

## Lesson 05 Demo 05

### Integrating Ant with Jenkins

**Objective:** To install the Ant plugin and integrate it with Jenkins for automating build and deployment processes in Java projects

**Tools required:** Git and Jenkins

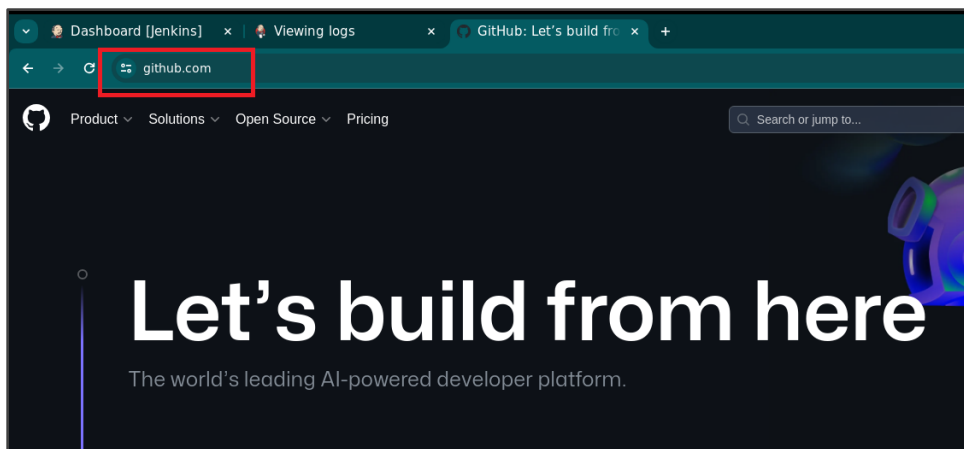
**Prerequisite:** NA

Steps to be followed:

1. Create a GitHub repository
2. Add build.xml file to the repository
3. Set the Global Tool Configuration
4. Integrate Ant with Jenkins

#### Step 1: Create a GitHub repository

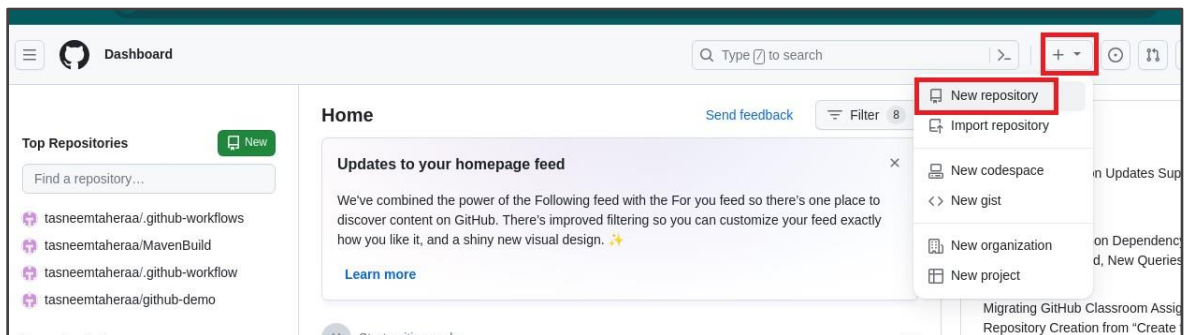
1.1 Open the browser and go to this link <https://github.com/>



## 1.2 Enter the credentials for your GitHub account and click on **Sign in**

The image shows the GitHub sign-in page. At the top is the GitHub logo. Below it is the text "Sign in to GitHub". There are two input fields: "Username or email address" with the value "tasneemtaheraa" and "Password" with masked characters. A green "Sign in" button is below the password field. To the right of the password field is a link "Forgot password?". Below the sign-in fields is a link "Sign in with a passkey" and a link "New to GitHub? Create an account".

## 1.3 Click on + icon and select **New repository** to create a new repository



## 1.4 Give an arbitrary name to the repository

The image shows the "Create a new repository" page on GitHub. The page title is "Create a new repository". Below the title is a description: "A repository contains all project files, including the revision history. Already have a project repository elsewhere? [Import a repository.](#)". There is a note: "Required fields are marked with an asterisk (\*)". The "Owner" field is set to "tasneemtaheraa". The "Repository name" field is highlighted with a red box and contains the text "Demo Ant". Below the name field is a green checkmark and the text: "Your new repository will be created as Demo-Ant. The repository name can only contain ASCII letters, digits, and the characters ., -, and \_." At the bottom, there is a link: "Great repository names are short and memorable. Need inspiration? How about [laughing-spork](#) ?".

### 1.5 Select the check box of **ADD a README file** and click on **Create repository**

Initialize this repository with:

☒ **Add a README file**  
This is where you can write a long description for your project. [Learn more about READMEs.](#)

**Add .gitignore**  
.gitignore template: **None** ▾  
Choose which files not to track from a list of templates. [Learn more about ignoring files.](#)

**Choose a license**  
License: **None** ▾  
A license tells others what they can and can't do with your code. [Learn more about licenses.](#)

This will set **main** as the default branch. Change the default name in your [settings](#).

ⓘ You are creating a public repository in your personal account.

**Create repository**

### 1.6 Click on the highlighted **Code** section

tasneemtaheraa / Demo-Ant

Code Issues Pull requests Actions Projects Wiki Security Insights Settings

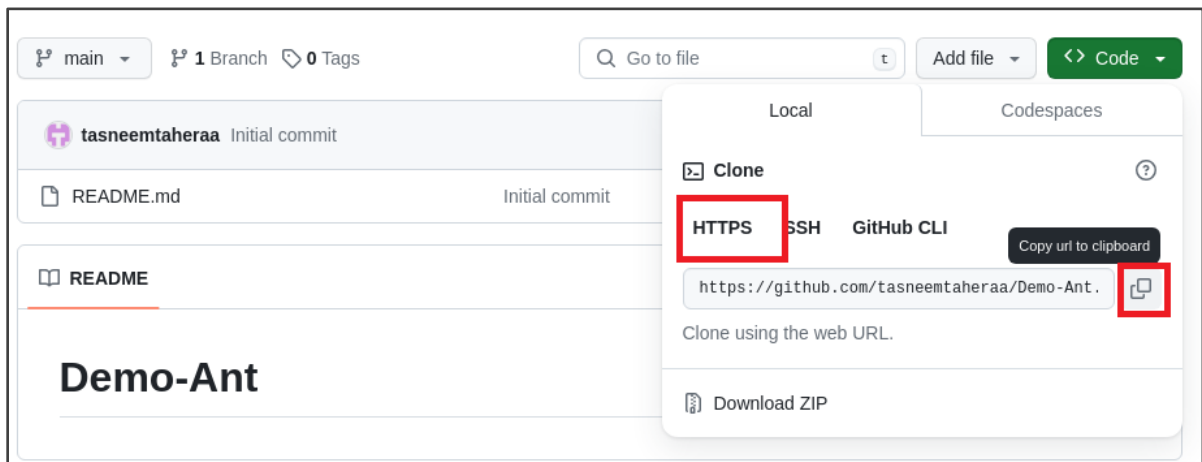
**Demo-Ant** Public Pin Unwatch 1 ▾

main ▾ 1 Branch 0 Tags Go to file t Add file ▾ **<> Code ▾**

tasneemtaheraa Initial commit 567b865 · now 1 Commits

README.md Initial commit now

## 1.7 Click on **HTTPS** and copy the repository URL



## Step 2: Add build.xml file to the repository

### 2.1 Open the terminal and execute the command **mkdir helloworld** to create a directory

```
labsuser@ip-172-31-40-202:~/helloworld$ mkdir helloworld
```

### 2.2 Execute the command **cd helloworld** to navigate to the hello-world directory

```
labsuser@ip-172-31-39-225:~$ mkdir helloworld
labsuser@ip-172-31-39-225:~$ cd helloworld
labsuser@ip-172-31-39-225:~/helloworld$
```

### 2.3 Execute the command **vi build.xml** to open the XML file

```
labsuser@ip-172-31-39-225:~$ mkdir helloworld
labsuser@ip-172-31-39-225:~$ cd helloworld
labsuser@ip-172-31-39-225:~/helloworld$ vi build.xml
labsuser@ip-172-31-39-225:~/helloworld$
```

2.4 Paste the below code into the file, save the file, and exit the editor:

```
<?xml version="1.0"?>
<project name="Hello World Project" default="info">
<target name="info">
<echo> Hello World - Welcome to Simplilearn</echo>
</target>
</project>
```

```
<?xml version="1.0"?>
<project name="Hello World Project" default="info">
<target name="info">
<echo> Hello World - Welcome to Simplilearn</echo>
</target>
</project>
~
~
~
~
~
```

2.5 Run the **git init** command to initialize a new repository

```
labsuser@ip-172-31-39-225:~/helloworld$ git init
hint: Using 'master' as the name for the initial branch. This default branch name
hint: is subject to change. To configure the initial branch name to use in all
hint: of your new repositories, which will suppress this warning, call:
hint:
hint:   git config --global init.defaultBranch <name>
hint:
hint: Names commonly chosen instead of 'master' are 'main', 'trunk' and
hint: 'development'. The just-created branch can be renamed via this command:
hint:
hint:   git branch -m <name>
Initialized empty Git repository in /home/labsuser/helloworld/.git/
labsuser@ip-172-31-39-225:~/helloworld$
```

2.6 Execute the following commands to add the new files:

```
git add .
git commit -m "Add new files"
```

```
labsuser@ip-172-31-39-225:~/helloworld$ git add .
labsuser@ip-172-31-39-225:~/helloworld$ git commit -m "Add new files"
[master (root-commit) a9196b2] Add new files
1 file changed, 7 insertions(+)
create mode 100644 build.xml
labsuser@ip-172-31-39-225:~/helloworld$
```

2.7 Execute the following commands to link the local repository to a remote repository and push local changes to the remote master branch:

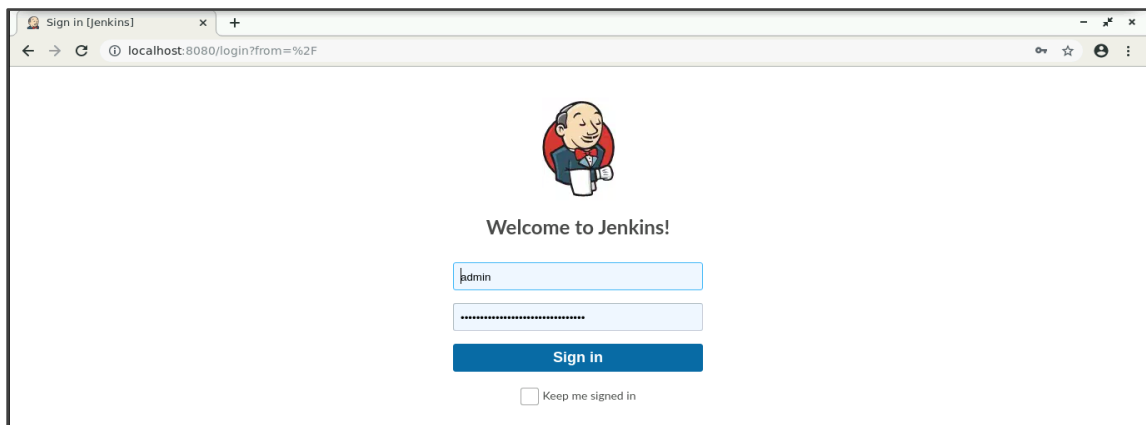
```
git remote add origin <Repository_URL.git>
git push -u origin master
```

```
labsuser@ip-172-31-32-183:~$ git remote add origin https://github.com/tasneemtaheraa/Demo-Ant.git

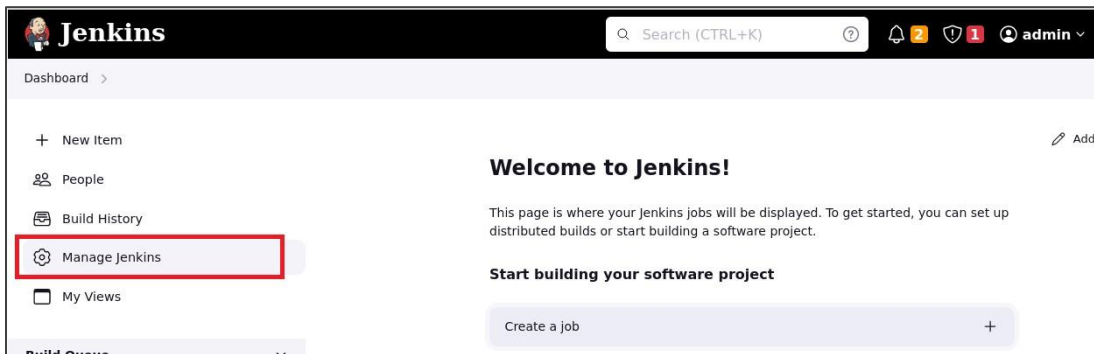
labsuser@ip-172-31-32-183:~$ git push -u origin master
Username for 'https://github.com': tasneemtaheraa
Password for 'https://tasneemtaheraa@github.com':
Enumerating objects: 647, done.
Counting objects: 100% (647/647), done.
Delta compression using up to 2 threads
```

