SOFTWARE ENGINEERING LAB

EXERCISE – 6

TOPIC – 2

BUILDING THE CI/CD FREESTYLE PIPELINE USING JENKINS FOR MAVEN WEB PROJECT WITH POLL SCM

Note: At every step take screenshots and save in a document

A CI/CD (Continuous Integration and Continuous Deployment) pipeline helps automate the building, testing, and deployment of a web project. Using Jenkins, you can monitor changes in the source code, automatically trigger builds, and deploy the project to a server. This guide details each step for setting up a Jenkins CI/CD pipeline for a Maven project, leveraging Poll SCM for automation and deploying the output to a Tomcat 9 server.

1. Setting Up Jenkins for Your Maven Web Project

Accessing Jenkins

1. **Open Jenkins:** In your browser, type `localhost:8080` to access Jenkins. This URL opens the Jenkins dashboard, where you manage projects and jobs.

Creating a New Project

1. **Start a New Project:** Click on "New Item" on the Jenkins dashboard.
2. **Enter a Project Name:** Choose a meaningful name for the project.
3. **Select Freestyle Project:** Freestyle is a flexible job type for configuring various build steps. Choose "Freestyle Project" and click "OK" to proceed.

2. Configuring Source Code Management (SCM)

SCM links Jenkins to your project's source code repository (such as Git). Jenkins can then fetch the code and build the project automatically whenever there are changes.

1. **Choose Git under SCM:** In the "Source Code Management" section, select "Git."
2. **Enter the Repository URL:** Paste the URL of your Git repository containing the Maven project.
    - **Example:** For a GitHub project, the URL may look like `https://github.com/username/projectname.git`.
3. **Branch Selection:** Specify the branch to build from, such as `main` or `master`, to ensure Jenkins monitors the correct code version.

3. Enabling Poll SCM for Automated Builds

Poll SCM in Jenkins allows you to set up an automated schedule for checking code changes.

1. **Enable Poll SCM:** In the "Build Triggers" section, check the "Poll SCM" option.
2. **Define the Schedule:** Enter a cron-like schedule to determine when Jenkins should check for changes.
    - ****Example Schedule (H/5 * * *):*** This schedule checks for updates every 5 minutes and triggers a build if changes are detected.
3. **Purpose of Poll SCM:** Poll SCM supports continuous integration by initiating builds automatically based on code updates.

4. Configuring the Build Environment and Steps

To compile and package the project, you need to set up a Maven build step.

1. **Add a Maven Build Step:** In the "Build" section, add a step called "Invoke top-level Maven targets."

2. **Enter Maven Goals:** Use `clean package` to clean old files and compile the project into a deployable WAR file.
    - **Why These Goals?**
        - `clean` deletes old build outputs to avoid conflicts.
        - `package` compiles the code into a WAR file, ready for deployment.
3. **Maven Configuration in Jenkins:** Ensure that Jenkins is configured with a Maven installation.
    - **Path:** Go to "Manage Jenkins > Global Tool Configuration" to add or verify the Maven path.
    - **Purpose:** This allows Jenkins to execute Maven commands for the build process.

5. Configuring Deployment to Tomcat 9 Server

After building the project, set up Jenkins to deploy the generated WAR file to a Tomcat 9 server.

1. **Install the Deploy to Container Plugin:**
    - Go to "Manage Jenkins > Manage Plugins" and install the "Deploy to Container" plugin. This plugin supports deployment to various servers, including Tomcat.
2. **Configure Deployment:**
    - In the "Post-build Actions" section, select "Deploy war/ear to a container."
3. **WAR File Location:** Specify the path to the WAR file, typically something like `**/*.war`.
4. **Configure the Container:**
    - Choose "Tomcat 9.x Remote" as the container.
    - Enter the Tomcat server details:
        - **Tomcat URL:** `http://yourserver:8080` (replace `yourserver` with the actual server IP or hostname).
        - **Credentials:** Provide a username and password for a Tomcat user with deployment privileges (usually configured in `tomcat-users.xml`).
5. **Why Deploy to Tomcat?**

- Tomcat is widely used for deploying Java web applications, and integrating it with Jenkins helps automate the deployment process after each successful build.

6. Setting Post-Build Actions

Post-build actions allow Jenkins to handle additional steps, like archiving artifacts or notifying team members after a successful build.

1. **Archive Build Artifacts:** In the "Post-build Actions" section, choose "Archive the artifacts" to store the generated WAR file.
   - **Pattern:** Use `**/*.war` to archive all WAR files produced by the build.
2. **Notification or Additional Steps (Optional):** Depending on your requirements, you can configure Jenkins to send notifications or trigger other jobs.
   - **Example:** Notify the team upon successful deployment or start another job for post-deployment tests.

7. Monitoring and Running the Pipeline

Once the configuration is complete, you can run the build process and monitor it.

1. **Start the Build Process:** Save your project settings and click "Build Now" on the dashboard.
   - **With Poll SCM:** Jenkins will automatically initiate a build based on the configured polling schedule whenever code changes are detected.
2. **Viewing Build Console Output:** During the build, click on the build number (in "Build History" on the left panel) and select "Console Output" to view real-time logs.
3. **Success Indicators:**
   - **Green Ball:** Indicates a successful build.
   - **Tomcat Deployment Confirmation:** Verify that the application is deployed by checking Tomcat's management console or by accessing the application URL.
4. **Failed Build Indicator:** A red ball means the build failed. Check the console output to troubleshoot and resolve issues.

Creating a Pipeline View for Monitoring

Setting up a pipeline view in Jenkins helps visualize and monitor the stages of the CI/CD pipeline.

1. **Install the Pipeline Plugin (if not already installed):** Go to "Manage Jenkins > Manage Plugins" and install the "Pipeline" and "Build Pipeline" plugins.
2. **Create a New View:**
    - From the Jenkins dashboard, select "New View."
    - Name the view (e.g., "Maven Project Pipeline") and choose "Build Pipeline View."
3. **Configure the Pipeline View:**
    - **Initial Job:** Select the initial job you created for this project.
    - **Display Options:** Customize how you want to view each stage, such as displaying build numbers, statuses, or logs.
4. **Save the View:** This new view provides a graphical representation of the pipeline, showing each stage (build, deploy) and their status.
    - **Purpose of the Pipeline View:** Makes it easier to track progress, spot issues, and access logs or details for each stage in the CI/CD process.