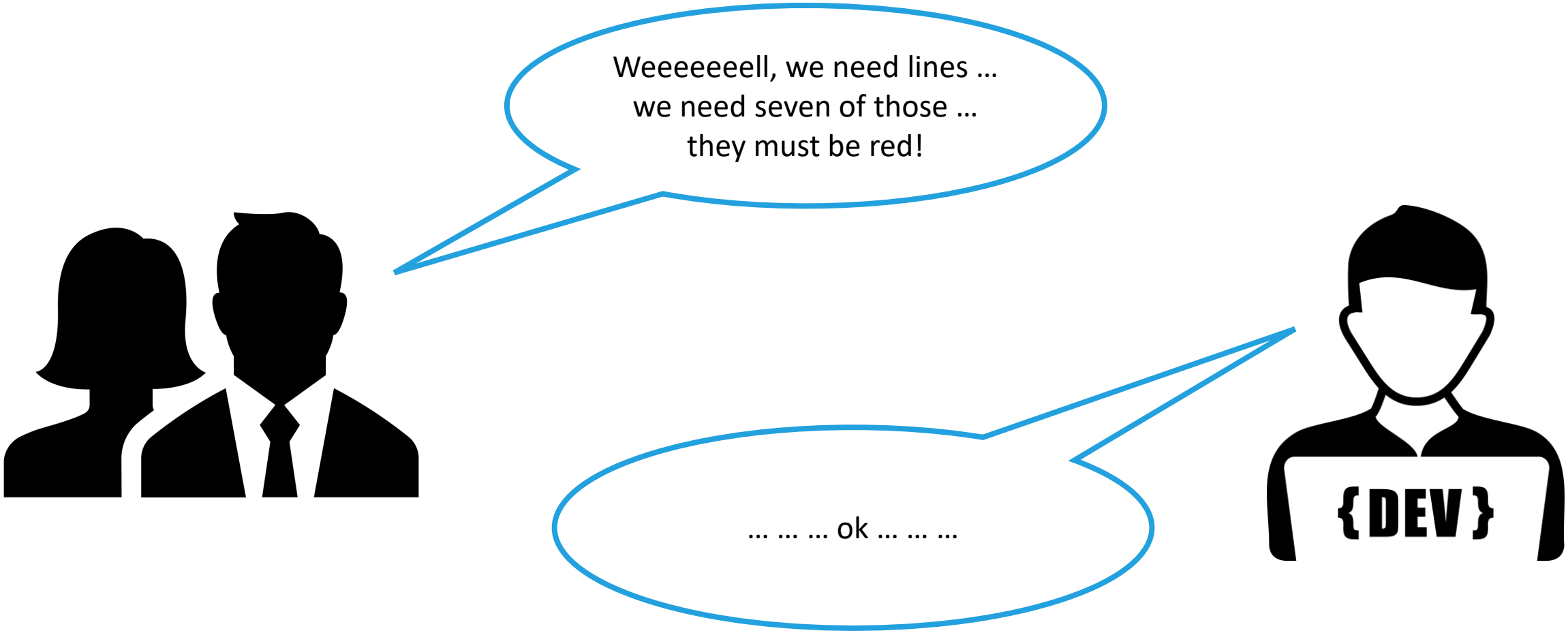
QUANTIA consulting

Code Versioning with Git

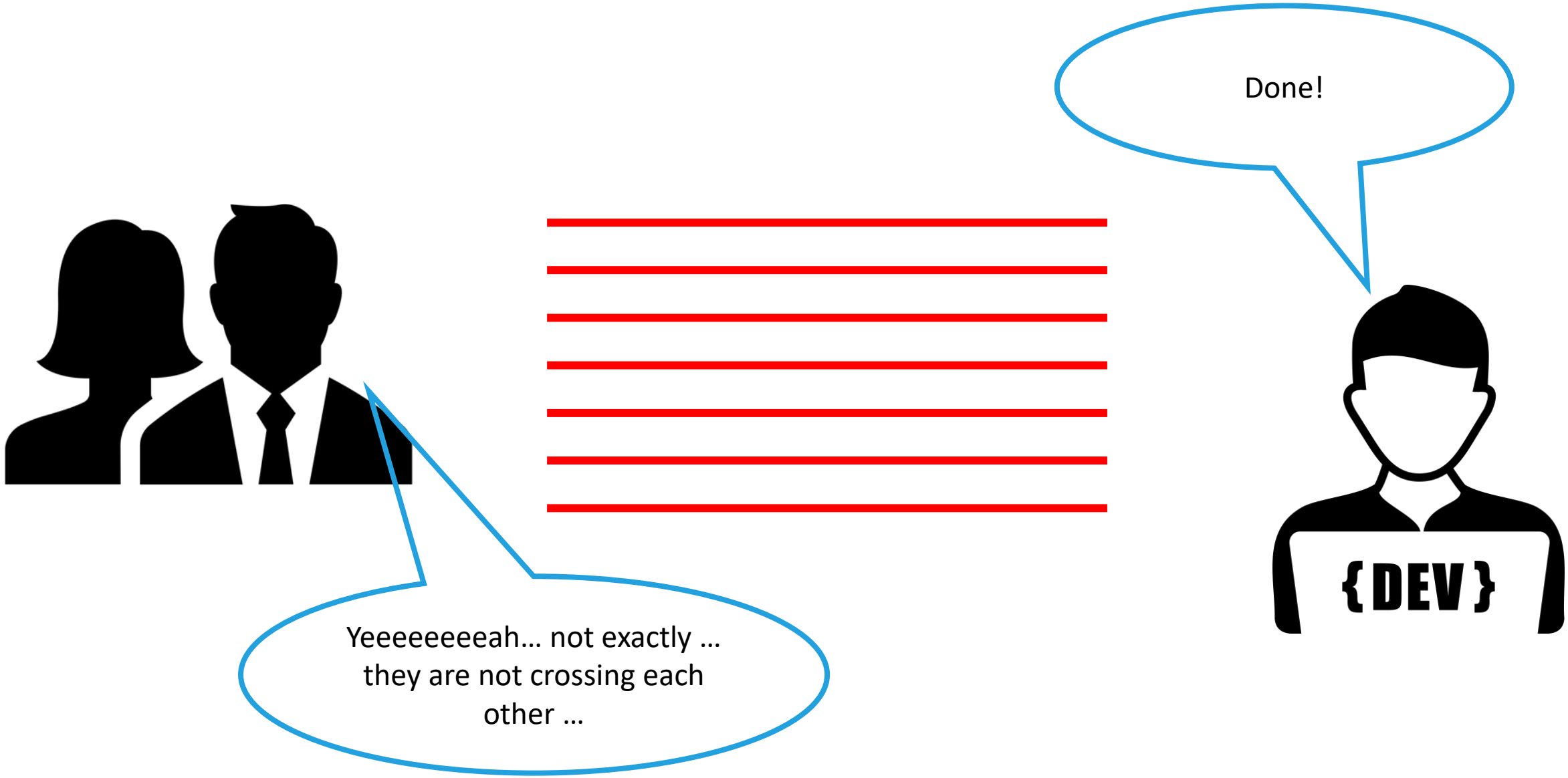
Software requirements elicitation is difficult

Software requirements elicitation is difficult

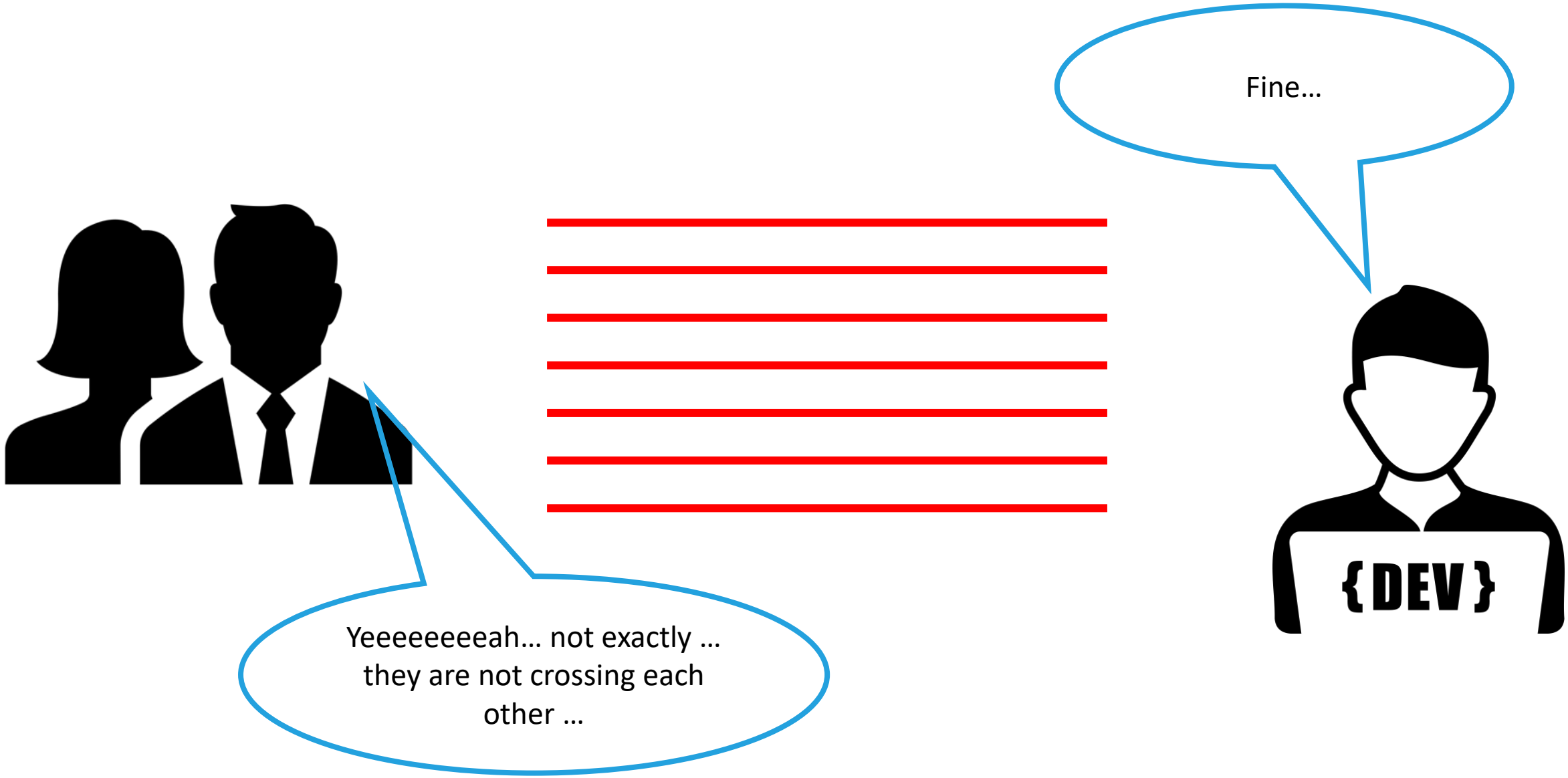


The development goes on …

… but there are some misunderstandings



… but there are some misunderstandings



Making the required changes …

Making the required changes …

How about this?

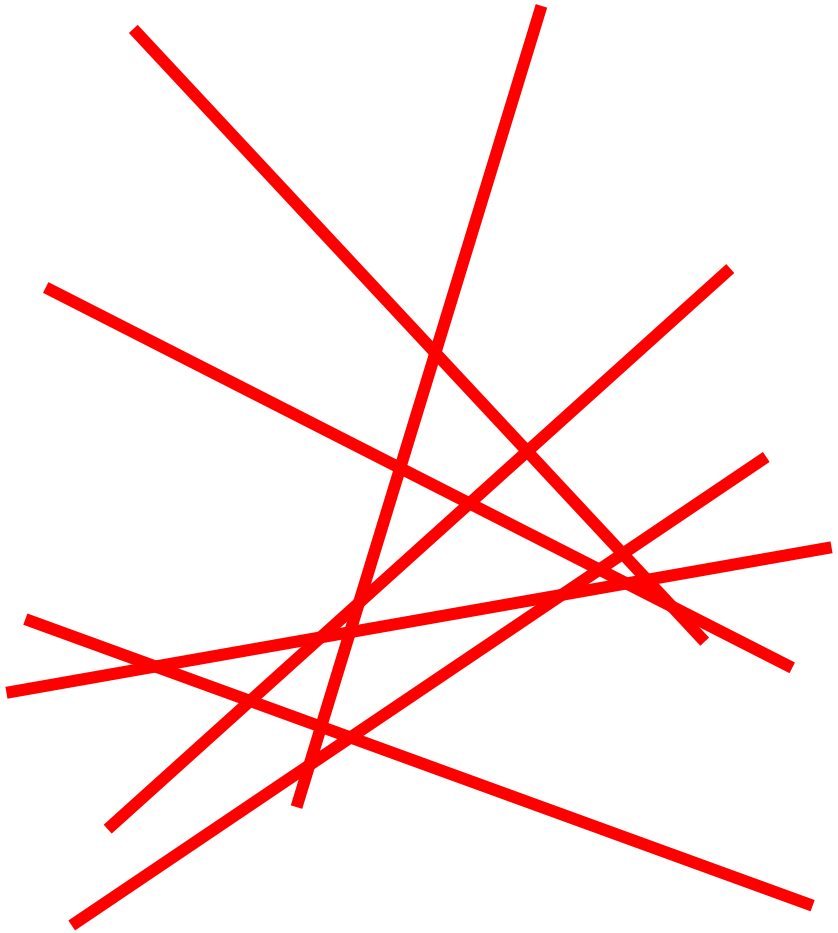
Ah! Too caotic!
What about … a green line and all the other 6 lines perpendicular to it?

Making the required changes …

Ah! Too caotic!
What about … a green line and all the other 6 lines perpendicular to it?

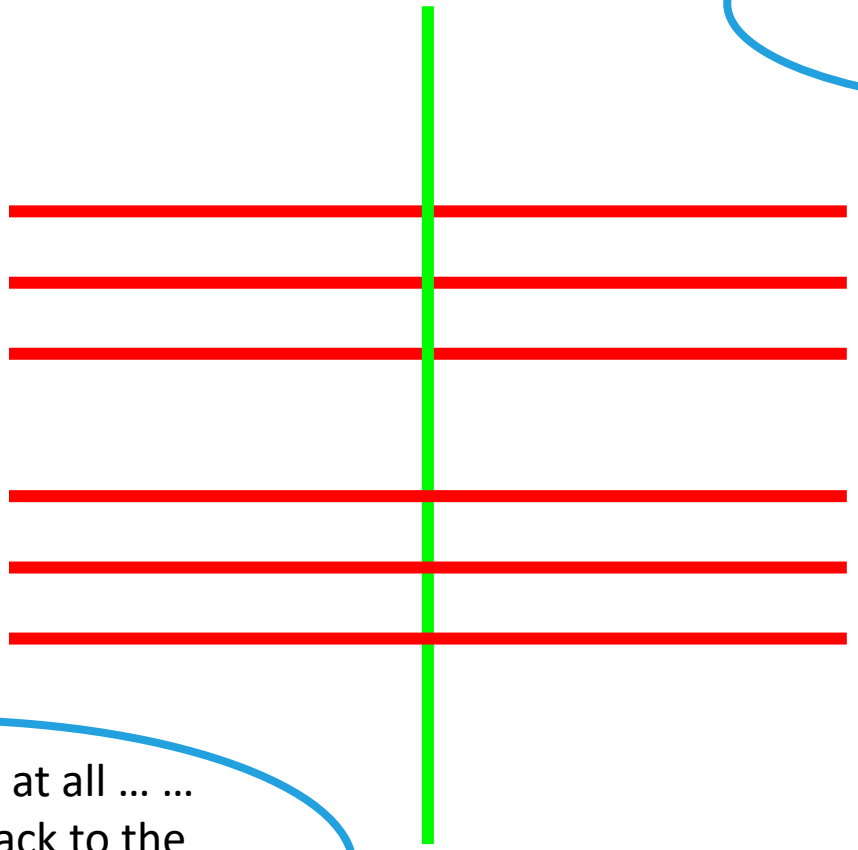
… … … …. …… ……
… fine … ….

Making the required changes …



… is still not enough

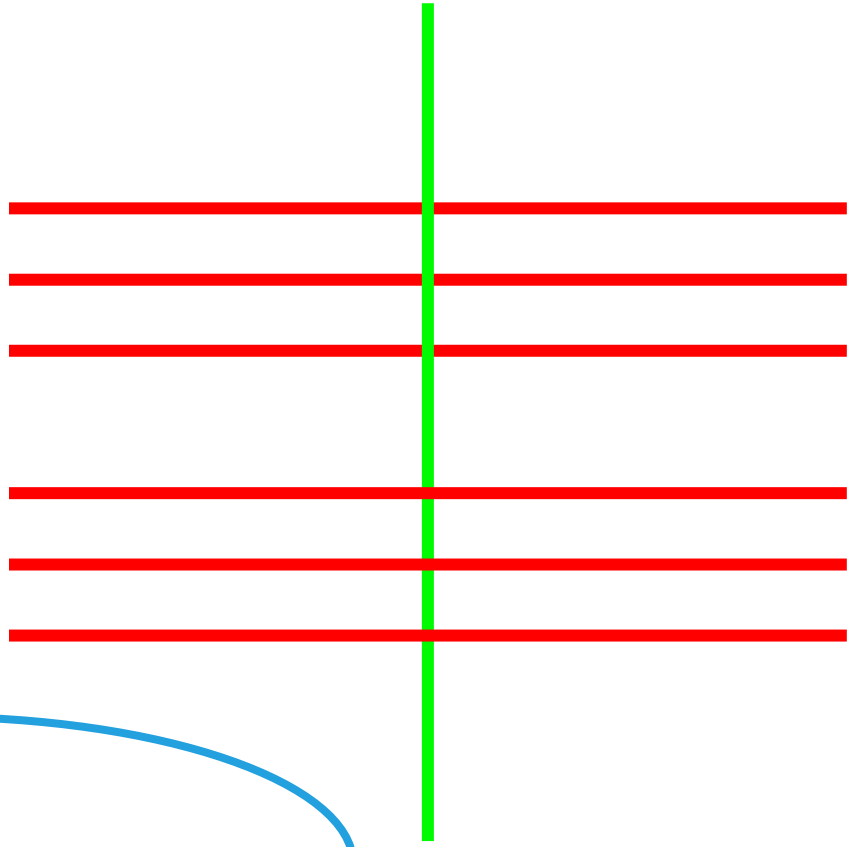
Sorry, uninspiring at all … … yeah … let's go back to the scrambled one you did before!



What about now?

Better the old solution !!! (…which was not good)

That
was
PERFECT



Everything has to be re-done !!!!

