

Control your GitHub releases with GitVersion and GitReleaseManager

Gary Ewan Park
Email: gep13@gep13.co.uk
Twitter: @gep13
Web: <http://www.gep13.co.uk>

Agenda

- What is Semantic Versioning?

- What is Semantic Versioning?
- What branching strategy should I use?

- What is Semantic Versioning?
- What branching strategy should I use?
- What is GitVersion?

- What is Semantic Versioning?
- What branching strategy should I use?
- What is GitVersion?
- Why do I need GitVersion?

- What is Semantic Versioning?
- What branching strategy should I use?
- What is GitVersion?
- Why do I need GitVersion?
- What is GitReleaseManager?

- What is Semantic Versioning?
- What branching strategy should I use?
- What is GitVersion?
- Why do I need GitVersion?
- What is GitReleaseManager?
- Putting it all together...

What is Semantic Versioning?

A definition...

"...simple set of rules and requirements that dictate how version numbers are assigned and incremented. These rules are based on, but not necessarily limited to, pre-existing widespread common practices in use in both closed and open-source software."

The rules

Given a version number MAJOR.MINOR.PATCH, increment the:

- MAJOR version when you make incompatible API changes
- MINOR version when you add functionality in a backwards-compatible manner
- PATCH version when you make backwards-compatible bug fixes

Additional labels for pre-release and build metadata are available as extensions to the MAJOR.MINOR.PATCH format.

Examples

- 0.1.0

Examples

- 0.1.0
- 1.0.0

Examples

- 0.1.0
- 1.0.0
- 0.3.13

Examples

- 0.1.0
- 1.0.0
- 0.3.13
- 0.2.0-unstable3

Examples

- 0.1.0
- 1.0.0
- 0.3.13
- 0.2.0-unstable3
- 0.2.0-unstable.3+Branch.develop.Sha.e6eb071cd30974b80d7e237b85e7729a1d791e1e

What branching strategy should I use?

