

Jenkins Build Pipeline: Graphical View and Chained Job Execution

1. Upstream and Downstream Jobs (Job Triggering Logic)

Upstream Job: A job that starts the build chain

Downstream Job: A job that is triggered automatically after the upstream job finishes

Example:

If **Job-A** is configured as the upstream and **Job-B** is set to trigger after **Job-A**, then when **Job-A** builds successfully, **Job-B** starts automatically.

2. Visualizing Jobs with Graphical Representation

To see the pipeline visually (graphical format), use the **Build Pipeline View**.

Steps to Create a Build Pipeline View:

1. In Jenkins dashboard, click the “+” icon (to create a new view)
2. Select **Build Pipeline View**
3. Give your view a name (e.g., **MyPipelineView**)

- + New Item
- 📁 Build History
- ⚙️ Manage Jenkins
- 📁 My Views

Build Queue ▾
No builds in the queue.

Build Executor Status ▴
(0 of 2 executors busy)

New view

Name

MyPipelineView

Type

- ☒ **Build Pipeline View**
Shows the jobs in a build pipeline view. The complete pipeline of jobs that a version propagates through are shown as a row in the view.
- ☐ **List View**
Shows items in a simple list format. You can choose which jobs are to be displayed in which view.
- ☐ **My View**
This view automatically displays all the jobs that the current user has an access to.

Create

← → ↻ Not Secure 18.218.230.170:8080/view/MyPipelineView/ ☆ | 🔍 🔔 1 🛡️ 2 👤 Abdul Rehman Shaik ▾ 🚪 log out

Jenkins

Dashboard > MyPipelineView >

Build Pipeline

▶ 📁 ⚙️ + 🗑️ 🔧
Run History Configure Add Step Delete Manage

Pipeline
#1

→ #1 JobA
Apr 28, 2025 2:40:01 AM
45 sec

→ #1 JobB
Apr 28, 2025 2:40:56 AM
50 ms

→ #1 JobC
Apr 28, 2025 2:41:06 AM
38 ms

← → ↻ Not Secure 18.218.230.170:8080/view/MyPipelineView/ ☆

Jenkins 🔍 🔔 1 🔴 2 👤 Abdul Rehman Shaik ▾ 🚪 log out

Dashboard > MyPipelineView >

Build Pipeline

▶ 📄 ⚙️ + 🗑️ 🔧
Run History Configure Add Step Delete Manage

Pipeline
#3

#3 JobA
📅 Apr 28, 2025 2:56:41 AM
🕒 7.2 sec and counting
👤 admin

→

JobB
📅 N/A
👤 N/A

→

JobC
📅 N/A
👤 N/A

REST API Jenkins 2.492.3

← → ↻ Not Secure 18.218.230.170:8080/view/MyPipelineView/ ☆

Jenkins 🔍 🔔 1 🔴 2 👤 Abdul Rehman Shaik ▾ 🚪 log out

Dashboard > MyPipelineView >

Build Pipeline

▶ 📄 ⚙️ + 🗑️ 🔧
Run History Configure Add Step Delete Manage

Pipeline
#3

#3 JobA
📅 Apr 28, 2025 2:56:41 AM
🕒 12 sec
👤 admin

→

#3 JobB
📅 Apr 28, 2025 2:57:01 AM
🕒 19 ms

→

#3 JobC
📅 Apr 28, 2025 2:57:11 AM
🕒 25 ms

REST API Jenkins 2.492.3

4. Click **OK**
5. In configuration:
 - Set **Initial Job** (starting point, e.g., `myfirstjob`)
 - Enable **Refresh Frequency** (e.g., 5 seconds to auto-refresh view)
6. Save the view

Note: Only *List View* and *My View* are default Jenkins views; Build Pipeline View needs a plugin.

3. Installing Required Plugin (Build Pipeline Plugin)

Steps:

1. Go to: **Manage Jenkins** → **Manage Plugins**
2. Click the **Available** tab
3. Search for **Build Pipeline Plugin**
4. Select it and choose:
 - **Install without restart**
OR
 - **Download now and install after restart** (if restart is required)
5. After installation, restart Jenkins if necessary

Use “**Check now**” to refresh plugin list
Always ensure no jobs are running before restart

4. Chaining Jobs (Configuring Downstream Jobs)

Scenario:

Trigger `mysecondjob` automatically when `myfirstjob` completes successfully.

Steps:

1. Go to **Configure** of the downstream job (`mysecondjob`)
2. Scroll to **Build Triggers**
3. Enable “**Build after other projects are built**”
4. Enter `myfirstjob` in the text box
5. Click **Apply** → **Save**

Now Jenkins understands the job dependency

5. Add Script to First Job (Optional)

To add a shell script to the first job:

- Go to: `myfirstjob` → **Configure** → **Build** section
- Add a **Build step**: Select **Execute shell**
- Example: `echo "This is first build"`

After saving, triggering `myfirstjob` will execute the shell script and then automatically trigger the downstream `mysecondjob`.

6. Colors in Pipeline View (Build Status Indicators)

In the **Build Pipeline View**, you'll notice:

- **Green**: Successful build
- **Red**: Build failed
- **Blue (or yellow depending on theme)**: Unstable build

These visual cues help track the status of each job in the pipeline.

7. Bonus: Triggering Build Through Scripting

You can trigger Jenkins jobs via command-line or script using:

a. Curl with Jenkins API: `curl -X POST http://<jenkins-url>/job/<job-name>/build \`

`--user username:api_token`

b. Jenkins CLI Tool: `java -jar jenkins-cli.jar -s http://<jenkins-url> build <job-name>`

You must generate an API token from your Jenkins account for authentication.

Real-World Example:

Use Case:

You have a 3-stage build pipeline:

- **Code-Checkout** (Git clone)
- **Build-and-Test** (compile + run unit tests)
- **Deploy-to-Dev**

This can be modeled in Jenkins as:

- Set **Code-Checkout** as the initial job in Build Pipeline View
- Set **Build-and-Test** → build after **Code-Checkout**
- Set **Deploy-to-Dev** → build after **Build-and-Test**

Each job is triggered automatically after the previous one completes successfully.

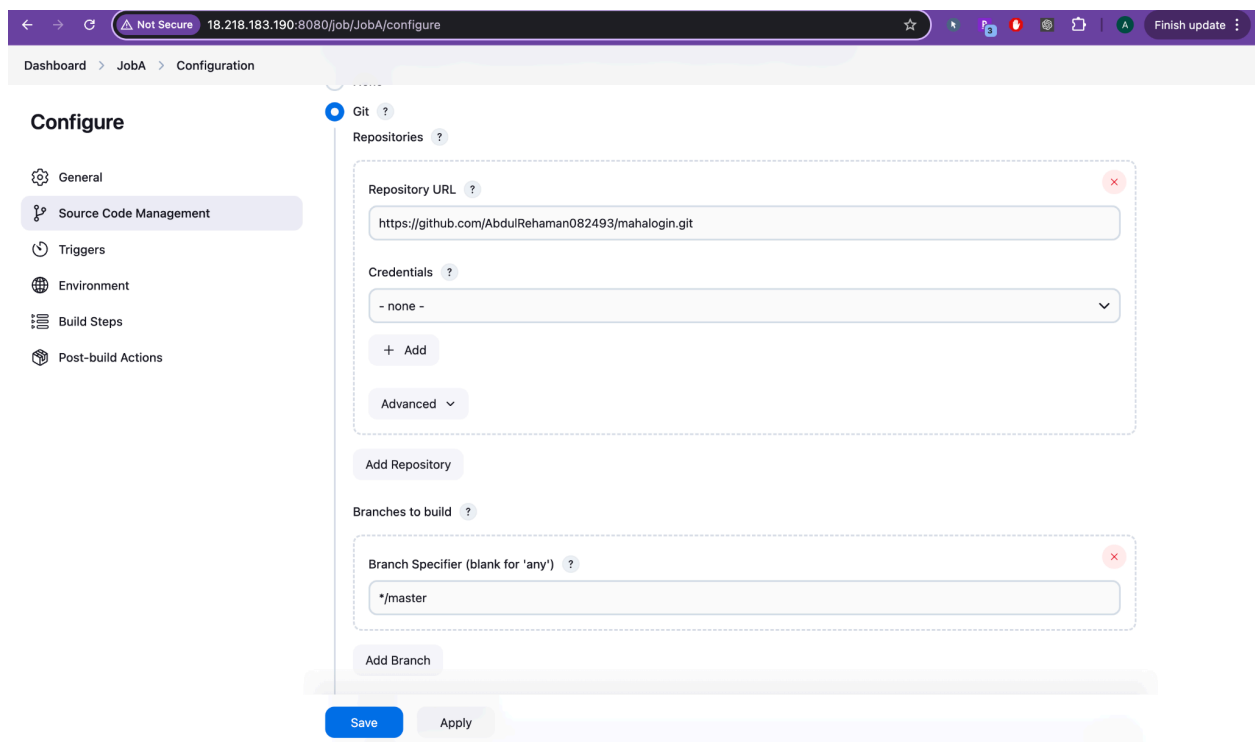
STEP-BY-STEP GUIDE (GITHUB → JENKINS → JOBA → JOBB → JOBC)

STEP 1: Correct Webhook URL

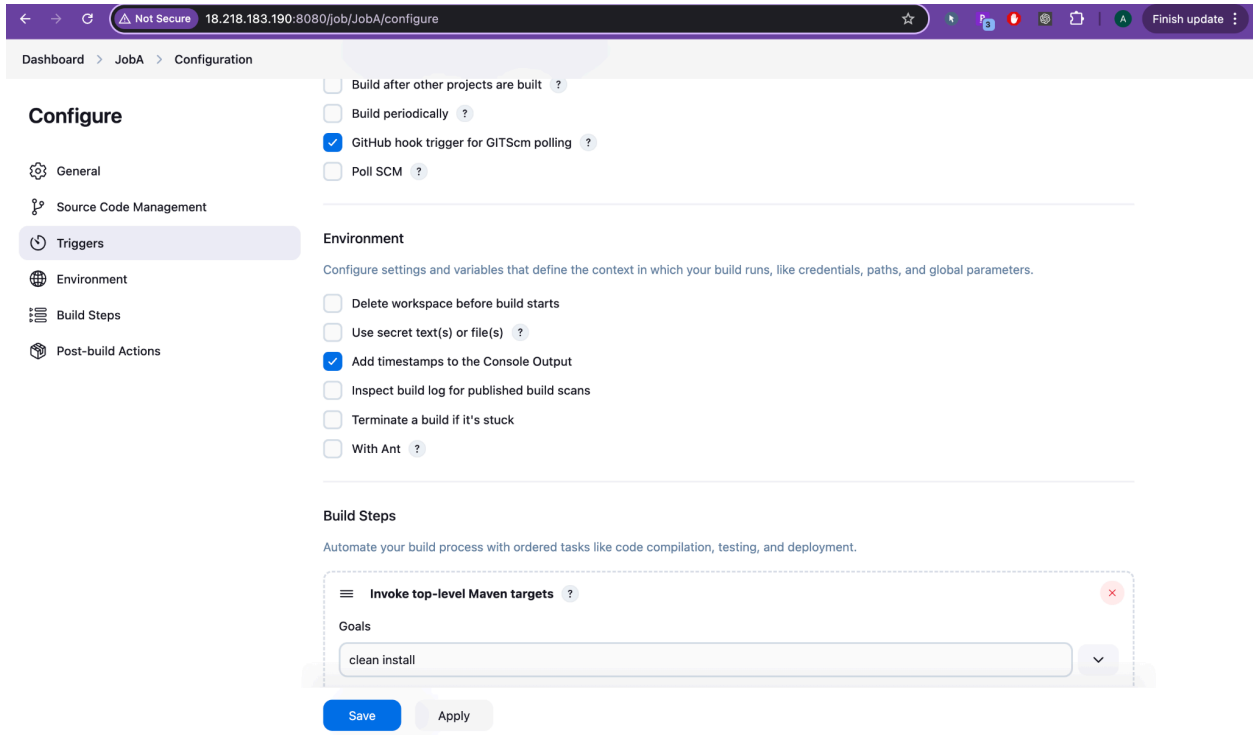
✓ Use this URL: `http://18.218.183.190:8080/github-webhook/`

STEP 2: Fix Jenkins JobA Settings

Inside **JobA**:



The screenshot shows the Jenkins JobA Configuration page. The browser address bar indicates the URL `18.218.183.190:8080/job/JobA/configure`. The left sidebar contains the 'Configure' section with options: General, Source Code Management (selected), Triggers, Environment, Build Steps, and Post-build Actions. The main content area is titled 'Git' and shows the 'Repositories' section. The 'Repository URL' field contains `https://github.com/AbdulRehaman082493/mahalogin.git`. The 'Credentials' dropdown is set to '- none -'. Below the repository settings, there is an 'Add Repository' button. The 'Branches to build' section shows the 'Branch Specifier (blank for 'any')' field set to `*/master`. At the bottom, there are 'Save' and 'Apply' buttons.



2.1 Source Code Management

- Git → Repository URL →
<https://github.com/AbdulRehaman082493/mahalogin.git>
- Credentials:
 - **If your repo is Private**, create credentials.
 - Go to Jenkins → Credentials → Add:
 - Kind: **Username and Password** (or Personal Access Token).
 - Username: GitHub username
 - Password: GitHub token

- Then select it here.
- If **Public Repo**, keep Credentials: - **none** -.
- Branches to build:
 - ✓ ***/master** (Correct if you are using master branch.)

2.2 Triggers

✓ Tick the checkbox:

- **GitHub hook trigger for GITScm polling** ✓
- **Poll SCM** (✗ Don't check. Not needed if webhook is coming.)

2.3 Build Steps

✓ Build step:

- Invoke top-level Maven targets.
- Goals:
`clean install`

STEP 3: Fix Jenkins JobB Settings

Inside **JobB**:

← → ↻ ⚠ Not Secure 18.218.183.190:8080/job/JobB/configure ☆ 🔍 🌐 📄 📁 📂 📅 📆 📇 📈 📉 📊 📋 📌 📍 📎 📏 📐 📑 📒 📓 📔 📕 📖 📗 📙 📚 📛 📜 📝 📞 📟 📠 📡 📢 📣 📤 📥 📦 📧 📨 📩 📪 📫 📬 📭 📮 📯 📰 📱 📲 📳 📴 📵 📶 📷 📸 📹 📺 📻 📼 📽 📾 📿 📠 📡 📢 📣 📤 📥 📦 📧 📨 📩 📪 📫 📬 📭 📮 📯 📰 📱 📲 📳 📴 📵 📶 📷 📸 📹 📺 📻 📼 📽 📾 📿

Dashboard > JobB > Configuration

Configure

⚙️ General

🔑 Source Code Management

🕒 Triggers

🌐 Environment

📋 Build Steps

📦 Post-build Actions

Source Code Management

Connect and manage your code repository to automatically pull the latest code for your builds.

☒ None

☐ Git ?

Triggers

Set up automated actions that start your build based on specific events, like code changes or scheduled times.

☐ Trigger builds remotely (e.g., from scripts) ?

☒ Build after other projects are built ?

Projects to watch

JobA

☒ Trigger only if build is stable

☐ Trigger even if the build is unstable

☐ Trigger even if the build fails

☐ Always trigger, even if the build is aborted

☐ Build periodically ?

☐ GitHub hook trigger for GITScm polling ?

☐ Poll SCM ?

Save

Apply

← → ↻ ⚠ Not Secure 18.218.183.190:8080/job/JobB/configure ☆ 🔍 🌐 📄 📁 📂 📅 📆 📇 📈 📉 📊 📋 📌 📍 📎 📏 📐 📑 📒 📓 📔 📕 📖 📗 📙 📚 📛 📜 📝 📞 📟 📠 📡 📢 📣 📤 📥 📦 📧 📨 📩 📪 📫 📬 📭 📮 📯 📰 📱 📲 📳 📴 📵 📶 📷 📸 📹 📺 📻 📼 📽 📾 📿

Dashboard > JobB > Configuration

Configure

⚙️ General

🔑 Source Code Management

🕒 Triggers

🌐 Environment

📋 Build Steps

📦 Post-build Actions

Environment

Configure settings and variables that define the context in which your build runs, like credentials, paths, and global parameters.

☐ Delete workspace before build starts

☐ Use secret text(s) or file(s) ?

☒ Add timestamps to the Console Output

☐ Inspect build log for published build scans

☐ Terminate a build if it's stuck

☐ With Ant ?

Build Steps

Automate your build process with ordered tasks like code compilation, testing, and deployment.

≡ Execute shell ?

Command

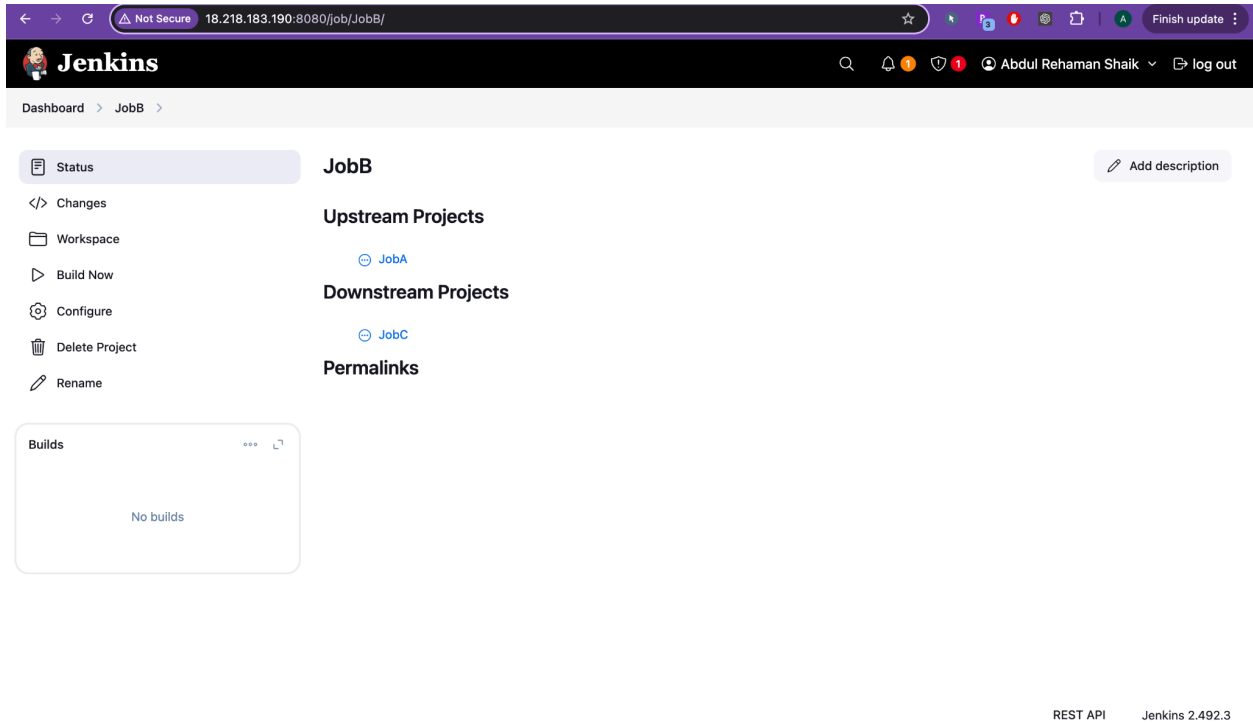
See [the list of available environment variables](#)

echo "This is second build"

