**Challenge-4**

1. **Create one Declarative pipeline job**

# <https://github.com/sunil-th/sunil-app.git>

# Jenkins Setup for Your Jenkinsfile

### 1. ****Install Required Plugins****

Go to **Manage Jenkins → Plugins** and ensure these are installed:

* **Pipeline** (Declarative + Scripted)
* **Pipeline Utility Steps** (for readMavenPom, findFiles)
* **Nexus Artifact Uploader**
* **SonarQube Scanner for Jenkins**
* **Slack Notification**

### 2. ****Configure Global Tools****

**Manage Jenkins → Global Tool Configuration**

* **Maven:**
  + Name: MVN\_HOME (matches your Jenkinsfile)
  + Install automatically or point to /usr/share/maven
* **SonarQube Scanner:**
  + Name: sonar\_scanner (matches your Jenkinsfile)
  + Provide installation path or let Jenkins auto-install

### 3. ****Add Server Integrations****

**a) SonarQube**

* Go to **Manage Jenkins → Configure System → SonarQube servers**
  + Name: sonarqube-server (matches your Jenkinsfile)
  + Add Server URL (e.g., http://<sonar-host>:9000)
  + Add authentication token (create in SonarQube UI → My Account → Security → Tokens)

**b) Slack**

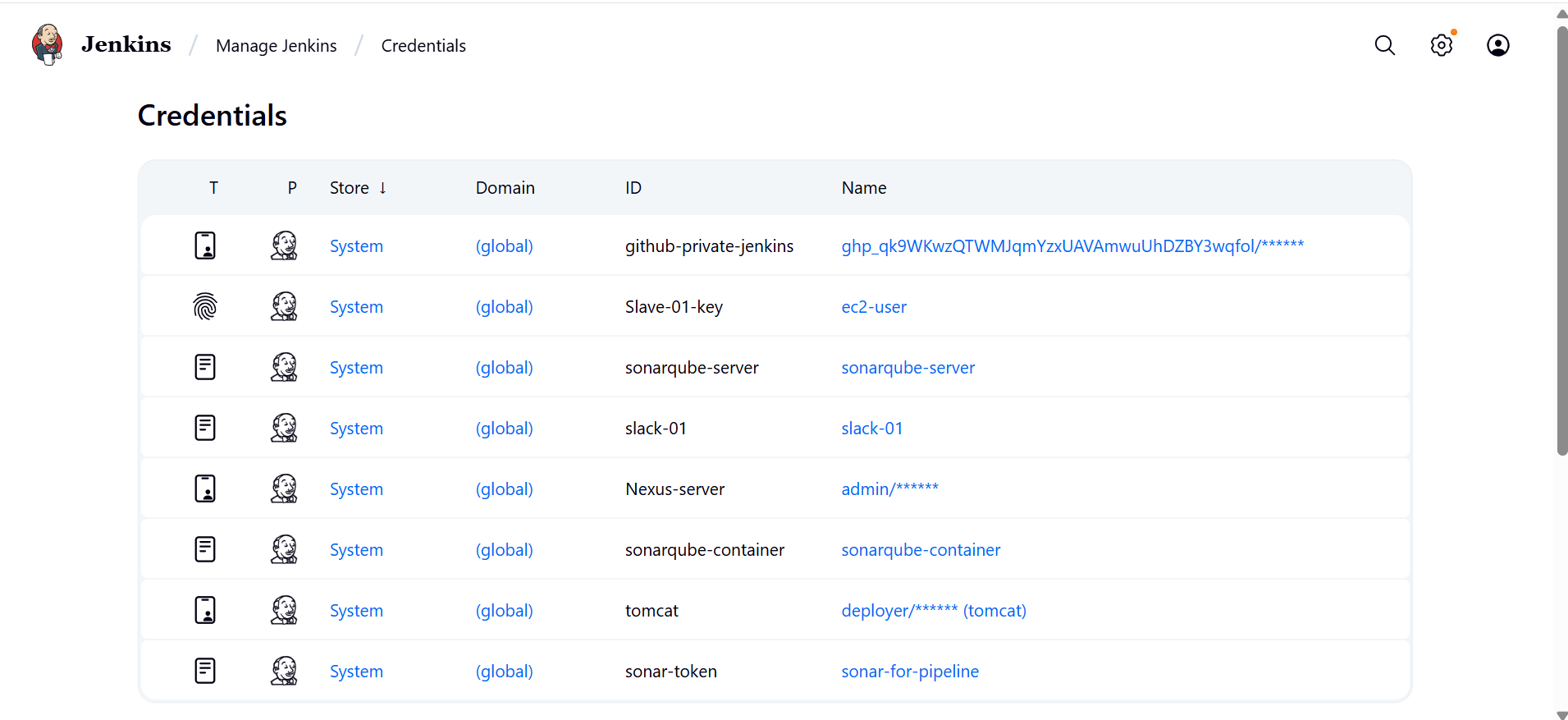
* Go to **Manage Jenkins → Configure System → Slack**
  + Workspace: your Slack team domain
  + Integration token (from Slack App)
  + Default channel: #jenkins-integration
* Test connection to ensure Jenkins can post.

### 4. ****Set Credentials****

**Manage Jenkins → Credentials → System → Global credentials**

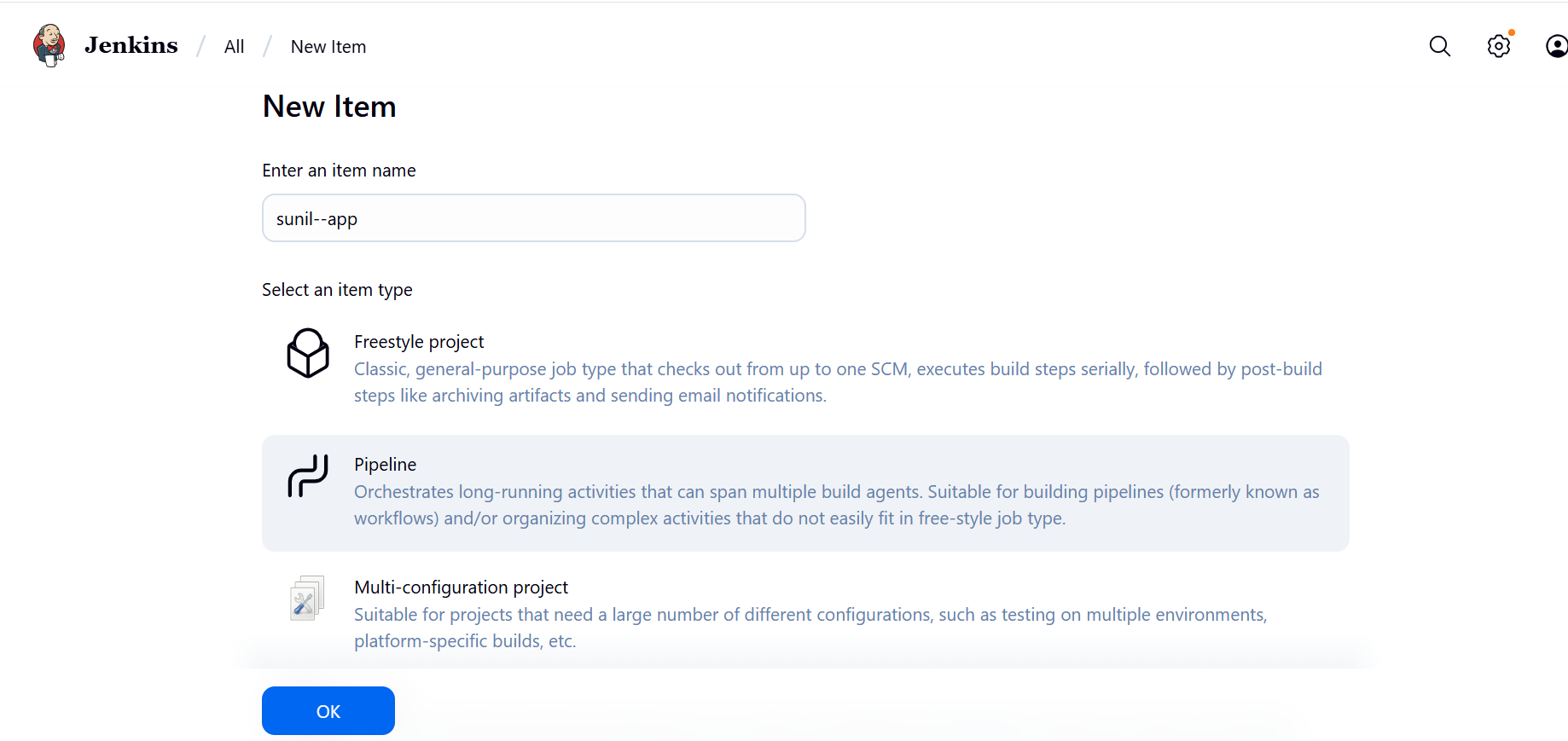
Add the following IDs (must match your Jenkinsfile):

* **Nexus-server**
  + Type: Username with password
  + Username: Nexus repo user
  + Password: Nexus repo password
* **tomcat**
  + Type: Username with password
  + Username: Tomcat manager user
  + Password: Tomcat manager password



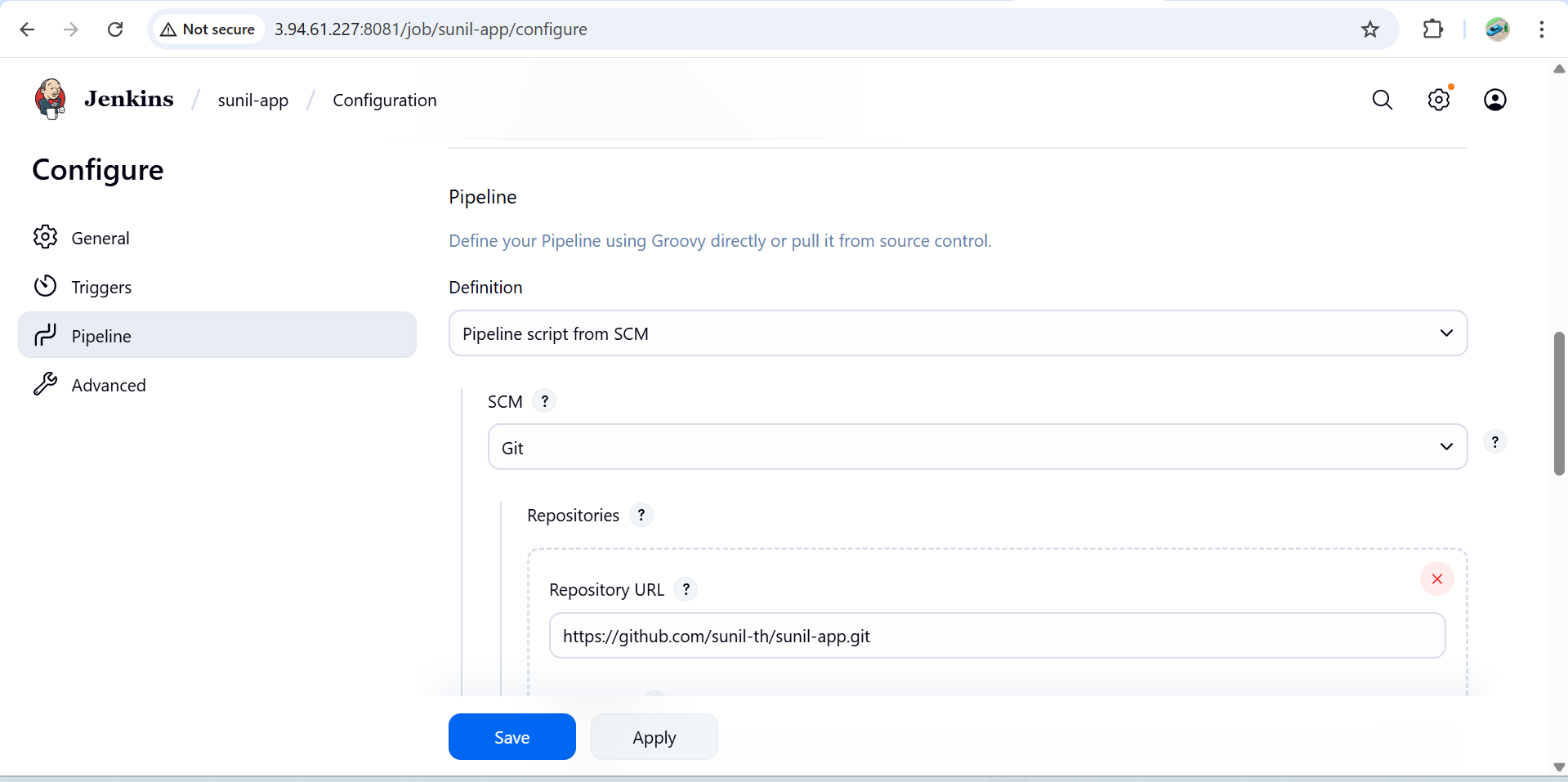
**5.Create a New Pipeline Job**

1. Open Jenkins dashboard → **New Item**.
2. Enter a job name (e.g., simple-customerapp-pipeline).
3. Select **Pipeline** → Click **OK**.

