DevOps

Git advanced

AP HOGESCHOOL ANTWERPEN

AP.BE

Git advanced

Best practices

AP HOGESCHOOL ANTWERPEN

AP.BE

Atomic commits

- 1 fix/feature per commit!
- Edit zo weinig mogelijk bestanden per commit
- Splits front en backend werk in verschillende commits
- Alleen werk dat bij elkaar hoort samen comitten

Duidelijke commit messages

- Gebruik 1 taal!
- Present tense (tegenwoordige tijd)
  bvb: ~~Fixed~~ Fix bug in interface
- Regel 1 is korte samenvatting
- Uitgebreide uitleg mogelijk in verdere regels

Stel git correct in

- `git config --global user.name "Voornaam Achternaam"`
- `git config --global user.email "email@example.org"`

Werk nooit rechtstreeks op master/main

- Werk met feature branches
  `git checkout -b feature_gui`
- Gebruik een duidelijke naming scheme
  bvb: `<auteur>_<type werk>_<naam werk>`
- Niet lokaal mergen met de master, wel pushen en pull request.

Gebruik git power tools

- Gebruik git rebase om je eigen branch op te kuisen
- Gebruik git reflog om vermiste commits op te sporen
- Gebruik git cherry pick om commits te redden
- Gebruik git stash voor tijdelijke opslag
- Gebruik git blame en git bisect om fouten op te sporen

Gebruik een .gitignore bestand op maat

- Commit geen locale config bestanden
- Commit geen gegenereerde bestanden
- Vermijd binaire bestanden
- Commit geen dependencies gebruik:
  - Git submodules
  - Dependency tool zoals npm, pip,gradle,...

Git advanced

Remote intermezzo

AP HOGESCHOOL ANTWERPEN

AP.BE

Simpeler remote werken

- Stel je ssh key in op github/gitlab/bitbucket/...
- Gebruik ssh voor al je verbindingen!
- Typ nooit nog een wachtwoord te veel in

Git advanced

Branch workflow

AP HOGESCHOOL ANTWERPEN

AP.BE

Bescherm de master

- Maak de master branch write protected
- Werk altijd op een lokale feature branch
- Push een branch pas als je er tevreden van bent
  - Er is geen weg terug!
- Werk met pull requests
- DEMO

Git stash

- Sla je wijzigingen tijdelijk op met git stash push
- Haal je wijzigingen terug op met git stash pop
- LET OP: wijzigingen worden naar de stash verplaatst en verdwijnen dus uit de working directory

Merge vs Rebase

- Merge
  - Maak een extra commit waarin 2 branches samenkomen
  - Geschiedenis blijft ongewijzigd
  - Mogelijks een cluwe van commits
- Rebase
  - Schuif de commits op je feature branch naar achter de laatste commit op de master
  - Geschiedenis wordt herschreven
  - Mooie lijn van commits
- DEMO

Merge conflicts

- Verschillende wijzigingen aan hetzelfde (stuk) bestand
- Handmatig te resolven
- Testen noodzakelijk
- Kennis van codebase nodig
- DEMO

Git advanced

Proper werken

AP HOGESCHOOL ANTWERPEN

AP.BE

Lokaal en remote

- Houd alles dat nog niet afgewerkt is lokaal
- Herschrijf geen publieke geschiedenis
- Herschrijf je eigen lokale geschiedenis alvorens je wijzigingen publiek te maken

Je eigen geschiedenis opschonen

- Gebruik: git rebase -i <base>
  - pick
    Houd een commit bij.
  - reword
    Houd commit bij maar herschrijf commit message
  - edit
    Maak het mogelijk nog iets toe te voegen aan de commit
  - squash
    Voeg commit toe aan vorige commit. Behoud message
  - fixup
    Idem aan squash maar zonder behoud message
  - drop
    Neem commit niet meer mee

Oepsies

- Lokaal
  - Gebruik git reset of git rebase -i om een commit aan te passen of te doen verdwijnen
  - Kijk je log na om te zien of je probleem verholpen is
- Publiek (eeuwige entry in the wall of shame)
  - Gebruik git revert om de commit ongedaan te maken met een inverse commit
  - Bij zeer grove fout kan een force push overwogen worden

BIG oepsies

- Don't panic! Git's got your back
- Alles dat gecommit is geweest is er nog!
- Triggerhappy met git reset:
  - Gebruik git reflog om je verloren commit op te sporen
  - Gebruik git cherry-pick om je commit terug te krijgen
- Triggerhappy met rebase, chery-pick,…
  - Idem aan hierboven.
  - Vergeet niet op te kuisen met rebase
- Alles naar de knoppen geholpen
  - Plaats de backup terug die je gemaakt hebt ;-)

Git advanced

Extra

AP HOGESCHOOL ANTWERPEN

AP.BE

Onbehandelde git functionaliteit

- Submodules
  De mogelijkheid om een git repo onder de jouwe toe te voegen. Jouw git repo houd enkel bij welke commit je nodig hebt van de andere repo
- Git LFS
  Maakt het mogelijk om grote bestanden op te slagen in je repo. Je doet hier dan geen versiebeheer op en ze moeten niet elke push en pull mee over en weer.
- Hooks
  Maakt het mogelijk scripts te runnen bij het uitvoeren van bepaalde git commando's

Git advanced

De opdracht

AP HOGESCHOOL ANTWERPEN

AP.BE

Opdracht git advanced lokaal deel

1. Clone de voorbeeld repo van git@github.com:DevOpsAP-22-23/OpdrachtGitAdvanced.git
2. Maak een nieuwe branch genaamd <voornaam>_merge_print-name die afsplitst van wouter_feature_print-name
3. Merge de master branch naar deze branch
4. Maak nog een nieuwe branch aan genaamd <voornaam>_rebase_print-name die afsplitst van wouter_feature_print-name
5. Rebase deze branch met als base de master branch. Doe in dit proces ook de lelijke fix typo commit uit de history verdwijnen.

Opdracht git advanced remote deel

1. Maak een nieuwe publieke repo aan op github.
2. Koppel repo als remote repo aan de voorbeeldrepo.
3. Push al je lokale branches naar github.

Opdracht git advanced inleveren

1. Maak een zip bestand van de voorbeeldrepo die je bewerkt hebt
2. Upload dit zip bestand op digitap.