MICROSERVICE ARCHITECTURE

SUBMITTED BY

RUTUJA G. KALE

2022-2023

HSNC UNIVERSITY

MASTERS OF SCIENCE IN

INFORMATION TECHNOLOGY

KISHINCHAND CHELLARAM COLLEGE

D.W.ROAD, CHURCHGATE, MUMBAI – 400 020.

Subject code – MS-FIT-2P1

MICROSERVICE ARCHITECTURE

KISHINCHAND CHELLARAM COLLEGE

CHURCHGATE, MUMBAI – 400 020.

DEPARTMENT OF INFORMATION TECHNOLOGY

M.SC. PART- I

CERTIFICATE

This is to certify that the practical done at **K.C. College** by

MR/MS. RUTUJA G. KALE

(Seat No : **KFMSCIT014**) in partial fulfilment for M.SC. (I.T.) Degree Examination has been found satisfactory. This Practical journal had not been submitted for any other examination and does not form part of any other course undergone by the candidate.

Signature	**Signature**	**Signature**
Lecturer-In-Charge	**External Examiner**	**Course Coordination**
Guided By	**Examined By**	**Certified By**

College Stamp

INDEX

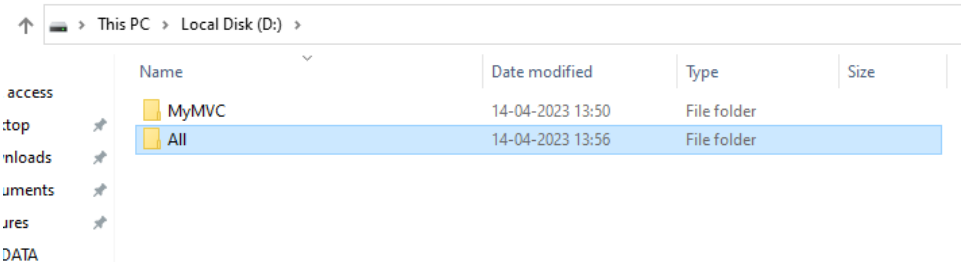
Sr. No.	Practical	Date	Sign.
1.	Building ASP.NET Core MVC Application		
2.	Building ASP. NET Core REST API		
3.	Working with docker, docker commands, docker images & Containers.		
4.	Installing s/w packages on docker.		
5.	working with docker volumes & networks.		
6.	working with CircleCI for Continuous integration		
7.	Creating Backing service with ASP.NET 2.0 core.		

PRACTICAL NO. : 01

Aim:- Building ASP.NET Core MVC Application

1)Install .Net Core Sdk

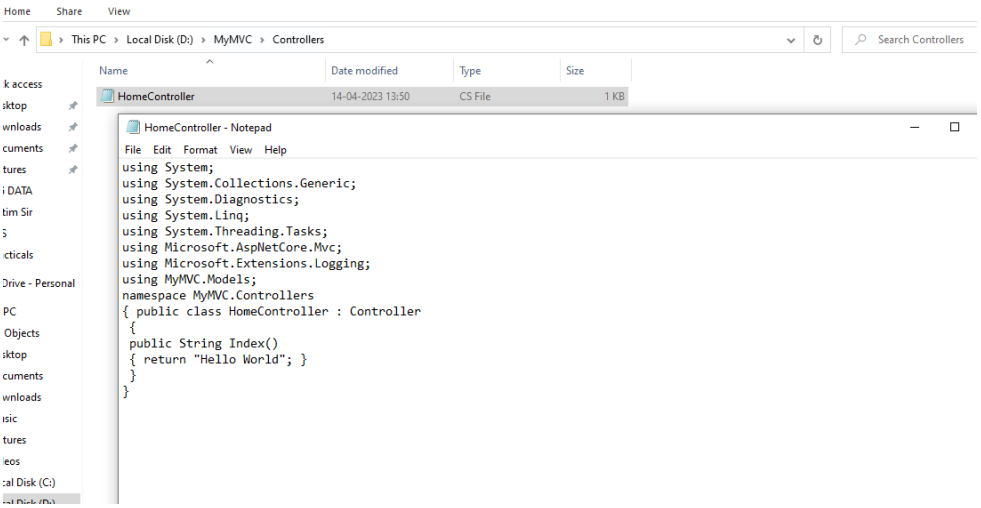
2)create folder MyMVC folder in D: drive or any other drive



3)open command prompt and perform following operations Command: to create mvc project dotnet new mvc --auth none

```
C:\Users\HP> D:

D:\>cd MyMVC

D:\MyMVC>dotnet new mvc --auth none
The template "ASP.NET Core Web App (Model-View-Controller)" was created s
This template contains technologies from parties other than Microsoft, se

Processing post-creation actions...
Restoring D:\MyMVC\MyMVC.csproj:
  Determining projects to restore...
  Restored D:\MyMVC\MyMVC.csproj (in 176 ms).
Restore succeeded.
```

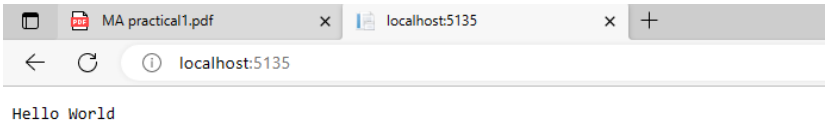
4) Go to controllers folder and modify HomeController.cs file to match following:



5) Run the project

```
D:\MyMVC>dotnet run
Building...
info: Microsoft.Hosting.Lifetime[14]
      Now listening on: http://localhost:5135
info: Microsoft.Hosting.Lifetime[0]
      Application started. Press Ctrl+C to shut down.
info: Microsoft.Hosting.Lifetime[0]
      Hosting environment: Development
info: Microsoft.Hosting.Lifetime[0]
      Content root path: D:\MyMVC
info: Microsoft.Hosting.Lifetime[0]
      Application is shutting down...
```

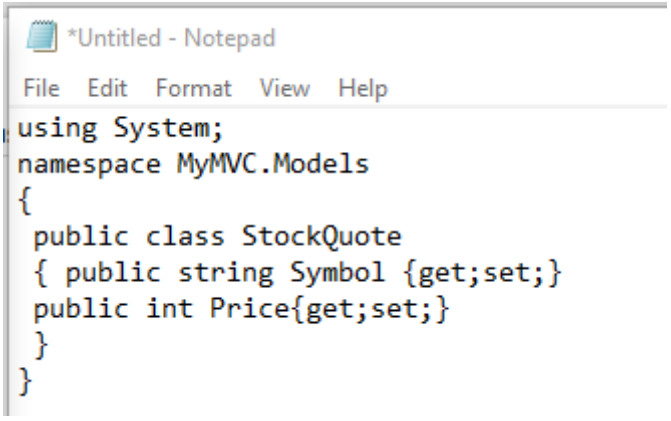
Now open browser and and type URL: localhost:5135

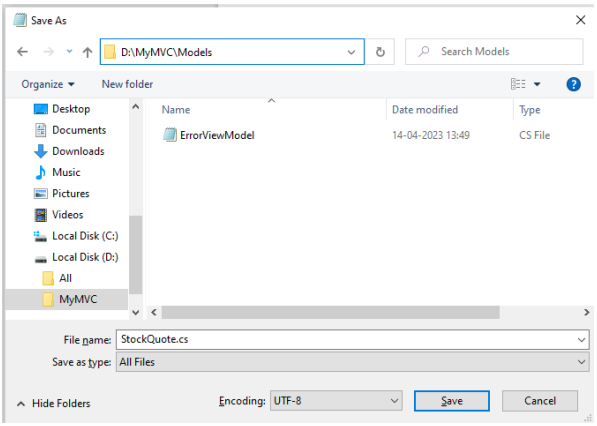


6) Now go back to command prompt and stop running project using CTRL+C

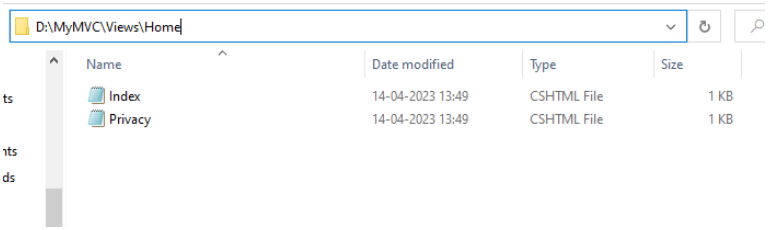
```
D:\mymvc>dotnet run
Building...
info: Microsoft.Hosting.Lifetime[14]
      Now listening on: http://localhost:5049
info: Microsoft.Hosting.Lifetime[0]
      Application started. Press Ctrl+C to shut down.
info: Microsoft.Hosting.Lifetime[0]
      Hosting environment: Development
info: Microsoft.Hosting.Lifetime[0]
      Content root path: D:\mymvc
warn: Microsoft.AspNetCore.HttpsPolicy.HttpsRedirectionMiddleware[3]
      Failed to determine the https port for redirect.
```

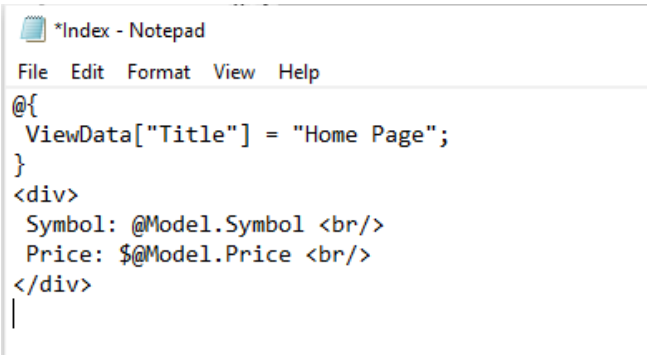
7)Go to models folder and add new file StockQuote.cs to it with following content



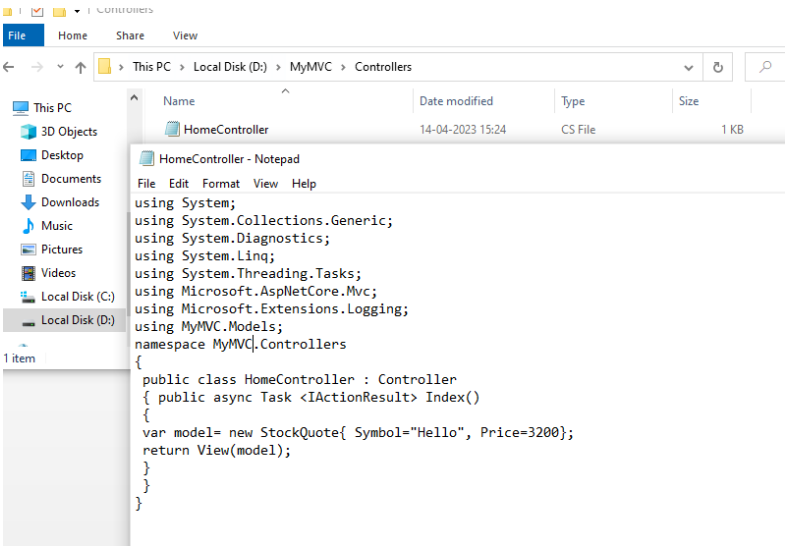


8)Now Add View to folder then home folder in it and modify index.cshtml file to match following

