**HOW TO CREATE JOBS ON JENKINS (FREESTYLE & PIPELINE)**

**STEPS TO CREATE A FREE STYLE JOB ON JENKINS: (Check About Jenkins Docs for detailed steps)**

* Click on New item on the left pane, and click create a new job(item name=Build, template=Freestyle project, click ok below)
* it would open a page, insert:
* Description=vprofile maven build project,
* scroll down to source code management
* click to check git (repository URL=paste the clone link from github)
* ignore any other part like credentials if it is a public repository, but if it is a private repository give the credentials
* scroll down to Branches to build (branch specifier=\*/(branch name in github containing the artifact)
* scroll down to build steps and click on add build steps
* (select: execute shell=insert your shell scripts like: mvn install) OR:

(select Invoke top-level Maven targets, Maven Version=Select the version installed: MAVEN3, Goals=install)

* professionally check other options, scroll down and click on save
* Click on build now on the left pane, you should see the action below on the same left pane, click on it to view and monitor the console output
* click on workspace on the left pane,
* click on the target directory and you will see the artifact. You can click on it to download and save.
* The workspace is not a safe place to store the artifacts becos you wipe out for new jobs. So click on Configure on the left pane, scroll down to post-build actions,
* click the drop down and select archive the artifacts, files to archive=\*\*/\*.war
* scroll down and click save at the end
* run the build now again on the left pane
* refresh the page (F5) and you should see details of the successful archive
* when you wipe out the workspace for a new job, you will still see the archived artifact

HOW TO CONFIGURE VERSIONING FOR RESTORE PURPOSE

* If you prefer versioning, remove the archive feature becos it comsumes more disk space maybe default to pick becos you will many saved names with nos
* So click on Configure on the left pane, scroll down to post-build actions, click the drop down and select archive the artifacts, files to archive=delete these --- \*\*/\*.war,
* Then scroll up a bit to build section, click on add build step and select execute shell:

mkdir –p versions

cp target/vprofile-v2.war versions/vprofile-V$BUILD\_ID.war

OR: (Through Parameter)

We can version our build by going to configuration, below description check the box for This project is parameterized, click add parameter, string parameter(Name=VERSION, scroll down to Execute shell=

mkdir –p versions

cp target/vprofile-v2.war versions/vprofile-V$VERSION.war, then scroll down and click on save)

* Click on Build with parameters on the left pane, then insert the version no you wish and click build

OR: (Through Plugins)

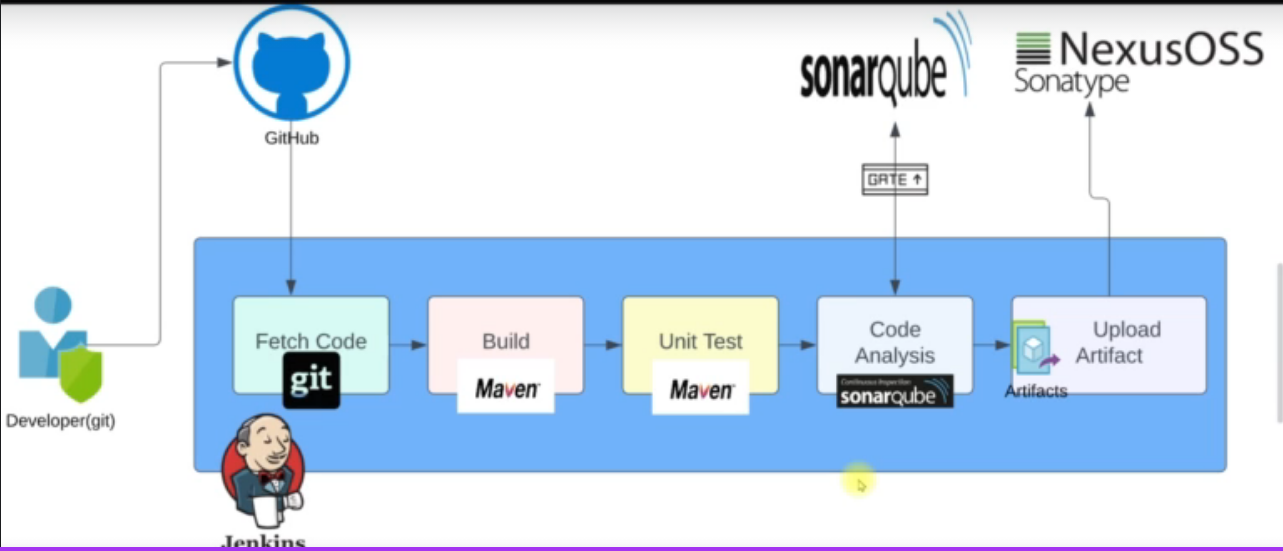
* Click on Dashboard, and then click on manage Jenkins (powerhouse of Jenkins), click on manage Plugins on the right.
* click on Available and search for timestamp plugin, then click on zentimestamp and click install without restart
* go to versioning-Builds on the right and then click on configure on the left pane, below description, remove: This project is parameterized, then click the check box for: change date pattern for the BUILD\_TIMESTAMP, type (yy-MM-dd\_HHmm) or click on the ? to know the format, scroll down to Execute shell=

mkdir –p versions

cp target/vprofile-v2.war versions/vprofile-V$BUILD\_ID- $BUILD\_TIMESTAMP.war, then scroll down and click on save)

* Click on Build on the left pane, then insert the version no you wish and click build

**The Flow of Continuous Integration Pipeline:**



**STEPS TO CREATE A PIPELINE JOB ON JENKINS:**

**The Flow of Execution:**

1. Jenkins Setup / Nexus Setup / Sonarqube Setup & Security Groups for each
2. Plugins for CI
3. Integrate (Nexus, Sonarqube)
4. Write the Pipeline script
5. Set notification

* **JENKINS SETUP / NEXUS SETUP /SONARQUBE**
* Launch an EC2 instance

|  |  |  |
| --- | --- | --- |
| **JENKINS SETUP** | **NEXUS SETUP** | **SONARQUBE** |
| Name = jenkinsServer | NexusServer | sonarServer |
| AMI = Ubuntu 20 | Centos7 | Ubuntu 18 |
| Instance type = t2 small  We need more config becos of the services | T2 medium | T2 medium |
| Keypair = Jenkinskey | Nexuskey | Sonarkey |
| Security Group (Jenkins-SG) =  ssh/TCP/22/from my IP  Custom TCP/TCP/port 8080/from anywhere  Custom TCP/TCP / port 80/ from anywhere (This is for Sonar to connect to Jenkins) | Nexus-SG  Ssh/TCP/22/from my IP  Custom TCP/TCP/port 8081/from anywhere | Sonar-SG  ssh/from my IP  Custom TCP / port 80/ from anywhere (This is for Nginx to route to port 9000 for sonar.)  Custom TCP / port 9000/ from anywhere |
| Provision using the Jenkins script | Provision using the Nexus script | Provision using the Sonar script |
| Click on launch | Click on launch | Click on launch |
|  |  |  |

* **SSH to each instance to check if they are running / CONFIGURATION STEPS:**

**JENKINS**

**(check the Doc ABOUT JENKINS for detailed steps to configure Jenkins)**

**To exit from an instance in git bash**

* logout
* close

NEXUS SERVICE

* ssh –i Downloads/ nexuskey.pem centos7@IP
* sudo –i
* sudo systemctl status nexus
* If any of the services isn’t running, run that particular service manually: ./nexus-setup.sh
* Get the public IP/port (e.g. IP:8081) of the instance and paste it in your browser and click on sign in to get the password path
* go to git bash and run: cat (password path) ………copy the password
* go to the browser running Nexus and paste to get access (Username = admin)
* configure your new password, click next and disable anonymous access for real life scenario. For practice enable it.

**To exit from an instance in git bash**

* logout
* close

SONARQUBE SERVICE

* ssh –i Downloads/ sonarkey.pem ubuntu@IP
* sudo –i
* sudo systemctl status sonarqube
* If any of the services isn’t running, run that particular service manually: ./ sonar-setup.sh
* Get the public IP/port (e.g. IP:80) of the instance and paste it in your browser and login with (username / password = admin / admin)

**To exit from an instance in git bash**

* logout
* close

1. **PLUGINS FOR CI**

Nexus

Sonarqube

Git

Pipeline Maven Integration Plugin

BuildTimestamp

Slack Notification

**Steps:**

* Login to Jenkins through the browser, click on manage Jenkins, then click on manage plugins
* click on available tab and search for and check the box each of:

Nexus Artifact uploader

SonarQube Scanner

Build Timestamp

Pipeline Maven Integration

Pipeline Utility steps

Slack Notification

Git (sometimes it is installed by default) …………….CLICK ON INSTALL WITHOUT RESTART

1. Writing the Pipeline script using Vscode editor:

def COLOR\_MAP = [

    'SUCCESS': 'good',

    'FAILURE': 'danger',

]

pipeline {

    agent any

    tools {

        maven "MAVEN3"

        jdk "OracleJDK8"

    } # tools

    stages {

        stage('Fetch code') {

            steps {

                git branch: 'vp-rem', url: 'https://github.com/devopshydclub/vprofile-repo.git'

            } #steps

        } # stage

        stage('Build'){

            steps{

                sh 'mvn install -DskipTests'

            } #steps

            post {

                success {

                    echo 'Now Archiving it......'

                    archiveArtifacts artifacts: '\*\*/target/\*.war'

                } # success

            } # post

        } # stage

        stage('UNIT TEST') {

            steps{

                sh 'mvn test'

            } # steps

        } # stage

        stage('Checkstyle Analysis'){

            steps {

                sh 'mvn checkstyle:checkstyle'

            }

        } # stage

        stage('Sonar Analysis') {

            environment {

                scannerHome = tool 'sonar4.7'

            } # environment

            steps {

                withSonarQubeEnv('sonar') {

                    sh '''${scannerHome}/bin/sonar-scanner -Dsonar.projectKey=vprofile \

                    -Dsonar.projectName=vprofile \

                    -Dsonar.projectVersion=1.0 \

                    -Dsonar.sources=src/ \

                    -Dsonar.java.binaries=target/test-classes/com/visualpathit/account/controllerTest/ \

                    -Dsonar.junit.reportsPath=target/surefire-reports/ \

                    -Dsonar.jacoco.reportsPath=target/jacoco.exec \

                    -Dsonar.java.checkstyle.reportPaths=target/checkstyle-result.xml'''

                } # withSonarQubeEnv

            } # steps

        }

        stage("Quality Gate") {

            steps {

              timeout(time: 1, unit: 'HOURS') {

                // Parameter indicates whether to set pipeline to UNSTABLE if Quality Gate fails

                // true = set pipeline to UNSTABLE, false = don't

                waitForQualityGate abortPipeline: true

              }

            }

        }

        stage("UploadArtifact") {

            steps{

                nexusArtifactUploader(

                    nexusVersion: 'nexus3',

                    protocol: 'http',

                    nexusUrl: 'private Ip of Nexus instance:8081',      don't put http, as it is in protocol

                    groupId: 'QA',

                    version: "${env.BUILD\_ID}-${env.BUILD\_TIMESTAMP}",

                    repository: 'the name of the repository created in Nexus server(Example:  vprofile-repo)',

                    credentialsId: 'nexuslogin',

                    artifacts: [

                        [artifactId: 'give the name of the artifact',

                        classifier: '',

                        file: 'target/vprofile-v2.war',

                        type: 'war']

                    ]

                )

            }

        }

    } # stages

    post {

        always {

            echo 'slack Notifications.'

            slackSend channel: '#jenkinscicd',

                color: COLOR\_MAP[currentBuild.currentResult],

                message: "\*${currentBuild.currentResult}:\* Job ${env.JOB\_NAME} build ${env.BUILD\_NUMBER} \n More info at: ${env.BUILD\_URL}"

        } #always

    } #post

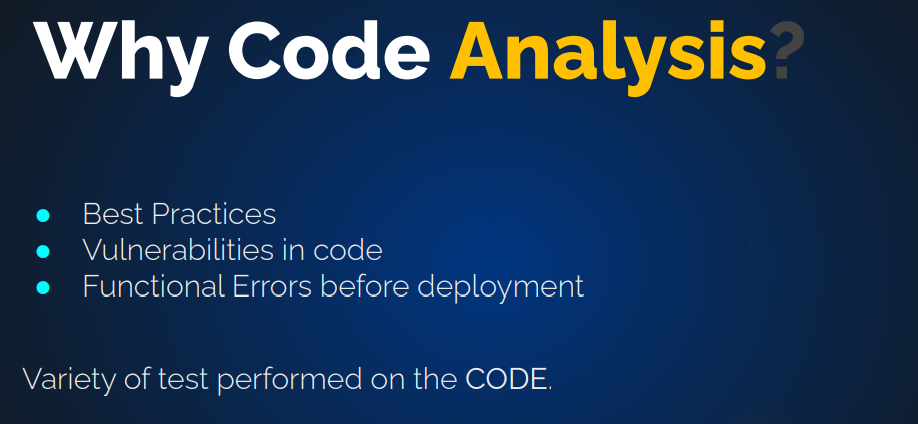
} # pipeline

* copy the pipeline code, login to Jenkins through your browser, click on new item (item name = sample-paac, then click on pipeline, and click ok)
* scroll down to the pipeline part and paste your code under script

OR:

* you can click the drop down and select git, then use professionally. I advice you use above
* click on save and click on build now on the left pane
* when the build is completed, click on the blue circle below build history on the left pane





1. **Integrating Sonarqube with Jenkins:**

* Let start with configuring Sonarqube, by getting your Sonarqube server up and running by signing in on the browser
* Open Jenkins with the browser,
* click on manage Jenkins and click on global tool configuration
* Scroll down to Sonarqube Scanner and click on add (name = sonar4.7, scroll down and click on save) take note of the name given becos it’ll be used in the pipeline code
* To Integrate Sonarqube, click on configure system,
* scroll down to Sonarqube servers if you don’t see it install the plugin,
* click to check environment variables,
* Under SonarQube installations: ….. Name: sonar, server url: (private IP of sonarserver from EC2 instance)
* If you want to use public IP, make sure security group allows port 80 on sonar from anywhere or Jenkins public IP
* for the token go to the sonarqube on the browser and click on the alphabet in a box on the right top of the page beside the search box
* click on my account, and click on security tab
* type jenkins and click on generate, copy the token and paste the token in a word file and save so as you don’t loose it, scroll down and click on save. )
* Click manage Jenkins, then click on configure system, the server authentication token should be active for clicking,
* click on add, click the drop down for kind and select secret text,
* paste the copied token in secret box,
* ID / Description = MySonarToken, click add and then click save.
* Write the pipeline, copy and then paste in the

def COLOR\_MAP = [

    'SUCCESS': 'good',

    'FAILURE': 'danger',

]

pipeline {

    agent any

    tools {

        maven "MAVEN3"

        jdk "OracleJDK8"

    } # tools

    stages {

        stage('Fetch code') {

            steps {

                git branch: 'vp-rem', url: 'https://github.com/devopshydclub/vprofile-repo.git'

            } #steps

        } # stage

        stage('Build'){

            steps{

                sh 'mvn install -DskipTests'

            } #steps

            post {

                success {

                    echo 'Now Archiving it......'

                    archiveArtifacts artifacts: '\*\*/target/\*.war'

                } # success

            } # post

        } # stage

        stage('UNIT TEST') {

            steps{

                sh 'mvn test'

            } # steps

        } # stage

        stage('Checkstyle Analysis'){

            steps {

                sh 'mvn checkstyle:checkstyle'

            }

        } # stage

        stage('Sonar Analysis') {

            environment {

                scannerHome = tool 'sonar4.7'

            } # environment

            steps {

                withSonarQubeEnv('sonar') {

                    sh '''${scannerHome}/bin/sonar-scanner -Dsonar.projectKey=vprofile \

                    -Dsonar.projectName=vprofile \

                    -Dsonar.projectVersion=1.0 \

                    -Dsonar.sources=src/ \

                    -Dsonar.java.binaries=target/test-classes/com/visualpathit/account/controllerTest/ \

                    -Dsonar.junit.reportsPath=target/surefire-reports/ \

                    -Dsonar.jacoco.reportsPath=target/jacoco.exec \

                    -Dsonar.java.checkstyle.reportPaths=target/checkstyle-result.xml'''

                } # withSonarQubeEnv

            } # steps

        }

        stage("Quality Gate") {

            steps {

              timeout(time: 1, unit: 'HOURS') {

                // Parameter indicates whether to set pipeline to UNSTABLE if Quality Gate fails

                // true = set pipeline to UNSTABLE, false = don't

                waitForQualityGate abortPipeline: true

              }

            }

        }

        stage("UploadArtifact") {

            steps{

                nexusArtifactUploader(

                    nexusVersion: 'nexus3',

                    protocol: 'http',

                    nexusUrl: 'private Ip of Nexus instance:8081',      don't put http, as it is in protocol

                    groupId: 'QA',

                    version: "${env.BUILD\_ID}-${env.BUILD\_TIMESTAMP}",

                    repository: 'the name of the repository created in Nexus server(Example:  vprofile-repo)',

                    credentialsId: 'nexuslogin',

                    artifacts: [

                        [artifactId: 'give the name of the artifact',

                        classifier: '',

                        file: 'target/vprofile-v2.war',

                        type: 'war']

                    ]

                )

            }

        }

    } # stages

    post {

        always {

            echo 'slack Notifications.'

            slackSend channel: '#jenkinscicd',

                color: COLOR\_MAP[currentBuild.currentResult],

                message: "\*${currentBuild.currentResult}:\* Job ${env.JOB\_NAME} build ${env.BUILD\_NUMBER} \n More info at: ${env.BUILD\_URL}"

        } #always

    } #post

} # pipeline

* copy the pipeline code, login to Jenkins through your browser,
* click on new item (item name = sample-paac, then click on pipeline, and click ok)
* scroll down to the pipeline part and paste your code under script, click save.
* click on build now,
* head over to the sonarqube in the browser to view your analysis.
* Refresh the page of the sonarqube browser if opened for long
* click on quality gates tab up,
* click on create on the left pane up (Name = vprofile-QG, click on save),
* click add condition on the right
* click to check on overall code,
* click on the quality Gate fails when drop down and select bug, value = 100 and click add condition)
* Go to your project (vprofile) showing on the sonarqube in your browser
* click project settings on the right,
* click Quality Gate, then select your quality gate (vprofile-QG)
* click project settings and click on web hooks,
* click on create (Name = jenkins-Ci-webhook,
* url = <http://(jenkins> public IP):8080/sonarqube-webhook, then click on create.
* click on build now, head over to the sonarqube in the browser to view your analysis if it is passed or failed.

**UPLOAD ARTIFACTS IN THE NEXUS REPOSITORY**

Storage location for our softwares

* Get the public IP of the Nexus instance, put it in your browser to access it. (public IP:8081)
* Login with the username = admin and pass = (enter the password you set when configuring the Nexus service)
* Click on settings and click on repositories on the left pane,
* click on create repository.
* look for maven2(hosted) and click on it (Name = vprofile-repo, scroll down and click on create repository.
* Please note if you want to store the artifact, you use hosted. To download you use proxy. Group is to group the repositories together
* Go to Jenkins service in the browser, click on manage Jenkins,
* scroll down to manage credentials, click on Jenkins below “stores scoped to Jenkins” and click on global credentials,
* look to the left pane and click on add credentials (username = admin, password = your set password, ID = nexuslogin, then click ok)
* Go to Jenkins Dashboard, click on manage Jenkins
* click on configure system,
* scroll down to Build Timestamp (check to enable BUILD\_TIMESTAMP, timezone = Etc/UTC, Pattern = yy-MM-dd\_HH-mm

SLACK NOTIFICATION FOR CODE PASS OR FAILURES

* Make sure you confirm you downloaded Slack plugin in Jenkins during setup
* Create and login to your slack account (slack.com)
* click on create workspace (Name = vprofilecicd, click on next),
* what you are working on = devopscicd,
* provide email addresses of people to add to your workspace, they would get a link.
* click on add channels,
* create a new channel ( Name = jenkinscicd, enable the bar for automatically add anyone who joins, click save)
* search for add apps to slack in google,
* click on the found link from slack and search for Jenkins CI,
* select it and click on add to slack,
* choose the channel: jenkinscicd, then click on add Jenkins CI Integration.
* List of steps will be given by slack
* scroll down to step 3 and copy the token, store it safely using word
* When you scroll down, you can see the token or regenerate it.
* scroll down and click on save
* Go to Jenkins server, click on manage Jenkins,
* click on manage plugins and search for Slack notification plugin
* click on manage jenkins
* click on configure system,
* scroll down to slack (Workspace = vprofilecicd, credentials = add and click Jenkins, kind = secret text, secret = paste the token from slack, ID = slacktoken, then click on add),
* Under credentials = click the drop down and select slacktoken, Default channel = #jenkinscicd, click on Test connection, when you see success click save. If you see failure signout of slack and sign in again, then start afresh
* Click Build now