

Table of Contents

- [Table of Contents](#)
 - [Jenkins Installation](#)
 - [Jenkins Freestyle Project](#)
 - [FreeStyle Job with No Source](#)
 - [Build with Parameters](#)
 - [FreeStyle Job with Git Url Source](#)
 - [Jenkins Jobs and Jenkins Workspace](#)
 - [Jenkins Environment Variables](#)
 - [Jenkins Credentials](#)
 - [Managing access control and authorization](#)
 - [Using Project-based Matrix Authorization Strategy](#)
 - [Role-Based-Authorization Strategy](#)
 - [Audit Trail Plugin](#)
-

Jenkins Installation

- Launch an EC2 instance with Amazon Linux 2 with below **userdata**
- As Jenkins is developed in Java, the server that will run Jenkins should have Java Installed.

```
#!/bin/bash
sudo yum update -y
sudo wget -O /etc/yum.repos.d/jenkins.repo http://pkg.jenkins-
ci.org/redhat/jenkins.repo
# sudo rpm --import https://pkg.jenkins.io/redhat/jenkins.io.key
sudo rpm --import https://pkg.jenkins.io/redhat-stable/jenkins.io-2023.key
sudo yum upgrade -y
# Add required dependencies for the jenkins package
# sudo yum install java-1.8.0 -y
sudo amazon-linux-extras install epel -y
# Java 11 is required for latest Jenkins Versions
sudo amazon-linux-extras install java-openjdk11 java-devel -y
sudo yum install fontconfig git tree -y
sudo yum install jenkins -y
sudo systemctl start jenkins
sudo systemctl enable jenkins
```

--

This package installation will perform below steps:

- Setup Jenkins as a daemon launched on start.
- Create a **jenkins** Linux user to run this service.

- Set Jenkins to listen on port **8080**. Access this port with your browser to start configuration.
- Login to EC2 Jenkins Server using ssh.

```
netstat -nltp
sudo hostnamectl set-hostname jenkins.example.com
sudo service jenkins status
[ec2-user@ip-172-31-27-182 ~]$ sudo service jenkins status
Redirecting to /bin/systemctl status jenkins.service
• jenkins.service - Jenkins Continuous Integration Server
   Loaded: loaded (/usr/lib/systemd/system/jenkins.service; enabled; vendor
   preset: disabled)
   Active: active (running) since Sat 2023-08-05 02:58:51 UTC; 1min 8s ago
   Main PID: 3718 (java)
   CGroup: /system.slice/jenkins.service
           └─3718 /usr/bin/java -Djava.awt.headless=true -jar
           /usr/share/java/jenkins.war --webroot=%C/jenkins/war --httpPort=8080

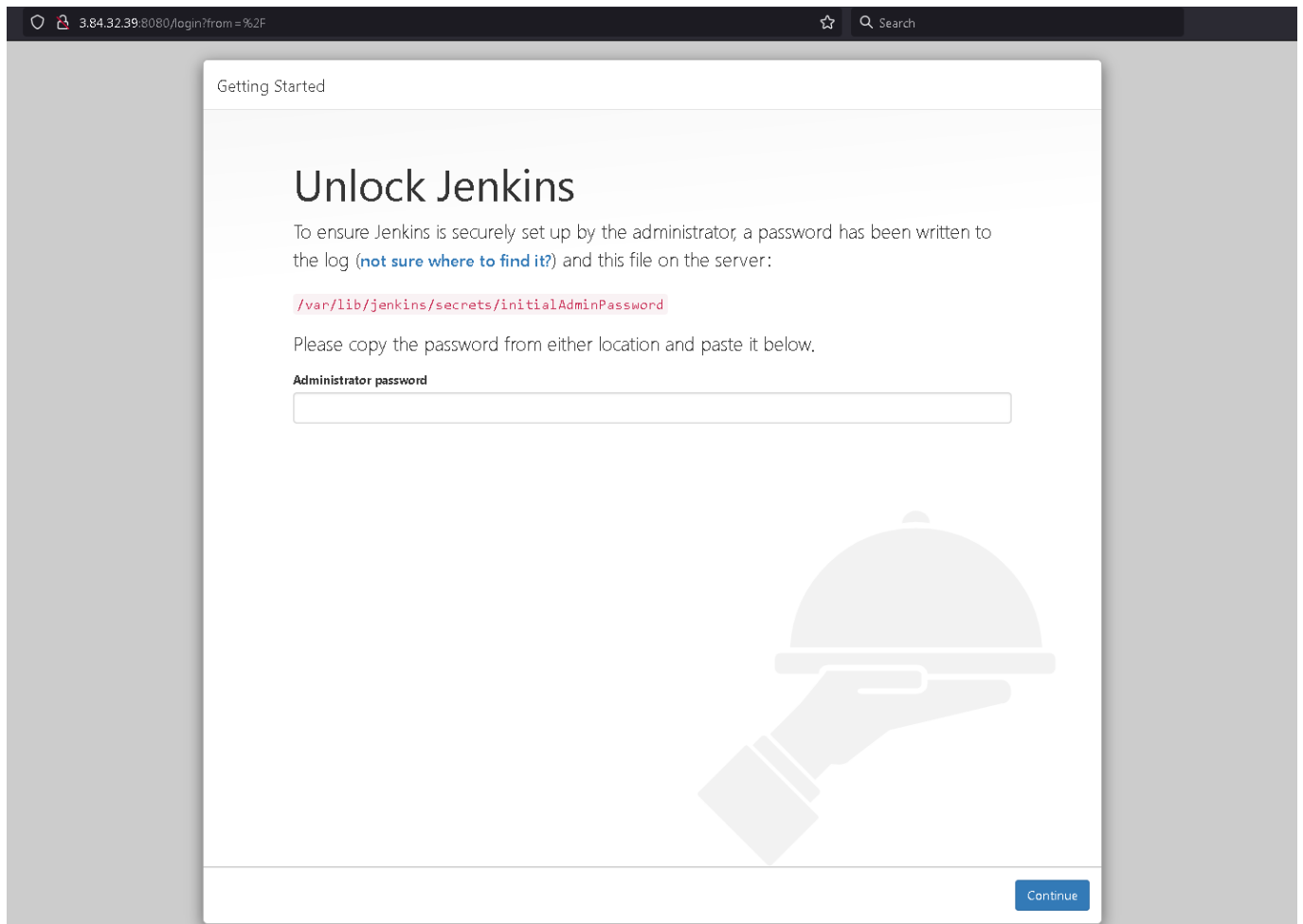
Aug 05 02:58:18 ip-172-31-27-182.ap-south-1.compute.internal jenkins[3718]:
ad071e10fbb24ad7853418f993fac8ca
Aug 05 02:58:18 ip-172-31-27-182.ap-south-1.compute.internal jenkins[3718]: This
may also be found at: /var/lib/jenkins/secrets/initialAdminPassword
Aug 05 02:58:18 ip-172-31-27-182.ap-south-1.compute.internal jenkins[3718]:
*****
Aug 05 02:58:18 ip-172-31-27-182.ap-south-1.compute.internal jenkins[3718]:
*****
Aug 05 02:58:18 ip-172-31-27-182.ap-south-1.compute.internal jenkins[3718]:
*****
Aug 05 02:58:51 ip-172-31-27-182.ap-south-1.compute.internal jenkins[3718]: 2023-
08-05 02:58:51.843+0000 [id=32]          INFO
jenkins.InitReactorRunner$1#on...ization
Aug 05 02:58:51 ip-172-31-27-182.ap-south-1.compute.internal jenkins[3718]: 2023-
08-05 02:58:51.864+0000 [id=25]          INFO
hudson.lifecycle.Lifecycle#onR...running
Aug 05 02:58:51 ip-172-31-27-182.ap-south-1.compute.internal systemd[1]: Started
Jenkins Continuous Integration Server.
Aug 05 02:58:52 ip-172-31-27-182.ap-south-1.compute.internal jenkins[3718]: 2023-
08-05 02:58:52.720+0000 [id=47]          INFO
h.m.DownloadService$Downloadab...staller
Aug 05 02:58:52 ip-172-31-27-182.ap-south-1.compute.internal jenkins[3718]: 2023-
08-05 02:58:52.721+0000 [id=47]          INFO          hudson.util.Retrier#start:
Per...empt #1
Hint: Some lines were ellipsized, use -l to show in full.
sudo service jenkins stop
sudo service jenkins restart
```

--

- Check Jenkins Port Information

```
ps -elf | grep jenkins
ps -elf | grep 8080
#Access the Jenkins UI
http://public-ip:8080
```

--



--

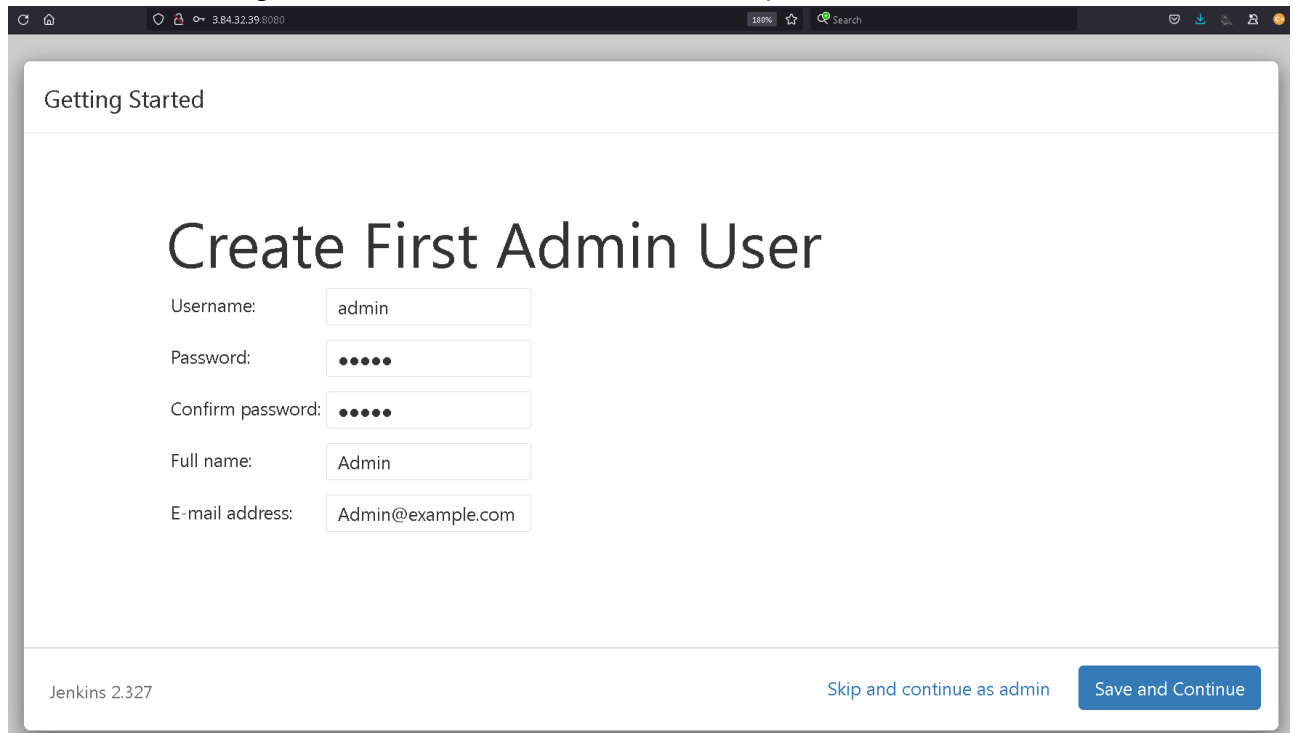
- Admin Password is written to a file, copy output of below command and enter in the browser.