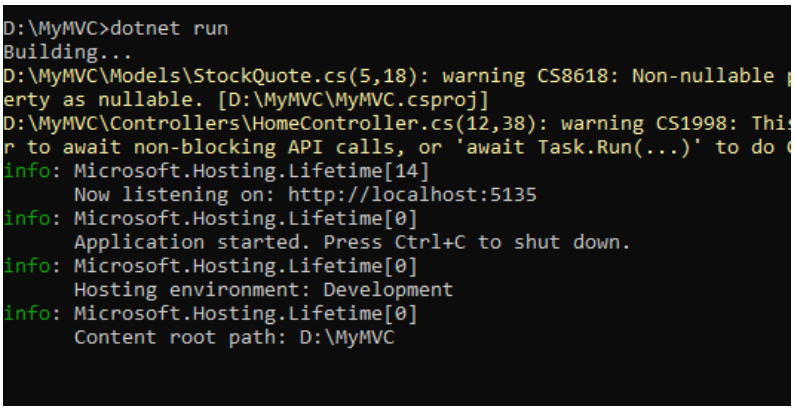




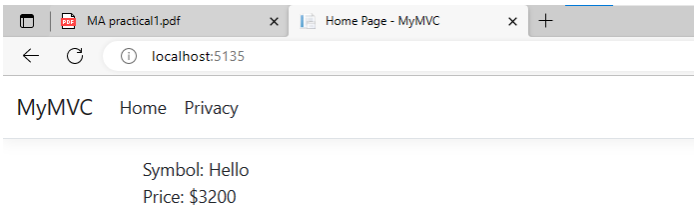
9)Now modify HomeController.cs file to match following:



10) Now run the project using



11)Now go back to browser and refresh to get modified view response



PRACTICAL NO. : 02

Aim:- Building ASP. NET Core REST API

1. Download and install dotnet
2. Check everything installed correctly

```
D:\MyMVC>dotnet

Usage: dotnet [options]
Usage: dotnet [path-to-application]

Options:
  -h|--help         Display help.
  --info            Display .NET information.
  --list-sdks       Display the installed SDKs.
  --list-runtimes   Display the installed runtimes.

path-to-application:
  The path to an application .dll file to execute.

D:\MyMVC>
```

Create your web API

1. Open two command prompts Command prompt 1: Command: dotnet new webapi -o Glossary

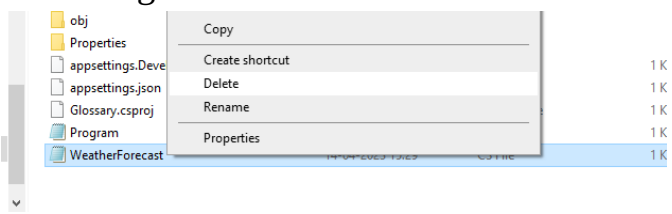
```
D:\MyMVC>cd..

D:\>dotnet new webapi -o Glossary
The template "ASP.NET Core Web API" was created successfully.

Processing post-creation actions...
Restoring D:\Glossary\Glossary.csproj:
  Determining projects to restore...
  Restored D:\Glossary\Glossary.csproj (in 435 ms).
Restore succeeded.



D:\>
```

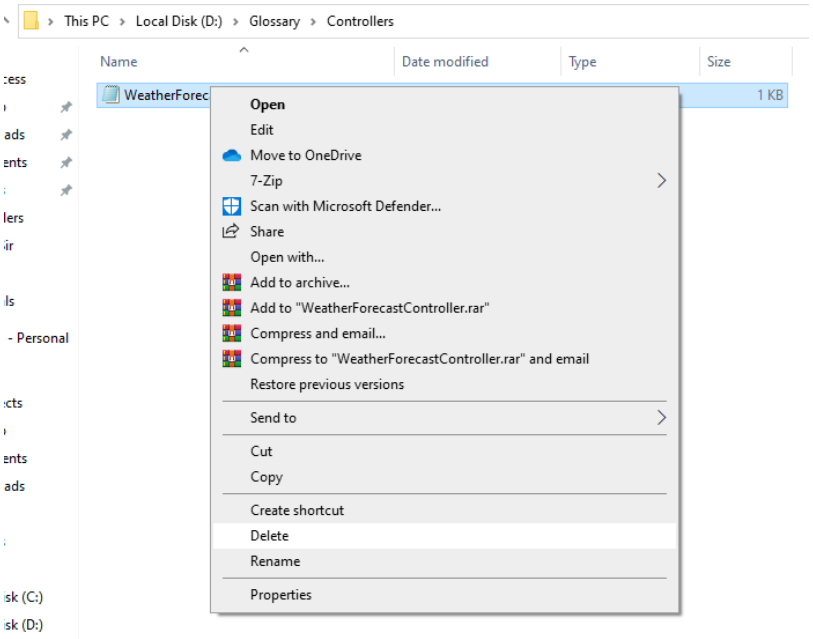
```
D:\>cd Glossary

D:\Glossary>dotnet run
Building...
info: Microsoft.Hosting.Lifetime[14]
      Now listening on: http://localhost:5176
info: Microsoft.Hosting.Lifetime[0]
      Application started. Press Ctrl+C to shut down.
info: Microsoft.Hosting.Lifetime[0]
      Hosting environment: Development
info: Microsoft.Hosting.Lifetime[0]
      Content root path: D:\Glossary
```

2. Command Prompt 2: (try running ready made weatherforecast class for testing) Command: curl --insecure https://localhost:5001/weatherforecast

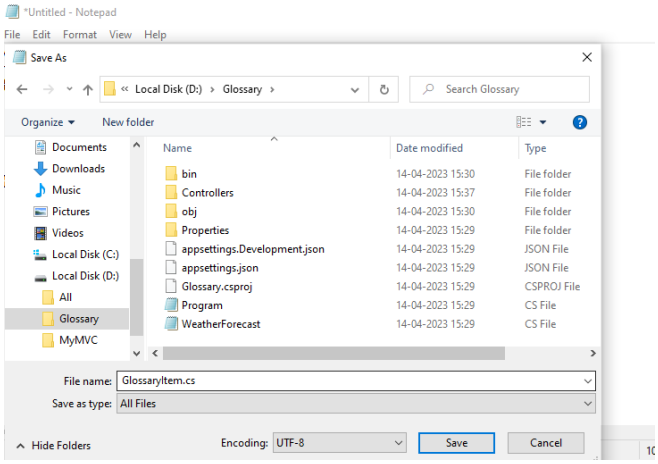
```
D:\>curl --insecure http://localhost:5176/weatherforecast
[{"date":"2023-04-15","temperatureC":-11,"temperatureF":13,"summary":"Balmy"},{"date":"2023-04-16","te
mperatureF":57,"summary":"Hot"},{"date":"2023-04-17","temperatureC":-11,"temperatureF":13,"summary":"M
3-04-18","temperatureC":-15,"temperatureF":6,"summary":"Cool"},{"date":"2023-04-19","temperatureC":1,"
summary":"Freezing"}]
D:\>
```

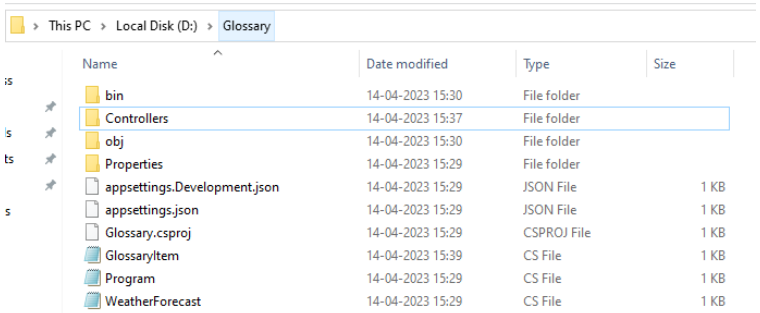
3. Now Change the content: To get started, remove the WeatherForecast.cs file from the root of the project and the WeatherForecastController.cs file from the Controllers folder. Add Following two files



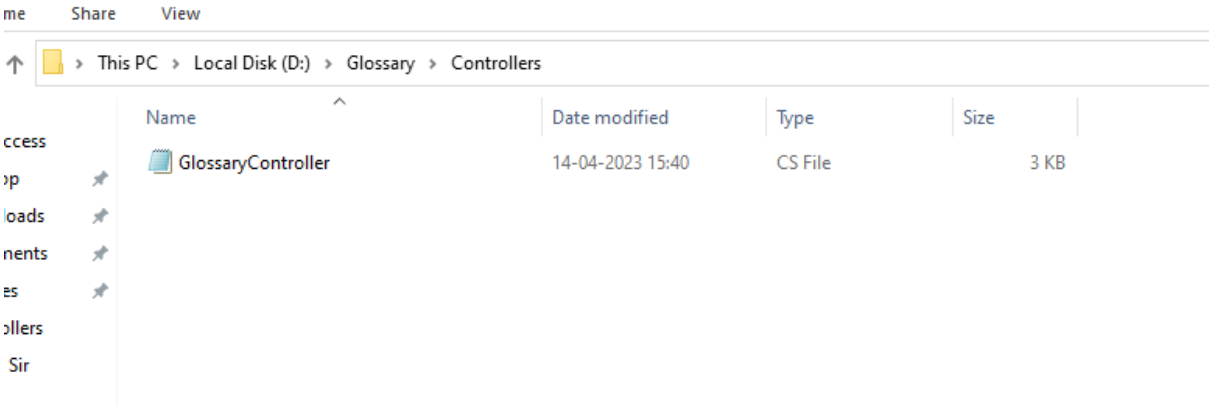


1) D:\Glossary\GlossaryItem.cs (type it in notepad and save as all files)





2) D:\Glossary\Controllers\ GlossaryController.cs (type it in notepad and save as all files)



3.Now stop running previous dotnet run on command prompt 1 using Ctrl+C. and Run it again for new code. On Command prompt1:

```
D:\Glossary>dotnet run
Building...
D:\Glossary\GlossaryItem.cs(6,16): warning CS8618: Non-nullable proper
ing constructor. Consider declaring the property as nullable. [D:\Glos
D:\Glossary\GlossaryItem.cs(7,16): warning CS8618: Non-nullable proper
n exiting constructor. Consider declaring the property as nullable. [D
D:\Glossary\Controllers\GlossaryController.cs(58,58): warning SYSLIB00
ri.EscapeUriString can corrupt the Uri string in some cases. Consider
nents instead.' [D:\Glossary\Glossary.csproj]
info: Microsoft.Hosting.Lifetime[14]
      Now listening on: http://localhost:5176
info: Microsoft.Hosting.Lifetime[0]
      Application started. Press Ctrl+C to shut down.
info: Microsoft.Hosting.Lifetime[0]
      Hosting environment: Development
info: Microsoft.Hosting.Lifetime[0]
      Content root path: D:\Glossary
```

