**CI/CD Pipeline Setup for Task Management System**

Below is a detailed step-by-step guide to set up Jenkins and create a job named **01-TMS-Prod-Build** with the specified configurations.

**Document: Setting up Jenkins for 01-TMS-Prod-Build**

**Step 1: Install Jenkins**

1. Install Jenkins on the server:

sudo apt update

sudo apt install openjdk-11-jre

wget -q -O - https://pkg.jenkins.io/debian/jenkins.io.key | sudo apt-key add -

sudo sh -c 'echo deb http://pkg.jenkins.io/debian-stable binary/ > /etc/apt/sources.list.d/jenkins.list'

sudo apt update

sudo apt install jenkins

sudo systemctl start jenkins

1. Access Jenkins via browser at <http://localhost:9090> (change the port number in /etc/default/Jenkins HTTP\_PORT).

**Step 2: Install Required Plugins**

1. Go to **Manage Jenkins > Manage Plugins > Available Plugins** and install the following:
   * **Git Plugin** (for pulling code from GitHub)
   * **Pipeline Plugin**
   * **JaCoCo Plugin** (for generating code coverage reports)
   * **Email Extension Plugin** (for email notifications)
   * **GitHub Pull Request Builder Plugin** (for pull request triggers)

**Step 3: Configure Jenkins Global Settings**

1. **Set Up Email Notifications**:
   * Go to **Manage Jenkins > Configure System**.
   * In the **Extended Email Notification** section:
     + Set SMTP Server: smtp.gmail.com.
     + Set SMTP Port: 587.
     + Use TLS and provide the email address (e.g., jd.saravanan93@gmail.com)
   * Add the credentials for the email account.
2. **Add GitHub Credentials**:
   * Go to **Manage Jenkins > Credentials > System > Global credentials**.
   * Add the GitHub credentials (personal access token).

**Step 4: Create Job: 01-TMS-Prod-Build**

1. **Create a New Job**:
   * Go to **New Item**.
   * Enter the job name as **01-TMS-Prod-Build** and select **Freestyle Project**.
   * Click **OK**.

A screenshot of a computer

Description automatically generated

1. **Configure Source Code Management**:
   * Go to the **Source Code Management** section and select **Git**.
   * Add the repository URL: <https://github.com/hema8293/tsm-final.git>
   * Use the respective credentials.

A screenshot of a computer

Description automatically generated

1. **Setup Triggers**:
   * Go to the **Build Triggers** section and select:
     + **Poll SCM**: Set the schedule as H/5 \* \* \* \* (every 5 minutes).
     + **Build periodically**: Set the schedule as H 17 \* \* \* (daily at 5 PM).
     + **GitHub Pull Request Builder**: Enable the trigger for PR approvals.
       - **Configure GitHub Integration**
         1. **Connect Jenkins to GitHub**:

Go to **Manage Jenkins > Configure System**.

Scroll to the **GitHub** section and click **Add GitHub Server**.

Provide a **Name** (e.g., GitHub-Server).

Select **Manage Credentials** & add personal access token from GitHub.

Generate the token in GitHub by navigating to **Settings > Developer Settings > Personal Access Tokens > Generate New Token**.

Enable permissions for repo, admin:repo\_hook, and write:public\_key.

* + - 1. **Test Connection**:
         * After adding the token, click **Test Connection** to ensure Jenkins can access the repository.
         * **Add Repository Hooks**:
         * Navigate to the GitHub repository: <https://github.com/hema8293/tsm-final.git>.
         * Go to **Settings > Webhooks > Add Webhook**.
         * **Payload URL**: <http://localhost:9090/github-webhook/>
         * **Content Type**: application/json
         * **Trigger Events**: Select **Pull Request Events**.
         * Save the webhook.
      2. **Configure the Job for PR Trigger**
         * **Select the GitHub Repository**:
         * In the **Source Code Management** section of the job, select **Git** and provide the repository URL: <https://github.com/hema8293/tsm-final.git>.
         * **Enable GitHub Pull Request Builder**:
         * Go to the **Build Triggers** section.
         * Select **GitHub Pull Request Builder**.
         * Specify the Build Conditions.

1. **Build the Project**:
   * Go to the **Build** section and select **Execute Shell**.
   * Add the following script to build the project:

mvn clean install

mvn spring-boot:run

A screenshot of a computer

Description automatically generated

1. **Generate JaCoCo Code Coverage Reports**:
   * Go to **Post-Build Actions** and add:
     + **Record JaCoCo Coverage Report**.
     + Specify the JaCoCo report paths: target/site/jacoco/index.html.

A screenshot of a computer

Description automatically generated

**Configuring Code Coverage Metrics:**

A screenshot of a computer

Description automatically generated

1. **Send Email Notifications**:
   * Go to **Post-Build Actions** and add:
     + **Editable Email Notification**:
       - Add recipients: [jd.saravanan93@gmail.com](mailto:jd.saravanan93@gmail.com), [yaminimerugu2022@gmail.com](mailto:yaminimerugu2022@gmail.com), [ankithreddyb90@gmail.com](mailto:ankithreddyb90@gmail.com)
       - Set the subject: Build Status for 01-TMS-Prod-Build.
       - Use $DEFAULT\_CONTENT for the email body.

A screenshot of a computer

Description automatically generated

**Step 5: Save and Test the Job**

1. Save the configuration and run the job manually to test the setup.
2. Verify:
   * The code is pulled from GitHub.
   * The build runs successfully.
   * The JaCoCo report is generated.
   * An email is sent with the build status.

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

**Conclusion**

We have now configured the Jenkins job **01-TMS-Prod-Build** to:

1. Pull code from GitHub and Set up the build triggers, such as Poll SCM, daily at 5 PM and on GitHub PR approval.
2. Build the project using Maven and Generate JaCoCo code coverage reports.
3. Send email notifications to [jd.saravanan93@gmail.com](mailto:jd.saravanan93@gmail.com), [yaminimerugu2022@gmail.com](mailto:yaminimerugu2022@gmail.com), [ankithreddyb90@gmail.com](mailto:ankithreddyb90@gmail.com)