GitHub Flow

Create a branch

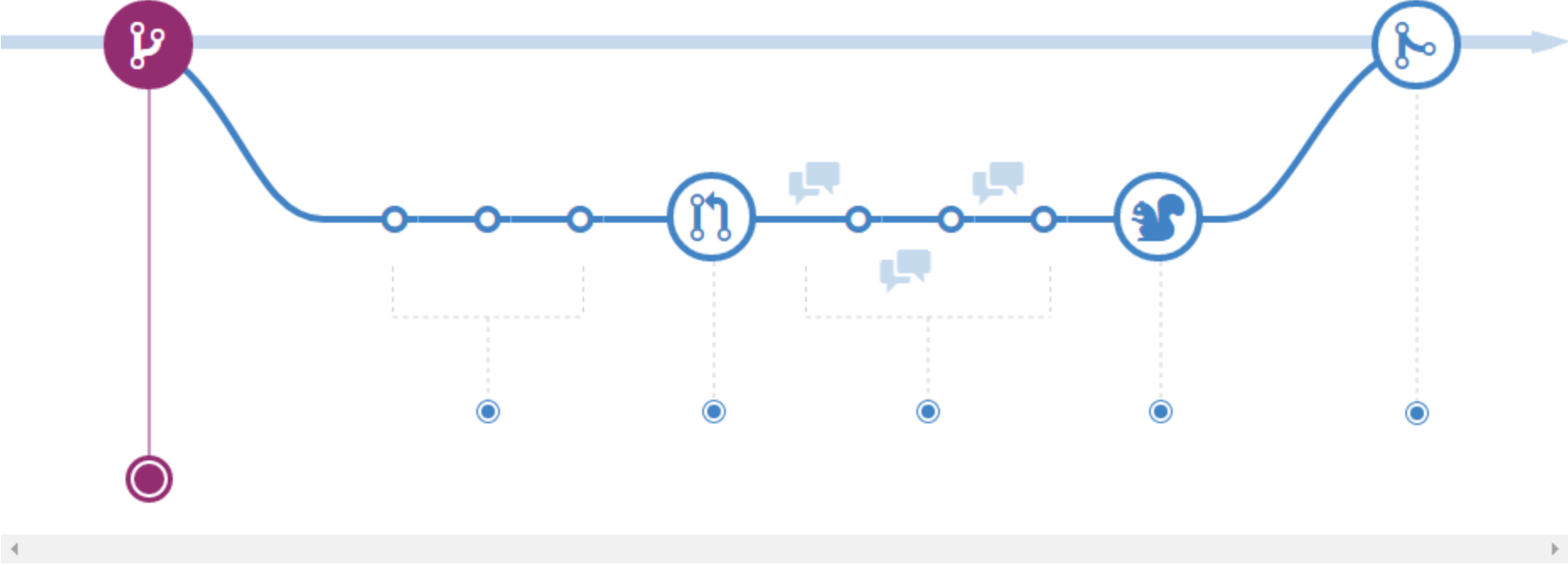


Image borrowed from the [GitHub Flow Tutorial](#)

Add commits

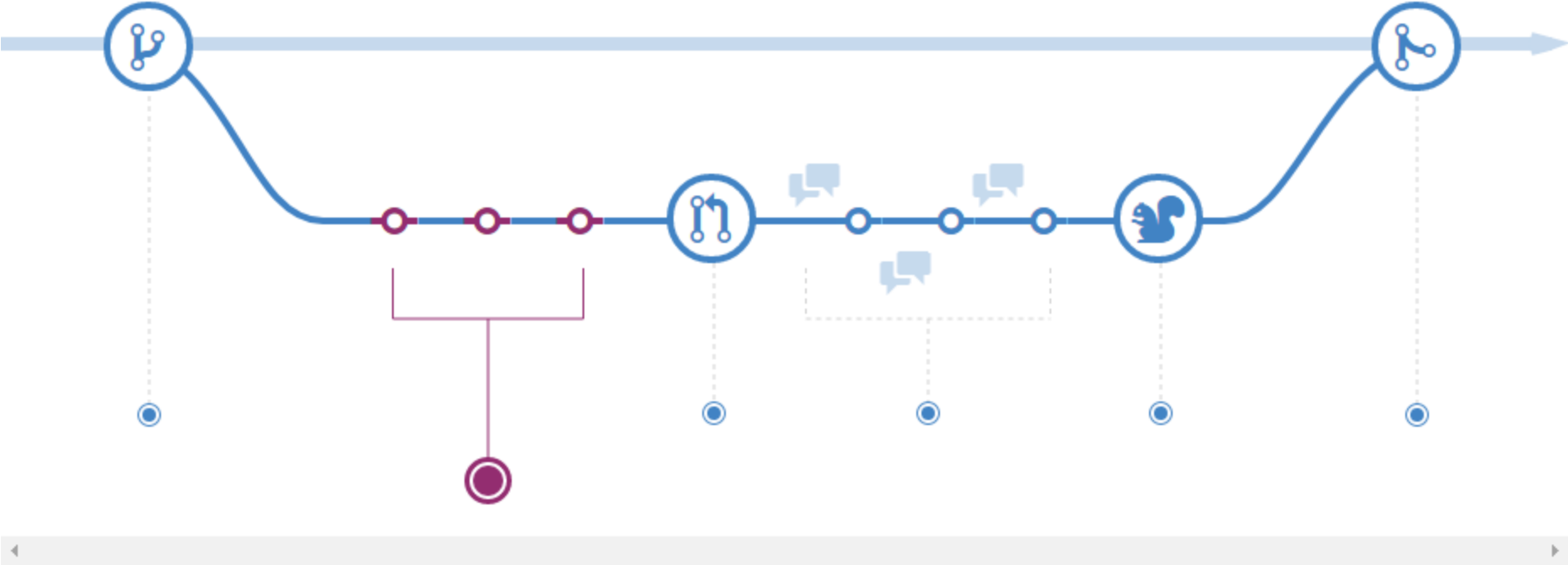


Image borrowed from the [GitHub Flow Tutorial](#)

Open a Pull Request

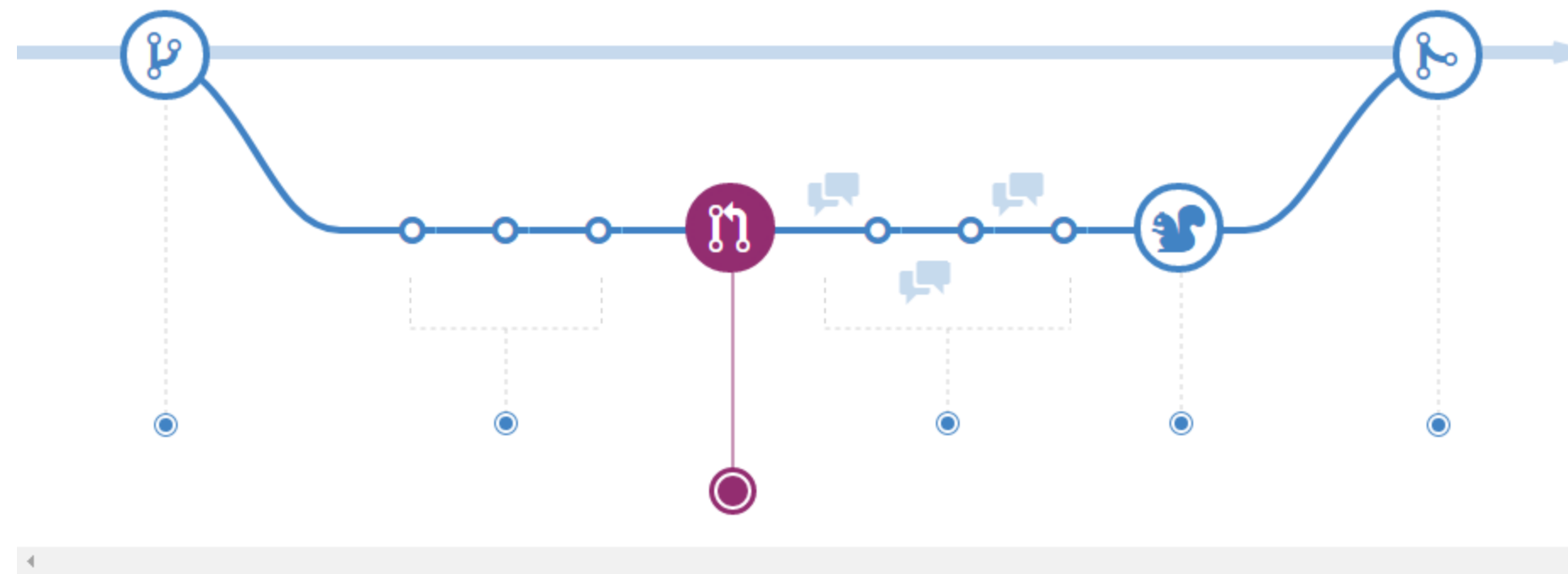


Image borrowed from the [GitHub Flow Tutorial](#)

Discuss and review your code

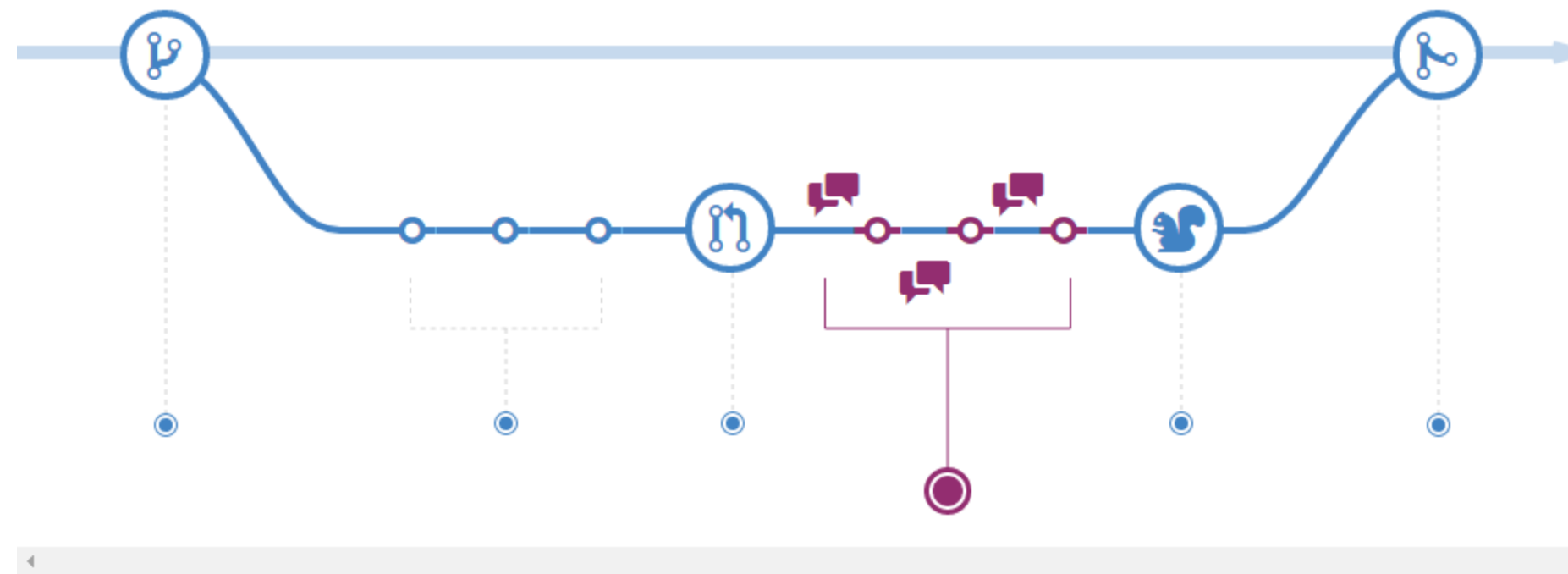


Image borrowed from the [GitHub Flow Tutorial](#)

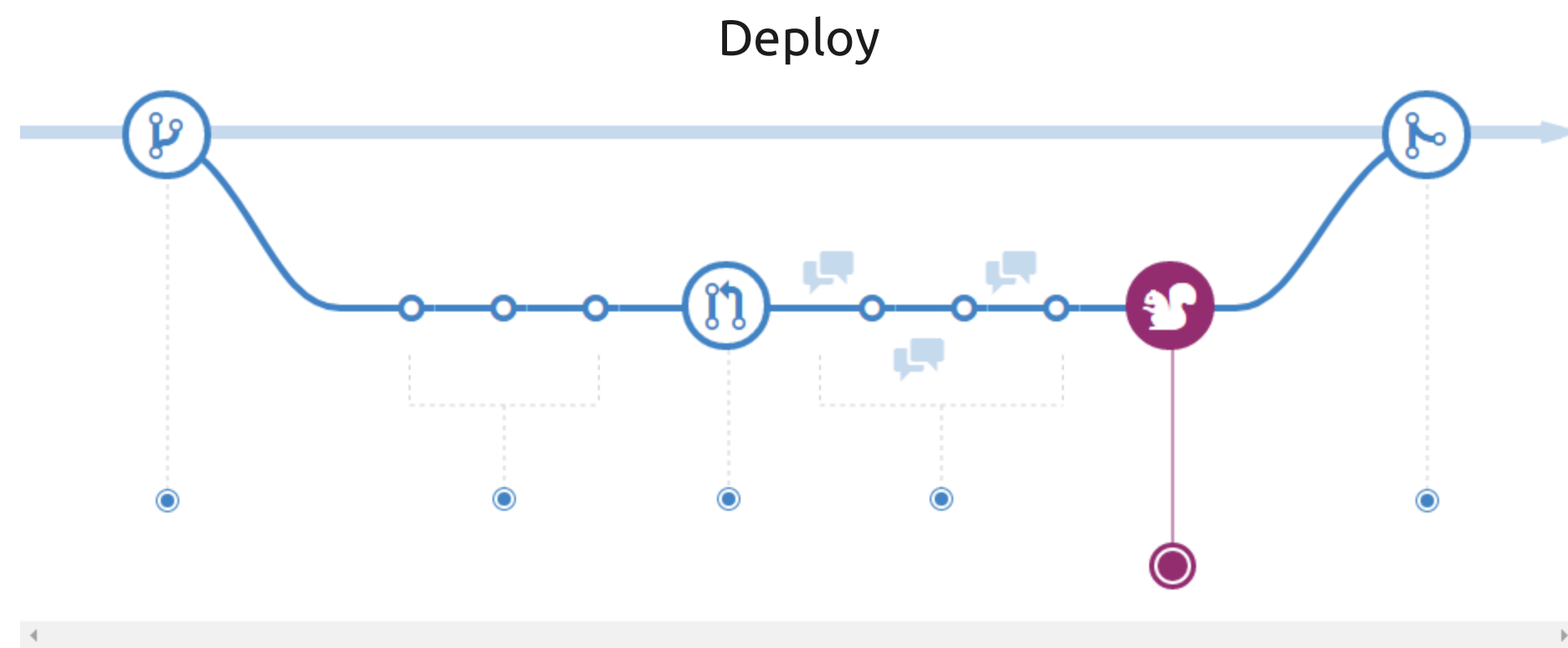


Image borrowed from the [GitHub Flow Tutorial](#)

Merge

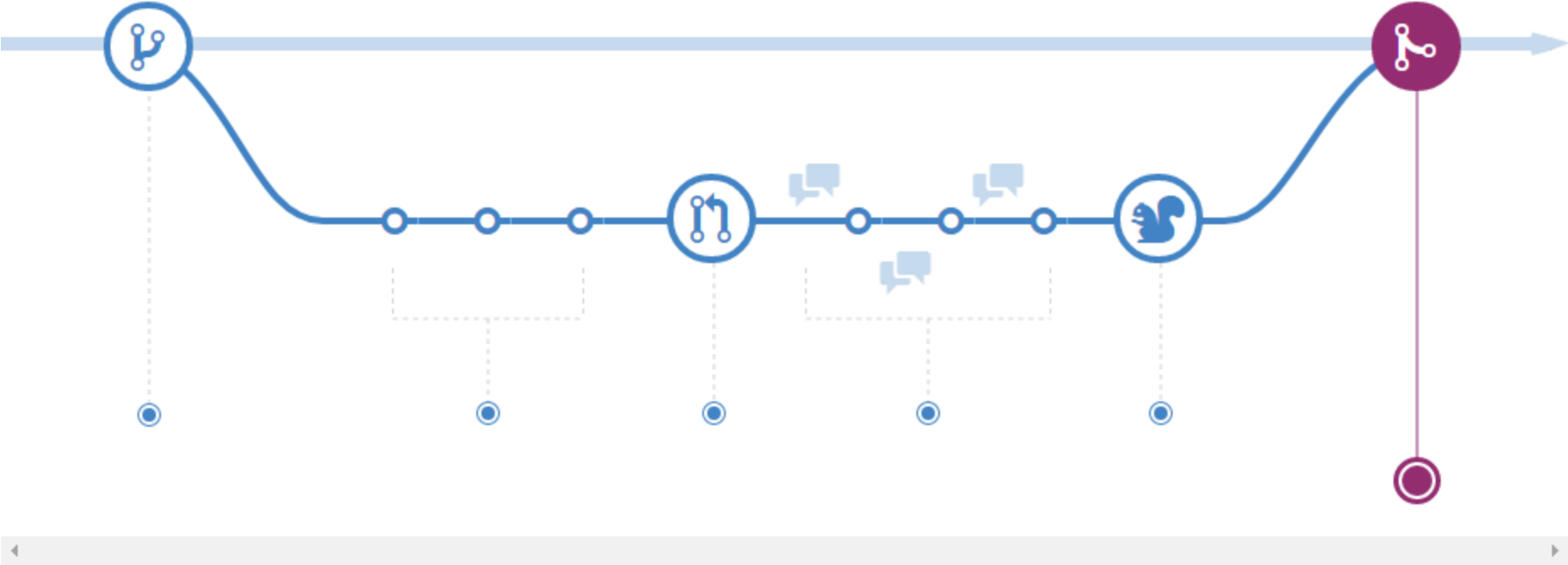


Image borrowed from the [GitHub Flow Tutorial](#)

GitFlow

Two Main Branches

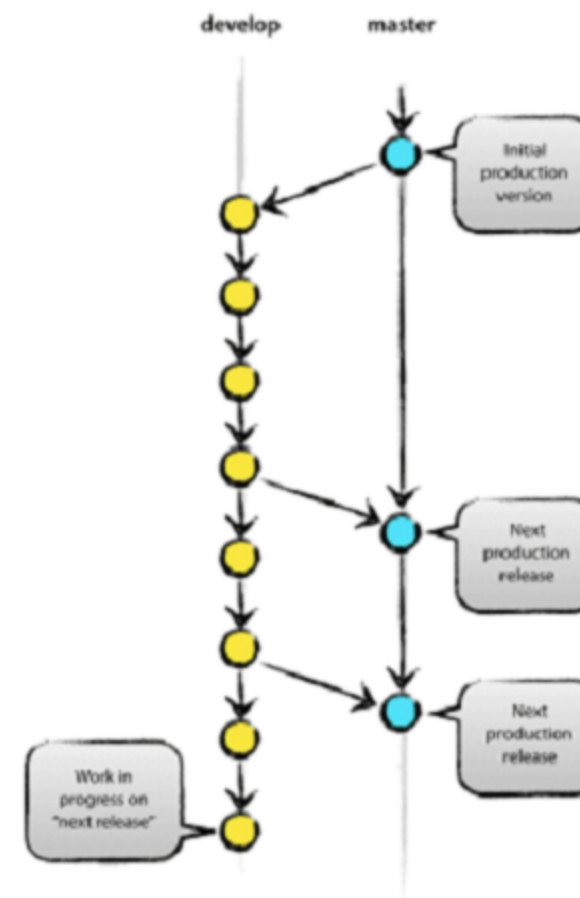


Image borrowed from the [A successful Git branching model](#) blog post

Feature Branches

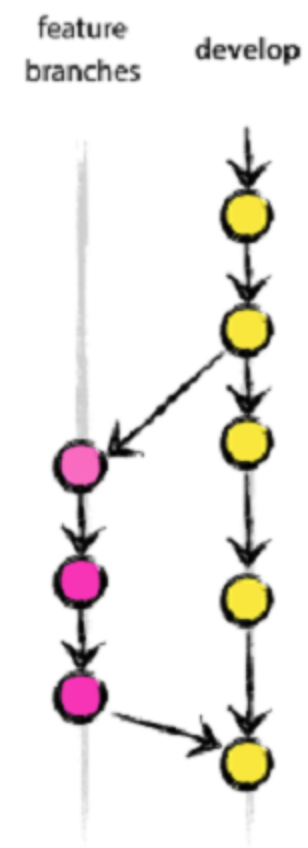


Image borrowed from the [A successful Git branching model](#) blog post

Hotfix Branches

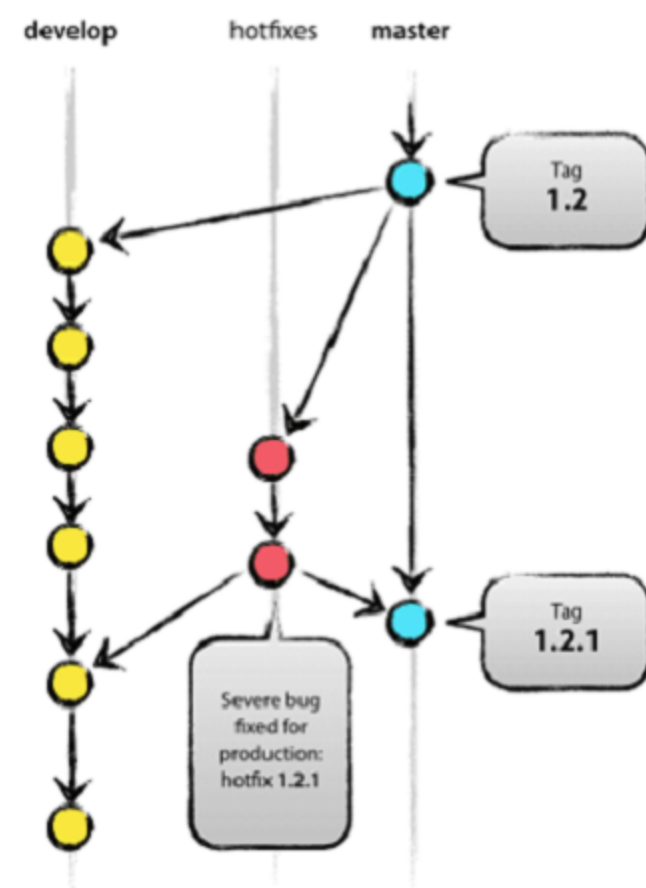


Image borrowed from the [A successful Git branching model](#) blog post

Release Branches

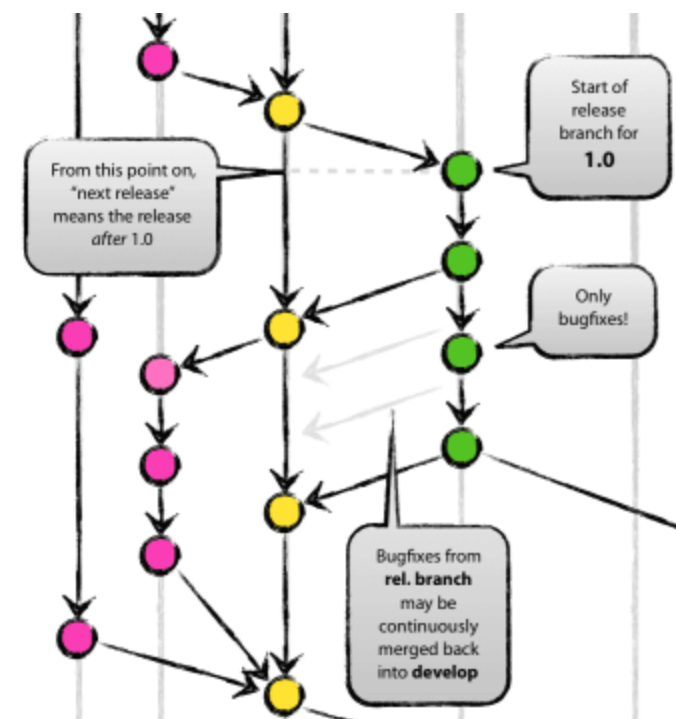


Image borrowed from the [A successful Git branching model](#) blog post

Overview

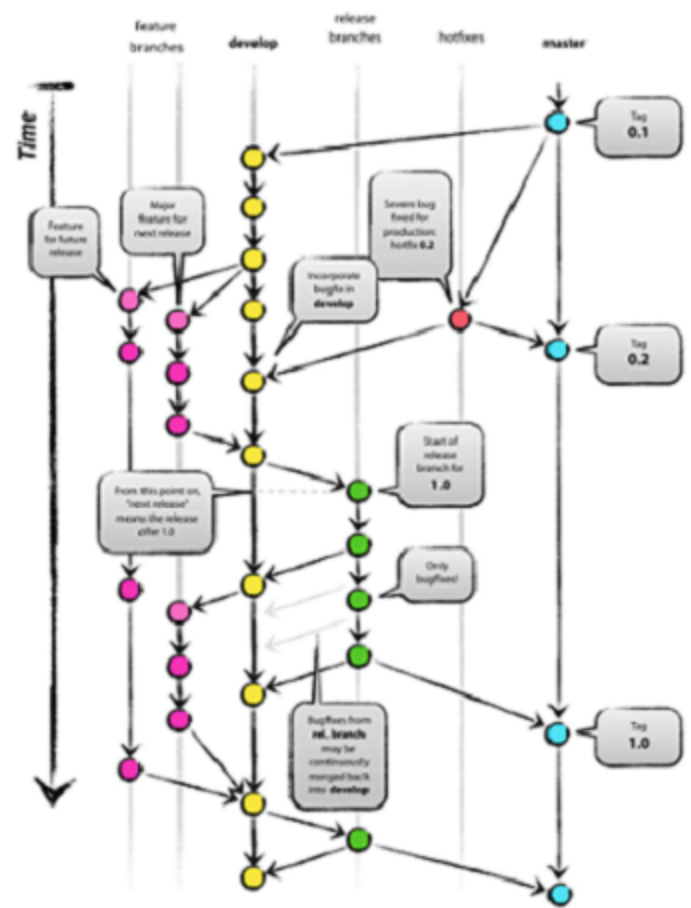


Image borrowed from the [A successful Git branching model](#) blog post
















What is GitVersion?

A defintion...

"...is a tool to help you achieve Semantic Versioning on your project."

Ok, but really, what is it?

Why do I need GitVersion?

Name	Date modified	Type	Size
 .git	22/06/2015 11:13	File folder	
 BuildScripts	22/06/2015 11:13	File folder	
 Documentation	22/06/2015 11:13	File folder	
 Prototypes	22/06/2015 11:13	File folder	
 Source	17/07/2015 09:45	File folder	
 SQL Scripts	22/06/2015 11:13	File folder	
 Testing	22/06/2015 11:13	File folder	
 Tools	22/06/2015 11:13	File folder	
 .editorconfig	22/06/2015 11:13	EDITORCONFIG File	1 KB
 .gitattributes	22/06/2015 11:13	Text Document	1 KB
 .gitignore	22/06/2015 11:13	Text Document	3 KB
 build.bat	22/06/2015 11:13	Windows Batch File	1 KB
 generateHelp.bat	22/06/2015 11:13	Windows Batch File	1 KB
 README.md	22/06/2015 11:13	MD File	1 KB
 version.txt	02/09/2015 20:02	Text Document	0 KB

```
[assembly: AssemblyTrademark("")]
[assembly: AssemblyCulture("")]
[assembly: NeutralResourcesLanguage("en-GB")]

[assembly: AssemblyVersion("0.2.0.0")]
[assembly: AssemblyFileVersion("0.2.0.0")]
[assembly: AssemblyInformationalVersion("0.2.0.0")]
```

Administration > <Root project> > CIDemo > Create Build Configuration

General Settings

Name: ^{*}

Build configuration ID: ^{*} [?]

This ID is used in URLs, REST API, HTTP requests to the server, and configuration settings in the TeamCity Data Directory.

Description:

Build number format: ^{*} [?]

The format may include '%build.counter%' as a placeholder for the build counter value, for example, **1.%build.counter%**.
It may also contain a reference to any other available parameter, for example, **%build.vcs.number.VCSRootName%**.
Note: The maximum length of a build number after all substitutions is **256** characters.

Build counter: ^{*} [?] [Reset counter](#)

Artifact paths: [?]

Newline- or comma-separated paths to build artifacts. Ant-style wildcards like **dir/**/*.*.zip** and target directories like ***.zip => winFiles, unix/distro.tgz => linuxFiles**, where **winFiles** and **linuxFiles** are target directories are supported.

Build options: [?]
☒ enable hanging builds detection
☐ enable status widget [?]
Limit the number of simultaneously running builds (0 — unlimited)

[VCS settings >>](#)

[Cancel](#)



What is GitReleaseManager?

A definition...

"...is a tool that will help create a set of release notes for your application/product. It does this using the collection of issues which are stored on the GitHub Issue Tracker for your application/product."

Ok, but really, what is it?

Putting it all together...

Demo

Questions?

Feel free to get in touch

Email: gep13@gep13.co.uk

Twitter: @gep13

Web: <http://www.gep13.co.uk>

=====

develop

Resources

- GitVersion Documentation
 - <http://gitversion.readthedocs.org/en/latest/>
- .Net Rocks Episode with Jake Ginnivan
 - <https://www.dotnetrocks.com/default.aspx?showNum=1178>
- Git Branching Strategies
 - <https://www.atlassian.com/git/tutorials/comparing-workflows>
- GitFlow
 - <http://nvie.com/posts/a-successful-git-branching-model/>