****

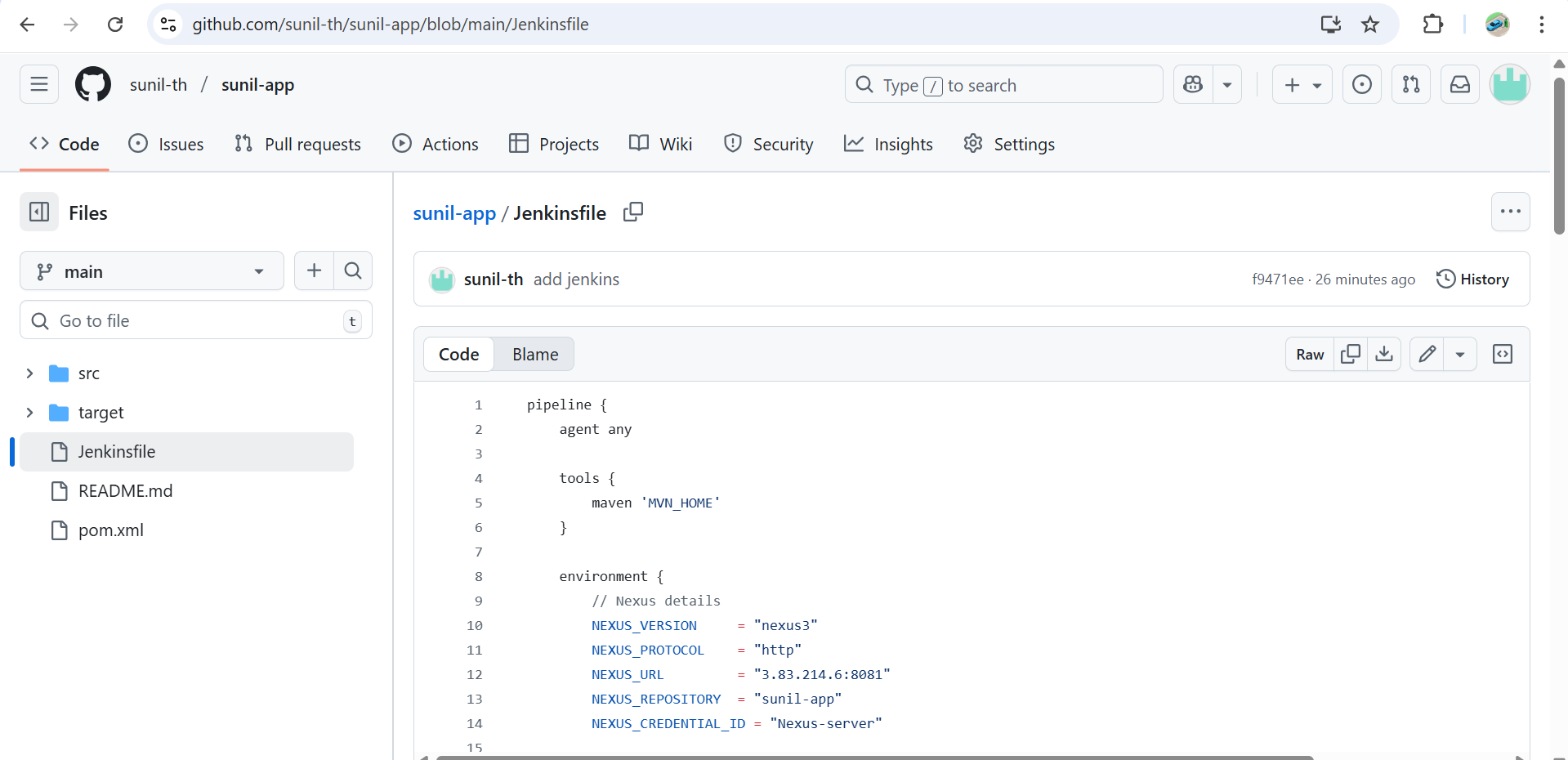
**Pipeline from SCM**

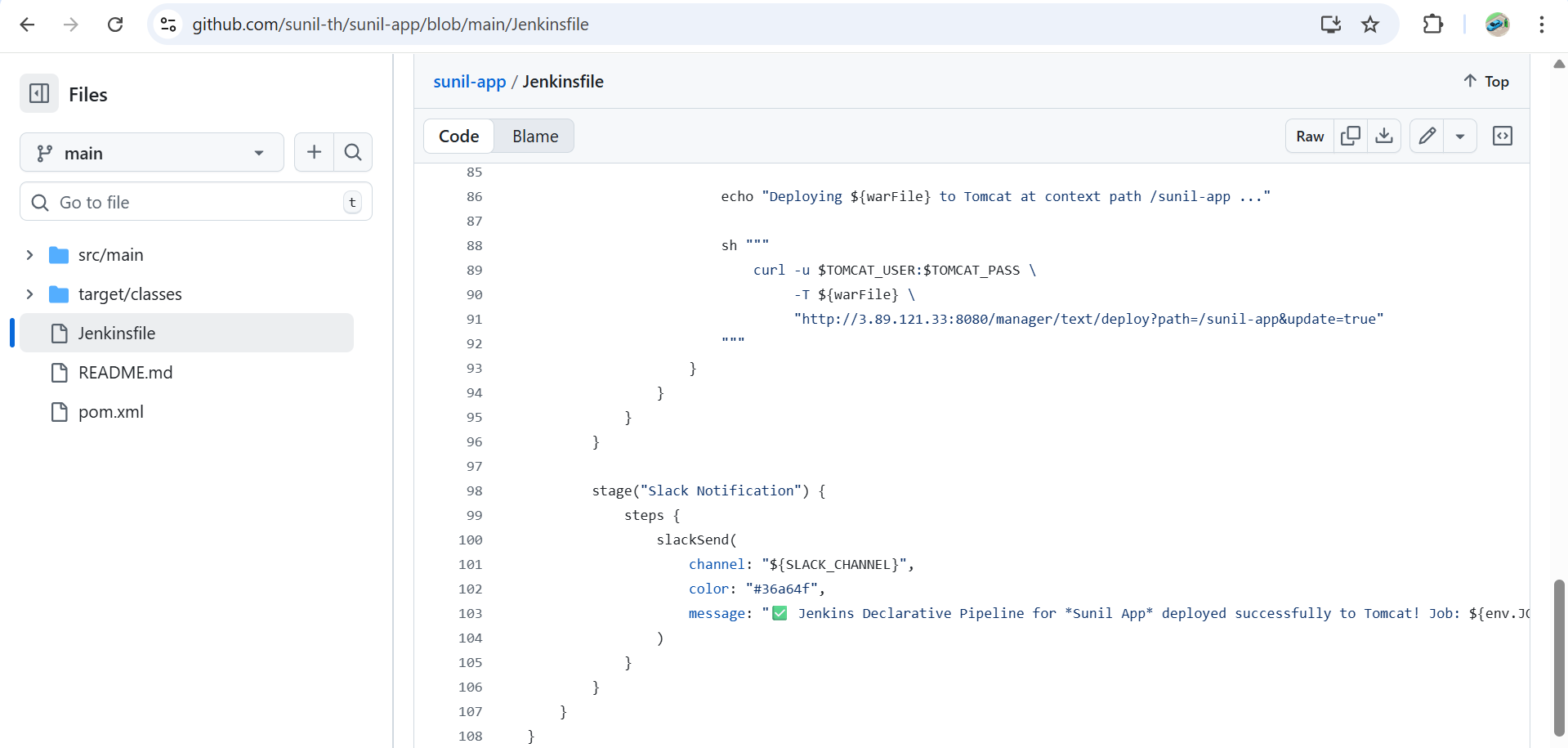
* **SCM:** Git
* **Repository URL:** https://github.com/betawins/sabear\_simplecutomerapp.git
* **Branch Specifier:** \*/main

****

**Drop this** Jenkinsfile **into the repo**

Stages: **Git Clone → SonarQube → Maven Compile → Nexus Upload → Deploy on Tomcat → Slack**

****

****

# How the Stages Work After Setup

1. **Clone Code**
   * Uses Jenkins Git plugin, no creds required since repo is public.
2. **Maven Build**
   * Compiles + packages the WAR, ignores test failures.
3. **SonarQube Scan**
   * Runs sonar-scanner CLI with your provided parameters.
   * Uses the sonarqube-server config and scanner tool.
4. **Publish to Nexus**
   * Reads pom.xml → finds WAR in target/
   * Uploads artifact + POM to Nexus repo pipe-snapshots at http://3.83.214.6:8081
5. **Deploy to Tomcat**
   * Grabs WAR file, extracts context name from filename,
   * Deploys to Tomcat Manager (http://3.89.121.33:8080/manager/text/deploy)
6. **Slack Notification**
   * Posts green success message with Job & Build info in #jenkins-integration

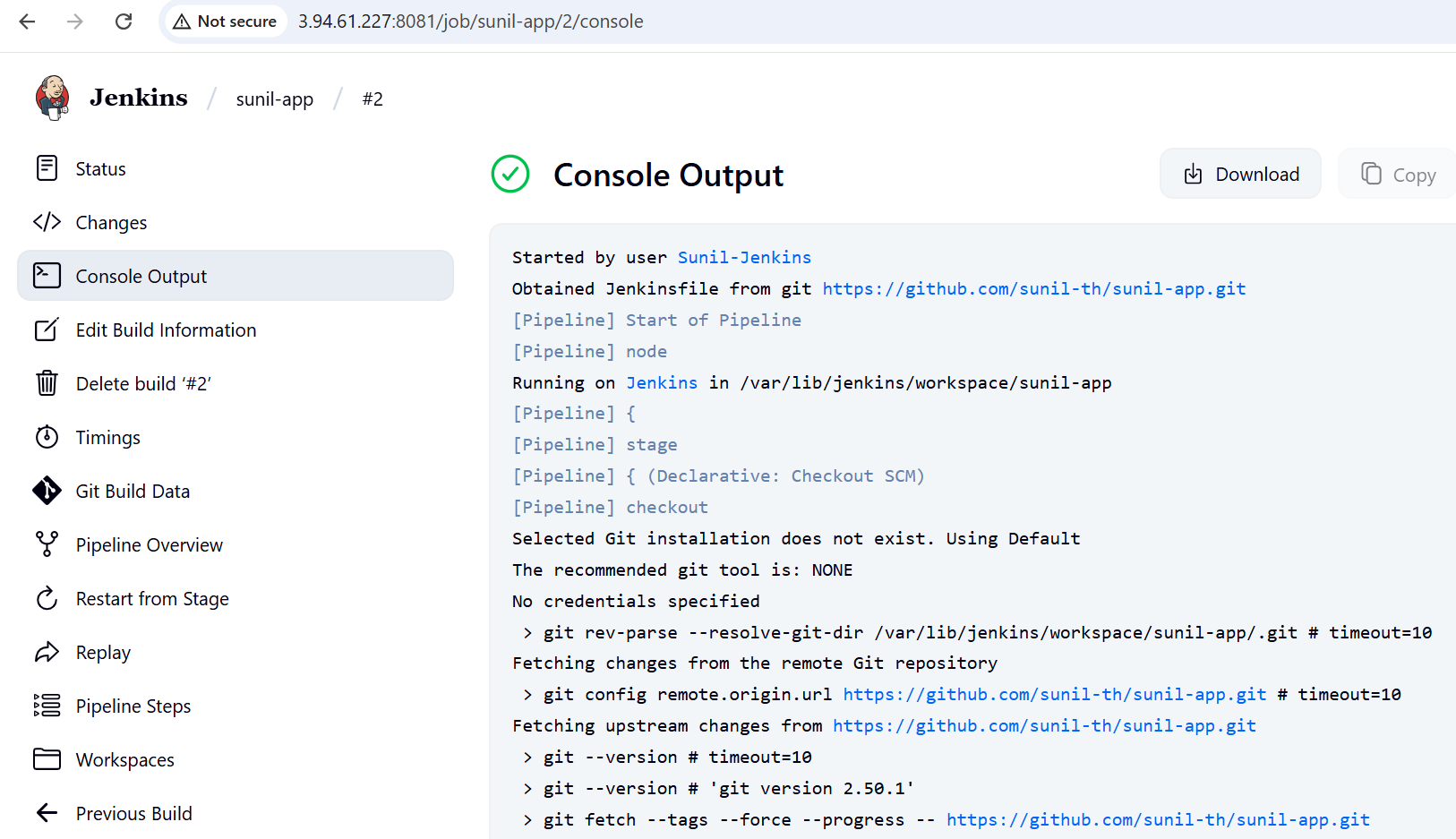
# Validation Checklist

Before running:

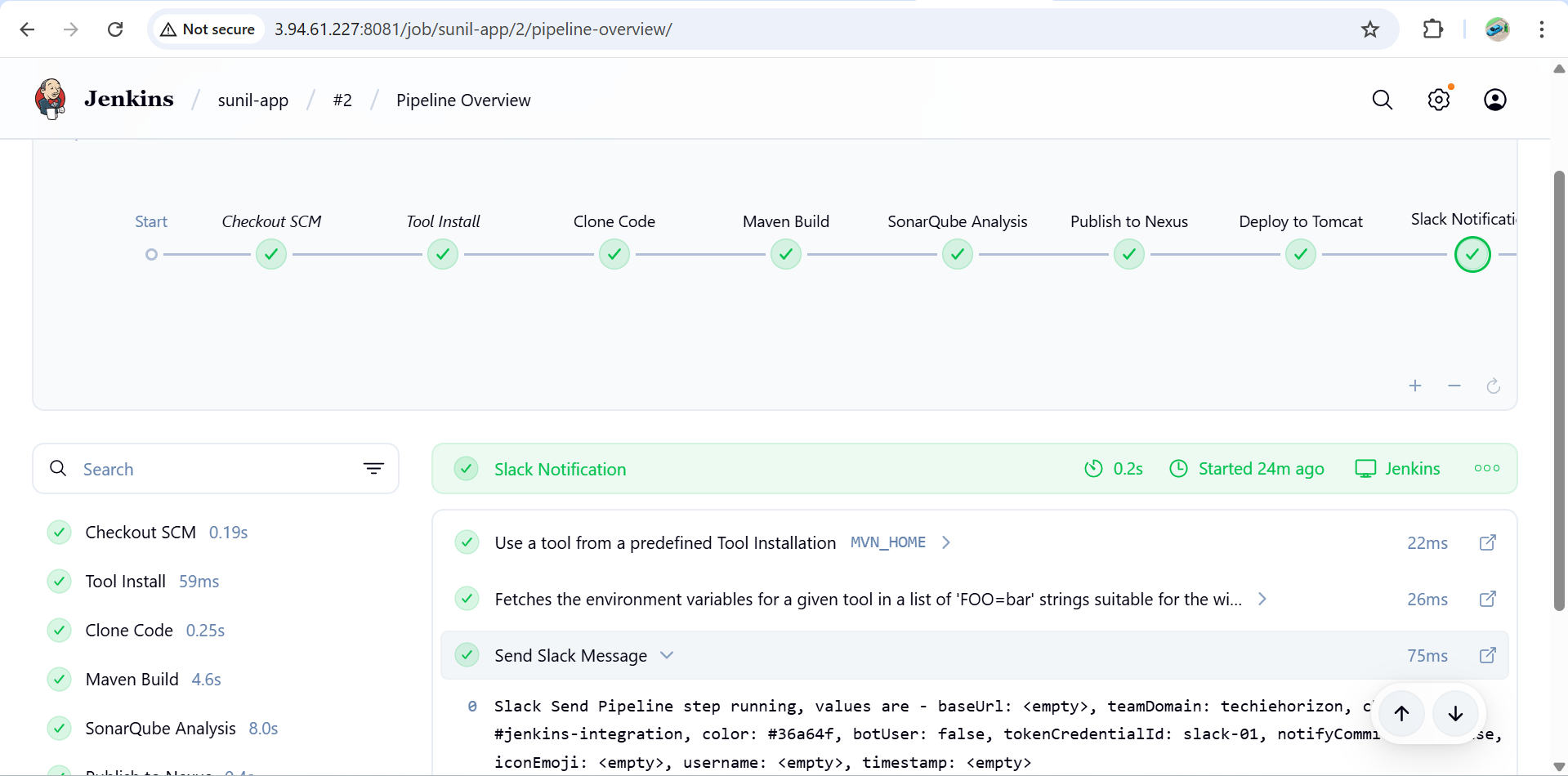
* Can Jenkins reach **SonarQube**, **Nexus**, and **Tomcat** (firewall rules)?
* Does the **Tomcat Manager app** run at http://3.89.121.33:8080/manager/html?
* Does nexusArtifactUploader show up in Jenkins pipeline syntax generator (proof plugin installed)?
* Run mvn clean install locally to ensure it builds a WAR in target/.

#### ****Save and Build****

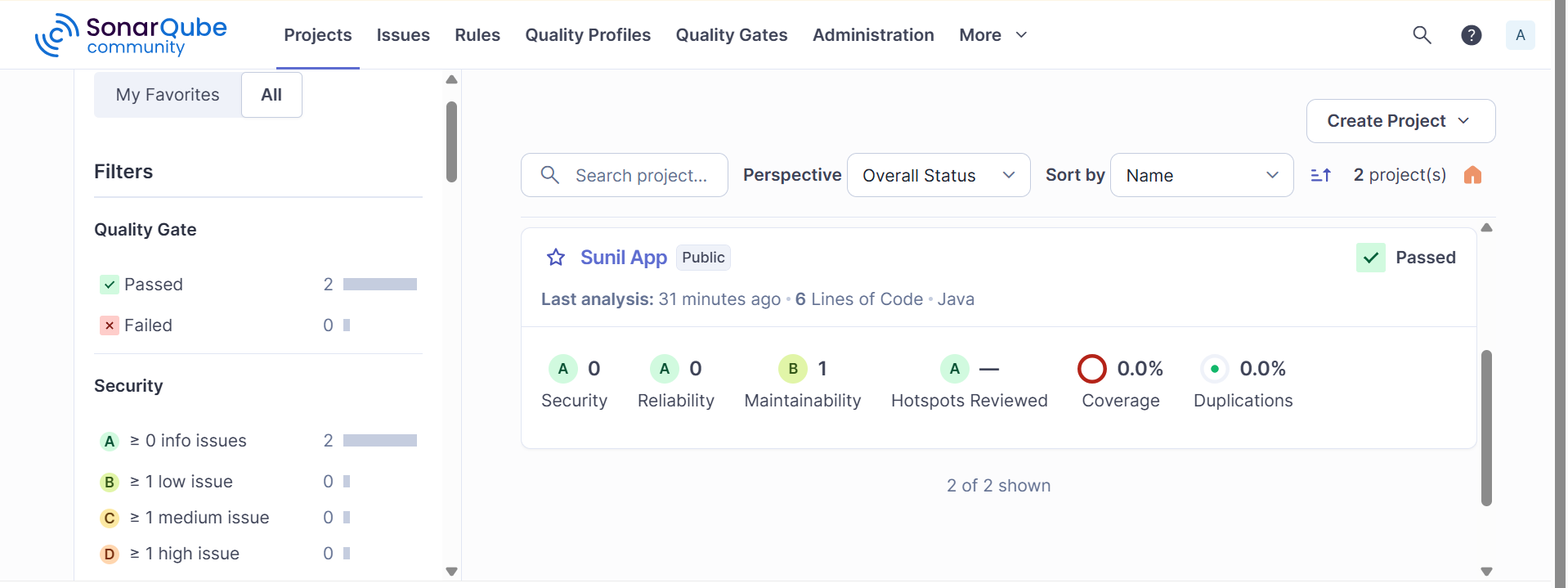
* Click **Save**.
* Run the job → Click **Build Now**.
* Check console output to verify pipeline execution.

