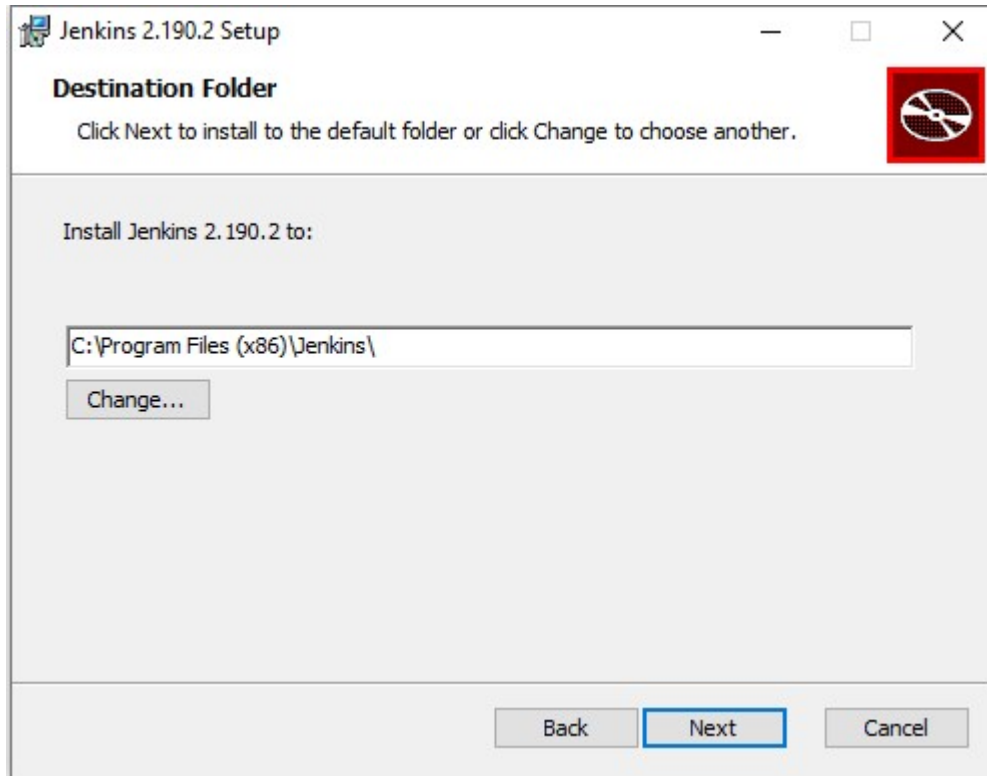
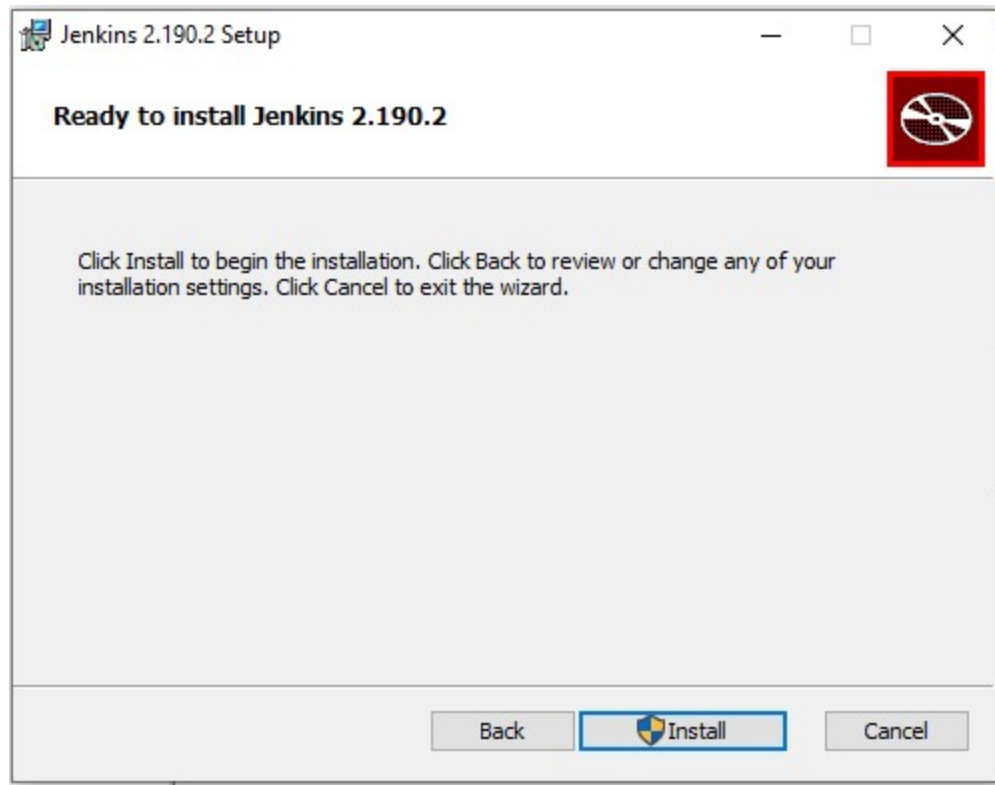


Kick Start Your Test Automation Suite in Jenkins – By Kushan Amarasiri

- Go to Jenkins Site and Download the Setup
<https://jenkins.io/download/thank-you-downloading-windows-installer-stable/>
- Run the installer
- Provide the setup location



- Click Install



- Once installed Jenkins configuration will start on your default browser



Please wait while Jenkins is getting ready to work ...

Your browser will reload automatically when Jenkins is ready.

- To unlock Jenkins find the default password stored in the given location and copy paste it to the input.



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:

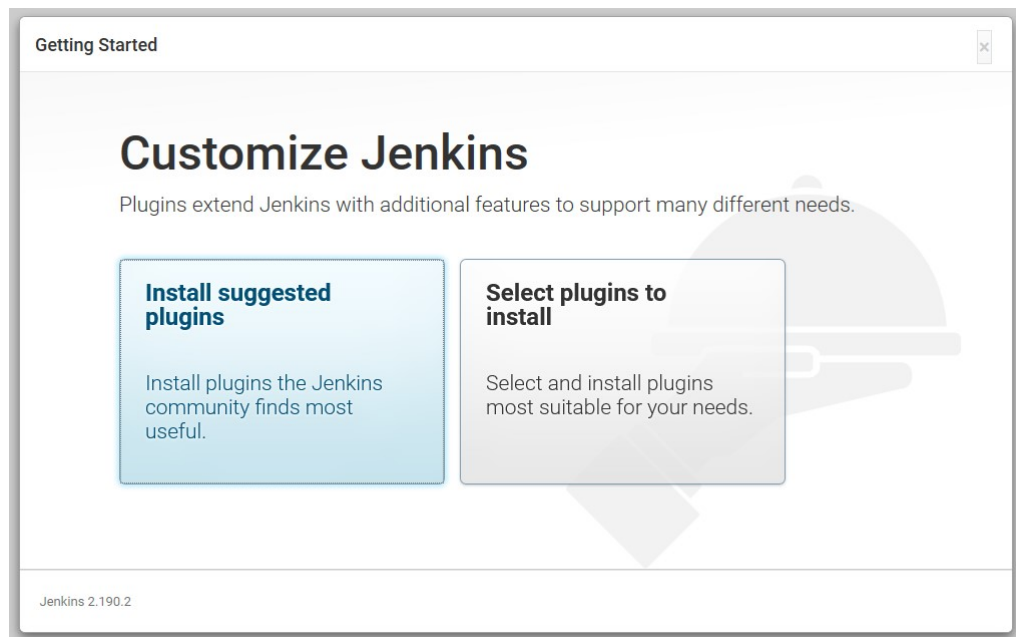
`C:\Program Files (x86)\Jenkins\secrets\initialAdminPassword`

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

Administrator password

Continue

- Click Install Suggested Plugging



Getting Started

Customize Jenkins

Plugins extend Jenkins with additional features to support many different needs.

Install suggested plugins

Install plugins the Jenkins community finds most useful.

Select plugins to install

Select and install plugins most suitable for your needs.

Jenkins 2.190.2

- Allow the default pluggings to download

Getting Started

Getting Started

✓ Folders	✓ OWASP Markup Formatter	✓ Build Timeout	✓ Credentials Binding
✓ Timestampers	✓ Workspace Cleanup	✓ Ant	✓ Gradle
⚙ Pipeline	⚙ GitHub Branch Source	⚙ Pipeline: GitHub Groovy Libraries	✓ Pipeline: Stage View
⚙ Git	⚙ Subversion	⚙ SSH Slaves	⚙ Matrix Authorization Strategy
⚙ PAM Authentication	⚙ LDAP	⚙ Email Extension	✓ Mailer

Gradle
** Pipeline: Milestone Step
** Jackson 2 API
** Pipeline: Input Step
** Pipeline: Stage Step
** Pipeline: Graph Analysis
** Pipeline: REST API
** JavaScript GUI Lib: Handlebars bundle
** JavaScript GUI Lib: Moment.js bundle
Pipeline: Stage View
** Pipeline: Build Step
** Pipeline: Model API
** Pipeline: Declarative
Extension Points API
** JSch dependency
** Git client
** GIT server
** Pipeline: Shared Groovy Libraries
** - required dependency

Jenkins 2.190.2

- Create the first admin user. Provide the necessary credentials

Getting Started

Create First Admin User

Username:

kushanik

Password:

••••••••

Confirm password:

••••••••

Full name:

Kushan Amarasiri

E-mail address:

test101k@gmail.com

Jenkins 2.190.2

Continue as admin

Save and Continue

- Keep the default URL for Jenkins to use.

Getting Started

Instance Configuration

Jenkins URL:

http://localhost:8080/✕

The Jenkins URL is used to provide the root URL for absolute links to various Jenkins resources. That means this value is required for proper operation of many Jenkins features including email notifications, PR status updates, and the `BUILD_URL` environment variable provided to build steps.

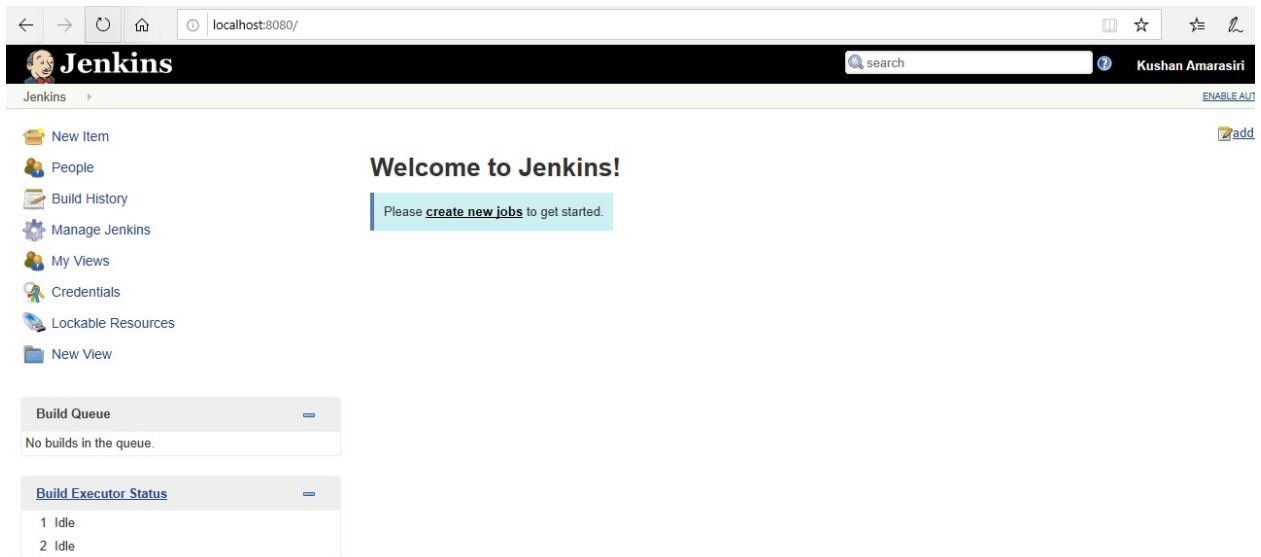
The proposed default value shown is not saved yet and is generated from the current request, if possible. The best practice is to set this value to the URL that users are expected to use. This will avoid confusion when sharing or viewing links.

Jenkins 2.190.2

Not now

Save and Finish

- Click Save and Finish and Jenkins will start right away



- Go to Manage Jenkins -> Global Tool Configuration
(<http://localhost:8080/configureTools/>) to install maven

Jenkins > Global Tool Configuration

Add Git

Gradle

Gradle installations

Add Gradle

List of Gradle installations on this system

Ant

Ant installations

Add Ant

List of Ant installations on this system


Maven

Maven installations


Add Maven

Maven

Name

 Required

☒ Install automatically



Install from Apache

Version

Delete Installer

- Provide name as maven, Select install from Apache and version 3.6.2.


Jenkins > Global Tool Configuration

Maven installations


Add Maven

Maven

Name

 Required

☒ Install automatically



Install from Apache

Version

Delete Installer

Add Installer

Add Maven

List of Maven installations on this system

Docker

Docker installations

Add Docker

List of Docker installations on this system

Save Apply

- Click Apply
- Next go to Jenkins Plugin Manager (<http://localhost:8080/pluginManager/available>)
- Search with the word maven and select the maven integration plugin and install it without restart

Jenkins > Plugin Manager

Back to Dashboard
Manage Jenkins
Update Center

Filter:

Updates Available Installed Advanced

Install	Name	Version
View Job Filters		
<input type="checkbox"/>	Create smart views with exactly the jobs you want. Your smart views can automatically include or exclude jobs by using things like the SCM path or type, the job type, build statuses or trends or triggers, relevance to the logged-in user, email recipients, Maven configuration, job parameterization, and user permissions. Mix and match filters to narrow down to exactly what you want.	2.1.1
<input type="checkbox"/>	Maven Artifact ChoiceListProvider (Nexus) This Plugin adds a new ChoiceListProvider for the "Extensible Choice Plugin" which is able to read Artifact information from a Nexus repository.	1.5.1
<input type="checkbox"/>	Maven Metadata Plugin for Jenkins CI server Adds a build parameter that presents versions of an artifact from a maven repository as a drop down list.	2.0.0
<input type="checkbox"/>	Dependency Analyzer This plugin search for the dependency; analyze results into the maven build output and summarize it.	0.7
<input checked="" type="checkbox"/>	Maven Integration This plug-in provides, for better and for worse, a deep integration of Jenkins and Maven: Automatic triggers between projects depending on SNAPSHOTS, automated configuration of various Jenkins publishers (JUnit, ...).	3.4
<input type="checkbox"/>	Maven SNAPSHOT Check This plugin is used to check if pom.xml contains SNAPSHOT	1.4

- Go to Global tool configuration and install Java. Remove install automatically and add the JAVA_HOME path. Click Apply.

Jenkins > Global Tool Configuration

JDK

JDK installations

Add JDK

JDK

Name

JAVA_HOME

☐ Install automatically


Delete JDK

Add JDK

List of JDK installations on this system

Git

- Go back to the Jenkins dashboard and Select new Item
- Select Maven project and provide a name for the project.

**Jenkins**


Kushan Amarasiri | [log out](#)


Jenkins > All >


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.

**Maven project**
Build a maven project. Jenkins takes advantage of your POM files and drastically reduces the configuration.

**Pipeline**
Orchestrates long-running activities that can span multiple build agents. Suitable for building pipelines (formerly known as workflows) and/or organizing complex activities that do not easily fit in free-style job type.

- Navigate to the created project and provide the location of your pom.xml file. Give goals and options as **clean test** and Click Apply

Jenkins > Selenium >

GeneralSource Code ManagementBuild TriggersBuild EnvironmentPre StepsBuildPost StepsBuild Settings

Post-build Actions

☐ Inspect build log for published Gradle build scans

☐ With Ant ?

Pre Steps

Add pre-build step

Build

Root POM × ?

Goals and options

Advanced...

Post Steps

☐ Run only if build succeeds ☐ Run only if build succeeds or is unstable ☒ Run regardless of build result

SaveApply

Post-build steps run only for successful builds, etc.

At the test suite side please add the following

- Create a testing.xml file and add the test classes that you are going to execute

Example –

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE suite SYSTEM "http://testng.org/testng-1.0.dtd" >
<suite name="example suite 1" verbose="1" >
  <test name="Regression suite 1" >
    <classes>
      <class name="SelPackage.Selenium1"/>
    </classes>
  </test>
</suite>
```

Next add the testing dependency and the following plugging in POM.xml file.

TestNG Dependency-

```
<dependency>
  <groupId>org.testng</groupId>
  <artifactId>testng</artifactId>
  <version>6.9.8</version>
  <scope>test</scope>
</dependency>
<dependency>
```

Plugging -

```
<build>
  <plugins>
    <plugin>
      <groupId>org.apache.maven.plugins</groupId>
      <artifactId>maven-compiler-plugin</artifactId>
      <version>3.8.1</version>
      <configuration>
        <source>1.8</source>
        <target>1.8</target>
      </configuration>
    </plugin>
    <plugin>
      <groupId>org.apache.maven.plugins</groupId>
      <artifactId>maven-surefire-plugin</artifactId>
      <version>3.0.0-M3</version>
      <configuration>
        <suiteXmlFiles>
          <suiteXmlFile>testng.xml</suiteXmlFile>
        </suiteXmlFiles>
      </configuration>
    </plugin>
  </plugins>
</build>
```