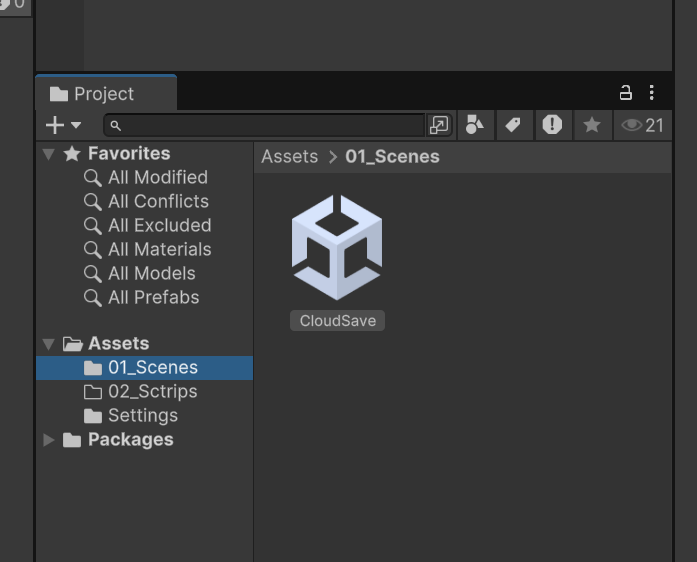
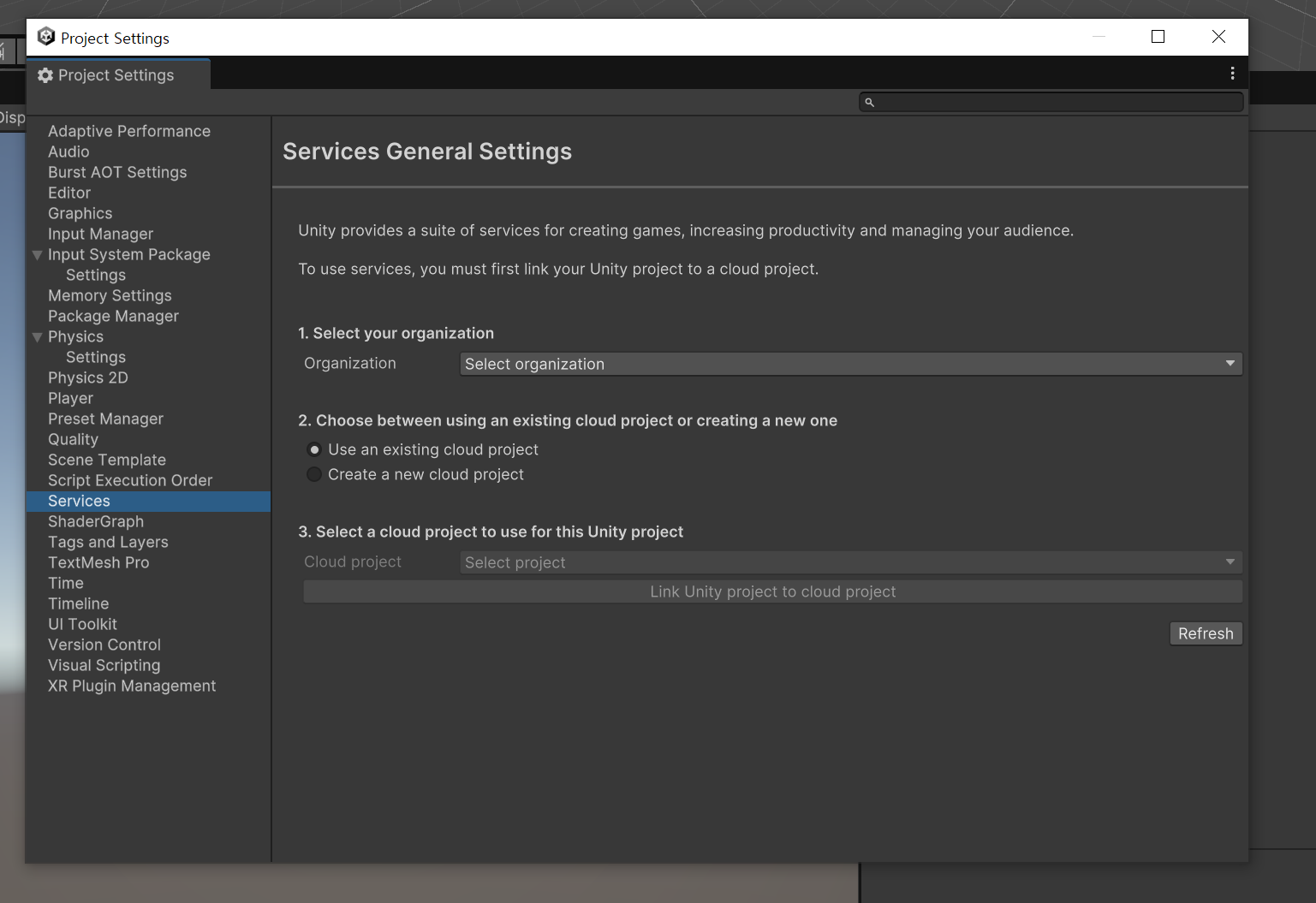
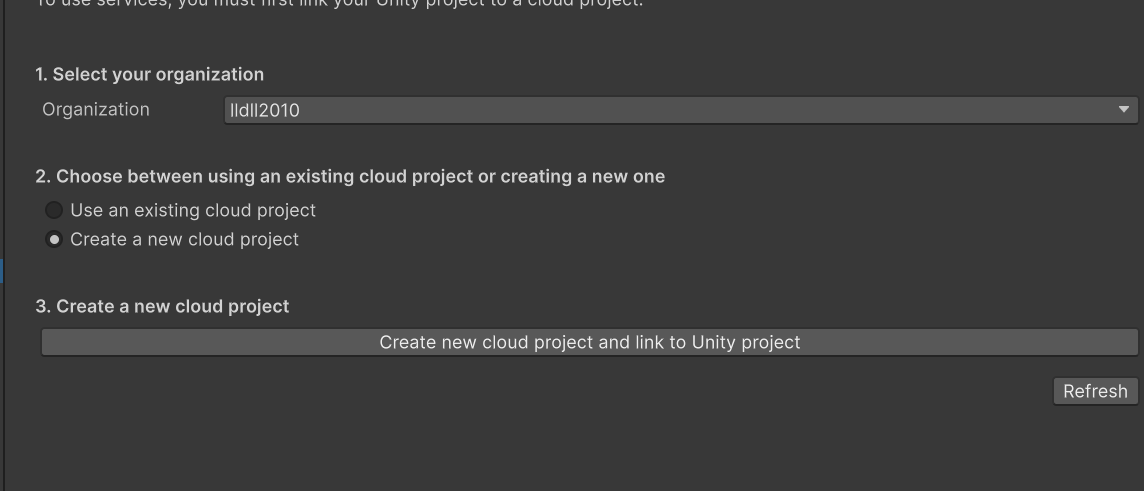
Readme 삭제 tutorial Info 삭제

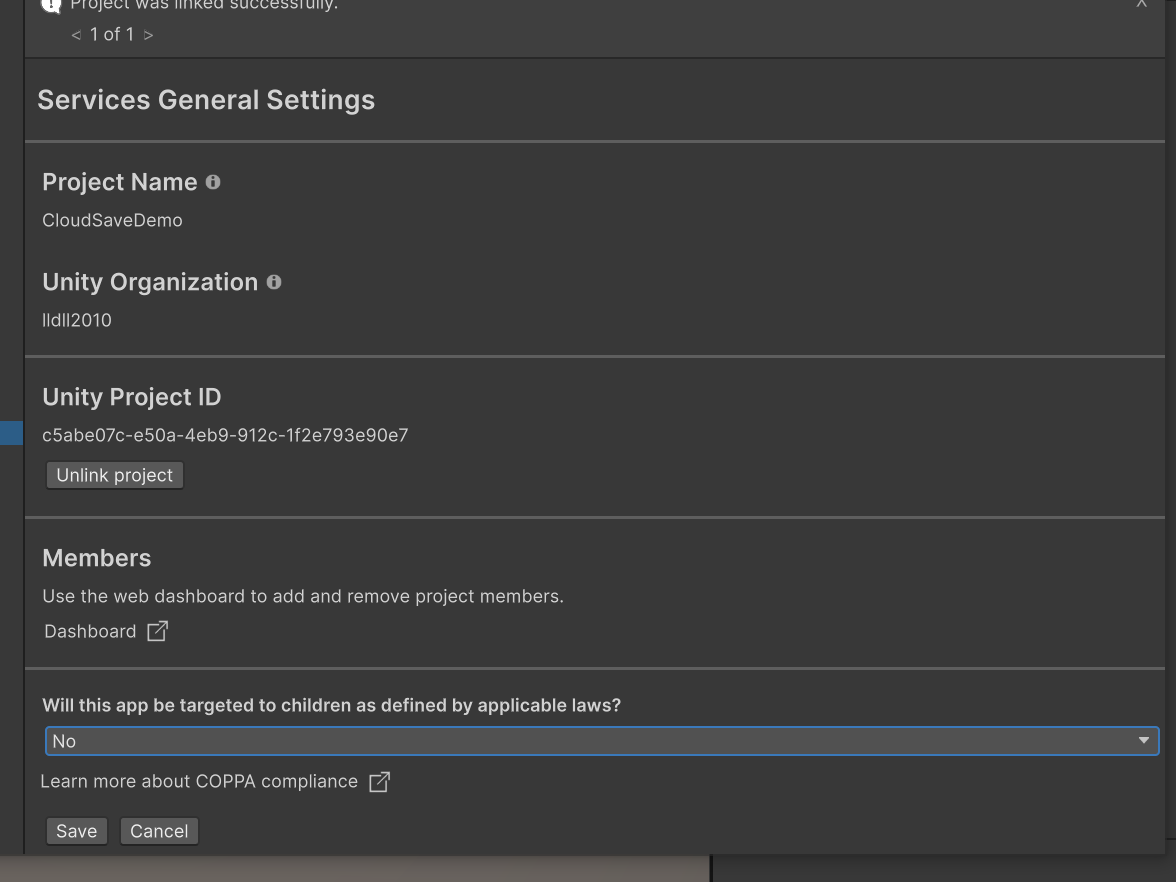
Inputsystem 을 settings로 옮기고 폴더 정리





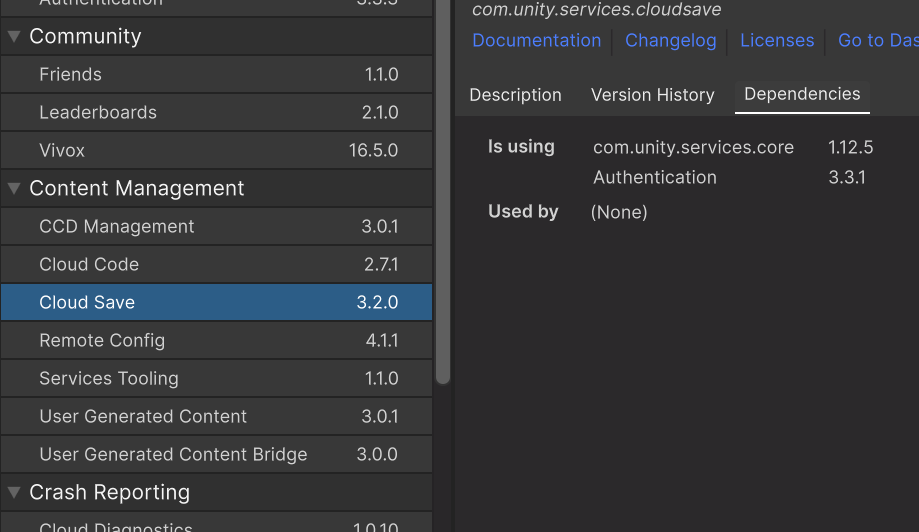
새로 만들겠다 creat a new cloud project,



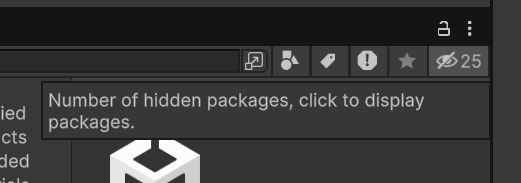


Save

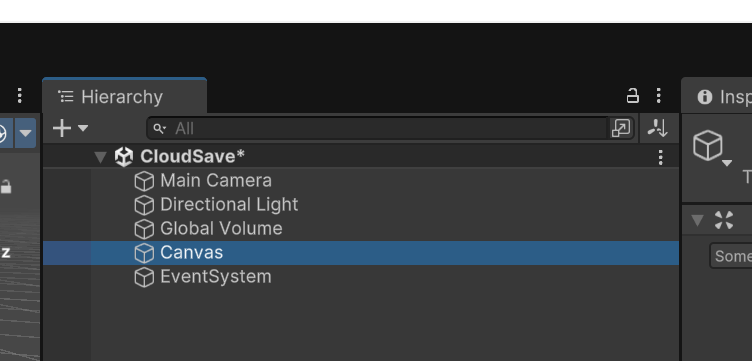
까지 하고 패키지 매니저 열기



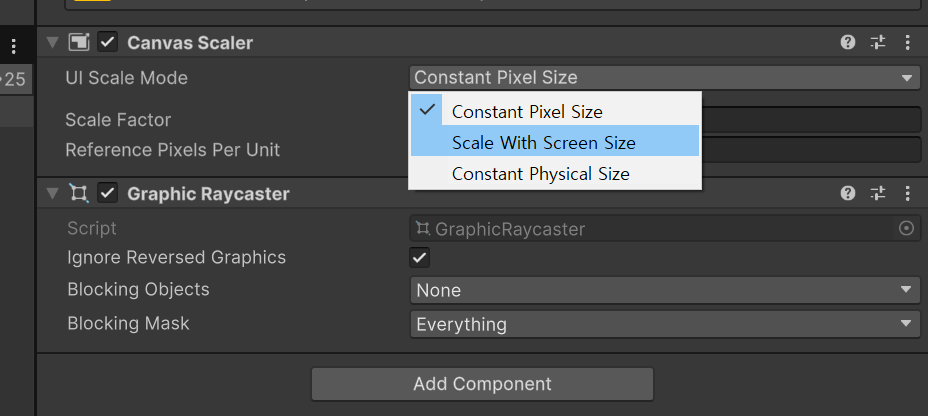
Cloud Save 깔면 Authentication 같이 설치된다. 인증을 해서 로그인 해야 저장할 수 있으니까



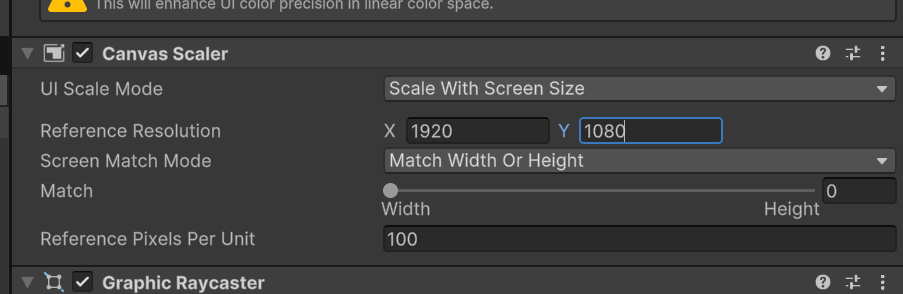
패키지가 너무 많이 보인다 싶으면 눈동자 눌러서 끌 수도 있다

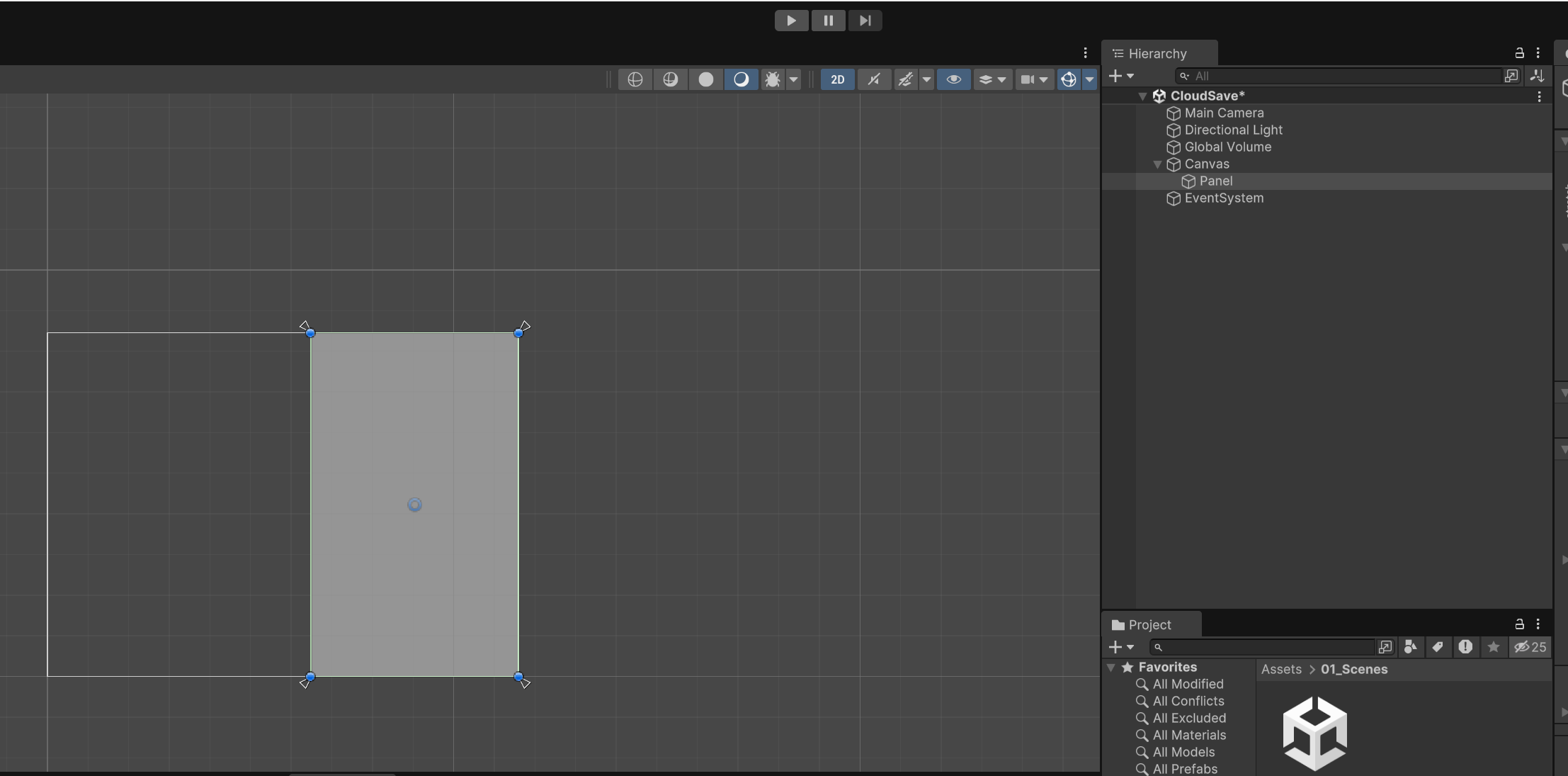


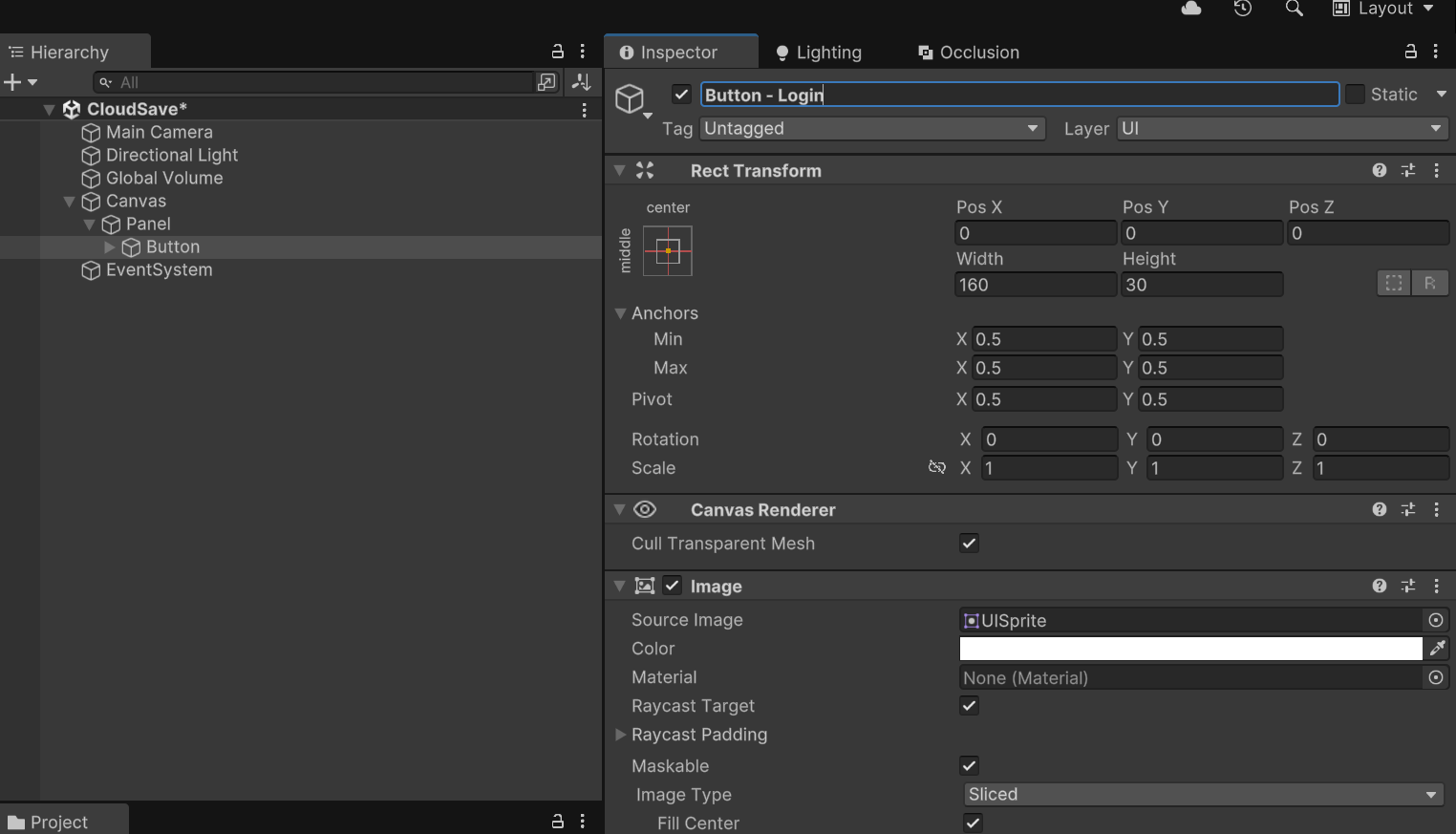
캔버스 추가

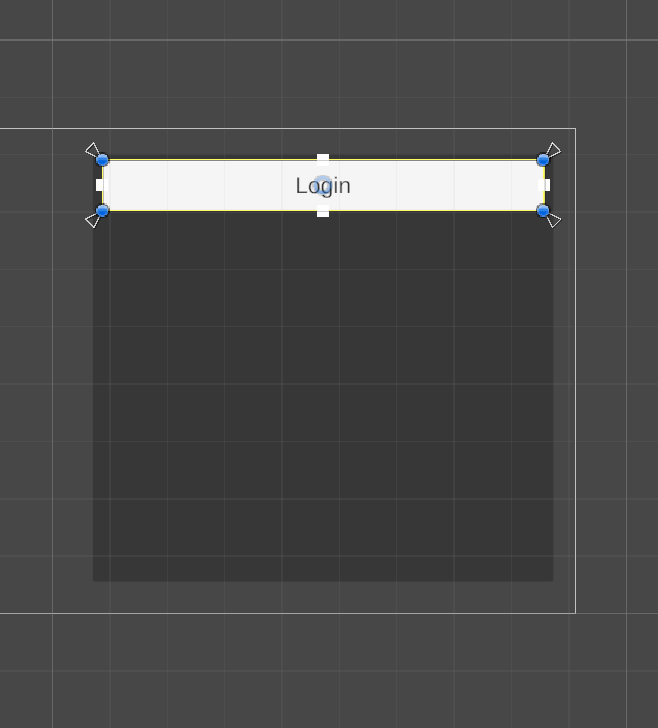


캔버스 추가하면 필수적으로 먼저 Scale with screen size

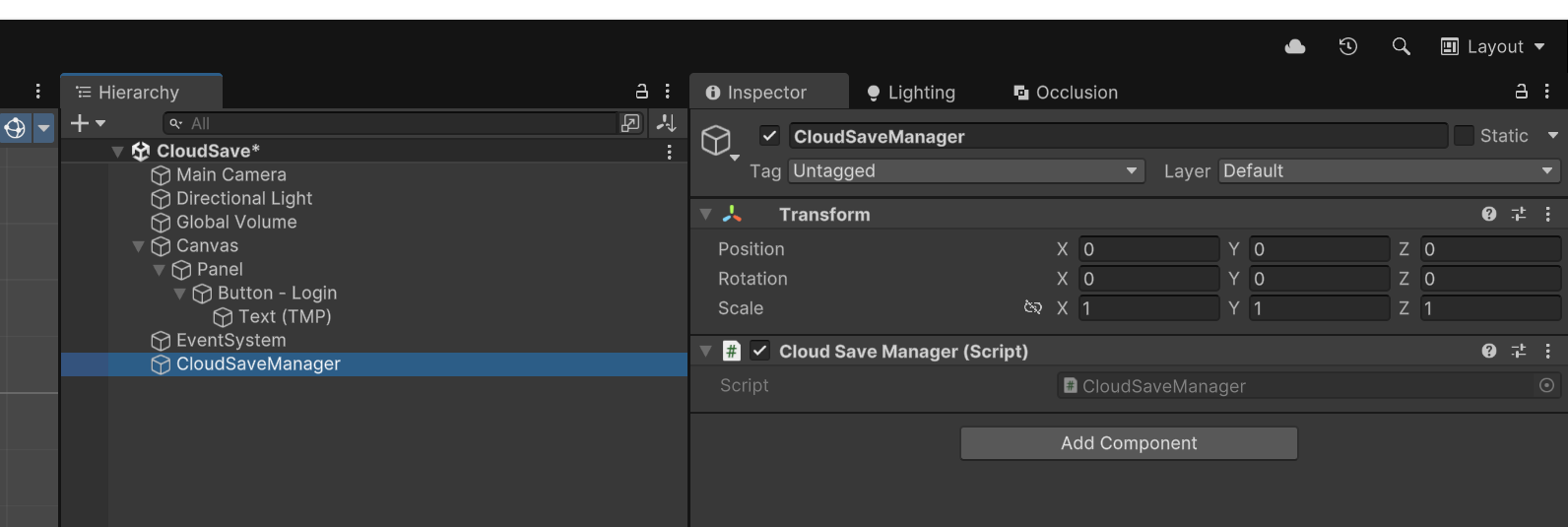


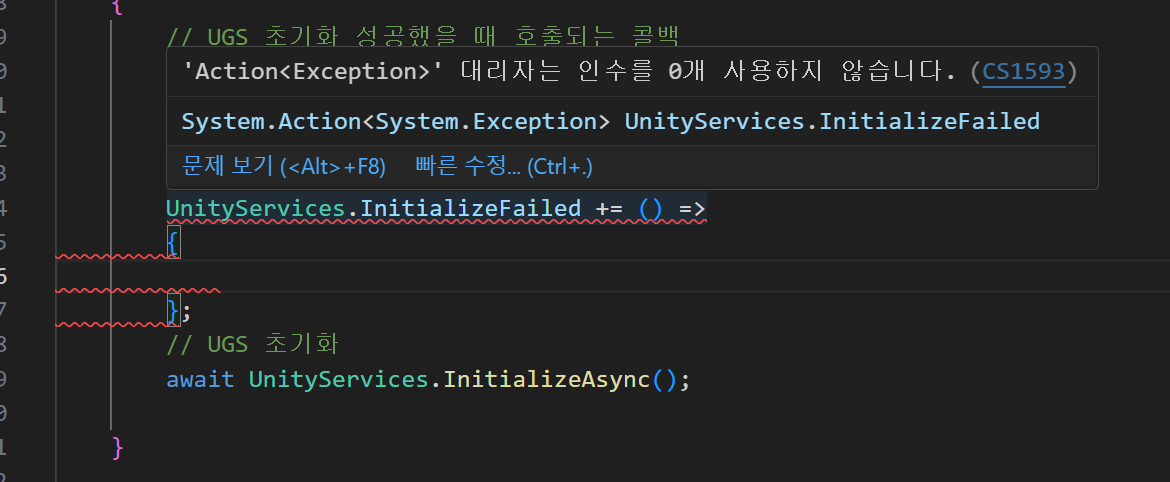
패널 추가, 앵커포인트 조절





이후 CloudSaveManager 빈게임오브젝트, 스크립트 생성





Failed 는 파라미터가 필요하다

여기까지 코드

public class CloudSaveManager : MonoBehaviour

{

    [SerializeField] Button signInButton;

    private async void Awake()

    {

        // UGS 초기화 성공했을 때 호출되는 콜백

        UnityServices.Initialized += () =>

        {

            Debug.Log("UGS 초기화 성공");

        };

        UnityServices.InitializeFailed += (ex) =>

        {

            Debug.Log($"UGS 초기화 실패 : {ex.Message}");

        };

        // UGS 초기화

        await UnityServices.InitializeAsync();

        // 익명 사용자로 로그인 성공했을 때 호출되는 콜백

        AuthenticationService.Instance.SignedIn += () =>

        {

            string playerId = AuthenticationService.Instance.PlayerId;

            Debug.Log($"익명 로그인 성공\nPlayerId: <color=#00ff00>{playerId}</color>");

        };

        signInButton.onClick.AddListener(async () =>

        {

            await SignIn();

        });

        // 익명 로그인

    }

    private async Task SignIn()

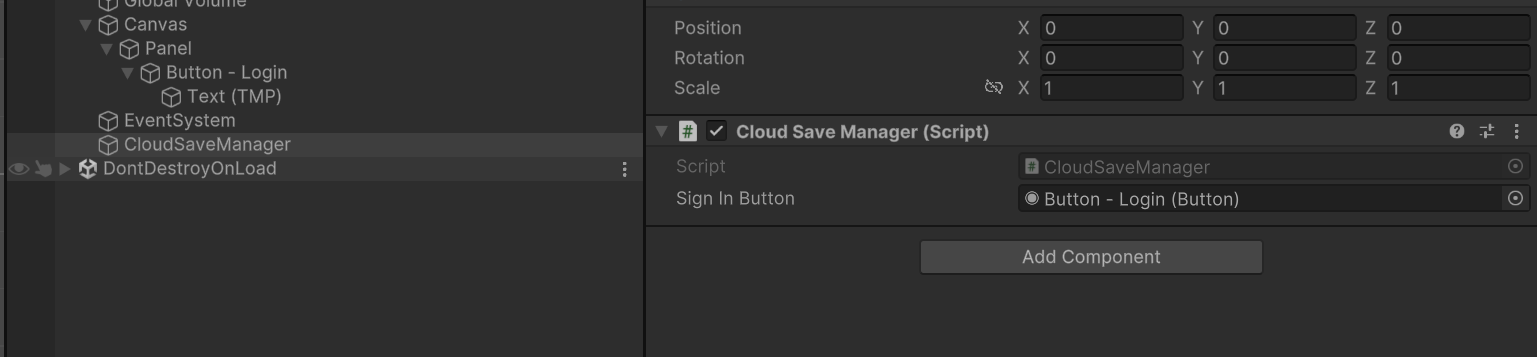
    {

        await AuthenticationService.Instance.SignInAnonymouslyAsync();

    }

}

버튼 만들려면 뭘 해야할지 모르겠다?

1. 버튼 변수 선언 변수(Button) 이름(signInButton)(
2. 유니티 상에서 버튼 연결해주기
3. 
4. // 로그인 버튼 클릭 이벤트 연결
5. signInButton.onClick.AddListener(async () =>
6. {
7. await SignIn();
8. });

SaveSingleData 도 똑같이 만들어서 연결

    // 단일 데이터 저장

    private async Task SaveSingleData()

    {

        // 저장할 데이터 선언

        // 원래 var 자리에 Dictionary<string, object> 가 맞지만 뒤에 명확하게 써있기 때문에 var 로 대체

        var data = new Dictionary<string, object>

        {

            {"Player Name", "DEBE"},

            {"Level", 30},

            {"Exp", 2000},

            {"gold", 100}

        };

        // Cloud 저장

        // 파라미터를 Dictionary 형식으로 넣어야 하는데 위에서 data 로 저장해뒀으니 (data)로 작성 가능

        await CloudSaveService.Instance.Data.Player.SaveAsync(data);

        Debug.Log("싱글 데이터 저장 완료");

    }

버튼 제작

    [SerializeField] Button saveSingleDataButton;

        // 싱글 데이터 버튼 클릭 이벤트 연결

        saveSingleDataButton.onClick.AddListener(async () =>

        {

            await SaveSingleData();

        });

playerData 랑 itemData 넣을 struct 작성

[System.Serializable] // 클래스, struct 직렬화 할 때 사용함

public struct PlayerData

{

    public string name;

    public int level;

    public int exp;

    public int gold;

    public List<ItemData> items;

}

[Serializable]

public struct ItemData

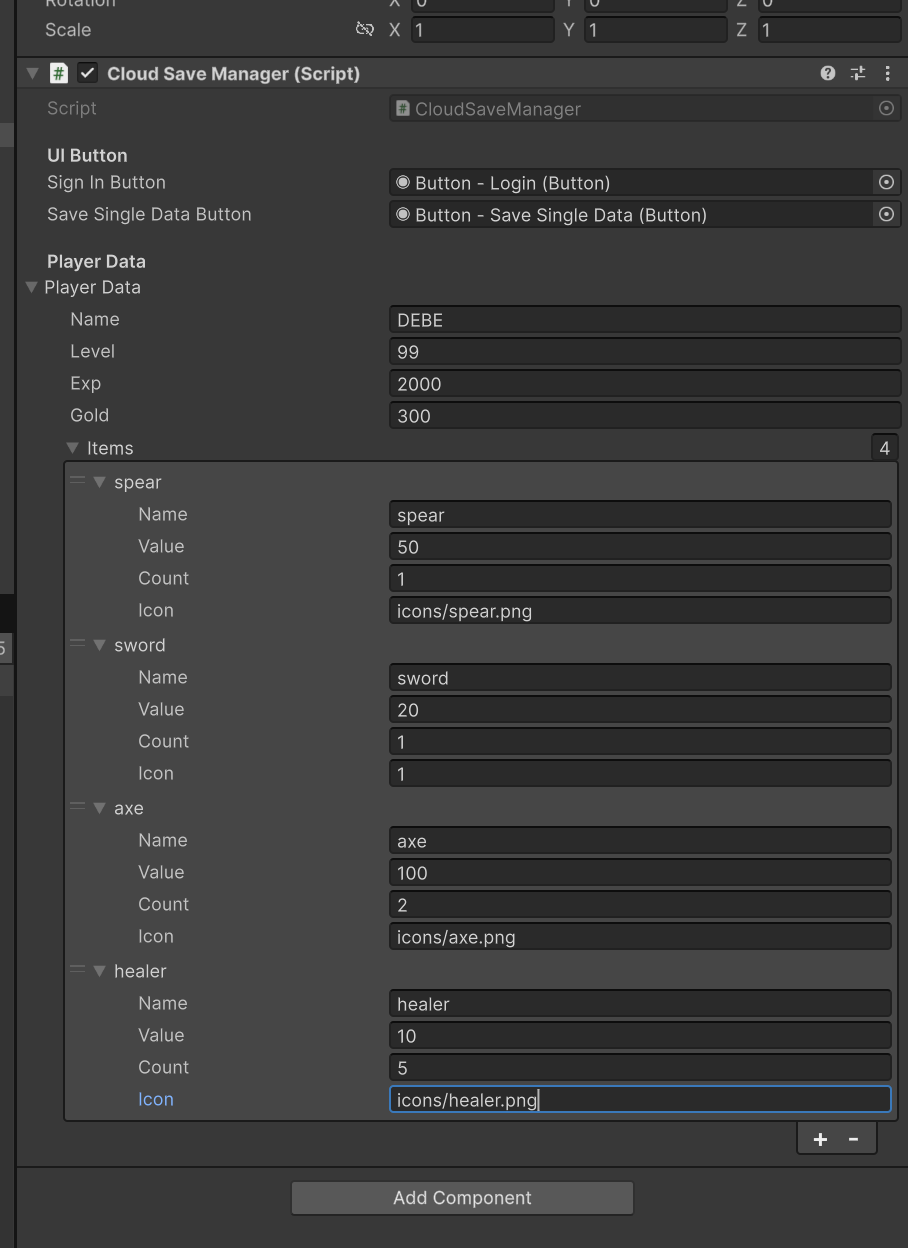
{

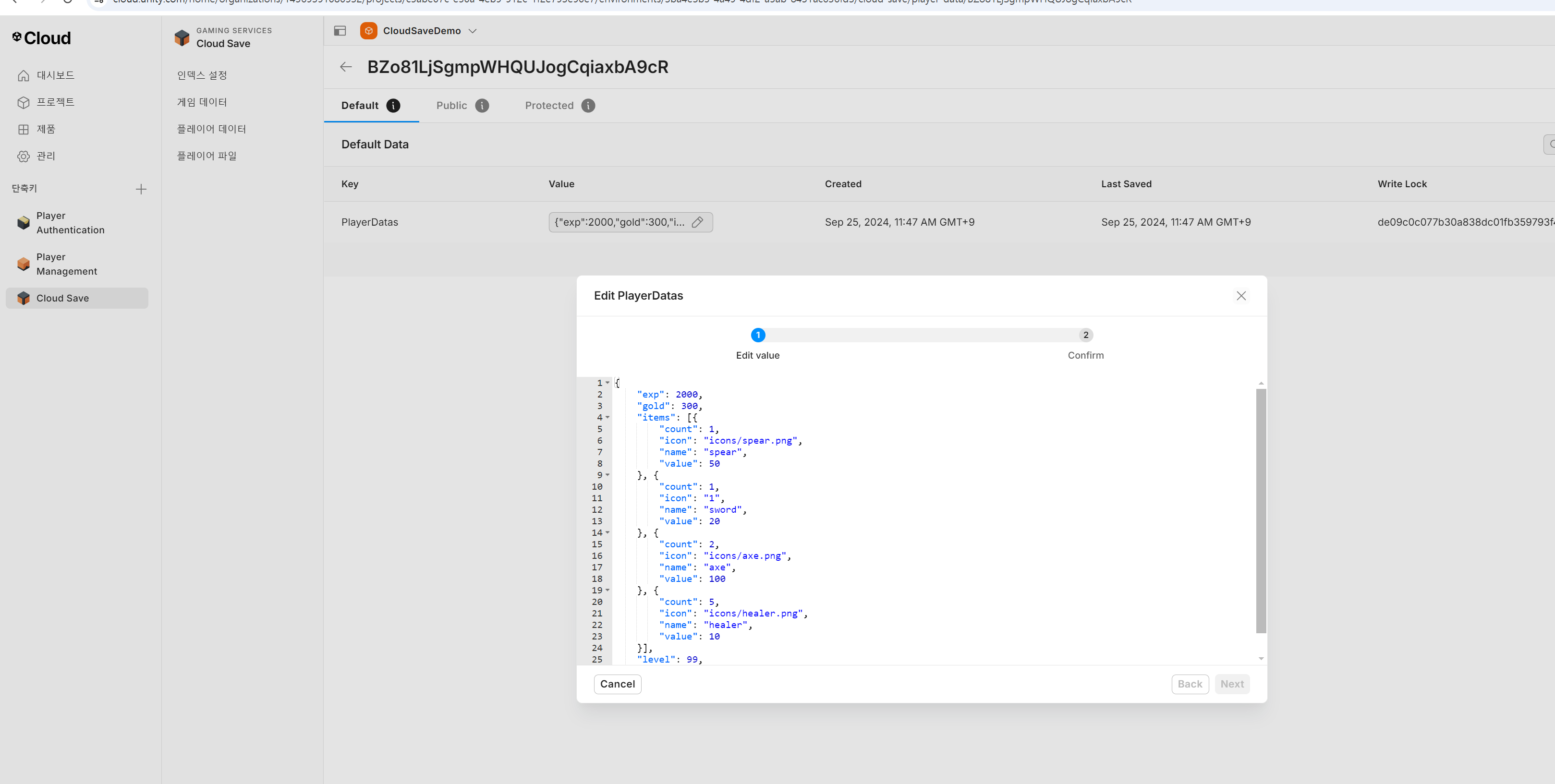
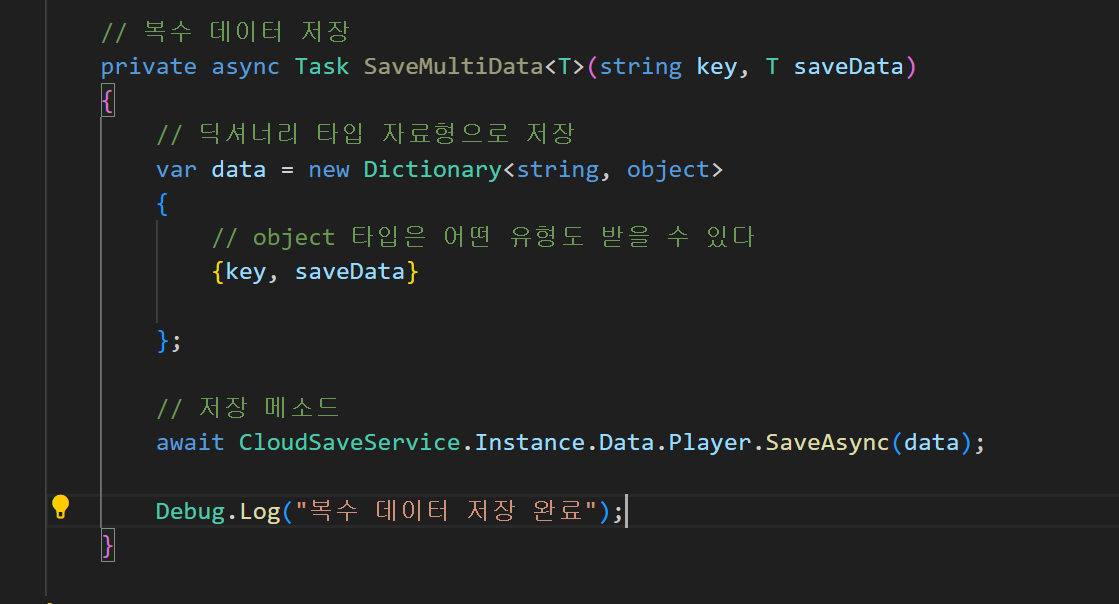
    public string name;

    public int value;

    public string icon;

}





################데이터 로드###################

    // 싱글 데이터 로드

    /\*

        HashSet<T> 자료형

        - 중복 값을 허용하지 않음

        - 검색 속도가 빠름(Hash 기반이라서 0, 1)

        - 인덱스 기반으로 값을 추출할 수 없음. TryGetValue 사용

        List<int> items = new List<int>();

        ....

        items[21]

    \*/

    private async Task LoadSingleData()

    {

        /\* 원론적인 방법

        var fields = new HashSet<string>();

        fields.Add("PlayerName");

        fields.Add("level");

        fields.Add("exp");

        fields.Add("gold");

        \*/

        // new 키워드로 초기화를 하면서 초기값을 설정하는 방법

        var fields = new HashSet<string>

        {

            "PlayerName", "level", "exp", "gold"

        };

        var data = await CloudSaveService.Instance.Data.Player.LoadAsync(fields);

        if (data.TryGetValue("PlayerName", out var playerName))

        { // PlayerName 값이 있으면 ~ playerName 을 반환한다.

            Debug.Log($"PlayerName : {playerName.Value.GetAs<string>()}");

        }

        if (data.TryGetValue("level", out var level))

        { // level 값이 있으면 ~ level 을 반환한다.

            Debug.Log($"Player Lv : {level.Value.GetAs<int>()}");

        }

        if (data.TryGetValue("gold", out var gold))

        { // gold 값이 있으면 ~ gold 을 반환한다.

            Debug.Log($"Gold : {gold.Value.GetAs<int>()}");

        }

        if (data.TryGetValue("exp", out var exp))

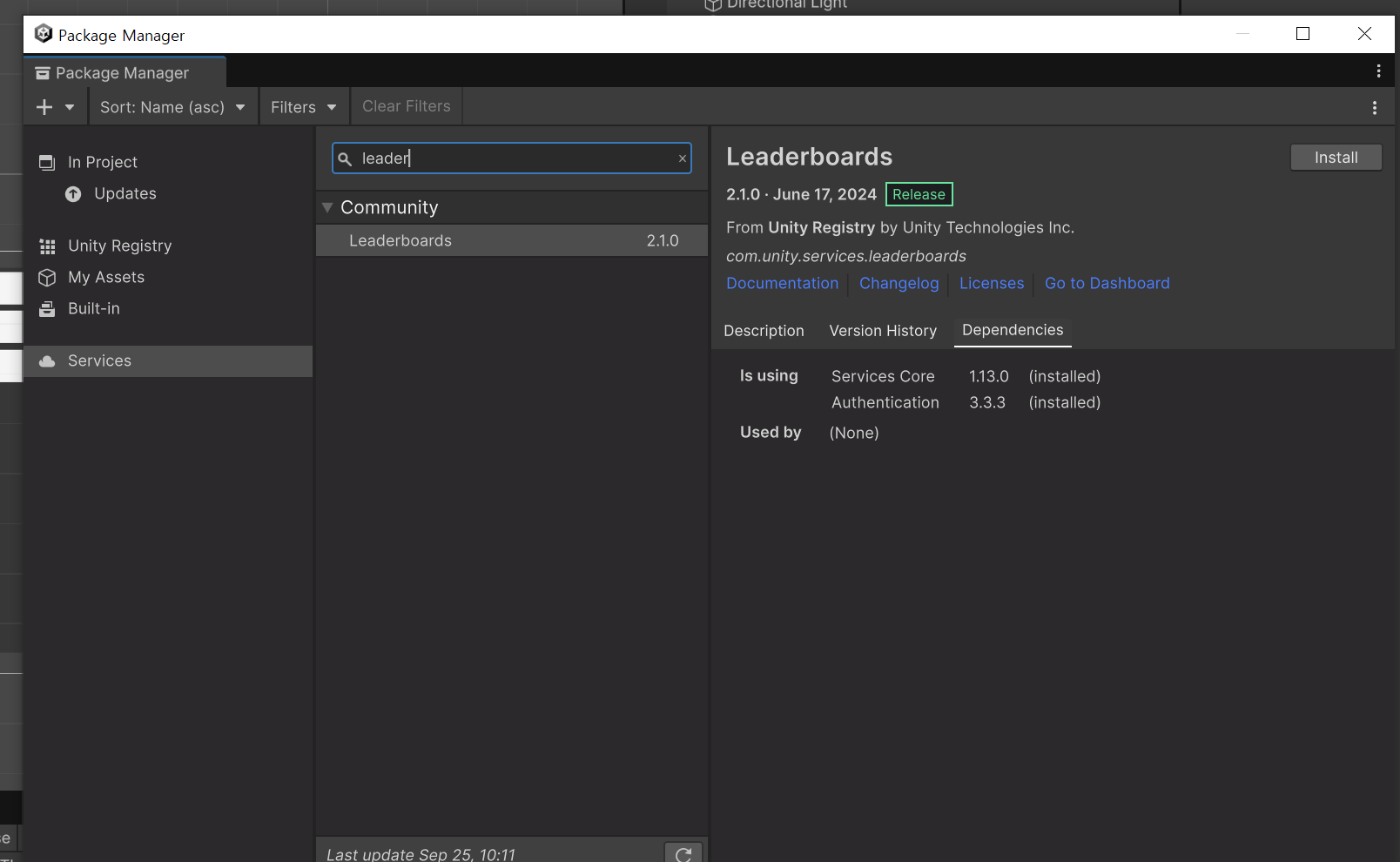
        { // exp 값이 있으면 ~ exp 을 반환한다.

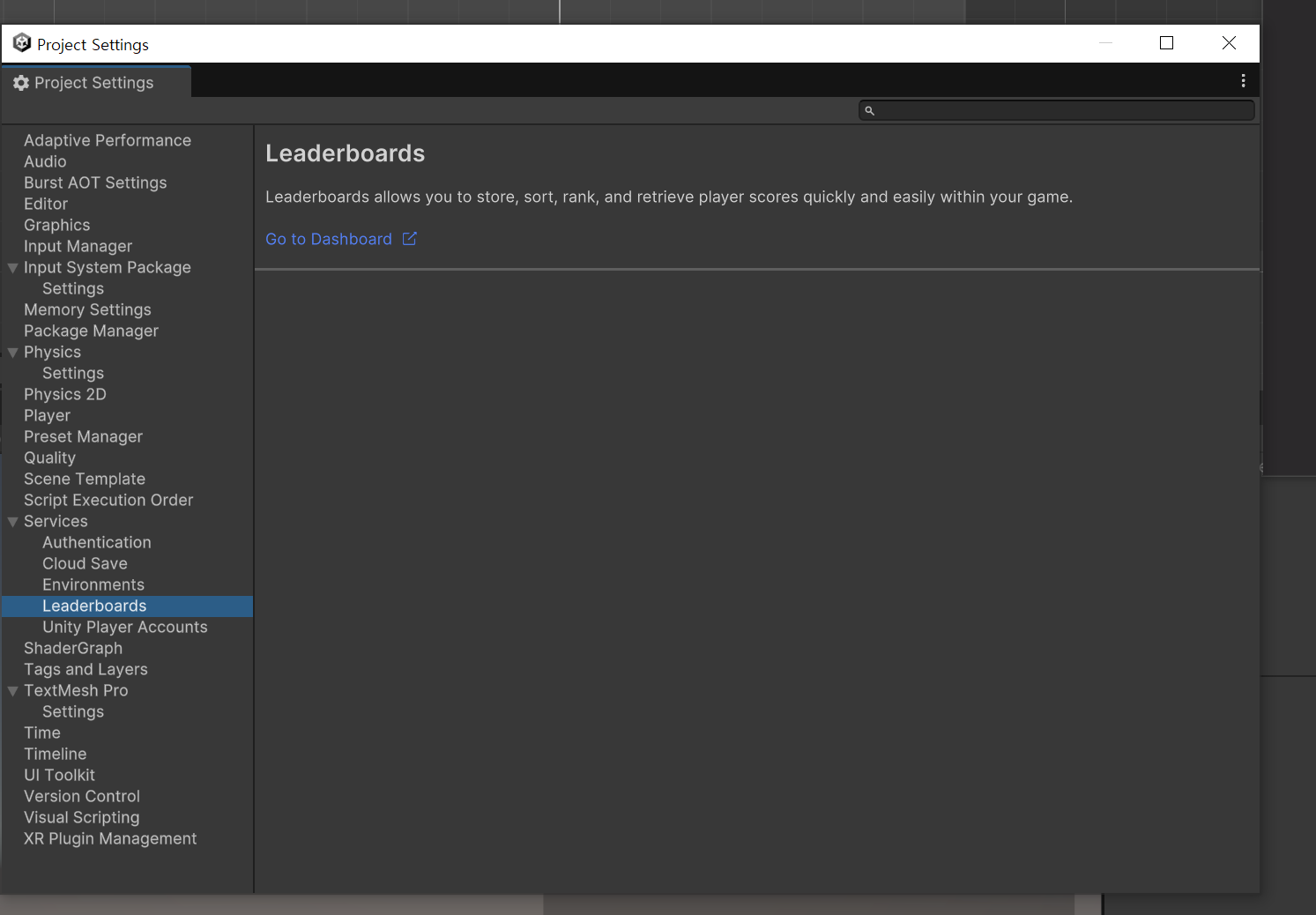
            Debug.Log($"Player EXP : {exp.Value.GetAs<int>()}");

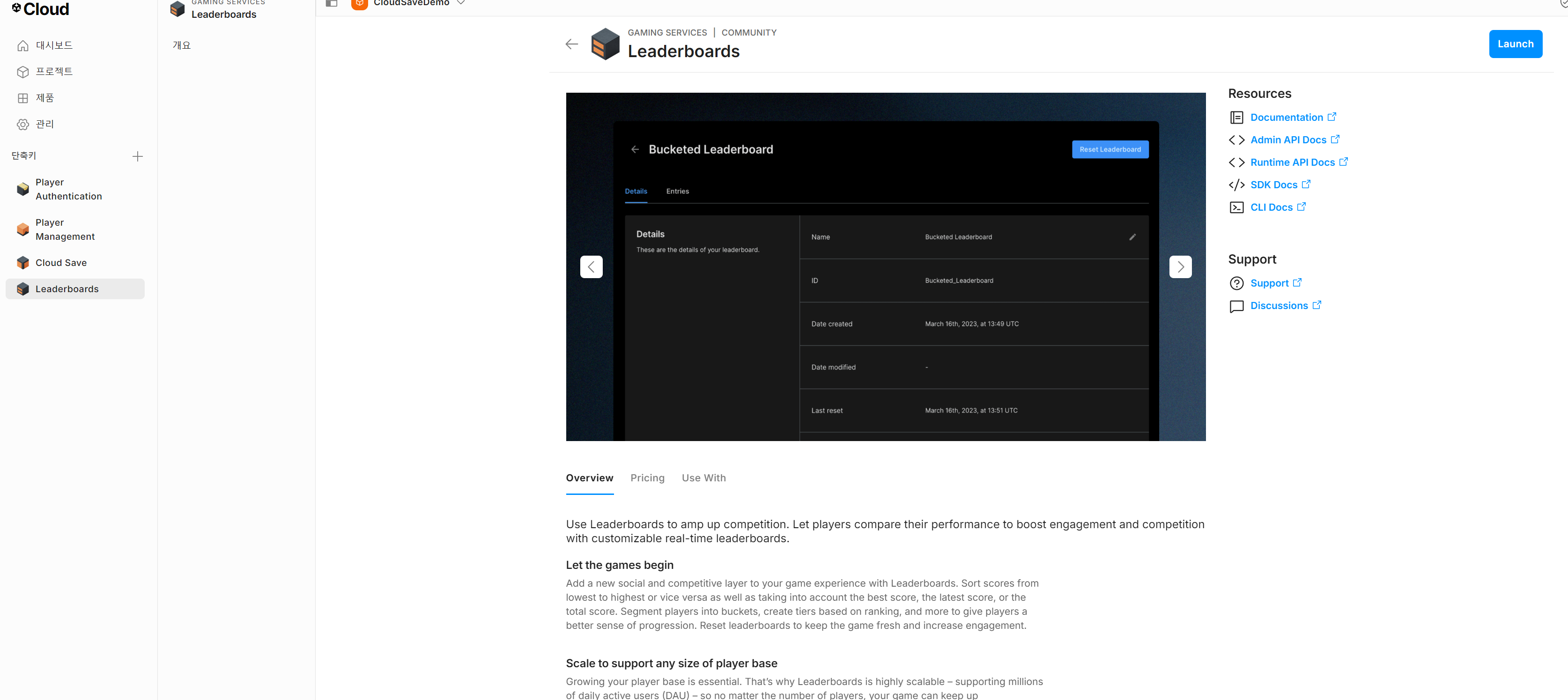
        }

    }

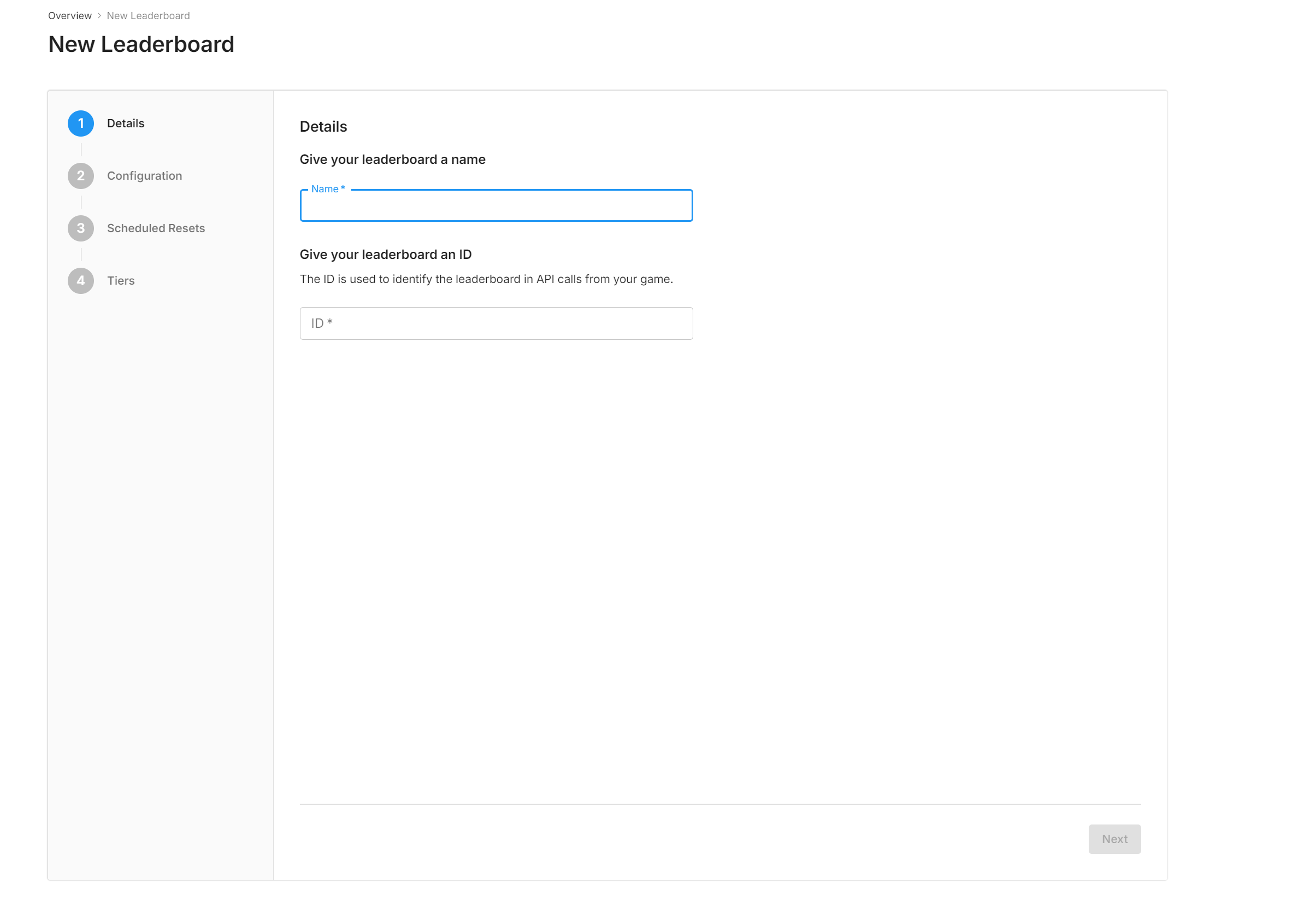
############################### 레더 보드 ##########################



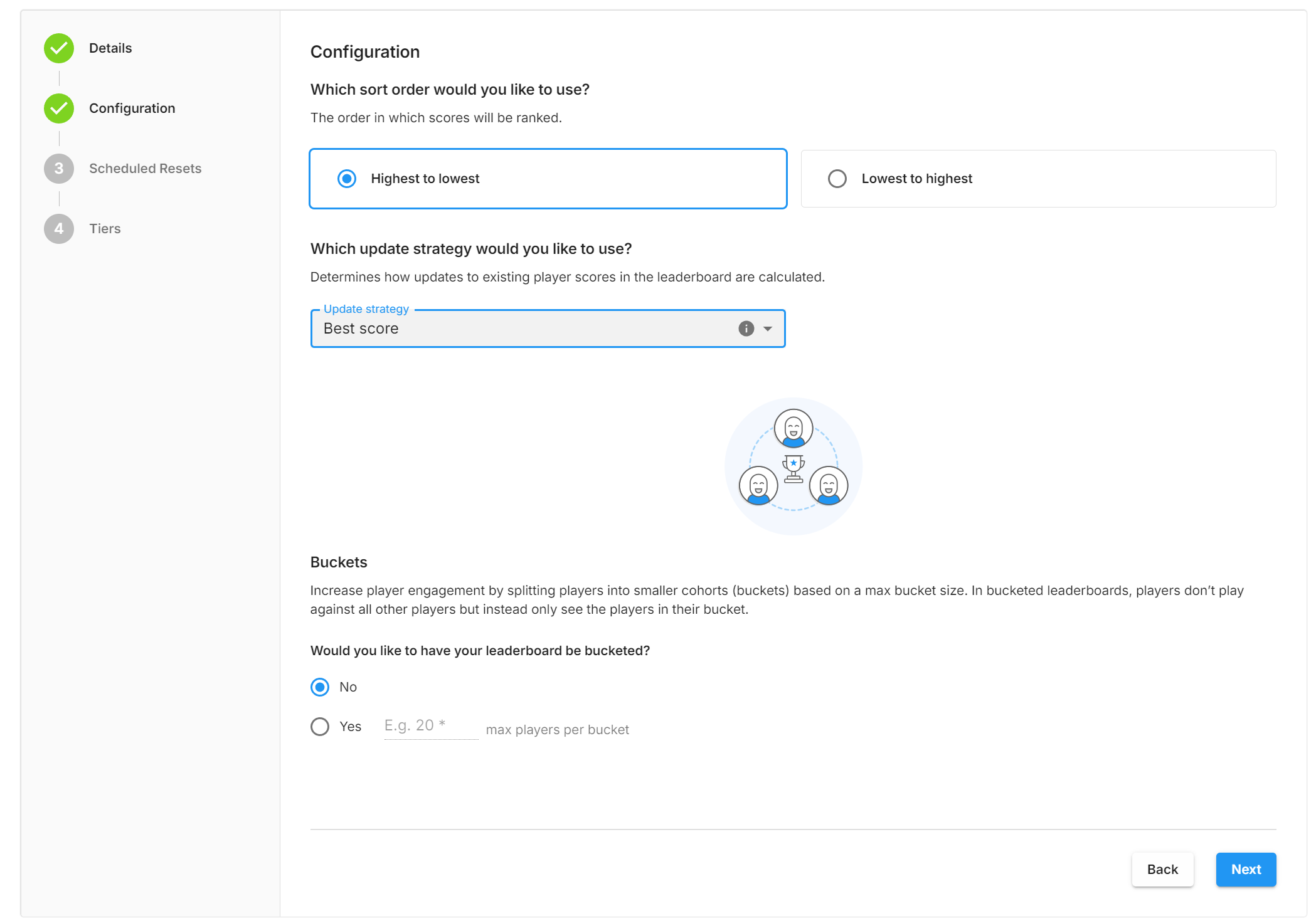




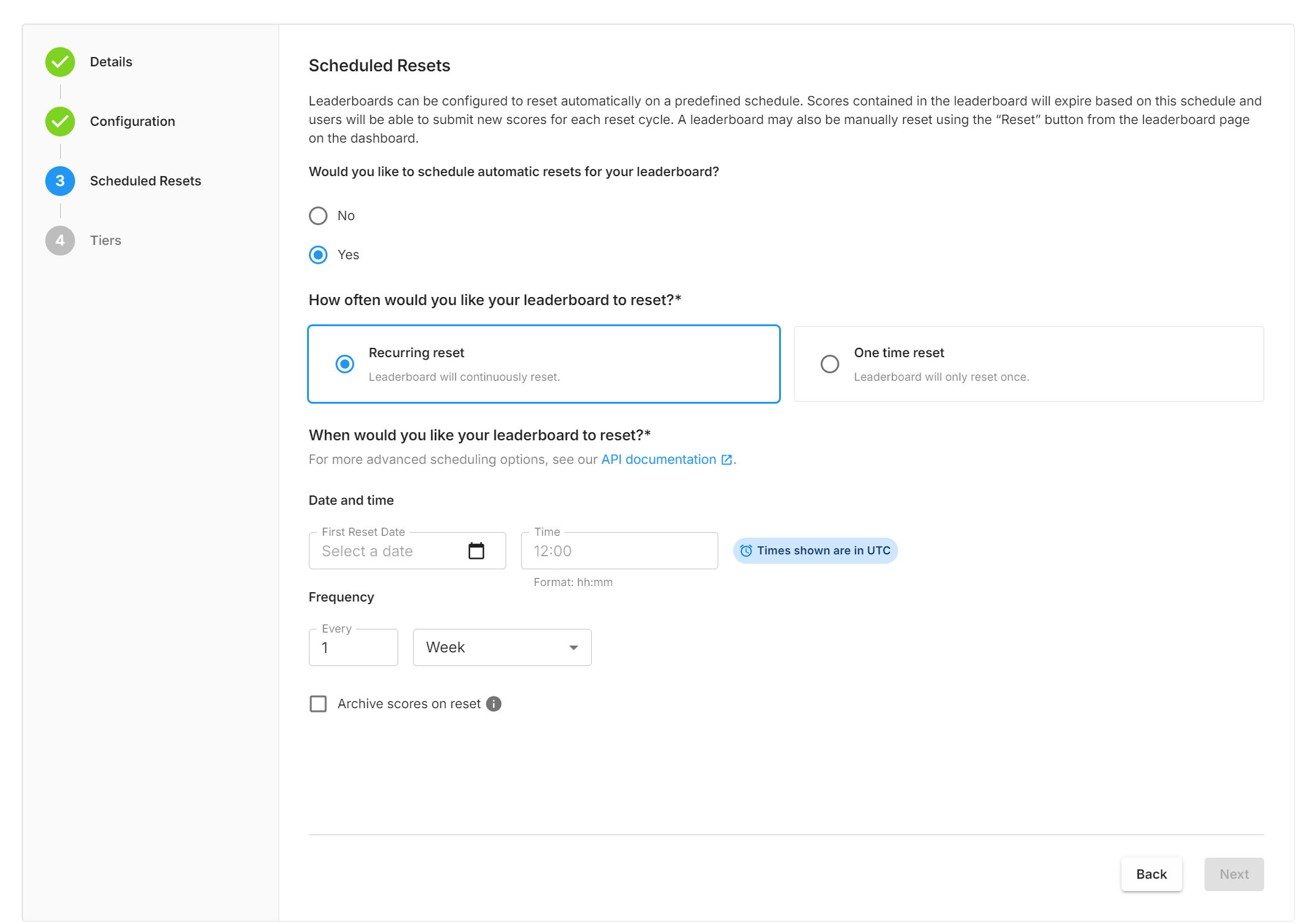
런치. 에드리더보드

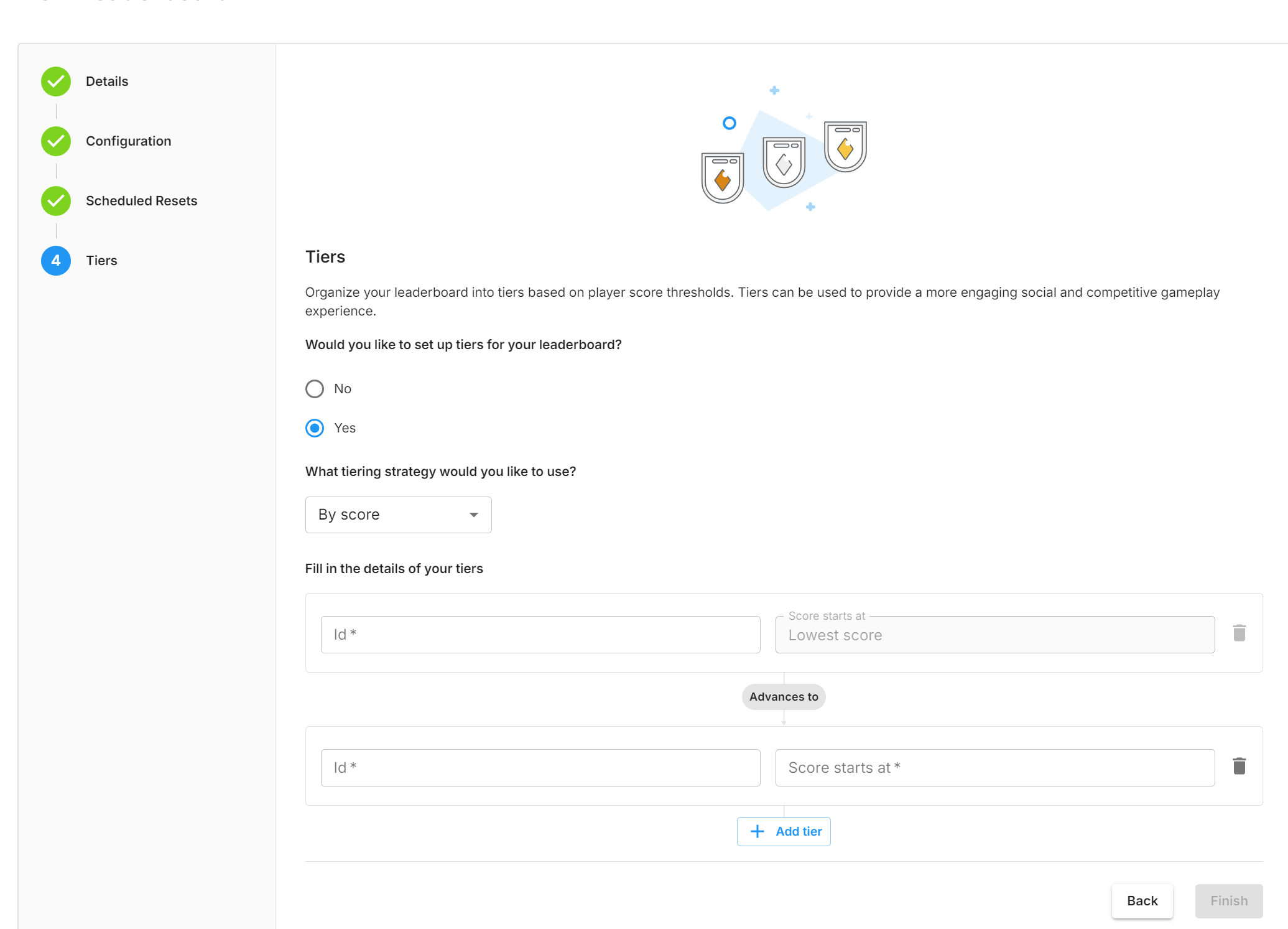


오름차순이나 내림차순

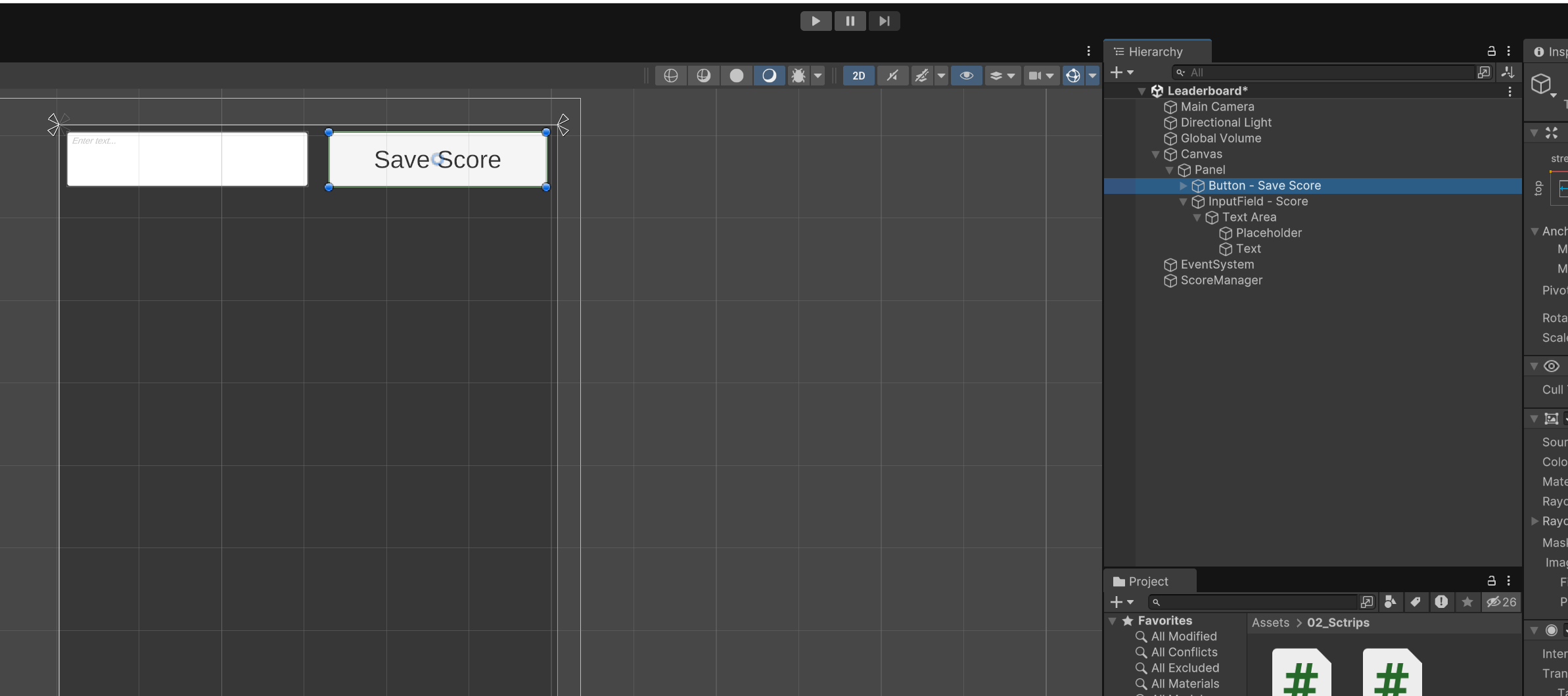


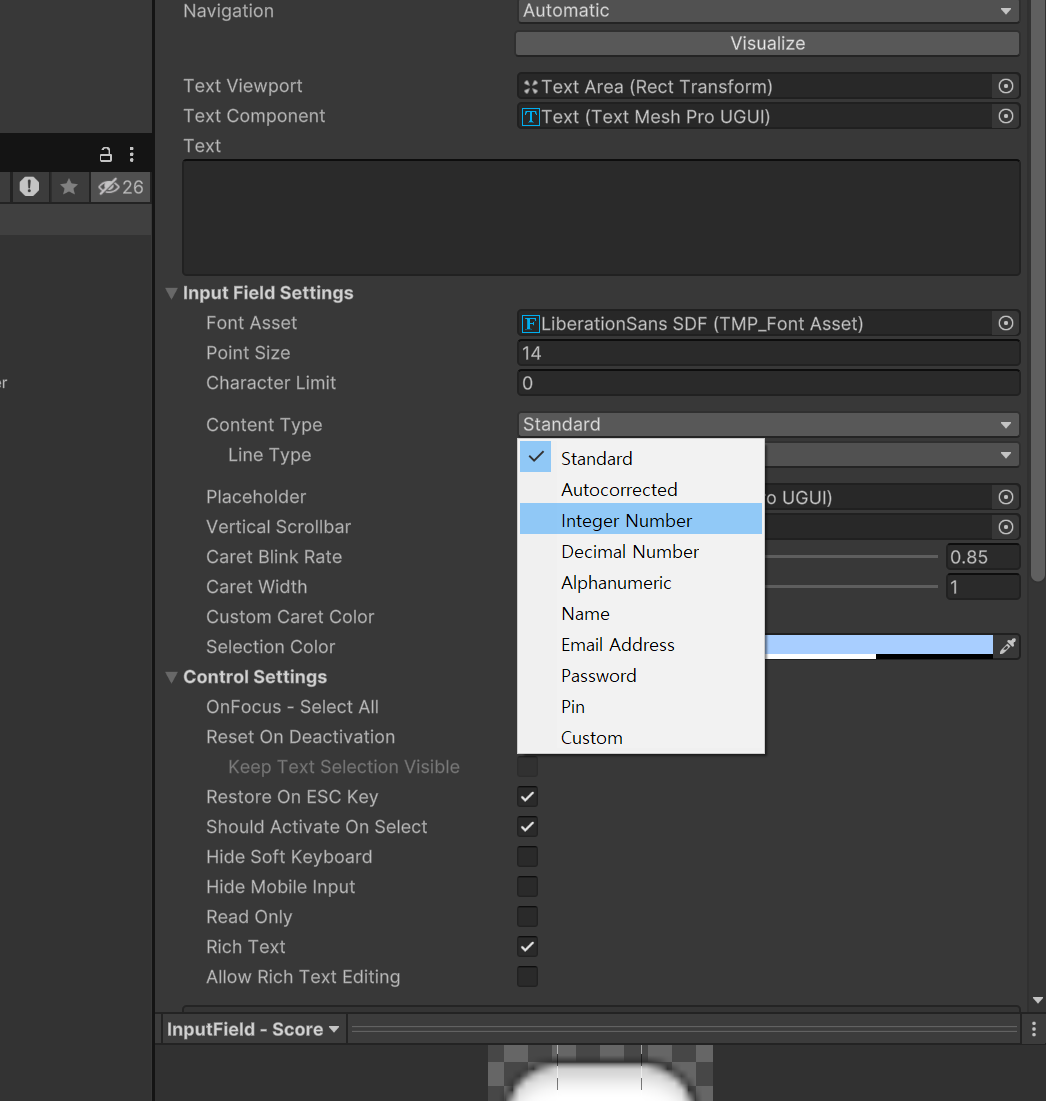
스케쥴 잡고 리셋 할 것인지, 한 번에 할 것인지

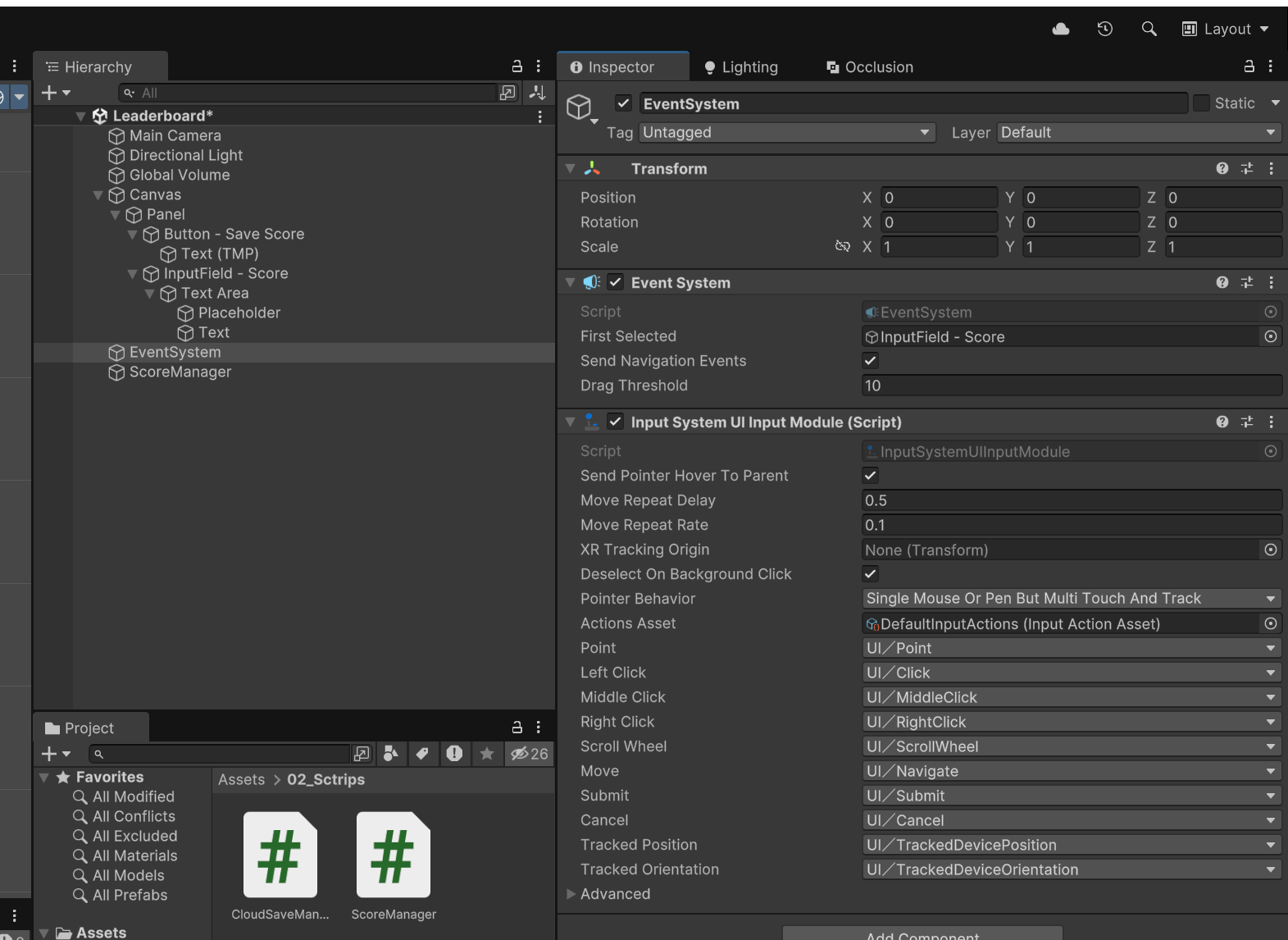




티어 정하기







First Selected 하면 처음 뷰가 절로 간다

public class ScoreManager : MonoBehaviour

{

    [Header("UI")]

    [SerializeField] private Button saveScoreButton;

    [SerializeField] private TMP\_InputField scoreIf;

    private async void Awake()

    {

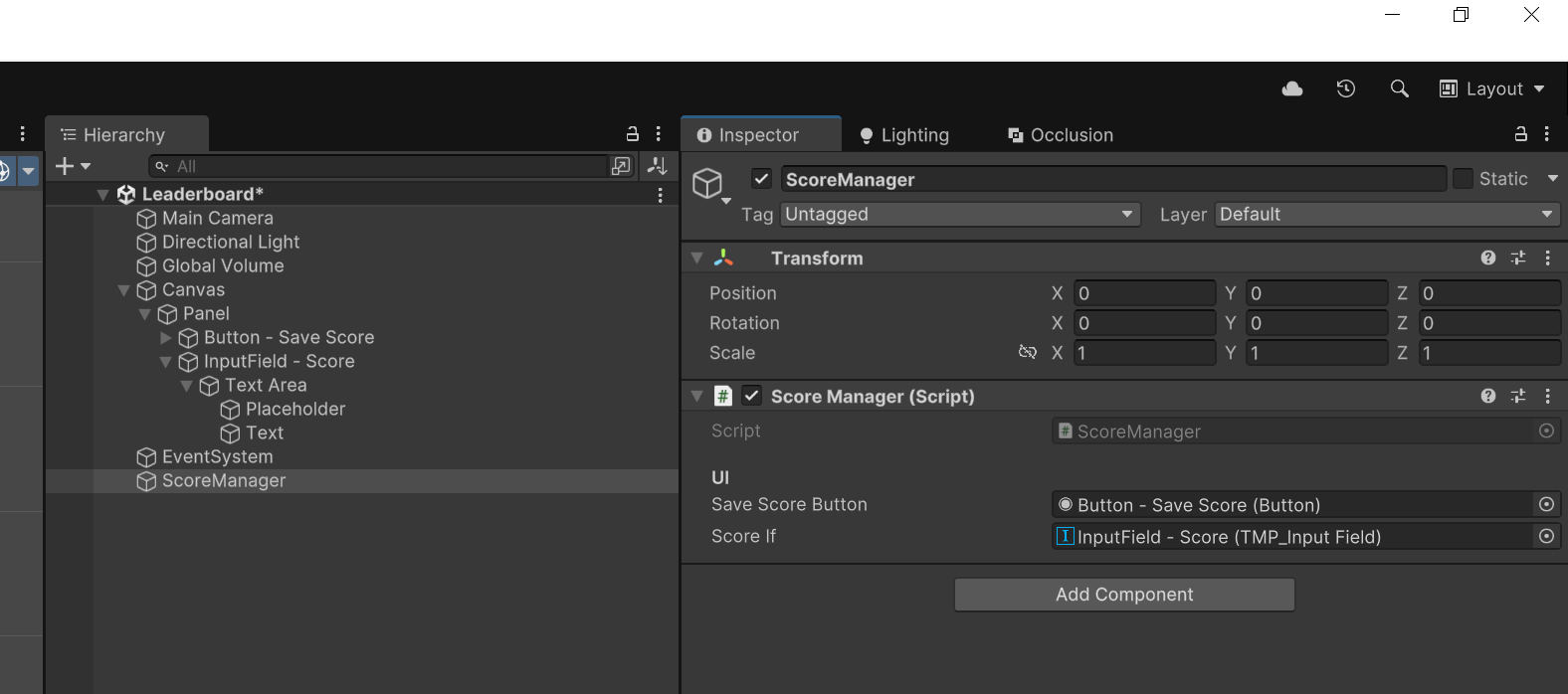
        await UnityServices.InitializeAsync();

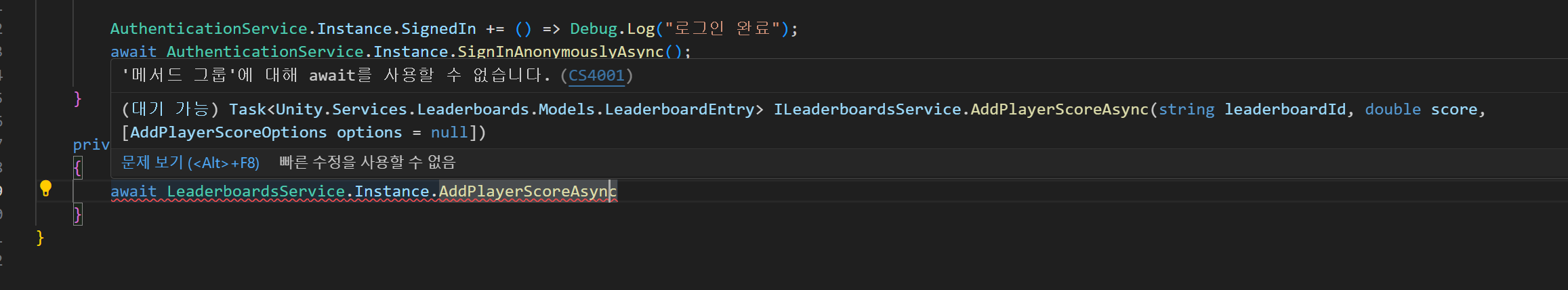
        AuthenticationService.Instance.SignedIn += () => Debug.Log("로그인 완료");

        await AuthenticationService.Instance.SignInAnonymouslyAsync();

    }

}





빌드 프로필

