

# Lab Manual

## for

### DevOps LAB

### (CM605PC)

**III B. TECH II SEMESTER (R20 AUTONOMOUS)**



**DEPARTMENT OF CSE**

**(ARTIFICIAL INTELLIGENCE AND MACHINE LEARNING)**



**ACE**  
**Engineering College**  
**An Autonomous Institution**



Ghatkesar, Hyderabad - 501 301, Telangana.

Approved by AICTE & Affiliated to JNTUH

NBA Accredited B.Tech Courses, Accorded NACC A-Grade with 3.20 CGPA

## DEVOPS LAB

B.Tech. III Year II Semester							
Course Code	Category	Hours/Week			Credits	Maximum Marks	
		L	T	P	C	CIA	SEE
Contact Classes: 45	Tutorial Classes: 15	Practical Classes: Nil			Total Classes: 60		

**Course Objectives:**

1. Describe the agile relationship between development and IT operations.
2. Understand the skill sets and high-functioning teams involved in
3. DevOps and related methods to reach a continuous delivery capability
4. Implement automated system update and DevOps lifecycle

**List of Experiments:**

1. Write code for a simple user registration form for an event.
2. Explore Git and GitHub commands.
3. Practice Source code management on GitHub. Experiment with the source code written in exercise 1.
4. Jenkins installation and setup, explore the environment.
5. Demonstrate continuous integration and development using Jenkins.
6. Explore Docker commands for content management.
7. Develop a simple containerized application using Docker.
8. Integrate Kubernetes and Docker
9. Automate the process of running containerized application developed in exercise 7 using Kubernetes.
10. Install and Explore Selenium for automated testing.
11. Write a simple program in JavaScript and perform testing using Selenium.
12. Develop test cases for the above containerized application using selenium.

**Text Books:**

1. Joakim Verona. Practical Devops, Second Edition. Ingram short title; 2nd edition (2018). ISBN-10: 1788392574
2. Deepak Gaikwad, Viral Thakkar. DevOps Tools from Practitioner's Viewpoint. Wiley publications. ISBN: 9788126579952

**REFERENCE BOOK:**

1. Len Bass, Ingo Weber, Liming Zhu. DevOps: A Software Architect's Perspective. AddisonWesley
2. Edureka DevOps Full Course - [https://youtu.be/S\\_0q75eD8Yc](https://youtu.be/S_0q75eD8Yc)

**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.**

<b>GitEnvironmentSetup-GitConfig command</b>	<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>
<b>Git init command</b>	<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 &lt;file&gt;..." 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 'Git-Example'. At the top, there's a summary bar with 2 commits, 1 branch, 0 releases, and 1 contributor. Below this, 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 the URL, there are options to 'Open in Desktop' or '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



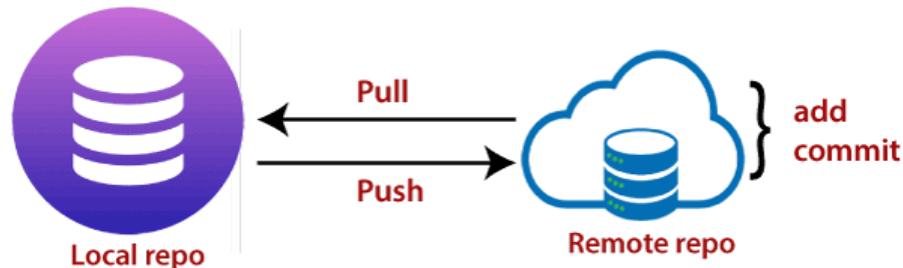
```
Shashank@DESKTOP-Shashank MINGW64 ~/Desktop/IIICSM (master)
$ git remote add origin "https://github.com/get002/csm.git"
```

```
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.*



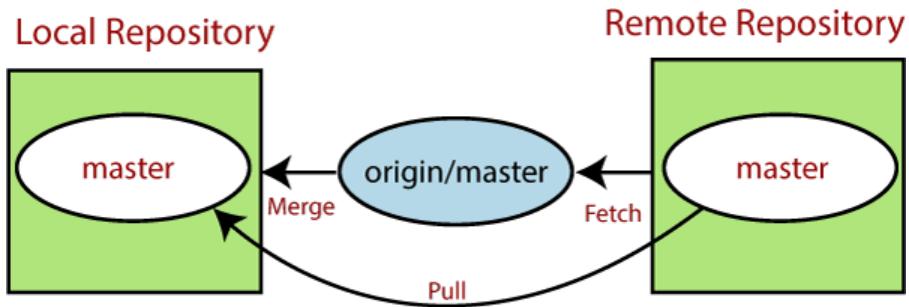
```
Shashank@DESKTOP-shashank MINGW64 ~/Desktop/IIICSM (master)
$ git push origin master
Enumerating objects: 14, done.
Counting objects: 100% (14/14), done.
Delta compression using up to 4 threads
Compressing objects: 100% (8/8), done.
Writing objects: 100% (13/13), 1.27 KiB | 118.00 KiB/s, done.
Total 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 &lt;branch name&gt;</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 &lt;branch name&gt;</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 &lt;branch name&gt;</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(-)

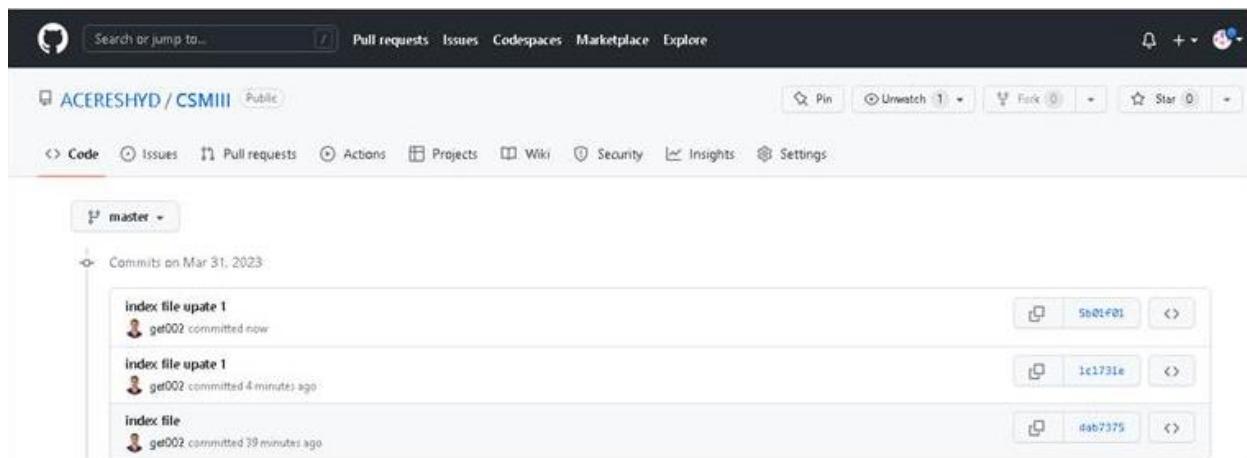
Shashank@DESKTOP-Shashank MINGW64 ~/Desktop/WebProject (master)
$ git push origin master
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 4 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 282 bytes | 94.00 KiB/s, done.
Total 3 (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
```



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

Date:

```
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

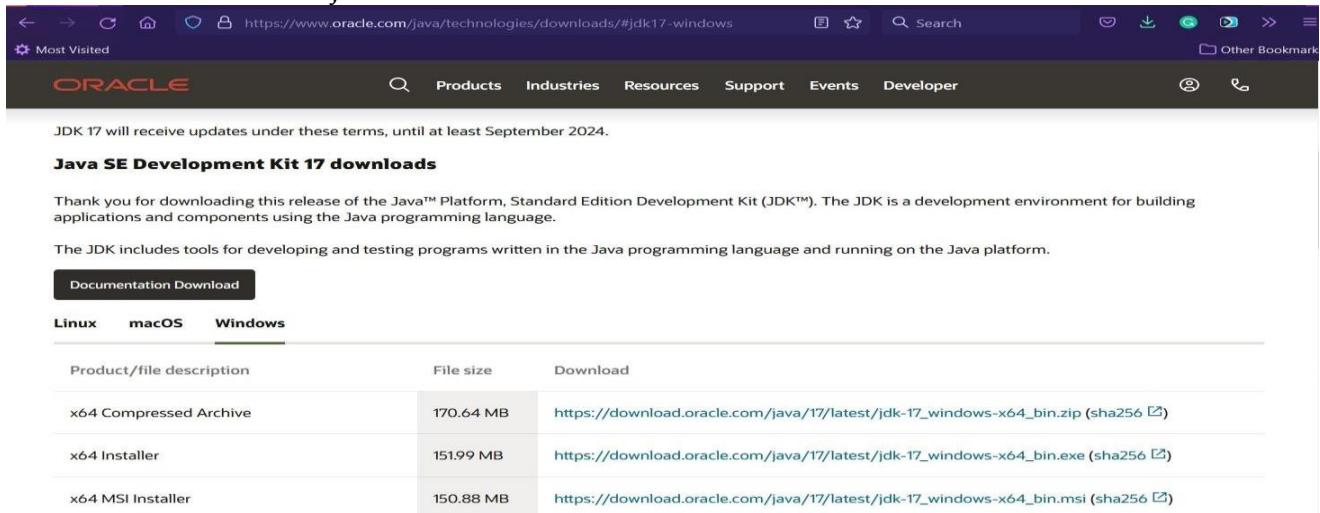
**Lab04:****Jenkins installation and setup, explore the environment.**

- 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 →

### Install Java Version 8

- Since Jenkins is a Java based application, therefore Java is a must.
- Download java8 from the below link:  
<https://www.oracle.com/java/technologies/downloads/#jdk17-windows>
- Then install the Java as follows:



The screenshot shows the Oracle Java SE Development Kit 17 downloads page. It features a navigation bar with links for Products, Industries, Resources, Support, Events, and Developer. Below the navigation bar, a message states "JDK 17 will receive updates under these terms, until at least September 2024." A section titled "Java SE Development Kit 17 downloads" includes a message of thanks for downloading the Java Platform, Standard Edition Development Kit (JDK). It notes that the JDK is a development environment for building applications and components using the Java programming language. A "Documentation Download" button is visible. Below this, there are tabs for "Linux", "macOS", and "Windows". The "Windows" tab is selected, showing a table with three rows of download links:

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

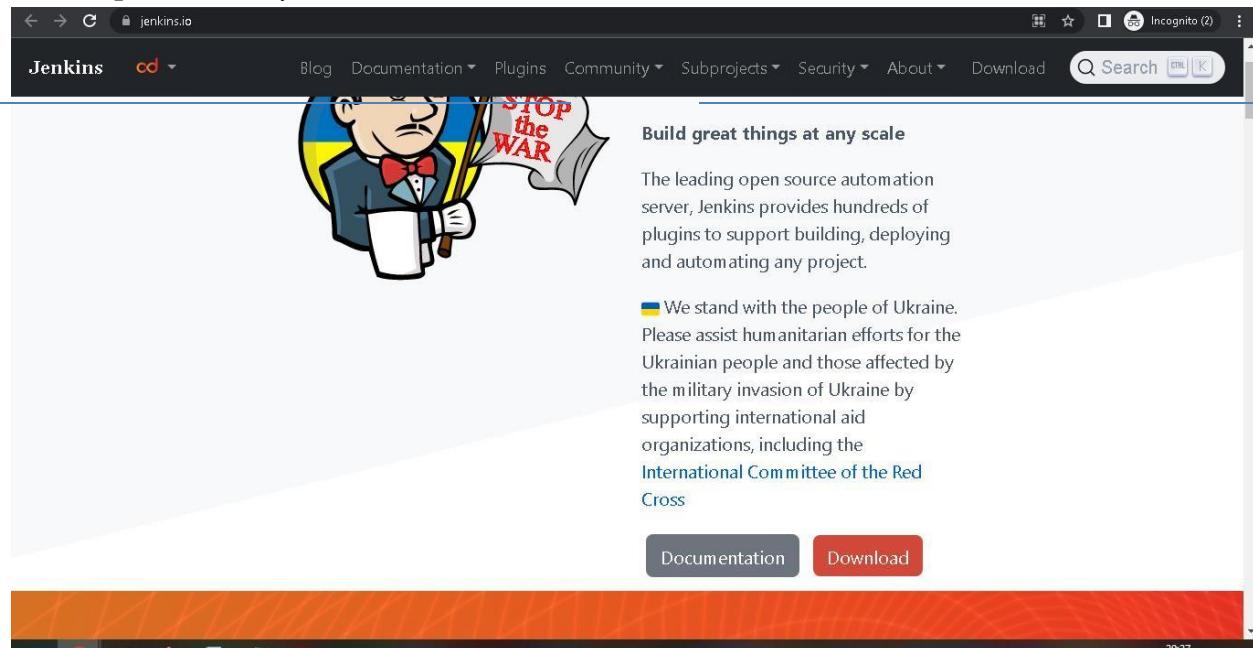
  


The screenshot shows the "Java(TM) SE Development Kit 17 (64-bit) - Setup" window. The title bar displays the Java logo and the text "Java(TM) SE Development Kit 17 (64-bit) - Setup". The main content area says "Welcome to the Installation Wizard for Java SE Development Kit 17". Below it, a message states "This wizard will guide you through the installation process for the Java SE Development Kit 17." At the bottom right, there are "Next >" and "Cancel" buttons.

Date:

## Download Jenkins war File

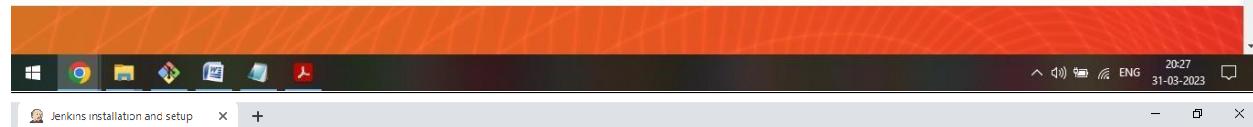
Download from the following link <https://www.jenkins.io/>



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

[Documentation](#) [Download](#)



Jenkins installation and setup

<https://jenkins.io/download/>

Changelog | Upgrade Guide | Past Releases

Deploy Jenkins 2.176.2

Deploy to Azure

Download Jenkins 2.176.2 for:

- Docker
- FreeBSD
- Gentoo
- Mac OS X
- OpenBSD
- openSUSE
- Red Hat/Fedora/CentOS
- Ubuntu/Debian
- Windows
- Generic Java package (.war)

Download Jenkins 2.187 for:

- Arch Linux
- Docker
- FreeBSD
- Gentoo
- Mac OS X
- OpenBSD
- openSUSE
- Red Hat/Fedora/CentOS
- Ubuntu/Debian
- OpenIndiana Hipster
- Windows
- Generic Java package (.war)

## Accessing Jenkins

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

**http://localhost:8080**

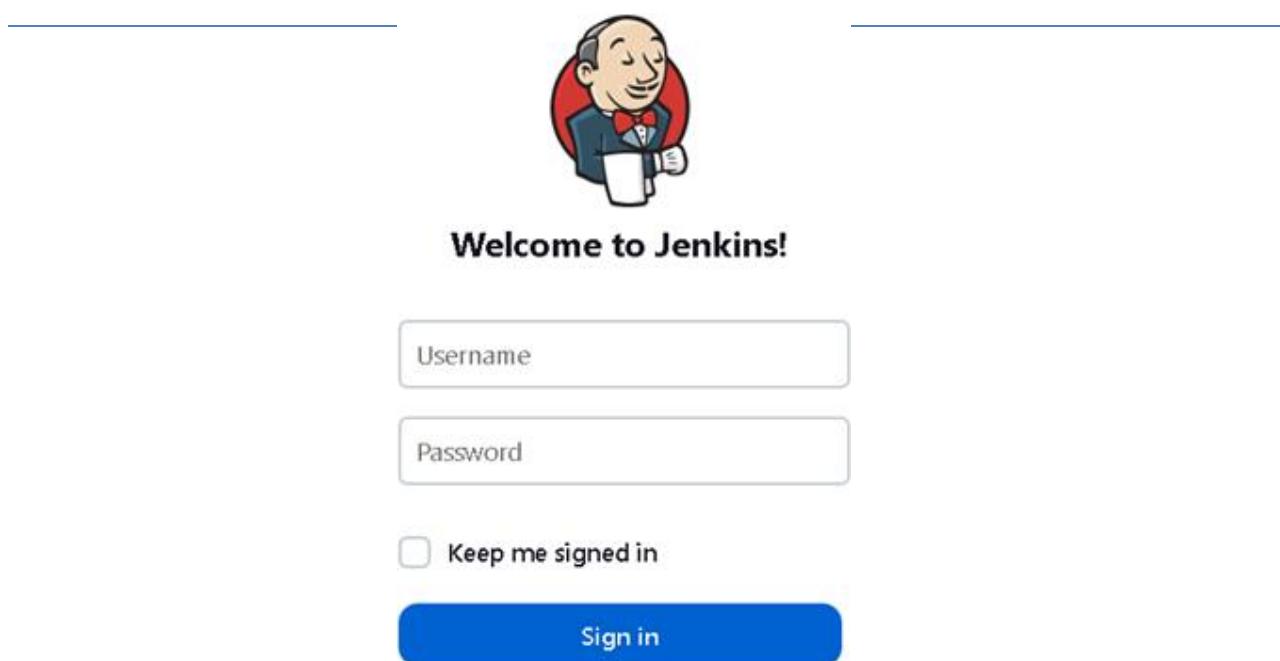
This url will bring up the Jenkins dashboard.

The screenshot shows the Jenkins dashboard with the following elements:

- Header:** Jenkins logo, search bar (Search (CTRL+K)), user info (Shashank Tiwari), and log out link.
- Left Sidebar:**
  - + New Item
  - People
  - Build History
  - Manage Jenkins
  - My Views
- Middle Content:**
  - Welcome to Jenkins!** heading.
  - A message: "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."
  - Start building your software project** button.
  - Build Queue** section: "No builds in the queue."
  - Create a job** button.
  - Build Executor Status** section: "1 Idle" and "2 Idle".
  - Set up a distributed build** and **Set up an agent** buttons.

## Exploring the Jenkin environment

- Log into your jenkin server



The screenshot shows the Jenkins dashboard with several annotations:

- Search bar to search job created on jenkin**: Points to the search bar at the top right.
- People connected on jenkin**: Points to the "People" link in the sidebar.
- Configure & Manage Jenkins server**: Points to the "Manage Jenkins" link in the sidebar.
- To create your personalized view**: Points to the "My Views" link in the sidebar.

The main content area displays a welcome message and a brief description of the dashboard's purpose.

The screenshot shows the "Create a job" dialog with the following details:

- Enter an item name**: A text input field containing "My First Job".
- Required field**: A note indicating the field is mandatory.
- Freestyle project**: Described as the central feature of Jenkins, combining any SCM with any build system.
- Pipeline**: Described as orchestrating long-running activities across multiple build agents.
- Multi-configuration project**: Suitable for projects with many configurations, like testing on multiple environments.
- Folder**: Creates a container for grouping items together.
- Multibranch Pipeline**: Creates a set of Pipeline projects based on detected branches in one SCM repository.
- Organization Folder**: Creates a set of multibranch project subfolders by scanning repositories.

At the bottom right, there are links for "Activate Windows" and "Go to Settings to activate Windows".

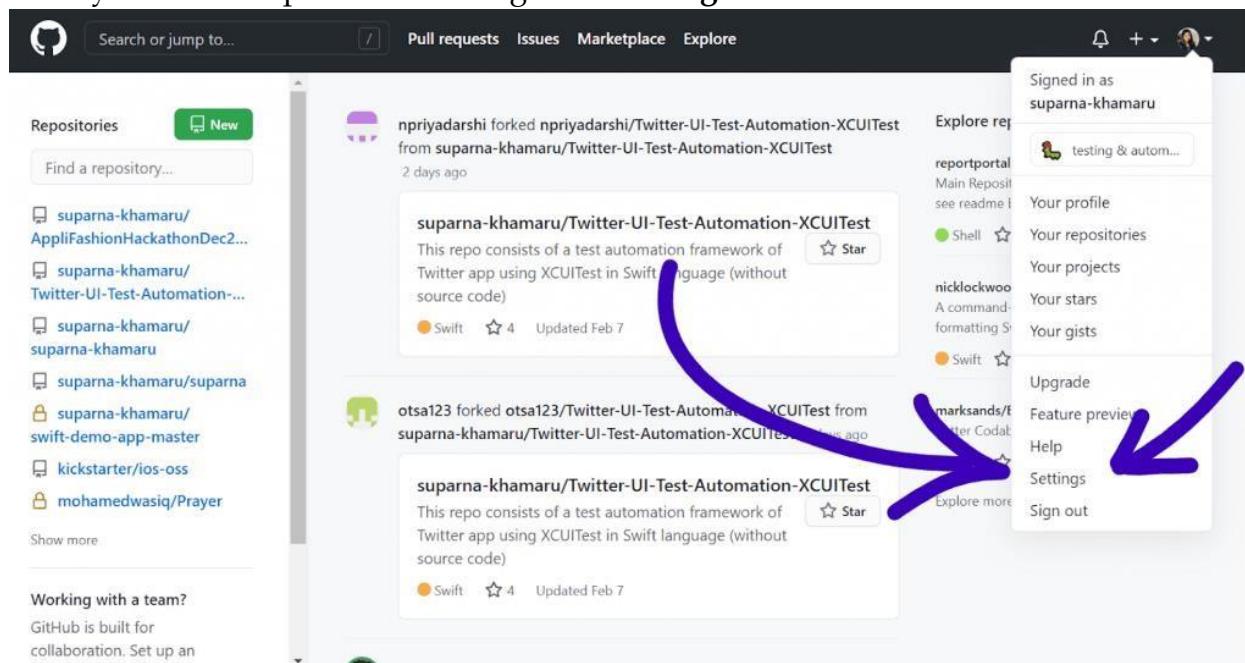
Next, define the parameter with the job

Date:

The image shows two screenshots of the Jenkins web interface. The top screenshot is the 'Configuration' page for a job named 'My First Job'. It includes sections for General (with a description field and a 'Enabled' toggle), Source Code Management (set to 'None'), Build Triggers (with several options like 'Trigger builds remotely' and 'GitHub hook trigger for GITScm polling' checked), and Build Environment. The bottom screenshot is the 'Dashboard' page, showing a summary of the job's status: 'My First Job' (Status: Failed, Last Success: N/A, Last Failure: 38 sec ago, Last Duration: 0.18 sec). It also displays sections for Build Queue (empty), Build Executor Status, and various Jenkins management links like New Item, People, Build History, Manage Jenkins, and My Views.

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

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	
Tokens you have generated that can be used to access the GitHub API.	
jenkins-integration — admin:repo_hook, notifications, repo <small>Last used within the last week</small> <a href="#">Delete</a>	

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.

## Configuring Jenkins for Git Hub Integration

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

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

## Creating Your First Jenkins Job Integrated Into a TestProject

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

The screenshot shows the Jenkins dashboard with a sidebar on the left containing links like 'Dashboard', 'New Item', 'People', etc. The main area has a 'Welcome to Jenkins!' message and sections for 'Start building your software project' and 'Set up a distributed build'.

Enter a project name in the requested textbox field and select the **Freestyle project**.

In the **General** section, check the field "**Git Hub project**" and enter a valid Ant project as given below.

Date:

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.

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:

Date:

Click on the “**Apply**” button and then click on the “**Save**” button.

**Build**

**Invoke Ant**

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

**Post-build Actions**

Add post-build action ▾

**Save** **Apply**

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-handside of the displayed dashboard screen.

**Jenkins**

Dashboard > GithubProject >

**Project GithubProject**

Back to Dashboard

Status

Changes

Workspace

Build Now

Configure

Delete Project

GitHub

Rename

**Build History** trend ^

Build #	Date
#16	02-Jan-2021, 11:35 PM

Recent Changes

Workspace

**Permalinks**

- 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

**Project GithubProject**

- Workspace
- Recent Changes

### Permalinks

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

### 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
18:59:38 > 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] Commit message: "test names demo"
18:59:38 > C:\Program Files\Git\bin\git.exe rev-list --no-walk be@af76702244d39c1e3369a095cd52bb432ae3d # timeout=10
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:**

- Docker is a platform that allows us to package our applications into deployable executables — called containers, with all its necessary OS libraries and dependencies.
- Docker is an open-source software platform.
- It is a containerization platform that allows developers to create, package, run and deploy their applications on any operating system or cloud platform.
- With Docker, developers can quickly and easily create, test, and deploy applications without having to worry about compatibility issues or hardware requirements.

**Docker Container:**

- Docker container is an isolated environment that is running instance of Docker images.

**Docker Image:**

- A Docker image is a lightweight, standalone, and executable package that contains everything needed to run an application, including the code, runtime environment, libraries, and system tools.

**Docker Hub:**

- Docker Hub is a cloud-based repository service provided by Docker for storing and sharing Docker images.
- Docker registry is of 2 type: public and private repositories.
- Public repositories are freely accessible to anyone, while private repositories require authentication and access control.

**Dockerfile:**

- A Dockerfile is a text document that contains all the commands and Docker can build images automatically by reading the instructions from a Dockerfile.

**Docker Commands:****1. docker --version**

→ To get the information of currently installed version of docker

```
C:\Shashank>docker --version
Docker version 26.1.1, build 4cf5afa
```

**2. docker run**

→ Docker run command is used to start a new container based on a Docker image.

→ Docker first check whether image exists on your local system or not. If not, docker will pull the image from the docker registry.

→ Docker will download the image file to your local system and stores it in a local repository.

```
docker run -p 8085:8085 mysql
```

```
C:\Shashank>docker run -p 8085:8085 mysql
Unable to find image 'mysql:latest' locally
latest: Pulling from library/mysql
d9a40b27c30f: Downloading [=====] 16.25MB/48.99MB
d948328c7651: Download complete
c1e267313ede: Download complete
7478f013875a: Downloading [=====] 4.451MB/6.712MB
9221a2250289: Download complete
d1f57baa52d5: Download complete
35c3d30e8624: Waiting
```

### 3. docker images

- Docker images command will list all of the images that are currently stored on your local system

```
C:\Shashank>docker images
REPOSITORY      TAG          IMAGE ID      CREATED        SIZE
mysql           latest       5cde95de907d  9 days ago    586M
tomcat          jre21       214192ee3a5a  2 weeks ago   299M
nginx           latest       e0c9858e10ed  2 weeks ago   188M
docker           latest       1acd5db2357a  5 weeks ago   366M
postgres         alpine3.20  f1bf85e5d935  5 weeks ago   242M
nginx           stable-alpine-perl  27f43866f4ac  6 weeks ago   85.4M
mysql           oraclelinux9  fcd86ff8ce8c  2 months ago  578M
hello-world     latest       d2c94e258dcba  14 months ago  13.5M
```

### 4. docker rmi imageName

- Removes/ un-tags images from the host machine (not from a registry). You cannot remove an image of a running container.

```
C:\Shashank>docker rmi hello-world
Untagged: hello-world:latest
Untagged: hello-world@sha256:94323f3e5e09a8b9515d74337010375a456c909543e1ff1538f5
Deleted: sha256:d2c94e258dcba3c5ac2798d32e1249e42ef01cba4841c2234249495f87264ac5a
→ Deleted: sha256:ac28800ec8bb38d5c35b49d45a6ac4777544941199075dff8c4eb63e093aa81e
```

### 5. docker container ls

- Lists all running containers on your system

C:\Shashank>docker container ls						
CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS	NAMES

### 6. docker container ls -a

- shows all the running and exited containers

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS
	NAMES			
9e9bac7a9eef	mysql	"docker-entrypoint.s..."	3 minutes ago	Exited (1) 3 minutes ago
e759b4555768	postgres:alpine3.20	"docker-entrypoint.s..."	6 minutes ago	Exited (1) 5 minutes ago
b629d0884dd9	nginx	/docker-entrypoint..."	13 days ago	Exited (0) 13 days ago
18e1c885a2b4	mysql:oraclelinux9	"docker-entrypoint.s..."	2 weeks ago	Exited (1) 2 weeks ago
04894a0a7d55	nginx:stable-alpine-perl	/docker-entrypoint..."	2 weeks ago	Exited (255) 13 days ago
0.0.0:8086->8086/tcp	kind_saha			
26427b3c58bf	nginx:stable-alpine-perl	/docker-entrypoint..."	4 weeks ago	Exited (0) 4 weeks ago
9247cb958ae8	postgres:alpine3.20	"docker-entrypoint.s..."	4 weeks ago	Exited (1) 4 weeks ago
	nifty_feynman			

### 7. docker image pull <image name> →

*fetches and downloads an image from the docker registry/docker Hub without starting a container. docker Hub contains many pre-built images.*

```
C:\Shashank>docker image pull hello-world
Using default tag: latest
latest: Pulling from library/hello-world
c1ec31eb5944: Pull complete
Digest: sha256:1408fec50309afee38f3535383f5b09419e6dc0925bc69891e79d84cc4cdcec6
Status: Downloaded newer image for hello-world:latest
docker.io/library/hello-world:latest
```

## 8. docker images

- Lists all the images that are locally stored with the Docker engine.

C:\Shashank>docker images				
REPOSITORY	TAG	IMAGE ID	CREATED	SIZE
mysql	latest	5cde95de907d	9 days ago	58M
tomcat	jre21	214192ee3a5a	2 weeks ago	29M
nginx	latest	e0c9858e10ed	2 weeks ago	18M
docker	latest	1acd5db2357a	5 weeks ago	36M
postgres	alpine3.20	f1bf85e5d935	5 weeks ago	24M
nginx	stable-alpine-perl	27f43866f4ac	6 weeks ago	85M
mysql	oraclelinux9	fcd86ff8ce8c	2 months ago	57G
hello-world	latest	d2c94e258dcf	14 months ago	13M

→

## 9. docker start [container\_id]

- Starts a docker container that has been stopped.

```
C:\Shashank>docker start b629d0884dd9
b629d0884dd9
```

→

## 10. docker restart [container\_id]

- Restarts a container that's running or stopped.

```
C:\Shashank>docker restart b629d0884dd9
b629d0884dd9
```

→

## 11. docker rm <container id>

- remove the Docker containers manually.

```
C:\Shashank>docker rm 9e9bac7a9eef
9e9bac7a9eef
```

→

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

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

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

Creating the Docker file 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 Docker file directory (.means current directory).

```
D:\Projects\ Docker\sample> docker build -t hello-docker .
[+] Building 44.8s (8/8) FINISHED
```

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

**Date:**

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

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

docker run hello-docker

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

**Lab 08:****Integrate Kubernetes and Docker.****What is Kubernetes?**

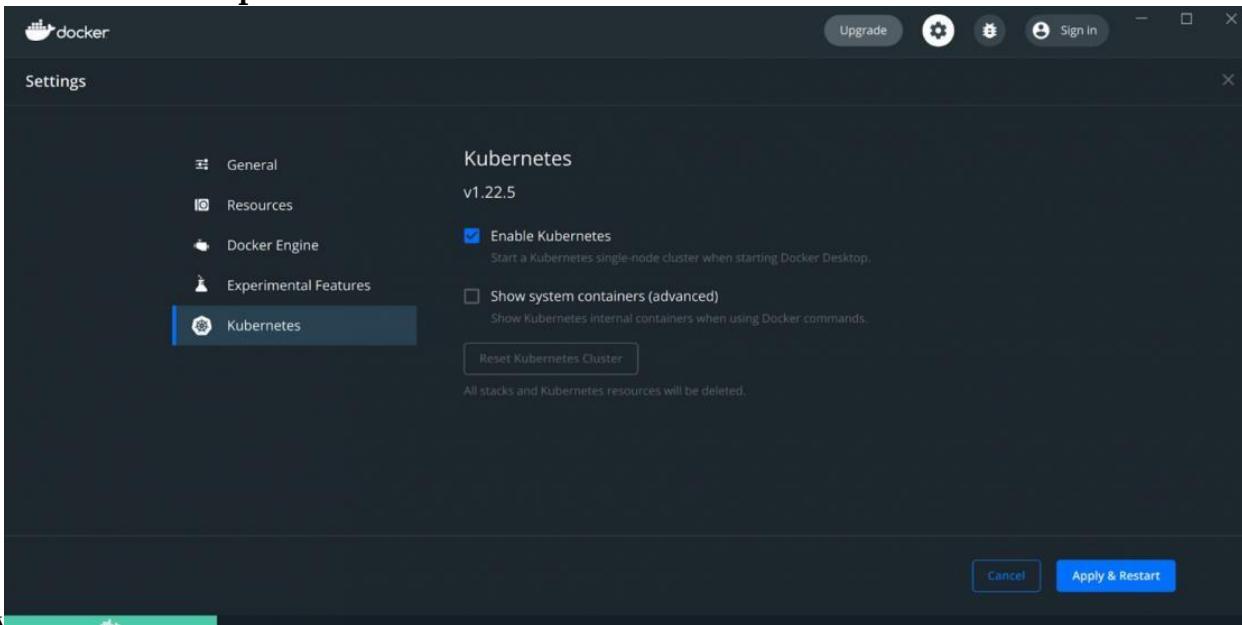
*Kubernetes* is an open-source orchestration tool that allows you to run and manage your container based workloads. [Kubernetes](#) K8s was Developed by google and was later donated to the Cloud Native Computing Foundation. Kubernetes helps you manage hundreds and thousands of containerized applications in different deployable environments be it physical machines, virtual machines, cloud 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 ship 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 the 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.

**Kubernetes setup →**

## 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
scratchCOPY
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

Define a deployment manifest, commonly done with a YAML or JSON file, to tell

```
# Hello World Deployment YAML
apiVersion: apps/v1
kind: Deployment
metadata:
  name: helloworld
spec:
  selector:
    matchLabels:
      app: helloworld
  template:
    metadata:
      labels:
        app: helloworld
    spec:
      containers:
        - name: helloworld
          image: boskey/helloworld
          resources:
            limits:
              memory:
                "128Mi"
              cpu:
                "500"
```

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 Command similar to the following.

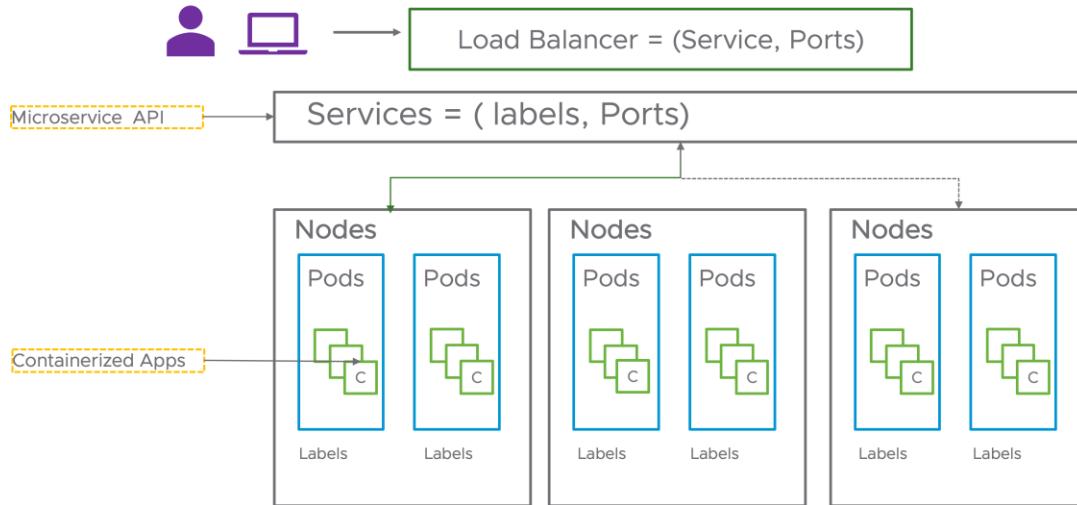
Kubectl apply -f <https://yourdomain.ext/application/helloworld.yaml> –record

## 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.

### Kubernetes Services and Labels



### Hello World service 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.

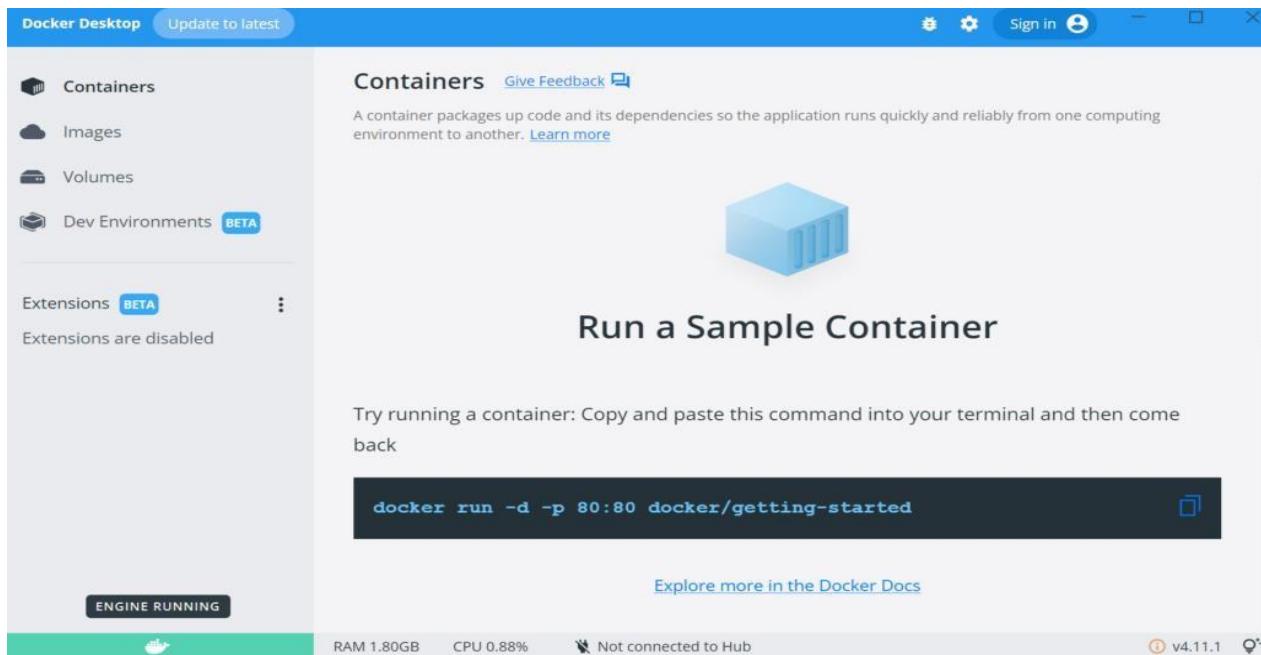
```

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:

- DockerEngine: For creating, building, and deploying application images.
- A Docker Hub Account for deploying Docker images to the Docker Hub registry.
- Minikube: Minikube is Kubernetes tool. It allows you to run Kubernetes locally on your computer. It runs as a single-node Kubernetes cluster with in 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 aNode.js app.



Date:

```
* 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. Like wise, 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 along side 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}`);
});
```

Then test the application by running the following:

node index.js

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

Hello Kubernetes!!!

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 DockerHub 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**

```
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 ./ ./
=> => 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
```

## **EXPERIMENT NO.: 10.**

Install and Explore Selenium for automated testing

**AIM:** To install the necessary software and explore Selenium WebDriver for automated web testing. This includes setting up the Java Development Kit (JDK), Selenium WebDriver, browser drivers, and an Integrated Development Environment (IDE), and then writing and executing a basic test script.

### **DESCRIPTION:**

- Selenium is a widely-used open-source framework for automating web browsers. It provides a suite of tools for automating web applications for testing purposes but is also used for web scraping and automating repetitive tasks.
- Selenium supports a variety of programming languages through the use of language-specific bindings or drivers. The languages supported by Selenium include java, C#, python, Ruby, Javascript, perl.
- Selenium test scripts can be written in any of the supported programming languages and executed in modern web browsers.
- Selenium supports a wide range of web browsers, including:
  - **Google Chrome**
  - **Mozilla Firefox**
  - **Internet Explorer**
  - **Microsoft Edge**
  - **Safari**

A simple example demonstrating how to use Selenium WebDriver with Java to open Google's homepage.

**Step1:** Install JDK and set up environmental variables.

**Step2:** Install selenium web driver

Selenium WebDriver is a set of APIs that allow you to control web browsers.

Go to <https://www.selenium.dev/downloads/> and download java driver under selenium clients and webdriver language bindings. Extract the zip file and place in C:\Drivers\

**Step3:** Install browser driver.

Each supported browser has a corresponding driver that Selenium WebDriver uses to communicate with the browser. For example:

- ChromeDriver: For Google Chrome
- GeckoDriver: For Mozilla Firefox
- IEDriverServer: For Internet Explorer
- SafariDriver: For Safari.

For chrome driver, check the google chrome version and go to <https://googlechromelabs.github.io/chrome-for-testing/>. Download the stable release, Extract the zip file and place in C:\Drivers\selenium-java-<version>\

**Step4:** Create a simple test application test.java

```

import org.openqa.selenium.WebDriver;
import org.openqa.selenium.chrome.ChromeDriver;

public class Test {
    public static void main(String[] args) {
        // Set the path to the ChromeDriver executable
        System.setProperty("webdriver.chrome.driver", "C:\\\\Drivers\\\\selenium-java-4.23.0\\\\chromedriver-win64\\\\chromedriver.exe");

        // Initialize the ChromeDriver
        WebDriver driver = new ChromeDriver();

        // Open a webpage
        driver.get("https://www.google.com");

        // Print the title of the page
        System.out.println("Page title is: " + driver.getTitle());

        // Close the browser
        driver.quit();
    }
}

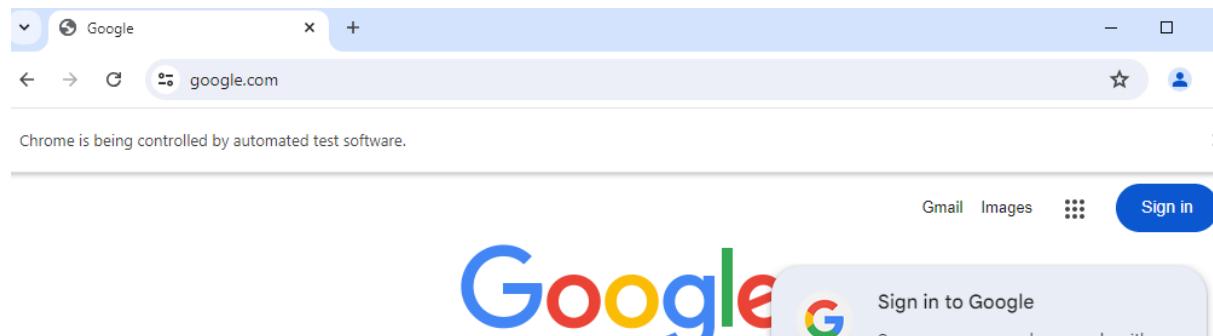
```

**Step5:** Compile and run the program with the following commands

```

C:\Users\kappa\testing>javac -cp "C:\Drivers\selenium-java-4.23.0\*" Test.java
C:\Users\kappa\testing>java -cp "C:\Drivers\selenium-java-4.23.0\*;." Test
Page title is: Google

```



### EXPERIMENT NO.: 11.

Write a simple program in JavaScript and perform testing using Selenium

**AIM:** To write a selenium test case for a simple JavaScript program that increments a counter on a webpage. The JavaScript program adds a counter functionality to a button. The goal is to test this functionality using Selenium.

#### Description:

**Step1: Write JavaScript Code:** Develop a web application with JavaScript that performs certain tasks (e.g., incrementing a counter).

**Step2: Write Selenium Test in Java:** Use Selenium to write a test in Java that interacts with the web page, triggering JavaScript actions and verifying outcomes.

**Step3:** Download the following jar files and place them in c:\Drivers

- **JUnit:** Defines and runs test cases in Java, checking that the Selenium WebDriver interactions produce the expected outcomes. Install Junit jar and place it in c:\Drivers
- **Hamcrest:** Provides expressive assertion methods for verifying conditions in your tests. Install Hamcrest jar and place it in c:\Drivers

#### counter.html

```
<!DOCTYPE html>
<html>
<head>
    <title>Simple JavaScript Program</title>
</head>
<body>
    <p id="output">0</p>
    <button id="increment-button">Increment</button>
    <script>
        const output = document.getElementById("output");
        const incrementButton = document.getElementById("increment-button");
        let count = 0;
        incrementButton.addEventListener("click", function() {
            count += 1;
            output.innerHTML = count;
        });
    </script>
</body>
</html>
```

#### Main.java

```
import org.openqa.selenium.By;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.chrome.ChromeDriver;
import org.junit.After;
import org.junit.Before;
import org.junit.Test;

public class Main {
    private WebDriver driver;

    @Before
    public void setUp() {
        System.setProperty("webdriver.chrome.driver", "C:\\Drivers\\selenium-java-4.23.0\\chromedriver-win64\\chromedriver.exe");
        driver = new ChromeDriver();
    }

    @Test
    public void testIncrementButton() throws InterruptedException {
        driver.get("file:///C:/Users/sample/counter.html");
        driver.findElement(By.id("increment-button")).click();
        String result = driver.findElement(By.id("output")).getText();
        assertEquals(result.equals("1") : "Expected output to be 1, but got " + result);
        Thread.sleep(5000);
    }

    @After
    public void tearDown() {
        driver.quit();
    }
}
```

**Step4: compile and run the program.**

```
C:\Users\sample>javac -cp "C:\Drivers\selenium-java-4.23.0\*" Main.java

C:\Users\sample>java -cp "C:\Drivers\selenium-java-4.23.0\*;." org.junit.runner.JUnitCore Main
JUnit version 4.13.2
.
Time: 10.976

OK (1 test)
```



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 do not have the selenium stand alone -chrome docker image, run the following command to download a copy of the image on to 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
e1a2b47f345f6: Download complete
ad4902342ec5: Download complete
4354c332bc95: Download complete
3a43cb3085cd: 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 rerunning 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 Web driver Docker container**

Upon pulling the selenium/standalone-chrome image on to the system, start the container

**Date:**

by running the following command:

```
docker run -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
```

Starting Docker Container

The command, when run, will return the Container ID.

Open the browser and navigate to <http://localhost:4444/>. It reflects Selenium Grid UI, as shown below.



Step3: Creating a sample test file

Selenium supports tests written in different languages of which Java and Python are most popularly used. In this example, using Python to create Selenium Test.

```
from selenium import webdriver
```

To open file in chrome browser, Chrome driver is necessary. Therefore, initializing Chrome Options to declare any connection and driver options necessary while instantiating a browser instance.

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

Creating an instance of the Remote webdriver and passing the selenium end point and chrome options defined in the previous step.

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

**Date:**

Navigate to the Browser Stack web site and click on the Get Started for Free button.  
Inducing waittime between the two actions allows viewing the execution of tests.

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

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

```
seleniumDockerTest.py
from selenium import webdriver

import time


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()

ime.sleep(10)

#navigate to browserstack.com

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

time.sleep(10)

#clickontheGetstartedforfreebutton
```

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

#close the browser

driver.close()driv
er.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>
```

