1 Git auf Windows Installieren

1. Öffne https://git-scm.com
   - (Website stellt Buch zu Git zur Verfügung, Dokumentation etc.)
   - (repository von Git auf Github https://github.com/git/git)
2. Lade das Programm herunter
3. Installiere das Programm Notepad++
4. Starte die Installation
5. Ädjusting your PATH environment"
   Wähle: "Git from command line and also from 3rd-party software"
   ermöglicht das benutzen von Git mit Git Bash und cmd
6. Bei Choosing HTTPS transport backend standard lassen, also:
   Use the OpenSSL library ausgewählt lassen.
7. Bei Configuring the line ending conversions wähle:
   Checkout Windows-style, commit Unix-style line endings
8. Configuring the terminal emulator to use with Git Bash wähle:
   Use MinTTY (the default terminal of MSYS2).
9. Bei Configuring extra options
   so lassen wie es ist.

asdfasdfasdf

2 Git Konfigurieren und mit einem Repository verbinden

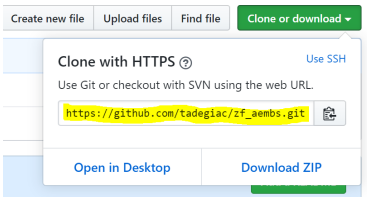
1. Windows CMD (Kommandozeilenprogramm) verwenden um zu Ordner navigieren, den man hochladen möchte.
   Mit dir kann man den Pfad und alle Inhalte des Ordners anzeigen lassen.

   mit cd verzeichnisname kann man in ein anderes Verzeichnis oder einen Ordner wechseln.

2. Git Benutzernamen Konfigurieren:
   git config –global user.name username

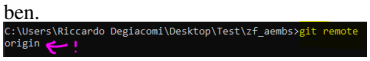
3. E-Mail angeben:
   git config –global user.email «your e-mail>"Jetzt ist es einem möglich sich mit einem Git Repository zu verbinden.

4. Gewünschtes Git-Repository öffnen und kopieren (Ctrl+C)



5. Zurück in der Kommandozeile
   (im richtigen Verzeichnis → Ordner für Projekt) mit dem Befehl:
   git clone
   Adresse des repositories einfügen.

   Nach dem Ausführen dieses Schrittes hat man jetzt einen neuen Ordner im Verzeichnis mit dem Namen des Git repositories.
6. Als nächstes sollte man in das Verzeichnis dieses Ordners wechseln (wieder mit cd). Im Verzeichnis kann man nun den befehl:
   git remote in der Kommandozeile eingeben.



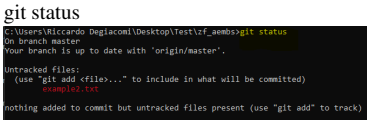
   Wenn alle bisherigen Schritte korrekt ausgeführt wurden, sollte jetzt origin in der Kommandozeile stehen. Mit git remot -v kann man die aliase anzeigen lassen.

   Jetzt ist man mit dem Repository verbunden und kann Dateine pullen und pushen.

3 Eine neue Datei erstellen und in das repository auf Github pushen

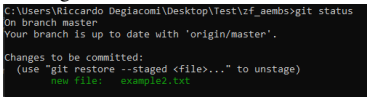
1. Um ein neues file zu erstellen kann man z.B:
   start notepad++ example.txt
   eingeben.

2. Schreibe etwas in den Texteditor und speichere die Datei
3. Tippe folgenden Befehl in die Konsole ein:
   git status



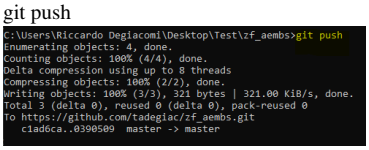
4. Um dem Programm mitzuteilen, dass wir diese Datei in das Repository comitten/laden möchten kann man diesen Befehl eingeben:
   git add example.txt
   (statt den namen der datei kann man auch einen ganzen ordner hochladen ;)

5. Nachtem enter gedrückt wurde weiss das Programm dass diese Datei neu ist und hochgeladen werden soll. Dieser Vorgang wird auch als staging bezeichnet. Wichtig ist, es ist noch ein anderer Befehl nötig um diese dann wirklich hochzuladen. Dies kann man sehen wenn man erneut git status eingibt.



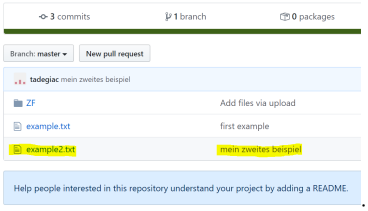
6. Um die Datei nun hoch zu laden verwendet man diesen Befehl:
   git commit -m "blabla"
   das -m ist dazu da dass nicht irgend ein komischer editor geöffnet wird den scheinbar niemand versteht. Das Zeugs in den Gänsefüsschen ist eine Art Kommentar den man zur Datei schreiben kann.

