

Jenkins: Introduction - Understanding continuous integration

- Developers pushes software to a repository
- Operations **builds** and deploys the application to one or many environments like testing staging)
- QA team performs performance tests and **releases** it to production use
- Continuous Integration (CI) is the process of **automating** the build and testing of code every time a team member **commits** changes to Version Control System

Jenkins: Introduction - Understanding continuous integration

1. Developers pushes software to a repository
2. Operations **builds** and deploys the application to one or many environments like testing staging)
3. QA team performs performance tests and **releases** it to production use



Can automate most of the repetitive tasks! This facilitates **continuous integration**!

Jenkins: Introduction - Understanding continuous integration

A **Build Pipeline** Components

1. Unit Test
2. Acceptance Test
3. Packaging
4. Reporting
5. Deployment
6. Notification/Alerts



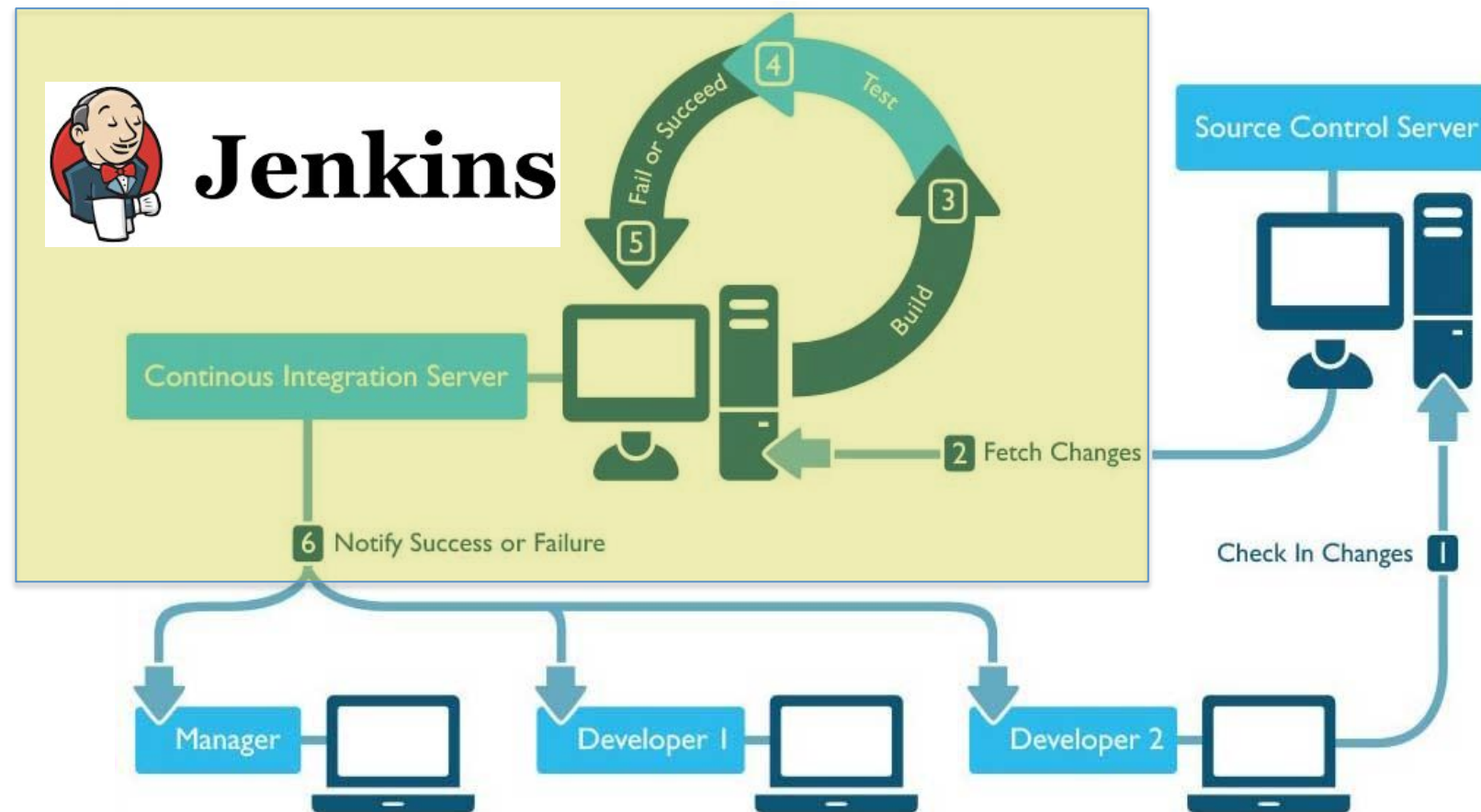
Can automate most of the repetitive tasks! This facilitates **continuous integration**!

Jenkins: Introduction - Understanding continuous integration



Jenkins

Jenkins: Introduction - Understanding continuous integration

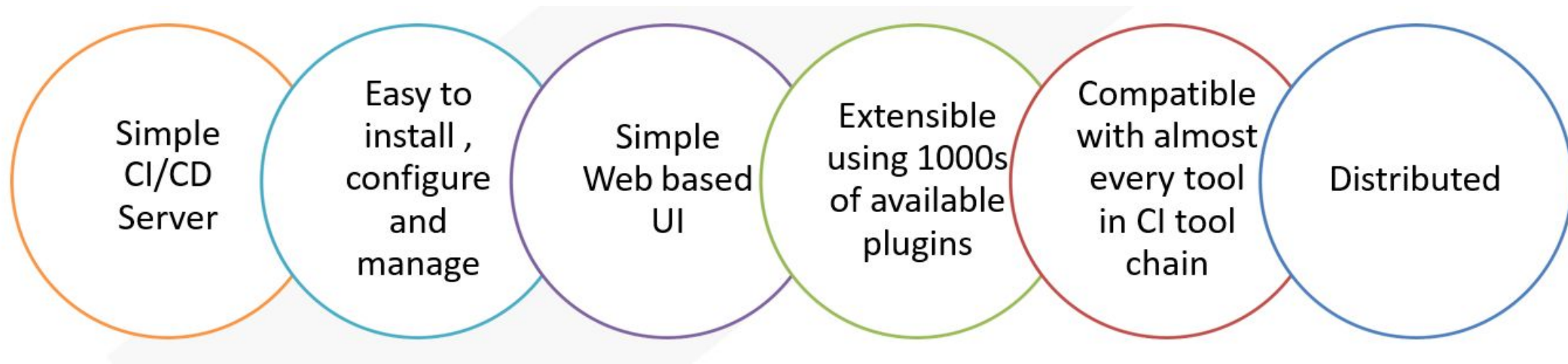


Jenkins: Introduction - Introduction about Jenkins

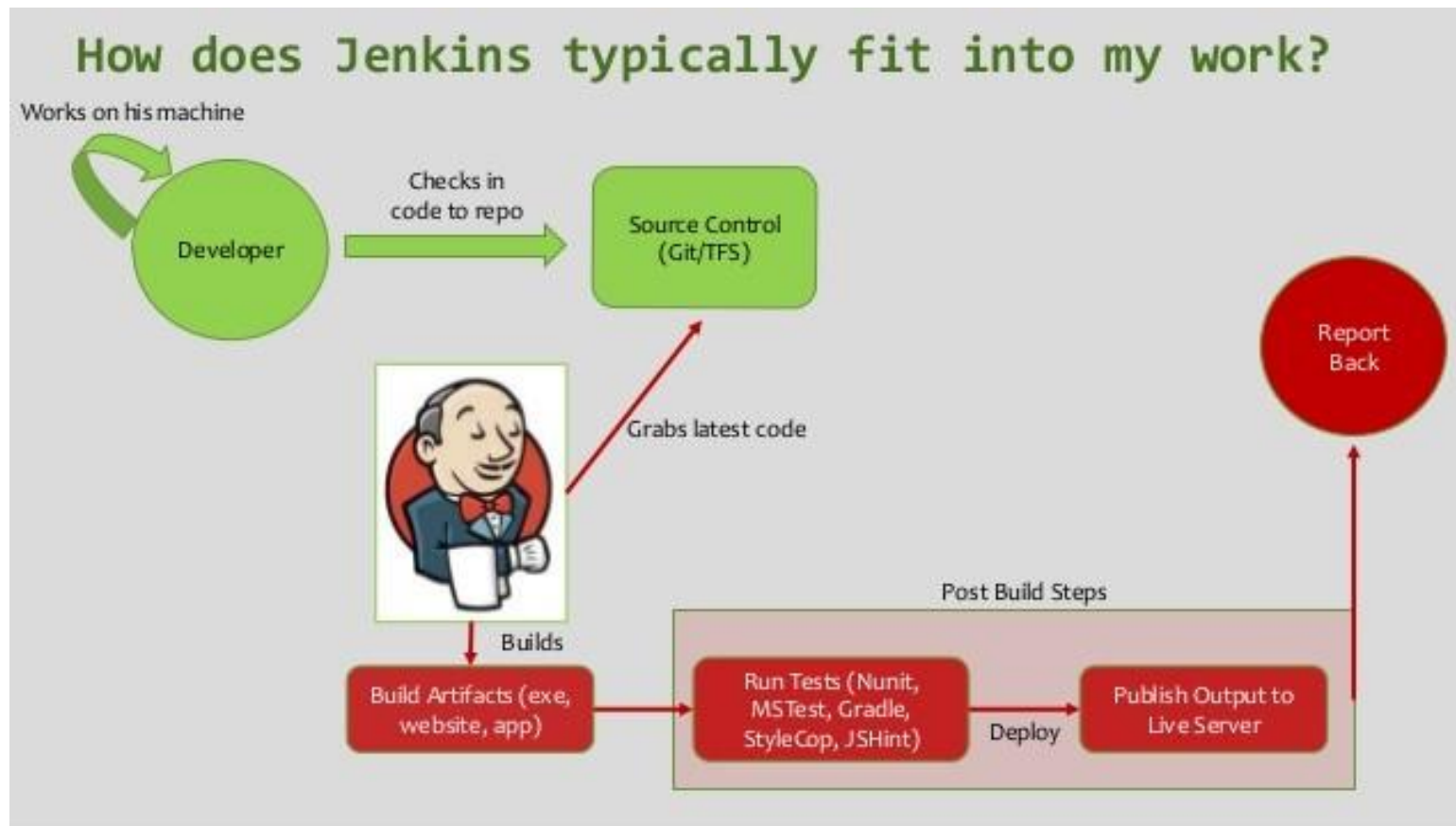
Jenkins is an open source automation server written in Java. Jenkins helps to automate the non-human part of software development process, with continuous integration and facilitates technical aspects of continuous delivery.

It is a server-based system that runs in servlet containers such as Apache Tomcat. It supports version control tools, including AccuRev, CVS, Subversion, Git, Mercurial, Perforce, ClearCase and RTC, and can execute Apache Ant, Apache Maven and sbt based projects as well as **arbitrary shell scripts and Windows batch commands**.

Jenkins: Features



Jenkins: Introduction - Introduction about Jenkins



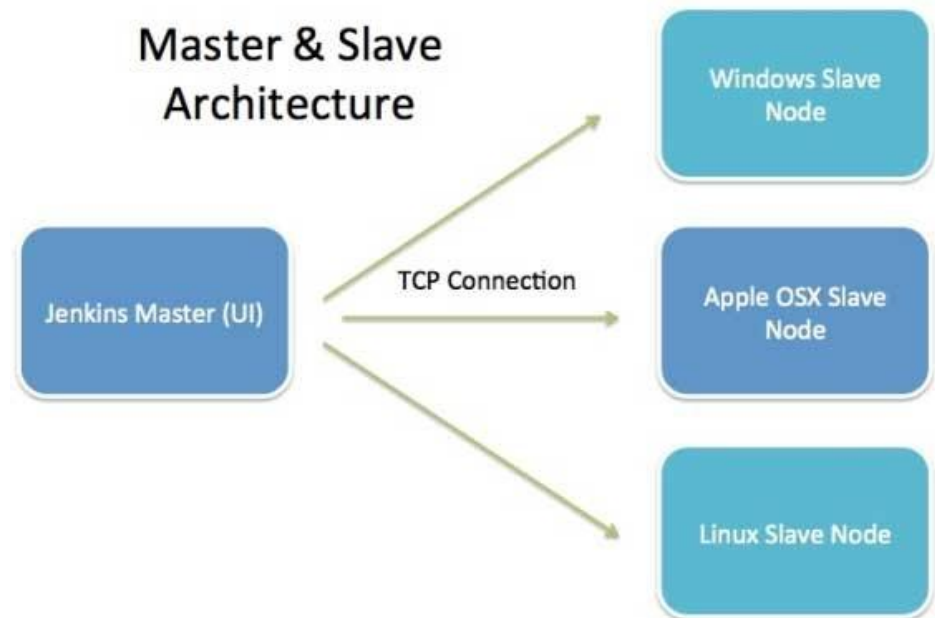
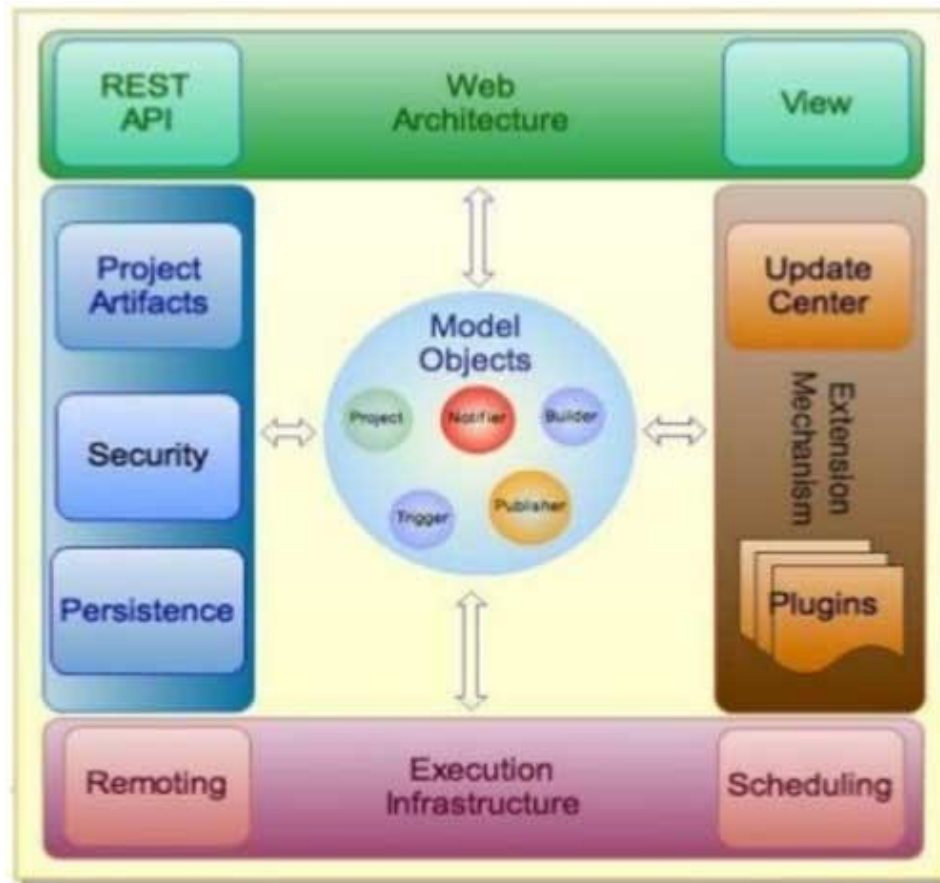
Jenkins: Introduction - Build Lifecycle

A build lifecycle is the process of building and distributing a particular artifact (project).

Build Lifecycle phases:

- **validate** - validates the project
- **compile** - compiles the source code of the project
- **test** - tests the compiled source code using a suitable unit testing framework.
- **package** - take the compiled code and package it in a distributable format
- **verify** – run integration tests
- **install** - install the package into the local repository (may be for dependencies)
- **deploy** - copies the final package to the remote repository for sharing with others

Jenkins: Introduction - Jenkins Architecture



Jenkins: Introduction - Jenkins Architecture

Master:

- Schedule Build Job
- Dispatches Builds to the Slave for Actual job Execution
- Monitoring the Slave and recording the build Results

Slave :

- Execute Builds jobs dispatched by master

Jenkins: Installation - Obtaining and installing Jenkins



Url: <https://jenkins.io/doc/book/getting-started/installing/>

Debian/Ubuntu

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

MacOS [installer package also available]
brew install jenkins

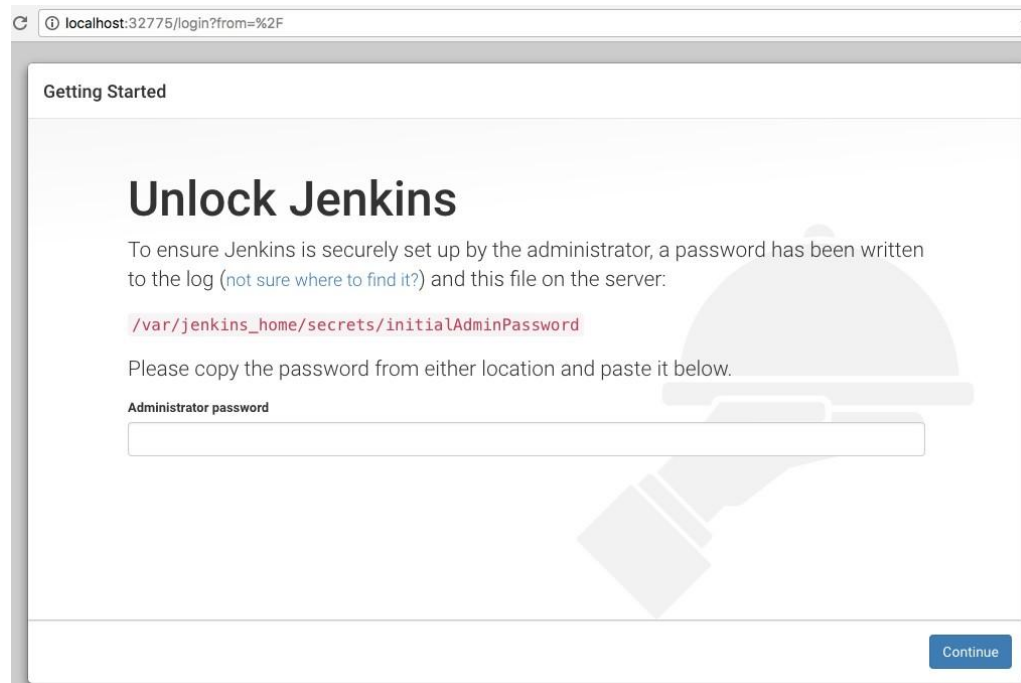
Docker

`docker pull jenkins/jenkins`

Windows

Using installer package

Jenkins: Installing



localhost:32775/login?from=%2F

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:

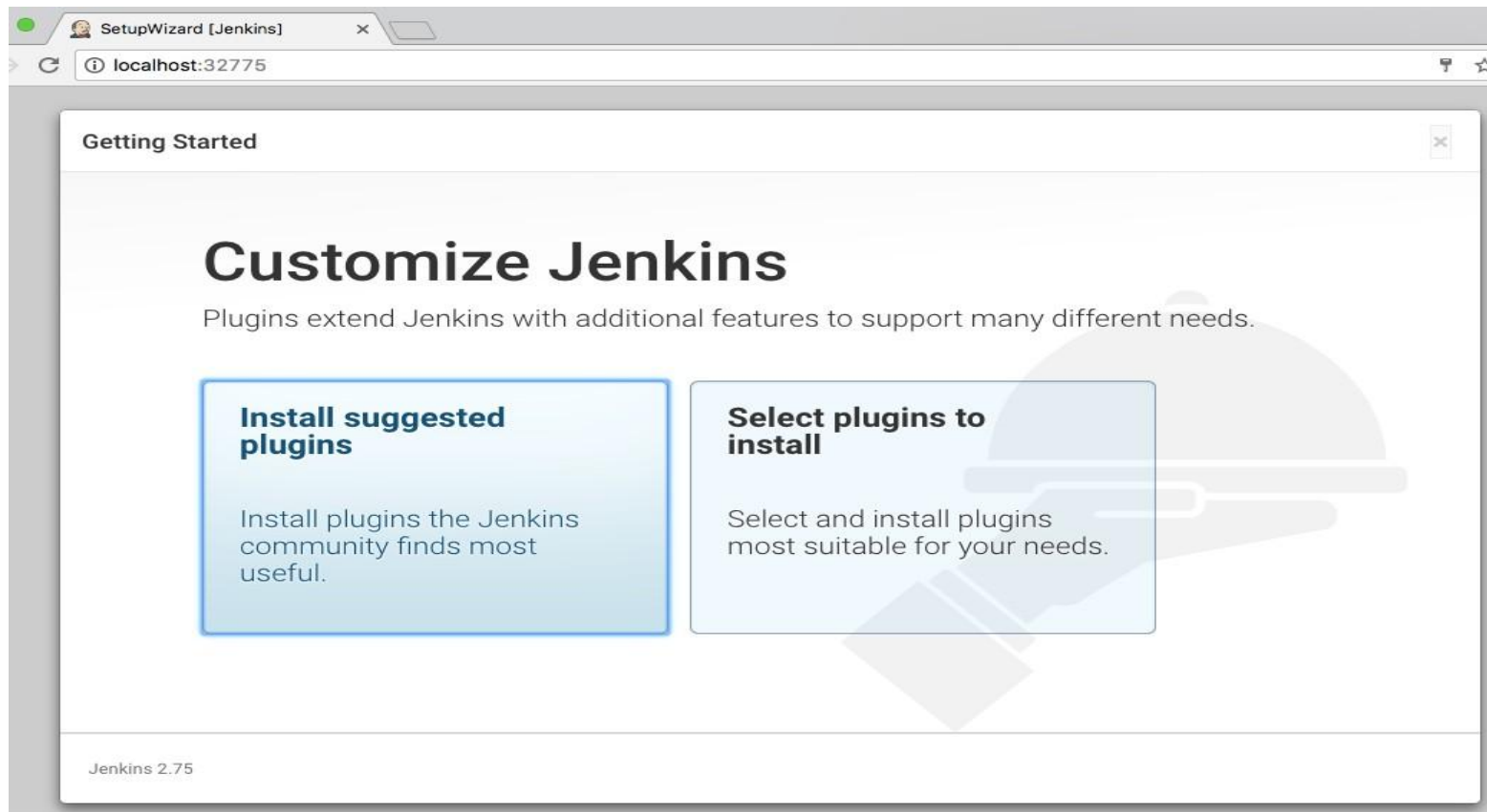
```
/var/jenkins_home/secrets/initialAdminPassword
```

Please copy the password from either location and paste it below.

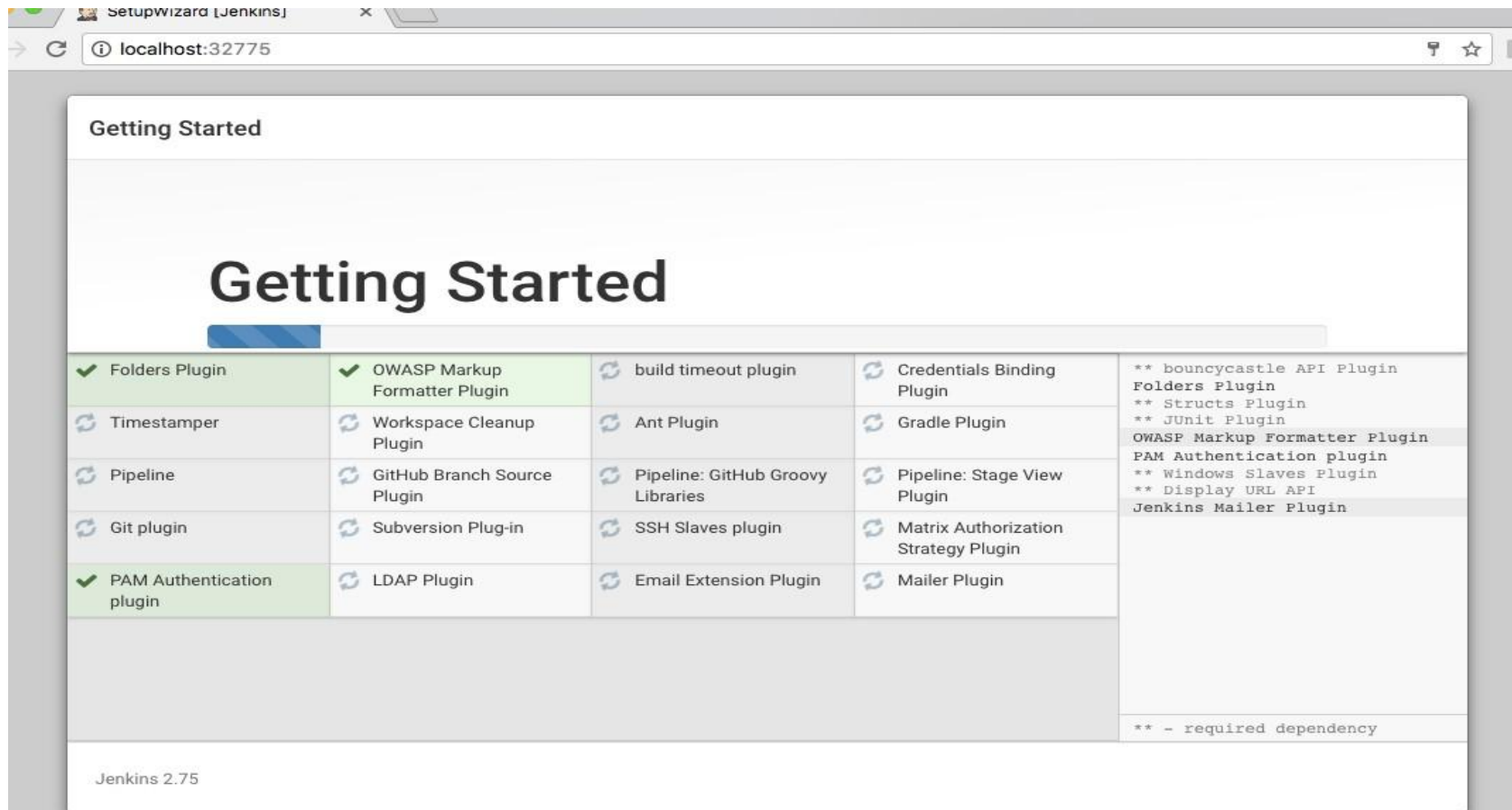
Administrator password

Continue

Jenkins: Installation - Docker + Jenkins: Browsing



Jenkins: Installation – Install Suggested Plugins



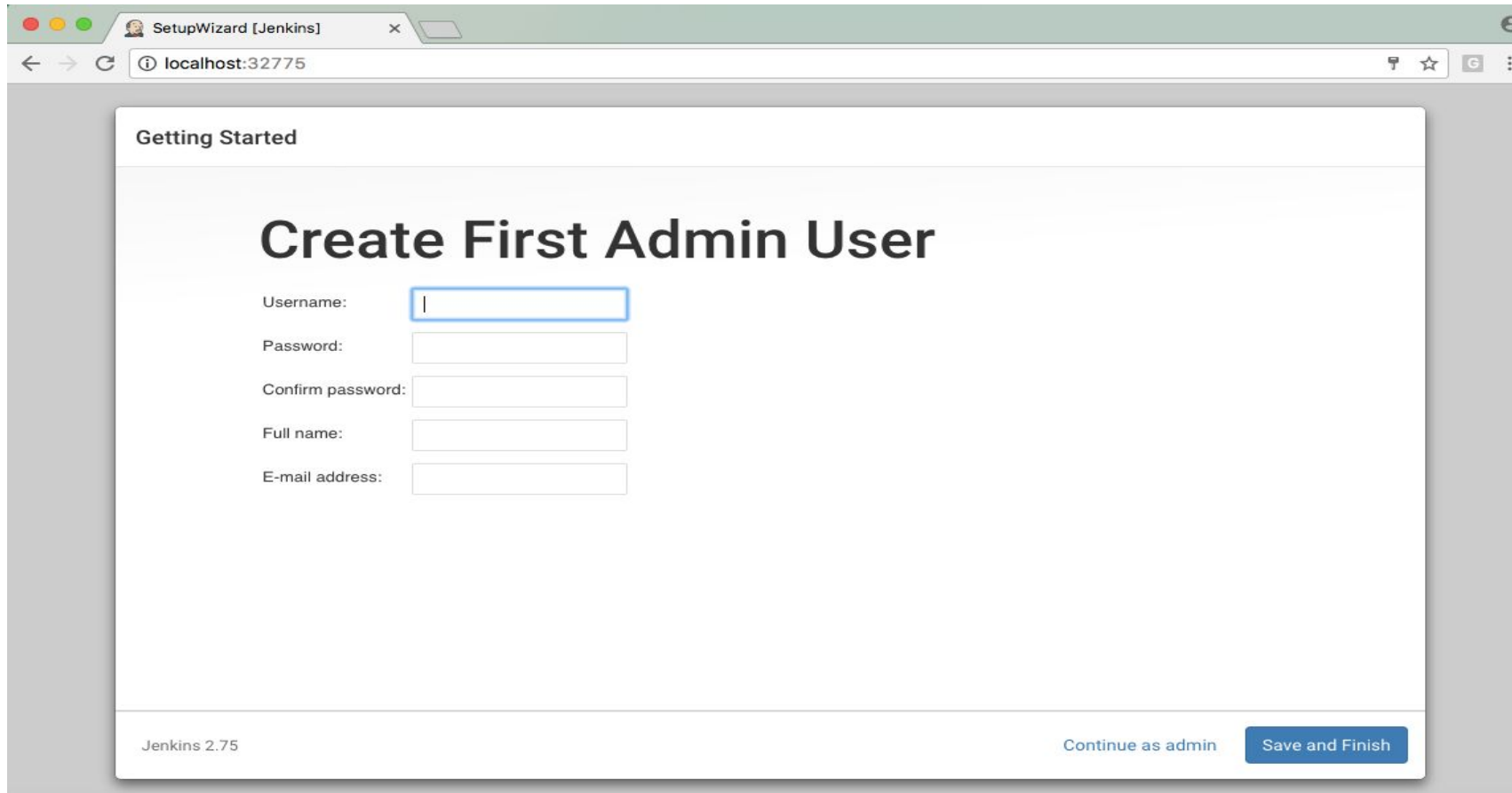
The screenshot shows the Jenkins Setup Wizard interface in a web browser. The page title is "Getting Started". Below the title, there is a progress bar. The main content area displays a table of suggested plugins. The table has four columns. The first column lists plugins with a green checkmark icon, indicating they are already installed. The second column lists plugins with a green checkmark icon and the text "OWASP Markup Formatter Plugin". The third column lists plugins with a blue circular arrow icon. The fourth column lists plugins with a blue circular arrow icon. To the right of the table, there is a list of required dependencies, each preceded by two asterisks (**).

✓ Folders Plugin	✓ OWASP Markup Formatter Plugin	↻ build timeout plugin	↻ Credentials Binding Plugin	** bouncycastle API Plugin
↻ Timestampers	↻ Workspace Cleanup Plugin	↻ Ant Plugin	↻ Gradle Plugin	Folders Plugin
↻ Pipeline	↻ GitHub Branch Source Plugin	↻ Pipeline: GitHub Groovy Libraries	↻ Pipeline: Stage View Plugin	** Structs Plugin
↻ Git plugin	↻ Subversion Plug-in	↻ SSH Slaves plugin	↻ Matrix Authorization Strategy Plugin	** JUnit Plugin
✓ PAM Authentication plugin	↻ LDAP Plugin	↻ Email Extension Plugin	↻ Mailer Plugin	OWASP Markup Formatter Plugin
				PAM Authentication plugin
				** Windows Slaves Plugin
				** Display URL API
				Jenkins Mailer Plugin

** - required dependency

Jenkins 2.75

Jenkins: Installation – Create First Admin User



The screenshot shows a web browser window with the title 'SetupWizard [Jenkins]'. The address bar shows 'localhost:32775'. The page content is titled 'Getting Started' and 'Create First Admin User'. It contains five input fields: 'Username:', 'Password:', 'Confirm password:', 'Full name:', and 'E-mail address:'. The 'Username:' field is currently active with a cursor. At the bottom, there is a footer with 'Jenkins 2.75' on the left and two buttons on the right: 'Continue as admin' and 'Save and Finish'.

Getting Started

Create First Admin User

Username:

Password:

Confirm password:

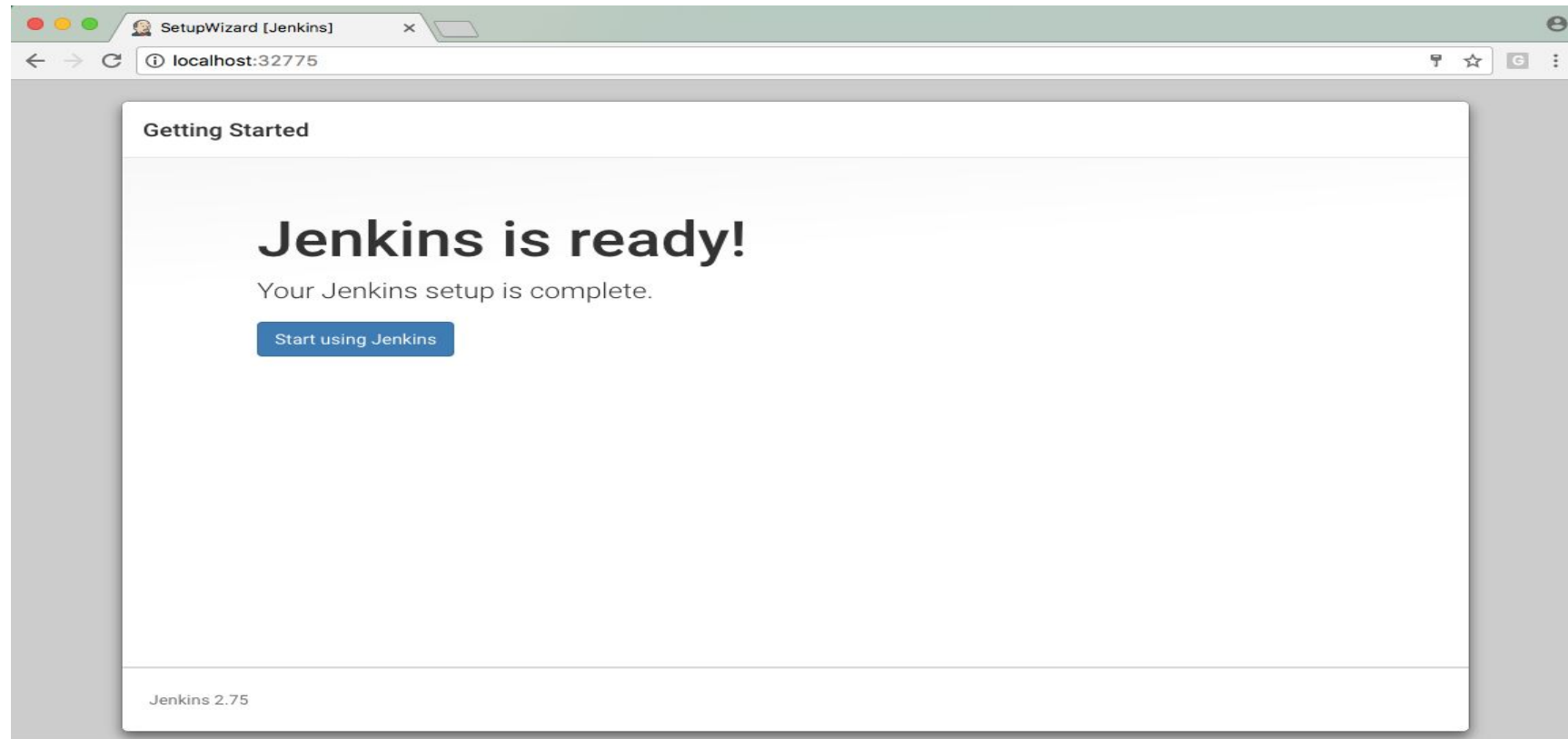
Full name:

E-mail address:

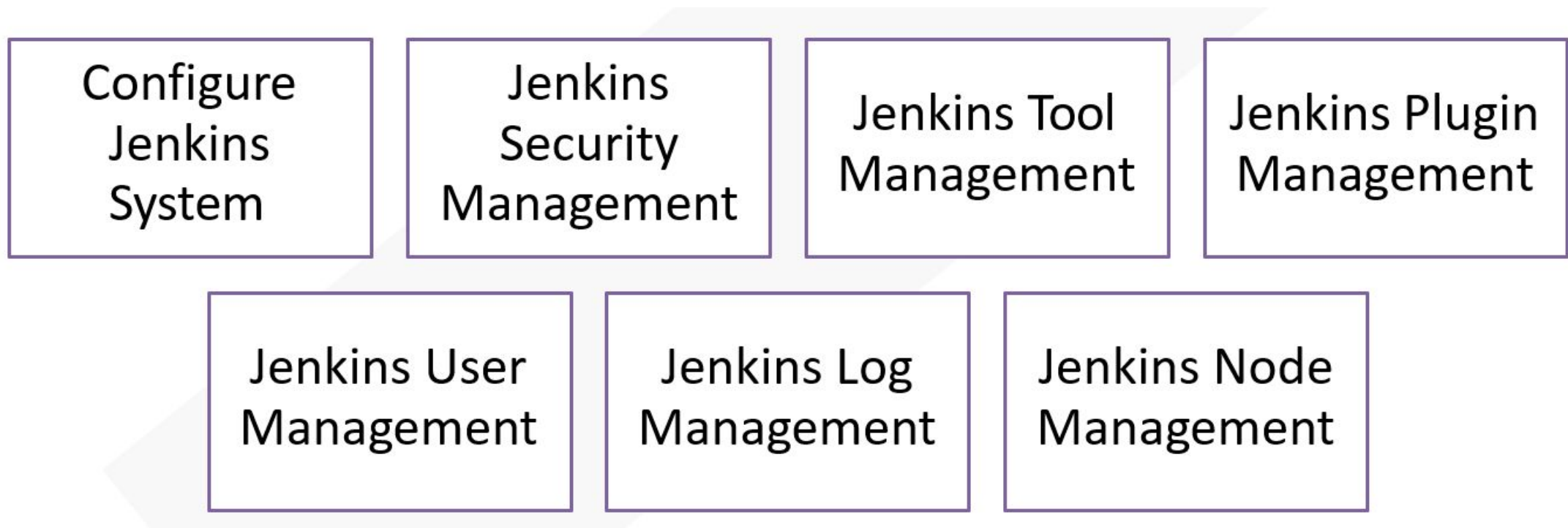
Jenkins 2.75

[Continue as admin](#) [Save and Finish](#)

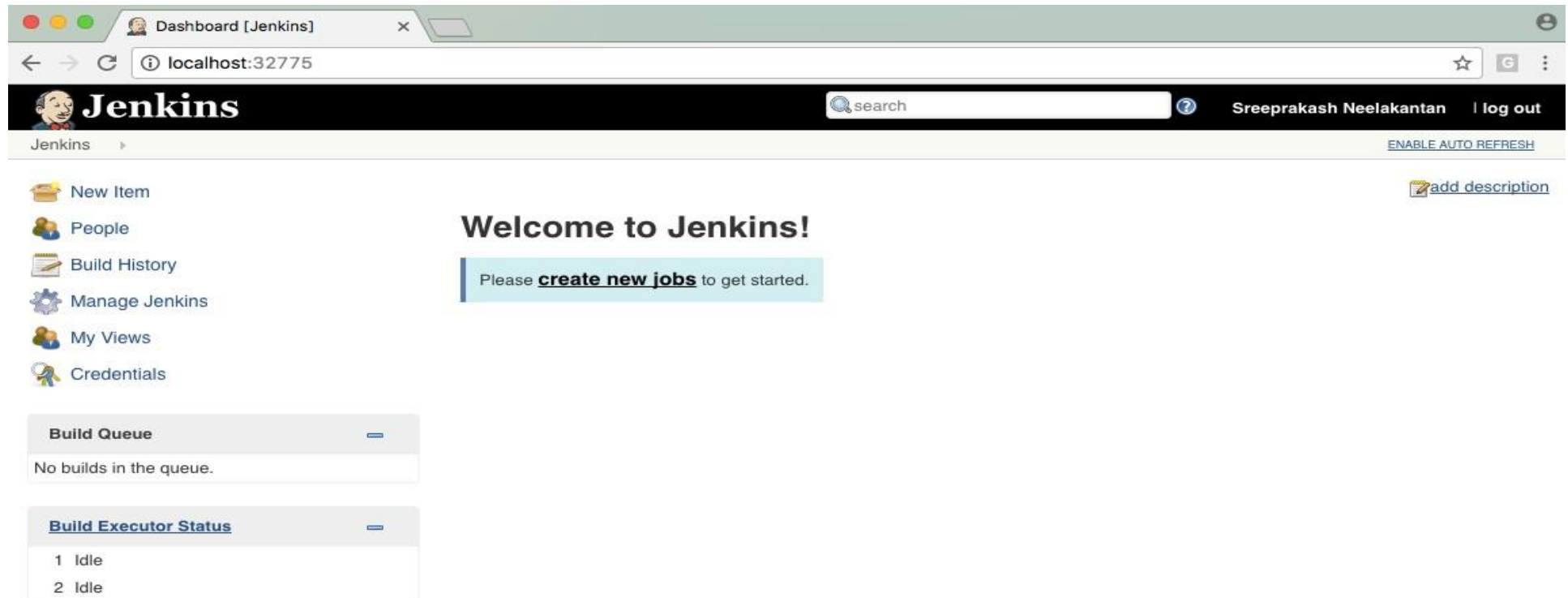
Jenkins: Installation – Ready to View Jenkins Dashboard



Jenkins: Management



Jenkins: Installation – Ready to View Jenkins Dashboard



The screenshot shows the Jenkins web interface in a browser window. The browser's address bar displays 'localhost:32775'. The Jenkins header includes the logo, a search bar, the user name 'Sreeprakash Neelakantan', and a 'log out' link. A sidebar on the left contains links to 'New Item', 'People', 'Build History', 'Manage Jenkins', 'My Views', and 'Credentials'. The main content area features a 'Welcome to Jenkins!' message with a prompt to 'create new jobs'. Below this, there are two expandable sections: 'Build Queue' (showing 'No builds in the queue.') and 'Build Executor Status' (showing two idle executors).

Dashboard [Jenkins] x

localhost:32775

Jenkins

search

Sreeprakash Neelakantan | log out

ENABLE AUTO REFRESH

[add description](#)

Welcome to Jenkins!

Please **create new jobs** to get started.


Build Queue

No builds in the queue.

Build Executor Status

- 1 Idle
- 2 Idle

Jenkins: Securing Jenkins


 **Jenkins**


search


Sreeprakash Neelakantan | log out


Jenkins


ENABLE AUTO REFRESH


 New Item

 People

 Build History

 **Manage Jenkins**

 My Views

 Credentials


Build Queue


No builds in the queue.


Build Executor Status


1	Idle
2	Idle


Manage Jenkins


 New version of Jenkins (2.76) is available for [download](#) ([changelog](#)).


 [Configure System](#)
Configure global settings and paths.


 [Configure Global Security](#)
Secure Jenkins; define who is allowed to access/use the system.


 [Configure Credentials](#)
Configure the credential providers and types


 [Global Tool Configuration](#)
Configure tools, their locations and automatic installers.


 [Reload Configuration from Disk](#)
Discard all the loaded data in memory and reload everything from file system. Useful when you modified config files directly on disk.

 [Manage Plugins](#)
Add, remove, disable or enable plugins that can extend the functionality of Jenkins. **(updates available)**

 [System Information](#)
Displays various environmental information to assist trouble-shooting.

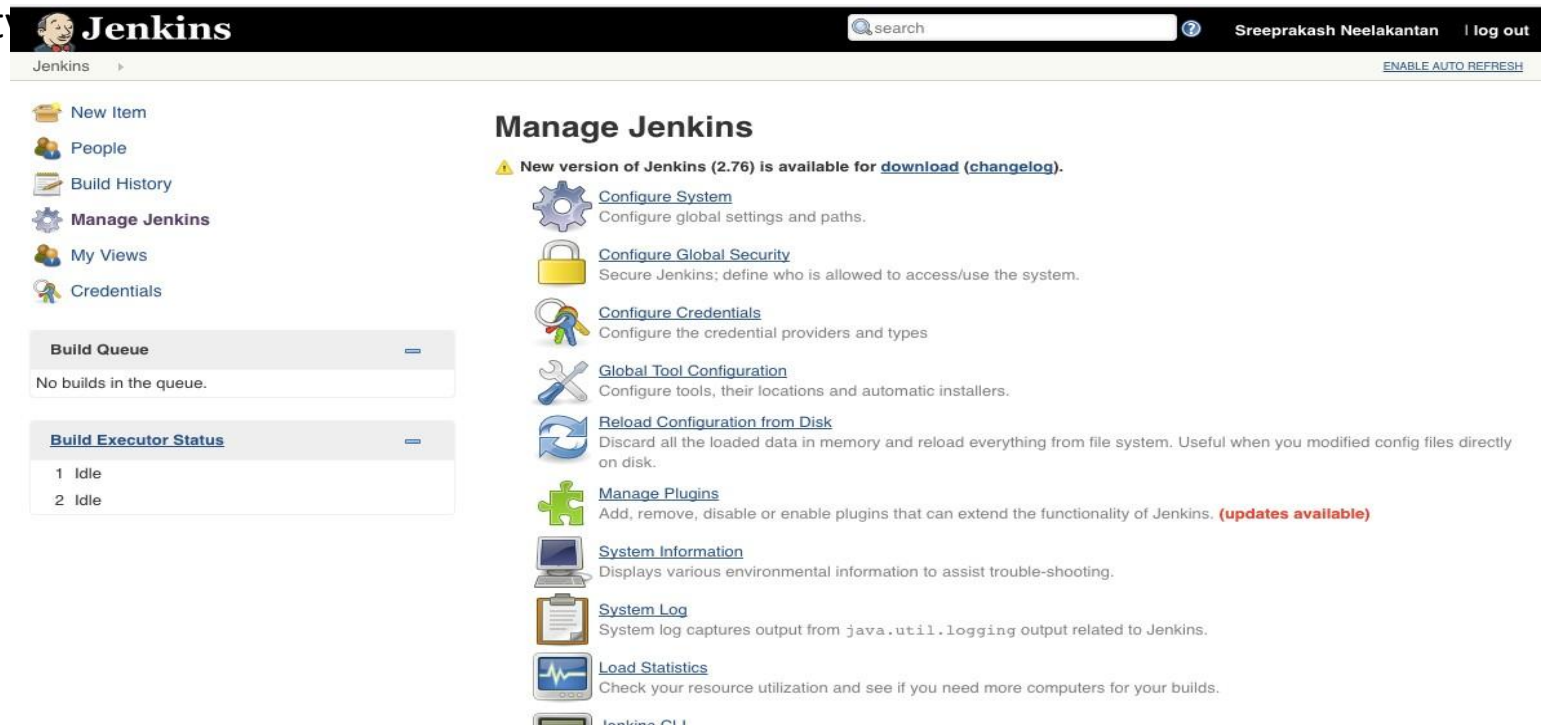
 [System Log](#)
System log captures output from `java.util.logging` output related to Jenkins.

 [Load Statistics](#)
Check your resource utilization and see if you need more computers for your builds.