****

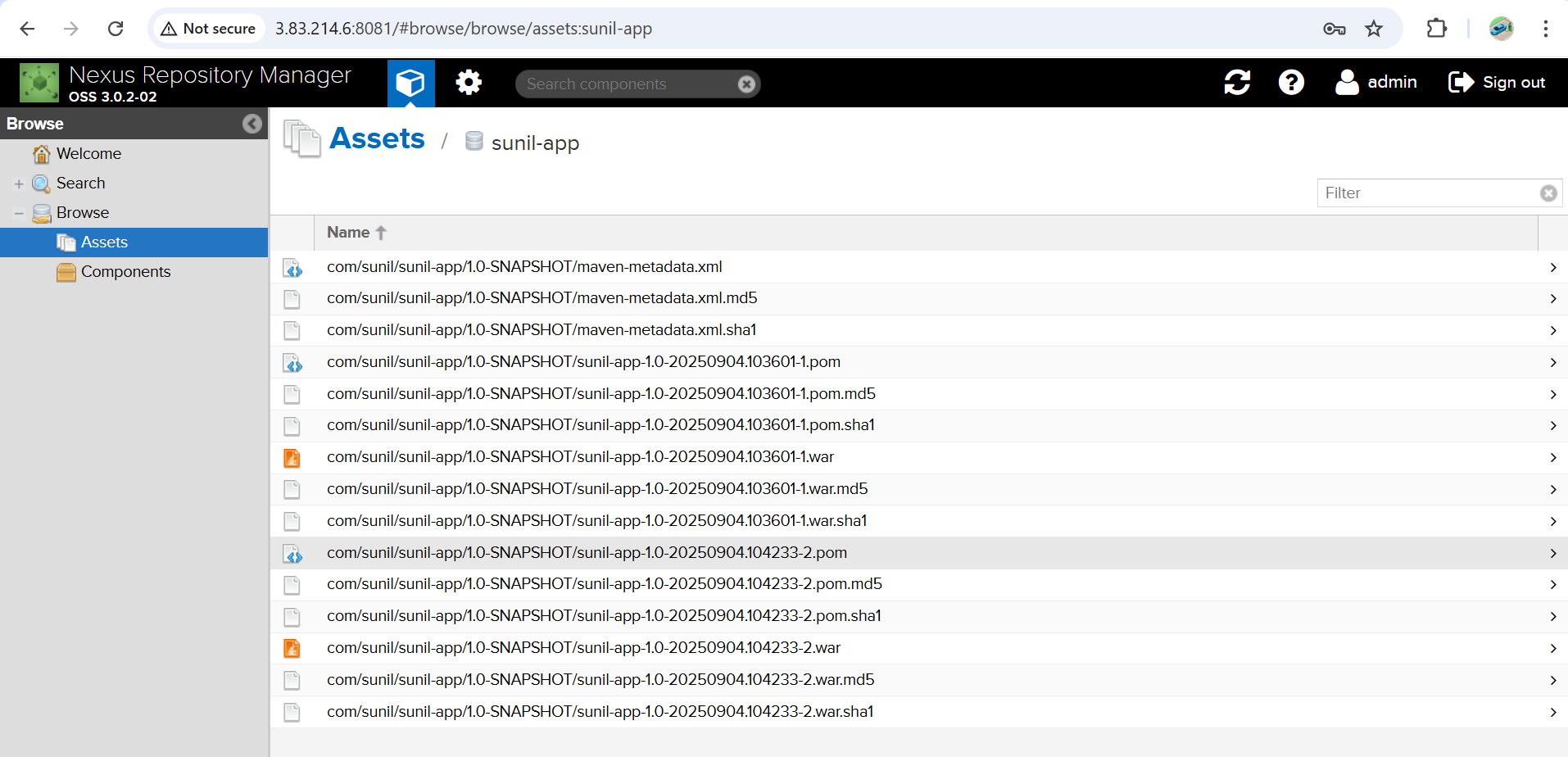
**Pipeline Flow**

****

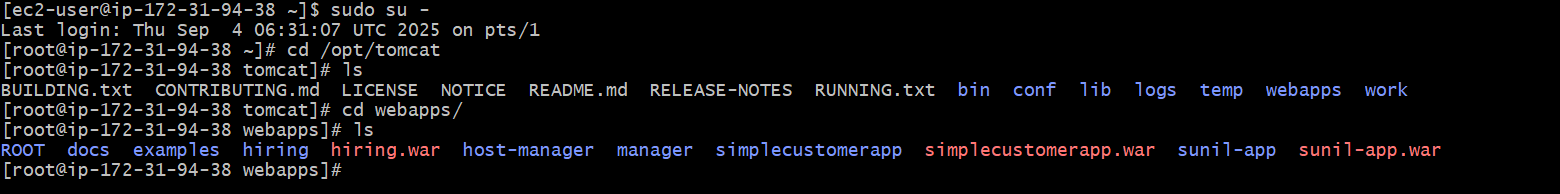
**SonarQube**

****

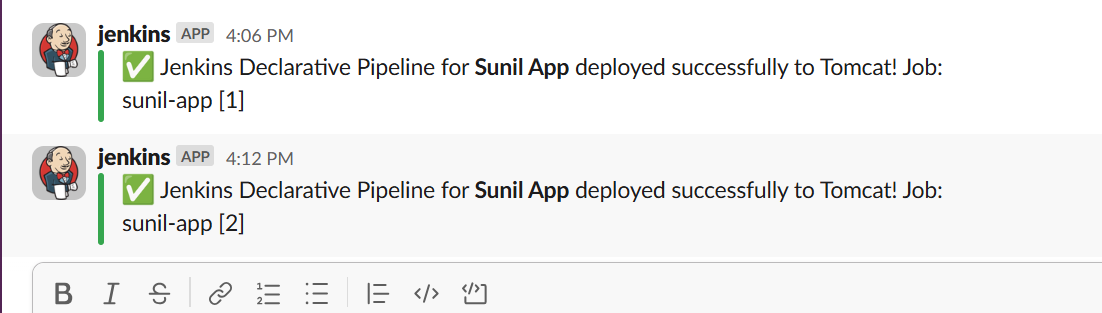
**Nexus**

****

**Tomcat Deploy**

****

**Check the slack**

****

Check on the web with the public ip and the deployed file name

<http://3.89.121.33:8080/sunil-app/>

****

1. **Create one Scripted pipeline job**

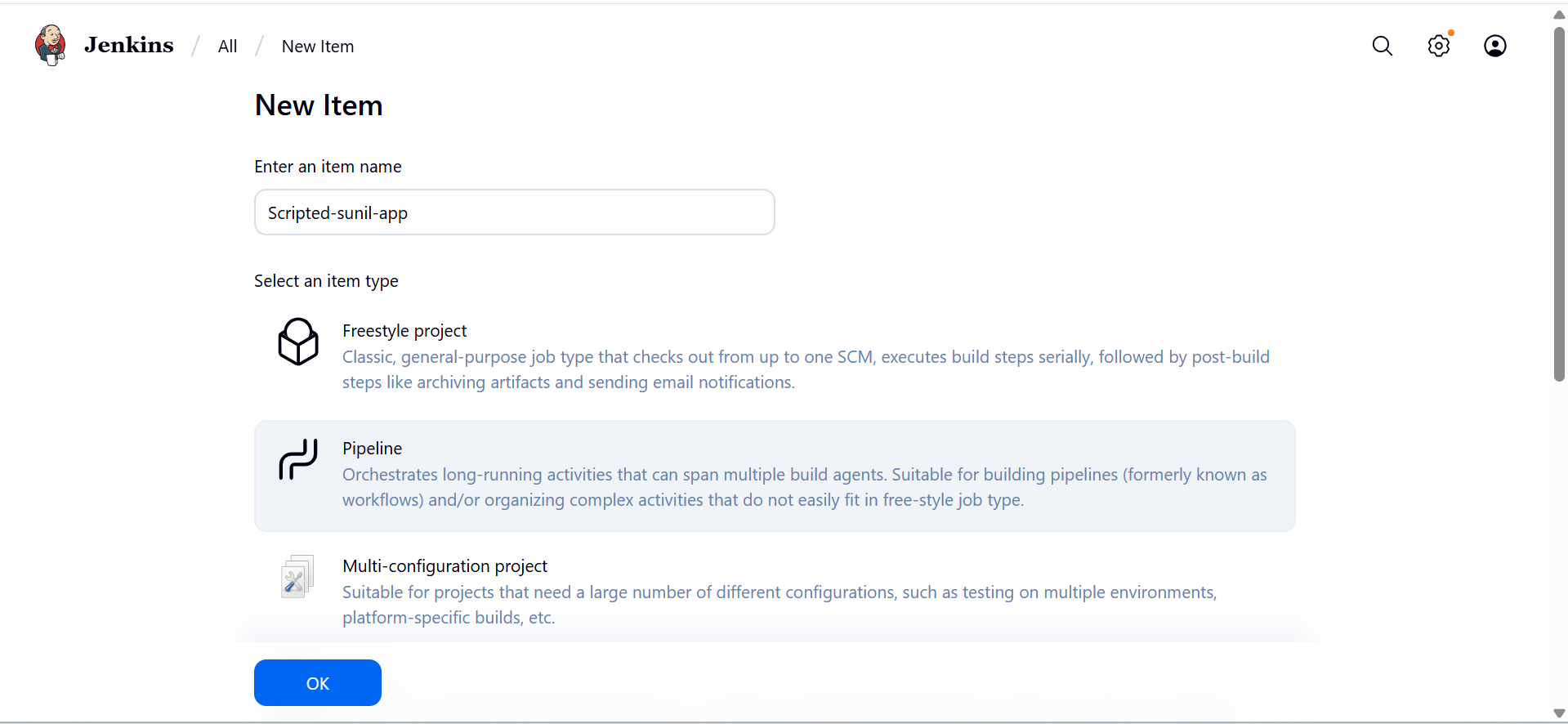
### Steps to Create a Scripted Pipeline in Jenkins

#### ****1. Log in to Jenkins****

* Open Jenkins in your browser (http://<jenkins-server>:8080/).
* Log in with your credentials.

#### ****2. Create a New Pipeline Job****

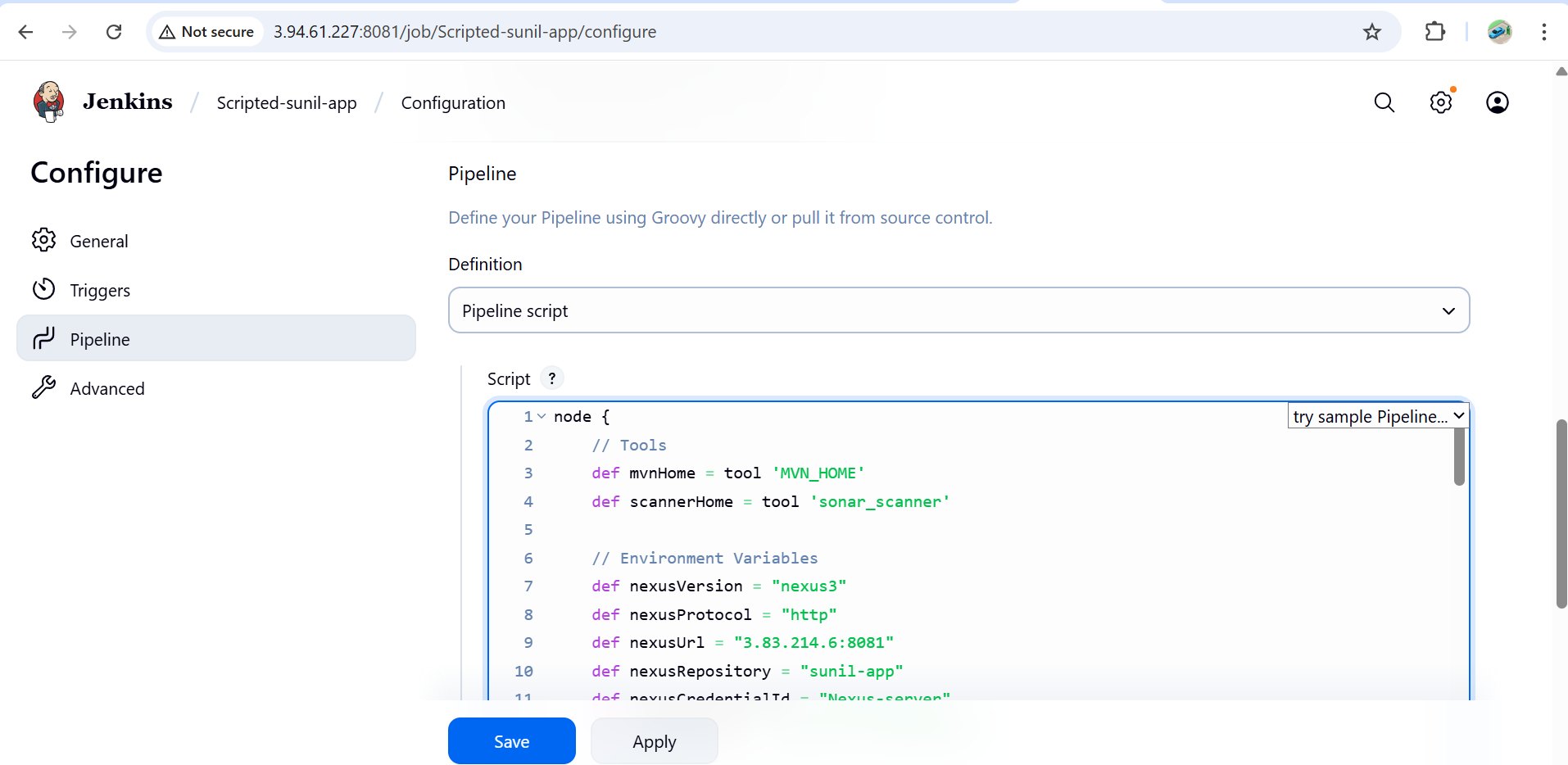
* From Jenkins Dashboard → Click **“New Item”**.
* Enter a **name** for your job (e.g., My-Scripted-Pipeline).
* Select **Pipeline** and click **OK**.

****

#### ****3. Configure Source Code Management (SCM)****

* In the job configuration:
  + Go to **Pipeline → Definition**.
  + Choose one of the two approaches:
    - **Pipeline script from SCM** (if your Jenkinsfile is already in your repo).
    - **Pipeline script** (if you want to write or paste the Scripted
    - Pipeline directly in Jenkins).

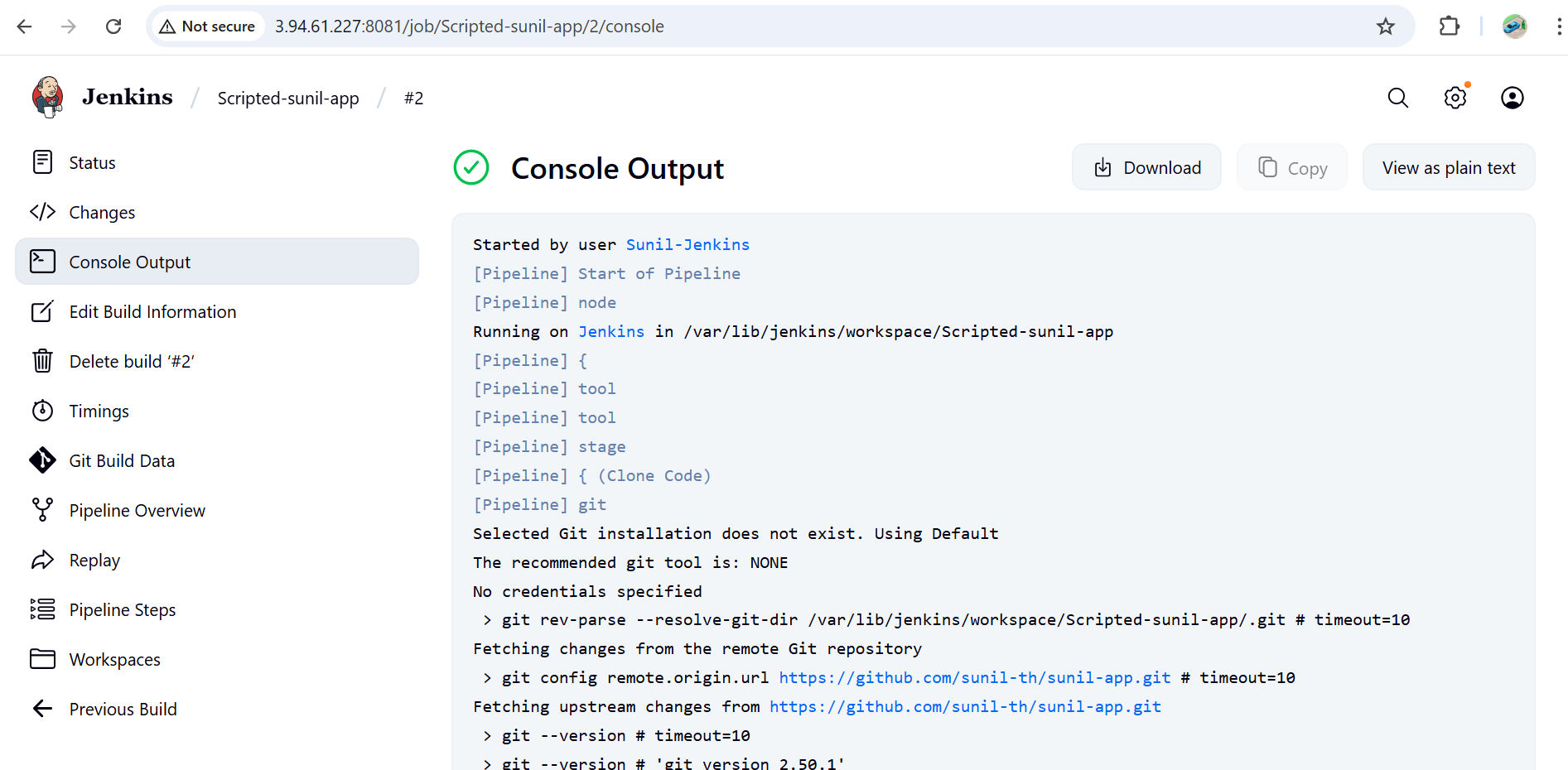
#### ****4. Scripted Pipeline Structure****

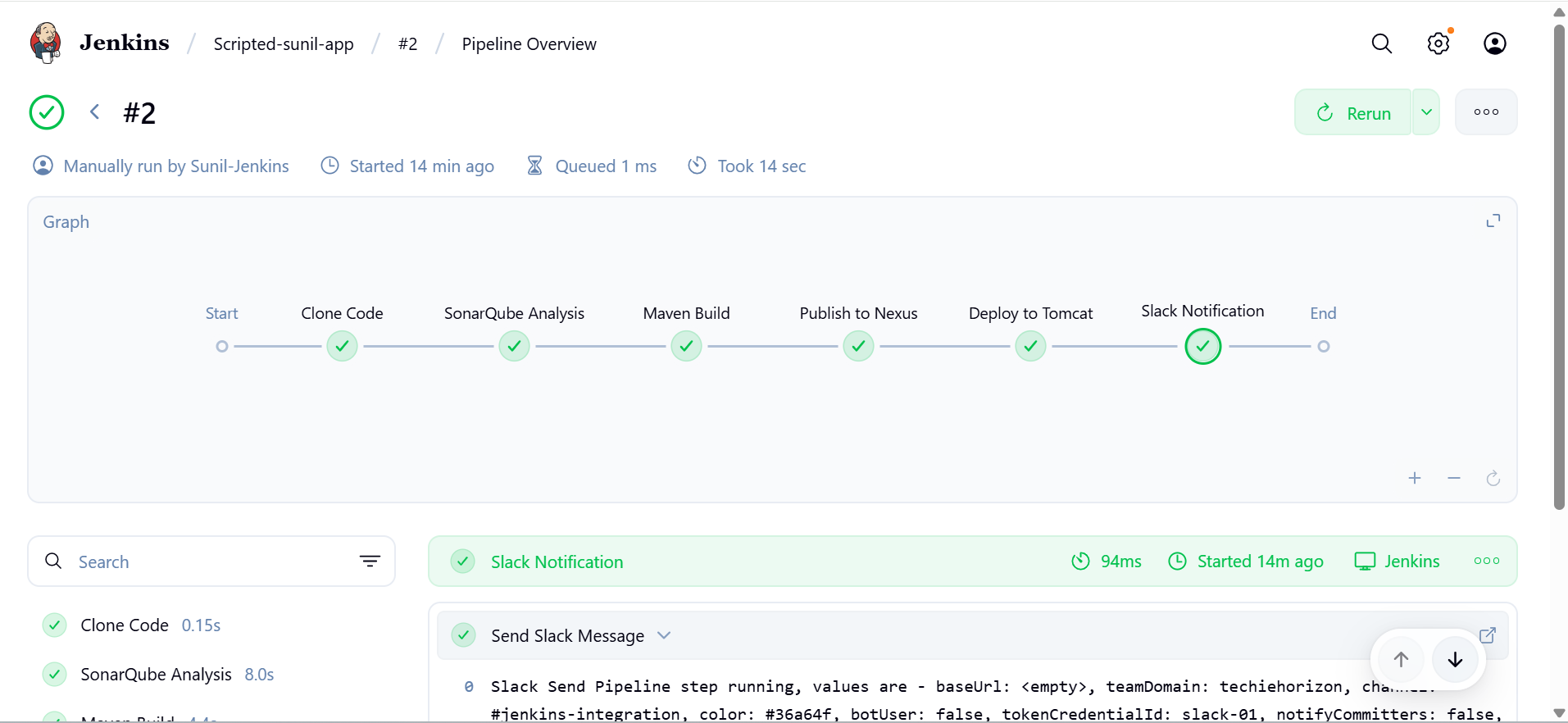
****

#### ****5. Save and Build****

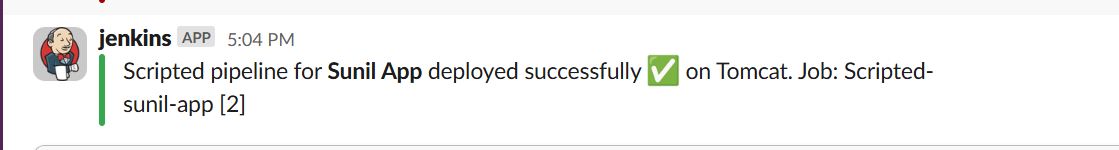
* Click **Save**.
* Run the job → Click **Build Now**.

Check console output to verify pipeline execution

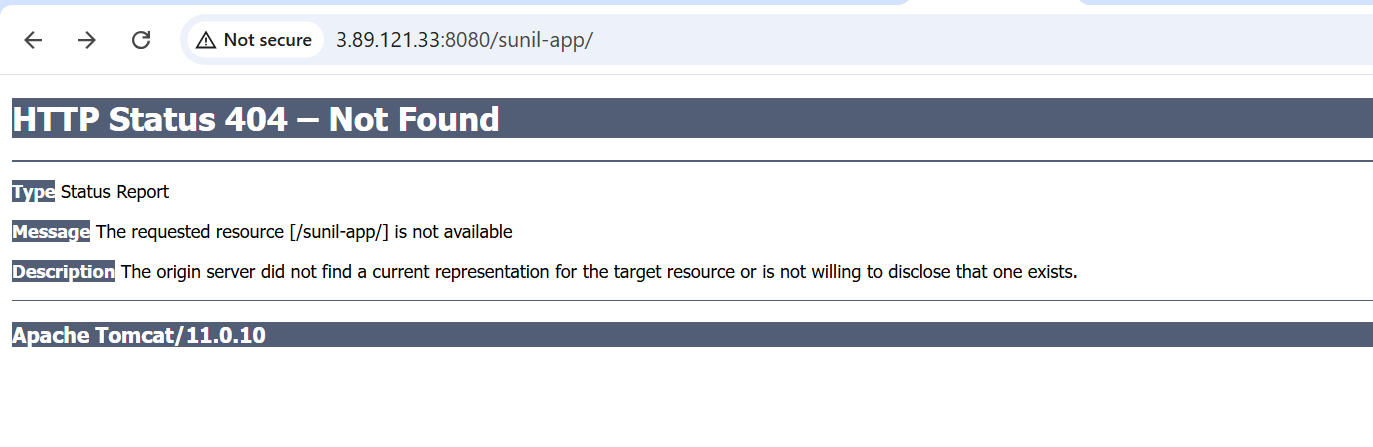
****

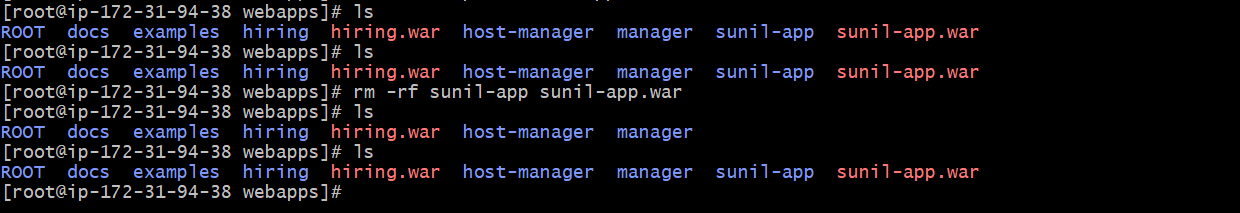
****

Slack Notification

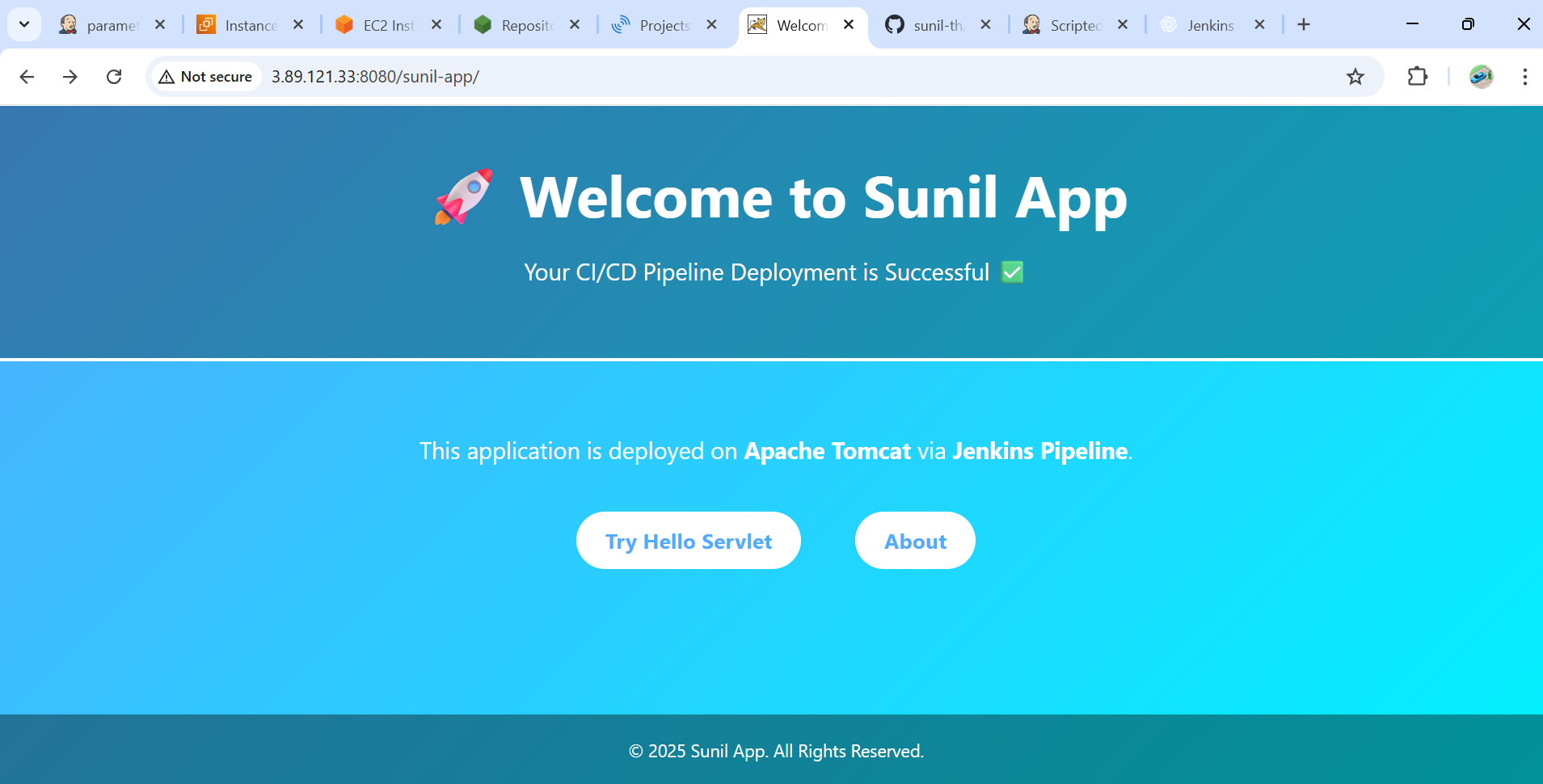
****

Remove the sunil-app file from web apps before you run because that app from declarative and check



****

After that run the scripted pipeline and check

****

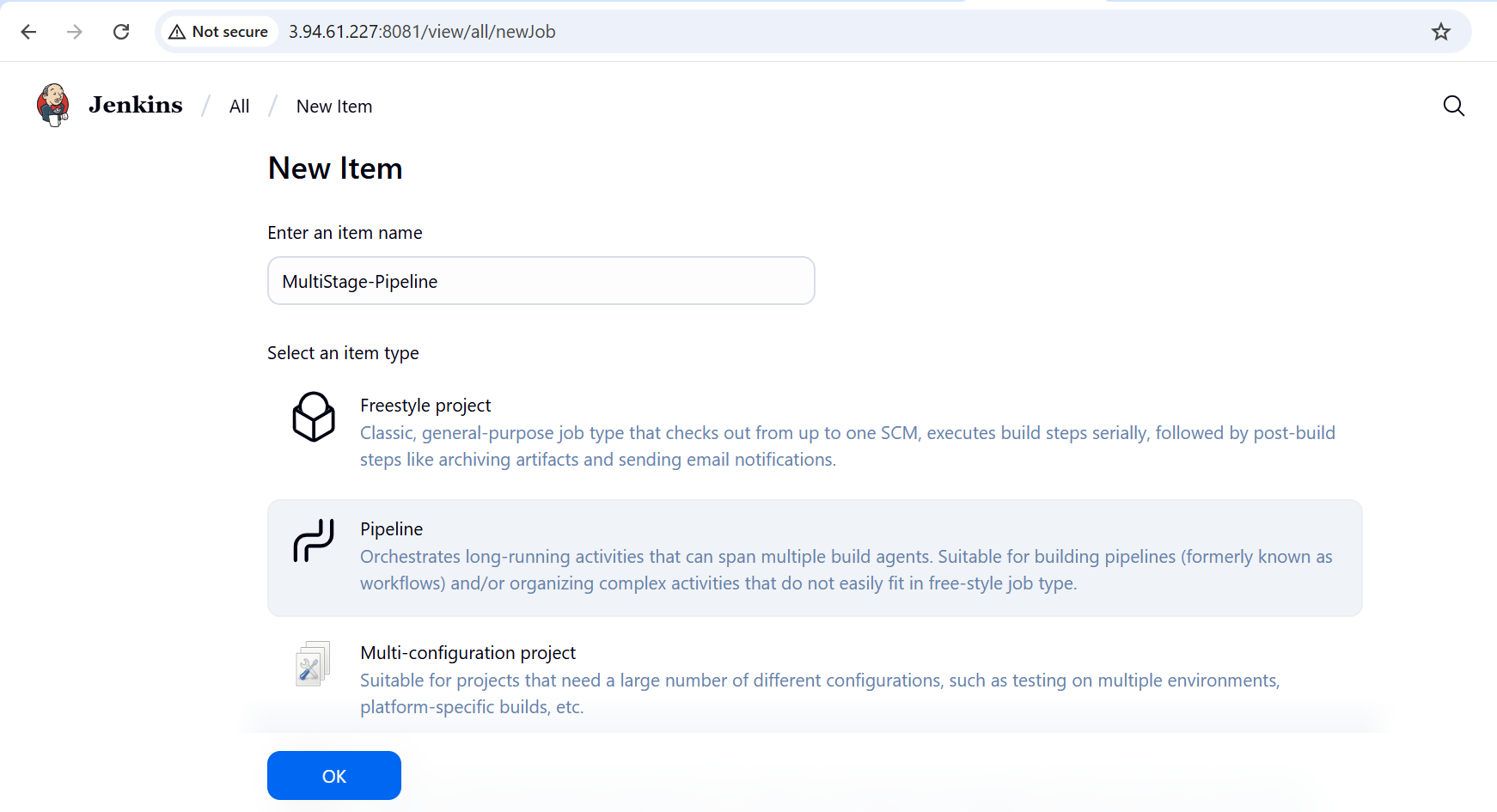
1. **Create one multi stage pipeline job**

### 1. Login to Jenkins

* Open Jenkins in your browser (usually http://<jenkins-server>:8080).
* Login with your credentials.

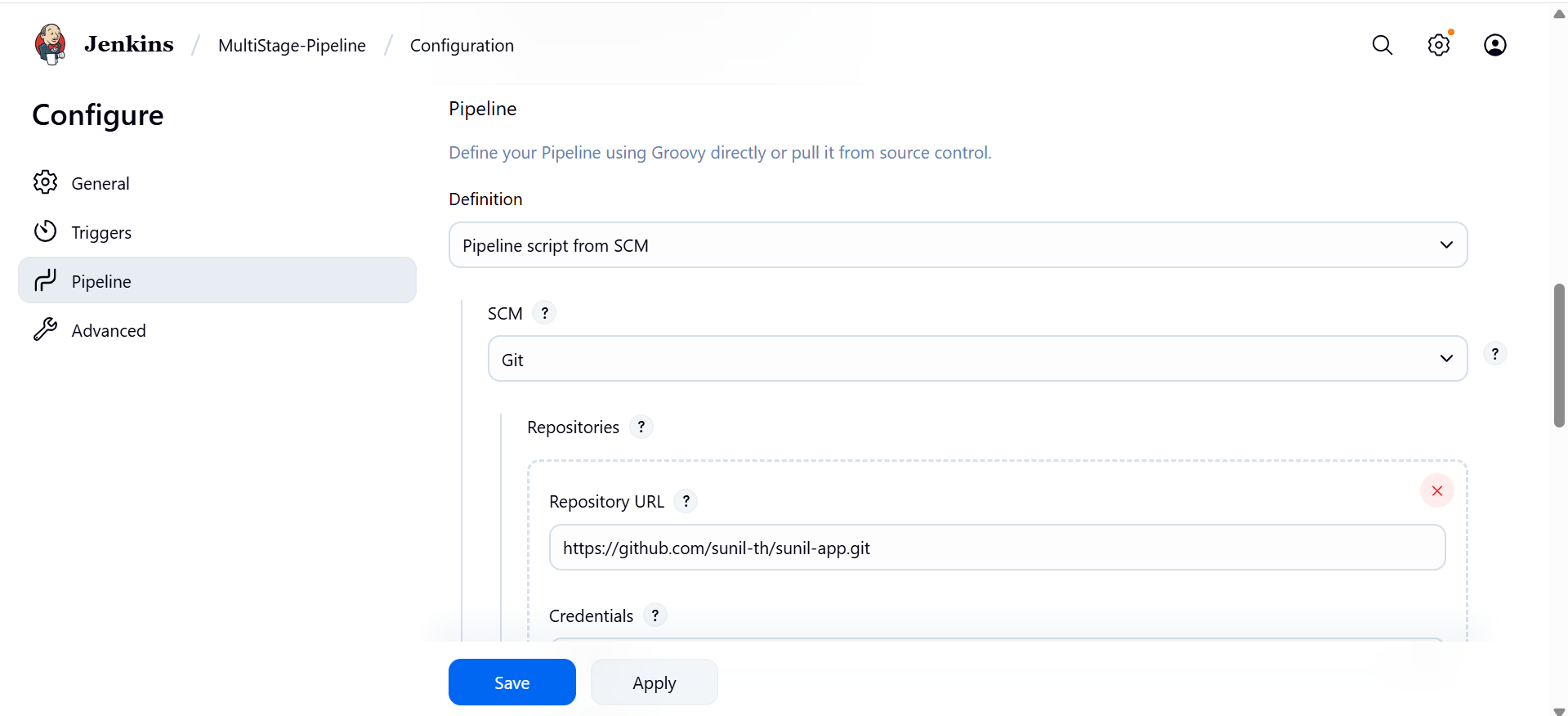
### 2. Create a New Pipeline Job

1. Click on **"New Item"**.
2. Enter a job name, e.g., MultiStage-Pipeline.
3. Select **Pipeline**.
4. Click **OK**.

****

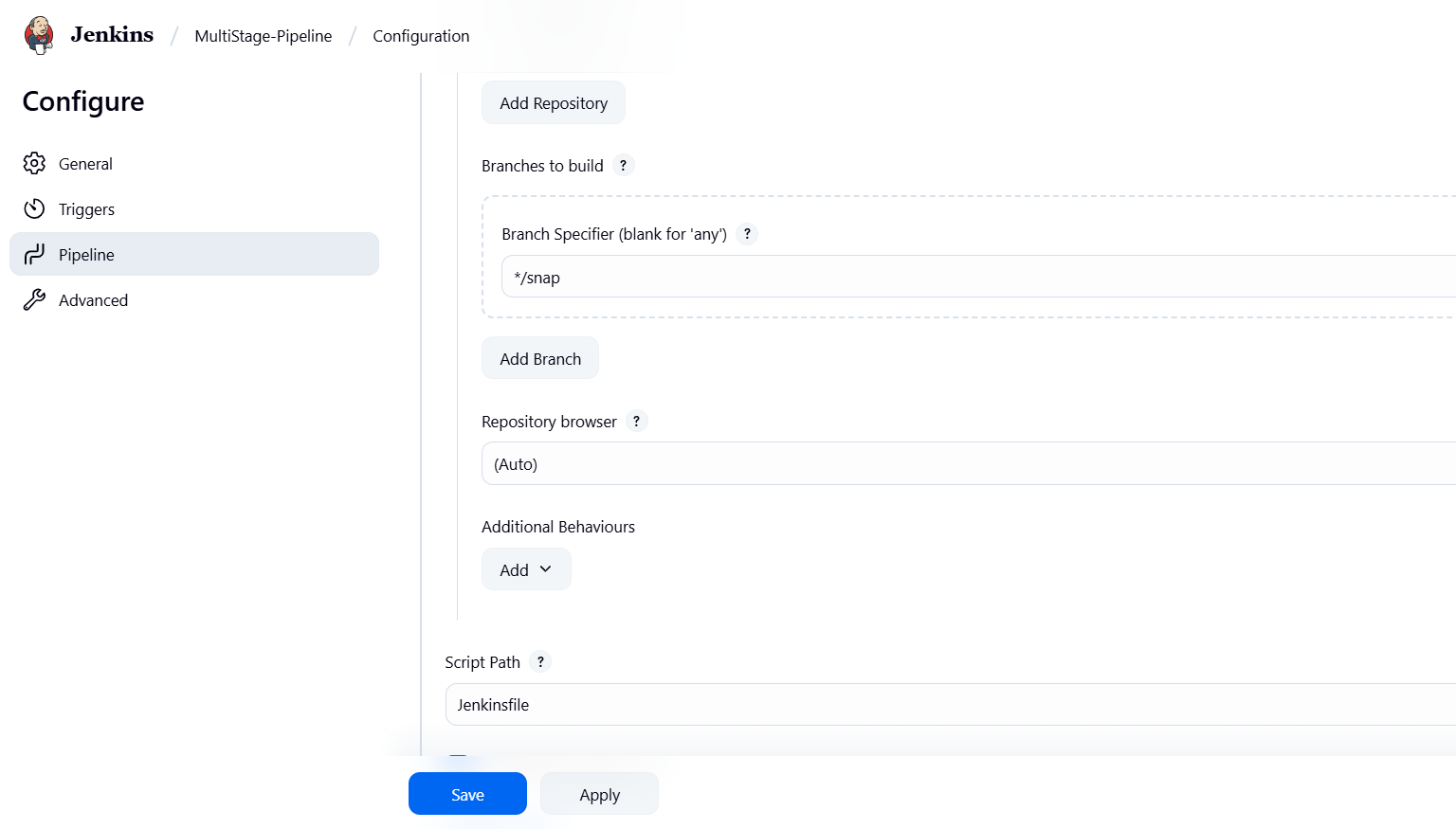
### 3. Configure the Job

* Scroll down to **Pipeline** section.
* Choose **Pipeline script** (or use Jenkinsfile from SCM if you want Git integration).
* I used scm here declarative syntax for multistage pipeline

