Linux for Data Scientists 24-25 - take-home scriptingopdracht

`git-helper.sh` is een script dat een Linux gebruiker helpt bij het beheren van lokale Git-repositories.

Het bijgevoegde bestand `git-helper-start.sh` bevat startcode met een basisstructuur met een aantal voorgedefinieerde functies waar je mee kan beginnen. **Hernoem het naar `git-helper.sh`.**

Hulp opvragen

Als je `help` opgeeft, drukt het script een hulpboodschap af en sluit meteen af met exit-status 0. Gebruik hiervoor een Here Document!

```
Usage: ./git-helper.sh COMMAND [ARGUMENTS]...

check
            check basic git user configuration
check DIR
            check basic git user configuration and check DIR for
            deviations of standard git practices
log
            display a brief overview of the git log of the PWD
stats
            display some brief stats about the PWD repository
undo
            undo last commit from git working tree while preserving
            local changes.
sync
            sync local branch with remote
```

Algemene requirements

- Zorg dat je naam en emailadres vermeld zijn op de voorziene plaats in de commentaar bovenaan het script!
- Gebruik shell-opties om de robuustheid van het script te verhogen (bv. behandelen van onbestaande variabelen).
- Gebruik ShellCheck om fouten te voorkomen!
- Gebruik `stdout` exclusief voor het afdrukken van informatie uit de Git-helper: oplijstingen, gevraagde instellingen, eventuele wijzigingen en commits ... M.a.w. alles wat onder het *normale* gedrag van dit script valt. Foutboodschappen, waarschuwingen enz. worden afgedrukt op `stderr`.
- Vermijd "hard-coded" waarden in de code, gebruik zoveel mogelijk variabelen!
- Het script wordt altijd opgeroepen met als eerste argument een commando. Als dit niet het geval is, wordt verondersteld dat de gebruiker `help` bedoelde.

Minimale requirements voor inhoudelijke beoordeling

De hieronder opgesomde criteria zijn noodzakelijk om een bestand als script te kunnen uitvoeren. Inzendingen die hier niet aan voldoen kunnen we dan ook niet inhoudelijk beoordelen en krijgen meteen 0:

- Het resultaat van zowel `bash -n git-helper.sh` als `shellcheck --severity=error git-helper.sh` moet succesvol zijn (dus zonder fouten).
- Het script mag geen DOS-regeleindes (CRLF) hebben, anders kan Bash het niet interpreteren
- Het script moet een geldige "shebang" hebben op de eerste regel
- Als we het script uitvoeren met optie `help`, dan moet dit lukken (we krijgen dus de Usage: boodschap te zien en de exit-status is 0)

Pas als al deze criteria voldaan zijn, kunnen we ook verder inhoudelijk beoordelen!

We verwachten verder ook dat je gebruik maakt van het aangeleverde sjabloon en de daarin gedefinieerde functies implementeert. Je mag uiteraard extra functies toevoegen als je dat nuttig vindt.

Functionaliteiten van het script

Algemene controle

Met `check` kan de gebruiker een algemene controle uitvoeren:

- Controleer of de **basisinstellingen** `user.name`, `user.email` en `push.default` zijn ingesteld. Indien niet, vraag de gebruiker om deze in te stellen. Het instellen van deze variabelen hoef je niet af te handelen binnen het script.

Voorbeeld:

```
$ ./git-helper.sh check
Git username not set. Set it using
    'git config --global user.name "Your Name"'
```

Extra controles

Met de optie `check <directory>` kan de gebruiker ook extra controles uitvoeren op een gegeven git-repository. Naast de algemene controle worden ook volgende controles uitgevoerd:

- Controleer of de gegeven directory wel deel uitmaakt van een git-repository. Toon een waarschuwing indien dit niet het geval is.
- Controleer dat de root-folder van de repository volgende bestanden bevat:
  - README.md, .gitignore en .gitattributes (hoofdlettergevoelig)
  - Toon een gepaste waarschuwing bij het niet aanwezig zijn van bovenstaande bestanden.
- Controleer of de gegeven repository een remote heeft. Indien niet, toon een gepaste waarschuwing.
- Controleer of alle shellscripts in de repository **executable** zijn. Worden er bestanden met de extensie `.sh` gevonden die niet uitvoerbaar zijn, bied dan aan om deze uitvoerbaar te maken voor de gebruiker. Indien gewenst, commit dan ook direct de wijzigingen met de commit-boodschap *"Make scripts executable"*.

- Controleer op ongepaste bestanden binnen deze repository: gebruik het commando `file` om ISO-, Word-, Excel- en ELF-bestanden te signaleren. Lijst deze op in een waarschuwing.

Voorbeeld:

```
$ ./git-helper.sh check my_project
Missing README.md
$ touch check my_project/README.md
$ ./git-helper.sh check my_project
Script my_project/world_peace.sh is not executable. Grant execute
permission to file owner [y/N]?   y
Execute permission granted.
Committed changes to branch 'main'.
$ ./git-helper.sh check my_project
Found unsupported file type: 'PC_problem_screenshot.docx'
```

Log

Met `log` toon je een verkorte, meer leesbare versie van de output van het `git log`-commando op de huidige directory. Elke commit wordt op 1 enkele regel weergegeven. Deze regel bevat de boodschap van de commit, de auteur en de datum in JJJJ/MM/DD-formaat, zoals in het voorbeeld hieronder:

```
$ ./git-helper.sh log
Bugfix after review | Developer B | 2024-10-11
Finished first function | Developer A | 2024-10-08
Initial commit | Developer A | 2024-10-07
```

Stats

Met `stats` toon je een overzicht van enkele statistieken over de huidige branch van de repository:

- Het aantal commits
- Het aantal contributors

Voorbeeld:

```
$ ./git-helper.sh stats
37 commits by 4 contributors
```

Undo

Met `undo` verwijder je de laatste commit uit je working tree, met behoud van de lokale wijzigingen in de working directory.

Voorbeeld:

```
$ ./git-helper.sh undo
Undo of last commit "Final change before go-live" successful.
```

Sync

Met `sync` synchroniseer je de lokale branch met de remote:

- Zijn er lokale wijzigingen? Doe eerst `git stash`
- Doe `git pull --rebase`. Als er conflicten zijn, dan stopt het hier en moet de gebruiker die eerst oplossen (geef output van `git status` om de gebruiker op weg te zetten).
- Doe `git push`
- Doe `git push` van alle labels (tags).
- Doe `git stash pop` als er lokale wijzigingen waren.

Voorbeeld:

```
$ ./git-helper.sh sync
Local changes detected.
Local changes stashed.
Remote changes pulled successfully.
Local changes unstashed.
```