Let’s say your team is developing a Java web application using Spring Boot. The code is stored on GitHub. You want Jenkins to:

1. Pull the latest code from GitHub.
2. Build the code using Maven (mvn clean package).
3. Run unit tests.
4. Archive the generated JAR file.
5. Send a build success/failure email.

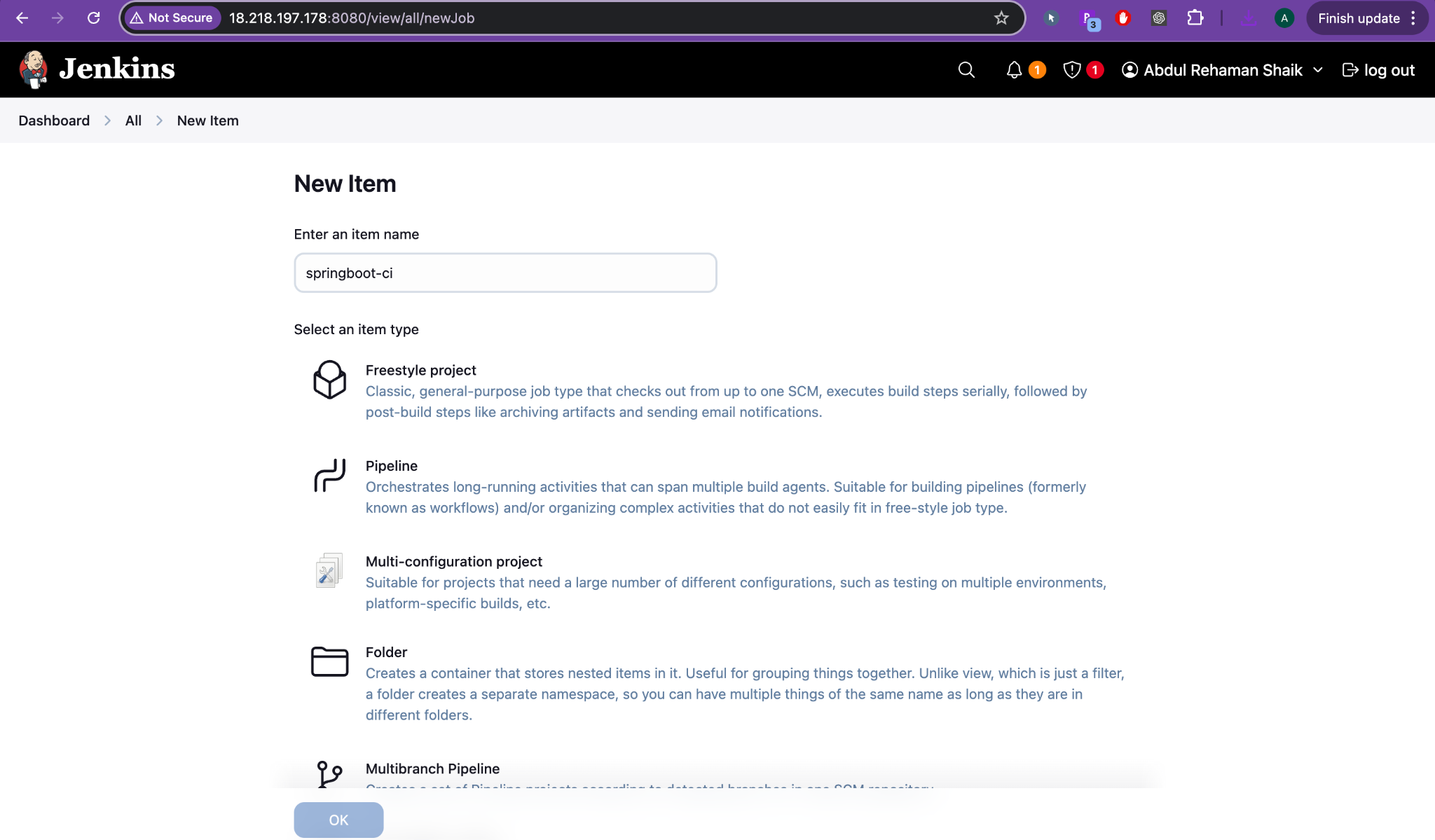
## **✅ Step-by-Step Breakdown in Jenkins (based on your screenshots)**

### **🔹 Step 1: Jenkins Dashboard (Screenshot 1)**

* This is your **home screen** after logging into Jenkins.
* You’ll start by clicking on **"New Item"** to create a job.

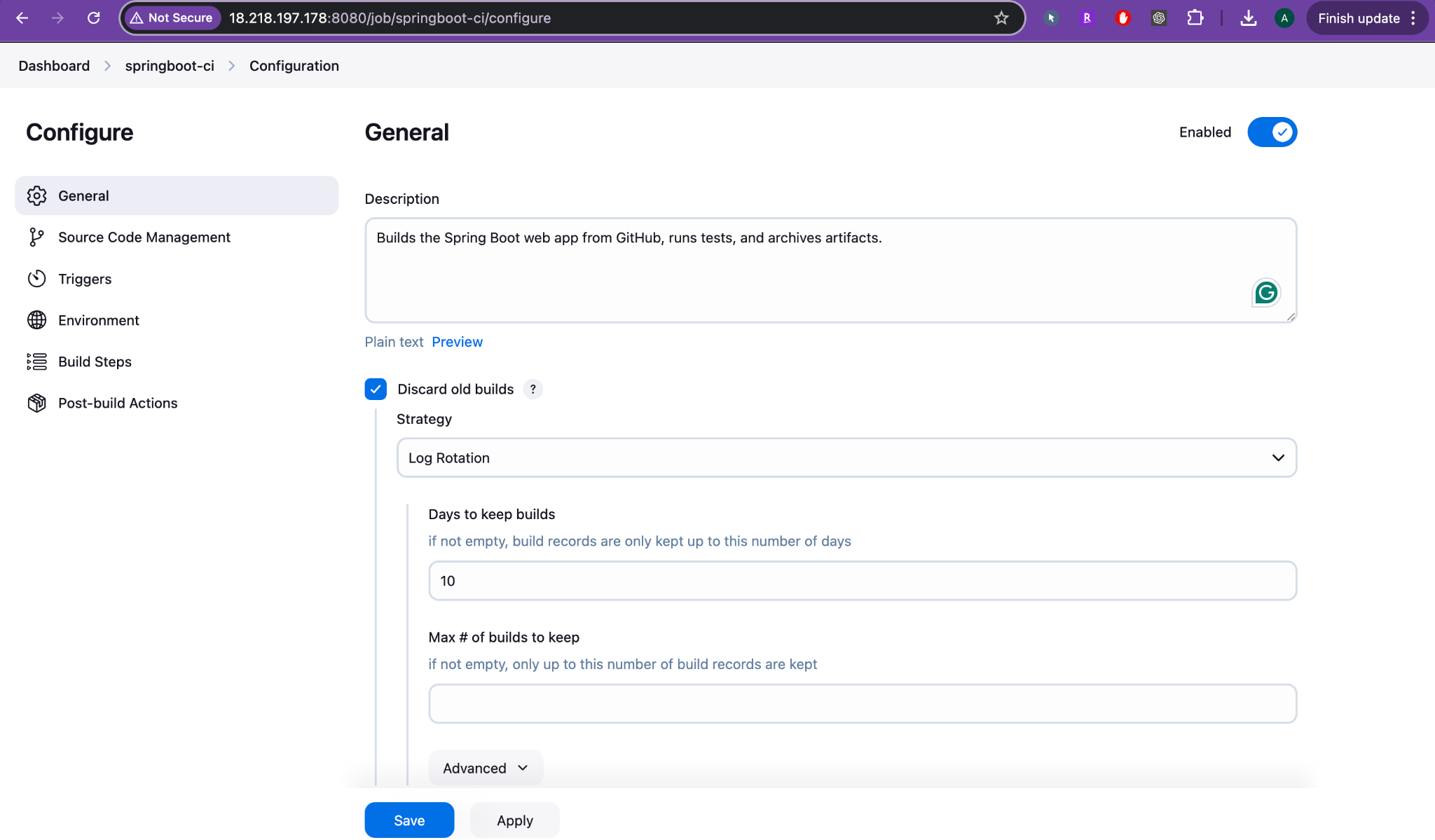
### **🔹 Step 2: Create a New Freestyle Project (Screenshot 2)**

