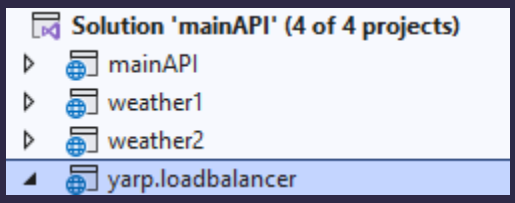
Implementing a Round-Robin Load Balancer with YARP



Added 3 sample web API projects and One web API project for loadbalancer proxy

```
"https": {
  "commandName": "Project",
  "dotnetRunMessages": true,
  "launchBrowser": true,
  "launchUrl": "swagger",
  "applicationUrl": "https://localhost:7201;http://localhost:5263",
  "environmentVariables": {
    "ASPNETCORE_ENVIRONMENT": "Development"
  }
},
```

```
"https": {
  "commandName": "Project",
  "dotnetRunMessages": true,
  "launchBrowser": true,
  "launchUrl": "swagger",
  "applicationUrl": "https://localhost:7202;http://localhost:5219",
  "environmentVariables": {
    "ASPNETCORE_ENVIRONMENT": "Development"
  }
},
```

```
"https": {
  "commandName": "Project",
  "dotnetRunMessages": true,
  "launchBrowser": true,
  "launchUrl": "swagger",
  "applicationUrl": "https://localhost:7203;http://localhost:5131",
  "environmentVariables": {
    "ASPNETCORE_ENVIRONMENT": "Development"
  }
```

Updated Launch settings files with port for continence mainAPI-7201, weather1-7202, weather2-7203.

Yarp.ReverseProxy by Microsoft
Reverse proxy toolkit for building fast proxy servers in .NET using the infrastructure from ASP.NET and .NET
2.2.0

Added Yarp.ReverseProxy Nuget package to load balancer project

yarp.loadbalancer

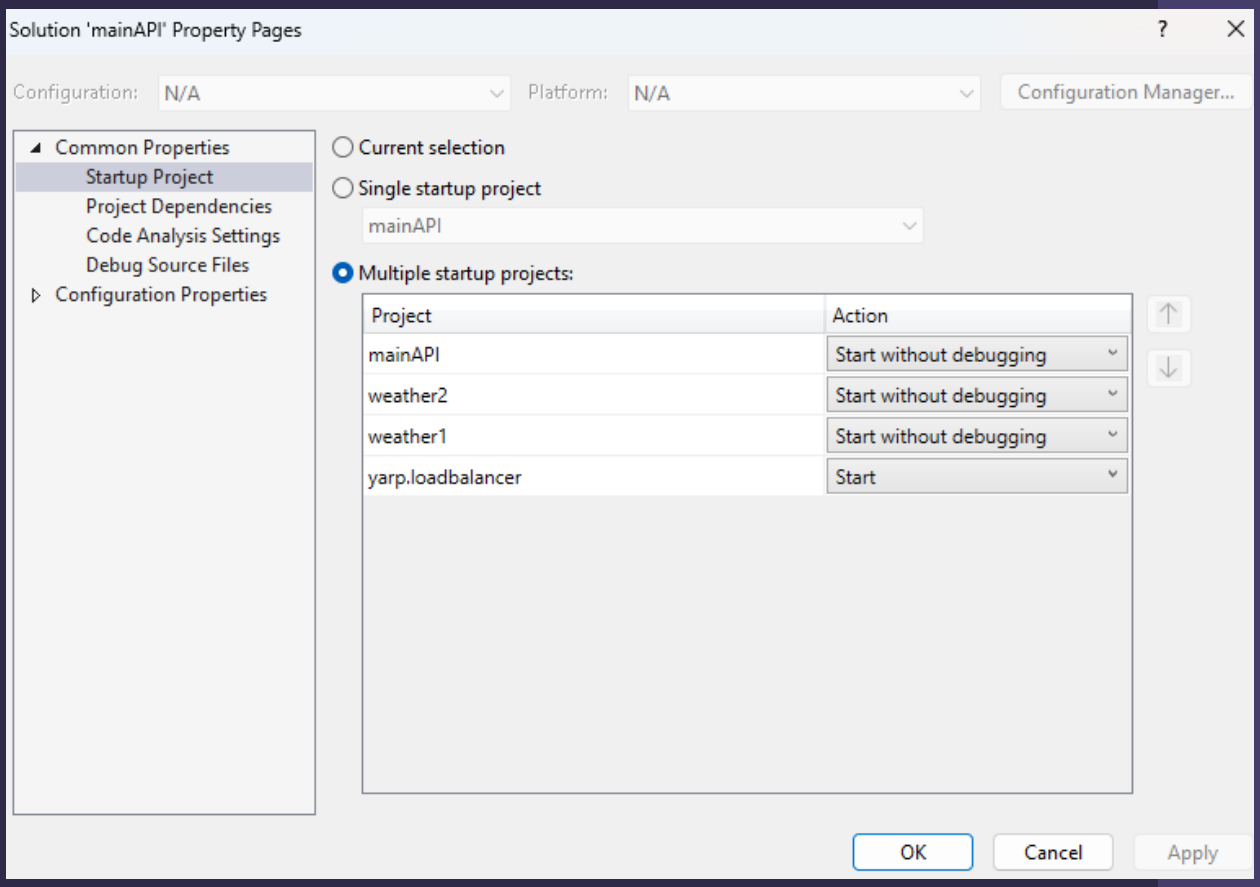
```
var builder = WebApplication.CreateBuilder(args);

builder.Services.AddEndpointsApiExplorer();
builder.Services.AddSwaggerGen();
builder.Services.AddReverseProxy()
    .LoadFromConfig(builder.Configuration.GetSection("ReverseProxy"));
builder.Services.AddHealthChecks();
var app = builder.Build();

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

app.UseHttpsRedirection();

app.MapReverseProxy();
app.MapHealthChecks("health");
app.Run();
```

Configured services for yarp and health check inside Program.cs of load balancer project

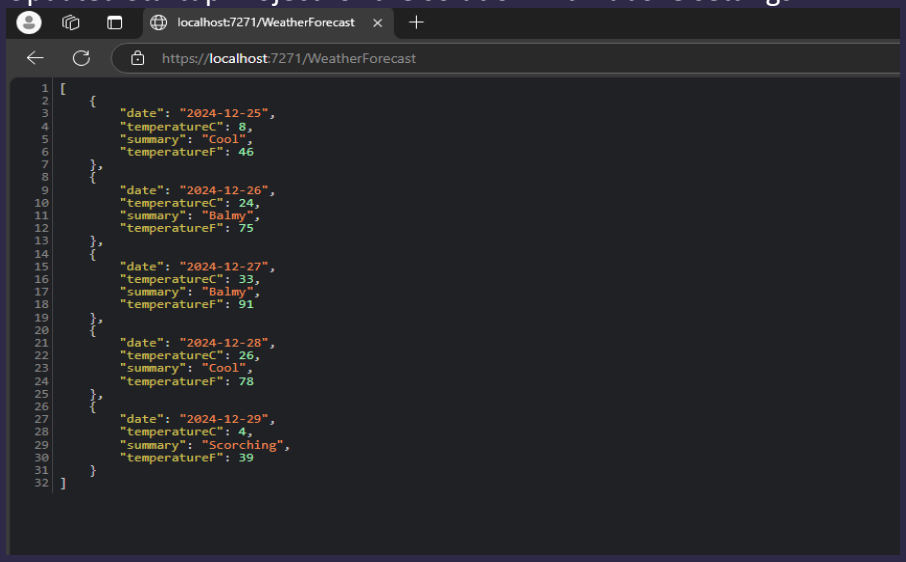
Schema: https://json.schemastore.org/appsettings.json

```
{
  "Logging": {
    "LogLevel": {
      "Default": "Information",
      "Microsoft.AspNetCore": "Warning"
    }
  },
  "AllowedHosts": "*",
  "ReverseProxy": {
    "Routes": {
      "weatherRoute": {
        "ClusterId": "apiCluster",
        "Match": {
          "Host": "localhost",
          "Path": "WeatherForecast/{**catchall}"
        }
      }
    },
    "Clusters": {
      "apiCluster": {
        "LoadBalancingPolicy": "RoundRobin",
        "Destinations": {
          "mainapp": {
            "Address": "https://localhost:7201/"
          },
          "weather1": {
            "Address": "https://localhost:7202/"
          },
          "weather2": {
            "Address": "https://localhost:7203/"
          }
        }
      }
    }
  }
}
```

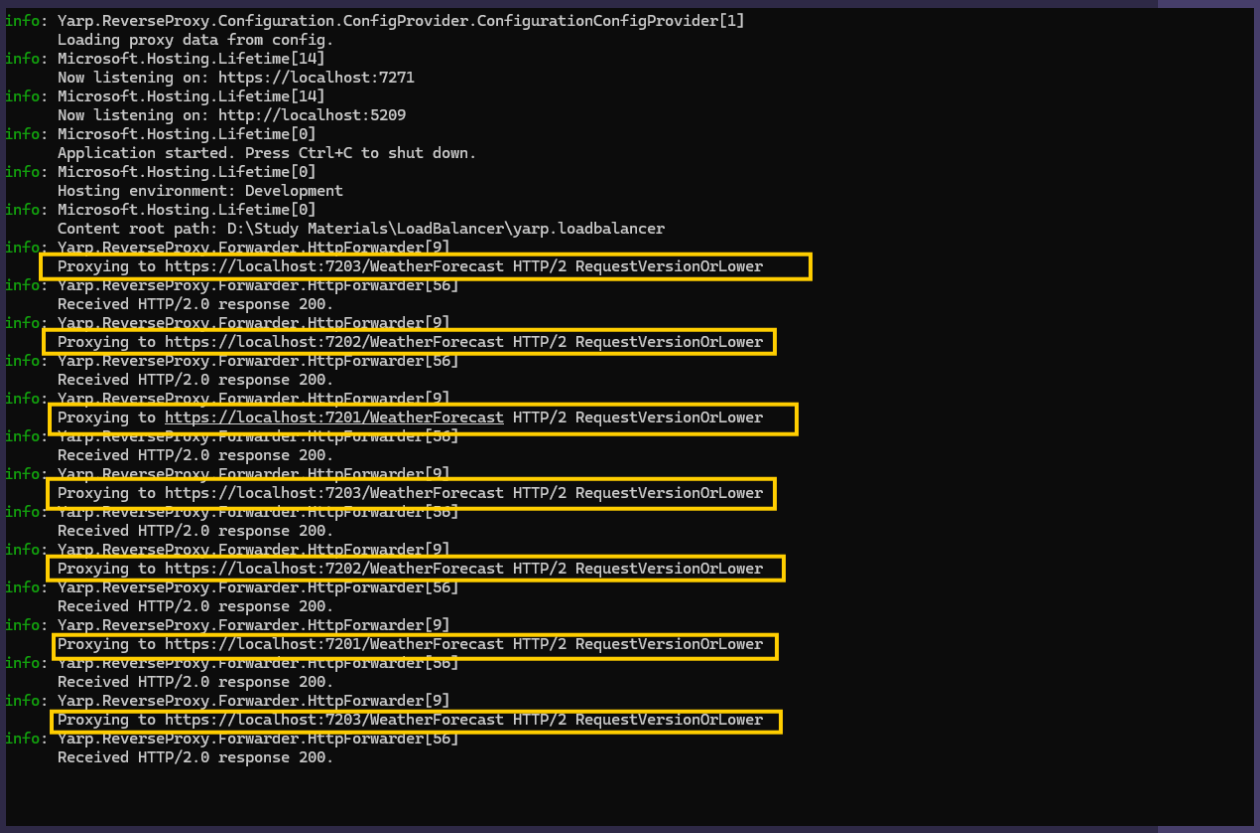
Added ReverseProxy section to application settings and configure Routes and Clusters.



Updated Startup Project for the solution with above settings



Started Project and Refreshed weather forecast endpoint to check RoundRobin assignment



You can find it is assigning Load balancer according to round robin manner.

Thanking You !