

Lab01 :**Write code for a simple user registration form for an event.**

```

<div class="container">
    <h1>EventRegistrationForm</h1>
    <form name="Registration" class="registration-form" onsubmit="return formValidation()">
        <table>
            <tr>
                <td><label for="name">Name:</label></td>
                <td><input type="text" name="name" id="name" placeholder="yourname"></td>
            </tr>
            <tr>
                <td><label for="email">Email:</label></td>
                <td><input type="text" name="email" id="email" placeholder="youremail"></td>
            </tr>
            <tr>
                <td><label for="password">Password:</label></td>
                <td><input type="password" name="password" id="password"></td>
            </tr>
            <tr>
                <td><label for="phoneNumber">PhoneNumber:</label></td>
                <td><input type="number" name="phoneNumber" id="phoneNumber"></td>
            </tr>
            <tr>
                <td><label for="gender">Gender:</label></td>
                <td>Male: <input type="radio" name="gender" value="male">
                    Female: <input type="radio" name="gender" value="female">
                    Other: <input type="radio" name="gender" value="other"></td>
                </tr>
                <tr>
                    <td><label for="language">language:</label></td>
                    <td>
                        <select name="language" id="language">

```

Date:

```

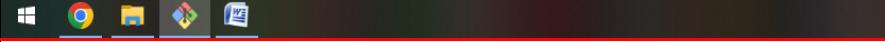
<optionvalue="">Selectlanguage</option>
<optionvalue="English">English</option>
<optionvalue="Spanish">Spanish</option>
<optionvalue="Hindi">Hindi</option>
<optionvalue="Arabic">Arabic</option>
<optionvalue="Russian">Russian</option>
</select>
</td>
</tr>
<tr>
    <td><labelfor="zipcode">ZipCode:</label></td>
    <td><input type="number"
name="zipcode" id="zipcode"></td>
</tr>
<tr>
    <td><labelfor="about">About:</label></td>
    <td><textarea name="about" id="about" placeholder="Writeaboutyourself..."></td>
</tr>
<tr>
    <td colspan="2"><input type="submit"
class="submit" value="Register"/></td>
</tr>
</table>
</form>
</div>

```

Event Registration Form

Name:	<input type="text" value="your name"/>
Email:	<input type="text" value="your email"/>
Password:	<input type="password"/>
Phone Number:	<input type="text"/>
Gender:	Male: <input type="radio"/> Female: <input type="radio"/> Other: <input type="radio"/>
language	<input type="button" value="Select language"/>
Zip Code:	<input type="text"/>
About:	<input type="text" value="Write about yourself..."/>
<input type="button" value="Register"/>	

Lab02 :
Explore Git and Github commands.

GitEnvironmentSetup-GitConfig command	<p><i>Git supports a command called git config that lets you get and set configuration variables that control all facets of how Git looks and operates. It is used to set Git configuration values on a global or local project level. Setting user.name and user.email are the necessary configuration options as your name and email will show up in your commit messages.</i></p> <pre>\$ git config --global user.name "Himanshu Dubey"</pre> <hr/> <pre>\$ git config --global user.email "himanshudubey481@gmail.com"</pre>
Git init command	<p><i>This command is used to create a local repository. The git init command is the first command that you will run on Git. The git init command is used to create a new blank repository.</i></p> <pre>Shashank@DESKTOP-Shashank MINGW64 ~ \$ pwd /c/Users/Shashank Shashank@DESKTOP-Shashank MINGW64 ~ \$ cd Desktop/IIICSM Shashank@DESKTOP-Shashank MINGW64 ~/Desktop/IIICSM \$ git init Initialized empty Git repository in C:/Users/shashank/Desktop/IIICSM/.git/ Shashank@DESKTOP-Shashank MINGW64 ~/Desktop/IIICSM (master) \$ </pre> 

Date:

Git status command	<p>The status command is used to display the state of the working directory and the staging area.</p> <pre>Shashank@DESKTOP-Shashank MINGW64 ~/Desktop/IIICSM (master) \$ git status On branch master No commits yet nothing to commit (create/copy files and use "git add" to track)</pre>
--------------------	---

Git add command	<p>This command is used to add one or more files to staging (Index) area. The git add command is used to add file contents to the Index (Staging Area). This command updates the current content of the working tree to the staging area. It also prepares the staged content for the next commit.</p> <pre>Shashank@DESKTOP-Shashank MINGW64 ~/Desktop/IIICSM (master) \$ git add CSE_AIML.txt Shashank@DESKTOP-Shashank MINGW64 ~/Desktop/IIICSM (master) \$ git status On branch master No commits yet Changes to be committed: (use "git rm --cached <file>..." to unstage) new file: CSE_AIML.txt Shashank@DESKTOP-Shashank MINGW64 ~/Desktop/IIICSM (master) \$ </pre> 
-----------------	---

Git commit command	<p>It is used to record the changes in the repository. It is the next command after the git add. This command commits any files added with git add in the repository and also commits any files you've changed since then.</p> <pre>Shashank@DESKTOP-Shashank MINGW64 ~/Desktop/IIICSM (master) \$ git commit -m "ACE Engineering College" [master (root-commit) a9241d6] ACE Engineering College 1 file changed, 1 insertion(+) create mode 100644 CSE_AIML.txt Shashank@DESKTOP-Shashank MINGW64 ~/Desktop/IIICSM (master) \$ </pre> 
--------------------	---

Date:

Gitlogcommand

*This command is used to check the commit history.
Git log is a utility tool to review and read a history of everything that happens to a repository.*

```
Shashank@DESKTOP-Shashank MINGW64 ~/Desktop/IIICSM (master)
$ git log
commit 8399385419c45c5308f4fd1150af2ce8f810e229 (HEAD -> master)
Author: Shashank <shashankt.ace@gmail.com>
Date:   Thu Feb 9 21:49:20 2023 +0530

    ACE Engineering College_

commit a9241d6f8e8ccc6fc530f6472cbc7f7ae8d233d3
Author: Shashank <shashankt.ace@gmail.com>
Date:   Thu Feb 9 21:46:13 2023 +0530

    ACE Engineering College

Shashank@DESKTOP-Shashank MINGW64 ~/Desktop/IIICSM (master)
$
```

**GitDiffcommand**

*It compares different versions of data sources.
Git diff is a command-line utility. It's a multi use Git command.
When it is executed, it runs a diff function on Git data sources. These data sources can be files, branches, commits, and more. It is used to show changes between commits, commit, and working tree, etc.*

```
Shashank@DESKTOP-Shashank MINGW64 ~/Desktop/IIICSM (master)
$ git diff
diff --git a/CSE_AIML.txt b/CSE_AIML.txt
index d17d2ea..5634494 100644
--- a/CSE_AIML.txt
+++ b/CSE_AIML.txt
@@ -1,3 +1 @@
-i am from CSM
-
-From ACE Engineering college
\ No newline at end of file
+i am from CSM
\ No newline at end of file

Shashank@DESKTOP-Shashank MINGW64 ~/Desktop/IIICSM (master)
$
```



Date:

GitClone

In Git, cloning is the act of making a copy of any target repository. The target repository can be remote or local. You can clone your repository from the remote repository to create a local copy on your system.

The screenshot shows a GitHub repository page for 'ImDwivedi1/Git-Example'. At the top, there's a summary bar with 2 commits, 1 branch, 0 releases, and 1 contributor. Below it, there are buttons for 'Create new file', 'Upload files', 'Find file', and 'Clone or download'. A prominent 'Clone with HTTPS' button is highlighted with a blue border, containing the URL 'https://github.com/ImDwivedi1/Git-Example'. Below this, there are buttons for 'Open in Desktop' and 'Download ZIP'. A large text box at the bottom contains the command '\$ git clone https://github.com/ImDwivedi1/Git-Example.git'.

GitOriginMaster

In Git, The term origin is referred to the remote repository where you want to publish your commits. The default remote repository is called origin, although you can work with several remotes having a different name at the same time. The origin is a short name for the remote repository that a project was initially being cloned. It is used in place of the original repository URL. Thus, it makes referencing much easier.

Central Repository



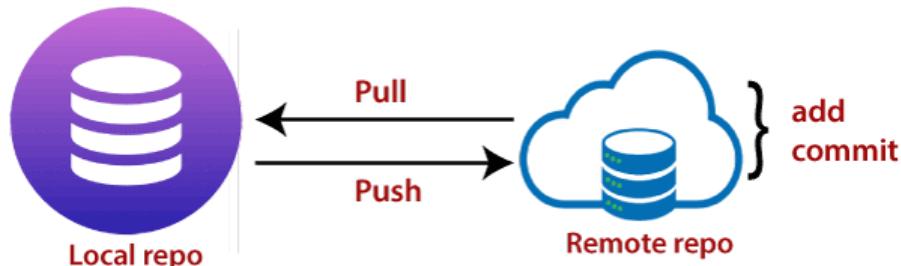
```
Shashank@DESKTOP-Shashank MINGW64 ~/Desktop/IIICSM (master)
$ git remote add origin "https://github.com/get002/csm.git"
```

```
Shashank@DESKTOP-Shashank MINGW64 ~/Desktop/IIICSM (master)
$ |
```



GitPushCommand

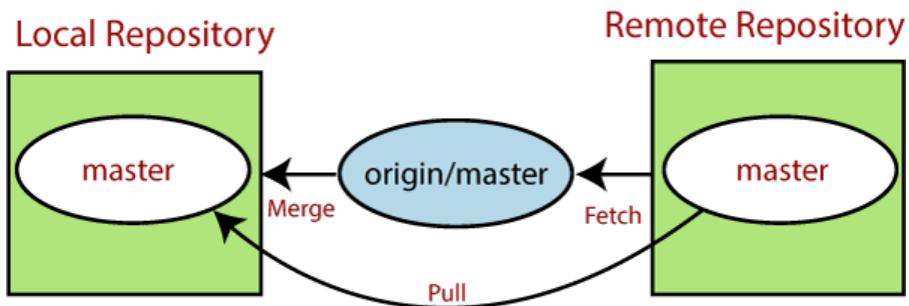
The push term refers to upload local repository content to a remote repository. Pushing is an act of transferring commits from your local repository to a remote repository. Pushing is capable of overwriting changes; caution should be taken when pushing.



```
Shashank@DESKTOP-shashank MINGW64 ~/Desktop/IIICSM (master)
$ git push origin master
Enumerating objects: 14, done.
Counting objects: 100% (14/14), done.
Delta compression using up to 4 threads
Compressing objects: 100% (8/8), done.
Writing objects: 100% (13/13), 1.27 KiB | 118.00 KiB/s, done.
Total 13 (delta 1), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (1/1), done.
remote:
remote: Create a pull request for 'master' on GitHub by visiting:
remote:   https://github.com/get002/csm/pull/new/master
remote:
To https://github.com/get002/csm.git
 * [new branch]      master -> master
Shashank@DESKTOP-shashank MINGW64 ~/Desktop/IIICSM (master)
$ |
```

GitPullCommand

The term *pull* is used to receive data from GitHub. It fetches and merges changes from the remote server to your working directory. The *git pull* command is used to pull a repository.



```

Shashank@DESKTOP-Shashank MINGW64 ~/Desktop/IIICSM (master)
$ git pull origin main
From https://github.com/get002/CSM
 * branch            main      -> FETCH_HEAD
fatal: refusing to merge unrelated histories
  
```

```

Shashank@DESKTOP-Shashank MINGW64 ~/Desktop/IIICSM (master)
$ git pull origin main --allow-unrelated-histories
From https://github.com/get002/CSM
 * branch            main      -> FETCH_HEAD
Merge made by the 'ort' strategy.
 README.md | 1 +
 1 file changed, 1 insertion(+)
 create mode 100644 README.md
  
```

```

Shashank@DESKTOP-Shashank MINGW64 ~/Desktop/IIICSM (master)
$ 
  
```

**GitFetchc
ommand**

Git "fetch" Downloads commits, objects and refs from another repository. It fetches branches and tags from one or more repositories. It holds repositories along with the objects that are necessary to complete their histories to keep updated remote-tracking branches.

```

HiMaNshU@HiMaNshU-PC MINGW64 ~/Desktop/Git-Example (master)
$ git fetch https://github.com/ImDwivedi1/Git-Example.git
warning: no common commits
remote: Enumerating objects: 6, done.
remote: Counting objects: 100% (6/6), done.
remote: Compressing objects: 100% (4/4), done.
remote: Total 6 (delta 0), reused 0 (delta 0), pack-reused 0
Unpacking objects: 100% (6/6), done.
From https://github.com/ImDwivedi1/Git-Example
 * branch            HEAD      -> FETCH_HEAD
  
```

Date:

OperationsonBranches-CreateBranch	<p>You can create a new branch with the help of the git branch command. This command will be used as:</p> <pre>\$ git branch <branch name></pre> <pre>HiMaNshU@HiMaNshU-PC MINGW64 ~/Desktop/GitExample2 (master) \$ git branch B1</pre>
ListBranch	<p>You can list all of the available branches in your repository by using the following command.</p> <p>Either we can use git branch - list or git branch command to list the available branches in the repository.</p> <pre>\$ git branch --list</pre> <pre>HiMaNshU@HiMaNshU-PC MINGW64 ~/Desktop/GitExample2 (master) \$ git branch B1 branch3 * master</pre> <pre>HiMaNshU@HiMaNshU-PC MINGW64 ~/Desktop/GitExample2 (master) \$ git branch --list B1 branch3 * master</pre>
OperationsonBranches-DeleteBranch	<p>You can delete the specified branch. It is a safe operation. In this command, Git prevents you from deleting the branch if it has unmerged changes.</p> <pre>\$ git branch -d <branch name></pre> <pre>HiMaNshU@HiMaNshU-PC MINGW64 ~/Desktop/GitExample2 (master) \$ git branch -d B1 Deleted branch B1 (was 554a122).</pre>
OperationsonBranches-DeleteaRemoteBranch	<p>You can delete a remote branch from Git desktop application.</p> <pre>\$ git push origin -delete <branch name></pre> <pre>HiMaNshU@HiMaNshU-PC MINGW64 ~/Desktop/GitExample2 (master) \$ git push origin --delete branch2 To https://github.com/ImDwivedil/GitExample2 - [deleted] branch2</pre> <pre>HiMaNshU@HiMaNshU-PC MINGW64 ~/Desktop/GitExample2 (master) \$</pre>

Operations on Branches - Switch Branch

Git allows you to switch between the branches without making a commit. You can switch between two branches with the git checkout command.

```
$ git branch -m <old branch name> <new branch name>
```

```
HiMANSHU@HIMANSHU-PC MINGW64 ~/Desktop/GitExample2 (master)
$ git branch -m branch4 renamedB1
```

```
HiMANSHU@HIMANSHU-PC MINGW64 ~/Desktop/GitExample2 (master)
$ git branch
* master
  renamedB1
```

```
HiMANSHU@HIMANSHU-PC MINGW64 ~/Desktop/GitExample2 (master)
$ |
```

Lab03 :

PracticesourcecodemanagementonGitHub.Experimentwiththesourcecodewritteninexercise1.

Check the status of local repository using git status command,

```
Shashank@DESKTOP-Shashank MINGW64 ~/Desktop/WebProject (master)
$ git status
On branch master

No commits yet

Untracked files:
  (use "git add <file>..." to include in what will be committed)
    index.html

nothing added to commit but untracked files present (use "git add" to track)
```

There is a file index.html at working area, Use git add command to add on staging area. Then commit the changes on local repository using git commit command.

```
Shashank@DESKTOP-Shashank MINGW64 ~/Desktop/WebProject (master)
$ git add index.html

Shashank@DESKTOP-Shashank MINGW64 ~/Desktop/WebProject (master)
$ git commit -m "index file"
[master (root-commit) dab7375] index file
 1 file changed, 92 insertions(+)
 create mode 100644 index.html
```

```
Shashank@DESKTOP-Shashank MINGW64 ~/Desktop/WebProject (master)
$ git log
commit dab7375858be19f5a9c3b7e35a230bdbe344a9b6 (HEAD -> master)
Author: Shashank <shashankt.ace@gmail.com>
Date:   Fri Mar 31 18:39:32 2023 +0530

  index file

Shashank@DESKTOP-Shashank MINGW64 ~/Desktop/WebProject (master)
$ git status
On branch master
nothing to commit, working tree clean
```

All changes updated on local repository.

