Introductie in versiebeheer met Git

Versiebeheer

Versiebeheer is een manier om veranderingen in een bestand of groep van bestanden over de tijd bij te houden, zodat je later specifieke versies kunt terugvinden en met elkaar kunt vergelijken.

Bij ontwikkeling van software maar ook bij bv. documenten is dit erg handig: eerdere versies inzien, backup van alle aanpassingen, inzicht in het proces enzovoort. Daarnaast maakt een online versiebeheersysteem het samenwerken aan eenzelfde project mogelijk zonder dat je elkaar daarbij in de weg zit[1].

Er zijn verschillende varianten op versiebeheer: lokale versiebeheersystemen, gecentraliseerde versiebeheersystemen en gedistribueerde versiebeheersystemen.

Git

Git is zodanig flexibel van opzet dat je het zowel lokaal, centraal als decentraal kunt gebruiken maar in feite is het een gedistribueerd versiebeheersysteem.
Github (https://github.com) is de bekendste web-based ‘git repository hosting service’.

Een account op github.com is voor iedereen zichtbaar, dus het is verstandig om een ‘beschaafde’ accountnaam te kiezen. Je account gaat ws. vele jaren mee en anders kun je in de toekomst mogelijke opdrachtgevers afstoten. Op github kun je zowel public als private repositories (losse opslagplaatsen) aanmaken. Public repositories worden veel gebruikt binnen de open-source community. Om private repositories te kunnen aanmaken moet je een github abonnement hebben óf jezelf registreren als student (een ‘Github Student Developer Pack’ biedt daarnaast nog meer voordelen).

[1] Nou ja.. in theorie dan toch...

Werken met een lokale repository

- Zorg dat je git op je eigen computer geïnstalleerd hebt, meer hierover vind je op: https://git-scm.com/book/nl/v1/Aan-de-slag-Git-installeren

- configureer lokaal je identiteit:
  `git config --global user.name "<firstname> <lastname>"`
  `git config --global user.email <yourEmailAddress>`

- Controleer of alles goed staat:
  `git config --global --list`

Lees altijd goed de feedback die git aan je teruggeeft op de command line. In deze feedback staat duidelijk aangegeven wanneer een door jou ingevoerde opdracht niet gelukt is en ook waar dit aan lag. Wanneer je zonder het lezen van deze feedback opdrachten aan GIT geeft, kun je vastlopen, omdat een eerdere stap niet goed uitgevoerd is zonder dat je dit doorhad.

Ga in de terminal naar een locatie waar je een nieuwe project-repository wilt aanmaken. Maak hierin een folder aan en initialiseer een git repository:

- Navigeer naar de locatie waarin je een nieuwe folder voor je project en repository wilt maken, bijvoorbeeld:
  `cd ~/Documents`

- Maak een nieuwe folder aan:
  `mkdir CSD2`

- Navigeer naar deze folder:
  `cd CSD2`

- Maak een nieuwe lege git repository binnen deze folder
  `git init`

- Aan de map CSD2 is nu een **.git** directory toegevoegd. Deze bevat alles wat Git nodig heeft voor versiebeheer van alles binnen de folder CSD2, inclusief alle subfolders.

- Bestanden en folders met een naam die met een . beginnen zie je niet zonder meer met ‘ls’, maar met de switch -a zie je ze wel:
  `ls -a`

Lokale repository in gebruik nemen
Je hebt nu lokaal een lege git repository gemaakt. We gaan hieraan bestanden toevoegen.

Maak nu een nieuwe text-file **hello.txt** met je favoriete (plain-) text editor (voor de beste hippocampus-training gebruik je Vim of Emacs) en voeg deze toe aan de lokale repository.

- Je kunt git een overzicht laten tonen van de files in deze repository. Als het goed is staat de nieuwe **hello.txt** nu onder “Untracked files”, wat betekent dat deze file nog niet door git beheerd wordt.
  `git status`

- Voeg **hello.txt** toe aan de zogenoemde ‘staging area’, hier worden alle files gezet die je wilt gaan toevoegen aan of updaten in de repository:
  `git add hello.txt`

- Wanneer je git opnieuw een overzicht laat tonen van de files in deze repository, staat nu de **hello.txt** file onder “Changes to be committed”, wat betekent dat hij zometeen met een ‘git commit’ aan het archief toegevoegd wordt.
  `git status`

- We kunnen nu alles wat bij ‘to be commited’ staat in de repository verwerken. Schrijf bij elke commit een stukje tekst waarin je aangeeft waarom je die commit doet, zoals bijvoorbeeld wat je hebt toegevoegd of veranderd en waarom. Deze *commit message* kun je meteen meegeven na -m (zie hieronder) of als je tekst wat langer is, dan laat je -m weg en wordt een editor geopend. Let op: schrijf in het Engels (net zoals je code zelf) én gebruik duidelijke ‘commit messages’! Aan iets als “added some new code” kun je niet aflezen wat de aanpassingen zijn geweest.
  Commit nu de files (in dit geval de **hello.txt** file) als volgt:
  `git commit -m "message"`

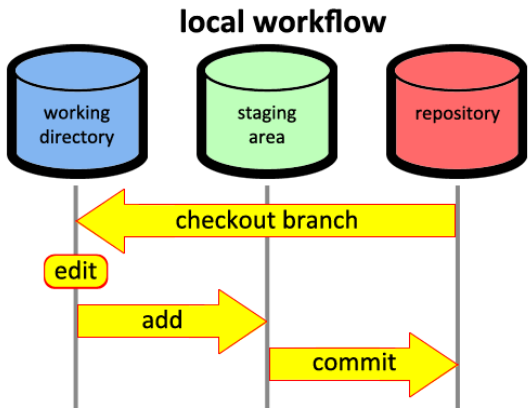
  `waarbij je meteen je commit message meegeeft.`

  Quick tutorial in Vim: https://www.dinkum.nl/software/editing/vim_basics.html

- Voer nu je eerste *commit message* in, bijvoorbeeld:
  `initial commit : added hello.txt file`

- Met [ESC] ga je uit de VIM insert mode terug naar command mode. Typ vervolgens :wq (dubbele punt, w, q), om op te slaan (w) en af te sluiten (q).

`[ESC]`
`[:wq]`



Tips voor gebruik van git (en dergelijke systemen)

- zet geen ‘rommel’ in productie-branches: geen code die nog niet compileert of waarvan je weet dat het nog bugs bevat, geen code die bij mede-ontwikkelaars tot problemen kan leiden en geen grote stukken ‘uitgecommentaarde code’. Dat laatste kun je in bv. een development branch wel doen.
- zet geen executables, .o files (met C++) en andere binaries in git, tenzij deze echt nodig zijn. In het algemeen kun je stellen dat bestanden die eenvoudig uit de overige bronnen zijn af te leiden niet in git hoeven te staan.
- schrijf bij elke commit een **zinvol** stukje commentaar waarin staat waarom je deze commit doet
- als je libraries gebruikt die niet van jou zijn, voeg deze dan niet integraal aan je eigen git toe. Je kunt ze beter als modules buiten jouw eigen code laten staan. Wanneer je zelf aanpassingen aan zo’n library gaat doen kun je wel overwegen om het onder eigen beheer te plaatsen. Let dan wel op hoe e.e.a. geregeld is m.b.t. licenties, met name als je jouw project publiekelijk beschikbaar maakt of wilt verkopen.

Online repository (voorbeeld aan de hand van github)

Je kunt elke git repository als remote aanwijzen, maar in dit voorbeeld gaan we uit van een repo op Github.

Maak een account aan op github.com.

Maak onder je eigen account een nieuwe repository. Kies voor deze eerste repository:
Repository name: CSD2
Description: (bijvoorbeeld) Uitwerkingen Creative Systems Design

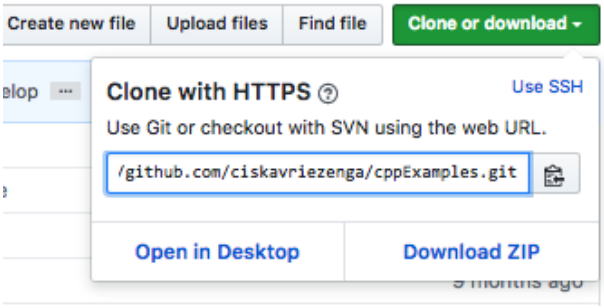
Deze repository kun je 'Public' maken. Dat kun je later nog aanpassen.

Laat github voor deze repository geen README maken en ook geen .gitignore !

Het is gebruikelijk dat een project een README of README.md file bevat met informatie over het project. Een .md file is een textfile in markdown, "Markdown is a way to style text on the web." https://guides.github.com/features/mastering-markdown/. Wanneer een git repository een README.md bevat, wordt de inhoud hiervan door github getoond op de online pagina van deze repository. Hierop wordt vaak een intro én uitleg gegeven over de inhoud. Kijk maar eens op de me-bash-scripts pagina van Marc https://github.com/marcdinkum/me-bash-scripts.

We laten README.md echter niet door github maken maar we maken hem in onze lokale repository.

Lokaal aan online koppelen
Lokaal is de remote repository (de online repository in dit geval) nog niet bekend. Om de lokale repository aan de online repository te koppelen, heeft git lokaal de locatie van deze remote repository nodig.



- De URL van je github repository kun je vinden op je github site. Wanneer je op de knop [Clone or download] klikt, wordt deze link weergegeven, zie de afbeelding hiernaast. Kies bijvoorkeur voor SSH.
  Koppel je github repository nu aan je lokale repository (gebruik als link de link naar je eigen repository):
  `git remote add origin` git@github.com:<username>/<repository>.git

- Controleer of het toevoegen van een remote is gelukt. Laat git je opgeslagen remotes tonen:

`git remote -v`
Zie je nu “origin” staan en daarachter de door jou opgegeven url, zoals hieronder?
`origin` git@github.com:<username>/<repository>.git

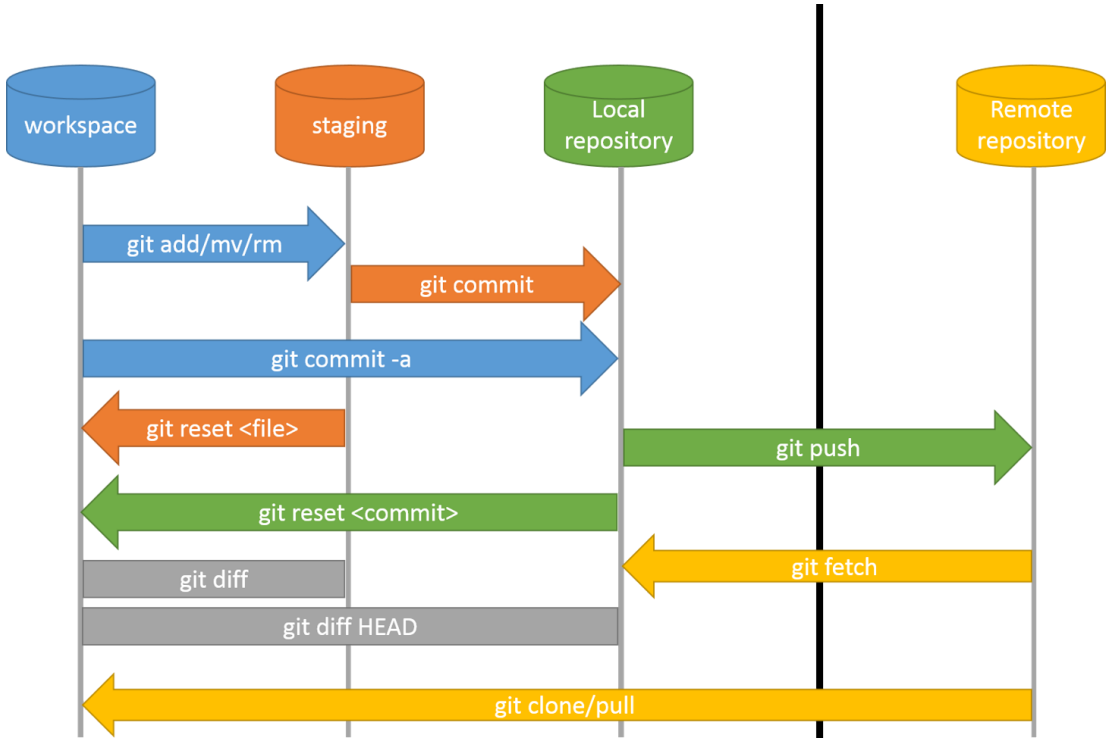
Remote repository synchroniseren
Om de README.md file toe te voegen aan je lokale repository maak je met (alweer) je favoriete plain-text editor een bestand README.md en schrijf hier iets in, bijvoorbeeld “My first repo”. Voeg de file toe aan je lokale repository op de manier die we hiervoor met hello.txt gebruikt hebben.

- Laat git je lokale repository synchroniseren met de remote:
  `git push -u origin main`

- In je remote repository is nu de README.md file aanwezig die je lokaal aangemaakt hebt, kijk maar in github (misschien even refreshen).

Vanaf nu kun je beide kanten op pushen en pullen.

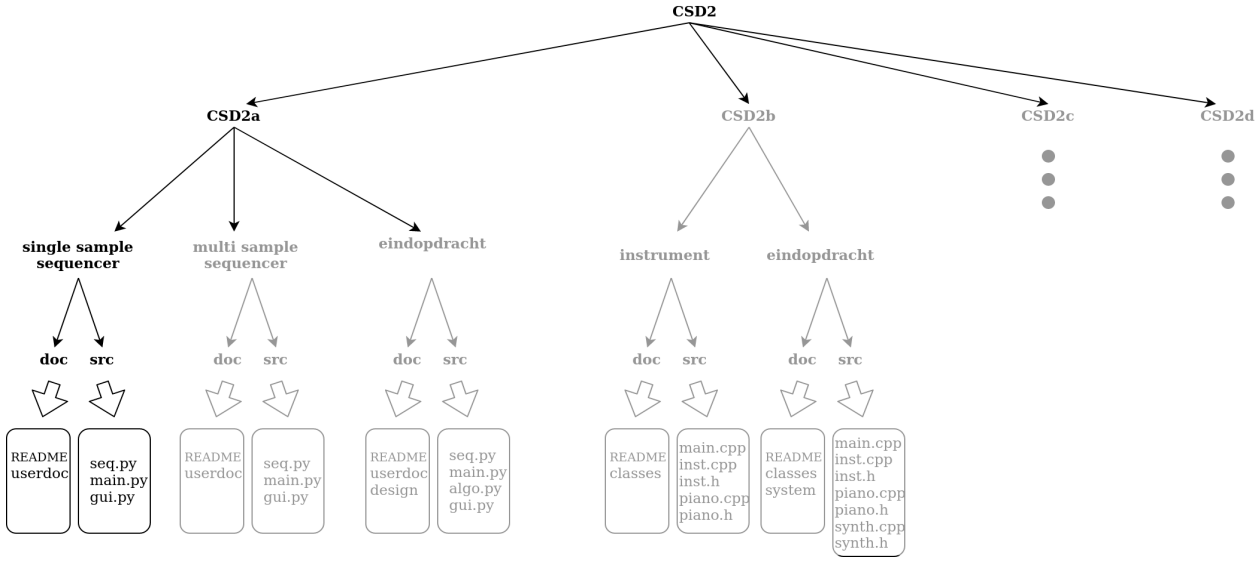
Hieronder staat een systematisch overzicht van oa. bovenstaande (add, commit, push).



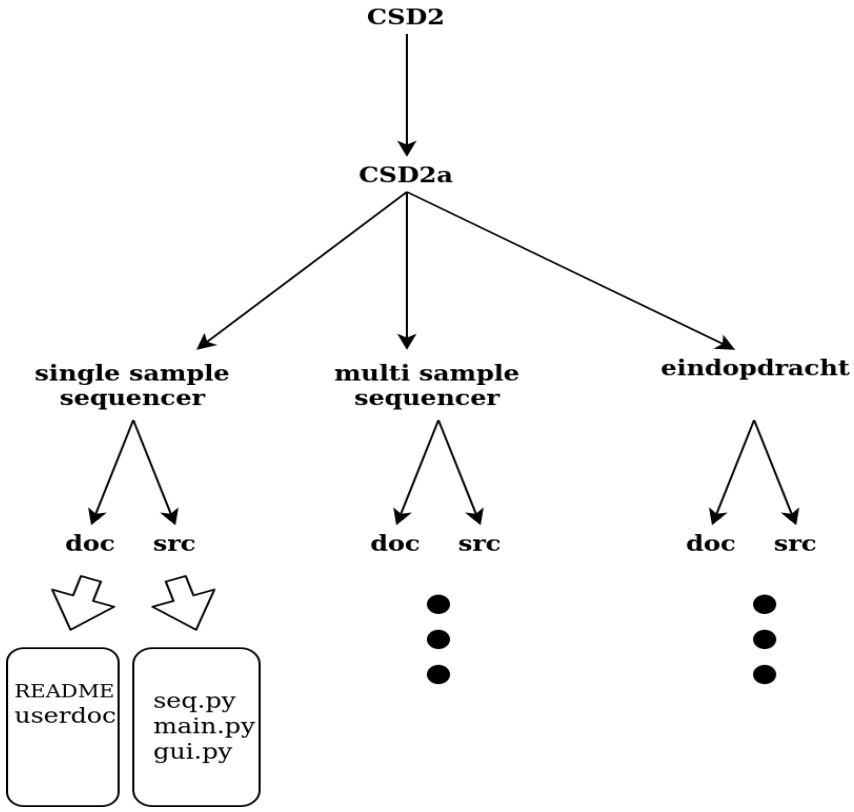
Tot slot. Lokaal kent git je username en password nog niet. Deze kun je op verschillende manieren opslaan, zie hiervoor:
- https://help.github.com/articles/caching-your-github-password-in-git/#platform-mac
- https://help.github.com/articles/connecting-to-github-with-ssh/

Folder-structuur voor huiswerk

Inleveren van huiswerk voor CSD2 gaat via Git. Hiervoor laat je de docenten weten waar ze jouw online git repository CSD2 kunnen vinden. Je werk wordt alleen beoordeeld als het volgens de onderstaande structuur wordt aangeleverd.



Geen paniek: dit bouw je gaandeweg jaar 2 op. Om te beginnen zie je hieronder een deel van CSD2a uitgelicht.



VRAGEN
Probeer het uit, zoek het online op, beredeneer, ...

- Wat doet `git diff`?
- Wat doet `git fetch`?
- Wat doet `git pull`?
- Wat doet `git log`?
-
- Wat doet `git push`?
- GIT **justification**, bedenk minimaal 3 redenen waarom het gebruiken van Git zinvol is.