**Jenkins Challenge -01**

* **Use the below code and implement our CICD stages.**
* **Source Code:** <https://github.com/betawins/hiring-app.git>
  + Repository contains the application code for the hiring app.
* Create one Declarative pipeline job
* Create one Scripted pipeline job
* Create one multi stage pipeline job
* Create one parallel stage pipeline job

**Overall Task Title**

Implementation of Jenkins CI/CD Pipelines Using Declarative, Scripted, Multi-Stage, and Parallel Stage Approaches

**Overall Task Objective**

To design and implement a complete **CI/CD pipeline solution in Jenkins** using different pipeline models—**Declarative Pipeline, Scripted Pipeline, Multi-Stage Pipeline, and Parallel Stage Pipeline**—for the *Hiring App* application.  
The objective is to automate the entire software delivery lifecycle, including source code checkout, code quality analysis, build, artifact storage, deployment, and notifications, while understanding the advantages of each pipeline approach.

**Overall Task Prerequisites**

* Jenkins server installed and running.
* Java installed on Jenkins node.
* Maven installed and configured.
* Git installed and configured.
* SonarQube server configured and integrated with Jenkins.
* Nexus Repository Manager configured with Maven hosted repository.
* Tomcat server configured with manager credentials.
* Slack workspace and Slack app integrated with Jenkins.
* Jenkins credentials configured for:
  + SonarQube
  + Nexus
  + Tomcat
  + Slack
* GitHub repository accessible:  
  https://github.com/betawins/hiring-app.git

**1.Create one Declarative pipeline job**

**Steps Performed**

**Jenkins Job Creation**

* Opened the Jenkins dashboard.
* Clicked **New Item**.
* Entered the job name **Jenkins Challenge-01**.
* Selected **Pipeline** as the job type.
* Clicked **OK** to create the job.
* Added the description as **Declarative pipeline**.
* Selected **Pipeline script** as the pipeline definition.
* Pasted the Declarative Jenkinsfile.
* Saved the job configuration.

**Git Clone**

* Jenkins connected to the GitHub repository.
* Cloned the source code from the **main** branch.
* Source code was downloaded to the Jenkins workspace.

**SonarQube Analysis**

* SonarQube analysis was triggered using the Jenkins SonarQube configuration.
* SonarQube server was temporarily unavailable, causing the analysis to fail.
* The pipeline continued execution as the SonarQube stage was configured to be non-blocking.
* Project visibility and analysis results were later verified in the SonarQube dashboard.

**Maven Build**

* Executed the Maven command mvn clean package -DskipTests.
* Maven build completed successfully.
* WAR file target/hiring.war was generated.

**Upload to Nexus (Release)**

* Jenkins uploaded the generated WAR file to the Nexus hosted repository.
* Repository used: jenkins\_challenge\_hiringapp.
* Artifact was uploaded as a **RELEASE** version with incremental versioning.
* Uploaded artifacts were verified in the Nexus repository.

**Deploy on Tomcat**

* Jenkins deployed the WAR file to the remote Tomcat 9 server.
* Application was redeployed successfully.
* Deployment was verified by accessing the application URL in a browser.

**Slack Notification**

* Jenkins sent a Slack notification after successful deployment.
* Notification included job name, build number, and success status.

**Pipeline Completion**

* All stages executed in the defined order.
* Pipeline finished with **SUCCESS** status.

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

**Declarative script:**

pipeline {

agent any

options {

timestamps()

}

environment {

// Maven

MVN\_HOME = '/opt/maven'

PATH = "${MVN\_HOME}/bin:${env.PATH}"

// SonarQube

SONAR\_PROJECT\_KEY = 'Hiring-app'

SONAR\_PROJECT\_NAME = 'Hiringapp'

// Nexus

NEXUS\_URL = '98.92.37.117:8081'

NEXUS\_REPOSITORY = 'jenkins\_challenge\_hiringapp'

// Tomcat

TOMCAT\_URL = 'http://44.223.72.232:8080'

}

stages {

stage('Git Clone') {

steps {

git branch: 'main',

url: 'https://github.com/betawins/hiring-app.git'

}

}

stage('SonarQube Analysis') {

steps {

catchError(buildResult: 'SUCCESS', stageResult: 'FAILURE') {

withSonarQubeEnv('sonarqube-server') {

sh '''