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

- On next page, Select **Install Suggested Plugins** only.
 - Here, Jenkins will Install Plugins that can be used be later

--

- It will ask for creating for first **Admin** user, enter details as required.



The screenshot shows the Jenkins 'Getting Started' page. The main heading is 'Create First Admin User'. Below it, there are five input fields: 'Username' (filled with 'admin'), 'Password' (filled with five dots), 'Confirm password' (filled with five dots), 'Full name' (filled with 'Admin'), and 'E-mail address' (filled with 'Admin@example.com'). At the bottom left, it says 'Jenkins 2.327'. At the bottom right, there are two buttons: 'Skip and continue as admin' and 'Save and Continue'.

--

- In Linux SSH Session of Jenkins Server, A linux user with name **jenkins** is created while Jenkins Installation, add **jenkins** user in linux to **sudoers** group.
- All Jobs in jenkins are executed by **jenkins** linux user

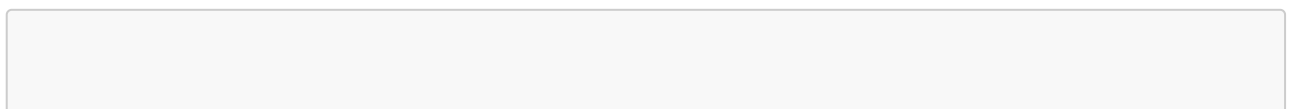
```
cat /etc/passwd | grep -i 'jenkins'
jenkins:x:995:993:Jenkins Automation Server:/var/lib/jenkins:/bin/false
id jenkins
sudo usermod -a -G wheel jenkins
id jenkins
# Current home directory for jenkins is /var/lib/jenkins
ls -ltr /var/lib/jenkins
```

- To view the Jenkins Systems Information, navigate to **Manage Jenkins > System Information**

Jenkins Freestyle Project

FreeStyle Job with No Source

- Click on **New Item** then enter an item name, select **Freestyle project**.
- Under **Source Code Management** Section, select **None**
- Under **Build Step > Add Build Step > Execute Shell** , Select enter below bash commands to be executed in the freestyle project.



An empty rectangular box intended for pasting the bash commands to be executed in the Jenkins Freestyle project.

```
id
echo "This is Jenkins FreeStyle Job"
printenv
ls -ltr
```

- Click on **Apply** and then **Save**
- Click on **Build Now** to start execution of Job.
- For every Jenkins Job execution, Jenkins creates a Build Number **#1** for first build. Click on **#1** and **Console Output**.
- Make multiple runs of the Job.

The screenshot shows the Jenkins web interface. At the top, there's a navigation bar with the Jenkins logo, a search bar, and user information (admin, log out). Below the navigation bar, the breadcrumb trail reads 'Dashboard > jenkins-freestyle-job > #1'. On the left sidebar, there are links for 'Back to Project', 'Status', 'Changes', 'Console Output' (which is highlighted), 'Edit Build Information', 'Delete build #1', and 'Next Build'. The main content area is titled 'Console Output' with a green checkmark icon. It displays the output of a build started by user 'admin'. The output text includes system information, workspace paths, Jenkins URL, and various environment variables like 'BUILD_TAG', 'WORKSPACE', 'JOB_URL', 'RUN_CHANGES_DISPLAY_URL', 'USER', 'BUILD_ID', 'NOTIFY_SOCKET', 'JOB_BASE_NAME', 'RUN_TESTS_DISPLAY_URL', 'SHLVL', 'HOME', 'CI=true', 'EXECUTOR_NUMBER', 'JENKINS_SERVER_COOKIE', 'WORKSPACE_TMP', 'NODE_LABELS', 'LOGNAME', 'HUDSON_HOME', 'NODE_NAME', 'JOB_DISPLAY_URL', 'BUILD_NUMBER', and 'HUDSON_COOKIE'.

Build with Parameters

- A Job in Jenkins supports **Runtime Parameters** that can be passed while executing the Job.
- This can be used to run a FreeStyle Job/Pipeline Job as per **SDLC Environment** or any other value to be passed on Job Runtime.
- Under a specific jenkins project, select **Configure** option, select the checkbox **This project is parameterized** and **Add Parameter**.
 - Add a **Choice Parameter** with parameter name as **EnvironmentName** and enter values
 - dev
 - qa
 - prod
 - test
- For testing the value of the runtime parameter, keep **Source Code Management** as **None**
- In **Execute Shell** add a below:

```
echo "This is the User Input Value for EnvironmentName parameter :  
$EnvironmentName"
```

- The parameters are available as **environment variables**. So a shell **\$PARAM_NAME**, can be used to access these values.
- Click on **Build Now** to start execution of Job.

--

FreeStyle Job with Git Url Source

- Under **Source Code Management** Section: Provide the Github Repository URL where Source Code is present, keep the branch as **master** or **main**.
- Go to **Jenkins Project -> Configure -> Under Build Environment Build Step > Select Execute Shell Script from dropdown > write shell commands**
- Click on **Build Now** to Build this Project.

Execute a shell script stored in Github repo by providing path.

- Create a shell script in Github, execute the shell script in Jenkins Job.
 - **bash test.sh**

--

Jenkins Jobs and Jenkins Workspace

The Jenkins home directory structure

Directory	Description
jobs	Path /var/lib/jenkins/jobs . It contains configuration details about the build jobs that Jenkins manages, as well as the artifacts and data resulting from these builds. Contains execution build log history.
workspace	Path /var/lib/jenkins/workspace . It is where Jenkins builds your project: it contains the source code Jenkins checks out, plus any files generated by the build itself. This workspace is reused for each successive build, there is only ever one workspace directory per project, and the disk space it requires tends to be relatively stable.

```
Building in workspace /var/lib/jenkins/workspace/jenkins-freestyle-demo-source
```

--

Jenkins Environment Variables

- To view all the environment variables simply append **env-vars.html** to your Jenkins Server's URL. For e.g **http://<JENKINS_IP>:8080/env-vars.html**

- Create a simple free style job to display the value of the environment variables that are set for a Jenkins Job:
- Under Build Section > Add build step > Execute shell , add below commands:

```
echo "BUILD_NUMBER" :: $BUILD_NUMBER
echo "BUILD_ID" :: $BUILD_ID
echo "BUILD_DISPLAY_NAME" :: $BUILD_DISPLAY_NAME
echo "JOB_NAME" :: $JOB_NAME
echo "WORKSPACE" :: $WORKSPACE
echo "JENKINS_HOME" :: $JENKINS_HOME
echo "JENKINS_URL" :: $JENKINS_URL
echo "BUILD_URL" :: $BUILD_URL
echo "JOB_URL" :: $JOB_URL
echo "GIT_COMMIT" :: $GIT_COMMIT
echo "GIT_BRANCH" :: $GIT_BRANCH
echo "GIT_URL" :: $GIT_URL
echo "Below output is all the environment variable in Jenkins"
printenv
```

- The **printenv** command prints all the Jenkins Environment Variables set for that specific Build.

--

- Executing Shell Script using Jenkins
 - Write a shell script and commit in Github Repo Branch.
 - Mention the **bash scriptname.sh** that is available in your Github Repo, add this in the Build Scripts in Job.

Jenkins Credentials

- Jenkins configuration to access private repo using HTTPS Clone URL.
 - In Github, navigate to: **Github Account > Settings > Developer settings > Personal access tokens > Generate new token**. Add this value in Jenkins Credentials.
 - Navigate to **Jenkins dashboard -> Manage Jenkins > Under Security select Credentials > System > Global credentials > Add credentials**.
 - Under UserName, Enter Git Username
 - Under Password, enter PAT Token
 - Under ID, Enter Unique Identifier
 - From dropdown select **Username with password** and specify the **ID , Username** and configure the **Password**, in this field enter the **Github PAT Token** created in the previous step.
 - Navigate to Job, Under the **Source Code Management** , under **Credentials**, select the **ID** created earlier

--

The screenshot shows the 'New credentials' form in Jenkins. The 'Kind' dropdown is set to 'Username with password'. The 'Scope' dropdown is set to 'Global (Jenkins, nodes, items, all child items, etc)'. The 'Username' field contains 'jenkins-https-pat-token'. The 'Treat username as secret' checkbox is checked. The 'Password' field is masked with dots. The 'ID' field contains 'jenkins-https-pat-token'. The 'Description' field contains 'jenkins-https-pat-token'. A 'Create' button is at the bottom.

