

Jenkins

1. [Download and install Jenkins on Windows x64, CentOS and Amazon Linux.](#)
2. [Managing and installing plugins.](#)
3. [First simple job examle in Jenkins.](#)
4. [Simple job example in Jenkins with deploy.](#)
5. [Simple job example in Jenkins with deploy using plug. “Publish over SSH”.](#)
6. [Simple CI/CD pipeline example.](#)
7. [Simple CI/CD pipeline example using AWS.](#)
8. [Jenkins nodes\(agent/slave\).](#)
9. [Simple job using Jenkins agent.](#)
10. [Simple Jenkis pipeline example.](#)
11. [Simple Jenkis pipeline script from SCM \(Source Control Managment\).](#)
12. [Jenkis for Java app using Maven.](#)

1. Download and install Jenkins on Windows x64, CentOS and Amazon Linux.

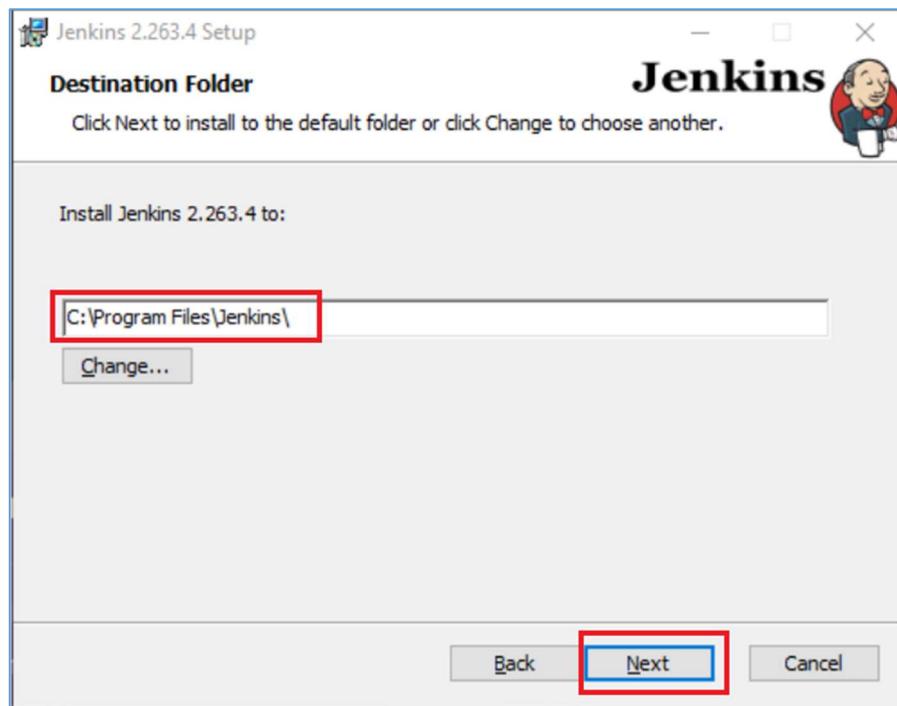
- **Windows**

- a. Download newest version Jenkins from [website](#) and run installer

 Download Jenkins 2.375.1 LTS for:	 Download Jenkins 2.381 for:
Generic Java package (.war) SHA-256: e96ae7f59d8a009bdbf3551d5e9facd97ff8a6d404c7ea2438ef267 988d53427	Generic Java package (.war) SHA-256: 62ca5dcecbf176452d94d4438488662e223ab9594dcc564f065c6 3832a47302
Docker	Docker
Ubuntu/Debian	Ubuntu/Debian
CentOS/Fedora/Red Hat	CentOS/Fedora/Red Hat
Windows	Windows
openSUSE	openSUSE
FreeBSD	Arch Linux
Gentoo	FreeBSD
macOS	Gentoo
OpenBSD	macOS



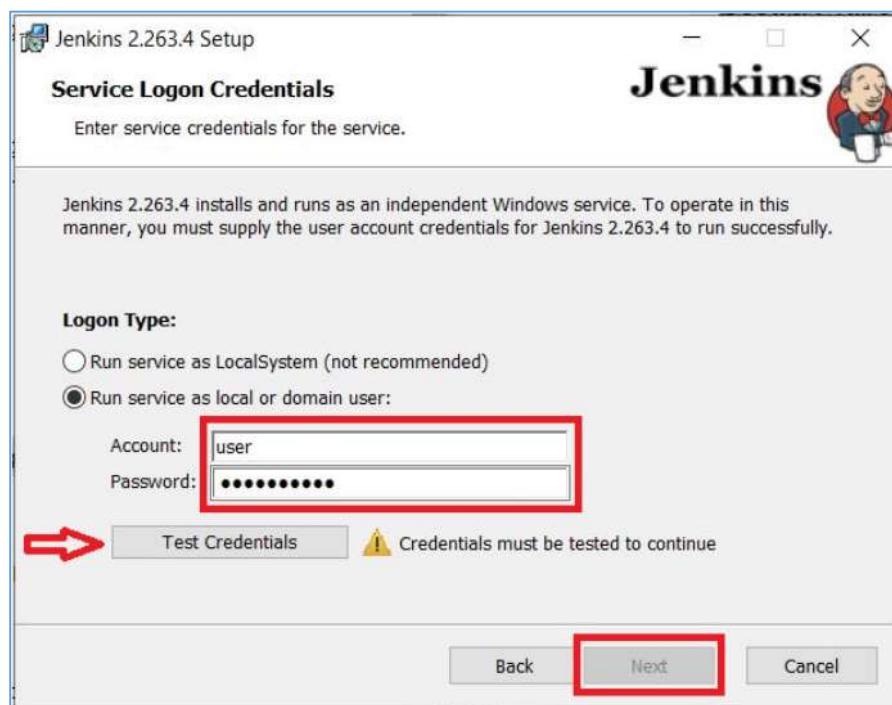
- b. Select destination folder



c. Service logon credentials

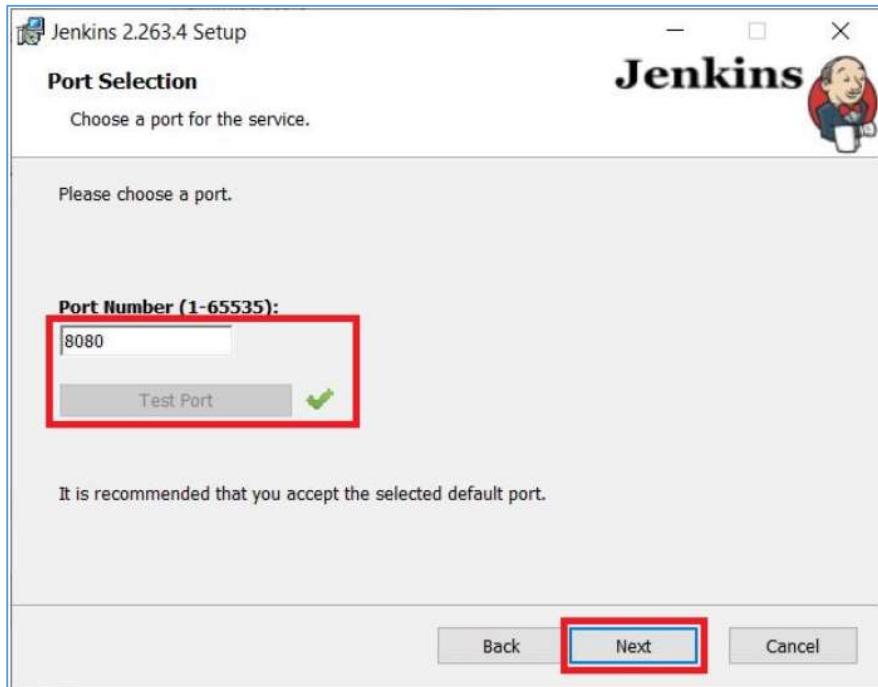
When Installing Jenkins, it is recommended to install and run Jenkins as an independent windows service using a **local or domain user** as it is much safer than running Jenkins using **LocalSystem(Windows equivalent of root)** which will grant Jenkins full access to your machine and services.

To run Jenkins service using a local or domain user, specify the domain user name and password with which you want to run Jenkins, click on Test Credentials to test your domain credentials and click on Next.



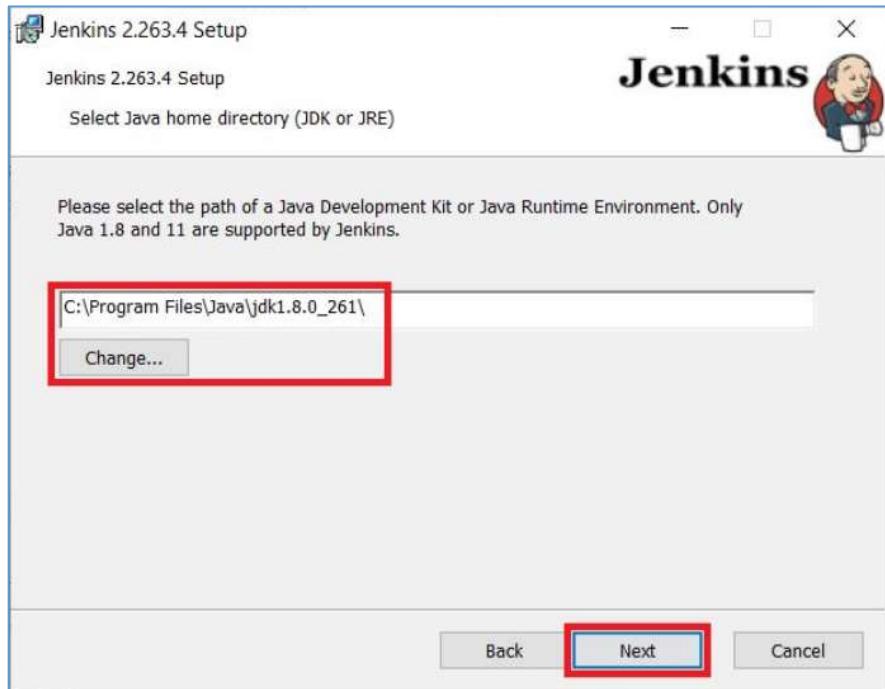
d. Port selection

Specify the port on which Jenkins will be running, Test Port button to validate whether the specified port is free on your machine or not. Consequently, if the port is free, it will show a green tick mark as shown below, then click on Next.



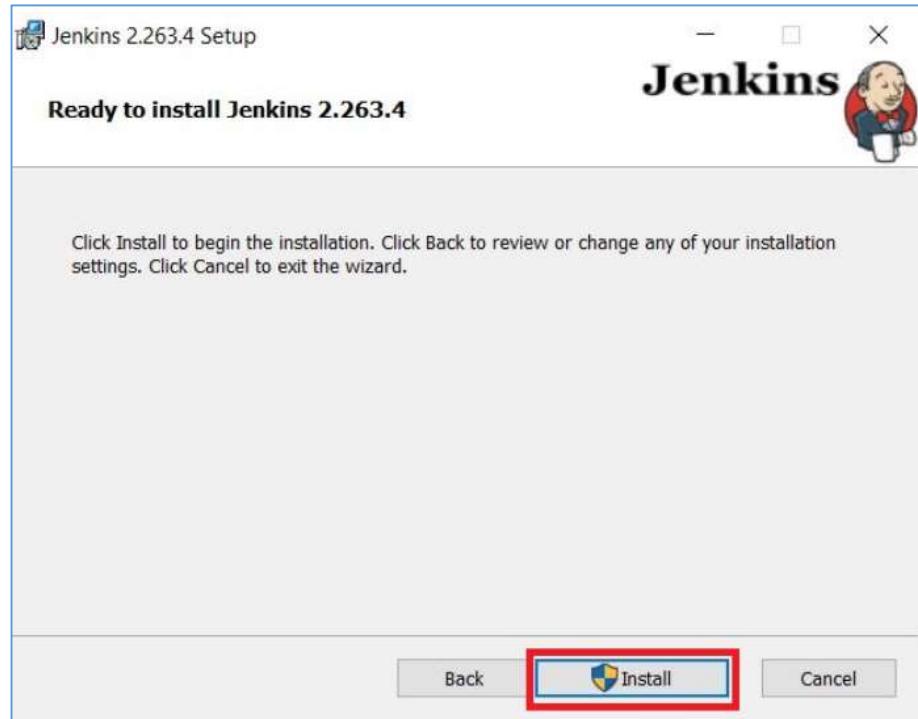
e. Select Java home directory

The installation process checks for Java on your machine and prefills the dialog with the Java home directory. If the needed Java version is not installed on your machine, you will be prompted to install it.



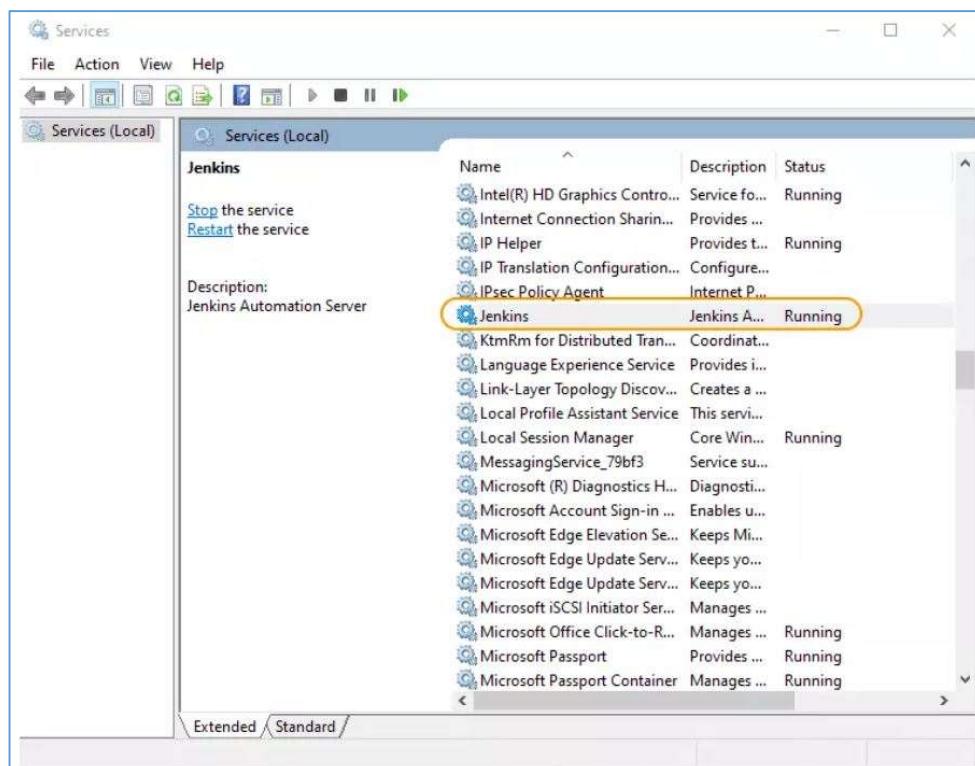
f. Install Jenkins

Click on the Install button to start the installation of Jenkins.



g. Finish Jenkins instalation

Once the installation completes, click on Finish to complete the installation.
Jenkins will be installed as a Windows Service. You can validate this by browsing the services section, as shown below:



- CentOS 7

a. Download and install (package manager)

You can install Jenkins through yum on Red Hat Enterprise Linux, CentOS, and other Red Hat based distributions. You need to choose either the Jenkins Long Term Support release or the Jenkins weekly release.

When installing the latest Jenkins version on Centos 7 make sure java 11 is the main alternative among other java versions installed:

```
$ alternatives --config java
```

Long Term Support release:

```
sudo wget -O /etc/yum.repos.d/jenkins.repo \
https://pkg.jenkins.io/redhat-stable/jenkins.repo
sudo rpm --import https://pkg.jenkins.io/redhat-stable/jenkins.io.key
sudo yum upgrade
# Add required dependencies for the jenkins package
sudo yum install java-11-openjdk
sudo yum install jenkins
sudo systemctl daemon-reload
```

Weekly release:

```
sudo wget -O /etc/yum.repos.d/jenkins.repo \
https://pkg.jenkins.io/redhat/jenkins.repo
sudo rpm --import https://pkg.jenkins.io/redhat/jenkins.io.key
sudo yum upgrade
# Add required dependencies for the jenkins package
sudo yum install java-11-openjdk
sudo yum install jenkins
```

b. Start Jenkins

You can enable the Jenkins service to start at boot with the command:

```
sudo systemctl enable jenkins
```

You can start the Jenkins service with the command:

```
sudo systemctl start jenkins
```

You can check the status of the Jenkins service using the command:

```
sudo systemctl status jenkins
```

```
[vagrant@vm2 ~]$ 
[vagrant@vm2 ~]$ sudo systemctl status jenkins
● jenkins.service - Jenkins Continuous Integration Server
  Loaded: loaded (/usr/lib/systemd/system/jenkins.service; enabled; vendor preset: disabled)
    Active: active (running) since Sun 2022-12-11 11:53:36 UTC; 57min ago
      Main PID: 653 (java)
        CGroup: /system.slice/jenkins.service
                  └─653 /usr/bin/java -Djava.awt.headless=true -jar /usr/share/java/jenkins.war --webroot=%C/jenk...
Dec 11 12:36:18 vm2 jenkins[653]: 2022-12-11 12:36:18.090+0000 [id=339]           INFO      c.n.j.p.d....check
Dec 11 12:36:18 vm2 jenkins[653]: 2022-12-11 12:36:18.090+0000 [id=339]           INFO      c.n.j.p.d....leted
Dec 11 12:41:18 vm2 jenkins[653]: 2022-12-11 12:41:18.091+0000 [id=372]           INFO      c.n.j.p.d....gered
Dec 11 12:41:18 vm2 jenkins[653]: 2022-12-11 12:41:18.091+0000 [id=372]           INFO      c.n.j.p.d.Docke...
Dec 11 12:41:18 vm2 jenkins[653]: 2022-12-11 12:41:18.092+0000 [id=372]           INFO      c.n.j.p.d....check
Dec 11 12:41:18 vm2 jenkins[653]: 2022-12-11 12:41:18.092+0000 [id=372]           INFO      c.n.j.p.d....leted
Dec 11 12:46:18 vm2 jenkins[653]: 2022-12-11 12:46:18.092+0000 [id=394]           INFO      c.n.j.p.d....gered
Dec 11 12:46:18 vm2 jenkins[653]: 2022-12-11 12:46:18.093+0000 [id=394]           INFO      c.n.j.p.d.Docke...
Dec 11 12:46:18 vm2 jenkins[653]: 2022-12-11 12:46:18.093+0000 [id=394]           INFO      c.n.j.p.d....check
Dec 11 12:46:18 vm2 jenkins[653]: 2022-12-11 12:46:18.093+0000 [id=394]           INFO      c.n.j.p.d....leted
Hint: Some lines were ellipsized, use -l to show in full.
[vagrant@vm2 ~]$
```

* If you have a firewall installed, you must add Jenkins as an exception. You must change YOURPORT in the script below to the port you want to use. Port 8080 is the most common.

```
YOURPORT=8080
PERM="--permanent"
SERV="$PERM --service=jenkins"

firewall-cmd $PERM --new-service=jenkins
firewall-cmd $SERV --set-short="Jenkins ports"
firewall-cmd $SERV --set-description="Jenkins port exceptions"
firewall-cmd $SERV --add-port=$YOURPORT/tcp
firewall-cmd $PERM --add-service=jenkins
firewall-cmd --zone=public --add-service=http --permanent
firewall-cmd --reload
```

- [**Amazon Linux**](#)

- a. [Prerequisites](#)

An AWS account. If you don't have one, you can register [here](#).

