ESCOLA DE NOVES TECNOLOGIES INTERACTIVES
centre adscrit a:
UNIVERSITAT DE BARCELONA

Computer Graphics

07. Setup a GIT repo for a C# Unity3D project

Dr Joan Llobera – joanllobera@enti.cat

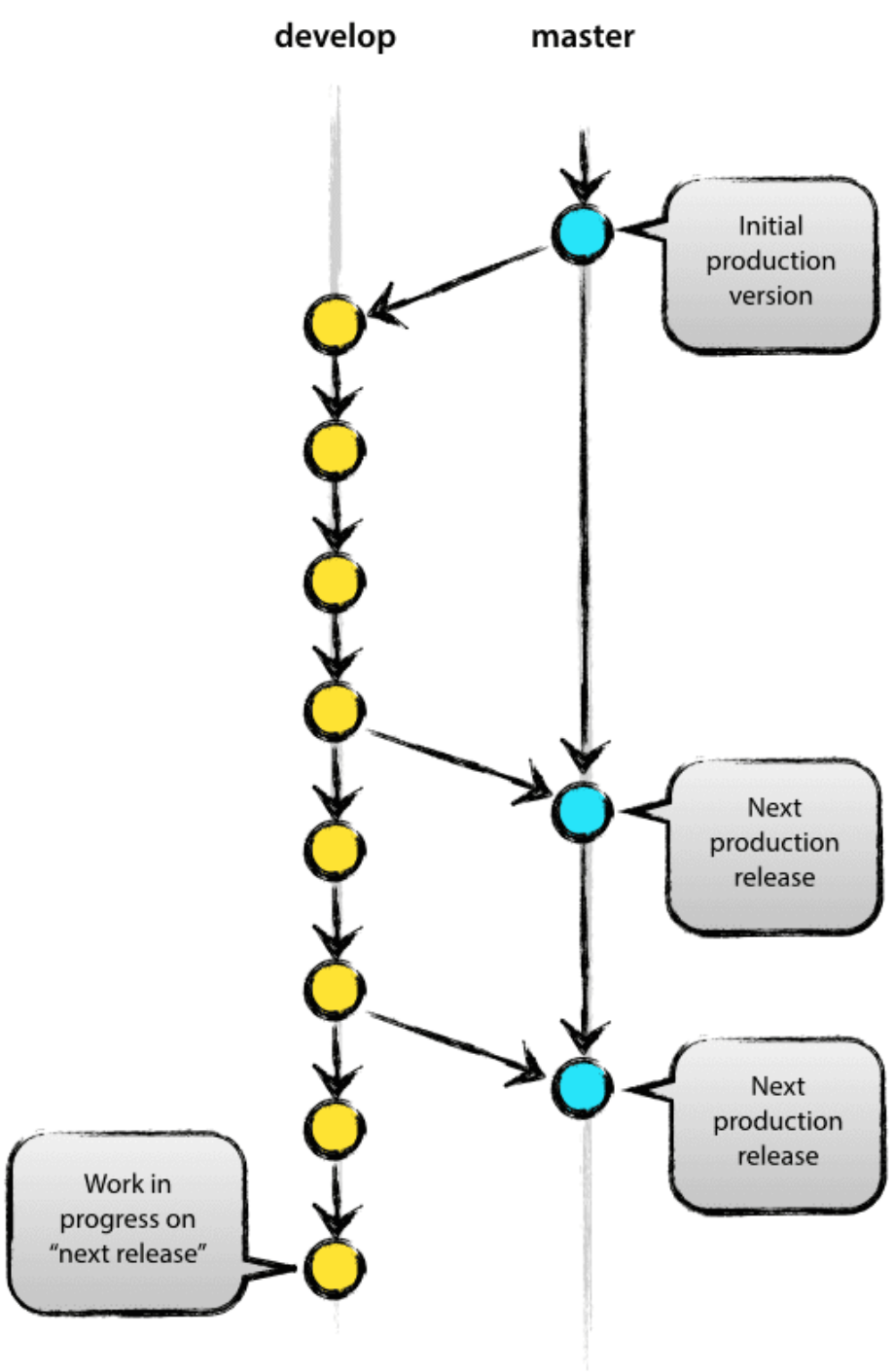
- Git flow: a way to organize your commits
- Naming Conventions
- Scene Conventions
- Repository Configuration

Git flow

It’s a way to organize your commits.

It was proposed by Vincent Driesen in 2010

https://nvie.com/posts/a-successful-git-branching-model/

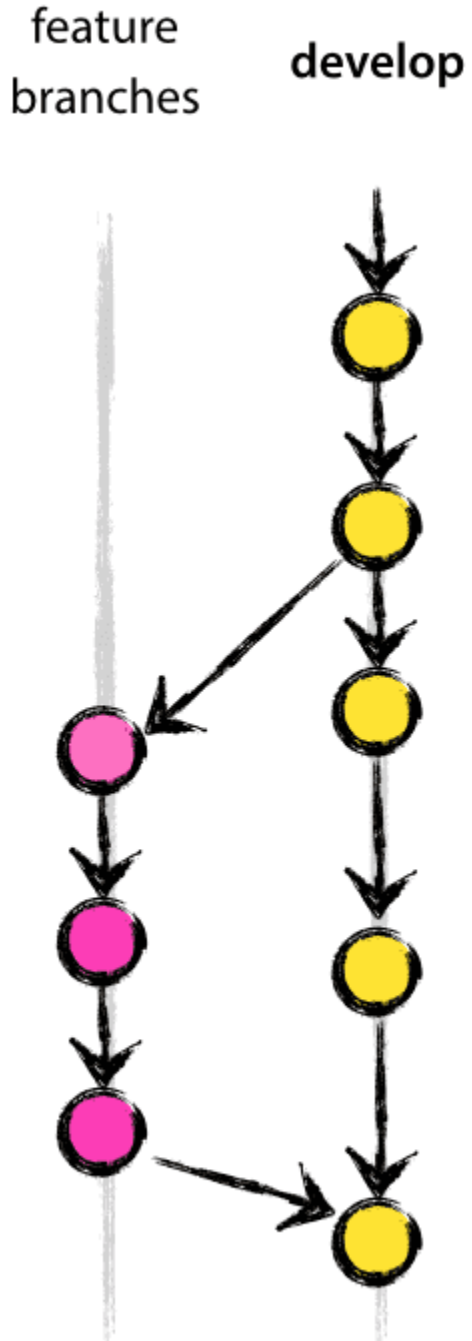


Git flow

It’s a way to organize your commits.

It was proposed by Vincent Driesen in 2010

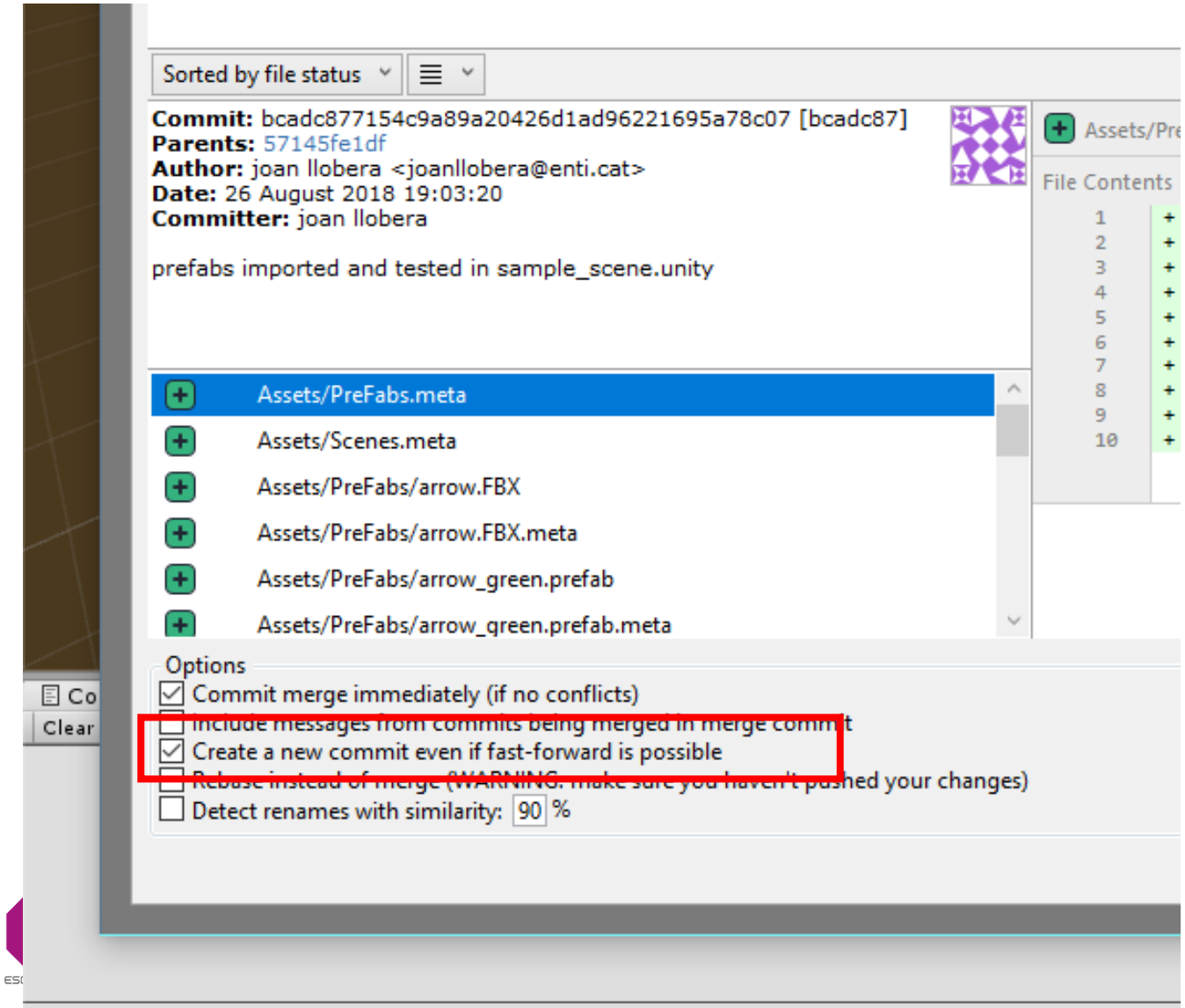
https://nvie.com/posts/a-successful-git-branching-model/

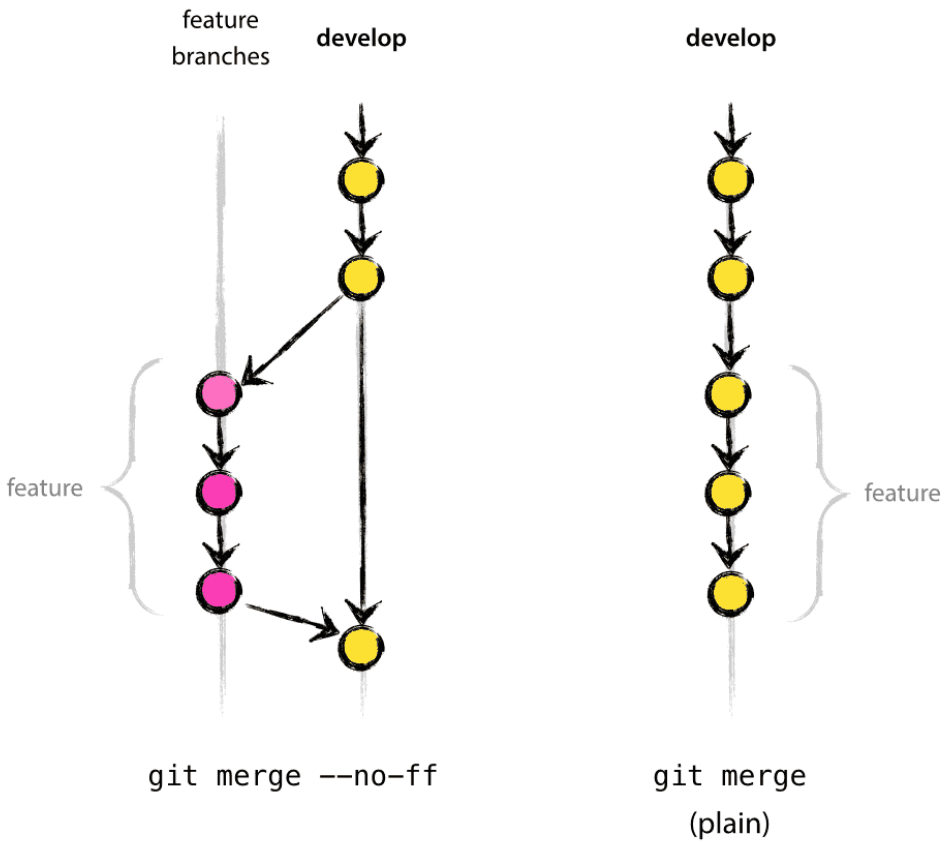


Merge! Some precautions to consider

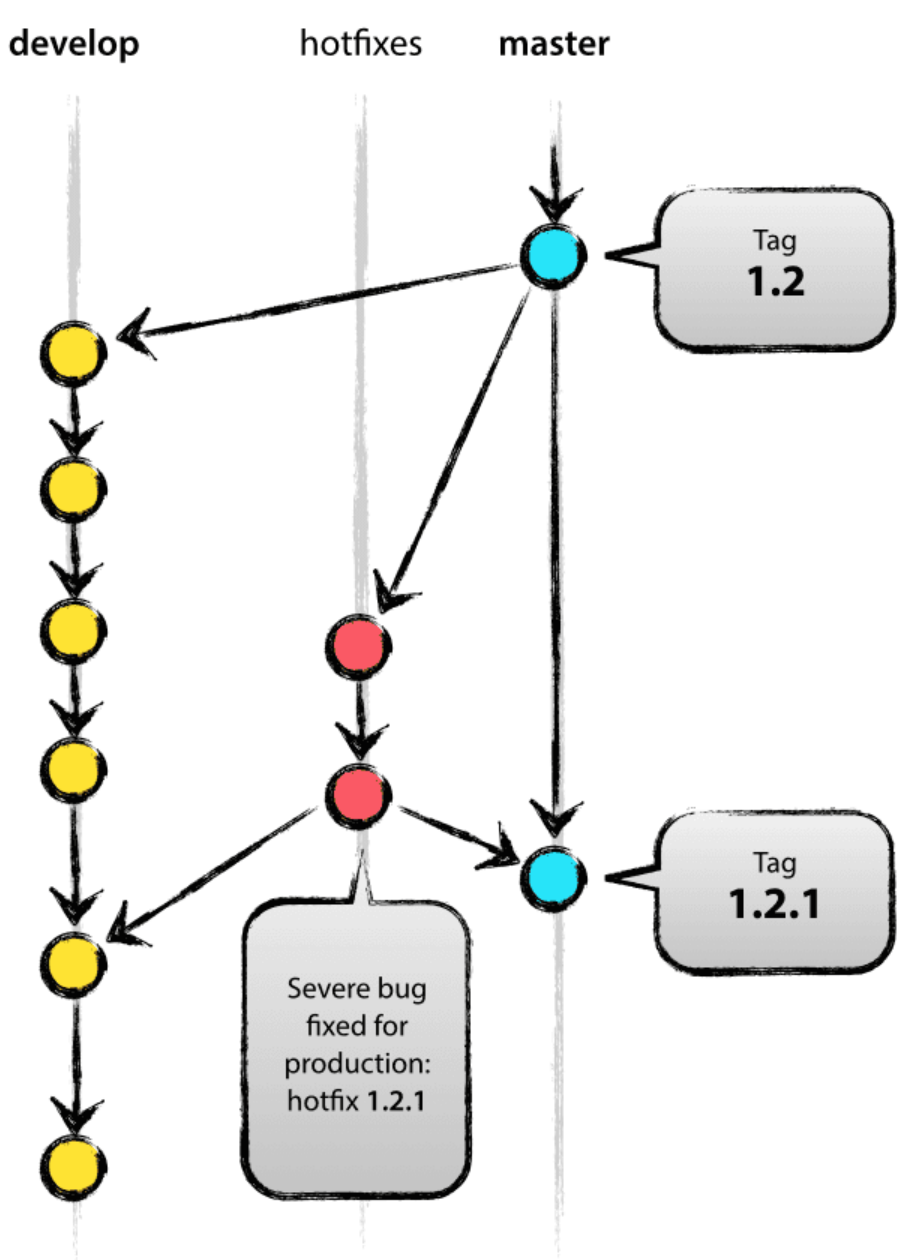
- Merging a feature that has been completed in develop can be done directly
  - ALL commits in develop should involve a working project (no errors)
- Merging a release from develop (or from release branch) to master
  - Needs to go with a TAG
  - Need to be committed as a merge request, for your partner to verify that the solution works, and accept the merge request

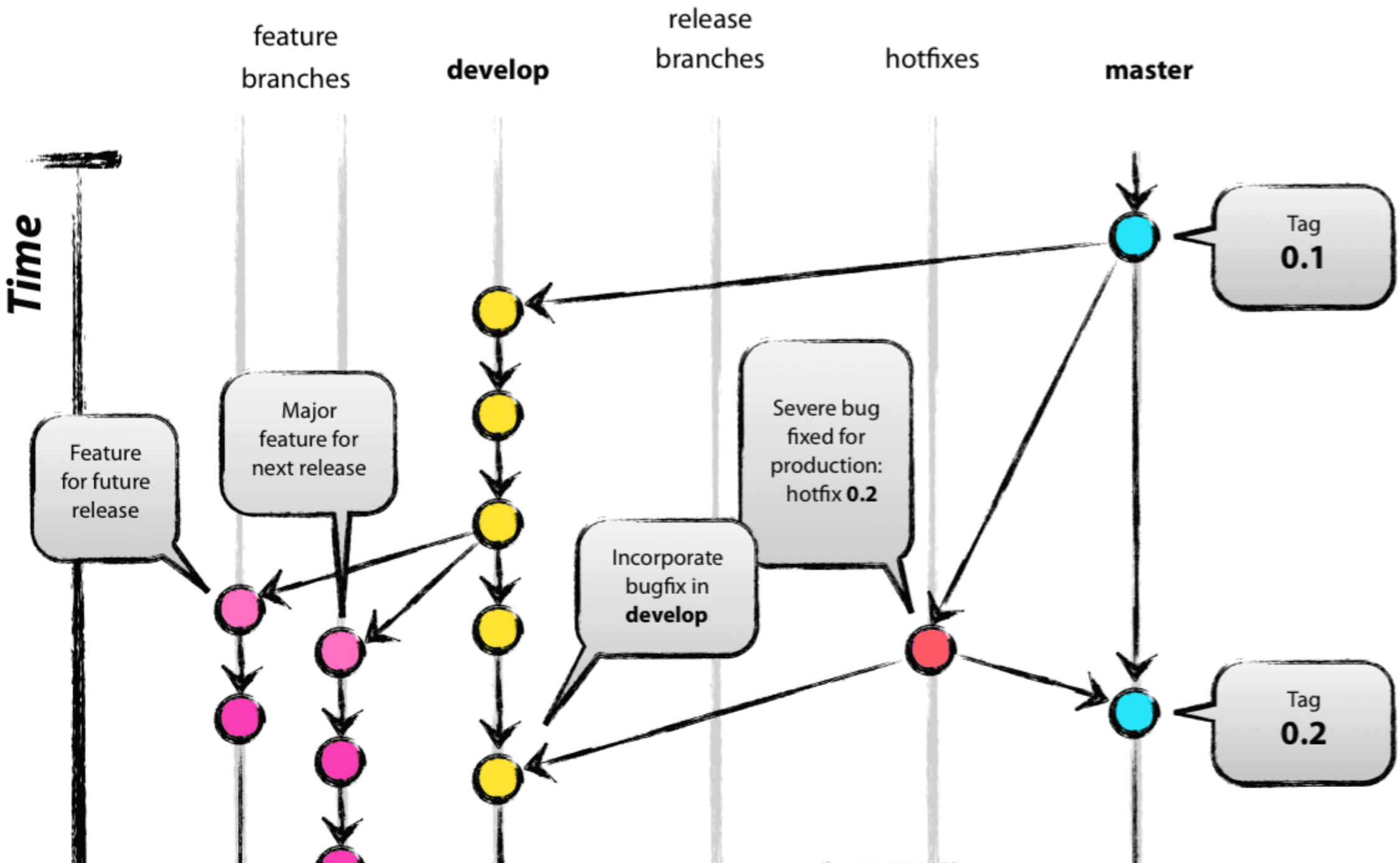
Important consideration when merging

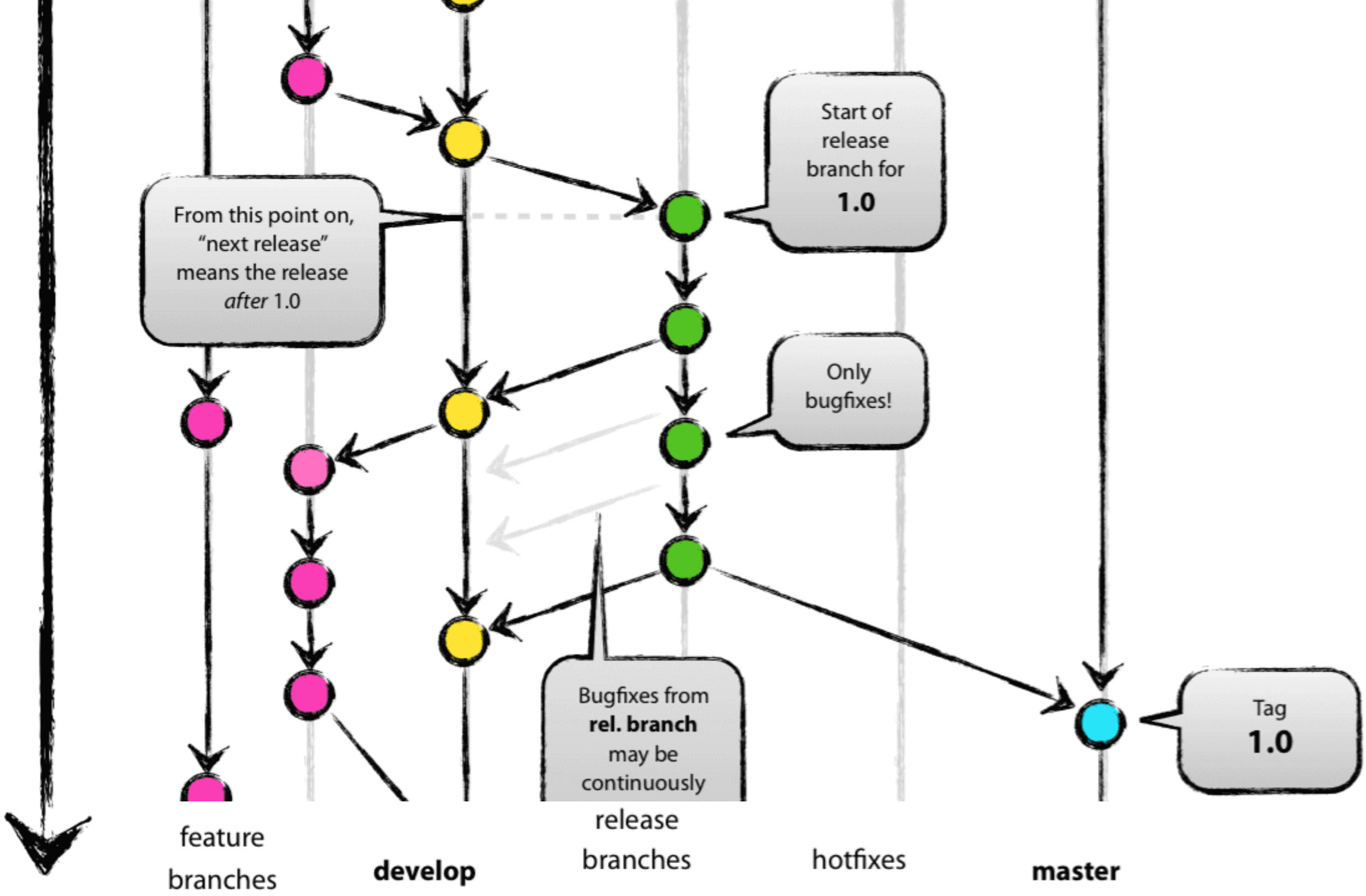




Hotfixes


develop
hotfixes
master
Tag
1.2
Tag
1.2.1
Severe bug
fixed for
production:
hotfix 1.2.1


feature branches
develop
release branches
hotfixes
master
Time
Tag 0.1
Feature for future release
Major feature for next release
Severe bug fixed for production: hotfix 0.2
Incorporate bugfix in develop
Tag 0.2

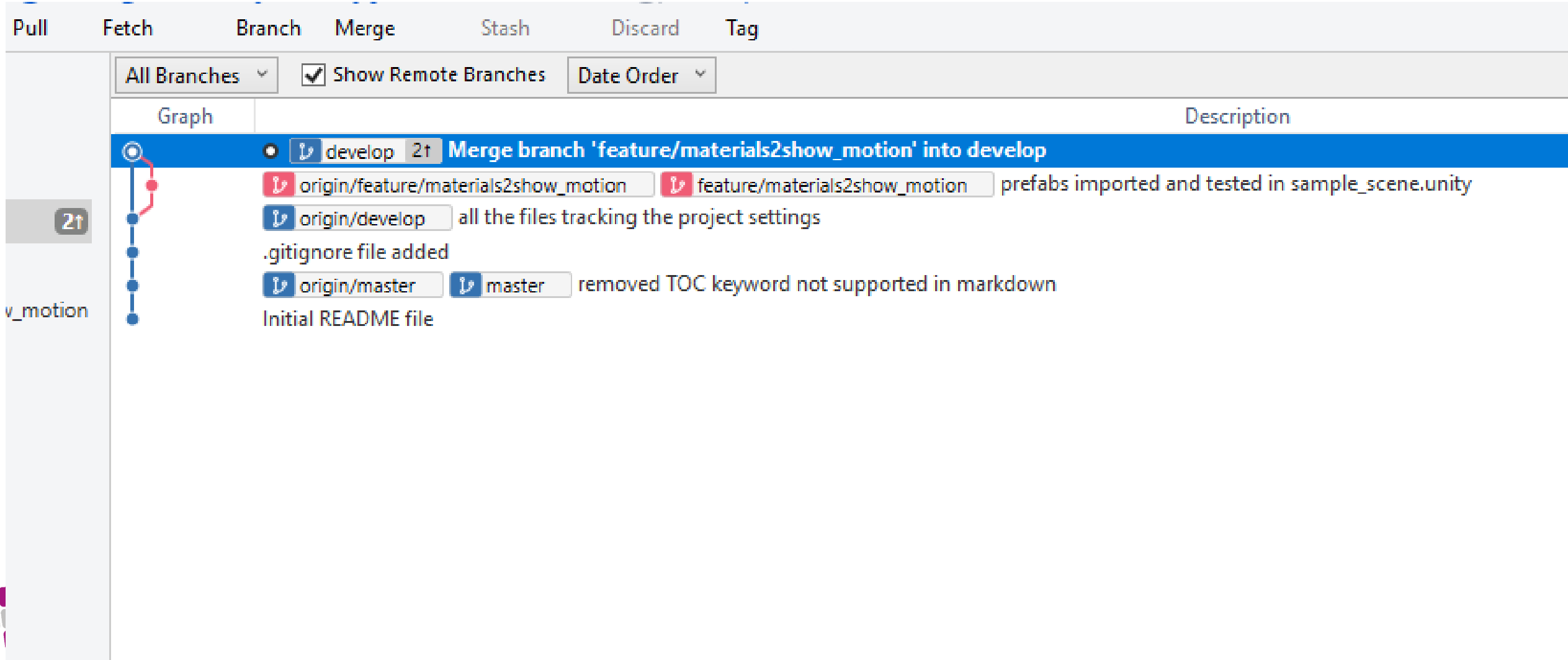

Start of release branch for 1.0
From this point on, "next release" means the release *after* 1.0
Only bugfixes!
Bugfixes from **rel. branch** may be continuously
Tag 1.0
feature branches
develop
release branches
hotfixes
master

Branching conventions

- All commits in develop compile
- Release in master is done through a Merge Request. Previous to acceptance, the entire project is tested
- Release is tagged
- Readme is clear, easy to read
- Hotfix branch only used if necessary

- Release branch unnecessary (we use Develop)
- We do not demand back-compatibility

How it looks on sourcetree



Naming conventions

In C#

- Classes, Functions and Namespaces use PascalCase
  MyFunctionName

- Variables and input fields use camelCase
  myVariableName

- Class fields, private ones, start with underscore _myVariableName

Example naming conventions:
https://www.dofactory.com/reference/csharp-coding-standards

C# Difference between a field and a property

```
using UnityEngine;//only for debug purposes:

public class MyClass  {
    public int MyField1;

    int _myField2;
    public int MyProperty2 {
        get { return _myField2; }
        set {
            if (value > 0)
                _myField2 = value;
            else
                Debug.LogError("MyField2 cannot be negative!");
            }
    }
}
```

What are the differences between a field and a property?

When should we use each?

C# Difference between a field and a property

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class MyComponent : MonoBehaviour {
    public int MyField1;

    [SerializeField]
    int _myField2;
    public int MyProperty2{ set { _myField2 = value; } get { return _myField2; } }


// Use this for initialization
void Start () {}

// Update is called once per frame
void Update () {}
}
```

What are the differences between a field and a property?

When should we use each?

C# Difference between class and struct

A class is a reference type. When an object of the class is created, the variable to which the object is assigned **holds only a reference to that memory**. When the object reference is assigned to a new variable, the new variable refers to the original object. Changes made through one variable are reflected in the other variable because they both refer to the same data.

A struct is a value type. When a struct is created, the variable to which the struct is assigned **holds the struct's actual data**. When the struct is assigned to a new variable, it is copied. The new variable and the original variable therefore contain two separate copies of the same data. Changes made to one copy do not affect the other copy.

C# Other useful keywords

`static`

`const`

`namespace`

`using`

`internal` (very useful)

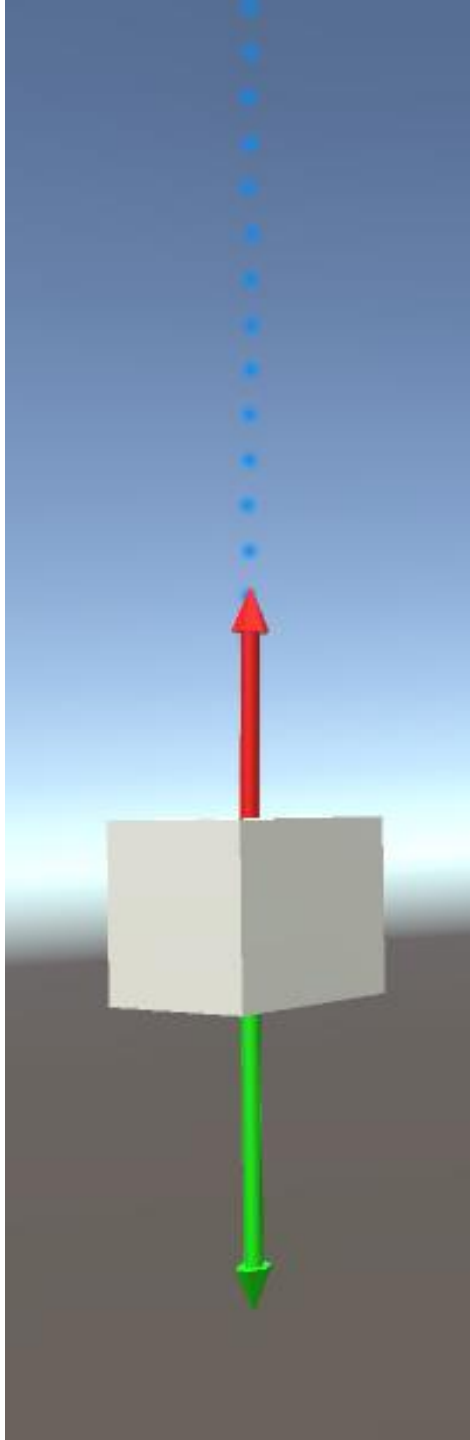
`partial` (avoid)

Unity3D Scene Game Object Conventions

- Respect the hierarchy
- If needed (for example, a manager), use underscore to highlight an element

Project Scene Game Object Conventions

- Acceleration and Forces are shown in Red
- Linear and angular velocity in Green
- Movement is shown in Blue



General Project Repository Conventions

- Commits in gitlab
- More info: https://docs.gitlab.com/ee/user/project/wiki/

Specific Unity3D Repository conventions

We want to use an external git manager

We want code repositories to be clean, and easy to follow

0. We want a repository properly configured

https://docs.unity3d.com/Manual/ExternalVersionControlSystemSupport.html

-make sure commits are in the right email, with name and surname

1. We want clean readme.md, easy to follow project

For markdown edition, you can use, for example:

https://pandao.github.io/editor.md/en.html

Note: keywords like [TOC] do not work in

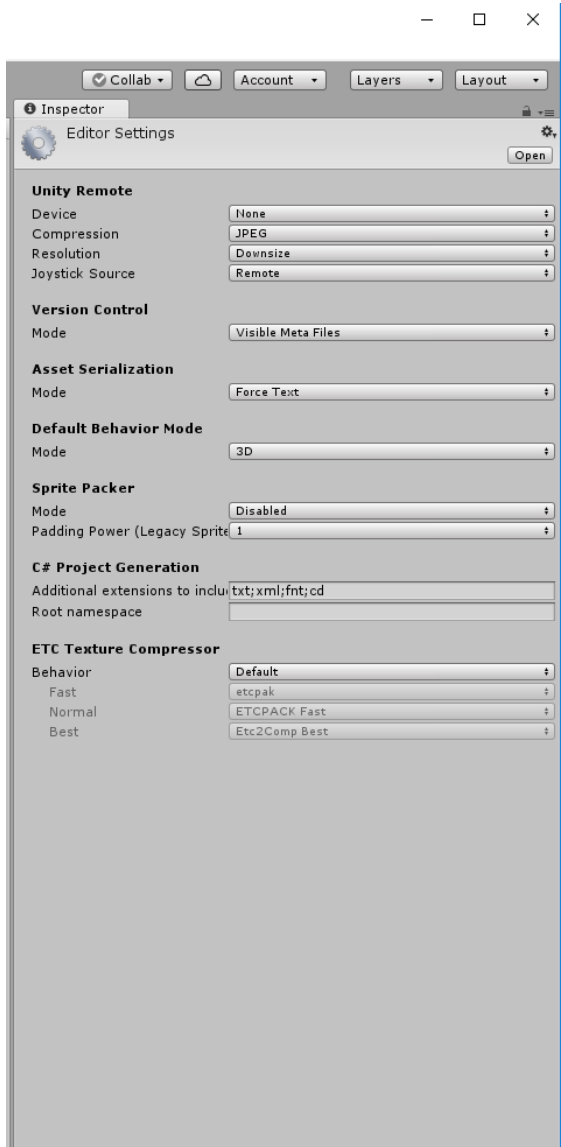
2. We want a .gitignore file that works properly

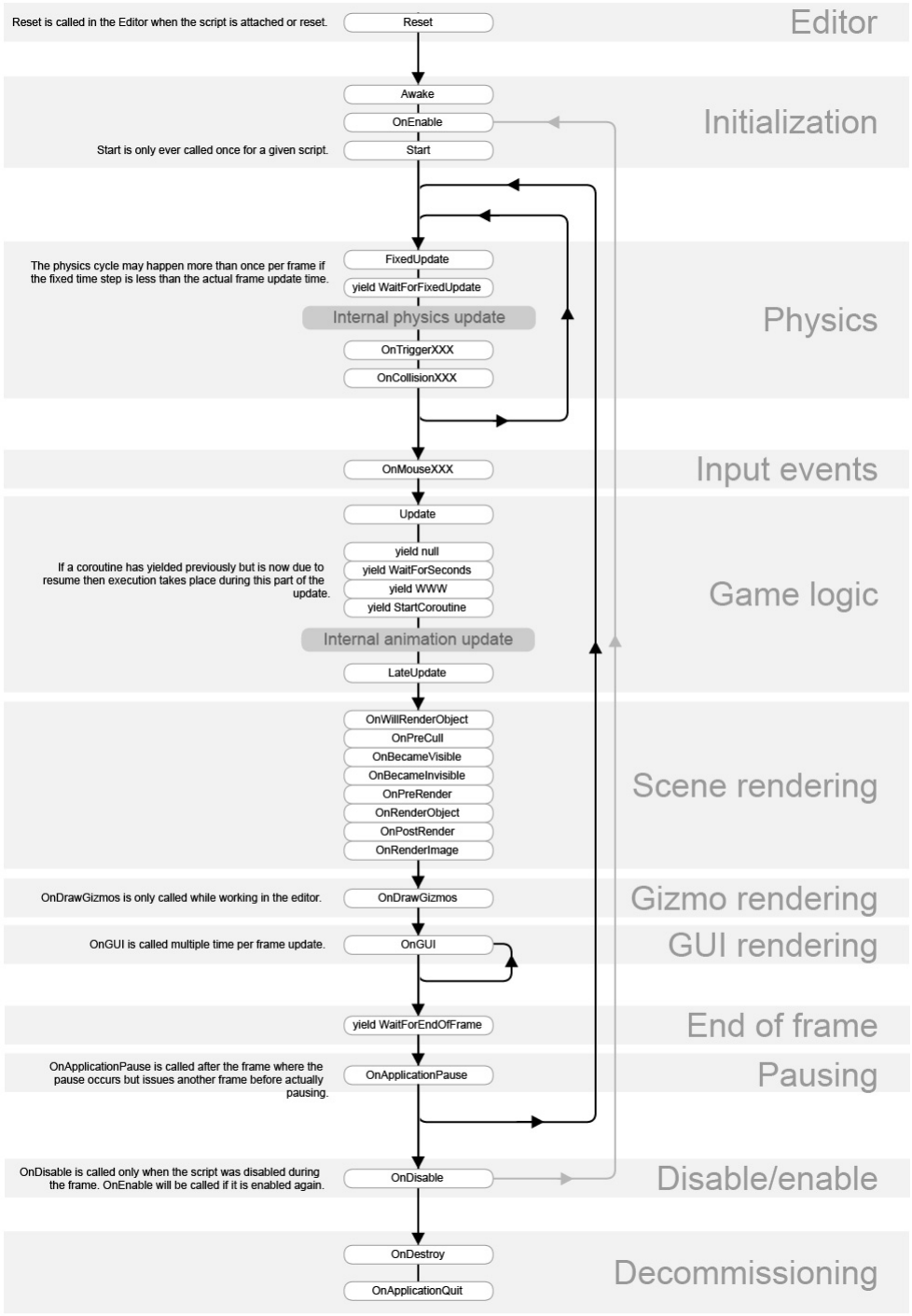
https://docs.unity3d.com/2017.2/Documentation/Manual/ExternalVersionControlSystemSupport.html

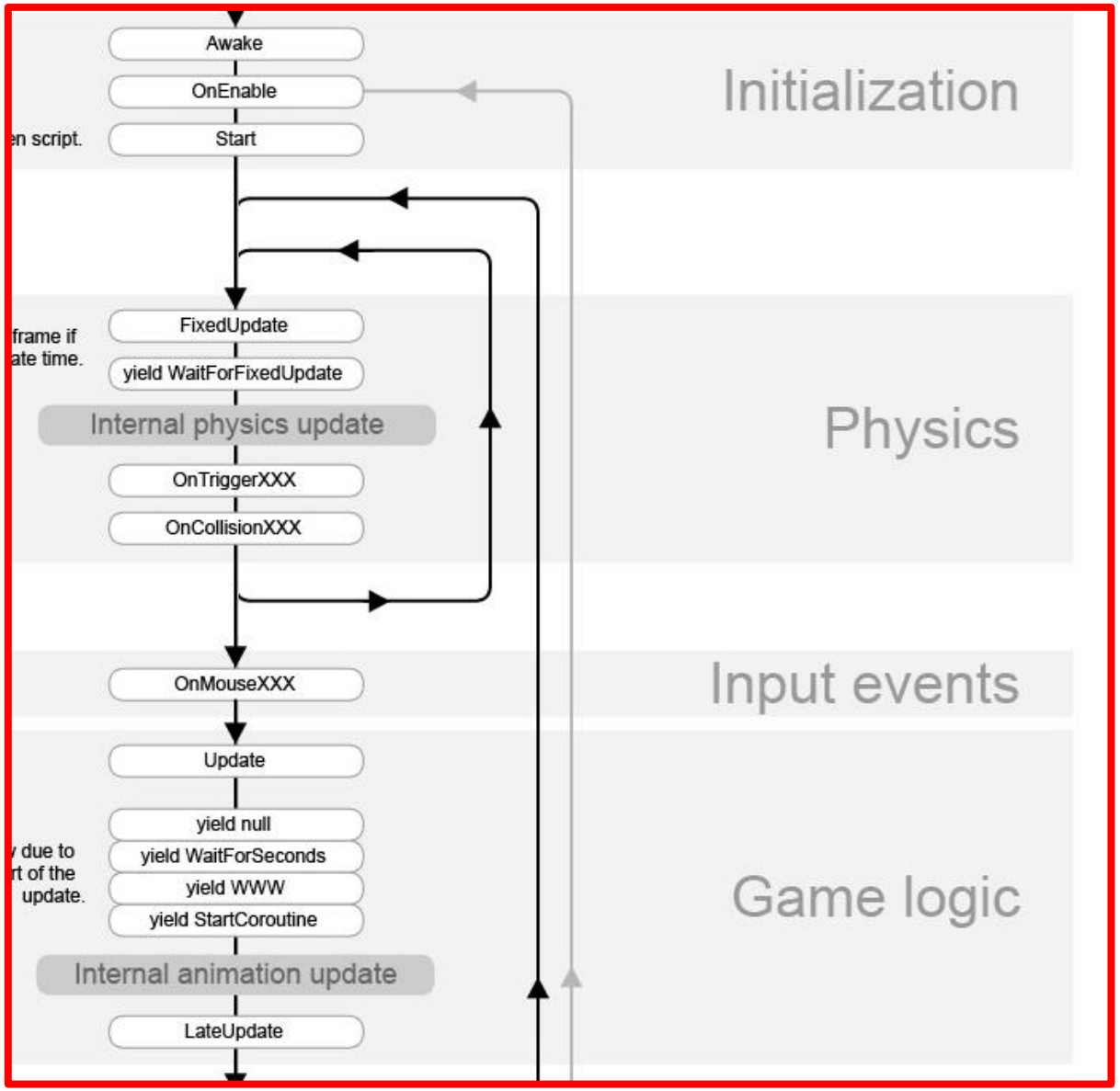
.meta files need to be commited

Assets, UnityPackageManager and ProjectSettings directory are versioned.

Always check a release works directly when downloading the repository







https://docs.unity3d.com/Manual/ExecutionOrder.html

http://www.lucedigitale.com/blog/unity3d-game-engine-script-lifecycle-flowchart/