Git Versie Controle Systeem

Git wordt in het HMT gebruikt om met meer mensen aan 1 project te werken. Het zorgt ervoor dat wijzigingen aan bestanden ongedaan gemaakt kunnen worden en het voorkomt conflicten tussen bestanden.

Git werkt met zogenaamde *repositories* , soms verkort tot *repo* . Een *repo* is een map op je computer waarvan git de inhoud controleert en in de gaten houd. Bij HMT werken we met 1 *repo* (op moment van schrijven is dat *leiden2017*). Iedereen kloont de de *repo* , werkt aan de bestanden erin, vertelt **Git** wat hij/zij heeft gedaan en *pusht* vervolgens deze wijzigingen weer naar de online versie van de *repo* zodat iedereen deze wijzigingen weer kan downloaden en ook heeft. Een *repo* is dus een map die online gesynchroniseerd wordt, denk aan **Dropbox** maar dan met veel meer controle.

Om met Git te werken zijn er de volgende belangrijke commando's met daaronder voor elk commando een korte uitleg:

1. `git clone`
2. `git status`
3. `git pull`
4. `git add`
5. `git commit`
6. `git push`

1. git clone

Om een *repo* te clonen gebruik je dit commando. Met dit commando download je voor de eerste keer een versie van de online *repo* op je computer. Als je je lokale kopie compleet vernield heb kan je altijd nog de *repo* verwijderen en de *repo* opnieuw clonen. NB: dan ben je wel al je werk dat je nog niet gecommit heb kwijt ben. (Meer over committen volgt nog).

Je kloont een *repo* als volgt:

```
git clone REPO_URL
```

In plaats van "REPO_URL" type je de URL van de repo die je kan vinden op de GitHub pagina van de repo die je wilt clonen.

2. git status

Dit commando kan je altijd invoeren als je wilt weten hoe de status van jouw lokale kopie van de repository is tegenover de laatste versie die online op **Github** gehost staat. Dus voer dit in gevolgd door `enter` of `return` om:

```
git status
```

De output die volgt vertelt je wat er veranderd is tegenover de meest recente versie. Als je dus gewerkt heb aan een bestand komt dit ook als output van dit commando te staan.

Stel je voor dat er wijzigingen zijn die je heb aangebracht aan `209v.xml`. Dan komt dat dus als output. Inclusief de volledige mappenstructuur relatief van waar je nu bent.

3. git pull

Door dit commando uit te voeren haal je de laatste wijzigingen van de online versie naar je lokale versie. Kortom, je download de updates van de online *repo* naar je lokale *repo*. Dit doe je door het volgende commando:

```
git pull
```

Hierna volgen enkele regels output die beschrijven wat er veranderd is door je download. Als er niets veranderd is dan meld git dit door te zeggen dat je `already up to date` bent.

4. git add

Nadat je allerlei ongetwijfeld uiterst belangrijk werk hebt verricht aan het project moeten je wijzigingen natuurlijk wel gedeeld met de rest van wereld en vooral de rest van het project. Dit doe je in een paar stappen. De eerste stap hiervan is deze.

```
git add BESTANDSNAAM
```

vervang hier BESTANDSNAAM door de naam en het pad naar het bestand dat je gewijzigd heb. Stel, je hebt in de **editions** maps **test.xml** gewijzigd, dan doe je dit:

```
git add editions/test.xml
```

Om de precieze locatie van de bestanden die je hebt gewijzigd uit te vinden kan je altijd `git status` uitvoeren, zodat je kan zien waar de wijzigingen zich bevinden.

Nadat je dit commando heb uitgevoerd ben je klaar om je te committen.

5. git commit

Git is geen programma voor mensen met 'commitment issues'. Elke wijziging die je hebt toegevoegd (met `git add`) moet je ook commiten met:

```
git commit -m "Type hier wat je heb gedaan"
```

Let op de "-m". Die is noodzakelijk. Tussen de aanhalingstekens bevindt zich de beschrijving die jij wilt geven om de wijzigingen die je hebt gemaakt aan de bestanden die je hebt toegevoegd. **Maak dit niet te lang, maar wel beschrijvend** .

Nu ben je klaar om je wijzigingen de grote boze buitenwereld in te sturen.

6. git push

Dit is het commando waarme je jouw commits de wereld instuurt. Nadat je dit commando hebt uitgevoerd is de online *repo* upgedate en zullen mensen die hierna `git pull` doen de nieuwe versie die jij hebt gewijzigd downloaden. Push je wijzingen naar de repo door dit uit te voeren:

```
git push
```

Let op:

1. Als je voor het eerst pusht met Git kan het zijn dat je je gebuikersnaam en wachtwoord van Github moet invoeren, dat is normaal en zal na de eerste keer niet meer hoeven.
2. Het is vaak wijs om voordat je `git push` uitvoert eerst `git pull` uit te voeren. Zo heb je de laatste versie op je lokale *repo* en daarna kan je jouw wijzigingen alsnog pushen.

Mogelijke problemen bij het pushen:

1. Als output krijg je dat je eerst moet pullen (lees de output van een push command goed), doe dan eerst `git pull` en voer nog een keer het `git push` commando uit.
2. Opeens opent er een teksteditor (dit zal vaak Atom zijn, maar dat is afhankelijk van je computer). Deze kan je gewoon sluiten.

Conclusie

Git blijft voor velen een *acquired taste* . Het is even wennen om ermee om te gaan, maar zodra je eraan gewend ben bied het veel mogelijkheden.

Je kan om even te oefenen, deze tutorial serie van **Github** proberen:
https://try.github.io/levels/1/challenges/1
(https://try.github.io/levels/1/challenges/1)