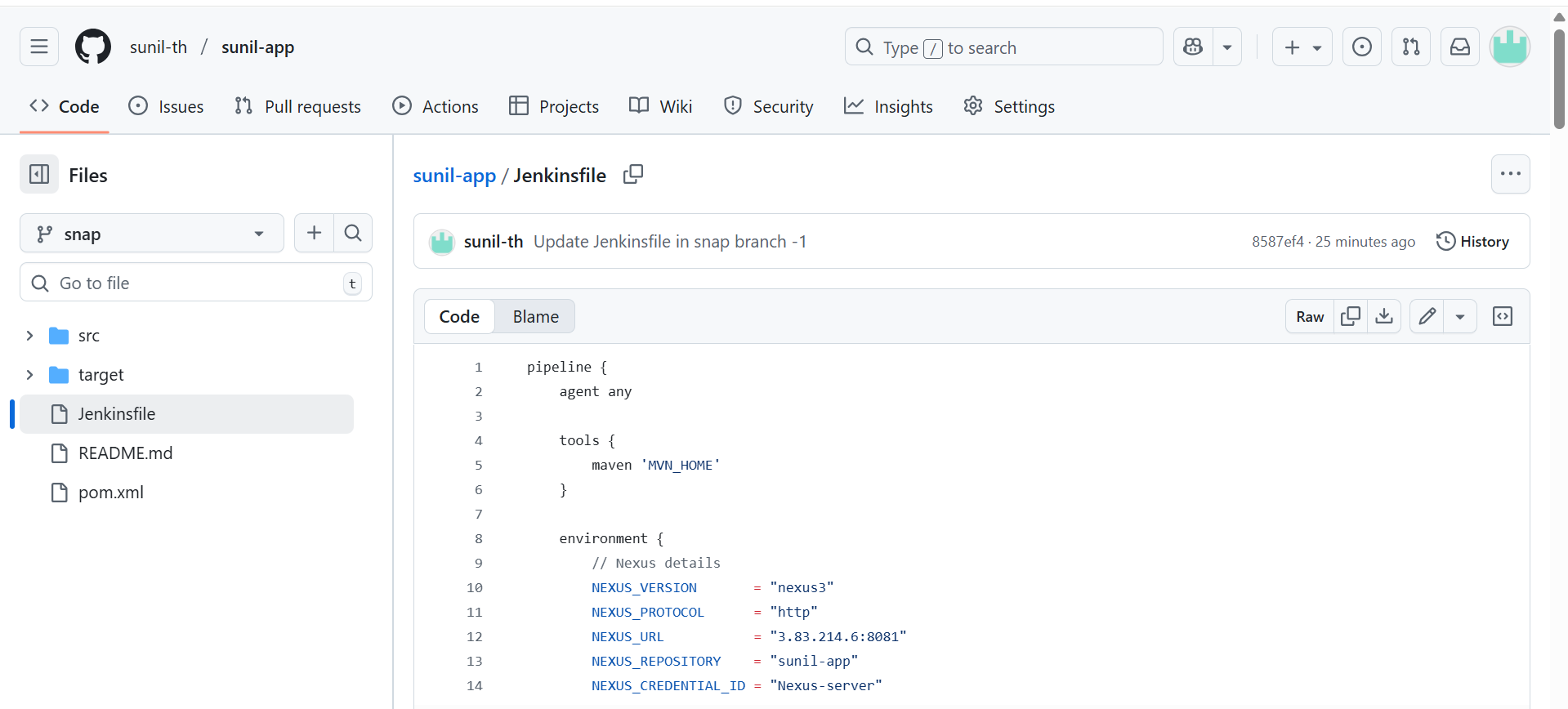
****

Mention the branch name in which branch you are adding the Jenkinsfile

And also script path give the file name in which you kept yourcode--Jenkinsfile

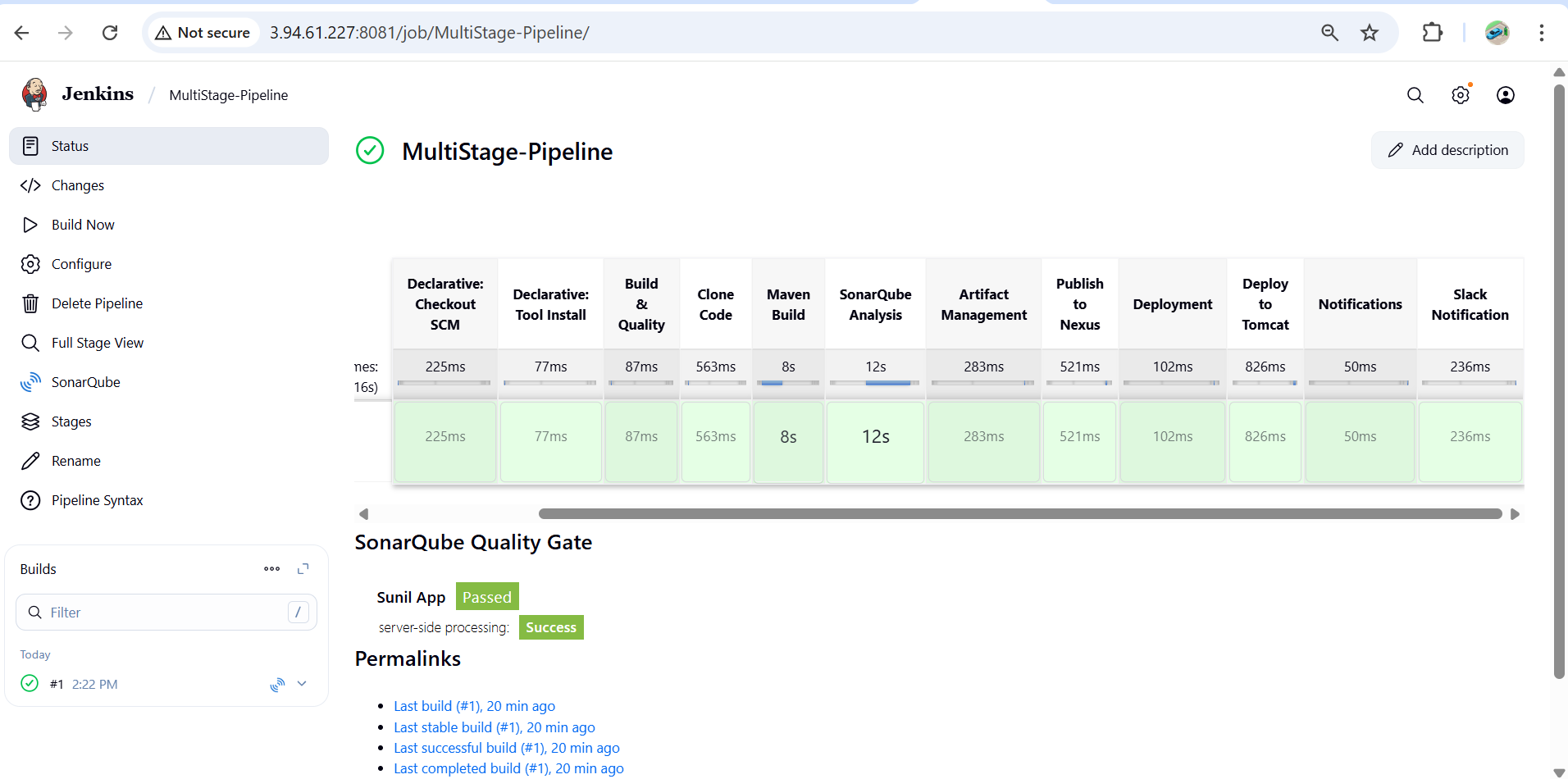
****

**Drop this** Jenkinsfile **into the repo**

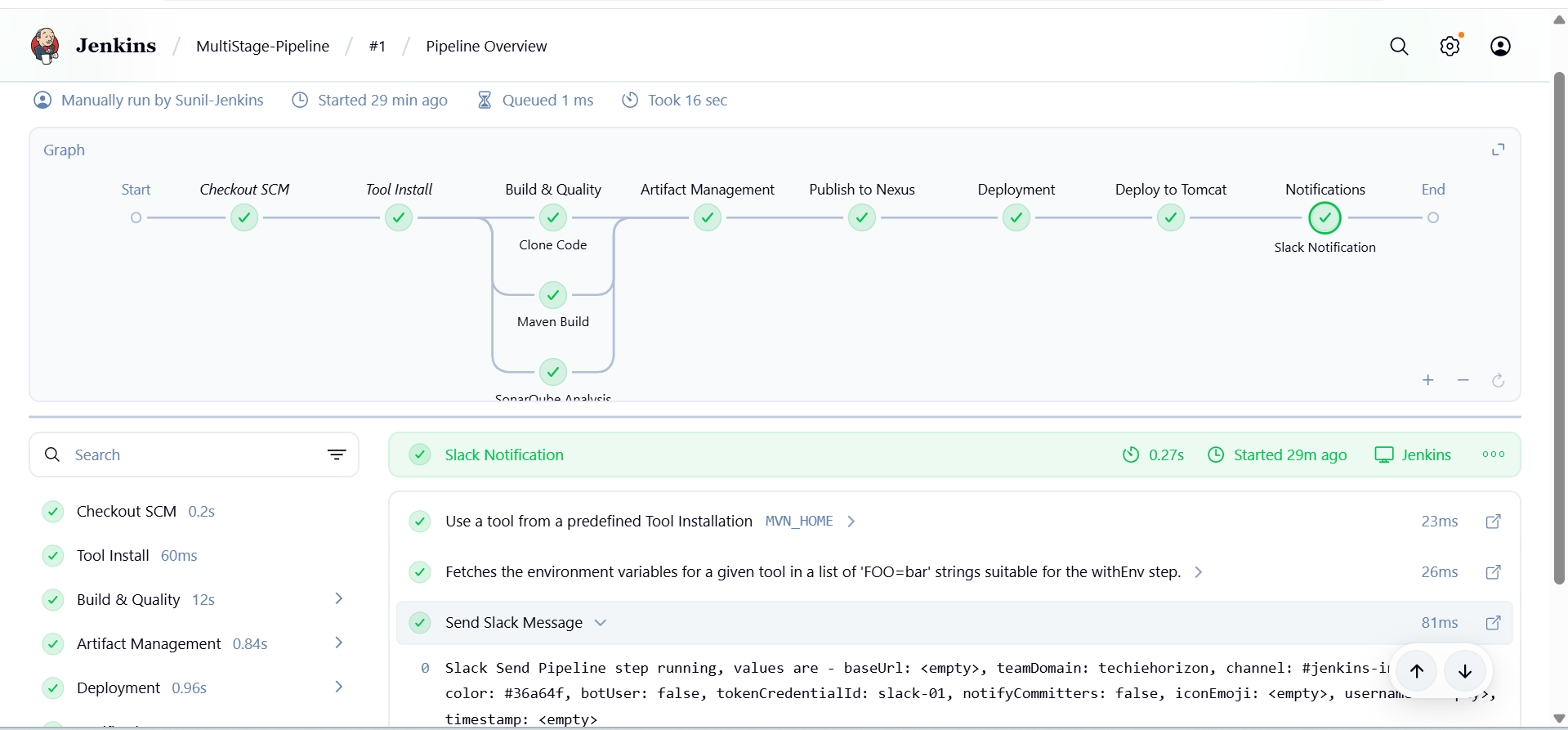


### 5. Save & Build

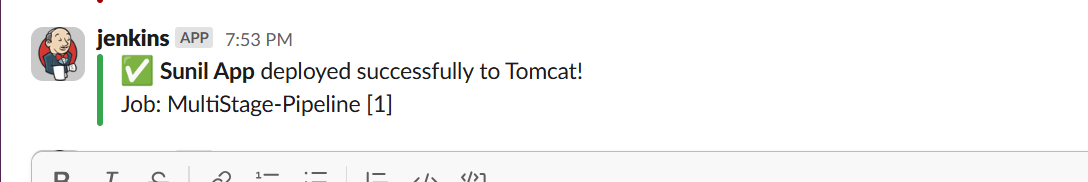
* Click **Save**.
* Run **Build Now**.
* You’ll see multiple **stages** in Jenkins UI (Git Clone → Build → Code Analysis → Test → Package → Deploy).



Multistage-PipeLine



**Slack:**



## ****Multi-Stage Pipeline****

* **Definition:** A pipeline divided into **multiple stages**, where each stage represents a step like Build, Test, Deploy.
* **Focus:** Structure and visualization of the workflow.
* **Can be implemented in both Declarative and Scripted syntax.**
* **Key Point:** “Multi-stage” refers to **the organization of the workflow**, not the syntax itself.

Example: Build → Test → Deploy is a **multi-stage pipeline**.

| * Feature | * Multi-Stage Pipeline | * Declarative Pipeline | * Scripted Pipeline |
| --- | --- | --- | --- |
| * Definition | * Pipeline with multiple stages | * Structured, readable Jenkins syntax | * Flexible, code-oriented syntax |
| * Purpose | * Organize workflow visually | * Easier to maintain and read | * Handle complex logic dynamically |
| * Syntax | * Can be declarative or scripted | * pipeline { stages { ... } } | * node { stage('...') { ... } } |
| * Complex Logic | * Usually simple sequential steps | * Limited, but sufficient for most CI/CD | * Very flexible, supports loops/conditions |
| * Recommended For | * Any pipeline needing stages | * Teams/new users | * Advanced users/complex pipelines |

**Summary:**

* **Multi-stage pipeline** = concept of dividing your workflow into stages (Build, Test, Deploy).
* **Declarative vs Scripted** = **syntax choices** to implement a pipeline.
* Declarative is **simpler and structured**, scripted is **more flexible and code-heavy**.
* Multi-stage pipelines are **usually written in declarative syntax** because it’s clearer, but can also be scripted.

1. **Create one parallel stage pipeline job**

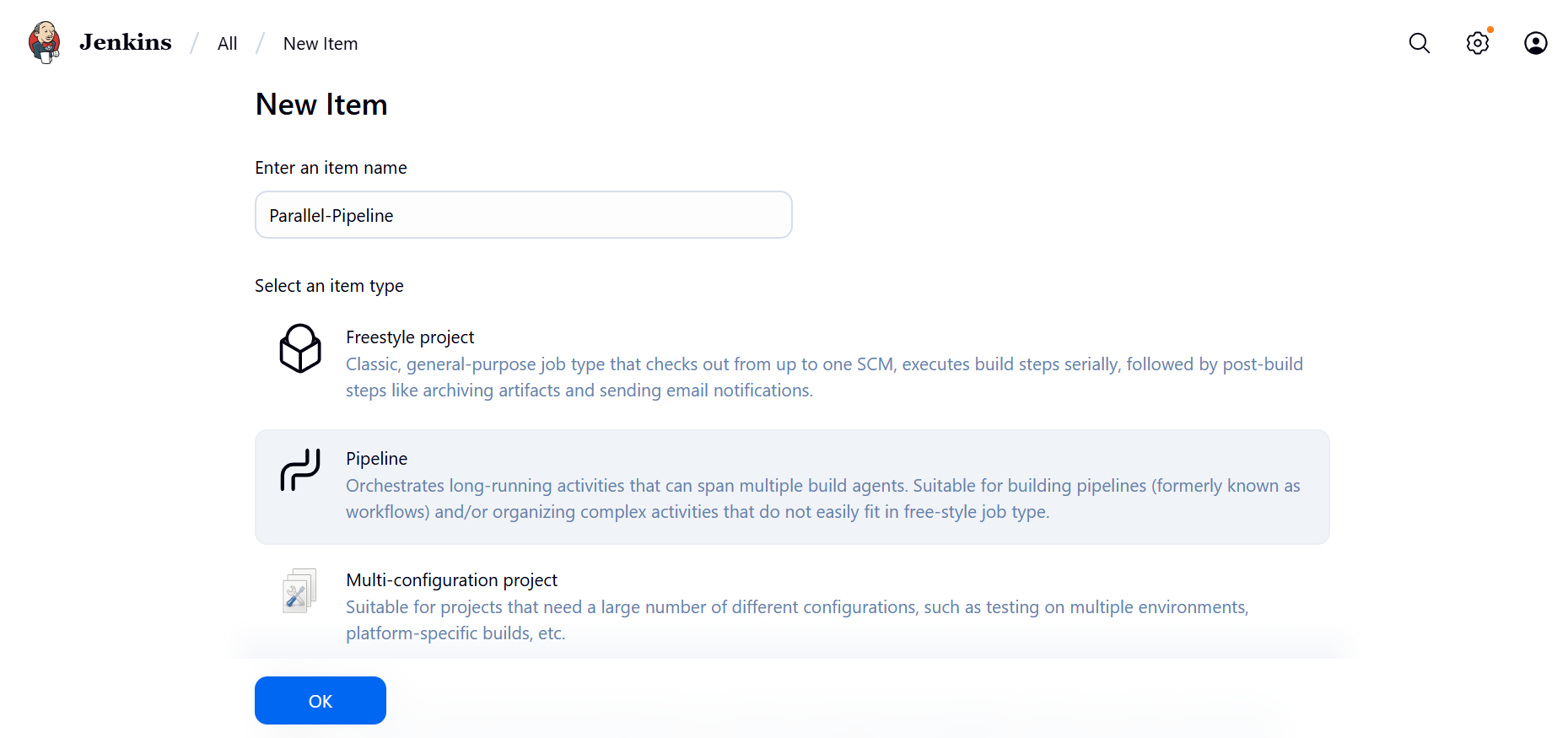
Steps to Create a Parallel Pipeline Job

**Login to Jenkins**

* Open Jenkins (http://<jenkins-server>:8080) and log in.

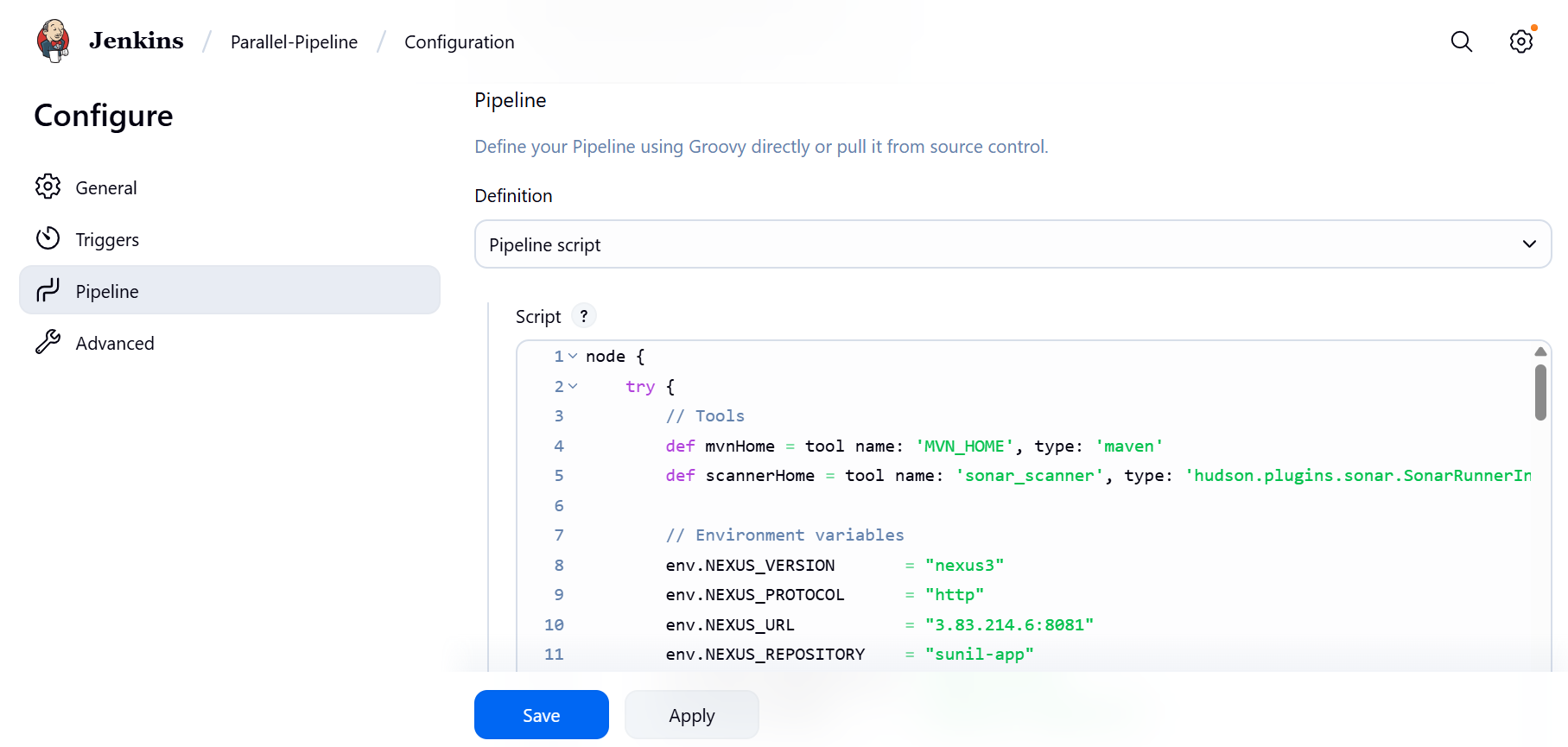
**Create a New Pipeline Job**

* Click **New Item**.
* Enter a name (e.g., Parallel-Pipeline).
* Select **Pipeline** and click **OK**.

****

**Configure the Pipeline**

* Scroll to the **Pipeline** section.
* Choose one of these:
  + **Pipeline script** → Paste your pipeline code directly.
  + **Pipeline script from SCM** → Fetch pipeline code (Jenkinsfile) from Git.

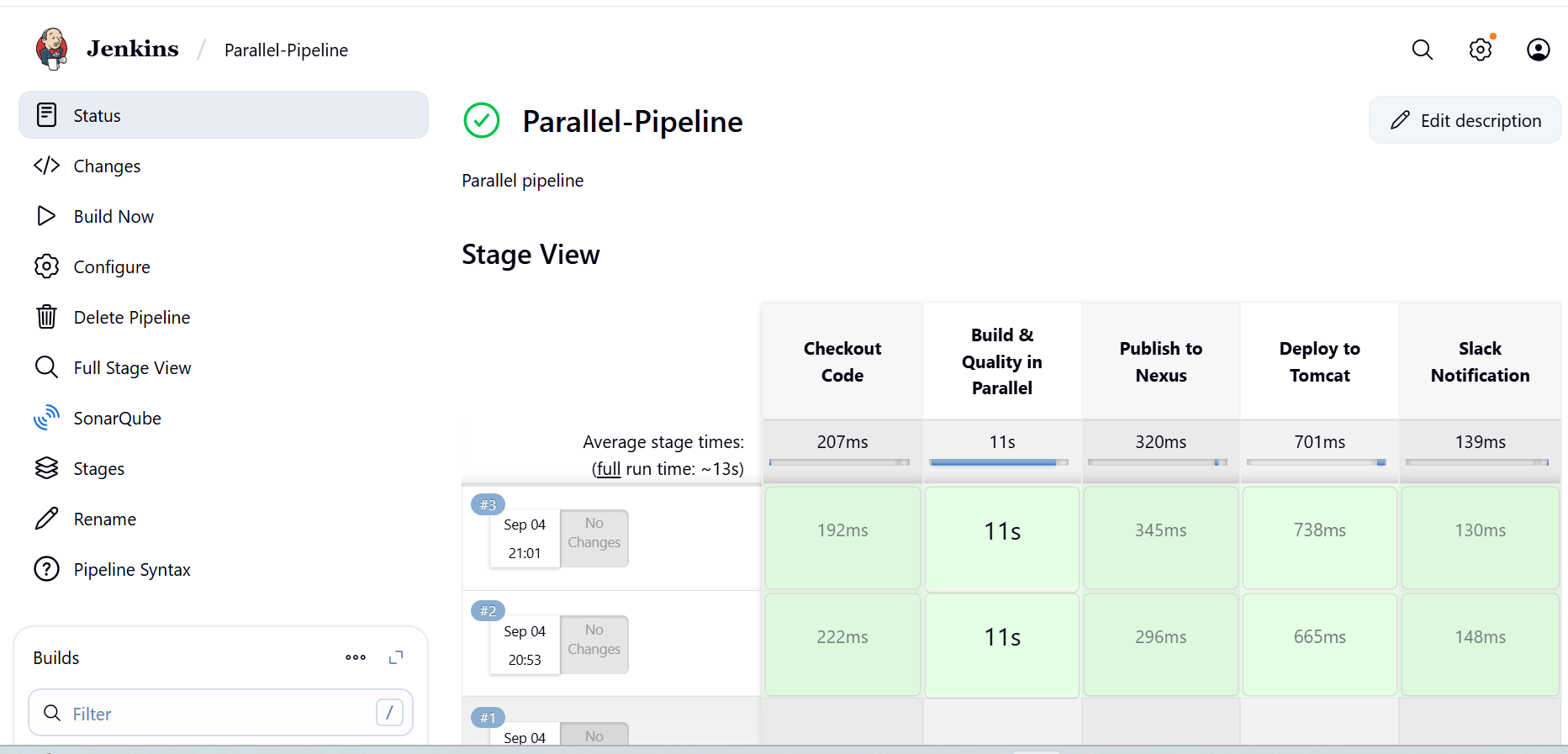


**Define Parallel Stages**

* Inside your pipeline, add a parallel {} block under a stage.
* Each branch inside parallel {} runs **at the same time**.

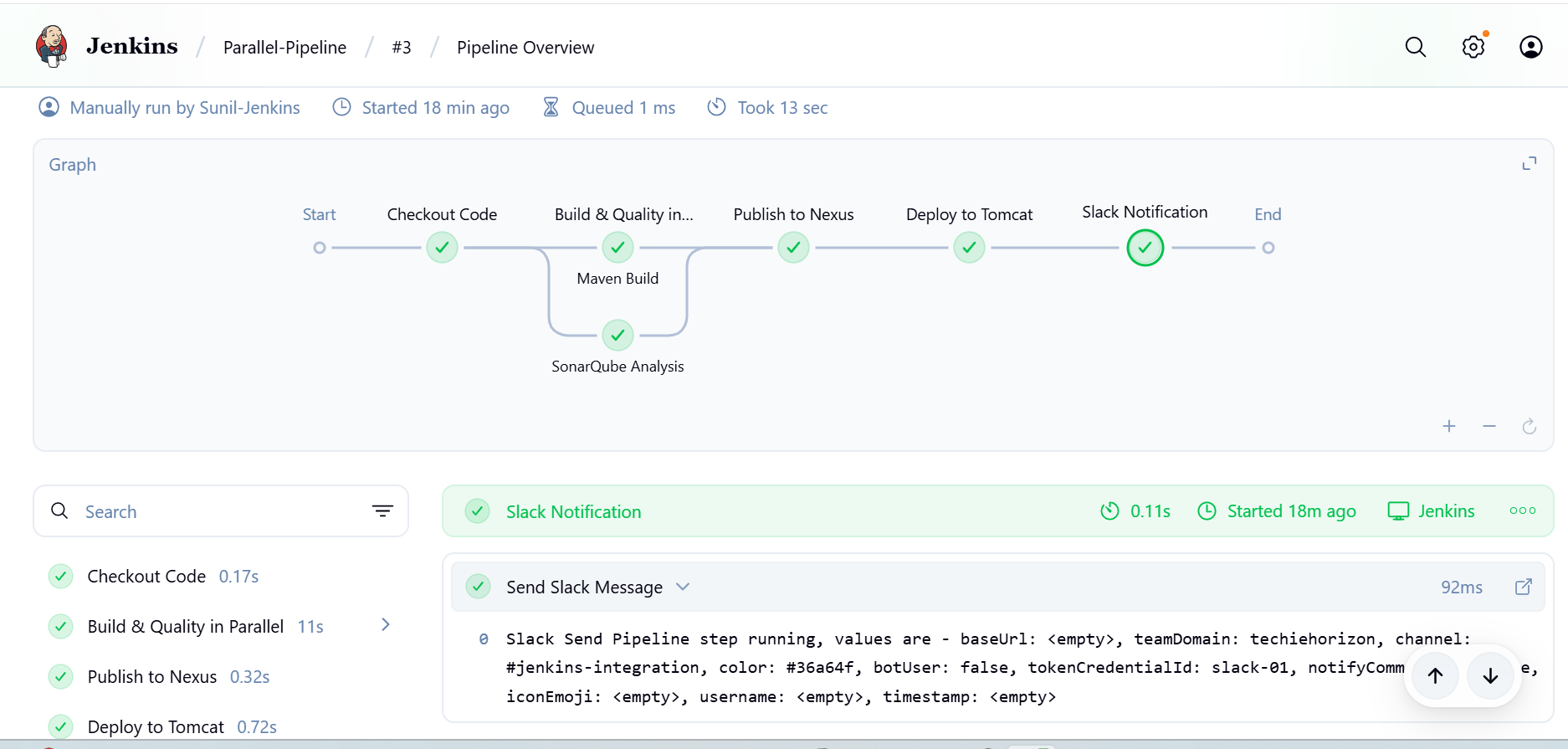
**Save and Build**

* Click **Save**.
* Click **Build Now**.

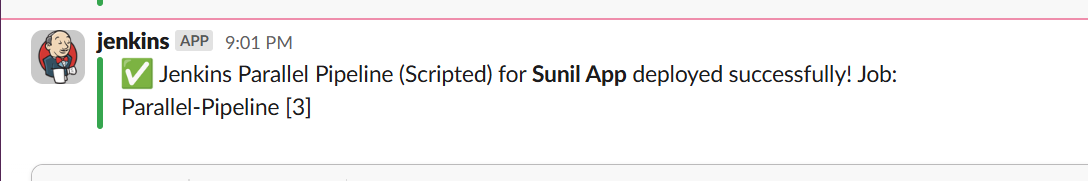


**View Results**

* Open the build.
* In Jenkins UI

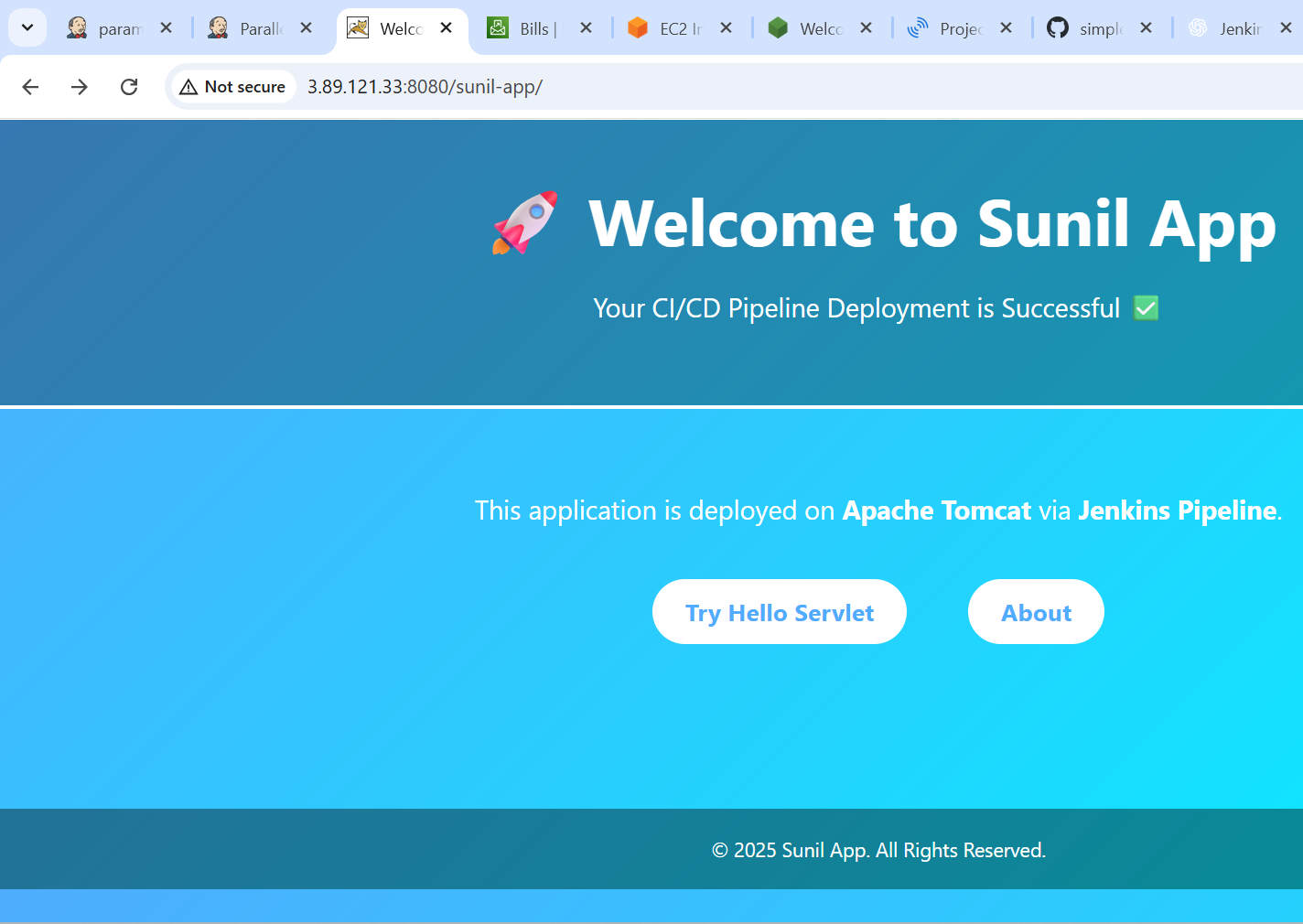


**Slack:**

****

**Tomacat Deploy**

****

****