On Command prompt2:

1) Getting a list of items: Command: curl --insecure http://localhost:5176/api/glossary

```
D:\>curl --insecure http://localhost:5176/api/glossary
[{"term":"HTML","definition":"Hypertext Markup Language"},{"term":"MVC
penID","definition":"An open standard for authentication"}]
D:\>
```

2) Getting a single item Command: curl --insecure ttp://localhost:5176/api/glossary/MVC

```
D:\>curl --insecure http://localhost:5176/api/glossary/MVC
{"term":"MVC","definition":"Model View Controller"}
D:\>
```

3) Creating an item Command: curl --insecure -X POST -d "{\"term\": \"MFA\", "definition\":\"An uthentication process.\"}" -H "ContentType:application/json" http://localhost:5176/api/glossary

```
D:\>curl --insecure -X POST -d "{\"term\": \"MFA\", \"definition\":\"An authentication process.\"}" -H "Content-Type:application/json" http://localhost:5176/api/glossar
y
{"term":"MFA","definition":"An authentication process."}
D:\>
```

4)Update Item

Command:

curl --insecure -X PUT -d "{\"term\": \"MVC\", \"definition\":\"Modified record of Model View Controller.\"}" -H "Content-Type:application/json" http://localhost:5176/api/glossary

```
D:\>curl --insecure -X PUT -d "{\"term\": \"MVC\", \"definition\":\"Modified record of Model View Controller.\"}" -H "C
ntent-Type:application/json" http://localhost:5176/api/glossary

D:\>curl --insecure http://localhost:5176/api/glossary
[{"term":"HTML","definition":"Hypertext Markup Language"},{"term":"MVC","definition":"Modified record of Model View Con
roller."},{"term":"OpenID","definition":"An open standard for authentication"},{"term":"MFA","definition":"An authentic
tion process."}]
D:\>S
```

5) Delete Item:

curl --insecure --request DELETE --url http://localhost:5176/api/glossary/openid

```
D:\>curl --insecure --request DELETE --url http://localhost:5176/api/glossary/openid

D:\>curl --insecure http://localhost:5176/api/glossary
[{"term":"HTML","definition":"Hypertext Markup Language"},{"term":"MVC","definition":"Modified record of Model View Cont
roller."},{"term":"MFA","definition":"An authentication process."}]
D:\>
```

Practical No.:03

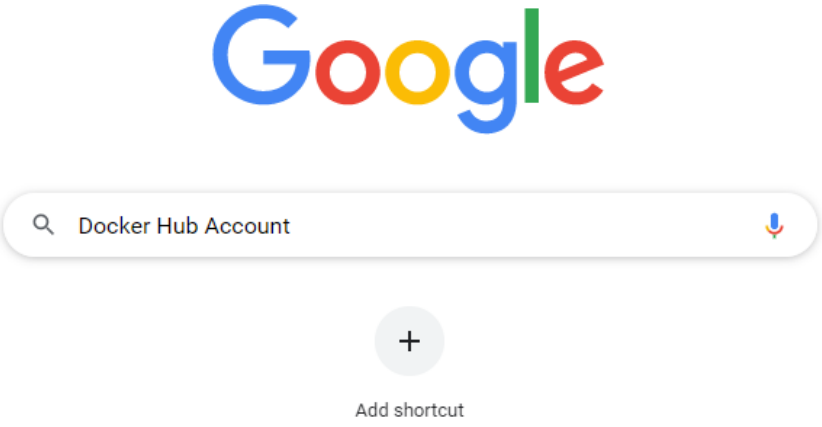
Aim:- Working with docker, docker commands, docker images & Containers.

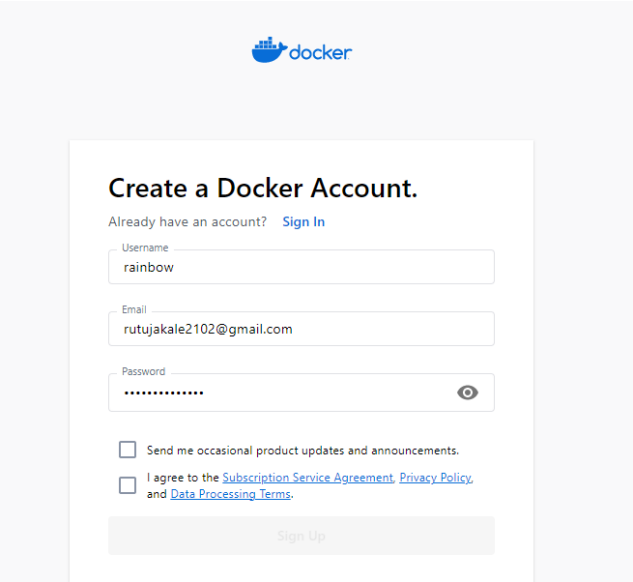
Step 1: Create Docker Hub Account from https://hub.docker.com/

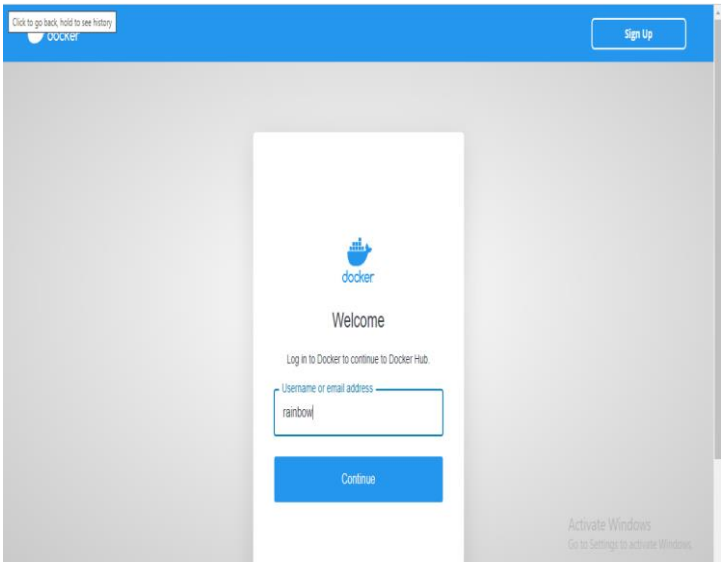
Step 2: Navigate to Play with Docker from https://labs.play-with-docker.com/

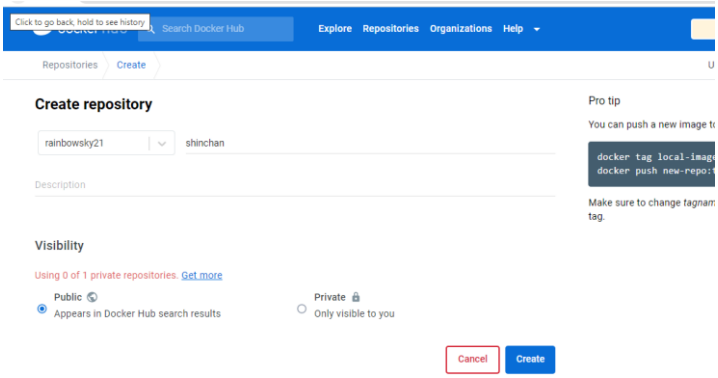
Step 3: Click on create Instance and enter the following command.

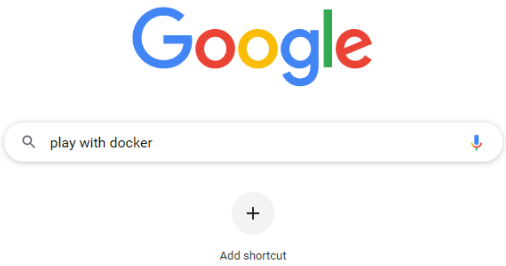
a) docker -v
b) docker –version

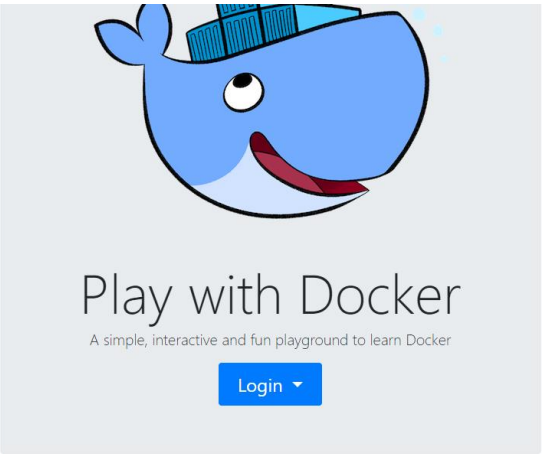












```
#                                                                    #
# The PWD team.                                                      #
######################################################################
[node1] (local) root@192.168.0.18 ~
$ docker -v
Docker version 20.10.17, build 100c701
[node1] (local) root@192.168.0.18 ~
$ docker pull rocker/verse
Using default tag: latest
latest: Pulling from rocker/verse
76769433fd8a: Pull complete
04aaad001c33: Pull complete
b993023d0f01: Pull complete
90f95b47708a: Pull complete
de634d6eeac2: Pull complete
66480f89de4f: Pull complete
38b8432a63be: Pull complete
```

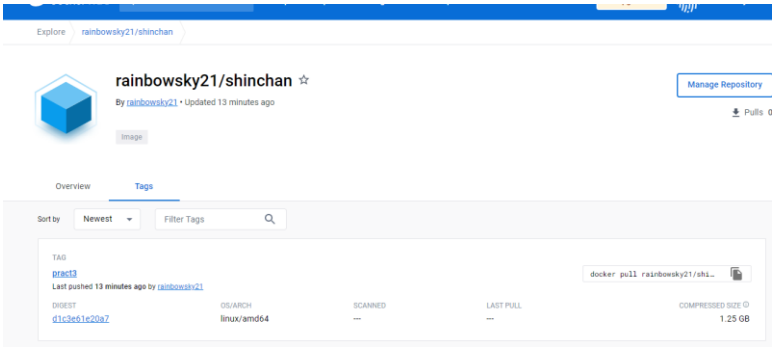
```
[node1] (local) root@192.168.0.18 ~
$ docker images
REPOSITORY      TAG       IMAGE ID       CREATED        SIZE
rocker/verse    latest    551e1a37de34   46 hours ago   3.43GB
[node1] (local) root@192.168.0.18 ~
$
```

```
$ docker login --username=rainbowsky21
Password:
WARNING! Your password will be stored unencrypted in /root/.docker/config.json.
Configure a credential helper to remove this warning. See
https://docs.docker.com/engine/reference/commandline/login/#credentials-store

Login Succeeded
[node1] (local) root@192.168.0.18 ~
$
```

```
[node1] (local) root@192.168.0.18 ~
$ docker images
REPOSITORY      TAG       IMAGE ID       CREATED        SIZE
rocker/verse    latest    551e1a37de34   46 hours ago   3.43GB
[node1] (local) root@192.168.0.18 ~
$ docker login --username=rainbowsky21
Password:
WARNING! Your password will be stored unencrypted in /root/.docker/config.json.
Configure a credential helper to remove this warning. See
https://docs.docker.com/engine/reference/commandline/login/#credentials-store

Login Succeeded
[node1] (local) root@192.168.0.18 ~
$ docker tag 551e1a37de34 rainbowsky21/shinchan
```

```
[node1] (local) root@192.168.0.18 ~
$ docker tag 551e1a37de34 rainbowsky21/shinchan:pract3
```

```
[node1] (local) root@192.168.0.18 ~
$ docker push rainbowsky21/shinchan:pract3
The push refers to repository [docker.io/rainbowsky21/shinchan]
6c1711f305ff: Pushed
54cc7e366446: Pushed
1e82ee1f79d4: Pushed
e4f6f141a475: Pushed
94644a51ea10: Pushed
99e44ef3e8e9: Pushed
fa35739b43d8: Pushed
a0f5608ee4a8: Pushed
e7484d5519b7: Pushed
202fe64c3ce3: Pushed
pract3: digest: sha256:d1c3e61e20a780eaadd48b8b214b4646dc343a60847577ed1220cc
[node1] (local) root@192.168.0.18 ~
```



3B) Build your own image file from docker file & push & pull & run the file.

Step 1: Create Docker file.

```
[node1] (local) root@192.168.0.8 ~
$ docker login --username=rainbowsky21
Password:
WARNING! Your password will be stored unencrypted in /root/.docker/config.json.
Configure a credential helper to remove this warning. See
https://docs.docker.com/engine/reference/commandline/login/#credentials-store

Login Succeeded
[node1] (local) root@192.168.0.8 ~
$ docker images
```

```
Login Succeeded
[node1] (local) root@192.168.0.8 ~
$ docker images
REPOSITORY      TAG       IMAGE ID       CREATED      SIZE
rocker/verse    latest    551e1a37de34   2 days ago   3.43GB
[node1] (local) root@192.168.0.8 ~
$ cat > dockerfile <<EOF
> FROM busybox
> CMD echo "Hello"
> EOF
```

Step 2: Enter command to build the ‘dockerfile’ created in step 1: docker build .\

```
[node1] (local) root@192.168.0.8 ~
$ docker build ./
Sending build context to Docker daemon   12.8kB
Step 1/2 : FROM busybox
latest: Pulling from library/busybox
205dae5015e7: Pull complete
Digest: sha256:7b3ccabffc97de872a30dfd234fd972a66d247c8cfc69b0550f276481852627c
Status: Downloaded newer image for busybox:latest
 ---> 66ba00ad3de8
Step 2/2 : CMD echo "Hello"
 ---> Running in b084668cc824
Removing intermediate container b084668cc824
 ---> 72d27fc1ec76
Successfully built 72d27fc1ec76
```

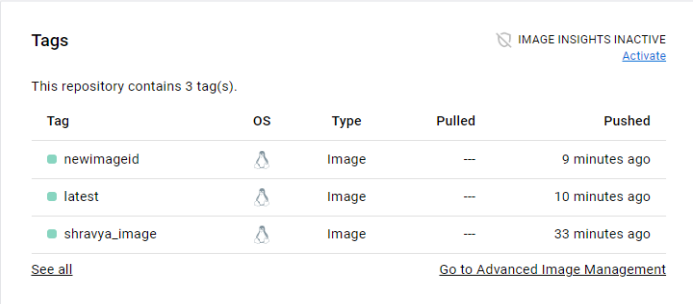
Step 3: List all the images from the docker: docker images

Step 4: docker run -p 80:80 imageid

```
[node1] (local) root@192.168.0.8 ~
$ docker images
REPOSITORY      TAG       IMAGE ID       CREATED          SIZE
<none>          <none>    72d27fc1ec76   11 seconds ago   4.87MB
rocker/verse    latest    551e1a37de34   2 days ago       3.43GB
busybox         latest    66ba00ad3de8   8 weeks ago      4.87MB
[node1] (local) root@192.168.0.8 ~
$ docker run -p 80:80 72d27fc1ec76
Hello
```

Step 5: Push the image to docker.

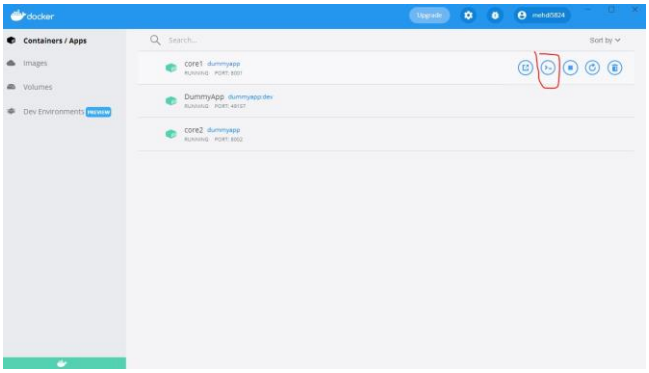
Step 6: Navigate to Docker repository and check the Image in Tags section



Practical No.:04

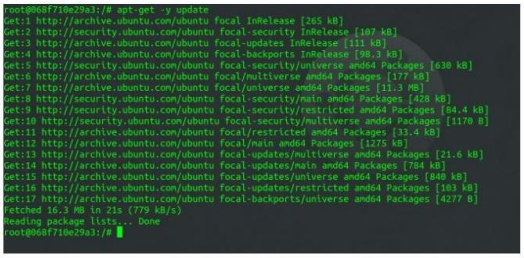
Aim:- Installing s/w packages on docker.

Step 1 – Go to CLI Option on the container in Docker Desktop



Step 2: Now, you have opened the bash of your Ubuntu Docker Container. To install any packages, you first need to update the OS.

```
apt-get -y update
```



Updating the Container

Step 3: After you have updated the Docker Container, you can now install the Firefox and Vim packages inside it.

```
apt-get -y install firefox
apt-get -y install vim
```

Installing Firefox

Installing Vim

You can now easily use these packages through the bash itself.

Step 4: Run vim to verify if the software package has been installed

Container volumes

With the previous experiment, we saw that each container starts from the image definition each time it starts. While containers can create, update, and delete files, those changes are lost when the container is removed and all changes are isolated to that container. With volumes, we can change all of this.

Volumes provide the ability to connect specific filesystem paths of the container back to the host machine. If a directory in the container is mounted, changes in that directory are also seen on the host machine. If we mount that same directory across container restarts, we'd see the same files.

Step 1

Working with Docker Volumes explained below:-

a) Let us create the volume first. For the reference we will type below command:-

➔ docker volume

```
F:\Microservices\getting-started-master\app>docker volume

Usage:  docker volume COMMAND

Manage volumes

Commands:
  create      Create a volume
  inspect     Display detailed information on one or more volumes
  ls          List volumes
  prune       Remove all unused local volumes
  rm          Remove one or more volumes

Run 'docker volume COMMAND --help' for more information on a command.

F:\Microservices\getting-started-master\app>
```

b) Now lets create the actual volume:-

➔ docker volume create myvol1

```
F:\Microservices\getting-started-master\app>docker volume create myvol1
myvol1

F:\Microservices\getting-started-master\app>
```

As you can see here our volume is created.

c) To list the volume we will write below command:-

```
F:\Microservices\getting-started-master\app>docker volume ls
DRIVER    VOLUME NAME
local     aba83257ee43df3f86bfea2b09c1d1ffe5a59b9ced82c6b7ea5f458e9e298e72
local     d247fdb49990ed914b54fffe365671c1f3b773d5871038817e18d36cf6e288bf
local     myvol1

F:\Microservices\getting-started-master\app>
```

d) To get the details of our volume we have to write below command:-

➔ docker volume inspect myvol1

```
F:\Microservices\getting-started-master\app>docker volume inspect myvol1
[
    {
        "CreatedAt": "2021-05-10T06:36:15Z",
        "Driver": "local",
        "Labels": {},
        "Mountpoint": "/var/lib/docker/volumes/myvol1/_data",
        "Name": "myvol1",
        "Options": {},
        "Scope": "local"
    }
]

F:\Microservices\getting-started-master\app>
```

Here you can see all the details of our myvol1 i.e. name, created time, driver, mountpoint.

Our volume is located at the path mentioned in Mountpoint section.

e) To remove your volume you can write below command:-

➔ docker volume rm myvol1

To remove all unused volumes we can write below command

➔ docker volume prune

These are the basic functionalities of docker volume. You can explore more functionalities as well.

Working with docker network explained below:-

To write this command below is the syntax:-

➔ `docker network COMMAND`

a) To Connect a container to a network

➔ docker network connect

b) To create a network we have to write below command:-

➔ docker network create

c) To disconnect a container from a network

➔ docker network disconnect

d) To display detailed information on one or more networks

➔ docker network inspect

e) To list the network:-

➔ docker network ls

f) To remove all unused networks

➔ docker network prune

g) To remove one or more networks

➔ docker network rm

Practical 05

Aim: Working with Docker Volumes and Networks.

Pre-Requisites:

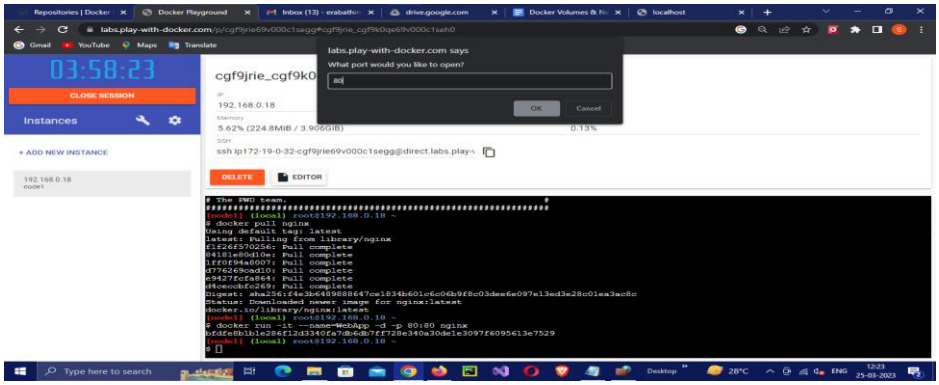
1. Open Windows Firewall
2. Click on Advanced Security
3. Click on Inbound Rules
4. Create a New Rule
    a. Which type of rule would you like to create → port
    b. Does this rule apply the local ports or specific local ports
    c. Select Specific local ports - 80
    d. What action should be taken when a connection matches the specified conditions? - Allow the connection
    e. When does this apply? – Domain, Private, Public
    f. Name: ReportServer
    g. Description: ReportServer

Step 1: Enter the following Commands

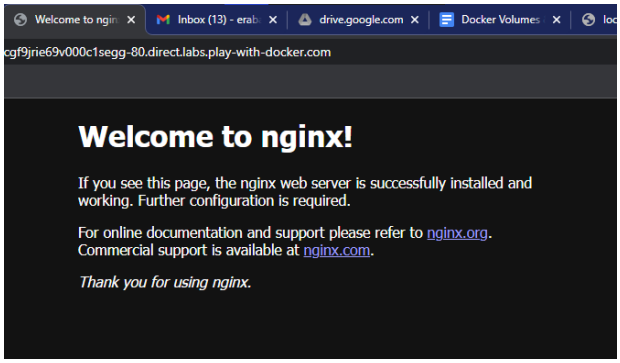
a) docker pull nginx – nginx : nginx is the image which is already available in docker
b) docker run -it –name=webApp -d -p 80:80 nginx: Create a webapp and run it with nginx image on port 80

```
# The PWD team.                                                    #
####################################################################
[node1] (local) root@192.168.0.18 ~
$ docker pull nginx
Using default tag: latest
latest: Pulling from library/nginx
f1f26f570256: Pull complete
84181e80d10e: Pull complete
1ff0f94a8007: Pull complete
d776269cad10: Pull complete
e9427fcfa864: Pull complete
d4ceccbfc269: Pull complete
Digest: sha256:f4e3b6489888647ce1834b601c6c06b9f8c03dee6e097e13ed3e28c01ea3ac8c
Status: Downloaded newer image for nginx:latest
docker.io/library/nginx:latest
[node1] (local) root@192.168.0.18 ~
$ docker run -it --name=WebApp -d -p 80:80 nginx
bfdfe8b1b1e286f12d3340fa7db6db7ff728e340a30de1e3097f6095613e7529
[node1] (local) root@192.168.0.18 ~
$
```

Step 2: Click on Port and enter 80 in the dropdown window and click OK.



Output: The below webpage will be visible



Step 3: Enter the below command in order to enter into bash shell and then open port 80.

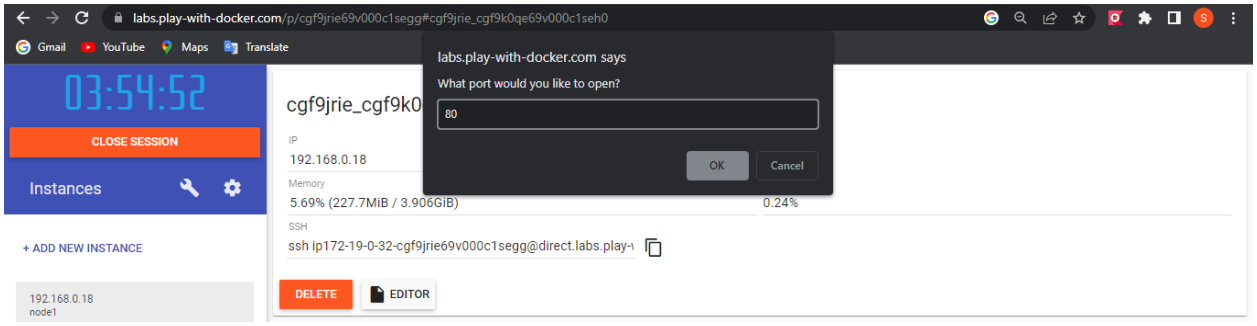
docker exec -it WebApp bash

Cd /usr/share/nginx/html

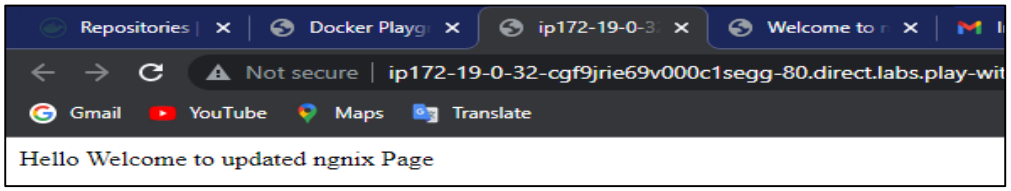
Echo “Hello Welcome to updated nginx Page.”> index.html

exit

```
[node1] (local) root@192.168.0.18 ~
$ docker exec -it WebApp bash
root@bfdfe8b1b1e2:/# cd /usr/share/ngnix/html
bash: cd: /usr/share/ngnix/html: No such file or directory
root@bfdfe8b1b1e2:/# cd /usr/share/nginx/html
root@bfdfe8b1b1e2:/usr/share/nginx/html# echo "Hello Welcome to updated ngnix Page" > index.html
root@bfdfe8b1b1e2:/usr/share/nginx/html# exit
exit
```



Output: The below webpage will be visible



Step 4: List all the running containers: docker ps

```
[node1] (local) root@192.168.0.18 ~
$ docker ps
CONTAINER ID   IMAGE     COMMAND                  CREATED         STATUS         PORTS                   NAMES
bfdfe8b1b1e2   nginx     "/docker-entrypoint.…"   4 minutes ago   Up 4 minutes   0.0.0.0:80->80/tcp      WebApp
[node1] (local) root@192.168.0.18 ~
$ docker run -it --name=WebApp1 -d -p
flag needs an argument: 'p' in -p
See 'docker run --help'.
[node1] (local) root@192.168.0.18 ~
$ docker stop webApp
Error response from daemon: No such container: webApp
[node1] (local) root@192.168.0.18 ~
$ docker stop WebApp
WebApp
```

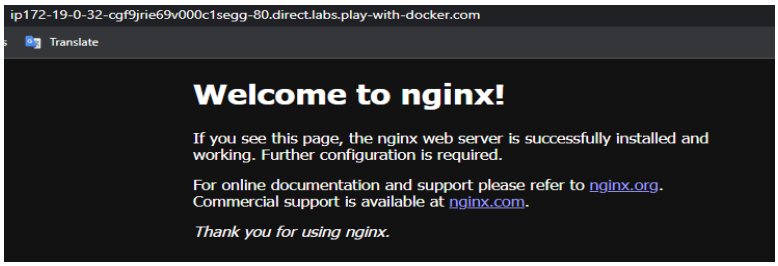
Step 5: Create another container in Docker: WebApp1

Docker run -it –name=WebApp1 -d -p 80:80 nginx:

```
[node1] (local) root@192.168.0.18 ~
$ docker run -it --name=WebApp1 -d -p 80:80 nginx
730d55e312d235cc564d53d21a5cdc33145329d2025bfb8a2f834784e9c19d57
[node1] (local) root@192.168.0.18 ~
```

Step 6: Click on port and enter 80 in the dropdown and click ok

Output: the welcome page of nginx should be visible



Problem: Updates made in one container is not reflected into another container.

Solution: - Volume

Update made in one container within the volume will be reflected in all the containers of that volume.

Step 7: Creation of Volume(MyVolume)

Command:

a) docker volume create MyVolume
b) docker volume ls

c) docker volume inspect MyVolume
d) docker stop WebApp1

```
[node1] (local) root@192.168.0.18 ~
$ docker volume create MyVolume
MyVolume
[node1] (local) root@192.168.0.18 ~
docker volume ls
bash: dkocdocker: command not found
[node1] (local) root@192.168.0.18 ~
$ docker volume ls
DRIVER    VOLUME NAME
local     MyVolume
```

```
$ docker volume inspect MyVolume
[
    {
        "CreatedAt": "2023-03-25T07:02:17Z",
        "Driver": "local",
        "Labels": {},
        "Mountpoint": "/var/lib/docker/volumes/MyVolume/_data",
        "Name": "MyVolume",
        "Options": {},
        "Scope": "local"
    }
]
[node1] (local) root@192.168.0.18 ~
$ docker stop WebApp1
WebApp1
```

Step 8: Create a container (WebApp2) inside the container MyVolume

docker run -d –name = WebApp2 –mount source=MyVolume,destination=/usr/share/nginx/html –p 80:80 nginx

```
[node1] (local) root@192.168.0.18 ~
$ docker run -d --name=WebApp2 --mount source=MyVolume,destination=/usr/share/nginx/html -p 80:80 nginx
1f2df23596af1c82ba334d10f320bc7101043a2c78ffdbb43dd925c41c5e5bb1
[node1] (local) root@192.168.0.18 ~
```

Step 9: Enter the below commands:

a) ls /
b) cd /var/lib/docker
c) ls
d) cd volumes
e) ls
f) cd MyVolume
g) ls
h) cd _data

```
[node1] (local) root@192.168.0.18 ~
$ ls /
bin     dev     etc     lib     media   opt     root    sbin    sys     usr
certs   docker.log home  lib64   mnt     proc    run     srv     tmp     var
[node1] (local) root@192.168.0.18 ~
$ cd /var/lib/docker
[node1] (local) root@192.168.0.18 /var/lib/docker
$ ls
buildkit    containers  network   plugins   swarm   trust
containerd  image       overlay2  runtimes  tmp     volumes
```

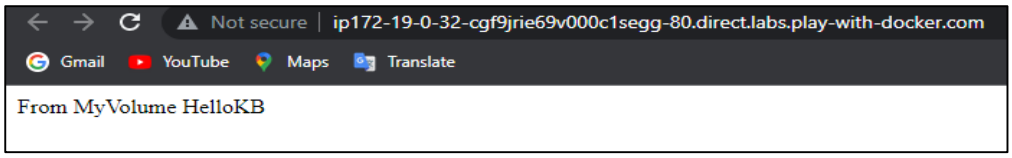
```
[node1] (local) root@192.168.0.18 /var/lib/docker
$ cd volumes
[node1] (local) root@192.168.0.18 /var/lib/docker/volumes
$ ls
MyVolume            backingFsBlockDev  metadata.db
[node1] (local) root@192.168.0.18 /var/lib/docker/volumes
$ cd MyVolume
[node1] (local) root@192.168.0.18 /var/lib/docker/volumes/MyVolume
$ ls
_data
```

```
[node1] (local) root@192.168.0.18 /var/lib/docker/volumes/MyVolume
$ cd _data
```

Step 10:Edit the index file with the below content to “Display the content on the Webpage”

```
[node1] (local) root@192.168.0.18 /var/lib/docker/volumes/MyVolume/_data
$ echo "From MyVolume HelloKB" > index.html
```

