**Step-by-Step Guide to Configuring Jenkins Jobs**

Jenkins is widely used for Continuous Integration/Continuous Deployment (CI/CD) pipelines. Below is a step-by-step procedure for configuring a Jenkins job.

**Step 1: Install & Setup Jenkins (If Not Installed)**

1. **Download Jenkins** from Jenkins official website.
2. **Install Jenkins** and start the service.
3. **Access Jenkins UI** at http://localhost:8080.
4. **Install Suggested Plugins** during setup.
5. **Create an Admin User** and log in.

**Step 2: Create a New Job in Jenkins**

1. **Go to Dashboard** → Click on **"New Item"**.
2. Enter a **job name** (e.g., My\_DevOps\_Pipeline).
3. Select **Freestyle Project** (for basic CI/CD) or **Pipeline** (for advanced automation).
4. Click **OK** to create the job.

**Step 3: Configure Source Code Management (SCM)**

1. Under the **General** section, check the box for **"GitHub Project"** (if applicable).
2. Scroll to the **Source Code Management** section:
   * Select **Git** or **SVN**.
   * Provide the **repository URL** (e.g., https://github.com/user/repo.git).
   * Add **credentials** if authentication is required.
   * Specify the branch (e.g., main or develop).

**Step 4: Configure Build Triggers**

1. Go to the **Build Triggers** section.
2. Choose one of the following options:
   * **Poll SCM** – Schedule automated builds using cron (H/5 \* \* \* \* runs every 5 minutes).
   * **Build Periodically** – Schedule builds at specific intervals.
   * **GitHub Webhook** – Trigger builds automatically on code push (requires webhook setup in GitHub).
   * **Trigger builds from other jobs** – Use **"Build after other projects are built"**.

**Step 5: Define the Build Steps**

1. Scroll down to the **Build** section.
2. Click **"Add Build Step"**, and choose:
   * **Execute Shell** – Run a shell script (for Linux/macOS).
   * **Execute Windows Batch Command** – Run a Windows batch script.
   * **Invoke Gradle, Maven, or Ant** – For Java projects.
   * **Invoke Docker commands** – For containerized builds.
3. Example Shell Script (if using Shell build step):

bash

CopyEdit

echo "Building the project..."

mvn clean install

**Step 6: Configure Post-Build Actions**

1. Scroll to the **Post-build Actions** section.
2. Add actions such as:
   * **Archive the Artifacts** (e.g., .jar, .war, or .zip files).
   * **Publish JUnit Test Results** (for automated test reports).
   * **Send Email Notifications** on build failure (requires email setup).
   * **Deploy to a server** using SCP or SSH plugins.

**Step 7: Save and Run the Job**

1. Click **"Apply"** and then **"Save"**.
2. Click **"Build Now"** to trigger the first build.
3. Monitor the build status in **"Build History"**.
4. Click on a build to view logs under **"Console Output"**.

**Step 8: Set Up a Jenkins Pipeline (Optional)**

If you prefer a **Jenkins Pipeline**, follow these steps:

1. Create a new job and select **Pipeline**.
2. Scroll to the **Pipeline** section and add a script like:

groovy

CopyEdit

pipeline {

agent any

stages {

stage('Checkout') {

steps {

git 'https://github.com/user/repo.git'

}

}

stage('Build') {

steps {

sh 'mvn clean install'

}

}

stage('Test') {

steps {

sh 'mvn test'

}

}

stage('Deploy') {

steps {

sh './deploy.sh'

}

}

}

}

1. Click **Save** and then **Build Now** to test the pipeline.

By following these steps, you can configure Jenkins jobs for automated builds and deployments