"Git" setup for this week

1. Check that you have Git installed:

   ```
   $ git --version
   ```

2. Sign up for an account on GitHub.com
3. Send your GitHub username to Oli on Slack
4. Check you have keys setup:

   ```
   $ ssh git@github.com
   ```

   "Hi [username]! You've successfully authenticated, but GitHub does not provide shell access."

5. If not, follow directions on next slide

DevelopMe_

Key generation

1. Create an SSH key pair (unless you have already):
   $ **ssh-keygen -t rsa -C "email@domain.com"** (hit enter at prompts)
2. To output your public key:
   $ **cat ~/.ssh/id_rsa.pub**
3. Then add key to your GitHub account in https://github.com/settings/ssh
4. Test your key with:
   $ **ssh git@github.com**
5. Full instructions are here

DevelopMe_

Make sure your identity is set

So that Git knows who your commits are made by:

1. `$ git config --global --edit`
2. Edit values as required, uncomment (remove #)
3. Then, to save and exit, `:wq`
4. If you'd like a more familiar editor for Git documents (and you have the **subl** CLI tool set up), run:

   `$ git config --global core.editor "subl -n -w"`

DevelopMe_

DevelopMe_

Coding Fellowship

Week 5:
Tooling: Git & GitHub

What is Git?

A tool (software) that allows us to:

- keep track of how our code changes over time
- collaborate with others in a controllable, ordered manner

DevelopMe_

Module Outline

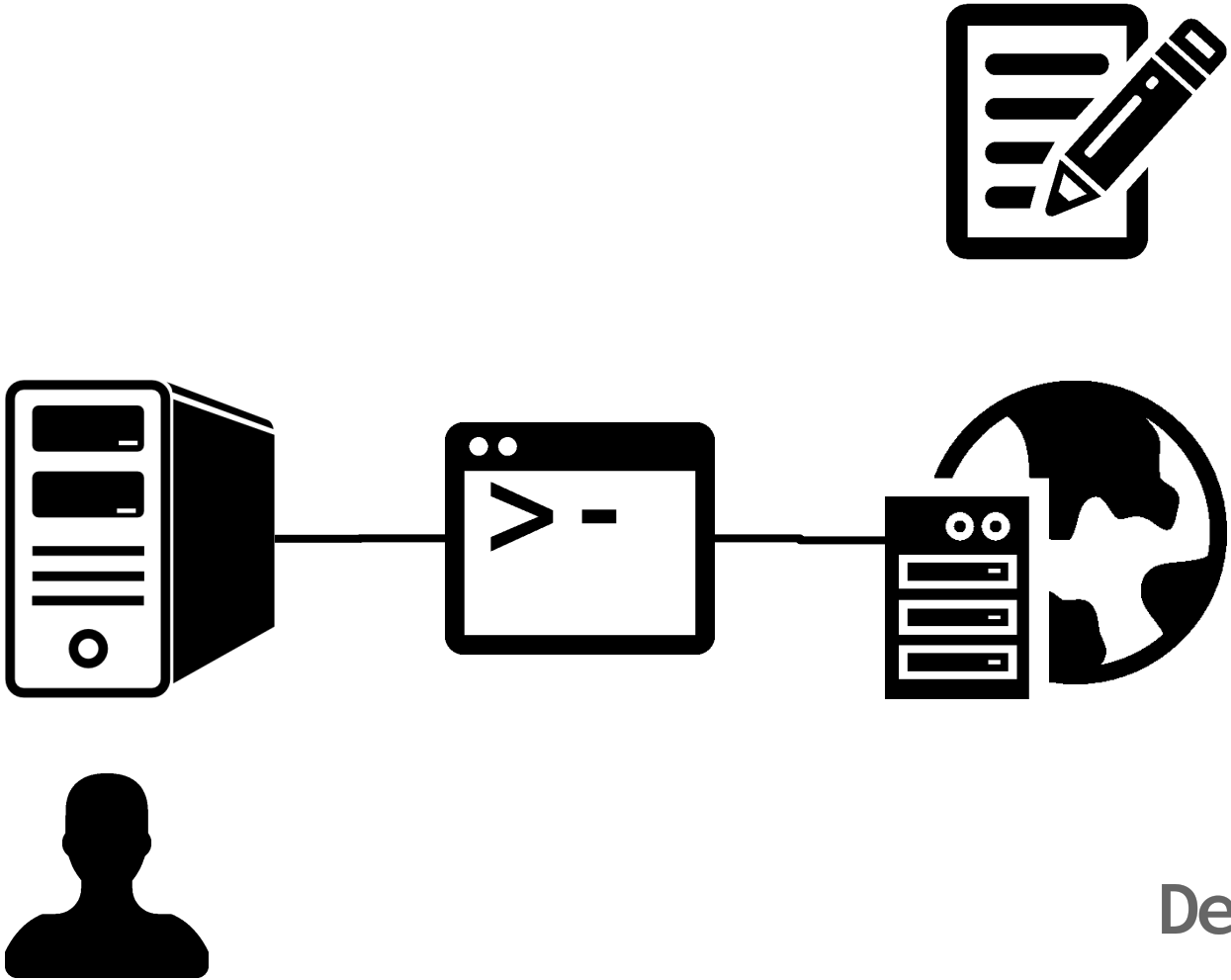
- Introduction to version management
- Git architecture and how it works
- Basic Git commands
- Git in the real world, giving it a go!
- Best practice for use
- Git workflows and project strategy
- Workflows applied

DevelopMe_

Why version manage?

DevelopMe_

Editing files directly on server



DevelopMe_

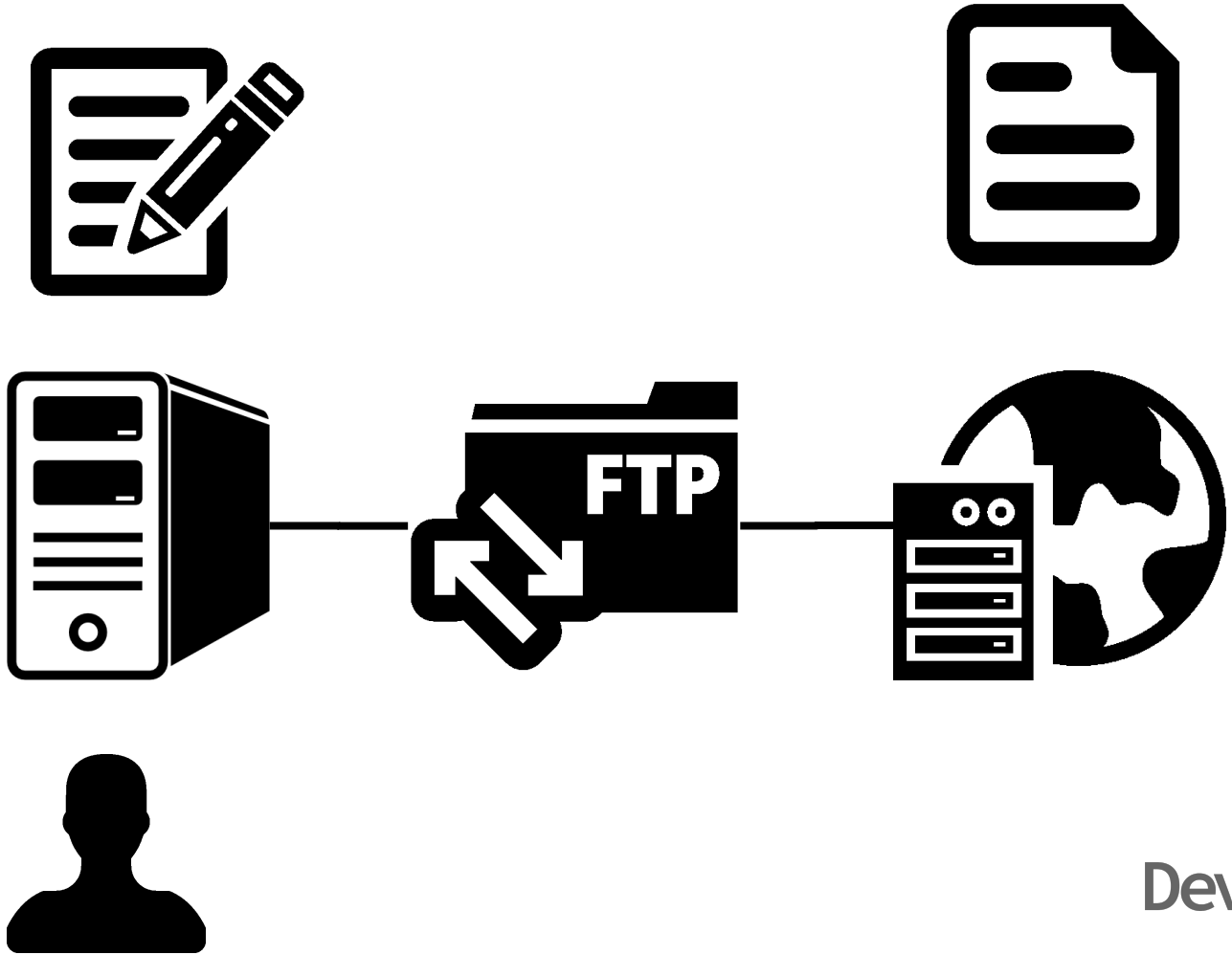
Working directly on server drawbacks

- no ability to automatically rollback changes
- requires you to manage keeping backups and different versions of files
- high risk!
- working with multiple contributors is tricky

DevelopMe_

Working with FTP

DevelopMe_

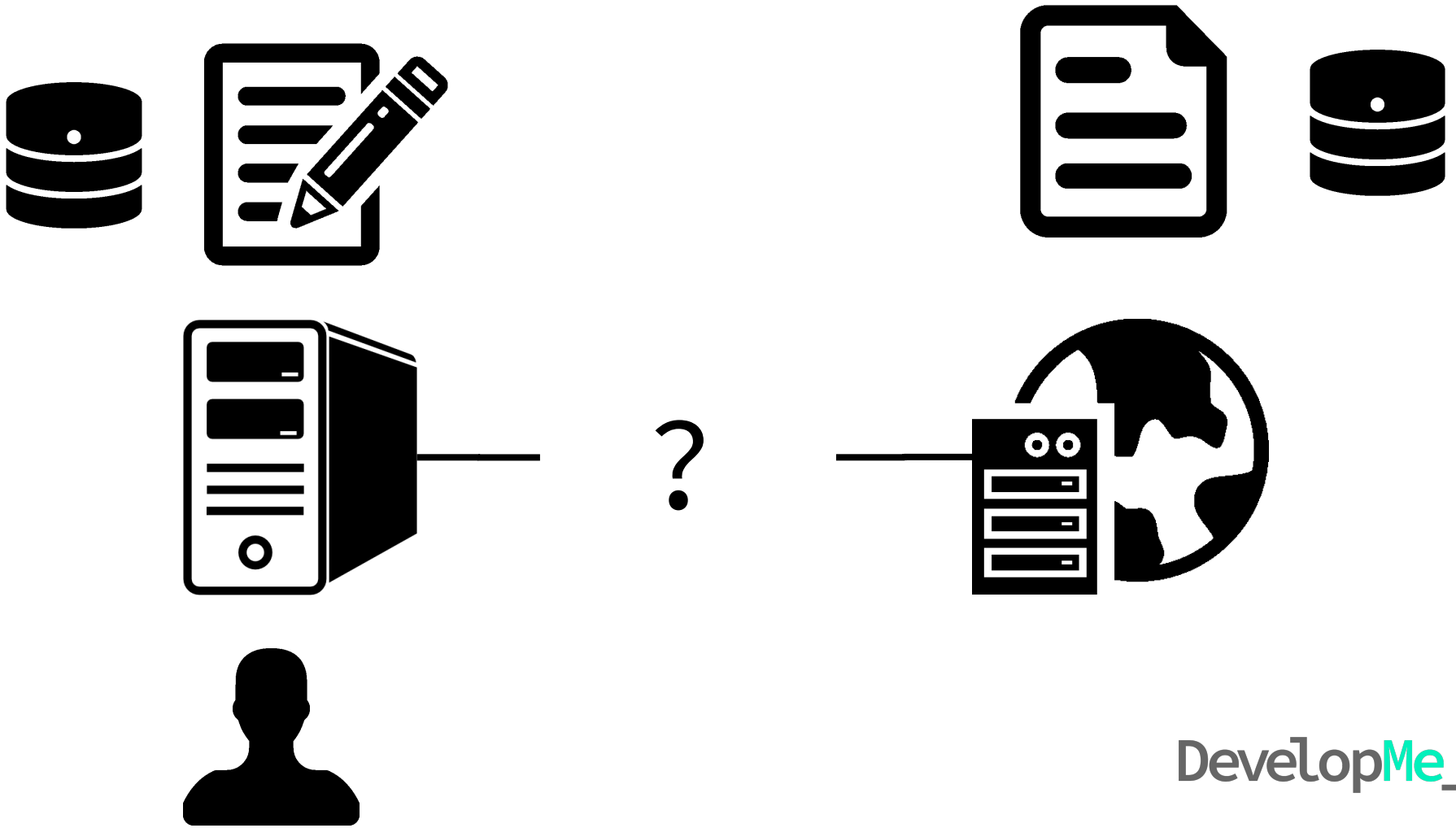


DevelopMe_

Working with FTP drawbacks

- no ability to automatically rollback changes
- requires you to manage keeping backups and different versions of files
- manually keep track of which files have changed and need to be uploaded
- slow, uploading and downloading each time
- working with multiple contributors gets messy quickly

