



Tag 1: Einführung in Git und GitLab

08.07.2024, Daniel Krämer

© Copyright 2024 anderScore GmbH

HECKER
CONSULTING

- **Tag 1 – Einführung in Git und GitLab**
 - Einführung & Kursüberblick
 - Grundlagen von Git
 - Git Rebase und Merge-Strategien
 - Git Remote
 - Grundlagen von GitLab
- **Tag 2 – Git-Workflows, CI/CD, GitLab CI**
 - Git-Workflow im Team
 - Gitflow-Workflow
 - Tags, Releases & deren Verwaltung
 - Einführung in GitLab CI/CD & gitlab-ci.yml
 - GitLab Runner
- **Tag 3 – Docker, GitOps, Deployment-Strategien**
 - Entwicklung mit Docker
 - Container/Docker-Registry
 - Erstellen von Release- und Tagged-Images
 - GitOps Grundlagen
 - Möglichkeiten des Deployments & Verwaltung von Konfiguration
 - Abschlussübung & Diskussion

- **Tag 1 – Einführung in Git und GitLab**
 - Einführung & Kursüberblick
 - Grundlagen von Git
 - Git Rebase und Merge-Strategien
 - Git Remote
 - Grundlagen von GitLab
- **Tag 2 – Git-Workflows, CI/CD, GitLab CI**
 - Git-Workflow im Team
 - Gitflow-Workflow
 - Tags, Releases & deren Verwaltung
 - Einführung in GitLab CI/CD & gitlab-ci.yml
 - GitLab Runner
- **Tag 3 – Docker, GitOps, Deployment-Strategien**
 - Entwicklung mit Docker
 - Container/Docker-Registry
 - Erstellen von Release- und Tagged-Images
 - GitOps Grundlagen
 - Möglichkeiten des Deployments & Verwaltung von Konfiguration
 - Abschlussübung & Diskussion

Arbeiten mit einem Git

Remote Repository

- Remote Repository ist eine zentrale Komponente in der Nutzung von Git mit mehreren Personen
- Oft einfach als Remote bezeichnet
- Wird im Netzwerk gehostet
 - Einfaches Git Repository auf eigenem Server
 - Zugriff über HTTPS oder SSH
 - Dienste wie GitLab, GitHub, BitBucket usw. ergänzen Hosting um weitere Features
- Unterscheidet sich technisch nicht von einem lokalen Repository
 - Remote Repositories sind häufig *Bare Repositories*
- Mehrere Remotes zu einem lokalen Repository möglich

Klonen eines vorhandenen Remote Repositories

- Erzeugt lokale Kopie (**local**) eines Remote Repository (**origin**) mittels
`git clone <remote-reference>`
- Erstellt Verknüpfung zwischen **local** und **origin**
(push, pull, fetch, ...)
- **local** besteht zunächst nur aus **default** Branch
- **Upstream** definiert den zu einem lokalen Branch zugehörigen remote Branch, auf dem beim fetch, pull oder push zugegriffen wird

Beispiel: Klonen eines Projektes

```
$ git clone gituser@gitlab.example.de:git_demo  
$ git branch  
* Main
```

- Alle Remote Branches anzeigen

```
$ git branch --all  
* main  
remotes/origin/HEAD -> origin/main  
remotes/origin/feature1  
remotes/origin/feature2  
...
```

Hinzufügen eines Remote Repository

- Remote zu lokalem Repository hinzuzufügen
`git remote add <name> <remote-uri>`
- `git remote add` legt alias für die Remote Adresse an
- Ruft Informationen über verfügbare Branches ab
- Beispiel:

```
$ git remote add other git@gitlab.example.de:git_demo_2
$ git branch --all
* main
remotes/origin/HEAD -> origin/main
remotes/origin/feature1
remotes/origin/feature2
...
remotes/other/other_feature
```


Abrufen von Änderungen aus Remote Repository

- Informationen über Commits, Branches, Tags, ... abrufen
`git fetch <remote>`
`git fetch --all` (alle verknüpften Remotes)
- Aktualisiert nur Remote Tracking Bereich des Repositorys
- Änderungen müssen mittels Merge oder Rebase in lokalen Branch übernommen werden

```
git checkout feature
```

```
git fetch origin
```

```
git merge origin/feature
```

oder

```
git rebase origin/feature
```

Pull

- `git pull` als Kombination von `git fetch` und `git merge` oder `git rebase`

```
git pull
```

```
git pull --merge
```

```
git pull --rebase
```

- Default bei `git pull` ist `--merge`
- Kann in `.gitconfig` umgestellt werden

```
git config --global pull.rebase false ➔ Merge
```

```
git config --global pull.rebase true ➔ Rebase
```

- Einstellung für einzelne Branches ebenfalls möglich

```
git config branch.<branch-name>.rebase true
```

- Mergeverhalten kann ebenfalls konfiguriert werden
 - `git config --global pull.ff [true | only | false]`
 - **true** (Default)
Versucht Fast-Forward Merge durchzuführen, ansonsten Merge-Commit
 - **false**
Kein FF, Merge-Commit wird immer erstellt
 - **only**
Nur FF, falls nicht möglich, wird pull abgebrochen

Remote Branch

- Einem lokalen Branch einen Remote Upstream hinzufügen
`git branch --set-upstream <remote> <branch>`
- <branch> ist optional
- Verknüpft lokalen Branch mit Remote Branch
- Lokaler Branch lässt sich mit `git reset` auf Stand des Remote Branches zurücksetzen

```
git reset --hard <remote>/<remote-branch>
```

Push

- Lokale Änderungen ins Remote Repository übertragen
`git push <remote> <remote-branch>`
- Ohne <remote> und <remote-branch> wird konfigurierter Upstream genutzt
- Mittels -u kann auch beim push ein Upstream eingerichtet werden
`git push -u <remote> <remote-branch>`
 - Kurzform für `git branch --set-upstream` und `git push`
 - Üblich bei neuen Branches, die Remote noch nicht existieren

Push

- `--all` pushed *alle* Branches
- `--tags` pushed zusätzlich zu dem angegebenen Branch alle Tags
- `--force` oder `-f` ermöglicht Push, auch wenn Commit-Historie nicht zusammenpasst (z.B. nach Rebase)

Push

- Schreibrechte auf jeweilige Remote benötigt
- Muss via Fast-Forward Merge im Remote Repository eingebaut werden können
- Git erlaubt ausschließlich Push in Bare Repositories