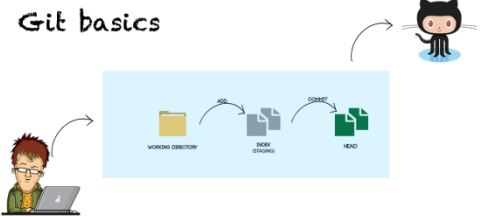


# Leittext-Aufgabe: Git Basics

## Ausgangslage:

Versionsverwaltungssysteme werden verwendet, um Änderungen an Dateien zu verfolgen. Dabei werden die Dateien in sogenannten Repositories (engl. Behälter, Aufbewahrungsorte) abgelegt. Typischerweise werden Versionsverwaltungssysteme in der Softwareentwicklung eingesetzt, vor allem dann, wenn mehrere Entwickler an derselben Software arbeiten. Bekannte Versionsverwaltungssysteme sind Git, CVS oder Subversion. In diesem Modul werden wir uns mit Git auseinandersetzen.



Zeitbedarf:

4 Lektionen

Hilfsmittel:

- Git Bash oder Git CMD
- Editor

Quelle: <http://devopscube.com/git-basics-every-developer-and-administrator-should-know/>

Weitere Quellen:

Git Tutorial und eine Git Schulung (Atlassian):

<https://www.atlassian.com/git/tutorials/setting-up-a-repository/git-init/>  
<https://www.atlassian.com/git/tutorials/learn-git-with-bitbucket-cloud/>

Methode/Sozialform:

Partnerarbeit

Kompetenzen:

- 5.1 Ich kenne den Nutzen eines Versionsverwaltungssystems und kann dieses für die Verwaltung einer Software einrichten.
- 5.2 Ich kann ein lokales Versionsverwaltungssystem mit einem zentralen System verbinden (Remote-, Cloud-Repository).
- 5.3 Ich kenne die täglichen Aufgaben die bei der Verwaltung einer Software anstehen können und kann diese bearbeiten und professionell erledigen.
- 5.4 Ich kenne Organisationsmittel wie Master und Branch und kann für eine Softwareänderung einen neuen Branch eröffnen, sowie meine Änderungen darin verwalten.
- 5.5 Ich kann durchgeführte Änderungen mit einem Commit-Befehl in die Versionsverwaltung einchecken und die Commit-Nachricht entsprechend formulieren.

Kompetenzstufen:

1	2	3	4
---	---	---	---

- 1 Ich habe die Begriffe und grundlegende Konzepte verstanden. Ich kann einfache Handlungen umsetzen und grundlegende Fragen beantworten. (Fortgeschrittener Anfänger)
- 2 Ich habe eine genügende Kompetenzstufe erreicht. Ich kann die Kompetenzen unter idealen Bedingungen anwenden und zeigen, sowie umfassend Auskunft geben. (Kompetenter Anfänger)
- 3 Ich habe eine gute Kompetenzstufe erreicht. Ich kann die Kompetenzen flexibel auch unter erschwerten Bedingungen zeigen. (Profi)
- 4 Ich habe eine sehr hohe Kompetenzstufe erreicht. Ich kann die Kompetenzen reflektieren und bewusst einsetzen. Ich kann anspruchsvolle Aufgaben mit Unbekannten zu lösen. (Experte)

nichts selektiert

Noch nicht bearbeitet, keine oder kaum Kompetenz vorhanden. (Neuling)

## 1 Auftrag – Git lokal anwenden

Wir werden wir ein eigenes HelloWorld-Projekt erstellen und den simplen **Git-Workflow** kennen zu lernen. Für den Auftrag verwenden Sie die Git Bash, oder Git CMD.

**Arbeiten Sie für diesen Auftrag zu zweit auf einem Rechner!**



### 1.1 Repository erstellen

Erstellen Sie in Ihrem Home-Verzeichnis (im Modul M426) ein Projekt-Verzeichnis namens „hello-world“:

```
H:\M426\Git>mkdir hello-world
H:\M426\Git>cd hello-world
```

Führen Sie nun folgenden Befehl aus um ein Repository zu erstellen:

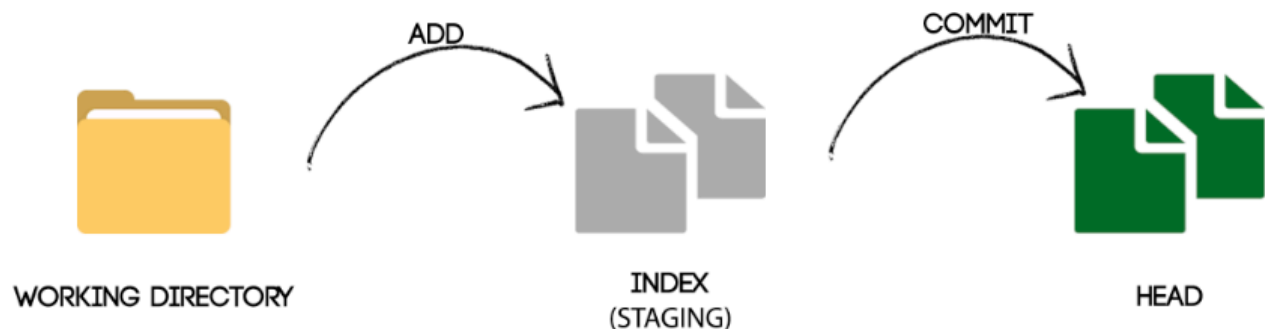
```
H:\M426\Git\hello-world>git init
Initialized empty Git repository in D:/GitHub/hello-world/.git/
```

Jedes Git Repository hat drei Bäume. Ein Arbeitsverzeichnis, einen Index und einen Head.

**Arbeitsverzeichnis:** enthält die aktuellen Dateien des Projekts.

**Index:** ist die Staging-Area in welche die Projekt-Dateien hinzugefügt (add) werden, welche für das Commit bereit sind.

**Head:** enthält die Referenzen zu vorherigen Commits.



### 1.2 Dateien zur Staging Area hinzufügen (add)

Wenn im Projekt neue Dateien erstellt werden, müssen diese zur Staging Area hinzugefügt werden. Wir erstellen nun in unserem Verzeichnis eine readme.txt Datei, welche wir zu Staging Area hinzufügen.

Erstellen Sie eine Datei:

```
H:\M426\Git\hello-world>echo "# hello-world" >> readme.txt
```

Überprüfe:

```
H:\M426\Git\hello-world >ls
Verzeichnis: H:\M426\Git\hello-world
Mode                LastWriteTime         Length Name
----                -
-a----             13.09.2016   14:38           32 readme.txt
```

Führen Sie folgenden Befehl aus, um die Datei zur Staging Area hinzuzufügen:

```
H:\M426\Git\hello-world>git add readme.txt
```

Mit **git status** können wir die Staging Area überprüfen:

```
H:\M426\Git\hello-world>git status
On branch master

Initial commit

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

        new file:   readme.txt
```

Sollen alle Dateien des Projektes hinzugefügt werden, wird folgender Befehl ausgeführt:

**git add \***

### 1.3 Neue Änderungen zum Repository hinzufügen (commit):

```
H:\M426\Git\hello-world> git commit -m "first commit"
[master (root-commit) 461eda0] first commit
1 file changed, 1 insertion(+)
create mode 100644 readme.txt
```

Einmal committed, kann eine Versionierung im lokalen Repository stattfinden. Wir können mit Git somit lokal arbeiten ohne ständig 'online' zu sein.

Unser nächster Schritt ist es nun ein entferntes, zentralisiertes Repository anzuwenden. Wir werden hierfür GitHub verwenden. GitHub bietet öffentliche (kostenlose), wie auch private (kostenpflichtige) Repositories an. Es gibt sogar eine Education-Version, mit welcher Sie kostenlos private Repositories erstellen können!

Auftrag – Git remotely anwenden

Arbeiten Sie für diesen Auftrag weiterhin zu zweit auf demselben Rechner!

# Git basics



## 1.4 Remote Repository erstellen

Für das weitere Vorgehen brauchen wir einen Account bei GitHub!

<https://education.github.com/>

Registrieren Sie sich bei GitHub und loggen Sie sich ein.

Erstellen Sie nun ein neues Repository (public) mit dem Namen „hello-world“. Sobald Sie das Erstellen des Repositories bestätigen, werden Ihnen verschiedene Möglichkeiten aufgezeigt, wie Sie mit dem erstellten Repository fortfahren können:

1. Quick setup — if you've done this kind of thing before
2. ...or create a new repository on the command line
3. ...or push an existing repository from the command line
4. ...or import code from another repository

## 1.5 Lokales mit Remote Repository verbinden

Wir wählen die Möglichkeit 3, **git remote add...**, aus, um unser lokales Repository mit dem neu erstellen, also remote Repository auf GitHub, zu verbinden:

```
H:\M426\Git\hello-world> git remote add origin https://github.com/<benutzername>/hello-world.git
```

Mit **git push**... updaten wir die entfernten Referenzen mit den lokalen.  
In diesem Schritt werden die Login-Daten (user/pw) verlangt:

```
H:\M426\Git\hello-world> git push -u origin master
Username for 'https://github.com': <benutzername>
Password for 'https://<benutzername>@github.com':
Counting objects: 3, done.
Writing objects: 100% (3/3), 222 bytes | 0 bytes/s, done.
Total 3 (delta 0), reused 0 (delta 0)
To https://github.com/<benutzername>/hello-world.git
 * [new branch]      master -> master
Branch master set up to track remote branch master from origin.
```

## 1.6 Staging Area untersuchen

Ändern Sie den Inhalt der readme.txt Datei (lokal) und überprüfen Sie die Staging Area:

```
H:\M426\Git\hello-world> git status
On branch master
Your branch is up-to-date with 'origin/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.txt

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

Fügen Sie die modifizierte Datei zu Staging Area:

```
H:\M426\Git\hello-world> git add readme.txt
```

Überprüfen Sie die Staging Area:

```
H:\M426\Git\hello-world> git status
On branch master
Your branch is up-to-date with 'origin/master'.
Changes to be committed:
  (use "git reset HEAD <file>..." to unstage)

        modified:   readme.txt
```

Committen Sie die Datei und überprüfen Sie nochmals die Staging Area:

```
H:\M426\Git\hello-world> git commit -m "second commit"
[master 523a4d4] second commit
1 file changed, 1 insertion(+), 1 deletion(-)
```

```
H:\M426\Git\hello-world> git status
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 directory clean
```

Publizieren Sie Ihre lokalen Änderungen auf dem Remote Repository (GitHub):

```
H:\M426\Git\hello-world> git push -u origin master
Counting objects: 3, done.
Writing objects: 100% (3/3), 263 bytes | 0 bytes/s, done.
Total 3 (delta 0), reused 0 (delta 0)
To https://github.com/<benutzername>/hello-world.git
 461eda0..523a4d4 master -> master
Branch master set up to track remote branch master from origin.
```

## 1.7 HelloWorld.java im Remote Repository ablegen

Wir werden nun eine HelloWorld.java Datei erstellen, in unserem lokalen Repository ablegen und danach auf GitHub publizieren:

Erstellen Sie mit Hilfe eines Editors eine neue leere HelloWorld Klasse:

```
public class HelloWorld {

}
```

Legen Sie diese Klasse ins selben Verzeichnis, wie die readme.txt Datei:

```
H:\M426\Git\hello-world> ls
HelloWorld.java readme.txt
```

Überprüfen Sie die Staging Area:

```
H:\M426\Git\hello-world> git status
On branch master
Your branch is up-to-date with 'origin/master'.
Untracked files:
  (use "git add <file>..." to include in what will be committed)

    HelloWorld.java

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

Fügen Sie die Datei in die Staging Area, committen und publizieren Sie sie:

```
H:\M426\Git\hello-world> git add HelloWorld.java
```


```
H:\M426\Git\hello-world> git status
On branch master
Your branch is up-to-date with 'origin/master'.
Changes to be committed:
  (use "git reset HEAD <file>..." to unstage)

    new file:   HelloWorld.java
```

```
H:\M426\Git\hello-world> git commit -m "HelloWorld included"
[master 2e0c3ad] HelloWorld included
1 file changed, 3 insertions(+)
create mode 100644 HelloWorld.java
```

```
H:\M426\Git\hello-world> git push origin master
Counting objects: 3, done.
Delta compression using up to 8 threads.
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 315 bytes | 0 bytes/s, done.
Total 3 (delta 0), reused 0 (delta 0)
To https://github.com/<benutzername>/hello-world.git
523a4d4..2e0c3ad master -> master
```

Überprüfen Sie die Änderungen auf GitHub:


[lcavuoti / hello-world](#)


Unwatch 1
Star 0
Fork 0



Code
Issues 0
Pull requests 0
Wiki
Pulse
Graphs
Settings


No description or website provided. — Edit

3 commits
1 branch
0 releases
1 contributor

Branch: master
New pull request
Create new file
Upload files
Find file
Clone or download


[lcavuoti HelloWorld included](#)
Latest commit 2e0c3ad 2 hours ago

 <a href="#">HelloWorld.java</a>	HelloWorld included	2 hours ago
 <a href="#">readme.txt</a>	second commit	2 hours ago

 [readme.txt](#)

```
# hello-world GitHub
```

## 2 Auftrag - Git auf 2. Rechnern anwenden

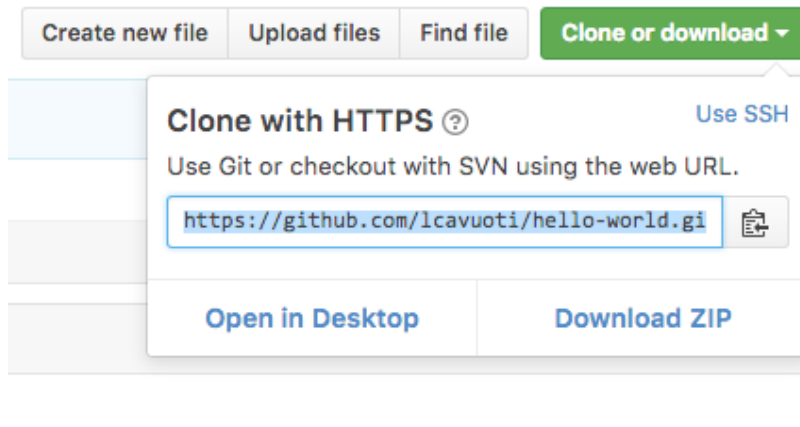
**Wechseln Sie für diesen Auftrag zusammen auf den zweiten Rechner!**



### 2.1 GitHub-Projekt klonen

Wir möchten nun dieses GitHub Projekt auf dem Rechner Ihrer Kollegin oder Ihres Kollegen klonen.

Benutzen Sie hierfür die entsprechende URL:



Geben Sie nun folgenden Befehl auf dem **2. Rechner** im entsprechenden Verzeichnis ein:

```
H:\M426\Git_2>git clone https://github.com/<benutzername>/hello-world.git
Cloning into 'hello-world'...
remote: Counting objects: 9, done.
remote: Compressing objects: 100% (4/4), done.
remote: Total 9 (delta 0), reused 9 (delta 0), pack-reused 0
Unpacking objects: 100% (9/9), done.
Checking connectivity... done.
```

So, das Projekt haben Sie nun auch auf dem 2. Rechner:

```
H:\M426\Git_2> ls
hello-world
H:\M426\Git_2> cd hello-world
D:\ GitHub_2\hello-world> ls
HelloWorld.java readme.txt
```

Jeder kann nun auf seinem eigenen Rechner arbeiten und mit...

...**git push** - Änderungen speichern

...**git pull** - Änderungen übernehmen

### 2.2 Änderungen übernehmen

Da der zweite Rechner nun aktuell ist, werden Sie keine Änderungen übernehmen können:

```
D:\ GitHub_2\hello-world> git pull
Already up-to-date.
```

Wir werden nun die Java Datei HelloWorld um einen Konstruktor erweitern, damit wir diese Änderungen remote auf dem GitHub Repository publizieren und somit auf dem ersten Rechner übernehmen können.



**Änderung im HelloWorld:**

```
public class HelloWorld {  
  
    public HelloWorld() {  
  
    }  
  
}
```

**git status**

```
D:\ GitHub_2\hello-world> git status  
On branch master  
Your branch is up-to-date with 'origin/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:   HelloWorld.java  
  
no changes added to commit (use "git add" and/or "git commit -a")
```

**git add/status**

```
D:\ GitHub_2\hello-world> git add HelloWorld.java  
D:\ GitHub_2\hello-world> git status  
On branch master  
Your branch is up-to-date with 'origin/master'.  
Changes to be committed:  
  (use "git reset HEAD <file>..." to unstage)  
  
        modified:   HelloWorld.java
```

**git commit**


```
D:\ GitHub_2\hello-world> git commit -m "constructor added"  
[master d16e77c] constructor added  
1 file changed, 4 insertions(+)
```

**git push origin master**




```
D:\ GitHub_2\hello-world> git push origin master  
Counting objects: 3, done.  
Delta compression using up to 8 threads.  
Compressing objects: 100% (3/3), done.  
Writing objects: 100% (3/3), 332 bytes | 0 bytes/s, done.  
Total 3 (delta 0), reused 0 (delta 0)  
To https://github.com/<benutzername>/hello-world.git  
2e0c3ad..d16e77c  master -> master
```

Die Änderungen sind nun auf GitHub publiziert:

Branch: master [hello-world](#) / [HelloWorld.java](#) [Find file](#) [Copy path](#)

 **lcavuoti** constructor added d16e77c 4 minutes ago

1 contributor

8 lines (4 sloc) | 71 Bytes [Raw](#) [Blame](#) [History](#)   

```
1 public class HelloWorld {
2
3     public HelloWorld() {
4
5     }
6
7 }
```

Fehlt nur noch das Aktualisieren der Dateien auf dem ersten Rechner:

```
D:\GitHub_2\hello-world> git pull
remote: Counting objects: 3, done.
remote: Compressing objects: 100% (3/3), done.
remote: Total 3 (delta 0), reused 3 (delta 0), pack-reused 0
Unpacking objects: 100% (3/3), done.
From https://github.com/lcavuoti/hello-world
 2e0c3ad..d16e77c master    -> origin/master
Updating 2e0c3ad..d16e77c
Fast-forward
 HelloWorld.java | 4 ++++
 1 file changed, 4 insertions(+)
```

### 3 Auftrag – Git Übersicht

Gehen Sie auf die Seite: <https://try.github.io>



Laden Sie Git-It herunter und vertiefen Sie die Git-Befehle der Tutorials und notieren Sie sich was Sie gerade getan haben! Erstellen Sie für sich eine Dokumentation der Git-Befehle welche Sie später nutzen können um nachzuschauen!

## Notizen

[illegible]