Git

Anton Obersteiner

Python-Kurs

25. Januar 2022

Gliederung

1. Wozu?
   - Beispiele
2. Dateien/Änderungen verschieben
3. Befehle: `git <...>`
   - Ebenen
   - Basic
   - Management
   - Online/Remote
   - Kultur am Ende

Wozu?

- Versionen und Änderungen speichern
- alte Zustände wiederherstellen
- Verschiedene Änderungen vergleichen
- und zusammenführen

Dateien/Änderungen verschieben

Code Module getrennt entwickeln

1. In § 1 Nummer 1 werden die Wörter „§ 3a Absatz 1 Nummer 1 Buchstabe a“ durch die Wörter „§ 3a Absatz 1 Nummer 1 Buchstabe a und c sowie Nummer 2 Buchstabe a“ ersetzt.

Abbildung: Erste Verordnung zur Änderung der Mobilitätsdatenverordnung Vom 6. Januar 2022

Dateien/Änderungen verschieben

Code Module getrennt entwickeln

Features entwickeln, ohne Funktionierendes zu gefärden

1. In § 1 Nummer 1 werden die Wörter „§ 3a Absatz 1 Nummer 1 Buchstabe a“ durch die Wörter „§ 3a Absatz 1 Nummer 1 Buchstabe a und c sowie Nummer 2 Buchstabe a“ ersetzt.

Abbildung: Erste Verordnung zur Änderung der Mobilitätsdatenverordnung Vom 6. Januar 2022

Dateien/Änderungen verschieben

Code Module getrennt entwickeln

Features entwickeln, ohne Funktionierendes zu gefärden

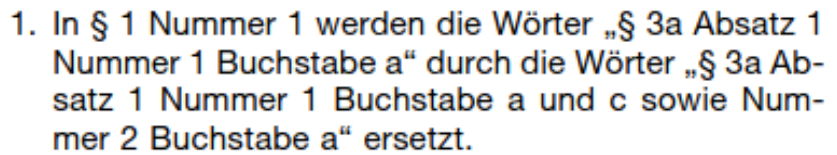
Bugs Finden: durch Vergleich von älteren Zuständen

1. In § 1 Nummer 1 werden die Wörter „§ 3a Absatz 1 Nummer 1 Buchstabe a“ durch die Wörter „§ 3a Absatz 1 Nummer 1 Buchstabe a und c sowie Nummer 2 Buchstabe a“ ersetzt.

Abbildung: Erste Verordnung zur Änderung der Mobilitätsdatenverordnung Vom 6. Januar 2022

Dateien/Änderungen verschieben

Code	Module getrennt entwickeln
Features	entwickeln, ohne Funktionierendes zu gefärden
Bugs	Finden: durch Vergleich von älteren Zuständen
Texte	Dokumentation & sonstige gemeinsame Texte

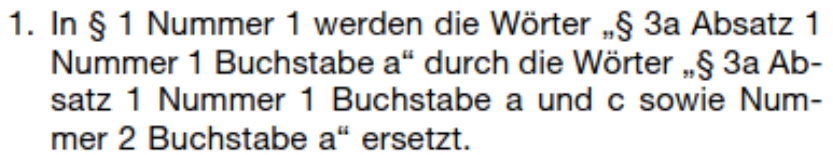


1. In § 1 Nummer 1 werden die Wörter „§ 3a Absatz 1 Nummer 1 Buchstabe a“ durch die Wörter „§ 3a Absatz 1 Nummer 1 Buchstabe a und c sowie Nummer 2 Buchstabe a“ ersetzt.

Abbildung: Erste Verordnung zur Änderung der Mobilitätsdatenverordnung
Vom 6. Januar 2022

Dateien/Änderungen verschieben

Code	Module getrennt entwickeln
Features	entwickeln, ohne Funktionierendes zu gefärden
Bugs	Finden: durch Vergleich von älteren Zuständen
Texte	Dokumentation & sonstige gemeinsame Texte
Recht	Gesetzesänderungen transparent darstellen



1. In § 1 Nummer 1 werden die Wörter „§ 3a Absatz 1 Nummer 1 Buchstabe a“ durch die Wörter „§ 3a Absatz 1 Nummer 1 Buchstabe a und c sowie Nummer 2 Buchstabe a“ ersetzt.

