

Table of Contents

Install Jenkins on Ubuntu 18.04:	2
Install Java	2
Add the Jenkins Debian repository	2
Install and Configure Jenkins	2
Starting Jenkins	2
Setting Up Jenkins	3
Installing suggested plugins	4
Open the Jenkins:	7
Creating a Freestyle Build Job : Java:	7
Creating a Freestyle Build Job : cpp:	16
Creating a Freestyle Build Job : qt:	16
Configuring the Node in Jenkins	17
Connection the Node in Jenkins:	22
Creating a Freestyle Build Job : libPiccoloCam : Run on Remote	22

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 update  
sudo 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)

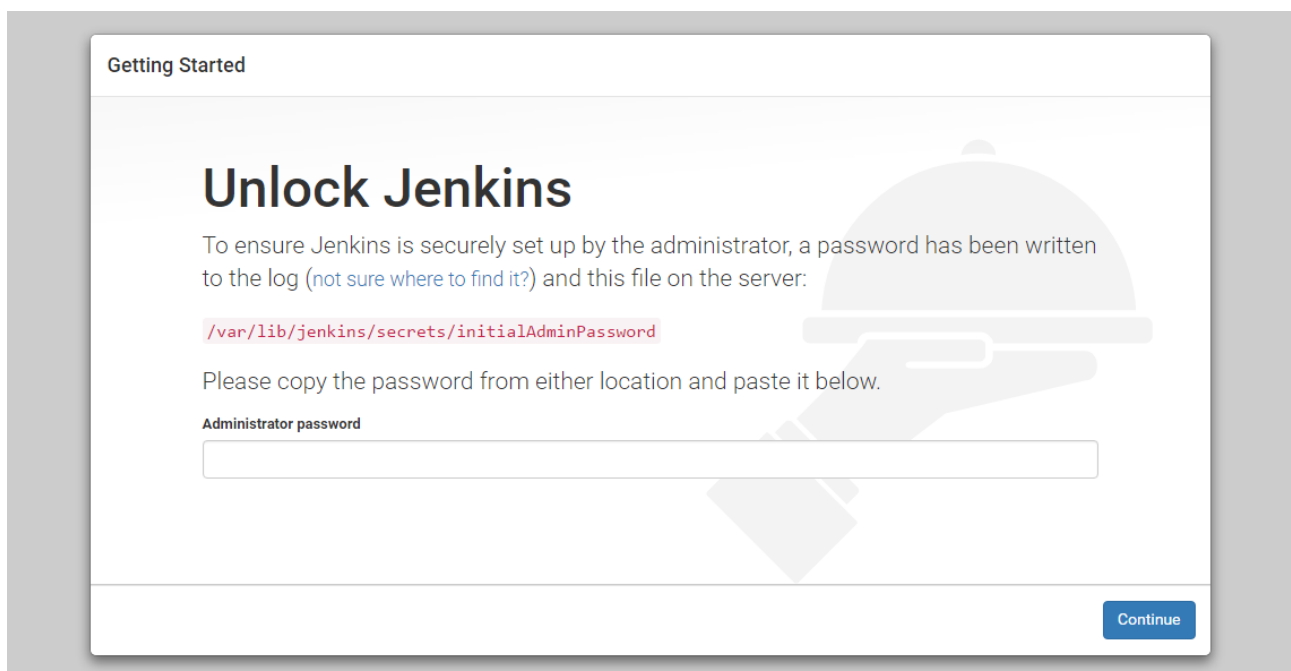
CGroup: /system.slice/jenkins.service

```
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`

The image shows the Jenkins 'Getting Started' screen. At the top, it says 'Getting Started'. Below that is the heading 'Unlock Jenkins'. A paragraph explains that 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`. It then asks the user to please copy the password from either location and paste it below. There is a label 'Administrator password' above a text input field. In the bottom right corner, there is a blue button labeled 'Continue'.

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

Continue

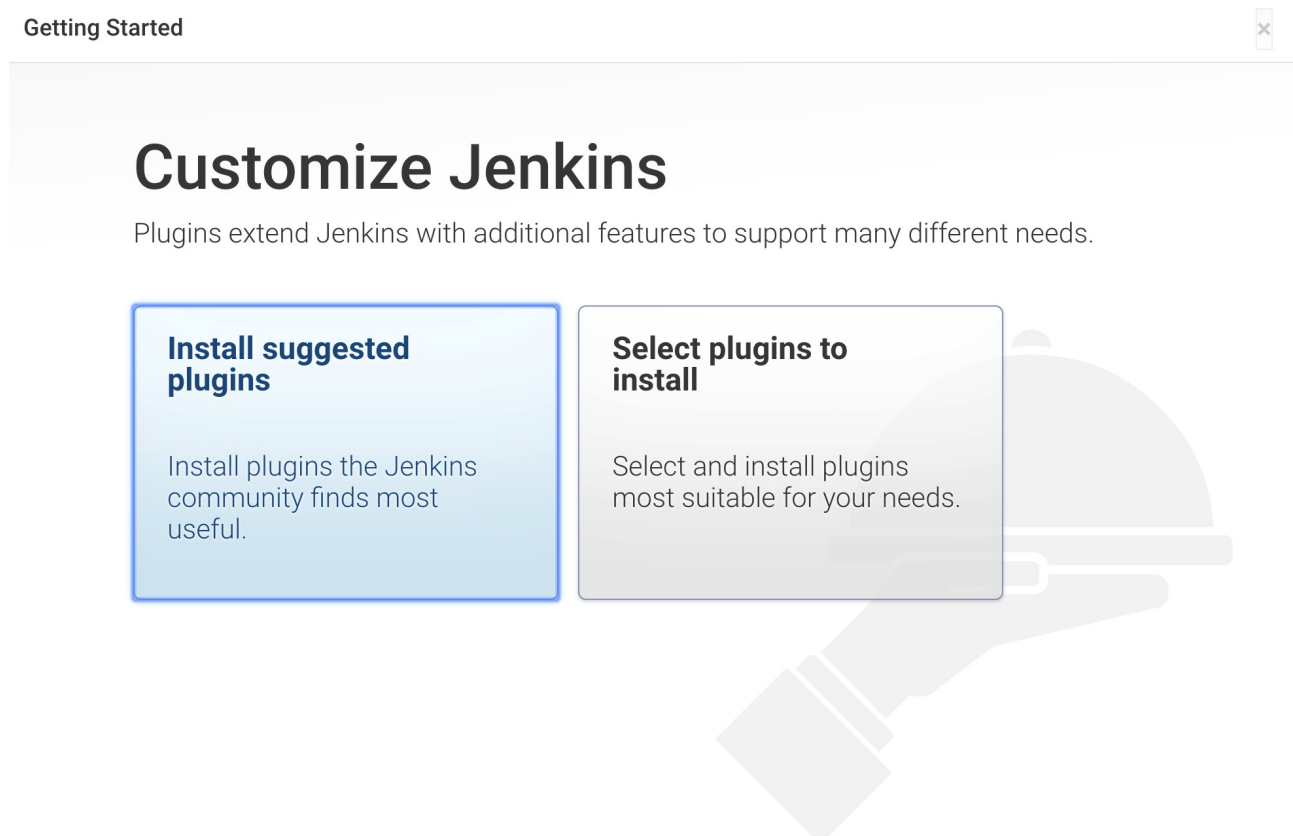
In the terminal window, use the cat command to display the password:

```
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:

Getting Started

✓ Folders	✓ OWASP Markup Formatter	✓ Build Timeout	✓ Credentials Binding	<pre>** Pipeline: Milestone Step ** JavaScript GUI Lib: jQuery bundles (jQuery and jQuery UI) ** Jackson 2 API ** JavaScript GUI Lib: ACE Editor bundle ** Pipeline: SCM Step ** Pipeline: Groovy ** Pipeline: Input Step ** Pipeline: Stage Step ** 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</pre>
✓ Timestampers	✓ 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	

Getting Started

Create First Admin User

Username:

Password:

Confirm password:

Full name:

E-mail address:

Jenkins 2.121.1

Continue as admin

Save and Continue

Enter the name and password for your user:

Getting Started

Create First Admin User

Username:

sammy

Password:

.....

Confirm password:

.....

Full name:

Sammy the Shark

E-mail address:

sammy@example.com

Jenkins 2.121.1

Continue as admin

Save and Continue

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:

Getting Started

Instance Configuration

Jenkins URL:

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 2.121.1

Not now

Save and Finish


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

 **Jenkins**

Jenkins ▶

1

User:

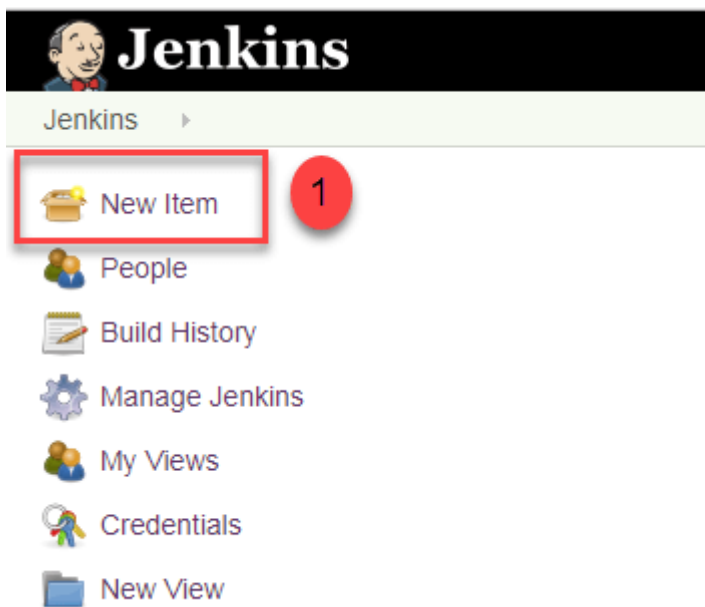
Password:

☐ Remember me on this computer

2

log in

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


Enter an item name

Hello World


1

Required field


2

**Freestyle project**


This is the central feature of Jenkins. Jenkins will build your project, combining any SCM with any tool used for something other than software build.

**Pipeline**


Orchestrates long-running activities that can span multiple build agents. Suitable for building pipelines 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 builds, etc.

**Folder**

Creates a container that stores nested items in it. Useful for grouping things together. Unlike view, separate namespace, so you can have multiple things of the same name as long as they are in different namespaces.

**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.

3

OK

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

General

Source Code Management

Build Triggers

Build Environment

Build

Post-build Actions

Description

Hello world java test program

G

[Plain text] [Preview](#)

☐ Discard old builds

☐ GitHub project

☐ This project is parameterized

☐ Throttle builds

☐ Disable this project

☐ Execute concurrent builds if necessary

?

?

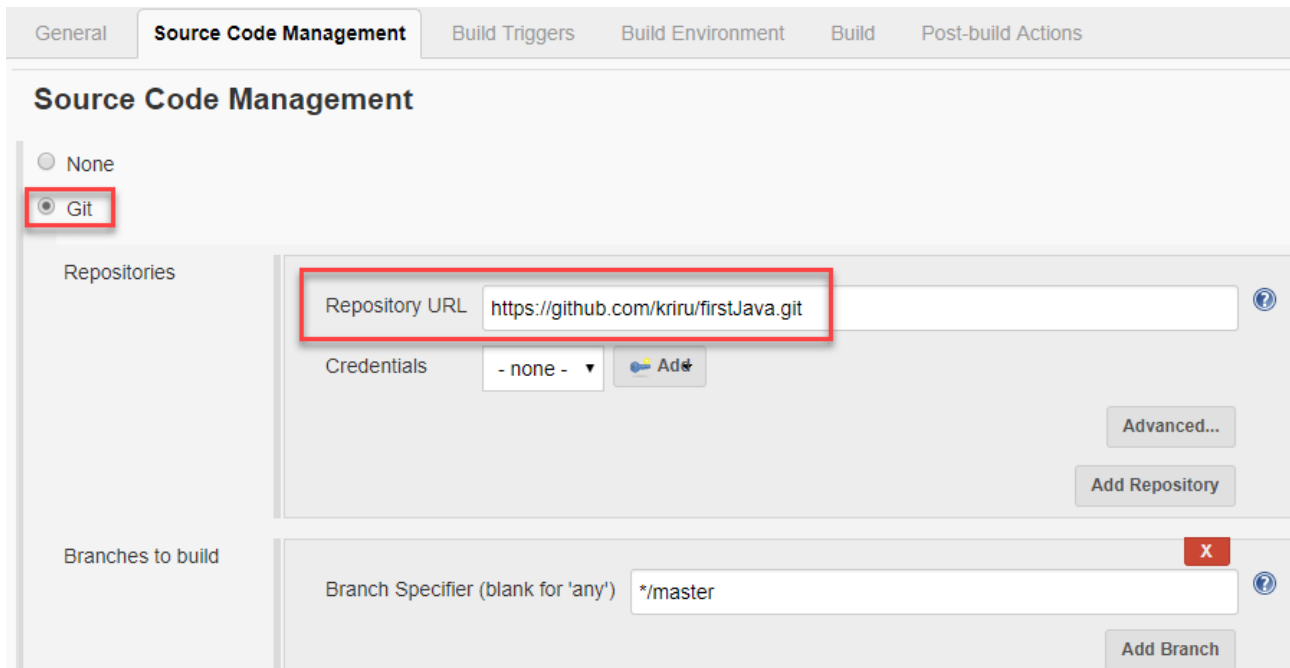
?

?

?

Advanced...

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



The screenshot shows the Jenkins configuration page for 'Source Code Management'. The 'Git' radio button is selected and highlighted with a red box. Below it, the 'Repository URL' field is also highlighted with a red box and contains the text 'https://github.com/kriru/firstJava.git'. The 'Credentials' dropdown is set to '- none -' with an 'Add' button next to it. There are 'Advanced...' and 'Add Repository' buttons to the right. In the 'Branches to build' section, the 'Branch Specifier (blank for 'any')' field contains '*/master' and is highlighted with a red box. There is a red 'X' icon and an 'Add Branch' button to the right of this field.

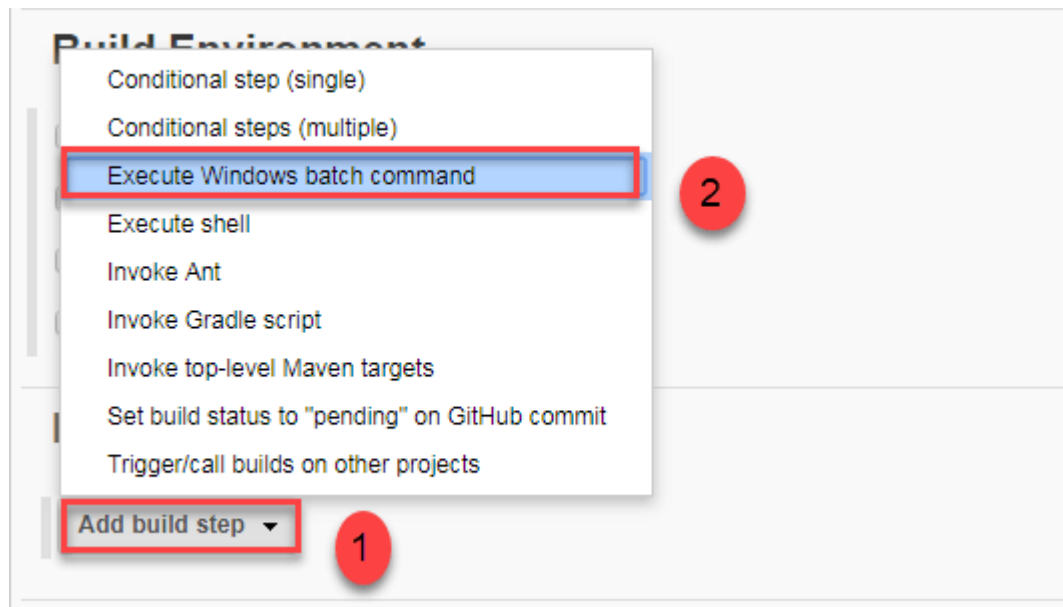
It is also possible for you to use a local repository.

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.

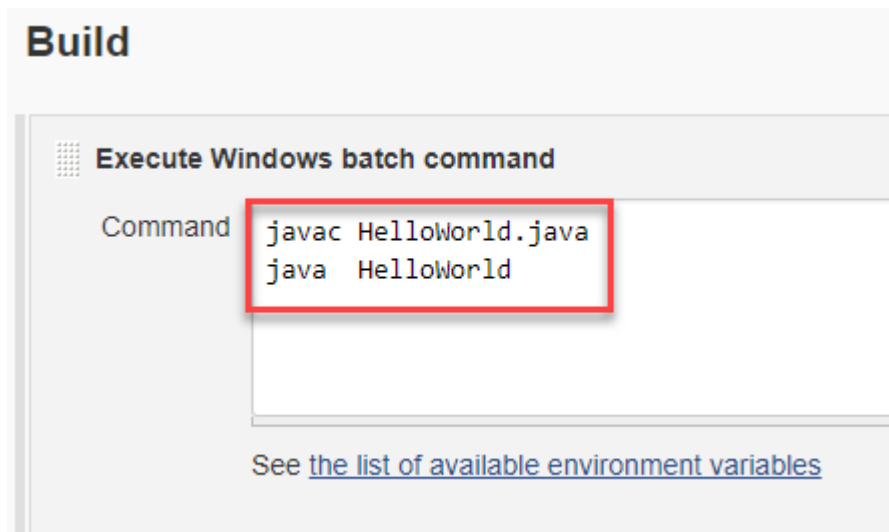


Here, I have added the java commands to compile the java code.

I have added the following windows commands:

```
javac HelloWorld.java
```

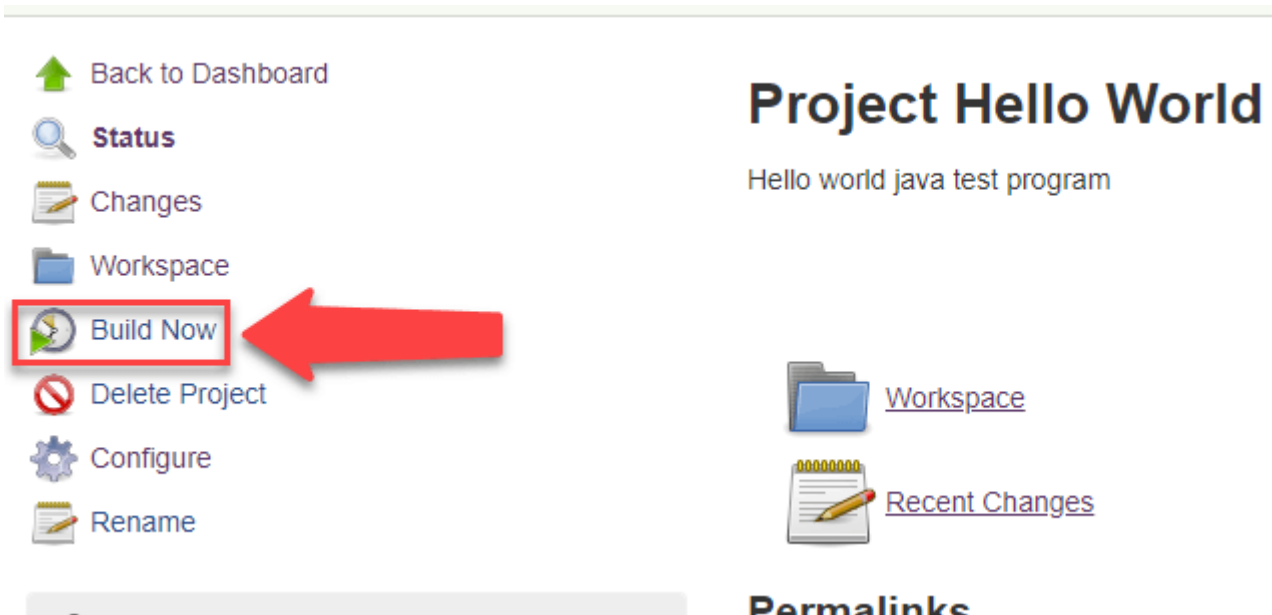
```
java HelloWorld
```



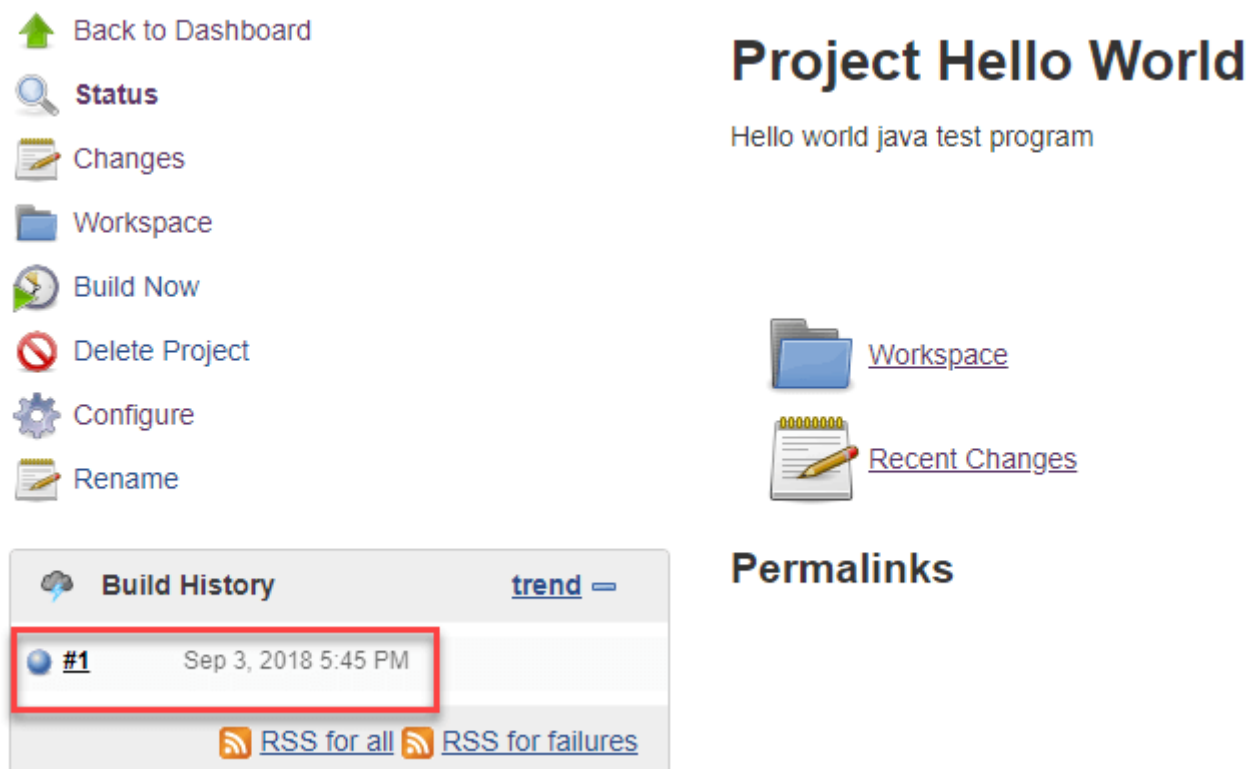
Step 7) When you have entered all the data,

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.

The screenshot shows the Jenkins web interface for a build named 'Hello World' (build #1). On the left sidebar, the 'Console Output' link is highlighted with a red box. A large green dashed arrow points from this link to the 'Finished: SUCCESS' status box at the bottom right of the console output area. The console output itself shows the build process: it starts by cloning a repository from GitHub, fetches upstream changes, and then runs a series of git commands to initialize the workspace and fetch the latest code. Finally, it runs 'javac HelloWorld.java' and 'java HelloWorld', which outputs 'Hello World'.

```
Started by user The_Guru99
Building in workspace C:\Program Files (x86)\Jenkins\workspace\Hello World
Cloning the remote Git repository
Cloning repository https://github.com/kriru/firstJava.git
> git.exe init C:\Program Files (x86)\Jenkins\workspace\Hello World # timeout=10
Fetching upstream changes from https://github.com/kriru/firstJava.git
> git.exe --version # timeout=10
> git.exe fetch --tags --progress https://github.com/kriru/firstJava.git +refs/heads/*:refs/remotes/origin/*
> git.exe config --add remote.origin.fetch +refs/heads/*:refs/remotes/origin/*
> git.exe config remote.origin.url https://github.com/kriru/firstJava.git # timeout=10
Fetching upstream changes from https://github.com/kriru/firstJava.git
> git.exe fetch --tags --progress https://github.com/kriru/firstJava.git +refs/heads/*:refs/remotes/origin/*
> git.exe rev-parse "refs/remotes/origin/master^{commit}" # timeout=10
> git.exe rev-parse "refs/remotes/origin/origin/master^{commit}" # timeout=10
> git.exe rev-parse "origin/master^{commit}" # timeout=10

C:\Program Files (x86)\Jenkins\workspace\Hello World>javac HelloWorld.java

C:\Program Files (x86)\Jenkins\workspace\Hello World>java HelloWorld
Hello World

Finished: SUCCESS
```

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

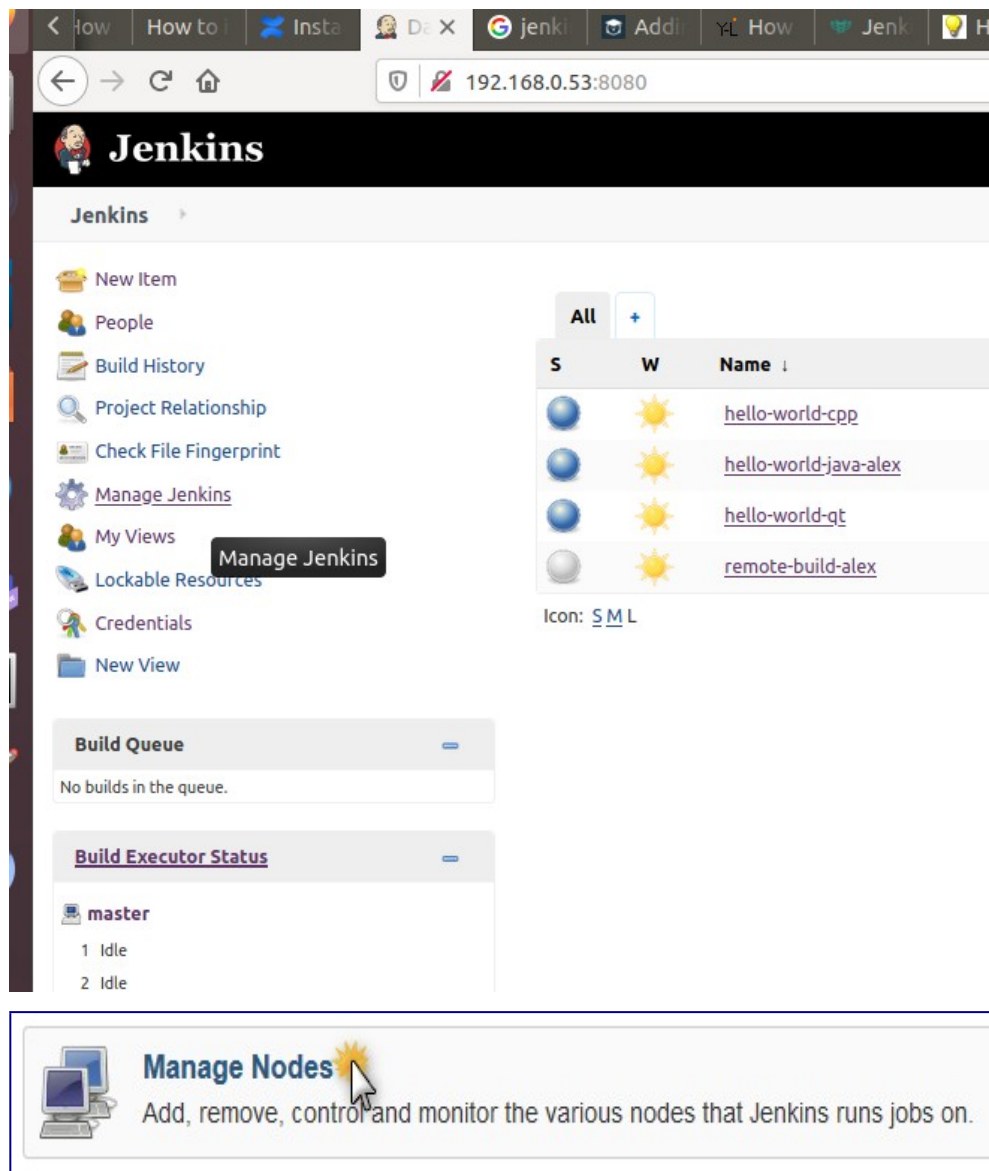
<http://yallalabs.com/devops/how-to-add-linux-slave-node-agent-node-jenkins/>

<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**



The screenshot shows the Jenkins web interface. The left sidebar contains a menu with items like 'New Item', 'People', 'Build History', 'Project Relationship', 'Check File Fingerprint', 'Manage Jenkins' (highlighted), 'My Views', 'Lockable Resources', 'Credentials', and 'New View'. The main content area shows a table of nodes with columns 'S', 'W', and 'Name'. The table lists four nodes: 'hello-world-cpp', 'hello-world-java-alex', 'hello-world-qt', and 'remote-build-alex'. Below the table, there is a 'Build Queue' section showing 'No builds in the queue.' and a 'Build Executor Status' section showing 'master' with two idle executors. At the bottom, a 'Manage Nodes' button is highlighted with a mouse cursor, with a tooltip that reads 'Add, remove, control and monitor the various nodes that Jenkins runs jobs on.'

S	W	Name ↓
		hello-world-cpp
		hello-world-java-alex
		hello-world-qt
		remote-build-alex

Icon: [S](#) [M](#) [L](#)

Build Queue

No builds in the queue.

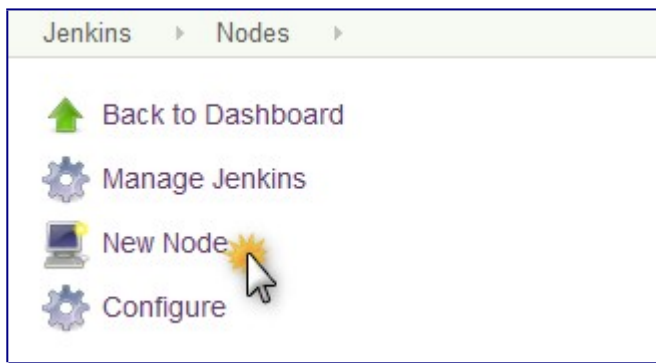
Build Executor Status

master

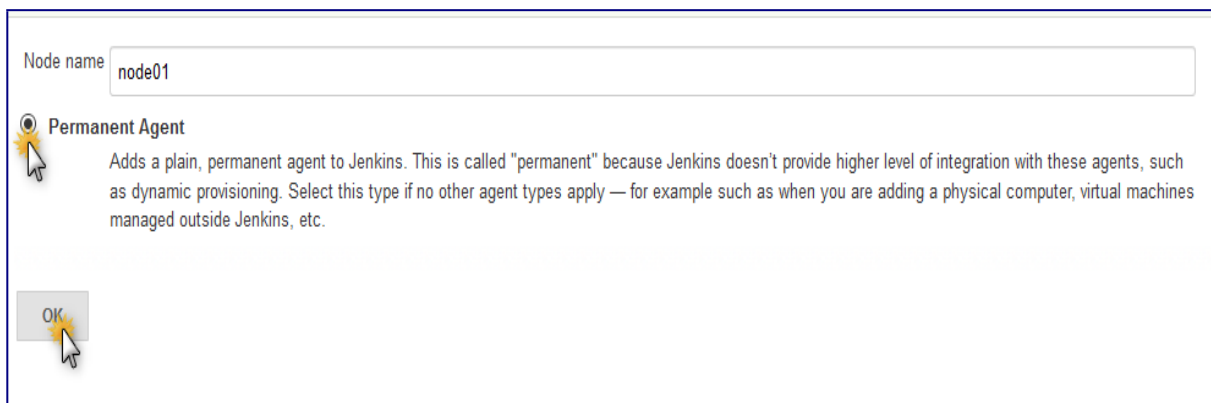
- 1 Idle
- 2 Idle

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

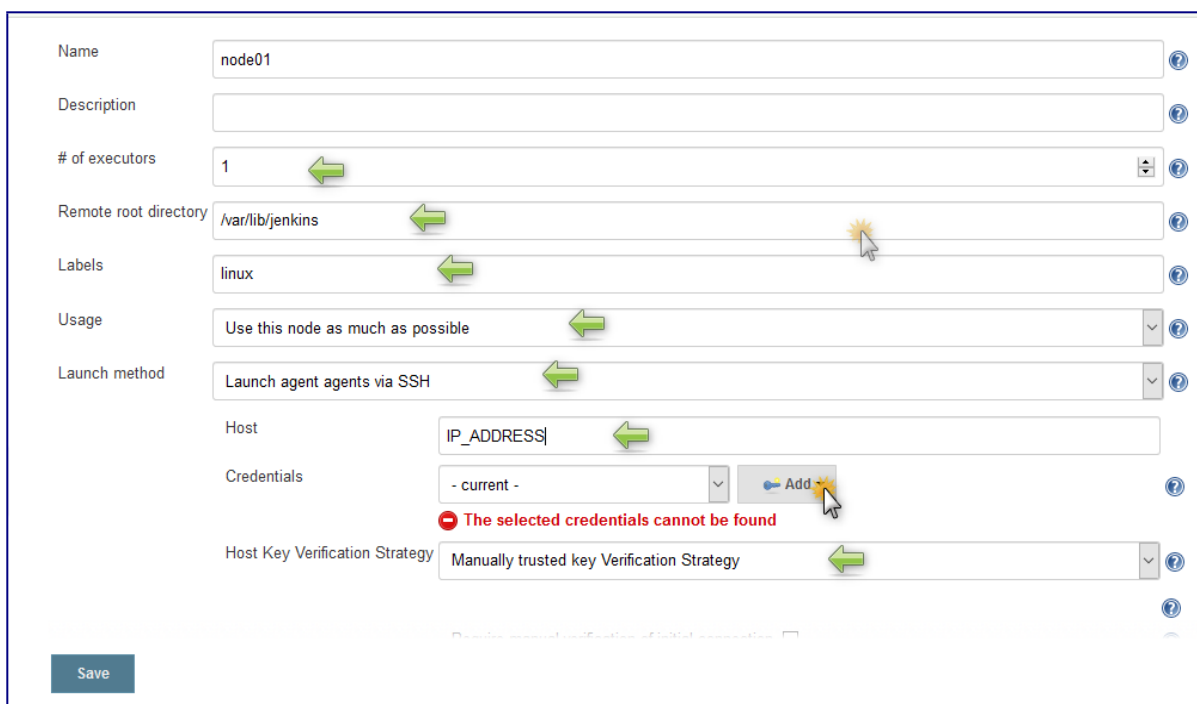
Then click on the **New Node** button:



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.

Jenkins Credentials Provider: Jenkins

Add Credentials

Domain: Global credentials (unrestricted)

Kind: SSH Username with private key

Scope: Global (Jenkins, nodes, items, all child items, etc)

Username: jenkins

Private Key: ☒ Enter directly

Key: -----BEGIN RSA PRIVATE KEY-----
-----END RSA PRIVATE KEY-----

Passphrase:



ID: ssh-jenkins-user

Description: ssh jenkins private Key

Add Cancel


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	<input type="text" value="SYR-DEEPLearn2"/>	?
Description	<input type="text" value="SYR-DEEPLearn2"/>	?
# of executors	<input type="text" value="2"/>	?
Remote root directory	<input type="text" value="/home/alexvatti/AlexBuildLocation"/>	?
Labels	<input type="text" value="SYR-DEEPLearn2"/>	?
Usage	<input type="text" value="Only build jobs with label expressions matching this node"/>	?
Launch method	<input type="text" value="Launch agents via SSH"/>	?
Host	<input type="text" value="172.28.149.86"/>	?
Credentials	<input type="text" value="alexvatti/*****"/> <input type="button" value="Add"/>	?
Host Key Verification Strategy	<input type="text" value="Non verifying Verification Strategy"/>	?









Jenkins Credentials Provider: Jenkins



Add Credentials

Domain	<input type="text" value="Global credentials (unrestricted)"/>	?
Kind	<input type="text" value="Username with password"/>	?
Scope	<input type="text" value="Global (Jenkins, nodes, items, all child items, etc)"/>	?
Username	<input type="text" value="alexvatti"/>	?
Password	<input type="password" value="*****"/>	?
ID	<input type="text"/>	?
Description	<input type="text"/>	?

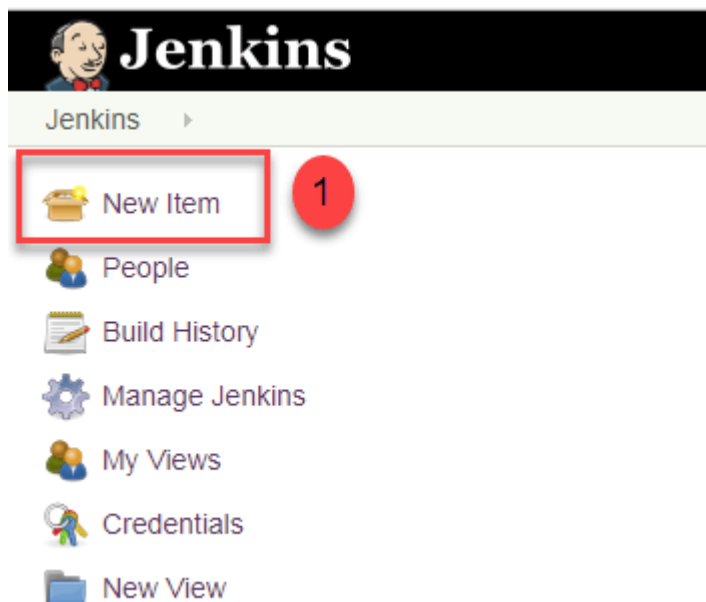
S	Name	Architecture	Clock Difference	Free Disk Space	Free Swap Space	Free Temp Space	Response Time
	master	Linux (amd64)	In sync	201.90 GB	976.00 MB	201.90 GB	0ms 
	SYR-DEEPLearn2	Linux (amd64)	In sync	117.07 GB	15.59 GB	53.08 GB	3036ms 
Data obtained		6 min 26 sec	6 min 26 sec	6 min 19 sec	6 min 22 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

remote-build-libPiccoloCam

General **Source Code Management** Build Triggers Build Environment Build Post-build Actions

Source Code Management

☐ None
☒ Git

Repositories

Repository URL

Failed to connect to repository : Command "git ls-remote -h -- git@git.assembla.com:novanta/JADAK-Piccolo.piccolocam.git HEAD" returned status code 128:
stdout:
stderr: Host key verification failed.
fatal: Could not read from remote repository.

Please make sure you have the correct access rights and the repository exists.

Credentials

Branches to build

Branch Specifier (blank for 'any')

Repository browser

Build Triggers

remote-build-libPiccoloCam

General Source Code Management **Build Triggers** Build Environment Build Post-build Actions

Additional Behaviours

☐ Subversion

Build Triggers

☐ Trigger builds remotely (e.g., from scripts)
☐ Build after other projects are built
☒ Build periodically

Schedule

Would last have run at Wednesday, 29 April, 2020 9:30:08 AM IST; would next run at Wednesday, 29 April, 2020 9:45:08 AM IST.

☐ GitHub hook trigger for GITScm polling
☐ Poll SCM

Build Environment

remote-build-libPiccoloCam

General Source Code Management **Build Triggers** Build Environment Build Post-build Actions

Would last have run at Wednesday, 29 April, 2020 9:30:08 AM IST; would next run at Wednesday, 29 April, 2020 9:45:08 AM IST.

☐ GitHub hook trigger for GITScm polling

☐ Poll SCM

Build Environment

☒ Delete workspace before build starts

Advanced...

☐ Use secret text(s) or file(s)

☐ Abort the build if it's stuck

☒ Add timestamps to the Console Output

☐ Inspect build log for published Gradle build scans

☐ With Ant

Build

remote-build-libPiccoloCam

General Source Code Management Build Triggers **Build Environment** Build Post-build Actions

☐ Use secret text(s) or file(s)

☐ Abort the build if it's stuck

☒ Add timestamps to the Console Output

☐ Inspect build log for published Gradle build scans

☐ With Ant

Build

Execute shell

Command `./RunCmake.sh Piccolo`

See [the list of available environment variables](#)

Advanced...

Add build step

Post-Build Actions

remote-build-libPiccoloCam

GeneralSource Code ManagementBuild TriggersBuild EnvironmentBuildPost-build Actions

Add build step

Post-build Actions

Archive the artifacts

Files to archivebuild-piccolocam-arm/lib/* , build-piccolocam-arm/bin/*

Advanced...

E-mail Notification

Recipientsalexander.vatti@jadaktech.com

Whitespace-separated list of recipient addresses. May reference build parameters like \$PARAM. E-mail will be sent when a build fails, becomes unstable or returns to stable.

☒ Send e-mail for every unstable build

☐ Send separate e-mails to individuals who broke the build

Add post-build action

Finally:

Jenkins

search

adminlog out

New ItemPeopleBuild HistoryProject RelationshipCheck File FingerprintManage JenkinsMy ViewsLockable ResourcesCredentialsNew View

Build QueueNo builds in the queue.

Build Executor Status

master1 Idle2 Idle

SYR-DEEPLearn21 remote-build-poky-pennywhistle #12 Idle

All

S	W	Name	Last Success	Last Failure	Last Duration
		hello-world-ccp	1 day 23 hr - #7	1 day 23 hr - #1	1,7 sec
		hello-world-java-alex	1 day 23 hr - #3	N/A	2.3 sec
		hello-world-gt	1 day 23 hr - #6	1 day 23 hr - #1	1.7 sec
		remote-build-clarifylinux	4 min 41 sec - #36	N/A	3 min 1 sec
		remote-build-libPiccoloCam	12 min - #34	N/A	6 min 18 sec
		remote-build-poky-pennywhistle	N/A	N/A	N/A
		remote-build-poky-piccolo	N/A	N/A	N/A

Icon: S M L

LegendAtom feed for allAtom feed for failuresAtom feed for just latest builds