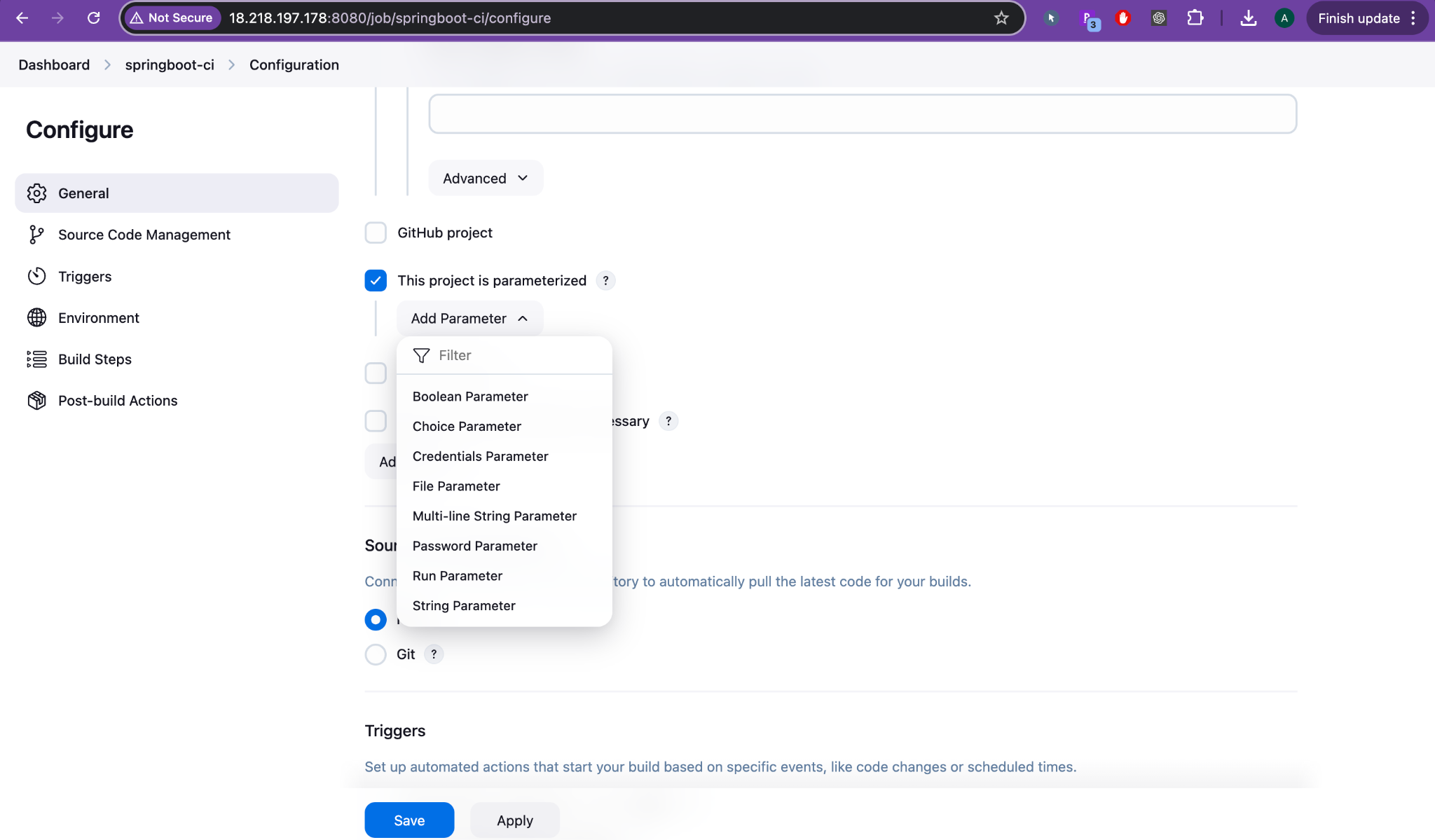
* **Name:** myjob01 (you can name it like springboot-ci)
* **Select Type:** Freestyle project
* ✅ Real-world reason to use Freestyle:  
    
    
   Ideal for simple CI jobs like build + test + notify, without complex pipeline scripts.



