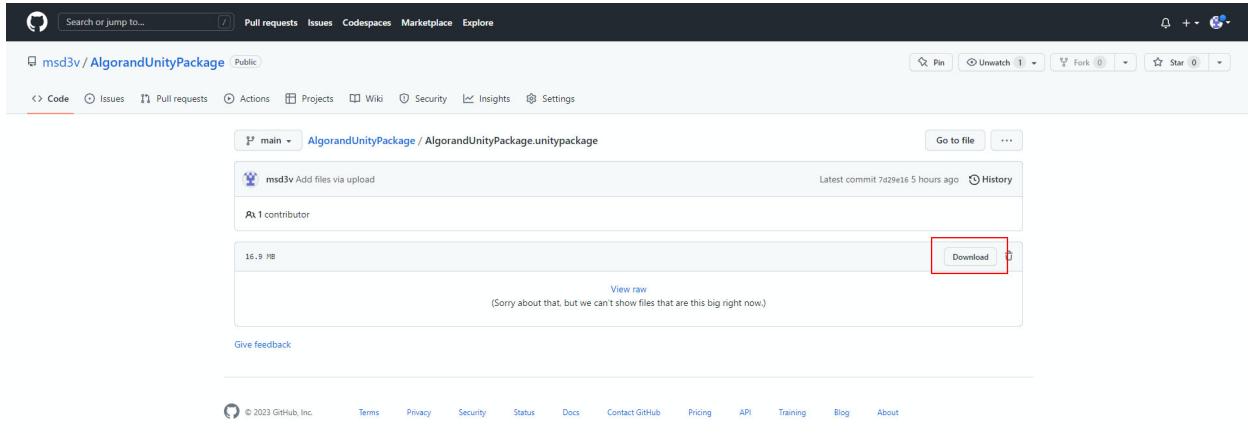
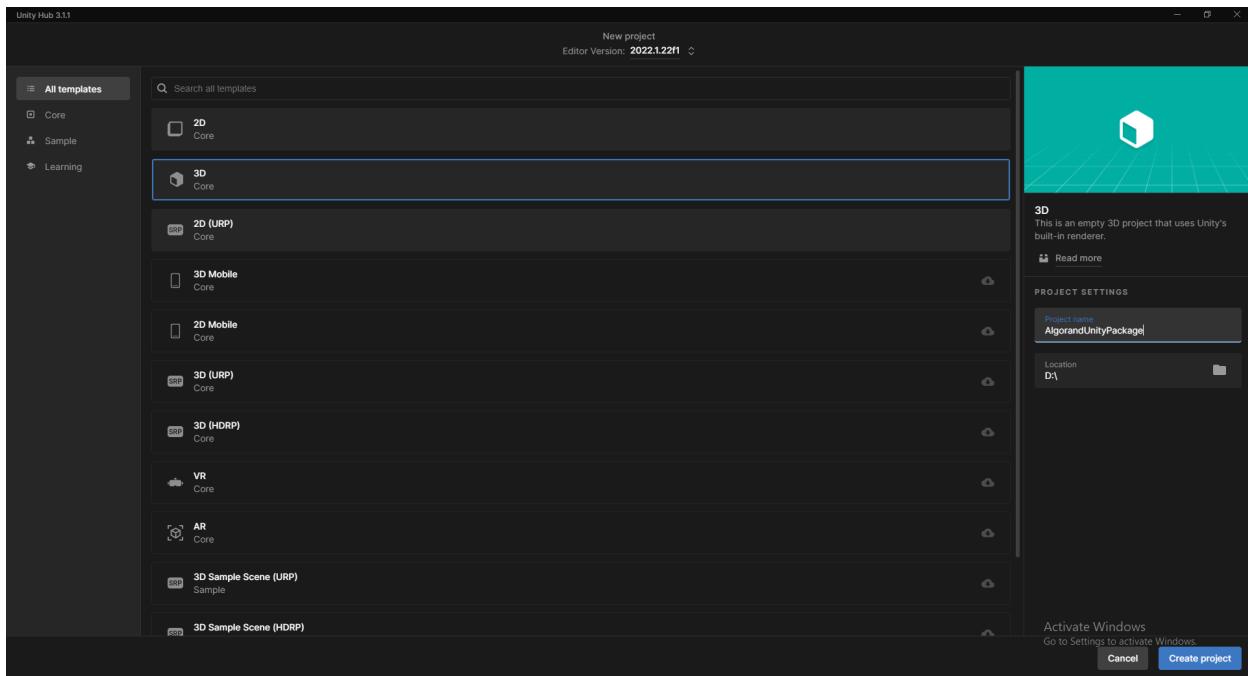


1. Go to:

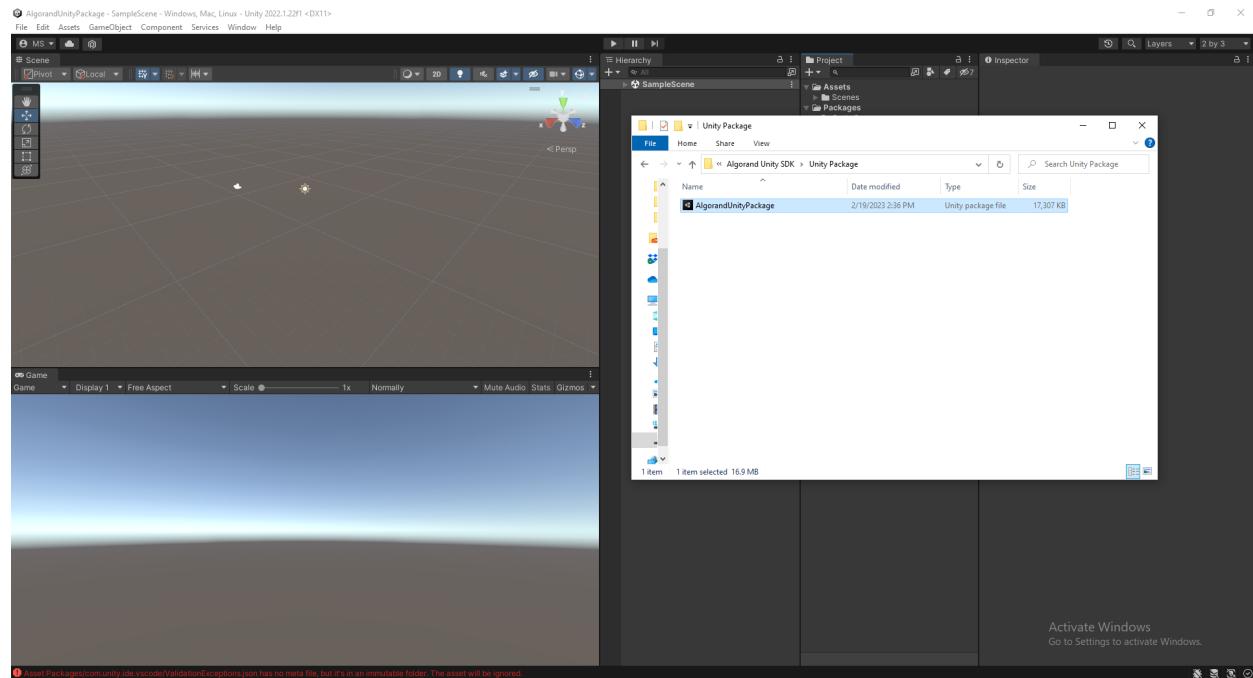
<https://github.com/msd3v/AlgorandUnityPackage/blob/main/AlgorandUnityPackage.unitypackage> and download the Algorand Unity package:



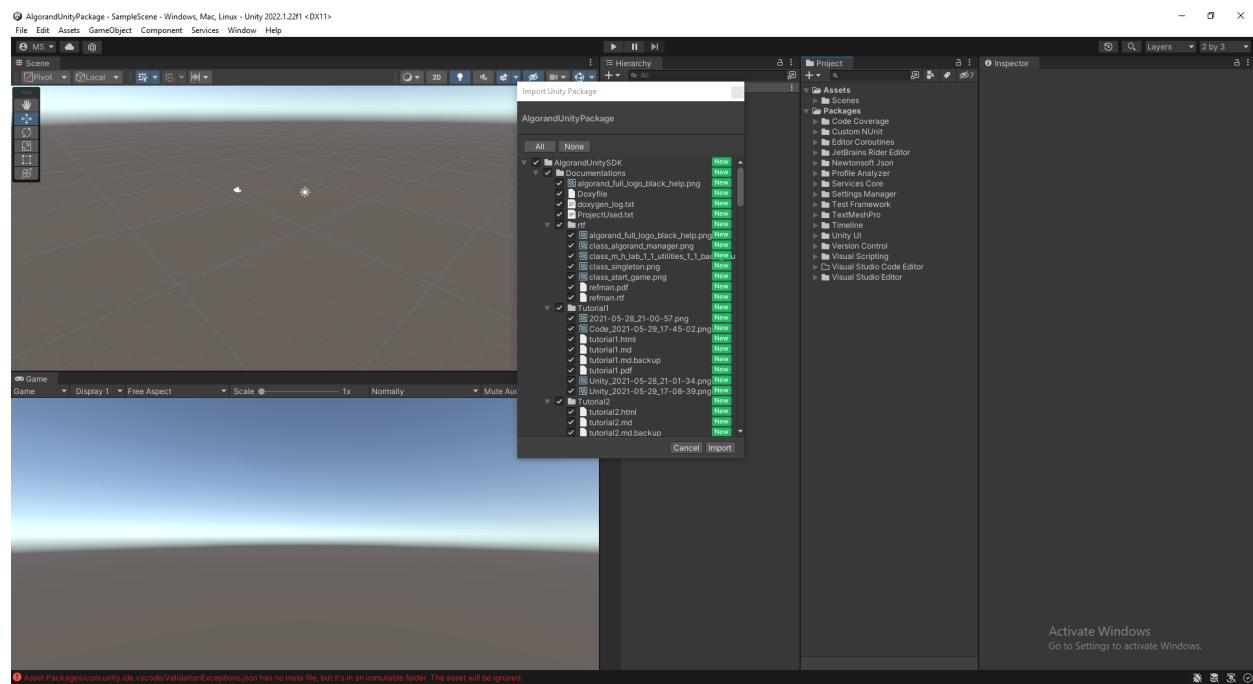
2. Create New Unity project



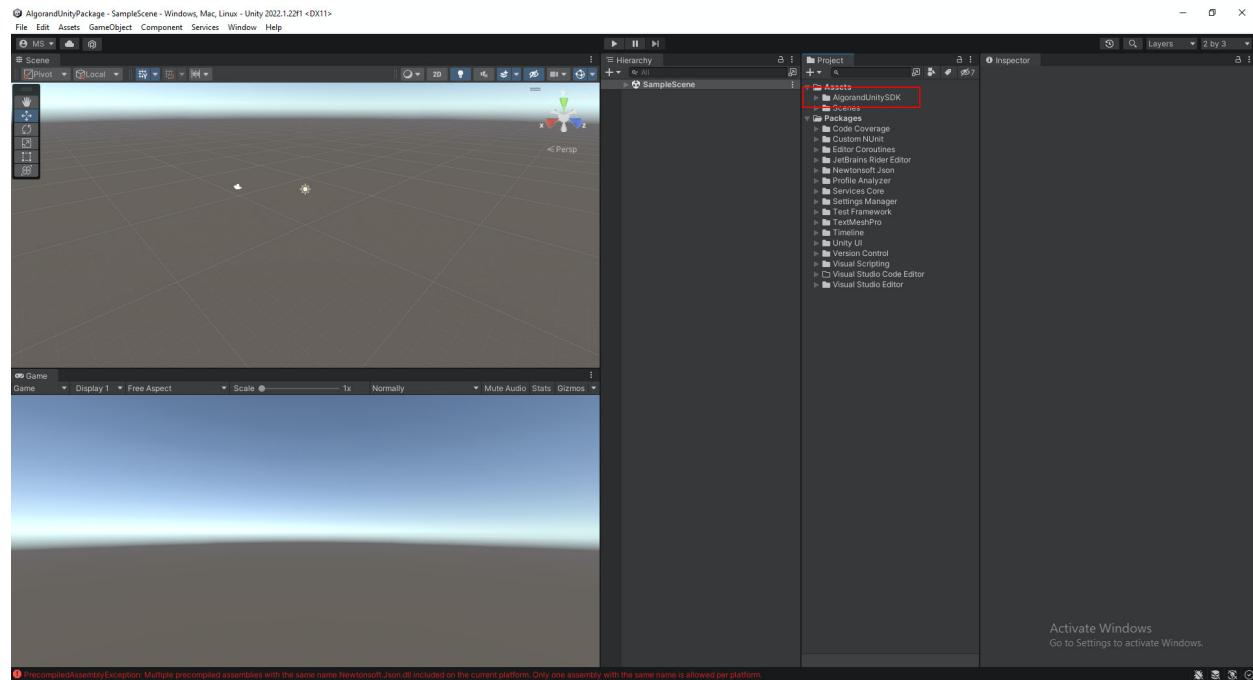
3. Drag n Drop AlgorandUnity package to the project



Click 'Import'

