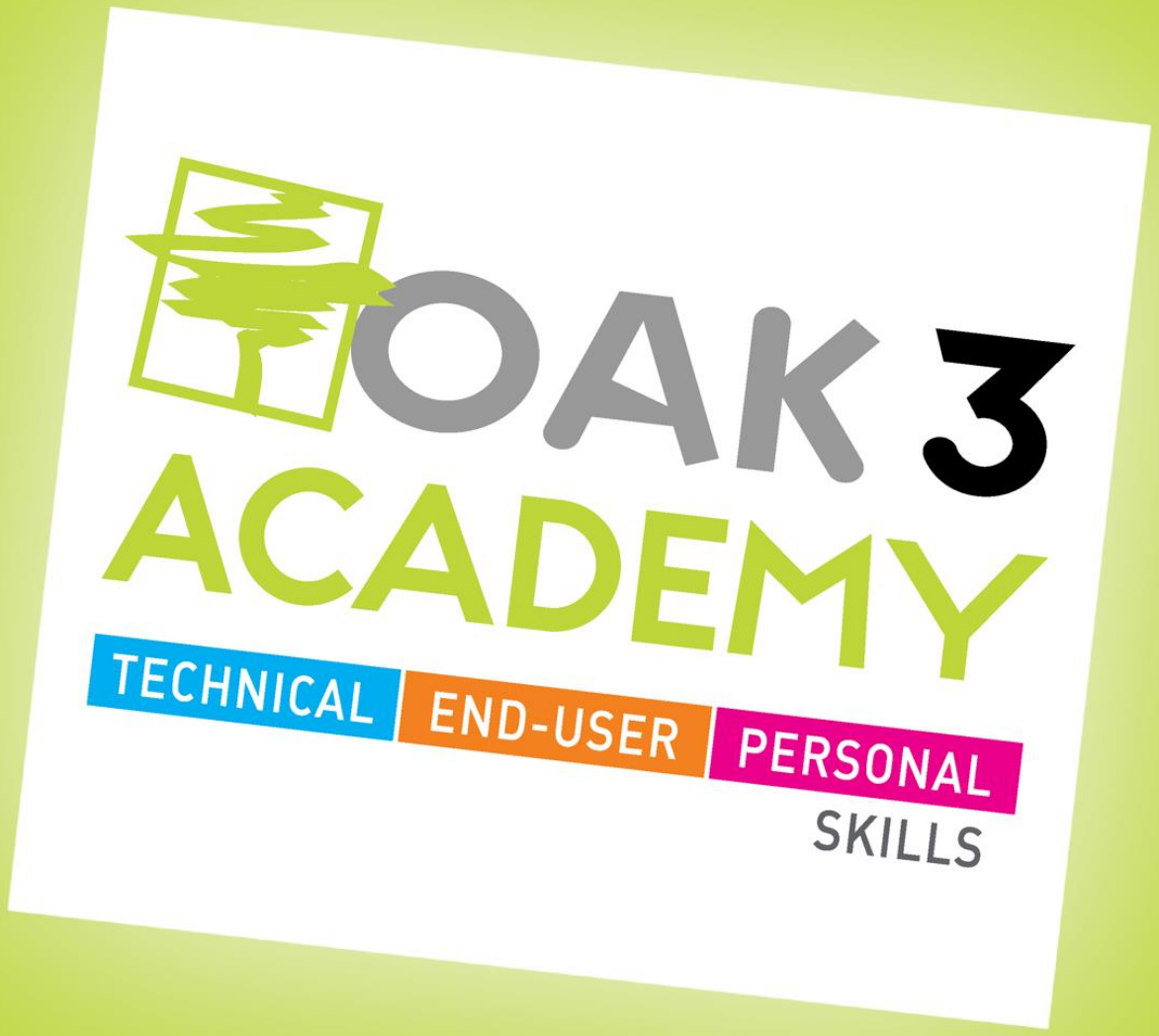
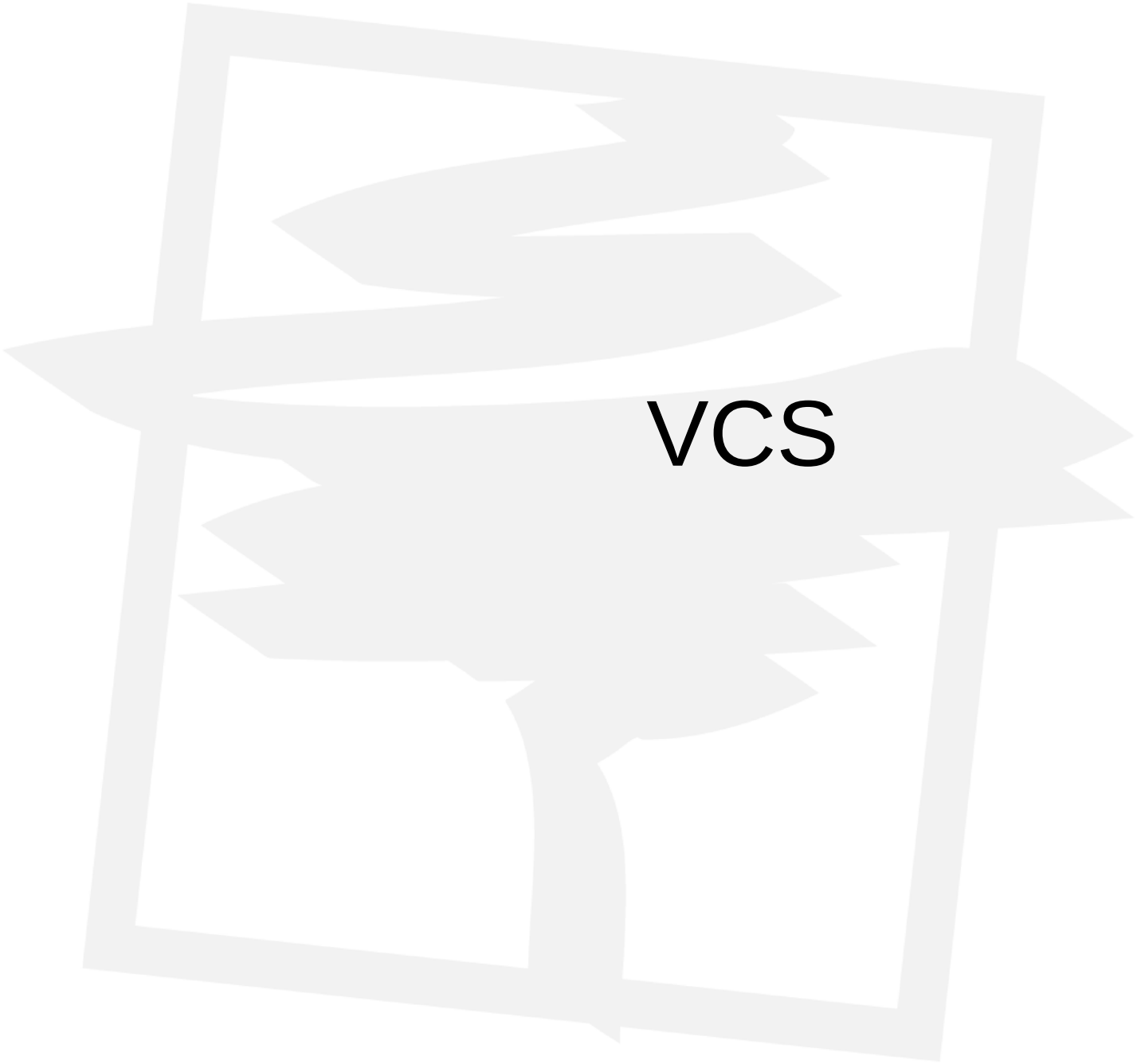


Git

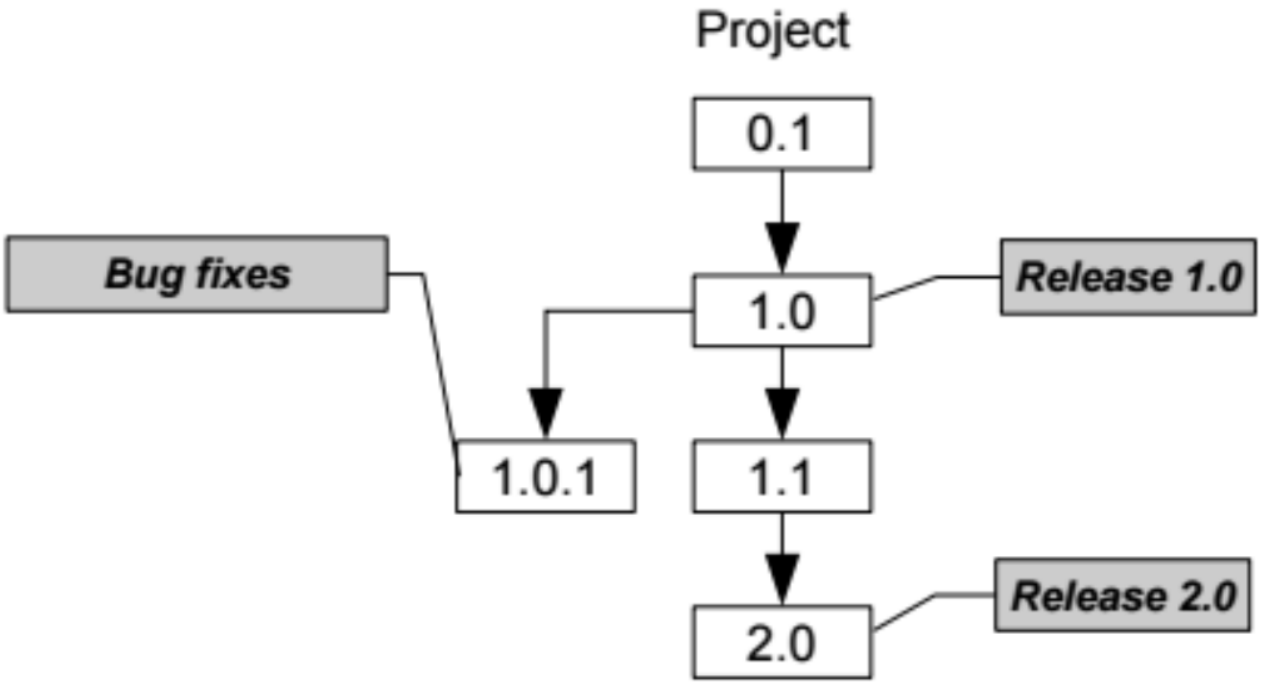
Table of contents

VCS



OAK 3 ACADEMY
TECHNICAL END-USER PERSONAL SKILLS

Introduction



What is VCS?

Keep track of files

Return to older versions

Repository: complete collection of files and versions

Central vs Distributed

Central: Management on 1 location

E.G.: Subversion

Distributed: At different locations

E.G.: Git

→ More popular: developers can work everywhere

Locking

Pessimistic locking: User A locked file. Nobody else can access the file.

Optimistic locking: make changes to local copy → Solve conflicts if necessary!

GIT

Work local

OAK 3 ACADEMY
TECHNICAL END-USER PERSONAL SKILLS

What is GIT?

VCS

Local copy

Choose when you syncronize with server

Disadvantage: Conflicts

OAK 3 ACADEMY
TECHNICAL END-USER PERSONAL SKILLS

Git info

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

Documentation: 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

Configure git

git config --list → lijst of settings

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

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

→ Git saves files with this info

OAK 3 ACADEMY
TECHNICAL END-USER PERSONAL SKILLS

Make local repo

git init <directory>

.git directory created→ Adminstrative files

Bv:

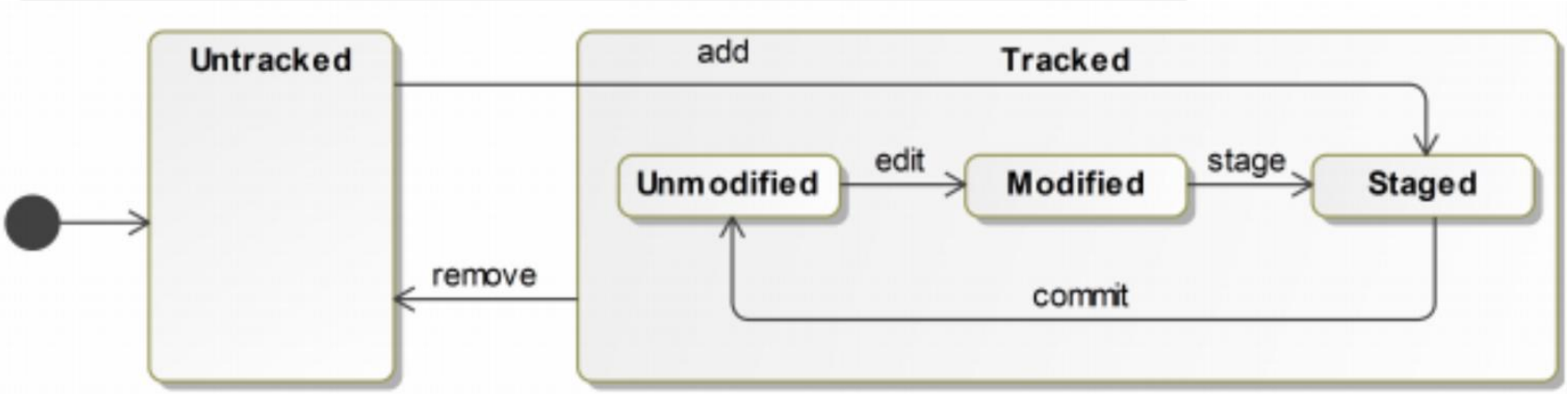
git init C:\Users\vangike\Documents\Cursussen OAK3\Git\MyFirstRepository

OAK 3 ACADEMY
TECHNICAL END-USER PERSONAL SKILLS

Exercise

- Download git
- Open CMD and run git --version
- Run:
  - git config –global --add user.name “Name”
  - git config –global --add user.email “email”
- Make new folder on desktop: GIT and a subfolder HelloWorld. Add HelloWorld.txt to this folder.
- Navigate to the new folder: run **GIT INIT**

Working Directory



Stage changes

git add --all or git add . : Add all files

git add hello.txt

git add *.txt

git rm --cached hello.txt: Delete staged file

Check status

git status

→Which changes have been staged, which are not?

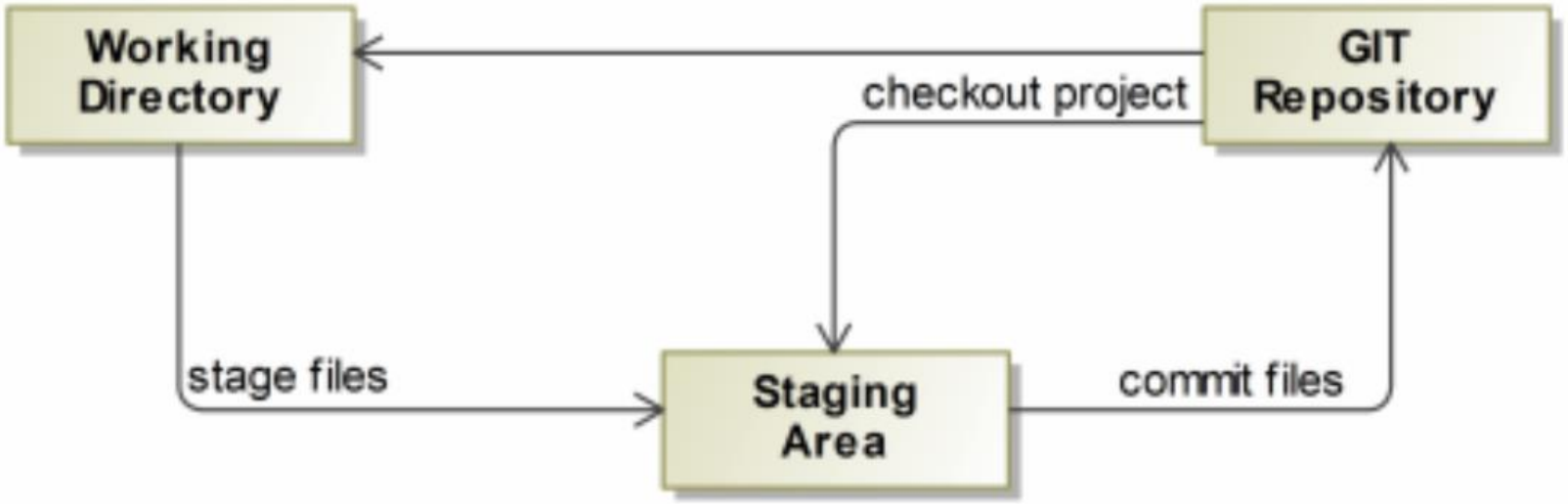
```
$ git status
On branch master
Changes to be committed:
  (use "git reset HEAD <file>..." to unstage)

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

Exercise

- Execute the following:
  - **git status**
  - **git add HelloWorld.txt**
  - **git status**

OAK 3 ACADEMY
TECHNICAL END-USER PERSONAL SKILLS

Commit



Commit staged changes

git add hello.txt

git commit –m “add textfile”

git add “hello2.txt”

git commit --amend

→ File added to last commit

When you commit, file is saved locally

Stage and commit

git commit -a -m “Message”

→In the background:

- git add .
- git commit -m “Message”

Reset

git reset

→Permanently undo a commit

→Used for local changes

git reset --hard: go back to last commit

Exercise

- Execute the following:
  - **git commit**
  - **git status**
- Add a line of text to HelloWorld.txt
- Execute the following:
  - **git status**
  - **git add HelloWorld.txt**
  - **git commit –m “Changed file”**
- Add a line of text to HelloWorld.txt
- Execute the following:
  - **git add .**
  - **git commit –m “Changed file”**

Delete

git rm File.txt

→Delete file from working directory

→File deleted from index

git commit → File removed from repo

Exercise

- Make new file Hello.tmp
- Add file to stage and commit
- Delete the file:
  - git rm Hello.tmp
- Check if the file is deleted

.gitignore

Files that may not be staged.

- Compiled code
- Logged files
- Hidden system files
- …

.gitignore

Create file: 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

Exercise

Create file: touch .gitignore

- Make new file Hello.tmp
- Add Hello.tmp to .gitignore file
- Commit:
  - git commit –m “.gitignore added” .gitignore
- git status: see if the file is ignored

OAK 3 ACADEMY
TECHNICAL END-USER PERSONAL SKILLS

Check logs

git log

→Show committed files→ List of project history

```
$ 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

Check logs

```
$ 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 of commit-object.

Check logs

git log -2: Check two last commits

git log --pretty=full: more readable output

…

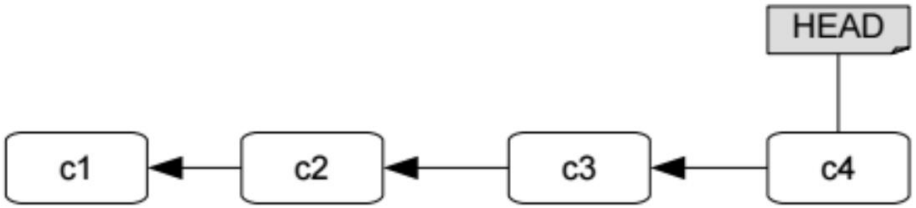
Checkout

git checkout

→Checking out files, commits or branches

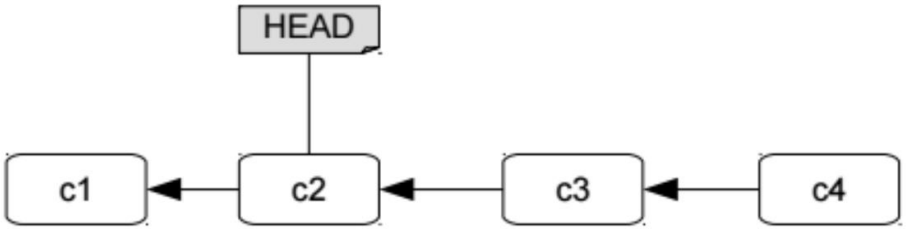
→ Not saved in master branch

Checkout



git checkout <hashcode>

→ Check out specific commit



Exercise

- git log --decorate --all
- git checkout <hashcode> (some previous version)
- Check content of HelloWorld.txt file
- git log --decorate
  git log –decorate --all
- git checkout <hashcode> (latest version)

Tags

git tag

- Lightweight: reference to a version

E.G: git tag V1.0.0

- Annotated: extra info (e.g.: name, date)

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

Tags

git checkout V1.0.0

→ Now you don’t have to know hashcode of commit.

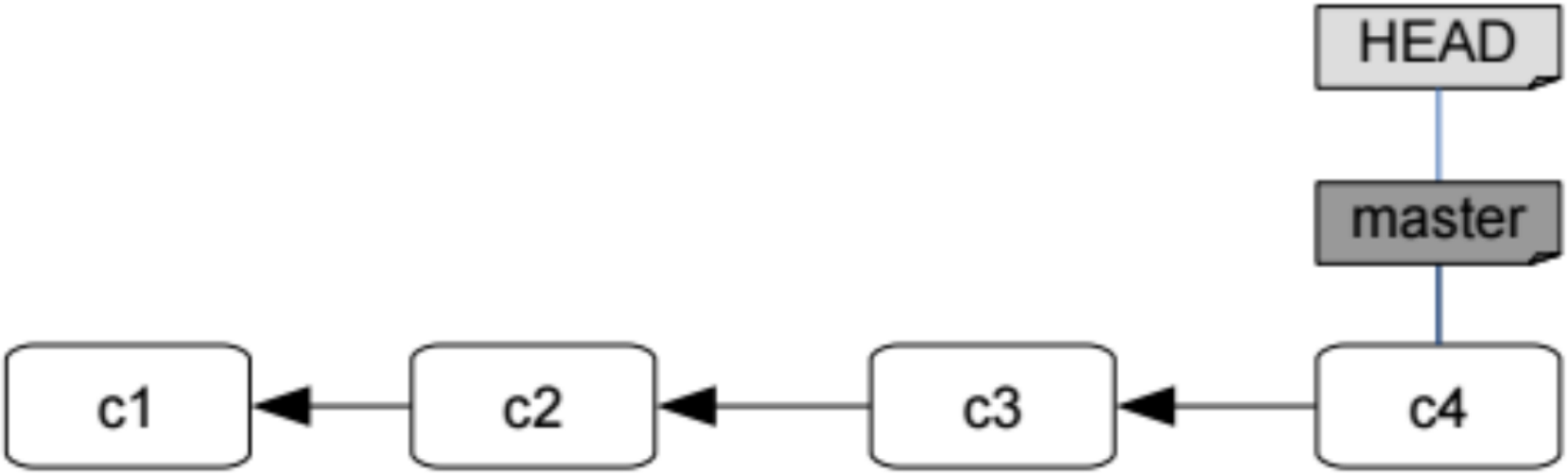
Exercise

- git log --decorate --all
- Choose a version of the past and add a tag: git tag --a V1.0.0 <hashcode> -m “Release V1.0.0”
- git tag: overview of all tags
- Add a tag to the current version (V1.0.1)
- git log --decorate --all: tags are in the logs
- git checkout v1.0.0
- git checkout v1.0.1

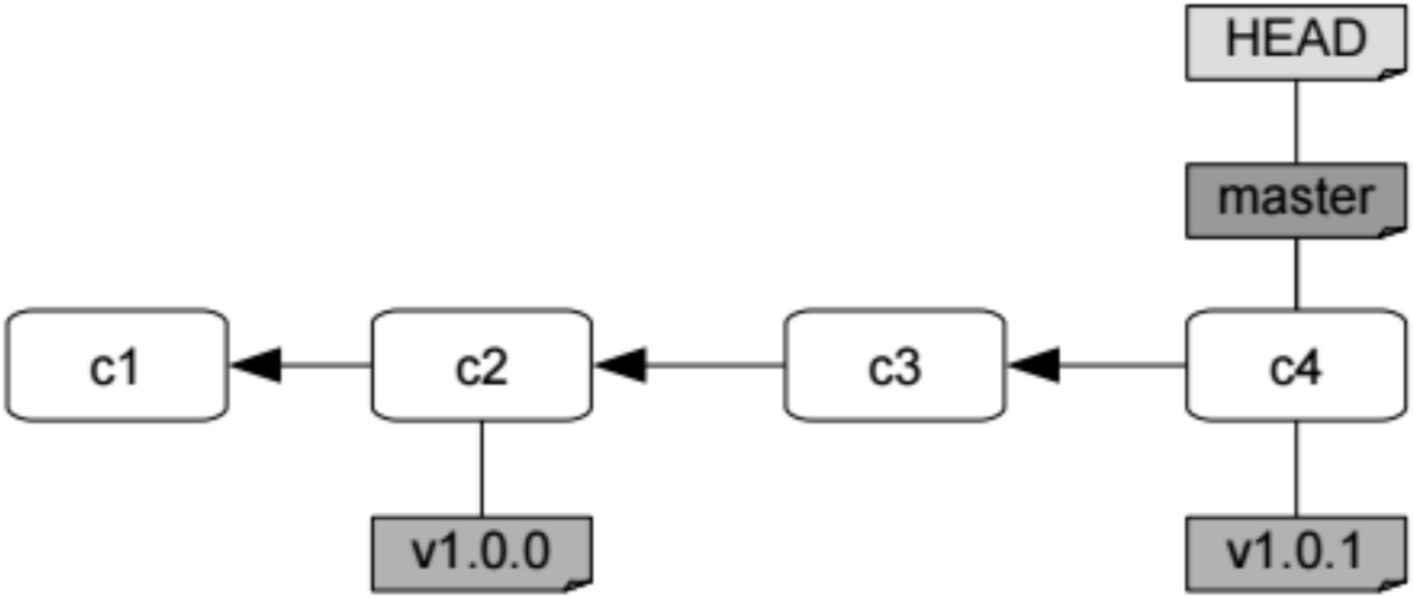
Branch

= Series of versions

MASTER



Branch



Go back to previous version: git checkout v1.0.0

Current version: git checkout master

OAK 3 ACADEMY
TECHNICAL END-USER PERSONAL SKILLS

Branch

= Independent development

git branch: list of branches in repo

git branch myBranch: create

git branch –d myBranch: delete

git branch –m betterName: renaming the current branch

New Branch

Create ‘backup’ branch:

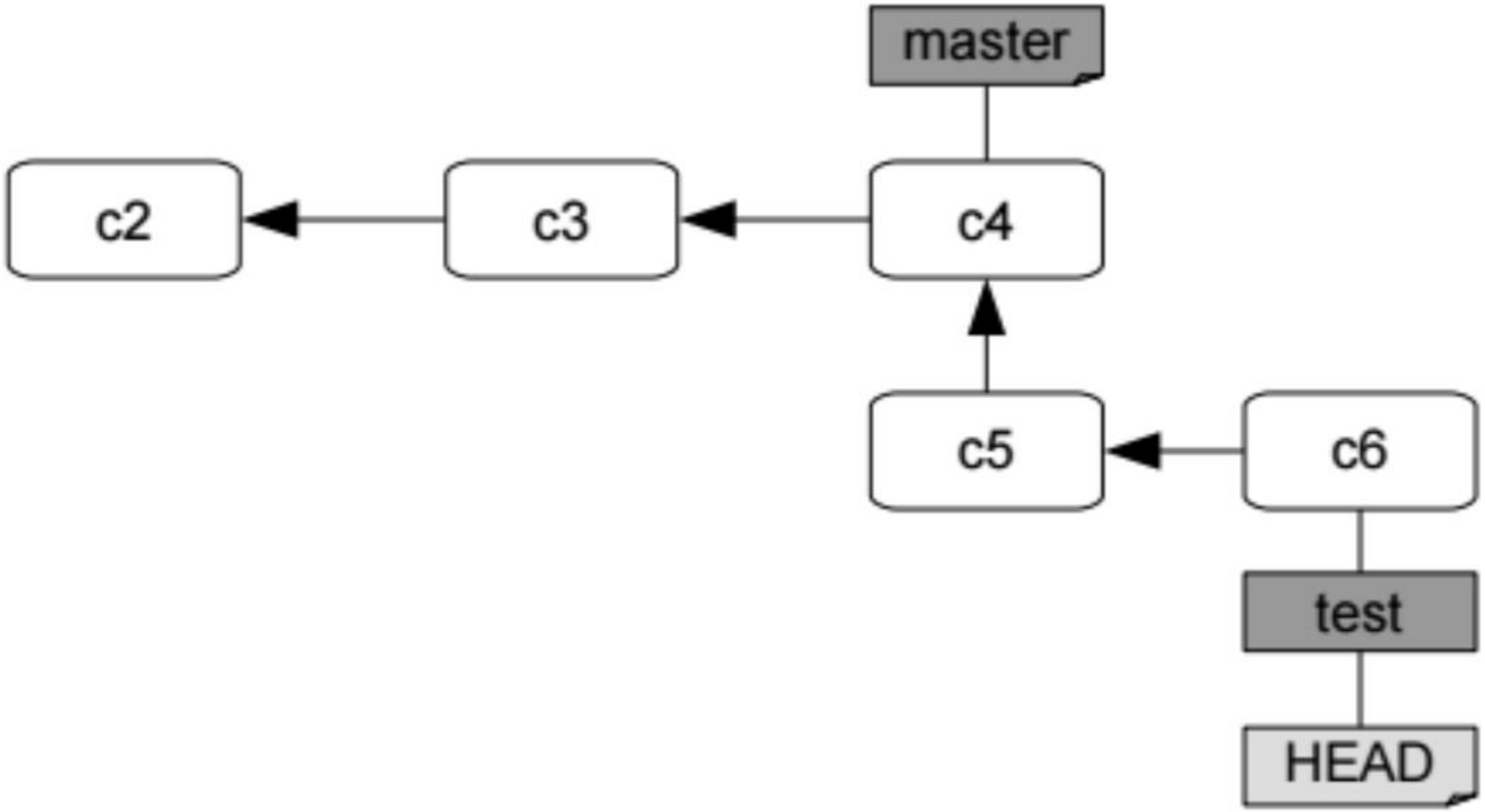
```
vangike@HTPXPF2 MINGW64 ~/Documents/Cursussen OAK3/Git/MyFirstRepository (master)
$ git branch backup
```

Work on this branch

```
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



Exercise

- 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"`

Merge Branch

Merge branch with 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

Notification Fast-forward: No changes in master branch.

= Simple 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

If changes have been made to the same file on the master and another branch at the same time.

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

Content working directory

Content test branch

OAK 3 ACADEMY
TECHNICAL END-USER PERSONAL SKILLS

Merge Conflicts

Remove signs. Commit file.

→ Conflict ok!

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 the past

Branch from a version in history.

EG: bugfix V1.0.0

→git checkout V1.0.0

→git branch bugfix

→Faster: git checkout –b bugfix V1.0.0

OAK 3 ACADEMY
TECHNICAL END-USER PERSONAL SKILLS

Exercise

- 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

→Save changes for later use. Local copy will be restored. For example: solving a fast bug fix (current work can not be lost)

git add stash.txt
git stash
git stash list → show all stashed files
git stash apply → place file back
git stash drop → delete a version

GIT

Synchronise with remote server

Remote Repository

Share code with others

Other team members can then make a local copy of it

Others can make a contribution

Repo Managment Service

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

→ Make github account

OAK 3 ACADEMY
TECHNICAL END-USER PERSONAL SKILLS

Remote

git remote

→Make, see and delete connections to other 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
```

Create repo on git

```
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)
```

List of remote connections

OAK 3 ACADEMY
TECHNICAL END-USER PERSONAL SKILLS

Push

Code the same locally with the remote repo.

git push origin master: place code locally on remote. Straight on the master branch.

git push --set --upstream origin master:
origin master is default repo

Exercise

- Now create a remote repository yourself on github and place your current project on it.
- Make this repo standard for this project.
- Change some files and place these changes here too.
- Request history using git log -- decorate -- graph.

Copy repo

git clone <directory>

E.G.:

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

Fetch

git fetch

Import commits from remote to local repo.

No effect on local work: only view the changes.

Pull

git pull

Code the same locally with the remote repo.

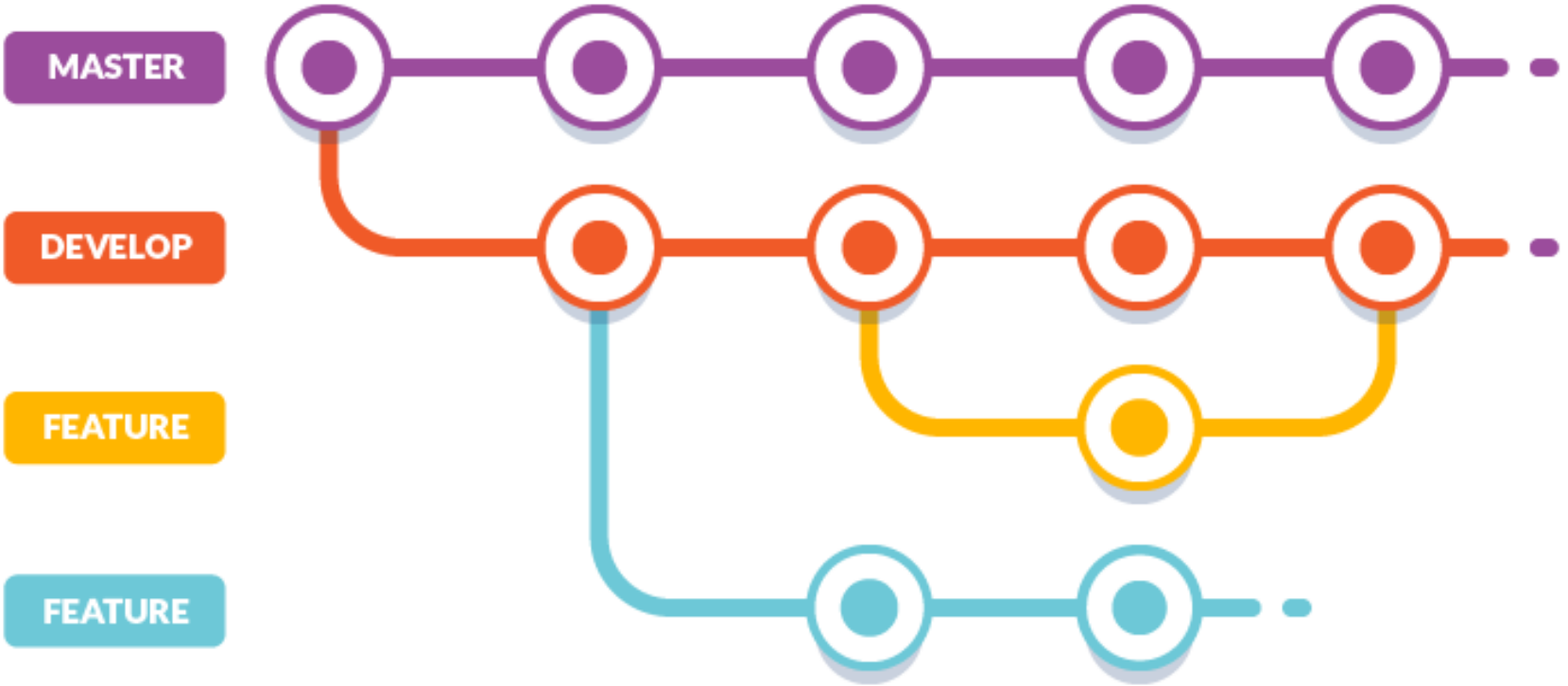
Solve any merge conflicts

Git workflow

- Init (clone) repo
- Change files
- Stage
- Review
- Commit with message
- Push

Git workflow

Scheme:



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

→Undo a commit with a new commit

→Project history is tracked

EXTRA: Clean

git clean

→Delete new files

→Cannot be undone

git clean –n: show the files

git clean –f: delete the files