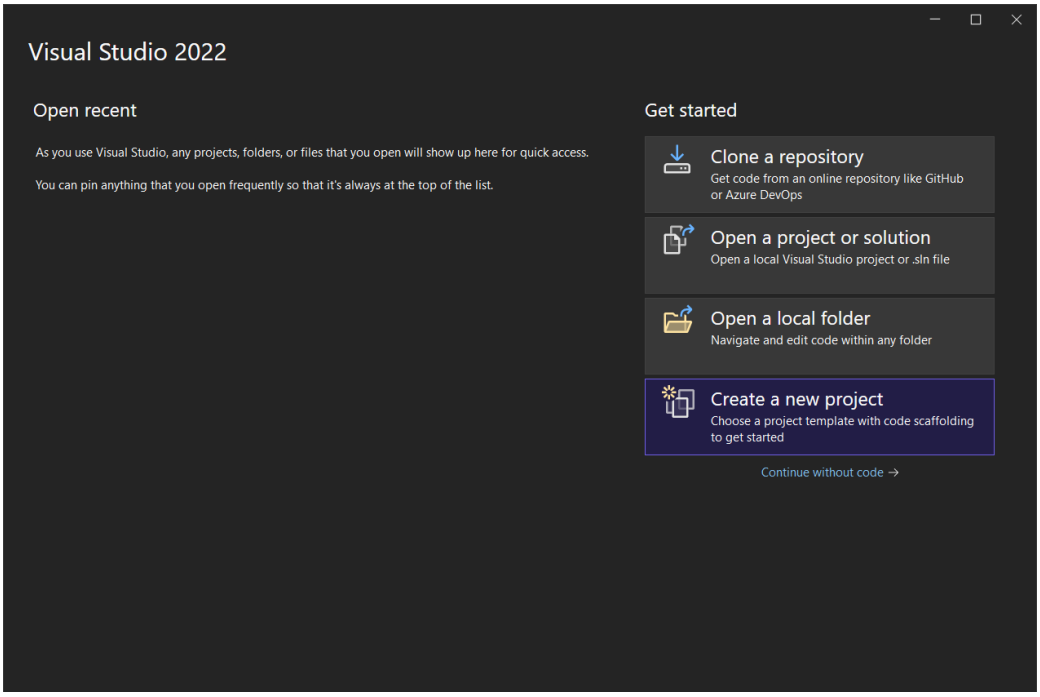
# Practical No. 01

**Aim: Build a Web Application using MVC.**

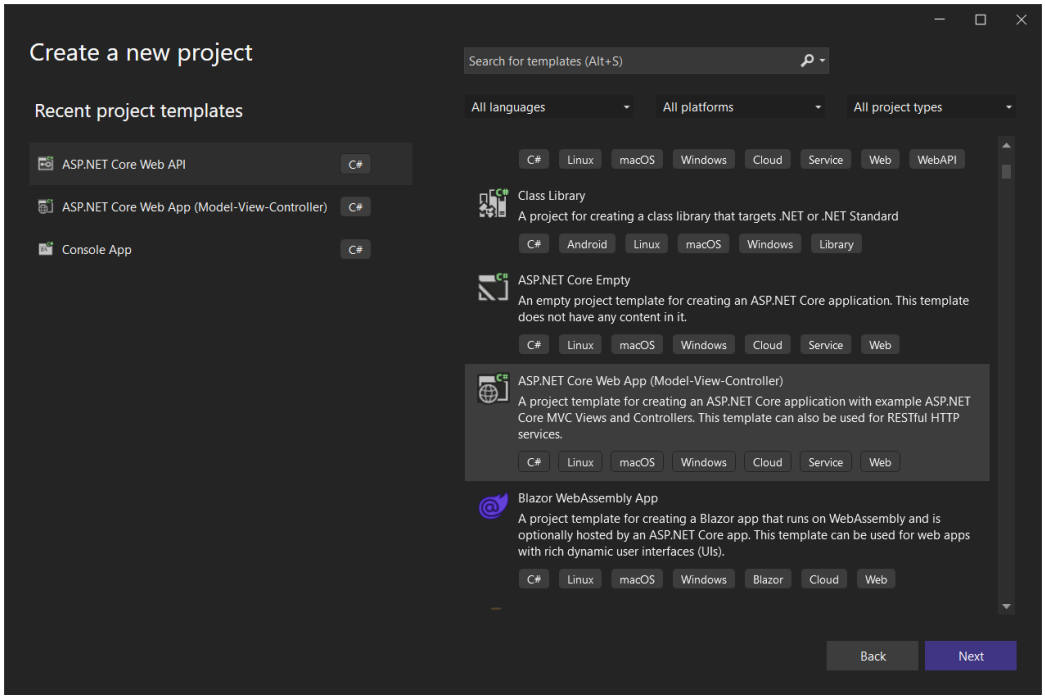
Firstly download, install & open **Microsoft Visual Studio 2022**

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

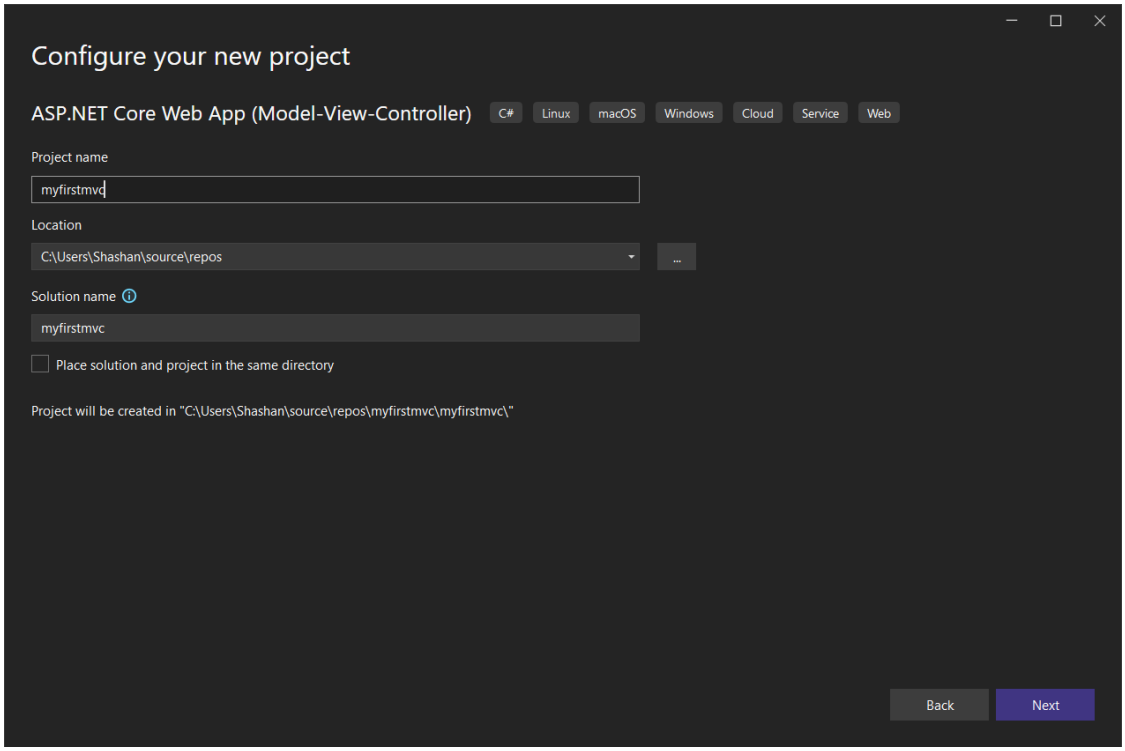
Create a new project:

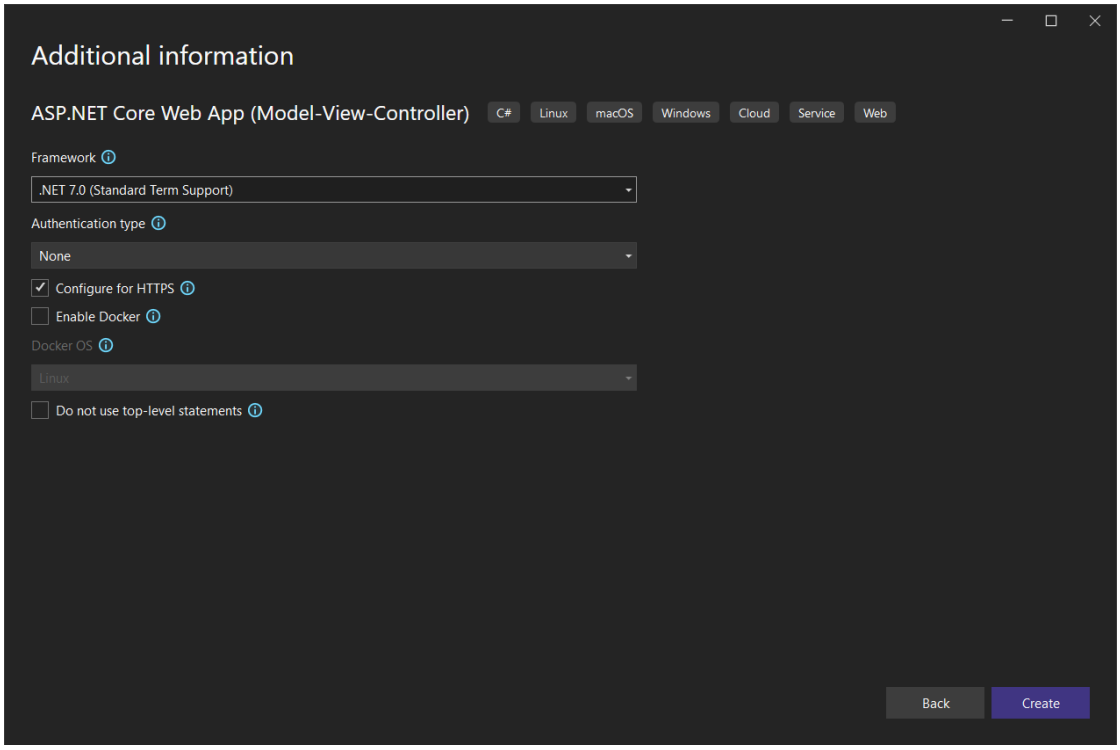



Select the **ASP.NET Core Web App (Model-View_Controller)**



Provide a name to the project:



**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:**

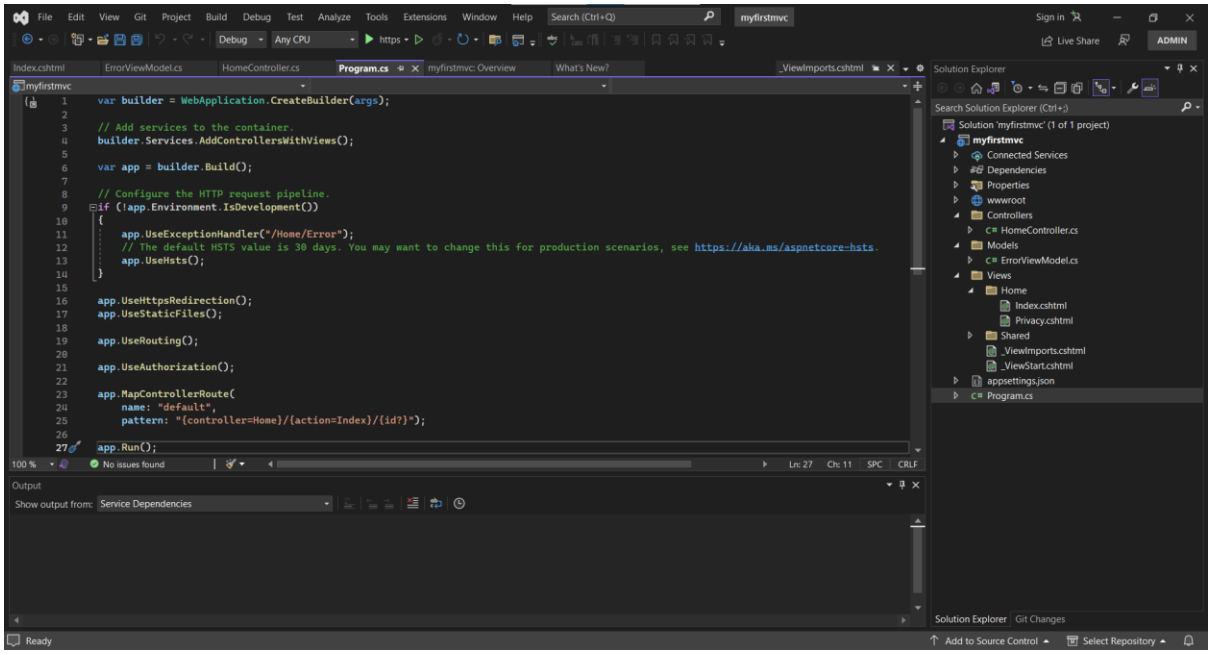
```
@{
   ViewData["Title"] = "Home Page";
}

<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>
```

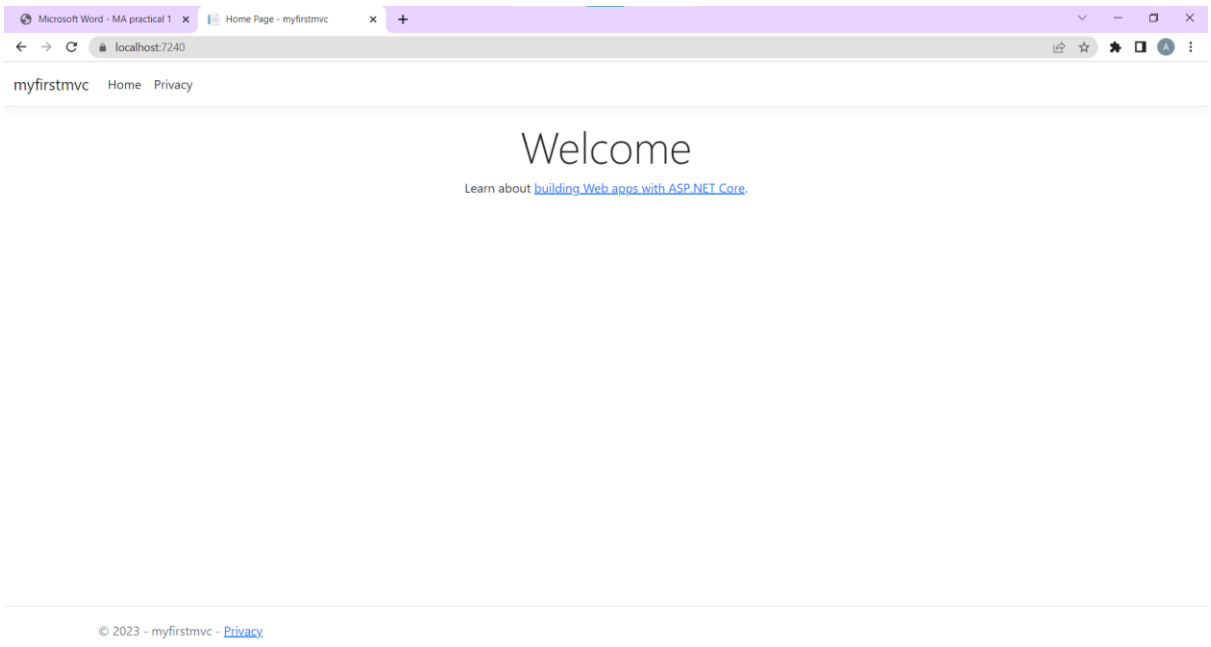
**Code for Program.cs:**

```
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

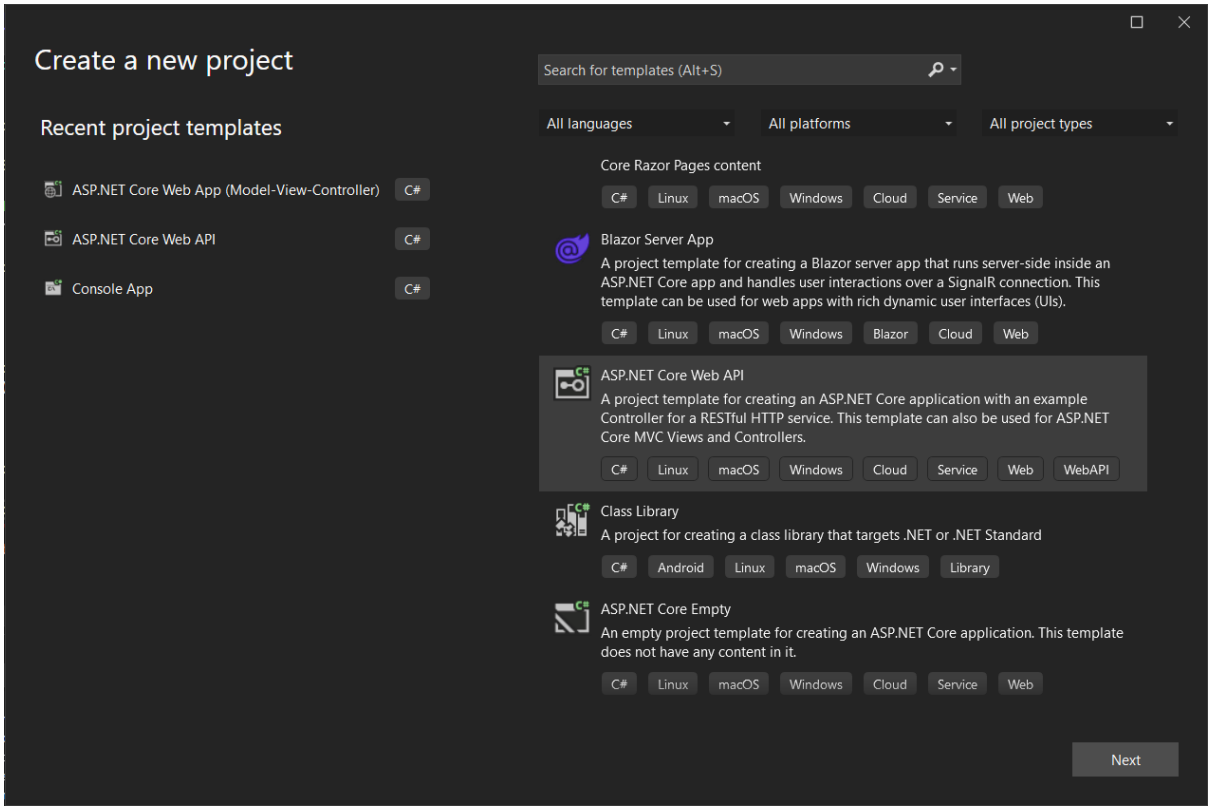
## Output:



# 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();
```

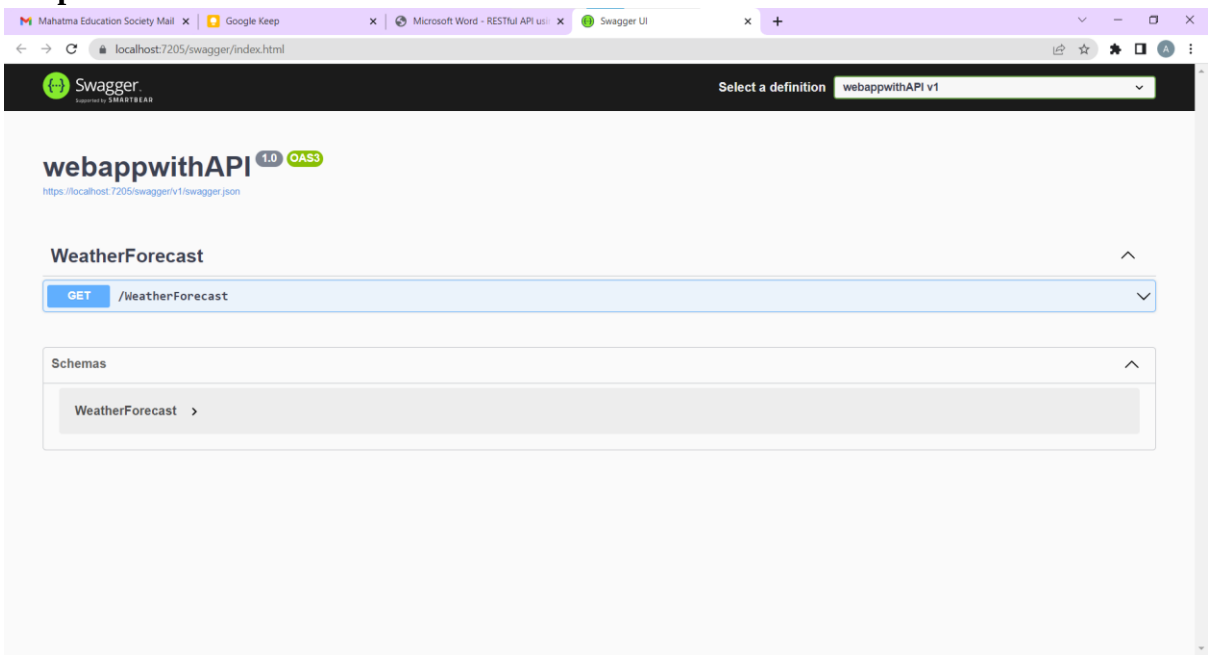
**Code for *WeatherForecastController.cs* under the folder *Controllers*:**

```
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
{
```

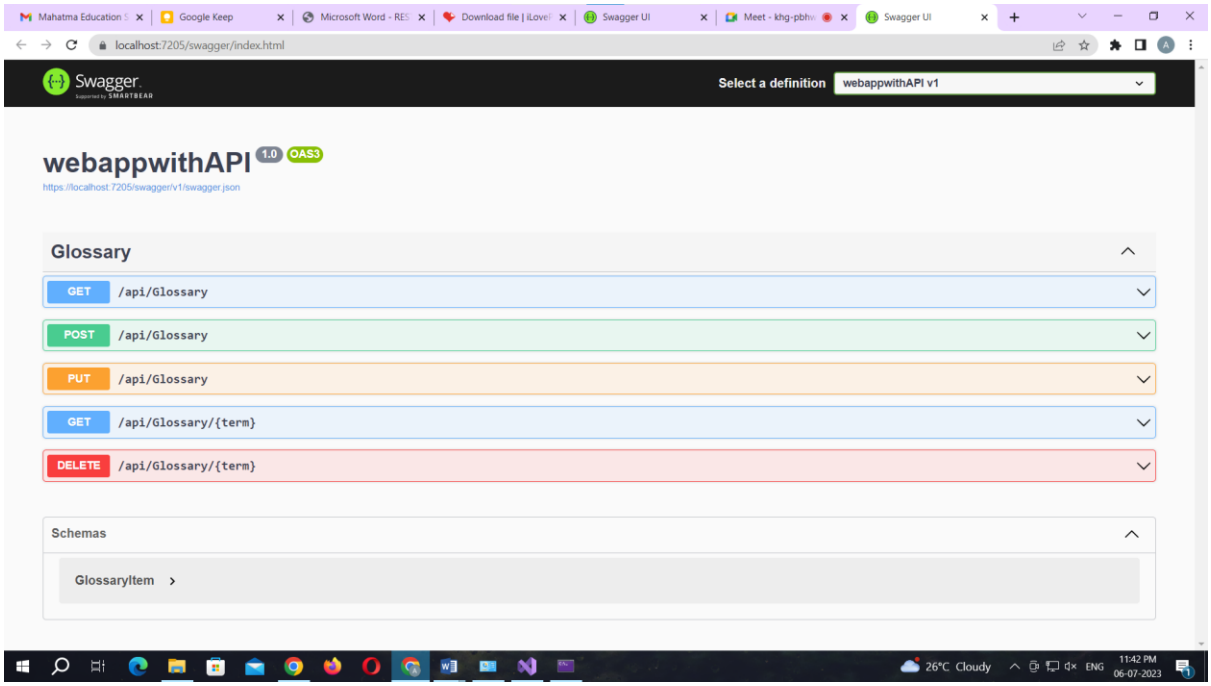
```
Term= "OpenID",
Definition = "An open standard for authentication"
}
};
      [HttpGet]
      public ActionResult<List<GlossaryItem>> Get()
      {
         return Ok(Glossary);
      }
      [HttpGet]
      [Route("{term}")]
      public ActionResult<GlossaryItem> Get(string term)
      {
         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:**

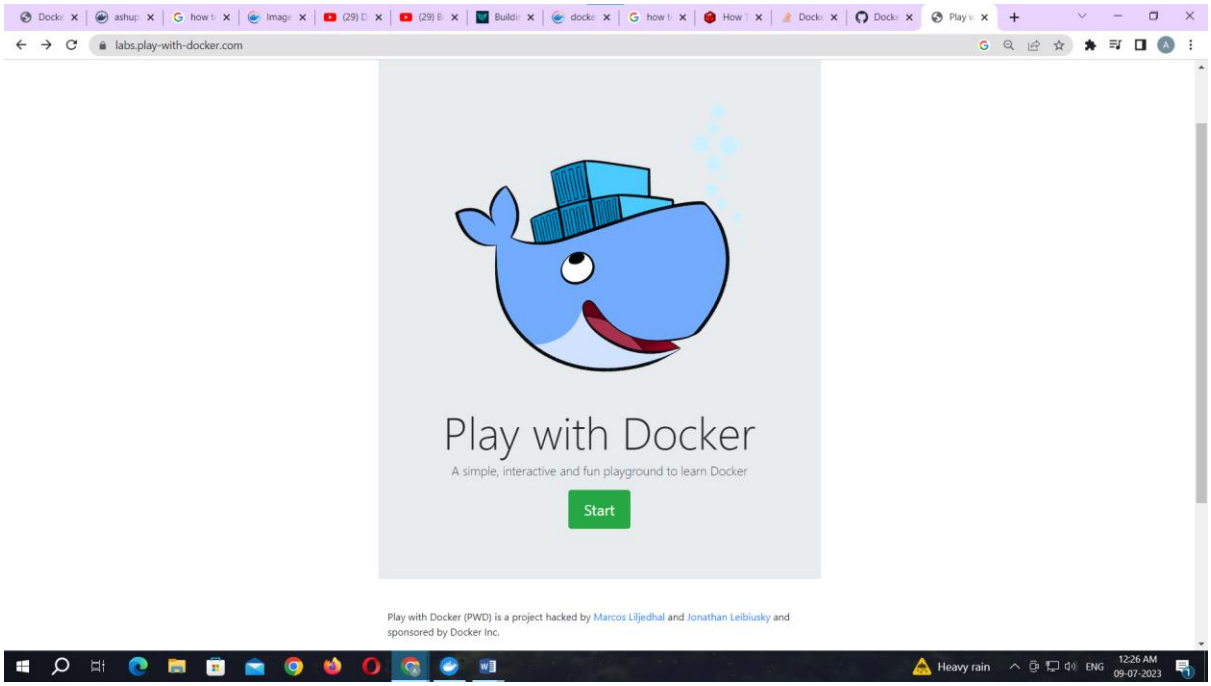

localhost:7205/swagger/index.html
Swagger
Select a definition
webappwithAPI v1
webappwithAPI 1.0 OAS3
https://localhost:7205/swagger/v1/swagger.json
Glossary
GET /api/Glossary
POST /api/Glossary
PUT /api/Glossary
GET /api/Glossary/{term}
DELETE /api/Glossary/{term}
Schemas
GlossaryItem
26°C Cloudy
ENG
11:42 PM
06-07-2023

# Practical No. 03

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

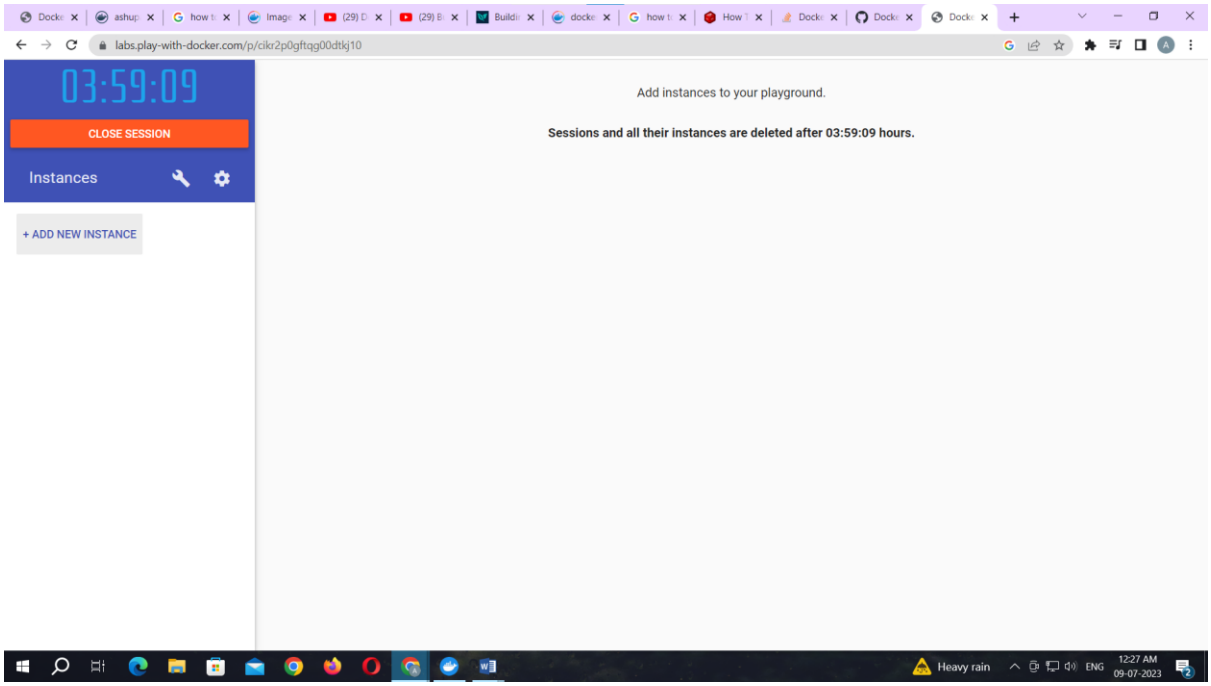
Create Docker Hub account (sign up)

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




Click on start

Add new instance

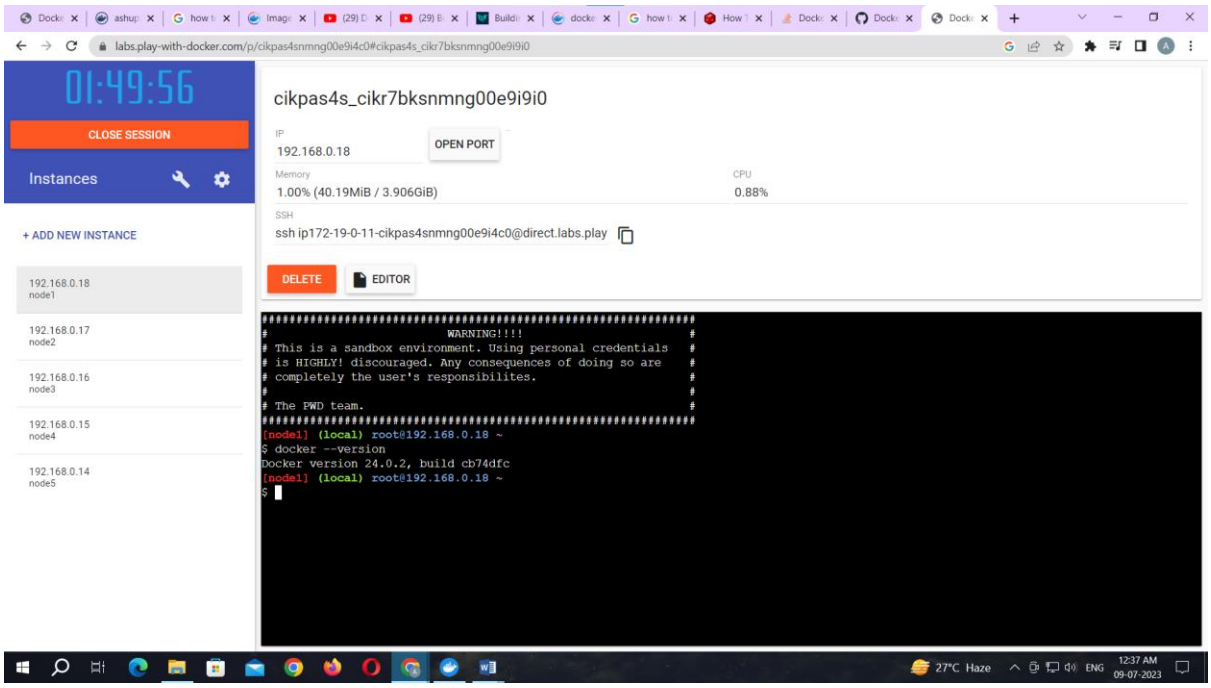



To pull and push images using docker

Command: to check docker version

docker –version

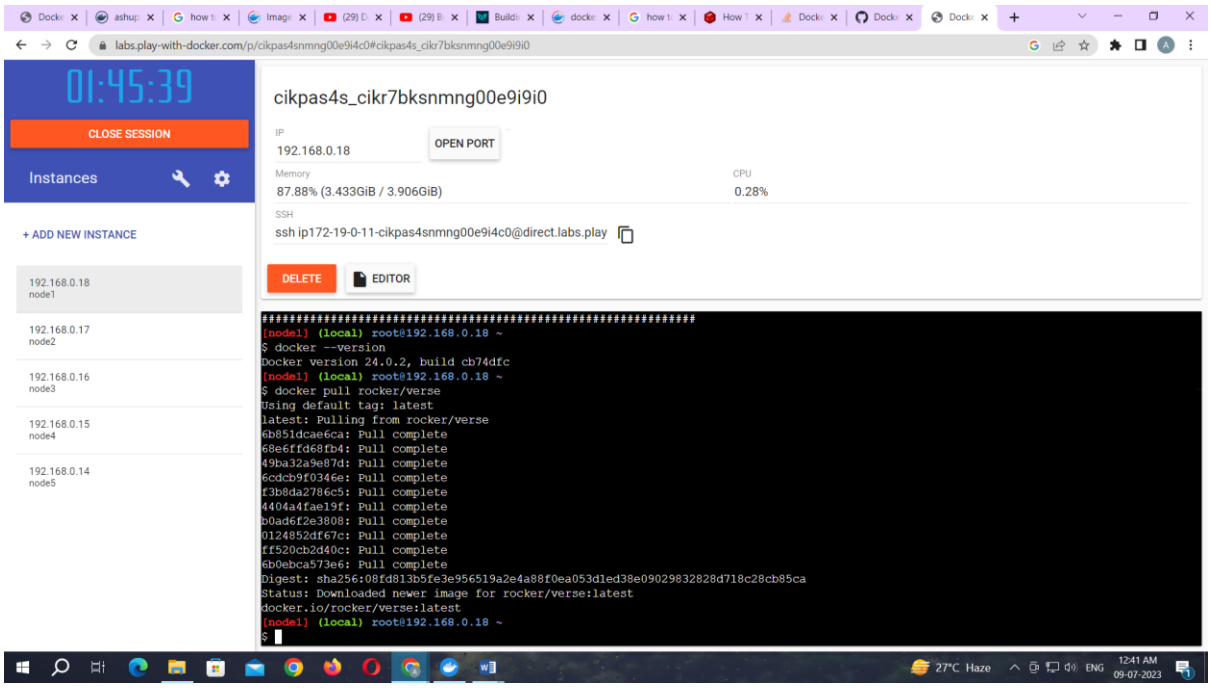
**Output:**




Command: to pull readymade image

docker pull rocker/verse

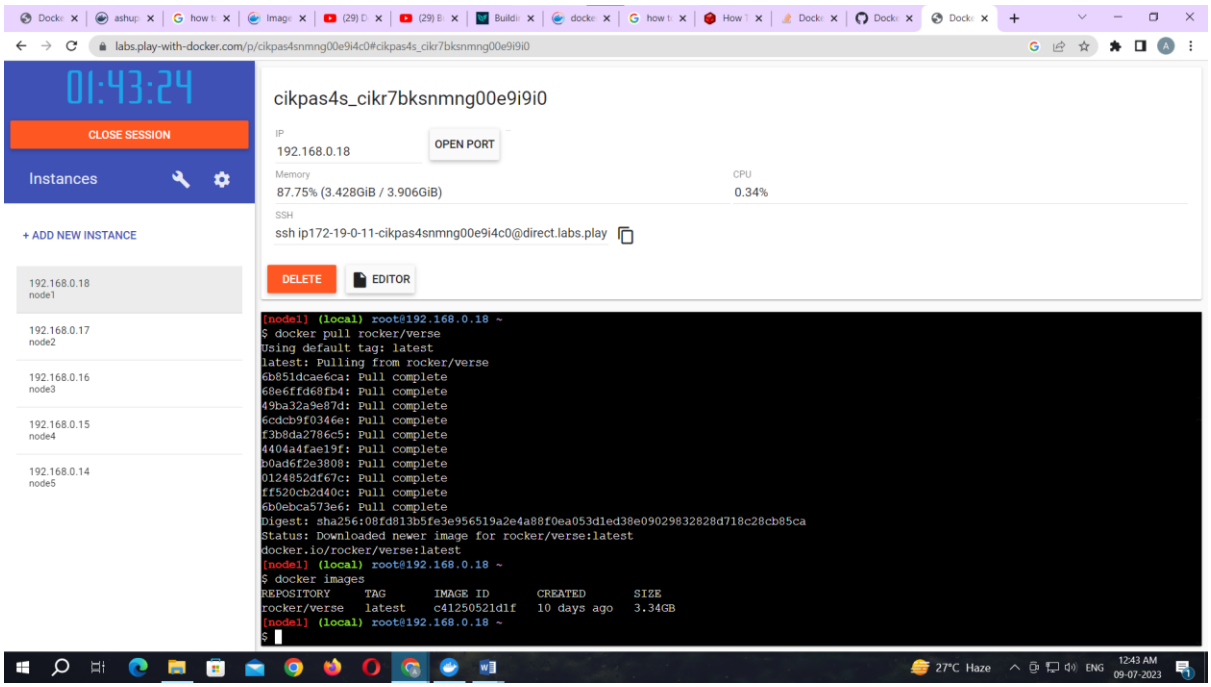
**Output:**




Command: to check images in docker

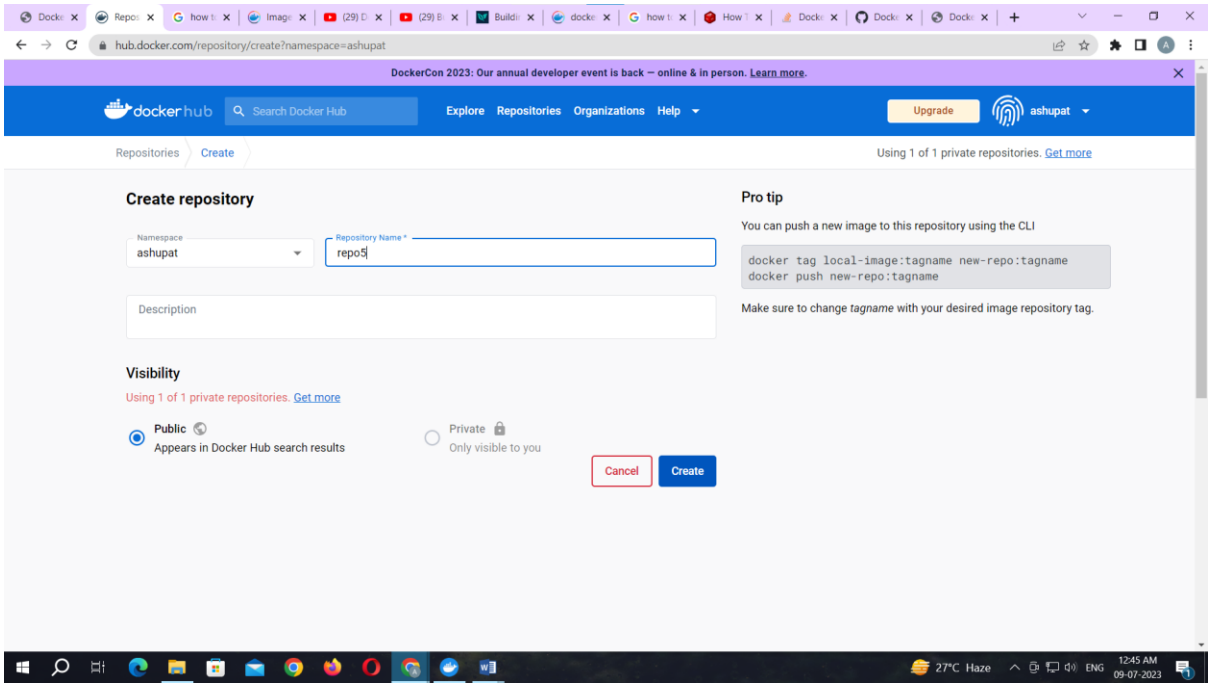
docker images

**Output:**



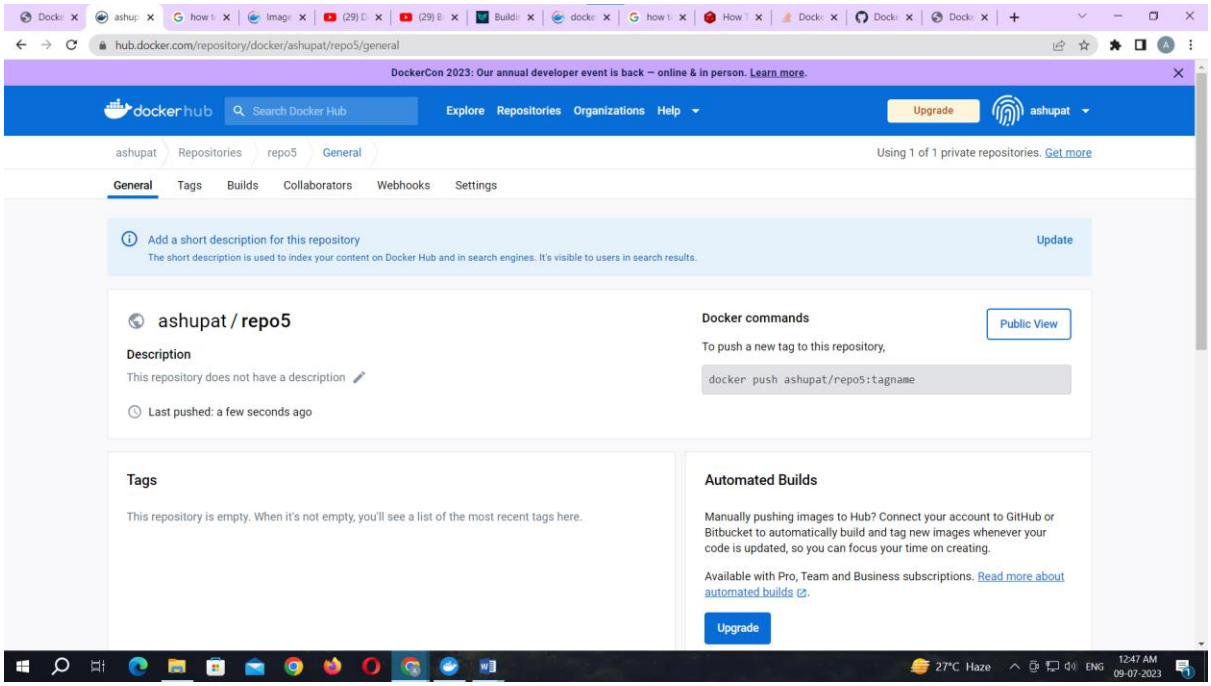

Now Login to docker hub and create repository

**Output:**




Click on Create button

Now check repository created

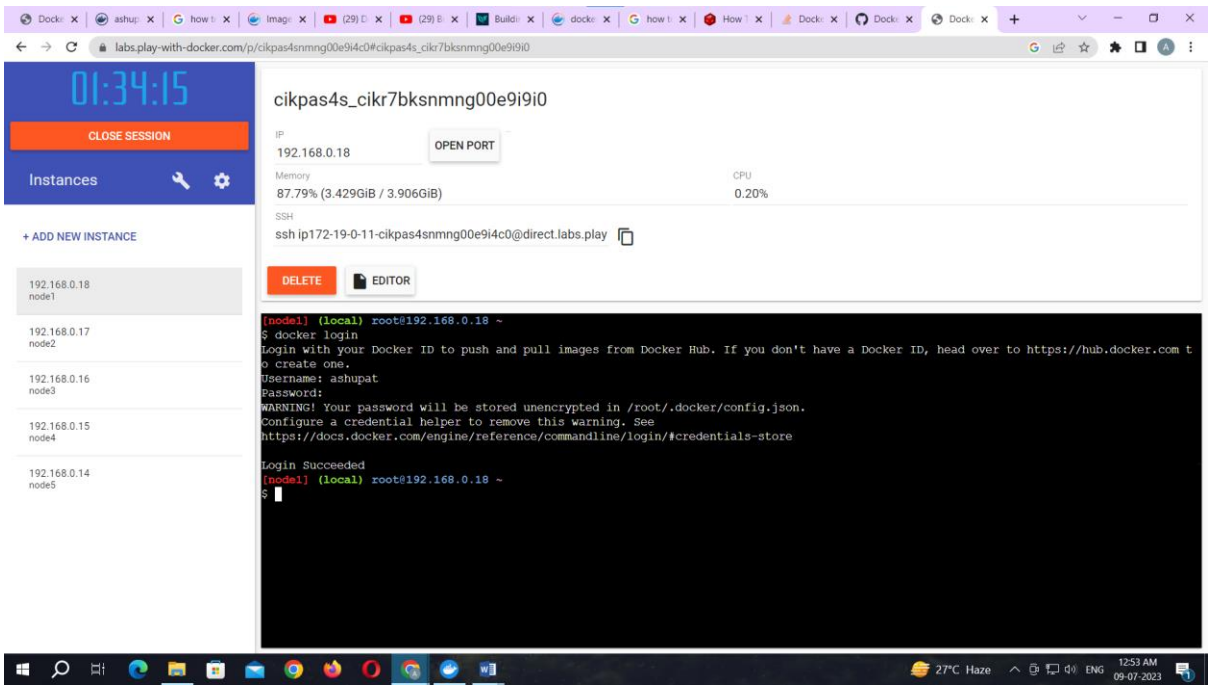



Command: to login to your docker account

docker login –username=ashupat password:

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

**Output:**

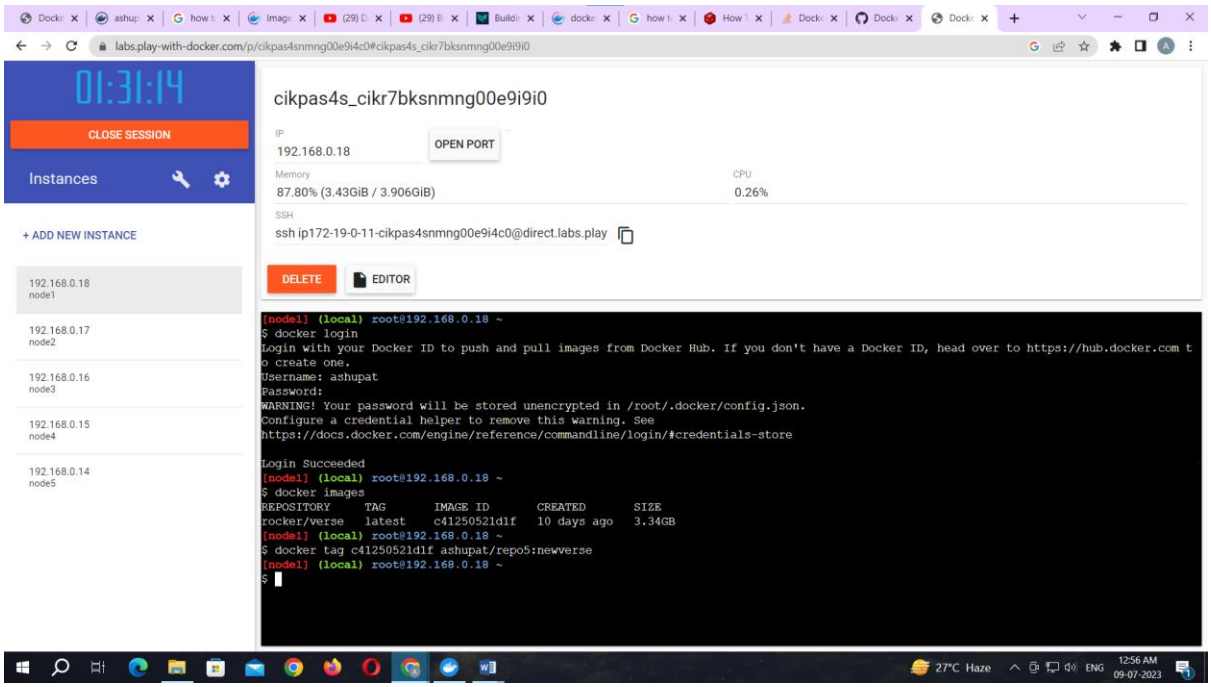



Command: to tag image

docker tag c41250521d1f ashupat/repo5:newverse

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

**Output:**

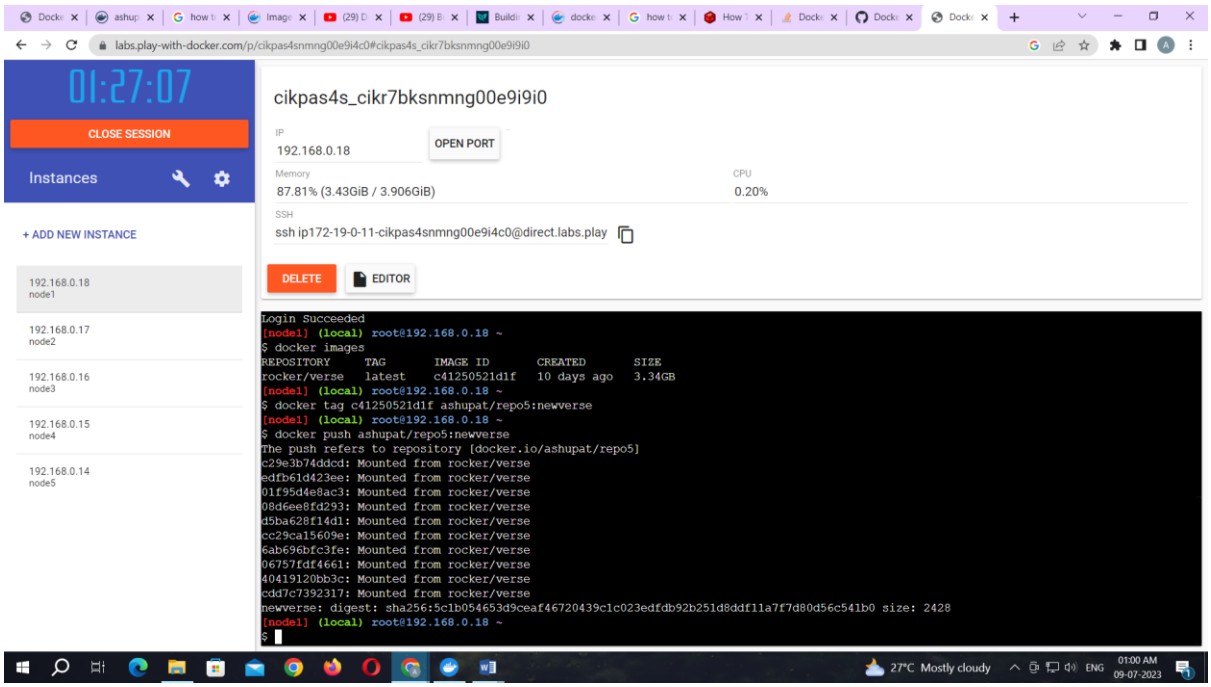



Command: to push image to docker hub account

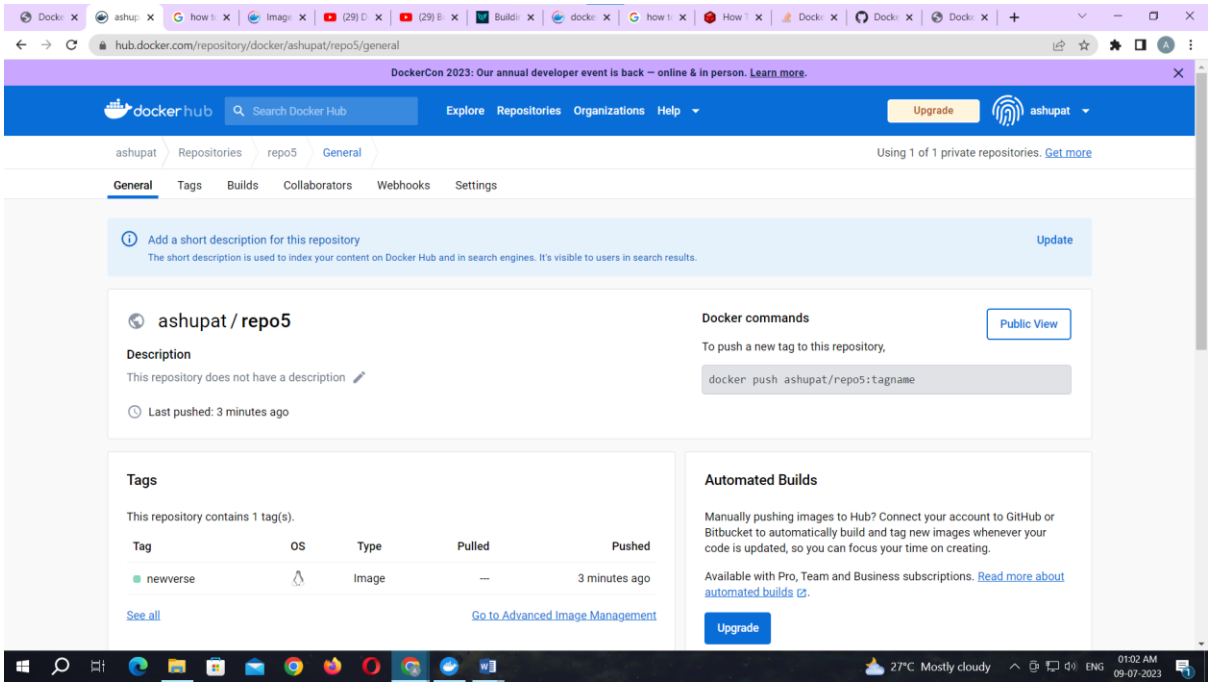
docker push ashupat/repo5:newverse

note: newverse is tag name created above.

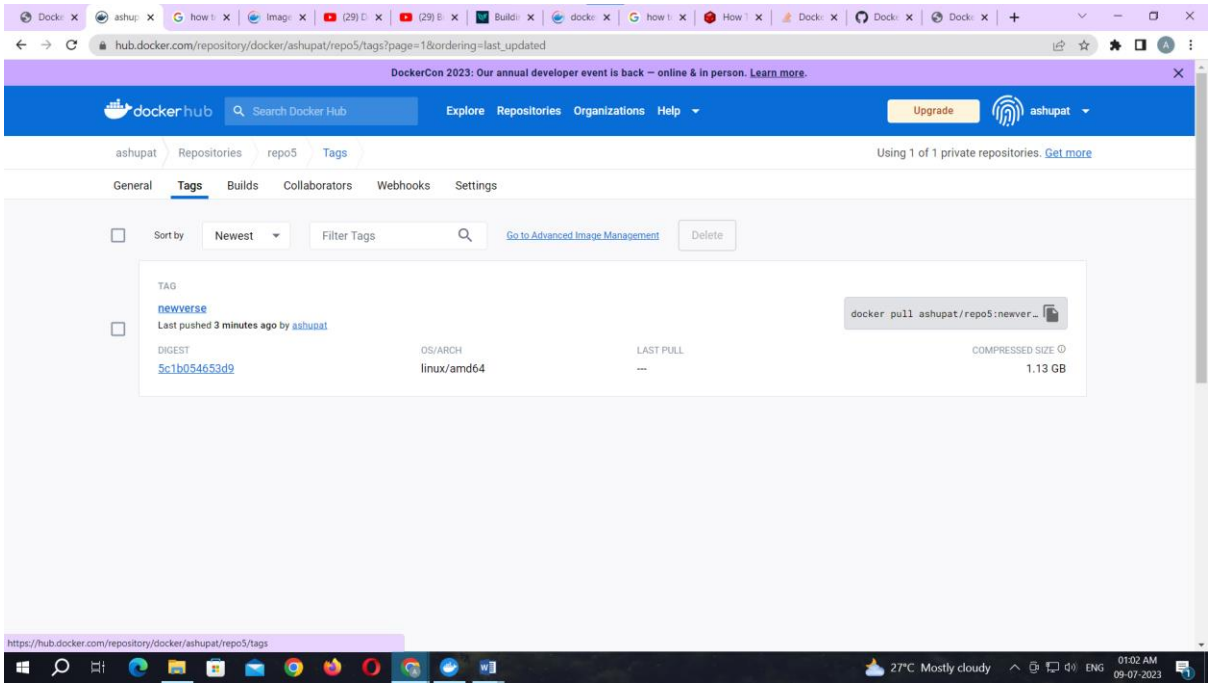
**Output:**



Check it in docker hub now




Click on tags and check.



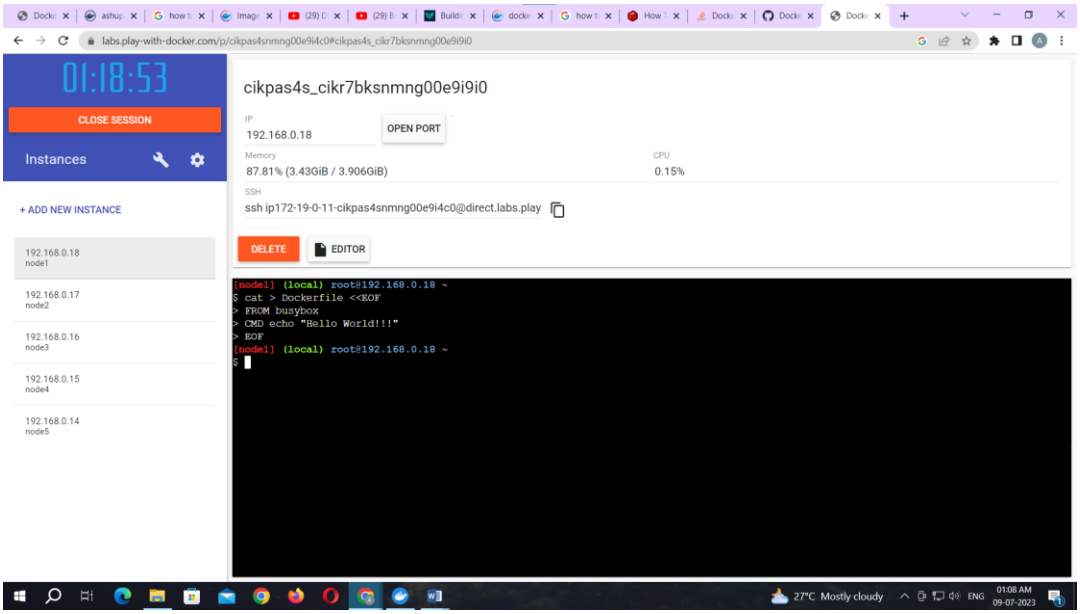
# Practical No. 04

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

Command: to create docker file

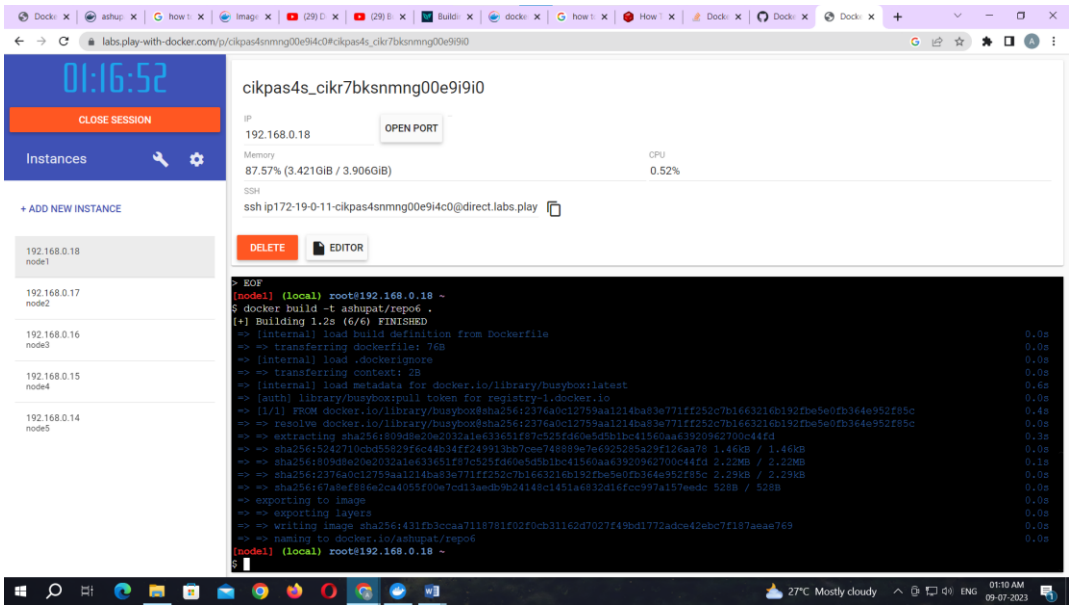
1) cat > Dockerfile <<EOF
2) FROM busybox
3) CMD echo "Hello world! This is my first Docker image."
4) EOF

**Output:**




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

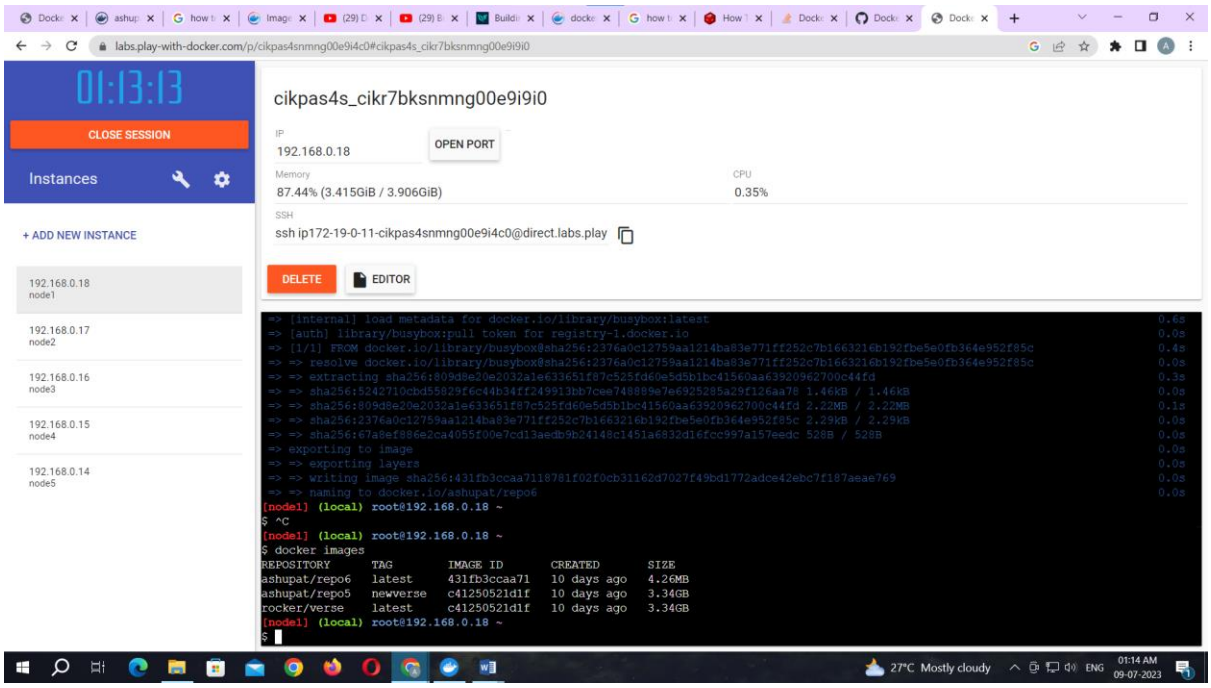
**Output:**



Command: to check docker images

docker images

**Output:**

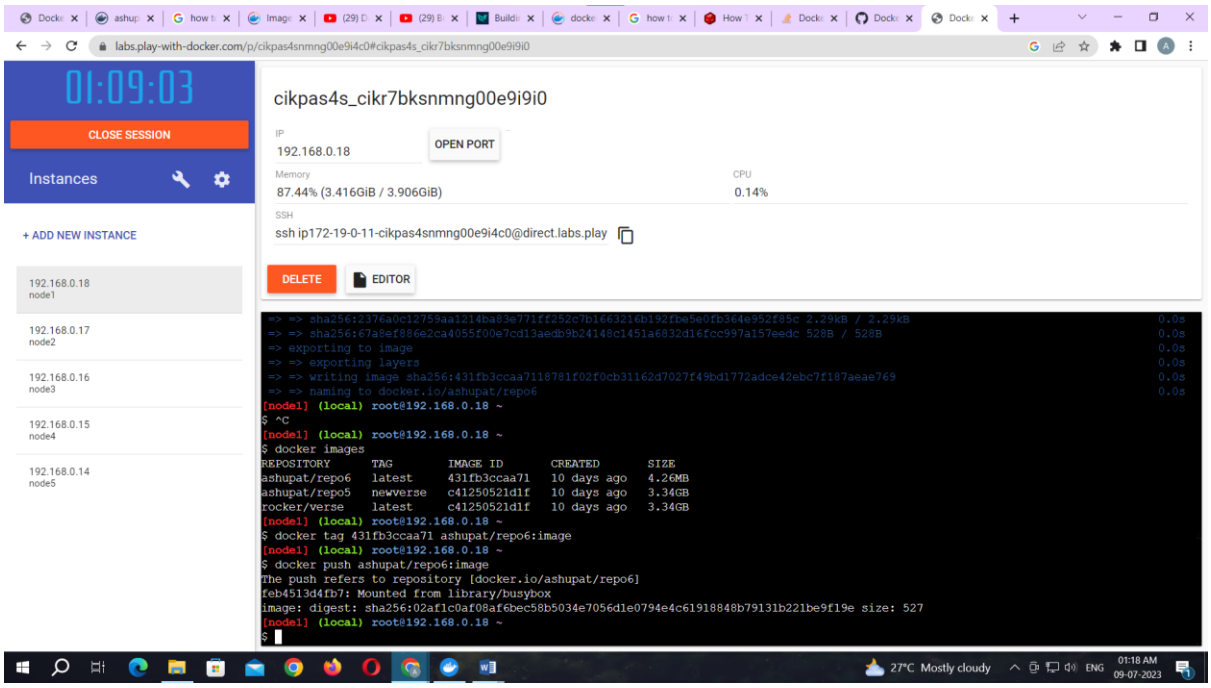



Command: to tag image

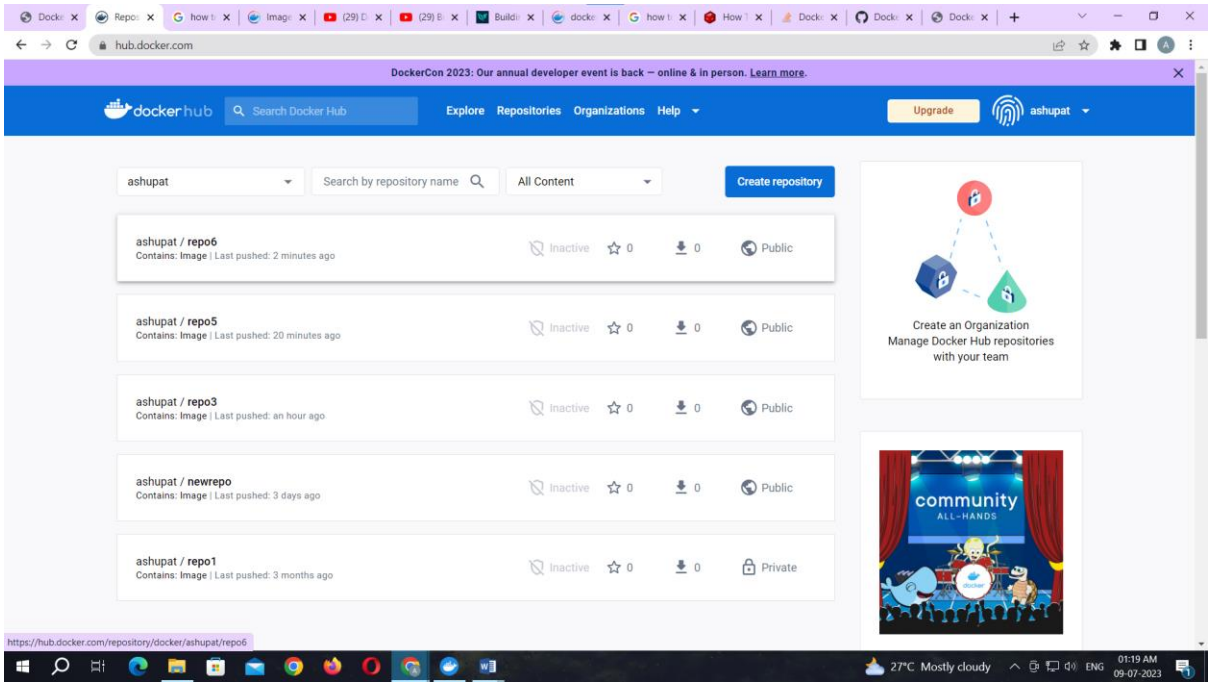
docker tag 431fb3ccaa71 ashupat/repo6:image

Command: to push image to docker hub

docker push ashupat/repo6:image



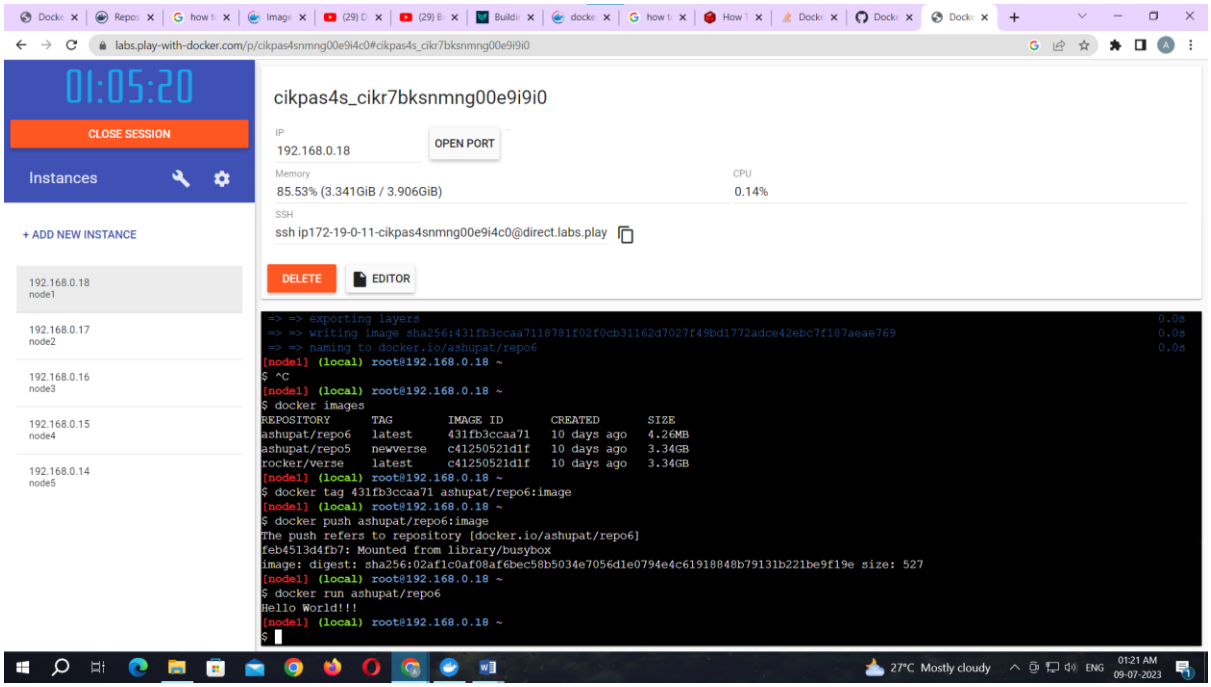
Now check it on docker hub.




Command: to run docker image:

docker run kbdocker11/repo2

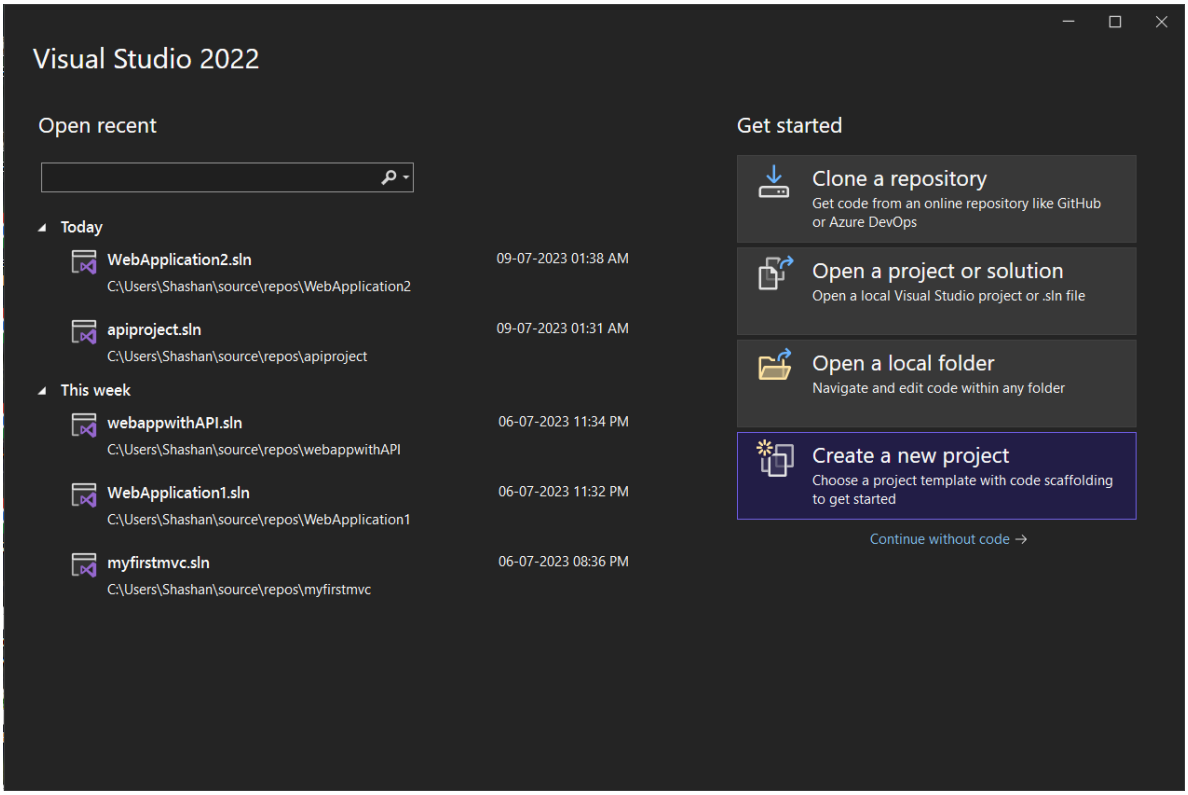
**Output:**



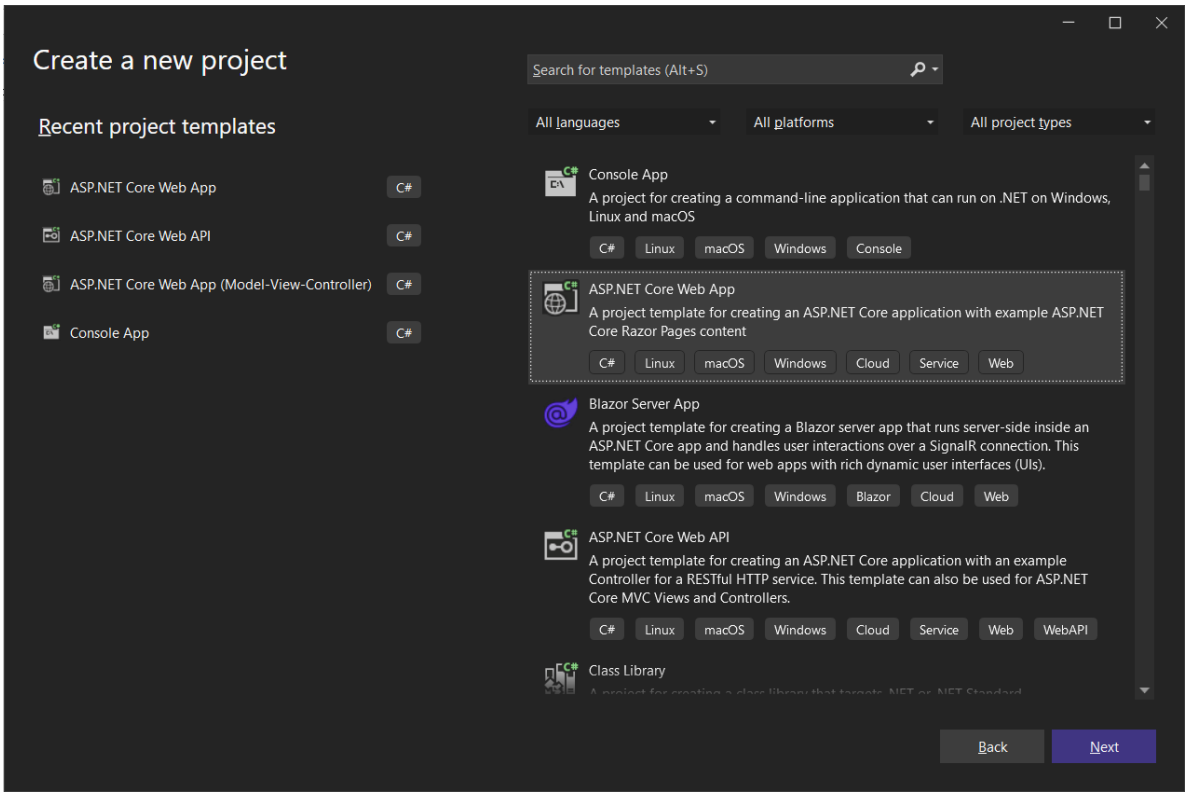
# Practical No. 05

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

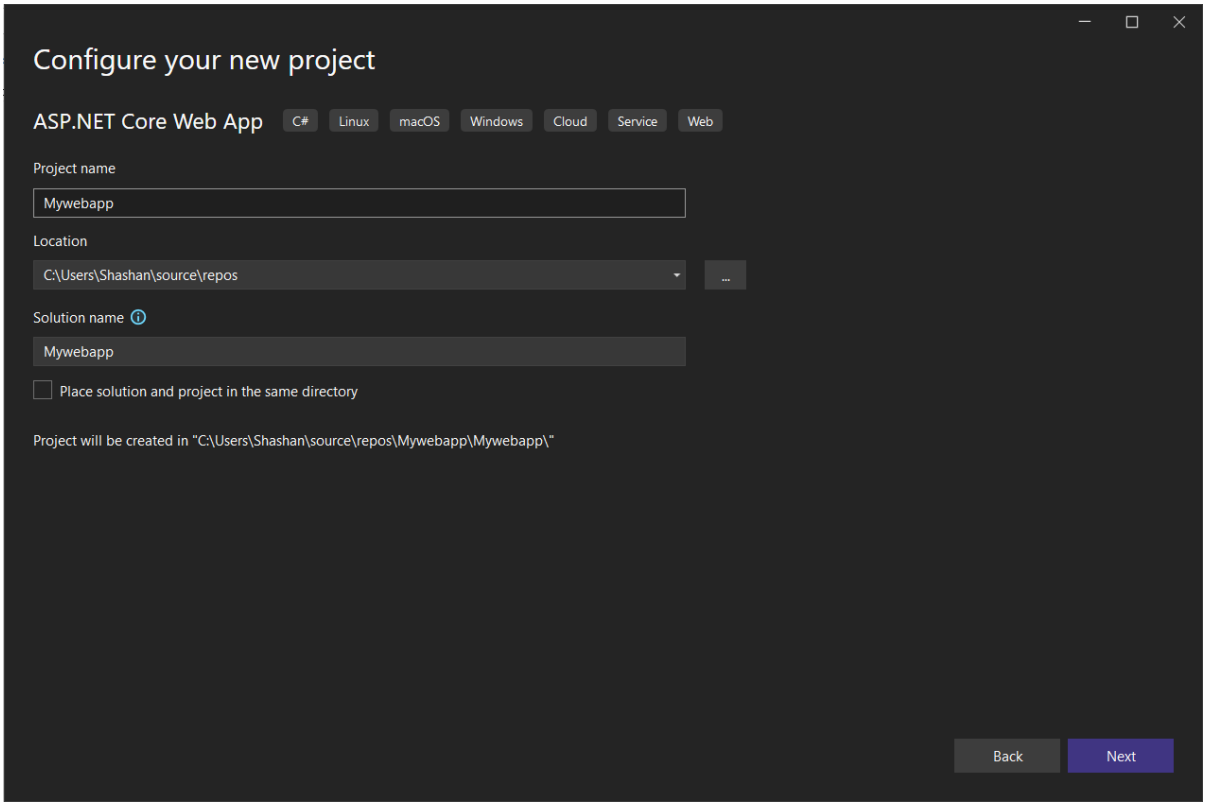
Let's start by creating a new project in the **Microsoft Visual Studio**



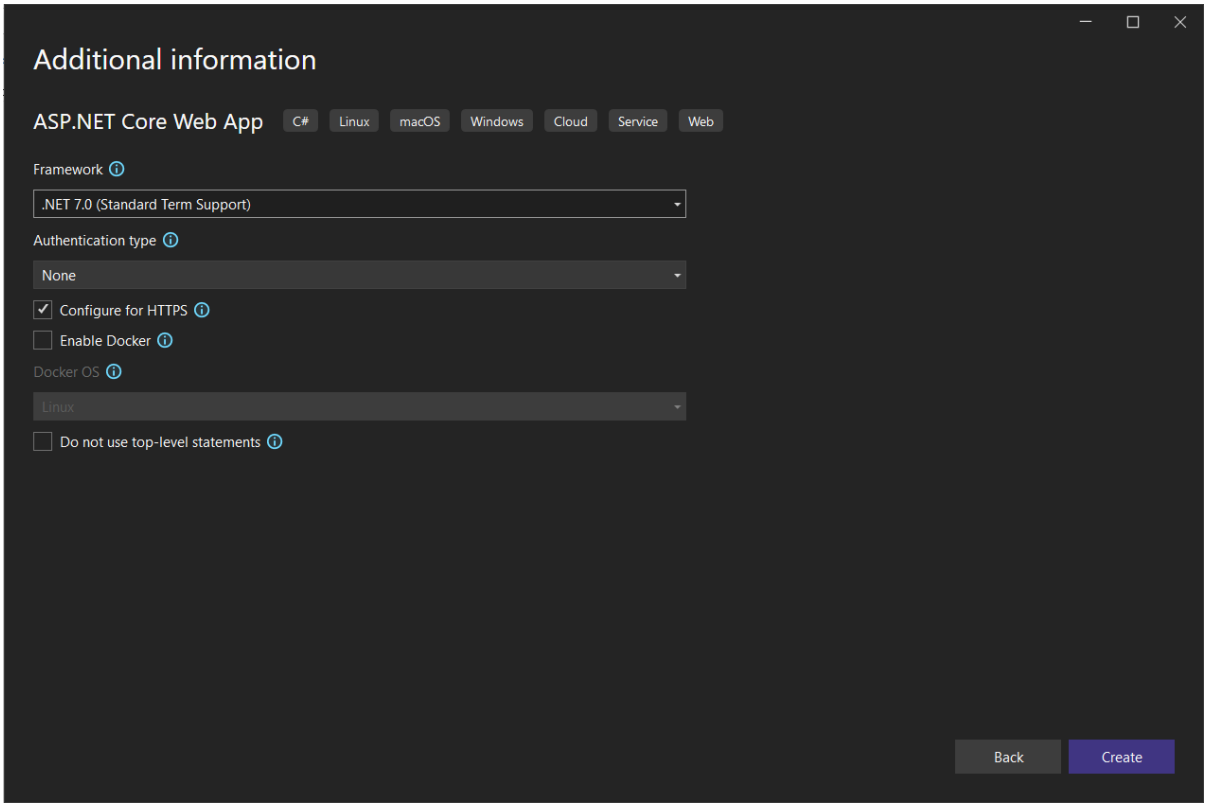

Select the **ASP.NET Core Web App** form the various options.



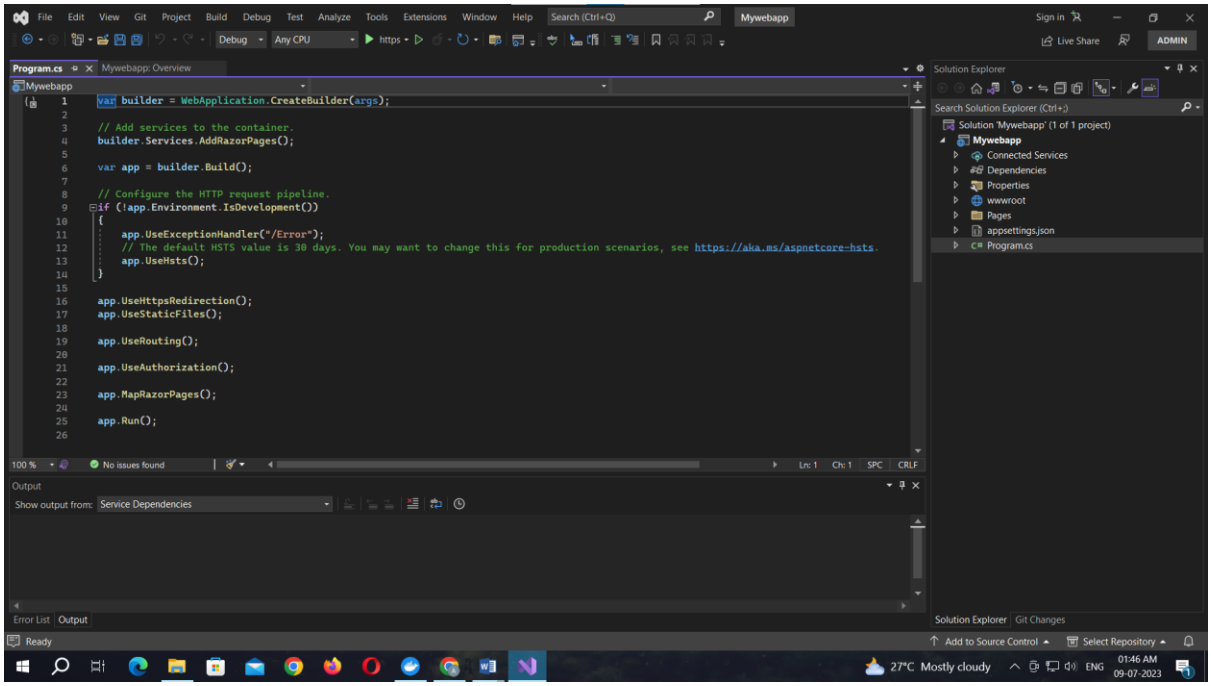
Give an appropriate name to the project.



