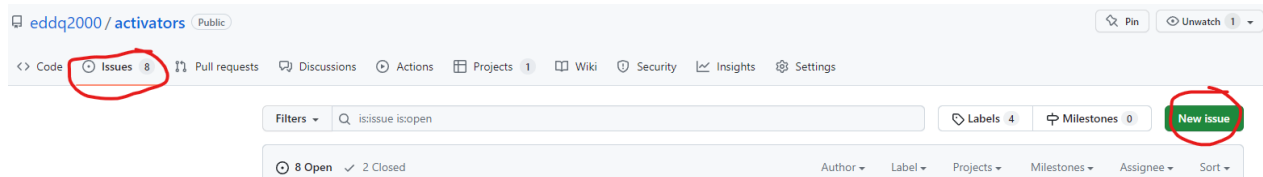


Workflow

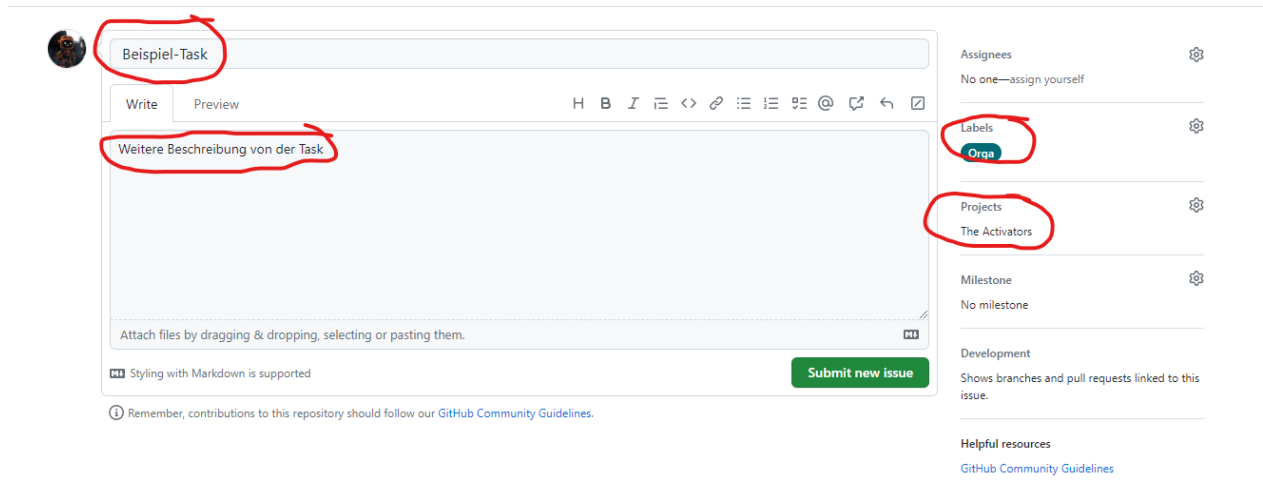
(1. Ticket erstellen)

Wenn es eine Aufgabe gibt wird immer als erstes ein Ticket erstellt. Im Repository kann man unter „Issues“ ein neues Ticket erstellen:

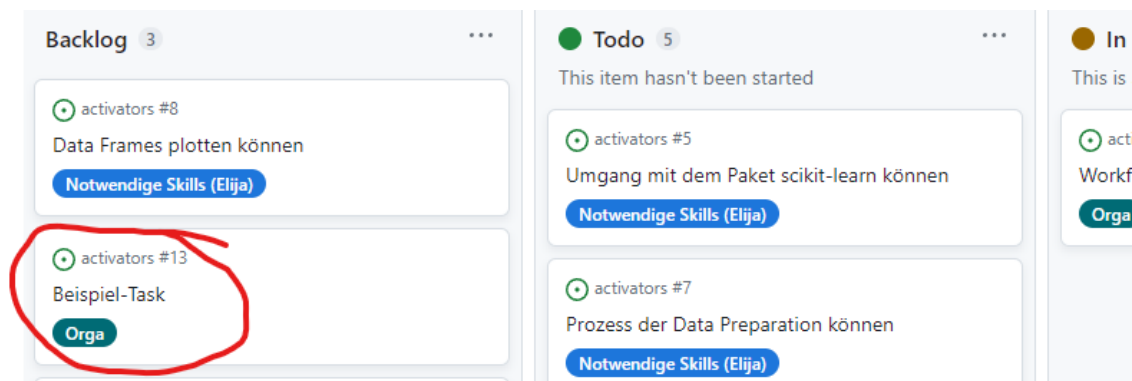


Beim Erstellen folgende Angaben machen:

- Titel geben
- Evtl. Mit Beschreibung erweitern
- Das Ticket einem Label zuordnen
- Das Ticket dem Projekt „Activators“ zuordnen

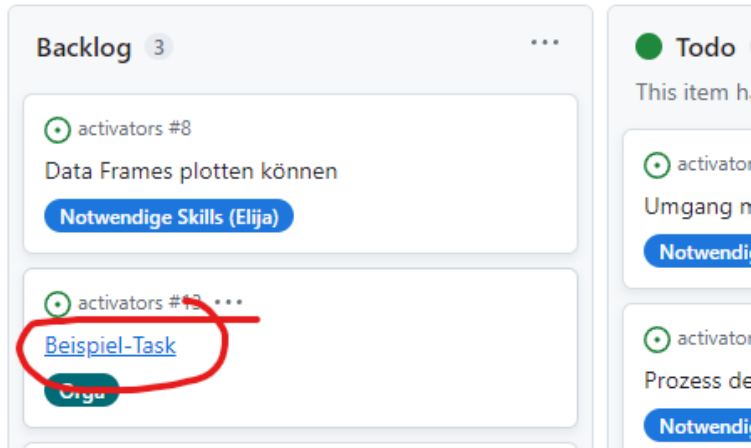


Sobald ihr das Ticket erstellt habt, sollte es auf dem projekt-Board zu sehen sein:

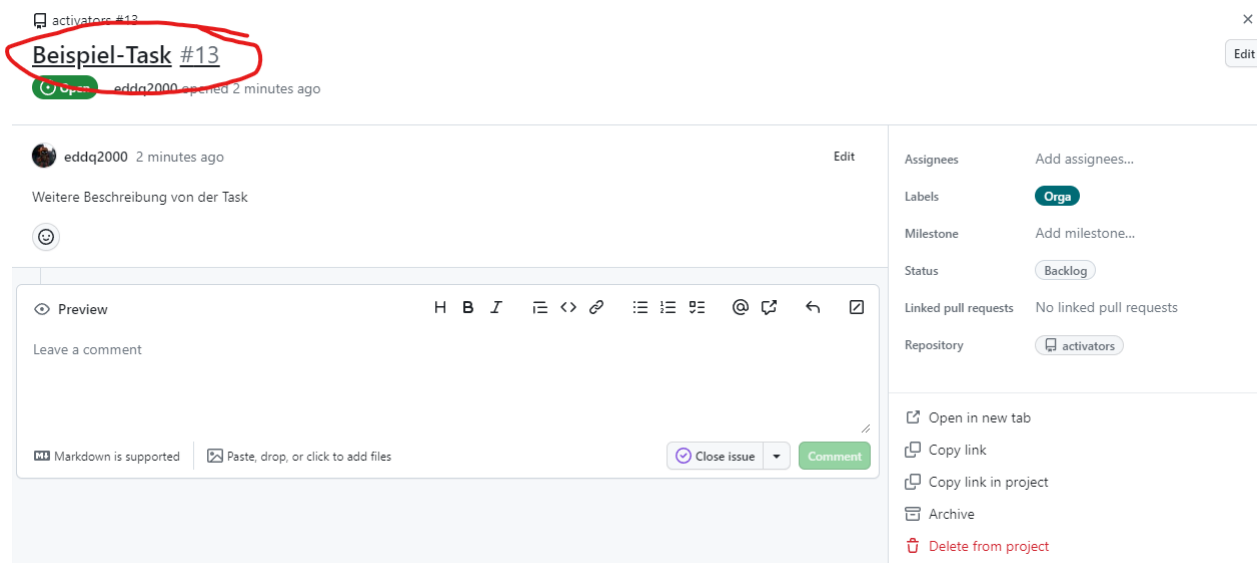


2. Ticket zuweisen

Um euch einem Ticket zuzuweisen geht ihr auf das Projekt-Board und klickt auf das Ticket, welches ihr bearbeiten wollt:



In dieser Ansicht klickt ihr dann nochmal auf den Titel des Tickets:



Dann sollte sich folgende Ansicht öffnen:

Beispiel-Task #13

Edit New issue

Open eddq2000 opened this issue 3 minutes ago · 0 comments

eddq2000 commented 3 minutes ago

Owner

Weitere Beschreibung von der Task

eddq2000 added the **Orga** label 3 minutes ago

eddq2000 added this to **The Activators** 3 minutes ago

eddq2000 moved this to Backlog in **The Activators** 3 minutes ago

Assignees

No one—assign yourself

Labels

Orga

Projects

The Activators

Status: Backlog

Milestone

No milestone

Development

Create a branch for this issue or link a pull request.

Jetzt könnt ihr euch rechts oben zu dem Ticket assignen, was bedeutet, dass jeder sehen kann, dass ihr diese Aufgabe bearbeitet:

eddq2000 commented 5 minutes ago

Owner

Weitere Beschreibung von der Task

eddq2000 added the **Orga** label 5 minutes ago

eddq2000 added this to **The Activators** 4 minutes ago

eddq2000 moved this to Backlog in **The Activators** 4 minutes ago

eddq2000 self-assigned this now

Assignees

eddq2000

Labels

Orga

Projects

The Activators

Status: Backlog

Milestone

No milestone

Development

Create a branch for this issue or link a pull request.

activators #13

Beispiel-Task

Orga

3. Branch erstellen und lokal bearbeiten

Um einen Branch zu erstellen clickt ihr in eurem Ticket einfach auf „Create a branch...“:

The screenshot shows a GitHub issue page. On the left, a list of activity: 'eddq2000 commented 1 hour ago' (with a comment box containing 'Weitere Beschreibung von der Task'), 'eddq2000 added the Orga label 1 hour ago', 'eddq2000 added this to The Activators 1 hour ago', 'eddq2000 moved this to Backlog in The Activators 1 hour ago', and 'eddq2000 self-assigned this 51 minutes ago'. On the right, the 'Assignees' section lists 'eddq2000'. The 'Labels' section shows 'Orga'. The 'Projects' section shows 'The Activators' with a status of 'Backlog'. The 'Milestone' section shows 'No milestone'. The 'Development' section has a link 'Create a branch for this issue or link a pull request.', which is circled in red.

Drückt dann einfach auf „Create Branch“:

The dialog box is titled 'Create a branch for this issue'. It has a 'Branch name' field with '13-beispiel-task' and a copy icon. Below it is the 'Repository destination' dropdown showing 'eddq2000/activators' with a 'Change branch source' link. Under 'What's next?', there are three radio buttons: 'Open in codespace', 'Checkout locally' (which is selected), and 'Open branch with GitHub Desktop'. At the bottom, there is a 'Beta' badge, a 'Share feedback' link, and a green 'Create branch' button, which is circled in red.

Jetzt öffnet ihr in eurem lokalen Repository die Git Bash-Konsole:

The screenshot shows a Git Bash terminal window. The title bar reads 'MINGW64: c:/Users/elija/Documents/Uni/Winf/Semester 5 (SoSe23)/Machine Learning for Business Advanced Concepts/Projekt/activators'. The terminal content shows the user 'elija@DESKTOP-I8L1UPL' in the directory '~/Documents/Uni/Winf/Semester 5 (SoSe23)/Machine Learning for Business Advanced Concepts/Projekt/activators (dev)'. The prompt is '\$'.

Als erstes wollt ihr den aktuellen Stand vom Repository auf Github (wo auch euer neuer branch ist) auf euer lokales Repository übertragen. Dazu gebt ihr in den Befehl „*git pull*“ ein:

```
elija@DESKTOP-I8L1UPL MINGW64 ~/Documents/Uni/Winf/Semester 5 (So
$ git pull
From https://github.com/eddq2000/activators
* [new branch]      13-beispiel-task -> origin/13-beispiel-task
Updating ac20268..7f313f3
Fast-forward
 Main Project/platzhalter.txt | 0
 Private Area/Alex/platzhalter.txt | 0
 Private Area/Aryan/platzhalter.txt | 0
 .../Python Dataframes/002-dataframe-basics.ipynb | 246 ++++++
+++++++
 ...gelaufen-2023-05-02_15-28-27-Accelerometer.json | 1 +
 .../_pycache__/helperFunctions.cpython-310.pyc | Bin 0 -> 70
6 bytes
 .../Elija/Python Dataframes/helperFunctions.py | 11 +
 .../createJsonFilesBySensorType.py | 2 +-
 Private Area/Frederik/platzhalter.txt | 0
 Secondary Projects/platzhalter.txt | 0
10 files changed, 259 insertions(+), 1 deletion(-)
create mode 100644 Main Project/platzhalter.txt
create mode 100644 Private Area/Alex/platzhalter.txt
create mode 100644 Private Area/Aryan/platzhalter.txt
create mode 100644 Private Area/Elija/Python Dataframes/002-data
frame-basics.ipynb
create mode 100644 Private Area/Elija/Python Dataframes/005_Hin_
und_her_gelaufen-2023-05-02_15-28-27-Accelerometer.json
create mode 100644 Private Area/Elija/Python Dataframes/_pycach
e__/helperFunctions.cpython-310.pyc
create mode 100644 Private Area/Elija/Python Dataframes/helperFu
nctions.py
rename {Python Scripts (Exploration) => Private Area/Elija/Pytho
n Scripts (Exploration)}/createJsonFilesBySensorType.py (88%)
create mode 100644 Private Area/Frederik/platzhalter.txt
create mode 100644 Secondary Projects/platzhalter.txt

elija@DESKTOP-I8L1UPL MINGW64 ~/Documents/Uni/Winf/Semester 5 (So
Se23)/Machine Learning for Business Advanced Concepts/Projekt/act
ivators (dev)
$
```

Jetzt wollt ihr auf den Branch wechseln den ihr erstellt habt. Dafür gebt ihr folgenden Befehl ein:

„*git checkout <branch-name>*“:

```
elija@DESKTOP-I8L1UPL MINGW64 ~/Documents/Uni/Winf/Semester 5 (So
Se23)/Machine Learning for Business Advanced Concepts/Projekt/act
ivators (dev)
$ git checkout 13-beispiel-task
Switched to a new branch '13-beispiel-task'
branch '13-beispiel-task' set up to track 'origin/13-beispiel-tas
k'.

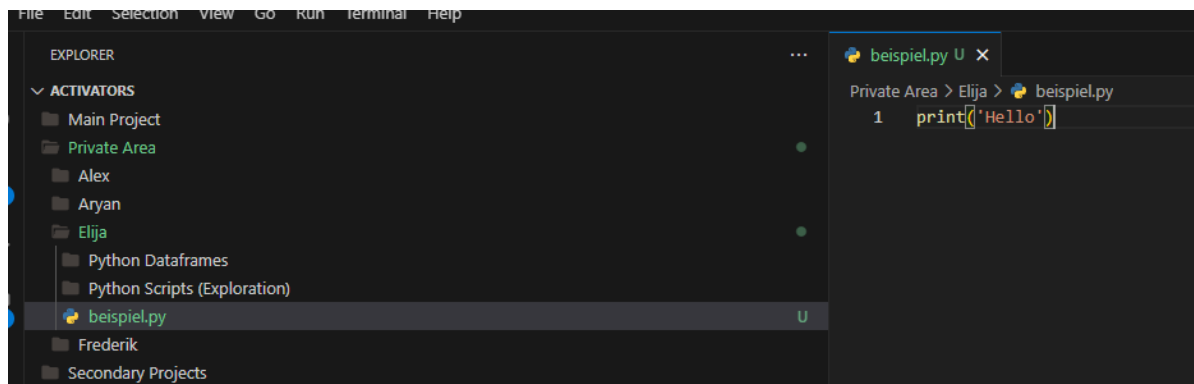
elija@DESKTOP-I8L1UPL MINGW64 ~/Documents/Uni/Winf/Semester 5 (So
Se23)/Machine Learning for Business Advanced Concepts/Projekt/act
ivators (13-beispiel-task)
$
```

Es sollte nun in der bash Konsole in Klammern der richtige branch stehen.

- ⇒ Das bedeutet, dass sich dein lokales Repository jetzt auf deinem eigenen Branch befindet und alles was du hier im Repository änderst nur auf diesem Branch passiert.

4. Arbeiten auf deinem Branch

Befindest du dich auf dem richtigen Branch, kannst du anfangen Änderungen zu machen. Z.B. fügst du eine neue Datei hinzu (oder änderst was in anderen Dateien im Repository):



Gibst du in die Bash-Konsole „*git status*“ eingibst, kannst du genau sehen, was für Änderungen du alles gemacht hast:

```
eliya@DESKTOP-I8L1UPL MINGW64 ~/Documents/Uni/Winf/Semester 5 (SoSe23)/Machine Learning for Business Advanced Concepts/Projekt/activators (13-beispiel-task)
$ git status
On branch 13-beispiel-task
Your branch is up to date with 'origin/13-beispiel-task'.

Untracked files:
  (use "git add <file>..." to include in what will be committed)
  Private Area/Elija/beispiel.py

nothing added to commit but untracked files present (use "git add" to track)
```

(Hier steht zum Beispiel, dass ich eine ganz neue Datei hinzugefügt habe)

4.1 Stagen und Committen

Sobald du deine Arbeit vom Ticket erledigt hast oder einen logischen Abschnitt fertig hast, kannst du deine Änderungen in einem Commit zusammenfassen und so **speichern**.

Dafür musst du zuerst deine Änderungen stagen, was so viel bedeutet wie, du machst deine Änderungen dafür bereit, um sie zu committen. Das machst du mit „*git add .*“ (Den Punkt nicht vergessen!). Das sieht dann so aus:

```
eliya@DESKTOP-I8L1UPL MINGW64 ~/Documents/Uni/Winf/Semester 5 (SoSe23)/Machine Learning for Business Advanced Concepts/Projekt/activators (13-beispiel-task)
$ git add .

eliya@DESKTOP-I8L1UPL MINGW64 ~/Documents/Uni/Winf/Semester 5 (SoSe23)/Machine Learning for Business Advanced Concepts/Projekt/activators (13-beispiel-task)
$
```

Mit „git status“ siehst du jetzt folgendes:

```
elija@DESKTOP-I8L1UPL MINGW64 ~/Documents/Uni/Winf/Semester 5 (SoSe23)/Machine Learning for Business/Projekt/activators (13-beispiel-task)
$ git status
On branch 13-beispiel-task
Your branch is up to date with 'origin/13-beispiel-task'.

Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
    new file:   Private Area/Elia/beispiel.py
```

Jetzt kannst du deine Änderungen comitten: `git commit -m "<Eine Commit Nachricht>"`

```
elija@DESKTOP-I8L1UPL MINGW64 ~/Documents/Uni/Winf/Semester 5 (SoSe23)/Machine Learning for Business/Projekt/activators (13-beispiel-task)
$ git commit -m "Ein Beispiel-File erstellt"
[13-beispiel-task 648c5d9] Ein Beispiel-File erstellt
 1 file changed, 1 insertion(+)
 create mode 100644 Private Area/Elia/beispiel.py
```

Gehst du nun nochmal auf „git status“, siehst du folgendes:

```
elija@DESKTOP-I8L1UPL MINGW64 ~/Documents/Uni/Winf/Semester 5 (SoSe23)/Machine Learning for Business/Projekt/activators (13-beispiel-task)
$ git status
On branch 13-beispiel-task
Your branch is ahead of 'origin/13-beispiel-task' by 1 commit.
  (use "git push" to publish your local commits)

nothing to commit, working tree clean
```

5. Deine Commits auf Github pushen

Um deine Commits, welche bisher nur lokal auf deinem Rechner sind, auf Github zu pushen gibst du ganz einfach „git push“ ein:

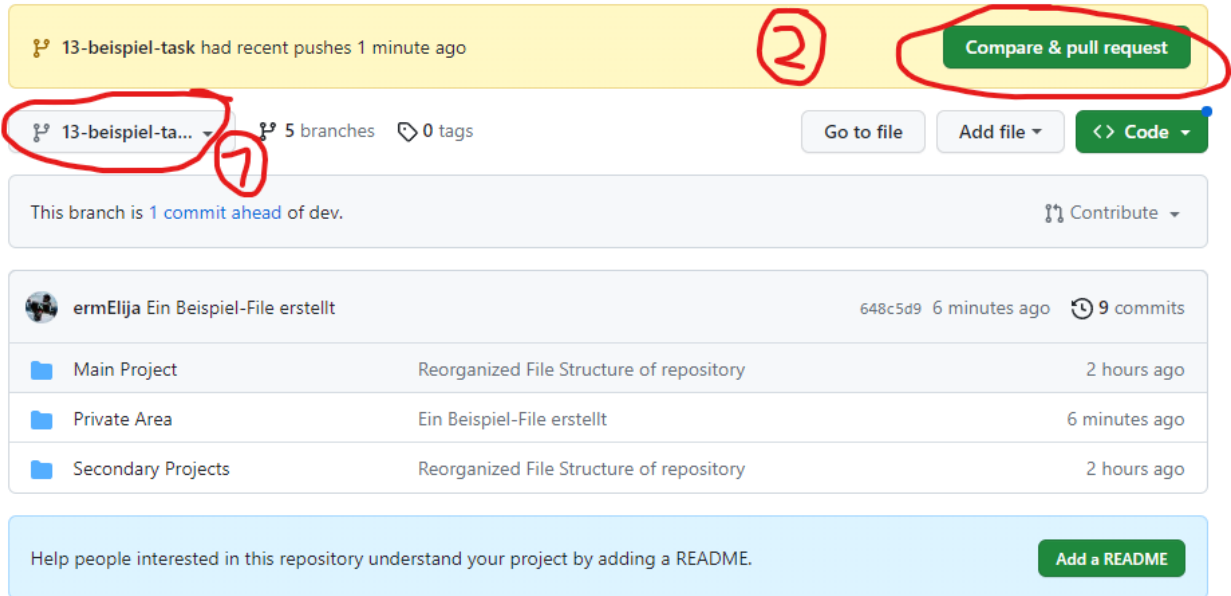
```
elija@DESKTOP-I8L1UPL MINGW64 ~/Documents/Uni/Winf/Semester 5 (SoSe23)/Machine Learning for Business/Projekt/activators (13-beispiel-task)
$ git push
Enumerating objects: 8, done.
Counting objects: 100% (8/8), done.
Delta compression using up to 24 threads
Compressing objects: 100% (4/4), done.
Writing objects: 100% (5/5), 462 bytes | 462.00 KiB/s, done.
Total 5 (delta 2), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (2/2), completed with 2 local objects.
To https://github.com/eddq2000/activators.git
  7f313f3..648c5d9 13-beispiel-task -> 13-beispiel-task
```

Jetzt sind deine Commits auf Github unter deinem Branch gespeichert.

6. Einen Pull-Request stellen

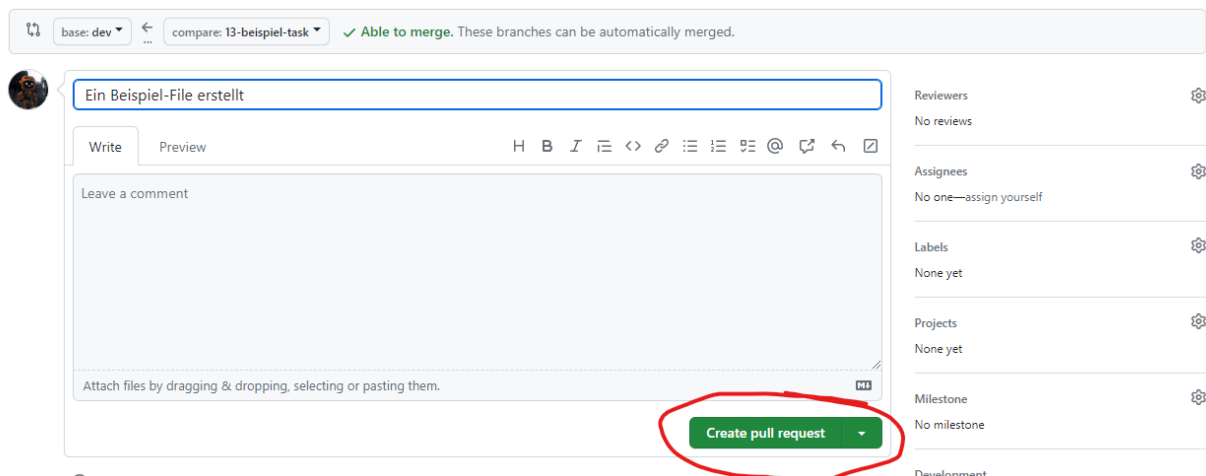
Bist du fertig mit der Arbeit für dein Ticket, dann kannst du einen Pull-Request stellen.

Bisher ist deine Arbeit noch auf dem speziellen Branch, den du für das Ticket erstellt hast. Wir wollen die Arbeit von jedem in einem Branch „dev“ zusammentragen. Dafür müssen der Branch „dev“ mit deinem Branch zusammengetragen (merged) werden. Dafür stellst du auf Github ganz einfach eine Pull-Request:

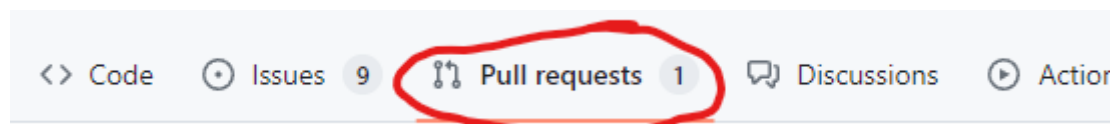


Open a pull request

Create a new pull request by comparing changes across two branches. If you need to, you can also [compare across forks](#).



Jetzt ist ein Pull-Request erstellt:



7. Reviewen und mergen

Bevor wir einen Branch mergen, sollte nochmal jemand anders kurz über deine Commits/Änderungen drüber schauen und sagen, ob das passt, oder ob etwas fehlt, bzw. nicht funktioniert etc.

Wurde dein Ticket von einer anderen Person reviewed, kann dein branch gemercht werden und deine Änderungen sind offiziell Teil unseres main branches („dev“)