GIT

git (http://git-scm.com) ist ein Versionskontrollsystem für Software. Da Git auf der Command Line bedient wird, hier eine kurze Einführung in typische Befehle auf der Command Line:

Command Prompt ¶

- `cd xxx` ins Verzeichnis xxx wechseln.
- `cd ..` Verzeichnis zurück (oben).
- `dir` (Windows) oder `ls` (Mac/Linux) Inhalt des aktuellen Verzeichnisses ausgeben.
- `xxx /?` (Windows) oder `xxx -h` (Mac/linux) Hilfe zu Befehl xxx bekommen.
- `mkdir xxx` erstellt Verzeichnis mit Name xxx.
- `rmdir xxx` löscht Verzeichnis mit Name xxx.
- `del xxx` (Windows) oder `rm xxx` (Mac/Linux) löscht Datei mit Name xxx.
- `copy xxx yyy` (Windows) oder `cp xxx yyy` (Mac/Linux) kopiert Datei xxx zu yyy.
- `move xxx yyy` (Windows) oder `mv xxx yyy` (Mac/Linux) verschiebt Datei xxx nach yyy.

Aufgabe

Erstelle eine Datei mit dem Explorer, dann benutze die Command Line um die Datei zu kopieren, umzubenennen und zu löschen.

Windows:

```
copy old_filename new_filename
move old_filename new_filename
del filename
```

Mac/Linux:

```
cp old_filename new_filename
mv old_filename new_filename
rm filename
```

Git Konfigurieren

Git benötigt einen User-Name und eine Email-Addresse. Diese müssen nicht euren echten Namen entsprechen, sollten euch aber für eure Projektteilnehmer eindeutig benennen:

```
git config --global user.name "Bastian Bechtold"
git config --global user.email "bastian.bechtold@jade-hs.de"
```

Desweiteren sollte für Git ein guter Editor installiert werden. Für Windows ist eine einfache Lösung GitPad (https://github.com/github/gitpad), welches nach dem Ausführen den Windows-Default-Texteditor für Textdateien verwendet.

Auf Mac/Linux kann der gewünschte Text Editor mittels

```
git config --global user.editor "Name Des Editors"
```

gesetzt werden.

Git verwenden

Immer, wenn hier eine Kommandozeile mit ! beginnt, ist dies ein Command-Line-Befehl. In der tatsächlichen Command Line darf dieses ! **NICHT** eingeben werden

Ist git installiert, sollte auf der Command Line der Befehl `git` eine ähnliche Ausgabe wie das folgende ausgeben:

In [3]:

```
!git
```

```
usage: git [--version] [--help] [-C <path>] [-c name=value]
           [--exec-path[=<path>]] [--html-path] [--man-path] [--info-path]
           [-p|--paginate|--no-pager] [--no-replace-objects] [--bare]
           [--git-dir=<path>] [--work-tree=<path>] [--namespace=<name>]
           <command> [<args>]

The most commonly used git commands are:
   add        Add file contents to the index
   bisect     Find by binary search the change that introduced a bug
   branch     List, create, or delete branches
   checkout   Checkout a branch or paths to the working tree
   clone      Clone a repository into a new directory
   commit     Record changes to the repository
   diff       Show changes between commits, commit and working tree, etc
   fetch      Download objects and refs from another repository
   grep       Print lines matching a pattern
   init       Create an empty Git repository or reinitialize an existing one
   log        Show commit logs
   merge      Join two or more development histories together

   mv         Move or rename a file, a directory, or a symlink
   pull       Fetch from and integrate with another repository or a local branch
   push       Update remote refs along with associated objects
   rebase     Forward-port local commits to the updated upstream head
   reset      Reset current HEAD to the specified state
   rm         Remove files from the working tree and from the index
   show       Show various types of objects
   status     Show the working tree status
   tag        Create, list, delete or verify a tag object signed with GPG

'git help -a' and 'git help -g' list available subcommands and some
concept guides. See 'git help <command>' or 'git help <concept>'
to read about a specific subcommand or concept.
```

Um ein neues Projekt mit Versionskontrolle zu starten, einen leeren Ordner anlegen, und in ihm `git init` eingeben.

In [14]:

```
!mkdir NeuesProjekt
!cd NeuesProjekt
```

In [17]:

```
!git init
```

```
Initialized empty Git repository in /Users/bb/Projects/lecture-dalgo2015/termin 4/
NeuesProjekt/.git/
```

Zu jeder Zeit kann der aktuelle Zustand eines Git-Repositories mit `git status` abgefragt werden:

In [19]:

```
!git status
```

```
On branch master

Initial commit

nothing to commit (create/copy files and use "git add" to track)
```

Nun erstellen wir beispielshalber eine neue Datei:

In [20]:

```
%%writefile README.m
This is a new project.

At the moment, this is the only file.
```

```
Writing README.m
```

Nun, da eine neue Datei existiert, wird diese auch von `git status` erkannt:

In [21]:

```
!git status
```

```
On branch master

Initial commit

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

        README.m

nothing added to commit but untracked files present (use "git add" to track)
```

Git weiß aber noch nicht, was es mit dieser Datei anfangen soll. Sie ist daher noch als "untracked" markiert.

Um diese Datei der Versionskontrolle zu unterstellen, und in einen ersten Snapshot des Projekts (Commit) abzuspeichern, sind zwei Schritte nötig:

In [22]:

```
!git add README.m
```

In [23]:

```
!git status
```

```
On branch master

Initial commit

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)

        new file:   README.m
```

`git add` merkt eine oder mehrere Dateien vor, um ihren Zustand danach gesammelt als einen Commit (gespeicherter Snapshot des Projekts) abzuspeichern.

In [24]:

```
!git commit
```

```
[master (root-commit) 1bb4763] meine erste Datei
 1 file changed, 3 insertions(+)
 create mode 100644 README.m
```

In [25]:

```
!git status
```

```
On branch master
nothing to commit, working directory clean
```

`git commit` speichert den Zustand aller vorgemerkten Dateien als neuer Commit. Jeder Commit hat außerdem eine Nachricht, die die Änderungen in diesem Commit im Vergleich zum vorigen Commit beschreibt. `git commit` öffnet hierzu eine Textdatei in einem Texteditor, in dem die Nachricht eingetragen werden kann.

Im obigen Fall wurde als Nachricht "meine erste Datei" eingetragen.

Wird nun die per `git commit` gespeicherte Datei geändert, wird git dies bemerken:

In [27]:

```
%%writefile README.m
This is a new project.

At the moment, this is the only file.

Now this readme contains another, new, line, that is not yet part of a commit.
```

```
Overwriting README.m
```

In [28]:

```
!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:   README.m

no changes added to commit (use "git add" and/or "git commit -a")
```

Wie vorher können diese Änderungen mit `git add` und `git commit` gespeichert werden.

In [29]:

```
!git add README.m
!git commit
```

```
[master cc48d8c] weitere Änderungen
 1 file changed, 3 insertions(+), 1 deletion(-)
```

Weitere nützliche Befehle:

- `git log`, um eine Liste aller bisherigen Commits anzuzeigen.
- `git diff`, um alle Änderungen seit dem letzten Commit anzuzeigen.
- `git show`, um die Änderungen des letzten Commits anzuzeigen.

Git mit Github verwenden

Um mit Git und Github mit Kommilitonen zusammen zu arbeiten, muss ein Repository auf einem Server erstellt werden, welches von allen Projektteilnehmern als Datenaustausch verwendet wird.

Am Einfachsten geht dies auf Github. Wenn man sich dort einen Account erstellt hat, kann man dort (also auf dem Server) neue Repositories anlegen.

Um diese Repositories von einem lokalen, bestehenden, Git-Projekt aus verwenden zu können, muss der Server als "Remote" eingetragen werden. Die entsprechende Server-URL findet sich in der Github-Seitenleiste wenn man sich das Repository ansieht.

Mit `git remote add` kann ein solcher Server etwa für mein aktuelles Projekt eingetragen werden:

In [31]:

```
!git remote add origin https://github.com/bastibe/Dalgo-Projekt.git
```

Wobei "origin" hier ein frei wählbarer Name des Online-Repositories ist. Nun können alle bisherigen Commits mit `git push` hochgeladen werden:

In [38]:

```
!git push -u origin master
```

```
Counting objects: 6, done.
Delta compression using up to 4 threads.
Compressing objects: 100% (4/4), done.
Writing objects: 100% (6/6), 546 bytes | 0 bytes/s, done.
Total 6 (delta 1), reused 0 (delta 0)
To git@github.com:bastibe/Dalgo-Projekt.git
 * [new branch]      master -> master
Branch master set up to track remote branch master from origin.
```

Der Zusatz "-u origin master" ist nur beim ersten Mal nötig. Er sagt aus, dass von nun an immer by default die Remote namens "origin" und die neueste Version aller Dateien hochgeladen werden soll. Ist dies einmal getan, kann in der Zukunft immer einfach nur `git push` verwendet werden.

Andere Projektteilnehmer können nun per `git clone https://github.com/bastibe/Dalgo-Projekt.git` die aktuellste Version des Projekts als Git-Repository herunterladen. Dies hat dann auch immer schon automatisch die Remote und das "-u origin master" gesetzt.

Damit andere Projektteilnehmer ebenfalls `git push`en können, müssen sie in Github als "Collaborators" eingetragen werden. (Github-Projekt-Settings -> Collaborators).

Um gepushte Änderungen anderer Projektteilnehmer herunterzuladen, wird `git pull` verwendet:

(Hier wurde zwischen dem letzten `push` und jetzt ein neuer Commit erstellt.)

In [46]:

```
!git pull
```

```
From github.com:bastibe/Dalgo-Projekt
 * branch            master     -> FETCH_HEAD
Updating cc48d8c..f88da28
Fast-forward
 README.m | 4 +++-
 1 file changed, 3 insertions(+), 1 deletion(-)
```

In [53]:

```
!git status
```

```
On branch master
Your branch is up-to-date with 'origin/master'.
nothing to commit, working directory clean
```

In [54]:

```
!git log
```

```
commit f88da281141db69f4a289a41d3d302d2f6647a53
Author: Bastian Bechtold <basti@bastibe.de>

Date:   Tue Mar 31 19:26:37 2015 +0200

    Änderung eines anderen Projektteilnehmers

commit cc48d8ca6aa516270cc4b322e35091c7ae393a07
Author: Bastian Bechtold <basti@bastibe.de>
Date:   Tue Mar 31 19:09:05 2015 +0200

    weitere Änderungen

commit 1bb47636bf33178612cd78d818dc7237ce4b0a9c
Author: Bastian Bechtold <basti@bastibe.de>
Date:   Tue Mar 31 19:04:24 2015 +0200

    meine erste Datei
```

In [55]:

```
!git show
```

```
commit f88da281141db69f4a289a41d3d302d2f6647a53
Author: Bastian Bechtold <basti@bastibe.de>
Date:   Tue Mar 31 19:26:37 2015 +0200

    Änderung eines anderen Projektteilnehmers

diff --git a/README.m b/README.m
index 3497d48..e3dce22 100644
--- a/README.m
+++ b/README.m
@@ -2,4 +2,6 @@ This is a new project.

 At the moment, this is the only file.

-Now this readme contains another, new, line, that is not yet part of a commit.
\ No newline at end of file
+Now this readme contains another, new, line, that is not yet part of a commit.
+
+Now this contains even more text!
\ No newline at end of file
```

In [ ]:

```

```