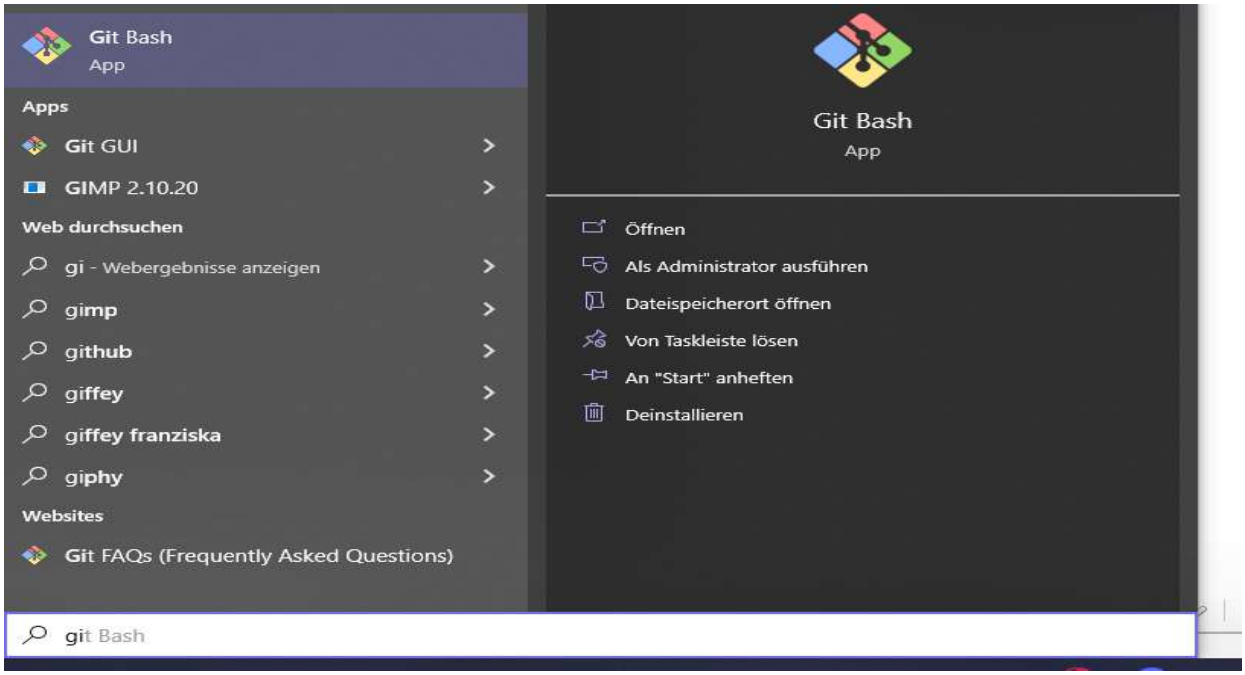
Git Cheat Sheet

Setting up Git:

Um mit Git arbeiten zu können müsst ihr folgendes machen:

Windows:

- Erst einmal Git installieren, dabei solltet ihr am besten auch die Git-Bash installieren (ist basically ein Bash Terminal Emulator):
  - https://git-scm.com/download/win
- Anschließend solltet ihr die Git Bash installiert finden




- nach dem öffnen müsst ihr hier euren Namen und eure E-mail angeben:

```
$ git config --global user.name "John Doe"
$ git config --global user.email johndoe@example.com
```

- ob das geklappt hat, könnt ihr überprüfen, indem ihr git config --list eingebt
- Sollte dann so in etwa aussehen:

```
Admin@DESKTOP-BSU24U8 MINGW64 ~
$ git config --list
diff.astextplain.textconv=astextplain
filter.lfs.clean=git-lfs clean -- %f
filter.lfs.smudge=git-lfs smudge -- %f
filter.lfs.process=git-lfs filter-process
filter.lfs.required=true
http.sslbackend=openssl
http.sslcainfo=C:/Program Files/Git/mingw64/ssl/certs/ca-bundle.crt
core.autocrlf=true
core.fscache=true
core.symlinks=false
pull.rebase=false
credential.helper=manager-core
credential.https://dev.azure.com.usehttppath=true
init.defaultbranch=master
user.name=Hannes Albert
user.email=hannes10042004@icloud.com
safe.directory=F:/FOP-2223-H08-Student
core.autocrlf=true
core.editor=vim

Admin@DESKTOP-BSU24U8 MINGW64 ~
$
```

- Wichtig ist hierbei nur, dass unter user.name euer Name und unter user.email eure email steht

MacOs / Linux:

- Git über das integrierte Terminal installieren
- ob die Installation geklappt hat mit git –version überprüfen
- anschließen können alle Schritte, welche ich oben für Windows in der GitBash erklärt habe auch im integrierten Terminal vollzogen werden

Mit Git arbeiten

Um loslegen zu können müsst ihr erst einmal müsst ihr erst einmal einen GitHub Account erstellen und diesen mir schicken, damit ich euch zu dem Repository hinzufügen kann.
Sobald ich euch hinzugefügt habe und ihr meine Anfrage angenommen habt (wenn ich euch eine Anfrage schicken sollte die Glocke oben rechts auf der GitHub Homepage einen blauen Punkt an der Seite haben), sollte das Projekt links unter „Top Repositories“ auftauchen.
Mit den Link des Repos, könnt ihr diesen in IntelliJ klonen.

General workflow with Git:
Wenn ihr seht, dass Changes zum Repo gemacht wurden, weil z.B. jemand in die Dc Gruppe geschrieben habt, öffnet ihr das in IntelliJ integrierte Terminal (shortcut: Alt + F12).
Dort tippt ihr dann ein: **git pull**
Das Ergebnis sollte dann in etwa so aussehen:

```
PS C:\Users\Admin\IdeaProjects\FOP2223-Projekt> git pull
remote: Enumerating objects: 1, done.
remote: Counting objects: 100% (1/1), done.
remote: Total 1 (delta 0), reused 0 (delta 0), pack-reused 0
Unpacking objects: 100% (1/1), 643 bytes | 22.00 KiB/s, done.
From https://github.com/w8ste/FOP2223-Projekt
   07f4667..059d7ec  main       -> origin/main
Updating 07f4667..059d7ec
Fast-forward
 src/main/java/projekt/Main.java | 2 ++
 1 file changed, 2 insertions(+)
PS C:\Users\Admin\IdeaProjects\FOP2223-Projekt>
```

Um später das Einführen von Bugs zu vermeiden, wäre es sinnvoll mit Branches zu arbeiten.
Im Folgenden Beispiel zeige ich kurz wie ich einen Branch namens test erstelle und diesen auf GitHub pushe.

- Einen neuen Branch erstellen wir wie folgt: **git checkout -b branchname**
  - In unserem Fall würde dies **git checkout -b test** bedeuten

```
PS C:\Users\Admin\IdeaProjects\FOP2223-Projekt> git checkout -b test
Switched to a new branch 'test'
```

  - checkout wird hierbei verwendet um zwischen branches zu wechseln (git checkout main würde uns somit wieder auf den main branch zurückbringen). -b sagt Git einfach, dass es einen neuen Branch erstellen soll.

- Sollte man eine Projekt Aufgabe bearbeiten wäre es natürlich sinnvoller den Branch in etwa z.B. H2 zu nennen.
- In unserem test branch bearbeiten wir nun aber nur die Main Methode, also lass uns einfach mal etwas am code ändern

```
public static void main(String[] args) throws ClassNotFoundException, InstantiationException, IllegalAccessException {
    System.out.println("KW für immer !");    //cheat sheet purposed

    startWithGUI();
    //startWithoutGUI(); //can bew used instead to run a simulation without a gui
```

- Wenn wir jetzt sagen wir sind mit unseren Changes zufrieden und wollen sie mit den anderen Gruppenmitgliedern teilen begeben wir uns wieder ins Terminal
- Wenn wir git status ausführen sehen wir eine Liste aller von uns veränderter Files

```
PS C:\Users\Admin\IdeaProjects\FOP2223-Projekt> git status
On branch test
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   src/main/java/projekt/Main.java

no changes added to commit (use "git add" and/or "git commit -a")
```

- Nun müssen wir unsere veränderten Files erst einmal „stagen“
  - was dies genau bedeutet ist für uns erst einmal nicht von belang
- Dies nun wir, indem wir **git add .** eintippen

```
PS C:\Users\Admin\IdeaProjects\FOP2223-Projekt> git add .
```

- Anschließend können wir endlich „commiten“ und unsere Changes final (zumindest für den Anfang, theoretisch könnte man diese selbstverständlich Rückgängig machen) machen
- Zum „commiten“ schreiben wir: git commit -m „Irgendeine Nachricht“
  - Also

```
PS C:\Users\Admin\IdeaProjects\FOP2223-Projekt> git commit -m "Kw rules"
[test 5c83da1] Kw rules
 1 file changed, 3 insertions(+), 3 deletions(-)
```

Solltet ihr -m „irgendeine Nachricht“ vergessen haben, sollte euer Terminal auf einmal so aussehen, dies liegt daran, dass ihr euch nun in vi befindet. Wie ihr vi benutzt möchte ich nicht erklären deswegen solltet ihr einfach tippen: :q dies sollte ich wieder ins Terminal zurückbringen und ihr könnt einfach nochmal neu commiten.

