Institute of Distance & Open Learning
M.Sc.I.T

UNIVERSITY OF MUMBAI

Certificate

This is to certify that **<u>Rajpurohit Harsh Hargopalsingh</u>** Seat no **<u>600257</u>** has successfully completed all the practical of paper titled "**<u>MicroService Architecture</u>**" for M.Sc. (Information Technology) Part 1 Sem 2 in the year 2022-2023

Signature
Faculty In-Charge

Head of the Department

Examiner

INDEX

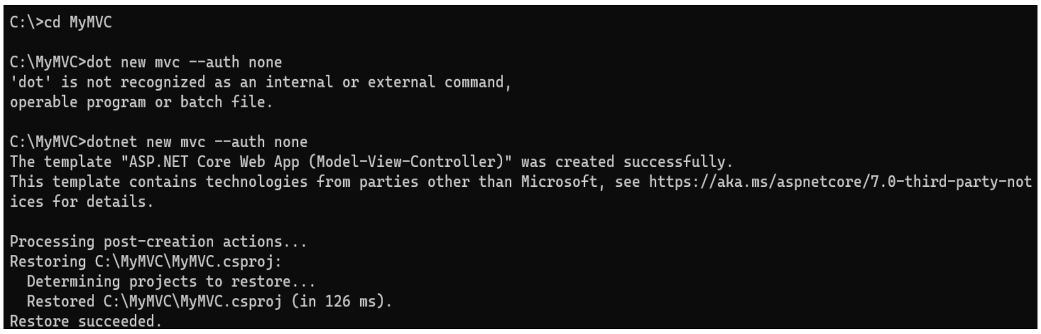
Practical No 1

Building APT.NET Core MVC Application.

1. Install .Net Core Sdk (Link: https://dotnet.microsoft.com/learn/dotnet/hello-world-tutorial/install)
2. create folder MyMVC folder in D: drive or any other drive
3. open command prompt and perform following operations

Command: to create mvc project

dotnet new mvc --auth none

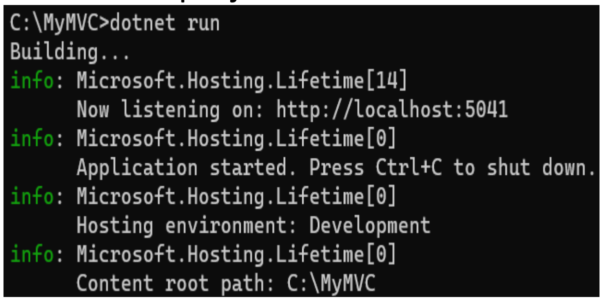


```
C:\>cd MyMVC

C:\MyMVC>dot new mvc --auth none
'dot' is not recognized as an internal or external command,
operable program or batch file.

C:\MyMVC>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/7.0-third-party-not
ices for details.

Processing post-creation actions...
Restoring C:\MyMVC\MyMVC.csproj:
  Determining projects to restore...
  Restored C:\MyMVC\MyMVC.csproj (in 126 ms).
Restore succeeded.
```

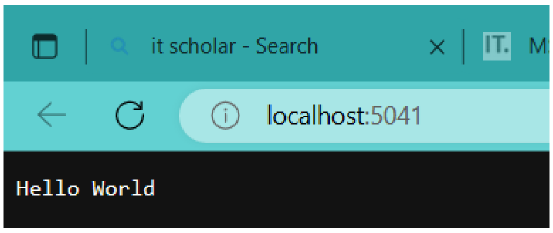
4. Go to controllers folder and modify HomeController.cs file to match following:

```
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 MyMVC.Models;
namespace MyMVC.Controllers
{ public class HomeController : Controller
{
public String Index()
{ return "Hello World"; }
}
}
```

5. Run the project

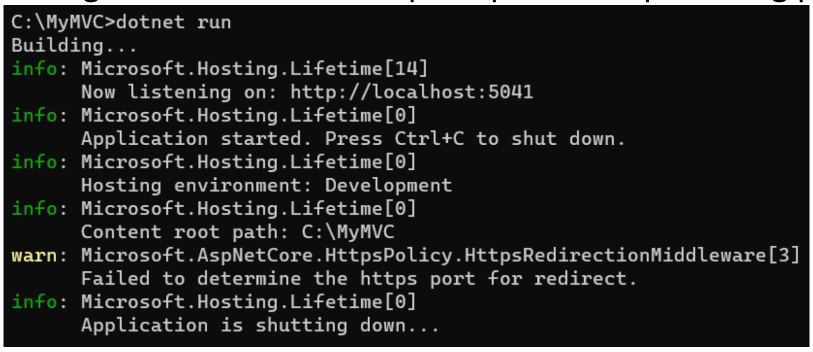


```
C:\MyMVC>dotnet run
Building...
info: Microsoft.Hosting.Lifetime[14]
      Now listening on: http://localhost:5041
info: Microsoft.Hosting.Lifetime[0]
      Application started. Press Ctrl+C to shut down.
info: Microsoft.Hosting.Lifetime[0]
      Hosting environment: Development
info: Microsoft.Hosting.Lifetime[0]
      Content root path: C:\MyMVC
```



```
Hello World
```

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



```
C:\MyMVC>dotnet run
Building...
info: Microsoft.Hosting.Lifetime[14]
      Now listening on: http://localhost:5041
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: C:\MyMVC
warn: Microsoft.AspNetCore.HttpsPolicy.HttpsRedirectionMiddleware[3]
      Failed to determine the https port for redirect.
info: Microsoft.Hosting.Lifetime[0]
      Application is shutting down...
```

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

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

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

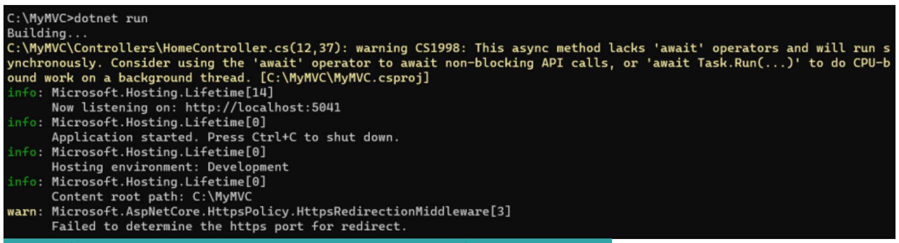
```
@{
ViewData["Title"] = "Home Page";
}
<div>
Symbol: @Model.Symbol <br/>
Price: $@Model.Price <br/>
</div>
```

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

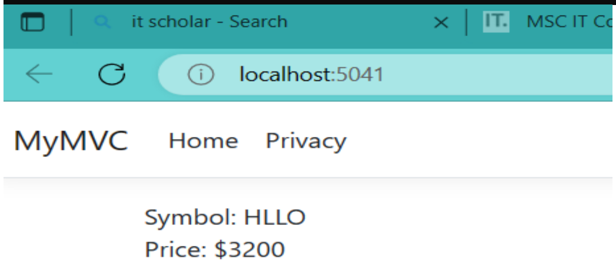
```
using System;
using System.Collections.Generic;
using System.Diagnostics;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using Microsoft.Extensions.Logging;
using MyMVC.Models;
namespace MyMVC.Controllers
{
public class HomeController : Controller
{ public async Task <IActionResult> Index()
{
var model= new StockQuote{ Symbol='HLLO', Price=3200};
return View(model);
}
}
}
```

10.Now run the project using

dotnet run



```
C:\MyMVC>dotnet run
Building...
C:\MyMVC\Controllers\HomeController.cs(12,37): warning CS1998: This async method lacks 'await' operators and will run s
ynchronously. Consider using the 'await' operator to await non-blocking API calls, or 'await Task.Run(...)' to do CPU-b
ound work on a background thread. [C:\MyMVC\MyMVC.csproj]
info: Microsoft.Hosting.Lifetime[14]
      Now listening on: http://localhost:5041
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: C:\MyMVC
warn: Microsoft.AspNetCore.HttpsPolicy.HttpsRedirectionMiddleware[3]
      Failed to determine the https port for redirect.
```



Symbol: HLLO
Price: $3200

Practical 2:
Building ASP.Net core REST API

Software requirement:

1. **Download and install:**

   To start building .NET apps you just need to download and install the .NET SDK (Software Development Kit version 3.0 above).Link: https://dotnet.microsoft.com/learn/dotnet/hello-world-tutorial/install

2. **Check everything installed correctly.**

   Once you've installed, open a new command prompt and run the following command:Command prompt > dotnet

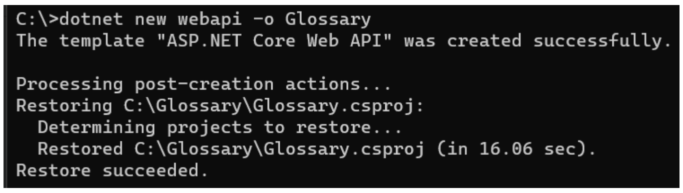
Create your web API

1. **Open two command prompts**

Command prompt 1:Command:

    dotnet new webapi -o Glossary

output:

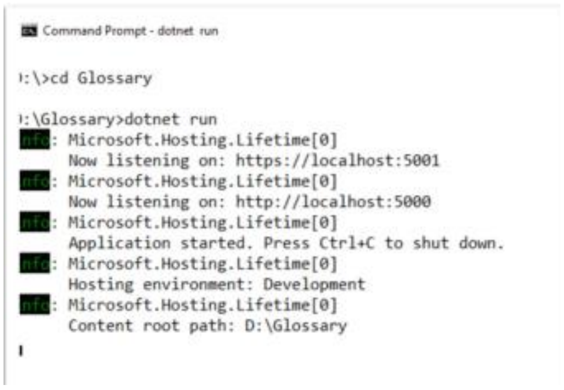


```
C:\>dotnet new webapi -o Glossary
The template "ASP.NET Core Web API" was created successfully.

Processing post-creation actions...
Restoring C:\Glossary\Glossary.csproj:
  Determining projects to restore...
  Restored C:\Glossary\Glossary.csproj (in 16.06 sec).
Restore succeeded.
```

Command:

    cd Glossary
    dotnet run

Output:

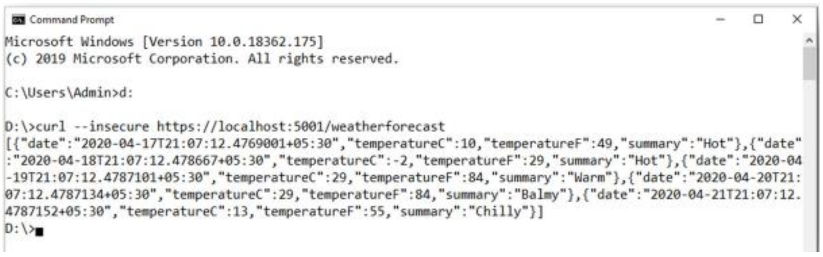


```
Command Prompt - dotnet run

:\>cd Glossary

:\Glossary>dotnet run
nfo: Microsoft.Hosting.Lifetime[0]
     Now listening on: https://localhost:5001
nfo: Microsoft.Hosting.Lifetime[0]
     Now listening on: http://localhost:5000
nfo: Microsoft.Hosting.Lifetime[0]
     Application started. Press Ctrl+C to shut down.
nfo: Microsoft.Hosting.Lifetime[0]
     Hosting environment: Development
nfo: Microsoft.Hosting.Lifetime[0]
     Content root path: D:\Glossary
```

2. **Command Prompt 2: (try running ready made weatherforecast class for testing) Command:**

    curl --insecure https://localhost:5001/weatherforecast

output:



```
Microsoft Windows [Version 10.0.18362.175]
(c) 2019 Microsoft Corporation. All rights reserved.

C:\Users\Admin>d:

D:\>curl --insecure https://localhost:5001/weatherforecast
[{"date":"2020-04-17T21:07:12.4769001+05:30","temperatureC":10,"temperatureF":49,"summary":"Hot"},{"date"
:"2020-04-18T21:07:12.478667+05:30","temperatureC":-2,"temperatureF":29,"summary":"Hot"},{"date":"2020-04
-19T21:07:12.4787101+05:30","temperatureC":29,"temperatureF":84,"summary":"Warm"},{"date":"2020-04-20T21:
07:12.4787134+05:30","temperatureC":29,"temperatureF":84,"summary":"Balmy"},{"date":"2020-04-21T21:07:12.
4787152+05:30","temperatureC":13,"temperatureF":55,"summary":"Chilly"}]
D:\>
```

3. Now Change the content:

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

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

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

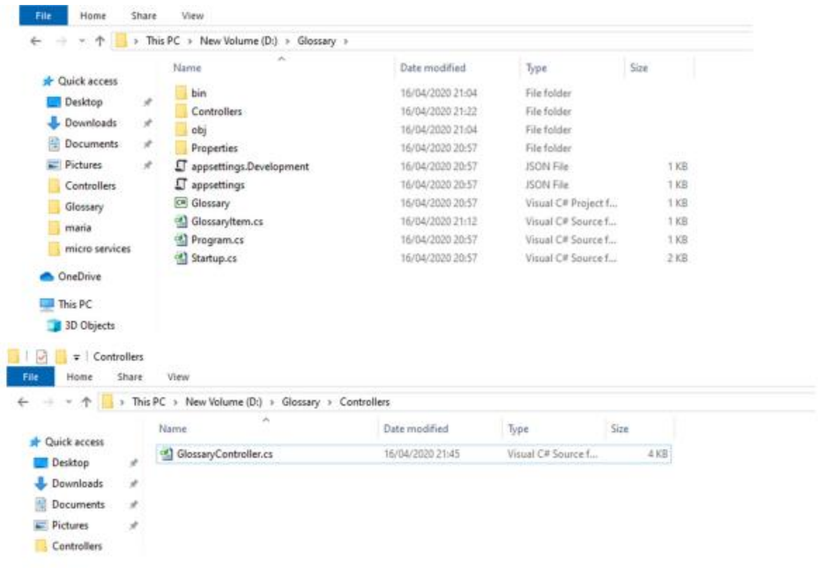
2) D:\Glossary\Controllers\ GlossaryController.cs (type it in notepad and save as all files)

```
//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();
```

```
        }
        }
        }
    }
```

Output:



This PC › New Volume (D:) › Glossary ›

Name	Date modified	Type	Size
bin	16/04/2020 21:04	File folder	
Controllers	16/04/2020 21:22	File folder	
obj	16/04/2020 21:04	File folder	
Properties	16/04/2020 20:57	File folder	
appsettings.Development	16/04/2020 20:57	JSON File	1 KB
appsettings	16/04/2020 20:57	JSON File	1 KB
Glossary	16/04/2020 20:57	Visual C# Project f...	1 KB
GlossaryItem.cs	16/04/2020 21:12	Visual C# Source f...	1 KB
Program.cs	16/04/2020 20:57	Visual C# Source f...	1 KB
Startup.cs	16/04/2020 20:57	Visual C# Source f...	2 KB

This PC › New Volume (D:) › Glossary › Controllers

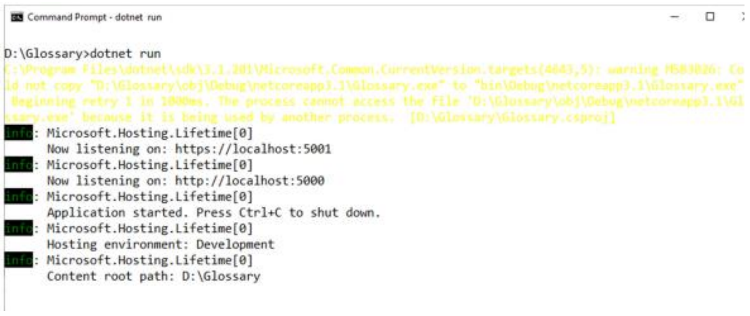
Name	Date modified	Type	Size
GlossaryController.cs	16/04/2020 21:45	Visual C# Source f...	4 KB

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

Command:

    dotnet run

output:



```
D:\Glossary>dotnet run
C:\Program Files\dotnet\sdk\3.1.201\Microsoft.Common.CurrentVersion.targets(4643,5): warning MSB3026: Co
ld not copy "D:\Glossary\obj\Debug\netcoreapp3.1\Glossary.exe" to "bin\Debug\netcoreapp3.1\Glossary.exe"
 Beginning retry 1 in 1000ms. The process cannot access the file 'D:\Glossary\obj\Debug\netcoreapp3.1\Gl
ssary.exe' because it is being used by another process.  [D:\Glossary\Glossary.csproj]
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:\Glossary
```

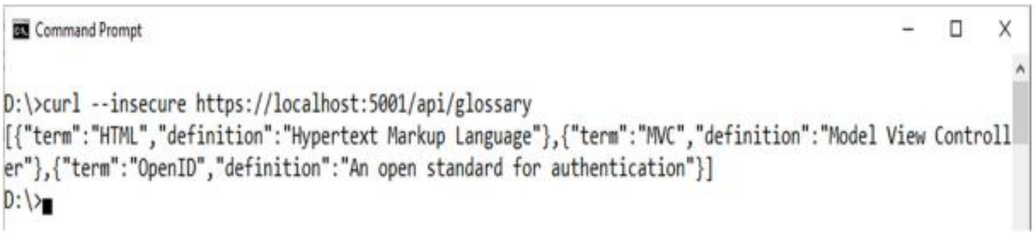
On Command prompt2:

1) Getting a list of items:

Command:

    curl --insecure https://localhost:5001/api/glossary

Output:



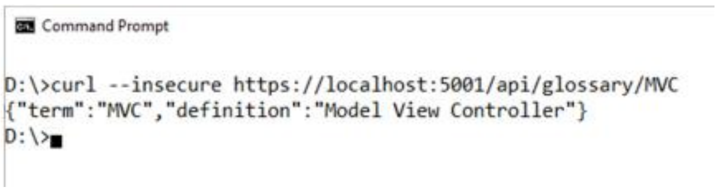
```
D:\>curl --insecure https://localhost:5001/api/glossary
[{"term":"HTML","definition":"Hypertext Markup Language"},{"term":"MVC","definition":"Model View Controll
er"},{"term":"OpenID","definition":"An open standard for authentication"}]
D:\>
```

2) Getting a single item

Command:

curl --insecure https://localhost:5001/api/glossary/MVC

Output:

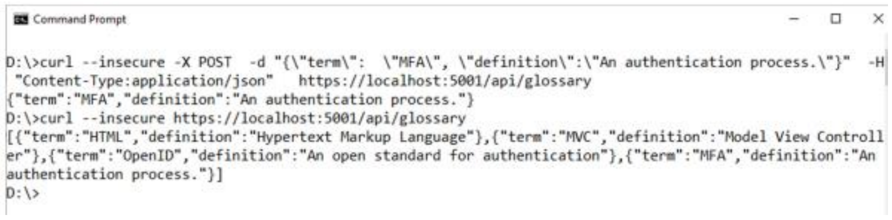


```
Command Prompt

D:\>curl --insecure https://localhost:5001/api/glossary/MVC
{"term":"MVC","definition":"Model View Controller"}
D:\>
```

3) Creating an item

Command: curl --insecure -X POST -d "{\"term\": \"MFA\", \"definition\":\"An authentication process.\"}" -H "Content-Type:application/json" https://localhost:5001/

Output:



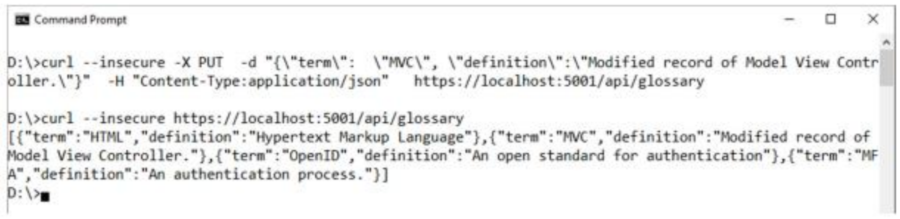
```
Command Prompt

D:\>curl --insecure -X POST  -d "{\"term\":  \"MFA\", \"definition\":\"An authentication process.\"}"  -H
 "Content-Type:application/json"   https://localhost:5001/api/glossary
{"term":"MFA","definition":"An authentication process."}
D:\>curl --insecure https://localhost:5001/api/glossary
[{"term":"HTML","definition":"Hypertext Markup Language"},{"term":"MVC","definition":"Model View Controll
er"},{"term":"OpenID","definition":"An open standard for authentication"},{"term":"MFA","definition":"An
authentication process."}]
D:\>
```

4) Update Item

Command:

curl --insecure -X PUT -d "{\"term\": \"MVC\", \"definition\":\"Modified record of Model View Controller.\"}" -H "Content-Type:application/json" https://localhost:5001/api/glossary

Output:



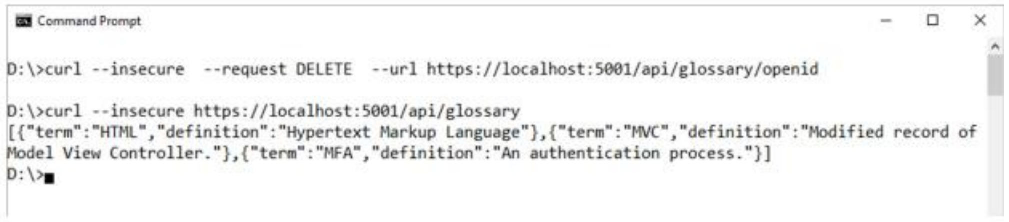
```
Command Prompt

D:\>curl --insecure -X PUT  -d "{\"term\":  \"MVC\", \"definition\":\"Modified record of Model View Contr
oller.\"}"  -H "Content-Type:application/json"   https://localhost:5001/api/glossary

D:\>curl --insecure https://localhost:5001/api/glossary
[{"term":"HTML","definition":"Hypertext Markup Language"},{"term":"MVC","definition":"Modified record of
Model View Controller."},{"term":"OpenID","definition":"An open standard for authentication"},{"term":"MF
A","definition":"An authentication process."}]
D:\>
```

5) Delete Item

Command:

Curl—insecure—request DELETE— url https://localhost:5001/api/glossary/openid

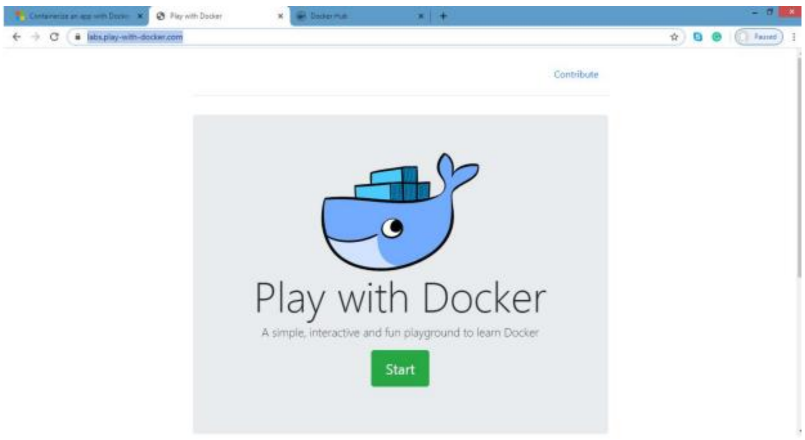
Output:



```
Command Prompt

D:\>curl --insecure  --request DELETE  --url https://localhost:5001/api/glossary/openid

D:\>curl --insecure https://localhost:5001/api/glossary
[{"term":"HTML","definition":"Hypertext Markup Language"},{"term":"MVC","definition":"Modified record of
Model View Controller."},{"term":"MFA","definition":"An authentication process."}]
D:\>
```

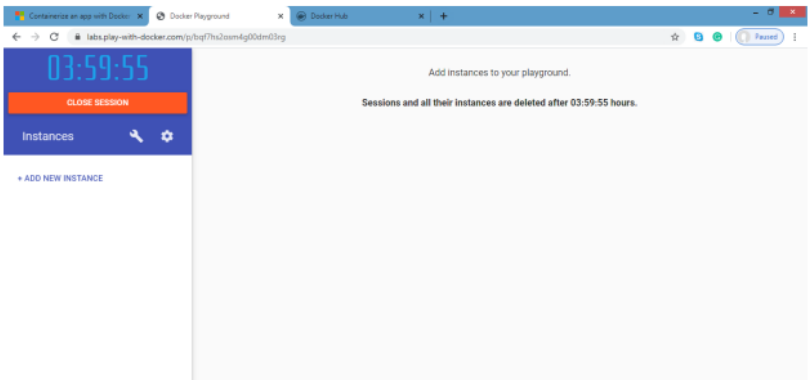
Practical 3:
Working with docker

1) Create Docker Hub account (sign up)
2) login to https://labs.play-with-docker.com/


labs.play-with-docker.com
Contribute
Play with Docker
A simple, interactive and fun playground to learn Docker
Start

Click on start

3) add new instance


03:59:55
CLOSE SESSION
Instances
+ ADD NEW INSTANCE
Add instances to your playground.
Sessions and all their instances are deleted after 03:59:55 hours.

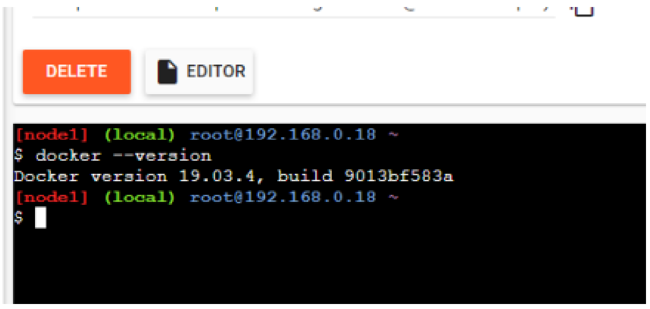
4) Perform following:

Method1:

To pull and push images using docker Command: to check docker version

docker –version

output:

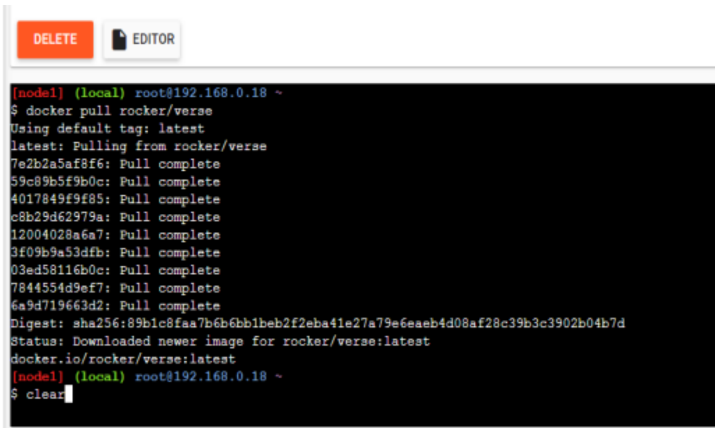


```
[node1] (local) root@192.168.0.18 ~
$ docker --version
Docker version 19.03.4, build 9013bf583a
[node1] (local) root@192.168.0.18 ~
$
```

Command: to pull readymade image

docker pull rocker/verse

output:

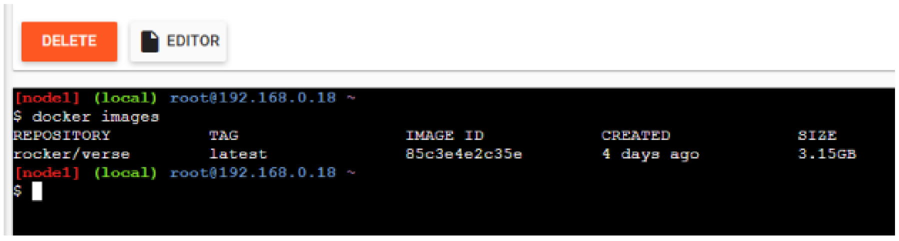


```
[node1] (local) root@192.168.0.18 ~
$ docker pull rocker/verse
Using default tag: latest
latest: Pulling from rocker/verse
7e2b2a5af8f6: Pull complete
59c89b5f9b0c: Pull complete
4017849f9f85: Pull complete
c8b29d62979a: Pull complete
12004028a6a7: Pull complete
3f09b9a53dfb: Pull complete
03ed58116b0c: Pull complete
7844554d9ef7: Pull complete
6a9d719663d2: Pull complete
Digest: sha256:89b1c8faa7b6b6bb1beb2f2eba41e27a79e6eaeb4d08af28c39b3c3902b04b7d
Status: Downloaded newer image for rocker/verse:latest
docker.io/rocker/verse:latest
[node1] (local) root@192.168.0.18 ~
$ clear
```

Command: to check images in docker

docker images

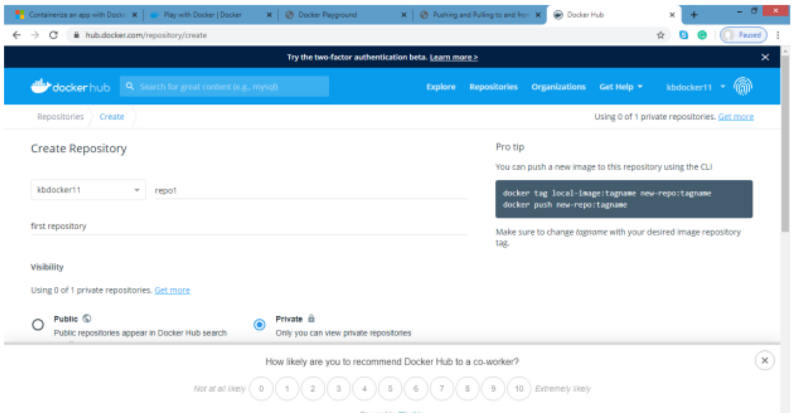
output:



```
[node1] (local) root@192.168.0.18 ~
$ docker images
REPOSITORY          TAG                 IMAGE ID            CREATED             SIZE
rocker/verse        latest              85c3e4e2c35e        4 days ago          3.15GB
[node1] (local) root@192.168.0.18 ~
$
```

Now Login to docker hub and create repository

Output:


Containerize an app with Docker
Play with Docker | Docker
Docker Playground
Pushing and Pulling to and from
Docker Hub
hub.docker.com/repository/create
Paused
Try the two-factor authentication beta. Learn more >
docker hub
Search for great content (e.g., mysql)
Explore
Repositories
Organizations
Get Help
kbdocker11
Repositories
Create
Not at all likely
0
1
2
3
4
5
6
7
8
9
10
Extremely likely

Using 0 of 1 private repositories. Get more

Create Repository

kbdocker11 repo1

first repository

Visibility

Using 0 of 1 private repositories. Get more

Public: Public repositories appear in Docker Hub search

[✓ Private: Only you can view private repositories]

Pro tip

You can push a new image to this repository using the CLI

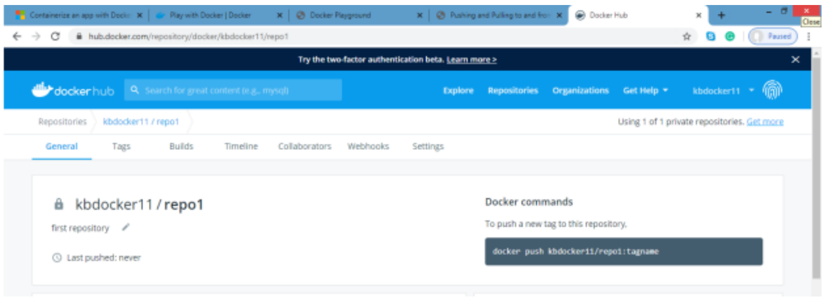
```
docker tag local-image:tagname new-repo:tagname
docker push new-repo:tagname
```

Make sure to change *tagname* with your desired image repository tag.

How likely are you to recommend Docker Hub to a co-worker?

Click on Create button

Now check repository created


Containerize an app with Docker
Play with Docker | Docker
Docker Playground
Pushing and Pulling to and from
Docker Hub
Close
hub.docker.com/repository/docker/kbdocker11/repo1
Paused
Try the two-factor authentication beta. Learn more >
docker hub
Search for great content (e.g., mysql)
Explore
Repositories
Organizations
Get Help
kbdocker11
Repositories
kbdocker11 / repo1
General
Tags
Builds
Timeline
Collaborators
Webhooks
Settings

Using 1 of 1 private repositories. Get more

kbdocker11 / repo1

first repository

Last pushed: never

Docker commands

To push a new tag to this repository.

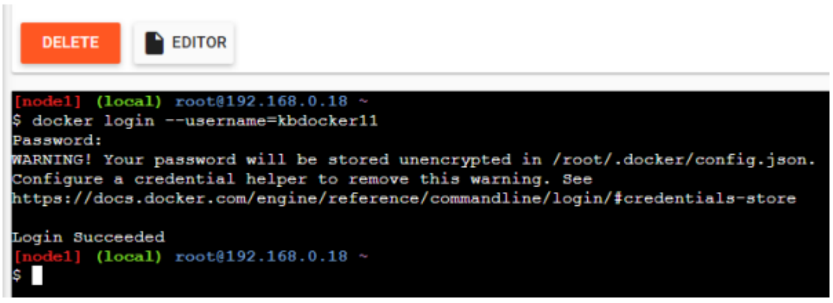
```
docker push kbdocker11/repo1:tagname
```

Command: to login to your docker account

docker login –username=kbdocker11

password:

Output:

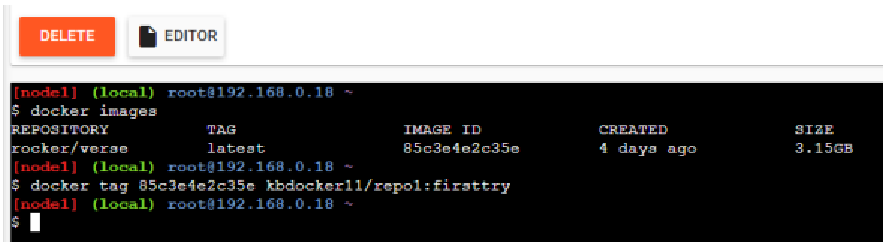


```
[node1] (local) root@192.168.0.18 ~
$ docker login --username=kbdocker11
Password:
WARNING! Your password will be stored unencrypted in /root/.docker/config.json.
Configure a credential helper to remove this warning. See
https://docs.docker.com/engine/reference/commandline/login/#credentials-store

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

Command : to tag image

docker tag 8c3e4e2c3e kbdocker11/repo1:firsttry

Output:

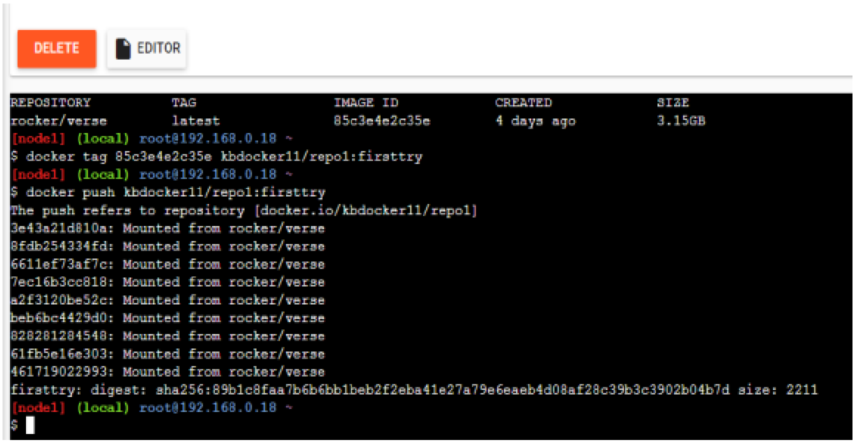


```
[node1] (local) root@192.168.0.18 ~
$ docker images
REPOSITORY          TAG                 IMAGE ID            CREATED             SIZE
rocker/verse        latest              85c3e4e2c35e        4 days ago          3.15GB
[node1] (local) root@192.168.0.18 ~
$ docker tag 85c3e4e2c35e kbdocker11/repo1:firsttry
[node1] (local) root@192.168.0.18 ~
$
```

Command: to push image to docker hub account

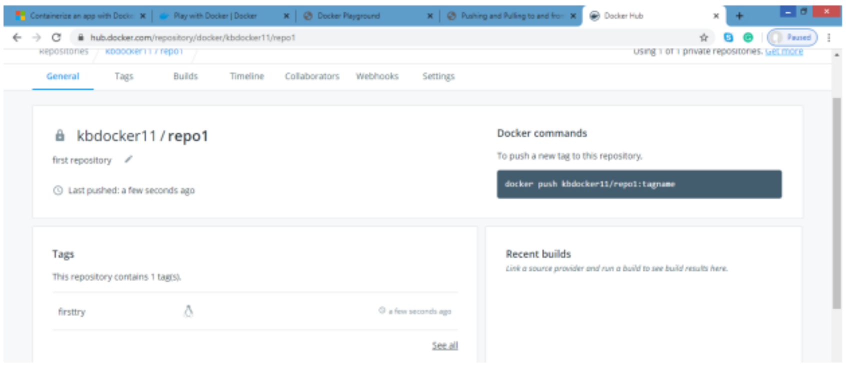
docker push kbdocker11/repo1:firsttry

Output



```
REPOSITORY          TAG                 IMAGE ID            CREATED             SIZE
rocker/verse        latest              85c3e4e2c35e        4 days ago          3.15GB
[node1] (local) root@192.168.0.18 ~
$ docker tag 85c3e4e2c35e kbdocker11/repo1:firsttry
[node1] (local) root@192.168.0.18 ~
$ docker push kbdocker11/repo1:firsttry
The push refers to repository [docker.io/kbdocker11/repo1]
3e43a21d810a: Mounted from rocker/verse
8fdb254334fd: Mounted from rocker/verse
6611ef73af7c: Mounted from rocker/verse
7ec16b3cc818: Mounted from rocker/verse
a2f3120be52c: Mounted from rocker/verse
beb6bc4429d0: Mounted from rocker/verse
828281284548: Mounted from rocker/verse
61fb5e16e303: Mounted from rocker/verse
461719022993: Mounted from rocker/verse
firsttry: digest: sha256:89b1c8faa7b6b6bb1beb2f2eba41e27a79e6eaeb4d08af28c39b3c3902b04b7d size: 2211
[node1] (local) root@192.168.0.18 ~
$
```

Check it in docker hub now



Containerize an app with Docke
Play with Docker | Docker
Docker Playground
Pushing and Pulling to and fro
Docker Hub
hub.docker.com/repository/docker/kbdocker11/repo1
Paused
Repositories / kbdocker11 / repo1
Using 1 of 1 private repositories. Get more
General
Tags
Builds
Timeline
Collaborators
Webhooks
Settings

kbdocker11 / repo1

first repository

Last pushed: a few seconds ago

Docker commands

To push a new tag to this repository.

```
docker push kbdocker11/repo1:tagname
```

Tags

This repository contains 1 tag(s).

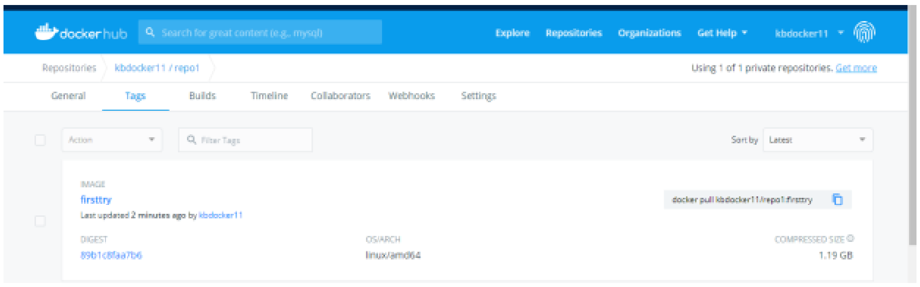
firsttry a few seconds ago

See all

Recent builds

Link a source provider and run a build to see build results here.

Click on tags and check



Repositories kbdocker11 / repo1

Using 1 of 1 private repositories. Get more

IMAGE
firsttry
Last updated 2 minutes ago by kbdocker11

docker pull kbdocker11/repo1:firsttry

DIGEST
89b1c8faa7b6

OS/ARCH
linux/amd64

COMPRESSED SIZE
1.19 GB

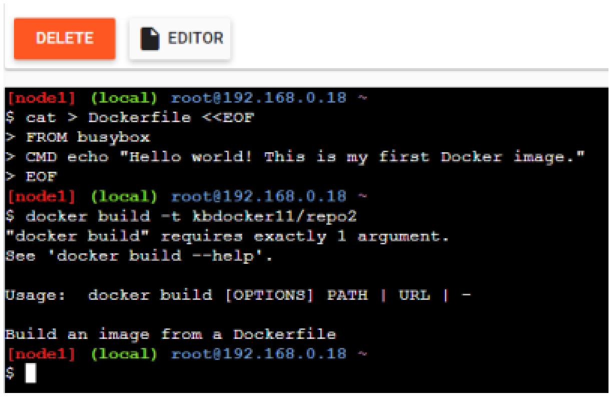
Method 2:

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:

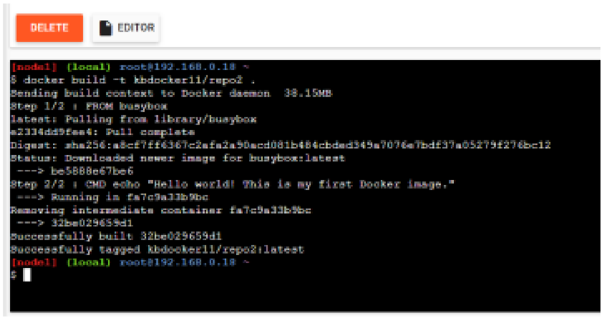


```
[node1] (local) root@192.168.0.18 ~
$ cat > Dockerfile <<EOF
> FROM busybox
> CMD echo "Hello world! This is my first Docker image."
> EOF
[node1] (local) root@192.168.0.18 ~
$ docker build -t kbdocker11/repo2
"docker build" requires exactly 1 argument.
See 'docker build --help'.

Usage:  docker build [OPTIONS] PATH | URL | -

Build an image from a Dockerfile
[node1] (local) root@192.168.0.18 ~
$ 
```

Command : to build image from docker file

dokcer build –t kbdocker11/repo2 .

Output:

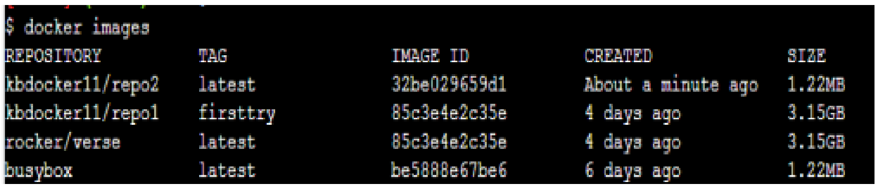


```
[node1] (local) root@192.168.0.18 ~
$ docker build -t kbdocker11/repo2 .
Sending build context to Docker daemon  38.15MB
Step 1/2 : FROM busybox
latest: Pulling from library/busybox
e2334dd9fee4: Pull complete
Digest: sha256:a8cf7ff6367c2afa2a90acd081b484cbded349a7076e7bdf37a05279f276bc12
Status: Downloaded newer image for busybox:latest
 ---> be5888e67be6
Step 2/2 : CMD echo "Hello world! This is my first Docker image."
 ---> Running in fa7c9a33b9bc
Removing intermediate container fa7c9a33b9bc
 ---> 32be029659d1
Successfully built 32be029659d1
Successfully tagged kbdocker11/repo2:latest
[node1] (local) root@192.168.0.18 ~
$ 
```

Command: to check docker images

docker images

output:

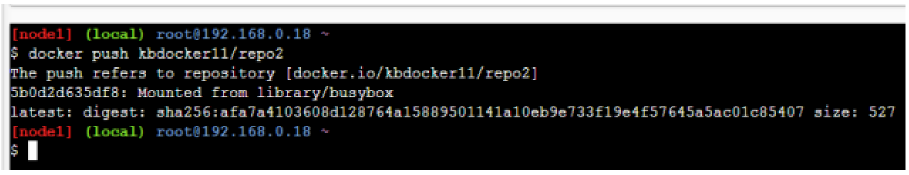


```
$ docker images
REPOSITORY          TAG                 IMAGE ID            CREATED              SIZE
kbdocker11/repo2    latest              32be029659d1        About a minute ago   1.22MB
kbdocker11/repo1    firsttry            85c3e4e2c35e        4 days ago           3.15GB
rocker/verse        latest              85c3e4e2c35e        4 days ago           3.15GB
busybox             latest              be5888e67be6        6 days ago           1.22MB
```

Command: to push image to docker hub

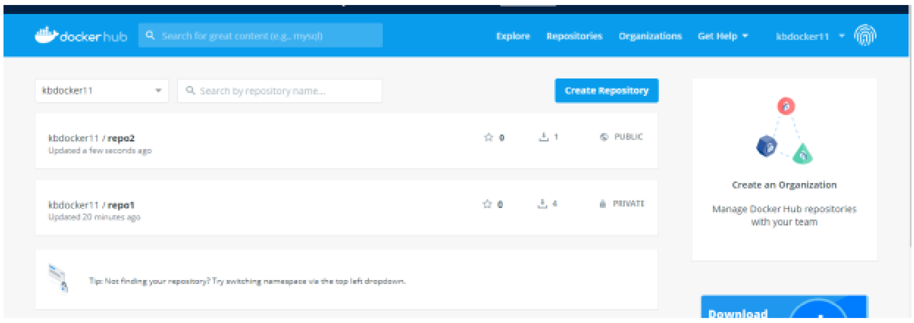
docker push kbdocker11/repo2 .

Output:



```
[node1] (local) root@192.168.0.18 ~
$ docker push kbdocker11/repo2
The push refers to repository [docker.io/kbdocker11/repo2]
5b0d2d635df8: Mounted from library/busybox
latest: digest: sha256:afa7a4103608d128764a1588950114la10eb9e733f19e4f57645a5ac01c85407 size: 527
[node1] (local) root@192.168.0.18 ~
$ 
```

Now check it on docker hub



docker hub
Search for great content (e.g., mysql)
Explore
Repositories
Organizations
Get Help
kbdocker11
Search by repository name...
Create Repository
Create an Organization
Manage Docker Hub repositories with your team
Download

kbdocker11 / repo2 — Updated a few seconds ago — ☆ 0 — 1 — PUBLIC

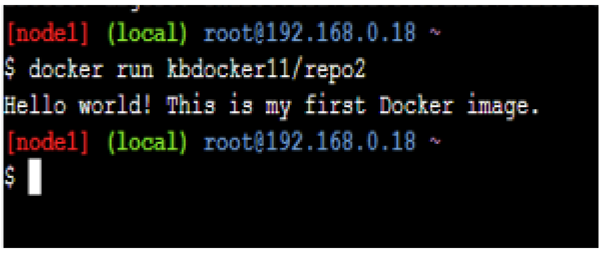
kbdocker11 / repo1 — Updated 20 minutes ago — ☆ 0 — 4 — PRIVATE

Tip: Not finding your repository? Try switching namespace via the top left dropdown.

command: to run docker image:

docker run kbdocker11/repo2

output:



```
[node1] (local) root@192.168.0.18 ~
$ docker run kbdocker11/repo2
Hello world! This is my first Docker image.
[node1] (local) root@192.168.0.18 ~
$ 
```

Now close session.

Practical 4

Installing software packages on Docker, Working with Docker Volumes and Networks

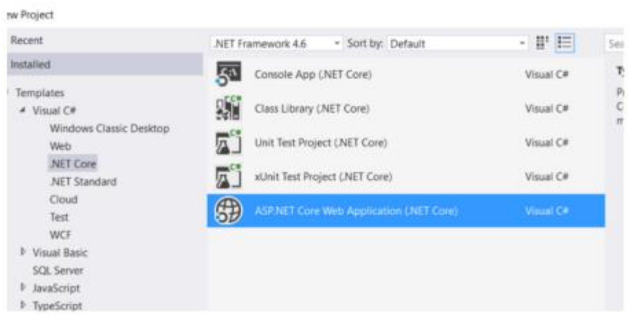
What is the software required for Windows?

- ☆ Windows 10 is required for Docker installation
- ☆ Visual Studio 2017 has built-in support for Docker, so this is highly recommended
- ☆ .Net Core SDK
- ☆ Docker for Windows
- ☆ Docker Tools

How to create a new microservice using .NET Core & then build & run it using Docker

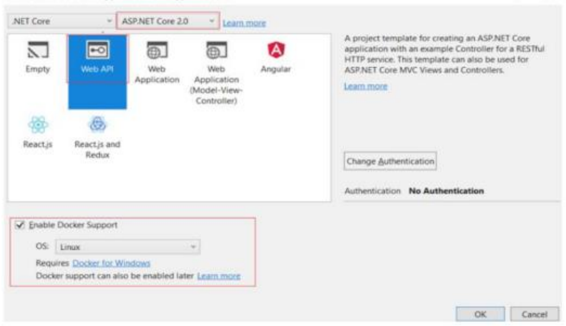
Step 1: Create a microservice(.NET Core WebAPI) with Docker support as shown below:

Select "ASP.NET Core Web Application (.NET Core)" from the drop down menu.



Recent
Installed
Templates
Visual C#
Windows Classic Desktop
Web
.NET Core
.NET Standard
Cloud
Test
WCF
Visual Basic
SQL Server
JavaScript
TypeScript
.NET Framework 4.6
Sort by: Default
Console App (.NET Core) Visual C#
Class Library (.NET Core) Visual C#
Unit Test Project (.NET Core) Visual C#
xUnit Test Project (.NET Core) Visual C#
ASP.NET Core Web Application (.NET Core) Visual C#

Select the "Enable Docker Support" option.

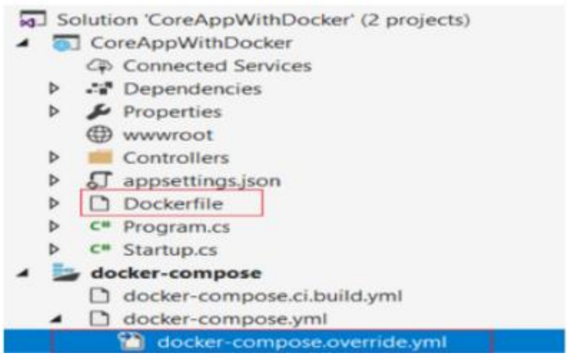


A project template for creating an ASP.NET Core application with an example Controller for a RESTful HTTP service. This template can also be used for ASP.NET Core MVC Views and Controllers.

Authentication: No Authentication

- [✓] Enable Docker Support
- OS: Linux
- Requires Docker for Windows
- Docker support can also be enabled later Learn more

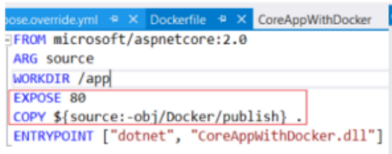
The following Application Sturcture will be created along with "Docker File".



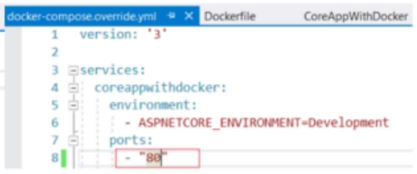
Dockerfile
This file is the entry point for running any Docker application.

It is used to build an image of the application's published code in "obj/Docker/publish."

docker-compose.override.yml
This file is used when running an application using Visual Studio.



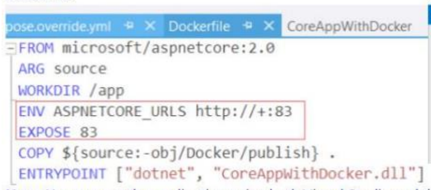
```
FROM microsoft/aspnetcore:2.0
ARG source
WORKDIR /app
EXPOSE 80
COPY ${source:-obj/Docker/publish} .
ENTRYPOINT ["dotnet", "CoreAppWithDocker.dll"]
```



```
version: '3'

services:
  coreappwithdocker:
    environment:
      - ASPNETCORE_ENVIRONMENT=Development
    ports:
      - "80"
```

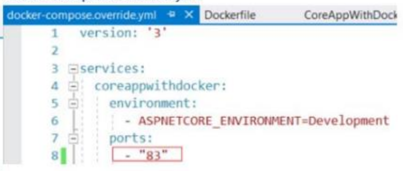
Step 2: Update Dockerfile & docker-compose.override.yml as shown below & build the application. 80 is the default Docker cointainer port, so you should update it to a different port number, like 83.

Dockerfile



```
FROM microsoft/aspnetcore:2.0
ARG source
WORKDIR /app
ENV ASPNETCORE_URLS http://+:83
EXPOSE 83
COPY ${source:-obj/Docker/publish} .
ENTRYPOINT ["dotnet", "CoreAppWithDocker.dll"]
```

docker-compose.override.yml

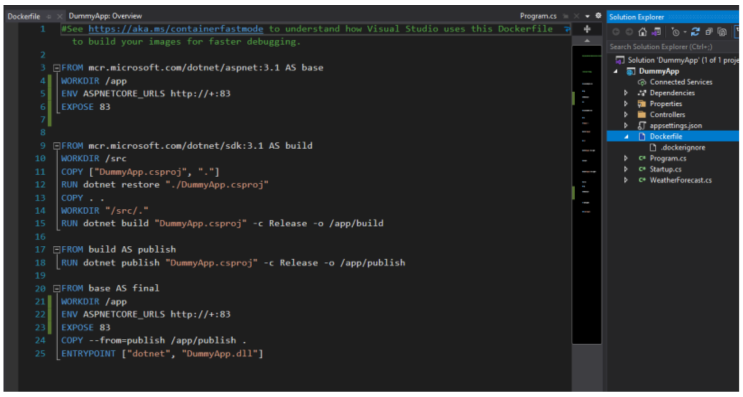


```
version: '3'

services:
  coreappwithdocker:
    environment:
      - ASPNETCORE_ENVIRONMENT=Development
    ports:
      - "83"
```

Note: You can run the application using both Visual Studio and the Docker command line.

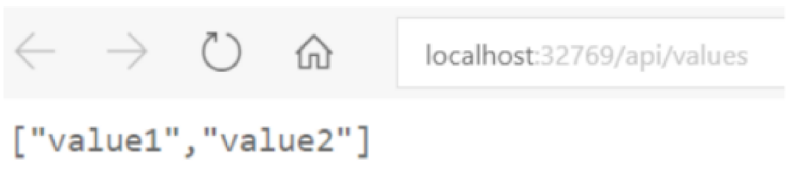
- ☆ First Line FROM Microsoft/aspnetcore:2.0 is the base image for this application in order to run the net core application
- ☆ ARG source is the argument which helps to pass data to the image.
- ☆ WORKDIR/app is the working directory of the image; it will store all DLLs inside the app folder.
- ☆ COPY will copy the DLLs of the application to the root directory of the image.
- ☆ ENTRYPOINT is responsible to run the main application with the help of ASPNETCOREAPP.dll.

In the case of .NET Core 3.1 the Docker file will look like this



```
#See https://aka.ms/containerfastmode to understand how Visual Studio uses this Dockerfile to build your images for faster debugging.

FROM mcr.microsoft.com/dotnet/aspnet:3.1 AS base
WORKDIR /app
ENV ASPNETCORE_URLS http://+:83
EXPOSE 83


FROM mcr.microsoft.com/dotnet/sdk:3.1 AS build
WORKDIR /src
COPY ["DummyApp.csproj", "."]
RUN dotnet restore "./DummyApp.csproj"
COPY . .
WORKDIR "/src/."
RUN dotnet build "DummyApp.csproj" -c Release -o /app/build

FROM build AS publish
RUN dotnet publish "DummyApp.csproj" -c Release -o /app/publish

FROM base AS final
WORKDIR /app
ENV ASPNETCORE_URLS http://+:83
EXPOSE 83
COPY --from=publish /app/publish .
ENTRYPOINT ["dotnet", "DummyApp.dll"]
```

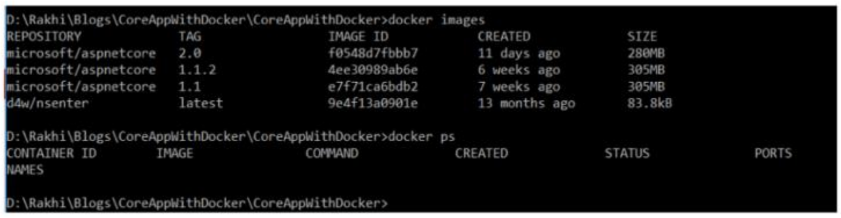
Step 3: Run the application using Visual Studio.



localhost:32769/api/values

```
["value1","value2"]
```

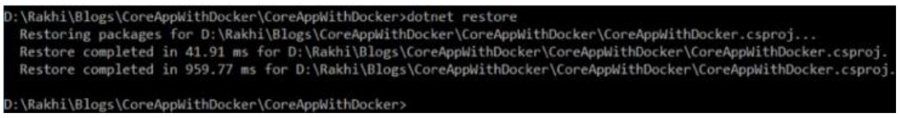
Step 4: Run the application using Docker Command. Open Application folder fromm command prompt & check the existing images using Docker images & running cointainers using Docker PS.



```
D:\Rakhi\Blogs\CoreAppWithDocker\CoreAppWithDocker>docker images
REPOSITORY          TAG                 IMAGE ID            CREATED             SIZE
microsoft/aspnetcore   2.0              f0548d7fbbb7        11 days ago         280MB
microsoft/aspnetcore   1.1.2            4ee30989ab6e        6 weeks ago         305MB
microsoft/aspnetcore   1.1              e7f71ca6bdb2        7 weeks ago         305MB
d4w/nsenter         latest              9e4f13a0901e        13 months ago       83.8kB

D:\Rakhi\Blogs\CoreAppWithDocker\CoreAppWithDocker>docker ps
CONTAINER ID        IMAGE               COMMAND             CREATED             STATUS              PORTS
NAMES

D:\Rakhi\Blogs\CoreAppWithDocker\CoreAppWithDocker>
```

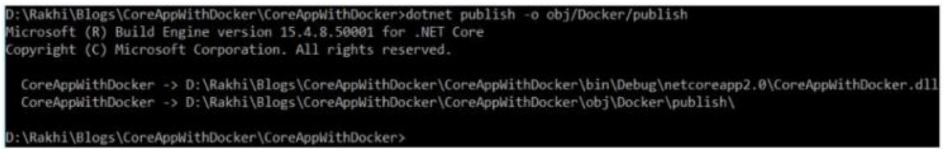
As you can see, there is no running cointainer. So, run the following commands to create build:

To restore packages: dotnet restore



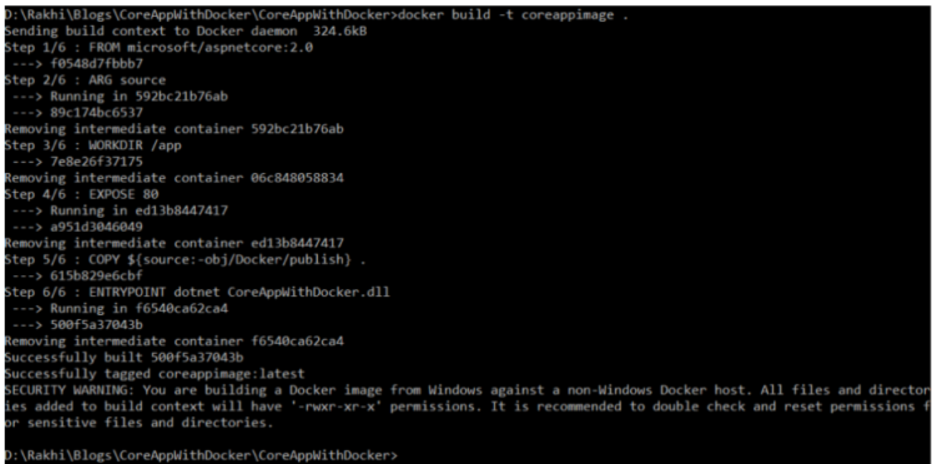
```
D:\Rakhi\Blogs\CoreAppWithDocker\CoreAppWithDocker>dotnet restore
  Restoring packages for D:\Rakhi\Blogs\CoreAppWithDocker\CoreAppWithDocker\CoreAppWithDocker.csproj...
  Restore completed in 41.91 ms for D:\Rakhi\Blogs\CoreAppWithDocker\CoreAppWithDocker\CoreAppWithDocker.csproj.
  Restore completed in 959.77 ms for D:\Rakhi\Blogs\CoreAppWithDocker\CoreAppWithDocker\CoreAppWithDocker.csproj.

D:\Rakhi\Blogs\CoreAppWithDocker\CoreAppWithDocker>
```

To publish the application code: dotnet publish-o obj/Docker/publish



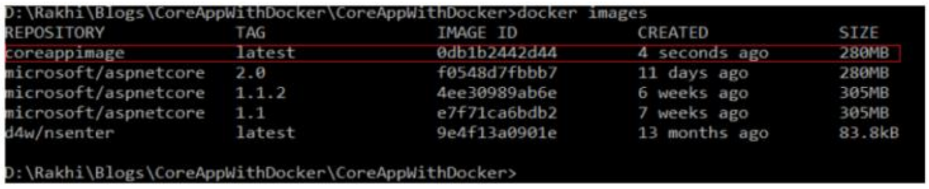
```
D:\Rakhi\Blogs\CoreAppWithDocker\CoreAppWithDocker>dotnet publish -o obj/Docker/publish
Microsoft (R) Build Engine version 15.4.8.50001 for .NET Core
Copyright (C) Microsoft Corporation. All rights reserved.

  CoreAppWithDocker -> D:\Rakhi\Blogs\CoreAppWithDocker\CoreAppWithDocker\bin\Debug\netcoreapp2.0\CoreAppWithDocker.dll
  CoreAppWithDocker -> D:\Rakhi\Blogs\CoreAppWithDocker\CoreAppWithDocker\obj\Docker\publish\

D:\Rakhi\Blogs\CoreAppWithDocker\CoreAppWithDocker>
```

To build the image: docker build-t imagename



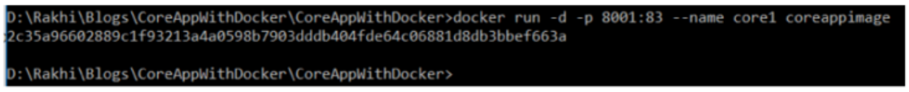
```
D:\Rakhi\Blogs\CoreAppWithDocker\CoreAppWithDocker>docker build -t coreappimage .
Sending build context to Docker daemon  324.6kB
Step 1/6 : FROM microsoft/aspnetcore:2.0
 ---> f0548d7fbbb7
Step 2/6 : ARG source
 ---> Running in 592bc21b76ab
 ---> 89c174bc6537
Removing intermediate container 592bc21b76ab
Step 3/6 : WORKDIR /app
 ---> 7e8e26f37175
Removing intermediate container 06c848058834
Step 4/6 : EXPOSE 80
 ---> Running in ed13b8447417
 ---> a951d3046049
Removing intermediate container ed13b8447417
Step 5/6 : COPY ${source:-obj/Docker/publish} .
 ---> 615b829e6cbf
Step 6/6 : ENTRYPOINT dotnet CoreAppWithDocker.dll
 ---> Running in f6540ca62ca4
 ---> 500f5a37043b
Removing intermediate container f6540ca62ca4
Successfully built 500f5a37043b
Successfully tagged coreappimage:latest
SECURITY WARNING: You are building a Docker image from Windows against a non-Windows Docker host. All files and director
ies added to build context will have '-rwxr-xr-x' permissions. It is recommended to double check and reset permissions f
or sensitive files and directories.

D:\Rakhi\Blogs\CoreAppWithDocker\CoreAppWithDocker>
```

Now, check the newly created image "coreappimage" in Docker images



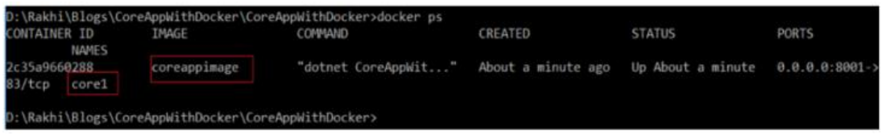
```
D:\Rakhi\Blogs\CoreAppWithDocker\CoreAppWithDocker>docker images
REPOSITORY             TAG                 IMAGE ID            CREATED             SIZE
coreappimage           latest              0db1b2442d44        4 seconds ago       280MB
microsoft/aspnetcore   2.0                 f0548d7fbbb7        11 days ago         280MB
microsoft/aspnetcore   1.1.2               4ee30989ab6e        6 weeks ago         305MB
microsoft/aspnetcore   1.1                 e7f71ca6bdb2        7 weeks ago         305MB
d4w/nsenter            latest              9e4f13a0901e        13 months ago       83.8kB

D:\Rakhi\Blogs\CoreAppWithDocker\CoreAppWithDocker>
```

Run in image in a container: docker run-d-p 8001:83 -name core1 coreappimage



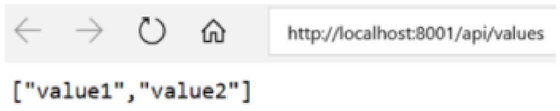
```
D:\Rakhi\Blogs\CoreAppWithDocker\CoreAppWithDocker>docker run -d -p 8001:83 --name core1 coreappimage
2c35a96602889c1f93213a4a0598b7903dddb404fde64c06881d8db3bbef663a

D:\Rakhi\Blogs\CoreAppWithDocker\CoreAppWithDocker>
```

Check the running container: Docker PS



```
D:\Rakhi\Blogs\CoreAppWithDocker\CoreAppWithDocker>docker ps
CONTAINER ID        IMAGE               COMMAND                  CREATED              STATUS              PORTS
        NAMES
2c35a9660288        coreappimage        "dotnet CoreAppWit..."   About a minute ago   Up About a minute   0.0.0.0:8001->
83/tcp   core1

D:\Rakhi\Blogs\CoreAppWithDocker\CoreAppWithDocker>
```

Now the application is running in the Core1 container with the http://localhost:8001.



http://localhost:8001/api/values

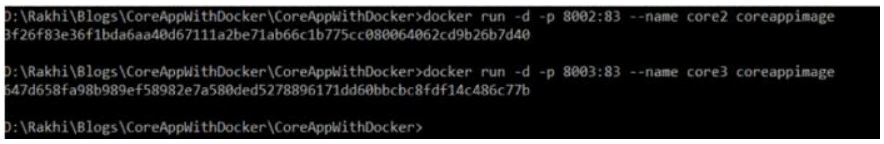
```
["value1","value2"]
```

Case1: Run the same image in multiple containers

We can run the same image in multiple containers at the same time by using:

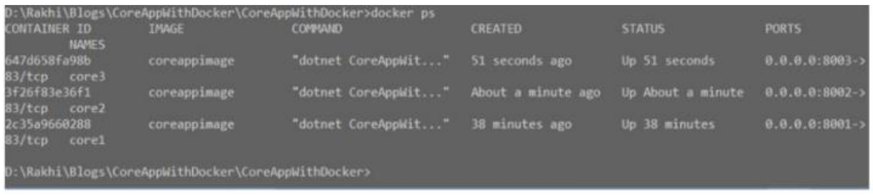
docker run-d-p 8002:83 -name core2 coreappimage

docker run-d-p 8003:83 -name core3 coreappimage



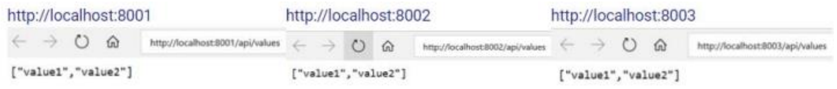
```
D:\Rakhi\Blogs\CoreAppWithDocker\CoreAppWithDocker>docker run -d -p 8002:83 --name core2 coreappimage
3f26f83e36f1bda6aa40d67111a2be71ab66c1b775cc080064062cd9b26b7d40

D:\Rakhi\Blogs\CoreAppWithDocker\CoreAppWithDocker>docker run -d -p 8003:83 --name core3 coreappimage
647d658fa98b989ef58982e7a580ded5278896171dd60bbcbc8fdf14c486c77b

D:\Rakhi\Blogs\CoreAppWithDocker\CoreAppWithDocker>
```

Check the running containers by using Docker PS.



```
D:\Rakhi\Blogs\CoreAppWithDocker\CoreAppWithDocker>docker ps
CONTAINER ID        IMAGE               COMMAND                  CREATED              STATUS              PORTS
        NAMES
647d658fa98b        coreappimage        "dotnet CoreAppWit..."   51 seconds ago       Up 51 seconds       0.0.0.0:8003->
83/tcp   core3
3f26f83e36f1        coreappimage        "dotnet CoreAppWit..."   About a minute ago   Up About a minute   0.0.0.0:8002->
83/tcp   core2
2c35a9660288        coreappimage        "dotnet CoreAppWit..."   38 minutes ago       Up 38 minutes       0.0.0.0:8001->
83/tcp   core1

D:\Rakhi\Blogs\CoreAppWithDocker\CoreAppWithDocker>
```

We can see that there are 3 containers running for the same image at 8001, 8002 & 8003.



http://localhost:8001 http://localhost:8002 http://localhost:8003

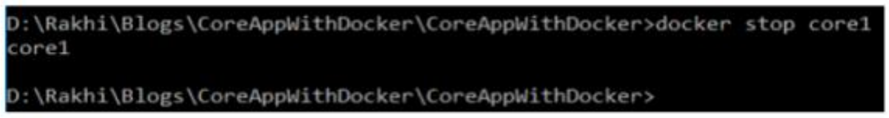
http://localhost:8001/api/values http://localhost:8002/api/values http://localhost:8003/api/values

```
["value1","value2"]    ["value1","value2"]    ["value1","value2"]
```

Case 2: Manage Cointainers: Stop/Start/Remove Containers

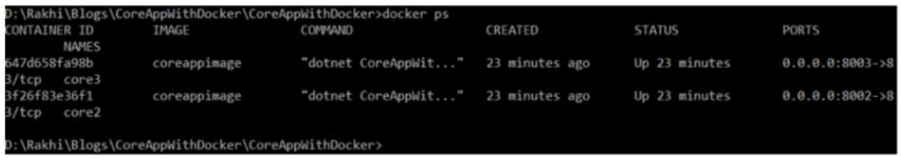
Stop Container:

We can stop any running containers using "docker stop containerid/ containername" docker stop core1.



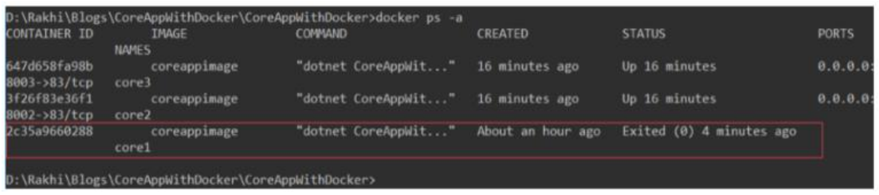
```
D:\Rakhi\Blogs\CoreAppWithDocker\CoreAppWithDocker>docker stop core1
core1

D:\Rakhi\Blogs\CoreAppWithDocker\CoreAppWithDocker>
```

Check running containers: docker ps:



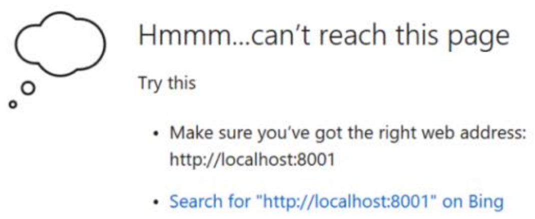
```
D:\Rakhi\Blogs\CoreAppWithDocker\CoreAppWithDocker>docker ps
CONTAINER ID        IMAGE               COMMAND                  CREATED              STATUS              PORTS
        NAMES
647d658fa98b        coreappimage        "dotnet CoreAppWit..."   23 minutes ago       Up 23 minutes       0.0.0.0:8003->8
3/tcp   core3
3f26f83e36f1        coreappimage        "dotnet CoreAppWit..."   23 minutes ago       Up 23 minutes       0.0.0.0:8002->8
3/tcp   core2

D:\Rakhi\Blogs\CoreAppWithDocker\CoreAppWithDocker>
```

Check all containers: docker ps-a



```
D:\Rakhi\Blogs\CoreAppWithDocker\CoreAppWithDocker>docker ps -a
CONTAINER ID        IMAGE               COMMAND                  CREATED             STATUS                        PORTS
               NAMES
647d658fa98b        coreappimage        "dotnet CoreAppWit..."   16 minutes ago      Up 16 minutes                 0.0.0.0:
8003->83/tcp   core3
3f26f83e36f1        coreappimage        "dotnet CoreAppWit..."   16 minutes ago      Up 16 minutes                 0.0.0.0:
8002->83/tcp   core2
2c35a9660288        coreappimage        "dotnet CoreAppWit..."   About an hour ago   Exited (0) 4 minutes ago
               core1

D:\Rakhi\Blogs\CoreAppWithDocker\CoreAppWithDocker>
```

localhost:8001/api/values



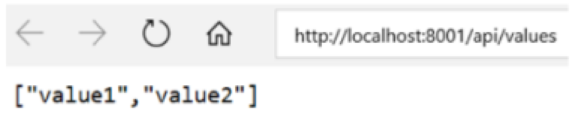
Hmmm...can't reach this page

Try this

- Make sure you've got the right web address: http://localhost:8001
- Search for "http://localhost:8001" on Bing

Start Container:

We can start a stopped container using "docker start containerid/ containername" => docker start core1



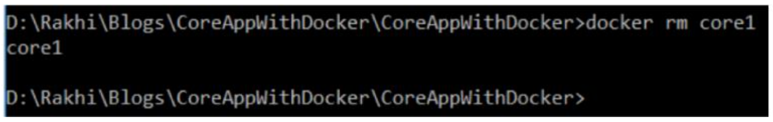
http://localhost:8001/api/values

```
["value1","value2"]
```

Remove Container:

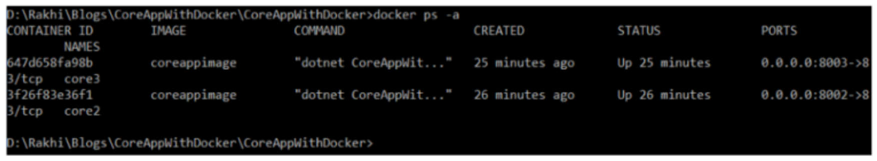
We can remove any stopped container, but then we will not be able to start it again.

so first we should stop the container before removing it: =>docker stop core1 => docker rm core1



```
D:\Rakhi\Blogs\CoreAppWithDocker\CoreAppWithDocker>docker rm core1
core1

D:\Rakhi\Blogs\CoreAppWithDocker\CoreAppWithDocker>
```

Now this container will not be listed on the containers list:



```
D:\Rakhi\Blogs\CoreAppWithDocker\CoreAppWithDocker>docker ps -a
CONTAINER ID        IMAGE               COMMAND                  CREATED             STATUS              PORTS
       NAMES
647d658fa98b        coreappimage        "dotnet CoreAppWit..."   25 minutes ago      Up 25 minutes       0.0.0.0:8003->8
3/tcp   core3
3f26f83e36f1        coreappimage        "dotnet CoreAppWit..."   26 minutes ago      Up 26 minutes       0.0.0.0:8002->8
3/tcp   core2

D:\Rakhi\Blogs\CoreAppWithDocker\CoreAppWithDocker>
```

Case 3 – Share the Application on Docker Hub Repository

Share the application

Create a repo

To push an image, we first need to create a repository on Docker Hub.

1. Sign up or Sign in to Docker Hub.
2. Click the Create Repository button.
3. For the repo name, use getting-started. Make sure the visibility is Public.
4. Click the Create button!

Docker Commands

To push a new tag to this repository,

```
docker push docker/getting-started:tagname
```

Push the image

1. In the command line, try running the push command on Docker Hub.

```
$ docker push docker/getting-started
The push refers to repository [docker.io/docker/getting-started]
An image does not exist locally with the tag: docker/getting-started
```

2. Login to the Docker Hub using the command docker login -u YOUR-USER-NAME.
3. Use the docker tag command to give the getting-started image a new name.

```
$ docker tag getting-started YOUR-USER-NAME/getting-started
```

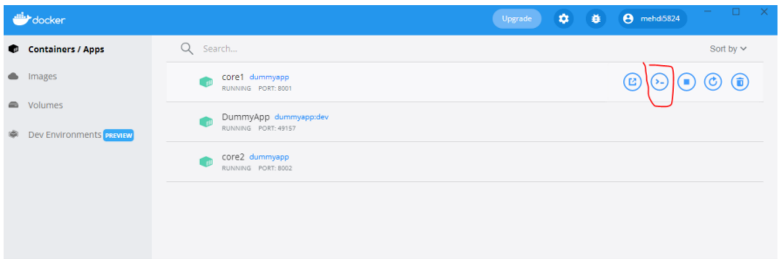
4. Now try push command again.

```
$ docker push YOUR-USER-NAME/getting-started
```

Part 2

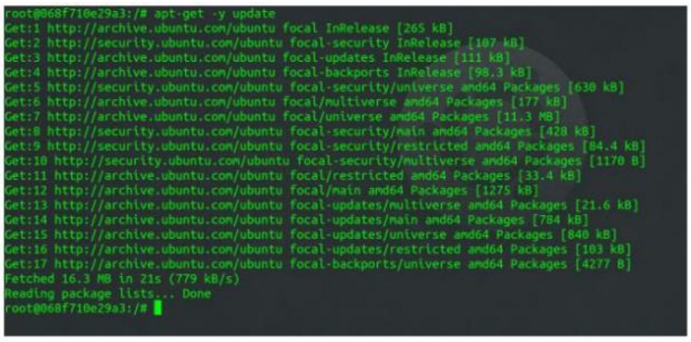
Installing Software Packages in Docker

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



Step2: Now, you have opened the bash of your Ubuntu Docker Container. To install any packages, need to update the OS.

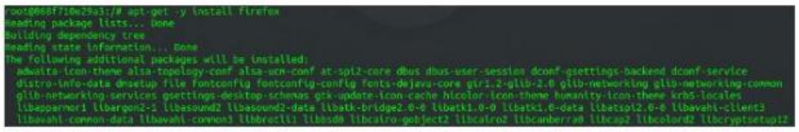
```
apt-get -y update
```



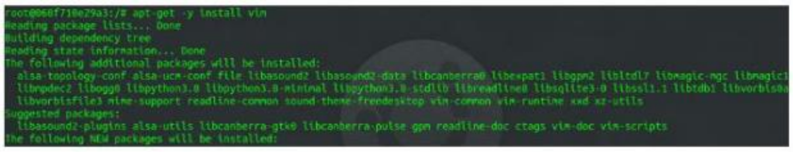
```
root@068f710e29a3:/# apt-get -y update
Get:1 http://archive.ubuntu.com/ubuntu focal InRelease [265 kB]
Get:2 http://security.ubuntu.com/ubuntu focal-security InRelease [107 kB]
Get:3 http://archive.ubuntu.com/ubuntu focal-updates InRelease [111 kB]
Get:4 http://archive.ubuntu.com/ubuntu focal-backports InRelease [98.3 kB]
Get:5 http://security.ubuntu.com/ubuntu focal-security/universe amd64 Packages [630 kB]
Get:6 http://archive.ubuntu.com/ubuntu focal/multiverse amd64 Packages [177 kB]
Get:7 http://archive.ubuntu.com/ubuntu focal/universe amd64 Packages [11.3 MB]
Get:8 http://security.ubuntu.com/ubuntu focal-security/main amd64 Packages [428 kB]
Get:9 http://security.ubuntu.com/ubuntu focal-security/restricted amd64 Packages [84.4 kB]
Get:10 http://security.ubuntu.com/ubuntu focal-security/multiverse amd64 Packages [1170 B]
Get:11 http://archive.ubuntu.com/ubuntu focal/restricted amd64 Packages [33.4 kB]
Get:12 http://archive.ubuntu.com/ubuntu focal/main amd64 Packages [1275 kB]
Get:13 http://archive.ubuntu.com/ubuntu focal-updates/multiverse amd64 Packages [21.6 kB]
Get:14 http://archive.ubuntu.com/ubuntu focal-updates/main amd64 Packages [784 kB]
Get:15 http://archive.ubuntu.com/ubuntu focal-updates/universe amd64 Packages [840 kB]
Get:16 http://archive.ubuntu.com/ubuntu focal-updates/restricted amd64 Packages [103 kB]
Get:17 http://archive.ubuntu.com/ubuntu focal-backports/universe amd64 Packages [4277 B]
Fetched 16.3 MB in 21s (779 kB/s)
Reading package lists... Done
root@068f710e29a3:/#
```

Step3: After you have updated the Docker Container, now install the Firefox & Vim packages inside it.

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



```
root@068f710e29a3:/# apt-get -y install firefox
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following additional packages will be installed:
  adwaita-icon-theme alsa-topology-conf alsa-ucm-conf at-spi2-core dbus dbus-user-session dconf-gsettings-backend dconf-service
  distro-info-data dmsetup file fontconfig fontconfig-config fonts-dejavu-core gir1.2-glib-2.0 glib-networking glib-networking-common
  glib-networking-services gsettings-desktop-schemas gtk-update-icon-cache hicolor-icon-theme humanity-icon-theme krb5-locales
  libapparmor1 libargon2-1 libasound2 libasound2-data libatk-bridge2.0-0 libatk1.0-0 libatk1.0-data libatspi2.0-0 libavahi-client3
  libavahi-common-data libavahi-common3 libbrotli1 libbsd0 libcairo-gobject2 libcairo2 libcanberra0 libcap2 libcolord2 libcryptsetup12
```



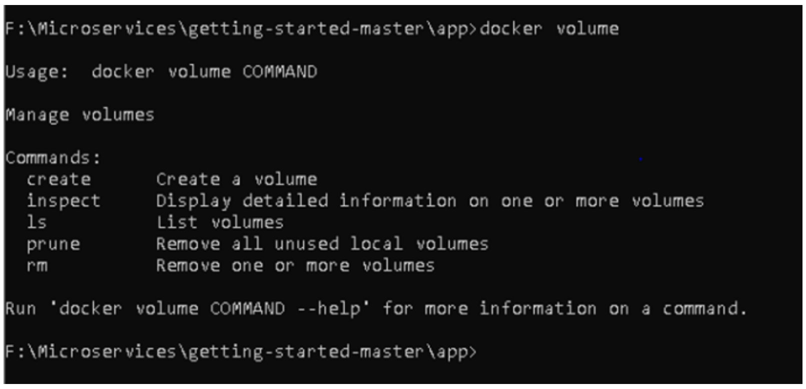
```
root@068f710e29a3:/# apt-get -y install vim
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following additional packages will be installed:
  alsa-topology-conf alsa-ucm-conf file libasound2 libasound2-data libcanberra0 libexpat1 libgpm2 libltdl7 libmagic-mgc libmagic1
  libmpdec2 libogg0 libpython3.8 libpython3.8-minimal libpython3.8-stdlib libreadline8 libsqlite3-0 libssl1.1 libtdb1 libvorbis0a
  libvorbisfile3 mime-support readline-common sound-theme-freedesktop vim-common vim-runtime xxd xz-utils
Suggested packages:
  libasound2-plugins alsa-utils libcanberra-gtk0 libcanberra-pulse gpm readline-doc ctags vim-doc vim-scripts
The following NEW packages will be installed:
```

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

Container Volumes:

Step 1: Working with Docker Volumes:

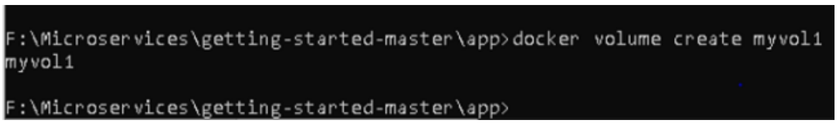
a. Let create a volume for that type command -> docker volume



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

Usage:  docker volume COMMAND

Manage volumes

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

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

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

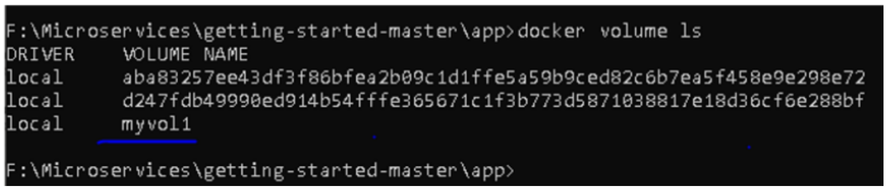
b. Now lets create the actual volume:-

➔ docker volume create myvo1



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

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

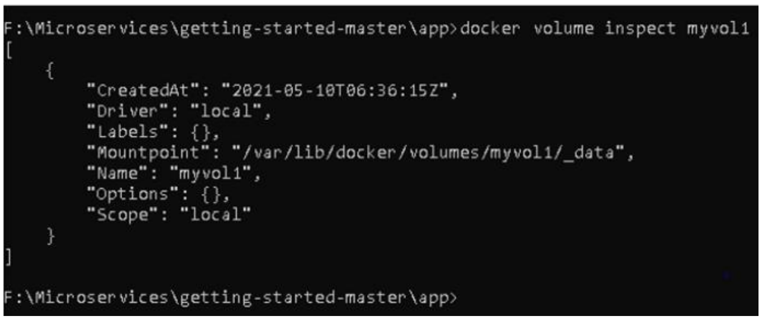
c. To list the volume we will write below command:



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

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

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

➔ docker volume inspect myvol1



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

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

e. To remove volume write command:-

➔ docker volume rm myvol1

f. To remove all unused volume write the command:

➔ docker volume prune

There are basic functionalities of docker volume.

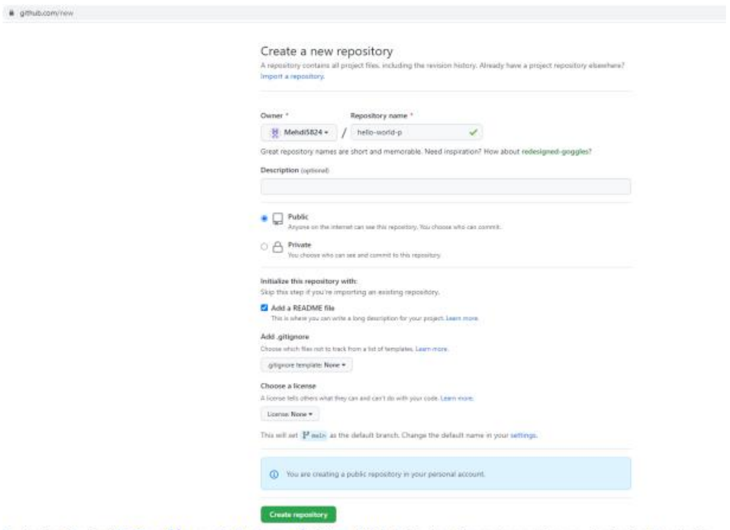
a. To connect a container to a network
   ➔ docker network connect
b. To create a network
   ➔ docker network create
c. To disconnect a container from a network
   ➔ docker network disconnect
d. To display detailed information on one or more networks
   ➔ docker network inspect
e. To list the network
   ➔ docker network ls
f. To remove one or more network
   ➔ docker network rm

Practical 5

Working with Circle CI for continuous integration

Step 1 - Create a repository

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



github.com/new

Create a new repository

A repository contains all project files, including the revision history. Already have a project repository elsewhere? Import a repository.

Owner * Mehdi5824 / Repository name * hello-world-p

Great repository names are short and memorable. Need inspiration? How about redesigned-goggles?

Description (optional)

[✓ Public] Anyone on the internet can see this repository. You choose who can commit.

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

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

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

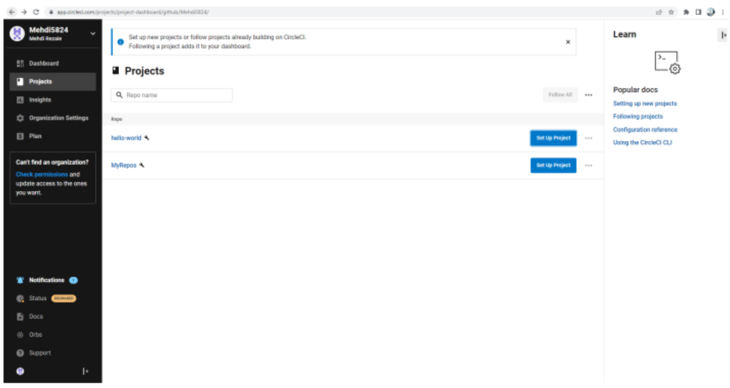
Add .gitignore
Choose which files not to track from a list of templates. Learn more.
.gitignore template: None

Choose a license
A license tells others what they can and can't do with your code. Learn more.
License: None

This will set main as the default branch. Change the default name in your settings.

You are creating a public repository in your personal account.

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



app.circleci.com/projects/project-dashboard/github/Mehdi5824/

Mehdi5824
Mehdi Rezaie

Set up new projects or follow projects already building on CircleCI.
Following a project adds it to your dashboard.

Projects

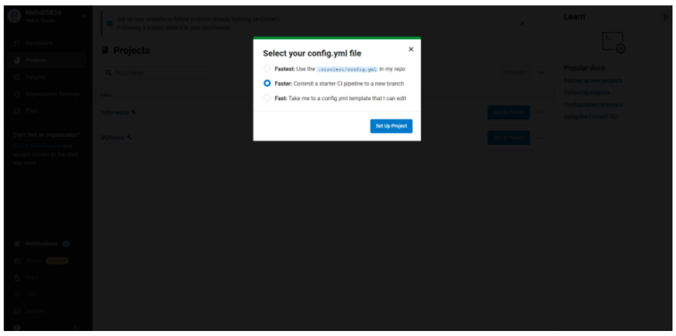
Repo	
hello-world	Set Up Project
MyRepos	Set Up Project

Can't find an organization? Check permissions and update access to the ones you want.

Popular docs: Setting up new projects, Following projects, Configuration reference, Using the CircleCI CLI

Step 2 - 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.


Select your config.yml file

Fastest: Use the `.circleci/config.yml` in my repo

[✓ Faster: Commit a starter CI pipeline to a new branch]

Fast: Take me to a config.yml template that I can edit

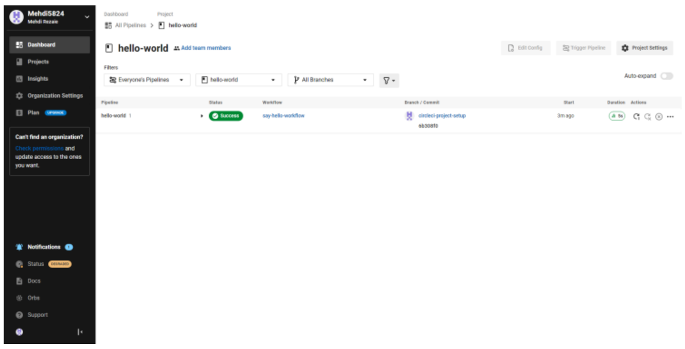
Set Up Project

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-helloworkflow).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

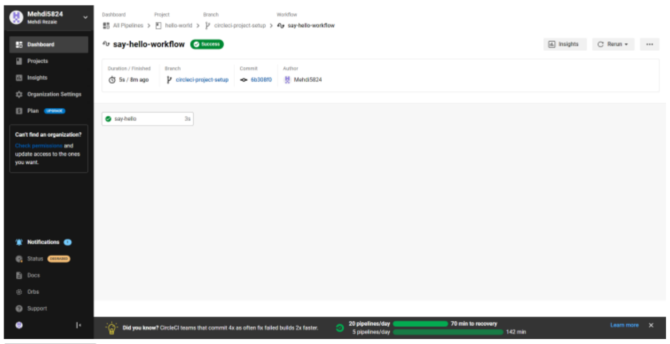

Dashboard > All Pipelines > Project > hello-world

hello-world — Add team members

Filters: Everyone's Pipelines | hello-world | All Branches

Pipeline	Status	Workflow	Branch / Commit	Start	Duration
hello-world 1	Success	say-hello-workflow	circleci-project-setup 6b308f0	3m ago	5s

Select "say-hello" Job with a green tick


All Pipelines > hello-world > circleci-project-setup > say-hello-workflow

say-hello-workflow — Success

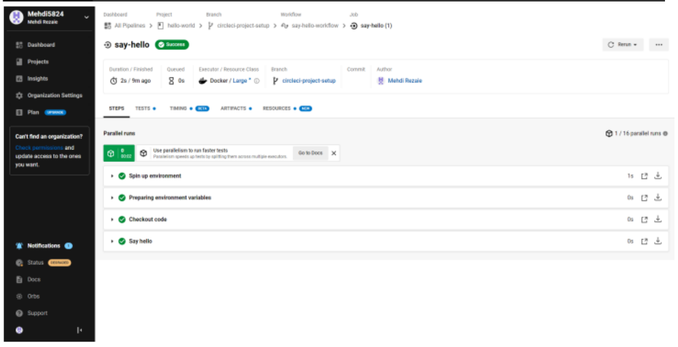
Duration / Finished: 5s / 8m ago | Branch: circleci-project-setup | Commit: 6b308f0 | Author: Mehdi5824

say-hello 3s

Did you know? CircleCI teams that commit 4x as often fix failed builds 2x faster.

20 pipelines/day — 70 min to recovery

5 pipelines/day — 142 min


All Pipelines > hello-world > circleci-project-setup > say-hello-workflow > say-hello (1)

say-hello — Success

Duration / Finished: 2s / 9m ago | Queued: 0s | Executor / Resource Class: Docker / Large | Branch: circleci-project-setup | Author: Mehdi Rezaie

STEPS | TESTS | TIMING | ARTIFACTS | RESOURCES

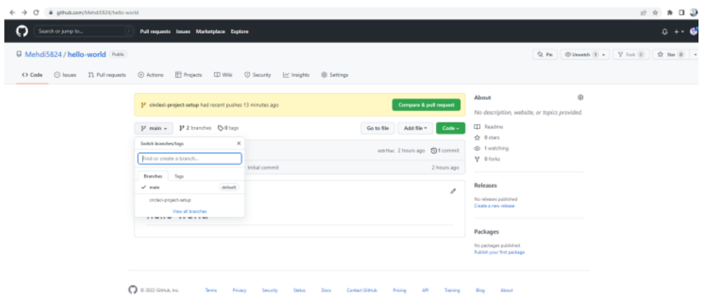
Parallel runs — 1 / 16 parallel runs

Use parallelism to run faster tests

Parallelism speeds up tests by splitting them across multiple executors.

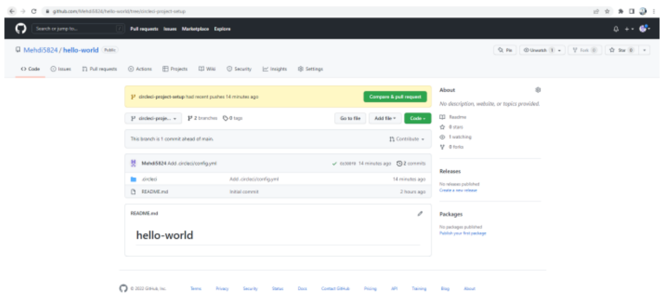
Step	Time
Spin up environment	1s
Preparing environment variables	0s
Checkout code	0s
Say hello	0s

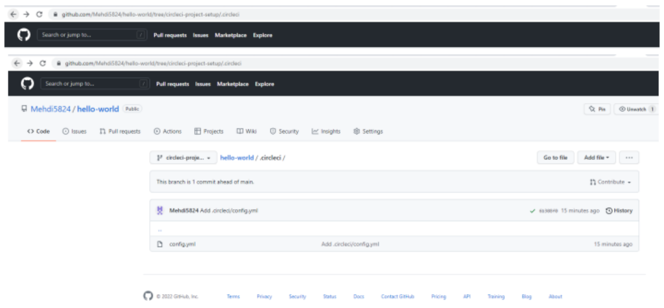
Select Branch and option circleci-project-setup

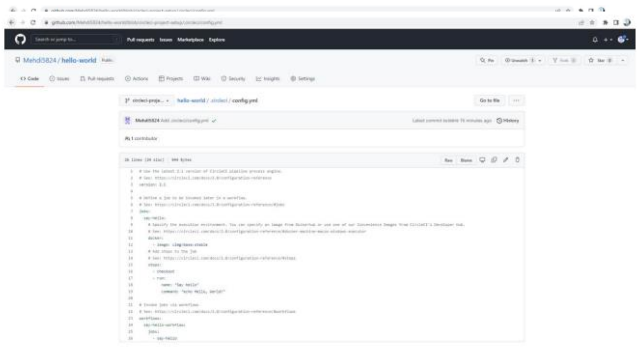


Step 4 - Break your build

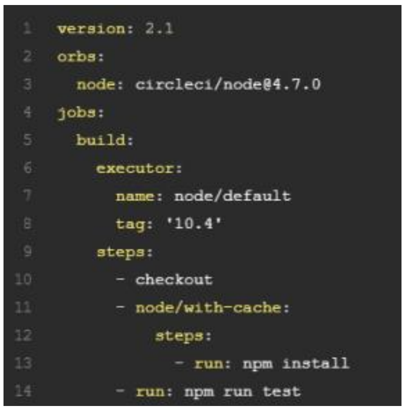
In this section, you will edit the .circleci/config.yml file and see what happens if a build does not complete successfully. It is possible to edit files directly on GitHub.





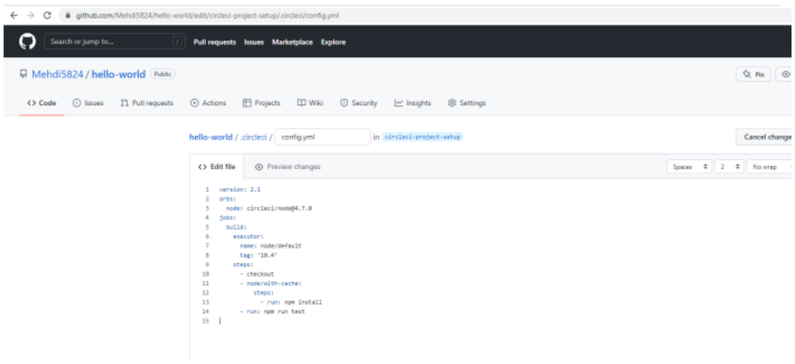


Lets use the node orb. Replace the existing config by pasting the following code.



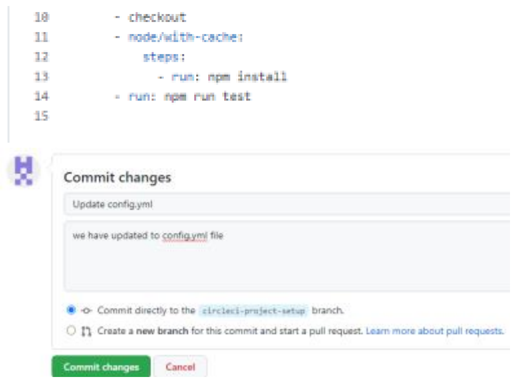
```yaml
version: 2.1
orbs:
  node: circleci/node@4.7.0
jobs:
  build:
    executor:
      name: node/default
      tag: '10.4'
    steps:
      - checkout
      - node/with-cache:
          steps:
            - run: npm install
      - run: npm run test
```

The GitHub file editor should look like this



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

Scroll down and Commit your changes on GitHub



```yaml
      - checkout
      - node/with-cache:
          steps:
            - run: npm install
      - run: npm run test
```

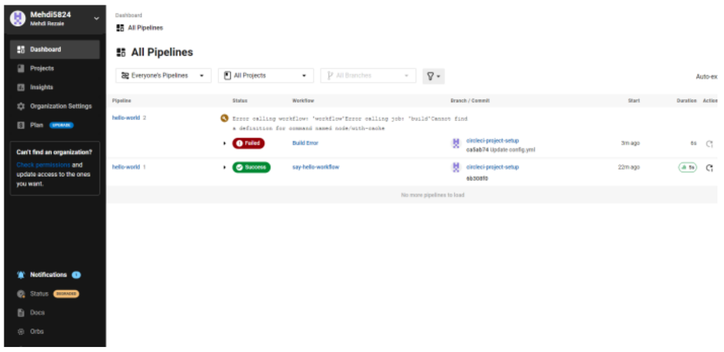
Commit changes

Update config.yml

we have updated to config.yml file

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.
Commit changes
Cancel

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 configuration to fail. To fix this, you need to set up a Node project in your repository.



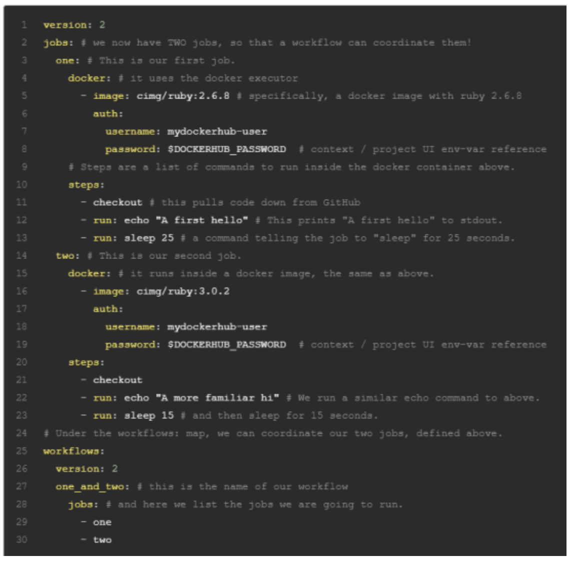
Pipeline	Status	Workflow	Branch / Commit	Start	Duration	Action
hello-world 2	Error calling workflow: 'workflow'Error calling job: 'build'Cannot find a definition for command named node/with-cache					
	Failed	Build Error	circleci-project-setup ca5ab74 Update config.yml	3m ago	6s	
hello-world 1	Success	say-hello-workflow	circleci-project-setup eb30810	22m ago	5s	

No more pipelines to load

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.

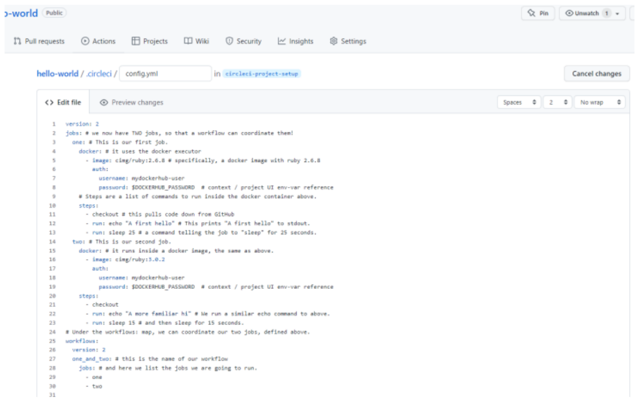
1) Take a moment and read the comments in the code block below. Then, to see workflows in action, edit your .circleci/config.yml file and copy and paste the following text into it



```yaml
version: 2
jobs: # we now have TWO jobs, so that a workflow can coordinate them!
  one: # This is our first job.
    docker: # it uses the docker executor
      - image: cimg/ruby:2.6.8 # specifically, a docker image with ruby 2.6.8
        auth:
          username: mydockerhub-user
          password: $DOCKERHUB_PASSWORD  # context / project UI env-var reference
    # Steps are a list of commands to run inside the docker container above.
    steps:
      - checkout # this pulls code down from GitHub
      - run: echo "A first hello" # This prints "A first hello" to stdout.
      - run: sleep 25 # a command telling the job to "sleep" for 25 seconds.
  two: # This is our second job.
    docker: # it runs inside a docker image, the same as above.
      - 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" # We run a similar echo command to above.
      - run: sleep 15 # and then sleep for 15 seconds.
# Under the workflows: map, we can coordinate our two jobs, defined above.
workflows:
  version: 2
  one_and_two: # this is the name of our workflow
    jobs: # and here we list the jobs we are going to run.
      - one
      - two
```

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

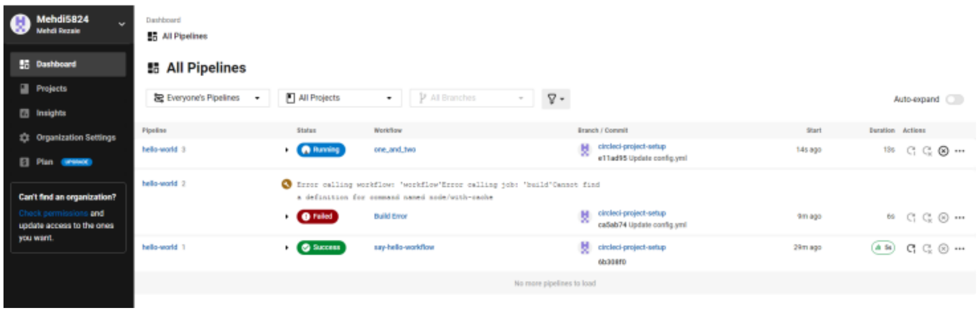
2) Commit these changes to your repository and navigate back to the CircleCI Pipelines page. You should see your pipeline running



hello-world / .circleci / config.yml in circleci-project-setup

```yaml
version: 2
jobs: # we now have TWO jobs, so that a workflow can coordinate them!
  one: # This is our first job.
    docker: # it uses the docker executor
      - image: cimg/ruby:2.6.8 # specifically, a docker image with ruby 2.6.8
        auth:
          username: mydockerhub-user
          password: $DOCKERHUB_PASSWORD  # context / project UI env-var reference
    # Steps are a list of commands to run inside the docker container above.
    steps:
      - checkout # this pulls code down from GitHub
      - run: echo "A first hello" # This prints "A first hello" to stdout.
      - run: sleep 25 # a command telling the job to "sleep" for 25 seconds.
  two: # This is our second job.
    docker: # it runs inside a docker image, the same as above.
      - 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" # We run a similar echo command to above.
      - run: sleep 15 # and then sleep for 15 seconds.
# Under the workflows: map, we can coordinate our two jobs, defined above.
workflows:
  version: 2
  one_and_two: # this is the name of our workflow
    jobs: # and here we list the jobs we are going to run.
      - one
      - two
```

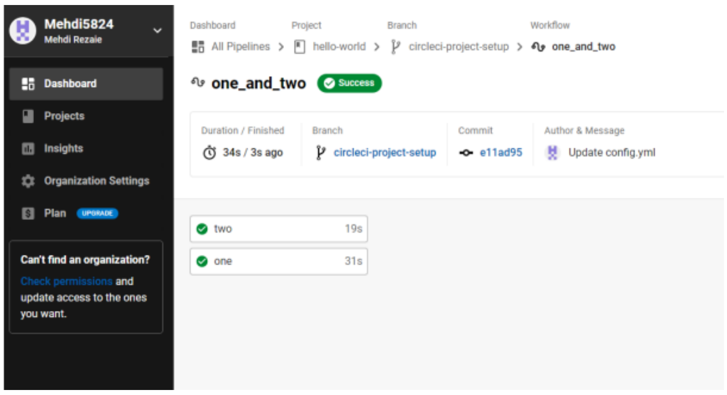
3) Click on the running pipeline to view the workflow you have created. You should see that two jobs ran (or are currently running!) concurrently



All Pipelines

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

No more pipelines to load


Mehdi5824
Mehdi Rezaie

Dashboard
Projects
Insights
Organization Settings
Plan UPGRADE

Can't find an organization?
Check permissions and update access to the ones you want.

Dashboard: All Pipelines > Project: hello-world > Branch: circleci-project-setup > Workflow: one_and_two

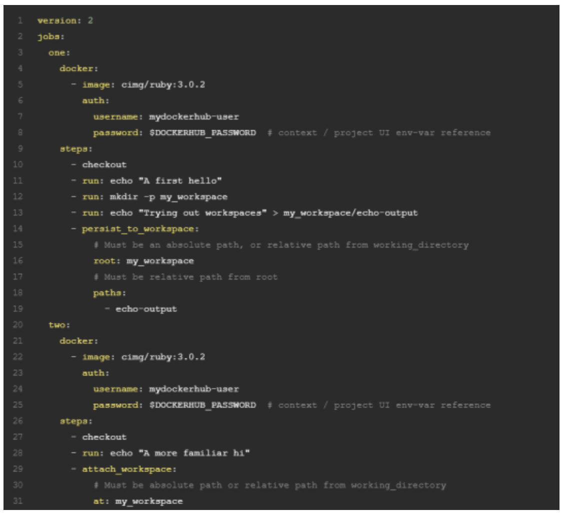
one_and_two Success

Duration / Finished	Branch	Commit	Author & Message
34s / 3s ago	circleci-project-setup	e11ad95	Update config.yml

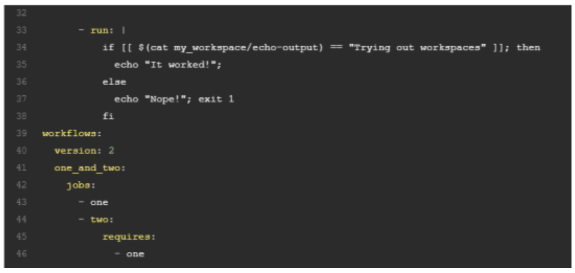
two	19s
one	31s

Step 5 – 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:

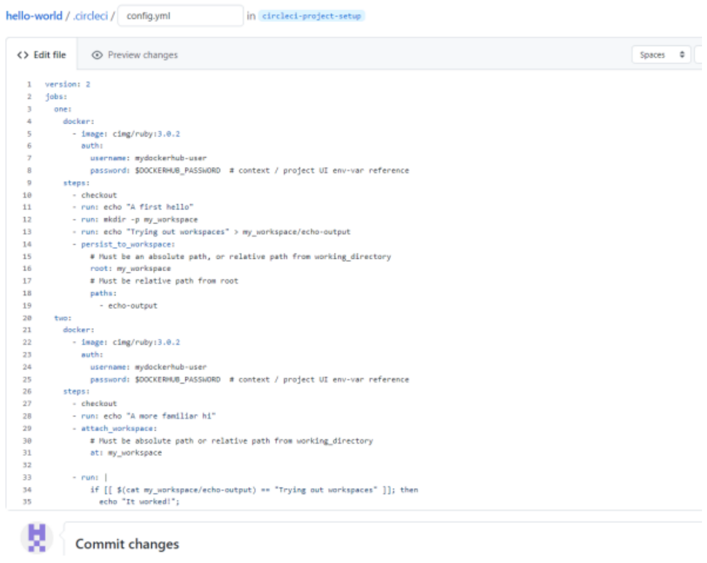


```yaml
version: 2
jobs:
  one:
    docker:
      - image: cimg/ruby:3.0.2
        auth:
          username: mydockerhub-user
          password: $DOCKERHUB_PASSWORD  # context / project UI env-var reference
    steps:
      - checkout
      - run: echo "A first hello"
      - run: mkdir -p my_workspace
      - 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
```



```yaml

      - run: |
          if [[ $(cat my_workspace/echo-output) == "Trying out workspaces" ]]; then
            echo "It worked!";
          else
            echo "Nope!"; exit 1
          fi
workflows:
  version: 2
  one_and_two:
    jobs:
      - one
      - two:
          requires:
            - one
```

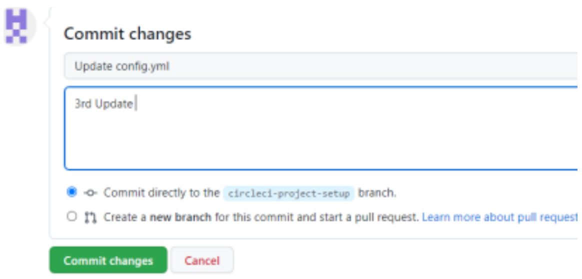
Updated config.yml in GitHub file editor should be updated like this


hello-world / .circleci / config.yml in circleci-project-setup

<> Edit file   Preview changes   Spaces

```yaml
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!";
```

Commit changes



Commit changes

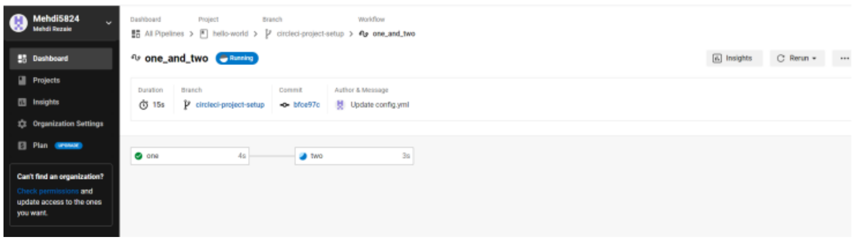
Update config.yml

3rd Update

Commit directly to the `circleci-project-setup` branch.

Create a **new branch** for this commit and start a pull request. Learn more about pull request

Finally your workflow with the jobs running should look like this



Dashboard: All Pipelines > Project: hello-world > Branch: circleci-project-setup > Workflow: one_and_two

one_and_two Running

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

one 4s → two 3s

Can't find an organization? Check permissions and update access to the ones you want.

Practical 6
Working with TeamService

(Install .Net core sdk first)

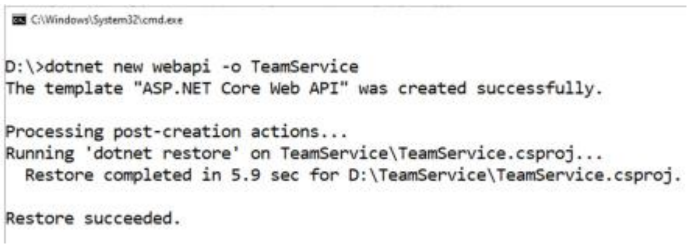
Link: https://dotnet.microsoft.com/learn/dotnet/hello-world-tutorial/install

1) Create new project:

Command :

```
dotnet new webapi -o TeamService
```

output:



```
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.
```

2) Remove existing weatherforecast files both model and controller files.
3) Add new files as follows:
4) 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;
 }
 }
}
```

5) 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;
 }
 }
}
```

6) 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) // I HATE NULLS, MUST FIXERATE THIS.
 { 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);
 }
 }
 }
}
```

7) 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());
 }
 }
 }
 [HttpPut]
 [Route("/teams/{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]/{teamMember.Me
mberID}", 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;
 }
 }
}
```

8) create folder "D:\TeamService\Persistence":

9) 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);
}
}
```

10) 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;
 }
 }
}
```

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

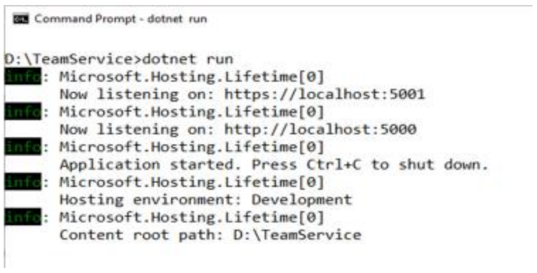
12) Now open two command prompts to run this project

13) On Command prompt 1: (go inside folder teamservice first)

Commands:

dotnet run

Output:



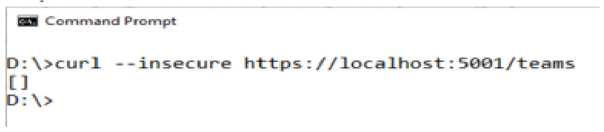
```
Command Prompt - dotnet run

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
```

14) On command prompt 2

Command: To get all teams

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

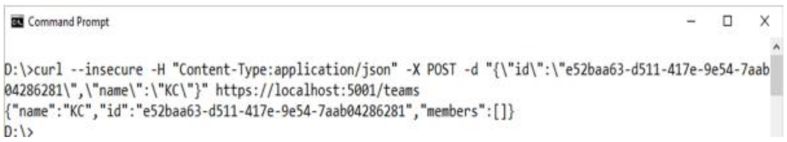
output:



```
Command Prompt

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

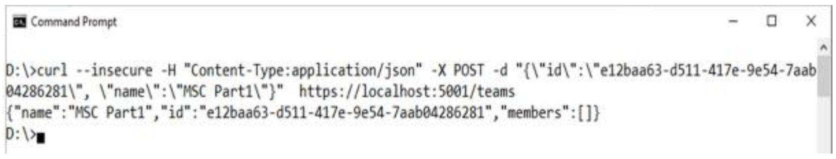
Command : To create new team

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



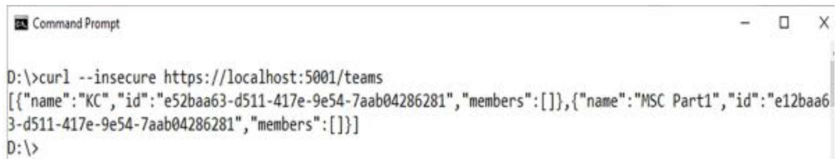
```
D:\>curl --insecure -H "Content-Type:application/json" -X POST -d "{\"id\":\"e52baa63-d511-417e-9e54-7aab
04286281\",\"name\":\"KC\"}" https://localhost:5001/teams
{"name":"KC","id":"e52baa63-d511-417e-9e54-7aab04286281","members":[]}
D:\>
```

Command : 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
output:



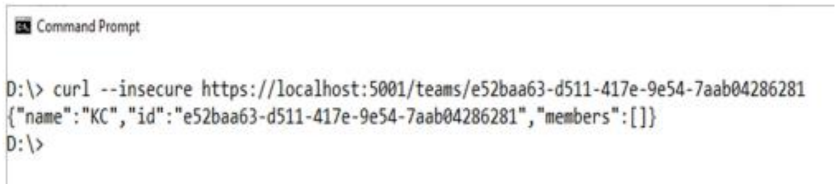
```
D:\>curl --insecure -H "Content-Type:application/json" -X POST -d "{\"id\":\"e12baa63-d511-417e-9e54-7aab
04286281\", \"name\":\"MSC Part1\"}"  https://localhost:5001/teams
{"name":"MSC Part1","id":"e12baa63-d511-417e-9e54-7aab04286281","members":[]}
D:\>
```

Command : To get all teams
curl --insecure https://localhost:5001/teams
Output:



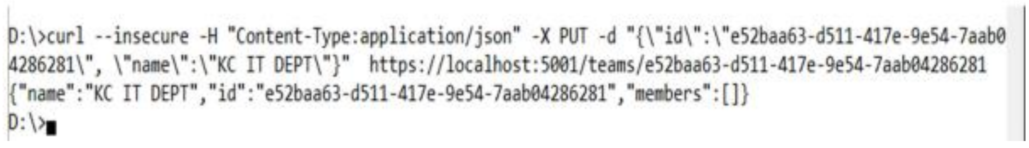
```
D:\>curl --insecure https://localhost:5001/teams
[{"name":"KC","id":"e52baa63-d511-417e-9e54-7aab04286281","members":[]},{"name":"MSC Part1","id":"e12baa6
3-d511-417e-9e54-7aab04286281","members":[]}]
D:\>
```

Command : to get single team with team-id as parameter
curl --insecure https://localhost:5001/teams/e52baa63-d511-417e-9e54-7aab04286281
output:



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

Command : 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
output:

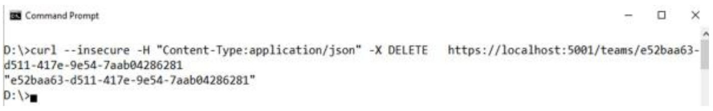


```
D:\>curl --insecure -H "Content-Type:application/json" -X PUT -d "{\"id\":\"e52baa63-d511-417e-9e54-7aab0
4286281\", \"name\":\"KC IT DEPT\"}"  https://localhost:5001/teams/e52baa63-d511-417e-9e54-7aab04286281
{"name":"KC IT DEPT","id":"e52baa63-d511-417e-9e54-7aab04286281","members":[]}
D:\>
```

Command: to delete team

curl --insecure -H “Content-Type:application/json” –X DELETE https://localhost:5001/teams/e52baa63-d511-417e9e54-7aab04286281

output:



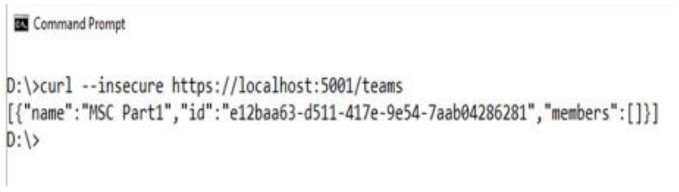
```
D:\>curl --insecure -H "Content-Type:application/json" -X DELETE   https://localhost:5001/teams/e52baa63-
d511-417e-9e54-7aab04286281
"e52baa63-d511-417e-9e54-7aab04286281"
D:\>
```

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

Command:

curl –insecure https://localhost:5001/teams

Output:



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

Practical No. 7

Building real-time microservices with ASP.NET.

✰ **Microservices using ASP.NET Core:**

In this growing fast-paced world, the amount of data and internet usage are proportionally increasing, and so more reliable and fast responding software systems are required, Unlike the older way of application development in Monolithic architecture which causes high maintenance cost, more downtime during upgrades made to existing monolithic architected software is not reliable. So, the Microservices Architecture of developing applications came into the picture.

Earlier software architecture build contains all business functionalities, Database calls, and UI designed in a single bundle. Like Asp.Net Webforms, MVC as a collection of single projects. It has its disadvantages, the larger the application grows, the harder it is to quickly resolve the technical bugs/problems and to update the app with the new features. The Microservice architecture-based approach for building applications helps solve these real-time issues and provides more space for agile development methods and faster response from applications.

✰ **What are Microservices?**

Microservices are the architectural approach to build applications from small to large scale applications. With this architectural approach, an application is broken down into the smallest components, independent of each other. Unlike Monolithic architecture, where all the functionalities are targeted to build into a single project/application, Microservices helps to separate functionalities to develop in a more modular way and all modules work together to accomplish the specific targeted tasks.

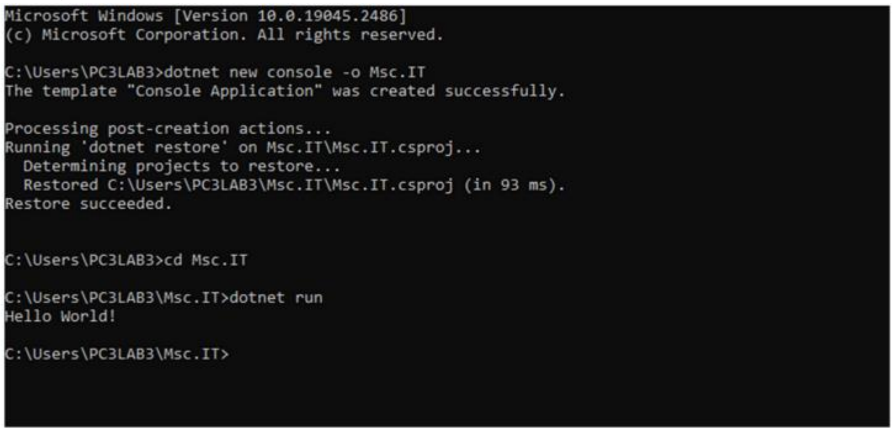
✰ **Need of a Microservice:**

This architectural approach to developing software gives more modularity, being lightweight, and the ability to share similar functionalities across multiple applications. It is a major way of designing and optimizing app development towards a cloud-native model.

Program.cs:

```
using System;

namespace Sohail
{
    class Program
    {
        static void Main(string[] args)
        {
            Console.WriteLine("Hello World!");
        }
    }
}
```

Output:



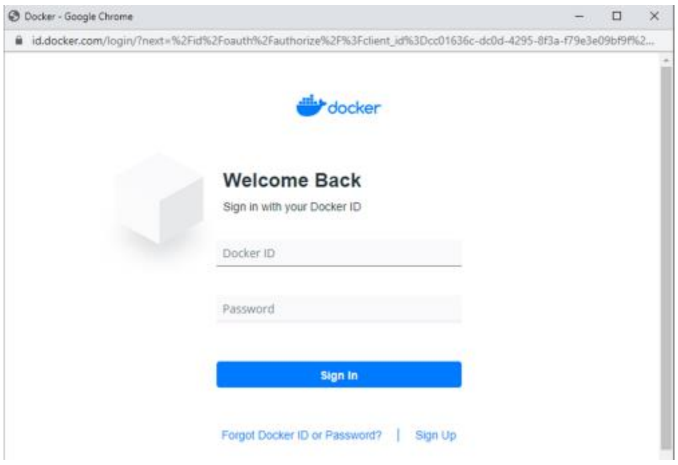
```
Microsoft Windows [Version 10.0.19045.2486]
(c) Microsoft Corporation. All rights reserved.

C:\Users\PC3LAB3>dotnet new console -o Msc.IT
The template "Console Application" was created successfully.

Processing post-creation actions...
Running 'dotnet restore' on Msc.IT\Msc.IT.csproj...
  Determining projects to restore...
  Restored C:\Users\PC3LAB3\Msc.IT\Msc.IT.csproj (in 93 ms).
Restore succeeded.


C:\Users\PC3LAB3>cd Msc.IT

C:\Users\PC3LAB3\Msc.IT>dotnet run
Hello World!

C:\Users\PC3LAB3\Msc.IT>
```

Practical 9

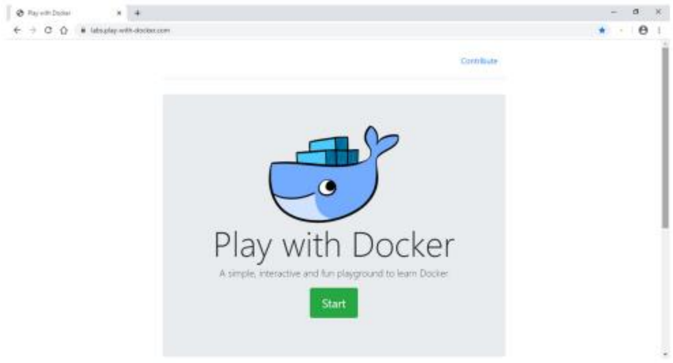
Backing Services

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

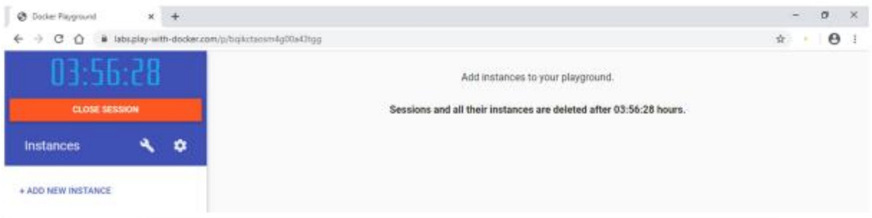
Now login in to Play-With-Docker

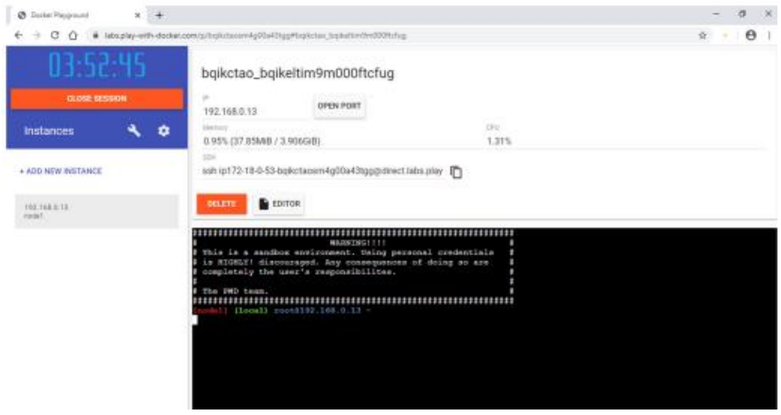

Docker - Google Chrome
id.docker.com/login/?next=%2Fid%2Foauth%2Fauthorize%2F%3Fclient_id%3Dcc01636c-dc0d-4295-8f3a-f79e3e09bf9f%2...
docker
Welcome Back
Sign in with your Docker ID
Docker ID
Password
Sign In
Forgot Docker ID or Password?
Sign Up

Click on Start



Click on Add New Instance


Docker Playground
03:56:28
CLOSE SESSION
Instances
+ ADD NEW INSTANCE
Add instances to your playground.
Sessions and all their instances are deleted after 03:56:28 hours.



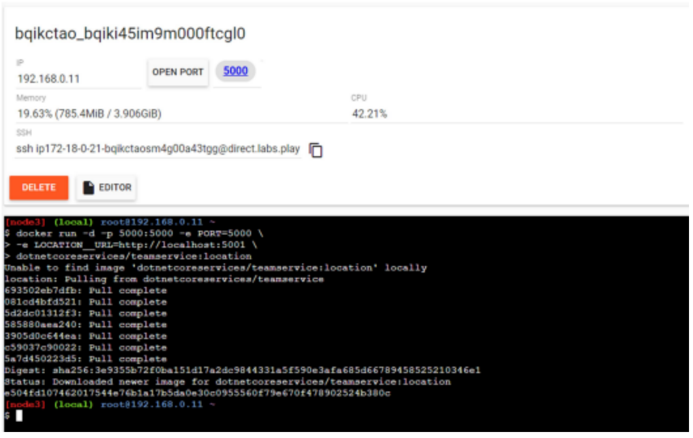
```
###############################################################
#                          WARNING!!!!                        #
# This is a sandbox environment. Using personal credentials   #
# is HIGHLY! discouraged. Any consequences of doing so are    #
# completely the user's responsibilites.                      #
#                                                             #
# The PWD team.                                               #
###############################################################
[node1] (local) root@192.168.0.13 ~
```

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)


bqikctao_bqiki45im9m000ftcgl0

IP
192.168.0.11

Memory
19.63% (785.4MiB / 3.906GiB)

CPU
42.21%

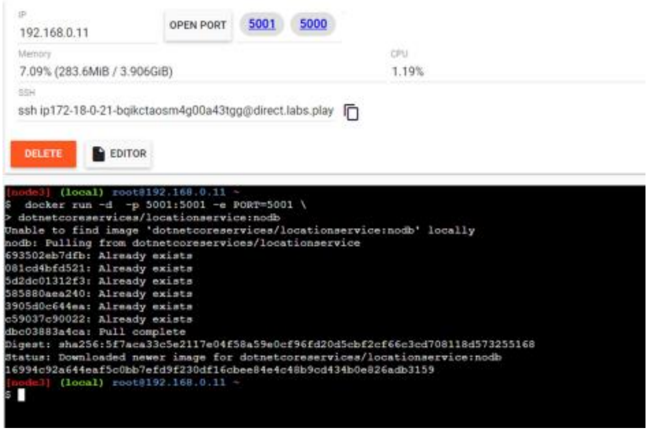
SSH
ssh ip172-18-0-21-bqikctaosm4g00a43tgg@direct.labs.play

```
[node3] (local) root@192.168.0.11 ~
$ docker run -d -p 5000:5000 -e PORT=5000 \
> -e LOCATION__URL=http://localhost:5001 \
> dotnetcoreservices/teamservice:location
Unable to find image 'dotnetcoreservices/teamservice:location' locally
location: Pulling from dotnetcoreservices/teamservice
693502eb7dfb: Pull complete
081cd4bfd521: Pull complete
5d2dc01312f3: Pull complete
585880aea240: Pull complete
3905d0c644ea: Pull complete
c59037c90022: Pull complete
5a7d450223d5: Pull complete
Digest: sha256:3e9355b72f0ba151d17a2dc9844331a5f590e3afa685d66789458525210346e1
Status: Downloaded newer image for dotnetcoreservices/teamservice:location
e504fd107462017544e76b1a17b5da0e30c0955560f79e670f478902524b380c
[node3] (local) root@192.168.0.11 ~
$ 
```

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)


IP
192.168.0.11

Memory
7.09% (283.6MiB / 3.906GiB)

CPU
1.19%

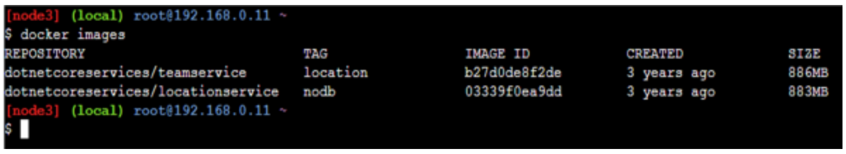
SSH
ssh ip172-18-0-21-bqikctaosm4g00a43tgg@direct.labs.play

```
[node3] (local) root@192.168.0.11 ~
$  docker run -d  -p 5001:5001 -e PORT=5001 \
> dotnetcoreservices/locationservice:nodb
Unable to find image 'dotnetcoreservices/locationservice:nodb' locally
nodb: Pulling from dotnetcoreservices/locationservice
693502eb7dfb: Already exists
081cd4bfd521: Already exists
5d2dc01312f3: Already exists
585880aea240: Already exists
3905d0c644ea: Already exists
c59037c90022: Already exists
dbc03883a4ca: Pull complete
Digest: sha256:5f7aca33c5e2117e04f58a59e0cf96fd20d5cbf2cf66c3cd708118d573255168
Status: Downloaded newer image for dotnetcoreservices/locationservice:nodb
16994c92a644eaf5c0bb7efd9f230df16cbee84e4c48b9cd434b0e826adb3159
[node3] (local) root@192.168.0.11 ~
$ 
```

Command : to check running images in docker

docker images

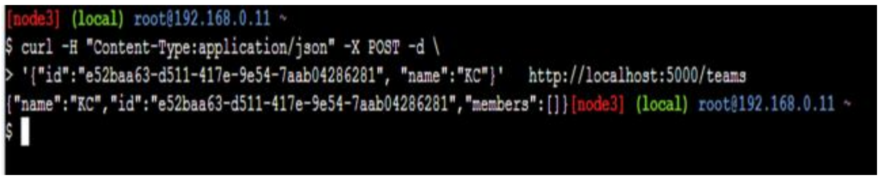
output:


```
[node3] (local) root@192.168.0.11 ~
$ docker images
REPOSITORY                          TAG                 IMAGE ID            CREATED             SIZE
dotnetcoreservices/teamservice      location            b27d0de8f2de        3 years ago         886MB
dotnetcoreservices/locationservice  nodb                03339f0ea9dd        3 years ago         883MB
[node3] (local) root@192.168.0.11 ~
$ 
```

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:

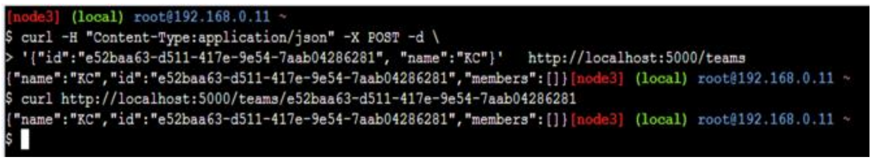


```
[node3] (local) root@192.168.0.11 ~
$ curl -H "Content-Type:application/json" -X POST -d \
> '{"id":"e52baa63-d511-417e-9e54-7aab04286281", "name":"KC"}'   http://localhost:5000/teams
{"name":"KC","id":"e52baa63-d511-417e-9e54-7aab04286281","members":[]}[node3] (local) root@192.168.0.11 ~
$ 
```

Command :To confirm that team is added

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

Output

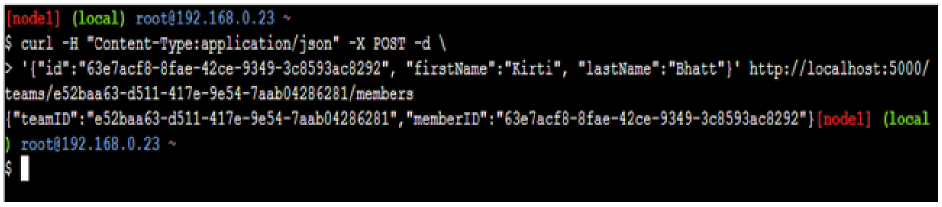


```
[node3] (local) root@192.168.0.11 ~
$ curl -H "Content-Type:application/json" -X POST -d \
> '{"id":"e52baa63-d511-417e-9e54-7aab04286281", "name":"KC"}'   http://localhost:5000/teams
{"name":"KC","id":"e52baa63-d511-417e-9e54-7aab04286281","members":[]}[node3] (local) root@192.168.0.11 ~
$ curl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

Output:

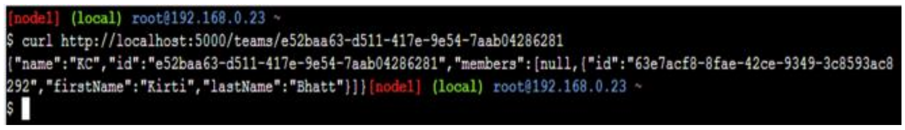


```
[node1] (local) root@192.168.0.23 ~
$ curl -H "Content-Type:application/json" -X POST -d \
> '{"id":"63e7acf8-8fae-42ce-9349-3c8593ac8292", "firstName":"Kirti", "lastName":"Bhatt"}' http://localhost:5000/
teams/e52baa63-d511-417e-9e54-7aab04286281/members
{"teamID":"e52baa63-d511-417e-9e54-7aab04286281","memberID":"63e7acf8-8fae-42ce-9349-3c8593ac8292"}[node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

Command :To confirm member added

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

output

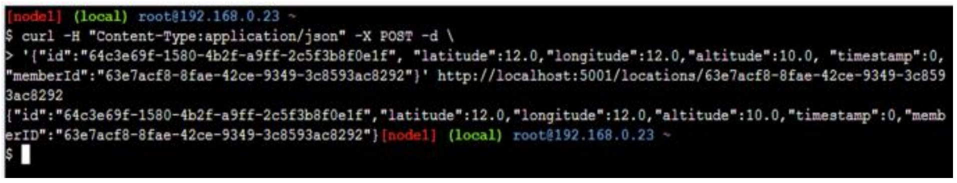


```
[node1] (local) root@192.168.0.23 ~
$ curl http://localhost:5000/teams/e52baa63-d511-417e-9e54-7aab04286281
{"name":"KC","id":"e52baa63-d511-417e-9e54-7aab04286281","members":[null,{"id":"63e7acf8-8fae-42ce-9349-3c8593ac8
292","firstName":"Kirti","lastName":"Bhatt"}]}[node1] (local) root@192.168.0.23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

Command : To add location for member

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

Output:

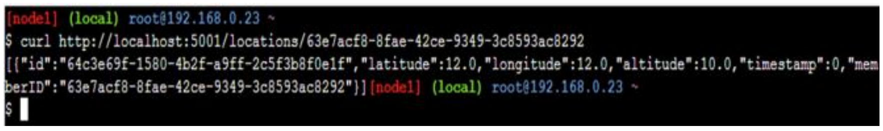


```
[node1] (local) root@192.168.0.23 ~
$ curl -H "Content-Type:application/json" -X POST -d \
> '{"id":"64c3e69f-1580-4b2f-a9ff-2c5f3b8f0e1f", "latitude":12.0,"longitude":12.0,"altitude":10.0, "timestamp":0,
"memberId":"63e7acf8-8fae-42ce-9349-3c8593ac8292"}' http://localhost:5001/locations/63e7acf8-8fae-42ce-9349-3c859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

Command : To confirm location is added in member

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

output:



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