**Jenkins Tutorial**

**Contents**

## [1.Jenkins Architecture Explained!!!](#_1.Jenkins_Architecture_Explained!!!)

## [2.Run Jenkins with different username](#_2.Run_Jenkins_with)

## [3. Jenkins – Role based Strategy](#_3._Jenkins_–)

## [4.Jenkins authorization - Matrix based Strategy](#_4.Jenkins_authorization_-)

## [5.Jenkins - Project matrix based strategy](#_5.Jenkins_-_Project)

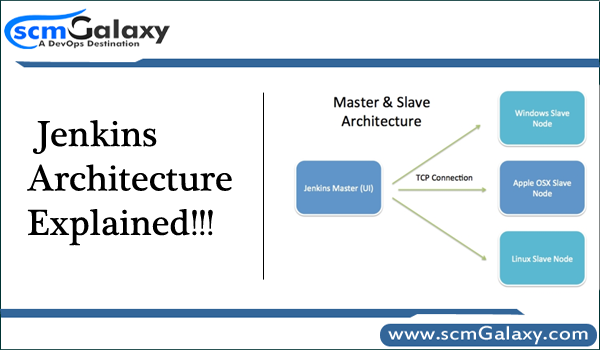
## [6.Jenkins Environment Variable](#_6.Jenkins_Environment_Variable)

## [7.How to Trigger builds remotely in Jenkins](#_7.How_to_Trigger)

## [8.How to Install Jenkins using Docker Step by step](#_8.How_to_Install)

## 9.[Jenkins - Integrating with Tomcat](#_9.Jenkins_-_Integrating)

## 1.Jenkins Architecture Explained!!!



Jenkins is continuous integration tools which help to integrate other tools to automate the software development activity and repetitive tasks. Jenkins has 2 releases. One, Community version which open source and free and another one is Jenkins Enterprise which is from Cloud bees for enterprise. Jenkins is extremely powerful with vast amount of plug-in supported. Jenkins is one of the most active open source projects so learning and finding the information is easily available on the internet. Jenkins uses master/slave architecture to manage distributed builds.  
Jenkins Architecture

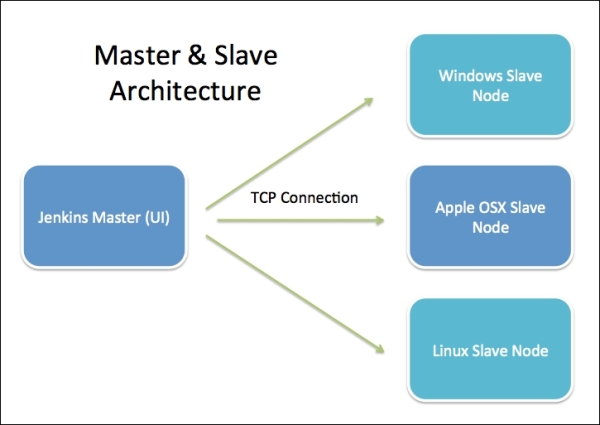
Jenkins Architecture is based on the distributed. This has 2 components.

* Jenkins Server
* Jenkins Node/Slave/Build Server

Jenkins server is a web dashboard which is nothing but powered from a war file, default run on 8080 ports. Using Dashboard, You can configure the Jobs/Projects but the build takes place in Nodes/Slave. By default one Nodes/Slave is configured and running in Jenkins Server. You can add more Nodes/Slave as well using IP address, user name, Password using the ssh/jnlp/webstart methods.

Your main Jenkins server is the master. In short, the master’s job is to handle scheduling build jobs, dispatching builds to the slaves for the actual execution, monitor the slaves (possibly taking them online and offline as required) and recording and presenting the build results. Even in a distributed architecture, a master instance of Jenkins can also execute build jobs directly.

The job of the slaves is to do as they are configured in the Jenkins Server, which involves executing build jobs dispatched by the master. You can configure a project to always run on a particular slave machine, or a particular type of slave machine, or simply let Jenkins pick the next available slave.  
Jenkins is developed using Java and Java is platform independent thus Jenkins servers and nodes/slave can be configured in any servers including Windows, Linux and Mac.



## 2.Run Jenkins with different username

1) Change the user name in the jenkins Configuration file /etc/sysconfig/jenkins. Change the user name value for the variable $JENKINS\_USER in this files. Add the new name with which you want to run the jenkins

$JENKINS\_USER="root"

2) Once the user is changed, we need to change the permission for the new user on the jenkins data directories like home, webroot and logs

chown -R root:root /var/lib/jenkins

chown -R root:root /var/cache/jenkins

chown -R root:root /var/log/jenkins

Then restarted the Jenkins jenkins and check the user has changed using a ps command

/etc/init.d/jenkins restart

ps -ef | grep jenkins

## 3. Jenkins – Role based Strategy

Jenkins provides its own user database to login but it does not have the facility to create groups/roles for the users.

If we want groups in Jenkins, we have few options

1.    Use Open LDAP with Jenkins

2.    Use Active Directory with Jenkins

3.    Use Role-based authorization strategy plugin in Jenkins

The default behavior (i.e. Can’t create group) is because it uses Jenkins user database for the security realm.

To verify this, login to Jenkins as admin, go to “Manage Jenkins”, click on “Configure Global Security”, and under the “Access Control” section, for the “Security Realm”, if you’ve selected “Jenkins’ own user database”, then you can only create users, and not groups.

There are 2 ways by which we can implement the project based authentication.

1.Project based matrix authorization strategy

2.Role based strategy

Project-based Matrix Authorization Strategy" is pre-installed and more easy to use for individual projects. "Role-based strategy"  is preferred when number of projects in Jenkins is very large. It uses pattern to match project names.

Install the Role based authorization strategy plugin

Login to Jenkins with your admin account -> Click on “Manage Jenkins” -> Click on “Manage Plugins” -> Click on “Available” tab -> Search for “role” in the Filter text box.

You’ll see “Role-based Authorization Strategy” in the results. Click on the “check-box” infront of it to select this item. Click on “Install without restart” button at the bottom

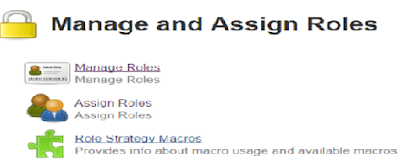
Change the Jenkins Authorization method

Once the plugin is installed, next step is to change the default Jenkins authorization method to use the role based plugin.

For this, go to “Manage Jenkins”, click on “Configure Global Security”, under the “Access Control” section, for the “Authorization”, click on “Role-Based Strategy”.

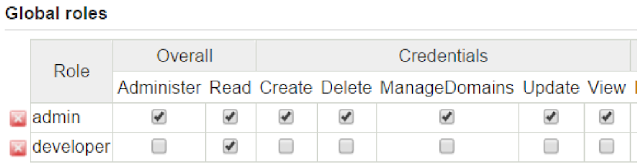
Manage and Assign Role Options

Now ,if we go to the “Manage Jenkins”, we will see a “Manage and assign Roles”.

[](https://4.bp.blogspot.com/-qp9G0dLtlh4/WaKbdszkXRI/AAAAAAAAGYY/RtUnjSIqSvwRGgPSNRYFVYmdh594aIBfgCLcBGAs/s1600/rl1.png)

Create a new global role -

Click on the “Manage roles”, from where we can create global roles that will be applicable to objects in Jenkins. The roles can be “admin”,” developer” and “Devops” etc. To add a global role, enter the role name in the “Role to add” text field and add. Once added provide the permissions to the role as below. The permissions for agents, jobs ,views are also available. Provide the correct permissions to use to give full control

[](https://4.bp.blogspot.com/-H1FZiI3U8Oo/WaKcLDGEFcI/AAAAAAAAGYg/KQ3t6ng5HQUBV_GK1hgh_JAu-rEvqRbmgCLcBGAs/s1600/rl2.png)

The following are the permissions available to be assigned to your new global role.

Overall – Administer, ConfigureUpdateCenter, Read, RunScripts, UploadPlugins

Credentials – Create, Delete, ManageDomains, Update, View

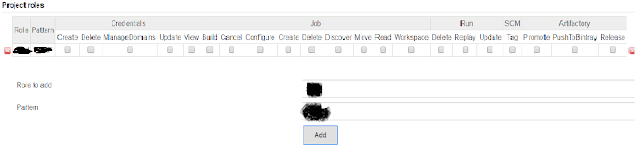
Agent – Build, Configure, Connect, Create, Delete, Disconnect, Provision

Job – Build, Cancel, Configure, Create, Delete, Discover, Move, Read, Workspace

Run – Delete, Replay, Update

View – Configure, Create, Delete, Read SCM – Tag

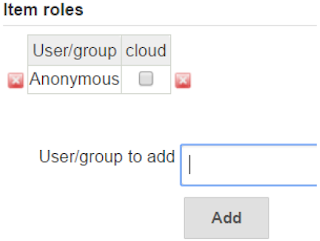
Project Roles Besides roles, we can also create roles for projects that will be applied to certain projects (jobs). For example we can create a project roles “web” which will apply only to all projects that start with the keyword “web\*”. We can create project with matching pattern so that we can allow certain users to check the jobs.

