🔄 What is "GitHub hook trigger for GITScm polling"?

This is a **trigger mechanism** used in Jenkins jobs where **GitHub Webhooks** notify Jenkins when there's a new commit/push. However, unlike "Poll SCM", Jenkins **does not poll on a schedule**. Instead:

1. **GitHub sends a webhook** to Jenkins when code is pushed.
2. The **GitHub Plugin in Jenkins** verifies:
    - Is this webhook from a GitHub repo?
    - Does this match the repo URL in the Jenkins job's SCM config?
3. If yes, the plugin **triggers a one-time SCM polling**.
4. **Git Plugin (not GitHub Plugin)** performs the actual polling.
    - If it detects a change → Jenkins job is triggered and a build is started.

🧠 Example Use Case

- You push code to GitHub.
- GitHub sends a webhook to Jenkins (configured in repo settings).
- Jenkins receives it (via GitHub Plugin).
- Jenkins polls the repo once.
- If new changes are detected, a build is triggered.

✅ Why Use This Instead of Poll SCM?

- **More efficient** — no constant scheduled checks.
- **Instant feedback** — builds run right after a push.
- **Less load** on GitHub and Jenkins.

🧩 Additional Setup Required

To use this:

1. **In Jenkins job** → Check: `GitHub hook trigger for GITScm polling`
2. **In GitHub repo** → Settings → Webhooks:
    - Payload URL: `http://<jenkins-url>/github-webhook/`
    - Content type: `application/json`
    - Events: Push events (default is fine)

✅ GitHub Hooks – Jenkins Integration

📌 **Purpose**:
When a **commit is pushed to a GitHub repository**, Jenkins gets notified and **triggers a job** automatically (without polling).

🛠️ Step-by-Step Setup

1. **Login to GitHub**
    - Go to: `https://github.com`
    - Sign in with **username** and **password**
    - Navigate to your **repository** (e.g., `mahalogin`)
2. **Add Webhook in GitHub**
    - Go to: `Settings` → `Webhooks`
    - Click on **Add webhook**
3. **Webhook Configuration**
    - **Payload URL**: http://<Public-IP>:8080/github-webhook/

      (Replace `<Public-IP>` with your Jenkins server's IP)

- **Content Type**: application/json
- **Events to trigger**: ✔ Select: **Just the push event**

🌐 Jenkins Configuration (Job Level)

1. **Go to your Jenkins job →** `Configure`

2. **Under Build Triggers, select:**
   ✅ `GitHub hook trigger for GITScm polling`

Great job setting up your Jenkins GitHub webhook integration! From the screenshots, here's a detailed verification summary and confirmation that GitHub hook trigger for GITScm polling is working perfectly in your Assignment05 job:

✅ Summary of Configuration

Component	Status
Job Type	Freestyle project (Correct for simple build+SCM setup)
SCM Config	Git URL: `https://github.com/AbdulRehaman082493/mahalogin.git`
Branch to Build	`*/master` (Standard pattern, correct)
Build Trigger	✅ `GitHub hook trigger for GITScm polling` selected
Build Step	Maven goal: `clean install`

Post-build Action	Archive Artifacts: `target/*.war`
GitHub Repo Change	README.md updated with commit (`my 2nd commit`)
Jenkins Response	✔️ Job triggered automatically by GitHub push
Artifacts Built	`mahaLogin-2.0.war` created and archived

🔗 How We Know It's Working

- In the **build history (#1)**, it clearly shows:

  🟢 "Started by GitHub push by AbdulRehaman082493"

- That means **GitHub webhook** → **Jenkins** → **GITScm polling** → **job trigger** is 100% working.

🔍 What You Could Add or Improve

Improvement	Benefit
✅ Add GitHub credentials in Jenkins	Helps in private repo access and avoids API rate limit issues
🛠️ Add Email/Post Build Notifier	Notifies if a build fails or succeeds

