

Git Tutorial

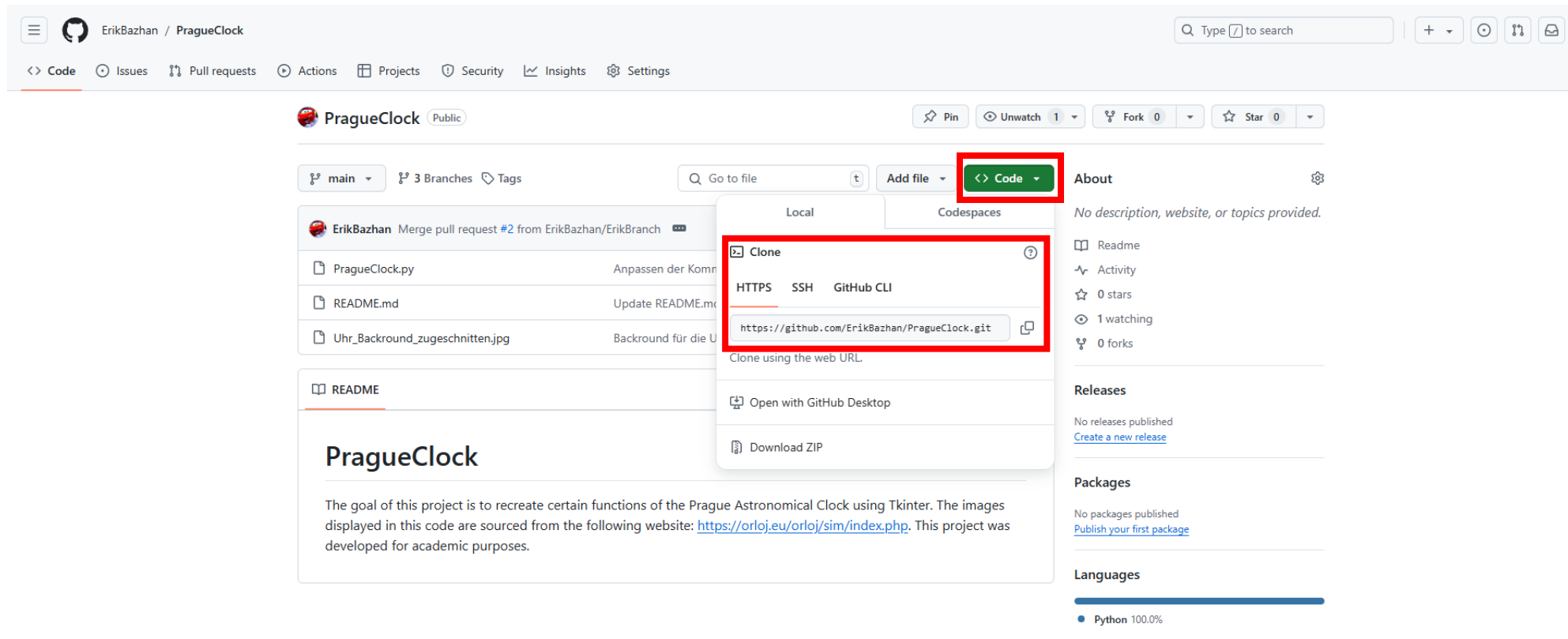
In diesem Tutorial werden die Basics erklärt, um mit Git arbeiten zu können. Bei Fragen kann man stets im Internet nachschauen, oder ChatGPT fragen. Ansonsten einfach bei Kollegen nachfragen. Mit dem folgenden Workflow muss man auch keine Angst haben etwas falsch zu machen.



Basics - Klonen

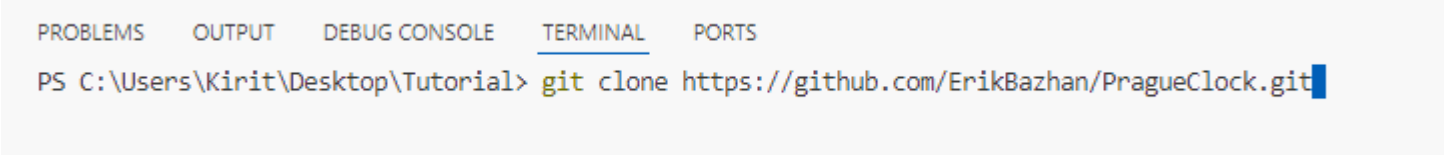
Wir beginnen mit dem Klonen des Repository. Mit dem Klonen wird eine Kopie von dem online Repository lokal erstellt. Hierzu ist folgendes zu tun:

- Aufrufen des Repos: <https://github.com/ErikBazhan/PragueClock>
- Auf Code klicken
- HTTPS Link kopieren



Basics - Klonen

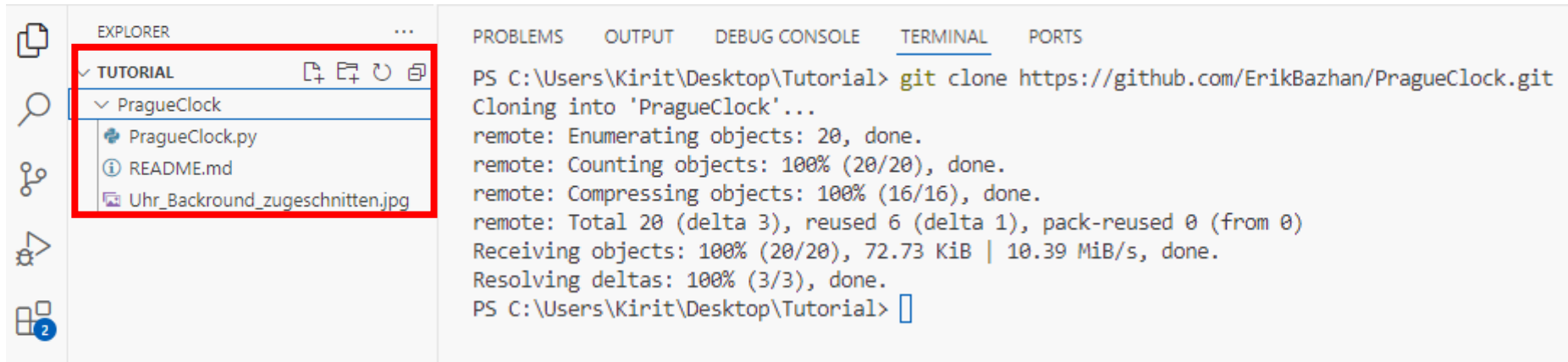
- In VSCode im Terminal eingeben: git clone (kopierter Link mit rechter Maustaste pasten)



PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
PS C:\Users\Kirit\Desktop\Tutorial> git clone https://github.com/ErikBazhan/PragueClock.git
```

Wenn alles erfolgreich war dann sieht man im Explorer den Ordner PragueClock und bekommt alles aus der online Repo lokal in den gewünschten Ordner



EXPLORER

- ✓ TUTORIAL
 - ▼ PragueClock
 - PragueClock.py
 - README.md
 - Uhr_Background_zugeschnitten.jpg

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
PS C:\Users\Kirit\Desktop\Tutorial> git clone https://github.com/ErikBazhan/PragueClock.git
Cloning into 'PragueClock'...
remote: Enumerating objects: 20, done.
remote: Counting objects: 100% (20/20), done.
remote: Compressing objects: 100% (16/16), done.
remote: Total 20 (delta 3), reused 6 (delta 1), pack-reused 0 (from 0)
Receiving objects: 100% (20/20), 72.73 KiB | 10.39 MiB/s, done.
Resolving deltas: 100% (3/3), done.
PS C:\Users\Kirit\Desktop\Tutorial>
```

Basics - Upstream

Jetzt bedarf es an einem Upstream. Ein Upstream verbindet den lokalen Ordner mit dem online Repo. Dies ist nützlich, um mit den folgenden Branches arbeiten zu können.

Um ein Upstream zu erstellen muss folgendes gemacht werden:

- In die Konsole **cd PragueClock** eingeben (damit die Konsole im richtigen Ordner sich befindet)

```
PS C:\Users\Kirit\Desktop\Tutorial> cd PragueClock
PS C:\Users\Kirit\Desktop\Tutorial\PragueClock> |
```

- **git remote -v** eingeben als Kontrolle

```
PS C:\Users\Kirit\Desktop\Tutorial\PragueClock> git remote -v
origin https://github.com/ErikBazhan/PragueClock.git (fetch)
origin https://github.com/ErikBazhan/PragueClock.git (push)
PS C:\Users\Kirit\Desktop\Tutorial\PragueClock> |
```

- **git remote add upstream <https://github.com/ErikBazhan/PragueClock.git>** eingeben und mit **git remote -v** überprüfen

```
PS C:\Users\Kirit\Desktop\Tutorial\PragueClock> git remote add upstream https://github.com/ErikBazhan/PragueClock.git
PS C:\Users\Kirit\Desktop\Tutorial\PragueClock> git remote -v
origin https://github.com/ErikBazhan/PragueClock.git (fetch)
origin https://github.com/ErikBazhan/PragueClock.git (push)
upstream https://github.com/ErikBazhan/PragueClock.git (fetch)
upstream https://github.com/ErikBazhan/PragueClock.git (push)
PS C:\Users\Kirit\Desktop\Tutorial\PragueClock> |
```

Wenn es wie in den Screenshots aussieht, dann wurde erfolgreich ein Upstream erstellt. Jetzt kann man aus dem lokalen IDE etwas in die online Repo hochladen

Basics - Branch

Nun wird eine eigene Branch erstellt. Die Branch ist eine Abzweigung vom main Branch. Innerhalb dieser Abzweigung kann man alles machen, was man will, ohne den main Branch zu ändern. Das heißt man kann neue Sachen implementieren und ausprobieren, ohne den ursprünglichen Code im online Repo zu verändern. Somit bleibt alles organisiert und der main Branch kann als Backup benutzt werden, falls ein Backup benötigt wird. Um ein Branch zu erstellen, kann man folgendermaßen vorgehen:

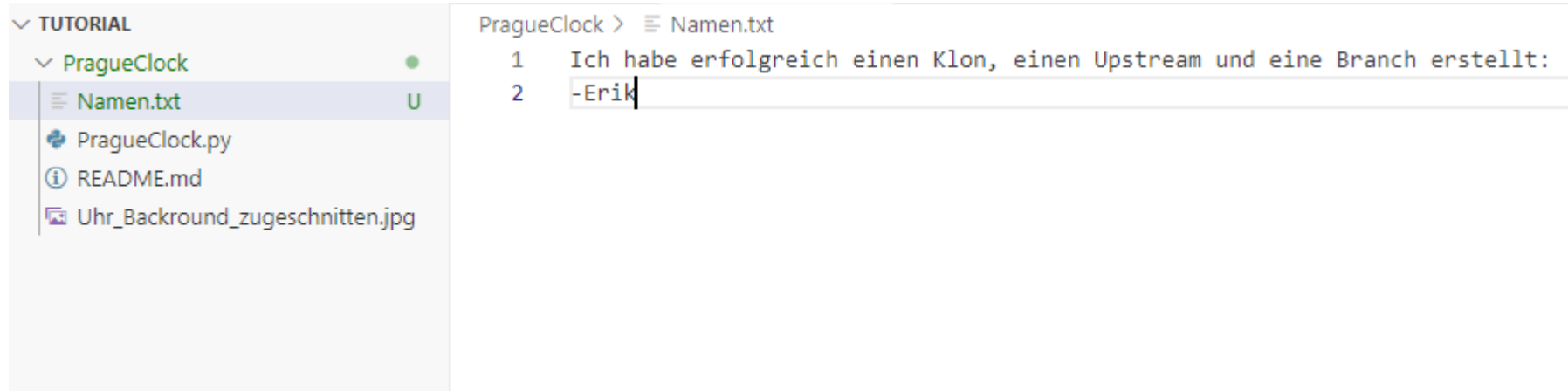
- In der Konsole **git checkout -b nameBranch** eingeben (NAMENSKONVEKTION BEIBEHALTEN!)
- **git branch** eingeben zur Kontrolle

```
PS C:\Users\Kirit\Desktop\Tutorial\PragueClock> git checkout -b TutorialBranch
Switched to a new branch 'TutorialBranch'
PS C:\Users\Kirit\Desktop\Tutorial\PragueClock> git branch
* TutorialBranch
  main
  status
PS C:\Users\Kirit\Desktop\Tutorial\PragueClock> git branch
* TutorialBranch
  main
  status
PS C:\Users\Kirit\Desktop\Tutorial\PragueClock> 
```

Jetzt wurde eine Branch erstellt mit dem Namen TutorialBranch und gleichzeitig auf diese Branch gewechselt. Jetzt kann man los legen und den Code bearbeiten.

Basics - Branch

Als kleinen Test gibt es eine Text Datei, welche Bearbeitet werden soll:



```
PragueClock > Namen.txt
1 Ich habe erfolgreich einen Klon, einen Upstream und eine Branch erstellt:
2 -Erik
```

Füge deinen Namen in die Text Datei hinzu und speicher die Datei (Strg + s). Nun folgt das Hochladen von Dateien von der eigenen Branch auf die main Branch. Man kann einen Blick auf den online Repo werfen. Es wurde die Datei bearbeitet und gespeichert, aber die main Branch wurde davon nicht betroffen. Das ist der Sinn hinter dem Upstream und Branch Workflow!

