Practical no 4

Create an Application Gateway Using Ocelot and Securing APIs with Azure AD

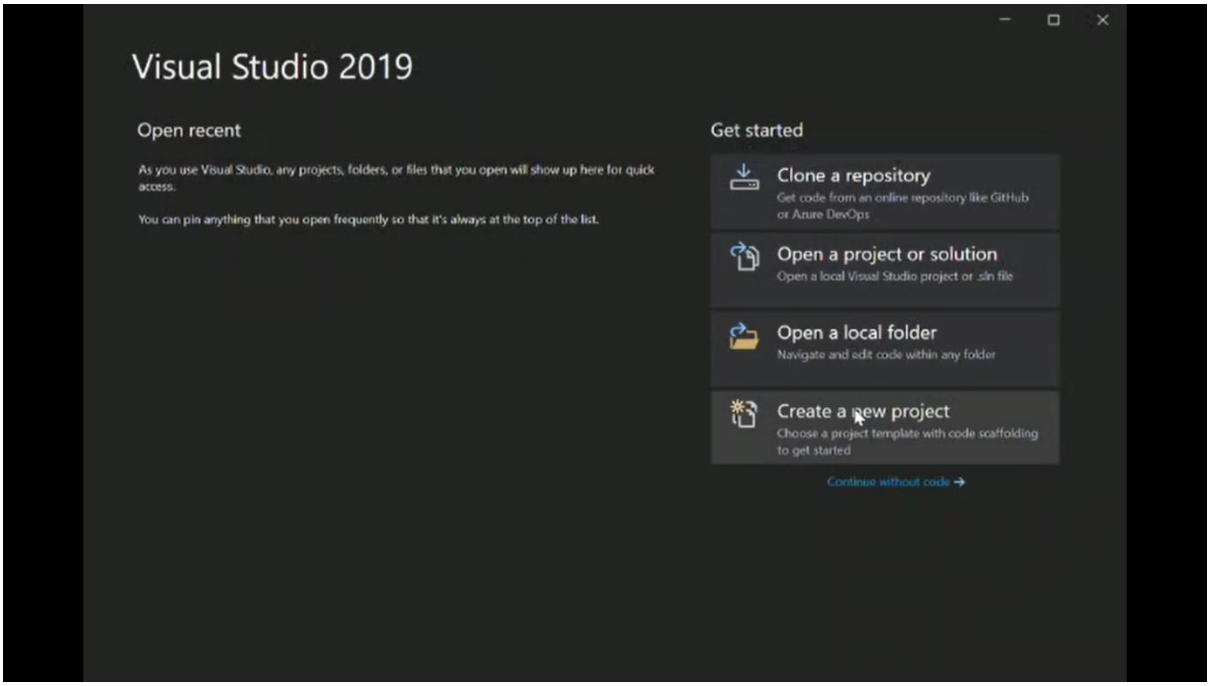
API Gateway is an API management tool that usually sits between the external caller (Web or Mobile) and the internal services.
The API Gateway can provide multiple features like:

1. Routing Request
2. Aggregations
3. Authentication
4. Authorization
5. Rate Limiting
6. Caching
7. Load Balancing ETC.

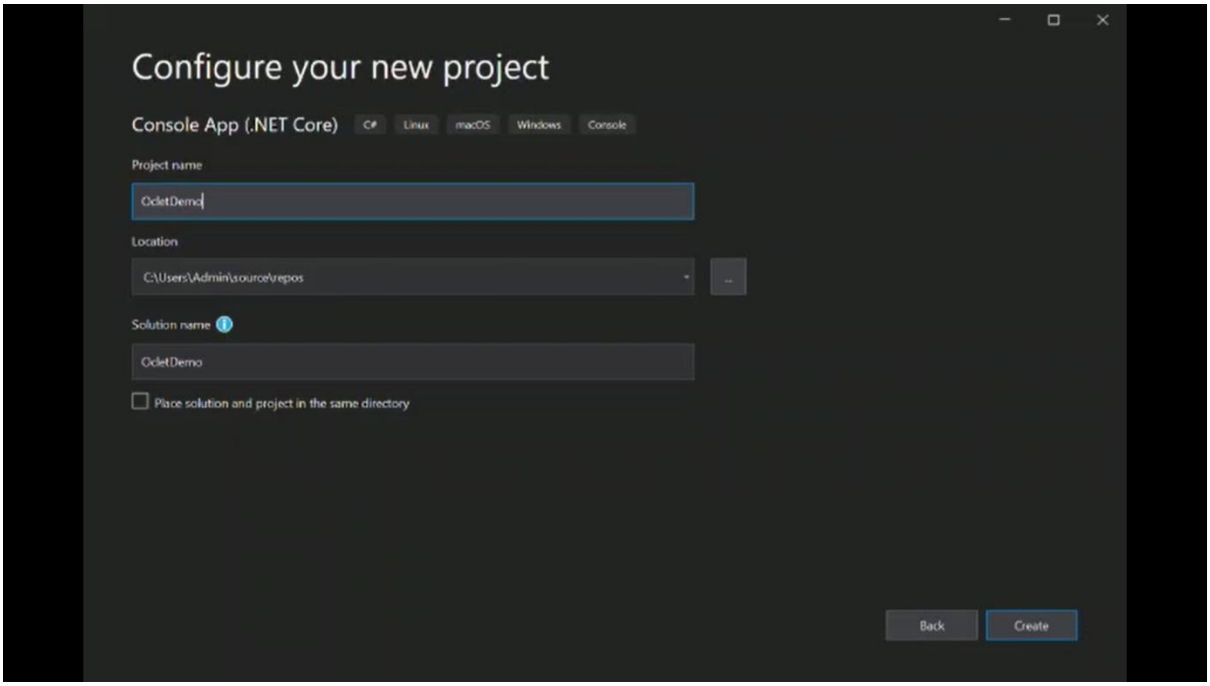
Ocelot is an ASP.Net Core (Supports .Net Core 3.1) API Gateway. It's a NuGet package, which can be added to any ASP.Net Core application to make it an API Gateway. Ocelot API Gateway supports all the features that any standard API Gateway does.

Steps:

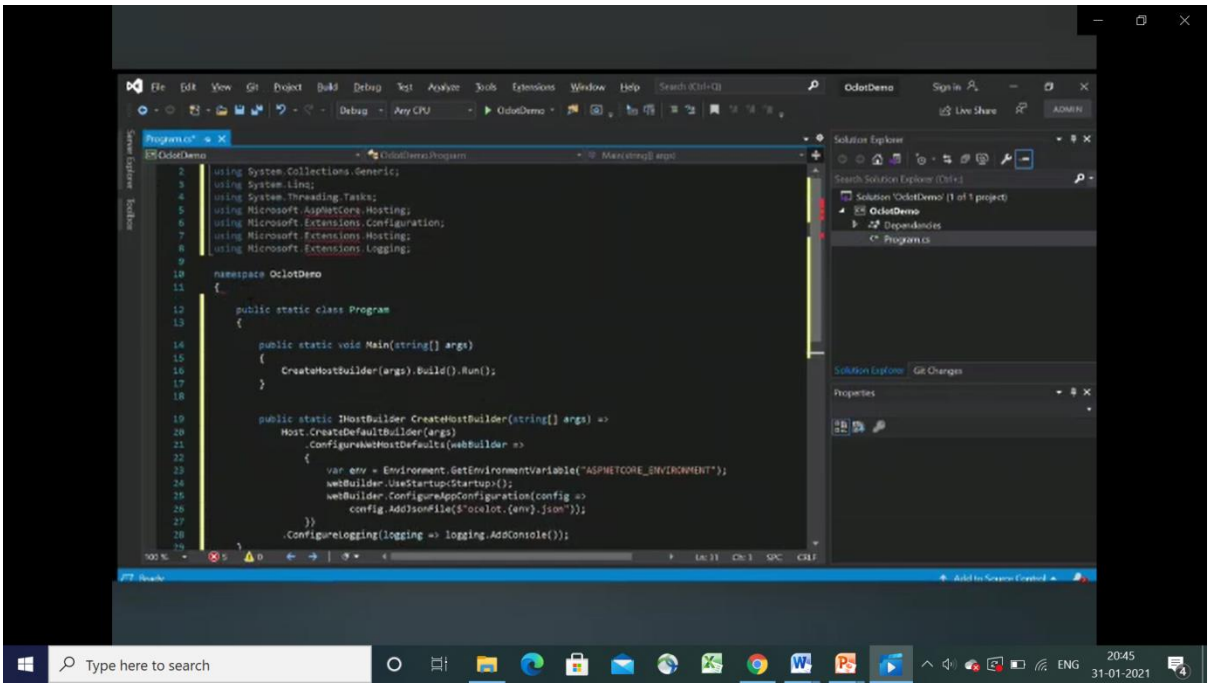
1. Create an ASP.NET Core Web Application Project.



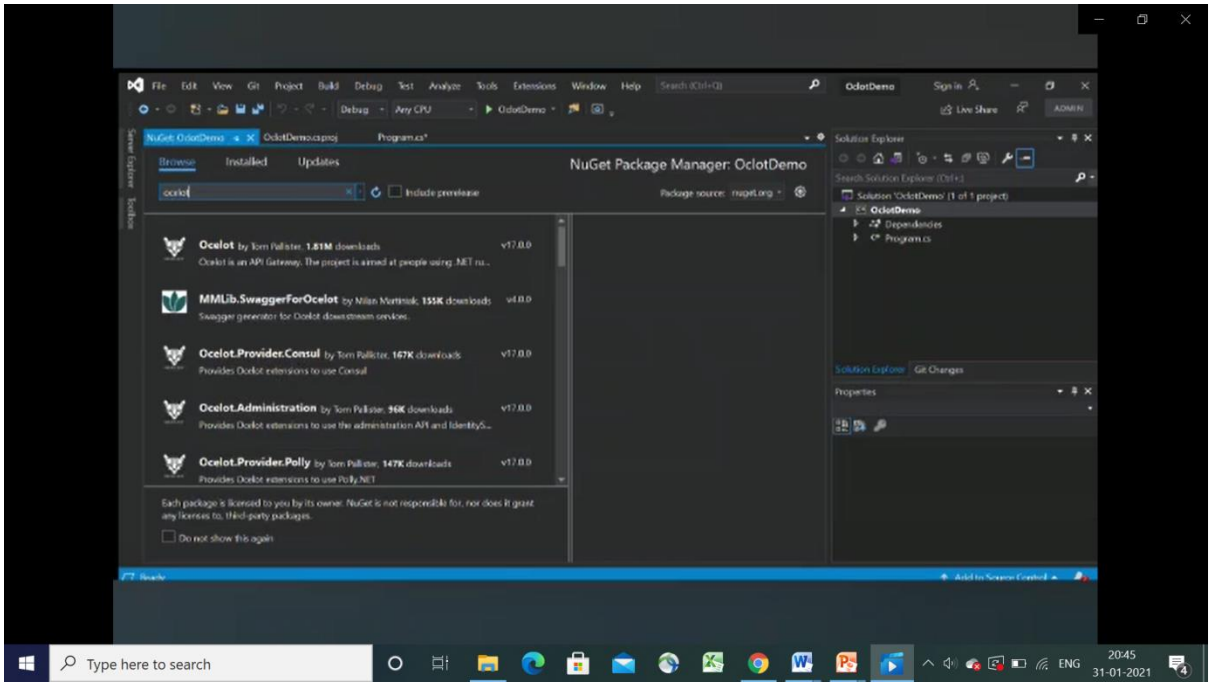
2. Create an empty ASP.NET Core 3.1 and and give a name of the Project.



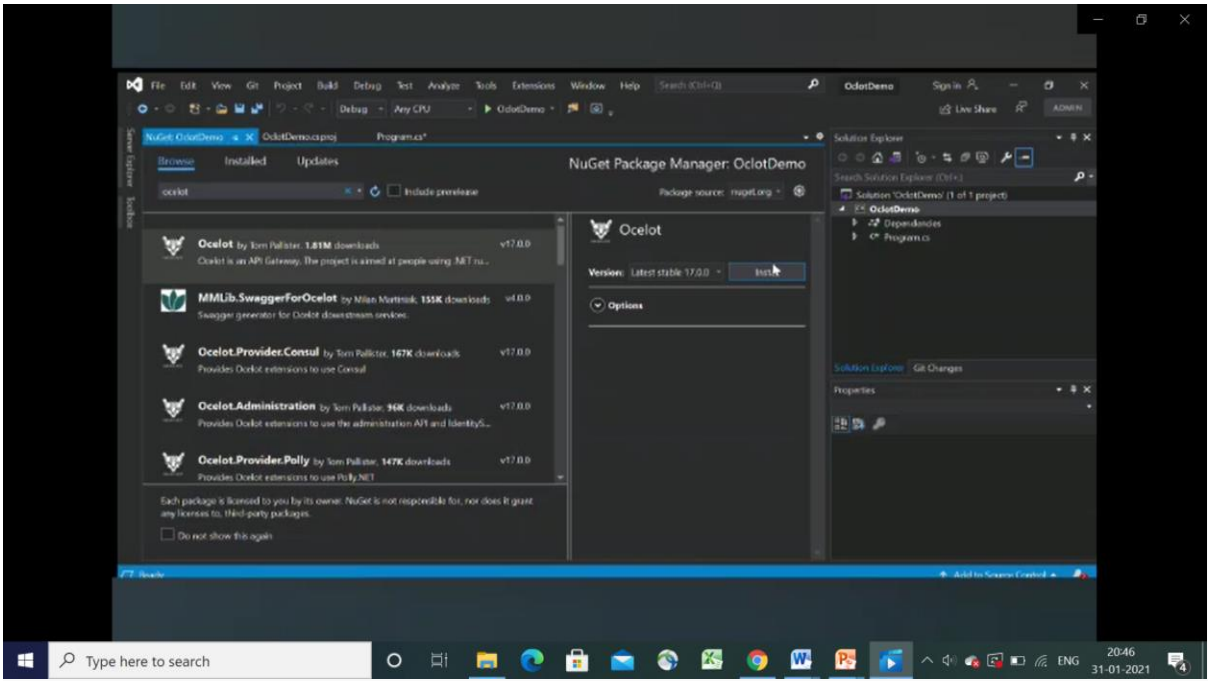
3. Go to Program.cs file and add logging code.



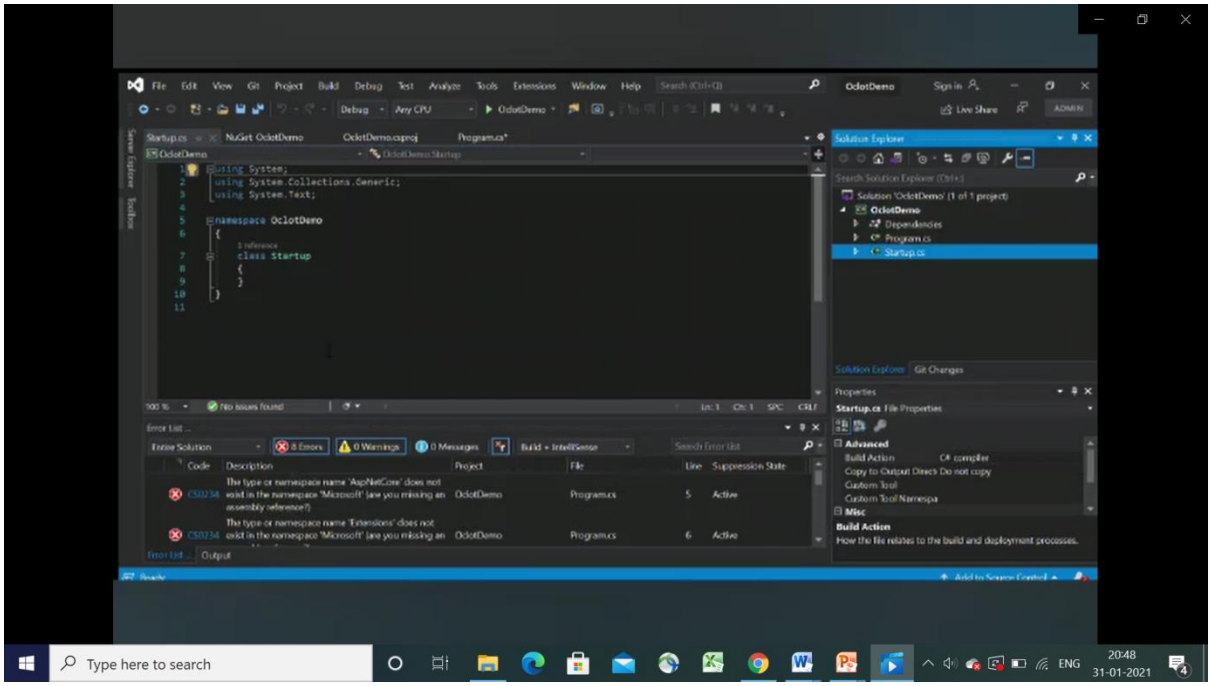
4. Now add new NuGet package for Ocelot



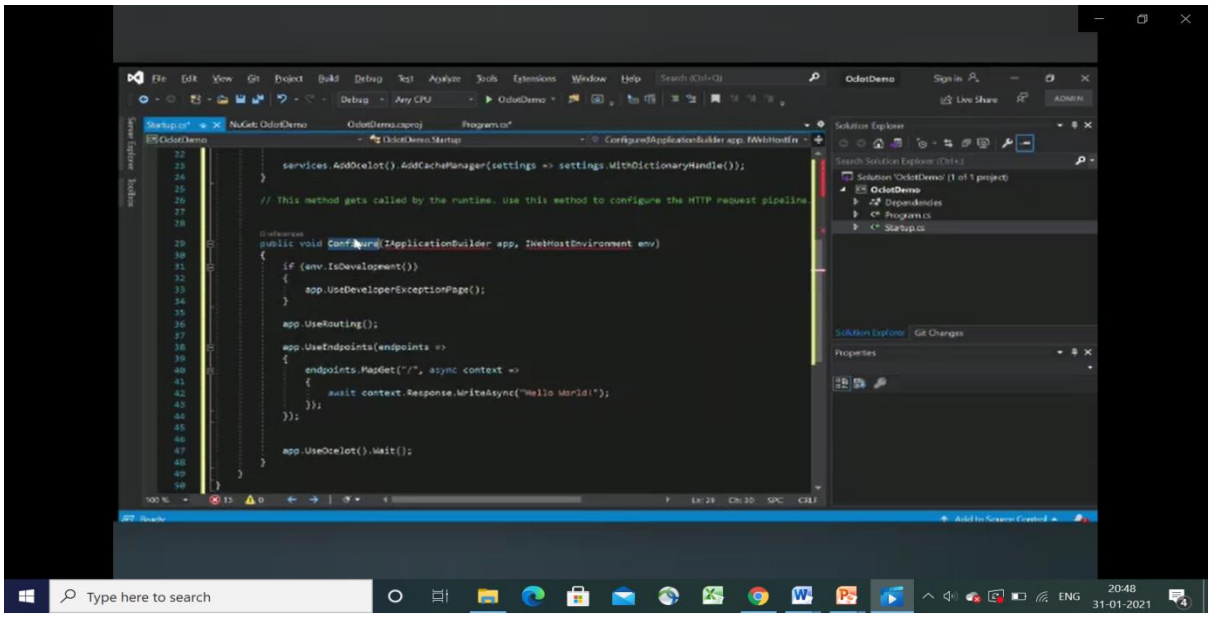
5. Click on Project -> Manage NuGet Package -> Click on Browse -> Search for ocelot package -> install.



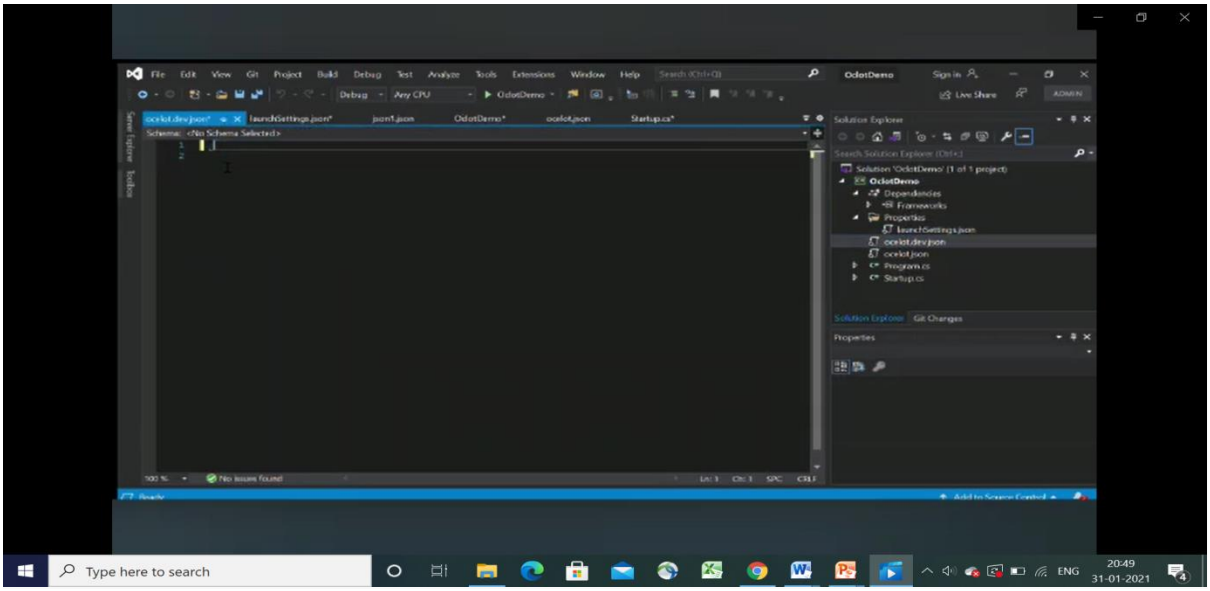
6. Once Ocelot Package is installed Go to Startup.cs



7. Now to Configure Ocelot add services.Addocelot() and app.UseOcelot().Wait() code.

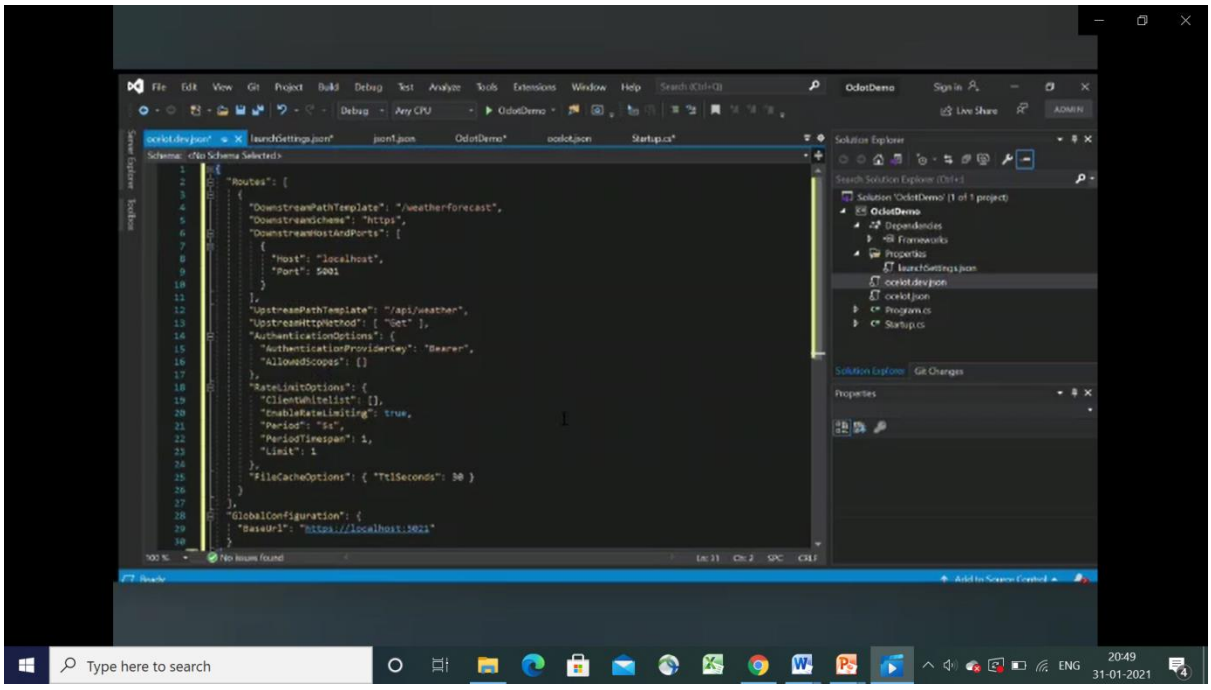


8. Now add the Ocelot JSON file.

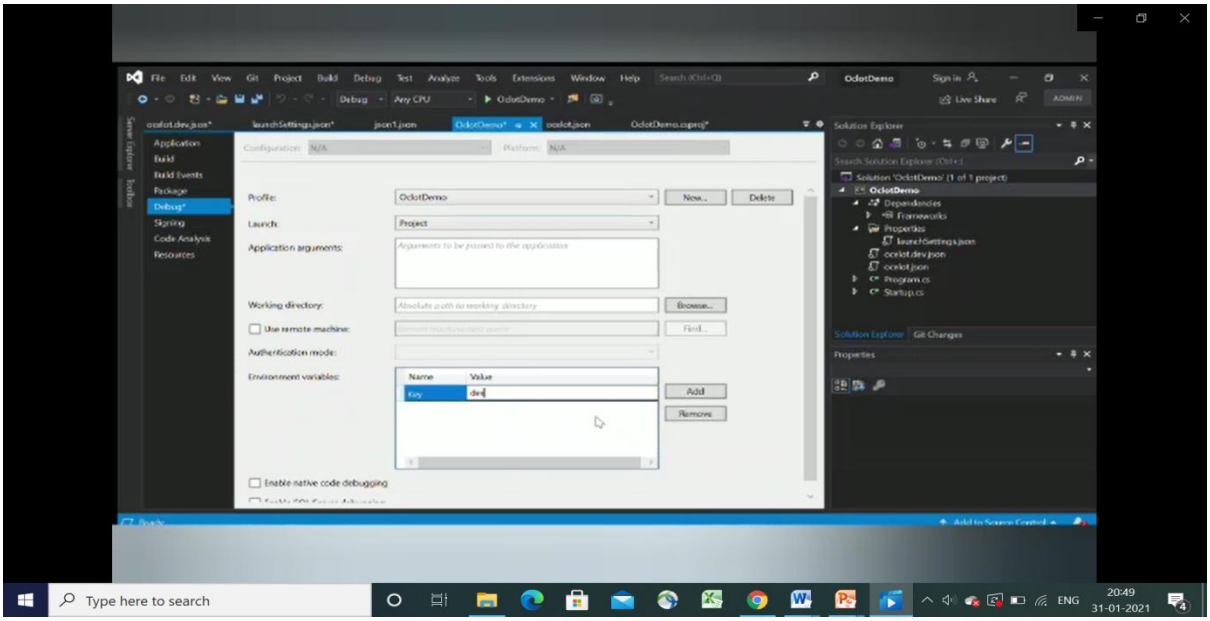


9. Click on Project name -> Add -> New Item -> JSON File ->(Give name)

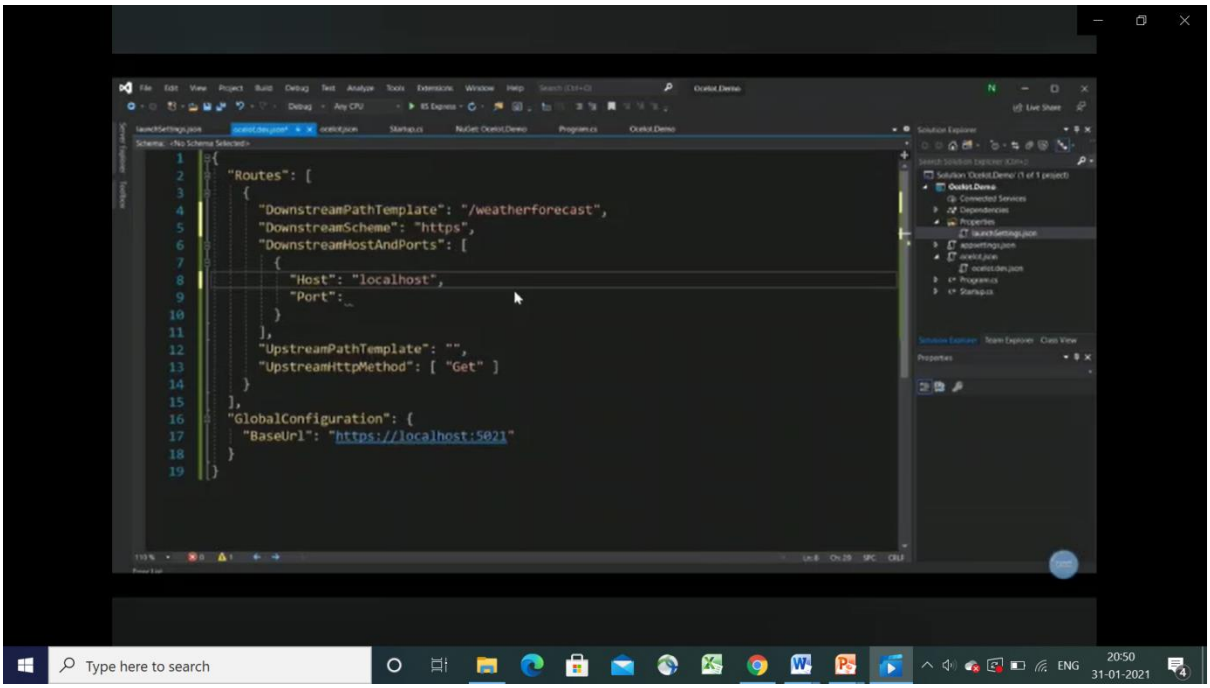
10. Add JSON Code.



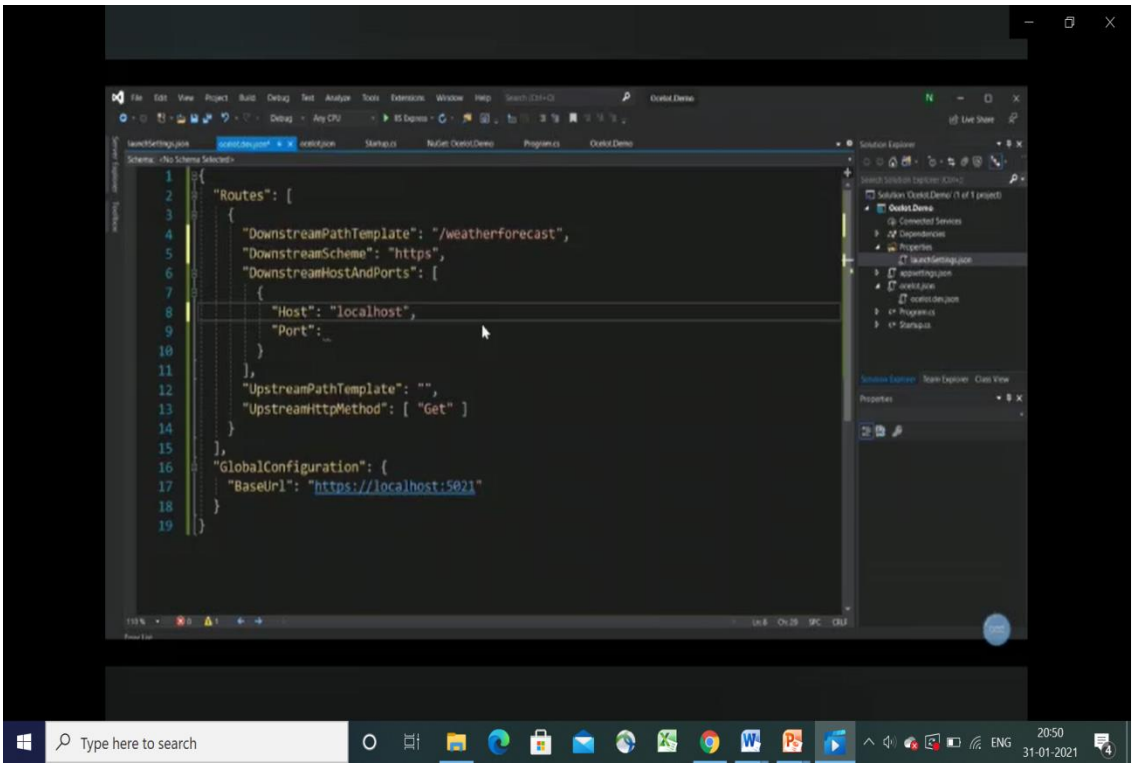
11. Create an Environmental Variable



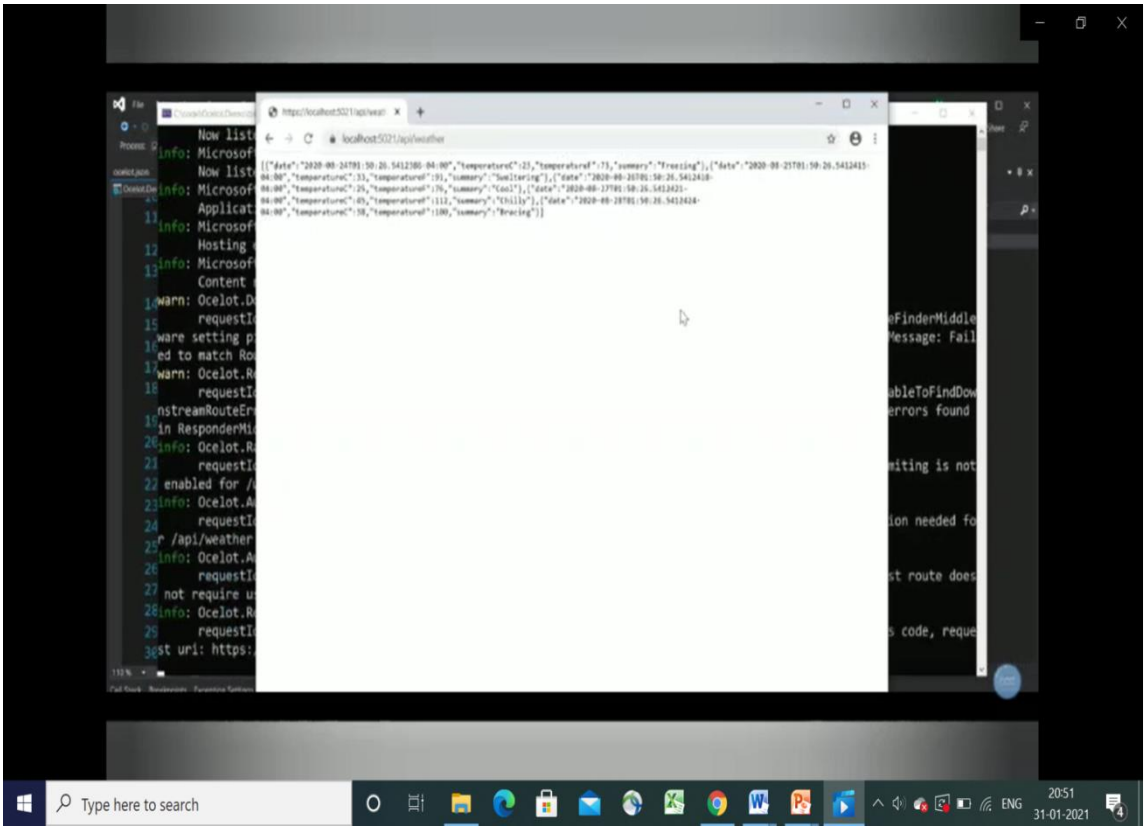
12. Change localhost to 5020 and 5021.



13. Add Configuration to the system.



14. Run the project.



Code:

Program.cs file

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Hosting;
using Microsoft.Extensions.Configuration;
using Microsoft.Extensions.Hosting;
using Microsoft.Extensions.Logging;
namespace Oclot.Demo
{
   public static class Program
   {
      public static void Main(string[] args)
      {
         CreateHostBuilder(args).Build().Run();  }
      public static IHostBuilder CreateHostBuilder(string[] args) =>
         Host.CreateDefaultBuilder(args)
            .ConfigureWebHostDefaults(webBuilder =>
            {
               var env =
Environment.GetEnvironmentVariable("ASPNETCORE_ENVIRONMENT");
               webBuilder.UseStartup<Startup>();
               webBuilder.ConfigureAppConfiguration(config =>
config.AddJsonFile($"ocelot.{env}.json"));
            })
         .ConfigureLogging(logging => logging.AddConsole());
   }
}
```

Startup.cs file

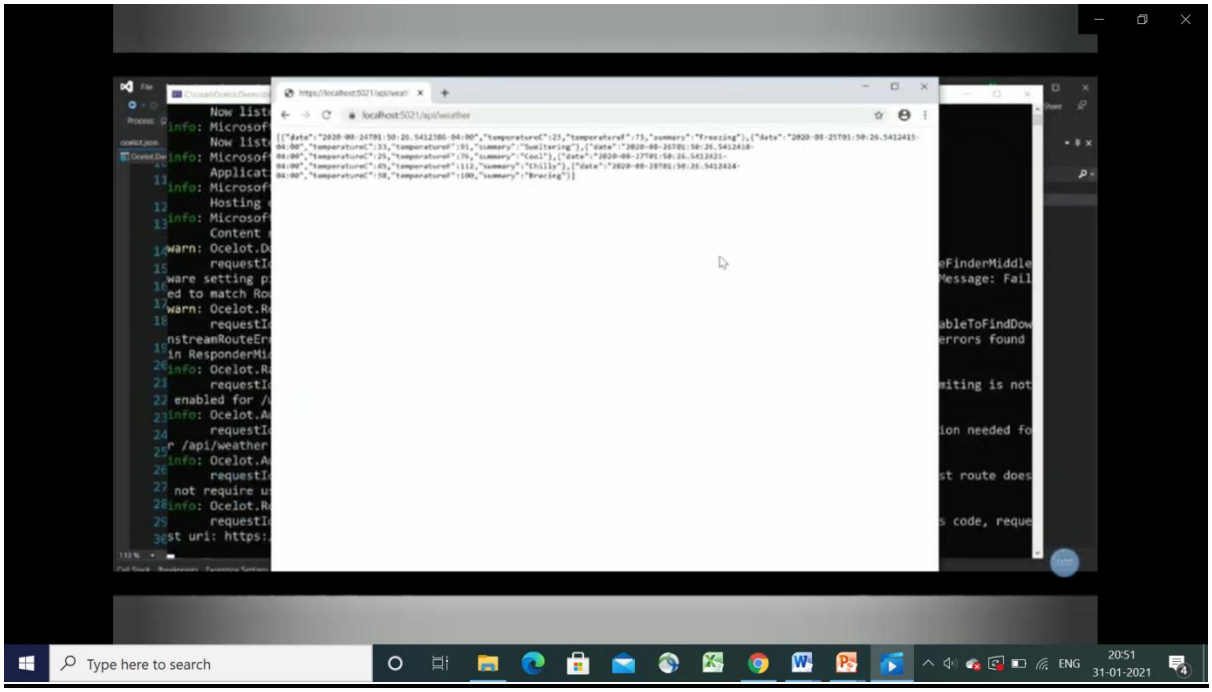
```
namespace Oclot.Demo
{
   public class Startup
   {
      public void ConfigureServices(IServiceCollection services)
      {
         services.AddOcelot().AddCacheManager(settings =>
settings.WithDictionaryHandle());
      }
      public void Configure(IApplicationBuilder app, IWebHostEnvironment env)
      {
         if (env.IsDevelopment())
         {
            app.UseDeveloperExceptionPage();
         }
         app.UseRouting();
         app.UseEndpoints(endpoints =>
         {
            endpoints.MapGet("/", async context =>
            {
```

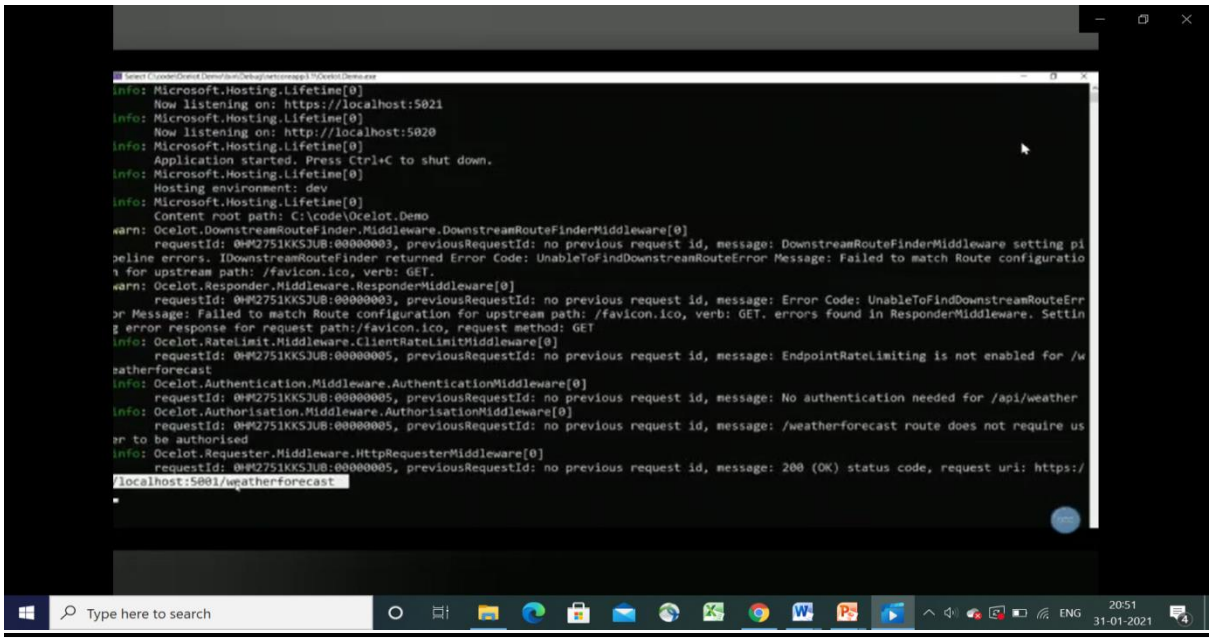
```
                await context.Response.WriteAsync("Hello World!");
            });
        });
        app.UseOcelot().Wait();
      }
    }
}
```

JSON file

```
{
  "Routes": [
    {
      "DownstreamPathTemplate": "/weatherforecast",
      "DownstreamScheme": "https",
      "DownstreamHostAndPorts": [
        {
          "Host": "localhost",
          "Port": 5001
        } ],
      "UpstreamPathTemplate": "/api/weather",
      "UpstreamHttpMethod": [ "Get" ],
      "AuthenticationOptions": {
        "AuthenticationProviderKey": "Bearer",
        "AllowedScopes": []  },
      "RateLimitOptions": {
        "ClientWhitelist": [],
        "EnableRateLimiting": true,
        "Period": "5s",
        "PeriodTimespan": 1,
        "Limit": 1 },
      "FileCacheOptions": { "TtlSeconds": 30 }
    }],
  "GlobalConfiguration": {
    "BaseUrl": "https://localhost:5021"
  }
}
```

OUTPUT


Type here to search
ENG
20:51
31-01-2021


info: Microsoft.Hosting.Lifetime[0]
Now listening on: https://localhost:5021
info: Microsoft.Hosting.Lifetime[0]
Now listening on: http://localhost:5020
info: Microsoft.Hosting.Lifetime[0]
Application started. Press Ctrl+C to shut down.
info: Microsoft.Hosting.Lifetime[0]
Hosting environment: dev
info: Microsoft.Hosting.Lifetime[0]
Content root path: C:\code\Ocelot.Demo
warn: Ocelot.DownstreamRouteFinder.Middleware.DownstreamRouteFinderMiddleware[0]
requestId: 0HM2751KKSJUB:00000003, previousRequestId: no previous request id, message: DownstreamRouteFinderMiddleware setting pipeline errors. IDownstreamRouteFinder returned Error Code: UnableToFindDownstreamRouteError Message: Failed to match Route configuration for upstream path: /favicon.ico, verb: GET.
warn: Ocelot.Responder.Middleware.ResponderMiddleware[0]
requestId: 0HM2751KKSJUB:00000003, previousRequestId: no previous request id, message: Error Code: UnableToFindDownstreamRouteError Message: Failed to match Route configuration for upstream path: /favicon.ico, verb: GET. errors found in ResponderMiddleware. Setting error response for request path:/favicon.ico, request method: GET
info: Ocelot.RateLimit.Middleware.ClientRateLimitMiddleware[0]
requestId: 0HM2751KKSJUB:00000005, previousRequestId: no previous request id, message: EndpointRateLimiting is not enabled for /weatherforecast
info: Ocelot.Authentication.Middleware.AuthenticationMiddleware[0]
requestId: 0HM2751KKSJUB:00000005, previousRequestId: no previous request id, message: No authentication needed for /api/weather
info: Ocelot.Authorisation.Middleware.AuthorisationMiddleware[0]
requestId: 0HM2751KKSJUB:00000005, previousRequestId: no previous request id, message: /weatherforecast route does not require user to be authorised
info: Ocelot.Requester.Middleware.HttpRequesterMiddleware[0]
requestId: 0HM2751KKSJUB:00000005, previousRequestId: no previous request id, message: 200 (OK) status code, request uri: https://localhost:5001/weatherforecast
Type here to search
ENG
20:51
31-01-2021