### Step 3: Set the Global Tool Configuration

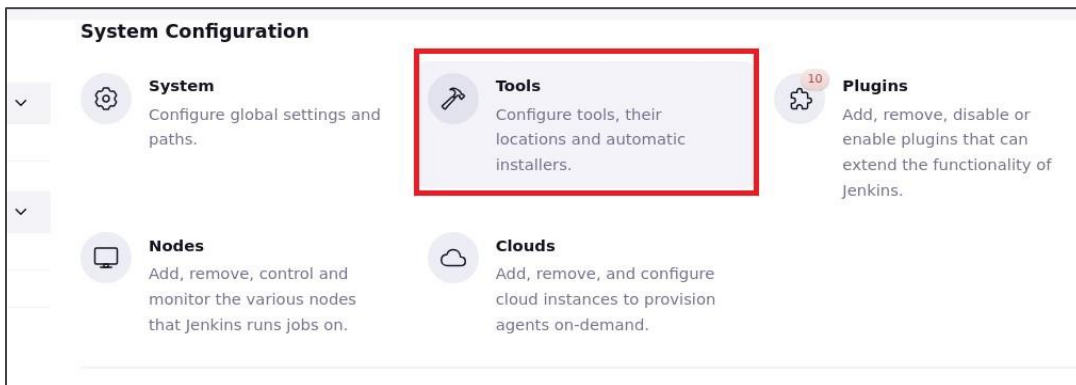
3.1 To access the Jenkins dashboard, open your browser, type **localhost:8080**, and enter the credentials, then click **Sign in**



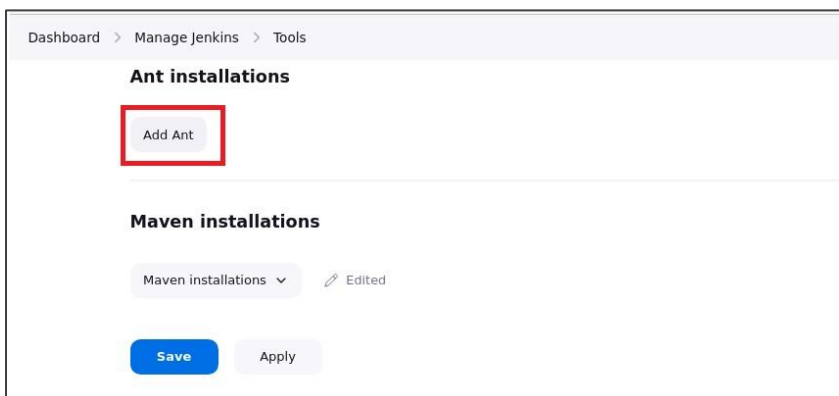
### 3.2 In the Jenkins Dashboard, click on **Manage Jenkins**



### 3.3 Choose **Tools** from the list of options



### 3.4 Click on **Add Ant** and provide the **Name** as **local\_ant**



Ant

Name

local\_ant

Required

☒ Install automatically

Install from Apache

Version

3.5 Select the **Version** you want to install, check the **Install automatically** box, then click on **Save**

localhost:8080/manage/configureTools/

Dashboard > Manage J

Ant

Name

lo

☒ Ins

1.10.14

1.10.13

1.10.12

1.10.11

1.10.10

1.10.9

1.10.8

1.10.7

1.10.6

1.10.5

1.10.4

1.10.3

1.10.2

1.10.1

1.10.0

1.9.16

1.9.15

1.9.14

1.9.13

1.9.12

1.10.14

☒ Install automatically

Install from Apache

Version

1.10.14

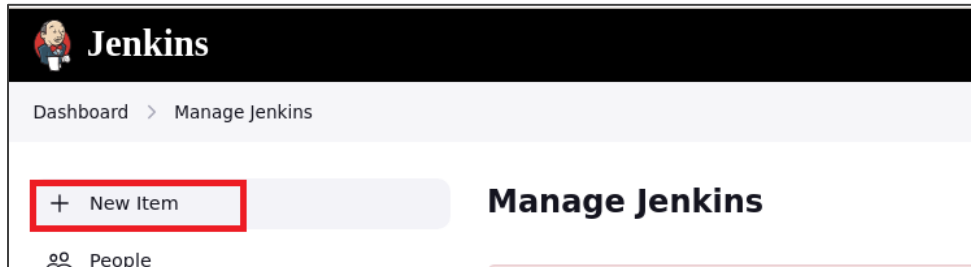
Save

Apply

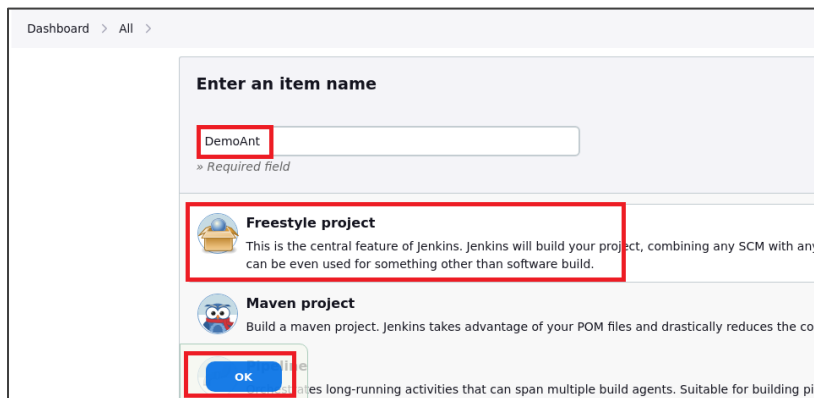


## Step 4: Integrate Ant with Jenkins

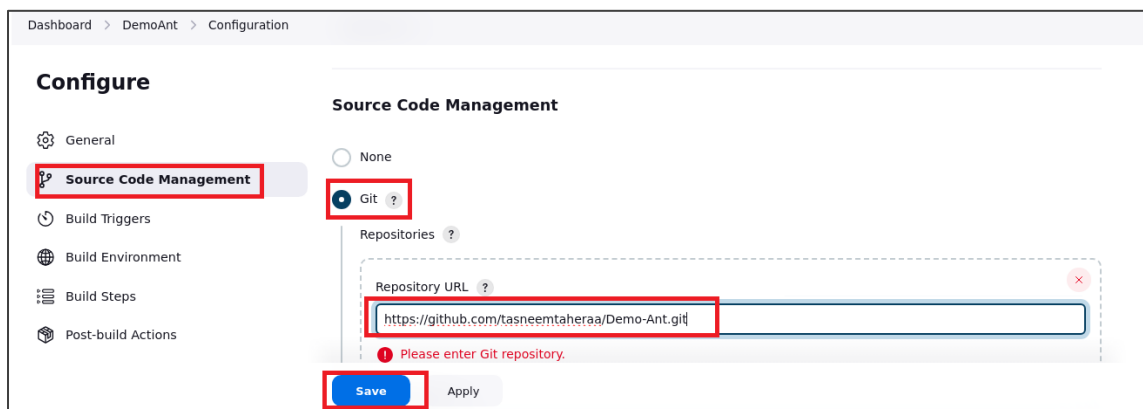
### 4.1 Click on **New Item** in the Jenkins dashboard



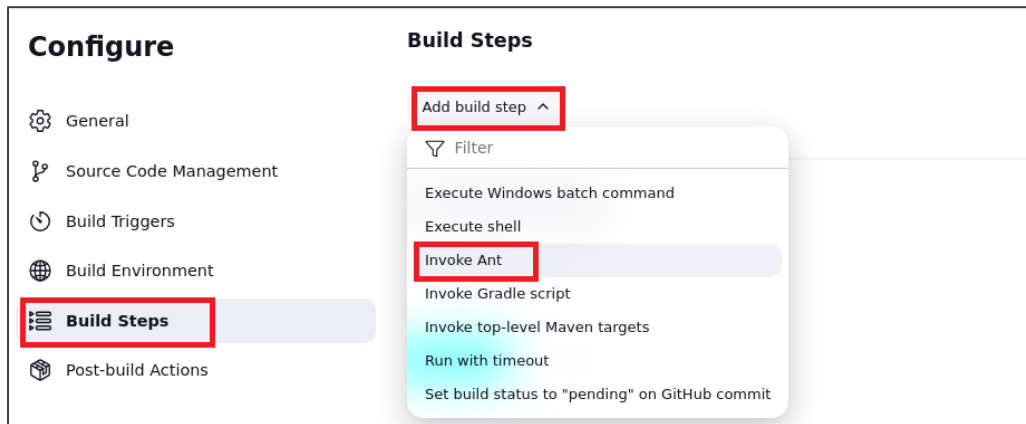
### 4.2 Enter an item name as **DemoAnt** for your project, select the **Freestyle project** as the build job type, and click on **OK**



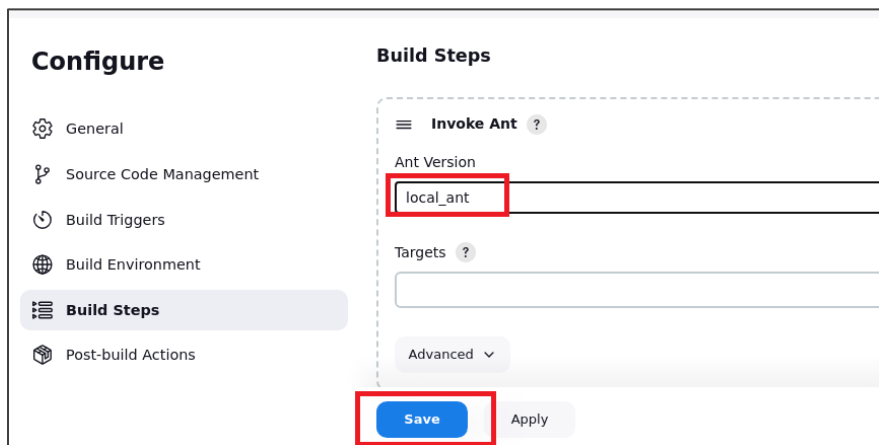
### 4.3 Scroll down to the **Source Code Management** section, select **Git**, enter the **Repository URL**, and click on **Save**



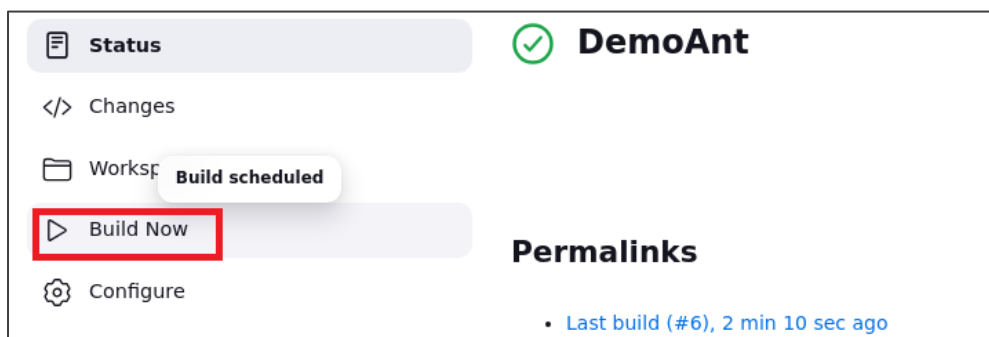
4.4 Navigate to **Build Steps**, click on the **Add build step** option, and select **Invoke Ant**



4.5 Select the **Ant Version** you want to work with and click on **Save**



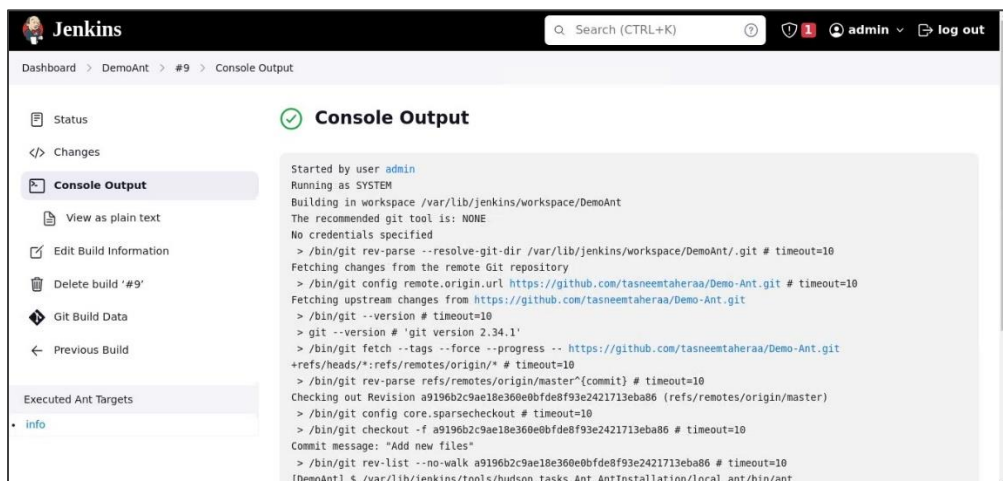
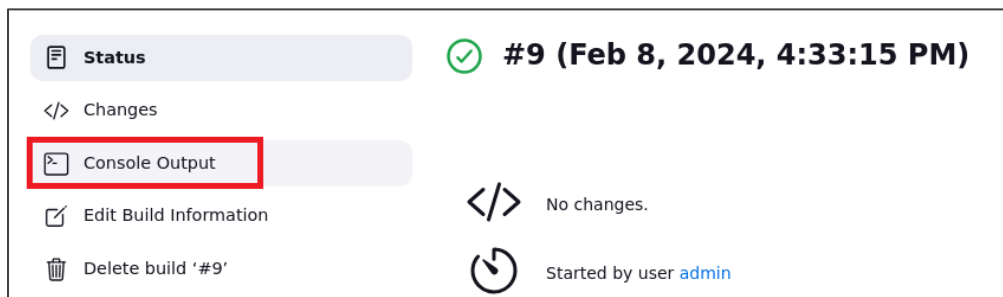
4.6 Then, click on **Build Now**



#### 4.7 Click on the **Build History** to view the build results



#### 4.8 Click on the **Console Output** to view the build logs



By following these steps, you have successfully installed the Ant plugin and integrated it with Jenkins, enabling the automation of build and deployment processes in Java projects.