

MICROSERVICE ARCHITECTURE

INDEX

| Practical No. | Title | Page No. |
|--------------------------|--|---------------------|
| 1 | Build a Web Application using MVC. | 01 |
| 2 | Build a Web Application using API. | 06 |
| 3 | Working with Docker Containers and Commands: Pulling and pushing image to docker. | 12 |
| 4 | Working with Docker Containers and Commands: Build an image then push it to docker and run it | 18 |
| 5 | Build a Web App and publish it to Docker. | 21 |
| 6 | Working with the CircleCI | 28 |
| 7 | Running location service in Docker. | 35 |

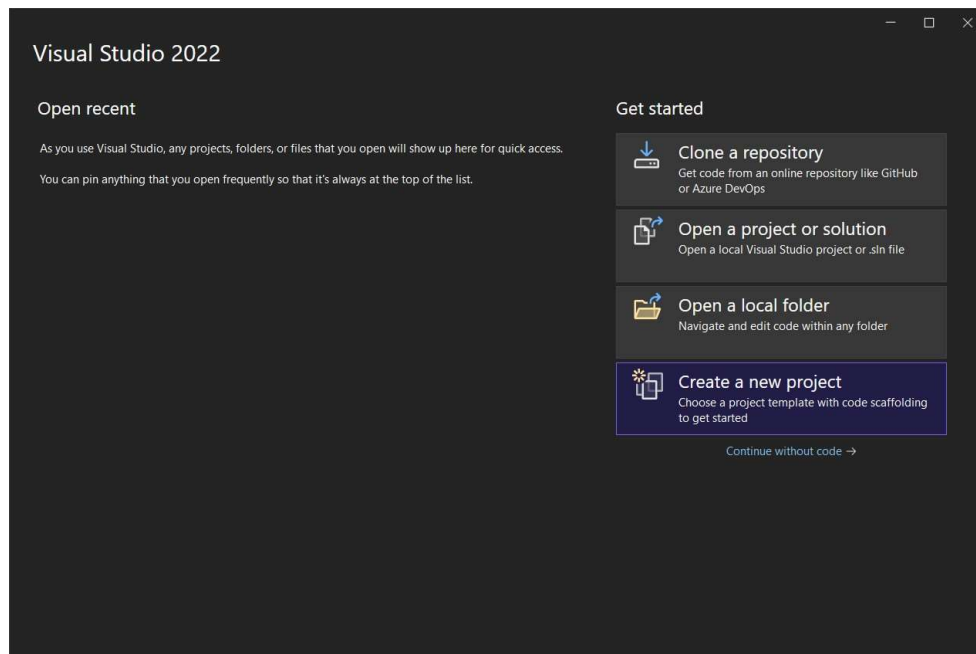
Practical No. 01

Aim: Build a Web Application using MVC.

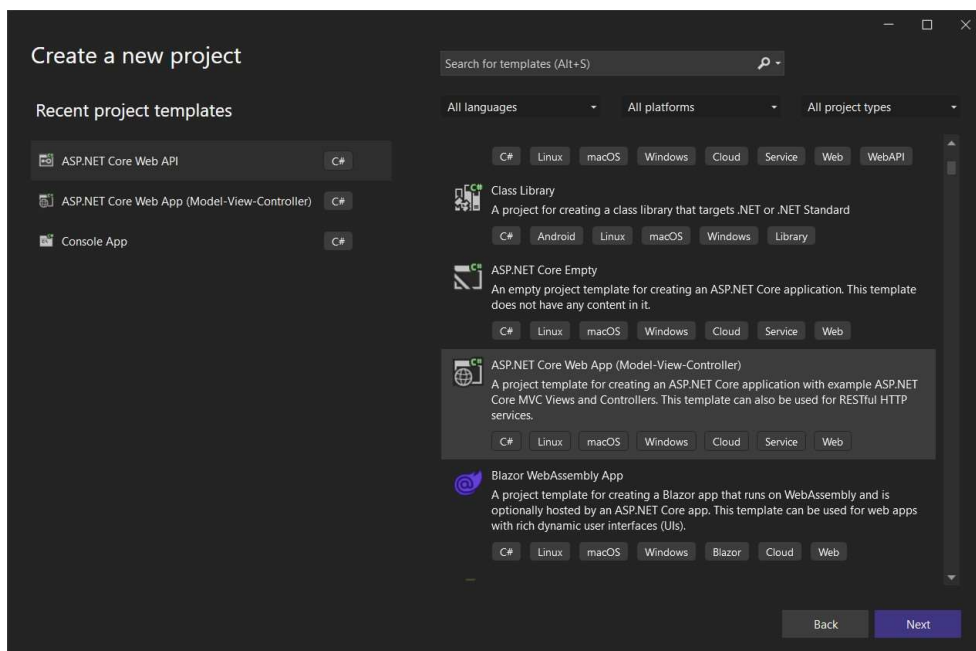
Firstly download, install & open **Microsoft Visual Studio 2022**

(<https://visualstudio.microsoft.com/vs/>)

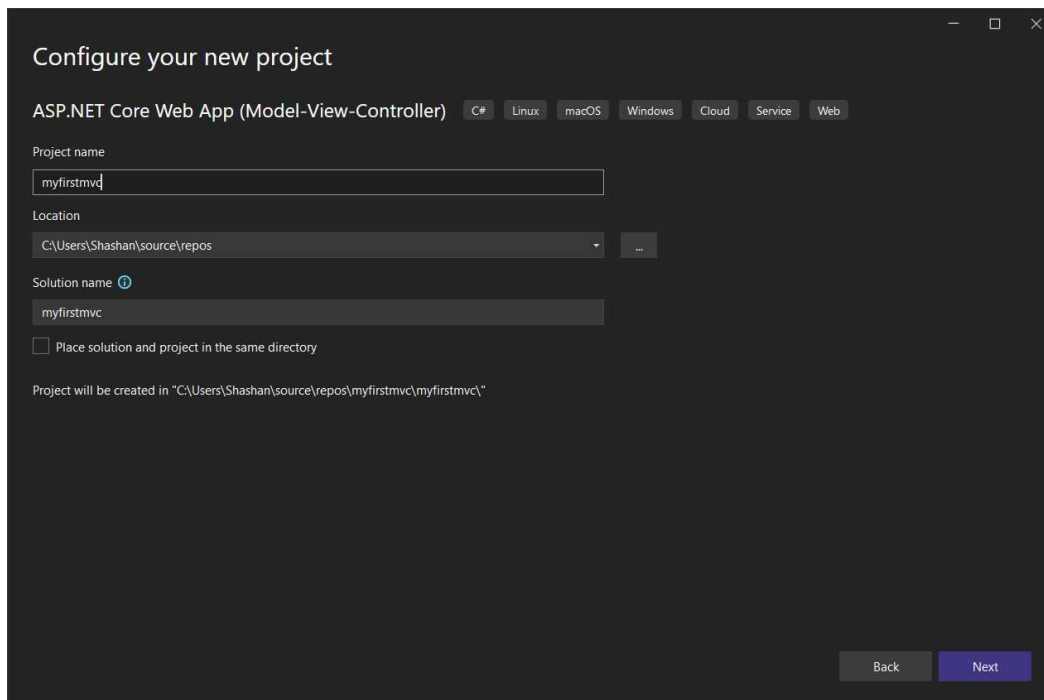
Create a new project:



Select the **ASP.NET Core Web App (Model-View_Controller)**



Provide a name to the project:



Configure your new project

ASP.NET Core Web App (Model-View-Controller) C# Linux macOS Windows Cloud Service Web

Project name
myfirstmvc

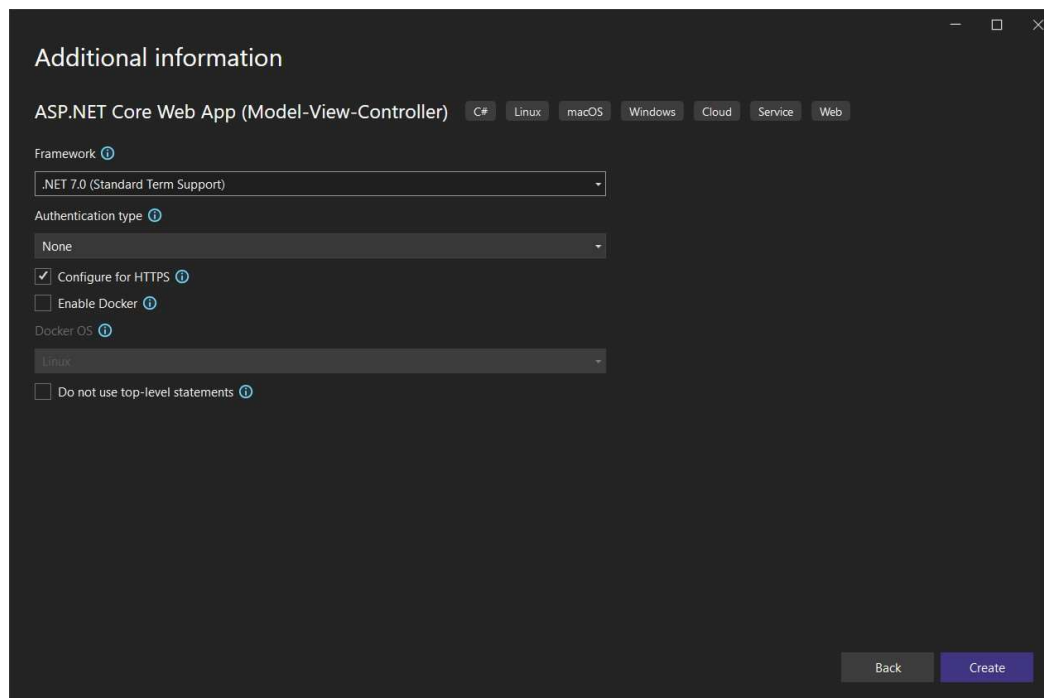
Location
C:\Users\Shashan\source\repos

Solution name ⓘ
myfirstmvc

☐ Place solution and project in the same directory

Project will be created in "C:\Users\Shashan\source\repos\myfirstmvc\myfirstmvc\."

Back Next



Additional information

ASP.NET Core Web App (Model-View-Controller) C# Linux macOS Windows Cloud Service Web

Framework ⓘ
.NET 7.0 (Standard Term Support)

Authentication type ⓘ
None

☒ Configure for HTTPS ⓘ

☐ Enable Docker ⓘ

Docker OS ⓘ
Linux

☐ Do not use top-level statements ⓘ

Back Create

Code for HomeController.cs under Controller folder:

```
using Microsoft.AspNetCore.Mvc;
using myfirstmvc.Models;
using System.Diagnostics;
namespace myfirstmvc.Controllers
{
    public class HomeController : Controller
    {
        private readonly ILogger<HomeController> _logger;

        public HomeController(ILogger<HomeController> logger)
        {
            _logger = logger;
        }
        public IActionResult Index()
        {
            return View();
        }
        public IActionResult Privacy()
        {
            return View();
        }
        [ResponseCache(Duration = 0, Location = ResponseCacheLocation.None, NoStore =
true)]
        public IActionResult Error()
        {
            return View(new ErrorViewModel { RequestId = Activity.Current?.Id ??
HttpContext.TraceIdentifier });
        }
    }
}
```

Code for ErrorViewModel.cs under Model folder:

```
namespace myfirstmvc.Models
{
    public class ErrorViewModel
    {
        public string? RequestId { get; set; }
        public bool ShowRequestId => !string.IsNullOrEmpty(RequestId);
    }
}
```

Code for Index.cshtml under View\Home folder:

```
@{
    ViewData["Title"] = "Home Page";
}

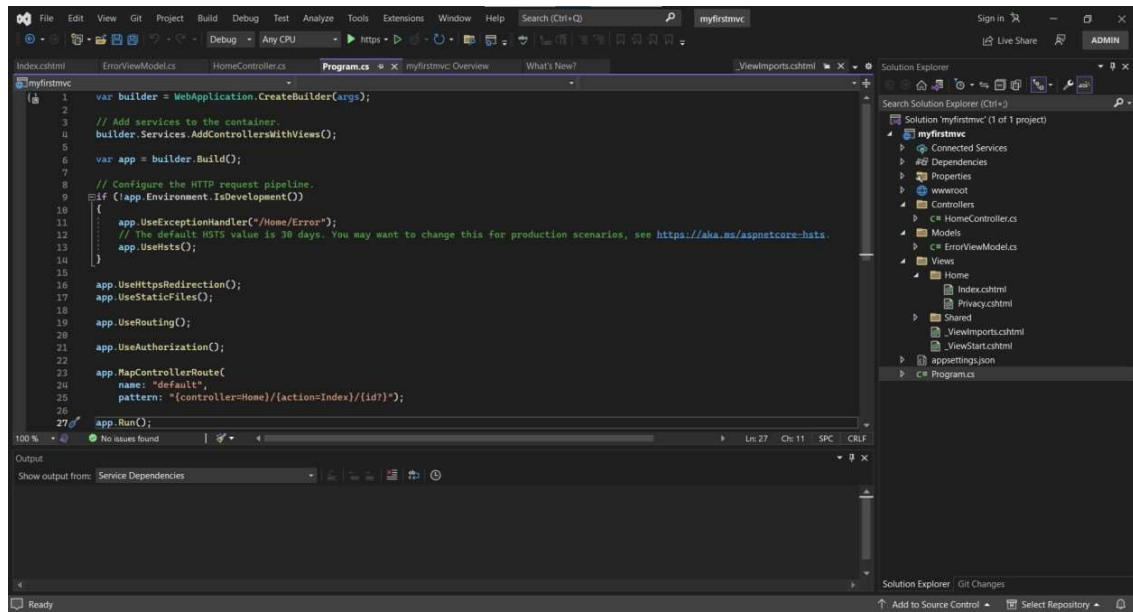
<div class="text-center">
    <h1 class="display-4">Welcome</h1>
    <p>Learn about <a href="https://docs.microsoft.com/aspnet/core">building Web apps with
ASP.NET Core</a>.</p>
</div>
```

Code for Privacy.cshtml under View\Home folder:

```
@{
    ViewData["Title"] = "Privacy Policy";
}
<h1>@ViewData["Title"]</h1>
<p>Use this page to detail your site's privacy policy.</p>
```

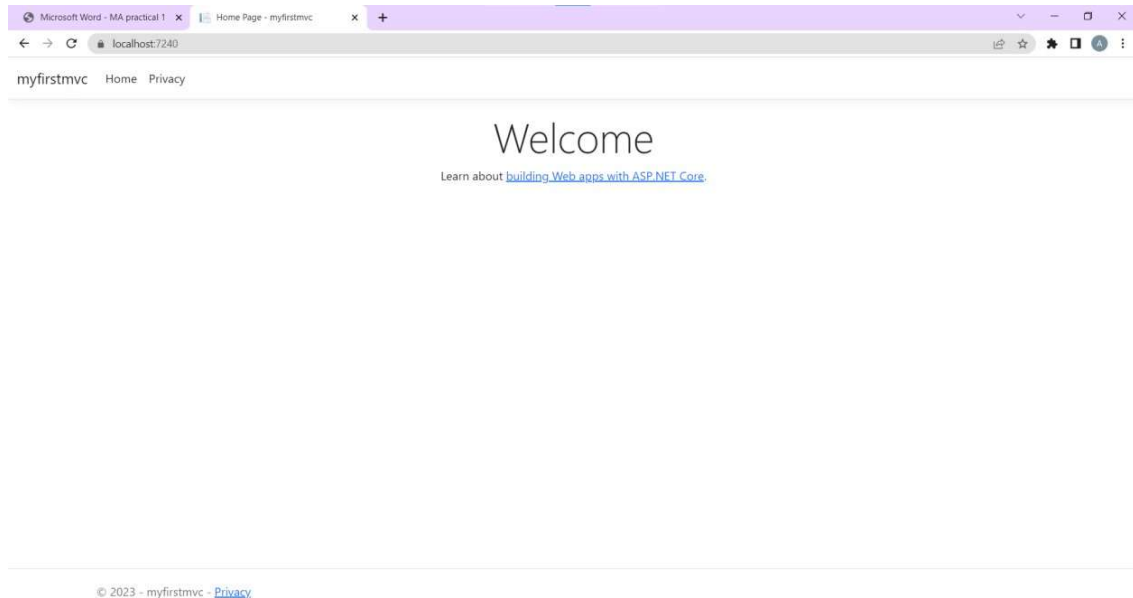
Code for Program.cs:

```
var builder = WebApplication.CreateBuilder(args);
// Add services to the container.
builder.Services.AddControllersWithViews();
var app = builder.Build();
// Configure the HTTP request pipeline.
if (!app.Environment.IsDevelopment())
{
    app.UseExceptionHandler("/Home/Error");
    // The default HSTS value is 30 days. You may want to change this for production
    scenarios, see https://aka.ms/aspnetcore-hsts.
    app.UseHsts();
}
app.UseHttpsRedirection();
app.UseStaticFiles();
app.UseRouting();
app.UseAuthorization();
app.MapControllerRoute(
    name: "default",
    pattern: "{controller=Home}/{action=Index}/{id?}");
app.Run();
```



After that click the run project button under the top ribbon

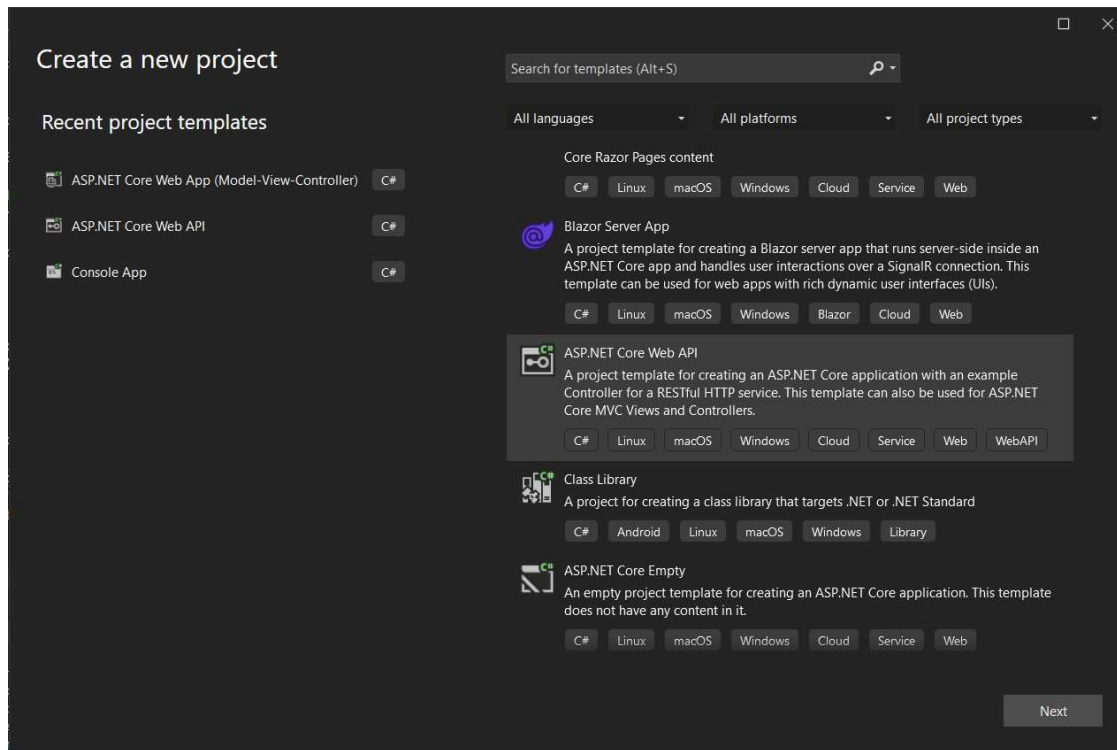
Output:



Practical No. 02

Aim: Build a Web Application using API.

Create a new project with ASP.NET Core Web API



Code for *Program.cs*:

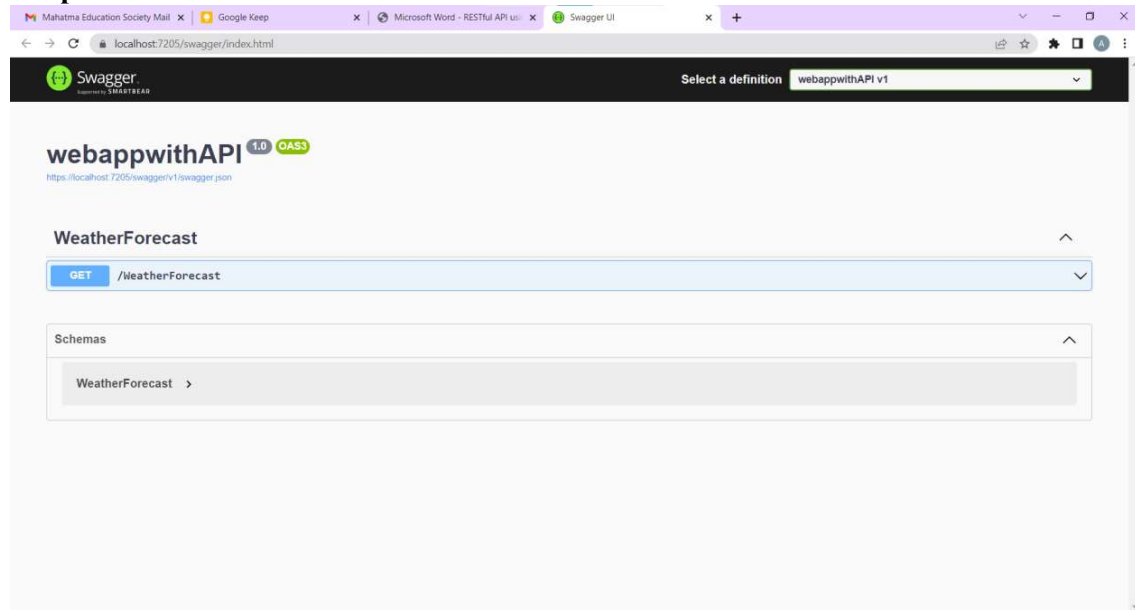
```
var builder = WebApplication.CreateBuilder(args);
// Add services to the container.
builder.Services.AddControllers();
// Learn more about configuring Swagger/OpenAPI at https://aka.ms/aspnetcore/swashbuckle
builder.Services.AddEndpointsApiExplorer();
builder.Services.AddSwaggerGen();
var app = builder.Build();
// Configure the HTTP request pipeline.
if (app.Environment.IsDevelopment())
{
    app.UseSwagger();
    app.UseSwaggerUI();
}
app.UseHttpsRedirection();
app.UseAuthorization();
app.MapControllers();
app.Run();
```


Code for *WeatherForecastController.cs* under the folder *Controllers*:

```
using Microsoft.AspNetCore.Mvc;
namespace webappwithAPI.Controllers
{
    [ApiController]
    [Route("[controller]")]
    public class WeatherForecastController : ControllerBase
    {
        private static readonly string[] Summaries = new[]
        {
            "Freezing", "Bracing", "Chilly", "Cool", "Mild", "Warm", "Balmy", "Hot",
            "Sweltering", "Scorching"
        };
        private readonly ILogger<WeatherForecastController> _logger;
        public WeatherForecastController(ILogger<WeatherForecastController> logger)
        {
            _logger = logger;
        }
        [HttpGet(Name = "GetWeatherForecast")]
        public IEnumerable<WeatherForecast> Get()
        {
            return Enumerable.Range(1, 5).Select(index => new WeatherForecast
            {
                Date = DateOnly.FromDateTime(DateTime.Now.AddDays(index)),
                TemperatureC = Random.Shared.Next(-20, 55),
                Summary = Summaries[Random.Shared.Next(Summaries.Length)]
            })
            .ToArray();
        }
    }
}
```

Code for *WeatherForecast.cs*:

```
namespace webappwithAPI
{
    public class WeatherForecast
    {
        public DateOnly Date { get; set; }
        public int TemperatureC { get; set; }
        public int TemperatureF => 32 + (int)(TemperatureC / 0.5556);
        public string? Summary { get; set; }
    }
}
```

Output:

Replace the *WeatherForecastController.cs* under the folder *Controllers* with *GlossaryController.cs*:

```
//Controllers/GlossaryController.cs using System;
using System.Collections.Generic;
using Microsoft.AspNetCore.Mvc;
using System.IO;
namespace Glossary.Controllers
{
    [ApiController]
    [Route("api/[controller]")]
    public class GlossaryController : ControllerBase
    {
        private static List<GlossaryItem> Glossary = new List<GlossaryItem> { new
GlossaryItem
        {
            Term= "HTML",
            Definition = "Hypertext Markup Language"
        },
        new GlossaryItem
        {
            Term= "MVC",
            Definition = "Model View Controller"
        },
        new GlossaryItem
        {
```

```
Term= "OpenID",
Definition = "An open standard for authentication"
}
};

[HttpGet]
public ActionResult<List<GlossaryItem>> Get()
{
    return Ok(Glossary);
}
[HttpGet]
[Route("{term}")]
public ActionResult<GlossaryItem> Get(string term)
{
    var glossaryItem = Glossary.Find(item =>
        item.Term.Equals(term, StringComparison.InvariantCultureIgnoreCase));

    if (glossaryItem == null)
    {
        return NotFound();
    }
    else
    {
        return Ok(glossaryItem);
    }
}
[HttpPost]
public ActionResult Post(GlossaryItem glossaryItem)
{
    var existingGlossaryItem = Glossary.Find(item =>
        item.Term.Equals(glossaryItem.Term,
StringComparison.InvariantCultureIgnoreCase));

    if (existingGlossaryItem != null)
    {
        return Conflict("Cannot create the term because it already exists.");
    }
    else
    {
        Glossary.Add(glossaryItem);
        var resourceUrl = Path.Combine(Request.Path.ToString(),
Uri.EscapeUriString(glossaryItem.Term)); return Created(resourceUrl, glossaryItem);
    }
}
[HttpPut]
public ActionResult Put(GlossaryItem glossaryItem)
{

```

```

        var existingGlossaryItem = Glossary.Find(item =>
item.Term.Equals(glossaryItem.Term, StringComparison.InvariantCultureIgnoreCase));
        if (existingGlossaryItem == null)
        {
            return BadRequest("Cannot update a nont existing term.");
        }
        else
        {
            existingGlossaryItem.Definition = glossaryItem.Definition; return Ok();
        }
    }
    [HttpDelete]
    [Route("{term}")]
    public ActionResult Delete(string term)
    {
        var glossaryItem = Glossary.Find(item =>
item.Term.Equals(term, StringComparison.InvariantCultureIgnoreCase));
        if (glossaryItem == null)
        {
            return NotFound();
        }
        else
        {
            Glossary.Remove(glossaryItem); return NoContent();
        }
    }
}

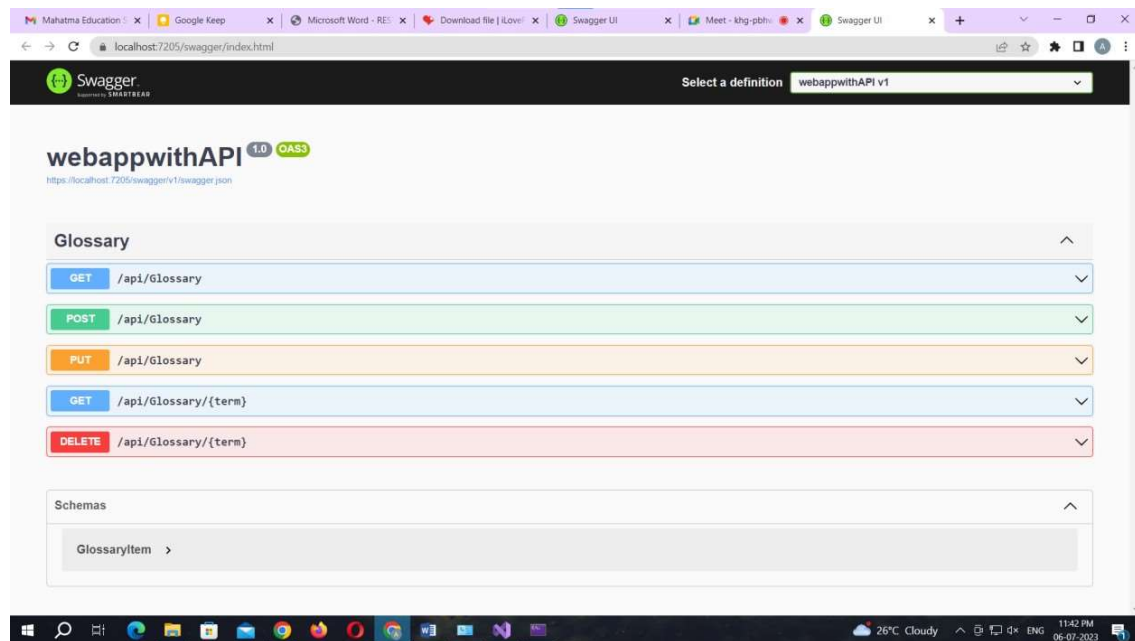
```

Replace the *WeatherForecast.cs* with *GlossaryItem.cs*:

```

//GlossaryItem.cs
namespace Glossary
{
    public class GlossaryItem
    {
        public string Term { get; set; }
        public string Definition { get; set; }
    }
}

```

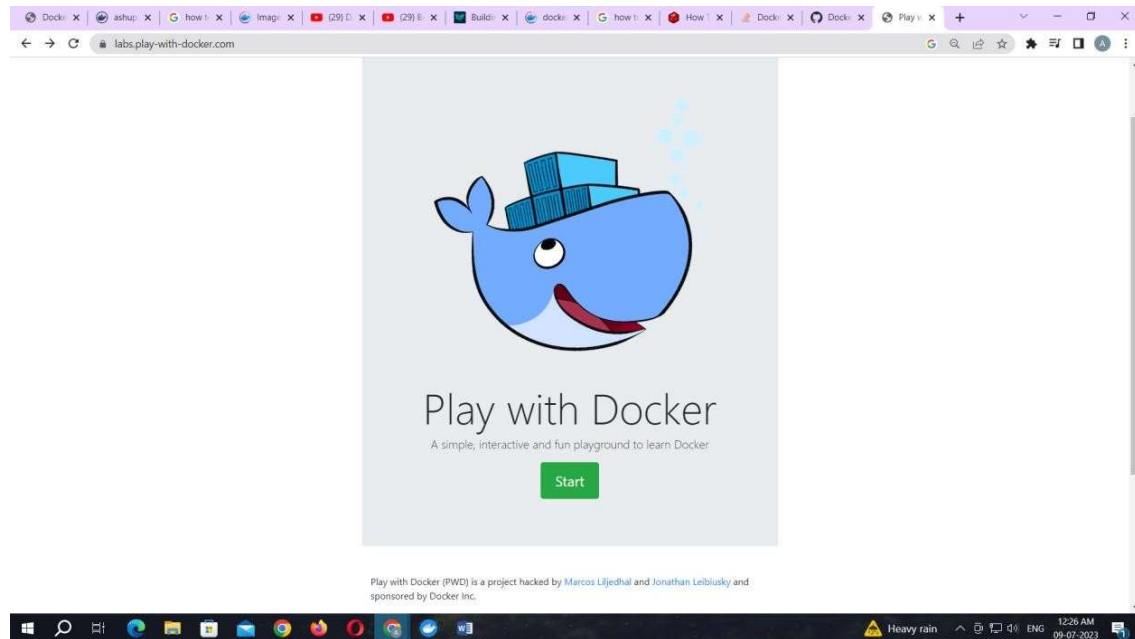
Output:

Practical No. 03

Aim: Working with Docker Containers and Commands: Pulling and pushing image to docker.

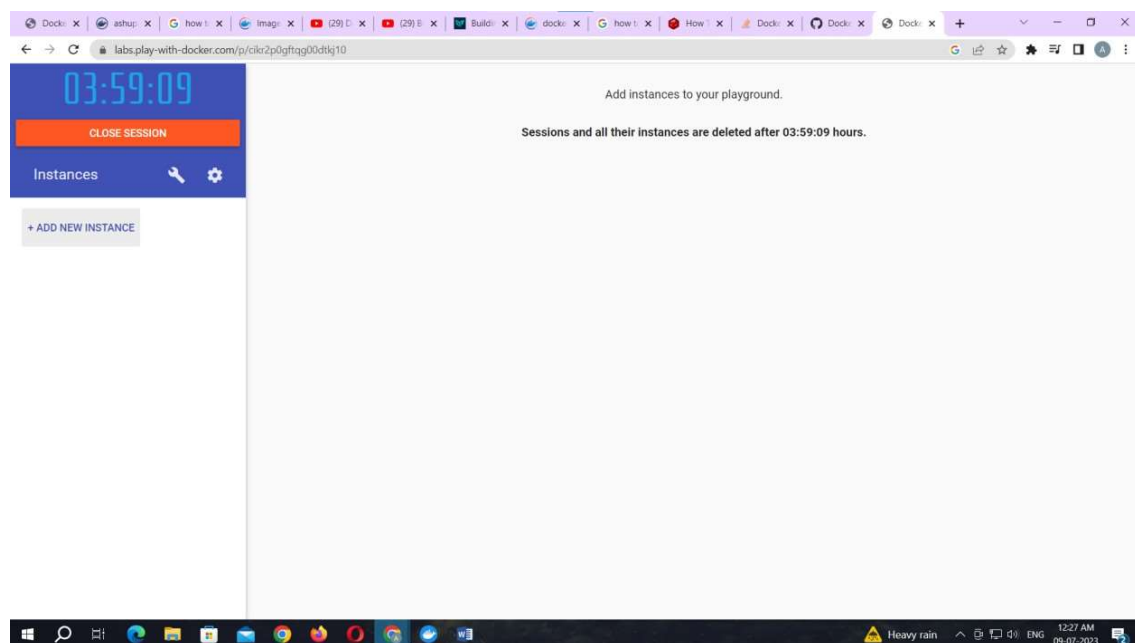
Create Docker Hub account (sign up)

Login to <https://labs.play-with-docker.com/>



Click on start

Add new instance

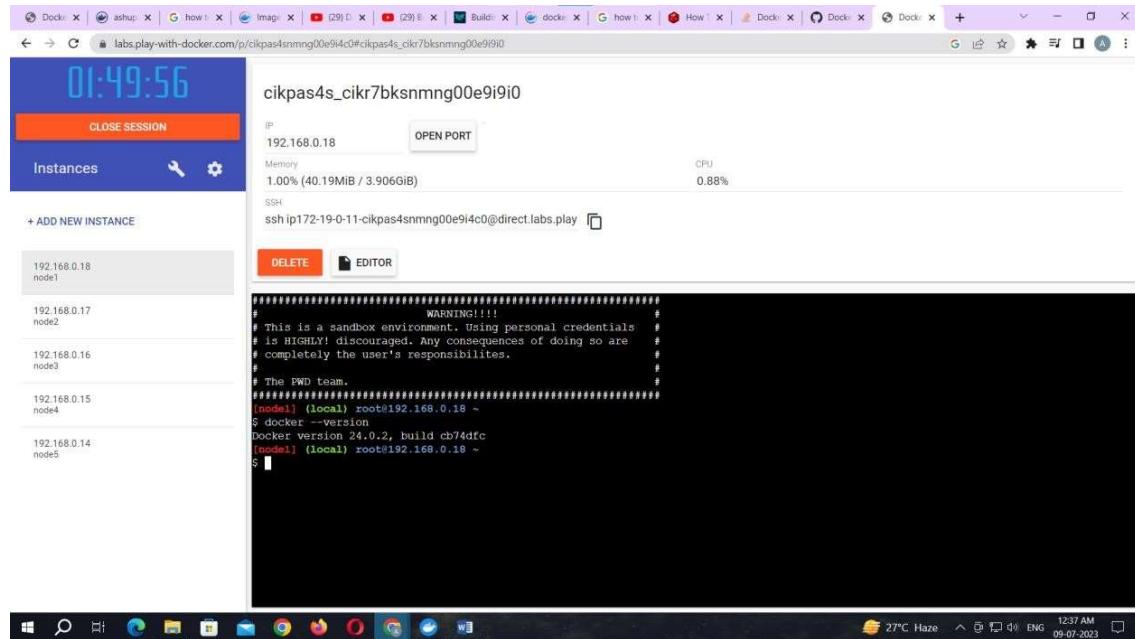


To pull and push images using docker

Command: to check docker version

docker --version

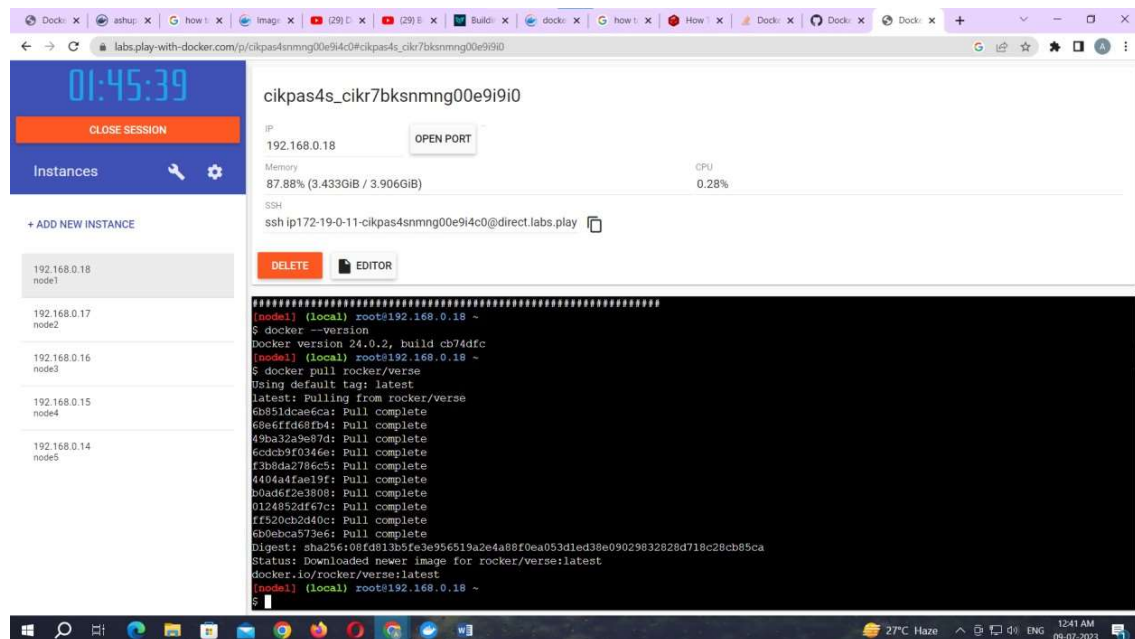
Output:



Command: to pull readymade image

docker pull rocker/verse

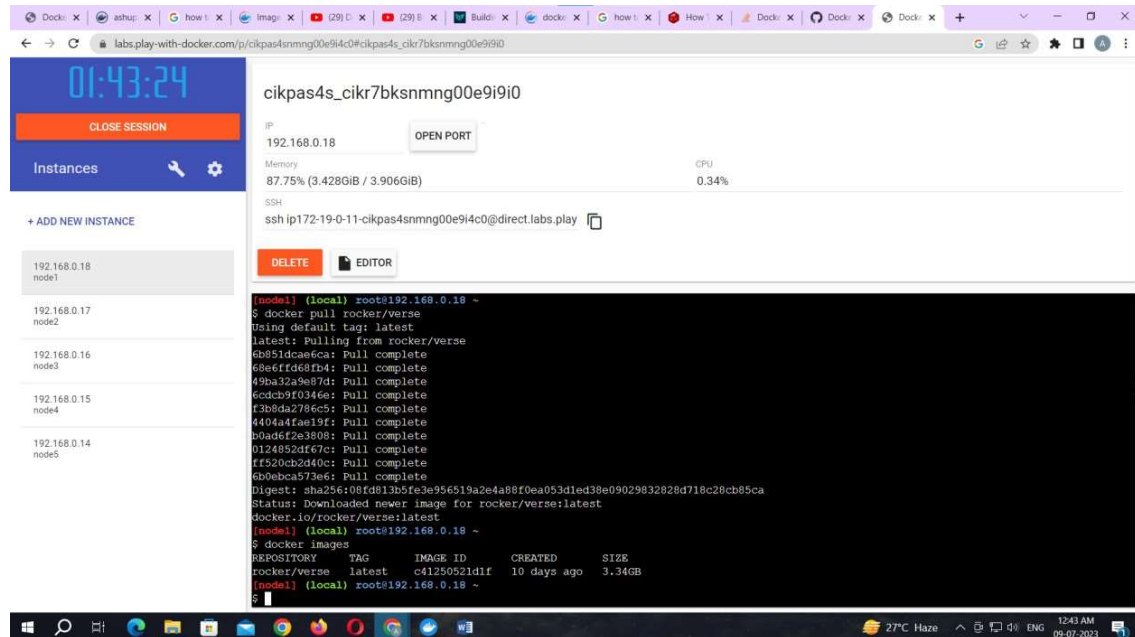
Output:



Command: to check images in docker

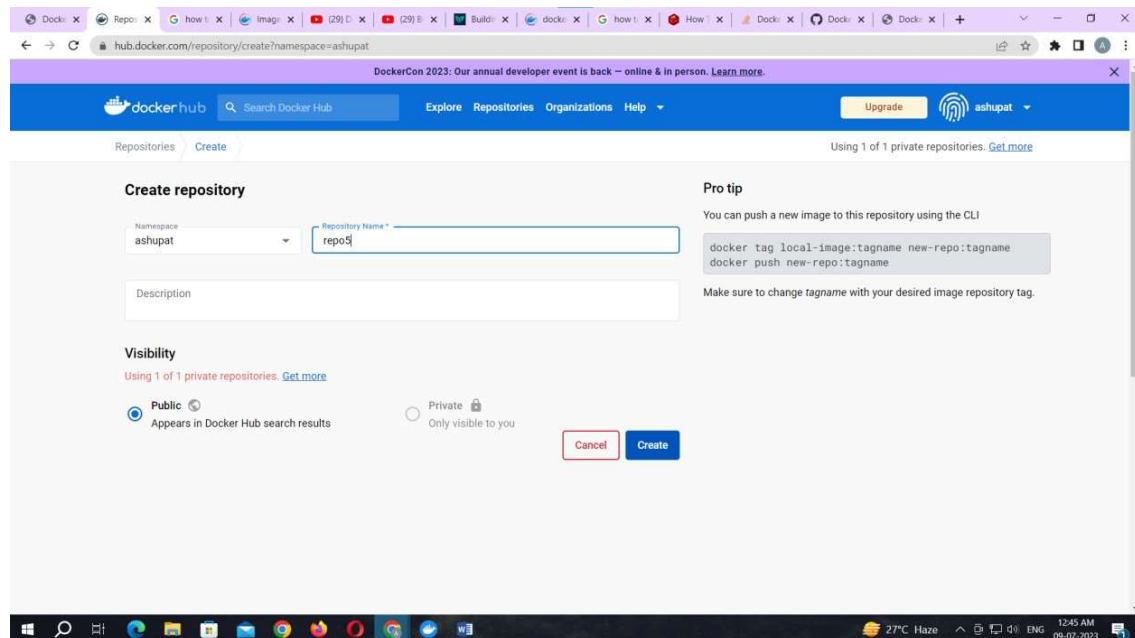
docker images

Output:



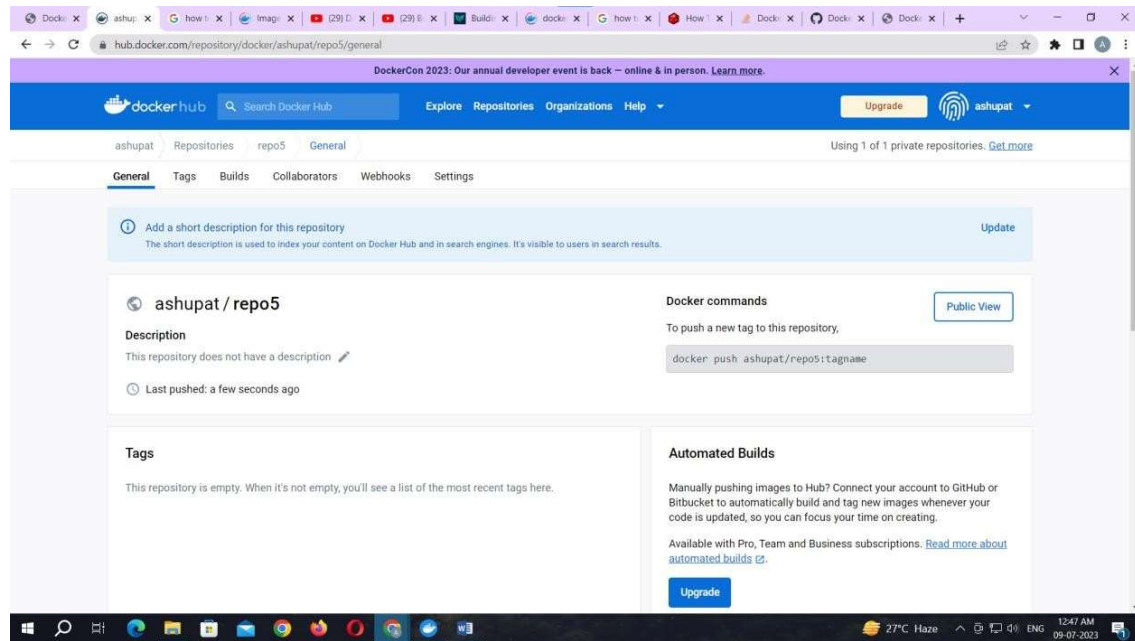
Now Login to docker hub and create repository

Output:



Click on Create button

Now check repository created

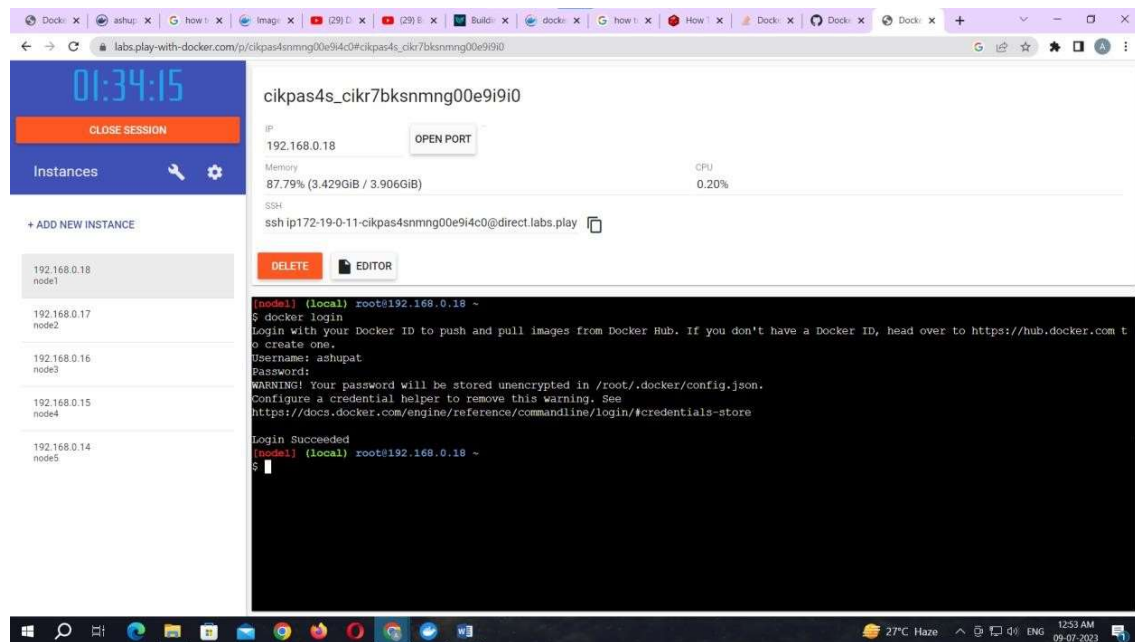


Command: to login to your docker account

docker login --username=ashupat password:

note:ashupat is my docker ID . You will use your docker ID here. And enter your password .

Output:



Command: to tag image

docker tag c41250521d1f ashupat/repo5:newverse

note: here c41250521d1f this is image id which you can get from docker images command.

Output:

01:31:14

CLOSE SESSION

Instances

+ ADD NEW INSTANCE

192.168.0.18 node1

192.168.0.17 node2

192.168.0.16 node3

192.168.0.15 node4

192.168.0.14 node5

cikpas4s_cikr7bksnmng00e9i9i0

IP: 192.168.0.18 OPEN PORT

Memory: 87.80% (3.43GiB / 3.906GiB) CPU: 0.26%

SSH: ssh ip172-19-0-11-cikpas4snmng00e9i4c0@direct.labs.play

DELETE EDITOR

```

(node1) (local) root@192.168.0.18 ~
$ docker login
Login with your Docker ID to push and pull images from Docker Hub. If you don't have a Docker ID, head over to https://hub.docker.com to create one.
Username: ashupat
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 images
REPOSITORY          TAG             IMAGE ID        CREATED         SIZE
rocker/verose        latest          c41250521d1f    10 days ago     3.34GB
(node1) (local) root@192.168.0.18 ~
$ docker tag c41250521d1f ashupat/verose:newverse
(node1) (local) root@192.168.0.18 ~

```

Command: to push image to docker hub account

docker push ashupat/verose:newverse

note: newverse is tag name created above.

Output:

01:27:07

CLOSE SESSION

Instances

+ ADD NEW INSTANCE

192.168.0.18 node1

192.168.0.17 node2

192.168.0.16 node3

192.168.0.15 node4

192.168.0.14 node5

cikpas4s_cikr7bksnmng00e9i9i0

IP: 192.168.0.18 OPEN PORT

Memory: 87.81% (3.43GiB / 3.906GiB) CPU: 0.20%

SSH: ssh ip172-19-0-11-cikpas4snmng00e9i4c0@direct.labs.play

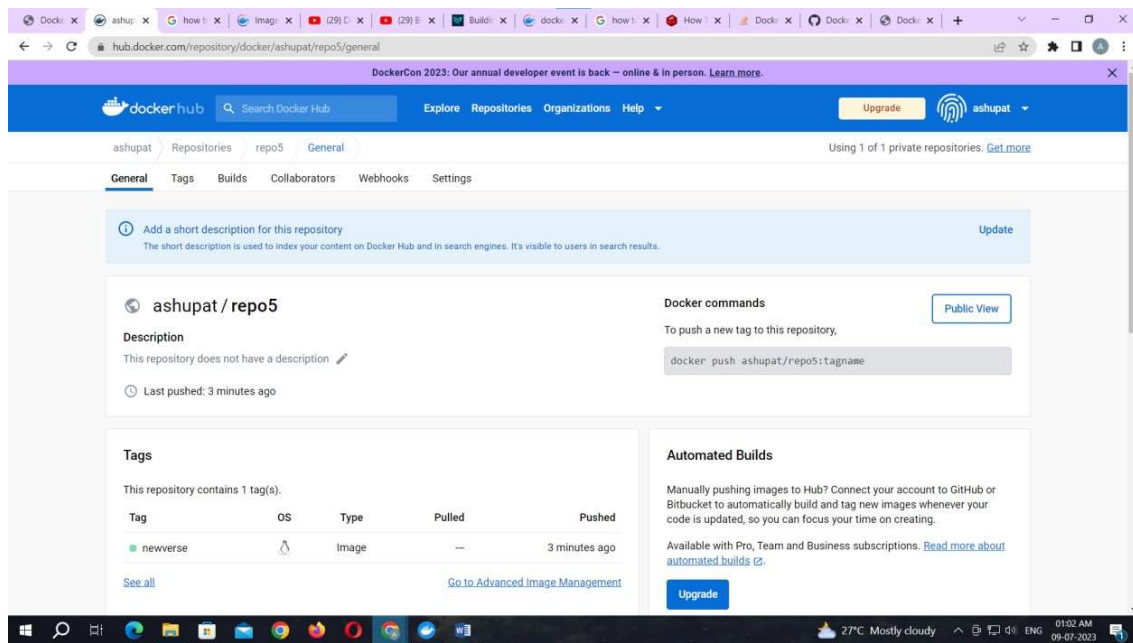
DELETE EDITOR

```

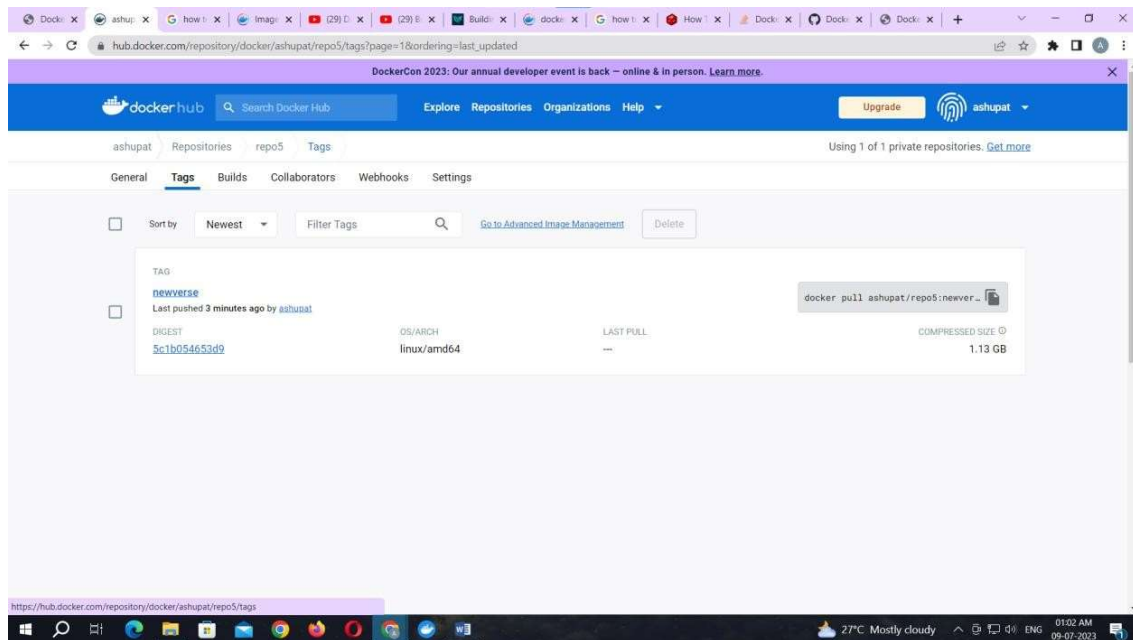
Login Succeeded
(node1) (local) root@192.168.0.18 ~
$ docker images
REPOSITORY          TAG             IMAGE ID        CREATED         SIZE
rocker/verose        latest          c41250521d1f    10 days ago     3.34GB
(node1) (local) root@192.168.0.18 ~
$ docker tag c41250521d1f ashupat/verose:newverse
(node1) (local) root@192.168.0.18 ~
$ docker push ashupat/verose:newverse
The push refers to repository (docker.io/ashupat/verose)
c29e3b74ddcd: Mounted from rocker/verose
edfb61d423ee: Mounted from rocker/verose
01f95d4e8ac3: Mounted from rocker/verose
9b66e6ff4231: Mounted from rocker/verose
4b8a628f14d1: Mounted from rocker/verose
cc29ca15609e: Mounted from rocker/verose
6ab696bfc3fe: Mounted from rocker/verose
06757fd4661: Mounted from rocker/verose
40419120bb3c: Mounted from rocker/verose
cdd7c7392317: Mounted from rocker/verose
newverse: digest: sha256:9c1b084653d9ceaf46720439c1c023edfdb92b251d8dd11a7f7d80d56c541b0 size: 2428
(node1) (local) root@192.168.0.18 ~

```

Check it in docker hub now



Click on tags and check.



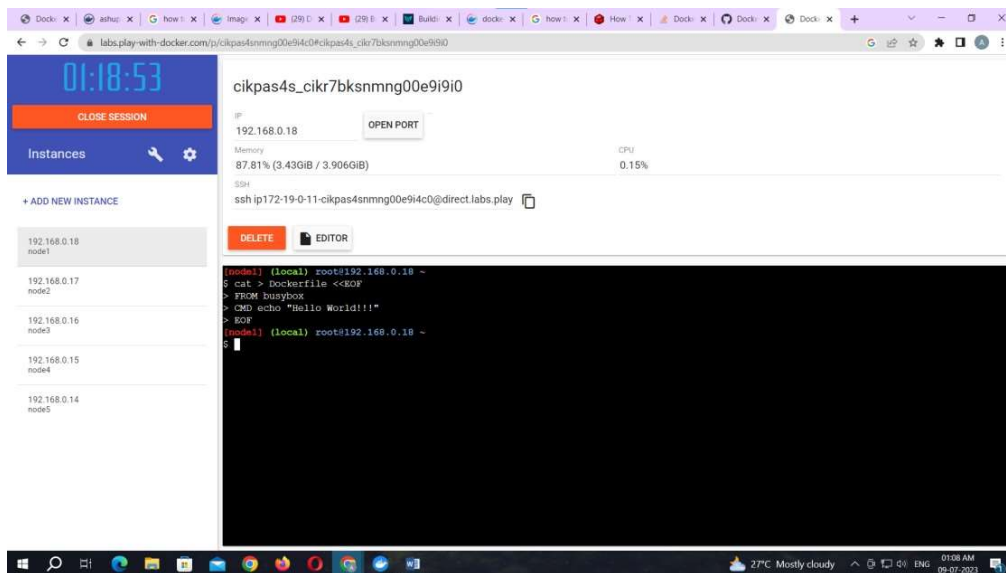
Practical No. 04

Aim: Working with Docker Containers and Commands: Build an image then push it to docker and run it

Command: to create docker file

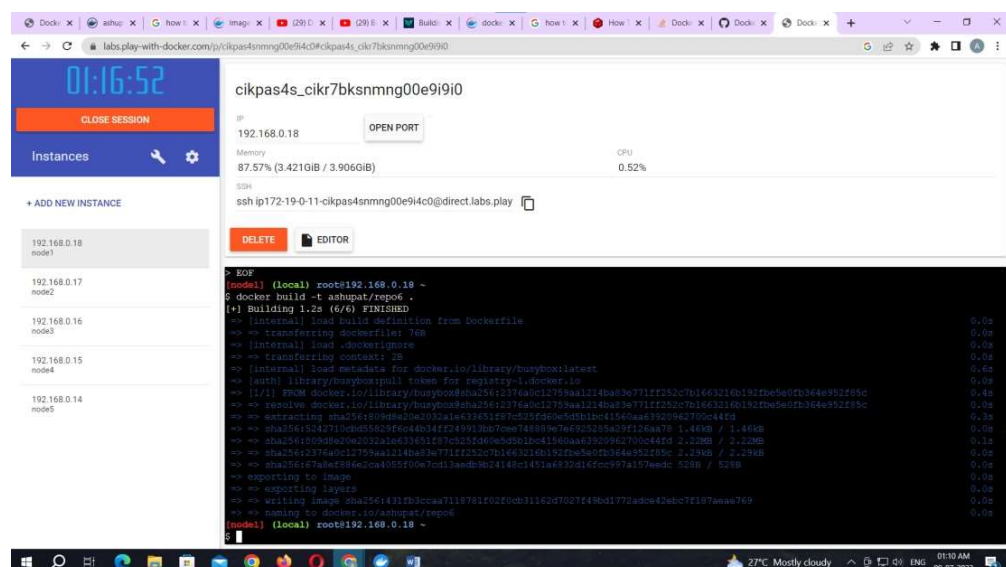
- 1) cat > Dockerfile <<EOF
- 2) FROM busybox
- 3) CMD echo "Hello world! This is my first Docker image."
- 4) EOF

Output:



Command: to build image from docker file
docker build -t ashupat/repo6 .

Output:



Command: to check docker images

docker images

Output:

```

[internal] load metadata for docker.io/library/busybox:latest
[auth] library/busybox:pull token for registry-1.docker.io
[1/1] FROM docker.io/library/busybox:latest
=> resolve docker.io/library/busybox:latest
=> extracting sha256:809d8e0e3032a1e613651f87c525d6d0e3b1bc41560aa63920962700c44fd
=> sha256:809d8e0e3032a1e613651f87c525d6d0e3b1bc41560aa63920962700c44fd 1.46kB / 1.46kB
=> sha256:809d8e0e3032a1e613651f87c525d6d0e3b1bc41560aa63920962700c44fd 3.34kB / 3.34kB
=> sha256:2376a0c12759a1214ba83e771ff252c7b1663216b192f0e50fb364e952f85c 2.29kB / 2.29kB
=> sha256:67aef986a2c4055f0e7cd13aedb9b24148c1451a6832d16fcc997a157eedc 528B / 528B
=> exporting to image
=> exporting layers
=> writing image sha256:431fb3ccaa7118781f02f0cb31162d7027f49bd1772adce42ebc7f187aaae769
=> naming to docker.io/ashupat/rep06
(node1) (local) root@192.168.0.18 ~
$ ^C
(node1) (local) root@192.168.0.18 ~
$ docker images
REPOSITORY          TAG             IMAGE ID        CREATED         SIZE
ashupat/rep06        latest          431fb3ccaa71    10 days ago    4.26MB
ashupat/rep05        newverse        c41250521d1f    10 days ago    3.34GB
rocker/verse         latest          c41250521d1f    10 days ago    3.34GB
(node1) (local) root@192.168.0.18 ~
$
  