Simple … with a version control system

git

Git Features

git

Create, edit, delete file(s)

Git Features

git

Create, edit, delete file(s)

Compare the changes

```
git status
git diff
```

Git Features

git

Create, edit, delete file(s)

Compare the changes

```
git status
git diff
```

Stage the changes for commit

```
git add file-name
```

Git Features

git

Create, edit, delete file(s)

Compare the changes

```
git status
git diff
```

Stage the changes for commit

```
git add file-name
```

Commit the staged files

```
git commit -m "The commit message"
```

Git Features

git

Create, edit, delete file(s)

Compare the changes

```
git status
git diff
```

Stage the changes for commit

```
git add file-name
```

Commit the staged files

```
git commit -m "The commit message"
```

Keep the history in check

```
git log
```

Credentials Setup

Configure you GitHub credentials!

```
git config --global user.name "Your Name"

git config --global user.email "you.email@address.com"
```

How to start the WebUI

```
cd your-git-repo

git webui                    (use CTRL+C to stop it..)
```

Hands-on: Version Control

Try and experiment on your first *gitted* project!

```
git init

git status
                                        (create the first text file)
git status

git add first.txt
git commit -m "My First Commit"
                                        (crate the second text file)
git log

git add second.txt
git commit -m "Adding a second file"
```

Hands-on: Version Control

git

Remote access

Local Branches

- master

matbelcao HEAD -> refs/heads/master 4/5/2022, 17:12:42 1311e08

Adding a second file

matbelcao 4/5/2022, 17:06:36 09adb36

My First Commit

Commit | Tree

Ignore Whitespace | Complete file | Context: 3 - + | Explore

```
commit 1311e085575bf1f5858d71d2402fc4a8dacc1c2d
Author: matbelcao <10503336@polimi.it>
Date:   Wed May 4 15:12:42 2022 +0000
    Adding a second file
diff --git a/second.txt b/second.txt
new file mode 100644
index 0000000..b19fb57
--- /dev/null
+++ b/second.txt
@@ -0,0 +1 @@
+This is my second commit !!!
\ No newline at end of file
```

Hands-on: Version Control

Try and experiment on your first *gitted* project!

```
git init

git status
                                        (create the first text file)
git status

git add first.txt
git commit -m "My First Commit"
                                        (crate the second text file)
git log

git add second.txt
git commit -m "Adding a second file"

git log

git diff idcommit_1 idcommit_2
git revert idcommit_2                   (undo the second commit)

git log
```

Hands-on: Version Control

git

Remote access

Local Branches

- master

matbelcao HEAD -> refs/heads/master 4/5/2022, 17:19:21 0bea87c
Revert "Adding a second file"

matbelcao 4/5/2022, 17:12:42 1311e08
Adding a second file

matbelcao 4/5/2022, 17:06:36 09adb36
My First Commit

Commit | Tree

Ignore Whitespace | Complete file | Context: 3 - + | Explore

```
commit 0bea87ccc2e1f0249de80cb92a4519fd2c8acc59
Author: matbelcao <10503336@polimi.it>
Date:   Wed May 4 15:19:21 2022 +0000
    Revert "Adding a second file"

    This reverts commit 1311e085575bf1f5858d71d2402fc4a8dacc1c2c
diff --git a/second.txt b/second.txt
deleted file mode 100644
index b19fb57..0000000
--- a/second.txt
+++ /dev/null
@@ -1 +0,0 @@
-This is my second commit !!!
\ No newline at end of file
```

Git Usage

git

Single developer

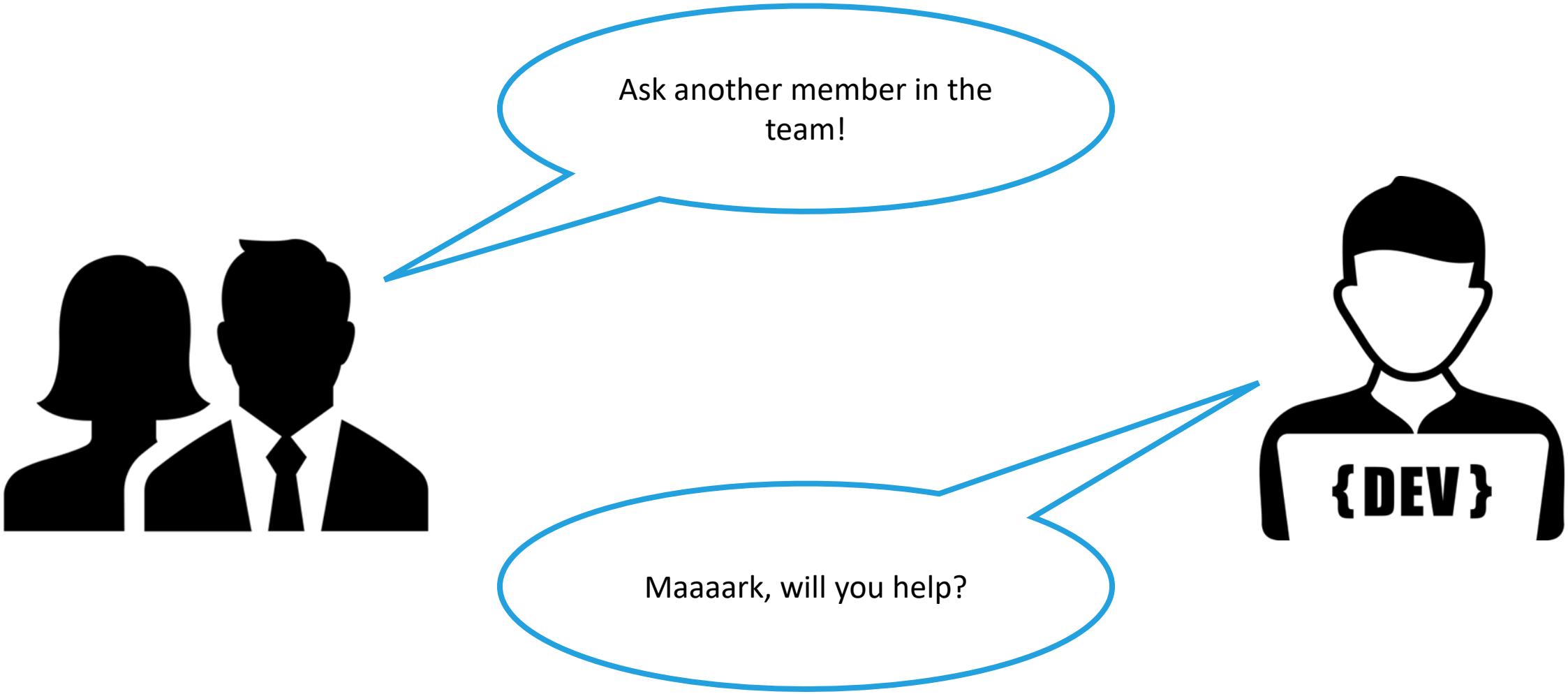
- Tracks evolution
- Builds history
- Navigate the history

Extending to a more complex project

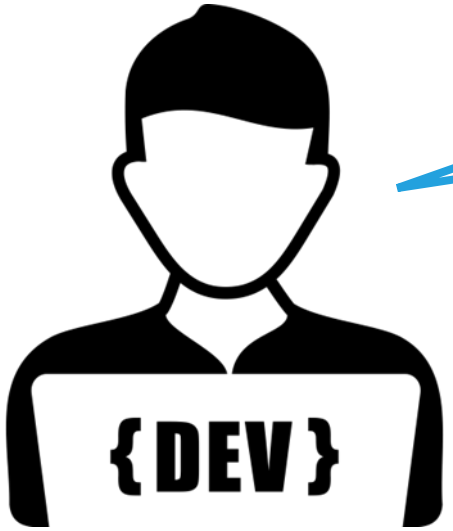
New feat req!
We need all the lines to be be perpendicular and half of them should be green!

I can’t make it on my own!

Extending to a more complex project



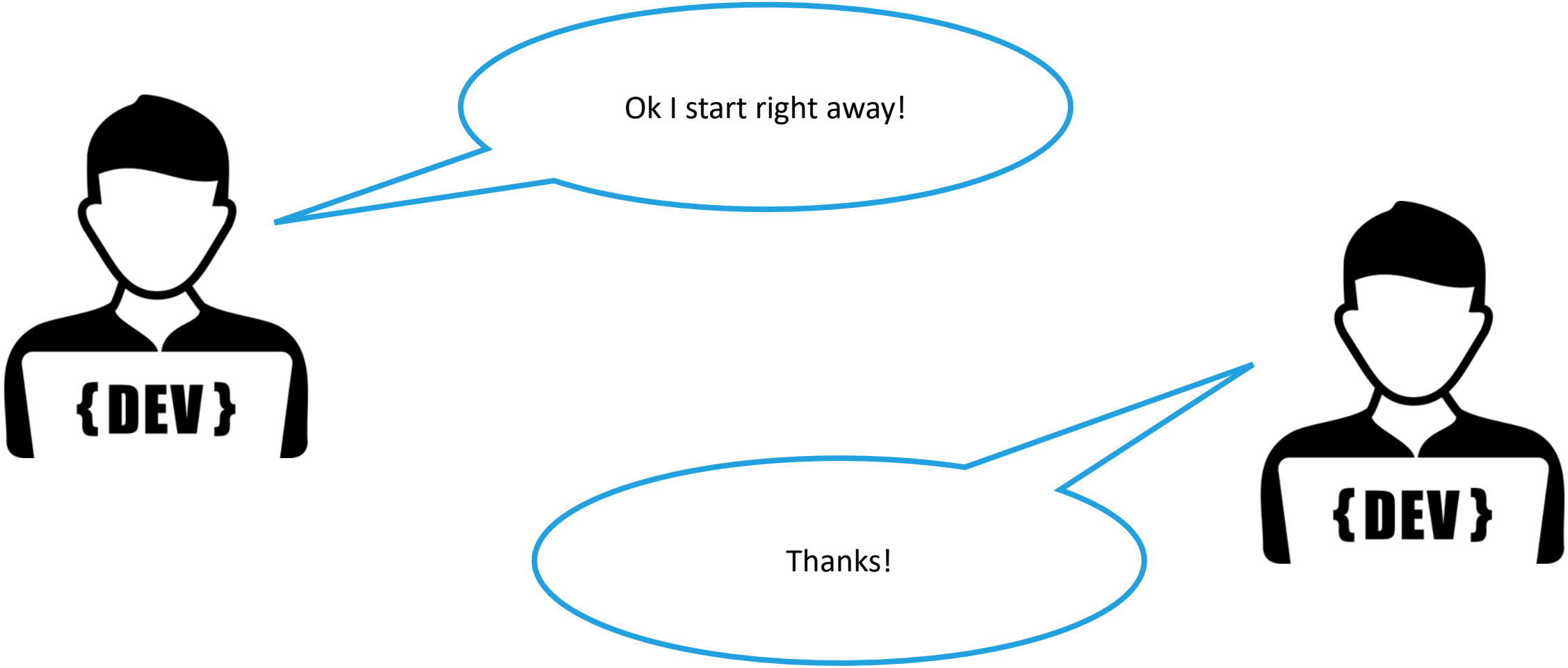
Developing within a team …



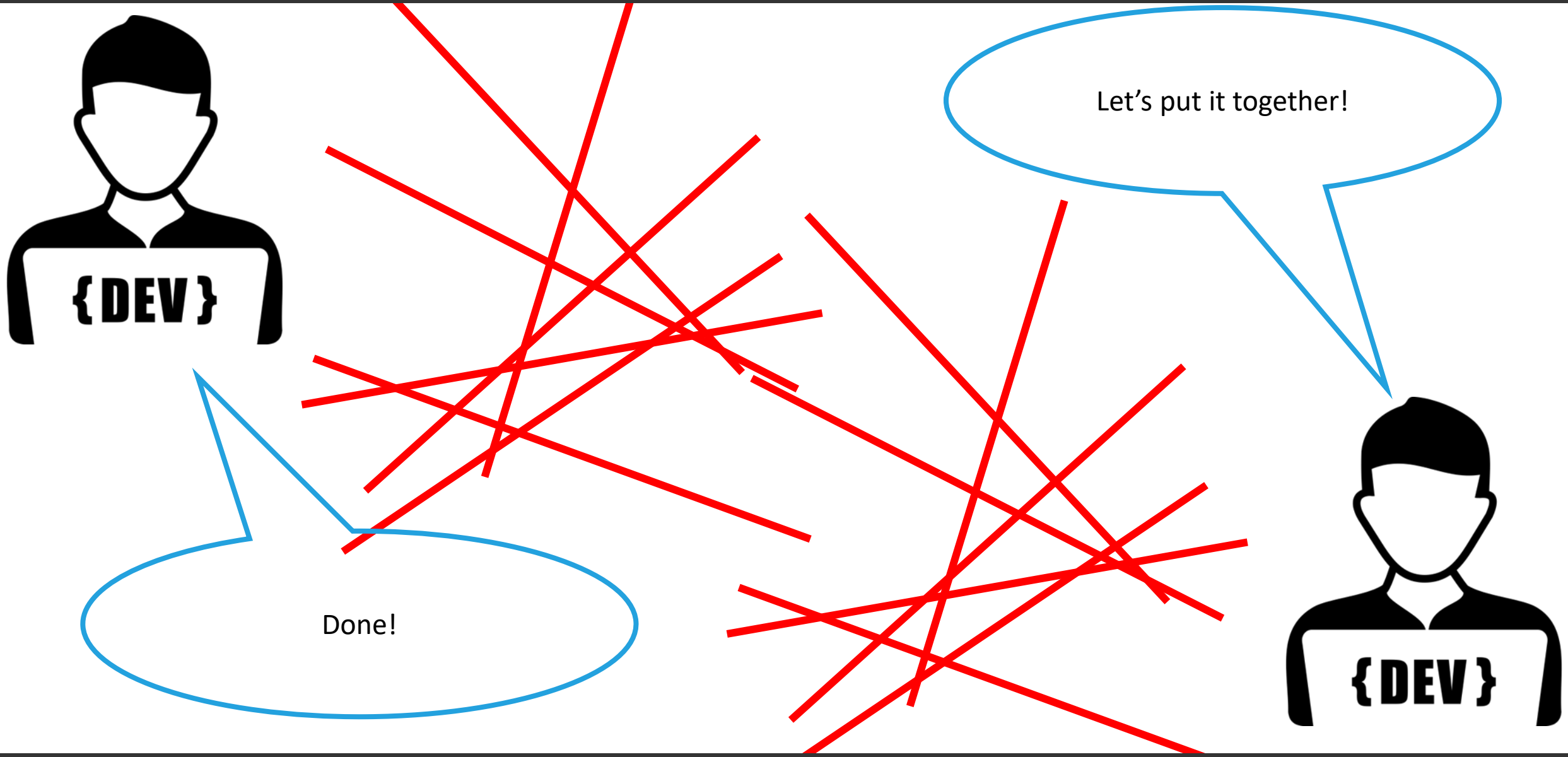
Sure! What can I do?

You color half of the lines green, I'll grid them up!

Developing within a team …



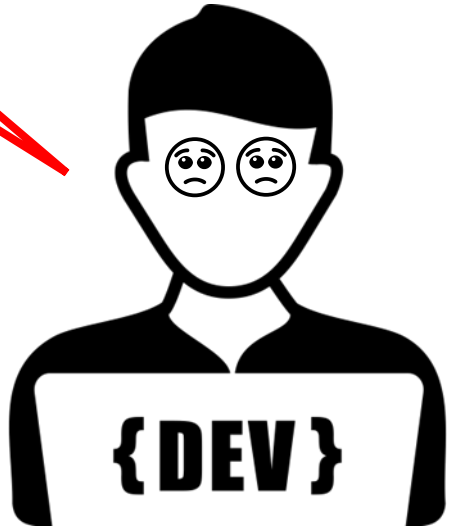
… requires coordination …



… to solve conflicts !!

HOW?!?!?!

HOW?!?!?!



Simple … with a version control system

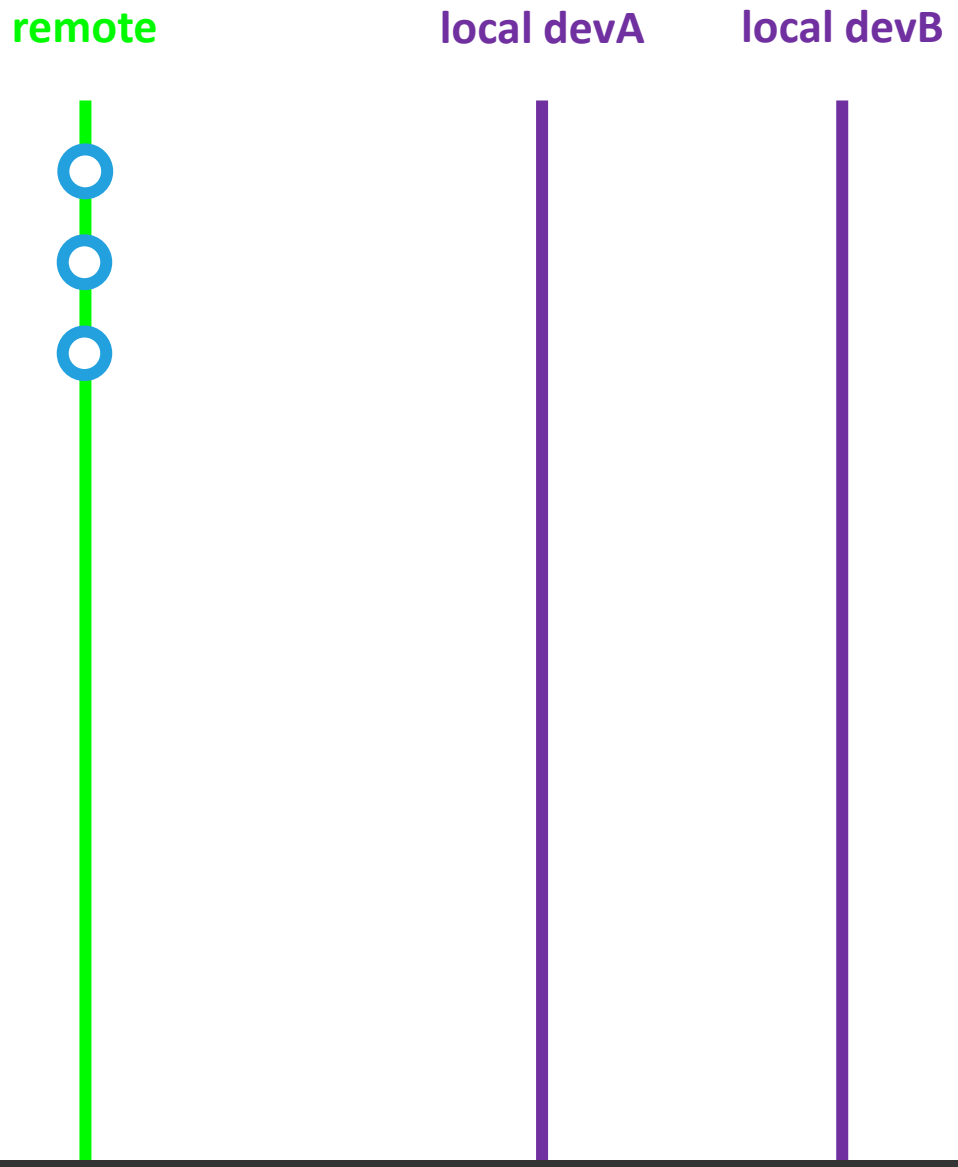
git

Team Collaboration

git

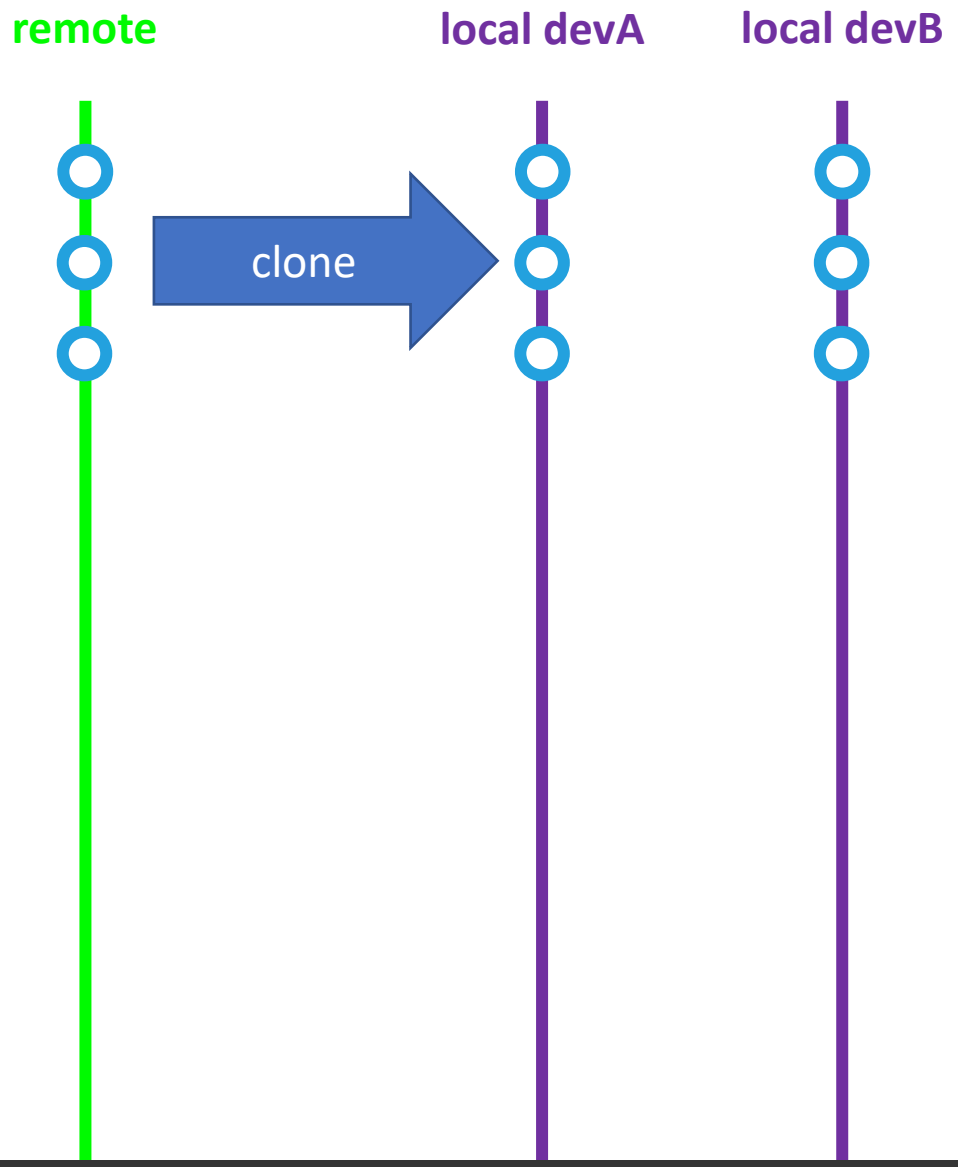
Team Collaboration

git



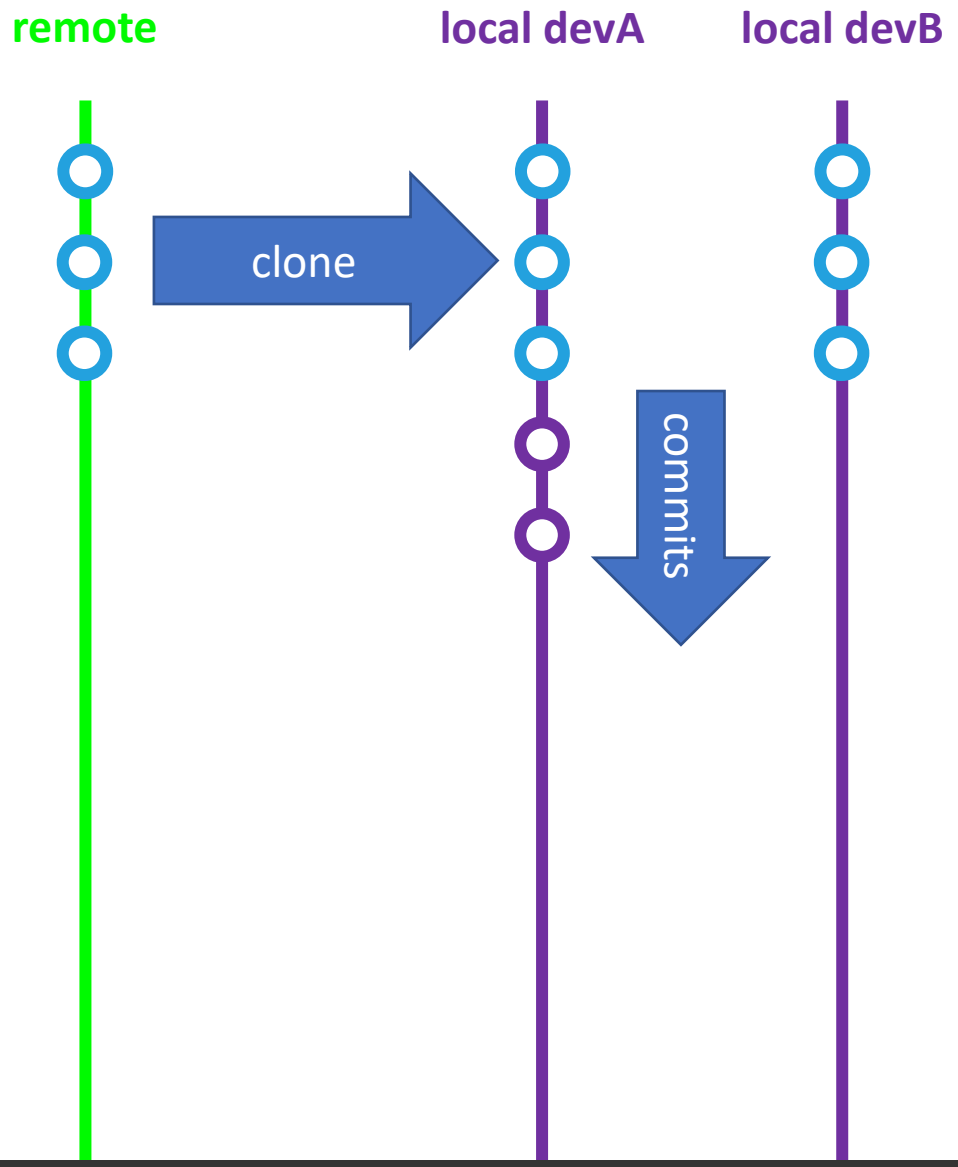
Team Collaboration

git



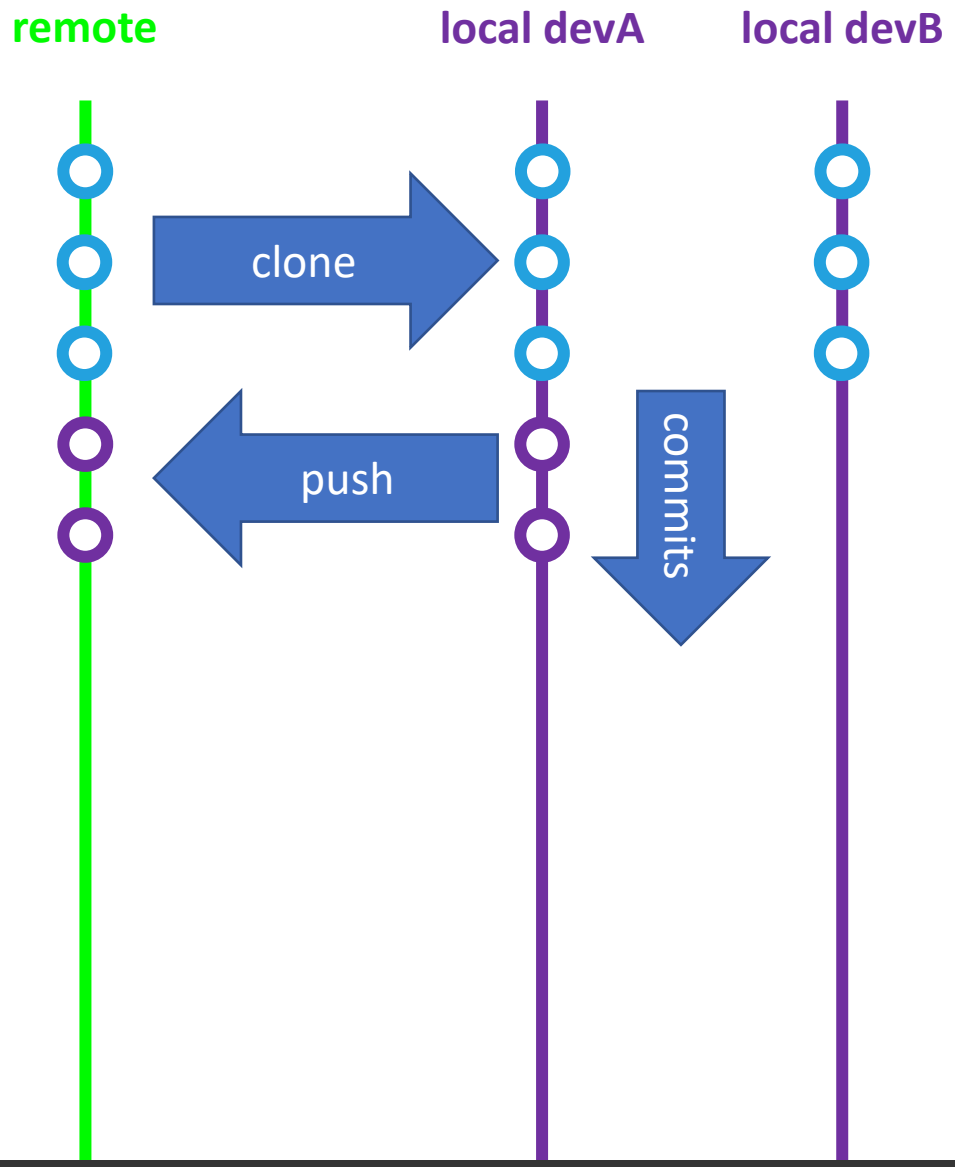
Team Collaboration

git



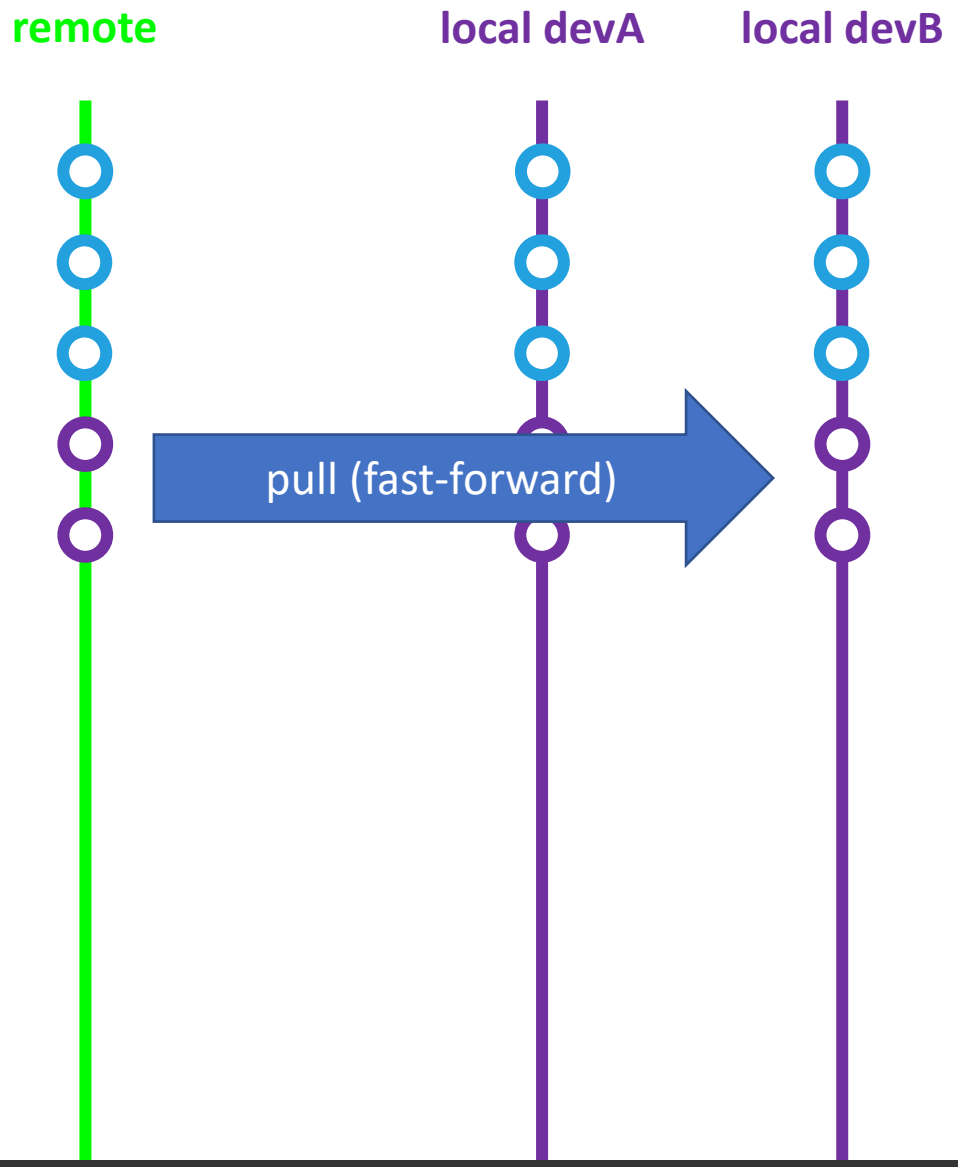
Team Collaboration

git



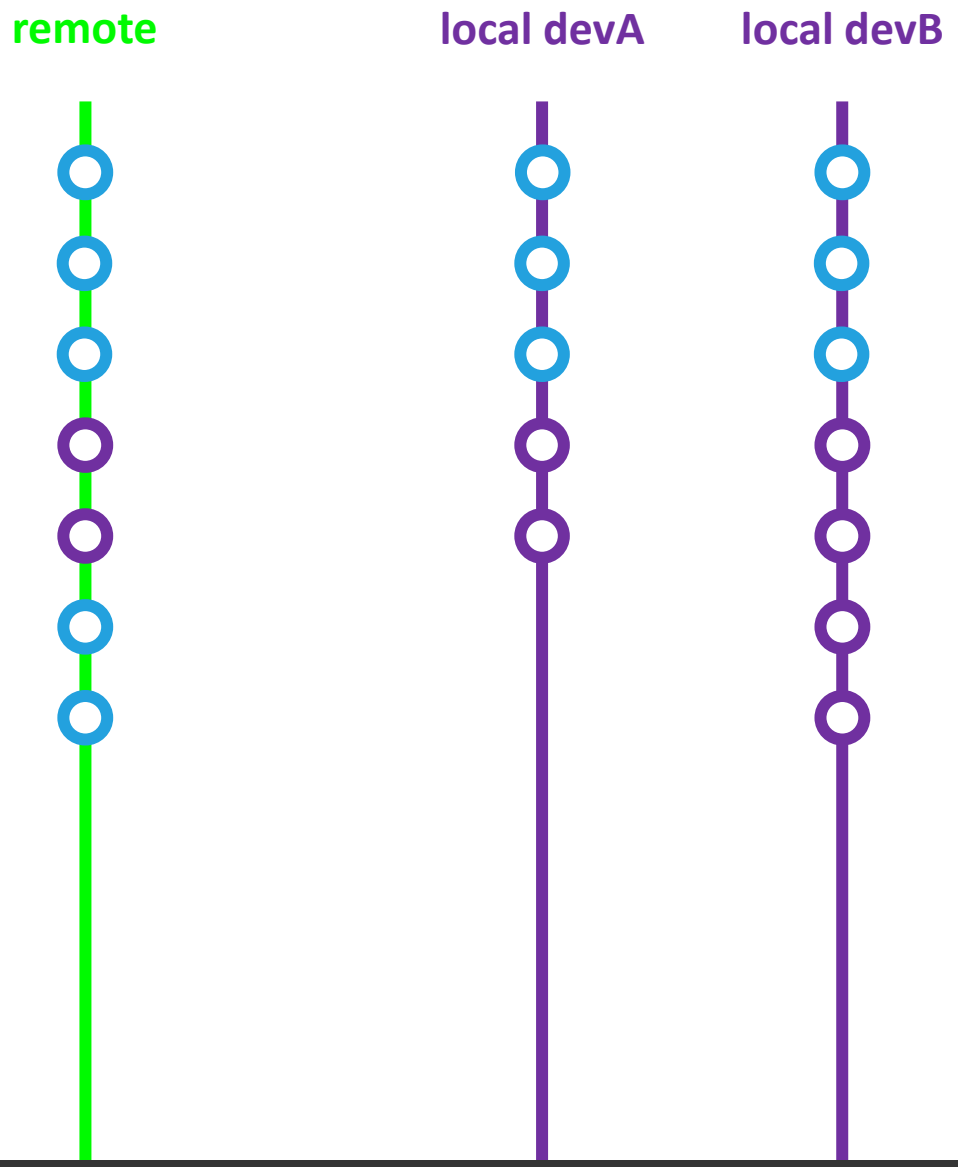
Team Collaboration

git



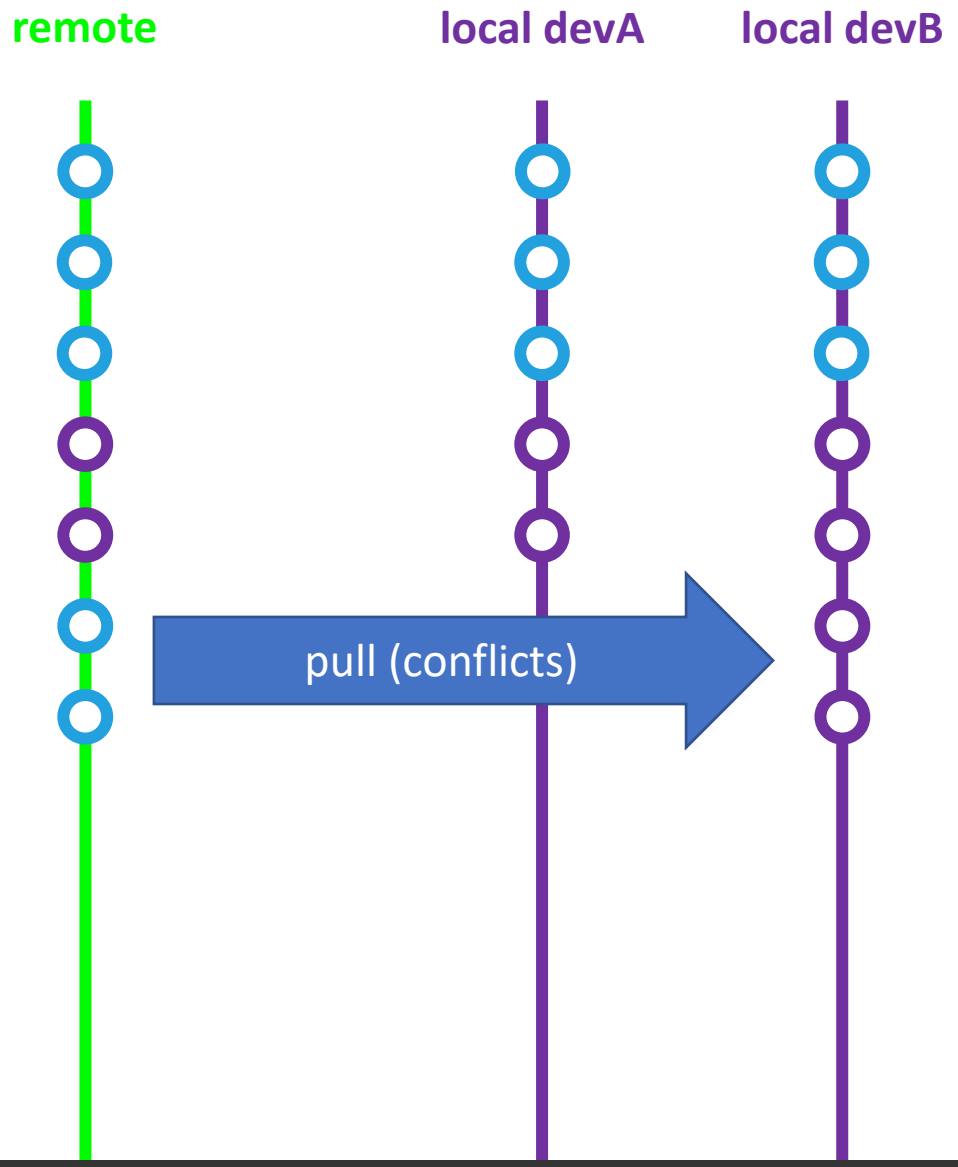
Team Collaboration

git



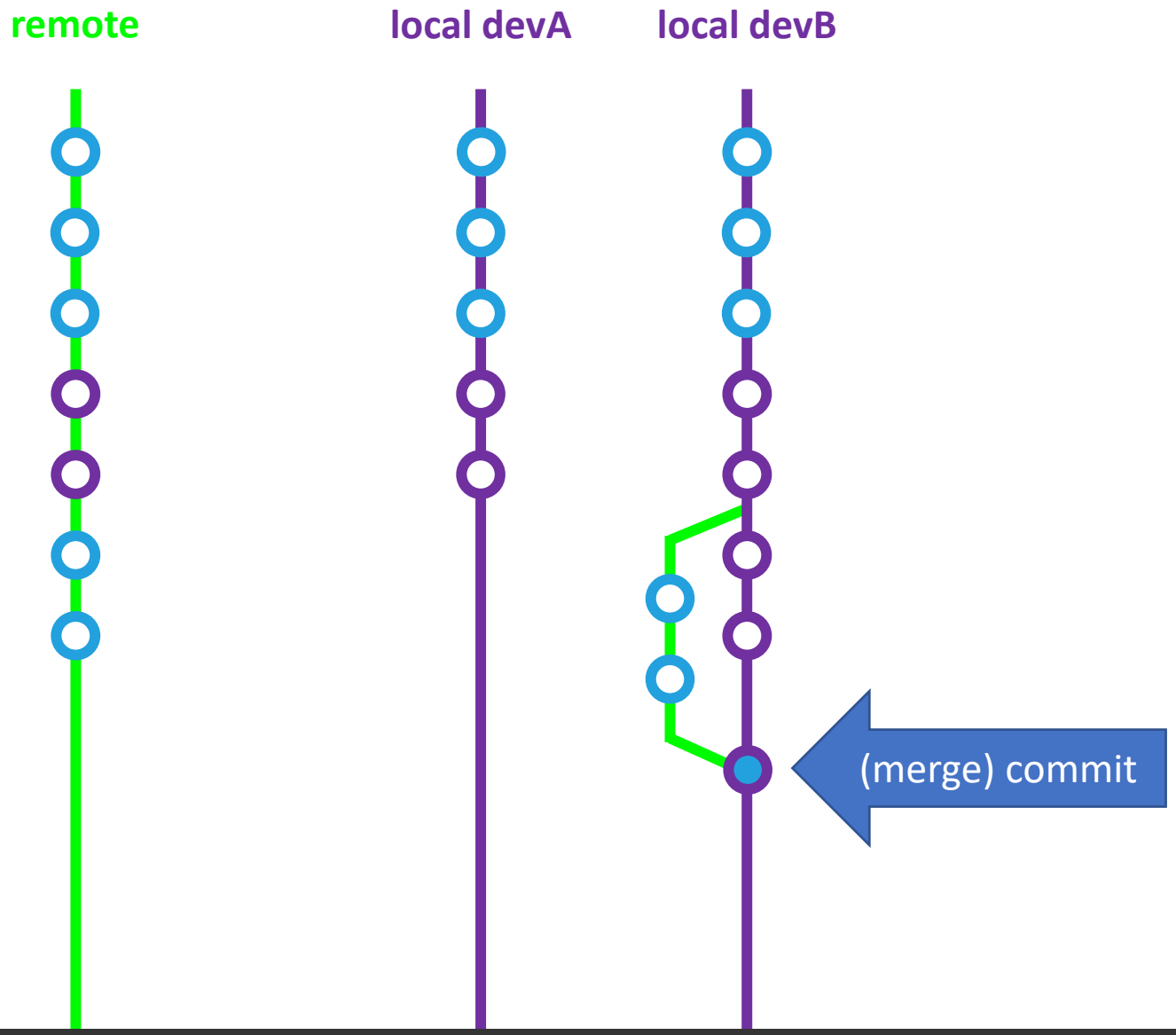
Team Collaboration

git



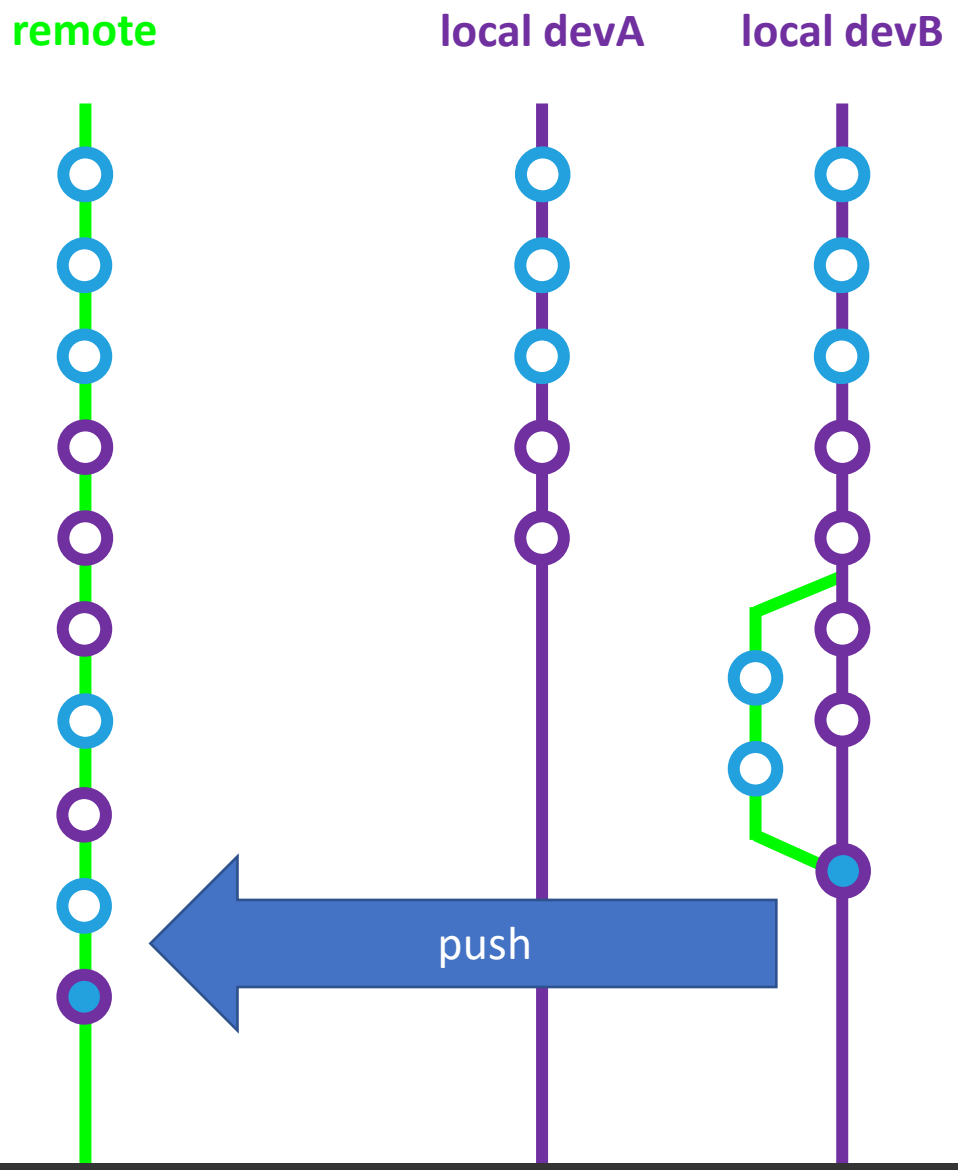
Team Collaboration

git



Team Collaboration

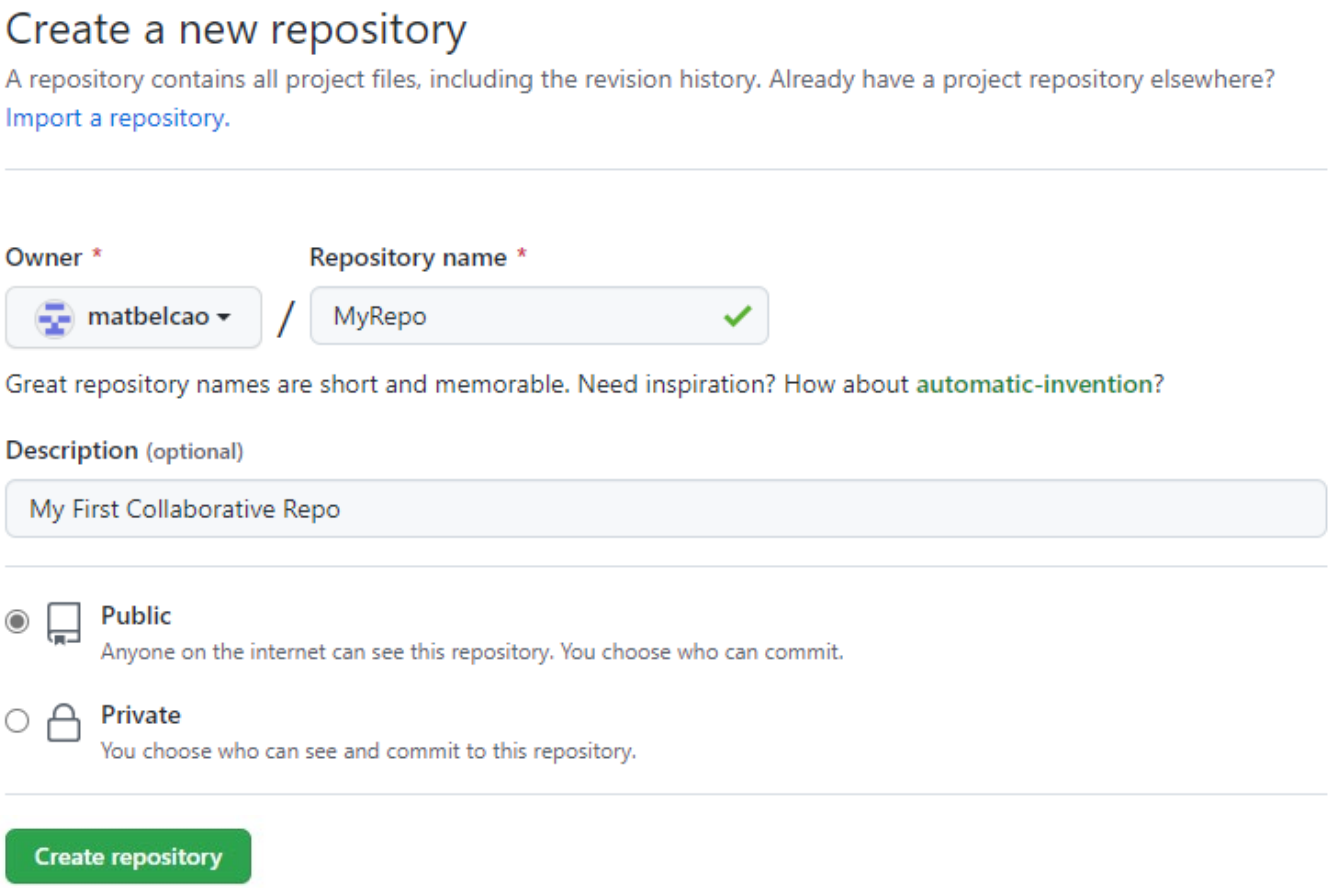
git



Hands-On: Collaborative Development

Create a repository on GitHub (use the GUI)

https://github.com/new



Hands-On: Collaborative Development

Try and experiment on your first **collaborative** *repo*!

git

```
git init
git remote add origin repository
```

Dave creates *main.txt*

```
git add main.txt
git commit -m "Dave file v1"
git push --set-upstream origin master
```

Dave modifies *main.txt*

```
git add main.txt
git commit -m "Dave file v2"
git push
```

```
git clone repository
git log origin/master
```

Mark doesn't see any change

```
git fetch origin master
git log origin/master
```

Mark see a log change

```
git pull
```

Mark receive the first change, modifies main.txt

```
git add main.txt
git commit -m "Mark file v1"
```

Hands-On: Collaborative Development

Try and experiment on your first **collaborative** *repo*!


```
git push
        Oops, there is a problem!!! Push Rejected
git pull
        Mark Receive the changes, but needs a merge …
git merge
        Mark fixes the conflicts in the file …
git add main.txt
git commit
git push
```

Git Usage

git

Single developer

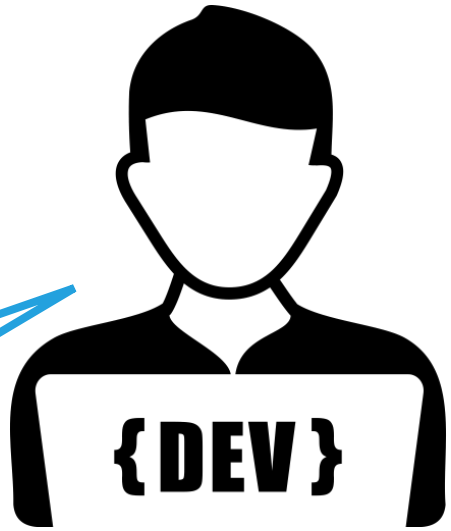
- Tracks evolution
- Builds history
- Navigate the history

Team of developers

- Allow concurrent development
- Track the responsible
- Support in merging changes

Concurrent features development …

New feat req!
We want 7 yellow lines!!!



Sure!

Concurrent features development …

New feat req!
We want 5 red spots!!!

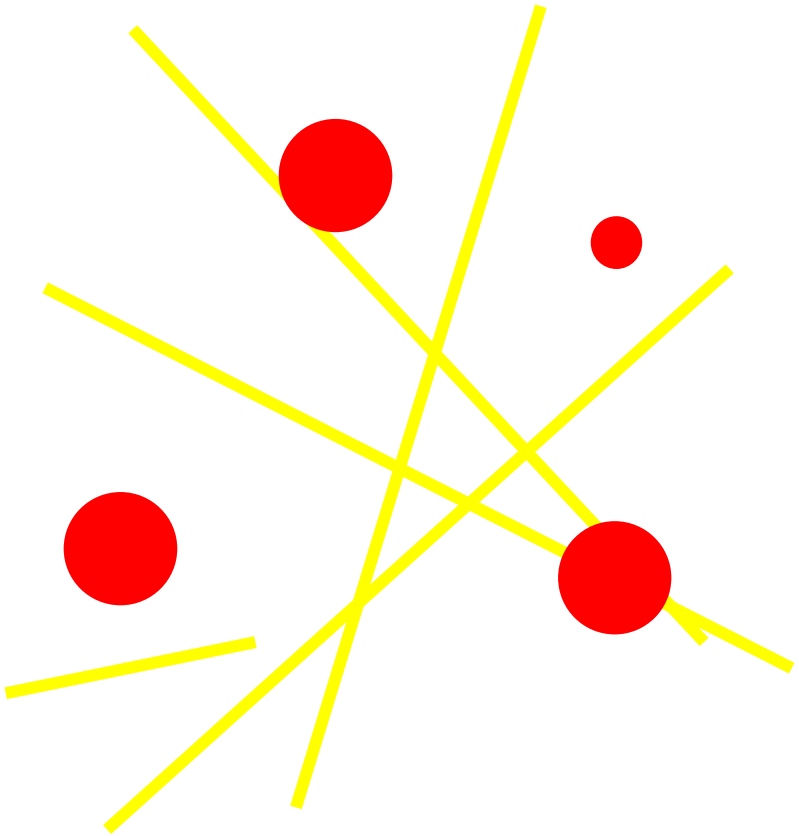
But … I'm still working on the yellow lines

Concurrent features development …

The spots are top priority!

Fine…

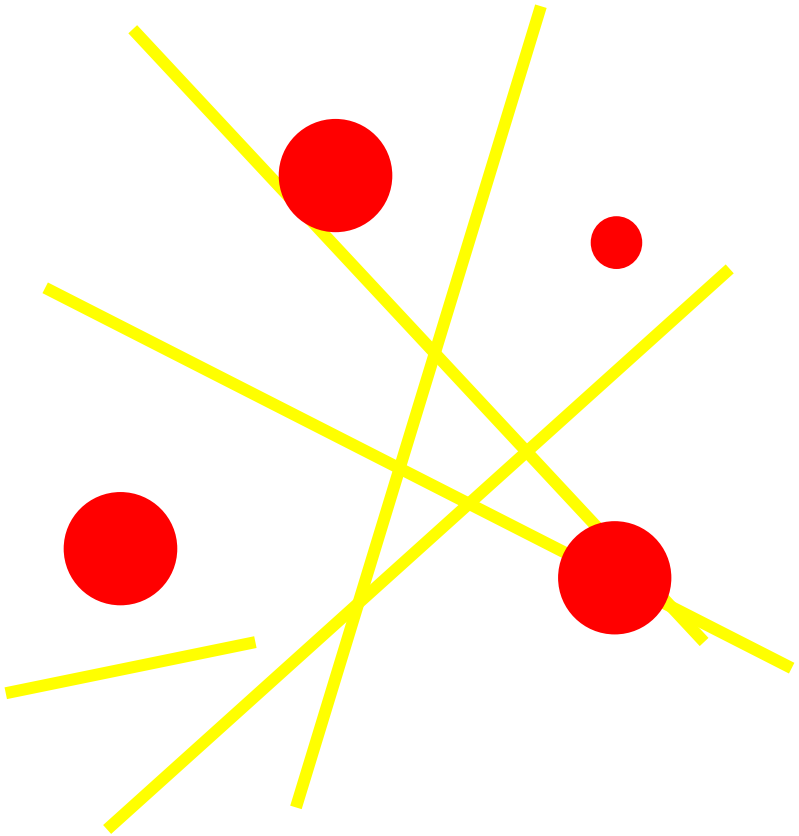
… requires to be managed carefully



OMG! Why are the spots all over the lines?!?!

Because I was working on the lines before…

… requires to be managed carefully



This is a mess! The customer wants the red spots without the lines! Separate them!

Because I was working on the lines before…

… requires to be managed carefully

Simple … with a version control system

git

Hands-on: Branching

Try to create two project's branches, then merge them!

```
git clone repository
```

create *main.txt*, write "test1234"

```
git add main.txt
git commit -m "Main branch commit"
git push
```

```
git branch helloworld
git checkout helloworld
```

write "test45678" in *main.txt*

```
git add main.txt
git commit -m "Secondary branch commit"
git push origin helloworld
```

Hands-on: Branching

Try to create two project's branches, then merge them!

```
git checkout master
```

check *main.txt* … what's the content?

```
git merge helloworld
git push
```

check *main.txt* … what's the content?

Exercise: Branching

Got to: https://learngitbranching.js.org/

Try to make the following two exercises:

- ex2: *Branching in Git*
- ex3: *Merging in Git*

[OPTIONAL]: if you have some time, try also ex4 (*Rebase Introduction*)

Git Usage

git

Single developer

- Tracks evolution
- Builds history
- Navigate the history

Team of developers

- Allow concurrent development
- Track the responsible
- Support in merging changes

Multiple features

- Handle multiple work in progress
- Explicit feature merging

Git Branching Model

git

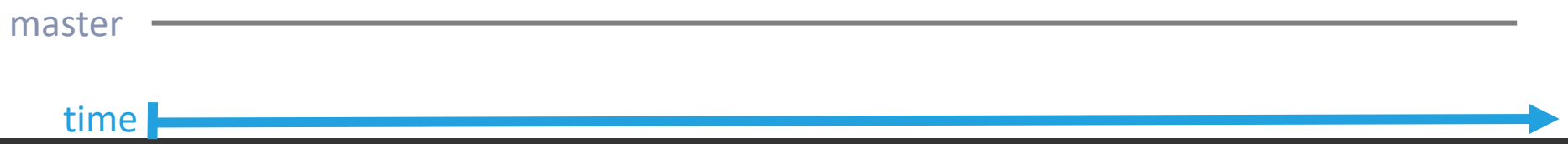
Git Branching Model

gitFlow

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

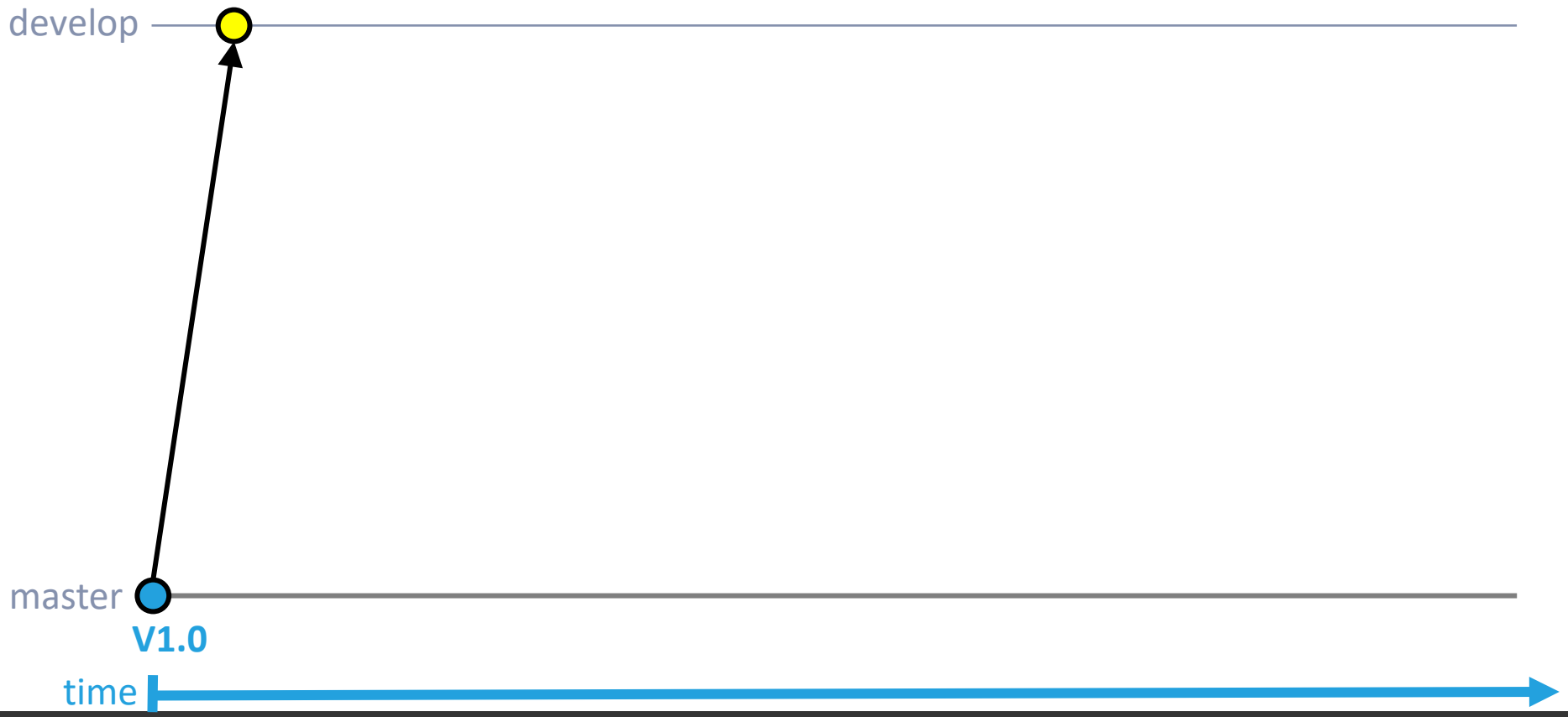
Git Flow

Author: Vincent Driessen
Original blog post:
http://nvie.com/archives/323



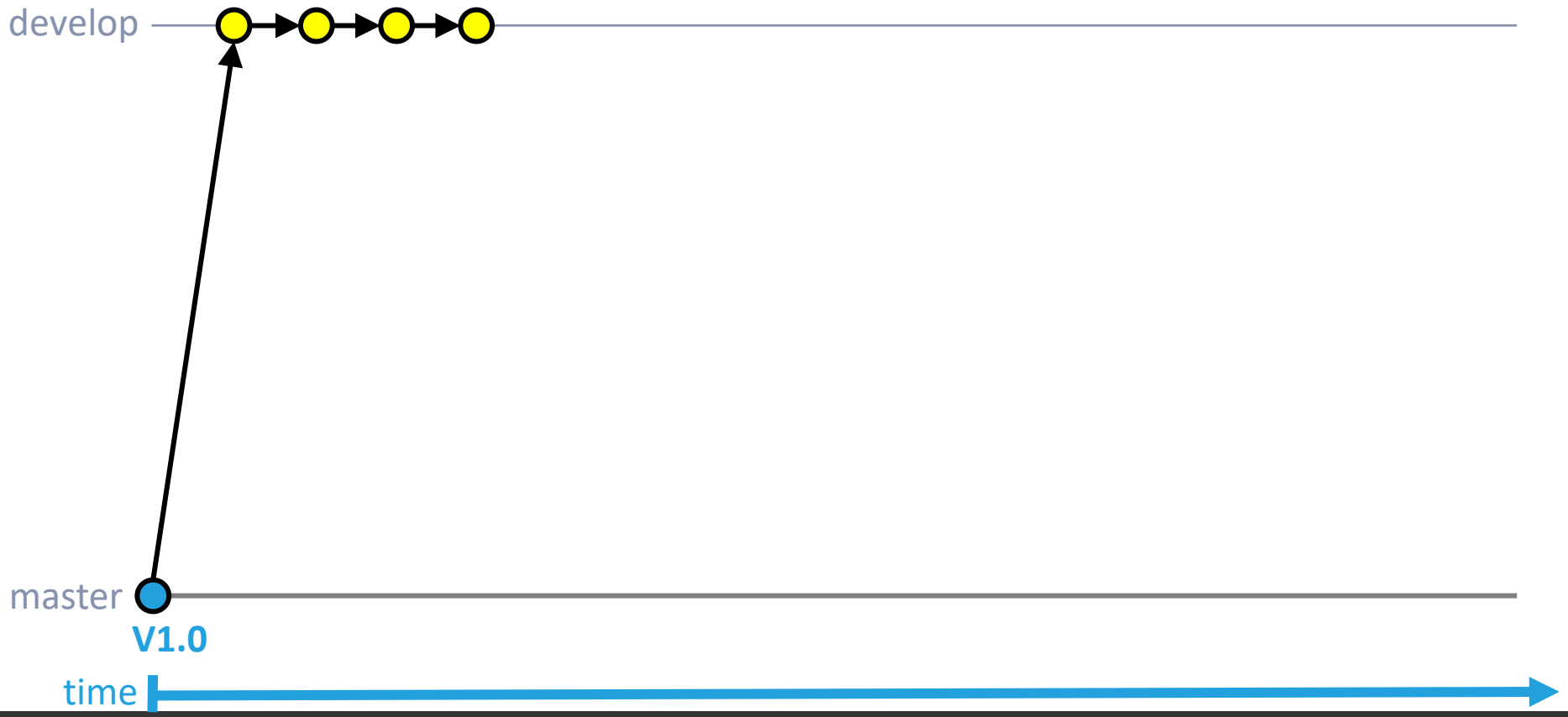
Git Flow

Author: Vincent Driessen
Original blog post:
http://nvie.com/archives/323



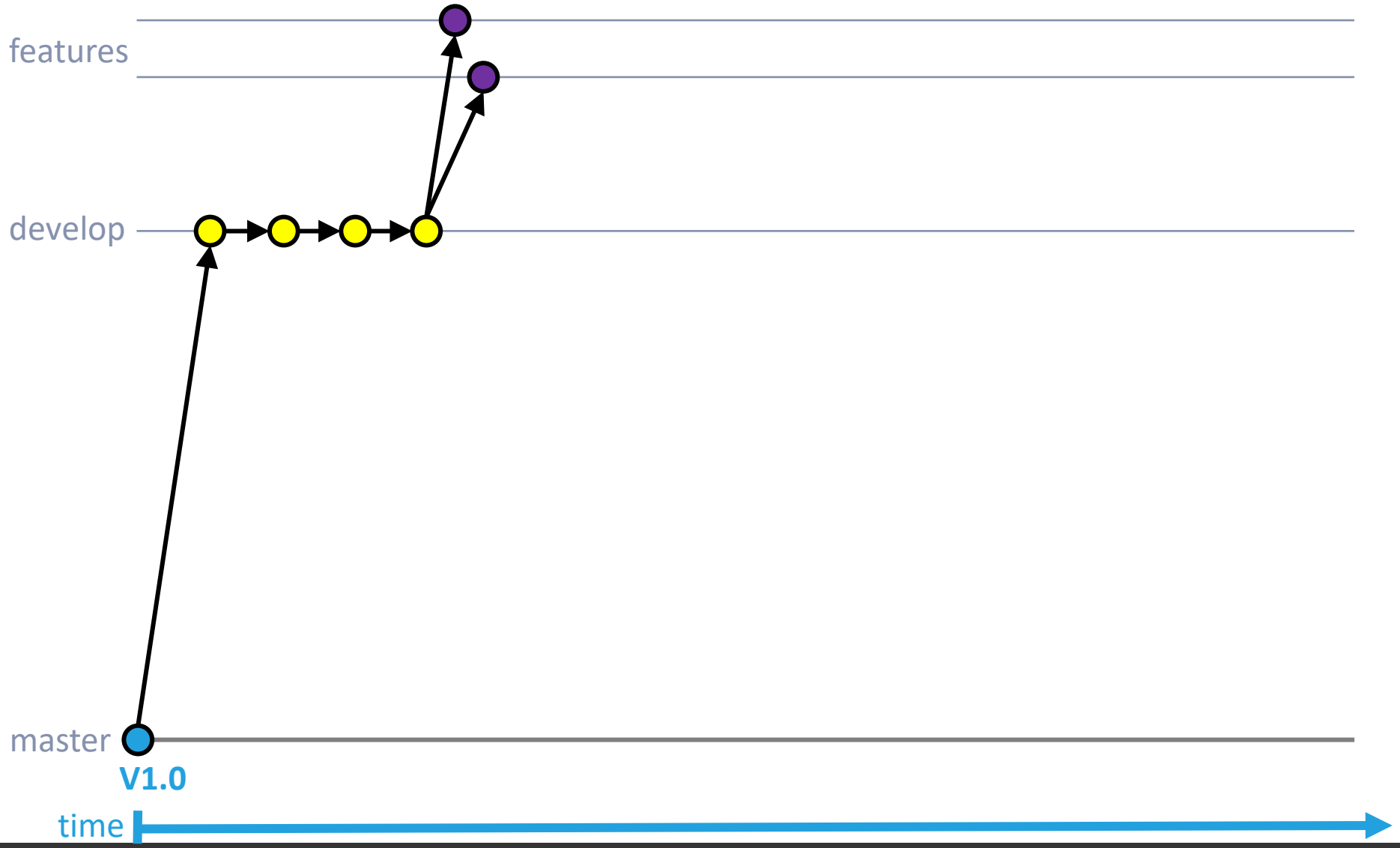
Git Flow

Author: Vincent Driessen
Original blog post:
http://nvie.com/archives/323



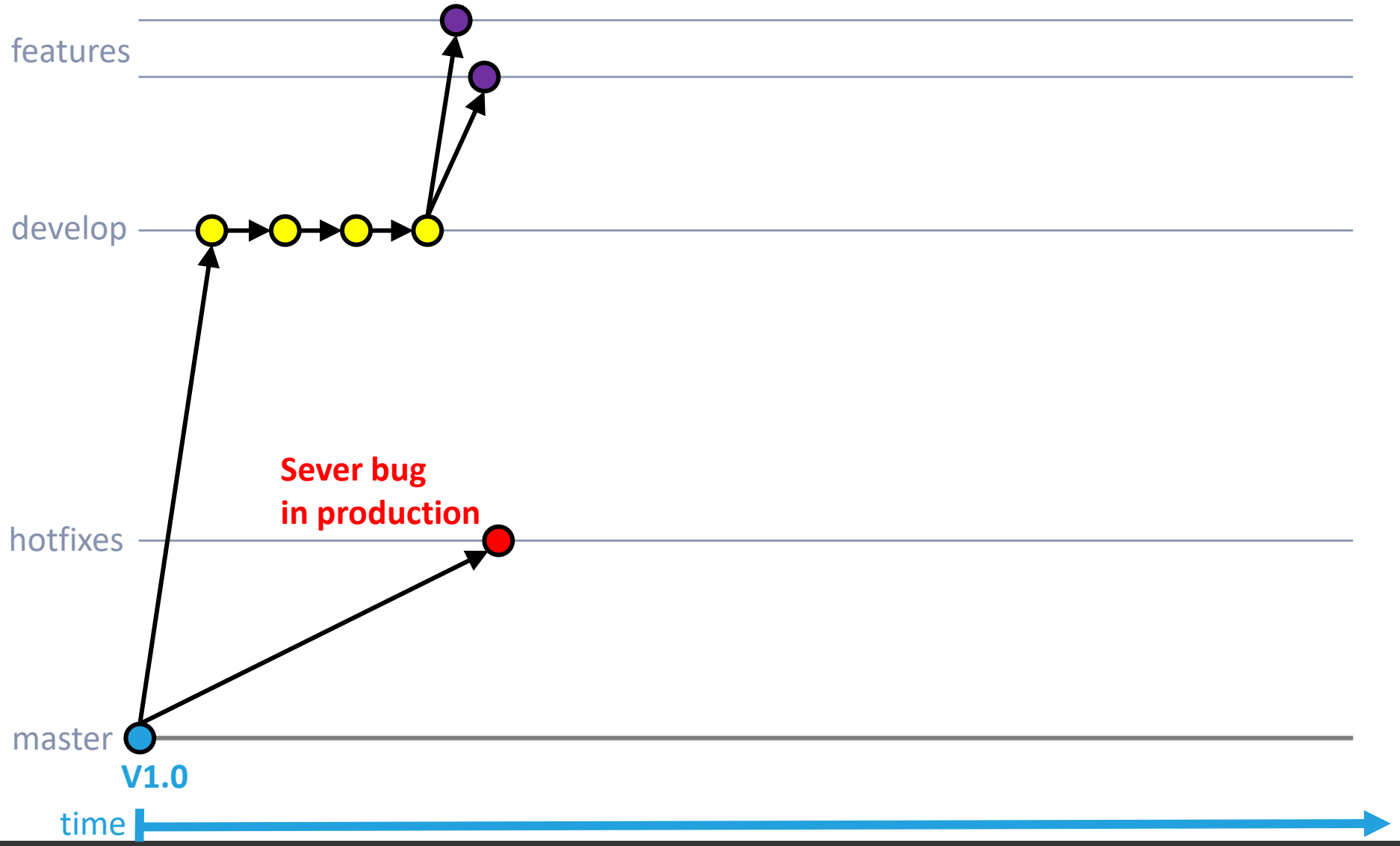
Git Flow

Author: Vincent Driessen
Original blog post:
http://nvie.com/archives/323



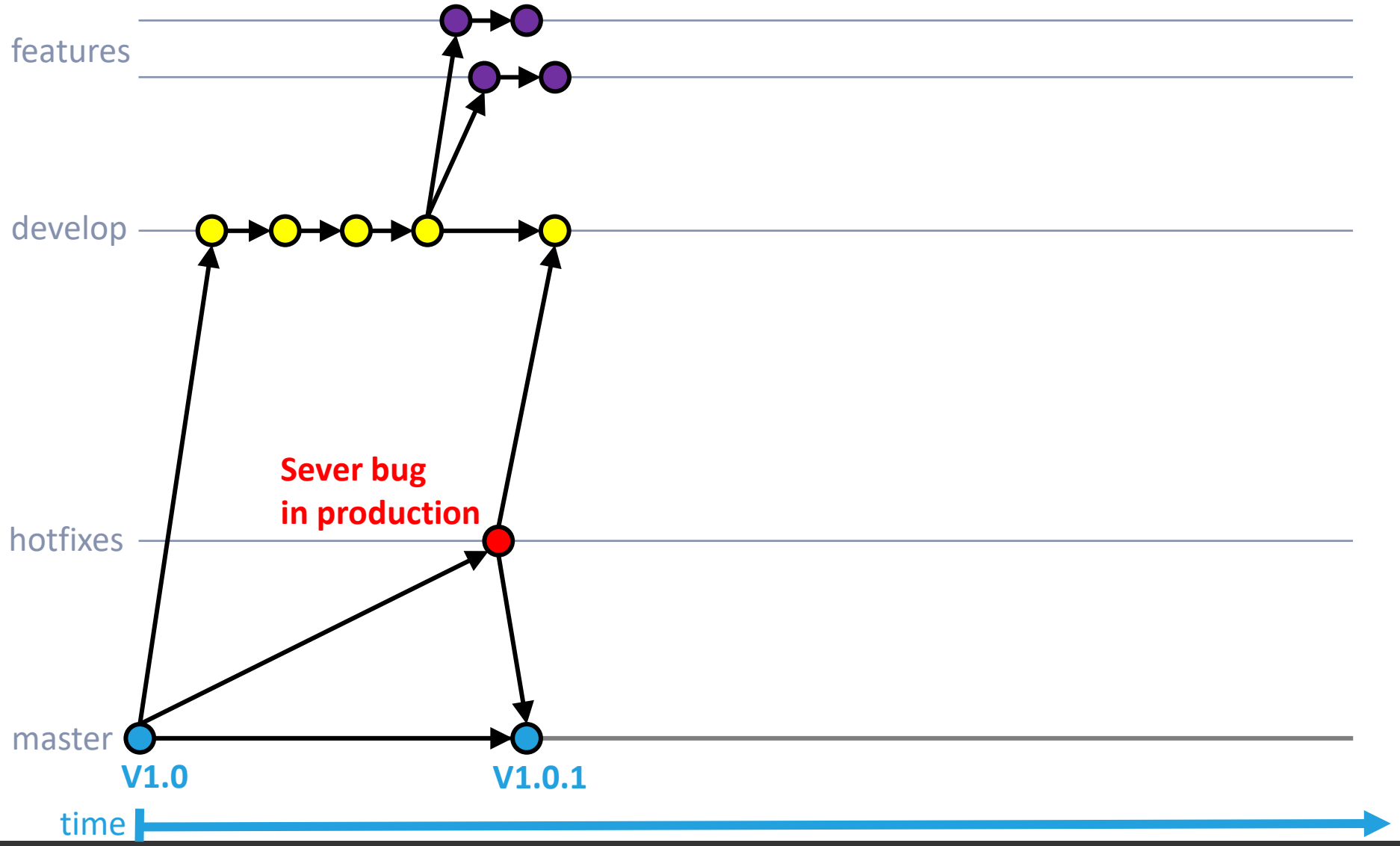
Git Flow

Author: Vincent Driessen
Original blog post:
http://nvie.com/archives/323



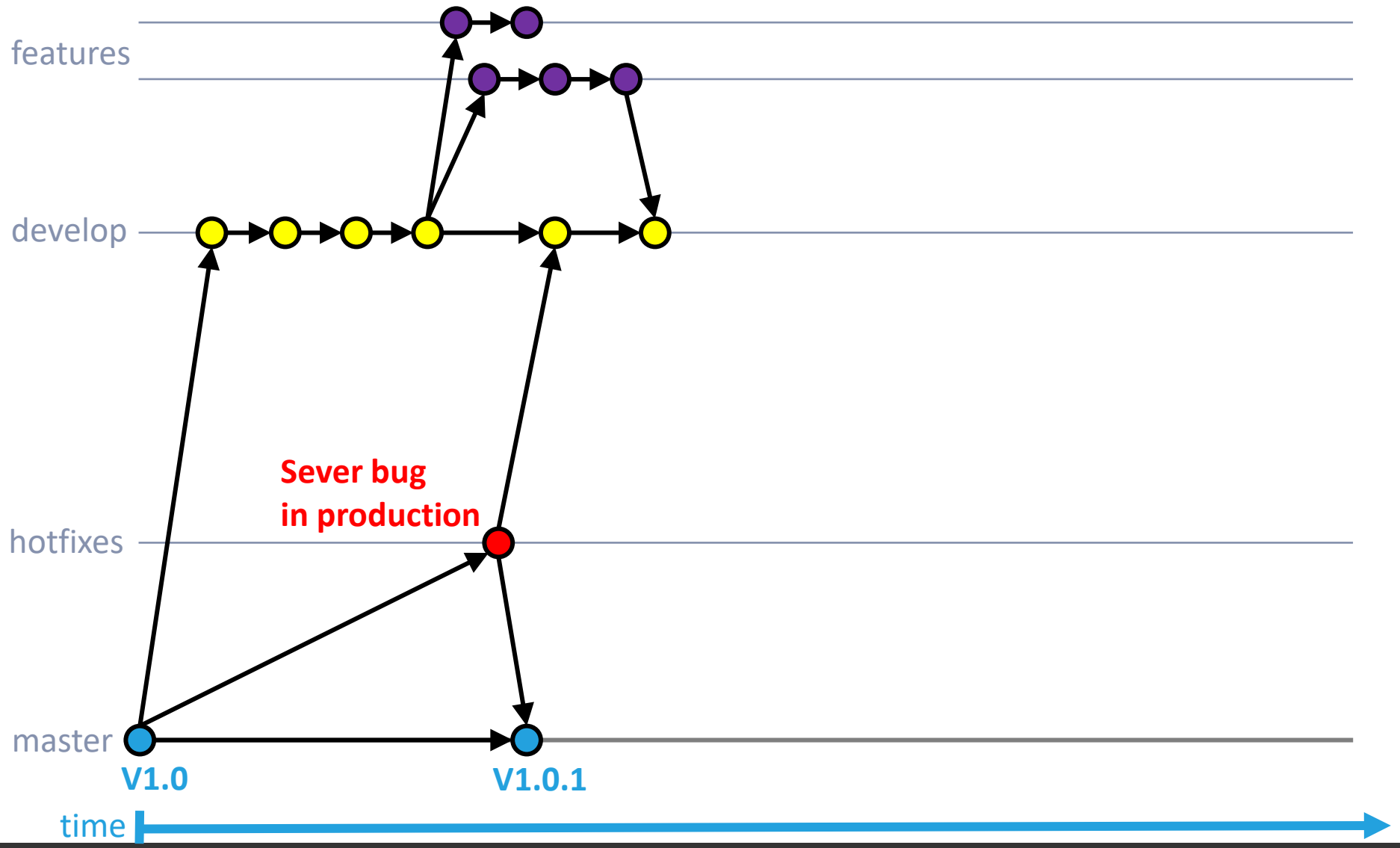
Git Flow

Author: Vincent Driessen
Original blog post:
http://nvie.com/archives/323



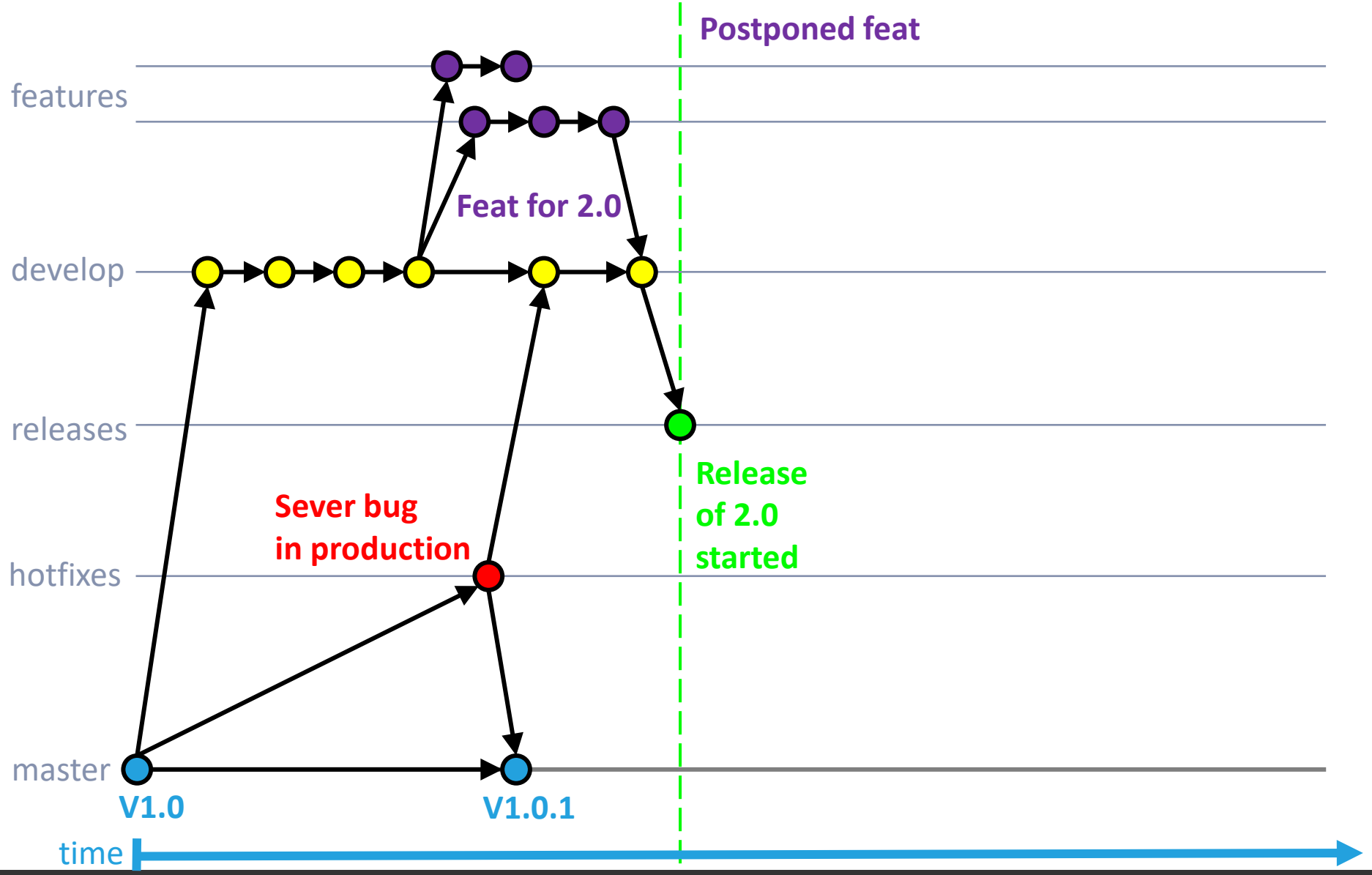
Git Flow

Author: Vincent Driessen
Original blog post:
http://nvie.com/archives/323



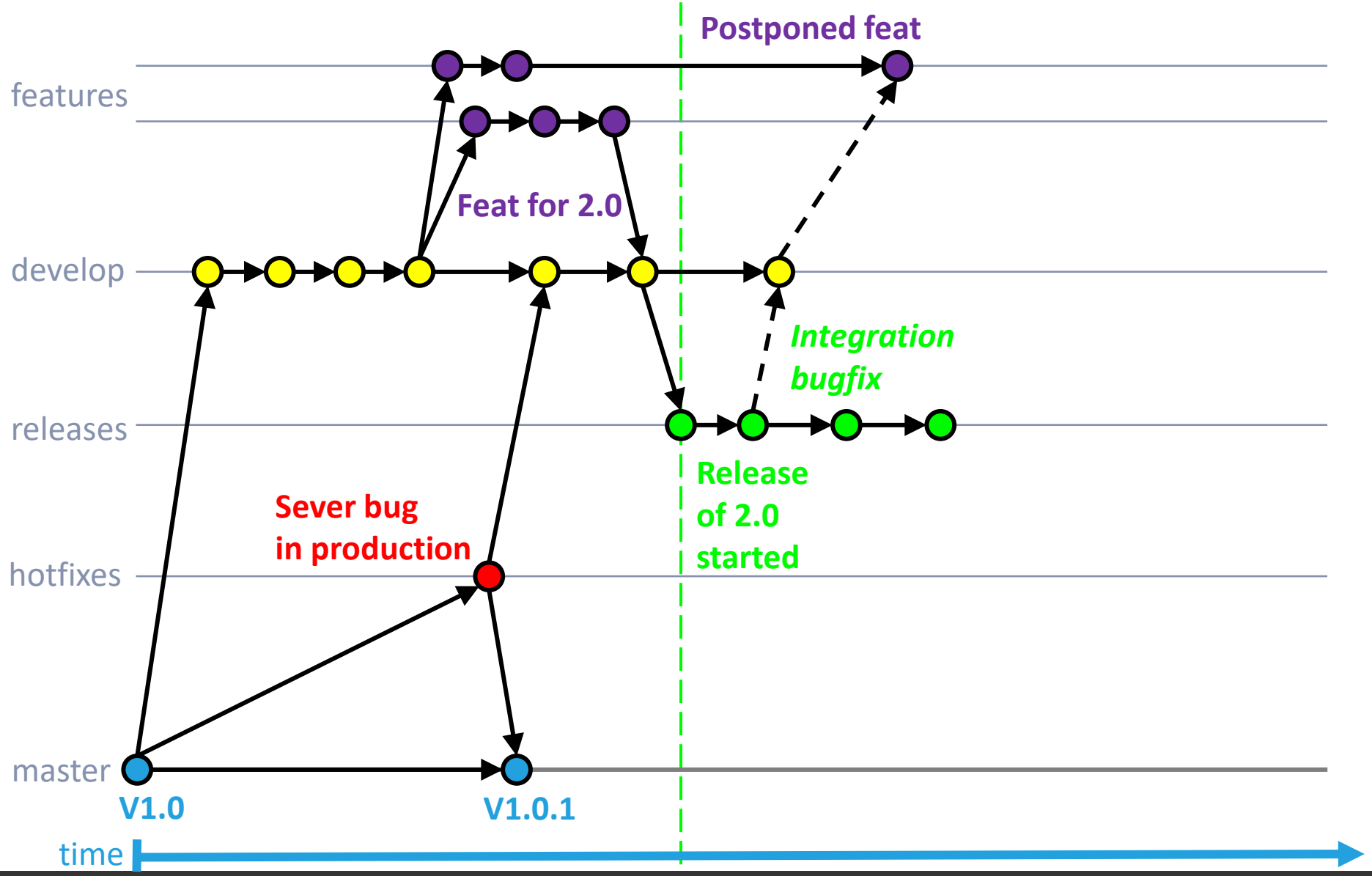
Git Flow

Author: Vincent Driessen
Original blog post:
http://nvie.com/archives/323



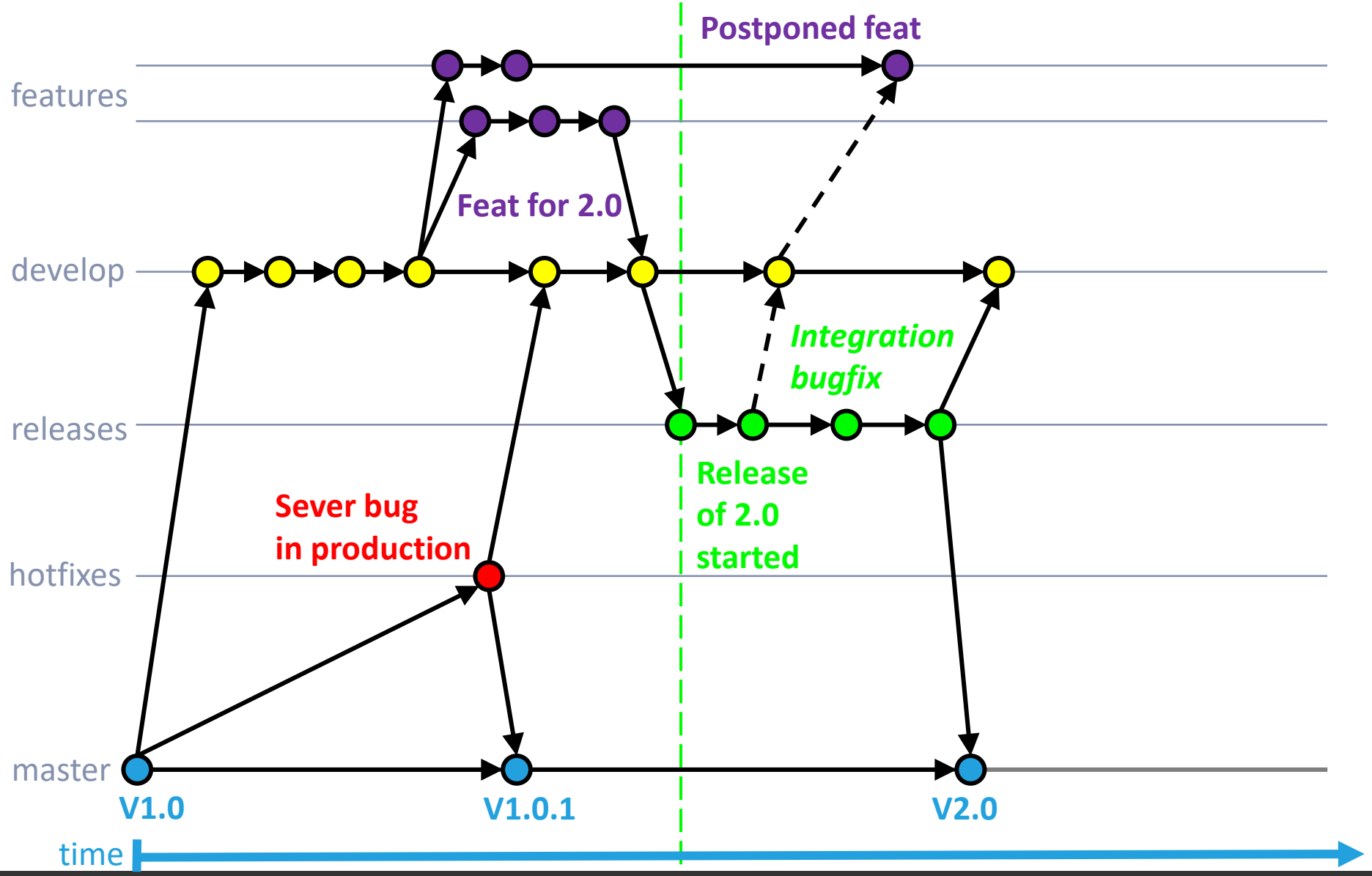
Git Flow

Author: Vincent Driessen
Original blog post:
http://nvie.com/archives/323



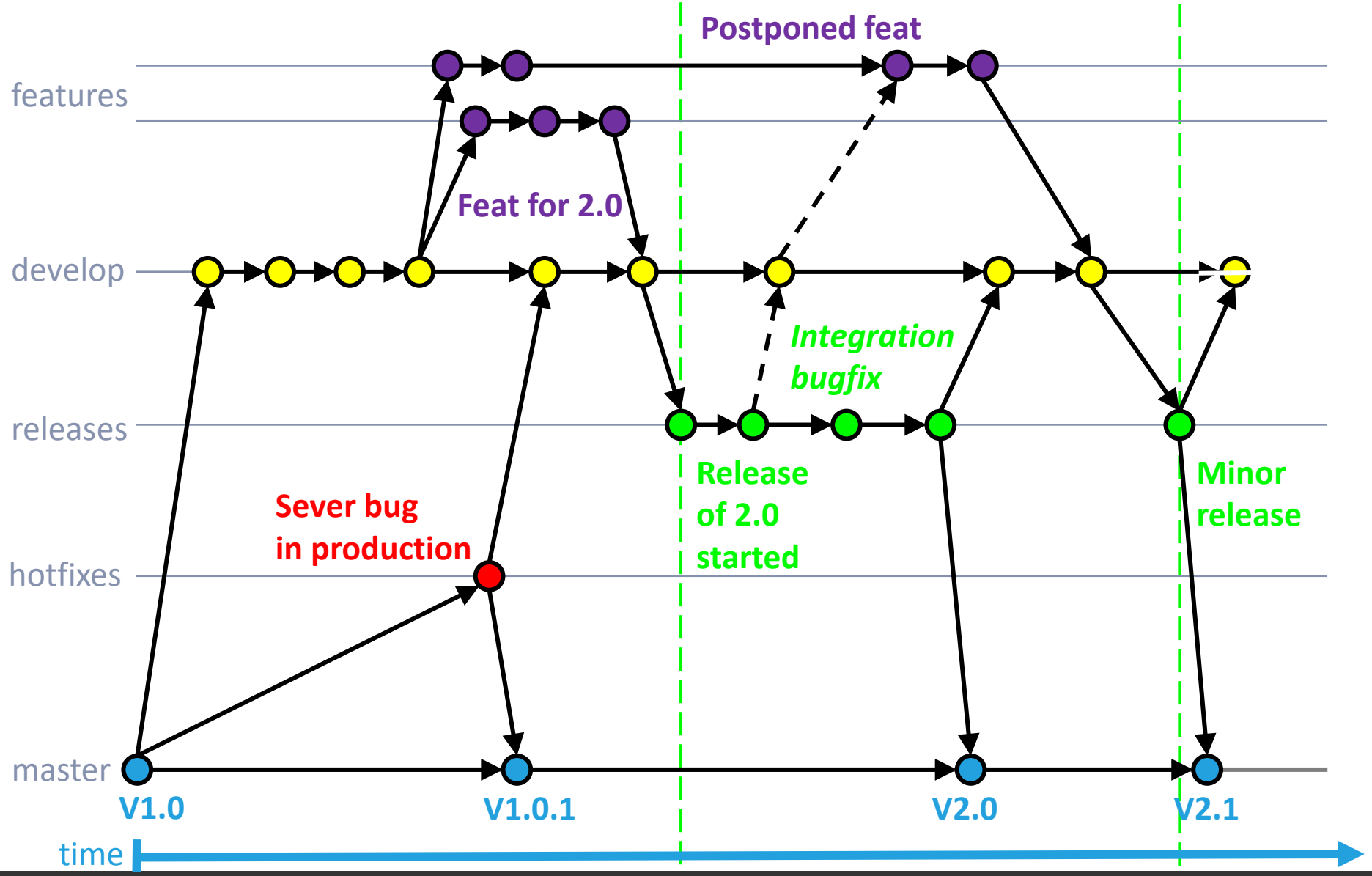
Git Flow

Author: Vincent Driessen
Original blog post:
http://nvie.com/archives/323



Git Flow

Author: Vincent Driessen
Original blog post:
http://nvie.com/archives/323



gitFlow

git

Additional Topics

git

Pull Requests

.gitignore

Issue Tracking

GUI Clients

Git: Pull Request

Feature available on GitHub and BitBucket

Useful to:

- Track feature changes
- Integrate new features (especially in open-source projects)

It helps to keep in one place all the discussion about the feature, and to avoid using several e-mails

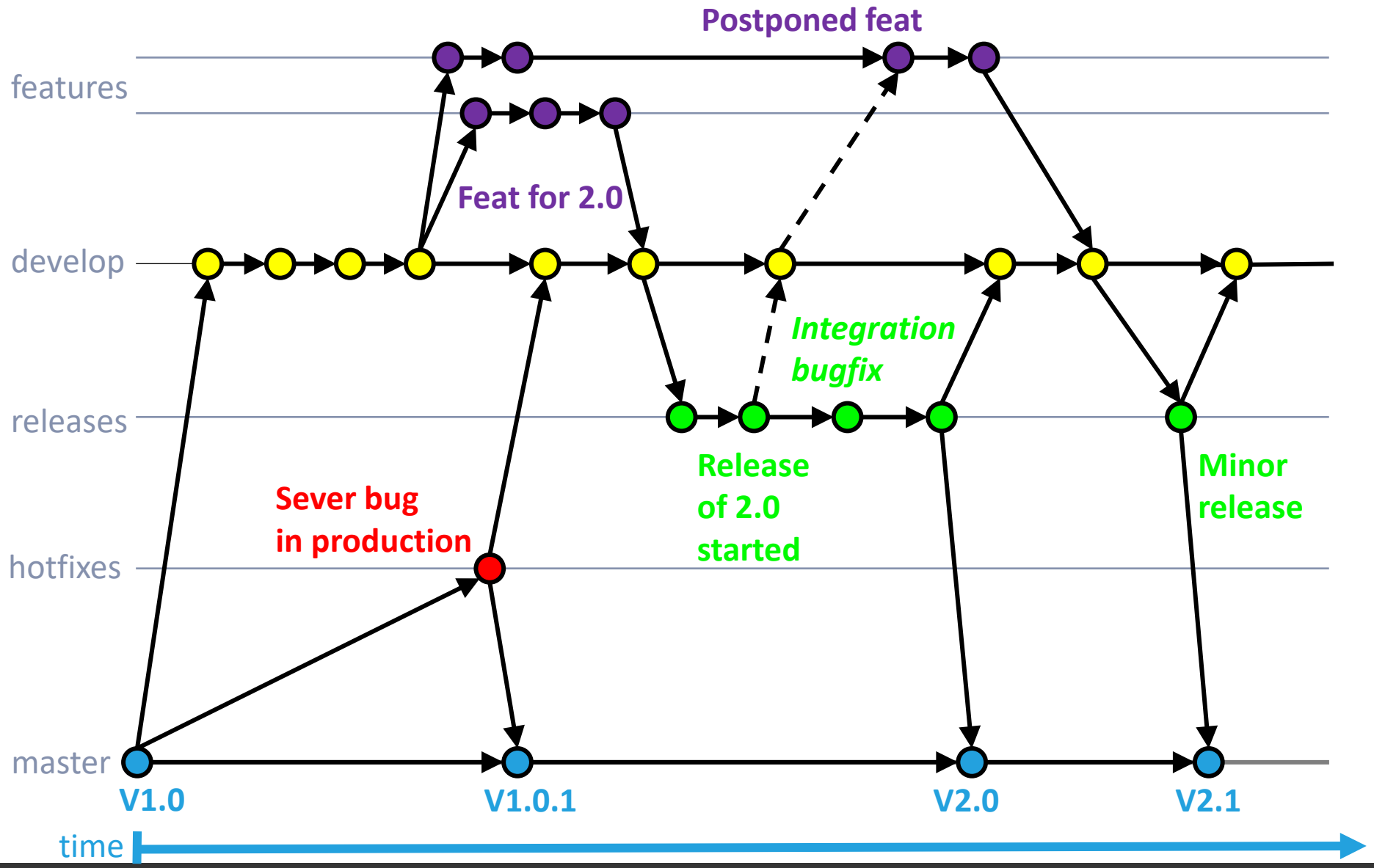
The first message is usually an user story, describing the changes

Then, the next messages ca be replies, code reviews, change requests, …

The pull request is finally *rejected* or *acceped*.

Git: Pull Request

Author: Vincent Driessen
Original blog post:
http://nvie.com/archives/323



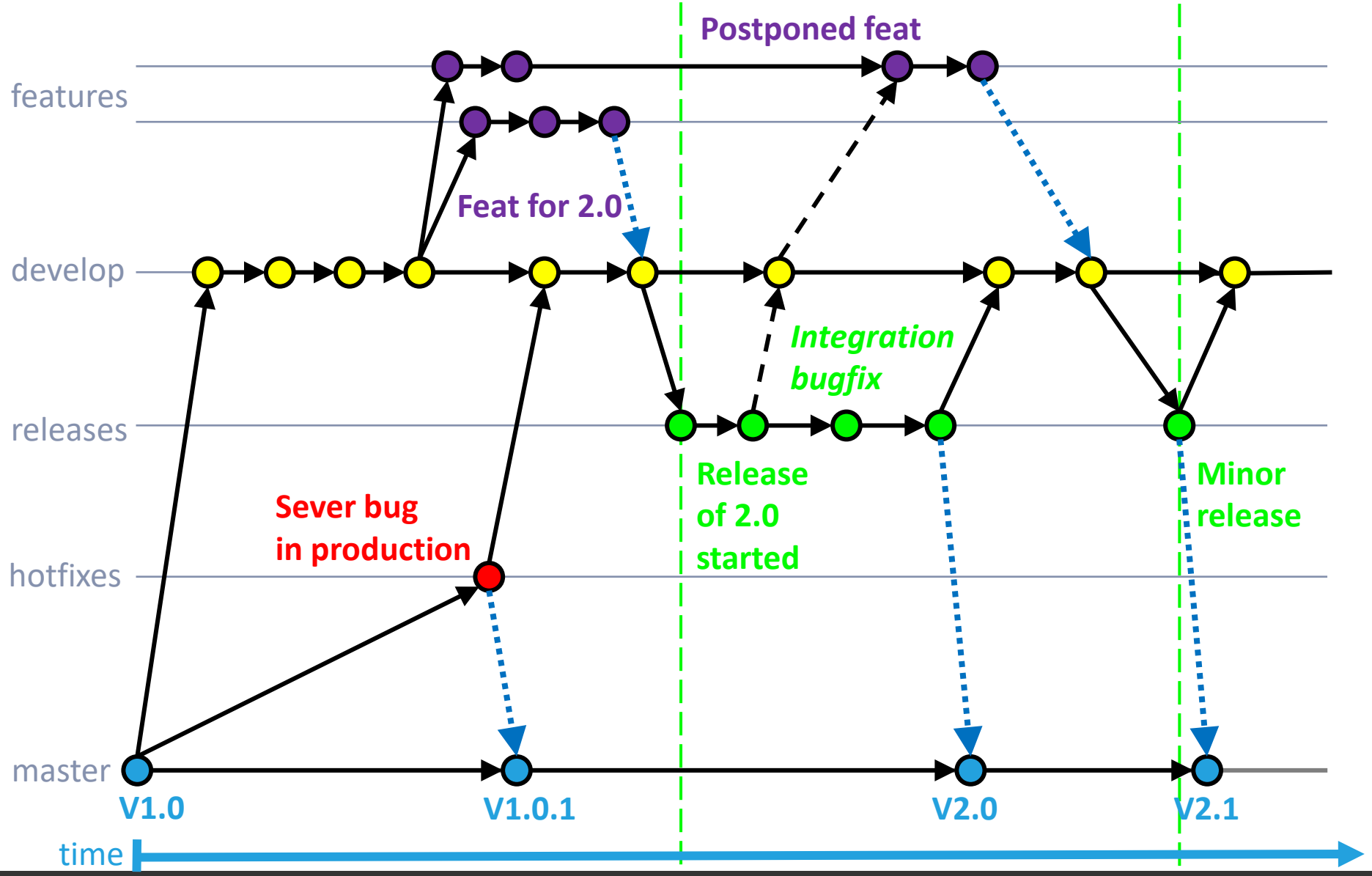
Git: Pull Request

Can you guess which are the pull requests in the flow?

(don’t look the solution in the next slide…)

Git: Pull Request

Author: Vincent Driessen
Original blog post:
http://nvie.com/archives/323



.gitignore

Is a text file telling which files or folders to ignore in a git project

A *.gitignore* file is usually placed in the root directory of a project

The entries in this file can also follow a matching pattern.

- * is used as a wildcard match
- / is used to ignore pathnames
- # is used to add comments

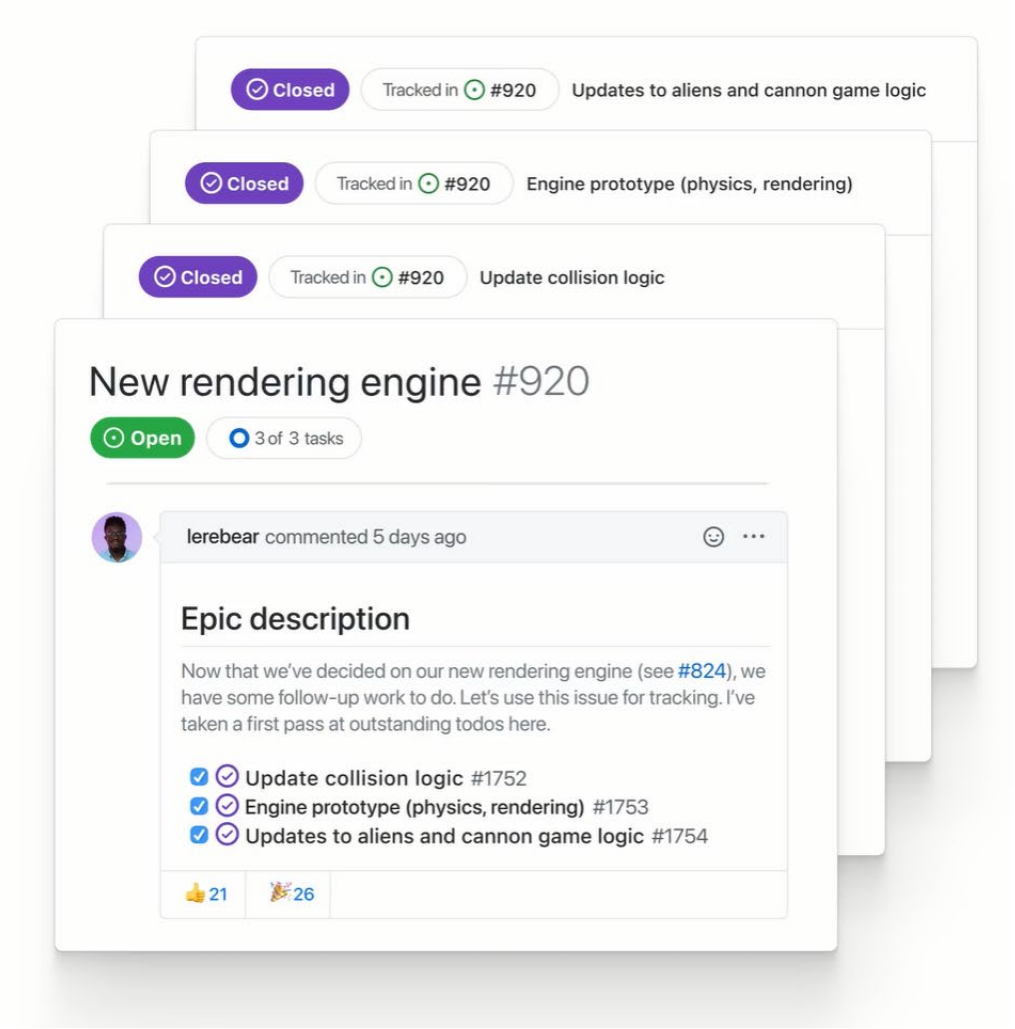
.gitignore example

```
# Ignore Mac system files
.DS_store

# Ignore .ipynb_checkpoints folder
.ipynb_checkpoints

# Ignore all text files
*.txt
```

Issue Tracking

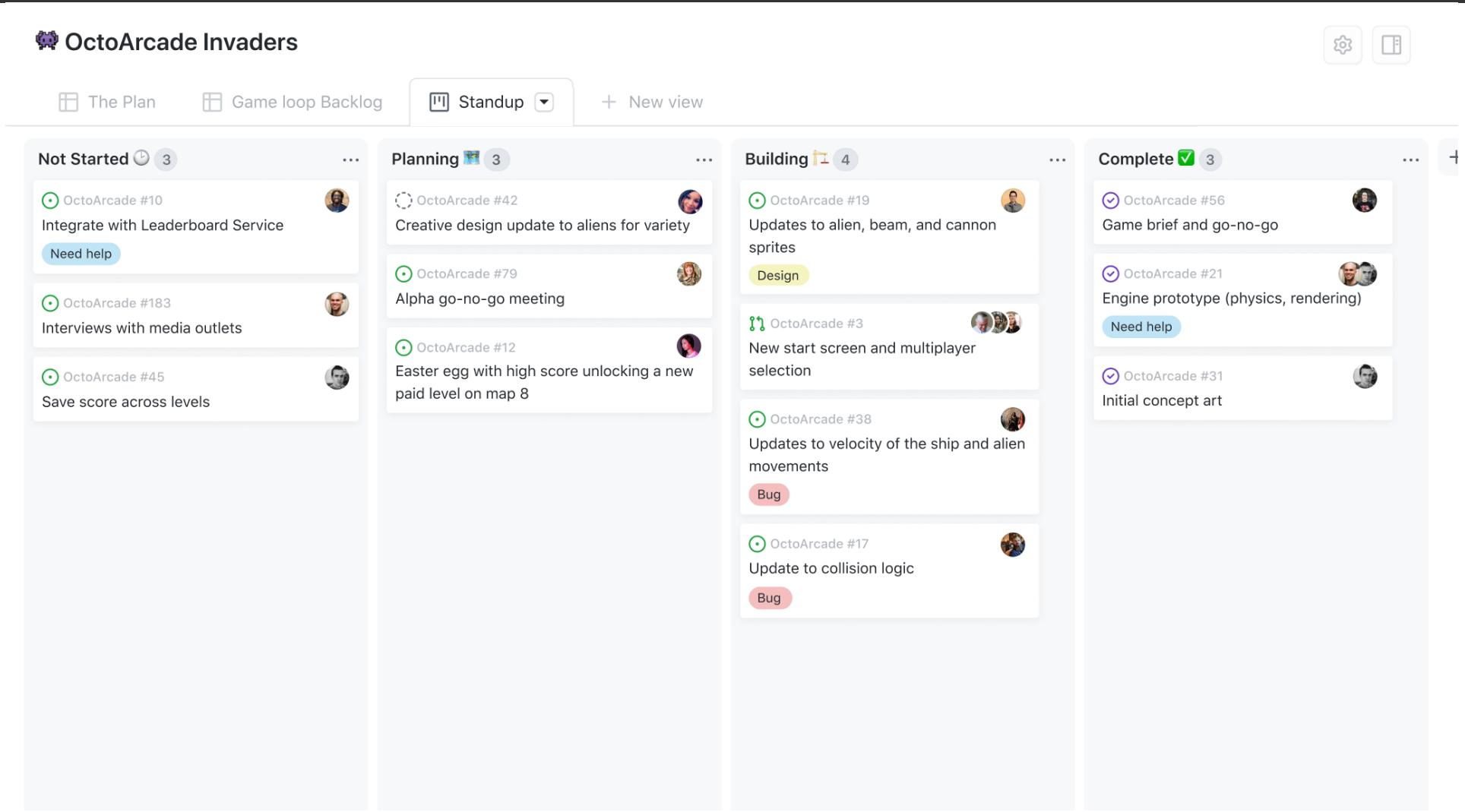
Break issues into actionable tasks

Tackle complex issues with task lists and track their status with new progress indicators. Convert tasks into their own issues and navigate your work hierarchy.



Source: [GitHub Issues · Project planning for developers]

Issue Tracking

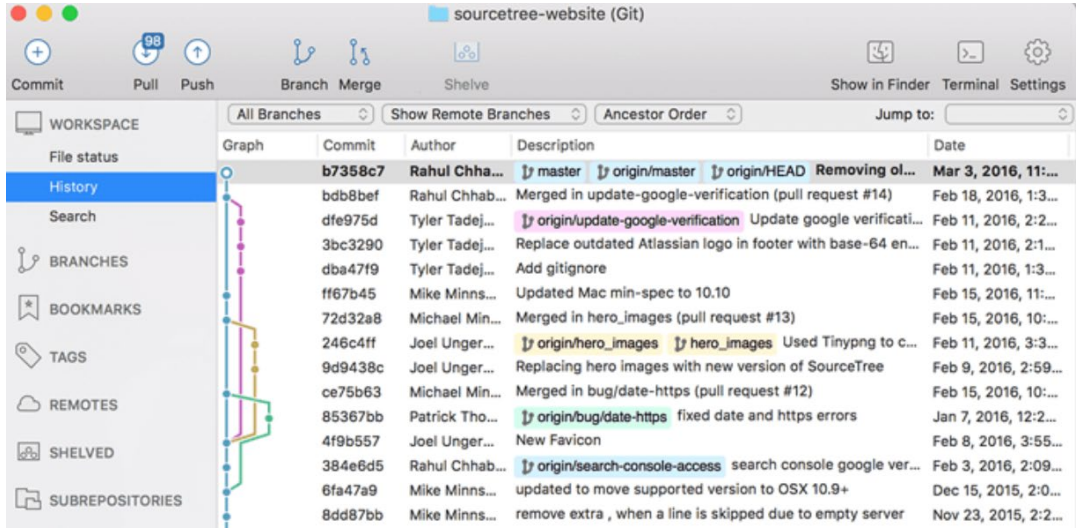


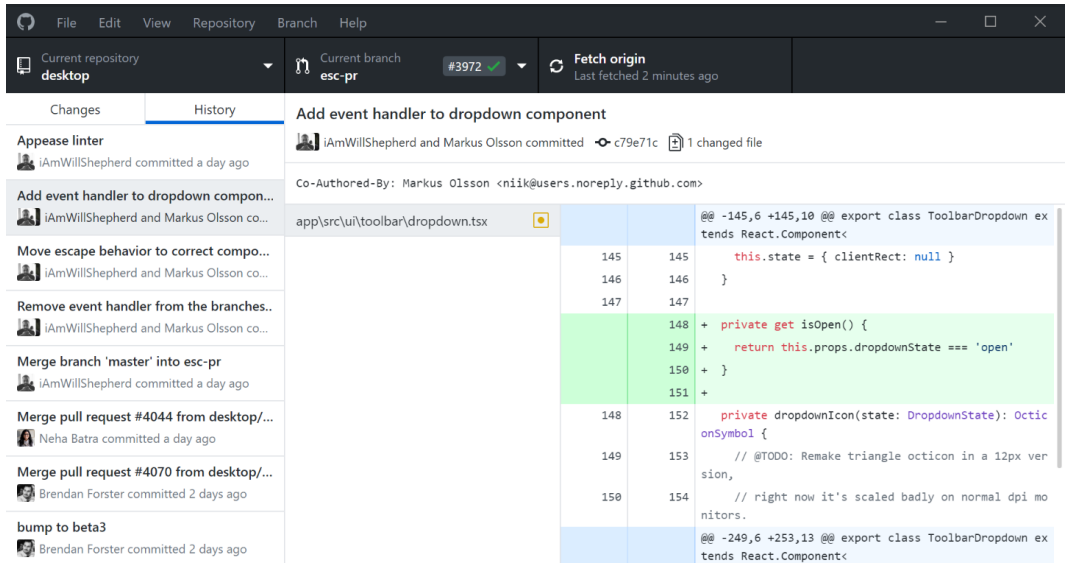
Source: [GitHub Issues · Project planning for developers]

GUI Clients

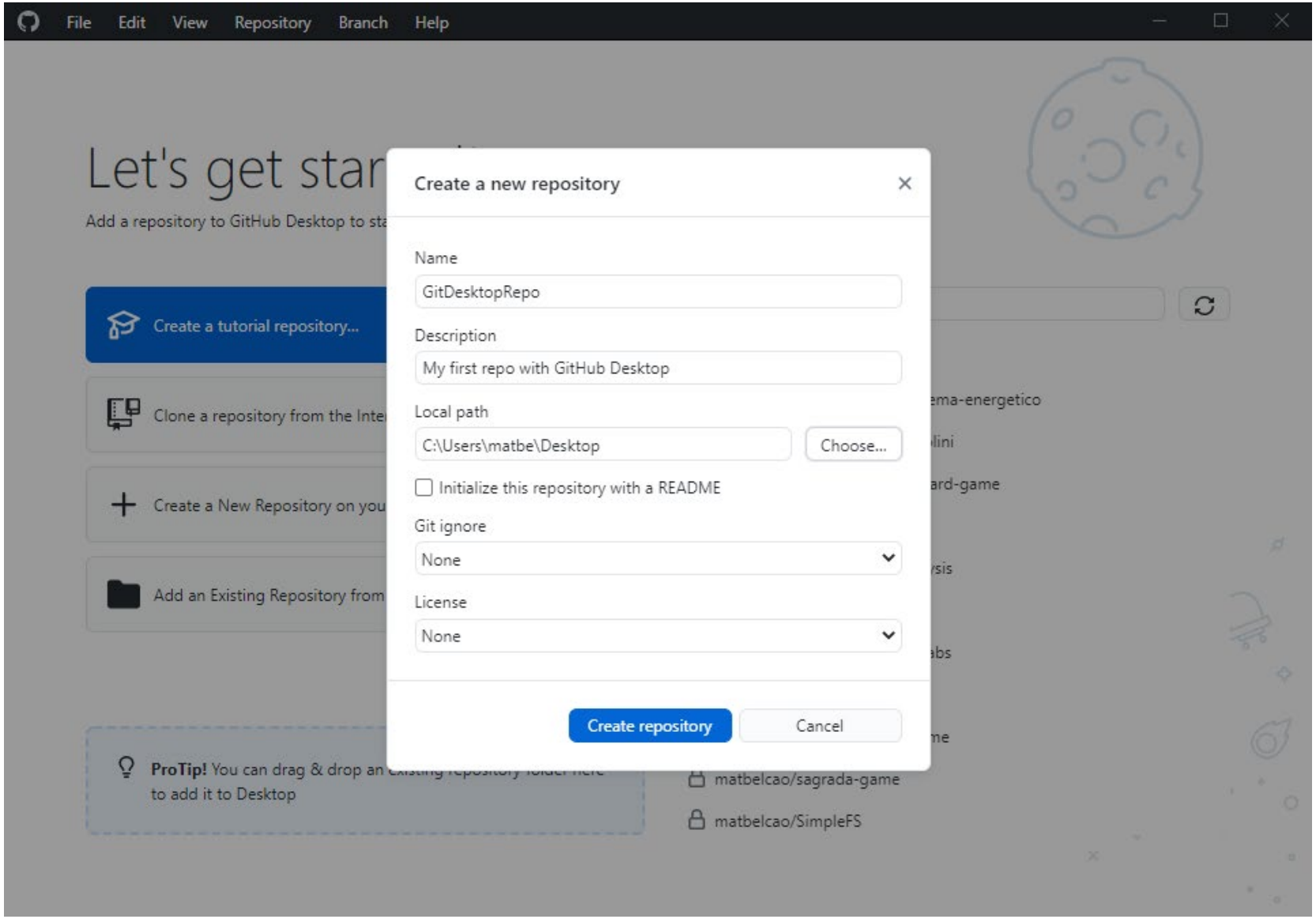
Several UI applications:

- Atlassian SourceTree
- GitHub Desktop / GitHub WebUI
- GitKraken
- other popular tools … here

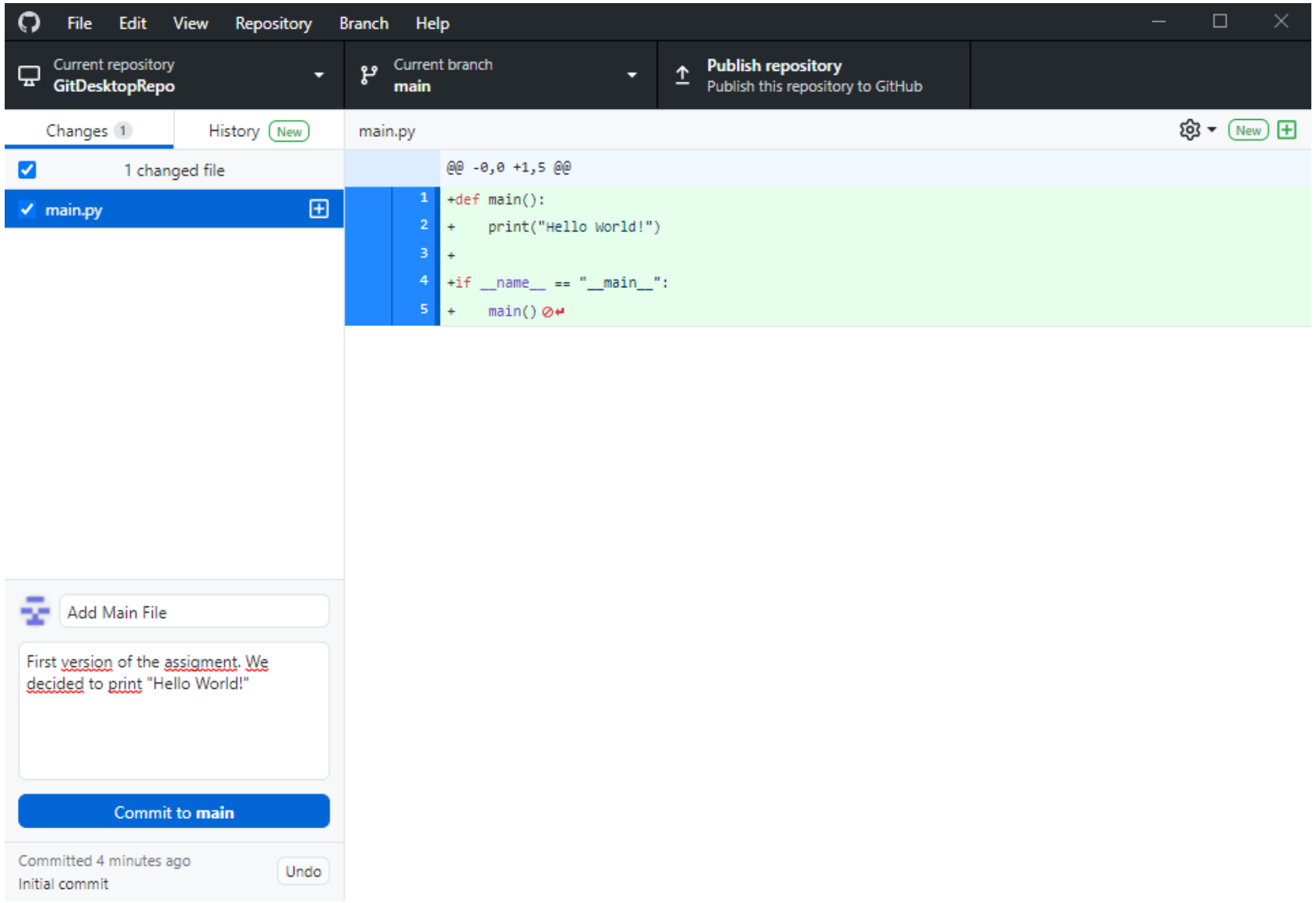




Demo: GitHub Desktop



Demo: GitHub Desktop



Demo: GitHub Desktop

File Edit View Repository Branch Help

Current repository
GitDesktopRepo

Current branch
main

Publish repository
Publish this repository to GitHub

Changes | History New

0 changed files

bugFix

New branch

Sorry, I can't find that branch
Do you want to create a new branch instead?

Create new branch

ProTip! Press Ctrl + Shift + N to quickly create a new branch from anywhere within the app

Choose a branch to merge into **main**

are some friendly suggestions for

machine. By publishing
ers.
Ctrl P

Publish repository

Open in Atom

Show in Explorer

Summary (required)

Description

Commit to **main**

Committed 1 minute ago
Add Main File

Undo

Demo: GitHub Desktop

File Edit View Repository Branch Help

Current repository
GitDesktopRepo

Current branch
bugFix

Publish repository
Publish this repository to GitHub

Changes 1 | History New

1 changed file

main.py

main.py New

```
@@ -1,5 +1,5 @@
 def main():
-    print("Hello World!")
+    print("Hello World! I'm testing the GitHub Desktop client!!!")

 if __name__ == "__main__":
     main()
```

Update main.py

Changed the printed message

Commit to bugFix

Committed 3 minutes ago
Add Main File

Undo

Demo: GitHub Desktop

File Edit View Repository Branch Help

Current repository
GitDesktopRepo

Current branch
main

Publish repository
Publish this repository to GitHub

Changes | History New

0 changed files

Merge into **main**

Filter

Default branch

main 5 minutes ago

Recent branches

bugFix 1 minute ago

This will merge **1 commit** from **bugFix** into **main**

Create a merge commit

y suggestions for

ishing

Publish repository

Open in Atom

Show in Explorer

Summary (required)

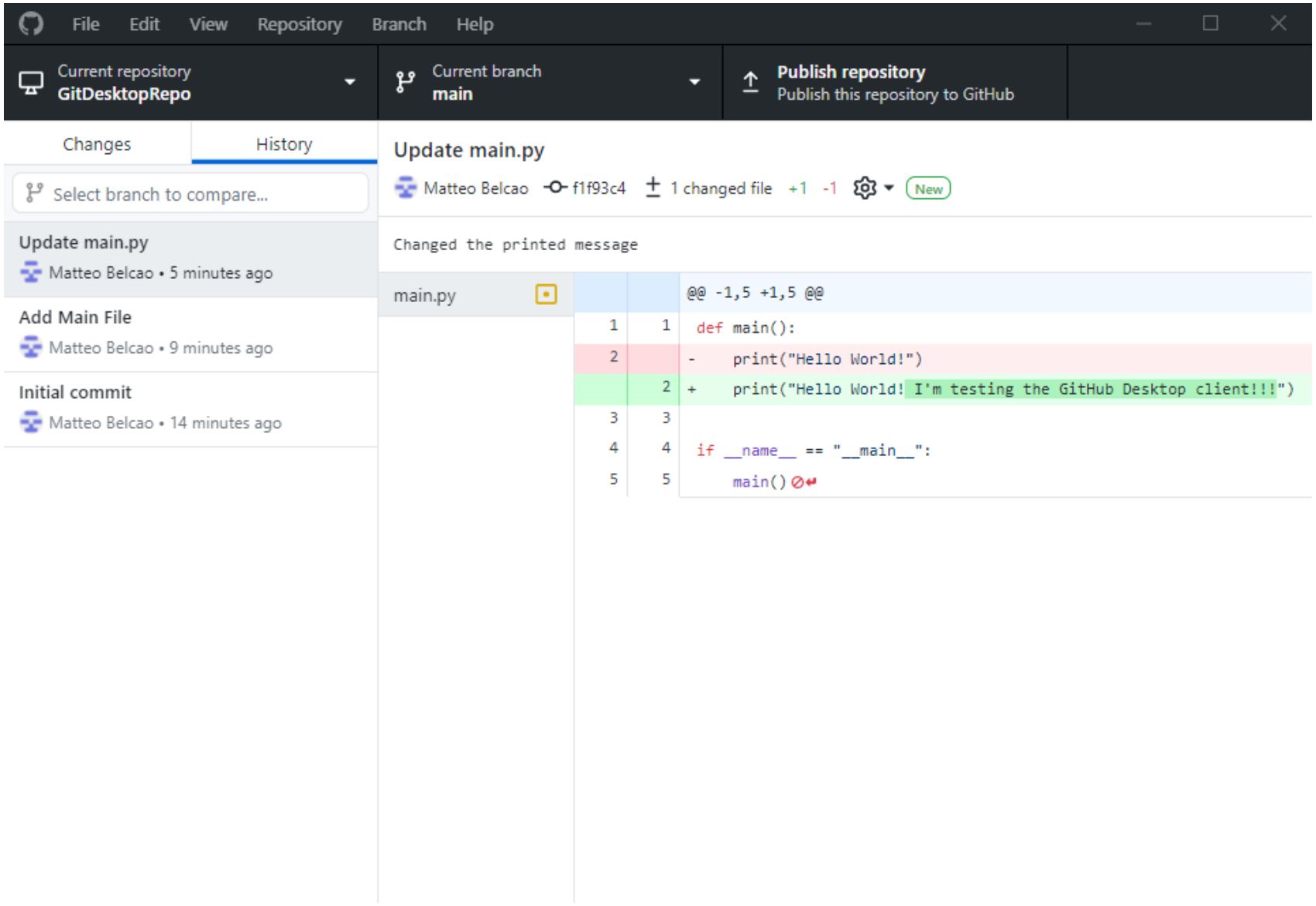
Description

Commit to **main**

Committed 5 minutes ago
Add Main File

Undo

Demo: GitHub Desktop



Advanced Topics

git

Documentation

submodules

history rewriting

Thank you !!