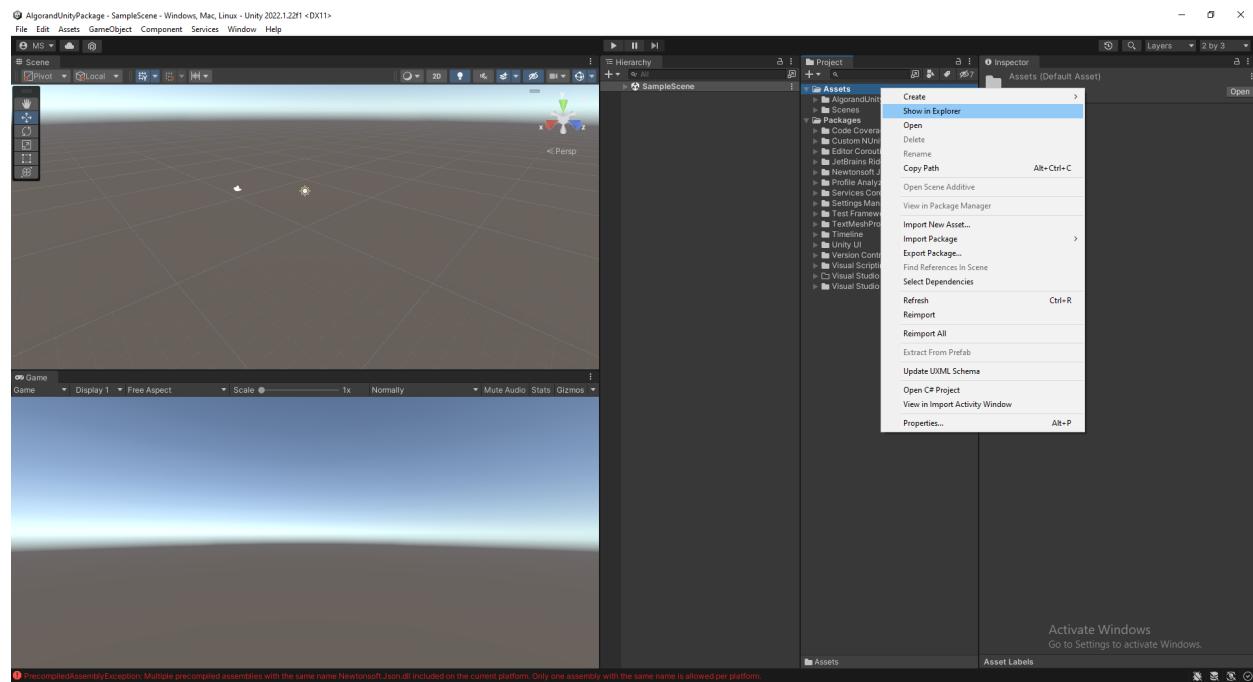


AlgorandUnitySDK imported !!!

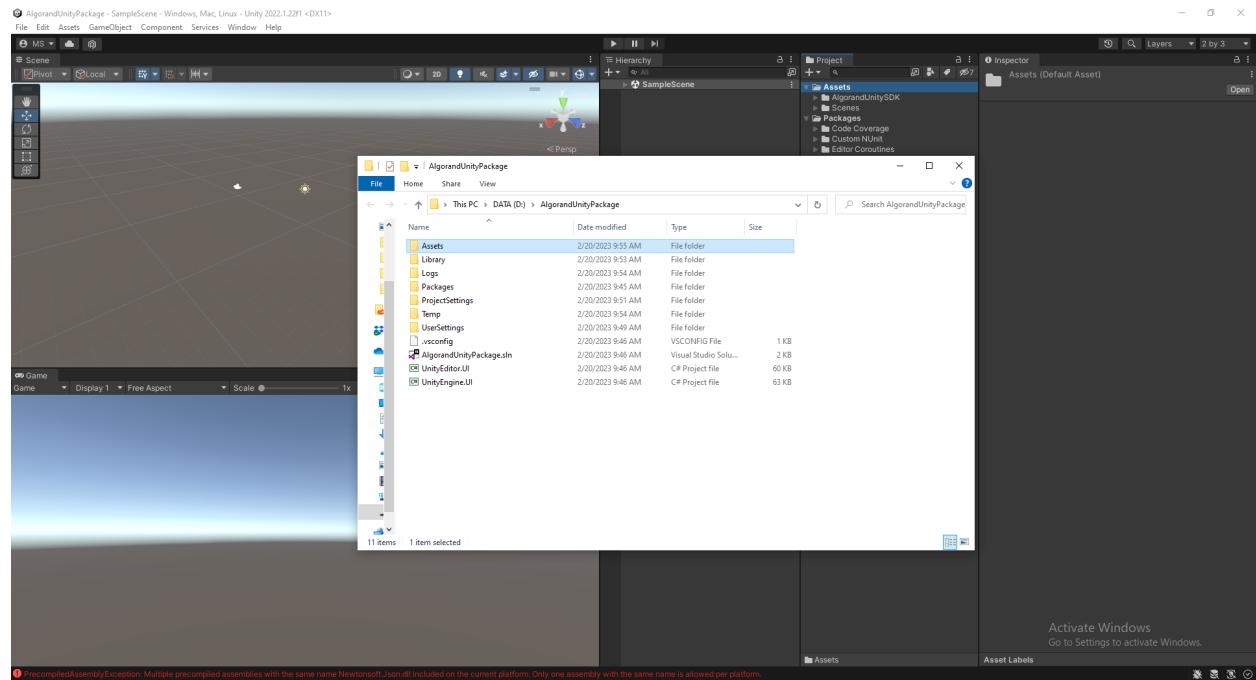


In this phase, you will see couple errors related with 'newtonsoft' and others.

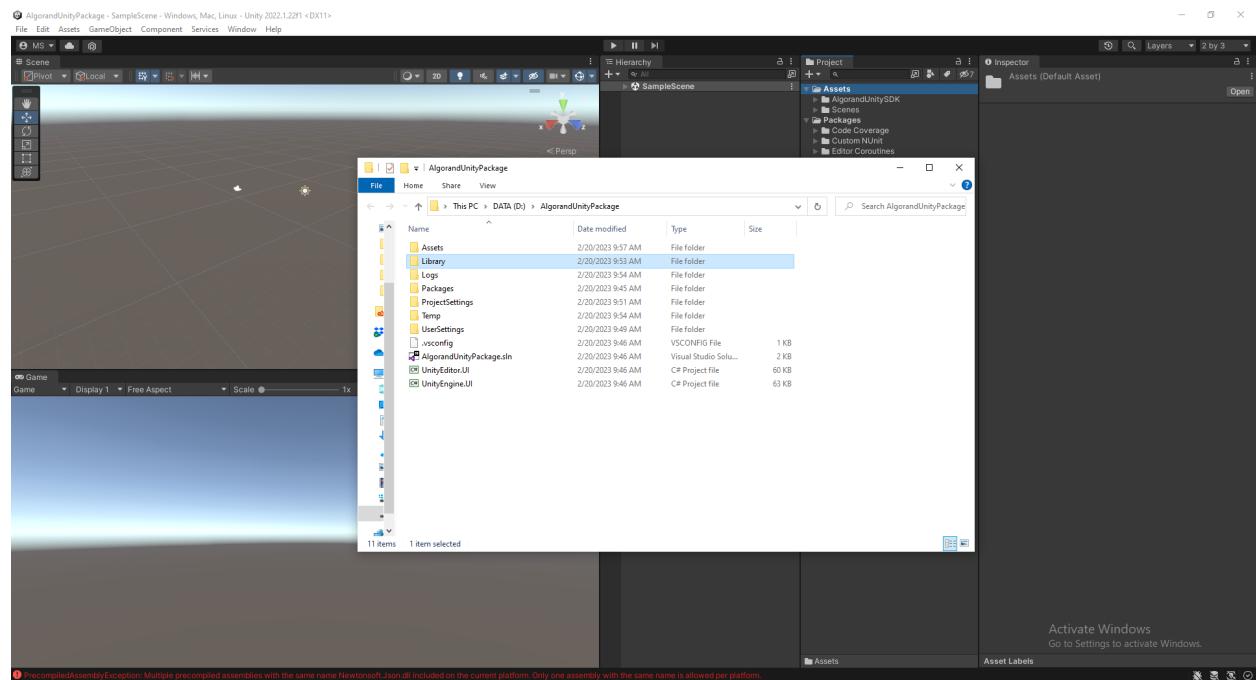
4. To solve these errors, right-click Assets then select 'Show in Explorer' (WIN) or 'Reveal in Finder' (MAC):



