

Exp No :15

# Deployment Strategies and Pipeline Visualization with

Date : **Monitoring**

---

## Aim:

To implement deployment strategies, visualize Jenkins pipeline stages, and monitor pipeline executions for a Java project using Jenkins and SonarQube.

## Procedure:

1. Install and configure **Jenkins**.
2. Set up **SonarQube** for code quality analysis.
3. Install **Java**, **Maven**, and **Git** on your machine.

Ensure the following Jenkins plugins are installed:

- **Pipeline**
- **Maven Integration**
- **SonarQube Scanner**

## 2. Create the Application:

1. Write a simple Java program:
  - Create a HelloWorldProject containing an App class with a main method.
  - The method should print "Hello World!" to the console.
2. Write JUnit test cases to validate the functionality of the App class.

## 3. Create and Configure Jenkins Pipeline:

1. Save the following Jenkinsfile in the root of your project

```
pipeline {  
    agent any  
    environment {  
        MAVEN_HOME = tool 'Maven'  
        SONARQUBE_SERVER = 'SonarQube'  
    }  
    stages {
```

```
stage('Checkout') {
    steps {
        git url: 'file:///path/to/your/local/repo' // Update this path
    }
}

stage('Build') {
    steps {
        script {
            try {
                sh "${MAVEN_HOME}/bin/mvn clean compile"
            } catch (Exception e) {
                error("Build failed: ${e.message}")
            }
        }
    }
}

stage('Unit Test') {
    steps {
        script {
            try {
                sh "${MAVEN_HOME}/bin/mvn test"
            } catch (Exception e) {
                error("Tests failed: ${e.message}")
            }
        }
    }
}

post {
    always {
        junit 'pom.xml'
    }
}
```

```
    }  
  }  
  stage('Quality Gate') {  
    steps {  
      script {  
        withSonarQubeEnv(SONARQUBE_SERVER) {  
          sh "${MAVEN_HOME}/bin/mvn sonar:sonar -Dsonar.login=YOUR_TOKEN"  
        }  
      }  
    }  
  }  
  stage('Deploy') {  
    steps {  
      script {  
        try {  
          echo "Deploying application locally..."  
        } catch (Exception e) {  
          error("Deployment failed: ${e.message}")  
        }  
      }  
    }  
  }  
}  
post {  
  success {  
    echo 'Pipeline completed successfully!'  
  }  
  failure {  
    echo 'Pipeline failed!'  
  }  
}
```

```
}  
  
}
```

## 4. Pipeline Execution:

- Open Jenkins and create a new pipeline job.
- Point the pipeline to your Jenkinsfile.
- Execute the pipeline and monitor the stages.

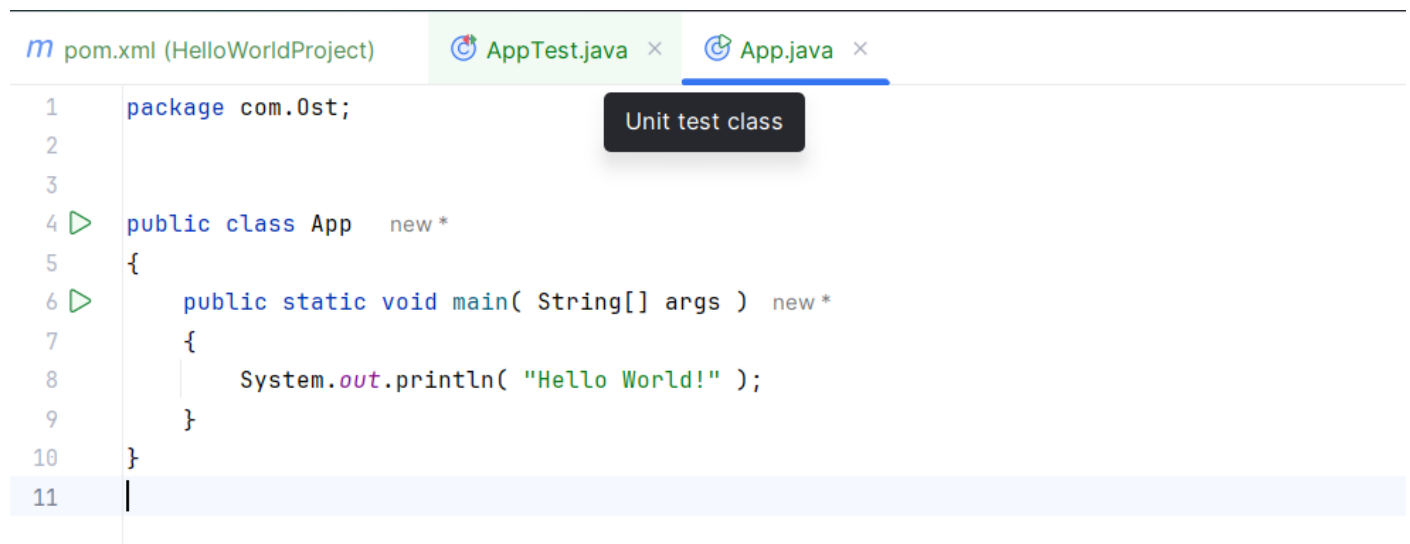
## 5. Deployment Strategies:

- Rolling Deployment: Gradually replace instances of the previous version with the new version.
- Canary Deployment: Deploy to a subset of users before rolling out widely.
- Blue-Green Deployment: Deploy the new version in parallel to the existing version to test before switching traffic.

## 6. Pipeline Visualization and Monitoring:

- Use the Jenkins Pipeline Visualization feature to view the execution of each stage.
- Monitor logs for detailed output and error debugging.
- Collect and analyze test reports and quality gate results for insights.

## Program:



```
m pom.xml (HelloWorldProject) AppTest.java × App.java ×  
1 package com.0st;  
2  
3  
4 public class App new *  
5 {  
6     public static void main( String[] args ) new *  
7     {  
8         System.out.println( "Hello World!" );  
9     }  
10 }  
11 |
```

Unit test class

## Testing Program :

```
m pom.xml (HelloWorldProject) AppTest.java x App.java
1 package com.0st;// src/test/java/HelloWorldTest.java
2 import org.junit.Test;
3
4 import static junit.framework.Assert.assertEquals;
5 import static org.junit.Assert.assertEquals;
6
7 public class AppTest { new *
8     @Test new *
9     public void testHelloWorldOutput() {
10         assertEquals( expected: "Hello, World!", actual: "Hello, World!");
11     }
12 }
13
```

Preparation	
Observation	
Output	
Viva	
Record	
Total	

## Result:

Thus the verification of the integration of application components using JUnit and enforce code quality standards using SonarQube has been completed.

# Output:

Dashboard > gfg pipeline > #2

Status

Changes

Console Output

View as plain text

Edit Build Information

Delete build '#2'

Git Build Data

Replay

Pipeline Steps

Workspaces

Previous Build

Console Output

Skipping 133 KB. [Full Log](#)

Downloading from central: <https://repo.maven.apache.org/maven2/org/apache/maven/maven-plugin-descriptor/2.0.6/maven-plugin-descriptor-2.0.6.jar>  
Downloaded from central: <https://repo.maven.apache.org/maven2/org/apache/maven/maven-repository-metadata/2.0.6/maven-repository-metadata-2.0.6.jar> (24 kB at 544 kB/s)  
Downloading from central: <https://repo.maven.apache.org/maven2/org/codehaus/plexus/plexus-interactivity-api/1.0-alpha-4/plexus-interactivity-api-1.0-alpha-4.jar>  
Progress (1): 4.1/14 kB  
Progress (1): 8.2/14 kB  
Progress (1): 12/14 kB  
Progress (1): 14 kB  
Progress (2): 14 kB | 4.1/13 kB  
Progress (2): 14 kB | 8.2/13 kB  
Progress (3): 14 kB | 8.2/13 kB | 4.1/30 kB  
Progress (3): 14 kB | 8.2/13 kB | 8.2/30 kB  
Progress (3): 14 kB | 8.2/13 kB | 12/30 kB  
Progress (3): 14 kB | 8.2/13 kB | 16/30 kB  
Progress (3): 14 kB | 8.2/13 kB | 20/30 kB  
Progress (3): 14 kB | 8.2/13 kB | 24/30 kB  
Progress (3): 14 kB | 8.2/13 kB | 28/30 kB  
Progress (3): 14 kB | 8.2/13 kB | 30 kB  
Progress (4): 14 kB | 8.2/13 kB | 30 kB | 4.1/29 kB  
Progress (4): 14 kB | 12/13 kB | 30 kB | 4.1/29 kB  
Progress (5): 14 kB | 12/13 kB | 30 kB | 4.1/29 kB | 4.1/37 kB

Dashboard > gfg pipeline > #2

WARNING: Illegal reflective access by com.thoughtworks.xstream.core.util.Fields  
(file:/var/lib/jenkins/.m2/repository/com/thoughtworks/xstream/xstream/1.3.1/xstream-1.3.1.jar) to field java.util.Properties.defaults  
WARNING: Please consider reporting this to the maintainers of com.thoughtworks.xstream.core.util.Fields  
WARNING: Use --illegal-access=warn to enable warnings of further illegal reflective access operations  
WARNING: All illegal access operations will be denied in a future release  
[INFO] Packaging webapp  
[INFO] Assembling webapp [maven-web-application] in [/var/lib/jenkins/workspace/gfg pipeline/target/maven-web-application]  
[INFO] Processing war project  
[INFO] Copying webapp resources [/var/lib/jenkins/workspace/gfg pipeline/src/main/webapp]  
[INFO] Webapp assembled in [105 msecs]  
[INFO] Building war: /var/lib/jenkins/workspace/gfg pipeline/target/maven-web-application.war  
[INFO] WEB-INF/web.xml already added, skipping  
[INFO] -----  
[INFO] BUILD SUCCESS  
[INFO] -----  
[INFO] Total time: 11.551 s  
[INFO] Finished at: 2023-03-31T10:31:52Z  
[INFO] -----  
[Pipeline] }  
[Pipeline] // stage  
[Pipeline] }  
[Pipeline] // stage  
[Pipeline] }  
[Pipeline] // node  
[Pipeline] End of Pipeline  
Finished: SUCCESS

REST API Jenkins 2.397