Basics – Update main branch

Wenn die Text Datei bearbeitet und gespeichert wurde soll folgendes eingegeben werden:

- **git add Namen.txt**
- **git status** (zur Überprüfung, dass es was zum Hochladen gibt)

```
PS C:\Users\Kirit\Desktop\Tutorial\PragueClock> git add Namen.txt
PS C:\Users\Kirit\Desktop\Tutorial\PragueClock> git status
On branch TutorialBranch
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        modified:   Namen.txt

PS C:\Users\Kirit\Desktop\Tutorial\PragueClock> 
```

- **git commit -m „“** (Schreibe kurze aber effektive Kommentare in die Anführungszeichen, das ist wichtig!)

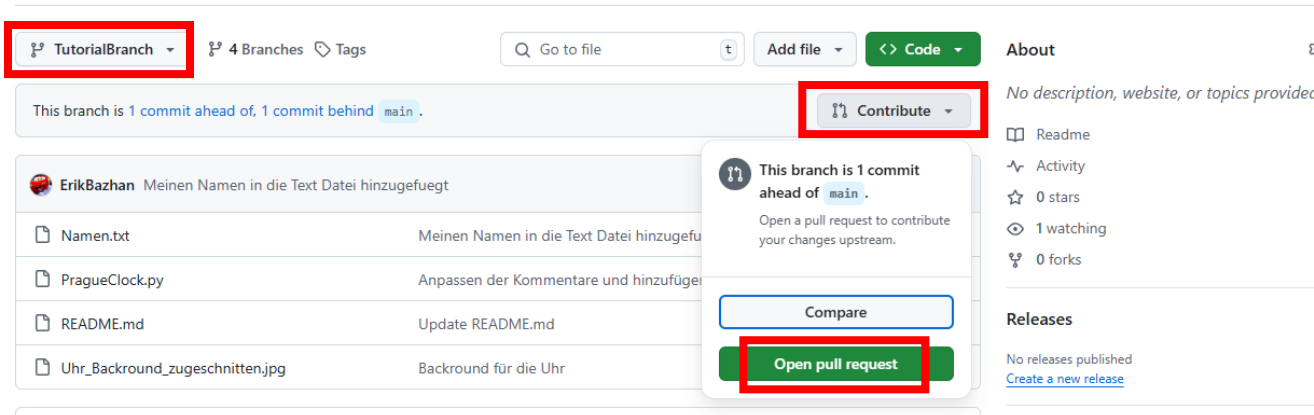
```
PS C:\Users\Kirit\Desktop\Tutorial\PragueClock> git commit -m "Meinen Namen in die Text Datei hinzugefuegt"
[TutorialBranch 4141679] Meinen Namen in die Text Datei hinzugefuegt
 1 file changed, 2 insertions(+), 1 deletion(-)
PS C:\Users\Kirit\Desktop\Tutorial\PragueClock> 
```

- **git push origin TutorialBranch** (Hier statt TutorialBranch Namen der eigenen Branch benutzen)

```
PS C:\Users\Kirit\Desktop\Tutorial\PragueClock> git push origin TutorialBranch
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 16 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 428 bytes | 428.00 KiB/s, done.
Total 3 (delta 1), reused 0 (delta 0), pack-reused 0 (from 0)
remote: Resolving deltas: 100% (1/1), completed with 1 local object.
To https://github.com/ErikBazhan/PragueClock.git
    fb4b229..4141679  TutorialBranch -> TutorialBranch
PS C:\Users\Kirit\Desktop\Tutorial\PragueClock> 
```

