Collaborating on Okainos

# Unity Teams

Collaborate (built-in tool which allows you to publish projects to the cloud) is a part of Unity Teams. Unity Teams allows small teams to save, share and sync a Unity Project in a cloud-hosted environment.

# GitHub

GitHub is a Git repository hosting service with version control and collaboration. Repository changes can be easily published with the GitHub desktop application.

# Unity Teams VS GitHub

|  |  |  |  |
| --- | --- | --- | --- |
| **Unity Teams** | | **GitHub** | |
| **Advantages** | **Disadvantages** | **Advantages** | **Disadvantages** |
| Very simple to use | Lacks lots of features | Has a lot of features like different branches, issue tracker | Requires a learning curve to understand how to use it |
| Comes built-in with Unity | Can be unstable at times, therefore not making it reliable as a single solution | Very reliable | Requires configuration and installing of programs and tools to use it |
| Can merge Unity scenes | Limited to 3 team seats unless you purchase more with Advanced which comes with Unity Pro | Unlimited collaborators | Can’t merge Unity scenes |
|  | 1GB with Basic, 25GB with Advanced | There is no limit to the number of and size of public and private repositories | However, there is a limit to separate file sizes, being 100MB max. Solutions like Git LFX (a Git extension) can be used for this. |
|  |  | Can link with Trello |  |
|  |  | Can be used along with other programs like SourceTree for visual representation of repos. |  |

# Bitbucket (alternative to GitHub)

Bitbucket is designed to mostly focus on private repos for mostly enterprise and business users while GitHub is designed to mostly focus on public repos with a huge open-source community.

# Using Git with Unity

## The problem

By default, Unity does not work well with Git. Problems include: hundreds of temporary files, project breaking when switching branches since object references that the editor keeps track of are not committed to Git properly via .meta files, unresolvable merge conflicts when editing the same file independently, large files like 3D models, sounds, images and fonts significantly slowing down Git workflow and waste storage space.

## The solution

Add Unity-specific .gitignore settings to ignore changes made to temporary files (a good Unity .gitignore template is provided by GitHub).

Change editor settings: Make .meta files visible to avoid broken object references by setting version control mode to “Visible Meta Files” in Edit > Project Settings > Editor

And use plain text serialisation to a avoid unresolvable merge conflicts by setting asset serialisation mode to “Force Text”.

And finally, use Git LFS to track large files over 100MB with Git while keeping them out of your actual repository. (This only works with servers that support Git LFX API like GitHub). To do this, you need to install the Git LFX command line extension as documented on the [Git LFS site](https://git-lfs.github.com/). Before using it to track file types you’d like Git LFS to manage, you’ll need to use the .gitattributes sample file to account for all the file types that Unity supports (3D models, audio, fonts, images).