Documentation

Introduction

This asset gives the ability to set the Game Version (*ProjectSettings/Player/Version*) to match your project Git Tag🏷. It could be set automatically when Unity builds the project or manually by using the Tools menu. There are various settings available to customize the asset usage.

Table of contents

Version from git lite

⚠ Important notice ⚠

This asset obviously works only if you have git installed and initialized for your Unity project. The version will be created by fetcher the last git tag found on your repo. This tag should follow the format vX.Y.Z or X.Y.Z (X : Major number, Y : Minor number, Z : Revision number).

More information about git tag here:

https://git-scm.com/book/en/v2/Git-Basics-Tagging

More information about software versioning format could be found here:

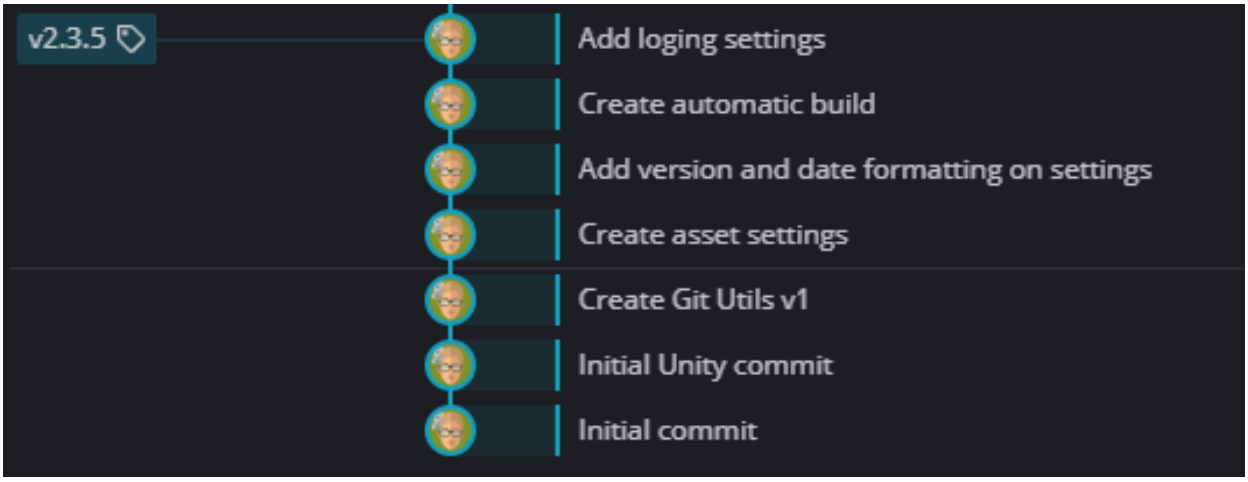
https://semver.org/

https://en.wikipedia.org/wiki/Software_versioning

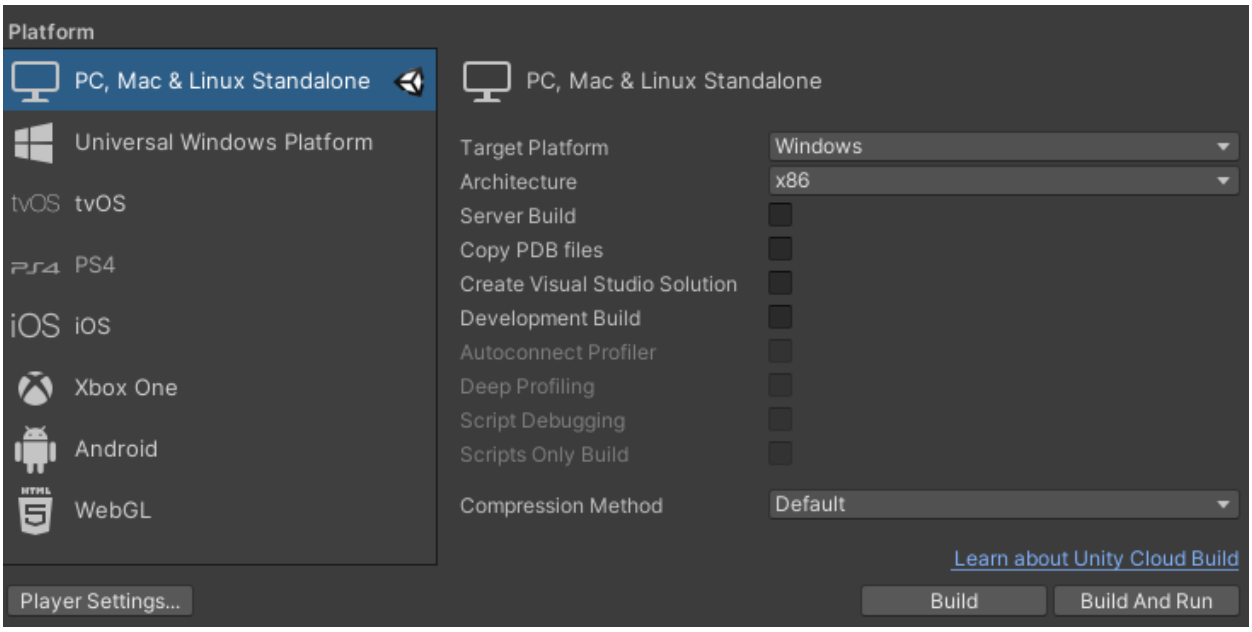
Version from git lite

Quick Start

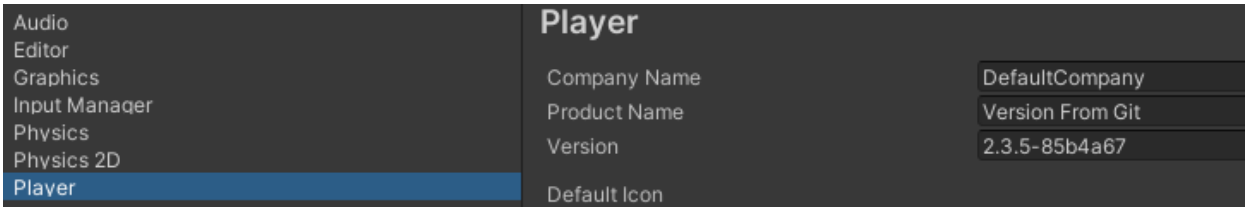
1. Create a tag on your Git repo on format “vX.Y.Z”.



2. Build your game



3. And the version will be set to the Unity Player Version field!



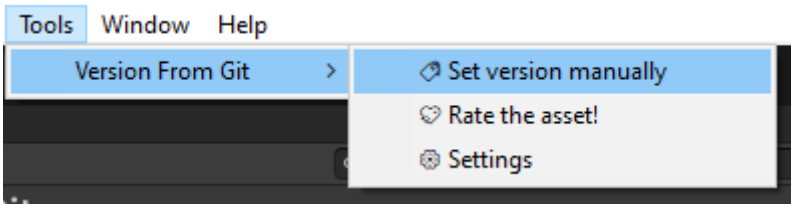
Usage

Automatically

If you have enabled the “Automatic On Build” setting the version will be set before each build. It will use the format settings for the version and the date. You don’t have to do anything.

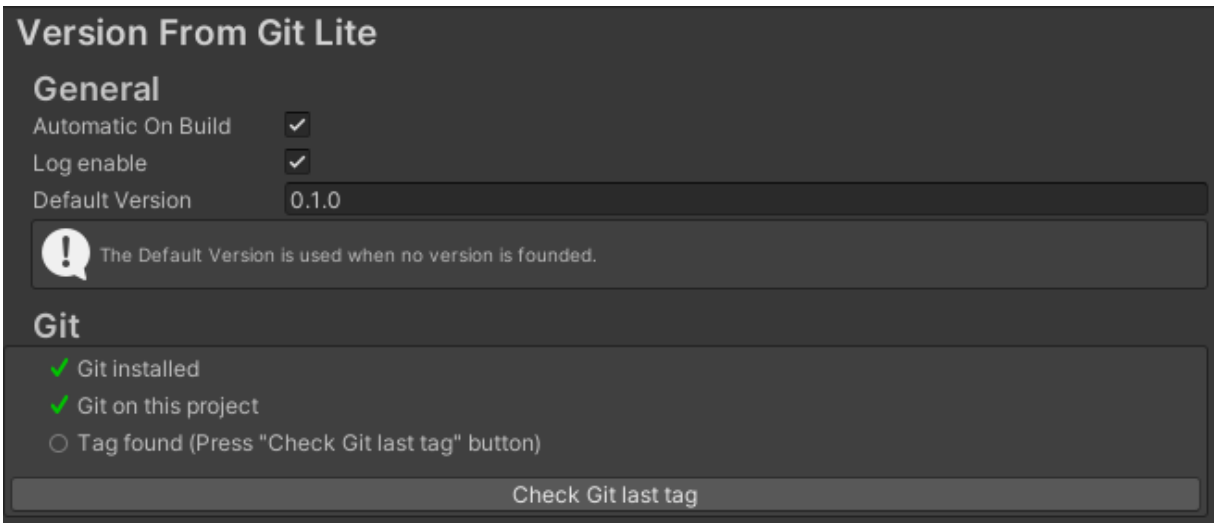
Manually

If you want to set the game version to the git version, you can use the option found on the Unity top menu “Tools/Version From Git/Set version manually”.



Settings

Asset's settings could be found on "ProjectSettings/Version From Git Lite" or by using the top menu "Tools/Version From Git Lite/Settings"



General settings

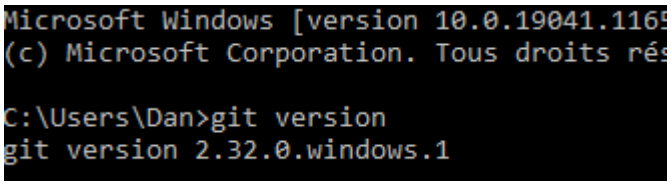
Automatic On Build: If you want the Version to be fetch and assign on every Build. Work when building with command line too.

Log Enable: When assigning tag or using the asset it will log some information about git fetching tag or git availability. It could log some error breaking the build if no tag or no git is found. Disabled this option if you don't want to see any log or error.

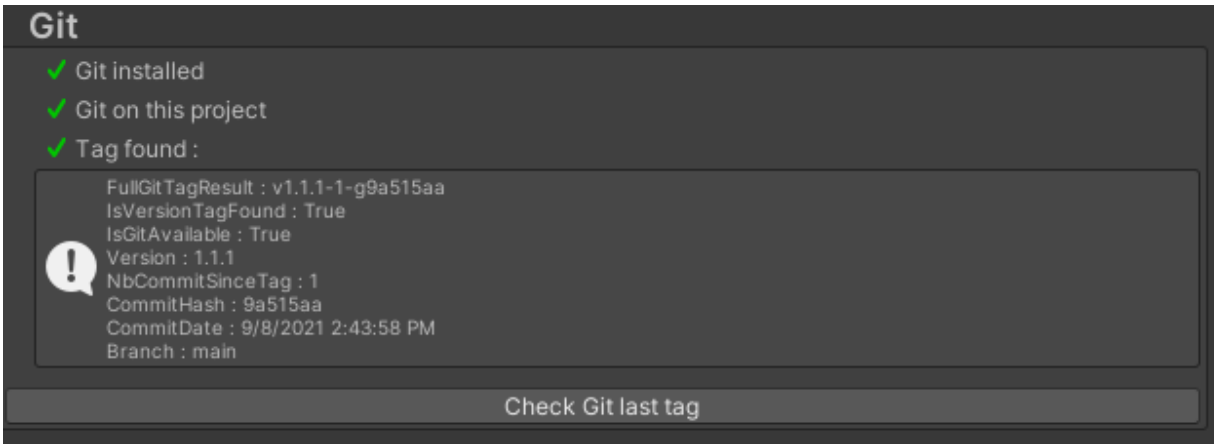
Default Version: When no git tag is found the default version is used. You can customize it.

Git settings

This section resumes if Git was found on your computer and if it's initialized for the current project. You should have two green checkmarks here. Otherwise, you should be sure that Git is installed and initialized for the current project. See the Git documentation for any further information about installing Git. For users on windows, be sure to have the Git path configured on your environment variable (don't forget to restart your computer after adding it). More information can be found on Google. If you want to check if the path is correctly installed just type "git version" on a command prompt and you should get this kind of output:



You can use the "Check Git last tag" button to check the git availability and the last tag. You'll get a full report looking like this:



ℹ Note: Settings are saved as a *ScriptableObject* in the *Assets/Editor/VersionFromGitSettings.asset* file. Deleting this file will reset the settings and a new file will be created at the same location.

API usage

If you want to handle the formatting of the version yourself, you can disable the "Automatic On Build" and creating your own "IPreprocessBuildWithReport" script to handle the version. You could retrieve all git information and format it as you wish with your custom logic.

GitData

This class can be created by using static method:

1. GitData.GetCurrentGitData(): Will have the last tag information and the current branch name.
2. GitData.GetExample(): Will give you an example of data who don't have any link with your project. You should not use this one.

You retrieve the current GitData information of your project like this:

```
GitData data = GitData.GetCurrentGitData();
```

This is the common way to assign the game version by script:

```
PlayerSettings.bundleVersion = version;
AssetDatabase.SaveAssets();
```

Custom IPreprocessBuildWithReport example

Here's an example/model how you can handle automatic versioning with your own logic for formatting it:

```
using NotInvited.VersionFromGit.Editor.Git;

public class ExampleVersionOnBuild : IPreprocessBuildWithReport
{
   public int callbackOrder { get; }
   public void OnPreprocessBuild(BuildReport report)
   {
       GitData data = GitData.GetCurrentGitData();

       PlayerSettings.bundleVersion = $"{data.Version.ToString()}-{data.CommitHash}-
       Preview-Android";
       AssetDatabase.SaveAssets();
    }
}
```

Version from git lite

Support

📧 You can write to notinvitedgames@gmail.com if you have any question or suggestion. I'm always looking for improving my asset and will taking in consideration any good idea! I speak English and French.

💖 Thank you for buying this asset. If you are happy with it, feel free to rate it on the asset store 🔗https://assetstore.unity.com/packages/slug/203089#reviews

Pro Version

If you like this asset and you can afford it, you can buy the Pro version of this asset who add possibility to fully customize the version and the date format. This is the best way to support me !

🔗https://assetstore.unity.com/packages/tools/utilities/version-from-git-203089?aid=1011lkCc4

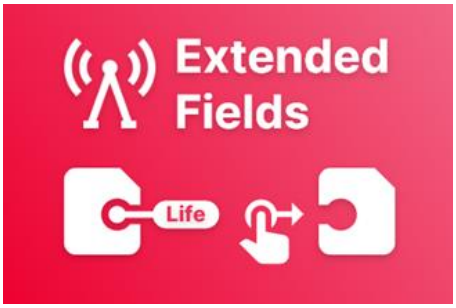
V1.0.0

Version from git lite

More from Not Invited

Find all my assets on my publisher page. Here's some of my other asset you may like to check out:

Extended fields



Extended fields let you easily share properties and fields between script easily.

It will let you use some of inner unity variable in one click.

https://assetstore.unity.com/packages/tools/utilities/extended-fields-199105?aid=1011lkCc4

Editor Tools Pro – Mega bundle

This bundle includes all my assets and all future assets

It's the best choice if you want to have all for a small price.

https://assetstore.unity.com/packages/tools/utilities/editor-tools-pro-mega-bundle-207894?aid=1011lkCc4