Open Port 80

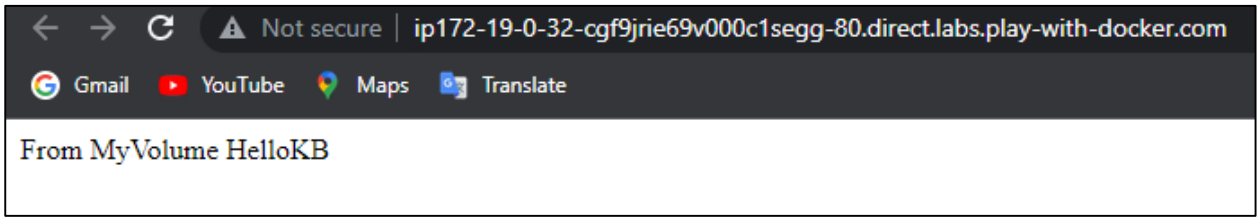


Step 11: Stop the above container (WebApp2) and Create another container within the volume (MyVolume)

```
[node1] (local) root@192.168.0.18 /var/lib/docker/volumes/MyVolume/_data
$ docker stop WebApp2
WebApp2
[node1] (local) root@192.168.0.18 /var/lib/docker/volumes/MyVolume/_data
$ docker run -d --name=WebApp3 --mount source=MyVolume,destination=/usr/share/nginx/html -p 80:80 nginx
8c5a64afec6835d8e473916fc7ed68356ff7861ba4177f38f5b78b900e6562c2
[node1] (local) root@192.168.0.18 /var/lib/docker/volumes/MyVolume/_data
```

Open port 80

Output: The edits made in one container of the volume will be reflected in all the containers of that volume

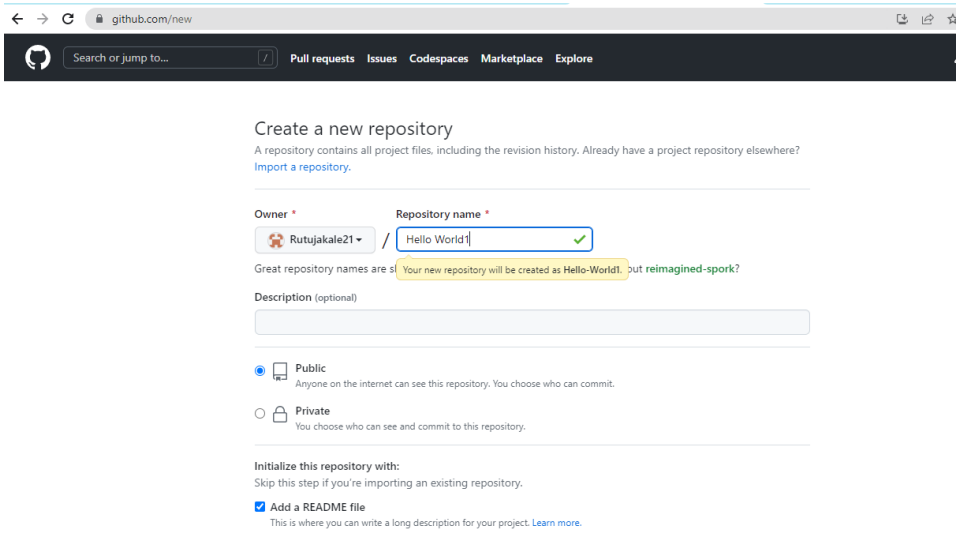


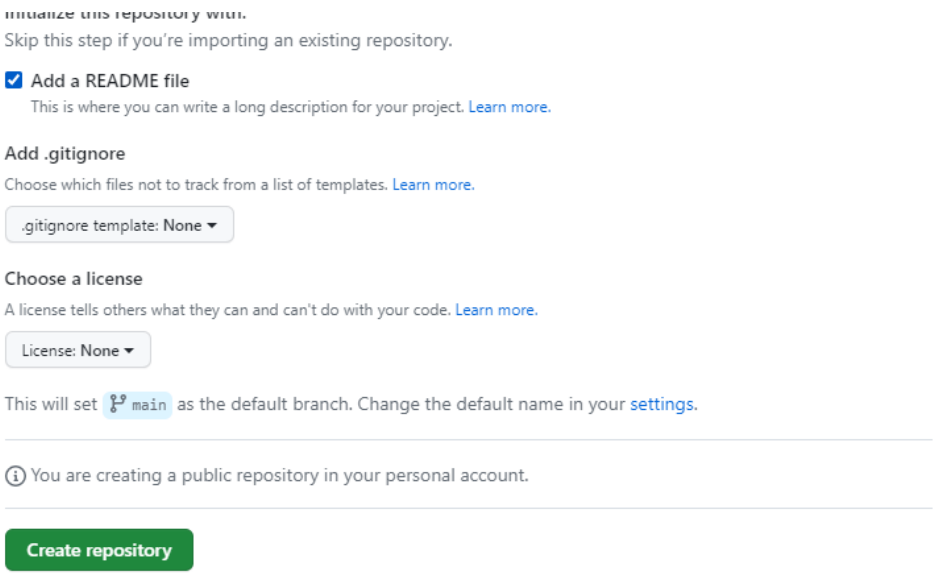
Practical 6

Aim: Working with Circle CI for continuous integration

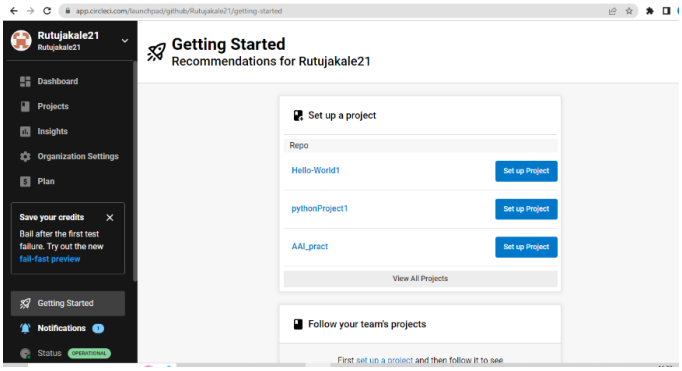
Step 1 - Create a repository

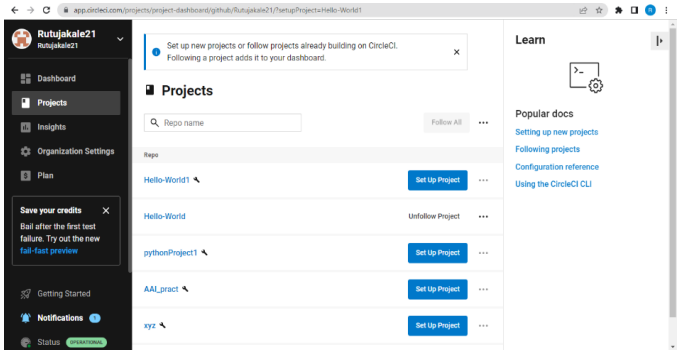
1. Log in to GitHub and begin the process to create a new repository.
2. Enter a name for your repository (for example, hello-world).
3. Select the option to initialize the repository with a README file.
4. Finally, click Create repository.
5. There is no need to add any source code for now

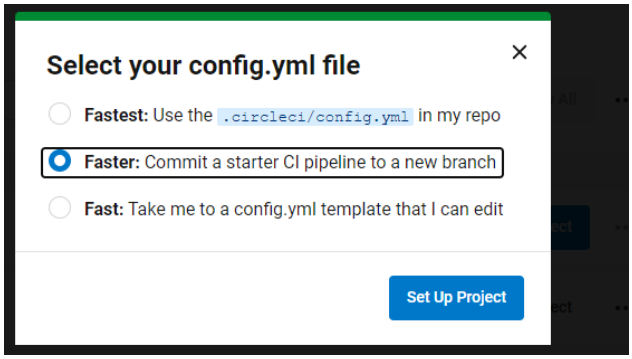


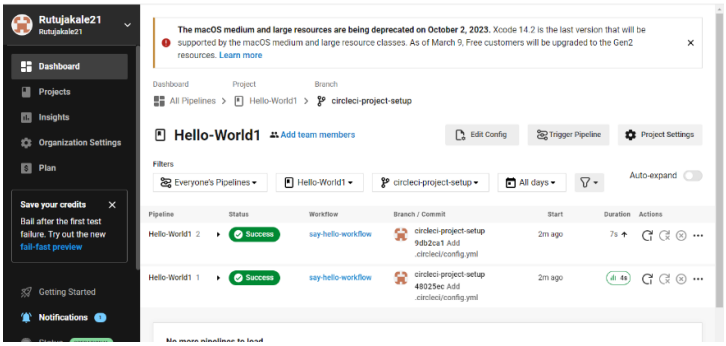


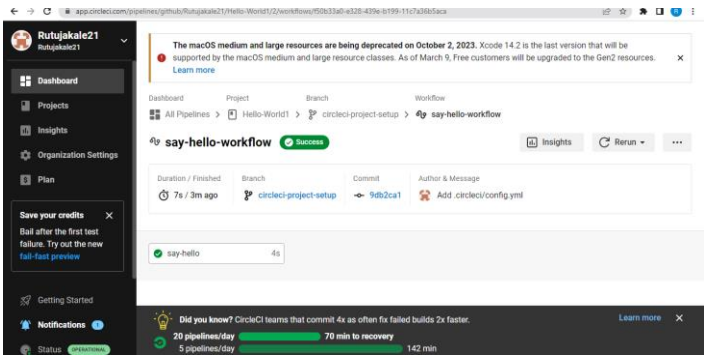
6. Login to Circle CI https://app.circleci.com/ Using GitHub Login, Once logged in navigate to Projects.

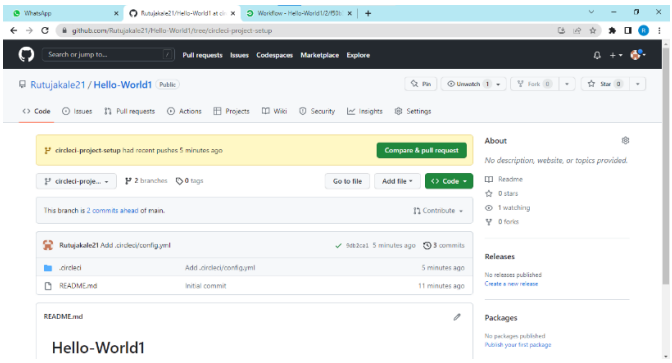


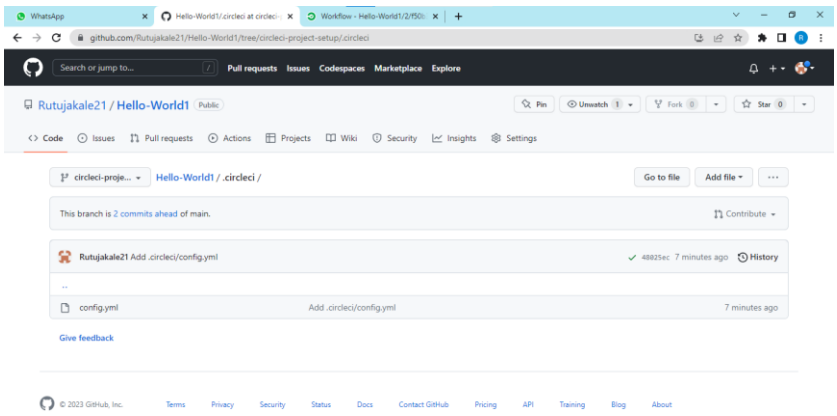


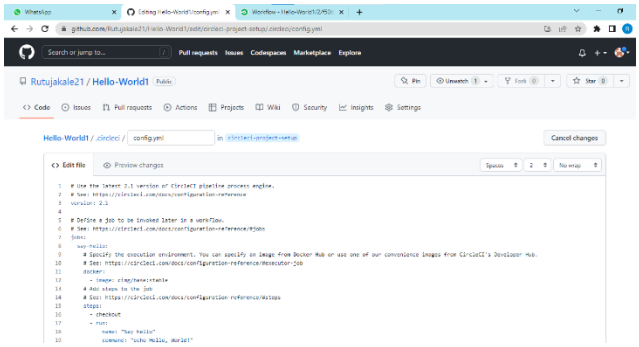


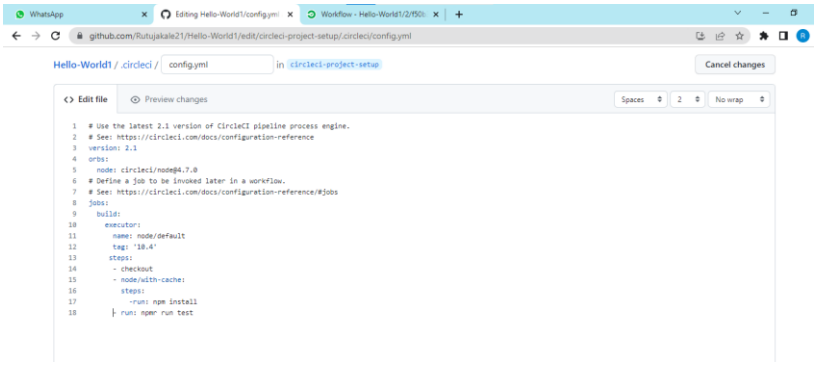


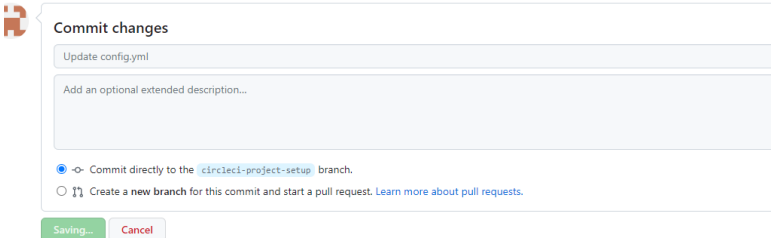


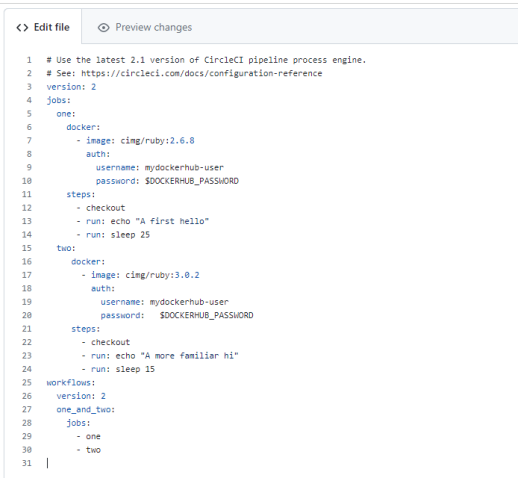


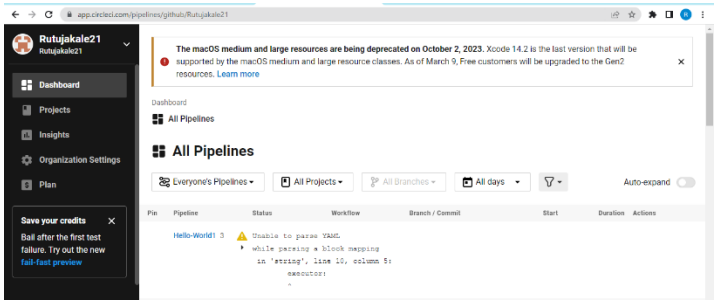


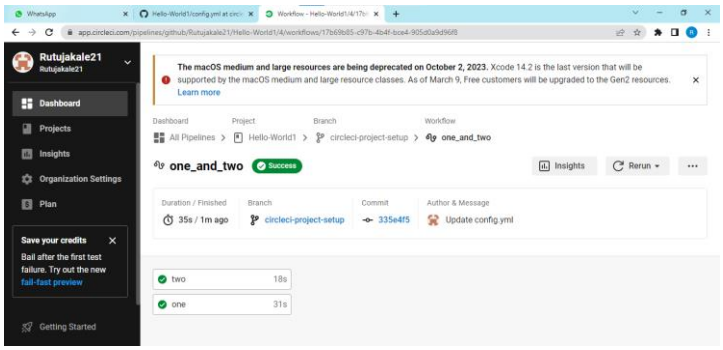

Rutujakale21 / Hello-World1
Hello-World1 / .circleci / config.yml
Cancel changes
Edit file
Preview changes

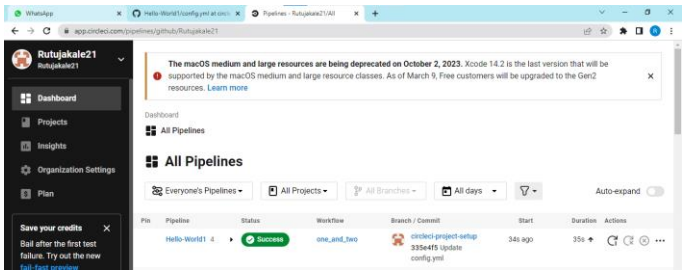

Hello-World1 / .circleci / config.yml
in circleci-project-setup
Cancel changes
Edit file
Preview changes
Use the latest 2.1 version of CircleCI pipeline process engine.
See: https://circleci.com/docs/configuration-reference
version: 2.1
orbs:
node: circleci/node@4.7.0
Define a job to be invoked later in a workflow.
See: https://circleci.com/docs/configuration-reference/#jobs
jobs:
build:
executor:
name: node/default
tag: '18.4'
steps:
- checkout
- node/with-cache:
steps:
-run: npm install
run: npm run test

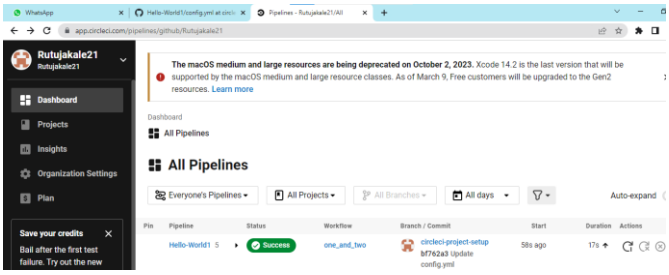

Commit changes
Update config.yml
Add an optional extended description...
Commit directly to the circleci-project-setup branch.
Create a new branch for this commit and start a pull request. Learn more about pull requests.
Saving...
Cancel

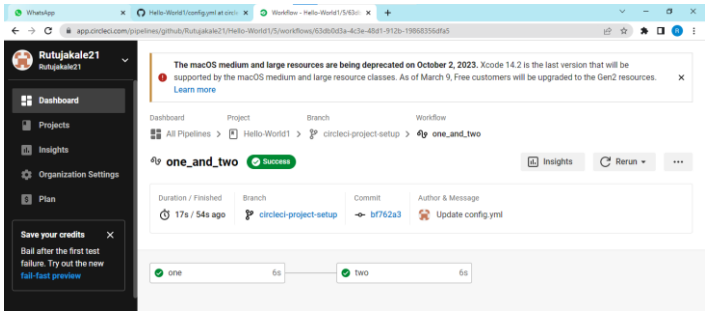

Edit file
Preview changes
Use the latest 2.1 version of CircleCI pipeline process engine.
See: https://circleci.com/docs/configuration-reference
version: 2
jobs:
one:
docker:
- image: cimg/ruby:2.6.8
auth:
username: mydockerhub-user
password: $DOCKERHUB_PASSWORD
steps:
- checkout
- run: echo "A first hello"
- run: sleep 25
two:
docker:
- image: cimg/ruby:3.0.2
auth:
username: mydockerhub-user
password: $DOCKERHUB_PASSWORD
steps:
- checkout
- run: echo "A more familiar hi"
- run: sleep 15
workflows:
version: 2
one_and_two:
jobs:
- one
- two


Rutujakale21
Dashboard
Projects
Insights
Organization Settings
Plan
The macOS medium and large resources are being deprecated on October 2, 2023. Xcode 14.2 is the last version that will be supported by the macOS medium and large resource classes. As of March 9, Free customers will be upgraded to the Gen2 resources. Learn more
All Pipelines
Everyone's Pipelines
All Projects
All Branches
All days
Auto-expand
Hello-World1 3
Unable to parse YAML.
while parsing a block mapping
in 'string', line 10, column 5:
executor:


Rutujakale21
Dashboard
Projects
Insights
Organization Settings
Plan
The macOS medium and large resources are being deprecated on October 2, 2023. Xcode 14.2 is the last version that will be supported by the macOS medium and large resource classes. As of March 9, Free customers will be upgraded to the Gen2 resources. Learn more
Dashboard
Project
Branch
Workflow
All Pipelines
Hello-World1
circleci-project-setup
one_and_two
one_and_two
Success
Insights
Rerun
Duration / Finished
35s / 1m ago
Branch
circleci-project-setup
Commit
335e4f5
Author & Message
Update config.yml
two 18s
one 31s
Getting Started


Rutujakale21
Dashboard
Projects
Insights
Organization Settings
Plan
The macOS medium and large resources are being deprecated on October 2, 2023. Xcode 14.2 is the last version that will be supported by the macOS medium and large resource classes. As of March 9, Free customers will be upgraded to the Gen2 resources. Learn more
All Pipelines
Everyone's Pipelines
All Projects
All Branches
All days
Auto-expand
Hello-World1 4
Success
one_and_two
circleci-project-setup
335e4f5 Update config.yml
34s ago
35s

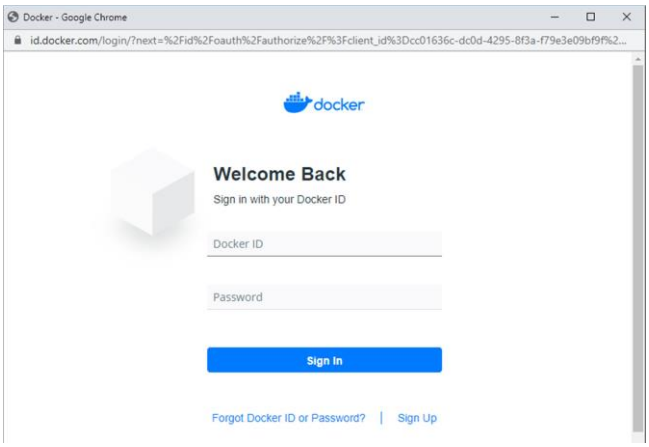

Rutujakale21
Dashboard
Projects
Insights
Organization Settings
Plan
The macOS medium and large resources are being deprecated on October 2, 2023. Xcode 14.2 is the last version that will be supported by the macOS medium and large resource classes. As of March 9, Free customers will be upgraded to the Gen2 resources. Learn more
All Pipelines
Everyone's Pipelines
All Projects
All Branches
All days
Auto-expand
Hello-World1 5
Success
one_and_two
circleci-project-setup
bf762a3 Update config.yml
58s ago
17s

