DOCKER

DOCKER COMMANDS

docker –version : To see the docker version

Docker: To view the list of docker commands

docker images: Displays the list of images

docker pull hello-world: To pull image

docker run -p 8000:80 hello-world:To run the image

docker ps: To get the ID of container

docker stop feb5d9fea6a5: To stop container

docker ps –a: To show stopped container and existing container

docker rm bc3fa9b9a2f0:To remove the stopped container

CREATING A MULTICONTAINER APP WITH DOCKER COMPOSE

Webapi

program.cs

```
using Microsoft.Extensions.Caching.Distributed;
using Microsoft.Extensions.Caching.StackExchangeRedis;
var builder = WebApplication.CreateBuilder(args);

// Add services to the container.

builder.Services.AddControllers();
// Learn more about configuring Swagger/OpenAPI at
https://aka.ms/aspnetcore/swashbuckle
```

```
builder.Services.AddEndpointsApiExplorer();
builder.Services.AddSwaggerGen();
builder.Services.AddStackExchangeRedisCache(options =>
{
    options.Configuration = "redis:6379"; // redis is the container name of the
redis service. 6379 is the default port
    options.InstanceName = "SampleInstance";
});
var app = builder.Build();

// Configure the HTTP request pipeline.
if (app.Environment.IsDevelopment())
{
    app.UseSwagger();
    app.UseSwaggerUI();
}

app.UseHttpsRedirection();

app.UseAuthorization();

app.MapControllers();

app.Run();
```

CounterController.cs

```
using Microsoft.AspNetCore.Mvc;
using Microsoft.Extensions.Caching.Distributed;
using StackExchange.Redis;

namespace WebApi.Controllers
{
    [ApiController]
    [Route("[controller]")]
    public class CounterController : ControllerBase
    {
        private readonly ILogger<CounterController> _logger;
        private readonly IDistributedCache _cache;

        public CounterController(ILogger<CounterController> logger,
IDistributedCache cache)
        {
            _logger = logger;
            _cache = cache;
        }

        [HttpGet(Name = "GetCounter")]
        public string Get()
        {
            string key = "Counter";
            string? result = null;
            try
            {
                var counterStr = _cache.GetString(key);
                if (int.TryParse(counterStr, out int counter))
                {
                    counter++;
                }
```

```
                else
                {
                    counter = 0;
                }
                result = counter.ToString();
                _cache.SetString(key, result);
            }
            catch (RedisConnectionException)
            {
                result = $"Redis cache is not found.";
            }
            return result;
        }
    }
}
```

Webapp

Index.cshtml.cs

```
using Microsoft.AspNetCore.Mvc;
using Microsoft.AspNetCore.Mvc.RazorPages;

namespace dockercompose.Pages
{
    public class IndexModel : PageModel
    {
        private readonly ILogger<IndexModel> _logger;

        public IndexModel(ILogger<IndexModel> logger)
        {
            _logger = logger;
        }

        public async Task OnGet()
        {
            using (var client = new System.Net.Http.HttpClient())
            {
                // Call *mywebapi*, and display its response in the page
                var request = new System.Net.Http.HttpRequestMessage();
                // webapi is the container name
                request.RequestUri = new Uri("http://webapi/Counter");
                var response = await client.SendAsync(request);
                string counter = await response.Content.ReadAsStringAsync();
                ViewData["Message"] = $"Counter value from cache :{counter}";
            }
        }
    }
}
```

Index.cshtml

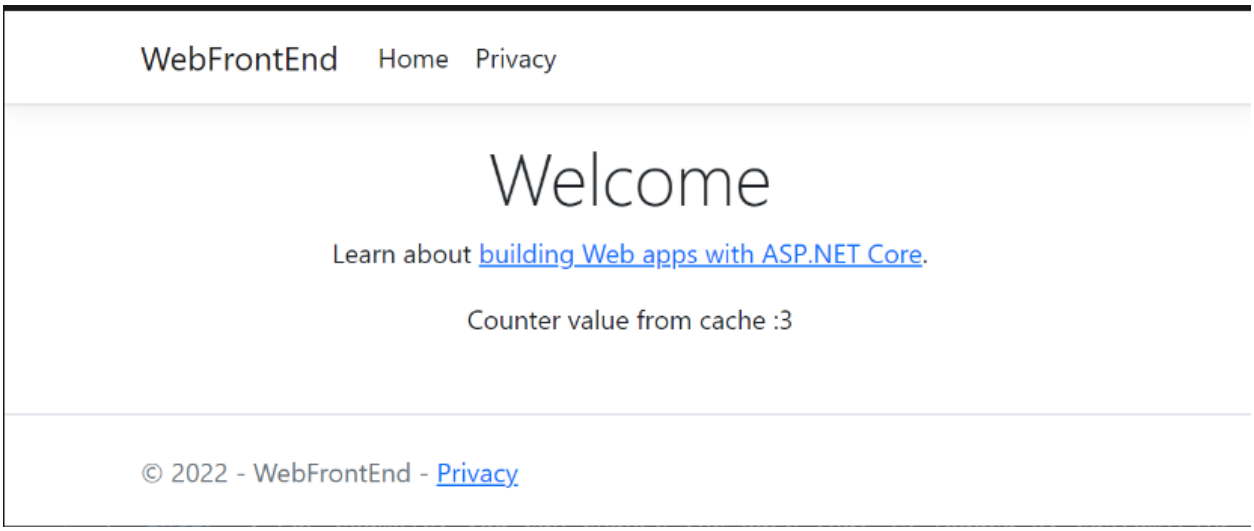
```
@page
@model IndexModel
@{
    ViewData["Title"] = "Home page";
}

<div class="text-center">
    <h1 class="display-4">Welcome</h1>
```

```
    <p>Learn about <a href="https://docs.microsoft.com/aspnet/core">building Web
apps with ASP.NET Core</a>.</p>
    <p>@ViewData["Message"]</p>
</div>
```

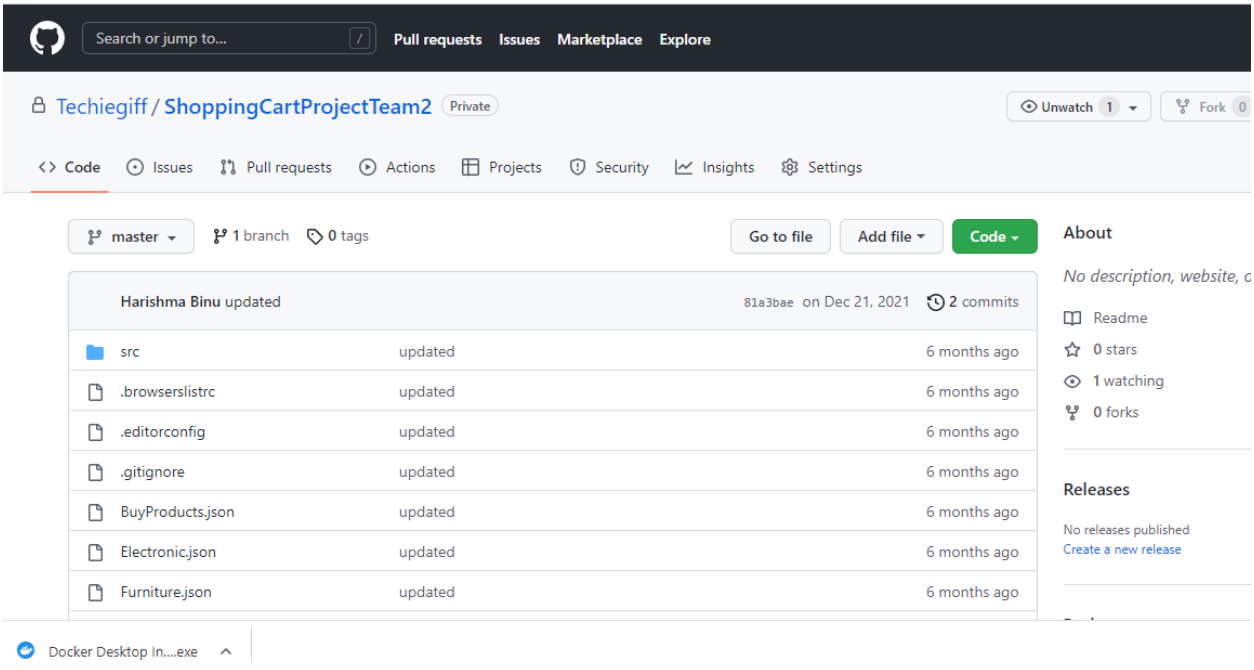
Dockercompose

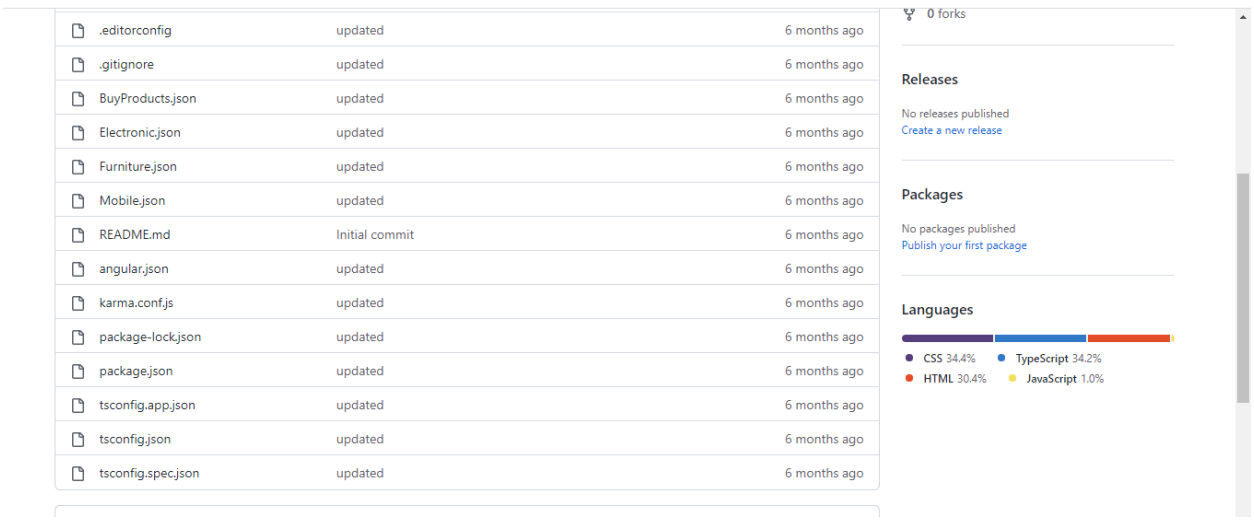
Docker-compose.yml

```
version: '3.4'

services:
  dockercompose:
    image: ${DOCKER_REGISTRY-}dockercompose
    build:
      context: .
      dockerfile: dockercompose/Dockerfile

  webapi:
    image: ${DOCKER_REGISTRY-}webapi
    build:
      context: .
      dockerfile: webapi/Dockerfile

  redis:
     image: redis
```



CI/CD PIPELINE FOR DOCKER



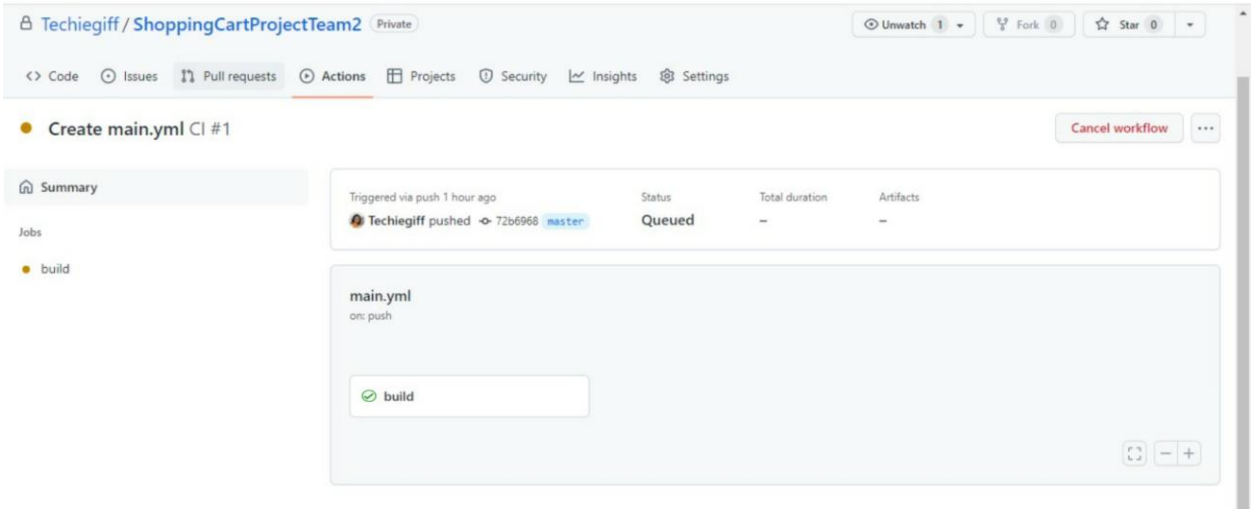


Register.yml

```
name: CI


# Controls when the workflow will run

on:

  # Triggers the workflow on push or pull
request events but only for the "master"
branch

  push:

    branches: [ "master" ]

  pull_request:

    branches: [ "master" ]



  # Allows you to run this workflow
manually from the Actions tab

  workflow_dispatch:
```

```
# A workflow run is made up of one or
more jobs that can run sequentially or in
parallel

jobs:

  # This workflow contains a single job
called "build"

  build:

    # The type of runner that the job
will run on

    runs-on: ubuntu-latest



    # Steps represent a sequence of tasks
that will be executed as part of the job

    steps:

      # Checks-out your repository under
$GITHUB_WORKSPACE, so your job can access
it

      - uses: actions/checkout@v3
```

```
      # Runs a single command using the
runners shell

      - name: Run a one-line script

        run: echo Hello, world!



      # Runs a set of commands using the
runners shell

      - name: Run a multi-line script

        run: |

          echo Add other actions to
build,

          echo test, and deploy your
project.
```


Techiegiff / ShoppingCartProjectTeam2
Private
Unwatch 1
Fork 0
Star 0
Code
Issues
Pull requests
Actions
Projects
Security
Insights
Settings
Create main.yml CI #1
Cancel workflow
Summary
Jobs
build
Triggered via push 1 hour ago
Techiegiff pushed 72b6968 master
Status
Queued
Total duration
–
Artifacts
–
main.yml
on: push
build