```

Command: to tag image

docker tag 431fb3ccaa71 ashupat/rep06:image

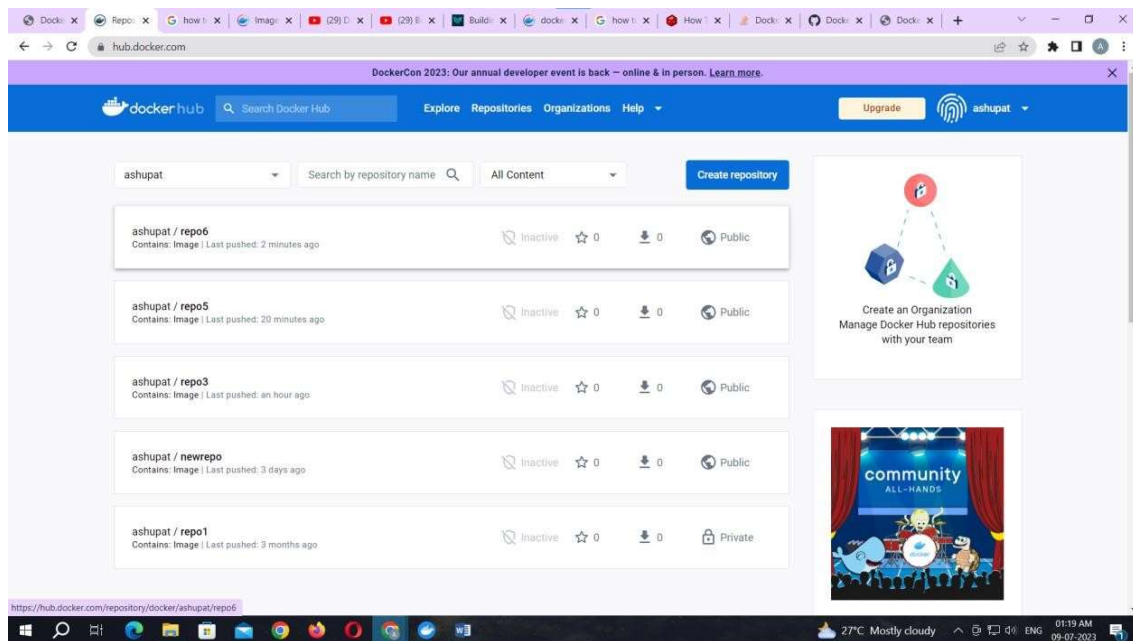
Command: to push image to docker hub

docker push ashupat/rep06:image

```

=> sha256:67aef986a2c4055f0e7cd13aedb9b24148c1451a6832d16fcc997a157eedc 528B / 528B
=> exporting to image
=> exporting layers
=> writing image sha256:431fb3ccaa7118781f02f0cb31162d7027f49bd1772adce42ebc7f187aaae769
=> naming to docker.io/ashupat/rep06
(node1) (local) root@192.168.0.18 ~
$ ^C
(node1) (local) root@192.168.0.18 ~
$ docker images
REPOSITORY          TAG             IMAGE ID        CREATED         SIZE
ashupat/rep06        latest          431fb3ccaa71    10 days ago    4.26MB
ashupat/rep05        newverse        c41250521d1f    10 days ago    3.34GB
rocker/verse         latest          c41250521d1f    10 days ago    3.34GB
(node1) (local) root@192.168.0.18 ~
$ docker tag 431fb3ccaa71 ashupat/rep06:image
(node1) (local) root@192.168.0.18 ~
$ docker push ashupat/rep06:image
The push refers to repository [docker.io/ashupat/rep06]
feb4513d4fb7: Mounted from library/busybox
image: digest: sha256:02a1f0a06a6ebec58b5034e7056d1e0794e4c6191884b79131b221be9f19e size: 527
(node1) (local) root@192.168.0.18 ~
$
  
```

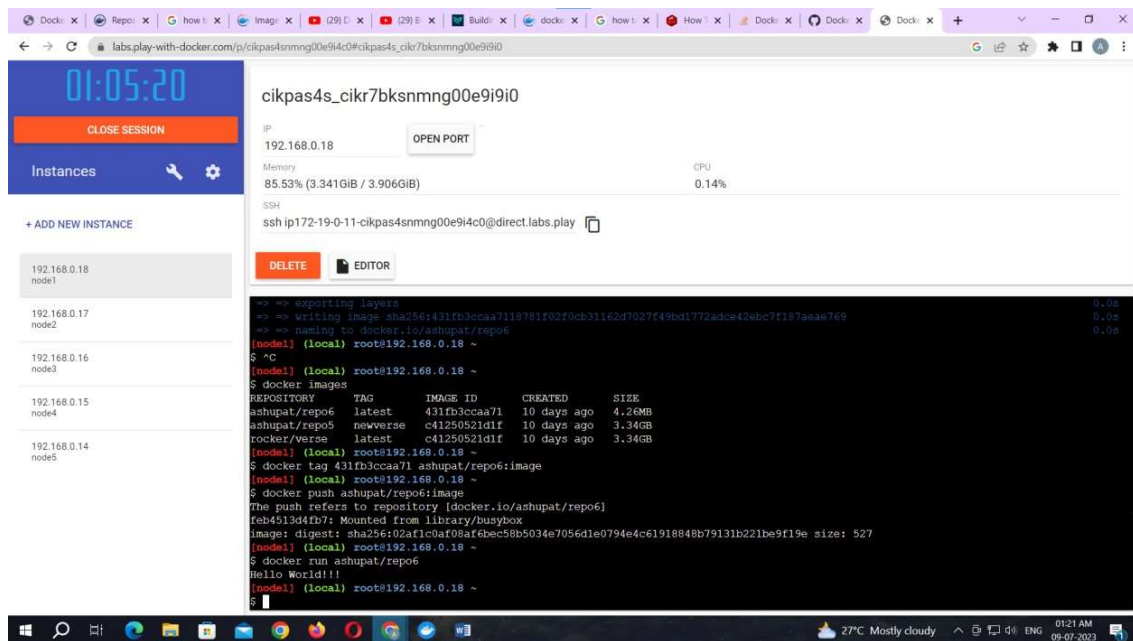

Now check it on docker hub.



Command: to run docker image:

`docker run kbdocker11/repo2`

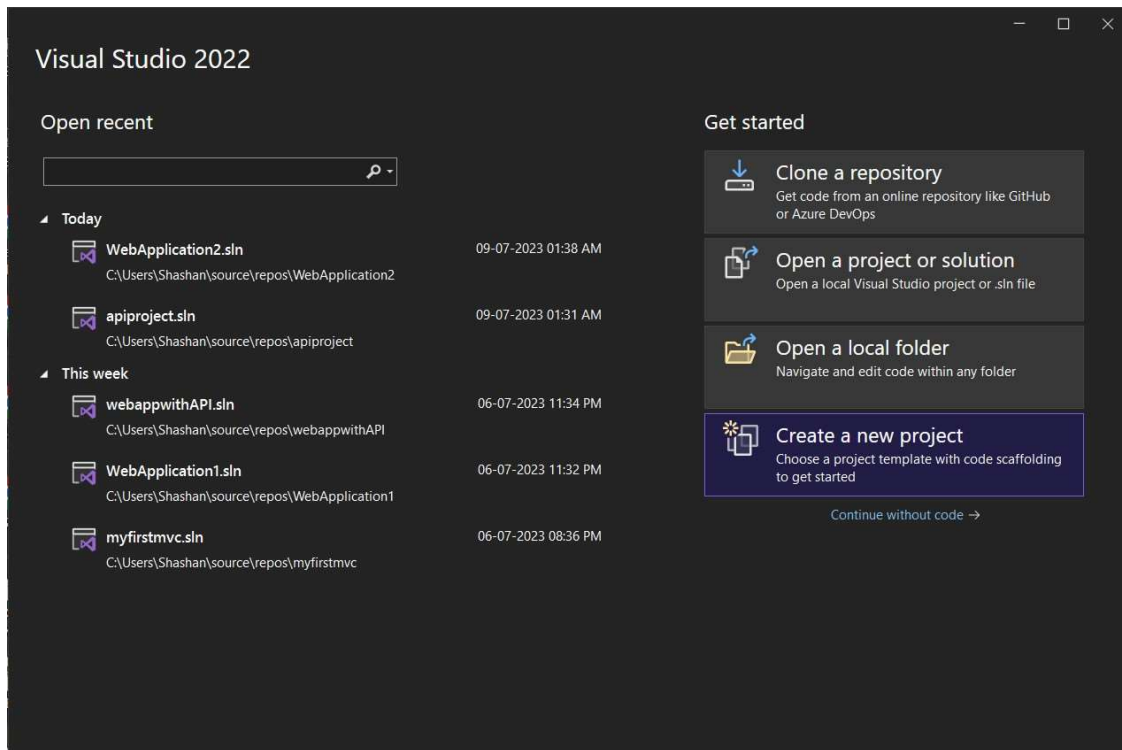
Output:



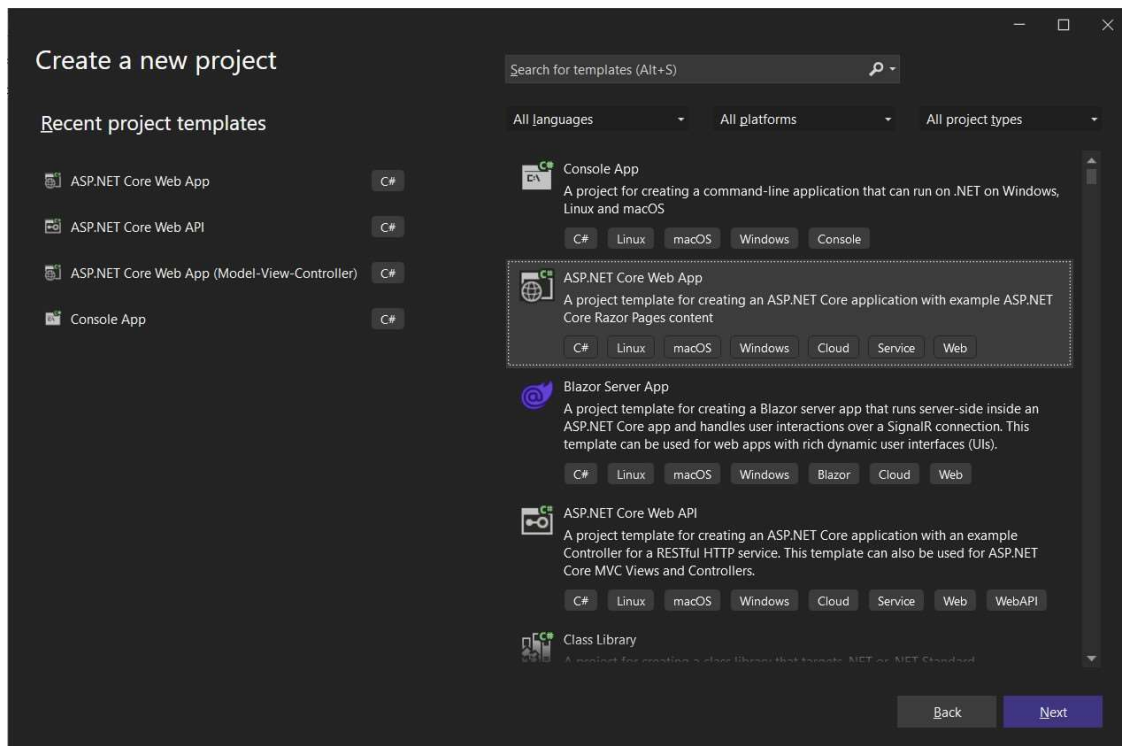
Practical No. 05

Aim: Build a Web App and publish it to Docker.

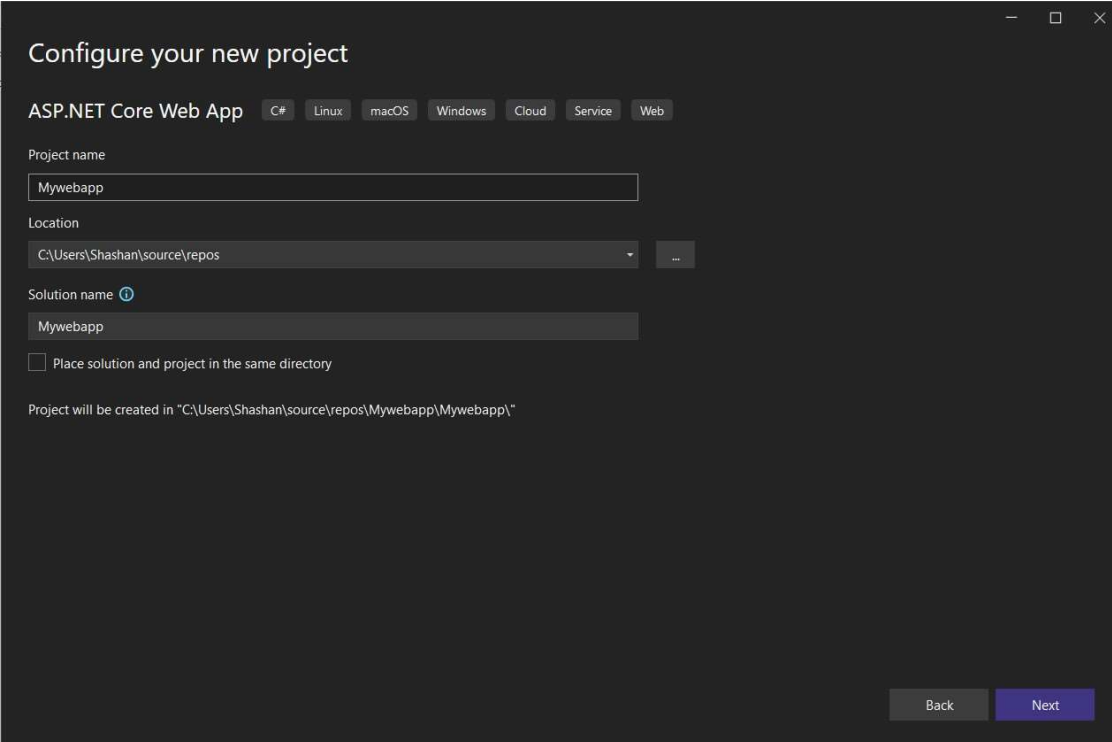
Let's start by creating a new project in the **Microsoft Visual Studio**



Select the **ASP.NET Core Web App** form the various options.



Give an appropriate name to the project.



Configure your new project

ASP.NET Core Web App C# Linux macOS Windows Cloud Service Web

Project name
Mywebapp

Location
C:\Users\Shashan\source\repos

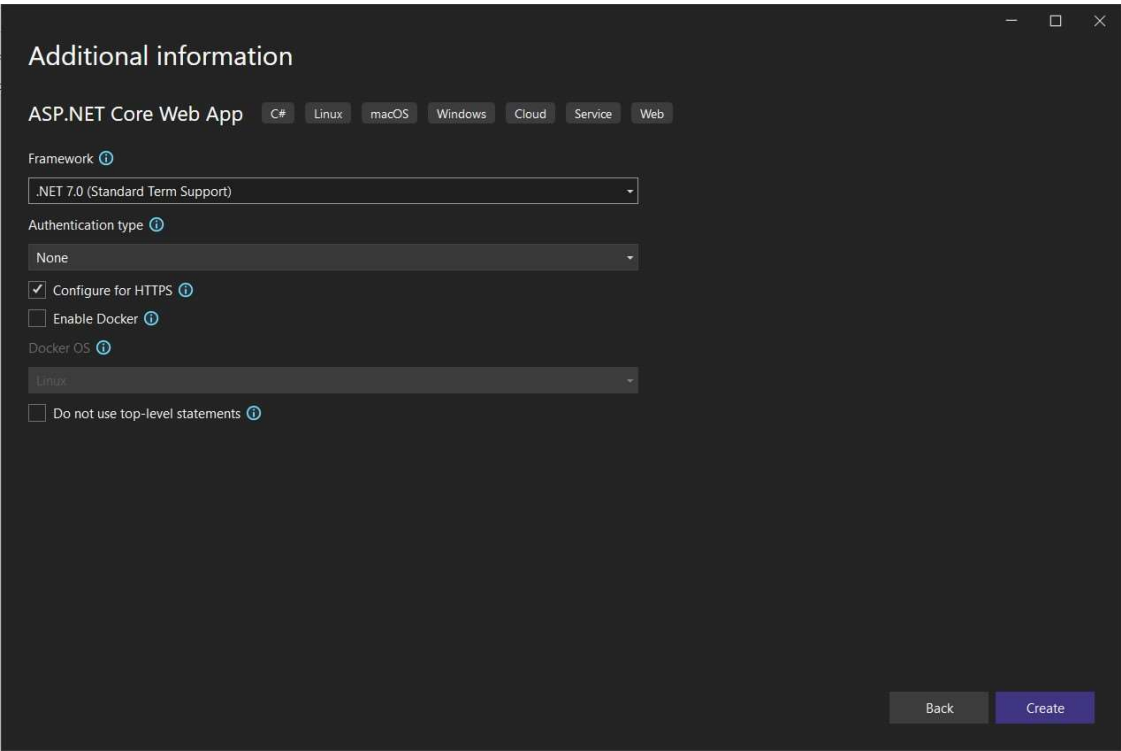
Solution name ⓘ
Mywebapp

☐ Place solution and project in the same directory

Project will be created in "C:\Users\Shashan\source\repos\Mywebapp\Mywebapp\."

Back Next

Click on the **Create** button.



Additional information

ASP.NET Core Web App C# Linux macOS Windows Cloud Service Web

Framework ⓘ
.NET 7.0 (Standard Term Support)

Authentication type ⓘ
None

☒ Configure for HTTPS ⓘ

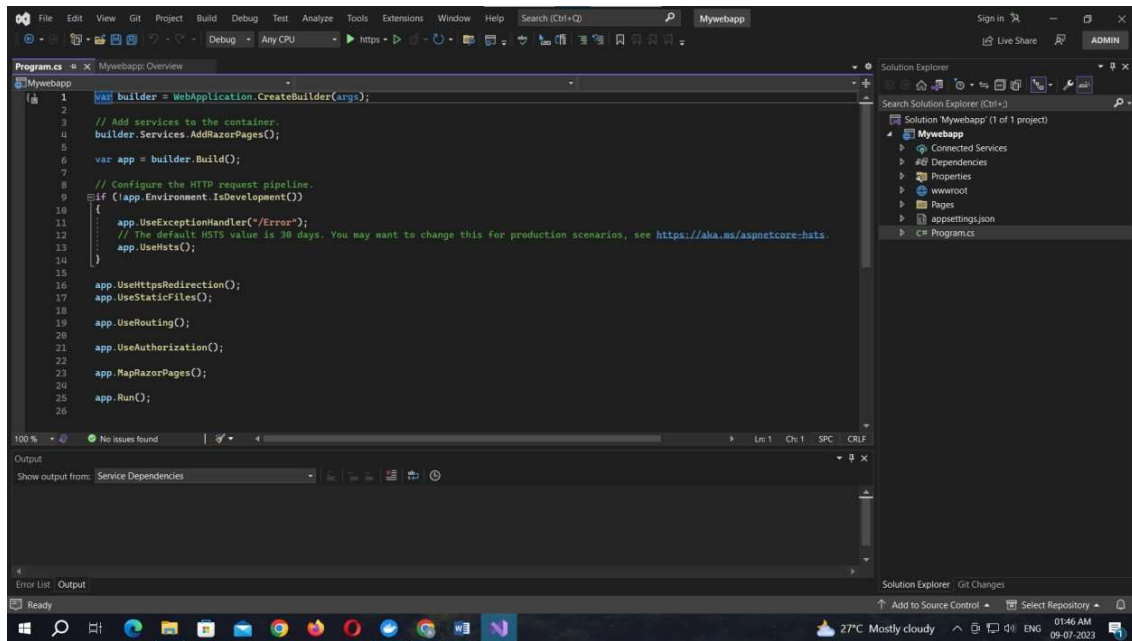
☐ Enable Docker ⓘ

Docker OS ⓘ
Linux

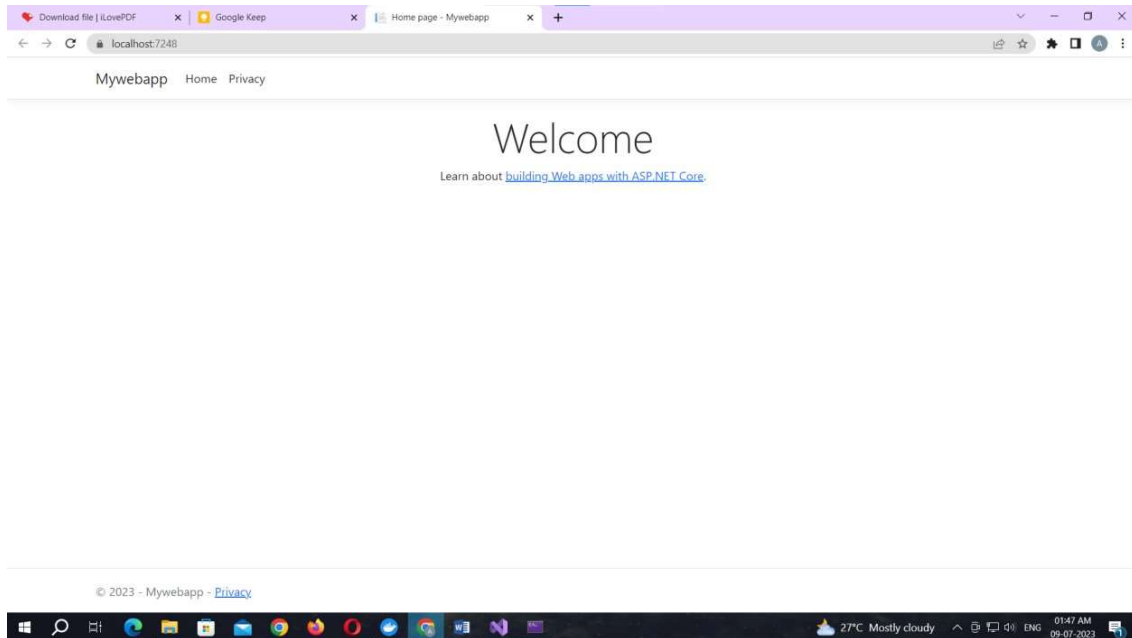
☐ Do not use top-level statements ⓘ

Back Create

Once the project is created, Try executing the project to make sure that it works.

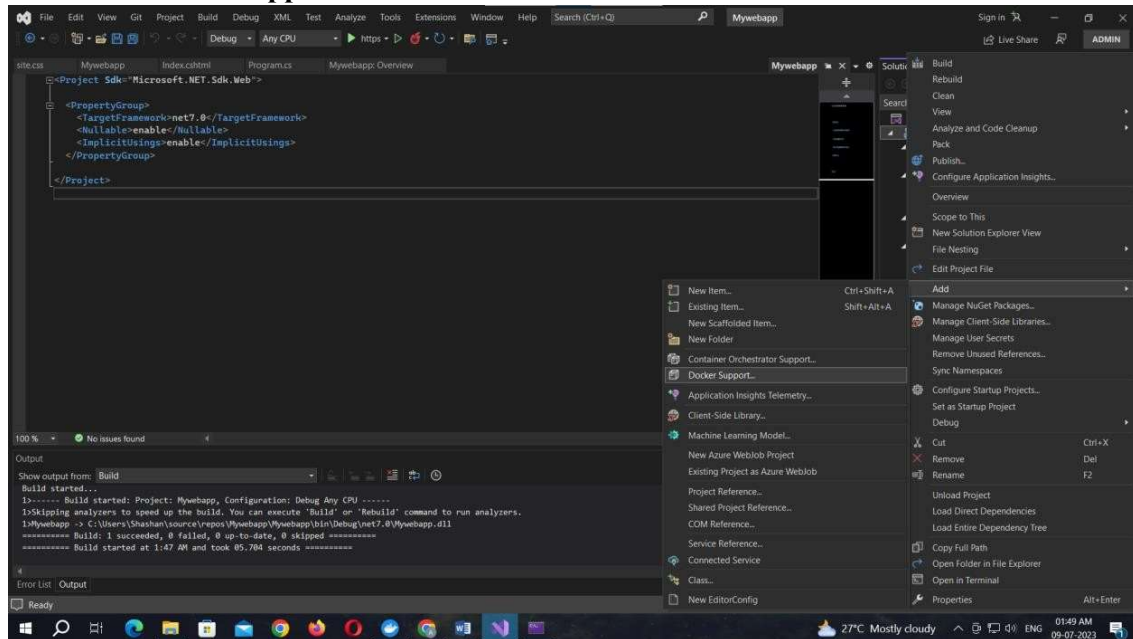


Output:

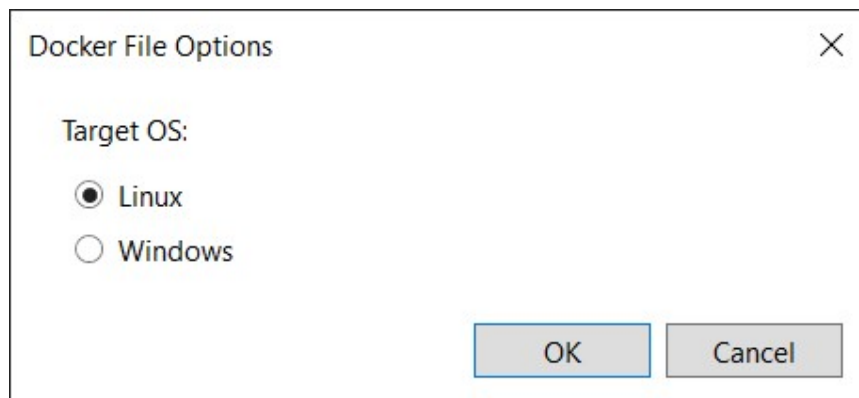


Right click on the app name then select the **Add** option form the menu.

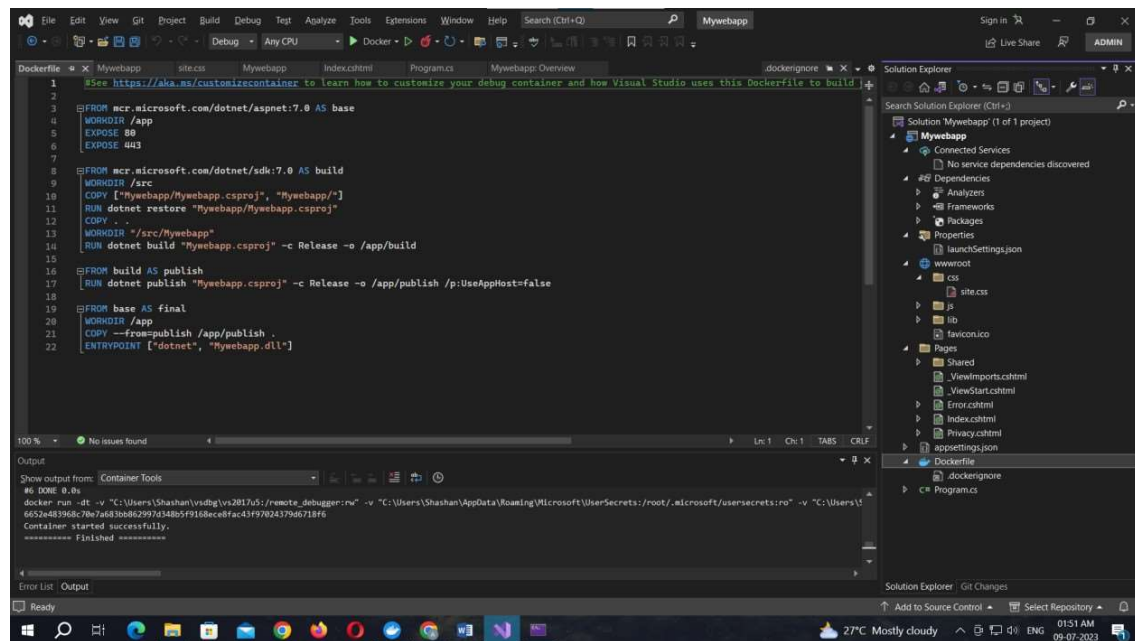
Select the **Docker Support** under the **Add** menu



Select the Target OS as Linux and click OK button.



The file naming Dockerfile will appear



Code for Dockerfile:

#See <https://aka.ms/customizecontainer> to learn how to customize your debug container and how Visual Studio uses this Dockerfile to build your images for faster debugging.

```

FROM mcr.microsoft.com/dotnet/aspnet:7.0 AS base
WORKDIR /app
EXPOSE 80
EXPOSE 443
  
```

```

FROM mcr.microsoft.com/dotnet/sdk:7.0 AS build
WORKDIR /src
COPY ["Mywebapp/Mywebapp.csproj", "Mywebapp/"]
RUN dotnet restore "Mywebapp/Mywebapp.csproj"
COPY . .
WORKDIR "/src/Mywebapp"
RUN dotnet build "Mywebapp.csproj" -c Release -o /app/build
  
```

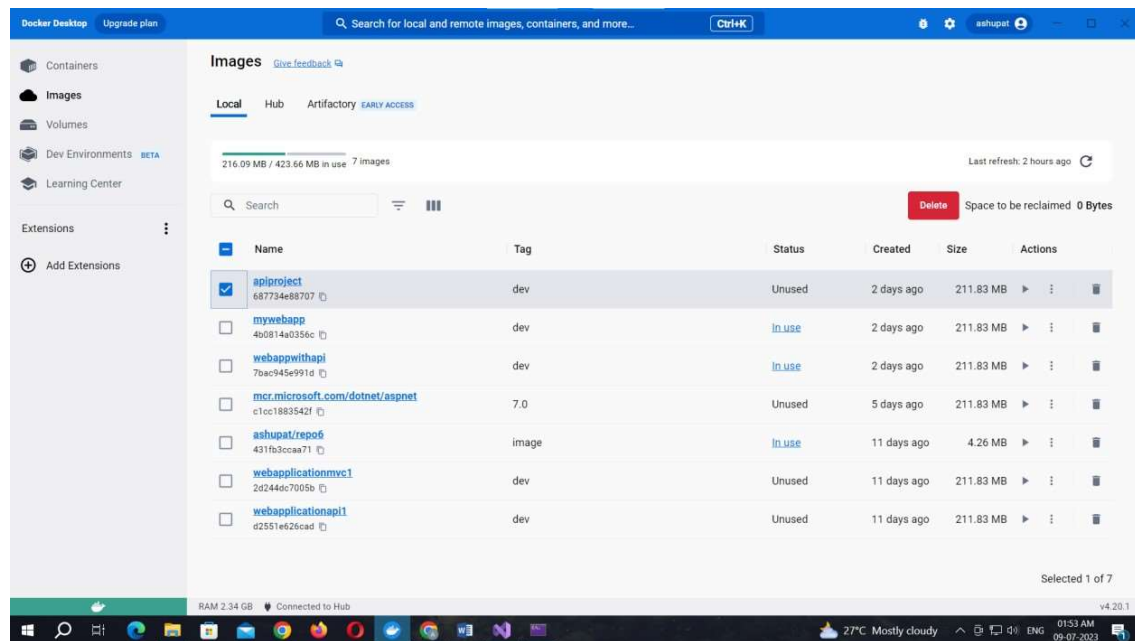
```

FROM build AS publish
RUN dotnet publish "Mywebapp.csproj" -c Release -o /app/publish /p:UseAppHost=false
  
```

```

FROM base AS final
WORKDIR /app
COPY --from=publish /app/publish .
ENTRYPOINT ["dotnet", "Mywebapp.dll"]
  
```

The project is uploaded to the docker.

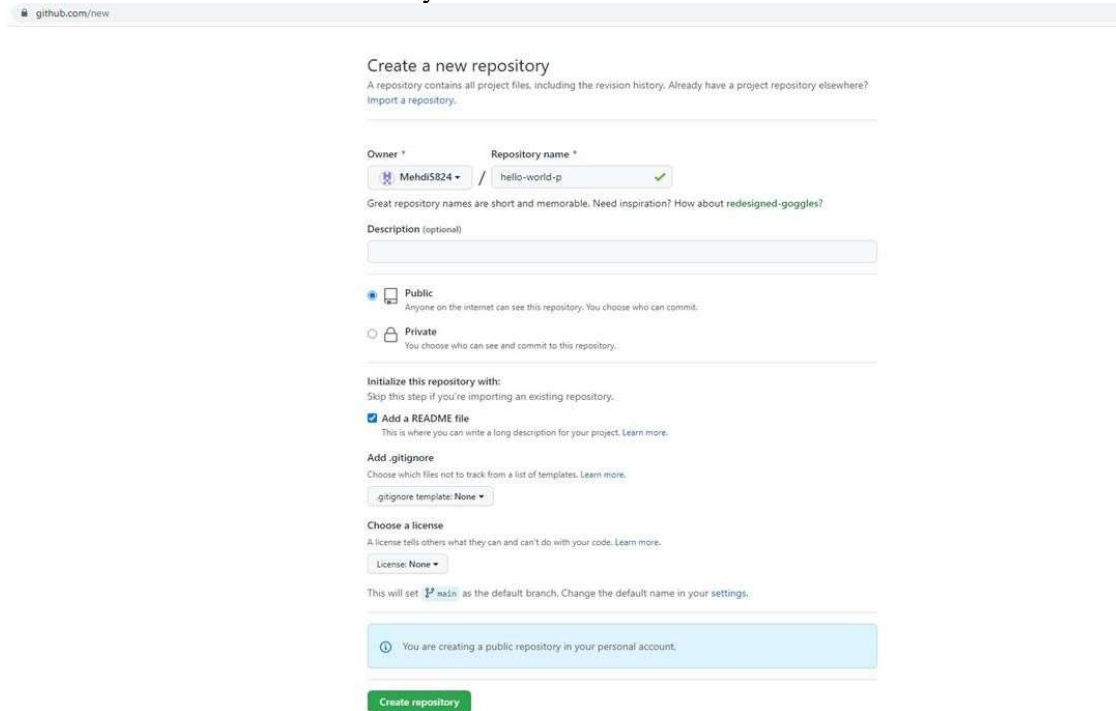


Practical No. 06

Aim: Working with the CircleCI

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.



github.com/new

Create a new repository

A repository contains all project files, including the revision history. Already have a project repository elsewhere? [Import a repository.](#)

Owner * Repository name *

Great repository names are short and memorable. Need inspiration? [How about redesigned-goggles?](#)

Description (optional)

☒ Public
Anyone on the internet can see this repository. You choose who can commit.

☐ Private
You choose who can see and commit to this repository.

Initialize this repository with:
Skip this step if you're importing an existing repository.

☒ Add a README file
This is where you can write a long description for your project. [Learn more.](#)

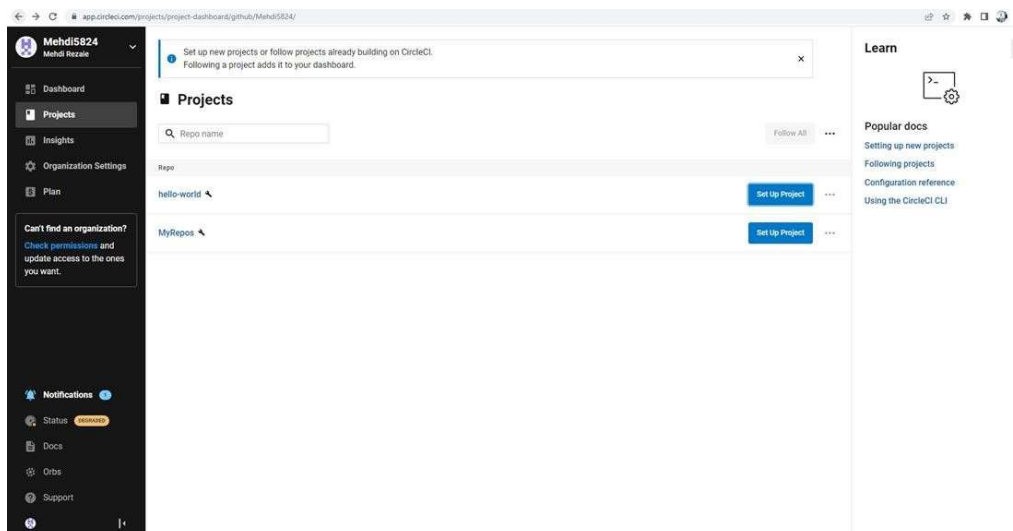
Add .gitignore
Choose which files not to track from a list of templates. [Learn more.](#)

Choose a license
A license tells others what they can and can't do with your code. [Learn more.](#)

This will set [main](#) as the default branch. Change the default name in your settings.

[Create repository](#)

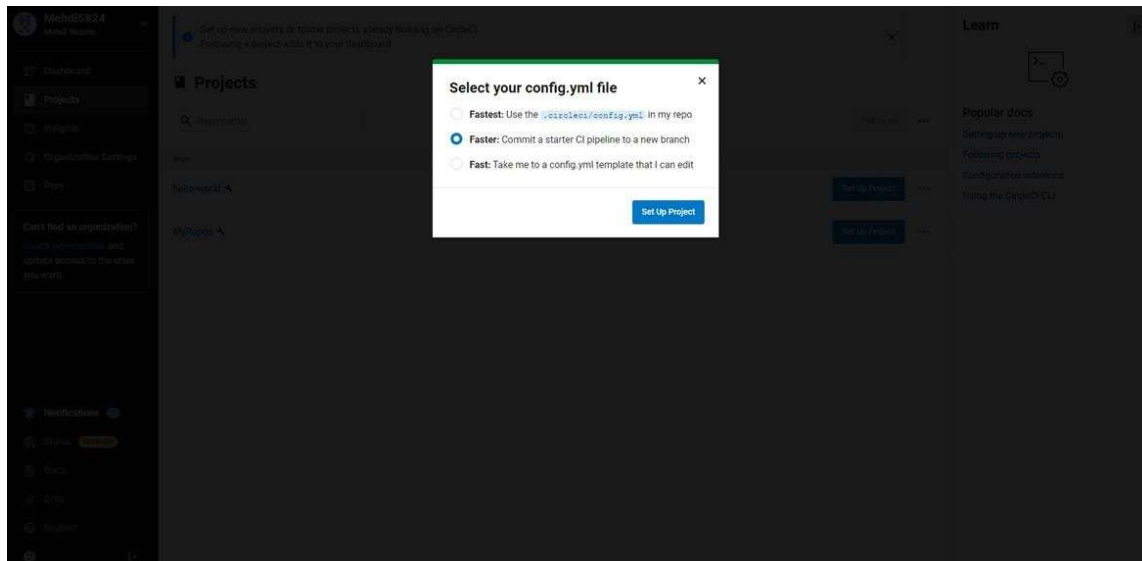
Login to Circle CI <https://app.circleci.com/> Using GitHub Login, Once logged in navigate to Projects.



Set up CircleCI

1. Navigate to the CircleCI Projects page. If you created your new repository under an organization, you will need to select the organization name.
2. You will be taken to the Projects dashboard. On the dashboard, select the project you want to set up (hello-world).
3. Select the option to commit a starter CI pipeline to a new branch, and click Set Up Project. This will create a file

.circleci/config.yml at the root of your repository on a new branch called circleci-project-setup.



Your first Pipeline

On your project's pipeline page, click the green Success button, which brings you to the workflow that ran (say-hello- workflow).

Within this workflow, the pipeline ran one job, called say-hello. Click say-hello to see the steps in this job:

- a. Spin up environment
- b. Preparing environment variables
- c. Checkout code
- d. Say hello

Now select the "say-hello-workflow" to the right of Success status column

The screenshot shows the CircleCI dashboard for user Mehdi5824. The left sidebar contains navigation links: Dashboard, Projects, Insights, Organization Settings, and Plan (UPGRADE). The main content area displays the 'hello-world' project pipeline. A table lists the pipeline 'hello-world 1' with a 'Success' status, workflow 'say-hello-workflow', branch 'circleci-project-setup', and commit '6b308f0'. The duration is 3m ago. Below the table, there's a section for 'say-hello-workflow' with a 'Success' status and a '5s' duration.

Select “say-hello” Job with a green tick

The screenshot shows the CircleCI dashboard for user Mehdi5824, specifically the 'say-hello-workflow' job details. The job is in a 'Success' state. The duration is 5s / 8m ago. The branch is 'circleci-project-setup' and the commit is '6b308f0'. The author is 'Mehdi5824'. Below the job details, there's a section for 'say-hello' with a '3s' duration. At the bottom, there's a banner with a lightbulb icon and text: 'Did you know? CircleCI teams that commit 4x as often fix failed builds 2x faster.' It also shows a progress bar for '20 pipelines/day' and '70 min to recovery'.

The screenshot shows the CircleCI dashboard for user Mehdi5824. The left sidebar contains navigation links: Dashboard, Projects, Insights, Organization Settings, and Plan. A notification banner asks to check permissions. The main area displays the 'say-hello' workflow run, which is successful. It shows a duration of 2s, finished 9m ago, and was executed on a Docker / Large resource class. The workflow steps are listed as follows:

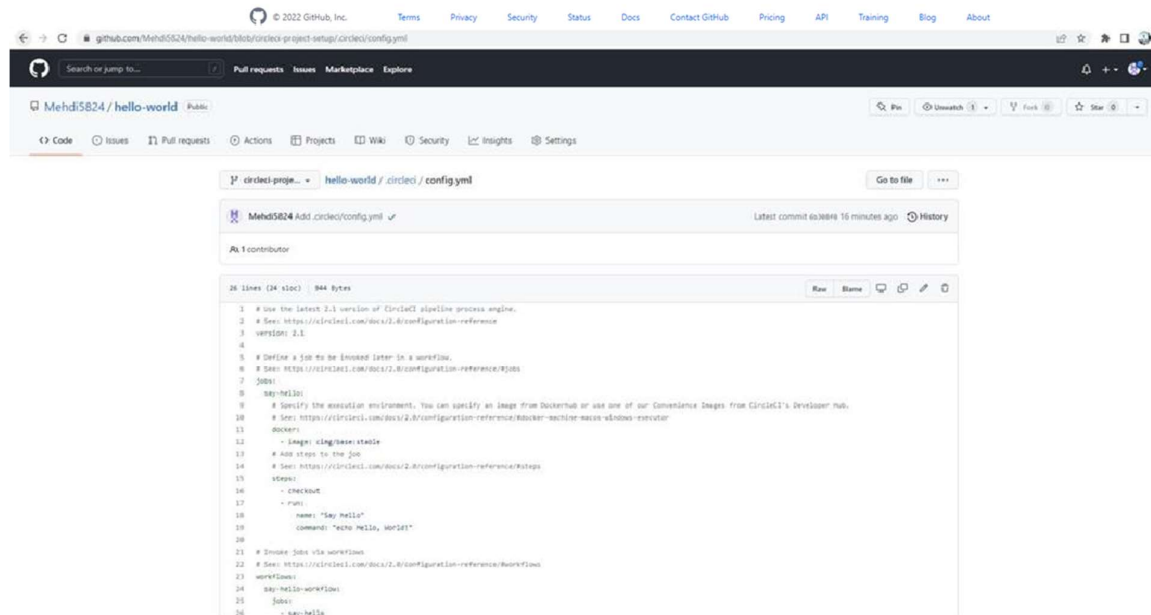
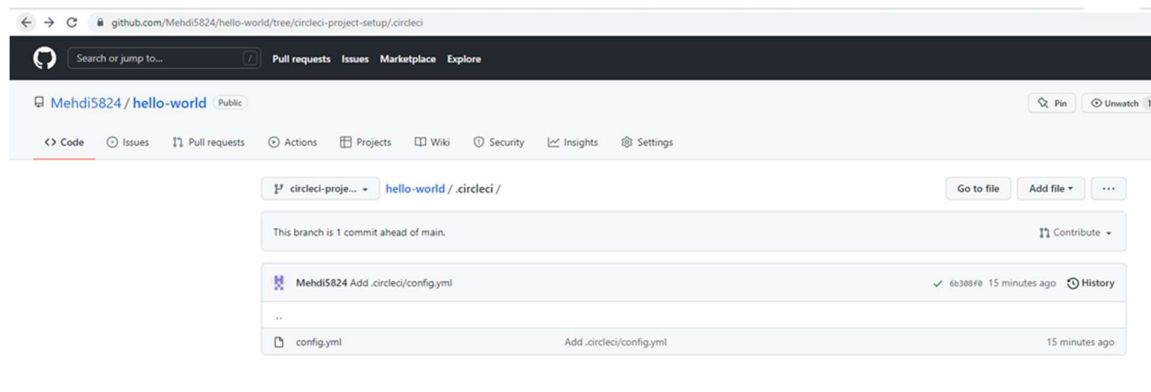
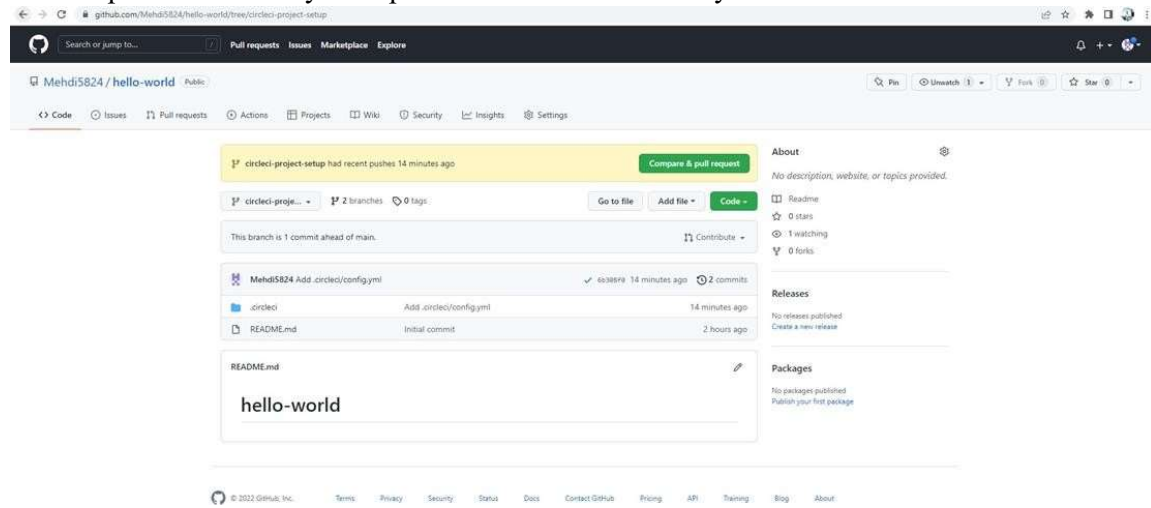
| Step | Duration | Status |
|---------------------------------|----------|---------|
| Spin up environment | 1s | Success |
| Preparing environment variables | 0s | Success |
| Checkout code | 0s | Success |
| Say hello | 0s | Success |

Select Branch and option circleci-project-setup

The screenshot shows the GitHub repository page for Mehdi5824/hello-world. The 'circleci-project-setup' branch is selected in the 'Switch branches/tags' dropdown menu. The repository has 2 branches and 0 tags. The 'circleci-project-setup' branch was created 2 hours ago and has 1 commit. The repository is public and has no releases or packages published.

Break your Build

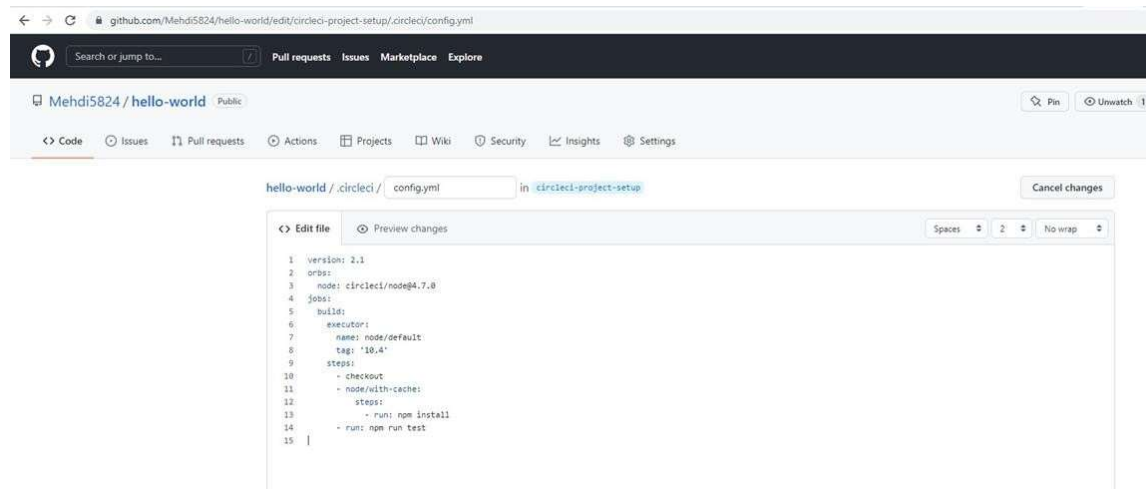
In this section, you will edit the `.circleci/config.yml` file and see what happens if a build does not complete successfully. It is possible to edit files directly on GitHub.



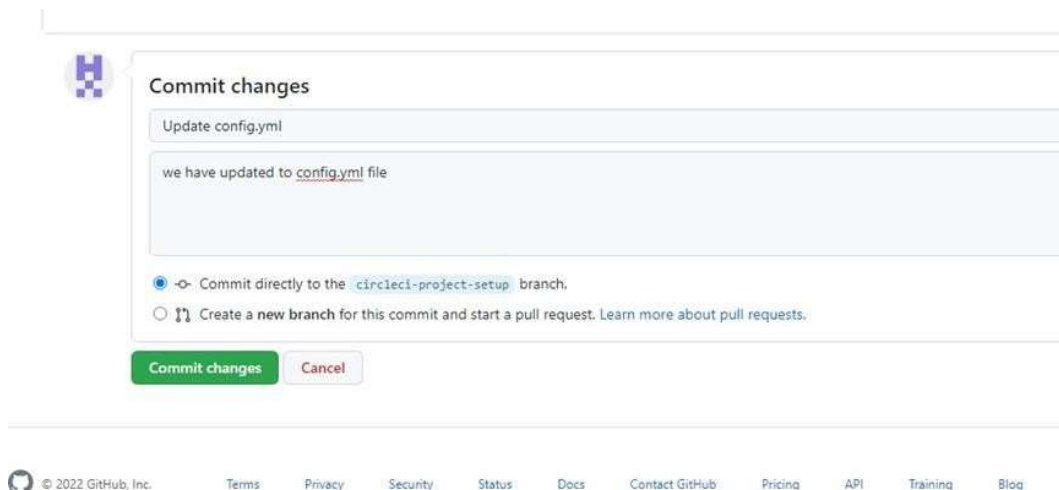
Let's use the Node orb. Replace the existing config by pasting the following code:

```
version: 2.1
orbs:
  node: circleci/node@4.7.0
jobs:
  build:
    executor:
      name: node/default
      tag: '10.4'
    steps:
      - checkout
      - node/with-cache:
          steps:
            - run: npm install
            - run: npm run test
```

The GitHub file editor should look like this



Scroll down and Commit your changes on GitHub



After committing your changes, then return to the Projects page in CircleCI. You should see a new pipeline running... and it will fail! What's going on? The Node orb runs some common Node tasks. Because you are working with an empty repository, running `npm run test`, a Node script, causes the configuration to fail. To fix this, you need to set up a Node project in your repository.

The screenshot displays the CircleCI dashboard for a user named Mehdi Rezaie. The left sidebar contains navigation links: Dashboard, Projects, Insights, Organization Settings, and Plan. A message states, "Can't find an organization? Check permissions and update access to the ones you want." Below this are links for Notifications, Status, Docs, Orbs, and Support. The main content area is titled "All Pipelines" and shows a table of pipeline runs. The table has columns for Pipeline, Status, Workflow, Branch / Commit, Start, Duration, and Actions. Two pipelines are listed: "hello-world 2" which failed with an error "Error calling workflow: 'workflow'Error calling job: 'build'Cannot find a definition for command named node/with-cache", and "hello-world 1" which succeeded with the workflow "say-hello-workflow".

| Pipeline | Status | Workflow | Branch / Commit | Start | Duration | Actions |
|---------------|---------|--------------------|---|---------|----------|---------|
| hello-world 2 | Failed | Build Error | circleci-project-setup ca5ab74 Update config.yml | 3m ago | 6s | ... |
| hello-world 1 | Success | say-hello-workflow | circleci-project-setup 6b308f0 | 22m ago | 14s | ... |

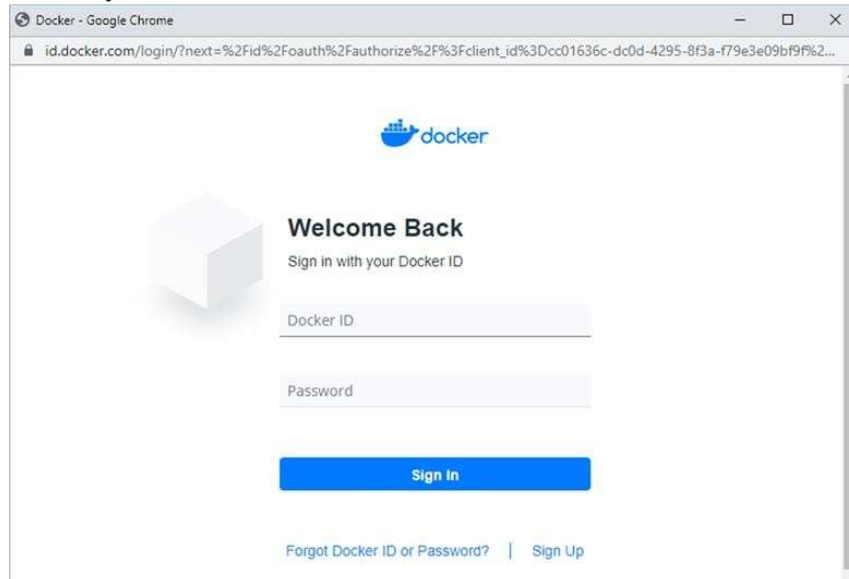
No more pipelines to load

Practical No. 07

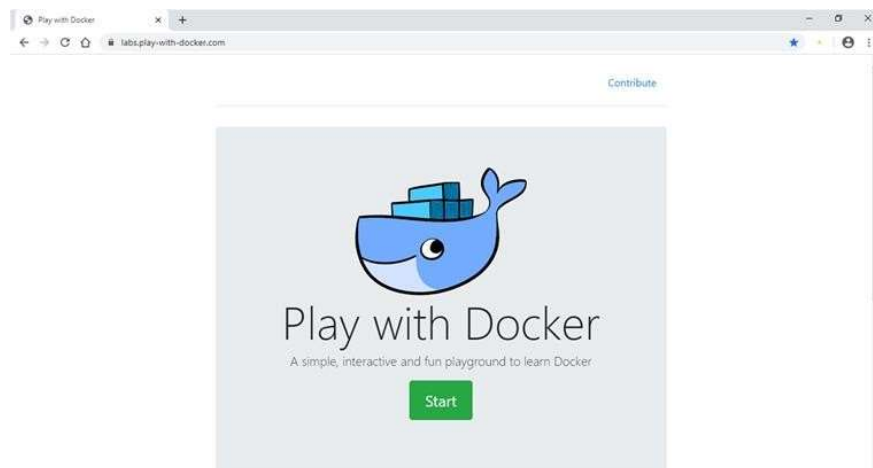
Aim: Running location service in Docker.

(Create docker hub login first to use it in play with docker)

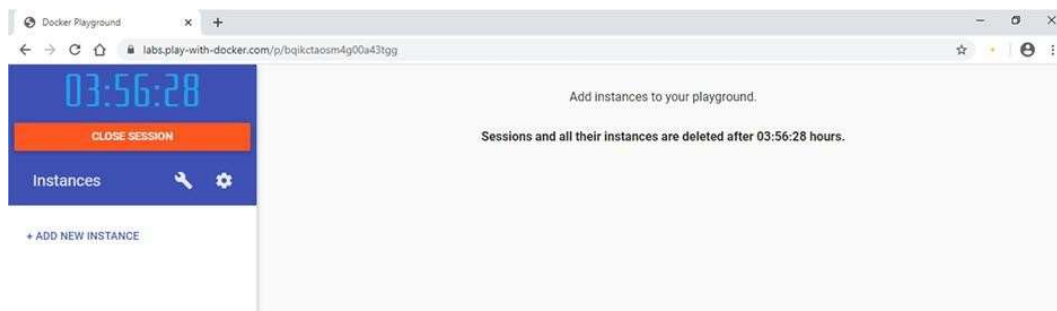
Now login in to Play-With-Docker

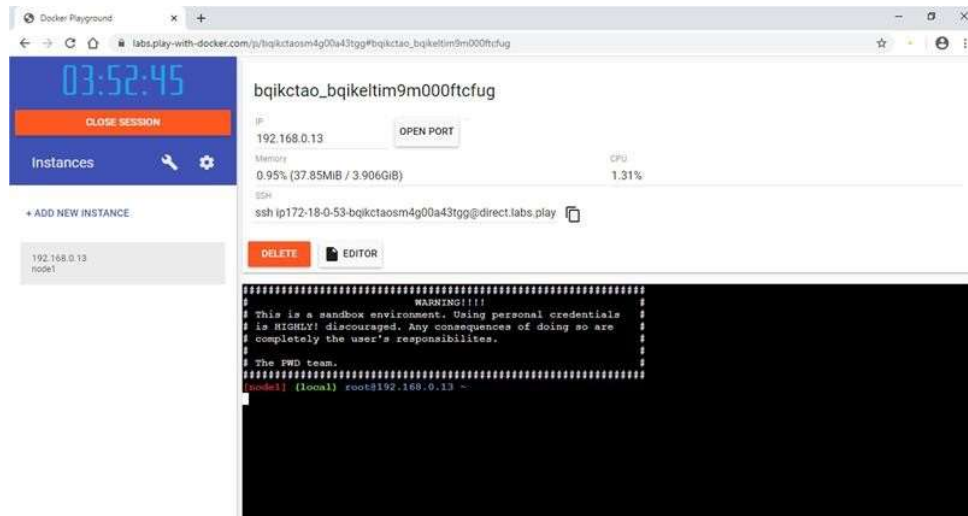


Click on Start.



Click on Add New Instance.



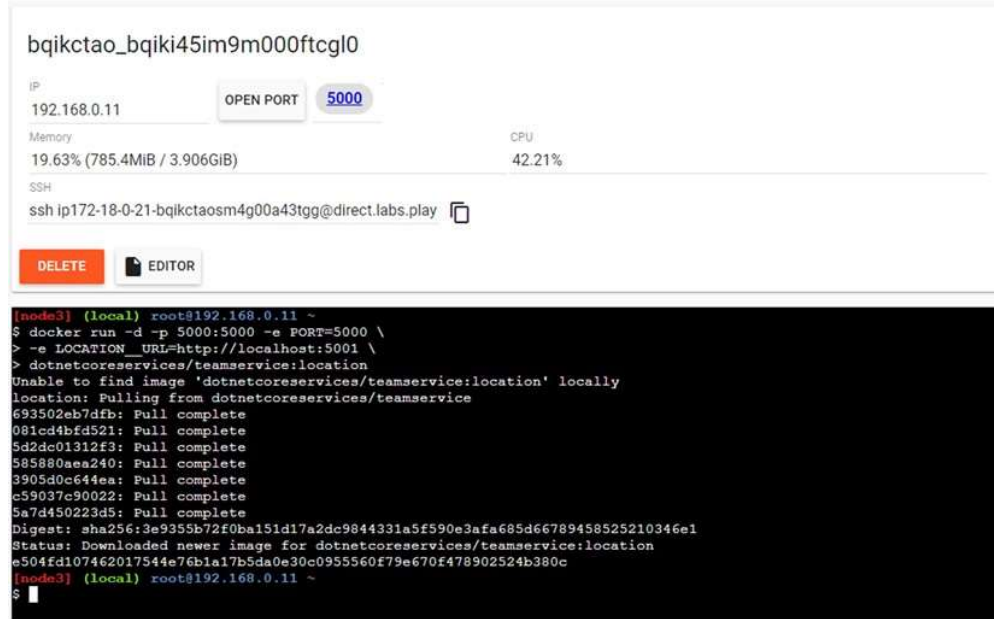


Start typing following commands

Command: To run teamservice

```
docker run -d -p 5000:5000 -e PORT=5000 \
-e LOCATION_URL=http://localhost:5001 \
dotnetcoreservices/teamservice:location
```

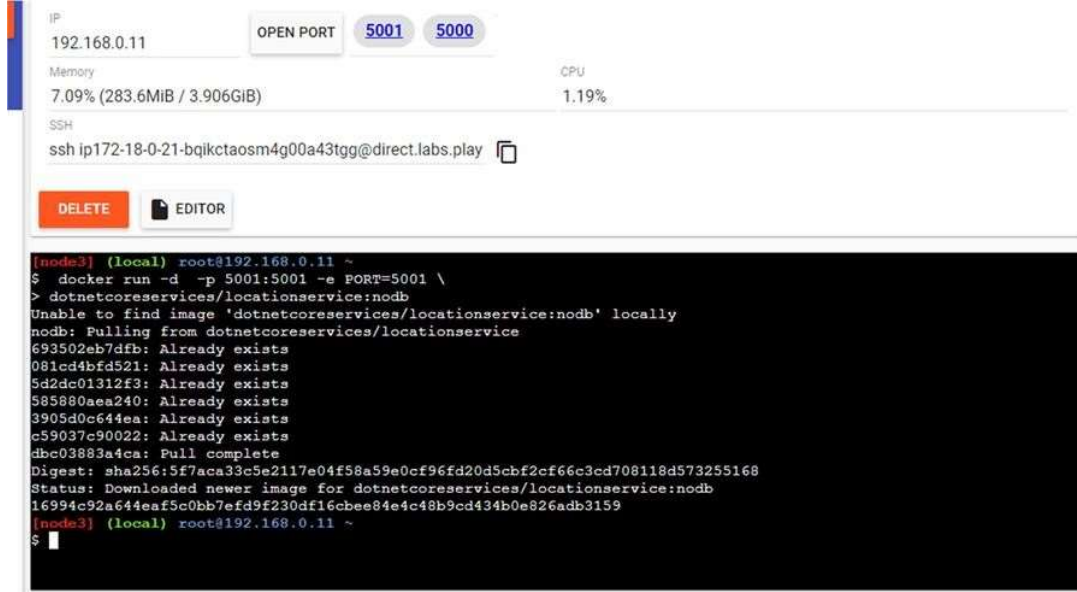
Output: (You can observe that it has started port 5000 on top)



Command: to run location service

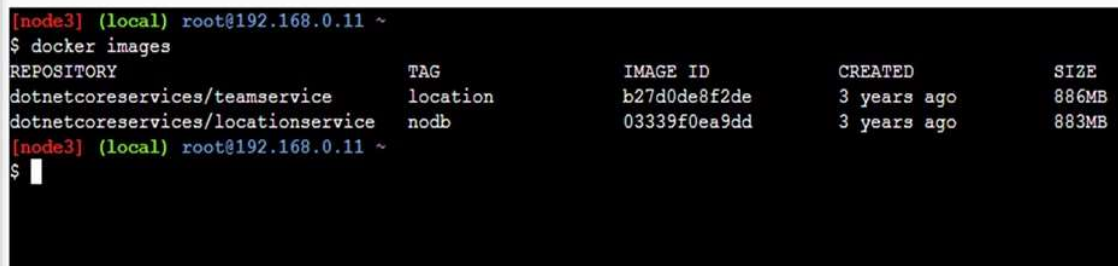
```
docker run -d -p 5001:5001 -e PORT=5001 \
dotnetcoreservices/location:location
```

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

```
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":"Monte",
"lastName":"Carlo"}' 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-9349-3c8593ac8292
{"id":"64c3e69f-1580-4b2f-a9ff-2c5f3b8f0e1f","latitude":12.0,"longitude":12.0,"altitude":10.0,"timestamp":0,"memberId":"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,"memberId":"63e7acf8-8fae-42ce-9349-3c8593ac8292"}] [node1] (local) root@192.168.0.23 ~
$
```