Abbildung: Erste Verordnung zur Änderung der Mobilitätsdatenverordnung
Vom 6. Januar 2022

Befehle: `git <...>`: Ebenen

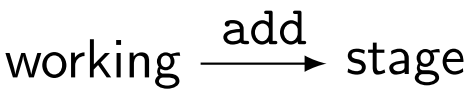
working Inhalt des Ordners

working

Befehle: `git <...>`: Ebenen

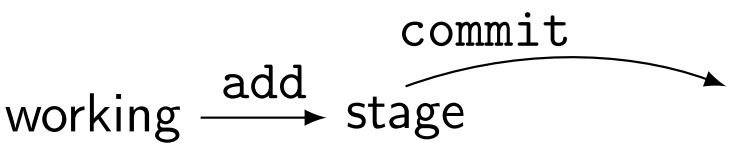
working Inhalt des Ordners

stage “Auf dem Tisch”, Fertig für Commit



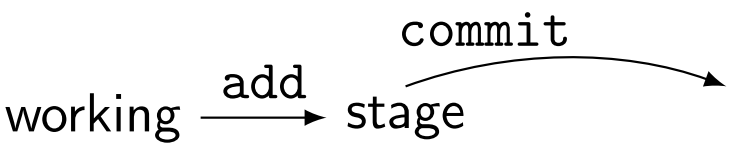
Befehle: `git <...>`: Ebenen

working	Inhalt des Ordners
stage	"Auf dem Tisch", Fertig für Commit
commit	Paket von Änderungen (klein, "nur eine Sache")



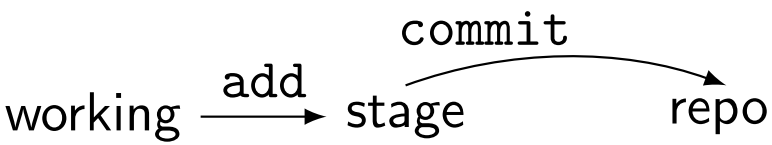
Befehle: `git <...>`: Ebenen

working	Inhalt des Ordners
stage	"Auf dem Tisch", Fertig für Commit
commit	Paket von Änderungen (klein, "nur eine Sache")
branch	Reihe zusammenhängender Commits (ein größeres Thema)



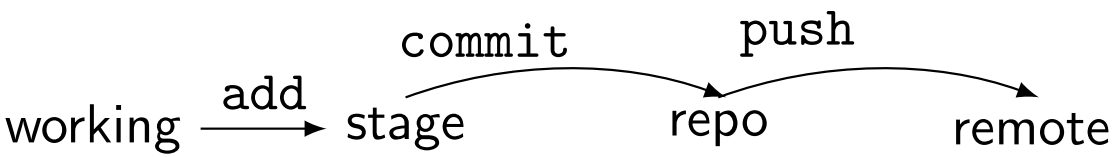
Befehle: `git <...>`: Ebenen

working	Inhalt des Ordners
stage	"Auf dem Tisch", Fertig für Commit
commit	Paket von Änderungen (klein, "nur eine Sache")
branch	Reihe zusammenhängender Commits (ein größeres Thema)
repo	Gesamt-Sammlung der Commits



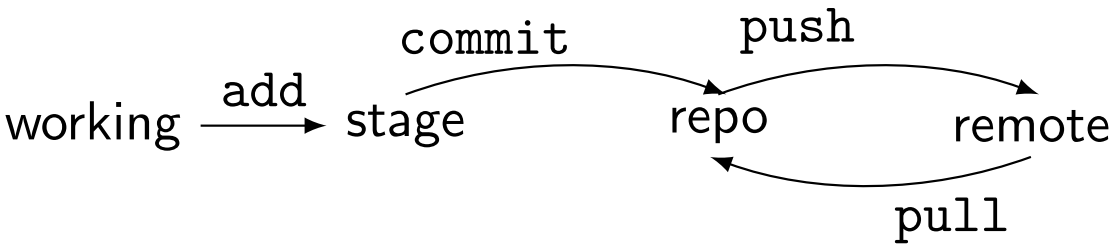
Befehle: `git <...>`: Ebenen

working	Inhalt des Ordners
stage	"Auf dem Tisch", Fertig für Commit
commit	Paket von Änderungen (klein, "nur eine Sache")
branch	Reihe zusammenhängender Commits (ein größeres Thema)
repo	Gesamt-Sammlung der Commits
remote	Online-Repo (Von gesamter Gruppe, ...)



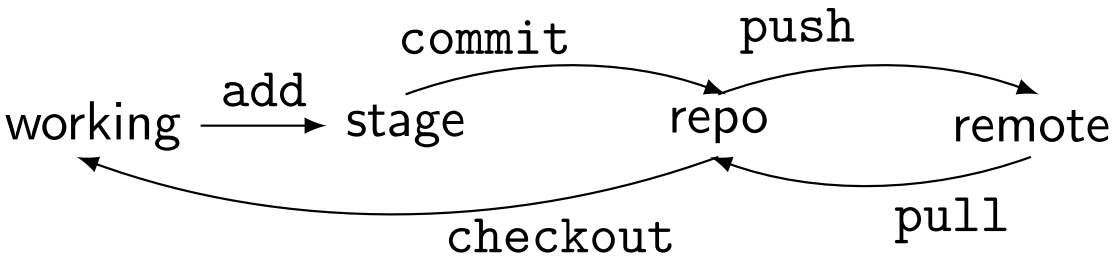
Befehle: `git <...>`: Ebenen

working	Inhalt des Ordners
stage	"Auf dem Tisch", Fertig für Commit
commit	Paket von Änderungen (klein, "nur eine Sache")
branch	Reihe zusammenhängender Commits (ein größeres Thema)
repo	Gesamt-Sammlung der Commits
remote	Online-Repo (Von gesamter Gruppe, ...)
commit	aktuelle Stage zusammenfassen und abspeichern



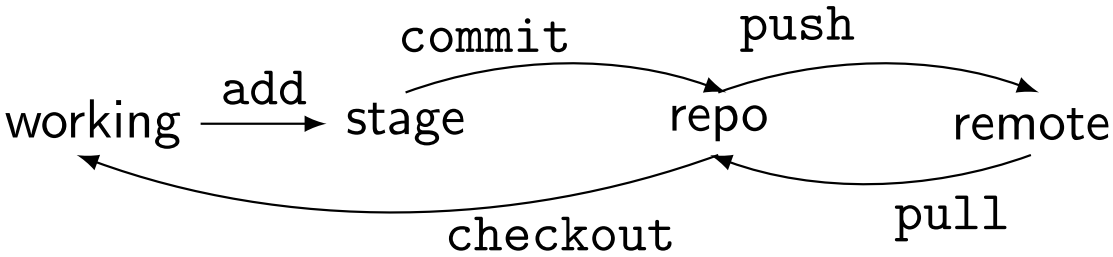
Befehle: `git <...>`: Ebenen

working	Inhalt des Ordners
stage	"Auf dem Tisch", Fertig für Commit
commit	Paket von Änderungen (klein, "nur eine Sache")
branch	Reihe zusammenhängender Commits (ein größeres Thema)
repo	Gesamt-Sammlung der Commits
remote	Online-Repo (Von gesamter Gruppe, ...)
commit	aktuelle Stage zusammenfassen und abspeichern



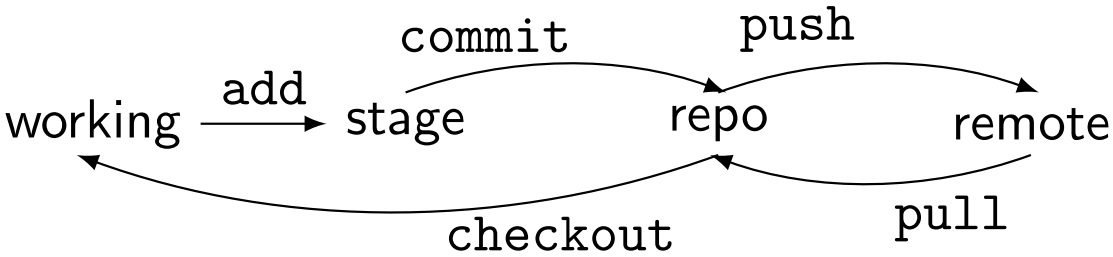
Befehle: `git <...>`: Basic

`init` neues GitRepo anlegen



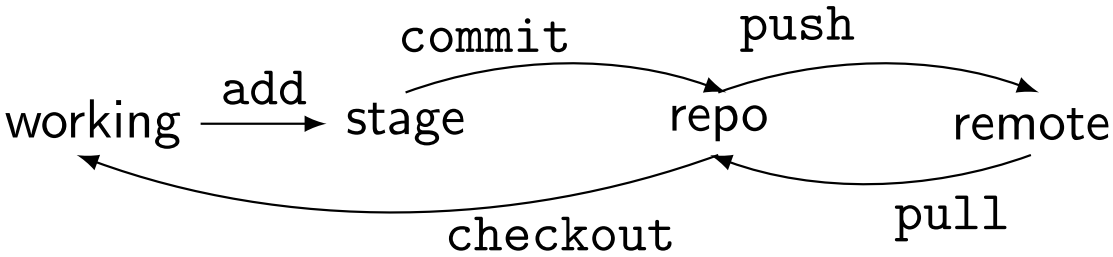
Befehle: `git <...>`: Basic

init	neues GitRepo anlegen
status	Was ist neu in Working und Stage



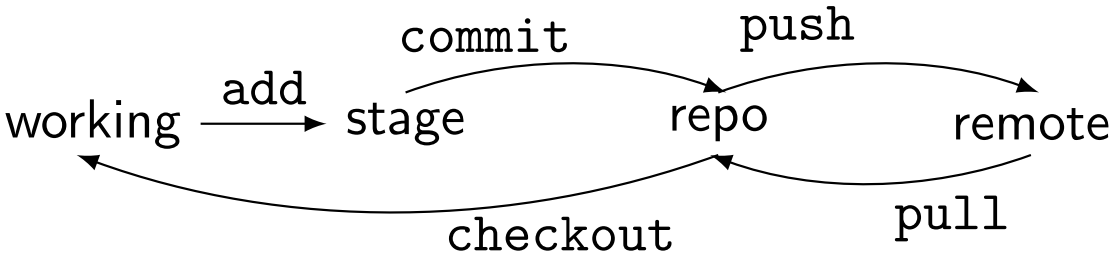
Befehle: `git <...>`: Basic

init	neues GitRepo anlegen
status	Was ist neu in Working und Stage
log	vergangene Commits/Änderungen



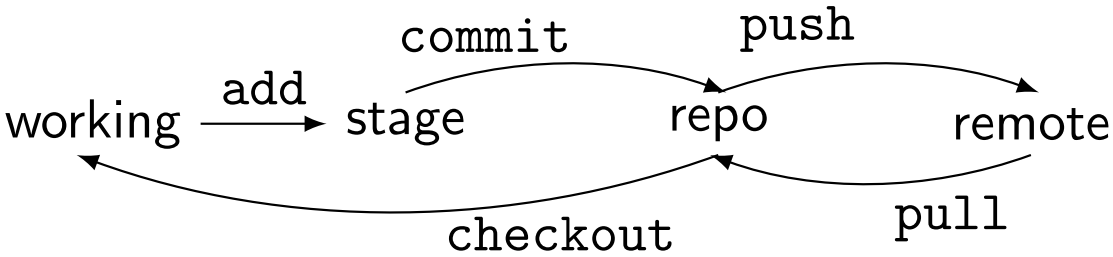
Befehle: `git <...>`: Basic

init	neues GitRepo anlegen
status	Was ist neu in Working und Stage
log	vergangene Commits/Änderungen
add	Dateien/Änderungen hinzufügen (zu Stage)



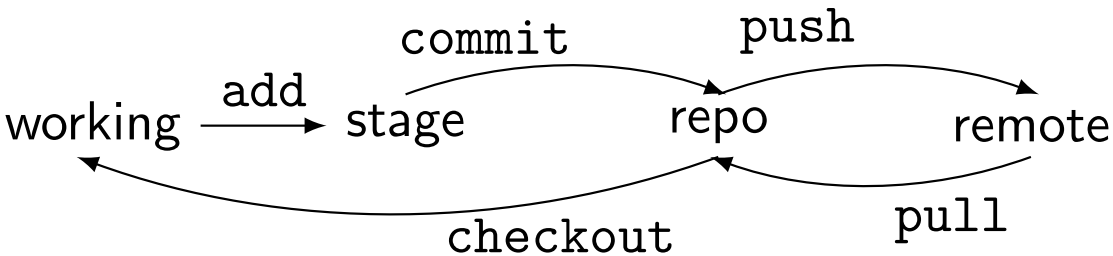
Befehle: `git <...>`: Basic

init	neues GitRepo anlegen
status	Was ist neu in Working und Stage
log	vergangene Commits/Änderungen
add	Dateien/Änderungen hinzufügen (zu Stage)
commit	aktuelle Stage zusammenfassen und abspeichern



Befehle: `git <...>`: Management

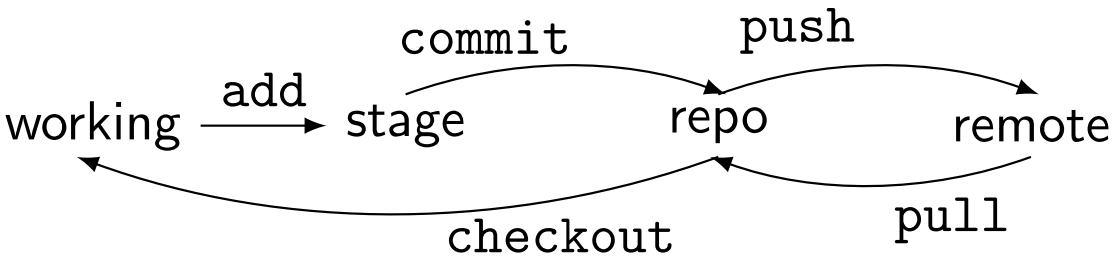
diff Unterschiede zwischen Repo und Working



Befehle: `git <...>`: Management

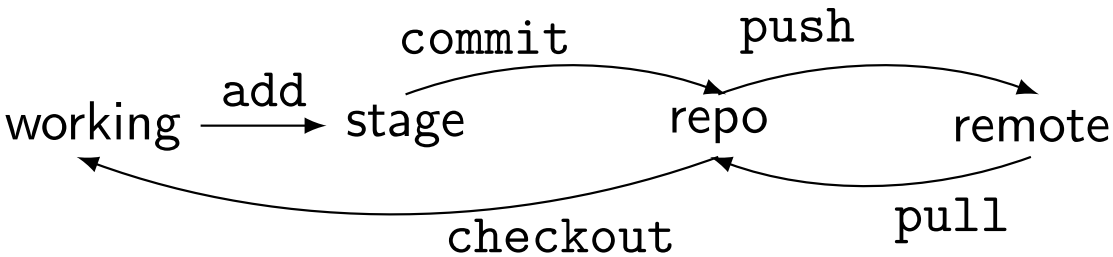
diff Unterschiede zwischen Repo und Working

rm Datei löschen (muss man committen)



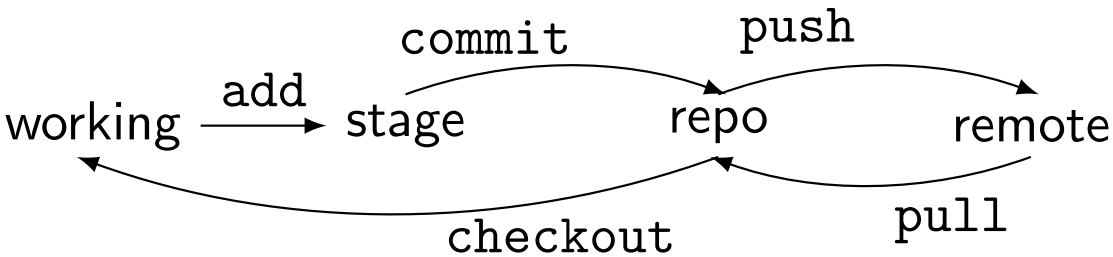
Befehle: `git <...>`: Management

diff	Unterschiede zwischen Repo und Working
rm	Datei löschen (muss man committen)
mv	Datei umbenennen (muss man committen)



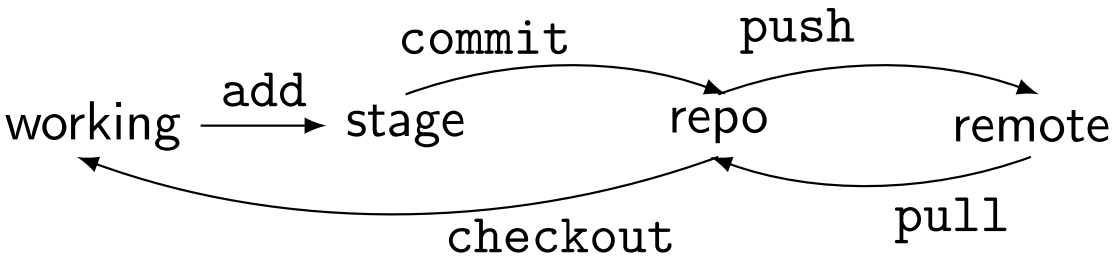
Befehle: `git <...>`: Management

diff	Unterschiede zwischen Repo und Working
rm	Datei löschen (muss man committen)
mv	Datei umbenennen (muss man committen)
checkout	Commit/Dateien aus Repo in Working laden



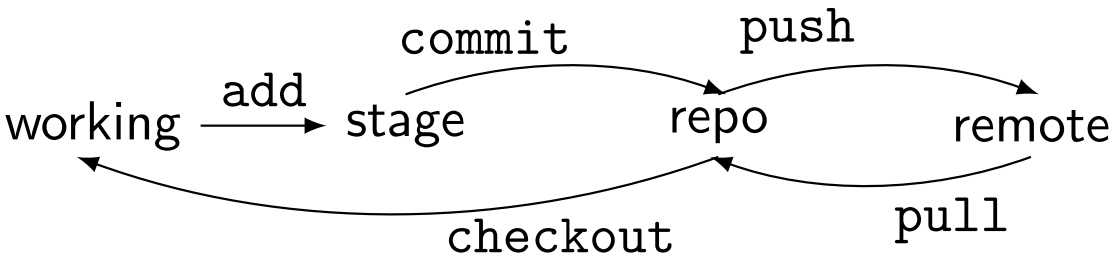
Befehle: `git <...>`: Management

diff	Unterschiede zwischen Repo und Working
rm	Datei löschen (muss man committen)
mv	Datei umbenennen (muss man committen)
checkout	Commit/Dateien aus Repo in Working laden
reset HEAD	aus Stage in Working schieben



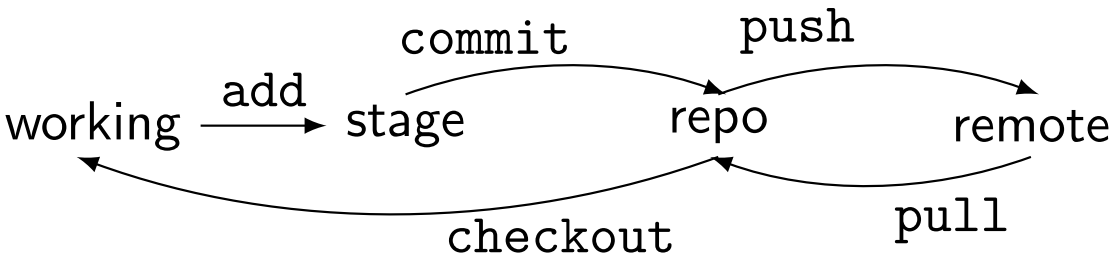
Befehle: `git <...>`: Management

diff	Unterschiede zwischen Repo und Working
rm	Datei löschen (muss man committen)
mv	Datei umbenennen (muss man committen)
checkout	Commit/Dateien aus Repo in Working laden
reset HEAD	aus Stage in Working schieben
branch	Ideen ausprobieren → nicht den Hauptzweig zuschreiben



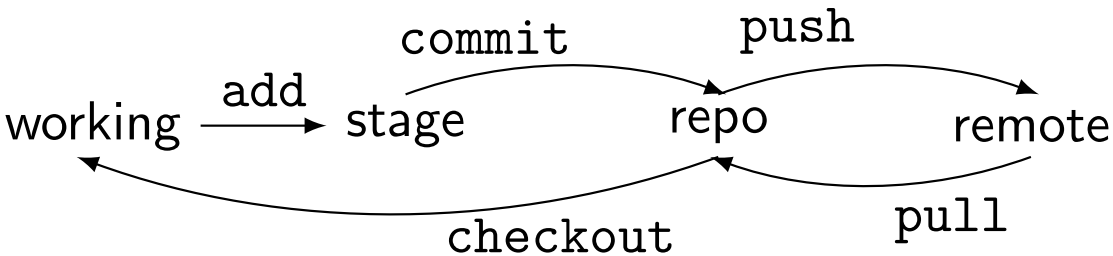
Befehle: `git <...>`: Management

diff	Unterschiede zwischen Repo und Working
rm	Datei löschen (muss man committen)
mv	Datei umbenennen (muss man committen)
checkout	Commit/Dateien aus Repo in Working laden
reset HEAD	aus Stage in Working schieben
branch	Ideen ausprobieren → nicht den Hauptzweig zuschreiben
merge	Änderungen in aktuellen Branch übernehmen



Befehle: `git <...>`: Online/Remote

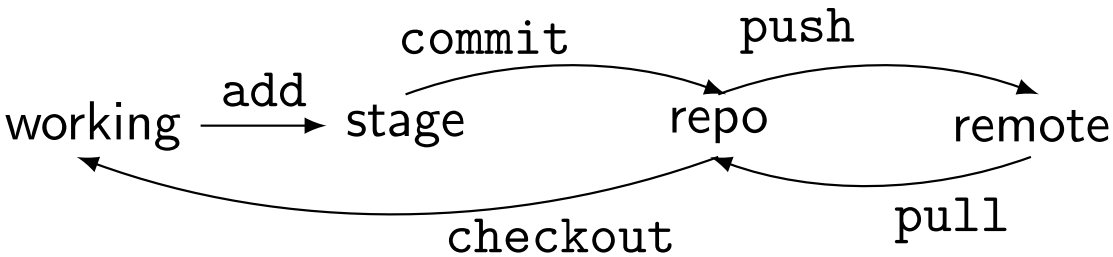
`remote` Online-repository anbinden



Befehle: `git <...>`: Online/Remote

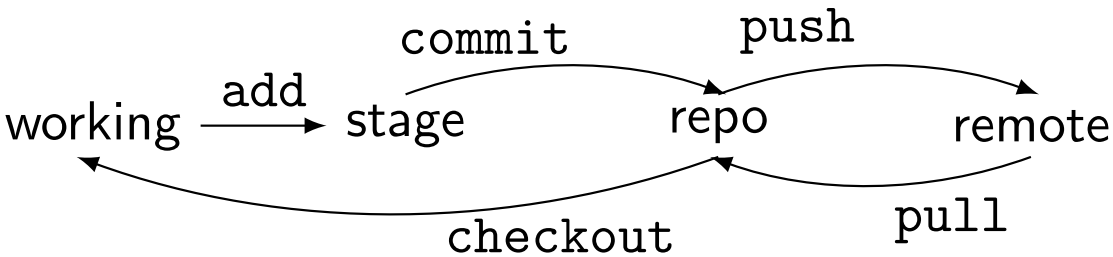
remote Online-repository anbinden

clone remote lokal kopieren (in Ordner ohne Git)



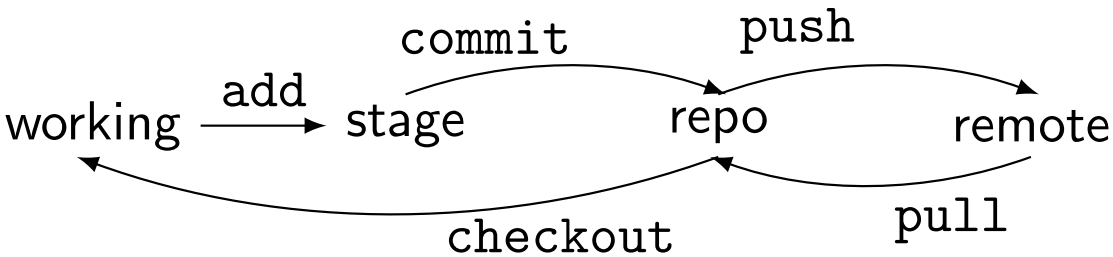
Befehle: `git <...>`: Online/Remote

remote	Online-repository anbinden
clone	remote lokal kopieren (in Ordner ohne Git)
fetch	remote anfragen



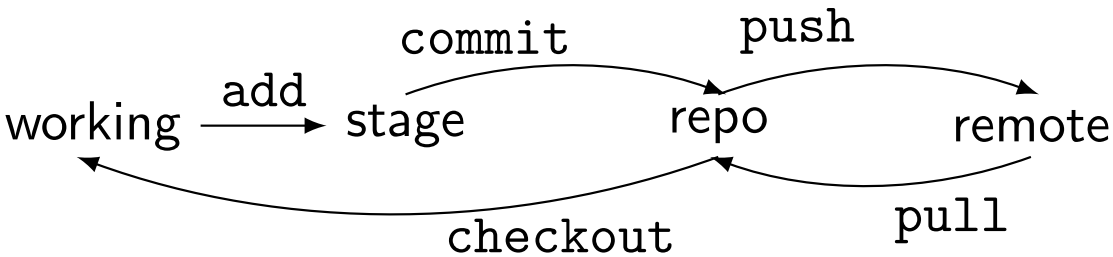
Befehle: `git <...>`: Online/Remote

remote	Online-repository anbinden
clone	remote lokal kopieren (in Ordner ohne Git)
fetch	remote anfragen
pull	remote in lokal mergen



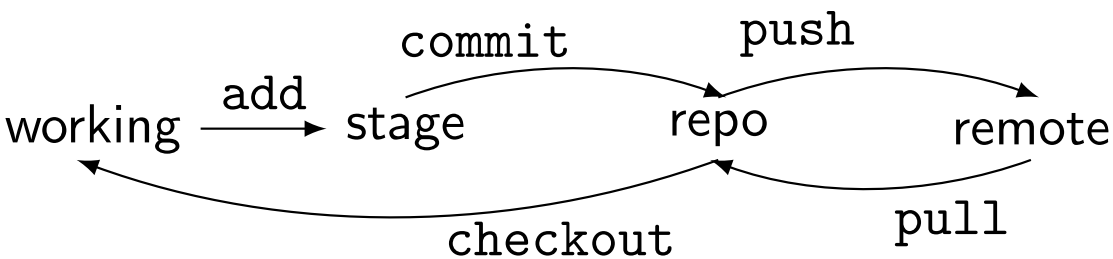
Befehle: `git <...>`: Online/Remote

remote	Online-repository anbinden
clone	remote lokal kopieren (in Ordner ohne Git)
fetch	remote anfragen
pull	remote in lokal mergen
push	lokal in remote mergen



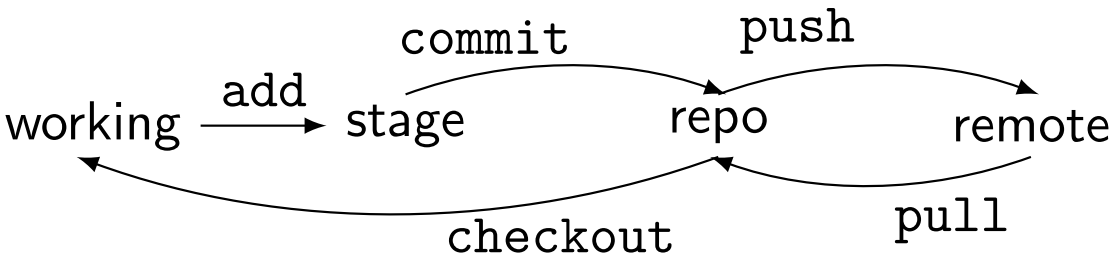
Befehle: `git <...>`: Online/Remote

remote	Online-repository anbinden
clone	remote lokal kopieren (in Ordner ohne Git)
fetch	remote anfragen
pull	remote in lokal mergen
push	lokal in remote mergen
fork	eigene Kopie von fremdem Repo erstellen



Befehle: `git <...>`: Online/Remote

remote	Online-repository anbinden
clone	remote lokal kopieren (in Ordner ohne Git)
fetch	remote anfragen
pull	remote in lokal mergen
push	lokal in remote mergen
fork	eigene Kopie von fremdem Repo erstellen
pull request	fremdem Repo Änderung vorschlagen



Kultur am Ende

commit Macht eine Sache (Plugin oder `git add -p`)

Kultur am Ende

commit Macht eine Sache (Plugin oder `git add -p`)

commit Kurze, aber präzise Beschreibung

Kultur am Ende

commit	Macht eine Sache (Plugin oder `git add -p`)
commit	Kurze, aber präzise Beschreibung
main	Stabiler Branch, in den man nur Fertiges mergt

Kultur am Ende

commit	Macht eine Sache (Plugin oder `git add -p`)
commit	Kurze, aber präzise Beschreibung
main	Stabiler Branch, in den man nur Fertiges mergt
develop	Für aktuellen gemeinsamen Stand

Kultur am Ende

commit	Macht eine Sache (Plugin oder `git add -p`)
commit	Kurze, aber präzise Beschreibung
main	Stabiler Branch, in den man nur Fertiges mergt
develop	Für aktuellen gemeinsamen Stand
user/topic	Mögliches Benennungsschema

Kultur am Ende

commit	Macht eine Sache (Plugin oder `git add -p`)
commit	Kurze, aber präzise Beschreibung
main	Stabiler Branch, in den man nur Fertiges mergt
develop	Für aktuellen gemeinsamen Stand
user/topic	Mögliches Benennungsschema
remerge	vermeiden, develop oder master in topic zu mergen