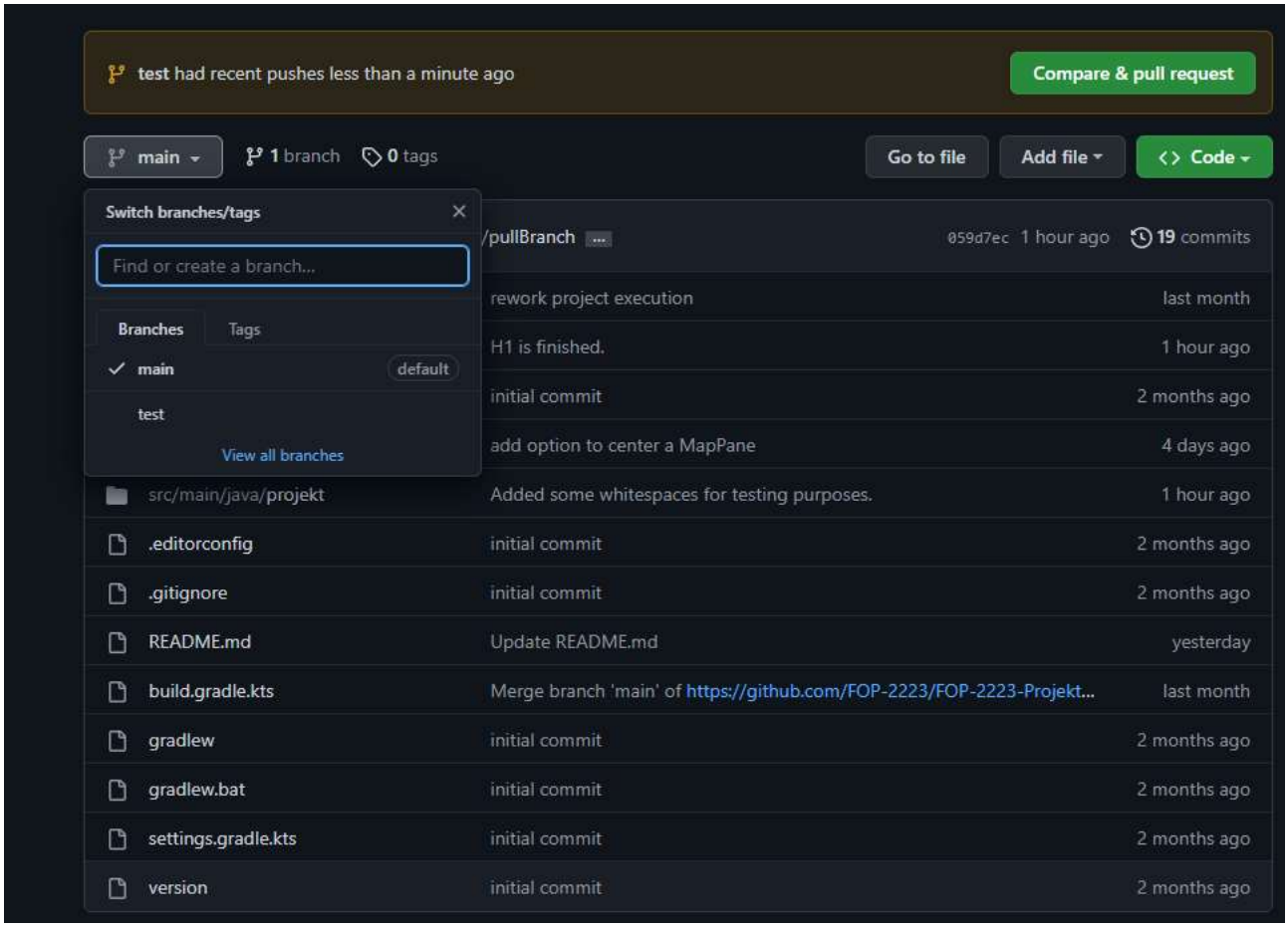
```
# Please enter the commit message for your changes. Lines starting
# with '#' will be ignored, and an empty message aborts the commit.
#
# On branch test
# Changes to be committed:
#       modified:   src/main/java/projekt/Main.java
#
~
~
~
~
.git/COMMIT_EDITMSG [unix] (12:36 14/02/2023)
"~/IdeaProjects/FOP2223-Projekt/.git/COMMIT_EDITMSG" [unix] 8L, 230B
```