🧪 Add test reports (`surefire`, `junit`)	See test results in Jenkins UI

✅ Objective

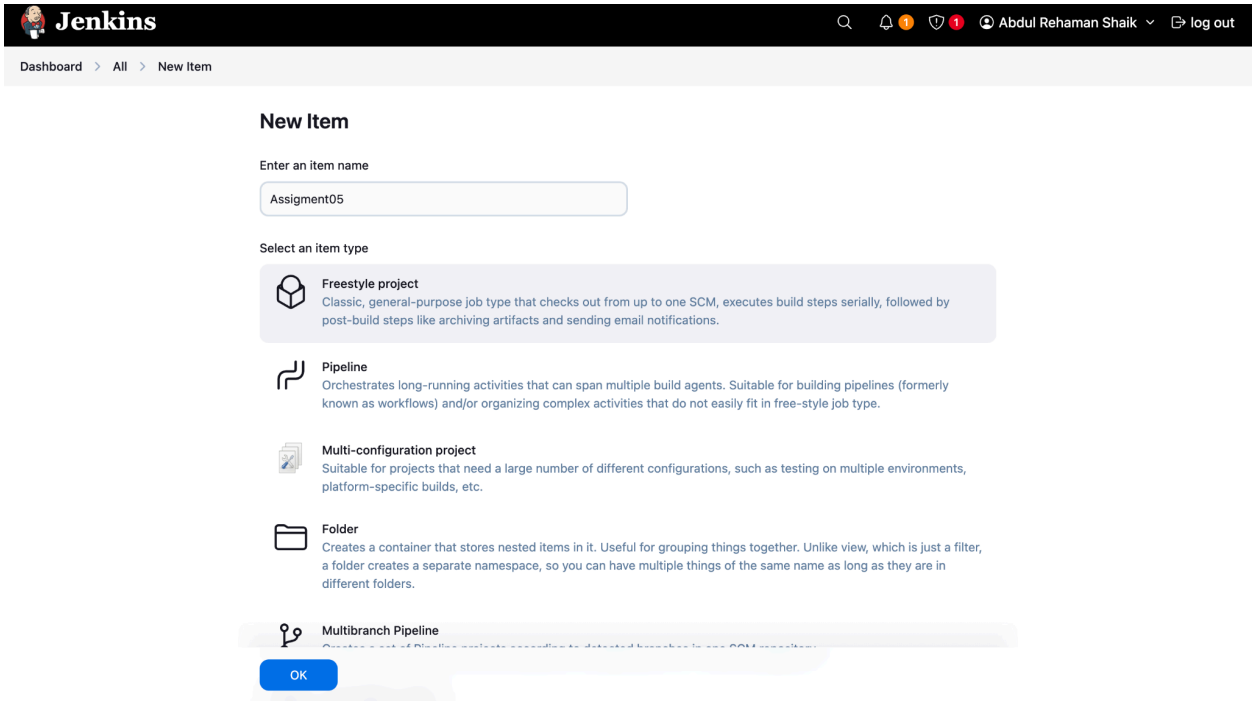
Trigger Jenkins job automatically when code is pushed to GitHub using **GitHub Webhooks** + **GITScm polling**.

🧩 Step-by-Step Jenkins Configuration

🔹 Step 1: Create a New Job in Jenkins

1. Navigate to Jenkins Dashboard.
2. Click **"New Item"**.
3. Enter an item name: `Assignment05`.
4. Select **"Freestyle project"**.
5. Click **OK**.

📸 Screenshot shows you correctly named it and selected the right job type.



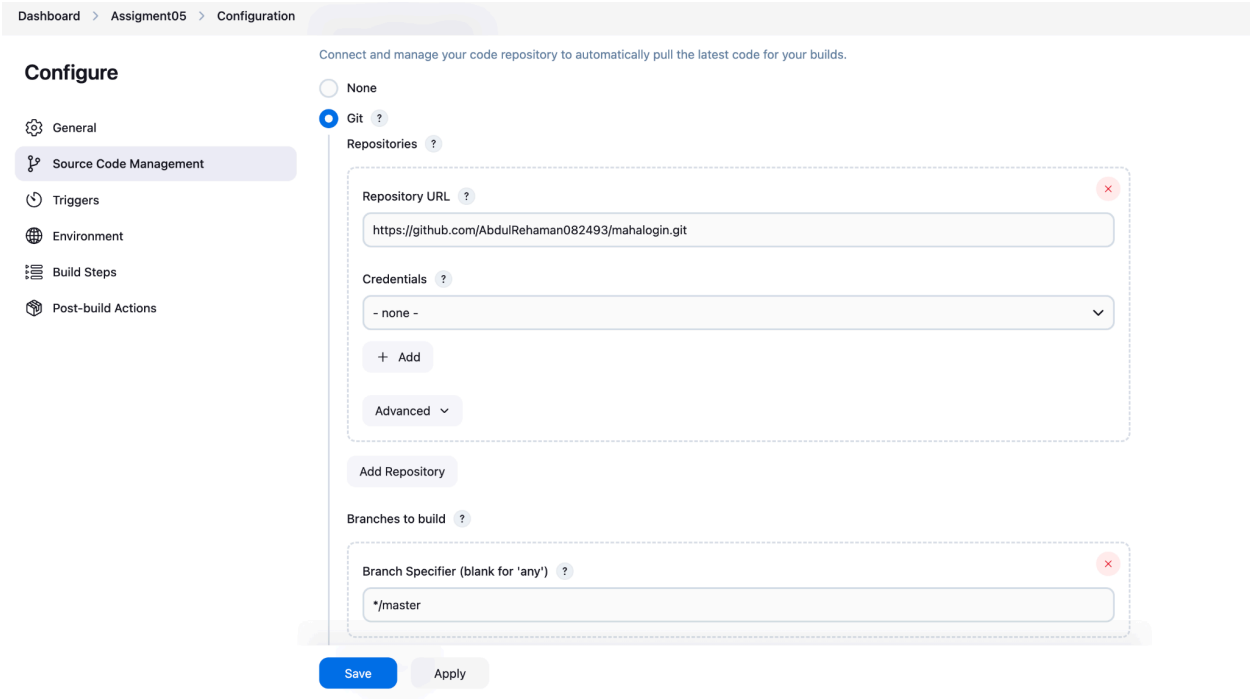
🔹 Step 2: Configure GitHub Repository (Source Code Management)

1. Go to the `Configure` section of the job.

2. Under **Source Code Management**, choose `Git`.

3. Enter your repository URL: https://github.com/AbdulRehaman082493/mahalogin.git
4. Leave credentials as `- none -` (since it's a public repo).

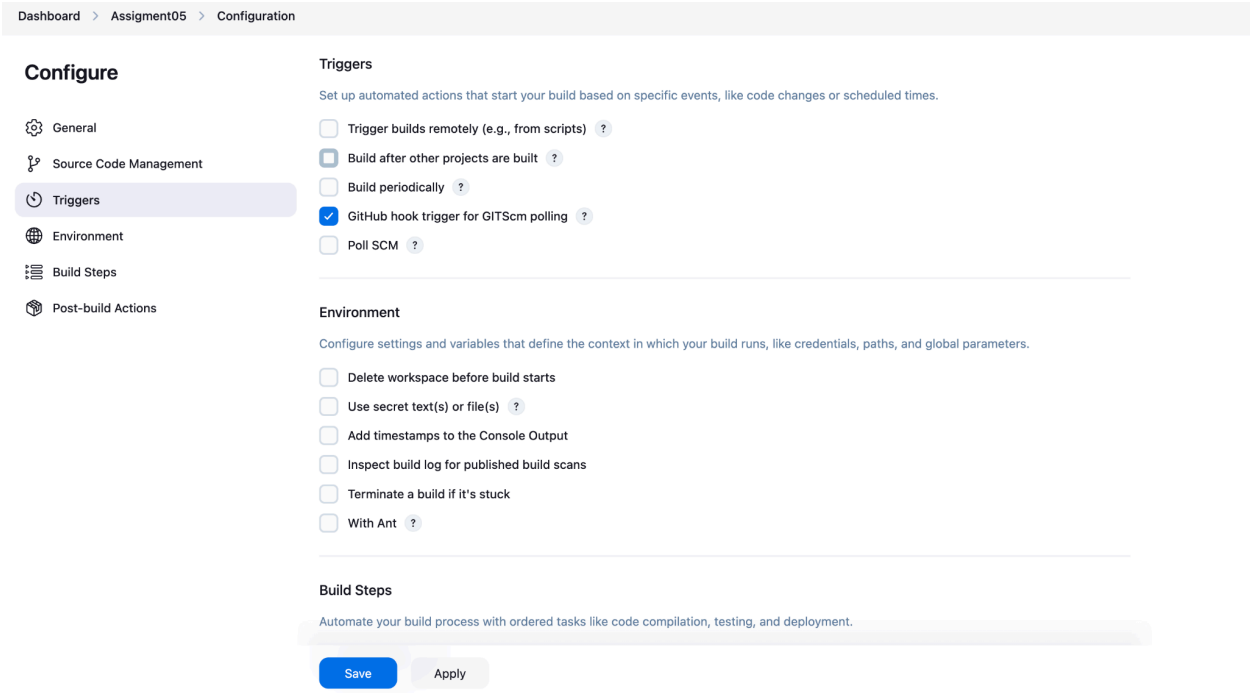
5. Set **Branch Specifier** as: */master



🔹 Step 3: Configure GitHub Trigger

1. Go to the **Build Triggers** section.
2. Check the box:
   ✅ `GitHub hook trigger for GITScm polling`

⏱️ This means Jenkins **waits for a GitHub push notification** (webhook), then **polls the repo once** to verify changes.



🔹 Step 4: Add Build Steps

1. Go to **Build Steps**.
2. Click `Add build step` → select **Invoke top-level Maven targets**.
3. In the **Goals** field, enter: clean install

📦 This compiles and packages your Java/Maven project.

🔹 Step 5: Archive WAR File

1. Go to **Post-build Actions**.
2. Click `Add post-build action` → select **Archive the artifacts**.
3. Set `Files to archive`: target/*.war

🗂️ This stores your build artifact (e.g., `mahaLogin-2.0.war`) for download or deployment.

🔁 Step-by-Step GitHub Webhook Configuration

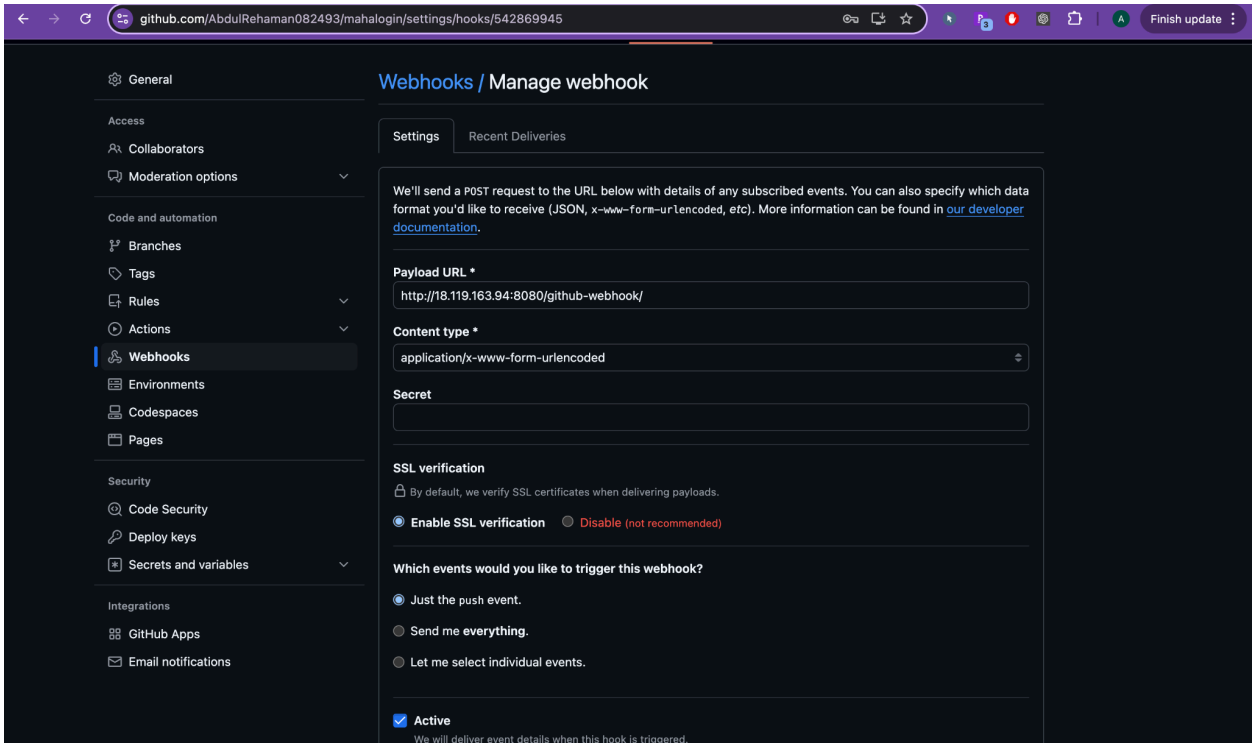
🔹 Step 6: Create Webhook in GitHub

1. Go to your GitHub repo:
   `https://github.com/AbdulRehaman082493/mahalogin`

2. Click on `Settings` → `Webhooks`.

3. Click **"Add webhook"**.

🔹 Step 7: Enter Webhook Details



- **Payload URL**: http://<your-public-IP>:8080/github-webhook/

  Replace `<your-public-IP>` with Jenkins server IP.

- **Content Type**: application/json
- **Trigger Event**: ✅ Select **Just the push event**

4. Click **"Add webhook"**.

📎 This notifies Jenkins every time you push code to GitHub

🚀 Step 8: Push to GitHub and Trigger Jenkins

1. Edit and commit code in your repo (you edited `README.md`).

2. Push the changes to GitHub:
   - git add README.md
   - git commit -m "my 2nd commit"
   - git push

✅ Step 9: Jenkins Automatically Builds

- Jenkins receives webhook from GitHub.
- Triggers **GITScm polling**.
- It detects change and builds your project.
- Builds the `.war` file and archives it.

📸 Screenshot shows:

> ✔️ *Started by GitHub push by AbdulRehaman082493*

🎉 Job `Assignment05` built successfully and archived `mahaLogin-2.0.war`.

📊 Summary Table

Step	Action	Outcome
1	Create Freestyle Project	Jenkins job created

2	Configure Git repo	Linked to GitHub repo
3	Enable GitHub trigger	Jenkins listens for webhook
4	Add Maven build step	Project compiles with `clean install`
5	Archive artifact	WAR file saved
6	Add GitHub Webhook	GitHub sends push events to Jenkins
7	Commit code to GitHub	Triggers webhook
8	Jenkins builds project	✅ Build triggered automatically