### **🔹 Step 3: General Configuration (Screenshot 3)**

* Add a **description** like:  
    
    
   *“Builds the Spring Boot web app from GitHub, runs tests, and archives artifacts.”*
* Check:  
  + Discard old builds: Retain only last N builds (say 10).
  + This project is parameterized: If you want to build with options (e.g., ENV=dev or ENV=prod).





### **🔹 Step 4: Source Code Management (Screenshot 4)**

* **Select:** Git

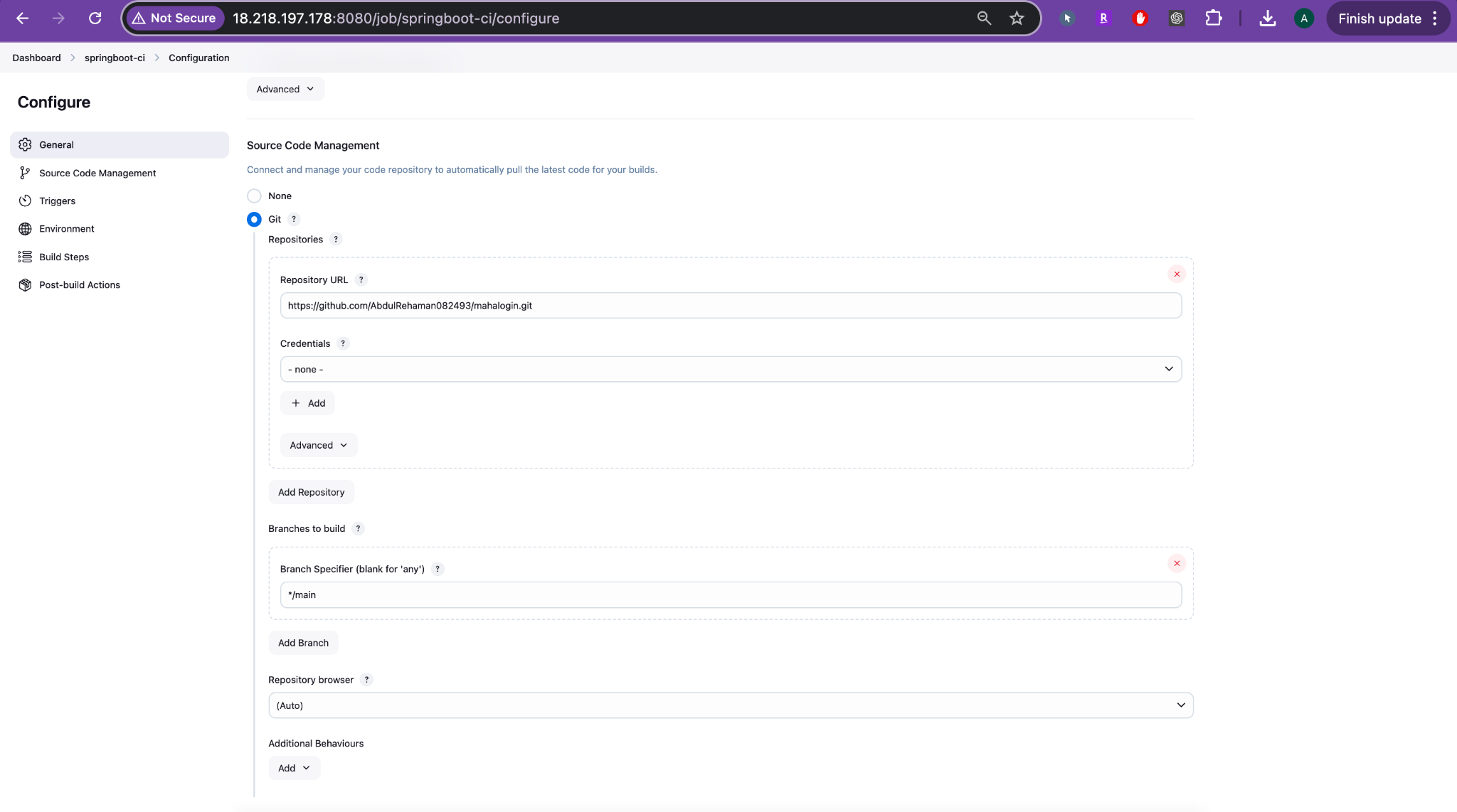
**Repository URL:**  
https://github.com/your-org/springboot-webapp.git

