Version Control Flowchart

Do you need it?

Yes

Install Git

No

Yes you do

Today, we will use Linux.
Git

a distributed revision control and source code management (SCM) system
Git

a quite simple way to track changes in your code and share it with others
Git

a quite simple way to track changes in your *project* and share it with others
Jelmer van der Linde

I have been using git for about 3 4 5 years
Why use git?

1. Experiment
2. History
3. Collaborate
4. (Workflow)
How use git
How use git

Approved by Sanne (fourth year AI student)
How use git

```bash
Last login: Sat Jan 25 16:45:56 on ttys003
→ php git:(master) x ~/Desktop
→ Desktop git clone git@github.com:jelmervdl/tutorial.git
Cloning into 'tutorial'...
warning: You appear to have cloned an empty repository.
Checking connectivity... done
→ Desktop cd tutorial
→ tutorial git:(master) ls
→ tutorial git:(master) vim ./README.md
→ tutorial git:(master) x git add ./README.md
→ tutorial git:(master) x git commit -m "First commit"
[master (root-commit) 5c9b56d] First commit
 1 file changed, 2 insertions(+)
create mode 100644 README.md
→ tutorial git:(master) git push
Counting objects: 3, done.
Writing objects: 100% (3/3), 261 bytes | 0 bytes/s, done.
Total 3 (delta 0), reused 0 (delta 0)
→ To git@github.com:jelmervdl/tutorial.git
  * [new branch] master → master
→ tutorial git:(master) x git log
→ tutorial git:(master) x
```

Today

1. History
   git add, git log

2. Experiment
   git branch, git reset

3. Collaborate
   git push, git pull
Git: Concepts

- Working copy
- Staging
- Local repository
- Remote repository

- add
- commit
- checkout or reset
- checkout
- push
- pull
Git: Concepts
Git: Concepts

Staging
Git: Concepts

Local repository
Git: Concepts

- **Repository**
- **Staging**
- **Working copy**
Getting started

1. Create a new repository
   git init

2. Add files to staging
   git add modified-file.html

3. Commit staged files to repository
   git commit -m “Add my name”

4. See log of your repository
   git log
Getting started

index.html
“jelmer”
Getting started

Working copy

index.html
“jelmer”

Staging

Local repository

`git init`
Getting started

git add index.html
Getting started

```
index.html
"jelmer"
```

```
git commit -m "Add my name"
```
But before you begin…

Set up your name and email so anyone can see who did what. Especially if you want to use Github or Bitbucket ;)

```bash
git config --global --add user.name “Jelmer van der Linde”
git config --global --add user.email jelmer@ikhoefgeen.nl
```
Small experiments

1. Delete something
   `rm important-file.html`

2. Modify something
   E.g. `gedit modified-file.html`

3. Revert one file
   `git checkout -- modified-file.html`

4. Revert *everything* (that is: everything in your working copy)
   `git reset --hard`
Small experiments

- Working copy
  - index.html "sybren"
  - email.html "gmail"

- Staging

- Local repository
  - index.html "jelmer"
  - email.html "hotmail"
Small experiments

```
git checkout index.html
```
Small experiments

git reset --hard
Large experiments

1. Create a branch
   \texttt{git branch en-translation}

2. Switch to new branch
   \texttt{git checkout en-translation}

3. Modify something
   E.g. \texttt{gedit index.html}

4. Add and commit
   \texttt{git commit -a -m "Translate slide"}
Large experiments
Branches
Feature
Branches
Large experiments

A branch in your local repository
Large experiments

master

“First commit”
Large experiments

master

“First commit”

en-translation

git branch en-translation
Large experiments

```
git checkout en-translation
```
Large experiments

master

"First commit"

en-translation

"Translate slide"

`git commit -m "Translate slide"`
Large experiments

```
git checkout master
```
Large experiments

```
git commit -m "Spelling error"
```
Large experiments

```
git checkout en-translation
```
Large experiments

```
master
  "First commit"
    ↗
  "Translating slide"
    ↗
  "Spelling error"
    ↗
  "Grammar error"
```

git pull origin master
Large experiments

master

“First commit” → “Spelling error” → “Grammar error”

en-translation

“Translate slide” → “Merge from master”

git merge master
Large experiments

```
git checkout master
```

Diagram:
```
master -> "First commit" -> "Spelling error" -> "Grammar error"
```

```
en-translation
```
```
"Translate slide" -> "Merge from master"
```
Collaborate

1. Clone repository
   `git clone git@github.com: jelmervdl/tutorial.git`

2. Modify something
   `gedit index.html`

3. Add and commit
   `git commit -a -m "Update name"`

4. Push
   `git push`
Collaborate

Remote repository

index.html

“jelmer”
Collaborate

Working copy

Staging

Local repository

Remote repository

index.html "jelmer"

index.html "jelmer"

index.html "jelmer"

index.html "jelmer"

git clone git@github.com:jelmervdl/tutorial.git
Collaborate

Working copy

```
index.html
"taart"
```

Staging

Local repository

index.html
"jelmer"

Remote repository

index.html
"jelmer"

vim index.html
Collaborate

Working copy

index.html “taart”

Staging

Local repository

index.html “taart”

Remote repository

index.html “jelmer”

git commit -a -m “Update name”
Collaborate

Working copy

Staging

Local repository

Remote repository

index.html
“taart”

index.html
“taart”

index.html
“taart”

git push
That was a bit too simple…
Collaborate

Working copy

index.html "jelmer"

Staging

Local repository

Commit A
Commit B
Commit C

Remote

Commit A
Commit B
Commit C

git clone (or git pull)
Collaborate

Working copy

Staging

Local repository

Remote

index.html “taart”

Commit A

Commit B

Commit C

Commit D

Commit A

Commit B

Commit C

index.html “taart”

git commit -a -m “Commit D”
Collaborate

- Working copy
  - index.html
    - "taart"

- Staging

- Local repository
  - Commit A
    - Commit B
    - Commit C
    - Commit D

- Remote
  - Commit A
    - Commit B
    - Commit C
    - Commit D

---

Git push
Collaborate

Working copy 1
- index.html
  - "jelmer"

Local repository 1
- Commit A
- Commit B
- Commit C

Remote
- Commit A
- Commit B
- Commit C

Local repository 2
- Commit A
- Commit B
- Commit C

Working copy 2
- index.html
  - "jelmer"
Collaborate

Client 2: git commit -a -m "Commit D"
Collaborate

Working copy 1

Local repository 1

Commit A
Commit B
Commit C

Remote

Commit A
Commit B
Commit C
Commit D

Local repository 2

Commit A
Commit B
Commit C
Commit D

Working copy 2

index.html “jelmer”

Client 2: git push
Collaborate

Client 1: git pull
Collaborate: Conflict
Collaborate: Conflict

Working copy 1
- index.html
  - "appel"

Local repository 1
- Commit A
- Commit B
- Commit C
- Commit E

Remote
- Commit A
- Commit B
- Commit C
- Commit D

Local repository 2
- Commit A
- Commit B
- Commit C
- Commit D

Working copy 2
- index.html
  - "taart"
Collaborate: Conflict

Client 1: git pull
Collaborate: Conflict

Client 1: git push
Collaborate: Conflict

1. <<<<<<<< HEAD

3. Here is the original change.

4. ========

5. Here is the modified change.

6. >>>>>>>> 58326c301d09b58f3ac23d616e73f7b478424cc5
Collaborate: Share

Atlassian

Bitbucket

github
SOCIAL CODING
Clients

- **Git**
  
  **Linux:** `sudo apt-get install git`
  
  **Mac:** `brew install git`
  
  **Other:** [http://git-scm.com/downloads](http://git-scm.com/downloads)

- **GitHub (win/mac)**
  
  [http://windows.github.com/](http://windows.github.com/)
  
  [http://mac.github.com/](http://mac.github.com/)

- **SourceTree (win/mac)**
  
Support

Try git help or just ask Google and StackOverflow
Thank you for your attention.
Now you can...

- Put your projects in repositories
- Share your code
  E.g. https://github.com/jelmervdl
  Pretty good as Curriculum Vitae.
- Clone the Cover website
  https://bitbucket.org/cover-webcie/cover-php
  Fork it and collaborate with the AC/DCee.
Tips for the assignments

This slide is intentionally left empty. 😑