

Ex.No: 1(a) INTRODUCTION TO CLOUD COMPUTING(Google Drive)

Date:

Aim:

To Upload, Share and download user files in cloud environments like Google Drive.

Concept:

- “Cloud” is short for “cloud computing,” and it refers to tasks and services provided or hosted via the internet on a pay-as-we-go basis.
- The cloud is a collection of servers and data centers scattered across the globe that store data.
- Essentially, it’s a digital storage unit where we can keep all our files. With the cloud, we can access our data from any device so long as it has an internet connection.

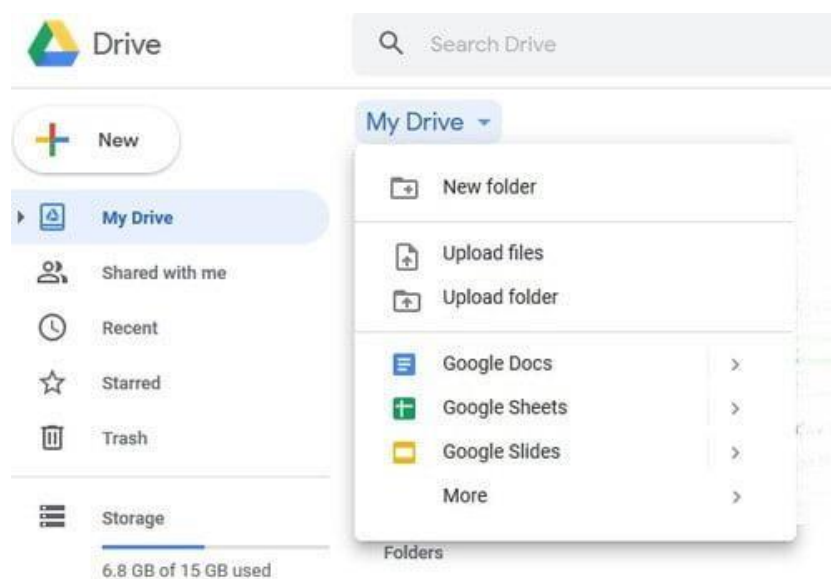
Process/Procedure:

- Go to Google Drive
- Upload or create files to google drive
- Share and organize files or folders, so other people can view, edit, or comment on them.

Experiment:

Step 1: Go to drive.google.com, On our computer, go to drive.google.com. We’ll see "My Drive," which has:

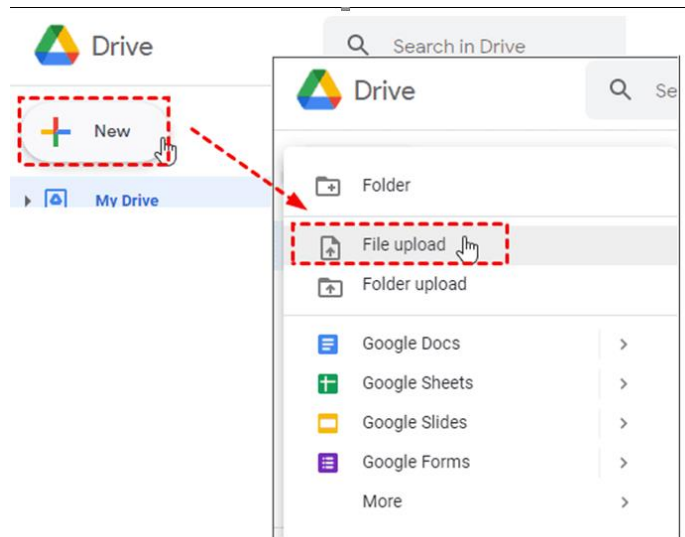
- Files and folders we upload or sync
- Google Docs, Sheets, Slides, and Forms we create



Learn How to Backup and sync files from mac or PC

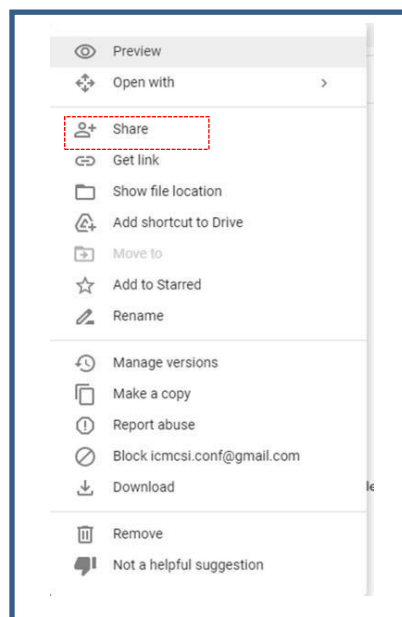
Step2: Upload or create files, We can upload files from our computer or create files in Google Drive.

- Upload files and folders to Google Drive
- Work with Office files
- Create, edit, and format Google Docs, Sheets, and Slides



Step 3: Share and organize files, We can share files or folders, so other people can view, edit, or comment on them.

- Share files from Google Drive
- Share folders from Google Drive
- Make someone else the owner of a file



Result:

The above steps are used for uploading, sharing and downloading files using google drive cloud environments.

Ex.No: 1(b) INTRODUCTION TO CLOUD COMPUTING(Dropbox)

Date:

Aim:

To Upload, Share and download user files in cloud environment like Dropbox

Concept:

In Dropbox, any business initiative, succeeding in sales enablement means crafting a solid strategy and executing on it. Much has been written on sales enablement strategy, and the field is constantly evolving as technology, buyers' habits, and other factors change over time.

Process/Procedure:

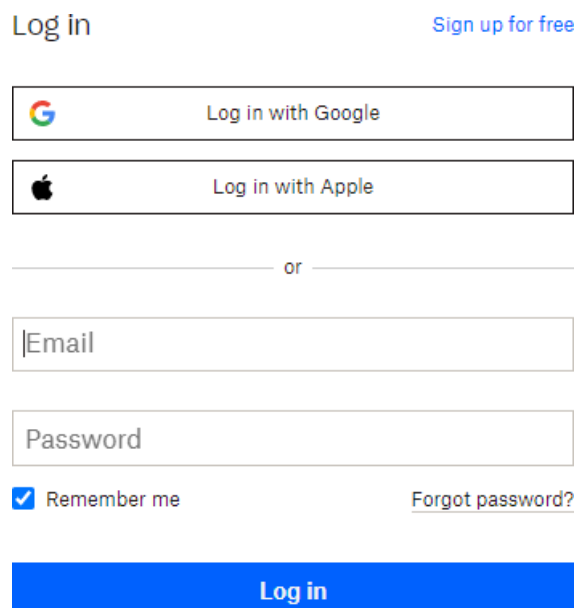
- Create DropBox Account by filling required details
- Upload Files - Navigate to "Upload Files/Folder" , Select the File , Select the file we want to upload and tap "open."
- Share a File or Folder - Locate the File in Our Dropbox Folder, Generate a Share Link , Copy the Link
- Download Files - Locate the File , Select the File and Download

Experiment:

Step 1: To begin using Dropbox, we'll first need to create an account. Thankfully, it only takes a couple of minutes to do, and we can follow the steps below to get started.

1. Locate the "Sign in" Prompt on the website

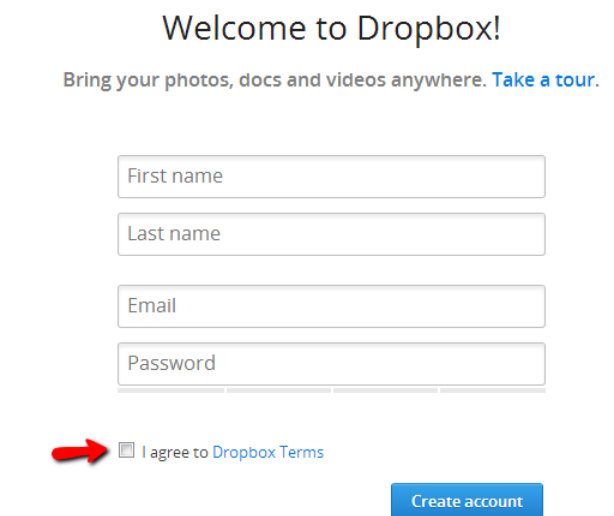
Once we have reached the Dropbox website, select "sign in" in the top right-hand corner.



The image shows the Dropbox login interface. At the top, there are links for "Log in" and "Sign up for free". Below these are two large buttons for social login: "Log in with Google" (featuring the Google logo) and "Log in with Apple" (featuring the Apple logo). A horizontal line with the word "or" in the center separates these from the standard login fields. There are two input fields: one for "Email" and one for "Password". Below the password field is a checkbox labeled "Remember me" which is checked, and a link for "Forgot password?". At the bottom is a large blue button labeled "Log in".

2. Select “Create an Account”

To create a new account, select the “create an account” option.



The image shows the Dropbox account creation page. At the top, it says "Welcome to Dropbox!" followed by "Bring your photos, docs and videos anywhere. [Take a tour.](#)". Below this are four input fields: "First name", "Last name", "Email", and "Password". At the bottom, there is a checkbox with a red arrow pointing to it, labeled "I agree to [Dropbox Terms](#)". To the right of the checkbox is a blue button labeled "Create account".

3. Enter our personal details

We’ll be prompted to **enter our name, email address and password**. (If we want to keep our passwords secure, check out our pick of the best password managers available.) For a slightly quicker process, we can also sign up to Dropbox using our Google account.

4. Select our Plan

On the next page, we will be asked to **select the plan** we would like to use. If we prefer to stick to the free plan, select the option “or continue with 2GB Dropbox Basic plan” located at the bottom of the screen.

Dropbox will send we email confirmation of our account. We will need to verify our email address before we can use the service.

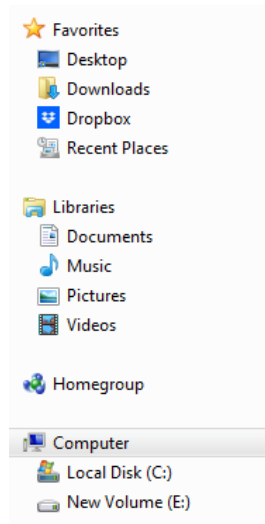
Once the sign-up process is complete, we will be prompted to download Dropbox to our computer. It’s a good idea to do this if we want to automatically sync files from our computer to the service.

Use the Dropbox folder on Desktop:

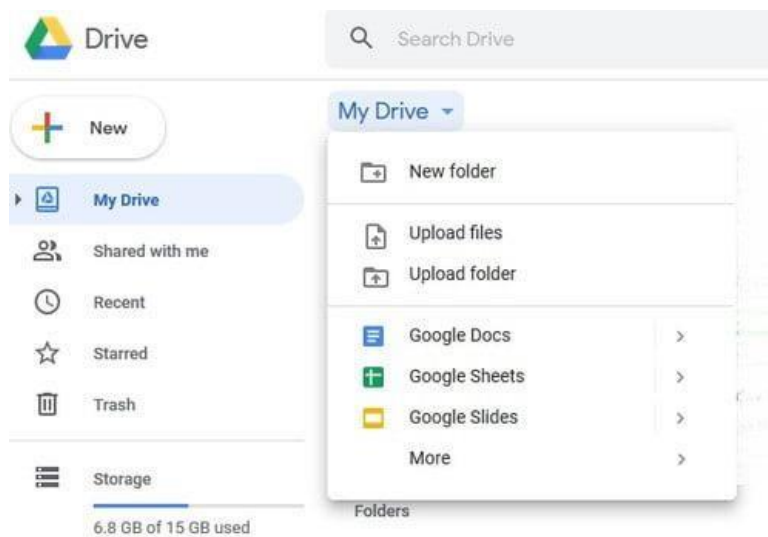
Step 2: Downloading Dropbox to our desktop automatically creates a Dropbox folder. We can use the steps below to sync files to this folder.

1. Locate the Dropbox

Open “**finder**” and locate “**Dropbox**” in our “**favorites**” tray.

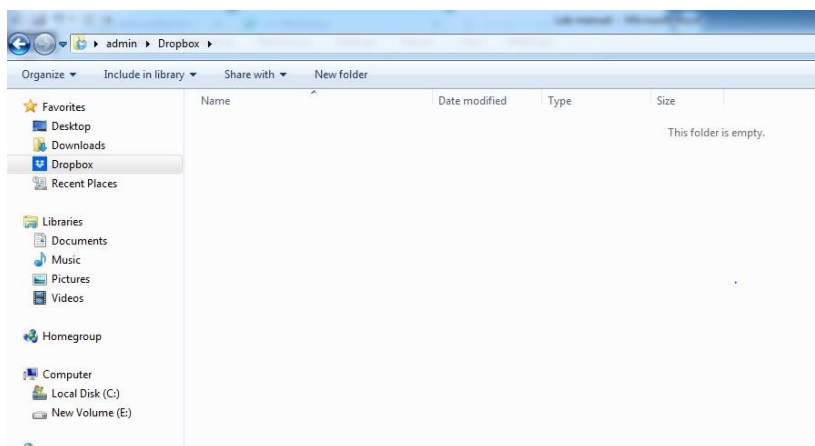


2. Drag and Drop Files into Our Folder



3. Save file to our Dropbox folder

When saving a file, we can choose to save it directly in our Dropbox folder. All files saved to Dropbox will automatically sync to our account.



Use Dropbox on Web

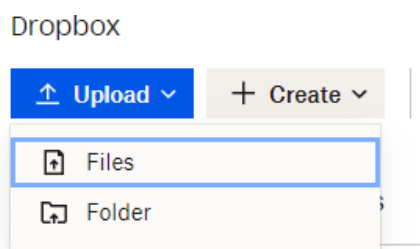
Step 3: If we prefer not to download Dropbox to our computer, we can upload files directly through the web application. It's easy to do; just follow the instructions below to begin backing up our files.

Uploading Files

The first thing we'll want to do with Dropbox is upload some files.

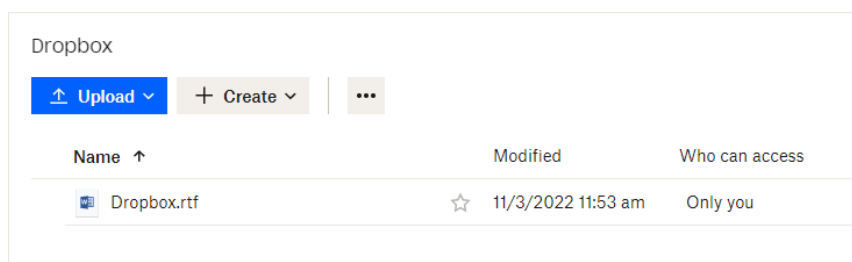
1. Navigate to "Upload Files/Folder"

Log in to our Dropbox account via our web browser. On the right-hand side of the home screen, we'll see the option to **upload a file or folder**.



2. Select the File

Select the file we want to upload and **tap "open."** The file will be uploaded to our Dropbox account. We can follow the same process to upload a folder.

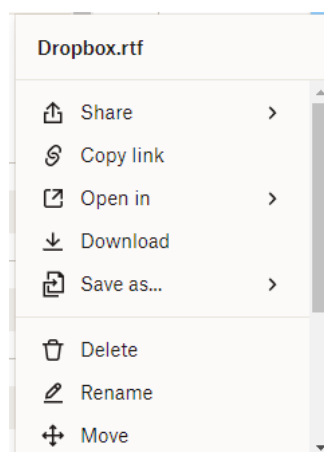


Navigate the Files Using Dropbox

Step 4: Finding files on the using the web application is straightforward and takes no time at all.

1. Select all files

On the right-hand side of the home screen, we will find six options. To find a file or folder, **tap "allfiles."** We can then browse this section to find the files we want.



2. Or Use the Search Bar

Alternatively, if we know the name of the file or folder we want to locate, we can type it into the search bar.

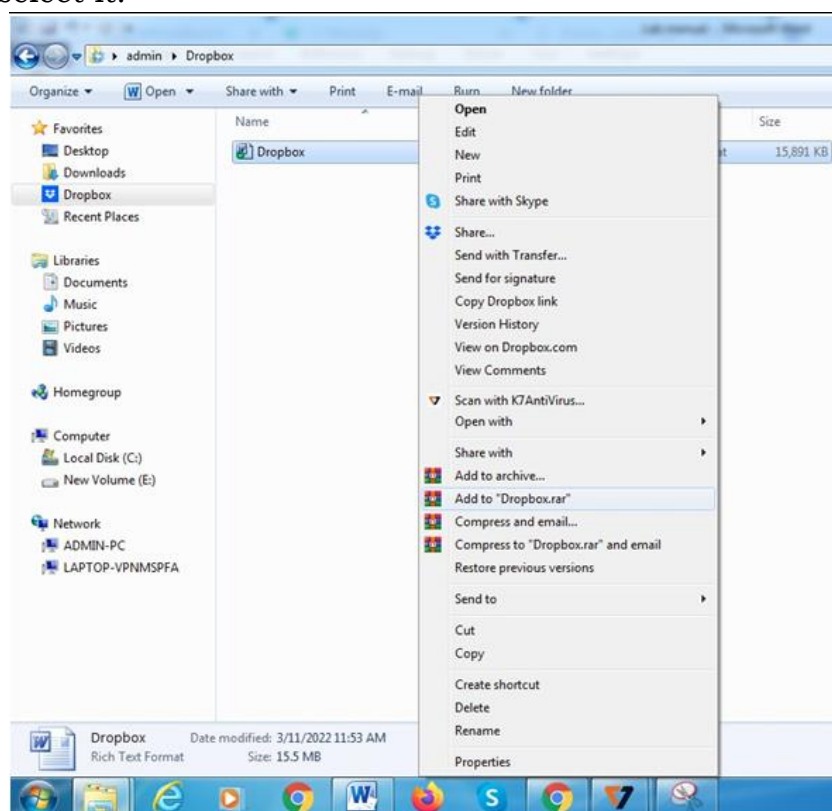


Use Dropbox to share a file or folder

There are multiple ways to share a file or folder through Dropbox.

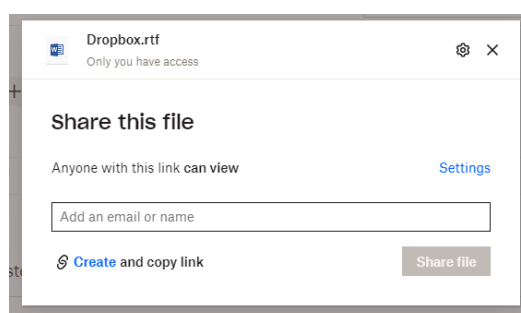
Locate the file In our Dropbox folder

Step 5: Go to the Dropbox folder on our desktop. Search for the files we would like to share and right click our mouse. Find “Share” in dropdown menu and select it.



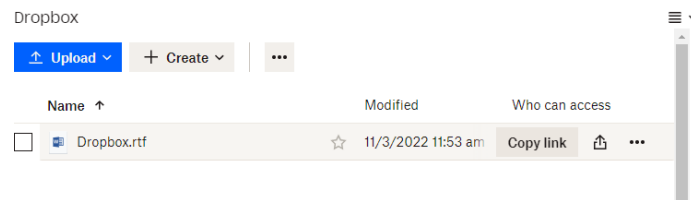
1. Email the File

To email the file access directly, **add the contact information** for the recipient in the “to:” section. Next, **click “share”** and access to the file will be sent directly to the person we selected.



2. Generate a share Link

Instead of emailing our file, we can generate a shareable link. After selecting “**share**,” we will see “**create link**” at the bottom right-hand side of the window.



Use Dropbox to Download Files

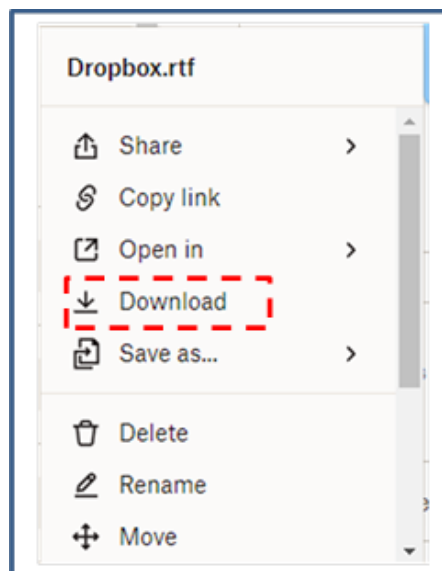
Step 6: Let's say that we're on a new device and we need access to a file we saved to Dropbox, but we don't have the service installed and don't want to install it. Instead, our option is to log in via our browser and navigate to the file we want to download.

1. Locate the file

To download a file onto our computer from the web application, first **locate the file** using the search function as outlined above.

2. Select the file and download

Select the file and tap the ellipsis. From the dropdown menu, select "download".

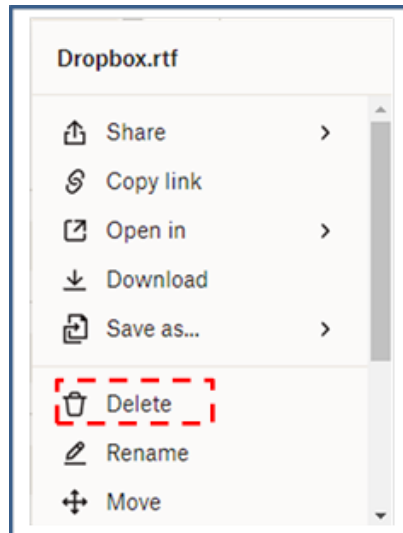


Use Dropbox to Restore files

Step 7: If we accidentally delete a file or folder, fear not: we have 30 days to recover it (120 days on Professional plans). Here's what we can do to recover our files.

1. Locate Deleted Files

On the left-hand side of the home screen in the web application, we will find "**deleted files.**" Select it to move forward.



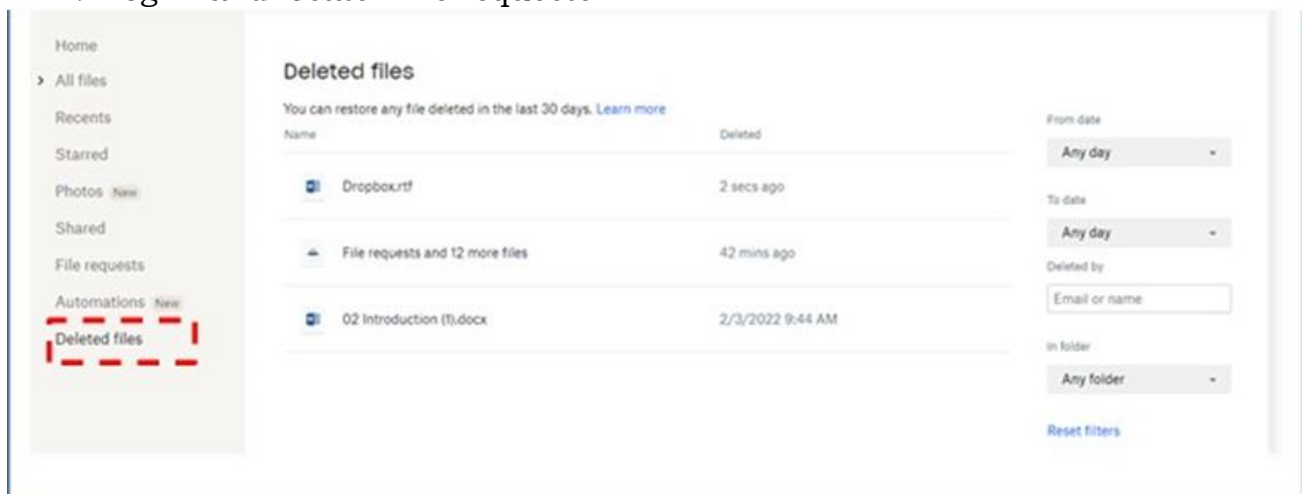
2. Select the file we want to restore

Select the file we would like to restore and press “**restore**” on the right-hand side of the screen. If we would rather delete the file completely, select “**delete permanently**,” located underneath the “**restore**” button

Use Dropbox to Request Files

Step 8: File request allow inviting people who don’t have a Dropbox account to upload files to ours. We may find that useful when we need to get files from employees or relatives who don’t use the platform. It’s also a good way for teachers to collect files from students.

1. Log in and locate “File requests”

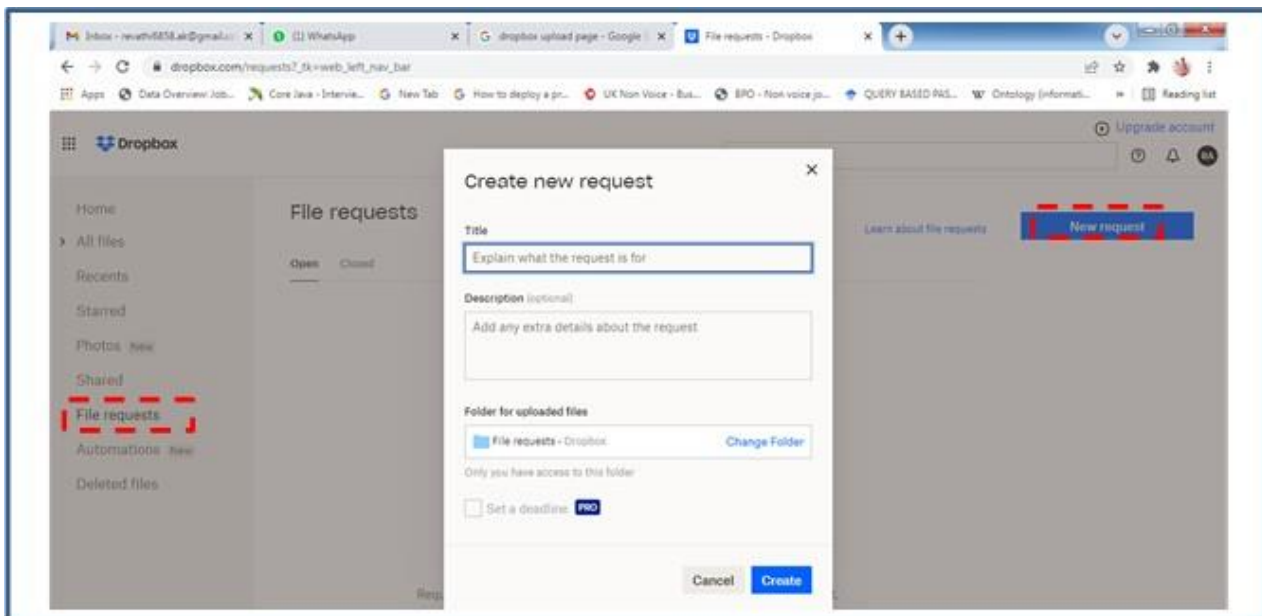


2. Select “New Request”

Next, select “**new request**” on the right-hand side of the screen.

3. Name the Folder and Add a Description

On the next page, we'll be asked to **name the files** we're requesting. For example, we could call them "birthday photos" or "tax documents." Once we've decided on a name, choose the folder we want Dropbox to put them in. Tap "**create**" to move forward.



4. Add Contacts to the Shared folder

On the next page, we can add the contacts we would like to have access to the folder. We can either send the request via email, or copy and paste the shareable link and manually send it to them.

Result:

The above steps are used for uploading, sharing and downloading files using dropbox cloud environments.

Ex.No: 2 DESIGNING OUR OWN WEBSITE USING ADOBE

Date:

AIM:

To explore the features provided by Cloud Resources in Adobe Creative Cloud Express account.

Concept:

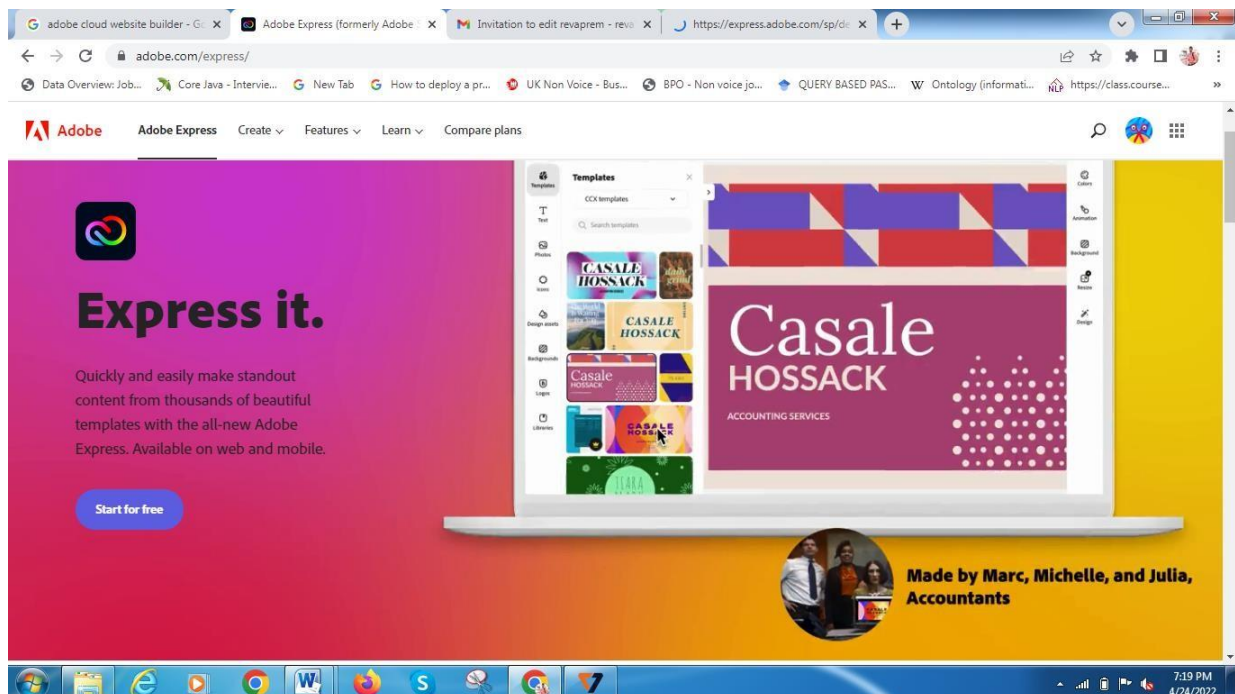
Creative Cloud Express is ideal for projects that don't require more than one page, such as portfolios, resumes, presentations, blog posts, and photo galleries. Creative Cloud Express can showcase a product catalog, advertise a special offer, or act as a weekly or monthly newsletter for businesses.

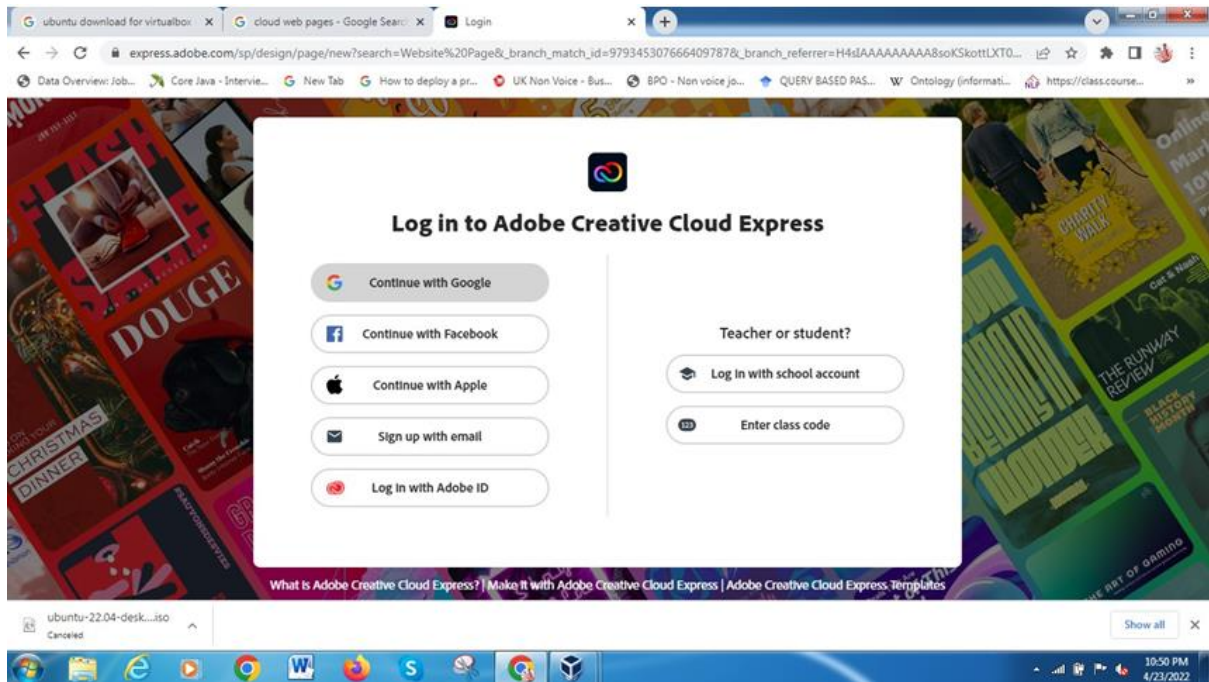
Process/Procedure:

- Create an account <https://www.adobe.com/express/create/website-page>
- Pick a theme
- Choose beautiful images to use
- Add different elements to your web page
- Share your page

Experiment

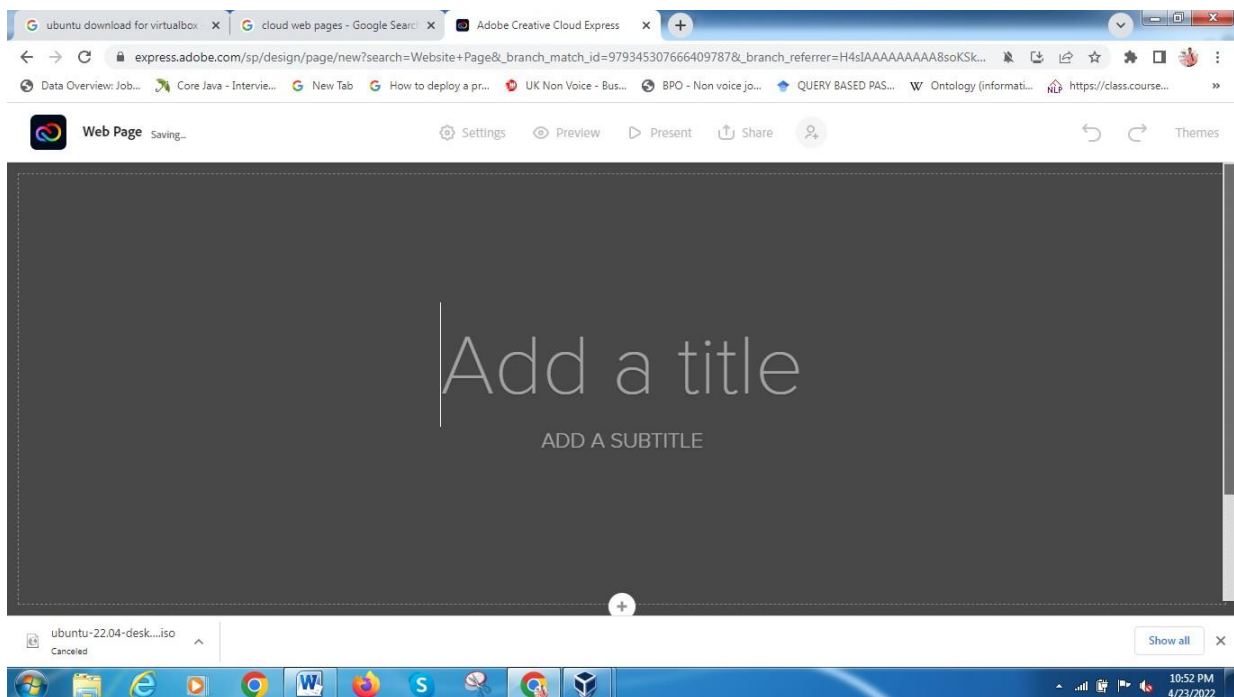
- Step 1: Create a free Creative Cloud Express account online or login using our existing accounts



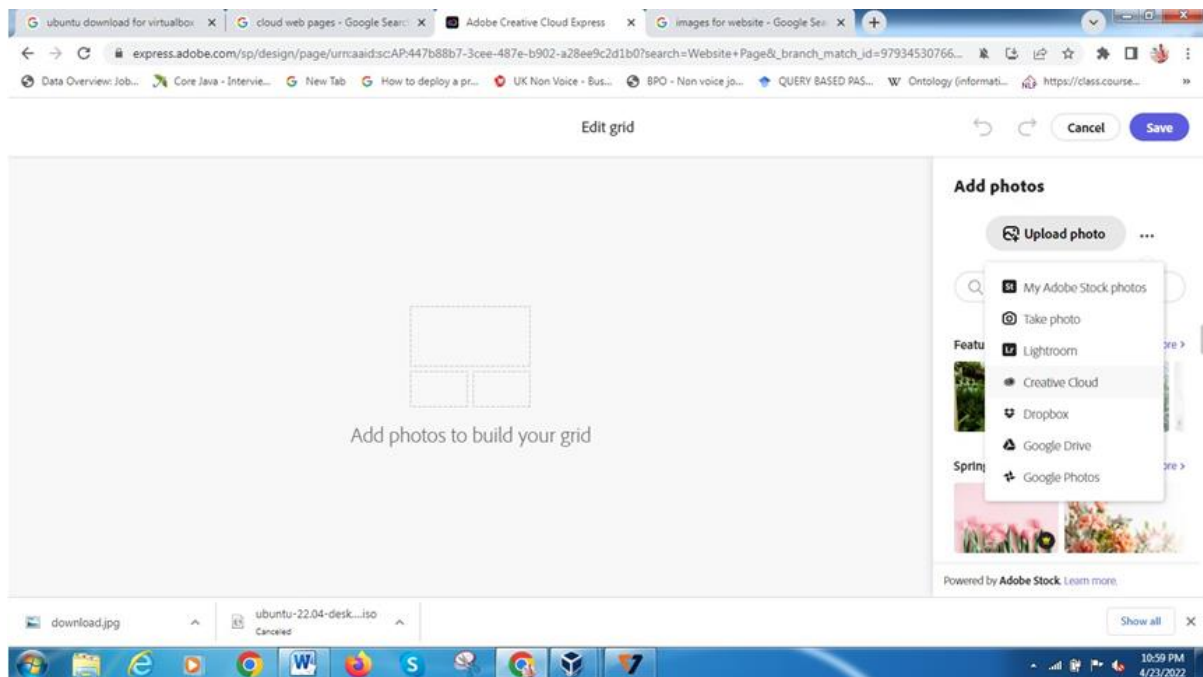


After LOGIN Explore the Following Features of adobe cloud express website design Online CloudEnvironment

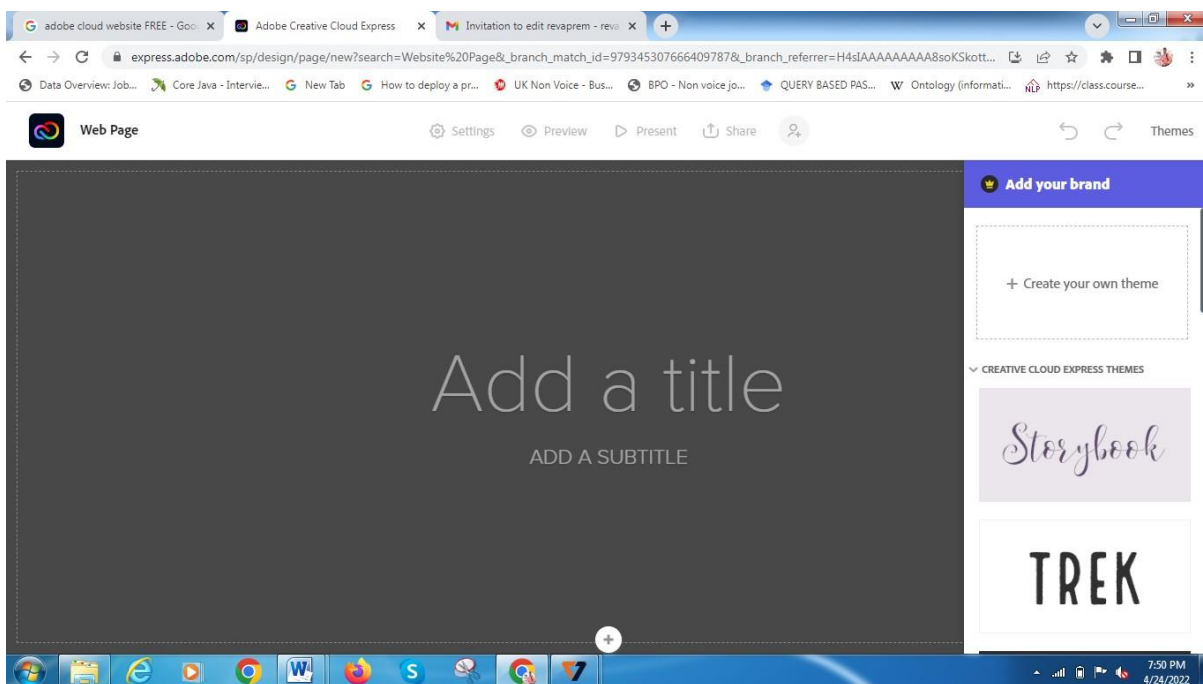
- Step 2: Add a title and subtitle for the webpage

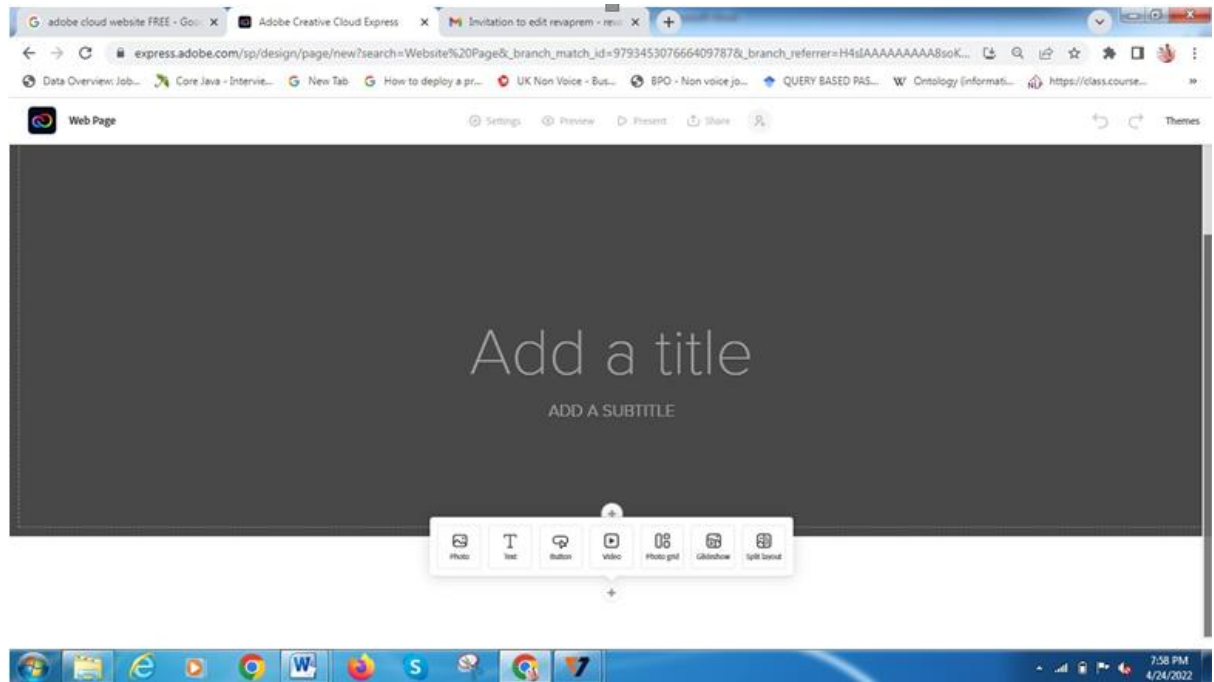


- Step 3: Choose a theme from the themes gallery and set fonts and styles that will completely transform

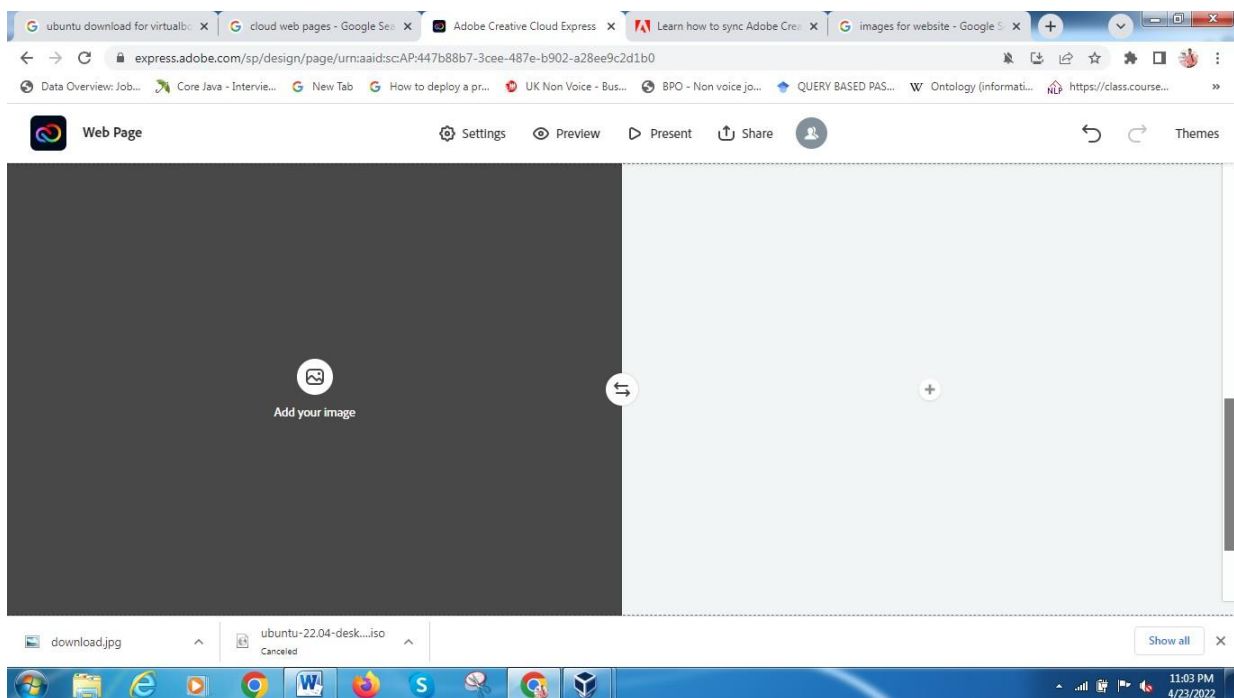


- Step 4: Click split layout for two side layout

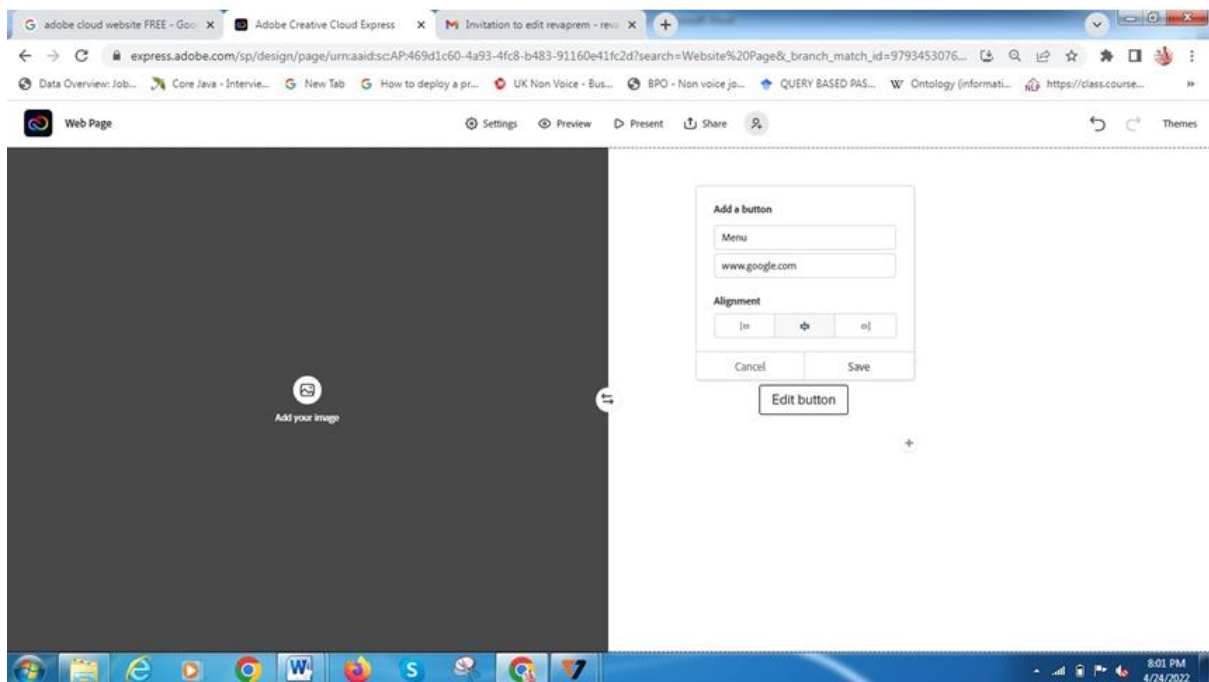




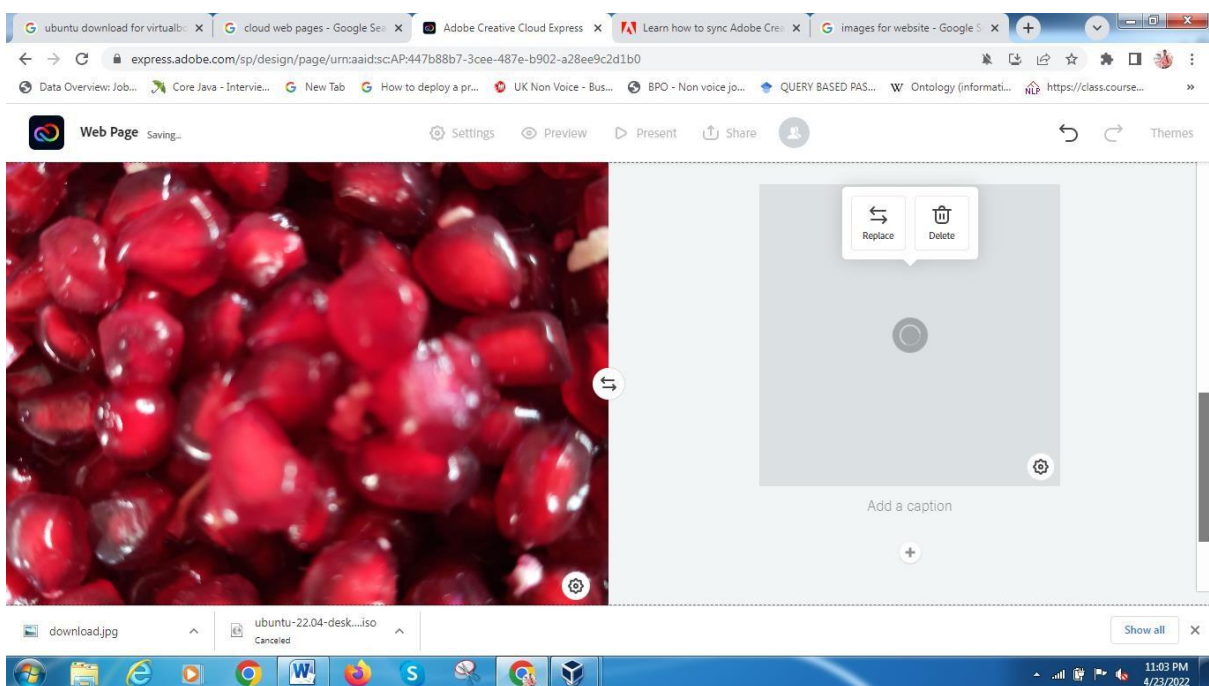
- Step 5: Left side add image and right side add a button and name the button



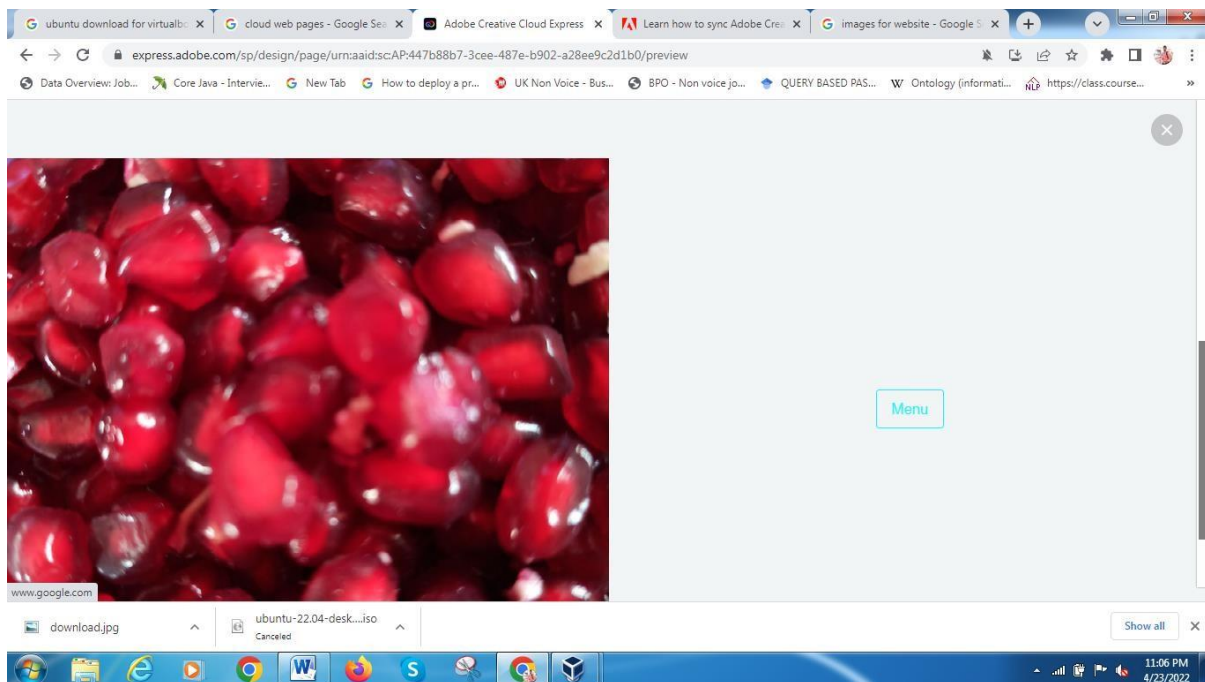
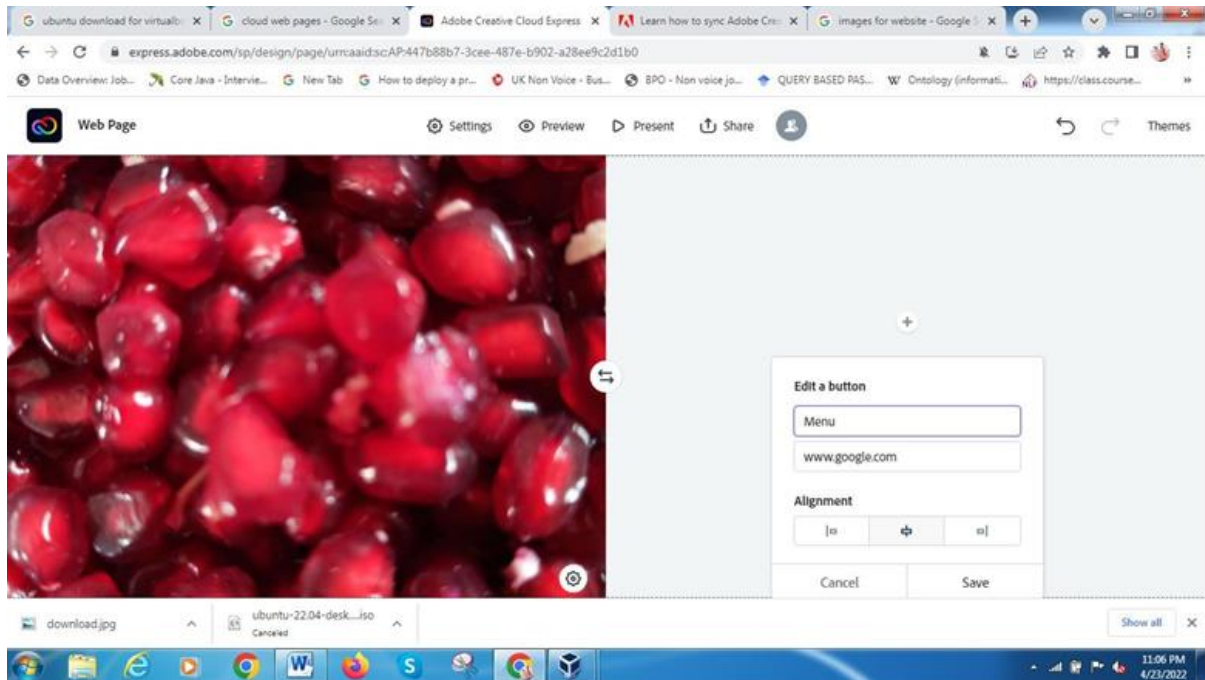
- Step 6: Create a button named ourselves and give some webpage



- Step 6: Create a button named ourselves and give some webpage.

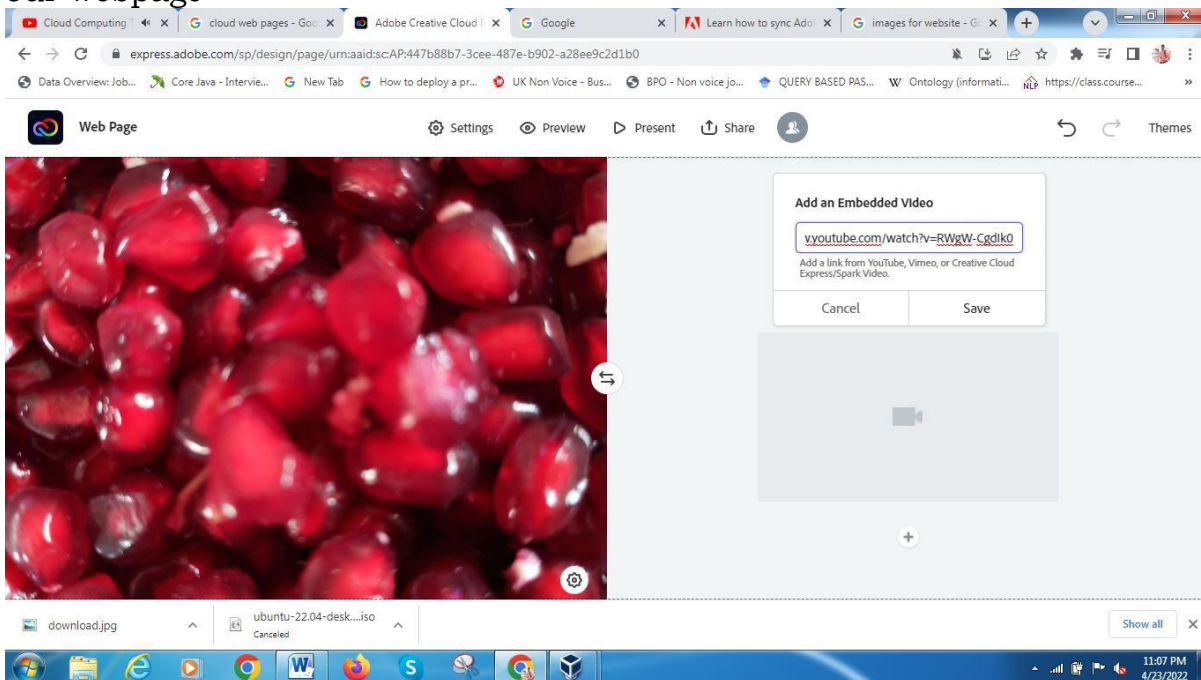


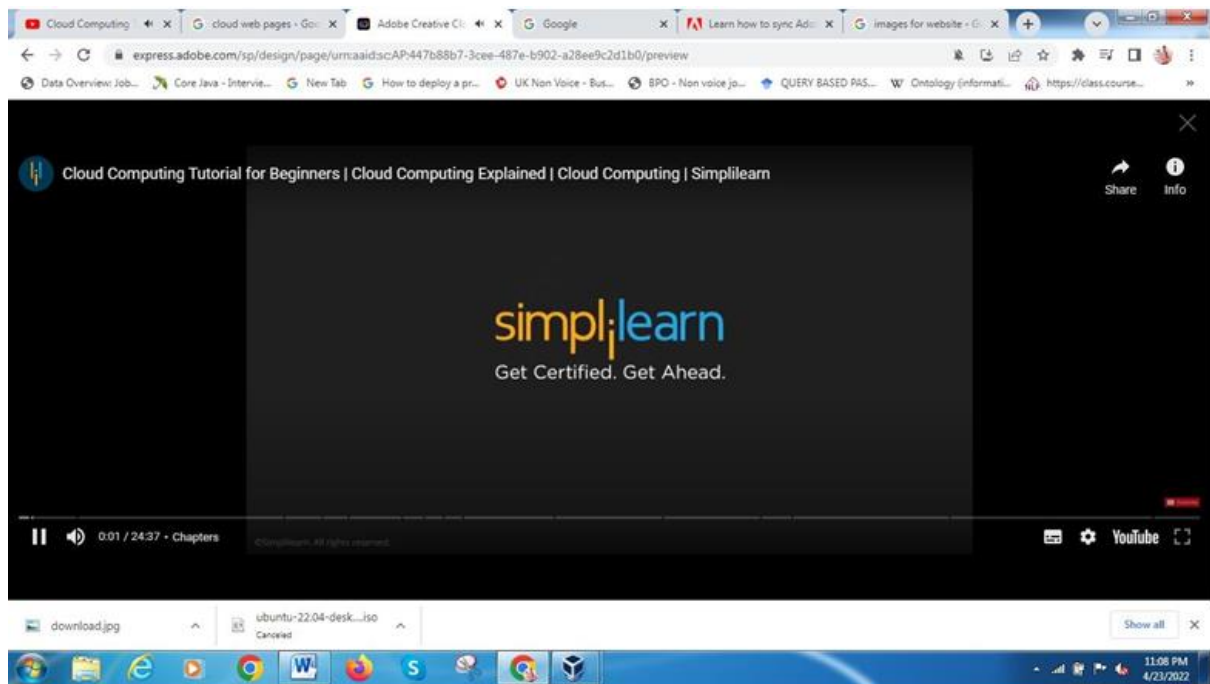
Step 7: Save the button and preview our website design and click the button. It will load the webpage that you added in our page button.





Step 8: Create next button and give some video link. It will load the video in our webpage





Results:

Using the above steps services offered by the adobe express website online cloud environment are explored.

Ex.No: 3 CREATING AN ECLIPSE CLOUD ACCOUNT USING REDHAT

Date:

Aim:

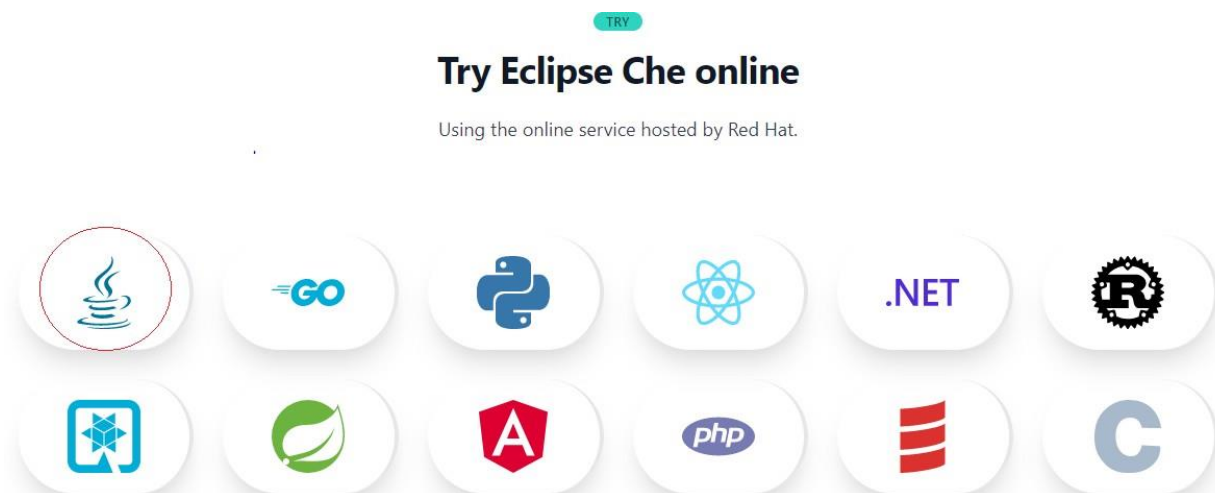
To explore the features provided by Cloud Resources in Eclipse with Redhat cloud account.

Concept:

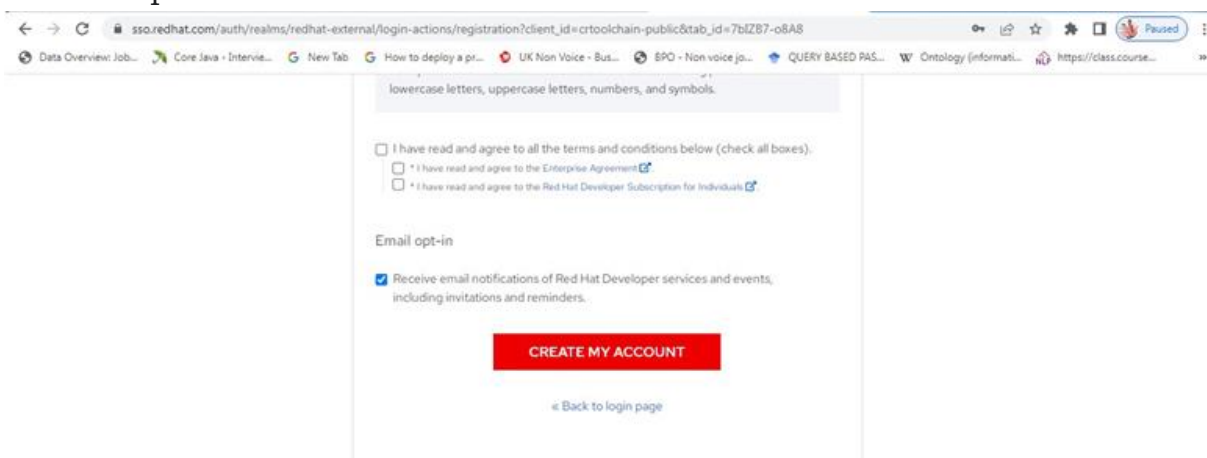
Eclipse Che hosted by Red Hat is an open source product based on Eclipse Che that is running on OpenShift Dedicated. The new service is part of the Developer Sandbox for Red Hat OpenShift offering, and is using CodeReady Workspaces, which is built upon Eclipse Che and is optimized for Red Hat OpenShift and Red Hat Linux.

Experiment:

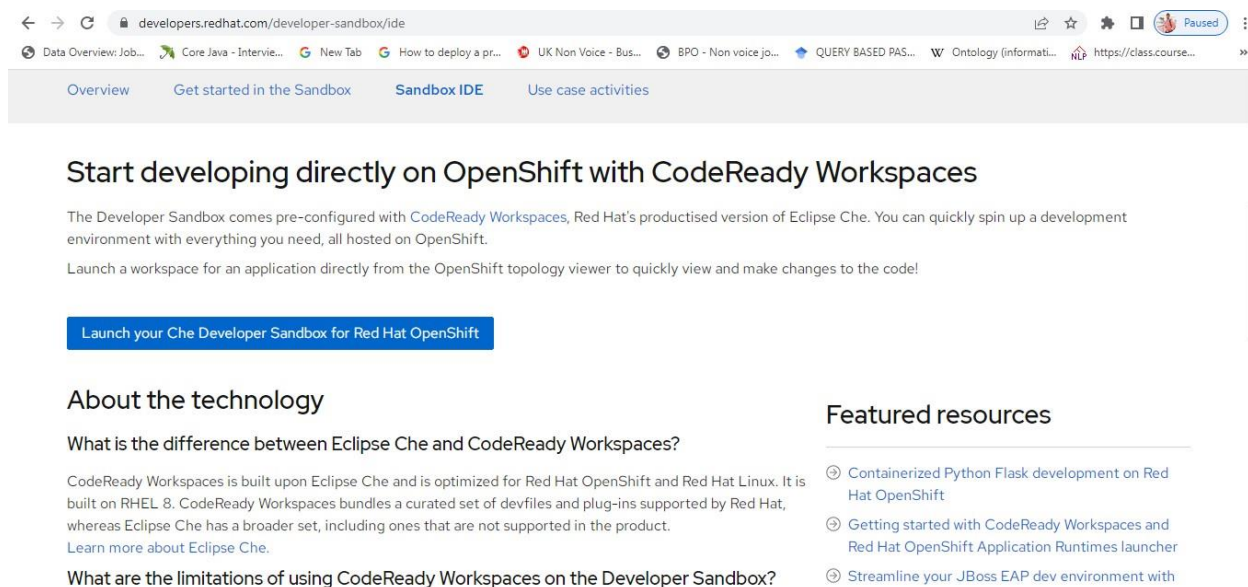
1. Step 1: Start <https://www.eclipse.org/che/gettingT-started/cloud/>
2. Step 2: Click any programming language icon.



3. Step 3: Create an account in redhat sandbox



4. Step 4: Verify your account from the registered email.
5. Step 5: Click on Launch you “Che Developer sandbox for redhat openshift”



← → ↻ 🔒 developers.redhat.com/developer-sandbox/ide

Data Overview: Job... Core Java - Intervie... New Tab How to deploy a pr... UK Non Voice - Bus... BPO - Non voice jo... QUERY BASED PAS... W Ontology (informati... https://class.course...

Overview Get started in the Sandbox **Sandbox IDE** Use case activities

Start developing directly on OpenShift with CodeReady Workspaces

The Developer Sandbox comes pre-configured with [CodeReady Workspaces](#), Red Hat's productised version of Eclipse Che. You can quickly spin up a development environment with everything you need, all hosted on OpenShift.

Launch a workspace for an application directly from the OpenShift topology viewer to quickly view and make changes to the code!

[Launch your Che Developer Sandbox for Red Hat OpenShift](#)

About the technology

What is the difference between Eclipse Che and CodeReady Workspaces?

CodeReady Workspaces is built upon Eclipse Che and is optimized for Red Hat OpenShift and Red Hat Linux. It is built on RHEL 8. CodeReady Workspaces bundles a curated set of devfiles and plug-ins supported by Red Hat, whereas Eclipse Che has a broader set, including ones that are not supported in the product.

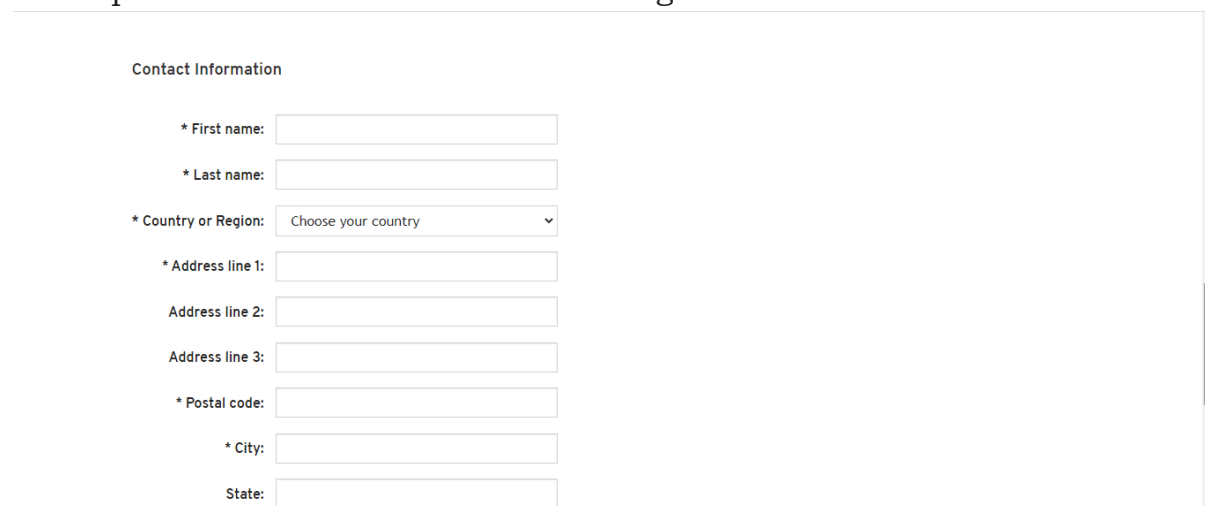
[Learn more about Eclipse Che.](#)

What are the limitations of using CodeReady Workspaces on the Developer Sandbox?

Featured resources

- ③ [Containerized Python Flask development on Red Hat OpenShift](#)
- ③ [Getting started with CodeReady Workspaces and Red Hat OpenShift Application Runtimes launcher](#)
- ③ [Streamline your JBoss EAP dev environment with](#)

6. Step 6: Fill in the further details for login



Contact Information

* First name:

* Last name:

* Country or Region:

* Address line 1:

Address line 2:

Address line 3:

* Postal code:


* City:

State:

7. Step 7: Click on Get Started button
8. Step 8: In the Available services page click Launch on RedHat Openshift

Available services


Now that your Sandbox is activated, these are all the cool things that are available to you, right in your Sandbox!



Red Hat
OpenShift

A cloud-native application platform with everything you need to manage your development life cycle securely, including standardized workflows, support for multiple environments, continuous integration, and release management.


[Launch](#) [Learn more](#)



Red Hat
Dev Spaces

A collaborative Kubernetes-native solution for rapid application development that delivers consistent developer environments on Red Hat OpenShift to allow anyone with a browser to contribute code in under two minutes.

[Launch](#) [Learn more](#)

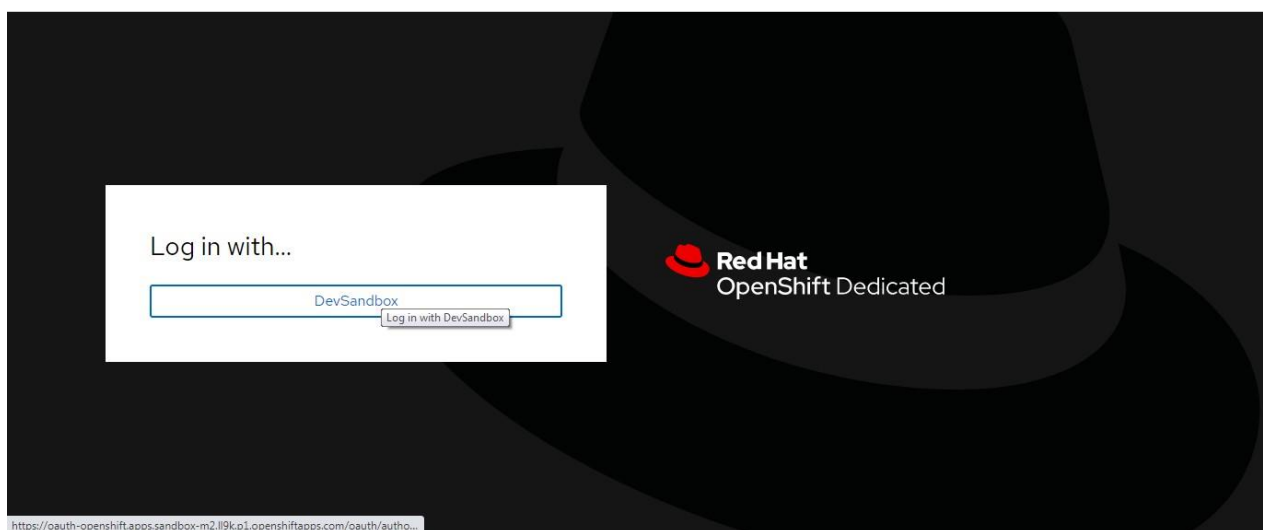


Red Hat
Data Science


Red Hat OpenShift Data Science is a part of the Red Hat OpenShift AI portfolio and provides tools across the AI/ML lifecycle.

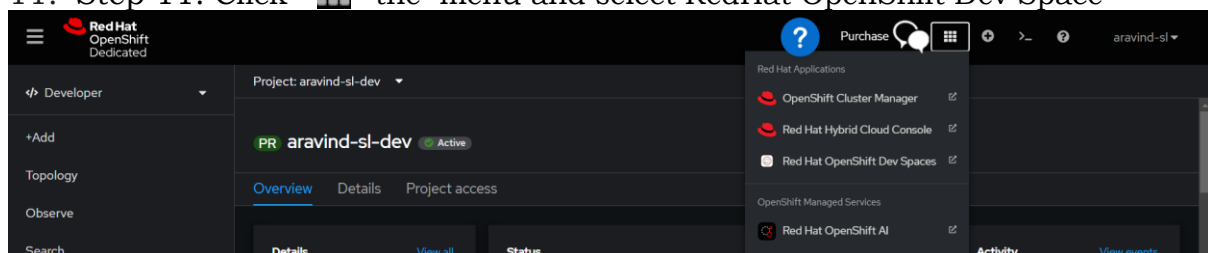
[Launch](#) [Learn more](#)

9. Step 9: Click on login with DevSandbox



10. Step 10: Agree to terms and conditions asked.

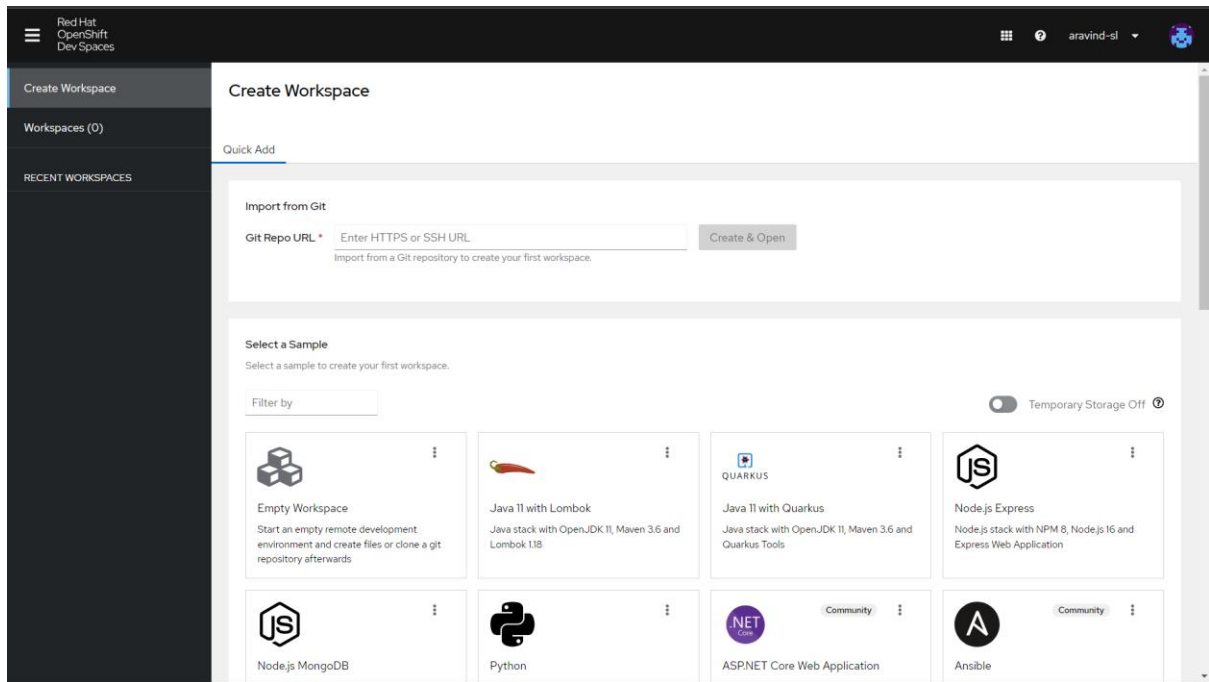
11. Step 11: Click  the menu and select RedHat OpenShift Dev Space



12. Step 12: Log in with OpenShift

13. Step 13: If asked for any permission click on checkbox and allow all permission

14. Step 14: Explore all the language present in the workspace.



Result:

Using the above steps services offered by Eclipse with RedHat Cloud environment is explored.


Aim:

To explore the features provided by Eclipse with Redhat cloud account and create Password using C++ Program Language.

Concept:

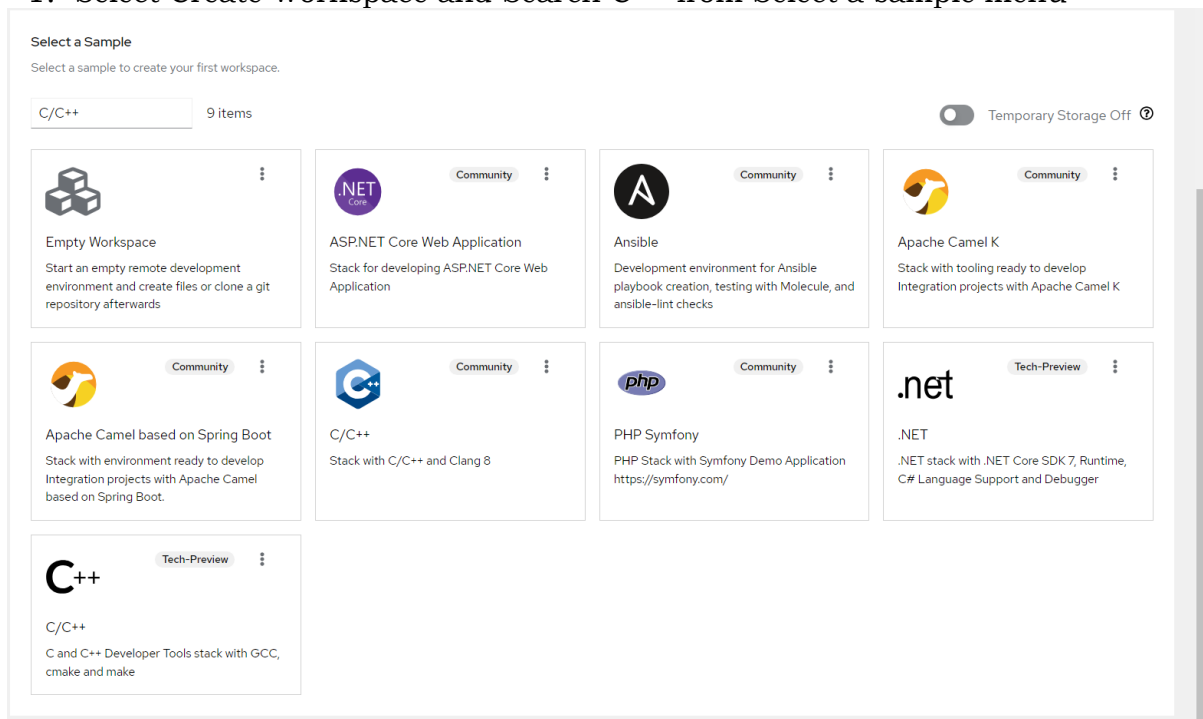
Eclipse Che hosted by Red Hat is an open source product based on Eclipse Che that is running on OpenShift Dedicated. The new service is part of the Developer Sandbox for Red Hat OpenShift offering, and is using CodeReady Workspaces, which is built upon Eclipse Che and is optimized for Red Hat OpenShift and Red Hat Linux

Process/Procedure:

1. Go to <https://www.eclipse.org/che/getting-started/cloud/>
2. Click any programming language icon.
3. Login to your account in redhat sandbox
4. Click on login with OpenShift
5. Click on login with DevSandbox
6. Click the  menu and select RedHat OpenShift Dev Space.

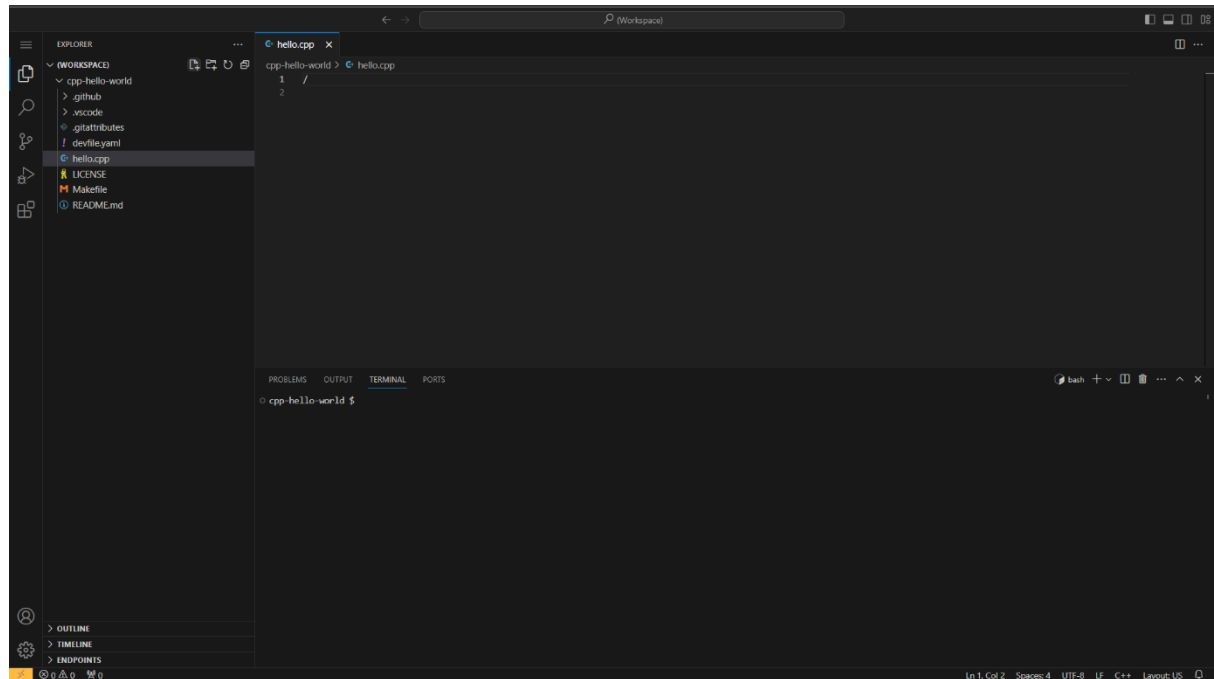
Environment

1. Select Create Workspace and Search C++ from Select a sample menu



2. Select C/C++ Community, RedHat will create a sandbox environment

3. In the Workspace Environment Create a New file  by clicking icon.



4. Create a New file and name it passwordCreation.cpp and write the following code.

```
#include<iostream>
#include<string.h>

using namespace std;

int main(){

    char pass[20], ePass[20]; int
    numOne, numTwo, sum;

    cout<<"Create a Password: ";
    cin>>pass;

    cout<<"\nEnter Two Numbers to Add: ";
    cin>>numOne>>numTwo;

    cout<<"\nEnter the Password to See the Result: ";
    cin>>ePass;

    if(!strcmp(pass, ePass)){
        sum = numOne + numTwo;

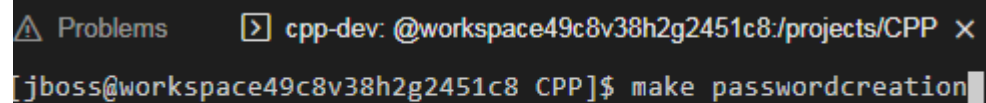
        cout<<endl<<numOne<<" + "<<numTwo<<" = "<<sum;
    }else

        cout<<"\nSorry! You've entered a Wrong Password!";
    cout<<endl;

    return 0;
```

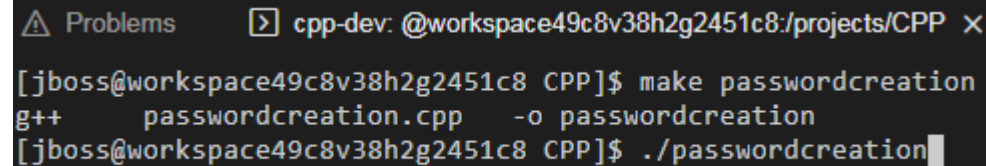

}

5. Press ctrl + shift + ` to open new terminal.
6. In the terminal write the command “make PasswordCreation”



A screenshot of a terminal window. At the top, there is a status bar with a yellow triangle icon, the word 'Problems', a yellow square icon, the text 'cpp-dev: @workspace49c8v38h2g2451c8:/projects/CPP', and a close button 'X'. The terminal prompt is '[jboss@workspace49c8v38h2g2451c8 CPP]\$' and the command 'make passwordcreation' is being typed, with a cursor at the end of the line.

7. Run the program using the command “./PasswordCreation”



A screenshot of a terminal window. At the top, there is a status bar with a yellow triangle icon, the word 'Problems', a yellow square icon, the text 'cpp-dev: @workspace49c8v38h2g2451c8:/projects/CPP', and a close button 'X'. The terminal prompt is '[jboss@workspace49c8v38h2g2451c8 CPP]\$'. The command 'make passwordcreation' is entered, followed by the output 'g++ passwordcreation.cpp -o passwordcreation'. Then, the command './passwordcreation' is entered, and the cursor is at the end of the line.

Output:

```
● cpp-hello-world $ ./PasswordCreation
Create a Password: 12345

Enter Two Numbers to Add: 3
4

Enter the Password to See the Result: 12345

3 + 4 = 7
○ cpp-hello-world $
```

Result:

Using the above steps services offered by Eclipse with RedHat cloud environment for develop and run the C++ Program.

Ex.No: 5

DEVELOPING JAVA PROGRAM FROM SANDBOX


Date:

ENVIRONMENT


Aim:

To explore the features provided by Eclipse with Redhat cloud account and develop the java program for pattern creation.

Process/Procedure:

1. Go to <https://www.eclipse.org/che/getting-started/cloud/>
2. Click any programming language icon.
3. Login to your account in redhat sandbox
4. Click on login with OpenShift
5. Click on login with DevSandbox
6. Click the  menu and select RedHat OpenShift Dev Space.

Experiment:

1. Select Create Workspace and Search Java from Select a sample menu.
2. Select Java Community, RedHat will create a sandbox environment
3. In the Workspace Environment Create a New file by clicking  icon.
4. Name the file PyramidPattern.java and write the following code for printing pyramid pattern using star(*).

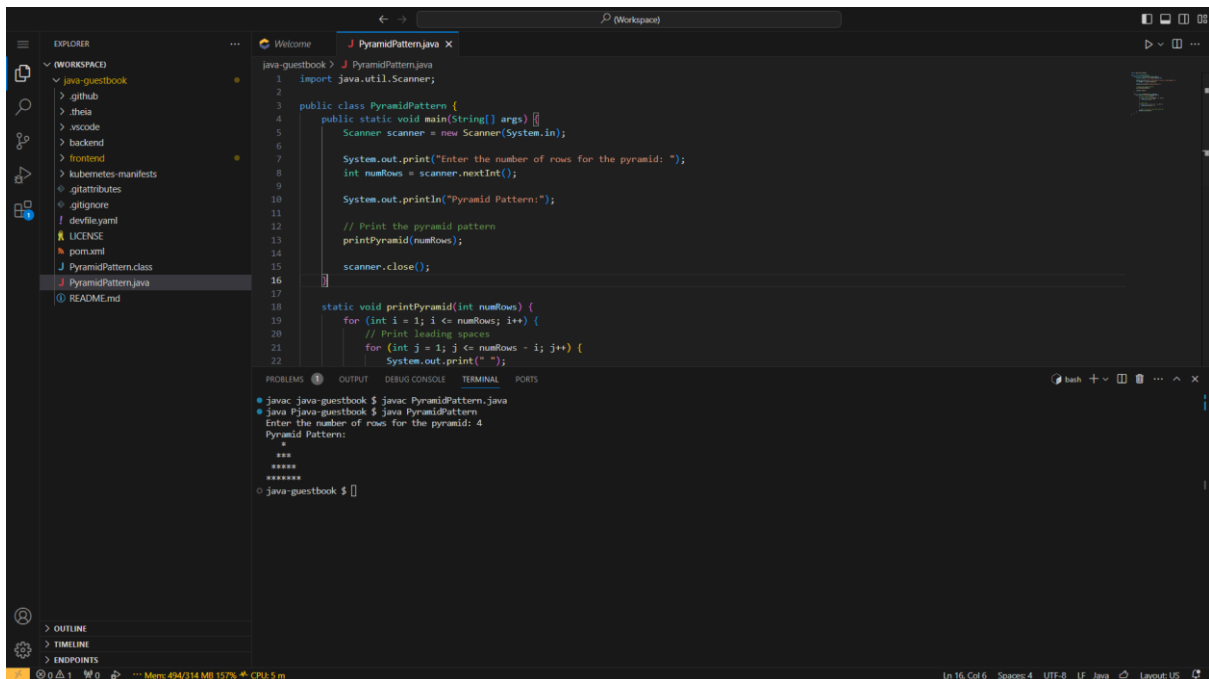
```
import java.util.Scanner;

public class PyramidPattern {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.print("Enter the number of rows for the pyramid: ");
        int numRows = scanner.nextInt();
        System.out.println("Pyramid Pattern:");
        printPyramid(numRows);
        scanner.close();
    }

    static void printPyramid(int numRows) {
        for (int i = 1; i <= numRows; i++) {
            // Print leading spaces
            for (int j = 1; j <= numRows - i; j++) {
                System.out.print(" ");
            }
            // Print stars
            for (int k = 1; k <= 2 * i - 1; k++) {
                System.out.print("*");
            }
            // Move to the next line after each row
            System.out.println();
        }
    }
}
```

```
}  
}
```

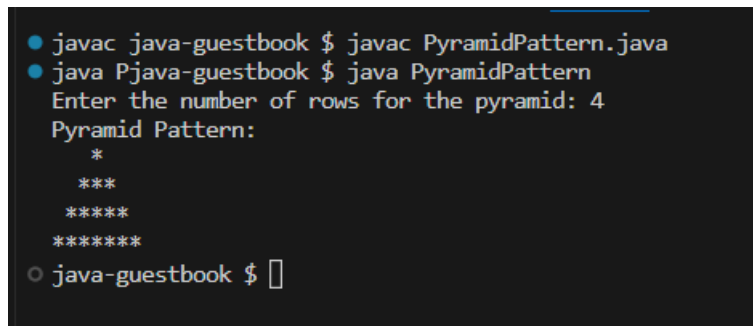
5. Press `ctrl + shift + `` to open new terminal.



6. Type the compile command “`javac PyramidPattern.java`”

7. Run the program using “`java PyramidPattern`”

Output:



Result:

Using the above steps services offered by Eclipse with RedHat cloud environment for develop andrun the Java Program.

Ex.No: 6

DEVELOPING PYTHON PROGRAM FROM SANDBOX


Date:

ENVIRONMENT


Aim:

To explore the features provided by Eclipse with Redhat cloud account and develop the python program for pattern creation.

Process/Procedure:

1. Go to <https://www.eclipse.org/che/getting-started/cloud/>
2. Click any programming language icon.
3. Login to your account in redhat sandbox
4. Click on login with OpenShift
5. Click on login with DevSandbox
6. Click the  menu and select RedHat OpenShift Dev Space.

Experiment:

1. Select Create Workspace and Search python from Select a sample menu.
2. Select python Community, RedHat will create a sandbox environment
3. In the Workspace Environment Create a New file by clicking  icon.
4. Name the file Fibonacci.py and write the following program that generates the Fibonacci sequence up to a specified number of terms.

```
def generate_fibonacci(n):  
    fibonacci_sequence = [0, 1]  
    while len(fibonacci_sequence) < n:  
        next_term = fibonacci_sequence[-1] + fibonacci_sequence[-2]  
        fibonacci_sequence.append(next_term)  
    return fibonacci_sequence  
  
if __name__ == "__main__":  
    num_terms = int(input("Enter the number of terms for the Fibonacci  
sequence: "))  
    if num_terms <= 0:  
        print("Please enter a positive integer.")  
    else:  
        fibonacci_result = generate_fibonacci(num_terms)  
        print("Fibonacci Sequence:", fibonacci_result)
```
5. Press ctrl + shift + ` to open new terminal.
6. Run the program using “python Fibonacci.py”

Output:

```
python-hello-world $ python Fibonacci.py
Enter the number of terms for the Fibonacci sequence: 5
Fibonacci Sequence: [0, 1, 1, 2, 3]
python-hello-world $
```

Result:

Using the above steps services offered by Eclipse with RedHat cloud environment for develop and run the Python Program.

Ex.No: 7 SETTING UP JAVA AND ECLIPSE IDE FOR WEBSERVICE

Date:

Aim:

To install and configure Java JDK and Eclipse IDE development environment.

Concept:

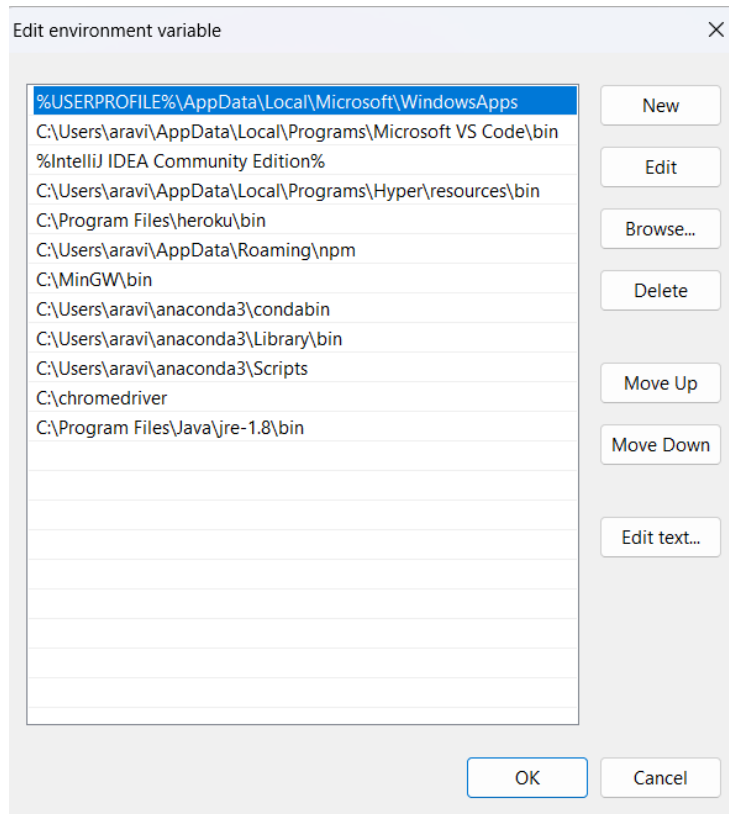
Eclipse IDE is a popular integrated development environment used for Java development and various other languages. Eclipse IDE for Java EE Developers includes additional tools for Java EE development. This lab will guide you through the installation process.

Experiment:

1. Step 1: Check java version
 - a. Open a command prompt on your Windows machine. You can do this by pressing “Win + R”, typing “cmd”, and pressing Enter. In the command prompt, type the following command “java -version”
 - b. The output should start like - java version "1.8.0_401"

```
C:\Users\aravi>java -version
java version "1.8.0_401"
Java(TM) SE Runtime Environment (build 1.8.0_401-b10)
Java HotSpot(TM) 64-Bit Server VM (build 25.401-b10, mixed mode)
```

- c. If Not, Install java 8 in your computer.
2. Step 2: Setting up Environment variables
 - a. Press windows key and search “Environment Variables”
 - b. Click on “Edit the system environment variables”
 - c. Click on the "Environment Variables" button.
 - d. Click “path” under the "System variables" and “User variables” section.
 - e. Select new and paste the java bin path i.e “C:\Program Files\Java\jre-1.8\bin”
3. Step 3: Eclipse Installation
 - a. Open your web browser and go to the link <https://www.eclipse.org/downloads/packages/> .and select Eclipse installer for windows x86_64 to download eclipse installer exe file.
 - b. Open the installer and choose “**Eclipse IDE for Enterprise Java and Web Developers** ”
 - c. Extract the Zip file
 - d. Inside the folder install the eclipse application.



4. Step 3: Eclipse Installation

- a. Open your web browser and go to the link <https://www.eclipse.org/downloads/packages/> .and select Eclipse installer for windows x86_64 to download eclipse installer exe file.
- b. Open the installer and choose “**Eclipse IDE for Enterprise Java and Web Developers** ”
- c. Extract the Zip file
- d. Inside the folder install the eclipse application.

Result:

Thus, Java JDK and Eclipse IDE has been successfully installed and setup in our local system

Aim:

To build a simple web service in Java to get user details like name and age. The goal is to show how to use this service through a test program for easy understanding and usage.

Concept:

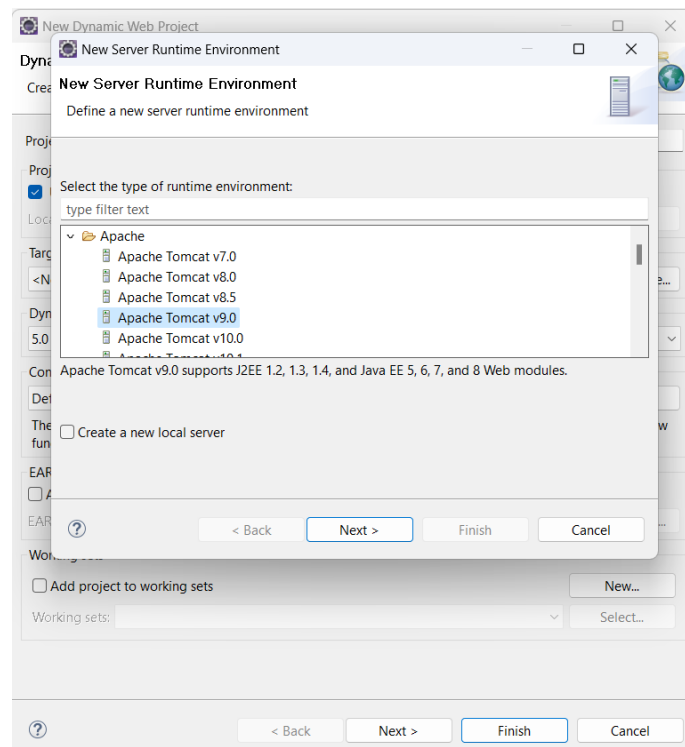
"SOAP web services facilitate communication between different systems over the internet, allowing them to exchange structured information. It relies on XML as the message format and uses standard protocols like HTTP for communication.

Process/Procedure:

1. Open your Web browser and go to the link <https://tomcat.apache.org/download-90.cgi>
2. Install tomcat for windows by Binary Distributions -> Core -> 64-bit Windows zip
3. After the download is complete locate the folder and extract the zip file
4. Store the path of the apache-tomcat-9.0.82 folder

Experiment:**1. Step 1: Create a Dynamic Web Project**

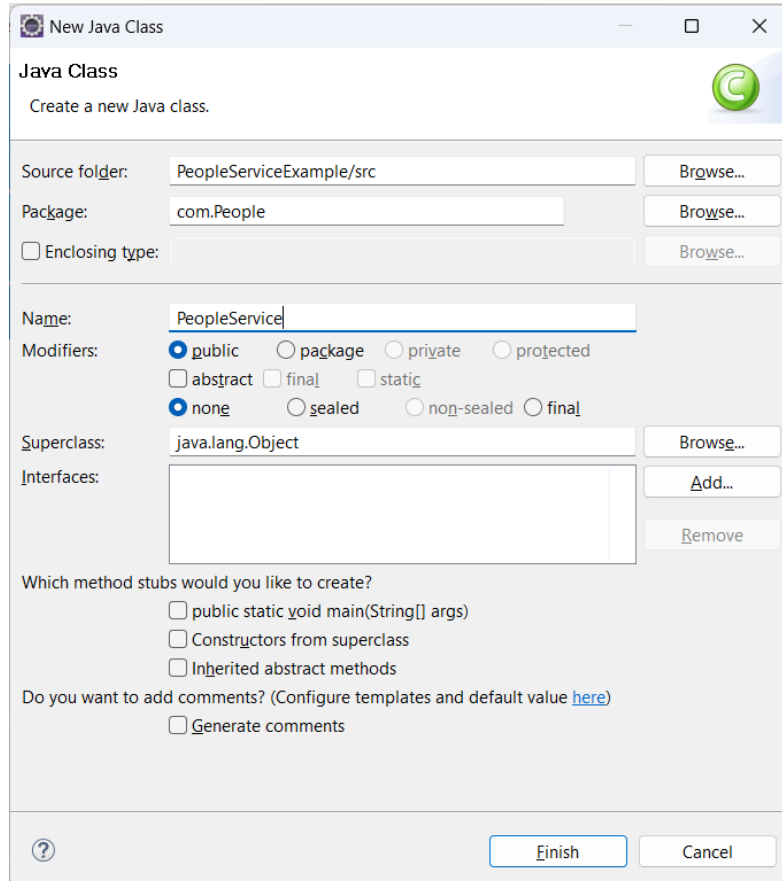
- a. Launch Eclipse IDE.
- b. Go to "File" -> "New" -> "Dynamic Web Project."
- c. Enter a project name (e.g., PeopleServiceExample) and click "Next."
- d. Select the new runtime in target runtime option and choose Apache Tomcat v9.0 and click next.



- e. For the Tomcat installation directory give the path for apache-tomcat-9.0.82
- f. Click "Finish" to create the project.

2. Step 2: Create a Service Endpoint Interface (SEI) Implementation

- a. Right-click on the "src" folder within your project.
- b. Select "New" -> "Class".
- c. Enter package name as "com.people"
- d. Enter a name for the class (e.g., PeopleService).



- e. Click "Finish" to create the class.
- f. Write the following program in the PeopleService.java file

```
package com.People;
```

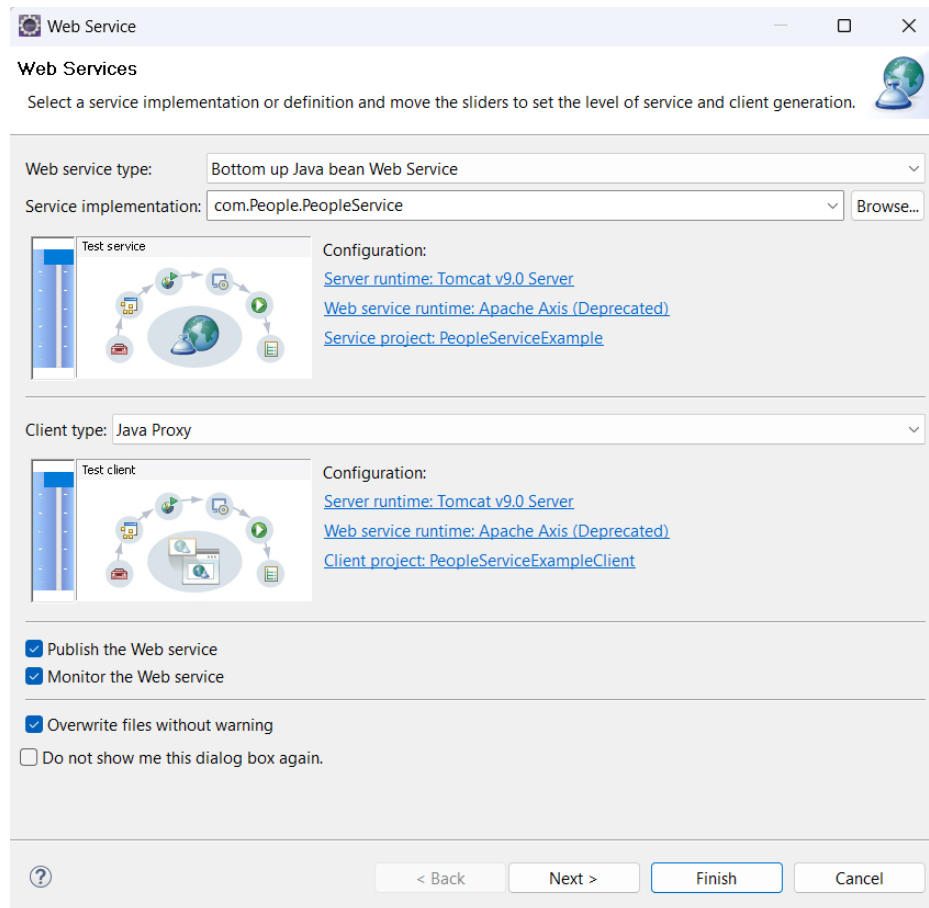
```
public class PeopleService {
    private String name = "";
    private int age = 0;
    private String gender = "";
```

```
    public String getUserDetail(String name, int age, String
gender) {
        String output = "Name :" + name + "\nage : " + age +
"\nGender : " + gender;
        this.name = name;
        this.age = age;
        this.gender = gender;
        return output;
    }
}
```

}

3. Step 3: Create a WebService for the project

- Right-click on the "PeopleServiceImpl.java" file and select Web Services -> Create a WebService.
- In the Pop up box set both the scroller to the maximum limit and check the "Publish the Webservice" and "Monitor the WebService" boxes.



- Click on "Finish".

4. Step 4: Test the WebService

- In the Opened pop up window click on "getUserDetails()" link
- Fill in the details of name, age and gender asked.
- Click on invoke button to display the output.

Result:

Thus, The program successfully retrieves and displays user details, demonstrating the functionality of the SOAP web service and was tested using a simple java project.

Aim:

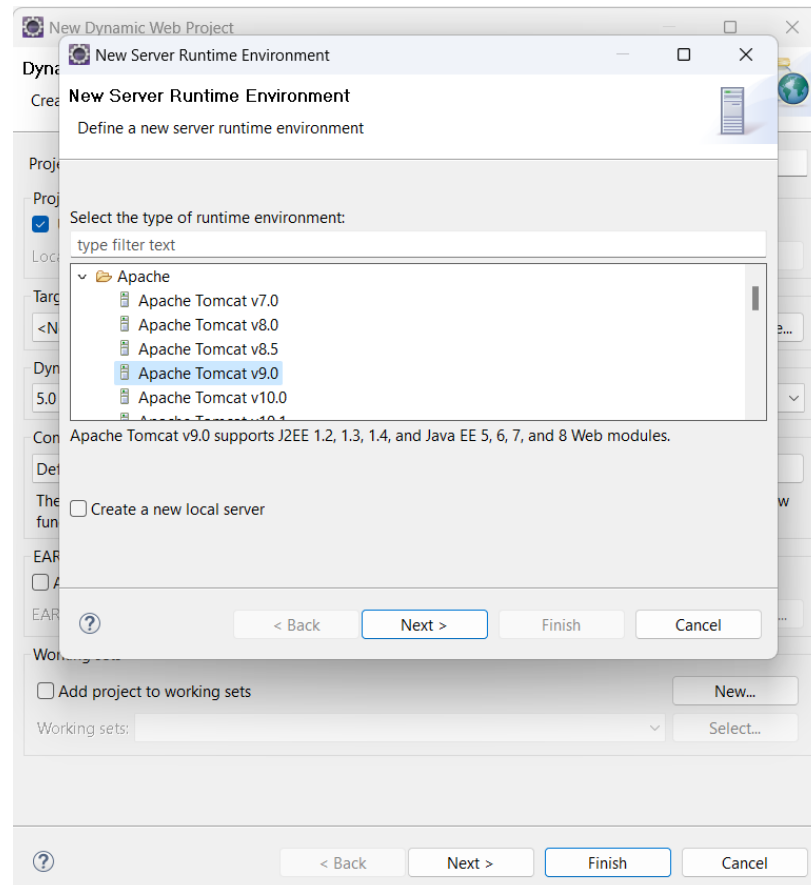
To build a simple web service in Java to get user details like name and age. The goal is to show how to use this service through a test program for easy understanding and usage.

Process/Procedure:

1. Open your Web browser and go to the link
<https://tomcat.apache.org/download-90.cgi>
2. Install tomcat for windows by Binary Distributions -> Core -> 64-bit Windows zip
3. After the download is complete locate the folder and extract the zip file
4. Store the path of the apache-tomcat-9.0.82 folder

Experiment:**1. Step 1: Create a Dynamic Web Project**

- a. Launch Eclipse IDE.
- b. Go to "File" -> "New" -> "Dynamic Web Project."
- c. Enter a project name (e.g., PrimeCheckerService) and click "Next."
- d. Select the new runtime in target runtime option and choose Apache Tomcat v9.0 and click next



- e. For the Tomcat installation directory give the path for

apache-tomcat-9.0.82

f. Click "Finish" to create the project.

2. Step 2: Create a Service Endpoint Interface (SEI) Implementation

- a. Right-click on the "src" folder within your project.
 - b. Select "New" -> "Class".
 - c. Enter a package name (e.g., com.prime).
 - d. Enter the class name (e.g., PrimeService).
 - e. Click "Finish" to create the class.
- i. Write the following code in CalculatorServiceImpl.java file.

```
package com.prime;

public class PrimeService {

    public String isPrime(int num) {
        boolean isPrime = true;
        for (int i = 2; i <= Math.sqrt(num); i++) {
            if (num % i == 0) {
                isPrime= false;
                break;
            }
        }

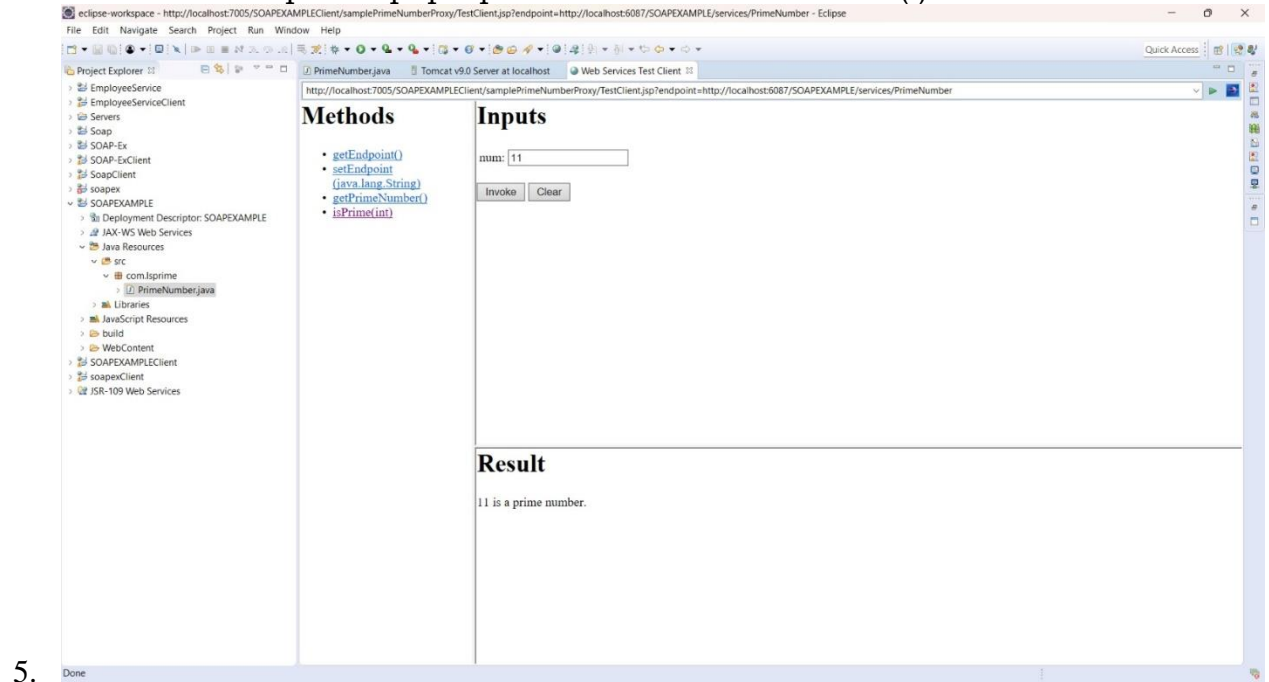
        if (isPrime) {
            System.out.println(num + " is a prime
number.");
        } else {
            System.out.println(num + " is not a
prime number.");
        }
    }
}
```

3. Step 3: Create a WebService for the project

- a. Right-click on the "PeopleServiceImpl.java" file and select Web Services -> Create a WebService.
- b. In the Pop up box set both the scroller to the maximum limit and check the "Publish the Webservice" and "Monitor the WebService" boxes.
- c. Click on "Finish".

4. Step 4: test the WebService

a. In the Opened pop up window click on “isPrime()” link



- a. Fill in the details of name, age and gender asked.
- b. Click on invoke button to display the output.

Result:

Thus, The program successfully retrieves and displays user details, demonstrating the functionality of the SOAP web service and was tested using a simple java project.

Ex.No: 10

NUMBER OF OCCURRENCE OF WORDS IN A BOOK

Date:

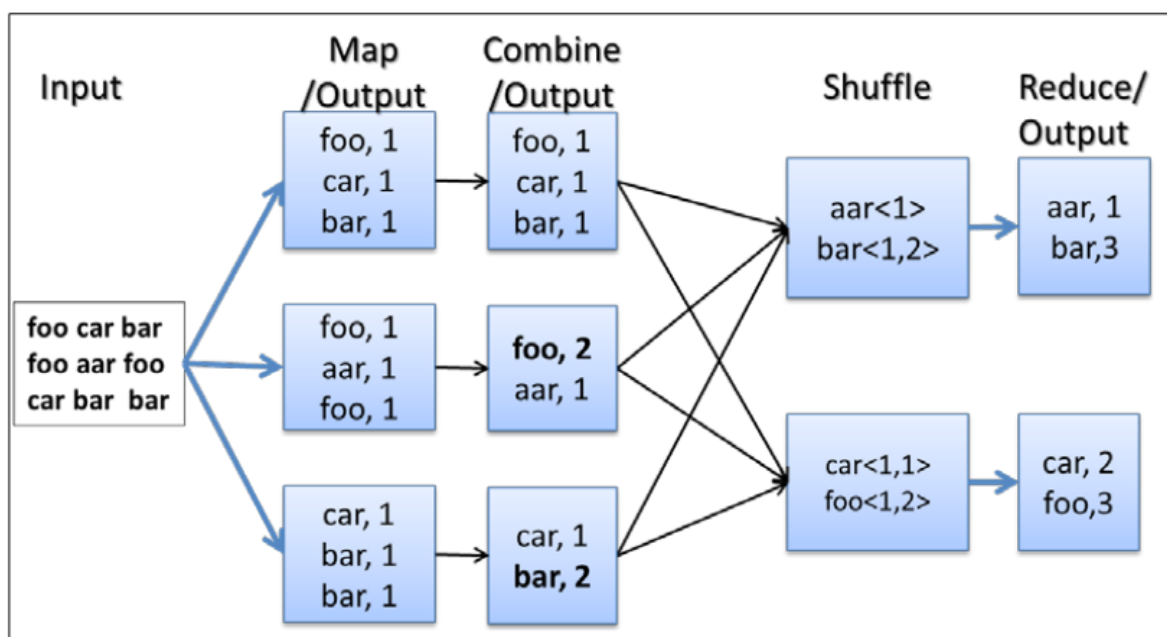
DATASET

Aim:

To write a MapReduce application in java to count the number of occurrences of words in a dataset and run it on single node Hadoop cluster.

Concept:

Hadoop WordCount is a simple distributed computing example where Hadoop processes large text data, counts the occurrences of each word, and produces a summary. It showcases Hadoop's ability to parallelize tasks across a cluster for efficient data processing.



Process/Procedure:

1. Start the Hadoop daemons by the following command “start-all.sh”
2. Ensure all the 5 daemons of Hadoop is running by the command “jps”
3. Download the dataset for the program from the following link
<https://github.com/Inndy/websec101/blob/master/src/xss/the-hunger-games-short.txt>
4. Create a new directory in HDFS by the following command

```
$ hdfs dfs -mkdir /user  
$ hdfs dfs -mkdir /user/wc  
$ hdfs dfs -mkdir /user/wc/input
```
5. Navigate to the folder the-hunger-games-short.txt file was downloaded and push the text file into HDFS by

```
$ hadoop fs -copyFromLocal the-hunger-games-short.txt /user/wc/input
```

Experiment:

1. Create a new java file and write the following code

```
import java.io.IOException;
import java.util.StringTokenizer;
import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Job;
import org.apache.hadoop.mapreduce.Mapper;
import org.apache.hadoop.mapreduce.Reducer;
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;
import org.apache.hadoop.util.GenericOptionsParser;
public class WordCount {
    public static class TokenizerMapper extends Mapper<Object,
Text, Text, IntWritable>{
        private final static IntWritable one = new IntWritable(1);
        private Text word = new Text();
        public void map(Object key, Text value, Context context)
throws IOException, InterruptedException {
            StringTokenizer itr = new
StringTokenizer(value.toString());
            while (itr.hasMoreTokens()) {
                word.set(itr.nextToken());
                context.write(word, one);
            }
        }
    }

    public static class IntSumReducer extends Reducer
<Text,IntWritable,Text,IntWritable> {
        private IntWritable result = new IntWritable();
        public void reduce(Text key, Iterable<IntWritable> values,
Context context)throws IOException, InterruptedException {
            int sum = 0;
```



```

        for (IntWritable val : values) {
            sum += val.get();
        }
        result.set(sum);
        context.write(key, result);
    }
}

public static void main(String[] args) throws Exception {
    Configuration conf = new Configuration();
    String[] otherArgs = new
    GenericOptionsParser(conf,args).getRemainingArgs();
    if (otherArgs.length < 2) {
        System.err.println("Usage: wordcount <in> <out>");
        System.exit(2);
    }
    Job job = new Job(conf, "word count");
    job.setJarByClass(WordCount.class);
    job.setMapperClass(TokenizerMapper.class);
    job.setCombinerClass(IntSumReducer.class);
    job.setReducerClass(IntSumReducer.class);
    job.setOutputKeyClass(Text.class);
    job.setOutputValueClass(IntWritable.class);
    FileInputFormat.addInputPath(job, new Path(otherArgs[0]));
    FileOutputFormat.setOutputPath(job, new Path(otherArgs[1]));
    System.exit(job.waitForCompletion(true) ? 0 : 1);
}
}

```

2. Compile the java program

```
$ javac -classpath $HADOOP_CLASSPATH WordCount.java
```

3. Create a jar file of all class file combined

```
$ jar -cvf wc.jar *.class
```

4. Execute the job in Hadoop

```
$ hadoop jar wc.jar WordCount /user/wc/input /user/wc/output
```

5. Copying output files from Hadoop to local directory

```
$ hadoop fs -copyToLocal /user/wc/output/
```

6. Viewing the output file

```
$ gedit part-r-00000
```

Sample Input and Output:

Sample content of the-hunger-games-short.txt

```
The Hunger Games
The Hunger Games 1by Suzanne Collins
PART I"THE TRIBUTES"
1.
When I wake up, the other side of the bed is cold. My fingers
stretch out, seeking Prims warmth but finding only the rough canvas
cover of the mattress. She must have had bad dreams and climbed in
with our mother. Of course, she did. This is the day of the
reaping. I prop myself up on one elbow. Theres enough light in the
bedroom to see them. My little sister, Prim, curled up on her side,
.
.
.
```

Sample Content of output file : part-r-00000

```
Awfully 1
Axminster 1
Ay 1
Ay! 2
Ay, 9
Ay. 4
Ayes 1
Azazel, 1
Azotes. 1
Aztec 1
Aztecs, 1
B 6
B, 2
B. 27
B.) 3
B., 2
B.A. 1
B.C. 1
BABES 1
BABY 1
BALANCE 1
BANTAM 1
BARBER: 1
.
.
.
```

Result:

Thus, a MapReduce application has been developed in java to count the number of occurrences of words in a dataset, executed on single node Hadoop cluster and responses have been verified.

Date:**Aim:**

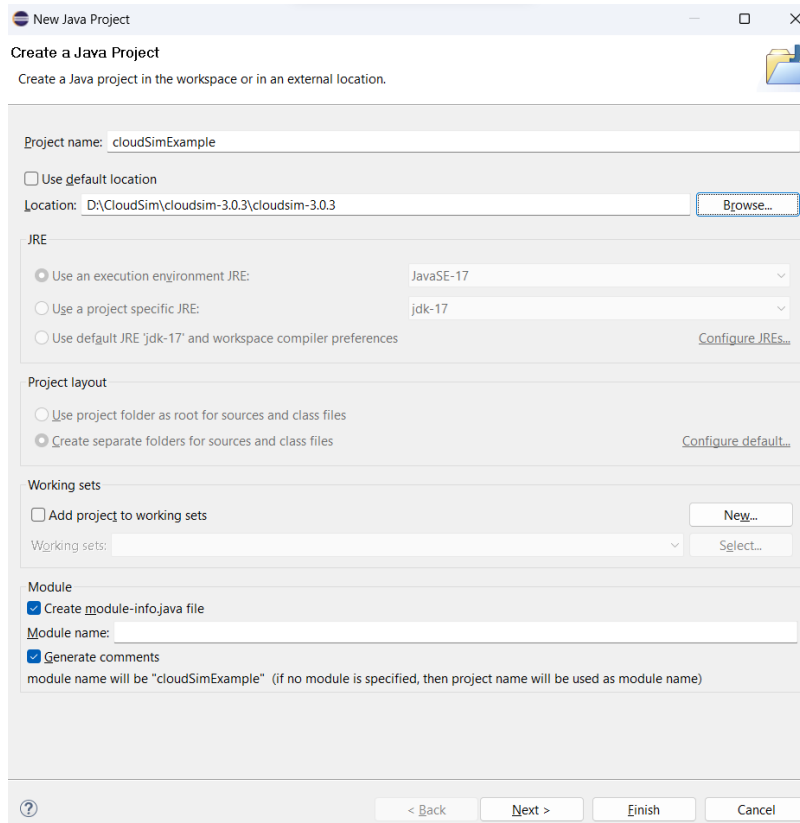
To install and configure CloudSim environment in Eclipse IDE.

Concept:

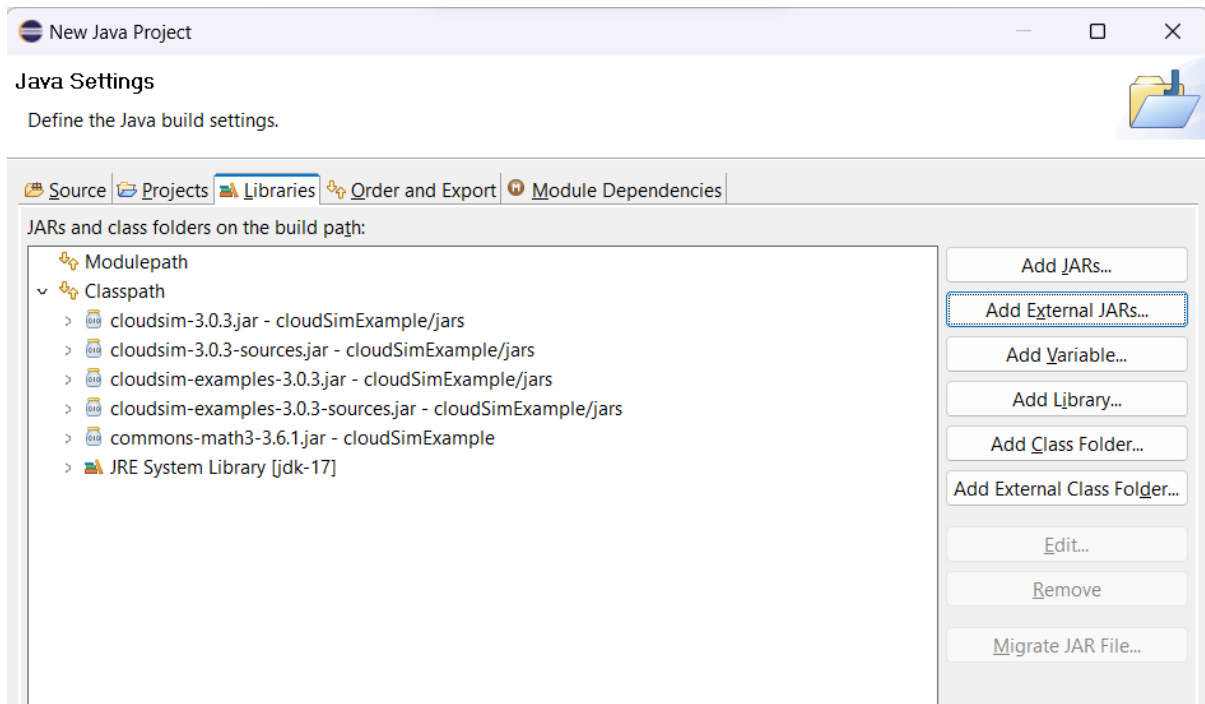
Eclipse IDE is a popular integrated development environment used for Java development and various other languages. Eclipse IDE for Java EE Developers includes additional tools for Java EE development. This lab will guide you through the installation process.

Experiment:

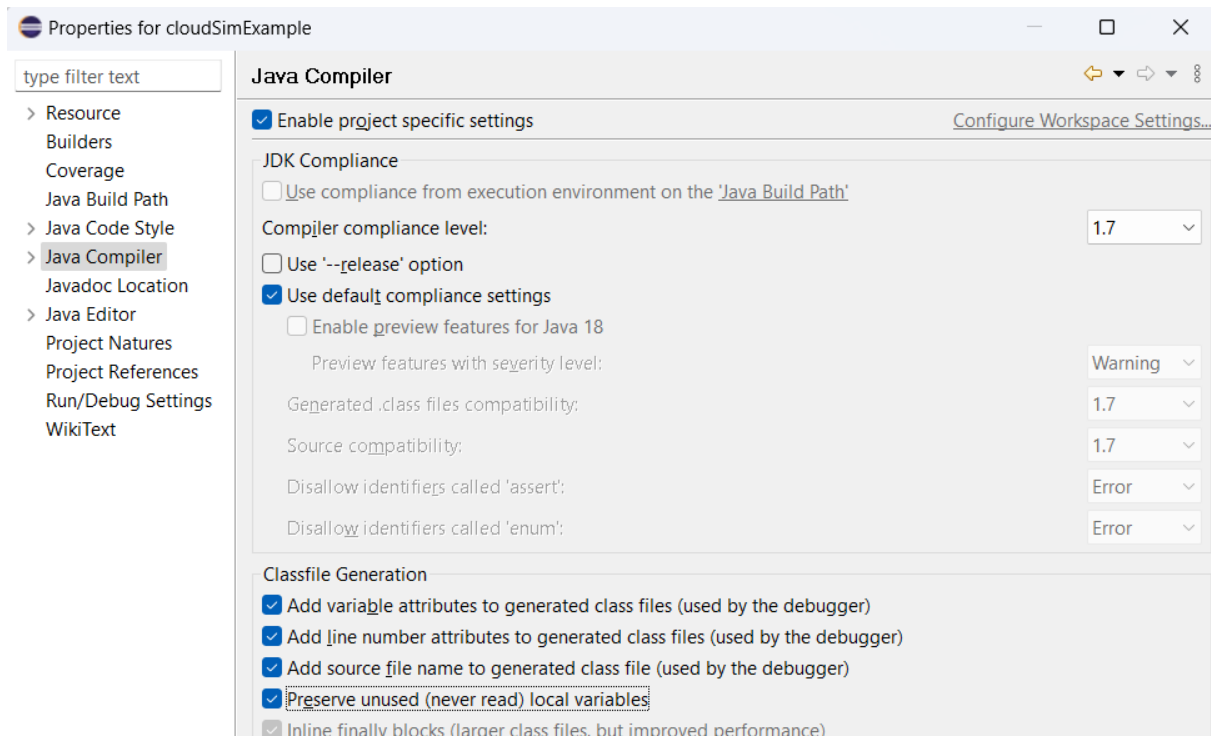
1. Step 1: Create a new folder in the D: directory
2. Step 2: Install Cloudsim
 - a. Go to the link:
<https://github.com/Cloudslab/cloudsim/releases/tag/cloudsim-3.0.3>
 - b. Download the package cloudsim- 3.0.3.zip in to the created directory.
 - c. Copy the file from the downloads folder to the created directory.
3. Step 3: Install Commons-math3
 - a. Go to the link
<https://archive.apache.org/dist/commons/math/binaries/commons-math3-3.6.1-bin.zip>
 - b. File will start to download.
 - c. Copy the file from the downloads folder to the created directory.
4. Step 4: Download Eclipse IDE for Java Developers from the given link
https://www.eclipse.org/downloads/download.php?file=/technology/epp/downloads/release/2024-03/R/eclipse-java-2024-03-R-win32-x86_64.zip
5. Step 5: Create a new Java project
6. Step 6: Enter the name of the project as CloudSimExample.
7. Step 7: Modify the location of the Project where CloudSim file is downloaded



8. Step 8: Click on Next, navigate to the Libraries tab, and click on 'Add External JARs.'
9. Step 9: Choose the installed commons-math3-3.6.1.jar file.



10. Step 10: Click on Apply and close.
11. Step 11: Right Click on CloudSimExample and click on properties.
12. Step 12: In the pop-up opened Select java compiler option.
13. Step 13: Choose the compiler compliance level to 1.7.
14. Step 14: Click on Apply and close.



15. Step 15: Navigate to example program given in cloudsims from the right side pane.
16. Step 16: Right click on the CloudSimExample1.java and select run as java application

Sample Output:

Result:

Thus the cloudsims is successfully installed and example program given in the package is executed successfully

Date:**Aim:**

The aim of this program is to simulate a cloud computing environment using CloudSim and demonstrate the allocation of virtual machines (VMs) in a datacenter.

Concept:

CloudSim is a cloud computing modeling and simulation framework. It includes components like Datacenter for VM allocation and cloudlet execution, Broker for VM management and cloudlet scheduling, VM representing virtual machines with CPU, RAM, and storage capacities. This framework facilitates the simulation of cloud computing infrastructures and services.

Experiment:

1. Step 1: Right click on CloudSimExample and click on new -> Class
2. Step 2: give package name as com.cloud
3. Step 3: Enter the name as CloudAllocation and click finish.

New Java Class

Java Class
Create a new Java class.

Source folder: cloudSimExample/examples Browse...

Package: com.cloud Browse...

☐ Enclosing type: Browse...

Name: CloudAllocation

Modifiers: ☒ public ☐ package ☐ private ☐ protected
☐ abstract ☐ final ☐ static

Superclass: java.lang.Object Browse...

Interfaces: Add... Remove

Which method stubs would you like to create?
☐ public static void main(String[] args)
☐ Constructors from superclass
☐ Inherited abstract methods

Do you want to add comments? (Configure templates and default value [here](#))
☐ Generate comments

Finish Cancel

4. Step 4: Write the following program in the CloudAllocation.java package com.cloud;

```
import org.cloudbus.cloudsim.*;
```

```

import org.cloudbus.cloudsim.provisioners.BwProvisionerSimple;
import org.cloudbus.cloudsim.provisioners.PeProvisionerSimple;
import org.cloudbus.cloudsim.provisioners.RamProvisionerSimple;
import java.text.DecimalFormat;
import java.util.ArrayList;
import java.util.Calendar;
import java.util.LinkedList;
import java.util.List;
public class CloudAllocation {
    public static void main(String[] args) {
        Log.println("Starting CloudSimVMAllocationExample...");
        try {
            int numUsers = 1;
            Calendar calendar = Calendar.getInstance();
            boolean traceFlag = false;
            CloudSim.init(numUsers, calendar, traceFlag);
            Datacenter datacenter =
createDatacenter("Datacenter_0");
            DatacenterBroker broker = createBroker();
            int brokerId = broker.getId();
            List<Vm> vmList = new ArrayList<>();
            Vm vm1 = new Vm(0, brokerId, 1000, 1, 512, 1000,
10000, "Xen", new CloudletSchedulerTimeShared());
            Vm vm2 = new Vm(1, brokerId, 1500, 1, 768, 1500,
15000, "Xen", new CloudletSchedulerTimeShared());
            vmList.add(vm1);
            vmList.add(vm2);
            broker.submitVmList(vmList);
            List<Cloudlet> cloudletList = new ArrayList<>();
            Cloudlet cloudlet1 = new Cloudlet(0, 400000, 1, 300,
300, new UtilizationModelFull(), new UtilizationModelFull(), new
UtilizationModelFull());

```

```

        Cloudlet cloudlet2 = new Cloudlet(1, 200000, 1, 300,
300, new UtilizationModelFull(), new UtilizationModelFull(), new
UtilizationModelFull());

        cloudletList.add(cloudlet1);
        cloudletList.add(cloudlet2);
        broker.submitCloudletList(cloudletList);
        CloudSim.startSimulation();
        CloudSim.stopSimulation();

        List<Cloudlet> newList =
broker.getCloudletReceivedList();
        printCloudletList(newList);
        Log.println("CloudSimVMAllocationExample
finished!");
    } catch (Exception e) {
        e.printStackTrace();
        Log.println("An error occurred");
    }
}

private static Datacenter createDatacenter(String name) {
    List<Host> hostList = new ArrayList<>();
    List<Pe> peList = new ArrayList<>();
    peList.add(new Pe(0, new PeProvisionerSimple(1000)));
    peList.add(new Pe(1, new PeProvisionerSimple(1500)));
    int hostId = 0;
    int ram = 2048;
    long storage = 1000000;
    int bw = 10000;

    hostList.add(new Host(hostId, new
RamProvisionerSimple(ram), new BwProvisionerSimple(bw), storage,
peList, new VmSchedulerTimeShared(peList)));

    String arch = "x86";
    String os = "Linux";
    String vmm = "Xen";
    double time_zone = 10.0;

```



```

        double cost = 3.0;
        double costPerMem = 0.05;
        double costPerStorage = 0.001;
        double costPerBw = 0.0;
        LinkedList<Storage> storageList = new LinkedList<>();

        DatacenterCharacteristics characteristics = new
DatacenterCharacteristics(arch, os, vmm, hostList, time_zone,
cost, costPerMem, costPerStorage, costPerBw);
        try {
            return new Datacenter(name, characteristics, new
VmAllocationPolicySimple(hostList), storageList, 0);
        } catch (Exception e) {
            e.printStackTrace();
            return null;
        }
    }

    private static DatacenterBroker createBroker() {
        try {
            return new DatacenterBroker("Broker");
        } catch (Exception e) {
            e.printStackTrace();
            return null;
        }
    }

    private static void printCloudletList(List<Cloudlet> list) {
        Log.println();
        for (Cloudlet cloudlet : list) {
            if (cloudlet.getCloudletStatus() == Cloudlet.SUCCESS)
{
                Log.println("Cloudlet ID: " +
cloudlet.getCloudletId() + "\tSUCCESS");
            } else {
                Log.println("Cloudlet ID: " +
cloudlet.getCloudletId() + "\tFAILED");
            }
        }
    }

```

```
    }  
  }  
}
```

Sample Output:

Cloudlet Id: 0	SUCCESS
Cloudlet Id: 1	SUCCESS

Result:

The program successfully simulates the allocation of VMs in a datacenter and executes cloudlets on these VMs.

Date:**Aim:**

The aim of this program is to simulate a FCFS scheduling algorithm in cloudsim based on their arrival time.

Concept:

The FCFS scheduling algorithm follows a non-preemptive approach, where tasks are executed in the order they arrive. When a task arrives, it is placed at the end of the queue and gets processed only when all earlier tasks are completed

Experiment:

1. Step 1: Right click on CloudSimExample and click on new -> Class
2. Step 2: give package name as com.cloud
3. Step 3: Enter the name as CloudScheduling and click finish.
4. Step 4: Write the following program in the CloudScheduling.java

```
package com.cloud;

import java.text.DecimalFormat;

import java.util.ArrayList;

import java.util.Calendar;

import java.util.LinkedList;

import java.util.List;

import org.cloudbus.cloudsim.*;

import org.cloudbus.cloudsim.core.CloudSim;

import org.cloudbus.cloudsim.provisioners.BwProvisionerSimple;

import org.cloudbus.cloudsim.provisioners.PeProvisionerSimple;

import org.cloudbus.cloudsim.provisioners.RamProvisionerSimple;

public class CloudScheduling {

    private static List<Cloudlet> cloudletList;

    private static List<Vm> vmList;

    public static void main(String[] args) {

        Log.println("Starting CloudSimExampleFCFS...");

        try {

            int num_user = 1;
```

```

Calendar calendar = Calendar.getInstance();
boolean trace_flag = false;
CloudSim.init(num_user, calendar, trace_flag);
Datacenter datacenter0 = createDatacenter("Datacenter_0");
DatacenterBroker broker = createBroker();
int brokerId = broker.getId();
vmList = new ArrayList<>();
int vmid = 0;
int mips = 1000;
long size = 10000;
int ram = 512;
long bw = 1000;
int pesNumber = 1;
String vmm = "Xen";

Vm vm = new Vm(vmid, brokerId, mips, pesNumber, ram, bw,
size, vmm, new CloudletSchedulerTimeShared());

vmList.add(vm);

broker.submitVmList(vmList);

cloudletList = new ArrayList<>();
int id = 0;
long length = 400000;
long fileSize = 300;
long outputSize = 300;

UtilizationModel utilizationModel = new UtilizationModelFull();

Cloudlet cloudlet1 = new Cloudlet(id, length, pesNumber, fileSize,
outputSize, utilizationModel, utilizationModel, utilizationModel);

cloudlet1.setUserId(brokerId);

cloudlet1.setVmId(vmid);

cloudletList.add(cloudlet1);

Cloudlet cloudlet2 = new Cloudlet(id + 1, length / 2, pesNumber,
fileSize, outputSize, utilizationModel, utilizationModel, utilizationModel);

cloudlet2.setUserId(brokerId);

cloudlet2.setVmId(vmid);

```

```

cloudletList.add(cloudlet2);

broker.submitCloudletList(cloudletList);

CloudSim.startSimulation();
CloudSim.stopSimulation();

List<Cloudlet> newList = broker.getCloudletReceivedList();
printCloudletList(newList);

Log.println("CloudSimExampleFCFS finished!");
} catch (Exception e) {
    e.printStackTrace();
    Log.println("Unwanted errors happen");
}
}

private static Datacenter createDatacenter(String name) {
    List<Host> hostList = new ArrayList<>();
    List<Pe> peList = new ArrayList<>();
    int mips = 1000;
    peList.add(new Pe(0, new PeProvisionerSimple(mips)));
    int hostId = 0;
    int ram = 2048;
    long storage = 1000000;
    int bw = 10000;

    hostList.add(new Host(hostId, new RamProvisionerSimple(ram), new
BwProvisionerSimple(bw), storage, peList, new
VmSchedulerTimeShared(peList)));

    String arch = "x86";

```

```

String os = "Linux";
String vmm = "Xen";
double time_zone = 10.0;
double cost = 3.0;
double costPerMem = 0.05;
double costPerStorage = 0.001;
double costPerBw = 0.0;
LinkedList<Storage> storageList = new LinkedList<>();

DatacenterCharacteristics characteristics = new
DatacenterCharacteristics(arch, os, vmm, hostList, time_zone, cost,
costPerMem, costPerStorage, costPerBw);

Datacenter datacenter = null;
try {
    datacenter = new Datacenter(name, characteristics, new
VmAllocationPolicySimple(hostList), storageList, 0);
} catch (Exception e) {
    e.printStackTrace();
}

return datacenter;
}

private static DatacenterBroker createBroker() {
    DatacenterBroker broker = null;
    try {
        broker = new DatacenterBroker("Broker");
    } catch (Exception e) {
        e.printStackTrace();
        return null;
    }
    return broker;
}

```

```

    }

    private static void printCloudletList(List<Cloudlet> list) {
        int size = list.size();
        Cloudlet cloudlet;
        for (int i = 0; i < size; i++) {
            cloudlet = list.get(i);
            if (cloudlet.getCloudletStatus() == Cloudlet.SUCCESS) {
                Log.println("Cloudlet ID: " + cloudlet.getCloudletId() +
"\tSUCCESS");
            } else {
                Log.println("Cloudlet ID: " + cloudlet.getCloudletId() +
"\tFAILED");
            }
        }
    }
}

```

Sample Output:

Cloudlet Id: 0	SUCCESS
Cloudlet Id: 1	SUCCESS

Result:

The CloudSim program for simulating the First-Come, First-Served (FCFS) scheduling algorithm has been successfully executed.