Working with version management


?
DevelopMe_

Working with version management

- each version of files in the project has a snapshot, and can be reverted to at any point
- automatically keeps track of which files have changed and need to be uploaded
- changes have an audit log, who made what change when

Version control: the basics

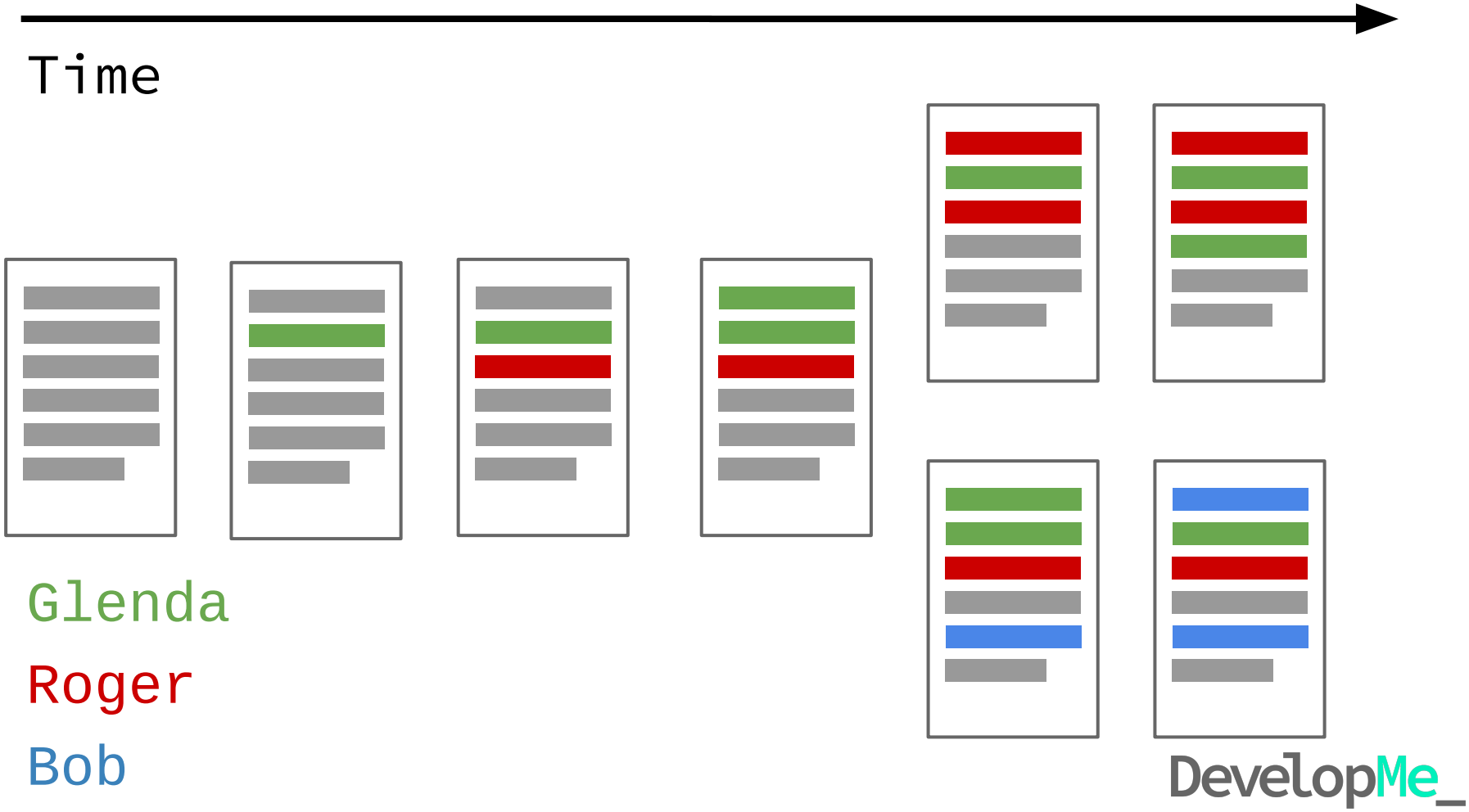
DevelopMe_

Example: text document

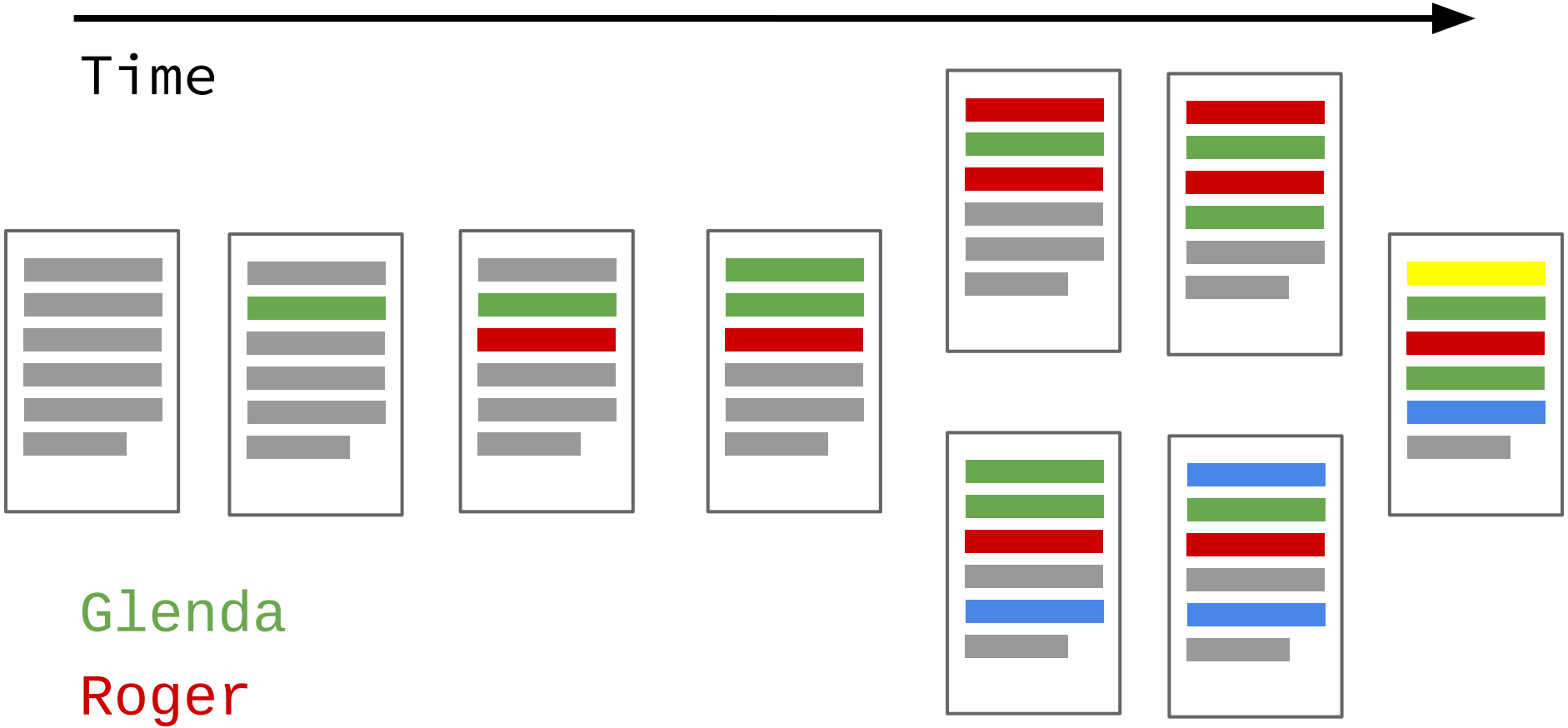
Multiple people editing the document.

Great, as long as people edit the same document sequentially.

If two people start editing different copies then someone has to manually combine the changes, what happens if the changes overlap?


Time
Glenda
Roger
Bob
DevelopMe_

DevelopMe_


Time
Glenda
Roger
Bob
DevelopMe_

Why use git?

DevelopMe_

Why Git?

- most widely used source code management tool
- 33.3% of professional software developers reporting use Git or GitHub as their primary source control system
- excellent tooling support, supporting existing workflows and editors

DevelopMe_

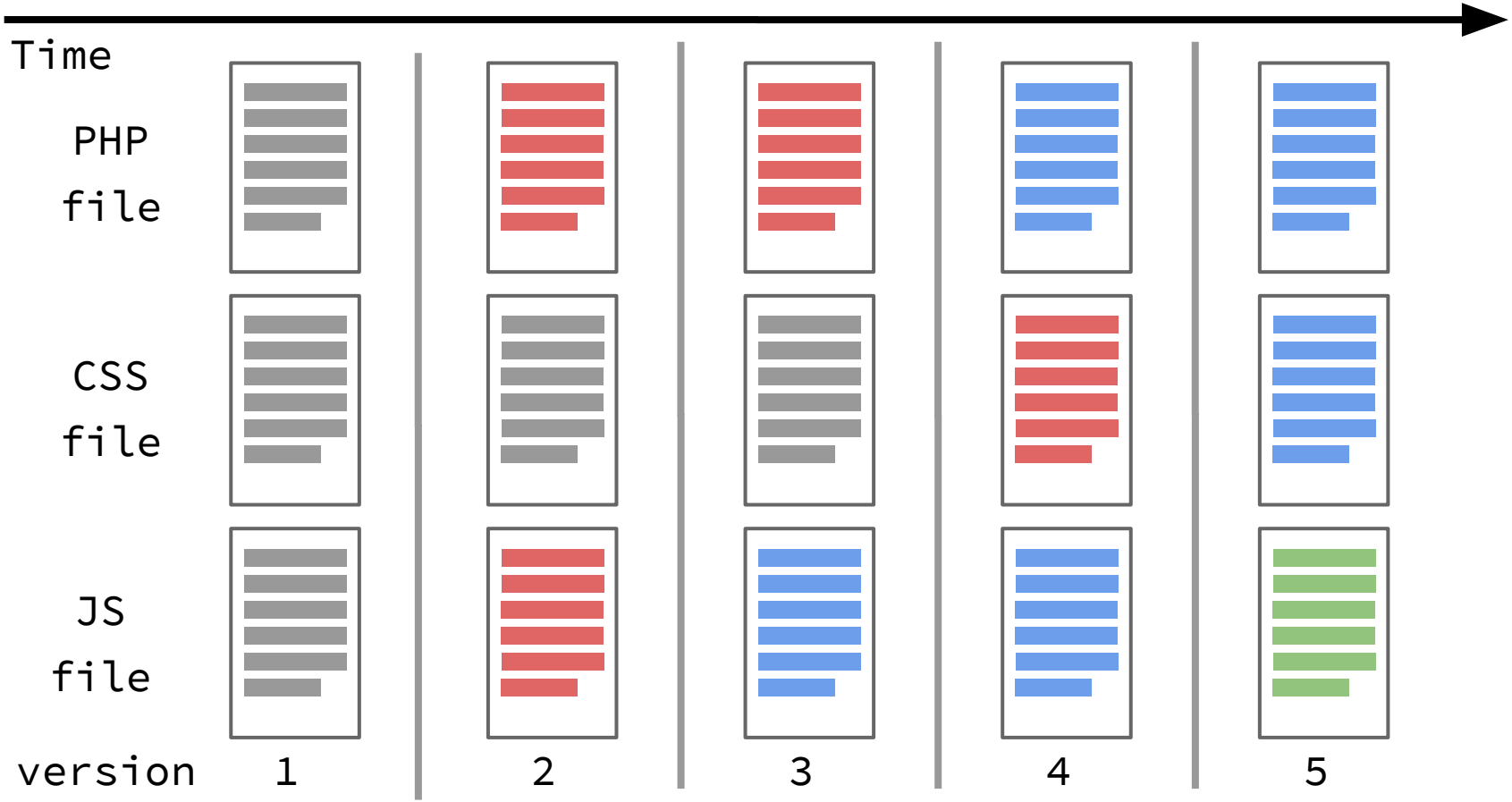
Git philosophy

DevelopMe_

Storing changes

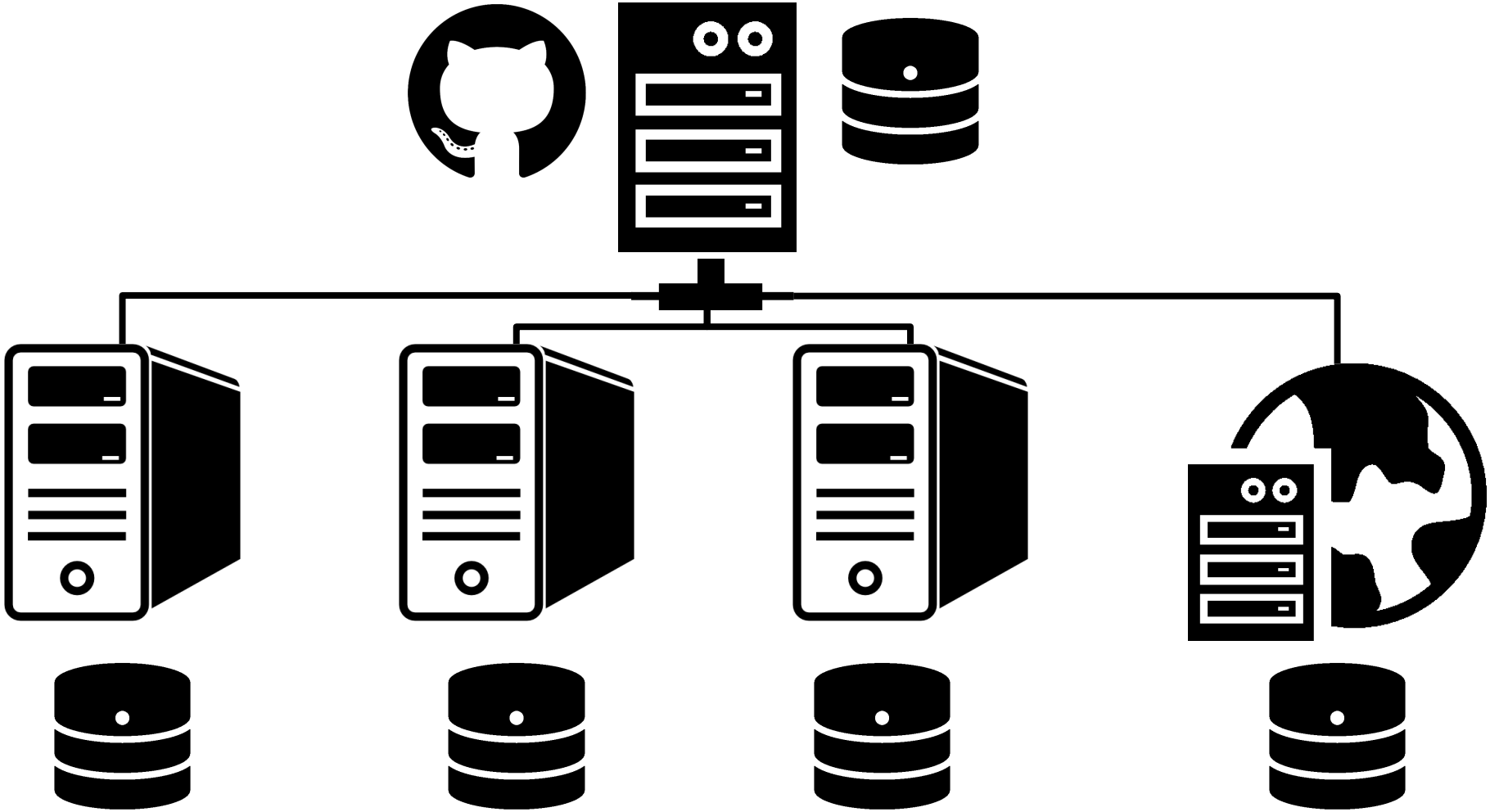
Other VCS

Git: snapshots, not differences



Git architecture

DevelopMe_



Git behaviour

DevelopMe_

(most) operations are local

- rolling back to early snapshot
- creating a branch
- creating a new snapshot
- merging branches

Reduces requests over network, increases speed+efficiency, allows working offline

DevelopMe_

Git has integrity

- Everything is check-summed before storage
- Snapshots are referred to by that checksum
- Impossible to change the contents of any file or directory without Git knowing about it
- Checksumming is SHA-1 hash, producing 40-character string e.g.: 24b9da6552252987aa493b52f8696cd6d3b00373

develop-me / git

Unwatch 1 Star 0 Fork 0

initial commit of Bootstrap site files

master

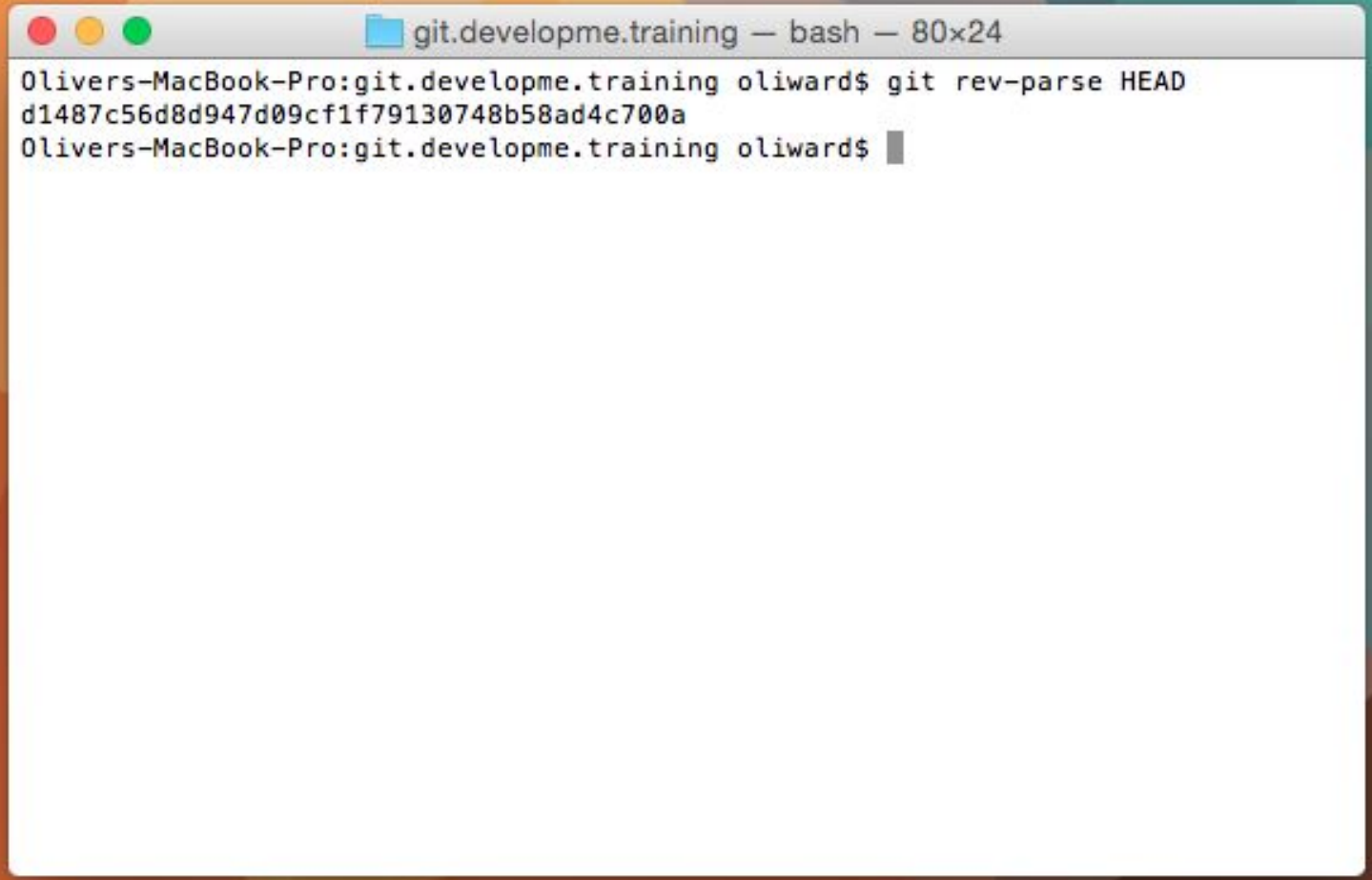
Browse code

oliward authored 21 hours ago 0 parents commit 13e901f9bc68d2ca78b159ad86d1f9996a29d406

Showing 20 changed files with 9,378 additions and 0 deletions. Unified Split

18 README.md

```
@@ -0,0 +1,18 @@
+# Develop Me Git Exercise
+
+This repository has been set up for the Git Workshop attendees to have a go working on and committing work to a Git repository.
+
+It is a simple [Bootstrap](http://getbootstrap.com/) webpage that has had the [Carousel](http://getbootstrap.com/examples/carousel/) theme applied.
+
+## Structure
+/css holds all of the stylesheets
+/fonts holds the fonts
+/js holds the Javascript
```

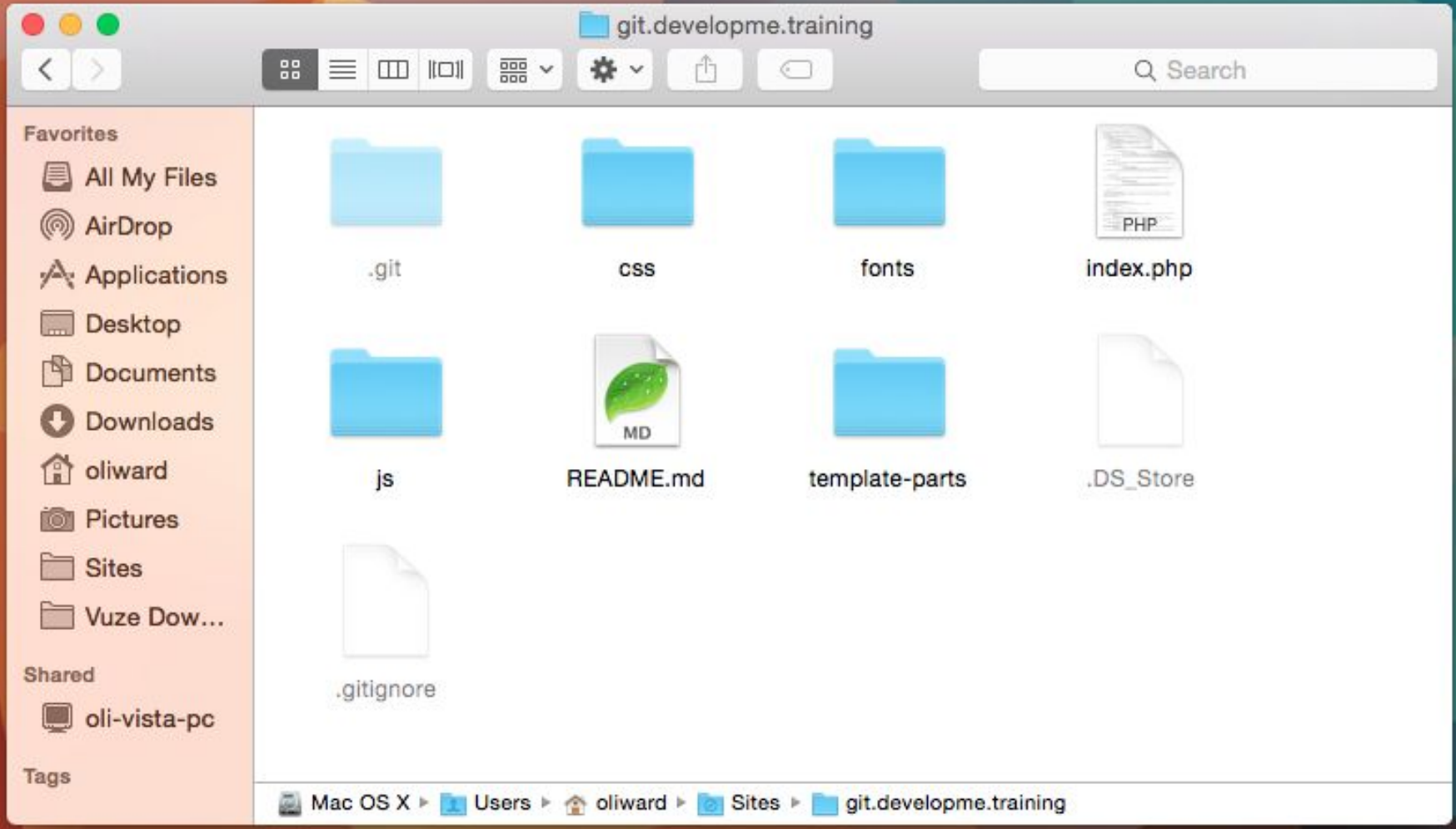

git.developme.training — bash — 80×24
Olivers-MacBook-Pro:git.developme.training oliward$ git rev-parse HEAD
d1487c56d8d947d09cf1f79130748b58ad4c700a
Olivers-MacBook-Pro:git.developme.training oliward$

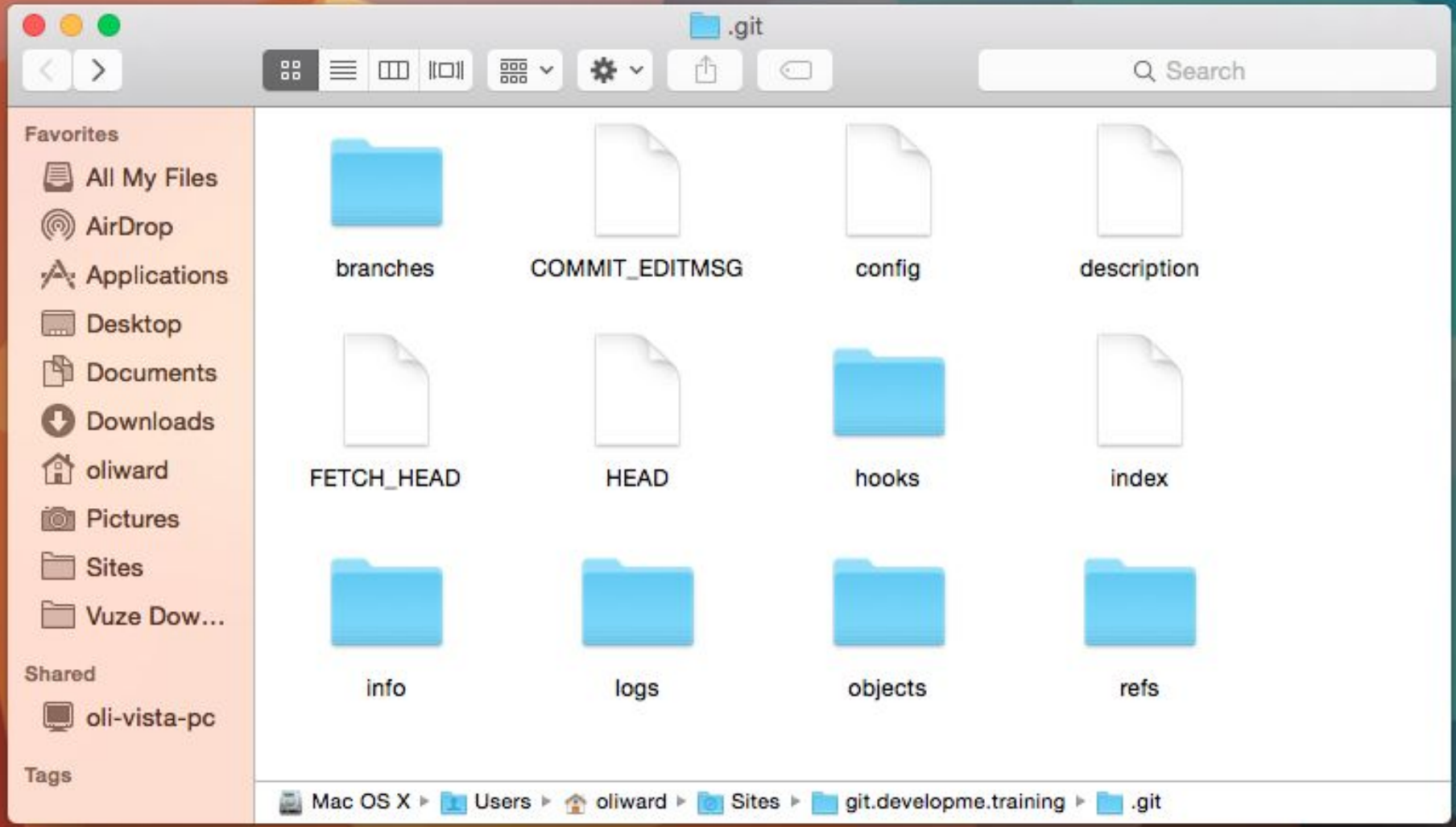
Git only adds data

- Actions in Git (nearly) only add data to the Git database
- It is hard to get the system to do anything that is not undoable or to make it erase data in any way
- Only way you can lose or mess up changes is if you haven’t pushed your work yet
- Great safety net for trying things out, and rolling back
- After you commit a snapshot it is very difficult to lose data, especially if you regularly push to a remote repository

DevelopMe_

Git database


git.developme.training
Search
Favorites
All My Files
AirDrop
Applications
Desktop
Documents
Downloads
oliward
Pictures
Sites
Vuze Dow...
Shared
oli-vista-pc
Tags
.git
css
fonts
PHP
index.php
js
MD
README.md
template-parts
.DS_Store
.gitignore
Mac OS X
Users
oliward
Sites
git.developme.training


.git
Search
Favorites
All My Files
AirDrop
Applications
Desktop
Documents
Downloads
oliward
Pictures
Sites
Vuze Dow...
Shared
oli-vista-pc
Tags
branches
COMMIT_EDITMSG
config
description
FETCH_HEAD
HEAD
hooks
index
info
logs
objects
refs
Mac OS X ▸ Users ▸ oliward ▸ Sites ▸ git.developme.training ▸ .git

Git repo hosts

DevelopMe_

search *“git repo hosting”*

Git: the basics

DevelopMe_

The language

repository / repo

project files and a versioning database, in our example this is hosted on GitHub, but can be hosted on any Git server or your local machine

fetch

fetching file versions and information from central repository server, e.g. GitHub

checkout

switch your project directory to a certain version of the project, replaces version managed files with the versions from this point in time

commit

create a point in time version of the current state of the project files

push

push your snapshots (work), to the central project repository, to allow other people to pull and checkout your changes

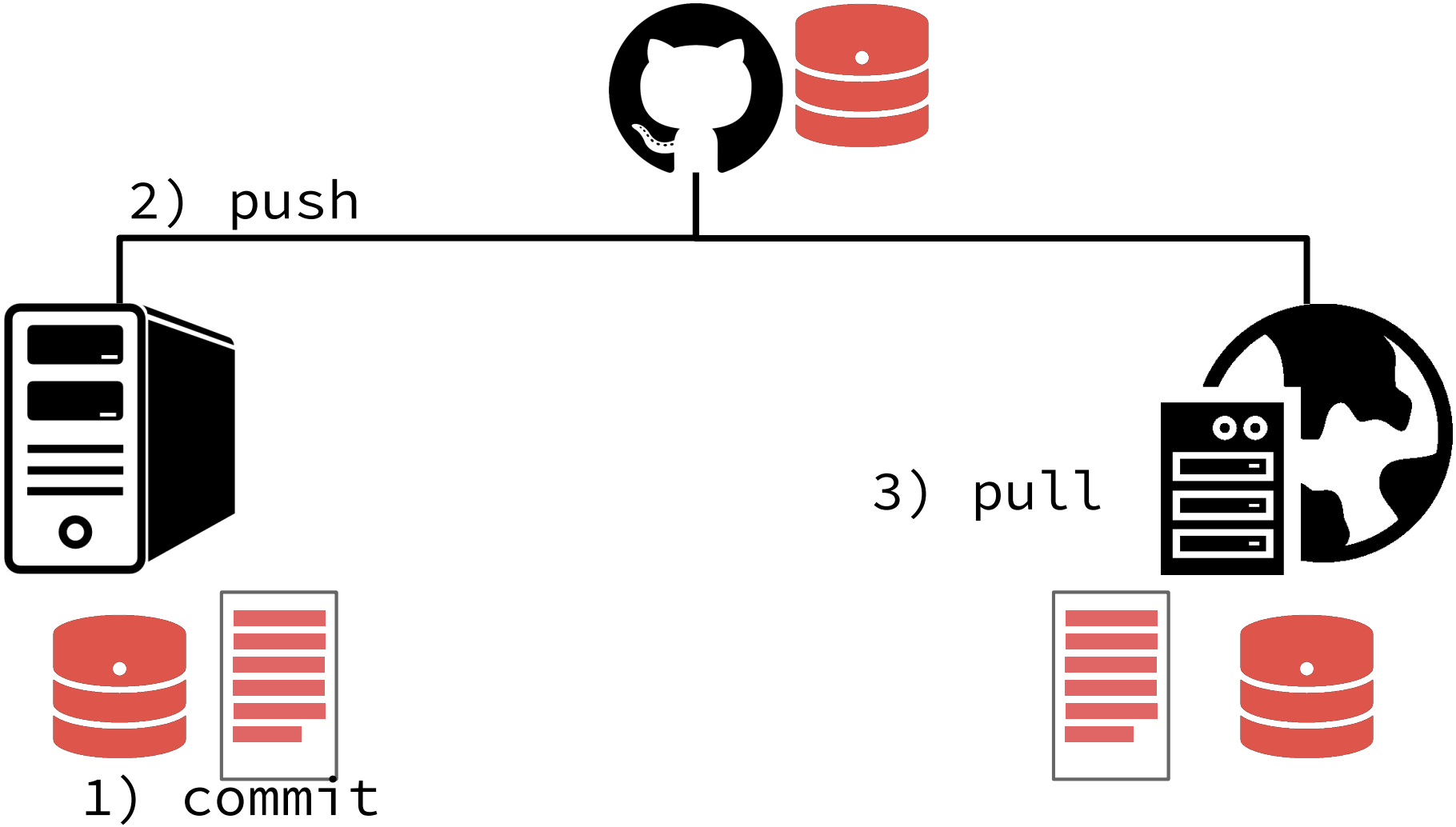
pull

pulling down from the central project repository and updating the branch you are working on

Basic workflow

Common workflow

1. Make changes
2. Commit work
3. Push work
4. Pull other people's work


2) push
3) pull
1) commit

Getting started

New project process

1. Set up a project on GitHub
2. Set up project folder locally
3. Start version managing your folder
4. Add the remote repository
5. Make changes
6. Commit changes
7. Push changes

DevelopMe_

Demo

DevelopMe_

Exercise 1

DevelopMe_

What are we going to do?

1. create a project repository on our GitHub accounts
2. create a project directory on your machine
3. start version managing it with Git
4. add a remote repo
5. start adding files to our project
6. add and commit those files
7. push those files to GitHub
8. make changes to the files and commit them

DevelopMe_

To consider: our first commit

There can be only 1 first commit on a project, so you should plan for this to either be:

1) On GitHub

Creating a repository with a README.md or other file in it

OR

2) On our computer

Doing our first commit in our project folder

DevelopMe_

Steps you'll need to take

1. make a new project directory
2. **cd** to your project folder = type command **cd /path/to/project** and hit enter

Pro tip: drag folder into terminal to get the path

DevelopMe_

Commands we'll need

Option 1: first commit on GitHub

```
$ git init
$ git remote add origin {repository URL, starts git@github.com….}
$ git fetch
$ git checkout master
$ git add {filename}
$ git commit -m "adding my first file"
$ git push
```

DevelopMe_

Option 2: first commit on your computer

```
$ git init
$ git remote add origin {repository URL}
$ git add {filename}
$ git commit -m "adding my first file"
$ git push
```

DevelopMe_

Using the GitHub interface

Now go to GitHub to see that the your file is there.

Editing files

Edit, commit, push

1. Make a change to your files
2. Then $ **git add {filename}** to stage that change
3. Then commit it with $ **git commit -m "some message"**
4. Finally $ **git push** and go to GitHub to see that the change is reflected there

DevelopMe_

Git cheat sheet

DevelopMe_

```
$ git init
```

initialize (start) git handling in the current directory

```
$ git remote add origin {repository URL}
```

add the remote repo location (GitHub)
should be SSH version, starting **git@github.com...**

```
$ git fetch
```

fetches all branches and revisions, to your local machine

DevelopMe_

```
$ git checkout master
```

change your project code to the most recent snapshot in the master branch

DevelopMe_

```
$ git add *
```

adds all files to be version managed

```
$ git add {filename}
```

add a specific file to be version managed

```
$ git commit -a -m 'message'
```

creates a point-in-time snapshot of all changed tracked files (-a), with commit message (-m)

```
$ git commit -m 'message'
```

creates a point-in-time snapshot of all files that have been added to staging, with commit message (-m)

```
$ git status
```

see what state your files are in

```
$ git rev-parse HEAD
$ git reflog
```

DevelopMe_

Command line cheat sheet

$ **pwd**

print working directory, where am I now?

$ **cd {directory name}**

change/move to working directory

$ **mkdir {directory name}**

create a new directory

DevelopMe_

```
$ git clone {repository URL} {folder to create}
```

Create a folder, checkout project into it

DevelopMe_

Exercise 2

DevelopMe_

What are we going to do?

1. create a project directory on your machine
2. start version managing it with Git
3. add a remote repo that already exists
4. fetch and checkout master branch on the project
5. make changes, stage changes and commit them
6. preview changes in browser
7. Don't do **git push** yet

DevelopMe_

Project name | Home | About | Contact | Dropdown

Slide 1 Heading

Cras justo odio, dapibus ac facilisis in, egestas eget quam. Donec id elit non mi porta gravida at eget metus. Nullam id dolor id nibh ultricies vehicula ut id elit.