[](https://4.bp.blogspot.com/-7PtVG79zMu0/WaKdhl7OT8I/AAAAAAAAGYs/W7pN6gavOns2vYbacFGi459ZivNoBvBhQCLcBGAs/s1600/rl3.png)

Some of the notable points regarding Project roles are The regular expression “web\*” will match all the Jenkins jobs that start with “web”. If you want case-insensitive, add “(?i)” to the pattern. For example (?i)web\* will match jobs starting with both “web” and “Web”. Once the project role are added, select the permissions that you want to assign for the project role.   
Below are the permissions available   
Credentials – Create, Delete, ManageDomains, Update, View   
Job – Build, Cancel, Configure, Create, Delete, Discover, Move, Read, Workspace   
Run – Delete, Replay, Update   
SCM – Tag  
  
Assigning Users to the Roles Or Groups After creating roles with permissions we need to assign roles to users. Click the “Assign Roles” in “Manage and Assign Roles” link under “Manage Jenkins” Assign the users the appropriate roles based on their

[](https://3.bp.blogspot.com/-tZp06K5WaUM/WaKfWPtZU6I/AAAAAAAAGY4/pak8HrMU-hsvXyroUnIKNMQq_ekRf9ySQCLcBGAs/s1600/rl4.png)

In the item roles section , add the users who can access the projects that are created in the Project roles.

[](https://4.bp.blogspot.com/-1Ubu8R28jso/WaKhb3Dzf8I/AAAAAAAAGZQ/_hA05BxperkUKF5uJcEqZ2IAIZCTDs1MgCLcBGAs/s1600/rl6.png)

Even within those matched projects, users can only perform certain activities based on the permissions that we assigned for that particular project role.

## 4.Jenkins authorization - Matrix based Strategy

Allowing what users can do when then login to a Jenkins server is quite important when running jenkins server in production. Jenkins does provide multiple ways of authorization. When we go to the "Manage Jenkins" -> Configure Global security, under the Authorization tab we can see the available authorization strategies

A matrix based security allows users to be configured in such a  way that only users will specific permission can be able to perform certain actions.

In the Manage Jenkins -> "Configure Global Security". Choose the "Matrix-based security". We will see a matrix with "user/group" details and there will be a text box below that to add users

[](https://1.bp.blogspot.com/-RcWLZ5A0-8s/WavlKGuz_XI/AAAAAAAAGaE/n0xQWD5MNqIVaoD80rATtO1ngrsvibmawCEwYBhgL/s1600/auth2.png)

2. Now add the admin user first and give full permissions by selecting all checkbox. This will make sure we will have one user who have all permissions to modify things in future and also work as admin users  
  
3. Now to the "Manage Jenkins"-> "Add Users". Create a user "worker" for our session.

4. Now go back to the "Manage Jenkins" -> "Configure global security" and in the matrix based security, add the "worker" user and provide only access that are necessary. In the above case ,we have provided access to the worker user in job section as Read. We also need to provide a overall access permissions as "Read" so that the user can have overall read permissions. Now save the configuration and logout.

5. Once we login we can only see certain jobs that the user has read only permissions. we can see some thing like below,

[](https://3.bp.blogspot.com/-J48-UESPVPg/WavlKX2_SwI/AAAAAAAAGaI/-KRPuamvctojylOHwUgXxAmKwAuVU5s7wCEwYBhgL/s1600/auth3.png)

We can see jobs that are given with a read permissions. Even though we try to execute the job it wont as we dont have enough permissions for the user.

## 5.Jenkins - Project matrix based strategy

The difference between a matrix and project based authorization are in the matrix based users with permissions can be able to perform certain actions and in the case project based authorization, there will be permission on  jobs that users can perform.

Choose the Project-based matrix security under the "Configure global Security" in "manage Jenkins". Now there will be a matrix and text box to add users. Add the admin user and worker user as earlier. Give all permissions to admin user and for the worker, give him the overall read permission.

1. Save and go to the specific project where you want your users to execute. Once the authorization is moved to project based, every project will have an Option called "Enable project-based security" in the configure project as below,

**Privilege Group 1: Overall**

* Administer – This permission grants the ability to make system-wide configuration changes, as well as perform highly sensitive operations that amounts to full local system access (within the scope granted by the underlying OS.)
* ConfigureUpdateCenter – This permission allows a user to configure update sites and proxy settings
* Read – The read permission is necessary for viewing almost all pages of Jenkins. This permission is useful when you don’t want unauthenticated users to see Jenkins pages: revoke this permission from the anonymous user, then add “authenticated” pseudo-user and grant the read access.
* RunScripts – Required for running scripts inside the Jenkins process, for example via the Groovy console or Groovy CLI command.
* UploadPlugin – This permission allows a user to upload arbitrary plugins

**Privilege Group 2: Credentials**

* Create – The create permission is necessary to add credentials to a credentials provider.
* Delete – The delete permission is necessary to remove credentials stored in a credentials provider.
* Manage Domains – The manage domains permission is necessary to add/remove/configure the credential domains of a credentials provider (where the credentials provider supports multiple credential domains).
* Update – The update permission is necessary to modify credentials in a credentials provider.
* View – The view permission is necessary to view the credentials stored in a credentials provider.

**Privilege Group 3: Agent**

* Build – This permission allows users to run jobs as them on agents.
* Configure – This permission allows users to configure agents.
* Connect – This permission allows users to connect agents or mark agents as online.
* Create – This permission allows users to create agents.
* Delete – This permission allows users to delete existing agents.
* Disconnect – This permission allows users to disconnect agents or mark agents as temporarily offline.

**Privilege Group 4: Job**

* Build – This permission grants the ability to start a new build.
* Cancel – This permission grants the ability to cancel a scheduled, or abort a running, build.
* Configure – Change the configuration of a job.
* Create – Create a new job.
* Delete – Delete a job.
* Discover – This permission grants discover access to jobs. Lower than read permissions, it allows you to redirect anonymous users to the login page when they try to access a job url. Without it they would get a 404 error and wouldnt be able to discover project names.
* Read – See a job. (You may deny this permission but allow Discover to force an anonymous user to log in to see the job.)
* Workspace – This permission grants the ability to retrieve the contents of a workspace Jenkins checked out for performing builds. If you don’t want a user to access files in the workspace (e.g. source code checked out from SCM or intermediate build results) through the workspace browser, you can revoke this permission.

**Privilege Group 5: Run**

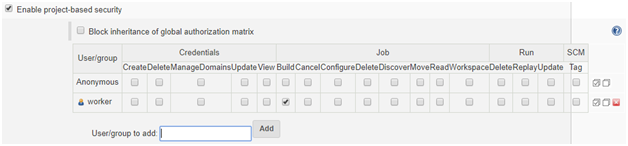
* Delete – This permission allows users to manually delete specific builds from the build history.
* Replay – Ability to perform a new Pipeline build with an edited script.
* Update – This permission allows users to update description and other properties of a build, for example to leave notes about the cause of a build failure

Privilege Group 6: View

* Configure – This permission allows users to change the configuration of views.
* Create – This permission allows users to create new views.
* Delete – This permission allows users to delete existing views.
* Read – This permission allows users to see views (implied by generic read access).

**Privilege Group 7: SCM**

* Tag – This permission allows users to create a new tag in the source code repository for a given build.

[](https://3.bp.blogspot.com/-WPyCTxtImVU/WavmbpRNAmI/AAAAAAAAGaY/lzxPSGtzGBY1UEP59T1nLjwTwM_ocphygCLcBGAs/s1600/auth4.png)

In the above case ,I have added worker user and given him the necessary permission. this needs to be done as a user who have admin access to the job so that he can give access to other users on the job

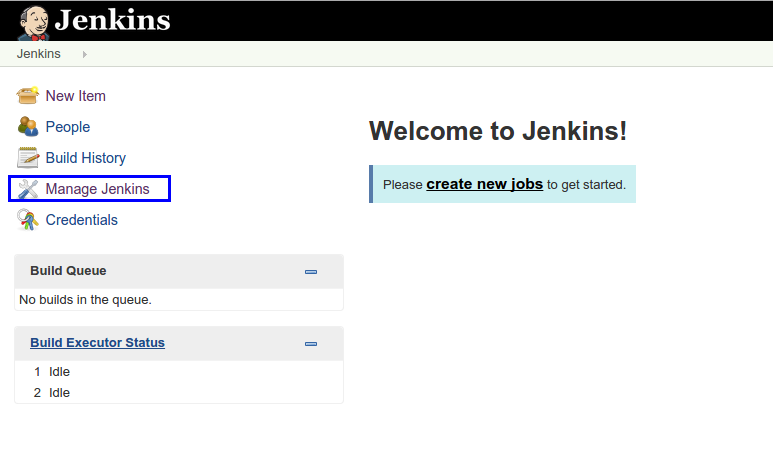
1. Now logout and login as worker to see that he can see the execute permission on the job.