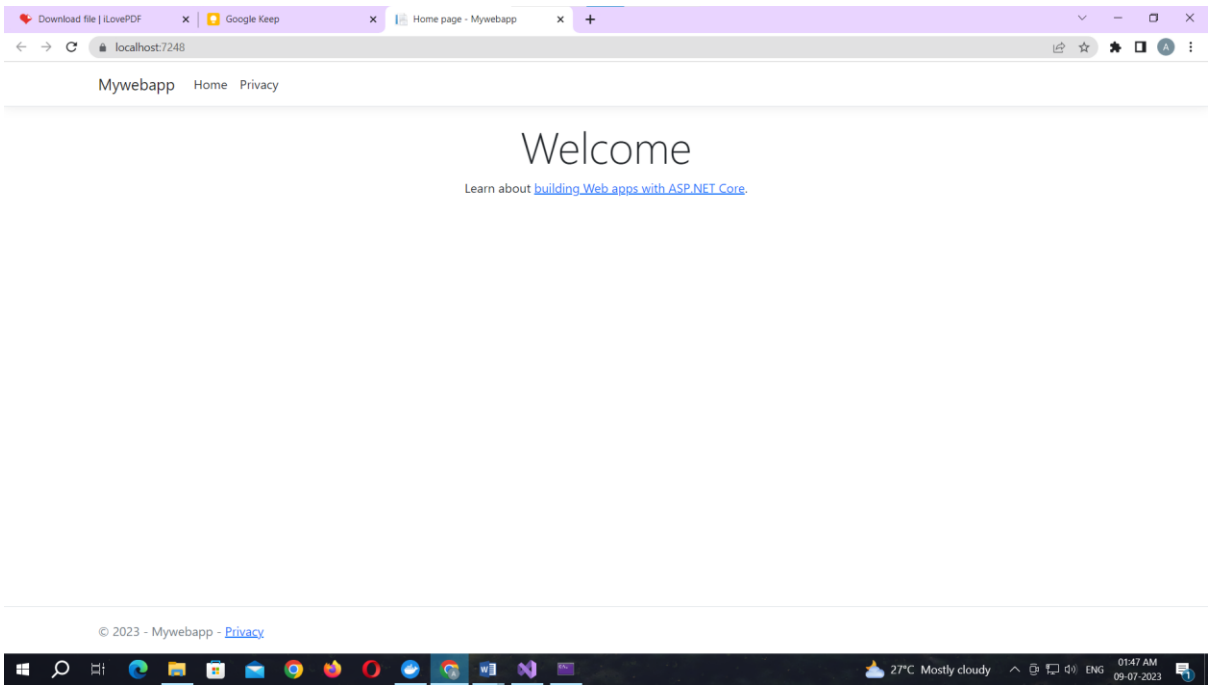

Click on the **Create** button.



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

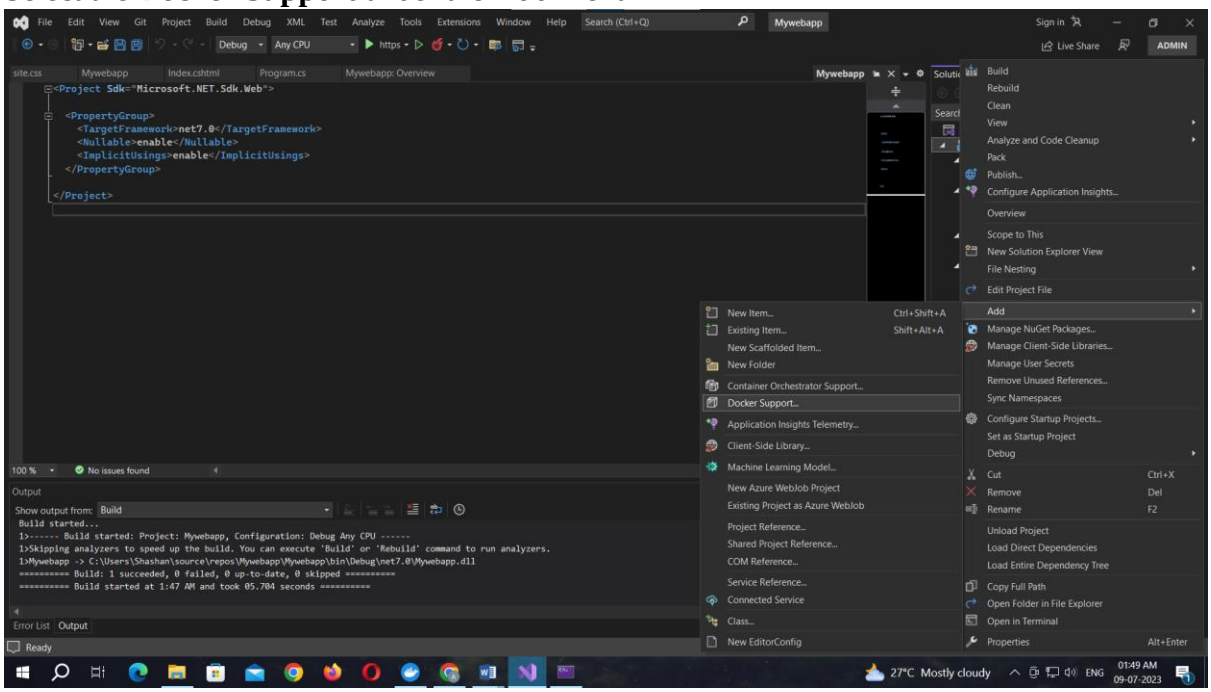



## Output:

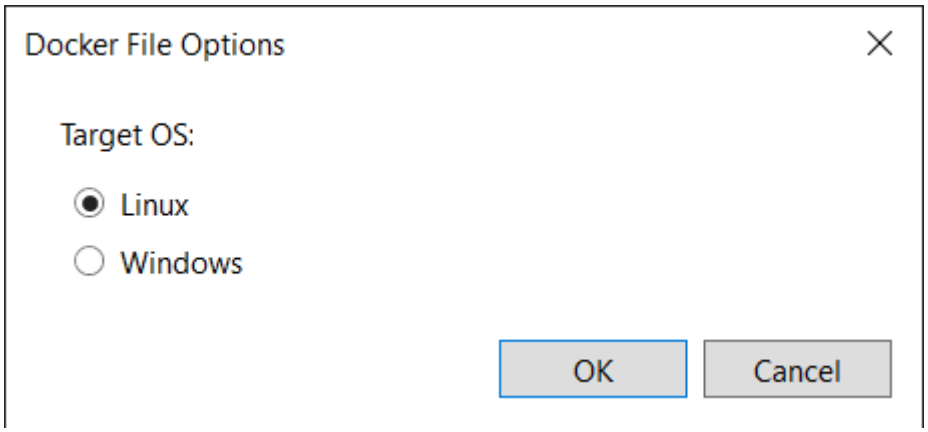


**Right click** on the app name then select the **Add** option form the menu.

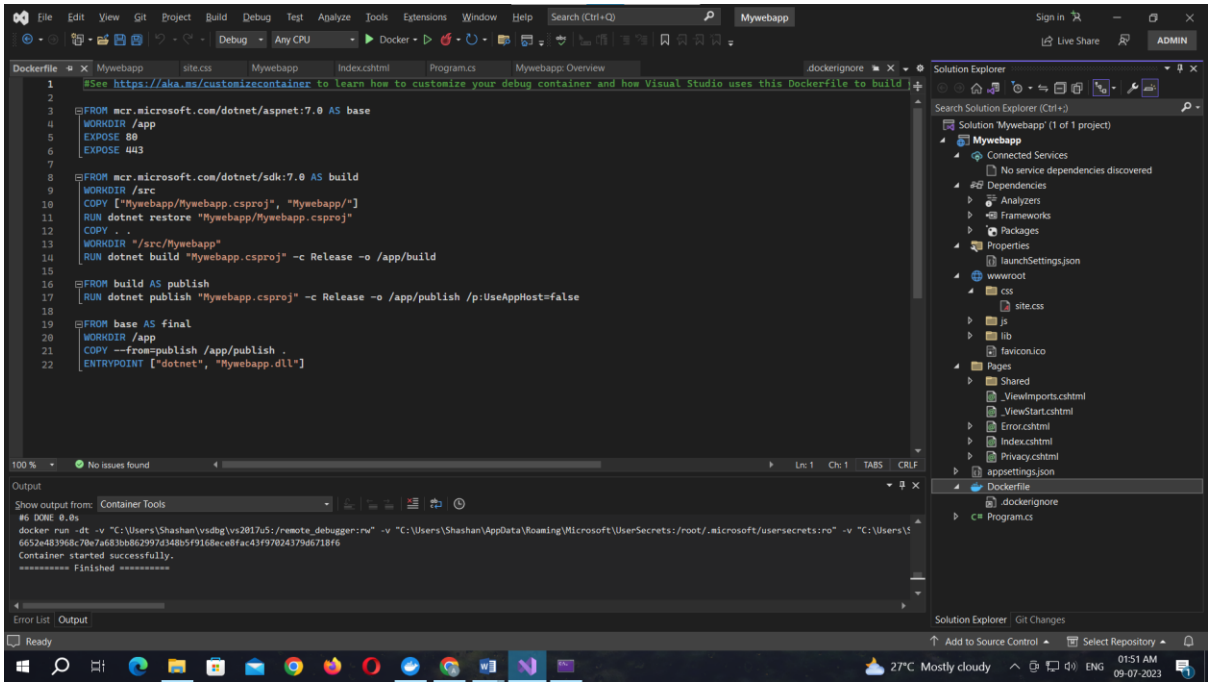
Select the **Docker Support** under the **Add** menu




Select the Target OS as Linux and click OK button.



The file naming Dockerfile will appear

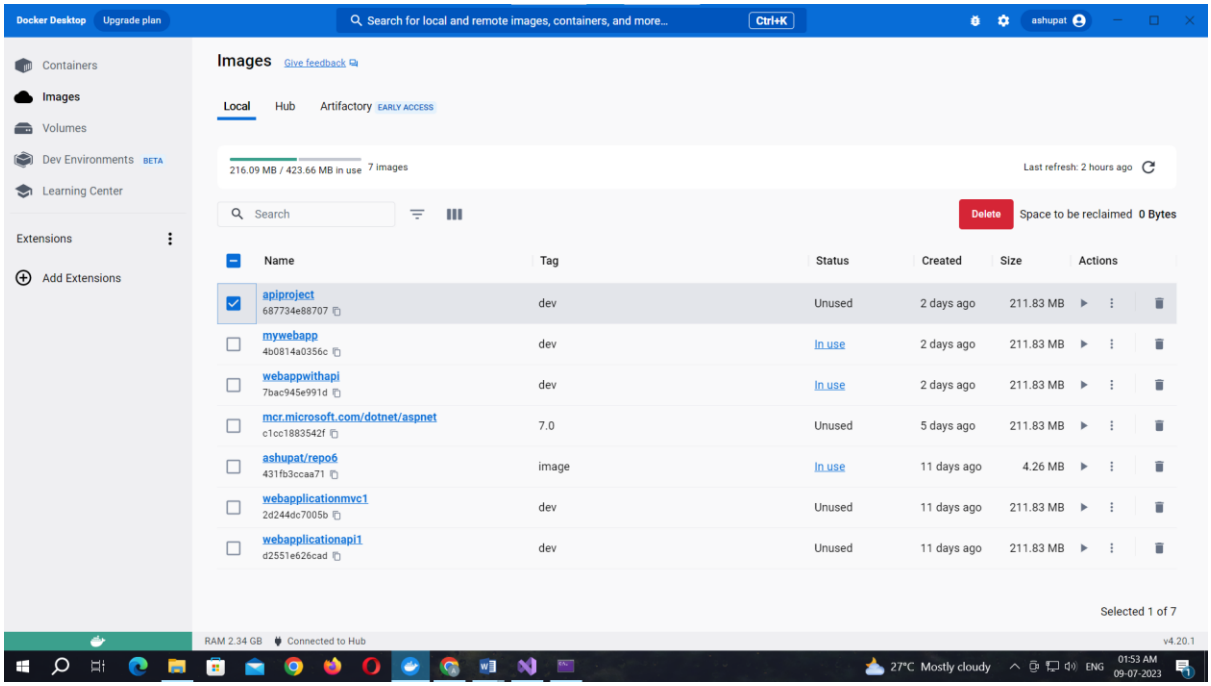



Code for Dockerfile:

#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.

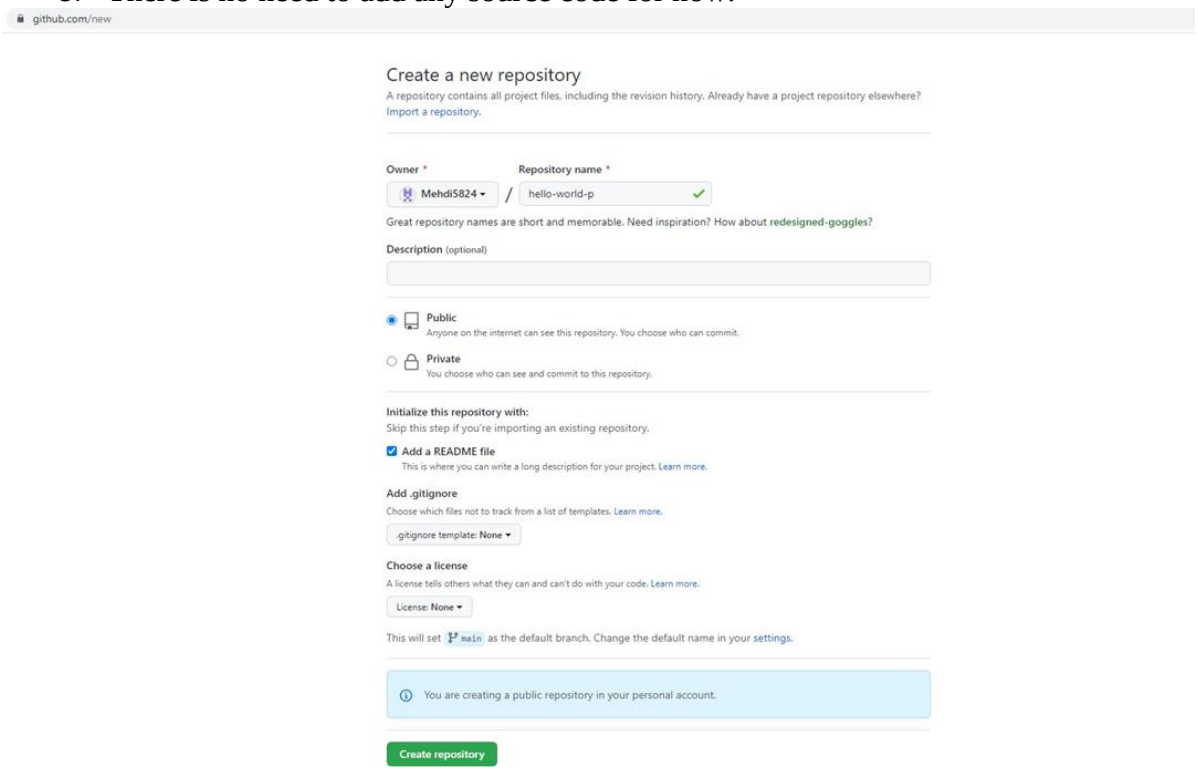


# Practical No. 06

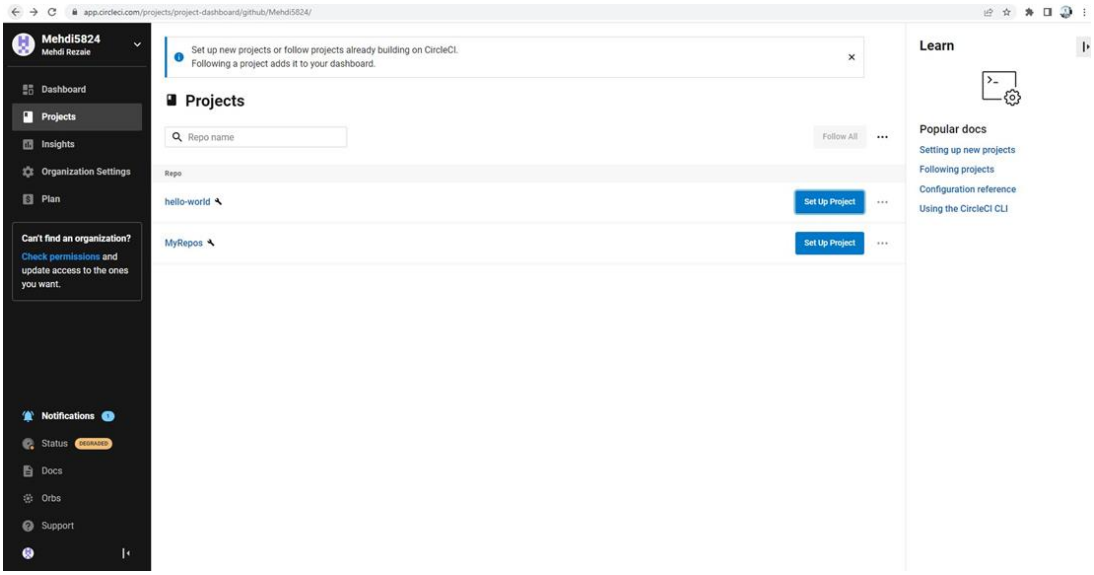
## Aim: Working with the CircleCI

## 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.



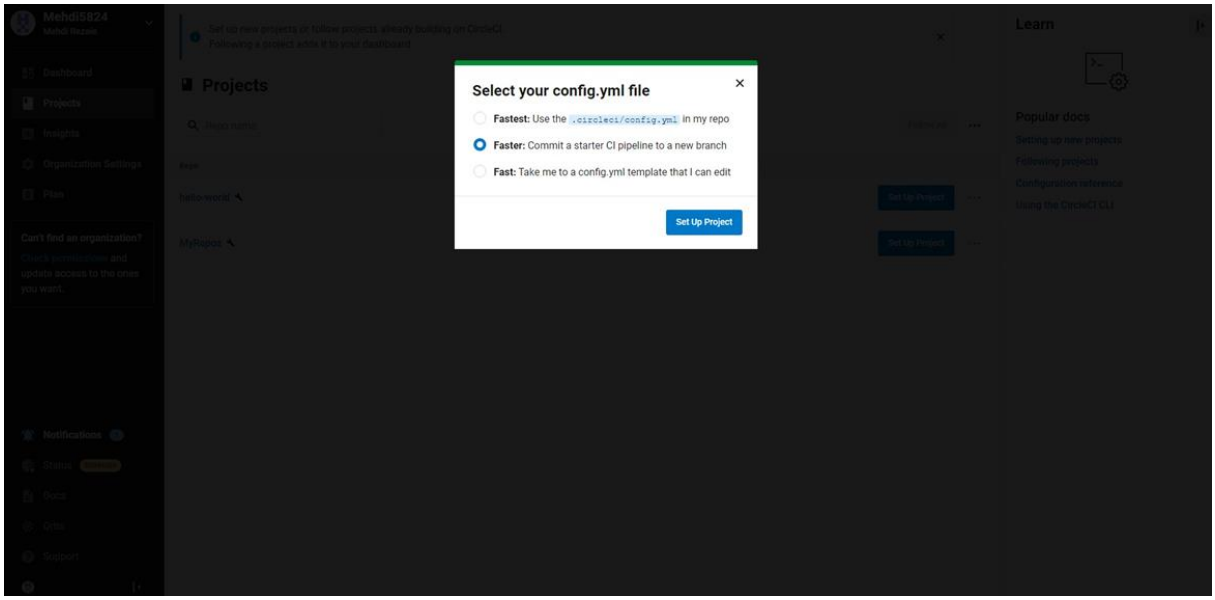

Login to Circle CI https://app.circleci.com/ Using GitHub Login, Once logged in navigate to Projects.



**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.



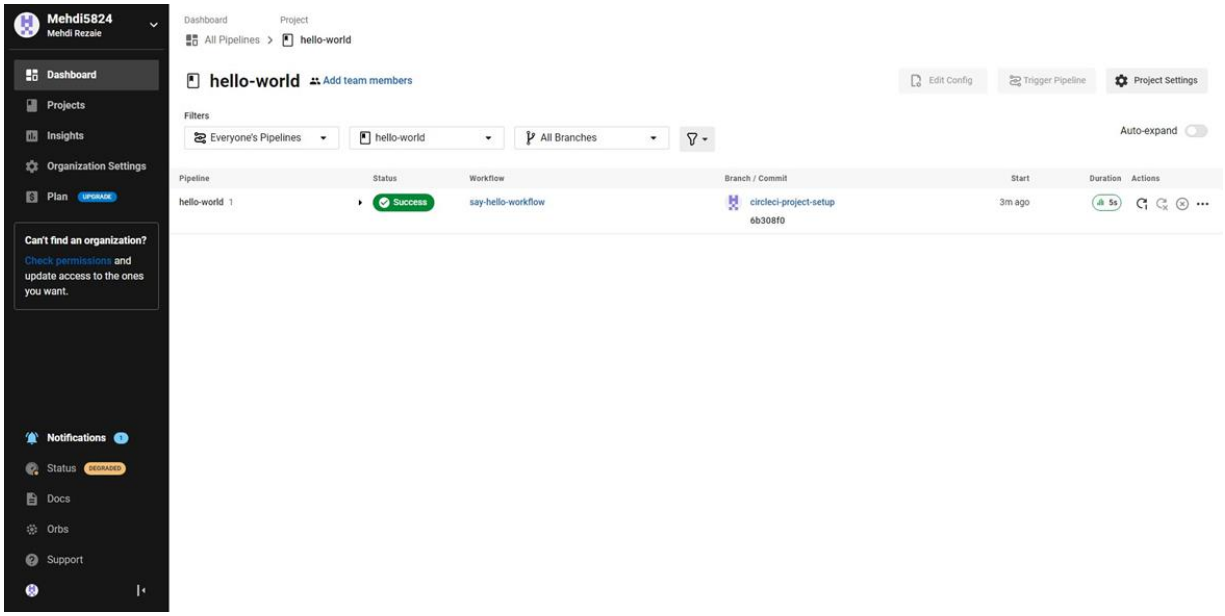

**Your first Pipeline**

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

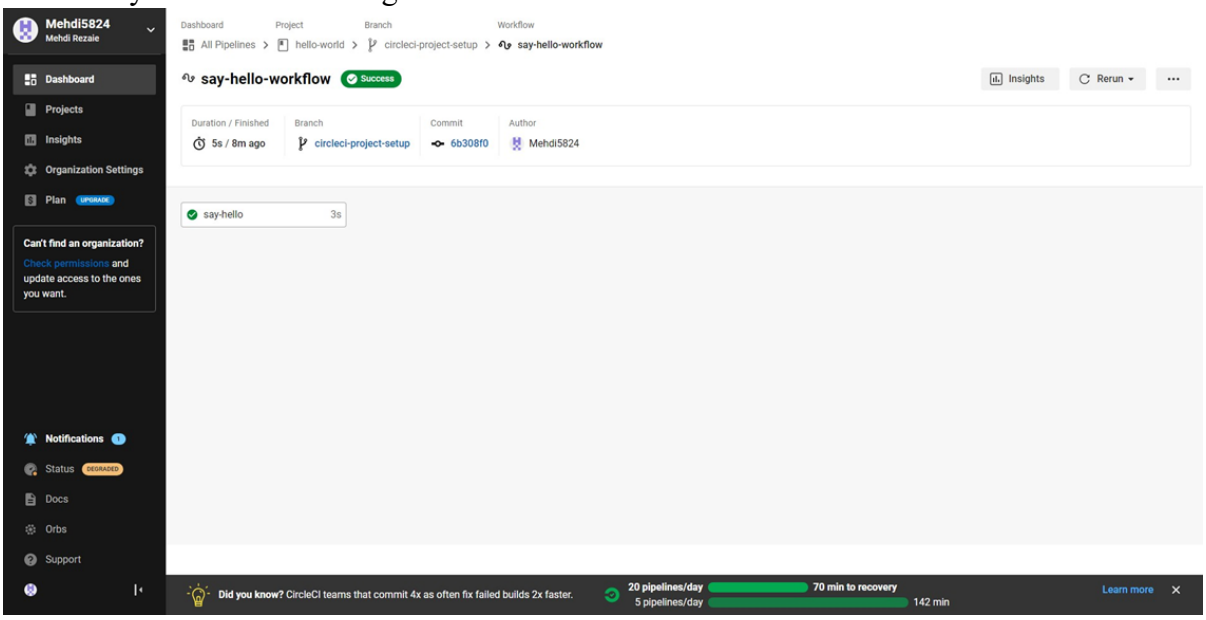
Within this workflow, the pipeline ran one job, called say-hello. Click say-hello to see the steps in this job:

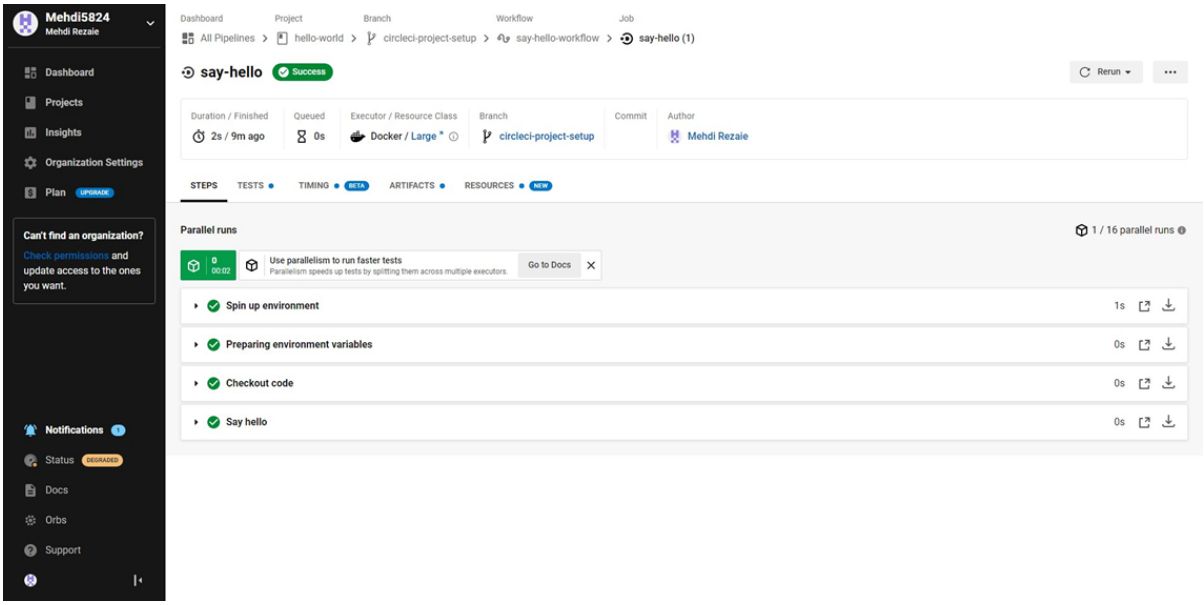
a. Spin up environment
b. Preparing environment variables
c. Checkout code
d. Say hello

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

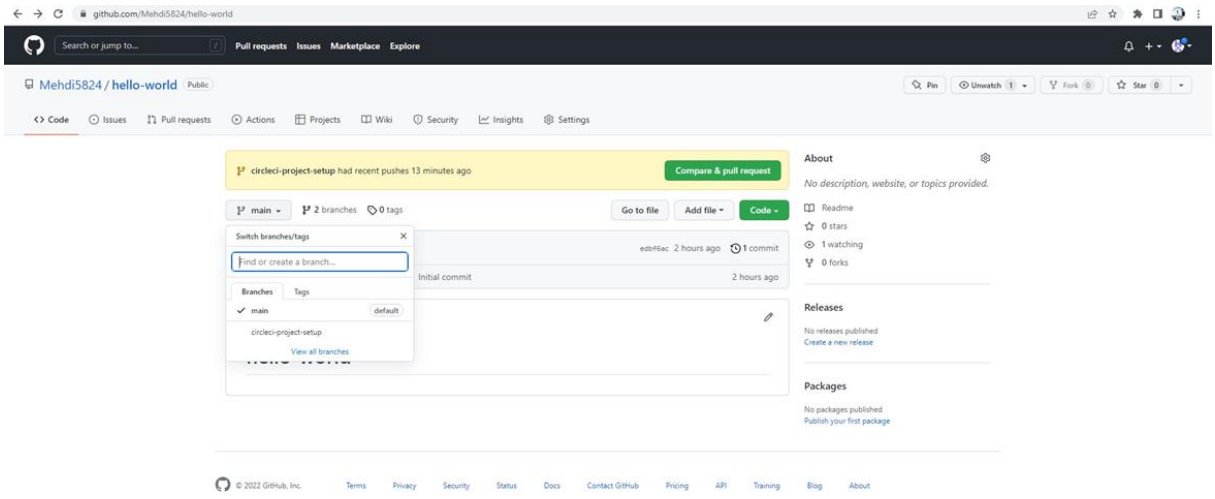



Select “say-hello” Job with a green tick



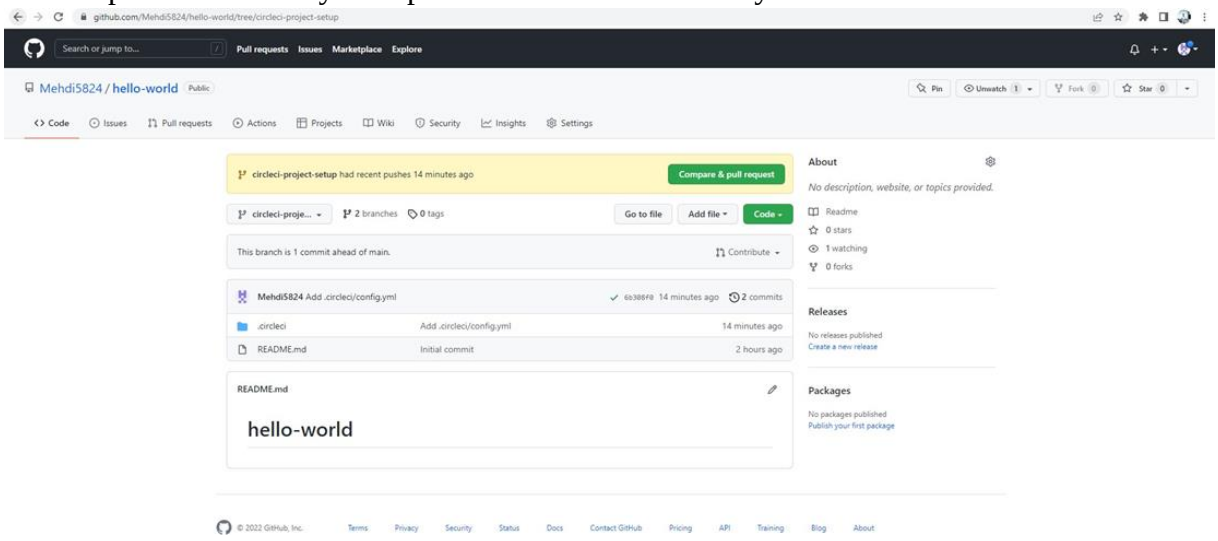


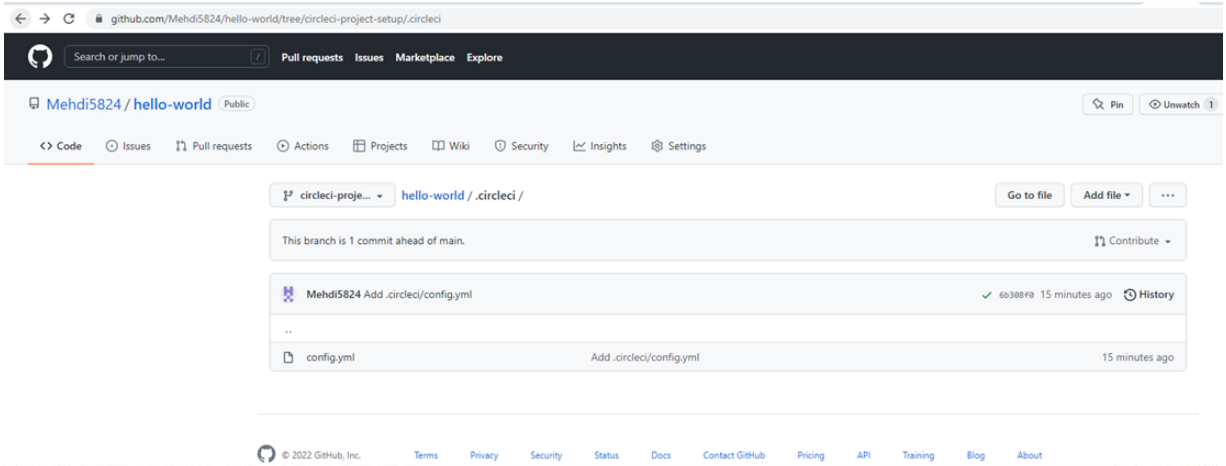

Select Branch and option circleci-project-setup

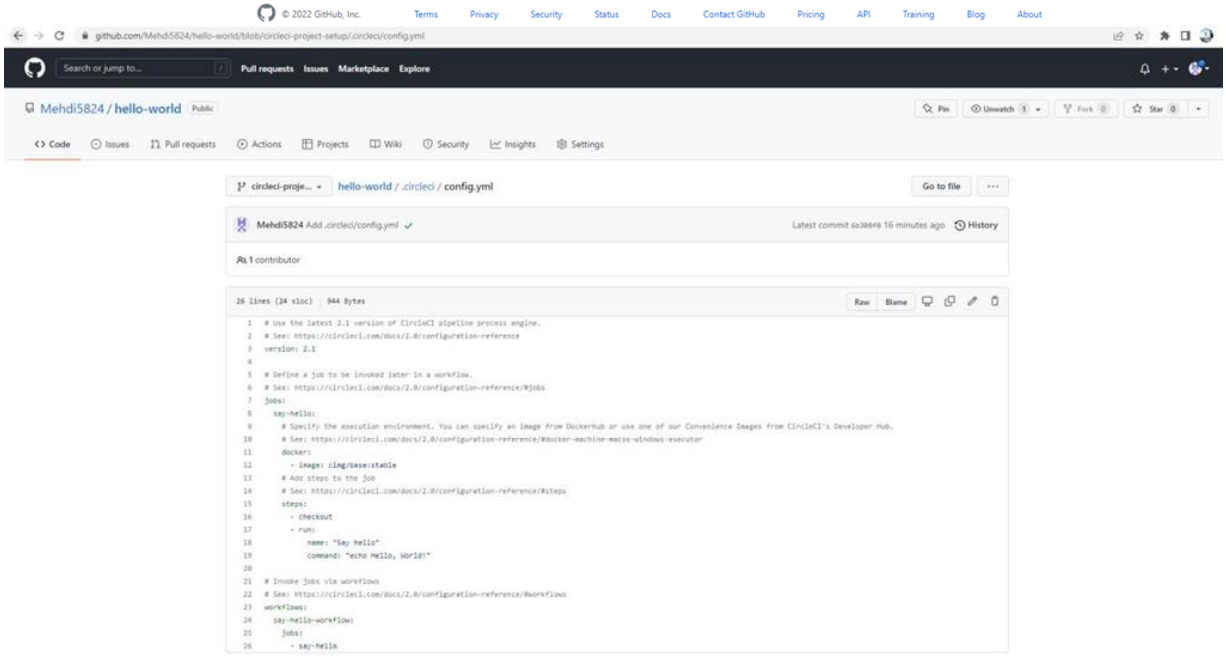


**Break your Build**

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.



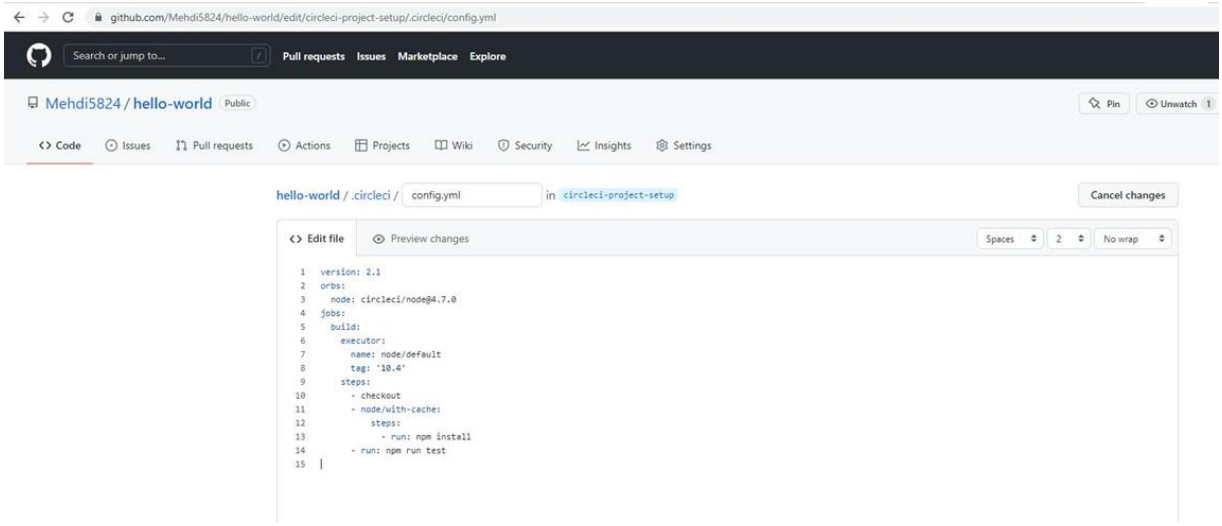



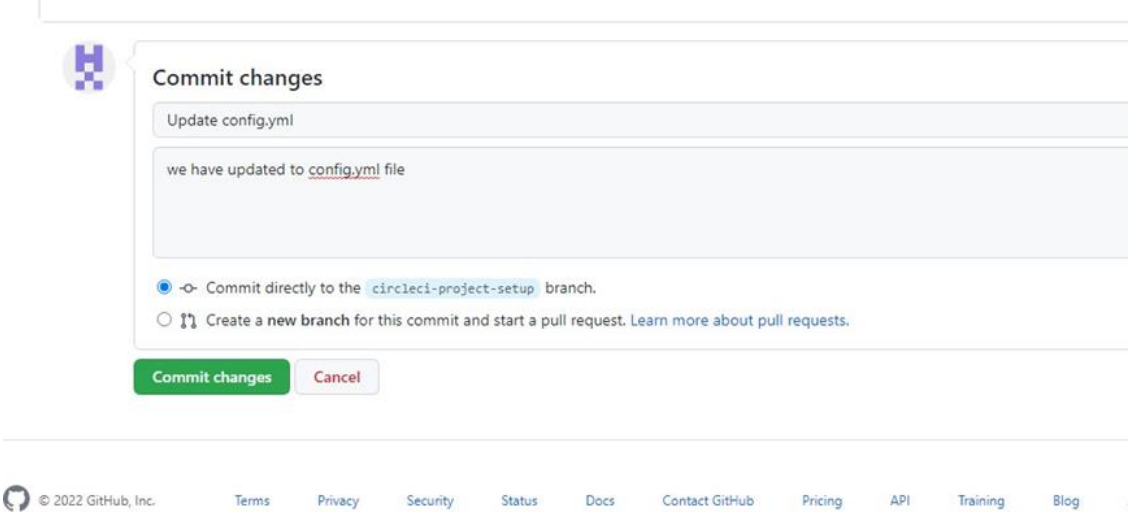
Let’s use the Node orb. Replace the existing config by pasting the following code:

```
version: 2.1
orbs:
  node: cirlceci/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
```

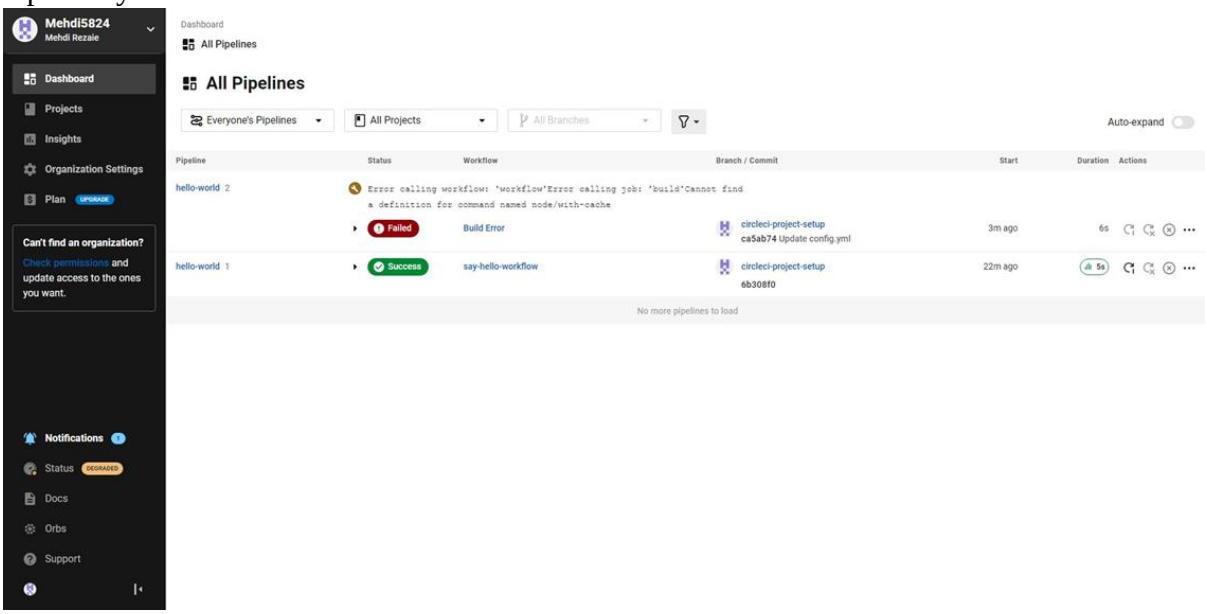
The GitHub file editor should look like this




Scroll down and Commit your changes on GitHub



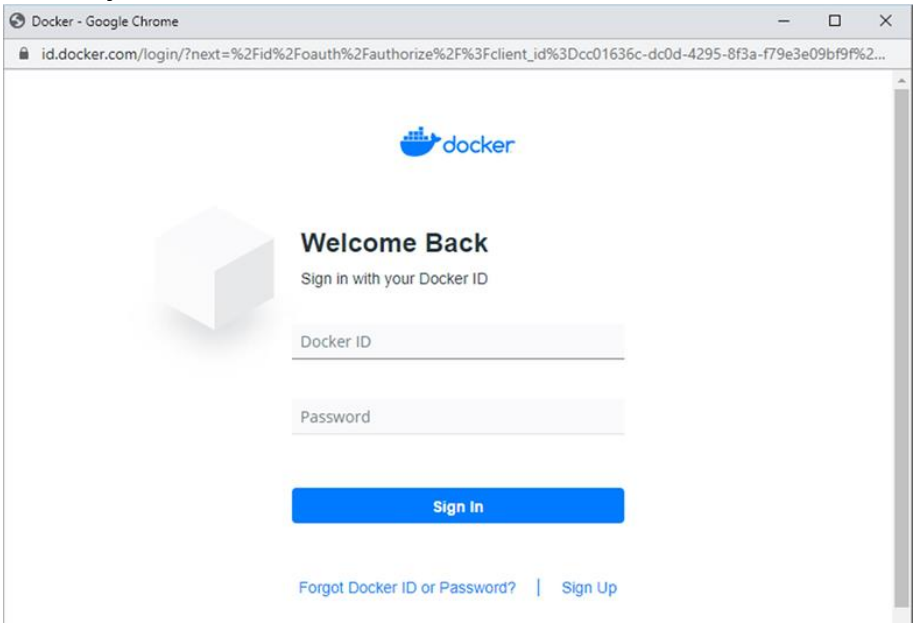
After committing your changes, then return to the Projects page in CircleCI. You should see a new pipeline running… and it will fail! What's going on? The Node orb runs some common Node tasks. Because you are working with an empty repository, running npm run test, a Node script, causes the configuration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



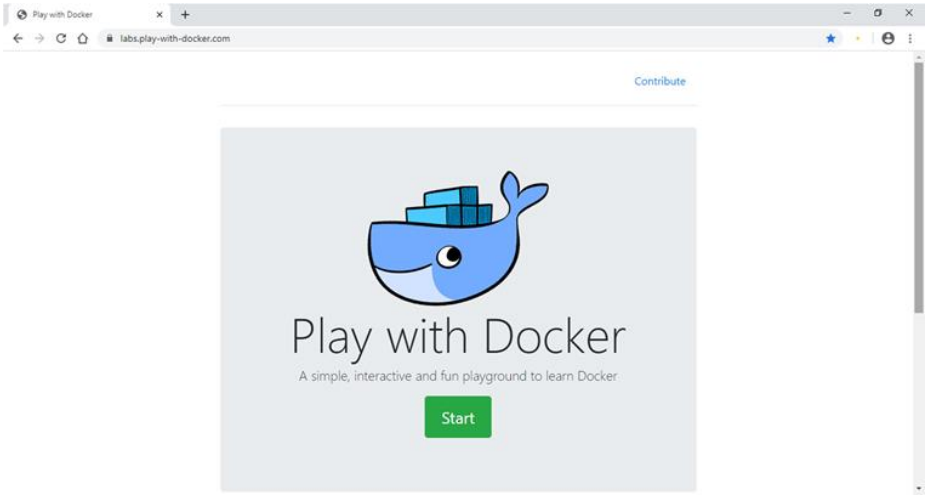
# Practical No. 07

**Aim: Running location service in Docker.**

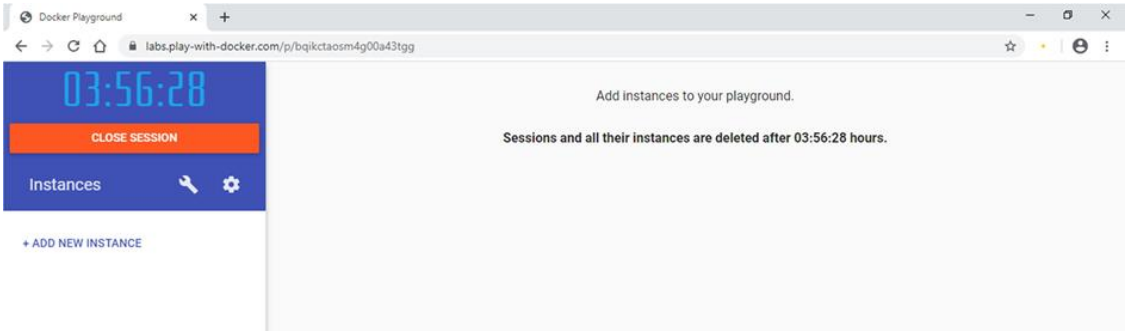
(Create docker hub login first to use it in play with docker)
Now login in to Play-With-Docker

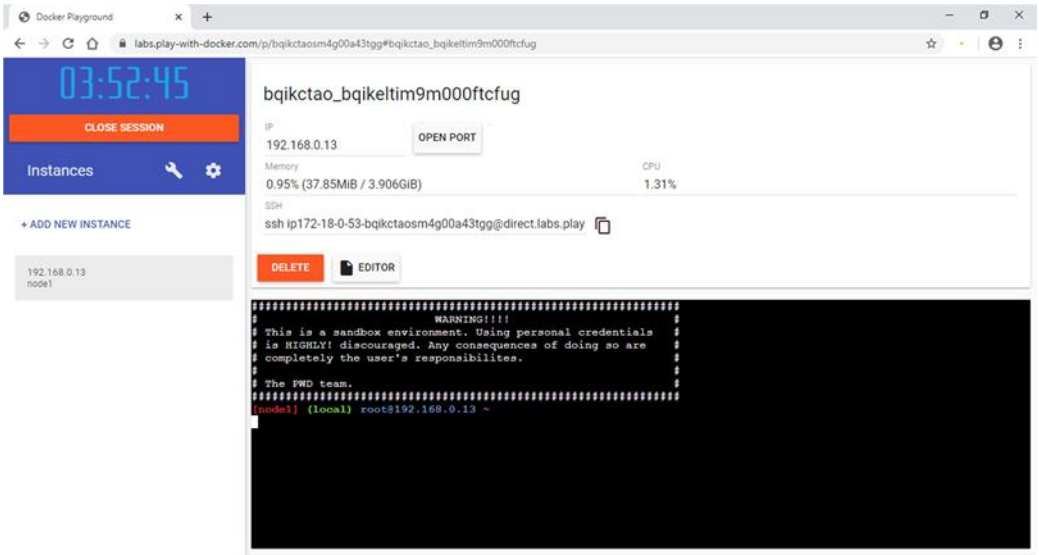



Click on Start.




Click on Add New Instance.



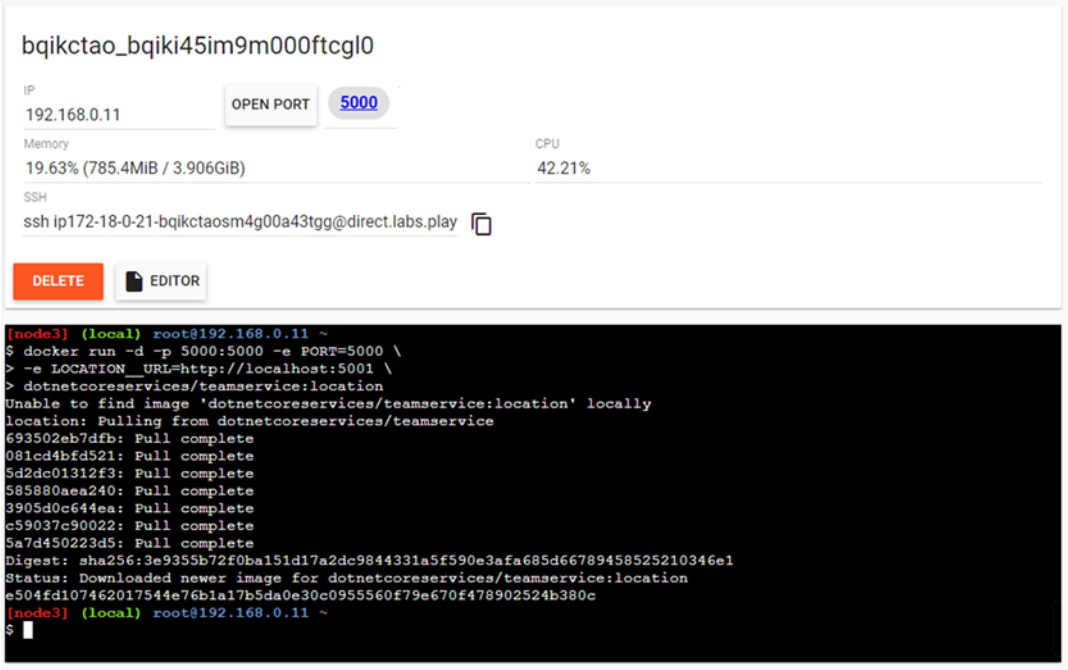



Start typing following commands

Command: To run teamservice

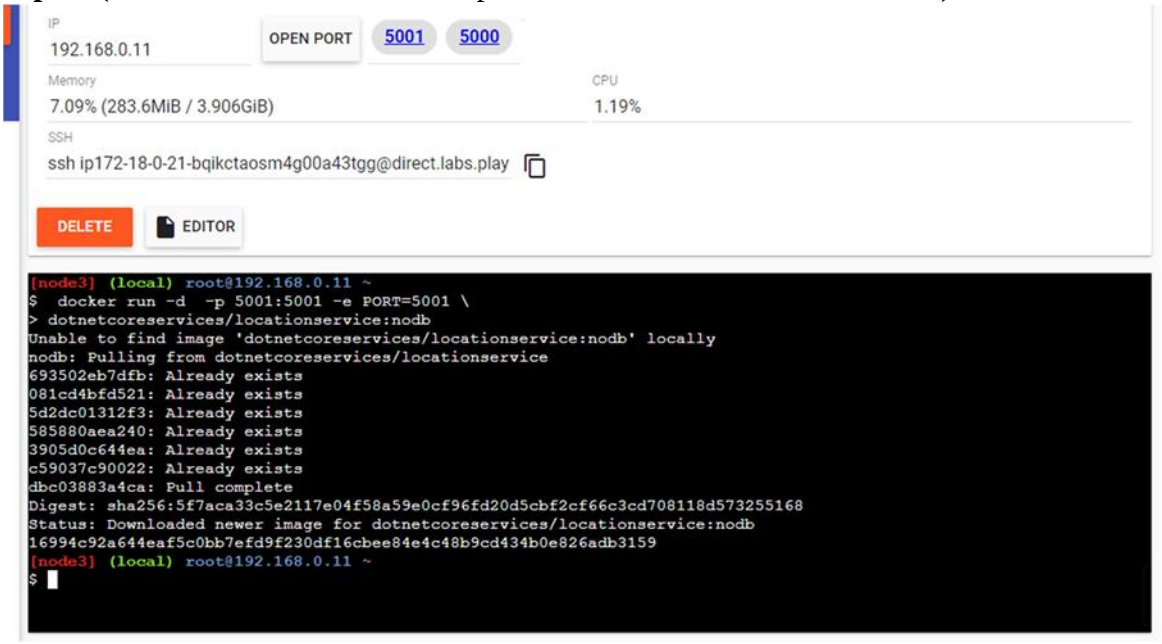
docker run -d -p 5000:5000 -e PORT=5000 \
-e LOCATION__URL=http://localhost:5001 \
dotnetcoreservices/teamservice:location

**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/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
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-9349-3c859
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 ~
$
```