An Amazon EC2 key pair. If you don't have one, refer to [Creating a key pair](#).

An AWS IAM User with programmatic key access and [permissions to launch EC2 instances](#)

- b. [Creating a key pair](#)

- c. [Creating security group](#)

- d. [Launching an Amazon EC2 instance](#)

- e. [Connecting to your Linux instance](#)

- f. [Installing and configuring Jenkins](#)

Completing the previous steps enables you to download and install Jenkins on AWS. To download and install Jenkins:

1. Ensure that your software packages are up to date on your instance by using the following command to perform a quick software update:

```
[ec2-user ~]$ sudo yum update -y
```

2. Add the Jenkins repo using the following command:

```
[ec2-user ~]$ sudo wget -O /etc/yum.repos.d/jenkins.repo \
https://pkg.jenkins.io/redhat-stable/jenkins.repo
```

3. Import a key file from Jenkins-CI to enable installation from the package:

```
[ec2-user ~]$ sudo rpm --import https://pkg.jenkins.io/redhat-stable/jenkins.io.key
[ec2-user ~]$ sudo yum upgrade
```

4. Install Java:

```
[ec2-user ~]$ sudo amazon-linux-extras install java-openjdk11 -y
```

5. Install Jenkins:

```
[ec2-user ~]$ sudo yum install jenkins -y
```

6. Enable the Jenkins service to start at boot:

```
[ec2-user ~]$ sudo systemctl enable jenkins
```

7. Start Jenkins as a service:

```
[ec2-user ~]$ sudo systemctl start jenkins
```

You can check the status of the Jenkins service using the command:

```
[ec2-user@ip-172-31-14-46 jobs]$ sudo systemctl status jenkins
● jenkins.service - Jenkins Continuous Integration Server
  Loaded: loaded (/usr/lib/systemd/system/jenkins.service; enabled; vendor preset: disabled)
  Active: active (running) since Sun 2022-12-04 12:16:03 UTC; 1 weeks 0 days ago
```

g. [Configuring Jenkins](#)

Jenkins is now installed and running on your EC2 instance. To configure Jenkins:

1. Connect to `http://<your_server_public_DNS>:8080` from your browser. You will be able to access Jenkins through its management interface:

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

2. As prompted, enter the password found in `/var/lib/jenkins/secrets/initialAdminPassword`.
Use the following command to display this password:
`[ec2-user ~]$ sudo cat /var/lib/jenkins/secrets/initialAdminPassword`
3. The Jenkins installation script directs you to the **Customize Jenkins** page. Click **Install suggested plugins**.
4. Once the installation is complete, the **Create First Admin User** will open. Enter your information, and then select **Save and Continue**.

Getting Started

Create First Admin User

Username:

Password:

Confirm password:

Full name:

E-mail address:

Jenkins 2.263.1 Skip and continue as admin Save and Continue

- On the left-hand side, select **Manage Jenkins**, and then select **Manage Plugins**.
- Select the **Available** tab, and then enter **Amazon EC2 plugin** at the top right.
- Select the checkbox next to **Amazon EC2 plugin**, and then select **Install without restart**.

Install	Name	Released
<input checked="" type="checkbox"/>	Amazon EC2 1.68 Cloud Providers Cluster Management Agent Management spotinst aws	3 mo 15 days ago
<input type="checkbox"/>	Amazon Elastic Container Service (ECS) / Fargate 1.41 Cluster Management Agent Management aws	3 mo 11 days ago
<input type="checkbox"/>	Amazon EC2 Container Service plugin with autoscaling capabilities 1.0 Cluster Management Agent Management	6 yr 0 mo ago

[Install without restart](#) [Download now and install after restart](#) Update information obtained: 1 hr 51 min ago [Check now](#)

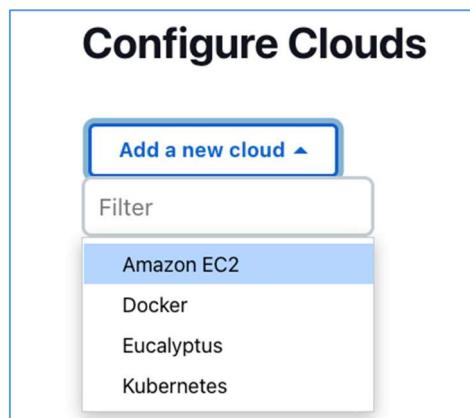
- Once the installation is done, select **Back to Dashboard**.
- Select **Configure a cloud** if there are no existing nodes or clouds. If you already have other nodes or clouds set up, select **Manage Jenkins**.

- After navigating to **Manage Jenkins**, select **Configure Nodes and Clouds** from the left hand side of the page.

11. From here, select **Configure Clouds**.



12. Select **Add a new cloud**, and select **Amazon EC2**. A collection of new fields appears.



13. Click **Add** under Amazon EC2 Credentials

A screenshot of the 'Configure Clouds' dialog box for 'Amazon EC2'. It shows a form with a 'Name' field containing 'JenkinsServer'. Below the form is a section titled 'Amazon EC2 Credentials ?' with a note about AWS IAM Access Key. At the bottom of the dialog is a red-bordered 'Add' button.

From the Jenkins Credentials Provider, select AWS Credentials as the Kind.

Jenkins Credentials Provider: Jenkins

Add Credentials

Domain

Kind

Scope ?

ID ?

Scroll down and enter in the IAM User programmatic access keys with permissions to launch EC2 instances and select **Add**.

ID ?

Description ?

Access Key ID ?

Secret Access Key

IAM Role Support

Advanced...

Add Cancel

Scroll down to select your region using the drop-down, and select **Add** for the EC2 Key Pair's Private Key.

Use EC2 instance profile to obtain credentials ?

Alternate EC2 Endpoint

The regions will be populated once the keys above are entered.

Region ?

us-east-1

EC2 Key Pair's Private Key ?

- none -

+ Add

No ssh credentials selected

From the Jenkins Credentials Provider, select SSH Username with private key as the Kind and set the Username to ec2-user.

The screenshot shows the Jenkins Credentials Provider configuration page. The 'Kind' field is set to 'SSH Username with private key'. The 'Username' field is set to 'ec2-user'. Other fields like 'Scope', 'ID', and 'Description' are empty or not highlighted.

Scroll down and select **Enter Directly** under Private Key, then select **Add**.

The screenshot shows the Jenkins Private Key configuration page. The 'Key' section has 'Enter directly' selected. The 'Add' button is highlighted with a red box.

Open the private key pair you created in the creating a key pair step and paste in the contents from "-----BEGIN RSA PRIVATE KEY-----" to "-----END RSA PRIVATE KEY-----". Select **Add** when completed.

The screenshot shows the Jenkins Private Key configuration page with a large text area containing a pasted private key. The 'Add' button is highlighted with a red box.

Scroll down to "Test Connection" and ensure it states "Success". Select **Save** when done

The screenshot shows the Jenkins Test Connection configuration page. The status message 'Success' is highlighted with a red box. The 'Save' button at the bottom left is highlighted with a red box.

2. Managing and installing 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.

Jenkins provides two methods for installing plugins on the controller:

- Using the "Plugin Manager" in the web UI.

The screenshot shows the Jenkins Manage Jenkins interface. On the left sidebar, under 'System Configuration', the 'Manage Jenkins' link is highlighted with a red box. In the main content area, there is a message about a new Jenkins version available for download, followed by a note about building on the built-in node. Below these are sections for 'Configure System', 'Global Tool Configuration', 'Manage Nodes and Clouds', and 'Manage Plugins'. The 'Manage Plugins' link is also highlighted with a red box.

The screenshot shows the Jenkins Plugin Manager page. The 'Available' tab is selected. A search bar contains the text 'Loca'. The results table lists three plugins: 'Localization: Chinese (Simplified)', 'Localization Support', and 'Locale'. The 'Localization Support' plugin is selected for installation, indicated by a checked checkbox. At the bottom, there are two buttons: 'Install without restart' and 'Download now and install after restart'. The status bar at the bottom right indicates 'Update information obtained: 7 hr 59 min'.

Install	Name	Released
<input type="checkbox"/>	Localization: Chinese (Simplified) 1.0.24 localization	2 yr 0 mo ago
<input checked="" type="checkbox"/>	Localization Support 1.2 Supporting infrastructure for standalone localization plugins. On its own, this plugin doesn't do a lot, it requires other plugins to work.	2 mo 24 days ago
<input type="checkbox"/>	Locale 226.v008e1b_58cb_b_0 localization User Interface	8 days 7 hr ago

b. Using the Jenkins CLI install-plugin command

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.

```
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.

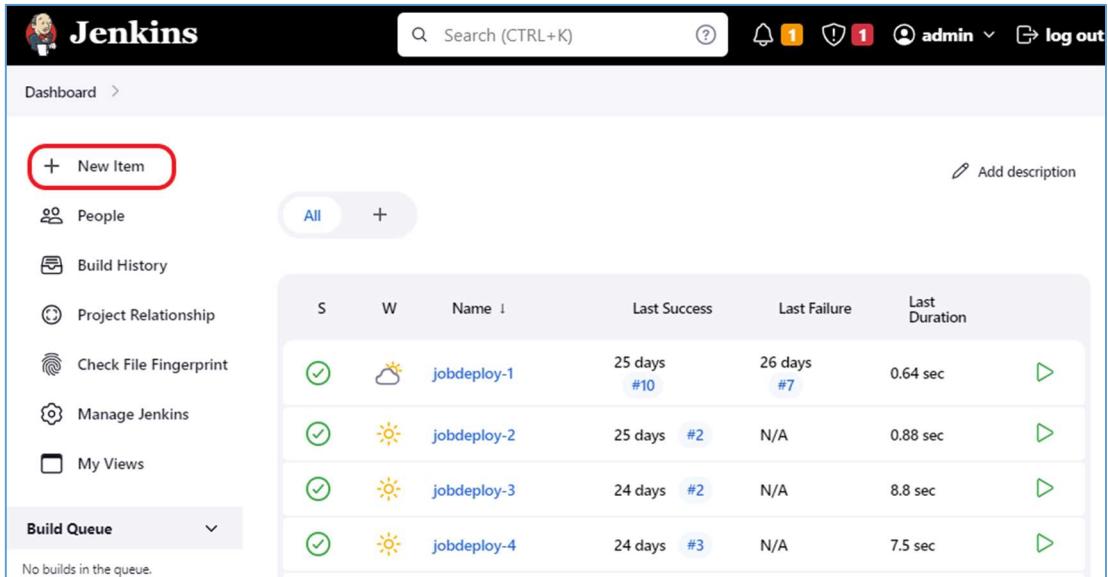
-deploy : Deploy plugins right away without postponing them until the reboot.

-name VAL : If specified, the plugin will be installed as this short name (whereas normally the name is inferred from the source name automatically).

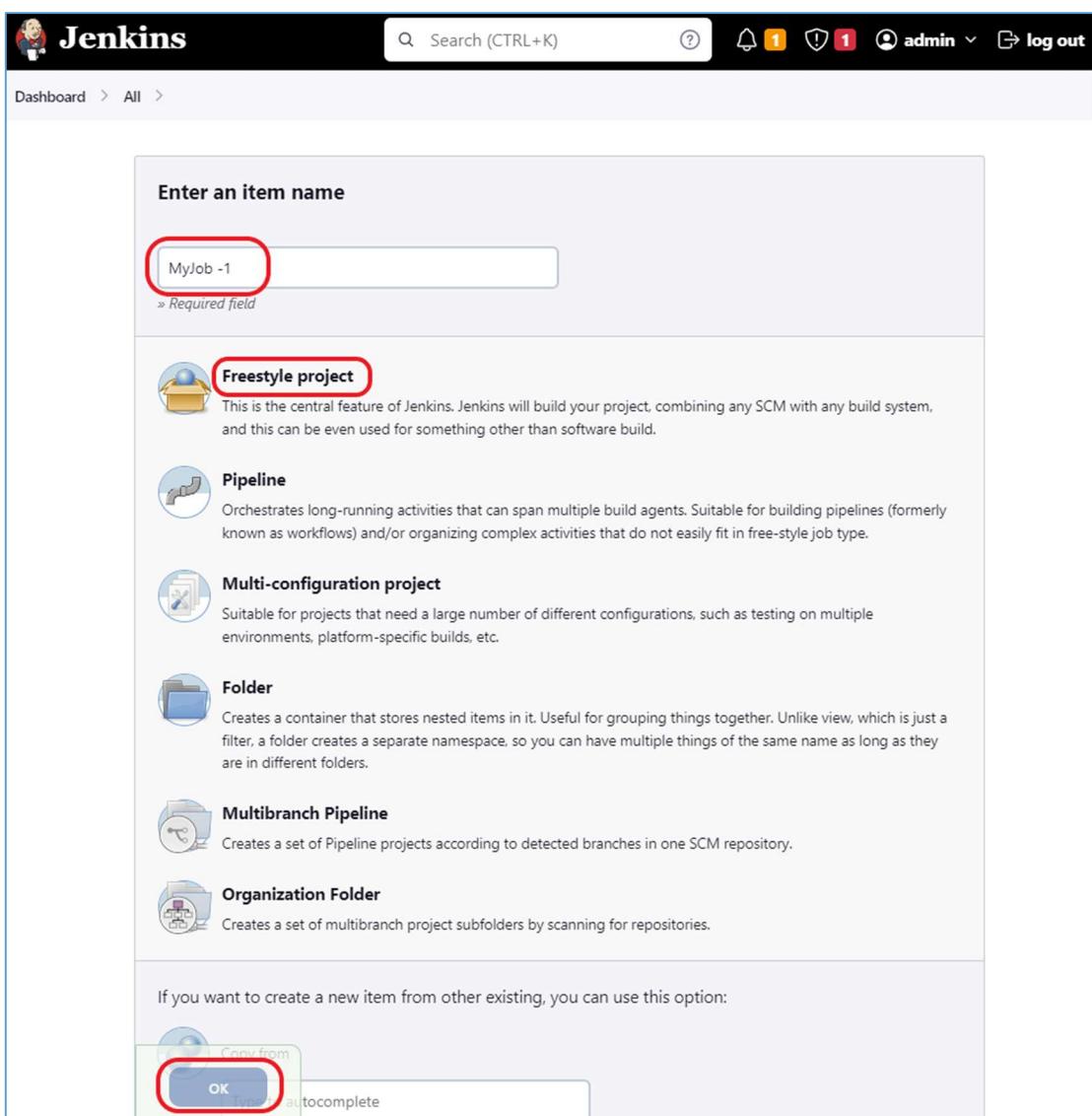
-restart : Restart Jenkins upon successful installation.

3. First simple job example in Jenkins.

a. Create new job.



The screenshot shows the Jenkins dashboard. At the top left, there's a 'New Item' button with a plus sign inside a red rounded rectangle. To its right is a search bar labeled 'Search (CTRL+K)'. On the far right, there are icons for help, notifications (1), security (1), user 'admin', and 'log out'. Below the header, the 'Dashboard' link is visible. On the left sidebar, there are links for 'People', 'Build History', 'Project Relationship', 'Check File Fingerprint', 'Manage Jenkins', and 'My Views'. A dropdown menu for 'Build Queue' is open, showing four build entries: 'jobdeploy-1' (25 days, #10), 'jobdeploy-2' (25 days, #2), 'jobdeploy-3' (24 days, #2), and 'jobdeploy-4' (24 days, #3). Each entry includes a green checkmark icon, a weather-like icon, the job name, the last success date, the last failure date, and the duration. To the right of the sidebar is a table with columns: S, W, Name, Last Success, Last Failure, and Last Duration.



The screenshot shows the 'Enter an item name' dialog box. In the input field, 'MyJob -1' is typed, with a red box highlighting the input field. Below the input field, a note says 'Required field'. There are several project types listed with their descriptions and icons:

- Freestyle project** (highlighted with a red box): 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.
- Multibranch Pipeline**: Creates a set of Pipeline projects according to detected branches in one SCM repository.
- Organization Folder**: Creates a set of multibranch project subfolders by scanning for repositories.

At the bottom of the dialog, there's a note: 'If you want to create a new item from other existing, you can use this option:' followed by a 'Copy from' button with a red box around it. Below that is an 'OK' button with a red box around it, and a text input field containing 'Type / to autocomplete'.

Jenkins

Search (CTRL+K)

admin log out

Dashboard > MyJob-1 > Configuration

Configure General

Enabled

General

Description: My first jenkins job

[Plain text] [Preview](#)

Commit agent's Docker container ?

Define a Docker template

Discard old builds ?

Strategy

Log Rotation

Days to keep builds: if not empty, build records are only kept up to this number of days

Max # of builds to keep: if not empty, only up to this number of build records are kept

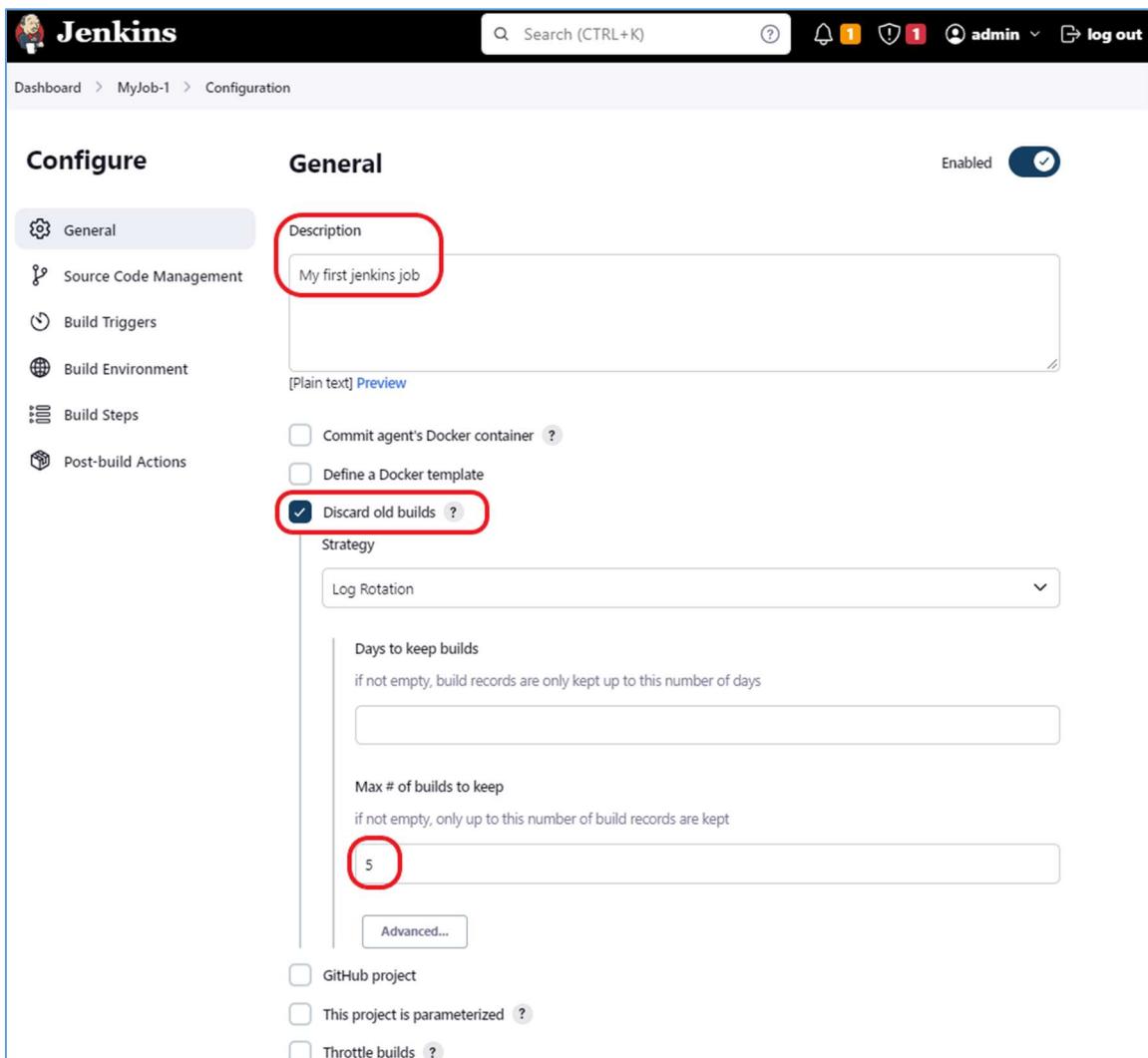
5

[Advanced...](#)

GitHub project

This project is parameterized ?

Throttle builds ?



Build Steps

Execute shell

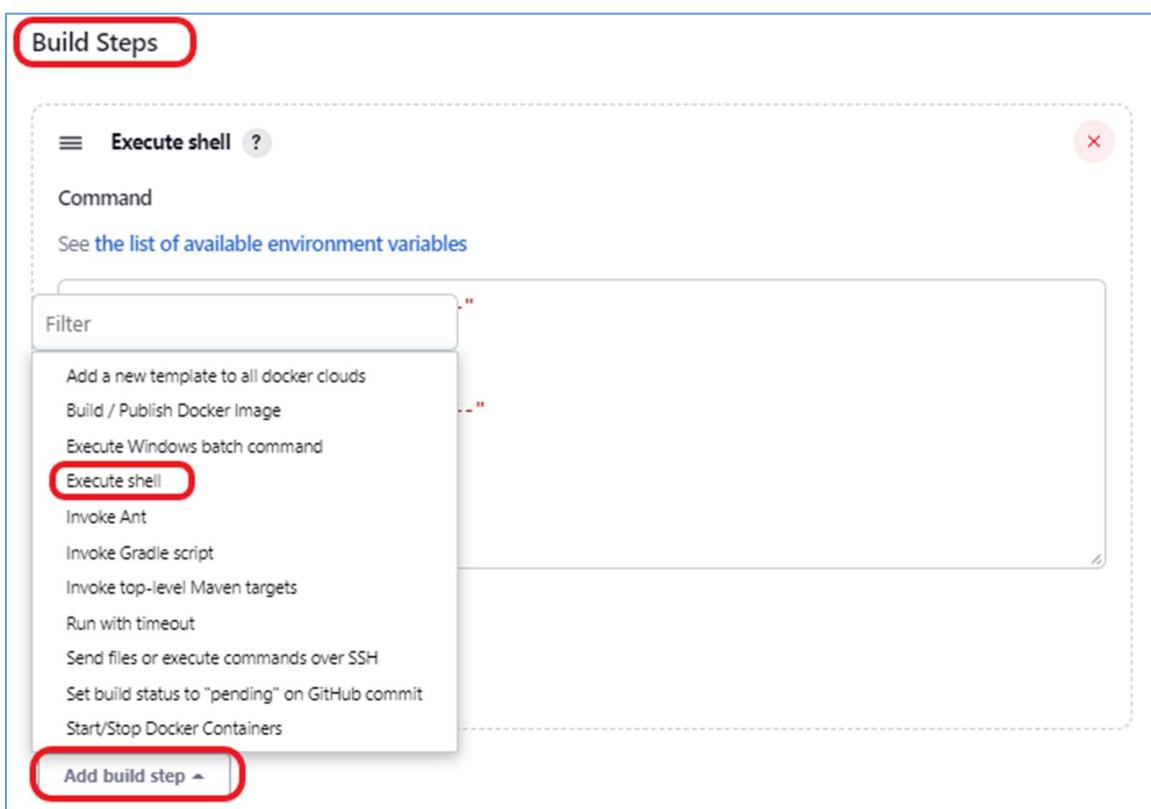
Command

See the list of available environment variables

Filter

- Add a new template to all docker clouds
- Build / Publish Docker Image
- Execute Windows batch command
- Execute shell**
- Invoke Ant
- Invoke Gradle script
- Invoke top-level Maven targets
- Run with timeout
- Send files or execute commands over SSH
- Set build status to "pending" on GitHub commit
- Start/Stop Docker Containers

[Add build step](#)



Build Steps

Execute shell

Command

See the list of available environment variables