Advanced ▾

Save

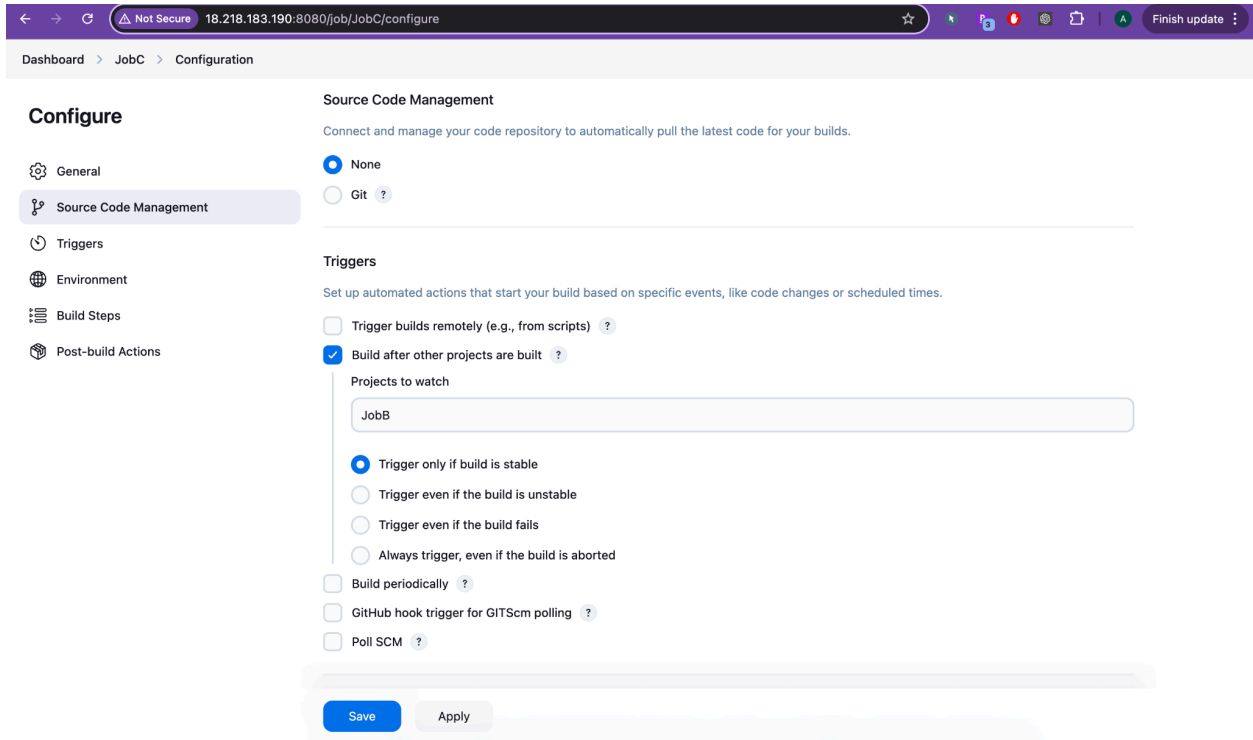
Apply



- Source Code Management → None (because no git clone needed here).
- Triggers:
 - Tick **Build after other projects are built** ✓
 - Projects to watch:
`JobA`
 - "Trigger only if build is stable" (default and good).
- Build Steps:
 - Execute Shell →
`echo "This is second build"`

STEP 4: Fix Jenkins JobC Settings

Inside **JobC**:



The screenshot shows the Jenkins JobC configuration interface. The browser address bar indicates the URL is 18.218.183.190:8080/job/JobC/configure. The left sidebar contains a 'Configure' section with a list of tabs: General, Source Code Management (selected), Triggers, Environment, Build Steps, and Post-build Actions. The main content area is titled 'Source Code Management' and includes a description: 'Connect and manage your code repository to automatically pull the latest code for your builds.' Below this, there are two radio buttons: 'None' (selected) and 'Git'. The 'Triggers' section follows, with a description: 'Set up automated actions that start your build based on specific events, like code changes or scheduled times.' It contains several checkboxes: 'Trigger builds remotely (e.g., from scripts)' (unchecked), 'Build after other projects are built' (checked), 'Build periodically' (unchecked), 'GitHub hook trigger for GITScm polling' (unchecked), and 'Poll SCM' (unchecked). The 'Build after other projects are built' checkbox is expanded, showing a list of 'Projects to watch' with 'JobB' entered. Below this, there are four radio buttons for triggering conditions: 'Trigger only if build is stable' (selected), 'Trigger even if the build is unstable', 'Trigger even if the build fails', and 'Always trigger, even if the build is aborted'. At the bottom of the configuration area, there are two buttons: 'Save' and 'Apply'.

Dashboard > JobC > Configuration

Configure

- General
- Source Code Management**
- Triggers
- Environment
- Build Steps
- Post-build Actions

Source Code Management

Connect and manage your code repository to automatically pull the latest code for your builds.

☒ None
☐ Git ?

Triggers

Set up automated actions that start your build based on specific events, like code changes or scheduled times.

☐ Trigger builds remotely (e.g., from scripts) ?
☒ Build after other projects are built ?

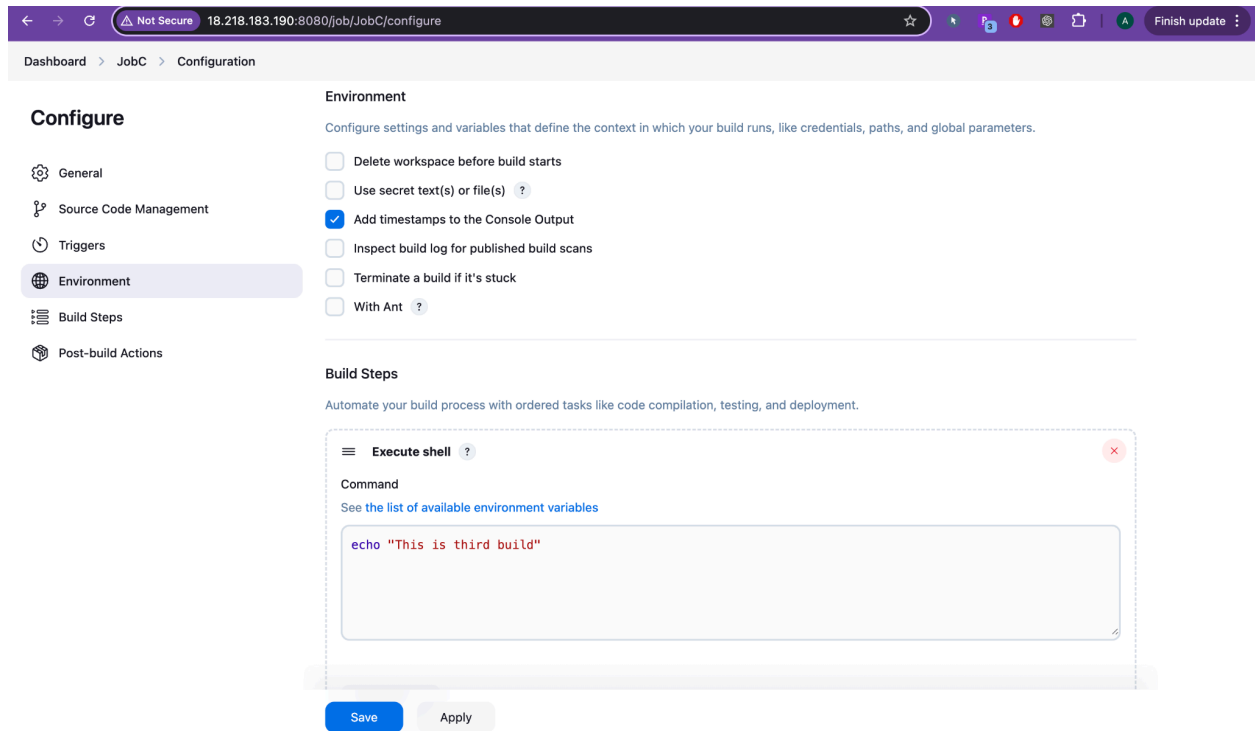
Projects to watch

JobB

☒ Trigger only if build is stable
☐ Trigger even if the build is unstable
☐ Trigger even if the build fails
☐ Always trigger, even if the build is aborted

☐ Build periodically ?
☐ GitHub hook trigger for GITScm polling ?
☐ Poll SCM ?


Save Apply



- Source Code Management → None.
- Triggers:
 - Tick **Build after other projects are built** ✓
 - Projects to watch:
JobB
- Build Steps:
 - Execute Shell →
echo "This is third build"

STEP 5: Configure GitHub Webhook

In GitHub:

- Open your repo: `mahalogin`
- Go to: Settings → Webhooks → Add Webhook
- Fill:
 - **Payload URL:**
`http://18.218.183.190:8080/github-webhook/`
 - **Content type:**
`application/json`
 - **Event:**
"Just the push event"
- Click **Add webhook** 

STEP 6: Test the Webhook

- Go to GitHub Repo → Make any small change (example: update README.md) and push.
- GitHub should send payload (green tick if successful).


 In Jenkins:

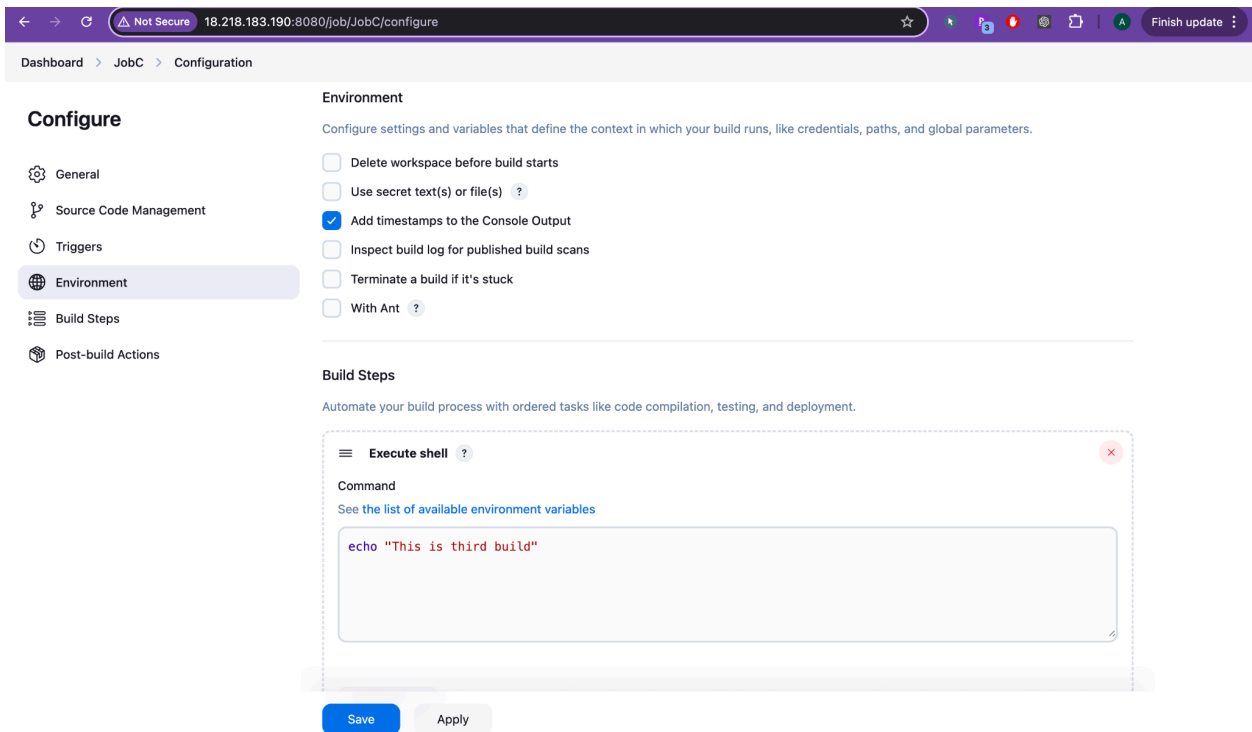
- JobA will start automatically.
- JobB will start after JobA success.
- JobC will start after JobB success.

 FINAL CHAIN

Git Push → GitHub Webhook → Jenkins JobA → Jenkins JobB → Jenkins JobC

IMPORTANT NOTES:

Topic	Details
Plugin	Make sure GitHub plugin is installed in Jenkins
Jenkins IP	IP must be Public and Port 8080 open (Security Group if AWS)
Credentials	Only if repo is Private
Webhook testing	Webhook → "Recent Deliveries" → Green tick  means success



The screenshot shows the Jenkins web interface at the URL `18.218.183.190:8080/job/JobC/configure`. The left sidebar contains a "Configure" section with a list of tabs: General, Source Code Management, Triggers, Environment (selected), Build Steps, and Post-build Actions. The main content area is titled "Environment" and includes a description: "Configure settings and variables that define the context in which your build runs, like credentials, paths, and global parameters." Below this, there are several checkboxes: "Delete workspace before build starts" (unchecked), "Use secret text(s) or file(s)" (unchecked with a help icon), "Add timestamps to the Console Output" (checked), "Inspect build log for published build scans" (unchecked), "Terminate a build if it's stuck" (unchecked), and "With Ant" (unchecked with a help icon). Below the Environment section is the "Build Steps" section, which says "Automate your build process with ordered tasks like code compilation, testing, and deployment." It contains a single step named "Execute shell" with a help icon. The "Command" field for this step contains the text `echo "This is third build"`. At the bottom of the configuration page are two buttons: "Save" and "Apply".

<http://18.218.230.170/>