7. Nach dem Commit kann man mit folgendem Befehl die Datei ins repository laden:
   git push



   Wenn der push funktioniert hat sieht das ganz in etwa so aus:

   Die Änderungen im repository sehen so aus:

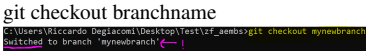


:

4 Einen Branch erstellen

1. Im Git master Verzeichnis folgendes eingeben:
   git branch branchname

   Dies erstellt einen branch. Dieser ist allerdings momentan nur lokal abgespeichert.
2. Als nächstes muss man noch in den neu erstellten Branch wechseln. Dies kann mit folgendem Befehl bewerkstelligt werden:
   git checkout branchname



   Wenn man möchte kann man beispielsweise die .txt Datei von Kap.3 ein wenig erweitern und abspeichern. Wenn alles funktioniert hat ist im Master nichts von der Änderung zu sehen, und der Branch ist auch nicht im Repository auffindbar.

3. Überprüfen ob branch bereits auf Repository:
   git branch -a

   Man kann zur Sicherheit auch im Repository nachschauen.

4. Branch auf repository pushen:
   git push –set-upsteram origin branchname

```
C:\Users\Riccardo Degiacomi\Desktop\Test\zf_aembs>git push
Enumerating objects: 4, done.
Counting objects: 100% (4/4), done.
Delta compression using up to 8 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 321 bytes | 321.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/tadegiac/zf_aembs.git
   c1ad6ca..0390509  master -> master
```

Nach dem pushen kann man überprüfen ob der erstellte branch auch wirklich im repository ist:

```
C:\Users\Riccardo Degiacomi\Desktop\Test\zf_aembs>git branch -a
* anotherbranch
  master
  mynewbranch
  omg_a_branch
  remotes/origin/HEAD -> origin/master
  remotes/origin/master
  remotes/origin/mynewbranch
```

Branch: mynewbranch ▾ zf_aembs / example2.txt

tadegiac mein erster branch mit veränderten inhalten

1 contributor

2 lines (2 sloc) | 86 Bytes

```
1  hello world
2  hello you! this is the world speaking!!! WTF do you actually want jackass?
```

5. Zurück zum Master wechseln:
   git checkout Master

5 Branch und Master vereinen

1. Möchte man die Teile die man im Branch erstellt hat wieder in den Master (oder einen anderen Branch) einfügen, so muss man in das Verzeichnis des Branches wechseln in den die Inhalte eingefügt werden sollen. Dies macht man wieder mit:
   git checkout branchname

```
C:\Users\Riccardo Degiacomi\Desktop\Test\zf_aembs>git branch -a
  anotherbranch
* master
  mynewbranch
  omg_a_branch
  remotes/origin/HEAD -> origin/master
  remotes/origin/master
  remotes/origin/mynewbranch
```

2. Ist man im Verzeichnis kann man die Dokumente wieder zusammenführen dies nennt man auch merge:
   git merge branchname

```
C:\Users\Riccardo Degiacomi\Desktop\Test\zf_aembs>git merge mynewbranch
Updating 0390509..83f7035
Fast-forward
 example2.txt | 3 ++-
 1 file changed, 2 insertions(+), 1 deletion(-)
```

3. add

```
C:\Users\Riccardo Degiacomi\Desktop\Test\zf_aembs>git add .
```

4. commit

```
C:\Users\Riccardo Degiacomi\Desktop\Test\zf_aembs>git commit -m "merged files"
On branch master
Your branch is ahead of 'origin/master' by 1 commit.
  (use "git push" to publish your local commits)

nothing to commit, working tree clean
```

5. push

```
C:\Users\Riccardo Degiacomi\Desktop\Test\zf_aembs>git push
Total 0 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/tadegiac/zf_aembs.git
   0390509..83f7035  master -> master
```

Auch im repo kann man nun die Änderungen sehen.

Branch: master ▾ zf_aembs / example2.txt

tadegiac mein erster branch mit veränderten inhalten

1 contributor

2 lines (2 sloc) | 86 Bytes

```
1  hello world
2  hello you! this is the world speaking!!! WTF do you actually want jackass?!
```

6 Lokalen Branch & Branch auf Repo löschen

1. Lokalen branch löschen (geht nicht solange man sich im Branch den man löschen möchte befindet >.<):
   git branch -d branchname
   Überprüfen in welchem branch man sich befindet.

```
C:\Users\Riccardo Degiacomi\Desktop\Test\zf_aembs>git branch -a
  anotherbranch
* master
  mynewbranch
  omg_a_branch
  remotes/origin/HEAD -> origin/master
  remotes/origin/master
  remotes/origin/mynewbranch
```

   Unbenötigte branches löschen.

```
C:\Users\Riccardo Degiacomi\Desktop\Test\zf_aembs>git branch -d omg_a_branch
Deleted branch omg_a_branch (was 0390509).

C:\Users\Riccardo Degiacomi\Desktop\Test\zf_aembs>git branch -d mynewbranch
Deleted branch mynewbranch (was 83f7035).
```

   Überprüfen ob branches lokal gelöscht wurden

```
C:\Users\Riccardo Degiacomi\Desktop\Test\zf_aembs>git branch -a
  anotherbranch
* master
  remotes/origin/HEAD -> origin/master
  remotes/origin/master
  remotes/origin/mynewbranch
```

2. branch auf Repository löschen:
   git push origin –delete branchname

```
C:\Users\Riccardo Degiacomi\Desktop\Test\zf_aembs>git push origin --delete mynewbranch
To https://github.com/tadegiac/zf_aembs.git
 - [deleted]         mynewbranch

C:\Users\Riccardo Degiacomi\Desktop\Test\zf_aembs>git branch -a
  anotherbranch
* master
  remotes/origin/HEAD -> origin/master
```

   Überprüfen ob im repo auch weg

Branch: master ▾ zf_aembs / example2.txt

Switch branches/tags

Find or create a branch...

