[언택트 시대 – 필수 교양 실시간 웹 메시징 기술]

.NET8 ASP.NET Core Signal R

초거대 메시징 서비스 개발하기

GitHub Source Address
https://github.com/eddy19740523/EddySignalRSolution

엠소프트웨어 강창훈 대표
Microsoft MVP 2023

Microsoft® Most Valuable Professional

Real Time Web Technologies

실시간 웹 (Real Time Web) ?

인터넷에서 사용자들로 하여금 창작자가 정보를 만들어내는 즉시 수신할 수 있도록 하는 기술 혹은 서비스들
대표 실시간 웹기술들 : HTML5 Websocket , WebRTC

- Facebook,Twitter등 각종 SNS, Web 기반 채팅 솔루션
- Slack, Jandi 각종 실시간기반 협업 툴
- 실시간 대시보드 및 챠트,웹 푸시기술
- 웹 게임
- WebRTC 등 실시간 화상통화 기술 등

HTML5 Web socket & Server Side Technologies

HTML5 WebSocket

- 표준 웹 브라우저에 탑재되어 있는 HTML5 실시간 메시징 API(Application Programming Interface) 기술
- 웹 브라우저에서 웹 브라우저 와 웹서버 간 연결기반 실시간 양방향 메시징 통신기술 제공
- **Web Browser Client Side 실시간 메시징 기술**

Server Side Web Socket 지원 기술들

- 웹 브라우저와 웹서버 간 연결 기반 통신을 위해서는 웹 서버측에도 Web socket 지원 환경 및 기술 필요
- **Node.js Socket.IO, ASP.NET SignalR , JAVA Spring SockeJS, Python websockets**

.NET 8 LTS(Preview, 8.0 rc.2) – 2023.10.10

.NET 8.0 다운로드

무엇을 다운로드할지 잘 모르시겠나요? 최신 버전의 .NET에 대한 권장 다운로드를 확인하세요.

8.0.0-rc.2 Go-live 보안 패치

릴리스 노트 최신 출시일 2023년 10월 10일

앱 빌드 - SDK

SDK 8.0.100-rc.2

OS	설치 관리자	바이너리
Linux		Arm32 | Arm32 Alpine | Arm64 | Arm64 Alpine | x64 | x64 Alpine
macOS	Arm64 | x64	Arm64 | x64
Windows	Arm64 | x64 | x86 | winget 지침	Arm64 | x64 | x86
모두	dotnet-install scripts	

풀 버전
8.0.100-rc.2.23502.2

Visual Studio 지원
Visual Studio 2022 (v17.8 latest Preview)
Visual Studio for Mac 2022 (v17.6.1)

앱 실행 - 런타임

ASP.NET 코어 런타임 8.0.0-rc.2

ASP.NET Core 런타임을 사용하면 기존 웹/서버 애플리케이션을 실행할 수 있습니다. **Windows에서는 .NET 런타임 및 IIS 지원이 포함된 Hosting 번들을 설치하는 것이 좋습니다.**

풀 버전
8.0.0-rc.2.23480.2

IIS 런타임 지원(ASP.NET Core 모듈 v2)
18.0.23273.0

OS	설치 관리자	바이너리
Linux		Arm32 | Arm32 Alpine | Arm64 | Arm64 Alpine | x64 | x64 Alpine
macOS		Arm64 | x64
Windows	Hosting Bundle | x64 | x86 | winget 지침	Arm64 | x64 | x86

.NET /

.NET Core

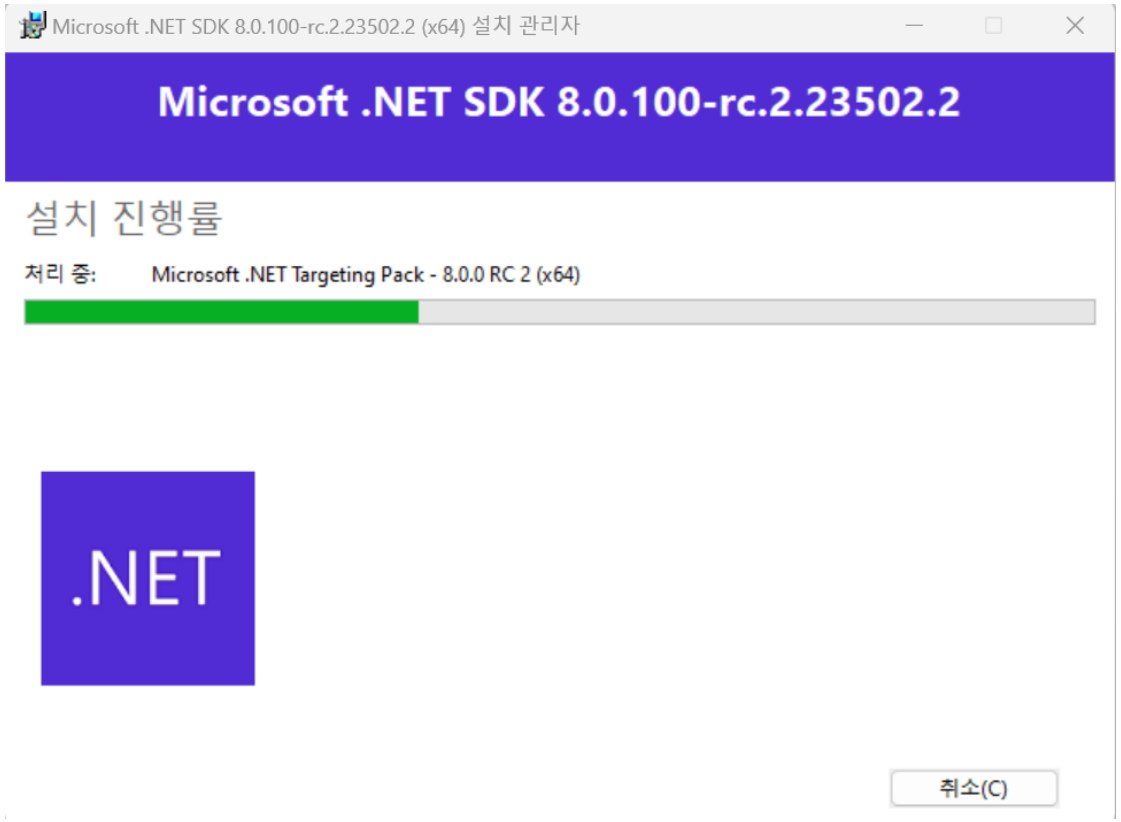
7.0.12	7.0.402
Latest runtime	Latest SDK

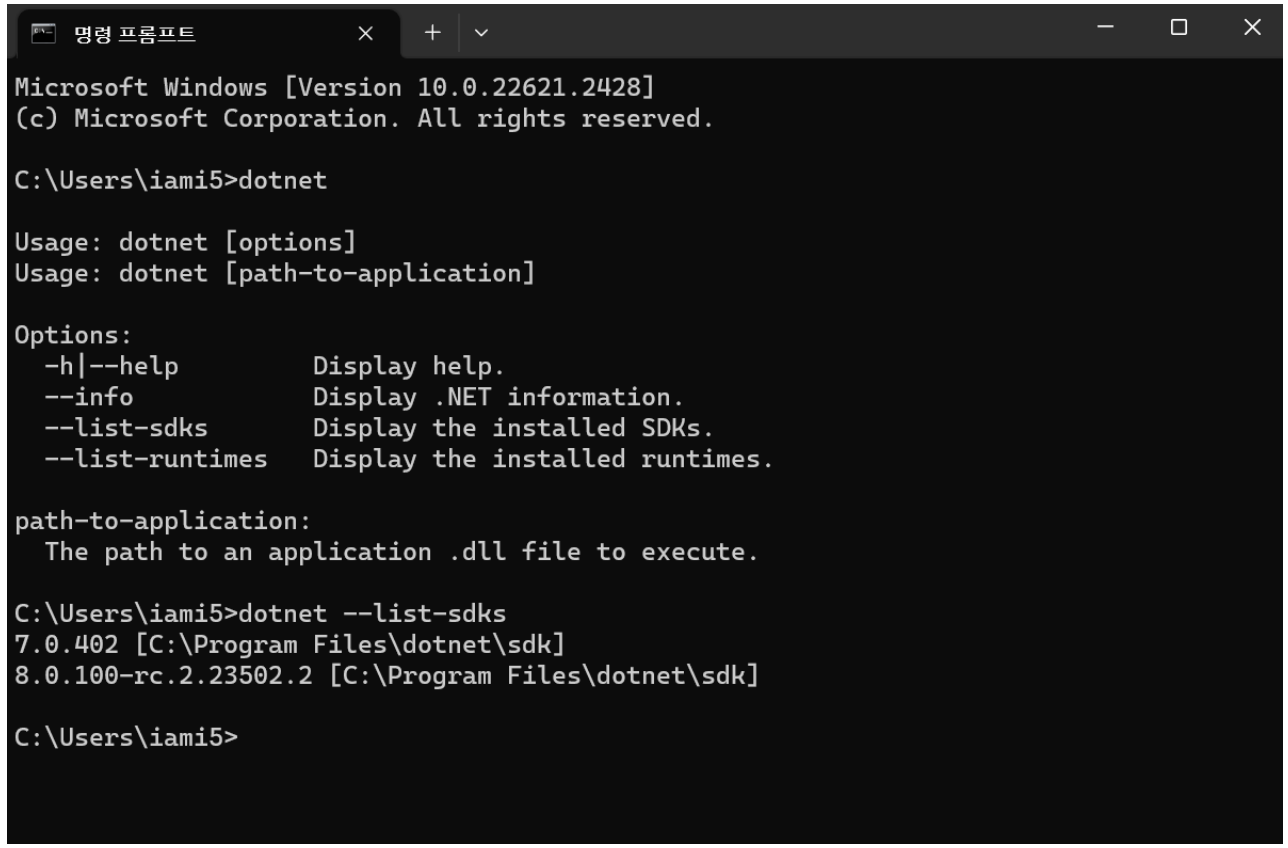
Channels

Channel	Support	Latest release	Latest release date	End of Life date
8.0	Preview / Go Live (LTS)	8.0.0-rc.2	2023-10-10	-
7.0	Active	7.0.12	2023-10-10	2024-05-14
6.0	Active (LTS)	6.0.23	2023-10-10	2024-11-12
5.0	End of Life	5.0.17	2022-05-10	2022-05-10
3.1	End of Life (LTS)	3.1.32	2022-12-13	2022-12-13
3.0	End of Life	3.0.3	2020-02-18	2020-03-03
2.2	End of Life	2.2.8	2019-11-19	2019-12-23
2.1	End of Life (LTS)	2.1.30	2021-08-19	2021-08-21
2.0	End of Life	2.0.9	2018-07-10	2018-10-01
1.1	End of Life (LTS)	1.1.13	2019-05-14	2019-06-27
1.0	End of Life (LTS)	1.0.16	2019-05-14	2019-06-27

OS별 SDK 다운로드 설치
https://dotnet.microsoft.com/ko-kr/download/dotnet/8.0

.NET 8 LTS(Preview, 8.0 rc.2) – 2023.10.10 SDK 설치

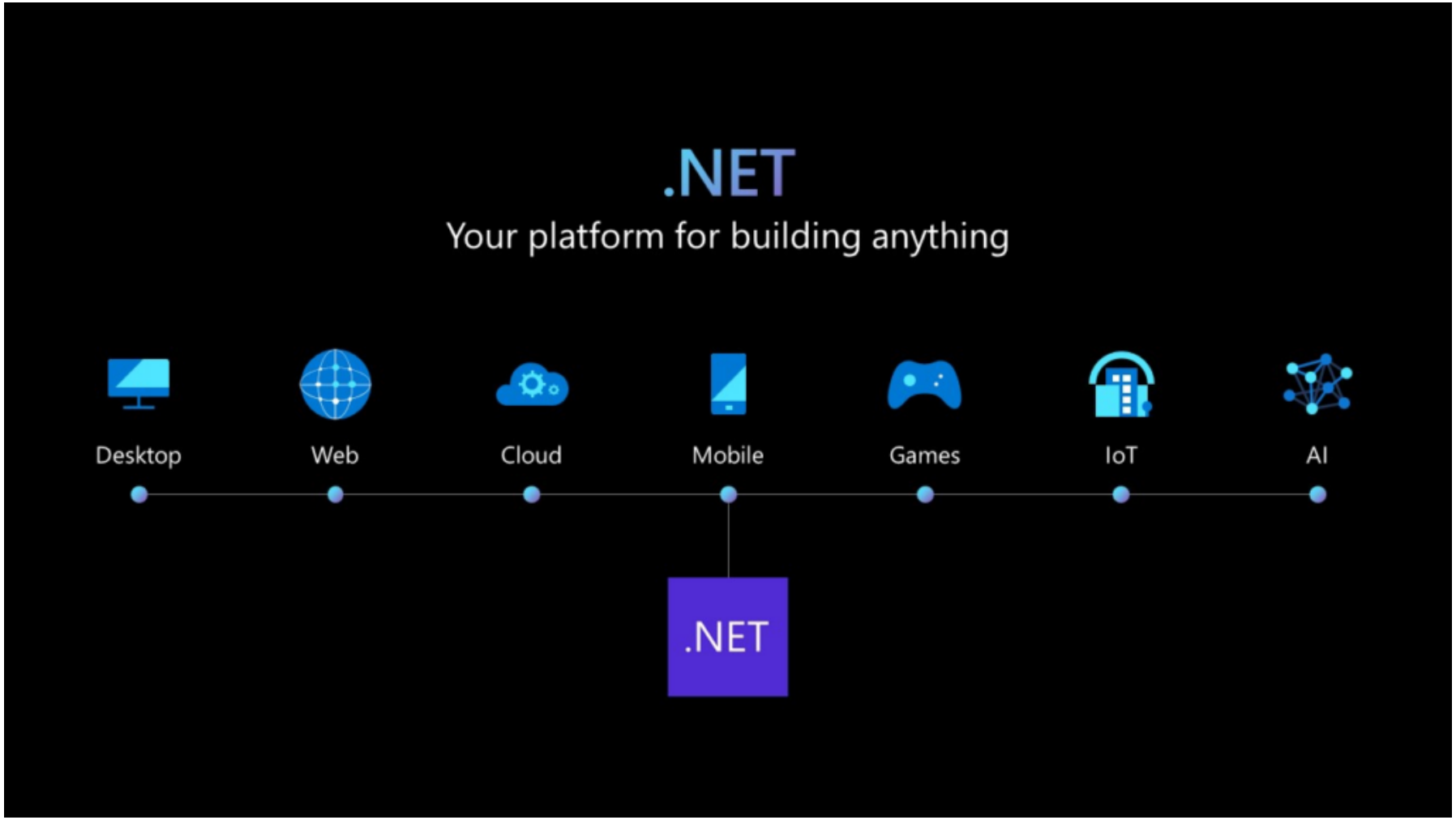




dotnet –list-sdks

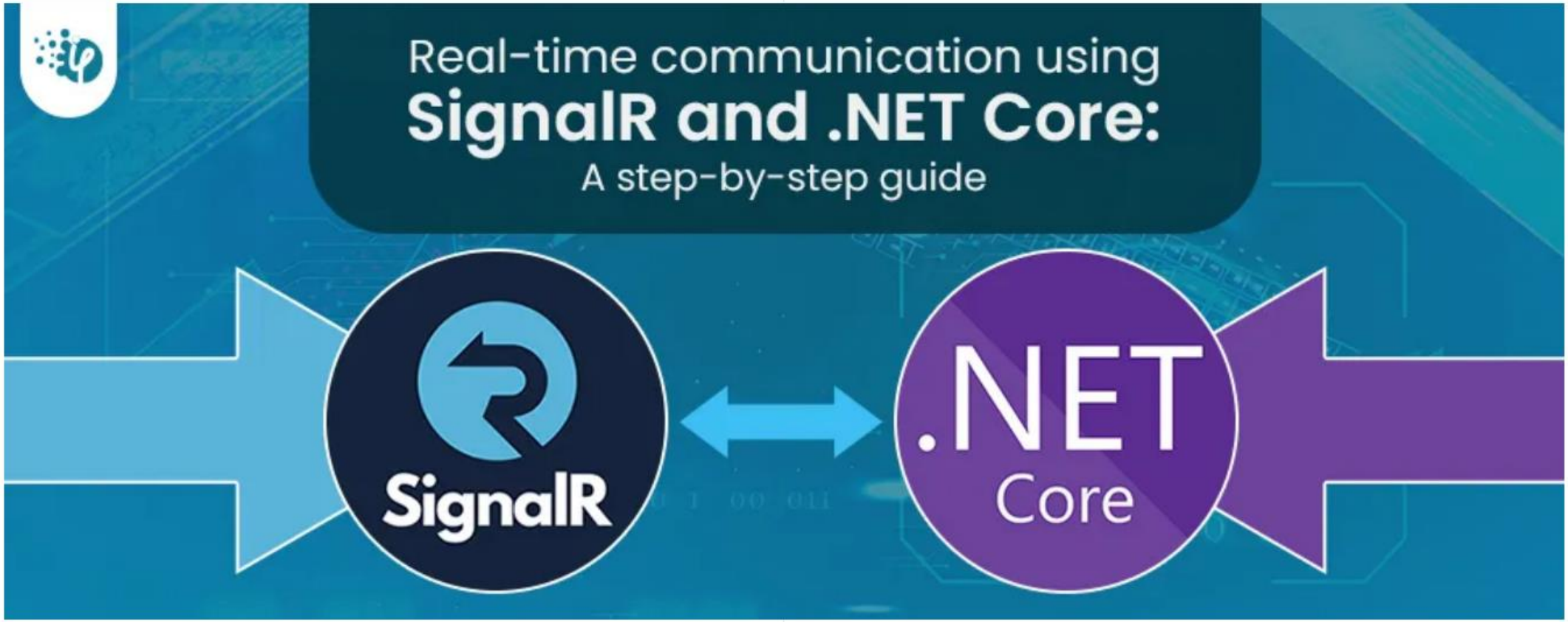
Windows용 .NET8 LTS 8.0 rc2 다운로드 설치
https://dotnet.microsoft.com/ko-kr/download/dotnet/thank-you/sdk-8.0.100-rc.2-windows-x64-installer

.NET 8 for building anything..



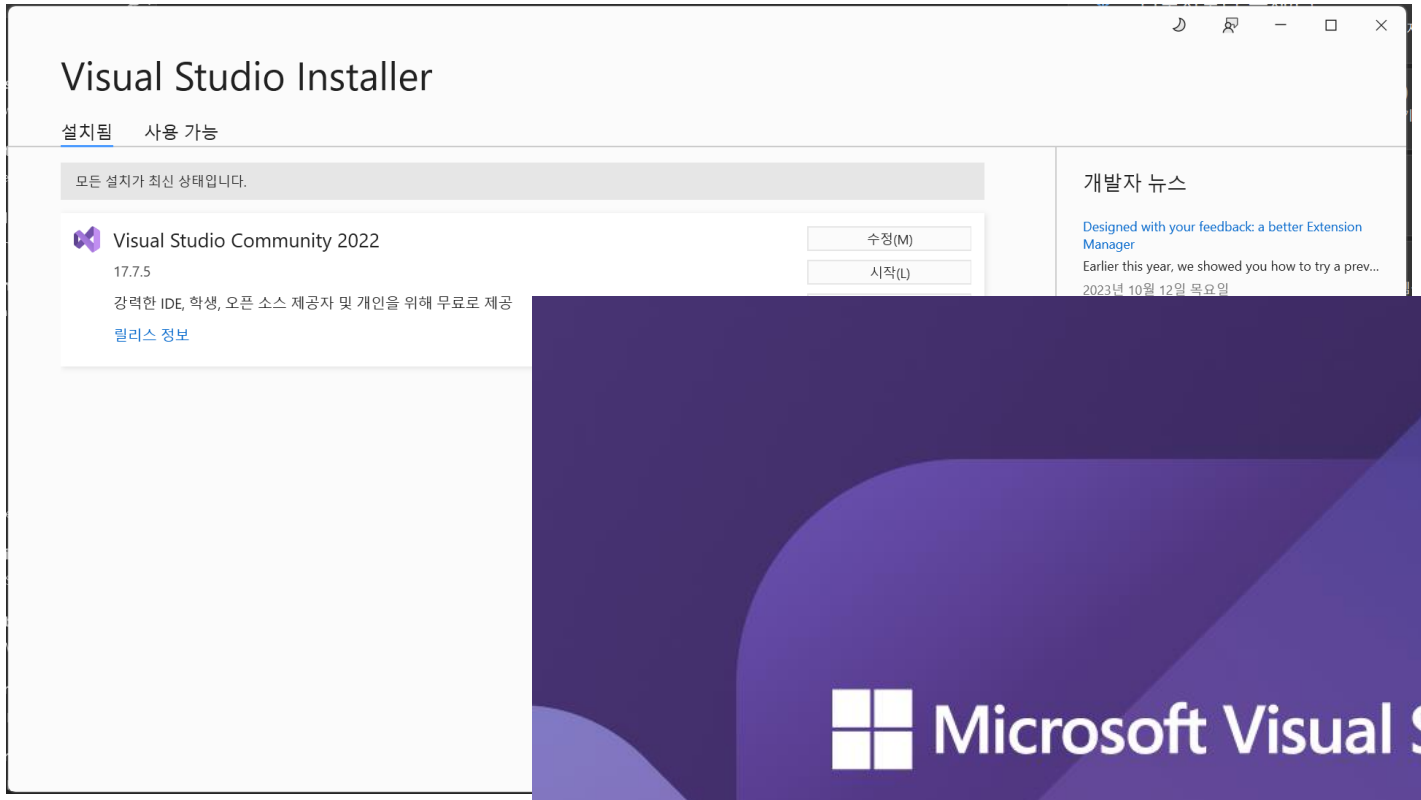
https://learn.microsoft.com/ko-kr/dotnet/core/whats-new/dotnet-8

ASP.NET Core 8.0 & SignalR



https://learn.microsoft.com/ko-kr/aspnet/core/release-notes/aspnetcore-8.0?view=aspnetcore-8.0
https://learn.microsoft.com/en-us/aspnet/core/signalr/introduction?view=aspnetcore-8.0

Visual Studio **2022 17.8 Preview 3 업데이트 필요**



1)Visual Studio Installer 17.7.x 업데이트 실시
-사용가능탭>Visual Studio 2022 17.8 Preview 3 설치 업데이트 필요

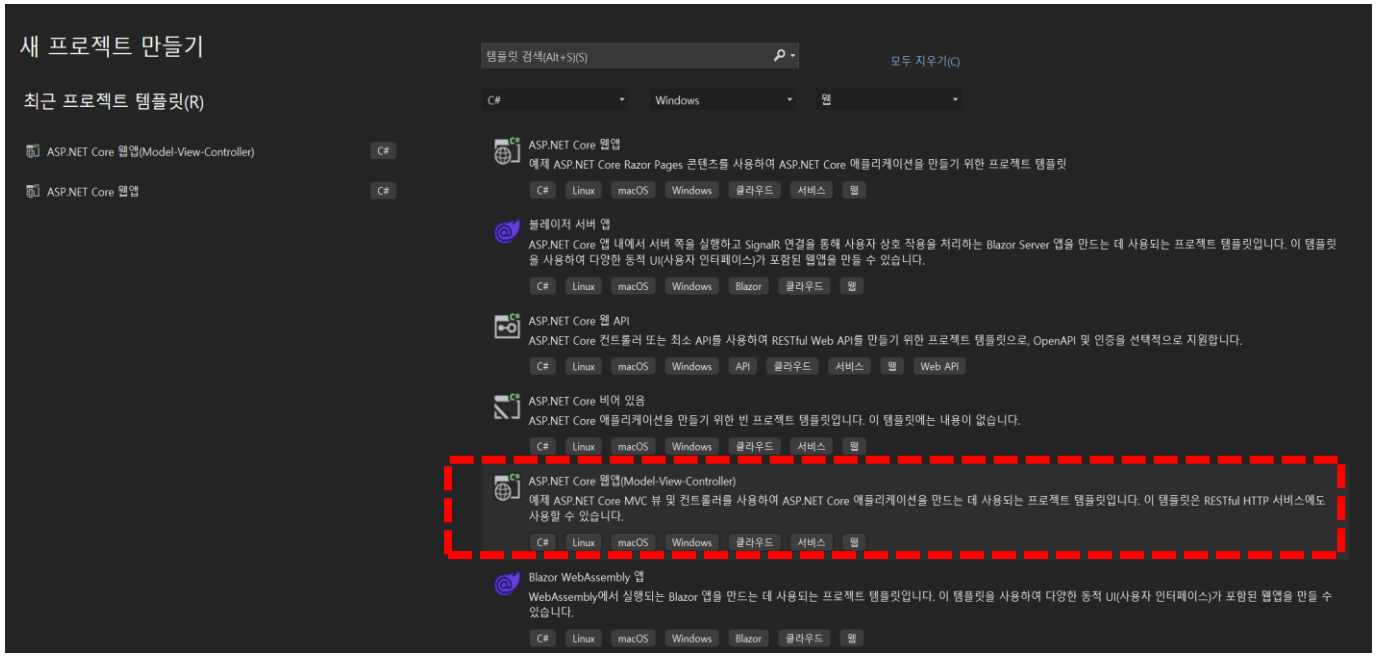
https://anthonygiretti.com/2023/06/10/net-8-why-net-8-preview-doesnt-show-up-in-visual-studio-2022/

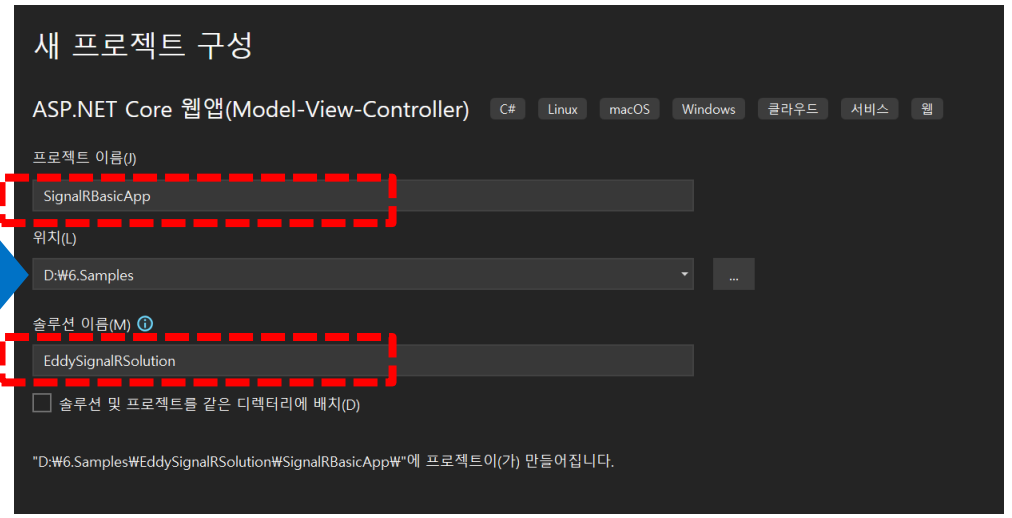
ASP.NET Core Signal R is RealTime Web Messaging Technology

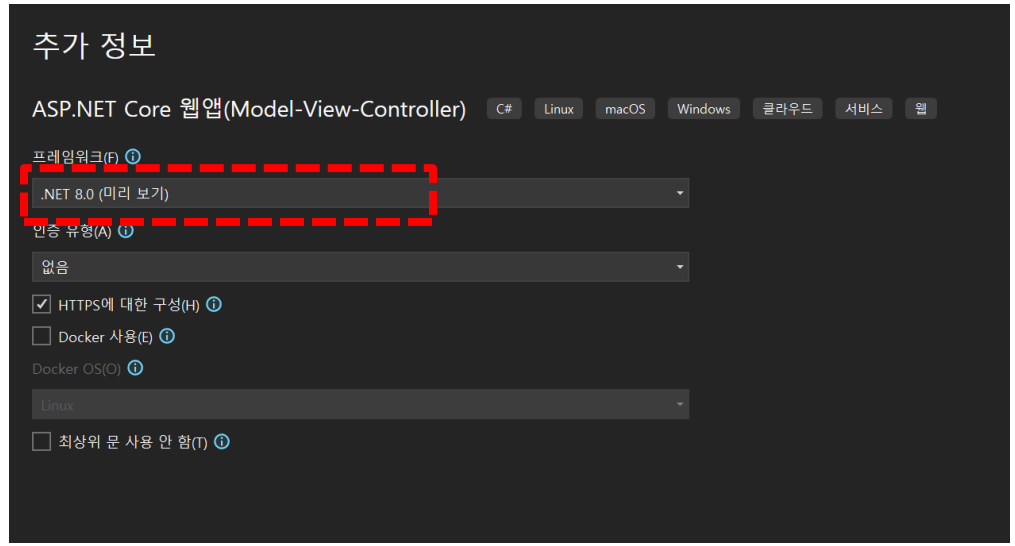
- ASP.NET Core 기반 **크로스 플랫폼 실시간 웹 개발 오픈소스 라이브러리**
- .NET 언어(C#,VB.NET)를 이용 서버에서 웹 브라우저로의 **푸시 기반 어플리케이션 개발** 용이
- **연결 관리 자동화 처리**
- 모든 연결 된 클라이언트에 브로드 캐스팅 기능 제공 및 개별 클라이언트,그룹 메시징 기능제공
- **다양한 스케일 아웃 방식** 제공

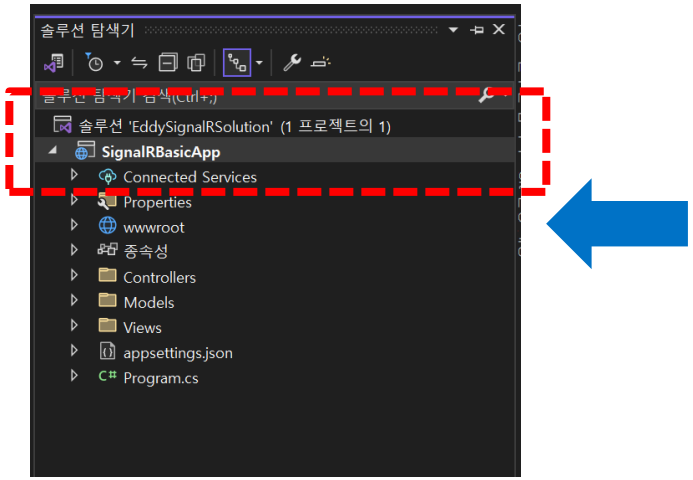
https://learn.microsoft.com/en-us/aspnet/core/signalr/introduction?view=aspnetcore-8.0

ASP.NET Core MVC 웹프로젝트 생성 및 빌드/실행









ASP.NET Core Signal R 실시간 웹 채팅 개발하기

STEP1 : ASP.NET CORE Web Project 만들기

STEP2: SignalR Client Library 추가 또는 CDN 참조하기

STEP3: SignalR Hub 클래스 만들기

STEP4: SignalR 지원 Project 구성하기

STEP5: 심플 웹 채팅 페이지(뷰) 구성하기

STEP6: 웹 채팅 테스트 하기

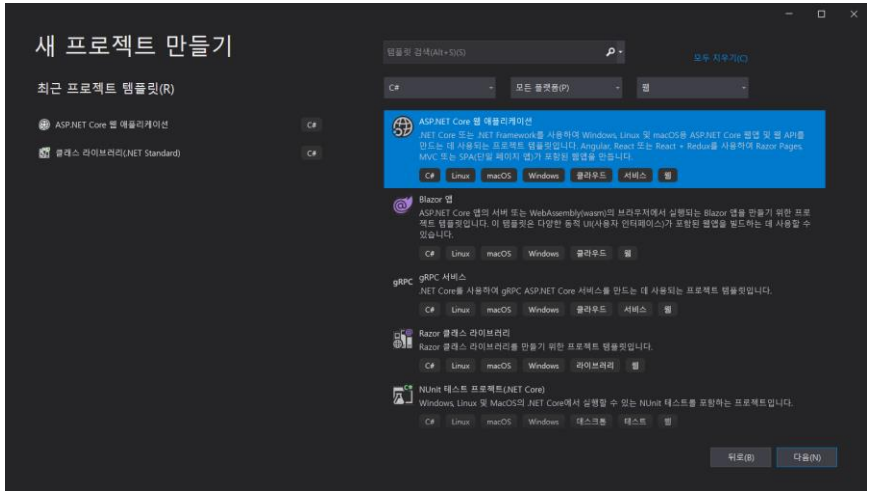
STEP7: 그룹채팅/타겟 메시징 처리하기

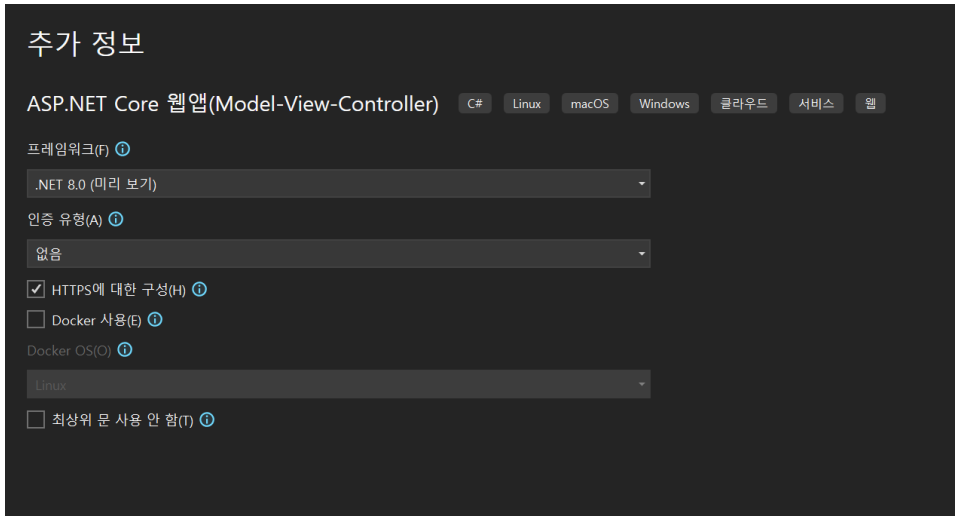
STEP8: SignalR 크로스 도메인(CORS) 지원 설정하기

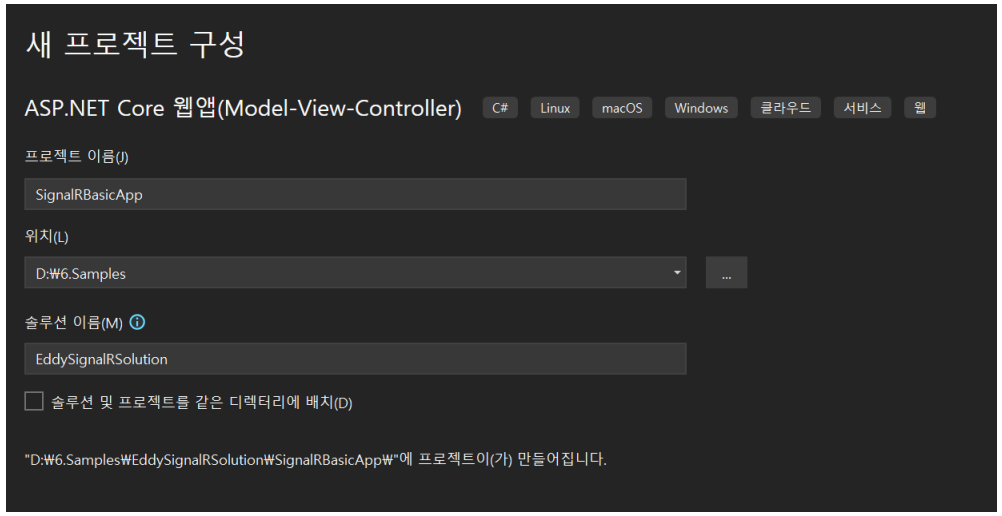
ASP.NET Core Signal R 실시간 웹 채팅 개발하기

STEP1 : ASP.NET CORE Web Project 만들기

1. Visual Studio 2022 Preview를 시작합니다.
2. 새프로젝트 만들기를 클릭합니다.
3. 언어를 C# 프로젝트 형식을 웹을 선택합니다.
   - ASP.NET Core 웹 어플리케이션 프로젝트 템플릿을 선택하고 다음>
4. 새프로젝트 구성 화면에서 아래 내용을 입력하고 만들기 클릭
   - 프로젝트명 과 솔루션명을 입력합니다. Ex) SignalRBasicApp / EddySignalRSolution
   - 솔루션 폴더가 생성될 경로를 지정합니다. Ex) D:\
5. 새 ASP.NET Core 웹 애플리케이션 만들기 화면에서 하기 내용 선택 후 만들기 클릭
   -.NET Core Framework 선택
   -NET 8.0 선택
   -웹 응용 프로그램(모델-뷰-컨트롤러) 프로젝트 템플릿 선택 후 만들기 클릭

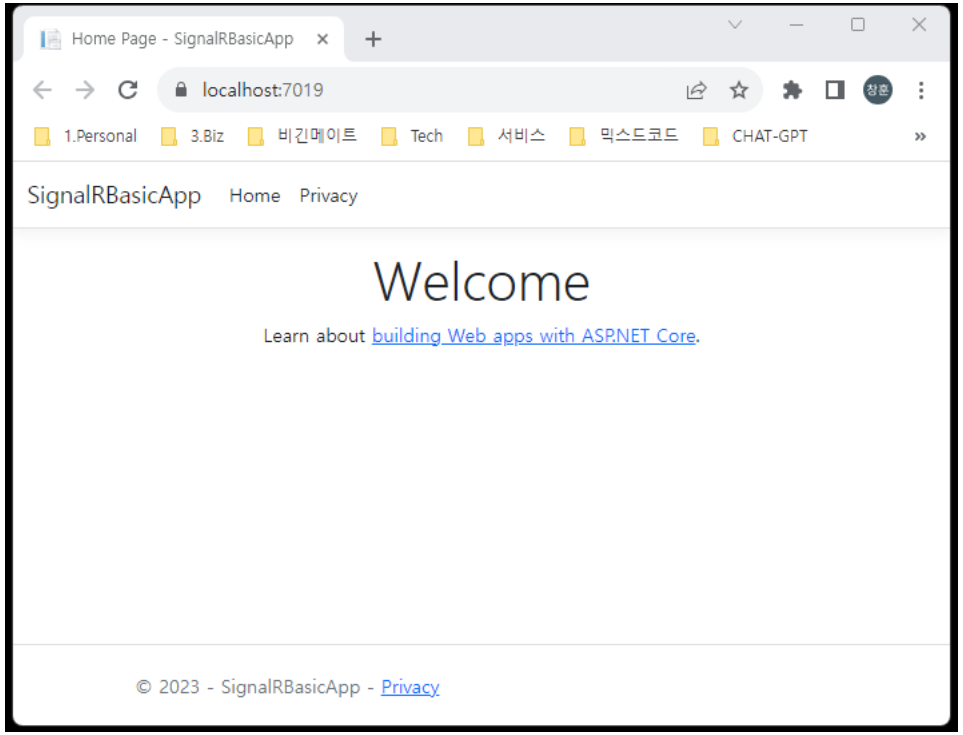


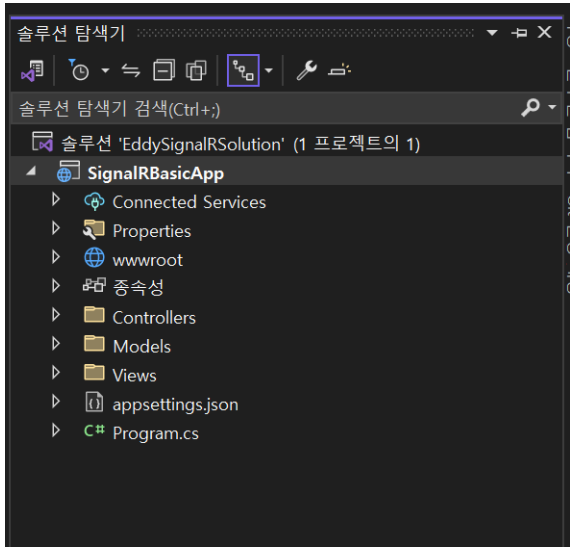




ASP.NET Core Signal R 실시간 웹 채팅 개발하기

STEP1 : ASP.NET CORE Web Project 실행





1. 솔루션 탐색기 확인
-EddySignalRSolution 솔루션
아래 프로젝트 구조확인

2. SignalRBasicApp 실행하기
- F5 또는 디버그 메뉴> 디버깅 시작 클릭
- 또는 화면내 초록색 디버깅 버튼 클릭

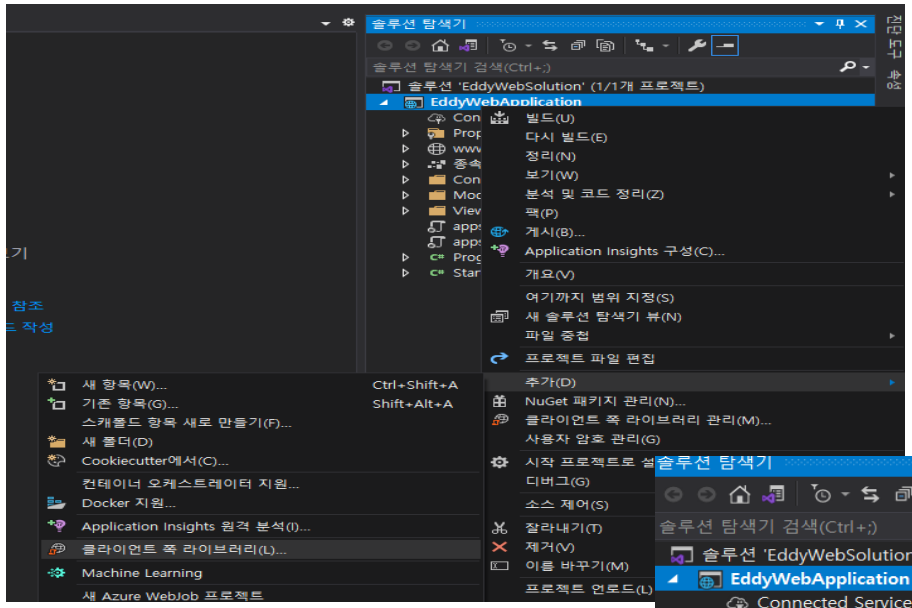
3.웹브라우저에서 해당 어플리케이션 실행확인
-Controllers : MVC 에 Controller 역할제공
-Models : MVC에 Model 역할제공
-Views : MVC에 View 역할 제공

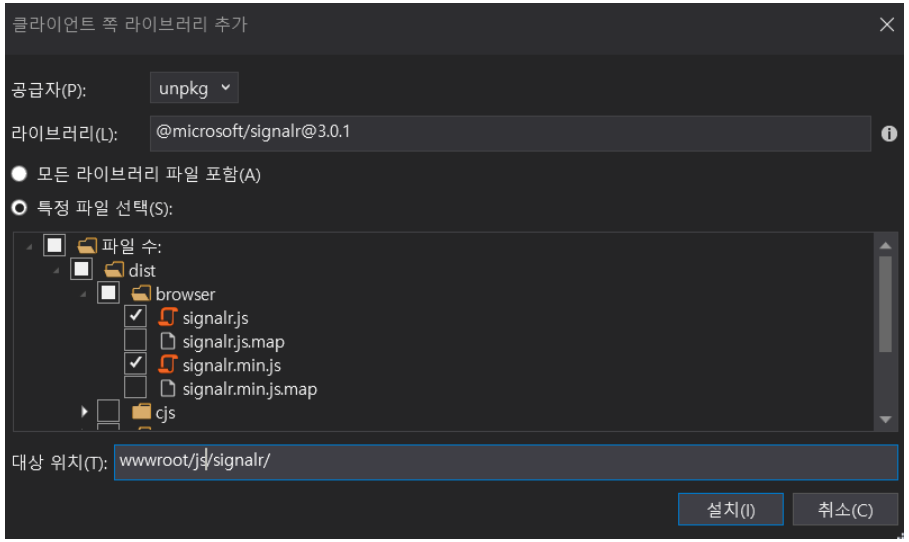
4.웹 브라우저를 닫으면 디버깅 자동 종료

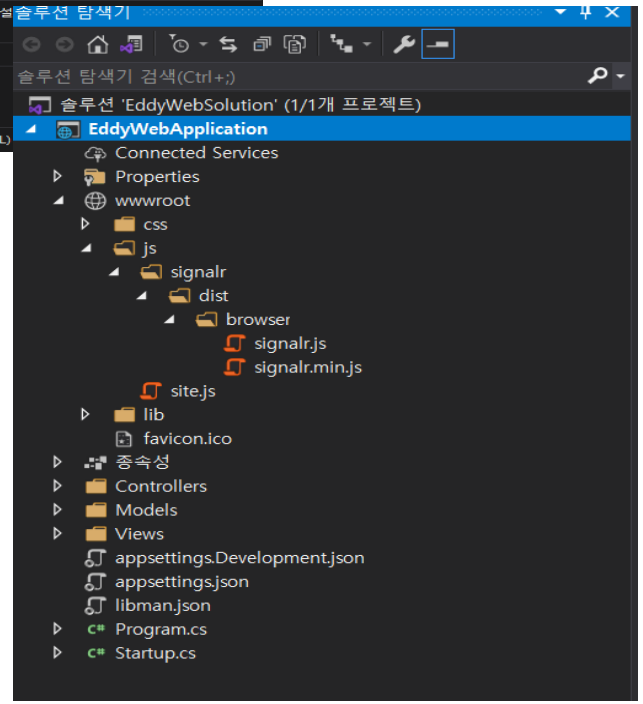
ASP.NET Core Signal R 실시간 웹 채팅 개발하기

STEP2: SignalR Client Library 추가하기

1. 프로젝트 오른쪽 마우스 클릭 > 추가 클릭>
2. 클라이언 쪽 라이브러리 추가를 클릭합니다.
3. 공급자: unpkg 라이브러리 : @microsoft/signalr@8.0.0-rc.2.23480.2 검색 후 해당 입력 후
4. 특정 파일 선택 후 대상위치를 지정후 설치 클릭
   파일선택 : dist/browser/**signalr.js** 과 **signalr.min.js** 파일 선택
   대상위치: **wwwroot/js/signalr** 로 지정
5. 솔루션 탐색기에 해당 자바스크립트 라이브러리 추가여부 확인
   wwwroot\js\signalr\dist\browser







CDN: https://unpkg.com/@microsoft/signalr@8.0.0-rc.2.23480.2/dist/browser/signalr.js

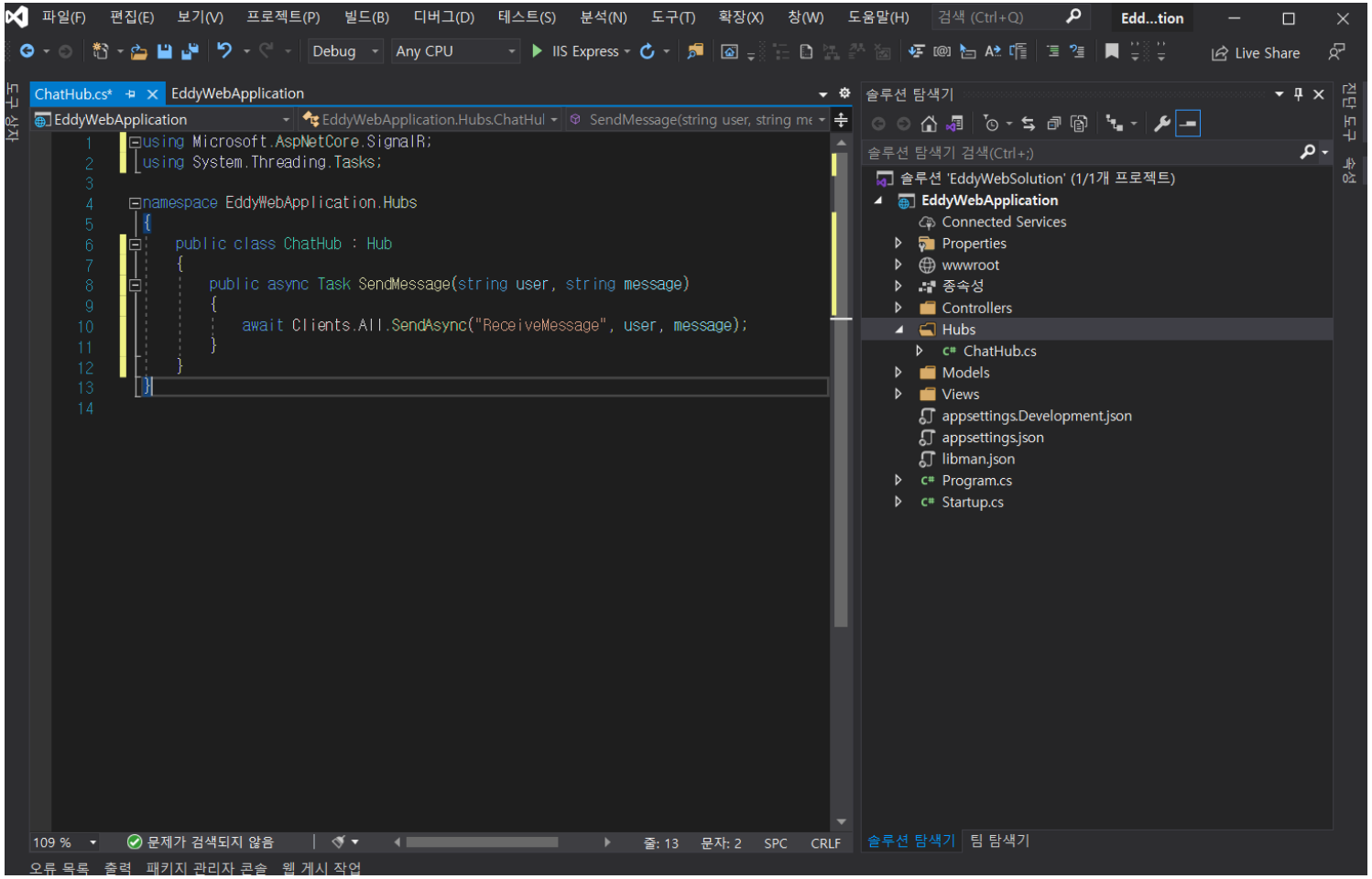
https://learn.microsoft.com/en-us/aspnet/core/signalr/javascript-client?view=aspnetcore-8.0&tabs=visual-studio

ASP.NET Core Signal R 실시간 웹 채팅 개발하기

STEP3: SignalR Hub 클래스 만들기

1. 프로젝트에 오른쪽 마우스 클릭 > 추가> 새폴더
2. 폴더명을 Hubs로 지정합니다.
3. Hubs폴더를 선택하고 오른쪽 마우스 클릭 > 추가 > 클래스
4. 클래스명을 ChatHub.cs로 지정후 추가합니다.
5. ChatHub.cs 에 우측과 같이 코딩을 진행합니다.

```
//참조추가
using Microsoft.AspNetCore.SignalR;
using System.Threading.Tasks;



//Hub 클래스 상속
Public class ChatHub : Hub
{
        //웹브라우저 송신 메시지 수신 및 모든 브라우저에게 재발송
        public async Task SendMessage(string user, string message)
        {
            await Clients.All.SendAsync("ReceiveMessage", user, message);
        }
}
```



https://learn.microsoft.com/en-us/aspnet/core/signalr/hubs?view=aspnetcore-8.0

ASP.NET Core Signal R 실시간 웹 채팅 개발하기

STEP4: SignalR 지원 Project 구성하기

1. 프로젝트 루트내 Programs.cs 열어 SignalR 서비스 환경설정 추가하기

```
//참조
using SignalRBasicApp.Hubs;

var builder = WebApplication.CreateBuilder(args);
builder.Services.AddControllersWithViews();
builder.Services.AddSignalR();

....
app.UseAuthorization();


app.MapControllerRoute(
    name: "default",
    pattern: "{controller=Home}/{action=Index}/{id?}");

//허브클래스 추가
app.MapHub<ChatHub>("/chatHub");
```

https://learn.microsoft.com/en-us/aspnet/core/signalr/hubs?view=aspnetcore-8.0

ASP.NET Core Signal R 실시간 웹 채팅 개발하기

STEP5: 심플 웹 채팅 페이지(뷰) 구성하기

1. 프로젝트내 Views폴더내 Index.cshtml 뷰페이지를 오픈합니다.

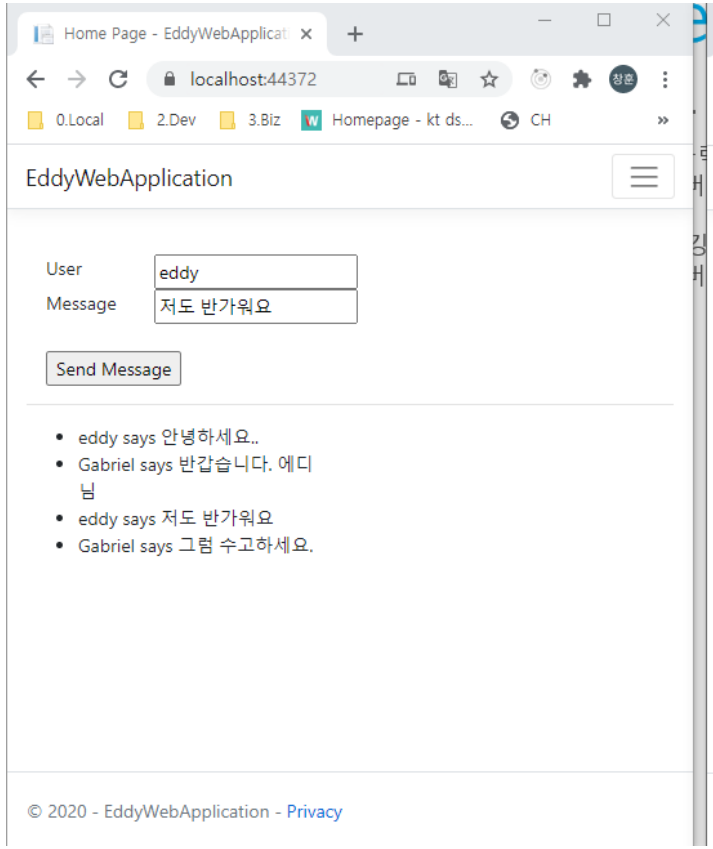
2. 뷰페이지내 모든 내용을 삭제하고 우측 내용을 붙여넣습니다.

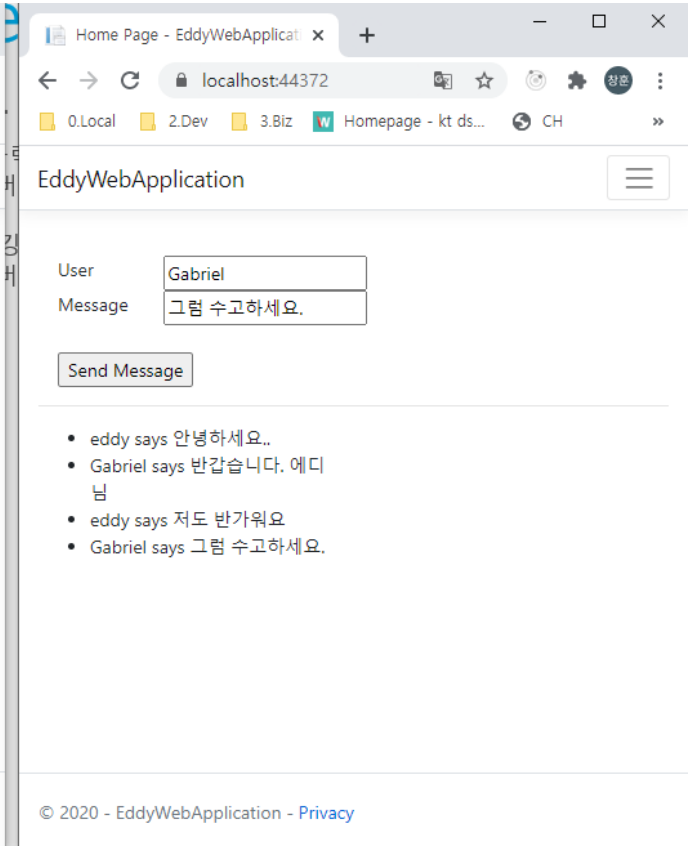
3. 맨 하단의 자바스크립트 라이브러리 참조경로 및 참조 파일을 확인합니다.
   (하기 링크 파일내 관련 소스 복사 붙여넣기)

ASP.NET Core Signal R 실시간 웹 채팅 개발하기

STEP6: 웹 채팅 테스트 하기

1. 상단 메뉴 파일> 모두저장을 클릭합니다.
2. F5 또는 상단 디버그메뉴> 디버깅 시작을 클릭하여 웹 어플리케이션을 디버깅 모드로 실행합니다.
   - F5 또는 디버그 메뉴> 디버깅 시작 클릭
   - 또는 화면내 초록색 디버깅 버튼 클릭
3. 두개의 웹브라우저 창를 오픈하고 웹페이지 주소를 통해 접속합니다. https://localhost:44372/
4. User에 대화명을 Message에는 대화내용을 입력 후 Send버튼을 클릭합니다.
5. 정상적으로 두 브라우저간 채팅이 이루어지면 성공!
6. 그렇지 않으면 질문??





ASP.NET Core Signal R 실시간 웹 채팅 개발하기

STEP7: 그룹채팅/타겟 메시징 처리하기

1. 그룹채팅을 위한 뷰페이지를 하나 추가합니다.
2. 프로젝트내 Controllers폴더내 HomeController.cs 페이지를 오픈합니다.
   - Index액션 메소드를 복사해서 Group 메소드를 만듭니다.
   - Group메소드에 오른쪽 마우스 클릭 뷰추가를 클릭합니다.
   - Razor 뷰를 선택하고 추가를 클릭합니다.

3. Views폴더내 Group.cshtml 뷰 페이지가 추가됨을 확인하고 오픈합니다.
   - Index.cshtml페이지 내용을 복사해 붙여넣기합니다.
4. 그룹채팅 로직을 구현합니다.

ASP.NET Core Signal R 실시간 웹 채팅 개발하기

STEP7: 그룹채팅/타겟 메시징 처리하기

```
public class ChatHub : Hub{

        //웹브라우저 송신 메시지 수신 및 모든 브라우저에게 재발송
        public async Task SendMessage(string user, string message)
        {
            await Clients.All.SendAsync("ReceiveMessage", user, message);
        }

        public async Task JoinGroup(string group, string user)
        {
            //SignalR Hub 채널 그룹 등록
            await Groups.AddToGroupAsync(Context.ConnectionId, group);

            //호출접속자 자신에게 발송
            await Clients.Caller.SendAsync("GroupJoined", $"{group}채팅방에 정상  접속하였습니다");

            //같은 채팅방내 다른 사용자들에게
            await Clients.OthersInGroup(group).SendAsync("GroupJoined", $"{user}님이 {group}채팅방에 입장하셨습니다");

            //특정 사용자에게 메시지 보내기
            //await Clients.Client("ConnectionId").SendAsync("TargetUserMsg", $"특정사용자에게 보내는 메시지입니다.");

        }


        public async Task GroupSendMessage(string group, string user, string message)
        {
            //해당 그룹내 모든 사용자에게 발송
            await Clients.Group(group).SendAsync("GroupReceiveMessage", user, message);
        }

    }
```

ASP.NET Core Signal R 실시간 웹 채팅 개발하기-CORS

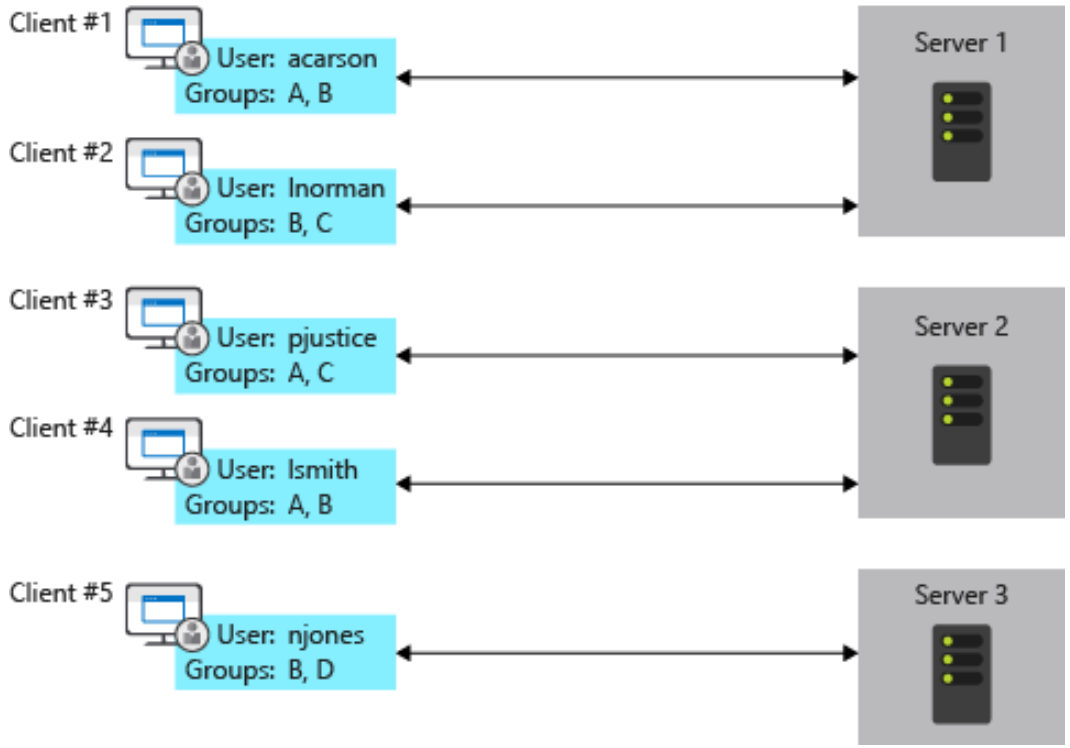
STEP8: 각종 클라이언트 지원 - SignalR 크로스 도메인(CORS) 지원 설정하기

1) CORS 설정하기 – Program.cs

```
builder.Services.AddCors(options =>
{
    options.AddDefaultPolicy(
        builder =>
        {
            builder.WithOrigins("https://localhost:7019/")
                .AllowAnyHeader()
                .WithMethods("GET", "POST")
                .AllowCredentials();
        });
});


...


// UseCors must be called before MapHub.
app.UseCors();
```

ASP.NET Core Signal R 어플리케이션 확장 이슈

- 로드 밸런싱 분산 환경 구성 시 클라이언트 연결정보 유지 어려움
- 웹서버에서 지원할 수 있는 동시 TCP 연결 수 제한 문제
- 모든 클라이언트와의 연결정보를 통합 관리 필요
- 영구적인 서버 연결로 실제 발생 메시징이 적을 경우 불필요한 서버 메모리 낭비
- 동일 서버내에서 제공하는 웹 앱에 영향을 미칠 수 있다.

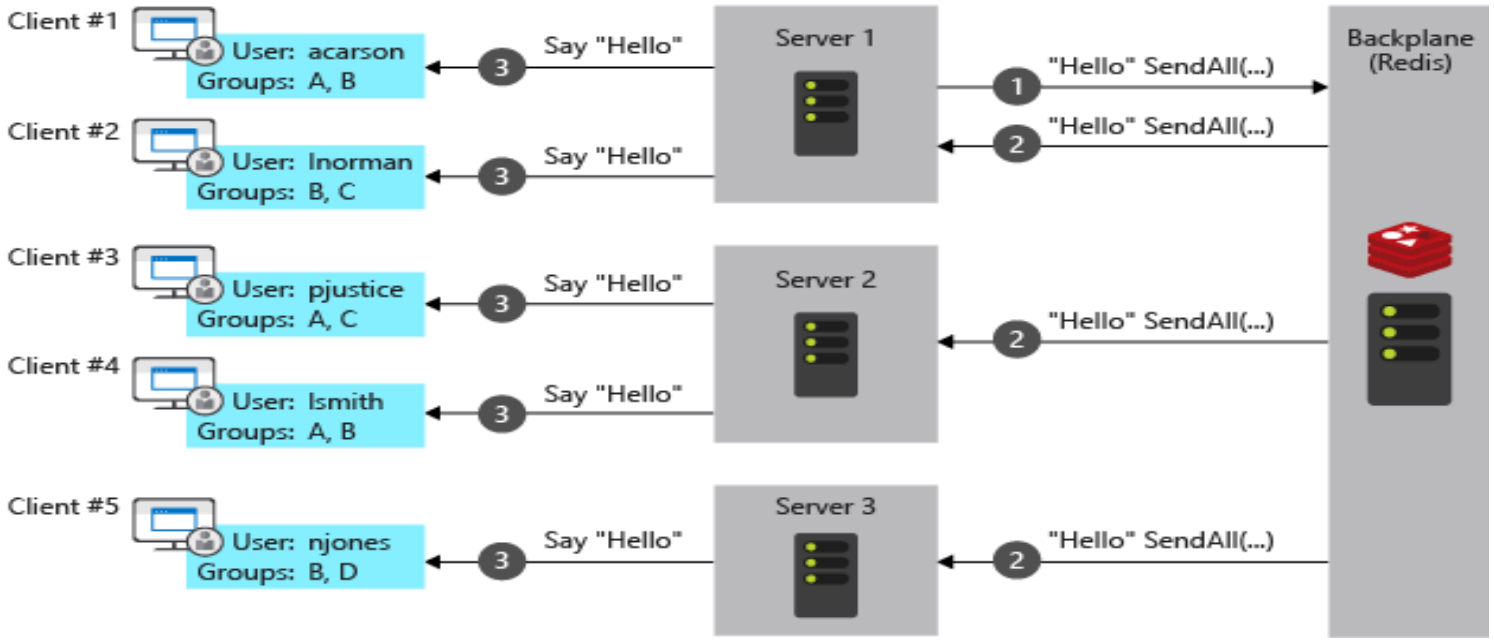


https://learn.microsoft.com/ko-kr/aspnet/signalr/overview/performance/scaleout-in-signalr

ASP.NET Core SignalR - 분산 메시징 기술1

Redis Backplane ?

- 레디스의 게시구독 모델을 이용하여 메시징 시스템을 지원
- 레디스 후면판은 PUB/SUB 기능을 이용해 특정 서버에서 전달된 메시지를 모든 서버에 전달하여 메시징 동기화처리 제공
- 분산서버내 웹앱은 클라이언트에 메시지를 보내기 위해 먼저 레디스 후면판에 메시지를 보낸다.
- 레디스 후면판은 모든 분산서버에게 수신 메시지를 게시한다.
- 모든 분산서버는 레디스 후면판에서 수신한 메시지를 현재 서버에 연결된 모든 클라이언트에 발신한다.



https://learn.microsoft.com/ko-kr/aspnet/core/signalr/redis-backplane?view=aspnetcore-8.0

•Microsoft.AspNetCore.SignalR.StackExchangeRedis

ASP.NET Core SignalR - 분산 메시징 기술 1

Redis Backplane 사용하기

- 윈도우즈용 Redis 설치
  https://github.com/microsoftarchive/redis/releases

- Nuget팩키지설치 7.0.12
  Microsoft.AspNetCore.SignalR.StackExchangeRedis

- 클라이언트 라이브러리 추가
  @microsoft/signalr

-Program.cs

```
//Redis 연결문자열
var connectionString = builder.Configuration.GetConnectionString("RedisNoSQLConnection");

//AddSignalR 추가 for RedisBackplain
//builder.Services.AddSignalR().AddStackExchangeRedis(connectionString);

builder.Services.AddSignalR().AddStackExchangeRedis(connectionString, options => {
      options.Configuration.ChannelPrefix = "SignalRRedisApp";
  });
```

-appsettings.json

```
{
  "Logging": {
    "LogLevel": {
      "Default": "Information",
      "Microsoft.AspNetCore": "Warning"
    }
  },
  "AllowedHosts": "*",
  "ConnectionStrings": {
    "RedisNoSQLConnection": "127.0.0.1,password=eddy524640!,ConnectTimeout=10000"
  }
}
```

ASP.NET Core SignalR - 분산 메시징 기술 1

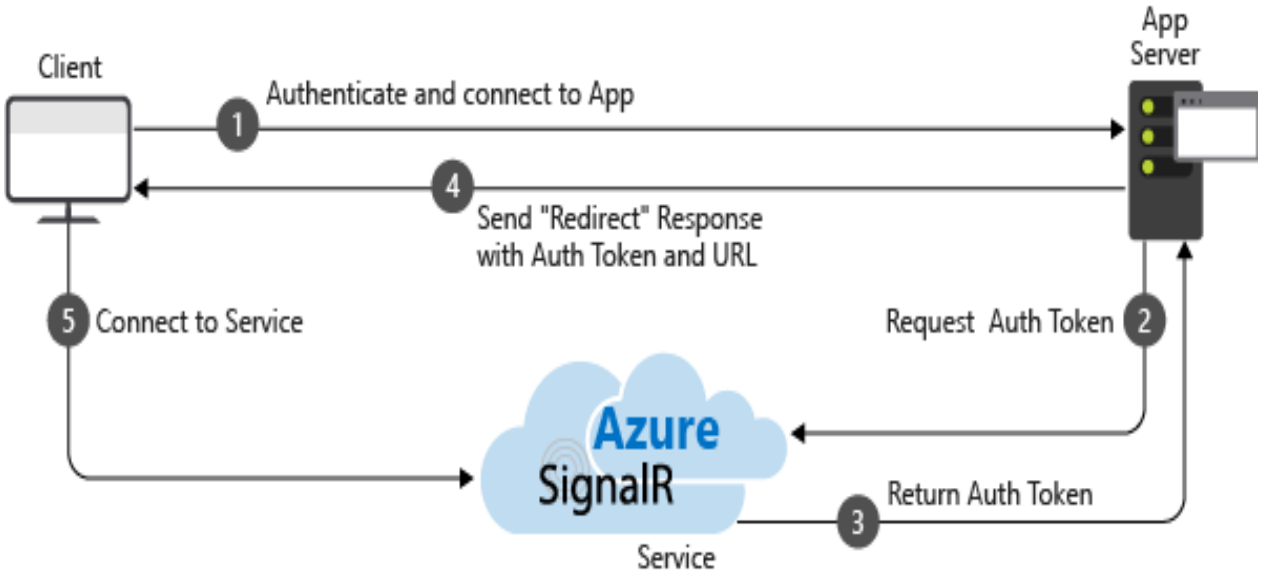
Redis Backplane 특징

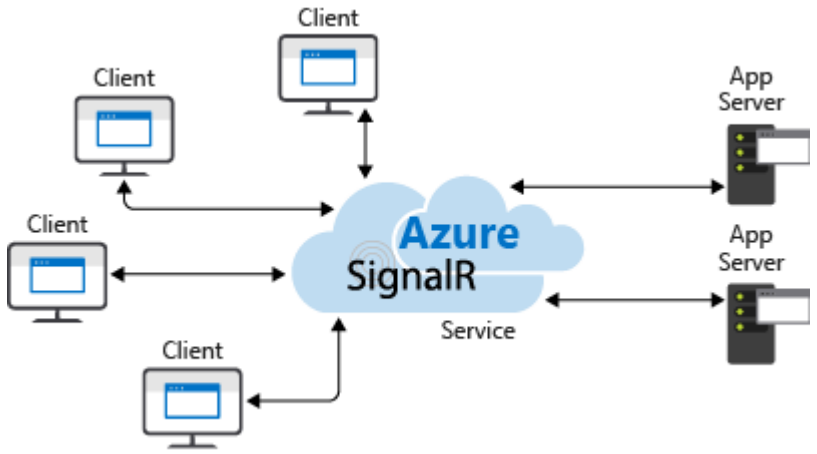
- 온프라미스 환경이나 클라우드 환경에서 독립적인 네트워크망을 구축하여 이용하는 경우
- 모든 클라이언트는 Websocket 만 사용하도록 구성된다.
- 서버 연결 이후 서버와의 지속적인 연결을 유지한다.

ASP.NET Core SignalR - 분산 메시징 기술 2

Azure SignalR Service ?

- 실시간 메시징 지원 Azure 기반 프록시 인프라 환경 지원
- 실시간 웹앱은 클라이언트 연결관리 만 담당(Azure SignalR 서비스 토큰기반)
- 클라이언트간 실시간 메시징 처리를 Azure SignalR 서비스에 전담한다.
- **Azure SignalR 은 자동 크기 조정 기능 제공**





https://learn.microsoft.com/ko-kr/azure/azure-signalr/signalr-overview

ASP.NET Core SignalR - 분산 메시징 기술 2

Azure SignalR Service 사용하기

Azure SignalR Service 개발환경 준비하기

- Azure Portal : 모든서비스 > 웹 및 모바일 > SignalR 서비스 신청 >키 및 엔드포인트 정보 확인
- Nuget 팩키지 설치: **Microsoft.Azure.SignalR 1.27.1**
- 프로젝트에 Azure SignalR Service 접속정보 세팅

-Program.cs

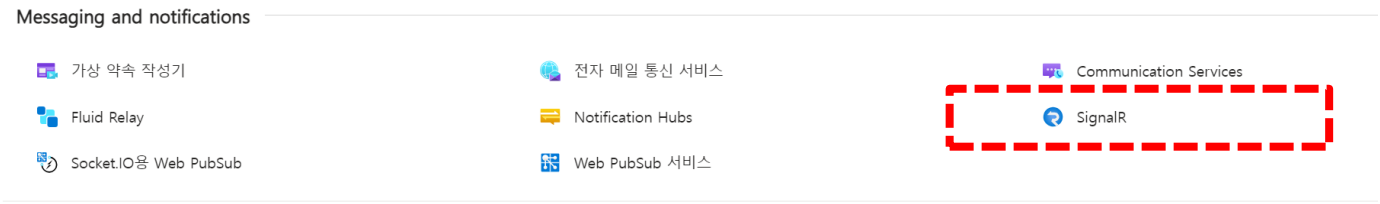
```
//Azure SignalR 서비스 연결문자열
var connectionString = builder.Configuration.GetConnectionString("AzureSignalRConnection");

//AddSignalR 추가 for AzureSignalR Service
builder.Services.AddSignalR().AddAzureSignalR(connectionString);
```

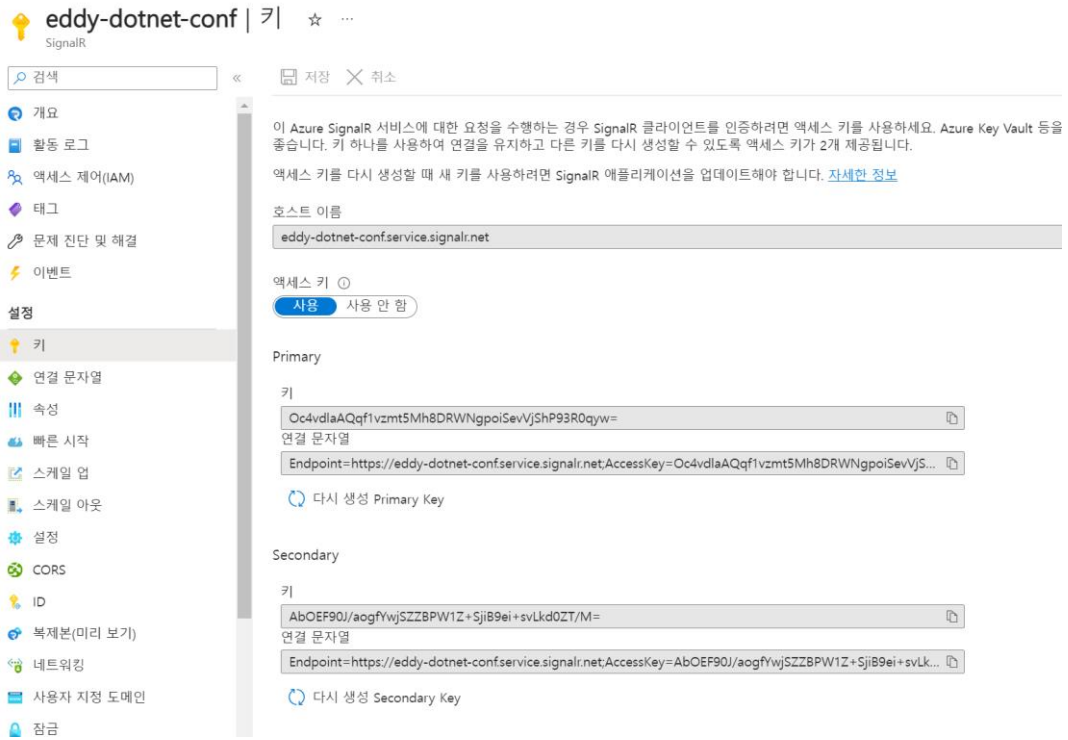
-appsettings.json

```
{
  "Logging": {
    "LogLevel": {
      "Default": "Information",
      "Microsoft.AspNetCore": "Warning"
    }
  },
  "AllowedHosts": "*",
  "AzureSignalRKey": "Oc4vdlaAQqf1vzmt5Mh8DRWNgpoiSevVjShP93R0qyw=",
  "ConnectionStrings": {
    "AzureSignalRConnection": "Endpoint=https://eddy-dotnet-conf.service.signalr.net;AccessKey=Oc4vdlaAQqf1vzmt5Mh8DRWNgpoiSevVjShP93R0qyw=;Version=1.0;"
  }
}
```

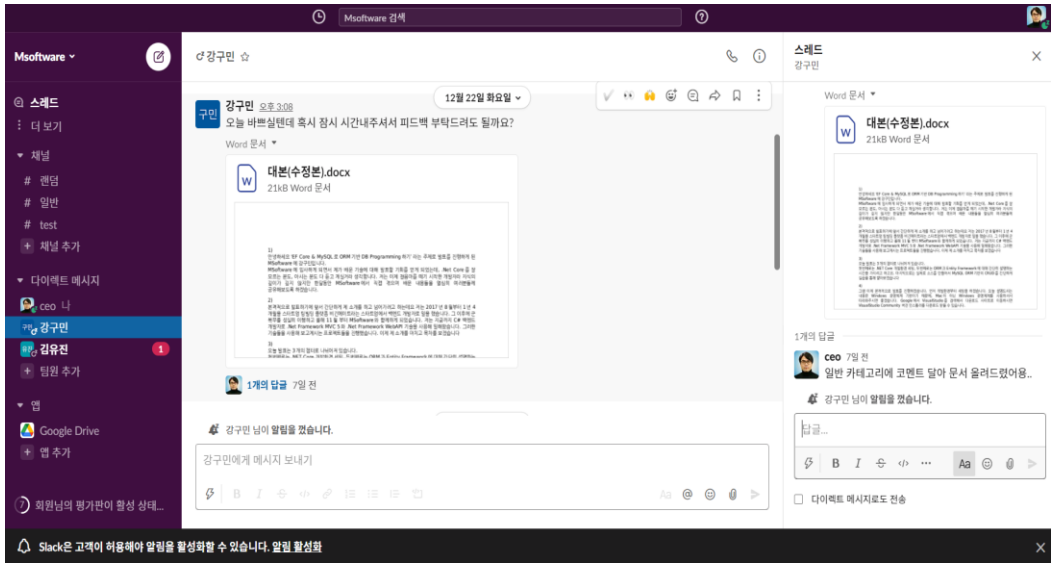
ASP.NET Core SignalR - 분산 메시징 기술 2

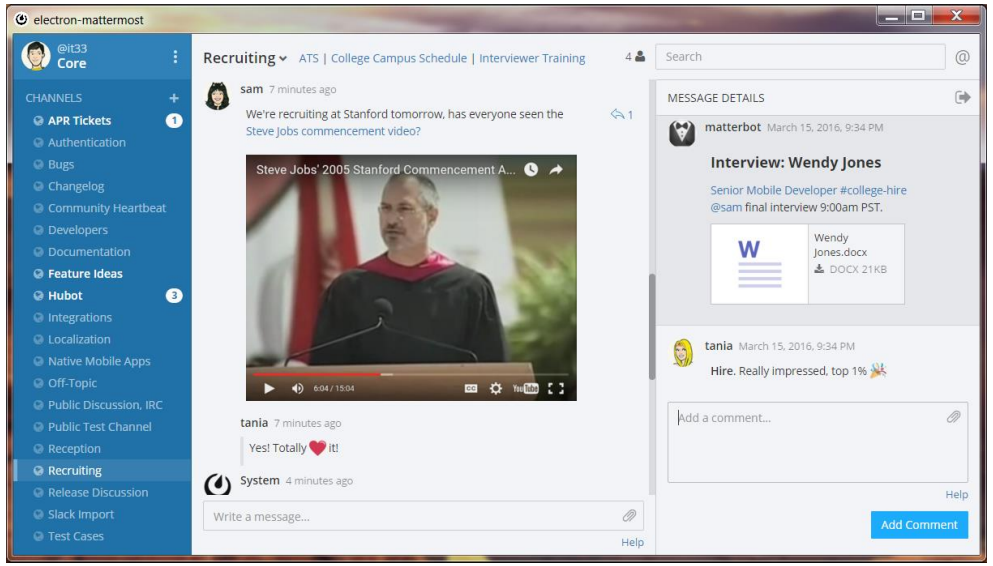
Azure SignalR Service 사용하기

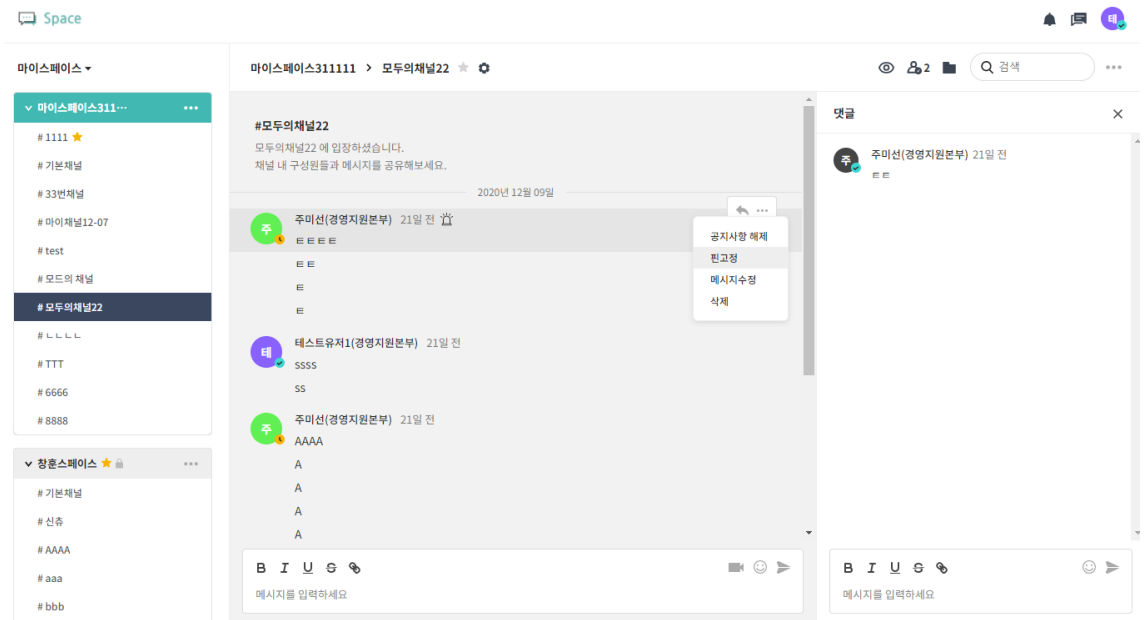




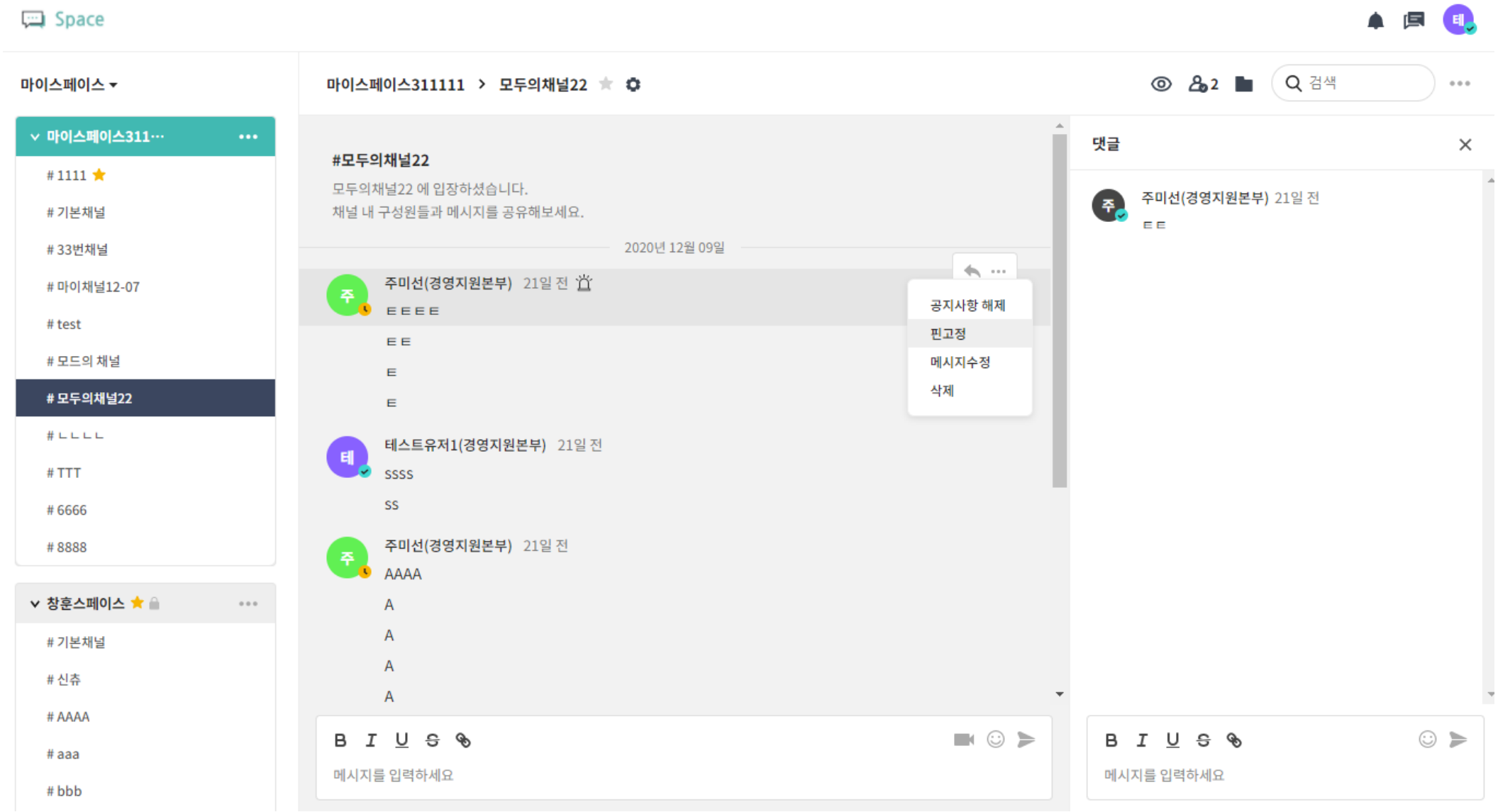
실시간 기술을 활용한 서비스 사례







실시간 기술을 활용한 서비스 사례



ASP.NET Core SignalR Application Architecture

Auth-Token Server

User Authentication

Token Publish

Messaging Web App Service

Messaging Interfaces (Ihttpcontext, Iconfiguration)

Hub Services

Logging Service

Service/Business Library

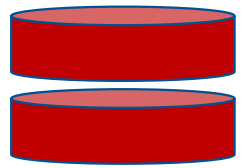
Business Services

Application Exception

Entity Models

DTO

Redis-NoSQL



React.js SPA Web App

React UI Components

Context/Redux

Axios AJAX Library

RESTful API Web App Service

Web API Interfaces

Web API Services

Logging Service

Swagger API Docs

Repository Data Access Library

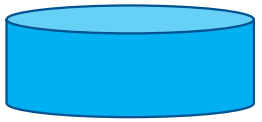
EF Core DBContext

Redis Cache Service

Data Models

SMS,Email Data API Client

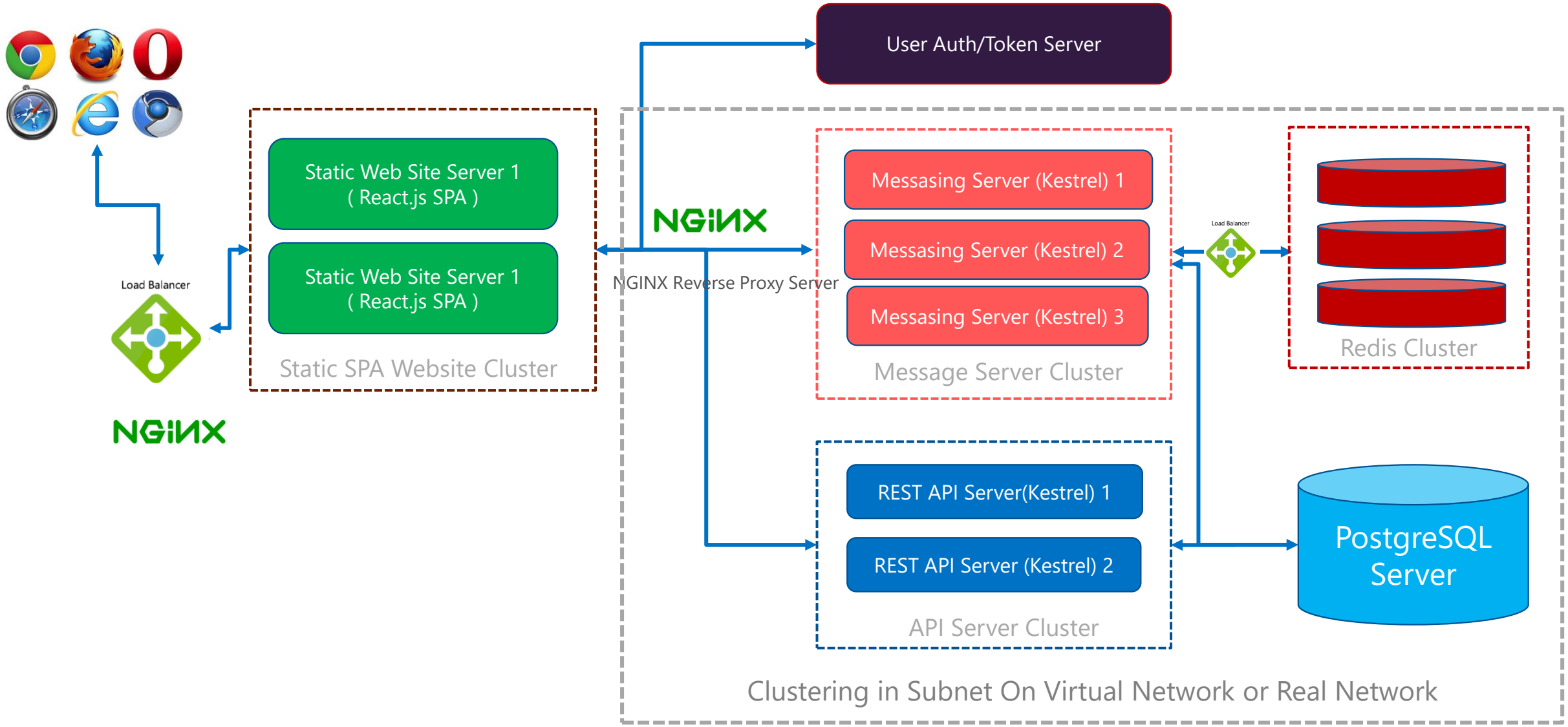
PostgreSQL-RDBMS



Client Side

Server Side(On-Promise/Public Cloud)

Redis Backplain 기반 분산 메시징 시스템 아키텍처



Server Infrastructure (On-Promise or Public Cloud)

Azure SignalR Service 기반 분산 메시징 시스템 아키텍처

Realtime PUB/SUB Message

Microsoft Azure SignalR Service

NGINX

NGINX Reverse Proxy Server

Messasing Server (Kestrel) 1:5000

Messasing Server (Kestrel) 2:5001

Messasing Server (Kestrel) 3:5005

Messasing Server (Kestrel) 4:6005

PostgreSQL Server

Message Server

Azure Cloud Infra

감사합니다.

GitHub Source Address
https://github.com/eddy19740523/EddySignalRSolution

엠소프트웨어 대표 강창훈
https://msoftware.co.kr
ceo@msoftware.co.kr
010-2760-5246