Was bringt Ihnen Git und Github?

An einer Projektarbeit arbeiten

Projektdatei in RMarkdown

```
#Hier führen Sie Code aus
library(readxl)

einlesen <- read_csv("mydata.csv")

# ... Code zur Bearbeitung ihrer Projektarbeit
```

Bearbeitung ihrer Ausarbeitung:

- ✚ Abspeichern als **"Projektausarbeitung.Rmd"**

An einer Projektarbeit arbeiten

Projektdatei in RMarkdown

```
#Hier führen Sie Code aus
library(readxl)

einlesen <- read_csv("mydata.csv")

# ... Code zur Bearbeitung ihrer Projektarbeit

# ... Code noch einmal Überarbeiten
```

Bearbeitung ihrer Ausarbeitung:

- Abspeichern als "Projektausarbeitung.Rmd"
- **"Projektausarbeitung.Rmd"** überschrieben

An einer Projektarbeit arbeiten

Projektdatei in RMarkdown

```
#Hier führen Sie Code aus
library(readxl)

einlesen <- read_csv("mydata.csv")

# ... Code zur Bearbeitung ihrer Projektarbeit

# ... Code noch einmal Überarbeiten

# ... Deskriptive Statistiken anfertigen
```

Bearbeitung ihrer Ausarbeitung:

- Abspeichern als "Projektausarbeitung.Rmd"
- "Projektausarbeitung.Rmd" überschrieben
- **"Projektausarbeitung.Rmd"** überschrieben

An einer Projektarbeit arbeiten

Projektdatei in RMarkdown

```
#Hier führen Sie Code aus
library(readxl)

einlesen <- read_csv("mydata.csv")

# ... Code zur Bearbeitung ihrer Projektarbeit

# ... überarbeiteten Code noch einmal Überarbeiten

# ... Deskriptive Statistiken anfertigen
```

Bearbeitung ihrer Ausarbeitung:

- Abspeichern als "Projektausarbeitung.Rmd"
- "Projektausarbeitung.Rmd" überschrieben
- "Projektausarbeitung.Rmd" überschrieben
- **"Projektausarbeitung.Rmd"** noch einmal mit den Überarbeitungen überschrieben

An einer Projektarbeit arbeiten

Projektdatei in RMarkdown

```
#Hier führen Sie Code aus
library(readxl)

einlesen <- read_csv("mydata.csv")

# ... Code zur Bearbeitung ihrer Projektarbeit

# ... überarbeiteten Code noch einmal Überarbeiten

# ... Modifizierter Code zerstört die deskriptiven
# ... vorherige Version war besser -> neu Coden!
```

Bearbeitung ihrer Ausarbeitung:

- Abspeichern als "Projektausarbeitung.Rmd"
- "Projektausarbeitung.Rmd" überschrieben
- "Projektausarbeitung.Rmd" überschrieben
- "Projektausarbeitung.Rmd" noch einmal mit den Überarbeitungen überschrieben
- **Modifikationen** wieder von Hand **rückgängig machen**

An einer Projektarbeit arbeiten

Projektdatei in RMarkdown

```
#Hier führen Sie Code aus
library(readxl)

einlesen <- read_csv("mydata.csv")

# ... Code zur Bearbeitung ihrer Projektarbeit

# ... überarbeiteten Code noch einmal Überarbeiten

# ... Modifizierter Code zerstört die deskriptiven
# ... vorherige Version war besser -> neu Coden!
```

Bearbeitung ihrer Ausarbeitung:

- Abspeichern als "Projektausarbeitung.Rmd"
- "Projektausarbeitung.Rmd" überschrieben
- "Projektausarbeitung.Rmd" überschrieben
- "Projektausarbeitung.Rmd" noch einmal mit den Überarbeitungen überschrieben
- Modifikationen wieder von Hand rückgängig machen 😖

Verschiedene Versionen abspeichern

Projektdatei in RMarkdown

```
#Hier führen Sie Code aus
library(readxl)

einlesen <- read_csv("mydata.csv")

# ... Code zur Bearbeitung ihrer Projektarbeit
```

