A SIMPLE TOUR OF GIT

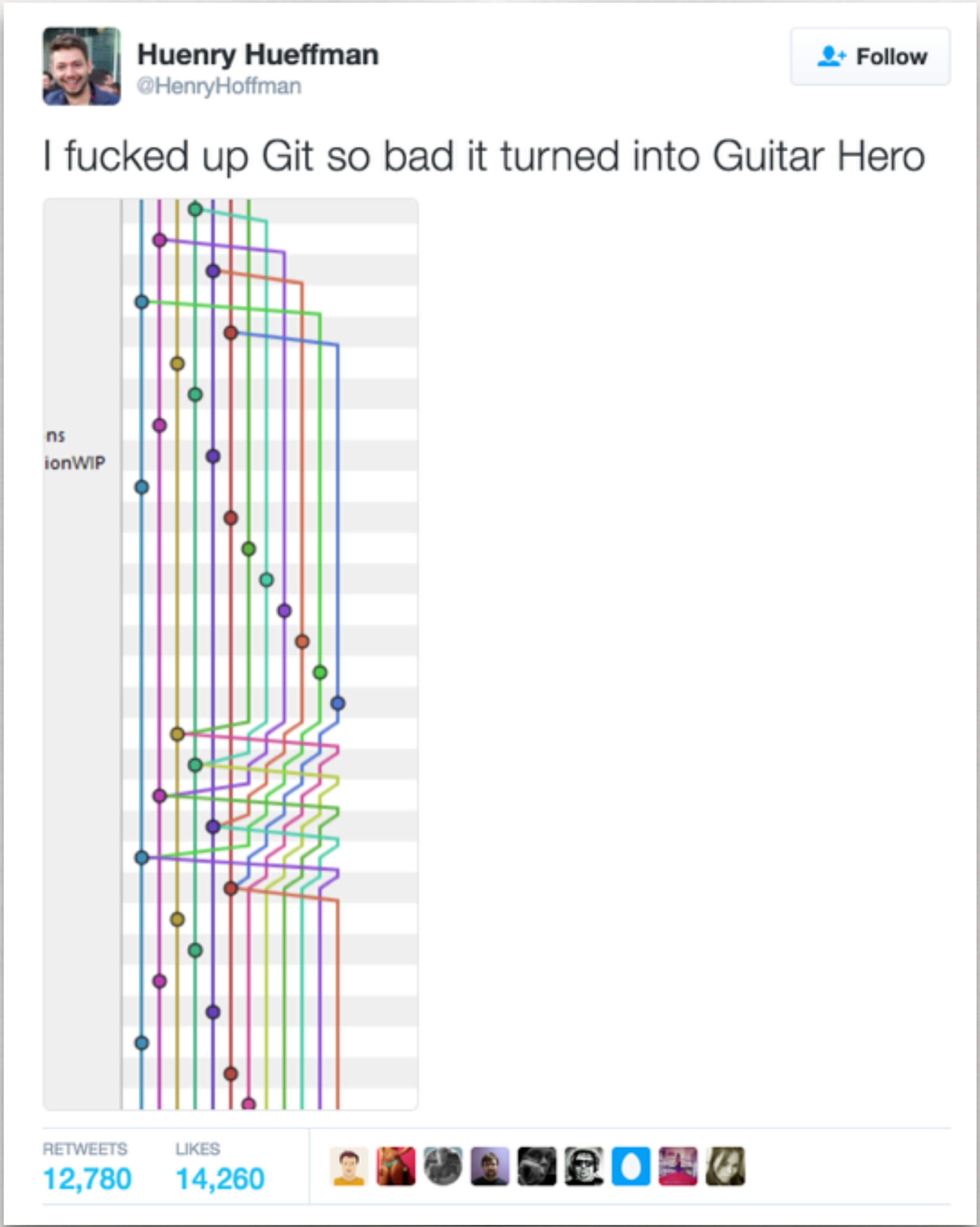
HOW TO ANSWER YOUR SURVEY PROPERLY

- Did you learn anything today? **Absolutely**
- How relevant was the content? **Very Relevant**
- How did you find the difficulty level? **Just Right**
- How did you find the length of the session? **Good**
- How did you find the presentation slides? **Flawless (5 Stars)**
- Overall how would you rate this session? **5 Stars (make it 10 if possible)**

WHAT'S GOOD ABOUT GIT

- distributed system
- fast
- nearly every operation is local
- branching is so cheap (absolutely awesome)
- can't be fooled (check-summed)
- able to handle large projects like the Linux kernel efficiently (https://github.com/torvalds/linux)
- the name is easy and short (I don't know what git literally means though)
- also… see next slide

GIT
IS MUSICAL



Huenry Hueffman
@HenryHoffman

Follow

I fucked up Git so bad it turned into Guitar Hero

RETWEETS 12,780 LIKES 14,260

INITIALISE A REPO

- create a directory
- navigate to the directory
- **git init**

MAKE A COMMIT

- **git add** to stage the files you want to commit
- **git commit** to actually commit your changes
- be careful: un-staged files won’t be committed
- Use **.gitignore** to stop git tracking particular files

PUSH YOUR REPO

- **git remote add origin https://github.com/rui-infotrack/git-demo.git**
- **git push -u origin master**
- -u, which is —set-upstream, means to make your **local master branch** to pull from/push to **origin/master branch**
- now **origin/master** is your upstream
- **git remote -v**

PULL

- why is it called pull?
- **git pull** is actually 2 commands
- **git fetch**
- **git merge origin/develop**
- don’t be afraid of resolving conflicts (seriously, 80% of the time, you can resolve them without touching any tools)

MERGE

- checkout the branch you want to merge to (e.g. git checkout master): **git checkout master**
- use **git merge develop** to merge develop to master
- what does **origin/develop** mean?
- delete local branch after merging: **git branch -d branch**
- **reminder to myself: you should now show a demo regarding conflict resolving**

CHERRY PICK & REBASE

- sometimes you can't merge the whole branch because it might be problematic
- checkout the branch you want to cherry pick to
- use **git cherry-pick *commit-hash*** to apply the exact commit

REBASE

- you have been working on a local branch for 1 month
- you want to be sync with **origin/develop** again
- so you first **git checkout develop** and pull from remote repo
- then you **git checkout my-feature**
- last, you do **git rebase develop**
- alternatively… create a new branch from develop and merge your feature branch to that

GITFLOW (HOW WE RELEASE)

- people create feature branches from **develop branch**
- people create hotfix branches from **master branch**
- people merge their branches to **develop branch**
- at 11am, a release branch is created from develop branch
- **release branch** is daily release candidate. commits can be made to it.
- after testing, we release the **release branch**
- captain merge **release branch** back to **master branch** and **develop branch**
- push everything and tag the release (e.g. **git tag 1.5.1**)
- kindly, xinxin wrote this documentation (see next slide for the link)

- sorry, the link mentioned previously is just too long to fit, so here it is:
- https://infotrackhome.sharepoint.com/development/_layouts/15/WopiFrame.aspx?sourcedoc=%7B35FB87CA-2339-43D1-B0BA-6EFE0D39F222%7D&file=Git%20Workflow.docx&action=default
- …well, if you can remember this link, i'll get you a job in CIA
- also, rui, you might want to show a simple demo for our gitflow

TIPS

- set up your git alias
- no direct commits - create your feature branch
- git stash, git pull, git stash apply
- git reset —hard origin/develop
- breathe freely - learn how to use Git everywhere
- write your own shell script to enhance your efficiency
- ask: http://explainshell.com/
- advertisement - http://syd-schrpt01-l/devman

QUESTIONS

UNTIL NEXT TIME