Teaching People Git

Emma Jane Hogbin Westby // @emmajanehw // emmajane
www.gitforteams.com
“I’ve tried! I can’t learn this stuff.”

A typical student in one of my workshops.
It is hard to learn.

It is harder to learn Git.
Git supports our work.

For most of us: git is not our work. It is merely a “stupid content tracker”.
It’s hard to “hello world” complicated processes.

Git is typically used on real work where the stakes are high. Git is typically taught as a series of commands to memorise.
The “Hello World” for Git Workshops

$ git init
$ touch file
$ git add .
$ git commit -m “First!”
Bloom’s Taxonomy

Source: http://lb.cm/bt
Motivating Adult Learners to Learn

- **Need to know**: Adults need to know the reason for learning something.
- **Foundation**: Trial and error provides the basis for learning activities.
- **Self-concept**: Adults need to be responsible for their decisions on education; involvement in the planning and evaluation of their instruction.
- **Readiness**: Adults are most interested in learning subjects having immediate relevance to their work and/or personal lives.
- **Orientation**: Adult learning is problem-centered rather than content-oriented.
- **Motivation**: Adults respond better to internal versus external motivators.

http://en.wikipedia.org/wiki/Andragogy
Constructivism

Piaget, Papert, Montessori
Source: http://bit.ly/1Cn5vsg
I've tried! I can't learn this stuff.
“Please memorise all Git commands and use only rebasing when merging your work.”

~ No client ever
Start with the whole to solve real problems.
Your problems are 90% social.
Teach teamwork first.
Put the commands in context.
What’s your role?
Who else is on your code team?

Write down a list of all of the people on your code team. This list may include:

- developers
- designers
- project managers
- clients
What are your tasks?

- Download work
- Create snapshot
- Share work
Where do you fit in?

Maybe you do everything. Maybe you only do some things. Write a list of all the tasks you are actually responsible for. This might include:

- Writing code.
- Reviewing code.
- Pushing tested code to the server.
- Fixing broken code.
What are your tools and restraints?

Often there are other things we need to fit into our workflow. Create a third list of any tools and restraints you are aware of. This list might include:

• Version control software (we’ll always assume Git)

• Code hosting system (Bitbucket, GitHub, self-hosted)

• Server ecosystem (dev / staging / live)

• Code editors & integrated developer environments (vim, Dreamweaver, Sublime, PHPstorm)

• Automated testing systems or review “gates”
I tricked you.

Now you’ve got a cheat sheet of topics to put into your diagrams and your documentation.
Sketch out your workflow

- Identify the roles on your team.
- Identify the relationships between the team members.
- Draw arrows to show how code flows between team members.
- Time: 5 minutes
Pedagogy of the Oppressed

Paulo Freire
Sketch out your branch management strategy

- Identify the roles on your team.
- Identify the relationships between the team members.
- Draw arrows to show how code flows between team members.
- Time: 5 minutes
Co-Create Resources

You want to remove changes to your file(s)

Has the change been committed?

No

Is the change in the staging index?

No

Are there changed files you want to preserve in the working directory?

No

reset --hard

This is a multi-step sequence.

reset <filename>

then

Yes

Are there changed files you want to preserve in the working directory?

Yes

reset <filename>

Has the change been committed?

Yes

Do you want to keep a reference to the committed change in your log?

No

Is this a shared branch?

No

It is not appropriate to re-write history.

Yes

Is the change in the most recent commit(s)?

No

rebase --interactive

Are there changed files you want to preserve in the working directory?

Yes

The change is in the committed history.
Recommendations for Teaching Git

• Put the learner first.

• Have learners “build” their knowledge by applying their own expert knowledge about their team to a diagram.

• Once you know the learner’s context, teach relevant workflows.

• Co-create visual maps and documentation to help the learner build their own understanding of Git.

• Teach people to talk to one-another about their work flow.
Anti-patterns for Teaching Git

• “It’s easy!”

• “It’s just a directed acyclic graph!”

• “You should use rebasing. It’s really important.”

• “Don’t ____. It’s really dangerous.”
Recommendations for Git

• Reduce the barrier to entry.

• Reclaim the word “git” and make it a friendly place to spend time.

• Modernise the language in Git to remove the rough edges.

• Make the help experience more consistent.

• Create a centralised repository for learning which is vendor-agnostic.
Teaching People Git

Emma Jane Hogbin Westby // @emmajanehw // emmajane
www.gitforteams.com