Bearbeitung ihrer Ausarbeitung:

- ✚ Abspeichern als **"Projektausarbeitung_01.Rmd"**

Verschiedene Versionen abspeichern

Projektdatei in RMarkdown

```
#Hier führen Sie Code aus
library(readxl)

einlesen <- read_csv("mydata.csv")

# ... Code zur Bearbeitung ihrer Projektarbeit

# ... überarbeiteten Code noch einmal Überarbeiten
```

Bearbeitung ihrer Ausarbeitung:

- Abspeichern als "Projektausarbeitung_01.Rmd"
- Abspeichern als **"Projektausarbeitung_02.Rmd"**

Verschiedene Versionen abspeichern

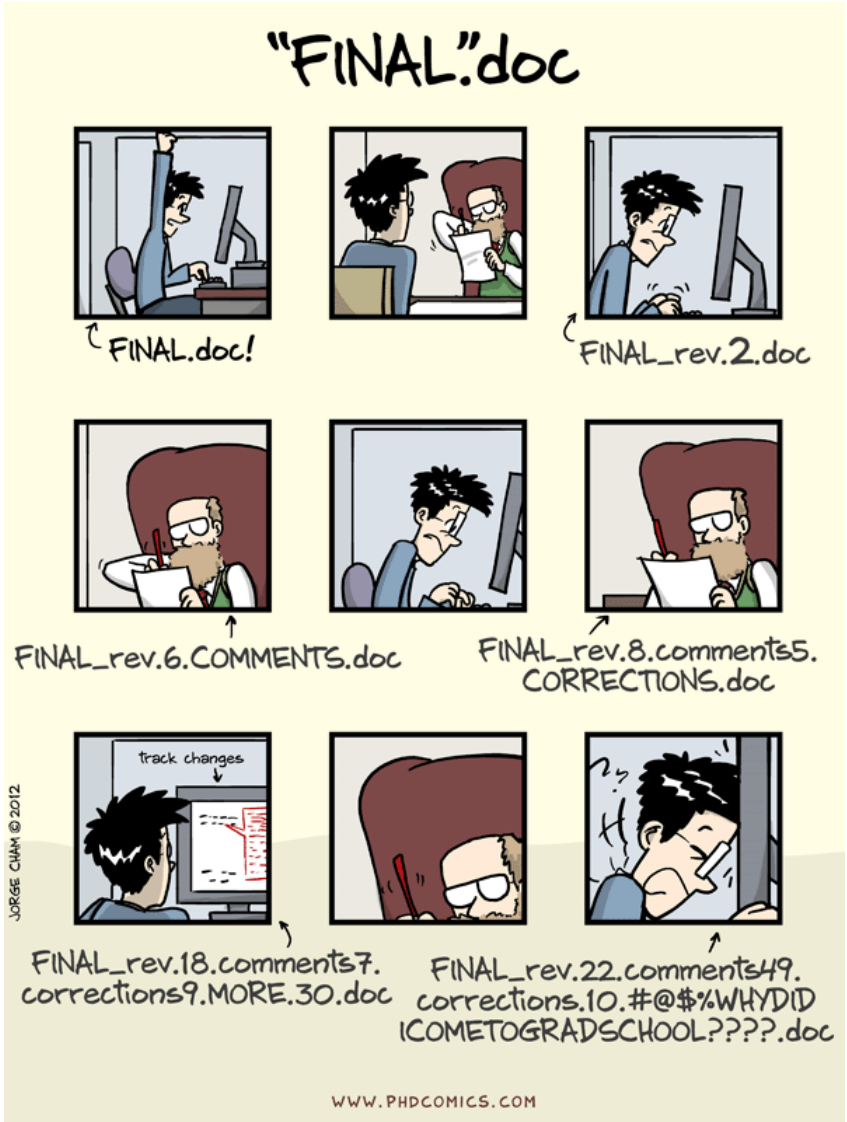
Projektdatei in RMarkdown

```
#Hier führen Sie Code aus
library(readxl)

einlesen <- read_csv("mydata.csv")

# ... Code zur Bearbeitung ihrer Projektarbeit

# ... überarbeiteten Code noch einmal Überarbeiten

# ... Deskriptiven Statistiken erstellen
```

Bearbeitung ihrer Ausarbeitung:

- Abspeichern als "Projektausarbeitung_01.Rmd"
- Abspeichern als "Projektausarbeitung_02.Rmd"
- Abspeichern als **"Projektausarbeitung_03.Rmd"**

Sechs Monate später…



Quelle: http://phdcomics.com/comics/archive/phd101212s.gif

Verschiedene Versionen abspeichern

- Weiterarbeiten am Projekt
- Projekt mit neuen Daten updaten
- Einen Fehler korrigieren

Welche Datei war nochmal die richtige?

- "Projektausarbeitung_final.Rmd"
- "Projektausarbeitung_final0.Rmd"
- "Projektausarbeitung_last.Rmd"
- "Projektausarbeitung_last_korrigiert.Rmd"
- ...

Achtung: **Nicht** Versionsname + Speicherdatum verwenden!

- Versionsname + Speicherdatum sind nicht eindeutig
  - Wurde das Dokument als letztes gespeichert oder nur kurz geöffnet?
  - Haben Sie eventuell zuvor schon die falsche Datei bearbeitet?
  - Wurde eventuell sogar die tatsächlich letzte Datei gelöscht?

Achtung: **Nicht** Versionsname + Speicherdatum verwenden!

- Versionsname + Speicherdatum sind nicht eindeutig
  - Wurde das Dokument als letztes gespeichert oder nur kurz geöffnet?
  - Haben Sie eventuell zuvor schon die falsche Datei bearbeitet?
  - Wurde eventuell sogar die tatsächlich letzte Datei gelöscht?
- Andere Probleme:
  - Was hat sich zwischen den Versionen geändert?
  - Was sollte noch gemacht werden?
  - Es können nicht mehrere Personen am gleichen Dokument arbeiten

Hier kommt Git und Github ins Spiel

Git und Github

- Sie arbeiten immer an **einer** Datei
- Mehrere Personen können gleichzeitg an der Datei arbeiten
  - D.h. Projekte können gemeinsam bearbeitet werden

In diesem Projektkurs setzen wir auf Github Desktop um ihnen den Einstieg in Git und Github zu erleichtern!

Git und Github

- Sie arbeiten immer an **einer** Datei
- Mehrere Personen können gleichzeitg an der Datei arbeiten
  - D.h. Projekte können gemeinsam bearbeitet werden

In diesem Projektkurs setzen wir auf Github Desktop um ihnen den Einstieg in Git und Github zu erleichtern!

Github Desktop

Für wen ist Github Desktop?

- Für Personen die neu mit Git in Berührung kommen
- Für die Bearbeitung gemeinsamer Projekte über eine Oberfläche

> Unsere Videos zu Github Desktop und das Arbeiten mit Git sollten Sie sich unbedingt anschauen!

Quellen

Ausarbeitung inspiriert von

- https://github.com/saghirb/Getting-Started-with-Git-and-GitHub-for-R-Users
- Happy Git with R

Zusammenfassung: Warum Git und Github?

- **Code teilen:** Durch Github können Sie einfach ihre Arbeit mit anderen teilen. Auch ihrem späteren Arbeitgeber können Sie so zeigen, was Sie können
- **Zusammenarbeit:** Wenn Sie ein zentrales Repositorium auf Github haben dann können viele Personen gleichzeitig an dem Projekt arbeiten. Dadurch ist jeder auf dem aktuellsten Stand. Durch sogenannte Pull-Requests können Anmerkungen gesendet werden, welche Änderungen am Code vorsehen. Sie können dann diese Anfragen annehmen oder ablehnen.
- **Versionskontrolle:** Das Beste an Git ist dessen Versionierungsfähigkeit von unterschiedlichsten Dokumenten. Sie können jede beliebige Datei so oft speichern wie Sie möchten und immer wieder zu einem alten Speicherstand zurückkehren, falls Sie dies wünschen. Sie können auch sogenannte "branches" kreieren, bei denen Sie sich austoben und Dinge ausprobieren können und diese nachher zu "Main" zumergen.

Git Commands

Es gibt sehr viele Aktionen, welche Sie mit Git machen können, wir wollen uns hier auf die vier Hauptaktionen konzentrieren:

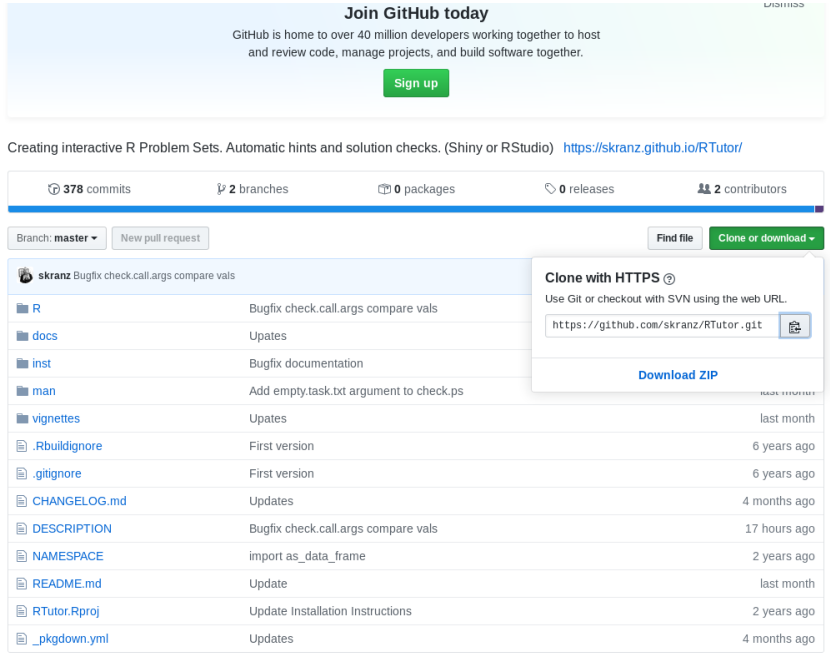
- **pull**: Sie importieren (pull) Änderungen von dem *remote* Repositorium (aka von Github)
- **add**: Fügen Sie Dateien hinzu, dazu wird auch *stage* gesagt
- **commit**: Änderungen an ihrem *lokalen* Repositorium
- **push**: Überträgt die lokal gemachten Änderungen an das *remote* Repositorium

Quelle: https://rafalab.github.io/dsbook/git.html

Clone

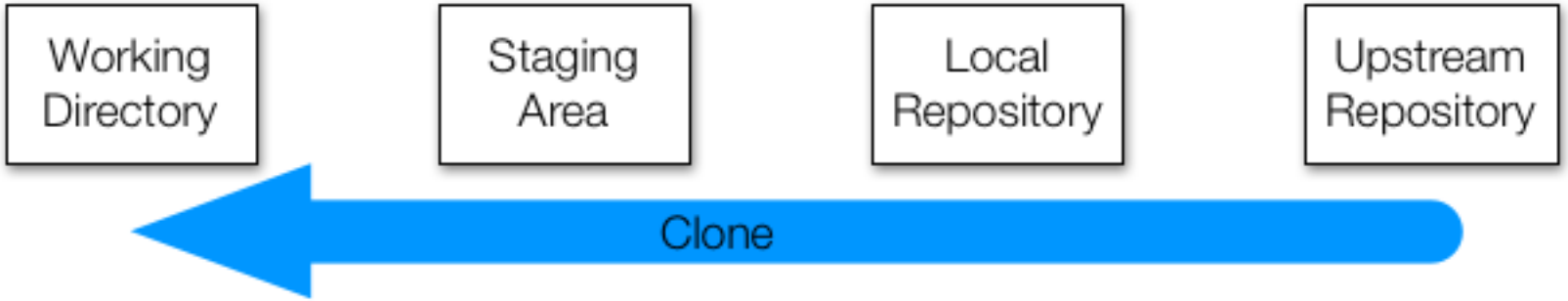
- Sie können ein *Upstream Repository* auf Github klonen
  - Es wird die komplette Struktur geklont
  - Wir können bspw. das RTutor Repositorium klonen
  - Der Link hierzu: https://github.com/skranz/RTutor.git

Siehe hierzu die Einführungsvideos zu Repositories klonen mit Github Desktop!



Clone

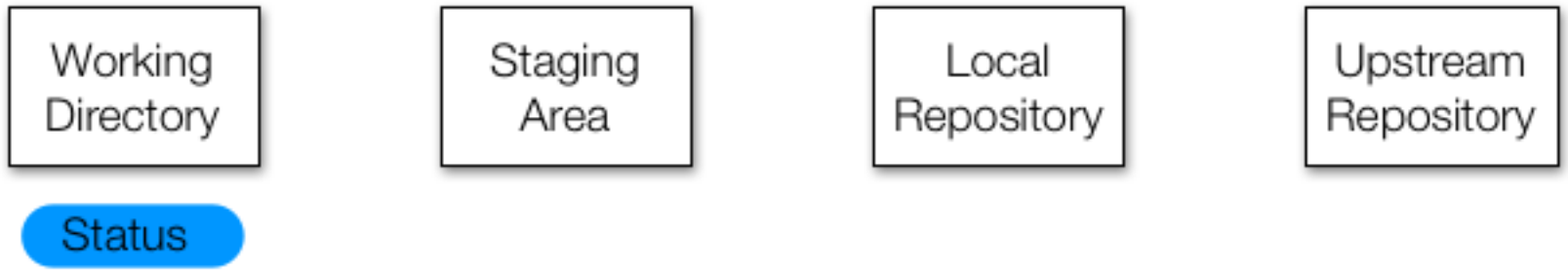
- Sie haben soeben das komplette RTutor Repositorium auf ihren PC geklont!
- Das *Working Directory* ist ihr aktuelles Verzeichnis
- Änderungen erfolgen immer *nur* in ihrem aktuellen Verzeichnis!



Quelle: https://rafalab.github.io/dsbook/git.html

Status

- Mit `git status` können Sie überprüfen, wie ihr aktueller Status ist über alle *areas* ist
- Haben Sie bereits Dateien geändert und müssen diese noch *committed* werden?



Quelle: https://rafalab.github.io/dsbook/git.html

Dateien hinzufügen

- Änderungen die Sie an Dateien vornehmen wollen Sie tracken und versioniert sehen
- **Doch:** Nicht jede kleinste Änderung soll in Github landen.
  - In der Regel wollen Sie nur Änderungen mit Github synchronisieren welche es auch wert sind
- Wenn Sie Änderungen in der *Staging area*, dann sind diese noch nicht in der Versionierung!

Beispiel: Hinzufügen einer Datei zu unserem Repositorium. Z.B. einer leeren Daten mit dem Namen `Neu.txt`

Stage

Quelle: https://rafalab.github.io/dsbook/git.html

Commit

Nun müssen Sie die neue Datei noch zu ihrem *lokalen Repositorium* hinzufügen, dies machen Sie mit `git commit`

- Sie sollten zu jedem Commit eine möglichst aussagekräftige Beschreibung dessen was Sie gemacht haben hinzufügen

  > git commit -m "Hier ihre Beschreibung als Commit"

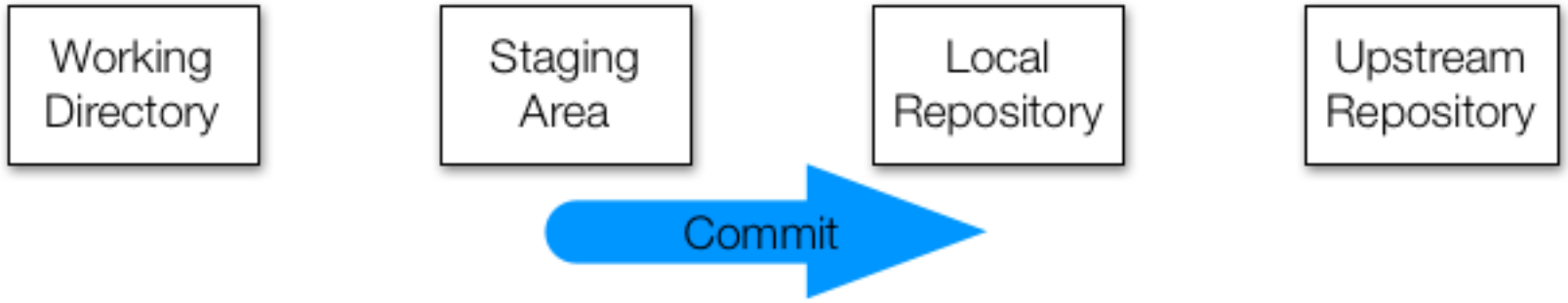
Commit

Nun müssen Sie die neue Datei noch zu ihrem *lokalen Repositorium* hinzufügen, dies machen Sie mit `git commit`

- Sie sollten zu jedem Commit eine möglichst aussagekräftige Beschreibung dessen was Sie gemacht haben hinzufügen

> git commit -m "Hier ihre Beschreibung als Commit"

Sie haben nun ihr *lokales Repositorium* geändert, dies können Sie mit `git status` nachvollziehen



Quelle: https://rafalab.github.io/dsbook/git.html

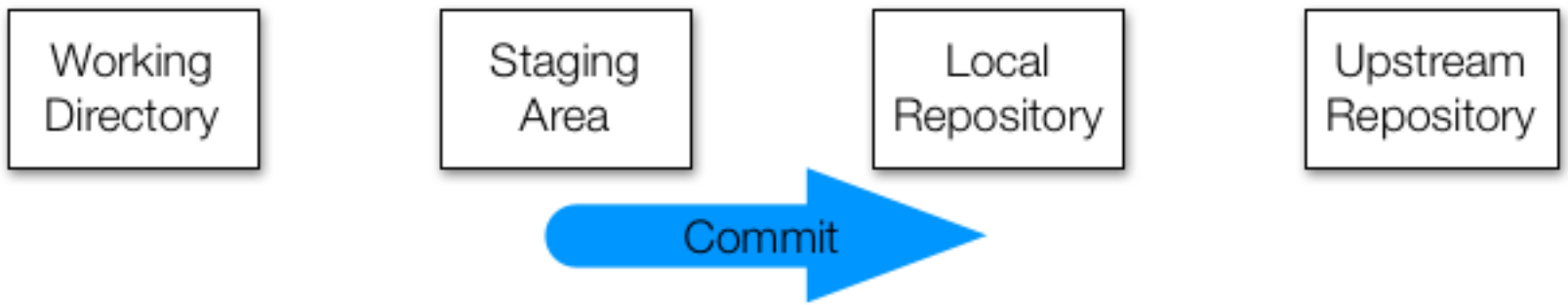
Commit

Nun müssen Sie die neue Datei noch zu ihrem *lokalen Repositorium* hinzufügen, dies machen Sie mit `git commit`

- ✚ Sie sollten zu jedem Commit eine möglichst aussagekräftige Beschreibung dessen was Sie gemacht haben hinzufügen

> git commit -m "Hier ihre Beschreibung als Commit"

Sie haben nun ihr *lokales Repositorium* geändert, dies können Sie mit `git status` nachvollziehen



Quelle: https://rafalab.github.io/dsbook/git.html

Wenn Sie nun ihre Datei ändern, dann tun Sie dies wieder nur in ihrem aktuellen Verzeichnis, d.h. dem *Working Directory*

> Änderungen müssen wieder committed werden!

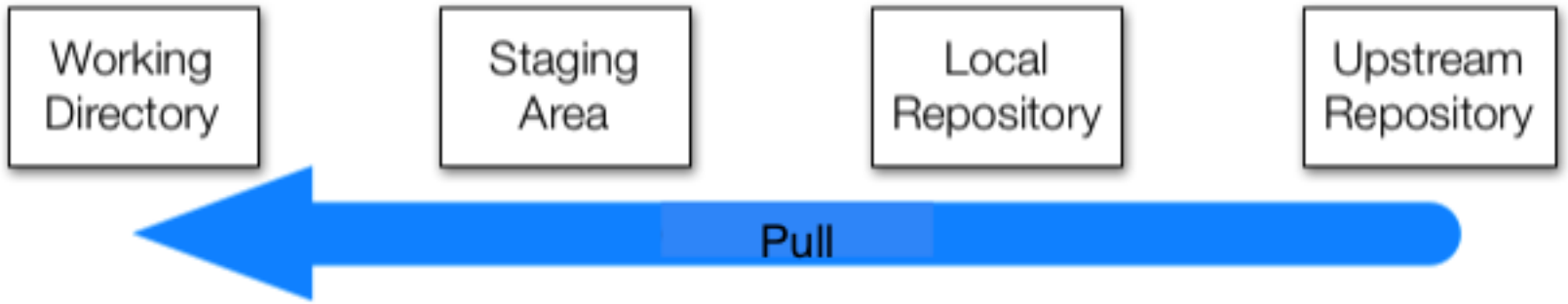
Push

Im letzten Schritt sollten Sie ihre Änderungen zu ihrem *remote Repositorium* **pushen**:

Quelle: https://rafalab.github.io/dsbook/git.html

Pull

Das *remote Repositorium* wird sich über die Zeit verändern, wenn mehrere Personen daran arbeiten. Sie sollten **immer**, d.h. jedes mal wenn Sie beginnen an ihrem Projekt zu arbeiten (z.B. jeden morgen) Änderungen vom *remote Repositorium* **pullen**



Quelle: https://rafalab.github.io/dsbook/git.html

Mergekonflikte

Mergekonflikte

- Können vorkommen, wenn mehrere Personen an der gleichen Datei arbeiten
- Wenn beide die gleiche Zeile bearbeiten und die Datei an das *remote Repositorium* pushen, dann meldet Git einen **Mergekonflikt**

> Gegeben Sie und einer ihrer Partner arbeiten an der gleichen Zeile. Ihr Partner pushed seine Änderungen vor ihnen. Was passiert?

Mergekonflikte

- Können vorkommen, wenn mehrere Personen an der gleichen Datei arbeiten
- Wenn beide die gleiche Zeile bearbeiten und die Datei an das *remote Repositorium* pushen, dann meldet Git einen **Mergekonflikt**

> Gegeben Sie und einer ihrer Partner arbeiten an der gleichen Zeile. Ihr Partner pushed seine Änderungen vor ihnen. Was passiert?

- Git zeigt ihnen einen Fehler an
- **Was ist zu tun?** $\rightarrow$ pullen Sie!
- Schauen Sie auf den Merge Konflikt
- Wählen Sie die richtige/präferierte Änderung der Datei aus und bereinigen so den Mergekonflikt
- Committen Sie ihre neue Datei mit einer entsprechenden Nachricht
- Pushen Sie.

Mergekonflikte vermeiden

> **Pullen** Sie **immer** bevor Sie mit ihrer Arbeit starten, dann sind Sie auf dem aktuellen Stand!

- Committen und pushen Sie oft um Mergekonflikte zu vermeiden, oder die Änderungen auf ein Minimum zu reduzieren
  - Erspart ihnen viel Arbeit!
- Wenn Sie eine Situation nicht lösen können, dann fragen Sie **sofort** nach Hilfe und warten nicht bis das Problem zu groß wird!