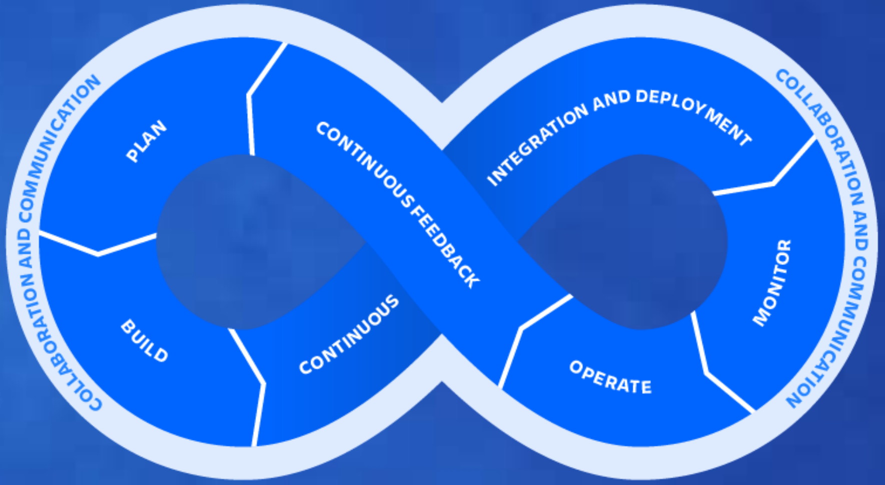

COLLABORATION AND COMMUNICATION
PLAN
BUILD
CONTINUOUS
CONTINUOUS FEEDBACK
INTEGRATION AND DEPLOYMENT
OPERATE
MONITOR
COLLABORATION AND COMMUNICATION

Session 7. Jobs in Jenkins

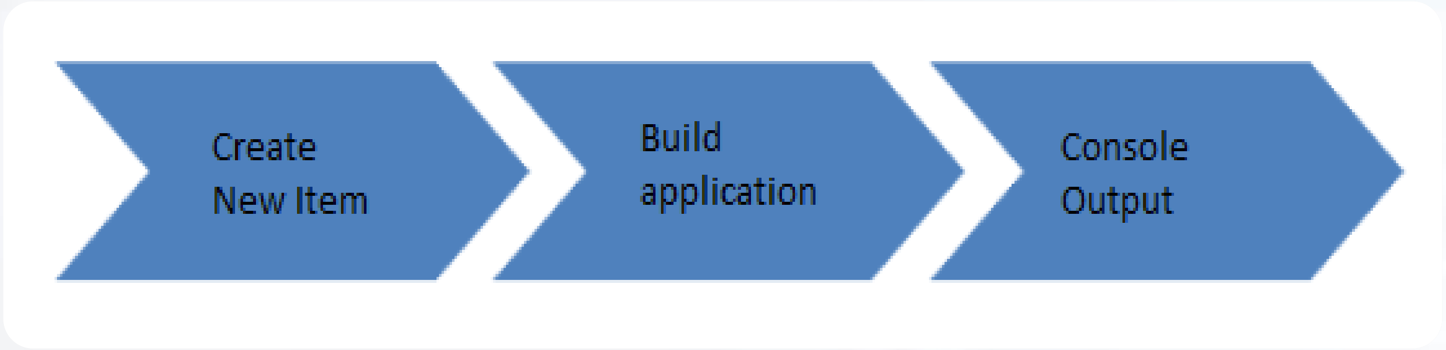
Ram N Sangwan

Agenda

- Creating Freestyle Jobs
- Running the Jobs
- Setting up the Global Environments for Jobs
- Adding and updating Plugins
- Disabling and deleting jobs

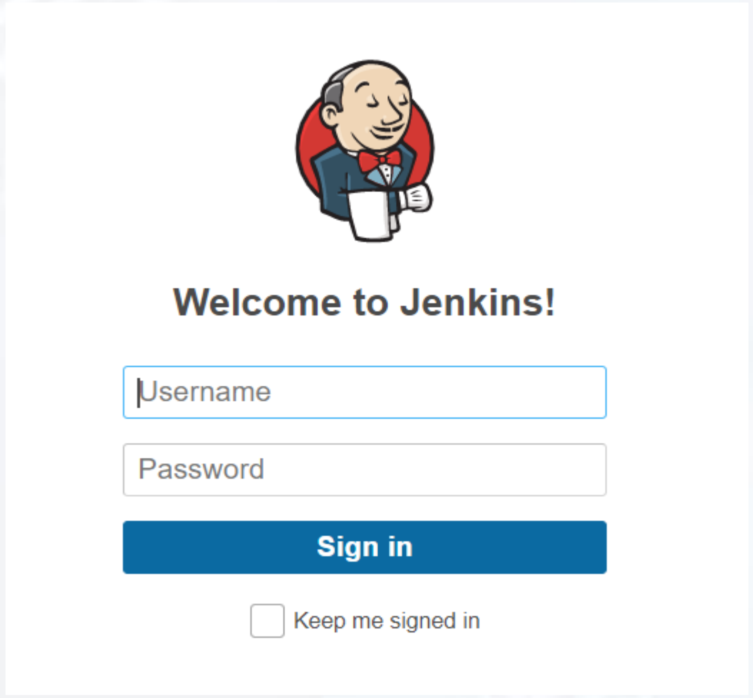
Jenkins Jobs

- A Jenkins project is a repeatable build job, which contains steps and post-build actions.
- The types of actions you can perform in a build step or post-build action are quite limited.
- There are many standard plugins available within a Jenkins Project to help you overcome this problem.
- They allow you to configure build triggers and offers project-based security for your Jenkins project.



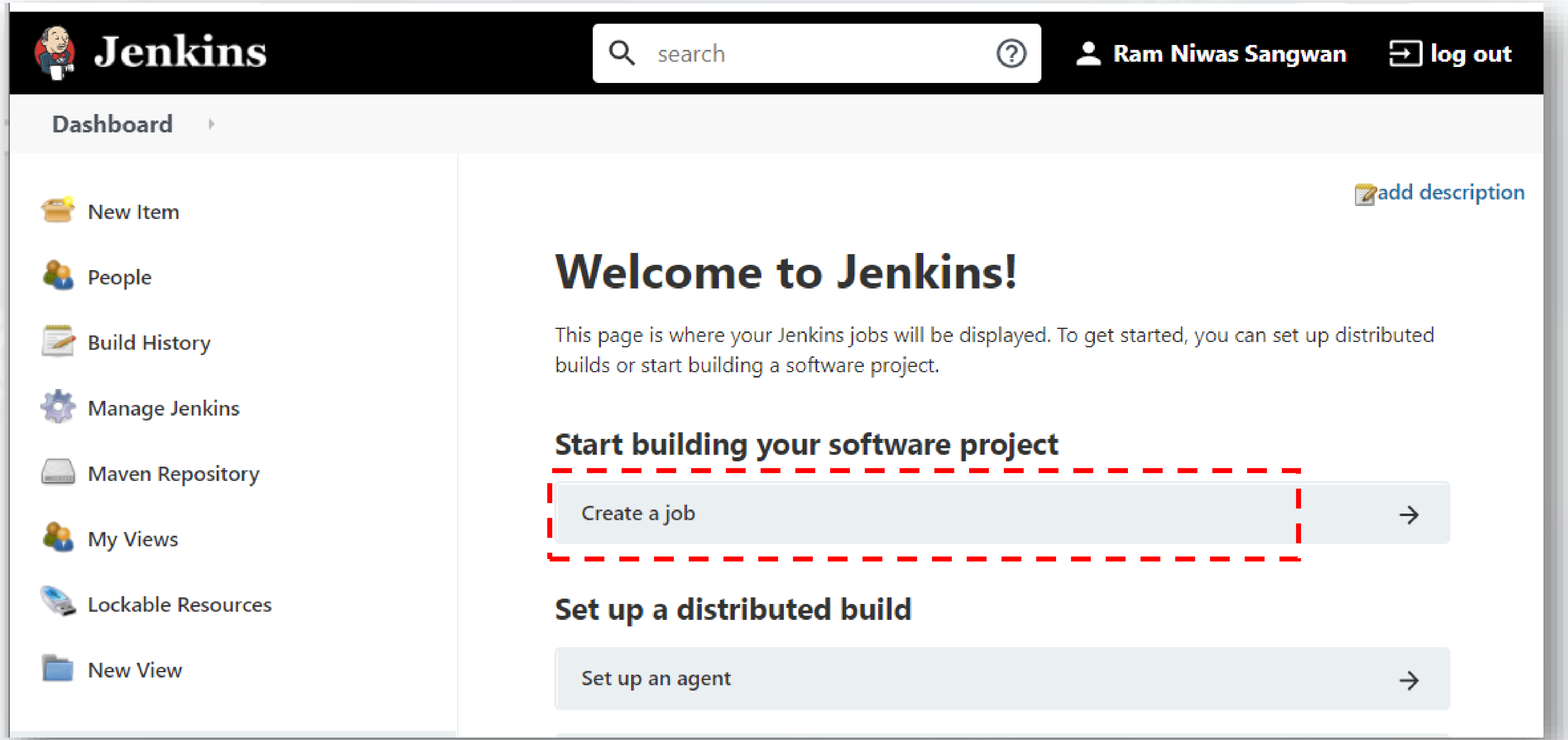
Creating a Freestyle Build Job

- The freestyle build job is a highly flexible and easy-to-use option.
- You can use it for any type of project; it is easy to set up, and many of its options appear in other build jobs.
- To create a Jenkins freestyle job, log on to your Jenkins dashboard.



Get Started

- Click on "**New Item**" at the centre of your dashboard



Welcome to Jenkins!

This page is where your Jenkins jobs will be displayed. To get started, you can set up distributed builds or start building a software project.

Start building your software project

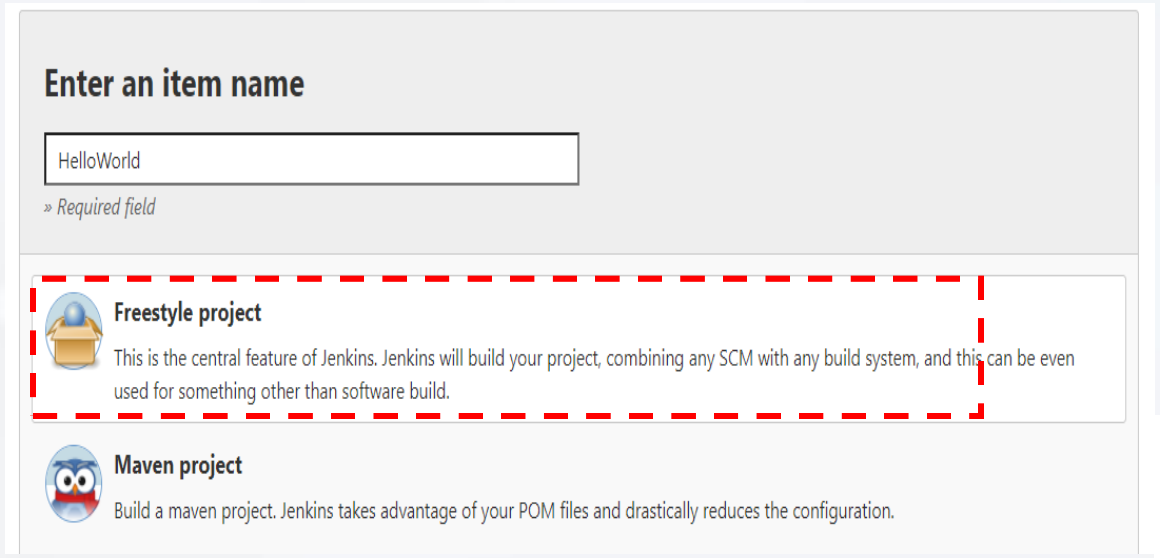
Create a job

Set up a distributed build

Set up an agent

Enter Name and Type of Project

- In the next screen, Enter the name of the item you want to create. We shall use the "Hello world" for this demo.
- Select Freestyle project.



Enter an item name

HelloWorld

» Required field

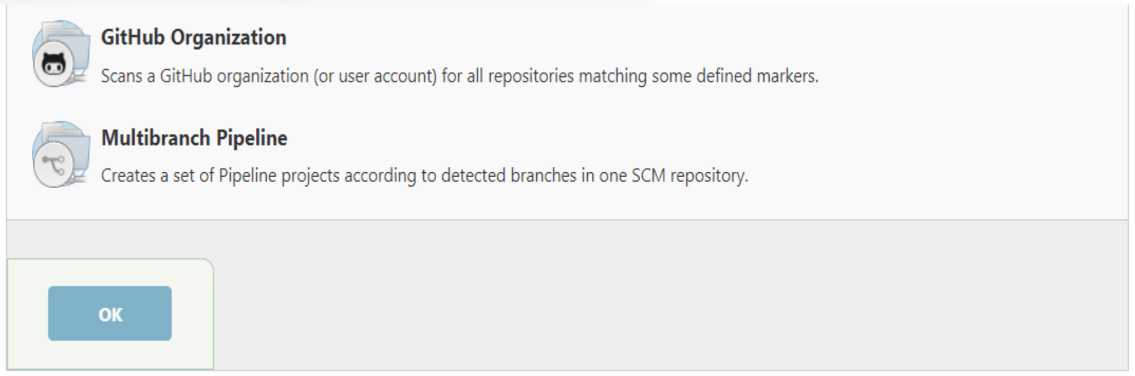
Freestyle project

This is the central feature of Jenkins. Jenkins will build your project, combining any SCM with any build system, and this can be even used for something other than software build.

Maven project

Build a maven project. Jenkins takes advantage of your POM files and drastically reduces the configuration.

- **Click Ok**



GitHub Organization

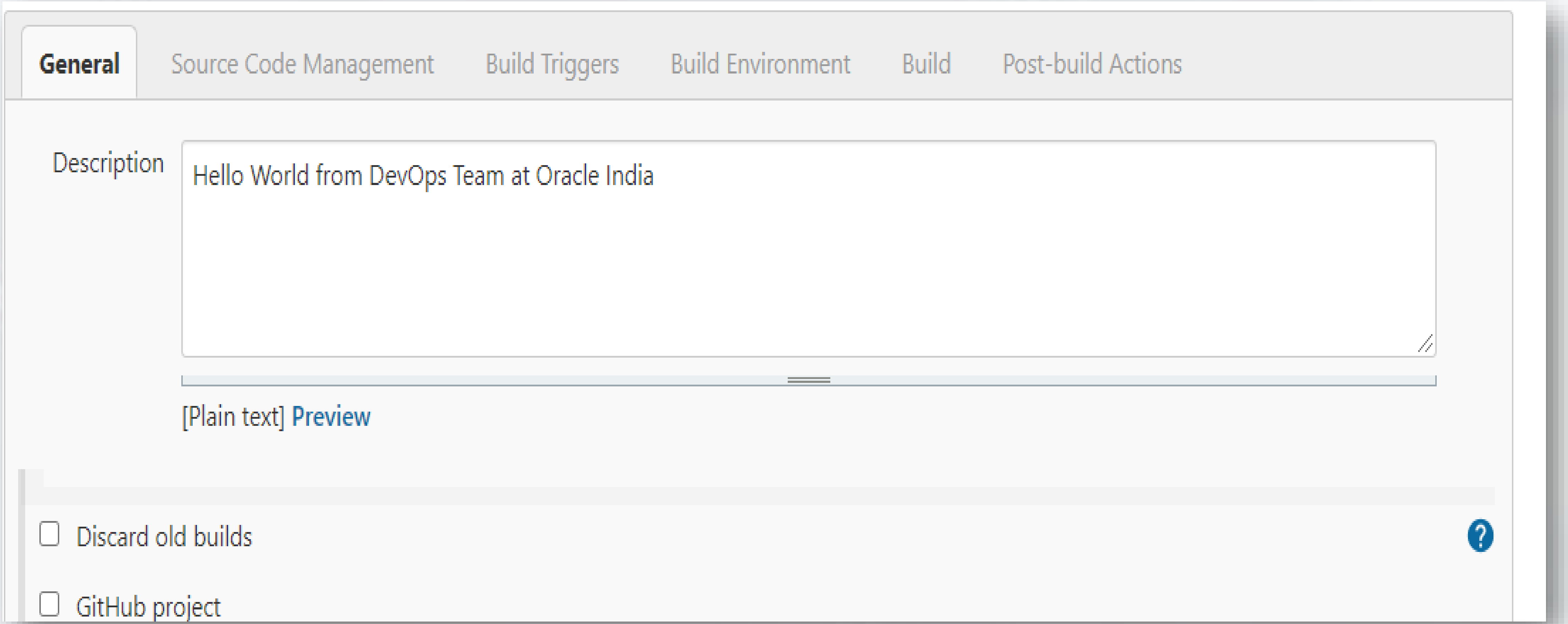
Scans a GitHub organization (or user account) for all repositories matching some defined markers.

Multibranch Pipeline

Creates a set of Pipeline projects according to detected branches in one SCM repository.

Describe Your Project

- Enter the details of the project you want to test.



Description: Hello World from DevOps Team at Oracle India

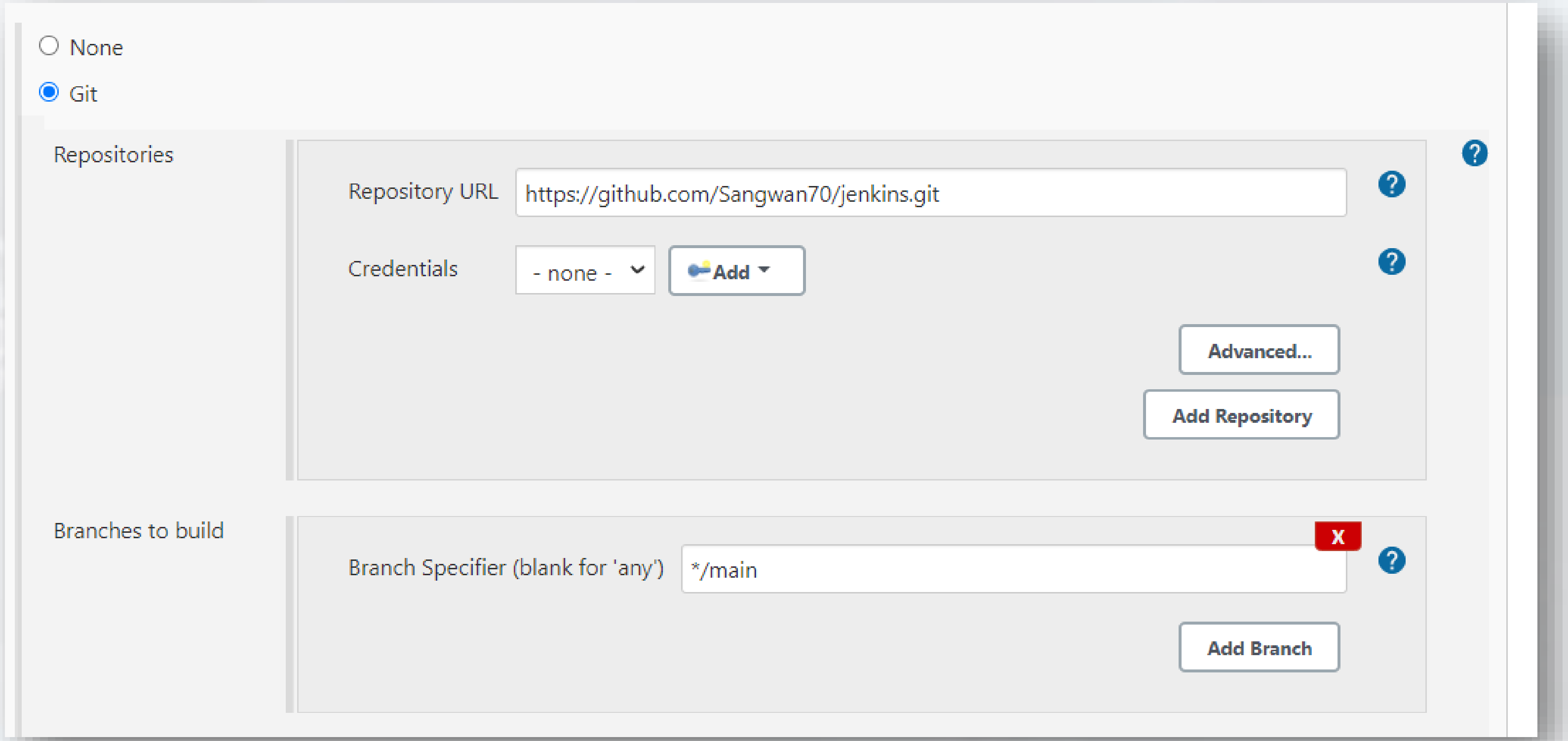
[Plain text] Preview

☐ Discard old builds

☐ GitHub project

Source Code Management

- If you have a Git Repository, Enter Details. Else, create one.
- Under Source Code Management, Enter your repository URL.
- A test repository is located at https://github.com/Sangwan70/jenkins.git



( ) None
(●) Git

Repositories

Repository URL: https://github.com/Sangwan70/jenkins.git

Credentials: - none -

Branches to build

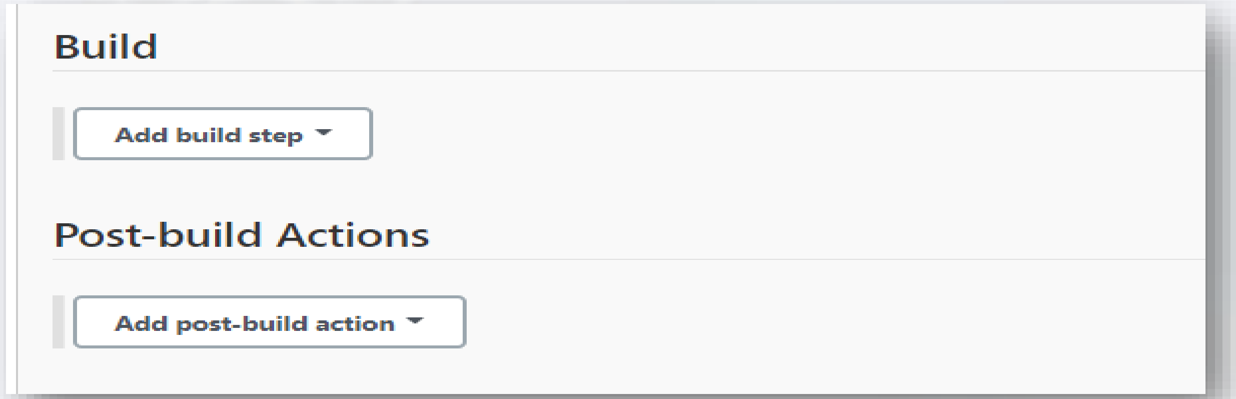
Branch Specifier (blank for 'any'): */main

More on Repository

- It is also possible for you to use a local repository.
- If your GitHub repository is private, Jenkins will first validate your login credentials with GitHub and only then pull the source code from your GitHub repository.

Build Steps for the Code

- Now that you have provided all the details, it's time to build the code.
- Tweak the settings under the **build** section to build the code at the time you want. You can even schedule the build to happen periodically, at set times.
- Under **build**, Click on "**Add build step**"

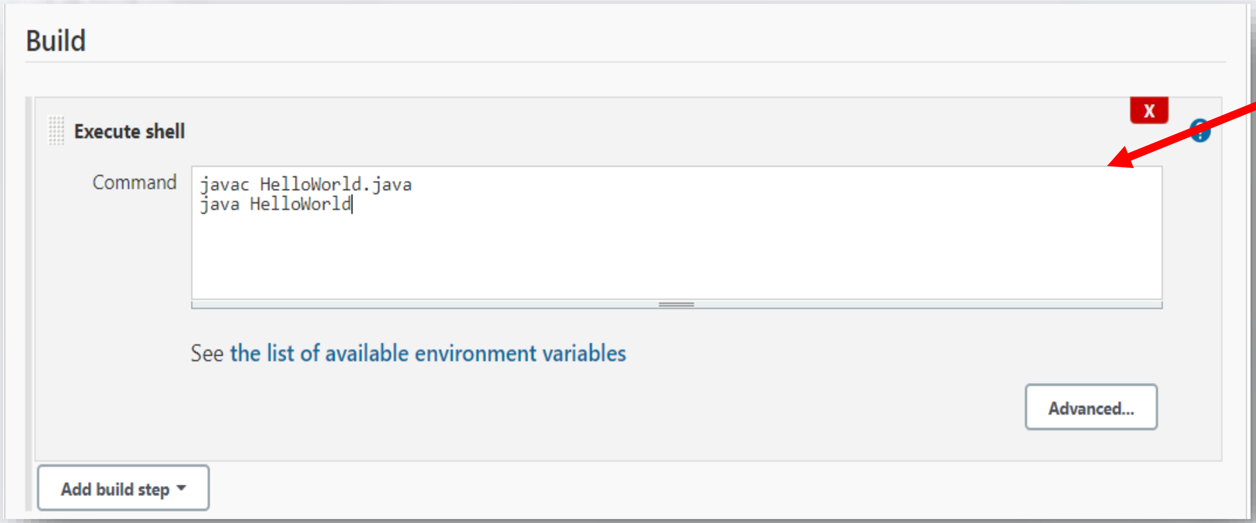


Commands to Execute Build

- Click on "**Execute shell**" and add the commands you want to execute during the build process.
- Here, I have added the java commands to compile the java code on my Linux.
- I have added the following Linux commands:

javac HelloWorld.java

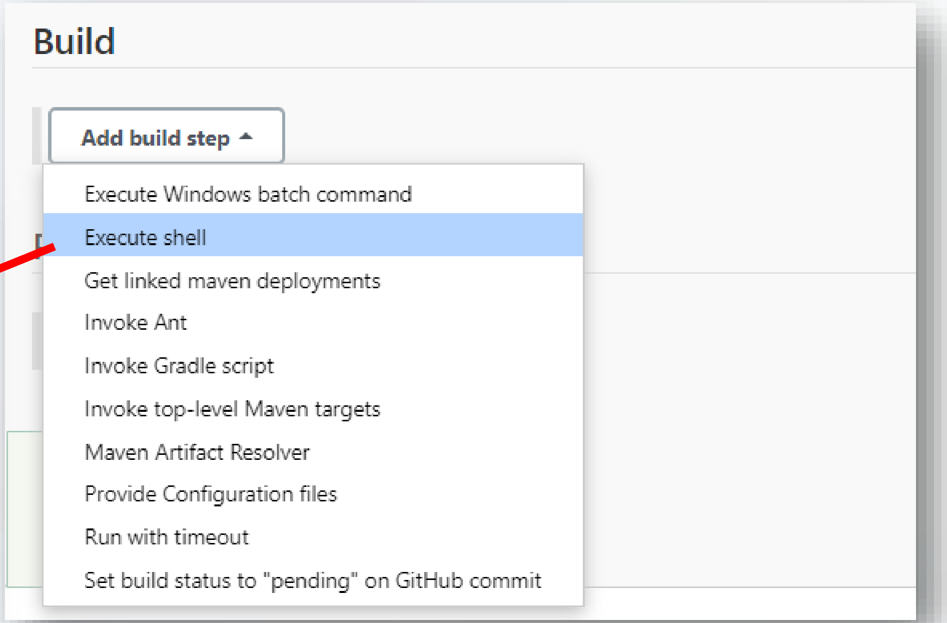
java HelloWorld



Command

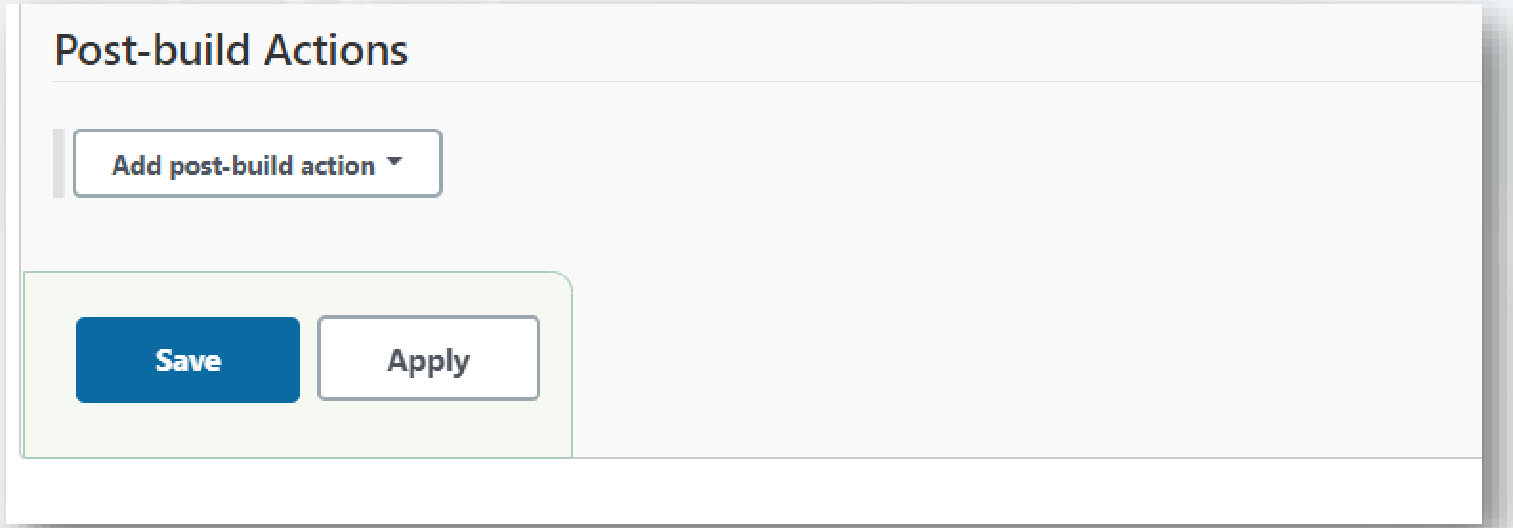
```
javac HelloWorld.java
java HelloWorld
```

See the list of available environment variables


Build
Add build step
Execute Windows batch command
Execute shell
Get linked maven deployments
Invoke Ant
Invoke Gradle script
Invoke top-level Maven targets
Maven Artifact Resolver
Provide Configuration files
Run with timeout
Set build status to "pending" on GitHub commit

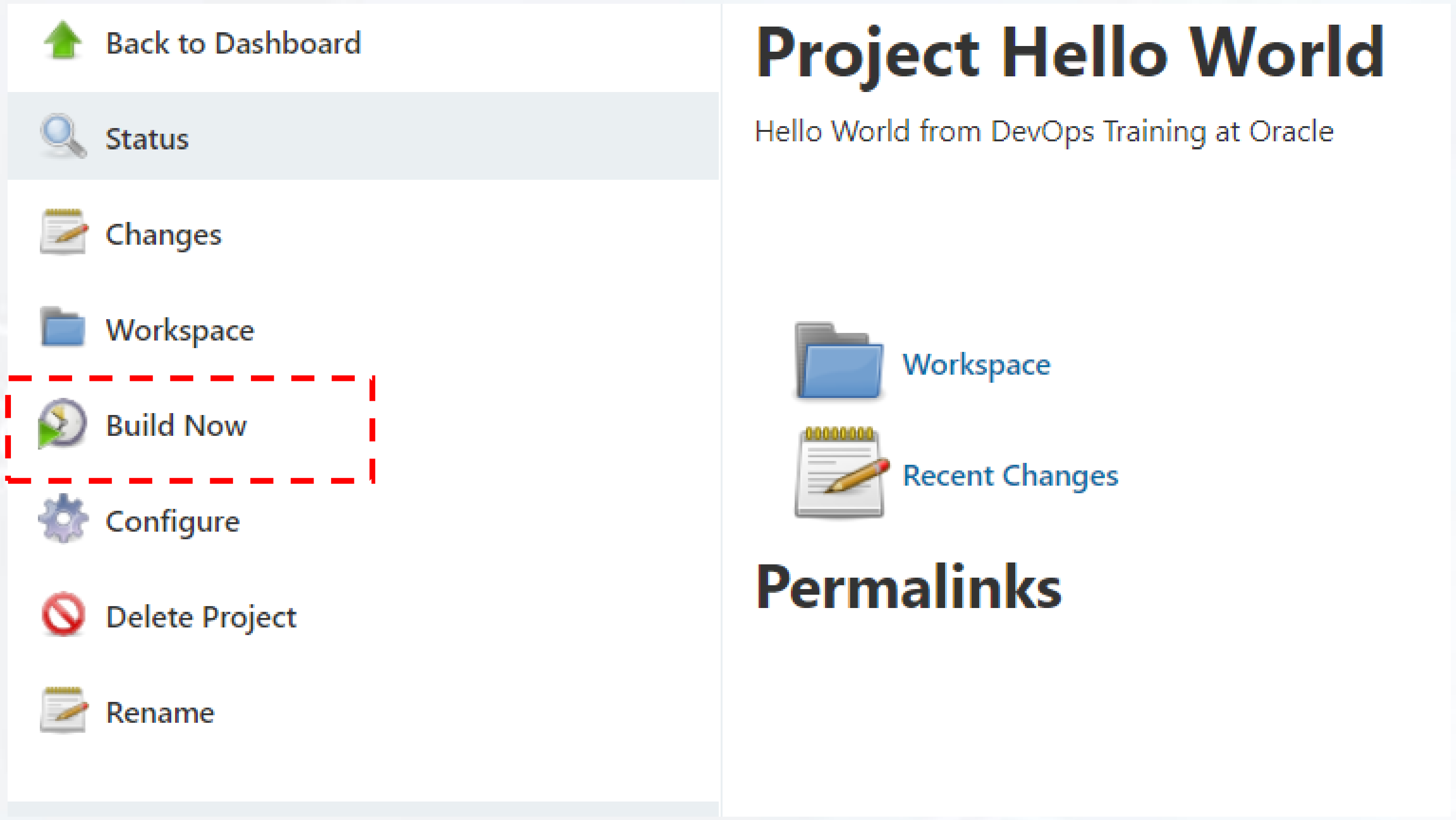
Save your Project

- When you have entered all the data,
- Click **Apply**
- **Save** the project.



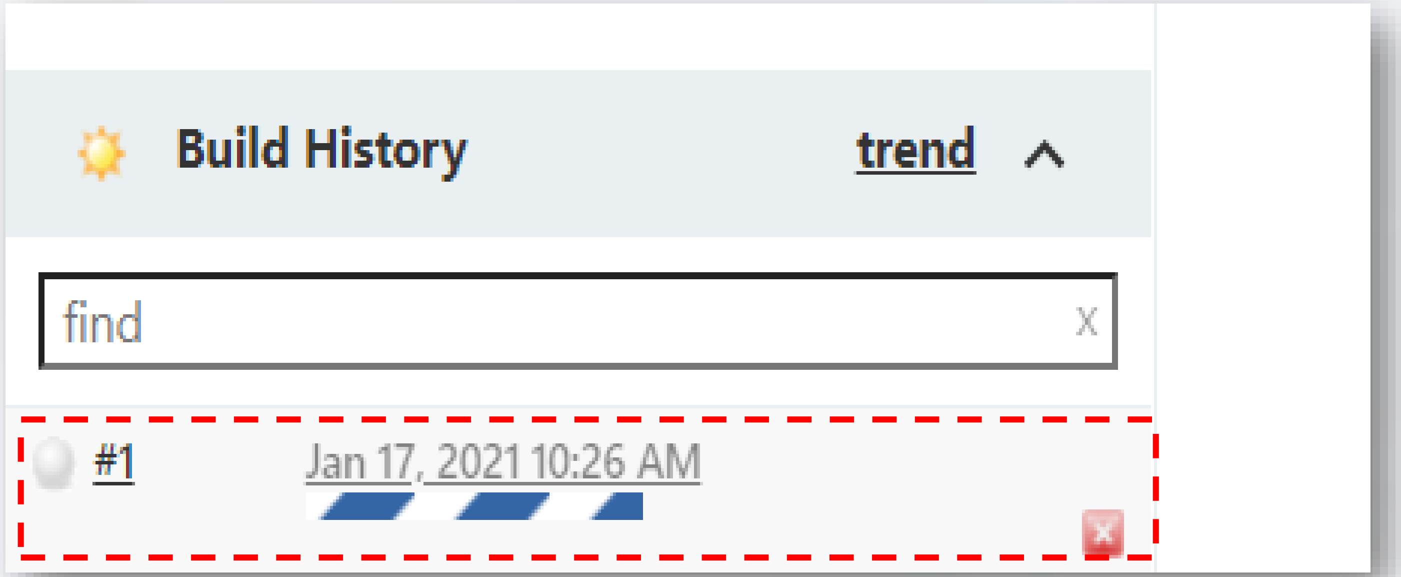
Build The Project

- Now, in the main screen, Click the **Build Now** button on the left-hand side to build the source code.



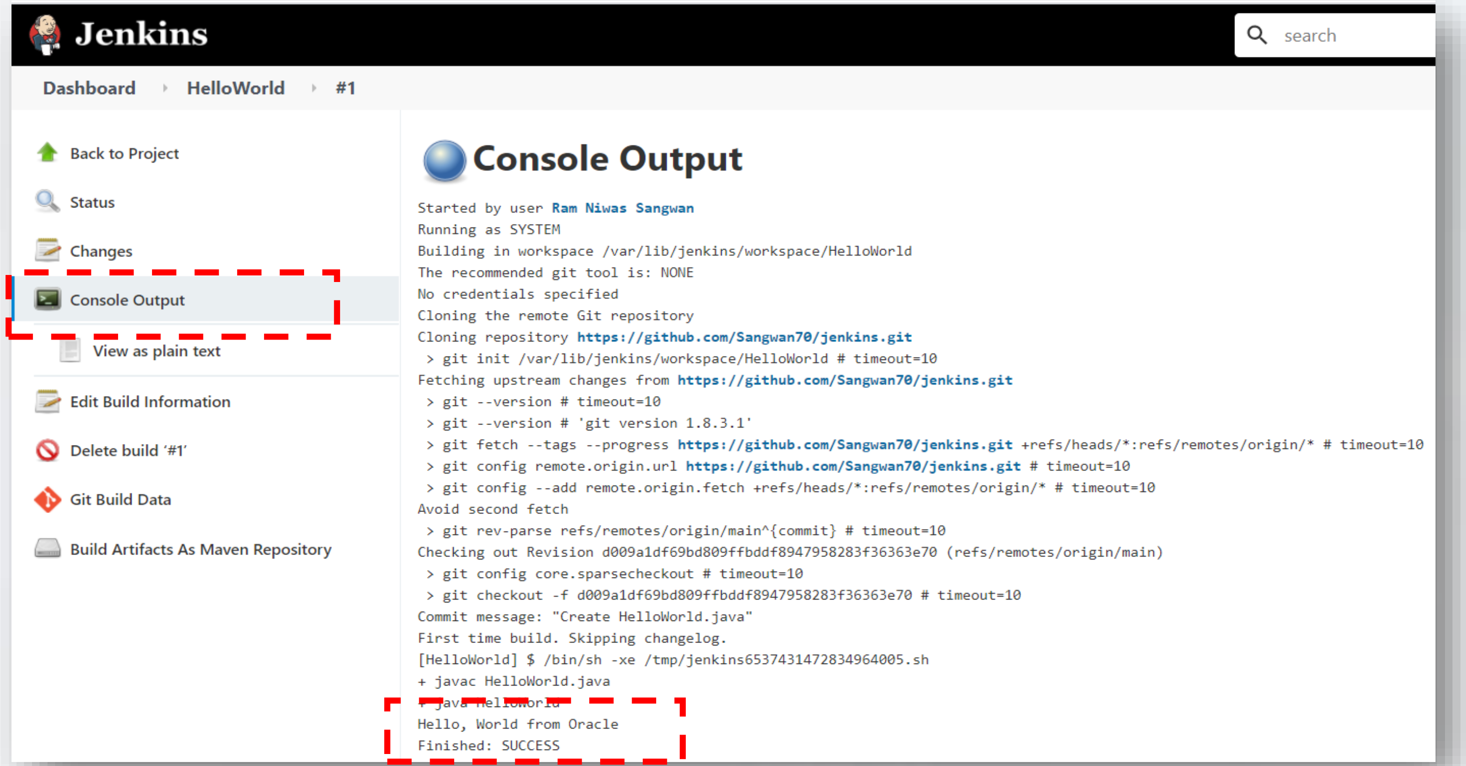
Build Status

- After clicking on **Build now,** you can see the status of the build you run under **Build History**.



Verify your Build

- Click on the **build number** and then Click on **console output** to see the status of the build you run.
- It should show you a success message, provided you have followed the setup properly.



Console Output

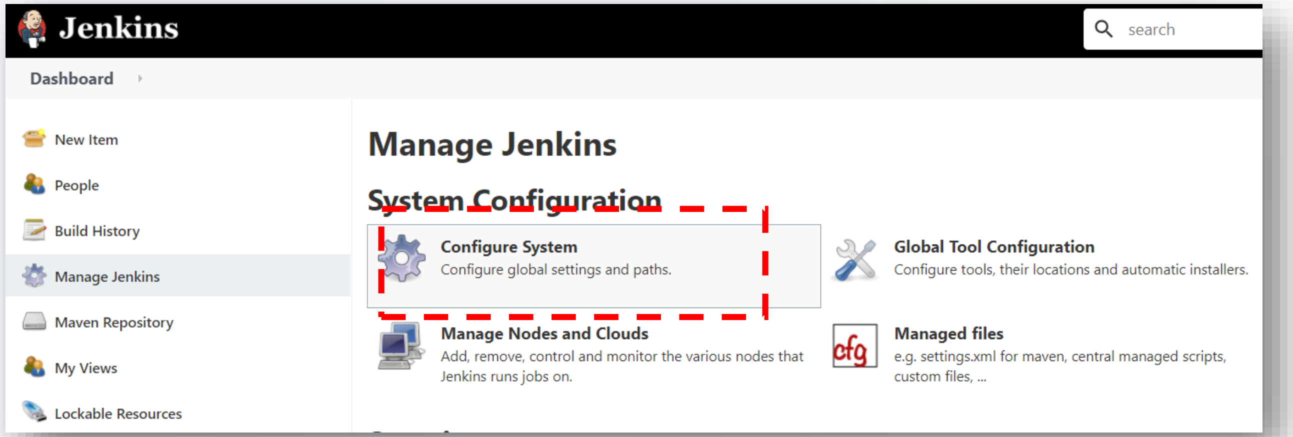
```
Started by user Ram Niwas Sangwan
Running as SYSTEM
Building in workspace /var/lib/jenkins/workspace/HelloWorld
The recommended git tool is: NONE
No credentials specified
Cloning the remote Git repository
Cloning repository https://github.com/Sangwan70/jenkins.git
 > git init /var/lib/jenkins/workspace/HelloWorld # timeout=10
Fetching upstream changes from https://github.com/Sangwan70/jenkins.git
 > git --version # timeout=10
 > git --version # 'git version 1.8.3.1'
 > git fetch --tags --progress https://github.com/Sangwan70/jenkins.git +refs/heads/*:refs/remotes/origin/* # timeout=10
 > git config remote.origin.url https://github.com/Sangwan70/jenkins.git # timeout=10
 > git config --add remote.origin.fetch +refs/heads/*:refs/remotes/origin/* # timeout=10
Avoid second fetch
 > git rev-parse refs/remotes/origin/main^{commit} # timeout=10
Checking out Revision d009a1df69bd809ffbddf8947958283f36363e70 (refs/remotes/origin/main)
 > git config core.sparsecheckout # timeout=10
 > git checkout -f d009a1df69bd809ffbddf8947958283f36363e70 # timeout=10
Commit message: "Create HelloWorld.java"
First time build. Skipping changelog.
[HelloWorld] $ /bin/sh -xe /tmp/jenkins6537431472834964005.sh
+ javac HelloWorld.java
+ java HelloWorld
Hello, World from Oracle
Finished: SUCCESS
```

The Hello World Program

- In sum, we have executed a HelloWorld program hosted on GitHub.
- Jenkin pulls the code from the remote repository and builds continuously at a frequency you define.

Setting up the Global Environments for Jobs

- Jenkins exposes characteristics of components using Environment variables.
- To configure global settings and paths, we need to go to Configure System in the Jenkins dashboard.
  1. Go to the Jenkins dashboard.
  2. Click on Manage Jenkins.
  3. Click on Configure System.
  4. Verify the Home directory available.



Manage Jenkins

System Configuration

Configure System Configure global settings and paths.

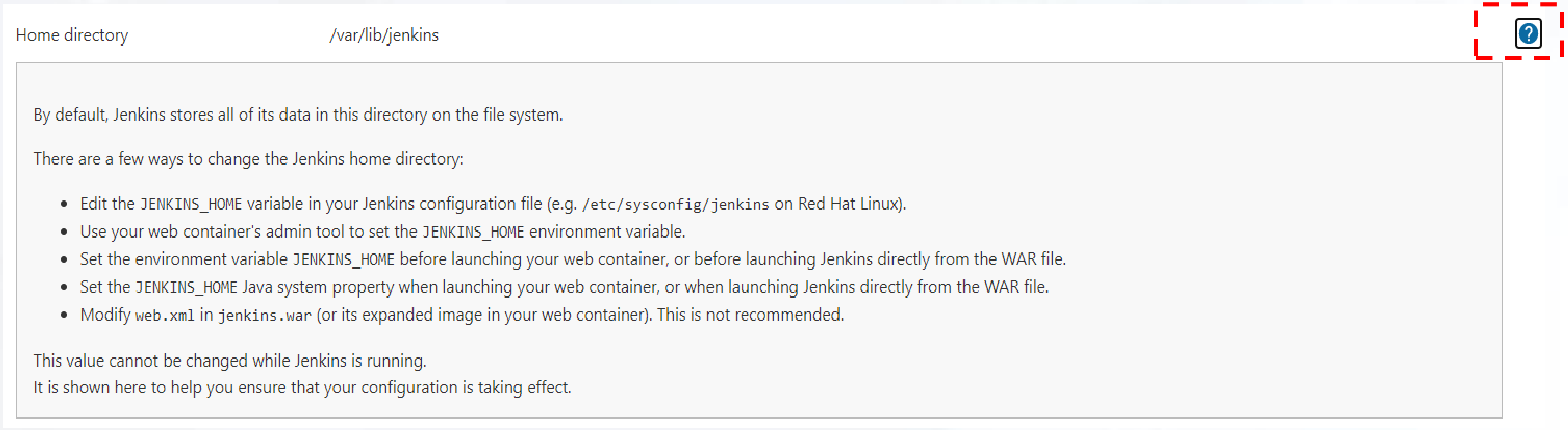
Global Tool Configuration Configure tools, their locations and automatic installers.

Manage Nodes and Clouds Add, remove, control and monitor the various nodes that Jenkins runs jobs on.

Managed files e.g. settings.xml for maven, central managed scripts, custom files, ...

Verify an Environment Variable

- In the **Configure System** page, we can define **Environment variables** too so that it can be used during the execution of build jobs.



Home directory /var/lib/jenkins

By default, Jenkins stores all of its data in this directory on the file system.

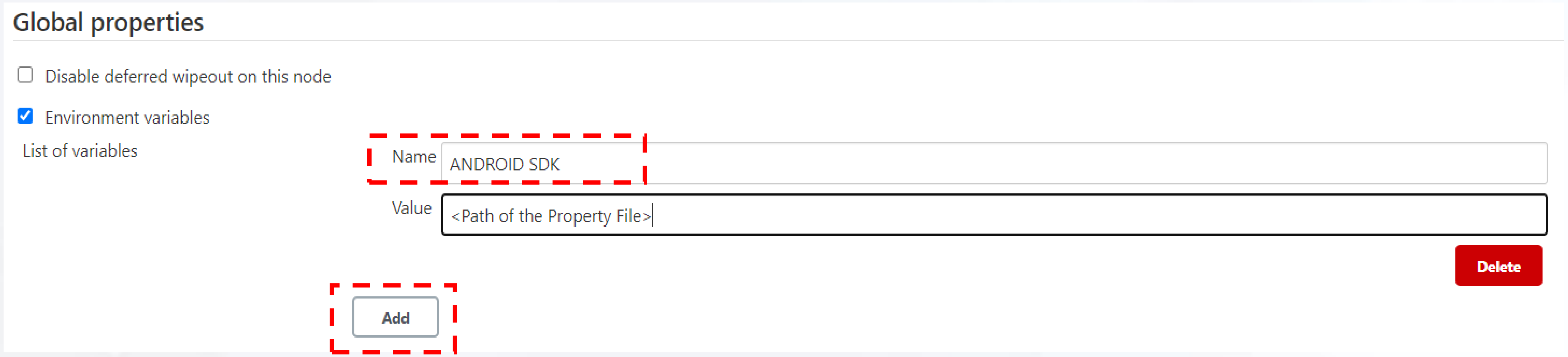
There are a few ways to change the Jenkins home directory:

- Edit the `JENKINS_HOME` variable in your Jenkins configuration file (e.g. `/etc/sysconfig/jenkins` on Red Hat Linux).
- Use your web container's admin tool to set the `JENKINS_HOME` environment variable.
- Set the environment variable `JENKINS_HOME` before launching your web container, or before launching Jenkins directly from the WAR file.
- Set the `JENKINS_HOME` Java system property when launching your web container, or when launching Jenkins directly from the WAR file.
- Modify `web.xml` in `jenkins.war` (or its expanded image in your web container). This is not recommended.

This value cannot be changed while Jenkins is running.
It is shown here to help you ensure that your configuration is taking effect.

Create Global Environment Variable

- For example, we can the ANDROID SDK path in **Environment variables** in to set Continuous Integration for Android Apps.


Global properties

☐ Disable deferred wipeout on this node

☑ Environment variables

List of variables

Name	ANDROID SDK
Value	<Path of the Property File>

Env Vars with Jenkinsfile

- We can set environment variables globally by declaring them in the environment directive of our Jenkinsfile.
- Let's see how to set two variables, DISABLE_AUTH and DB_ENGINE:

Jenkinsfile (Declarative Pipeline)

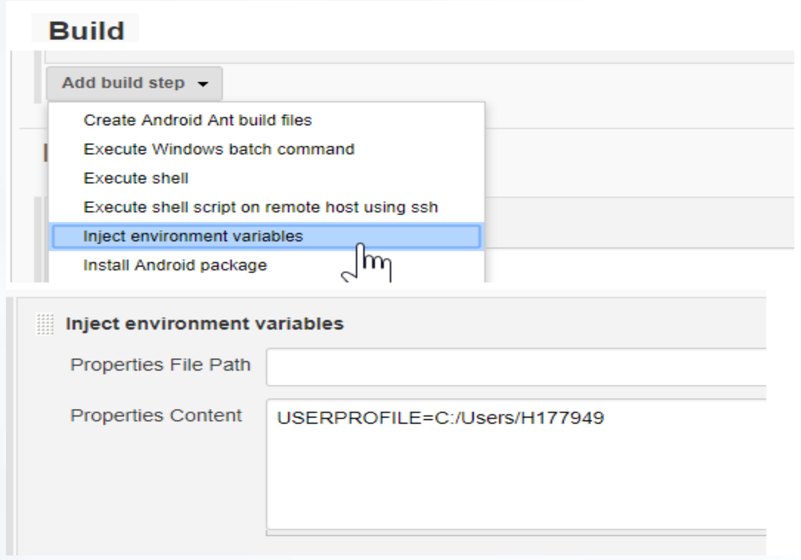
```
pipeline {
    //Setting the environment variables DISABLE_AUTH and DB_ENGINE
    environment {
        DISABLE_AUTH = 'true'
        DB_ENGINE    = 'mysql'
    }
}
```

- This approach of defining the variables in the Jenkins file is useful for instructing the scripts; for example, a Make file.

Env Vars with EnvInject

- We can install and use the EnvInject plugin to inject Env Vars during the build startup.
- In the build configuration window, we select the “***Inject environment variables***” option in the “***Add build step***” combo box.
- We can then add the environment variables in the properties content text box.
- For example, we can specify the user profile.
- Now, we can use any of our Env Var by surrounding the name in ${}:

echo "Database engine is ${DB_ENGINE}"


Build
Add build step
Create Android Ant build files
Execute Windows batch command
Execute shell
Execute shell script on remote host using ssh
Inject environment variables
Install Android package
Inject environment variables

Properties File Path

Properties Content: USERPROFILE=C:/Users/H177949

Adding and Updating Plugins

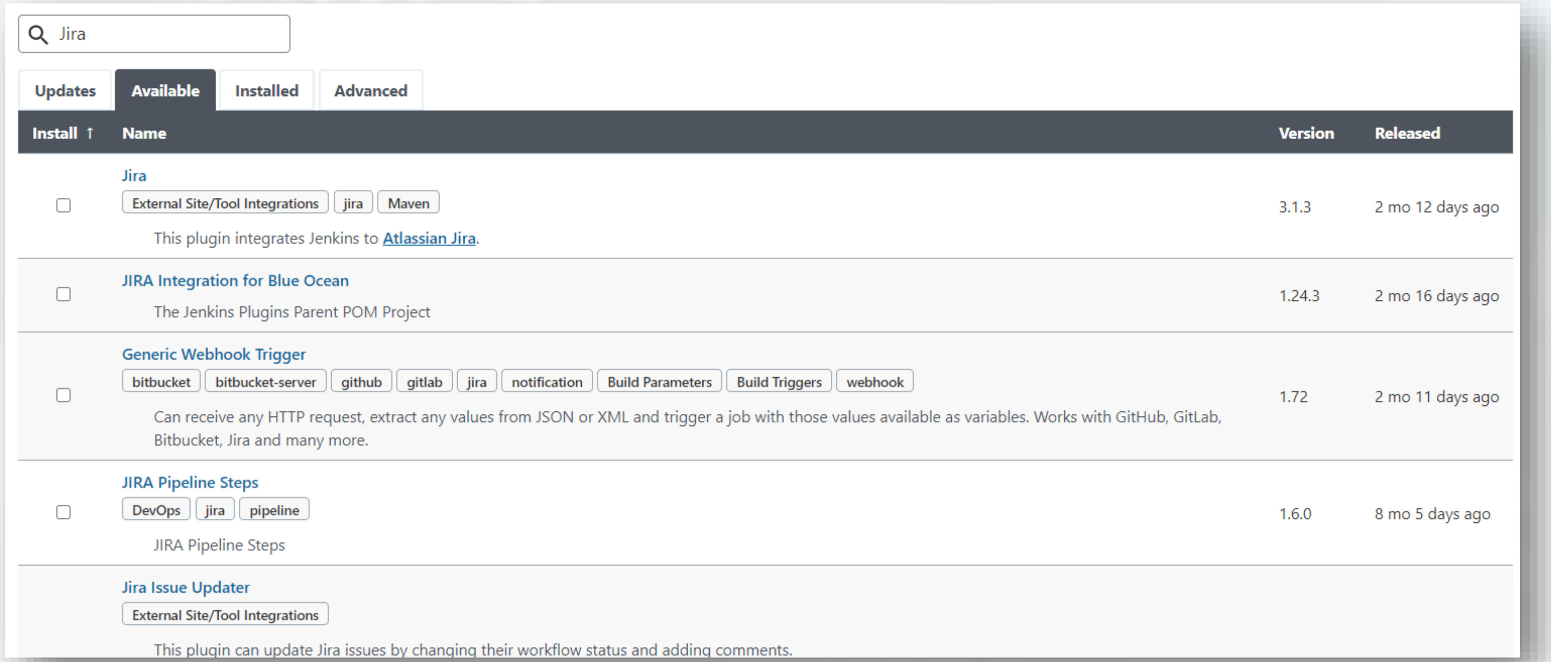
- Plugins are the primary means of enhancing the functionality of a Jenkins environment to suit organization or user-specific needs.
- There are over a thousand different plugins which can be installed on a Jenkins controller and to integrate various build tools, cloud providers, analysis tools, and much more.
- Plugins can be automatically downloaded, with their dependencies, from the Update Center.
- The Update Center is a service operated by the Jenkins project which provides an inventory of open source plugins which have been developed and maintained by various members of the Jenkins community.

Installing a plugin

- Jenkins provides a couple of different methods for installing plugins on the master:
  1. Using the "Plugin Manager" in the web UI.
  2. Using the Jenkins CLI install-plugin command.
- Each approach will result in the plugin being loaded by Jenkins but may require different levels of access and trade-offs in order to use.
- The two approaches require that the Jenkins controller be able to download meta-data from an Update Center, whether the primary Update Center operated by the Jenkins project, or a custom Update Center.
- The plugins are packaged as self-contained ***.hpi*** files, which have all the necessary code, images, and other resources which the plugin needs to operate successfully.

From the web UI

- The simplest and most common way of installing plugins is through the **Manage Jenkins** > **Manage Plugins** view, available to administrators of a Jenkins environment.
- Under the **Available** tab, plugins available for download from the configured Update Center can be searched and considered:



Jira

Updates | Available | Installed | Advanced

Install ↑	Name	Version	Released
☐	Jira External Site/Tool Integrations · jira · Maven This plugin integrates Jenkins to Atlassian Jira.	3.1.3	2 mo 12 days ago
☐	JIRA Integration for Blue Ocean The Jenkins Plugins Parent POM Project	1.24.3	2 mo 16 days ago
☐	Generic Webhook Trigger bitbucket · bitbucket-server · github · gitlab · jira · notification · Build Parameters · Build Triggers · webhook Can receive any HTTP request, extract any values from JSON or XML and trigger a job with those values available as variables. Works with GitHub, GitLab, Bitbucket, Jira and many more.	1.72	2 mo 11 days ago
☐	JIRA Pipeline Steps DevOps · jira · pipeline JIRA Pipeline Steps	1.6.0	8 mo 5 days ago
	Jira Issue Updater External Site/Tool Integrations This plugin can update Jira issues by changing their workflow status and adding comments.		

Using the Jenkins CLI

- Administrators may also use the Jenkins CLI which provides a command to install plugins.
- Scripts to manage Jenkins environments, or configuration management code, may need to install plugins without direct user interaction in the web UI.
- The Jenkins CLI allows a command line user or automation tool to download a plugin and its dependencies.

Jenkins CLI Syntax

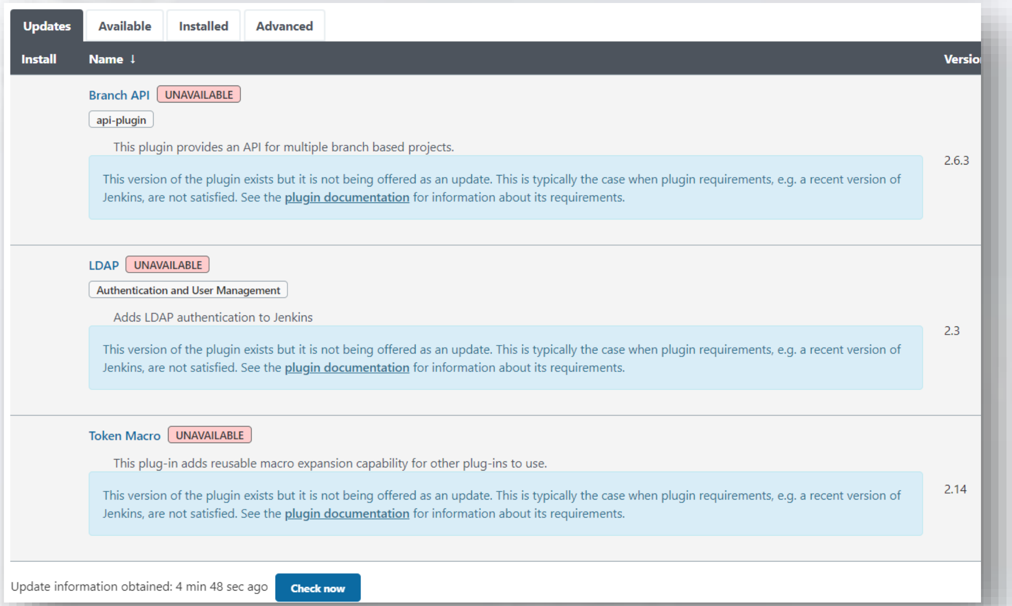
java -jar jenkins-cli.jar -s http://localhost:8080/ install-plugin SOURCE ... [-deploy] [-name VAL] [-restart]

Installs a plugin either from a file, an URL, or from update center.

SOURCE	*: If this points to a local file, that file will be installed. If this is an URL, Jenkins downloads the URL and installs that as a plugin. Otherwise the name is assumed to be the short name of the plugin in the existing update center (like "findbugs"),and the plugin will be installed from the update center.*
-deploy	*: Deploy plugins right away without postponing them until the reboot.*
-name VAL	*: If specified, the plugin will be installed as this short name (whereas normally the name is inferred from the source name automatically).*
-restart	*: Restart Jenkins upon successful installation.*

Updating a plugin

- Updates are listed in the **Updates** tab of the **Manage Plugins** page and can be installed by checking the checkboxes of the desired plugin updates and clicking the **Download now and install after restart** button.
- By default, the Jenkins controller will check for updates from the Update Center once every 24 hours. To manually trigger a check for updates, simply click on the **Check now** button in the **Updates** tab.



Branch API UNAVAILABLE
api-plugin
This plugin provides an API for multiple branch based projects.
This version of the plugin exists but it is not being offered as an update. This is typically the case when plugin requirements, e.g. a recent version of Jenkins, are not satisfied. See the plugin documentation for information about its requirements.
2.6.3

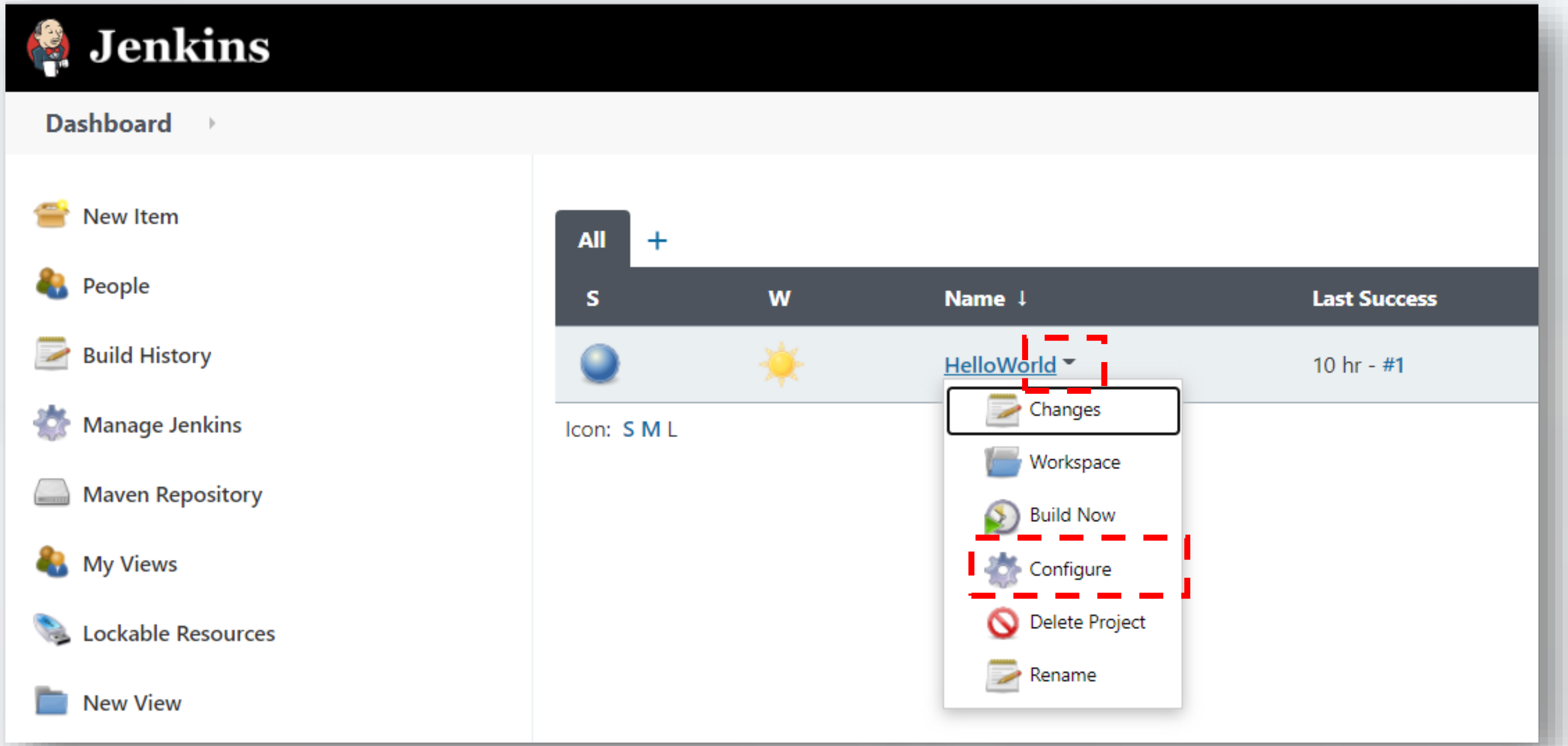
LDAP UNAVAILABLE
Authentication and User Management
Adds LDAP authentication to Jenkins
This version of the plugin exists but it is not being offered as an update. This is typically the case when plugin requirements, e.g. a recent version of Jenkins, are not satisfied. See the plugin documentation for information about its requirements.
2.3

Token Macro UNAVAILABLE
This plug-in adds reusable macro expansion capability for other plug-ins to use.
This version of the plugin exists but it is not being offered as an update. This is typically the case when plugin requirements, e.g. a recent version of Jenkins, are not satisfied. See the plugin documentation for information about its requirements.
2.14

Update information obtained: 4 min 48 sec ago

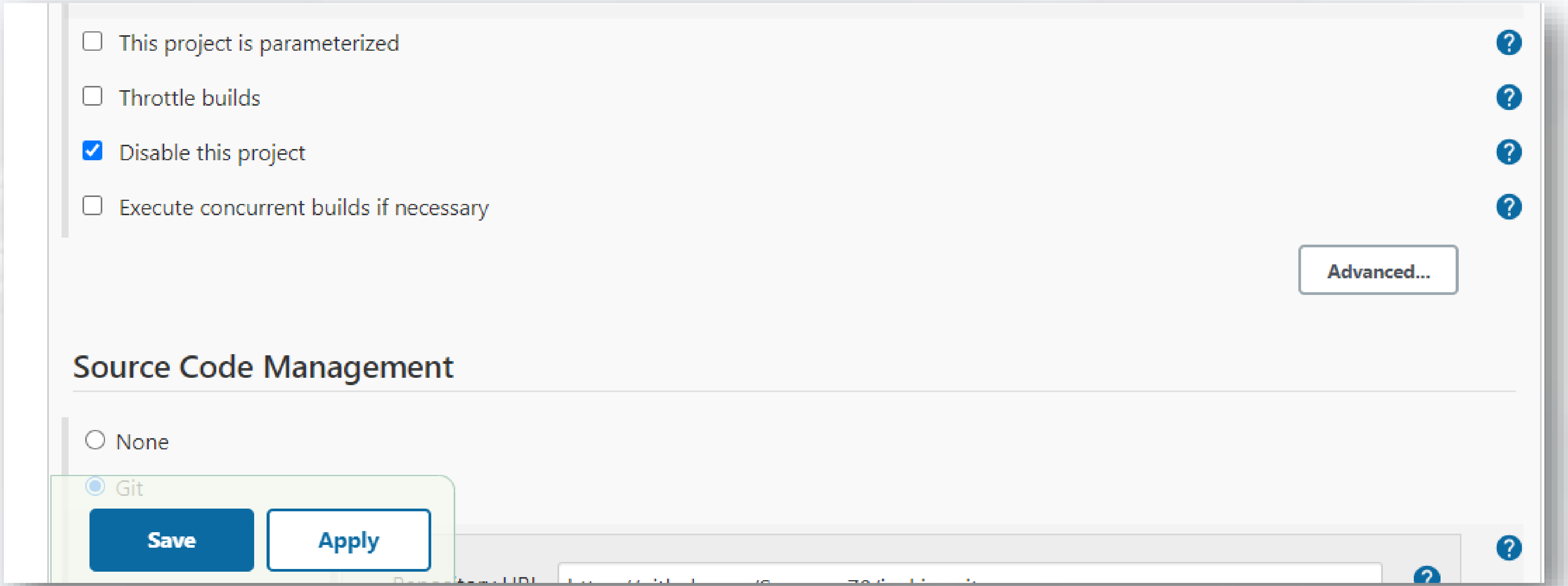
Disabling Jobs

- From the Project Dashboard, Click on the small down arrow at the end of your Project Name.
- From the menu that appears, click on Configure.



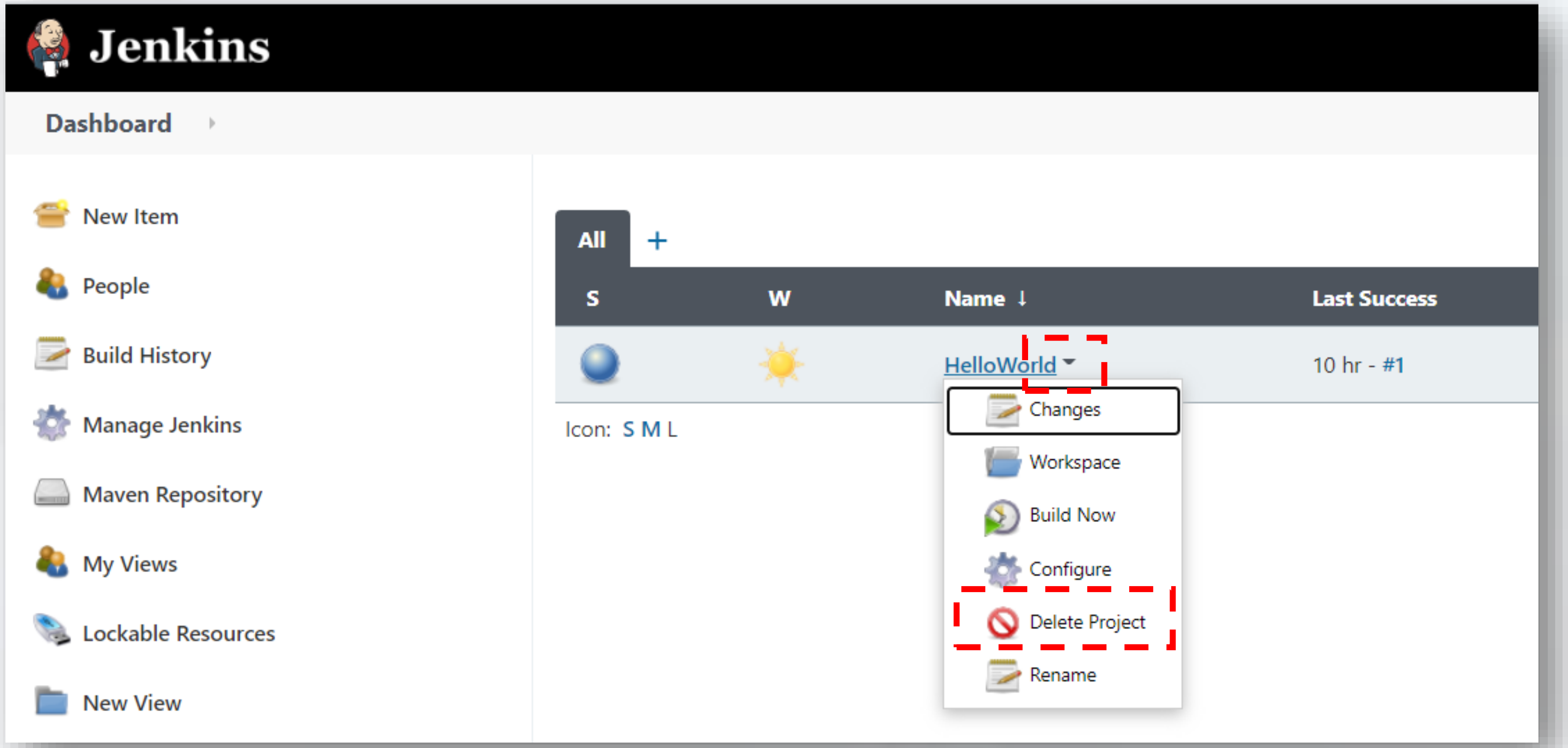
Apply and Save

- Under the General Tab, Click on the Checkbox ***Disable this Project***.
- ***Apply*** and ***Save*** your Project


- [ ] This project is parameterized
- [ ] Throttle builds
- [x] Disable this project
- [ ] Execute concurrent builds if necessary

Deleting Jobs

- From the Project Dashboard, Click on the small down arrow at the end of your Project Name.
- From the menu that appears, click on Delete Project.



Thank You