Basics – Update main branch

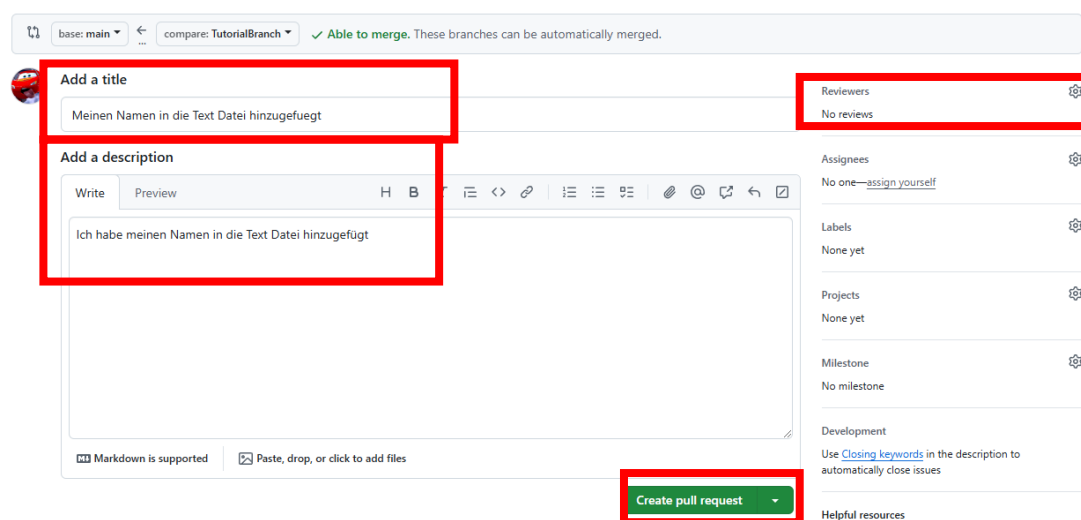
- Auf Github einen Pull Request erstellen, dafür eigenen Branch im Dropdown Menü auswählen



- Namen und Beschreibungen hinzufügen und einen Reviewer hinzufügen

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](#). [Learn more about diff comparisons here](#).





Beim Reviewer am besten mich (ErikBazhan) oder Daniel (Sap1998) auswählen. Wir bekommen eine Nachricht und überprüfen, ob der Pull Request angenommen wird oder nicht


Basics – Update main branch

- Fertig. Bitte nicht selber mergen, sondern mich(ErikBazhan) oder Daniel (Sap1998) als Reviewer einfügen



Meinen Namen in die Text Datei hinzugefuegt #4


 Open ErikBazhan wants to merge 1 commit into `main` from `TutorialBranch` 

Conversation 0 Commits 1 Checks 0 Files changed 1

 ErikBazhan commented 10 minutes ago Owner ...

Ich habe meinen Namen in die Text Datei hinzugefügt

  Meinen Namen in die Text Datei hinzugefügt 4141679

 Continuous integration has not been set up
[GitHub Actions](#) and [several other apps](#) can be used to automatically catch bugs and enforce style.

✓ This branch has no conflicts with the base branch
Merging can be performed automatically. Update branch ▼

Merge pull request ▼ You can also [open this in GitHub Desktop](#) or view [command line instructions](#).

Bitte nicht selber machen, das machen Daniel und ich. Bei Problemen bitte melden

Basics – Update main branch

Nun kann man die eigenen Implementierungen auf dem main Branch sehen. Alle anderen müssen nach dem Merge Request **git fetch --all** in die Konsole tippen, um die Änderungen lokal auf ihrem Branch zu bekommen.

Vorher:

```
1 ■■■■ Namen.txt
...  @@ -0,0 +1 @@
1  + Ich habe erfolgreich einen Klon, einen Upstream und eine Branch erstellt:
```

Nachher:

Files

main

Go to file

Namen.txt

PragueClock.py

README.md

Uhr_Background_zugeschnitten.jpg

PragueClock / Namen.txt

ErikBazhan

Meinen Namen in die Text Datei hinzugefuegt

4141679 · 24 minutes ago

History

Code

Blame

2 lines (2 loc) · 79 Bytes

Code 55% faster with GitHub Copilot

Raw

1

Ich habe erfolgreich einen Klon, einen Upstream und eine Branch erstellt:

2

-Erik

Basics – Zusammenfassung

Nach dem Tutorial können wir nun gemeinsam an einem Code arbeiten. Es wurde das online Repo **geklont**, um die Dateien lokal zu haben. Es wurde mit einem **Upstream** ermöglicht, lokale Dateien auf das online Repo hochzuladen. Es wurde eine **Branch** erstellt, um den main Code im main Branch nicht zu verändern/beschädigen. Es wurde mit **git add** die veränderten Dateien für den Upload vorbereitet. Mit **git commit** wurden die vorbereiteten Dateien mit einer Nachricht versehen. Mit **git push origin** wurde von der eigenen Branch auf die origin (main Branch) hochgeladen. Mit einem **Pull Request** wurden die hochgeladenen Änderungen online vorbereitet. Nach dem Merge wurde der main Branch auf die Änderungen von der eigenen Branch aktualisiert. Nun können alle die neuen Änderungen erhalten mit dem Befehl **git fetch --all**.

Workflow

Somit sieht der Workflow wie folgt aus:

