Jenkinsfile_ci for project_Next

1- stage : checkout

A- Description:

- the source code management can be used to manage a projects source code
- in the stage checkout jenkins will clone the repository in the branch main using url and credantielID (essential for integration between gitlab and jenkins) with a commitID
- and declare it in a variable " COMMIT_ID "
- jenkins can be check a repository for every changes

B-Code:

```
stage('Checkout') {
    steps {
      script {
        checkout([$class: 'GitSCM',
          branches: [[name: "main"]],
          doGenerateSubmoduleConfigurations: false,
          extensions: [],
          submoduleCfg: [],
          userRemoteConfigs: [[credentialsId: 'wiss-git', url:
'https://gitlab.com/wissemsghaier2000/wissem.git']]
        ])
        sh "git rev-parse --short HEAD > commit_hash.txt"
        COMMIT_ID = readFile('commit_hash.txt').trim()
        echo "Commit Hash: ${COMMIT_ID}"
      }
    }
  }
```

2- stage : install package

A- Description:

- in this stage allow to manage dependencies for next
- npm install : download all package who developers used and install of the project.

B-Code :

```
stage ('istall package ') {
    steps {
      nodejs(nodeJSInstallationName: 'NodeJS'){
        sh ' npm install'
     }
    }
  }
```

3- stage : build

A- Description:

- npm run build allow to compile your code and create directory in racine directory for performance and generate ready build that can be deploy to a web server

B-Code :

```
stage ('build ') {
    steps {
      nodejs(nodeJSInstallationName: 'NodeJS'){
        sh " npm run build"
      }
    }
```

4- stage : test

A- Description:

- this stage alow to run unit test (test individual methods or functions)
- npm run test allow to run alll taest in directory __test__

B-Code :

```
stage ('test') {
    steps{
      nodejs(nodeJSInstallationName: 'NodeJS'){
        sh " npm run test "
      }
    }
```

5- stage : Code Quality Check via SonarQube

A- Description:

- this stage is responsible for checking code using sonarqube (it is a tools to analyzes the quality of source code)
- The scannerHome variable is set to the SonarQube scanner tool
- withSonarQubeEnv is variabble to connect to SonarQube server using url and token
- run SonarQube scanner with the project name, project key, and source code location
- when SonarQube scanner finished analysis the results are sent to the SonarQube server

B-Code :

```
stage('Code Quality Check via SonarQube') {
    steps {
      script {
        def scannerHome = tool 'sonarqube-scanner';
        withSonarQubeEnv("SonarQube") {
          sh "${tool("sonarqube-scanner")}/bin/sonar-scanner \
          -Dsonar.projectName=solutions \
          -Dsonar.projectKey=solutions\
          -Dsonar.sources=. \
          -Dsonar.host.url=http://172.20.0.1:9001/  \
          -Dsonar.exclusions=**/node_modules/** \
          -Dsonar.language=js \
          -Dsonar.login=squ_2077a7e9cbf7524524a713696e19787ef2c747d2"
        }
      }
    }
```

6- stage : Quality Gate

A- Description:

- The waitForQualityGate used to pause the pipeline until the SonarQube analysis is completed
- if the quality gate failure indicate the code is not desired quality standards and should not be deployed

B-Code :

```
 stage("Quality Gate") {
       steps {
         sleep 60
         waitForQualityGate abortPipeline: true
       }
     }
```

7- stage : Publish to Nexus Repository Manager

A- Description:

- nexus is open source and can be push to nexus by code and docker image
- zip project_laravel because
- upload file in repositoryID with tag commitID

B:Code :

```
stage(" Publish to Nexus Repository Manager") {
      steps {
        sh "apt-get install zip"
        sh "zip -r project_nextjs.zip ."
        sh "curl -v -u wissem:wissem --upload-file project_nextjs.zip
http://172.20.0.10:8081/repository/maven-releases/next/quantum/soluttions/next-js/$
{COMMIT_ID}/next-js-${COMMIT_ID}.zip"
      }
}
```

8- stage : cleanWS

A- Description:

- used in jenkins pipeline to clean workspaces after finiching the latest build (deleting all files created during the build)
- post : the workspace always be cleaned upbuild completed

