

Eine Einführung in Git

Paul Nykiel

3. Mai 2020

1 Einleitung

2 Konzept

3 Nutzung

4 GitLab

5 Git bei uns

6 Abschluss

7 Praxis

Einleitung

Konzept

Nutzung

GitLab

Git bei uns

Abschluss

Praxis

Einleitung

Wofür ein Versionsverwaltungssystem?

Situation: Mehrere Leute arbeiten über längere Zeit an einer Codebase

Wofür ein Versionsverwaltungssystem?

Situation: Mehrere Leute arbeiten über längere Zeit an einer Codebase

Probleme:

Wofür ein Versionsverwaltungssystem?

Situation: Mehrere Leute arbeiten über längere Zeit an einer Codebase

Probleme:

- Datei Austausch zwischen Nutzer

Wofür ein Versionsverwaltungssystem?

Situation: Mehrere Leute arbeiten über längere Zeit an einer Codebase

Probleme:

- Datei Austausch zwischen Nutzer
- Aber: nicht sofort, erst wenn fertig

Wofür ein Versionsverwaltungssystem?

Situation: Mehrere Leute arbeiten über längere Zeit an einer Codebase

Probleme:

- Datei Austausch zwischen Nutzer
- Aber: nicht sofort, erst wenn fertig
- Alte Codestände sollten getestet werden können

Wofür ein Versionsverwaltungssystem?

Einleitung

Konzept

Nutzung

GitLab

Git bei uns

Abschluss

Praxis

Situation: Mehrere Leute arbeiten über längere Zeit an einer Codebase

Probleme:

- Datei Austausch zwischen Nutzer
- Aber: nicht sofort, erst wenn fertig
- Alte Codestände sollten getestet werden können

Einleitung

Konzept

Nutzung

GitLab

Git bei uns

Abschluss

Praxis

Was noch?

- Authentisierung: von wem ist der Code

Einleitung

Konzept

Nutzung

GitLab

Git bei uns

Abschluss

Praxis

Was noch?

- Authentisierung: von wem ist der Code
- Kein permanenter Internetzugriff

Einleitung

Konzept

Nutzung

GitLab

Git bei uns

Abschluss

Praxis

Was noch?

- Authentisierung: von wem ist der Code
- Kein permanenter Internetzugriff
- Einfache Nutzung

Einleitung

Konzept

Nutzung

GitLab

Git bei uns

Abschluss

Praxis

Was noch?

- Authentisierung: von wem ist der Code
- Kein permanenter Internetzugriff
- Einfache Nutzung
- Schnell

Einleitung

Konzept

Nutzung

GitLab

Git bei uns

Abschluss

Praxis

Was noch?

- Authentisierung: von wem ist der Code
- Kein permanenter Internetzugriff
- Einfache Nutzung
- Schnell
- Sicher

Einleitung

Konzept

Nutzung

GitLab

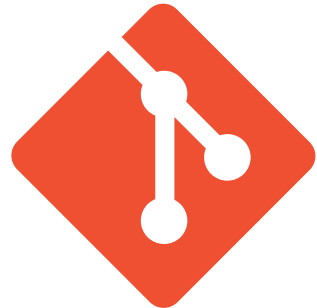
Git bei uns

Abschluss

Praxis

- Freie Software zur Versionsverwaltung

Git



Einleitung

Konzept

Nutzung

GitLab

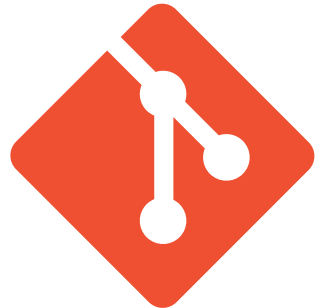
Git bei uns

Abschluss

Praxis

- Freie Software zur Versionsverwaltung
- Dezentral

Git



Einleitung

Konzept

Nutzung

GitLab

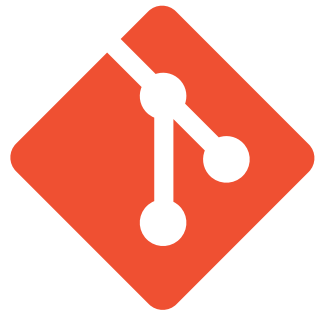
Git bei uns

Abschluss

Praxis

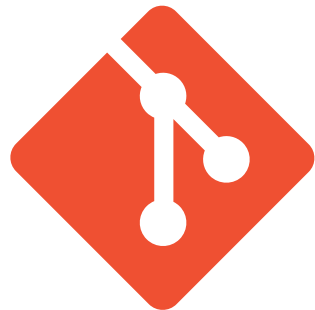
- Freie Software zur Versionsverwaltung
- Dezentral
- Wurde 2005 von Linus Torvals für Linux initiiert

Git



- Freie Software zur Versionsverwaltung
- Dezentral
- Wurde 2005 von Linus Torvals für Linux initiiert
- Defakto Standard

Git

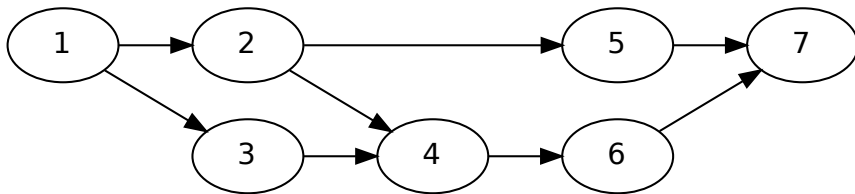


Konzept

- Repository: Eine zusammenhängende Codebase/Projekt

- Repository: Eine zusammenhängende Codebase/Projekt
- Ein Repository ist ein (azyklischer) gerichteter Graph aus Commits

- Repository: Eine zusammenhängende Codebase/Projekt
- Ein Repository ist ein (azyklischer) gerichteter Graph aus Commits



Commit

- Code-Stand

Commit

- Code-Stand
- Eindeutige
Bezeichnung

Commit

- Code-Stand
- Eindeutige
Bezeichnung
- Autor

Commit

- Code-Stand
- Eindeutige
Bezeichnung
- Autor
- Datum

- Code-Stand
- Eindeutige
Bezeichnung
- Autor
- Datum
- Nachricht

```
commit feeecb67fe1fa0490a2b836d5ba35da5812a3d27
```

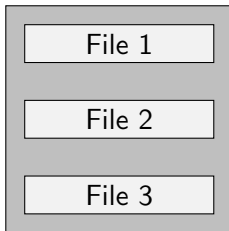
```
Author: Paul Nykiel
```

```
Date:   Mon Apr 20 22:55:41 2020 +0200
```

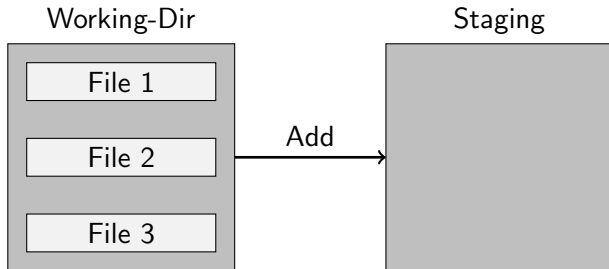
```
Added ADTF type header
```

Einen Commit anlegen

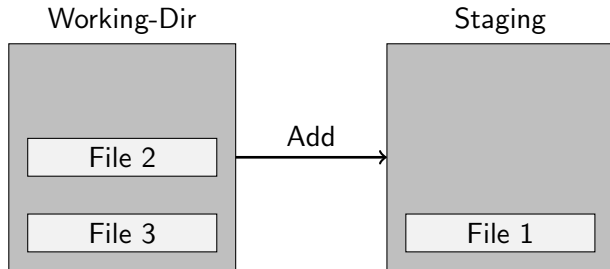
Working-Dir



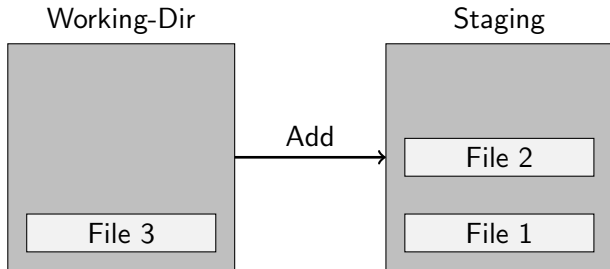
Einen Commit anlegen



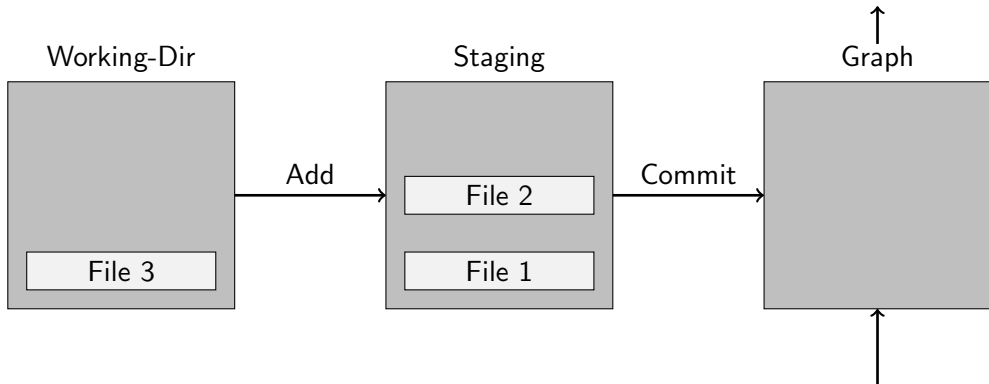
Einen Commit anlegen



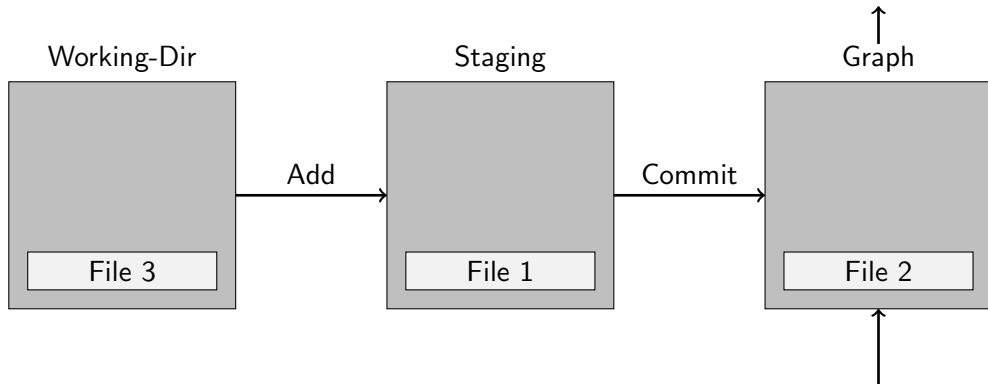
Einen Commit anlegen



Einen Commit anlegen

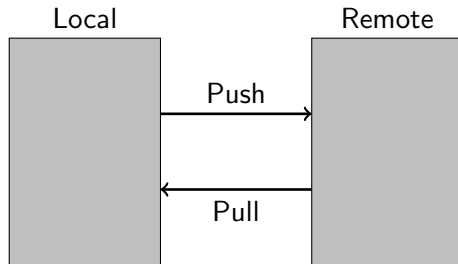


Einen Commit anlegen



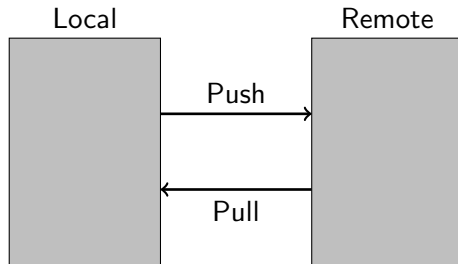
- Bis jetzt alles lokal

Remotes



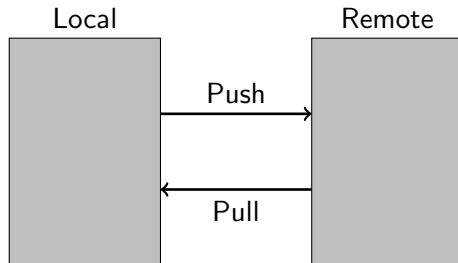
- Bis jetzt alles lokal
- Codebase auf anderem Host:
„Remote“

Remotes



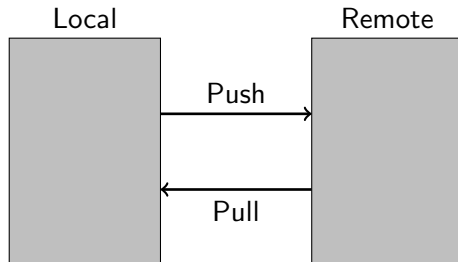
- Bis jetzt alles lokal
- Codebase auf anderem Host: „Remote“
- Operationen: Commits von Remote kopieren, Commits zu Remote kopieren

Remotes



- Bis jetzt alles lokal
- Codebase auf anderem Host: „Remote“
- Operationen: Commits von Remote kopieren, Commits zu Remote kopieren
- Oftmals ein zentraler Server

Remotes



Branches

Einleitung

Konzept

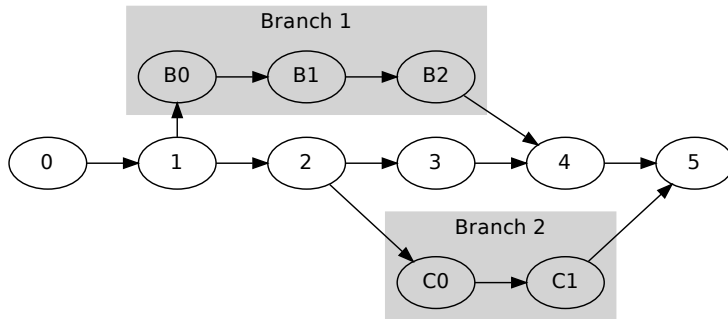
Nutzung

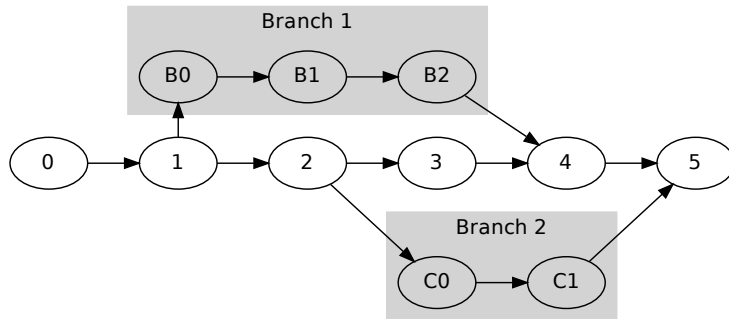
GitLab

Git bei uns

Abschluss

Praxis





Ein Branch ist eine Liste von Commits

Nutzung

- Manipulation des Zustandes über Git (<https://git-scm.com/>)

- Manipulation des Zustandes über Git (<https://git-scm.com/>)
- Nutzung: `git command arguments`

- Manipulation des Zustandes über Git (<https://git-scm.com/>)
- Nutzung: `git command arguments`
- Häufige Befehle:

- Manipulation des Zustandes über Git (<https://git-scm.com/>)
- Nutzung: `git command arguments`
- Häufige Befehle:
 - `add`

- Manipulation des Zustandes über Git (<https://git-scm.com/>)
- Nutzung: `git command arguments`
- Häufige Befehle:
 - `add`
 - `commit`

- Manipulation des Zustandes über Git (<https://git-scm.com/>)
- Nutzung: `git command arguments`
- Häufige Befehle:
 - `add`
 - `commit`
 - `merge`

- Manipulation des Zustandes über Git (<https://git-scm.com/>)
- Nutzung: `git command arguments`
- Häufige Befehle:
 - `add`
 - `commit`
 - `merge`
 - `checkout`

- Manipulation des Zustandes über Git (<https://git-scm.com/>)
- Nutzung: `git command arguments`
- Häufige Befehle:
 - `add`
 - `commit`
 - `merge`
 - `checkout`
 - `push`

- Manipulation des Zustandes über Git (<https://git-scm.com/>)
- Nutzung: `git command arguments`
- Häufige Befehle:
 - `add`
 - `commit`
 - `merge`
 - `checkout`
 - `push`
 - `pull`

- Manipulation des Zustandes über Git (<https://git-scm.com/>)
- Nutzung: `git command arguments`
- Häufige Befehle:
 - `add`
 - `commit`
 - `merge`
 - `checkout`
 - `push`
 - `pull`
 - `status`

- Manipulation des Zustandes über Git (<https://git-scm.com/>)
- Nutzung: `git command arguments`
- Häufige Befehle:
 - `add`
 - `commit`
 - `merge`
 - `checkout`
 - `push`
 - `pull`
 - `status`
 - `log`

Add

```
git add FILES...
```

Add

```
git add FILES...
```

```
git add -A
```

Add

```
git add FILES...
```

```
git add -A
```

```
git add -u
```

Commit

```
git commit -m "message"
```

Merge

```
git merge BRANCH
```


Checkout

`git checkout IDENTIFIER`
mit Identifier: Branch Name, Commit-Hash, ...

Checkout

`git checkout IDENTIFIER`
mit Identifier: Branch Name, Commit-Hash, ...

`git checkout -b NEW_BRANCH`

Push

```
git push REMOTE BRANCH
```

Pull

```
git pull REMOTE BRANCH
```

Information

```
git status
```

Information

```
git status
```

```
git log
```

Demo: Einfache Nutzung

GitLab

GitLab ist ein Git-Server („Remote“) mit Web-Frontend

GitLab ist ein Git-Server („Remote“) mit Web-Frontend

Features:

GitLab ist ein Git-Server („Remote“) mit Web-Frontend

Features:

- Zugriffskontrolle

GitLab ist ein Git-Server („Remote“) mit Web-Frontend

Features:

- Zugriffskontrolle
- Merge Requests

GitLab ist ein Git-Server („Remote“) mit Web-Frontend

Features:

- Zugriffskontrolle
- Merge Requests
- Issue Tracker

GitLab ist ein Git-Server („Remote“) mit Web-Frontend

Features:

- Zugriffskontrolle
- Merge Requests
- Issue Tracker
- Continuous Integration

Demo: GitLab

Git bei uns

- Definition von Abläufen, die für stabilen, aber auch aktuellen Code sorgen sollen
- Zwei langläufige Branches: `master` und `develop`



Abschluss

Praxis

Repository-Einrichten

- Legt ein Repository mit eurem Namen in der Gruppe GitWorkshop an

Repository-Einrichten

- Legt ein Repository mit eurem Namen in der Gruppe GitWorkshop an
- Erzeugt eine lokale Kopie des Repositories auf eurem Rechner (`clone`)

Repository-Einrichten

- Legt ein Repository mit eurem Namen in der Gruppe GitWorkshop an
- Erzeugt eine lokale Kopie des Repositories auf eurem Rechner (`clone`)
- Legt eine Datei `.gitignore` an

Repository-Einrichten

- Legt ein Repository mit eurem Namen in der Gruppe GitWorkshop an
- Erzeugt eine lokale Kopie des Repositories auf eurem Rechner (`clone`)
- Legt eine Datei `.gitignore` an
- Legt die Branchstruktur für Git-Flow an

Repository-Einrichten

- Legt ein Repository mit eurem Namen in der Gruppe GitWorkshop an
- Erzeugt eine lokale Kopie des Repositories auf eurem Rechner (`clone`)
- Legt eine Datei `.gitignore` an
- Legt die Branchstruktur für Git-Flow an
- Legt ein Issue an und setzt euch als Verantwortlichen

Ein erster MR

- Legt einen Feature Branch an

Ein erster MR

- Legt einen Feature Branch an
- Committed den Code vom C++-Workshop (Teil 2) auf den Feature-Branch

Ein erster MR

- Legt einen Feature Branch an
- Committed den Code vom C++-Workshop (Teil 2) auf den Feature-Branch
- Legt einen WIP-MR dafür an, setzt mich als Verantwortlichen

Ein erster MR

- Legt einen Feature Branch an
- Committed den Code vom C++-Workshop (Teil 2) auf den Feature-Branch
- Legt einen WIP-MR dafür an, setzt mich als Verantwortlichen
- Ich werde euch Feedback in Form von Anmerkungen geben

Ein erster MR

- Legt einen Feature Branch an
- Committed den Code vom C++-Workshop (Teil 2) auf den Feature-Branch
- Legt einen WIP-MR dafür an, setzt mich als Verantwortlichen
- Ich werde euch Feedback in Form von Anmerkungen geben
- Arbeitet die Anmerkungen ein

MR mergen

- Fügt zu eurem WIP-MR den Code vom C++ Workshop Teil 3 hinzu

MR mergen

- Fügt zu eurem WIP-MR den Code vom C++ Workshop Teil 3 hinzu
- Ich werde wieder Feedback geben, das ihr einarbeiten sollt

MR mergen

- Fügt zu eurem WIP-MR den Code vom C++ Workshop Teil 3 hinzu
- Ich werde wieder Feedback geben, das ihr einarbeiten sollt
- Wenn alles OK ist bekommt ihr einen 👍

MR mergen

- Fügt zu eurem WIP-MR den Code vom C++ Workshop Teil 3 hinzu
- Ich werde wieder Feedback geben, das ihr einarbeiten sollt
- Wenn alles OK ist bekommt ihr einen 👍
- Dann könnt ihr den MR mergen

MR mergen

- Fügt zu eurem WIP-MR den Code vom C++ Workshop Teil 3 hinzu
- Ich werde wieder Feedback geben, das ihr einarbeiten sollt
- Wenn alles OK ist bekommt ihr einen 👍
- Dann könnt ihr den MR mergen
- Aktualisiert euere lokalen Branches