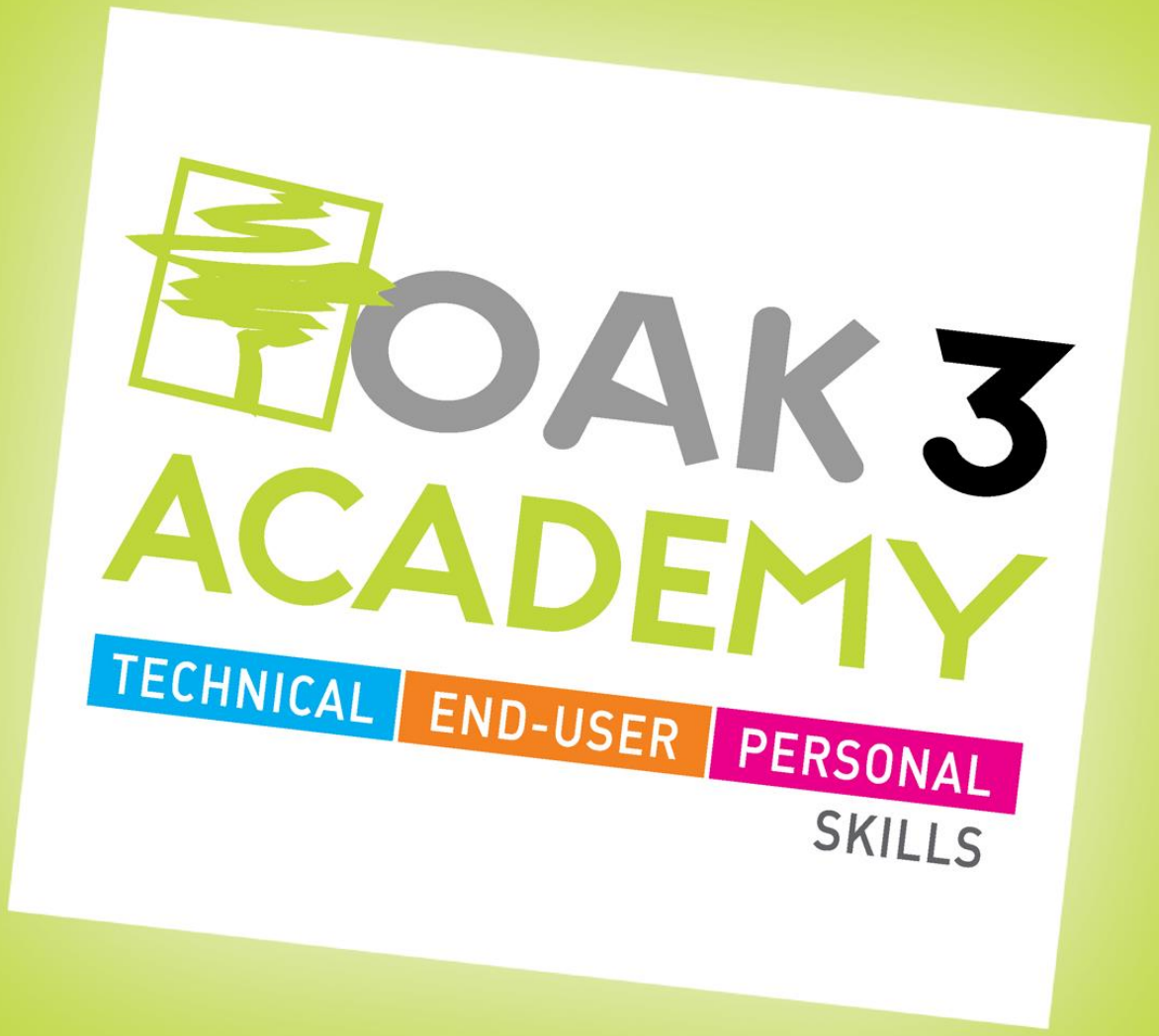


Git

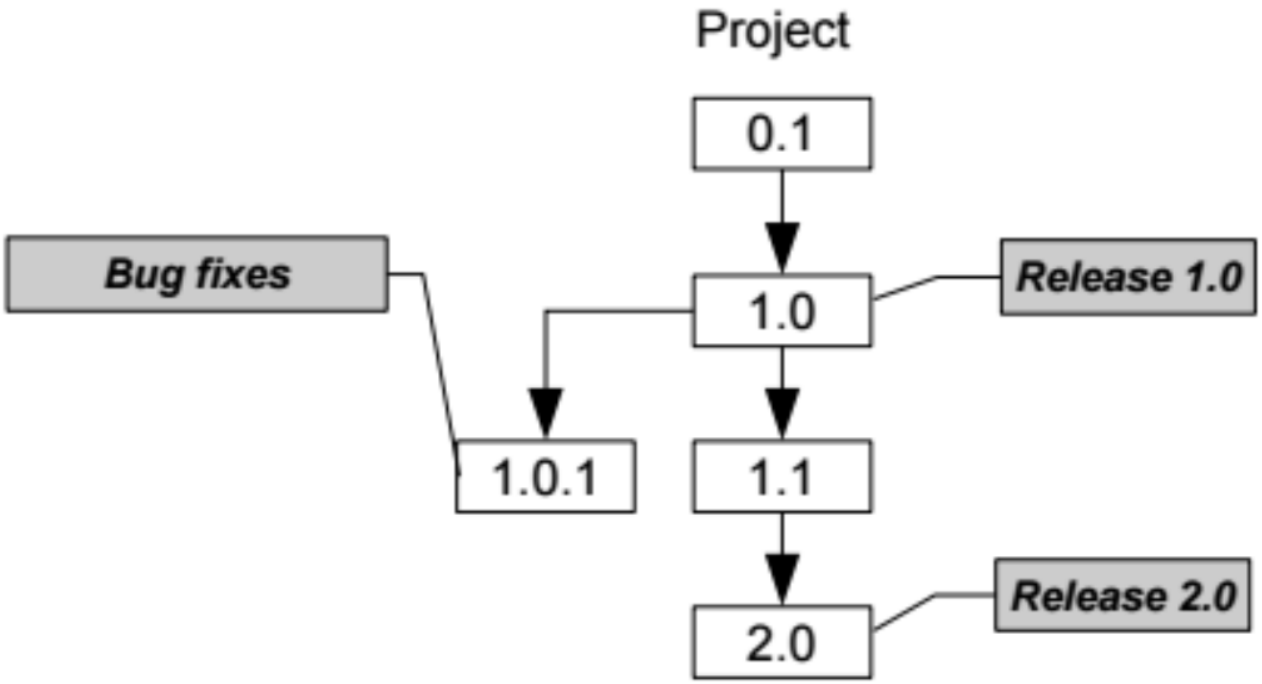
Table of contents

OAK 3 ACADEMY
TECHNICAL END-USER PERSONAL SKILLS

Versiebeheersysteem

OAK 3 ACADEMY
TECHNICAL END-USER PERSONAL SKILLS

Inleiding



OAK 3 ACADEMY
TECHNICAL END-USER PERSONAL SKILLS

Wat is het?

Bestanden bijhouden

Terugkeren naar oudere versies

Repository: volledige collectie bestanden en versies

OAK 3 ACADEMY
TECHNICAL END-USER PERSONAL SKILLS

Centraal vs Gedistribueerd

Centraal: beheer centraal op 1 locatie.

BV: Subversion

Gedistribueerd: Op verschillende plaatsen

BV: Git

→ Veel populairder: gebruikers kunnen offline werken.

Locking

Pessimistic locking: Gebruiker A vergrendeld bestand. Niemand anders kan eraan.

Optimistic locking: op lokale kopie aanpassingen doen → Conflicten oplossen indien nodig!

GIT

Hoe lokaal met git werken

OAK 3 ACADEMY
TECHNICAL END-USER PERSONAL SKILLS

Wat is het?

Versiebeheersysteem

Lokale kopie

Keuze wanneer synchroniseren met server

Nadeel: Conflicten

OAK 3 ACADEMY
TECHNICAL END-USER PERSONAL SKILLS

Git info

Downloaden: https://git-scm.com/downloads

Documentatie: https://git-scm.com/docs

Git cheat cheet: https://services.github.com/on-demand/downloads/github-git-cheat-sheet.pdf

OAK 3 ACADEMY
TECHNICAL END-USER PERSONAL SKILLS

Git configureren

git config --list → lijst van instellingen.

git config –global --add user.name “Kenneth”

git config –global --add user.email “kenneth.vangijsel@oak3.be”

→ Met deze info bewaart git alles.

OAK 3 ACADEMY
TECHNICAL END-USER PERSONAL SKILLS

Lokale repository aanmaken

git init <directory>

.git map wordt aangemaakt→ Administratieve bestanden

Bv:

git init C:\Users\vangike\Documents\Cursussen OAK3\Git\MyFirstRepository

OAK 3 ACADEMY
TECHNICAL END-USER PERSONAL SKILLS

Opdracht

In deze opdracht maken we een eenvoudig project en voegen we vervolgens een *repository* toe.

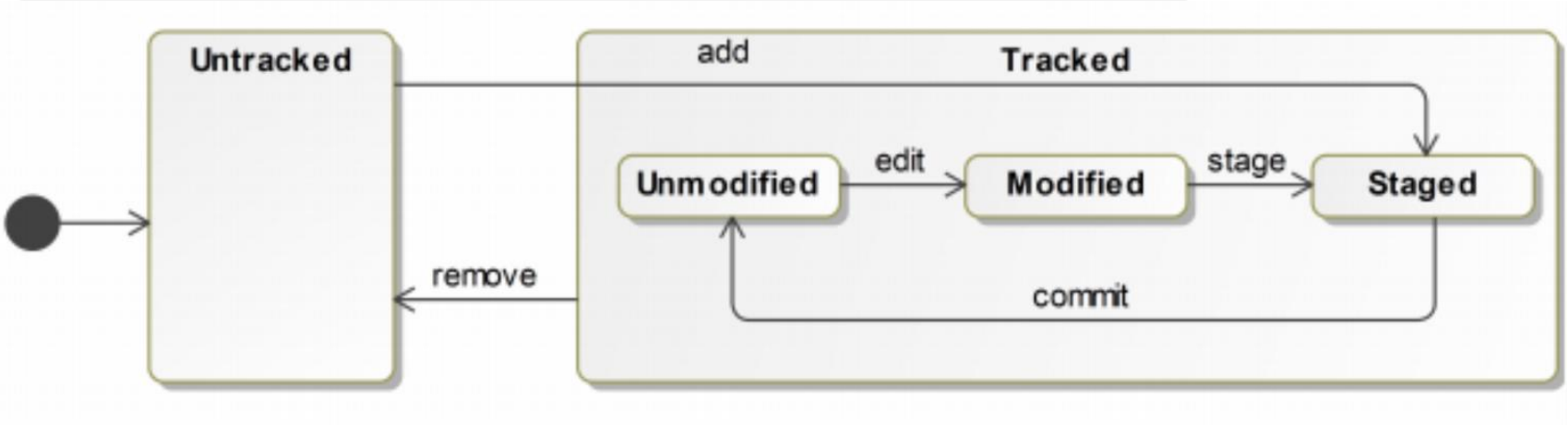
- Maak een nieuwe map voor een nieuw project. Bijvoorbeeld ***C:\GIT\HelloWorld***
- Voeg in deze map een nieuwe bestand ***HelloWorld.txt*** toe met de volgende tekst "Hello World".
- Open een commandovenster in de projectmap (of navigeer naar de projectmap) en voer het volgende commando uit:

```
git init
```

- Ga na of de map ***.git*** aangemaakt werd en wat de inhoud ervan is.

OAK 3 ACADEMY
TECHNICAL END-USER PERSONAL SKILLS

Working Directory



OAK 3 ACADEMY
TECHNICAL END-USER PERSONAL SKILLS

Stage changes

git add --all of git add . : alle files toevoegen

git add hello.txt

git add *.txt

git rm --cached hello.txt: verwijderen van staged file

Status bekijken

git status

→Welke wijzigingen zijn gestaged, welke niet?

```
$ git status
On branch master
Changes to be committed:
  (use "git reset HEAD <file>..." to unstage)

        new file:   .gitignore
        new file:   stash.txt
```

OAK 3 ACADEMY
TECHNICAL END-USER PERSONAL SKILLS

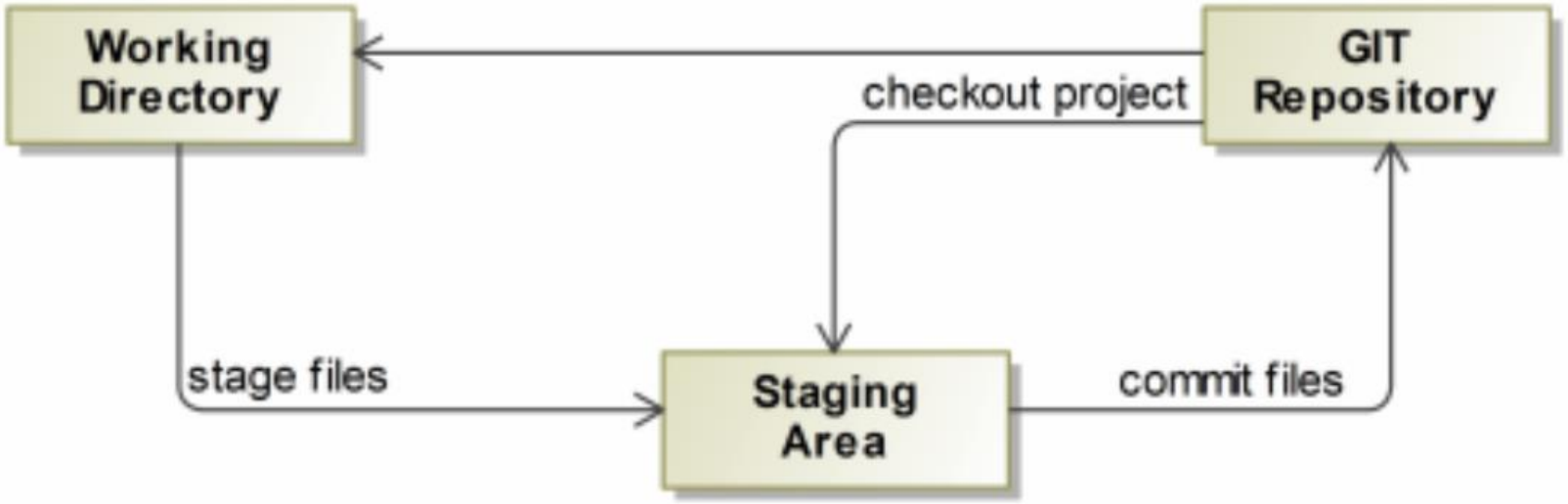
Opdracht

In deze opdracht gaan we ons bestand toevoegen zodat het in de toestand *Tracked/Staged* terechtkomt.

- Voer de volgende commando's uit:

```
git status
git add HelloWorld.txt
git rm --cached HelloWorld.txt
git add HelloWorld.txt
git status
```

Commit



Commit staged changes

git add hello.txt
git commit –m “add textfile”

git add “hello2.txt”
git commit --amend
→ File wordt toegevoegd aan vorige commit

Vanaf je een commit gedaan hebt wordt er een versie lokaal bijgehouden

Stage and commit

git commit -a -m “Message”

→Achterliggend:

- git add .
- git commit -m “Message”

Reset

git reset

→Permanent ongedaan maken van een commit.

→Vooral gebruikt voor lokale wijzigingen

git reset --hard: teruggaan naar laatste commit

In deze opdracht gaan we het bestand dat zich in de toestand *Staged* bevindt, committen. Verder gaan we experimenteren met maken van wijzigen, stagen, committen en ook verwijderen uit de index.

- Voer de volgende commando's uit:
```
git commit
git status
```

- Voeg een regel tekst toe in het bestand ***HelloWorld.txt***.
- Voer de volgende commando's uit:
```
git status
git add HelloWorld.txt
git commit -m "Just another commit"
```

- Maak opnieuw een wijziging in het bestand en voer de volgende commando's uit:
```
git add *
git commit -m "Message"
```

- Maak weer een wijziging in het bestand en voer de volgende commando's uit:
```
git commit -a -m "Message"
```

- Maak nog een wijziging in het bestand en voer de volgende commando's uit:
```
git add HelloWorld.txt
git status
git reset HelloWorld.txt
git status
git commit -a -m "Message"
```

Verwijderen

git rm File.txt

→Bestand wordt verwijderd uit working directory

→Bestand wordt verwijderd uit index

git commit → Bestand wordt definitief verwijderd uit repository

Opdracht

- Maak een nieuw bestand ***Hello.tmp.***
- Voeg dit bestand aan de index (*staging area)* toe en commit:

```
git add Hello.tmp
git commit -m "Hello.tmp added" Hello.tmp
git status
```

- Verwijder nu dit bestand met:

```
git rm Hello.tmp
```

- Ga na of het bestand zich nog in de *working directory* bevindt.
- Maak het bestand ***Hello.tmp*** opnieuw aan en commit:

```
git add Hello.tmp
git commit -m "Hello.tmp added" Hello.tmp
```

- Verwijder nu het bestand enkel uit de index en commit.

```
git rm --cached Hello.tmp
git commit -m "Hello.tmp removed"
git status
```

.gitignore

Bestanden die niet mogen gestaged worden.

- Gecompileerde code
- Logged files
- Hidden system files
- …

.gitignore

File aanmaken: touch .gitignore

Voorbeelden:

`Hello.tmp`	Negeert het bestand ***Hello.tmp***.
`*.tmp`	Negeert alle bestanden met extensie ***tmp.***
`tmp/`	Negeert de map ***tmp*** en alle onderliggende mappen.
`/tmp/`	Negeert de map ***tmp*** maar niet de onderliggende mappen.
`!Test.tmp`	Negeert het bestand ***Test.tmp*** niet.

OAK 3 ACADEMY
TECHNICAL END-USER PERSONAL SKILLS

Opdracht

- Maak het bestand ***.gitignore*** aan en voeg hierin de volgende tekst toe:
  ```
  Hello.tmp
  ```
- Commit dit bestand als volgt:
  ```
  git add .gitignore
  git commit -m ".gitignore added" .gitignore
  ```
- Ga na of GIT het bestand ***Hello.tmp*** inderdaad negeert:
  ```
  git status
  ```

Logs bekijken

git log

→Toon committed files → Lijst van de project geschiedenis

```
$ git log
commit 4b3193f16467650db660c64dfd7b6dd5cd6f0cff
Author: Kenneth <kenneth.vangijsel@gmail.com>
Date:   Wed Aug 30 13:43:08 2017 +0200

    added stashed file and gitignore file

commit 1fc6962aa17f2af39c2693f17433cffe7f7f8024
Author: Kenneth <kenneth.vangijsel@gmail.com>
Date:   Wed Aug 30 12:46:41 2017 +0200

    new text file
```

OAK 3 ACADEMY
TECHNICAL END-USER PERSONAL SKILLS

Logs bekijken

```
$ git log
commit 4b3193f16467650db660c64dfd7b6dd5cd6f0cff
Author: Kenneth <kenneth.vangijsel@gmail.com>
Date:   Wed Aug 30 13:43:08 2017 +0200

    added stashed file and gitignore file

commit 1fc6962aa17f2af39c2693f17433cffe7f7f8024
Author: Kenneth <kenneth.vangijsel@gmail.com>
Date:   Wed Aug 30 12:46:41 2017 +0200

    new text file
```

Hashcode van commit-object.

Logs bekijken

git log -2: 2 laatste commits

git log --pretty=full: Uitvoer wordt beter leesbaar

…

Opdracht

- Voer de volgende commando's uit:

```
git log --help
git log
git log -4
git log --pretty=oneline
git log --pretty=short
git log --pretty=full
git log --pretty=fuller
```

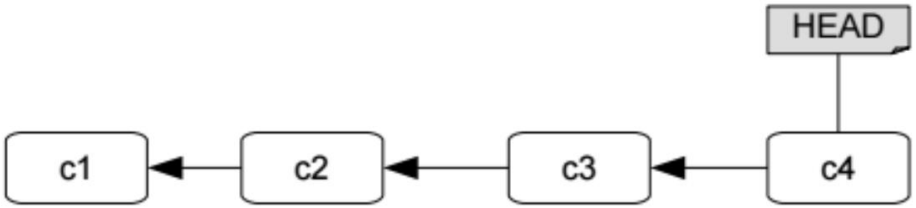
Checkout

git checkout

→Checking out files, commits or branches

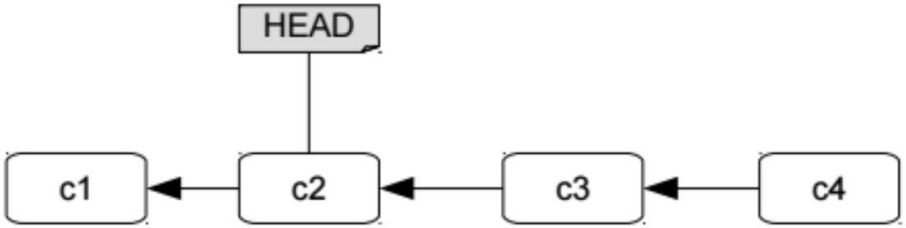
→ Wordt niet opgeslagen in master branch

Checkout



git checkout <hashcode>

→ Specifieke commit uitchecken



Opdracht

- Vraag de volledige geschiedenis op met het volgende commando:
  `git log --decorate --all`
- Merk de positie van HEAD op.
- Kies vervolgens een versie uit het verleden en check deze uit met de verkorte *hashcode*:
  `git checkout <hashcode>`
- Ga na wat de inhoud van het bestand ***HelloWorld.txt*** nu is.
- Vraag de geschiedenis op:
  `git log --decorate`
  `git log --decorate --all`
- Merk de positie van HEAD op.
- Check opnieuw de laatste versie van de geschiedenis uit door de *hashcode* van deze versie op te geven.

Tags

git tag

- Lightweight: verwijzing naar versie

Bv: git tag V1.0.0

- Annotated: extra info (Bv: naam, datum)

Bv: git tag -a V1.0.0 –m “Release V1.0.0”

Tags

git checkout V1.0.0

→ Zo moet je hashcode van commit niet kennen.

Opdracht

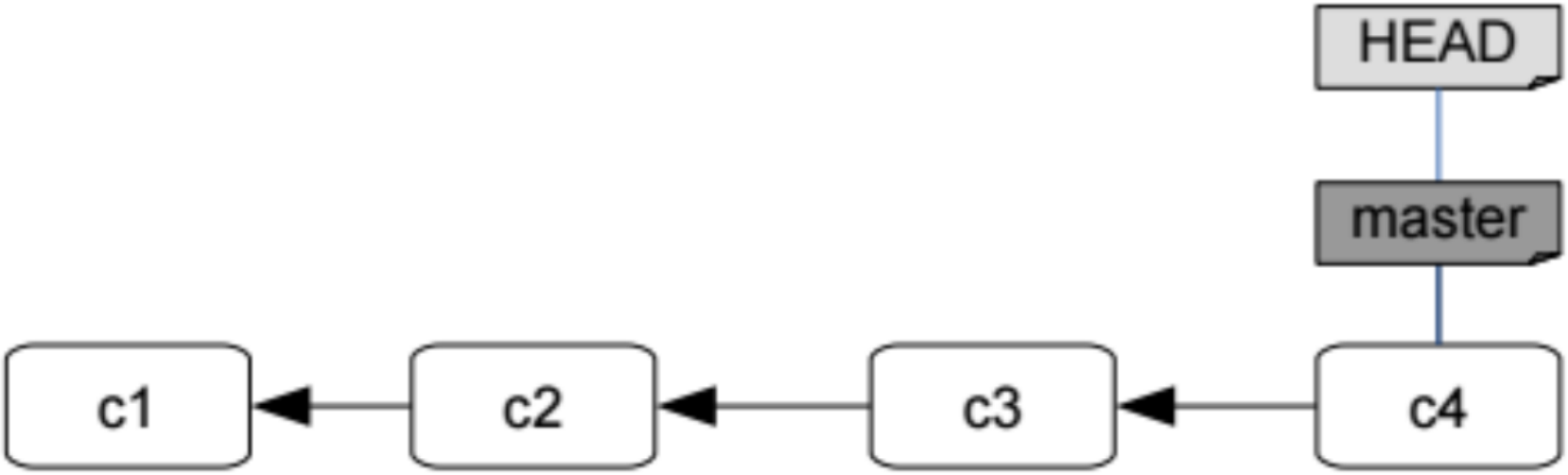
In deze opdracht gaan we een *tag* toevoegen aan een bepaalde versie uit het verleden. Nadien checken we deze versie uit op basis van de *tag*.

- Vraag de volledige geschiedenis op met het volgende commando:
  `git log --decorate --all`
- Kies een versie uit het verleden en voeg hieraan een *tag* toe:
  `git tag -a V1.0.0 <hashcode> -m "Release V1.0.0"`
- Vraag een overzicht van alle *tags* op:
  `git tag`
- Voeg aan de huidige uitgecheckte versie ook een *tag* toe:
  `git tag -a V1.0.1 -m "Release V1.0.1"`
- Vraag opnieuw een overzicht van alle *tags* op:
  `git tag`
- Vraag de volledige geschiedenis op met:
  `git log --decorate --all`
- Merk op dat de *tags* in het overzicht zijn opgenomen.
- Check versie V1.0.0 uit:
  `git checkout V1.0.0`
- Check opnieuw V1.0.1 uit:
  `git checkout V1.0.1`

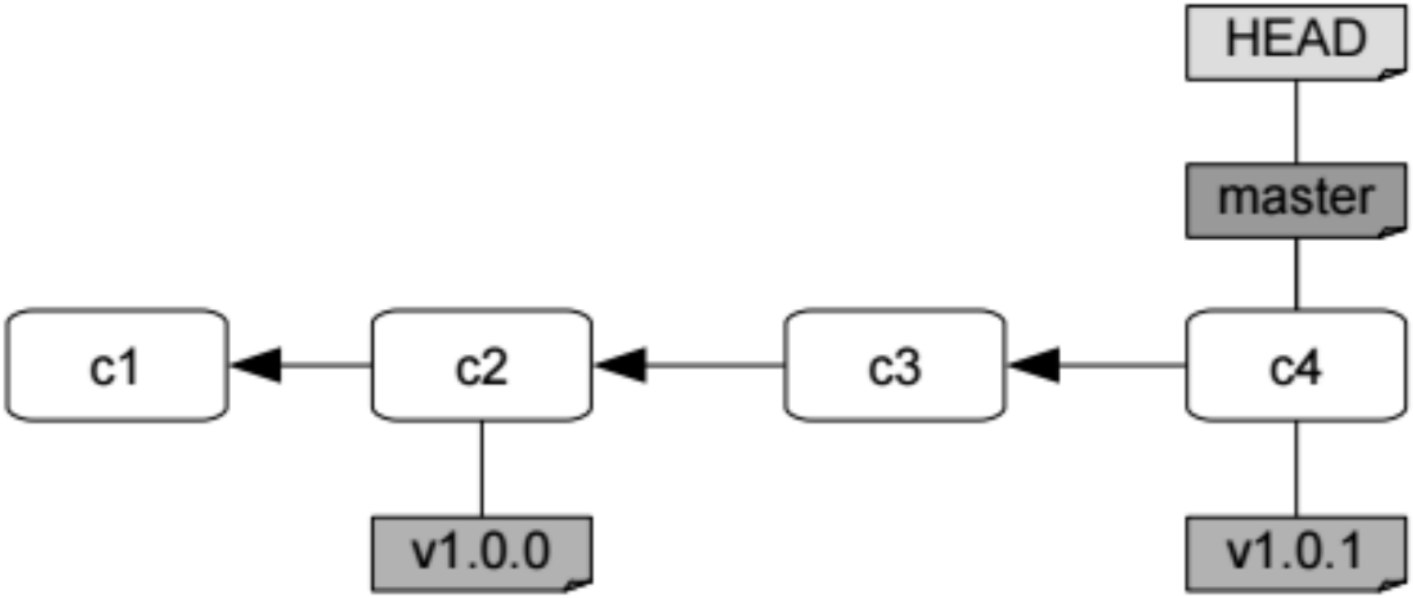
Branch

= Reeks van versies

Hoofdbranch = **MASTER**



Branch



Terugkeren naar vroegere versie: git checkout v1.0.0

Huidige versie: git checkout master

Branch

= Onafhankelijk ontwikkelen

git branch: lijst van alle branches in repo

git branch myBranch: aanmaken

git branch –d myBranch: verwijderen

git branch –m betterName: hernoemen van huidige branch

Nieuwe Branch

Backup branch aanmaken:

```
vangike@HTPXPF2 MINGW64 ~/Documents/Cursussen OAK3/Git/MyFirstRepository (master)
$ git branch backup
```

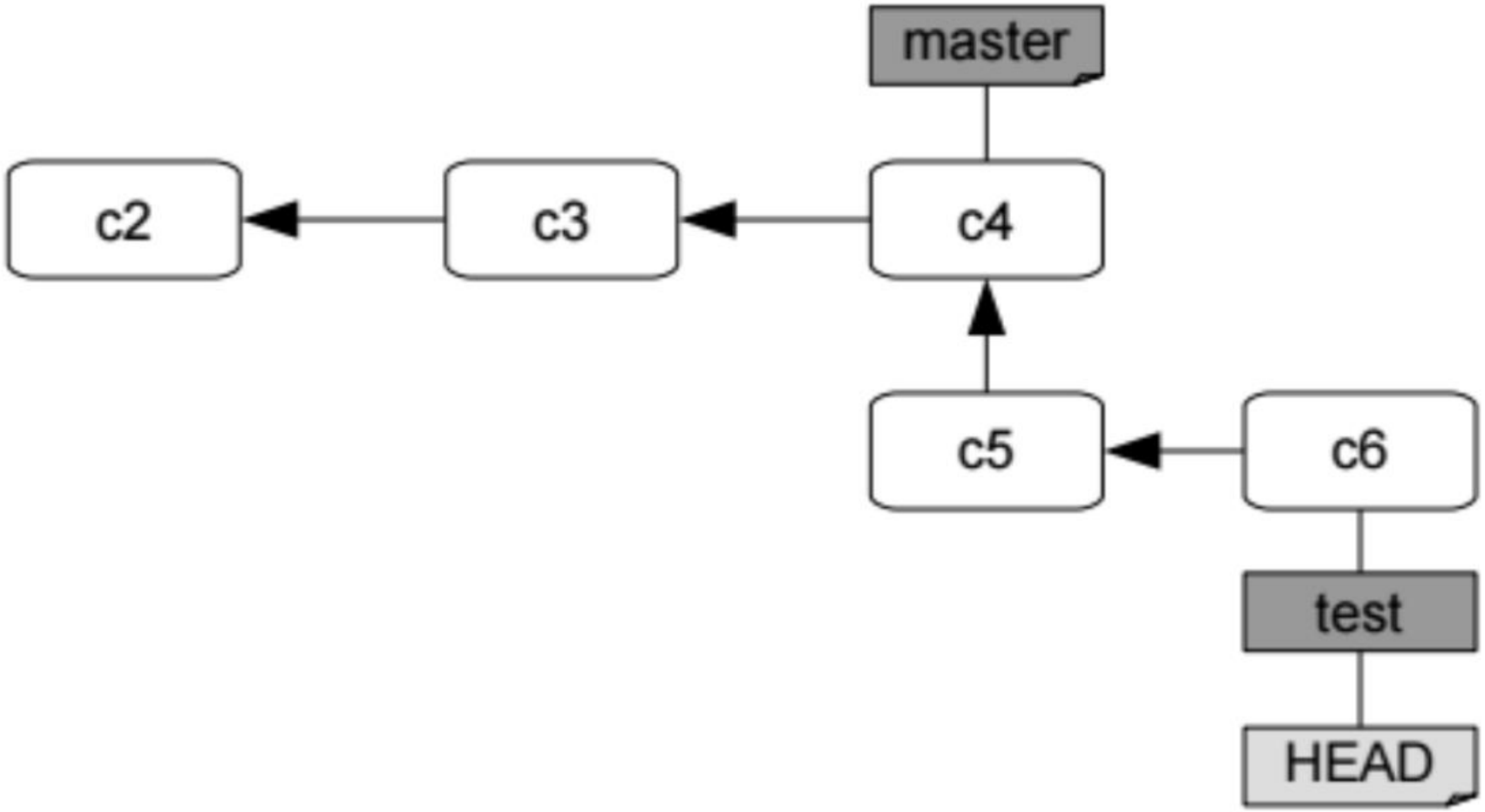
Op deze branch verder werken:

```
vangike@HTPXPF2 MINGW64 ~/Documents/Cursussen OAK3/Git/MyFirstRepository (master)
$ git checkout backup
Switched to branch 'backup'

vangike@HTPXPF2 MINGW64 ~/Documents/Cursussen OAK3/Git/MyFirstRepository (backup)
$
```

OAK 3 ACADEMY
TECHNICAL END-USER PERSONAL SKILLS

Branch



Opdracht

- Check de laatste versie van de hoofdtak uit:
  `git checkout master`
- Maak een nieuwe zijtak met de naam **test**:
  `git branch test`
- Check deze zijtak uit:
  `git checkout test`
- Controleer de status:
  `git status`
- Breng een wijziging aan in het bestand ***HelloWorld.txt***.
- Commit de wijziging:
  `git commit -a -m "Just a test"`
- Vraag de volledige geschiedenis op:
  `git log --decorate`
- Check de laatste versie van de tak **master** uit:
  `git checkout master`
- Check opnieuw de laatste versie van de tak **test** uit:
  `git checkout test`
- Maak nog enkele wijzigingen en check die vervolgens in:
  `git commit -a -m "Keep on testing"`

OAK 3 ACADEMY
TECHNICAL END-USER PERSONAL SKILLS

Merge Branch

Merge branch met master:

```
vangike@HTPXPF2 MINGW64 ~/Documents/Cursussen OAK3/Git/MyFirstRepository (backup)
$ git checkout master
Switched to branch 'master'
Your branch is up-to-date with 'origin/master'.

vangike@HTPXPF2 MINGW64 ~/Documents/Cursussen OAK3/Git/MyFirstRepository (master)
$ git merge backup
Updating 574e663..72cf2cd
Fast-forward
 Backup/.gitignore | 37 +++++++++++++++++++++++++++++++++++++
 Backup/File.txt   |  0
 Backup/Hello.txt  |  0
 Backup/Hello2.txt |  0
 Backup/README.md  |  1 +
 Backup/new.txt    |  1 +
 Backup/stash.txt  |  0
 7 files changed, 39 insertions(+)
 create mode 100644 Backup/.gitignore
 create mode 100644 Backup/File.txt
 create mode 100644 Backup/Hello.txt
 create mode 100644 Backup/Hello2.txt
 create mode 100644 Backup/README.md
 create mode 100644 Backup/new.txt
 create mode 100644 Backup/stash.txt

vangike@HTPXPF2 MINGW64 ~/Documents/Cursussen OAK3/Git/MyFirstRepository (master)
$
```

Merge Branch

Melding Fast-forward: Geen wijzigingen in master branch.

= Eenvoudige merge

Opdracht

- Check de zijtak **master** uit:
  `git checkout master`
- Voeg de zijtak **test** samen met de huidige tak **master**:
  `git merge test`
- Vraag de geschiedenis op:
  `git log --decorate --all`
- Verwijder de zijtak **test**:
  `git branch -d test`
- Vraag opnieuw de geschiedenis op:
  `git log --decorate --all`

Merge Conflicts

Indien er op de master en een andere branch tegelijk wijzigingen zijn aangebracht aan hetzelfde bestand.

```
Hello Moon
Hello Mars
Hello bla
A test
Another addition in test
<<<<<<< HEAD
Change in branch master
=======
Change in branch test
>>>>>>> test
```

Inhoud working directory

Inhoud test branch

Merge Conflicts

Tekens weghalen. Bestand committen.

→ Conflict in orde!

OAK 3 ACADEMY
TECHNICAL END-USER PERSONAL SKILLS

Opdracht

- Check de hoofdtak (*master)* opnieuw uit:
  `git checkout master`
- Maak een zijtak met de naam **featureX**:
  `git branch featureX`
- Check vervolgens deze zijtak uit:
  `git checkout featureX`
- Breng enkele wijzigingen aan in het bestand ***HelloWorld.txt.***
- Check deze versie opnieuw in:
  `git commit -a -m "Change in branch featureX"`
- Check de hoofdtak uit:
  `git checkout master`
- Breng enkele wijzigingen aan in het bestand ***HelloWorld.txt***.
- Check deze versie opnieuw in:
  `git commit -a -m "Change in branch master"`
- Voeg de zijtak samen met de hoofdtak:
  `git merge featureX`
- Merk de melding van het conflict op.
- Los het conflict op.
- Check deze versie opnieuw in:
  `git commit -a -m "Merged with branch featureX"`
- Herhaal deze stappen enkele keren door telkens een nieuwe zijtak te creëren.

Branch in het verleden

Branch op basis van versie uit verleden.

BV: bugfix V1.0.0

→git checkout V1.0.0

→git branch bugfix

→Sneller: git checkout –b bugfix V1.0.0

Opdracht

- Check de versie met *tag* ***V1.0.0*** uit en maak tegelijkertijd een nieuwe zijtak:
  `git checkout -b bugfix V1.0.0`
- Breng een wijziging aan in het bestand ***HelloWorld.txt.***
- Commit de nieuwe versie:
  `git commit -a -m "Bug fixed"`
- Check de tak **master** uit:
  `git checkout master`
- Voeg de zijtak **bugfix** samen met **master**:
  `git merge bugfix`
- Los de conflicten op.
- Check de samengevoegde versie opnieuw in:
  `git commit -a -m "Merged with branch bugfix"`
- Vraag de geschiedenis van het project op:
  `git log --decorate --graph`

Stashing changes

git stash

→Wijzigingen opslaan voor later gebruik. Lokale kopie wordt teruggeplaatst. Bv: snelle bugfix oplossen (huidig werk mag niet verloren gaan)

git add stash.txt
git stash
git stash list → toon alle stashed files
git stash apply → plaats file terug
git stash drop → verwijderen van een versie

GIT

Synchroniseren met remote server

OAK 3 ACADEMY
TECHNICAL END-USER PERSONAL SKILLS

Remote Repository

Code delen met anderen

Andere teamleden kunnen dan een lokale kopie ervan maken

Andere kunnen hun bijdrage doen

Repo Managment Service

- GitHub: https://github.com/
- BitBucket: https://bitbucket.org/
- GitLab: https://about.gitlab.com/

→ Account aanmaken

Remote

git remote

→Aanmaken, bekijken en verwijderen van connecties naar andere repo’s

git remote add <name> <url>: aanmaken

Remote

```
vangike@HTPXPF2 MINGW64 ~/Documents/Cursussen OAK3/Git/MyFirstRepository (master)
$ git remote add origin https://github.com/KennethVG/MyFirst.git
```

Aanmaken van repo op github

```
vangike@HTPXPF2 MINGW64 ~/Documents/Cursussen OAK3/Git/MyFirstRepository (master)
$ git remote -v
origin  https://github.com/KennethVG/MyFirst.git (fetch)
origin  https://github.com/KennethVG/MyFirst.git (push)
```

Lijst van remote connecties

OAK 3 ACADEMY
TECHNICAL END-USER PERSONAL SKILLS

Push

Code lokaal gelijk trekken met deze van remote repo.

git push origin master: code lokaal op remote plaatsen. Rechtreeks op de master branch.

git push --set --upstream origin master: origin master is nu standaard remote repo

Opdracht

- Maak nu zelf een remote repository aan op github en plaats je huidige projectje erop.
- Maak voor dit project deze repo standaard
- Wijzig wat files en plaats deze wijzingen hier ook op.
- Vraag met behulp van git log --decorate --graph de geschiedenis op

Repository kopiëren

git clone <directory>

Bv:

git clone https://github.com/KennethVG/TestProject.git

OAK 3 ACADEMY
TECHNICAL END-USER PERSONAL SKILLS

Fetch

git fetch

Commits importeren van remote naar lokale repo.

Geen effect op lokale werk: enkel bekijken van de wijzigingen.

Pull

git pull

Code lokaal gelijk trekken met deze van remote repo.

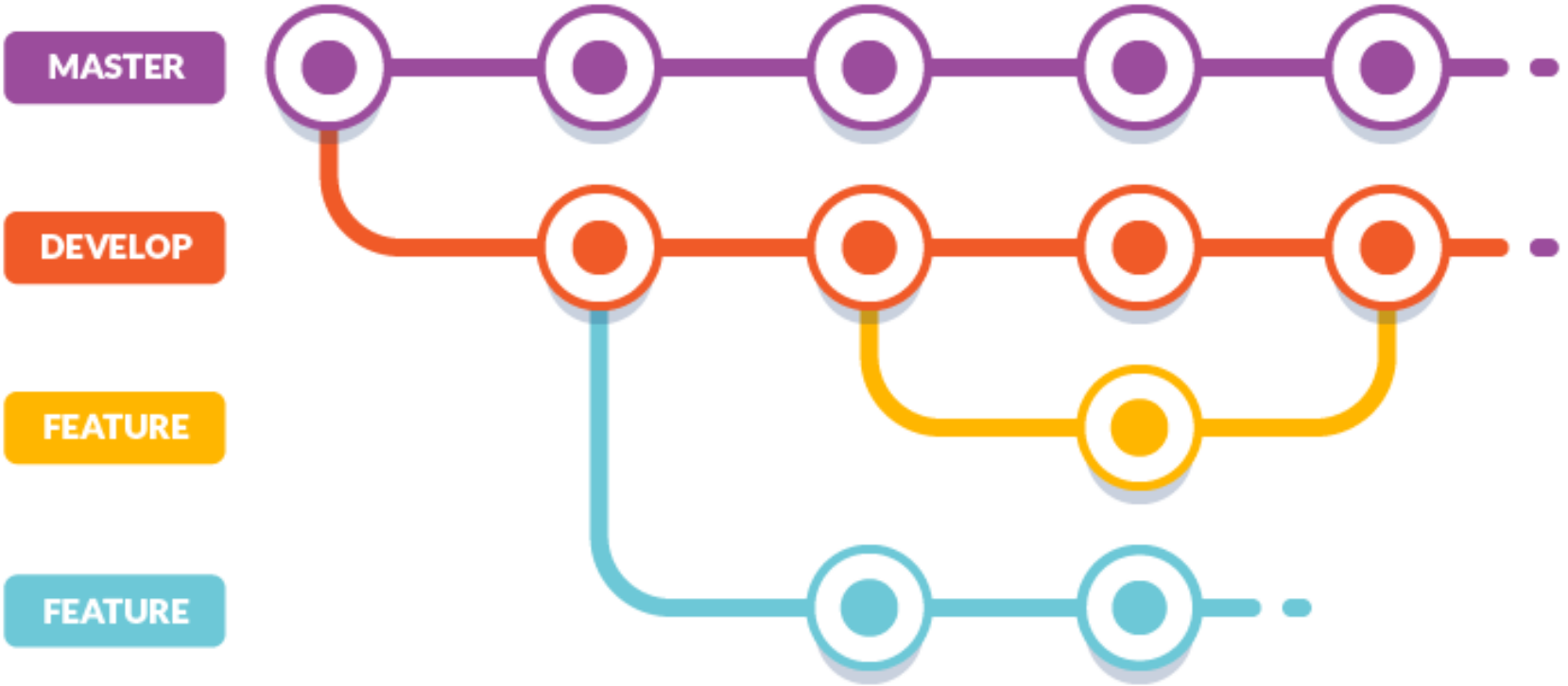
Eventuele merge conflicts oplossen

Git workflow

- Init (clone) repo
- Wijzig bestanden
- Stage
- Review
- Commit met duidelijk bericht
- Push

Git workflow

Schema:



EXTRA: Rebase Branch

Rebase branch met master:

- Cleaner history of git.
- Merge zonder commit

OAK 3 ACADEMY
TECHNICAL END-USER PERSONAL SKILLS

EXTRA: Revert

git revert

→Ongedaan maken van een commit met een nieuwe commit

→Project geschiedenis wordt bijgehouden

OAK 3 ACADEMY
TECHNICAL END-USER PERSONAL SKILLS

EXTRA: Clean

git clean

→Nieuwe files verwijderen

→Kan niet ongedaan worden

git clean –n: toont de files

git clean –f: verwijderd de files

OAK 3 ACADEMY
TECHNICAL END-USER PERSONAL SKILLS

Eindopdracht

Clone volgende repo:
https://github.com/KennethVG/vdab_genk

- Maak lokaal een mapje aan met jouw naam en push deze naar de remote
- Doe regelmatig een pull om de laatste wijzigingen ook lokaal te hebben.

OAK 3 ACADEMY
TECHNICAL END-USER PERSONAL SKILLS