dataport

anderScore
trust in competence

Tag 1: Einführung in Git und GitLab, Git-Workflow im Team

17.06.2024, Daniel Krämer & Malte Fischer

Agenda

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

Agenda

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

Arbeiten mit einem Git

Remote Repository

Remote Repository

- Arbeiten mit Remote Repository ist eine zentrale Komponente in der Nutzung von Git mit mehreren Personen
- Auch oft einfach nur als Remote bezeichnet
- Wird im Netzwerk gehostet
  - Kann als einfaches Git Repository auf eigenem Server gehostet werden
  - Andere Konzepte wie GitLab, GitHub, BitBucket usw. ergänzen reines Hosting von Repositories um viele nützliche Features und sind weit verbreitet
- Unterscheidet sich technisch nicht von einem lokalen Repository
  - Man könnte bspw. das Repository des Kollegen bei sich als Remote Repository hinzufügen
  - Remote Repositories sind häufig sogenannte **Bare** Repositories, besitzen also kein lokalen Workspace
  - Bare Repositories können keine Dateien auschecken und daher auch keine potenziellen Konflikte lösen
- Man kann mehrere Remotes zu einem lokalen Repository hinzufügen

Remote Repository

anderScore trust in competence

Clonen eines vorhandenen Remote Repositories

- Erzeugt eine lokale Kopie (**local**) eines bereits existierenden Remote Repository (**origin**) mittels

```
git clone <remote-refernce>
```

- Erstellt Verknüpfung zwischen **local** und **origin**, um Synchronisation zu ermöglichen (`push`, `pull`, `fetch`, …)
- **local** besteht zunächst nur aus **default** Branch
- Informationen über andere Branches werden trotzdem abgerufen und können auch read-only mittels `git checkout origin/<branch>` angeschaut werden (**Remote Tracking** Bereich)
- Erst durch `git checkout <branch>` wird eine volle lokale Kopie in **local** angelegt und der zugehörige Remote Branch mit diesem als **Upstream** verknüpft
- **Upstream** definiert den zu einem lokalen Branch zugehörigen Branch, auf dem beim `fetch`, `pull` oder `push` zugegriffen wird

Remote Repository

anderScore trust in competence

Beispiel clonen 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
...
```

Remote Repository

Hinzufügen eines Remote Repository

- Um einem lokalen Repository ein Remote hinzuzufügen

```
git remote add <name> <remote-uri>
```

- `git remote add` legt alias für die Remote Addresse an und ruft Informationen über verfügbare Branches ab, läd diese aber nicht herunter
- Kann sowohl genutzt werden, um nach `git clone` weitere Remotes hinzuzufügen, als auch um initiales Remote anzugeben (ohne vorher geklont zu haben)
- 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
```

Remote Repository

anderScore trust in competence

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 Repositories
- 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
```

Remote Repository

anderScore trust in competence

Pull Befehl

- `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
```

Remote Repository

- Mergeverhalten kann ebenfalls konfiguriert warden

  git config --global pull.ff [true | only | false]

  - **true** (Default)
    Versucht beim Pull ein Fast-Forward Merge durchzuführen. Falls nicht möglich, wird ein Merge-Commit erstellt
  - **false**
    Kein FF, Merge-Commit wird immer erstellt
  - **only**
    Nur FF, wenn nicht möglich wird pull abgebrochen

Remote Repository

anderScore trust in competence

Remote Branch

- Einem lokalen Branch einen Remote Upstream hinzufügen

```
git branch --set-upstream <remote> <branch>
```

- `<branch>` ist optional und kann bei gleicher Benennung weggelassen werden
- Verknüpft lokalen Branch mit Remote Branch und ermöglicht `fetch`, `pull` und `push` Operationen
- Ein lokaler Branch lässt sich mit `git reset` auf den Remote Branch zurücksetzen

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

Remote Repository

anderScore trust in competence

Push Befehl

- Lokale Änderungen ins Remote Repository übertragen

```
git push <remote> <remote-branch>
```

- Ohne explizite Angabe von `<remote>` und `<remote-branch>` wird der konfigurierte 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
- `--all` ermöglicht pushen von allen Branches
- `--tags` pushed zusätzlich zu dem angebenen Branch alle Tags
- `--force` oder `-f` ermöglicht Push, auch wenn Commit-Historie nicht zusammenpasst

Remote Repository

- Für einen Push benötigt man Schreibrechte auf dem jeweiligen Remote Repository
- Änderungen des Pushes müssen via Fast-Forward Merge im Remote Repository eingebaut werden können
  - Es dürfen keine Commits im Remote Repository vorhanden sein, die lokal nicht existieren
  - Ggf. lokalen Branch via pull updaten, um Konflikt zu beheben
- Git erlaubt ausschließlich Push in Bare Repositories
  - Push in ein Repository mit lokalem Workspace würde inkonsistenten Zustand zwischen lokalem Workspace und Remote Tracking Bereich verursachen