

Git Version Control Professional

Prüfungscode IFW-GIT-300-01 · Fassung 1.0 · Stand 18.09.2026

Git für Fortgeschrittene: Historie umschreiben, integrieren und wiederherstellen, Historie analysieren, Konfiguration und Automatisierung sowie große Repositorys.

Die Prüfung im Überblick

Bearbeitungszeit 2 Stunden	Fragen 70
Bestehensgrenze 70 %	Fragetypen Einfach- und Mehrfachauswahl
Prüfungssprache Deutsch	Zertifikat gültig 36 Monate
Durchführung Online, zeitlich überwacht	Preis 99,00 €

Themenblöcke

Block	Thema	Gewichtung	Fragen
1	Historie umschreiben, integrieren und wiederherstellen	30 %	21
2	Historie analysieren	30 %	21
3	Konfiguration und Automatisierung	20 %	14
4	Arbeitskopien und große Repositorys	20 %	14
Gesamt		100 %	70

Die Gewichtung gibt an, welchen Anteil ein Block an den Prüfungsfragen hat. Die Fragen werden beim Start zufällig und anteilig je Block aus dem Fragenpool gezogen.

Historie umschreiben, integrieren und wiederherstellen

Commits gezielt umbauen und übertragen – und Fehler dabei sicher wieder rückgängig machen.

- 1.1** Interaktiv rebasen
`pick, reword, edit, squash, fixup, drop; git rebase --onto`
- 1.2** Einzelne Commits übertragen
`git cherry-pick`, Option `-x`, Konflikte
- 1.3** Merge-Strategien unterscheiden
`-s ours` gegenüber `-X ours`; Konfliktlösungen wiederverwenden mit `rerere`
- 1.4** Umgeschriebene Historie sicher veröffentlichen
`--force-with-lease` und seine Grenzen bei automatischem `fetch`
- 1.5** Verlorene Commits wiederherstellen
`git reflog`, `git reset` auf einen früheren Stand
- 1.6** Merges rückgängig machen
`git revert -m 1` bei Merge-Commits
- 1.7** Versehentlich veröffentlichte Geheimnisse behandeln
Zugangsdaten sofort ändern, Historie mit `git filter-repo` bereinigen

Historie analysieren

Herausfinden, wann, warum und von wem eine Änderung eingeführt wurde.

- 2.1** Revisionen adressieren
`HEAD~n`, `HEAD^2` bei Merge-Commits
- 2.2** Bereiche vergleichen
`main..feature` gegenüber `main...feature` bei `log` und `diff`
- 2.3** Änderungen im Code suchen
`git log -S` und `-G`
- 2.4** Zeilen ihrem Ursprung zuordnen
`git blame`, verschobene Zeilen mit `-M` und `-C`, Commits ignorieren
- 2.5** Fehler per Binärsuche finden
`git bisect`, automatisiert mit `git bisect run` und Exit-Codes

Konfiguration und Automatisierung

Das Verhalten von Git je Repository festlegen und Arbeitsschritte absichern.

- 3.1** Git-Hooks einsetzen
pre-commit und pre-receive, Ablage in `.git/hooks`; clientseitige Hooks sind umgehbar
- 3.2** Dateiattribute festlegen
`.gitattributes` für Zeilenenden, Binärdateien, Diff- und Merge-Treiber
- 3.3** Commits signieren
`git commit -s` mit GPG oder SSH und was eine Signatur bestätigt

Arbeitskopien und große Repositorys

Mit mehreren Ständen gleichzeitig und mit sehr großen Repositorys arbeiten.

- 4.1** Mehrere Arbeitsverzeichnisse nutzen
`git worktree add`
- 4.2** Nur Teile eines Repositorys laden
Sparse Checkout, Partial Clone mit `--filter`
- 4.3** Fremde Repositorys einbinden
Submodules gegenüber `git subtree`

Institut für Webentwicklung · unabhängiges Lernangebot ohne Verbindung zu den genannten Technologieanbietern. Maßgeblich ist die Fassung auf der Zertifizierungsseite.