Um die Changes nun nach GitHub zu pushen, tippen wir ins Terminal ein:

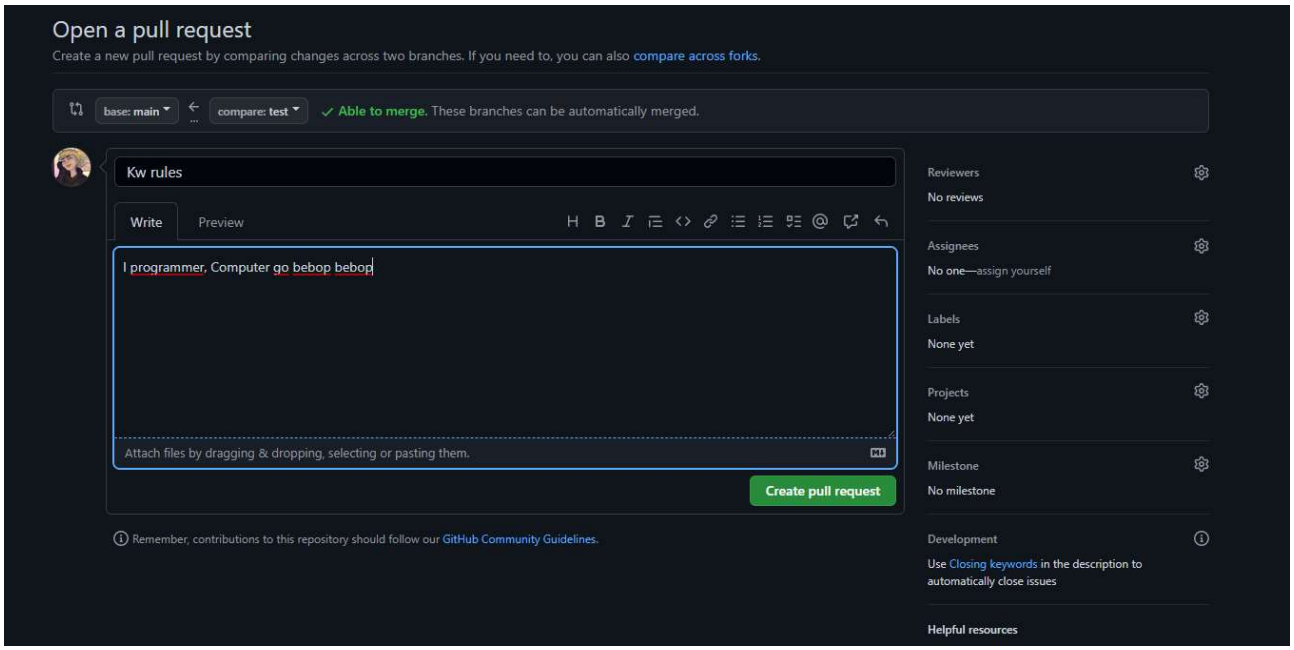
- **git push origin <branchname>**
- **Achtung: beim ersten mal pushen git push -u origin <branchname>**
  - in unsrem Fall also: git push origin test

```
On branch main
Your branch is up to date with 'origin/main'.
Enumerating objects: 13, done.
Counting objects: 100% (13/13), done.
Delta compression using up to 4 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (7/7), 553 bytes | 553.00 KiB/s, done.
Total 7 (delta 2), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (2/2), completed with 2 local objects
remote:
remote: Create a pull request for 'test' on GitHub by visiting:
remote:      https://github.com/w8ste/FOP2223-Projekt/pull/new/test
remote:
To https://github.com/w8ste/FOP2223-Projekt
 * [new branch]      test -> test
PS C:\Users\Admin\IdeaProjects\FOP2223-Projekt>
```

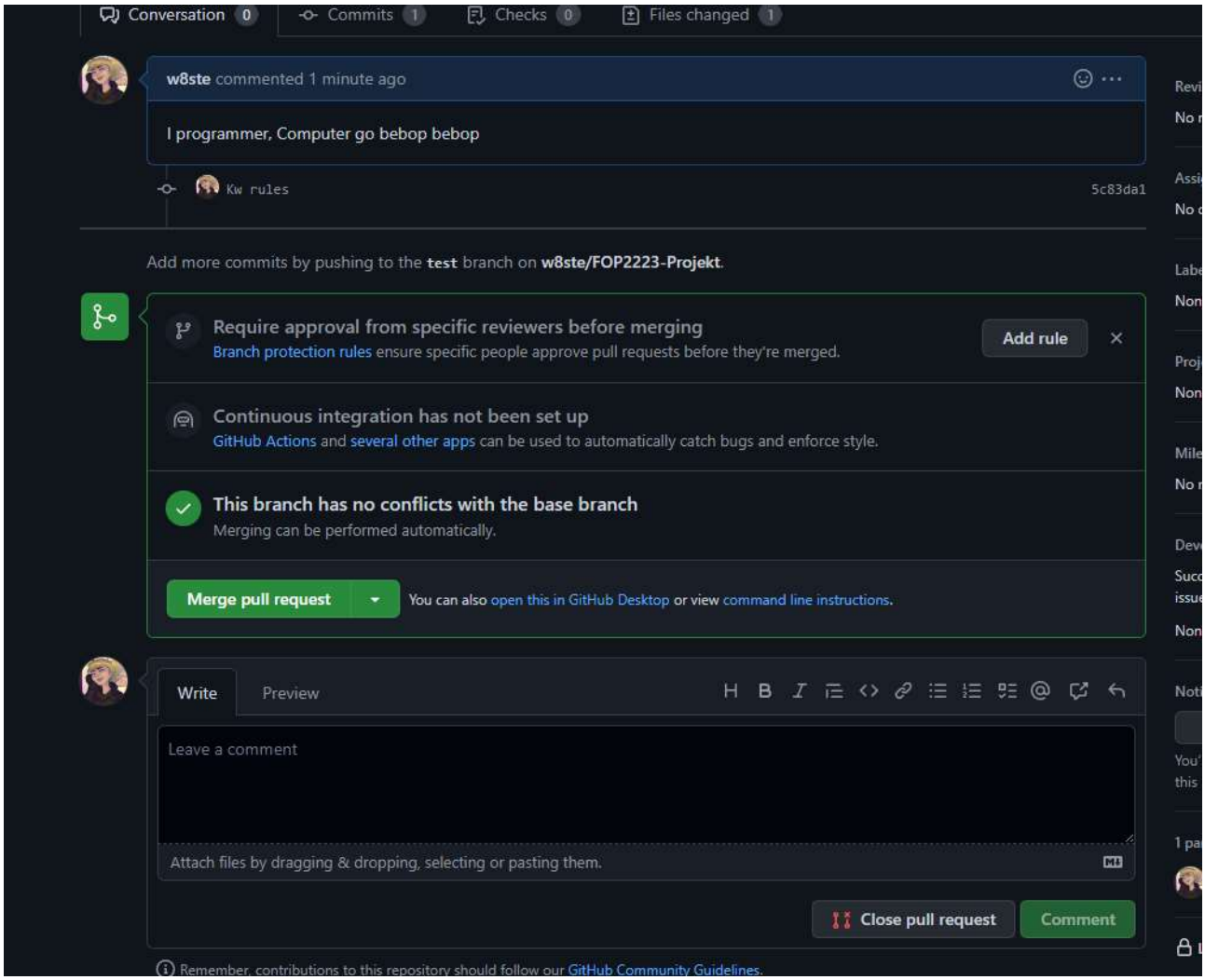
Wenn wir nun auf GitHub uns das Repository anschauen sehen wir, dass der Branch test gepushed wurde:



Wir clicken nun auf das Feld: „Compare & Pull Request“ und erstellen einen Kommentar, welcher unsere changes beschreibt.




Mit „Create Pull request“ erstellt ihr diese nun.
Danach sollte euer Fenster so aussehen:



Am besten wäre es, wenn ihr eure Pull request nicht selber merged, sondern die anderen Mitglieder kurz Benachrichtigt, damit zumindest eine andere Person sich mal eure Changes anschaut.

Anschließend könnt ihr mit **git checkout main** wieder zurück auf den main-branch kehren. Wenn ihr seht, dass eure Changes im Git Repo gemerged wurden, könnt ihr selbstverständlich den alten Branch löschen.
Davor müsst ihr diesen, allerdings erst einmal in euren lokalen main branch mergen.
Diest macht ihr wie folgt (wenn ihr wieder auf main seid): **git merge <branchname>**
Löschen geht dann wie folgt: **git branch -d <branch name>**
In unserem Fall wäre dies also

```
PS C:\Users\Admin\IdeaProjects\FOP2223-Projekt> git merge test
Updating 059d7ec..5c83da1
Fast-forward
 src/main/java/projekt/Main.java | 6 +++---
 1 file changed, 3 insertions(+), 3 deletions(-)
PS C:\Users\Admin\IdeaProjects\FOP2223-Projekt> git branch -d test
Deleted branch test (was 5c83da1).
PS C:\Users\Admin\IdeaProjects\FOP2223-Projekt>
```