Now to push these changes on remote repository, first create a remote repository.

Date:

ACERESHYD / CSMIII Public

<> Code Issues Pull requests Actions Projects Wiki Security Insights Settings

main 1 branch 0 tags Go to file Add file <> Code

ACERESHYD Initial commit 7a18321 15 minutes ago 1 commit

README.md Initial commit 15 minutes ago

README.md

CSMIII

```
shashank@DESKTOP-shashank MINGW64 ~/Desktop/WebProject (master)
$ git remote add origin git@github.com:ACERESHYD/CSMIII.git
```

Now push the changes from local repository to remote repository using git push origin master

shashank@DESKTOP-shashank MINGW64 ~/Desktop/WebProject (master)

```
$ git push origin master
Enumerating objects: 3, done.
Counting objects: 100% (3/3), done.
Delta compression using up to 4 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 1018 bytes | 169.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
remote:
remote: Create a pull request for 'master' on GitHub by visiting:
remote:     https://github.com/ACERESHYD/CSMIII/pull/new/master
remote:
To github.com:ACERESHYD/CSMIII.git
 * [new branch]      master -> master
```

Search or jump to... Pull requests Issues Codespaces Marketplace Explore

ACERESHYD / CSMIII Public Pin Unwatch Fork Star

<> Code Issues Pull requests Actions Projects Wiki Security Insights Settings

master

Commits on Mar 31, 2023

index file dab7375 get002 committed 30 minutes ago

Task : Makes atleast 6 commits on your remote repository from git bash and also from github to practise source code management.

```
Shashank@DESKTOP-Shashank MINGW64 ~/Desktop/WebProject (master)
$ git add index.html

Shashank@DESKTOP-Shashank MINGW64 ~/Desktop/WebProject (master)
$ git commit -m "index file update 1"
[master 1c1731e] index file update 1
 1 file changed, 1 insertion(+), 1 deletion(-)

Shashank@DESKTOP-Shashank MINGW64 ~/Desktop/WebProject (master)
$ git push origin master
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 4 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 282 bytes | 94.00 KiB/s, done.
Total 3 (delta 1), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (1/1), completed with 1 local object.
To github.com:ACERESHYD/CSMIII.git
  dab7375..1c1731e master -> master

Shashank@DESKTOP-Shashank MINGW64 ~/Desktop/WebProject (master)
$ git log
commit 1c1731eeef8150d21459a893f660aa0e603d9db7b (HEAD -> master, origin/master)
Author: Shashank <shashankt.ace@gmail.com>
Date:   Fri Mar 31 19:14:24 2023 +0530

    index file update 1

commit dab7375858be19f5a9c3b7e35a230bdbe344a9b6
Author: Shashank <shashankt.ace@gmail.com>
Date:   Fri Mar 31 18:39:32 2023 +0530

    index file
```

The screenshot shows a GitHub repository page for 'ACERESHYD / CSMIII'. The 'Code' tab is selected. The 'master' branch dropdown shows three commits:

- index file update 1 (commit 1c1731e, authored by get002 on Mar 31, 2023)
- index file update 1 (commit dab7375, authored by get002 4 minutes ago)
- index file (commit 5b01e01, authored by get002 39 minutes ago)

Here HEAD always represents current commit content. Now comparing latest commit with the previous commit using git diff command

```
shashank@DESKTOP-Shashank MINGW64 ~/Desktop/WebProject (master)
$ git diff 5b01f017 1c1731ee
diff --git a/index.html b/index.html
index 377aeea..f76e7b0 100644
--- a/index.html
+++ b/index.html
@@ -6,7 +6,6 @@
 <h2 align="center"> An Autonomous institute</h2>
 <hr/>
 <title>Student Registration Form</title>
-<hr/>
 <style type="text/css">
 th {
 color: #FFF;
```

Code Issues Pull requests Actions Projects Wiki Security Insights Settings

index file update 1

master

get002 committed 12 minutes ago 1 parent 1c1731e commit 5b01f01

Showing 1 changed file with 1 addition and 0 deletions.

Split Unified

index.html

0 comments on commit 5b01f01

Lock conversation

Lab04:**Jenkins installation and setup, explore the environment.**

- Jenkins is an open source automation tool written in Java programming language.
- Jenkins is free and allows continuous integration.
- What is continuous integration?
 - Continuous integration refers to the build and unit testing stages of the software release process. Every revision that is committed triggers an automated build and test. With continuous delivery, code changes are automatically built, tested, and prepared for a release to production.
- Jenkins is a server-based application and requires a web server like Apache Tomcat.
- Jenkins builds and tests our software projects continuously.
 - This is the main reason for Jenkins to become so popular, continuously monitoring of repeated tasks which arise during the development of a project.
 - Example, if your team is developing a project, Jenkins will continuously test your project builds and show you the errors in early stages of your development.
- Benefits of using Jenkins CI
 - **Reduced Development Cycle** – Since every commit is built and tested, it allows releasing new features to the user faster and with fewer errors.
 - **Shorter Time to Integrate Code** – Before the use of Jenkins CI, integration of code was done manually, thus taking a few days. In some cases, it might happen that the code is not running and it is hard to debug as it might have gone through various commits in the repository. Integrating code after every commit ensures that the functionality is not broken after a commit.
 - **Faster Feedback Loops** – Developers get feedback and improve the code whenever a test breaks during a commit. Otherwise, debugging the issue can be very difficult, given teams would not be sure which commit resulted in the bug.
 - **Automated Workflow** – Teams should not worry about running a manual test for each commit. The Jenkins CI pipeline checks the latest code and builds the code along with the tests. The test can deploy the project in a specific environment if it is green. It can notify the developer by breaking the build.

Jenkins installation

Date:**InstallJavaVersion8**

- Since Jenkins is a Java-based application, therefore Java is a must.
- Download java 8 from the below link: <https://www.oracle.com/java/technologies/downloads/#java8>
- Then install the Java as follows:

Date:

JDK 17 will receive updates under these terms, until at least September 2024.

Java SE Development Kit 17 downloads

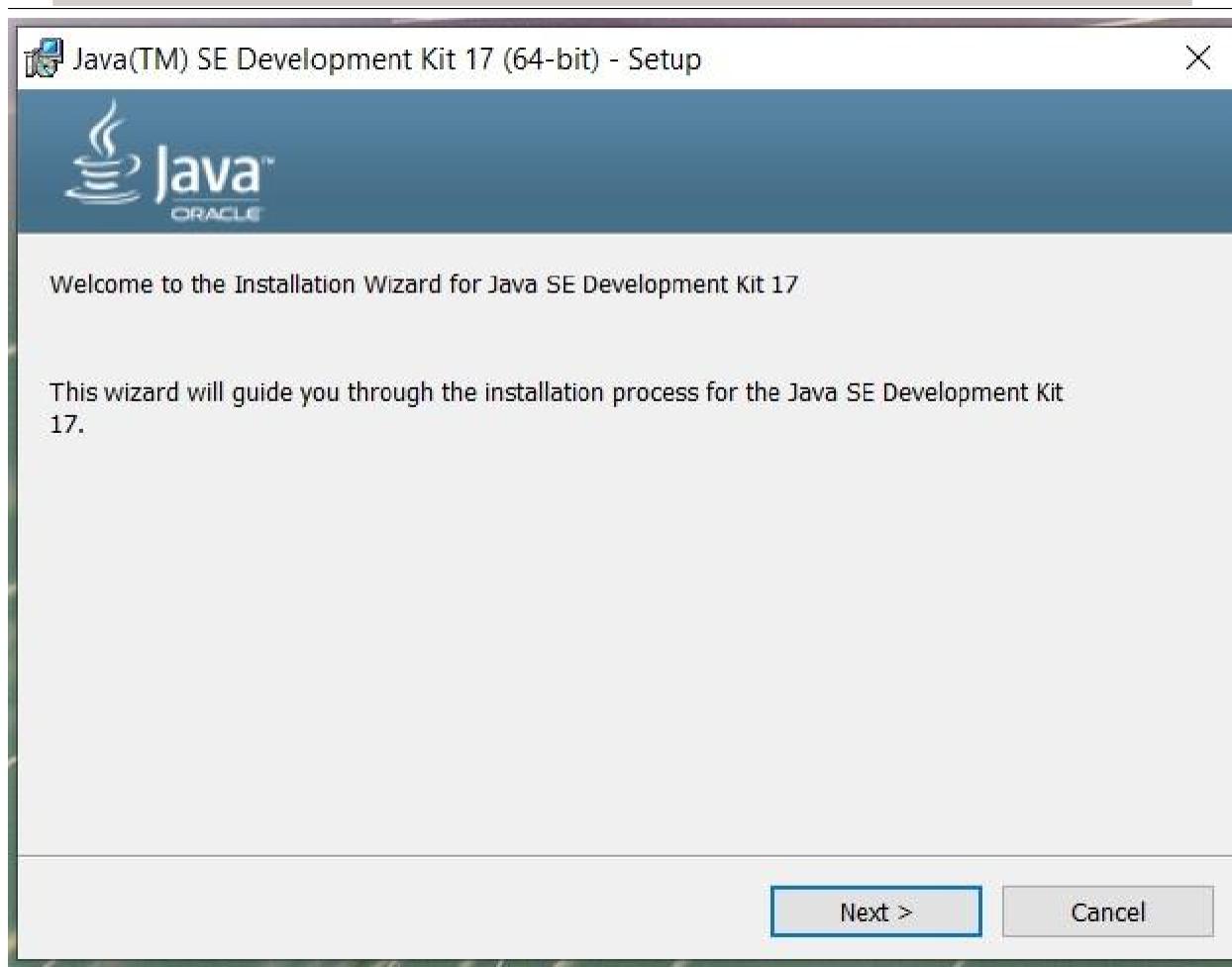
Thank you for downloading this release of the Java™ Platform, Standard Edition Development Kit (JDK™). The JDK is a development environment for building applications and components using the Java programming language.

The JDK includes tools for developing and testing programs written in the Java programming language and running on the Java platform.

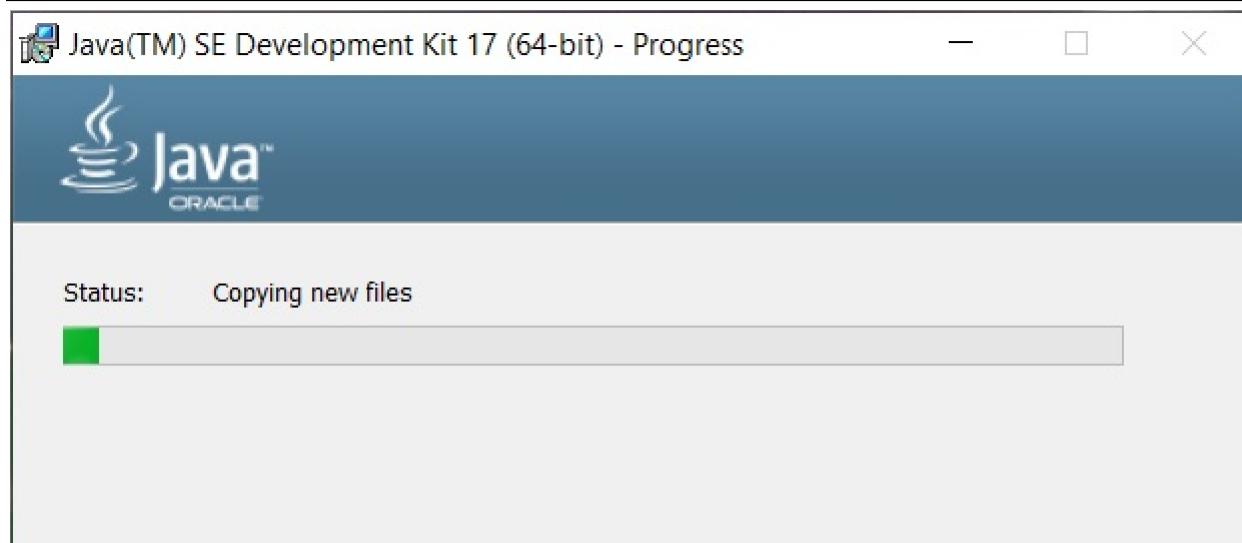
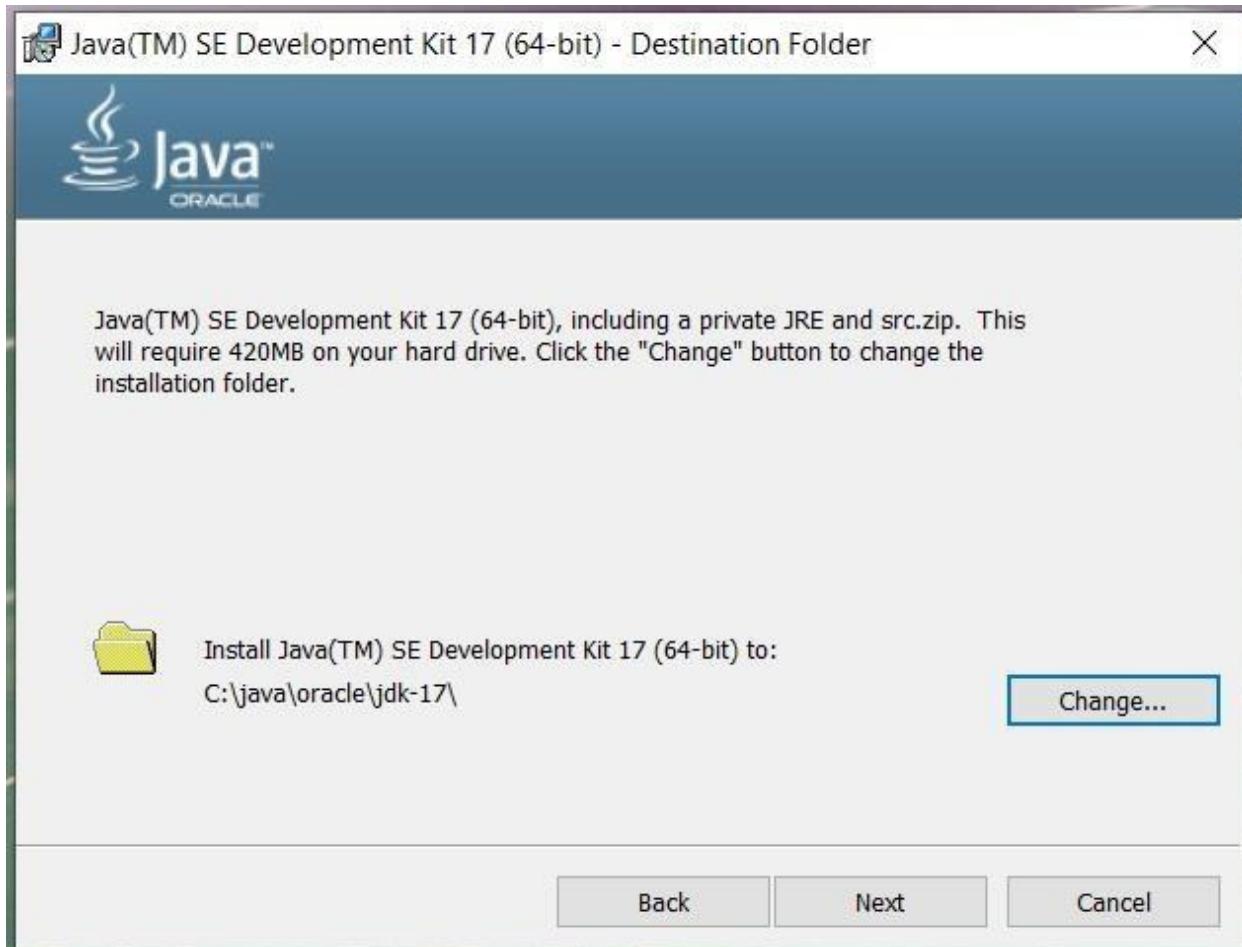
[Documentation Download](#)

[Linux](#) [macOS](#) [Windows](#)

Product/file description	File size	Download
x64 Compressed Archive	170.64 MB	https://download.oracle.com/java/17/latest/jdk-17_windows-x64_bin.zip (sha256
x64 Installer	151.99 MB	https://download.oracle.com/java/17/latest/jdk-17_windows-x64_bin.exe (sha256
x64 MSI Installer	150.88 MB	https://download.oracle.com/java/17/latest/jdk-17_windows-x64_bin.msi (sha256



Date:



Date:



```
Command Prompt
Microsoft Windows [Version 10.0.19042.1237]
(c) Microsoft Corporation. All rights reserved.

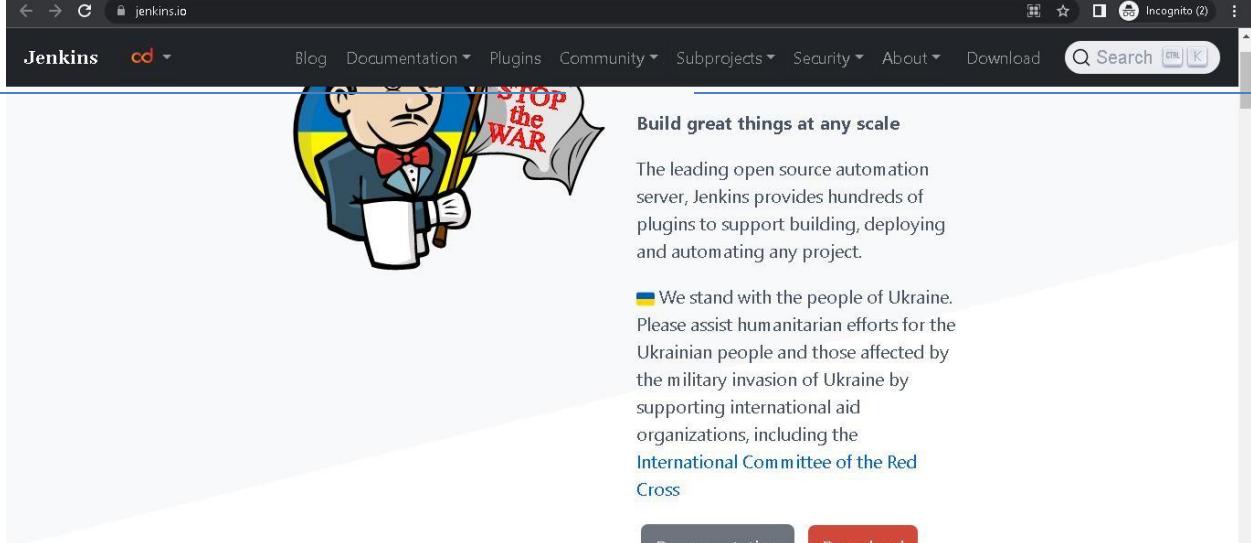
C:\Users\bhagw>java -version
java version "17" 2021-09-14 LTS
Java(TM) SE Runtime Environment (build 17+35-LTS-2724)
Java HotSpot(TM) 64-Bit Server VM (build 17+35-LTS-2724, mixed mode, sharing)

C:\Users\bhagw>
```

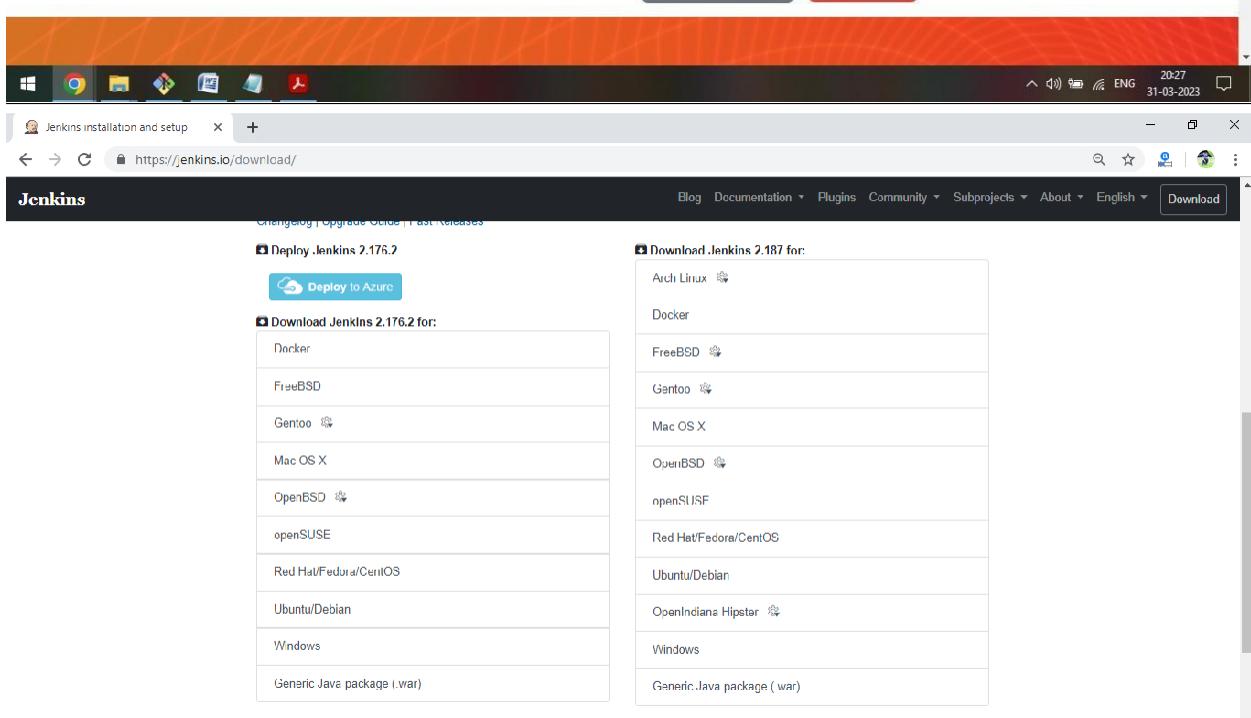
Date:

Download Jenkins war File

Download from the following link
<https://www.jenkins.io/>



The screenshot shows the Jenkins homepage. A cartoon character in a tuxedo holds a sign that says "STOP the WAR". The text on the page reads: "Build great things at any scale. The leading open source automation server, Jenkins provides hundreds of plugins to support building, deploying and automating any project. We stand with the people of Ukraine. Please assist humanitarian efforts for the Ukrainian people and those affected by the military invasion of Ukraine by supporting international aid organizations, including the International Committee of the Red Cross." Below this, there are "Documentation" and "Download" buttons.



The screenshot shows a Windows desktop with a taskbar at the bottom. A browser window is open to the Jenkins download page (<https://jenkins.io/download/>). The page displays the same content as the homepage, including the "STOP the WAR" message and download links for Jenkins 2.176.2 and Jenkins 2.187 across various platforms like Docker, macOS, and Windows.

- Click on Generic Java Package (.war) to download the Jenkins war file.
- Open the command prompt and go to the directory where the Jenkins.war file is located. And then run the following command:

C:/Java-jarJenkins.war

- When you run this command, various tasks will run, one of which

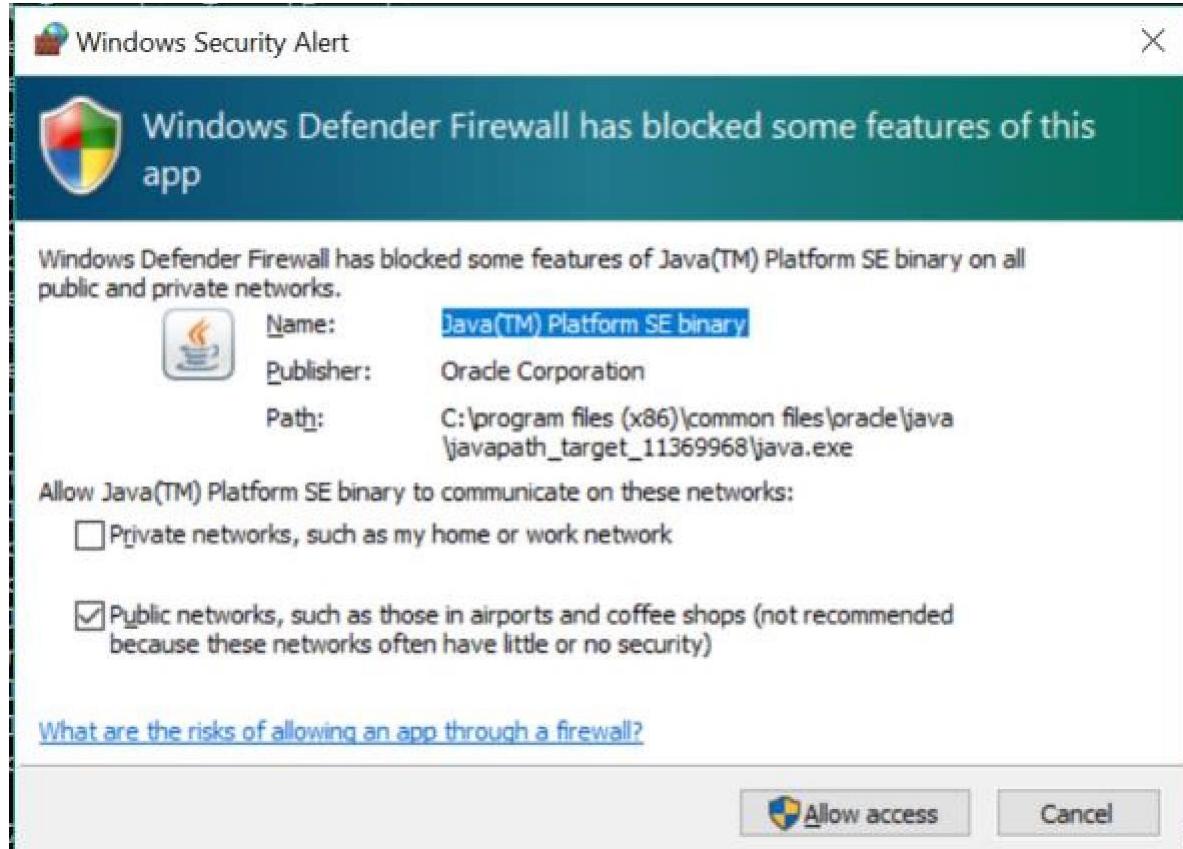
Date:

is the extraction of the war file which is done by an embedded web server called winstone.

```
Command Prompt - Java -jar jenkins.war
Microsoft Windows [Version 10.0.17134.765]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\Users\Nikita>cd c:\jenkins tutorial

c:\Jenkins Tutorial>Java -jar Jenkins.war
Running from: C:\Jenkins Tutorial\jenkins.war
webroot: $user.home/.jenkins
[Winstone 2019/07/06 18:11:26] - Beginning extraction from war file
hudson home directory: C:\Users\Nikita\.jenkins found at: $user.home/.jenkins
[Winstone 2019/07/06 18:11:26] - HTTP Listener started: port=8080
[Winstone 2019/07/06 18:11:26] - AJP13 Listener started: port=8009
Using one-time self-signed certificate
[Winstone 2019/07/06 18:11:26] - Error starting listener instance
java.lang.reflect.InvocationTargetException
    at sun.reflect.NativeConstructorAccessorImpl.newInstance0(Native Method)
    at sun.reflect.NativeConstructorAccessorImpl.newInstance(Unknown Source)
    at sun.reflect.DelegatingConstructorAccessorImpl.newInstance(Unknown Source)
    at java.lang.reflect.Constructor.newInstance(Unknown Source)
    at winstone.Launcher.spawnListener(Launcher.java:232)
    at winstone.Launcher.<init>(Launcher.java:205)
    at winstone.Launcher.main(Launcher.java:391)
    at sun.reflect.NativeMethodAccessorImpl.invoke0(Native Method)
    at sun.reflect.NativeMethodAccessorImpl.invoke(Unknown Source)
    at sun.reflect.DelegatingMethodAccessorImpl.invoke(Unknown Source)
    at java.lang.reflect.Method.invoke(Unknown Source)
    at Main._main(Main.java:214)
    at Main.main(Main.java:61)
Caused by: java.lang.NoClassDefFoundError: sun/security/x509/CertAndKeyGen
    at winstone.ssl.HttpsListener.<init>(HttpsListener.java:111)
    ... 13 more
Caused by: java.lang.ClassNotFoundException: sun.security.x509.CertAndKeyGen
```



Click on **Allow access** button to allow the access.

Date:

```
Command Prompt - Java -jar Jenkins.war
... 14 more

[Winstone 2019/07/06 18:11:26] - Winstone Servlet Engine v0.9.10 running: controlPort=disabled
Jul 06, 2019 6:11:27 PM hudson.model.Hudson$5 onAttained
INFO: Started initialization
Jul 06, 2019 6:11:27 PM hudson.model.Hudson$5 onAttained
INFO: Listed all plugins
Jul 06, 2019 6:11:27 PM hudson.model.Hudson$5 onAttained
INFO: Prepared all plugins
Jul 06, 2019 6:11:27 PM hudson.model.Hudson$5 onAttained
INFO: Started all plugins
Jul 06, 2019 6:11:27 PM hudson.model.Hudson$5 onAttained
INFO: Augmented all extensions
Jul 06, 2019 6:11:27 PM hudson.model.Hudson$5 onAttained
INFO: Loaded all jobs
Jul 06, 2019 6:11:28 PM hudson.model.Hudson$5 onAttained
INFO: Completed initialization
Jul 06, 2019 6:11:28 PM hudson.TcpSlaveAgentListener <init>
INFO: JNLP slave agent listener started on TCP port 55609
Jul 06, 2019 6:12:31 PM hudson.model.DownloadService$Downloadable doPostBack
INFO: Obtained the updated data file for hudson.tasks.Ant.AntInstaller
Jul 06, 2019 6:12:31 PM hudson.model.DownloadService$Downloadable doPostBack
INFO: Obtained the updated data file for hudson.tasks.Maven.MavenInstaller
Jul 06, 2019 6:12:40 PM hudson.model.DownloadService$Downloadable doPostBack
INFO: Obtained the updated data file for hudson.tools.JDKInstaller
Jul 06, 2019 6:12:55 PM hudson.model.UpdateSite doPostBack
INFO: Obtained the latest update center data file for UpdateSource default
```

Accessing Jenkins

Now you can access the Jenkins. Open your browser and type the following url on your browser:

<http://localhost:8080>

This url will bring up the Jenkins dashboard.

The screenshot shows the Jenkins dashboard with the following details:

- Header:** Shows the Jenkins logo, a user profile for "Shashank Tiwari", and a "log out" button.
- Left Sidebar:** Includes links for "Dashboard", "+ New Item", "People", "Build History", "Manage Jenkins", and "My Views".
- Welcome Message:** "Welcome to Jenkins!" with a sub-instruction: "This page is where your Jenkins jobs will be displayed. To get started, you can set up distributed builds or start building a software project."
- Central Area:**
 - "Start building your software project" section with "Create a job" and "→" button.
 - "Build Queue" section showing "No builds in the queue."
 - "Build Executor Status" section showing "1 Idle" and "2 Idle".
 - "Set up a distributed build" and "Set up an agent" sections with "→" buttons.

Exploring the Jenkins environment

- Logintoyourjenkinserver



Welcome to Jenkins!

Keep me signed in

Sign in

Jenkins

Dashboard >

Search (CTRL+K) ? ! 1 ! 1 Shashank Tiwari log out

Welcome to Jenkins!

This page is where your Jenkins jobs will be displayed. To get started, you can set up distributed builds or start building a software project.

Add description

[+ New Item](#) ← People connected on jenkin
[People](#) ← Configure & Manage jenkin server
[Build History](#)
[Manage Jenkins](#)
[My Views](#) ← To create your personalized view

Build Queue Create a job

No builds in the queue.

Build Executor Status Set up a distributed build

Start building your software project

Search bar to search job created on jenkin

Enter an item name

My First Job (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 agents. Suitable for building pipelines (formerly known as workflows) and/or organizing complex activities that do not easily fit in free-style job type.

Multi-configuration project
Suitable for projects that need a large number of different configurations, such as testing on multiple environments, platform-specific builds, etc.

Folder
Creates a container that stores nested items in it. Useful for grouping things together. Unlike view, which is just a filter, a folder creates a separate namespace, so you can have multiple things of the same name as long as they are in different folders.

Multibranch Pipeline
Creates a set of Pipeline projects according to detected branches in one SCM repository.

Organization Folder
Creates a set of multibranch project subfolders by scanning for repositories.

OK

Activate Windows
Go to Settings to activate Windows.

Next, define the parameter with the job

localhost:9090/job/My%20First%20Job/configure

General

Description

Build Triggers

Source Code Management

Build Environment

Save Apply

You can define your GitHub project

Date:

The screenshot shows the Jenkins job configuration page for 'My First Job'. The 'General' tab is selected. In the 'Description' field, it says 'This is my first jenkins job'. Under 'Source Code Management', the 'GitHub project' option is checked, and the 'Project url' field is empty. There are three advanced options below: 'This project is parameterized', 'Throttle builds', and 'Execute concurrent builds if necessary', all of which are unchecked. At the bottom are 'Save' and 'Apply' buttons.

Sourcecodemanagementsection

The screenshot shows the Jenkins job configuration page for 'My First Job'. The 'Source Code Management' tab is selected. It is set to 'Git'. The 'Repositories' section has a 'Repository URL' field which is empty and has a red error message: 'Please enter Git repository.' Below it is a 'Credentials' section with a dropdown menu set to '- none -' and an 'Add' button. At the bottom are 'Save' and 'Apply' buttons.

Date:

Next, Set the trigger of build in build trigger section

The screenshot shows the Jenkins configuration interface for a job named "My First Job". The left sidebar has tabs: General, Source Code Management, Build Triggers (which is selected and highlighted in blue), Build Environment, Build Steps, and Post-build Actions. The main content area is titled "Configure" and contains a "Build Triggers" section. It lists several options with checkboxes: "Trigger builds remotely (e.g. from scripts)" (unchecked), "Build after other projects are built" (unchecked), "Build periodically" (unchecked), "GitHub hook trigger for GITScm polling" (unchecked), and "Poll SCM" (unchecked). At the bottom are "Save" and "Apply" buttons.

We can define the tasks in the build section

The screenshot shows the Jenkins configuration interface for the same job. The left sidebar has tabs: General, Source Code Management, Build Triggers, Build Environment (selected and highlighted in blue), Build Steps, and Post-build Actions. The main content area is titled "Configure" and contains a "Build Environment" section. It has three checkboxes: "Delete workspace before build starts" (unchecked), "Use secret text(s) or file(s)" (unchecked), and "Add timestamps to the Console Output" (checked). A "Filter" dropdown menu is open, listing: Execute Windows batch command, Execute shell, Invoke Ant, Invoke Gradle script, Invoke top-level Maven targets, Run with timeout, and Set build status to "pending" on GitHub commit. At the bottom are "Save" and "Apply" buttons.

Date:

The screenshot shows the Jenkins dashboard interface. At the top, there is a navigation bar with the Jenkins logo, a search bar containing "Search (CTRL+K)", and user information for "Shashank Tiwari". Below the navigation bar, there is a breadcrumb trail "Dashboard >".

On the left side, there are several links: "+ New Item", "People", "Build History", "Manage Jenkins", and "My Views".

In the center, there is a table titled "All" showing a single job entry:

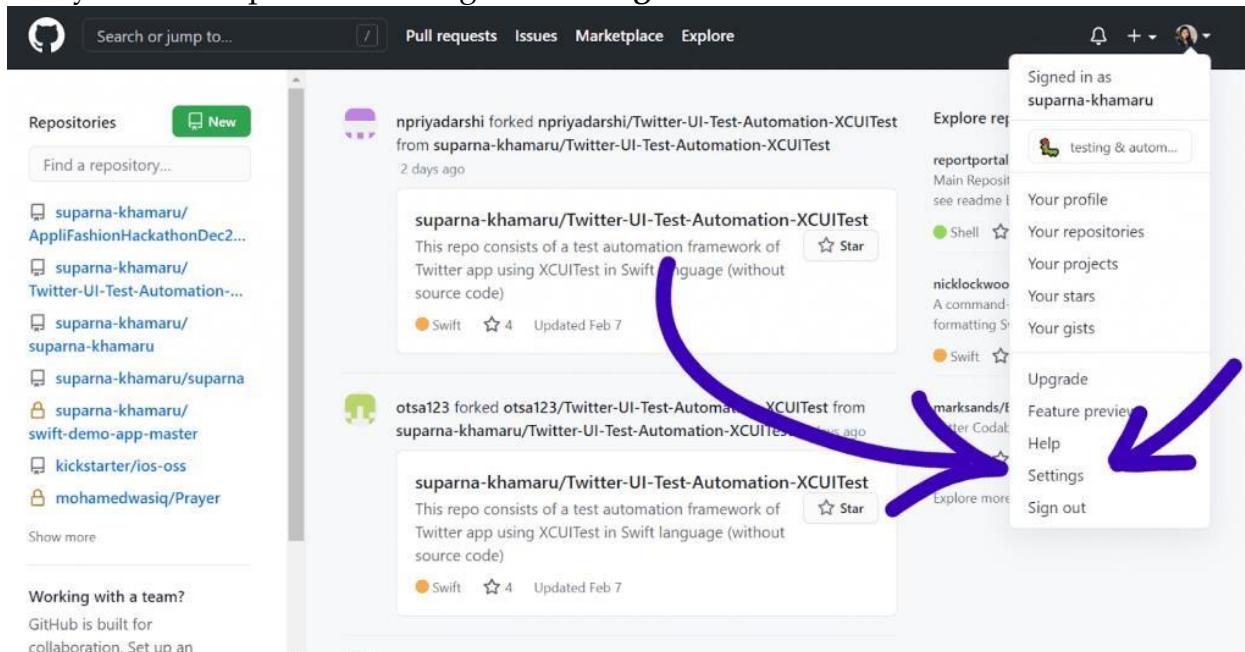
S	W	Name	Last Success	Last Failure	Last Duration
		My First Job	N/A	38 sec #1	0.18 sec

Below the table, there are sections for "Build Queue" (which is empty) and "Build Executor Status".

At the bottom, there is an "Icon legend" and links for "Atom feed for all", "Atom feed for failures", and "Atom feed for just latest builds".

Lab05:**Demonstrate continuous integration and development using Jenkins.**

[Jenkins with GitHub]

Go to your GitHub profile and navigate to **Settings**.

In the settings screen, click on the "Developer settings" menu and click on "Personal access tokens."

In the "Personal access tokens" tab, click on the "Generate new token" button and provide necessary details as desired (an example is provided in the below figure), and click on the "Generate token" button.

Personal access tokens	
jenkins-integration — admin:repo_hook, notifications, repo	Last used within the last week
Delete	

Personal access tokens function like ordinary OAuth access tokens. They can be used instead of a password for Git over HTTPS, or can be used to [authenticate to the API over Basic Authentication](#).

Post successful creation of the token, the newly generated secret text in GitHub is to be copied for future usage in Jenkins.

Configuring Jenkins for GitHub Integration

Next, go to Jenkins dashboard.

Date:

- Click on “Manage Jenkins.”
- Click on “Configure System” and go to the following ‘GitHub’ section.

GitHub Servers

GitHub Server

Name: github

API URL: https://api.github.com

Credentials: jenkins-integration

Credentials verified for user suparna-khamaru, rate limit: 4999

Manage hooks

Note: If the above “GitHub Server” section is found missing in Jenkins, make sure to manually install the missing GitHub plugin from the installed list of tools shown below.

Steps to follow - Go to: Jenkins Dashboard -> Manage Jenkins -> Manage Plugins -> Available tab -> Enter Git in search bar and filter -> Install required plugin

localhost:8080/pluginManager/installed

Plugin Manager

Git client plugin	Utility plugin for Git support in Jenkins	3.6.0	<input type="button" value="Uninstall"/>
Git plugin	This plugin integrates Git with Jenkins.	4.5.2	<input type="button" value="Uninstall"/>
GIT server Plugin	Allows Jenkins to act as a Git server.	1.9	<input type="button" value="Uninstall"/>
GitHub	This plugin integrates GitHub to Jenkins.	1.32.0	<input type="button" value="Uninstall"/>
GitHub API Plugin	This plugin provides GitHub API for other plugins.	1.122	<input type="button" value="Uninstall"/>
GitHub Branch Source	Multibranch projects and organization folders from GitHub. Maintained by CloudBees, Inc.	2.9.4	<input type="button" value="Uninstall"/>

Make sure to add the copied secret key in the above credentials by:

- Click on the “Add” -> “Jenkins” button in the GitHub Server section’s Credentials field.

Date:

- Select "Secrettext" from the "Kind" dropdown.

Jenkins Credentials Provider: Jenkins

Add Credentials

Domain: Global credentials (unrestricted)

Kind: **Secret text**

Password: [empty field]

ID: [empty field]

Description: [empty field]

Add **Cancel**

Paste the previously copied secret text from GitHub in the **Secret** field as displayed below, while providing an ID such as "jenkins-integration," and then click on the "Add" button.

Jenkins Credentials Provider: Jenkins

Add Credentials

Domain: Global credentials (unrestricted)

Kind: Secret text

Scope: Global (Jenkins, nodes, items, all child items, etc)

Secret:

ID: jenkins-integration1

Description: [empty field]

Add **Cancel**

Once the Secret text is successfully added, let us test the connection by clicking on the "Testconnection" button and verify the confirmation message as highlighted below.

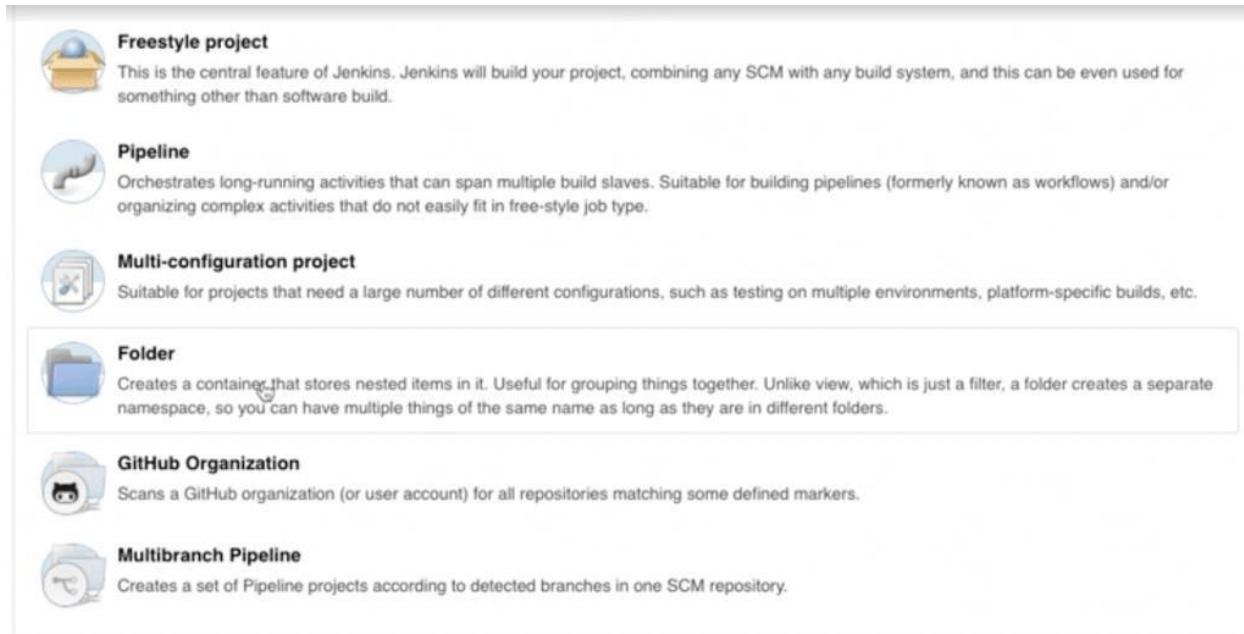
Date:

Creating Your First Jenkins Job Integrated Into a Test Project

Click on "New Item" in the Jenkins user interface dashboard on the left side.

Let us start with creating a Freestyle project in Jenkins for ease of understanding for a beginner. However, we can choose to select any of the following:

Date:



Enter a project name in the requested textbox field and select the **Freestyle project** as shown on the image above.

In the **General** section,

check the field “**GitHub project**” and enter a valid Ant project as given below.

localhost:8080/job/GithubProject/configure

Jenkins

Dashboard > GithubProject >

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

Description

[Plain text] Preview

Discard old builds GitHub project

Project url: https://github.com/suparna-khamaru/rps-ant/ Advanced...

This build requires lockable resources

Then, go to the “**Source Code Management**” section and enter the same link as above with an extra .git extension at the end of the URL, as shown in the below screen.

Date:

Source Code Management

- Repository URL: `https://github.com/suparna-khamaru/rps-ant.git`
- Credentials: - none -
- Branch Specifier (blank for 'any'): `*/master`
- Repository browser: (Auto)

Go to the "Build Environment" section and add the following steps shown in the image below.
 Make sure to add the command in the Targets field under the Ant Build step:
`clean compile test package war`

Build Environment

- With Ant
- Ant Version: ant 1.10.9

Build

Invoke Ant

- Ant Version: ant 1.10.9
- Targets: `clean compile test package war`

Date:

Click on the "Apply" button and then click on the "Save" button.

The screenshot shows the Jenkins configuration interface. In the 'Build' section, there is a 'Invoke Ant' panel with 'Ant Version' set to 'ant 1.10.9' and 'Targets' set to 'clean compile test package war'. Below this is an 'Add build step' button. In the 'Post-build Actions' section, there is an 'Add post-build action' dropdown. At the bottom are 'Save' and 'Apply' buttons.

Once the new Jenkins configurations are successfully saved, the user shall be navigated back to the Project dashboard screen, where the user now clicks on the "Build Now" menu on the left-hand side of the displayed dashboard screen.

The screenshot shows the Jenkins Project GitHubProject dashboard. The left sidebar includes links for Back to Dashboard, Status (highlighted in grey), Changes, Workspace, Build Now, Configure, Delete Project, GitHub, and Rename. The main content area has a title 'Project GitHubProject'. It features 'Recent Changes' and 'Permalinks' sections. The 'Permalinks' section lists the following recent builds:

- Last build (#16), 18 min ago
- Last stable build (#16), 18 min ago
- Last successful build (#16), 18 min ago
- Last failed build (#13), 37 min ago
- Last unsuccessful build (#14), 36 min ago
- Last completed build (#16), 18 min ago

localhost:8080/job/GithubProject/

Jenkins

Dashboard > GithubProject >

- Back to Dashboard
- Status
- Changes
- Workspace
- Build Now**
- Configure
- Delete Project
- Rename

Project GithubProject

- Workspace
- Recent Changes

Permalinks

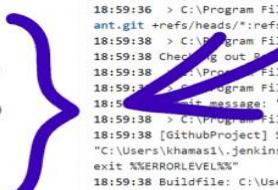


Click on the "Build Now" button and allow the scheduled build to complete.

localhost:8080/job/GithubProject/19/console

Console Output

```
18:59:36 Started by user Suparna Khamaru
18:59:36 Running as SYSTEM
18:59:36 Building in workspace C:\Users\khamas1\.jenkins\workspace\GithubProject
18:59:36 The recommended git tool is: NONE
18:59:36 No credentials specified
18:59:36 > C:\Program Files\Git\bin\git.exe rev-parse --is-inside-work-tree # timeout=10
18:59:36 Fetching changes from the remote Git repository
18:59:36 > C:\Program Files\Git\bin\git.exe config remote.origin.url https://github.com/suparna-khamaru/rps-ant.git #
timeout=10
18:59:36 Fetching upstream changes from https://github.com/suparna-khamaru/rps-ant.git
18:59:36 > C:\Program Files\Git\bin\git.exe --version # timeout=10
18:59:36 > git --version # 'git' version 2.30.0.windows.2'
18:59:36 > C:\Program Files\Git\bin\git.exe fetch --tags --force --progress -- https://github.com/suparna-khamaru/rps-
ant.git +refs/heads/*:refs/remotes/origin/* timeout=10
18:59:36 > C:\Program Files\Git\bin\git.exe rev-parse "refs/remotes/origin/master^{commit}" # timeout=10
18:59:38 Checking out Revision be@af76702244d39c1e3369a095cd52bb432ae3d (refs/remotes/origin/master)
18:59:38 > C:\Program Files\Git\bin\git.exe config core.sparsecheckout # timeout=10
18:59:38 > C:\Program Files\Git\bin\git.exe checkout -f be@af76702244d39c1e3369a095cd52bb432ae3d # timeout=10
18:59:38 [git] fatal: not a git repository (or any of the upstream paths): .git
18:59:38 [git] fatal: failed to read object 39c1e3369a095cd52bb432ae3d
18:59:38 [GithubProject] $ cmd.exe /C
"C:\Users\khamas1\.jenkins\tools\hudson.tasks.Ant_AntInstallation\ant_1.10.9\bin\ant.bat clean compile test package war &&
exit %ERRORLEVEL%"
18:59:38 Buildfile: C:\Users\khamas1\.jenkins\workspace\GithubProject\build.xml
18:59:39
```



Lab06:**Explore Docker commands for content management.**

docker-version	<p>This command is used to get the currently installed version of docker</p> <pre>edureka@Manager-1: ~ edureka@Manager-1:~\$ docker --version Docker version 17.05.0-ce, build 89658be edureka@Manager-1:~\$</pre>
dockerpull	<p>Usage: docker pull <imagename></p> <p>This command is used to pull images from the docker repository (hub.docker.com)</p> <pre>edureka@Manager-1: ~ edureka@Manager-1:~\$ docker pull ubuntu Using default tag: latest latest: Pulling from library/ubuntu ae79f2514705: Pull complete c59d01a7e4ca: Pull complete 41ba73a9054d: Pull complete f1bbfd495cc1: Pull complete 0c346f7223e2: Pull complete Digest: sha256:6eb24585b1b2e7402600450d289e Status: Downloaded newer image for ubuntu:latest edureka@Manager-1:~\$</pre>
dockerrun	<p>Usage: docker run -it -d <imagename></p> <p>This command is used to create a container from an image</p> <pre>edureka@Manager-1: ~ edureka@Manager-1:~\$ docker run -it -d ubuntu f49b58b66d1ebc7c2d9e42280c1e24019b3202fb3e69050c edureka@Manager-1:~\$</pre>
dockersps	<p>This command is used to list the running containers</p> <pre>edureka@Manager-1: ~ edureka@Manager-1:~\$ docker ps CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS f49b58b66d1e ubuntu "/bin/bash" 6 minutes ago Up 6 mi nutes edureka@Manager-1:~\$</pre>

dockerps -a	<p>This command is used to show all the running and exited containers</p> <pre>edureka@Manager-1:~\$ docker ps -a CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS fe6e370a1c9c ubuntu "/bin/bash" 12 seconds ago Up 11 seconds boring_ritchie 2b86a0703d4f ubuntu "/bin/bash" About a minute ago Exited (0) 21 seconds ago infallible_galileo edureka@Manager-1:~\$</pre>
dockerexec	<p>Usage: docker exec -it <containerid> bash</p> <p>This command is used to access the running container</p> <pre>root@fe6e370a1c9c:/ edureka@Manager-1:~\$ docker exec -it fe6e370a1c9c bash root@fe6e370a1c9c:/#</pre>
dockerstop	<p>Usage: docker stop <containerid></p> <p>This command stops a running container</p> <pre>edureka@Manager-1:~\$ docker stop fe6e370a1c9c fe6e370a1c9c edureka@Manager-1:~\$</pre>

dockerkill	<p>Usage: docker kill <containerid></p> <p>This command kills the container by stopping its execution immediately. The difference between 'docker kill' and 'docker stop' is that 'docker stop' gives the container time to shutdown gracefully, in situations when it is taking too much time for getting the container to stop, one can opt to kill it</p> <pre>edureka@Manager-1:~\$ docker kill d611cbc3789c d611cbc3789c edureka@Manager-1:~\$</pre>
-------------------	--

dockercommit	<p>Usage: dockercommit<containerid><username/imagename></p> <p>This command creates a new image of an edited container on the local system</p> <pre>edureka@Manager-1:~\$ docker commit fe6e370a1c9c hshar/ubuntunew sha256:0678ee2e6b1e6a66ae7179c3be31610e5338d3004c52d25fc9f65fd2a63dc164 edureka@Manager-1:~\$</pre>
dockerlogin	<p>This command is used to log into the Docker Hub repository</p> <pre>edureka@Manager-1:~\$ docker login Login with your Docker ID to push and pull images from Docker Hub. If you don't have a Docker ID, head over to https://hub.docker.com to create one. Username (hshar): hshar Password: Login Succeeded edureka@Manager-1:~\$</pre>
dockerpush	<p>Usage: dockerpush<username/imagename></p> <p>This command is used to push an image to the Docker Hub repository</p> <pre>edureka@Manager-1:~\$ docker push hshar/ubuntunew The push refers to a repository [docker.io/hshar/ubuntu] 3e69d0f539f1: Pushed 174a611570d4: Mounted from library/ubuntu f51f76255b02: Mounted from library/ubuntu 51db18d04d72: Mounted from library/ubuntu f1c896f31e49: Mounted from library/ubuntu 0f5ff0cf6a1c: Mounted from library/ubuntu latest: digest: sha256:fdecdf8f2195a17e35b41e87f size: 1564 edureka@Manager-1:~\$</pre>

dockerimages	<p>This command lists all the locally stored Docker images</p> <pre>edureka@Manager-1:~\$ docker images REPOSITORY TAG IMAGE ID CREATED SIZE hshar/ubuntunew latest 0678ee2e6b1e 6 minutes ago 122MB ubuntu latest dd6f76d9cc90 9 days ago 122MB edureka@Manager-1:~\$</pre>
dockerrm	<p>Usage: dockerrm<containerid></p> <p>This command is used to delete a stopped container</p> <pre>edureka@Manager-1:~\$ docker rm 2b86a0703d4f 2b86a0703d4f edureka@Manager-1:~\$</pre>

dockerrmi	<p>Usage: dockerrmi <image-id></p> <p>This command is used to delete an image from local storage</p> <pre>edureka@Manager-1:~\$ docker rmi 0678ee2e6b1e Untagged: hshare/ubuntu:latest Untagged: hshare/ubuntu@sha256:fdecdf8f2195a17e35b41e87fdd96293baf85ec102d045a9deb80d714a1d6950 Deleted: sha256:0678ee2e6b1e6a66ae7179c3be31610e5338d3004c52d25fc9f65fd2a63dc164 Deleted: sha256:f3aea8a0f4950bab665799e13f12d394e06c766df8f2465c7583db95397a5c24 edureka@Manager-1:~\$</pre>
dockerbuild	<p>Usage: dockerbuild <path to dockerfile></p> <p>This command is used to build an image from a specified dockerfile</p> <pre>edureka@Manager-1:~/hello\$ docker build . Sending build context to Docker daemon 3.072kB Step 1/2 : FROM ubuntu --> dd6f76d9cc90 Step 2/2 : RUN echo successfully ran the dockerfile --> Running in 835217ac24f5 successfully ran the dockerfile --> bf3de26a35f0 Removing intermediate container 835217ac24f5 Successfully built bf3de26a35f0 edureka@Manager-1:~/hello\$</pre>

CleanUpCommands

Command	Explanation
dockerimageprune	Clears an unused image
dockerimageprune-a	Clears all images that are not being used by containers
dockersystemprune	Removes all stopped containers, all networks not used by containers, all dangling images, and all build cache
docker imagermimage	Removes an image
dockerrmcontainer	Removes a running container
dockerkill\$(dockerps-q)	Stops all running containers
dockerswarmleave	Leaves a swarm
dockerstackrmstackname	Removes a swarm

Date:

dockervolumerm\$(dockervolu mels -fdangling=true-q)	Removes all dangling volumes
dockerrm\$(dockerps-a-q)	Removes all stopped containers
dockerkill\$(dockerps-q)	Stops all running containers

<u>Container Interaction Commands</u>	<u>and</u>
<u>Command</u>	<u>Explanation</u>
docker start container	Starts a new container
docker stop container	Stops a container
docker pause container	Pauses a container
docker unpause container	Unpauses a container
docker restart container	Restarts a container
docker wait container	Blocks a container
docker export container	Exports container contents to a tar archive
docker attach container	Attaches to a running container
docker wait container	Waits until the container terminates and shows the exit code
docker commit -m “commit message” - a “author” container user na me / image_name : tag	Saves a running container as an image
docker logs -f container	Follows container logs
docker exec -t container script.sh	Runs a command in a container

Date:

dockercommitcontainer image

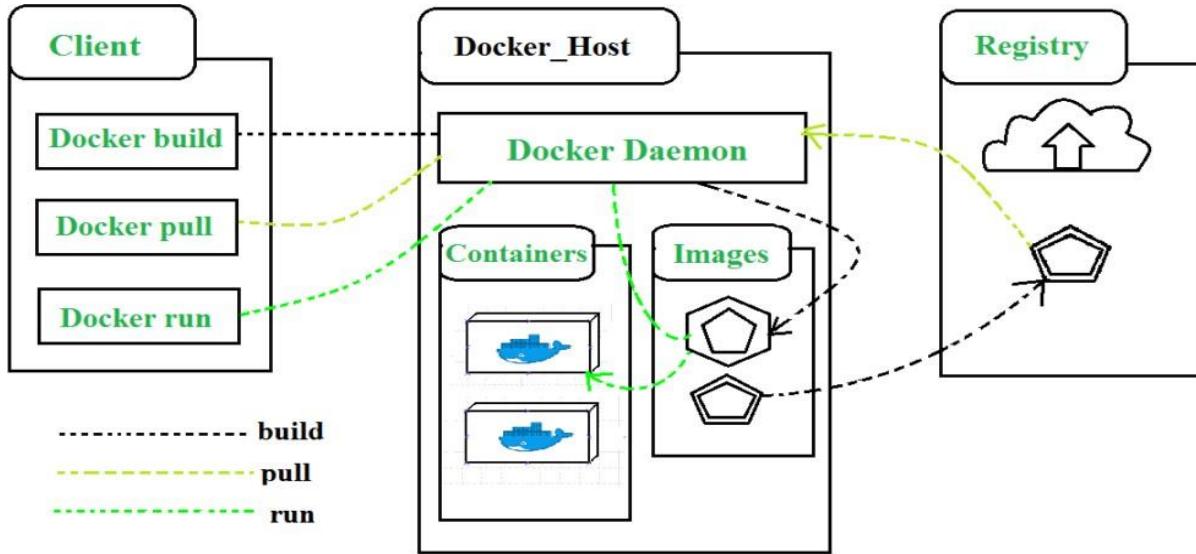
Creates a new image from a container

dockercreateimage

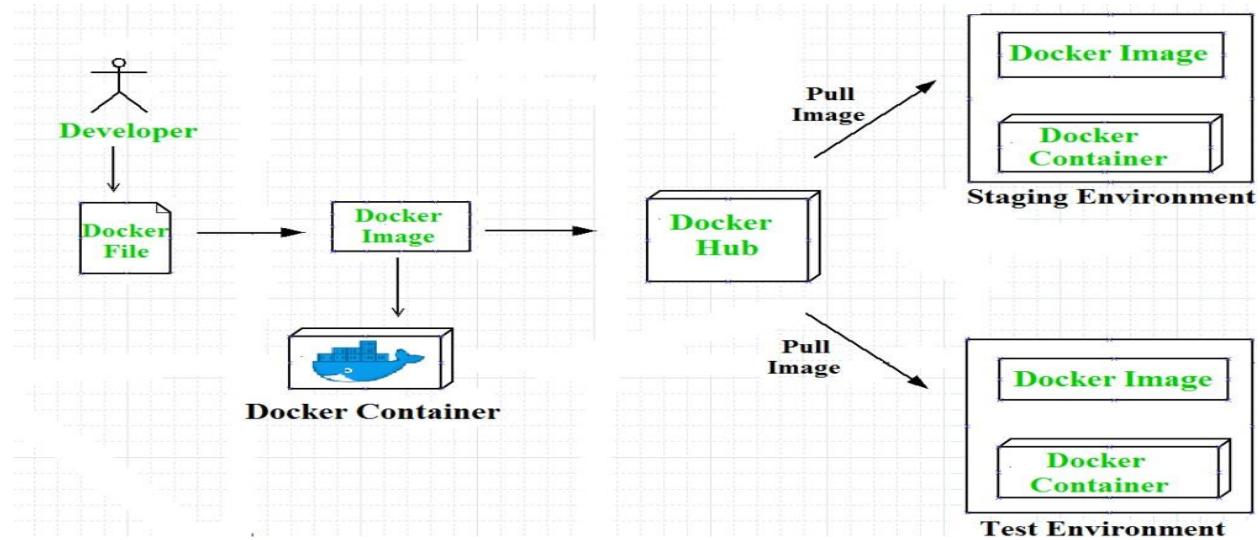
Creates a new container from an image

Lab07:Develop a simple containerized application using Docker.**Docker Architecture**

Docker architecture consists of Docker client, Docker Daemon running on Docker Host, and Docker Hub repository.

**Components of Docker**

The main components of Docker include - Docker clients and servers, Docker images, Dockerfile, Docker Registries, and Docker containers.



First create a JavaScript file with a single line of code. It is just a print statement on the console.

```
// just a print statement on console
```

Date:

```
console.log('HelloDocker!');
```

Creating the Dockerfile is start with the base image. This base image has a bunch of files. So, we take those files and add additional customization to them. I used Node image which builds on top of the Linux image. You can find any official images from the docker hub.

FROM

node:alpine

```
COPY ./app
WORKDIR
/app
CMD node app.js
```

FROM – Define the base image

COPY – Copy files from the current directory to /app directory in the image
WORKDIR – Set the current working directory in the image

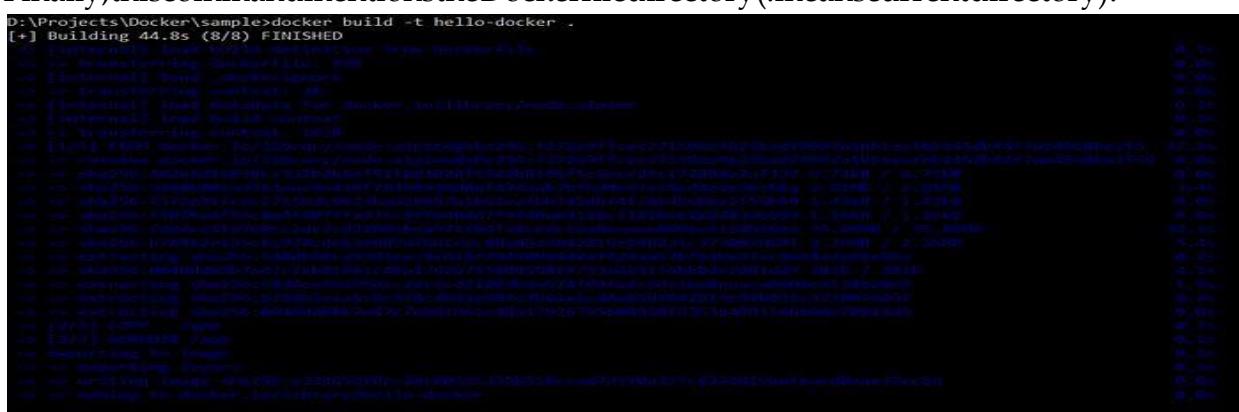
CMD –

Write the command that should be executed (Here, I execute the node command) Then we need to execute a command as below. It will build the Docker package for our application.

docker build -t hello-docker .

The -t is a tag to identify the image. Then, the hello-docker is a specific name that can be used to access it.

Finally, this command mentions the Dockerfile directory (. means current directory).



Then, I can see all the images on my computer with this command.

D:\Projects\ Docker\sample> docker images				
REPOSITORY	TAG	IMAGE ID	CREATED	SIZE
hello-docker	latest	e33465df0cc2	About a minute ago	113MB
docker/getting-started	latest	3ba8f2ff0727	2 months ago	27.9MB

docker images

Now, we can run this image on any computer that is running Docker. Let's run it with this command.

```
dockerrunhello-docker
```

```
D:\Projects\ Docker\sample> docker run hello-docker
Hello Docker!
```

Lab08:

Integrate Kubernetes and Docker.

What is Kubernetes?

[Kubernetes](#) is an open-source orchestration tool that allows you to run and manage your container-based workloads. [Kubernetes](#) was developed by Google and was later donated to the Cloud Native Computing Foundation. Kubernetes helps you manage hundreds and thousands of containerized applications in different deployable environments, be it physical machines, virtual machines, clouds or even hybrid environments!

How Does Kubernetes Work?

Kubernetes was made to solve the problems of the Monolith approach of application deployment used to face. This is where the Microservice type of approach comes into the picture. To get to know why Microservice is the business as of today, have a look at Monolithic vs Microservices. Kubernetes and the Microservice approach has made the lives of so many tech-giants easier!

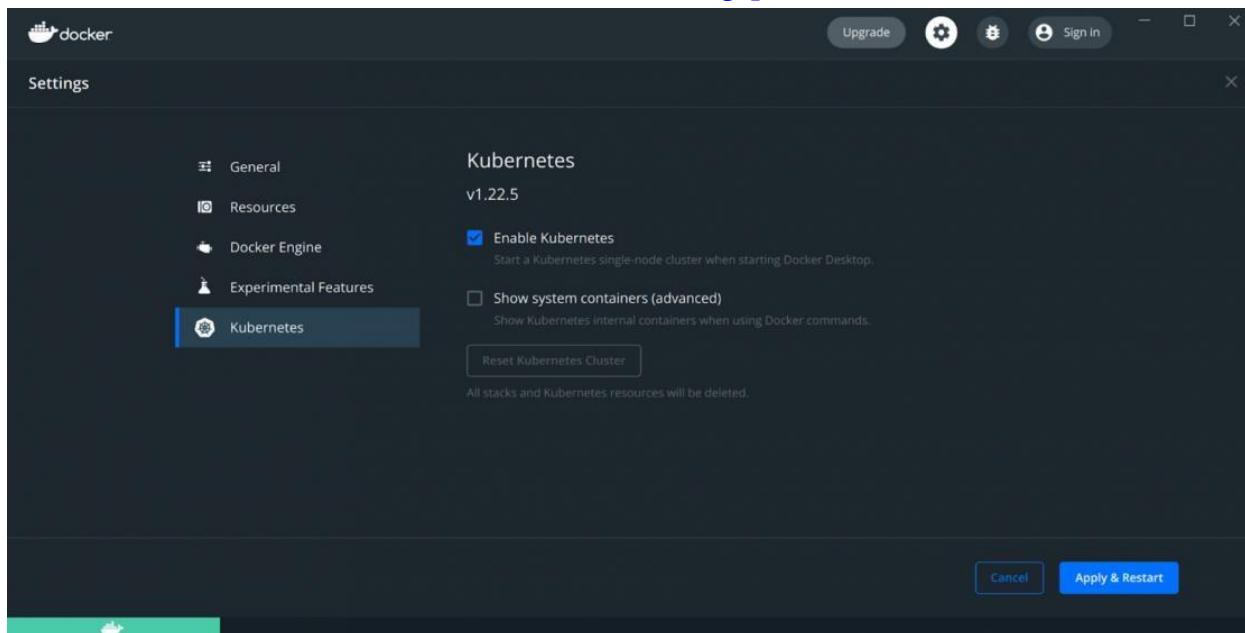
How Does Kubernetes Work with Docker?

You'd have figured it by now that Docker helps to "create" containers whereas K8s enables you to "manage" them at runtime. Docker is used in packing and shipping your application. In the same way, K8s empowers you to deploy and scale your application. Kubernetes comes into the picture only when there are more containers that need to be managed. While the big tech-giants are in a race to adapt to Kubernetes, small startups preferably wouldn't be needing K8s for managing their applications. But, as the companies grow, their infrastructure needs will rise; hence, the number of containers will increase, which can be difficult to manage. Voila! this is where Kubernetes comes in.



Kubernetes setup

Kubernetes can be enabled from the [Kubernetes settings panel](#) shown below.



Dockerfile to Create a HelloWorld Container Image

A manifest, called a Dockerfile, describes how the image and its parts are to run in a container deployed on a host. To make the relationship between the Dockerfile and the image concrete, here's an example of a Dockerfile that creates a "HelloWorld" app from scratch:

```
FROM
scratch
COPY
hello
```

When you give this Dockerfile to a local instance of Docker by using the docker build command, it creates a container image with the "HelloWorld" app installed in it.

Creating a Kubernetes Deployment for HelloWorld

Next you need to define a deployment manifest, commonly done with a YAML or JSON file, to tell

```
# Hello World Deployment
YAMLapiVersion: apps/v1
kind:
Deploymentmeta
ta:
  name:
helloworldspec:
  selector:
    matchLabels:
      app:
helloworldtemplate:
  metadata:
    labels:
      app:
helloworldspec:
  containers:
    -name:helloworldimage:
      boskey/helloworldresourc
es:
  limits:
    memory:
```

Kubernetes how to run "HelloWorld" based on the container image:

To deploy the application on a Kubernetes cluster, you can submit a YAML file using a kubectl command similar to the following.

kubectl apply -f https://yourdomain.ext/application/helloworld.yaml --record

Creating a Kubernetes Service

The container is now deployed to Kubernetes but there is no way to communicate with it, the next step is to turn the deployment into a Service by establishing communication.

In Kubernetes, a Service is an abstraction which defines a logical set of pods and a policy by which to access them. This guide demonstrates a basic method of providing services to pods.

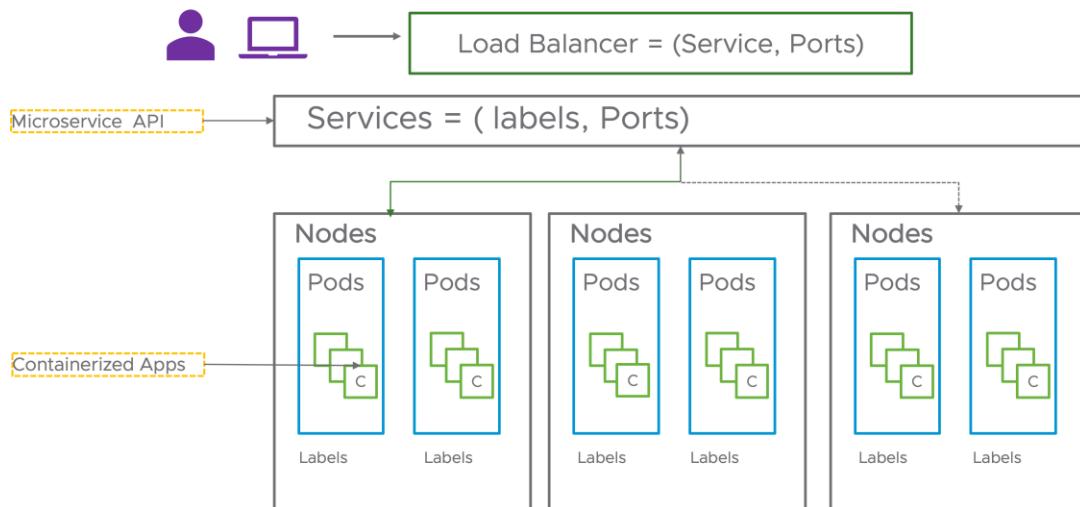
Application Labels and Service

sLabels

A very interesting aspect of Kubernetes is the way Kubernetes combines the use of Labels and Services to create tremendous possibilities.

At the heart of Kubernetes is a pod. A pod contains running instances of one or more containers. When a pod is deployed in Kubernetes, apart from other specifications, the pod can be assigned labels. Ideally a pod is given a label identifying which part of the overall application the pod belongs to. For example, if the pod being deployed is for the application "frontend" and within "frontend" the pod is running code for login, upon deployment it can be labeled [app=frontend, label=login]. Other pods deployed as part of this tier can be given the same label.

Kubernetes Services and Labels



Services

Service Type	Depends on	What it Does	Traffic Type Handled
ClusterIP	ClusterNetwork	Uses the ClusterNetwork to map podIP/port	Internal to the Cluster
NodePort	ClusterIP	Uses a port on Kubernetes Node + creates a mapping of Nodeport to the ClusterIP	External
LoadBalancer	ClusterIP/None	Creates an External Load Balancer that maps to either a Cluster IP/NodePort	External

Services enable Kubernetes to route traffic to pods. Pods in Kubernetes are deployed on

An overlay network. Pods across Kubernetes nodes cannot access each other nor can any external/ingress traffic access pods unless a Service type resource is defined. A service is routed to the correct

Date:

app using a label. So when a service gets created with label login, the service will send traffic to pods that contain the login app based on the label match. Services are needed for both East-West communication, when two pods from different apps need to talk to each other, and for North-South communication, when external traffic (outside of the Kubernetes cluster) needs to talk to a pod. Kubernetes has different service types to address both scenarios. Some common services are listed below:

HelloWorldservice definition

A corresponding service definition for the earlier "Hello World" deployment manifest is shown below. Notice line 5 onward. With the selector as "app: hello world" the service will forward traffic coming to port 80 on the cluster network to pods that match this label.

```
apiVersion: v1
kind: Service
metadata:
  name: helloworld
spec:
  selector:
    app: helloworld
  ports:
    - port: 80
```

Lab09:**Automate the process of running containerized application developed in exercise 7 using Kubernetes.**

To get a Kubernetes Cluster up and running, we will use the following tools:

- Docker Engine: For creating, building, and deploying application images.
- A Docker Hub Account for deploying Docker images to the Docker Hub registry.
- Minikube: Minikube is a Kubernetes tool. It allows you to run Kubernetes locally on your computer. It runs a single-node Kubernetes cluster within your local computer, making it easy to develop the Kubernetes app.
- Kubectl: Kubectl is a Kubernetes command-line (CLI) tool. It allows you to run commands related to deploying, inspecting, and managing cluster resources against Kubernetes.
- Node.js runtime: For creating a Node.js app.

It is essential to ensure the above tools are installed, tools and correctly configured on your computer before deploying your container to the Kubernetes cluster using Docker.

You can check if Node.js is correctly installed using the following command: To check if Docker is installed correctly, run the following:

docker

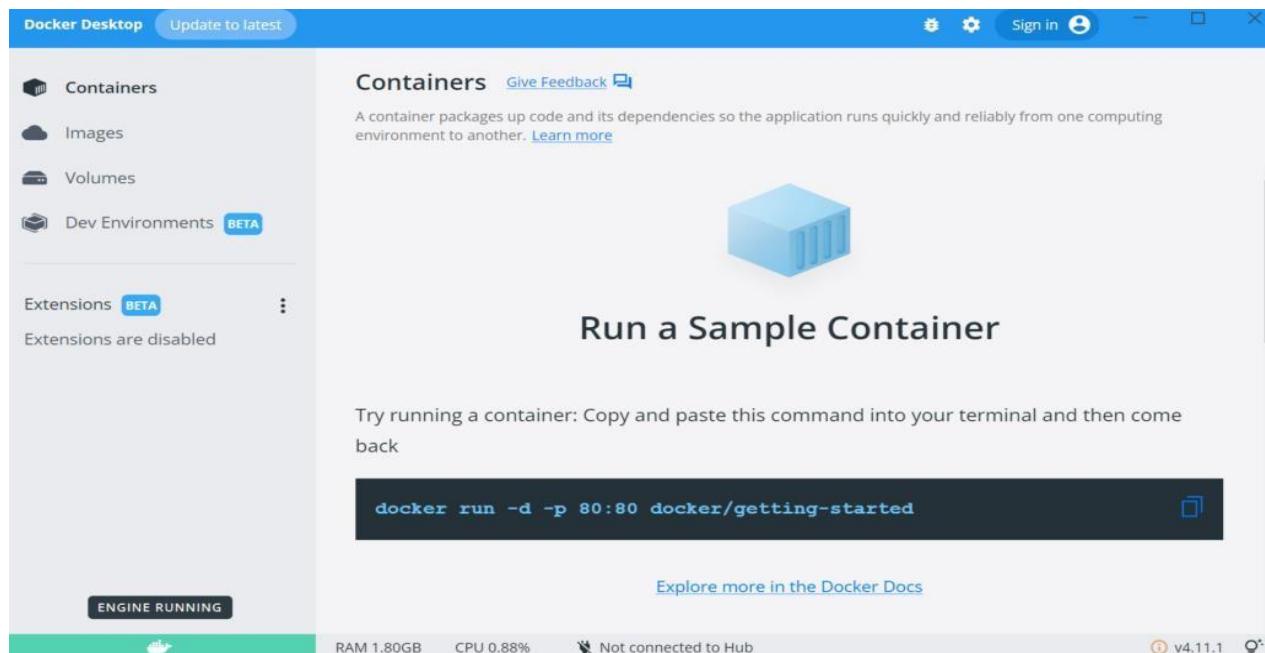
version #client

t

version

#server Docker Desktop

Date:



Once you have installed Minikube on your computer, run the following command to get it up and running within your installed Docker Engine:

`minikube start`

```
* docker "minikube" container is missing, will recreate.
* Creating docker container (CPUs=2, Memory=2200MB) ...
* Preparing Kubernetes v1.24.3 on Docker 20.10.17 ...
  - Generating certificates and keys ...
  - Booting up control plane ...
  - Configuring RBAC rules ...
* Verifying Kubernetes components...
! Executing "docker container inspect minikube --format={{.State.Status}}"
took an unusually long time: 4.3711286s
  - Using image gcr.io/k8s-minikube/storage-provisioner:v5
* Restarting the docker service may improve performance.
  - Using image kubernetesui/metrics-scraper:v1.0.8
  - Using image kubernetesui/dashboard:v2.6.0
* Enabled addons: default-storageclass, storage-provisioner, dashboard
* Done! kubectl is now configured to use "minikube" cluster and "default" namespace by default
```

Once you have the PATH ready, run the following command to check if your set Kubectl is ready to execute Kubernetes commands:

`kubectl cluster-info`

```
C:\Users\kim>kubectl cluster-info
Kubernetes control plane is running at https://127.0.0.1:8547
CoreDNS is running at https://127.0.0.1:8547/api/v1/namespaces/kube-system
/services/kube-dns:dns/proxy

To further debug and diagnose cluster problems, use 'kubectl cluster-info
dump'.
```

Kubernetes allows you to manage large applications. Likewise, Docker can deploy this application to a Kubernetes cluster seamlessly. For this guide, we will just create a simple Node.js app to demonstrate how you can get your application running on a Kubernetes cluster.

If you have an application reader, you can skip this step and use the application alongside this guide.

First, navigate to your project directory and

initialize Node.js using the following command: `npm init -y`

Just like any application, you go ahead and add dependencies/libraries to run your application. In this case, we create a minimal Node.js app. Therefore, we will use the Express library to create a simple Node.js server. Install Express as follows:

`npm install express`

Then create an `index.js` file and create the server as

follows: `const express =`

```
require('express'); const app = express();
const port = 3000; app.get('/', (req, res) => {
  res.send('Hello Kubernetes!!!');
});
app.listen(port, () => {
  console.log(`Server started on port ${port}`);
});
```

To test the application by running the following:
node index.js

Finally, open <http://localhost:3000> on the browser to test the application:

Hello Kubernetes!!!

To get the application ready, we will first write a few commands that will instruct Docker to build an image for this specific application.

Note: Each application

Dockerfile differs. Therefore, always ensure you have the right Dockerfile for running your application. In this example, we will write the Dockerfile as follows: Inside the project directory, create a Dockerfile file and add the following Docker commands.

```
# pull the Node.js from the Docker
registry FROM node:18-alpine3.15
# set default dir so that subsequent commands execute in /src/app
WORKDIR /src/app
# copy dependencies
files COPY package.json ./COPY package-lock.json.
# will execute npm install
in /src/app because of WORKDIR
RUN npm install
# copy application files (index.js)
COPY ./.
# Port to expose the application on
Docker EXPOSE 3000
# The command for running the application
CMD ["node", "index.js"]
```

Once you have the Dockerfile ready, let's dive in and run the application using Docker. Here we will:

- Build the application image
- Allows us to test if the application is correctly running on Docker.
- Build a global registry image for Docker Hub

Let's now dive in and learn how to build the container image using the Dockerfile we have created.

First, let's build the image and test it locally. Inside your Dockerfile root directory, run the following command:

docker build -t node-app .

```
C:\Users\kim>kubectl cluster-info
Kubernetes control plane is running at https://127.0.0.1:8547
CoreDNS is running at https://127.0.0.1:8547/api/v1/namespaces/kube-system
/services/kube-dns:dns/proxy

To further debug and diagnose cluster problems, use 'kubectl cluster-info
dump'.
```

Now run a container from this image and see that it works as expected: **docker run -it -p 3000:3000 node-app**. If you open <http://localhost:3000>, you should be served the same application we run locally. However, this time the application will be loaded from a Docker container.

Hello Kubernetes!!!

To deploy this application to Kubernetes, we first need to push the image to a Docker Hub registry. To do so, log into your Docker Hub account using a terminal by running:

docker login

Provide your correct Docker Hub username and password. Proceed and build an image using your username as follows:

docker build -t <Enter your Docker Hub username>/node-app .

Here, ensure you have the correct username added to the above command. This will form the image name of your application. For example:

docker build -t kimafro/node-app .

Date:

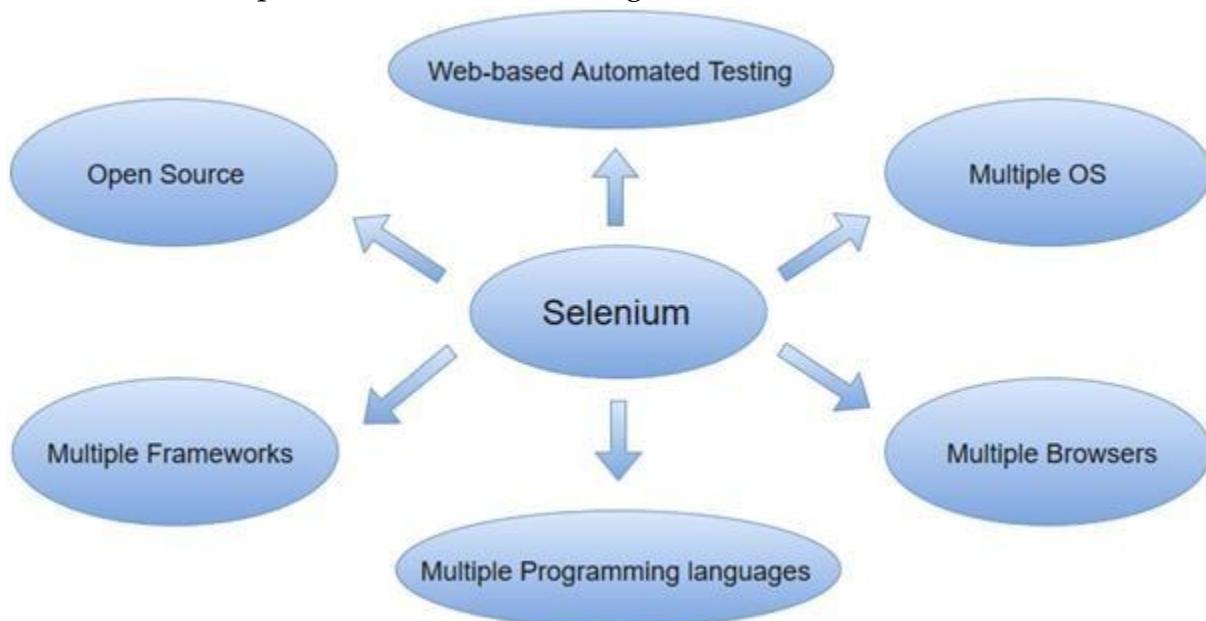
```
PS E:\Docker cluster> docker build -t kimafro/node-app .
[+] Building 2.3s (2/3)
=> [internal] load build definition from Dockerfile
[+] Building 2.6s (2/3)
=> [internal] load build definition from Dockerfile
[+] Building 5.5s (11/11) FINISHED
=> [internal] load build definition from Dockerfile
=> => transferring dockerfile: 32B
=> [internal] load .dockerignore
=> => transferring context: 34B
=> [internal] load metadata for docker.io/library/node:18-alpine3.15
=> [1/6] FROM docker.io/library/node:18-alpine3.15@sha256:13ebaacc762e1753763d031613efaa5aed77f74cab089ef2b57
=> [internal] load build context
=> => transferring context: 5.34kB
=> CACHED [3/6] COPY package.json ./
=> CACHED [4/6] COPY package-lock.json .
=> [6/6] COPY ./ ./ 
=> => exporting to image
=> => writing image sha256:83071df26e101b75237b8cc9260c340f34379de6fc7d6dcf9f8c27954fec8b31
=> => naming to docker.io/kimafro/node-app

PS E:\Docker cluster> docker push kimafro/node-app:latest
[2022-10-17T11:41:16.143372300Z][docker-credential-desktop][W]
nnot find the file specified.
59ac79eb7605: Pushed
e9c06cc4a8fe: Pushed
9cdf0ba12940: Pushed
2d0511c67e79: Pushed
f62c2bc83ebf: Mounted from library/node
a602394ab7c2: Mounted from library/node
latest: digest: sha256:6e32f3326949a55adccc7b5962621321b04081e
```

Lab10:**InstallandExploreSeleniumforautomatedtesting.**

Selenium is one of the most widely used open source Web UI (User Interface) automation testing suite. It was originally developed by Jason Huggins in 2004 as an internal tool at Thought Works. Selenium supports automation across different browsers, platforms and programming languages.

Selenium supports a variety of programming languages through the use of drivers specific to each language. Languages supported by Selenium include C#, Java, Perl, PHP, Python and Ruby. Currently, Selenium Web driver is most popular with Java and C#. Selenium test scripts can be coded in any of the supported programming languages and can be run directly in most modern web browsers. Browsers supported by Selenium include Internet Explorer, Mozilla Firefox, Google Chrome and Safari.

**Selenium Features**

- Selenium is an open source and portable Web testing framework.
- Selenium IDE provides a playback and record feature for authoring tests without the need to learn a test scripting language.
- It can be considered as the leading cloud-based testing platform which helps testers to record their actions and export them as a reusable script with a simple-to-understand and easy-to-use interface.
- Selenium supports various operating systems, browsers and programming languages. Following is the list:

Date:

- Programming Languages: C#, Java, Python, PHP, Ruby, Perl, and JavaScript
- Operating Systems: Android, iOS, Windows, Linux, Mac, Solaris.
- Browsers: Google Chrome, Mozilla Firefox, Internet Explorer, Edge, Opera, Safari, etc.
- It also supports parallel test execution which reduces time and increases the efficiency of tests.
- Selenium can be integrated with frameworks like Ant and Maven for source code compilation.
- Selenium can also be integrated with testing frameworks like TestNG for application testing and generating reports.
- Selenium requires fewer resources as compared to other automation test tools.
- WebDriver API has been indulged in selenium which is one of the most important modifications done to selenium.
- Selenium webdriver does not require server installation, test scripts interact directly with the browser.
- Selenium commands are categorized in terms of different classes which make it easier to understand and implement.
- Selenium Remote Control (RC) in conjunction with WebDriver API is known as Selenium 2.0. This version was built to support the vibrant web pages and Ajax.

Selenium IDE is available only as Firefox and Chrome plug-in, we assume that you have already installed Mozilla Firefox browser in your system. However, you can download the latest version of Firefox through their official website provided under the link given below.

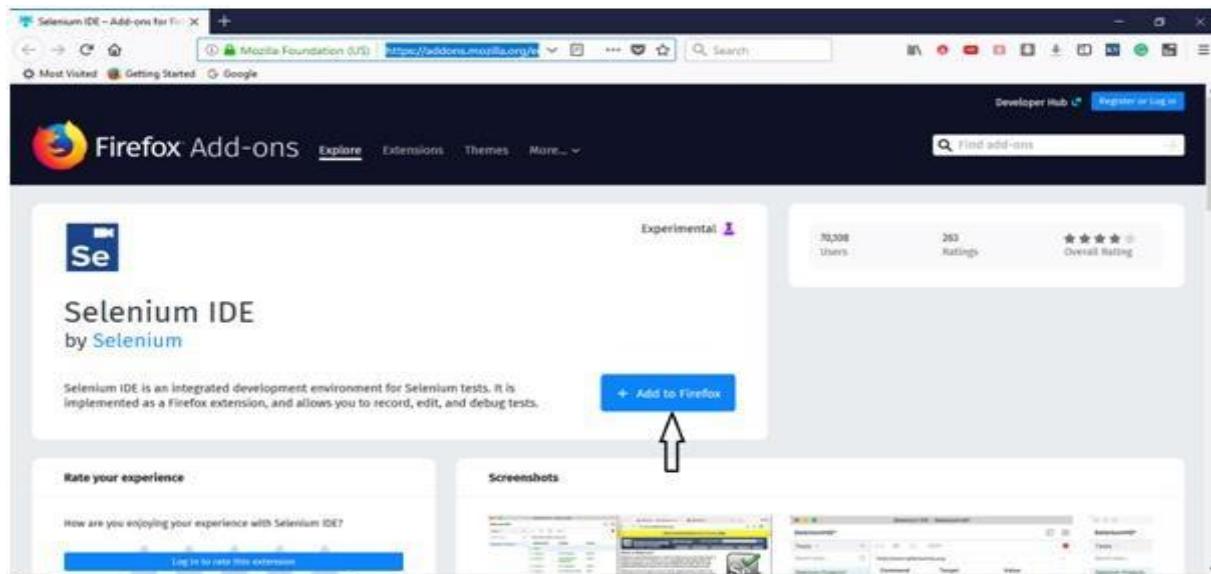
<https://www.mozilla.org/en-US/firefox/new/>

Selenium IDE

Download and Install Launch Mozilla Firefox browser.

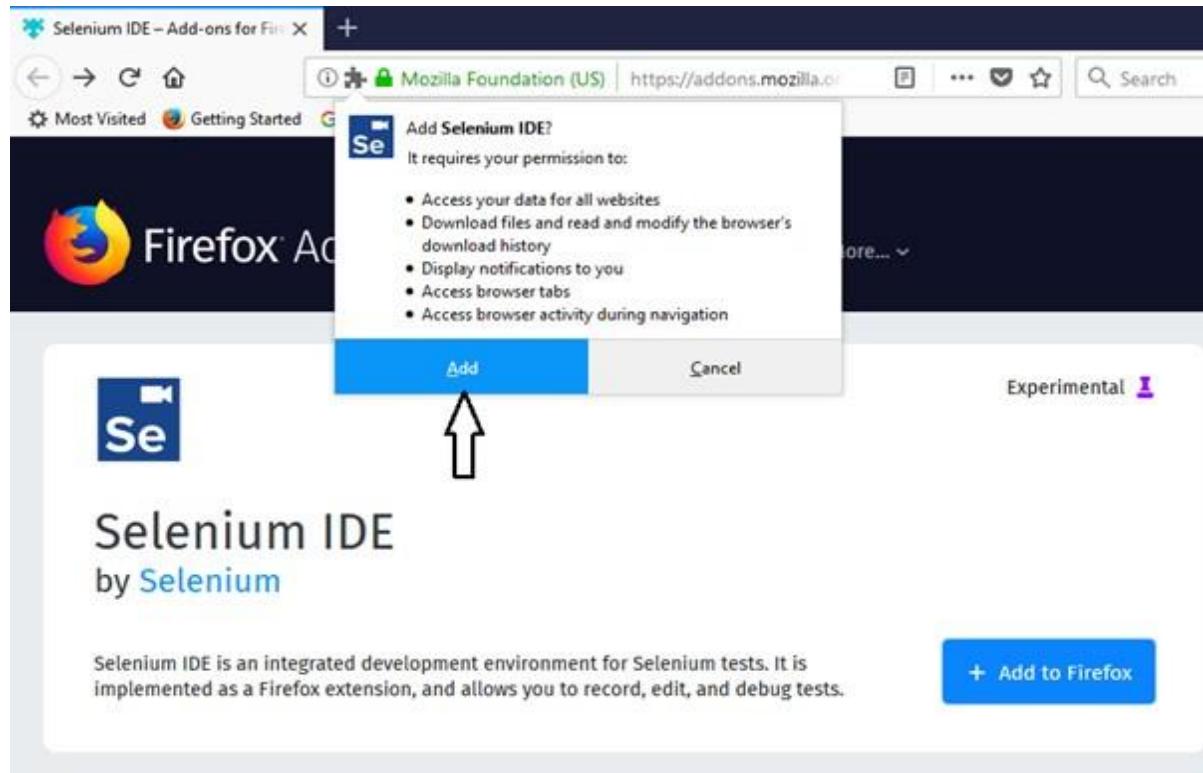
Open URL <https://addons.mozilla.org/en-us/firefox/addon/selenium-ide/> It will redirect you to the official add-on page of Firefox.

Click on "Add to Firefox" button.



A pop-up dialog box will appear asking you to add Selenium IDE as an extension to your Firefox browser.

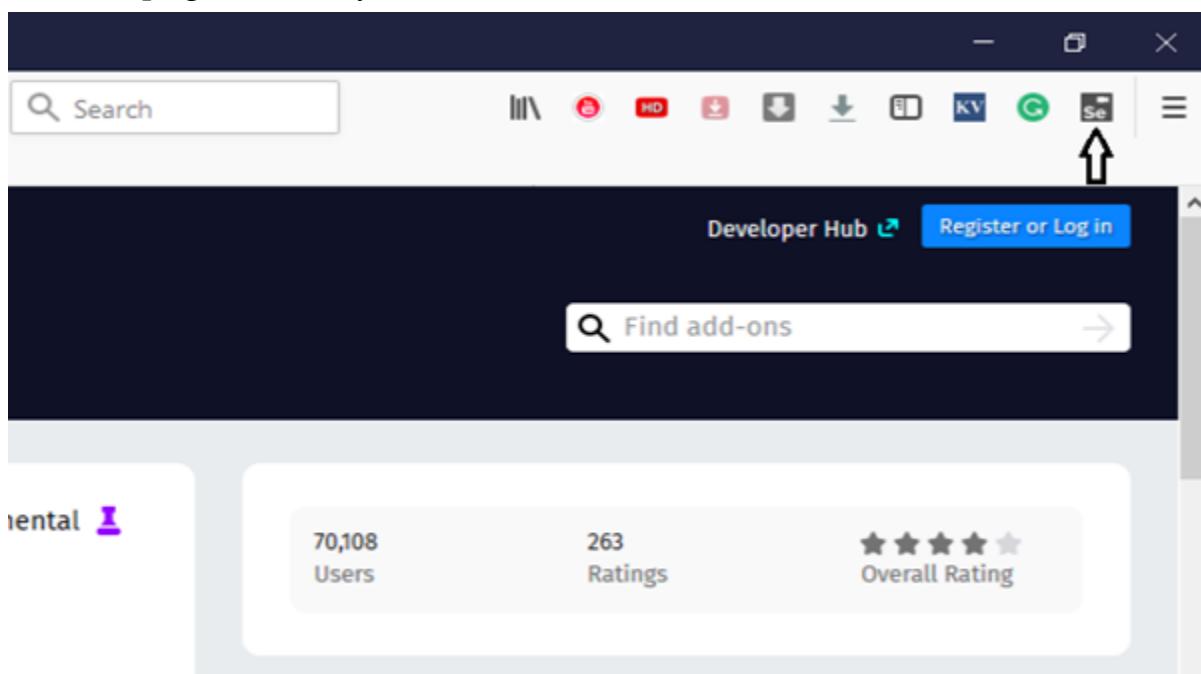
Click on "Add" button.



Restart your Firefox browser.

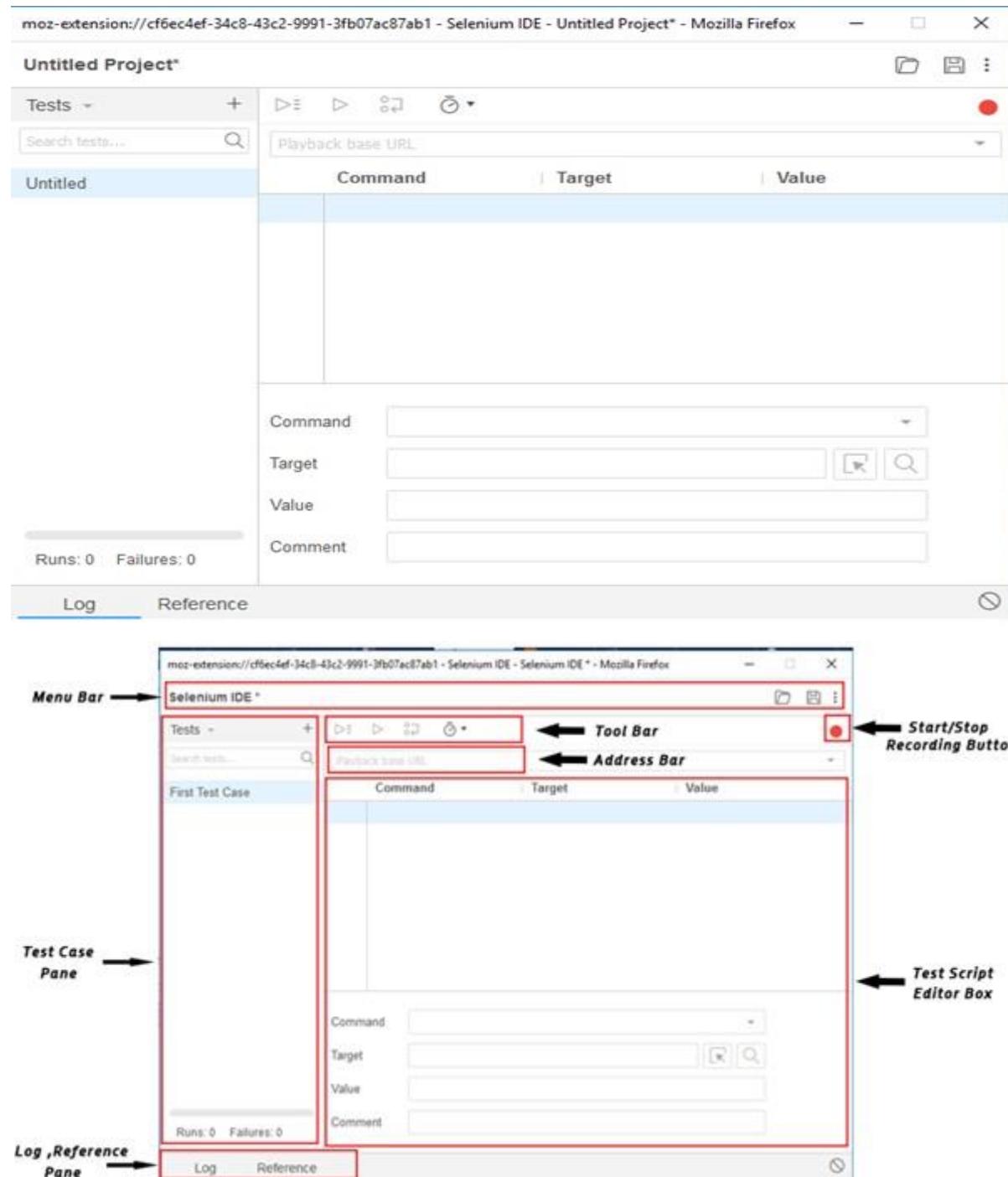
Date:

Go to the top right corner of your Firefox browser and look for the Selenium IDE icon.



Click on that icon to launch Selenium IDE.

Date:



Date:

moz-extension://cf6ec4ef-34c8-43c2-9991-3fb07ac87ab1 · Selenium IDE - Demo Test · Mozilla Firefox

Demo Test*

Tests - +

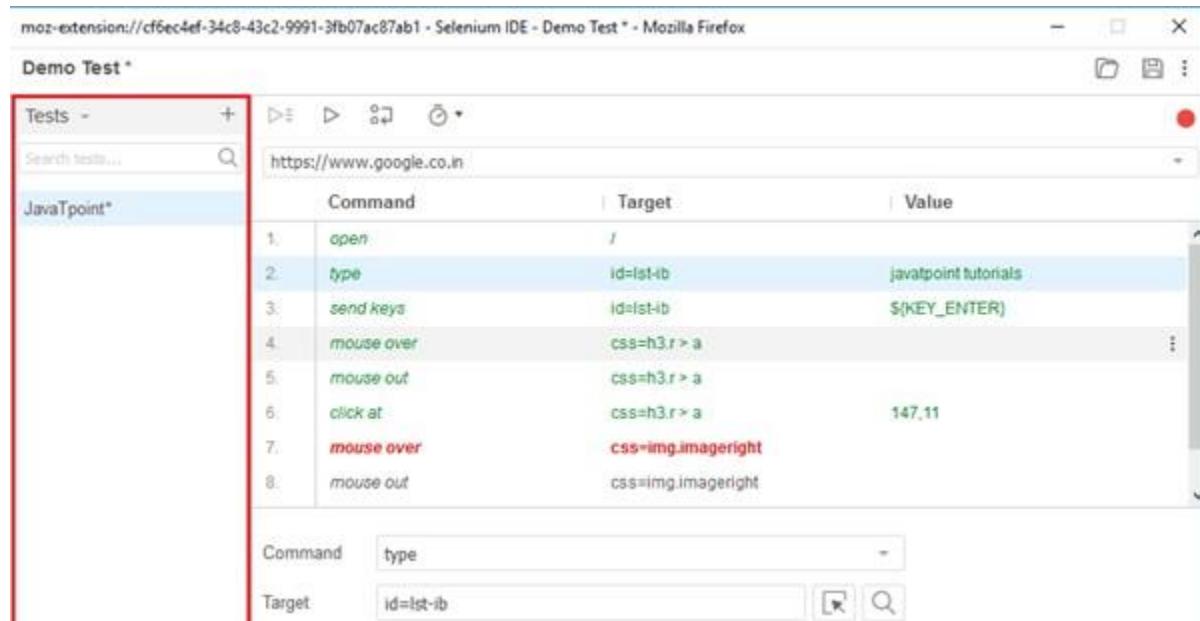
Search tests...

JavaPoint*

	Command	Target	Value
1.	open	/	
2.	type	id=lst-ib	javatpoint tutorials
3.	send keys	id=lst-ib	\$(KEY_ENTER)
4.	mouse over	css=h3.r > a	
5.	mouse out	css=h3.r > a	
6.	click at	css=h3.r > a	147,11
7.	mouse over	css=img.imageright	
8.	mouse out	css=img.imageright	

Command: type

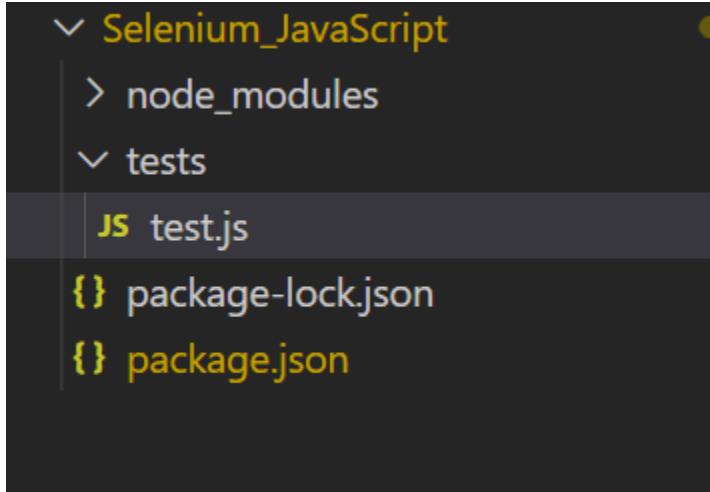
Target: id=lst-ib



Lab11:**Write a simple program in JavaScript and perform testing using Selenium.**

Step1: Create a folder named "tests" in the project root. Inside this folder, we create our first test file named "test.js".

Shown below is the folder structure:



Step2: Inside test.js, pull all the required functions

```
from node_modules const {By, Key, Builder} = require("selenium-
webdriver"); require("chromedriver");
```

Step3: Create a function example(), which will be containing your test script. Feel free to provide any function name of your choice.

```
async function example(){
}
```

Step 4: Inside the function, let us write our test script using Selenium and JavaScript.

```
var searchString="Automation testing with Selenium and JavaScript";
// To wait for browser to build and launch properly
let driver=await new Builder().forBrowser("chrome").build();
// To fetch http://google.com from the browser without
code.await driver.get("http://google.com");
// To send a search query bypassing the value in searchString.
await driver.findElement(By.name("q")).sendKeys(searchString,Key.RETURN);
// Verify the page title and
```

```

print
itvar.title=awaitdriver.getTitle
();console.log('Titleis:',title);

//Itisalwaysasafepracticetoquitthebrowserafterexecutio
nawaitdriver.quit();

}

```

Step5: Now simply add the function call.

Here is how the overall implementation looks like: const{By,Key,Builder}=require("selenium-webdriver");

```
require("chromedriver")
```

```
;asyncfunctionexample(
```

```
){
```

```
varsearchString="AutomationtestingwithSelenium";
```

```
//Towaitforbrowsertobuildandlaunchproperly
```

```
letdriver=awaitnewBuilder().forBrowser("chrome").build();
```

```
//Tofetchhttp://google.comfromthebrowser without
```

```
code.awaitdriver.get("http://google.com");
```

```
//TosendasearchquerybypassingthevalueinsearchString.
```

```
awaitdriver.findElement(By.name("q")).sendKeys(searchString,Key.RETURN);
```

```
//Verify the page title and
```

```
print
```

```
itvar.title=awaitdriver.getTitle
```

```
();console.log('Titleis:',title);
```

```
//Itisalwaysasafepracticetoquitthebrowserafterexecutio
```

```
nawaitdriver.quit();
```

```
}
```

```
example()
```

Step6: Time for some action... let's run the code by giving the following command on the terminal:
 nodetest.js

An instance of Chrome browser is instantiated and the necessary [Selenium WebDriver actions](#) are performed on the WebElements.

Date:

Lab12:**Develop test cases for the above containerized application using selenium.****Step1: Pull the docker image**

To get a list of all the already existing images on the system, run the following command in the command prompt:

```
docker images
```

```
C:\>docker images
REPOSITORY      TAG      IMAGE ID      CREATED      SIZE
debian          latest   f776cfb21b5e  4 months ago  124MB
centos          latest   5d0da3dc9764  5 months ago  231MB
busybox          latest   6d5fcfe5ff17  2 years ago  1.22MB
C:\>
```

If you don't already have the selenium standalone-chrome docker image, run the following command to download a copy of the image onto the system.

```
docker pull selenium/standalone-chrome
```

```
C:\>docker pull selenium/standalone-chrome
Using default tag: latest
latest: Pulling from selenium/standalone-chrome
ea362f368469: Pull complete
ad903aa3d58c: Pull complete
e203573541ba: Downloading [=====] 62.83MB/95.05MB
99389543bb26: Download complete
3ff44411c39f: Download complete
d8f8badb1764: Download complete
5bf49a2cbd1b: Download complete
1dd85dc76a5b: Download complete
e1a2b4715967: Downloading [=====] 42.06MB/96.34MB
2d0a47f345f6: Download complete
ad4902342ec5: Download complete
4354c332bc95: Download complete
3a43cb3085cd: Downloading [=====] 14.22MB/22.75MB
de60e11e5762: Waiting
036ab721e480: Waiting
a8d1acfdc810: Waiting
```

Upon re-running the command docker images, selenium/standalone-chrome image appears in the list.

```
C:\>docker images
REPOSITORY      TAG      IMAGE ID      CREATED      SIZE
selenium/standalone-chrome  latest   d0b6caec4485  9 days ago  1.2GB
debian          latest   f776cfb21b5e  4 months ago  124MB
centos          latest   5d0da3dc9764  5 months ago  231MB
busybox          latest   6d5fcfe5ff17  2 years ago  1.22MB
```

Step2: Running the Selenium Webdriver Docker container

Upon pulling the selenium/standalone-

chromeimageontothesystem,startthecontainerbyrunningthefollowingcommand:

```
dockerrun-d -p 4444:4444 -v /dev/shm:/dev/shm selenium/standalone-chrome
```

```
C:\Users\bshrikanth>docker run -d -p 4444:4444 selenium/standalone-chrome
837f375519d620fb2f53f70ded60eaabf87756e5819757a736df1aca62fe1bf4
```

StartingDockerContainer

Thecommand,whenrun,willreturntheContainerID.

Openthebrowserandnavigatetohttp://localhost:4444/.ItreflectsSeleniumGridUI,asshown below.



Step3:Creatingasampletestfile

SeleniumsupportstestswrittenindifferentlanguagesofwhichJavaandPythonaremostpopularlyused.Inthis example,usingPythontocreate SeleniumTest.

```
fromseleniumimportwebdriver
```

Toopenfileinchromebrowser,Chromedriverisnecessary.Therefore,initializingChromeOptions to declare any connection and driver options necessary while instantiating abrowserinstance.

```
options =
webdriver.ChromeOptions()options.add_argument(
"--ignore-ssl-errors=yes")
```

CreatinganinstanceoftheRemotewebdriverandpassingtheseleniumendpointandchromeoptionsdefinedintheprevious step.

```
driver =
webdriver.Remote(command_executor='http://localhost
:4444/wd/hub',options=options
)
```

Date:

avagatetotheBrowserStackwebsiteandclickontheGetStartedfor
Freebutton.Inducingwaittime betweenthe twoactions allowsviewingtheexecutionoftests.

```
driver.get("https://www.browserstack.com/")

driver.find_element_by_link_text("Getstartedfree").click()
```

seleniumDockerTest.py

```
from selenium import

webdriverimporttime

print("Test Execution

Started")options=

webdriver.ChromeOptions()

options.add_argument('--ignore-ssl-
errors=yes')options.add_argument('--ignore-
certificate-errors')driver =

webdriver.Remote(command_executor='http://localhost
:4444/wd/hub',options=options

)

#maximize the window

size=driver.maximize_window

time.sleep(10)

#navigate to

browserstack.comdriver.get("https://www.bro
wserstack.com/")time.sleep(10)
```

```
driver.find_element_by_link_text("Getstartedfree").click()time.sleep(10)

#close the

browser.close()

e()driver.quit()

print("TestExecutionSuccessfullyCompleted!")
```

Step4: Executing the test case

Python test case file can be run using either an IDE or command prompt. To run it from the command prompt, open a command prompt and run the following command:

```
python<filename>
```

