Experiment:01

Aim: Workflow within business activity: Edrawmax

Theory: A workflow diagram is a visual layout of a process, project or job in the formof a flow chart. It's a highly effective way to impart the steps more easily in a businessprocess, how each one will be completed, by whom and in what sequence.

Workflow diagrams are commonly used to do the following:

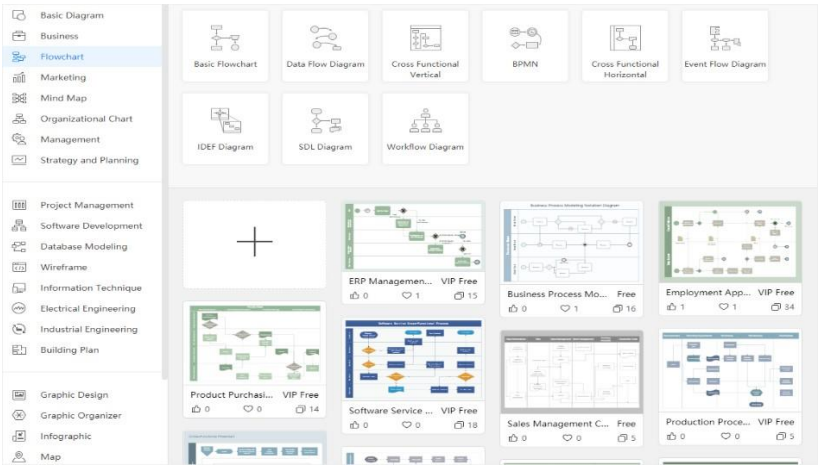
- Employ a holistic visual of business processes and information flows
- Equip employees with a better understanding of their roles and responsibilities
- Reveal process redundancies and bottlenecks
- Safeguard against risk

You can create a workflow diagram manually or use software automation to create one more easily. There are many software options on the market today that enable you to make workflow diagrams. Likewise, many organizations use software to automate the workflow process.

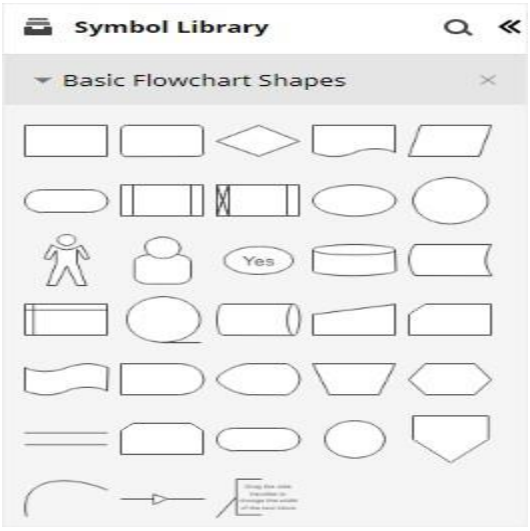
Procedure:

Step 1

When you're into **EdrawMax**, you can begin with a blank drawing page and make a new flowchart from scratch. However, if you don't want to waste too much time in building the structure of the flowchart, you can open a pre-made flowchart template in our template library and customize it thereafter.



Step 2: Click the icon to open Symbol Library and find Basic Flowchart Shapes in the diagram type list to add this category or other symbol category into the left library pane.

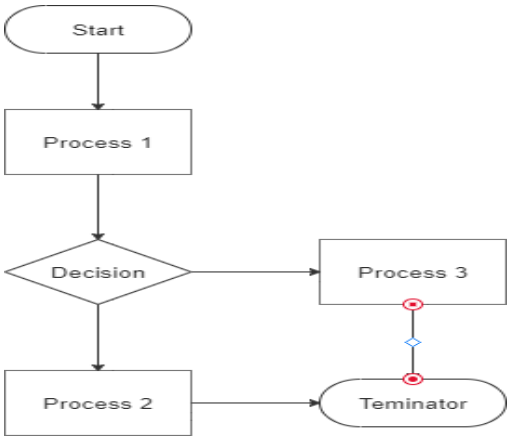


Step 3

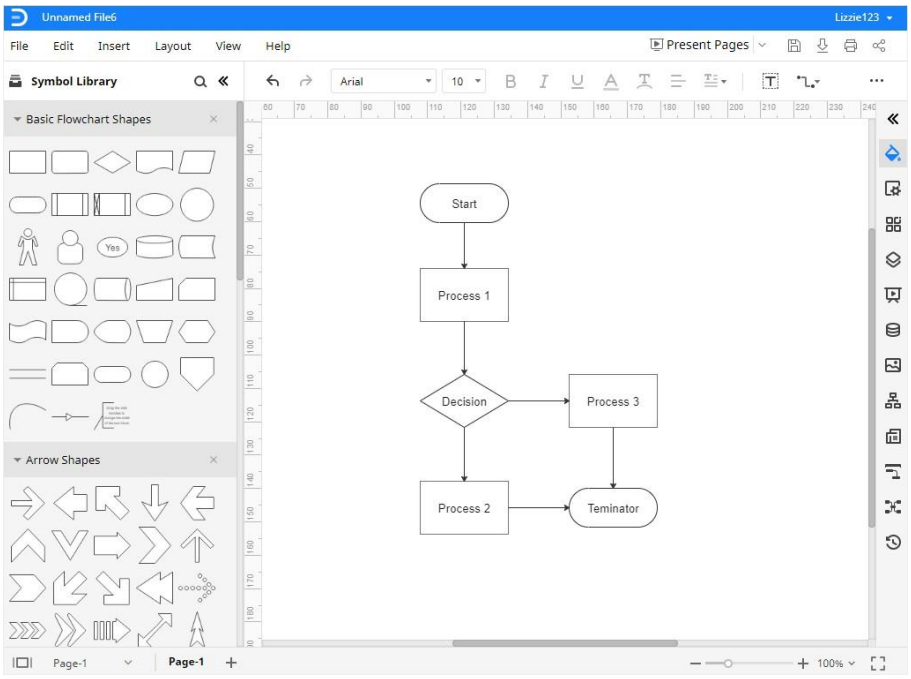
Click on **Start/Teminator** symbol, drag and drop it onto the drawing page. Then add other symbols that you need to make the main structure via the same method.

Step 4

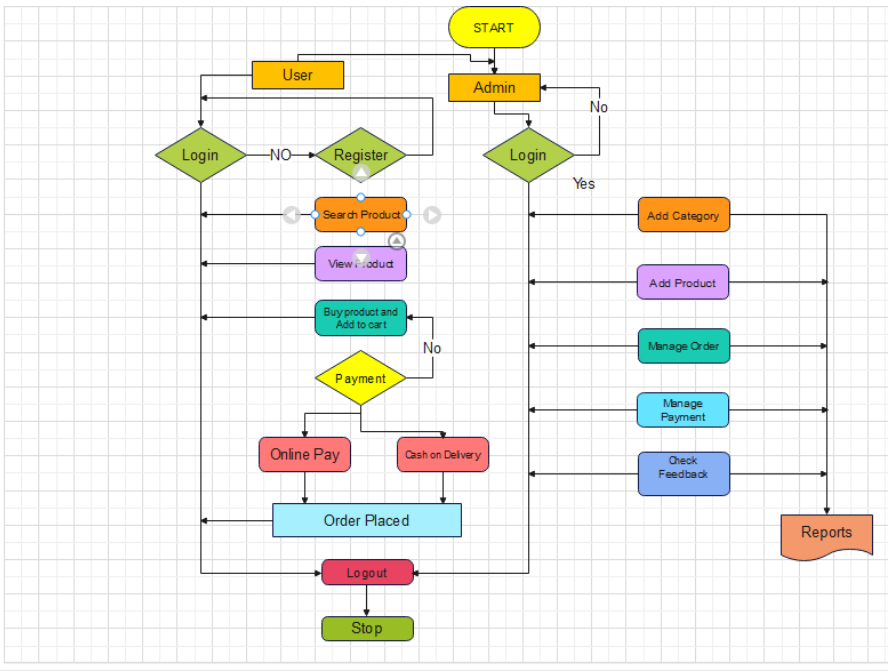
Click **Connector** button on the toolbar and select a type of connector to add connection lines between symbols. Usually, the connectors will be attached to the connection points on the flowchart symbols automatically.



Step 5 : After you finish adding symbols and connectors, the complete flowchart is shown on the drawing page. Then you can customize the flowchart with plentiful formatting tools in EdrawMax.



Final Workflow diagram for a E-commerce Website:



Experiment:02

Aim: How to create product backlog for project and user story creation.

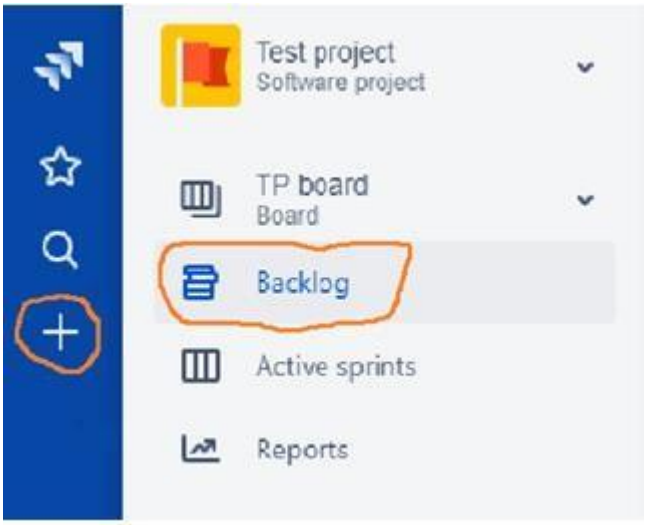
Theory: A product backlog is **a prioritized list of work for the development team that is derived from the roadmap and its requirements**. The most important items are shown at the top of the product backlog so the team knows what to deliver first.

A user story is **an informal, general explanation of a software feature written from the perspective of the end user or customer**. The purpose of a user story is to articulate how a piece of work will deliver a particular value back to the customer.

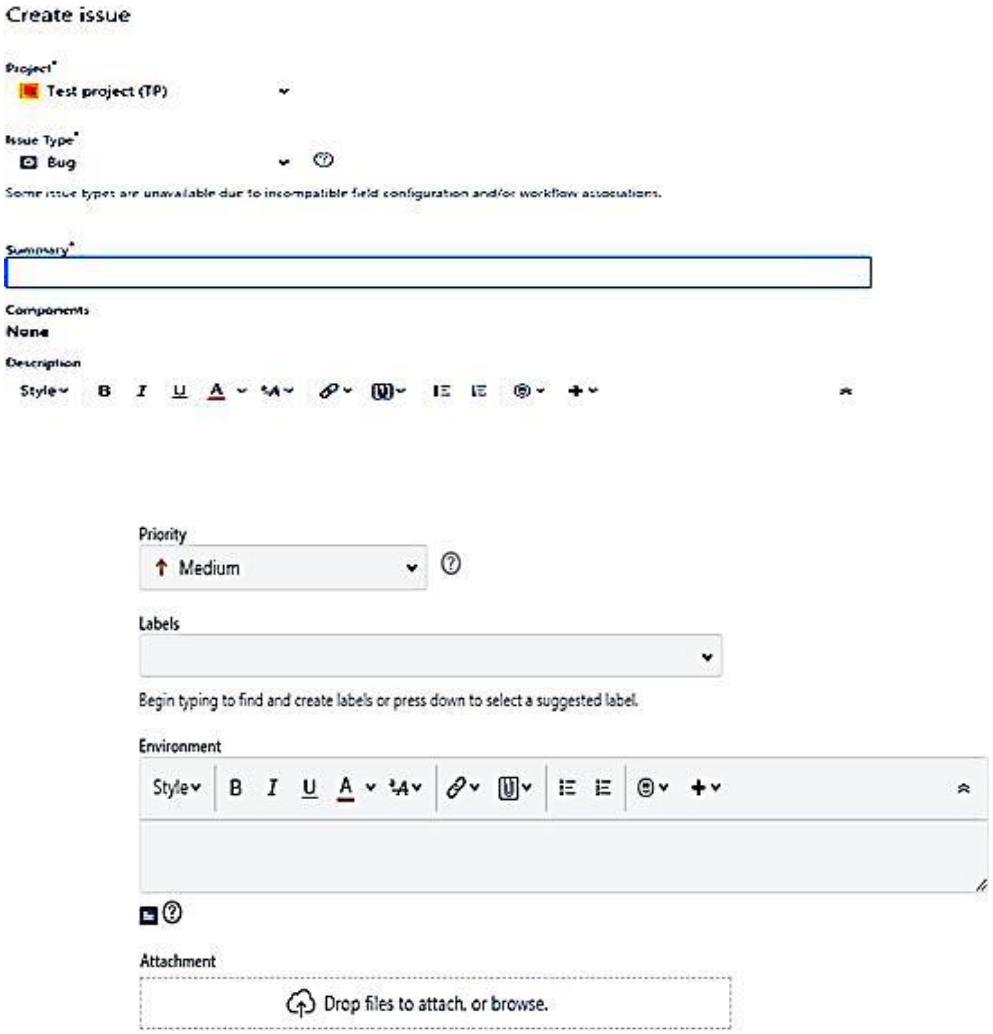
Procedure:

Creating Product Backlog in Jira:

Step 1: Click on the "+" sign at the JIRA dashboard and click on "Backlog" Icon.



Step 2: Fill the details in the create issue page i.e. select the name of the project, select Issue/task Type(It may Bug or Epic or Story), Write down Summary for Issue/task(Overall description), Description about it, Priority level(Low or Medium or High), Labels, Environment, Attachment related task, Linked Issues, Assigned To, Epic Link, Sprint, etc.

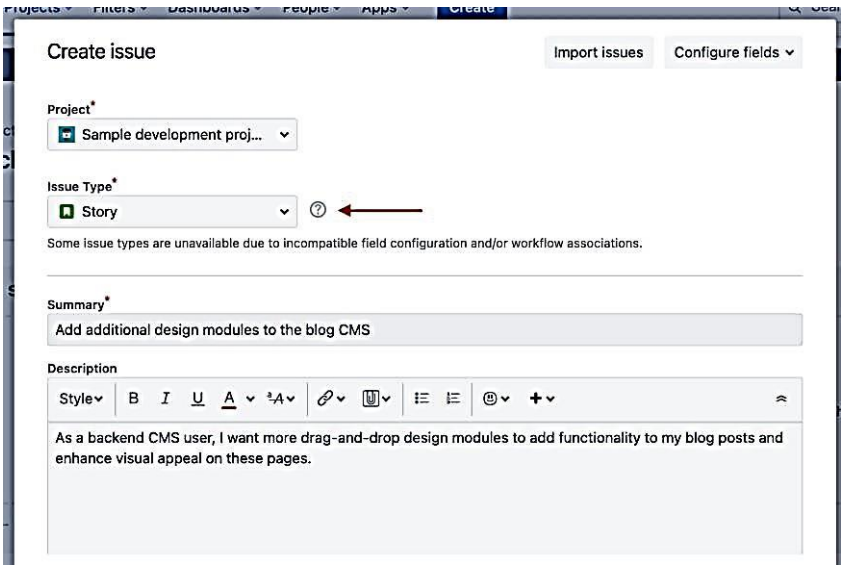


Step 3: Click on the "Create" button to create a new Backlog.

Creation of UserStory in Jira:

To add a story to your backlog:

- Navigate to the 'Create' screen in your Jira Scrum or Kanban project
- Create a new Jira issue and select the appropriate project from the dropdownmenu
- Make sure 'Story' is selected as the issue type. This will be the default setting
- Follow your organisation's guidelines for formatting your story. Don't worry ifyou have no guidelines to follow - we give specifics below!
- Your story will then go into the backlog to be assigned and actioned by theproject manager, project leader, productowner or other relevant stakeholders



Experiment:03

Aim: Create UI/UX Design for created user Stories (Wire Framing).

Theory: UX design involves designing the app's user experience, mainly how they interact with it. Is the navigation intuitive, or is it nonsense? Is the experience easy and smooth, or is it confusing and clunky? Do users feel that they are getting things done, or are they flailing around hopelessly? Is the app enjoyable?

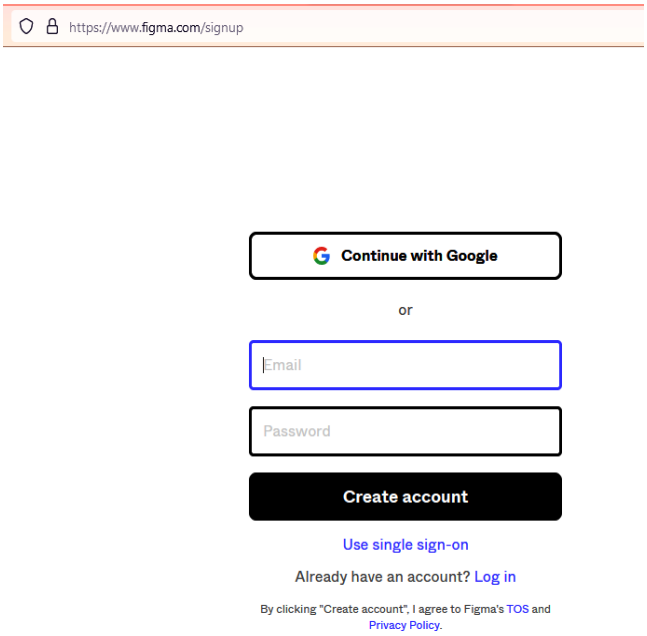
These are the kinds of questions a UX designer must face. UX designers are responsible for the app's structure, functionality, organization, and how its different elements work together. Ultimately, the UX designer's job is to give the user a good experience with the app.

UI design deals with the user interface and the app's graphical layout. This design covers the buttons users click, the text they see, text entry fields, images, sliders, etc. UI covers any item the user interacts with, including transitions, screen layout, interface animations, and every micro-interaction. Any part of the app that the user sees and interacts with must be designed, and that's the UI designer's job.

Procedure:

How to create User Interface for Sign-in page using Figma.

First, we are going to copy and paste the desktop splash page. Why not create it from scratch? I want all the desktop screens(rectangles) to be the same size. And each screenis going to need labels. I don't want to keep doing the same task over and over again. But feel free to do it that way. Do whatever is comfortable and intuitive for you.



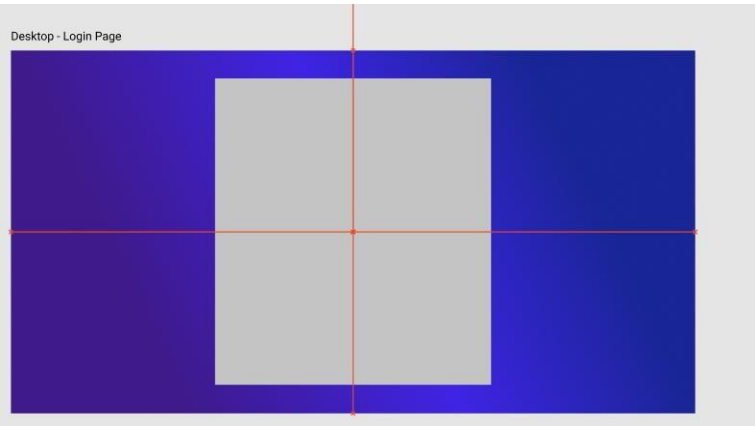
Above is a duplicate of the Splash Page for desktop. I renamed the components to acknowledge its for the login page, and I deleted the logo.

So for the login page, I want the basics. I want two input boxes and a submit button. I want the logo somewhere in there. I want a welcome message and a prompt for the user to log in. Also, maybe "Forgot Password?" and "Sign Up" anchor links, since every website has that. And if this isn't intuitive for you, I like checking out websites Ilove. What are the elements and features that most login pages have? Look around.

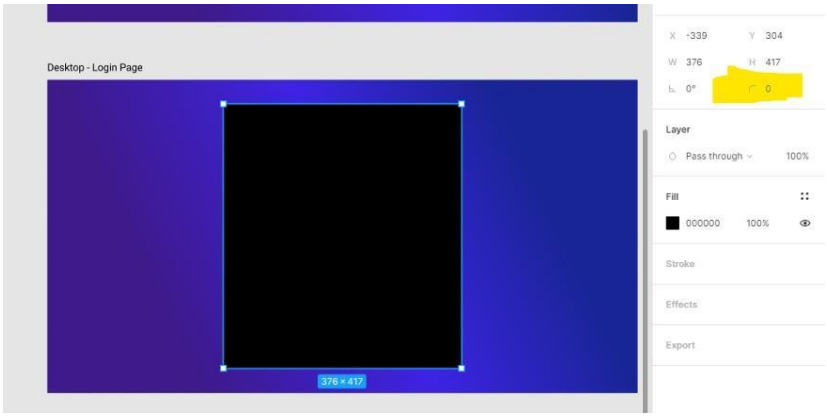
For the actual design, I think I want the whole login form contained within a containercentered in the middle. And I want that with rounded edges. Right now, I'm envisioning a transparent black container, but I could also see it being solid white or black. I will test all of it out — and that's the whole point of Figma.

With design tools like Figma, we can just design it here. We don't have to create it with HTML & CSS. We can design it here, test out multiple designs, pick one we love,and then actually create it.

So let's get started.



Here, I created a new rectangle that is going to house my login form. Using those red lines — which Figma offers as a positioning tool — I am centering the rectangle on thepage.

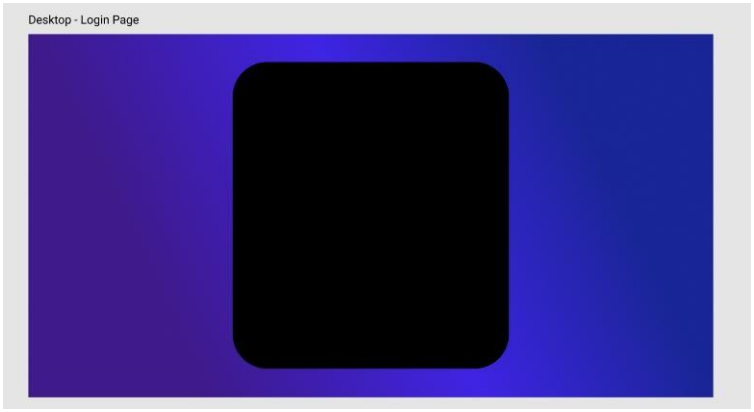


Round the edges using this tool in Design Panel

After, I changed the fill color to black. (*Remember, you must select the rectangle layer in order to access the design panel for the rectangle)*

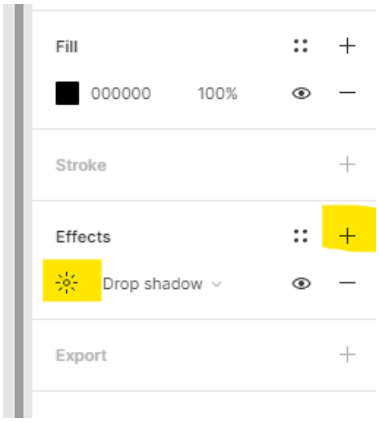
Now, I want to round out the edges. Unlike images, Figma shapes have to be rounded using the Design panel (on the right-hand side). I highlighted it for you in the screenshot above.

All we are going to do is click the 0. Instead of writing a number — which you can definitely do — we are going to just use the arrow keys. We are going to hold down the up-arrow. This will increment the number until you release the up-arrow key. I prefer doing it this way, instead of using numbers, so I can stop when I think it looks okay.

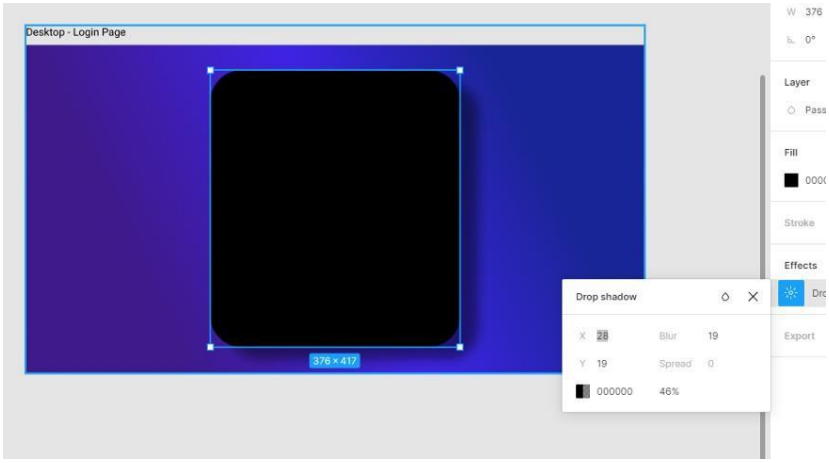


I stopped at a corner radius of 47, as shown above. If you want to go for a higher or lower number, go for it.

Next, I want to add a drop-shadow to the rectangle so there’s depth and dimension.



Click the + to add a new effect (in the Effects category). By default, the effect will be adrop shadow. Click the sun icon to change the customizations, as shown below.

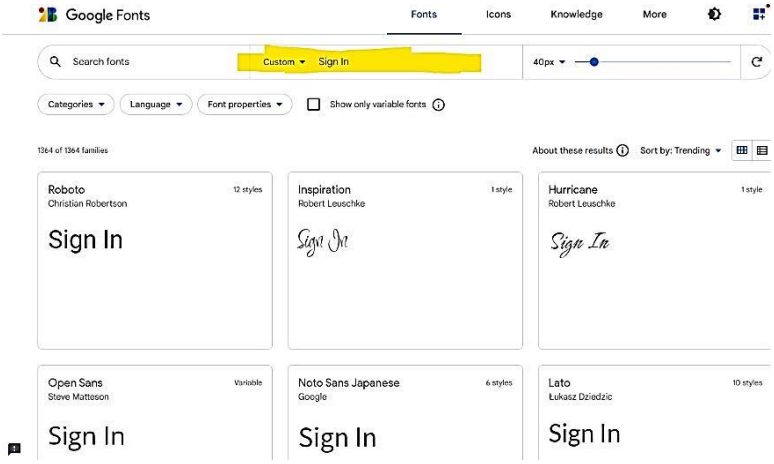


The x-value moves it along the x-axis — which means left to right. The y-value movesit up and down. And for this, feel free to use the up and down arrows to raise and lowerthe values. Play around with the numbers and see what happens.

This is our current product. I think I like the black so we are going to march forward. Now, I am going to introduce all the text elements.

To start, we are going to create the “Sign In” prompt. After selecting the text, we are going to access the design panel to change the font and the font size. This is all highlighted in the screenshot above.

If you’re a more visual person, head over to Google Fonts.



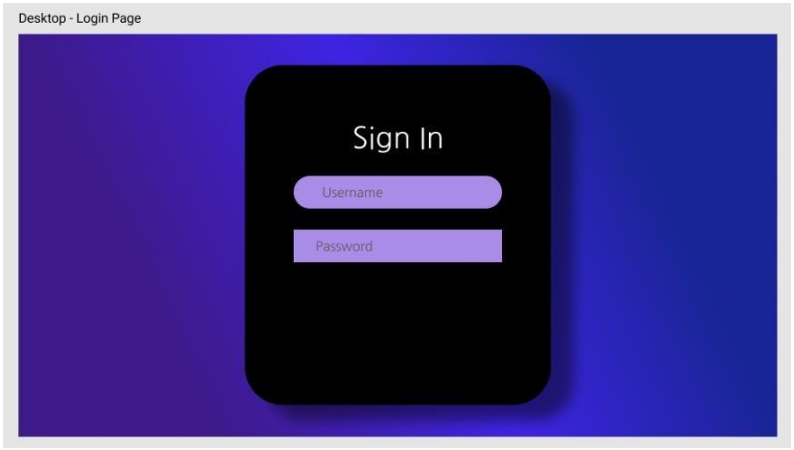
In the Custom text-box, as shown in the screenshot above, type in your custom message which is “Sign In”. Scroll until you see a font you think would look great in your project. All these fonts available in Google Fonts are available in Figma.

This is what mine looks like. This is the font NanumGothic with a size of 36. I like theskinny-thin look

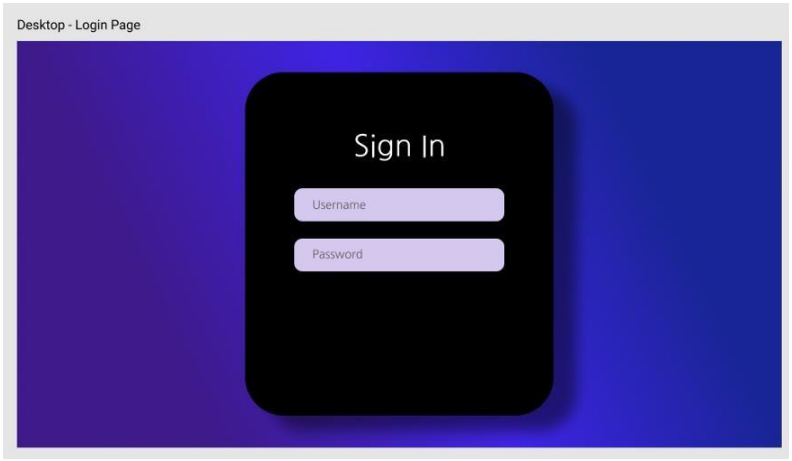
of the characters — and I think it looks modern and elegant. Use whatever font you want.

Moving on, I'm going to add two input boxes: one for username and one for password. And I will indicate what each box is for using placeholder text.

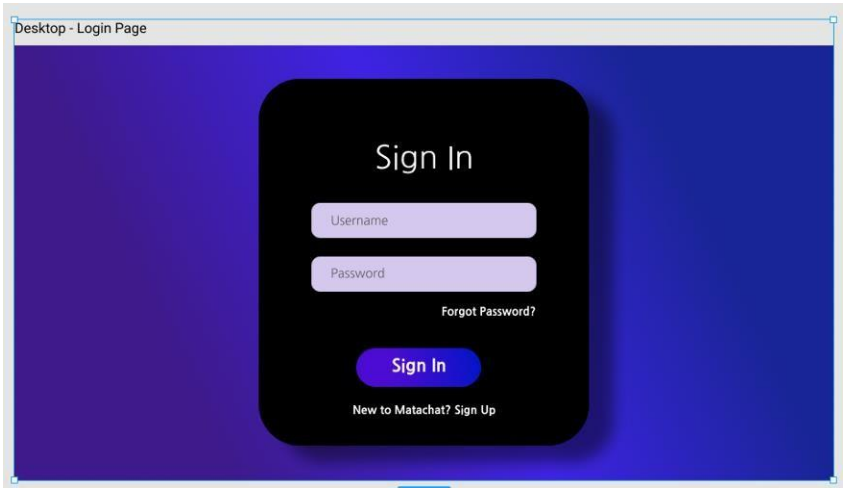
And to create these input boxes, it's nothing special. We are still using shapes. Remember, this is a design tool. It doesn't actually need to function.



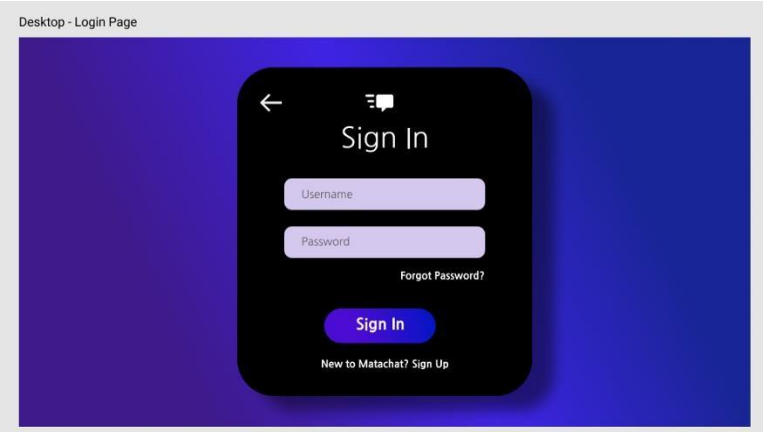
Do we like the rounded input boxes or the squarish ones? And do we like the pink background color?



This is what we have: slightly rounded corners, faint and tiny placeholder text, and a nearly-white pink background color. I'm sold. One note about this: Make sure your username and password text isn't the same height as the text box. Whitespace is a power. Whitespace is gorgeous. Use whitespace.



I threw in a sign-in button, a “Forgot Password?” anchor tag, and a sign-up anchor tag.Now that I look at it, a skinny back button would look great at the top left.



Experiment:04

Aim: Git Client installation and Setup and perform basic local Git operations.

- Creating repository.
- Cloning a repository.
- Staging and Committing changes.
- Viewing the history of the changes.
- Git branching and merging.

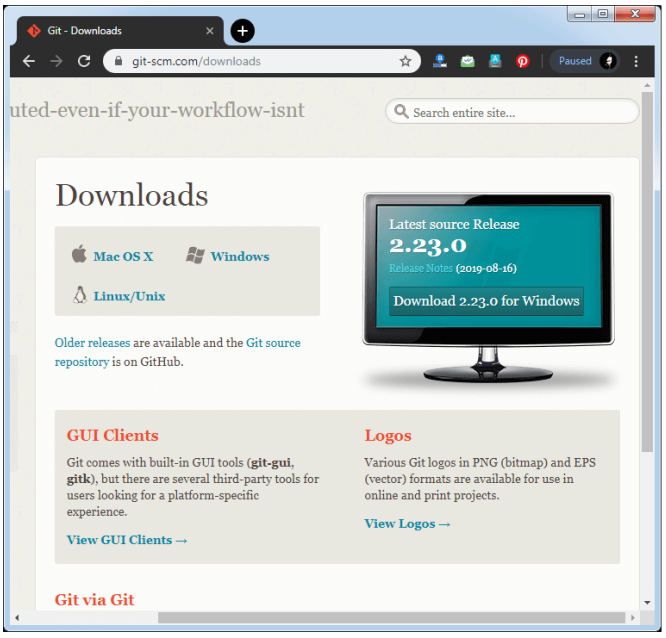
Theory: The Git GUI client is **a tool that allows the developer to work with this version control system in a visual mode**. It does not require writing commands manually, offering a convenient graphical interface with the in-built options. This way, one can perform the tasks faster and in a more comfortable manner.

Procedure:

- Steps to Install and Setup Git Client:

Step1

To download the Git installer, visit the Git's official site and go to download page. The link for the download page is https://git-scm.com/downloads. The page looks like as



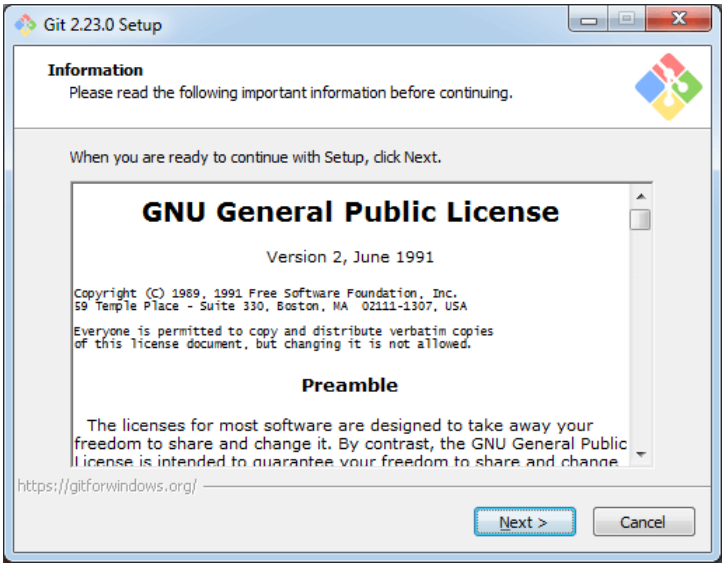
Click on the package given on the page as **download 2.23.0 for windows**. The download will start after selecting the package.

Now, the Git installer package has been downloaded.

Install Git

Step2

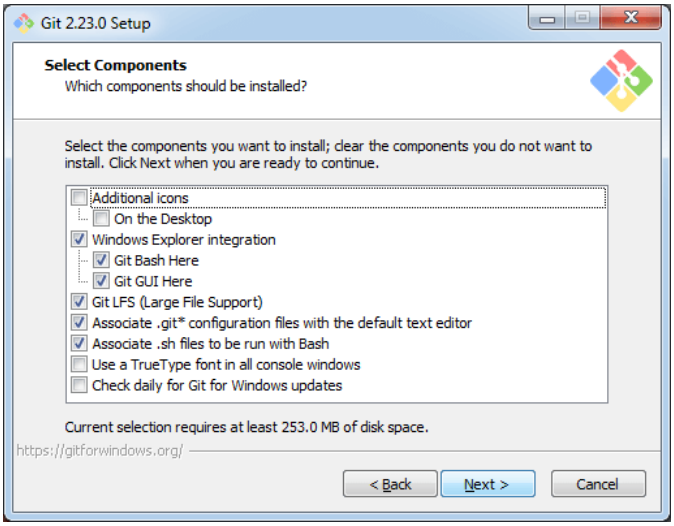
Click on the downloaded installer file and select **yes** to continue. After the selecting **yes** the installation begins, and the screen will look like as



Click on **next** to continue.

Step3

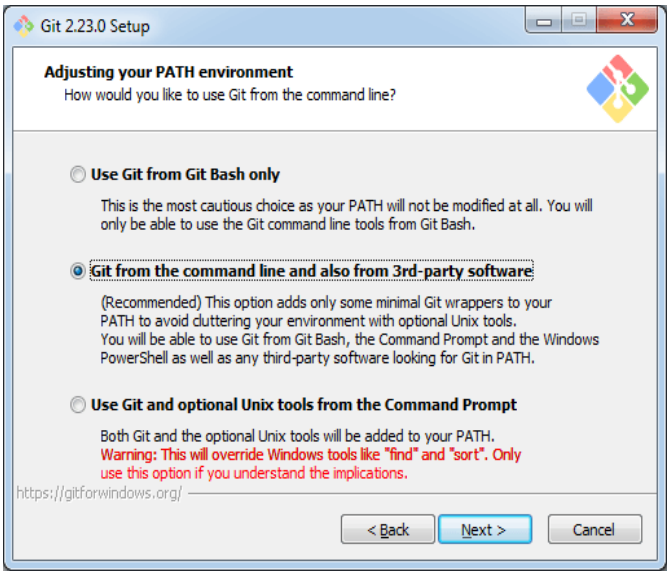
Default components are automatically selected in this step. You can also choose yourrequired part.



Click next to continue.

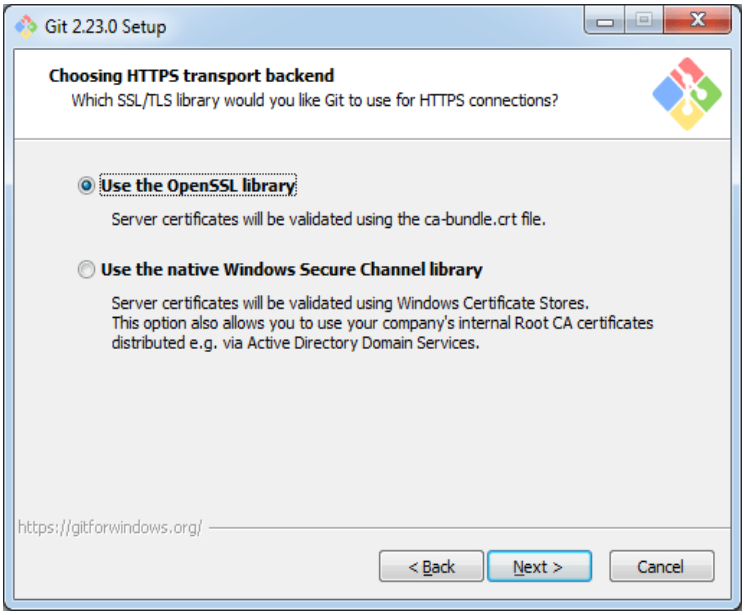
Step4

The default Git command-line options are selected automatically. You can choose yourpreferred choice. Click **next** to continue.