```
echo "-----BEGINING-----"
echo "Die Hard !!!"
hostnamectl status
echo "-----FINISHING-----"
echo "Build number # "
echo $BUILD_NUMBER
# cat file.txt
```

Advanced...

Add build step ▾

Post-build Actions

Add post-build action ▾

Save **Apply**

b. Build job.

Jenkins

Dashboard >

+ New Item All +

People Build History Project Relationship Check File Fingerprint Manage Jenkins My Views

Build Queue No builds in the queue.

Build Executor Status 1 Idle 2 Idle Icon: S M L

S	W	Name ↓	Last Success	Last Failure	Last Duration
✓	cloud	jobdeploy-1	26 days #10	26 days #7	0.64 sec
✓	sun	jobdeploy-2	26 days #2	N/A	0.88 sec
✓	sun	jobdeploy-3	24 days #2	N/A	8.8 sec
✓	sun	jobdeploy-4	24 days #3	N/A	7.5 sec
⚠	sun	jobdeploy-5	4 days 6 hr #4	N/A	10 sec
✓	sun	MyJob-1	27 days #6	N/A	41 ms
✓	cloud		25 days #6	25 days #3	6.6 sec

Changes Workspace Build Now Configure Delete Project Rename

Atom feed for all Atom feed for failures Atom feed for just latest builds

The screenshot shows the Jenkins Project MyJob-1 dashboard. On the left, there's a sidebar with various options: Status, Changes, Workspace, Build Now (which is highlighted with a red box), Configure, Delete Project, and Rename. Below this is a 'Build History' section with tabs for 'Build History' (selected) and 'trend'. A search bar says 'Filter builds...'. The build history lists five builds (#6, #5, #4, #3, #2) with green checkmarks, each with a timestamp. At the bottom of the history section are links for 'Atom feed for all' and 'Atom feed for failures'. To the right of the sidebar, the main content area has a title 'Project MyJob-1' and a note 'My first jenkins job' (also highlighted with a red box). Below the note is a 'Permalinks' section with a bulleted list of recent builds.

Project MyJob-1

My first jenkins job

Build Now

Permalinks

- Last build (#6), 27 days ago
- Last stable build (#6), 27 days ago
- Last successful build (#6), 27 days ago
- Last completed build (#6), 27 days ago

c. Console output.

The screenshot shows the Jenkins Build #2 details page. The top navigation bar includes a 'Dashboard' link, a 'MyJob-1' link (highlighted with a red box), and a '#2' link. The main content area starts with a large green checkmark icon and the text 'Build #2 (Nov 17, 2022, 7:58:51 PM)'. There are buttons for 'Keep this build forever', 'Add description', and links for 'Started 27 days ago' and 'Took 18 ms'. On the left, a sidebar lists options: Status, Changes, Console Output (highlighted with a red box), Edit Build Information, Delete build '#2', Timings, and Next Build. The main content area also includes sections for 'No changes.', 'Started by user admin', 'This run spent:', and a bulleted list of execution times.

Build #2 (Nov 17, 2022, 7:58:51 PM)

Keep this build forever

Started 27 days ago
Took 18 ms

Add description

No changes.

Started by user admin

This run spent:

- 22 ms waiting;
- 18 ms build duration;
- 40 ms total from scheduled to completion.

Jenkins

Search (CTRL+K)

admin log out

Dashboard > MyJob-1 > #2

Status Changes Console Output (highlighted with a red box)

View as plain text Edit Build Information Delete build '#2' Timings Next Build

Console Output

```
Started by user admin
Running as SYSTEM
Building in workspace /var/lib/jenkins/workspace/MyJob-1
[MyJob-1] $ /bin/sh -xe /tmp/jenkins6264323146518311264.sh
+ echo -----BEGINING-----
-----BEGINING-----
+ echo 'Die Hard !!!'
Die Hard !!!
+ hostnamectl status
  Static hostname: vm2
    Icon name: computer-vm
    Chassis: vm
    Machine ID: eba3a15dc4245b4b8e481959d8c90101
    Boot ID: 0f8575a335a046798d3eb7065784b143
    Virtualization: kvm
    Operating System: CentOS Linux 7 (Core)
      CPE OS Name: cpe:/o:centos:centos:7
      Kernel: Linux 3.10.0-1160.80.1.el7.x86_64
      Architecture: x86-64
+ echo -----FINISHING-----
-----FINISHING-----
+ echo 'Build number # '
Build number #
+ echo 2
2
Finished: SUCCESS
```

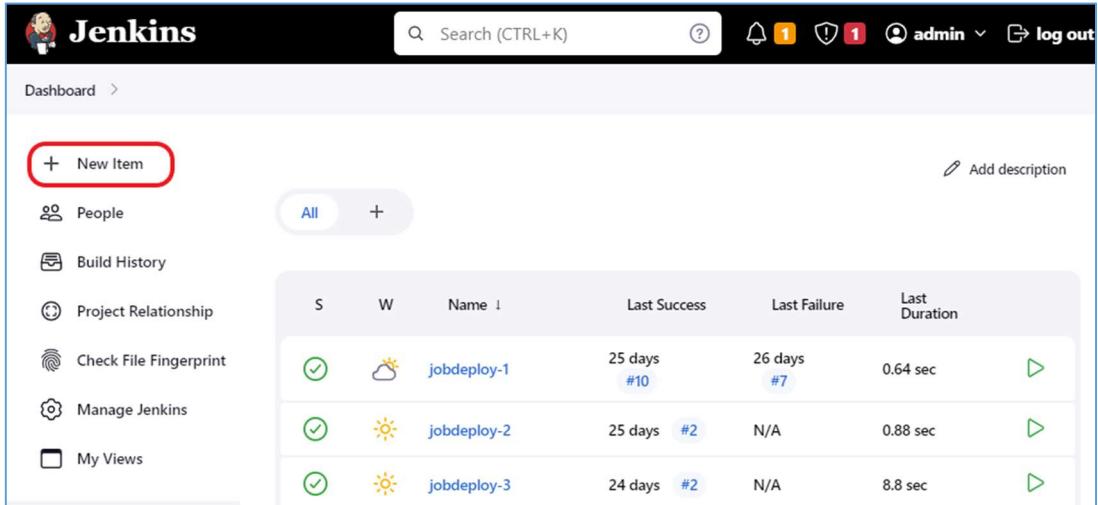
```
Last login: Thu Dec 15 13:32:36 2022 from 10.0.2.2
[vagrant@vm2 ~]$ 
[vagrant@vm2 ~]$ 
[vagrant@vm2 ~]$ cd /var/lib/jenkins/workspace/ (highlighted with a red box)
[vagrant@vm2 workspace]$ 
[vagrant@vm2 workspace]$ 
[vagrant@vm2 workspace]$ ll
total 0
drwxr-xr-x. 2 jenkins jenkins 24 Nov 19 07:15 jobdeploy-1
drwxr-xr-x. 2 jenkins jenkins 24 Nov 19 14:36 jobdeploy-2
drwxr-xr-x. 3 jenkins jenkins 53 Nov 20 18:04 jobdeploy-3
drwxr-xr-x. 2 jenkins jenkins 6 Nov 21 10:58 jobdeploy-3@tmp
drwxr-xr-x. 3 jenkins jenkins 53 Nov 21 12:20 jobdeploy-4
drwxr-xr-x. 2 jenkins jenkins 6 Nov 21 12:20 jobdeploy-4@tmp
drwxr-xr-x. 3 jenkins jenkins 53 Dec 11 11:58 jobdeploy-5
drwxr-xr-x. 2 jenkins jenkins 6 Dec 11 11:58 jobdeploy-5@tmp
drwxr-xr-x. 2 jenkins jenkins 6 Nov 17 19:58 MyJob-1 (highlighted with a red box)
drwxr-xr-x. 3 jenkins jenkins 53 Nov 20 17:11 mypipeline_example-1
drwxr-xr-x. 2 jenkins jenkins 6 Nov 20 16:54 mypipeline_example-1@tmp
```

```
[vagrant@vm2 jenkins]$ cd jobs/
[vagrant@vm2 jobs]$
[vagrant@vm2 jobs]$
[vagrant@vm2 jobs]$ ll
total 0
drwxr-xr-x. 3 jenkins jenkins 61 Nov 19 14:03 jobdeploy-1
drwxr-xr-x. 3 jenkins jenkins 61 Nov 19 14:40 jobdeploy-2
drwxr-xr-x. 3 jenkins jenkins 61 Nov 21 10:58 jobdeploy-3
drwxr-xr-x. 3 jenkins jenkins 61 Nov 21 12:20 jobdeploy-4
drwxr-xr-x. 3 jenkins jenkins 84 Dec 11 11:58 jobdeploy-5
drwxr-xr-x. 3 jenkins jenkins 61 Nov 18 16:39 MyJob-1
drwxr-xr-x. 3 jenkins jenkins 61 Nov 21 10:57 mypipeline_example-1
[vagrant@vm2 jobs]$ cd MyJob-1/
[vagrant@vm2 MyJob-1]$
[vagrant@vm2 MyJob-1]$ ll
total 8
drwxr-xr-x. 7 jenkins jenkins 86 Nov 18 16:56 builds
-rw-r--r--. 1 jenkins jenkins 1202 Nov 18 16:39 config.xml
-rw-r--r--. 1 jenkins jenkins 2 Nov 17 20:02 nextBuildNumber
[vagrant@vm2 MyJob-1]$
[vagrant@vm2 MyJob-1]$ cd builds/
[vagrant@vm2 builds]$
[vagrant@vm2 builds]$ ll
total 4
drwxr-xr-x. 2 jenkins jenkins 55 Nov 17 19:58 2
drwxr-xr-x. 2 jenkins jenkins 55 Nov 17 19:58 3
drwxr-xr-x. 2 jenkins jenkins 55 Nov 17 19:59 4
drwxr-xr-x. 2 jenkins jenkins 55 Nov 17 19:59 5
drwxr-xr-x. 2 jenkins jenkins 55 Nov 17 20:02 6
-rw-r--r--. 1 jenkins jenkins 0 Nov 17 19:54 legacyIds
-rw-r--r--. 1 jenkins jenkins 126 Nov 17 20:02 permalinks
[vagrant@vm2 builds]$
```

```
[vagrant@vm2 builds]$ cd 2/
[vagrant@vm2 2]$
[vagrant@vm2 2]$ ll
total 12
-rw-r--r--. 1 jenkins jenkins 1220 Nov 17 19:58 build.xml
-rw-r--r--. 1 jenkins jenkins 6 Nov 17 19:58 changelog.xml
-rw-r--r--. 1 jenkins jenkins 1064 Nov 17 19:58 log
[vagrant@vm2 2]$
```

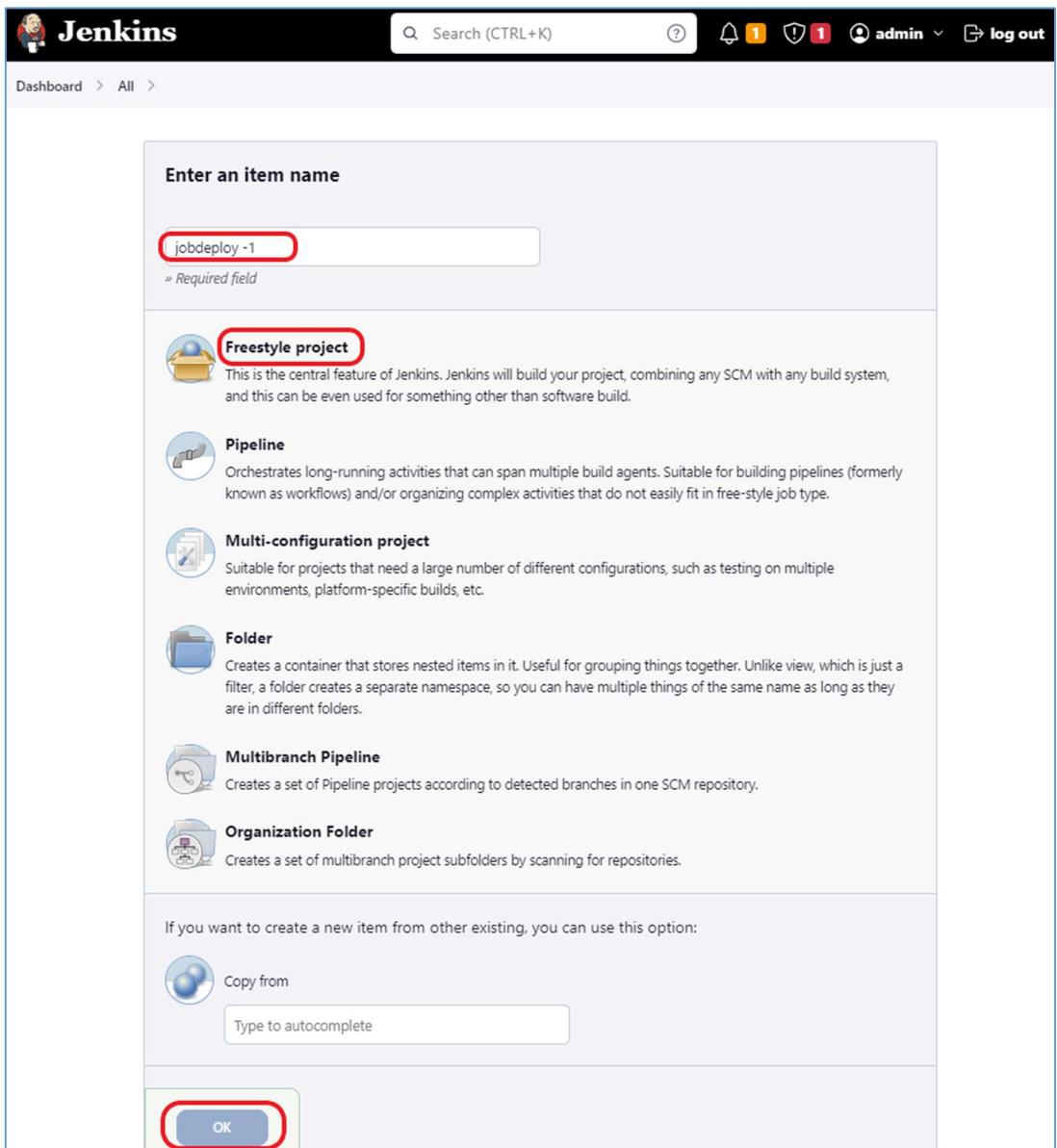
4. Simple job example in Jenkins with deploy.

a. Create new job.



The screenshot shows the Jenkins dashboard. On the left, there's a sidebar with links like 'Dashboard', 'New Item' (which is highlighted with a red box), 'People', 'Build History', 'Project Relationship', 'Check File Fingerprint', 'Manage Jenkins', and 'My Views'. The main area displays a table of existing Jenkins items:

S	W	Name	Last Success	Last Failure	Last Duration
✓	Cloud icon	jobdeploy-1	25 days #10	26 days #7	0.64 sec
✓	Sun icon	jobdeploy-2	25 days #2	N/A	0.88 sec
✓	Sun icon	jobdeploy-3	24 days #2	N/A	8.8 sec



The screenshot shows the 'Enter an item name' dialog. The input field contains 'jobdeploy -1' (highlighted with a red box). Below the input field, it says '» Required field'. A list of project types is shown:

- Freestyle project** (highlighted with a red box)
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.
- Multibranch Pipeline**
Creates a set of Pipeline projects according to detected branches in one SCM repository.
- Organization Folder**
Creates a set of multibranch project subfolders by scanning for repositories.

If you want to create a new item from other existing, you can use this option:

Copy from
Type to autocomplete

OK

The screenshot shows the Jenkins 'Configuration' page for a job named 'jobdeploy-1'. The 'General' tab is selected. A red box highlights the 'Description' field, which contains the text 'First job with deploy'. Another red box highlights the 'Discard old builds' section, specifically the 'Strategy' dropdown set to 'Log Rotation'. A third red box highlights the 'Max # of builds to keep' field, which is set to '5'.

b. Establish SSH connection with or without login/password.

- Generate ssh key-pair

```
$ ssh-keygen -t rsa -f "key_name"
```

- Copy public key to the remote host where you will deploy app

```
$ ssh-copy-id "user_name@host_name"
```

- We have to use **-o StrictHostKeyChecking=no** to deploy index.html to the remote host without login/password

```
$ scp -v -o StrictHostKeyChecking=no index.html "user_name@host_name":/var/www/html
```

instead of preconfigured fingerprint:

```
$ scp -v index.html "user_name@host_name":/var/www/html
```

- We have to copy file "**id_rsa**"(chmod 400) to **/var/lib/jenkins/.ssh**
- The **/var/lib/jenkins/.ssh** directory and files inside of it should be owned by **jenkins**

c. Install Apache server and establish privileges for /var/www.

- Install Apache on remote host where you will deploy app (example Ubuntu 20.04)

```
$ sudo apt update  
$ sudo apt install apache2  
$ sudo ufw allow 'Apache'  
$ sudo ufw status  
$ sudo systemctl status apache2
```

- Establish privileges for /var/www

```
vagrant@vm1:/var/www/html$ sudo su  
root@vm1:/var/www/html#  
root@vm1:/var/www/html# cd /var/www/  
root@vm1:/var/www# ls -lh  
total 4.0K  
drwxrwxr-x 2 root vagrant 4.0K Nov 19 10:25 html  
root@vm1:/var/www# sudo chmod -R g+rw /var/www/  
root@vm1:/var/www#  
root@vm1:/var/www# ls -la  
total 12  
drwxrwxr-x 3 root vagrant 4096 Nov 11 16:19 .  
drwxr-xr-x 14 root root 4096 Nov 11 16:19 ..  
drwxrwxr-x 2 root vagrant 4096 Nov 19 10:25 html  
root@vm1:/var/www# cd html/  
root@vm1:/var/www/html#  
root@vm1:/var/www/html# ls -la  
total 24  
drwxrwxr-x 2 root vagrant 4096 Nov 19 10:25 .  
drwxrwxr-x 3 root vagrant 4096 Nov 11 16:19 ..  
-rw-rw-r-- 1 vagrant vagrant 2918 Nov 21 12:20 index.html  
-rwxrwxr-x 1 root vagrant 10918 Nov 19 07:32 index_old.html  
root@vm1:/var/www/html# sudo chgrp -R vagrant /var/www/  
root@vm1:/var/www/html# sudo chmod -R g+rw /var/www/  
root@vm1:/var/www/html#  
root@vm1:/var/www/html# ls -la  
total 24  
drwxrwxr-x 2 root vagrant 4096 Nov 19 10:25 .  
drwxrwxr-x 3 root vagrant 4096 Nov 11 16:19 ..  
-rw-rw-r-- 1 vagrant vagrant 2918 Nov 21 12:20 index.html  
-rwxrwxr-x 1 root vagrant 10918 Nov 19 07:32 index_old.html  
root@vm1:/var/www/html# _
```

d. Build Steps in Jenkins.

- Find out IP and hostname of the remote host(webserver Ubuntu 20.04)

```
vagrant@vm1:/var/www/html$ hostname -I  
10.0.2.15 192.168.1.113  
vagrant@vm1:/var/www/html$ whoami  
vagrant  
vagrant@vm1:/var/www/html$
```

- Copy the file “index.html” to the remote host via ssh

Build Steps

Execute shell

Command

See the list of available environment variables

```
echo "start"  
cat <<EOF > index.html ← index.html  
<html>  
<head>  
    <title>TEST</title>  
</head>  
<body bgcolor=green>  
    <p style = "color: red; text-align: center; font-size: 100px;">Hello World!</p>  
</body>  
</html>  
EOF  
echo "finish"  
ls -la  
echo "-----deploy-----"  
# copy file "index.html" to remote host over ssh.  
scp -v /var/lib/jenkins/workspace/jobdeploy-1/index.html vagrant@192.168.1.113:/var/www/html  
  
# copy file "index.html" to remote host over ssh and without checking key.  
# scp -v -o StrictHostKeyChecking=no index.html vagrant@192.168.1.113:/var/www/html
```

Advanced...

e. Build job.

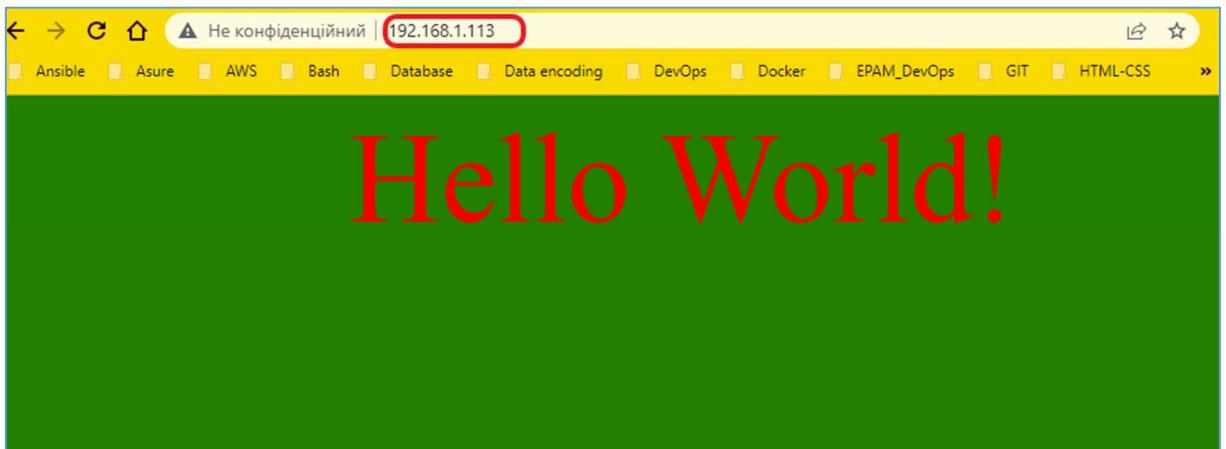
- Console output

The screenshot shows the Jenkins interface for a build named 'jobdeploy-1' (run #10). The 'Console Output' tab is selected. The log output shows the deployment process:

```
Started by user admin
Running as SYSTEM
Building in workspace /var/lib/jenkins/workspace/jobdeploy-1
[jobdeploy-1] $ /bin/sh -xe /tmp/jenkins14203991903844644727.sh
+ echo start
start
+ cat
+ echo finish
finish
+ ls -la
total 4
drwxr-xr-x. 2 jenkins jenkins 24 Nov 19 07:15 .
drwxr-xr-x. 4 jenkins jenkins 40 Nov 19 07:15 ..
-rw-r--r--. 1 jenkins jenkins 167 Nov 19 14:03 index.html
+ echo -----deploy-----
-----deploy-----
+ scp -v /var/lib/jenkins/workspace/jobdeploy-1/index.html vagrant@192.168.1.113:/var/www/html
Executing: program /usr/bin/ssh host 192.168.1.113, user vagrant, command scp -v -t /var/www/html
OpenSSH_7.4p1, OpenSSL 1.0.2k-fips 26 Jan 2017
```

At the bottom of the log, there is a red box highlighting the 'Finished: SUCCESS' message.

- Checking our website



5. Simple job example in Jenkins with deploy using plugin “Publish over SSH”.

a. Configuration and settings of the plugin “Publish over SSH”.

The screenshot shows the Jenkins Manage Jenkins configuration page. The left sidebar has a 'Manage Jenkins' section selected. The main area displays the 'System Configuration' section with a 'Configure System' button highlighted by a red box. Below it is a 'Global Tool Configuration' section. The 'Publish over SSH' section is also highlighted with a red box. It contains fields for 'Jenkins SSH Key', 'Passphrase', 'Path to key', and a 'Key' input field containing an RSA private key. A 'Disable exec' checkbox is present. The 'SSH Servers' section is highlighted with a red box and contains a configuration for a server named 'vm1_ubuntu20.04_webserver' with host '192.168.1.113', user 'vagrant', remote directory '/var/www/html', and an 'Advanced...' button. A 'Success' message is shown at the bottom left, and a 'Test Configuration' button is highlighted with a red box at the bottom right.

New version of Jenkins (2.382) is available for download (changelog).

Building on the built-in node can be a security issue. You should set up distributed builds. See the documentation.

Set up agent Set up cloud Dismiss

System Configuration

Configure System Configure global settings and paths.

Global Tool Configuration Configure tools, their locations and automatic

Publish over SSH

Jenkins SSH Key ?

Passphrase ?

Concealed Change Password

Path to key ?

Key ?

```
-----BEGIN RSA PRIVATE KEY-----
MIIEpAIBAAQCAQEAwuGmMpHsb8o+0XVb17D3d51VedbZSJvhW76EB0c7qK8u7+F2
vdMt5UEdrSXjgt8e5ICmwYJOCHum3zfvVta2mWG6IX40C01BpUTSBw==
-----END RSA PRIVATE KEY-----
```

Disable exec ?

SSH Servers

SSH Server Name ?

vm1_ubuntu20.04_webserver

Hostname ?

192.168.1.113

Username ?

vagrant

Remote Directory ?

/var/www/html

Advanced...

Success

Test Configuration

* User (vagrant) must have access to the directory (/var/www/html)

b. Create new job.

The screenshot shows the Jenkins dashboard with two jobs listed:

S	W	Name	Last Success	Last Failure	Last Duration
✓	☀️	jobdeploy-1	22 min #12	N/A	0.48 sec
✓	☀️	jobdeploy-2	27 days #2	N/A	0.88 sec

Configuration for jobdeploy-2:

General Tab:

- Description: first job with deploy and with plugin Publish Over SSH
- Discard old builds

Build Steps Tab:

Execute shell:

```
echo "start"
cat <<EOF > index.html
<html>
<head>
    <title>TEST</title>
</head>
<body bgcolor=green>
    <p style = "color: red; text-align: center; font-size: 100px;">Hello World!</p>
    <p style = "color: yellow; text-align: center; font-size: 70px;">Deployed by Publish Over SSH!</p>
</body>
</html>
EOF
echo "finish"
ls -la
echo "-----deploy-----"
```

- c. Establish **SSH** connection using plugin “**Publish over SSH**”.

In the SSH Server Name section use the ssh server name which you configured earlier.

The screenshot shows a configuration dialog for "Send files or execute commands over SSH".

SSH Publishers

SSH Server Name: vm1_ubuntu20.04_webserver

Transfers

Transfer Set

Source files: index.html

Remote directory: (empty)

Exec command: echo \$BUILD_ID

d. Install Apache server and establish privileges for /var/www.

- Install **Apache** on remote host where you will deploy app (example Ubuntu 20.04)

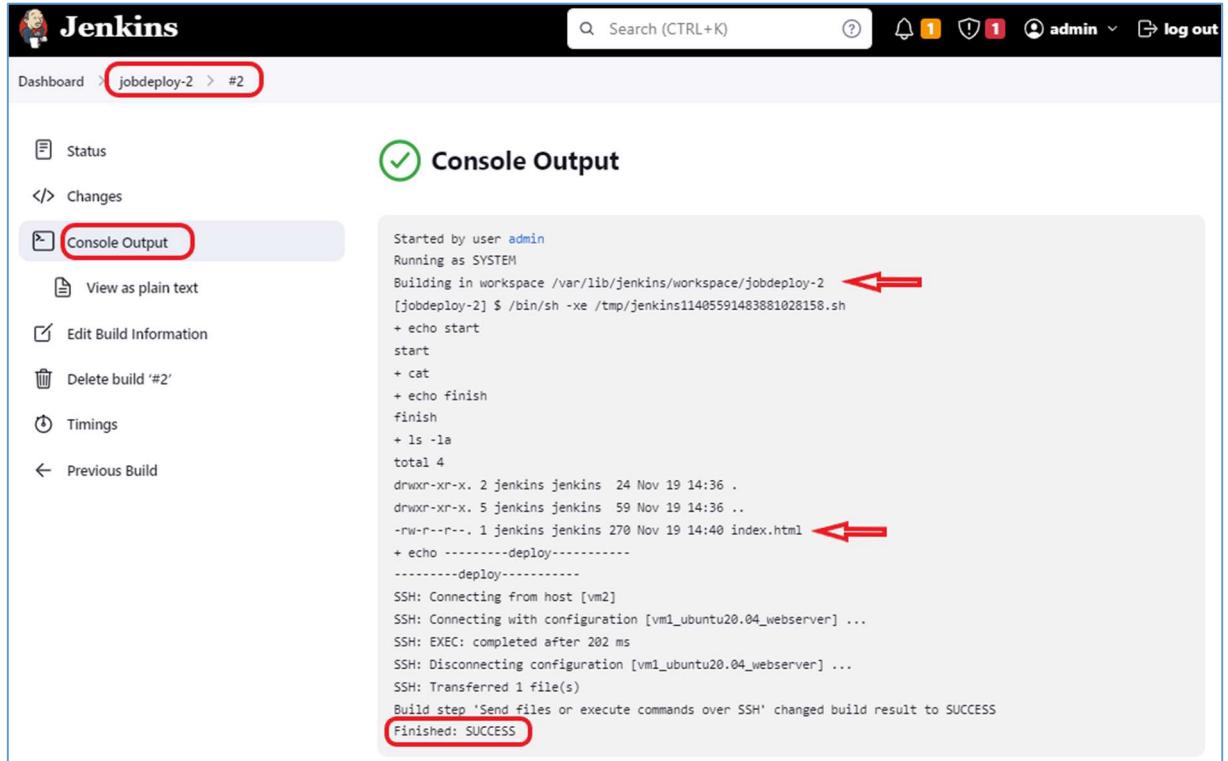
```
$ sudo apt update  
$ sudo apt install apache2  
$ sudo ufw allow 'Apache'  
$ sudo ufw status  
$ sudo systemctl status apache2
```

- Establish privileges for **/var/www**

```
vagrant@vm1:/var/www/html$ sudo su  
root@vm1:/var/www/html#  
root@vm1:/var/www/html# cd /var/www/  
root@vm1:/var/www# ls -lh  
total 4.0K  
drwxrwxr-x 2 root vagrant 4.0K Nov 19 10:25 html  
root@vm1:/var/www# sudo chmod -R g+rw /var/www/  
root@vm1:/var/www#  
root@vm1:/var/www# ls -la  
total 12  
drwxrwxr-x 3 root vagrant 4096 Nov 11 16:19 .  
drwxr-xr-x 14 root root 4096 Nov 11 16:19 ..  
drwxrwxr-x 2 root vagrant 4096 Nov 19 10:25 html  
root@vm1:/var/www# cd html/  
root@vm1:/var/www/html#  
root@vm1:/var/www/html# ls -la  
total 24  
drwxrwxr-x 2 root vagrant 4096 Nov 19 10:25 .  
drwxrwxr-x 3 root vagrant 4096 Nov 11 16:19 ..  
-rw-rw-r-- 1 vagrant vagrant 2918 Nov 21 12:20 index.html  
-rwxrwxr-x 1 root vagrant 10918 Nov 19 07:32 index_old.html  
root@vm1:/var/www/html# sudo chgrp -R vagrant /var/www/  
root@vm1:/var/www/html# sudo chmod -R g+rw /var/www/  
root@vm1:/var/www/html#  
root@vm1:/var/www/html# ls -la  
total 24  
drwxrwxr-x 2 root vagrant 4096 Nov 19 10:25 .  
drwxrwxr-x 3 root vagrant 4096 Nov 11 16:19 ..  
-rw-rw-r-- 1 vagrant vagrant 2918 Nov 21 12:20 index.html  
-rwxrwxr-x 1 root vagrant 10918 Nov 19 07:32 index_old.html  
root@vm1:/var/www/html# _
```

e. Build job.

- Console output



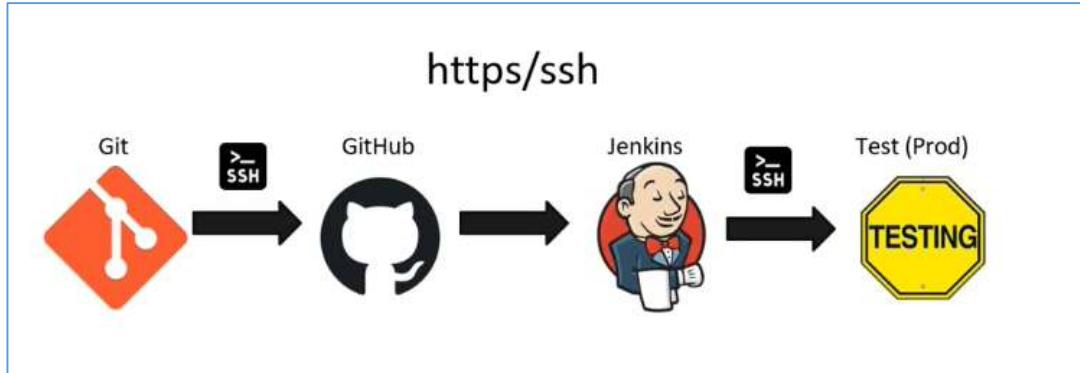
The screenshot shows the Jenkins interface for a build job named 'jobdeploy-2'. The 'Console Output' tab is selected, displaying the command-line logs of the build process. Red arrows highlight two specific lines: one pointing to the command 'index.html' being deployed, and another pointing to the final status message 'Finished: SUCCESS'.

```
Started by user admin
Running as SYSTEM
Building in workspace /var/lib/jenkins/workspace/jobdeploy-2
[jobdeploy-2] $ /bin/sh -xe /tmp/jenkins11405591483881028158.sh
+ echo start
start
+ cat
+ echo finish
finish
+ ls -la
total 4
drwxr-xr-x. 2 jenkins jenkins 24 Nov 19 14:36 .
drwxr-xr-x. 5 jenkins jenkins 59 Nov 19 14:36 ..
-rw-r--r--. 1 jenkins jenkins 270 Nov 19 14:40 index.html
+ echo -----deploy-----
-----deploy-----
SSH: Connecting from host [vm2]
SSH: Connecting with configuration [vml_ubuntu20.04_webserver] ...
SSH: EXEC: completed after 202 ms
SSH: Disconnecting configuration [vml_ubuntu20.04_webserver] ...
SSH: Transferred 1 file(s)
Build step 'Send files or execute commands over SSH' changed build result to SUCCESS
Finished: SUCCESS
```

- Checking our website



6. Simple CI/CD pipeline example.



- a. Configuration and settings credentials GitHub.

- [Generate ssh key-pair](#) on your Linux host

```
$ ssh-keygen -t rsa -C "your_GitHub_email@example.com"
```

- [Adding your SSH key to the ssh-agent](#)

```
$ eval "$(ssh-agent -s)"  
$ ssh-add ~/.ssh/"key_name"
```

- Adding a new SSH public key to your GitHub account

```
$ clip < ~/.ssh/"public_key_name.pub"
```

* If clip isn't working, you can locate the hidden .ssh folder, open the file in your favorite text editor, and copy it to your clipboard.

Then go to the your GitHub account in section **Settings - SSH and GPG keys** and add new ssh public key. (*see next page*)

- Adding [github.com](#) to the `/var/lib/jenkins/.ssh/known_hosts`

```
$ sudo ssh-keyscan github.com >>  
/var/lib/jenkins/.ssh/known_hosts"
```

- Dashboard > Manage Jenkins > Configure Global Security > Git Host Key Verification Configuration > Host Key Verification Strategy > Known hosts file
 - We have to copy file "`key_name`"(chmod 400) to `/var/lib/jenkins/.ssh`
 - Testing your connection to the GitHub from **jenkins** user

```
$ ssh -T git@github.com
```

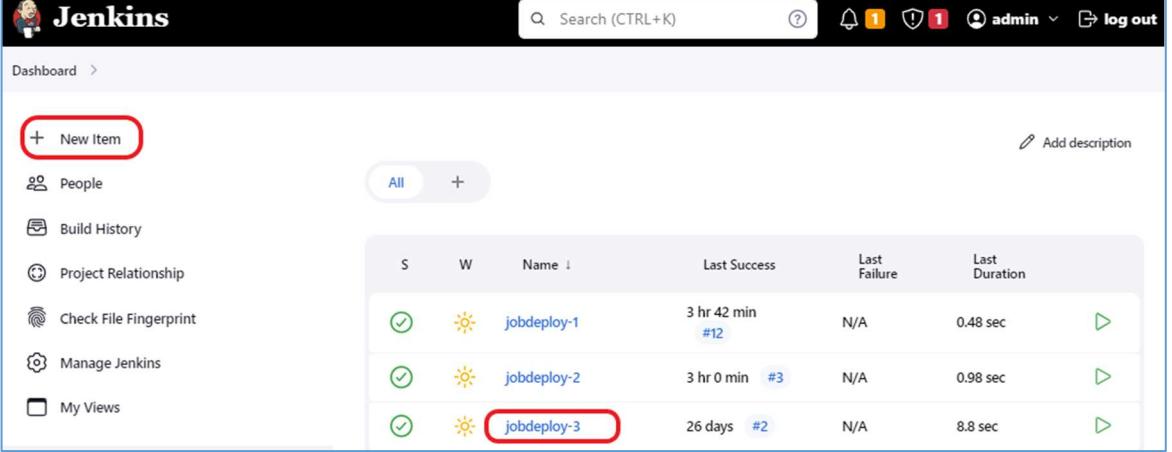
```
-bash-4.2$ ssh -T git@github.com  
Hi thestig1990! You've successfully authenticated, but GitHub does not provide shell access.  
-bash-4.2$
```

The screenshot shows a GitHub profile page for a user named 'thestig1990'. The profile picture is a Mars rover. The bio section contains a message about the Russian invasion of Ukraine. A yellow button labeled 'Help Ukraine Now →' is present. The sidebar on the right includes links for 'Your profile', 'Your repositories', 'Your projects', 'Your stars', 'Your gists', 'Your sponsors', 'Upgrade', 'Try Enterprise', 'Feature preview', 'Help', and 'Settings' (which is highlighted with a red box). Other options like 'Sign out' are also visible.

This screenshot shows the 'SSH keys / Add new' page. On the left, there's a sidebar with various account settings like 'Public profile', 'Account', 'Appearance', etc., and a 'SSH and GPG keys' section which is highlighted with a red box. The main area has fields for 'Title' (containing 'vm2-jenkins'), 'Key type' (set to 'Authentication Key'), and a 'Key' text area with a detailed explanation of the key format. A green 'Add SSH key' button is at the bottom.

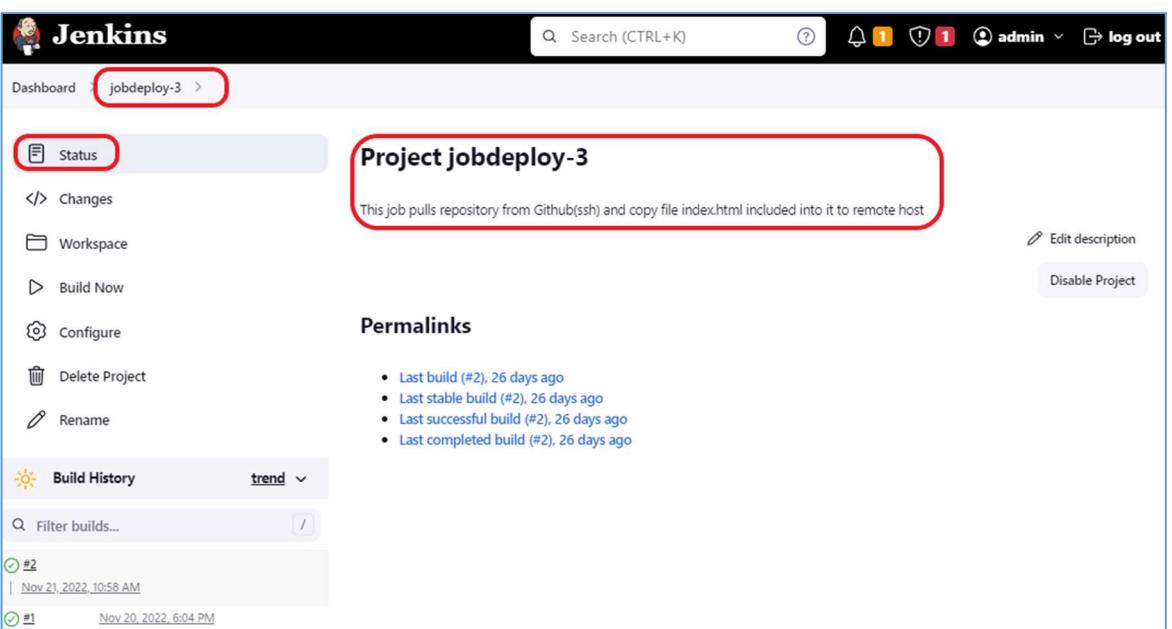
This screenshot shows the 'SSH keys' page. The 'SSH and GPG keys' section in the sidebar is highlighted with a red box. The main area lists two SSH keys: 'git_essential' and 'vm2-jenkins'. Both keys have a 'Delete' link next to them. A green 'New SSH key' button is located in the top right corner of the main content area.

b. Create new job.



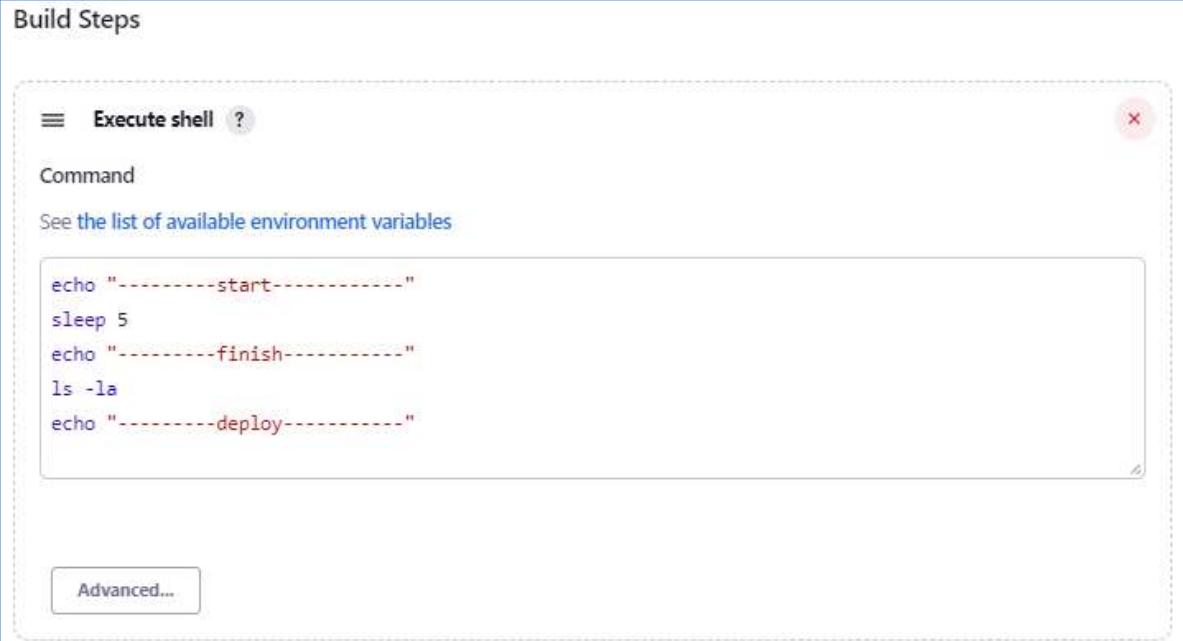
The screenshot shows the Jenkins dashboard. On the left, there's a sidebar with links like 'People', 'Build History', 'Project Relationship', 'Check File Fingerprint', 'Manage Jenkins', and 'My Views'. A red box highlights the '+ New item' button. On the right, there's a table listing existing Jenkins jobs:

S	W	Name	Last Success	Last Failure	Last Duration
✓	☀️	jobdeploy-1	3 hr 42 min #12	N/A	0.48 sec
✓	☀️	jobdeploy-2	3 hr 0 min #3	N/A	0.98 sec
✓	☀️	jobdeploy-3	26 days #2	N/A	8.8 sec



The screenshot shows the details for the 'jobdeploy-3' project. The 'Status' tab is selected. The main content area displays the project description: 'This job pulls repository from Github(ssh) and copy file index.html included into it to remote host'. Below this, there's a 'Permalinks' section with a bulleted list of recent builds. The 'Build History' section shows two builds: '#2' (Nov 21, 2022, 10:58 AM) and '#1' (Nov 20, 2022, 6:04 PM).

c. Build steps.



The screenshot shows the 'Build Steps' configuration for a Jenkins job. It includes an 'Execute shell' step with the following command:

```
echo "-----start-----"
sleep 5
echo "-----finish-----"
ls -la
echo "-----deploy-----"
```

Below the command, there's an 'Advanced...' button.

≡ Send files or execute commands over SSH ?

SSH Publishers

SSH Server

Name ?

Advanced...

Transfers

Transfer Set

Source files ?

Remove prefix ?

Remote directory ?

Exec command ?

All of the transfer fields (except for Exec timeout) support substitution of [Jenkins environment variables](#)

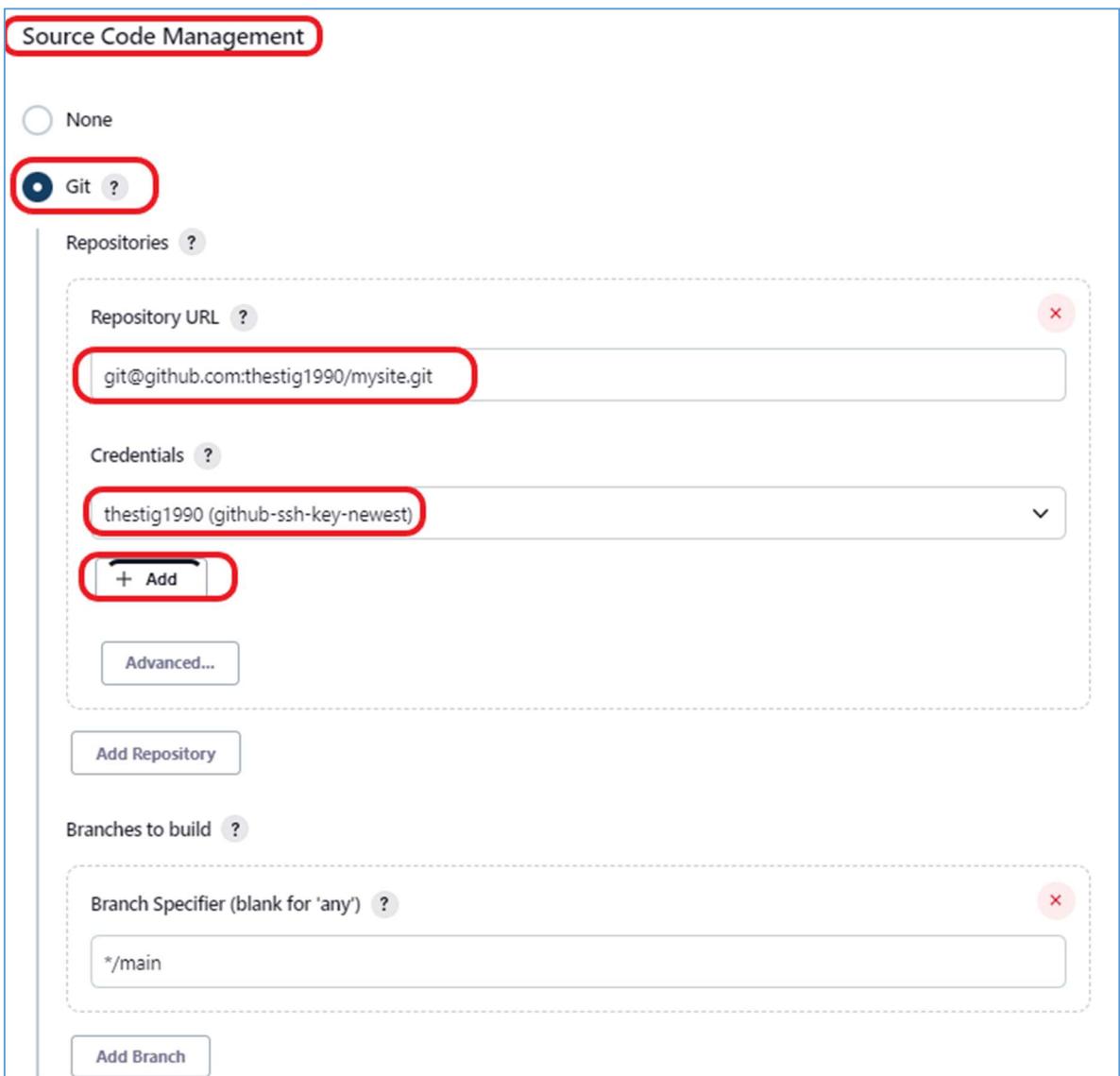
Advanced...Add Transfer Set

d. Adding Jenkins Credentials Provider.

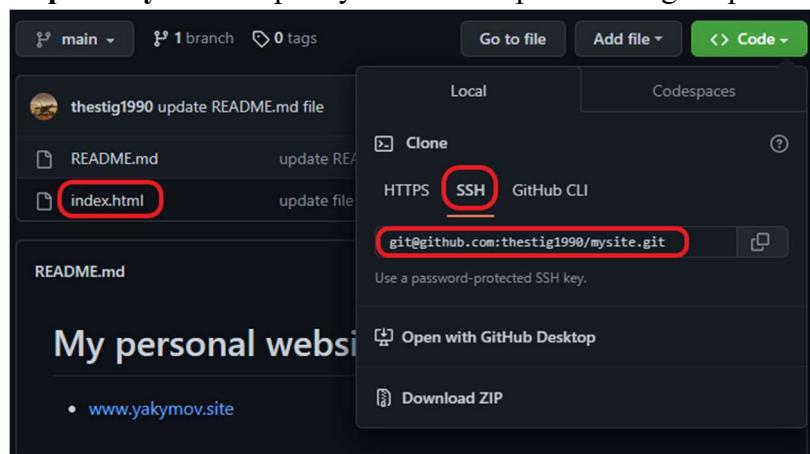
The screenshot shows the Jenkins 'Update credentials' page. The URL in the browser is `Manage Jenkins > Credentials > System > Global credentials (unrestricted) > thestig1990 (github-ssh-key-newest)`. The page has a red box around the 'Update credentials' section, which contains fields for Scope (set to 'Global'), ID ('github-ssh-key-newest'), Description ('github-ssh-key-newest'), and Username ('thestig1990'). Below these is a checkbox for 'Treat username as secret'. A second red box surrounds the 'Private Key' section, which includes an 'Enter directly' radio button and a 'Key' field containing 'Concealed for Confidentiality' with a 'Replace' button.

or

The screenshot shows the Jenkins 'Add Credentials' page. The URL in the browser is `Global credentials (unrestricted)`. The page has a red box around the 'Add Credentials' section. It includes fields for Domain ('Global credentials (unrestricted)'), Kind ('SSH Username with private key'), Scope ('Global'), ID ('github-ssh-key_newest'), Description ('github-ssh-key-newest'), Username ('thestig1990'), and a checkbox for 'Treat username as secret'. A second red box surrounds the 'Private Key' section, which includes an 'Enter directly' radio button and a 'Key' field containing 'No Stored Value'. The 'Add' button at the bottom right is also highlighted with a red circle.



- **ID** - An internal unique ID by which these credentials are identified from jobs and other configuration
- **Username** - your username in GitHub
- **Private key** – private key that you create earlier.
- **Repository URL** - specify the URL or path of the git repository.



- **Credentials** - Credential used to check out sources

e. Build job.

- Console output

Jenkins

Dashboard > jobdeploy-3 > #2

Status Changes Console Output

View as plain text Edit Build Information Delete build '#2' Timings Git Build Data Previous Build

```
Started by user admin
Running as SYSTEM
Building in workspace /var/lib/jenkins/workspace/jobdeploy-3
The recommended git tool is: NONE
using credential github-ssh-key-newest
> git rev-parse --resolve-git-dir /var/lib/jenkins/workspace/jobdeploy-3/.git # timeout=10
Fetching changes from the remote Git repository
> git config remote.origin.url git@github.com:thestig1990/mysite.git # timeout=10
Fetching upstream changes from git@github.com:thestig1990/mysite.git
> git --version # timeout=10
> git --version # 'git version 2.38.1'
using GIT_SSH to set credentials github-ssh-key-newest
[INFO] Currently running in a labeled security context
[INFO] Currently SELinux is 'enforcing' on the host
> /usr/bin/chcon --type=ssh_home_t /var/lib/jenkins/workspace/jobdeploy-3@tmp/jenkins-gitclient-ssh13978008027431207999.key
Verifying host key using known hosts file
> git fetch --tags --force --progress -- git@github.com:thestig1990/mysite.git
+refs/heads/*:refs/remotes/origin/* # timeout=10
> git rev-parse refs/remotes/origin/main^{commit} # timeout=10
Checking out Revision fc4023f3179f55e3b5bf5dd5bd8e83a18ddf5b02 (refs/remotes/origin/main)
> git config core.sparsecheckout # timeout=10
> git checkout -f fc4023f3179f55e3b5bf5dd5bd8e83a18ddf5b02 # timeout=10
Commit message: "update index.html"
> git rev-list --no-walk fc4023f3179f55e3b5bf5dd5bd8e83a18ddf5b02 # timeout=10
```

```
[jobdeploy-3] $ /bin/sh -xe /tmp/jenkins5239587303670422097.sh
+ echo -----start-----
-----start-----
+ sleep 5
+ echo -----finish-----
-----finish-----
+ ls -la
total 8
drwxr-xr-x. 3 jenkins jenkins 53 Nov 20 18:04 .
drwxrwxr-x. 9 jenkins jenkins 161 Nov 20 18:04 ..
drwxr-xr-x. 8 jenkins jenkins 162 Nov 21 10:58 .git
-rw-r--r--. 1 jenkins jenkins 2918 Nov 20 18:04 index.html
-rw-r--r--. 1 jenkins jenkins 42 Nov 20 18:04 README.md
+ echo -----deploy-----
-----deploy-----
SSH: Connecting from host [vm2]
SSH: Connecting with configuration [vm1_ubuntu20.04_webserver] ...
SSH: EXEC: completed after 203 ms
SSH: Disconnecting configuration [vm1_ubuntu20.04_webserver] ...
SSH: Transferred 1 file(s)
Build step 'Send files or execute commands over SSH' changed build result to SUCCESS
Finished: SUCCESS
```

- Checking the availability of the website



f. More about Build Triggers.

If you wanna that the process of pulling file from GitHub repo and copying this file to the web server to be automatic you need to use Build Triggers.

- Build periodically (use **Schedule** like crontab by Linux)

Build Triggers

Trigger builds remotely (e.g., from scripts) ?

Build after other projects are built ?

Build periodically ?

Provides a [cron](#)-like feature to periodically execute this project. This feature is primarily for using Jenkins as a cron replacement, and it is **not ideal for continuously building software projects**. When people first start continuous integration, they are often so used to the idea of regularly scheduled builds like nightly/weekly that they use this feature. However, the point of continuous integration is to start a build as soon as a change is made, to provide a quick feedback to the change. To do that you need to [hook up SCM change notification to Jenkins](#).

So, before using this feature, stop and ask yourself if this is really what you want.

Schedule ?

H H(11-13) 21 11 *

- Poll SCM (in this case Jenkins cheking githup repo for changes every 10 min and if there are the build is launched automatically)

Build Triggers

Trigger builds remotely (e.g., from scripts) ?

Build after other projects are built ?

Build periodically ?

GitHub hook trigger for GITScm polling ?

Poll SCM ?

Configure Jenkins to poll changes in SCM.
Note that this is going to be an expensive operation for CVS, as every polling requires Jenkins to scan the entire workspace and verify it with the server. Consider setting up a "push" trigger to avoid this overhead, as described in [this document](#)

Schedule ?

H/10 * * * *

7. Simple CI/CD pipeline example using AWS.

- Launch the AWS EC2 Instance in accordance with the steps in AWS hometask.

The screenshot shows the AWS EC2 Instances page. On the left, there's a navigation sidebar with options like EC2 Dashboard, Instances, Images, and Elastic Block Store. Under Instances, 'Instances' is selected and highlighted with a red box. The main pane displays a table of instances:

Name	Instance ID	Instance state	Instance type	Status check
JenkinsServer	i-0655a9022e4f52489	Running	t2.micro	2/2 checks pass
MyWebServer	i-0e49692a5bf1e282b	Stopped	t2.micro	-

Below the table, a detailed view for the JenkinsServer instance is shown. It includes tabs for Details, Security, Networking, Storage, Status checks, Monitoring, and Tags. The Details tab is selected. Under Instance summary, it shows the Public IPv4 address (3.12. [REDACTED] open address) and Private IPv4 addresses (172.31.14.46). The Public IPv4 address is also highlighted with a red box.

The screenshot shows the AWS Security Groups page. On the left, there's a navigation sidebar with options like EC2 Dashboard, Instances, Images, and Network & Security. Under Network & Security, 'Security Groups' is selected and highlighted with a red box. The main pane displays a table of security groups:

Name	Security group ID	Security group name	VPC ID
-	sg-0e367306d0f48a061	default	vpc-08d25191affda6188
-	sg-0b0fcdf5f75ef434f2	WebServerGroup	vpc-08d25191affda6188
JenkinsGroup	sg-0e79cbba8a5742cb4	JenkinsGroup	vpc-08d25191affda6188

Below the table, a detailed view for the JenkinsGroup is shown. It includes tabs for Details, Inbound rules, Outbound rules, and Tags. The Inbound rules tab is selected. It shows three rules:

Type	Protocol	Port range	Source
Custom TCP	TCP	8080	0.0.0.0/0
HTTP	TCP	80	0.0.0.0/0
SSH	TCP	22	[REDACTED]

The first two rules (Custom TCP and HTTP) are highlighted with a red box.

MobaXterm Personal Edition v22.1 •
(SSH client, X server and network tools)

- SSH session to `ec2-user@3.127`
 - Direct SSH : ✓
 - SSH compression : ✓
 - SSH-browser : ✓
 - X11-forwarding : ✘ (disabled or not supported by server)
- For more [info](#), [ctrl+click](#) on [help](#) or visit our [website](#).

Last login: Thu Dec 15 18:45:22 2022 from 95.150

[ec2-user@ip-172-31-14-46 ~]\$

- Install and integrate Jenkins with AWS EC2 Instance ([see instruction](#)).
 - Create job.
 - **Description** of the job and turn on **GitHub project** with github repo url.

The screenshot shows the Jenkins configuration interface for a job named "DeployMyWebsite". The top navigation bar includes the Jenkins logo, a search bar with placeholder "Search (CTRL+K)", and user notifications for 1 new item. The current user is "admin". The main title is "Configuration" and the specific section is "General". A red box highlights the "General" tab in the sidebar and the "Description" field. The "Description" field contains the text: "This job pulls repository(mywebsite) from Github over SSH and copies file index.html included into it to remote host(AWS instance) if there have been any changes." Below the description are "Plain text" and "Preview" options. Under "Build Steps", a red box highlights the "GitHub project" section, which is checked and has a "Project url" input field containing "git@github.com:thestig1990/mysite.git". Other build step options like "Discard old builds" and "Advanced..." are also visible.

Dashboard > DeployMyWebsite >

Configuration

General

Enabled

General

Source Code Management

Build Triggers

Build Environment

Build Steps

Post-build Actions

Description

This job pulls repository(mywebsite) from Github over SSH and copies file index.html included into it to remote host(AWS instance) if there have been any changes.

[Plain text] Preview

Discard old builds ?

GitHub project

Project url ?

git@github.com:thestig1990/mysite.git

Advanced...

This project is parameterized ?

- Description of the Source Code Management ([see instruction](#))

The screenshot shows the Jenkins 'Configuration' page for a job named 'DeployMyWebsite'. The 'Source Code Management' section is highlighted with a red box. Under 'Source Code Management', the 'Git' option is selected (indicated by a blue circle with a dot). The 'Repositories' section contains a 'Repository URL' field with the value 'git@github.com:thestig1990/mysite.git' and a 'Credentials' dropdown set to 'thestig1990 (ssh-github-key)'. There are buttons for '+ Add' and 'Advanced...'. Below the repositories section is a 'Branches to build' section with a 'Branch Specifier' field containing '/main'.

- Build Triggers using GitHub trigger for GITScm polling

The screenshot shows the Jenkins 'Configuration' page for the same job 'DeployMyWebsite'. The 'Build Triggers' section is highlighted with a red box. Under 'Build Triggers', the 'GitHub hook trigger for GITScm polling' checkbox is checked (indicated by a blue circle with a checkmark). A tooltip explains that Jenkins receives a GitHub push hook and checks if it came from a GitHub repository matching the SCM/Git section. If it does and the option is enabled, GitHub Plugin triggers a one-time polling on GITScm. The tooltip also notes that the last sentence describes the behavior of the Git plugin, not the GitHub plugin. The tooltip text is: 'When Jenkins receives a GitHub push hook, GitHub Plugin checks to see whether the hook came from a GitHub repository which matches the Git repository defined in SCM/Git section of this job. If they match and this option is enabled, GitHub Plugin triggers a one-time polling on GITScm. When GITScm polls GitHub, it finds that there is a change and initiates a build. The last sentence describes the behavior of Git plugin, thus the polling and initiating the build is not a part of GitHub plugin.' The source of the tooltip is cited as '(from [GitHub plugin](#))'.

- Build Steps

The screenshot shows the 'Build Steps' configuration section of the Jenkins interface. On the left, a sidebar lists 'General', 'Source Code Management', 'Build Triggers', 'Build Environment', 'Build Steps' (which is selected and highlighted with a red box), and 'Post-build Actions'. The main area is titled 'Build Steps' and contains a single step named 'Execute shell'. The 'Command' field contains the following script:

```
echo "-----start-----"
sleep 5
echo "-----finish-----"
ls -la
```

An 'Advanced...' button is located at the bottom right of the step's configuration panel.

- Post-build Actions ([Publish Over SSH](#))

The screenshot shows the 'Post-build Actions' configuration section of the Jenkins interface. On the left, a sidebar lists 'General', 'Source Code Management', 'Build Triggers', 'Build Environment', 'Build Steps', and 'Post-build Actions' (which is selected and highlighted with a red box). The main area is titled 'Post-build Actions' and contains a single action named 'Send build artifacts over SSH'. The 'SSH Publishers' section includes a 'SSH Server' configuration with 'Name ?' set to 'MyWebServer'. The 'Transfers' section includes a 'Transfer Set' configuration with 'Source files ?' set to '*' (indicated by a red box). A note below explains that the string is a comma-separated list of includes for an Ant fileset, and the base directory is the workspace. A link '(from [Publish Over SSH](#))' is at the bottom right.

- Record the settings on GitHub in the **Webhooks** section.

The image consists of three vertically stacked screenshots of the GitHub Settings page, specifically the Webhooks section. Each screenshot has a red box highlighting a specific element:

- Screenshot 1 (Top):** Shows the main GitHub repository page for 'thestig1990 / mysite'. The 'Settings' tab is highlighted with a red box. The 'Webhooks' tab is also highlighted with a red box.
- Screenshot 2 (Middle):** Shows the 'General' settings page. The 'Webhooks' tab is highlighted with a red box. The 'Add webhook' button is also highlighted with a red box.
- Screenshot 3 (Bottom):** Shows the 'Webhooks / Add webhook' configuration page. The 'Payload URL' input field, which contains the value 'http://3.1.1.145:8080/github-webhook/' (with the first '3' redacted), is highlighted with a red box. The 'Add webhook' button at the bottom is also highlighted with a red box.

* **Payload URL** - `http://jenkins_ip:port/github-webhook/`
`http://192.168.1.145:8080/github-webhook/` - example.

- Build job.

In our case jobs Build will start when we push some changes to the github repo

- Push some changes to the github repo

```

Yevhen Yakymov@DESKTOP-9L1MFHN MINGW64 ~/Desktop/Programming/GIT/GitHubRepos/mysite (main)
$ git commit -m "update README.md file"
[main a8d15c5] update README.md file
 1 file changed, 1 insertion(+)

Yevhen Yakymov@DESKTOP-9L1MFHN MINGW64 ~/Desktop/Programming/GIT/GitHubRepos/mysite (main)
$ git status
On branch main
Your branch is ahead of 'origin/main' by 1 commit.
  (use "git push" to publish your local commits)

nothing to commit, working tree clean

Yevhen Yakymov@DESKTOP-9L1MFHN MINGW64 ~/Desktop/Programming/GIT/GitHubRepos/mysite (main)
$ git push
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 12 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 350 bytes | 350.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/thestig1990/mysite.git
  8b7b132..a8d15c5  main -> main

```

The screenshot shows a GitHub repository interface. At the top, there's a commit history for a pull request from 'thestig1990' to 'mysite'. The commit message is 'update README.md file'. Below the commit history, there are two files listed: 'README.md' and 'index.html'. The 'README.md' file has a commit message 'update README.md file'. The 'index.html' file has a commit message 'update file index.html'. Below the files, there's a preview of the 'README.md' content, which includes a bulleted list:

- [www.yakymov.site](#)
- [yakymov.site](#)

A red arrow points from the word 'changes' in the commit message for 'index.html' to the second bullet point in the 'README.md' preview, indicating that the change was reflected in the website's content.

- Job description

The screenshot shows the Jenkins dashboard with the 'DeployMyWebsite' project selected. The project name is 'Project DeployMyWebsite'. The 'Status' section shows the build is 'Up-to-date'. A tooltip provides a detailed description of the job: 'This job pulls repository(mysite) from Github over SSH and copies file index.html included into it to the remote host(AWS instance - MyWebServer) if there have been any changes with helping by Webhooks'. There are also links for 'Edit description', 'Disable Project', and 'Workspace'.

- Build History

The screenshot shows the Jenkins Build History interface. At the top, there is a search bar labeled "Filter builds...". Below it, two builds are listed: build #12 (Dec 18, 2022, 12:04 PM) and build #11 (Nov 26, 2022, 1:44 PM). The build #12 row is highlighted with a red box, indicating it is selected. To the right of the build list is a vertical toolbar with up and down arrows.

- Console output

The screenshot shows the Jenkins Console Output page for build #12. The top navigation bar includes "Dashboard", "DeployMyWebsite", and "#12", with "#12" highlighted by a red box. On the left, there is a sidebar with links: "Changes", "Console Output" (which is selected and highlighted by a red box), "View as plain text", "Edit Build Information", "Delete build '#12'", "Polling Log", "Git Build Data", and "Previous Build". The main content area displays the build logs:

```
</> Changes
Started by GitHub push by thestig1990
Running as SYSTEM
Building on the built-in node in workspace /var/lib/jenkins/workspace/DeployMyWebsite
The recommended git tool is: NONE
using credential ssh-github-key
> git rev-parse --resolve-git-dir /var/lib/jenkins/workspace/DeployMyWebsite/.git # timeout=10
Fetching changes from the remote Git repository
> git config remote.origin.url git@github.com:thestig1990/mysite.git # timeout=10
Fetching upstream changes from git@github.com:thestig1990/mysite.git
> git --version # timeout=10
> git --version # 'git' version 2.37.1'
using GIT_SSH to set credentials ssh-github-key
Verifying host key using known hosts file
> git fetch --tags --force --progress -- git@github.com:thestig1990/mysite.git
+refs/heads/*:refs/remotes/origin/* # timeout=10
> git rev-parse refs/remotes/origin/main^{commit} # timeout=10
Checking out Revision a8d15c5ee78b6076da30410cb74722b7bf92ef9f (refs/remotes/origin/main)
> git config core.sparsecheckout # timeout=10
> git checkout -f a8d15c5ee78b6076da30410cb74722b7bf92ef9f # timeout=10
Commit message: "update README.md file"
> git rev-list --no-walk 8b7b132967b365cab872a5cce46a433d3fed11 # timeout=10
[DeployMyWebsite] $ /bin/sh -xe /tmp/jenkins13885904089589833644.sh
+ echo -----start-----
-----start-----
+ sleep 5
+ echo -----finish-----
-----finish-----
+ ls -la
total 8
drwxr-xr-x 3 jenkins jenkins 53 Dec 18 12:04 .
drwxr-xr-x 5 jenkins jenkins 75 Nov 22 20:25 ..
drwxr-xr-x 8 jenkins jenkins 162 Dec 18 12:04 .git
-rw-r--r-- 1 jenkins jenkins 2761 Nov 22 20:55 index.html
-rw-r--r-- 1 jenkins jenkins 80 Dec 18 12:04 README.md
+ echo -----deploy-----
-----deploy-----
SSH: Connecting from host [ip-172-31-14-46.eu-central-1.compute.internal]
SSH: Connecting with configuration [MyWebServer] ...
SSH: EXEC: completed after 201 ms
SSH: Disconnecting configuration [MyWebServer] ...
SSH: Transferred 2 file(s)
Finished: SUCCESS
```

- Polling Log

Dashboard > DeployMyWebsite > #12 > Polling Log

Polling Log

This page captures the polling log that triggered this build.

Started on Dec 18, 2022, 12:04:22 PM
 Started by event from 140.82.115.100 => http://3.1.180.800/github-webhook/ on Sun Dec 18 12:04:22 UTC 2022
 Using strategy: Default
 [poll] Last Built Revision: Revision 8b7b132967b365cab872a5ccee46a4336d3fed11 (refs/remotes/origin/main)
 The recommended git tool is: NONE
 using credential ssh-github-key
 > git --version # timeout=10
 > git --version # 'git version 2.37.1'
 using GIT_SSH to set credentials ssh-github-key
 Verifying host key using known hosts file
 > git ls-remote -h -- git@github.com:thestig1990/mysite.git # timeout=10
 Found 1 remote heads on git@github.com:thestig1990/mysite.git
 [poll] Latest remote head revision on refs/heads/main is: a8d15c5ee78b6076da30410cb74722b7bf92ef9f
 Done. Took 1 sec
 Changes found

Back to Project | Status | Changes | Console Output | Edit Build Information | Delete build '#12' | Polling Log | View as plain text | Git Build Data | Previous Build

- Checking the availability of the website

← → C ⌂ ▲ Не конфіденційний 52.58

Ansible Asure AWS Bash Database Data encoding DevOps Docker EPAM_DevOps GIT HTML-CSS »

Hi there 🙋, I'm Yevhen Yakymov

EPAM Cloud&DevOps Fundamentals Autumn 2022

List of AWS services I have worked with:

1. [Amazon Elastic Compute Cloud \(Amazon EC2\)](#)
2. [Amazon Simple Storage Service \(Amazon S3\)](#)
3. [Amazon Lightsail\(virtual private server\)](#)
4. [Amazon Relational Database Service \(Amazon RDS\)](#)
5. [AWS Lambda \(serverless\)](#)
6. [Amazon Virtual Private Cloud \(Amazon VPC\)](#)
7. [Amazon Route 53 \(Domain Name System\)](#)
8. [Amazon CloudFront \(content delivery network\)](#)
9. [AWS Identity and Access Management \(IAM\)](#)
10. [Amazon CloudWatch \(monitor AWS resources\)](#)

List of successfully completed courses on AWS services:

1. [AWS Cloud Practitioner Essentials](#)
2. [To be continued... 😊](#)

Yours truly, Yevhen Yakymov
 email: yakymov1yevhen@gmail.com

8. Jenkins nodes(agent/slave).

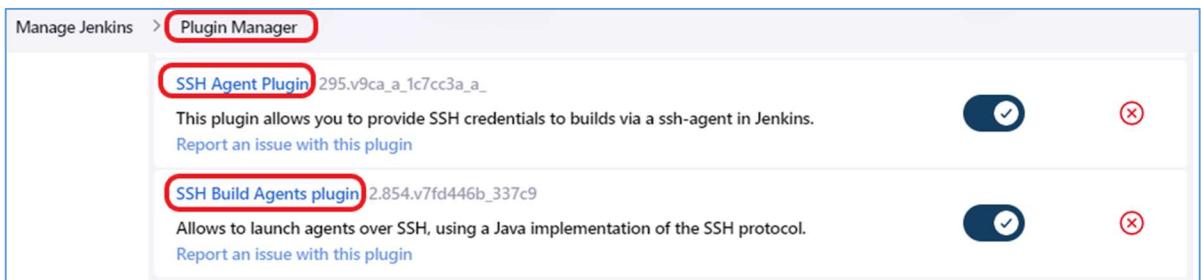
Jenkins nodes(agent):

- Run more builds on Jenkins nodes(agent)
- Decrease load to Jenkins controller
- Different OS'es can be used on Jenkins nodes(agent) for different application

Jenkins controller require:

- Java instalation on Jenkins agent

```
$ sudo apt-get install openjdk-11-jdk (for Ubuntu)
```
- Network connection beetwen Jenkins agent and Jenkins controller
- Jenkins controller needs two plugins to communicate with Jenkins agent:



- You must generate **ssh key** on Jenkins controller and copy **public key** to the Jenkins agent before their usage will be available:

```
$ ssh-keygen -t rsa -f "key_name"  
$ ssh-copy-id "user_name@host_name"
```

- On Jenkins agent (Ubuntu) should be done:

\$ sudo apt update	#update repo list
*\$ ip a	#check IP for SSH
*\$ pwd	#check where are you now
*\$ mkdir -p /home/jenkins	#home directory for Jenkins
\$ sudo apt-get install openjdk-11-jdk	#needed for Jenkins

Creating Jenkins nodes(agent):

Jenkins

Dashboard > Manage Jenkins

Manage Jenkins

New version of Jenkins (2.375.1) is available for download (changelog).

Building on the built-in node can be a security issue. You should set the number of executors on the built-in node to 0. See [the documentation](#).

Manage Jenkins

System Configuration

- Configure System: Configure global settings and paths.
- Global Tool Configuration: Configure tools, their locations and automatic installers.
- Manage Plugins: Add, remove, disable or enable plugins that can extend the functionality of Jenkins.
- Manage Nodes and Clouds: Add, remove, control and monitor the various nodes that Jenkins runs jobs on.

Build Queue: No builds in the queue.

Build Executor Status: Built-In Node, 1 Idle.

Jenkins

Dashboard > Manage Jenkins > Nodes

Manage nodes and clouds

Refresh status

New Node

S	Name	Architecture	Clock Difference	Free Disk Space	Free Swap Space	Free Temp Space	Response Time
	Built-In Node	Linux (amd64)	In sync	3.46 GB	! 0 B	3.46 GB	0ms
	MyAgent1	Linux (amd64)	In sync	5.55 GB	! 0 B	5.55 GB	4ms

Provision via JenkinsServer

Data obtained 56 min 56 min 56 min 56 min 56 min 56 min 56 min

Dashboard > Manage Jenkins > Nodes >

New node

Node name: MyAgent

Type: Permanent Agent

Permanent Agent: Adds a plain, permanent agent to Jenkins. This is called "permanent" because Jenkins doesn't provide higher level of integration with these agents, such as dynamic provisioning. Select this type if no other agent types apply — for example such as when you are adding a physical computer, virtual machines managed outside Jenkins, etc.

Copy Existing Node

Create

9. Simple job using Jenkins agent.

- Create and configure Jenkins agent.

The screenshot shows the Jenkins Node configuration page for a node named 'MyAgent1'. The 'Configure' button is highlighted with a red box. The 'Number of executors' field is also highlighted with a red box and contains the value '2'. A note below explains that the maximum number of concurrent builds is determined by the number of CPU cores. It also notes that agents must have at least one executor. A note for built-in nodes states that setting executors to zero prevents local execution on the controller. The 'Remote root directory' field is set to '/home/jenkins/'. A note explains that this should be a path local to the agent machine. The 'Build Executor Status' dropdown shows two idle executors. A note about relative paths for the remote root directory is present, along with a bulleted list of considerations for launcher types. The 'Labels' field is highlighted with a red box and contains 'apache' and 'aws'. A note explains that labels group agents into logical groups, and multiple labels must be separated by a space. It also notes that labels can represent operating systems or tools. A note about special characters in labels is at the bottom.

Dashboard > Nodes > MyAgent1

Name: MyAgent1

Description: my first agent

Configure

Number of executors: 2

The maximum number of concurrent builds that Jenkins may perform on this node. A good value to start with would be the number of CPU cores on the machine. Setting a higher value would cause each build to take longer, but could increase the overall throughput. For example, one build might be CPU-bound, while a second build running at the same time might be I/O-bound — so the second build could take advantage of the spare I/O capacity at that moment.

Agents (nodes that are not the built-in node) must have at least one executor. To temporarily prevent any builds from being executed on an agent, use the *Mark this node temporarily offline* button on the agent's page.

For the built-in node, set the number of executors to zero to prevent it from executing builds locally on the controller. *Note: The built-in node will always be able to run flyweight tasks including Pipeline's top-level task.*

Remote root directory: /home/jenkins/

An agent needs to have a directory dedicated to Jenkins. Specify the path to this directory on the agent. It is best to use an absolute path, such as `/var/jenkins` or `c:\jenkins`. This should be a path local to the agent machine. There is no need for this path to be visible from the controller.

Agents do not maintain important data: all job configurations, build logs and artifacts are stored on the controller, so it would be possible to use a temporary directory as the agent root directory. However, by giving an agent a directory that is not deleted after a machine reboot, for example, the agent can cache data such as tool installations, or build workspaces. This prevents unnecessary downloading of tools, or checking out source code again when builds start to run on this agent again after a reboot.

If you use a relative path, such as `./jenkins-agent`, the path will be relative to the working directory provided by the *Launch method*.

- For launchers where Jenkins controls starting the agent process, such as SSH, the current working directory will typically be consistent, e.g. the user's home directory.
- For launchers where Jenkins has no control over starting the agent process, such as inbound agents launched from the command line, the current working directory may change between launches of the agent and use of a relative path may prove problematic.

The principal issue encountered when using relative paths with inbound launchers is the proliferation of stale workspaces and tool installation on the agent machine. This can cause disk space issues.

Labels: apache aws

Labels (or tags) are used to group multiple agents into one logical group. For example, if you have multiple Windows agents and you have a job that must run on Windows, then you could configure all your Windows agents to have the label `windows`, and then tie that job to this label. This would ensure that your job runs on one of your Windows agents, but not on any agents without this label.

Labels do not necessarily have to represent the operating system on the agent: you can also use labels to note the CPU architecture, or that a certain tool is installed on the agent.

Multiple labels must be separated by a space. For example, `windows docker` would assign two labels to the agent: `windows` and `docker`.

Labels may contain any non-space characters, but you should avoid special characters such as any of these: `!&|<>()`, as other Jenkins features allow for defining label expressions, where these characters may be used.

Trust SSH Host Key

Usage ?

Use this node as much as possible

Launch method ?

Launch agents via SSH

Host ?

ec2-52-56-171-11.eu-central-1.compute.amazonaws.com

Credentials ?

ec2-user

+ Add

Host Key Verification Strategy ?

Manually trusted key Verification Strategy

Require manual verification of initial connection ?

Advanced...

Availability ?

Keep this agent online as much as possible

Build Executor Status

1 Idle

2 Idle

Node Properties

Disable deferred wipeout on this node ?

Environment variables

List of variables ?

Name	Value
PWD	/home/jenkins/workspace

Add

Tool Locations

Save

- Create job

Jenkins

Dashboard > DeployMyWebsite1 >

Status

Project DeployMyWebsite1

This job pulls repository(mysite) from Github over SSH if there have been any changes with helping by Webhooks. The job is executed on the Node - MyAgent1(MyWebServer-AWS EC2 Instance).

Configuration

General

Enabled

Description

This job pulls repository(mysite) from Github over SSH if there have been any changes with helping by Webhooks. The job is executed on the Node - MyAgent1(MyWebServer-AWS EC2 Instance).

[Plain text] [Preview](#)

Discard old builds ?

GitHub project

This project is parameterized ?

Throttle builds ?

Execute concurrent builds if necessary ?

Restrict where this project can be run ?

Label Expression ?

apache

we configured Labels when we were creating new Jenkins agent - "MyAgent1"

Label apache matches 1 node. Permissions or other restrictions provided by plugins may further reduce that list.

[Advanced...](#)

Source Code Management

Configuration

None

Git ?

Source Code Management

Repositories ?

Repository URL ?

git@github.com:thestig1990/mysite.git

Credentials ?

thestig1990 (ssh-github-key)

+ Add

Advanced...

Dashboard > DeployMyWebsite1 >

Configuration

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

Build Triggers

- Trigger builds remotely (e.g., from scripts) ?
- Build after other projects are built ?
- Build periodically ?
- GitHub hook trigger for GITScm polling ?
- Poll SCM ?

Dashboard > DeployMyWebsite1 >

Configuration

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

Build Steps

Execute shell ?

Command

See the list of available environment variables

```
echo "-----start-----"
sleep 5
pwd
whoami
hostname
cp * /home/ec2-user/programming/github/mysite
echo "-----finish-----"
```

Advanced...

Add build step ▾

Post-build Actions

Add post-build action ▾

Save **Apply**

- Build job

The screenshot shows the Jenkins dashboard with the following details:

- Build Queue:** Displays a list of jobs:

S	W	Name	Last Success	Last Failure	Last Duration
✓	⌚	DeployMyWebsite	1 day 3 hr #12	N/A	7 sec
✓	⌚	DeployMyWebsite1	17 sec #22	23 days #14	6.2 sec
✓	⌚	java-maven-test	22 days #1	N/A	16 sec
✓	⌚	jobdeploy-1	27 days #1	N/A	0.22 sec
✓	⌚	test-pipeline	22 days #14	N/A	0.62 sec
✓	⌚	test-pipeline-SCM	22 days #2	N/A	3.5 sec
- Build Executor Status:** Shows the status of build agents:

Icon	S	M	L
Idle	1	0	0
MyAgent1	1	0	0
Idle	2	0	0

The screenshot shows the configuration page for the DeployMyWebsite1 project:

- Project DeployMyWebsite1:** A summary section with the following text:

This job pulls repository(mysite) from Github over SSH if there have been any changes with helping by Webhooks.
The job is executed on the Node - MyAgent1(MyWebServer-AWS EC2 Instance).
- Status:** A tabbed interface showing the current status of the project.
- Build History:** A table showing the history of builds:

#	Build Status	Duration	Timestamp
#23	Success	2 min 23 sec ago	Dec 19, 2022, 3:11 PM

- Console output

Dashboard > DeployMyWebsite1 > #23

</> Changes Started by user admin
 Console Output Running as SYSTEM
 View as plain text
 Edit Build Information
 Delete build '#23'
 Git Build Data
 Previous Build

```

Building remotely on MyAgent1 (apache aws) in workspace /home/jenkins/workspace/DeployMyWebsite1
The recommended git tool is: NONE
using credential ssh-github-key
> git rev-parse --resolve-git-dir /home/jenkins/workspace/DeployMyWebsite1/.git # timeout=10
Fetching changes from the remote Git repository
> git config remote.origin.url git@github.com:thestig1990/mysite.git # timeout=10
Fetching upstream changes from git@github.com:thestig1990/mysite.git
> git --version # timeout=10
> git --version # 'git version 2.37.1'
using GIT_SSH to set credentials ssh-github-key
Verifying host key using known hosts file
> git fetch --tags --force --progress -- git@github.com:thestig1990/mysite.git
+refs/heads/*:refs/remotes/origin/* # timeout=10
> git rev-parse refs/remotes/origin/main^{commit} # timeout=10
Checking out Revision a8d15c5ee78b6076da30410cb74722b7bf92ef9f (refs/remotes/origin/main)
> git config core.sparsecheckout # timeout=10
> git checkout -f a8d15c5ee78b6076da30410cb74722b7bf92ef9f # timeout=10
Commit message: "update README.md file"
> git rev-list --no-walk a8d15c5ee78b6076da30410cb74722b7bf92ef9f # timeout=10
[DeployMyWebsite1] $ /bin/sh -xe /tmp/jenkins13054472884212539878.sh
+ echo -----start-----
-----start-----
+ sleep 5
+ pwd
/home/jenkins/workspace/DeployMyWebsite1
+ whoami
ec2-user
+ hostname
ip-172-31-23-81.eu-central-1.compute.internal
+ cp index.html README.md /home/ec2-user/programming/github/mysite
+ echo -----finish-----
-----finish-----
+ ls -la
total 8
drwxrwxr-x 3 ec2-user ec2-user 53 Dec 18 12:08 .
drwxrwxr-x 7 ec2-user ec2-user 135 Nov 27 12:35 ..
drwxrwxr-x 8 ec2-user ec2-user 162 Dec 19 15:11 .git
-rw-rw-r-- 1 ec2-user ec2-user 2761 Nov 26 12:25 index.html
-rw-rw-r-- 1 ec2-user ec2-user 80 Dec 18 12:08 README.md
+ echo -----deploy-----
-----deploy-----
Finished: SUCCESS

```

<https://aws.amazon.com/amazon-linux-2/>
34 package(s) needed for security, out of 54 available
Run "sudo yum update" to apply all updates.
[ec2-user@ip-172-31-23-81 ~]\$
[ec2-user@ip-172-31-23-81 ~]\$ hostname
ip-172-31-23-81.eu-central-1.compute.internal ←
[ec2-user@ip-172-31-23-81 ~]\$ cd programming/github/mysite/
[ec2-user@ip-172-31-23-81 mysite]\$ ll
total 8
-rw-rw-r-- 1 ec2-user ec2-user 2761 Dec 19 15:11 index.html
-rw-rw-r-- 1 ec2-user ec2-user 80 Dec 19 15:11 README.md
[ec2-user@ip-172-31-23-81 mysite]\$

10. Simple Jenkins pipeline example.

- Create pipeline

The screenshot shows the Jenkins 'Create Pipeline' page. At the top, there is a search bar with the placeholder 'Search (CTRL+K)' and a user icon for 'admin'. Below the search bar, the URL 'Dashboard > All >' is visible. A red box highlights the input field 'Enter an item name' at the top left. Inside this field, the text 'test_pipeline' is entered. Below the input field, a note says '» Required field'. To the right, there are two options: 'Freestyle project' (represented by a blue icon) and 'Pipeline' (represented by a red box). A note under 'Freestyle project' states: '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.' A note under 'Pipeline' states: '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.'

The screenshot shows the Jenkins 'test-pipeline' configuration page. At the top, there is a search bar with the placeholder 'Search (CTRL+K)' and a user icon for 'admin'. Below the search bar, the URL 'Dashboard > test-pipeline >' is visible. The page has tabs for 'Configuration' and 'General'. The 'General' tab is selected and highlighted with a red box. It contains a 'Description' field with the text 'my first declarative pipeline script'. To the right of the 'Description' field is a 'Enabled' checkbox which is checked. Below the 'General' tab, there are three buttons: 'General' (highlighted with a red box), 'Advanced Project Options', and 'Pipeline'. The 'Pipeline' button is also highlighted with a red box. At the bottom of the 'General' section, there are links for '[Plain text] Preview' and 'Advanced...'.

The screenshot shows the Jenkins 'test-pipeline' configuration page, specifically the 'Pipeline' section. The 'Pipeline' tab is selected and highlighted with a red box. Within the 'Pipeline' section, the 'Definition' dropdown is set to 'Pipeline script' and is highlighted with a red box. Below the dropdown, a code editor displays a Groovy pipeline script:

```
1 pipeline {
2     agent any
3
4     stages {
5         stage('Build') {
6             steps {
7                 echo 'Building...'
8             }
9         }
10        stage('Test') {
11            steps {
12                echo 'Testing...'
13            }
14        }
15        stage('Deploy') {
16            steps {
17                echo 'Deploying...'
18            }
19        }
20    }
21}
```

Below the code editor, there is a checkbox labeled 'Use Groovy Sandbox' which is checked. At the bottom of the page, there are two buttons: 'Save' and 'Apply', both highlighted with a red box.

- Build pipeline

The screenshot shows the Jenkins Pipeline test-pipeline page. At the top, there's a navigation bar with 'Dashboard' and 'test-pipeline'. Below it is a sidebar with options like 'Status', 'Changes', 'Build Now' (which is highlighted with a red box), 'Configure', 'Delete Pipeline', 'Full Stage View', 'Rename', 'Pipeline Syntax', 'Build History' (with a 'trend' dropdown), and a 'Filter builds...' search bar. A red box highlights the 'Build Now' button. To the right, the 'Pipeline test-pipeline' title is shown above the 'Stage View' section. The Stage View table has columns for 'Build', 'Test', and 'Deploy'. It shows two stages: one for build (#14) and one for test (#13). The build stage for #14 has an average time of 77ms. The test stage for #14 has an average time of 61ms. The deploy stage for #14 has an average time of 64ms. The build stage for #13 has an average time of 58ms. The test stage for #13 has an average time of 47ms. The deploy stage for #13 has an average time of 43ms. A red box highlights the entire Stage View table. At the bottom left, a red box highlights the 'Build History' section, specifically the entry for build #14 on Nov 27, 2022, at 10:57 AM.

- Console output

The screenshot shows the Jenkins Pipeline test-pipeline build #14 console output page. The navigation bar includes 'Dashboard', 'test-pipeline', and '#14'. The sidebar on the left includes 'Status', 'Changes', 'Console Output' (which is highlighted with a red box), 'View as plain text', 'Edit Build Information', 'Delete build #14', 'Restart from Stage', 'Replay', 'Pipeline Steps', 'Workspaces', and 'Previous Build'. The main area is titled 'Console Output' with a green checkmark icon. It displays the Jenkins pipeline logs for build #14. The logs show the pipeline starting, running on Jenkins in /var/lib/jenkins/workspace/test-pipeline, building, testing, and deploying. It ends with a 'Finished: SUCCESS' message. A red box highlights the log entry 'Running on Jenkins in /var/lib/jenkins/workspace/test-pipeline'.

```

Started by user admin
[Pipeline] Start of Pipeline
[Pipeline] node
Running on Jenkins in /var/lib/jenkins/workspace/test-pipeline
[Pipeline] {
[Pipeline] stage
[Pipeline] { (Build)
[Pipeline] echo
Building...
[Pipeline] }
[Pipeline] // stage
[Pipeline] stage
[Pipeline] { (Test)
[Pipeline] echo
Testing...
[Pipeline] }
[Pipeline] // stage
[Pipeline] stage
[Pipeline] { (Deploy)
[Pipeline] echo
Deploying...
[Pipeline] }
[Pipeline] // stage
[Pipeline] }
[Pipeline] // node
[Pipeline] End of Pipeline
Finished: SUCCESS

```

11. Simple Jenkins pipeline script from SCM (Source Control Management).

- Create pipeline

The screenshot shows the Jenkins dashboard. On the left, there's a sidebar with links like 'Dashboard', 'New Item', 'People', 'Build History', 'Project Relationship', 'Check File Fingerprint', 'Manage Jenkins', 'My Views', 'Build Queue' (empty), 'Build Executor Status' (empty), and 'Built-In Node'. The main area shows a table of existing pipelines. One row, 'test-pipeline-SCM', is highlighted with a red box.

S	W	Name	Last Success	Last Failure	Last Duration
✓	⌚	DeployMyWebsite	1 day 4 hr #12	N/A	7 sec
✓	⌚	DeployMyWebsite1	1 hr 19 min #23	23 days #14	6.2 sec
✓	⌚	java-maven-test	22 days #1	N/A	16 sec
✓	⌚	jobdeploy-1	27 days #1	N/A	0.22 sec
✓	⌚	test-pipeline	22 days #14	N/A	0.62 sec
✓	⌚	test-pipeline-SCM	22 days #2	N/A	3.5 sec

This screenshot shows the configuration page for the 'test-pipeline-SCM' job. It's under the 'General' tab. The 'Enabled' switch is turned on. In the 'Description' field, the text 'my first declarative pipeline script from SCM(Source Control Management)' is entered. The sidebar on the left has tabs for 'General', 'Advanced Project Options' (which is highlighted with a red box), and 'Pipeline'.

This screenshot shows the configuration page for the 'test-pipeline-SCM' job, specifically the 'Pipeline' tab. Under 'Definition', it says 'Pipeline script from SCM'. Under 'SCM', it says 'Git'. In the 'Repositories' section, the 'Repository URL' is set to 'git@github.com:thestig1990/jenkinsfile-library.git' and the 'Credentials' are set to 'thestig1990 (ssh-github-key)'. The sidebar on the left has tabs for 'General', 'Advanced Project Options' (highlighted with a red box), and 'Pipeline'.

Script Path ?

Jenkinsfile

Relative location (/ -separated regardless of platform) within the checkout of your Pipeline script. Note that it will always be run inside a Groovy sandbox. Jenkinsfile is conventional and allows you to switch easily to a multibranch project (just use checkout scm to retrieve sources from the same location as is configured here).
(from [Pipeline: Groovy](#))

Lightweight checkout ?

Pipeline Syntax

Save **Apply**

theстig1990 / jenkinsfile-library Public

Code Issues Pull requests Actions Projects Wiki

main Jenkinsfile

theстig1990 added first Jenkinsfile

1 contributor

```
21 lines (20 sloc) | 358 Bytes
```

```
1 pipeline {
2     agent any
3
4     stages {
5         stage('Build') {
6             steps {
7                 echo 'Building...'
8             }
9         }
10        stage('Test') {
11            steps {
12                echo 'Testing...'
13            }
14        }
15        stage('Deploy') {
16            steps {
17                echo 'Deploying...'
18            }
19        }
20    }
21 }
```

- Build pipeline

The screenshot shows the Jenkins Pipeline test-pipeline-SCM page. On the left, there's a sidebar with various options like Status, Changes, Build Now (which is highlighted with a red box), Configure, Delete Pipeline, Full Stage View, Rename, Pipeline Syntax, Build History (with a trend dropdown and a filter bar), and a list of builds (#2 and #1). The main area is titled "Pipeline test-pipeline-SCM" and contains a "Stage View" section. This view shows a grid of stages: Declarative: Checkout SCM (1s), Build (69ms), Test (50ms), and Deploy (57ms). Below the grid, it says "Average stage times: (Average full run time: ~4s)". There are two build entries: #2 (Nov 27 11:17) and #1 (Nov 27 10:54), both labeled "No Changes". A red box highlights the "Declarative: Checkout SCM" stage.

- Console output

The screenshot shows the Jenkins Console Output page for build #2. The sidebar on the left includes options like Status, Changes, Console Output (which is highlighted with a red box), View as plain text, Edit Build Information, Delete build '#2', Git Build Data, Restart from Stage, Replay, Pipeline Steps, Workspaces, and Previous Build. The main content area is titled "Console Output" and shows the log output for build #2. The log starts with "Started by user admin" and "Obtained Jenkinsfile from git git@github.com:thestig1990/jenkinsfile-library.git". It then details the pipeline execution, including the checkout step where it uses the Default Git installation and the NONE git tool with ssh-github-key credential. It fetches changes from the remote repository and performs a git rev-parse command. The log concludes with "Deploying...", "Finished: SUCCESS", and "Verifying host key using known hosts file". A red box highlights the line "Obtained Jenkinsfile from git git@github.com:thestig1990/jenkinsfile-library.git".

12. Jenkins for Java app using Maven.

Apache Maven is a software project management and comprehension tool. Based on the concept of a project object model (POM), **Maven** can manage a project's build, reporting and documentation from a central piece of information.

- [Download and install Maven](#)

The screenshot shows two main sections of the Apache Maven Project website:

Download Apache Maven 3.8.6

System Requirements

Java Development Kit (JDK)	Maven 3.3+ require JDK 1.7 or above to execute - they still allow you to build against 1.3 and other JDK versions by Using Toolchains
Memory	No minimum requirement
Disk	Approximately 10MB is required for the Maven installation itself. In addition to that, additional disk space will be used for your local Maven repository. The size of your local repository will vary depending on usage but expect at least 500MB.
Operating System	No minimum requirement. Start up scripts are included as shell scripts and Windows batch files.

Files

Maven is distributed in several formats for your convenience. Simply pick a ready-made binary distribution archive and follow the [installation instructions](#). Use a source archive if you intend to build Maven yourself.

In order to guard against corrupted downloads/installations, it is highly recommended to [verify the signature](#) of the release bundles against the public [KEYS](#) used by the Apache Maven developers.

Link	Checksums	Signature
Binary tar.gz archive	apache-maven-3.8.6-bin.tar.gz	apache-maven-3.8.6-bin.tar.gz.asc
Binary zip archive	apache-maven-3.8.6-bin.zip	apache-maven-3.8.6-bin.zip.sha512

Installing Apache Maven

The installation of Apache Maven is a simple process of extracting the archive and adding the `bin` folder with the `mvn` command to the `PATH`.

Detailed steps are:

- Have a JDK installation on your system. Either set the `JAVA_HOME` environment variable pointing to your JDK installation or have the `java` executable on your `PATH`.
- Extract distribution archive in any directory

```
1. unzip apache-maven-3.8.6-bin.zip
```

or

```
1. tar xzvf apache-maven-3.8.6-bin.tar.gz
```

Alternatively use your preferred archive extraction tool.

- Add the `bin` directory of the created directory `apache-maven-3.8.6` to the `PATH` environment variable
- Confirm with `mvn -v` in a new shell. The result should look similar to:

```
1. Apache Maven 3.8.6 (84538c998a25aec085021c365c560670ad80f63)
2. Maven home: /opt/apache-maven-3.8.6
3. Java version: 1.8.0_45, vendor: Oracle Corporation
4. Java home: /Library/Java/JavaVirtualMachines/jdk1.8.0_45.jdk/Contents/Home/jre
5. Default locale: en_US, platform encoding: UTF-8
6. OS name: "mac os x", version: "10.8.5", arch: "x86_64", family: "mac"
```

Windows Tips

- Check environment variable value e.g.

```
1. echo %JAVA_HOME%
2. C:\Program Files\Java\jdk1.7.0_51
```

- Adding to `PATH`: Add the unpacked distribution's `bin` directory to your user `PATH` environment variable by opening up the system properties (WinKey + Pause), selecting the Advanced tab, and the Environment Variables button, then adding or selecting the `PATH` variable in the user variables with the value `C:\Program Files\Apache-Maven-3.8.6\bin`. The same dialog can be used to set `JAVA_HOME` to the location of your JDK, e.g. `C:\Program Files\Java\jdk1.7.0_51`
- Open a new command prompt (Winkey + R then type `cmd`) and run `mvn -v` to verify the installation.

Unix-based Operating System (Linux, Solaris and Mac OS X) Tips

- Check environment variable value

```
1. echo $JAVA_HOME
2. /Library/Java/JavaVirtualMachines/jdk1.8.0_45.jdk/Contents/Home
```

- Adding to `PATH`

```
1. export PATH=/opt/apache-maven-3.8.6/bin:$PATH
```

- Java project structure

```
[vagrant@vm2 java-test]$ tree
.
└── pom.xml
    └── src
        ├── main
        │   └── java
        │       └── com
        │           └── mycompany
        │               └── app
        │                   └── App.java
        └── test
            └── java
                └── com
                    └── mycompany
                        └── app
                            └── AppTest.java
11 directories, 3 files
[vagrant@vm2 java-test]$
```

```
[vagrant@vm2 java-test]$ cat src/main/java/com/mycompany/app/App.java
package com.mycompany.app;

/**
 * Hello world!
 *
 */
public class App
{
    public static void main( String[] args )
    {
        System.out.println( "Hello World!" );
    }
}
[vagrant@vm2 java-test]$ cat src/test/java/com/mycompany/app/AppTest.java
package com.mycompany.app;

import static org.junit.Assert.assertTrue;

import org.junit.Test;

/**
 * Unit test for simple App.
 */
public class AppTest
{
    /**
     * Rigorous Test :-)
     */
    @Test
    public void shouldAnswerWithTrue()
    {
        assertTrue( true );
    }
}
```

- Add Maven bin to the PATH and checking installation

```
[vagrant@vm2 apache-maven-3.8.6]$ export PATH=$PATH:/home/vagrant/install/apache-maven-3.8.6/bin/
[vagrant@vm2 apache-maven-3.8.6]$ mvn -v
Apache Maven 3.8.6 (84538c9988a25aec085021c365c560670ad80f63)
Maven home: /home/vagrant/install/apache-maven-3.8.6
Java version: 11.0.17, vendor: Red Hat, Inc., runtime: /usr/lib/jvm/java-11-openjdk-11.0.17.0.8-2
.el7_9.x86_64
Default locale: en_US, platform encoding: UTF-8
OS name: "linux", version: "3.10.0-1160.80.1.el7.x86_64", arch: "amd64", family: "unix"
[vagrant@vm2 apache-maven-3.8.6]$
```

- Run Maven test for java app

```
[vagrant@vm2 apache-maven-3.8.6]$ cd /home/vagrant/git/java-test/
[vagrant@vm2 java-test]$
[vagrant@vm2 java-test]$
[vagrant@vm2 java-test]$ ll
total 4
-rw-rw-r--. 1 vagrant vagrant 2653 Dec 20 14:58 pom.xml
drwxrwxr-x. 4 vagrant vagrant 30 Dec 20 14:58 src
[vagrant@vm2 java-test]$ mvn test
[INFO] Scanning for projects...
[INFO]
[INFO] ----- < com.mycompany.app:my-app > -----
[INFO] Building my-app 1.0-SNAPSHOT
[INFO] ----- [ jar ] -----
Downloading from central: https://repo.maven.apache.org/maven2/org/apache/maven/plugins/maven-resources-plugin/3.0.2/maven-resources-plugin-3.0.2.pom
Downloaded from central: https://repo.maven.apache.org/maven2/org/apache/maven/plugins/maven-resources-plugin/3.0.2/maven-resources-plugin-3.0.2.pom (7.1 kB at 8.5 kB/s)
Downloaded from central: https://repo.maven.apache.org/maven2/org/apache/maven/surefire/surefire-providers/2.22.1/surefire-providers-2.22.1.pom (2.5 kB at 34 kB/s)
Downloading from central: https://repo.maven.apache.org/maven2/org/apache/maven/surefire/surefire-junit4/2.22.1/surefire-junit4-2.22.1.jar
Downloaded from central: https://repo.maven.apache.org/maven2/org/apache/maven/surefire/surefire-junit4/2.22.1/surefire-junit4-2.22.1.jar (85 kB at 813 kB/s)
[INFO]
[INFO] -----
[INFO] T E S T S
[INFO] -----
[INFO] Running com.mycompany.app.AppTest
[INFO] Tests run: 1, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.085 s - in com.mycompany.app.AppTest
[INFO]
[INFO] Results:
[INFO] -----
[INFO] Tests run: 1, Failures: 0, Errors: 0, Skipped: 0
[INFO]
[INFO] -----
[INFO] BUILD SUCCESS
[INFO] -----
[INFO] Total time: 20.422 s
[INFO] Finished at: 2022-12-20T15:20:41Z
[INFO] -----
```

- Global Tool Configuration. Adding Maven to the Jenkins.

The screenshot shows the Jenkins Manage Jenkins interface. On the left sidebar, under 'System Configuration', there is a 'Global Tool Configuration' section. This section contains a link to 'Configure tools, their locations and automatic installers.' A red box highlights this link. At the bottom of the page, there are 'Save' and 'Apply' buttons, which are also highlighted with red boxes.

The screenshot shows the Jenkins Global Tool Configuration page for Maven installations. It lists an existing Maven installation named 'maven-3.8.6'. Below it, there is a configuration section for adding a new Maven installation. This section includes fields for 'Name' (set to 'maven-3.8.6') and 'Version' (set to '3.8.6'). A checkbox labeled 'Install automatically' is checked. A red box highlights the 'Name' field and the 'Version' dropdown. At the bottom of the page, there are 'Add Maven' and 'Save' buttons, both of which are highlighted with red boxes.

- Create job

Configuration

General

Description
first maven job

GitHub project
Project url: https://github.com/dimdimuzun/java-test.git/

Enabled

Configuration

Source Code Management

Repository URL
https://github.com/dimdimuzun/java-test.git

Branches to build
Branch Specifier (blank for 'any'): */master

Dashboard > **java-maven-test** >

Configuration

Build Environment

Delete workspace before build starts

Use secret text(s) or file(s) ?

Send files or execute commands over SSH before the build starts ?

Send files or execute commands over SSH after the build runs ?

Add timestamps to the Console Output

Inspect build log for published build scans

SSH Agent

Terminate a build if it's stuck

With Ant ?

Build Steps

Invoke top-level Maven targets ?

Maven Version
maven-3.8.6

Goals
clean test

- Build job

Dashboard > **java-maven-test** >

Project java-maven-test

first maven job

Permalinks

- Last build (#1), 23 days ago
- Last stable build (#1), 23 days ago
- Last successful build (#1), 23 days ago
- Last completed build (#1), 23 days ago

- Console output

Jenkins

Dashboard > java-maven-test > #1

[Back to Project](#)

[Status](#)

[Changes](#)

[Console Output](#) (highlighted with a red box)

[View as plain text](#)

[Edit Build Information](#)

[Delete build '#1'](#)

[Git Build Data](#)

Console Output

Skipping 38 KB.. Full Log

```

Progress (5): 213/222 kB | 81/153 kB | 98/472 kB | 36/167 kB | 4.1/209 kB
Progress (5): 217/222 kB | 81/153 kB | 98/472 kB | 36/167 kB | 4.1/209 kB
Progress (5): 221/222 kB | 81/153 kB | 98/472 kB | 36/167 kB | 4.1/209 kB
Progress (5): 222 kB | 81/153 kB | 98/472 kB | 36/167 kB | 4.1/209 kB
Progress (5): 222 kB | 81/153 kB | 98/472 kB | 48/167 kB | 4.1/209 kB
Progress (5): 222 kB | 81/153 kB | 98/472 kB | 45/167 kB | 4.1/209 kB
Progress (5): 222 kB | 81/153 kB | 98/472 kB | 49/167 kB | 4.1/209 kB
Progress (5): 222 kB | 81/153 kB | 98/472 kB | 53/167 kB | 4.1/209 kB
Progress (5): 222 kB | 81/153 kB | 98/472 kB | 57/167 kB | 4.1/209 kB
Progress (5): 222 kB | 81/153 kB | 98/472 kB | 61/167 kB | 4.1/209 kB
Progress (5): 222 kB | 81/153 kB | 98/472 kB | 65/167 kB | 4.1/209 kB
Progress (5): 222 kB | 81/153 kB | 98/472 kB | 69/167 kB | 4.1/209 kB
Progress (5): 222 kB | 81/153 kB | 98/472 kB | 73/167 kB | 4.1/209 kB
Progress (5): 222 kB | 85/153 kB | 98/472 kB | 73/167 kB | 4.1/209 kB
Progress (5): 222 kB | 90/153 kB | 98/472 kB | 73/167 kB | 4.1/209 kB
Progress (5): 222 kB | 94/153 kB | 98/472 kB | 73/167 kB | 4.1/209 kB
Progress (5): 222 kB | 98/153 kB | 98/472 kB | 73/167 kB | 4.1/209 kB
Progress (5): 222 kB | 182/153 kB | 98/472 kB | 73/167 kB | 4.1/209 kB
Progress (5): 222 kB | 106/153 kB | 98/472 kB | 73/167 kB | 4.1/209 kB
Progress (5): 222 kB | 110/153 kB | 98/472 kB | 73/167 kB | 4.1/209 kB
Progress (5): 222 kB | 110/153 kB | 98/472 kB | 73/167 kB | 8.2/209 kB
Progress (5): 222 kB | 110/153 kB | 98/472 kB | 73/167 kB | 12/209 kB
Progress (5): 222 kB | 110/153 kB | 98/472 kB | 73/167 kB | 16/209 kB
Progress (5): 222 kB | 110/153 kB | 98/472 kB | 77/167 kB | 16/209 kB
Progress (5): 222 kB | 110/153 kB | 98/472 kB | 81/167 kB | 16/209 kB

Downloaded from central: https://repo.maven.apache.org/maven2/org/codehaus/plexus/plexus-utils/2.0.4/plexus-utils-2.0.4.jar (222 kB at 949 kB/s)
Progress (4): 110/153 kB | 102/472 kB | 81/167 kB | 16/209 kB
Progress (4): 110/153 kB | 106/472 kB | 81/167 kB | 16/209 kB
```

Downloaded from central: <https://repo.maven.apache.org/maven2/org/apache/maven/surefire/surefire-junit4/2.22.1/surefire-junit4-2.22.1.jar> (85 kB at 3.5 MB/s)

```

[INFO]
[INFO] -----
[INFO] T E S T S
[INFO] -----
[INFO] Running com.mycompany.app.AppTest
[INFO] Tests run: 1, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.09 s - in com.mycompany.app.AppTest
[INFO]
[INFO] Results:
[INFO]
[INFO] Tests run: 1, Failures: 0, Errors: 0, Skipped: 0
[INFO]
[INFO] -----
[INFO] BUILD SUCCESS
[INFO] -----
[INFO] Total time: 11.289 s
[INFO] Finished at: 2022-11-27T12:35:32Z
[INFO] -----
```

Finished: SUCCESS