/opt/sonar-scanner/bin/sonar-scanner \

-Dsonar.projectKey=${SONAR\_PROJECT\_KEY} \

-Dsonar.projectName=${SONAR\_PROJECT\_NAME} \

-Dsonar.sources=. \

-Dsonar.exclusions=\*\*/\*.java

'''

}

}

}

}

stage('Maven Build') {

steps {

sh 'mvn clean package -DskipTests'

}

}

stage('Upload to Nexus (RELEASE)') {

steps {

script {

def WAR\_FILE = 'target/hiring.war'

def RELEASE\_VERSION = "1.${env.BUILD\_NUMBER}"

nexusArtifactUploader(

nexusVersion: 'nexus3',

protocol: 'http',

nexusUrl: '98.92.37.117:8081',

repository: 'jenkins\_challenge\_hiringapp',

credentialsId: 'nexus\_creds',

groupId: 'in.javahome',

version: RELEASE\_VERSION,

artifacts: [

[

artifactId: 'hiring',

classifier: '',

type: 'war',

file: WAR\_FILE

]

]

)

}

}

}

stage('Deploy on Tomcat') {

steps {

deploy adapters: [

tomcat9(

credentialsId: 'tomcat\_cred-01',

url: "${TOMCAT\_URL}"

)

],

contextPath: 'sabear',

war: 'target/hiring.war'

}

}

stage('Slack Notification') {

steps {

slackSend(

color: 'good',

message: "Sahith | ${env.JOB\_NAME} - #${env.BUILD\_NUMBER} DEPLOYED SUCCESSFULLY (<${env.BUILD\_URL}|Open>)",

tokenCredentialId: 'ss'

)

}

}

}

post {

failure {

slackSend(

color: 'danger',

message: "Sahith | ${env.JOB\_NAME} - #${env.BUILD\_NUMBER} FAILED (<${env.BUILD\_URL}|Open>)",

tokenCredentialId: 'ss'

)

}

}

}

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer program

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

A white rectangular object with a white border

AI-generated content may be incorrect.

A screenshot of a social media chat

AI-generated content may be incorrect.

**Error steps:**

**SonarQube Connection Error**

* During the **SonarQube Analysis** stage, Jenkins attempted to connect to the SonarQube server.
* The connection failed with **Connection refused** on port **9000**.
* Error occurred because the SonarQube service was temporarily **down or unreachable**.
* Jenkins could not fetch the SonarQube server version.
* Sonar Scanner execution returned a non-zero exit code.

**Pipeline Behaviour on SonarQube Failure**

* The SonarQube stage was configured as **non-blocking** using error handling.
* Due to this configuration, the pipeline **did not stop** when SonarQube failed.
* The SonarQube stage was marked as **FAILED**, but the overall pipeline continued.

**Skipped Impact on Other Stages**

* Maven Build stage executed normally despite SonarQube failure.
* Nexus upload stage was not affected by the SonarQube error.
* Tomcat deployment proceeded successfully.
* Slack notification was still triggered with overall pipeline success.

**Root Cause Summary**

* The error was caused by **SonarQube service unavailability**, not by Jenkins file syntax.
* No issues were found in Git, Maven, Nexus, or Tomcat configurations.

**Resolution Applied**

* SonarQube failure was handled gracefully to avoid blocking CI/CD flow.
* This ensured deployment continuity in a non-production environment.

**2.Create one Scripted pipeline job**

**Task Title**

Setup and Execute Jenkins Scripted CI/CD Pipeline with SonarQube, Nexus, Tomcat, and Slack Integration (Challenge-02)

**Objective:**

To design and execute a Scripted Jenkins pipeline that automates the complete CI/CD workflow: cloning source code from GitHub, performing code quality analysis using SonarQube, building the application with Maven, publishing artifacts to Nexus, deploying the application to a Tomcat server, and sending build status notifications to Slack.

**Prerequisites**

* Jenkins installed and accessible.
* Java and Maven installed on the Jenkins server.
* Git installed on the Jenkins server.
* SonarQube server running and configured in Jenkins.
* Nexus Repository Manager configured with a hosted Maven repository.
* Tomcat server configured with manager access.
* Jenkins credentials configured for:
  + SonarQube
  + Nexus
  + Tomcat
  + Slack
* Slack workspace and channel configured for Jenkins notifications.
* GitHub repository accessible: hiring-app.

**Steps performed:**

**Jenkins Pipeline Job Creation**

* Opened the Jenkins dashboard.
* Clicked **New Item**.
* Entered the job name **Jenkins Challenge-02**.
* Selected **Pipeline** as the job type.
* Clicked **OK** to create the job.

**General Configuration**

* Added description as **Scripted Pipeline**.
* Kept the job **Enabled**.
* Left all optional settings unchecked.
* Saved the configuration.

**Pipeline Script Configuration**

* Navigated to the **Pipeline** section.
* Selected **Pipeline script** as the definition.
* Pasted the scripted Jenkins pipeline.
* Used **Groovy Sandbox**.
* Saved the pipeline configuration.

**Slack Notification – Build Started**

* Triggered Slack notification at the beginning of the pipeline.
* Jenkins sent a **Started** message to the Slack channel.
* Slack notification was delivered successfully.

**Git Clone**

* Jenkins cloned the source code from the GitHub repository.
* Repository used: https://github.com/betawins/hiring-app.git.
* Branch used: main.
* Source code was downloaded into the Jenkins workspace.

**SonarQube Integration**

* Jenkins executed SonarQube analysis using the configured SonarQube server.
* SonarScanner analyzed project files.
* Analysis completed successfully.
* Project status was updated in the SonarQube dashboard with **Passed** quality gate.

**Maven Compilation**

* Jenkins executed mvn clean package -DskipTests.
* Maven build completed successfully.
* WAR file hiring.war was generated in the target directory.

**Nexus Artifact Upload**

* Jenkins uploaded the WAR file to Nexus Repository.
* Repository used: jenkins\_challenge\_hiringapp.
* Artifact uploaded with incremented version.
* Artifact availability was verified in Nexus UI.

**Deploy on Tomcat**

* Jenkins deployed the WAR file to the remote Tomcat server.
* Existing application was undeployed.
* New WAR file was deployed successfully.
* Application URL was accessible and working.

**Slack Notification – Build Success**

* Jenkins sent a **Success** notification to Slack.
* Slack message confirmed successful deployment.
* Notifications appeared correctly in the Slack channel.

**Pipeline Completion**

* All pipeline stages executed successfully.
* Jenkins marked the build as **SUCCESS**.
* Stage View showed all stages in green.

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

Scripted pipeline:

node {

// ---------------- ENV SETUP ----------------

env.MAVEN\_HOME = '/opt/maven'

env.PATH = "${env.MAVEN\_HOME}/bin:${env.PATH}"

// SonarQube

env.SONAR\_PROJECT\_KEY = 'Hiring-app'

env.SONAR\_PROJECT\_NAME = 'Hiringapp'

// Nexus

env.NEXUS\_URL = '98.92.37.117:8081'

env.NEXUS\_REPOSITORY = 'jenkins\_challenge\_hiringapp'

// Tomcat

env.TOMCAT\_URL = 'http://44.223.72.232:8080'

try {

// ---------------- SLACK : STARTED ----------------

stage('Slack Notification - Started') {

slackSend(

color: '#439FE0',

message: "Sahith | ${env.JOB\_NAME} - #${env.BUILD\_NUMBER} Started (<${env.BUILD\_URL}|Open>)",

tokenCredentialId: 'ss'

)

}

// ---------------- GIT CLONE ----------------

stage('Git Clone') {

git branch: 'main',

url: 'https://github.com/betawins/hiring-app.git'

}

// ---------------- SONARQUBE ----------------

stage('SonarQube Integration') {

withSonarQubeEnv('sonarqube-server') {

sh """

/opt/sonar-scanner/bin/sonar-scanner \

-Dsonar.projectKey=${SONAR\_PROJECT\_KEY} \

-Dsonar.projectName=${SONAR\_PROJECT\_NAME} \

-Dsonar.sources=. \

-Dsonar.exclusions=\*\*/\*.java

"""

}

}

// ---------------- MAVEN BUILD ----------------

stage('Maven Compilation') {

sh 'mvn clean package -DskipTests'

}

// ---------------- NEXUS UPLOAD ----------------

stage('Nexus Artifactory') {

def warFile = 'target/hiring.war'

def releaseVersion = "1.${env.BUILD\_NUMBER}"

nexusArtifactUploader(

nexusVersion: 'nexus3',

protocol: 'http',

nexusUrl: env.NEXUS\_URL,

repository: env.NEXUS\_REPOSITORY,

credentialsId: 'nexus\_creds',

groupId: 'in.javahome',

version: releaseVersion,

artifacts: [[

artifactId: 'hiring',

classifier: '',

type: 'war',

file: warFile

]]

)

}

// ---------------- TOMCAT DEPLOY ----------------

stage('Deploy On Tomcat') {

deploy adapters: [

tomcat9(

credentialsId: 'tomcat\_cred-01',

url: env.TOMCAT\_URL

)

],

contextPath: 'hiring',

war: 'target/hiring.war'

}

// ---------------- SLACK : SUCCESS ----------------

stage('Slack Notification - Success') {

slackSend(

color: 'good',

message: "Sahith | ${env.JOB\_NAME} - #${env.BUILD\_NUMBER} Success (<${env.BUILD\_URL}|Open>)",

tokenCredentialId: 'ss'

)

}

} catch (err) {

// ---------------- SLACK : FAILURE ----------------

slackSend(

color: 'danger',

message: "Sahith | ${env.JOB\_NAME} - #${env.BUILD\_NUMBER} Failed (<${env.BUILD\_URL}|Open>)",

tokenCredentialId: 'ss'

)

throw err

}

}A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

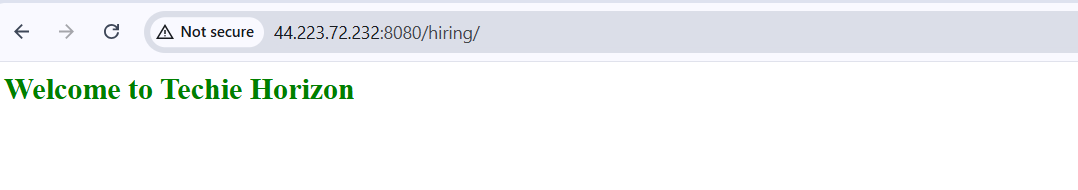
AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.



A screenshot of a computer

AI-generated content may be incorrect.

**Error Steps:**

* Initially, Slack notifications failed with the error:
* invalid\_auth
* The issue occurred because the pipeline was configured with botUser: true while using a non-bot Slack token.
* Jenkins Test Connection showed success, but runtime Slack API calls failed.
* The error was resolved by removing botUser: true so Jenkins used the same Slack integration mode as the previous challenge.

**Validation:**

* Jenkins pipeline executed all stages successfully.
* Git source code was cloned without errors.
* SonarQube analysis completed successfully and showed **Passed** status in the dashboard.
* Maven build generated the hiring.war artifact.
* Artifact was uploaded and visible in the Nexus repository.
* Application was deployed successfully on the Tomcat server and accessible via browser.
* Slack notifications for **Started** and **Success** were delivered correctly.
* Jenkins Stage View showed all stages in green.

**Conclusion**

The Jenkins Challenge-02 pipeline was successfully implemented using a **Scripted Pipeline** approach. All CI/CD stages executed as expected, including quality analysis, build, artifact management, deployment, and notifications. The Slack authentication issue was identified and resolved by aligning the Slack configuration with the working setup from Challenge-01. This task demonstrates a complete and functional end-to-end CI/CD automation workflow using Jenkins.

**3.Create one multi stage pipeline job**

**Task Title**

Multi-Stage Jenkins Declarative Pipeline for CI/CD (Git → SonarQube → Maven → Nexus → Tomcat → Slack)

**Objective**

To design and execute a **multi-stage Jenkins declarative pipeline** that automates the complete CI/CD process for a Java web application, including code checkout, quality analysis, build, artifact storage, deployment, and Slack notifications.

**Prerequisites**

* Jenkins installed and running
* Required Jenkins plugins installed:
  + Git
  + Pipeline
  + SonarQube Scanner
  + Nexus Artifact Uploader
  + Deploy to Container (Tomcat)
  + Slack Notification
* SonarQube server configured in Jenkins (sonarqube-server)
* Nexus Repository available and reachable
* Apache Tomcat server running and accessible
* Slack workspace configured with valid token credential (ss)
* Maven installed on Jenkins server (/opt/maven)
* GitHub repository accessible

**Steps Performed**

**1. Create Pipeline Job**

* Created a new Jenkins job named **Jenkins Challenge-03**
* Selected **Pipeline** as the job type
* Added description: *multi stage pipeline*

**2. Configure Pipeline Script**

* Chose **Pipeline script** as the definition
* Enabled Groovy Sandbox
* Defined environment variables for:
  + Maven
  + SonarQube
  + Nexus
  + Tomcat

**3. Git Clone Stage**

* Pulled source code from GitHub repository
* Used the main branch
* Verified successful checkout in workspace

**4. SonarQube Analysis Stage**

* Executed SonarQube scan using sonar-scanner
* Passed project key, project name, and source directory
* Excluded Java files where required
* Quality analysis completed successfully
* Dashboard updated in SonarQube

**5. Maven Build Stage**

* Ran mvn clean package -DskipTests
* Generated WAR file (hiring.war)
* Build completed successfully without test execution

**6. Upload to Nexus (RELEASE) Stage**

* Uploaded generated WAR file to Nexus repository
* Versioned artifacts using build number
* Verified successful upload in Nexus UI

**7. Deploy on Tomcat Stage**

* Deployed WAR file to remote Tomcat server
* Used Tomcat credentials configured in Jenkins
* Application redeployed successfully
* Verified application access via browser

**8. Slack Notification Stage**

* Sent success notification to Slack channel
* Included job name, build number, and build URL
* Confirmed message delivery in Slack workspace

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

**Multi stage pipeline script:**

**node {**

**// ---------------- ENV SETUP ----------------**

**env.MAVEN\_HOME = '/opt/maven'**

**env.PATH = "${env.MAVEN\_HOME}/bin:${env.PATH}"**

**env.SONAR\_PROJECT\_KEY = 'Hiring-app'**

**env.SONAR\_PROJECT\_NAME = 'Hiringapp'**

**env.NEXUS\_URL = '98.92.37.117:8081'**

**env.NEXUS\_REPOSITORY = 'jenkins\_challenge\_hiringapp'**

**env.TOMCAT\_URL = 'http://44.223.72.232:8080'**

**try {**

**// ---------------- GIT CLONE ----------------**

**stage('Git Clone') {**

**git branch: 'main',**

**url: 'https://github.com/betawins/hiring-app.git'**

**}**

**// ---------------- SONARQUBE ANALYSIS ----------------**

**stage('SonarQube Analysis') {**

**try {**

**withSonarQubeEnv('sonarqube-server') {**

**sh """**

**/opt/sonar-scanner/bin/sonar-scanner \**

**-Dsonar.projectKey=${SONAR\_PROJECT\_KEY} \**

**-Dsonar.projectName=${SONAR\_PROJECT\_NAME} \**

**-Dsonar.sources=. \**

**-Dsonar.exclusions=\*\*/\*.java**

**"""**

**}**

**} catch (err) {**

**echo "SonarQube analysis failed, continuing pipeline"**

**}**

**}**

**// ---------------- MAVEN BUILD ----------------**

**stage('Maven Build') {**

**sh 'mvn clean package -DskipTests'**

**}**

**// ---------------- UPLOAD TO NEXUS (RELEASE) ----------------**

**stage('Upload to Nexus (RELEASE)') {**

**def WAR\_FILE = 'target/hiring.war'**

**def RELEASE\_VERSION = "1.${env.BUILD\_NUMBER}"**

**nexusArtifactUploader(**

**nexusVersion: 'nexus3',**

**protocol: 'http',**

**nexusUrl: env.NEXUS\_URL,**

**repository: env.NEXUS\_REPOSITORY,**

**credentialsId: 'nexus\_creds',**

**groupId: 'in.javahome',**

**version: RELEASE\_VERSION,**

**artifacts: [[**

**artifactId: 'hiring',**

**classifier: '',**

**type: 'war',**

**file: WAR\_FILE**

**]]**

**)**

**}**

**// ---------------- DEPLOY ON TOMCAT ----------------**

**stage('Deploy on Tomcat') {**

**deploy adapters: [**

**tomcat9(**

**credentialsId: 'tomcat\_cred-01',**

**url: env.TOMCAT\_URL**

**)**

**],**

**contextPath: 'sabear',**

**war: 'target/hiring.war'**

**}**

**// ---------------- SLACK SUCCESS ----------------**

**stage('Slack Notification - Success') {**

**slackSend(**

**color: 'good',**

**message: "Sahith | ${env.JOB\_NAME} - #${env.BUILD\_NUMBER} DEPLOYED SUCCESSFULLY (<${env.BUILD\_URL}|Open>)",**

**tokenCredentialId: 'ss'**

**)**

**}**

**} catch (err) {**

**// ---------------- SLACK FAILURE ----------------**

**slackSend(**

**color: 'danger',**

**message: "Sahith | ${env.JOB\_NAME} - #${env.BUILD\_NUMBER} FAILED (<${env.BUILD\_URL}|Open>)",**

**tokenCredentialId: 'ss'**

**)**

**throw err**

**}**

**}**

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

**Error Steps:**

**Slack Authentication Issue (Earlier Builds)**

* Slack notifications initially failed with invalid\_auth
* Root cause:
  + Slack token mismatch or incorrect credential usage
* Resolution:
  + Verified Slack app configuration
  + Used correct token credential ID (ss)
  + Retested Slack connection from Jenkins
* Result:
  + Slack notifications started working successfully

**SonarQube Connectivity Issues (Earlier Builds)**

* Temporary connection refusal due to SonarQube service downtime
* Pipeline configured with catchError to continue build
* Issue resolved once SonarQube service became available

**Validation**

* Jenkins Stage View shows all stages in green
* SonarQube dashboard shows successful analysis
* Nexus repository contains uploaded WAR versions
* Application accessible via Tomcat URL
* Slack channel received deployment success message
* Jenkins console output shows Finished: SUCCESS

**Conclusion**

A fully functional **multi-stage Jenkins declarative pipeline** was successfully implemented.  
The pipeline automates code quality checks, build, artifact management, deployment, and notifications, providing a reliable and repeatable CI/CD workflow suitable for real-world DevOps practices.

**4.Create one parallel stage pipeline job**

**Task Title**

Implementation of Jenkins Parallel Stage CI/CD Pipeline with SonarQube, Maven, Nexus, Tomcat, and Slack Integration

**Objective**

To design and execute a **Jenkins parallel stage** that optimizes build time by running code quality analysis and build processes concurrently, followed by artifact upload, deployment, and real-time Slack notifications.

**Prerequisites**

* Jenkins server installed and running.
* Java and Maven installed on Jenkins node.
* Git installed and configured.
* SonarQube server running and added to Jenkins configuration.
* Nexus Repository Manager with Maven hosted repository configured.
* Tomcat server configured with manager credentials.
* Jenkins credentials configured for:
  + SonarQube
  + Nexus
  + Tomcat
  + Slack
* GitHub repository accessible (hiring-app).

**Steps Performed**

**1. Jenkins Job Creation**

* Created a new Jenkins job named **Jenkins Challenge-04**.
* Selected **Pipeline** as the job type.
* Added description as *Parallel stage pipeline*.

**2. Pipeline Script Configuration**

* Chose **Pipeline script** as the definition.
* Enabled **Groovy Sandbox**.
* Defined environment variables for Maven, SonarQube, Nexus, and Tomcat.
* Enabled timestamps for better log visibility.

**3. Git Clone Stage**

* Cloned source code from GitHub repository.
* Checked out the main branch.
* Prepared workspace for further stages.

**4. Parallel Quality & Build Stage**

* Configured a **parallel stage** named *Quality & Build (Parallel)*.
* Executed **SonarQube Analysis** in one parallel branch.
* Executed **Maven Build** in another parallel branch.
* Reduced pipeline execution time by running both tasks simultaneously.
* SonarQube quality gate results were captured.

**5. Upload to Nexus Repository**

* Generated WAR file was uploaded to Nexus.
* Artifact version was dynamically generated using the Jenkins build number.
* Verified successful upload in Nexus repository.

**6. Deploy on Tomcat**

* Deployed the WAR file to Tomcat server.
* Existing application version was undeployed.
* New version was deployed successfully.
* Application became accessible via browser.

**7. Slack Notification**

* Sent Slack notification on successful deployment.
* Message included job name, build number, and build URL.
* Slack integration worked successfully.

