Version Control
Git
Github

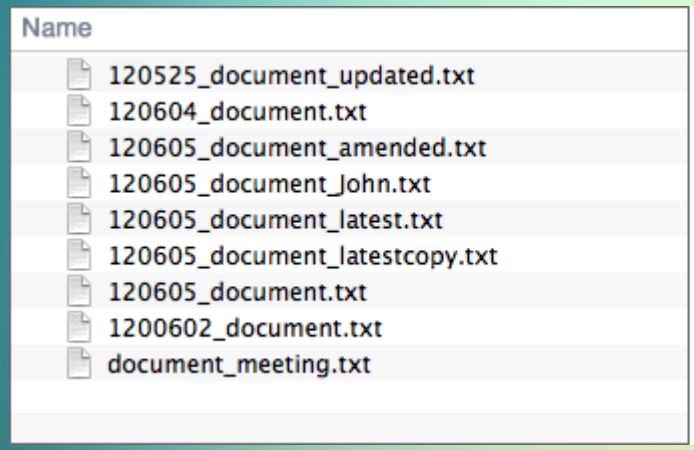
Nick Reynolds

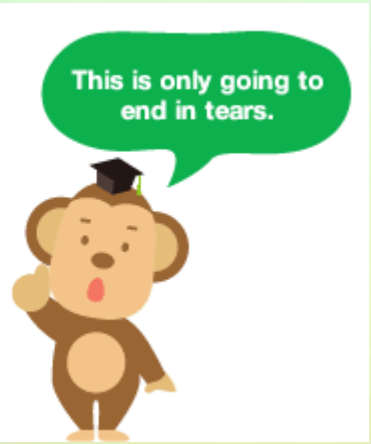
Goal

Introduce version control and git and walk through the mechanics of using it. Hopefully you walk away with a confidence to start using it and learn on your own.

Motivation

Imagine you are writing a paper and you delete a section during edits. The next day you decide you want to keep that section, but you saved over the old version! Maybe you decide to save a unique file for every iteration... (edit1.pdf,edit2.pdf...)





Maybe you add a line of code to your programming project and now it breaks at runtime. If you can't find that new line again you are left to sift through your code...

What is Version Control?

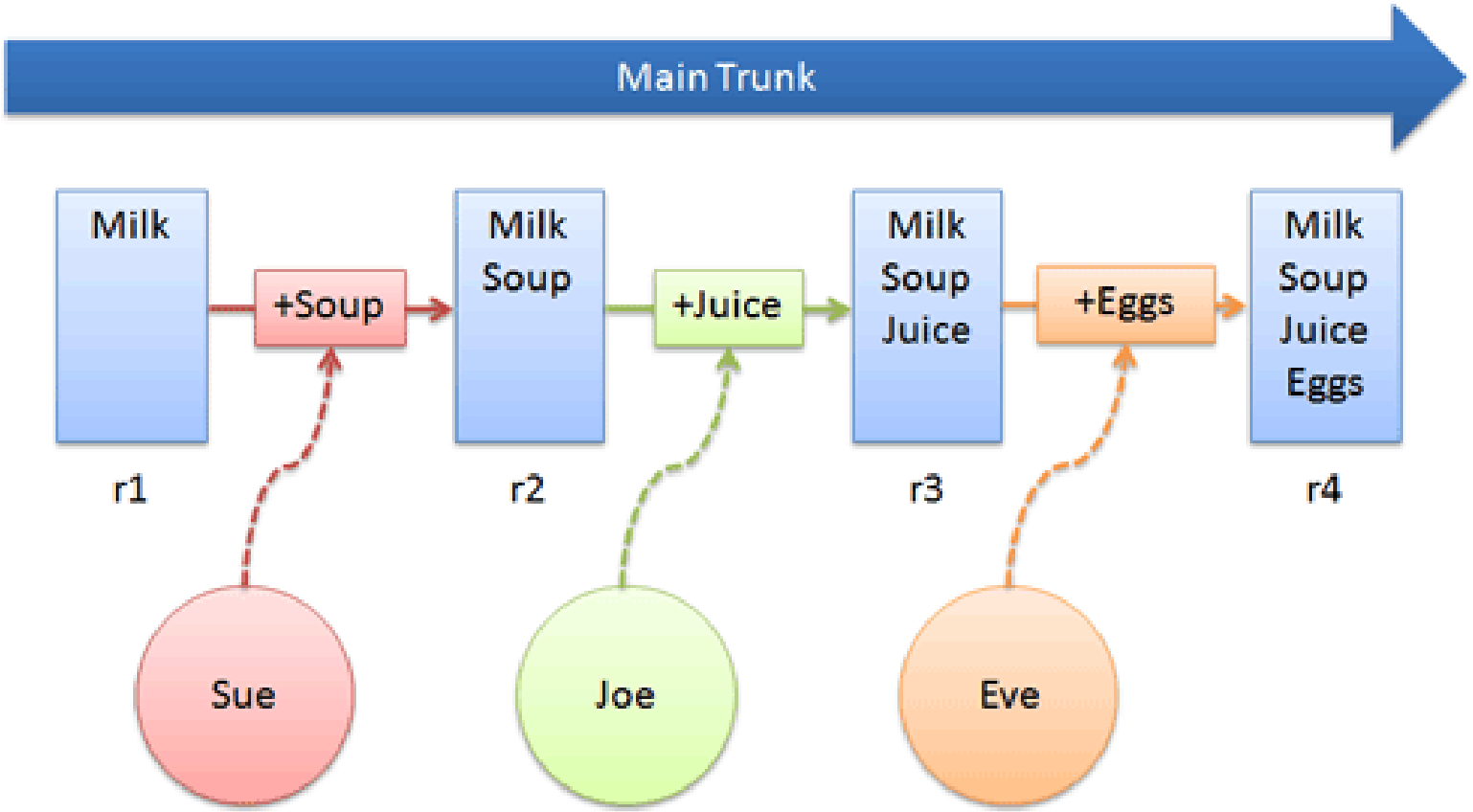
‘[VC] is the management of changes to documents, computer programs, large web sites, and other collections of information.’ -Wiki

[VC] allows you to:

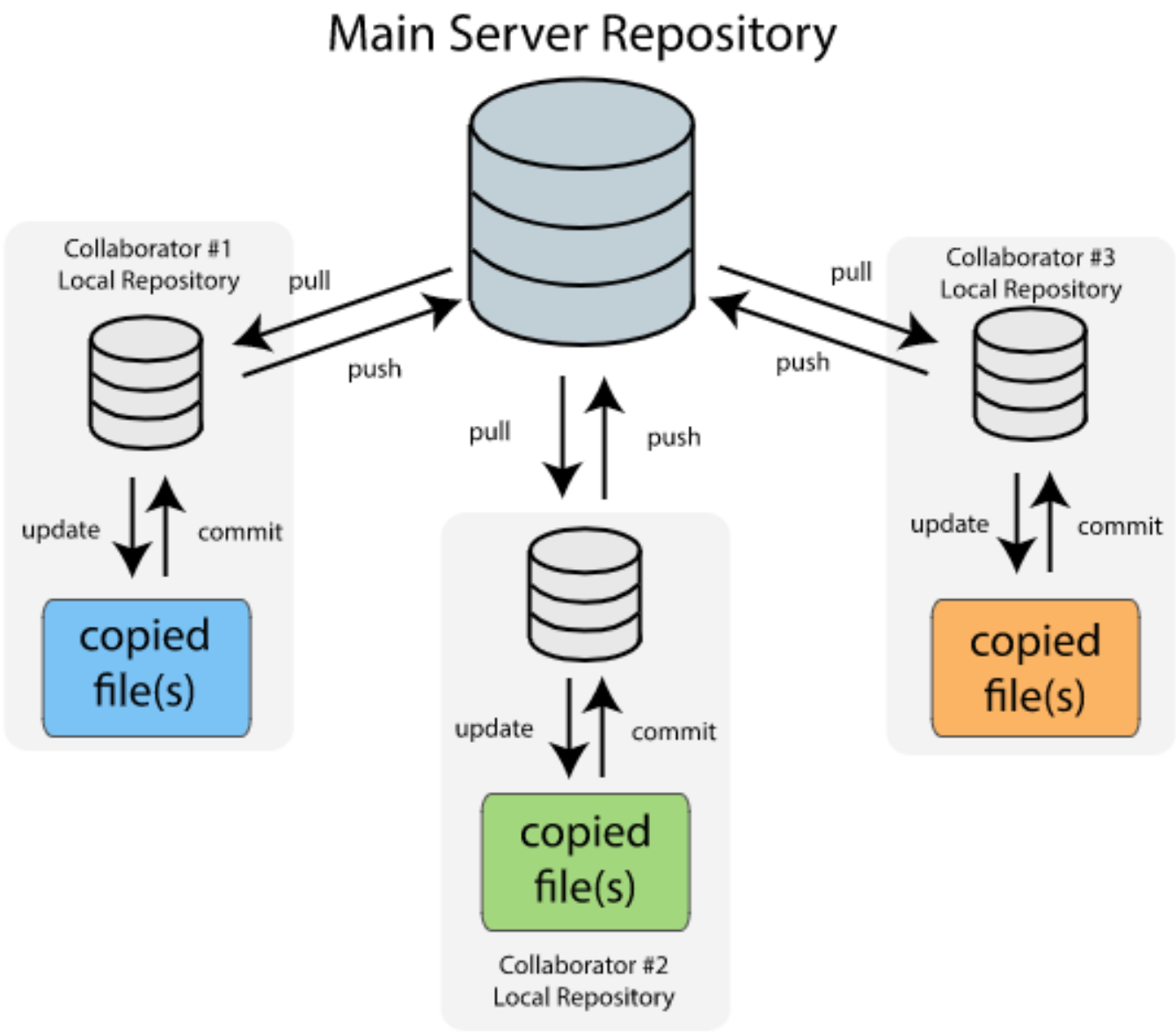
- record iterations without saving them separately
- navigate these iterations and restore them if needed

In short: maintain a SINGLE set of files while tracking what changes have occurred

Centralized VCS



Distributed Version Control



A Distributed Version Control System. Each collaborator has a local copy of the repository, so no Internet connection is required.

Git vs Github vs GitLab

Git is a popular distributed version control system created in 2005 by Linus Torvalds in response to the DVCS used for the development of the Linux kernel becoming a paid service. Git has many nice features in addition to the ones that come from being DVCS (offline work, it is ‘small’, etc.)

GitHub is a web service that hosts Git repositories. It is best known for its website and is in many ways a social platform.

GitLab is similar to GitHub, however GitHub has focused more on a social aspect versus GitLab which is aimed towards enterprise (this is changing)

Starting with Git

Installation:

Chances are you already have it installed!
Otherwise, consult the official website:
<https://git-scm.com/book/en/v2/Getting-Started-Installing-Git>

Configure User

```
$ git config --global user.name "Jane Doe"
$ git config --global user.email jdoe@ou.edu
```

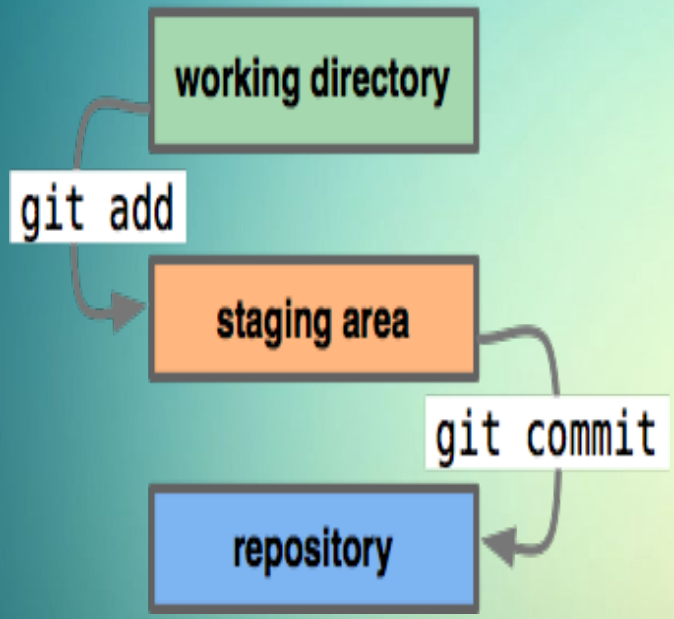
Create repository

```
$ git init
  OR
$ git clone https://www.URL.com/project.git
```

Try cloning this repository
<github.com/nickalaskreynolds/REU.git>

Functional Git

An understanding of how Git operates is needed. The history of changes in a repository come in the form of commits. Changes must be staged before they can be committed.




```
$ git status
```

CHECK CHANGES...

```
$ git add <files-to-be-staged>
```

...STAGE...

```
$ git commit -m "commit msg"
```

... and COMMIT them

Remote Repositories

These are commands related to communicating with remote repositories.

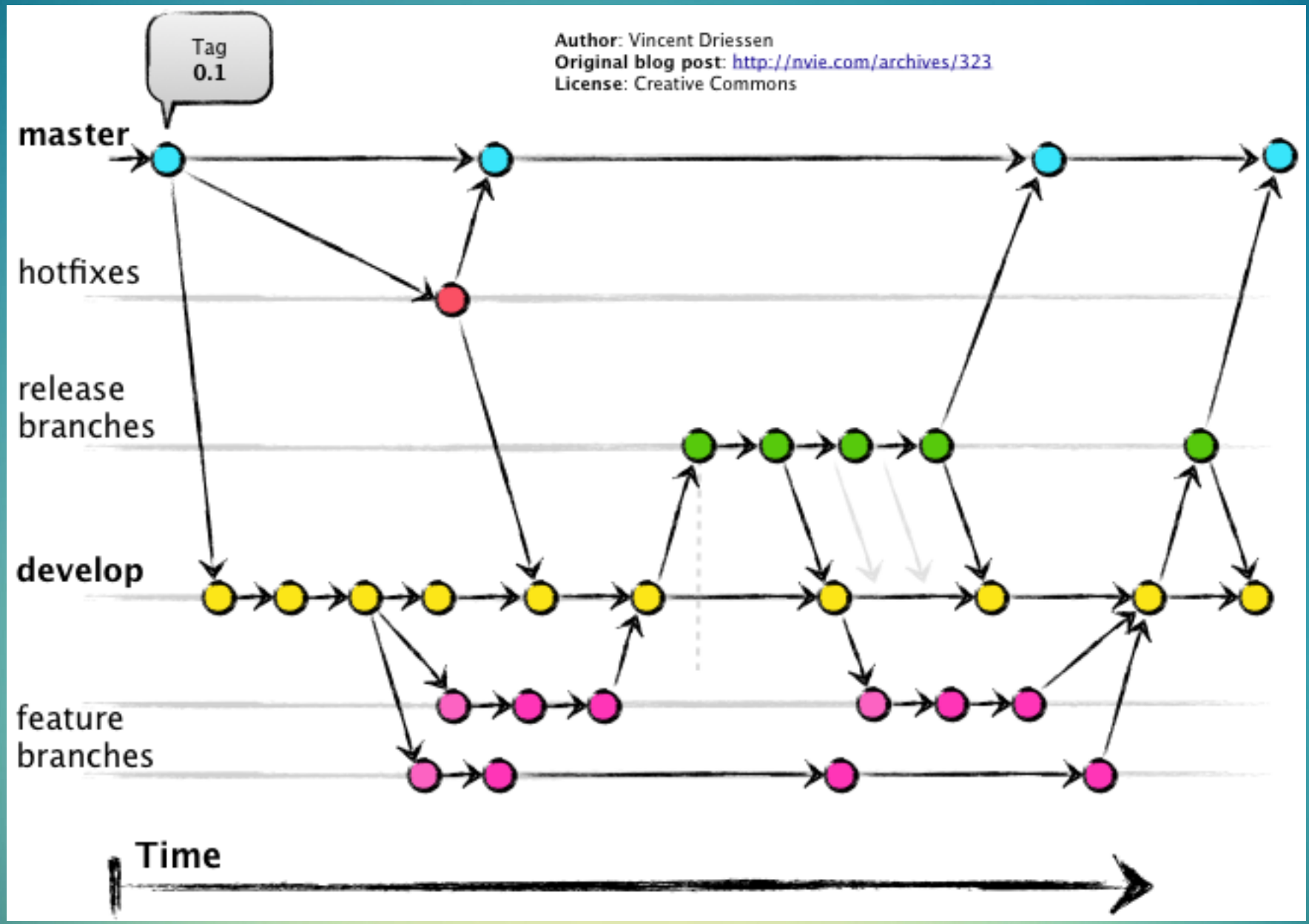
```
$ git remote add <shortname> <url>
```

```
$ git push <remote> <branch>
```

