In Github Account erstellen.

Das Repository in Github müsste --bared --shared entsprechen

Github: "Start a project" zum Repository anlegen klicken

Von Github die Gitshell runterladen, die einen Teil von Linux auf dem

Windows-Rechner emuliert: https://desktop.github.com/

Shell-Kommandos:

pwd

Zeigt aktuellen Pfad an

alias

Gibt alle Aliase für Kommandos aus und zeigt die Schreibweise für die Zuweisung

an, um selbst Aliase anzulegen

ls

Das "dir" von Linux

ls -l

Lange Liste von ls

less Dateiname

Datei im Editor lesend anschauen, ohne aus Versehen schreiben zu können, mit ESC aus dem Editor wieder raus

cat Dateiname

Anzeige der Datei in der Konsole

chown

Besitzer für Datei ändern

chmod

Rechte selbst ändern

./

Aktuellen Pfad referenzieren (Absicherung gegen Angriffe)

echo $PATH

Globale Systempfade anzeigen

history 10

Zeigt die letzten 10 eingegebenen Kommandos an

ctrl + "a"

Zum Anfang der Zeile springen

ctrl + "e"

Zum Ende der Zeile springen

TAB

Vorschläge für ausgeschriebene Kommandos

Mittlere Maustaste

Einfügen (entspricht strg + V in Windows)

GIT

ERSTELLT GIT-PROJEKT IM AKTUELLEN ORDNER:

git init

GEMEINSAMES REPOSITORY ANLEGEN:

In den gewünschten Ordner, in dem ich das Repository haben möchte, auf dem Repsitory-Server gehen und eingeben:

git init --bare --shared

PROJEKTKOPIE (z.B. auf zusätzlichem Rechner) ANLEGEN:

git clone <source> <destination>

Wird <destination> nicht angegeben, wird ein Verzeichnis mit dem Verzeichnisname von <source> angelegt

und dort das gesamte Projekt, soweit es von Git erfasst ist, mit dem Ordner .git und den Dateien hinein

geclont.

PROJEKTKOPIE AUF AKTUELLEN STAND BRINGEN:

git pull

GIT PUSH SCHLÄGT FEHL WEGEN KONFLIKT:

Git pull eingeben, so dass Git

Merge-Versionen mit den Konflikten erzeugt, die wir jetzt bearbeiten müssen.

Nach der Bearbeitung neu commiten.

Git push lädt jetzt die Dateien ins Repository.

GEÄNDERTES PROJEKT MIT REMOTE-DATEN AUFSETZEN (Elegante Version):

Das Repository wird auf den Stand des Remotes gesetzt

git pull --rebase

1. Mit Git-Status ungestagte Dateien anzeigen lassen.

2. Ungestagte Dateien bearbeiten.

3. Bearbeitete Dateien mit git add adden.

4. Mit git rebase --continue das Rebase abschließen.

5. Aus dem Rebase-Branch wieder in den Master-Branch gehen, das Rebase mergen.

GEÄNDERTES PROJEKT MIT REMOTE-DATEIEN ÜBERSCHREIBEN (Harte Version, unerwünscht):

1. Schritt: Aktuellen Stand des Remotes holen:

git fetch --all

2. Schritt: Eigene Änderungen im Master mit den Remote-Daten überschreiben:

git reset --hard <remote short name>/master (Eigene Änderungen im Master überschreiben)

REMOTE IN MEIN PROJEKT CLONEN:

<destination> muss ein leerer Ordner sein:

git clone <source> <destination>

ERSTES PUSH AUF REMOTE REPOSITORY:

Ohne Merken: Ein Master branch auf dem Repository anlegen:

git push <remote shortname> master

Der Master soll nicht nur angelegt, sondern auch gemerkt werden:

git push -u <remote shortname> master

PUSH NACH ANLEGEN DES GEMERKTEN MASTER BRANCHES:

git push <remote shortname>

PUSH BEI NICHT GEMERKTEM MASTER BRANCH:

git push -u <remote shortname> master

MEIN REPOSITORY AN ANDERES REMOTE REPOSITORY ANSCHLIESSEN:

git remote add <remote shortname> <url>

MEIN REPOSITORY VOM REMOTE REPOSITORY TRENNEN:

git remote rm <remote shortname>

REMOTE REPOSITORIES ANZEIGEN:

git remote -v

REMOTE REPOSITORY DURCHSTÖBERN:

git remote show <remote shortname>

DATEN VON REPOSITORY LADEN:

git pull <remote shortname>

git fetch <remote shortname>

Anschließend in den Branch gehen, in dem ich die Daten mergen möchte (in der Regel master). anschließend die Daten mit meinen Daten mergen:

git merge <shortname>/master

Jetzt müsste ich mit "git log" die neueste History haben, die dem Remote-Server entspricht.

FÜGT DATEI ZUM STAGING-BEREICH HINZU:

git add test\*

FÜGT ALLE AKTUELL GEÄNDERTEN (NUR DIESE!) DATEIEN ZUM STAGING-BEREICH:

git add

BEARBEITUNGSSTAND COMMITTEN:

git commit -m "Versionsbeschreibung"

STATUS ANSEHEN:

git status

HISTORIE ANSEHEN:

git log

git log --stat

git log --since=2.weeks

git log 2008-01-15

git log 2 years 1 day 3 minutes ago

git log author

git log committer

Branch pointers ansehen:

git log --oneline --decorate

Historie im Remote ansehen:

git log origin/master

UNTERSCHIEDE ANSEHEN:

git diff

git diff --staged / --cached (beide Optionen machen das Gleiche)

git diff HEAD

Änderungen SEIT letztem Commit rückgängig machen, aber intern nicht löschen:

git stash

Rückgängig machen des Stashs, Wiederherstellen:

git stash pop

Löschen der Änderungen seit letztem Commit:

git stash drop

DATEIEN IGNORIEREN:

In Datei .gitignore die Dateien oder regul�ren Ausdr�cke zu Dateien auflisten

REPOSITORY AUF NAS-SERVER SCHAFFEN:

net use Z: \\OSC-TEAM\OSC-Drive

(Z: Selbstgew�hlter Laufwerksbuchstabe)

DATEIENSTAND ZURÜCKSETZEN:

Dateien auf Zustand von vorherigem Commit setzen:

git checkout 18d9 <ggf. einzelne Dateinamen>

(18d9: Teil der Checksum)

Oder:

git checkout :/"Meine erste Version"

Oder:

git checkout master~5

(der 5.-letzte Commit)

Oder:

git checkout src/AppBundle/.../IndexController.php

Zurückgehen zum aktuellen Commit:

git checkout master

Alle neueren Spielstände als den Geladenen im Repository löschen:

git reset --hard

AKTUELLSTEN COMMIT LOESCHEN:

- Mit "git checkout master" zum aktuellen Commit gehen.

- "git reset --hard HEAD~1" eingeben, so geht man einen Commit zurück und es wird der Commit, auf dem man gesessen hat, gel�scht.

Glaub, auch löschen des Commits:

git revert

Wird nicht gelöscht und die Dateien bearbeitet, wird ein neuer Branch erstellt!!!

EINZELNEN GIT COMMIT LÖSCHEN:

git commit -a

git revert 1b6d

ÄNDERUNGSPROTOKOLL ERSTELLEN:

git log > Changelog

HERAUSFINDEN, WAS ICH SEIT LETZTEM COMMIT GETAN HABE:

git diff

Oder seit gestern:

git diff "@{gestern}"

Oder zwischen irgendeiner Version und der vorletzten:

git diff 1b6d "master~2"

Diff vom anderen Branch, z.B. im Remote, ausgeben:

git diff origin/master

Versuche auch:

git whatchanged --since="2 weeks ago"

GIT-BEFEHLE:

GIT KONFIGURIEREN:

git config --global user.name "John Doe"

git config --global user.email johndoe@example.com

git config --global core.editor "'C:/Program Files (x86)/Notepad++/notepad++.exe' -multiInst -nosession"

KONFIGURATION ANSEHEN:

git config --list

REPOSITORY ANLEGEN:

In den obersten Produktionsordner (z.B. htdocs) gehen und eingeben:

git init

DATEIEN ADDEN:

git add <Dateiname/n>, z.B.:

git add test\*

COMMITTEN:

git commit

git commit -m "Versionsbeschreibung"

ADDEN UND COMMITEN IN EINEM BEFEHL: G�LTIG F�R DATEI�NDERUNGEN VON DATEIEN, DIE BEREITS INS STAGING GEADDED WURDEN:

git commit -a -m "Versionskommentar"

AKTUELLSTEN COMMIT LOESCHEN:

- Mit "git checkout master" zum aktuellen Commit gehen.

- "git reset --hard HEAD~1" eingeben, so geht man einen Commit zurück und es wird der Commit, auf dem man gesessen hat, gel�scht.

ANDERE VERSION EINSCHALTEN (Dateien auf ihren Stand bringen):

git checkout <Beginn der Checksum oder Branchname>

GIT BRANCHING:

Mit neuem Branch z.B. Hotfix erstellen:

// Kurzform von "git branch hotfix" und "git checkout hotfix" eingeben, um neuen Branch herzustellen

und reinzugehen:

git checkout -b hotfix

//Dateien bearbeiten und commiten:

git commit -a -m 'added a new Datetime-Object'

// Zum Master zurück gehen:

git checkout master

// Hotfix einbinden:

git merge hotfix

Git erstellt automatisch einen Commit, wenn kein Merge-Konflikt vorliegt!!!

Liegt hingegen ein Merge-Konflikt vor, muss man in die Datei gehen, die Stellen mit <<<<<< HEAD ====== >>>>>

bearbeiten und dann den Commit selbst machen.

Anschließend den Branch löschen:

git branch -d hotfix

// Wenn noch nicht gemergte Daten im zu löschenden Branch sind, kann man ihn

mit Datenverlust ZWANGSweise löschen:

git branch -D hotfix

// Branch im Remote speichern:

git push origin <branchname>

// Branch im Remote löschen:

git push origin --delete <branchname>

// Branch umbenennen:

git branch -m oldname newname

// Current Branch umbenennen:

git branch -m newname

// Alle letzten Commits der Branches anzeigen lassen:

git branch -v

// Alle Branches anzeigen, die in den aktuellen Branch gemergt sind:

git branch --merged

// Alle Branches anzeigen, die noch nicht in den aktuellen Branch gemergt wurden:

git branch --no-merged

// Tracking branch zu einem Remote-Branch erstellen:

git checkout -b trackingbranch origin/master

Wenn ich im Master jetzt "git push" oder "git pull" eingebe, weiß Git,

wohin bzw. von wo ich pullen / pushen möchte.

REMOTEDATEN MIT MEINEN DATEN MERGEN:

git merge <remote shortname>/master

Ich kann aber mich auch an ein Repository "dranhängen" und die Daten mit "pull" bekommen,

wodurch meine Daten automatisch aktualisiert werden.

DATEI LÖSCHEN:

git rm file1.txt

git commit rm --remove file1.txt�

Datei nur aus dem Repository l�schen:

git rm --cached file1.txt