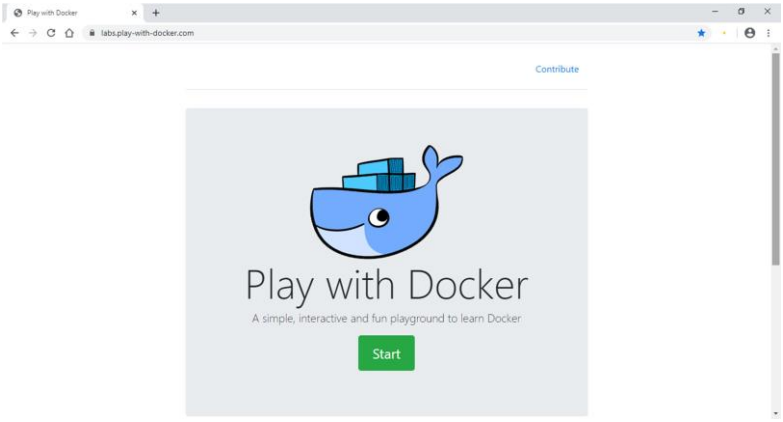

Rutujakale21
Dashboard
Projects
Insights
Organization Settings
Plan
The macOS medium and large resources are being deprecated on October 2, 2023. Xcode 14.2 is the last version that will be supported by the macOS medium and large resource classes. As of March 9, Free customers will be upgraded to the Gen2 resources. Learn more
Dashboard
Project
Branch
Workflow
All Pipelines
Hello-World1
circleci-project-setup
one_and_two
one_and_two
Success
Insights
Rerun
Duration / Finished
17s / 54s ago
Branch
circleci-project-setup
Commit
bf762a3
Author & Message
Update config.yml
one 6s
two 6s

Practical No.:07

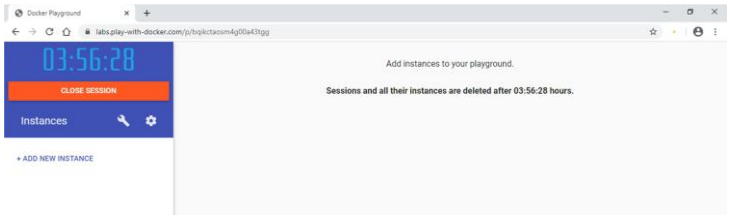
Aim:- Creating Backing service with ASP.NET 2.0 core.

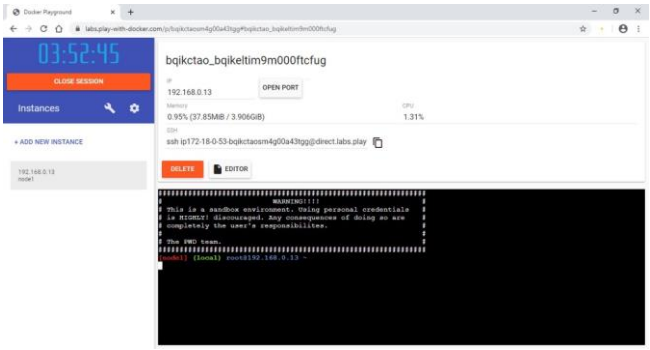
Now login in to Play-With-Docker





Click on Add New Instance

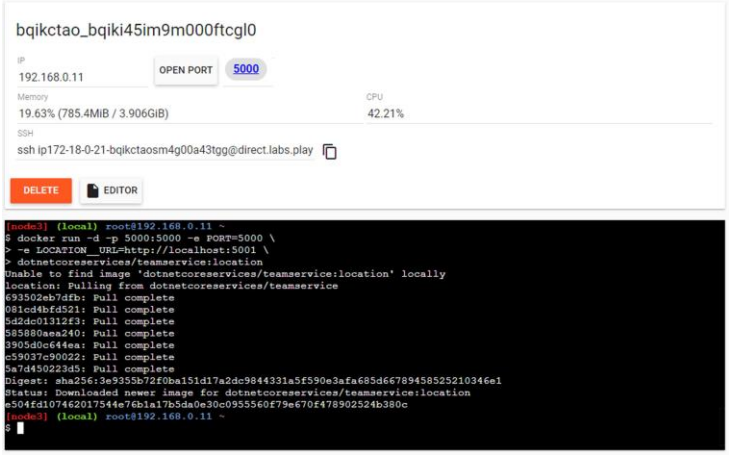




Start typing following commands

Command :docker run -d -p 5000:5000 -e PORT=5000 \-e LOCATION URL=http://localhost:5001 \dotnetcoreservices/teamservice:location

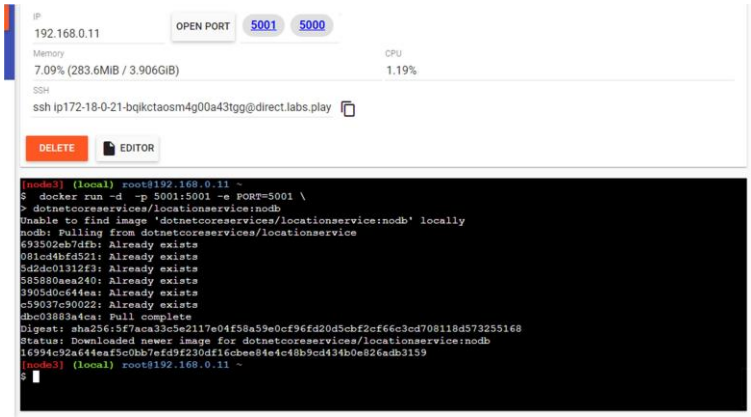
output: (you can observe that it has started port 5000 on top)



Command: to run location service

docker run -d -p 5001:5001 -e PORT=5001 dotnetcoreservices/locationservice:nodb

output: (now it has started one more port that is 5001 for location service)



Command : to check running images in docker $docker images

output:

Command: to create new team

curl -H "Content-Type:application/json" -X POST -d \'{"id":"e52baa63-d511-417e-9e54-7aab04286281", "name":"KC"}' http://localhost:5000/teams

Output:Command :To confirm that team is added

```
[node3] (local) root@192.168.0.11 ~
$ curl -H "Content-Type:application/json" -X POST -d \
> '{"id":"e52baa63-d511-417e-9e54-7aab04286281", "name":"KC"}'   http://localhost:5000/teams
{"name":"KC","id":"e52baa63-d511-417e-9e54-7aab04286281","members":[]}[node3] (local) root@192.168.0.11 ~
$
```

curl http://localhost:5000/teams/e52baa63-d511-417e-9e54-7aab04286281

Output

```
[node3] (local) root@192.168.0.11 ~
$ curl -H "Content-Type:application/json" -X POST -d \
> '{"id":"e52baa63-d511-417e-9e54-7aab04286281", "name":"KC"}'   http://localhost:5000/teams
{"name":"KC","id":"e52baa63-d511-417e-9e54-7aab04286281","members":[]}[node3] (local) root@192.168.0.11 ~
$ curl http://localhost:5000/teams/e52baa63-d511-417e-9e54-7aab04286281
{"name":"KC","id":"e52baa63-d511-417e-9e54-7aab04286281","members":[]}[node3] (local) root@192.168.0.11 ~
$
```

Command : to add new member to team

curl -H "Content-Type:application/json" -X POST -d \'{"id":"63e7acf8-8fae-42ce-9349-3c8593ac8292", "firstName":"Kirti", "lastName":"Bhatt"}' http://localhost:5000/teams/e52baa63-d511-417e-9e54-7aab04286281/members

Output:

```
[node1] (local) root@192.168.0.23 ~
$ curl -H "Content-Type:application/json" -X POST -d \
> '{"id":"63e7acf8-8fae-42ce-9349-3c8593ac8292", "firstName":"Kirti", "lastName":"Bhatt"}' http://localhost:5000/
teams/e52baa63-d511-417e-9e54-7aab04286281/members
{"teamID":"e52baa63-d511-417e-9e54-7aab04286281","memberID":"63e7acf8-8fae-42ce-9349-3c8593ac8292"}[node1] (local
) root@192.168.0.23 ~
$
```

Command :To confirm member added

curl http://localhost:5000/teams/e52baa63-d511-417e-9e54-7aab04286281

output:

```
[node1] (local) root@192.168.0.23 ~
$ curl http://localhost:5000/teams/e52baa63-d511-417e-9e54-7aab04286281
{"name":"KC","id":"e52baa63-d511-417e-9e54-7aab04286281","members":[null,{"id":"63e7acf8-8fae-42ce-9349-3c8593ac8
292","firstName":"Kirti","lastName":"Bhatt"}]}[node1] (local) root@192.168.0.23 ~
$
```

Command : To add location for member

curl -H "Content-Type:application/json" -X POST -d \'{"id":"64c3e69f-1580-4b2f-a9ff-2c5f3b8f0e1f", "latitude":12.0,"longitude":12.0,"altitude":10.0, "timestamp":0,"memberId":"63e7acf8-8fae-42ce-9349-3c8593ac8292"}' http://localhost:5001/locations/63e7acf8-8fae-42ce-9349-3c8593ac8292

output:

```
[node1] (local) root@192.168.0.23 ~
$ curl -H "Content-Type:application/json" -X POST -d \
> '{"id":"64c3e69f-1580-4b2f-a9ff-2c5f3b8f0e1f", "latitude":12.0,"longitude":12.0,"altitude":10.0, "timestamp":0,
"memberId":"63e7acf8-8fae-42ce-9349-3c8593ac8292"}' http://localhost:5001/locations/63e7acf8-8fae-42ce-9349-3c859
3ac8292
{"id":"64c3e69f-1580-4b2f-a9ff-2c5f3b8f0e1f","latitude":12.0,"longitude":12.0,"altitude":10.0,"timestamp":0,"memb
erID":"63e7acf8-8fae-42ce-9349-3c8593ac8292"}[node1] (local) root@192.168.0.23 ~
$
```

Command : To confirm location is added in member

curl http://localhost:5001/locations/63e7acf8-8fae-42ce-9349-3c8593ac8292

output:

```
[node1] (local) root@192.168.0.23 ~
$ curl http://localhost:5001/locations/63e7acf8-8fae-42ce-9349-3c8593ac8292
[{"id":"64c3e69f-1580-4b2f-a9ff-2c5f3b8f0e1f","latitude":12.0,"longitude":12.0,"altitude":10.0,"timestamp":0,"mem
berID":"63e7acf8-8fae-42ce-9349-3c8593ac8292"}][node1] (local) root@192.168.0.23 ~
$
```