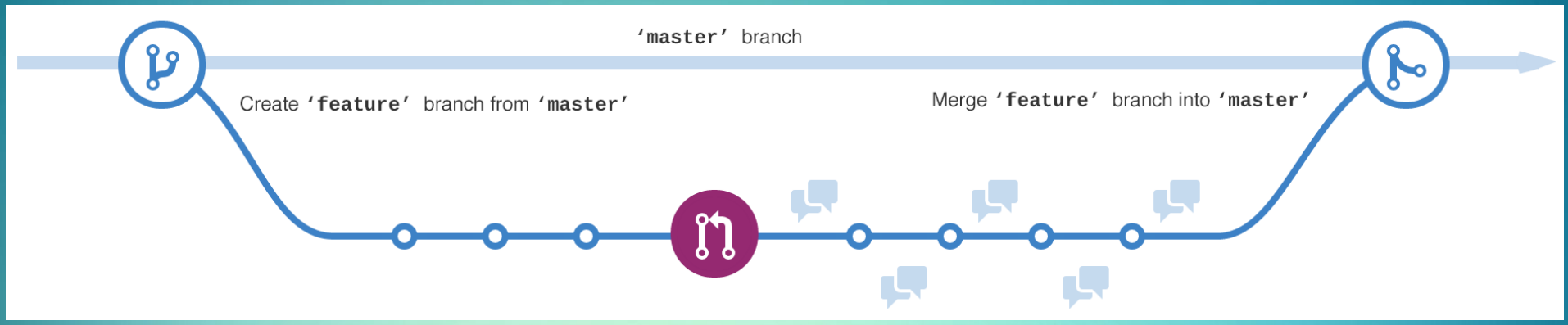
```
$ git pull <remote> <branch>
```

=

```
$ git fetch
$ git merge
```


Tag
0.1
Author: Vincent Driessen
Original blog post: http://nvie.com/archives/323
License: Creative Commons
master
hotfixes
release
branches
develop
feature
branches
Time

Branches



List all branches

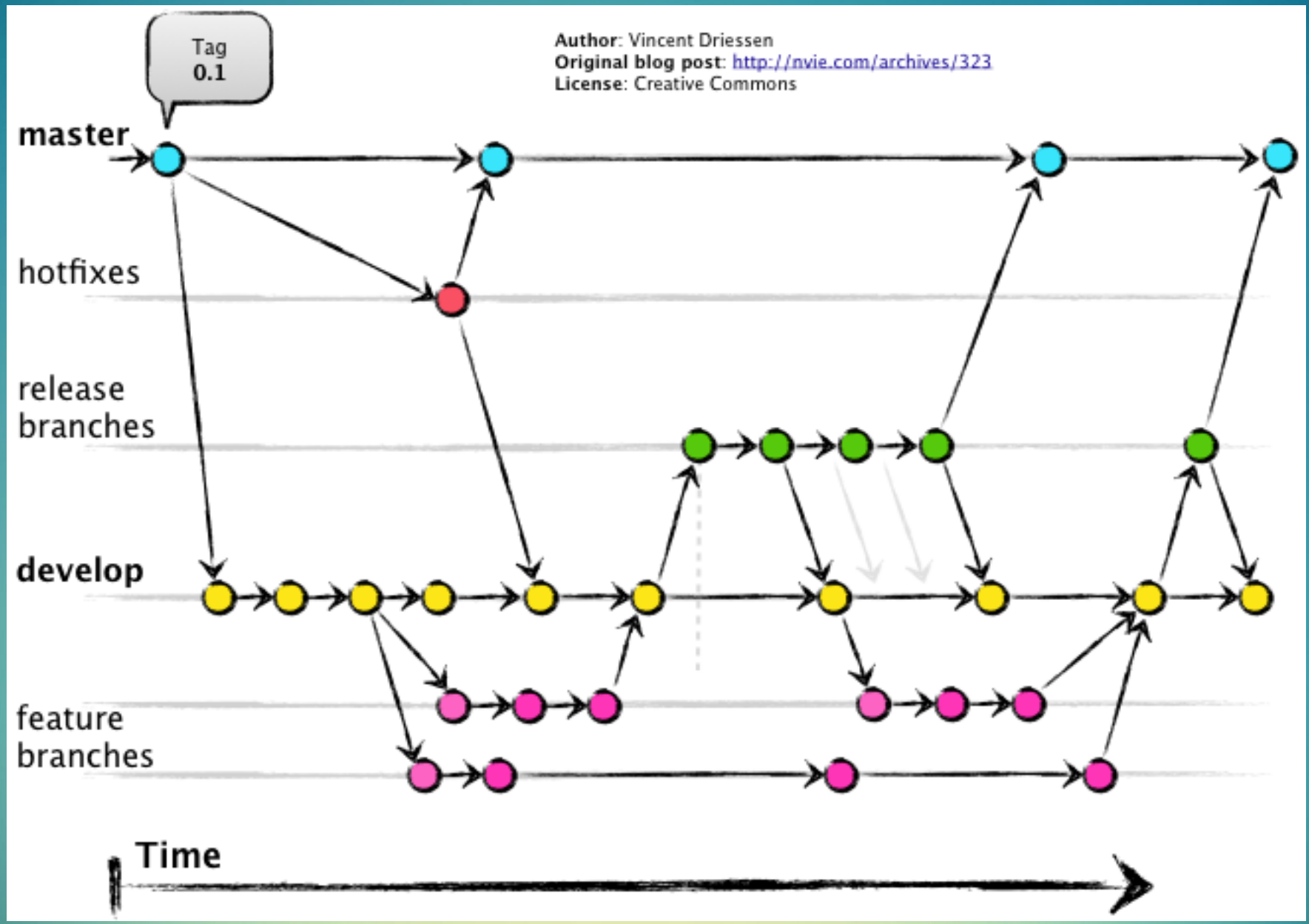
```
$ git branch -av
```

Switch HEAD

```
$ git checkout <branch>
```

Create new Branch from HEAD

```
$ git branch <new-branch>
```


Tag
0.1
Author: Vincent Driessen
Original blog post: http://nvie.com/archives/323
License: Creative Commons
master
hotfixes
release
branches
develop
feature
branches
Time

Other Commands

Differences of files

```
$ git diff
```

Show all commits, starting with newest

```
$ git log
```

Who dun it?

```
$ git blame <file>
```

Revert Changes

```
$ git reset
```

Store changes temporarily

```
$ git stash
```


Explore GitHub