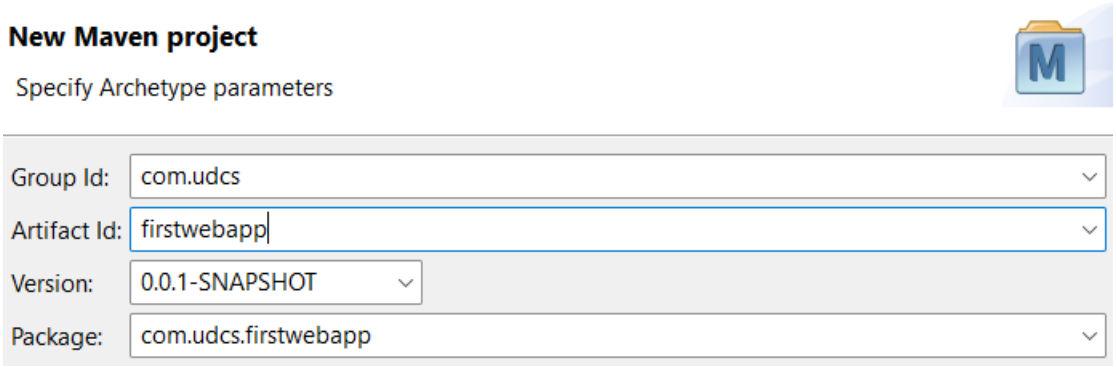
Practical No. 1

Aim: Using the software like JDK 1.8, Eclipse IDE, Apache tomcat server 7.0 Servlets, Spring framework design and develop Web applications using MVC Framework

Implementation:

1. Open Eclipse IDE and create a new Maven project
2. Select the catalog as internal and below that select the last option of the list mentioning ‘web_app’ and click ‘Next’
3. Enter the group_id: com.udcs and artifact_id: firstwebapp



4. Open pom.xml and edit to match the following

```
<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
http://maven.apache.org/maven-v4_0_0.xsd">
        <modelVersion>4.0.0</modelVersion>
        <groupId>com.gjccs</groupId>
        <artifactId>FirstWebApp</artifactId>
        <packaging>war</packaging>
        <version>0.0.1-SNAPSHOT</version>
        <name>FirstWebApp Maven Webapp</name>
        <properties>
                <jdk.version>1.7</jdk.version>
                <spring.version>3.2.13.RELEASE</spring.version>
                <jstl.version>1.2</jstl.version>
```

```
</properties>
<dependencies>
	<dependency>
		<groupId>org.springframework</groupId>
		<artifactId>spring-webmvc</artifactId>
		<version>${spring.version}</version>
	</dependency>
	<dependency>
		<groupId>javax.servlet</groupId>
		<artifactId>jstl</artifactId>
		<version>${jstl.version}</version>
	</dependency>
</dependencies>
<build>
	<plugins>
		<plugin>
			<groupId>org.apache.maven.plugins</groupId>
			<artifactId>maven-compiler-plugin</artifactId>
			<version>3.3</version>
			<configuration>
				<source>${jdk.version}</source>
				<target>${jdk.version}</target>
			</configuration>
		</plugin>
		<plugin>
			<groupId>org.apache.maven.plugins</groupId>
			<artifactId>maven-war-plugin</artifactId>
			<version>3.3.1</version>
		</plugin>
		<plugin>
			<groupId>org.eclipse.jetty</groupId>
```

```
                <artifactId>jetty-maven-plugin</artifactId>
                <version>9.2.11.v20150529</version>
                <configuration>
                    <scanIntervalSeconds>10</scanIntervalSeconds>
                    <webApp>
                        <contextPath>/spring3</contextPath>
                    </webApp>
                </configuration>
            </plugin>
            <plugin>
                <groupId>org.apache.maven.plugins</groupId>
                <artifactId>maven-eclipse-plugin</artifactId>
                <version>2.9</version>
                <configuration>
                    <downloadSources>true</downloadSources>
                    <downloadJavadocs>true</downloadJavadocs>
                    <wtpversion>2.0</wtpversion>
                    <wtpContextName>spring3</wtpContextName>
                </configuration>
            </plugin>
        </plugins>
    </build>
</project>
```

5. Go to web.xml in src/main/webapp/web-inf and edit it as shown below

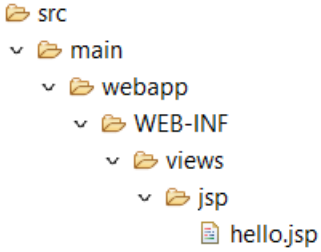
```
<web-app
    xmlns="http://java.sun.com/xml/ns/javaee"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
http://java.sun.com/xml/ns/javaee/web-app_2_5.xsd"
version="2.5">
    <display-name>MVC Application</display-name>
    <servlet>
        <servlet-name>spring-web</servlet-name>
        <servletclass>org.springframework.web.servlet.DispatcherServlet
        </servlet-class>
```

```
                <load-on-startup>1</load-on-startup>
                <!--
<init-param><param-name>contextConfigLocation</param-name><param-value>/WEB-
INF/spring-mvc-config.xml</param-value></init-param>
-->
        </servlet>
        <servlet-mapping>
                <servlet-name>spring-web</servlet-name>
                <url-pattern>/</url-pattern>
        </servlet-mapping>
</web-app>
```

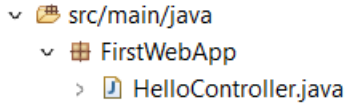
6. Right click on web-inf and create XML file named ‘spring-web-servlet.xml’ and edit as follows

```
<beans
        xmlns="http://www.springframework.org/schema/beans"
        xmlns:context="http://www.springframework.org/schema/context"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xmlns:mvc="http://www.springframework.org/schema/mvc"
xsi:schemaLocation="
 http://www.springframework.org/schema/beans
 http://www.springframework.org/schema/beans/spring-beans-3.2.xsd
 http://www.springframework.org/schema/mvc
 http://www.springframework.org/schema/mvc/spring-mvc-3.2.xsd
 http://www.springframework.org/schema/context
 http://www.springframework.org/schema/context/spring-context-3.2.xsd">
        <context:component-scan base-package="com.udcs" />
        <bean
class="org.springframework.web.servlet.view.InternalResourceViewResolver">
                <property name="prefix">
                        <value>/WEB-INF/views/</value>
                </property>
                <property name="suffix">
                        <value>.jsp</value>
                </property>
        </bean>
        <mvc:resources mapping="/resources/**" location="/resources/" />
        <mvc:annotation-driven />
</beans>
```

7. Create new folder in WEB-INF as ‘views and a subfolder as ‘jsp’ and create a ‘hello.jsp’ file in it as follows



8. Right click on src/main/resources in the file explorer and create new folder named ‘java’ but select ‘main’ in the folder creating wizard before that
9. Create a java class in java folder named ‘hellocontroller.java’ as shown below



10. Edit HelloController.java as follows

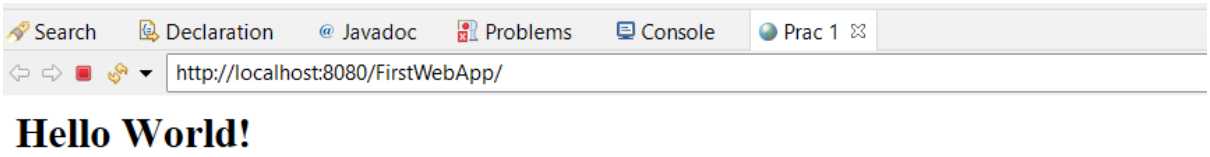
```
import java.util.regex.Matcher;
import java.util.regex.Pattern;

public class RegexMatches {
   public static void main(String args[]) {

      // String to be scanned to find the pattern.
      String line = "This order was placed for QT3000! OK?";
      String pattern = "(.*)(\\d+)(.*)";

      // Create a Pattern object
      Pattern r = Pattern.compile(pattern);

      // Now create matcher object.
      Matcher m = r.matcher(line);
      if (m.find()) {
         System.out.println("Found value: " + m.group(0));
         System.out.println("Found value: " + m.group(1));
         System.out.println("Found value: " + m.group(2));
      } else {
         System.out.println("NO MATCH");
      }
   }
}
```

11. Edit the ‘hello.jsp’ file as follows

```
mySelection = app.activeDocument.selection;
myDoc = app.activeDocument;
if (mySelection instanceof Array)
{
        selSwatches = myDoc.swatches.getSelected();

        if(selSwatches.length != 0)
                for (i=0; i<mySelection.length; i++)
                {
                        if(mySelection[i].typename == "PathItem" ||
mySelection[i].typename == "CompoundPathItem")
                        {
                                selItem = mySelection[i];
                                selItem.filled = true;
```

```
                                swatchIndex = Math.round( Math.random() *
(selSwatches.length - 1 ));

                                if(selItem.typename == "PathItem")
                                        selItem.fillColor = selSwatches[swatchIndex].color;
                                else
                                        selItem.pathItems[0].fillColor =
selSwatches[swatchIndex].color;

                        }
                }
}
```

12. In IDE, right click on firstwebapp > Maven > update Project > check on ‘force update snapshot’ > Finish
13. Right click on project > run as > Maven build
14. In the following pop-up, enter the goals as ‘clean install’
15. Right click on project > Run as > Run on server > Apache Tomcat v9 > select any file in configured > Finish
    To see the output



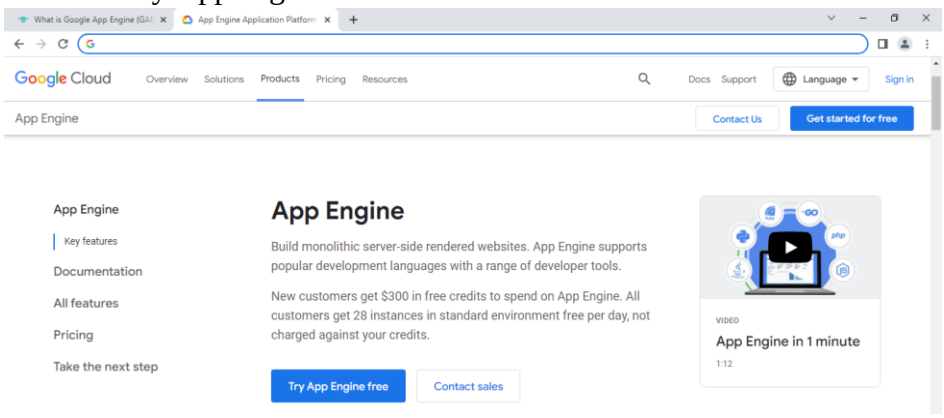
Practical No. 2

Aim: Installing and configuring the required platform for Google App Engine

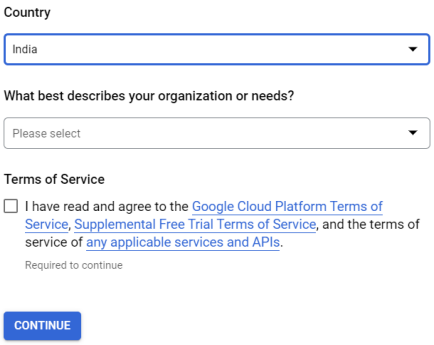
Implementation:

A. Making Google App Engine account

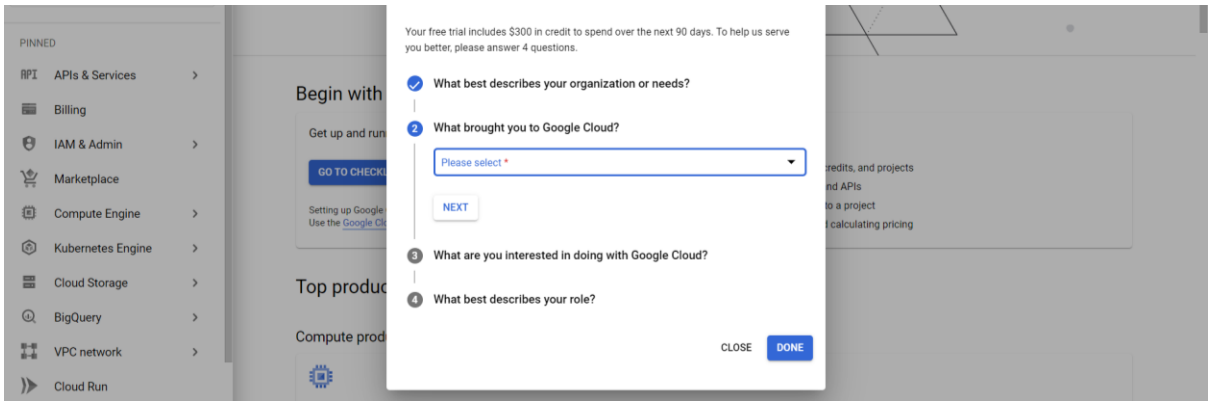
1. Open your google account and go to the following link
   https://cloud.google.com/appengine
2. Click on Try App engine free button



3. This page will appear, add country and choose other in describes and click Continue.



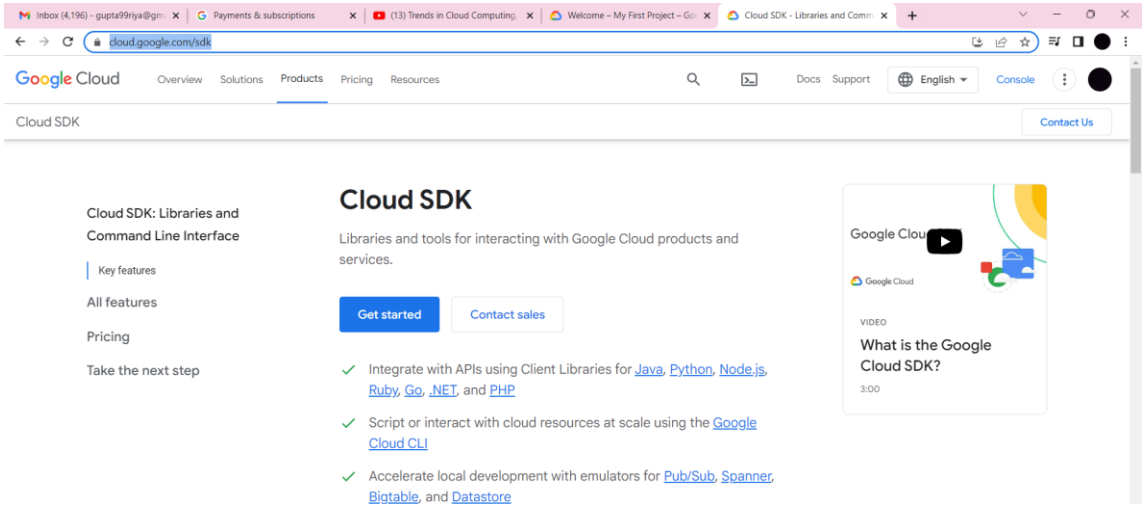
4. Add you card details (Visa or MasterCard only) and do the payment of Rs. 2
   Fill this according to you purpose



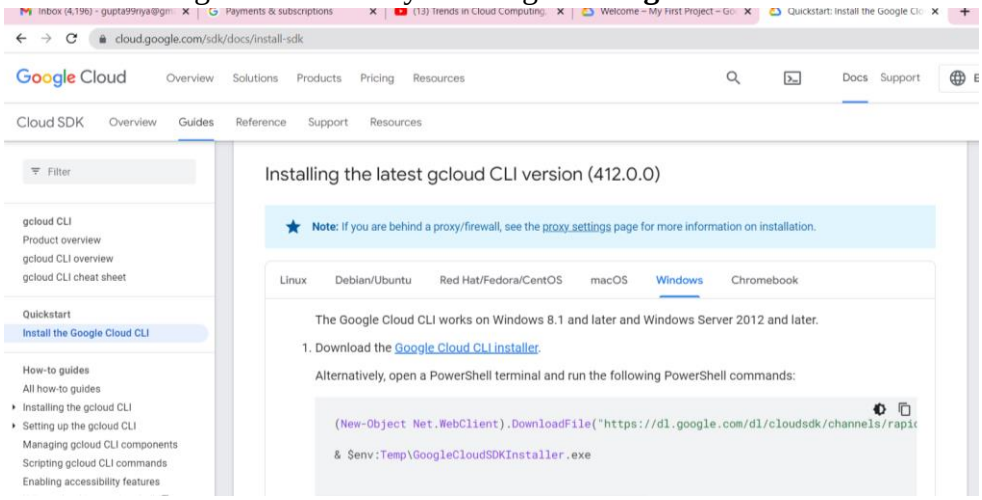
After all procedure we will get,

B. Using GAE account, to download Google Cloud SDK.

5. Go to google and search **google cloud sdk** go to site https://cloud.google.com/sdk then Get Started (you should use this with same account used in google app engine)

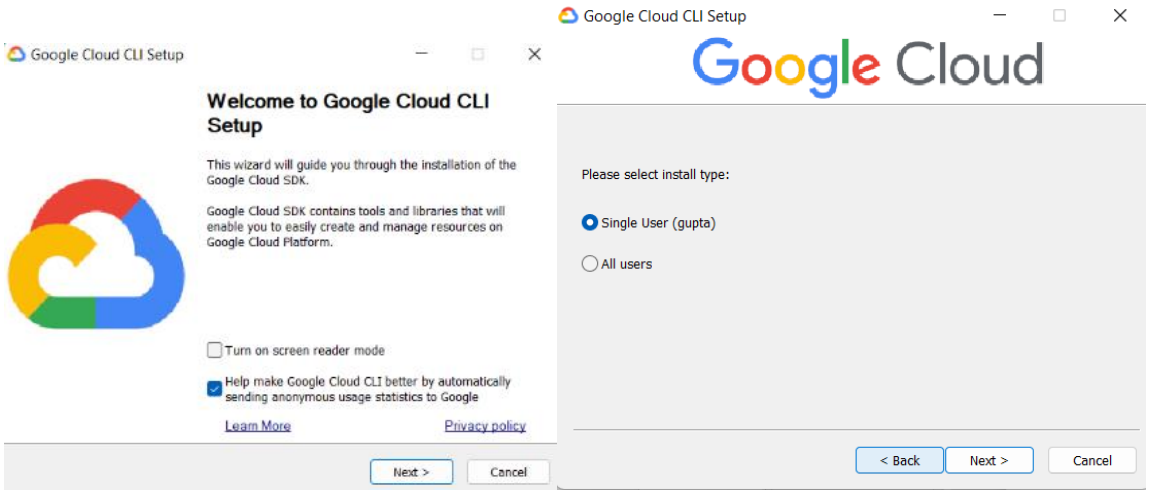


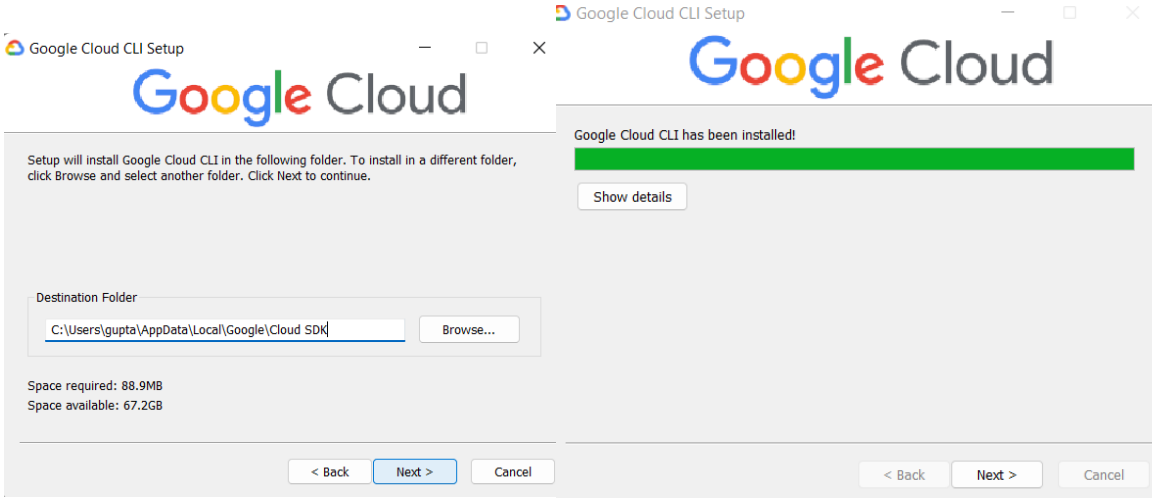
6. Install the Google Cloud CLI by clicking on **Google Cloud CLI installer**



7. Launch the installer and follow the prompts. The installer is signed by Google LLC.

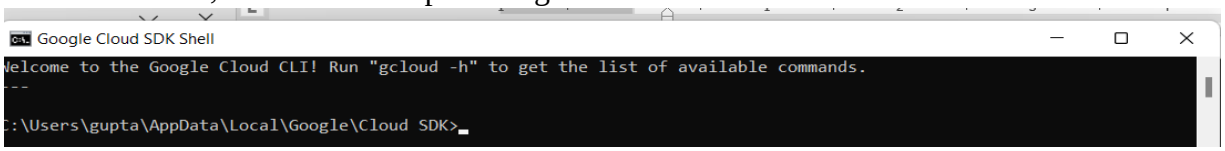
8.





After installing it will ask for log in as shown.

Select 5 from it, close this and open Google Cloud SDK Shell it will look like



Add the installer path in system also.

Run command ‘ gcloud components install app-engine-java’

Practical No. 3

Aim: Studying the features of the GAE PaaS model.

Platform as a Service (PaaS) - What is it?

PaaS is a category of cloud computing services that provide a computing platform and a solution stack as a service.

Along with software as a service (SaaS) and infrastructure as a service (laaS), it is a service model of cloud computing. In this model, the consumer creates the software using tools and/or libraries from the provider/vendor. The consumer also controls software deployment and configuration settings.

The provider provides the networks, servers, storage and other services. PaaS offerings facilitate the deployment of applications without the cost and complexity of buying and managing the underlying hardware and software and provisioning hosting capabilities.

PaaS Key Features

1. Services to develop, test, deploy, host and maintain applications in the same integrated development environment
2. Web-based management/administration consoles
   - ✓ Reducing the need for system administration/dev ops
   - ✓ Resource utilization monitoring capabilities
   - ✓ Easily identify bottlenecks
   - ✓ Multi-tenant architecture
   - ✓ Certain PaaS offerings attempt to support use of the application by many concurrent users, by providing concurrency management, scalability, fail-over and security
   - ✓ Support for development team collaboration
   - ✓ Pay for what you use billing model Stop

PaaS - Popular offerings

Heroku

One of the first cloud platforms, has been in development since June 2007, when it supported only the Ruby programming language, but has since added support for Java, Node.js, Scala, Clojure, Python and (undocumented) PHP

Windows Azure

Microsoft's cloud computing platform used to build, deploy and manage applications through a global network of Microsoft- managed datacenters

dotCloud

Founded in 2008 by Solomon Hykes, dotCloud is the first application platform designed from the ground up for modern service-oriented development

Cloud Foundry

- Developed by VMware released under the terms of the Apache License 2.0
- Primarily written in Ruby
- AppCloud runs on Cloud Foundry
- Since it is open sourced, ActiveState has created a commercial distribution of the Cloud Foundry **software for enterprises to host their own private PaaS**

Engine Yard

A San Francisco, California based, privately held platform as a service company focused on Ruby on Rails and PHP, and recently announced support for Node.js deployment and management

Google App Engine (often referred to as GAE or simply App Engine, and also used by the acronym GAE/J)

- A cloud computing platform for developing and hosting web applications in Google-managed data centers
- Applications are sandboxed and run across multiple servers
- Offers automatic scaling for web applications-as the number of requests increases for an application, App Engine automatically allocates more resources for the web application to handle the additional demand
- Is free up to a certain level of consumed resources. Fees are charged for additional storage, bandwidth, or instance hours required by the application
- First released as a preview version in April 2008, and came out of preview in September 2011

What is Google App Engine?

Google App Engine lets you run web applications on Google's infrastructure. App Engine applications are easy to build, easy to maintain, and easy to scale as your traffic and data storage needs grow. With App Engine, there are no servers to maintain: You just upload your application, and it's ready to serve your users.

The Application Environment

Google App Engine makes it easy to build an application that runs reliably, even under heavy load and with large amounts of data. App Engine includes the following features;

- Dynamic web serving, with full support for common web technologies
- Persistent storage with queries, sorting and transactions
- Automatic scaling and load balancing
- APIs for authenticating users and sending email using Google Accounts
- A fully featured local development environment that simulates Google App Engine on your computer

Your application can run in one of three runtime environments: the Go environment, the Java environment, and the Python environment, which gives you a choice of Python 2.5 or Python 2.7.

Why App Engine?

Pros

- Easy to Get Started
- Automatic Scalability
- The Reliability, Performance, and Security of Google's Infrastructure
- Costs less
- There is a generous free usage quota and you only pay for what you use

Cons

- Sandboxed environment limits the scope of your application
- Although we can pay for certain additional resources, there are some that have a hard limit

Traditional Way

1. Write your code · · · ·
2. Configure & Deploy Web server (Apache/Tomcat)
3. Configure & Deploy SQL database
4. Maintain all of these infrastructure
5. Cost of building and maintaining the infrastructure

App Engine Way

1. Write your code
2. A set of simple configurations to let App Engine know how to serve your application

Tools Bundled with the SDK

Development Server

Uploading and Managing an App

Uploading and Downloading Data

ProtoRPC

webapp Framework

Local Unit Testing

Appstats

Included Libraries (Python 2.5)

- Django, PyCrypto, YAML, zipimport

Included Libraries (Python 2.7)

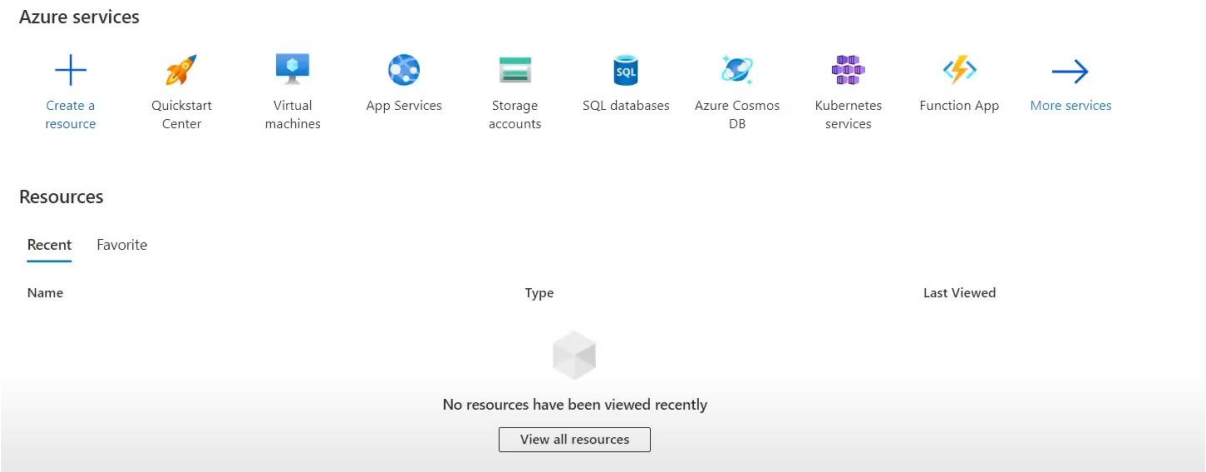
- Jinja2, PIL, webapp2, etc QBurst I meet.google.com is sharing your screen.

Practical No. 4

Aim: Developing an ASP.NET based web application on the Azure platform

Implementation:

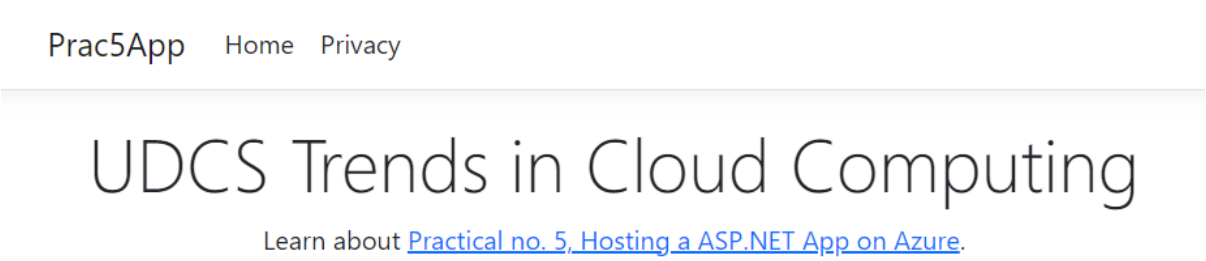
1. Go to the Azure portal and click on free account at top right of the home page and go to start page
2. Create a new account and enter debit card details to finalize the account generation process
3. You can view the following image once done



4. Make sure to have installed Visual Studio 2022, and open it
5. Create a new project > ASP.NET Core Web App > Next > Create
6. In your File tree, open pages folder and edit the index.chtml like so

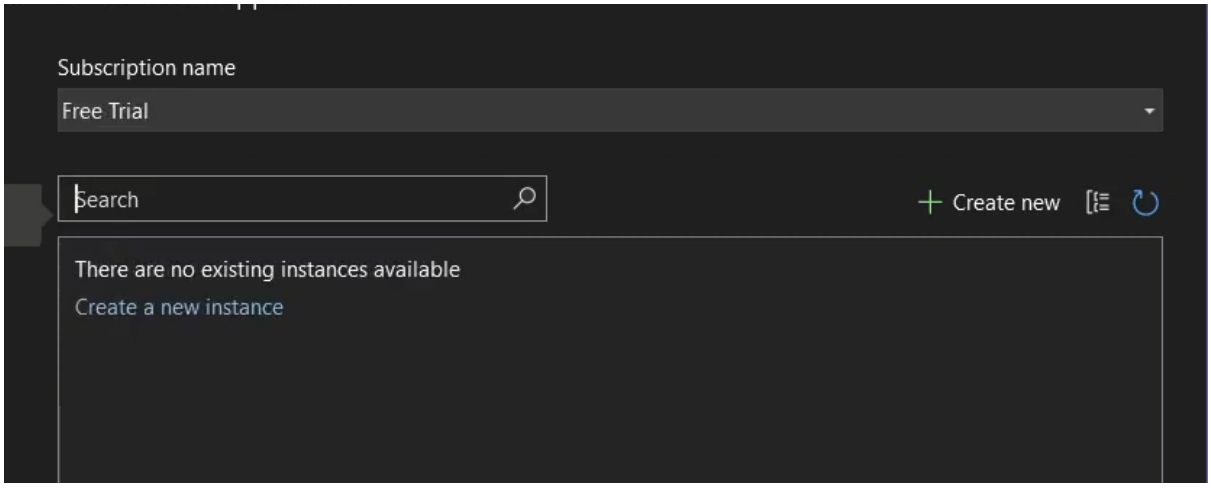
```
<div class="text-center">
    <h1 class="display-4">UDCS Trends in Cloud Computing</h1>
    <p>Learn about <a href="https://docs.microsoft.com/aspnet/core">Practical no. 5, Hosting a ASP.NET App on Azure</a>.</p>
</div>
```

7. Run in localhost to verify its working

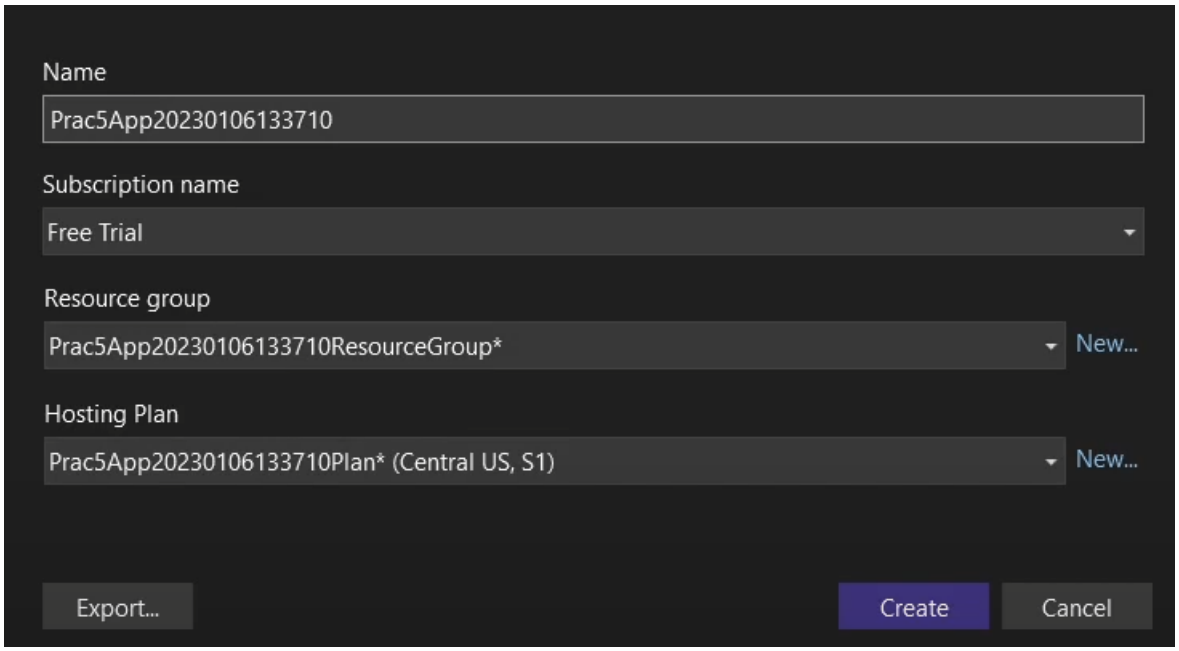


8. Now in Build tab, click on public [AppName]

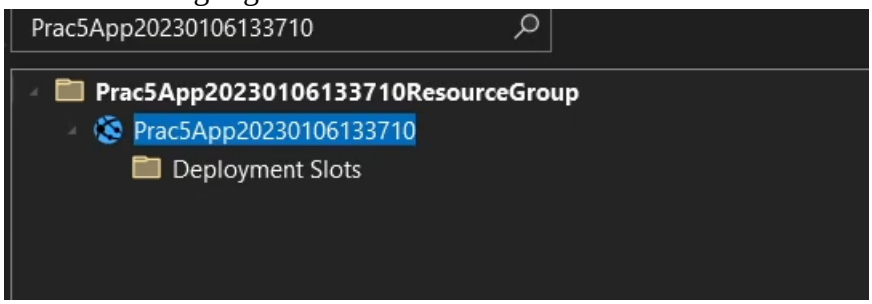
9. Select ‘Azure’ > Next > Azure App Service (windows) > Next
10. Make sure your Microsoft account used for Azure is signed in and currently selected
11. Go for free trial and click on the ‘+ Create new’ to create a new resource



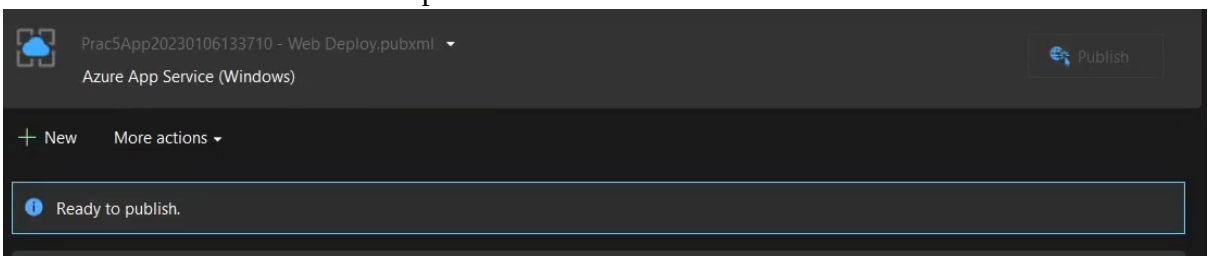
12. Click on create



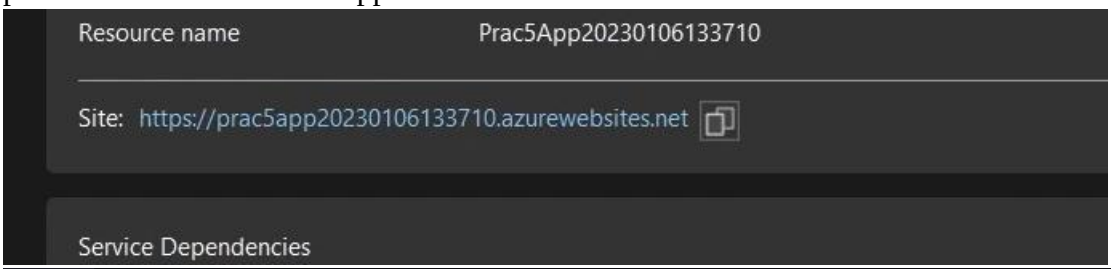
13. Click on the highlighted resource and click on Finish

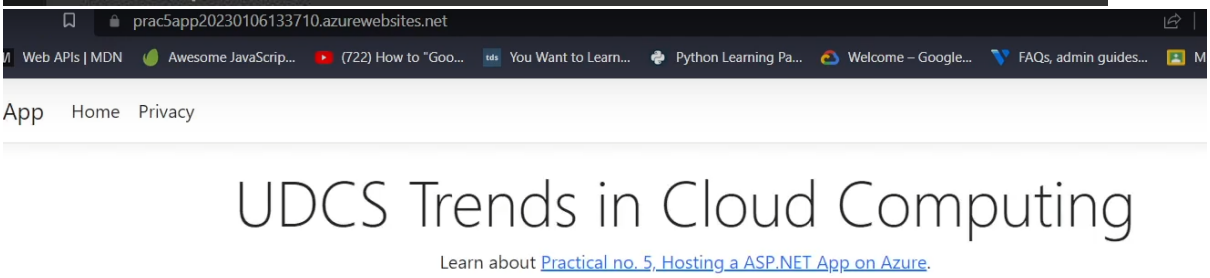


14. Click on close and then click on publish



15. Scroll down to find and copy the URL generated and paste it in a browser to view the published and hosted web app



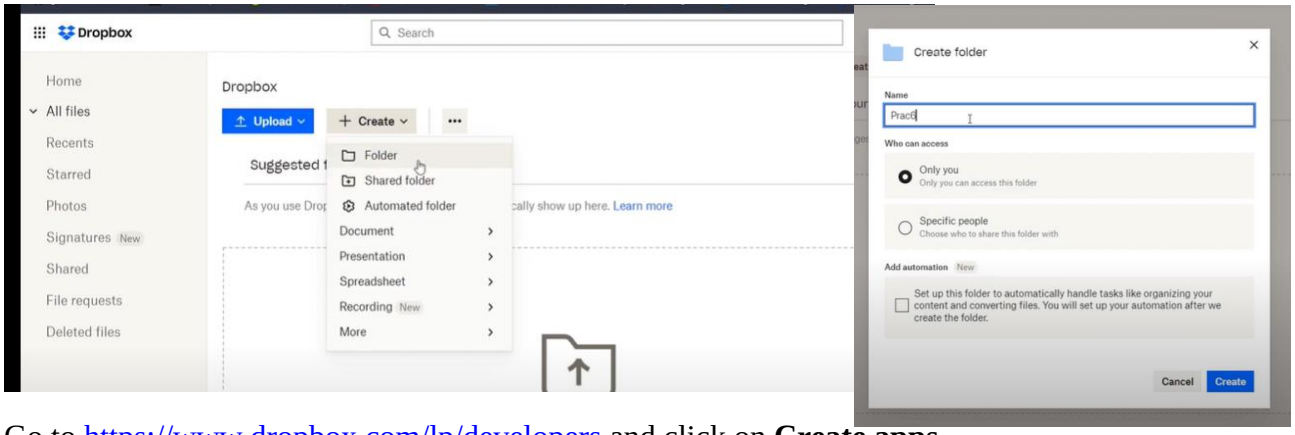


Practical 5

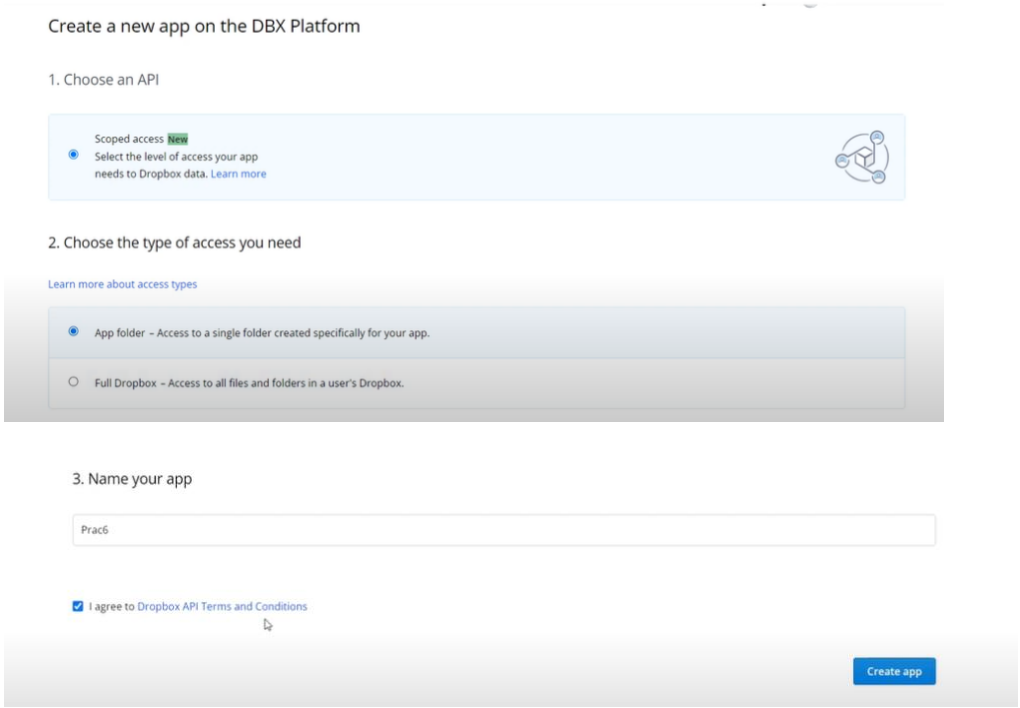
Aim: Creating an application in Dropbox to store data securely. Develop a source code using Dropbox API for updating and retrieving files.

Implementation:

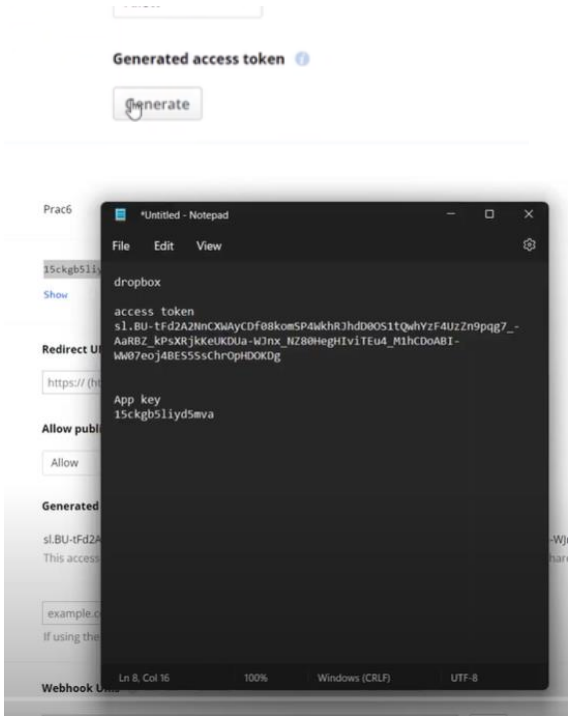
1. Go to https://www.dropbox.com/ and create an account. You can sign in with your google account.
2. Click on Continue with 2 GB Dropbox Basic plan at the end of the page. Create a 'Prac6' folder.



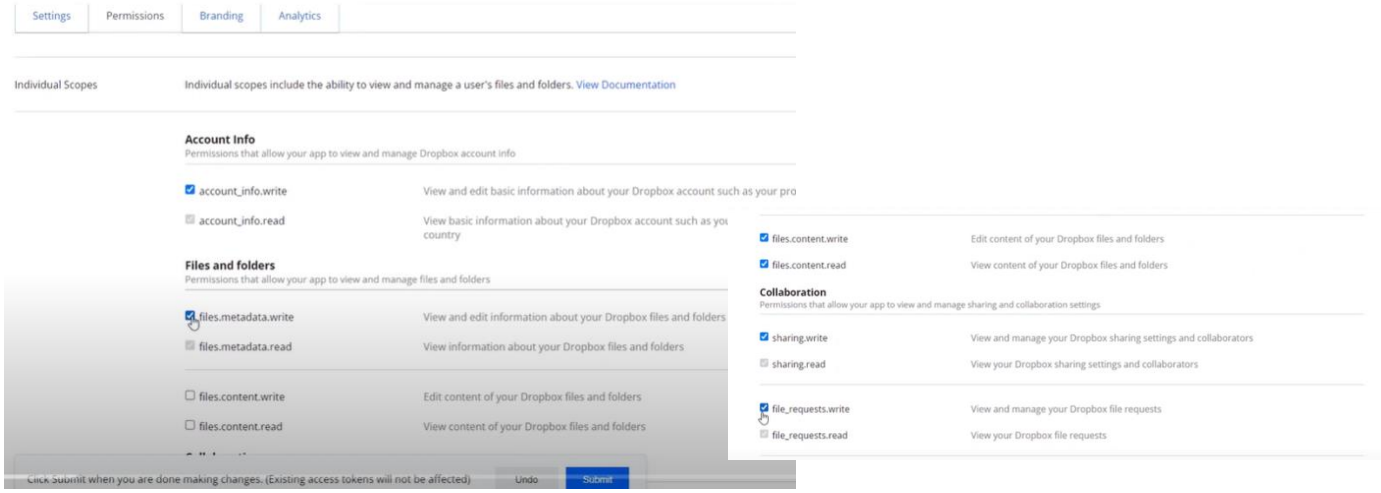
3. Go to https://www.dropbox.com/lp/developers and click on **Create apps**.

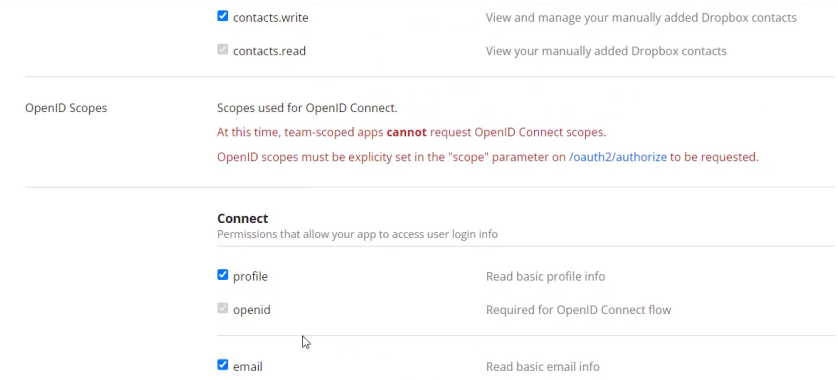


4. Click on generate, copy the generated access token and App key into a notepad file for later use.



5. Go to permissions and check all the following boxes. After that click on submit.



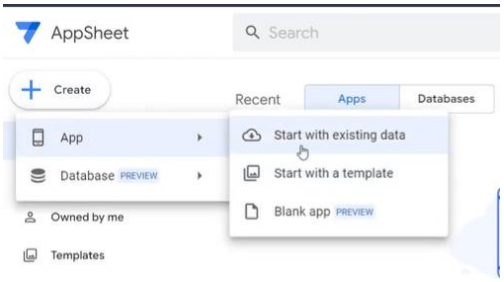


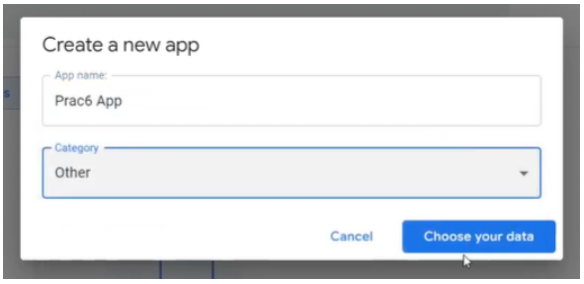
6. Go to https://colab.research.google.com/, create a new notebook and write the following code. Create a notepad file with some texts, upload it to dropbox folder 'Prac6'.

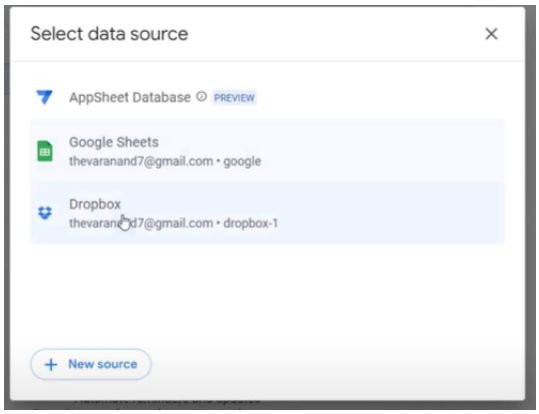
```
import dropbox
dropbox_access_token=
'sl.BUTEvih4dpk6Cma3FXLv2ToJfIGwWDJ6WmruJdbLeDFVLGqoE7g_Jcmy2Yfujqz_eHH
0Rr82G0gWDIASZUQgDo6co4WrbN-
YiuG5JgKiOGBRc7WlanGHglTejJhuqZ8LMLxdaWq9gvRF'    #Enter your own access
token
dropbox_path= "/Prac6/aa.txt"
computer_path="aa.txt"

client = dropbox.Dropbox(dropbox_access_token)
print("[SUCCESS] dropbox account linked")

client.files_upload(open(computer_path, "rb").read(), dropbox_path)
print("[UPLOADED] {}".format(computer_path))
```

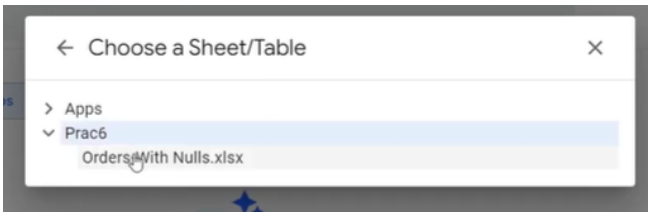
7. Go to https://www.wisdomaxis.com/technology/software/data/for-reports/, get your data sample from here and also upload the excel file in dropbox folder 'Prac6'.

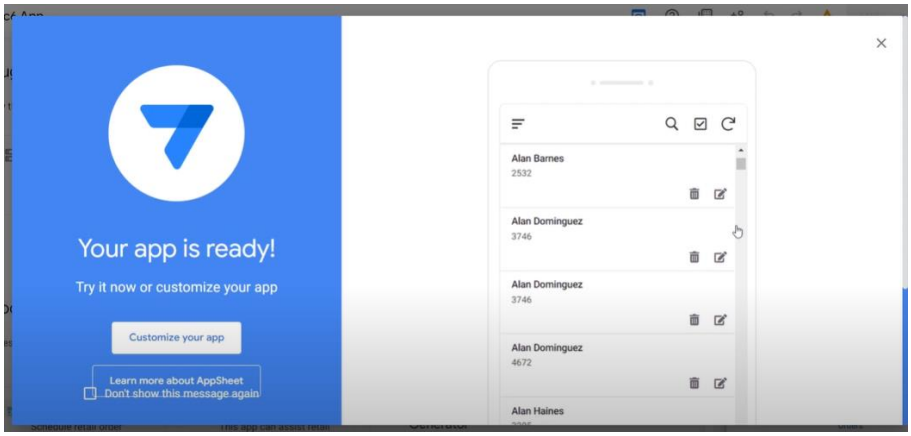
8. Go to https://about.appsheet.com/, sign in with google. Create a new app. Choose the sample for the data.











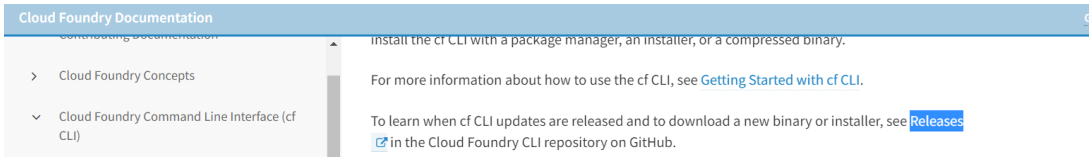
Application created.

Practical No. 6

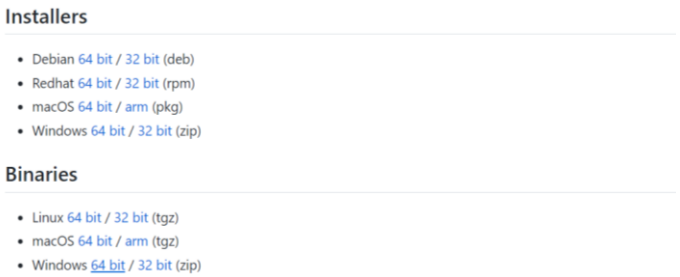
Aim: Installing Cloud Foundry in localhost and exploring CF commands.

Implementation:

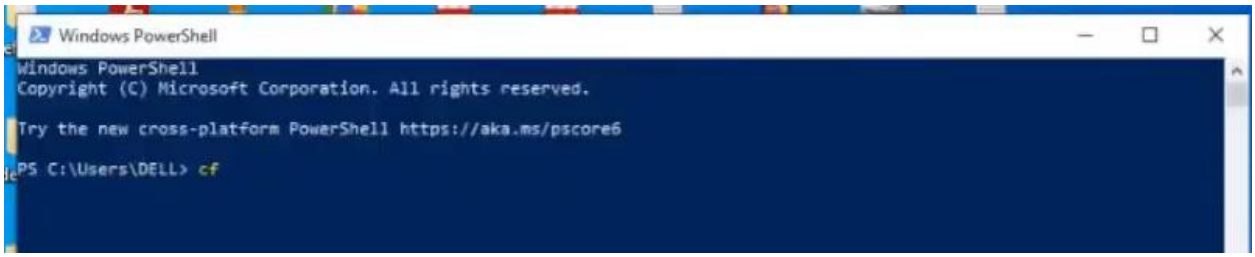
1. **Installing** Windows Powershell for CloudFoundry https://docs.cloudfoundry.org/cf-cli/install-go-cli.html

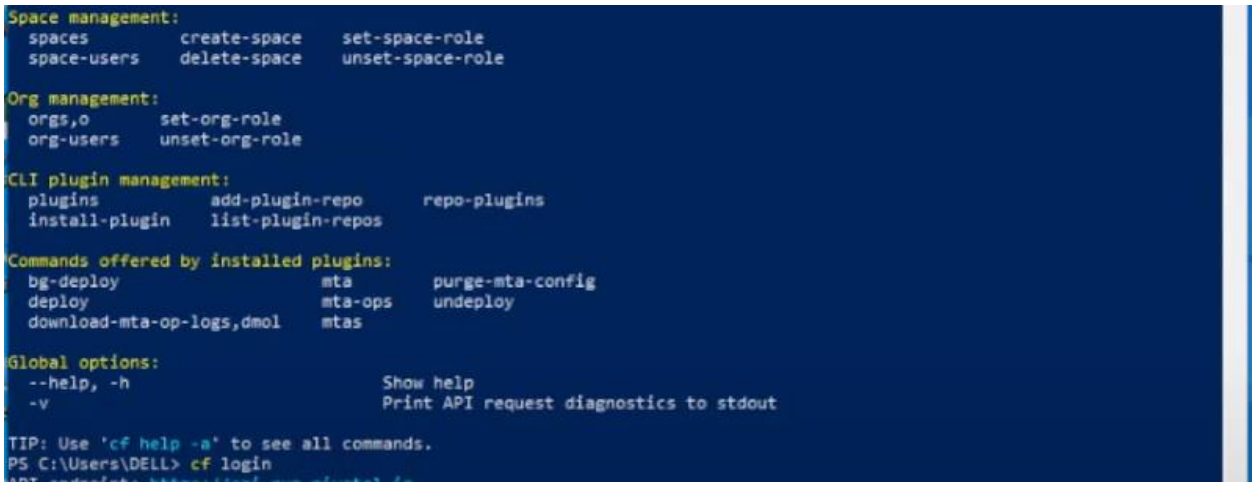


Download installer or binary file:



Install the file and set a system path. Then open the Windows Powershell. Type >cf to check, and if you're getting the information regarding CF, then you've successfully installed it.

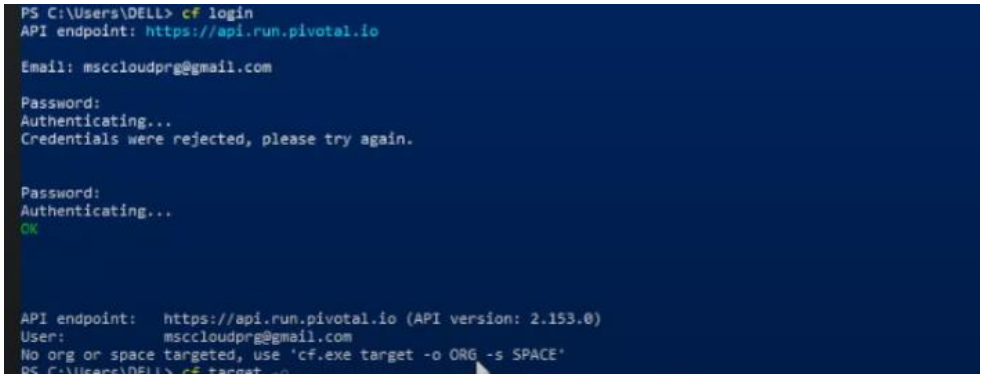




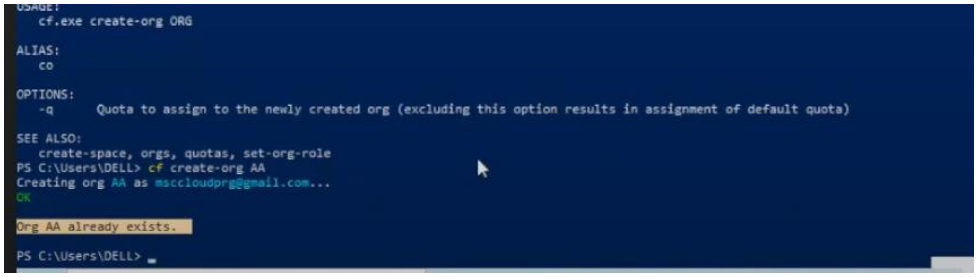
Exploring Commands

2. Login with the command **>cf login**

It will have the same endpoint as IBM and we will use the same email and password as the account created in the IBM Cloud practical.



3. Generate source target using the command **>cf create-org AA**

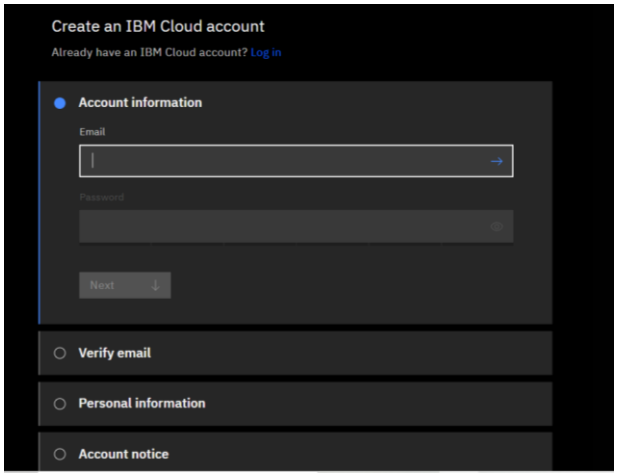


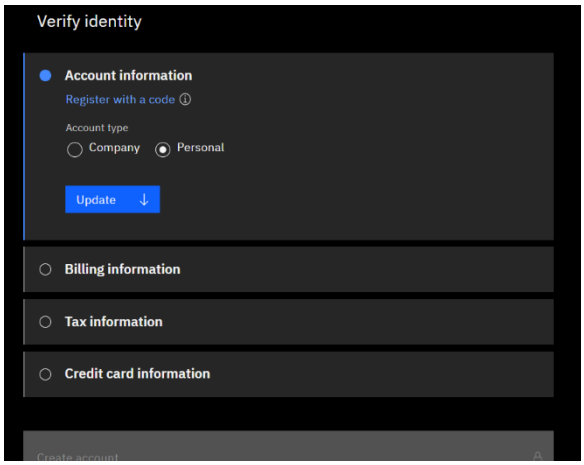
Practical No. 7

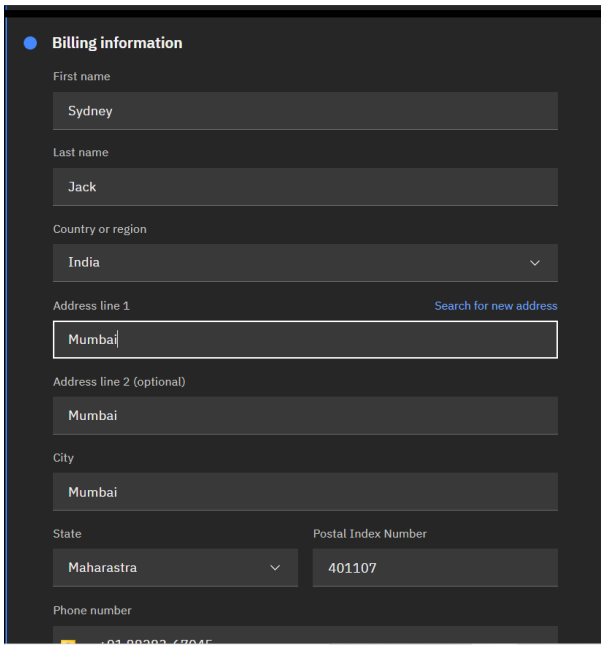
Aim: Cloud application development using IBM Bluemix Cloud.

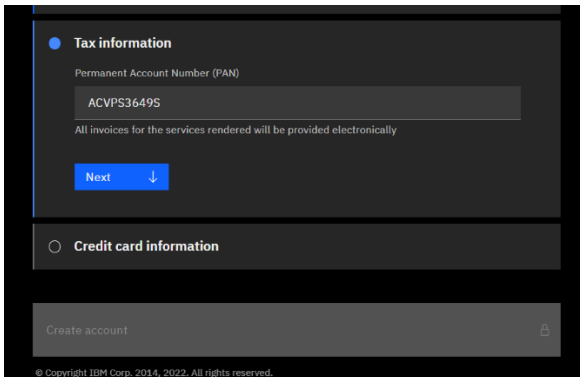
Implementation:

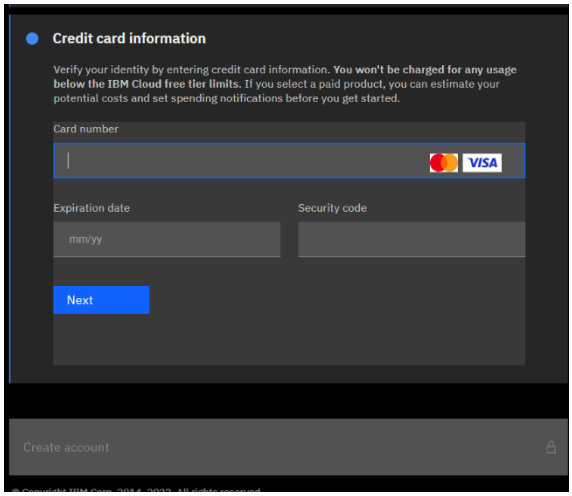
1. Log in to IBM Cloud at cloud.ibm.com.



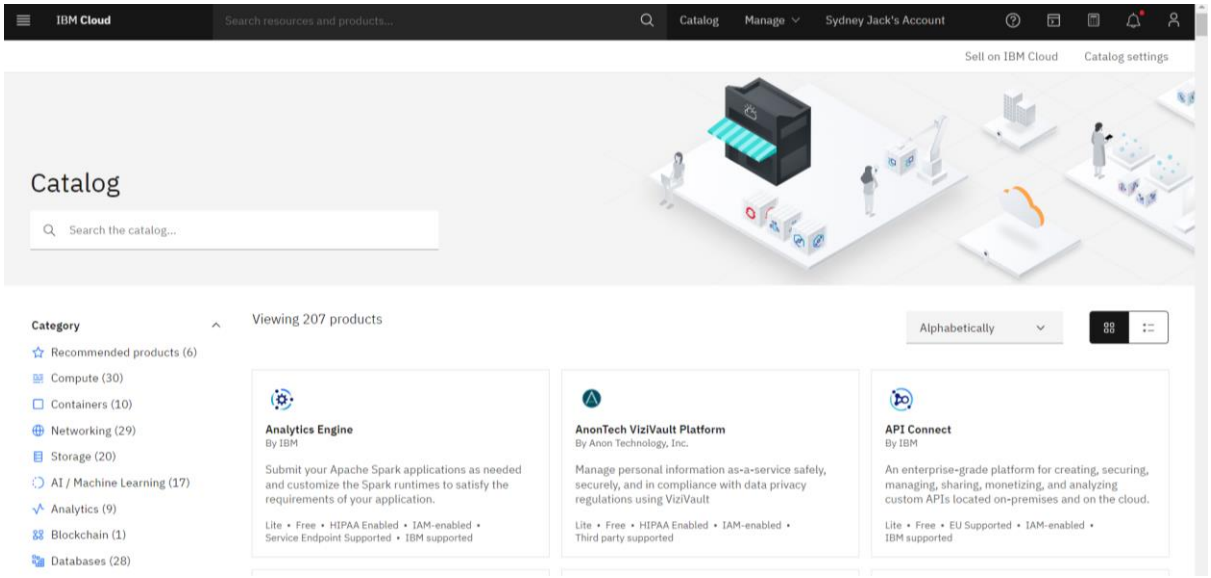




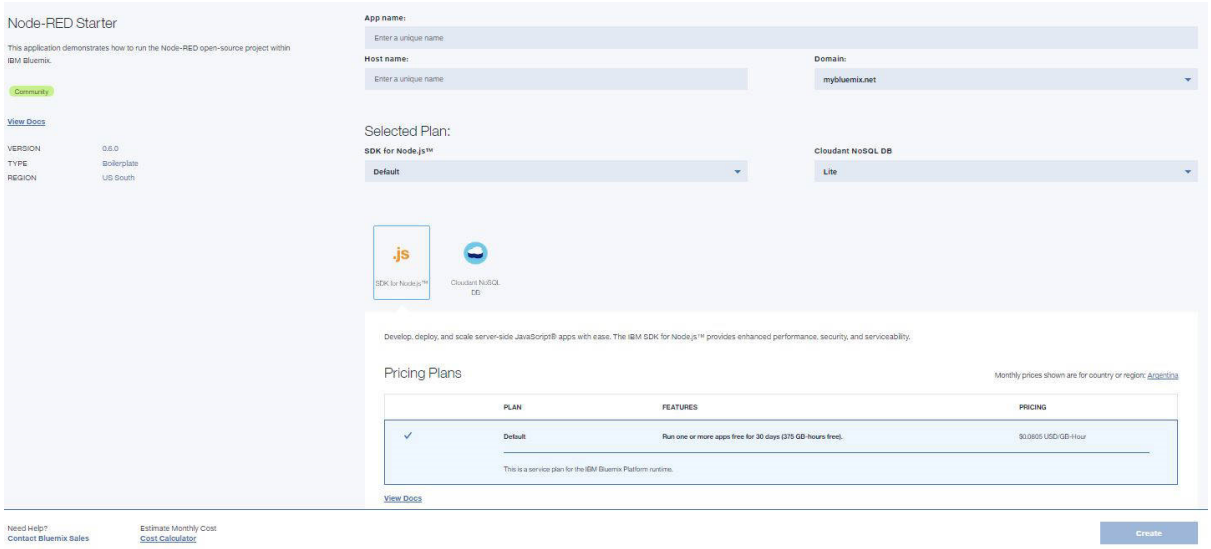




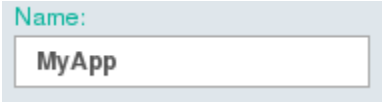
2. Go to the catalog (cloud.ibm.com/catalog).



3. Click the **Node-RED** boilerplate and fill in the required data to create an instance.

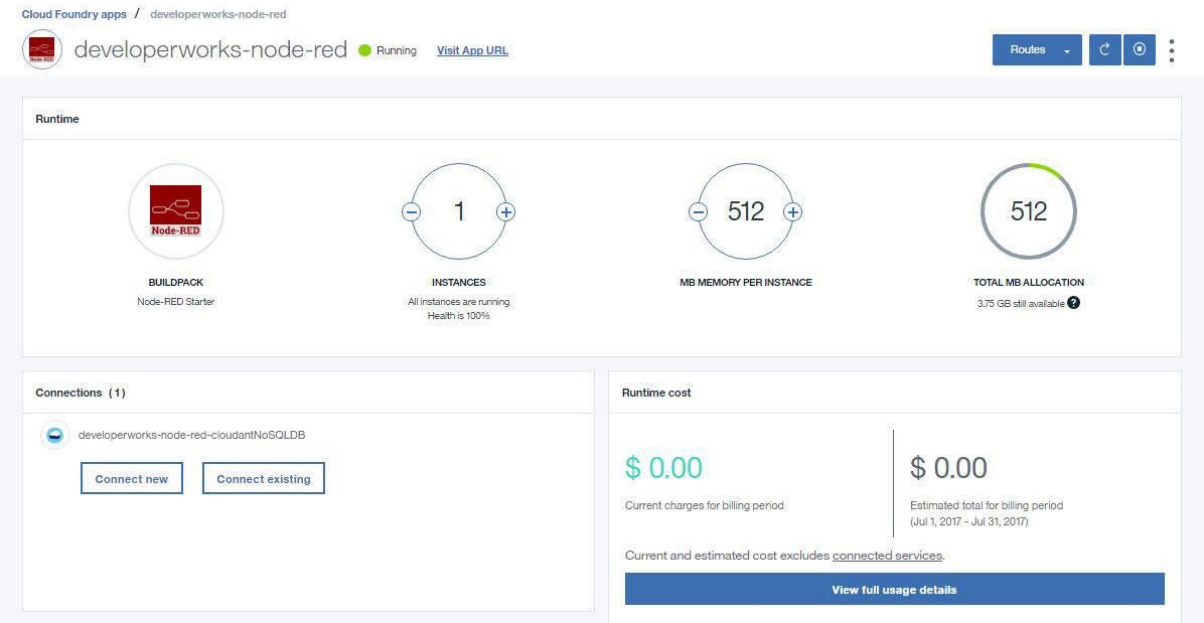


4. Specify an app name and click **Create**.



After the process completes, the environment is ready to use. When you access the IBM Cloud dashboard, the Node-RED instance is in Running state.

5. Click **Overview** on the left to access the application information. The information instance is displayed.

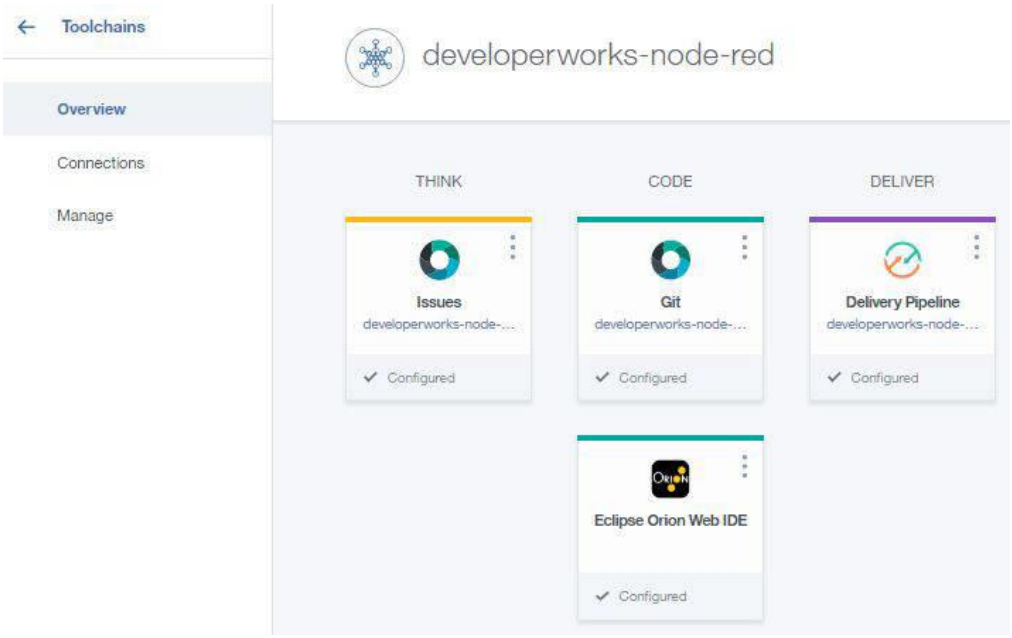


If you named the app app101-node-red, its route would be:

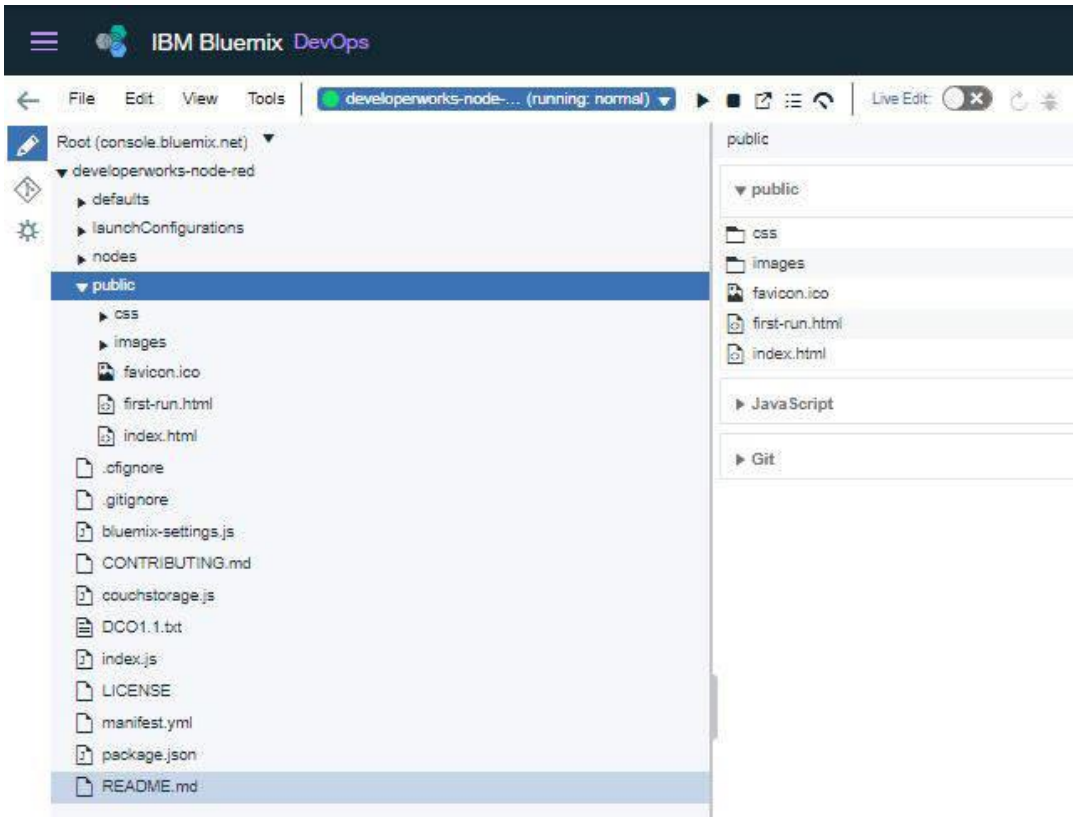
`http://app101-node-red.mybluemix.net`.

6. To modify this app to meet your requirements, you need to have access to its code. IBM Cloud provides a way to allocate space in a GIT repository, where you can access application code and files. Create this space by clicking **Enable**, located in the lower-right corner, and then **Create** in the next panel.

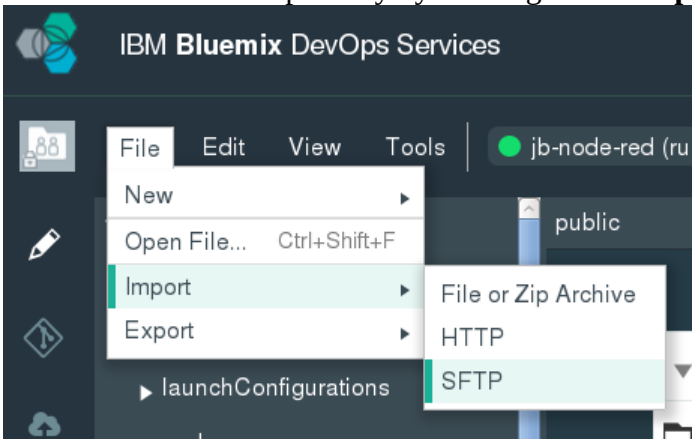
   When the process finishes, you see the Toolchain.



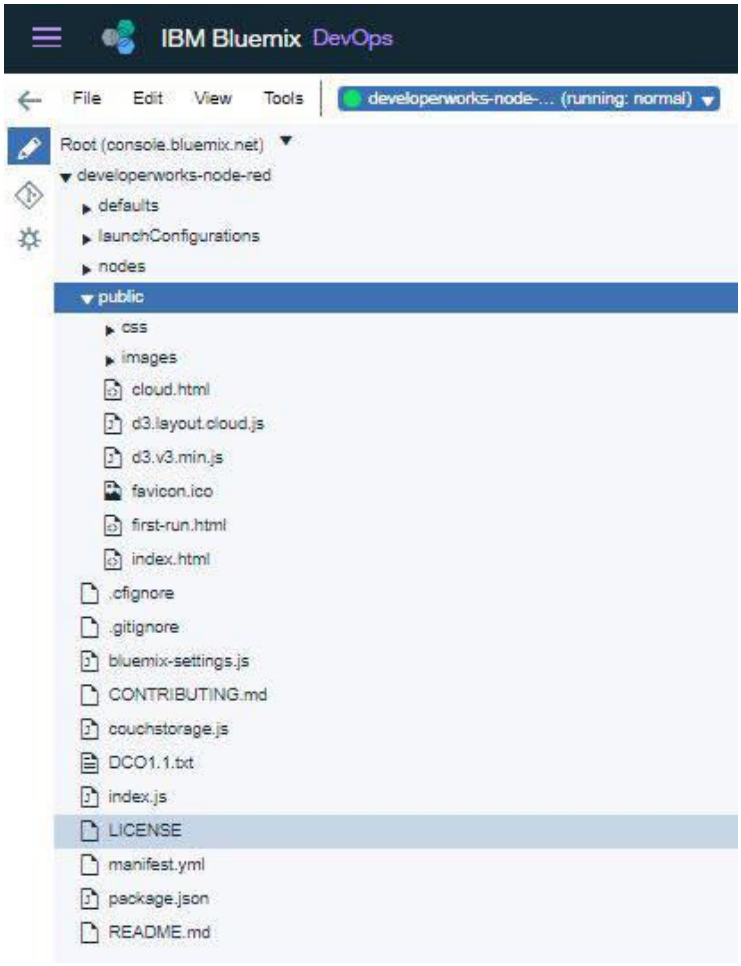
7. To access the code, click **Eclipse Orion Web IDE**, select your application on the left-hand side, and then click the public directory.



8. To set up your app, you need to add and modify these files in the public directory:
   - cloud.html
   - d3.layout.cloud.js
   - d3.v3.min.jsDownload these files to your workstation from GitHub at https://github.com/barabasj/Bluemix-App.
9. Upload the files to the GIT repository by clicking **File > Import > File or Zip Archive**.

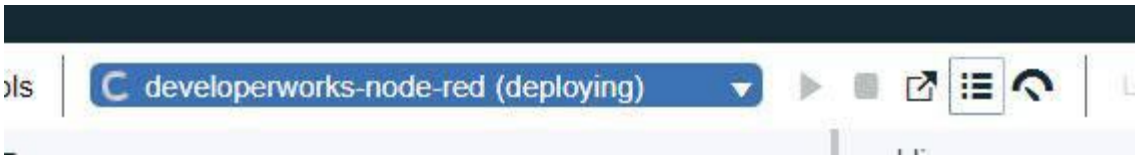


10. After you upload all the files, you need to publish all of the contents of the GIT repository to the running instance on IBM Cloud.

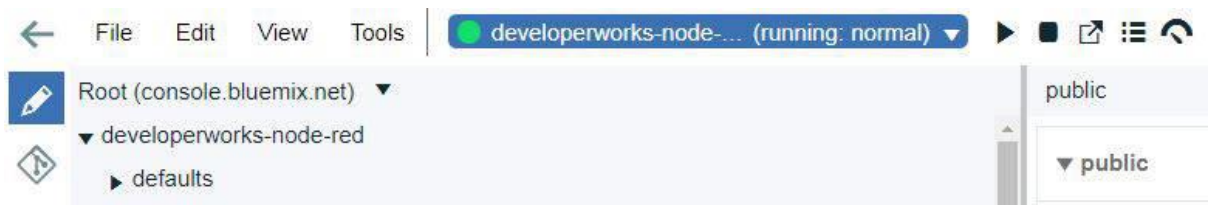


To deploy the changes, press the arrow button. Another option is to enable the **LiveEdit** switch to deploy every change in auto mode.

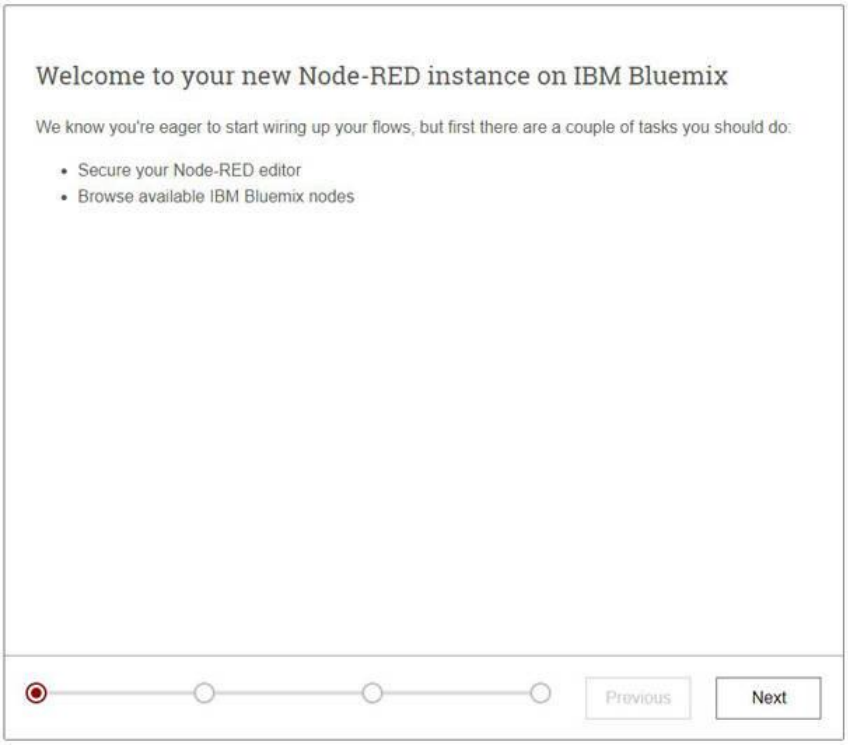
You see a (deploying) state while it is processing.



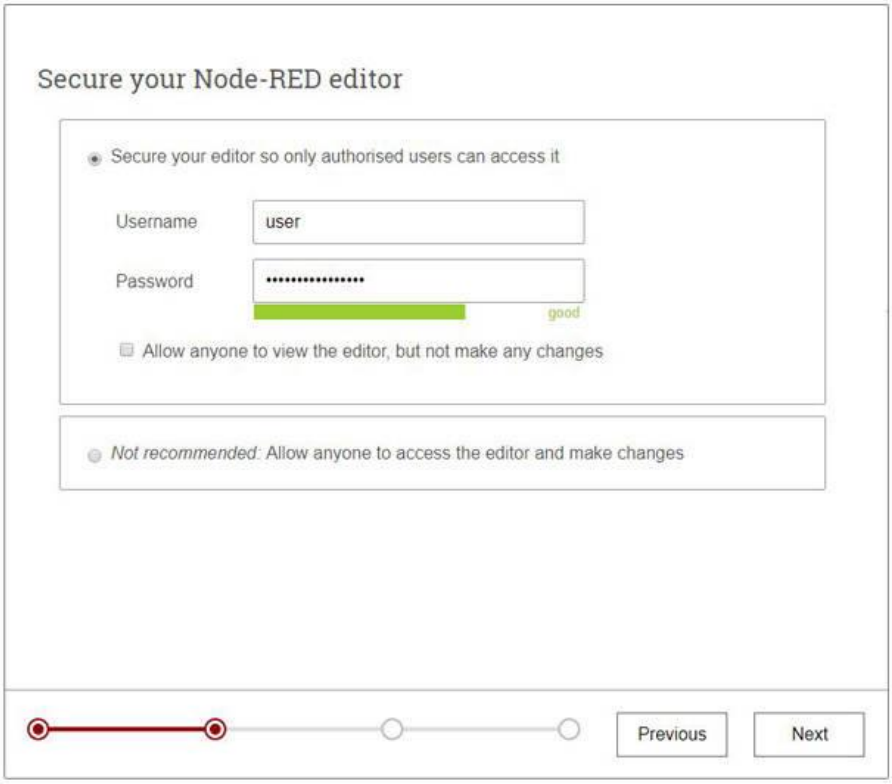
When the deployment process finishes, the green sign shows (running: normal) again.



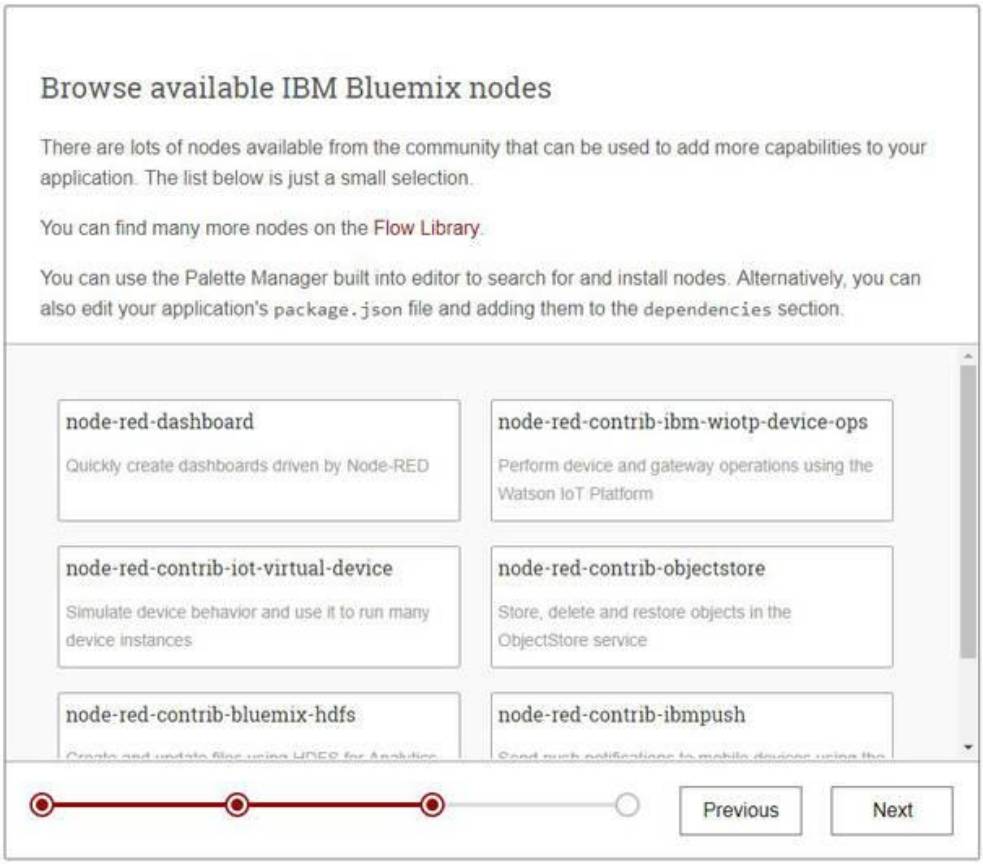
11. To create the Node-RED app that will feed the cloud.html file that you just uploaded and deployed, open the Node-RED editor in the browser by clicking the **Link to Application** icon Welcome panel to access the deployed application.
12. The first time you run a Node-Red instance, you need to specify its properties. On the first panel, click **Next**.



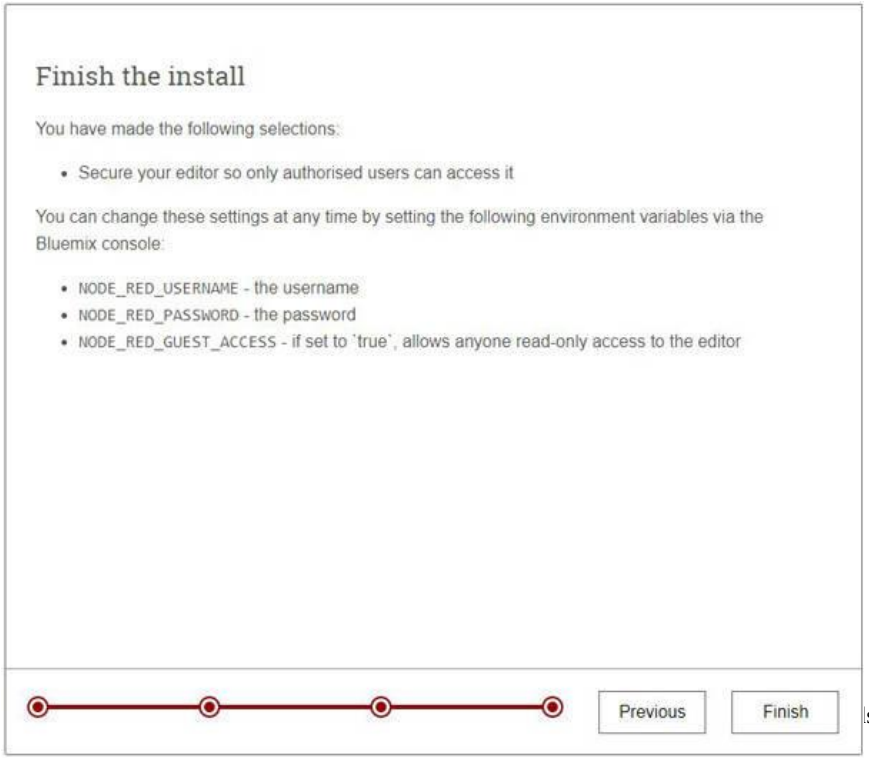
13. Enter your username and password, and click **Next**.



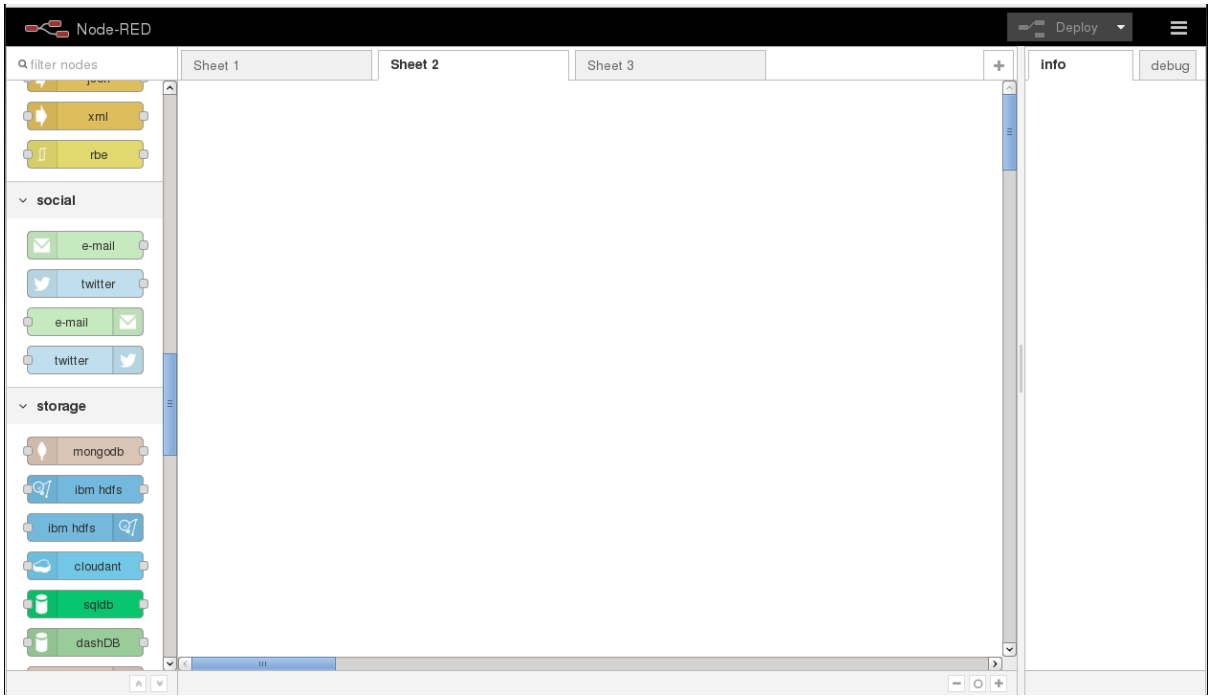
14. Read through the general information panel, and click **Next**.



15. Click **Finish** to complete the installation. Your configuration is saved and the Node-Red instance starts.

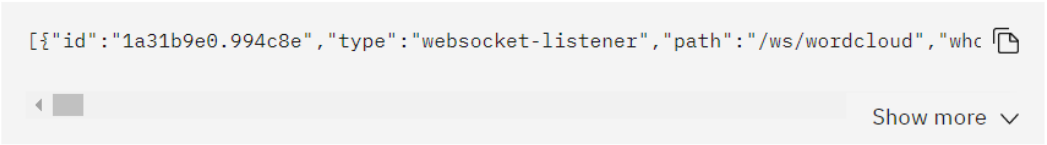


16. Select **Go to your Node-RED flow editor** to access the Workflow Editor. Notice your application URL: {your-instance-name}.mybluemix.net.
17. Enter your username and password and click **Login** to open the flow editor.

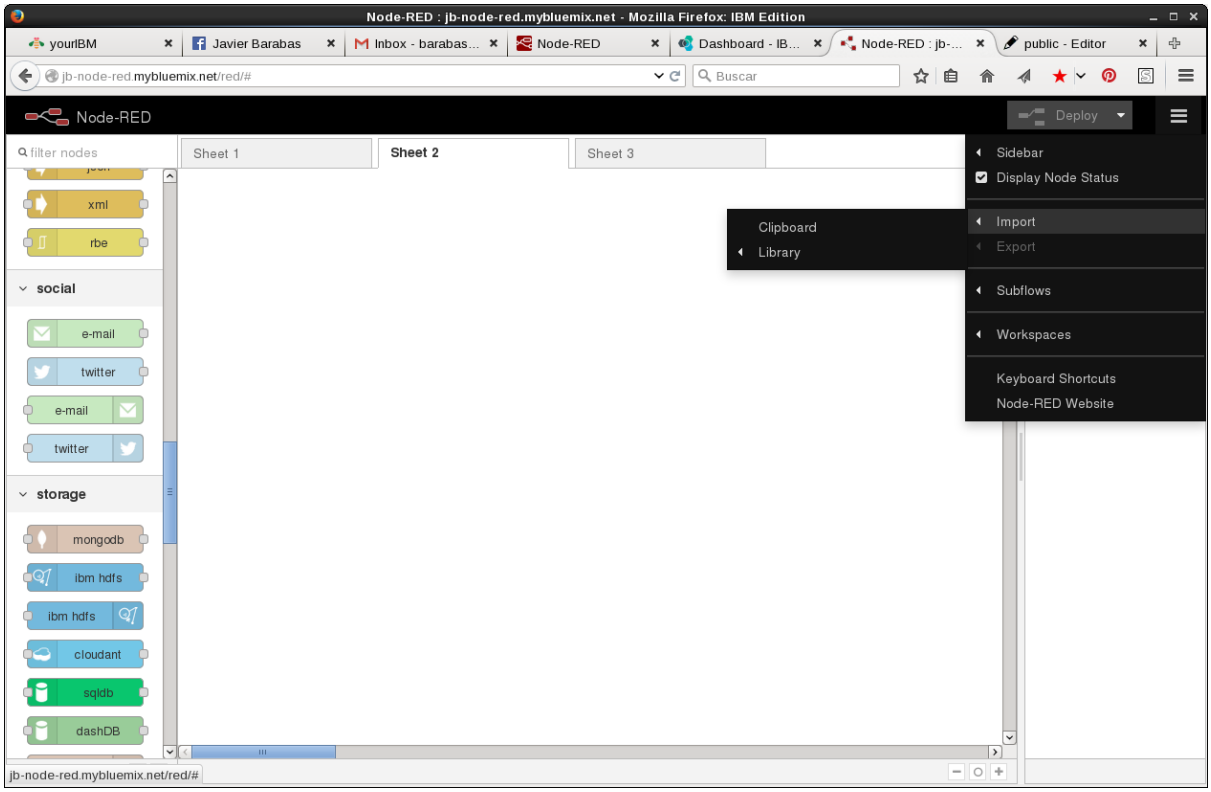


The left-hand nav bar contains all of the tools, services, and functions that you need to compose IBM Cloud apps inside the Node-RED environment. Using the simple drag-and-drop interface, you can build just about any complex app you like. In addition, you can import and export complex code to transfer and reuse. You can use this process to populate your app quickly and easily.

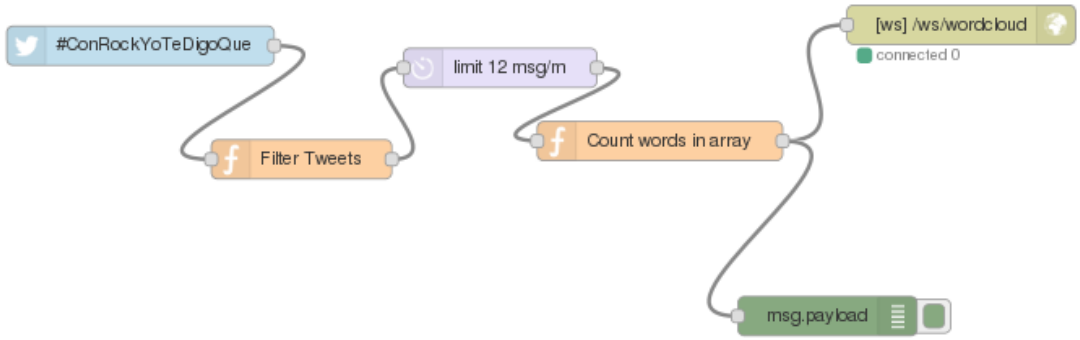
18. One of the files in the GitHub repo mentioned above is wordcloud.txt. This file contains the text that's exported from the app that you are creating. Select and copy the contents of wordcloud.txt



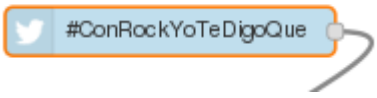
19. Next, click **Import > Clipboard** in the Node-RED editor.



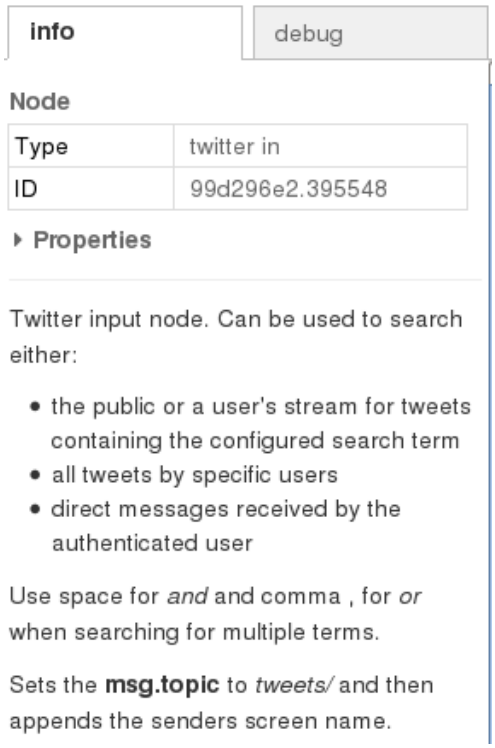
The objects that represent the application are shown in the following image:



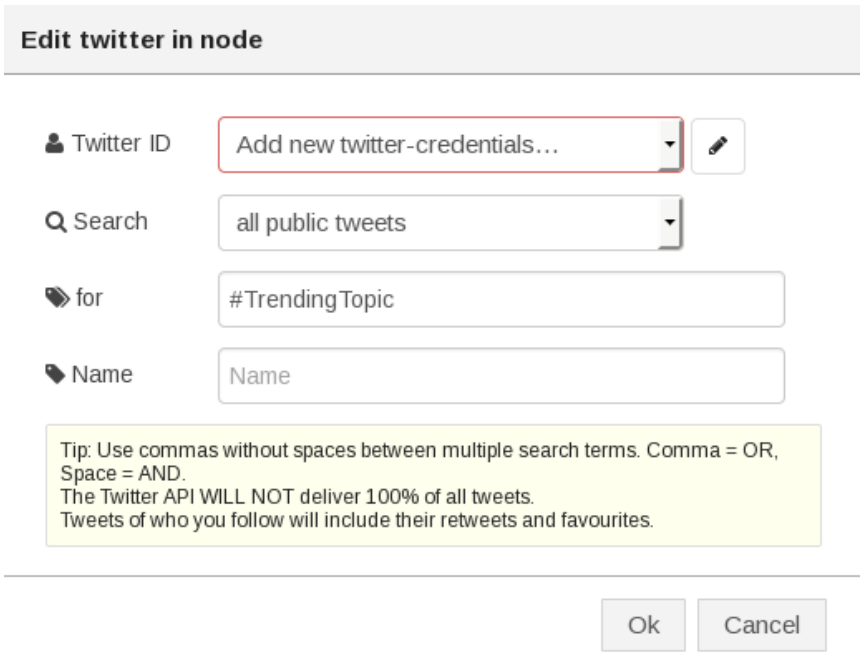
20. You need to configure some nodes in order to get the application to work. The flow starts reading public tweets accessed by a personal account, filtering the results with a trending topic that ensures that you have matches to be processed by your app. Open the first node (Twitter input):



The help information is displayed on the right:

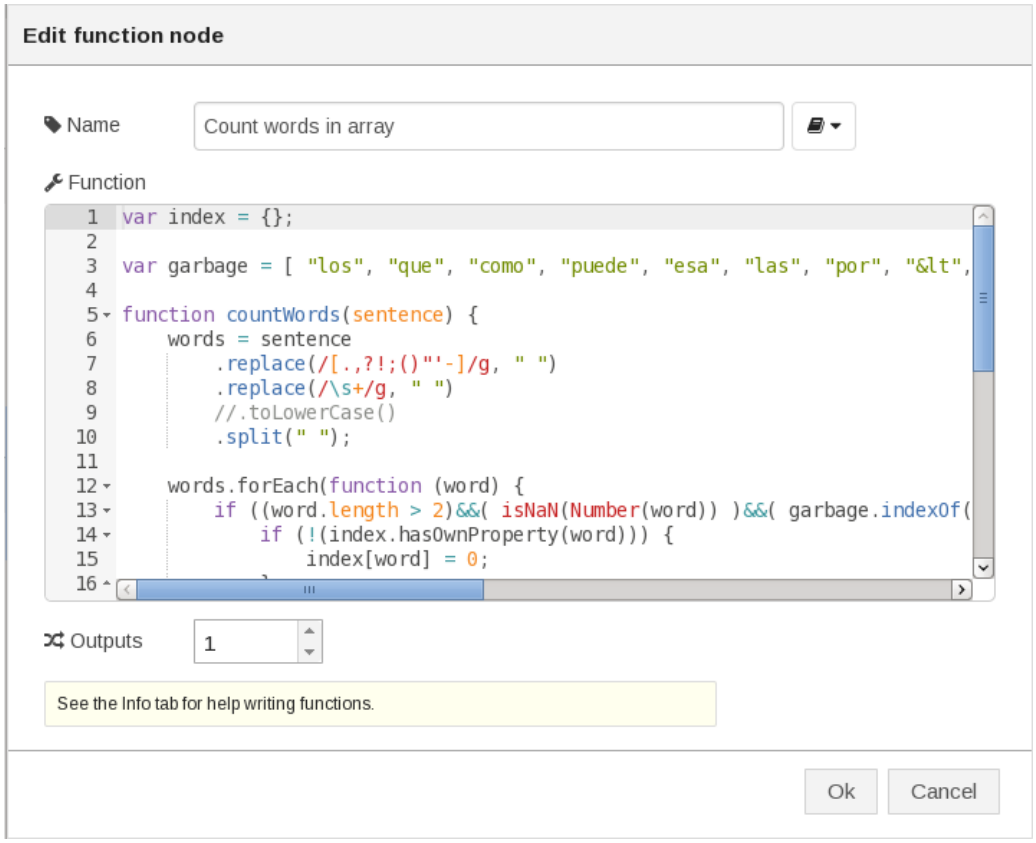


21. Open the Twitter node by double-clicking on it.



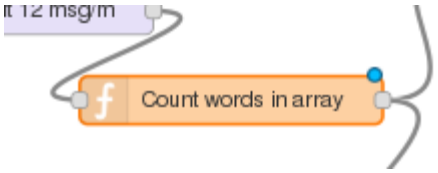
22. Enter your Twitter ID and any topic you want to display. You can obtain better results by specifying Trending Topic.

To exclude all non-significant words from the resulting tweets, depending on the language of the matching tweets, you can update the "Count words in array" function node (line 3) to reflect the selected ones. The variable "garbage" must contain all words selected to be ignored by the counters.

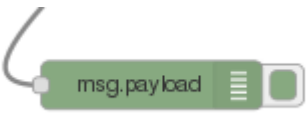


23. To publish the changes, click **Deploy** in the upper right.

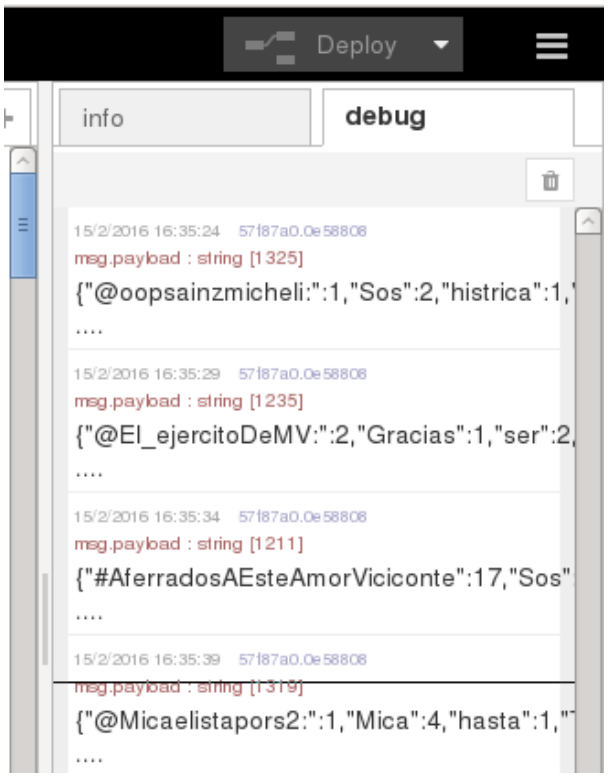
Every time a change to an object is made, a little dot is displayed on the node. Once the app has been deployed, the dot disappears.



To debug your app and display the processed tweets, a debug node has been added to the flow:



Matching tweets are displayed in the debug window of the GUI:



24. To display the word cloud created by the app, access the following URL:

http://_<app_name>_.mybluemix.net/cloud.html.

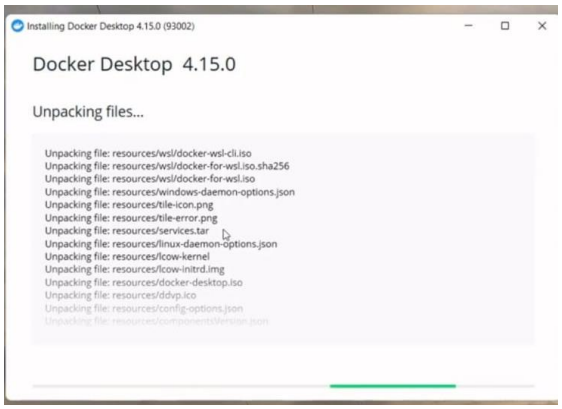
Practical No. 8

Aim: Installing and Configuring Dockers in localhost and running multiple images on a Docker Platform.

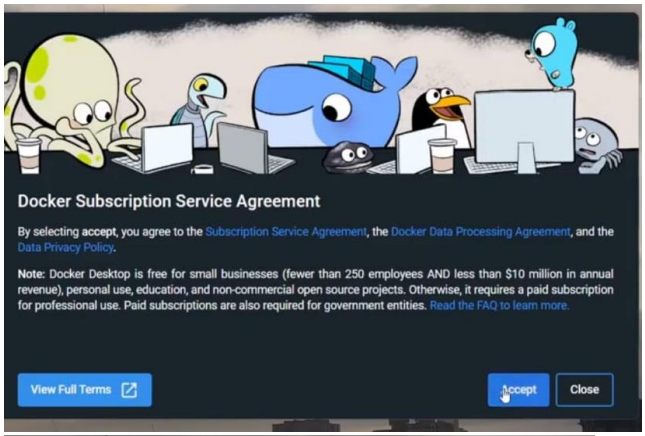
Implementation:

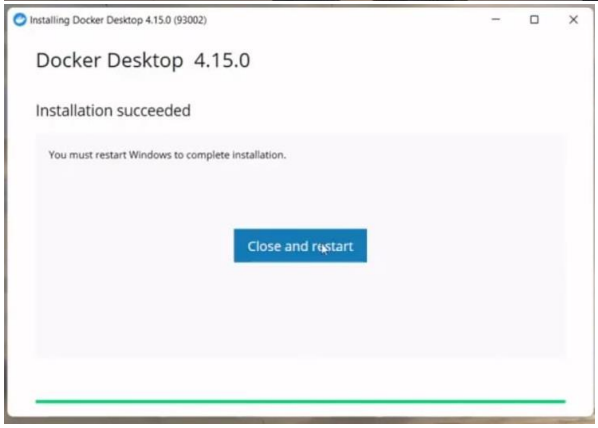
1. **Installing** Docker Desktop https://www.docker.com/products/docker-desktop/

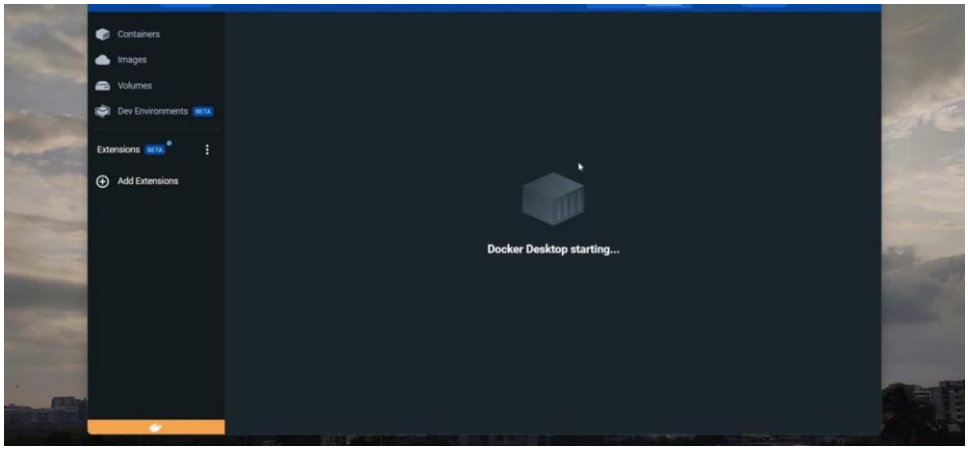
Click on Get Started and download Docker Desktop for Windows



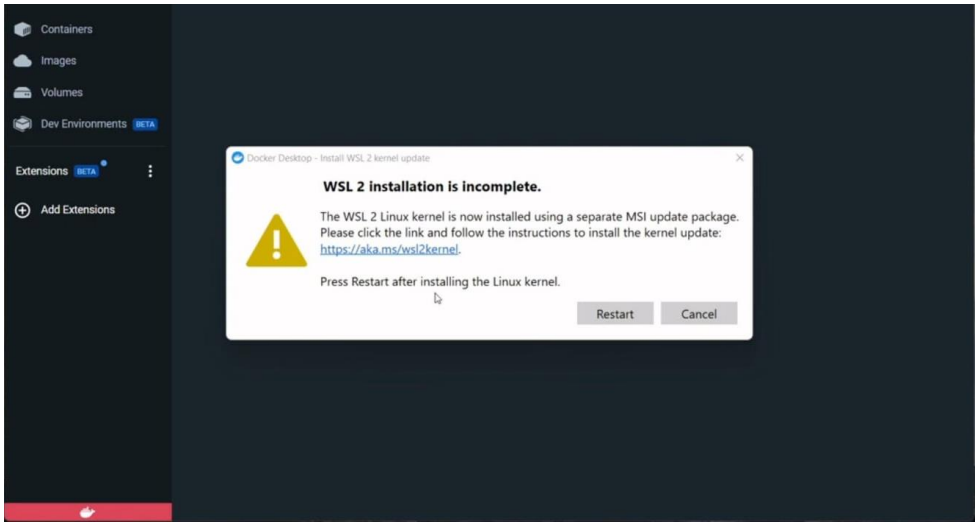
Click on Accept and complete the installation of Docker Desktop

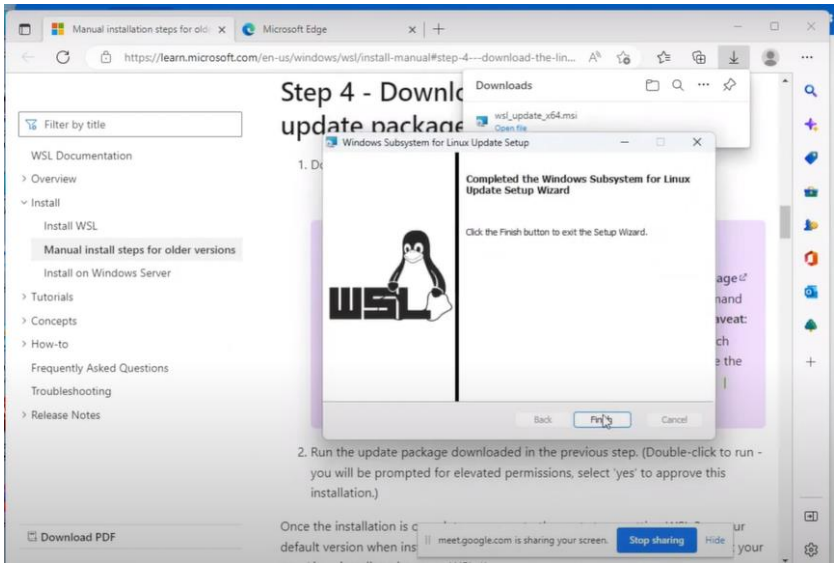






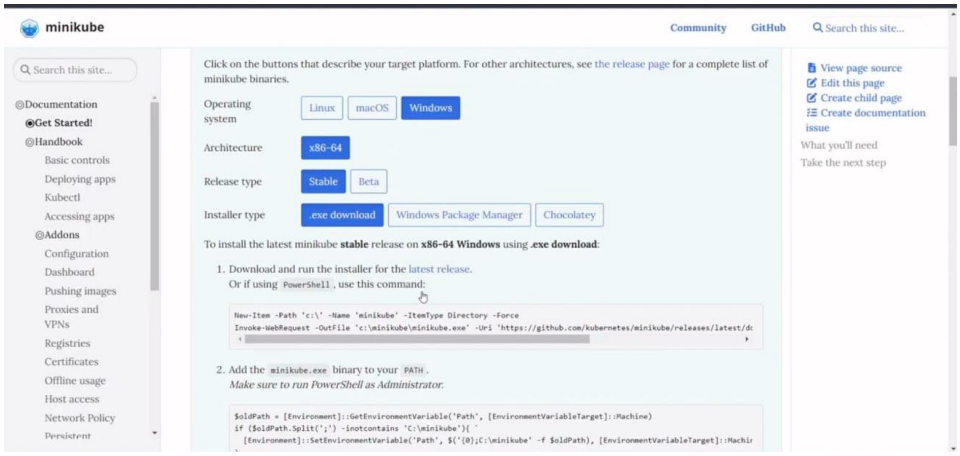
2. **Download and Install the updated version of WSL2** Click on the link and download and install the updated version





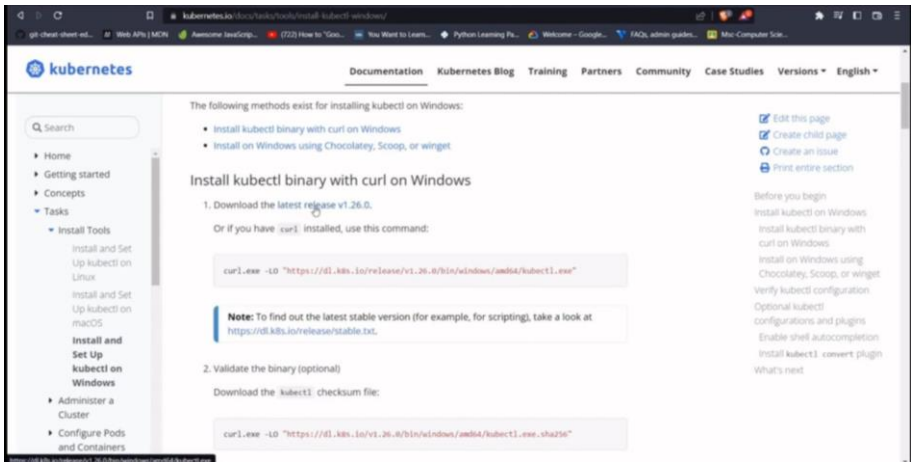
3. **Install** minikube https://minikube.sigs.k8s.io/docs/start/

Click on the .exe download to download minikube and Install minikube

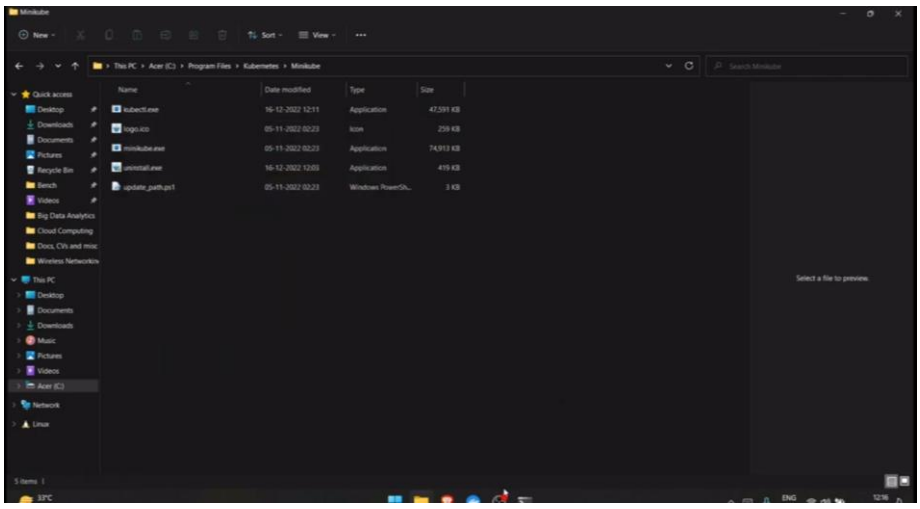


4. **Install** kubectl for windows **https://kubernetes.io/docs/tasks/tools/install-kubectl-windows/**

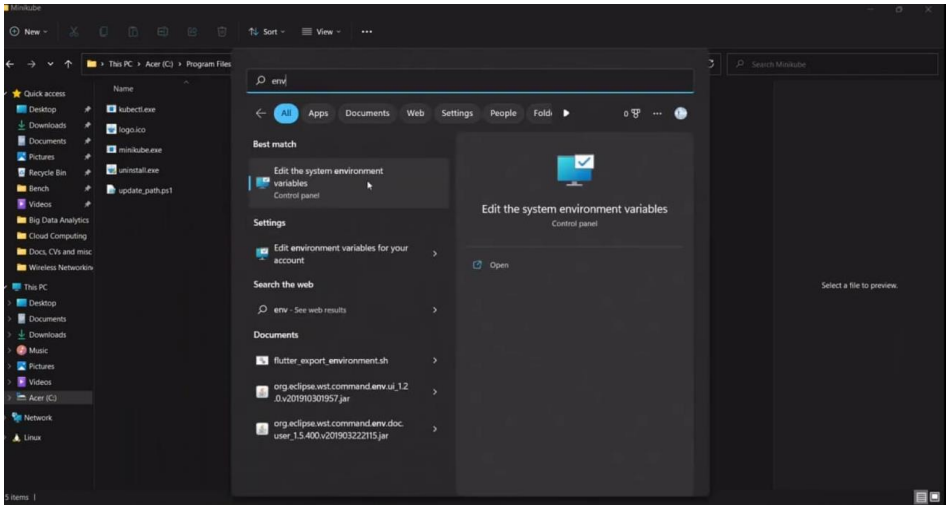
Click on the latest release and download the kubectl



Copy the kubcetl where minikube is saved and then copy the path

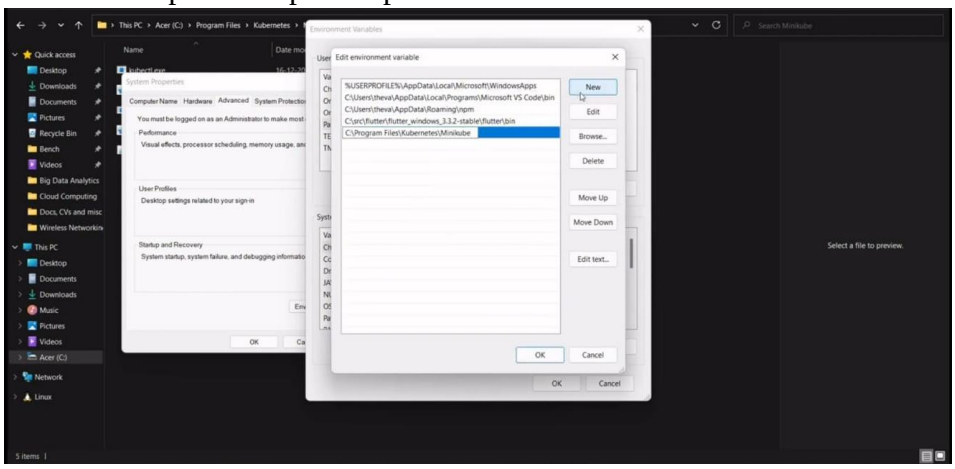


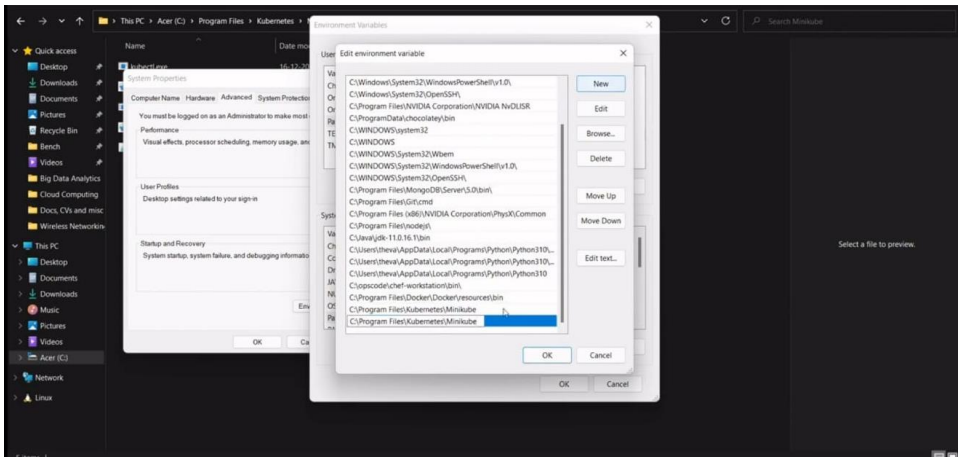
Go to environment variable:



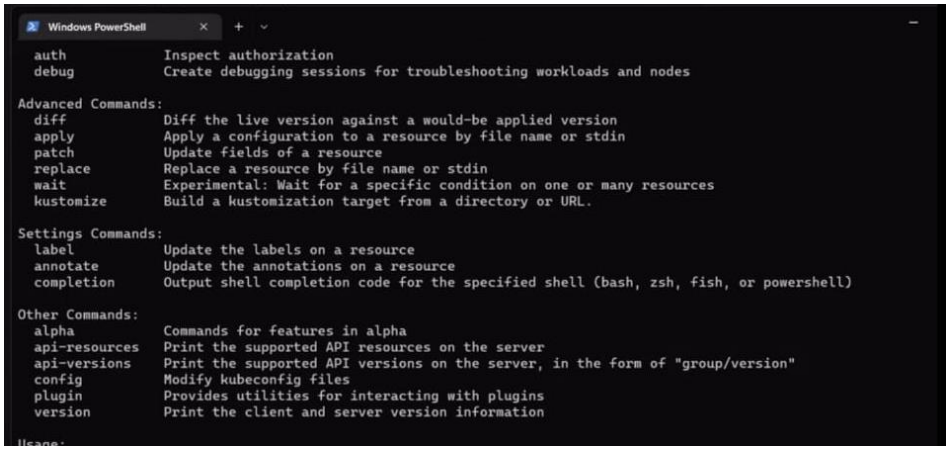
Set the path in environment variable for the user as well as the system

Click new > paste the path copied before

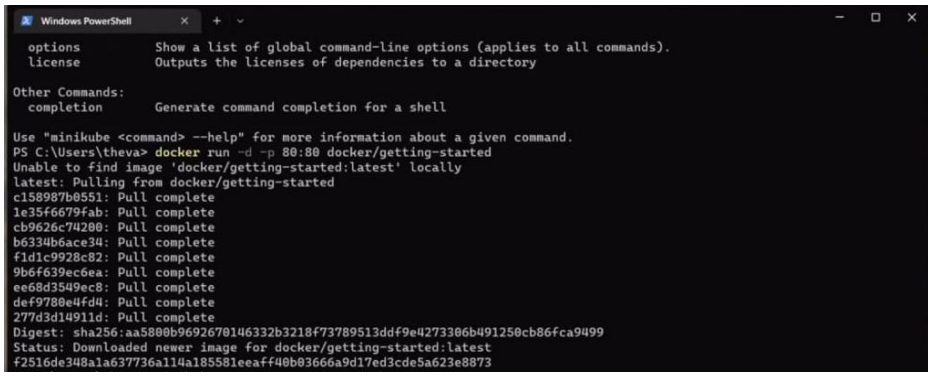




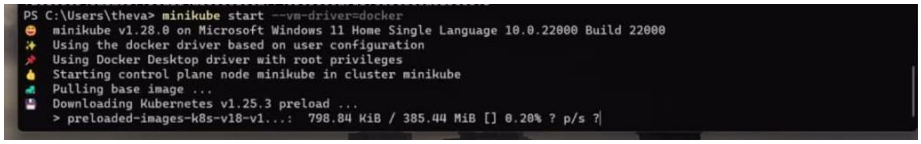
Check whether Kubectl is installed



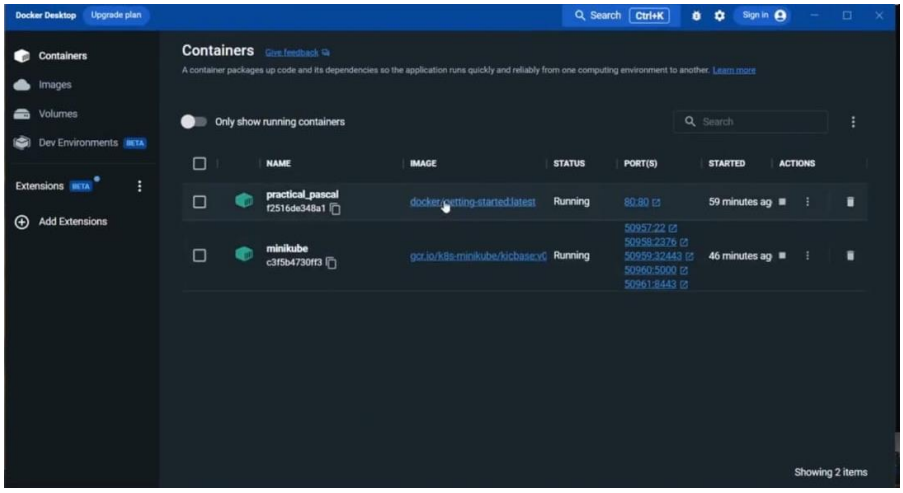
Open Docker and copy the line below and paste in the command prompt

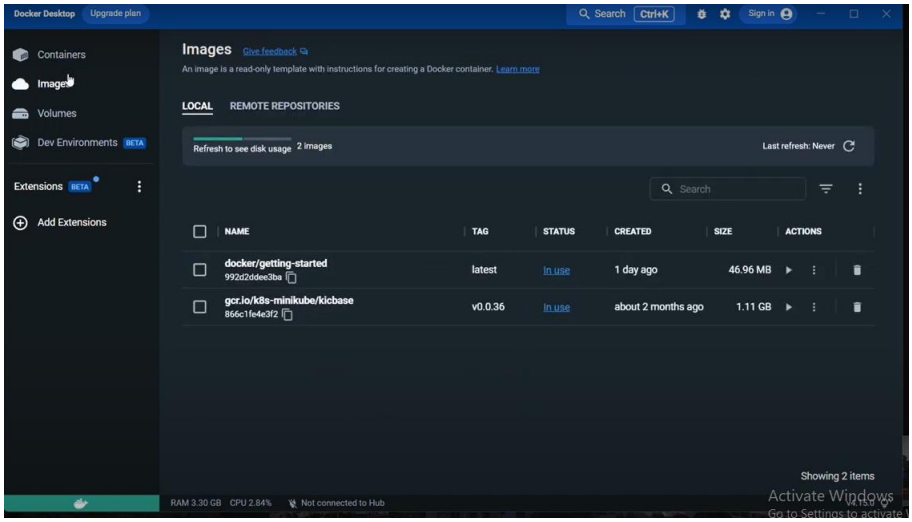


Pull the images in minikube



Check for the container status below


Docker Desktop
Upgrade plan
Search Ctrl+K
Sign in
Containers
Images
Volumes
Dev Environments BETA
Extensions BETA
Add Extensions
Containers
Give feedback
A container packages up code and its dependencies so the application runs quickly and reliably from one computing environment to another. Learn more
Only show running containers
Search
NAME
IMAGE
STATUS
PORT(S)
STARTED
ACTIONS
practical_pascal
f2516de348a1
docker/getting-started:latest
Running
80:80
59 minutes ag
minikube
c3f5b4730ff3
gcr.io/k8s-minikube/kicbase:v(
Running
50957:22
50958:2376
50959:32443
50960:5000
50961:8443
46 minutes ag
Showing 2 items

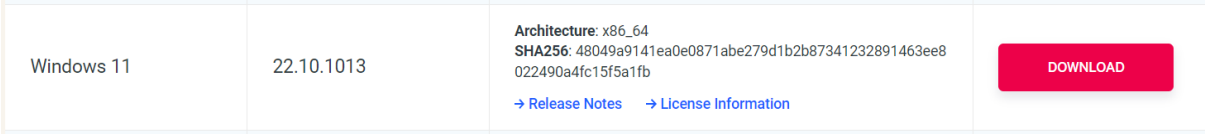

Docker Desktop
Upgrade plan
Search Ctrl+K
Sign in
Containers
Images
Volumes
Dev Environments BETA
Extensions BETA
Add Extensions
Images
Give feedback
An image is a read-only template with instructions for creating a Docker container. Learn more
LOCAL
REMOTE REPOSITORIES
Refresh to see disk usage
2 images
Last refresh: Never
Search
NAME
TAG
STATUS
CREATED
SIZE
ACTIONS
docker/getting-started
992d2ddee3ba
latest
In use
1 day ago
46.96 MB
gcr.io/k8s-minikube/kicbase
866c1fe4e3f2
v0.0.36
In use
about 2 months ago
1.11 GB
Showing 2 items
RAM 3.30 GB
CPU 2.84%
Not connected to Hub
Activate Windows
Go to Settings to activate

Practical No. 9

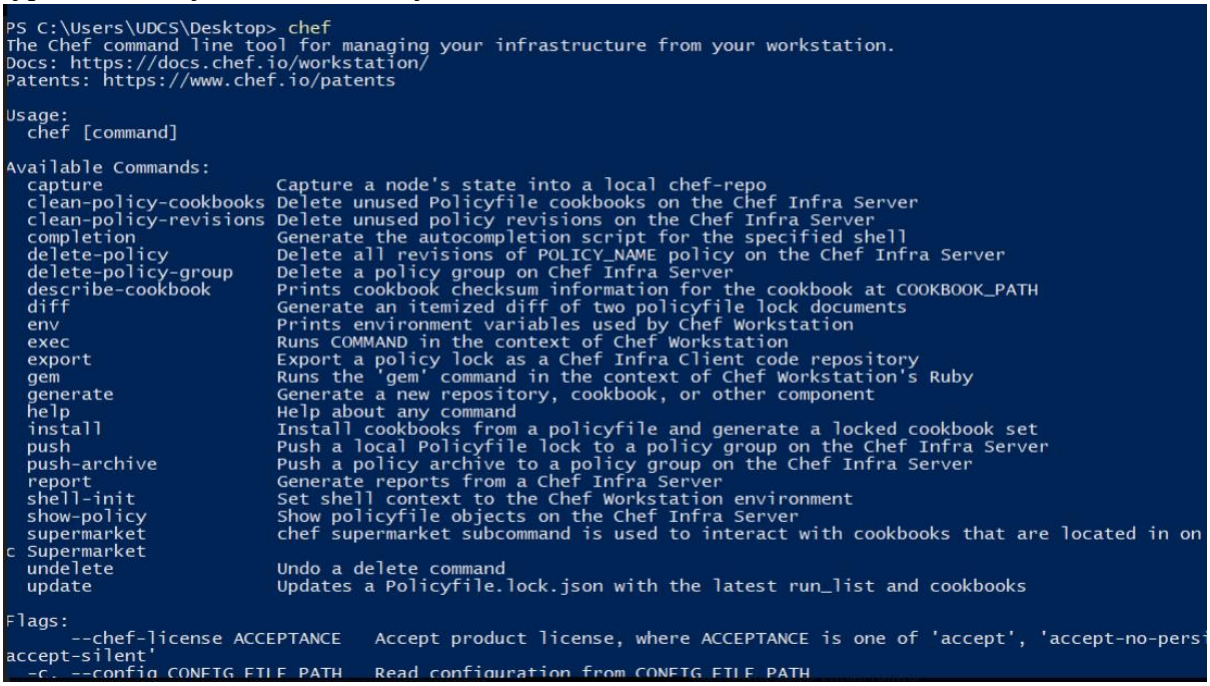
Aim: Configuring and deploying VMs/Dockers using Chef/Puppet Automation tool

Implementation:

1. To install Chef workstation, search on google for ‘chef workstation windows setup’ and select the first link
2. Select the windows downloads page and download the most stable version of the installer



3. Run the downloaded MSI installer and once finished, open the cw powershell and type ‘chef’ on your cmd to verify



4. Enter the following command
   > Chef generate cookbook admin

```
            └──__pycache__
PS C:\Users\UDCS\Desktop> chef generate cookbook admin
Generating cookbook admin
- Ensuring correct cookbook content
- Committing cookbook files to git

================================================================================
Error executing action `run` on resource 'execute[git-commit-new-files]'
================================================================================

Mixlib::ShellOut::ShellCommandFailed
------------------------------------
Expected process to exit with [0], but received '128'
---- Begin output of git commit -m "Add generated cookbook content" ----
STDOUT:
STDERR: Author identity unknown

*** Please tell me who you are.

Run

  git config --global user.email "you@example.com"
  git config --global user.name "Your Name"

to set your account's default identity.
Omit --global to set the identity only in this repository.

fatal: unable to auto-detect email address (got 'UDCS@DESKTOP-588IHIF.(none)')
---- End output of git commit -m "Add generated cookbook content" ----
Ran git commit -m "Add generated cookbook content" returned 128

Resource Declaration:
---------------------
# In C:/opscode/chef-workstation/embedded/lib/ruby/gems/3.0.0/gems/chef-cli-5.6.1/lib/chef-cli/
```

5. Type tree once done to see your file structure

```
PS C:\Users\UDCS\Desktop> tree
Folder PATH listing
Volume serial number is 1A80-EB15
C:.
├───.ipynb_checkpoints
├───AAkIF
├───crypt
├───demo
│   ├───.kitchen
│   │   └───logs
│   ├───compliance
│   │   ├───inputs
│   │   ├───profiles
│   │   └───waivers
│   ├───recipes
│   └───test
│       └───integration
│           └───default
├───image processing
├───IoT practical required libs
│   └───Arduino UNO Library for Proteus
├───Proteus.Pro.8.13.SP0.31525
├───saurabh
│   ├───.kitchen
│   │   └───logs
│   ├───compliance
│   │   ├───inputs
│   │   ├───profiles
│   │   └───waivers
│   ├───recipes
│   └───test
```

6. Now type the following commands

   >chef gem install kitchen-docker

7. Go to the directory of the admin cookbook and edit the kitchen.yml file as follows

   Driver:

   Name: docker

   Transport:

   Name: docker

Platforms:

-name: exec
Driver:
Name: exec
-name:exec

8. Now type the following commands
   >kitchen create
   >kitchen list
   >kitchen converge
   >kitchen list

```
PS C:\Users\UDCS\Desktop\admin> kitchen create
-----> Starting Test Kitchen (v3.3.2)
-----> Creating <default-exec>...
       Finished creating <default-exec> (0m0.00s).
-----> Creating <default-exec>...
       Finished creating <default-exec> (0m0.00s).
-----> Test Kitchen is finished. (0m3.23s)
PS C:\Users\UDCS\Desktop\admin> kitchen list
Instance      Driver  Provisioner  Verifier  Transport  Last Action  Last Error
default-exec  Exec    ChefInfra    Inspec    Exec       Created      <None>
default-exec  Exec    ChefInfra    Inspec    Exec       Created      <None>
PS C:\Users\UDCS\Desktop\admin> kitchen converge
-----> Starting Test Kitchen (v3.3.2)
-----> Converging <default-exec>...
```

```
PS C:\Users\theva\Desktop\prac10> kitchen list
C:/Users/theva/AppData/Local/chef/gem/ruby/3.0.0/gems/kitchen-docker-2.13.0/lib/docker/versio
nitialized constant Docker::VERSION
C:/opscode/chef-workstation/embedded/lib/ruby/gems/3.0.0/gems/docker-api-2.2.0/lib/docker/ver
s definition of VERSION was here
Instance      Driver  Provisioner  Verifier  Transport  Last Action  Last Error
default-exec  Exec    ChefInfra    Inspec    Exec       Converged    <None>
default-exec  Exec    ChefInfra    Inspec    Exec       Converged    <None>
```