



GIT: FAST VERSION CONTROL



CSIR

our future through science

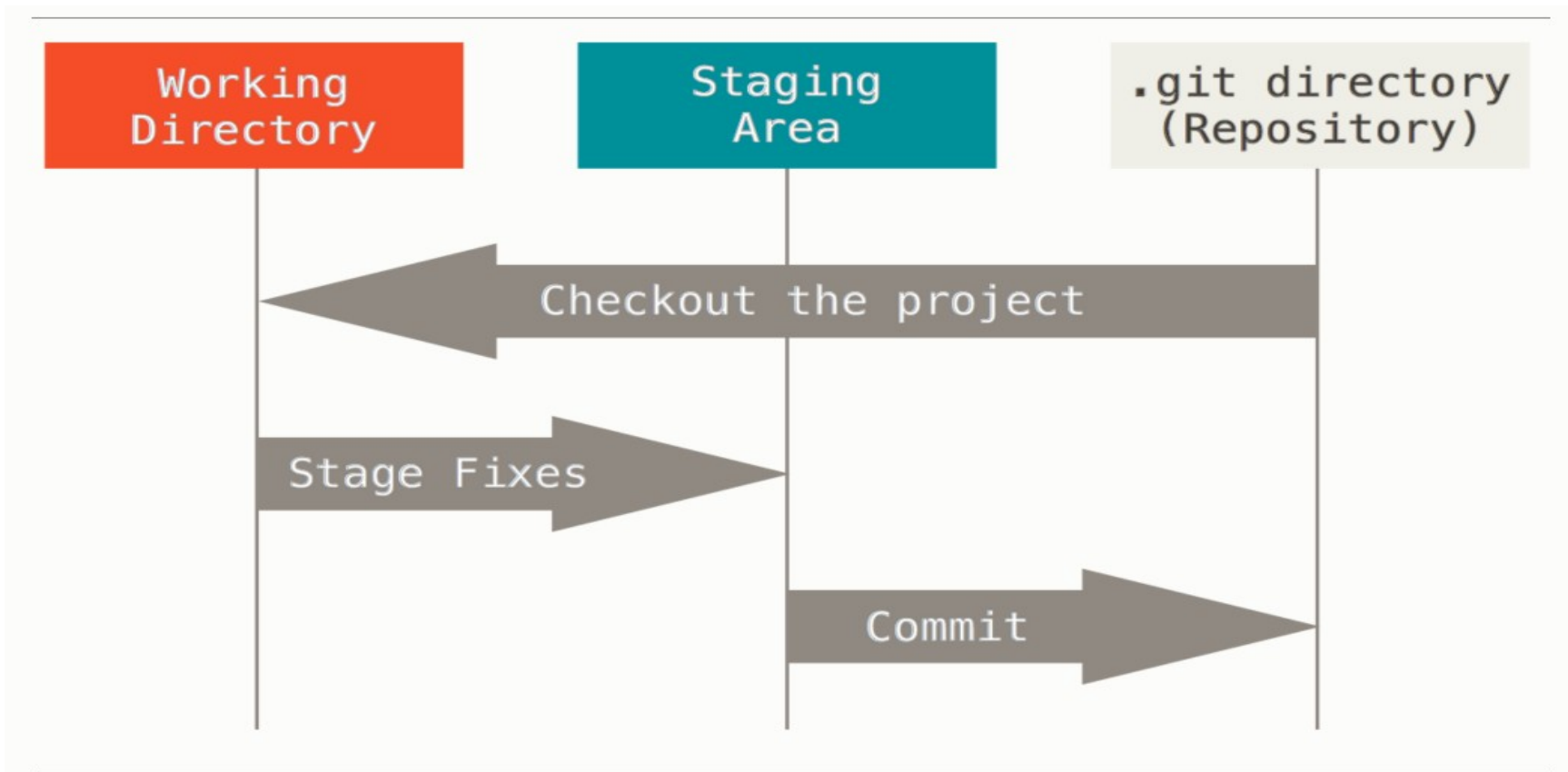
About version control: VCS

- A system that **records changes** to a file or set of files over time so that you can recall specific versions later.
- Yep...you can do this yourself: **error** prone
- Types of VCS:
 - Local VCSs: on local pc keeps all changes to files under revision control.
 - Your pc gone...all gone 
 - Centralized VCS CVCSs: collaboration—single server contains all the versioned files, an number of clients check out files from that central place.
 - Server goes down, no collaboration or saving your own versions...if server gets corrupted: everyone loses everything. 
 - Distributed VCS DVCSs (Git): clients don't just check out latest snapshot of files: they fully mirror the repository. Server dies, any of the client repositories can be copied back to restore it: Every clone is full backup of the data.