 [Jenkins CLI](#)

Jenkins: Securing Jenkins - Authentication

1. Configure Global Security

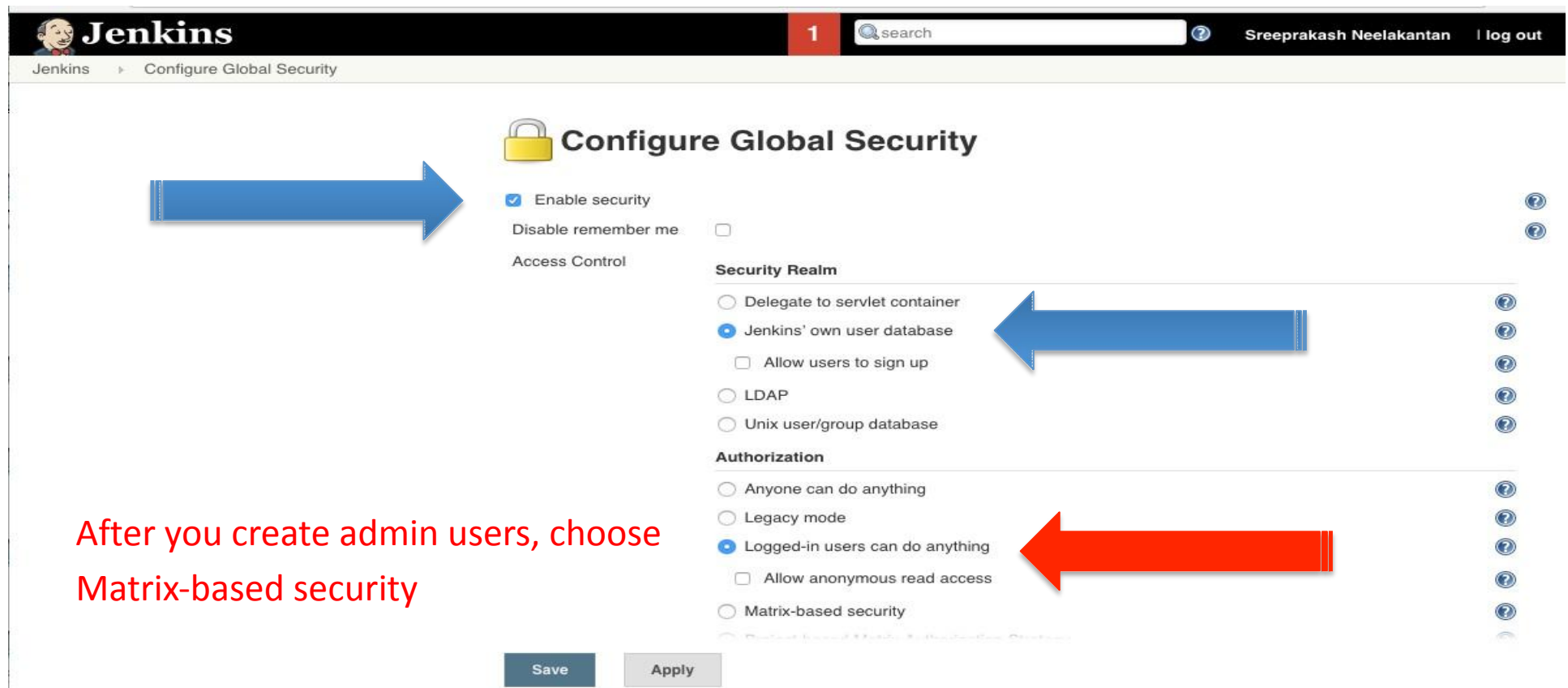


The screenshot displays the Jenkins web interface. At the top, there's a header with the Jenkins logo, a search bar, and the user name 'Sreeprakash Neelakantan' with a 'log out' link. Below the header, the left sidebar contains navigation links: 'New Item', 'People', 'Build History', 'Manage Jenkins' (highlighted), 'My Views', and 'Credentials'. The main content area is titled 'Manage Jenkins' and features a warning about a new version (2.76) being available for download. Below this warning, there's a list of configuration options, each with an icon and a description:

- Configure System**: Configure global settings and paths.
- Configure Global Security**: Secure Jenkins; define who is allowed to access/use the system.
- Configure Credentials**: Configure the credential providers and types.
- Global Tool Configuration**: Configure tools, their locations and automatic installers.
- Reload Configuration from Disk**: Discard all the loaded data in memory and reload everything from file system. Useful when you modified config files directly on disk.
- Manage Plugins**: Add, remove, disable or enable plugins that can extend the functionality of Jenkins. (updates available)
- System Information**: Displays various environmental information to assist trouble-shooting.
- System Log**: System log captures output from `java.util.logging` output related to Jenkins.
- Load Statistics**: Check your resource utilization and see if you need more computers for your builds.

On the left side of the main content area, there are two expandable sections: 'Build Queue' (showing 'No builds in the queue.') and 'Build Executor Status' (showing two 'Idle' executors).

Jenkins: Securing Jenkins - Authentication



The screenshot shows the Jenkins 'Configure Global Security' page. A blue arrow points to the 'Enable security' checkbox, which is checked. Another blue arrow points to the 'Jenkins' own user database' radio button under the 'Security Realm' section. A red arrow points to the 'Logged-in users can do anything' radio button under the 'Authorization' section. The page includes a 'Save' button and an 'Apply' button.

Jenkins 1 search Sreeprakash Neelakantan | log out

Jenkins » Configure Global Security

Configure Global Security

☒ Enable security
Disable remember me ☐
Access Control

Security Realm

- ☐ Delegate to servlet container
- ☒ Jenkins' own user database
- ☐ Allow users to sign up
- ☐ LDAP
- ☐ Unix user/group database

Authorization

- ☐ Anyone can do anything
- ☐ Legacy mode
- ☒ Logged-in users can do anything
- ☐ Allow anonymous read access
- ☐ Matrix-based security

Save Apply

After you create admin users, choose
Matrix-based security

Jenkins: Securing Jenkins - Creating users

1. Login as Admin -> Manage Jenkins -> Manage Users -> Create User



Jenkins 1 search

Jenkins > Jenkins' own user database

[Back to Dashboard](#)

[Manage Jenkins](#)

[Create User](#)

Create User

Username: user1

Password:

Confirm password:

Full name: User1

E-mail address: user1@localhost

Create User

Jenkins: Securing Jenkins - Creating users

1. Login as Admin -> Manage Jenkins -> Configure Global Security -> Matrix-based
2. Allow only essential features for the users. Remember to check **admin** for at least one.



Configure Global Security

☒ Enable security

☐ Disable remember me

Access Control

☐

Security Realm

- ☐ Delegate to servlet container
- ☒ Jenkins' own user database
 - ☐ Allow users to sign up
- ☐ LDAP
- ☐ Unix user/group database

Authorization

- ☐ Anyone can do anything
- ☐ Legacy mode
- ☐ Logged-in users can do anything
- ☒ Matrix-based security

Be careful when allowing users to delete jobs

[illegible]

Jenkins: Securing Jenkins - Creating users

1. Logout and Login as User1



User:

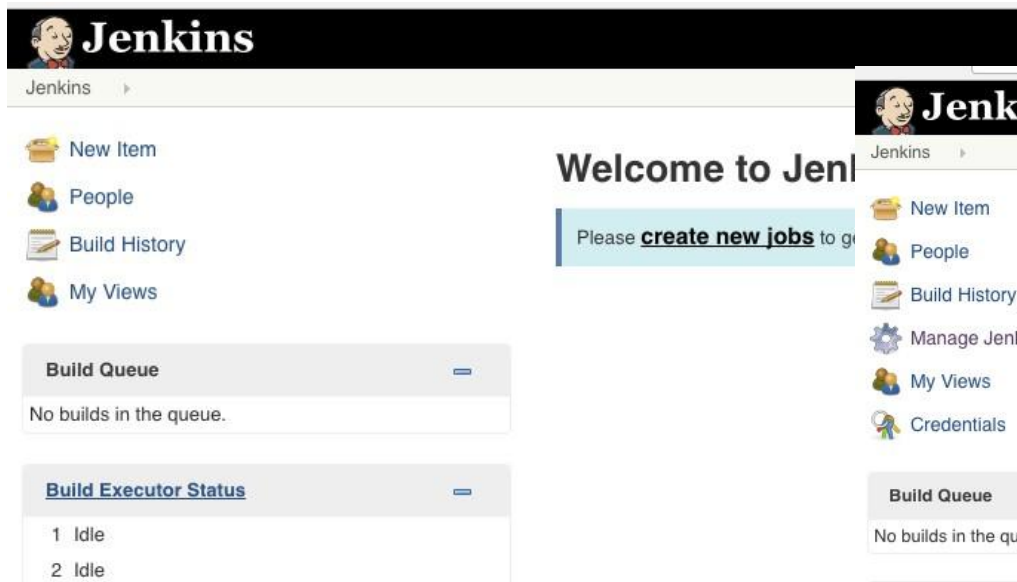
Password:

☐ Remember me on this computer

Jenkins: Securing Jenkins - Creating users

1. User1 logged in (see the difference between the admin user and user1)

User1



The screenshot shows the Jenkins web interface for a user named 'User1'. The top navigation bar is black with the Jenkins logo and name. Below it, a light green bar contains the text 'Jenkins' and a right-pointing arrow. The main content area is divided into a left sidebar and a central panel. The sidebar contains links for 'New Item', 'People', 'Build History', and 'My Views'. The central panel has a 'Welcome to Jenkins!' message, a blue box with the text 'Please [create new jobs](#) to get started.', and two expandable sections: 'Build Queue' (showing 'No builds in the queue.') and 'Build Executor Status' (showing two idle executors).

Jenkins

Jenkins

New Item

People

Build History

My Views

Build Queue

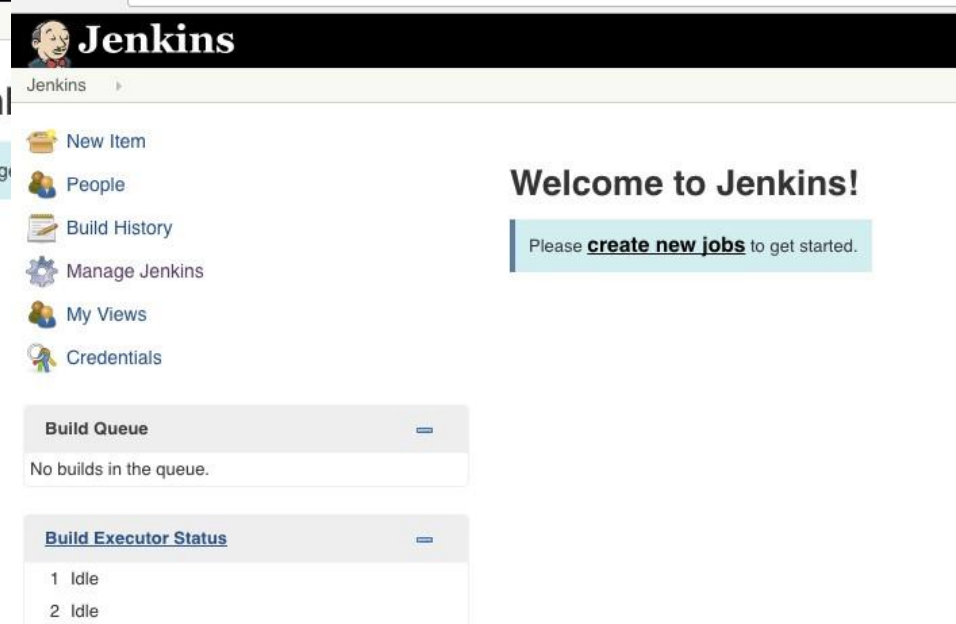
No builds in the queue.

Build Executor Status

1 Idle

2 Idle

Admin



The screenshot shows the Jenkins web interface for an administrator. The top navigation bar is black with the Jenkins logo and name. Below it, a light green bar contains the text 'Jenkins' and a right-pointing arrow. The main content area is divided into a left sidebar and a central panel. The sidebar contains links for 'New Item', 'People', 'Build History', 'Manage Jenkins', 'My Views', and 'Credentials'. The central panel has a 'Welcome to Jenkins!' message, a blue box with the text 'Please [create new jobs](#) to get started.', and two expandable sections: 'Build Queue' (showing 'No builds in the queue.') and 'Build Executor Status' (showing two idle executors).

Jenkins

Jenkins

New Item

People

Build History

Manage Jenkins

My Views

Credentials

Build Queue

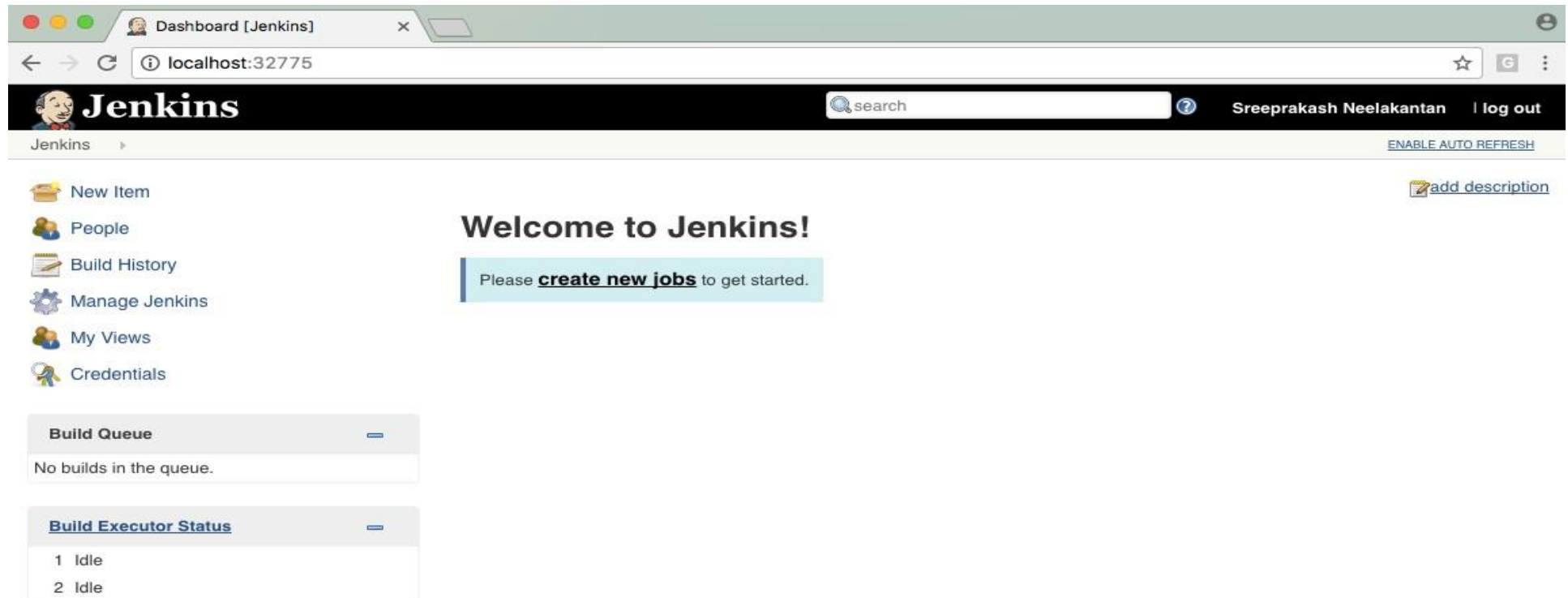
No builds in the queue.

Build Executor Status

1 Idle

2 Idle

Jenkins: Installation - Exploring Jenkins Dashboard



The screenshot shows the Jenkins web interface in a browser. The browser's address bar displays 'localhost:32775'. The Jenkins header includes the logo, a search bar, the user name 'Sreeprakash Neelakantan', and a 'log out' link. A sidebar on the left contains links to 'New Item', 'People', 'Build History', 'Manage Jenkins', 'My Views', and 'Credentials'. The main content area features a 'Welcome to Jenkins!' message with a prompt to 'create new jobs'. Below this, there are two expandable sections: 'Build Queue' (showing 'No builds in the queue.') and 'Build Executor Status' (showing two idle executors).

Dashboard [Jenkins] x

localhost:32775

Jenkins

search

Sreeprakash Neelakantan | log out

ENABLE AUTO REFRESH

[add description](#)

New Item

People

Build History

Manage Jenkins

My Views

Credentials

Build Queue


No builds in the queue.

Build Executor Status

1 Idle

2 Idle

Jenkins: Jobs - Creating Jobs

 **Jenkins**

[?](#) **Sreeprakash Neelakantan** | [log out](#)

Jenkins [ENABLE AUTO REFRESH](#)

 [New Item](#)

 [People](#)

 [Build History](#)

 [Manage Jenkins](#)

 [My Views](#)

 [Credentials](#)

 [add description](#)

Welcome to Jenkins!

Please **create new jobs** to get started.


Build Queue 

No builds in the queue.

Build Executor Status 

1	Idle
2	Idle


Jenkins: Jobs - Creating Jobs


 **Jenkins** [Sreepakash Neelakantan](#) | [log out](#)


Jenkins ▶


Enter an item name


» Required field


**Freestyle project**
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 slaves. Suitable for building pipelines (formerly known as workflows) and/or organizing complex activities that do not easily fit in free-style job type.

**External Job**
This type of job allows you to record the execution of a process run outside Jenkins, even on a remote machine. This is designed so that you can use Jenkins as a dashboard of your existing automation system.

**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.

**GitHub Organization**
Scans a GitHub organization (or user account) for all repositories matching some defined markers.

**Multibranch Pipeline**
Scans a set of Pipeline scripts according to detected branches from SCM repository.

Jenkins: Jobs - Creating Jobs

Jenkins » Sree1 »

Click to go back, hold to see history

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

Project name

Description

[Plain text] [Preview](#)

☐ Discard old builds ?

☐ GitHub project ?

☐ This project is parameterized ?

☐ Throttle builds ?

☐ Disable this project ?

☐ Execute concurrent builds if necessary ?

[Advanced...](#)

Source Code Management

☒ None

☐ Git ?

☐ Subversion ?

Build Triggers

Jenkins: Jobs - Creating Jobs

Jenkins > Sree1 >

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

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

Build Environment

☐ Delete workspace before build starts
☐ Abort the build if it's stuck
☐ Add timestamps to the Console Output
☐ Use secret text(s) or file(s) ?
☐ With Ant ?

Build

Add build step ▾

- Execute Windows batch command
- Execute shell**
- Invoke Ant
- Invoke Gradle script
- Invoke top-level Maven targets
- Run with timeout
- Set build status to "pending" on GitHub commit

Save Apply

Jenkins: Jobs - Creating Jobs

The screenshot shows the Jenkins job configuration interface for a job named 'Sree1'. The breadcrumb navigation at the top indicates the path 'Jenkins > Sree1'. The configuration is organized into tabs: 'General', 'Source Code Management', 'Build Triggers', 'Build Environment', 'Build', and 'Post-build Actions'. The 'Build Triggers' tab is currently selected, showing options for 'GitHub hook trigger for GITScm polling' and 'Poll SCM', both of which are unchecked. Below this is the 'Build Environment' section, which includes options for 'Delete workspace before build starts', 'Abort the build if it's stuck', 'Add timestamps to the Console Output', 'Use secret text(s) or file(s)', and 'With Ant', all of which are also unchecked. The 'Build' section is the active area, featuring an 'Execute shell' step. The command entered in the text area is `echo "Test 123" > /var/jenkins_home/sree.txt`. A link to 'See the list of available environment variables' is provided below the command field. At the bottom of the configuration area, there is an 'Add build step' button and two buttons, 'Save' and 'Apply', for saving the configuration.

Jenkins > Sree1

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

☐ GitHub hook trigger for GITScm polling

☐ Poll SCM

Build Environment

☐ Delete workspace before build starts

☐ Abort the build if it's stuck

☐ Add timestamps to the Console Output

☐ Use secret text(s) or file(s)

☐ With Ant

Build

Execute shell

Command `echo "Test 123" > /var/jenkins_home/sree.txt`


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

Advanced...

Add build step

Save Apply


Jenkins: Jobs - Running the Jobs – Jobs Details


 **Jenkins**


search


Sreeprakash Neelakantan | log out


Jenkins > Sree1 > [ENABLE AUTO REFRESH](#)


 [Back to Dashboard](#)


 **Status**



 [Changes](#)


 [Workspace](#)



 [Build Now](#)


 [Delete Project](#)


 [Configure](#)

 **Build History** [trend](#) 

find 


 [RSS for all](#)  [RSS for failures](#)

 [Workspace](#)

 [Recent Changes](#)

Project Sree1


Freestyle job

 [edit description](#)

[Disable Project](#)

Permalinks

Jenkins: Jobs - Running the Jobs – Build Details


 **Jenkins**


search


Sreeprakash Neelakantan | log out


Jenkins > Sree1 > #1


[ENABLE AUTO REFRESH](#)


 [Back to Project](#)


 **Status**

 [Changes](#)


 [Console Output](#)


 [Edit Build Information](#)


 [Delete Build](#)

 **Build #1 (Sep 9, 2017 2:08:38 AM)**

Started 10 sec ago
Took [0.13 sec](#)

 [add description](#)

 No changes.

 Started by user [Sreeprakash Neelakantan](#)

Jenkins: Jobs - Disabling and Enabling jobs

search


?


Sreeprakash Neelakantan | log out

ENABLE AUTO REFRESH

Project Sree1

Freestyle job

[Workspace](#)


[Recent Changes](#)


trend

for failures

Permalinks

- [Last build \(#1\), 7 min 32 sec ago](#)
- [Last stable build \(#1\), 7 min 32 sec ago](#)
- [Last successful build \(#1\), 7 min 32 sec ago](#)
- [Last completed build \(#1\), 7 min 32 sec ago](#)



 [edit description](#)

Disable Project

Project Sree1

Freestyle job



[Workspace](#)



[Recent Changes](#)

trend

x

for failures

Permalinks

- [Last build \(#1\), 7 min 32 sec ago](#)
- [Last stable build \(#1\), 7 min 32 sec ago](#)
- [Last successful build \(#1\), 7 min 32 sec ago](#)
- [Last completed build \(#1\), 7 min 32 sec ago](#)

 [edit description](#)

Disable Project

Jenkins: Jobs - Disabling and Enabling jobs

The screenshot shows the Jenkins web interface for a project named 'Sree1'. The top navigation bar includes the Jenkins logo and a search bar. The sidebar on the left contains links for 'Back to Dashboard', 'Status', 'Changes', 'Workspace', 'Delete Project', and 'Configure'. The main content area displays 'Project Sree1' as a 'Freestyle job'. A yellow warning icon indicates 'This project is currently disabled', with an 'Enable' button next to it. Below this, there are links for 'Workspace' and 'Recent Changes'. The 'Build History' section shows a single build (#1) from Sep 9, 2017 2:08 AM, with RSS feeds for all builds and failures. A large blue arrow points to the 'Enable' button.

Jenkins search

Jenkins > Sree1 >

Back to Dashboard

Status

Changes

Workspace

Delete Project

Configure

Build History [trend](#)

find

#1 Sep 9, 2017 2:08 AM

[RSS for all](#) [RSS for failures](#)

Project Sree1

Freestyle job

This project is currently disabled **Enable**


[Workspace](#)

[Recent Changes](#)


Permalinks


- [Last build \(#1\), 7 min 46 sec ago](#)
- [Last stable build \(#1\), 7 min 46 sec ago](#)
- [Last successful build \(#1\), 7 min 46 sec ago](#)
- [Last completed build \(#1\), 7 min 46 sec ago](#)


Jenkins: Jobs - Deleting jobs


**Jenkins**


Jenkins > Sree1 >


 [Back to Dashboard](#)



 [Status](#)


 [Changes](#)


 [Workspace](#)



 [Delete Project](#)

 [Configure](#)

 **Build History** [trend](#) 




 **#1** Sep 9, 2017 2:08 AM


 [RSS for all](#)  [RSS for failures](#)


localhost:32769 says:
Are you sure about deleting the Project 'Sree1'?

[Cancel](#) [OK](#)

Freestyle job

 **This project is currently disabled** [Enable](#)

 [Workspace](#)

 [Recent Changes](#)

Permalinks

- [Last build \(#1\), 7 min 46 sec ago](#)
- [Last stable build \(#1\), 7 min 46 sec ago](#)
- [Last successful build \(#1\), 7 min 46 sec ago](#)
- [Last completed build \(#1\), 7 min 46 sec ago](#)

Jenkins: Schedule Build

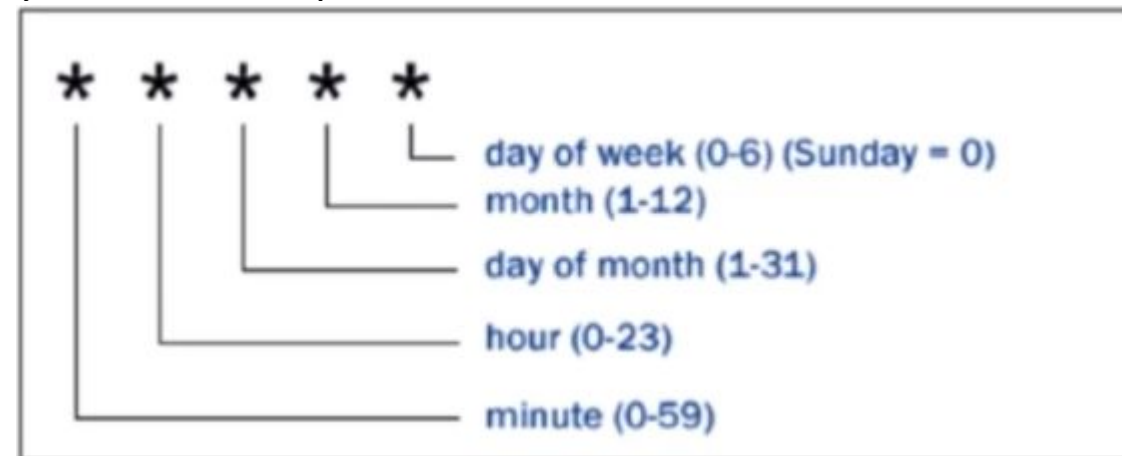
In cron, Each line consists of 5 fields separated by tab or Whitespace

To specify the Multiple Values for one field

- -> all values
- M-N A range of values
- A,B,Z Enumerates multiple values

0 0 * * * : Every data at midnight

0 2-4 * * * : 2 a.m , 3 a.m ,4 a.m Every day

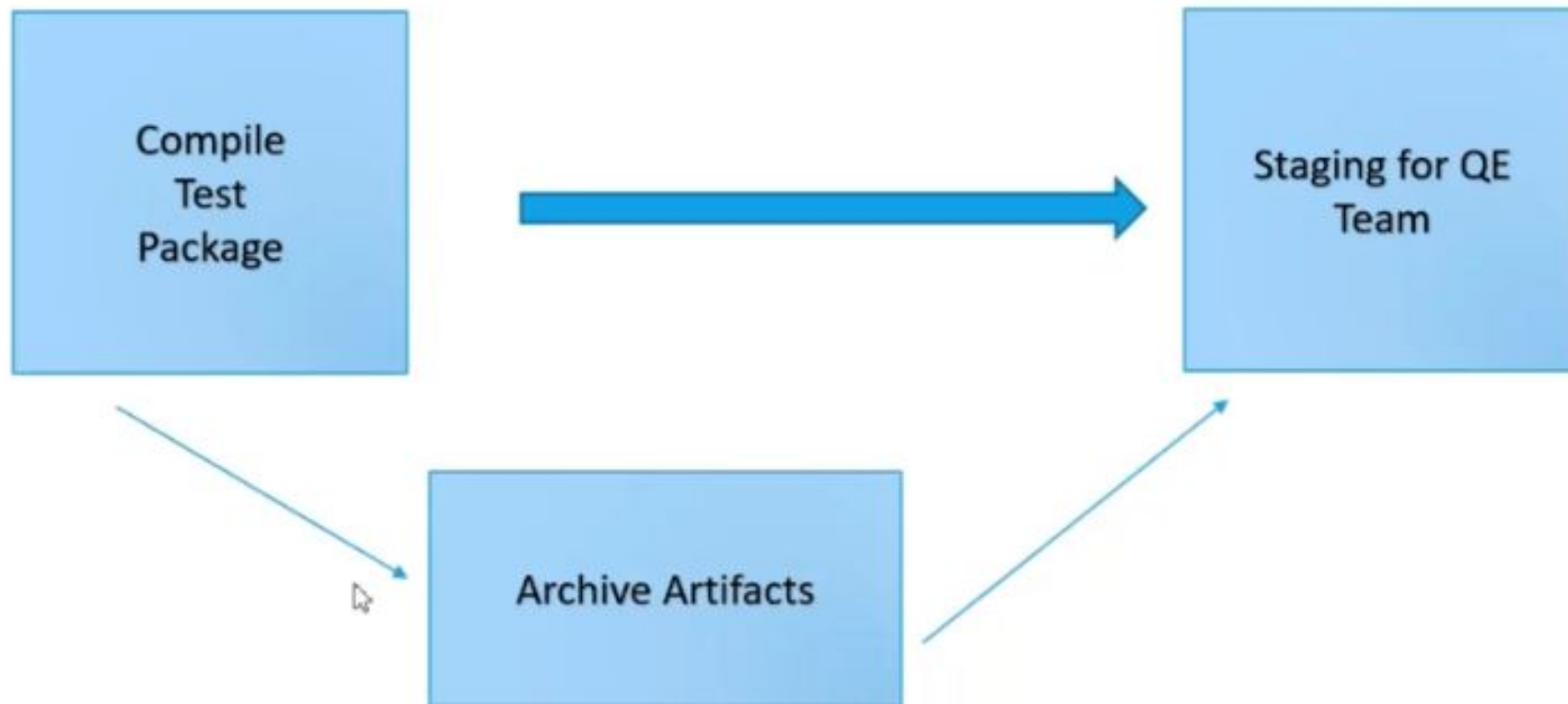


GIT HUB Hook

<https://wiki.jenkins-ci.org/display/JENKINS/GitHub+Plugin>

Jenkins: Artifacts

.



Jenkins : Tomcat

- Tomcat is an Open source HTTP webserver that will deploy and run applications inside the tomcat container
- Default port no of tomcat was :8080
- If we want to change the port for tomcat ,you can go through conf/server.xml and search for connector port and change your port no
- IF you want to run the server , you need to go bin directory and running startup from command prompt
- we need to keep WAR file inside the WebApps folder
- Once we kept WAR file inside then need to restart /start the server

Jenkins : Deploy Artifacts


- Install copy artifact and deploy to container plugin
- Deploy our application to staging environments

Jenkins: Build Pipeline



- Configure and install build pipe line
- Create own custom view dashboard

Jenkins: Jobs - Adding and updating Plugins

 **Jenkins**

[Sreeprakash Neelakantan](#) | [log out](#)

Jenkins > Plugin Manager

[Back to Dashboard](#)

[Manage Jenkins](#)

Filter:

UpdatesAvailableInstalledAdvanced

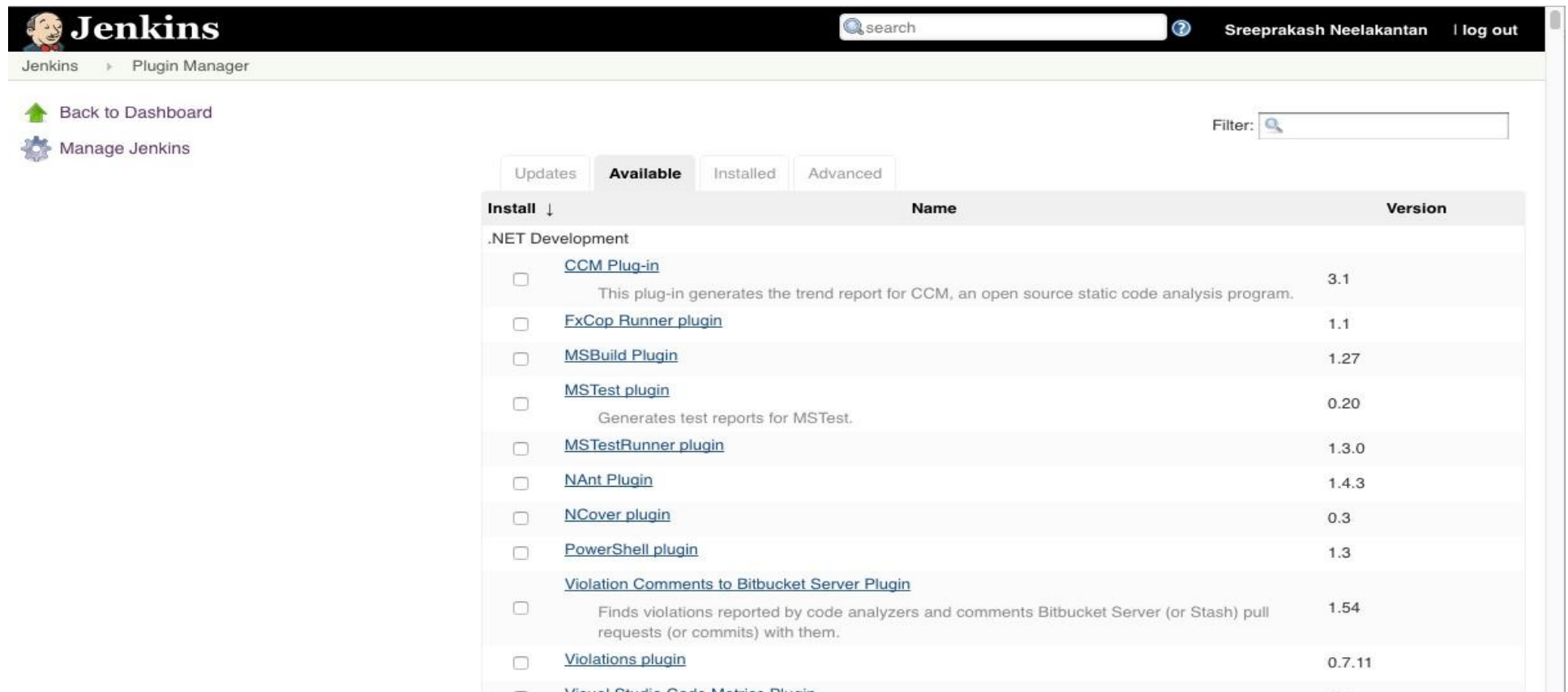
Install	Name ↓	Version	Installed
No updates			

Update information obtained: 1 hr 19 min ago [Check now](#)

Select: [All](#), [None](#)

This page lists updates to the plugins you currently use.

Jenkins: Jobs - Adding and updating Plugins



The screenshot displays the Jenkins web interface, specifically the Plugin Manager. The top navigation bar includes the Jenkins logo, a search bar, the user name 'Sreeprakash Neelakantan', and a 'log out' link. Below the navigation bar, the breadcrumb 'Jenkins > Plugin Manager' is visible. On the left sidebar, there are links for 'Back to Dashboard' and 'Manage Jenkins'. The main content area shows the 'Available' tab selected, with sub-tabs for 'Updates', 'Installed', and 'Advanced'. A filter input field is present on the right. The plugin list is organized into categories, with the '.NET Development' category expanded. It lists several plugins with checkboxes for installation, their names, descriptions, and versions.

Install ↓	Name	Version
.NET Development		
<input type="checkbox"/>	CCM Plug-in This plug-in generates the trend report for CCM, an open source static code analysis program.	3.1
<input type="checkbox"/>	FxCop Runner plugin	1.1
<input type="checkbox"/>	MSBuild Plugin	1.27
<input type="checkbox"/>	MSTest plugin Generates test reports for MSTest.	0.20
<input type="checkbox"/>	MSTestRunner plugin	1.3.0
<input type="checkbox"/>	NAnt Plugin	1.4.3
<input type="checkbox"/>	NCover plugin	0.3
<input type="checkbox"/>	PowerShell plugin	1.3
<input type="checkbox"/>	Violation Comments to Bitbucket Server Plugin Finds violations reported by code analyzers and comments Bitbucket Server (or Stash) pull requests (or commits) with them.	1.54
<input type="checkbox"/>	Violations plugin	0.7.11
<input type="checkbox"/>	Visual Studio Code Metrics Plugin	1.7

Jenkins: Jobs - Adding and updating Plugins

The screenshot shows the Jenkins Plugin Manager interface. The top navigation bar includes 'Jenkins' and 'Plugin Manager'. On the left, there are links for 'Back to Dashboard' and 'Manage Jenkins'. The main area has tabs for 'Updates', 'Available', 'Installed', and 'Advanced'. A search filter 'Docker' is applied. The 'Available' tab is active, displaying a table of plugins. A blue arrow points to the 'Docker plugin' row.