* **Credentials:** Add GitHub token (so Jenkins can access private repos).

**Branches to build:**  
\*/main

✅ **Real-World Purpose**:

Jenkins pulls the latest code from GitHub main branch before every build.



### **🔹 Step 5: Build Triggers (Screenshot 4)**

Choose how the job should start:

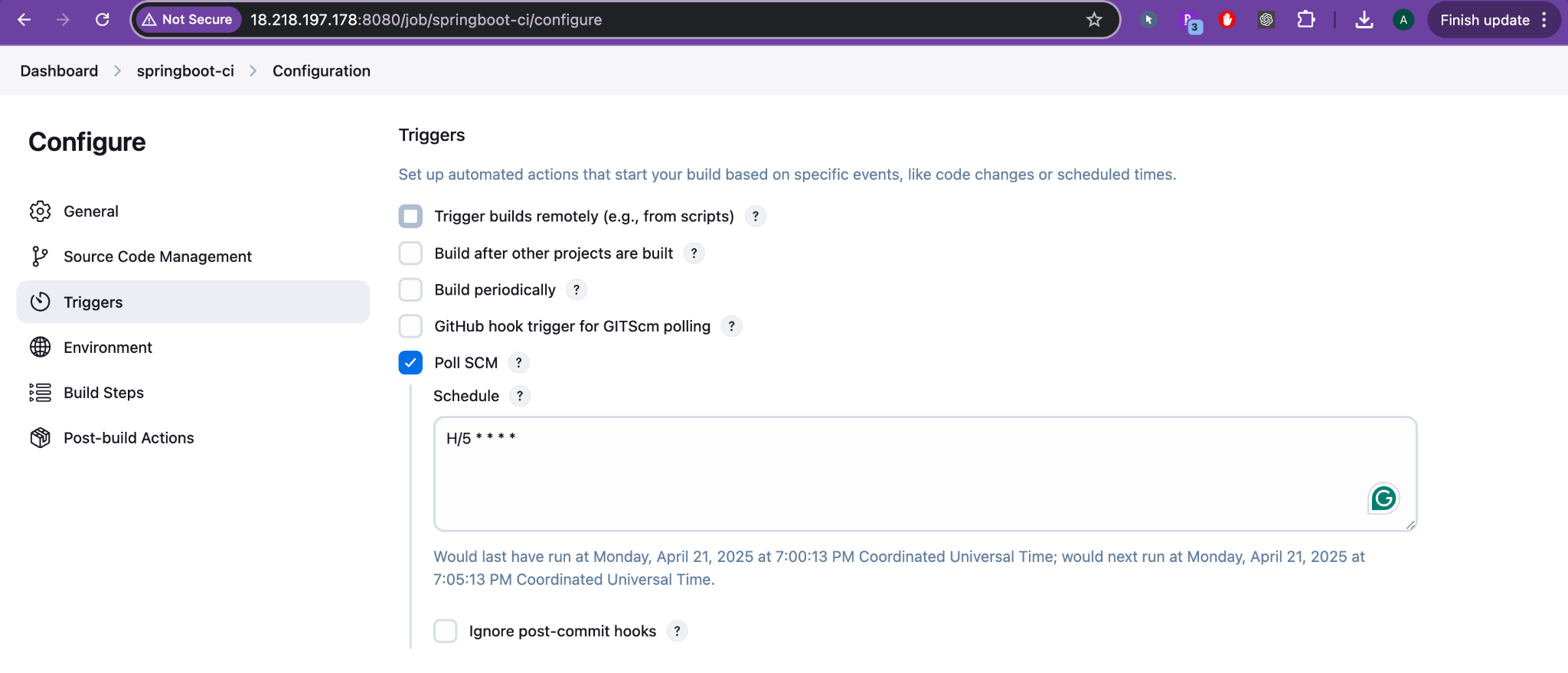
Poll SCM:

H/5 \* \* \* \*

* → Check for new commits every 5 minutes.
* GitHub hook trigger for GITScm polling:  
   → Automatically trigger a build when someone pushes code to GitHub (needs webhook set in GitHub).

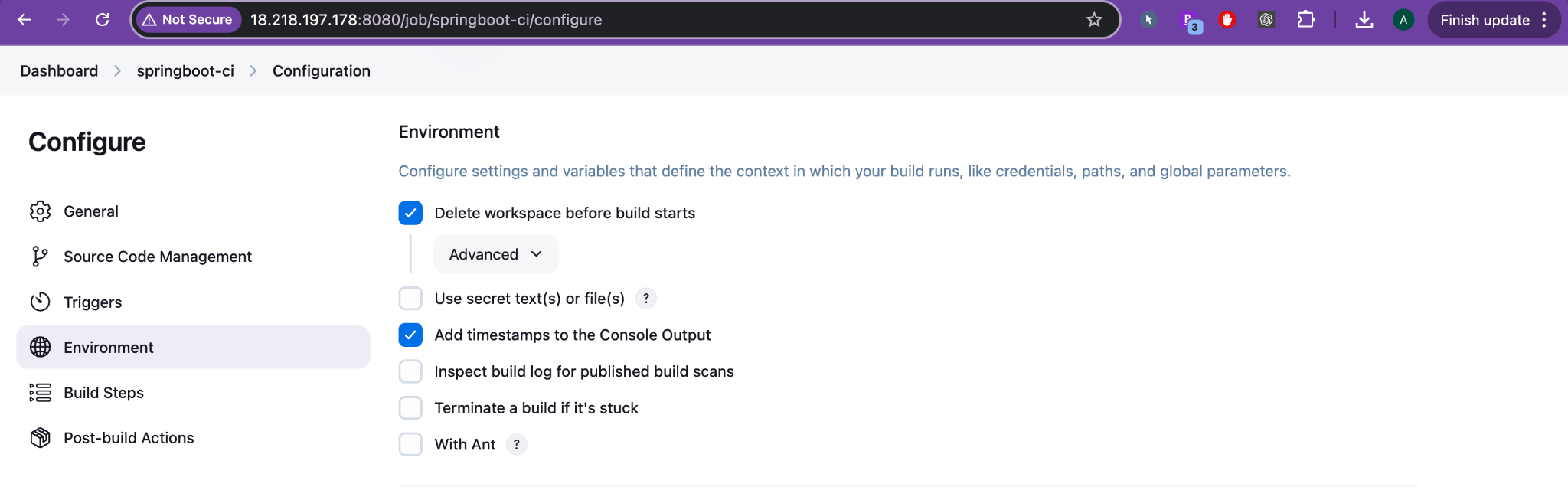
✅ **Real-World Use**:

Automatically run CI pipeline whenever developers push code.



### **🔹 Step 6: Build Environment (Screenshot 5)**

* Check:  
  + Delete workspace before build: Clean start.
  + Add timestamps: Easier log reading.



### **🔹 Step 7: Build Steps (Screenshot 5)**

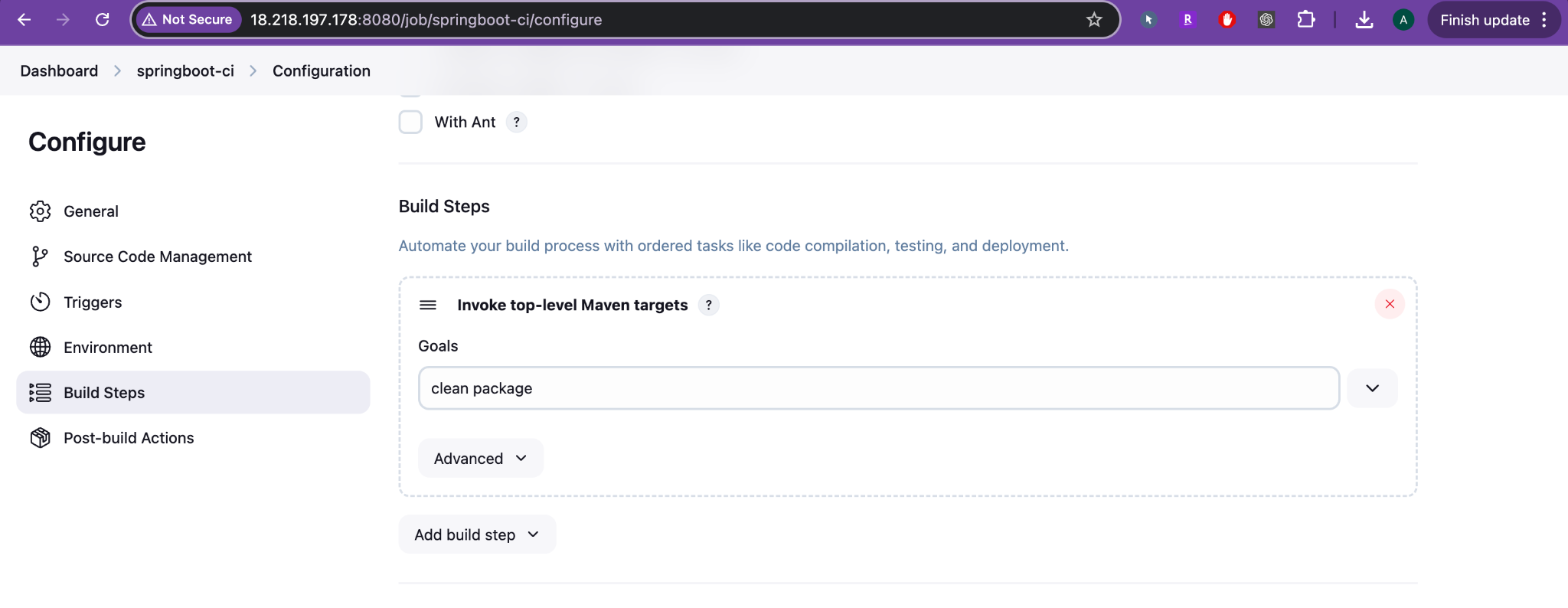
👉 **Add Build Step > Invoke top-level Maven targets**

**Goals**:  
clean package

**Root POM**:  
pom.xml

✅ What Happens:

This will compile the code, run unit tests, and package the app into a JAR or WAR file using mvn clean package.



### **🔹 Step 8: Post-Build Actions (Screenshot 6)**

#### **1. 📦 Archive the Artifacts**

Files to archive:  
  
target/\*.jar

✅ Why?

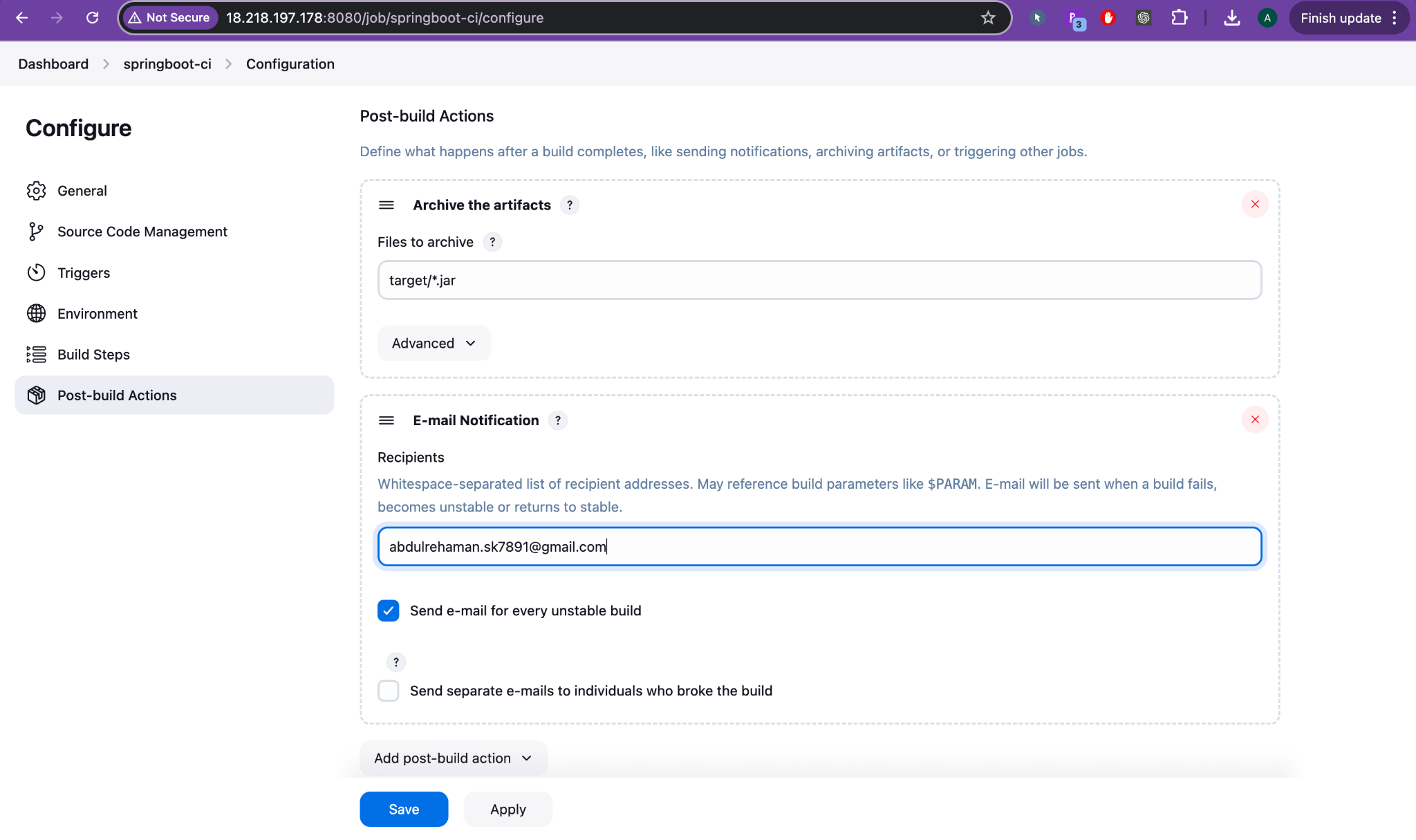
This makes the built JAR downloadable from Jenkins UI and useful for deployment steps.

#### **2. 📧 Email Notification**

* Add "Email Notification":  
  + Recipients: dev-team@example.com
  + Notify on failure/success

✅ Why?

Team is immediately informed if build fails or succeeds.



When someone pushes new code to GitHub:

1. Jenkins gets triggered via webhook or polling.
2. It pulls the latest code.
3. Runs mvn clean package.
4. Runs unit tests (fails the build if tests fail).
5. Archives the generated .jar.
6. Sends build status emails to the team.

Folder Structure of Project (GitHub)

springboot-webapp/

├── src/

│ └── main/

│ └── java/

│ └── com/example/

│ └── App.java

├── src/

│ └── test/

│ └── java/

│ └── com/example/

│ └── AppTest.java

├── pom.xml

