SETUP

```
# You should set these two:
git config --global user.name "Your Name"
git config --global user.email "you@example.com"

# Optional, but recommended:
git config --global color.ui "auto"

# Optionally, pick one editor to use:
git config --global core.editor "nano"         # Default, if you do nothing.
git config --global core.editor "gedit -w -s" # Recommended option.
git config --global core.editor "vim"
git config --global core.editor "emacs"
```

Setup up a GitHub account if you don't already have one:

www.github.com

DISCLAIMER:

I tend to talk really fast when I get excited, may use magical shortcuts and hotkeys without thinking about it, so if I do something you don't understand:

PLEASE tell me to "slow down and explain yourself!"

GIT AND GITHUB

INTRODUCTION TO SOURCE CONTROL

Michael Sergio

PROBLEM 1:

You have code that works.

Now you have to implement a new feature.

After changing a few things, nothing works anymore.

What do you do?

PROBLEM 2:

You're working on a project with someone.

You make some changes.

How do you give those changes back?

POSSIBLE SOLUTIONS TO PROBLEM 2.

Carrier Pidgeon (RFC 1149)

Dropbox

Sneakernet - USB

```
Never underestimate the bandwidth of a station wagon full
of tapes hurtling down the highway.
                                    - Andrew S. Tanenbaum
```

How do you put the data back together so that it works?

BETTER TOOLS - A HISTORY

1974 Douglas McIlroy writes diff for Unix 5th Edition.

```
--- /path/to/original''timestamp''
+++ /path/to/new''timestamp''
@@ -1,3 +1,9 @@
+This is an important +notice! It should therefore be located at
+the beginning of this document!
+
 This part of the document has stayed the same from version to
@@ -5,16 +11,10 @@
 be shown if it doesn't change.  Otherwise, that would not be helping to
-compress the size of the changes.
-
-This paragraph contains
-text that is outdated.
-It will be deleted in the
-near future.
+compress anything.
```

1985 Larry Wall writes patch for sending diffs through networks.

HISTORY

- 1972 **SCSS** - Source Code Control System - *Bell Labs*
- 1982 **RCS** - Revision Control System
- 1986 **CVS** - Concurrent Versions System
- 2000 **SVN** - Subversion
- 2002 **DCVS** - Distributed Concurrent Versions System
- 2005 **Bazaar**
- 2005 **Mercurial**
- 2005 **Git**

PROBLEM 3

You're in charge of a project used by millions of users.