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

parallel stage pipeline script:

pipeline {

agent any

options {

timestamps()

}

environment {

// Maven

MVN\_HOME = '/opt/maven'

PATH = "${MVN\_HOME}/bin:${env.PATH}"

// SonarQube

SONAR\_PROJECT\_KEY = 'Hiring-app'

SONAR\_PROJECT\_NAME = 'Hiringapp'

// Nexus

NEXUS\_URL = '98.92.37.117:8081'

NEXUS\_REPOSITORY = 'jenkins\_challenge\_hiringapp'

// Tomcat

TOMCAT\_URL = 'http://44.223.72.232:8080'

}

stages {

stage('Git Clone') {

steps {

git branch: 'main',

url: 'https://github.com/betawins/hiring-app.git'

}

}

stage('Quality & Build (Parallel)') {

parallel {

stage('SonarQube Analysis') {

steps {

catchError(buildResult: 'SUCCESS', stageResult: 'FAILURE') {

withSonarQubeEnv('sonarqube-server') {

sh '''

/opt/sonar-scanner/bin/sonar-scanner \

-Dsonar.projectKey=${SONAR\_PROJECT\_KEY} \

-Dsonar.projectName=${SONAR\_PROJECT\_NAME} \

-Dsonar.sources=. \

-Dsonar.exclusions=\*\*/\*.java

'''

}

}

}

}

stage('Maven Build') {

steps {

sh 'mvn clean package -DskipTests'

}

}

}

}

stage('Upload to Nexus (RELEASE)') {

steps {

script {

def WAR\_FILE = 'target/hiring.war'

def RELEASE\_VERSION = "1.${env.BUILD\_NUMBER}"

nexusArtifactUploader(

nexusVersion: 'nexus3',

protocol: 'http',

nexusUrl: env.NEXUS\_URL,

repository: env.NEXUS\_REPOSITORY,

credentialsId: 'nexus\_creds',

groupId: 'in.javahome',

version: RELEASE\_VERSION,

artifacts: [

[

artifactId: 'hiring',

classifier: '',

type: 'war',

file: WAR\_FILE

]

]

)

}

}

}

stage('Deploy on Tomcat') {

steps {

deploy adapters: [

tomcat9(

credentialsId: 'tomcat\_cred-01',

url: "${TOMCAT\_URL}"

)

],

contextPath: 'hiring',

war: 'target/hiring.war'

}

}

stage('Slack Notification - Success') {

steps {

slackSend(

color: 'good',

message: "Sahith | ${env.JOB\_NAME} - #${env.BUILD\_NUMBER} DEPLOYED SUCCESSFULLY (<${env.BUILD\_URL}|Open>)",

tokenCredentialId: 'ss'

)

}

}

}

post {

failure {

slackSend(

color: 'danger',

message: "Sahith | ${env.JOB\_NAME} - #${env.BUILD\_NUMBER} FAILED (<${env.BUILD\_URL}|Open>)",

tokenCredentialId: 'ss'

)

}

}

}

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

**Error Steps:**

* In earlier builds, SonarQube stage failed due to connectivity issues.
* Slack notifications previously failed because of incorrect authentication settings.
* Errors were resolved by:
  + Verifying SonarQube server availability.
  + Aligning Slack configuration with Jenkins global Slack settings.
* Final pipeline execution completed without errors.

**Validation:**

* Jenkins Stage View showed successful execution of all stages.
* Parallel execution was visible in Pipeline Overview.
* SonarQube dashboard showed **Passed** quality gate.
* Nexus repository contained uploaded artifact versions.
* Application URL opened successfully in browser.
* Slack channel received deployment success messages.

**Conclusion:**

The Jenkins Challenge-04 parallel pipeline was successfully implemented, demonstrating efficient CI/CD automation by executing quality analysis and build stages concurrently. This approach reduced overall pipeline execution time while maintaining build reliability, code quality checks, artifact management, automated deployment, and real-time notifications. The pipeline meets CI/CD best practices and improves performance and scalability.

**Overall Task Conclusion:**

All four Jenkins pipeline types—**Declarative, Scripted, Multi-Stage, and Parallel Stage pipelines**—were successfully implemented for the Hiring App application.  
Each pipeline automated the CI/CD workflow, including Git checkout, SonarQube analysis, Maven build, Nexus artifact upload, Tomcat deployment, and Slack notifications.

This exercise provided practical understanding of:

* Structured pipelines using Declarative syntax
* Flexible logic handling using Scripted pipelines
* Clear stage separation using Multi-Stage pipelines
* Performance optimization using Parallel Stage pipelines

Overall, the CI/CD implementation improved automation, reduced manual effort, ensured consistent deployments, and followed DevOps best practices for scalable and efficient application delivery.