



# Git / GitHub

---



SEW  
DI Thomas Helml

SJ 2017/18



# Inhalt

---

- IT Motivation
- IT Arten von VCS
- IT Git Grundlagen
- IT GitHub
- IT Git Shell Grundlagen



# Wozu VCS?

---

- ① VCS (Version Control System)
  - ① dt. Versionsverwaltungssystem
  - ① Erlaubt Teamarbeit – mehrere Leute arbeiten an der gleichen Codebasis
  - ① Art „Zeitmaschine“ – Backup der Entwicklungsschritte
  - ① Ermöglichen CI (Continuous Integration)



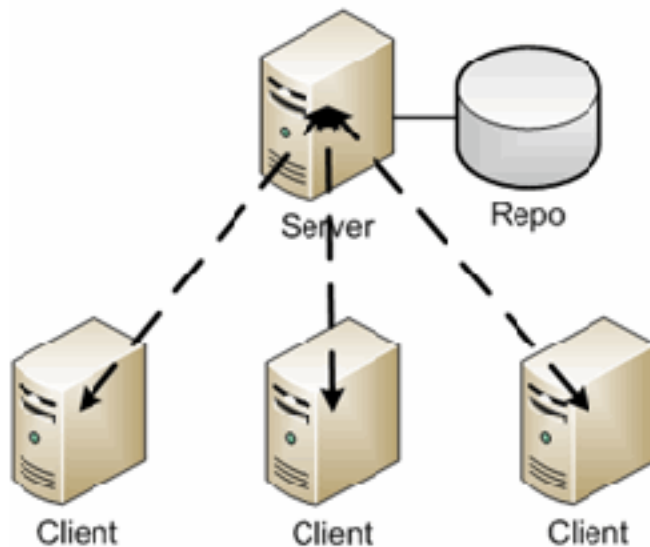
# Arten von VCS

---

- IT Local
  - IT RCS (Mac früher)
- IT Centralized
  - IT Subversion /SVN)
  - IT CVS
  - IT Perforce
- IT Distributed
  - IT Git
  - IT Mercurial

# Arten von VCS

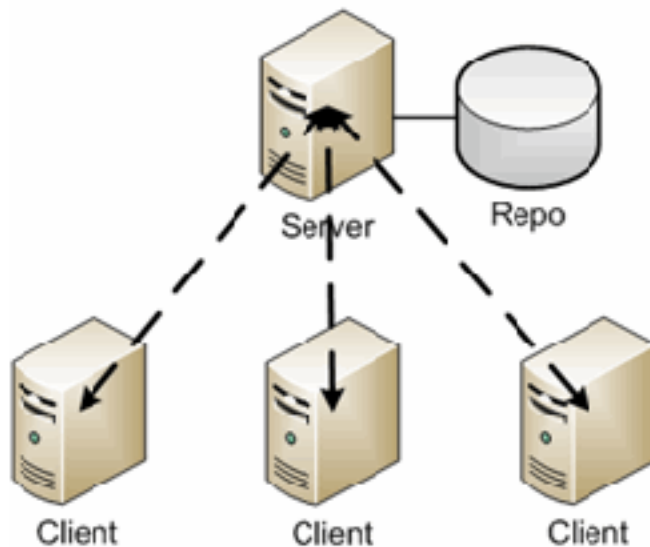
## Centralized



- IT Funktionsweise
- IT 1 Server:
  - IT Speichert alle Versionen der Files
- IT viele Clients
  - IT Client checkt lokale Kopie aus
  - IT Bearbeitet sie
  - IT Pusht sie an Server zurück

# Arten von VCS

## Centralized



### IT Vorteile

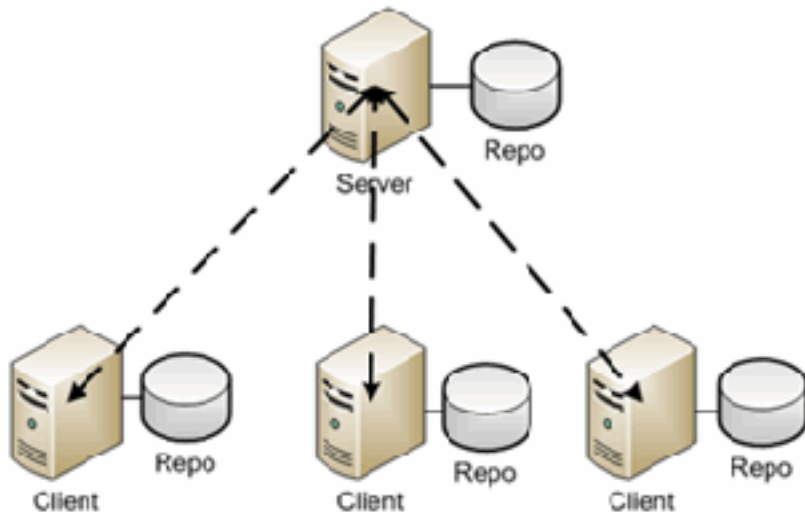
- IT schaut simpel aus, ist simpel
- IT gute Verbreitung, gute IDE Integration
- IT funktioniert

### IT Nachteile

- IT Man kann nicht offline commiten
- IT Man kann nicht mehrere Repositories in ein Projekt einbinden
- IT Schwierigkeiten bei großen Teams (Open Source)

# Arten von VCS

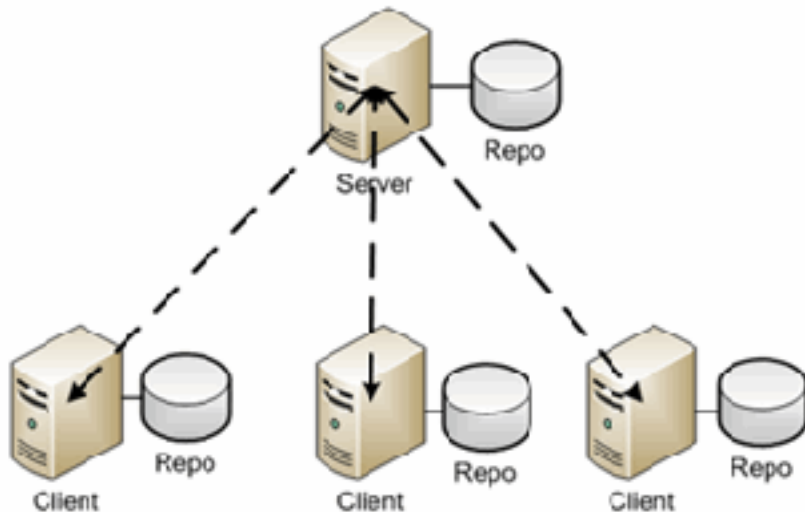
## Distributed



- ① Ein (mehrere) Server
- ① Client hat volle Kopie der Repository lokal, die mit Server gemerged werden können

# Arten von VCS

## Distributed



### ⓘ Vorteile:

- ⓘ Kein Netzwerk notwendig/  
Offline möglich
- ⓘ Private Arbeit möglich ohne  
Muss des publishen
- ⓘ Kann beliebig viele Repos  
einbinden
- ⓘ Kein Single Point of Failure
- ⓘ Zentrale Kontrolle von „Server  
Releases“ möglich
- ⓘ Client hat volle Kopie der  
Repository lokal

### ⓘ Nachteile:

- ⓘ Schwieriger zu lernen
- ⓘ Komplexerer Workflow

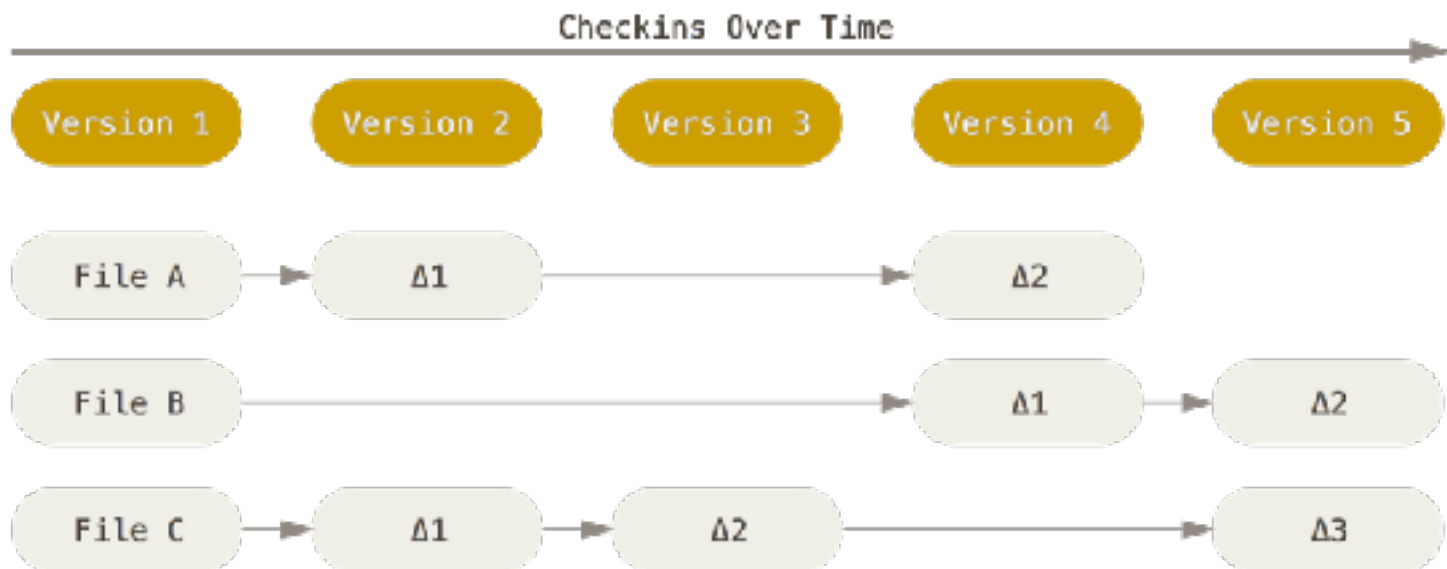




# Arten von VCS

## IT Interne Datenrepräsentation – SVN

IT Änderungen über die Zeit werden gespeichert

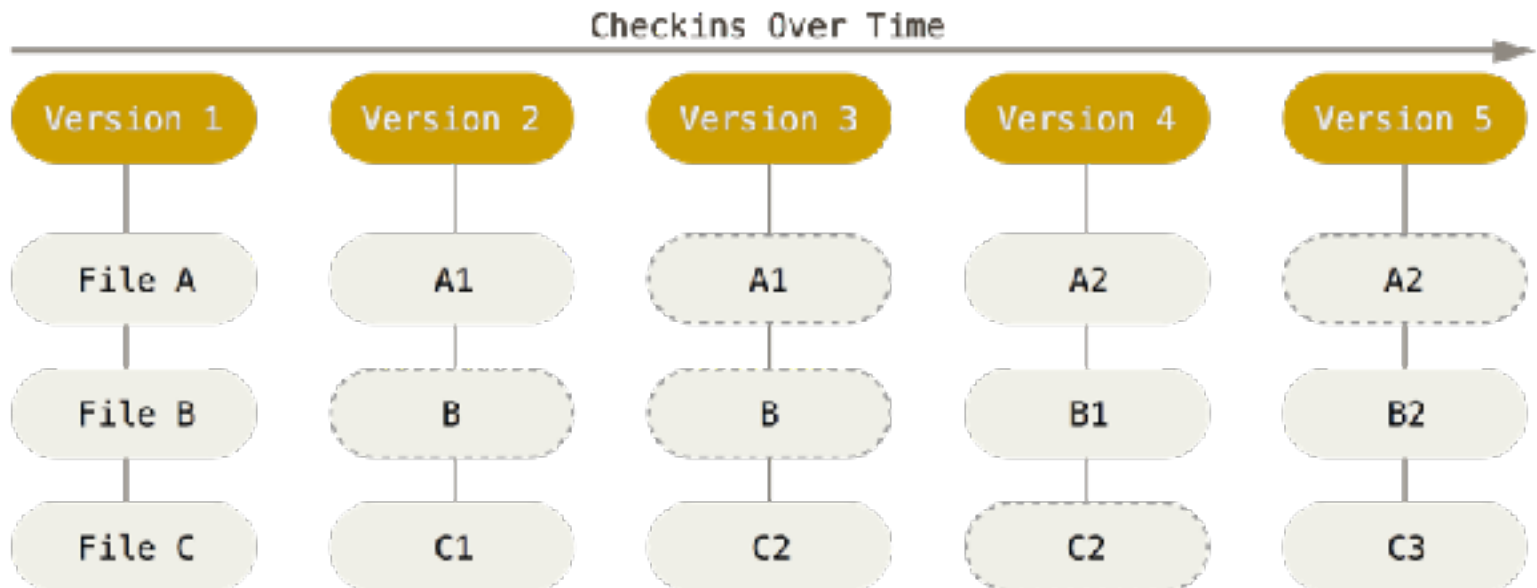




# Arten von VCS

## IT Interne Datenrepräsentation – git

### IT Daten sind Stream von „Snapshots“










# Git

---

## Geschichte

-  Linux Kernel – relativ großes Projekt
-  1991-2002: kein VCS, Archives + Patches
-  Ab 2002: BitKeeper
-  2005: Bruch, BitKeeper wird kostenpflichtig
-  Linus Torwards entwickelt Git



# Die 3 Hauptzustände

---

IT Git hat 3 Hauptzustände, in denen sich die Files befinden können:

IT Committed

IT Daten sind sicher in lokaler Datenbank (Repository) gespeichert

IT Modified

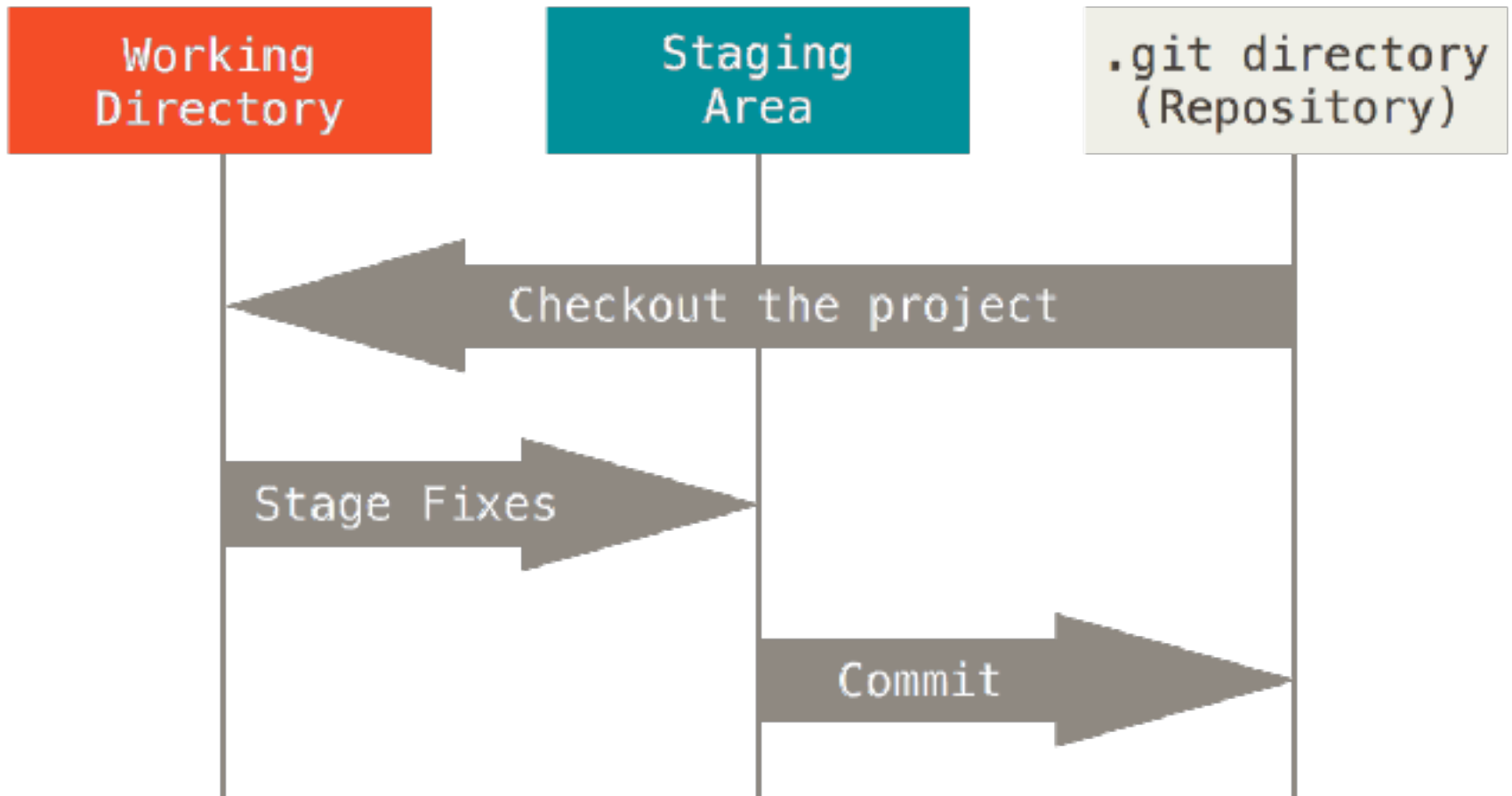
IT Daten sind geändert worden, aber noch nicht in Repo gespeichert (committed)

IT Staged

IT Modifizierte Daten sind markiert, dass sie beim nächsten commit in Repo gespeichert werden



# Die 3 Hauptzustände





# Die 3 Hauptzustände

## ① .git Ordner:

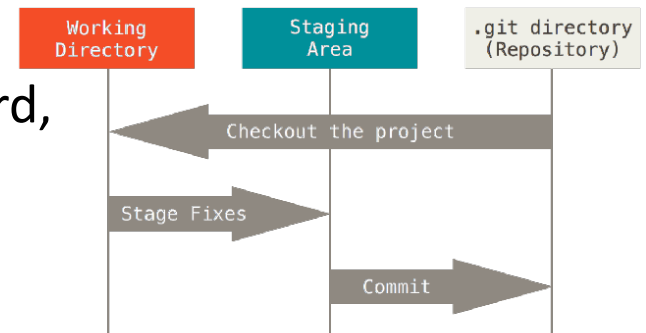
- ① hier speichert Git seine Daten (lokale Repo)

## ② Staging Area

- ① Datei im .git Ordner, in der gespeichert wird, was beim nächsten commit passieren soll
- ① Wird öfters auch als „Index“ bezeichnet

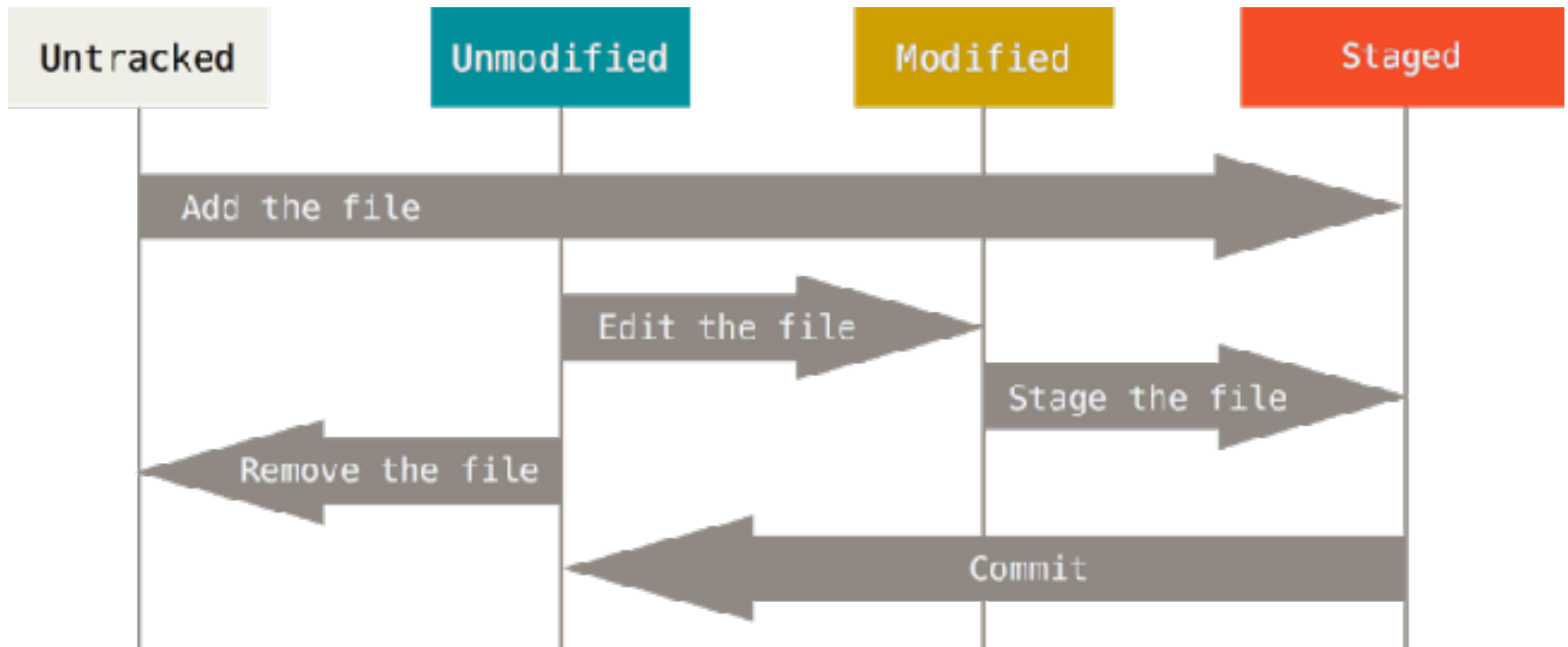
## ③ Working Directory:

- ① Checkout von einer Version des Projekts, werden aus Repo herausgezogen (pull)
- ① Dateien werden hier bearbeitet





# Zustände der Files





# Zustand der Files

## IT Untracked

- IT Dateien, die nicht von git beobachtet werden

## IT Unmodified

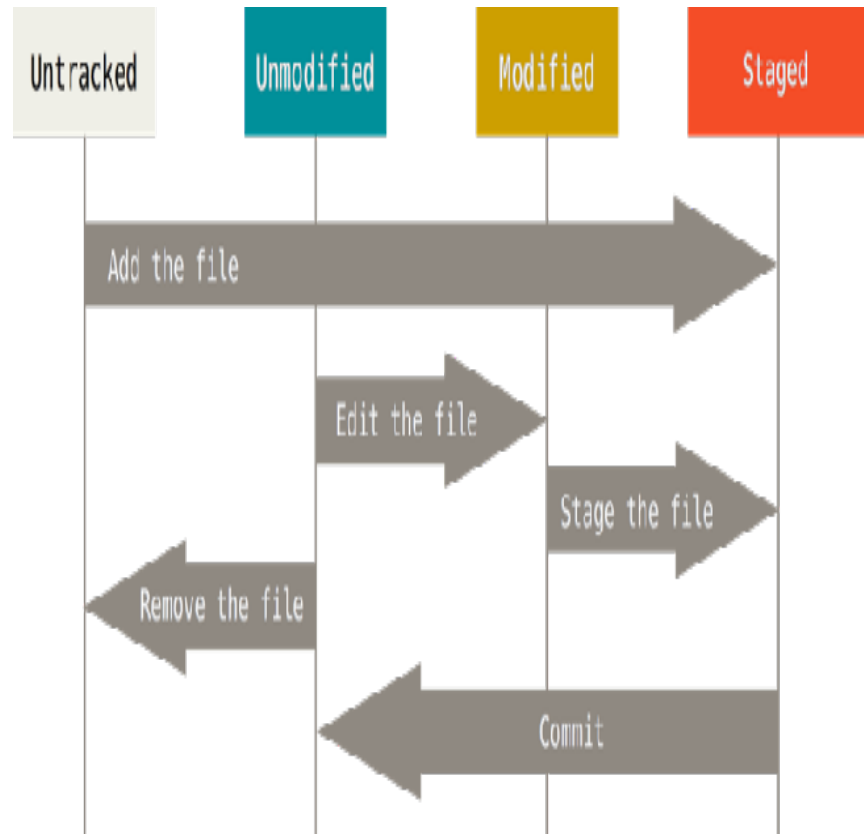
- IT Datei wurde nicht geändert, entspricht also dem ausgecheckten Zustand

## IT Modified

- IT Datei hat sich geändert

## IT Staged

- IT Datei wird bei Commit in die Repository kopiert





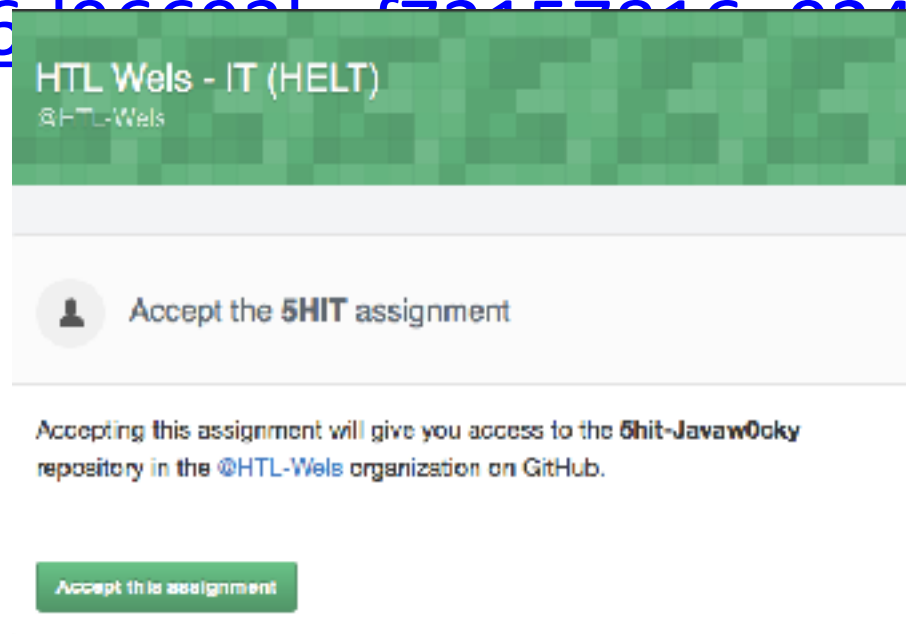


- 
- IT GitHub Inc. wurde 2007 gegründet
  - IT 2008: Start von GitHub
  - IT > 10 Mio reg. User (08/2015)
  - IT Stellt (public) Git-Repositories für Open Source Entwicklungen gratis zur Verfügung
  - IT Private Repos sind zu bezahlen (ausser für Bildungsbereich)



# GitHub

 <https://classroom.github.com/assignment-invitations/49849e26-1066081-670157016-001219>






# GitHub

**GitHub Classroom**GitHub Education

## HTL Wels - IT (HELT)

@HTL-Wels

 Accepted the **5HIT** assignment

### You are ready to go!

You may receive an invitation to join [@HTL-Wels](#) via email invitation on your behalf. No further action is necessary.

Your assignment has been created here: <#>



# GitHub

The screenshot shows the GitHub interface for the repository 'ITL-Wells / 5hit-Javew0cky-1'. At the top, there's a search bar and navigation links for 'Pull requests', 'Issues', and 'Git'. Below the repository name, there are statistics: '1' commit, '0' stars, and '0' forks. The main content area is titled 'Quick setup — If you've done this kind of thing before'. It offers three options: 'Set up in Desktop', 'HTTPS', and 'SSH'. The 'HTTPS' option is selected, showing the URL 'https://github.com/ITL-Wells/5hit-Javew0cky-1.git'. Below this, there's a section for creating a new repository on the command line, with a code block containing the following commands:

```
echo "# 5hit-Javew0cky-1" >> README.md
git init
git add README.md
git commit -m "first commit"
git remote add origin https://github.com/ITL-Wells/5hit-Javew0cky-1.git
git push -u origin master
```

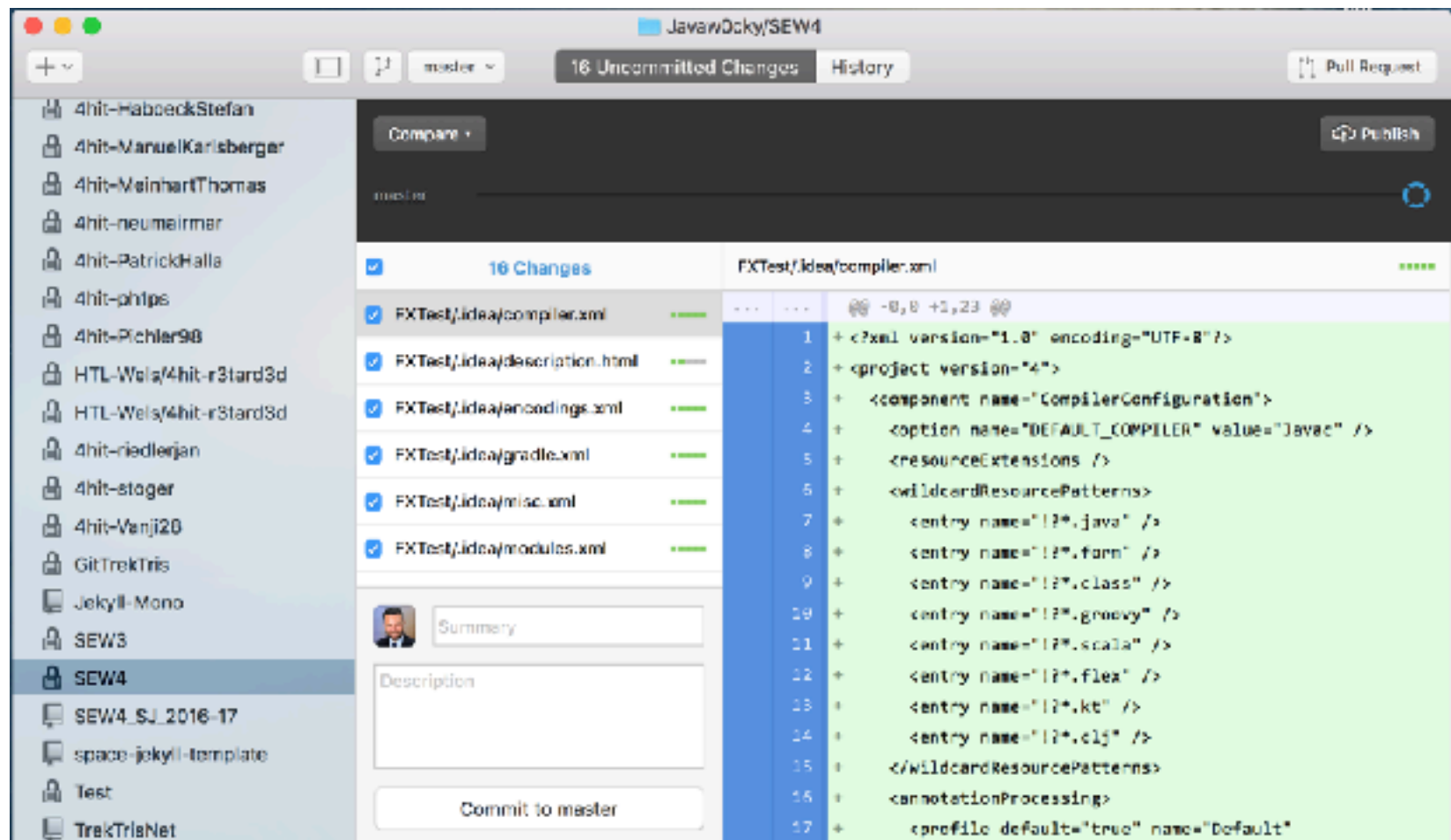
Below this, there's a section for pushing an existing repository from the command line, with a code block containing the following commands:

```
git remote add origin https://github.com/ITL-Wells/5hit-Javew0cky-1.git
git push -u origin master
```

Finally, there's a section for importing code from another repository, with a note that you can initialize the repository with code from a Subversion, Mercurial, or TFS project. At the bottom, there's a footer with the GitHub logo and links for 'Contact GitHub', 'API', 'Training', 'Shop', 'Blog', and 'About'.



# GitHub Desktop





# GitHub Desktop

---

- IT IntelliJ Projekt erstellen in Ordner
- IT GitHub Desktop aufrufen
- IT Initial commit
- IT Publish
- IT GitHub Webseite aufrufen
- IT Class Files, IntelliJ Projektdateien entfernen
- IT README.md erstellen



# Git Shell Befehle

---

- IT Neuen Projekt Ordner erstellen
- IT Powershell in GitHub Desktop öffnen
- IT Powershell: cd Projekt Ordner



# Repository erstellen

---

## git init

```
BatBook-Pro:IdeaProjects tom$ cd gittest/
```

```
BatBook-Pro:gittest tom$ ls -la
total 0
drwxr-xr-x  2 tom  staff   68  5 Feb 13:27 .
drwxr-xr-x 33 tom  staff  1122  5 Feb 13:27 ..
```

```
BatBook-Pro:gittest tom$ git init
Initialized empty Git repository in /Users/tom/IdeaProjects/gittest/.git/
```

```
BatBook-Pro:gittest tom$ ls -la
total 0
drwxr-xr-x  3 tom  staff   102  5 Feb 13:27 .
drwxr-xr-x 33 tom  staff  1122  5 Feb 13:27 ..
drwxr-xr-x 10 tom  staff   340  5 Feb 13:27 .git
```





# Status der Git Repo

---

## git status

```
BatBook-Pro:gittest tom$ git status
```

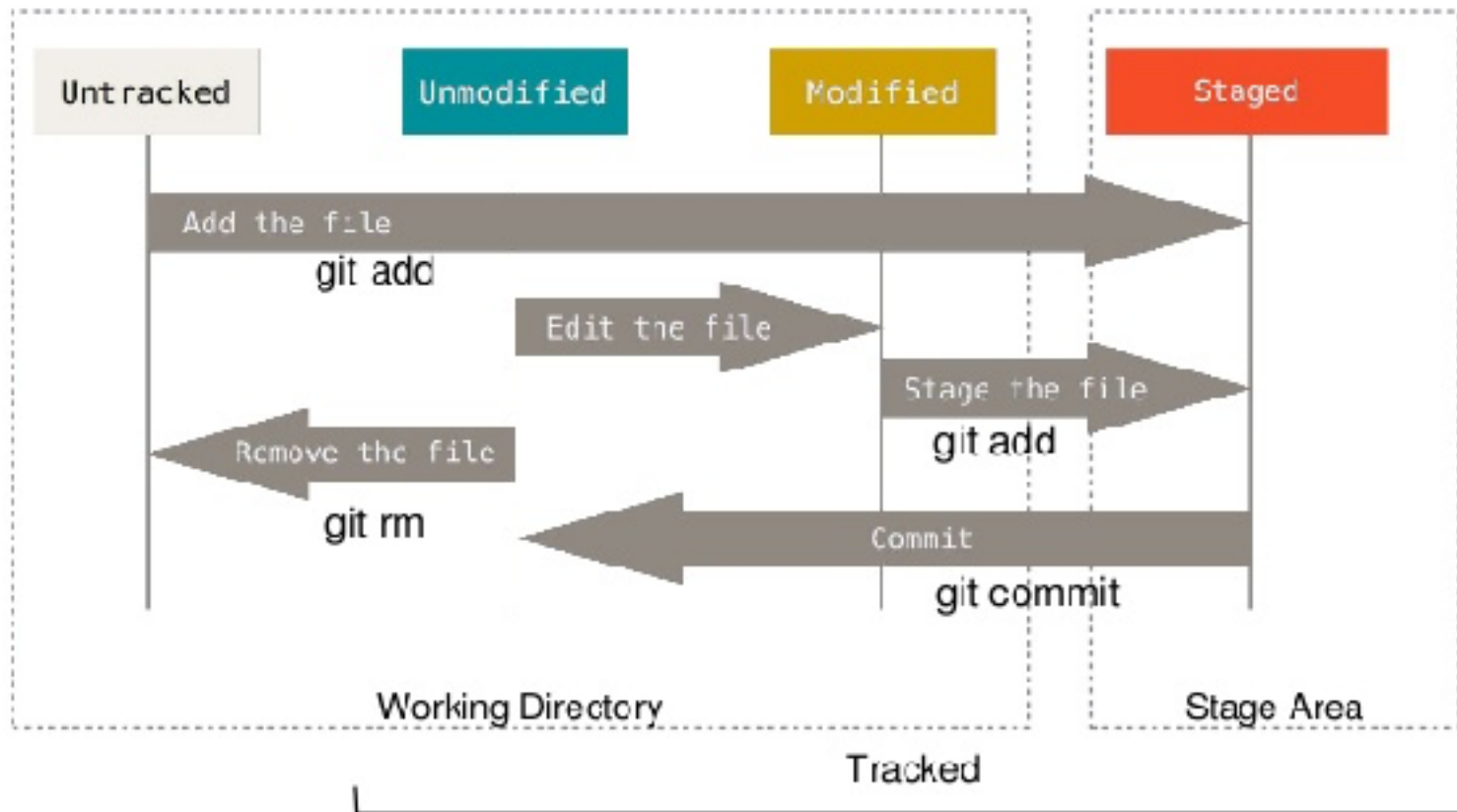
```
On branch master
```

```
Initial commit
```

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



# File Status





# Datei erstellen

---

```
BatBook-Pro:gittest tom$ echo "#Git Test" > README.md
```

```
BatBook-Pro:gittest tom$ ls -la
```

```
total 8
drwxr-xr-x  4 tom  staff   136  5 Feb 13:34 .
drwxr-xr-x 33 tom  staff  1122  5 Feb 13:27 ..
drwxr-xr-x 10 tom  staff   340  5 Feb 13:29 .git
-rw-r--r--  1 tom  staff    10  5 Feb 13:34 README.md
```

```
BatBook-Pro:gittest tom$ git status
```

```
On branch master
```

```
Initial commit
```

```
Untracked files:
```

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

README.md

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



# Tracking Files

---

```
BatBook-Pro:gittest tom$ git add README.md
```

```
BatBook-Pro:gittest tom$ git status
```

On branch master

Initial commit

Changes to be committed:

(use "git rm --cached <file>..." to unstage)

new file: README.md



# Commit

---

```
BatBook-Pro:gittest tom$ git commit -m "Initial Commit"
```

```
[master (root-commit) 6425d14] Initial Commit  
1 file changed, 1 insertion(+)  
create mode 100644 README.md
```

```
BatBook-Pro:gittest tom$ git status
```

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



# IntelliJ Projekt erstellen

---

```
BatBook-Pro:gittest tom$ ls -la
total 16
drwxr-xr-x  8 tom  staff  272  5 Feb 14:02 .
drwxr-xr-x 33 tom  staff 1122  5 Feb 13:27 ..
drwxr-xr-x 13 tom  staff  442  5 Feb 14:02 .git
drwxr-xr-x 10 tom  staff  340  5 Feb 14:02 .idea
-rw-r--r--  1 tom  staff   21  5 Feb 13:45 README.md
-rw-r--r--  1 tom  staff  425  5 Feb 14:02 gittest.iml
drwxr-xr-x  3 tom  staff  102  5 Feb 14:02 out
drwxr-xr-x  3 tom  staff  102  5 Feb 14:02 src
BatBook-Pro:gittest tom$ 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.md

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

        .idea/
        gittest.iml
        out/
        src/

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



# Staging Modified Files

---

```
BatBook-Pro:gittest tom$ git add .
```

```
BatBook-Pro:gittest tom$ git status
```

On branch master

Changes to be committed:

(use "git reset HEAD <file>..." to unstage)

```
new file:   .idea/compiler.xml
new file:   .idea/description.html
new file:   .idea/encodings.xml
new file:   .idea/misc.xml
new file:   .idea/modules.xml
new file:   .idea/vcs.xml
new file:   .idea/workspace.xml
modified:   README.md
new file:   gittest.iml
new file:   out/production/gittest/Main.class
new file:   src/Main.java
```



# Unnötige Files entfernen

---

```
BatBook-Pro:gittest tom$ git rm -r --cached out/  
rm 'out/production/gittest/Main.class'
```

```
BatBook-Pro:gittest tom$ git status
```

On branch master

Changes to be committed:

(use "git reset HEAD <file>..." to unstage)

```
new file:   .idea/compiler.xml  
new file:   .idea/description.html  
new file:   .idea/encodings.xml  
new file:   .idea/misc.xml  
new file:   .idea/modules.xml  
new file:   .idea/vcs.xml  
new file:   .idea/workspace.xml  
modified:   README.md  
new file:   gittest.iml  
new file:   src/Main.java
```

Untracked files:

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

out/





# Unnötige Files entfernen

---

```
BatBook-Pro:gittest tom$ git rm -r --cached .idea/  
rm '.idea/compiler.xml'  
rm '.idea/description.html'  
rm '.idea/encodings.xml'  
rm '.idea/misc.xml'  
rm '.idea/modules.xml'  
rm '.idea/vcs.xml'  
rm '.idea/workspace.xml'
```

```
BatBook-Pro:gittest tom$ git rm --cached gittest.iml  
rm 'gittest.iml'
```



# Unnötige Files entfernen

---

```
atBook-Pro:gittest tom$ git status
```

```
On branch master
```

```
Changes to be committed:
```

```
(use "git reset HEAD <file>..." to unstage)
```

```
    modified:   README.md
```

```
    new file:   src/Main.java
```

```
Untracked files:
```

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

```
.idea/  
gittest.iml  
out/
```



# Commit

---

```
BatBook-Pro:gittest tom$ git commit -m "Hello World hinzugefügt"
[master 903a610] Hello World hinzugefügt
 2 files changed, 7 insertions(+), 1 deletion(-)
 create mode 100644 src/Main.java
```

```
BatBook-Pro:gittest tom$ git status
```

On branch master

Untracked files:

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

```
.idea/
gittest.iml
out/
```

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



# Dateien umbenennen bzw. verschieben

---

- ① Werden Dateien verschoben bzw. umbenannt, so bekommt Git das nicht korrekt mit
- ① Daher gibt es ein extra Kommando dafür: `mv`

```
BatBook-Pro:gittest tom$ git mv README.md readme.md
```

```
BatBook-Pro:gittest tom$ git status
```

```
On branch master
```

```
Changes to be committed:
```

```
(use "git reset HEAD <file>..." to unstage)
```

```
renamed:    README.md -> readme.md
```



# .gitignore

---

IT in der Datei .gitignore können Dateien angegeben werden, die automatisch von git ignoriert werden:

```
# class files
*.class
# Package Files #
*.jar*.war*.ear
```



# git log

---

```
BatBook-Pro:gittest tom$ git log
commit 903a610db0d57a89c877b5b95785d8a31134399c
Author: Thomas Helml <thomas.helml@htl-wels.at>
Date:    Sun Feb 5 14:14:40 2017 +0100
```

Hello World hinzugefügt

```
commit 6425d142503278730378e1d446c98a1e1cf799a9
Author: Thomas Helml <thomas.helml@htl-wels.at>
Date:    Sun Feb 5 13:39:07 2017 +0100
```

Initial Commit



# Änderungen rückgängig

---

```
BatBook-Pro:gittest tom$ echo "neuer inhalt" > readme.md
```

```
BatBook-Pro:gittest tom$ cat readme.md  
neuer inhalt
```

```
BatBook-Pro:gittest tom$ git checkout readme.md  
BatBook-Pro:gittest tom$ cat readme.md  
#Git Test - modified
```



# Auf alten Commit rückgängig

---

```
BatBook-Pro:gittest tom$ git log
commit 903a610db0d57a89c877b5b95785d8a31134399c
Author: Thomas Helml <thomas.helml@htl-wels.at>
Date:    Sun Feb 5 14:14:40 2017 +0100
```

Hello World hinzugefügt

```
commit 6425d142503278730378e1d446c98a1e1cf799a9
Author: Thomas Helml <thomas.helml@htl-wels.at>
Date:    Sun Feb 5 13:39:07 2017 +0100
```

Initial Commit

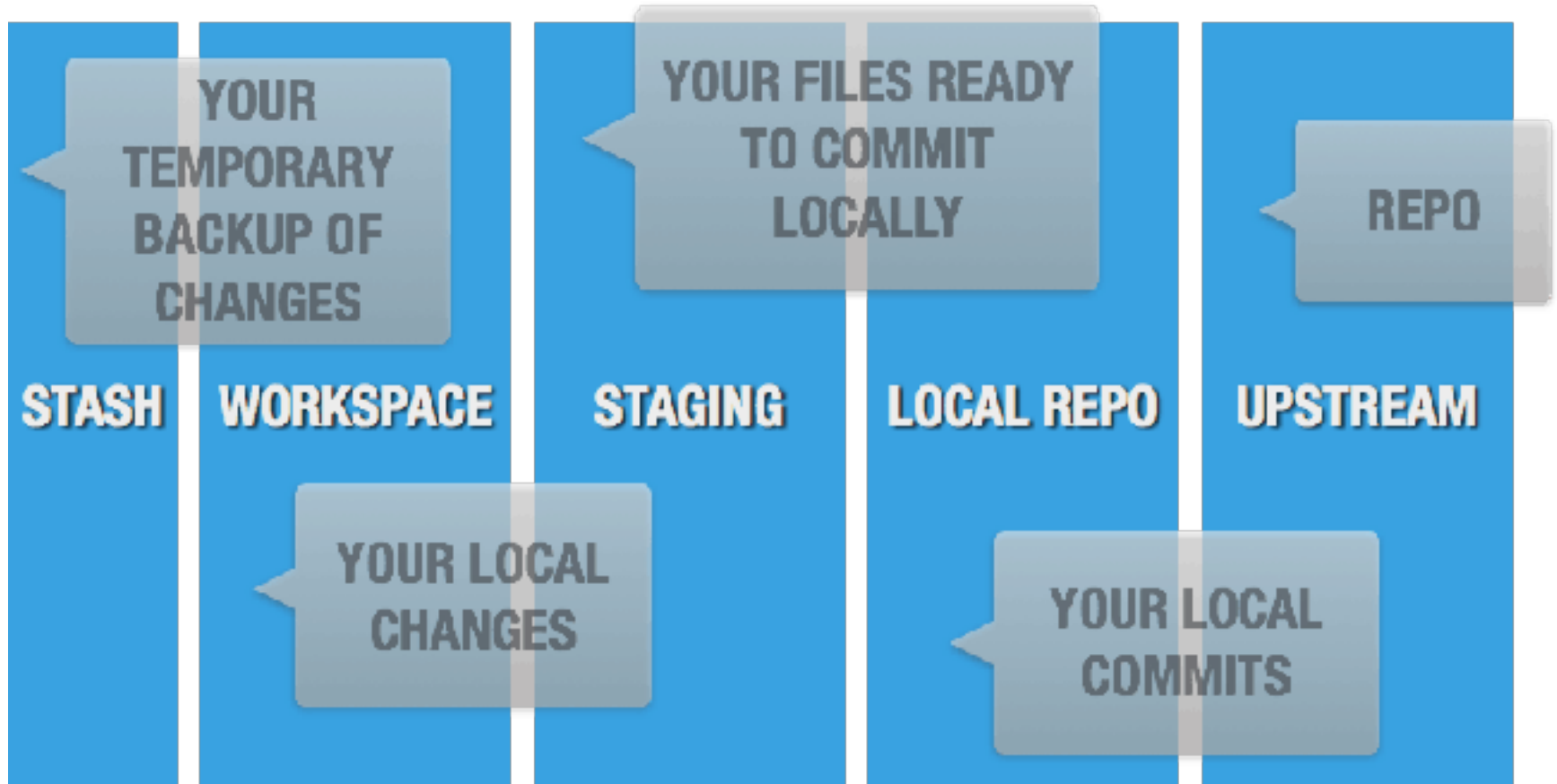
```
git checkout 6425d142503278730378e1d446c98a1e1cf799a9
Previous HEAD position was 903a610... Hello World hinzugefügt
HEAD is now at 6425d14... Initial Commit
```





# Die Areas

---





# Quellen

---

- ① <http://www.slideshare.net/paradigmatecnologico/git-vs-subversion>
- ① <https://github.com/CourseReps/ECEN489-Fall2015/wiki/git>
- ① <https://git-scm.com/book/en/v2/Getting-Started-Git-Basics>