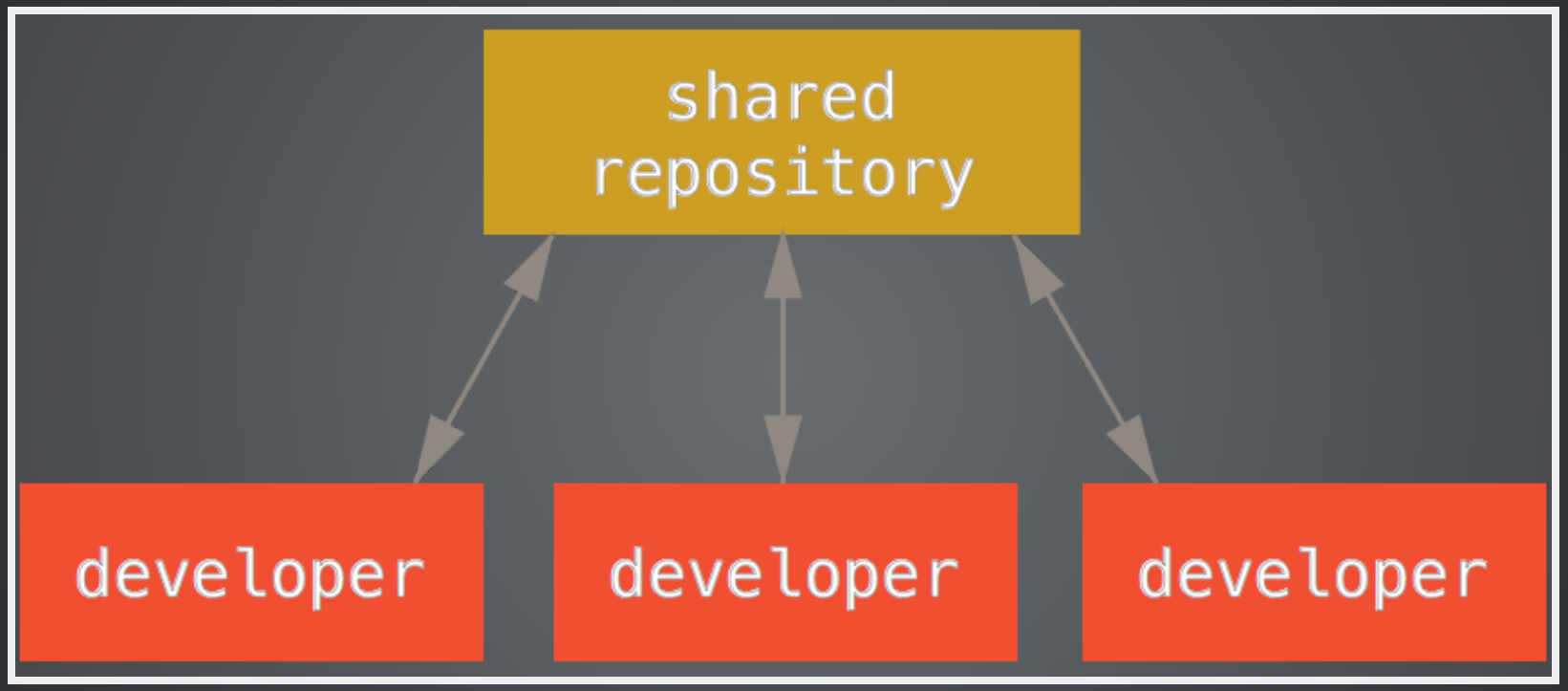
You receive thousands of patches per release.

You don't trust everyone.

You're Finnish.

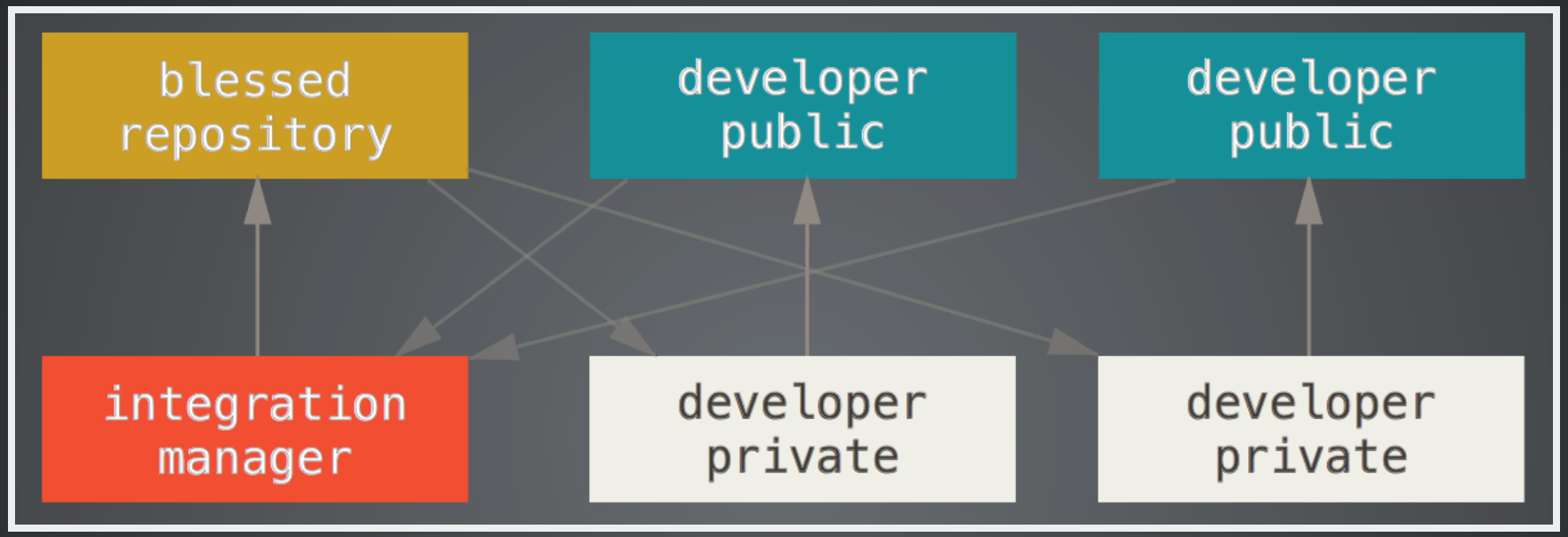
WHAT DOESN'T WORK

Centralized Workflow



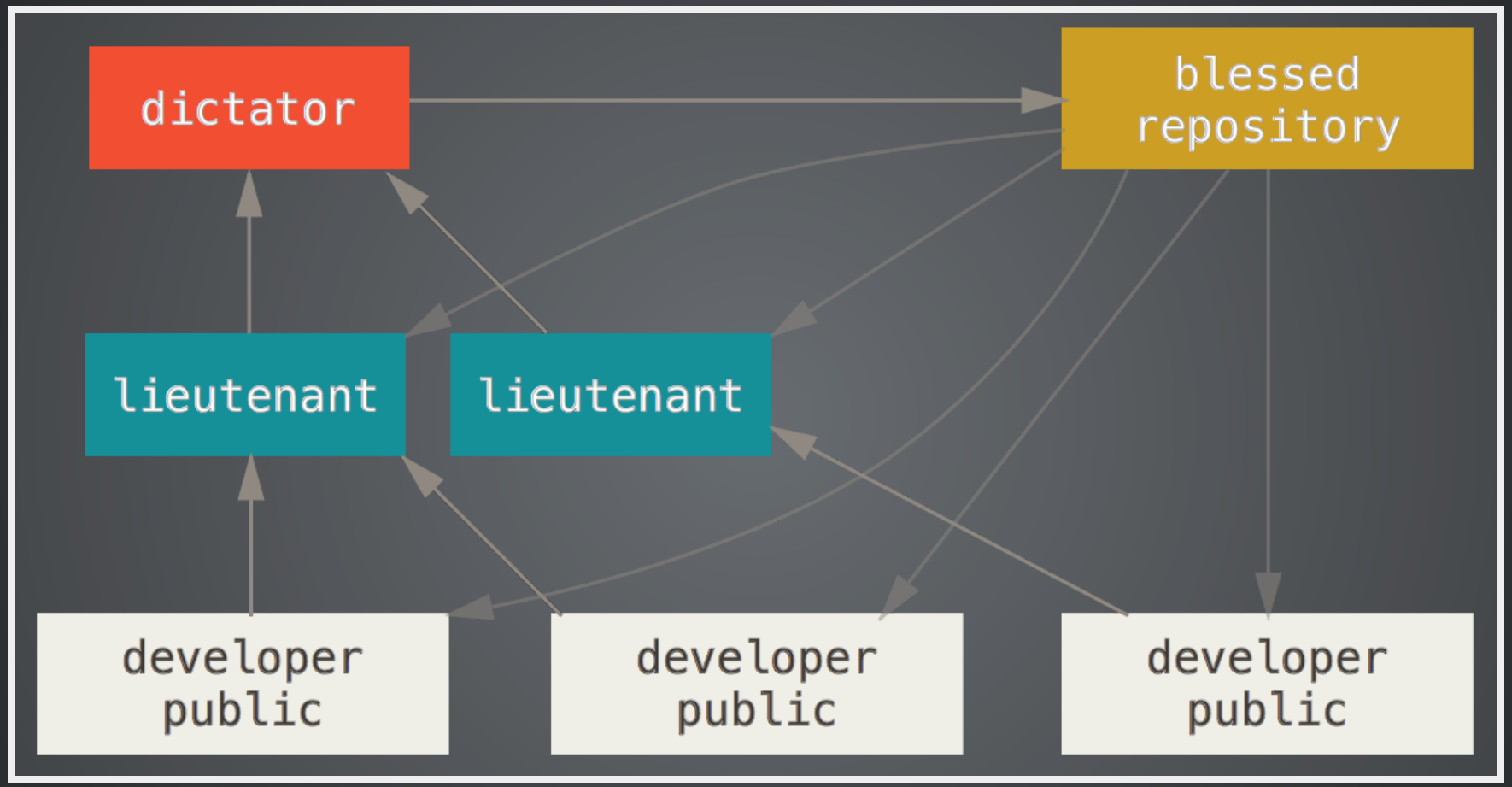
Push and **Pull** from Central Repository

DISTRIBUTED WORKFLOW



1. Developer **pulls** a fork from the Blessed Repo.
2. Developer **commits** changes to private repo.
3. Developer **pushes** to their own public repo.
4. Integration Manager **pulls** from each developers public repo.
5. When well tested, manager **pushes** back to blessed repo.

LINUX WORKFLOW



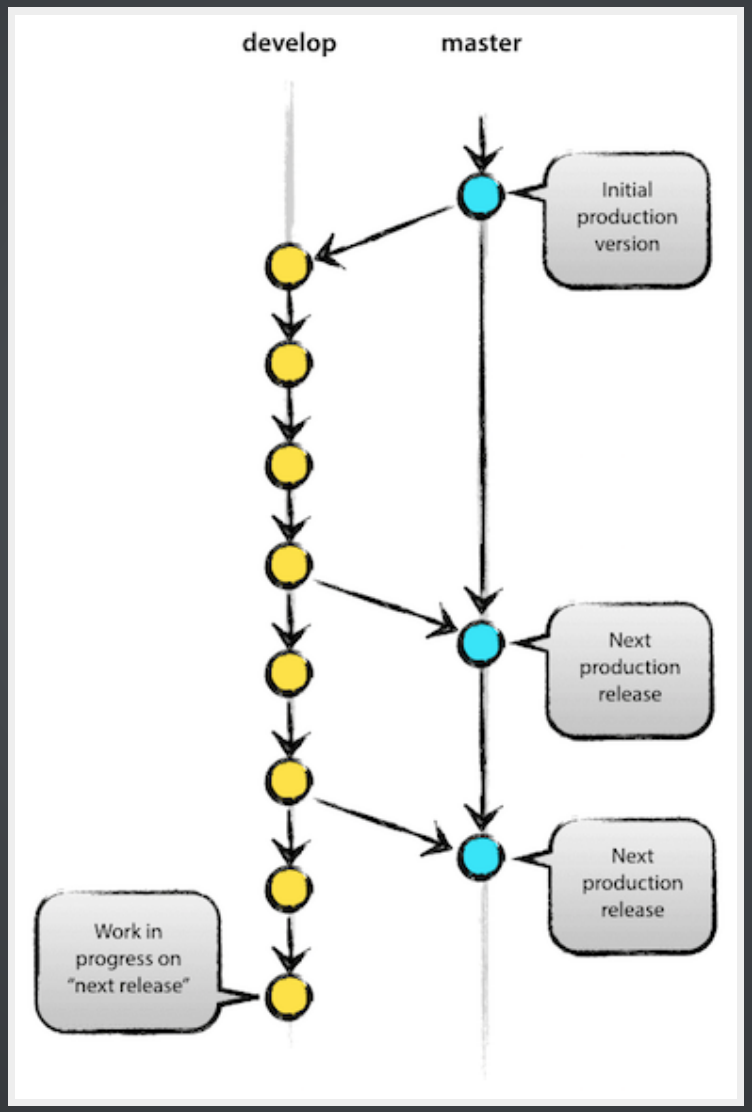
For the Linux Kernel, different sub-systems have their own maintainers: networking, CPU, power, all the various drivers, etc...

WHY USE A DISTRIBUTED SYSTEM?

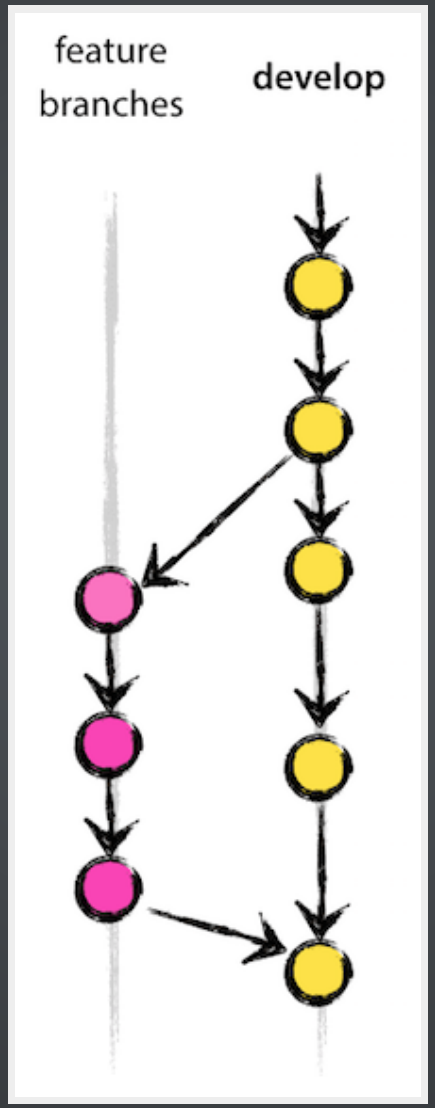
Must get **forking** and **merging** right

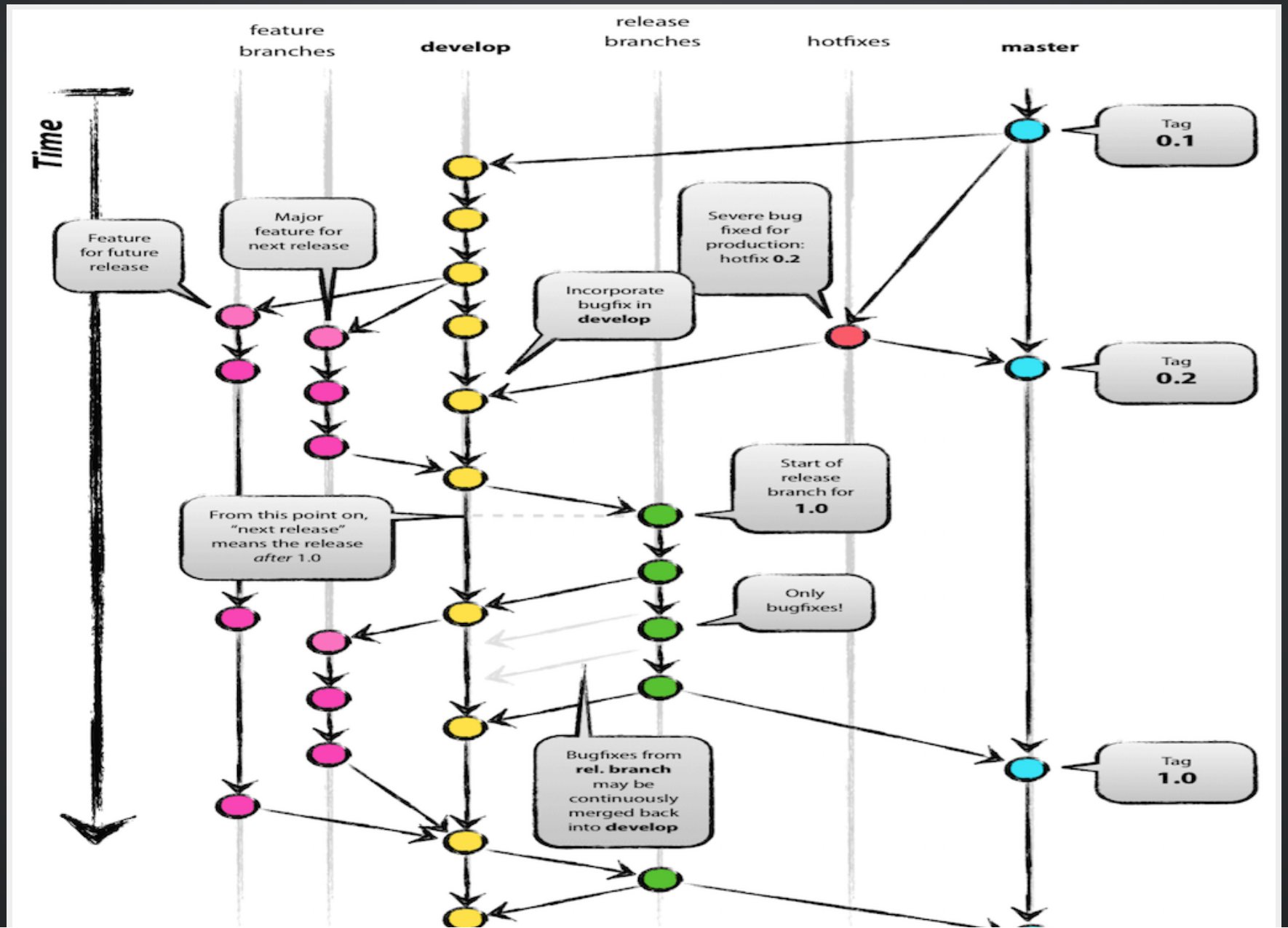
Otherwise everything falls apart.

DEVELOPMENT WORKFLOW



DEVELOPMENT WORKFLOW




feature branches
develop
release branches
hotfixes
master
Time
Feature for future release
Major feature for next release
Severe bug fixed for production: hotfix 0.2
Incorporate bugfix in develop
Tag 0.1
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 merged back into develop
Tag 1.0

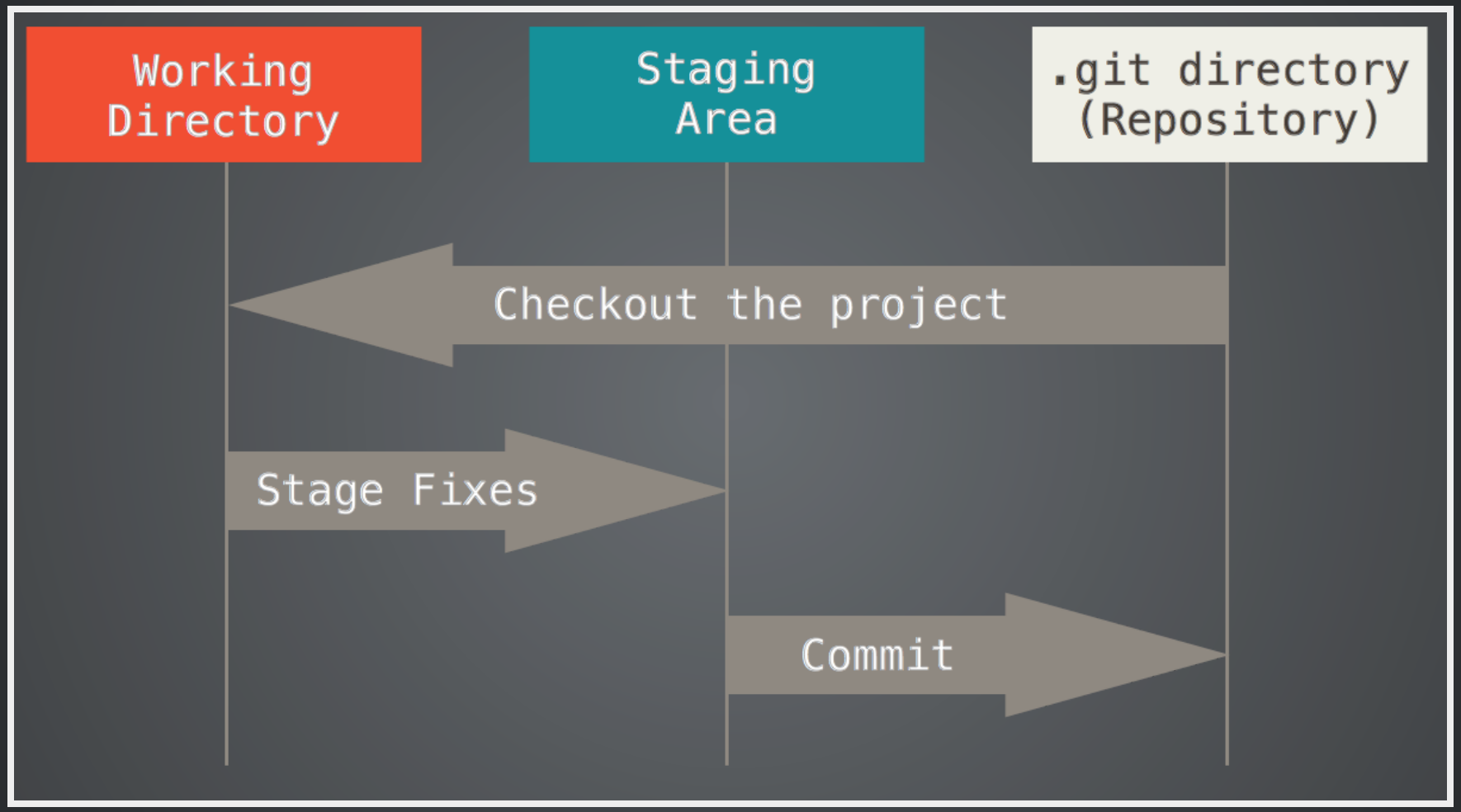
GOT IT?
GOOD.
LETS DO IT!

HTTPS://GITHUB.COM/MICHAELSERGIO/PURRFECT

PULL REQUEST

https://github.com/blog/1124-how-we-use-pull-requests-to-build-github

STAGING


Working Directory
Staging Area
.git directory (Repository)
Checkout the project
Stage Fixes
Commit

GIT ETTIQUITE

- If itâs not in source control, it doesn't exist!
- Commit early, commit often!
- Always look at your changes before committing them.
- Compilation output does not belong in source control. Use .gitignore files!
- Remember the axe-murderer when writing commit messages.

APPLICATIONS OF GITHUB

- Package Mangers
- Continious Integration
- Discover new projects and tools.
- Click Explore on the GitHub main page for more.
- https://github.com/explore

BEYOND CODE

- Supports Viewing of Maps
- Supports 3D models
- Books
- German Law - bundestag
- https://github.com/showcases

CONTINUOUS INTEGRATION

You should not ever edit code directly on a server!

Push code to a remote service.

Have it tested and deployed automatically.

Etsy deploys more than 50 times a day.

GIT IS A TOOL

PROFESSIONALS INVEST IN THEIR TOOLS!

Learn your Unix tools, your Text Editor, your Source Control, your scripting languages.

Don't be afraid!

"Civilised tool for a civilized age" - some guy on the internet

WHY DO WE USE VERSION CONTROL?

To protect us from ourselves.

APPENDIX

CONTRIBUTING ETTIQUE

- Pay attention to style guides used.
- Always include a README.md!
- Your README should always include how to build and run your project.

SOFTWARE LICENSES

TLDR Legal

WHY IS GIT SO FAST?

IT'S A DAG

Directed Acyclic Graph

QUESTIONS?

REFERENCES

- Coming from SVN
- Git is a DAG
- 20 GitTips

TUTORIALS

- Stanford Git Tutorial
- Git Architecture
- Learn Git Branching
- Fundamentals
- Walkthorugh

BOOKS

- Pragmatic Guide to Git - Good Intro
- Git SCM - Online, Free, Up-to-date
- Pragamtic Programmer - Classic for every developer

LINKS

History of SCM

GIVEAWAY!