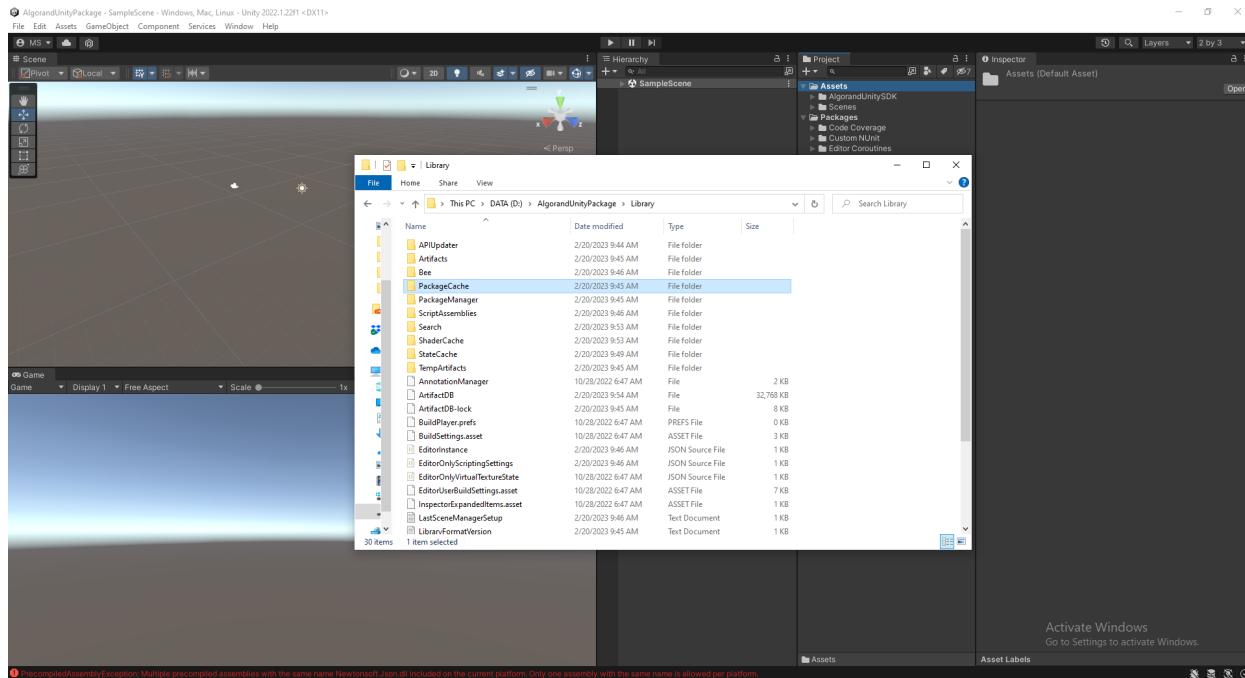
And this window will open:



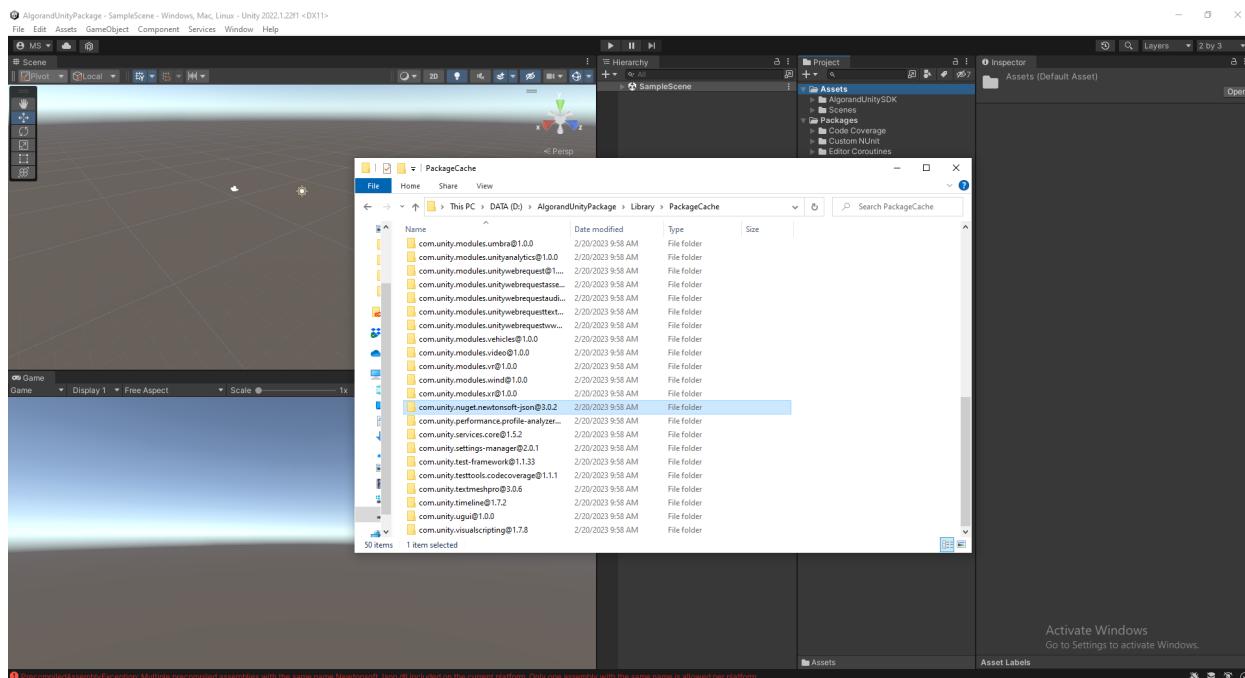
Now go to 'Library'



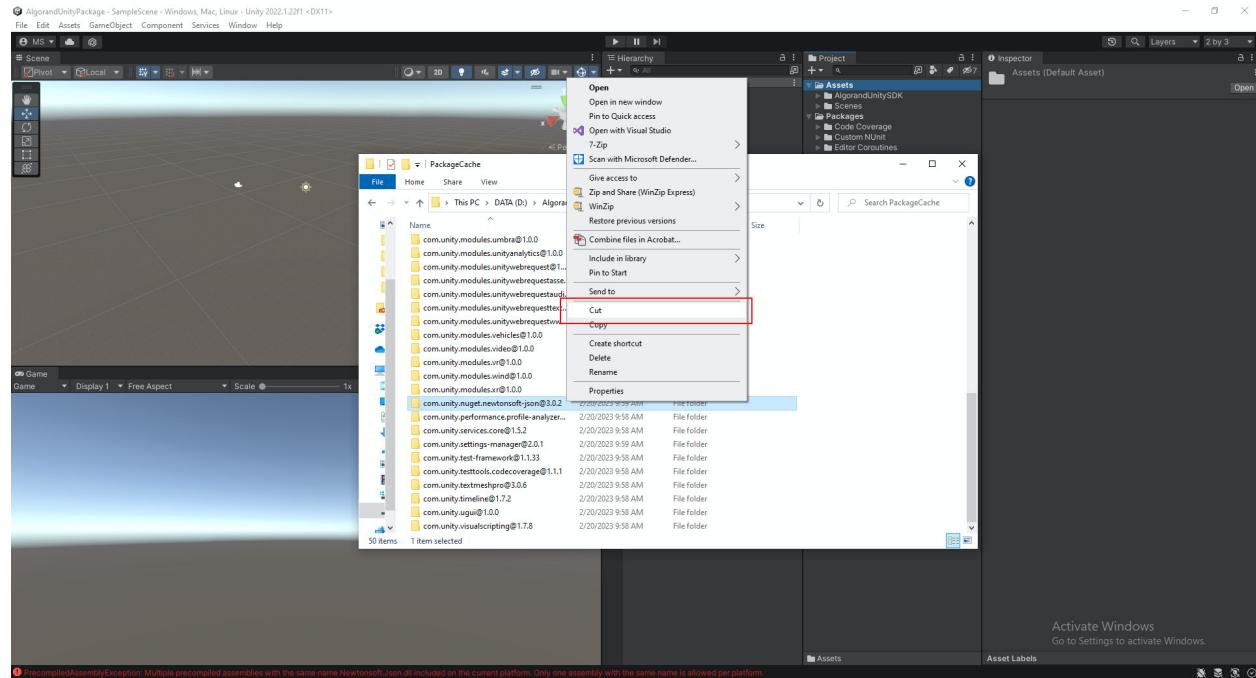
Open ‘Package Cache’



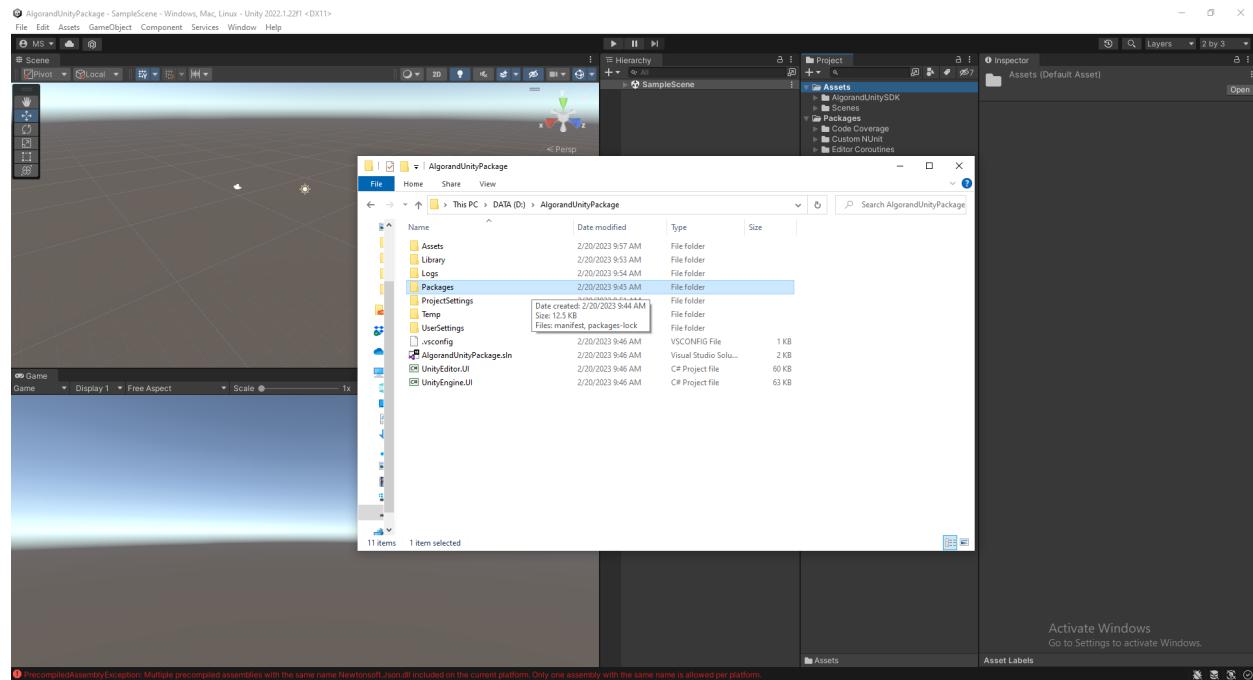
and find 'com.unity.nuget.newtonsoft-json' folder



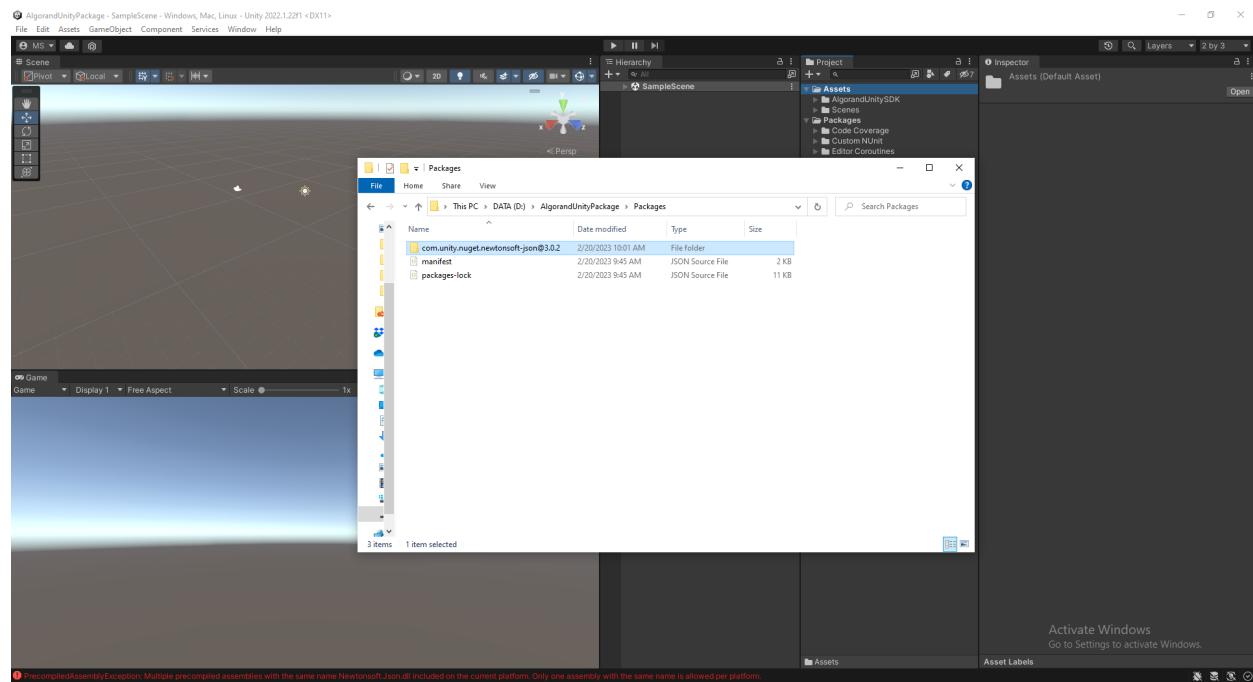
then 'Cut' (ctrl X) the whole folder



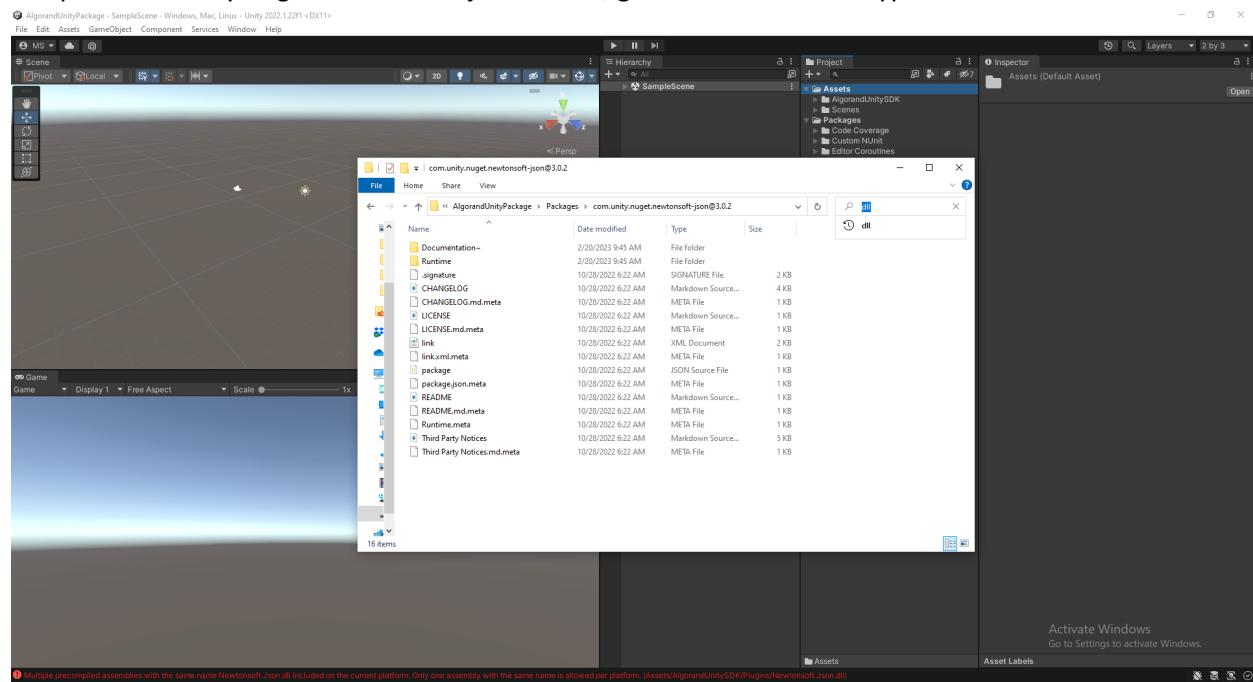
5. back to main folder, and find 'Packages', open it.....



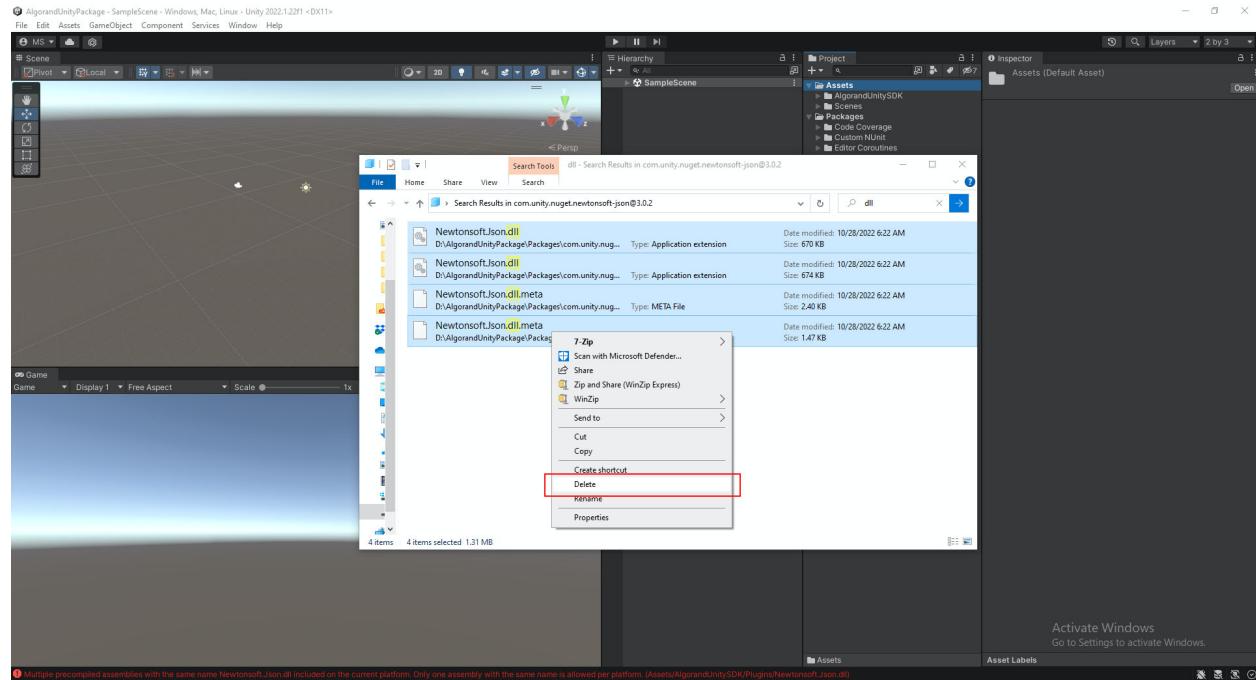
paste 'com.unity.nuget.newtonsoft-json' folder here.



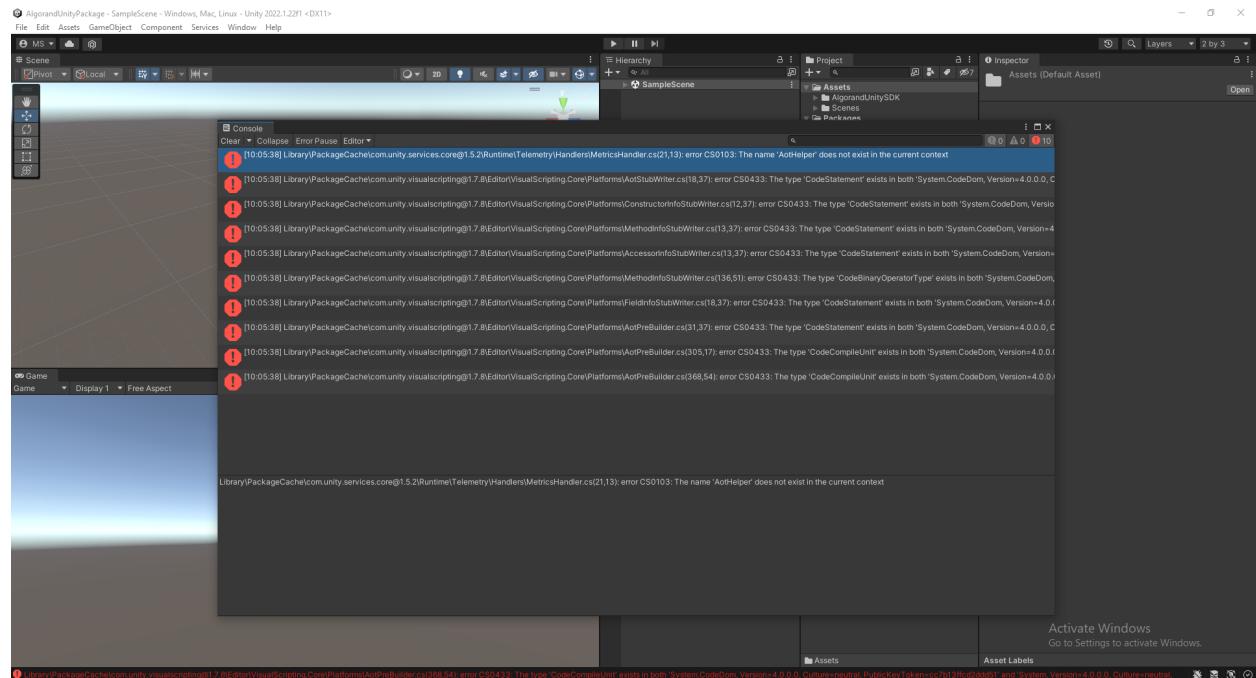
6. Open 'com.unity.nuget.newtonsoft-json' folder, go to search bar and type 'dll'



Select all these files and delete all



7. back to Unity, all newtonsoft-related errors should disappear. But now, we got couple other errors for i.e. 'AotHelper' and 'Visual Scripting' errors. Let's working on it.....

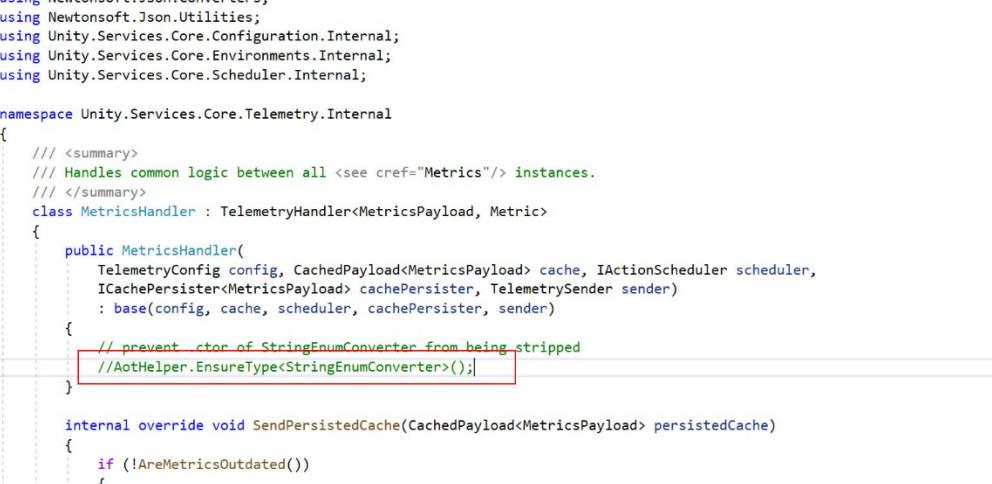


8. Follow these steps to remove 'AotHelper'-related errors. Open Console, double click the 1st AotHelper error. Once Visual Studio opened, disable this line by adding '//' in front of the code.

The screenshot shows the Unity Editor's code editor with the MetricsHandler.cs file open. The code is part of the Unity.Services.Core.Telemetry.Internal namespace. It defines a MetricsHandler class that implements TelemetryHandler<MetricsPayload, Metric>. The constructor takes a TelemetryConfig, CachedPayload<MetricsPayload> cache, and IActionScheduler scheduler, and initializes them via base. A note in the code prevents the StringEnumConverter from being stripped by the compiler. The internal SendPersistedCache method checks if metrics are outdated and then sends the payload.

```
1 using System;
2 using Newtonsoft.Json.Converters;
3 using Newtonsoft.Json.Utilities;
4 using Unity.Services.Core.Configuration.Internal;
5 using Unity.Services.Core.Environments.Internal;
6 using Unity.Services.Core.Scheduler.Internal;
7
8 namespace Unity.Services.Core.Telemetry.Internal
9 {
10     /// <summary>
11     /// Handles common logic between all <see cref="Metrics"/> instances.
12     /// </summary>
13     class MetricsHandler : TelemetryHandler<MetricsPayload, Metric>
14     {
15         public MetricsHandler(
16             TelemetryConfig config, CachedPayload<MetricsPayload> cache, IActionScheduler scheduler,
17             ICachePersister<MetricsPayload> cachePersister, TelemetrySender sender)
18             : base(config, cache, scheduler, cachePersister, sender)
19         {
20             // prevent ctor of StringEnumConverter from being stripped
21             //AotHelper.EnsureType<StringEnumConverter>();
22         }
23
24         internal override void SendPersistedCache(CachedPayload<MetricsPayload> persistedCache)
25         {
26             if (!AreMetricsOutdated())
27             {
28                 m_Sender.SendAsync(persistedCache.Payload);
29             }
30         }
31     }
32 }
```

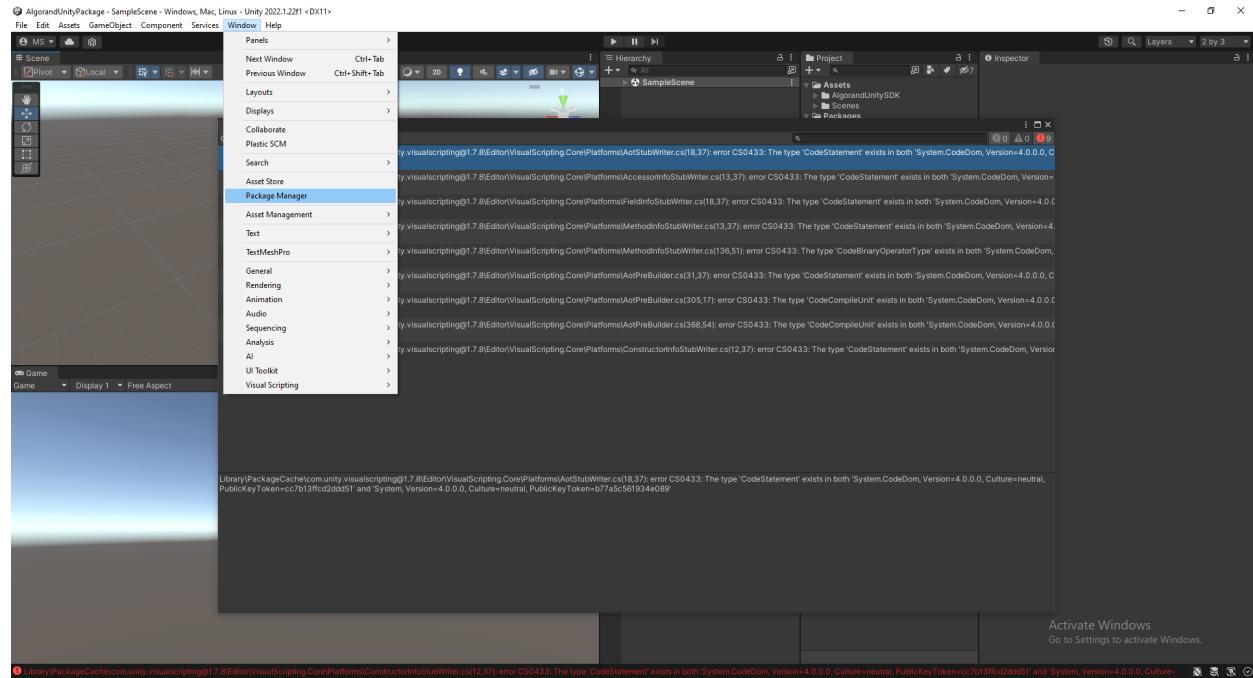
In my case, after I did the 1st step, same AotHelper error still occurred so we can repeat the step. Go to Console, double click the AotHelper error notification, visual studio will open and disable the AotHelper line:



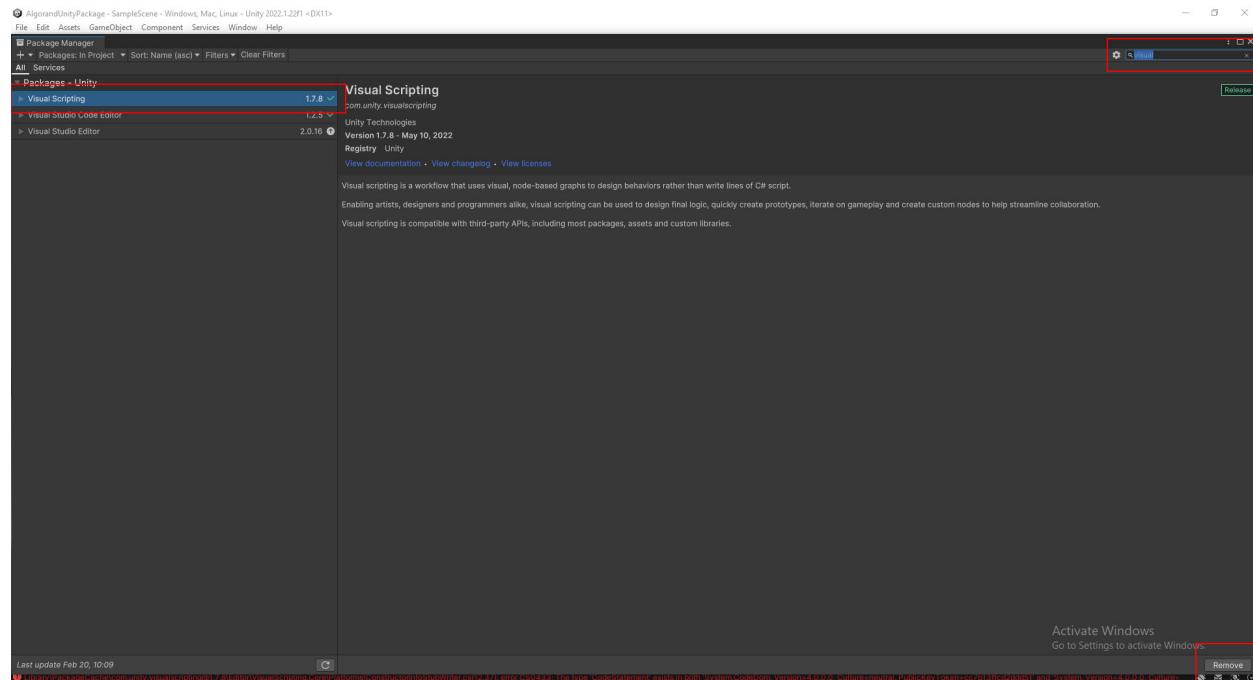
The screenshot shows the Visual Studio IDE with the MetricsHandler.cs file open. The code defines a MetricsHandler class that inherits from TelemetryHandler<MetricsPayload, Metric>. The constructor is annotated with a note to prevent the StringEnumConverter from being stripped. A red box highlights the call to //AutHelper.EnsureType<StringEnumConverter>();

```
1  using System;
2  using Newtonsoft.Json.Converters;
3  using Newtonsoft.Json.Utilities;
4  using Unity.Services.Core.Configuration.Internal;
5  using Unity.Services.Core.Environments.Internal;
6  using Unity.Services.Core.Scheduler.Internal;
7
8  namespace Unity.Services.Core.Telemetry.Internal
9  {
10     /// <summary>
11     /// Handles common logic between all <see cref="Metrics"/> instances.
12     /// </summary>
13     class MetricsHandler : TelemetryHandler<MetricsPayload, Metric>
14     {
15         public MetricsHandler(
16             TelemetryConfig config, CachedPayload<MetricsPayload> cache, IActionScheduler scheduler,
17             ICachePersister<MetricsPayload> cachePersister, TelemetrySender sender)
18             : base(config, cache, scheduler, cachePersister, sender)
19         {
20             // prevent ctor of StringEnumConverter from being stripped
21             //AutHelper.EnsureType<StringEnumConverter>();
22         }
23
24         internal override void SendPersistedCache(CachedPayload<MetricsPayload> persistedCache)
25         {
26             if (!AreMetricsOutdated())
27             {
28                 m_Sender.SendAsync(persistedCache.Payload);
29             }
30         }
31     }
32 }
```

9. To fix Visual Scripting related error is pretty straightforward. Go to Window >> Package Manager



Search for 'visual' and you should find 'Visual Scripting' and hit 'Remove' button



10. Your Algorand-based Unity project should be ready to go !!!

Unity Package Source: <https://github.com/Vytek/AlgorandUnitySDK>

Happy Coding !!!