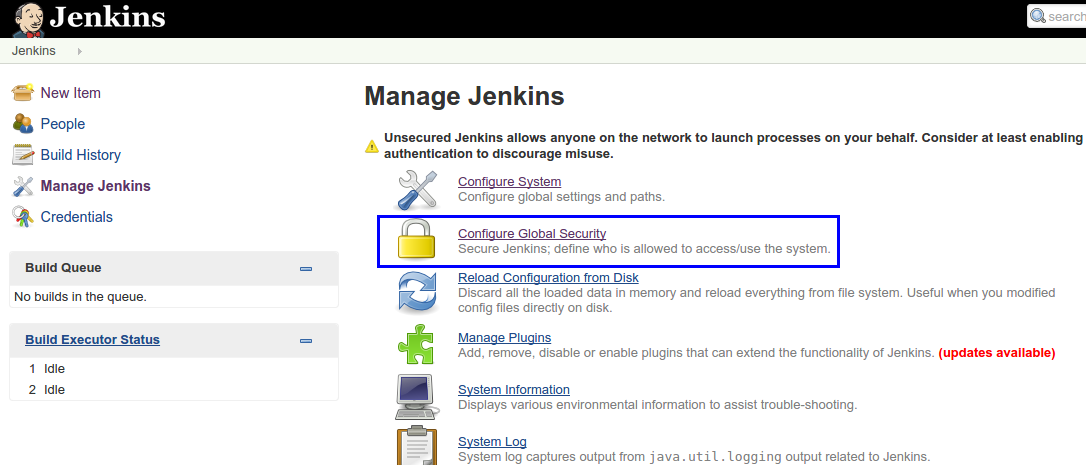
Project-based Matrix Authorization Strategy is an authorization method using which we can define which user or group can do what actions on which job. This gives us a fine-grained control over user/group permissions per project.

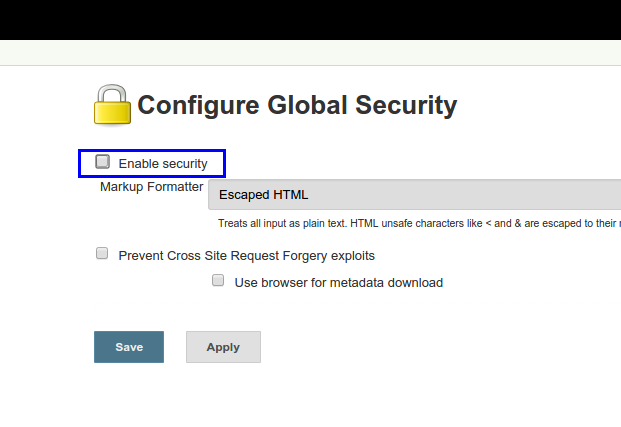
We are starting with a fresh Jenkins installation. Jenkins could be installed using the following commands:

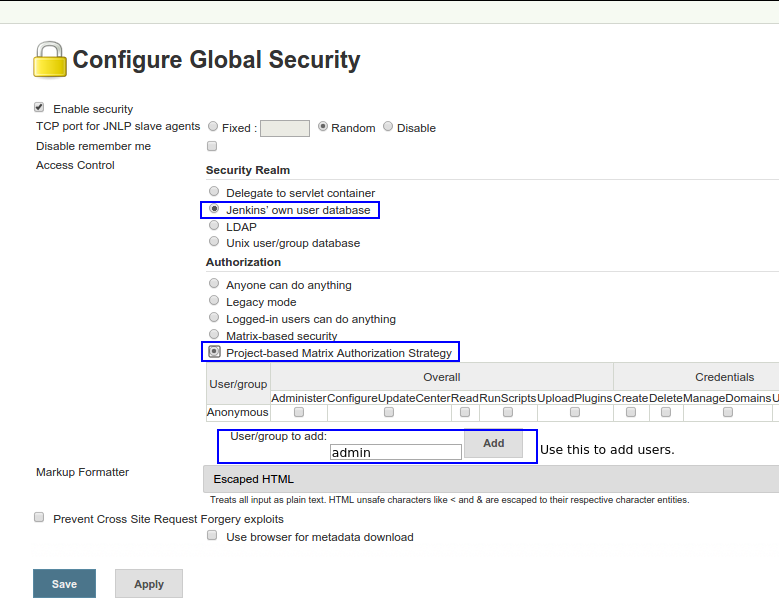
|  |  |
| --- | --- |
|  | wget -q -O - https://jenkins-ci.org/debian/jenkins-ci.org.key | sudo apt-key add -  sudo sh -c 'echo deb http://pkg.jenkins-ci.org/debian binary/ > /etc/apt/sources.list.d/jenkins.list'  sudo apt-get update  sudo apt-get install jenkins |

Jenkins uses 8080 port by default so it could be accessible browsing below link

We would be welcomed by this page:  


Now , click on “Manage Jenkins”, we would be navigated to the following page:  


Click on “Configure Global Security” to move to Global Security page:  


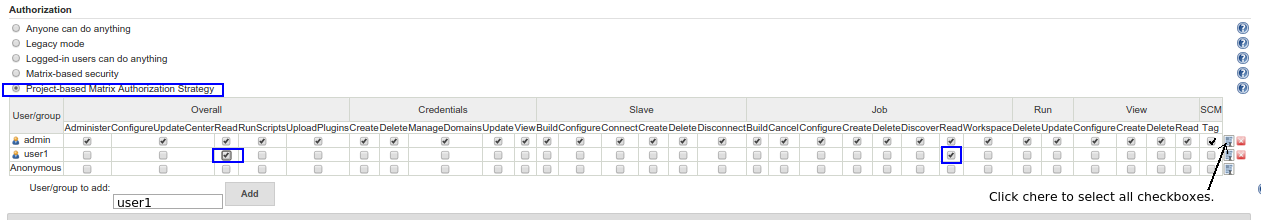
Check Enable security and more options would appear as shown below:  


As we are creating new users in Jenkins, we would be using “Jenkins’ own user database”. Select this radio button under “Security Realm”.

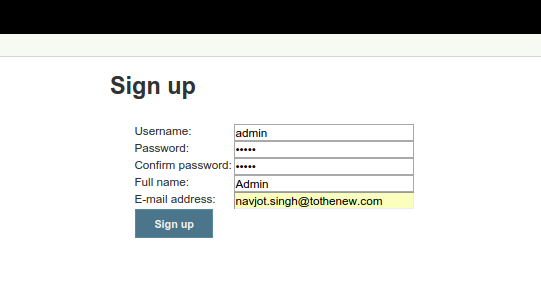
Under Authorization, select “Project-based Matrix Authorization Strategy” and add two users, one administrator (say admin) and a regular user (say user1).

All the checkboxes present besides users are for setting global permissions. Select all checkboxes against admin user to give admin full permissions.

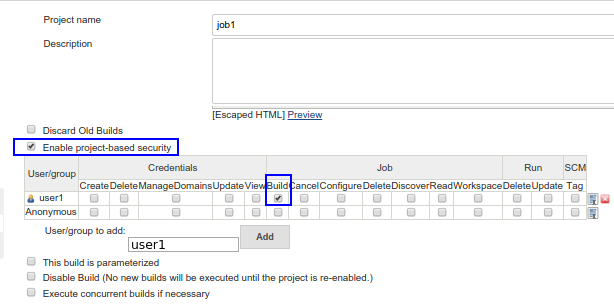
For user1, we are selecting read permissions under jobs. With this, user1 would now have read permission to view all jobs which we would be creating later on.

We have to provide read permission under “Overall” category to any regular user otherwise the user won’t be able to see anything after login.  


Save this setting and we would be navigated to login page.

Sign up with user “admin” and set a password and other details and log in using admin user.  


We can create “user1” the same way after logging out of admin.

Now, login in as admin user and create a job (say job1) and in configuration, select checkbox “Enable Project-based security” as shown below.  


Add user “user1”, and give it “build” permissions under “Job” category and save the settings. This would enable “user1” to build this job when user1 would log in.

Alternatively, if we want user1 to have build permissions for every job which admin would create later on, we can select the “build” checkbox under “Job” category in “Configure Global Security.”

Since, we have provided all permission to “admin” while configuring global security, we do not need to add per project permission for admin.

## 6.Jenkins Environment Variable

The following variables are available to shell scripts

**BRANCH\_NAME**

For a multibranch project, this will be set to the name of the branch being built, for example in case you wish to deploy to production from master but not from feature branches; if corresponding to some kind of change request, the name is generally arbitrary (refer to CHANGE\_ID and CHANGE\_TARGET).

**CHANGE\_ID**

For a multibranch project corresponding to some kind of change request, this will be set to the change ID, such as a pull request number, if supported; else unset.

**CHANGE\_URL**

For a multibranch project corresponding to some kind of change request, this will be set to the change URL, if supported; else unset.

**CHANGE\_TITLE**

For a multibranch project corresponding to some kind of change request, this will be set to the title of the change, if supported; else unset.

**CHANGE\_AUTHOR**

For a multibranch project corresponding to some kind of change request, this will be set to the username of the author of the proposed change, if supported; else unset.

**CHANGE\_AUTHOR\_DISPLAY\_NAME**

For a multibranch project corresponding to some kind of change request, this will be set to the human name of the author, if supported; else unset.

**CHANGE\_AUTHOR\_EMAIL**

For a multibranch project corresponding to some kind of change request, this will be set to the email address of the author, if supported; else unset.

**CHANGE\_TARGET**

For a multibranch project corresponding to some kind of change request, this will be set to the target or base branch to which the change could be merged, if supported; else unset.

**BUILD\_NUMBER**

The current build number, such as "153"

**BUILD\_ID**

The current build ID, identical to BUILD\_NUMBER for builds created in 1.597+, but a YYYY-MM-DD\_hh-mm-ss timestamp for older builds

**BUILD\_DISPLAY\_NAME**

The display name of the current build, which is something like "#153" by default.

**JOB\_NAME**

Name of the project of this build, such as "foo" or "foo/bar".

**JOB\_BASE\_NAME**

Short Name of the project of this build stripping off folder paths, such as "foo" for "bar/foo".

**BUILD\_TAG**

String of "jenkins-*${JOB\_NAME}*-*${BUILD\_NUMBER}*". All forward slashes ("/") in the JOB\_NAME are replaced with dashes ("-"). Convenient to put into a resource file, a jar file, etc for easier identification.

**EXECUTOR\_NUMBER**

The unique number that identifies the current executor (among executors of the same machine) that’s carrying out this build. This is the number you see in the "build executor status", except that the number starts from 0, not 1.

**NODE\_NAME**

Name of the agent if the build is on an agent, or "master" if run on master

**NODE\_LABELS**

Whitespace-separated list of labels that the node is assigned.

**WORKSPACE**

The absolute path of the directory assigned to the build as a workspace.

**JENKINS\_HOME**

The absolute path of the directory assigned on the master node for Jenkins to store data.

**JENKINS\_URL**

Full URL of Jenkins, like http://server:port/jenkins/ (note: only available if *Jenkins URL* set in system configuration)

**BUILD\_URL**

Full URL of this build, like http://server:port/jenkins/job/foo/15/ (*Jenkins URL* must be set)

**JOB\_URL**

Full URL of this job, like http://server:port/jenkins/job/foo/ (*Jenkins URL* must be set)

**GIT\_COMMIT**

The commit hash being checked out.

**GIT\_PREVIOUS\_COMMIT**

The hash of the commit last built on this branch, if any.

**GIT\_PREVIOUS\_SUCCESSFUL\_COMMIT**

The hash of the commit last successfully built on this branch, if any.

**GIT\_BRANCH**

The remote branch name, if any.

**GIT\_LOCAL\_BRANCH**

The local branch name being checked out, if applicable.

**GIT\_URL**

The remote URL. If there are multiple, will be GIT\_URL\_1, GIT\_URL\_2, etc.

**GIT\_COMMITTER\_NAME**

The configured Git committer name, if any.

**GIT\_AUTHOR\_NAME**

The configured Git author name, if any.

**GIT\_COMMITTER\_EMAIL**

The configured Git committer email, if any.

**GIT\_AUTHOR\_EMAIL**

The configured Git author email, if any.

**MERCURIAL\_REVISION**

Full ID of revision checked out.

**MERCURIAL\_REVISION\_SHORT**

Abbreviated ID of revision checked out.

**MERCURIAL\_REVISION\_NUMBER**

Number of revision checked out (not portable across clones).

**MERCURIAL\_REVISION\_BRANCH**

Branch of revision checked out, if not checking out by branch head.

**MERCURIAL\_REPOSITORY\_URL**

URL of repository.

**SVN\_REVISION**

Subversion revision number that's currently checked out to the workspace, such as "12345"

**SVN\_URL**

Subversion URL that's currently checked out to the workspace.

Sample test example

echo "BUILD\_NUMBER" :: $BUILD\_NUMBER  
echo "BUILD\_ID" :: $BUILD\_ID  
echo "BUILD\_DISPLAY\_NAME" :: $BUILD\_DISPLAY\_NAME  
echo "JOB\_NAME" :: $JOB\_NAME  
echo "JOB\_BASE\_NAME" :: $JOB\_BASE\_NAME  
echo "BUILD\_TAG" :: $BUILD\_TAG  
echo "EXECUTOR\_NUMBER" :: $EXECUTOR\_NUMBER  
echo "NODE\_NAME" :: $NODE\_NAME  
echo "NODE\_LABELS" :: $NODE\_LABELS  
echo "WORKSPACE" :: $WORKSPACE  
echo "JENKINS\_HOME" :: $JENKINS\_HOME  
echo "JENKINS\_URL" :: $JENKINS\_URL  
echo "BUILD\_URL" ::$BUILD\_URL  
echo "JOB\_URL" :: $JOB\_URL

## 7.How to Trigger builds remotely in Jenkins

1. **Create a user** – You need to create a user in jenkins using you would like to trigger a jenkins jobs from remote loction or script

How to create users in Jenkins?

**Manage Jenkins –> Manage Users –> Create User**

2. Assign a right privillage to the specific user?

How to assign privillage to the user?

**Manage Jenkins –> Configure Global Security –> Enabled “Anyone can do anything**”

OR

**Manage Jenkins –> Configure Global Security –> Configure “Matrix-based security” for the specific users and assign atleast following Permissions.**

**Overall – Read**

**Job – Build**

**Job – Read**

**Job – Workspace**

**3. Find out jenkins user “API Token”**

**How to find jenkins user “API Token”?**

“

Click on the user name located at top right(e.g <http://54.171.140.1:8080/user/admin1/>) –> Configure –> Locate the “API Token” section.

**4. Enabled “Trigger builds remotely” in Jenkins Job Configuration**.

Click on the desired job –> Configure –> Locate the “Trigger builds remotely” under “Build Triggers” Tab

Enabled the check box of “Trigger builds remotely”

Provide some Authentication Token e.g – iFBDOBhNhaxL4T9ass93HRXun2JF161Z

$ Save

**5. Formulate the command to run using curl.**

# curl –user userid:API-Token [http://IP](http://ip/) OR HOST:PORT/job/JOB\_NAME/build?token=Authentication\_Token

eg.curl –user admin1:91367cf0389eaf89669f74c9963c9fb3 <http://54.171.140.1:8080/job/ANT-BUILD/build?token=iFBDOBhNhaxL4T9ass93HRXun2JF161Z>

Some of other formats which is being tried in google but need to be tested with specific users. there are working with “Anonymous”

#curl -X POST <http://admin1:91367cf0389eaf89669f74c9963c9fb3@54.171.140.1:8080/job/ANT-BUILD/build?token=iFBDOBhNhaxL4T9ass93HRXun2JF161Z>

# wget <http://admin1:91367cf0389eaf89669f74c9963c9fb3@54.171.140.1:8080/job/ANT-BUILD/build?token=iFBDOBhNhaxL4T9ass93HRXun2JF161Z>

WORKING WITH NEW JENKINS  
# wget –auth-no-challenge –user=admin –password=5ad344f0518640f62d0483084bb889bc [http://13.126.143.49:8080/job/ANT//build?token=iFBDOBhNhaxL4T9ass93HRXun2JF161Z](http://13.126.143.49:8080/job/ANT/build?token=iFBDOBhNhaxL4T9ass93HRXun2JF161Z)

If you are using wget 1.11 against Jenkins version 1.586 and above with the JENKINS-25169 fix, you might need to use the following options:

wget –auth-no-challenge –http-user=user –http-password=apiToken –secure-protocol=TLSv1 <http://jenkins.yourcompany.com/job/your_job/build?token=TOKEN>

If you are using wget 1.11, you might need to use the following options:

wget –auth-no-challenge –http-user=user –http-password=apiToken <http://jenkins.yourcompany.com/job/your_job/build?token=TOKEN>

With wget 1.10.x the following is enough (but will not work with 1.11.x) :  
wget <http://user:apiToken@jenkins.yourcompany.com/job/your_job/build?token=TOKEN>

If you are a Windows User!

‘gitbash’ is a program combined of git and bash. A bash is shell that runs commands once you type the name of command and press enter.

Download  the git bash from here <https://git-scm.com/download/win> and install it.

## 8.How to Install Jenkins using Docker Step by step

Step 1: Installing Docker

$ apt-get install docker (Ubuntu)

$ yum install docker  (RHEL/CENTOS)

For more info, please following this <http://www.scmgalaxy.com/scm/software-containers/how-to-install-docker-in-linux.html>

Step 2:  First, pull the official jenkins image from Docker repository.

$ docker pull jenkins

**Step 3: Next, run a container using this image and map data directory from the container to the host; e.g in the example below /var/jenkins\_home from the container is mapped to jenkins/ directory from the current path on the host. Jenkins 8080 port is also exposed to the host as 49001.**

**Mapping port 8080 on the host to the container (the web ui), port 50000 to port 50000 (for build agents). Run with `-p 50000:50000` so you can connect JNLP slaves. For port 50000. This is to handle connections from JNLP based build slaves. This will store the workspace in /var/jenkins\_home. All Jenkins data lives in there including plugins and configuration.**

$ docker run -d -p 8080:8080 -p 50000:50000 jenkins

**This will store the jenkins data in /your/home on the host. Ensure that /your/home is accessible by the jenkins user in container (jenkins user – uid 1000) or use -u some\_other\_user parameter with docker run.**

$ docker run -d -p 8080:8080 -p 50000:50000 -u root -v $PWD/jenkins:/var/jenkins\_home jenkins

**Other Example:**

docker run -d -p 49001:8080 -v $PWD/jenkins:/var/jenkins\_home -t jenkins -u root

**This will store the jenkins data in /your/home on the host. Ensure that /your/home is accessible by the jenkins user in container (jenkins user – uid 1000) or use -u some\_other\_user parameter with docker run. This information is also found in the Dockerfile. So all you need to do is to ensure that the directory $PWD/jenkins is own by UID 1000:**

$ mkdir jenkins

$ chown 1000 jenkins

$ docker run -d -p 49001:8080 -v $PWD/jenkins:/var/jenkins\_home -t jenkins

How to see the Jenkins log?

$ docker exec name tail -f /var/log/jenkins/jenkins.log

Where name = --name

Step 3:  Access to j=Jenkins

As we have successfully run Jenkins Container, we can browse Jenkins Web Interface using our Web Browser by pointint to [http://ip-address:49001](http://ip-address:49001/) or [http://localhost:49001](http://localhost:49001/) according to the configuration.

## 9.Jenkins - Integrating with Tomcat

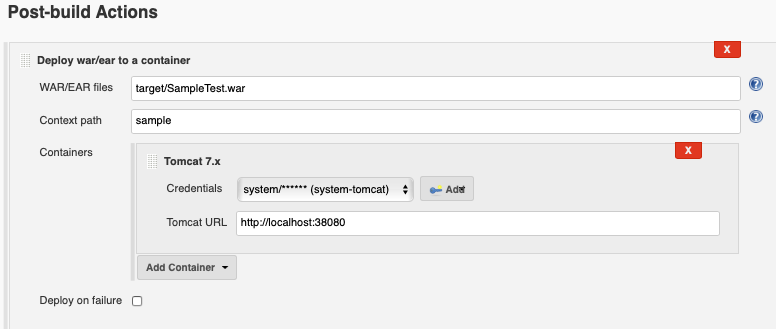
As a part of CI/CD , Some time Jenkins will be doing the part of the Deployment tool. Jenkins multiple plunges to deploy packages to different types of end points. One of the normal use case is to deploy your java war package to a web server like tomcat. In this article we will see how one can deploy a java war file to a tomcat server. For the demo purpose we will be using the “Deploy to a container” plugin.

1.Install the Plugin from the Manage Jenkins -> Package Manager

2. We don’t need to configure any thing for this plugin.

3. This plugin supports deployment to multiple containers like jboss, tomcat etc

4. Go to your maven job , and Select “Deploy ear/war to a Container” and we will see seeing a window like below attached,

****

We need to fill some of the fields in the window as,

War/Ear file location

Context path to deploy to container

Now choose the container ( tomcat 7.x here ) and fill the details of the url where that server can be accessed and also credentials to login to the machine. Once done , run the job and we can see that the war file is deployed to the tomcat and can be see with the same context that we have given.

Note - There are few issues with the tomcat roles while configuring this plugin. The role and user in tomcat should be configured correctly to enable the communication between Jenkins and Tomcat.

Below is the my configuration in the tomcat-user.xml file for more details. Iam using the “system/system” credentials configured in Jenkins for accessing the server.

 <role rolename="manager-gui"/>

<role rolename="manager-script"/>

<user username="tomcat" password="tomcat" roles="manager-script,manager-gui,admin-gui"/>

<user username="system" password="system" roles=“manager-script,admin-script,manager-gui"/>

**Step by Step guide to install Jenkins on ubuntu 16.04**

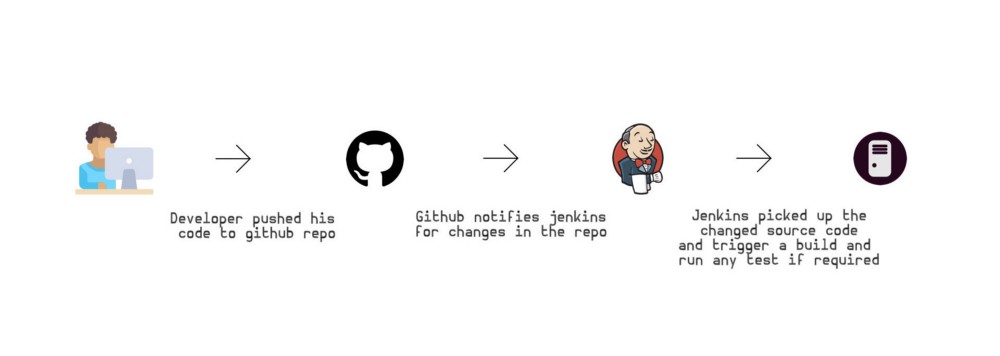
**Introduction**

[Jenkins](https://jenkins.io/) is an open source automation server intended to automate repetitive technical tasks involved in the continuous integration and delivery of software.

[*Continuous Integration*](https://www.digitalocean.com/community/tutorials/an-introduction-to-continuous-integration-delivery-and-deployment)*is a development practice that requires developers to integrate code into a shared repository at regular intervals. This concept was meant to remove the problem of finding later occurrence of issues in the build lifecycle. Continuous integration requires the developers to have frequent builds. The common practice is that whenever a code commit occurs, a build should be triggered.*

Jenkins is Java-based and can be installed from Ubuntu packages or by downloading and running its [WAR](https://en.wikipedia.org/wiki/WAR_%28file_format%29) file — used to distribute a collection of JavaServer Pages, Java Servlets, Java classes and other resources that together constitute a web application.

Jenkins will be installed on a server where the central build will take place. The following flowchart demonstrates a very simple workflow of how Jenkins works.



**Prerequisites**

Before proceeding further, you mush have a server on which we are going to install Jenkins. If you do not have a server right now, [follow this guide before proceeding further](https://www.digitalocean.com/community/tutorials/how-to-create-your-first-digitalocean-droplet-virtual-server).

*⚠️ Its recommended to start with at least 1GB of RAM.*

When the server is set up, you are ready to follow along.

**Installing Jenkins**

Install Jenkins on Ubuntu 16.04

Run the following command to add Jenkins key to the server

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

**OK**

Once you add the key, add the Jenkins repository too

**echo 'deb https://pkg.jenkins.io/debian-stable binary/' | tee -a /etc/apt/sources.list**

Update the repositories

**apt-get update**

**sudo apt-get install default-jdk**

On Debian based distros such as Ubuntu, Jenkins can be installed through the ‘apt-get’ package manager. It will install Jenkins, Java version 8 and all necessary dependencies.

apt-get install jenkins

4. Start Jenkins on Ubuntu 16.04

Once Jenkins is installed, start Jenkins

**systemctl start jenkins**

And check its status by running the following command

**systemctl status jenkins**

If everything is OK, you should receive an output similar to the following

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

Loaded: loaded (/etc/init.d/jenkins; bad; vendor preset: enabled)

Active: active (exited) since Wed 2017-09-13 10:35:57 CDT; 15s ago

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

5. Login to Jenkins

Jenkins by default is listening on port 8080, so open your favorite web browser and navigate it to http://IP\_Address:8080 to access Jenkins. The first screen will prompt you to enter and administrative password. For security reasons Jenkins creates this password and stores it to the log file and the ‘/var/lib/jenkins/secrets/initialAdminPassword’ file.

Run the following command to show the password

**cat /var/lib/jenkins/secrets/initialAdminPassword**

56cf3d2095804755b46437f5e1e9d16b

Opening the firewall

By default, Jenkins runs on port 8080, so we’ll open that port using ufw:

**sudo ufw allow 8080**

⚠️ First make sure ufw is enabled. If it is not then enable it using sudo ufw enable. You can also enable it after running above command. Want to read more about ufw [click here](https://www.digitalocean.com/community/tutorials/how-to-set-up-a-firewall-with-ufw-on-ubuntu-16-04).

We can see the new rules by checking UFW’s status.

sudo ufw status

We should see that traffic is allowed to port 8080 from anywhere:

Status: active  
   
 To Action From  
 -- ------ ----  
 OpenSSH ALLOW Anywhere  
 8080 ALLOW Anywhere  
 OpenSSH (v6) ALLOW Anywhere (v6)  
 8080 (v6) ALLOW Anywhere (v6)

Now that Jenkins is installed and the firewall allows us to access it, we can complete the initial setup.

Setting up Jenkins

To set up our installation, we’ll visit Jenkins by opening web browser and open the link below. Make sure you change ip\_address\_or\_domain\_name with your the server domain name or IP address :

http://ip\_address\_or\_domain\_name:8080

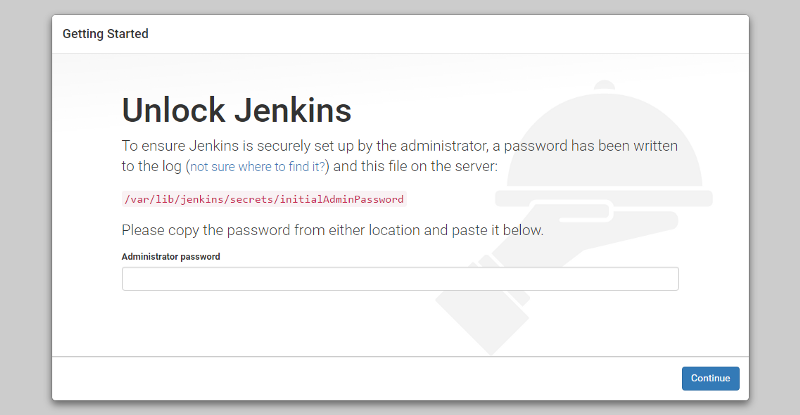
Don’t forget to use port 8080.

We should see “Unlock Jenkins” screen, which displays the location of the initial password

To change Jenkins http port listen

**vi /etc/default/Jenkins**

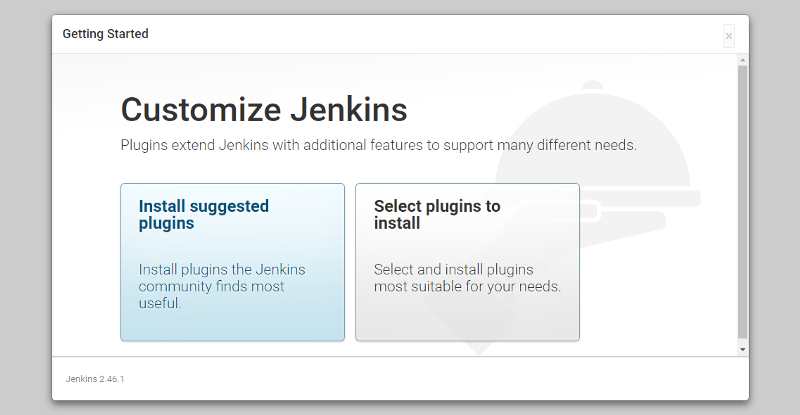
**change port 8080 –to 9090**



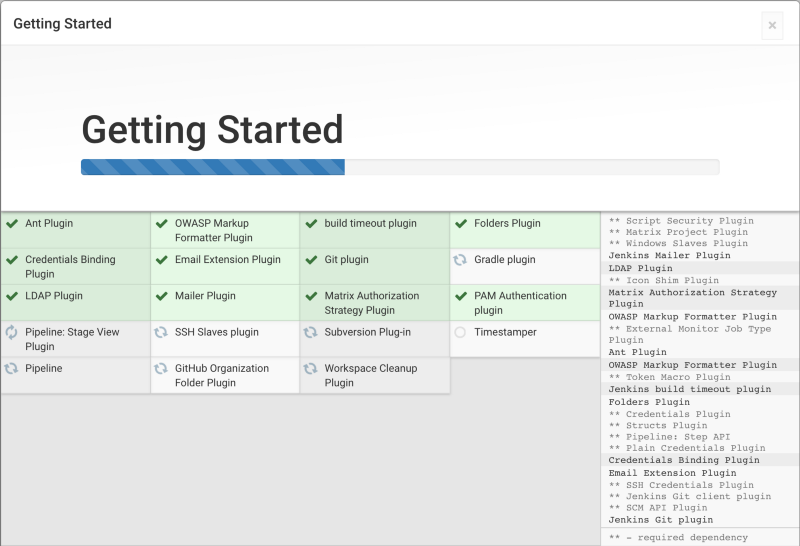
In the terminal window, we’ll use the cat command to display the password :

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

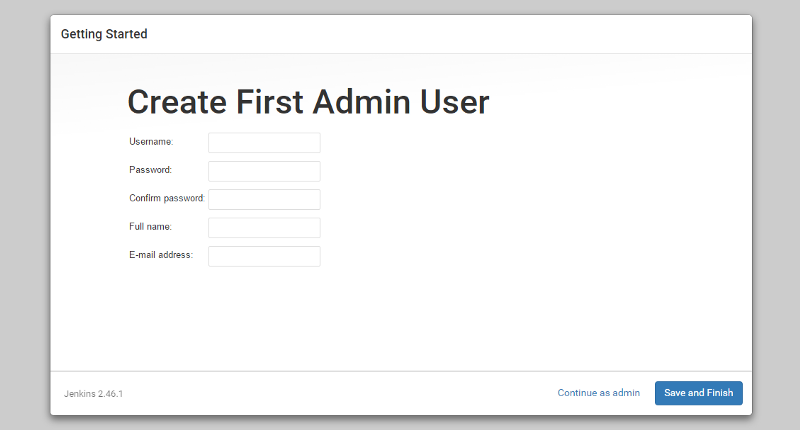
We’ll copy the 32-character alphanumeric password from the terminal and paste it into the “Administrator password” field, then click “Continue”. The next screen presents the option of installing suggested plugins or selecting specific plugins.



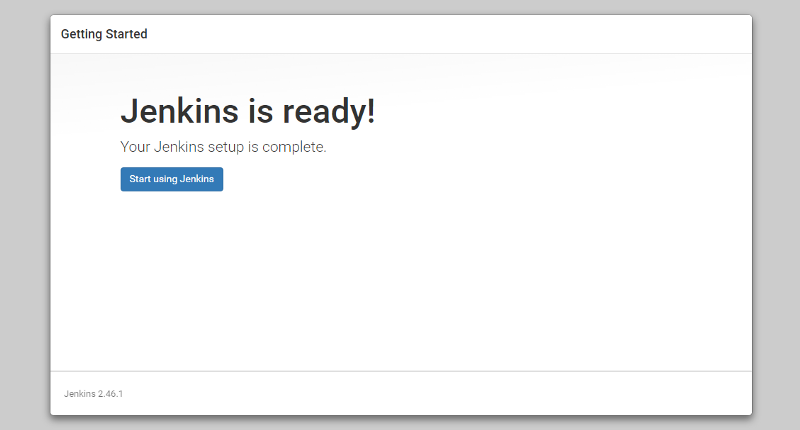
We’ll click the “Install suggested plugins” option, which will immediately begin the installation process



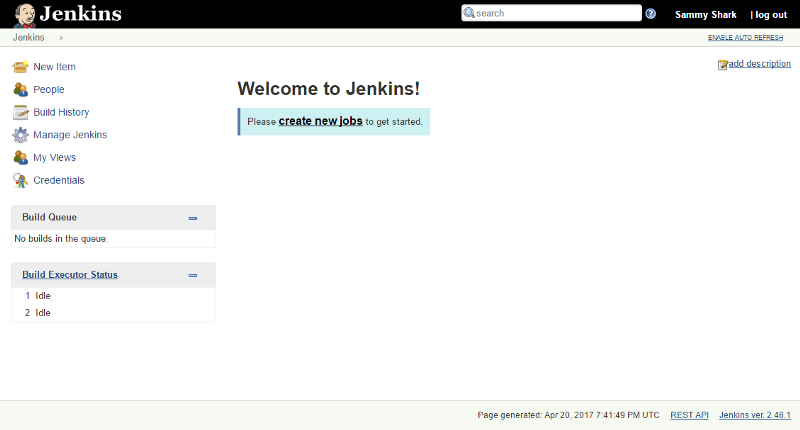
When the installation is complete, we’ll be prompted to set up the first administrative user. It’s possible to skip this step and continue as admin using the initial password we used above, but we’ll take a moment to create the user.



Once the first admin user is in place, you should see a “Jenkins is ready!” confirmation screen.



Click “Start using Jenkins” to visit the main Jenkins dashboard :



At this point, Jenkins has been successfully installed.

Samplewebsite deploy job with shell script

1.General---project name=>samplewebsitedeply

2.Source code management🡺https://smansh@bitbucket.org/smansh/devops.git

Select Credentials

3.Build---Excute shell

#/bin/bash /var/lib/jenkins/script/deploy.sh samplewebsitedeploy

4.create file in /var/lib/jenkins/script/deploy.sh

#!/bin/bash

#

jobName="$1"

servers="127.0.0.1 192.168.1.200 192.168.1.10 192.168.1.11"

for server in $servers

do

rsync -avz -e 'ssh -p 22' /var/lib/jenkins/workspace/$jobName/ root@$server:/var/www/html/

ssh -l root -p 22 $server "/etc/init.d/httpd restart"

done

------------------------------------------------------------------------------------------------------

sampleRemoteExcute job

project name-sampleRemoteExcute

Build—Excute shell script to remotehost using ssh

Ssh [site—root@122.176.59.218:2212](mailto:site—root@122.176.59.218:2212)

Command-/bin/bash/test.sh

Vi /test.sh

#!/bin/bash

clear

echo “This information is brought to you by $0.”

#Welcome the user

echo “Welcome, $USER”

echo

echo “Today is `date`.”

echo

#Currently active users.

echo “Following users are presently active:”

w | cut -d ' ' -f 1 | grep -v USER | sort -u

echo

#System information using uname command

echo “This is `uname -s` running on a `uname -m` processor.”

echo

#Information of uptime

echo “Following is the uptime information:”

uptime

echo

#Showing free memory

echo “Memory Details:”

free

echo

#Disk space usage using df command

echo “Disk Space Utilization:”

df -mh

echo

sample-triggering-post-script

project name-sample-triggering-post-script

source code management-GIT

Repositories--https://github.com/smansh/cloudtech.git

Build trigger

Copy-Authentication token like--dac397da5ead6737f1888072a669b2f8

Note-Click on Login Jenkins Username—click configure

Api token-click show API token and copy it and also userid

Create a post commit script in /.git/hooks/post-commit

#!/bin/sh

#

curl --user manoj:dac397da5ead6737f1888072a669b2f8 http://192.168.1.141:8085/job/Auto-hookstrigeer/build?token=dac397da5ead6737f1888072a669b2f8

echo "jenkins build from external script"

Select-Build when change is pushed to Github

Select –poll scm like --\* \* \* \* \*

Post-Build Action-Email Notification

[Recipients-manojs@techroutes.com](mailto:Recipients-manojs@techroutes.com)

Check-send email for every unstable build

Check-send seprate email to who broke the Build

Sample-Maven-project

Source code management—Git

Repositories-https://github.com/jenkins-docs/simple-java-maven-app.git

Build trigger-

Check Build whenever a Snapshot dependency is built

Build—Root pom—pom.xml

Goals and option—clean package

Post step

Check Run regardless of build result

Httprequestjob

Project name-httprequestjob

Build-Http request

URL--http://122.176.59.218:8080/CRM1/

Httpd mode-HEAD

Ignore ssl error-No

Jekins-configuer system

* 1. For css theme

url of css theme-http://afonsof.com/jenkins-material-theme/dist/material-teal.css

local ip to public ip using ngrok.com

Jenkins Interview Questions

Q1. What is Jenkins?

My suggestion is to start this answer by giving a definition of Jenkins.

Jenkins is an open source automation tool written in Java with plugins built for Continuous Integration purpose. Jenkins is used to build and test your software projects continuously making it easier for developers to integrate changes to the project, and making it easier for users to obtain a fresh build. It also allows you to continuously deliver your software by integrating with a large number of testing and deployment technologies.

Once you have defined Jenkins give an example, you can refer the below mentioned use case:

* First, a developer commits the code to the source code repository. Meanwhile, the Jenkins server checks the repository at regular intervals for changes.
* Soon after a commit occurs, the Jenkins server detects the changes that have occurred in the source code repository. Jenkins will pull those changes and will start preparing a new build.
* If the build fails, then the concerned team will be notified.
* If built is successful, then Jenkins deploys the built in the test server.
* After testing, Jenkins generates a feedback and then notifies the developers about the build and test results.
* It will continue to check the  source code repository for changes made in the source code and the whole process keeps on repeating.



Interviewer now knows what is Jenkins but why we use it, there are many other CI tools as well, so why Jenkins?, the next question in this Jenkins interview questions will deal with that answer.

Q2. What are the benefits of using Jenkins?

I will suggest you to include the following benefits of Jenkins, if you can recall any other benefit apart from the below mentioned points you can include that as well.

* At integration stage, build failures are cached.
* For each change in the source code an automatic build report notification is generated.
* To notify developers about build report success or failure, it is integrated with LDAP mail server.
* Achieves continuous integration agile development and test driven development.
* With simple steps, maven release project is automated.
* Easy tracking of bugs at early stage in development environment than production.

Interviewer: Okay Jenkins looks like a really cool tool, but what are the requirements for using Jenkins?

Q3. What are the pre-requisites for using Jenkins?

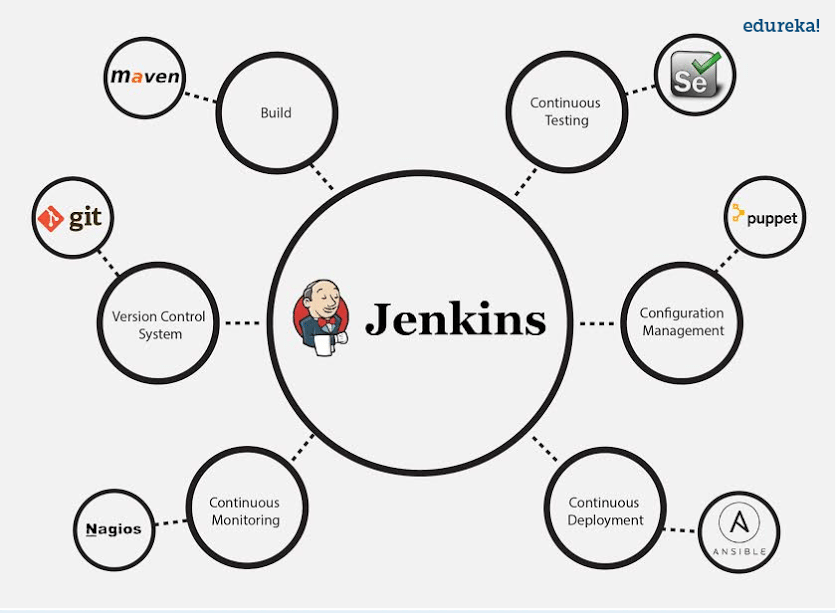
Answer to this is pretty straightforward To use Jenkins you require:

* A source code repository which is accessible, for instance, a Git repository.
* A working build script, e.g., a Maven script, checked into the repository.

Remember, you have mentioned Plugins in your previous answer, so next question in this Jenkins interview questions blog will be regarding Plugins.

Q4. Mention some of the useful plugins in Jenkins?

* Below I have mentioned some important Plugins:
* Maven 2 project
* Git
* Amazon EC2
* HTML publisher
* Copy artifact
* Join
* Green Balls



These Plugins I feel are the most useful plugins, if you want to include any other Plugin that is not mentioned above, you can add that as well, but make sure you first mention the above stated plugins and then add your own.

Q5. Mention what are the commands you can use to start Jenkins manually?

For this answer I will suggest you to go with the below mentioned flow:

To start Jenkins manually open Console/Command line, then go to your Jenkins installation directory. Over there you can use the below commands:

To start Jenkins: jenkins.exe start  
To stop Jenkins: jenkins.exe stop  
To restart Jenkins: jenkins.exe restart

Q6. Explain how you can set up Jenkins job?

My approach to this answer will be to first mention how to create Jenkins job.

Go to Jenkins top page, select “New Job”, then choose “Build a free-style software project”.

Now you can tell the elements of this freestyle job:

* Optional SCM, such as CVS or Subversion where your source code resides.
* Optional triggers to control when Jenkins will perform builds.
* Some sort of build script that performs the build (ant, maven, shell script, batch file, etc.) where the real work happens.
* Optional steps to collect information out of the build, such as archiving the artifacts and/or recording javadoc and test results.
* Optional steps to notify other people/systems with the build result, such as sending e-mails, IMs, updating issue tracker, etc..

Q7. Explain how to create a backup and copy files in Jenkins?

Answer to this question is really direct.

To create a backup all you need to do is to periodically back up your JENKINS\_HOME directory. This contains all of your build jobs configurations, your slave node configurations, and your build history. To create a back-up of your Jenkins setup, just copy this directory. You can also copy a job directory to clone or replicate a job or rename the directory.

Q8. How will you secure Jenkins?

* The way I secure Jenkins is mentioned below, if you have any other way to do it than mention that:
* Ensure global security is on.
* Ensure that Jenkins is integrated with my company’s user directory with appropriate plugin.
* Ensure that matrix/Project matrix is enabled to fine tune access.
* Automate the process of setting rights/privileges in Jenkins with custom version controlled script.
* Limit physical access to Jenkins data/folders.
* Periodically run security audits on same.

I hope you have enjoyed the above set of Jenkins interview questions, the next set of questions will be more challenging, so be prepared.

Q9 Explain how you can deploy a custom build of a core plugin?

Below are the steps to deploy a custom build of a core plugin:

Stop Jenkins.

* Copy the custom HPI to $Jenkins\_Home/plugins.
* Delete the previously expanded plugin directory.
* Make an empty file called <plugin>.hpi.pinned.
* Start Jenkins.

Q10. What is the relation between Hudson and Jenkins?

You can just say Hudson was the earlier name and version of current Jenkins. After some issue, the project name was changed from Hudson to Jenkins.

Q11. What you do when you see a broken build for your project in Jenkins?

There can be multiple answers to this question I will approach this task in the following way:

I will open the console output for the broken build and try to see if any file changes were missed. If I am unable to find the issue that way, then I will clean and update my local workspace to replicate the problem on my local and try to solve it.

If you do it in a different way then just mention that in your answer.

Q12. Explain how you can move or copy Jenkins from one server to another?

I will approach this task by copying the jobs directory from the old server to the new one. There are multiple ways to do that, I have mentioned it below:

You can:

* Move a job from one installation of Jenkins to another by simply copying the corresponding job directory.
* Make a copy of an existing job by making a clone of a job directory by a different name.
* Rename an existing job by renaming a directory. Note that if you change a job name you will need to change any other job that tries to call the renamed job.

Q13. What are the various ways in which build can be scheduled in Jenkins?

You can schedule a build in Jenkins in the following ways:

* By source code management commits
* After completion of other builds
* Can be scheduled to run at specified time ( crons )
* Manual Build Requests

Q14. What is the difference between Maven, Ant and Jenkins?

Maven and Ant are Build Technologies whereas Jenkins is a continuous integration tool.

Q15. Which SCM tools Jenkins supports?

Below are Source code management tools supported by Jenkins:

* AccuRev
* CVS,
* Subversion,
* Git,
* Mercurial,
* Perforce,
* Clearcase
* RTC

Now, the next set of Jenkins interview questions will test your experience with Jenkins.

Q16. What are the two components Jenkins is mainly integrated with?

According to me Jenkins is mainly integrated with the following:

* Version Control system like GIT,SVN.
* Build tools like Apache Maven.

If you have anything else in your mind then mention that as well but make sure you include the above two components in your answer.

Docker Interview Questions

### Q1. What are the differences between Docker and Hypervisors?

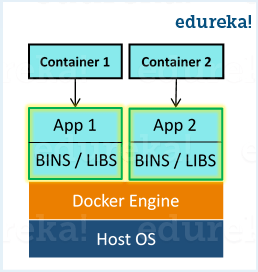
|  |  |  |
| --- | --- | --- |
| Docker vs Hypervisors | | |
| **Features** | **Hypervisors** | **Docker** |
| *Default Security Support* | To a great degree | To a slightly less degree |
| *Memory on disk required* | Complete OS plus apps | App requirement only |
| *Time Taken to start up* | Substantially longer as it requires boot of OS plus app loading | Substantially shorter as apps only need to start as the kernel is already running |
| *Portability* | Portable with proper preparation | Portable within image format; typically smaller |
| *Operating System* | Supports multiple OS | It uses the host OS |

**Q2. What is Docker?**

I will suggest you to start with a small definition of Docker.

* Docker is a containerization platform which packages your application and all its dependencies together in the form of containers so as to ensure that your application works seamlessly in any environment be it development or test or production.
* Docker containers, wrap a piece of software in a complete filesystem that contains everything needed to run: code, runtime, system tools, system libraries etc. anything that can be installed on a server.
* This guarantees that the software will always run the same, regardless of its environment.

You can refer the diagram shown below, as you can see that containers run on a single machine share the same operating system kernel, they start instantly as only apps need to start as the kernel is already running and uses less RAM.

**Note: Unlike Virtual Machines which has its own OS Docker containers uses the host OS**  

*Next set of Docker interview questions will focus on various components of Docker.*

### Q3. What is Docker image?

I will suggest you to go with the below mentioned flow:

Docker image is the source of Docker container. In other words, Docker images are used to create containers. Images are created with the build command, and they’ll produce a container when started with run. Images are stored in a Docker registry such as registry.hub.docker.com because they can become quite large, images are designed to be composed of layers of other images, allowing a minimal amount of data to be sent when transferring images over the network.  
**Tip: Be aware of Dockerhub in order to answer questions on pre-available images.**

### Q4. What is Docker container?

This is a very important question so just make sure you don’t deviate from the topic and I will advise you to follow the below mentioned format:

Docker containers include the application and all of its dependencies, but share the kernel with other containers, running as isolated processes in user space on the host operating system. Docker containers are not tied to any specific infrastructure: they run on any computer, on any infrastructure, and in any cloud.  
Now explain how to create a Docker container, Docker containers can be created by either creating a Docker image and then running it or you can use Docker images that are present on the Dockerhub.

Docker containers are basically runtime instances of Docker images.

**Q5 What is Docker hub?**

Answer to this question is pretty direct.

Docker hub is a cloud-based registry service which allows you to link to code repositories, build your images and test them, stores manually pushed images, and links to Docker cloud so you can deploy images to your hosts. It provides a centralized resource for container image discovery, distribution and change management, user and team collaboration, and workflow automation throughout the development pipeline.

Q6. How is Docker different from other container technologies?

According to me, below, points should be there in your answer:

Docker containers are easy to deploy in a cloud. It can get more applications running on the same hardware than other technologies, it makes it easy for developers to quickly create, ready-to-run containerized applications and it makes managing and deploying applications much easier. You can even share containers with your applications.  
If you have some more points to add you can do that but make sure the above the above explanation is there in your answer.

Q7. What is Docker Swarm?

You should start this answer by explaining Docker Swarn.

Docker Swarm is native clustering for Docker. It turns a pool of Docker hosts into a single, virtual Docker host. Docker Swarm serves the standard Docker API, any tool that already communicates with a Docker daemon can use Swarm to transparently scale to multiple hosts.

I will also suggest you to include some supported tools:

* Dokku
* Docker Compose
* Docker Machine
* Jenkins

Q8. What is Dockerfile used for?

This answer, according to me should begin by explaining the use of Dockerfile.

Docker can build images automatically by reading the instructions from a Dockerfile.

Now I will suggest you to give a small definition of Dockerfle.

A Dockerfile is a text document that contains all the commands a user could call on the command line to assemble an image. Using docker build users can create an automated build that executes several command-line instructions in succession.

Now, the next set of Docker interview questions will test your experience with Docker.

Q9. Can I use json instead of yaml for my compose file in Docker?

You can use json instead of yaml for your compose file, to use json file with compose, specify the filename to use for eg:  
docker-compose -f docker-compose.json up

Q10. Tell us how you have used Docker in your past position?

Explain how you have used Docker to help rapid deployment. Explain how you have scripted Docker and used Docker with other tools like Puppet, Chef or Jenkins.

If you have no past practical experience in Docker and have past experience with other tools in a similar space, be honest and explain the same. In this case, it makes sense if you can compare other tools to Docker in terms of functionality.

Q11. How to create Docker container?

I will suggest you to give a direct answer to this.

We can use Docker image to create Docker container by using the below command:

|  |  |
| --- | --- |
| 1 | docker run -t -i command name |

This command will create and start a container.

You should also add, If you want to check the list of all running container with the status on a host use the below command:

|  |  |
| --- | --- |
| 1 | docker ps -a |

Q12. How to stop and restart the Docker container?

In order to stop the Docker container you can use the below command:

|  |  |
| --- | --- |
| 1 | docker stop container ID |

Now to restart the Docker container you can use:

|  |  |
| --- | --- |
| 1 | docker restart container ID |

Q13 How far do Docker containers scale?

Large web deployments like Google and Twitter, and platform providers such as Heroku and dotCloud all run on container technology, at a scale of hundreds of thousands or even millions of containers running in parallel.

Q14. What platforms does Docker run on?

I will start this answer by saying Docker runs on only Linux and Cloud platforms and then I will mention the below vendors of Linux:

* Ubuntu 12.04, 13.04 et al
* Fedora 19/20+
* RHEL 6.5+
* CentOS 6+
* Gentoo
* ArchLinux
* openSUSE 12.3+
* CRUX 3.0+

Cloud:

* Amazon EC2
* Google Compute Engine
* Microsoft Azure
* Rackspace

Note that Docker does not run on Windows or Mac.

Q15. Do I lose my data when the Docker container exits?

You can answer this by saying, no I won’t lose my data when Docker container exits, any data that your application writes to disk gets preserved in its container until you explicitly delete the container. The file system for the container persists even after the container halts.

Q16. Mention some commonly used Docker command?

Below are some commonly used Docker commands:

