To install Jenkins on your Ubuntu system, follow these steps:

1. Initial ubuntu setup

```
curl -sL https://raw.githubusercontent.com/prabhatpankaj/ubuntustarter/master/initial.sh | sh
```

2. Install Java

- Since Jenkins is a Java application, the first step is to install Java. Update the package index and install the Java 8 OpenJDK package with the following commands:

```
sudo apt update
sudo apt install openjdk-8-jdk
```

- Setting the JAVA_HOME Environment Variable

Many programs written using Java use the JAVA_HOME environment variable to determine the Java installation location.
To set this environment variable, first determine where Java is installed. Use the update-alternatives command:

```
sudo update-alternatives --config java
```

- Copy the path from your preferred installation. Then open /etc/environment using nano or your favorite text editor:

```
sudo nano /etc/environment
```

- At the end of this file, add the following line, making sure to replace the highlighted path with your own copied path:

```
JAVA_HOME="/usr/lib/jvm/java-8-openjdk-amd64/jre/bin/"
```

- Modifying this file will set the JAVA_HOME path for all users on your system.

```
source /etc/environment
```

- Verify that the environment variable is set:

```
echo $JAVA_HOME
```

4. **Install Node.js from the repositories:**

```
sudo apt install nodejs
```

- If the package in the repositories suits your needs, this is all you need to do to get set up with Node.js. In most cases, you'll also want to also install npm, the Node.js package manager. You can do this by typing:

```
sudo apt install npm
```

3. Install Jenkins

- The current version of Jenkins does not support Java 10 (and Java 11) yet. If you have multiple versions of Java installed on your machine make sure Java 8 is the default Java version.

- Add the Jenkins Debian repository.

- Import the GPG keys of the Jenkins repository using the following wget command:

```
wget -q -O - https://pkg.jenkins.io/debian/jenkins.io.key | sudo apt-key add -
```

- The commands above should output OK which means that the key has been successfully imported and packages from this repository will be considered trusted.

- Next, add the Jenkins repository to the system with:

```
sudo sh -c 'echo deb http://pkg.jenkins.io/debian-stable binary/ > /etc/apt/sources.list.d/jenkins.list'
```

- When both of these are in place, run update so that apt will use the new repository:

```
sudo apt update
```

- Finally, install Jenkins and its dependencies:

```
sudo apt install jenkins
```

- Now that Jenkins and its dependencies are in place, we'll start the Jenkins server.

4. Start Jenkins

- Let's start Jenkins using systemctl:

```
sudo systemctl start jenkins
```

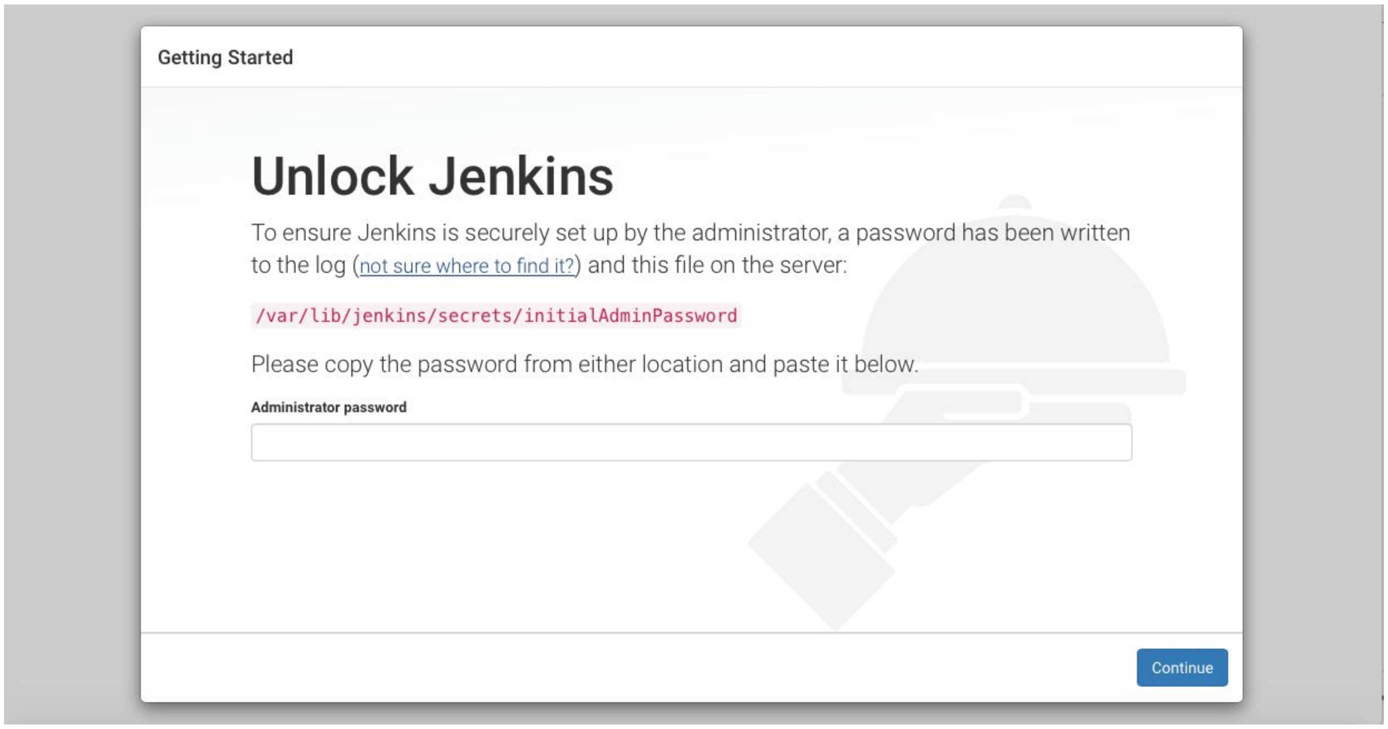
- Since systemctl doesn't display output, you can use its status command to verify that Jenkins started successfully:

```
sudo systemctl status jenkins
```

If everything went well, the beginning of the output should show that the service is active and configured to start at boot:

Output

```
● jenkins.service - LSB: Start Jenkins at boot time
  Loaded: loaded (/etc/init.d/jenkins; generated)
```


Getting Started

Unlock Jenkins

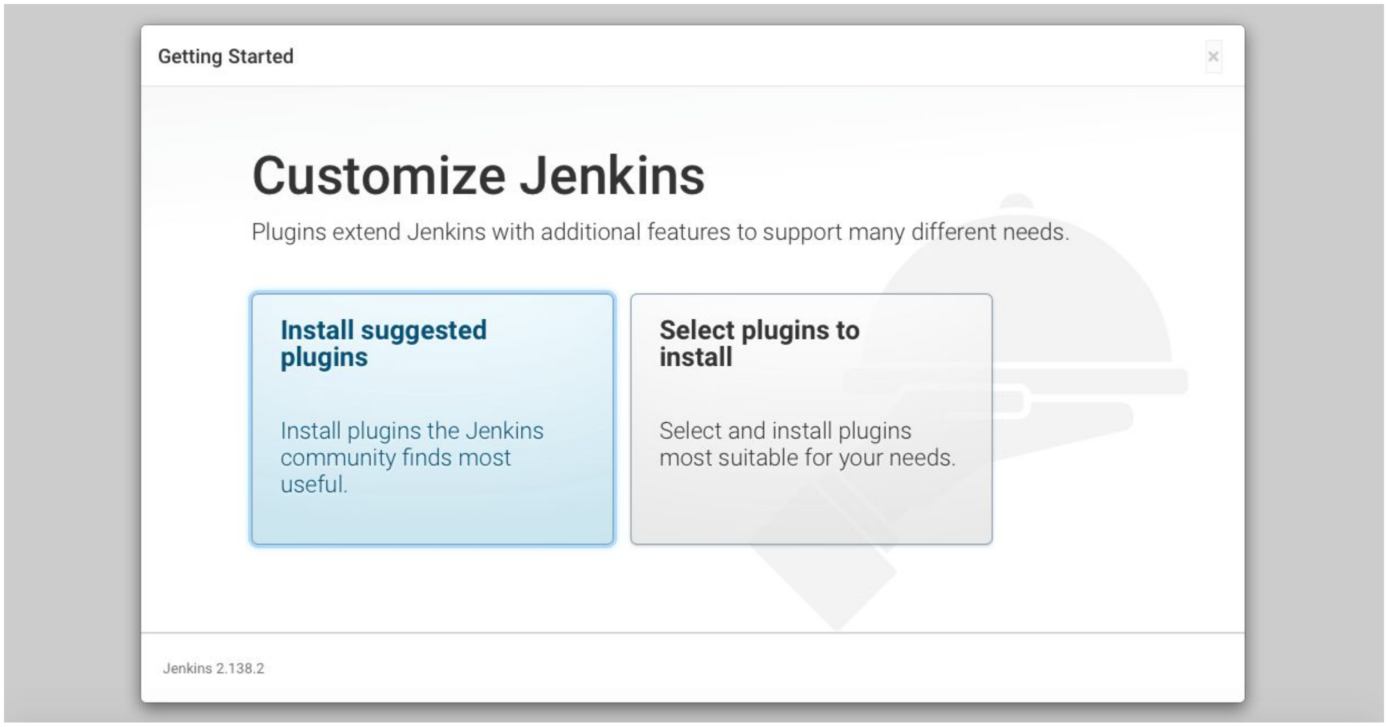
To ensure Jenkins is securely set up by the administrator, a password has been written to the log (not sure where to find it?) and this file on the server:

`/var/lib/jenkins/secrets/initialAdminPassword`

Please copy the password from either location and paste it below.

Administrator password

sudo cat /var/lib/jenkins/secrets/initialAdminPassword


Getting Started

Customize Jenkins

Plugins extend Jenkins with additional features to support many different needs.

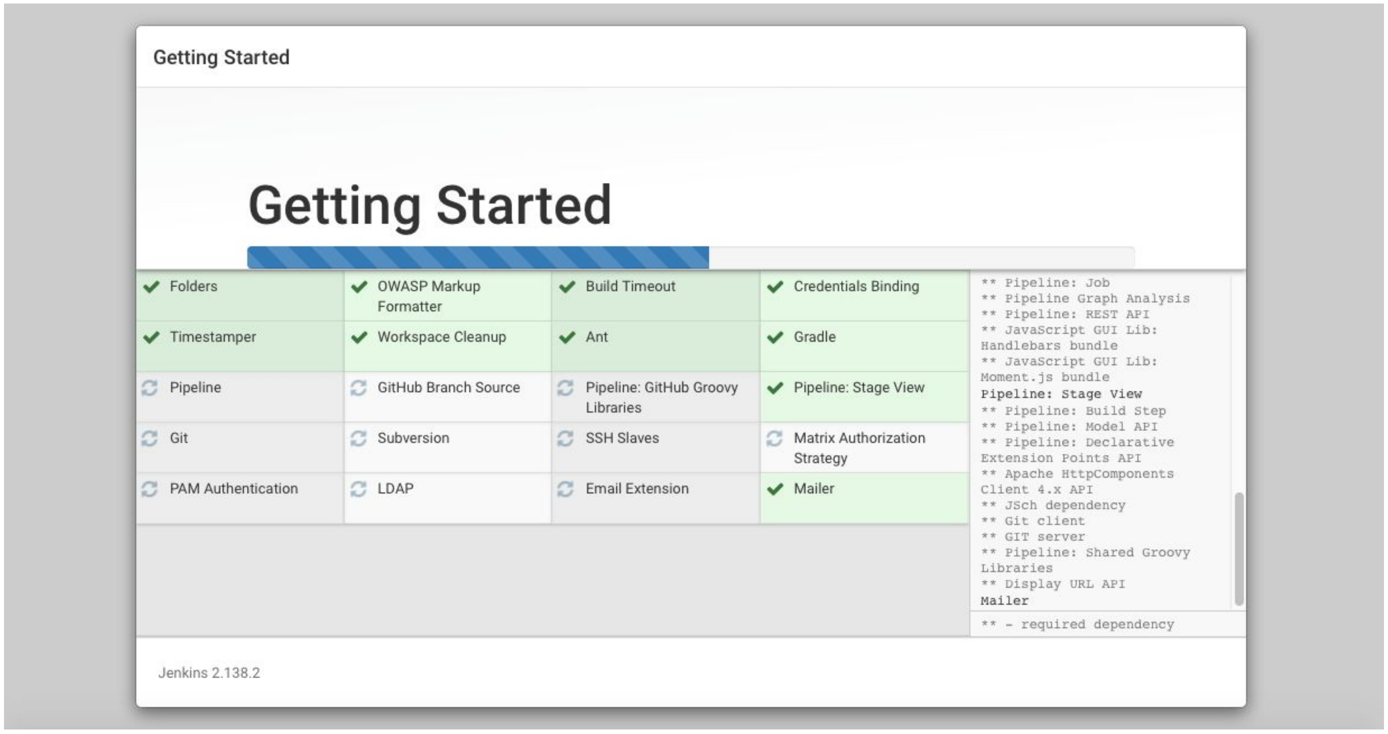
Install suggested plugins

Install plugins the Jenkins community finds most useful.

Select plugins to install

Select and install plugins most suitable for your needs.

Jenkins 2.138.2

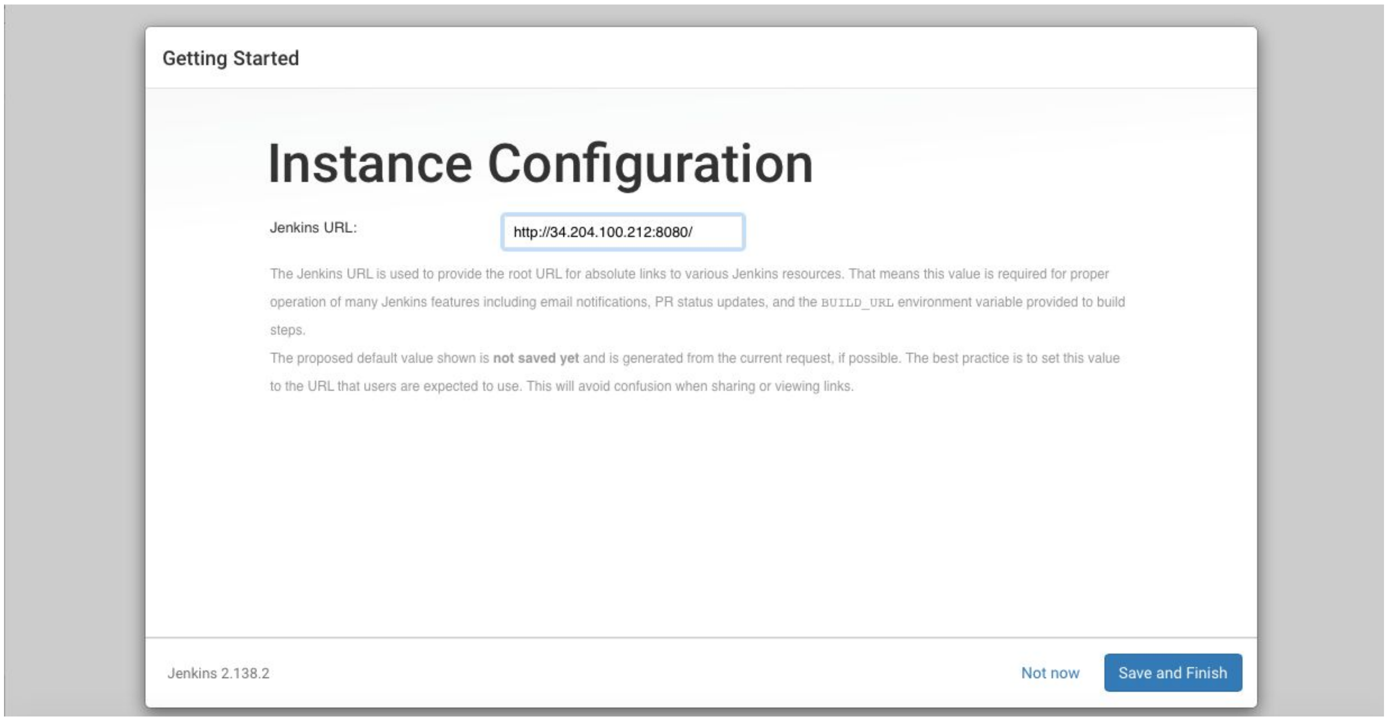

Getting Started

Getting Started

✓ Folders	✓ OWASP Markup Formatter	✓ Build Timeout	✓ Credentials Binding
✓ Timestamper	✓ Workspace Cleanup	✓ Ant	✓ Gradle
Pipeline	GitHub Branch Source	Pipeline: GitHub Groovy Libraries	✓ Pipeline: Stage View
Git	Subversion	SSH Slaves	Matrix Authorization Strategy
PAM Authentication	LDAP	Email Extension	✓ Mailer

```
** Pipeline: Job
** Pipeline Graph Analysis
** Pipeline: REST API
** JavaScript GUI Lib:
Handlebars bundle
** JavaScript GUI Lib:
Moment.js bundle
Pipeline: Stage View
** Pipeline: Build Step
** Pipeline: Model API
** Pipeline: Declarative
Extension Points API
** Apache HttpComponents
Client 4.x API
** JSch dependency
** Git client
** GIT server
** Pipeline: Shared Groovy
Libraries
** Display URL API
Mailer
** - required dependency
```

Jenkins 2.138.2


Getting Started

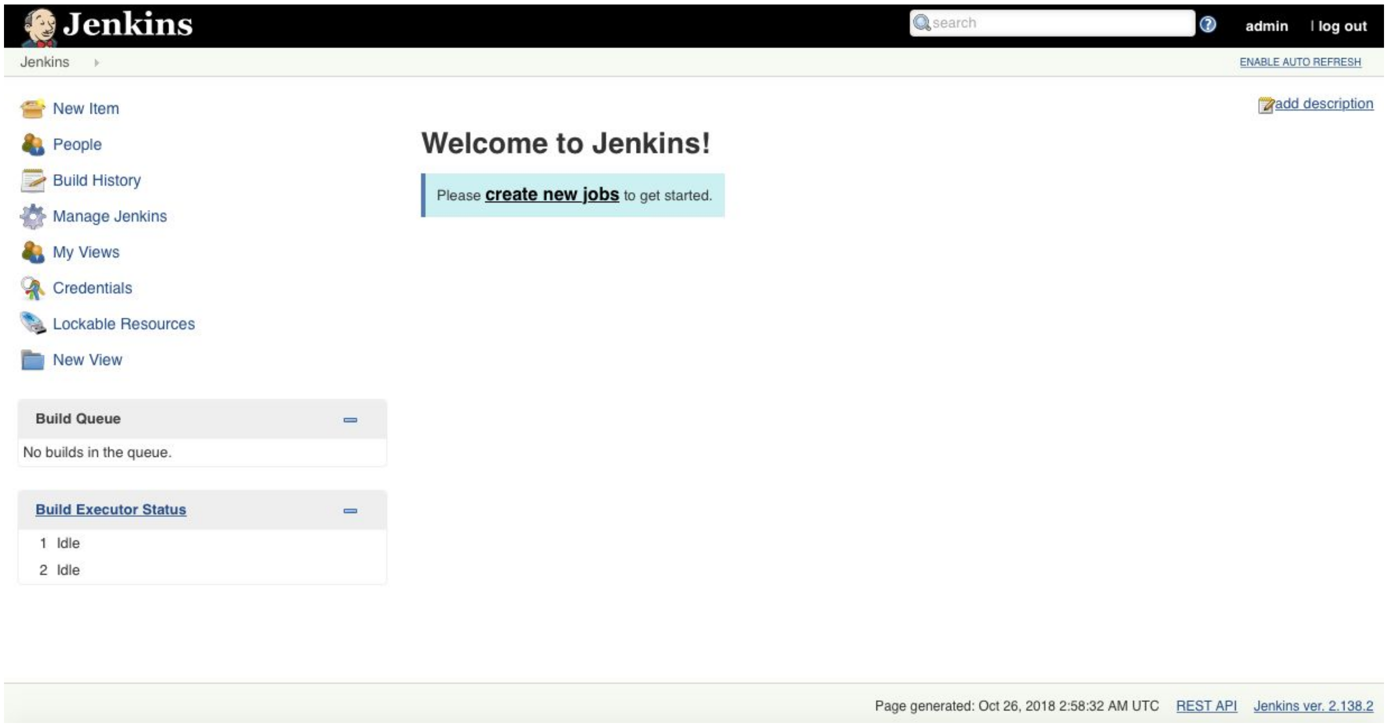
Instance Configuration

Jenkins URL: http://34.204.100.212:8080/

The Jenkins URL is used to provide the root URL for absolute links to various Jenkins resources. That means this value is required for proper operation of many Jenkins features including email notifications, PR status updates, and the `BUILD_URL` environment variable provided to build steps.

The proposed default value shown is **not saved yet** and is generated from the current request, if possible. The best practice is to set this value to the URL that users are expected to use. This will avoid confusion when sharing or viewing links.

Jenkins 2.138.2



Welcome to Jenkins!

Please **create new jobs** to get started.

Build Queue

No builds in the queue.

Build Executor Status

1 Idle

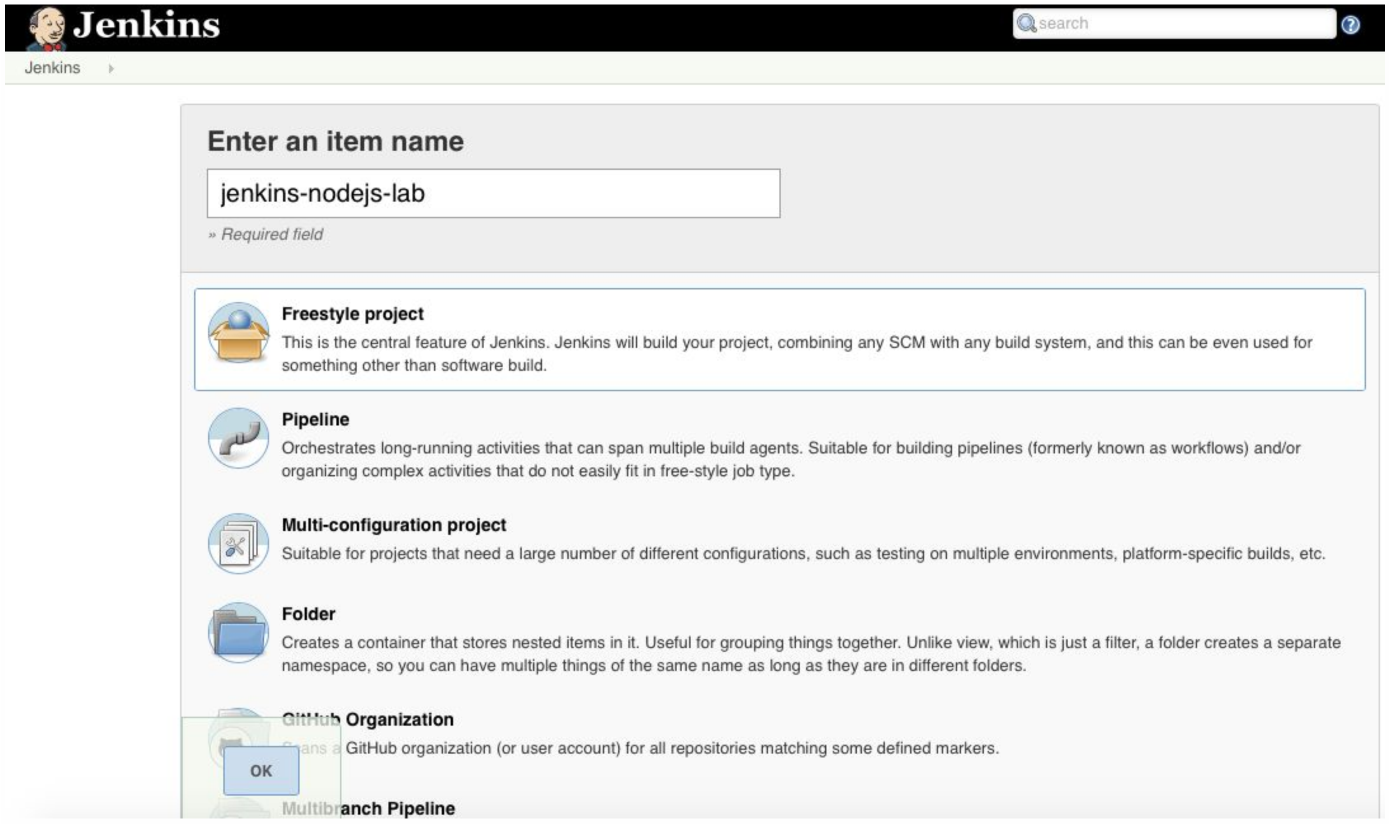
2 Idle

Page generated: Oct 26, 2018 2:58:32 AM UTC REST API Jenkins ver. 2.138.2

Click the **Manage Jenkins** link, and then the **Manage Plugins** link. Switch to the **Available** tab, and search for the GitHub Integration Click the **Install** checkbox, and then the **Download now and install after restart** button.

This will initiate the install sequence. The GitHub plugin has several dependencies, so multiple plugins will be installed. At the bottom of the page, check the **Restart Jenkins when installation is complete and no jobs are running** - this will prompt Jenkins to restart once the installations are complete.

Once Jenkins has restarted, it's time to add our project. Click the **New Item** button. Use "hello-jenkins" for the item name, select **Build a free-style software project**, and click the button labeled **OK**.


Jenkins

Jenkins

Enter an item name

jenkins-nodejs-lab

» 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.

Pipeline
Orchestrates long-running activities that can span multiple build agents. Suitable for building pipelines (formerly known as workflows) and/or organizing complex activities that do not easily fit in free-style job type.

Multi-configuration project
Suitable for projects that need a large number of different configurations, such as testing on multiple environments, platform-specific builds, etc.

Folder
Creates a container that stores nested items in it. Useful for grouping things together. Unlike view, which is just a filter, a folder creates a separate namespace, so you can have multiple things of the same name as long as they are in different folders.

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

OK

Multibranch Pipeline

Once the project is setup, you'll find yourself on the project's settings page. Add our project's GitHub URL to the **GitHub project** box:

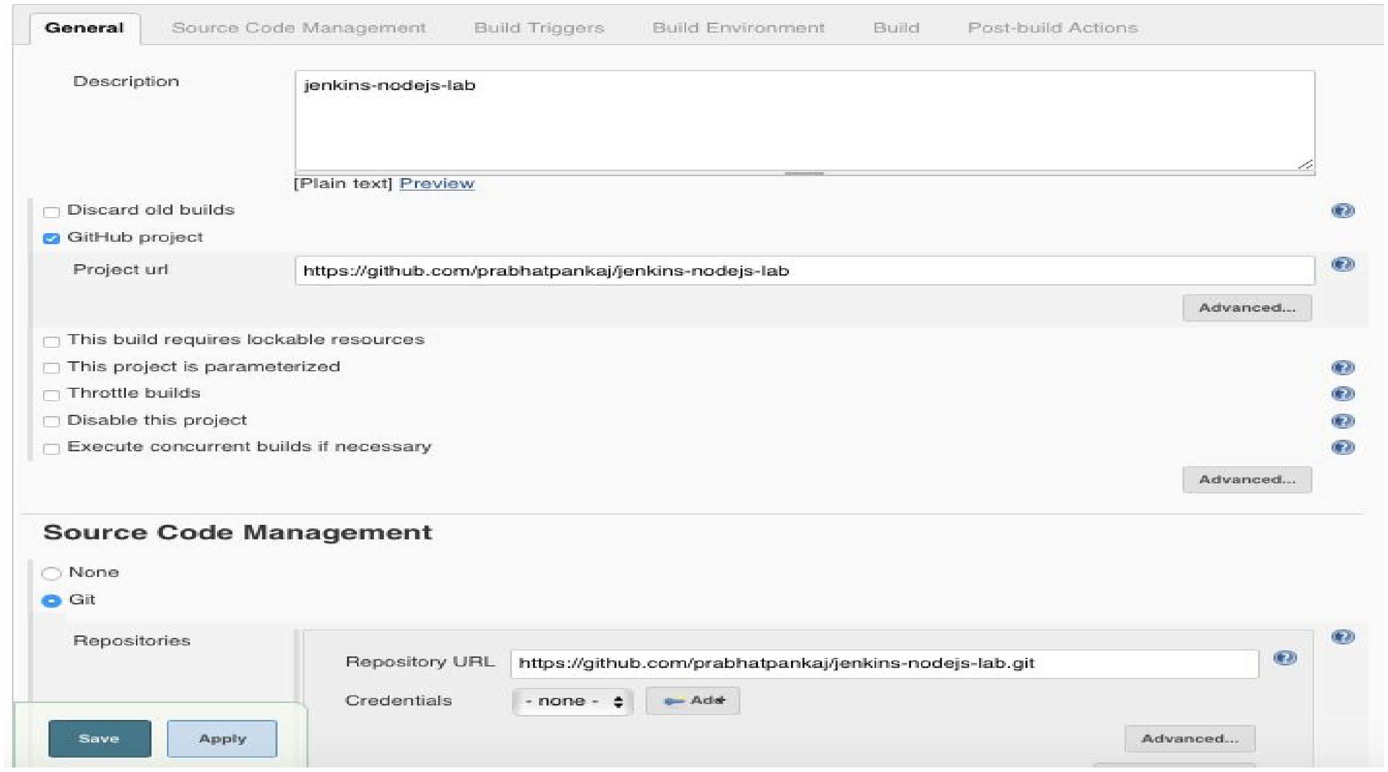
```
https://github.com/prabhatpankaj/jenkins-nodejs-lab
```

Next, select the **Git** option under **Source Code Management**. In the newly appeared fields, add the URL to our GitHub project repo to the **Repository URL**field:

```
https://github.com/prabhatpankaj/jenkins-nodejs-lab.git
```

Scroll a little further down and click the box to enable **GitHub hook trigger for GITScm polling.** With this option checked, our project will build every time we push to our GitHub repo. Of course, we need Jenkins to know what to do when it runs a build. Click the **Add build step** drop-down, and select **Execute shell**. This will make a **Command** dialogue available, and what we put in this dialogue will be run when a build initiates. Add the following to it:

```
npm install
```



Description: jenkins-nodejs-lab

[Plain text] Preview

☐ Discard old builds
☑ GitHub project

Project url: https://github.com/prabhatpankaj/jenkins-nodejs-lab

Advanced...

☐ This build requires lockable resources
☐ This project is parameterized
☐ Throttle builds
☐ Disable this project
☐ Execute concurrent builds if necessary

Advanced...

Source Code Management

○ None
● Git

Repositories

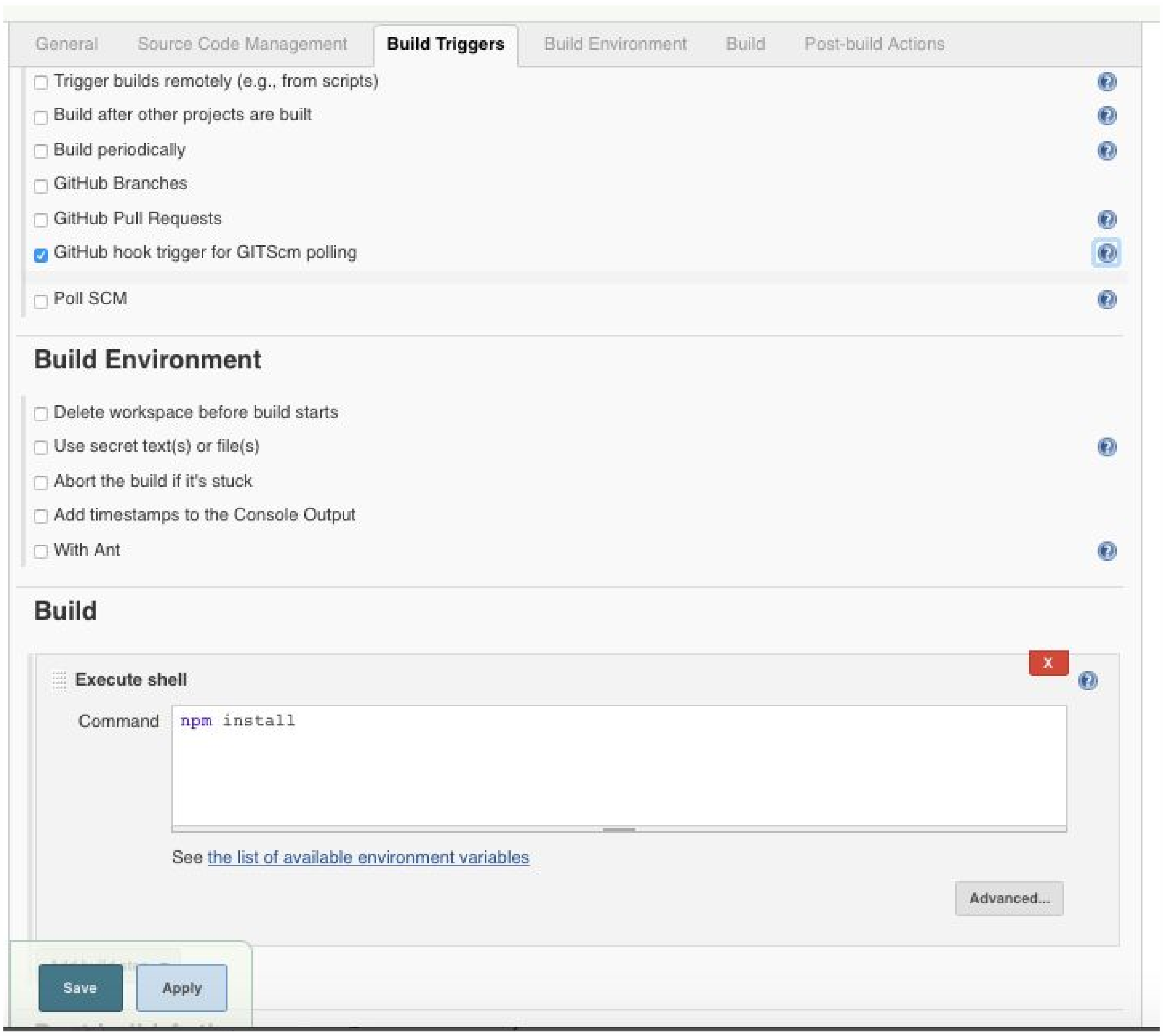
Repository URL: https://github.com/prabhatpankaj/jenkins-nodejs-lab.git

Credentials: - none -

Add

Advanced...

Save Apply



- [ ] Trigger builds remotely (e.g., from scripts)
- [ ] Build after other projects are built
- [ ] Build periodically
- [ ] GitHub Branches
- [ ] GitHub Pull Requests
- [x] GitHub hook trigger for GITScm polling
- [ ] Poll SCM

Build Environment

- [ ] Delete workspace before build starts
- [ ] Use secret text(s) or file(s)
- [ ] Abort the build if it's stuck
- [ ] Add timestamps to the Console Output
- [ ] With Ant

Build

Execute shell

Command

```
npm install
```

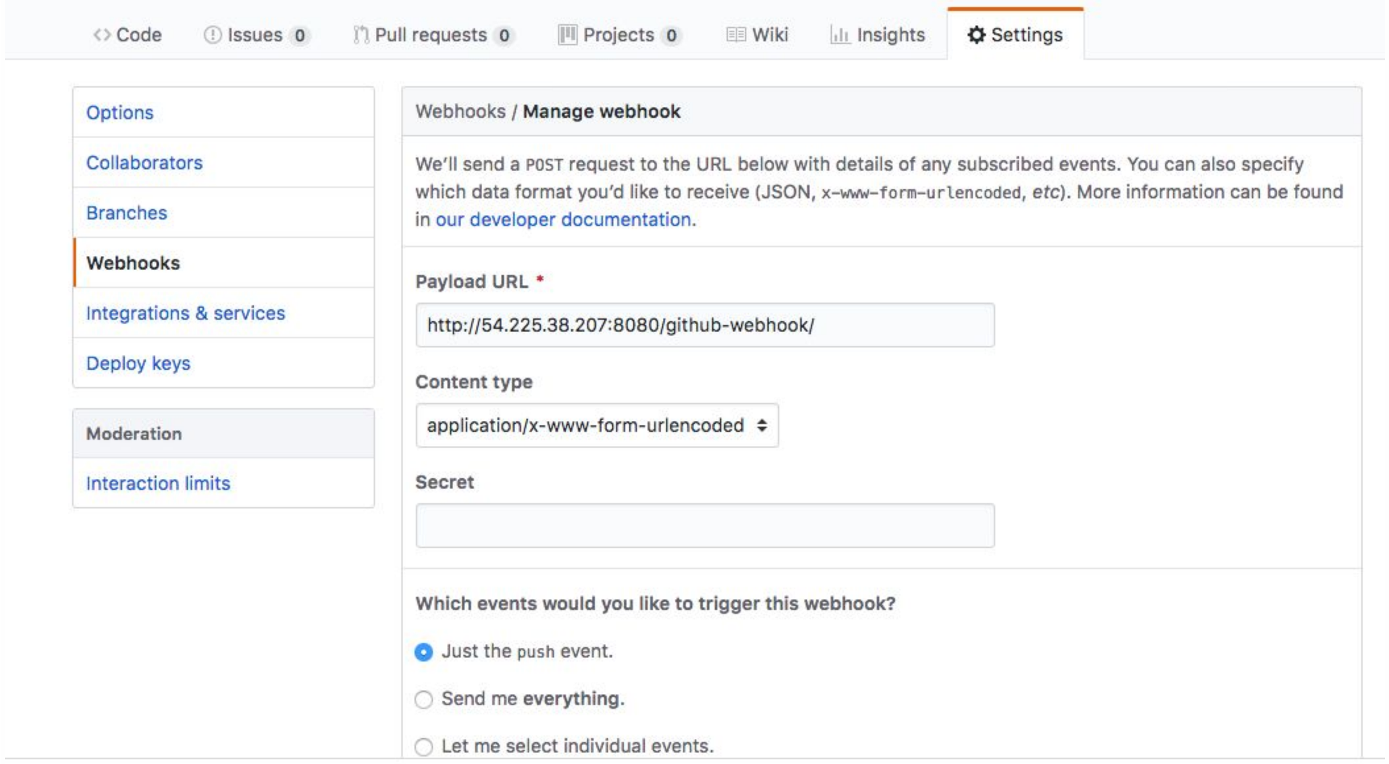
See the list of available environment variables

Click "**Save**".

To finish setting up the integration, head over to the GitHub repo, and click **Settings**. Click the **Webhooks** tab, and then the **Add webhooks** drop-down.

Add the following as the **Payload url**:

```
http://54.225.38.207:8080/github-webhook/
```



Webhooks / **Manage webhook**

We'll send a POST request to the URL below with details of any subscribed events. You can also specify which data format you'd like to receive (JSON, x-www-form-urlencoded, *etc*). More information can be found in our developer documentation.

Payload URL *: http://54.225.38.207:8080/github-webhook/

Content type: application/x-www-form-urlencoded

Secret:

Which events would you like to trigger this webhook?

[✓ Just the push event.]

Send me everything.

Let me select individual events.

Watch Jenkins - once again, you'll see a build is automatically started, and this time, it succeeds!

This is the flow of continuous integration. The test server is continually testing any new code you push so you are quickly informed of any failing tests.

Get It Deployed

The Key to Authentication

When Jenkins installs, it creates a new user called `jenkins`. Jenkins executes all commands with this user, so we need to generate our key with the `jenkins` user so that it has the appropriate access to it.
While logged in as `admin` on the `jenkins-server`, execute the following:

```
sudo su
```

Provide your `admin` password, and it'll switch you to the `root` user. Then execute:

```
su jenkins
```

Now you are acting as the `jenkins` user. Generate an SSH key:

```
ssh-keygen -t rsa
```

Save the file in the default location (`/var/lib/jenkins/.ssh/id_rsa`), *and make sure to not use a passphrase* (otherwise SSH access will require a password and won't work when automated).

Next, we need to copy the public key that was created. Run this:

```
cat ~/.ssh/id_rsa.pub

ssh-rsa
AAAAB3NzaC1yc2EAAAADAQABAAABAQDBmmhndwBUsIoXzxK2j1cNu5E
K02xnewRiDAfkizW/+o1nrsKVojTGxNRHEvhP7P94xwDBfpNIw6W4m0
WawX+V9z4uzuQ1nbgq1mWLUsKvltwrVvDgMjQj3HI4pKrcH5SvRP0Xs
```

```
BhZPNbDTDfaSPcw5b79qWxN/FgcoDvOFFSzmvw/ufabNdD4auQECtqb
4CcWrlVQH54iouGp6ZBlCrLNFjEw5LQnoUgz8WLuvlsskvmqVu9+CiU
CW9BK7WkIS6K97c4HNjgnjWxy+XFgTW6e/ldt5Hb7bMqCAzvx5ZhJEv
m92JkOa0BERdYThZ+wqrrTX8SbmJOrQoRK8dL8rA57
jenkins@ip-172-31-38-27
```

log back into our app server (jenkins-nodejs-lab) as the `app` user. We need to create a file named `authorized_keys` in our `app` user's`.ssh` folder:

```
mkdir ~/.ssh

nano ~/.ssh/authorized_keys

ssh-rsa
AAAAB3NzaC1yc2EAAAADAQABAAABAQDBmmhndwBUsIoXzxK2j1cNu5EK0
2xnewRiDAfkizW/+o1nrsKVojTGxNRHEvhP7P94xwDBfpNIw6W4m0WawX
+V9z4uzuQ1nbgq1mWLUsKvltwrVvDgMjQj3HI4pKrcH5SvRP0XsBhZPNb
DTDfaSPcw5b79qWxN/FgcoDvOFFSzmvw/ufabNdD4auQECtqb4CcWrlVQ
H54iouGp6ZBlCrLNFjEw5LQnoUgz8WLuvlsskvmqVu9+CiUCW9BK7WkIS
6K97c4HNjgnjWxy+XFgTW6e/ldt5Hb7bMqCAzvx5ZhJEvm92JkOa0BERd
YThZ+wqrrTX8SbmJOrQoRK8dL8rA57 jenkins@ip-172-31-38-27
```

In order for this file to properly work, it needs to have strict permissions set on it:

```
chmod 700 ~/.ssh

chmod 600 ~/.ssh/*
```

Head back to the `jenkins` server, switch to the `jenkins` user, and verify that you can login to our app server without entering a password:

```
    ssh app@APP.SERVER.IP.ADDRESS
```

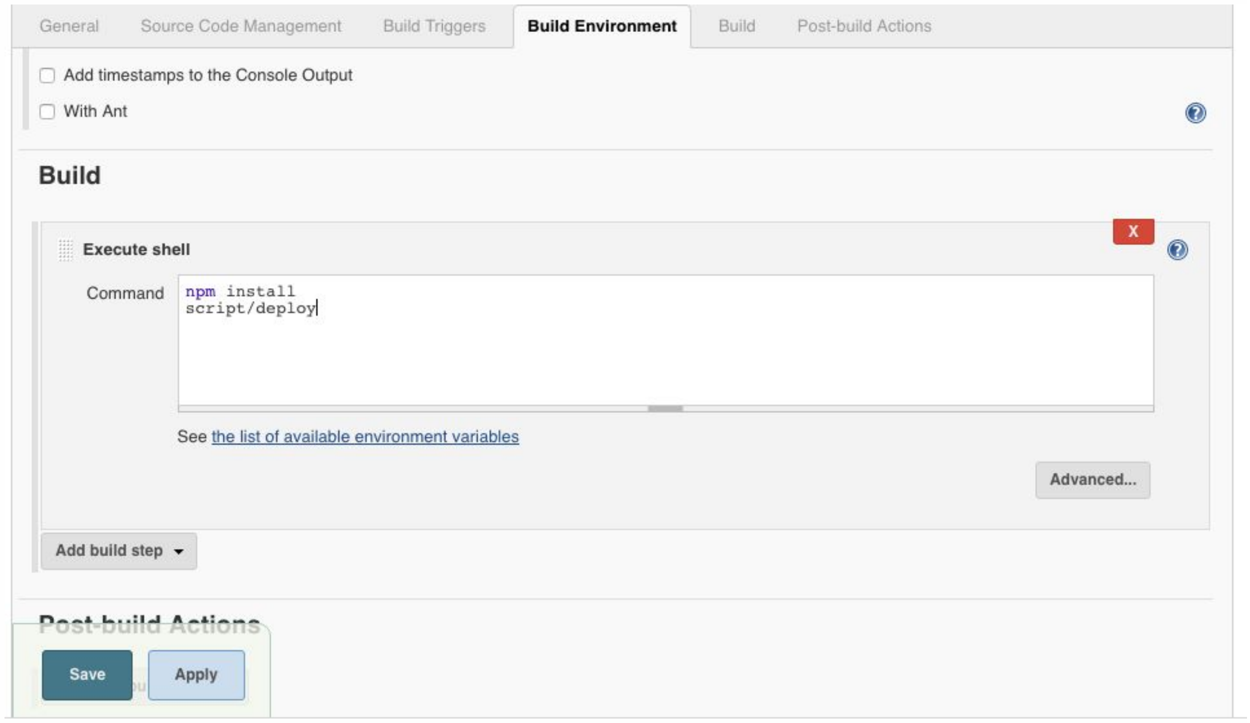
You should successfully login to the app server without having to enter the password. With that established.

Ship It Automatically

Update jenkins build trigger

```
npm install

script/deploy
```



```
npm install
script/deploy
```

See the list of available environment variables script file executable: # update ip address in script/deploy from your local computer

```
cd jenkins-nodejs-lab

chmod +x script/deploy

git commit -m ‘changed’

git push origin master
```