Git & GitHub Integration with Katalon Studio

Section A: Initial GitHub Setup and Project Sharing

Step 1: Create a GitHub Account

- Visit https://github.com/.
- Click **Sign Up** and complete the registration process.

Step 2: Sign In to GitHub

- Go to https://github.com/login.
- Enter your credentials and sign in to your GitHub account.

Step 3: Create a Repository on GitHub

1. Click the "+" icon in the upper-right corner of the GitHub dashboard.
2. Select **New repository**.
3. Enter the repository name (e.g., `MyKatalonProject`).
4. Set visibility to **Public** or **Private**.
5. Click **Create repository**.

Section B: Enable Git Integration in Katalon Studio

Step 4: Enable Git in Katalon Studio

1. Open **Katalon Studio**.
2. Navigate to:
   - **Windows** > **Katalon Studio Preferences** > **Katalon** > **Git**.
3. Check the box **Enable Git Integration**.
4. Click **Apply and Close**.

Step 5: Share Your Katalon Project to Git

1. In the main toolbar, go to **Git** > **Share Project**.
2. Enter the remote repository URL (e.g., `https://github.com/yourusername/MyKatalonProject.git`).
3. Authenticate with your GitHub username and personal access token (if prompted).

Section C: Commit and Push Changes

Step 6: Initial Commit

💡 *Tip: If there are files you don't want to push (e.g., logs or reports), add them to a* `.gitignore` *file before committing.*

1. Go to **Git** > **Commit**.

Step 7: Stage Files for Commit

1. In the **Git Staging** view:
    - Drag files from the **Unstaged Changes** section to the **Staged Changes** section.

Step 8: Commit and Push

1. Enter a **commit message** describing your changes.
2. Click **Commit and Push**.
3. Enter your **GitHub repository URL** and **credentials** if prompted.

Step 9: Verify Push to GitHub

- Go to your GitHub repository in a browser.
- You should see your project files and recent commit listed.

How to Get Git Credentials for Jenkins

Step 1: Generate a GitHub Personal Access Token (PAT)

1. Log in to your GitHub account.
2. Go to **https://github.com/settings/tokens**.
3. Click **"Generate new token"** > choose **"Personal Access Token (classic)"**.
4. Give your token a name (e.g., `Jenkins Git Access`).
5. Select scopes (permissions):
    - ☑ `repo` → Full access to private repositories

- ☑ `read:org` → If you are accessing organization repositories

6. Click **Generate token**.
7. Copy the token. **Save it securely!** You won't see it again.

Step 2: Add Token as Jenkins Credentials

1. Open Jenkins → go to **Manage Jenkins → Credentials**.
2. Click your credentials domain (usually **(global)**).
3. Click **Add Credentials**.
4. Choose:
   - **Kind**: `Username with password`
   - **Username**: Your GitHub username
   - **Password**: Paste your Personal Access Token here
   - **ID (optional)**: e.g., `github-pat`
   - **Description**: e.g., `GitHub token for Jenkins access`
5. Click **OK** to save.

🔧 Open Command Prompt as Administrator

1. Press `Windows + S` and type: `cmd`
2. Right-click **Command Prompt**
3. Click **"Run as Administrator"**
4. Click **"Yes"** if prompted by User Account Control (UAC)
5. Type '**git config --system core.longpaths true**' & press enter.

Section D: Cloning an Existing Project

🔧 Prerequisites

Before you begin, make sure the following are in place:

- ☑ **Git** is installed on the Jenkins host machine
- ☑ Jenkins has the **Git plugin** installed
- ☑ You have a **remote Git repository** (e.g., on GitHub, GitLab, Bitbucket)
- ☑ Jenkins is set up with proper **Git credentials** if the repository is private
-

📋 Step-by-Step Configuration

Step 1: Access Jenkins Job Configuration

1. Log in to Jenkins (`http://localhost:8080` or your Jenkins URL).
2. Go to the **Dashboard**.
3. Select an existing **freestyle project** or click **New Item** to create one.
4. If creating a new job, provide a name, select **Freestyle project**, and click **OK**.

Step 2: Select Git in Source Code Management

1. Scroll to the **Source Code Management** section.
2. Select **Git**.

   If Git is not visible, install the Git plugin from **Manage Jenkins > Plugin Manager**.

Step 3: Enter the Repository URL

1. Paste the **clone URL** of your Git repository.

Step 4: Apply and Save

1. Scroll to the bottom of the configuration page.
2. Click **Apply** and then **Save**.

Step 5: Run the Jenkins Job

1. On the project page, click **Build Now**.
2. A new build will be triggered. Check the **Build History** panel.

Step 6: Monitor Build Logs

1. Click on the latest build number (e.g., `#1`).
2. Click **Console Output**.
3. Verify that:
    - Jenkins successfully cloned the repository.
    - The workspace path is shown (e.g., `/var/lib/jenkins/workspace/YourJobName`).

🗁 Optional: Use a Custom Workspace Directory

Step 7: Set a Custom Workspace

1. Return to the job **Configuration** page.
2. Under the **General** section, click **Advanced**.
3. Check the box: **Use custom workspace**.
4. Enter a custom directory path (example: `D:\JenkinsProjects\MyKatalonProject` on Windows).

Step 8: Update Build Commands with Custom Path

1. In the **Build** section, add your command/script.
2. Use the custom workspace path if needed:
    - For **Windows Batch Command**:

```
D:
cd D:\JenkinsProjects\MyKatalonProject
katalonc.exe -noSplash -runMode=console
```

Step 9: Save and Run

1. Click **Save**.
2. Click **Build Now** to execute the updated configuration.
3. Verify the logs again to confirm successful cloning and build execution.