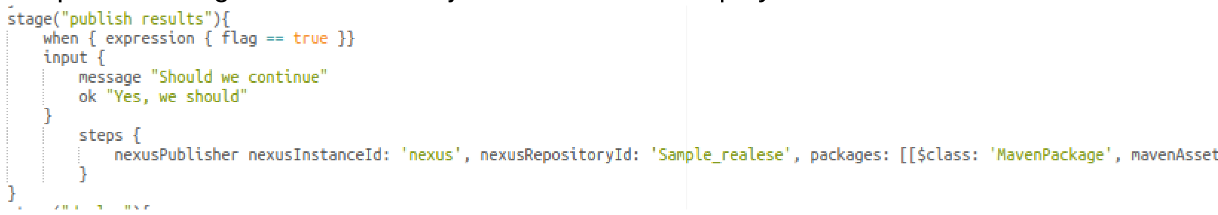


```
stage("packaging"){
    steps{
            sh '''tar czvf /var/lib/jenkins/workspace/GradlePipelane/build/pipelane-$BUILD_NUMBER.tar.gz  /var/lib/jenkins/workspace/Gr
            script { flag = true }
        }
```

c) Should attach this artefact to current job and Push to Nexus

6) ‘Asking for manual approval’

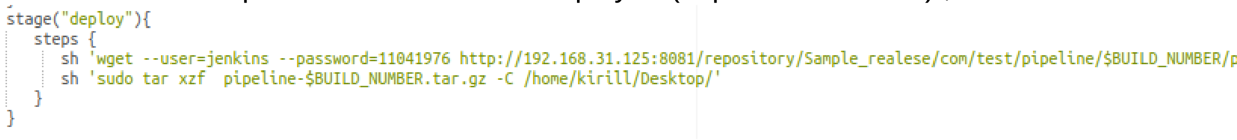
Once previous stage successful the job should ask for deployment this artefact.



```
stage("publish results"){
    when { expression { flag == true }}
    input {
        message "Should we continue"
        ok "Yes, we should"
    }
        steps {
            nexusPublisher nexusInstanceId: 'nexus', nexusRepositoryId: 'Sample_realese', packages: [[$class: 'MavenPackage', mavenAsset
        }
}
```

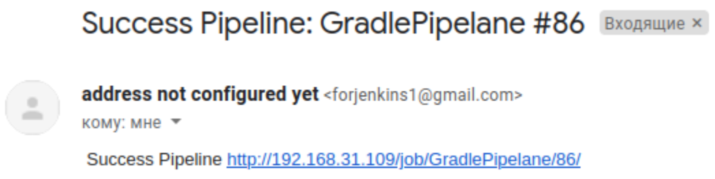
7) 'Deployment'

Artefact should be pulled from Nexus and deployed (unpack in our case) ;



```
stage("deploy"){
   steps {
      sh 'wget --user=jenkins --password=11041976 http://192.168.31.125:8081/repository/Sample_realese/com/test/pipeline/$BUILD_NUMBER/p
      sh 'sudo tar xzf  pipeline-$BUILD_NUMBER.tar.gz -C /home/kirill/Desktop/'
   }
}
```

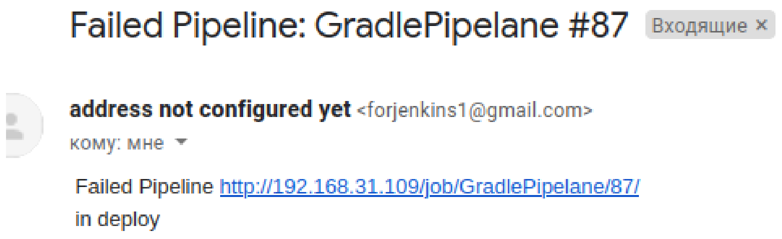
8) If pipeline failed – send email with name of stage build failed on. If pipeline status is ‘Success’ – send final email as well.



Success Pipeline: GradlePipelane #86 Входящие ×

address not configured yet <forjenkins1@gmail.com>
кому: мне

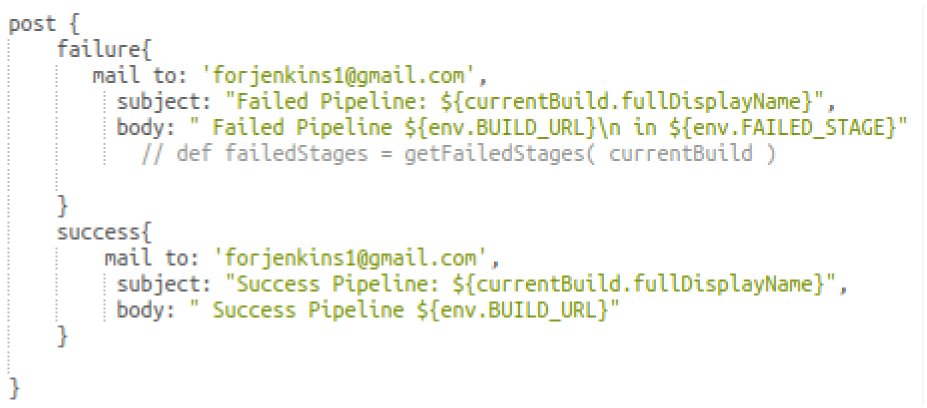
Success Pipeline http://192.168.31.109/job/GradlePipelane/86/



Failed Pipeline: GradlePipelane #87 Входящие ×

address not configured yet <forjenkins1@gmail.com>
кому: мне

Failed Pipeline http://192.168.31.109/job/GradlePipelane/87/
in deploy



```
post {
    failure{
       mail to: 'forjenkins1@gmail.com',
         subject: "Failed Pipeline: ${currentBuild.fullDisplayName}",
         body: " Failed Pipeline ${env.BUILD_URL}\n in ${env.FAILED_STAGE}"
           // def failedStages = getFailedStages( currentBuild )

    }
    success{
        mail to: 'forjenkins1@gmail.com',
         subject: "Success Pipeline: ${currentBuild.fullDisplayName}",
         body: " Success Pipeline ${env.BUILD_URL}"
    }

}
```