B-Code :

```
post {
     always {
       cleanWs()
     }
   }
 }
```

jenkinsfile :

```
def COMMIT_ID
 pipeline {
   agent any
   stages {
     stage ("started "){
       steps {
         slackSend channel: "#jenkins-alerts-pfe-2023", message: " STARTED:job  '$
{env.JOB_NAME} ${env.BUILD_NUMBER}' (${env.BUILD_URL})"
       }
     }
     stage('Checkout') {
       steps {
         script {
           checkout([$class: 'GitSCM',
             branches: [[name: "main"]],
             doGenerateSubmoduleConfigurations: false,
             extensions: [],
             submoduleCfg: [],
             userRemoteConfigs: [[credentialsId: 'jenkins-scm', url:
'https://gitlab.quantum-solutions.xyz/devops/pfe-2023/project_laravel.git']]
           ])
           sh "git rev-parse --short HEAD > commit_hash.txt"
           COMMIT_ID = readFile('commit_hash.txt').trim()
           echo "Commit Hash: ${COMMIT_ID}"
         }
       }
       post {
         success {
           slackSend (color: 'good', message: "checkout of pipeline  succeeded!")
         }
         failure {
           slackSend (color: 'danger', message: "checkout  of pipeline  failed!")
         }
       }
     }
     stage('install packages') {
       steps {
         //sh "composer update"
         sh "composer install"

       }
       post {
         success {
```

```
      slackSend (color: 'good', message: "install packages of pipeline  succeeded!")
    }
    failure {
      slackSend (color: 'danger', message: "install packages of pipeline  failed!")
    }
  }
}

stage ('pre-configure'){
  steps{
    sh 'cp .env.example .env'
    sh 'php artisan key:generate'
  }
  post {
    success {
      slackSend (color: 'good', message: "pre-configure of pipeline  succeeded!")
    }
    failure {
      slackSend (color: 'danger', message: "pre-configure of pipeline  failed!")
    }
  }
}
stage('test') {
  steps {
    sh "php artisan test "
    sh './vendor/bin/phpunit'
  }
  post {
    success {
      slackSend (color: 'good', message: "Test of pipeline  succeeded!")
    }
    failure {
      slackSend (color: 'danger', message: "Test of pipeline  failed!")
    }
  }
}
stage('Code Quality Check via SonarQube') {
  steps {
    script {
      def scannerHome = tool 'sonarqube-scanner';
      withSonarQubeEnv("SonarQube") {
        sh "${tool("sonarqube-scanner")}/bin/sonar-scanner \
        -Dsonar.projectName=project_laravel122\
        -Dsonar.projectKey=project_laravel122 \
        -Dsonar.sources=. \
```

```
            -Dsonar.host.url=http://172.20.0.1:9001/  \
            -Dsonar.login=squ_e798c50ee42eb24e478af95e51491fd9e75bcc5e"
          }
        }
      }
      post {
        success {
          slackSend (color: 'good', message: "Code Quality of pipeline  succeeded!")
        }
        failure {
          slackSend (color: 'danger', message: "Code Qualityof pipeline  failed!")
        }
      }
    }
    stage("Quality Gate") {
      steps {
        sleep 60
        waitForQualityGate abortPipeline: true
      }
      post {
        success {
          slackSend (color: 'good', message: "Quality Gate of pipeline  succeeded!")
        }
        failure {
          slackSend (color: 'danger', message: "Quality Gate of pipeline  failed!")
        }
      }
    }
    stage(" Publish to Nexus Repository Manager") {
      steps {
        sh "apt-get install zip"
        sh "zip -r project_laravel.zip ."
        sh "curl -v -u wissem:wissem --upload-file project_laravel.zip
http://172.20.0.10:8081/repository/maven-releases/quantum/solutions/io/next-gen-
radio/${COMMIT_ID}/next-gen-radio-${COMMIT_ID}.zip"
      }
      post {
        success {
          slackSend (color: 'good', message: " Publish to Nexus of pipeline
succeeded!")
        }
        failure {
          slackSend (color: 'danger', message: " Publish to Nexus of pipeline  failed!")
        }
      }
```

```
    }
  }
  post {
    always {
      cleanWs()
    }
    success {
      slackSend channel: '#jenkins-alerts-pfe-2023', message: " build $
{currentBuild.result} for job ${env.JOB_NAME} #${env.BUILD_NUMBER}
(duration: ${currentBuild.durationString}). Check out the build at $
{env.BUILD_URL}"
      emailext (
        to: 'wissemsghaier2000@gmail.com',
        subject: "Job '${env.JOB_NAME}'",
        body: "build ${currentBuild.result} for job  ${env.JOB_NAME} at  $
{env.BUILD_URL} and the build number $BUILD_NUMBER"
      )
    }
    failure {
      slackSend channel: '#jenkins-alerts-pfe-2023', message: "Build $
{currentBuild.result} for ${env.JOB_NAME} #${env.BUILD_NUMBER} (duration:
${currentBuild.durationString}). Check out the build at ${env.BUILD_URL}"
      emailext (
        to: 'wissemsghaier2000@gmail.com',
        subject: "Job '${env.JOB_NAME}'",
        body: "build ${currentBuild.result} for job  ${env.JOB_NAME} at  $
{env.BUILD_URL} and the build number $BUILD_NUMBER"
      )
    }
  }
 }
```