Branches Tags

✓ master default

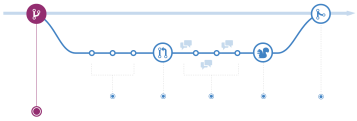
```
1  hello world
2  hello you! this is the world speaking!!!
```

7 Konflikte beim mergen

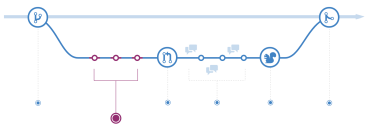
...todo...

8 Github Workflow

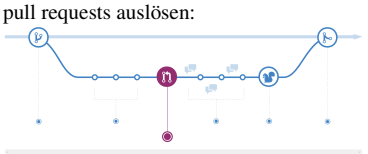
1. Einen Branch erstellen:



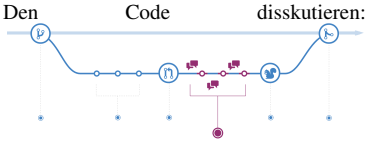
   Branches erstellt man um neue Idee für ein Projekt ausprobieren zu können, ohne gleich das ganze Projekt verändern zu müssen. Zusätzlich ermöglicht asynchrones zusammenarbeiten an verschiedenen Problemen des selben Projekts. Grundsätzlich sollte man nur vom Master branchen. Der branchname sollte deskriptiv für die Idee sein die im branch verfolg wird damit den anderen Usern klar ist wozu der branch dient. Branchnamen Beispiele: zf-aembs-kapitel-5, aembs-codebeispiele-ledtreiber-interrupts etc.
2. commits hinzufügen:



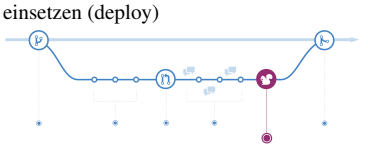
   Sobald man einen neuen Branch erstellt hat, beginnt man für gewöhnlich an seiner Idee zu arbeiten. Grundsätzlich sollte man nach jeder signifikanten Änderung einen commit machen. Jeder commit wird von git separat aufgeführt. Somit sind die einzelnen Arbeitsschritte als commits aufgeführt. Dies ist äusserst nützlich da so der Arbeitsfortschritt aufgezeichnet wird. Weiterhin kann man zu jedem commit einen Kommentar schreiben der ebenfalls hinzugefügt wird. So können die einzelnen Teammitglieder nachvollziehen was in welchem Arbeitsschritt gemacht wurde. Jeder commit wird als separate Änderung gespeichert. Falls Probleme auftreten kann man einzelene commits Rückgängig machen und so "debuggen".
3. pull requests auslösen:



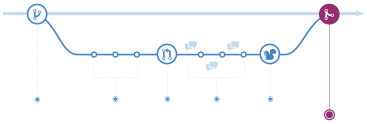
   Wenn ein pullrequest akzeptiert wird, wird der master mit dem branch gemerged. Dies bedeutet, dass jeder der am Projekt arbeitet die Änderungen die im master vorgenommen würden sehen kann. Dies dient dazu einen Diskurs auszulösen. Jeder kann die Änderungen betrachte und noch seinen Senf dazugeben. Pullrequests sind nützlich für opensource Projekte mit öffentlichen repositories. Sie helfen ein codereview zu starten und eine disskusion über vorgeschlagenen Änderungen auszulösen. Es besteht die Möglichkeit pull requests mit bestimmten Schlüsselwörtern (issues) zu verknüpfen. Mehr dazu hier: https://help.github.com/en/github/managing-your-work-on-github/linking-a-pull-request-to-an-issue
4. Den Code disskutieren:



   Nachdem ein pullrequest ausgelöst wurde und die damit einhergehende Disskusion, kann man vortlaufend neue commits machen und pullrequests. Das feedback etc. ist ebenfalls ersichtlich und dient dem Überblick. Pullrequests werdne in Markdown geschrieben (https://de.wikipedia.org/wiki/Markdown)
5. einsetzen (deploy)



   todo ?
6. merge:



Nachdem die Änderungen akzeptiert wurden kann man diese im Master einbetten. Die pullrequests werden protokolliert, so dass diese nach dem merge noch existieren. Dies ermöglicht das Nachvollziehen und Verstehen der enizelnen Entscheidunen die in der Vergangenheit getroffen wurden.