Basic Git workflow

- Modify → stage → commit:



Getting started: Installing

- Linux:
 - `$ sudo apt-get install git-all`
- Mac: Mavericks (10.9) or above you can do this simply by trying to run git from the Terminal
- Windows: <https://git-scm.com/download/win>

First-time Git setup

- Set your user name and email address.
 - `$ git config --global user.name "Jane Doe"`
 - `$ git config --global user.email janedoe@example.com`
- Only once if you pass `--global` command: this is info git will use for anything you do on the system
- Configure default text editor:
 - `$ git config --global core.editor emacs/scite/`

Git basics: Getting a git repo

- Two main approaches:
 - Initializing repo in an **existing** directory:
 - Navigate to the directory
 - Type: `$ git init`
 - Creates a **.git** directory
 - Excluding files: create `.gitignore`
 - Add files to be excluded into `.gitignore`
 - Add files you want to start version-controlling
 - `$ git add *.f90`
 - `$ git add *.doc`
 - `$ git commit -m 'initial project version'`
 - **Cloning** an existing repository:
 - `$ git clone <path to remote repo>`

Start working!

- Check status of your files:
 - \$ git status
- Should have a clean directory, your branch is master
- Create a new file in the directory
 - \$ git status: nothing added to commit
 - \$ git add README
 - \$ git commit
- Removing files you added but decided not to track:
 - \$ git rm --cached *filename*
 - \$ git add .
 - \$ git commit -am "Removed ignored Files"

Viewing exact differences

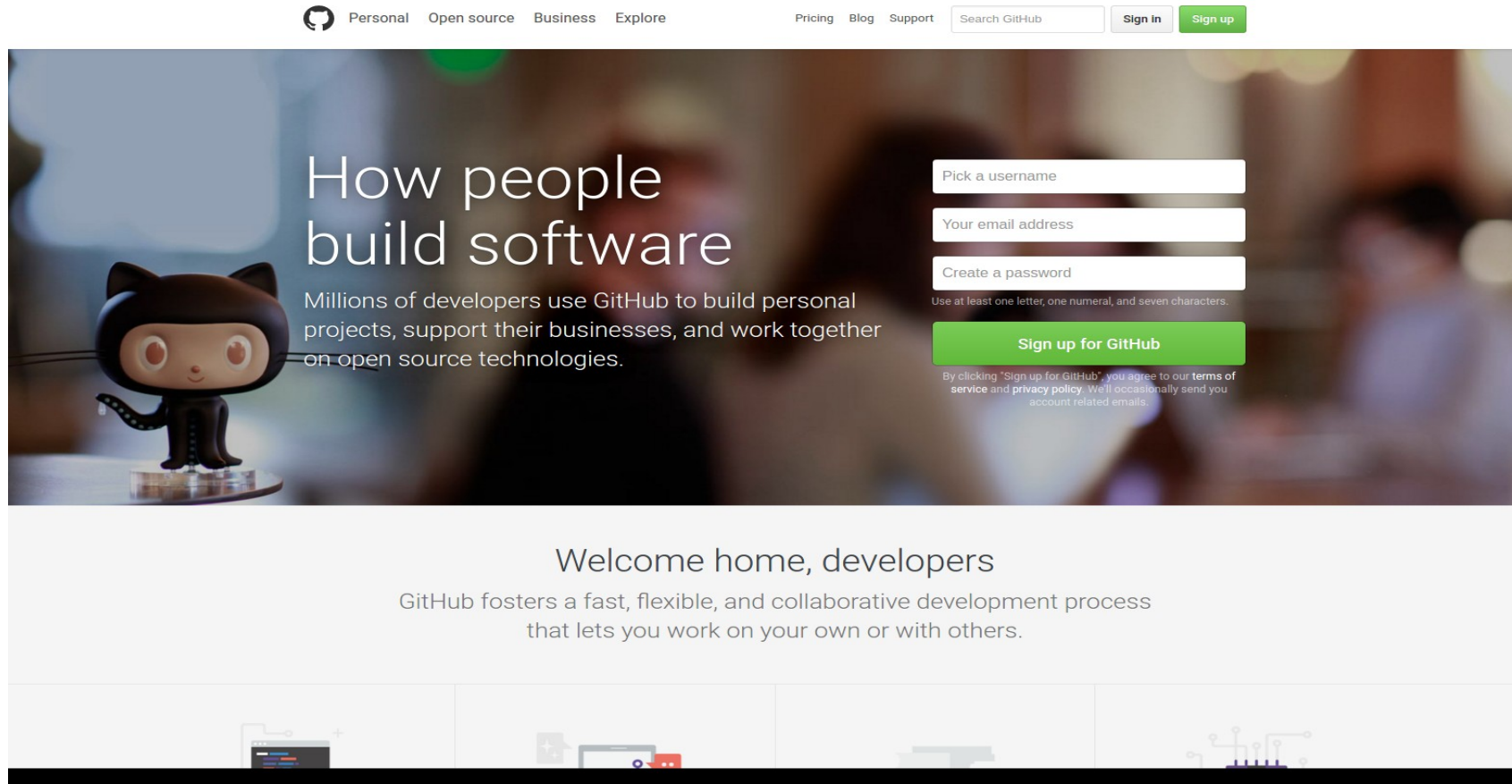
- `$ git diff`
- `$ git diff --staged`
- Git Diff in an External Tool
- <http://www.wiredforcode.com/blog/2011/06/04/git-with-meld-diff-viewer-on-ubuntu/>
- Viewing diffs between two checkouts
 - `Git log --oneline`
 - `git diff e3c0709 1c58e1e`

Branches

- Create a branch for something your not yet sure about:
 - \$ git branch <crazy experiment>
 - \$ git branch
 - \$ git checkout crazy experiment
- Start doing wacky stuff
- Commit wacky stuff
- It works? Want to keep it? Let's merge
- Git merges **into the current branch**:
 - git checkout master
 - git merge --no-ff branch

Using the distributed idea

- <https://github.com/>



The screenshot shows the GitHub homepage. At the top, there is a navigation bar with links for Personal, Open source, Business, and Explore. To the right of these links are links for Pricing, Blog, and Support, followed by a search bar labeled 'Search GitHub', a 'Sign in' button, and a green 'Sign up' button. The main content area features a large background image of a person's face. On the left, there is a GitHub Octocat mascot. The text 'How people build software' is prominently displayed. Below this, it says 'Millions of developers use GitHub to build personal projects, support their businesses, and work together on open source technologies.' On the right side of the main content area, there is a sign-up form with three input fields: 'Pick a username', 'Your email address', and 'Create a password'. Below the password field, there is a small note: 'Use at least one letter, one numeral, and seven characters.' A green 'Sign up for GitHub' button is positioned below the form. At the bottom of the main content area, there is a section titled 'Welcome home, developers' with the text 'GitHub fosters a fast, flexible, and collaborative development process that lets you work on your own or with others.' Below this, there is a row of four small icons representing different development tools or concepts.

Personal Open source Business Explore Pricing Blog Support Search GitHub Sign in Sign up

How people build software

Millions of developers use GitHub to build personal projects, support their businesses, and work together on open source technologies.

Pick a username

Your email address

Create a password

Use at least one letter, one numeral, and seven characters.

Sign up for GitHub

By clicking "Sign up for GitHub", you agree to our [terms of service](#) and [privacy policy](#). We'll occasionally send you account related emails.

Welcome home, developers

GitHub fosters a fast, flexible, and collaborative development process that lets you work on your own or with others.

Pushing from local to online repo

- Create new repository
- On Local machine:
 - \$ git remote add origin <https://github.com/FineWilms/newrepo.git>
- Check it:
 - \$ Git remote -v
- Push your local to the remote site:
 - \$ Git push origin master
- Now we have backup safely on server.
- Work on local machine, commit there, push to remote

Cloning: Recovery or collaboration

- Just shift-delete your thesis??
- Clone it: **COMPLETE COPY**
 - \$ git clone <path to remote repo>
- Collaboration:
 - \$ git fetch
 - \$ git merge
- Automated back-ups: crontab
- <https://help.ubuntu.com/community/CronHowto>
- Create shell script

Example of shell script:

- Create empty file run-git-push.sh and paste in working dir
- Contents:

```
#!/bin/bash
cd /home/jwilms/test/
git add .
git commit -m "auto update via crontab"
git push https://UserName:password@github.com/FineWilms/test master

#send yourself an email:
mail -s "Git Backup" fine.wilms@gmail.com < /dev/null
```

Example of \$ crontab -e

```
GNU nano 2.2.6                                     File: /tmp/crontab
# Edit this file to introduce tasks to be run by cron.
#
# Each task to run has to be defined through a single line
# indicating with different fields when the task will be run
# and what command to run for the task
#
# To define the time you can provide concrete values for
# minute (m), hour (h), day of month (dom), month (mon),
# and day of week (dow) or use '*' in these fields (for 'any').#
# Notice that tasks will be started based on the cron's system
# daemon's notion of time and timezones.
#
# Output of the crontab jobs (including errors) is sent through
# email to the user the crontab file belongs to (unless redirected).
#
# For example, you can run a backup of all your user accounts
# at 5 a.m every week with:
# 0 5 * * 1 tar -zcf /var/backups/home.tgz /home/
#
# For more information see the manual pages of crontab(5) and cron(8)
#
# m h  dom mon dow   command
MAILTO="fine.wilms@gmail.com"
0 * * * * /home/jwilms/test/touch.sh > /home/jwilms/test/cron.log 2>&1
0 * * * * /home/jwilms/test/git-backup-script.sh > /home/jwilms/test/cron2.log 2>&1
```