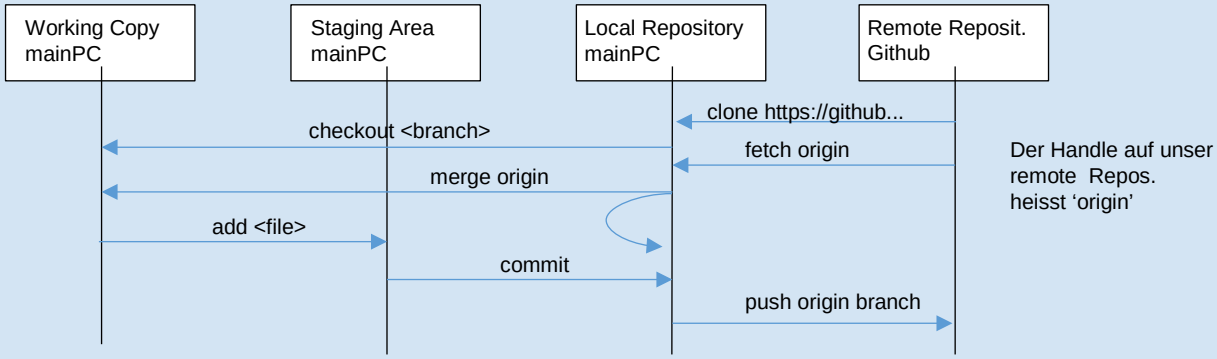
Anmerkung: Alle folgenden Darstellungen zeigen die wichtigsten Schritte mit git.

git hat ein vierstufiges Verfahren:



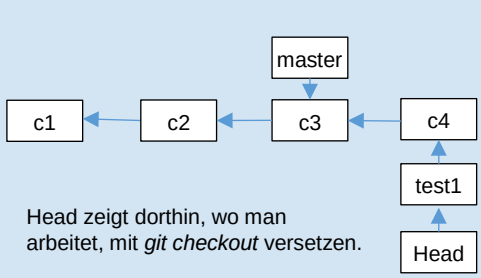
git add	Kann man auch für neue (untracked) Files machen.
git status	Zeigt an, welche Files verändert wurden und welche sich bereits in 'Staging Area' befinden.
git commit	Der Commit wird ins lokale Repository geschrieben, dazu gehört ein sinnvoller Kommentar.
git fetch	Dateien werden vom Server ins lokale Repo. geholt. Sind aber noch nicht im Dateisystem sichtbar.
git checkout <branch>	Checkt Branch aus. Dateien sind jetzt sichtbar.
git diff <file>	Zeigt die Unterschiede eines Files zum gleichen File im lokalen Repository.
	Sobald man ein lokales File abändert (egal ob bereits schon in Staging Area) kommt es in 'Working Copy'.
git push <remote> <branch>	Veröffentliche lokalen Branch auf remote Repository. Jetzt für alle zugänglich.
git rm <file>	Löscht ein File aus dem Repository. Man kann es auch einfach sonst löschen (z.B. im Dolphin), git merkt das meistens. Allerdings ist es dann erst weg, wenn es mit git commit ins Repository übertragen worden ist.
git log	Zeigt alle Commit- Kommentare, beginnend beim letzten, -4 zeigt die letzten 4 commits.
git ll	Zeigt die History in übersichtlicher Form.

git Setup:

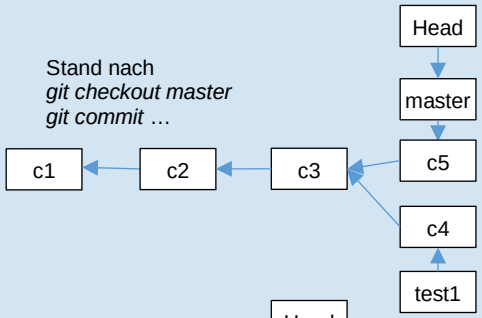
1. es gibt globale Einstellungen (für alle User) in /etc/gitconfig
2. User-spezifische in ~/.gitconfig
3. Projekt-spezifische in .git/config.

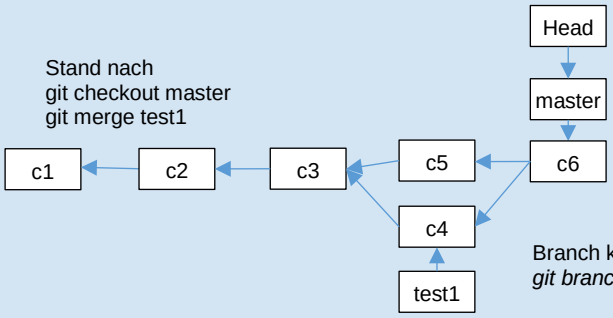
Die Einstellungen in den Levels überschreiben sich in dieser Reihenfolge. Mach die folgenden Einstellungen einmal
git config --global user.name «Urs Graf»
git config --global user.email «urs.graf@ntb.ch»
Es lohnt sich in ~/.gitconfig einige Aliases zu definieren. Kopiere diese aus https://wiki.ntb.ch/infoportal/software/git/gitconfig

Branching



Head zeigt dorthin, wo man arbeitet, mit *git checkout* versetzen.





'master' ist einfach der Name eines Branches, wenn es keine Branches hat, gibt es nur den 'master'. Der 'master' ist wie ein ganz normaler Branch.

Branch kann gelöscht werden
git branch –d test1

Was tun wenn:

Ich habe ein File abgeändert und möchte es wieder auf den Stand des Repos. zurückändern.	*git checkout <file>*
Alle lokalen Änderungen rückgängig machen.	*git checkout .*
Ich weiss nicht mehr, was ich alles geändert habe.	*git diff <file>*
Ich möchte etwas an der Software ändern.	1. Zu einem passenden Stand gehen: *git checkout master* 2. Branch erstellen: *git branch test1* 3. Auf den neuen Branch gehen: *git checkout test1* 4. Im neuen Branch arbeiten: *git commit* 5. Branch veröffentlichen, sodass andere diesen auch benutzen können: *git push origin test1* 6. Branch in master übernehmen: *git checkout master, git merge test1*
Ich möchte den 'master' verändern.	1. In separatem Branch alles testen 2. Nach master mergen 3. Testen 4. Wenn funktionsfähig, dann kann der master gepusht werden
Bestimmte Dateien wie build oder install Ordner und *.kdev Dateien sollen nie ins Repository.	In Datei .gitignore aufführen. Das hat den Vorteil, dass ein *git status* diese Dateien nicht immer als untracked auflistet.

Regeln:

a. Auf den Master werden nur Änderungen gepusht, die getestet worden sind. Die master-Version muss immer lauffähig sein.
b. Alle lokalen Dateien können von jedem gelöscht werden. Wenn Änderungen gesichert werden sollen (auch ungetestete Änderungen und 'work in progress') kann problemlos auf einen selbst erstellten Branch gepusht werden. Die Arbeit von anderen wird so nicht beeinflusst.