Sign up today

Heading

Donec sed odio dui. Etiam porta sem malesuada

Heading

Duis mollis, est non commodo luctus, nisi erat porttitor ligula, eget lacinia odio sem nec elit. Cras

Heading

Donec sed odio dui. Cras justo odio, dapibus ac facilisis in, egestas eget quam. Vestibulum id ligula

Getting started

GitHub repository:

git@github.com:develop-me/git-simple.git

DevelopMe_

Steps you'll need to take

1. make a new project directory
2. **cd** to your project folder = type command **cd /path/to/project** and hit enter
3. **$ git init**
4. **$ git remote add origin git@github.com:develop-me/git-simple.git**
5. **$ git fetch**
6. **$ git checkout master**

Or

1. **$ git clone git@github.com:develop-me/git-simple.git {folder name}**

DevelopMe_

Steps you'll need to take

1. make changes to files
2. review what has changed with $ **git status**
3. stage changes with $ **git add ***
4. commit changes with $ **git commit -m "my change message"**
5. preview changes in browser by viewing index.html
6. Experiment with both $ **git commit -am "message"** and $ **git commit -a -m "message"**

DevelopMe_

Git in teams

DevelopMe_

git push

pushes commits to the remote repository for others to access

git pull

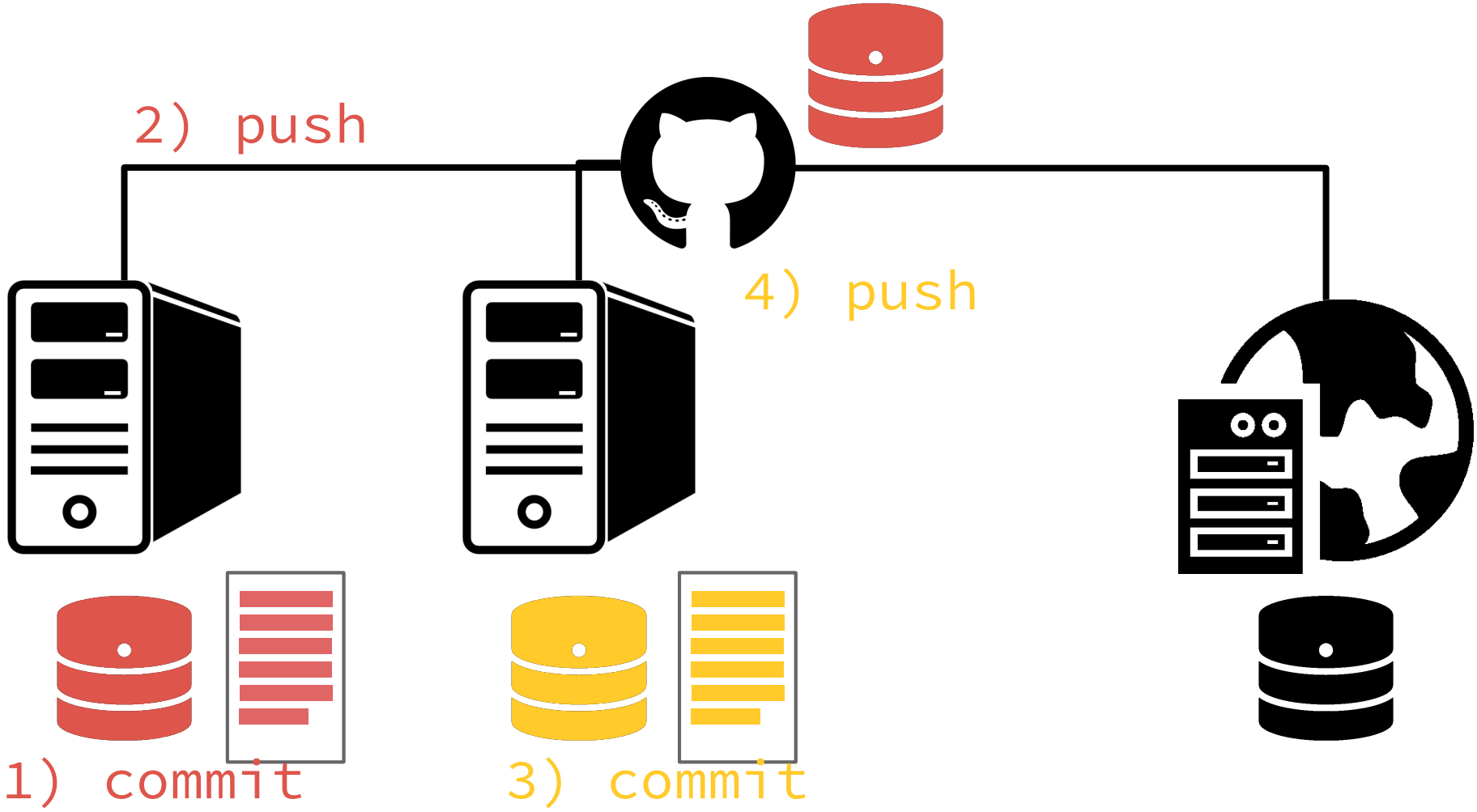
pull changes on current branch down from the repository and update working directory to that snapshot

conflict

Git is pretty smart at merging together changes in different parts of a file, but sometimes there are overlapping changes

in this situation Git will warn you and you'll need to fix the conflict yourself

Push rejection

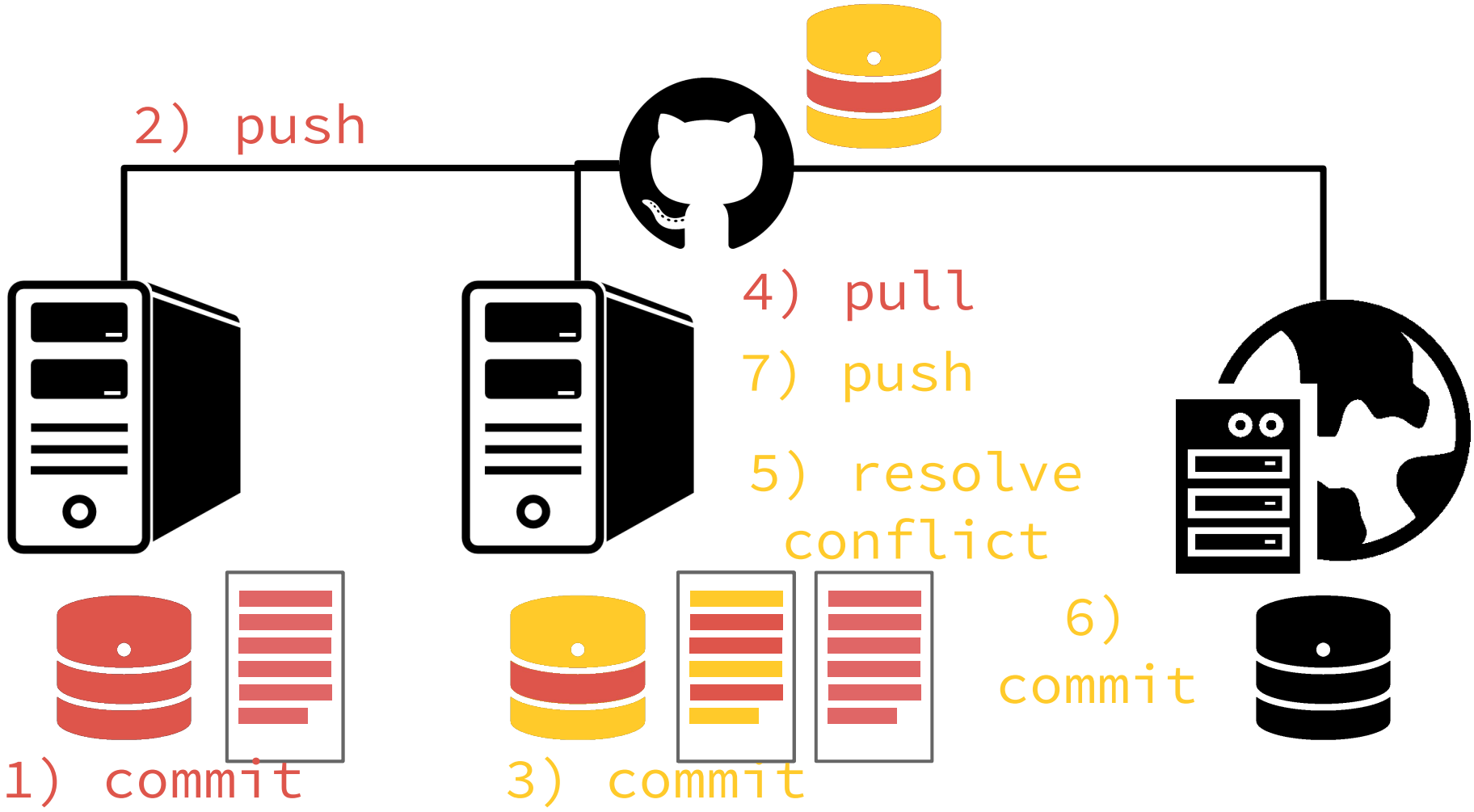

2) push
4) push
1) commit
3) commit

git push

```
To git@github.com:develop-me/bdf.git

 ! [rejected]        master -> master
(non-fast-forward)

error: failed to push some refs to
'git@github.com:develop-me/bdf.git'

hint: Updates were rejected because the tip of your
current branch is behind

hint: its remote counterpart. Merge the remote
changes (e.g. 'git pull')

hint: before pushing again.

hint: See the 'Note about fast-forwards' in 'git
push --help' for details.
```

Conflicts

DevelopMe_


2) push
4) pull
7) push
5) resolve conflict
6) commit
1) commit
3) commit

git pull

```
From github.com:develop-me/bdf
 * branch            master     -> FETCH_HEAD
Auto-merging README.md
CONFLICT (add/add): Merge conflict in README.md
Automatic merge failed; fix conflicts and then
commit the result.
```

DevelopMe_

git status

```
# On branch master
# You have unmerged paths.
#   (fix conflicts and run "git commit")
#
# Unmerged paths:
#   (use "git add <file>..." to mark resolution)
#
#  both added:          README.md
```

DevelopMe_

Demo

DevelopMe_

Editing conflicted file

```
<<<<<<< HEAD
I made these changes to the file.
=======
But someone else made these changes
8dsk329cxlsd93 >>>>>>>
```

DevelopMe_

Auto-merge commit message


gitsimple.developme.training — vagrant@creditcall: /var/www/public — nano ◂ git pull — 77×31
GNU nano 2.0.6 File: ...simple.developme.training/.git/MERGE_MSG
Merge branch 'master' of github.com:develop-me/git-simple
Please enter a commit message to explain why this merge is necessary,
especially if it merges an updated upstream into a topic branch.
#
Lines starting with '#' will be ignored, and an empty message aborts
the commit.
[Read 7 lines]
^G Get Help ^O WriteOut ^R Read File ^Y Prev Page ^K Cut Text ^C Cur Pos
^X Exit ^J Justify ^W Where Is ^V Next Page ^U UnCut Tex ^T To Spell

When doing **git pull** leads to an automatic merge you sometimes get stuck in Vim editor, to get out:

:q [Enter]

This happens as a new snapshot it made after the files have been merged, and this is Git's way of giving the chance of setting a commit message:

"Merge branch 'master' of..."

Exercise 3

DevelopMe_

Starting to push and pull

- now we're going to start using **`git push`** to move our commits up to GitHub, and **`git pull`** to get other people's work
- you'll start to get:
  - push rejections (when other people have changed GitHub while you've been working)
  - automatic merges (when others have changed the same file, but a different part)
  - conflicts (where you've both edited the same part of a file)

DevelopMe_

Best Practice

DevelopMe_

Best practice

- communication
- keep up-to-date with remote (GitHub), by pulling frequently
- don’t commit credentials/configs - security and environment-dependant

Useful commit messages

“another update”

Vs.

“previous styling changes had an issue on Safari mobile, this CSS hack will resolve”

See also: GDS Git style guide, How to Write a Git Commit Message and 5 Useful Tips For A Better Commit Message

DevelopMe_

Use Git to move or delete files

```
git rm {file}
```

```
git mv {file} {directory}
```

DevelopMe_

Removing files

Use Git to move or delete files

git rm
Delete file and stage deletion.

git rm --cached README
Remove file from tracking, but retain the file.

Add to .gitignore file
Ensure file is not added back into tracking in future.

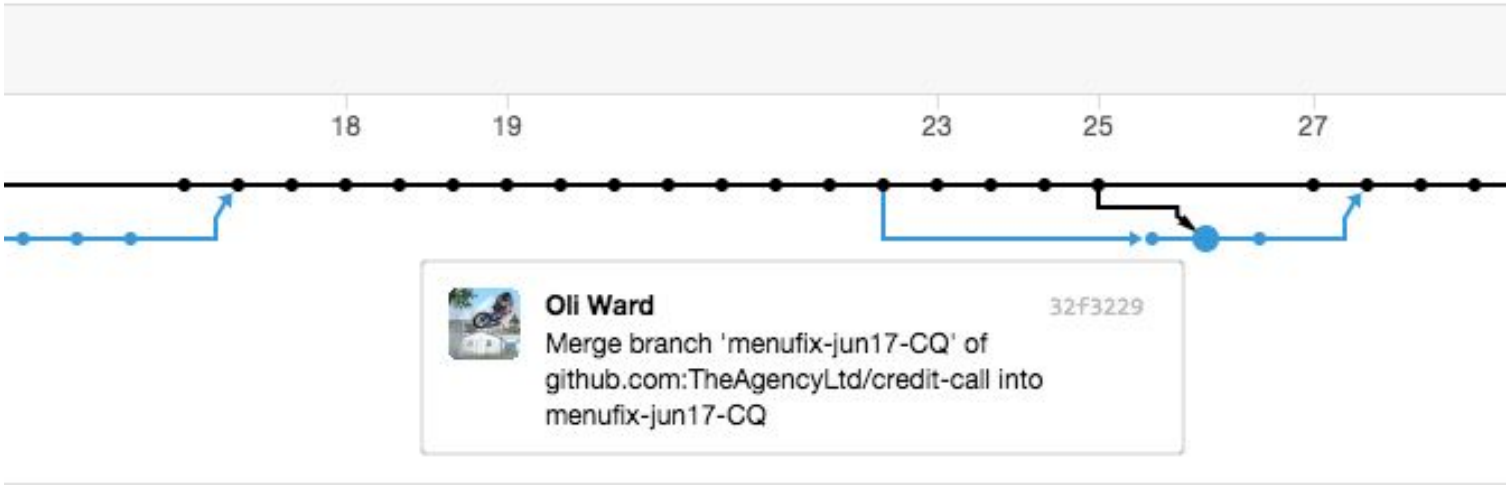
Branches

DevelopMe_

Git branches

A valuable tool for your development workflow

Can be used in many different ways, for different purposes



branch

a single development stream, or split in the project, allowing you to try things out or develop features without affecting other work

merge

merging of two branches together, for example, your experimental branch back into the normal working branch, to release that experimental code

Git cheat sheet

DevelopMe_

git branch my-experimental-branch

creates a new branch from the current commit

git checkout my-experimental-branch

checks out your new 'my-experimental-branch' branch for working on (switches your active branch, so new commits go onto this branch)

`git push origin my-experimental-branch`

pushes your new branch to the remote repository for others to use

For subsequent changes you can use the command to push only changes to your working branch to the remote, else it will also try to push changes to all other branches too, e.g. master

Merging branches

git checkout master

move to working on the master branch

git pull

make sure your master branch is up-to-date

git merge my-experimental-branch

merge the my-experimental-branch into master

DevelopMe_

git branch -a

list the branches that exist locally and remotely, with * for current working branch

git branch -d my-experimental-branch

delete local branch, use -D to force

git push origin :my-experimental-branch

delete remote branch

DevelopMe_

For overwriting one branch with another

force overwrite master with branch reset-branch, replaces master code with branch without merging (replaces all files):

```
git checkout reset-branch

git merge -s ours master

git checkout master

git merge reset-branch
```

DevelopMe_

Full branching process

Branch & merge process

1. **git checkout master**
2. **git pull**
3. **git branch my-experimental-branch**
4. **git checkout my-experimental-branch**
5. [do work]
6. **git commit -am "commit message"**
7. **git push origin my-experimental-branch**
8. **git checkout master**
9. **git pull**
10. **git merge my-experimental-branch**
11. **git push**

Exercise 4

DevelopMe_

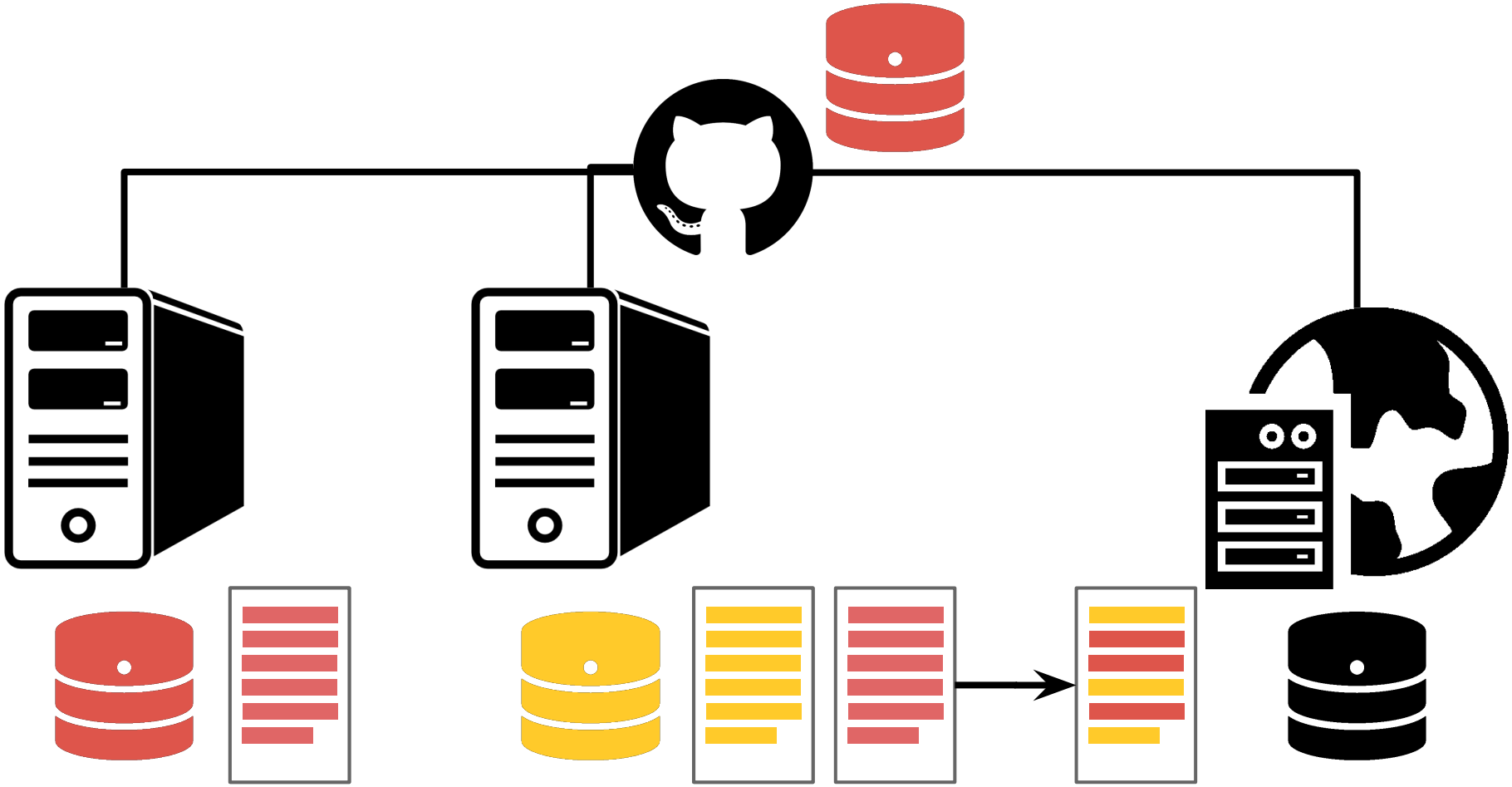
What are we going to do?

1. create a branch and check it out
2. make some changes on that branch
3. merge that branch back into master
4. push those changes
5. review the network graph to help you understand branch history

DevelopMe_

Workflows

DevelopMe_



Planning and choosing a workflow

Which workflow to use depends on:

- team structure
- project requirements
- project roadmap

DevelopMe_

Considerations

Who?

- Who’s going to work on what? Task delegation/ownership.
- Can the project work be chopped up into chunks/tasks?
- Is there a clear delegation based on team roles? Front-end, back-end, tech lead.

DevelopMe_

When?

- What is the project timeline?
- What are the milestones?
- Are certain tasks dependant on others?
- When are things going live?

How?

- Is that work going to be reviewed? How?
- How is it going to be tested?
- How is it going merged and deployed?

Other considerations

- Are we releasing in phases?
- What happens if there's a bug in the live system?
- Have we got a branch setup that supports hotfixes? Or are we working straight on master with new features? Or merging back into master for testing?

DevelopMe_

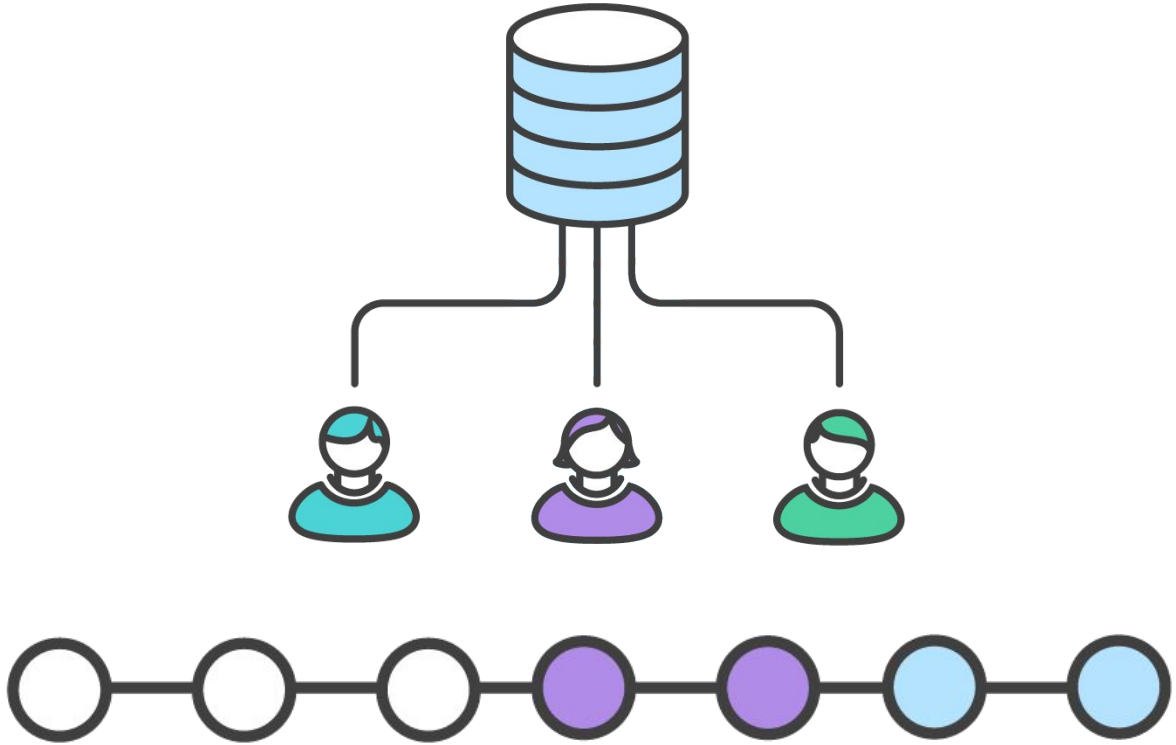
Git workflows

DevelopMe_

- Centralized Workflow
- Feature Branch Workflow
- Gitflow Workflow
- Forking Workflow

DevelopMe_

Centralised workflow

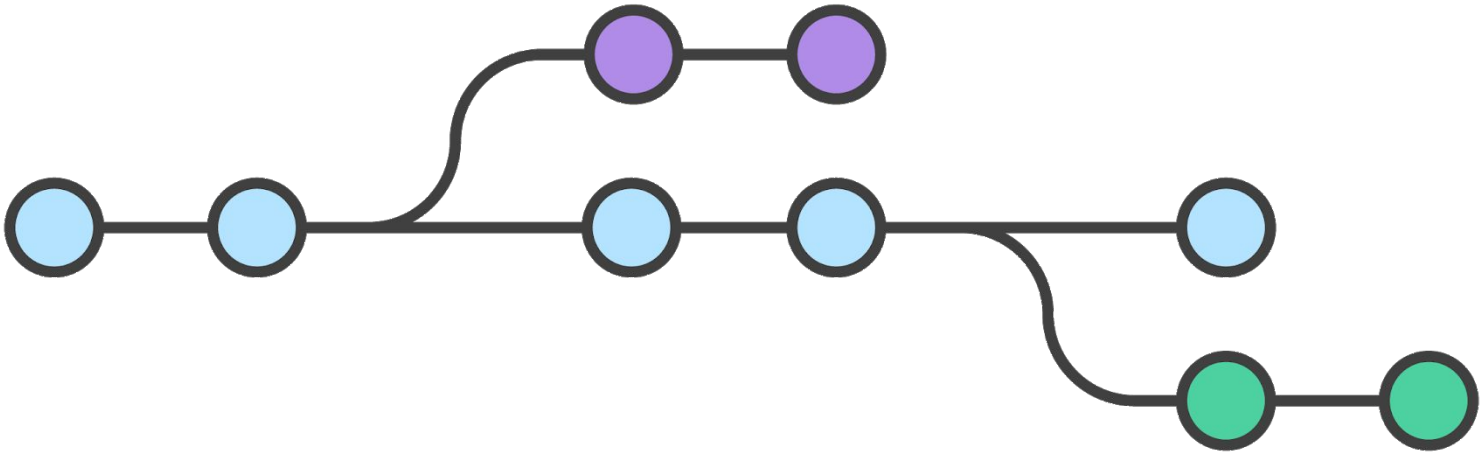


From: https://www.atlassian.com/git/tutorials/comparing-workflows

DevelopMe_

Feature branch workflow

All feature development takes place in a dedicated branch instead of the master branch, with branches merged back into master when ready.



DevelopMe_

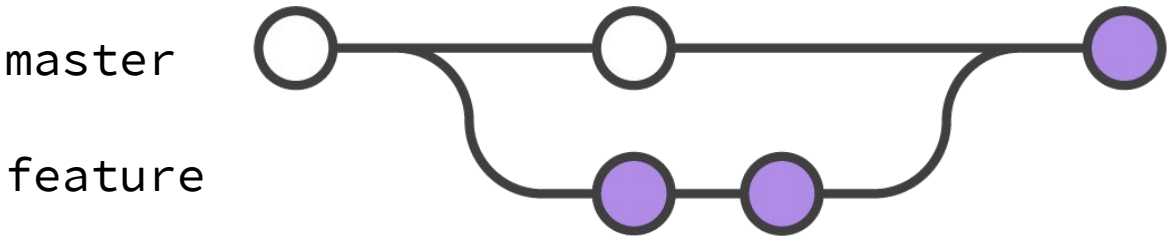
Feature branch workflow features

- Adding feature branches to your development process is an easy way to encourage collaboration and streamline communication
- Easy for multiple developers to work on a particular feature without disturbing the main codebase.
- **master** branch will never contain broken code, good for continuous integration environments.

Pull requests

Can then use pull requests, which are a way to initiate discussions around a branch.

Give other developers or lead developer the opportunity to sign off on a feature before it gets integrated into the official project.



DevelopMe_

TheAgencyLtd / credit-call PRIVATE

Unwatch 4 Star 0 Fork

feature-addnewlogos ... master Edit

Title

Write Preview Markdown supported Edit in fullscreen

Leave a comment

Attach images by dragging & dropping, selecting them, or pasting from the clipboard.

✔ Able to merge.

These branches can be automatically merged.

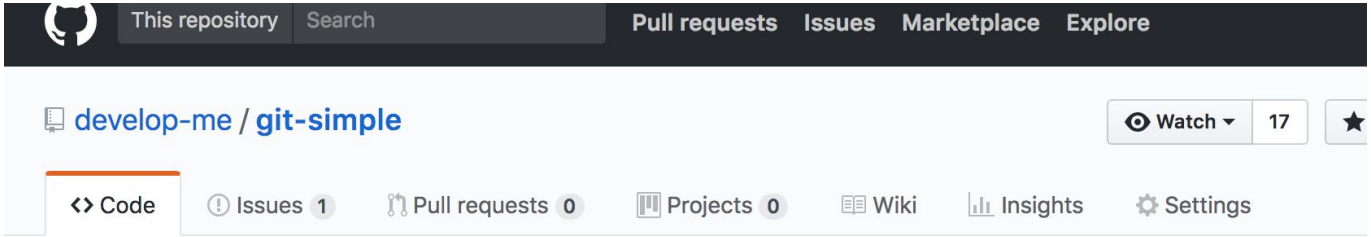
Create pull request

Process with Pull Requests

1. **git checkout master**
2. **git pull**
3. **git branch my-feature-branch**
4. **git checkout my-feature-branch**
5. [do work]
6. **git commit -am "commit message"**
7. **git push origin my-feature-branch**
8. **Go to GitHub**
9. **Find your branch in Code > Branches**
10. **Create pull request for your branch**

To receive notifications on pull request

1. Watch the repository



2. Update your notification settings:
https://github.com/settings/notifications

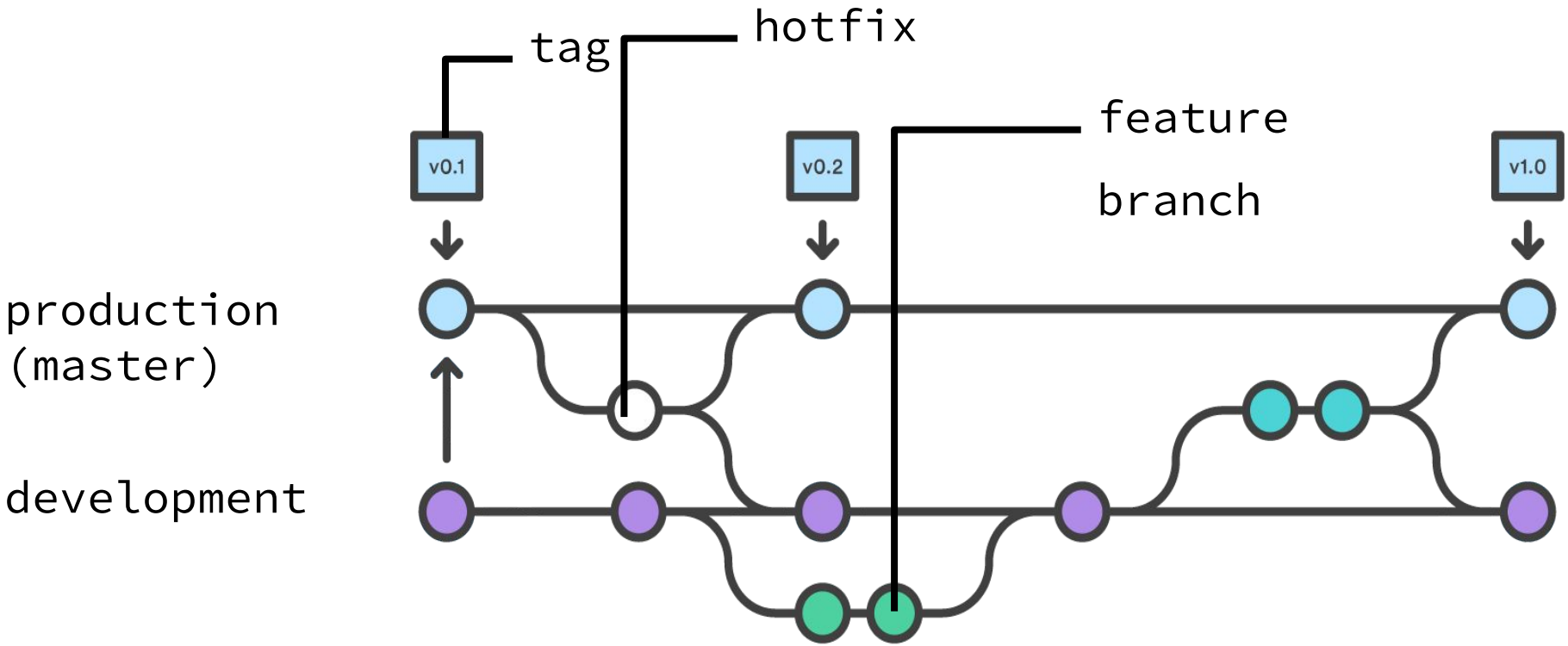
Exercise 5

DevelopMe_

What are we going to do?

1. assign a Tech Lead to review PRs
2. continue with feature branch workflow, but request pull requests instead of merging yourself
3. rotate Tech Lead

Gitflow workflow


tag
hotfix
feature
branch
v0.1
v0.2
v1.0
production
(master)
development

DevelopMe_

Gitflow workflow features

- All development should take place in a dedicated branch instead of the **master** branch
- New features should reside in their own branch
- Instead of branching off of **master**, feature branches use **development** as their parent branch
- When a feature is complete, it gets merged back into **development**
- Features should never interact directly with **master**
- Features go live with **development** merging to **master**

DevelopMe_

git tag v1.0
Tag your commit

git tag -a v1.4 -m 'my version 1.4'
Annotated Tags

git push origin --tags
Sharing (Pushing) Tags

git push origin <tag_name>
Push a single tag name to remote

Process with gitflow

1. **git checkout development**
2. **git pull**
3. **git branch my-feature-branch**
4. **git checkout my-feature-branch**
5. [do work]
6. **git commit -am "commit message"**
7. **git push origin my-feature-branch**
8. **Go to GitHub**
9. **Find your branch in Code > Branches**
10. **Create pull request for your branch**
11. **continue with next task (back to 1.)**

Exercise 6

DevelopMe_

What are you going to do?

Collaborate in your team using the Gitflow workflow

Discuss your working practice, and who will do what in terms of setting up and ongoing roles:

- Tech lead / pull request reviewer (merging to **development**)
- Release manager (merging to **master**)
- Product owner (documenting new features)
- Developers

DevelopMe_

Product Owner

- Drives direction of the product
- Understands our customer needs and issues
- Creates Issues in GitHub documenting new things to be built by development team

DevelopMe_

Tech Lead

- Manages development team(s)
- Reviews their pull requests, merging feature branches into **development**

Release Manager

- Schedules new product releases
- Ensures documentation is updated, customer support teams are trained
- Reviews pull requests merging **development** into **master**
- Tags the commit of a new release
- Writes release notes (in GitHub)

DevelopMe_

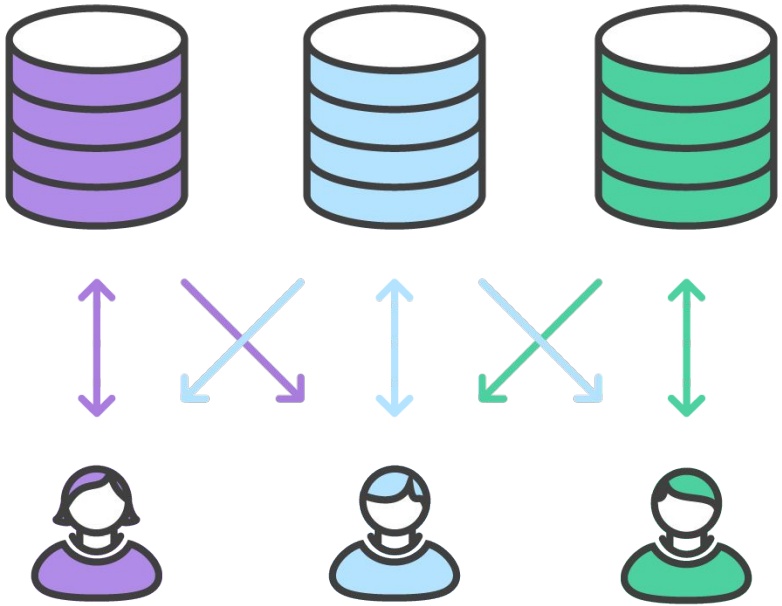
Developers

- Takes Issues and work on them
- Mark themselves as assignee
- Branches from **development** branch
- Issues pull requests back into **development** branch, with Tech Lead as reviewer

DevelopMe_

Forking workflow

Instead of using a single server-side repository to act as the “central” codebase, it gives every developer a server-side repository.



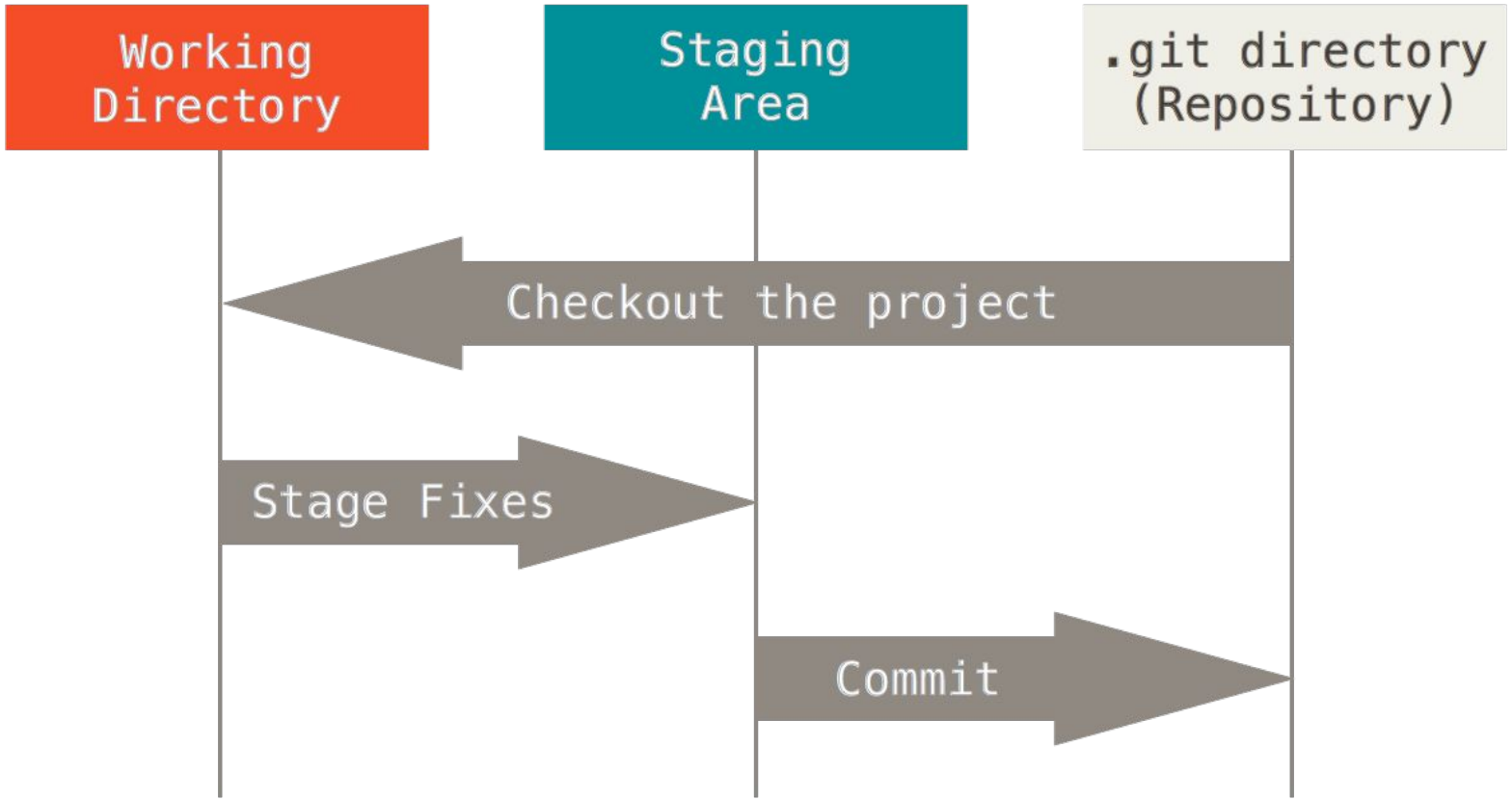
DevelopMe_

- Contributions can be integrated without the need for everybody to push to a single central repository.
- Developers push to their own server-side repositories
- Only the project maintainer can push to the official repository
- Allows the maintainer to accept commits from any developer without giving them write access to the official codebase.
- Distributed workflow that provides a flexible way for large, organic teams (including untrusted third-parties) to collaborate securely.
- This also makes it an ideal workflow for open source projects.

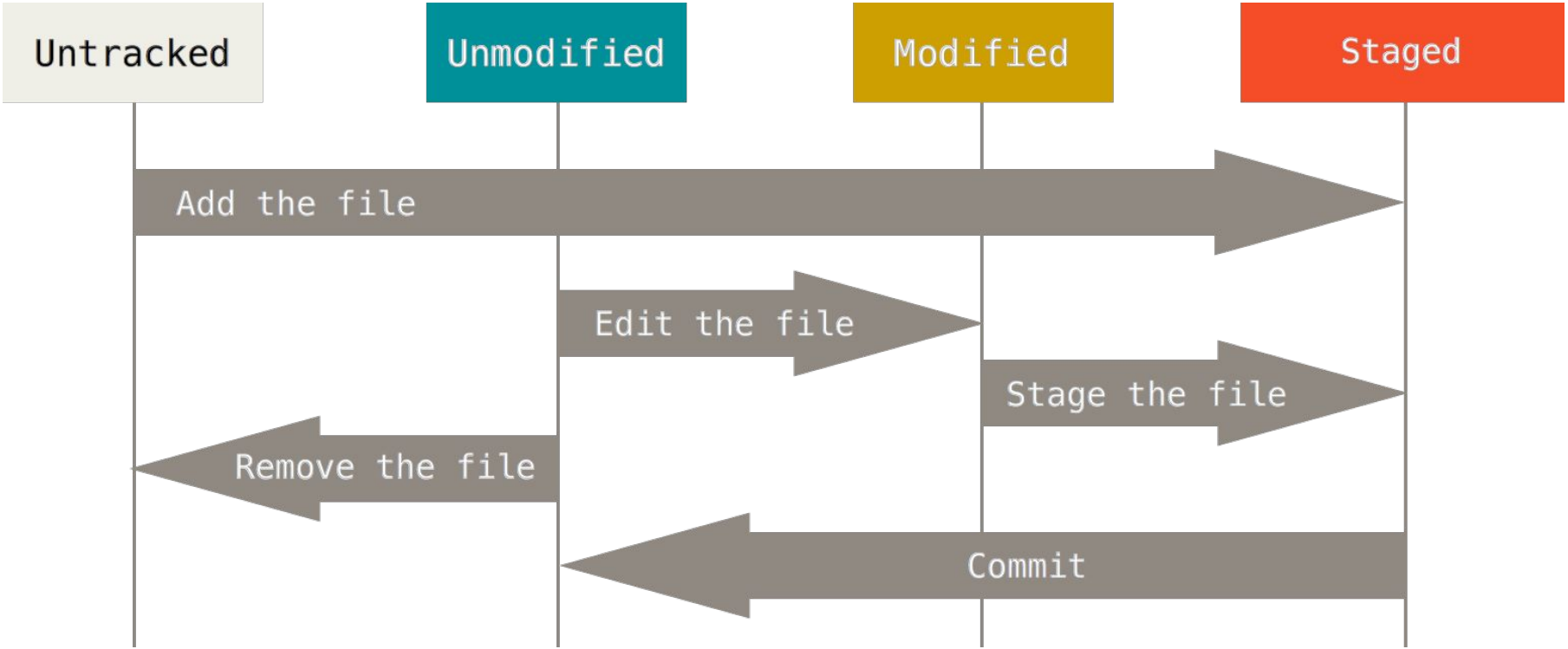
staged vs. committed

DevelopMe_

Staging

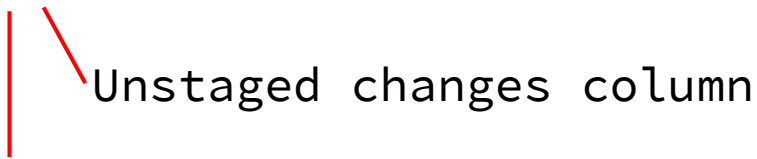

Working Directory
Staging Area
.git directory (Repository)
Checkout the project
Stage Fixes
Commit

Adding and editing files



Modified / staged / committed

```
# git status -s
 M README
MM Rakefile
A  lib/git.rb
M  lib/simplegit.rb
?? LICENSE.txt
```



DevelopMe_

Markdown

DevelopMe_

What is markdown?

A **lightweight markup language** with plain text formatting syntax.

It is **designed so that it can be converted to HTML** and many other formats using a tool by the same name.

Good for...

- GitHub readme page (make a **readme.md** file in project root)
- Compiling into simple, static, HTML sites
- Note taking, faster than using MS Word, creating formatting with syntax
- Developer documentation

Some tools that use markdown

- Turn markdown into slides: https://github.com/partageit/markdown-to-slides
- Turn markdown into notes: https://github.com/joeyespo/grip

Example 1

```
# This is a h1 title

## This is a h2

This is paragraph text.

*Italic text*

* First bullet

* Second bullet
```

This is a h1 title

This is a h2

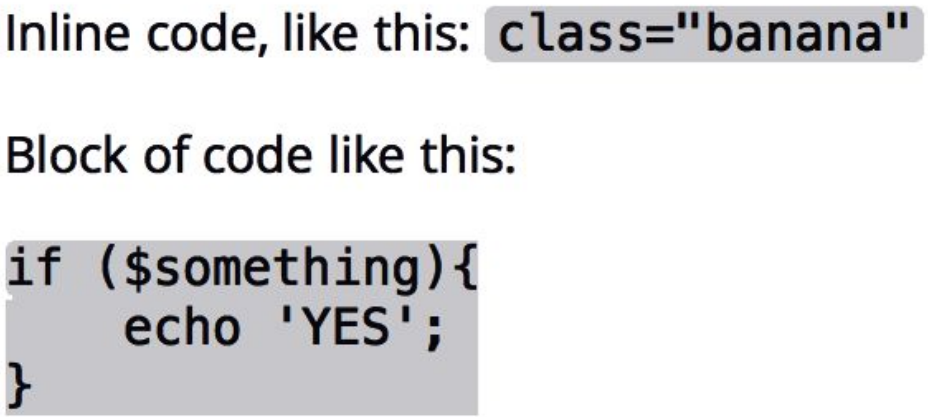
This is paragraph text.

Italic text

- First bullet
- Second bullet

Example 2

````
Inline code, like this: `class="banana"`

Block of code like this:

```
if ($something){
    echo 'YES';
}
```
````



(Some) markdown syntax also works in:

- Slack
- WhatsApp
- Skype
- (probably more)

Preview markdown in Sublime

- Bring up command palette with ⌘**+Shift+P** or **Ctrl+Shift+P**
- Search for: "Package Control: Install Package"
- Search for: "MarkdownLivePreview"
- Hit **Enter** to install

Then:

- Make a **.md** markdown file
- Hit **Alt+M** to bring up preview

DevelopMe_

Cheatsheet of syntax

https://github.com/adam-p/markdown-here/wiki/Markdown-Cheatsheet

DevelopMe_

Useful resources

DevelopMe_

Some helpful Git resources

- When it all goes wrong: ohshitgit.com
- All about **`git stash`**
- Git branch and command autocomplete with [tab]

Quiz

DevelopMe_

1) Getting setup with Git

What does this command do?

```
git remote add origin git@github.com:develop-me/drupal1.git
```

2) Committing files

How would I stage the images folder and index.html for a commit with message "Apple" and then commit hello.html with message "Banana"?

```
Oli-Wards-MacBook-Pro:git oliward$ git status
On branch master
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git checkout -- <file>..." to discard changes in working directory)

        modified:   hello.html

Untracked files:
  (use "git add <file>..." to include in what will be committed)

        images/
        index.html
```

3) Oh noes #1

What command have I just run and what has gone wrong?

What should I do next?

```
To git@github.com:develop-me/git-simple.git
 ! [rejected]        development -> development (fetch first)
error: failed to push some refs to 'git@github.com:develop-me/git-simple.git'
hint: Updates were rejected because the remote contains work that you do
hint: not have locally. This is usually caused by another repository pushing
hint: to the same ref. You may want to first integrate the remote changes
```

4) Oh noes #2

What command have I just run and what has gone wrong?

What should I do next?

```
remote: Counting objects: 8, done.
remote: Compressing objects: 100% (7/7), done.
remote: Total 8 (delta 2), reused 0 (delta 0), pack-reused 1
Unpacking objects: 100% (8/8), done.
From github.com:develop-me/git-simple
   847f44a..7e0d545  development -> origin/development
Updating 847f44a..7e0d545
error: Your local changes to the following files would be overwritten by merge:
        index.html
Please, commit your changes or stash them before you can merge.
Aborting
```

5) Oh noes #3

What command have I just run and what has gone wrong?

What should I do next?

```
Auto-merging index.html
CONFLICT (content): Merge conflict in index.html
Automatic merge failed; fix conflicts and then commit the result.
```

6) Burn it all!

How do you 'throw away' all work since last commit,

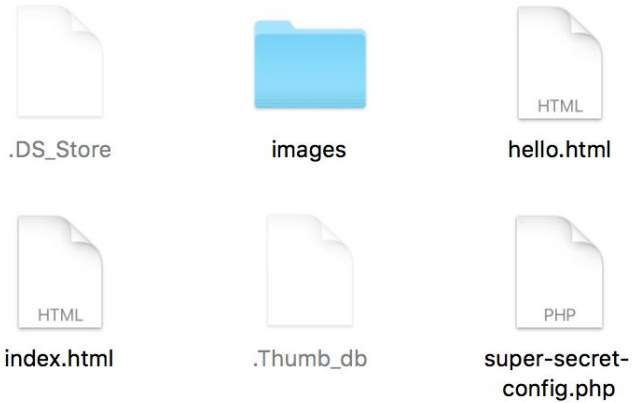
and revert to how the files were at that moment?

7) (Sort of) Burn it all!

How do you 'throw away' the last commit (**bbbbbb**), going back to the commit before last (**aaaaaa**), but keep the files as they are (in commit **bbbbbb**), to not lose that work.

8) Ignore it all!

Create a .gitignore file that will cause Git to only track hello.html in my folder:



DevelopMe_

9) Describe the process

You're working in a team that uses gitflow.

Document the steps, and git commands, from starting a new piece of work, to submitting it as a pull request.

1. `$ git command1`
2. `$ git command2`
3. `$ git command3`
4. ...

DevelopMe_

Thank you.