

Index

Sr. No.	Date	Title	Page No.	Signature
1.	19/03/2022	Create a console based ASP.net core application	1	
2.	24/03/2022	Create a MVC Project in ASP.net core	4	
3.	31/03/2022	Usage of Docker Desktop	7	
4.	07/04/2022	Working with Docker	10	
5.	16/04/2022	Building ASP.Net core REST API	14	



Edit with WPS Office

6.	21/04/2022	Working with Circle CI for continuous integration	20	
7.	22/04/2022	Working with Team Service	32	



Edit with WPS Office

Practical No 1

Aim: Create a console based ASP.net core application.

Source Code:

Step 1:

- Download the asp.net core sdk from
<https://dotnet.microsoft.com/learn/dotnet/hello-worldtutorial/install>
- Install the asp.net core sdk.
- To check whether the asp.net sdk is successful install, open command prompt and type command: **dotnet**

```
C:\Users\Shraddha Shah>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.

C:\Users\Shraddha Shah>
```

- To check the version of the dotnet

```
C:\Users\Shraddha Shah>dotnet --version
6.0.202
```

Step 2:

- Go to the drive where you want to create the console application. Create a folder in the drive and go to that folder. Type the following command in the command prompt to create the application.

```
D:\MSA Pracs\prac1>cd..
D:\MSA Pracs>md HelloWorld
D:\MSA Pracs>cd HelloWorld
D:\MSA Pracs\HelloWorld>dotnet new console
The template "Console App" was created successfully.

Processing post-creation actions...
Running 'dotnet restore' on D:\MSA Pracs\HelloWorld\HelloWorld.csproj...
  Determining projects to restore...
  Restored D:\MSA Pracs\HelloWorld\HelloWorld.csproj (in 63 ms).
Restore succeeded.
```

- Restore the project and run the application

```
D:\MSA Pracs>cd HelloWorld
D:\MSA Pracs\HelloWorld>dotnet restore
  Determining projects to restore...
  Restored D:\MSA Pracs\HelloWorld\HelloWorld.csproj (in 25.91 sec).

D:\MSA Pracs\HelloWorld>dotnet run
Hello, World!
```



Step 3:

- Now open **HelloWorld.csproj** file, edit the code

```
<Project Sdk="Microsoft.NET.Sdk">

    <PropertyGroup>
        <OutputType>Exe</OutputType>
        <TargetFramework>net6.0</TargetFramework>
        <ImplicitUsings>enable</ImplicitUsings>
        <Nullable>enable</Nullable>
    </PropertyGroup>
    <ItemGroup>
        <PackageReference Include="Microsoft.AspNetCore.Mvc" Version="1.1.1"/>
        <PackageReference Include="Microsoft.AspNetCore.Server.Kestrel"
Version="1.1.1"/>
        <PackageReference Include="Microsoft.Extensions.Logging"
Version="1.1.1"/>
        <PackageReference Include="Microsoft.Extensions.Logging.Console"
Version="1.1.1"/>
        <PackageReference Include="Microsoft.Extensions.Logging.Debug"
Version="1.1.1"/>
        <PackageReference
Include="Microsoft.Extensions.Configuration.CommandLine" Version="1.1.1"/>
    </ItemGroup>
</Project>
```

- Open **Program.cs** file and edit the code

```
using System;
using Microsoft.AspNetCore.Builder;
using Microsoft.AspNetCore.Hosting;
using Microsoft.Extensions.Logging;
using Microsoft.AspNetCore.Http;
using Microsoft.Extensions.Configuration;
namespace HelloWorld // Note: actual namespace depends on the project name.
{
    internal class Program
    {
        static void Main(string[] args)
        {
            var config = new ConfigurationBuilder()
                .AddCommandLine(args)
                .Build();
            var host = new WebHostBuilder()
                .UseKestrel()
                .UseStartup<Startup>()
                .UseConfiguration(config)
                .Build();
            host.Run();
        }
    }
    public class Startup
    {
```



```
public Startup(IHostingEnvironment env) {}
public void Configure(IApplicationBuilder app, IHostingEnvironment env,
ILoggerFactory loggerFactory)
{
    app.Run(async (context) => { await
context.Response.WriteAsync("Hello, world!");});
}
}
```

Step 4:

Restore the project.

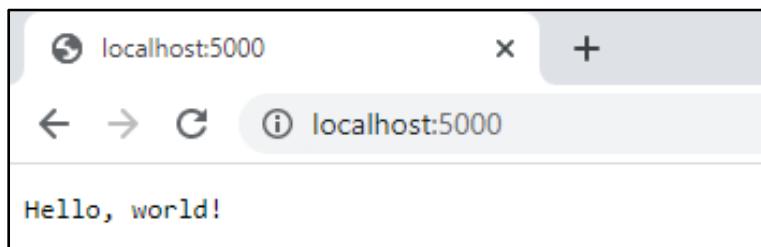
```
D:\MSA Pracs\HelloWorld>dotnet restore
Determining projects to restore...
All projects are up-to-date for restore.
```

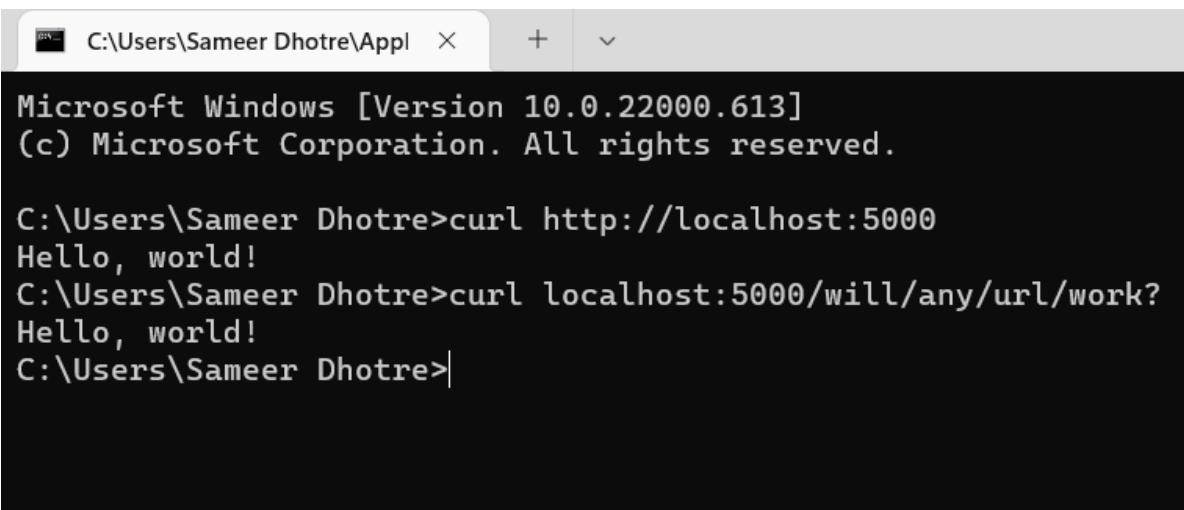
Output:

Run the application

```
D:\MSA Pracs\HelloWorld>dotnet run
Hosting environment: Production
Content root path: D:\MSA Pracs\HelloWorld\bin\Debug\net6.0\
Now listening on: http://localhost:5000
Application started. Press Ctrl+C to shut down.
Application is shutting down...
```

Now open the browser open the url: <http://localhost:5000>





A screenshot of a Windows command prompt window titled "C:\Users\Sameer Dhotre\Appl". The window shows the following text:

```
Microsoft Windows [Version 10.0.22000.613]
(c) Microsoft Corporation. All rights reserved.

C:\Users\Sameer Dhotre>curl http://localhost:5000
Hello, world!
C:\Users\Sameer Dhotre>curl localhost:5000/will/any/url/work?
Hello, world!
C:\Users\Sameer Dhotre>
```

Practical No 2

Aim: Create a MVC Project in ASP.net core

Source Code:

Step 1:

Create a mvc project

dotnet new mvc –auth none

```
D:\Microservices Architecture\Practice Practical\Practs\pracs2>dotnet new mvc --auth none
The template "ASP.NET Core Web App (Model-View-Controller)" was created successfully.
This template contains technologies from parties other than Microsoft, see https://aka.ms/aspnetcore/6.0-third-party-notices for details.

Processing post-creation actions...
Running 'dotnet restore' on D:\Microservices Architecture\Practice Practical\Practs\pracs2\pracs2.csproj...
  Determining projects to restore...
    Restored D:\Microservices Architecture\Practice Practical\Practs\pracs2\pracs2.csproj (in 278 ms).
Restore succeeded.

D:\Microservices Architecture\Practice Practical\Practs\pracs2>
```

Step 2:

Restore, build and run the program.

Use the first url of the command prompt in the browser and see the output

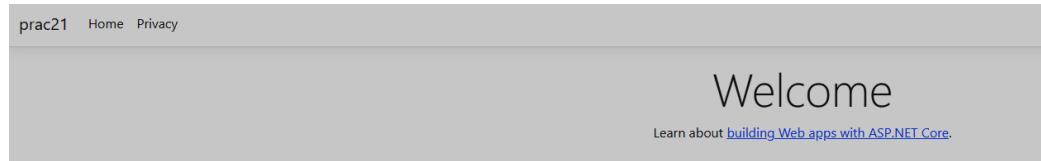
```
D:\Microservices Architecture\Practice Practical\Practs\pracs2>dotnet build
Microsoft (R) Build Engine version 17.1.1+ad02f73656 for .NET
Copyright (C) Microsoft Corporation. All rights reserved.

Determining projects to restore...
All projects are up-to-date for restore.
pracs2 -> D:\Microservices Architecture\Practice Practical\Practs\pracs2\bin\Debug\net6.0\pracs2.dll

Build succeeded.
  0 Warning(s)
  0 Error(s)

Time Elapsed 00:00:04.72

D:\Microservices Architecture\Practice Practical\Practs\pracs2>dotnet run
Building...
info: Microsoft.Hosting.Lifetime[14]
  Now listening on: https://localhost:7091
info: Microsoft.Hosting.Lifetime[14]
  Now listening on: http://localhost:5103
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:\Microservices Architecture\Practice Practical\Practs\pracs2\
```



Step 3:

Go to Models Folder and create StockQuote.cs file in it.

```
using System;
namespace pracs.Models
{
    public class StockQuote
    {
        public string Symbol {get;set;}
        public int Price{get;set;}
    }
}
```

Step 4:

Now go to views folder and then in home folder. Edit the index.cshtml file

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

<div class="text-center">
    <h1 class="display-4">Welcome</h1>
    Symbol: @Model.Symbol <br/>
    Price: $@Model.Price <br/>
</div>
```

Step 5:

Now go to controller folder and edit HomeController.cs

```
using System;
using System.Collections.Generic;
using System.Diagnostics;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using Microsoft.Extensions.Logging;
using pracs2.Models;

namespace pracs2.Controllers;

public class HomeController : Controller
{
    public async Task <ActionResult> Index()
    {
        var model= new StockQuote{ Symbol="Nike", Price=3200};
        return View(model);
    }
}
```

Step 6:

```
D:\Microservices Architecture\Practice Practical\Practs\pracs2>dotnet build
Microsoft (R) Build Engine version 17.1.1+ad02f73656 for .NET
Copyright (C) Microsoft Corporation. All rights reserved.

Determining projects to restore...
All projects are up-to-date for restore.
pracs2 -> D:\Microservices Architecture\Practice Practical\Practs\pracs2\bin\Debug\net6.0\pracs2.dll

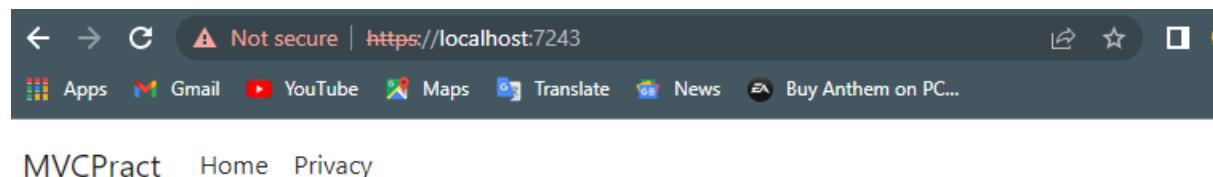
Build succeeded.
  0 Warning(s)
  0 Error(s)

Time Elapsed 00:00:04.31

D:\Microservices Architecture\Practice Practical\Practs\pracs2>dotnet run
Building...
info: Microsoft.Hosting.Lifetime[14]
      Now listening on: https://localhost:7091
info: Microsoft.Hosting.Lifetime[14]
      Now listening on: http://localhost:5103
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:\Microservices Architecture\Practice Practical\Practs\pracs2\
```

Output:

Open the first url in the browser and see the output



MVCPract Home Privacy

Welcome

Symbol: Addidas

Price: \$3200

Practical No 3

Aim: Usage of Docker Desktop

Commands & its output:

Open command prompt

- To check whether docker is installed properly
\$ docker

```
D:\msa>docker

Usage: docker [OPTIONS] COMMAND

A self-sufficient runtime for containers

Options:
  --config string      Location of client config files (default "C:\\Users\\Admin\\.docker")
  -c, --context string Name of the context to use to connect to the daemon (overrides DOCKER_HOST env var and default context set with "docker context use")
  -D, --debug          Enable debug mode
  -H, --host list      Daemon socket(s) to connect to
  -l, --log-level string Set the logging level ("debug"|"info"|"warn"|"error"|"fatal")
                        (default "info")
  --tls                Use TLS; implied by --tlsverify
  --tlscacert string   Trust certs signed only by this CA (default "C:\\Users\\Admin\\.docker\\ca.pem")
  --tlscert string     Path to TLS certificate file (default "C:\\Users\\Admin\\.docker\\cert.pem")
  --tlskey string       Path to TLS key file (default "C:\\Users\\Admin\\.docker\\key.pem")
  --tlsverify          Use TLS and verify the remote
  -v, --version         Print version information and quit

Management Commands:
  builder    Manage builds
  buildx*   Docker Buildx (Docker Inc., v0.8.2)
  compose*  Docker Compose (Docker Inc., v2.4.1)
  config     Manage Docker configs
  container  Manage containers
  context    Manage contexts
  image      Manage images
  manifest   Manage Docker image manifests and manifest lists
  network   Manage networks
  node      Manage Swarm nodes
  plugin    Manage plugins
  sbom*     View the packaged-based Software Bill Of Materials (SBOM) for an image (Anchore Inc., 0.6.0)
  scan*    Docker Scan (Docker Inc., v0.17.0)
  secret    Manage Docker secrets
  service   Manage services
  stack     Manage Docker stacks
  swarm     Manage Swarm
  system    Manage Docker
  trust     Manage trust on Docker images
  volume   Manage volumes
```

- To see the version of the docker
\$ docker -v



```
D:\msa>docker -v
Docker version 20.10.14, build a224086
D:\msa>_
```

- To run hello-world image

```
$ docker run -p 8080:8080 dotnetcoreservices/hello-world
```

```
D:\msa>docker run -p 8080:8080 dotnetcoreservices/hello-world
Hosting environment: Production
Content root path: /pipeline/source/app/publish
Now listening on: http://0.0.0.0:8080
Application started. Press Ctrl+C to shut down.
```

- Run localhost in the browser

```
http://localhost:8080
```



- To see the output in the command prompt

```
$ curl http://localhost:8080/will/itblend?
```

```
CH\ Command Prompt
Microsoft Windows [Version 10.0.19044.1706]
(c) Microsoft Corporation. All rights reserved.

C:\Users\Admin>curl http://localhost:8080/will/it/blend?
Hello, world!

C:\Users\Admin>
```

- To see the images in the docker
\$ docker ps

```
C:\Users\Admin>docker ps
CONTAINER ID        IMAGE               COMMAND             CREATED            STATUS              PORTS               NAMES
3ec44618b74        dockerforcosrviccs/hello World   "/bin/shcnc/source/spl..."   3 minutes ago   Up 3 minutes   0.0.0.0:8080 >8080/tcp   thirty_moneclockv
10cd4394cc14       8cc238cc07cc          "/kube_venkkit_lorw..."   0 minutes ago   Up 0 minutes   K8s vpmkit controller vpmkit controller kube system 3bc0c1d2 code 4180 0094 20x00cc1c30 46
12b20c15246        90189471478           "/storage_provision..."   7 hours ago     Up 7 hours    K8s storage provisioner storage provisioner kube system 3bc0b13b code 4180 4276 2076 335c416e042 4
12b20c15246        90189471478           "/storage_provision..."   7 hours ago     Up 7 hours    K8s storage provisioner storage provisioner kube system 3bc0b13b code 4180 4276 2076 335c416e042 4
43611c6587         8cc238cc07cc          "/conmons contl /etc..."   7 hours ago     Up 7 hours    K8s conmons controller 781cd00978 whmz kube system 010c286 4277 4bc0 cd72 751893268c1d 10
43611c6587         8cc238cc07cc          "/conmons contl /etc..."   7 hours ago     Up 7 hours    K8s kube proxy kube proxy kube system 01bc386 4277 4bc0 cd72 751893268c1d 2
12381c281c28        8119f0007264           "/usr/local/bin/kubco..."   7 hours ago     Up 7 hours    K8s kube controller kubco controller kube system 01bc386 4277 4bc0 cd72 751893268c1d 2
12381c281c28        8119f0007264           "/usr/local/bin/kubco..."   7 hours ago     Up 7 hours    K8s kube controller kubco controller kube system 01bc386 4277 4bc0 cd72 751893268c1d 2
0411b14c023        8cc238cc07cc          "/psuse"               7 hours ago     Up 7 hours    K8s pod psuse 01cd00978 xwhmz kube system 010c286 4277 4bc0 cd72 751893268c1d 10
13871a695cc1       8cc238cc07cc          "/psuse"               7 hours ago     Up 7 hours    K8s pod vpmkit controller kube 3bc0c1d2 code 4180 0094 20x00cc1c36 2
7380cc06e3c3       8cc238cc07cc          "/psuse"               7 hours ago     Up 7 hours    K8s pod vpmkit controller kube 3bc0c1d2 code 4180 0094 20x00cc1c36 2
44621d5587         8cc238cc07cc          "/psuse"               7 hours ago     Up 7 hours    K8s pod vpmkit controller kube 3bc0c1d2 code 4180 0094 20x00cc1c36 2
51b7841c402d       664461818864           "/code convertific cl..."   7 hours ago     Up 7 hours    K8s ctcd ctcd docker docker kube system e5421dc444251c0651a62cd1cda1c 2
15329508467        6595cc06e177         "/kube episcrver .ed..."   7 hours ago     Up 7 hours    K8s kube episcrver kube episcrver docker docker kube system 3cc03021716e8470271c37887418b712 2
6821d1cd5c54       64188cc8864           "/kube controller run..."   7 hours ago     Up 7 hours    K8s kube controller manager docker docker kube system 3cc1166c131dd1c1c6735cc218b77
...
3c321d5587         038681c2d252           "/kube scheduler ou..."   7 hours ago     Up 7 hours    K8s kube scheduler kube scheduler docker docker kube system d10e80707cc00c28618bc058c1cde2c40 2
3c1b5414c028        8cc238cc07cc          "/psuse"               7 hours ago     Up 7 hours    K8s pod ctcd docker docker kube system d5422d0d4441c5100953202d1c0421c 2
3c1b5414c028        8cc238cc07cc          "/psuse"               7 hours ago     Up 7 hours    K8s pod vpmkit controller kube 3bc0c1d2 code 4180 0094 20x00cc1c36 2
3c1b5414c028        8cc238cc07cc          "/psuse"               7 hours ago     Up 7 hours    K8s pod vpmkit controller kube 3bc0c1d2 code 4180 0094 20x00cc1c36 2
3c1b5414c028        8cc238cc07cc          "/psuse"               7 hours ago     Up 7 hours    K8s pod kube controller manager docker docker kube system 3cc1166c131dd1c1c6735cc218b77cc 2
3c1b5414c028        8cc238cc07cc          "/psuse"               7 hours ago     Up 7 hours    K8s pod kube scheduler docker kube system e50c80707cc00c28618bc058c1cde2c40 2
```

- To terminate the image in the docker.
note the container id of the docker that you want to terminal and replace the <Containerid> in the below command

\$ docker kill <containerid>

```
C:\Users\Admin>docker kill 35c840e18b74
35c840e18b74
```

To check whether the docker is terminated or not

\$ docker ps

```
C:\Users\Admin>docker ps
CONTAINER ID        IMAGE               COMMAND             CREATED            STATUS              PORTS               NAMES
3ec44618b74        dockerforcosrviccs/hello World   "/bin/shcnc/source/spl..."   3 minutes ago   Up 3 minutes   thirty_moneclockv
10cd4394cc14       8cc238cc07cc          "/kube_venkkit_lorw..."   0 minutes ago   Up 0 minutes   K8s vpmkit controller vpmkit controller kube system 3bc0c1d2 code 4180 0094 20x00cc1c30 46
12b20c15246        90189471478           "/storage_provision..."   7 hours ago     Up 7 hours    K8s storage provisioner storage provisioner kube system 3bc0b13b code 4180 4276 2076 335c416e042 4
12b20c15246        90189471478           "/storage_provision..."   7 hours ago     Up 7 hours    K8s storage provisioner storage provisioner kube system 3bc0b13b code 4180 4276 2076 335c416e042 4
43611c6587         8cc238cc07cc          "/conmons contl /etc..."   7 hours ago     Up 7 hours    K8s conmons controller 781cd00978 whmz kube system 010c286 4277 4bc0 cd72 751893268c1d 10
43611c6587         8cc238cc07cc          "/conmons contl /etc..."   7 hours ago     Up 7 hours    K8s kube proxy kube proxy kube system 01bc386 4277 4bc0 cd72 751893268c1d 2
12381c281c28        8119f0007264           "/usr/local/bin/kubco..."   7 hours ago     Up 7 hours    K8s kube controller kubco controller kube system 01bc386 4277 4bc0 cd72 751893268c1d 2
12381c281c28        8119f0007264           "/usr/local/bin/kubco..."   7 hours ago     Up 7 hours    K8s kube controller kubco controller kube system 01bc386 4277 4bc0 cd72 751893268c1d 2
0411b14c023        8cc238cc07cc          "/psuse"               7 hours ago     Up 7 hours    K8s pod psuse 01cd00978 xwhmz kube system 010c286 4277 4bc0 cd72 751893268c1d 10
13871a695cc1       8cc238cc07cc          "/psuse"               7 hours ago     Up 7 hours    K8s pod vpmkit controller kube 3bc0c1d2 code 4180 0094 20x00cc1c36 2
7380cc06e3c3       8cc238cc07cc          "/psuse"               7 hours ago     Up 7 hours    K8s pod vpmkit controller kube 3bc0c1d2 code 4180 0094 20x00cc1c36 2
44621d5587         8cc238cc07cc          "/psuse"               7 hours ago     Up 7 hours    K8s pod vpmkit controller kube 3bc0c1d2 code 4180 0094 20x00cc1c36 2
51b7841c402d       664461818864           "/code convertific cl..."   7 hours ago     Up 7 hours    K8s ctcd ctcd docker docker kube system e5421dc444251c0651a62cd1cda1c 2
15329508467        6595cc06e177         "/kube episcrver .ed..."   7 hours ago     Up 7 hours    K8s kube episcrver kube episcrver docker docker kube system 3cc03021716e8470271c37887418b712 2
6821d1cd5c54       64188cc8864           "/kube controller run..."   7 hours ago     Up 7 hours    K8s kube controller manager docker docker kube system 3cc1166c131dd1c1c6735cc218b77
...
3c321d5587         038681c2d252           "/kube scheduler ou..."   7 hours ago     Up 7 hours    K8s kube scheduler kube scheduler docker docker kube system d10e80707cc00c28618bc058c1cde2c40 2
3c1b5414c028        8cc238cc07cc          "/psuse"               7 hours ago     Up 7 hours    K8s pod ctcd docker docker kube system d5422d0d4441c5100953202d1c0421c 2
3c1b5414c028        8cc238cc07cc          "/psuse"               7 hours ago     Up 7 hours    K8s pod vpmkit controller kube 3bc0c1d2 code 4180 0094 20x00cc1c36 2
3c1b5414c028        8cc238cc07cc          "/psuse"               7 hours ago     Up 7 hours    K8s pod vpmkit controller kube 3bc0c1d2 code 4180 0094 20x00cc1c36 2
3c1b5414c028        8cc238cc07cc          "/psuse"               7 hours ago     Up 7 hours    K8s pod kube controller manager docker docker kube system 3cc1166c131dd1c1c6735cc218b77cc 2
3c1b5414c028        8cc238cc07cc          "/psuse"               7 hours ago     Up 7 hours    K8s pod kube scheduler docker kube system e50c80707cc00c28618bc058c1cde2c40 2
```



Practical No 4

Aim: Working with Docker

Commands and its output:

Step 1:

Create a account in the docker hub. Remember the username and password of the account

Step 2:

- Now to go <https://labs.play-with-docker.com/> and click on **Start** button.
- Click on **Add New Instance**. You will see the editor open in the right pane. Give the commands in the editor

Step 3:

- To check the version of the docker

```
$ docker --version
```

```
[node1] (local) root@192.168.0.18 ~
$ docker --version
Docker version 20.10.0, build 7287ab3
[node1] (local) root@192.168.0.18 ~
```

- To pull the readymade image

```
$ docker pull hello-world
```

```
$ docker pull hello-world
Using default tag: latest
latest: Pulling from library/hello-world
2db29710123e: Pull complete
Digest: sha256:80f31dalac7b312ba29d65080fddf797dd76acfb870e677f390d5acba9741b17
Status: Downloaded newer image for hello-world:latest
docker.io/library/hello-world:latest
[node1] (local) root@192.168.0.18 ~
```

- To check the images in docker

```
$ docker images
```

```
$ docker images
REPOSITORY      TAG          IMAGE ID      CREATED        SIZE
hello-world    latest        feb5d9fea6a5   7 months ago   13.3kB
[node1] (local) root@192.168.0.18 ~
```



Part 1: To pull and Push images in docker

Step 4:

- Open the new tab in the browser and login to hub.docker.com
- Click on **Repositories** and then click on **Create Repositories**
- Give the name of the repository as “repo1” and in description add “My first repository”
- Make visibility as **Private**
- And now click on **Create** button and check whether the repository is created or not.

Step 5:

- Now come to the <https://labs.play-with-docker.com/> and give the following command
- Login into docker account

```
$ docker login -username= your_user_name  
password:
```

```
[node1] (local) root@192.168.0.18 ~  
$ docker login --username=vishwakarma1919  
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
```

Note: Give your username and password that you have used to login to hub.docker.com

- To tag an image in docker

```
$ docker tag <image id> <username>/repo1:firsttry
```

```
[node1] (local) root@192.168.0.18 ~  
$ docker tag feb5d9fea6a5 vishwakarma1919/repo1:firsttry  
[node1] (local) root@192.168.0.18 ~  
$
```

- To push the image to docker account



```
$ docker push <username>/repo1:firsttry
```

```
[node1] (local) root@192.168.0.18 ~
$ docker push vishwakarma1919/repo1:firsttry
The push refers to repository [docker.io/vishwakarma1919/repo1]
e07ee1baac5f: Mounted from library/hello-world
firsttry: digest: sha256:f54a58bc1aac5ea1a25d796ae155dc228b3f0e11d046ae276b39c4bf2f13d8c4 size: 525
[node1] (local) root@192.168.0.18 ~
s
```

Note: firsttry is tag name created above.

- Check it in hub.docker.com now in tags tab

The screenshot shows a table with four columns: TAG, OS, PULLED, and PUSHED. The 'TAG' column contains 'firsttry'. The 'OS' column contains a small icon of a Linux kernel. The 'PULLED' column contains '---'. The 'PUSHED' column contains '4 minutes ago'. Above the table, there is a message: 'This repository contains 1 tag(s.)'. To the right of the table, there is a 'VULNERABILITY SCANNING - DISABLED' section with a 'Enable' button.

TAG	OS	PULLED	PUSHED
firsttry		---	4 minutes ago

[See all](#)

Part 2: Build and image and then push and run in the docker

Step 6:

- In <https://labs.play-with-docker.com/> give the following command

```
cat > Dockerfile <<EOF
FROM busybox
CMD echo "Hello world! This is my first Docker image."
EOF
```

```
[node1] (local) root@192.168.0.18 ~
$ cat > Dockerfile <<EOF
> FROM busybox
> CMD echo "Hello World! This Is My First Docker Image."
> EOF
```



- To build the image from docker file

```
$ docker build -t <username>/repo2 .
```

```
[node1] (local) root@192.168.0.18 ~
$ docker build -t vishwakarma1919/repo2 .
Sending build context to Docker daemon    47MB
Step 1/2 : FROM busybox
latest: Pulling from library/busybox
50e8d59317eb: Pull complete
Digest: sha256:d2b53584f580310186df7a2055ce3ff83cc0df6caacf1e3489bff8cf5d0af5d8
Status: Downloaded newer image for busybox:latest
--> 1a80408de790
Step 2/2 : CMD echo "Hello World! This Is My First Docker Image."
--> Running in 523badc76755
Removing intermediate container 523badc76755
--> 58a88ef19a6a
Successfully built 58a88ef19a6a
Successfully tagged vishwakarma1919/repo2:latest
[node1] (local) root@192.168.0.18 ~
c [
```

- Check images in docker

```
$ docker images
```

```
[node1] (local) root@192.168.0.18 ~
$ docker images
REPOSITORY          TAG      IMAGE ID      CREATED       SIZE
vishwakarma1919/repo2  latest   58a88ef19a6a  26 seconds ago  1.24MB
busybox              latest   1a80408de790  4 weeks ago   1.24MB
hello-world          latest   feb5d9fea6a5  7 months ago   13.3kB
vishwakarma1919/repo1 firsttry  feb5d9fea6a5  7 months ago   13.3kB
[node1] (local) root@192.168.0.18 ~
c [
```

- To push the image on the docker hub

```
$ docker push <username>/repo2.
```

```
[node2] (local) root@192.168.0.8 ~
$ docker push vishwakarma1919/repo2
Using default tag: latest
The push refers to repository [docker.io/vishwakarma1919/repo2]
eb6b01329ebe: Mounted from library/busybox
latest: digest: sha256:4452bb83a562a0ce6a5e1fa11159957b8ad3cc62dffb6ad14b60dd4e5dd29bf3 size: 527
c [
```

- Check it in hub.docker.com now in tags tab



- Come back to the <https://labs.play-with-docker.com/> and give the below command to run the docker image

```
$ docker run <username>/repo2
```

```
[node2] (local) root@192.168.0.8 ~
$ docker run vishwakarma1919/repo2
Hello world! This is My First Docker Image
[node2] (local) root@192.168.0.8 ~
$
```

- Close the session

Practical No 5

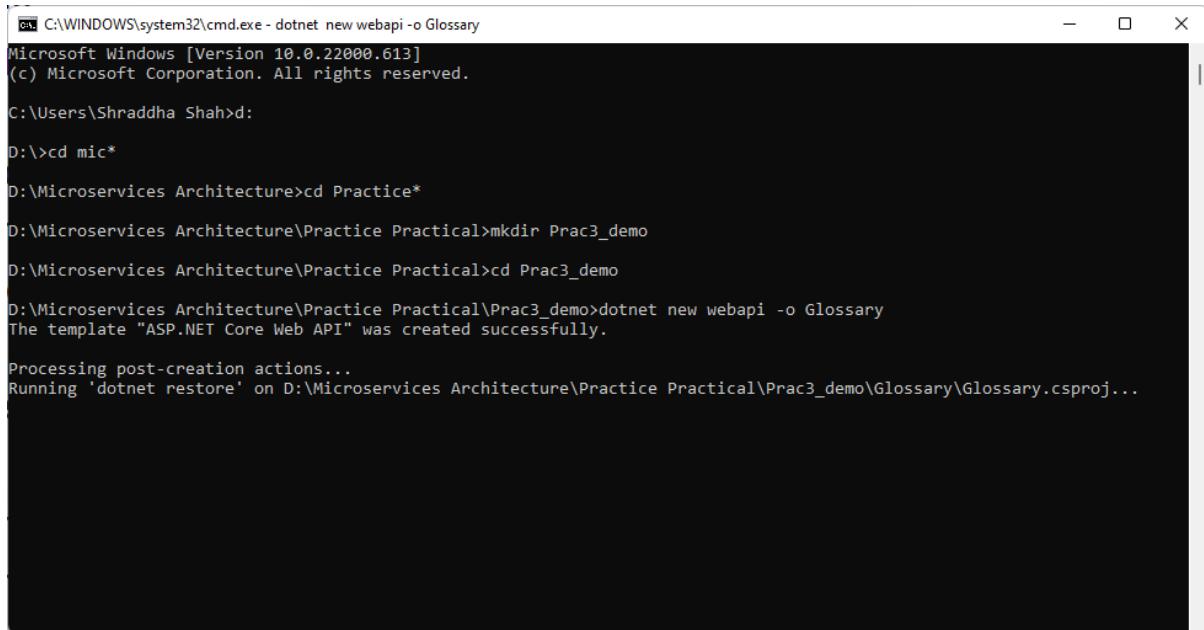
Aim: Building ASP.Net core REST API

Source Code:

Step 1: Create a webAPI

Open command prompt and give the command

dotnet new webapi -o Glossary

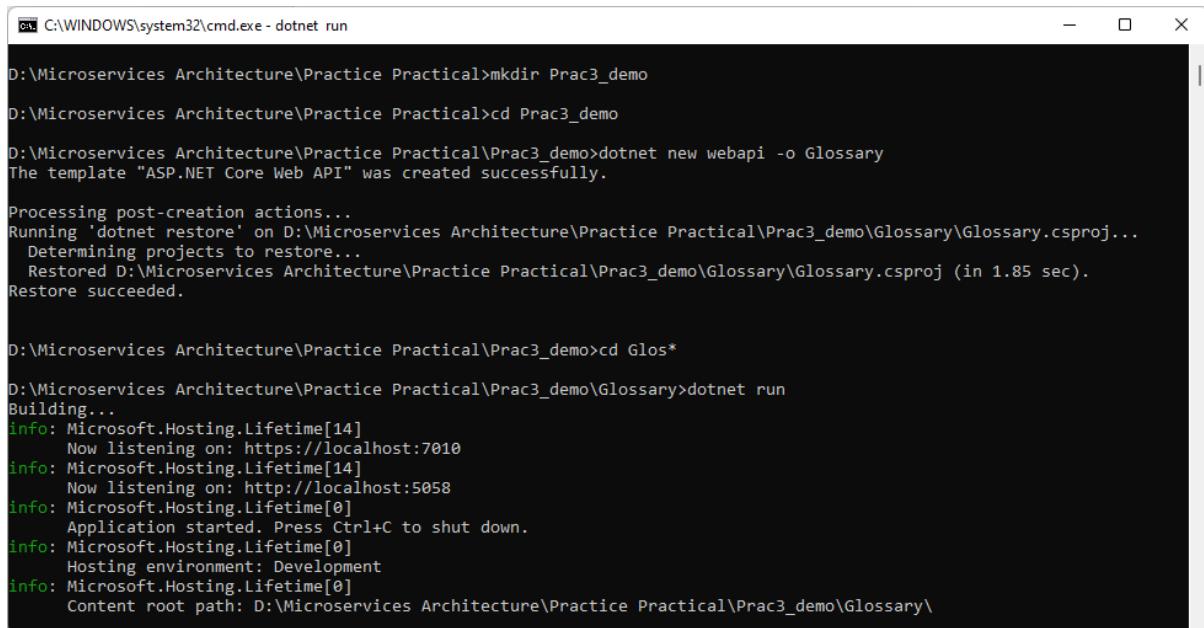


```
C:\WINDOWS\system32\cmd.exe - dotnet new webapi -o Glossary
Microsoft Windows [Version 10.0.22000.613]
(c) Microsoft Corporation. All rights reserved.

C:\Users\Shraddha Shah>d:
D:>cd mic*
D:\Microservices Architecture>cd Practice*
D:\Microservices Architecture\Practice Practical>mkdir Prac3_demo
D:\Microservices Architecture\Practice Practical>cd Prac3_demo
D:\Microservices Architecture\Practice Practical\Prac3_demo>dotnet new webapi -o Glossary
The template "ASP.NET Core Web API" was created successfully.

Processing post-creation actions...
Running 'dotnet restore' on D:\Microservices Architecture\Practice Practical\Prac3_demo\Glossary\Glossary.csproj...
```

Now enter into the glossary folder and then run the project



```
C:\WINDOWS\system32\cmd.exe - dotnet run

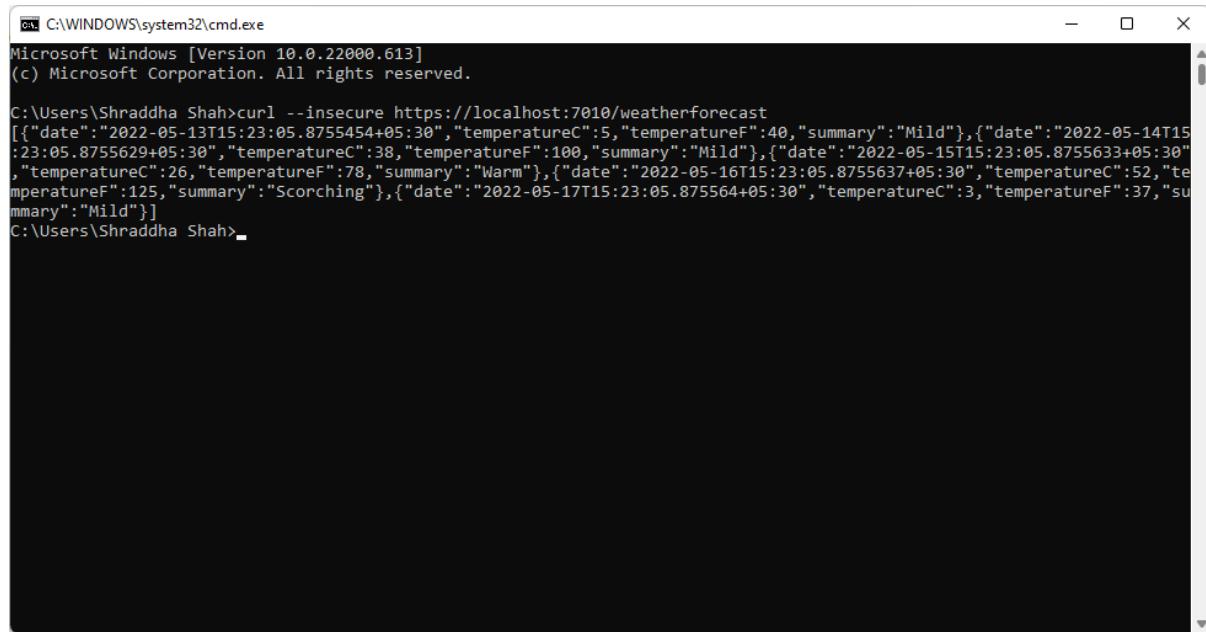
D:\Microservices Architecture\Practice Practical>mkdir Prac3_demo
D:\Microservices Architecture\Practice Practical>cd Prac3_demo
D:\Microservices Architecture\Practice Practical\Prac3_demo>dotnet new webapi -o Glossary
The template "ASP.NET Core Web API" was created successfully.

Processing post-creation actions...
Running 'dotnet restore' on D:\Microservices Architecture\Practice Practical\Prac3_demo\Glossary\Glossary.csproj...
Determining projects to restore...
Restored D:\Microservices Architecture\Practice Practical\Prac3_demo\Glossary\Glossary.csproj (in 1.85 sec).
Restore succeeded.

D:\Microservices Architecture\Practice Practical\Prac3_demo>cd Glos*
D:\Microservices Architecture\Practice Practical\Prac3_demo\Glossary>dotnet run
Building...
Info: Microsoft.Hosting.Lifetime[14]
  Now listening on: https://localhost:7010
Info: Microsoft.Hosting.Lifetime[14]
  Now listening on: http://localhost:5058
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:\Microservices Architecture\Practice Practical\Prac3_demo\Glossary\
```

Step 2: Open another command prompt & give curl command to view the output





```
C:\Windows\system32\cmd.exe
Microsoft Windows [Version 10.0.22000.613]
(c) Microsoft Corporation. All rights reserved.

C:\Users\Shraddha Shah>curl --insecure https://localhost:7010/weatherforecast
[{"date": "2022-05-13T15:23:05.8755454+05:30", "temperatureC": 5, "temperatureF": 40, "summary": "Mild"}, {"date": "2022-05-14T15:23:05.8755629+05:30", "temperatureC": 15, "temperatureF": 59, "summary": "Mild"}, {"date": "2022-05-15T15:23:05.8755633+05:30", "temperatureC": 26, "temperatureF": 78, "summary": "Warm"}, {"date": "2022-05-16T15:23:05.8755637+05:30", "temperatureC": 32, "temperatureF": 125, "summary": "Scorching"}, {"date": "2022-05-17T15:23:05.875564+05:30", "temperatureC": 3, "temperatureF": 37, "summary": "Mild"}]
C:\Users\Shraddha Shah>
```

Step 3: Delete the weatherforecast.cs from the Glossary Folder i.e root folder and also from the Controller Folder.

Step 4: Create a class file in the Glossary folder named “GlossaryItem.cs”

```
namespace Glossary
{
    public class GlossaryItem
    {
        public string Term { get; set; }
        public string Definition { get; set; }
    }
}
```

Step 5: Create a class file in the Controller folder named “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 non-existing term.");
    }
    else
    {
        existingGlossaryItem.Definition = glossaryItem.Definition;
        return Ok();
    }
}
```



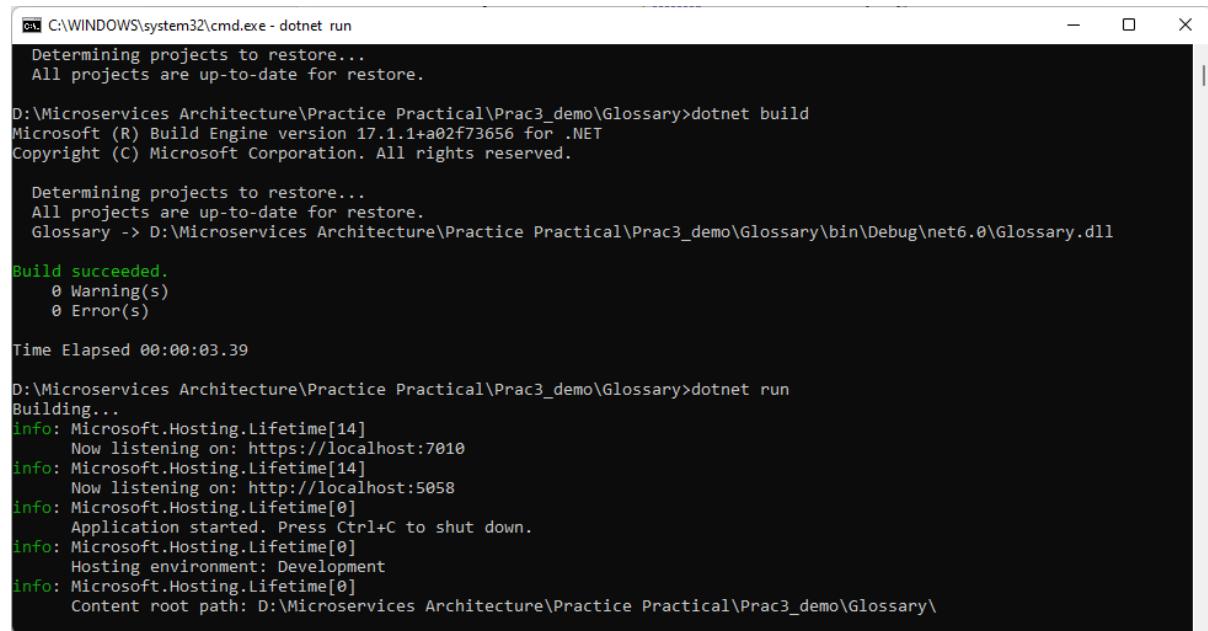
```
        }

    }

    [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();
        }
    }
}
```

Step 6: To stop the application running on command prompt do **Ctrl+c**

Step 7: Now restore, build and then run the program



```
C:\WINDOWS\system32\cmd.exe - dotnet run
Determining projects to restore...
All projects are up-to-date for restore.

D:\Microservices Architecture\Practice Practical\Prac3_demo\Glossary>dotnet build
Microsoft (R) Build Engine version 17.1.1+ad02f73656 for .NET
Copyright (C) Microsoft Corporation. All rights reserved.

Determining projects to restore...
All projects are up-to-date for restore.
Glossary -> D:\Microservices Architecture\Practice Practical\Prac3_demo\Glossary\bin\Debug\net6.0\Glossary.dll

Build succeeded.
  0 Warning(s)
  0 Error(s)

Time Elapsed 00:00:03.39

D:\Microservices Architecture\Practice Practical\Prac3_demo\Glossary>dotnet run
Building...
Info: Microsoft.Hosting.Lifetime[14]
  Now listening on: https://localhost:7010
Info: Microsoft.Hosting.Lifetime[14]
  Now listening on: http://localhost:5058
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:\Microservices Architecture\Practice Practical\Prac3_demo\Glossary\
```

Output:

Open the other command prompt and give the following command.

Kindly note the port number that you will get in the previous command prompt and change the port number in the curl

1. Getting the List of Items

```
curl --insecure https://localhost:7010/api/glossary
```



2. Getting Single Item

- a. curl --insecure <https://localhost:7010/api/glossary/MVC>
- b. curl --insecure <https://localhost:7010/api/glossary/HTML>
- c. curl --insecure <https://localhost:7010/api/glossary/OpenID>

3. Creating an item

```
curl --insecure -X POST -d "{\"term\": \"MFA\", \"definition\": \"An authentication process.\"} -H \"Content-Type:application/json\" https://localhost:7010/api/glossary
```

4. Updating an Item

```
curl --insecure -X PUT -d "{\"term\": \"MVC\", \"definition\": \"Modified record of Model View Controller.\"} -H \"Content-Type:application/json\" https://localhost:7010/api/glossary
```

5. Delete an Item

```
curl --insecure --request DELETE --url https://localhost:7010/api/glossary/openid
```

Output:-

Open the other command prompt and give the following command.