Install ↓	Name	Version
<input type="checkbox"/>	Amazon EC2 Container Service plugin Jenkins plugin to run dynamic slaves in a Amazon ECS/Docker environment	1.11
<input type="checkbox"/>	Amazon EC2 Container Service plugin with autoscaling capabilities Jenkins plugin to run dynamic slaves in a Amazon ECS/Docker environment	1.0
<input type="checkbox"/>	Yet Another Docker Plugin Yet Another Docker plugin. Provides docker Cloud provisioning	0.1.0-rc39
<input type="checkbox"/>	Aqua Security Scanner Scans Docker images for vulnerabilities	2.0
<input type="checkbox"/>	CloudBees Docker Build and Publish plugin This plugin provides the ability to build projects with a Dockerfile, and publish the resultant tagged image (repo) to a Docker registry.	1.3.2
<input type="checkbox"/>	docker-build-step	1.43
<input type="checkbox"/>	CloudShare Docker-Machine Plugin Run Docker commands on a dedicated CloudShare VM	1.1.0
<input type="checkbox"/>	Docker plugin Provide Cloud Provisioning and other Docker features	0.16.2
<input type="checkbox"/>	Kubernetes plugin Jenkins plugin to run dynamic slaves in a Kubernetes/Docker environment	1.0
<input type="checkbox"/>	CloudBees Docker Custom Build Environment Plugin	1.6.5

At the bottom, there are buttons for 'Install without restart', 'Download now and install after restart', and 'Check for updates'.

Jenkins: Jobs - Adding and updating Plugins

... Jenkins plugin to run dynamic slaves in a Kubernetes/Docker environment

<input type="checkbox"/>	Yet Another Docker Plugin Yet Another Docker plugin. Provides docker Cloud provisioning
<input type="checkbox"/>	Aqua Security Scanner Scans Docker images for vulnerabilities
<input type="checkbox"/>	CloudBees Docker Build and Publish plugin This plugin provides the ability to build projects with a Dockerfile, and publish to a Docker registry.
<input type="checkbox"/>	docker-build-step
<input type="checkbox"/>	CloudShare Docker-Machine Plugin Run Docker commands on a dedicated CloudShare VM
<input checked="" type="checkbox"/>	Docker plugin Provide Cloud Provisioning and other Docker features
<input type="checkbox"/>	Kubernetes plugin Jenkins plugin to run dynamic slaves in a Kubernetes/Docker environment
<input type="checkbox"/>	CloudBees Docker Custom Build Environment Plugin
<input type="checkbox"/>	Docker Slaves Plugin Uses Docker containers to run Jenkins build agents
	CloudBees Docker Traceability

[Install without restart](#) [Download now and install after restart](#)

11/2

Jenkins: Jobs - Adding and updating Plugins



The screenshot shows the Jenkins web interface. At the top is a black header with the Jenkins logo and a search bar. Below the header is a light green navigation bar with 'Jenkins' and 'Update Center' links. On the left side, there is a sidebar with three links: 'Back to Dashboard', 'Manage Jenkins', and 'Manage Plugins'. The main content area is titled 'Installing Plugins/Upgrades'. Under this title, there is a 'Preparation' section with a bulleted list: 'Checking internet connectivity', 'Checking update center connectivity', and 'Success'. Below this, there is a progress bar for the 'Docker plugin' which is labeled 'Installing' and shows a small progress indicator. At the bottom, there are two green arrow icons pointing right. The first points to a link 'Go back to the top page' with the text '(you can start using the installed plugins right away)' below it. The second points to a checkbox labeled 'Restart Jenkins when installation is complete and no jobs are running'.

Jenkins search

Jenkins » Update Center

- Back to Dashboard
- Manage Jenkins
- Manage Plugins

Installing Plugins/Upgrades

Preparation

- Checking internet connectivity
- Checking update center connectivity
- Success

Docker plugin  Installing

➡ [Go back to the top page](#)
(you can start using the installed plugins right away)

➡ ☐ Restart Jenkins when installation is complete and no jobs are running

Jenkins: Jobs - Adding and updating Plugins

<input checked="" type="checkbox"/>	Secret Commons Plugin Provides the common shared functionality for various Docker-related plugins.	1.8	Uninstall
<input checked="" type="checkbox"/>	Docker Pipeline Build and use Docker containers from pipelines.	1.12	Uninstall
<input checked="" type="checkbox"/>	Docker plugin This plugin integrates Jenkins with Docker	0.16.2	Uninstall
<input checked="" type="checkbox"/>	Durable Task Plugin Library offering an extension point for processes which can run outside of Jenkins yet be monitored.	1.14	Uninstall
	Email Extension Plugin This plugin is a replacement for Jenkins's email publisher. It		

Jenkins: Build Deployments - Java with Tomcat

TASK #1: Build the Java Application:

docker run -rm -v \$PWD/.m2:/root/.m2 -v \$PWD/my_host_folder/SreeJavaExample:/project -w /project maven mvn clean package

TASK #2: Deploy the application to Tomcat

docker rm -f my-tcc

docker run -d \

-p 8123:8080 \

--name my-tcc \

-v \$PWD/my_host_folder/SreeJavaExample/target/SreeJavaExample.war:/usr/local/tomcat/webapps/sree-example.war \

tomcat

NOTES:

- When using docker commands in Jenkins NEVER use the -ti option.
- Initially, while testing, preferably use the full path instead of \$PWD because \$PWD points to your task's workspace which may not have your files.
- To browse the application: <http://localhost:8123/gsa-example/>

Jenkins: Build Deployments - Java with Tomcat

In case you are not a Java developer and you want to quickly create a sample Java Web Application to test the pipeline flow, use the 'scaffolding' option of Maven


```
docker run --rm -it -v $PWD/my_host_folder:/external \  
-v $PWD/.m2:/root/.m2 \  
-w /external maven mvn archetype:generate \  
-DgroupId=com.schogini.dockermvn.example \  
-DartifactId=GsaJavaExample \  
-DarchetypeArtifactId=maven-archetype-webapp \  
-DinteractiveMode=false
```

Remember, the .m2 folder that you are mapping should be a valid maven repository. If you are not sure what that is then, remove it from the command before you execute it.

Jenkins: CLI

Ensure that you have the latest version of Java installed else, you may get an error like this:


Exception in thread "main" java.lang.UnsupportedClassVersionError: hudson/cli/CLI : Unsupported major.minor version 52.0


 **Jenkins**


search


Gayatri | log out


Jenkins > Jenkins CLI


 New Item


 People


 Build History


 Project Relationship

 Check File Fingerprint

 Manage Jenkins

 My Views


 Credentials

 New View

Build Queue

No builds in the queue.

Build Executor Status

 **Jenkins CLI**

You can access various features in Jenkins through a command-line tool. See [the documentation](#) for more details of this feature. To get started, download [jenkins-cli.jar](#), and run it as follows:

```
java -jar jenkins-cli.jar -s http://localhost:8080/ help
```

Available Commands

<u>add-job-to-view</u>	Adds jobs to view.
<u>build</u>	Builds a job, and optionally waits until its completion.
<u>cancel-quiet-down</u>	Cancel the effect of the "quiet-down" command.
<u>clear-queue</u>	Clears the build queue.
<u>connect-node</u>	Reconnect to a node(s)
<u>console</u>	Retrieves console output of a build.
<u>copy-job</u>	Copies a job.
<u>create-credentials-by-xml</u>	Create Credential by XML
<u>create-credentials-domain-by-xml</u>	Create Credentials Domain by XML

Jenkins: Pipeline As Code

- Pipeline code uses a DSL
- DSL allows you to perform the tests
- Jenkins text file defined in a txt file, called a Jenkinsfile
- Can defined version controls of Jenkinsfile
- less error –prone execution of jobs
- logic based execution of steps

pipeline : This pipe line is a set of instructions given in the delivery of entail build process

- Node : The machine on which Jenkins is running
- Agent :An agent is an directive multiple builds will runs in single Jenkins instance
- Stage :A stage block containers serous of steps in the pipeline , That is build ,test , and deploy process are in one stage
- Step : A step is a single task that executes a specific process
-

```
1 pipeline {
2     agent any
3     stages {
4         stage ('Initialize') {
5             steps {
6                 sh '''
7                     echo "PATH = ${PATH}"
8                     echo "M2_HOME = ${M2_HOME}"
9                     ...
10                '''
11            }
12        }
13        stage ('Build') {
14            steps {
15                echo 'Hello World'
16            }
17        }
18    }
19 }
```

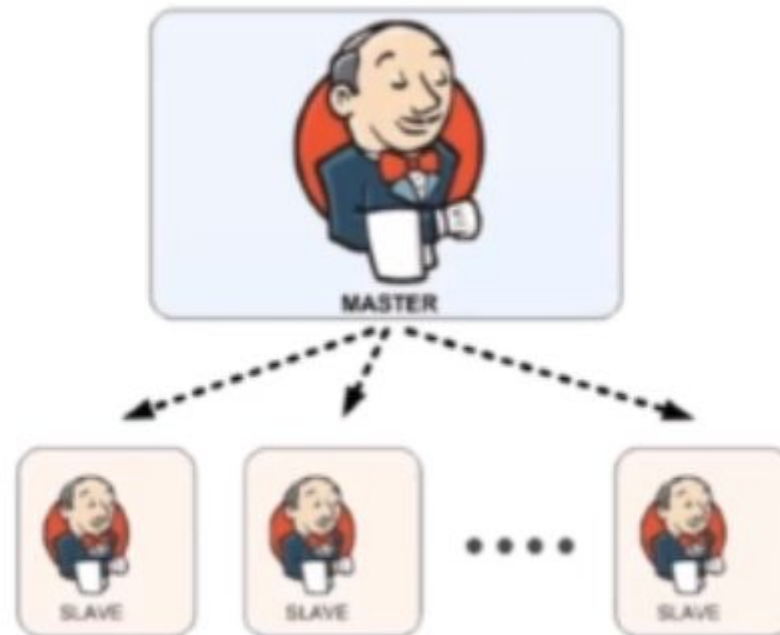
Jenkins – Distributed Builds

Master:

- Schedule build job
- Dispatches Builds to the slave for actual
- Monitoring the slaves and recording the

Slave :

Executes builds jobs dispatched by master



Jenkins – Email configuration

E-mail Notification

SMTP server

smtp.gmail.com

Default user e-mail suffix

☒ Use SMTP Authentication

User Name

.....@gmail.com

provide your email

Password

.....

Use SSL



SMTP Port

465

Reply-To Address

Charset

UTF-8

☒ Test configuration by sending test e-mail

Test e-mail recipient

.....@gmail.com

provide test email

Email was successfully sent

Test configuration

Jenkins - Best Practices

- Always secure Jenkins.
- In larger systems, don't build on the master.
- Always build from Source Control – Clean Builds
- Connect Issue Management or Help Desk System with Jenkins
- Backup Jenkins Home regularly.
- Limit project names to a sane (e.g. alphanumeric) character set
- The most reliable builds will be clean builds, which are built fully from Source Code Control.
- Integrate tightly with your issue tracking system, like JIRA or bugzilla, to reduce the need for maintaining a Change Log
- Always configure your job to generate trend reports and automated testing when running a Java build
- Set up Jenkins on the partition that has the most free disk-space
- Archive unused jobs before removing them.
- Setup a different job/project for each maintenance or development branch you create
- Prevent resource collisions in jobs that are running in parallel.
- Avoid scheduling all jobs to start at the same time
- Set up email notifications mapping to ALL developers in the project, so that everyone on the team has his pulse on the project's current status.
- Take steps to ensure failures are reported as soon as possible.
- Write jobs for your maintenance tasks, such as cleanup operations to avoid full disk problems.
- Tag, label, or baseline the codebase after the successful build.