- While setting up a Job in Jenkins, add the credentials created above to the credentials section in **Source Code Management** under the **Repository URL**.

The screenshot shows the 'Source Code Management' configuration page in Jenkins. The 'General' tab is selected. Under 'Source Code Management', the 'Git' radio button is selected. The 'Repositories' section is expanded, showing a 'Repository URL' field with 'https://github.com/cloudmllops/git-practical.git'. The 'Credentials' dropdown is set to 'jenkins-https-pat-token'. There are '+ Add' and 'Advanced...' buttons. Below the 'Repositories' section is an 'Add Repository' button. The 'Branches to build' section is expanded, showing a 'Branch Specifier (blank for \'any\')' field with '*/master'. At the bottom are 'Save' and 'Apply' buttons.

- Execute the Job and test whether Jenkins Job is able to check out Github Specified Branch.
- Validate the Job Execution Details.

- Jenkins configuration to access private repo using **SSH Clone URL**.
 - Use below steps for Github SSH Keys Configuration and Jenkins Github SSH Integration.

- Generate ssh keys and add to **Github Account OR Specific Github Repo**

```
ssh-keygen -t rsa -b 4096 -C "your_email@example.com"
```

- This creates a new ssh key pair, using the provided email as a label.
- SSH Keys can be configured as per below:

Github Account SSH Keys

Github Repository SSH Keys

Github UI > Settings > SSH and GPG Keys > New SSH key > Add SSH Key > Add Public Key Content > Confirm password

Github Repository > Settings > Deploy keys > Add Deploy Key > Enter Name and Public ssh key Allow Write Access > Add Key

- Navigate to **Jenkins dashboard -> Managed Jenkins > Manager > Jenkins > Credentials -> System -> Global credentials -> Add credentials.**
- From dropdown select **SSH Username with Private Key** and specify the **ID , Username** and configure the **SSH Private Key** which is stored in **.ssh** folder under the file name **id_rsa**.
- While setting up a Job in Jenkins, add the credentials created above to the credentials section in **Source Code Management** under the **Repository URL**.
 - Check for <https://plugins.jenkins.io/git-client/#plugin-content-ssh-host-key-verification>
- Execute the Job and test whether Jenkins Job is able to check out Github Specified Branch

Managing access control and authorization

- Creating Users in Jenkins : Select **Manage Jenkins > Under Security > Users > Create a user > Provide Username and Required details**

Using Project-based Matrix Authorization Strategy

- Lets configure some users in Jenkins, create a read only user **readonlyuser**
- Go to **Manage Jenkins > Security > under Authorization > Select Project-based Matrix Authorization Strategy**
 - Add **User/Group** configure security based on different sections such as **Overall,Credentials,Job,View** and so on.
 - Click on **Apply and Save**.

--

This allows you to say things like "Joe can access project A, B, and C but he can't see D." See the help of "Matrix-based security" for the concept of matrix-based security in general.


ACLs are additive, so the access rights granted below will be effective for all the projects.

(from [Matrix Authorization Strategy Plugin](#))

User/group	Overall	Credentials	Agent	Job	Run	View	SCM	Lockable Resources										
	Administer	Create Read Delete	Manage Domains Update View	Configure Build	Connect	Create Delete	Discover Move	Read Workspace	Delete Replay Update Configure	Create Delete Read	Tag Reserve	Steal Unlock	View					
Anonymous	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>				
Authenticated Users	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>			
test user	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>			
<div>Add user...</div>	<div>Add group...</div>	<div>?</div>																

- Use another browser and try to access the Jenkins dashboard with a newly added user. This User will not be able to see all the options in Jenkins Dashboard.

—


Jenkins

test user
log out

Dashboard

People

Build History

Project Relationship

Check File Fingerprint

My Views

Build Queue

Build Executor Status

All

S	W	Name	Last Success	Last Failure	Last Duration
		jenkins-freestyle-job	12 hr #12	N/A	2.5 sec
		jenkins-pipeline-test-job	6 days 12 hr #20	6 days 12 hr #18	5.9 sec

Icon: S M L

Icon legend


Atom feed for all


Atom feed for failures

Atom feed for just latest builds

— —

- Here, the user is not able to see the **Configure** Option under a specific Job, because Job Specific Permissions are granted.

 Jenkins

test user  log out

Dashboardjenkins-freestyle-job

Back to Dashboard

Status

Changes

Build with Parameters

GitHub Hook Log

Build History trend

#12

Mar 2, 2022 3:47 AM

#11

Mar 2, 2022 3:43 AM

#10

Mar 2, 2022 3:26 AM

#9

Mar 2, 2022 3:24 AM

#8


Mar 2, 2022 3:02 AM

#7

Feb 22, 2022 3:26 AM

Project jenkins-freestyle-job

This is Freestyle Job

 Recent Changes

Permalinks

- Last build (#12), 12 hr ago
- Last stable build (#12), 12 hr ago
- Last successful build (#12), 12 hr ago
- Last completed build (#12), 12 hr ago

Role-Based-Authorization Strategy

- **Manage Jenkins > Under System Configuration > Select Plugins > Under Available Tab > Install the Role-based Authorization Strategy Plugin.**
- Add plugin from available tab in Plugins Manager i.e **Role Based Authorized Strategy**
- To enable **Role-Based Strategy**, Go to **Manage Jenkins > Security > under Authorization > Select Role-based Strategy**

DashboardConfigure Global Security

Authorization

☐ Anyone can do anything ?

☐ Legacy mode ?

☐ Logged-in users can do anything ?





☐ Matrix-based security ?

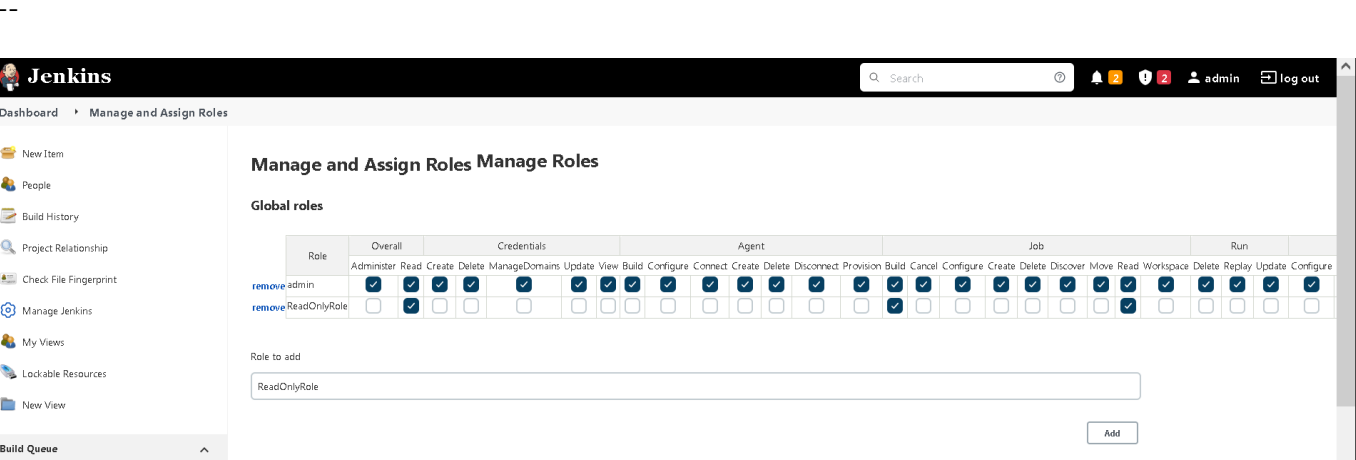
☐ Project-based Matrix Authorization Strategy ?

☒ Role-Based Strategy ?

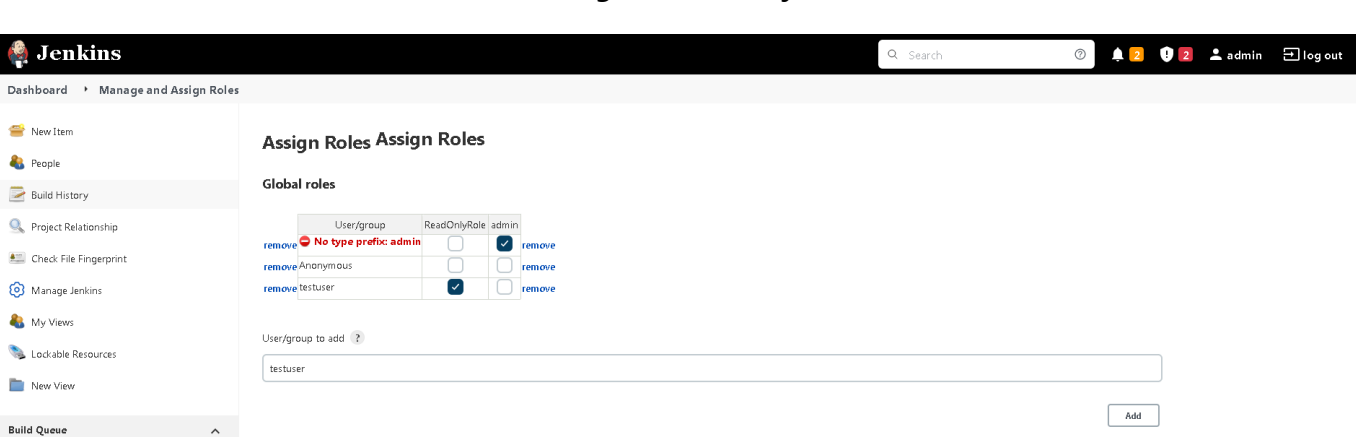
- To Create a Role, Go to **Manage Jenkins > Manage and Assign Roles > Manage Roles > Role to add > Enter ReadOnlyRole.**

Security

-  **Configure Global Security**
Secure Jenkins; define who is allowed to access/use the system.
-  **Manage Credentials**
Configure credentials
-  **Manage and Assign Roles**
Handle permissions by creating roles and assigning them to users/groups
-  **Manage Users**
Create/delete/modify users that can log in to this Jenkins



- Click on **Apply and Save.** --
- To Assign this Role to a user, Go to **Manage Jenkins > Manage and Assign Roles > Assign Roles > Enter Username > Select Role to be assigned ReadOnlyRole.**



- Use another browser and try to access the Jenkins dashboard with a user to check permissions in the Role are applied.
- Item Roles:
 - This can be created for specific Jobs access and can be assigned to user/group.

Audit Trail Plugin

- **Manage Jenkins > Manage Plugins > Install the *Audit Trail* Plugin.**
- Go to **Manage Jenkins** > Under **System Configuration** > select **System** > **Audit Trail** > **Add Logger** > Select **Log File**
- Provide the Log Location as ***/var/lib/jenkins/log/audit-%g.log***, provide Log File Size as **50** and Log File Count **10**, enter File Separator as **|**
- After executing some build job for some Jenkins Project, check the content of the audit file.

```
ls -ltr /var/lib/jenkins/log/
```

Audit Trail Plugin keeps a log of users who performed particular Jenkins operations, such as configuring jobs, executing jobs. This plugin adds an Audit Trail section in the main Jenkins configuration page. Here you can configure log location and settings (file size and number of rotating log files), and a URI pattern for requests to be logged. The default options select most actions with significant effect such as creating/configuring/deleting jobs and views or delete/save-forever/start a build. The log is written to disk as configured and recent entries can also be viewed in the Manage / System Log section.