# ✅ YOUR PARALLEL PIPELINE (FINAL REFERENCE)

```
pipeline {
    agent any

    stages {

        stage('Git Clone') {
            steps {
                git branch: 'main',
                url: 'https://github.com/betawins/hiring-app.git'
            }
        }

        stage('Read Version From POM') {
            steps {
                script {
                    VERSION = sh(
                        script: "mvn help:evaluate -Dexpression=project.version -q -DforceStdout",
                        returnStdout: true
                    ).trim()
                }
            }
        }

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

```
        stage('SonarQube Scan') {
            steps {
                withSonarQubeEnv('SonarQube') {
                    sh 'mvn sonar:sonar'
                }
            }
        }

        stage('Maven Build') {
            steps {
                sh 'mvn clean install -DskipTests'
            }
        }
    }
}

stage('Upload To Nexus') {
    steps {
        nexusArtifactUploader(
            nexusVersion: 'nexus3',
            protocol: 'http',
            nexusUrl: '184.73.134.76:8081',
            groupId: 'in.javahome',
            version: VERSION,
            repository: 'ParallelStage-Hiring',
            credentialsId: 'Nexus',
            artifacts: [[artifactId: 'hiring', classifier: '', file: 'target/hiring.war', type: 'war']]
        )
    }
}
```

```
        stage('Deploy To Tomcat') {
            steps {
                withCredentials([usernamePassword(credentialsId: 'tomcat-cred', usernameVariable: 'USER',
passwordVariable: 'PASS')]) {
                    sh 'curl -u $USER:$PASS --upload-file target/hiring.war
"http://13.218.232.37:8082/manager/text/deploy?path=/hiring-parallel&update=true"'
                }
            }
        }
    }

    post {
        success {
            slackSend channel: '#jenkins-integration',
            tokenCredentialId: 'Viqaas-17',
            message: "PARALLEL SUCCESS ${env.JOB_NAME} #${env.BUILD_NUMBER}"
        }
        failure {
            slackSend channel: '#jenkins-integration',
            tokenCredentialId: 'Viqaas-17',
            message: "PARALLEL FAILED ${env.JOB_NAME} #${env.BUILD_NUMBER}"
        }
    }
}
```

## 🧠 HOW PARALLEL IS WRITTEN (CORE IDEA)

Structure to remember forever:

```
stage
 └── parallel
      ├── stage
      │    └── steps
      └── stage
           └── steps
```

Example from your script:

```
Parallel Quality & Build
 ├── SonarQube Scan
 └── Maven Build
```

**Interview one-liner:**

👉 *"Parallel block runs multiple stages simultaneously inside one parent stage."*

## ⚙️ EXPLANATION OF EACH COMMAND (VERY SIMPLE)

### ☑️ `git branch:'main', url:'repo'`

Downloads source code from GitHub.

👉 Without this, nothing else works.

**One-liner:**

👉 *"Git step checks out application source for pipeline execution."*

### ☑️ `script { VERSION = sh(...) }`

Reads version from `pom.xml`.

Command used:

```
mvn help:evaluate -Dexpression=project.version
```

Why?

👉 Release Nexus repo needs stable version.

## ☑ `parallel {}`

Splits pipeline into multiple paths.

Jenkins runs:

```
SonarQube Scan  +  Maven Build
```

at the same time.

## ☑ `withSonarQubeEnv('SonarQube')`

Injects SonarQube server settings automatically.

Inside:

```
mvn sonar:sonar
```

Runs code quality scan.

## ☑ `sh 'mvn clean install -DskipTests'`

Builds WAR file.

Breakdown:

```
clean     → delete old build
install   → build artifact
-DskipTests → faster build
```

## ✅ `nexusArtifactUploader`

Uploads WAR to Nexus repo.

Key parts:

```
groupId  → folder structure
version  → artifact version
file     → target/hiring.war
```

## ✅ `withCredentials`

Securely loads username/password.

Never hardcode secrets.

## ✅ `curl --upload-file`

Deploys WAR to Tomcat Manager API.

Creates:

```
/hiring-parallel
```

## ✅ `post { success / failure }`

Slack notification after pipeline finishes.

## ⚠ CAUTIONS WHEN WRITING PARALLEL PIPELINES

### ✕ Don’t parallelize dependent tasks

Bad example:

```
Build + Deploy parallel
```

Deploy needs WAR → will fail.

### ✕ Don’t write to same file in two branches

Can corrupt workspace.

### ✕ Don’t overload Jenkins agent

Parallel = more CPU/RAM usage.

### ✕ Avoid secrets in shell commands

Always use `withCredentials`.

## ☑ PREREQUISITES BEFORE USING PARALLEL

Make sure:

✓ Jenkins node has enough resources

✓ SonarQube configured

✓ Maven installed

✓ Nexus credentials exist

✓ Tomcat credential exists

## 🧩 HOW YOU CAN DESIGN YOUR OWN PARALLEL JOBS

Follow this simple thinking process:

### Step 1 — Draw flow

```
Checkout
   ↓
Parallel Checks
   ├── Security Scan
   ├── Unit Tests
   └── Sonar
   ↓
Build
   ↓
Deploy
```

### Step 2 — Ask ONE question

```
Can these run without waiting for each other?
```

If YES → parallel.

### Step 3 — Convert drawing to code

```
stage('Something') {
    parallel {
        stage('Task A') { steps { ... } }
        stage('Task B') { steps { ... } }
    }
}
```

That's it.

**Interview one-liner:**

👉 *"Parallel stages are designed by identifying independent pipeline tasks."*

## 🎯 FINAL MEMORY CHEAT SHEET

```
Normal Stage:
stage → steps

Parallel Stage:
stage → parallel → stage → steps
```