

# Implementasi Firebase Sederhana pada Incremental Game

Site: DILo Game Academy  
Course: Game Programming Studi Independen  
Book: Implementasi Firebase Sederhana pada Incremental Game  
Printed by: 408 Dewa Sinar Surya  
Date: Monday, 4 October 2021, 7:56 PM

# Table of contents

1. Introduction
2. Step 1 - Save/Loda User Progress (Local)
3. Step 2 - Setup Firebase SDK
4. Step 3 - Save/Load User Progress (Cloud)
5. Step 4 - Setup and Fire Analytics

# 1. Introduction

Materi kali ini merupakan lanjutan dari materi Incremental Game, yaitu mengimplementasi Cloud Save dan Analytics pada Unity menggunakan Firebase. Cloud Save merupakan fitur seperti Local Save, namun penyimpanan progress user menggunakan server sebagai tempat penyimpanannya, bukan lokal. Jadi, player membutuhkan koneksi untuk melakukan save/load dengan metode ini. Sedangkan Analytics merupakan fitur yang digunakan untuk me-record setiap data tentang behaviour player yang dapat digunakan sebagai fasilitas untuk menganalisa sesuatu berhubungan dengan game tersebut.

Berikut adalah penjelasan singkat tentang Cloud Save dalam bentuk diagram:



Berikut adalah penjelasan singkat tentang Analytics dalam bentuk diagram:



Selama menyelesaikan materi ini kita juga akan mempelajari tentang:

- Bagaimana Save/Load data di Local Storage (Device yang sedang digunakan)
- Meng-convert JSON ke object class dan sebaliknya.

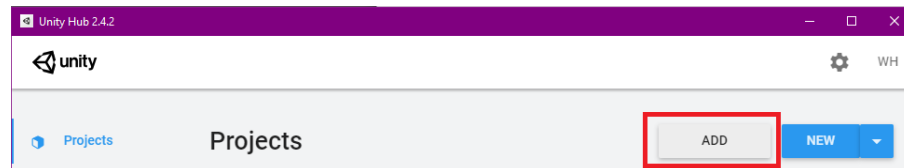
- Meng-convert Array Byte ke object class dan sebaliknya.
- Membuat .apk Android dari Unity.
- Menggunakan Firebase SDK di Unity.
- Cara kerja dan implementasi Cloud Save di Unity.
- Menembakkan Events dan User Properties (Analytics) dari Unity.

Sebelum masuk ke tahap lanjutan Incremental Game (Firebase), clone/download terlebih dahulu project sebelumnya pada link berikut:

<https://drive.google.com/file/d/1YtEjNsaKABrM2Y7WA83LklcWPP3iyT9f/view?usp=sharing>

Atau kamu boleh menggunakan project kamu sendiri yang sudah diselesaikan pada materi Incremental Game.

Setelah itu tambahkan project unity tersebut dari Unity Hub dengan menekan tombol Add.



Lalu, buka project unity tersebut.

Setelah ini kamu siap untuk mulai mengimplementasikan materi ini.

## 2. Step 1 - Save/Load User Progress (Local)

### Step 1 - Save/Load User Progress (Local)

#### a. Setup User Progress Data

- Duplikat atau copy project Incremental Game-mu dari materi sebelumnya.
- Buatlah script UserProgressData sebagai object untuk menyimpan progress user dengan kode berikut.

```
using System.Collections.Generic;  
  
[System.Serializable]  
public class UserProgressData  
{  
    public double Gold = 0;  
    public List<int> ResourcesLevels = new List<int> ();  
}
```

Jangan lupa untuk menambahkan System.Serializable agar object nantinya dapat di-serialize menjadi string untuk disimpan dan kemudian di-deserialize kembali menjadi object untuk di-load.

#### b. Menambahkan User Data Manager

- Buatlah script UserDataManager untuk mengatur script UserProgressData agar dapat disimpan atau di-load.

```
using UnityEngine;

public static class UserDataManager
{
    private const string PROGRESS_KEY = "Progress";

    public static UserProgressData Progress;

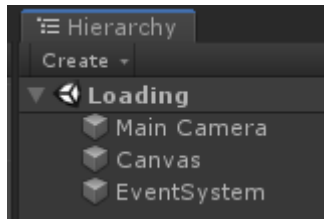
    public static void Load ()
    {
        // Cek apakah ada data yang tersimpan sebagai PROGRESS_KEY
        if (!PlayerPrefs.HasKey (PROGRESS_KEY))
        {
            // Jika tidak ada, maka buat data baru
            Progress = new UserProgressData ();
            Save ();
        }
        else
        {
            // Jika ada, maka timpa progress dengan yang sebelumnya
            string json = PlayerPrefs.GetString (PROGRESS_KEY);
            Progress = JsonUtility.FromJson<UserProgressData> (json);
        }
    }

    public static void Save ()
```

```
{  
    string json = JsonUtility.ToJson (Progress);  
    PlayerPrefs.SetString (PROGRESS_KEY, json);  
}  
}
```

c. Membuat Scene Loading

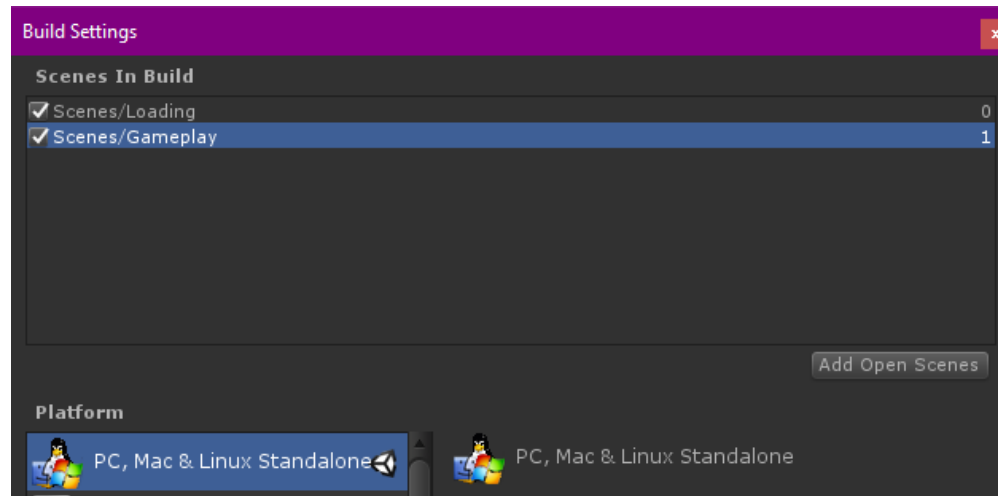
- Pilih scene Gameplay pada window Project dan duplikat scene tersebut dengan cara Ctrl+D. Lalu ubah namanya menjadi Loading.
- Buka scene Loading dan hapus semua game object, kecuali Main Camera, Canvas, dan EventSystem karena masih kita butuhkan. Untuk sekarang, hapus juga semua child dari Canvas.



- Tambahkan script LoadingController untuk memanggil fungsi Load pada UserDataManager dan kemudian membuka scene Gameplay.

```
using UnityEngine;  
using UnityEngine.SceneManagement;  
  
public class LoadingController : MonoBehaviour  
{  
    private void Start ()  
    {  
        UserDataManager.Load ();  
        SceneManager.LoadScene (1);  
    }  
}
```

- Sebelum mencoba, jangan lupa untuk menambahkan scene Loading dan Gameplay pada menu File à Build Settings... agar bisa dibaca oleh SceneManager. (Cara menambahkannya adalah drag file scene ke dalam window Scenes In Build)



Pastikan urutannya sesuai dengan urutan terbukanya scene, yaitu seperti gambar di atas.

- Buatlah gameobject kosong pada scene Loading, beri nama Loading dan tambahkan script LoadingController.
- Saat di-play dari scene Loading, maka game akan otomatis load data dari PlayerPrefs dan membuka scene Gameplay. Namun untuk sekarang, data yang di-load belum diimplementasikan pada tampilan game.

