Table of Contents

Install Jenkins on Ubuntu 18.04:

https://linuxize.com/post/how-to-install-jenkins-on-ubuntu-18-04/
https://www.serverlab.ca/tutorials/linux/administration-linux/how-to-install-jenkins-on-ubuntu-18-04-bionic-beaver/

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 updatesudo apt install openjdk-8-jdk

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 -

sudo apt-add-repository "deb https://pkg.jenkins.io/debian-stable binary/"
sudo apt-add-repository "deb http://pkg.jenkins-ci.org/debian binary/"

Install and Configure Jenkins

With Java 8 installed and the Jenkins repository added, we can now install and configure Jenkins on our Ubuntu server.

Install Jenkins.

sudo apt install jenkins

Starting Jenkins

Let’s start Jenkins using systemctl:

sudo systemctl start jenkins

avatti@office.zone@qindl124:~/projects/jadak/Piccolo$ systemctl status jenkins

- jenkins.service - LSB: Start Jenkins at boot time

  Loaded: loaded (/etc/init.d/jenkins; generated)

  Active: active (exited) since Mon 2020-04-27 10:06:03 IST; 2min 45s ago

  Docs: man:systemd-sysv-generator(8)

  Tasks: 0 (limit: 4915)

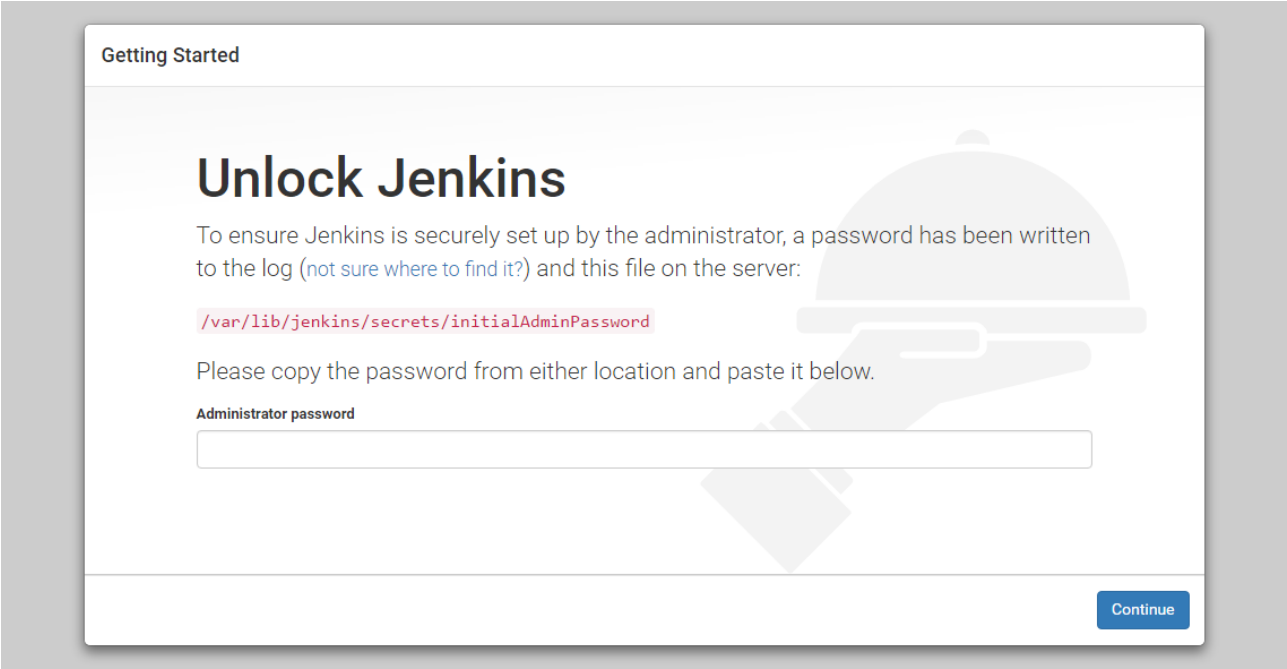
CGroup: /system.slice/jenkins.service

avatti@office.zone@qindl124:~/projects/jadak/Piccolo$ sudo cat /var/lib/jenkins/secrets/initialAdminPassword

481c0e3fde3d48af85748c2c44f5a334

Setting Up Jenkins

To set up your installation, visit Jenkins on its default port, `8080`, using your server domain name or IP address: `http://:8080`



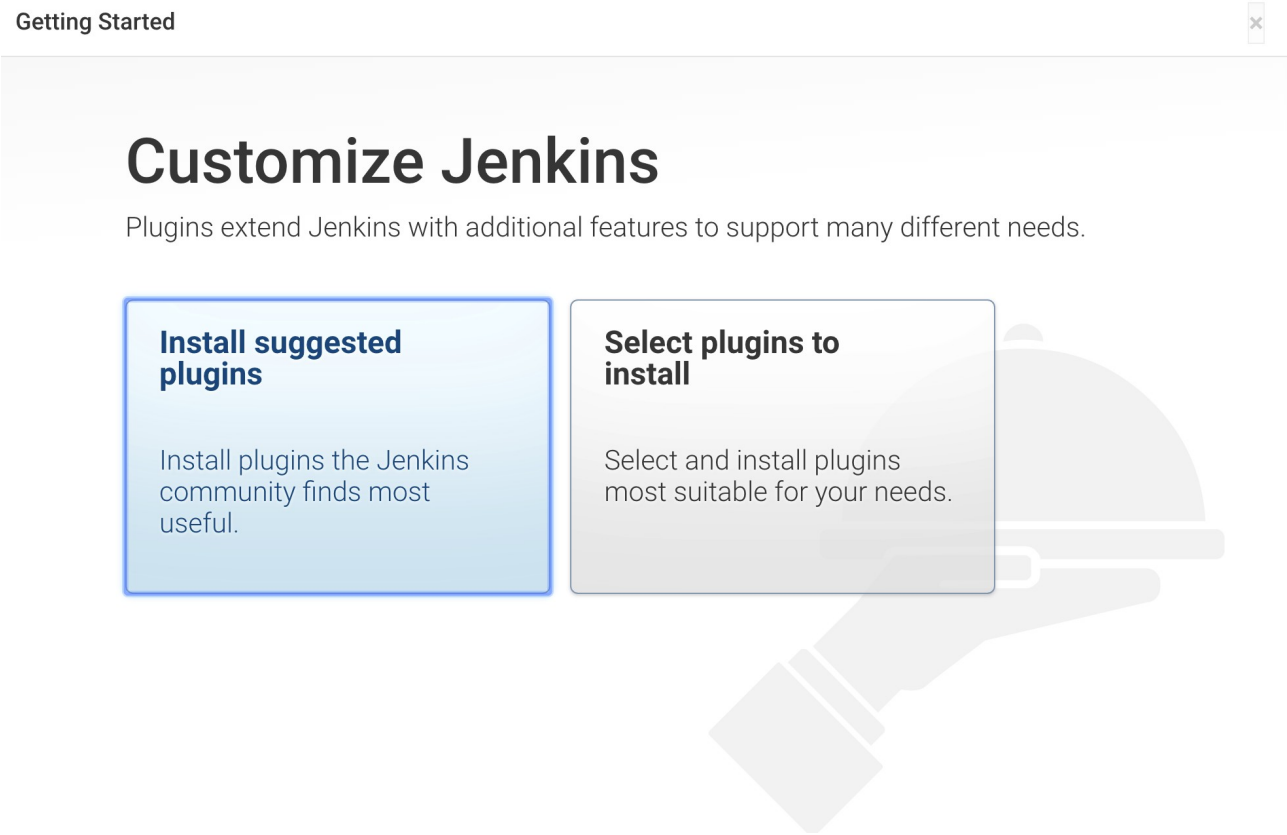
In the terminal window, use the cat command to display the password:

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

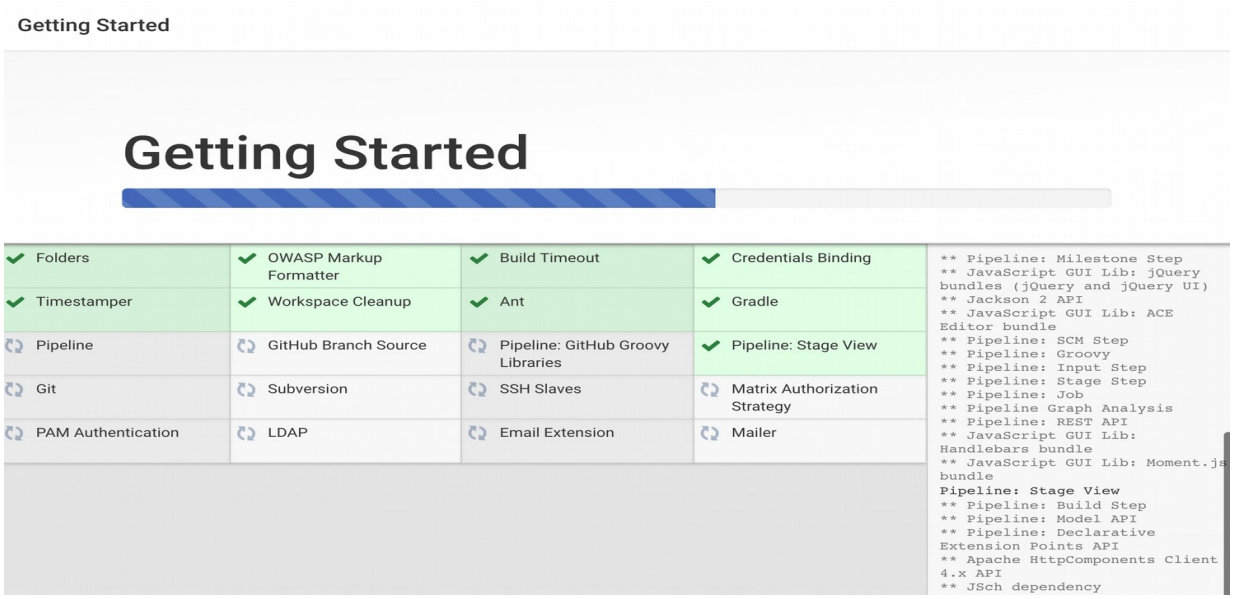
Copy the 32-character alphanumeric password from the terminal and paste it into the **Administrator password** field, then click **Continue**.

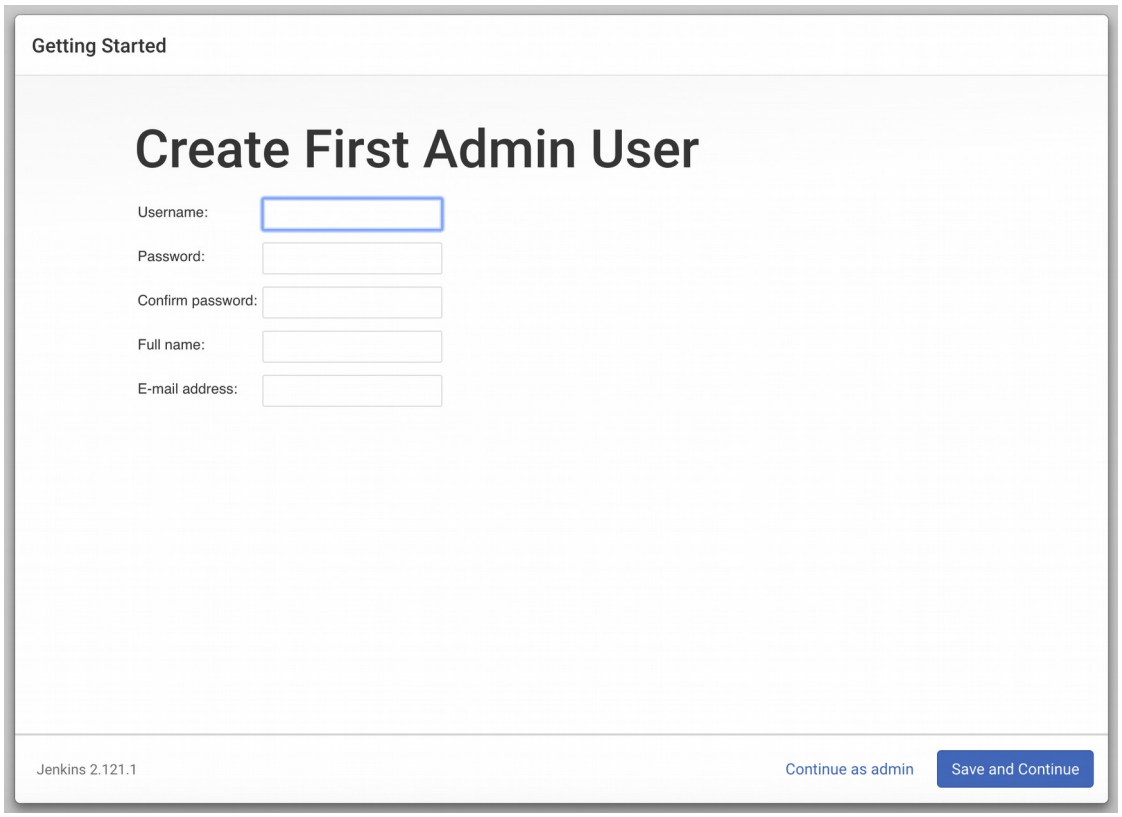
Installing suggested plugins

The next screen presents the option of installing suggested plugins or selecting specific plugins:

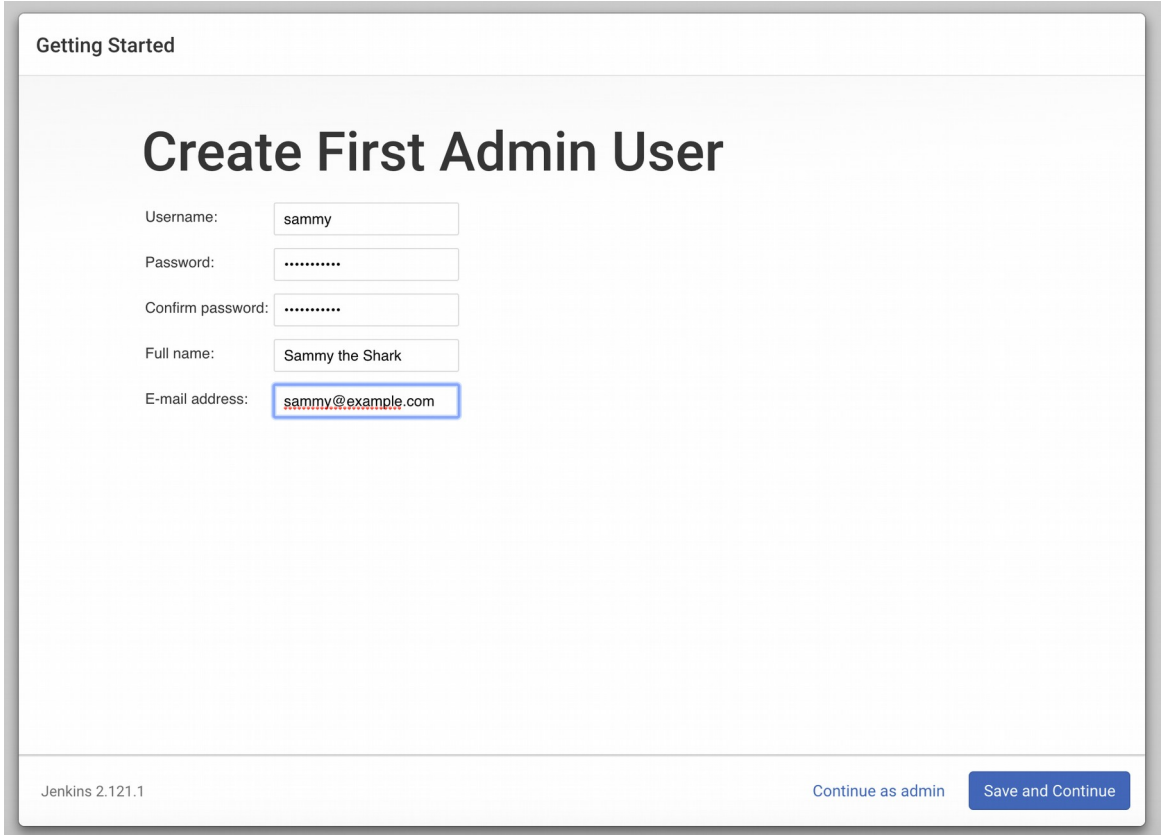


We'll click the **Install suggested plugins** option, which will immediately begin the installation process:

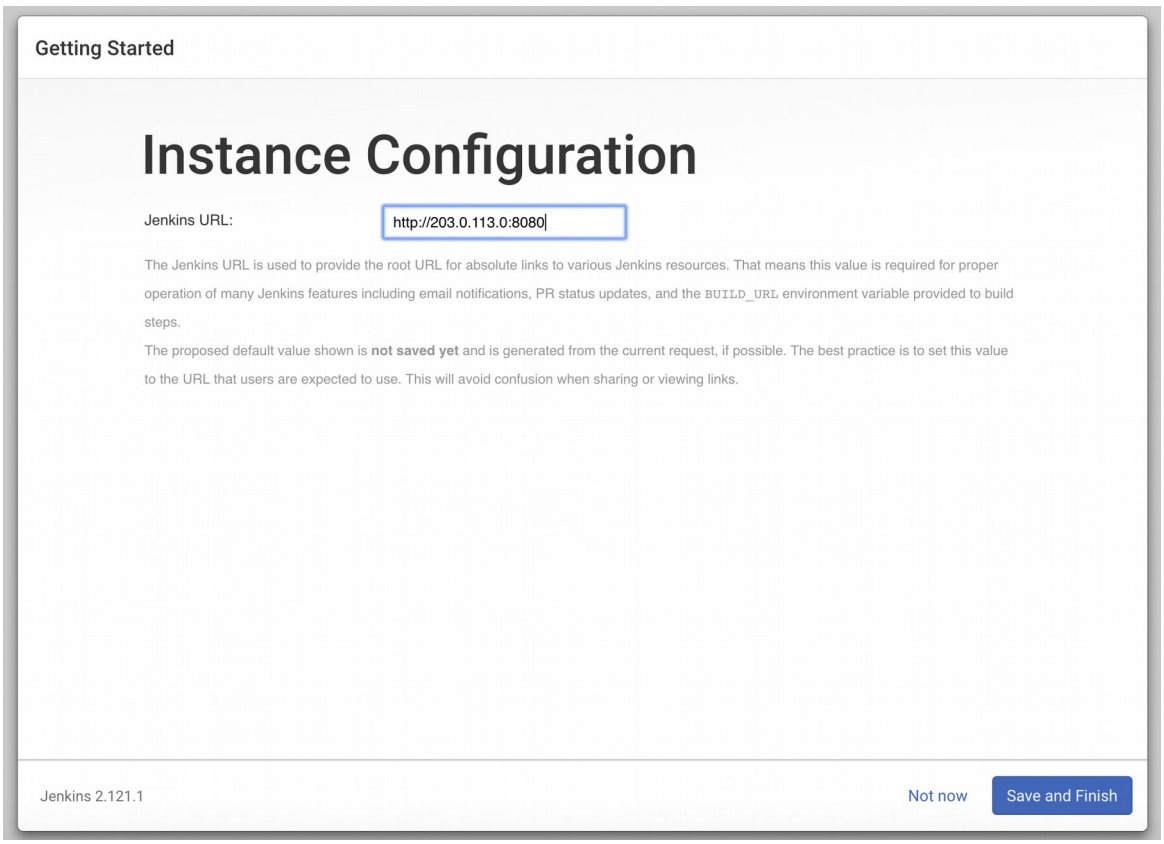




Enter the name and password for your user:



You will see an **Instance Configuration** page that will ask you to confirm the preferred URL for your Jenkins instance. Confirm either the domain name for your server or your server's IP address:



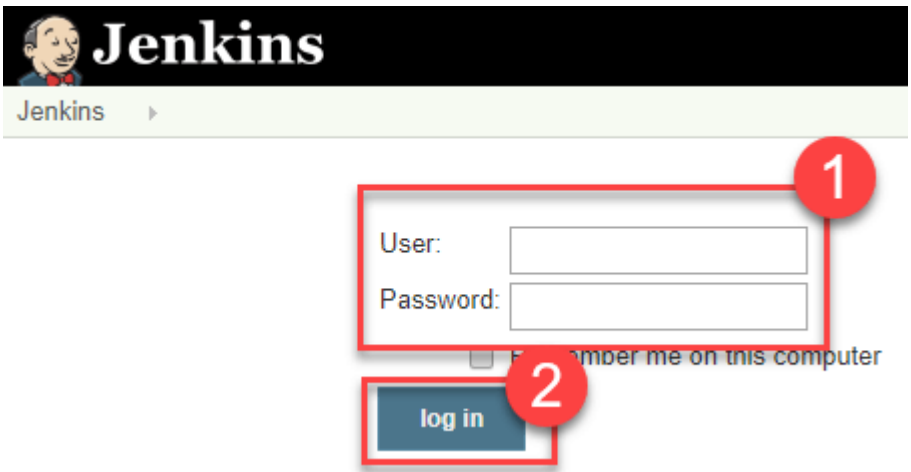
Open the Jenkins:

If you are connected to the vpn can you access this site?

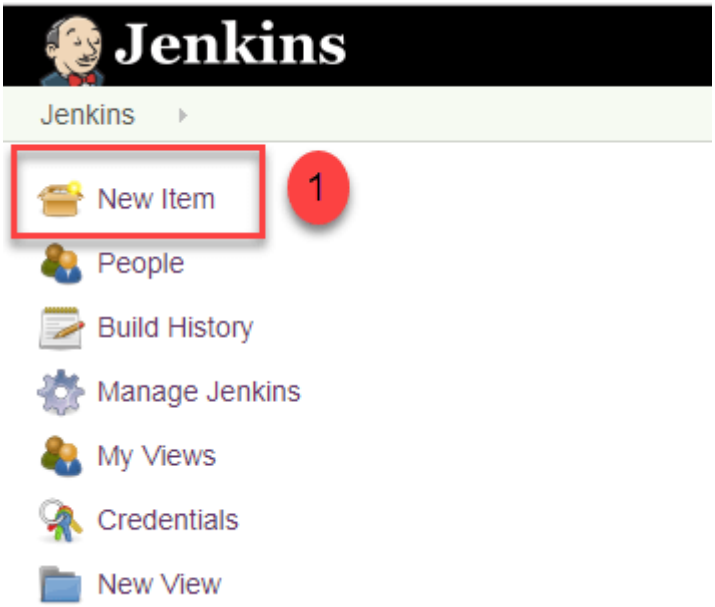
http://192.168.0.53:8080

Creating a Freestyle Build Job : Java:

Step 1) Enter the User credentials

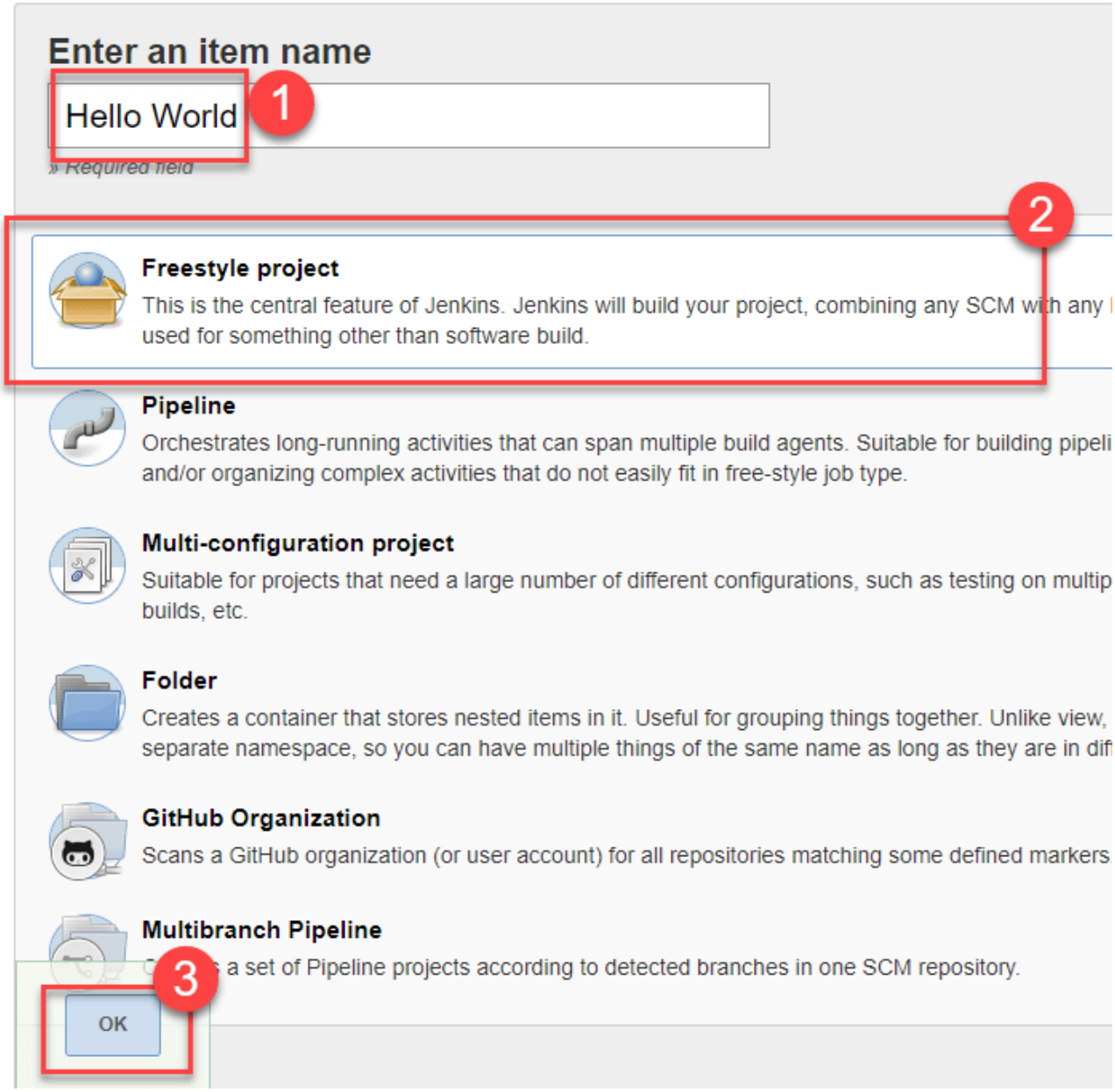


Step 2) Click on "**New Item**" at the top left-hand side of your dashboard.

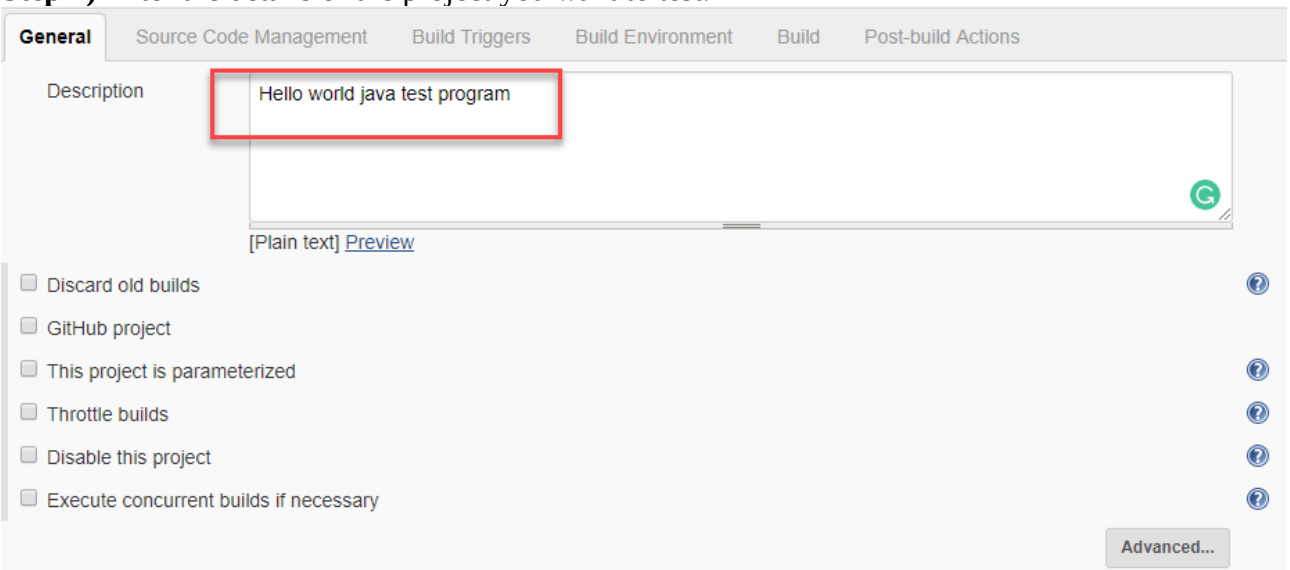


Step 3) In the next screen,

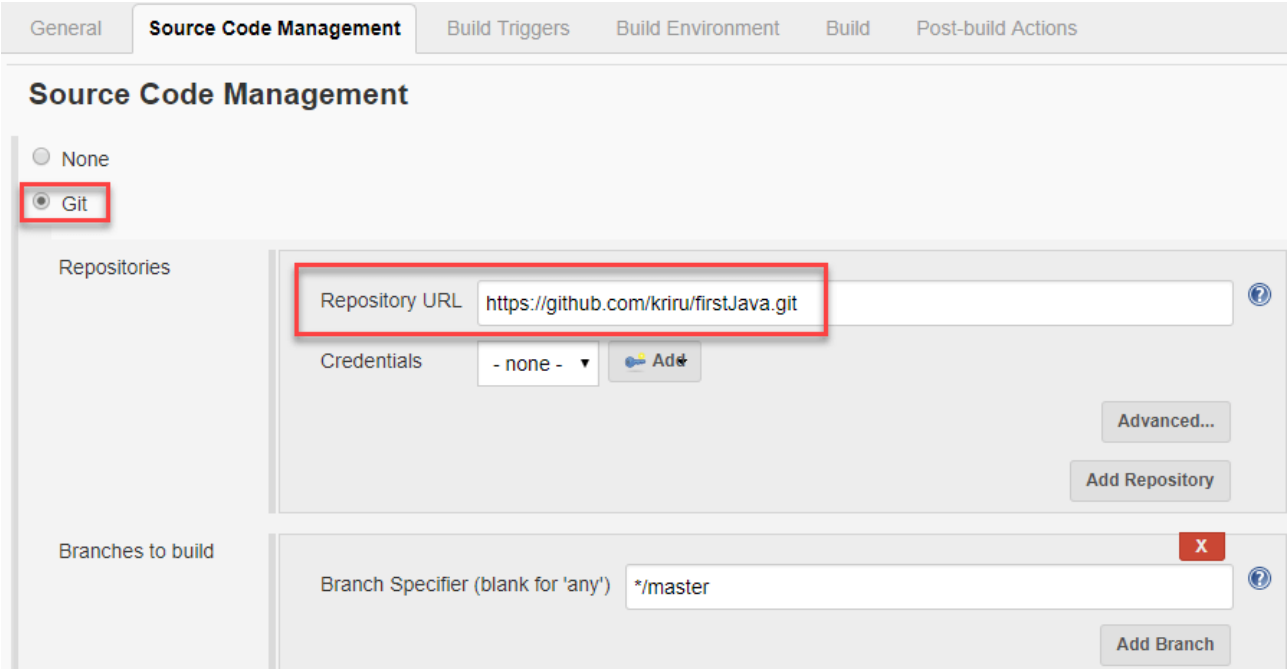
1. Enter the name of the item you want to create. We shall use the "Hello world" for this demo.
2. Select Freestyle project
3. Click Okay



Step 4) Enter the details of the project you want to test.



Step 5) Under Source Code Management, Enter your repository URL. We have a test repository located at https://github.com/kriru/firstJava.git



It is also possible for you to use a local repository.

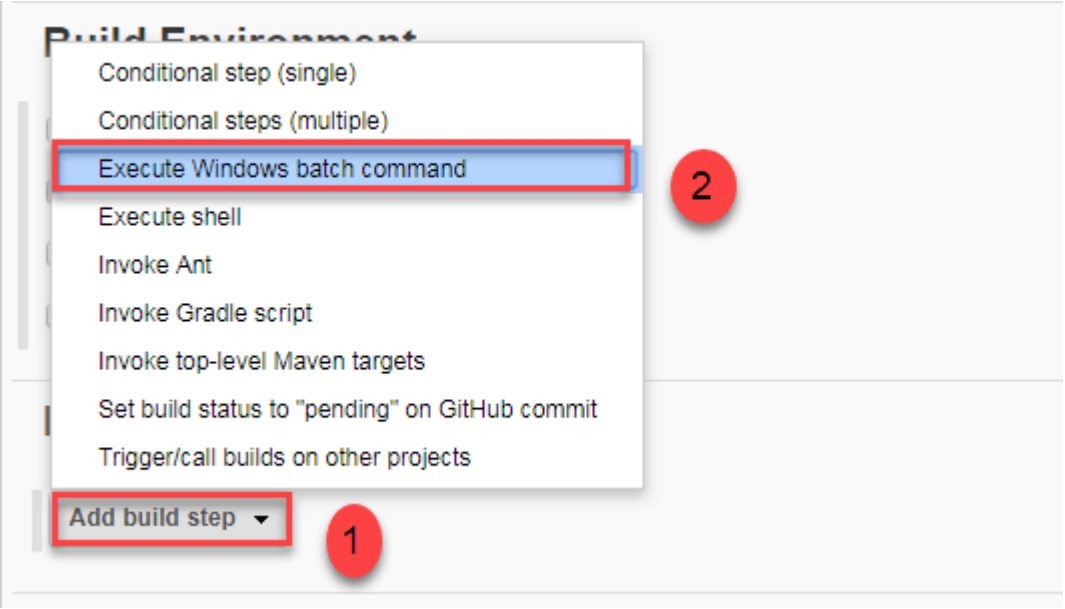
If your GitHub repository is private, Jenkins will first validate your login credentials with GitHub and only then pull the source code from your GitHub repository.

Step 6) 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**,

1. Click on "**Add build step**"

2. Click on "**Execute Windows batch command**" and add the commands you want to execute during the build process.

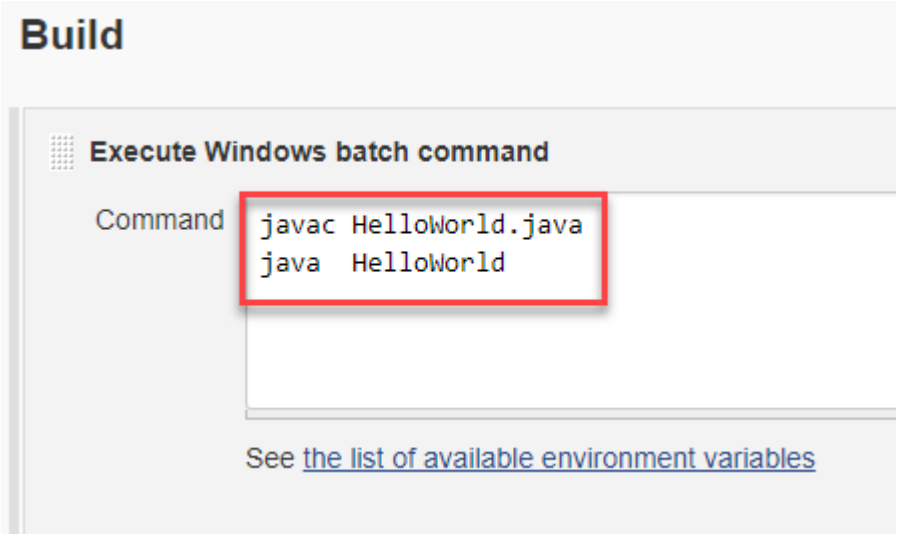


Here, I have added the java commands to compile the java code.

I have added the following windows commands:

javac HelloWorld.java

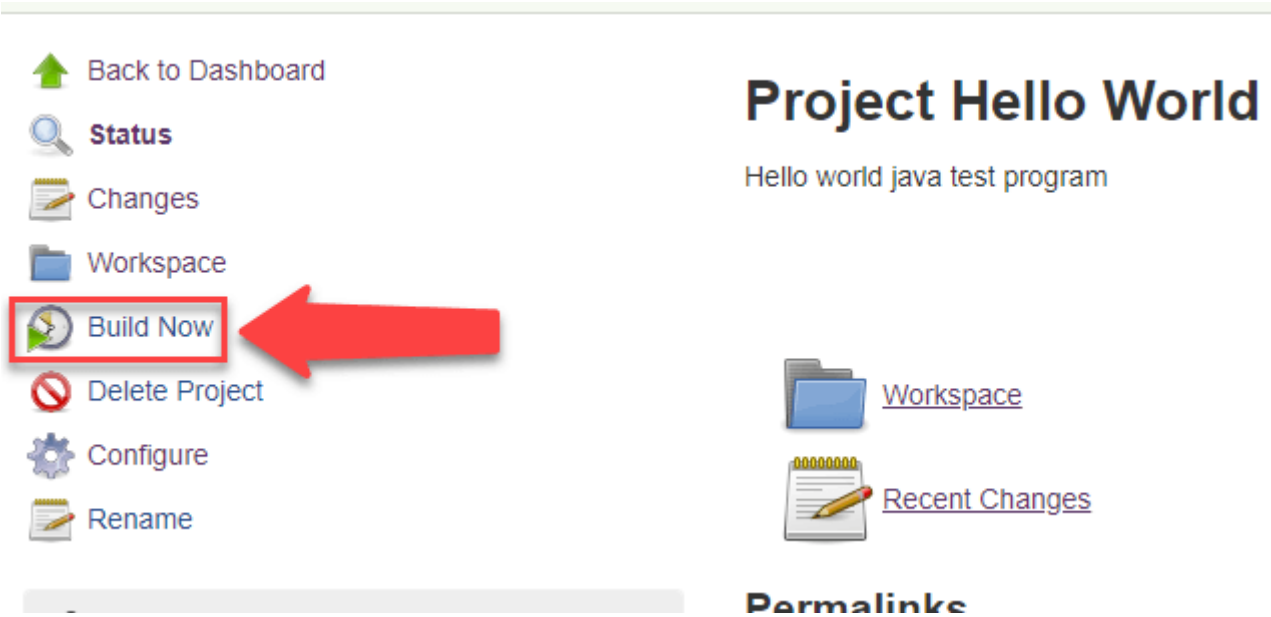
java HelloWorld



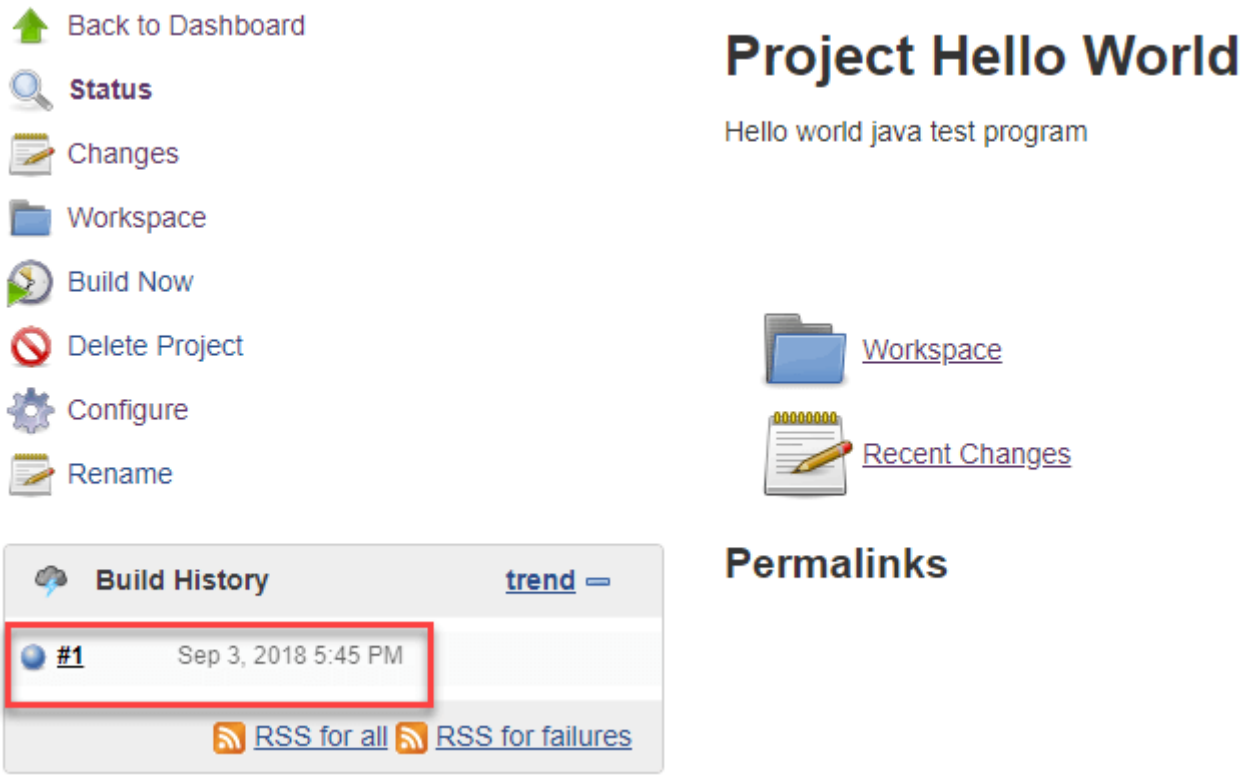
Step 7) When you have entered all the data,

1. Click **Apply**
2. **Save** the project.

Step 8) Now, in the main screen, Click the **Build Now** button on the left-hand side to build the source code.



Step 9) After clicking on **Build now,** you can see the status of the build you run under **Build History**.



Step 10) 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.

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.

Creating a Freestyle Build Job : cpp:

Source Code Management :

https://github.com/jbankes/Hello_Jenkins.git

Execute shell commnad on ubuntu

cd original

make

hello_exec

./hello_exec

Creating a Freestyle Build Job : qt:

Source Code Management :

https://github.com/alexvatti/qt-example.git

Execute shell commnad on ubuntu

qmake hello.pro

make

./hello

Configuring the Node in Jenkins

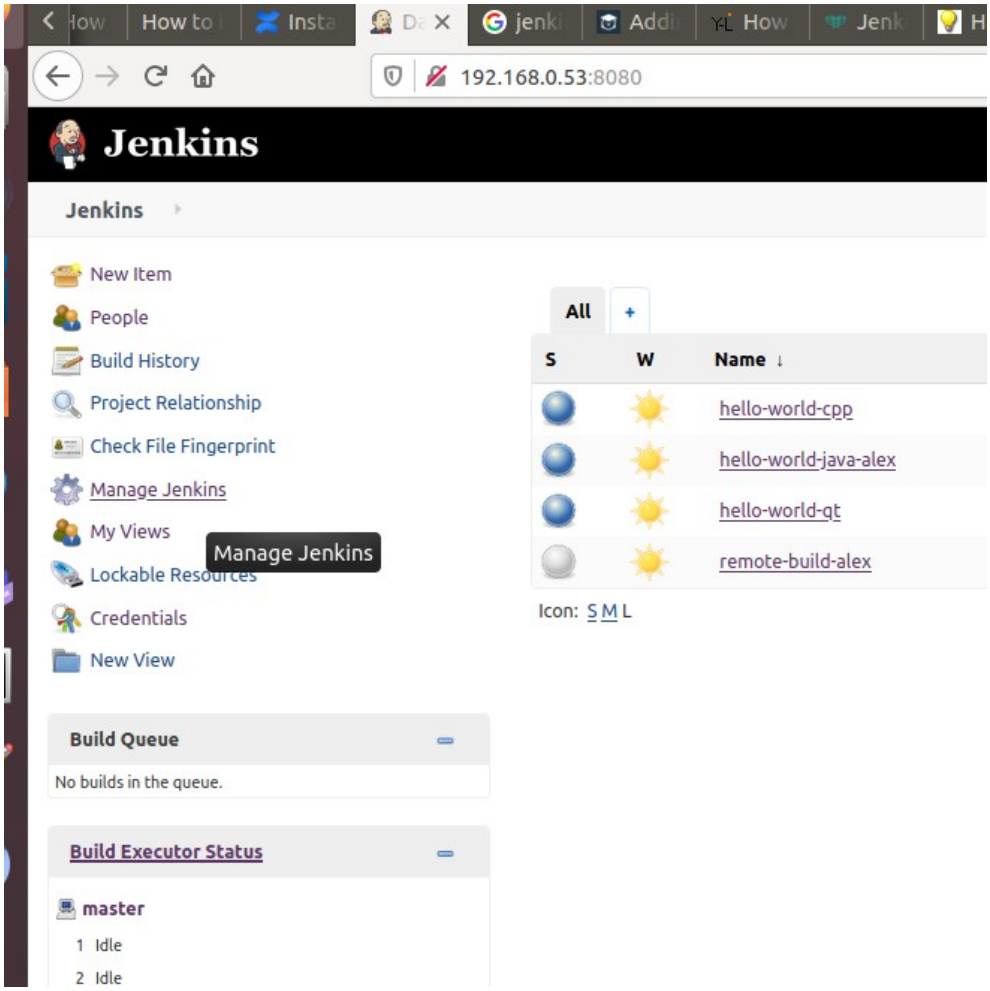
http://yallalabs.com/devops/how-to-add-linux-slave-node-agent-node-jenkins/

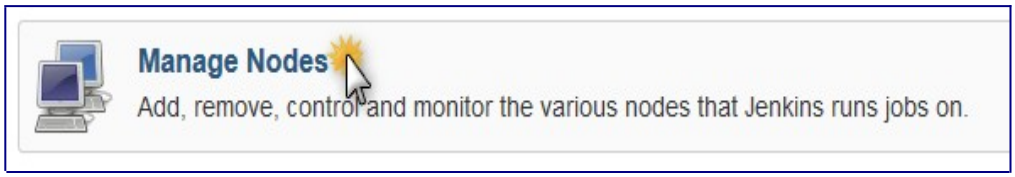
https://linuxacademy.com/blog/linux-academy/adding-a-jenkins-agent-node/

https://embeddedartistry.com/blog/2018/01/11/jenkins-configuring-a-linux-slave-node/

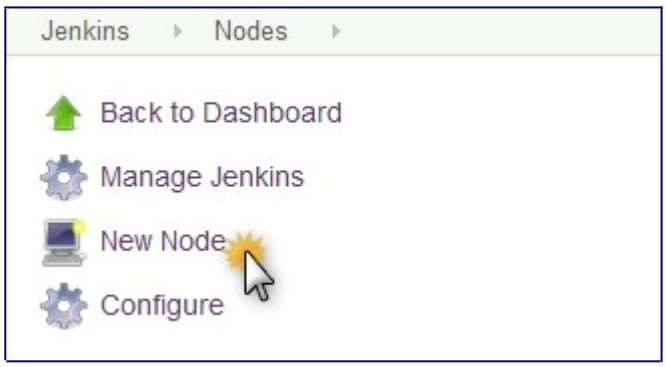
To configure a new node, navigate to “Manage Jenkins” in the classic Jenkins interface or “Administration” in Blue Ocean.

Go to **`Manage Jenkins`** -> **`Manage Nodes`**

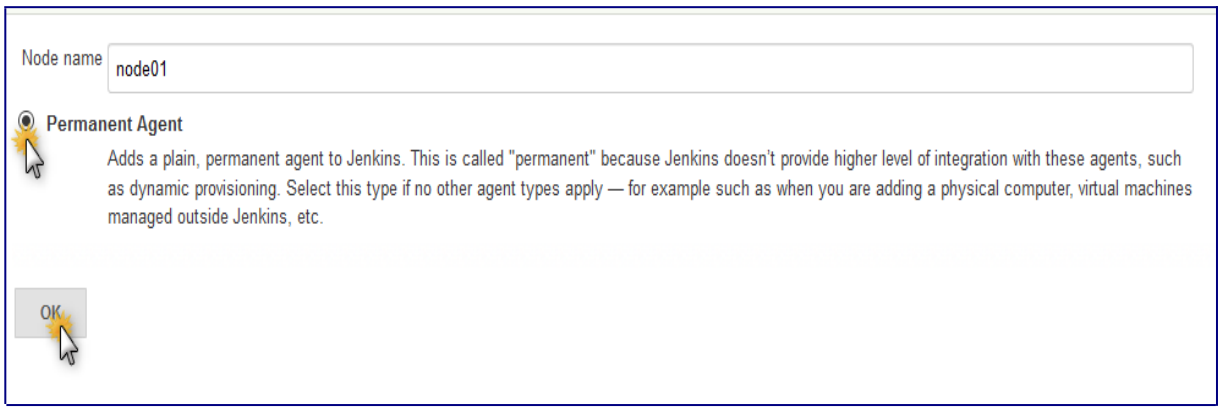




Then click on the **`New Node`** button:

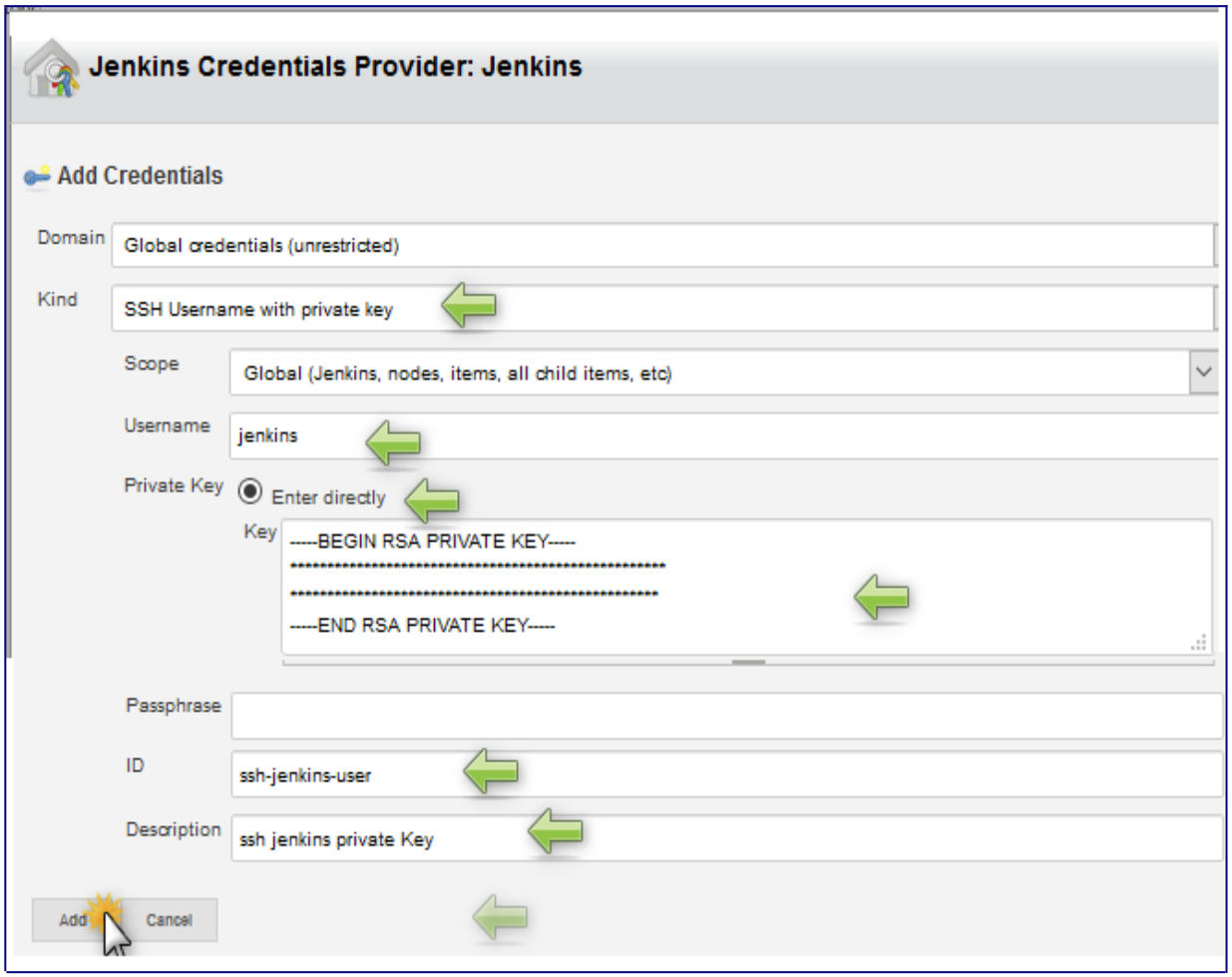


Configure the **name of the agent**, select `Permanent agent` and click on the `Ok` button:



After creating the new node, you have to configure the node settings. Fill in the `Remote root directory` with a path the user on the agent is allowed to write to, set the `Host` value with the hostname of the agent, and press the `Add` button for `Credentials`:

Choose SSH Username with private key option, fill the Username value with the user account on the agent machine, in our example is jenkins, and choose Private Key -> Enter directly and paste the key from your OS clipboard, and give an ID and a useful Description for this credential. Finally click the add button.

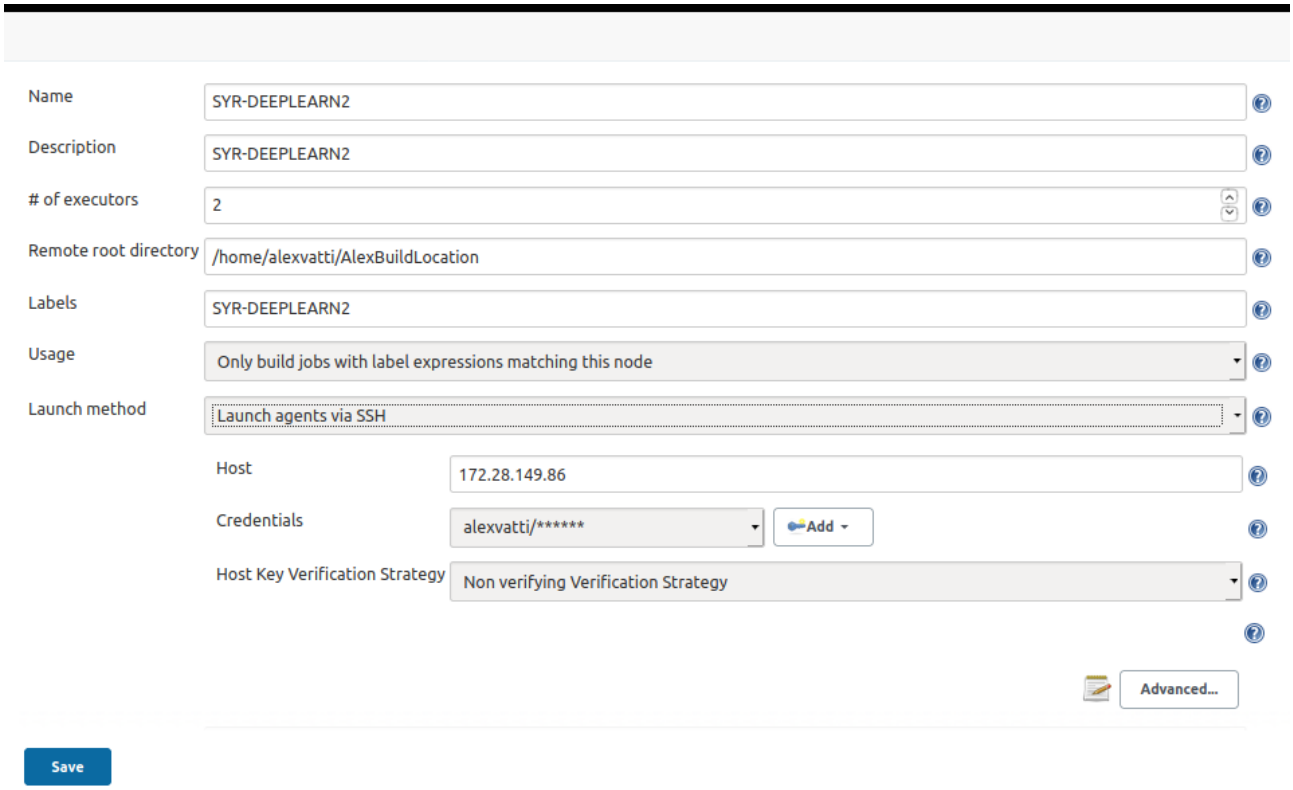


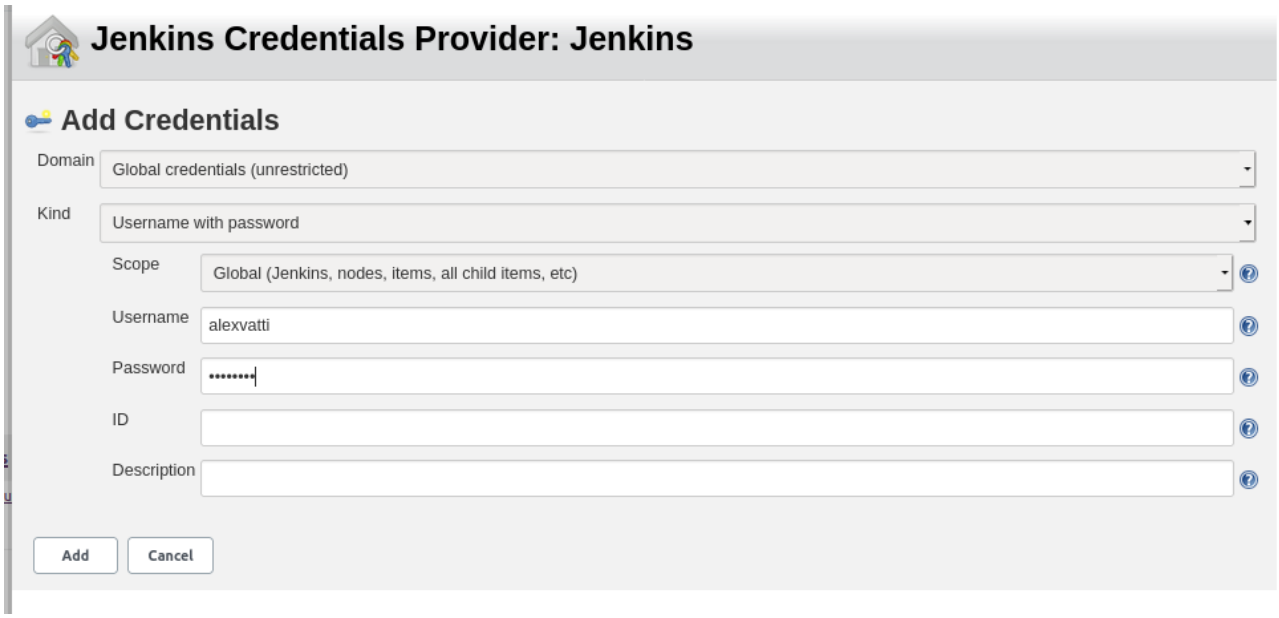
Select the `Manually trusted key Verification Strategy` value of the `Host Key Verification Strategy` menu and click `save` button.

Your new node should now appear in the list of nodes. You may notice a red X on the node's icon. This indicates that it is not connected yet. Wait a few seconds and refresh the page, and the red X will go away, indicating that the node is successfully connected.

S	Name ↓	Architecture	Clock Difference
	master	Linux (amd64)	In sync
	node01	Linux (amd64)	In sync

In our case:


Name
SYR-DEEPLEARN2
Description
SYR-DEEPLEARN2
# of executors
2
Remote root directory
/home/alexvatti/AlexBuildLocation
Labels
SYR-DEEPLEARN2
Usage
Only build jobs with label expressions matching this node
Launch method
Launch agents via SSH
Host
172.28.149.86
Credentials
alexvatti/******
Add
Host Key Verification Strategy
Non verifying Verification Strategy
Advanced...
Save


Jenkins Credentials Provider: Jenkins
Add Credentials
Domain
Global credentials (unrestricted)
Kind
Username with password
Scope
Global (Jenkins, nodes, items, all child items, etc)
Username
alexvatti
Password
ID
Description
Add
Cancel

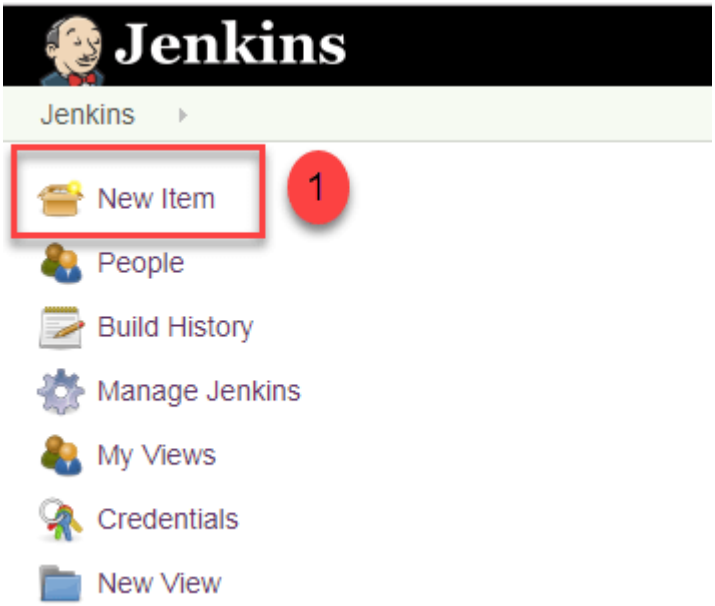
S	Name ↓	Architecture	Clock Difference	Free Disk Space	Free Swap Space	Free Temp Space	Response Time
	master	Linux (amd64)	In sync	201.90 GB	976.00 MB	201.90 GB	0ms
	SYR-DEEPLEARN2	Linux (amd64)	In sync	117.07 GB	15.59 GB	53.08 GB	3036ms
	Data obtained	6 min 26 sec	6 min 26 sec	6 min 19 sec	6 min 22 sec	6 min 19 sec	6 min 25 sec

Refresh status

Connection the Node in Jenkins:

Creating a Freestyle Build Job : libPiccoloCam : Run on Remote

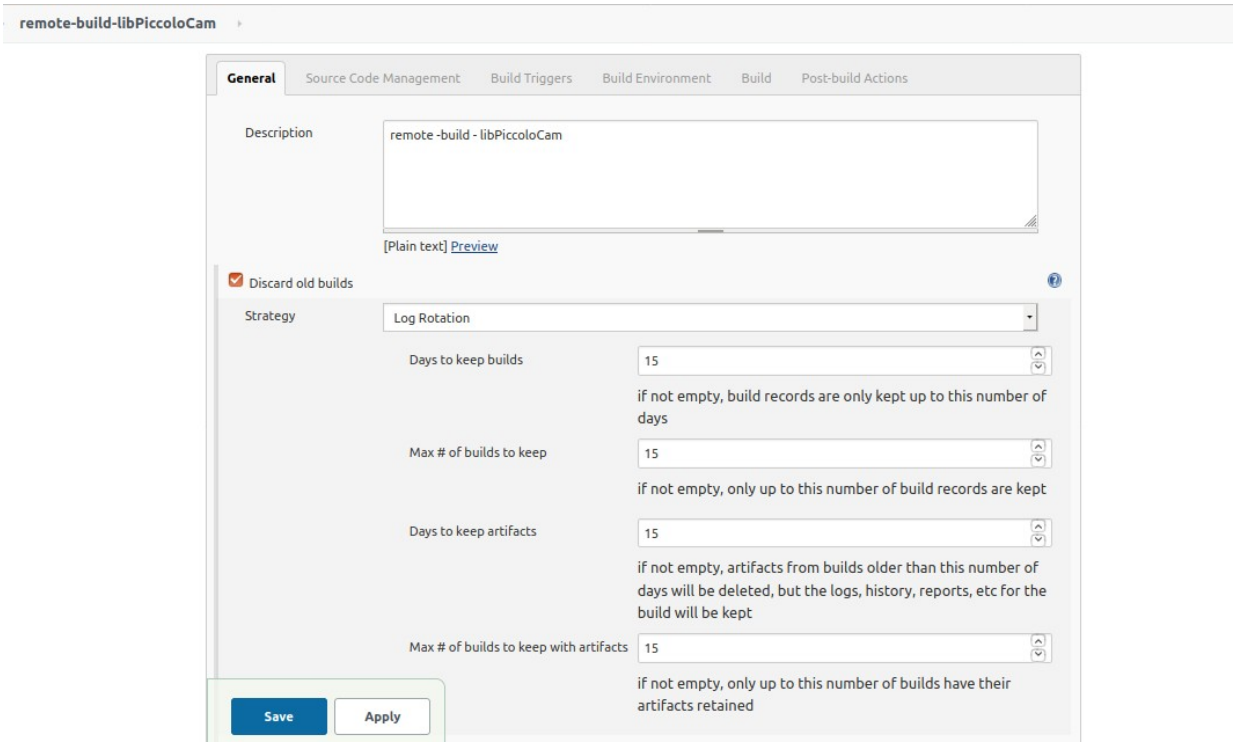
Step 1) Click on "**New Item**" at the top left-hand side of your dashboard.

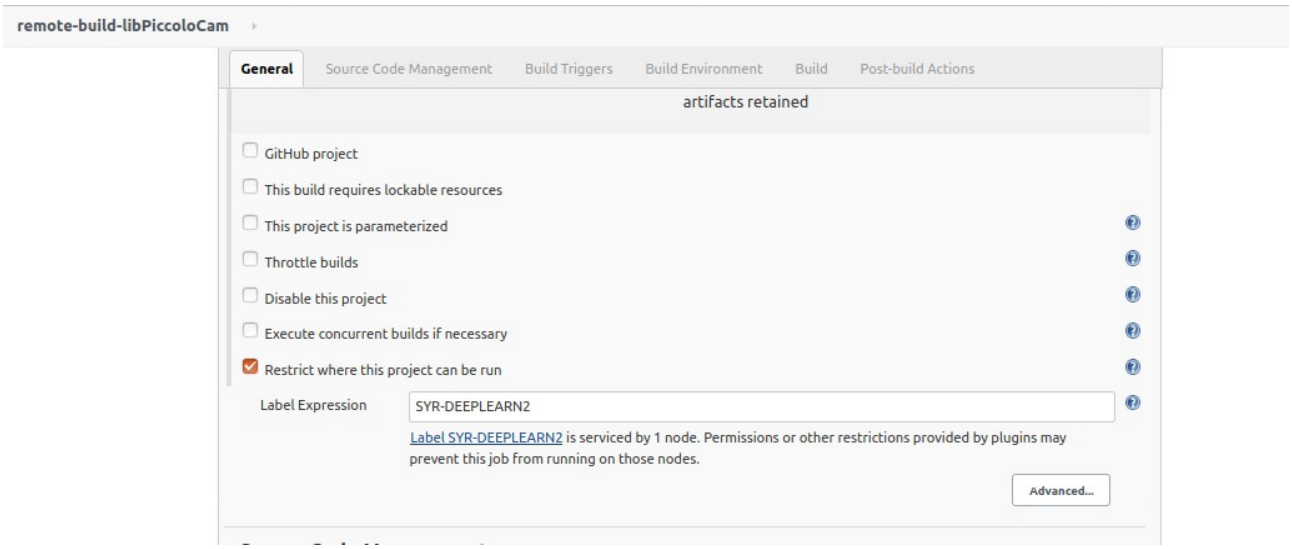


Step 2) In the next screen,

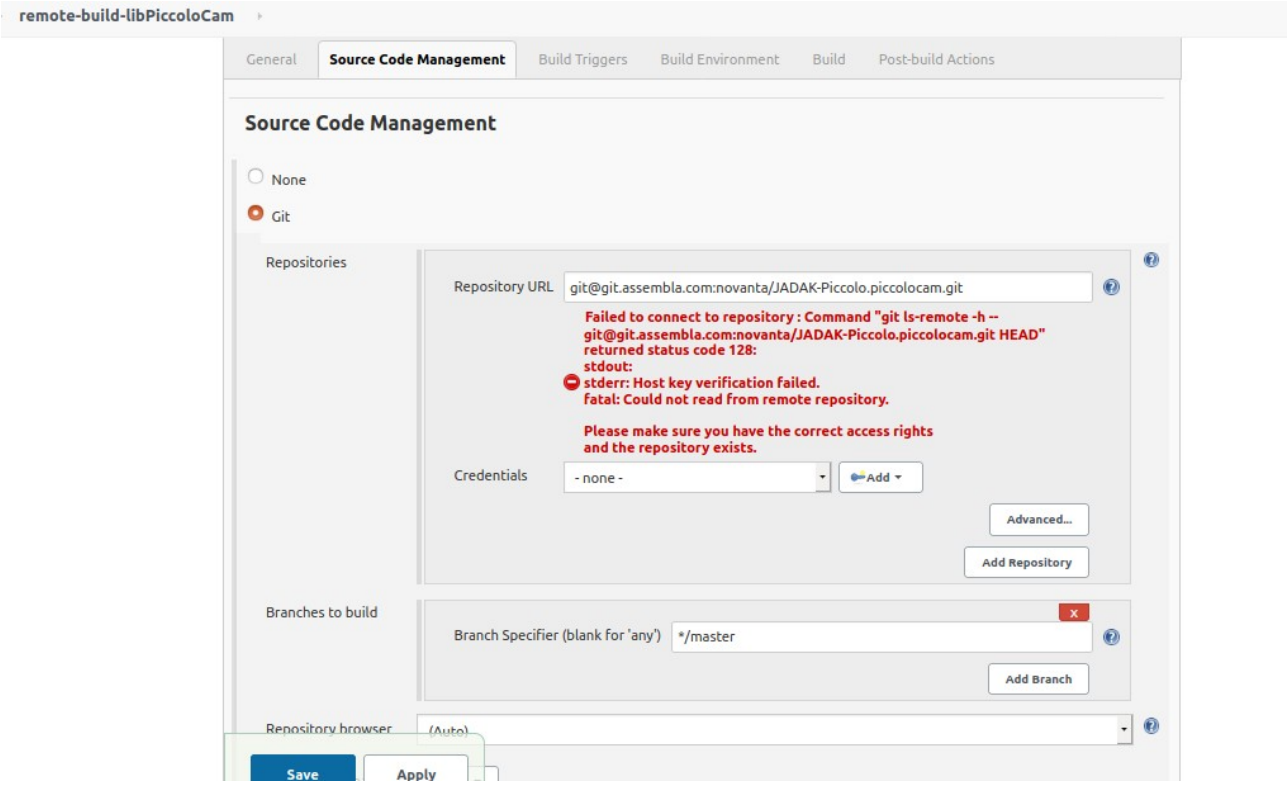
Enter the name of the item you want to create. We shall use the "remote-build-libpiccolocam" for this demo.
Select Freestyle project

General:

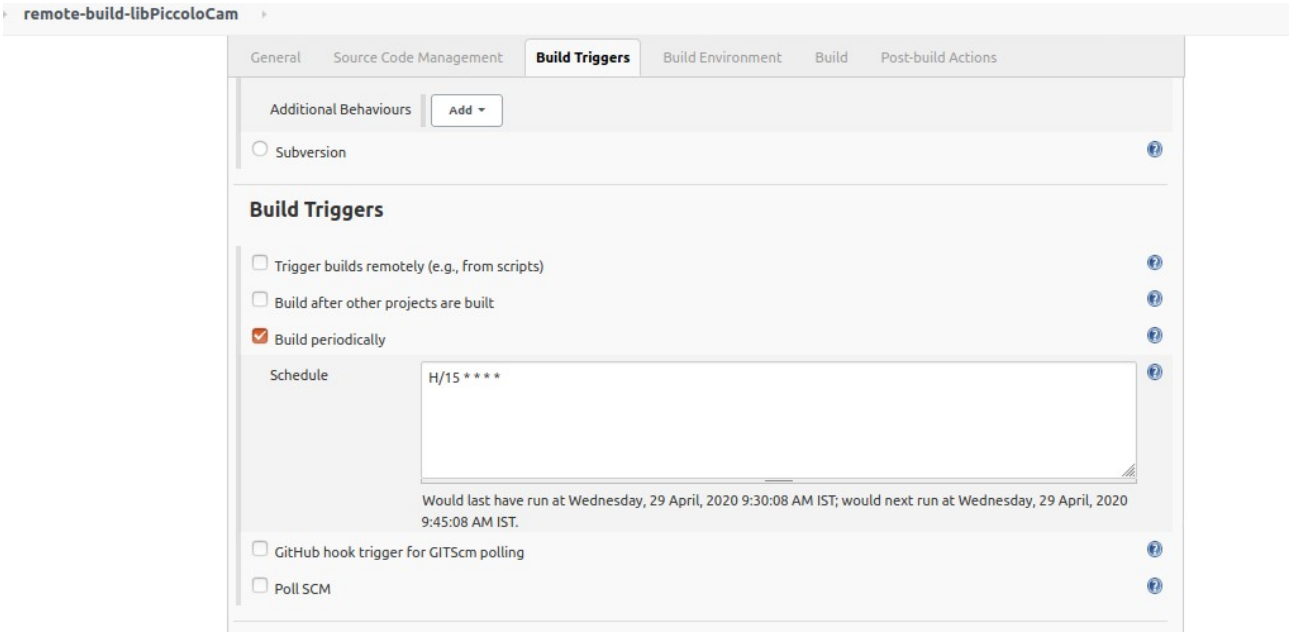

remote-build-libPiccoloCam
General
Source Code Management
Build Triggers
Build Environment
Build
Post-build Actions
Description
remote -build - libPiccoloCam
[Plain text] Preview
Discard old builds
Strategy
Log Rotation
Days to keep builds
15
if not empty, build records are only kept up to this number of days
Max # of builds to keep
15
if not empty, only up to this number of build records are kept
Days to keep artifacts
15
if not empty, artifacts from builds older than this number of days will be deleted, but the logs, history, reports, etc for the build will be kept
Max # of builds to keep with artifacts
15
if not empty, only up to this number of builds have their artifacts retained
Save
Apply


remote-build-libPiccoloCam
General
Source Code Management
Build Triggers
Build Environment
Build
Post-build Actions
artifacts retained
GitHub project
This build requires lockable resources
This project is parameterized
Throttle builds
Disable this project
Execute concurrent builds if necessary
Restrict where this project can be run
Label Expression
SYR-DEEPLEARN2
Label SYR-DEEPLEARN2 is serviced by 1 node. Permissions or other restrictions provided by plugins may prevent this job from running on those nodes.
Advanced...

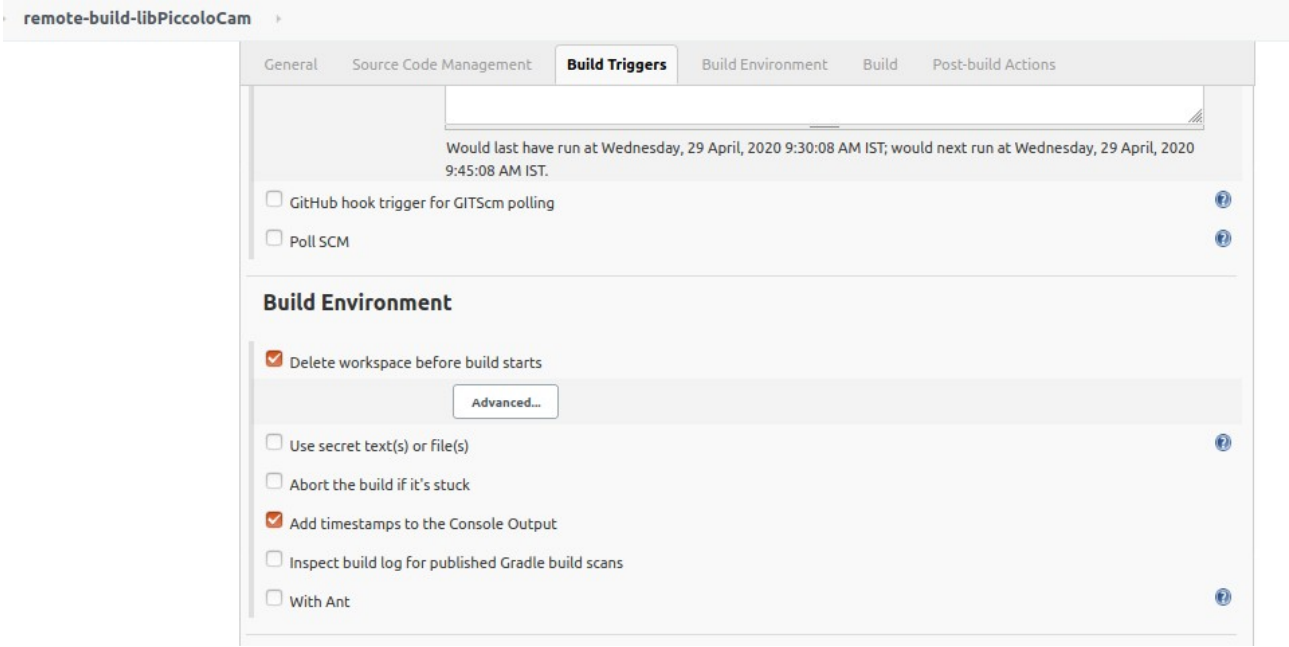
Source code Management



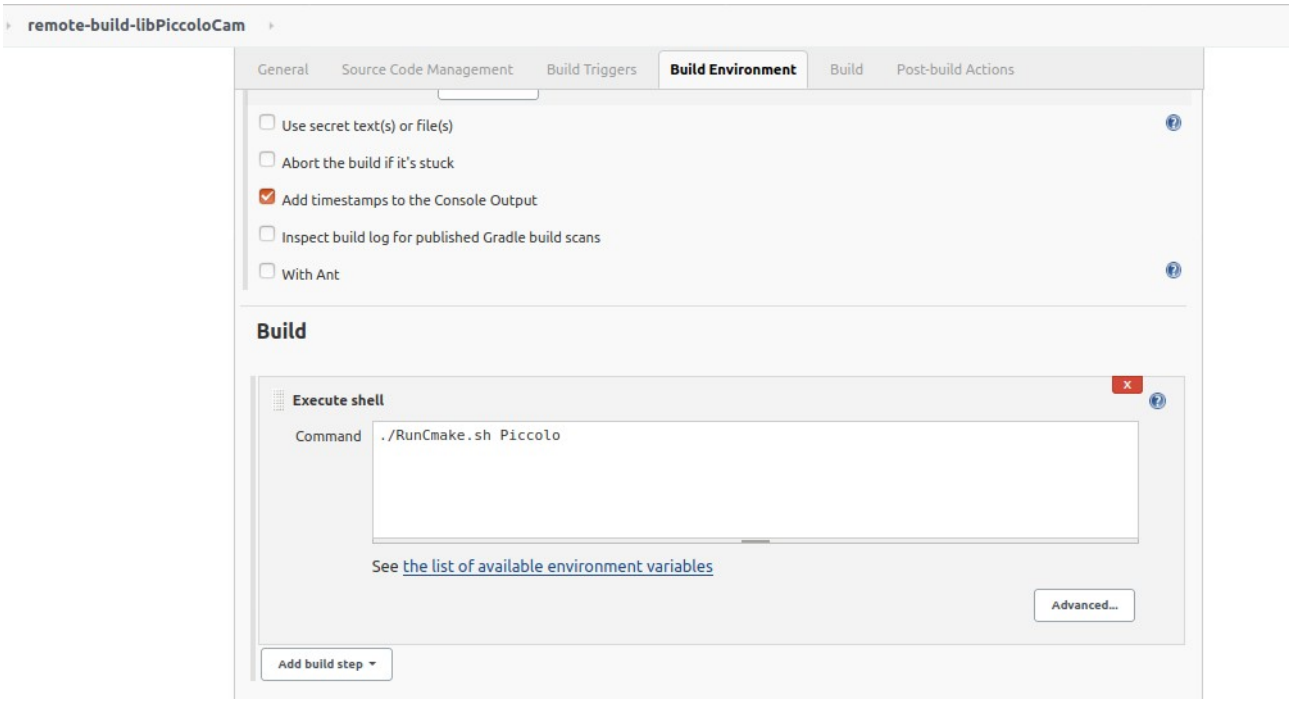
Build Triggers



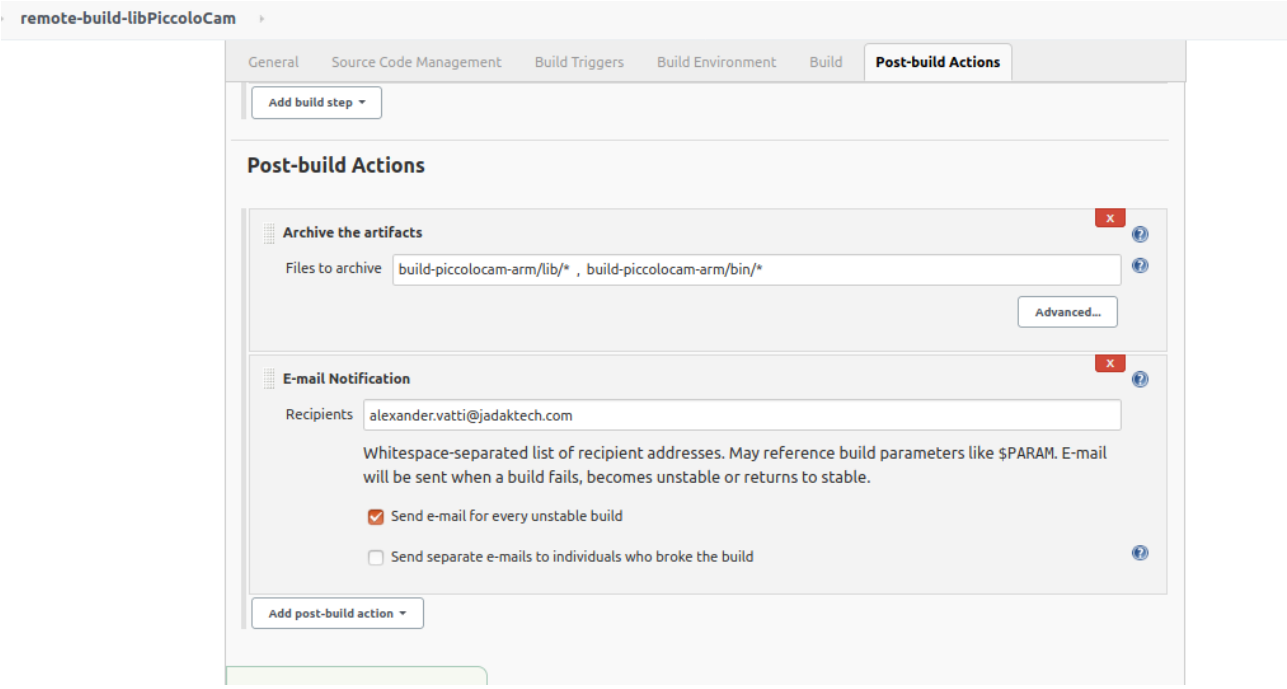
Build Environment



Build



Post-Build Actions



Finally: