

What is "Build Periodically"?

The **"Build Periodically"** option in Jenkins uses **cron syntax** to schedule jobs at specific times—**regardless of code changes**. It's often used for:

- Nightly builds
- Frequent CI runs (e.g., every 15 minutes)
- Regular test deployments

17 Cron Syntax in Jenkin

Format:

MIN HOUR DOM MONTH DOW

Field	Description	Possible Values
MIN	Minute	0–59
HOUR	Hour	0–23
DOM	Day of Month	1–31
MONTH	Month	1–12 or JAN–DEC
DOW	Day of Week	0–7 (0 and 7 = Sunday)

You can also use special characters:

- *: every value
- H: hash-based (spreads load)
- /: increments (e.g., H/15)
- ,: separate values (e.g., 1, 15)
- -: ranges (e.g., 1-5)
- H/2 * * * *: every 2 minutes (spreads load)

✓ Examples with Real-World Use Cases

Cron Pattern	Description	Use Case Example
H/2 * * * *	Every 2 minutes	Quick feedback loop for fast-moving projects
0 * * * *	Every hour	Hourly data sync or job summary builds
H 3 * * *	Every day at 3 AM	Nightly regression test suite
15 2 * * 1-5	2:15 AM on weekdays	Weekday-only early morning build jobs
H 22 * * 5	Every Friday at 10 PM	Weekly backup or environment refresh
0 0 1 * *	Midnight on the 1st of every month	Monthly reports or billing process
*/15 9-17 * * 1-5	Every 15 min between 9 AM–5 PM on weekdays	Frequent CI builds during work hours

Jenkins Special: Use of H

The **H** spreads jobs across time to **avoid load spikes** if multiple jobs use the same schedule.

- Example: If you set **H 4 * * ***, Jenkins will pick a random but stable minute for each job at hour 4 to distribute load.

Your Configuration:

From your screenshot:

H/2 * * * *

- **Meaning:** Jenkins will run the job **every 2 minutes**.
- **Real-World Example:** Perfect for a continuously running integration pipeline that watches for issues quickly, such as a **staging environment build or test**.

Jenkins Job: Assignment03 – Step-by-Step Detailed Explanation

Objective: Automatically build your Maven project every 2 minutes using Jenkins' **Build Periodically** feature (without requiring code changes).

.

STEP 1: Job Setup

- Job Name: **Assignment03**
- Type: **Freestyle Project**
- Purpose: Clone the repo, compile, and package it via Maven every 2 minutes.

Dashboard > Assignment03 > 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** ?

Repositories ?

Repository URL ?

Credentials ?

+ Add

Advanced ▾

Add Repository

Branches to build ?

Branch Specifier (blank for 'any') ?

Add Branch

Repository browser ?

Save Apply

STEP 2: Source Code Management (Git)

Location: [Configure](#) > [Source Code Management](#)

Field	Value
Repository URL	https://github.com/AbdulRehaman082493/mahalogin.git
Credentials	- none - (public repo)
Branch	*/master (default main branch)

✓ This allows Jenkins to pull code from the [master](#) branch of your GitHub repository.

The screenshot shows the Jenkins web interface for configuring a job named 'Assignment03'. The left sidebar contains a 'Configure' menu with options: General, Source Code Management, Triggers (selected), Environment, Build Steps, and Post-build Actions. The main content area is titled 'Configure' and has a sub-header 'Triggers'. Below this, it says 'Set up automated actions that start your build based on specific events, like code changes or scheduled times.' There are three checkboxes: 'Trigger builds remotely (e.g., from scripts)', 'Build after other projects are built', and 'Build periodically' (which is checked). Below the 'Build periodically' checkbox is a 'Schedule' field containing the text 'H/2 * * * *'. A tooltip below the schedule field states: 'Would last have run at Thursday, April 24, 2025 at 5:42:13 PM Coordinated Universal Time; would next run at Thursday, April 24, 2025 at 5:44:13 PM Coordinated Universal Time.' Below the schedule field are two more unchecked checkboxes: 'GitHub hook trigger for GITScm polling' and 'Poll SCM'. Below the Triggers section is an 'Environment' section with the text 'Configure settings and variables that define the context in which your build runs, like credentials, paths, and global parameters.' It contains several unchecked checkboxes: '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', and 'With Ant'. At the bottom of the configuration page are two buttons: 'Save' and 'Apply'.

🕒 STEP 3: Build Trigger – Build periodically

Location: [Configure](#) > [Triggers](#)

You checked ☒ **Build periodically** and used this schedule:

H/2 * * * *

Field	Meaning
H/2	Randomized minute offset; builds every 2 minutes
* (next four fields)	Every hour, day, month, and day of the week

🔄 **Effect:** Jenkins triggers this job **every 2 minutes** continuously — **even if no code changes** have happened.

☒ Useful for:

- Continuous testing
- Monitoring
- Scheduled builds

📌 **Not dependent** on Git commits or polling — it's purely based on time.

⚙️ STEP 4: Build Step – Maven

Location: [Configure](#) > [Build Steps](#)

Field	Value
Build Step	Invoke top-level Maven targets
Goals	<code>clean install</code>

The screenshot shows the Jenkins web interface for configuring a build job named 'Assignment03'. The 'Configuration' page is active, with the 'Build Steps' tab selected in the left sidebar. The 'Build Steps' section is titled 'Invoke top-level Maven targets' and shows a single step with the goal 'clean install'. The 'Goals' field is a text input with a dropdown arrow. Below the goal field is an 'Advanced' dropdown. At the bottom of the 'Build Steps' section is an 'Add build step' button. The 'Post-build Actions' section is also visible, with an 'Add post-build action' button. The 'Save' button is highlighted in blue.

✓ What it does:

- **clean**: Deletes **target/** directory to ensure a fresh build.
- **install**: Compiles, runs unit tests, packages **.war**, and installs it to the local Maven repository.

Resulting artifact:

/var/lib/jenkins/workspace/Assignment03/target/mahaLogin-2.0.war

STEP 5: Environment and Post-Build (Optional)

None configured in your setup. You may optionally:

- Delete workspace before build
- Add post-build steps like notifications or deployment

STEP 6: Jenkins Build Output

Location: **Assignment03** > **Status**

- You see a list of builds triggered every 2 minutes (✓ Build #1 through #7).
- Each build was successful (green check).
- Builds are automatically triggered without manual clicks.

✓ Final Outcome

The screenshot displays the Jenkins web interface for a job named 'Assigment03'. The browser address bar shows the URL '3.144.17.135:8080/job/Assigment03/'. The Jenkins logo and navigation menu are visible at the top. The left sidebar contains links for Status, Changes, Workspace, Build Now, Configure, Delete Project, and Rename. The main content area shows the job name 'Assigment03' and a 'Permalinks' section. Below this, a 'Builds' section lists recent builds, including build #1 through #7, all of which are successful. The footer of the page shows 'REST API' and 'Jenkins 2.492.3'.

Componet Configuration

Git Repo	GitHub repo with master branch
Trigger	Every 2 minutes (H/2 * * * * *)
Build Step	Maven: <code>clean install</code>
Output	<code>.war</code> file built and installed to local <code>.m2</code>
Status	Multiple successful automated builds

Optional Enhancements

Want to:

- Deploy `.war` file to Tomcat?
- Add webhook for GitHub triggers?

- Archive build artifacts?
- Email build results?

Let me know and I'll guide you step-by-step!