#### d. Save/Load Gold ke Elemen Game

- Bukalah script GameManager dan hapus variable TotalGold, lalu ganti semua yang memakai TotalGold menjadi UserDataManager.Progress.Gold. Tambahkan juga fungsi Save ketika menambahkan Gold.



```
// ...

private void Start ()
{
    AddAllResources ();

    GoldInfo.text = $"Gold: {
UserDataManager.Progress.Gold.ToString ("0") }";
}

// ...

private void CheckResourceCost ()
{
    foreach (ResourceController resource in _activeResources)
    {
        bool isBuyable = false;

        if (resource.IsUnlocked)
        {
            isBuyable = UserDataManager.Progress.Gold >=
resource.GetUpgradeCost ();
        }
        else
        {
            isBuyable = UserDataManager.Progress.Gold >=
resource.GetUnlockCost ();
        }
    }
}
```

```
        resource.ResourceImage.sprite = ResourcesSprites[isBuyable ? 1 : 0];
    }
}

// ...

public void AddGold (double value)
{
    UserDataManager.Progress.Gold += value;
    GoldInfo.text = $"Gold: { UserDataManager.Progress.Gold.ToString
("0") }";
    UserDataManager.Save ();
}

// ...
```

- Ubah juga fungsi yang memakai Total Gold pada class ResourceController.

```
// ...

public void UpgradeLevel ()
{
    double upgradeCost = GetUpgradeCost ();
    if (UserDataManager.Progress.Gold < upgradeCost)
    {
        return;
    }

    GameManager.Instance.AddGold (-upgradeCost);
    _level++;

    ResourceUpgradeCost.text = $"Upgrade Cost\n{ GetUpgradeCost () }";
    ResourceDescription.text = $" { _config.Name } Lv. { _level } \n+ {
GetOutput ().ToString ("0") }";
}

public void UnlockResource ()
{
    double unlockCost = GetUnlockCost ();
    if (UserDataManager.Progress.Gold < unlockCost)
    {
        return;
    }
}
```

```
        SetUnlocked (true);  
        GameManager.Instance.ShowNextResource ();  
        AchievementController.Instance.UnlockAchievement  
(AchievementType.UnlockResource, _config.Name);  
    }  
  
    // ...
```

e. Save/Load Resources Data ke Elemen Game

- Tambahkan index dan ubah variable level pada ResourceController.

```
// ...  
  
private int _index;  
private int _level  
{  
    set  
    {  
        // Menyimpan value yang di set ke _level pada Progress Data  
        UserDataManager.Progress.ResourcesLevels[_index] = value;  
        UserDataManager.Save ();  
    }  
  
    get  
    {  
        // Mengecek apakah index sudah terdapat pada Progress Data  
        if (!UserDataManager.HasResources (_index))  
        {  
            // Jika tidak maka tampilkan level 1  
            return 1;  
        }  
  
        // Jika iya maka tampilkan berdasarkan Progress Data  
        return UserDataManager.Progress.ResourcesLevels[_index];  
    }  
}  
  
// ...
```

```
public void SetConfig (int index, ResourceConfig config)
{
    _index = index;
    _config = config;

    // ToString("0") berfungsi untuk membuang angka di belakang koma
    ResourceDescription.text = $"{ _config.Name } Lv. { _level }\n+{
GetOutput ().ToString ("0") }";

    ResourceUnlockCost.text = $"Unlock Cost\n{ _config.UnlockCost }";
    ResourceUpgradeCost.text = $"Upgrade Cost\n{ GetUpgradeCost () }";

    SetUnlocked (_config.UnlockCost == 0 ||
UserDataManager.HasResources (_index));
}

// ...

public void SetUnlocked (bool unlocked)
{
    IsUnlocked = unlocked;

    if (unlocked)
    {
        // Jika resources baru di unlock dan belum ada di Progress Data,
maka tambahkan data

        if (!UserDataManager.HasResources (_index))
        {
```

```
        UserDataManager.Progress.ResourcesLevels.Add (_level);  
        UserDataManager.Save ();  
    }  
}  
  
ResourceImage.color = IsUnlocked ? Color.white : Color.grey;  
ResourceUnlockCost.gameObject.SetActive (!unlocked);  
ResourceUpgradeCost.gameObject.SetActive (unlocked);  
}  
}
```

- Jangan lupa pula untuk mengupdate fungsi SetConfig pada GameManager, karena perubahan fungsi.

```
// ...

private void AddAllResources ()
{
    bool showResources = true;
    int index = 0;
    foreach (ResourceConfig config in ResourcesConfigs)
    {
        GameObject obj = Instantiate (ResourcePrefab.gameObject,
ResourcesParent, false);
        ResourceController resource = obj.GetComponent<ResourceController>
();

        resource.SetConfig (index, config);
        obj.gameObject.SetActive (showResources);

        if (showResources && !resource.IsUnlocked)
        {
            showResources = false;
        }

        _activeResources.Add (resource);
        index++;
    }
}
```



```
// ...
```

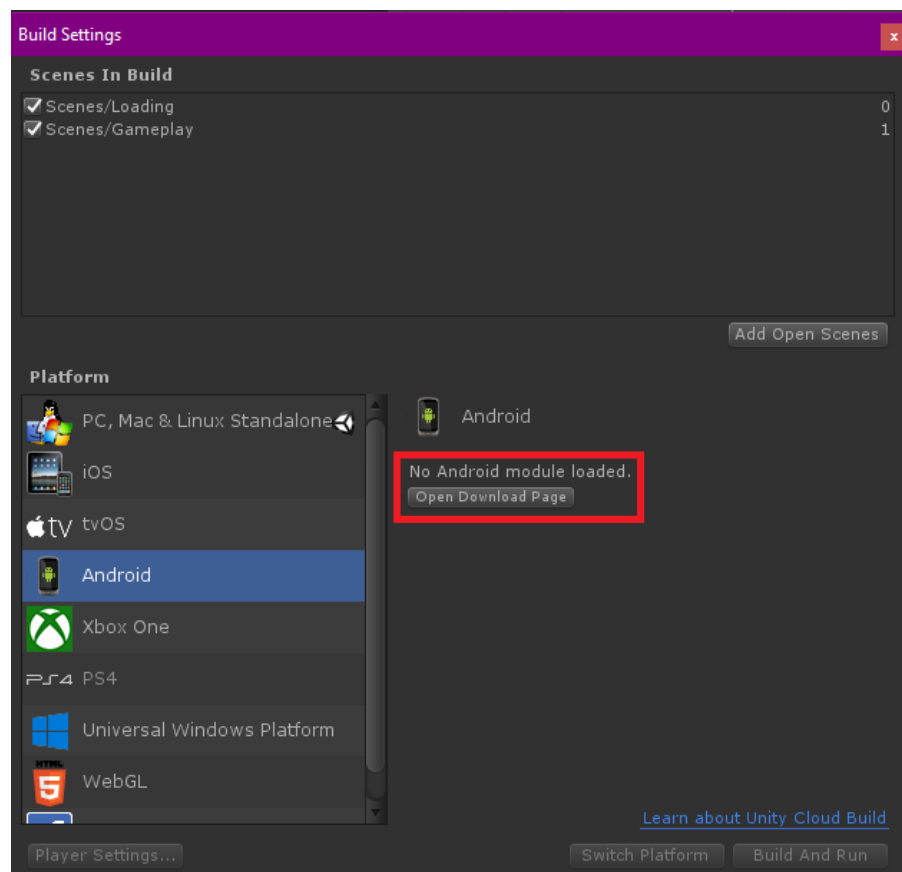
- Selamat, progress-mu selama bermain akan selalu tersimpan di penyimpanan lokal. Klik menu Edit à Clear All PlayerPrefs untuk membersihkan progress yang disimpan.

### 3. Step 2 - Setup Firebase SDK

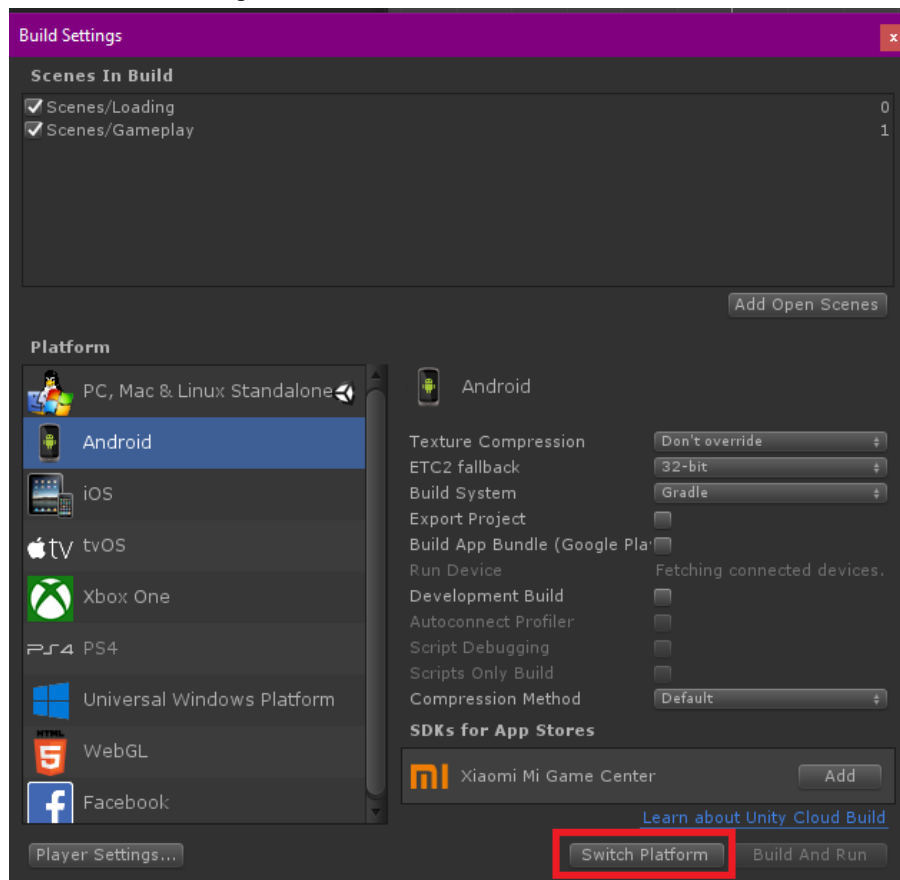
#### Step 2 - Setup Firebase SDK

a. Change Target Build to Android

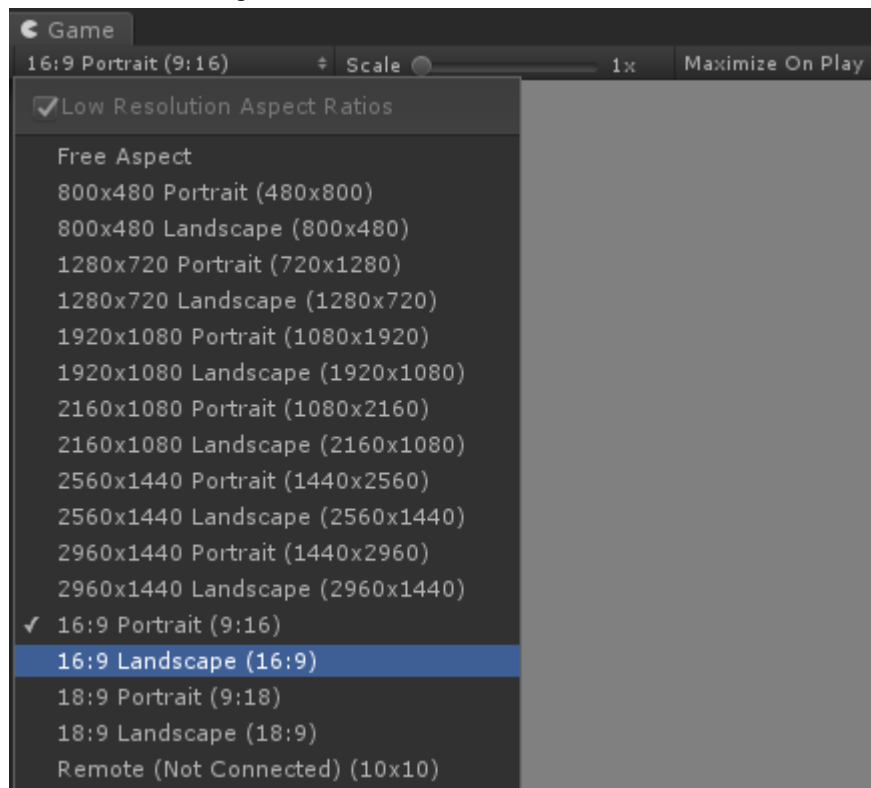
- Untuk Firebase SDK kebanyakan hanya bisa digunakan pada platform Android/iOS, maka dari itu kita akan mengganti target build ke Android.
- Klik menu File à Build Settings, lalu pada Platform pilihlah Android, jika belum terinstall maka Download dan Install terlebih dahulu pada tombol Open Download Page.



- Jika sudah maka tekan Switch Platform di bagian bawah.



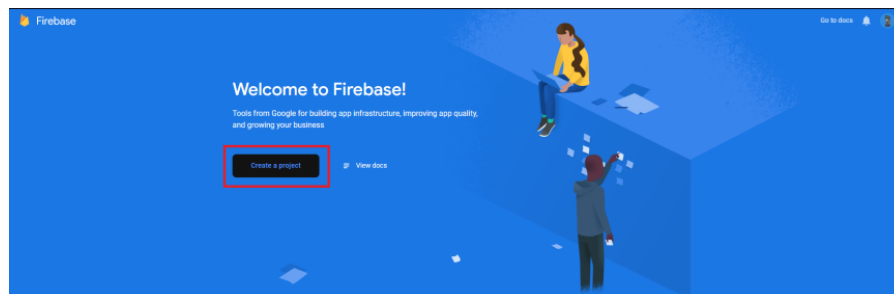
- Ubahlah tampilan menjadi Landscape, karena game yang akan kita buat berorientasi landscape.



- Sekarang kamu sudah menjalankan game pada platform Android.

#### b. Create Firebase Project Account

- Bukalah situs [console.firebase.google.com](https://console.firebase.google.com), ikuti langkahnya hingga ke halaman berikut.



Klik tombol Create a Project.

- Masukkan nama project dan centang agreement-nya. Nama project di sini juga sekaligus menggenerate id project-nya nanti yang tidak bisa diubah lagi.


× Create a project (Step 1 of 3)

## Let's start with a name for your project<sup>?</sup>

Project name

**DILo - Incremental Game**

---

 dilo---incremental-game

☒ I accept the [Firebase terms](#)

**Continue**

- Tekan continue hingga masuk ke halaman berikut. Ubah lokasi analytics menjadi Indonesia (sesuaikan dengan target release utama kalian), lalu centang saja semuanya dan tekan Create Project.

✕ Create a project (Step 3 of 3)

## Configure Google Analytics

Analytics location ⓘ

Indonesia

Data sharing settings and Google Analytics terms

☒ Use the default settings for sharing Google Analytics data. [Learn more](#)

- ✓ Share your Analytics data with Google to improve Google Products and Services
- ✓ Share your Analytics data with Google to enable Benchmarking
- ✓ Share your Analytics data with Google to enable Technical Support
- ✓ Share your Analytics data with Google Account Specialists

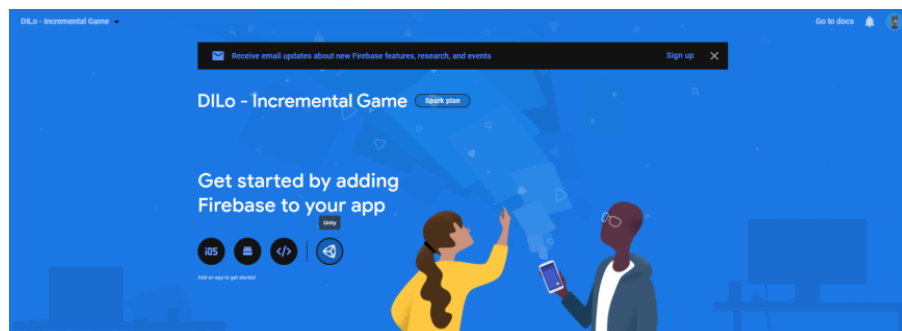
☒ I accept the [Measurement Controller-Controller Data Protection terms](#) and acknowledge I am subject to the [EU End User Consent Policy](#). This is required when sharing Google Analytics data to improve Google Products and Services. [Learn more](#)

☒ I accept the [Google Analytics terms](#)

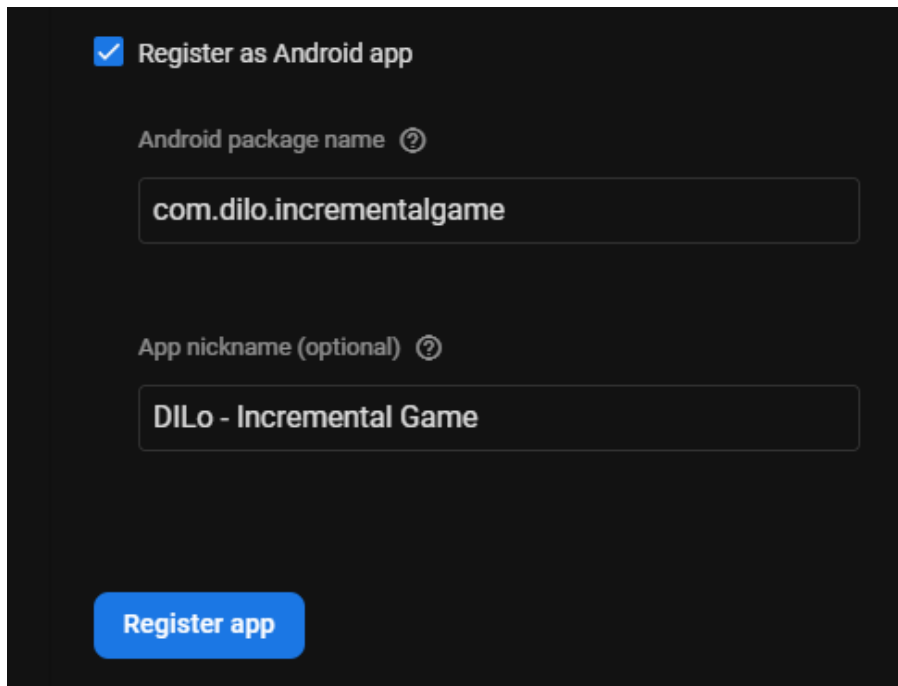
Upon project creation, a new Google Analytics property will be created and linked to your Firebase project. This link will enable data flow between the products. Data exported from your Google Analytics property into Firebase is subject to the Firebase terms of service, while Firebase data imported into Google Analytics is subject to the Google Analytics terms of service. [Learn more](#)

[Previous](#) [Create project](#)

- Setelah semuanya berhasil, kamu akan ke page dashboard utama, klik logo Unity yang terdapat di sana dan kamu akan dibawa ke form setup.



- Centang bagian Register as Android app saja pada bagian pertama, lalu isikan Android package name-nya sesuai nanti yang akan diisi pada Unity (tidak harus mengikuti contoh) tapi biasanya formatnya `com.CompanyName.ProductName`. Untuk app nickname hanya bersifat optional. Setelah selesai, tekan Register app.



☒ Register as Android app

Android package name ⓘ

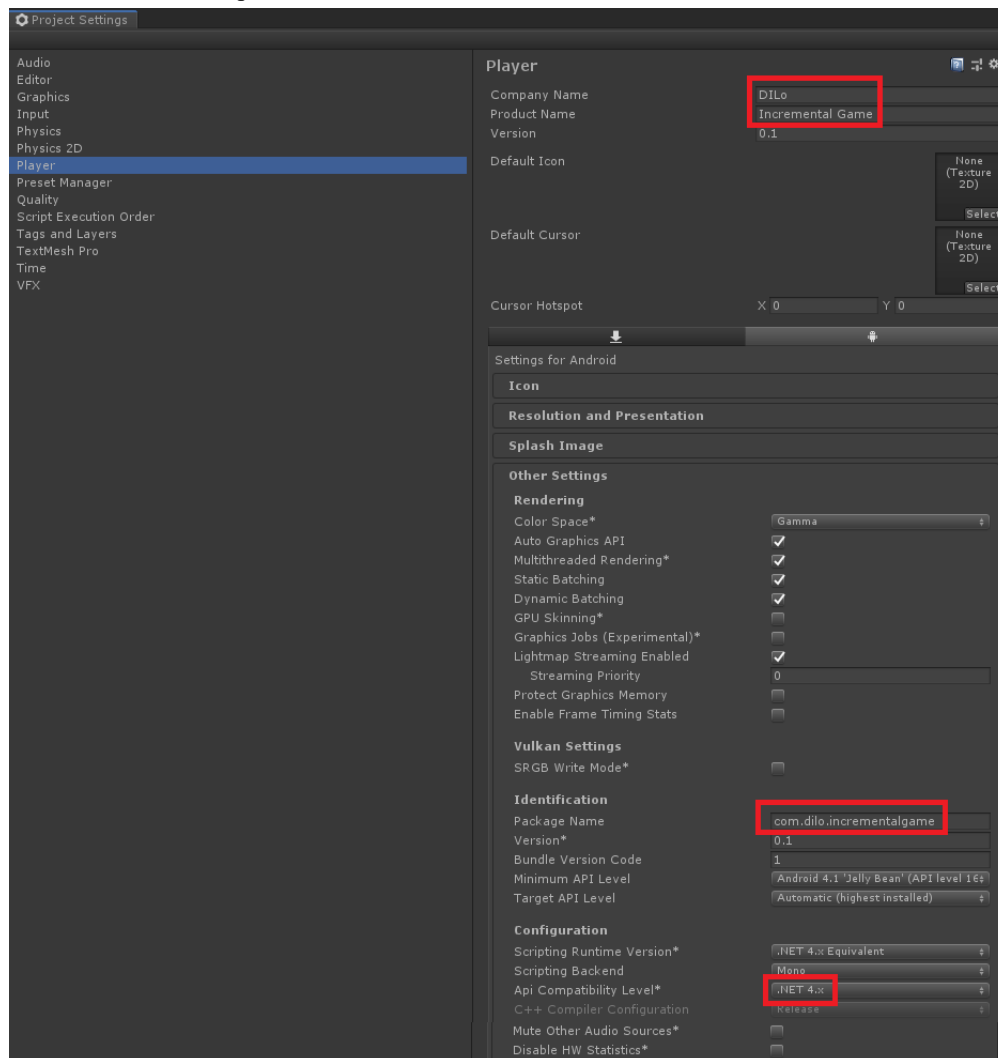
com.dilo.incrementalgame

App nickname (optional) ⓘ

DILo - Incremental Game

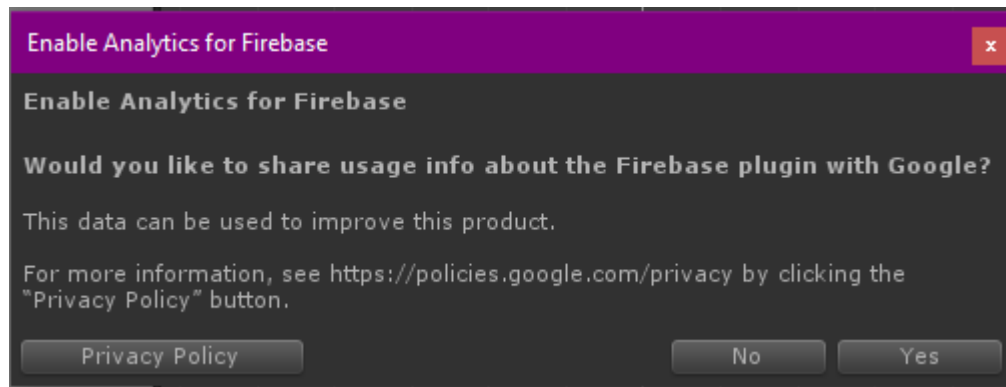
Register app

- Pada tahap selanjutnya, download google-services.json, masukkan hasilnya ke dalam folder Assets pada project Unity kalian. Lalu tekan Next.
  - Selanjutnya download Firebase Unity SDK dan extract hasilnya, untuk import ke Unity kita skip dulu dan tekan Next, lalu Continue to Console.
- c. Import Firebase SDK to Unity
- Buka project Unity kalian, lalu klik menu Edit à Project Settings, pilih bagian Player.
  - Sesuaikan Company dan Product Name sesuai dengan Package Name (seperti yang diinstruksikan sebelumnya, package name harus sama dengan yang sudah di set di Firebase. Ubah API Compatibility-nya menjadi .NET 4.x karena kebutuhan Firebase. Jika sudah, close saja window tersebut.

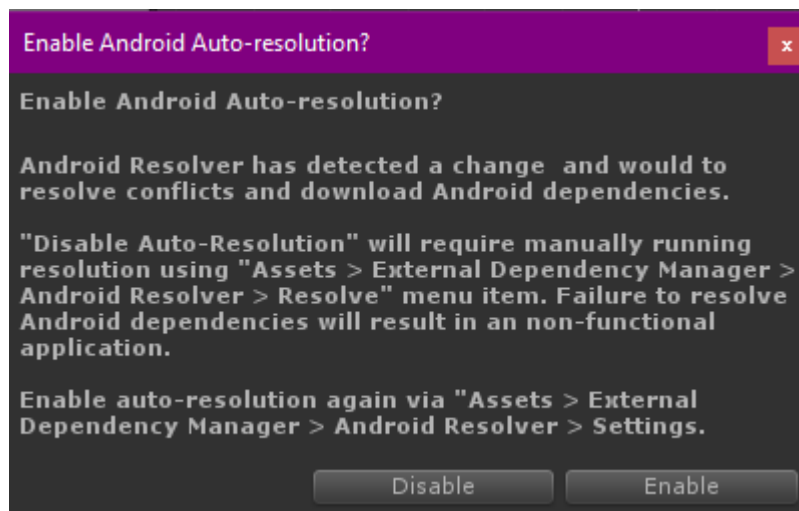


- Pilih menu Assets à Import Package à Custom Package untuk memasukkan package Firebase. Pilih package firebase\_unity\_sdk/dotnet4/FirebaseAnalytics.unitypackage (untuk kebutuhan Analytics) dari hasil download yang sudah di-extract sebelumnya. Lalu masukkan juga FirebaseStorage.unitypackage (untuk kebutuhan Cloud Save) yang bisa kalian temukan di folder yang sama.
- Anda bebas memilih Yes/No setelah muncul pop up berikut untuk menghidupkan fitur analytics penggunaan Firebase agar mengimprove product Google.

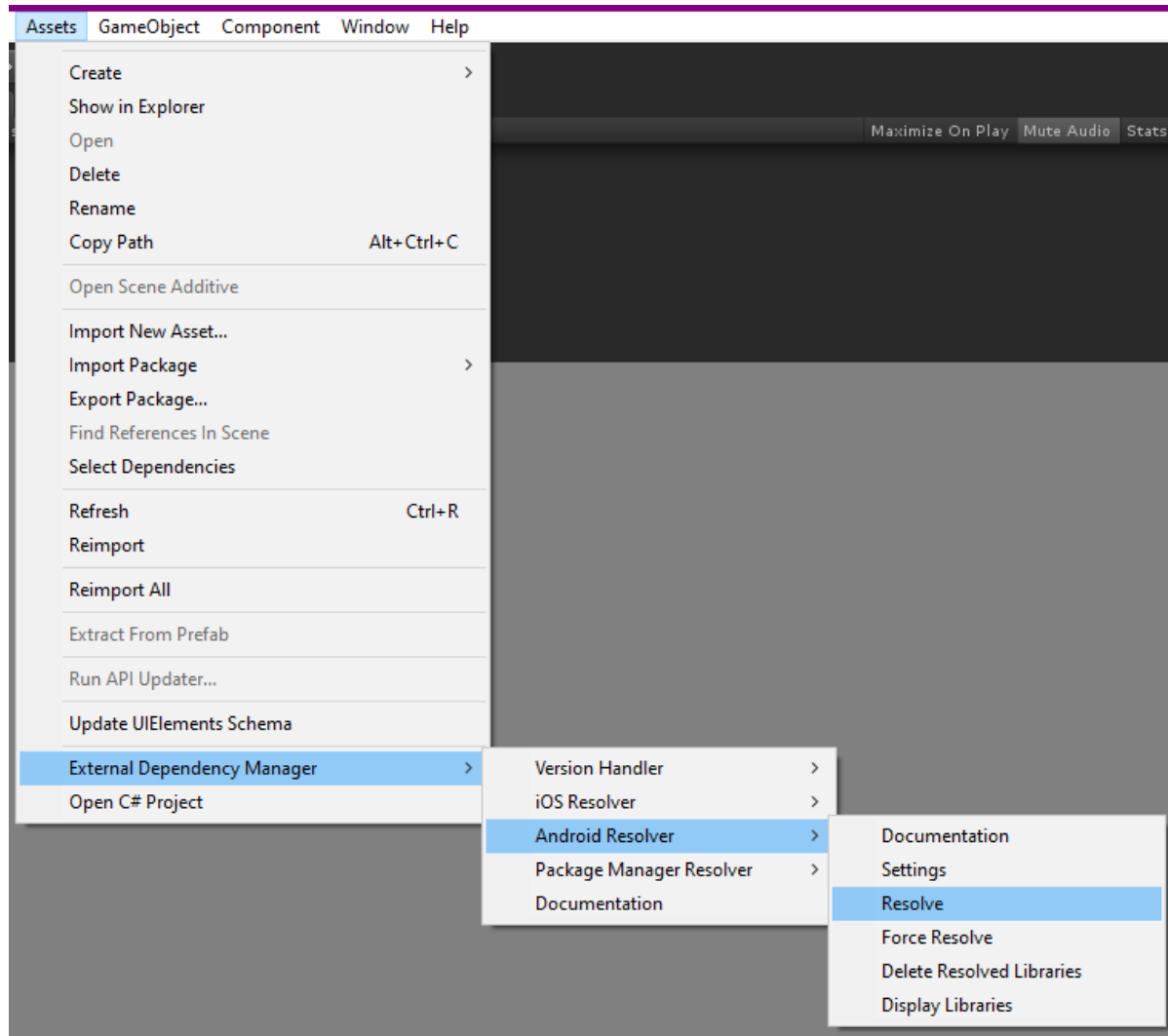




- Dan Pilih Disable setelahnya, karena kita tidak membutuhkan Auto-Resolve ini, nantinya kita akan melakukan Manual-Resolve saja. Resolve ini digunakan untuk mengimport plugin-plugin yang dibutuhkan agar Firebase dapat bekerja dengan baik.



- Pilih menu Assets à External Dependency Manager à Android Resolver à Resolve untuk melakukan Manual Resolve dan pastikan internet kalian terkoneksi, karena biasanya membutuhkan data untuk mendownload plugin-plugin-nya.

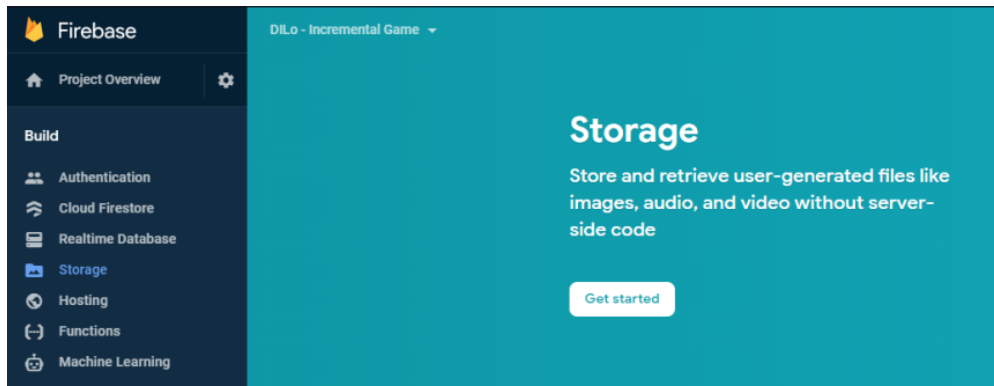


- Setelah sukses, berarti kalian sudah berhasil mengimport Firebase ke dalam project Unity kalian.

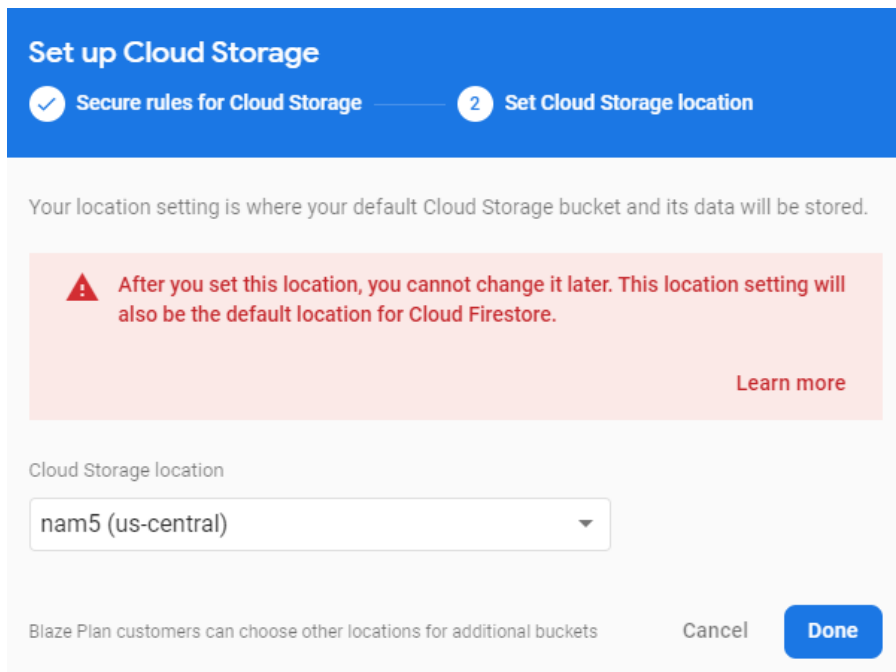
## 4. Step 3 - Save/Load User Progress (Cloud)

### Step 3 - Save/Load User Progress (Cloud)

- a. Setup Storage Dashboard
- Kembali ke situs Firebase Console, buka menu Storage, lalu tekan Get Started.

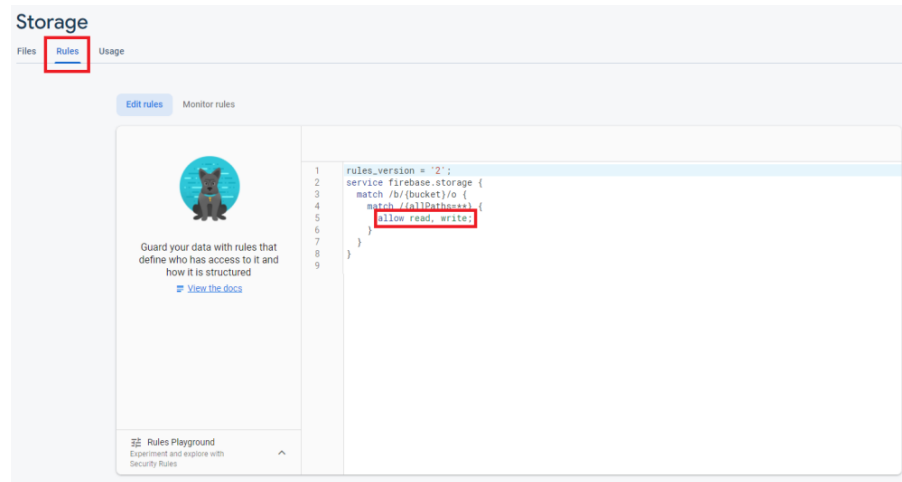


- Lanjutkan setiap langkahnya, dan pada window di bawah ini langsung Continue saja, tidak perlu mengubah region, karena secara default yang terpilih itu sudah multi-region.



- Setelah berhasil, buka menu Rules, lalu ubah rules menjadi seperti ini. (Kalian hanya perlu mengubah *allow read, write*; agar semua player

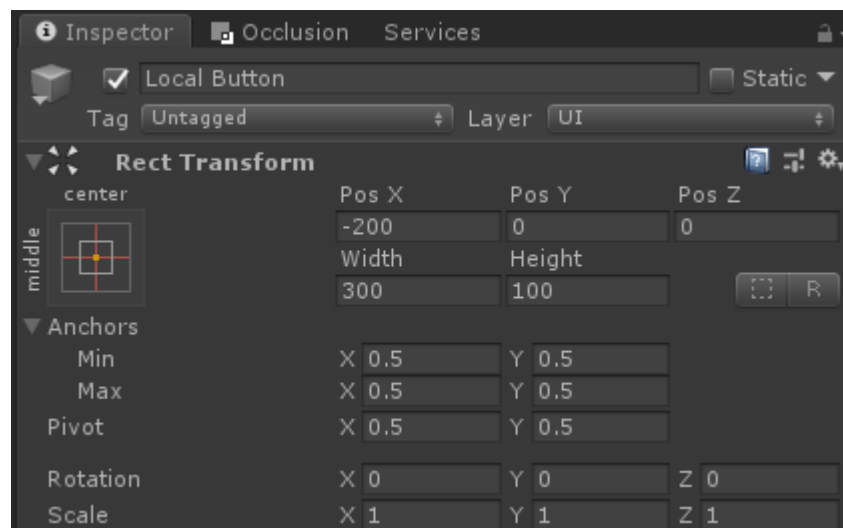
bisa mengakses secara public)



- Lalu tekan Publish.

#### b. Download Progress from Cloud

- Buatlah gameobject Button sebagai child dari Canvas di scene Loading. Ubah namanya menjadi Local Button, karena digunakan untuk load data dari lokal. Ubah ukuran dan posisinya menjadi:



- Lalu ubah isi text-nya menjadi Load from Local.
- Duplikat button tersebut dan beri nama button baru Cloud Button, ubah posisi x nya dari -200 menjadi 200 agar bersampingan dengan button sebelumnya, lalu ubah isi text-nya menjadi Load from Cloud.
- Setelah ini, kita perlu menambahkan kode pada script UserDataManager untuk mensupport load data dari cloud.

```
using Firebase.Storage;
using System.Collections;
using System.Text;
using System.Threading.Tasks;
using UnityEngine;

public static class UserDataManager
{
    private const string PROGRESS_KEY = "Progress";

    public static UserProgressData Progress = new UserProgressData ();

    public static void LoadFromLocal ()
    {
        // ...
    }

    public static IEnumerator LoadFromCloud (System.Action
onComplete)
    {
        StorageReference targetStorage = GetTargetCloudStorage ();

        bool isCompleted = false;
        bool isSuccessfull = false;
        const long maxAllowedSize = 1024 * 1024; // Sama dengan 1 MB
        targetStorage.GetBytesAsync (maxAllowedSize).ContinueWith
```

```
((Task<byte[]> task) =>
{
    if (!task.IsFaulted)
    {
        string json = Encoding.Default.GetString (task.Result);
        Progress = JsonUtility.FromJson<UserProgressData> (json);
        isSuccessfull = true;
    }

    isCompleted = true;
});

while (!isCompleted)
{
    yield return null;
}

// Jika sukses mendownload, maka simpan data hasil download
if (isSuccessfull)
{
    Save ();
}
else
{
    // Jika tidak ada data di cloud, maka load data dari local
    LoadFromLocal ();
}
```

```
}

    onComplete?.Invoke ();
}

public static void Save ()
{
// ...
}

private static StorageReference GetTargetCloudStorage ()
{
    // Gunakan Device ID sebagai nama file yang akan disimpan di
cloud
    string deviceID = SystemInfo.deviceUniqueIdentifier;

    FirebaseStorage storage = FirebaseStorage.DefaultInstance;

    return storage.GetReferenceFromUrl
($"${storage.RootReference}/{deviceID}");
}
// ...
}
```

- Lalu, karena kita tidak melakukan auto load dari local seperti sebelumnya, maka ubahlah semua bagian script Loading Controller menjadi:

```
using UnityEngine;
using UnityEngine.SceneManagement;
using UnityEngine.UI;

public class LoadingController : MonoBehaviour
{
    [SerializeField] private Button _localButton;
    [SerializeField] private Button _cloudButton;

    private void Start ()
    {
        _localButton.onClick.AddListener (() =>
        {
            SetButtonInteractable (false);
            UserDataManager.LoadFromLocal ();
            SceneManager.LoadScene (1);
        });

        _cloudButton.onClick.AddListener (() =>
        {
            SetButtonInteractable (false);
            StartCoroutine (UserDataManager.LoadFromCloud (() =>
SceneManager.LoadScene (1))));
        });

        // Button didisable agar mencegah tidak terjadinya spam klik ketika
```



```
// proses onclick pada button sedang berjalan
}

// Mendisable button agar tidak bisa ditekan
private void SetButtonInteractable (bool interactable)
{
    _localButton.interactable = interactable;
    _cloudButton.interactable = interactable;
}
}
```

- Kemudian assign variable button-button pada komponen Loading Controller. Kalian sudah bisa load dari Cloud atau Local. Untuk melakukan load dari cloud kalian harus memiliki file yang tersimpan pada storage Firebase, sehingga tahap selanjutnya adalah Upload Data ke Firebase.

- Firebase juga menyediakan guide untuk mendownload file di Storage untuk info yang lebih detail: <https://firebase.google.com/docs/storage/unity/download-files>.

#### c. Upload Progress to Cloud

- Pertama-tama kita harus menambahkan upload ke cloud pada fungsi Save, seperti berikut dan menambahkan parameter true pada setiap pemanggilan Save yang bersifat penting.

```
public static void LoadFromLocal ()
{
    // Cek apakah ada data yang tersimpan sebagai PROGRESS_KEY
    if (!PlayerPrefs.HasKey (PROGRESS_KEY))
    {
        // Jika tidak ada, maka simpan data baru
        // dan upload ke Cloud
        Save (true);
    }
    else
    {
        // Jika ada, maka timpa progress dengan yang sebelumnya
        string json = PlayerPrefs.GetString (PROGRESS_KEY);
        Progress = JsonUtility.FromJson<UserProgressData> (json);
    }
}

// ...
```

```
public static void Save (bool uploadToCloud = false)
{
    string json = JsonUtility.ToJson (Progress);
    PlayerPrefs.SetString (PROGRESS_KEY, json);

    if (uploadToCloud)
    {
```

```
byte[] data = Encoding.Default.GetBytes (json);  
  
StorageReference targetStorage = GetTargetCloudStorage ();  
  
targetStorage.PutBytesAsync (data);  
}  
}  
  
// ...
```

- Lalu pada ResourceController juga tambahkan parameter true pemanggilan fungsi Save karena Level Up dan Unlock Resource merupakan event penting.

```
// ...

private int _level

{
    set
    {
        // Menyimpan value yang di set ke _level pada Progress Data
        UserDataManager.Progress.ResourcesLevels[_index] = value;
        UserDataManager.Save (true);
    }

    get
    {
        // Mengecek apakah index sudah terdapat pada Progress Data
        if (!UserDataManager.HasResources (_index))
        {
            // Jika tidak maka tampilkan level 1
            return 1;
        }

        // Jika iya maka tampilkan berdasarkan Progress Data
        return UserDataManager.Progress.ResourcesLevels[_index];
    }
}

// ...
```

```
public void SetUnlocked (bool unlocked)
{
    IsUnlocked = unlocked;
    if (unlocked)
    {
        // Jika resources baru di unlock dan belum ada di Progress Data, maka
        // tambahkan data baru
        if (!UserDataManager.HasResources (_index))
        {
            UserDataManager.Progress.ResourcesLevels.Add (_level);
            UserDataManager.Save (true);
        }
    }

    ResourceImage.color = IsUnlocked ? Color.white : Color.grey;
    ResourceUnlockCost.gameObject.SetActive (!unlocked);
    ResourceUpgradeCost.gameObject.SetActive (unlocked);
}
```

- Khusus pada GameManager, fungsi Save di sana bertujuan untuk menyimpan Gold, karena Gold di update setiap frame, maka sebaiknya kita tidak mengupload data setiap frame juga, karena tidak baik secara design dan terlalu men-spam server, bahkan kuota player. Sehingga kita hanya perlu menyimpan data gold setiap 5 detik (sesuai kebutuhan).

```
// ...

public float AutoCollectPercentage = 0.1f;
public float SaveDelay = 5f;
// ...

private float _collectSecond;
private float _saveDelayCounter;
// ...

private void Update ()
{
    float deltaTime = Time.unscaledDeltaTime;
    _saveDelayCounter -= deltaTime;

    // Fungsi untuk selalu mengeksekusi CollectPerSecond setiap detik
    _collectSecond += deltaTime;
    if (_collectSecond >= 1f)
    {
        CollectPerSecond ();
        _collectSecond = 0f;
    }

    CheckResourceCost ();

    CoinIcon.transform.localScale = Vector3.LerpUnclamped
(CoinIcon.transform.localScale, Vector3.one * 2f, 0.15f);

    CoinIcon.transform.Rotate (0f, 0f, Time.deltaTime * -100f);
}
```

```
// ...

public void AddGold (double value)
{
    UserDataManager.Progress.Gold += value;

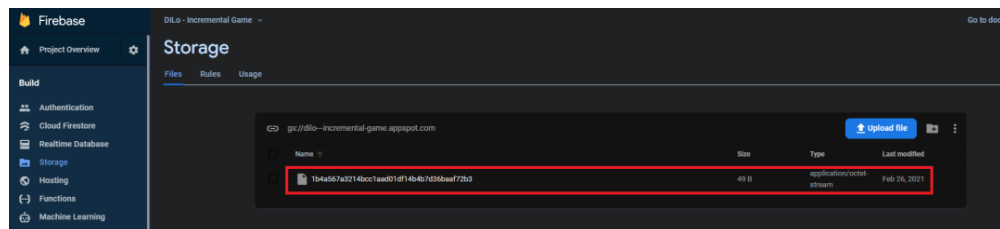
    GoldInfo.text = $"Gold: { UserDataManager.Progress.Gold.ToString
("0") }";

    UserDataManager.Save (_saveDelayCounter < 0f);

    if (_saveDelayCounter < 0f)
    {
        _saveDelayCounter = SaveDelay;
    }
}

// ...
```

- Setelah di-play dan terkoneksi dengan internet, kalian sudah bisa menyimpan dan me-load data dari cloud. Jika cloud berhasil di-upload, maka akan muncul pada halaman Storage Firebase.



- Firebase juga menyediakan guide untuk mengupload file di Storage untuk info yang lebih detail: <https://firebase.google.com/docs/storage/unity/upload-files>.

## 5. Step 4 - Setup and Fire Analytics

### Step 4 - Setup and Fire Analytics

#### a. Create Analytics Manager

- Analytics memiliki Events dan User Properties. Event digunakan sebagai data atau history ketika player melakukan suatu hal yang memiliki insight untuk developer, misalnya ketika player melakukan level up, unlock sebuah fitur, mendapatkan sebuah item, dan lain sebagainya tergantung keperluan developer. Sedangkan User Properties merupakan properti atau atribut yang melekat pada player selama progress player memainkan game tersebut dan lebih sering ditembakkan, tidak seperti event yang hanya sesekali. Misalnya jumlah gold yang dimiliki, atau total spending gold player tersebut. User Property dapat dijadikan sebagai filter report data analytics.
- Pertama-tama, kita perlu membuat script AnalyticsManager yang akan menembakkan Events dan UserProperties Analytics.



```
using Firebase.Analytics;

public static class AnalyticsManager
{
    private static void LogEvent (string eventName, params Parameter[]
parameters)
    {
        // Method utama untuk menembakkan Firebase
        FirebaseAnalytics.LogEvent (eventName, parameters);
    }

    public static void LogUpgradeEvent (int resourceIndex, int level)
    {
        // Kita memakai Event dan Parameter yang tersedia di Firebase (tidak
memakai yang custom)
        // agar dapat muncul sebagai report data di Analytics Firebase
        LogEvent (
            FirebaseAnalytics.EventLevelUp,
            new Parameter (FirebaseAnalytics.ParameterIndex,
resourceIndex.ToString ()),
            new Parameter (FirebaseAnalytics.ParameterLevel, level)
        );
        // Karena resourceIndex digunakan sebagai ID, maka seharusnya kita
menyimpannya
        // sebagai string bukan integer
    }
}
```

```
public static void LogUnlockEvent (int resourceIndex)
{
    LogEvent (
        FirebaseAnalytics.EventUnlockAchievement,
        new Parameter (FirebaseAnalytics.ParameterIndex,
resourceIndex.ToString ())
    );
}

public static void SetUserProperties (string name, string value)
{
    FirebaseAnalytics.SetUserProperty (name, value);
}
}
```

b. Track Analytics Log Events and User Properties

- Tambahkan pada script ResourceController untuk memanggil fungsi dari AnalyticsManager.

```
// ...
public void UpgradeLevel ()
{
    double upgradeCost = GetUpgradeCost ();
    if (UserDataManager.Progress.Gold < upgradeCost)
    {
        return;
    }

    GameManager.Instance.AddGold (-upgradeCost);
    _level++;

    ResourceUpgradeCost.text = $"Upgrade Cost\n{ GetUpgradeCost () }";
    ResourceDescription.text = $" { _config.Name } Lv. { _level } \n+ {
GetOutput ().ToString ("0") }";
    AnalyticsManager.LogUpgradeEvent (_index, _level);
}

public void UnlockResource ()
{
    double unlockCost = GetUnlockCost ();
    if (UserDataManager.Progress.Gold < unlockCost)
    {
        return;
    }
}
```

```

SetUnlocked (true);

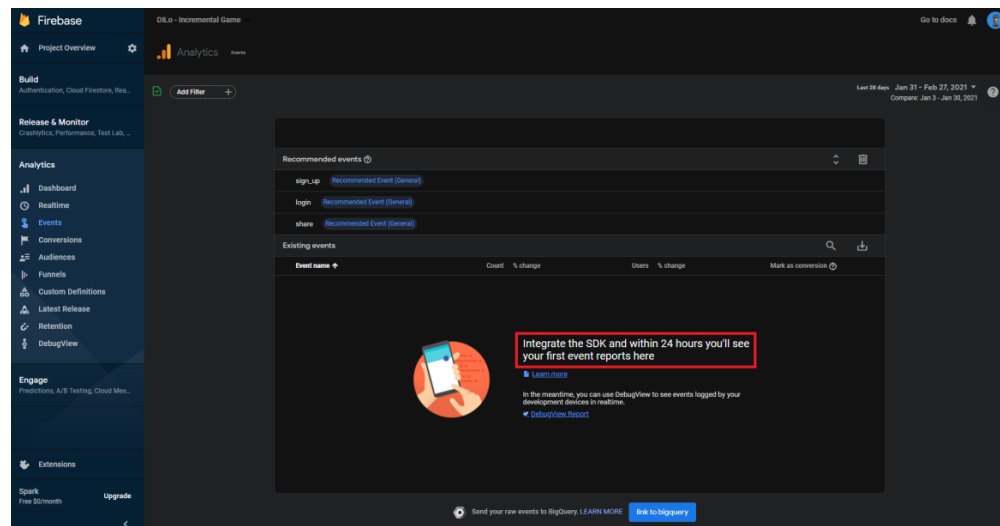
GameManager.Instance.ShowNextResource ();

AchievementController.Instance.UnlockAchievement
(AchievementType.UnlockResource, _config.Name);

AnalyticsManager.LogUnlockEvent (_index);
}
// ...

```

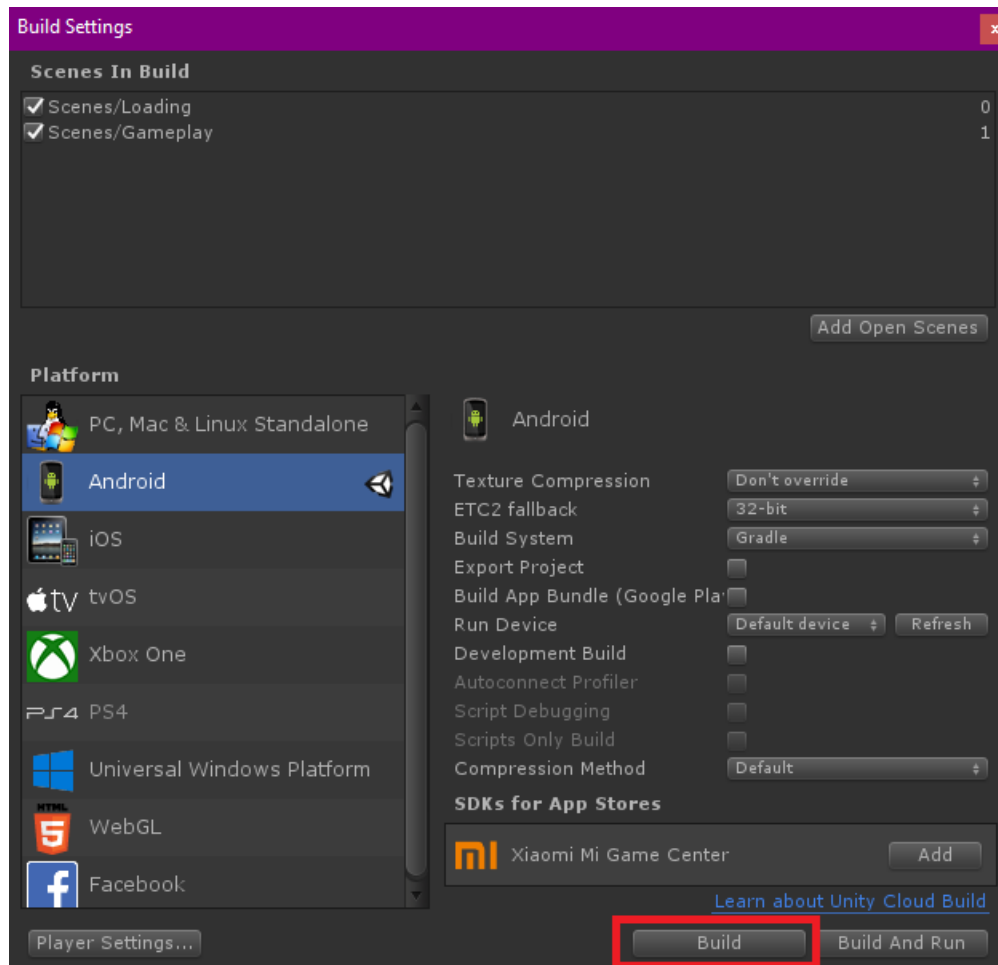
- Events yang sudah tertembak akan muncul setelah 24 jam setelah ditembakkan pada page Firebase Console kalian, lihat pada bagian.



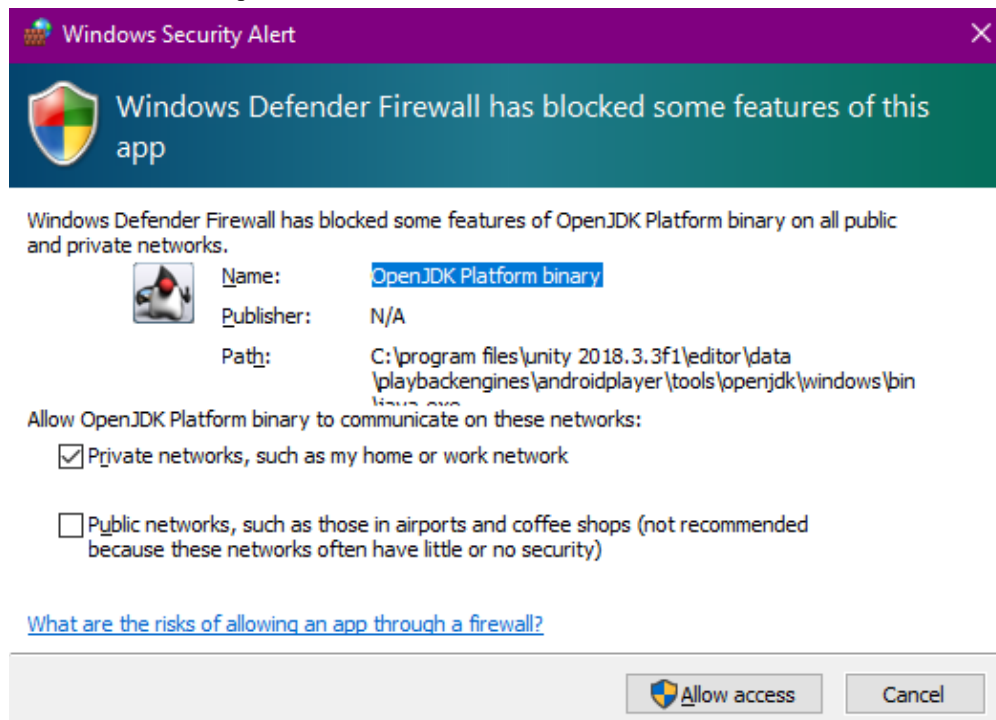
- Untuk User Properties, kita hanya akan menambahkan gold sebagai User Property. Tambahkan fungsi SetUserProperties pada UserDataManager setiap cloud save dilakukan agar tidak terlalu banyak menembakkan analytics dan lebih sesuai dengan data yang tersimpan di cloud.

```
// ...  
  
public static void Save (bool uploadToCloud = false)  
{  
    string json = JsonUtility.ToJson (Progress);  
    PlayerPrefs.SetString (PROGRESS_KEY, json);  
  
    if (uploadToCloud)  
    {  
        AnalyticsManager.SetUserProperties ("gold",  
Progress.Gold.ToString ());  
  
        byte[] data = Encoding.Default.GetBytes (json);  
        StorageReference targetStorage = GetTargetCloudStorage ();  
  
        targetStorage.PutBytesAsync (data);  
    }  
}  
  
// ...
```

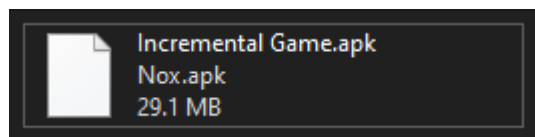
- Analytics tidak bekerja pada Editor unity, tidak seperti Storage, jadi kalian perlu build game kalian ke .apk Android. Klik menu File à Build Settings. Lalu klik Build dan pastikan internet kalian terkoneksi, karena biasanya pertama kali proses build memerlukan koneksi untuk update-update library dari Firebase yang digunakan.



- Jika muncul pop up berikut, silahkan di-Allow access saja, pop up ini muncul karena kalian menggunakan OpenJDK dan ini juga membutuhkan koneksi.

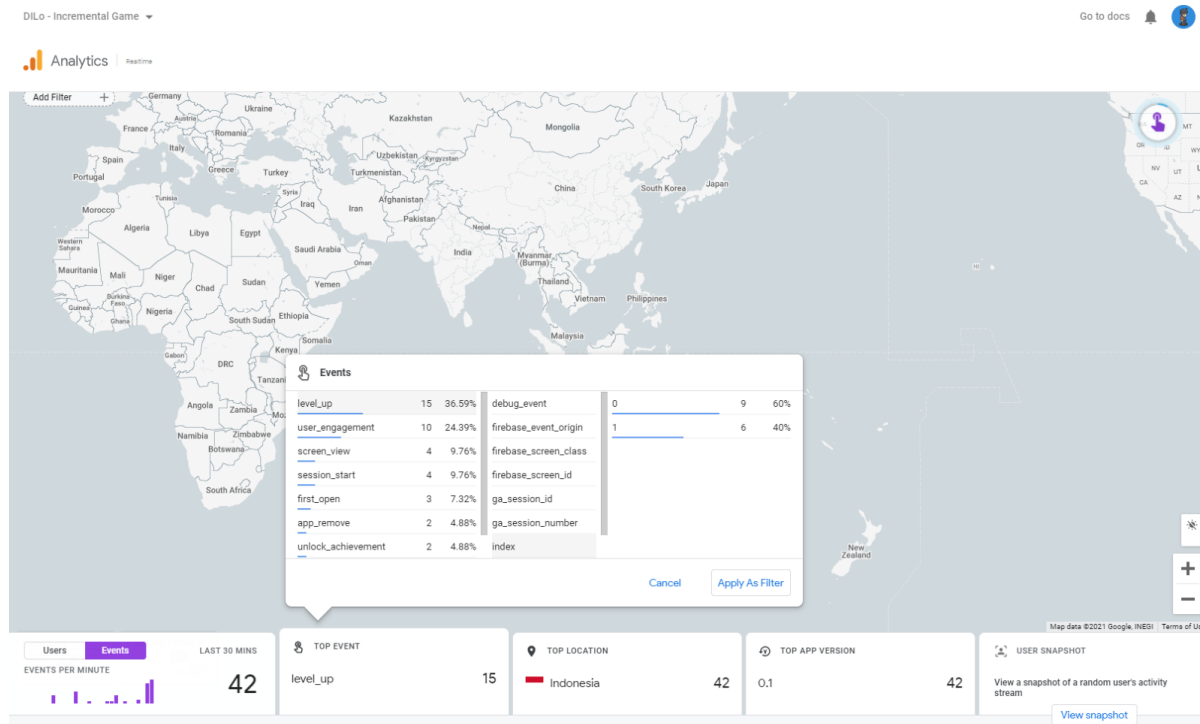


- Setelah menjadi .apk, masukkan file tersebut ke dalam device android kalian dan install.



- Untuk mengetes Events dan User Properties yang sudah ditembakkan sementara kalian dapat melihat pada menu Realtime. Di bagian bawah terdapat data berikut.

1. Events:



## 2. User Properties:

