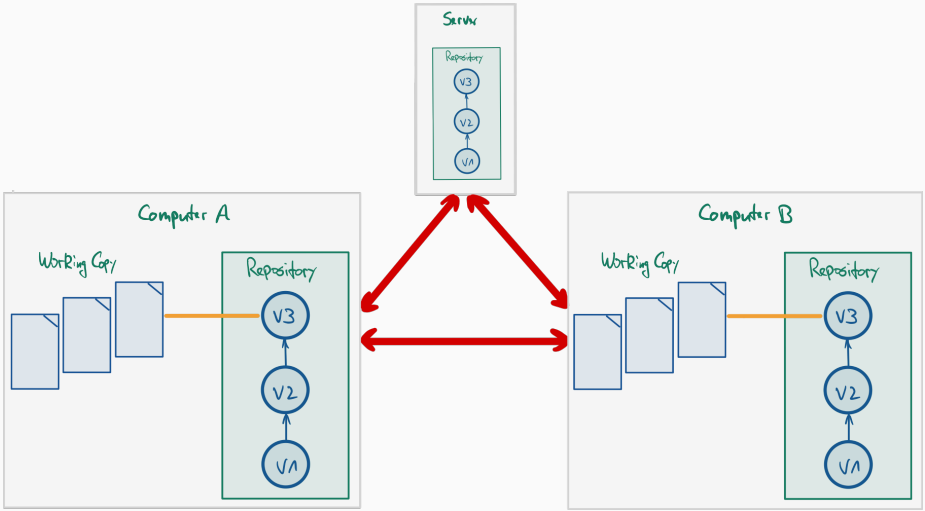
# Arbeiten mit Git Remotes (dezentrale Repos)

Carsten Gips (HSBI)

# Nutzung von Git in Projekten: Verteiltes Git (und Workflows)



# Clonen kann sich lohnen . . .

```
https://github.com/Programmiermethoden-CampusMinden/Prog2-Lecture

---C---D---E  master
```

=> `git clone https://github.com/Programmiermethoden-CampusMinden/Prog2-Lecture`

```
./Prog2-Lecture/  (lokaler Rechner)

---C---D---E  master
           ^origin/master
```

Hinweis auf Protokolle, Beispiel

# Eigener und entfernter *master* entwickeln sich weiter ...

```
https://github.com/Programmiermethoden-CampusMinden/Prog2-Lecture

---C---D---E---F---G  master
```

```
./Prog2-Lecture/  (lokaler Rechner)

---C---D---E---H  master
           ^origin/master
```

# Änderungen im Remote holen und Branches zusammenführen

```
https://github.com/Programmiermethoden-CampusMinden/Prog2-Lecture

---C---D---E---F---G  master
```

=> `git fetch origin`

```
./Prog2-Lecture/  (lokaler Rechner)

---C---D---E---H  master
            \
             F---G  origin/master
```

# Branches und Remotes

- Eigenen (neuen) lokalen Branch ins remote Repo schicken
  - `git push <remote> <branch>`

- Neuer Branch im remote Repo
  - `git fetch <remote>` holt (auch) alle neuen Branches
  - Lokale Änderungen an remote Branches nicht möglich!
    => **Remote Branch in lokalen Branch mergen** (oder auschecken)

Beispiel

# Zusammenfassung: Arbeiten mit Remotes

1. Änderungen vom Server holen: `git fetch <remote>`
   => Holt alle Änderungen vom Repo `<remote>` ins eigene Repo (Workingcopy bleibt unangetastet!)

2. Aktuellen lokalen Branch auffrischen: `git merge <remote>/<branch>`
   (oder alternativ `git pull <remote> <branch>`)

3. Eigene Änderungen hochladen: `git push <remote> <branch>`

# Vereinfachung: Tracking Branches

- **Tracking Branch**: lokaler Branch, der remote Branch "verfolgt"
  - Beispiel: lokaler `master`-Branch folgt `origin/master` per Default

- **Vereinfachung im Workflow**:
  - `git pull` entspricht
    1. `git fetch <remote>` **plus**
    2. `git merge <remote>/<branch>`
  - `git push` entspricht `git push <remote> <branch>`

Vorsicht: `pull` und `push` beziehen sich nur auf ausgecheckten Tracking Branch

# Einrichten von Tracking Branches

- `git clone`: lokaler `master` trackt automatisch `origin/master`
- Remote Branch als Tracking Branch einrichten:
  1. `git fetch <remote>`
  2. `git checkout -t <remote>/<branch>`
- Lokalen Branch als Tracking Branch einrichten:
  1. `git checkout -b <branch>`
  2. `git push -u <remote> <branch>`

Beispiel

# Hinzufügen eines (weiteren) Remote Repository


```
git remote add <name> <url>
```

# Wrap-Up

- Synchronisierung des lokalen Repos mit anderen Repos
  - Repo kopieren: `git clone <url>`
  - Interner Name fürs fremde Repo: `origin`
  - Änderungen vom fremden Repo holen: `git fetch <remote>`
  - Änderungen in lokalen Branch einpflegen: `git merge <remote>/<branch>`
  - Eigene Änderungen ins fremde Repo schieben: `git push <remote> <branch>`

- Tracking Branches (Konzept, Anwendung)
  - Remote Branches können lokal nicht verändert werden:
    - In lokale Branches mergen, oder
    - Tracking Branches anlegen => einfaches `pull` und `push` nutzen
  - Tracking Branches sind lokale Branches, die remote Branches verfolgen (“tracken”)

# LICENSE