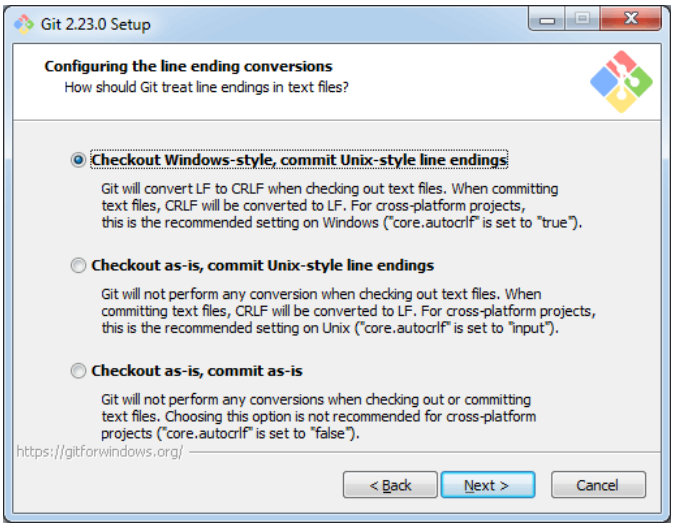
Step5

The default transport backend options are selected in this step. Click **next** to continue.



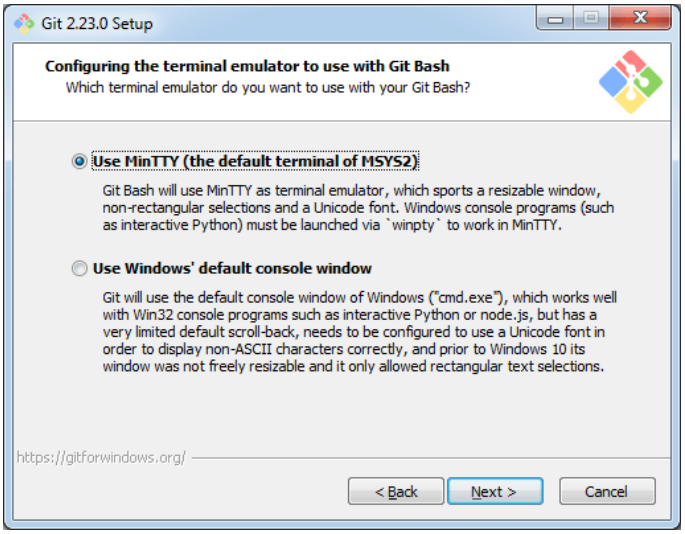
Step6

Select your required line ending option and click next to continue.



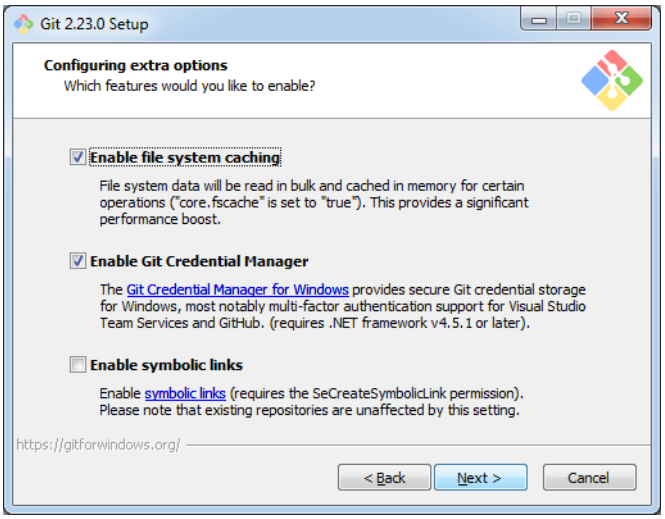
Step7

Select preferred terminal emulator clicks on the **next** to continue.



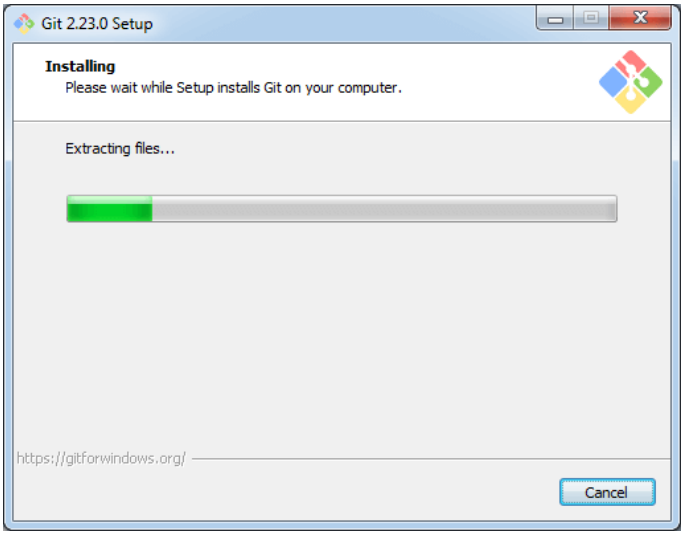
Step8

This is the last step that provides some extra features like system caching, credentialmanagement and symbolic link. Select the required features and click on the **next** option.

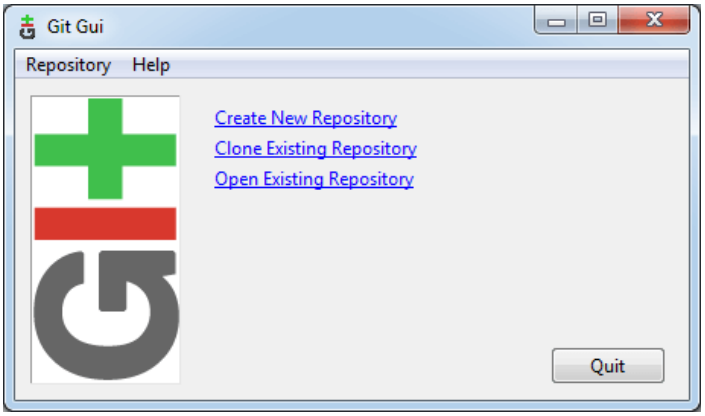


Step9

The files are being extracted in this step.



Therefore, The Git installation is completed. Now you can access the **Git Gui** and **Git Bash**.The **Git Gui** looks like as



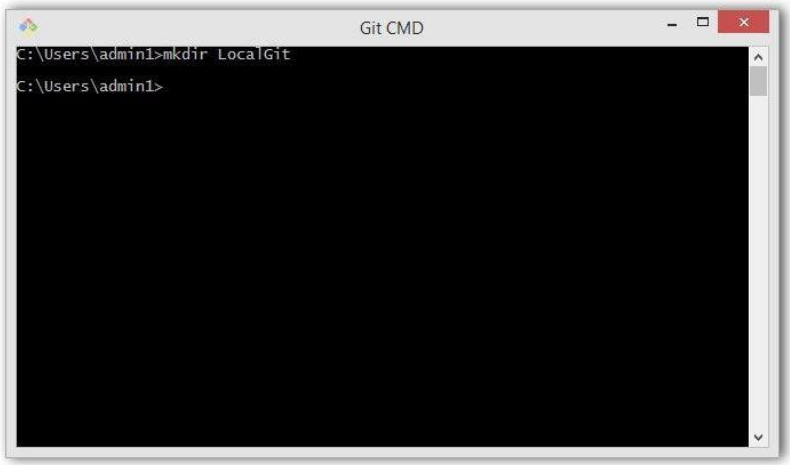
It facilitates with three features.

- Create New Repository
- Clone Existing Repository
- Open Existing Repository

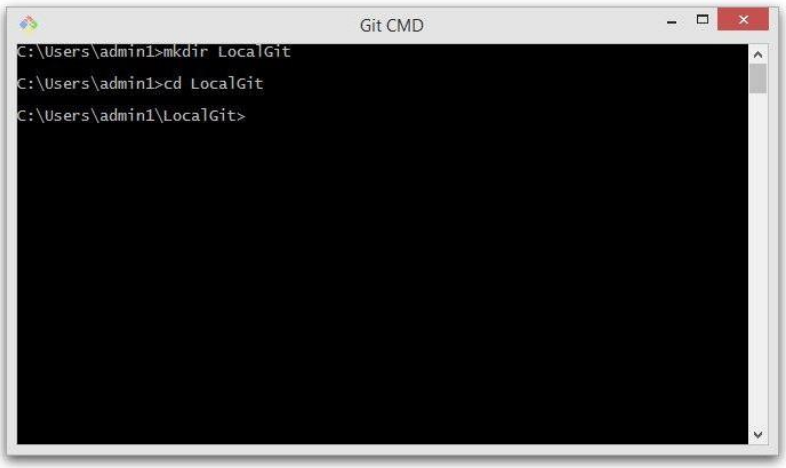
- Creating a repository:

Step 1: Create a New Project/Folder - Now that we have our command line tool open, let us ***create a project folder***. Creating a project folder with a good name is very important. If you have a Good and Relevant name of your project it will become easier for you to identify and relate to the project when you come back toit in the future.

Command to create a folder on a Windows and Mac system is mkdir <folderName>. Where folder name is the project name. Let us name our firstproject ***LocalGit***. With this name, the command becomes: mkdir LocalGit

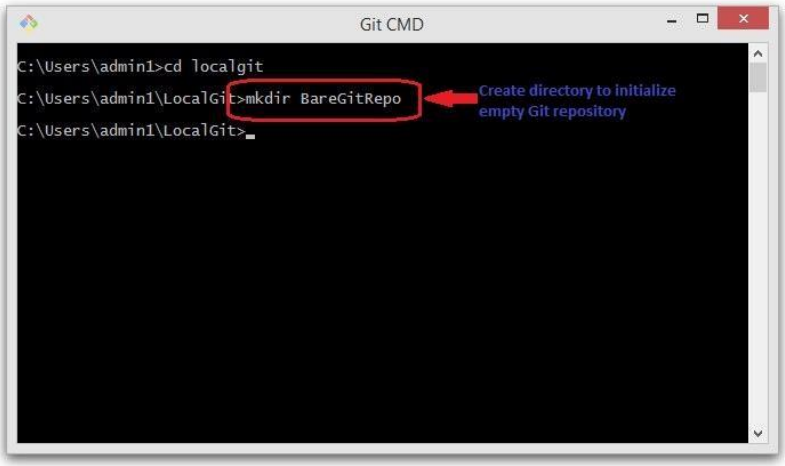


Navigate to this folder by using the command cd LocalGit, both on Windows andMac systems.

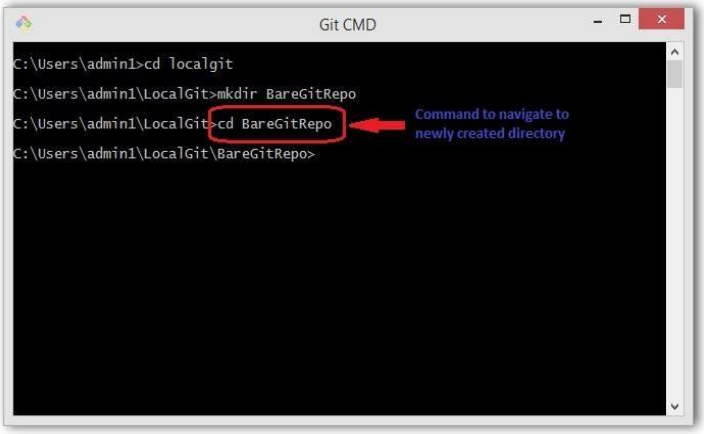


Now you are inside the repository folder where we'll create our Git repositories using the three approaches mentioned above.

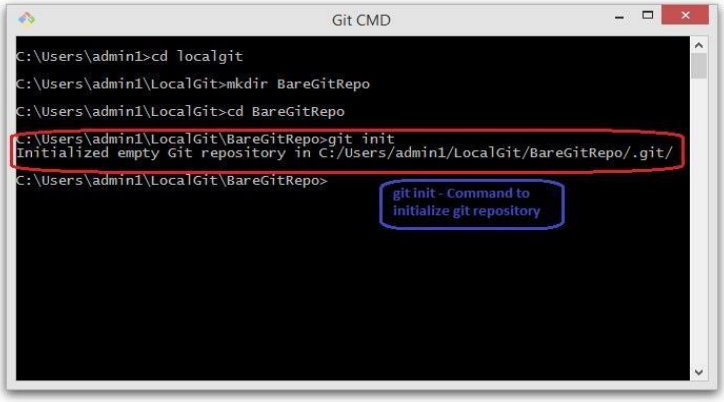
Bare Git repository means an empty directory with just a hidden ***.git*** folder. Let us name the project as ***BareGitRepo***. Ensure that you're in the ***LocalGit*** folder before proceeding with following steps. Enter command mkdir BareGitRepo.



Step 2: Browse to New Project - Navigate to the project created in the previous step using the command *cd BareGitRepo*.



Step 3: Initialize Bare Git Repository for the Project - Enter the command gitinit this command is used to Create Git Repository. Consequently, the execution of this command creates a hidden ***.git*** folder therein. Or in other words, an empty ***Git Repository is initialized***. You'll notice a message stating that an empty ***Git*** repository is created.



Note: ***git init*** *is a standard* ***GIT*** *command and it initializes the directory with a* ***.git*** *folder used*

for tracking versions of project artifacts.

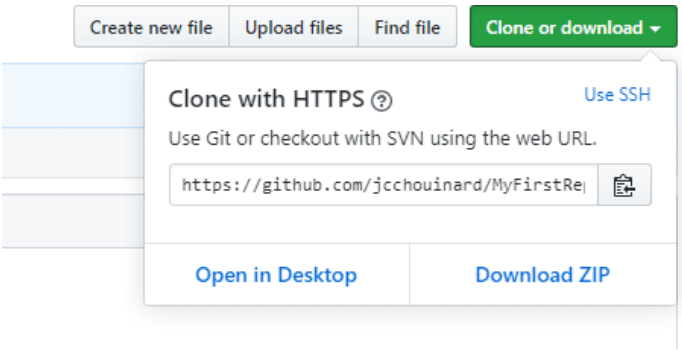
- **Cloning a repository:**

1. **Open Git Bash**
   If Git is not already installed, it is super simple. Just go to the Git DownloadFolder and follow the instructions.
2. **Go to the current directory where you want the cloned directory to beadded.**

To do this, input cd and add your folder location. You can add the folder location by dragging the folder to Git bash**.**

$ cd '/c/Users/j-c.chouinard/My First Git Project'

1. Go to the page of the repository that you want to clone
2. Click on "Clone or download" and copy the URL.



3. **Use the git clone command along with the copied URL from earlier.**

$ git clone https://github.com/USERNAME/REPOSITORY

4. **Press Enter.**

```
$ git clone https://github.com/USERNAME/REPOSITORY
Cloning into Git…
remote: Counting objects: 13, done.
remote: Compressing objects: 100% (13/13), done.
remove: Total 13 (delta 1), reused 0 (delta 1)
Unpacking objects: 100% (13/13), done.
```

- Staging and Committing changes:

Git add command is a straight forward command. It adds files to the staging area. We can addsingle or multiple files at once in the staging area. It will be run as:

1. $ git add <File name>

The above command is added to the git staging area, but yet it cannot be shared on the version control system. A commit operation is needed to share it. Let's understand the below scenario.

We have created a file for our newly created repository in **NewDirectory**. To create a file, use the touch command as follows:

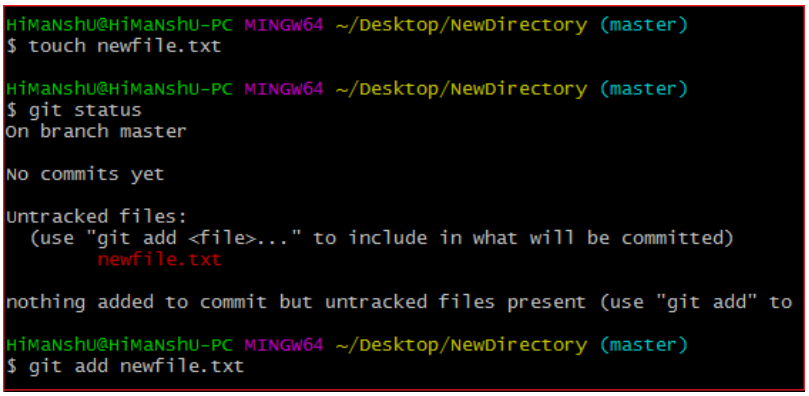
1. $ touch newfile.txt

And check the status whether it is untracked or not by git status command as follows:

1. $ git status

The above command will display the untracked files from the repository. These files can be added to our repository. As we know we have created a newfile.txt, so to add this file, run thebelow command:

1. $ git add newfile.txt Consider the below output:



From the above output, we can see newfile.txt has been added to our repository. Now, we have to commit it to share on Git.

Git Add All

We can add more than one files in Git, but we have to run the add command repeatedly. Git facilitates us with a unique option of the add command by which we can add all the available files at once. To add all the files from the repository, run the add command with -A option. We can use '.' Instead of -A option. This command will stage all the files at a time. It will run as follows:

1. $ git add -A

Or

1. $ git add .

The above command will add all the files available in the repository. Consider the below scenario:

We can either create four new files, or we can copy it, and then we add all these files at once. Consider the below output:

```
HiMaNshU@HiMaNshU-PC MINGW64 ~/Desktop/NewDirectory (master)
$ git add newfile.txt

HiMaNshU@HiMaNshU-PC MINGW64 ~/Desktop/NewDirectory (master)
$ touch newfile1.txt

HiMaNshU@HiMaNshU-PC MINGW64 ~/Desktop/NewDirectory (master)
$ touch newfile2.txt

HiMaNshU@HiMaNshU-PC MINGW64 ~/Desktop/NewDirectory (master)
$ touch newfile3.txt

HiMaNshU@HiMaNshU-PC MINGW64 ~/Desktop/NewDirectory (master)
$ git status
On branch master

No commits yet

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)
        new file:   newfile.txt

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        newfile1.txt
        newfile2.txt
        newfile3.txt
```

In the above output, all the files are displaying as untracked files by Git. To track all of these files at once, run the below command:

1. $ git add -A

The above command will add all the files to the staging area. Remember, the -A option is case sensitive. Consider the below output:

In the above output, all the files have been added. The status of all files is displaying as staged.

Now committing the Changes:

Git commit -m
The -m option of commit command lets you to write the commit message on the command line. This command will not prompt the text editor. It will run as follows:

1. $ git commit -m "Commit message."

The above command will make a commit with the given commit message. Consider the below output:

```
HiMaNshU@HiMaNshU-PC MINGW64 ~/Desktop/NewDirectory (master)
$ git commit -m "Introduced newfile4"
[master 64d1891] Introduced newfile4
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 newfile4.txt
```

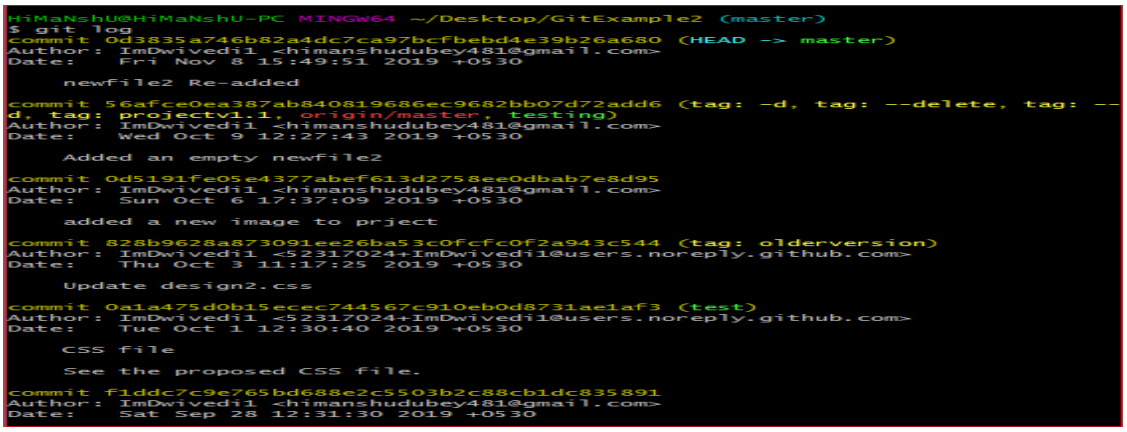
In the above output, a **newfile4.txt** is committed to our repository with a commit message.

- Viewing the history of all the changes:

Git log command is one of the most usual commands of git. It is the most useful command for Git. Every time you need to check the history, you have to use the git log command. The basic git log command will display the most recent commits and the status of the head. It will use as:

1. $ git log

The above command will display the last commits. Consider the below output:



The above command is listing all the recent commits. Each commit contains some unique sha-id, which is generated by the SHA algorithm. It also includes the date, time, author, and some additional details.

- **Git Branching and merging:**

Create Branch

You can create a new branch with the help of the **git branch** command. Thiscommand will be used as:

Syntax:

1. $ git branch **<branch** name>**Output:**

This command will create the **branch B1** locally in Git directory.

List Branch

You can List all of the available branches in your repository by using the followingcommand. Either we can use **git branch - list** or **git branch** command to list the availablebranches in the repository.

Syntax:

1. $ git branch --list

or

1. $ git branch

Output:

```
HiMaNshU@HiMaNshU-PC MINGW64 ~/Desktop/GitExample2 (master)
$ git branch
  B1
  branch3
* master

HiMaNshU@HiMaNshU-PC MINGW64 ~/Desktop/GitExample2 (master)
$ git branch --list
  B1
  branch3
* master
```

Here, both commands are listing the available branches in the repository. The symbol * is representing currently active branch.

Delete Branch

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. Below is the command to do this.

Syntax:

1. $ git branch -d<**branch** name>**Output:**

```
HiMaNshU@HiMaNshU-PC MINGW64 ~/Desktop/GitExample2 (master)
$ git branch -d B1
Deleted branch B1 (was 554a122).
```

This command will delete the existing branch B1 from the repository.

The **git branch d** command can be used in two formats. Another format of this command is **git branch D**. The '**git branch D**' command is used to delete the specified branch.

Branch Merging:

Git allows merging the whole branch in another branch. Suppose you have made many changes on a branch and want to merge all of that at a time. Git allows you to doso. See the below example:

```
HiMaNshU@HiMaNshU-PC MINGW64 ~/Desktop/GitExample2 (test2)
$ git add newfile1.txt
```

In the given output, I have made changes in newfile1 on the test branch. Now, I have committed this change in the test branch.

```
HiMaNshU@HiMaNshU-PC MINGW64 ~/Desktop/GitExample2 (test2)
$ git commit -m "edit newfile1"
[test2 a3644e1] edit newfile1
 1 file changed, 1 insertion(+)
```

Now, switch to the desired branch you want to merge. In the given example, I have switched to the

master branch. Perform the below command to merge the whole branch in the active branch.

1. $ git merge **<branchname>**

```
HiMaNshU@HiMaNshU-PC MINGW64 ~/Desktop/GitExample2 (test)
$ git checkout master
Switched to branch 'master'
Your branch is ahead of 'origin/master' by 1 commit.
  (use "git push" to publish your local commits)

HiMaNshU@HiMaNshU-PC MINGW64 ~/Desktop/GitExample2 (master)
$ git merge test2
Updating 2852e02..a3644e1
Fast-forward
 newfile1.txt | 3 ++-
 1 file changed, 2 insertions(+), 1 deletion(-)

HiMaNshU@HiMaNshU-PC MINGW64 ~/Desktop/GitExample2 (master)
$
```

As you can see from the given output, the whole commits of branch test2 have merged to branch master.

Experiment: 05

Aim: GitHub:

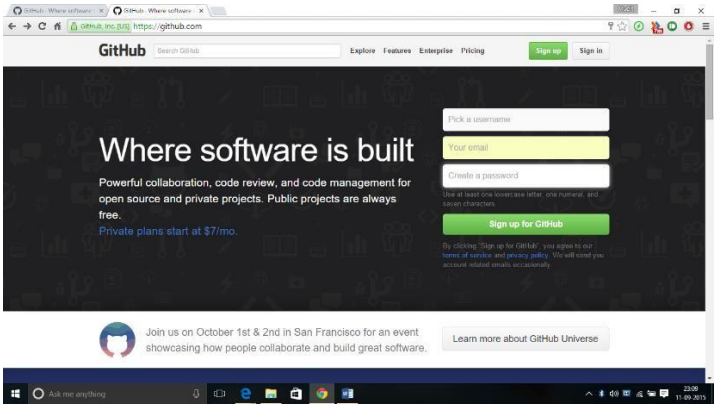
- Account creation and configuration.
- Demonstrate How to create Central repository.
- Create and Push to repository.

Theory: GitHub is a web-based interface that uses Git, the open source version control software that lets multiple people make separate changes to web pages at the same time. As Carpenter notes, because it allows for real-time collaboration, GitHub encourages teams to work together to build and edit their site content.

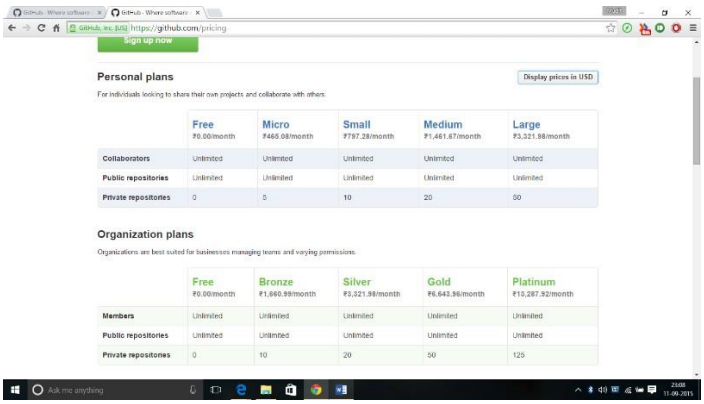
Procedure:

- Account creation and configuration:

Step 1: Go to github.com and enter the required user credentials asked on the site and then click on the SignUp for GitHub button.

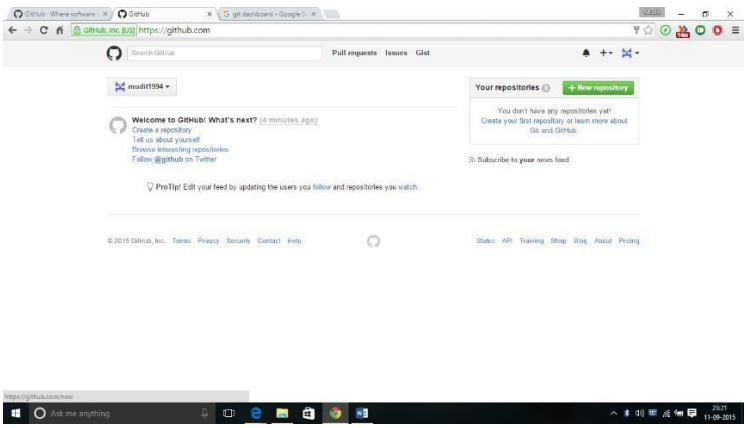


Step 2: Choose a plan that best suits you. The following plans are available as shownin below media as depicted:



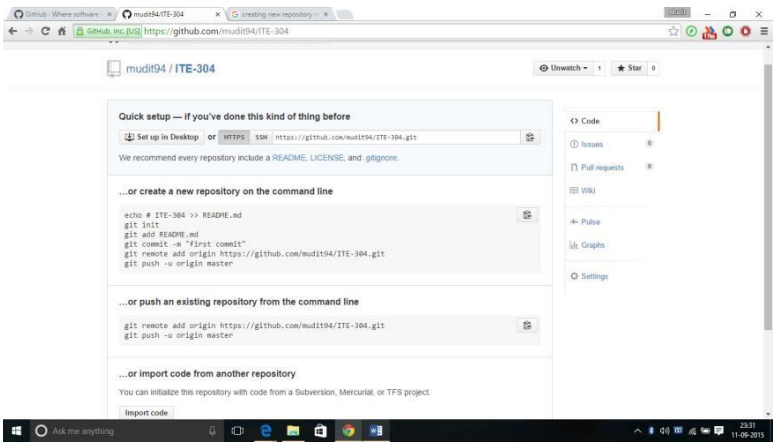
Step 3: Then Click on Finish Sign Up.

The account has been created. The user is automatically redirected to yourDashboard.



- Creating a Central repository:

- Login to your Github account
- On the dashboard click on the Green Button starting New repository.
- Make sure to **verify the Github account** by going into the mail which wasprovided when creating the account.
- Start by giving a repository name, description(optional) and select the visibilityand accessibility mode for the repository
- **Click** on Create repository
- The repository (in this case ITE-304 is the repository) is now created. Therepository can be created looks like:



- **Create and push to repository:**
  - The **system should have git installed** in it if not install git. Make sure tochoose Run git from Windows Command prompt option during installation. Otherwise, open git bash in place of step 2.
  - Open the command prompt (for Windows and Linux users).
  - Change the current working directory to your local project
  - Initialize the local directory as a git repository in different ways as described in the image.

$ git init

- A **new .git folder** is **created** in the directory which is by **default hidden**.
- Add the files in your new local repository. This stages them for the firstcommit.

$ git add

#Adds the files in the local repository and stages them for commit. To unstage a file,use

$ git reset HEAD ***YOUR_FILE***

Commit the files that you've staged in your local repository.

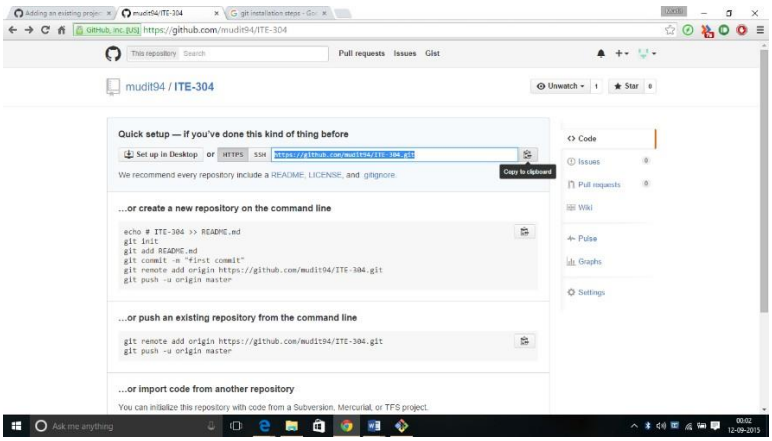
$ git commit -m 'First commit'

To remove this commit and modify the file, use

$ git reset --soft HEAD~1

And commit and add the file again.

At the top of the GitHub repository's Quick Setup page, click on the icon shown andcopy the remote repository URL.



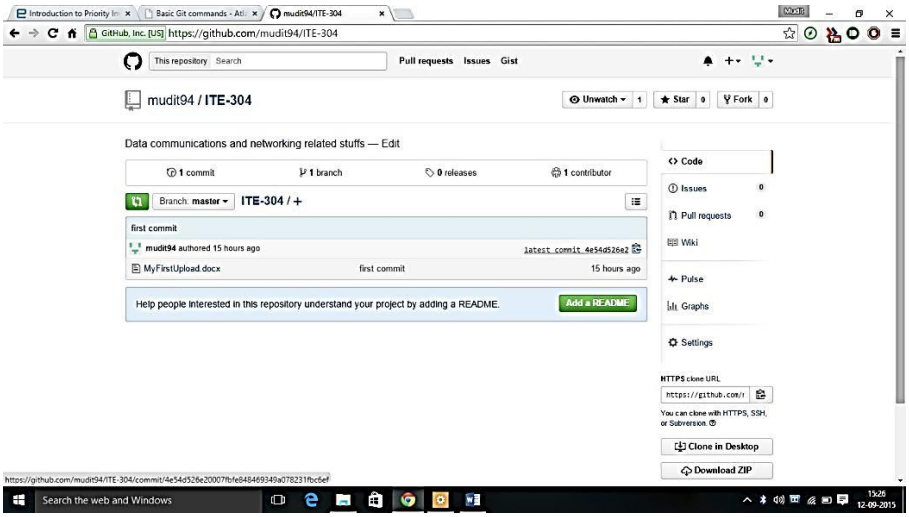
In the Command prompt, **add the URL for the remote repository** where your localrepository will be pushed.

$ git remote add origin remote repository URL# Connects to the remote repository
$ git remote -v
# Verifies the new remote URL

Push the changes in your local repository to GitHub.

$ git push origin master
# Pushes the changes in your local repository up to the remote repository youspecified as the origin.

And here you go…


https://github.com/mudit94/ITE-304
This repository Search
Pull requests Issues Gist
mudit94 / ITE-304
Unwatch 1 Star 0 Fork 0
Data communications and networking related stuffs — Edit
1 commit 1 branch 0 releases 1 contributor
Branch: master ITE-304 / +
first commit
mudit94 authored 15 hours ago
latest commit 4e54d526e2
MyFirstUpload.docx first commit 15 hours ago
Help people interested in this repository understand your project by adding a README.
Add a README
Code
Issues 0
Pull requests 0
Wiki
Pulse
Graphs
Settings
HTTPS clone URL
https://github.com/
You can clone with HTTPS, SSH, or Subversion.
Clone in Desktop
Download ZIP
Search the web and Windows
12-09-2015

Experiment: 06

Aim: Create Cloud Services account and Demonstrate:

- Create and Setup Virtual Machine.
- Create a simple web app using Cloud Services:

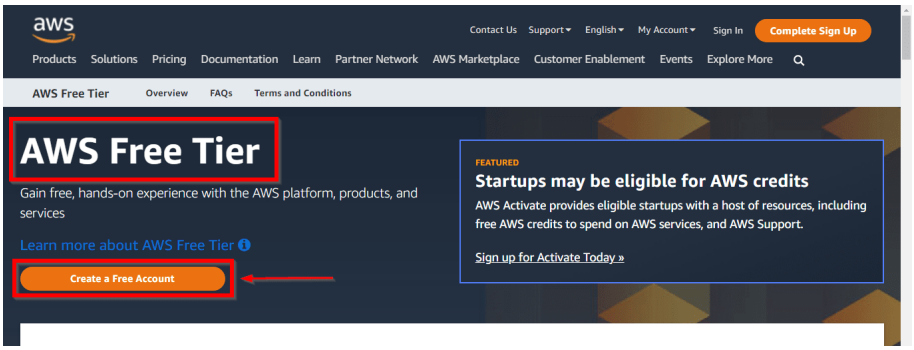
Theory: The term "cloud services" refers to a wide range of services delivered on demand to companies and customers over the internet. These services are designed to provide easy, affordable access to applications and resources, without the need for internal infrastructure or hardware. From checking email to collaborating on documents, most employees use cloud services throughout the workday, whetherthey're aware of it or not.
Cloud services are fully managed by cloud computing vendors and service providers. They're made available to customers from the providers' servers, so there's no need for a company to host applications on its own on-premises servers.

Virtual Machine: A VM is **a virtualized instance of a computer that can perform almost all of the same functions as a computer, including running applications and operating systems**. Virtual machines run on a physical machine and accesscomputing resources from software called a hypervisor.

Procedure:

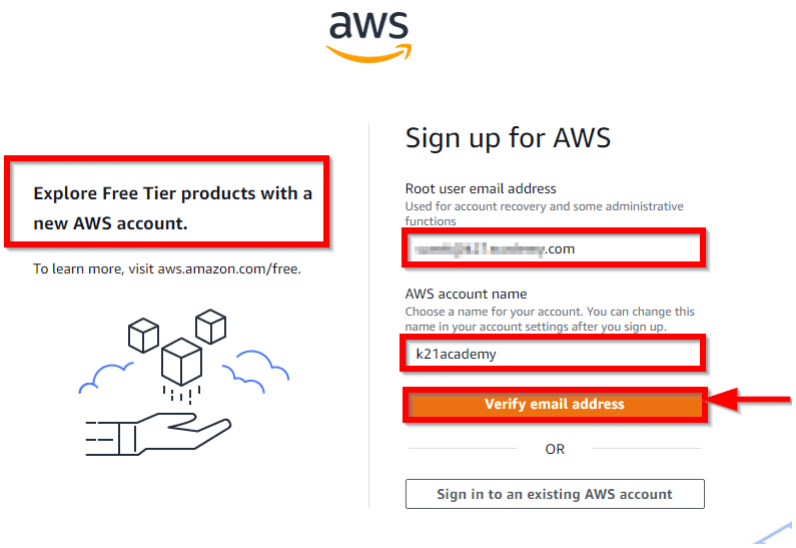
- Creating a AWS Cloud Account:

1. First Open your web browser and navigate to **AWS Free Tier Page**
2. On middle-click on Create a Free Account



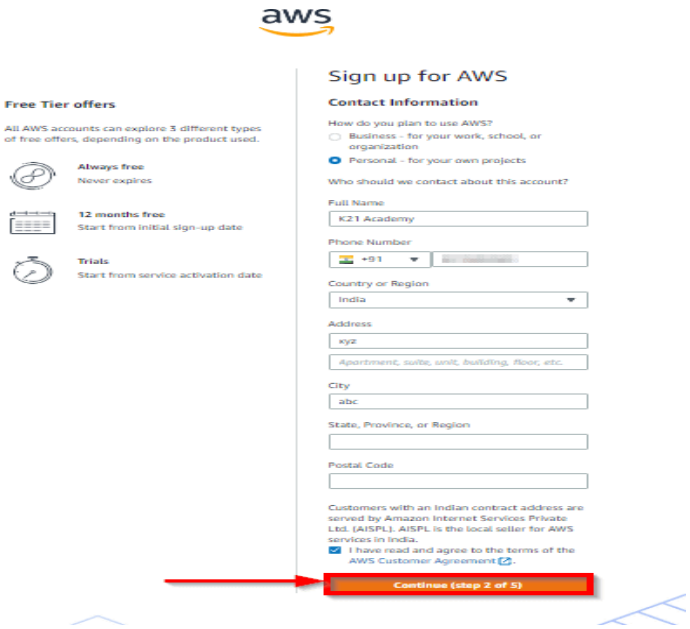
3. Issue the details which you want to use for login your AWS account and click onContinue

- **Email address:** Provide the mail id which hasn't been registered yet withAmazonAWS.
- **Password:** Type your password.
  Confirm password: Authenticate the password.
- **AWS Account name:** Choose a name for your account. You can change this name in your account settings after you sign up.



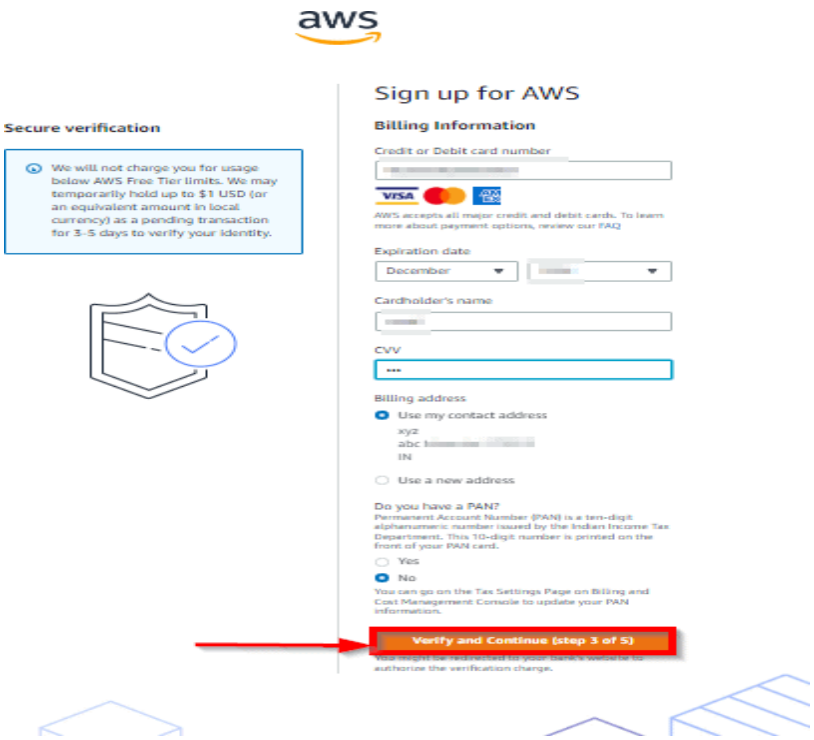
4. Contact Information

Select your AWS type (Profesional/ Personal) Fill in the correct information to validate your account if you're going to create personal use then click on "Personal Account" else use "Company Account", Accepts the Terms and condition and then click on Create Account and Continue

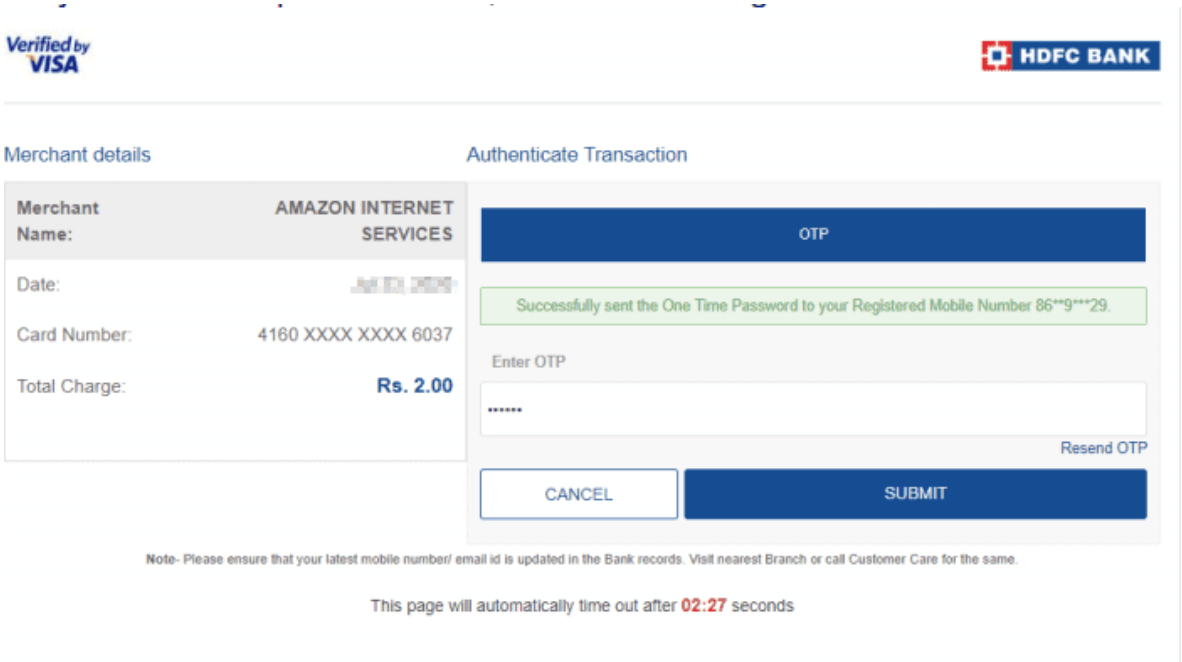


Note: Make sure to provide proper contact details and mobile number to getthe Verification code from AWS.

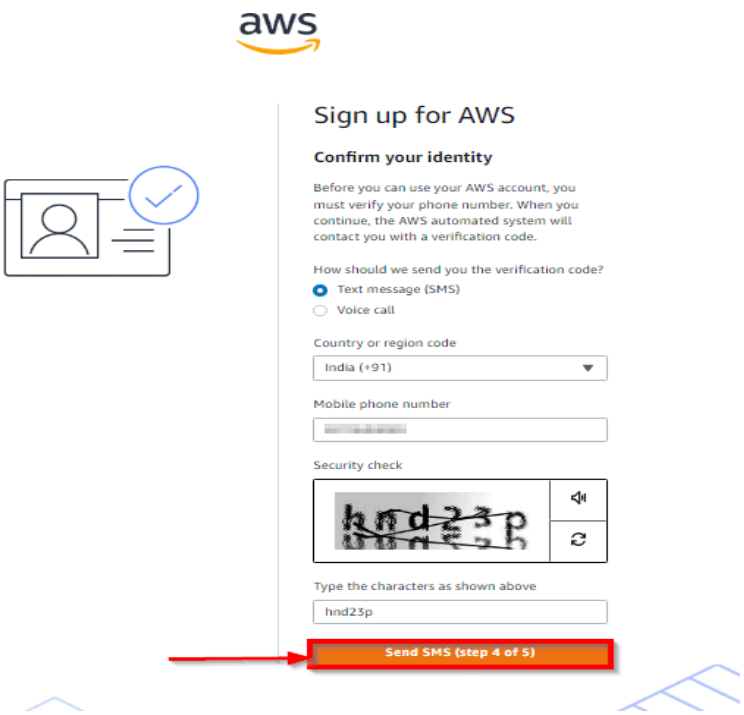
5. Payment and PAN information: In this step, you must fill in your credit card /Debit Card info and billing address and click on Secure Submit.



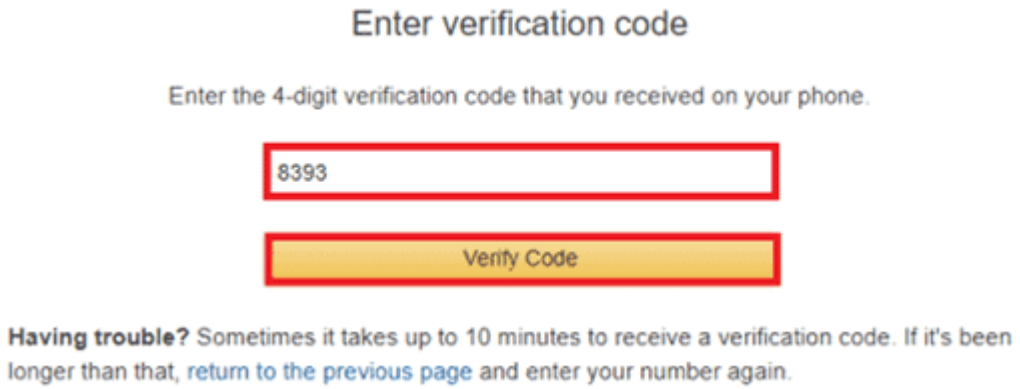
6. In this step, it will take you to the payment gateway to validate your paymentinformation and for your credit card verification, Amazon will charge theminimum price based on Country. Here I have provided India, so Amazon charged 2 INR.



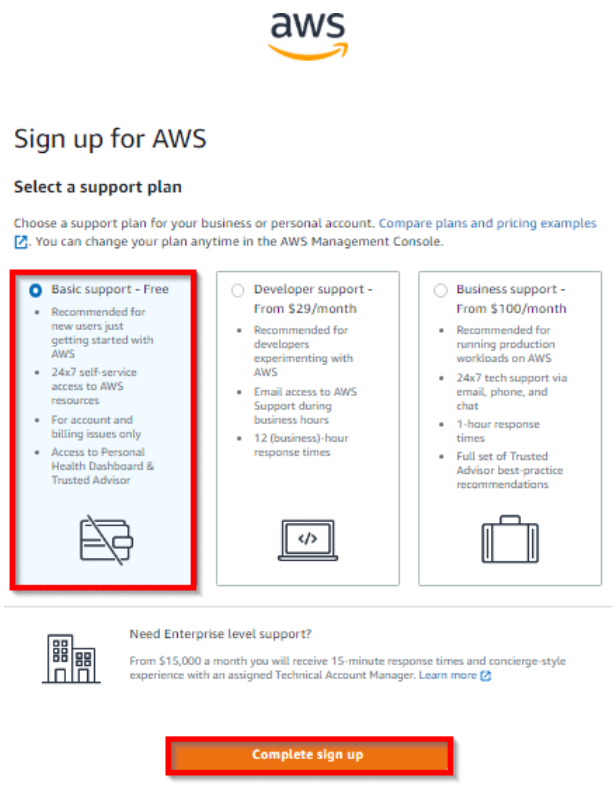
7. Phone verification: Here you will be taken to an identity verification page that will already have your phone number, so you just have to select either “Text message or Voice call” Provide a valid phone number, Solve the captcha and then click on Send SMS or Call Me Now(depending upon your selection).



8. After clicking on Send SMS or Call me Now, you will immediately receive acall or SMS from Amazon, for verification code, Enter your code then click on Verify Code.

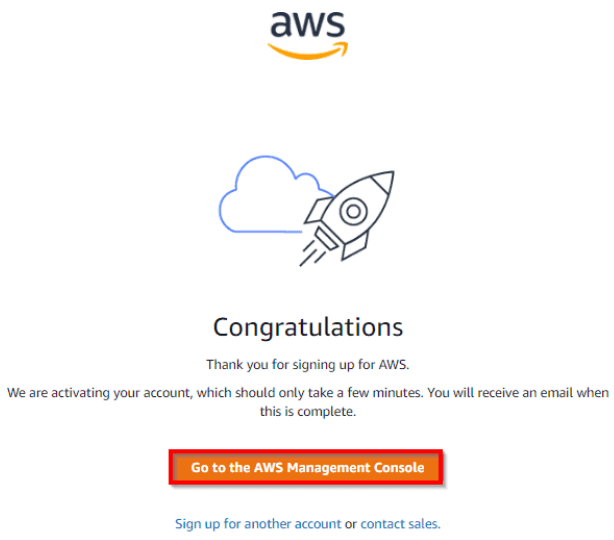


9. Support plan: AWS support offers a selection of plans to meet your business needs. Select your suitable plan then click continue.



Note: All customers receive free basic support.

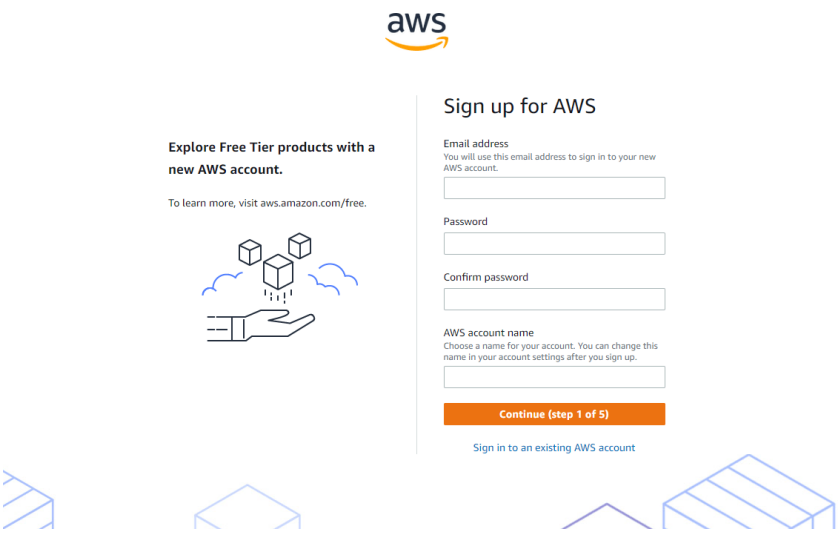
10. Registration Confirmation page.Once you completed all the above steps and processes. You'll get the confirmation page as below. Now your account will be processed for activation. It may take somewhere between 30 minutes to 1 hour for you to receive an email confirmation that your Amazon Cloud Services account has been activated.



- Create and Setup Virtual Machine:

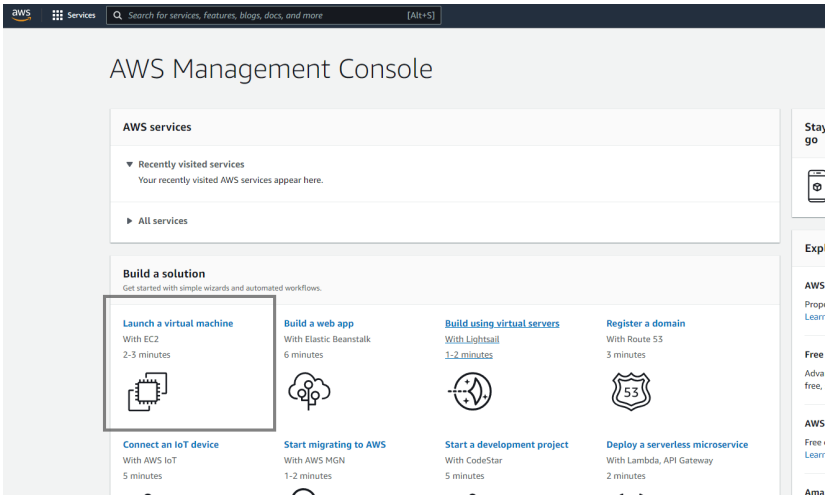
1. Create an AWS account

   You can easily create an AWS account on the AWS Console. All new sign-ups geta free-tier offer.



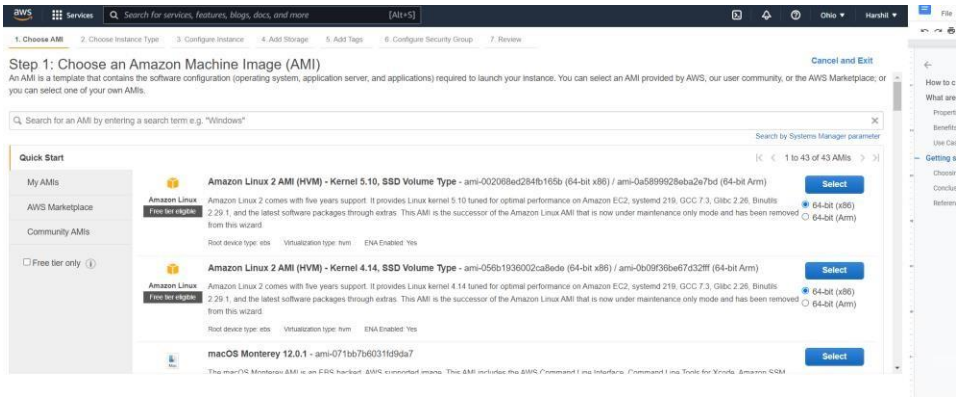
2. Launch AWS virtual machine

   Once you finish setting up your account, you can click on the AWS logo on the top left corner or search "console" on the search bar. You'll find a number of options in the AWS console. Select "**Launch a virtual machine**" to get started with VMs. If you're a new user, it can take up to 24 hours for your account to activate.



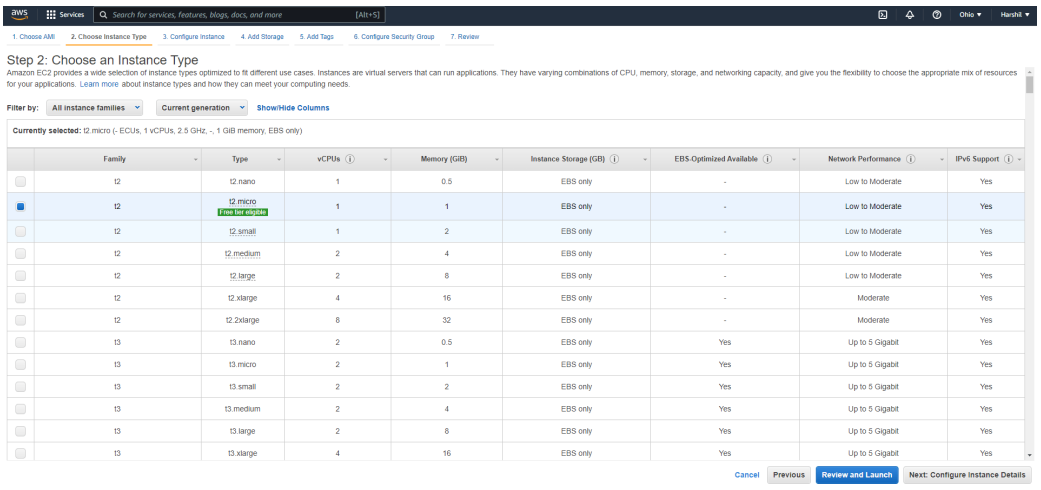
3. Choose AMI

Amazon Machine Image (AMI) highlights the software setup (OS, application server, and apps). You can select Mac, Linux, or Windows OS. We'll look at the setup for Windows virtual machines here.

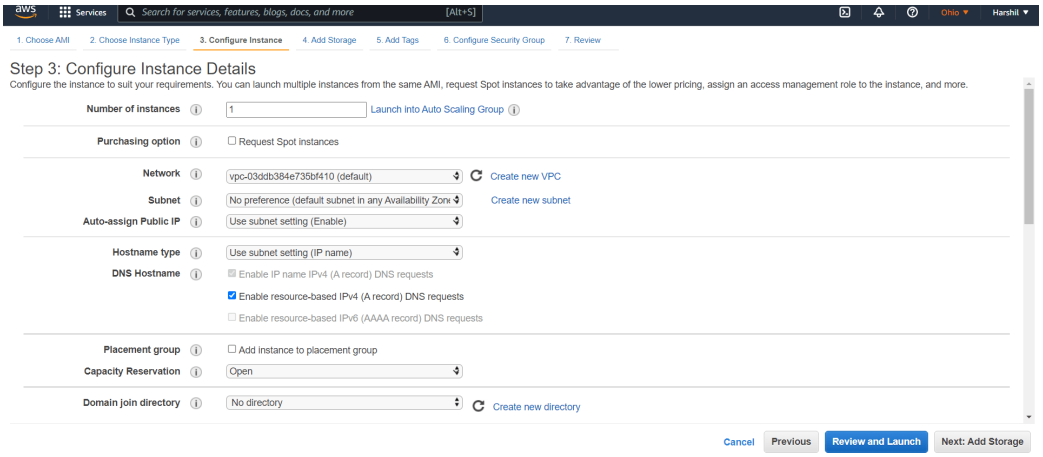


4. Choose and configure instance type

After choosing your operating system, you need to pick an instance type. Amazon EC2 offers many instance types tailored to specific use cases. **An instance is a virtualserver or virtual machine.** They come in a variety of CPU, memory, storage, networking, and a lot more.



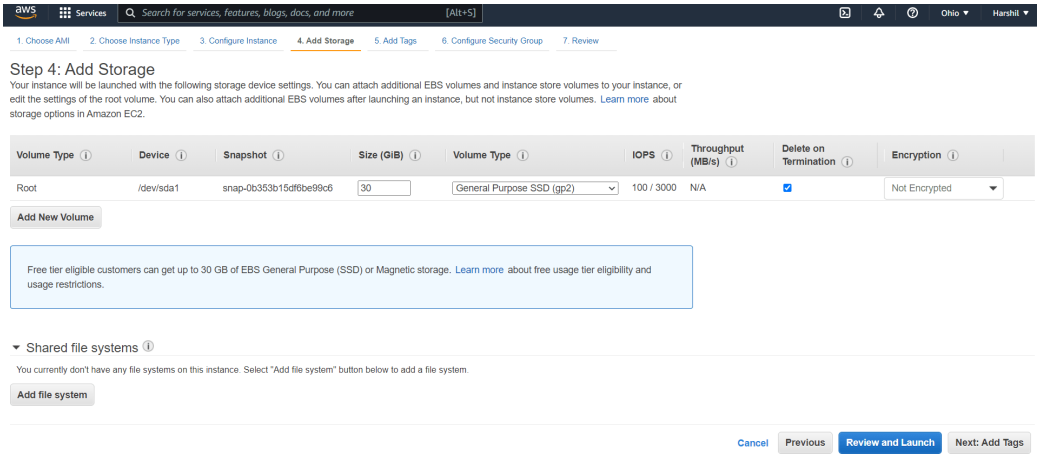
You can configure instance details, such as the number of instances, network, host type, and so on. Here, we'll use one instance and keep the remaining details default.



5. Add storage and tags

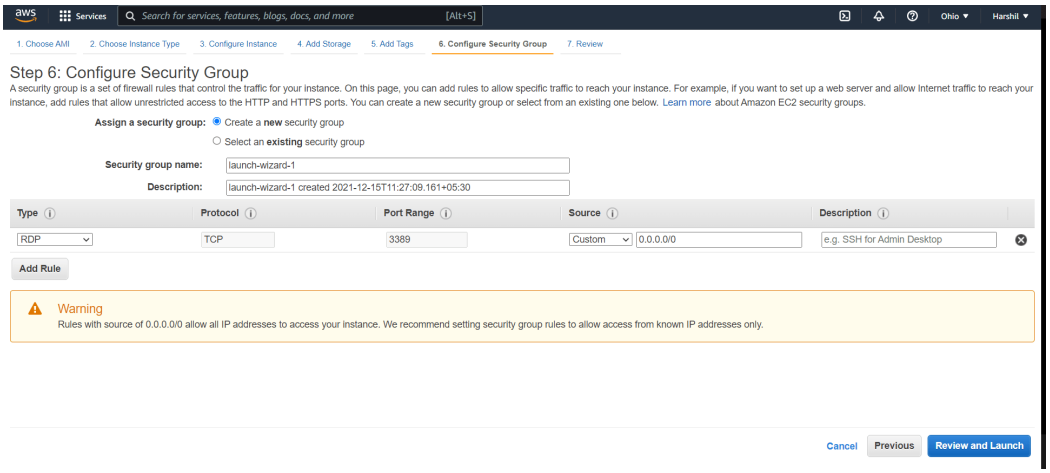
Once you configure an instance type, you can add or update storage info. AWS allowsyou to add more EBS volumes and instance store volumes, as well as change the root volume's parameters. *Amazon Elastic Block Store (EBS) provides block-level storage volumes for use with EC2 instances. It behaves like raw, unformatted block devices. You can mount these volumes as devices on your instances.*

The next step is adding tags. A **tag is a label** applied to an AWS resource. Each tag has a key and an optional value, which users define.



6. Configure security

A **security group** is a set of firewall rules that control data entering and exiting your instance. You may either recreate it or pick an existing security group.

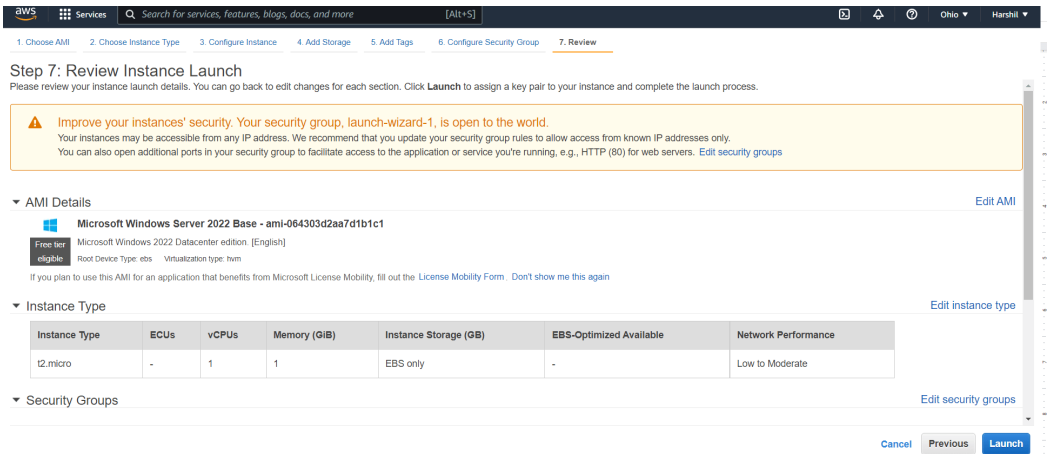


Security is a major concern when working on public clouds like AWS and Google Cloud Platform (GCP). Attackers can launch different attacks on public cloud deployments for lack of provider security. These attacks include DOS, DDOS, websitedefacement, and brute-force.

- Public clouds have **poor security**, but with the right set of rules, they can be improved.
- Public clouds offer **limited customization**. Clients can choose the operating system and size of the virtual machine.
- Cloud **data breaches** are frequently caused by misconfigured cloud security settings. Many companies' cloud security posture management solutions are insufficient for safeguarding their cloud-based infrastructure.

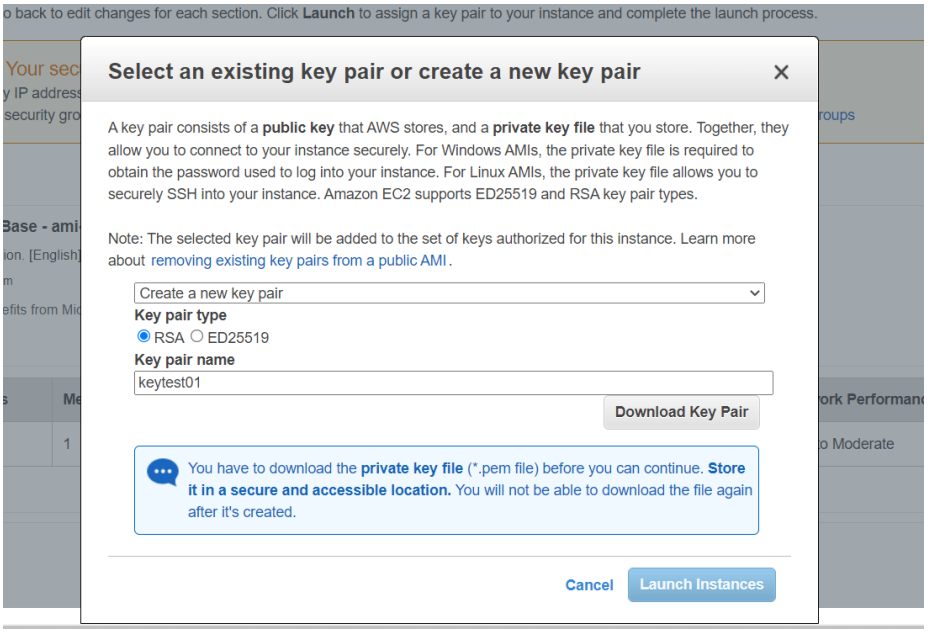
7. Review and launch your AWS virtual machine

The final step in creating an AWS virtual machine is to go through your instance details. Make sure every detail is correct, then click **"Launch"**.



When you click "Launch," you need to provide a key. To create a new key, select **"Create a new key pair"** from the drop-down menu and set a key name, for example, keytask, keytest1, and so on. Make sure you **download "key pair" before** launching your instance**.**

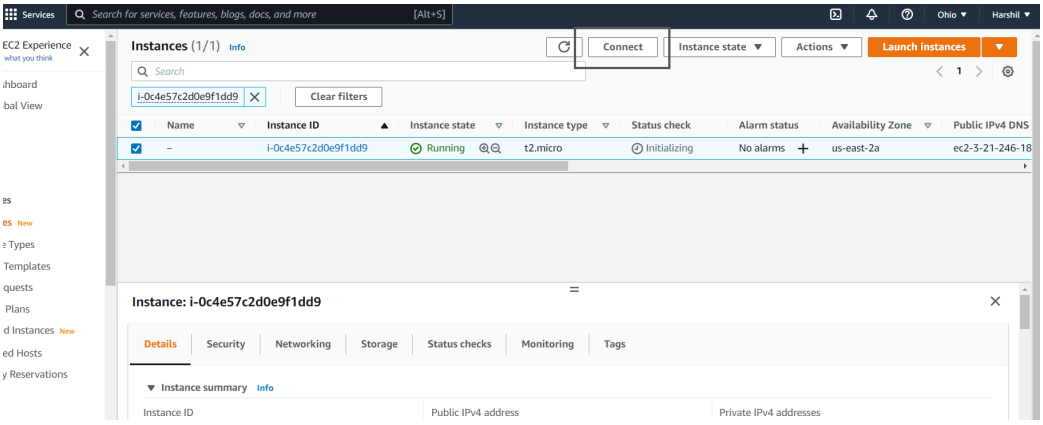
A key pair is made up of a public key stored by AWS and your private key file. They work together to allow you to connect to your instance safely.



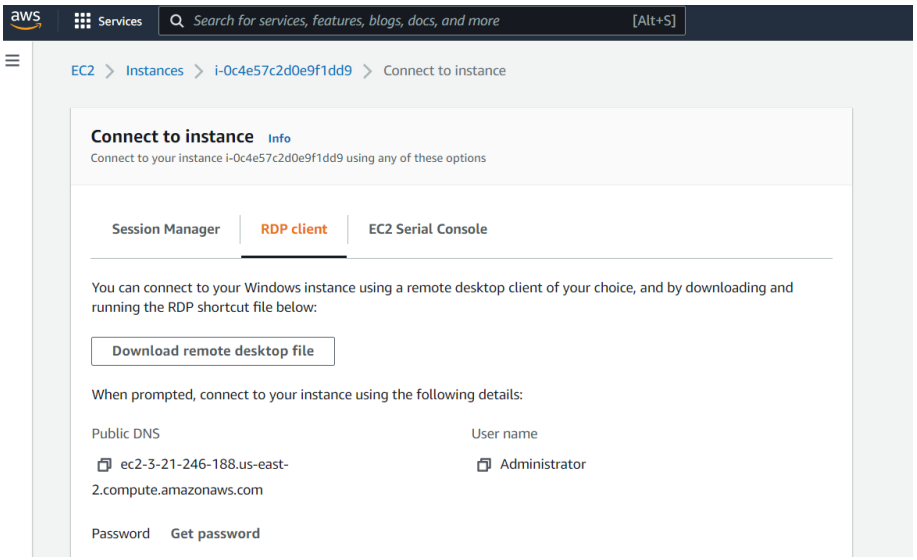
Voila! You successfully created and launched a virtual machine on AWS. Now, checkthe launch status and connect to an instance.

8. Connect to an instance

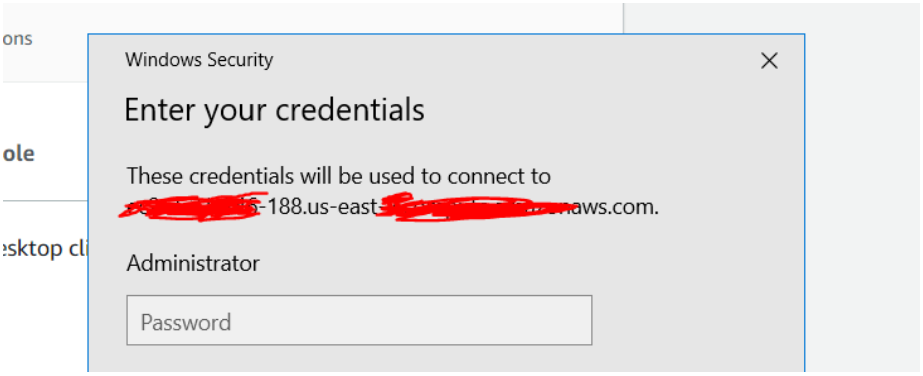
After starting the instance, you can check the status using **"Dashboard>Instances"**. Select your instance in the instance dashboard and click **"Connect"**.



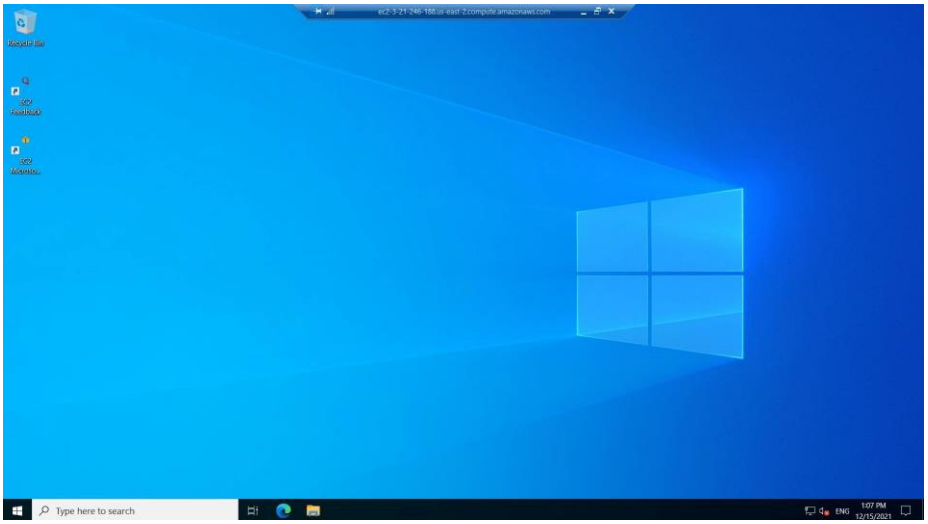
Select "**RDP client,"** click "**Get password,"** then upload the key pair downloaded when the instance launched (in step 7). After uploading the file, click **"decrypt password"** and download the remote desktop file**.**



Open the downloaded file and enter your password.

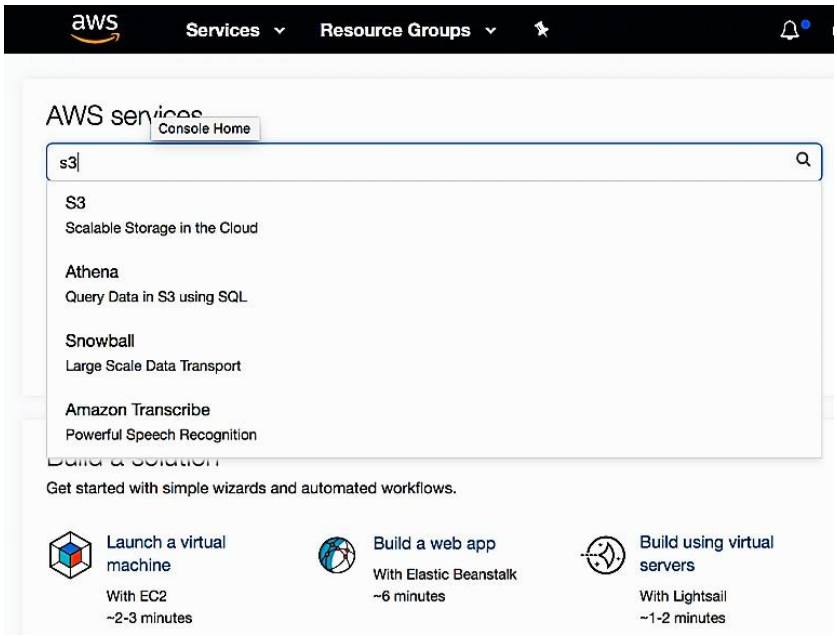


You should now see a screen similar to the one below, indicating that your **AWSWindows virtual machine** successfully launched!

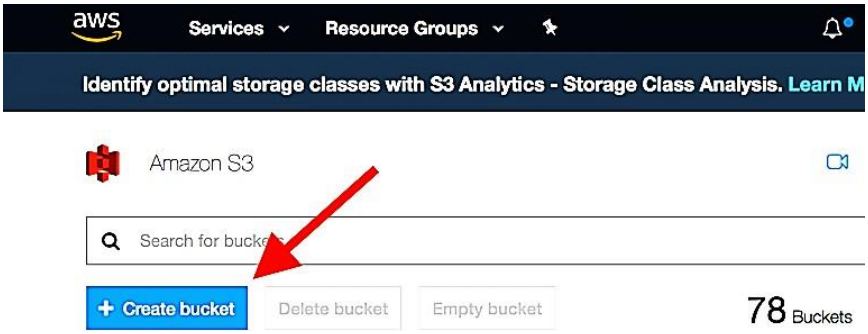


- Create a simple web app using Cloud Services:

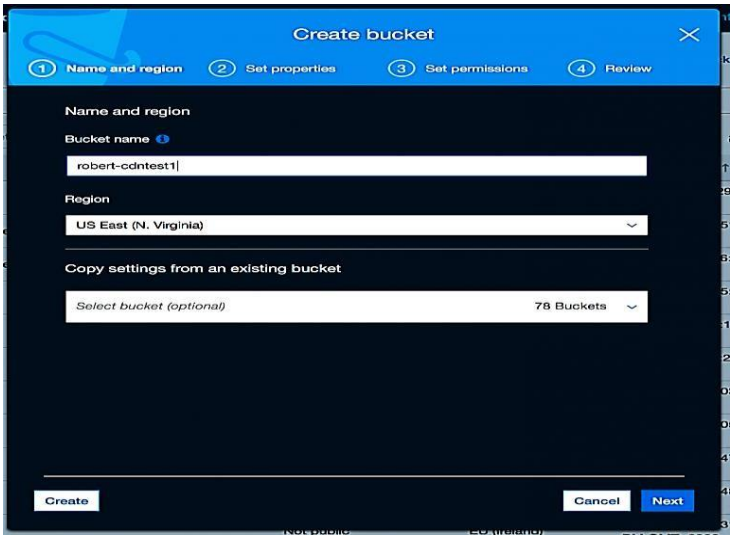
Once you've signed up for the console, open it, and search for S3, Amazon's objectstorage solution.



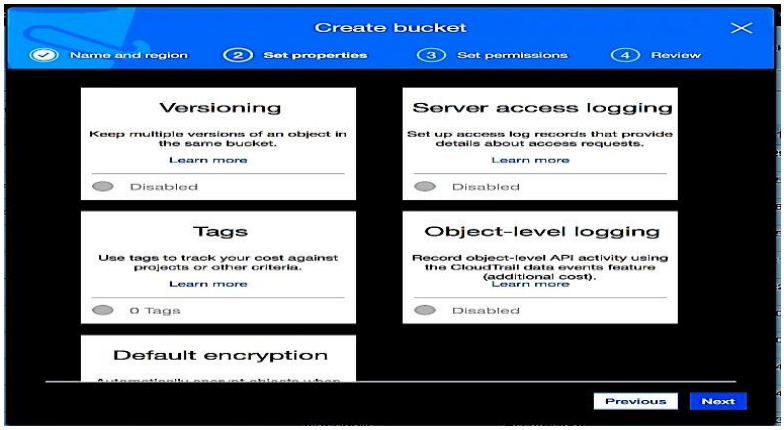
Now click **Create bucket**.



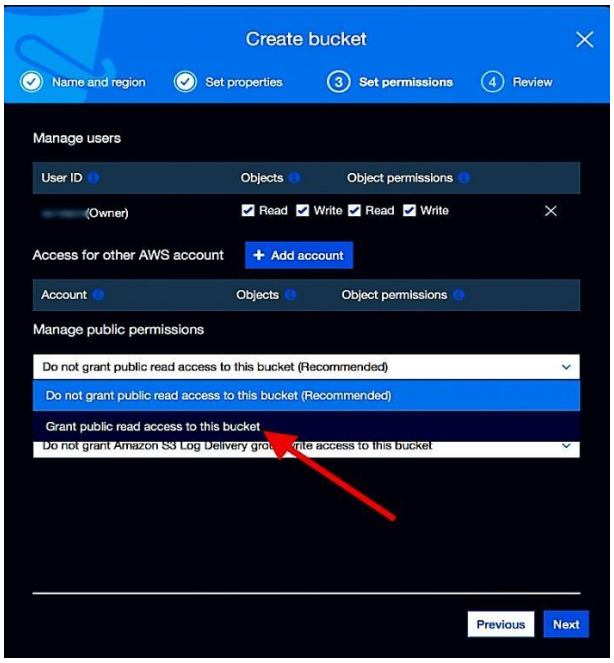
Enter the bucket's name, specify the region and click *Next*.



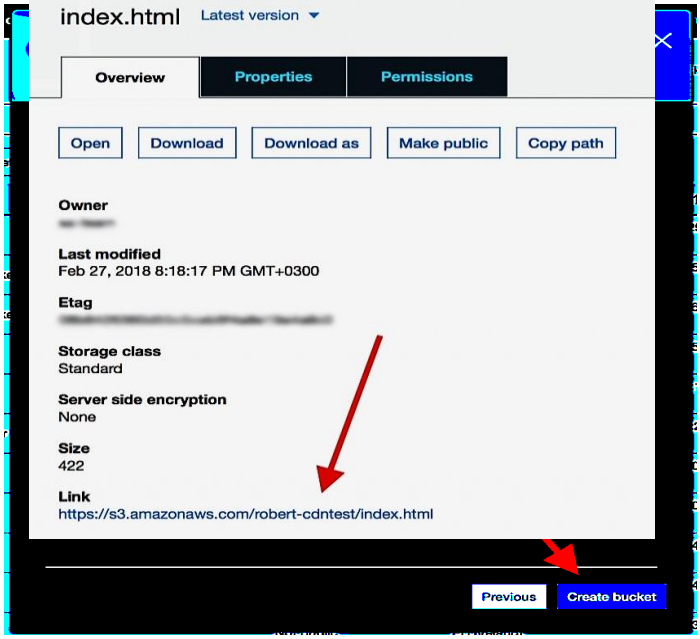
In the next step, you can enable versioning, server access logging, tagging, object- level logging, and default encryption. We don't need that for now, so just skip these options and click **Next**.



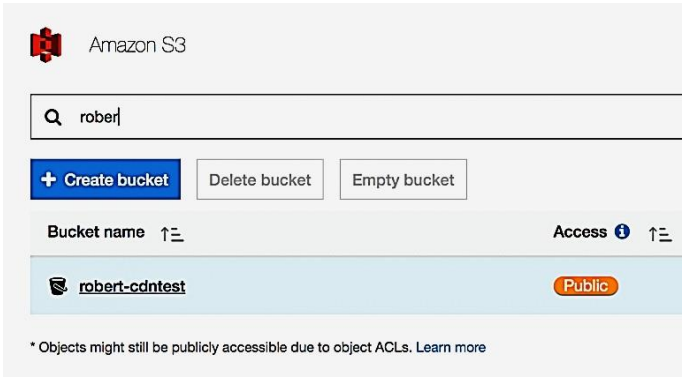
In the next step, you can specify the required permissions. It is important that you select **Grant public read access to this bucket** in the public permissions drop-down menu. Otherwise, it will be impossible to access the website's contents located in this bucket.



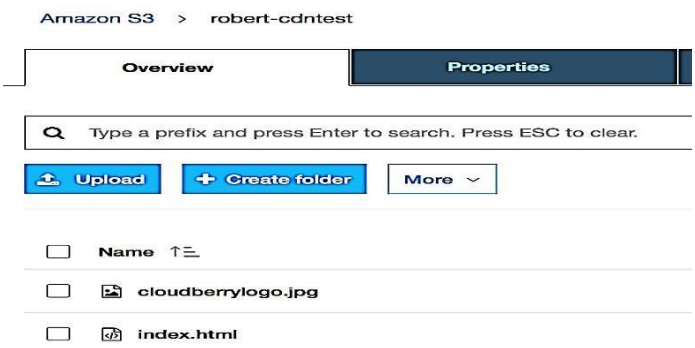
Review the bucket's properties and permissions, and click **Create bucket**.



Now, locate your bucket in the bucket list and click on it.



Click **Upload** and upload the assets. In our case, we'll simply add an HTML file andan image to host a simple static page.



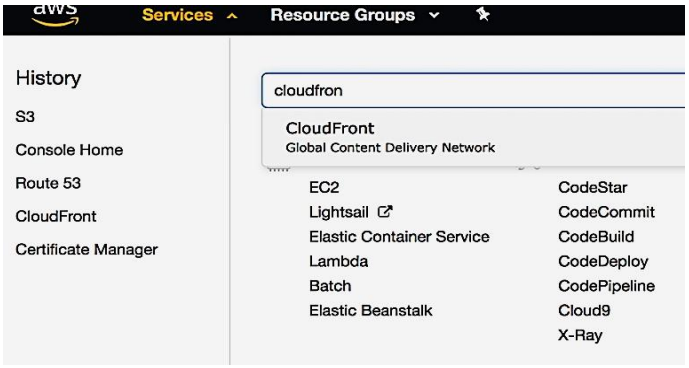
Our hosting is now ready, and we can access the assets from anywhere. Just click on the HTML file and try to open the link. You should see the webpage you've uploaded to the bucket. If you only want to use AWS for hosting and want to get the domain elsewhere, you're done! Register your domain at a different company andpoint it to the link displayed in the screenshot below.

Now that we've covered hosting, let's move over to network optimization withCloudFront.

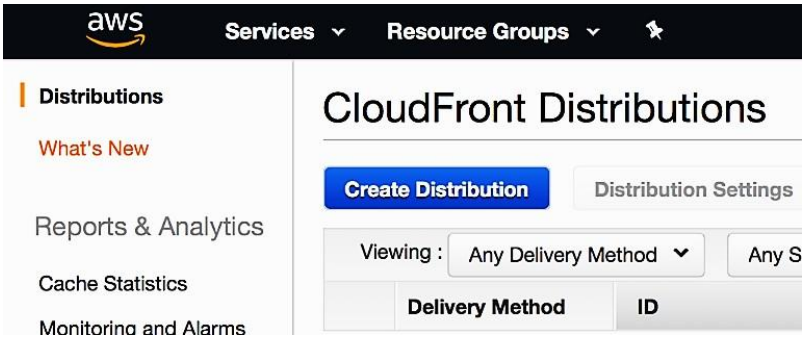
Leveraging Amazon CloudFront

Amazon CloudFront is a global content delivery network (CDN) service that securely delivers data to your users with low latency and high transfer speeds. CloudFront delivers your content through a worldwide network of data centers called edge locations. When a user requests content that you're serving with CloudFront, the user is routed to the edge location that provides the lowest latency (time delay), so thatcontent is delivered with the best possible performance.

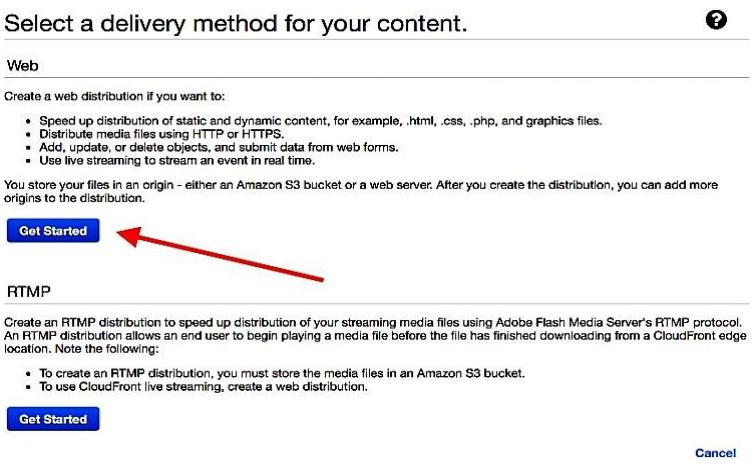
Search for CloudFront in the AWS Console search bar. Open it.



At this point, you need to create a CloudFront distribution to ensure that yourwebsite's assets are accessible from anywhere quickly. Click **Create Distribution**.



Click **Get Started** under *Web*.



The next step is one of the most complicated. Here, you need to specify the distribution's origin settings, default cache behavior settings, and general distribution settings. This lengthy article from Amazon provides an in-depth description of each option.
We'd like to point out just a few essential options that you need to configure:

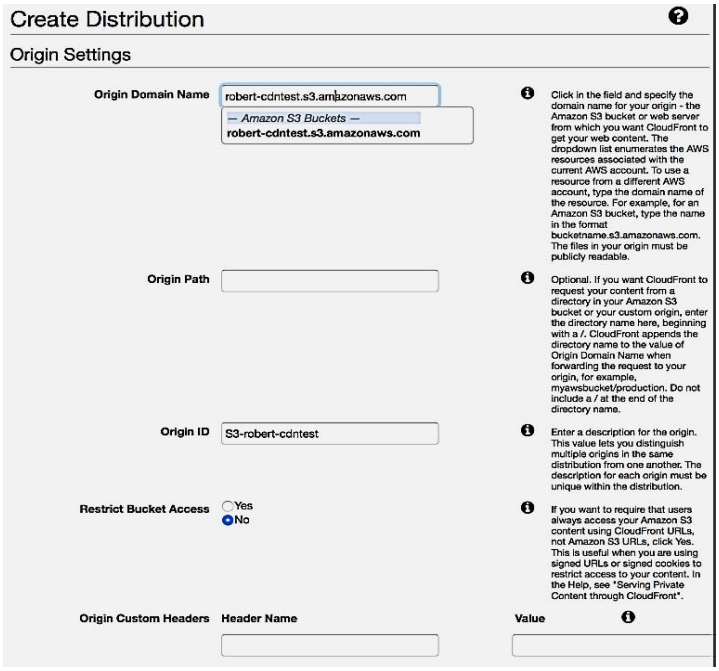
Origin Domain Name. This is the source bucket that contains all the data. Once you start typing in the bucket's name, Amazon will automatically suggest the right option.

Alternate Domain Names (CNAMEs). If you want to use subdomains to access your website—as we're going to do—specify them. We're going to enter *test.cloudberrylab.io.*

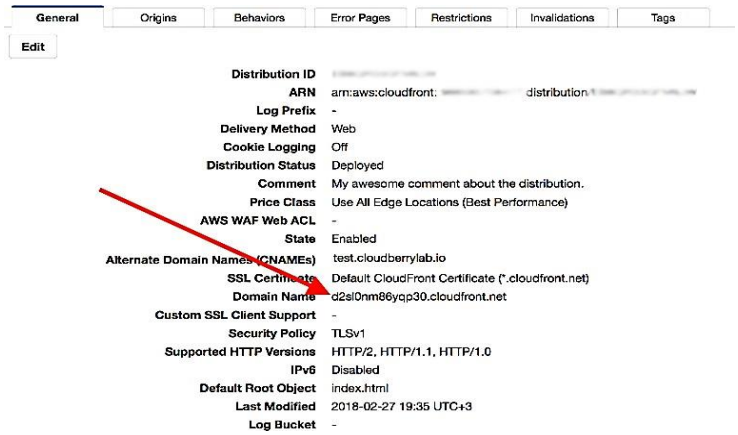
Default Root Object. By default, if you try to open the website's address, you'll encounter an error because *test.cloudberrylab.io* is not an HTML page, while *test.cloudberrylab.io/index.html* is. At this point, requiring users to specify the index page is outdated. To enter the page's name, and users will automatically be redirected

from*test.cloudberrylab.io to test.cloudberrylab.io/index.html.* Do not write "/index.html"—simply enter "index.html" without a slash.

IPv6. It's easier if you disable IPv6 for now. If it's enabled, you'll have to create two alias records when redirecting your domain to CloudFront. You can do that later; for now, just disable it.



When done, click Create Distribution. Once it's created, click on it, and the console will display the general information. The Domain Name is your CloudFront address. When accessing the website's content using this link, users will receive the data from the fastest location. Copy and paste it into the browser address field and hit Enter. Youshould see your page!



It's important that your HTML files refer to the images using the CloudFront addresses to ensure that images are loaded quickly. Every asset is available at the following address:

CloudFront Domain name + the asset's path in the S3 bucket.

Here's how that looks within our sample HTML file:

```
<html>

<body>

    <p>This static page is stored in an S3 bucket and can be quickly reached with the help of
    CloudFront. The image is loaded from the same S3 bucket. CloudFront and S3 are leveraged under
    the free 12-month tier while the domain name is provided for an extra fee.</p>


    <p><img style="max-width:150px;" src="https://d2sl0nm86yqp30.cloudfront.net/cloudberrylogo.jpg">

</body>

</html>
```

And here's our page:

All of the assets are stored in S3 and are reached through CloudFront for the best performance.

Under the AWS Free Tier, you get 50 GB Data Transfer Out, and 2,000,000 HTTP and HTTPS Requests from Amazon CloudFront free of charge for 12 months.

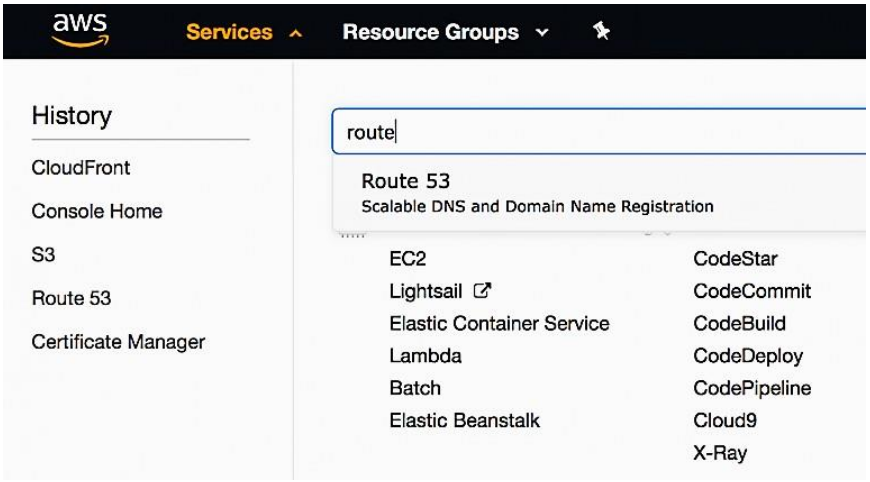
We're close to finished here. In the last section, we will explain how to register a domain and redirect it to CloudFront's domain name using AWS Route 53. Naturally, this comes at an extra price. So if you were looking only for the free solutions, you can close the article now and enjoy your website!

Registering a Domain Using AWS Route 53

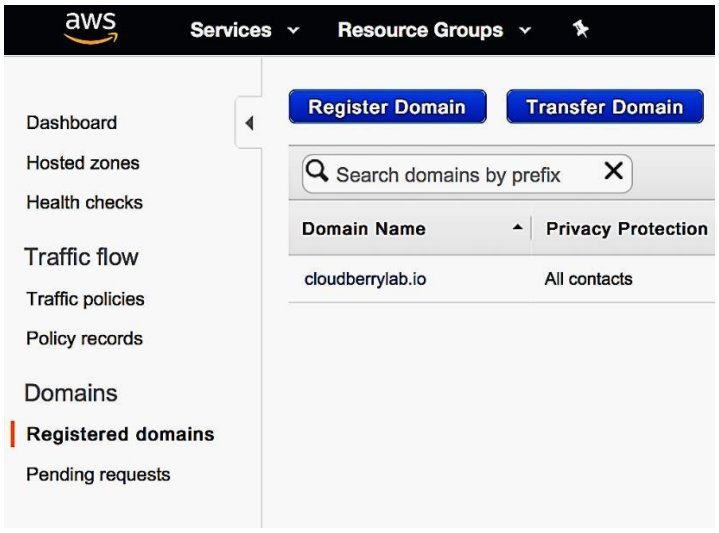
You can use Amazon Route 53 to help you get a website or web application up and running. Route 53 performs three main functions:

1. Registering domain names
2. Routing Internet traffic to the resources for your domain
3. Checking the health of your resources.

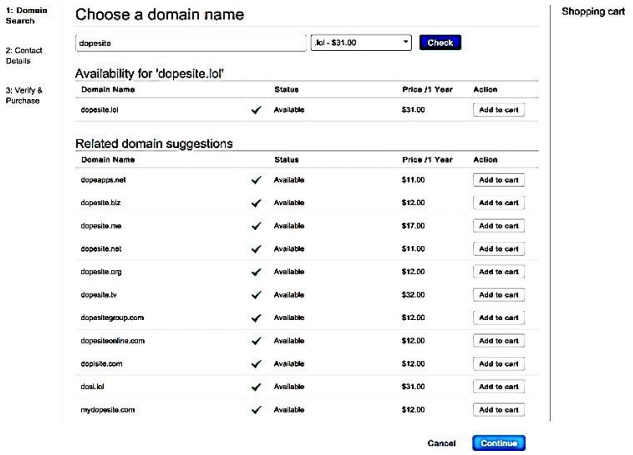
Go to the AWS Route 53 service.



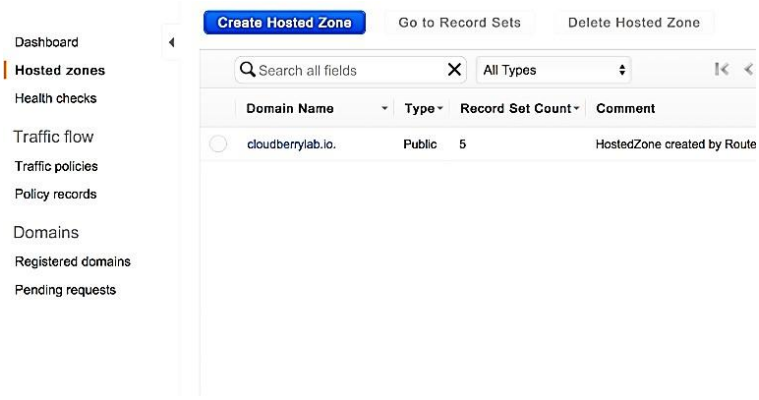
Under *Registered Domains*, click **Register Domain**.



Go through the three steps .



When done, go to the *Hosted zones* and click on your domain.

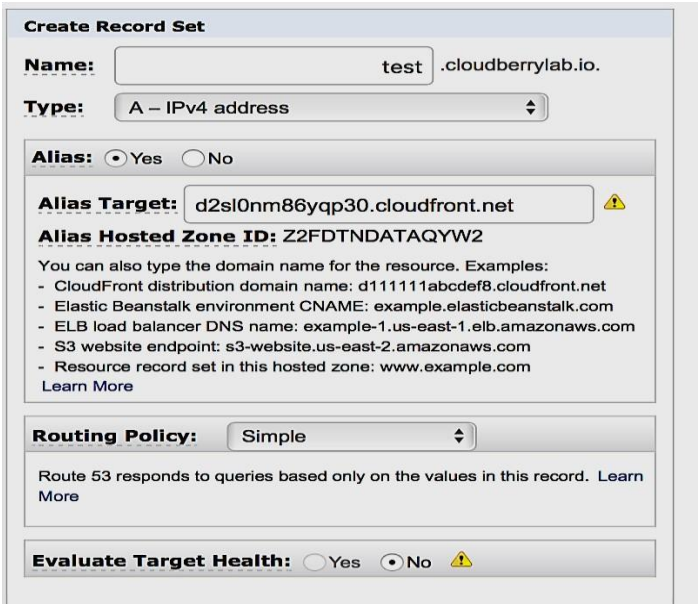


Now you need to create a r*ecord set* which will route Internet traffic to your CloudFront distribution. Click **Create Record Set**.

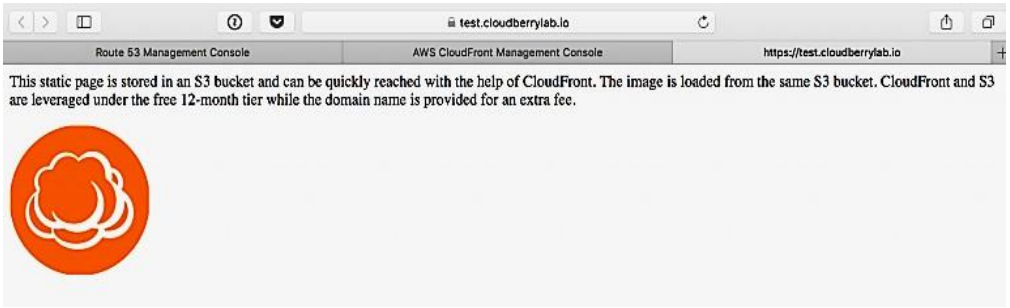
Now, specify the following parameters:

- **Name.** Enter the domain name that you want to use to route traffic to yourCloudFront distribution. In this example we'll type in *test;*
- **Type**. Select **A – IPv4 address;**
- **Alias**. Select **Yes;**
- **Alias Target.** Enter the domain name of your CloudFront distribution;
- **Routing Policy.** Leave the default value of **Simple**;
- **Evaluate Target Health**. Select **No**.

Click **Create**.



It may take a couple of minutes for your traffic to be routed to your CloudFront distribution. If you did everything correctly, your content should be available at the specified domain.



Conclusion : Creating and maintaining a website using AWS is indeed a very easy task. Amazon's well-integrated ecosystem enables you to effortlessly register a domain, host your website, and ensure fast load times with the help of CloudFront.

Experiment: 07

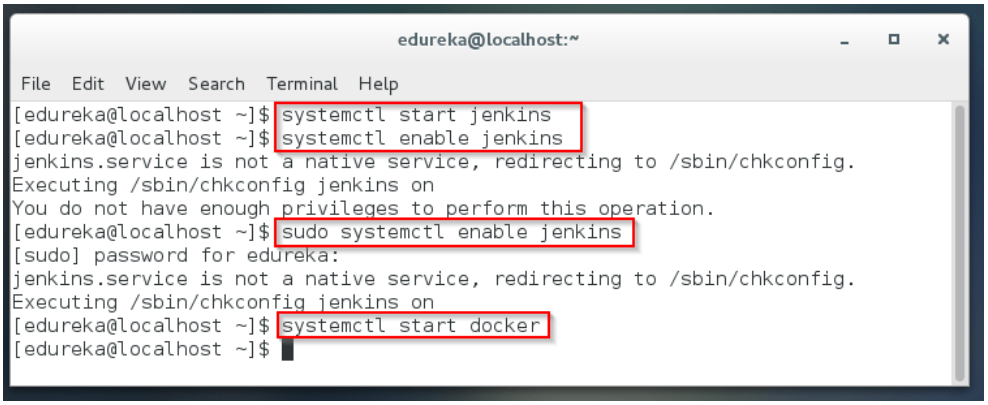
Aim: Use any suitable build tool Such as Jenkins, Bitbucket, GitHub, Actions or Cloud based services to Create a build pipeline having setups code build, test, code quality check.

Theory: Build tools are commonly known as **programs that automate the processof building an executable application from source code**. This building process includes activities like compiling, linking and packaging the code into an executable form.

Procedure:

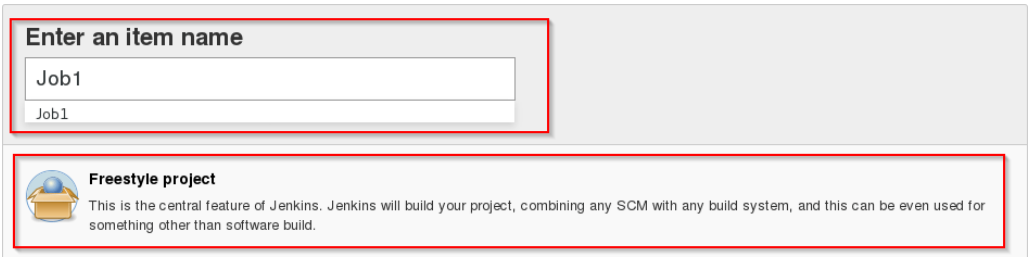
Step 1: Open your terminal in your VM. Start Jenkins and Docker using thesecommands:

systemctl start jenkins systemctl enable jenkinssystemctl start docker

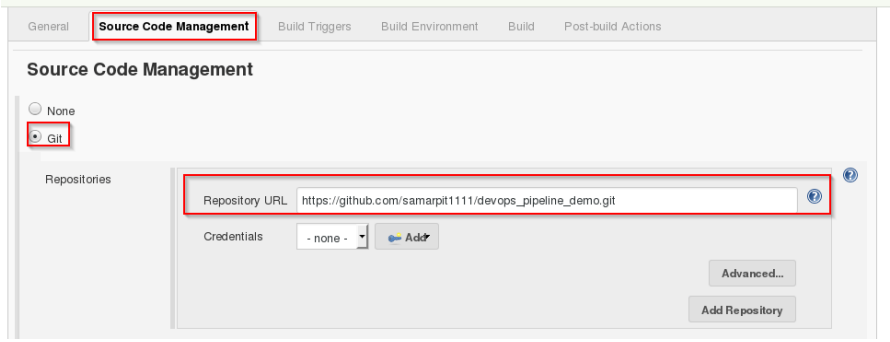


Step 2: Open Jenkins on your specified port. Click on **New Item** to create a Job.

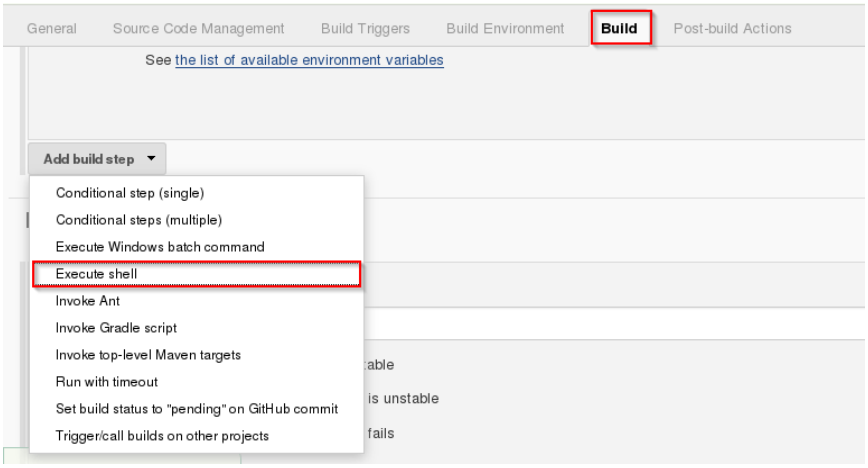
Step 3: Select a **freestyle** project and provide the item name (here I have givenJob1)and click OK.



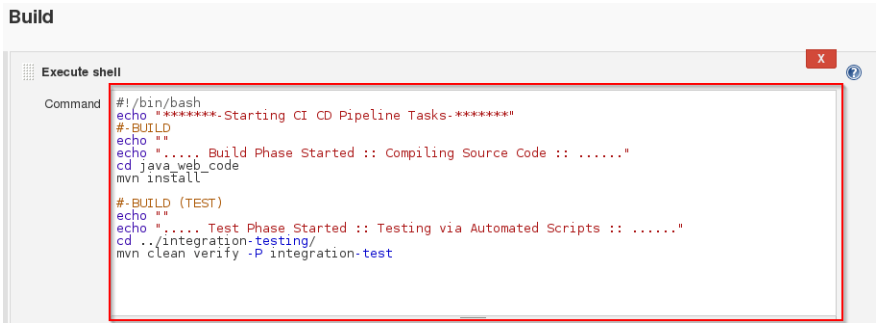
Step 4: Select **Source Code Management** and provide the **Git** repository. Click on **Apply** and **Save** button.



Step 5: Then click on **Build->Select Execute Shell**.

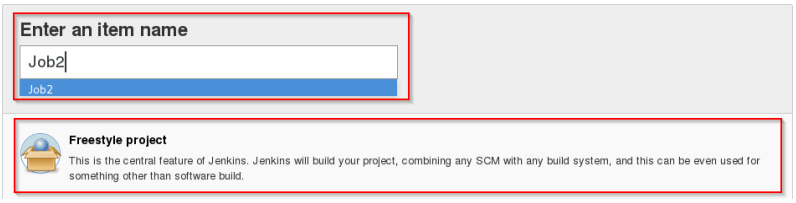


Step 6: Provide the shell commands. Here, it will build the archive file to get a war file. After that, it will get the code which is already pulled and then it uses maven to install the package. It simply installs the dependencies and compiles the application.

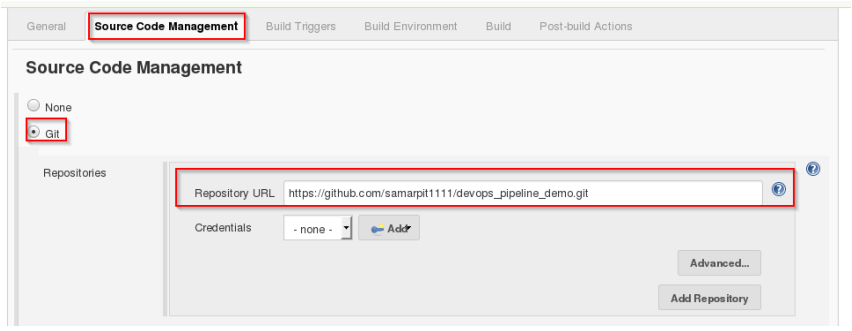


Step 7: Create the new **Job** by clicking on New Item.

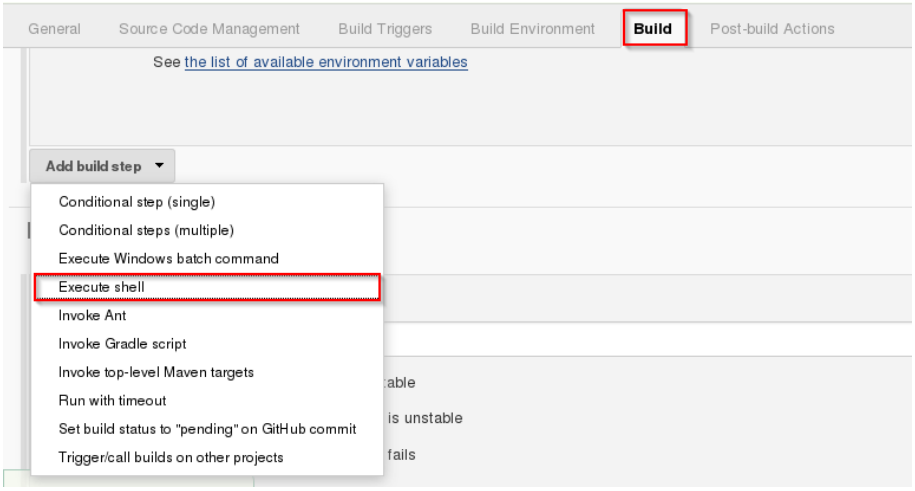
Step 8: Select **freestyle** project and provide the item name (here I have given Job2)and click on OK.



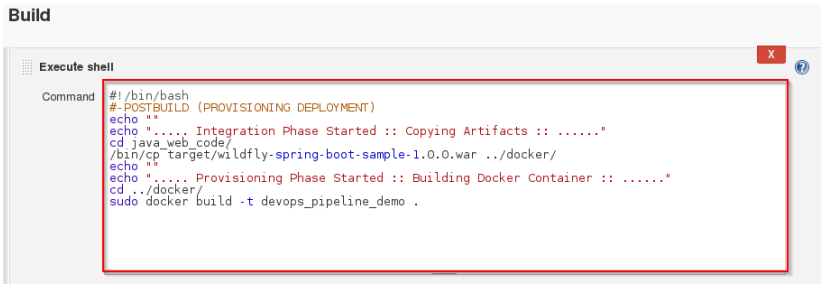
Step 9: Select **Source Code Management** and provide the **Git** repository. Clickon **Apply** and **Save** button.



Step 10: Then click on **Build->Select Execute Shell**.

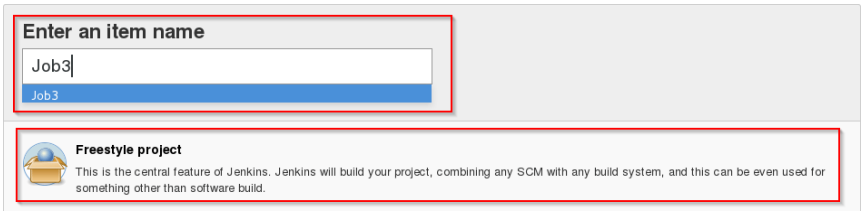


Step 11: Provide the shell commands. Here it will start the integration phaseand **build** the Docker Container.

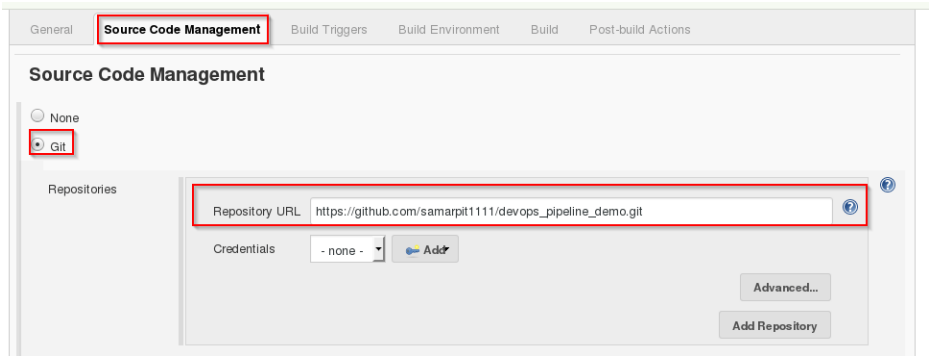


Step 12: Create the new **Job** by clicking on New Item.

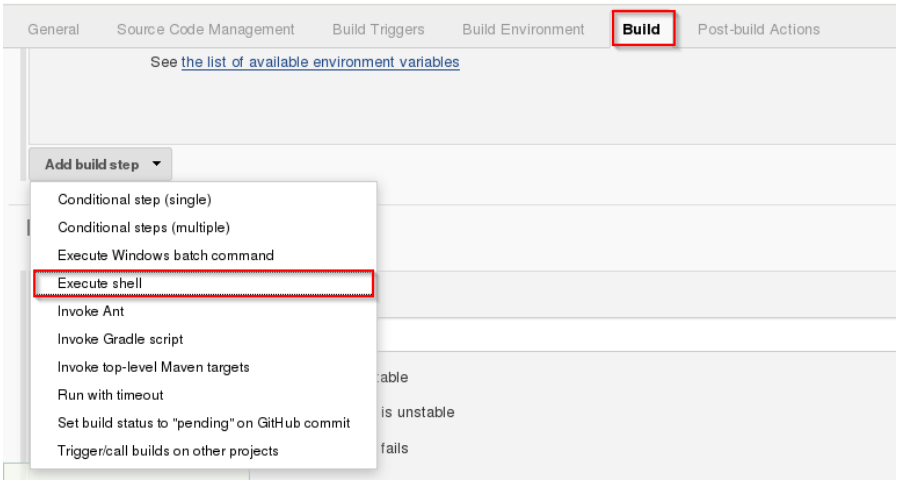
Step 13: Select **freestyle** project and provide the item name (here I have given Job3)and click on OK.



Step 14: Select **Source Code Management** and provide the **Git** repository. Clickon **Apply** and **Save** button.



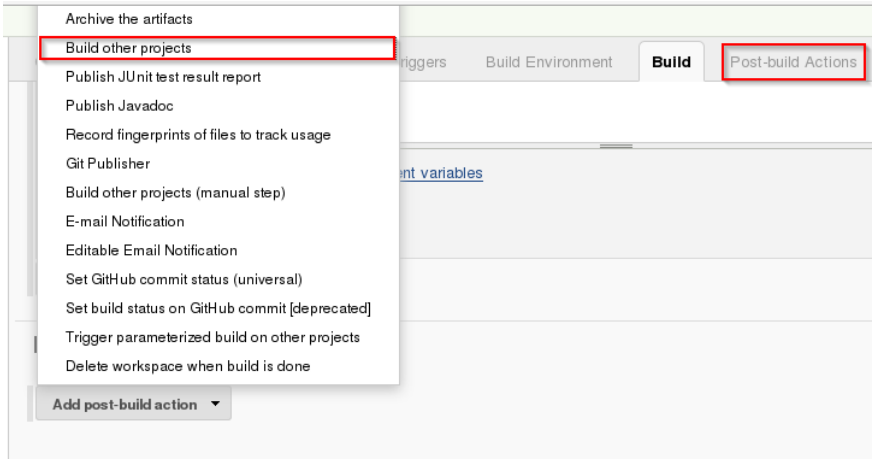
Step 15: Then click on **Build->Select Execute Shell**.



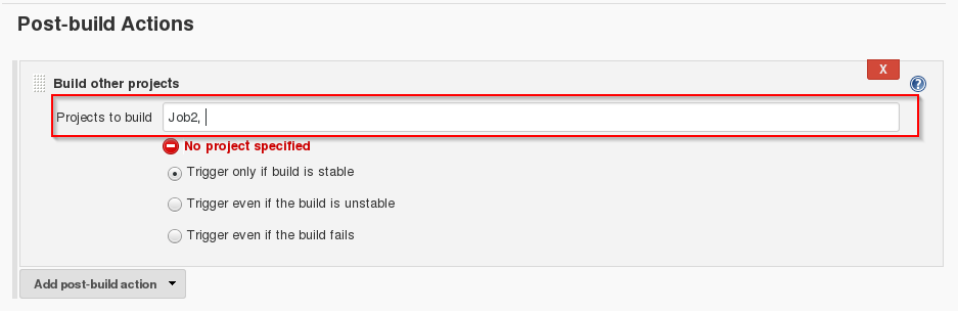
Step 16: Provide the shell commands. Here it will check for the Docker Container fileand then deploy it on port number 8180. Click on Save button.

Step 17: Now click on **Job1 -> Configure**.

Step 18: Click on **Post-build Actions -> Build other projects**.

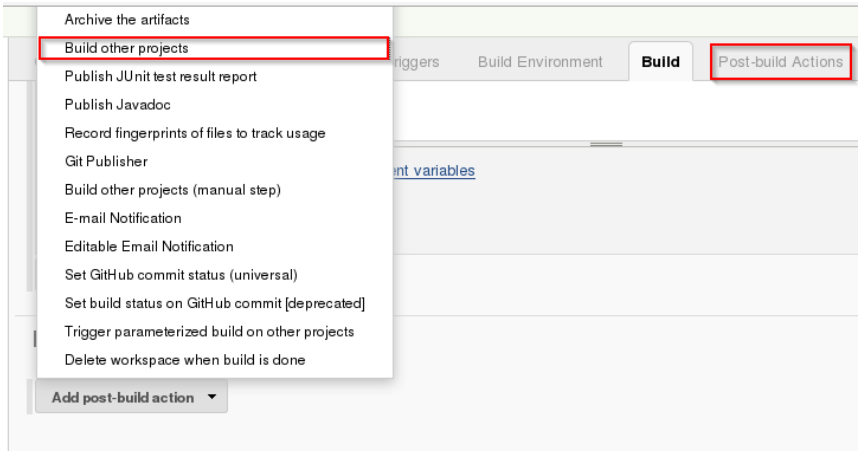


Step 19: Provide the project name to build after Job1 (here is Job2) and then clickon **Save**.

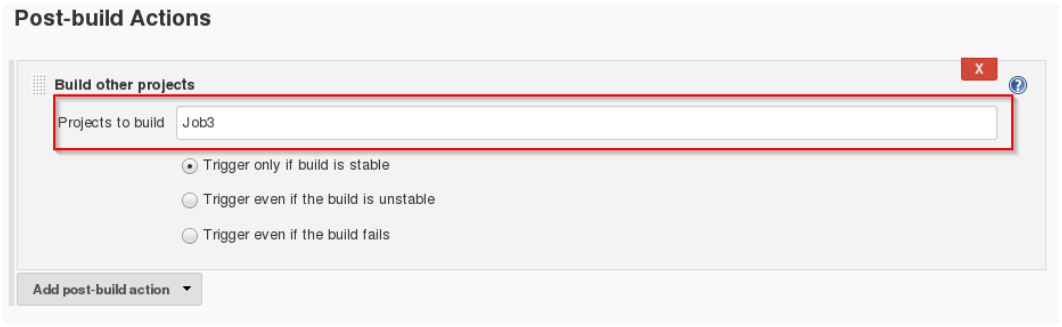


Step 20: Now click on **Job2 -> Configure**.

Step 21: Click on **Post-build Actions -> Build other projects**.

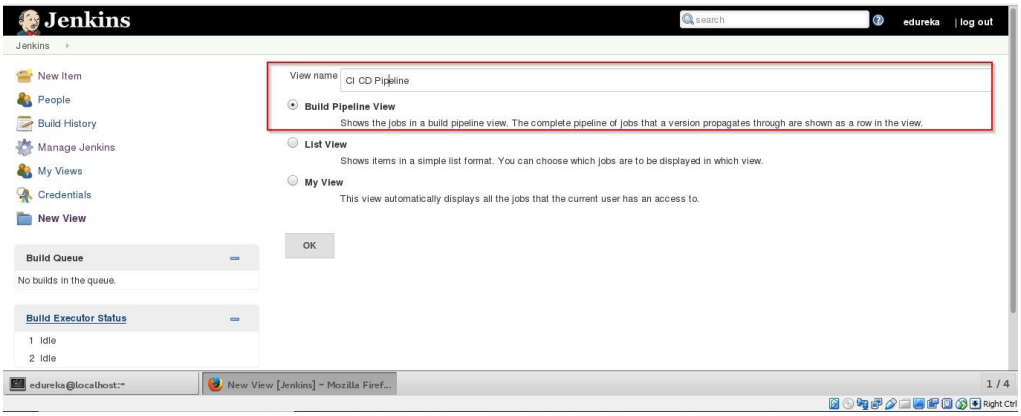


Step 22: Provide the project name to build after Job2 (here is Job3) and then clickon **Save**.

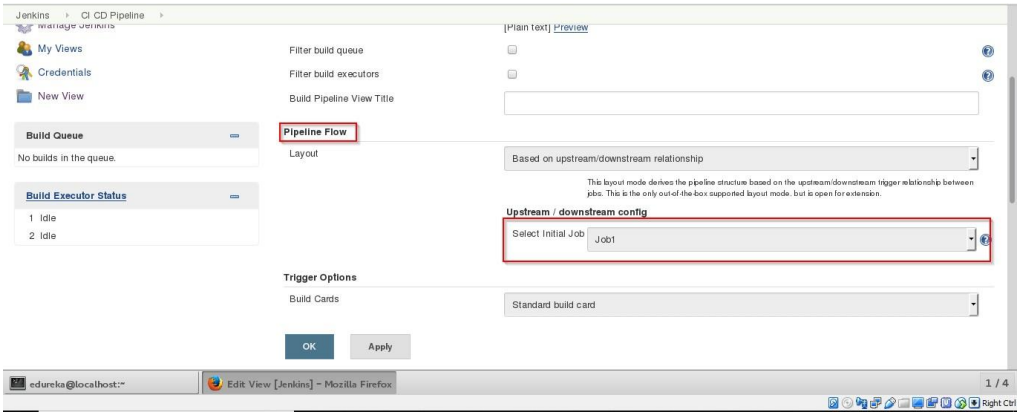


Step 23: Now we will be creating a Pipeline view. Click on the "+" sign.

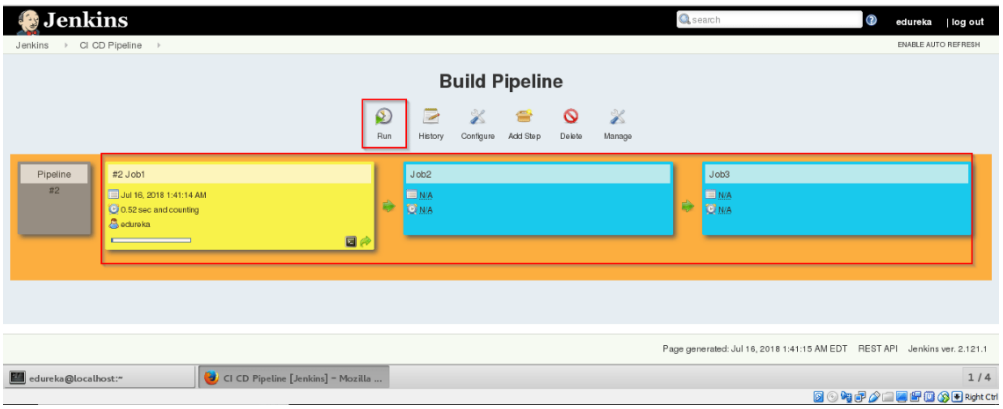
Step 24: Select **Build Pipeline View** and provide the view name (here I haveprovided CI CD Pipeline).



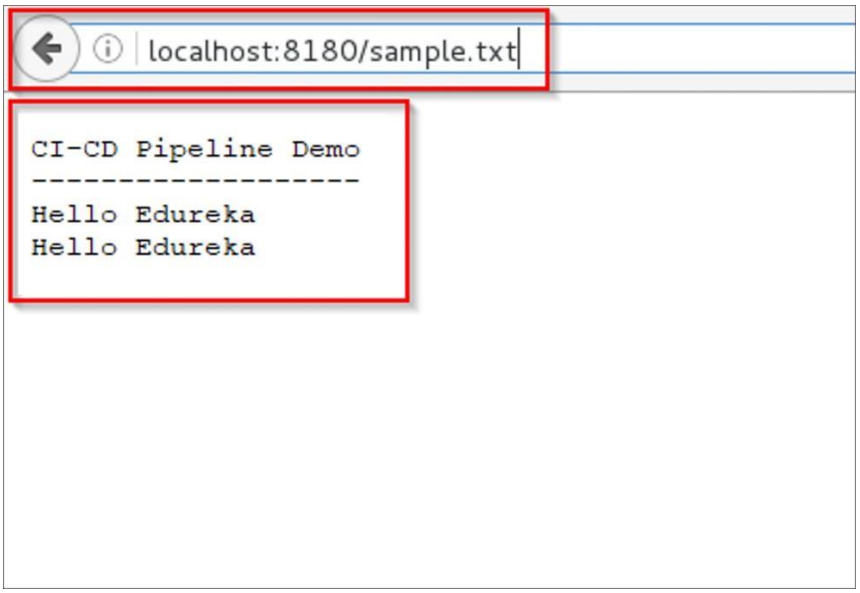
Step 25: Select the **initialJob** (here I have provided Job1) and click on OK.



Step 26: Click on **Run** button to start the CI/CD process.



Step 27: After successful build open **localhost:8180/sample.text**. It will run theapplication.


localhost:8180/sample.txt
CI-CD Pipeline Demo
Hello Edureka
Hello Edureka

Experiment: 08

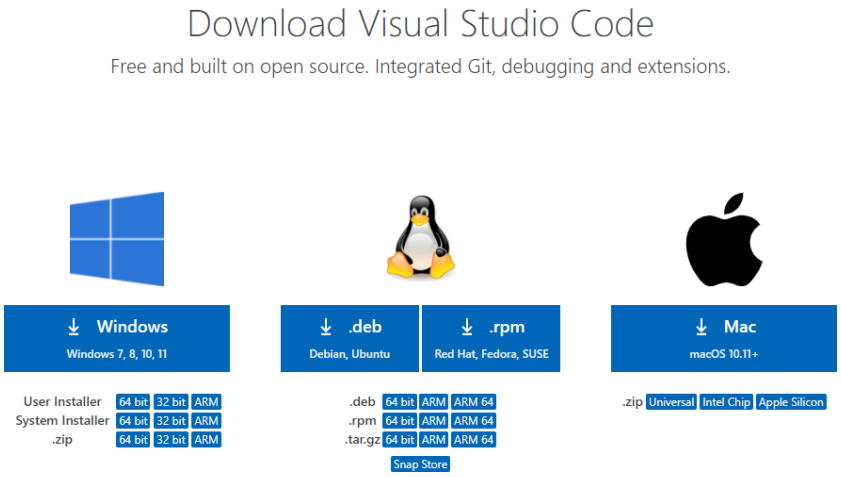
Aim: Installing VS code and its extensions.

Theory: Visual Studio Code (famously known as **VS Code**) is a free open-source text editor by Microsoft. VS Code is available for Windows, Linux, and macOS. Although the editor is lightweight, it includes some powerful features that have made VS Code one of the most popular development environment tools in recent times.

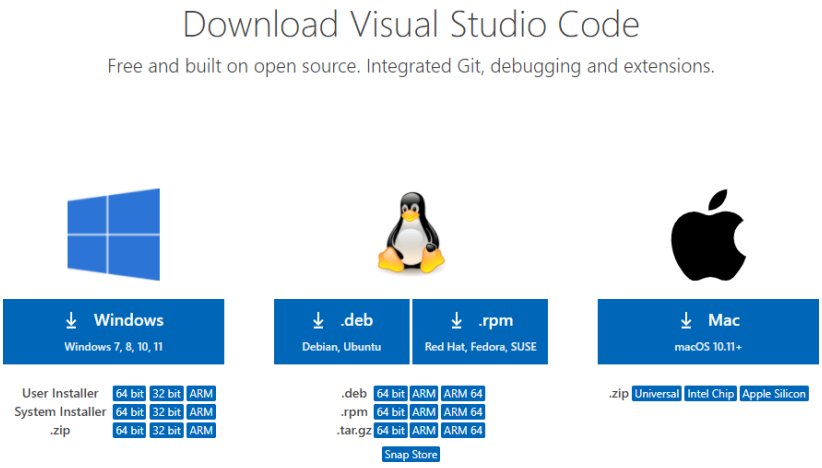
VS Code extensions **let you add languages, debuggers, and tools to your installation to support your development workflow**. VS Code's rich extensibility model lets extension authors plug directly into the VS Code UI and contribute functionality through the same APIs used by VS Code.

Procedure:

Step 1: Visit the official website of the **Visual Studio Code** using any web browser like Google Chrome, Microsoft Edge, etc.



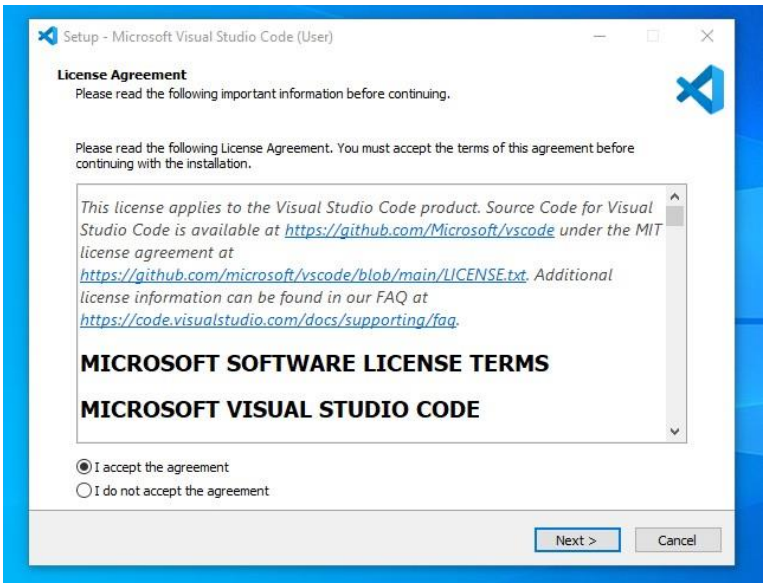
Step 2: Press the "**Download for Windows**" button on the website to start the download of the Visual Studio Code Application.



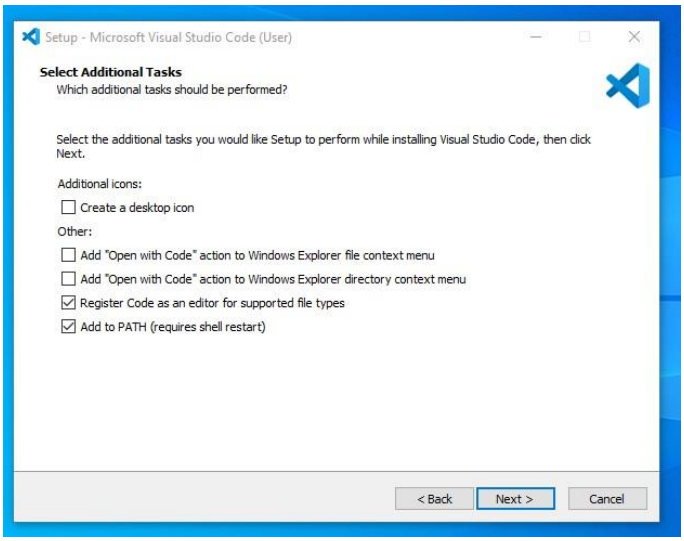
Step 3: When the download finishes, then the Visual Studio Code icon appears inthe downloads folder.

Step 4: Click on the installer icon to start the installation process of the Visual Studio Code.

Step 5: After the Installer opens, it will ask you for accepting the terms and conditions of the Visual Studio Code. Click on **I accept the agreement** and thenclick the **Next** button.

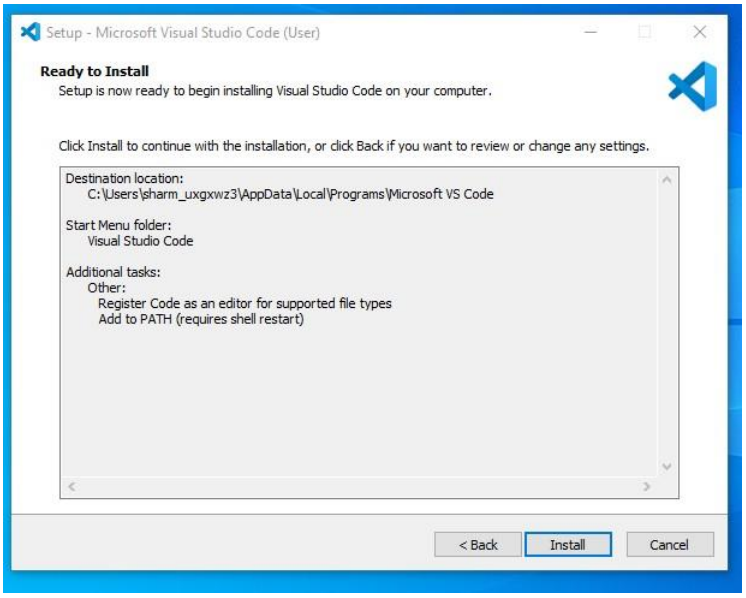


Step 6: Choose the location data for running the Visual Studio Code. It will then ask you for browsing the location. Then click on **Next** button.

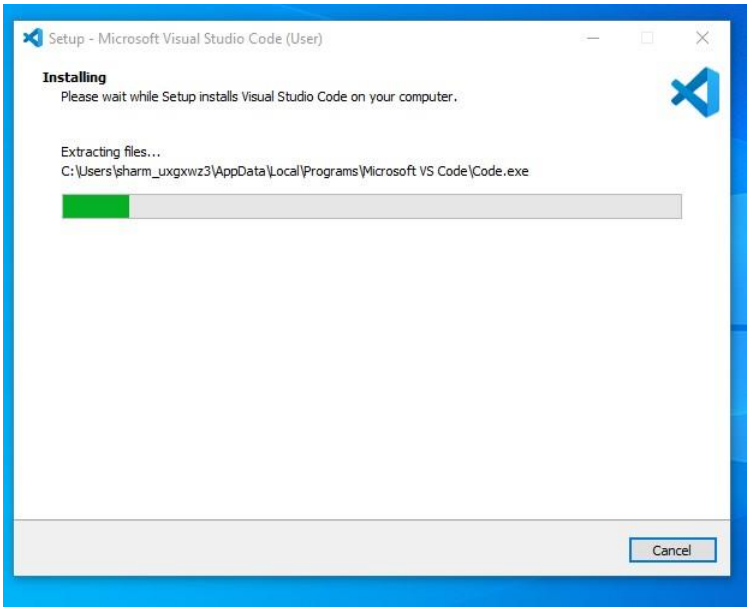


Step 7: Then it will ask for beginning the installing setup. Click onthe

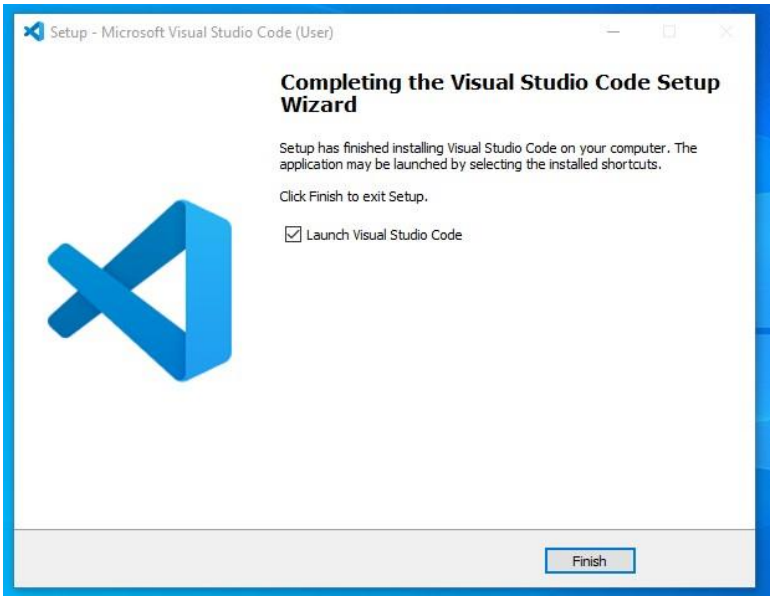
Install button.



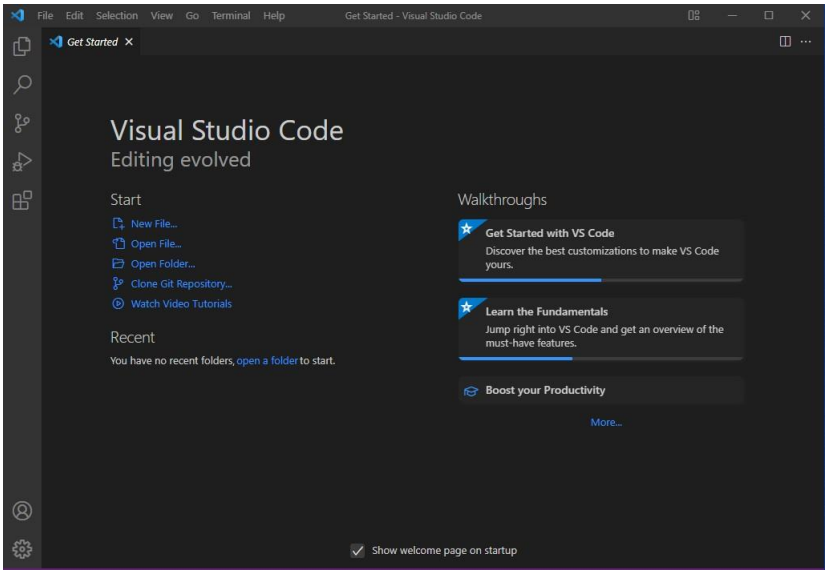
Step 8: After clicking on Install, it will take about 1 minute to install the VisualStudio Code on your device.



Step 9: After the Installation setup for Visual Studio Code is finished, it will show a window like this below. Tick the "**Launch Visual Studio Code**" checkbox and then click **Next**.



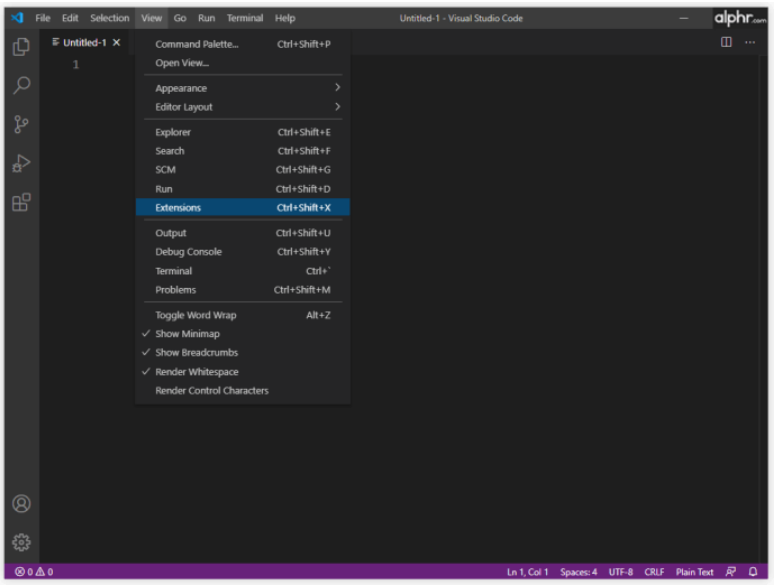
Step 10: After the previous step, the **Visual Studio Code window** opens successfully. Now you can create a new file in the Visual Studio Code window and choose a language of yours to begin your programming journey!



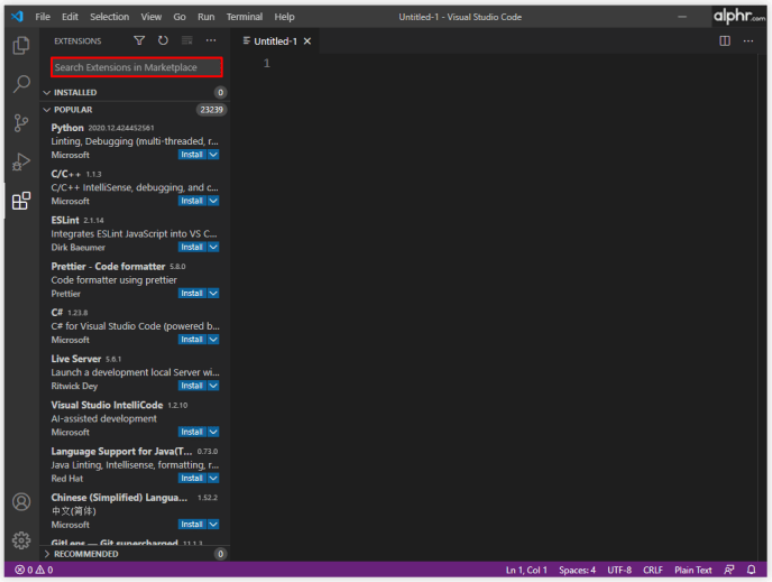
So this is how we successfully installed **Visual Studio Code** on our Windowssystem.

Now we will install VS code Extensions:

1. Click on the “Extensions” button in the Activity Bar. It’s located on the side of VS Code’s client. Alternatively, you can use the keyboard shortcut “Ctrl+Shift+X” to openthe “Extensions” screen.

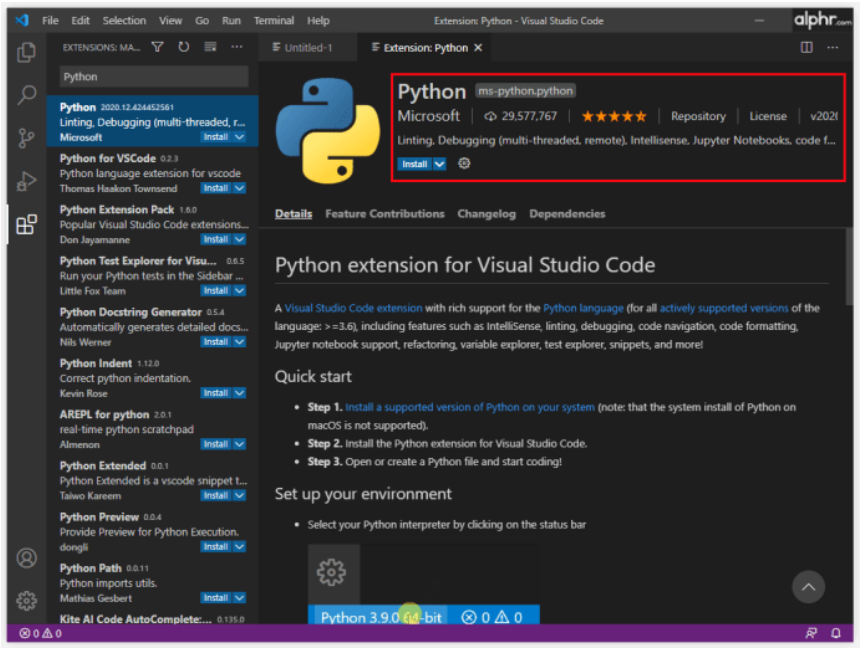


2. This will bring you to the "Extensions" list. VS Code automatically sorts Extensions by popularity.

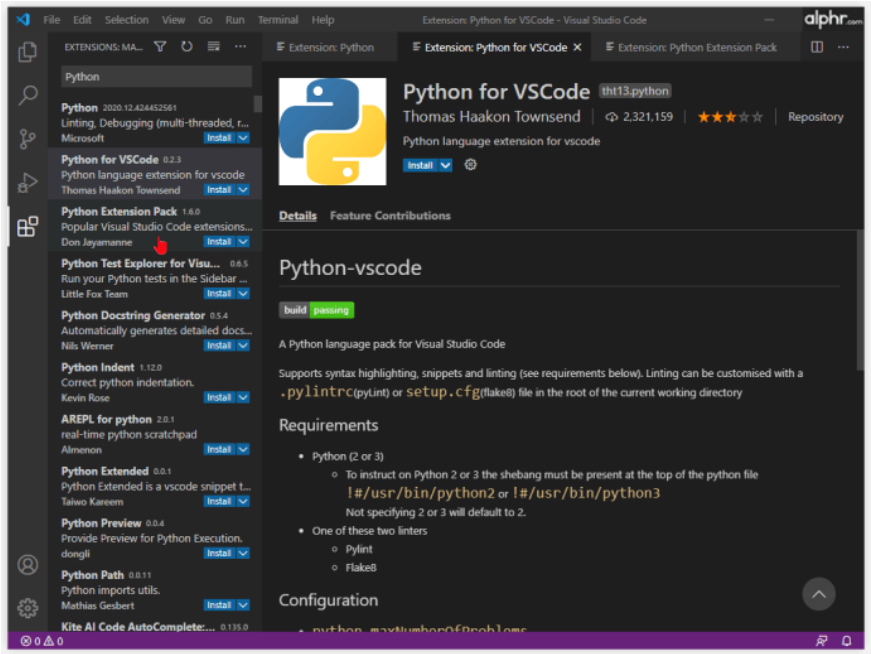


You can also use the search box on the top of the page to filter your results.

3. Each extension in the list will have a brief description, the download count (the number of times it has been downloaded), the publisher's name, and a rating from zero to five stars.



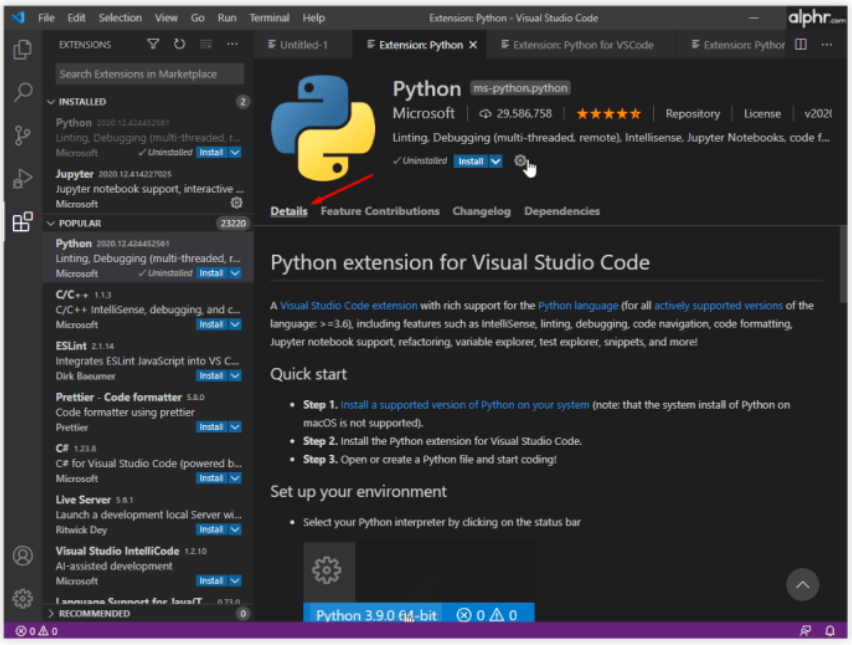
4. You can click each extension on the list to see more details before committing to a download. Details



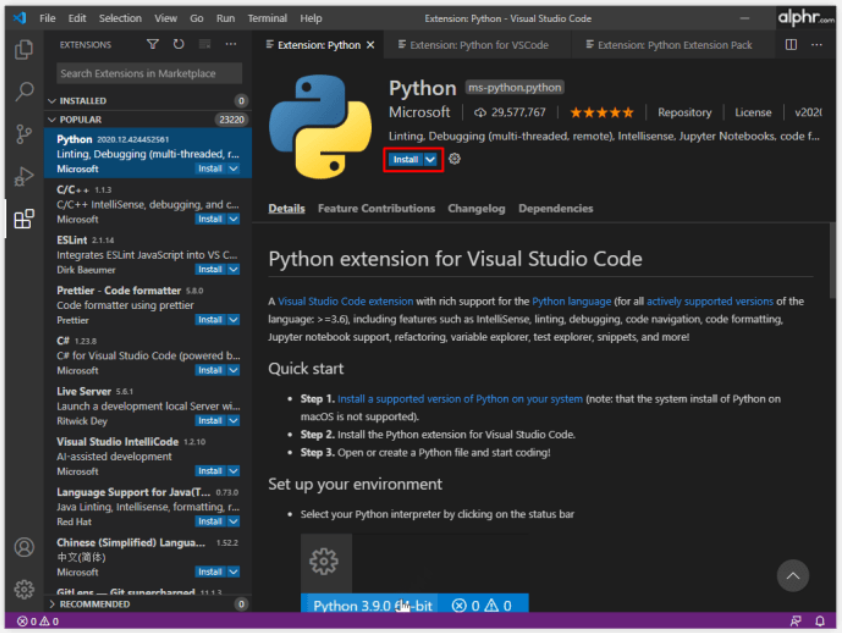
include a changelog, frequently asked questions, and a list of contributions and dependencies the extension gives to and requires from VS Code, respectively.

5. If you're using a proxy to access the internet, you'll need to set up VS Code touse a proxy server as well to connect to the internet and download extensions.

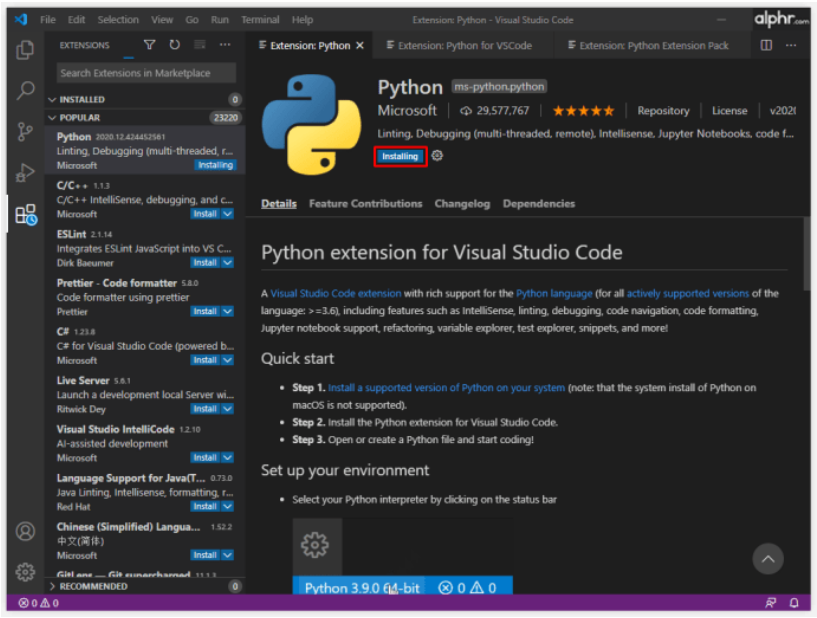
6. Once you've found an extension you like to install, click on its details.



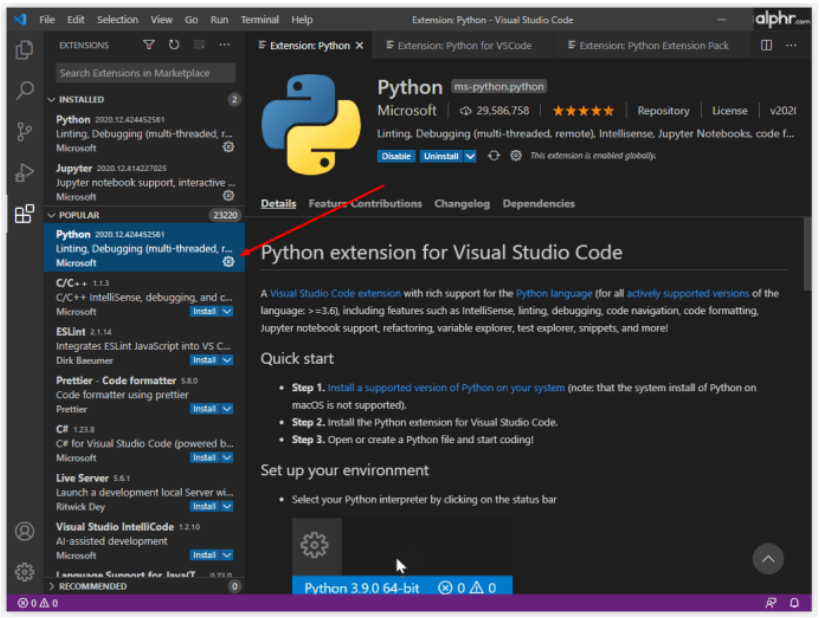
7. You'll see an "Install" button under the extension's name.



8. Click the "Install" button, and the extension will automatically download andinstall on your device.



9. The “Install” button will now change to a “Manage” button that looks like agear.



Experiment:09

Aim: Create and run "Hello World" program using TypeScript.

Theory: TypeScript is just JavaScript. TypeScript starts with JavaScript and ends with JavaScript. Typescript adopts the basic building blocks of your program from JavaScript. Hence, you only need to know JavaScript to use TypeScript. All TypeScript code is converted into its JavaScript equivalent for the purpose of execution.

Procedure:

Visual Studio Code includes TypeScript language support but does not include the TypeScript compiler, tsc. You will need to install the TypeScript compiler either globally or in your workspace to transpile TypeScript source code to JavaScript (tsc HelloWorld.ts).

The easiest way to install TypeScript is through npm, the Node.js Package Manager. Ifyou have npm installed, you can install TypeScript globally (-g) on your computer by:

```
npm install -g typescript
```

You can test your install by checking the version.

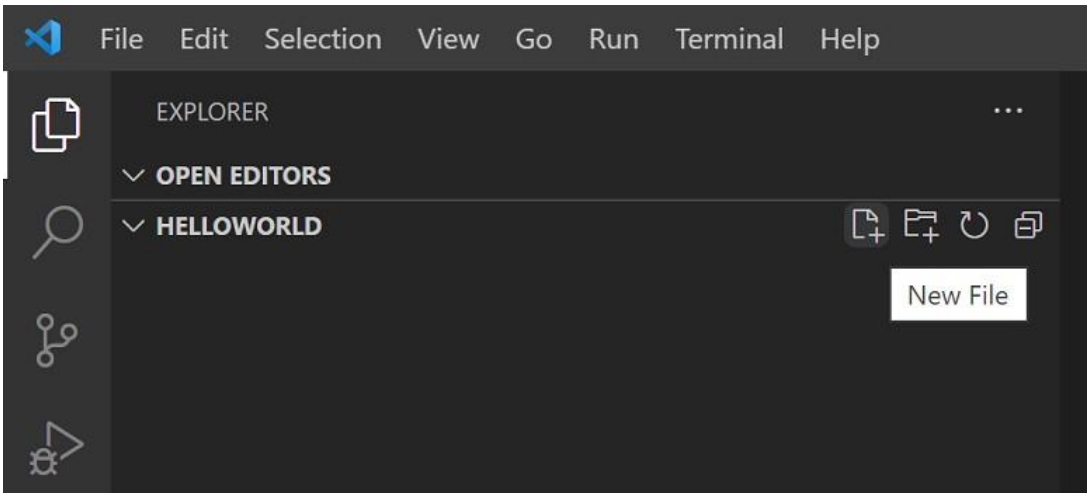
```
tsc --version
```

Hello World

Let's start with a simple Hello World Node.js example. Create a newfolder HelloWorld and launch VS Code.

```
mkdir HelloWorld

cd HelloWorld

code .
```

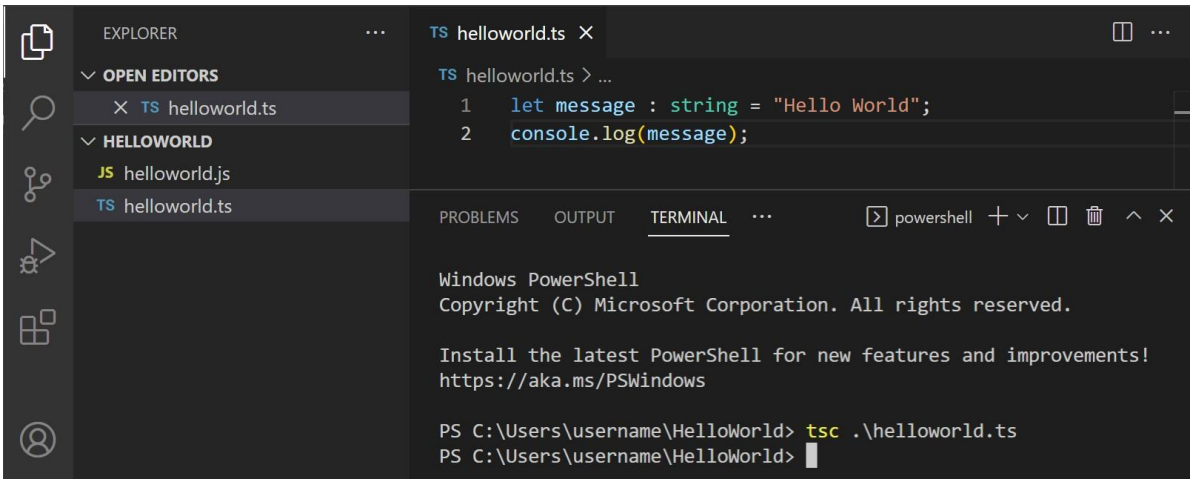
From the File Explorer, create a new file called helloworld.ts.



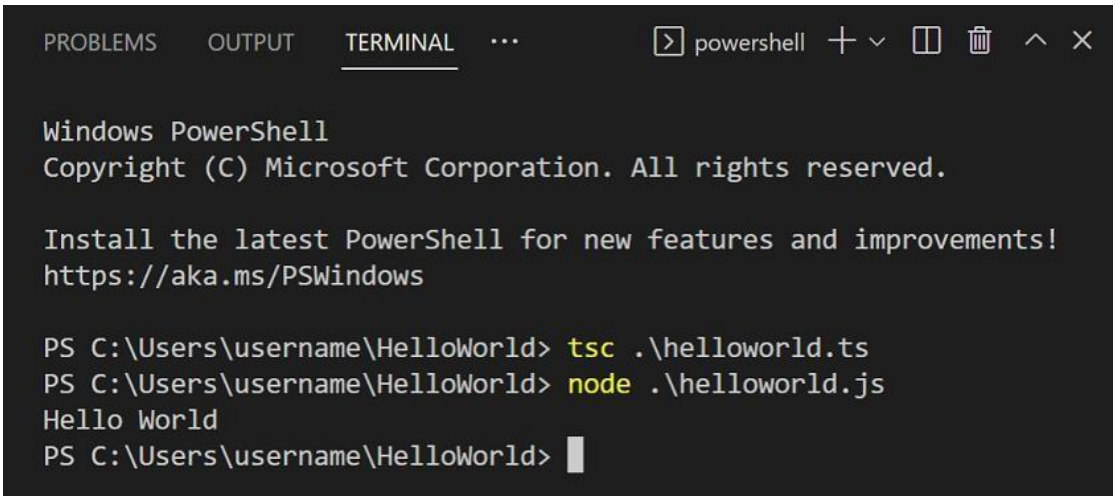
Now add the following TypeScript code. You'll notice the TypeScript keyword let andthe string type declaration.

```
let message: string = 'Hello World';

console.log(message);
```

To compile your TypeScript code, you can open the Integrated Terminal (Ctrl+`) and type tsc helloworld.ts. This will compile and create a new helloworld.js JavaScriptfile.



If you have Node.js installed, you can run node helloworld.js.



If you open helloworld.js, you'll see that it doesn't look very different from helloworld.ts. Thetype information has been removed and let is now var.

```
var message = 'Hello World';

console.log(message);
```

Experiment: 10

Aim: Create and run a React.js app.

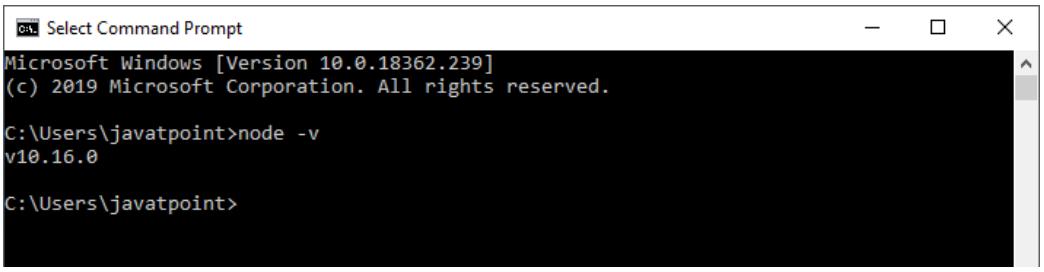
Theory: ReactJS is a **declarative**, **efficient**, and flexible **JavaScript library** for building reusable UI components. It is an open-source, component-based front end library which is responsible only for the view layer of the application. It was initially developed and maintained by Facebook and later used in its products like WhatsApp& Instagram.

Procedure:

Let us check the current version of **Node** and **NPM** in the system.

Run the following command to check the Node version in the command prompt.

$ node -v



Run the following command to check the NPM version in the command prompt.

$ npm -v

Installation

Here, we are going to learn how we can install React using ***CRA*** tool. For this, we need to follow the steps as given below.

Install React

We can install React using npm package manager by using the following command. There is no need to worry about the complexity of React installation. The create-react-app npm package manager will manage everything, which needed for React project.

C:\Users\javatpoint> npm install -g create-react-app

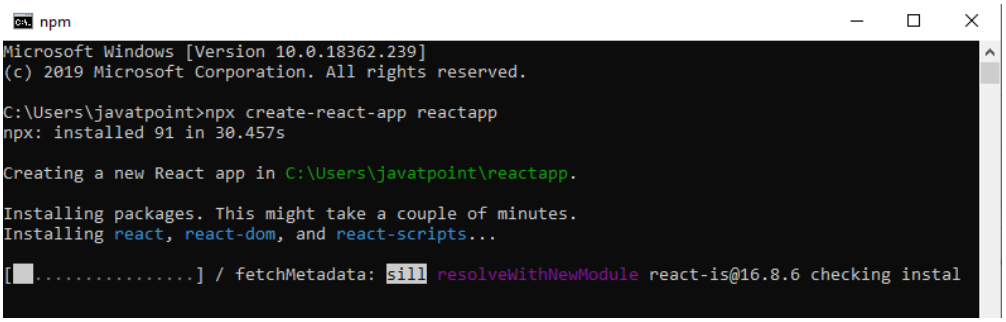
Create a new React project

Once the React installation is successful, we can create a new React project using create-react-app

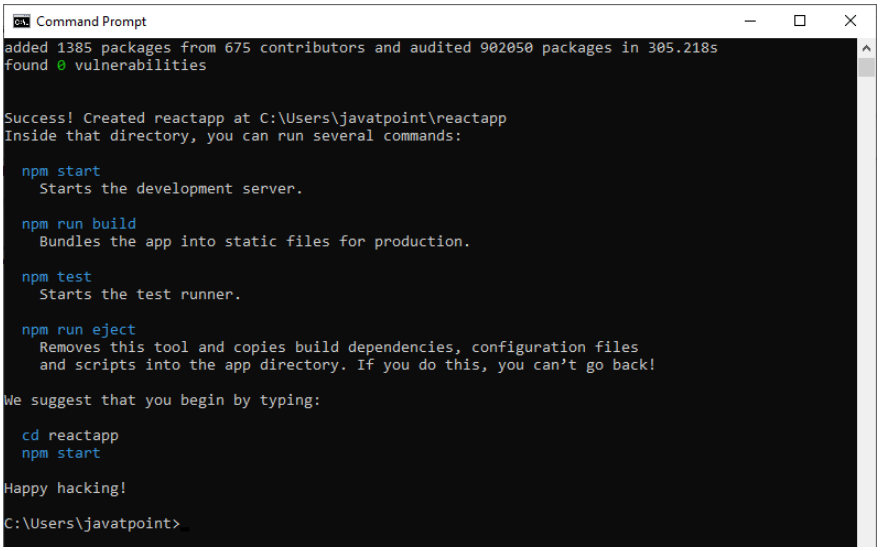
command. Here, I choose "reactproject" name for my project.

C:\Users\javatpoint> create-react-app reactproject

C:\Users\javatpoint> npx create-react-app reactproject



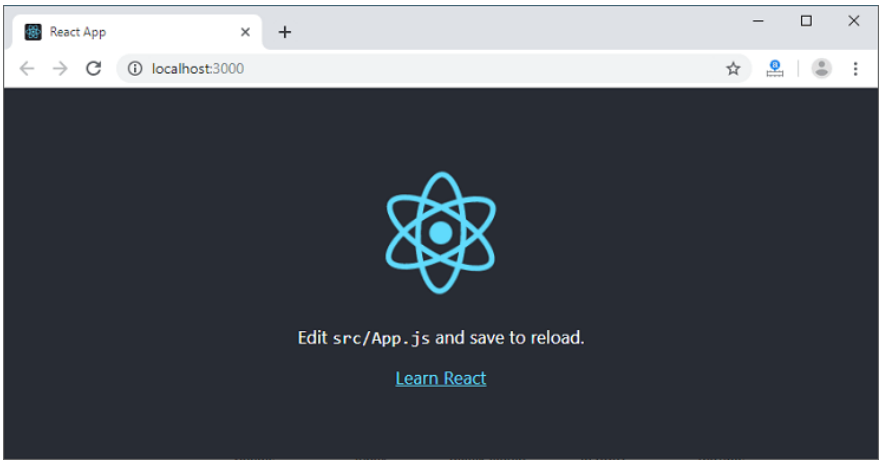
The above command will take some time to install the React and create a new project with the name "reactproject." Now, we can see the terminal as like below.



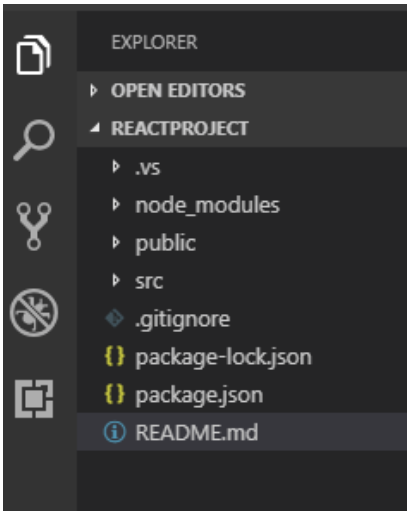
The above screen tells that the React project is created successfully on our system. Now, we need to start the server so that we can access the application on the browser. Type the following command in the terminal window.

1. $ cd Desktop
2. $ npm start

NPM is a package manager which starts the server and access the application atdefault server http://localhost:3000. Now, we will get the following screen.



Next, open the project on Code editor. Here, I am using Visual Studio Code. Ourproject's default structure looks like as below image.



Experiment: 11

Aim: Create your first React component.

Theory: Components are **independent and reusable bits of code**. They serve the same purpose as JavaScript functions, but work in isolation and return HTML. Components comein two types, Class components and Function components, in this tutorial we will concentrateon Function components.

Procedure:

1. Create a file *Jedi.jsx*, and give it the following content:import React from 'react';

```
class Jedi extends React.Component {render() {
 return (
 <div>I am a Jedi Component</div>
 );
 }
 }

 export default Jedi;
```

Above we are defining the class Jedi and have it inherit from React.Component. Thereafter we define the method render() that defines what our component will output. We return a JSXstatement as output.

Use component

Now that we have our component we can easily use it.

1. Open the file *index.js* and add the following row at the top:import Jedi from './Jedi';
1. Locate the part of the code that says ReactDOM.render and change it to look like so:

```
ReactDOM.render(
<Jedi />, document.getElementById('root')
);
```

The <Jedi> component has been added to the markup.

1. Test your project by running the following command at the root:npm start

2. Open up a browser and navigate to **http://localhost:8080**. You should see thefollowing text on the webpage:

I am a Jedi ComponentSuccess!

Experiment: 12

Aim: Implement navigation using React Router.

Theory: React Router is **a library built on top of React for enabling programmatic React navigation by handling routes in a web application**. The react navigation library allows the UI to be in sync with the browser URL by conditionally displaying components.

Procedure:

Installing React Router

As usual, installing a package using npm is as simple as running a single command:

$ npm install react-router-dom

Creating New Routes with React Router

The react-router-dom package makes it simple to create new routes. To begin, you wrap the entire application with the <BrowserRouter> tag. We do this to gain access to the browser's history object. Then you define your router links, as well as the components that will be used for each route.

To demonstrate this, let's create a new file called About.js in the /src folder: const About = () => {

```
return (
<div>
<h1>About page here!</h1>
<p>

Lorem ipsum dolor sit amet consectetur adipisicing elit. Fugit, modi!

</p>
</div>
);
};
export default About;
```

Now, let's update the src/index.js page and import About from the file we've just created. Within the <BrowserRouter> tag, we'll define our routes and components associated with them:

```
import { render } from "react-dom";

import { BrowserRouter, Routes, Route } from "react-router-dom";import App from "./App";

import About from "./About";
render(
<BrowserRouter>
<Routes>
<Route path="/" element={<App />} />
```

```
<Route path="about" element={<About />} />
</Routes>
</BrowserRouter>, document.getElementById("root")
);
```

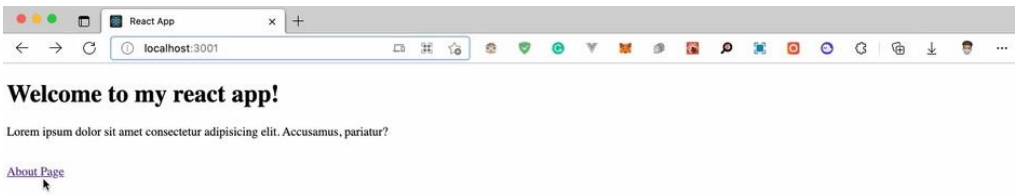
We've imported the <BrowserRouter> here, and we'd wrapped our entire application around it. We'd also selected App.js as the component for our home page (under the / endpoint),and About.js as the component for the /about page.

Finally, let's adapt the App.js file, which will, again, be the main entry point for theapplication and serve our home page:

```
import { Link } from "react-router-dom";function App() {
return (
<div className="App">
<h1>Welcome to my react app!</h1>
<p>
Lorem ipsum dolor sit amet consectetur adipisicing elit. Accusamus,pariatur?
</p>
<br />
<Link to="/about">About Page</Link>
</div>
);
}
export default App;
```

At this stage, we've created two routes: the entry route (/) and the /about route, and we should be able to easily navigate from the home page to the about page when we visit our application:

navigating between pages with react



Experiment:13

Aim: Build single page application like Shopping cart Using React Hooks.

Theory: React Hooks are **simple JavaScript functions that we can use to isolate the reusable part from a functional component**. Hooks can be stateful and can manage side- effects. React provides a bunch of standard in-built hooks: useState : To manage states. Returns a stateful value and an updater function to update it.

Procedure:

First, I will show an example of how to use the useState hook.import React, { useState } from "react";

```
const Shop = () => {
const [open, setOpen] = useState(true);console.log(open)


const closeStore = () => {setOpen(false)
}
return(
<div>
<input type="submit" value="close" onClick={() => closeStore()} />
</div>
)
}

export default Shop;
```

In this example, **open** is a key that holds the **useState** argument as its value.useState(**true**), **open = true**.

setOpen is a function that takes a value as an argument. **setOpen** will set **open** to the new value passed to **setOpen**.**setOpen(false)**, sets **open = false**

This shows a button that can be clicked in order to change the value of **open** from true tofalse.

Let's try a more complex use case.

In App.js we will return a div with our soon to be created Shop component:import React from "react";
import Shop from "./shop/Shop";

```
function App() {return (
<div>
<Shop />
</div>
);
}
export default App;
```

We will create the shop component next. Create a folder in src named shop. Then create a filein that folder named Shop.js

The finished Shop.js code is at the bottom of the article.

We are going to return an empty functional component to get us started:import React, { useState,

```
useEffect } from "react";
const Shop = () => {return <div />
}
export default Shop;
```

Let's add our inventory as an array labeled items:

```
const Shop = () => {
const items = [
{
id: 1,
name: "overwatch",price: 20,
},
{
id: 2,
name: "minecraft",price: 32,
},
{
id: 3,
name: "fortnite",price: 51,
},
];
return <div />
}
```

We are only selling these three video games. We need to display them. So, we will create anew formatted array called listItems using .map(). Now, we should return listItems:

```
const listItems = items.map((el) => (
<div key={el.id}>
{`${el.name}: $${el.price}`}
<input type="submit" value="add" onClick={() => addToCart(el)} />
</div>
));

eturn(<div>{listItems}</div>)
```

Above **items**, we will create our first useState hook:

```
const [cart, setCart] = useState([]);
```

The const **cart** is where we will hold our cart state. We can call **setCart()** and pass in thestate changes we want to make to **cart**. Let's create our addToCart function using setCart:

```
const addToCart = (el) =>
{
setCart([...cart, el]);
};
```

addToCart takes the element selected and adds it to the cart array.

We are going to display the cart, in our app, under our store. First, make a new formattedarray from the cart array:

```
const cartItems = cart.map((el) => (
<div key={el.id}>
{`${el.name}: $${el.price}`}
<input type="submit" value="remove" onClick={() => removeFromCart(el)} />
</div>
```

```
));
```

We can create our removeFromCart function using the filter method. note* We will make acopy of the cart state before filtering:

```
const removeFromCart = (el) => {let hardCopy = [...cart];
hardCopy = hardCopy.filter((cartItem) => cartItem.id !== el.id);setCart(hardCopy);
};
```

Change the return statement to include cartItems:return (

```
<div> STORE
<div>{listItems}</div>
<div>CART</div>
<div>{cartItems}</div>
</div>
);
```

Finally, we will keep track of the total using useState and useEffect:const [cartTotal, setCartTotal] = useState(0);

```
useEffect(() => {total();
}, [cart]);

const total = () => {
let totalVal = 0;
 for (let i = 0; i < cart.length; i++)
{
  totalVal += cart[i].price;
 }
setCartTotal(totalVal);
};
```

The useEffect hook contains an arrow function. Inside the arrow function, we call our totalfunction.

The second argument in useEffect is the dependency array containing [**cart**].

useEffect will detect changes in the variables named within its dependency array. When itdetects a change, it will run again.

Every time an item is added or removed from the cart, useEffect will detect a changein **cart** and run the total function.

Finally, place **total** in your return:

```
import React, { useState, useEffect } from "react";

const Shop = () => {
const [cart, setCart] = useState([]);
const [cartTotal, setCartTotal] = useState(0);const items = [
{
id: 1,
name: "overwatch",price: 20,
},
{
id: 2,
name: "minecraft",price: 32,
```

```
  },
  {
  id: 3,
  name: "fortnite",price: 51,
  },
  ];
useEffect(() => {total();
  }, [cart]);

const total = () => {let totalVal = 0;
  for (let i = 0; i < cart.length; i++)
  {
  totalVal += cart[i].price;
  }
  setCartTotal(totalVal);
  };
 const addToCart = (el) => {setCart([...cart, el]);
  };
const removeFromCart = (el) => {let hardCopy = [...cart];
  hardCopy = hardCopy.filter((cartItem) => cartItem.id !== el.id);setCart(hardCopy);
  };

  const listItems = items.map((el) => (
  <div key={el.id}>
  {`${el.name}: $${el.price}`}
  <input type="submit" value="add" onClick={() => addToCart(el)} />
  </div>
  ));

  const cartItems = cart.map((el) => (
  <div key={el.id}>
  {`${el.name}: $${el.price}`}
  <input type="submit" value="remove" onClick={() => removeFromCart(el)} />
  </div>
  ));

  return (
<div> STORE
  <div>{listItems}</div>
  <div>CART</div>
  <div>{cartItems}</div>
  <div>Total: ${cartTotal}</div>
  </div>
  );
  };

  export default Shop;
```

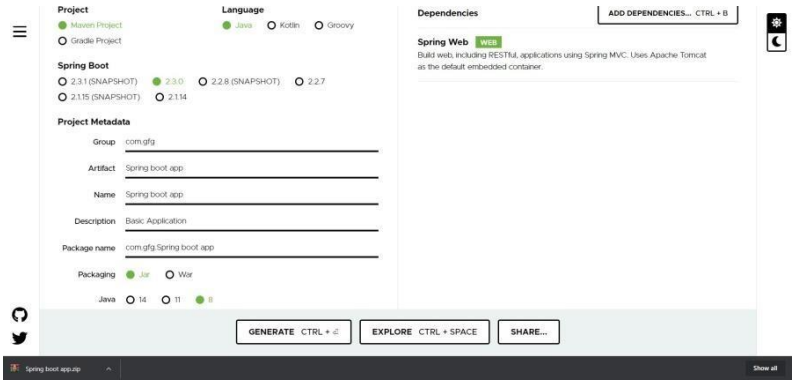
Experiment: 14

Aim: Create a Spring Application with Spring Initializr using dependencies the Spring Web,Spring Data JPA.

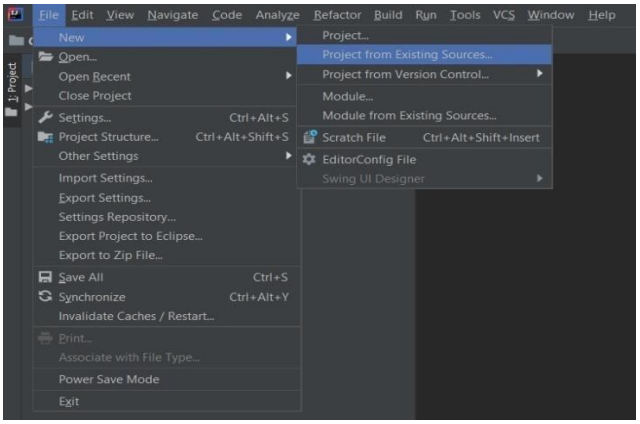
Theory: Spring Initializr **provides an extensible API to generate JVM-based projects, and to inspect the metadata used to generate projects, for instance to list the available dependencies and versions**.

Procedure:

Step 1: Go to Spring Initializr



Step 2: Fill in the details as per the requirements. For this application:
Project: Maven **Language:** Java **Spring Boot:** 2.2.8**Packaging:** JAR **Java:** 8
Dependencies: Spring Web and Spring Data JPA.
Step 3: Click on **Generate** which will download the starter project.
Step 4: Extract the zip file. Now open a suitable IDE and then go to File->New->Project from existing sources->Spring-boot-app and select pom.xml. Click on import changes on prompt and wait for the project to sync.

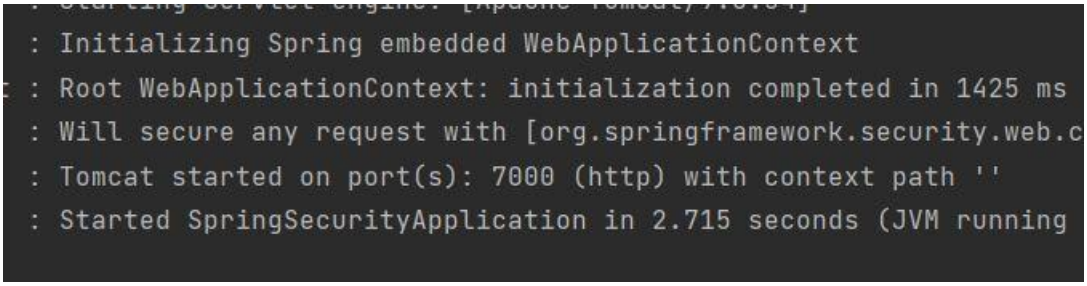


Note: *In the Import Project for Maven window, make sure you choose the same version ofJDK which you selected while creating the project.*

We need to change the port number using the application.properties file in the projectstructure of the spring application.

application.properties:
server.port=7000

Step 5: Go to ***src -> main -> java -> com.gfg.Spring.boot.app and run the mainapplication.***
Terminal output:



Tomcat is running on server 7000.

Experiment: 15

Aim: Demonstrate how to create RESTful Service.

Theory: RESTful Web Services are basically REST Architecture based Web Services. In REST Architecture everything is a resource. RESTful web services are light weight, highly scalable and maintainable and are very commonly used to create APIs for web-based applications. This tutorial will teach you the basics of RESTful Web Services and contains chapters discussing all the basic components of RESTful Web Services with suitable examples.

Procedure:

Creating a RESTful Web Service Using NetBeans IDE

This section describes, using a very simple example, how to create a Jersey-annotated web application from NetBeans IDE.

1. In NetBeans IDE, create a simple web application. This example creates a very simple“Hello, World” web application.

a. Open NetBeans IDE.
b. Select File->New Project.
c. From Categories, select Java Web. From Projects, select Web Application.Click Next.

Note – For this step, you could also create a RESTful web service in a Maven web project by selecting Maven as the category and Maven Web Project as the project. The remaining steps would be the same.

d. Enter a project name, **HelloWorldApplication**, click Next.
e. Make sure the Server is Sun GlassFish v3 (or similar wording.)
f. Click Finish. You may be prompted for your server Administrator User Nameand Password. If so, enter this information.

2. The project will be created. The file index.jsp will display in the Source pane.
3. Right-click the project and select New, then select RESTful Web Services fromPatterns.

a. Select Simple Root Resource. Click Next.
b. Enter a Resource Package name, like **helloWorld**.
c. Enter **helloworld** in the Path field. Enter **HelloWorld** in the Class Name field.For MIME Type select text/html.
d. Click Finish.
e. The REST Resources Configuration page displays. Select OK.

A new resource, HelloWorld.java, is added to the project and displays in theSource pane. This file provides a template for creating a RESTful web service.

4. In HelloWorld.java, find the getHtml() method. Replace the //TODO comment andthe exception with the following text, so that the finished product resembles the following method.

Note –Because the MIME type that is produces is HTML, you can use HTML tags in yourreturn statement.

```
/**
* Retrieves representation of an instance of helloWorld.HelloWorld
* @return an instance of java.lang.String
*/ @GET
@Produces("text/html") public String getHtml() {
return "<html><body><h1>Hello, World!!</body></h1></html>";
}
```

5. Test the web service. To do this, right-click the project node and click Test RESTfulWeb Services.

   This step will deploy the application and bring up a test client in the browser.

6. When the test client displays, select the helloworld resource in the left pane, and clickthe Test button in the right pane.

   The words Hello, World!! will display in the Response window below.

7. Deploy and Run the application.

a. Set the Run Properties. To do this, right-click the project node, select Properties, and then select the Run category. Set the Relative URL to the location of the RESTful web service relative to the Context Path, which for this example is resources/helloworld.

   Tip –
   You can find the value for the Relative URL in the Test RESTful Web Services browser window. In the top of the right pane, after Resource, is the URL for the RESTful web service being tested. The part following the Context Path (http://localhost:8080/HelloWorldApp) is the Relative URL that needs to be entered here.
   If you don't set this property, by default the file index.jsp will display when the application is run. As this file also contains Hello World as its default value, you might not notice that your RESTful web service isn't running, so just be aware of this default and the need to set this property, or update index.jsp to provide a link to the RESTful web service.

b. Right-click the project and select Deploy.
c. Right-click the project and select Run.

   A browser window opens and displays the return value of Hello, World!!

Experiment: 16

Aim: Create Spring REST controller with:

- Controller Layer
- Service Layer
- Repository Layer

Theory: Spring RestController annotation is **used to create RESTful web services usingSpring MVC**. Spring RestController takes care of mapping request data to the defined request handler method. Once response body is generated from the handler method, it converts it to JSON or XML response.

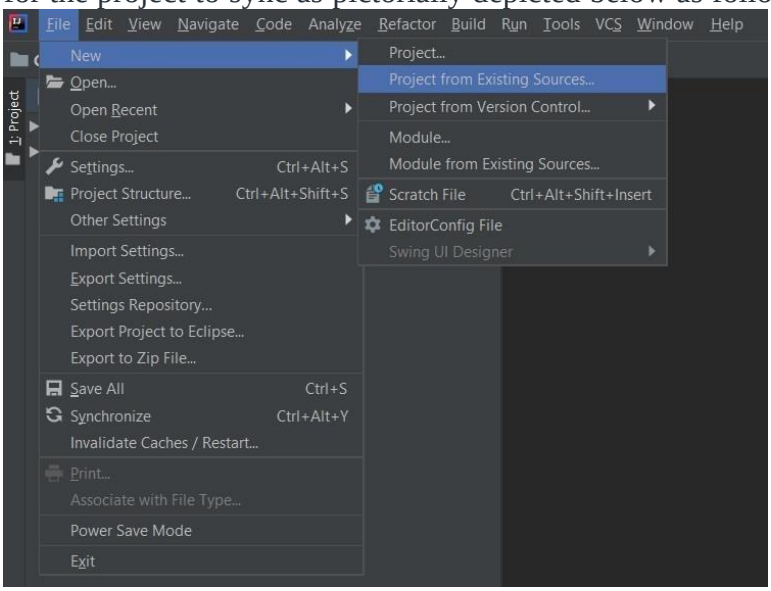
Procedure:

Step 1: Go to Spring Initializr
Fill in the details as per the requirements. For this application:

Project: Maven Language: Java Spring Boot: 2.2.8Packaging: JAR Java: 8

Dependencies: Spring Web

Step 2: Click on Generate which will download the starter project
Step 3: Extract the zip file. Now open a suitable IDE and then go to File > New > Project from existing sources > Spring-boot-app and select pom.xml. Click on import changes on prompt and wait for the project to sync as pictorially depicted below as follows:



Note: *In the Import Project for Maven window, make sure you choose the same version ofJDK which you selected while creating the project.*

Step 4: Go to src > main > java > com.gfg.Spring.boot.app, create a java class with the name Controller and add the annotation @RestController and other class named as Details.**Details:**

```
Java
public class Details {


	// Creating an object of ArrayList
	static ArrayList<Details> Data = new ArrayList<Details>();
	int number;
	String name;
	Details(int number, String name)
	{
		// This keyword refers
		// to parent instance itself
		this.number = number;
		this.name = name;
	}
}
```

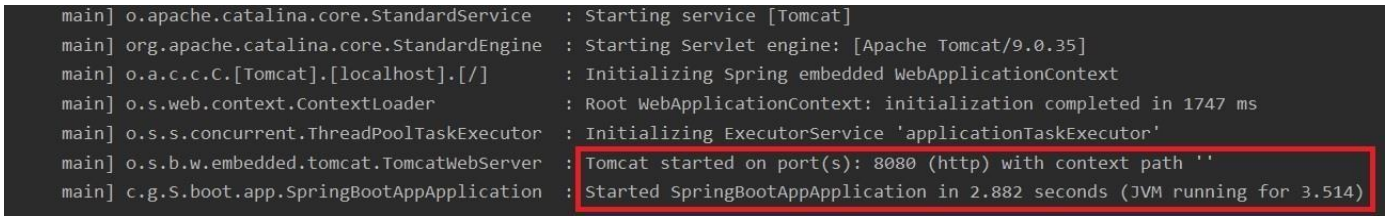
Controller:

```
@RestController
// Class
public class Controller {
// ConstructorController()
{
a.add(1);
a.add(2);
}

@GetMapping("/hello/{name}/{age}")
public void insert(@PathVariable("name") String name,
@PathVariable("age") int age)
{

// Print and display name and age
```

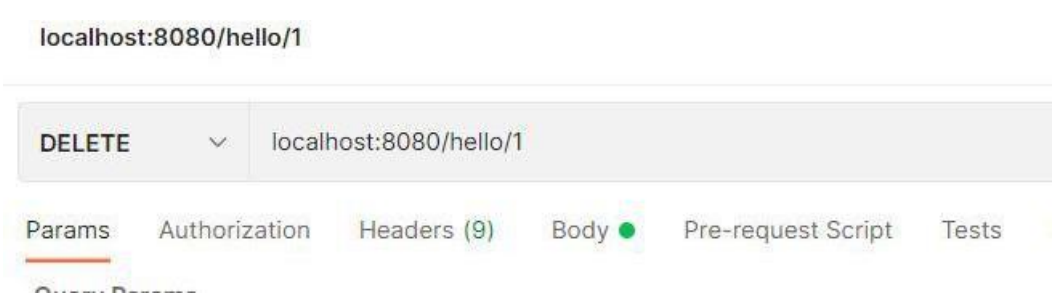
```
System.out.println(name);System.out.println(age);
}

// Creating an empty ArrayList ArrayList<Integer> a = new ArrayList<>();

// Annotation @DeleteMapping("/hello/{id}")
// Method
public void deleteById(@PathVariable("id") int id)
{
a.remove(new Integer((id)));print();
}

// Handling post request @PostMapping("/EnterDetails")
String insert(@RequestBody Details ob)
{
// Storing the incoming data in the list Data.add(new Details(ob.number, ob.name));

// Iterating using foreach loopfor (Details obd : Data) {
System.out.println(obd.name + " " + ob.number);
}
return "Data Inserted";
}

// Method void print()
{
for (int elements : a) {
System.out.print(elements);
}
}
```

This application is now ready to run.

Step 5: Run the SpringBootAppApplication class and wait for the Tomcat server to start.



Let’s make a delete request from the postman



Output: As generated on console2

Experiment: 17

Aim: Create User Registration Form with Spring Rest Controller and Versioning RESTAPI.

Theory: Versioning is the most important and difficult part of the API as it takes backward API compatible. Versioning helps us to iterate faster when the changes are identified. We should always version our Web API.

Consider a scenario in which we have a Web API that is up (status) and running. The users are consuming that API. Now we want to add more functionality in the Web API but want to keep the existing functionality unchanged. There may be few users who still want to use the old API while the other users want a new version of API with new or extended features. It is the scenario where Web API versioning comes into existence.

Procedure:

1. Create a Spring boot application

Spring Boot provides a web tool called **Spring Initializer** to bootstrap an application quickly.Just go to **https://start.spring.io/** and generate a new spring boot project.

Use the below details in the Spring boot creation:

Project Name: springboot-blog-rest-api

Project Type: Maven

Choose dependencies: Spring Web, Lombok, Spring Data JPA, Spring Security, Dev Toolsand MySQL Driver

Package name: net.javaguides.springboot

Packaging: Jar

Download the Spring Boot project as a zip file, unzip it and import it in your favorite IDE.

2. Maven Dependencies

Here is the pom.xml file for your reference:

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
https://maven.apache.org/xsd/maven-4.0.0.xsd">
        <modelVersion>4.0.0</modelVersion>
        <parent>
                <groupId>org.springframework.boot</groupId>
```

```
<artifactId>spring-boot-starter-parent</artifactId>
<version>3.0.0</version>
<relativePath/> <!-- lookup parent from repository -->
</parent>
<groupId>com.springboot.blog</groupId>
<artifactId>springboot-blog-rest-api</artifactId>
<version>0.0.1-SNAPSHOT</version>
<name>springboot-blog-rest-api</name>
<description>Spring boot blog application rest api&amp;apos;s</description>
<properties>
<java.version>17</java.version>
</properties>
<dependencies>
<dependency>
<groupId>org.springframework.boot</groupId>
<artifactId>spring-boot-starter-data-jpa</artifactId>
</dependency>
<dependency>
<groupId>org.springframework.boot</groupId>
<artifactId>spring-boot-starter-web</artifactId>
</dependency>

<dependency>
<groupId>org.springframework.boot</groupId>
<artifactId>spring-boot-devtools</artifactId>
<scope>runtime</scope>
<optional>true</optional>
</dependency>
<dependency>
<groupId>mysql</groupId>
<artifactId>mysql-connector-java</artifactId>
<scope>runtime</scope>
</dependency>
<dependency>
<groupId>org.projectlombok</groupId>
<artifactId>lombok</artifactId>
<optional>true</optional>
</dependency>
<dependency>
<groupId>org.springframework.boot</groupId>
<artifactId>spring-boot-starter-validation</artifactId>
</dependency>
<dependency>
<groupId>org.springframework.boot</groupId>
<artifactId>spring-boot-starter-security</artifactId>
</dependency>
<dependency>
<groupId>org.springframework.boot</groupId>
<artifactId>spring-boot-starter-test</artifactId>
```

```
                        <scope>test</scope>
                    </dependency>
                </dependencies>

                <build>
                    <plugins>
                        <plugin>
                            <groupId>org.springframework.boot</groupId>
                            <artifactId>spring-boot-maven-plugin</artifactId>
                            <configuration>
                                <excludes>
                                    <exclude>

                <groupId>org.projectlombok</groupId>

                <artifactId>lombok</artifactId>
                                    </exclude>
                                </excludes>
                            </configuration>
                        </plugin>
                    </plugins>
                </build>

</project>
```

3. Configure MySQL Database

Let's first create a database in MySQL server using the below command:

```
create database myblog
```

Since we're using MySQL as our database, we need to configure the database **URL**, **username**, and **password** so that Spring can establish a connection with thedatabase on startup. Open src/main/resources/application.properties file and add the following properties to it:

```
spring.datasource.url =
jdbc:mysql://localhost:3306/myblog?useSSL=false&serverTimezone=UTC
spring.datasource.username = root
spring.datasource.password = root

# hibernate properties
spring.jpa.properties.hibernate.dialect = org.hibernate.dialect.MySQLDialect

# Hibernate ddl auto (create, create-drop, validate, update)
spring.jpa.hibernate.ddl-auto = update
```

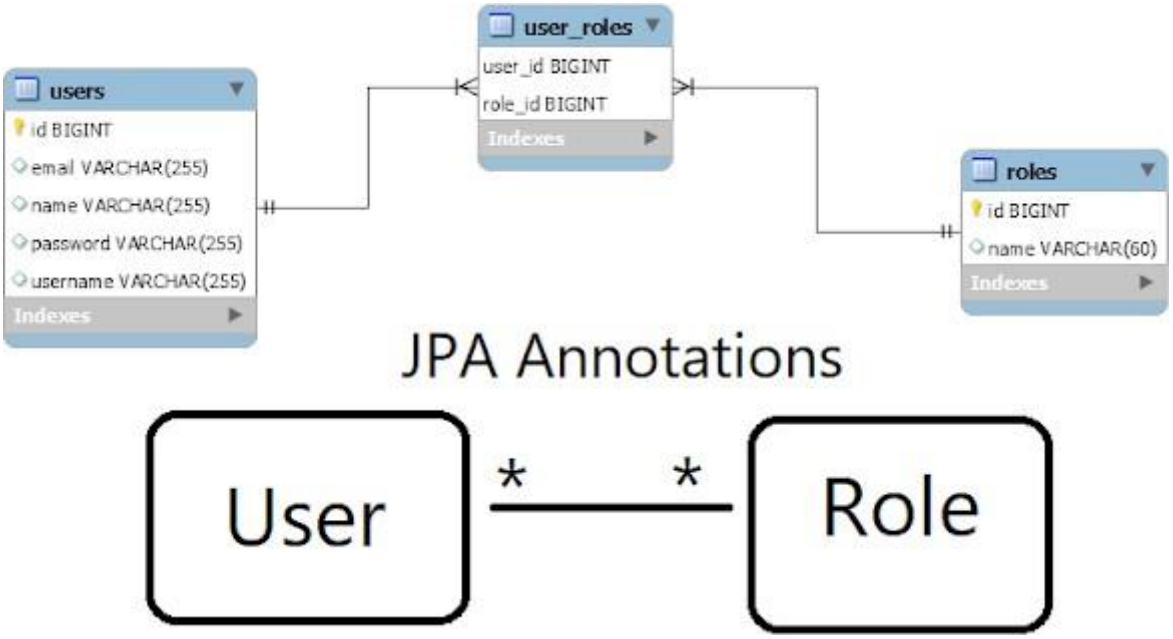
```
logging.level.org.springframework.security=DEBUG
```

Don't forget to change the spring.datasource.username and spring.datasource.password as per your MySQL installation.

You don't need to create any tables. The tables will automatically be created by Hibernate from the Employee entity that we will define in the next step. This is made possible by the property spring.jpa.hibernate.ddl-auto = update.

4. Model Layer - Create JPA Entities

In this step, we will create *User* and *Role* JPA entities and establish **MANY-to- MANY** relationships between them. Let's use JPA annotations to establish **MANY-to-MANY** relationships between *User* and *Role* entities.



User JPA Entity

```
package com.springboot.blog.entity;

import lombok.Data;

import jakarta.persistence.*;
import java.util.Set;

@Data
@Entity
@Table(name = "users", uniqueConstraints = {
```

```
        @UniqueConstraint(columnNames = {"username"}),
        @UniqueConstraint(columnNames = {"email"})
})
public class User {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private long id;
    private String name;
    private String username;
    private String email;
    private String password;

    @ManyToMany(fetch = FetchType.EAGER, cascade = CascadeType.ALL)
    @JoinTable(name = "user_roles",
        joinColumns = @JoinColumn(name = "user_id", referencedColumnName = "id"),
        inverseJoinColumns = @JoinColumn(name = "role_id", referencedColumnName =
"id"))
    private Set<Role> roles;
}
```

Role JPA Entity

```
package com.springboot.blog.entity;

import lombok.Getter;
import lombok.Setter;

import jakarta.persistence.*;

@Setter
@Getter
@Entity
@Table(name = "roles")
public class Role {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private long id;

    @Column(length = 60)
    private String name;
}
```

5. Repository LayerUserRepository

```
package com.springboot.blog.repository;
```

```
import com.springboot.blog.entity.User;
import org.springframework.data.domain.Example;
import  org.springframework.data.jpa.repository.JpaRepository;

import java.util.Optional;

public interface UserRepository extends JpaRepository<User, Long> {
    Optional<User> findByEmail(String email);
    Optional<User> findByUsernameOrEmail(String username, String email);
    Optional<User> findByUsername(String username);
    Boolean existsByUsername(String username);
    Boolean existsByEmail(String email);
}
```

RoleRepository

```
package com.springboot.blog.repository;

import com.springboot.blog.entity.Role;
import org.springframework.data.jpa.repository.JpaRepository;

import java.util.Optional;

public interface RoleRepository extends JpaRepository<Role, Long> {
    Optional<Role> findByName(String name);
}
```

6. Service Layer - CustomUserDetailsService

Let's write a logic to load user details by name or email from the database.

Let's create a *CustomUserDetailsService* which implements the *UserDetailsService* interface(Spring security in-build interface) and provides an implementation for the *loadUserByUername()* method:

```
import com.springboot.blog.entity.User;
import com.springboot.blog.repository.UserRepository;
import org.springframework.security.core.GrantedAuthority;
import org.springframework.security.core.authority.SimpleGrantedAuthority;
import org.springframework.security.core.userdetails.UserDetails;
import org.springframework.security.core.userdetails.UserDetailsService;
import org.springframework.security.core.userdetails.UsernameNotFoundException;
import org.springframework.stereotype.Service;

import java.util.Set;
```

```
import java.util.stream.Collectors;

@Service
public class CustomUserDetailsService implements UserDetailsService {

    private UserRepository userRepository;

    public CustomUserDetailsService(UserRepository userRepository) {
        this.userRepository = userRepository;
    }

    @Override
    public UserDetails loadUserByUsername(String usernameOrEmail) throws
UsernameNotFoundException {
        User user = userRepository.findByUsernameOrEmail(usernameOrEmail,
usernameOrEmail)
                .orElseThrow(() ->
                        new UsernameNotFoundException("User not found with username or email:
"+ usernameOrEmail));

        Set<GrantedAuthority> authorities = user
                .getRoles()
                .stream()
                .map((role) -> new
SimpleGrantedAuthority(role.getName())).collect(Collectors.toSet());

        return new org.springframework.security.core.userdetails.User(user.getEmail(),
                user.getPassword(),
                authorities);
    }
}
```

Spring Security uses the UserDetailsService interface, which contains the loadUserByUsername(String username) method to lookup UserDetails for agiven username.

The UserDetails interface represents an authenticated user object and Spring Securityprovides an out-of-the-box implementation of org.springframework.security.core.userdetails.User.

7. Spring Security Configuration

Let's create a class *SecurityConfig* and add the following configuration to it:

```
package com.springboot.blog.config;
```

```
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;import
org.springframework.http.HttpMethod;
import org.springframework.security.authentication.AuthenticationManager;import
org.springframework.security.config.Customizer;
import org.springframework.security.config.annotation.authentication.configuration.AuthenticationC
onfiguration;
import org.springframework.security.config.annotation.method.configuration.EnableMethodSecurity
;
import org.springframework.security.config.annotation.web.builders.HttpSecurity;import
org.springframework.security.core.userdetails.User;
import org.springframework.security.core.userdetails.UserDetails;
import org.springframework.security.core.userdetails.UserDetailsService; import
org.springframework.security.crypto.bcrypt.BCryptPasswordEncoder;import
org.springframework.security.crypto.password.PasswordEncoder;
import org.springframework.security.provisioning.InMemoryUserDetailsManager;import
org.springframework.security.web.SecurityFilterChain;

@Configuration @EnableMethodSecurity public class SecurityConfig {

private UserDetailsService userDetailsService;

public SecurityConfig(UserDetailsService userDetailsService){this.userDetailsService =
userDetailsService;
}

@Bean
public static PasswordEncoder passwordEncoder(){return new BCryptPasswordEncoder();
}

@Bean
public AuthenticationManager authenticationManager(
                AuthenticationConfiguration configuration) throws Exception {return
configuration.getAuthenticationManager();
}

@Bean
SecurityFilterChain securityFilterChain(HttpSecurity http) throws Exception {

http.csrf().disable()
.authorizeHttpRequests((authorize) ->
//authorize.anyRequest().authenticated() authorize.requestMatchers(HttpMethod.GET,
"/api/**").permitAll()
```

```
                        .requestMatchers("/api/auth/**").permitAll()
                        .anyRequest().authenticated()

                );

        return http.build();
    }
}
```

In Spring Security 5.6, we can enable annotation-based security using the *@EnableMethodSecurity* annotation on any *@Configuration* instance. *@EnableMethodSecurity* enables @PreAuthorize,@PostAuthorize, @PreFilter, and @PostFilter by default.

We are allowing anyone to access login REST API with the below security configuration:

```
authorize.requestMatchers(HttpMethod.GET, "/api/**").permitAll()
```

We are using the Spring security provided *BCryptPasswordEncoder* class to encrypt the passwords.

8. DTO or Payload Classes

Let's create DTO classes to transfer data or payload between client and server and vice-versa.LoginDto

```
package com.springboot.blog.payload;

import lombok.Data;

@Data
public class LoginDto {
    private String usernameOrEmail;
    private String password;
}
```

SignUpDto

```
package com.springboot.blog.payload;

import lombok.Data;

@Data
public class SignUpDto {
    private String name;
    private String username;
```

```
    private String email;
    private String password;
}
```

9. Controller Layer - Login/Sign-in and Register/SignUp REST APIs

Now it's time to code Login/Sign-in and Register/SignUp REST APIs. Let's create aclass *AuthController* and add the following code to it:

```
package com.springboot.blog.controller;

import  com.springboot.blog.entity.Role; import  com.springboot.blog.entity.User; import
com.springboot.blog.payload.LoginDto;
import com.springboot.blog.payload.SignUpDto; import
com.springboot.blog.repository.RoleRepository;import
com.springboot.blog.repository.UserRepository;
import org.springframework.beans.factory.annotation.Autowired;import
org.springframework.http.HttpStatus;
import org.springframework.http.ResponseEntity;
import org.springframework.security.authentication.AuthenticationManager;import
org.springframework.security.authentication.UsernamePasswordAuthenticationToken; import
org.springframework.security.core.Authentication;
import org.springframework.security.core.context.SecurityContextHolder;import
org.springframework.security.crypto.password.PasswordEncoder; import
org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.bind.annotation.RequestBody; import
org.springframework.web.bind.annotation.RequestMapping;import
org.springframework.web.bind.annotation.RestController;

import java.util.Collections;

@RestController @RequestMapping("/api/auth")public class AuthController {

@Autowired
private AuthenticationManager authenticationManager;

@Autowired
private UserRepository userRepository;

@Autowired
private RoleRepository roleRepository;

@Autowired
private PasswordEncoder passwordEncoder;
```

```
@PostMapping("/signin")
 public ResponseEntity<String> authenticateUser(@RequestBody LoginDto loginDto)
{Authentication authentication = authenticationManager.authenticate(new
 UsernamePasswordAuthenticationToken( loginDto.getUsernameOrEmail(),
loginDto.getPassword()));

SecurityContextHolder.getContext().setAuthentication(authentication);
return new ResponseEntity<>("User signed-in successfully!.", HttpStatus.OK);
}

@PostMapping("/signup")
public ResponseEntity<?> registerUser(@RequestBody SignUpDto signUpDto){

// add check for username exists in a DB
if(userRepository.existsByUsername(signUpDto.getUsername())){
        return new ResponseEntity<>("Username is already taken!",HttpStatus.BAD_REQUEST);
}

// add check for email exists in DB if(userRepository.existsByEmail(signUpDto.getEmail())){
        return new ResponseEntity<>("Email is already taken!",HttpStatus.BAD_REQUEST);
}

// create user object User user = new User();
user.setName(signUpDto.getName()); user.setUsername(signUpDto.getUsername());
user.setEmail(signUpDto.getEmail());
user.setPassword(passwordEncoder.encode(signUpDto.getPassword()));

Role roles = roleRepository.findByName("ROLE_ADMIN").get();
user.setRoles(Collections.singleton(roles));

userRepository.save(user);

return new ResponseEntity<>("User registered successfully", HttpStatus.OK);

}
}
```

Login/Sign in REST API:

```
    @PostMapping("/signin")
    public ResponseEntity<String> authenticateUser(@RequestBody LoginDto loginDto){
```

```
        Authentication authentication = authenticationManager.authenticate(new
UsernamePasswordAuthenticationToken(
                loginDto.getUsernameOrEmail(), loginDto.getPassword()));

        SecurityContextHolder.getContext().setAuthentication(authentication);
        return new ResponseEntity<>("User signed-in successfully!.", HttpStatus.OK);
    }
```

Register/SignUp REST API:

```
    @PostMapping("/signup")
    public ResponseEntity<?> registerUser(@RequestBody SignUpDto signUpDto){

        // add check for username exists in a DB
        if(userRepository.existsByUsername(signUpDto.getUsername())){
            return new ResponseEntity<>("Username is already taken!",
HttpStatus.BAD_REQUEST);
        }

        // add check for email exists in DB
        if(userRepository.existsByEmail(signUpDto.getEmail())){
            return new ResponseEntity<>("Email is already taken!",
HttpStatus.BAD_REQUEST);
        }

        // create user object
        User user = new User();
        user.setName(signUpDto.getName());
        user.setUsername(signUpDto.getUsername());
        user.setEmail(signUpDto.getEmail());
        user.setPassword(passwordEncoder.encode(signUpDto.getPassword()));

        Role roles = roleRepository.findByName("ROLE_ADMIN").get();
        user.setRoles(Collections.singleton(roles));

        userRepository.save(user);

        return new ResponseEntity<>("User registered successfully", HttpStatus.OK);

    }
```

10. Run Spring Boot Application

We have successfully developed Login and Registration Rest APIs.

Now it's time to deploy our application in a servlet container(embedded tomcat).Two ways we can start the standalone Spring boot application.

1. From the root directory of the application and type the following command to run it -

```
$ mvn spring-boot:run
```

2. From your IDE, run the Application.main() method as a standalone Java class that will start the embedded Tomcat server on port 8080 and point the browserto **http://localhost:8080/**.

Important

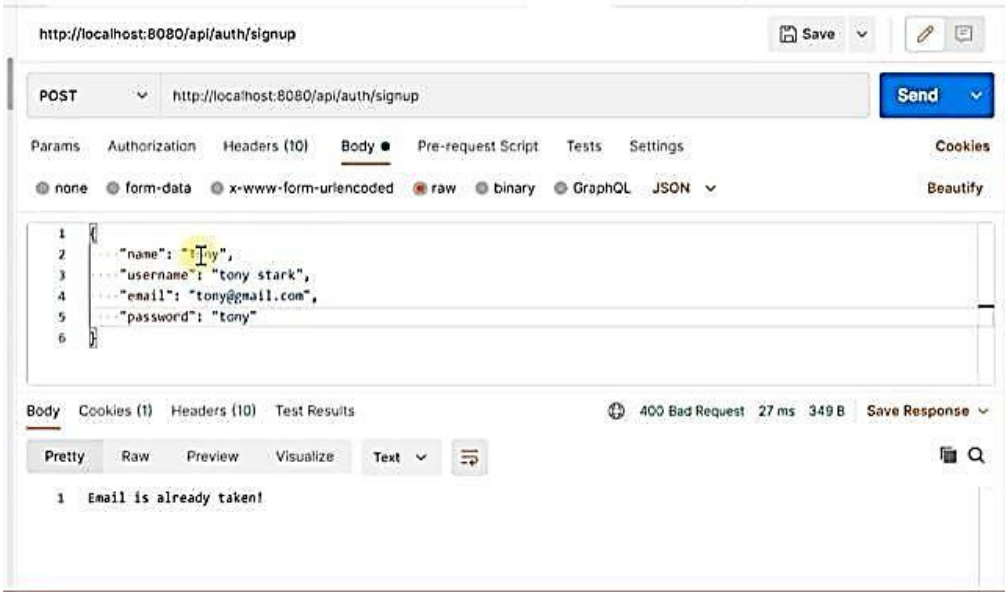
Once you start the Spring boot application, make sure that you add therole **ROLE_ADMIN** record in the *roles* table.

Execute below INSERT SQL statements to add **ROLE_ADMIN** role to the *roles* databasetable:

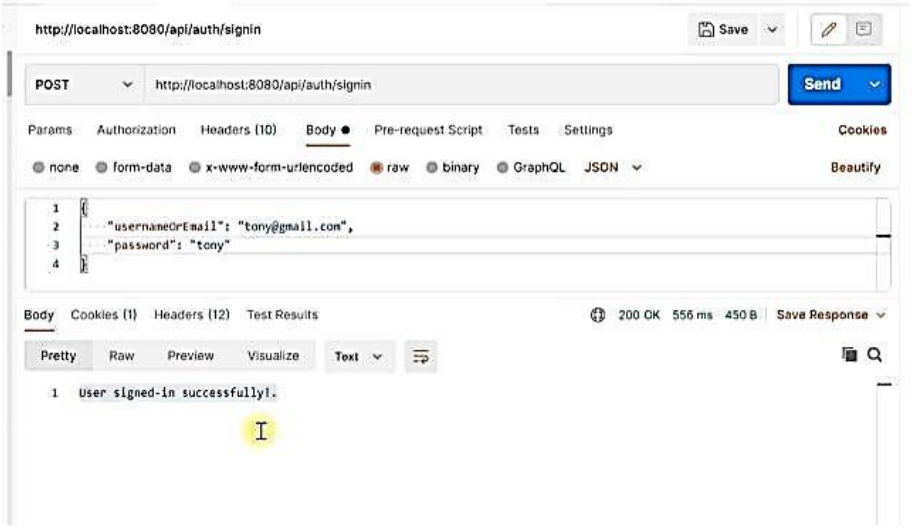
```
INSERT INTO roles VALUES('ROLE_ADMIN');
```

11. Test using Postman

Refer to the below screenshots to test Login and Registration REST API using Postman:SignUp REST API:



SignIn/Login REST API:



Experiment: 18

Aim: Build User Authentication Flow and Authorization using Spring Security.

Theory: Authentication and authorization are **two vital information security processes that administrators use to protect systems and information**. Authentication verifies the identityof a user or service, and authorization determines their access rights.

Procedure:

Step 1: Open **pom.xml** and add the **spring-boot-starter-security**. It automatically configures the basic security for us.

```
<dependency>
<groupId>org.springframework.boot</groupId>
<artifactId>spring-boot-starter-security</artifactId>
</dependency>
```

Step 2: Restart the server, we get a **password** in the log. Each time the server starts up the password will be different.

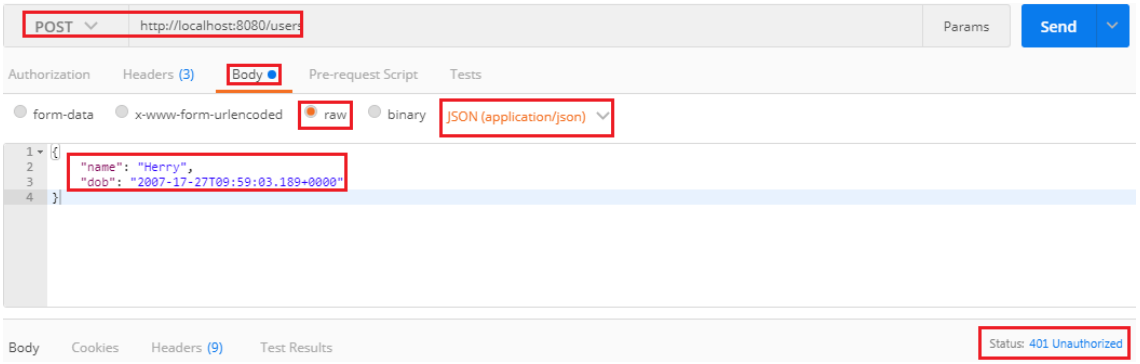
```
2019-10-16 10:36:48.589  INFO 5548 --- [  restartedMain] .s.s.UserDetailsServi

Using generated security password: ed69a2ac-2aed-4dc4-87bf-43f41f598285

2019-10-16 10:36:48.972  INFO 5548 --- [  restartedMain] o.s.s.web.DefaultSecu
```

Step 3: Copy the password from the log.

Step 4: Open the REST Client **Postman** and send a POST request. We are sending a POST to create a user.

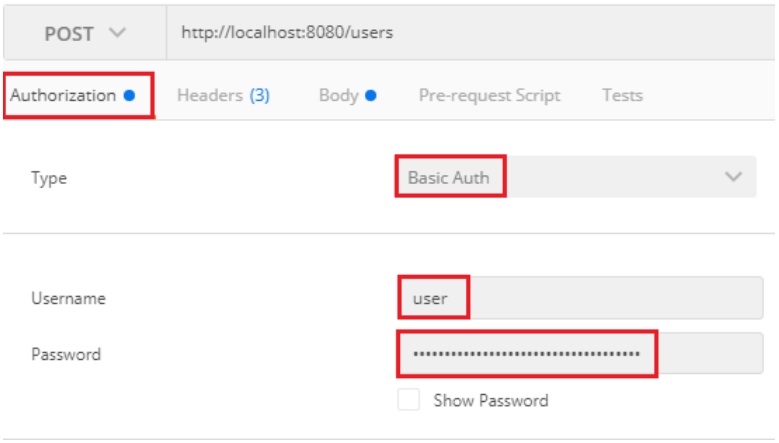
- Provide URI http://localhost:8080/users.
- Click on the Body tab and select the raw radio button.
- Select the media type JSON (application/json).
- Provide name and dob.
- Click on the Send button.

It returns the **Status: 401 Unauthorized**.



Step 5: In the REST client **Postman**, click on the **Authorization** tab and do the following:

- Select the type of authentication Basic Auth.
- Provide the Username. The default username is user.
- Paste the password, which we have copied from the log.
- Click on the Send button.

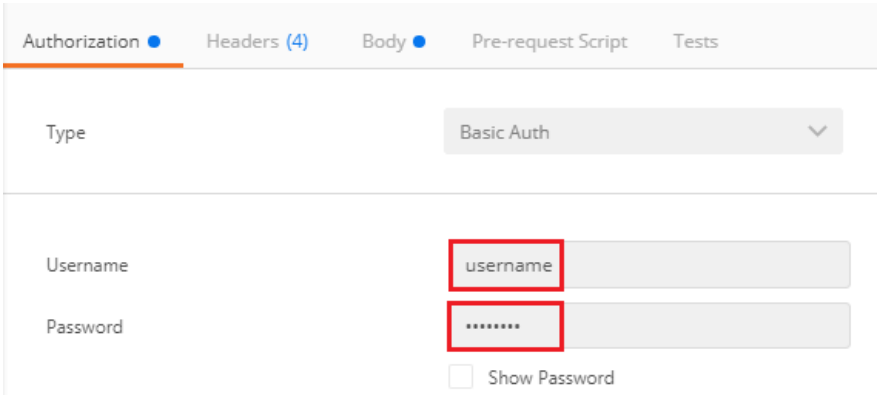


It shows the **Status: 201 Created**. There is a disadvantage that when we restart the server, the password changes again and again. The solution to this problem is that configure the username and password in the application.properties file.

application.properties

1. spring.security.user.name=user
2. spring.security.user.password=password

Now, move to Postman and try to send a POST request that returns **Status: 401 Unauthorized**. It is because we are still using an old password. So we are required to change the username and password with the new one. Provide the username and password which we have configured in the properties file. We get the Status: 201 Created.


Authorization
Headers (4)
Body
Pre-request Script
Tests
Type
Basic Auth
Username
username
Password
Show Password

Experiment: 19

Aim: Create MongoDB and demonstrate CRUD operations on document.

Theory: MongoDB is an open-source document-oriented database that is designed to store a large scale of data and also allows you to work with that data very efficiently. It is categorized under the NoSQL (Not only SQL) database because the storage and retrieval ofdata in the MongoDB are not in the form of tables.

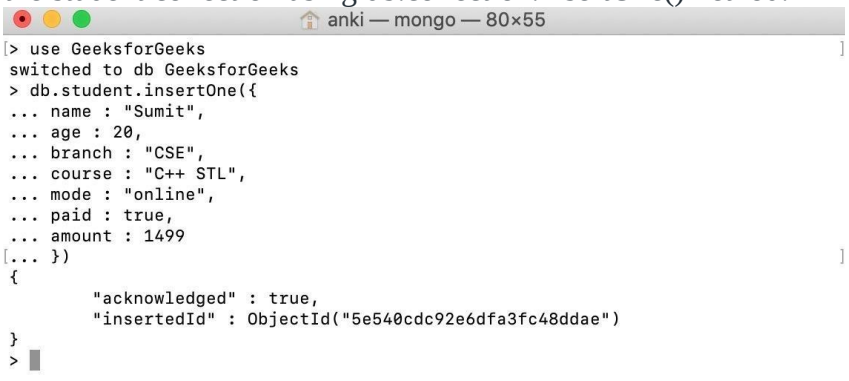
The MongoDB database is developed and managed by MongoDB.Inc under SSPL(Server Side Public License) and initially released in February 2009. It also provides official driversupport for all the popular languages like C, C++, C#, and .Net, Go, Java, Node.js, Perl, PHP, Python, Motor, Ruby, Scala, Swift, Mongoid. So, that you can create an application using any of these languages. Nowadays there are so many companies that used MongoDBlike Facebook, Nokia, eBay, Adobe, Google, etc. to store their large amount of data.

Procedure:

Create Operations –
The create or insert operations are used to insert or add new documents inthe collection. If a collection does not exist, then it will create a new collection in the database. You can perform, create operations using the following methods provided by the MongoDB:

Method	Description
db.collection.insertOne()	It is used to insert a single document in the collection.
db.collection.insertMany()	It is used to insert multiple documents in the collection.
db.createCollection()	It is used to create an empty collection.

Example 1: In this example, we are inserting details of a single student inthe form of document in the student collection using db.collection.insertOne()method.



Example 2: In this example, we are inserting details of the multiple studentsin the form of documents in the student collection using

db.collection.insertMany()method.

```
anki — mongo — 80×55
> use GeeksforGeeks
switched to db GeeksforGeeks
> db.student.insertMany([
... {
... name : "Sumit",
... age : 20,
... branch : "CSE",
... course : "C++ STL",
... mode : "online",
... paid : true,
... amount : 1499
... },
...
... {
... name : "Rohit",
... age : 21,
... branch : "CSE",
... course : "C++ STL",
... mode : "online",
... paid : true,
... amount : 1499
... }
...
... ])
{
        "acknowledged" : true,
        "insertedIds" : [
                ObjectId("5e540d3192e6dfa3fc48ddaf"),
                ObjectId("5e540d3192e6dfa3fc48ddb0")
        ]
}
>
```

Read Operations –

The Read operations are used to retrieve documents from the collection, orin other words, read operations are used to query a collection for a document. You can perform read operation using the following method provided by the MongoDB:

Method	Description
db.collection.find()	It is used to retrieve documents from the collection.

Example : In this example, we are retrieving the details of students from thestudent collection usingdb.collection.find()method.

```
anki — mongo — 80×55
> use GeeksforGeeks
switched to db GeeksforGeeks
> db.student.find().pretty()
{
        "_id" : ObjectId("5e540cdc92e6dfa3fc48ddae"),
        "name" : "Sumit",
        "age" : 20,
        "branch" : "CSE",
        "course" : "C++ STL",
        "mode" : "online",
        "paid" : true,
        "amount" : 1499
}
{
        "_id" : ObjectId("5e540d3192e6dfa3fc48ddaf"),
        "name" : "Sumit",
        "age" : 20,
        "branch" : "CSE",
        "course" : "C++ STL",
        "mode" : "online",
        "paid" : true,
        "amount" : 1499
}
{
        "_id" : ObjectId("5e540d3192e6dfa3fc48ddb0"),
        "name" : "Rohit",
        "age" : 21,
        "branch" : "CSE",
        "course" : "C++ STL",
        "mode" : "online",
        "paid" : true,
        "amount" : 1499
}
>
```

Update Operations –

The update operations are used to update or modify the existing document inthe collection. You can perform update operations using the following methods provided by the MongoDB:

Method	Description
db.collection.updateOne()	It is used to update a single document in the collection that satisfy the given criteria.
db.collection.updateMany()	It is used to update multiple documents in the collection that satisfy the given criteria.
db.collection.replaceOne()	It is used to replace single document in the collectionthat satisfy the given criteria.

Example 1: In this example, we are updating the age of Sumit in the studentcollection using db.collection.updateOne()method.

Example 2: In this example, we are updating the year of course in all thedocuments in the student collection using

db.collection.updateMany()method.

Delete Operations –

The delete operation are used to delete or remove the documents from a collection. You can perform delete operations using the following methodsprovided by the MongoDB:

Method	Description
db.collection.deleteOne()	It is used to delete a single document from the collection that satisfy the given criteria.
db.collection.deleteMany()	It is used to delete multiple documents from thecollection that satisfy the given criteria.

Example 1: In this example, we are deleting a document from the studentcollection using db.collection.deleteOne() method.

```
> use GeeksforGeeks
switched to db GeeksforGeeks
> db.student.find().pretty()
{
        "_id" : ObjectId("5e540cdc92e6dfa3fc48ddae"),
        "name" : "Sumit",
        "age" : 24,
        "branch" : "CSE",
        "course" : "C++ STL",
        "mode" : "online",
        "paid" : true,
        "amount" : 1499,
        "year" : 2020
}
{
        "_id" : ObjectId("5e540d3192e6dfa3fc48ddaf"),
        "name" : "Sumit",
        "age" : 20,
        "branch" : "CSE",
        "course" : "C++ STL",
        "mode" : "online",
        "paid" : true,
        "amount" : 1499,
        "year" : 2020
}
{
        "_id" : ObjectId("5e54103592e6dfa3fc48ddb1"),
        "name" : "Rohit",
        "age" : 21,
        "branch" : "CSE",
        "course" : "C++ STL",
        "mode" : "online",
        "paid" : true,
        "amount" : 1499
}
> db.student.deleteOne({name: "Sumit"})
{ "acknowledged" : true, "deletedCount" : 1 }
> db.student.find().pretty()
{
        "_id" : ObjectId("5e540d3192e6dfa3fc48ddaf"),
        "name" : "Sumit",
        "age" : 20,
        "branch" : "CSE",
        "course" : "C++ STL",
        "mode" : "online",
        "paid" : true,
        "amount" : 1499,
        "year" : 2020
}
{
        "_id" : ObjectId("5e54103592e6dfa3fc48ddb1"),
        "name" : "Rohit",
        "age" : 21,
        "branch" : "CSE",
        "course" : "C++ STL",
        "mode" : "online",
        "paid" : true,
        "amount" : 1499
}
```

Example 2: In this example, we are deleting all the documents from thestudent collection using db.collection.deleteMany()
method.

Experiment: 20

Aim: Create and manage users and roles (MongoDB).

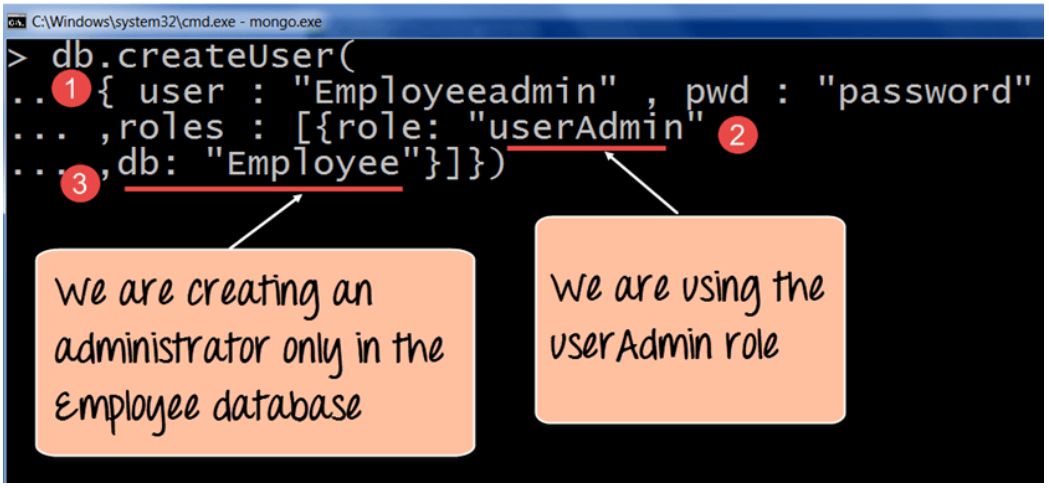
Theory: MongoDB allows us to create a new user within the system in a very efficient way. If the user which we are going to insert already exists, then it will return an error in response. If there is no user that already exists then insert that record within the system.

We can create an independent role within MongoDB by expressing permissions on all privileges explicitly. We can also inherit privileges from a different role in the new roles.

Procedure:

To create a user who will manage a single database, we can use the same command asmentioned above but we need to use the “userAdmin” option only.

The following example shows how this can be done;



db.createUser(
{
user: "Employeeadmin",pwd: "password",

roles:[{role: "userAdmin" , db:"Employee"}]})

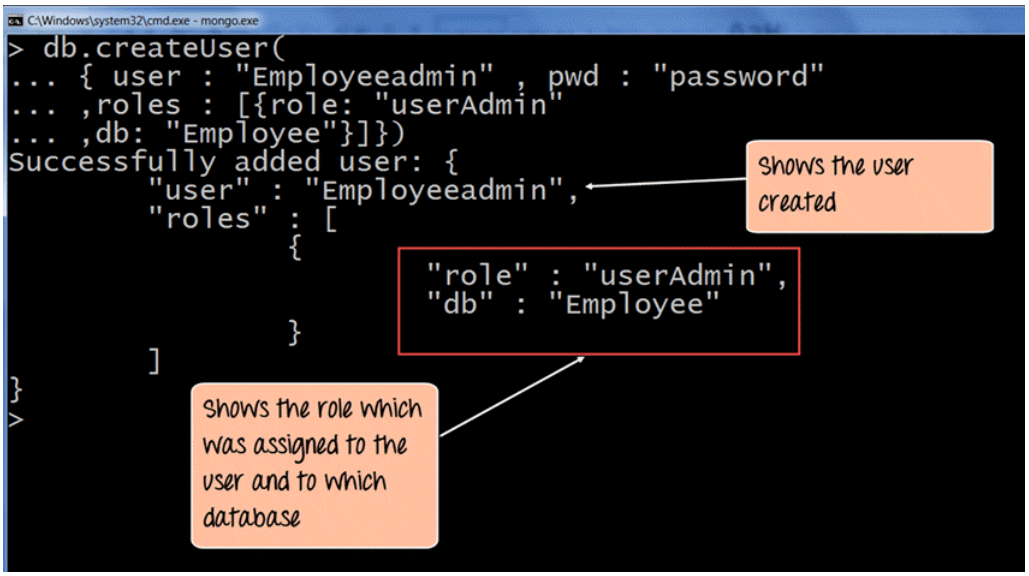
Code Explanation:

1. The first step is to specify the “username” and “password” which needs to be created.
2. The second step is to assign a role for the user which in this case since it needs to be adatabase administrator is assigned to the “userAdmin” role. This role allows the user to have administrative privileges only to the database specified in the db option.

3. The db parameter specifies the database to which the user should have administrativeprivileges on.

If the command is executed successfully, the following Output will be shown:

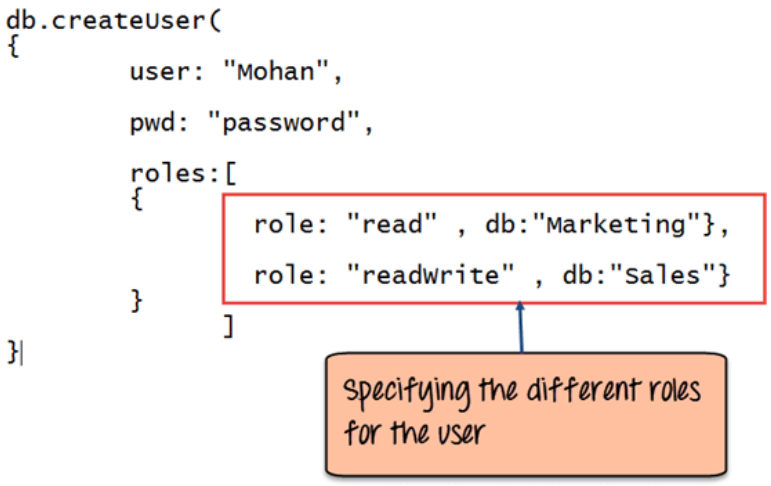
Output:



The output shows that a user called “Employeeadmin” was created and that user hasprivileges only on the “Employee” database.

Managing users

First understand the roles which you need to define. There is a whole list of role available in MongoDB. For example, there is a the “read role” which only allows read only access to databases and then there is the “readwrite” role which provides read and write access to the database , which means that the user can issue the insert, delete and update commands on collections in that database.



```
db.createUser(
{
user: "Mohan", pwd: "password",

roles:[
{
                        role: "read" , db:"Marketing"},role: "readWrite" , db:"Sales"}
    {
                        ]
}

})
```

The above code snippet shows that a user called Mohan is created, and he is assigned multiple roles in multiple databases. In the above example, he is given read only permissionto the “Marketing” database and readWrite permission to the “Sales” database.

Experiment: 21

Aim: Demonstrate:

- ACID Transactions in MongoDB.
- Perform CRUD operations on MongoDB through API using Spring Starter.
- How to run mongoDB on Cloud.

Theory: ACID Transactions in MongoDB:

Single-document updates have always been atomic in MongoDB. The Document model is well suited to storing related data that is accessed in a single Document(compared to the relational model where related data may be normalised and split between multiple tables). A well-designed Document schema, in most cases, allows you to work without the need for multi-Document transactions.

When multi-document ACID compliance is required, MongoDB transactions work across your cluster and perform as expected. Transactions have a performance overhead in a distributed system, so you need to keep your resource constraints and performance goals inmind.

Having understood what ACID transactions are, let’s now look at how to implement them MongoDB ACID Transactions. The following article will show how to implement MongoDBACID transactions using Mongo Shell.

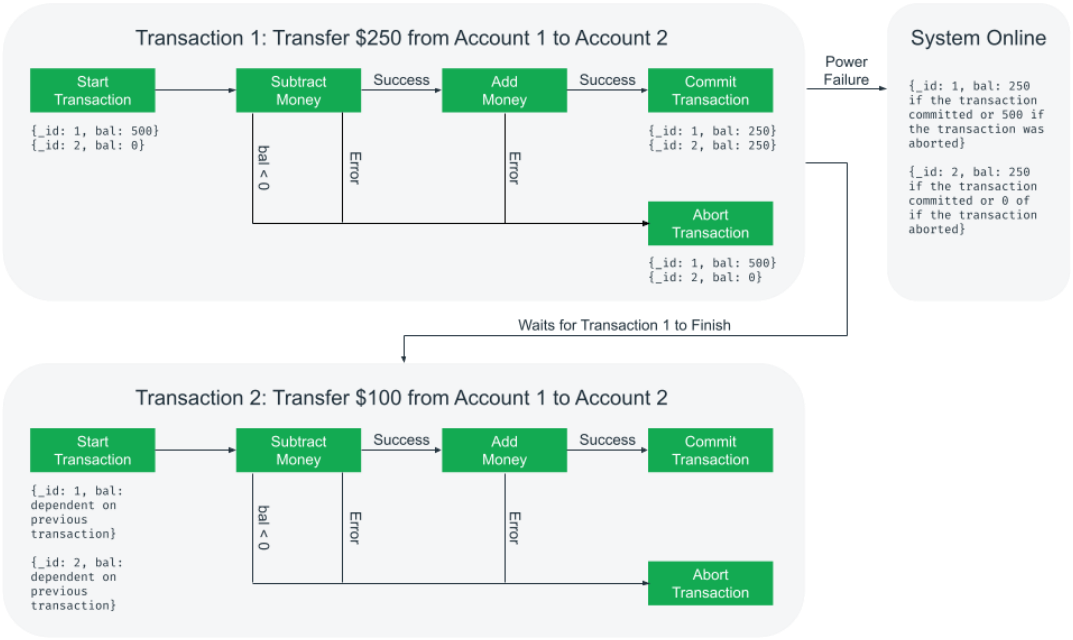
Procedure:

- **Demonstration of ACID Transactions in MongoDB:**

Let's continue with the banking example we discussed earlier in the article where money is being transferred from one account to another. Let's examine each of the ACID properties inthis example:

- Atomicity: Money needs to both be removed from one account and added to the other, or the transaction will be aborted. Removing money from one account without adding it to another would leave the data in an inconsistent state.
- Consistency: Consider a database constraint that an account balance cannot drop below zero dollars. All updates to an account balance inside of a transaction mustleave the account with a valid, non-negative balance, or the transaction should beaborted.
- Isolation: Consider two concurrent requests to transfer money from the same bank account. The final result of running the transfer requests concurrently should be thesame as running the transfer requests sequentially.

- Durability: Consider a power failure immediately after a database has confirmed thatmoney has been transferred from one bank account to another. The database should still hold the updated information even though there was an unexpected failure.



The diagram demonstrates how the ACID properties impact the flow of transferring moneyfrom one bank account to another.

MongoDB added support for multi-document ACID transactions in version 4.0 in 2018 andextended that support for distributed multi-document ACID transactions in version 4.2 in 2019.

MongoDB's document model allows related data to be stored together in a single document.The document model, combined with atomic document updates, obviates the need for transactions in a majority of use cases. Nonetheless, there are cases where true multi- document, multi-collection MongoDB transactions are the best choice.

MongoDB transactions work similarly to transactions in other databases. To use a transaction, start a MongoDB session through a driver. Then, use that session to execute yourgroup of database operations. You can run any of the CRUD (create, read, update, and delete)operations across multiple documents, multiple collections, and multiple shards.

- **Perform CRUD operations on MongoDB through API using Spring Starter.**

Step 1: Create a Spring Boot project.

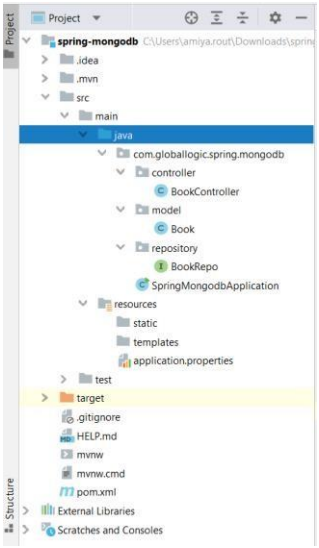
Step 2: Add the following dependency

- Spring Web

- MongoDB
- Lombok
- DevTools

Step 3: Create 3 packages and create some classes and interfaces inside these packages asseen in the below image

- model
- repository
- controller



***Note**:*

- *Green Rounded Icon 'I' Buttons are Interface*
- *Blue Rounded Icon 'C' Buttons are Classes*

Step 4: Inside the entity package
Creating a simple POJO class inside the Book.java file.
Example

- Java

```
// Java Program to illustrate Book File

// Importing required classes
import lombok.AllArgsConstructor;import lombok.Data;


import lombok.NoArgsConstructor;
import org.springframework.data.annotation.Id;
import org.springframework.data.mongodb.core.mapping.Document;


// Annotations@Data


@NoArgsConstructor @AllArgsConstructor @Document(collection = "Book")
```

```
// Class

public class Book {


// Attributes

@Id private int  id; private String bookName;


private String authorName;

}
```

Step 5: Inside the repository package
Create a simple interface and name the interface as **BookRepo**. This interface is going toextend the **MongoRepository** as we have discussed above.
Example

- Java

```
// Java Program to Illustrate BookRepo File


import com.globallogic.spring.mongodb.model.Book;

import org.springframework.data.mongodb.repository.MongoRepository;


public interface BookRepo

extends MongoRepository<Book, Integer> {

}
```

Step 6: Inside the controller package
Inside the package create one class named as **BookController**.

```
import com.globallogic.spring.mongodb.model.Book;

import com.globallogic.spring.mongodb.repository.BookRepo; import

org.springframework.beans.factory.annotation.Autowired;import

org.springframework.web.bind.annotation.*;

import java.util.List;
```

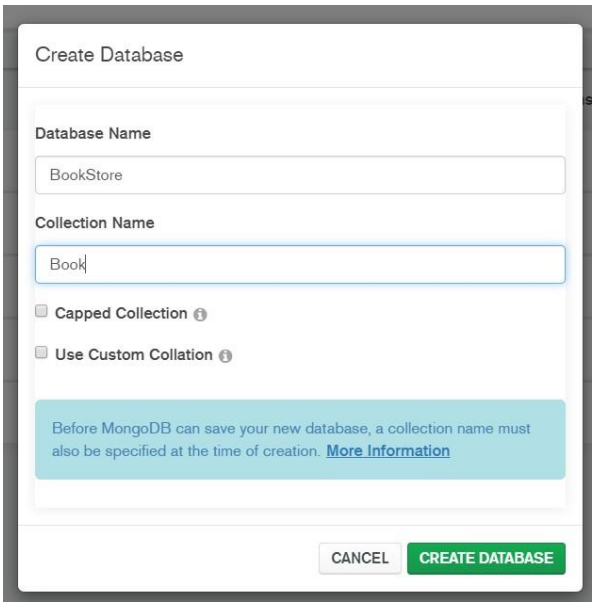
```
// Annotation

@RestController

// Class

public class BookController {

@Autowired

private BookRepo repo;

@PostMapping("/addBook")

public String saveBook(@RequestBody Book book){repo.save(book);

return "Added Successfully";
}

@GetMapping("/findAllBooks")public List<Book> getBooks() {

return repo.findAll();

}

@DeleteMapping("/delete/{id}")

public String deleteBook(@PathVariable int id){repo.deleteById(id);

return "Deleted Successfully";

}

}
```

Step 7: Below is the code for the application.properties fileserver.port = 8989

MongoDB Configuration server.port:8989 spring.data.mongodb.host=localhost

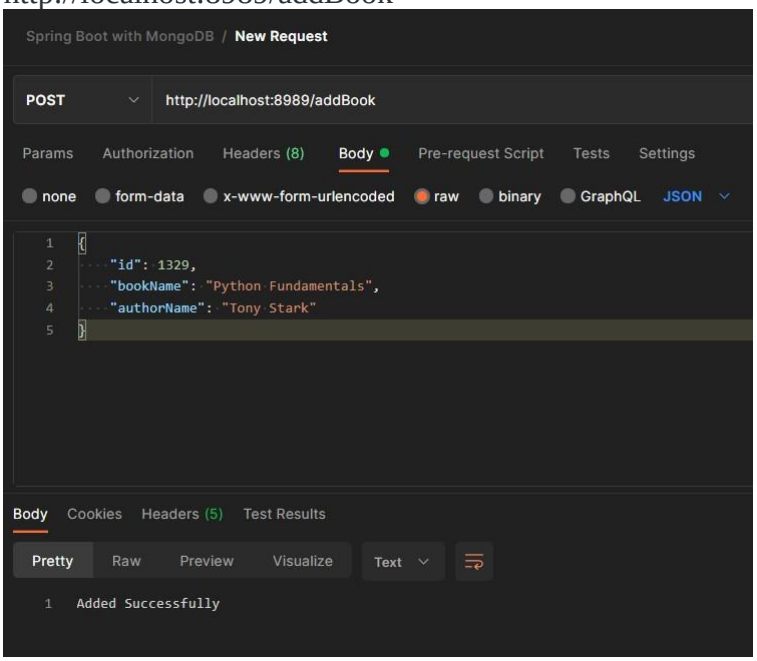
spring.data.mongodb.port=27017 spring.data.mongodb.database=BookStore

Step 8: Inside the MongoDB Compass
Go to your MongoDB Compass and create a Database named **BookStore** and inside thedatabase create a collection named **Book** as seen in the below image

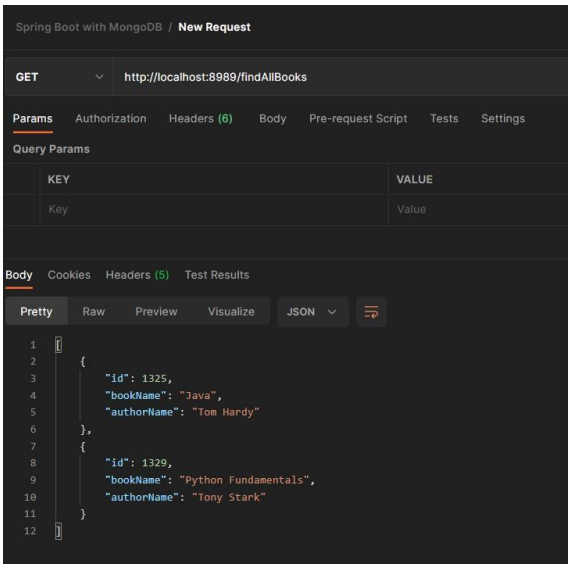


Now run your application and let's test the endpoints in Postman and also refer to ourMongoDB Compass.

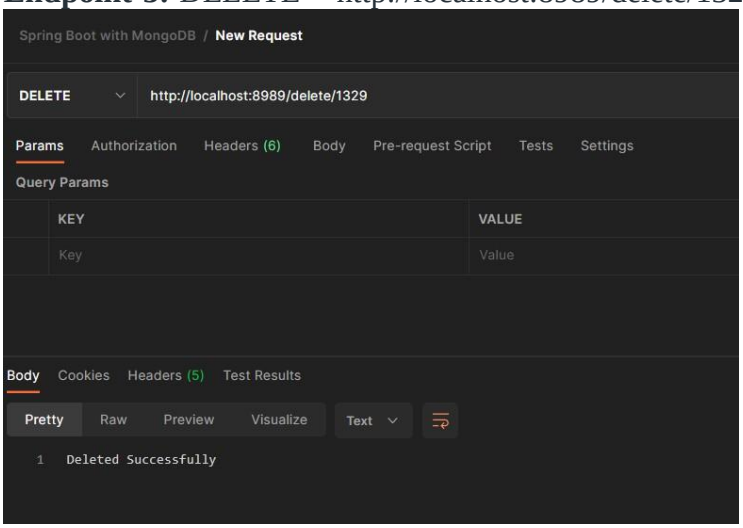
Testing the Endpoint in PostmanEndpoint 1: POST – http://localhost:8989/addBook



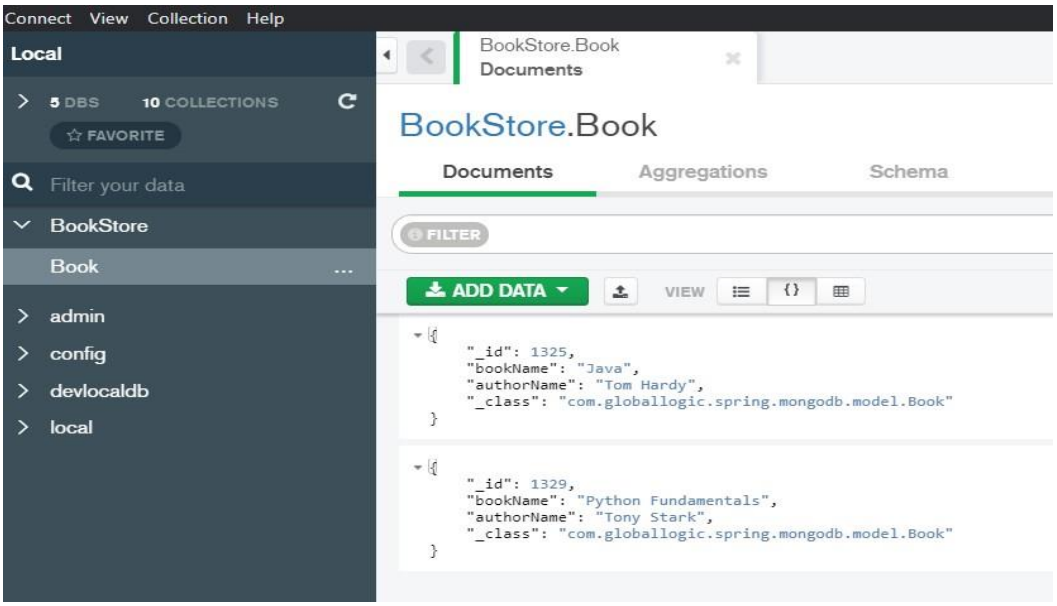
Endpoint 2: GET – http://localhost:8989/findAllBooks



Endpoint 3: DELETE – http://localhost:8989/delete/1329

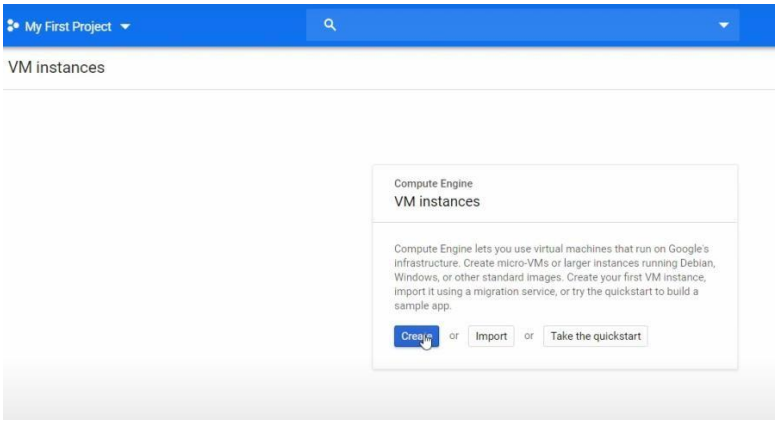


Finally, MongoDB Compass is as depicted in the below image as shown below as follows:

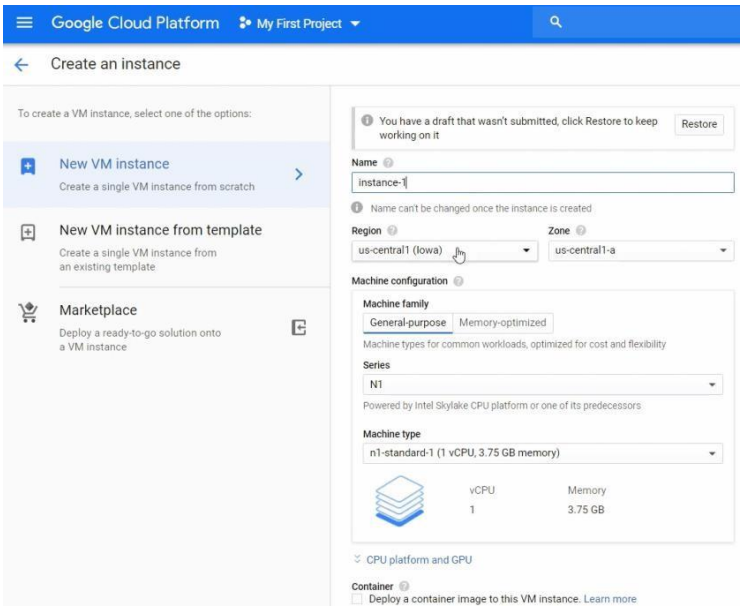


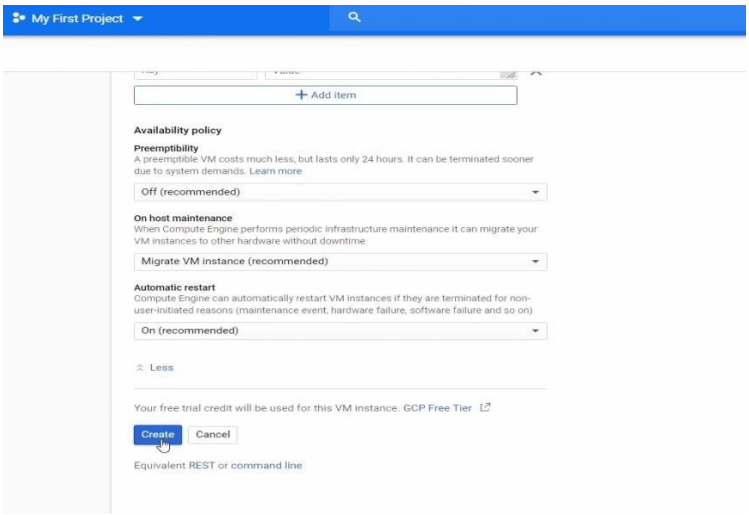
- **How to run mongoDB on cloud:**

Step 1: Go to Compute Engine in GCP and click Create Instance.



Step 2: Fill in your instance type details and click Create. This may take a fewseconds – if so, don't worry.





Step 3: MongoDB listens on port 27017. You have to make sure that this port is allowed in your firewall rules for the specific IP addresses you want to grant accessto. Be careful, don't open this port to the world. One way to restrict access is to create a new firewall rule in your VPC like this: **Source IP address ranges are allowed server IP addresses that allow access, protocols, and ports.** At this point, you have set up your instance. The next step is the installation of MongoDB on the google cloud Platform.

Installation of MongoDB in GCP

Step 1: Firstly, you need SSH in the Ubuntu instance you have created. You can dothis in many ways. One way is to use SSH from the browser option that GCP provides. **From the command line, run the command below to import the GPG MongoDB public key:**

wget -qO – https://www.mongodb.org/static/pgp/server-4.0.asc | sudo apt-key add –

Step 2: Create a file using this command:

echo "deb [arch=amd64] https://repo.mongodb.org/apt/ubuntu bionic/mongodb-org/4.2 multiverse" |

sudo tee /etc/apt/sources.list.d/mongodb-org-4.2.list

Finally, the installation of MongoDB begins using this command:

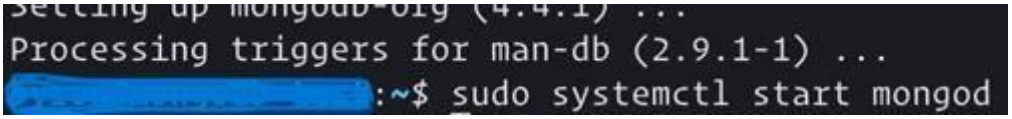
sudo apt-get update

Step 3: Installation of MongoDB:

sudo apt-get install -y mongodb-org

And then, you should have MongoDB installed. At this point, if MongoDB isinstalled but not running. Start MongoDB with this command:

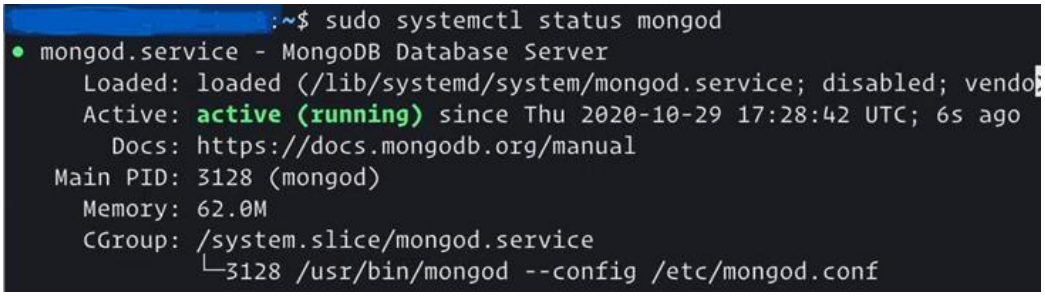
sudo systemctl start mongod



Now you'll want to take a moment to verify that MongoDB has been installedproperly. To do this, You have to run this command:

sudo systemctl status mongod

If everything is well, you will have a screen like this:



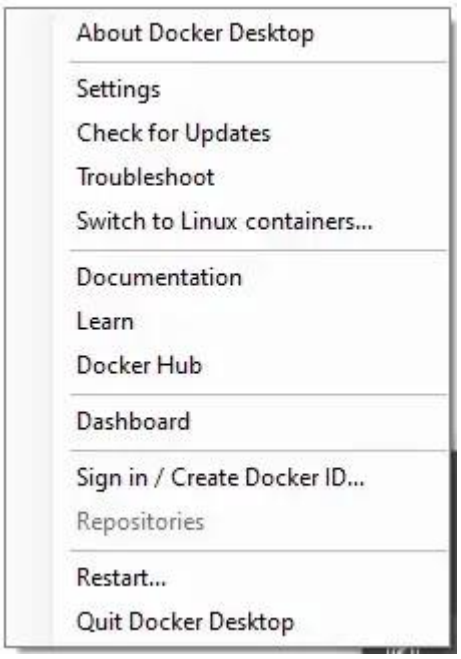
Experiment: 22

Aim: Create Docker Container using Docker image and run the Docker Container.

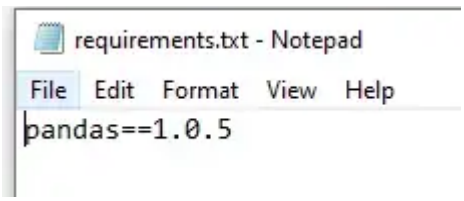
Theory: Docker is an open source platform that enables developers to build, deploy, run, update and manage *containers*—standardized, executable components that combine application source code with the operating system (OS) libraries and dependencies required to run that code in any environment.

Procedure:

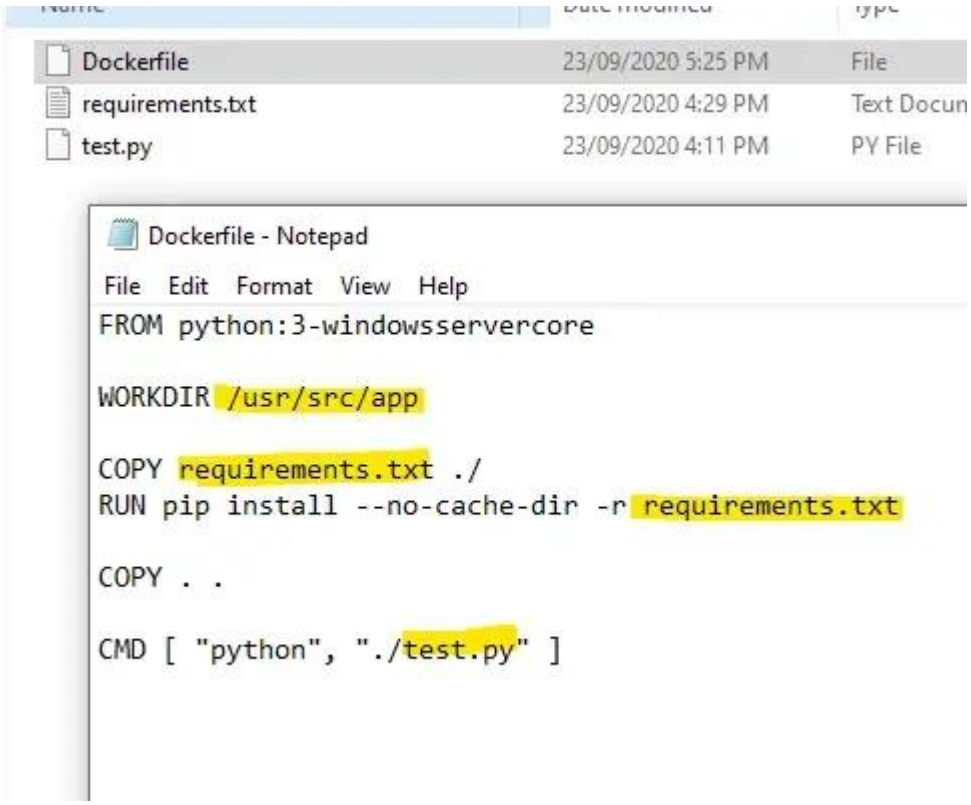
1. Download and Install Docker Desktop from Docker Website: https://docs.docker.com/get-docker/
2. Get ready your python script, libraries name (dependencies)
3. Right click the docker icon at your taskbar, and make sure its switched to Windows. If it has been switch to windows, the option will not appeared anymore and you will be able to see 'Switch to Linux containers' instead.



4.Create a requirements.txt, with required libraries and version



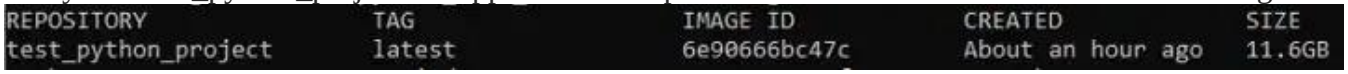
5.Create a docker file with name *Dockerfile* with the file hierachy you wish to have in the docker container. Complete the *Dockerfile* with your python file name.



6. Put all of them into the same folder

Set up

1. Start the docker from your windows. **Note: Make sure you have minimum > 12 gb of space in your machine just for the image alone.
2. Open command prompt, change the directory to the folder where the code and files reside. Examplecd "C:\Users\Folder_Name"
3. Key in the docker build -t test_python_project . into the command prompt. Note: test_python_project is the name i have given the the image. End the code with fullstop or it will result in error.
4. Make a cup of coffee, sit at the TV, and wait for the image to be created. I didn't keep track of the time, maybe it will take 30 minutes (depending on your network and machine specs).
5. Check if your docker images has successfully created by typing docker images into the cmd. My file is small because I only have Pandas library included. Once you see your image being created successfully, in my case test_python_project has appeared. I will proceed to create a container based on this image.



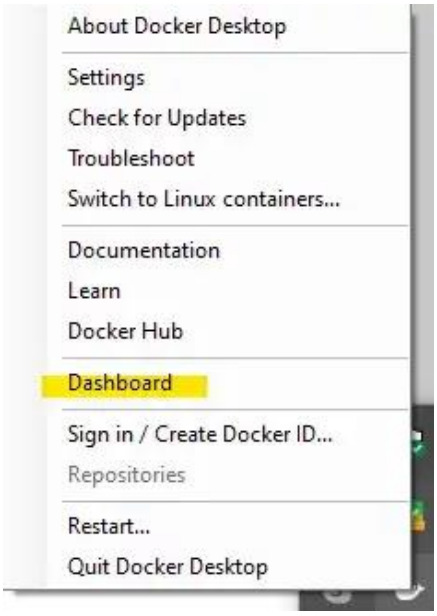
6. key into the cmd

docker run — publish 8000:8080 — detach — name test_container test_python_project

and again you can walk your dog if you are impatient and again — i didnt keep track how long it took to create the container.

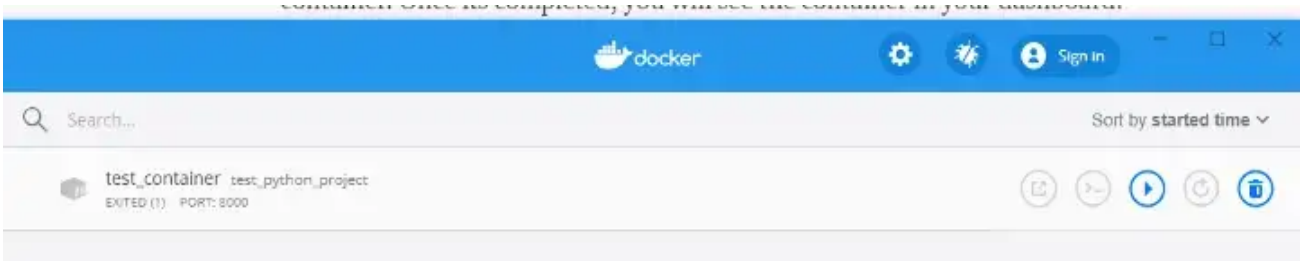
7. Once its completed, you will see the container in your dashboard. Right click on your docker icon again, click dashboard. In the script, i named my container *test_container* and it is based on the image we have previous created, i.e. *test_python_project.* To list all the container you in the cmd, just type

docker container ls -all



8. Voila, we are done. Hover to the container and you will be able to start the container with the UI. Else, if you prefer the scripting way — key in the code into cmd.

docker start test_pyton_project.



Experiment: 23

Aim: Create a cluster to deploy an app by using Kubernets tool.

Theory: A Kubernetes cluster is **a set of nodes that run containerized applications**. Containerizing applications packages an app with its dependences and some necessary services. They are more lightweight and flexible than virtual machines.

Procedure:

Step 1 - Get each server ready to run Kubernetes

We will start with creating three Ubuntu 16.04 servers. This will give you three servers to configure. To get this three member cluster up and running, you will need to select Ubuntu16.04, 4GM RAM servers and enable Private Networking.

Create 3 hosts and call them kube-01, kube-02 and kube-03. You need to be running hostswith a minimum of 4GB RAM for the Weave Socks Shop Demo.

Set your hostnames for your servers as follows:

Server	Hostname
1	kube-01
2	kube-02
3	kube-03

Kubernetes will need to assign specialized roles to each server. We will setup one server toact as the master:

Hostname	Role
kube-01	Master

Hostname	Role
kube-02	Node
kube-03	Node

Step 2 - Set up each server in the cluster to run Kubernetes.

SSH to each of the servers you created. Proceed with executing the following commands asroot. You may become the root user by executing sudo -i after SSH-ing to each host.

On each of the three Ubuntu 16.04 servers run the following commands as root:

```
1apt-get update && apt-get install -y apt-transport-https
2curl -s https://packages.cloud.google.com/apt/doc/apt-key.gpg | apt-key add -
3cat <<EOF >/etc/apt/sources.list.d/kubernetes.list
4deb http://apt.kubernetes.io/ kubernetes-xenial main
5EOF
6apt-get update
7apt-get install -y kubelet=1.15.4-00 kubeadm=1.15.4-00 kubectl=1.15.4-00 docker.io
```

Step 3 - Setup the Kubernetes Master
On the kube-01 node run the following command:

```
1kubeadm init
```

This can take a minute or two to run, the result will look like this:

To start using your cluster, you need to run the following as a regular user:

```
1mkdir -p $HOME/.kube
2sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config
3sudo chown $(id -u):$(id -g) $HOME/.kube/config
```

Your Kubernetes master has initialized successfully!Run the following commands on kube-01:

```
1mkdir -p $HOME/.kube
2sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config
3sudo chown $(id -u):$(id -g) $HOME/.kube/config
```

Step 4 - Join your nodes to your Kubernetes cluster
You can now join any number of machines by running the kubeadm join command on eachnode as root. This command will be created for you as displayed in your terminal for you tocopy and run.
An example of what this looks like is below:

```
1kubeadm join --token 702ff6.bc7aacff7aacab17 174.138.15.158:6443 --discovery-token-ca-
cert-hash
sha256:68bc22d2c631800fd358a6d7e3998e598deb2980ee613b3c2f1da8978960c8ab
```

When you join your kube-02 and kube-01 nodes you will see the following on the node:

```
1This node has joined the cluster:
2* Certificate signing request was sent to master and a response was received.
3* The Kubelet was informed of the new secure connection details.
```

To check that all nodes are now joined to the master run the following command on theKubernetes master kube-01:

```
1kubectl get nodes
```

The successful result will look like this:

```
1N        STATUS          ES          AGE   VERSION
A                 R       8m  v1.9.3
M         OL
E         Ready
2k                maste
ube       r
-01
3k        R       <n      6       v1.9.3
ube       e       on      m
-02       a       e>
          d
          y
4k        R       <n      6       v1.9.3
ube       e       on      m
-03       a       e>
          d
          y
```

You will notice that the nodes do not have a role set on join, there is an open PR to resolvethis.

Step 5 - Setup a Kubernetes Add-On For Networking Features And Policy

Kubernetes Add-Ons are pods and services that implement cluster features. Pods extend the functionality of Kubernetes. You can install addons for a range of cluster features including **Networking and Visualization**.

We are going to install the Weave Net Add-On on the kube-01 master which provides networking and network policy, will carry on working on both sides of a network partition,and does not require an external database. Read more about the Weave Net Add-on in the Weave Works Docs.

Next you will deploy a pod network to the cluster.

The options are listed at: https://kubernetes.io/docs/concepts/cluster-administration/addons/

Installing the Weave Net Add-On

Get the Weave Net yaml:

```
1curl -o weave.yaml https://cloud.weave.works/k8s/v1.8/net.yaml
```

Inspect the yaml contents:

```
1cat weave.yaml
```

On the kube-01 Kubernetes master node run the following commands:

```
1kubectl apply -f weave.yaml
```

The result will look like this:

```
1serviceaccount "weave-net" created
2clusterrole "weave-net" created
3clusterrolebinding "weave-net" created
4role "weave-net" created
5rolebinding "weave-net" created
6daemonset "weave-net" created
```

It may take a minute or two for DNS to be ready, continue to check for DNS to be readybefore moving on by running the following command:

```
1kubectl get pods --all-namespaces
```

The successful result will look like this, every container should be running:

```
1NAMESPACE                     NAME      READY     STATUS
              RESTARTS  AGE
2kube-system   etcd-kube-01            1/1       Running   0         5m 3kube-system
kube-apiserver-kube-01                 1/1       Running   0         6m
4kube-system   kube-controller-manager-kube-01   1/1Running   0         5m
5kube-system   kube-dns-6f4fd4bdf-whbhd        3/3                 Running   0
                                       6m6kube-system   kube-proxy-2hdhk
                                       1/1       Running   0         6m
7kube-         kube-proxy-tvhjk          1/        Running          5m
system                                   1         0
8kube-         kube-proxy-wspmv          1/        Running          5m
system                                   1         0
9kube-         kube-scheduler-           1/        Running          6m
system         kube-01                   1         0
10kube-        weave-net-9ghn5           2/        Running          5m
system                                   2         1
11kube-        weave-net-lh8tq           2/        Running          5m
system                                   2         0
12kube-        weave-net-qhr25           2/        Running          5m
system                                   2         0
```

Congratulations, now your Kubernetes cluster running on Ubuntu 16.04 is up and ready foryou to deploy a microservices application.

Step 6 - Deploying The Weaveworks Microservices Sock Shop

Next we will deploy a demo microservices application to your kubernetes cluster.First, on kube-01,

clone the microservices sock shop git repo:

```
1git clone https://github.com/microservices-demo/microservices-demo.git
```

Go to the microservices-demo/deploy/kubernetes folder:

```
1kubectl create namespace sock-shop
```

You will see the following result:

```
1namespace "sock-shop" created
```

Next apply the demo to your kubernetes cluster:

```
1kubectl apply -f complete-demo.yaml
```

You will see the following result:

```
1deployment "carts-db" created
2service "carts-db" created
3deployment "carts" created
4service "carts" created
5deployment "catalogue-db" created
6service "catalogue-db" created
7deployment "catalogue" created
8service "catalogue" created
9deployment "front-end" created
10service "front-end" created
11deployment "orders-db" created
12service "orders-db" created
13deployment "orders" created
14service "orders" created
15deployment "payment" created
16service "payment" created
17deployment "queue-master" created
18service "queue-master" created
19deployment "rabbitmq" created
20service "rabbitmq" created
21deployment "shipping" created
22service "shipping" created
23deployment "user-db" created
24service "user-db" created
25deployment "user" created
26service "user" created
```

Check to see if all of your pods are running:

```
1kubectl get pods --namespace sock-shop
```

You will see the following result when all pods are ready, they will have the status of“Running”:

```
1NAMESPACE     NAME                    READY     STATUS    RESTARTS   AGE
2kube-system   etcd-kube-01            1/1       Running   0          23m
```

```
3kube-system   kube-apiserver-kube-01           1/1     Running   0          24m
4kube-system   kube-controller-manager-kube-01  1/1     Running   0          23m
5kube-system   kube-dns-6f4fd4bdf-whbhd         3/3     Running   0          24m
6kube-system   kube-proxy-2hdhk                 1/1     Running   0          24m
7kube-system   kube-proxy-tvhjk                 1/1     Running   0          23m
8kube-system   kube-proxy-wspmv                 1/1     Running   0          23m
9kube-system   kube-scheduler-kube-01           1/1     Running   0          24m
10kube-system   weave-net-9ghn5                 2/2     Running   1          23m
11kube-system   weave-net-lh8tq                 2/2     Running   0          23m
12kube-system   weave-net-qhr25                 2/2     Running   0          23m
13sock-shop     carts-74f4558cb8-h9924          1/1     Running 0          11m
14sock-shop     carts-db-7fcddfbc79-v64fw       1/1     Running   0          11m
15sock-shop     catalogue-676d4b9f7c-55n4g      1/1     Running   0          11m
16sock-shop     catalogue-db-5c67cdc8cd-hvk96   1/1     Running   0          11m
17sock-shop     front-end-977bfd86-hq9x9        1/1     Running 0          11m
18sock-shop     orders-787bf5b89f-xfdl6         1/1     Running 0          11m
19sock-shop     orders-db-775655b675-gv456      1/1     Running   0          11m
20sock-shop     payment-75f75b467f-4zzqs        1/1     Running   0          11m
21sock-shop     queue-master-5c86964795-t8sjg   1/1     Running   0
```

Experiment: 24

Aim: ES6: Create a form for registration and Submit the details the hide the created from enable the display section.

Theory: ES6 or ECMAScript 6 is a scripting language specification which is standardized by ECMAScript International. This specification governs some languages such as JavaScript, ActionScript, and Jscript. ECMAScript is generally used for client-side scripting, and it is also used for writing server applications and services by using Node.js.

Procedure:
Step 1: Open any IDE to write the HTML code where ES6 is embedded in it.
Step 2: Type the Following code and Save it:

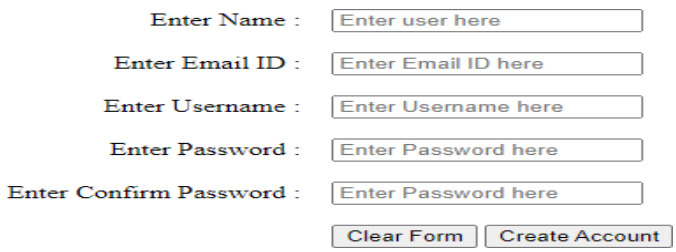
```
<head>
<title>Welcome To Registration Form</title>
</head>
<body>
<script>
function registration()
{
let name= document.getElementById("t1").value;
let email= document.getElementById("t2").value;
let uname= document.getElementById("t3").value;
let pwd= document.getElementById("t4").value;
let cpwd= document.getElementById("t5").value;
//email id expression code
let pwd_expression = /^(?=.*?[A-Z])(?=.*?[a-z])(?=.*?[0-9])(?=.*?[#?!@$%^&*-
])/;
let letters = /^[A-Za-z]+$/;
let filter = /^([a-zA-Z0-9_\.\-])+\@(([a-zA-Z0-9\-])+\.)+([a-zA-Z0-9]{2,4})+$/;
if(name==")
{
alert('Please enter your name');
}
else if(!letters.test(name))
{
alert('Name field required only alphabet characters');
}
```

```
else if(email=='')
{
alert('Please enter your user email id');
}
else if (!filter.test(email))
{
alert('Invalid email');
}
else if(uname=='')
{
alert('Please enter the user name.');
}
else if(!letters.test(uname))
{
alert('User name field required only alphabet characters');
}
else if(pwd=='')
{
alert('Please enter Password');
}
else if(cpwd=='')
{
alert('Enter Confirm Password');
}
else if(!pwd_expression.test(pwd))
{
alert ('Upper case, Lower case, Special character and Numeric letter are
required inPassword filed');
}
else if(pwd != cpwd)
{
alert ('Password not Matched');
}
else if(document.getElementById("t5").value.length < 6)
{
alert ('Password minimum length is 6');
}
else if(document.getElementById("t5").value.length > 12)
{
alert ('Password max length is 12');
}
```

```
else
{
alert('Thank You for Login & You are Redirecting to Campuslife Website');
// Redirecting to other page or webste code.
window.location = "http://www.campuslife.co.in";
}
disable();
}
function clearFunc()
{
document.getElementById("t1").value="";
document.getElementById("t2").value="";
document.getElementById("t3").value="";
document.getElementById("t4").value="";
document.getElementById("t5").value="";
}
function disable()
{
document.getElementById("t1").disabled=true;
document.getElementById("t2").disabled=true;
document.getElementById("t3").disabled=true;
document.getElementById("t4").disabled=true;
document.getElementById("t5").disabled=true;
}
</script>
<body>
<!-- Main div code -->
<div id="main">
<div class="h-tag">
<h2>Create Your Account</h2>
</div>
<!-- create account div -->
<div class="login">
<table cellspacing="2" align="center" cellpadding="8" border="0">
<tr>
<td align="right">Enter Name :</td>
<td><input type="text" placeholder="Enter user here" id="t1" class="tb"
/></td>
</tr>
<tr>
<td align="right">Enter Email ID :</td>
<td><input type="text" placeholder="Enter Email ID here" id="t2" class="tb"
```

```
/></td>
</tr>
<tr>
<td align="right">Enter Username :</td>
<td><input type="text" placeholder="Enter Username here" id="t3" class="tb"
/></td>
</tr>
<tr>
<td align="right">Enter Password :</td>
<td><input type="password" placeholder="Enter Password here" id="t4"
class="tb"
/></td>
</tr>
<tr>
<td align="right">Enter Confirm Password :</td>
<td><input type="password" placeholder="Enter Password here" id="t5"
class="tb"
/></td>
</tr>
<tr>
<td></td>
<td>
<input type="reset" value="Clear Form" onclick="clearFunc()" id="res"
class="btn" />
<input type="submit" value="Create Account" class="btn"
onclick="registration()"
/></td>
</tr>
</table>
</div>
<!-- create account box ending here.. -->
</div>
<!-- Main div ending here... -->
</html>
```

Step 3: Now open the Following File in any Browser. The Following Output looks like this



Step 4: Here after entering all the information when we press “Create Account” the datagoes to the Database. For clearing the Form we have added the “Clear Form” button.

This is how we Create User Registration Form Using ES6.