Kindly note the port number that you will get in the previous command prompt and change the port number in the curl

1. Getting the List of Items

```
curl --insecure https://localhost:7010/api/glossary
```

```
D:\>curl --insecure https://localhost:7136/api/glossary
[{"term": "HTML", "definition": "Hypertext Markup Language"}, {"term": "MVC", "definition": "Model View Controller"}, {"term": "OpenID", "definition": "An open standard for authentication"}]
D:\>
```

2. Getting Single Item



a. curl --insecure <https://localhost:7010/api/glossary/MVC>

```
D:\>curl --insecure https://localhost:7136/api/glossary/MVC
>{"term":"MVC","definition":"Model View Controller"}
```

b. curl --insecure <https://localhost:7010/api/glossary/HTML>

```
D:\>curl --insecure https://localhost:7136/api/glossary/HTML
>{"term":"HTML","definition":"Hypertext Markup Language"}
D:\>
```

c. curl --insecure <https://localhost:7010/api/glossary/OpenID>

```
D:\>curl --insecure https://localhost:7136/api/glossary/OpenID
>{"term":"OpenID","definition":"An open standard for authentication"}
D:\>
```

3. Creating an item

curl --insecure -X POST -d "{\"term\": \"MFA\", \"definition\": \"An authentication process.\"}" -H "Content-Type:application/json" <https://localhost:7010/api/glossary>

```
D:\>curl --insecure -X POST -d "{\"term\": \"MFA\", \"definition\": \"An authentication process.\"}" -H "Content-Type:application/json" https://localhost:7136/api/glossary
>{"term":"MFA","definition":"An authentication process."}
D:\>
```

4. Updating an Item

curl --insecure -X PUT -d "{\"term\": \"MVC\", \"definition\": \"Modified record of Model View Controller.\"}" -H "Content-Type:application/json" <https://localhost:7010/api/glossary>

```
curl --insecure -X PUT -d "{\"term\": \"MVC\", \"definition\": \"Modified record of Model View Controller.\"}" -H "Content-Type:application/json" https://localhost:7136/api/glossary
{"term":"MVC","definition":"Modified record of Model View Controller."}
```

5. Delete an Item

curl --insecure --request DELETE --url <https://localhost:7010/api/glossary/openid>

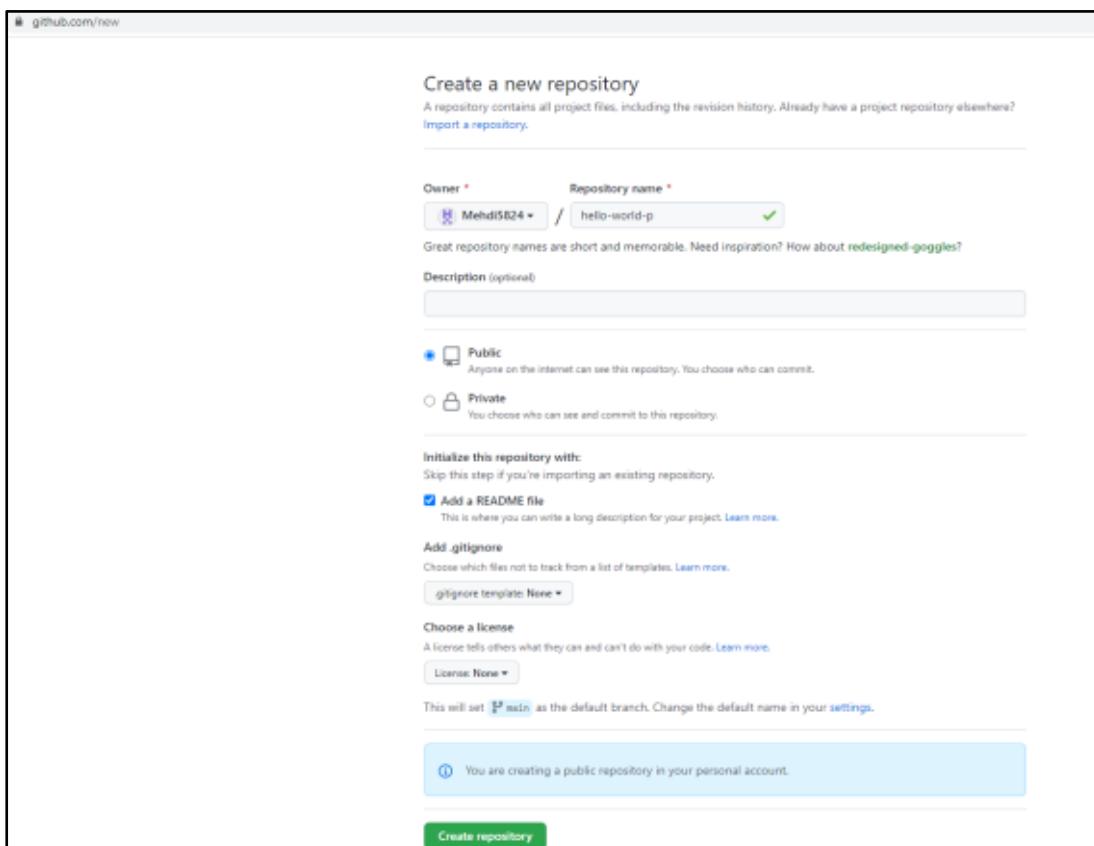
Practical No 6

Aim: Working with Circle CI for continuous integration

Steps and its output:

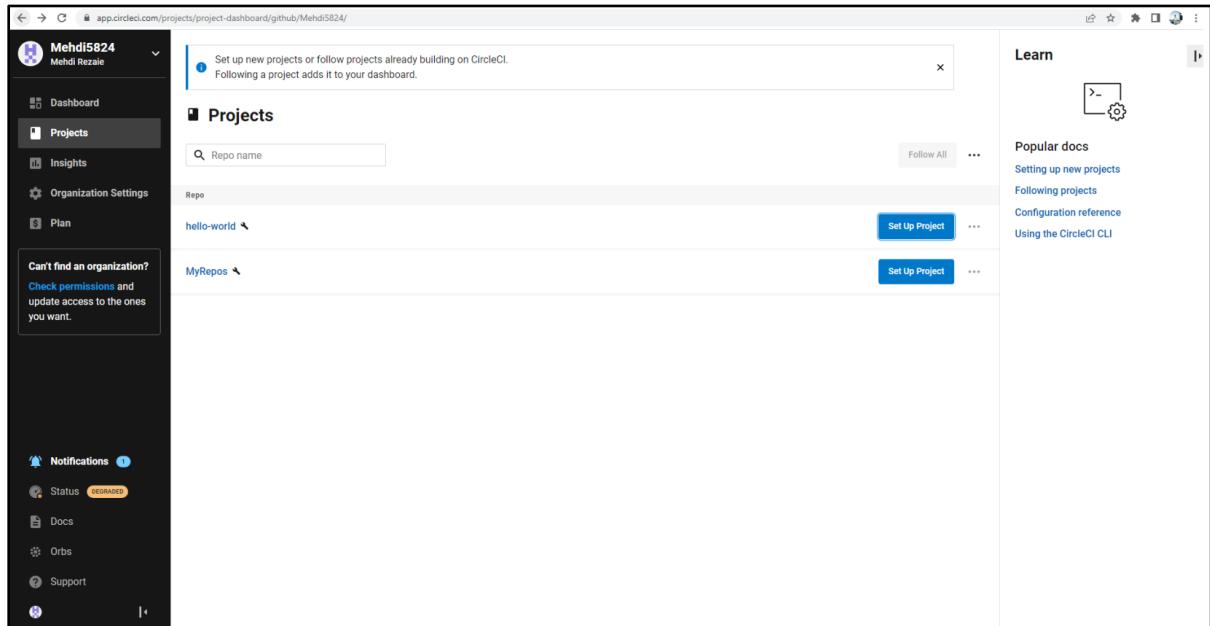
Step 1: Create a repository

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.

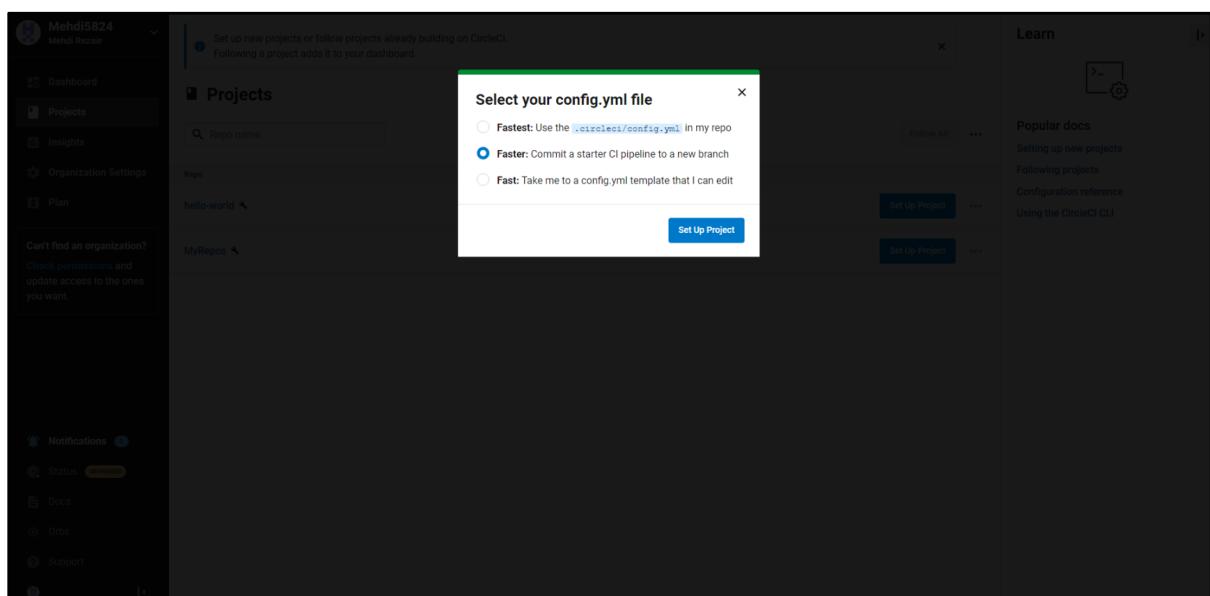


Step 2: Set up CircleCI

- Login to Circle CI <https://app.circleci.com/> using GitHub Login
- Navigate to the CircleCI Projects page. If you created your new repository under an organization, you will need to select the organization name.



- You will be taken to the Projects dashboard. On the dashboard, select the project you want to set up (hello-world).
- 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`.



Step 3: Your first pipeline

- On your project's pipeline page, click the green Success button, which brings you to the workflow that ran (say-helloworld).
- Within this workflow, the pipeline ran one job, called say-hello. Click say-hello to see the steps in this job:
 - Spin up environment
 - Preparing environment variables
 - Checkout code
 - Say hello
- Now select the “say-hello-workflow”

The screenshot shows the CircleCI web interface. In the top left, there's a user profile for 'Mehdi5824' and 'Mehti Rezaie'. The main navigation bar includes 'Dashboard', 'Projects', 'Insights', 'Organization Settings', and 'Plan' (with an 'UPGRADE' button). A sidebar on the left lists 'Notifications' (1), 'Status' (DEGRADED), 'Docs', 'Orbs', 'Support', and a 'Logout' button.

The central area displays the 'hello-world' pipeline under the 'Project' section. It shows a single run of the 'hello-world' pipeline, which has a status of 'Success' and was triggered by 'circleci-project-setup' commit '6b308f0' 3 minutes ago. The duration of the run is listed as '5s'. There are buttons for 'Edit Config', 'Trigger Pipeline', and 'Project Settings'.

Below the pipeline details, there's a message: 'Can't find an organization? Check permissions and update access to the ones you want.' The bottom of the interface features a footer with the WPS Office logo and the text 'Edit with WPS Office'.

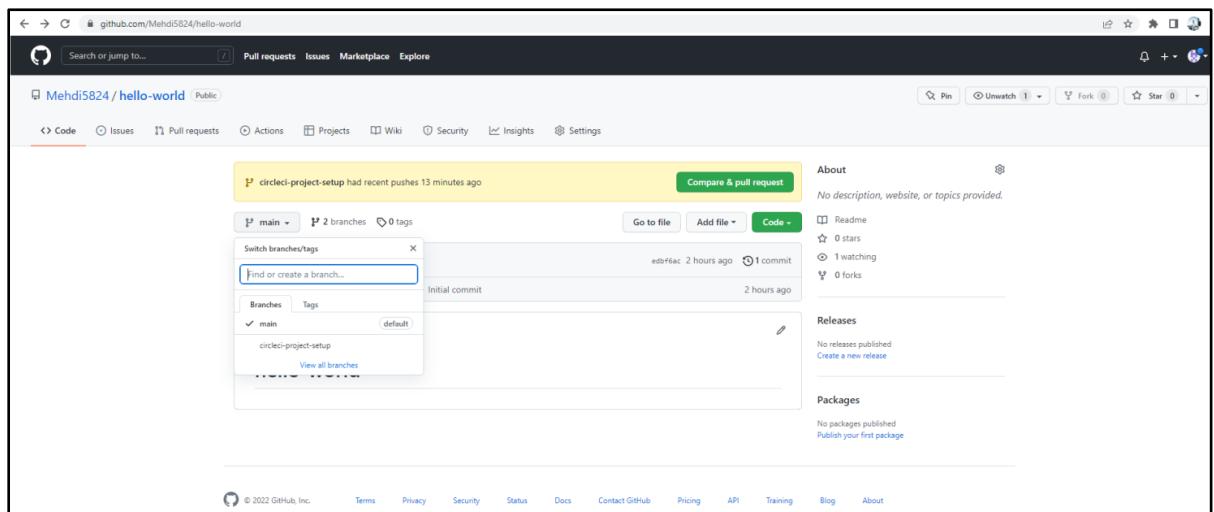
- Select “say-hello” Job with a green tick



This screenshot shows the CircleCI dashboard for a user named 'Mehdi5824'. The main view displays a successful workflow named 'say-hello-workflow' under the project 'hello-world'. The workflow was completed 5 minutes ago and has a duration of 3 seconds. The commit hash is '6b308f0' and the author is 'Mehdi5824'. The sidebar on the left includes links for Dashboard, Projects, Insights, Organization Settings, Plan (with an 'UPGRADE' button), Notifications, Status (DEGRADED), Docs, Orbs, and Support. A banner at the bottom right provides performance metrics: 20 pipelines/day, 70 min to recovery, 5 pipelines/day, and 142 min.

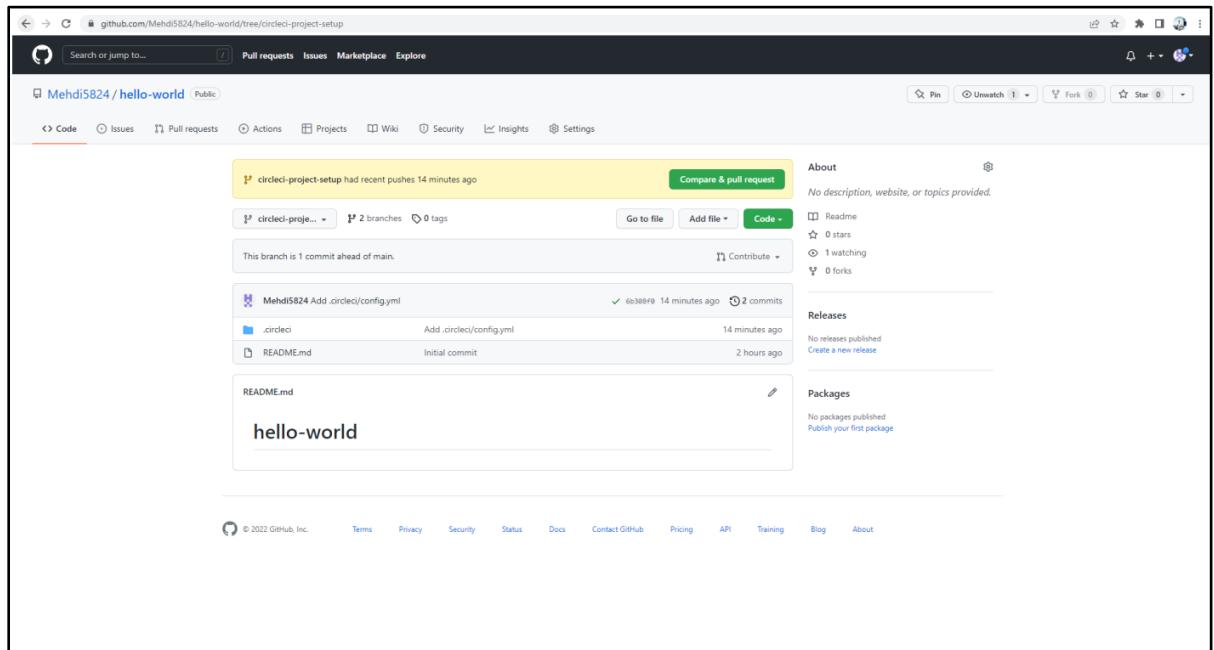
This screenshot shows the detailed view of the 'say-hello' job within the 'say-hello-workflow'. The job was queued 9 minutes ago and is currently running on a 'Docker / Large' executor. The status is 'Success'. The pipeline consists of four steps: 'Spin up environment', 'Preparing environment variables', 'Checkout code', and 'Say hello'. The first step is currently executing, taking 00:02. A tooltip suggests using parallelism to run faster tests. The sidebar and bottom banner are identical to the previous screenshot.

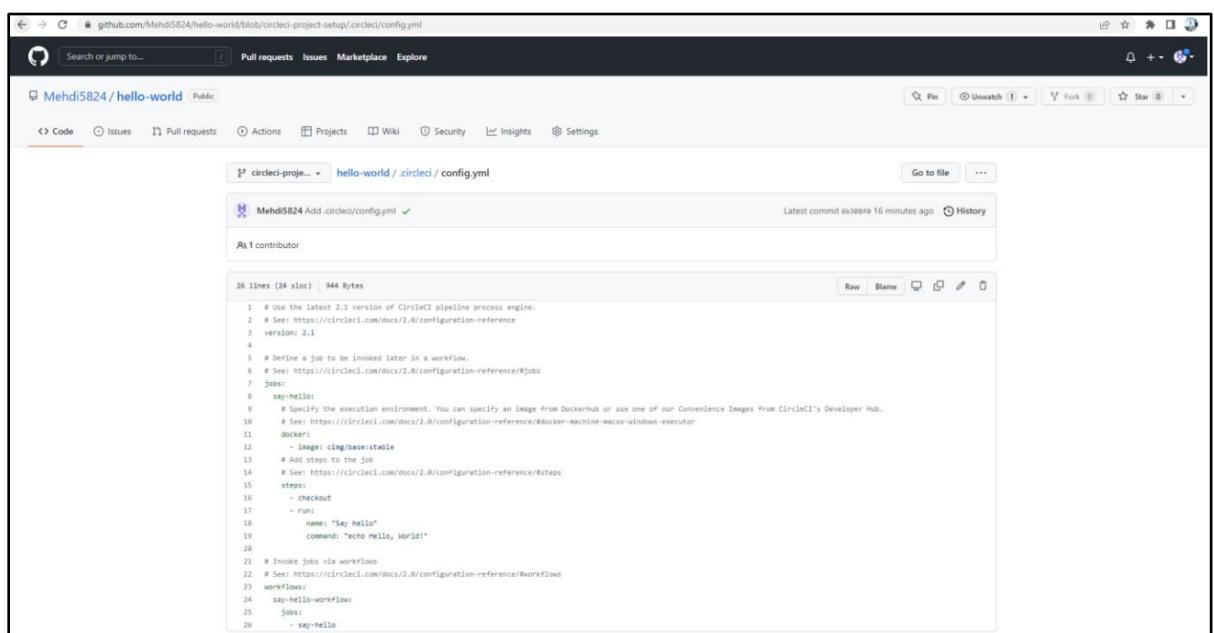
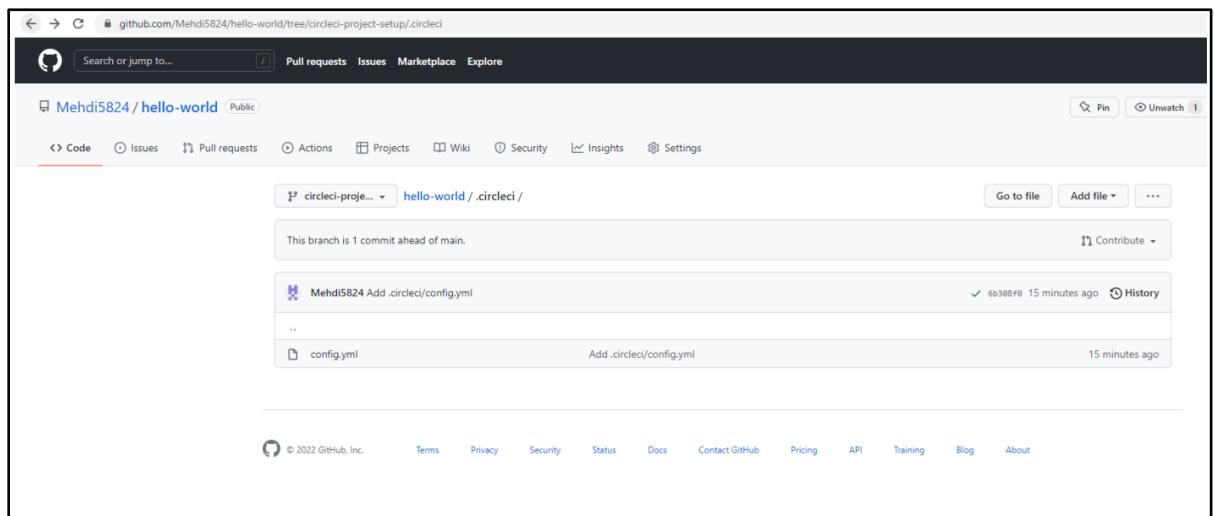
- Select Branch and option circleci-project-setup



Step 4: 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.

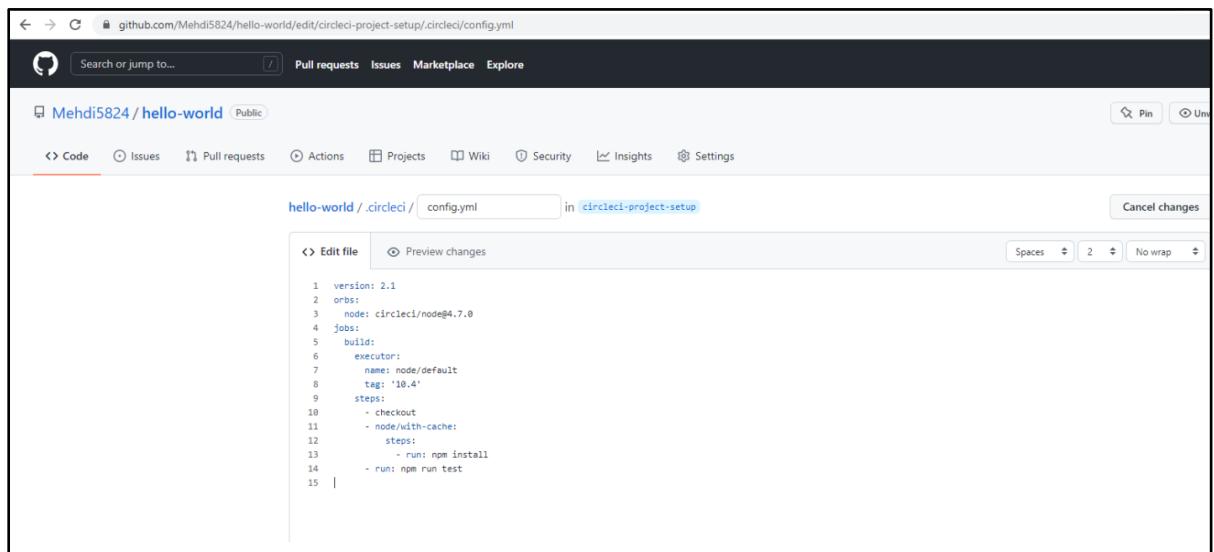




Let's use the **Node orb**. Replace the existing config by pasting the following code:

```
1  version: 2.1
2  orbs:
3    node: circleci/node@4.7.0
4  jobs:
5    build:
6      executor:
7        name: node/default
8        tag: '10.4'
9      steps:
10     - checkout
11     - node/with-cache:
12       steps:
13         - run: npm install
14     - run: npm run test
```

The GitHub file editor should look like this



A screenshot of a GitHub repository page showing the configuration file for CircleCI. The URL in the address bar is `github.com/Mehdi5824/hello-world/edit/circleci-project-setup/.circleci/config.yml`. The page title is "Mehdi5824 / hello-world [Public]". The navigation bar includes links for Code, Issues, Pull requests, Actions, Projects, Wiki, Security, Insights, and Settings. The main content area shows the file `hello-world/.circleci/config.yml` with the following content:

```
1  version: 2.1
2  orbs:
3    node: circleci/node@4.7.0
4  jobs:
5    build:
6      executor:
7        name: node/default
8        tag: '10.4'
9      steps:
10     - checkout
11     - node/with-cache:
12       steps:
13         - run: npm install
14     - run: npm run test
15  |
```

Scroll down and Commit your changes on GitHub



The screenshot shows a GitHub commit dialog titled "Commit changes". It displays the following message: "Update config.yml" and "we have updated to [config.yml](#) file". Below the message, there are two radio button options: one selected for "Commit directly to the `circleci-project-setup` branch" and another for "Create a new branch for this commit and start a pull request". At the bottom are "Commit changes" and "Cancel" buttons. The footer of the GitHub page includes links for Terms, Privacy, Security, Status, Docs, Contact GitHub, Pricing, API, Training, Blog, and About.

- After committing your changes, then return to the Projects page in CircleCI. You should see a new pipeline running... and it will fail! 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.

Pipeline	Status	Workflow	Branch / Commit	Start	Duration	Actions
hello-world 2	Failed	Error calling workflow: 'workflow' Error calling job: 'build' Cannot find a definition for command named node/with-cache	circleci-project-setup ca5ab74 Update config.yml	3m ago	6s	Details Logs Cancel Re-run ...
hello-world 1	Success	say-hello-workflow	circleci-project-setup 6b308f0	22m ago	41s	Details Logs Cancel Re-run ...

Step 5: Use Workflows

You do not have to use orbs to use CircleCI. The following example details how to create a custom configuration that also uses the workflow feature of CircleCI.

- Take a moment and read the comments in the code block below. Then, to see workflows in action, edit your `.circleci/config.yml` file and copy and paste the following text into it.

```
1  version: 2
2  jobs: # we now have TWO jobs, so that a workflow can coordinate them!
3    one: # This is our first job.
4      docker: # it uses the docker executor
5        - image: cimg/ruby:2.6.8 # specifically, a docker image with ruby 2.6.8
6        auth:
7          username: mydockerhub-user
8          password: $DOCKERHUB_PASSWORD # context / project UI env-var reference
9      # Steps are a list of commands to run inside the docker container above.
10     steps:
11       - checkout # this pulls code down from GitHub
12       - run: echo "A first hello" # This prints "A first hello" to stdout.
13       - run: sleep 25 # a command telling the job to "sleep" for 25 seconds.
14   two: # This is our second job.
15     docker: # it runs inside a docker image, the same as above.
16       - image: cimg/ruby:3.0.2
17     auth:
18       username: mydockerhub-user
19       password: $DOCKERHUB_PASSWORD # context / project UI env-var reference
20     steps:
21       - checkout
22       - run: echo "A more familiar hi" # We run a similar echo command to above.
23       - run: sleep 15 # and then sleep for 15 seconds.
24   # Under the workflows: map, we can coordinate our two jobs, defined above.
25 workflows:
26   version: 2
27   one_and_two: # this is the name of our workflow
28     jobs: # and here we list the jobs we are going to run.
29       - one
30       - two
```

You don't need to write the comments which are the text after #

- Commit these changes to your repository and navigate back to the CircleCI Pipelines page. You should see your pipeline running.



```

1 version: 2
2 jobs:
3   one: # This is our first job.
4     docker: # it uses the docker executor
5       image: cimg/ruby:2.6.8 # specifically, a docker image with ruby 2.6.8
6       auth:
7         username: mydockerhub-user
8         password: $DOCKERHUB_PASSWORD # context / project UI env-var reference
9     steps:
10      - checkout # this pulls code down from GitHub
11      - run: echo "A first hello" # This prints "A first hello" to stdout.
12      - run: sleep 25 # a command telling the job to "sleep" for 25 seconds.
13   two: # This is our second job.
14     docker: # it runs inside a docker image, the same as above.
15       image: cimg/ruby:3.0.2
16       auth:
17         username: mydockerhub-user
18         password: $DOCKERHUB_PASSWORD # context / project UI env-var reference
19     steps:
20      - checkout
21      - run: echo "A more familiar hi" # We run a similar echo command to above.
22      - run: sleep 15 # and then sleep for 15 seconds.
23
24 workflows:
25   version: 2
26   one_and_two: # this is the name of our workflow
27     jobs:
28       - one
29       - two
30
31

```

- Click on the running pipeline to view the workflow you have created. You should see that two jobs ran (or are currently running!) concurrently.

Pipeline	Status	Workflow	Branch / Commit	Start	Duration	Actions
hello-world 3	Running	one_and_two	circleci-project-setup e11ad95 Update config.yml	14s ago	13s	↻ C K ...
hello-world 2	Failed	Error calling workflow: 'workflow' Error calling job: 'build' Cannot find a definition for command named node/with-cache	circleci-project-setup ca5ab74 Update config.yml	9m ago	6s	↻ C K ...
hello-world 1	Success	say-hello-workflow	circleci-project-setup 6b308f0	29m ago	5s	↻ C K ...

The screenshot shows the CircleCI web interface. On the left, there's a sidebar with a user profile for 'Mehdi5824' and 'Mehdi Rezaie'. The sidebar includes links for Dashboard, Projects, Insights, Organization Settings, Plan (with an 'UPGRADE' button), and a note about finding an organization. The main area shows a pipeline named 'one_and_two' under the 'hello-world' project. The run is marked as 'Success'. It shows details like duration (34s / 3s ago), branch (circleci-project-setup), commit (e11ad95), and author (Update config.yml). Below this, two workspace steps are listed: 'two' (19s) and 'one' (31s), both marked with green checkmarks.

Step 6: Add some changes to use workspaces

- Each workflow has an associated workspace which can be used to transfer files to downstream jobs as the workflow progresses. You can use workspaces to pass along data that is unique to this run and which is needed for downstream jobs. Try updating config.yml to the following:

```

1  version: 2
2  jobs:
3    one:
4      docker:
5        - image: cimg/ruby:3.0.2
6          auth:
7            username: mydockerhub-user
8            password: $DOCKERHUB_PASSWORD # context / project UI env-var reference
9        steps:
10       - checkout
11       - run: echo "A first hello"
12       - run: mkdir -p my_workspace
13       - run: echo "Trying out workspaces" > my_workspace/echo-output
14       - persist_to_workspace:
15         # Must be an absolute path, or relative path from working_directory
16         root: my_workspace
17         # Must be relative path from root
18         paths:
19           - echo-output
20   two:
21     docker:
22       - image: cimg/ruby:3.0.2
23       auth:
24         username: mydockerhub-user
25         password: $DOCKERHUB_PASSWORD # context / project UI env-var reference
26     steps:
27       - checkout
28       - run: echo "A more familiar hi"
29       - attach_workspace:
30         # Must be absolute path or relative path from working_directory
31         at: my_workspace

```

```

32       - run: |
33         if [[ $(cat my_workspace/echo-output) == "Trying out workspaces" ]]; then
34           echo "It worked!";
35         else
36           echo "Nope!";
37           exit 1
38         fi
39   workflows:
40     version: 2
41   one_and_two:
42     jobs:
43       - one
44       - two:
45         requires:
46           - one

```

- Updated config.yml in GitHub file editor should be updated like this



The screenshot shows the CircleCI web interface. At the top, there's a header with 'hello-world / .circleci / config.yml' and a 'Cancel changes' button. Below the header is a code editor window containing the following YAML configuration:

```

version: 2
jobs:
  one:
    docker:
      - image: cimg/ruby:3.0.2
      auth:
        username: mydockerhub-user
        password: $DOCKERHUB_PASSWORD # context / project UI env-var reference
    steps:
      - checkout
      - run: echo "A first hello"
      - run: mkdir -p my_workspace
      - run: echo "Trying out workspaces" > my_workspace/echo-output
      - persist_to_workspace:
          # Must be an absolute path, or relative path from working_directory
          root: my_workspace
          # Must be relative path from root
          paths:
            - echo-output
  two:
    docker:
      - image: cimg/ruby:3.0.2
      auth:
        username: mydockerhub-user
        password: $DOCKERHUB_PASSWORD # context / project UI env-var reference
    steps:
      - checkout
      - run: echo "A more familiar hi"
      - attach_workspace:
          # Must be absolute path or relative path from working_directory
          at: my_workspace
      - run: |
          if [[ $(cat my_workspace/echo-output) == "Trying out workspaces" ]]; then
            echo "It worked!"
          fi

```

Below the code editor is a 'Commit changes' dialog. It has a 'Commit changes' button and a 'Cancel' button. The dialog contains a text input field with '3rd Update' and two radio button options:

- 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.](#)

- Finally your workflow with the jobs running should look like this

The screenshot shows the CircleCI dashboard for the 'one_and_two' pipeline. The pipeline is currently running, as indicated by the 'Running' status next to the pipeline name. The pipeline consists of two sequential steps: 'one' and 'two'. The 'one' step took 4s and the 'two' step took 3s. The pipeline summary table shows:

Duration	Branch	Commit	Author & Message
15s	circleci-project-setup	bfce97c	Update config.yml

Practical No 7

Aim: Working with TeamService

Source Code:

Step 1:

- Open command prompt and create a web api

```
C:\Windows\System32\cmd.exe

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

Processing post-creation actions...
Running 'dotnet restore' on TeamService\TeamService.csproj...
  Restore completed in 5.9 sec for D:\TeamService\TeamService.csproj.

Restore succeeded.
```

- Remove existing weatherforecast files both model and controller files.

Step 2:

- Add new files as follows:
- Add Member.cs to “D:\TeamService\Models” folder

```
using System;
namespace TeamService.Models
{
    public class Member
    {
        public Guid ID { get; set; }

        public string FirstName { get; set; }
        public string LastName { get; set; }
        public Member() {}
        public Member(Guid id) : this()
        {
            this.ID = id;
        }
        public Member(string firstName, string lastName, Guid id) : this(id)
        {
            this.FirstName = firstName;
            this.LastName = lastName;
        }
        public override string ToString()
        {
            return this.LastName;
        }
    }
}
```



}

- Add Team.cs to “D:\TeamService\Models” folder

```
using System;
using System.Collections.Generic;
namespace TeamService.Models
{
    public class Team
    {
        public string Name { get; set; }
        public Guid ID { get; set; }
        public ICollection<Member> Members { get; set; }public Team()
        {
            this.Members = new List<Member>();
        }
        public Team(string name) : this()
        {
            this.Name = name;
        }
        public Team(string name, Guid id) : this(name)
        {
            this.ID = id;
        }
        public override string ToString()
        {
            return this.Name;
        }
    }
}
```

- add TeamsController.cs file to “D:\TeamService\Controllers” folder

```
using System;
using Microsoft.AspNetCore.Hosting;
using Microsoft.AspNetCore.Builder;
using Microsoft.AspNetCore.Mvc;
```



```
using System.Collections.Generic;
using System.Linq;
using TeamService.Models;
using System.Threading.Tasks;
using TeamService.Persistence;
namespace TeamService
{
    [Route("[controller]")]
    public class TeamsController : Controller
    {
        ITeamRepository repository;
        public TeamsController(ITeamRepository repo)
        {
            repository = repo;
        }
        [HttpGet]
        public virtual IActionResult GetAllTeams()
        {
            return this.Ok(repository.List());
        }
        [HttpGet("{id}")]
        public IActionResult GetTeam(Guid id)
        {
            Team team = repository.Get(id);
            if (team != null)
            {
                return this.Ok(team);
            }else
            {
                return this.NotFound();
            }
        }
        [HttpPost]
        public virtual IActionResult CreateTeam([FromBody]Team newTeam)
        {
```



```
repository.Add(newTeam);

return this.Created($"/teams/{newTeam.ID}", newTeam);

}

[HttpPut("{id}")]
public virtual IActionResult UpdateTeam([FromBody]Team team, Guid id)
{

    team.ID = id;

    if(repository.Update(team) == null)

    {

        return this.NotFound();

    }

    else

    {

        return this.Ok(team);

    }

}

[HttpDelete("{id}")]
public virtual IActionResult DeleteTeam(Guid id)
{

    Team team = repository.Delete(id);

    if (team == null)

    {

        return this.NotFound();

    }

    else

    {

        return this.Ok(team.ID);

    }

}

}
```

- Add MembersController.cs file to “D:\TeamService\Controllers” folder
 - using System;
 - using Microsoft.AspNetCore.Hosting;



```
using Microsoft.AspNetCore.Builder;
using Microsoft.AspNetCore.Mvc;
using System.Collections.Generic;
using System.Linq;
using TeamService.Models;
using System.Threading.Tasks;
using TeamService.Persistence;namespace TeamService
{
    [Route("/teams/{teamId}/{controller}")]
    public class MembersController : Controller
    {
        ITeamRepository repository;
        public MembersController(ITeamRepository repo)
        {
            repository = repo;
        }
        [HttpGet]
        public virtual IActionResult GetMembers(Guid teamID)
        {
            Team team = repository.Get(teamID);
            if(team == null)
            {
                return this.NotFound();
            }
            else
            {
                return this.Ok(team.Members);
            }
        }
        [HttpGet]
        [Route("/teams/{teamId}/{controller}/{memberId}")]
        public virtual IActionResult GetMember(Guid teamID, Guid memberId)
        {
            Team team = repository.Get(teamID);
            if(team == null)
```



```
{  
    return this.NotFound();  
}  
else  
{  
    var q = team.Members.Where(m => m.ID == memberId);  
    if(q.Count() < 1)  
    {  
        return this.NotFound();  
    }  
    else  
{  
        return this.Ok(q.First());  
    }  
}  
}  
[HttpPost]  
[Route("/{teamId}/{controller}/{memberId}")]  
public virtual IActionResult UpdateMember([FromBody]Member updatedMember,  
Guid teamID, Guid memberId)  
{  
    Team team = repository.Get(teamID);  
    if(team == null)  
    {  
        return this.NotFound();  
    }  
    else  
{  
        var q = team.Members.Where(m => m.ID == memberId);  
        if(q.Count() < 1)  
        {  
            return this.NotFound();  
        }  
        else  
{  
    }
```

```
        team.Members.Remove(q.First());
        team.Members.Add(updatedMember);
        return this.Ok();
    }
}

}

[HttpPost]
public virtual IActionResult CreateMember([FromBody]Member newMember, Guid
teamID)
{
    Team team = repository.Get(teamID);
    if(team == null)
    {
        return this.NotFound();
    }
    else
    {
        team.Members.Add(newMember);
        var teamMember = new {TeamID = team.ID, MemberID =
newMember.ID};
        return
this.Created($"/teams/{teamMember.TeamID}/{[controller]}/{tea
mMember.MemberID}", teamMember);
    }
}

[HttpGet]
[Route("/members/{memberId}/team")]
public IActionResult GetTeamForMember(Guid memberId)
{
    var teamId = GetTeamIdForMember(memberId);
    if (teamId != Guid.Empty)
    {
        return this.Ok(new {TeamID = teamId });
    }
    else
    {
```

```
        return this.NotFound();
    }

}

private Guid GetTeamIdForMember(Guid memberId)
{
    foreach (var team in repository.List())
    {
        var member = team.Members.FirstOrDefault( m => m.ID == memberId);

        if (member != null)
        {
            return team.ID;
        }
    }

    return Guid.Empty;
}

}
```

Step 3:

- Create folder “D:\TeamService\Persistence”
- Add file ITeamReposiroty.cs in “D:\TeamService\Persistence” folder

```
using System;
using System.Collections.Generic;
using TeamService.Models;
namespace TeamService.Persistence
```

```
{  
    public interface ITeamRepository  
    {  
        IEnumerable<Team> List();  
        Team Get(Guid id);  
        Team Add(Team team);  
        Team Update(Team team);  
        Team Delete(Guid id);  
    }  
}
```



- Add MemoryTeamRepository.cs in “D:\TeamService\Persistence” folder

```
using System;
using System.Collections.Generic;
using System.Linq;
using TeamService;
using TeamService.Models;
namespace TeamService.Persistence
{
    public class MemoryTeamRepository : ITeamRepository
    {
        protected static ICollection<Team> teams;
        public MemoryTeamRepository()
        {
            if(teams == null)
            {
                teams = new List<Team>();
            }
        }
        public MemoryTeamRepository(ICollection<Team> teams)
        {
            MemoryTeamRepository.teams = teams;
        }
        public IEnumerable<Team> List()
        {
            return teams;
        }
        public Team Get(Guid id)
        {
            return teams.FirstOrDefault(t => t.ID == id);
        }
        public Team Update(Team t)
        {
            Team team = this.Delete(t.ID);
            if(team != null)
```



```
{  
    team = this.Add(t);  
}  
return team;  
}  
public Team Add(Team team)  
{  
    teams.Add(team);  
    return team;  
}  
public Team Delete(Guid id)  
{  
    var q = teams.Where(t => t.ID == id);  
    Team team = null;  
    if (q.Count() > 0)  
    {  
        team = q.First();  
        teams.Remove(team);  
    }  
    return team;  
}  
}
```

Step 4:

- Add following line to Startup.cs in public void ConfigureServices(IServiceCollection services) method
services.AddScoped<ITeamRepository, MemoryTeamRepository>();

Output:

- Open two command prompt
- Command Prompt 1: go inside folder teamservice first



```
Command Prompt - dotnet run

D:\TeamService>dotnet run
info: Microsoft.Hosting.Lifetime[0]
      Now listening on: https://localhost:5001
info: Microsoft.Hosting.Lifetime[0]
      Now listening on: http://localhost:5000
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:\TeamService
```

- On Command Prompt 2:

To get all teams

```
curl --insecure https://localhost:5001/teams
```

```
D:\>curl --insecure https://localhost:5001/teams
[]
```

To create new team

```
curl --insecure -H "Content-Type:application/json" -X POST -d "{\"id\":\"e52baa63-d511-417e-9e54-7aab04286281\", \"name\":\"KC\"}" https://localhost:5001/teams
```

```
D:\>curl --insecure -H "Content-Type:application/json" -X POST -d "{\"id\":\"e52baa63-d511-417e-9e54-7aab04286281\", \"name\":\"KC\"}" https://localhost:5001/teams
{"name": "KC", "id": "e52baa63-d511-417e-9e54-7aab04286281", "members": []}
D:\>
```

To create one more new team

```
curl --insecure -H "Content-Type:application/json" -X POST -d "{\"id\":\"e12baa63-d511-417e-9e54-7aab04286281\", \"name\":\"MSC Part1\"}" https://localhost:5001/teams
```

```
D:\>curl --insecure -H "Content-Type:application/json" -X POST -d "{\"id\":\"e12baa63-d511-417e-9e54-7aab04286281\", \"name\":\"MSC Part1\"}" https://localhost:5001/teams
{"name": "MSC Part1", "id": "e12baa63-d511-417e-9e54-7aab04286281", "members": []}
D:\>
```

To get all teams

```
curl --insecure https://localhost:5001/teams
```



```
D:\>curl --insecure https://localhost:5001/teams
[{"name": "KC", "id": "e52baa63-d511-417e-9e54-7aab04286281", "members": []}, {"name": "MSC Part1", "id": "e12baa63-d511-417e-9e54-7aab04286281", "members": []}]
D:\>
```

To get single team with team-id as parameter

```
curl --insecure https://localhost:5001/teams/e52baa63-d511-417e-9e54-7aab04286281
```

```
D:\> curl --insecure https://localhost:5001/teams/e52baa63-d511-417e-9e54-7aab04286281
{"name": "KC", "id": "e52baa63-d511-417e-9e54-7aab04286281", "members": []}
D:\>
```

To update team details (change name of first team from "KC" to "KC IT DEPT")

```
curl --insecure -H "Content-Type:application/json" -X PUT -d "{\"id\": \"e52baa63-d511-417e-9e54-7aab04286281\", \"name\": \"KC IT DEPT\"}" https://localhost:5001/teams/e52baa63-d511-417e-9e54-7aab04286281
```

```
D:\>curl --insecure -H "Content-Type:application/json" -X PUT -d "{\"id\": \"e52baa63-d511-417e-9e54-7aab04286281\", \"name\": \"KC IT DEPT\"}" https://localhost:5001/teams/e52baa63-d511-417e-9e54-7aab04286281
{"name": "KC IT DEPT", "id": "e52baa63-d511-417e-9e54-7aab04286281", "members": []}
D:\>
```

To delete team

```
curl --insecure -H "Content-Type:application/json" -X DELETE
https://localhost:5001/teams/e52baa63-d511-417e-9e54-7aab04286281
```

```
Command Prompt - □ ×
D:\>curl --insecure -H "Content-Type:application/json" -X DELETE https://localhost:5001/teams/e52baa63-d511-417e-9e54-7aab04286281
D:\>
```

Confirm: with get all teams now it shows only one team (first one is deleted)

```
curl --insecure https://localhost:5001/teams
```

```
Command Prompt
D:\>curl --insecure https://localhost:5001/teams
[{"name": "MSC Part1", "id": "e12baa63-d511-417e-9e54-7aab04286281", "members": []}]
D:\>
```