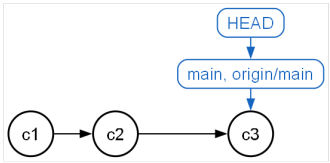
1. (4 Punkte) Du hast ein lokales Repository mit den Branches `main` und `feature`. Auf GitHub hast du ein leeres Repository mit der Adresse `https://github.com/anna/spiel.git` angelegt.
   (a) Gib die Befehle an, mit denen du beide Branches nach GitHub bringst, so dass später ein einfaches `git push` reicht.
   (b) Was bedeuten `origin` und die Option `-u`?

Lösung:

(a)
```
git remote add origin https://github.com/anna/spiel.git
git push -u origin main
git push -u origin feature
```

(b) `origin` ist der Name, unter dem das Remote-Repository (die Adresse auf GitHub) eingetragen wird. `-u` verknüpft den lokalen Branch mit dem gleichnamigen Branch auf GitHub (upstream). Danach weiß Git bei `git push` und `git pull`, wohin bzw. woher.

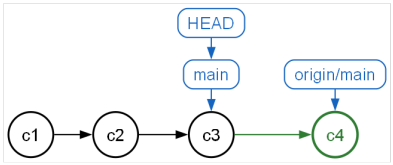
2. (5 Punkte) In der Schule sieht dein Repository so aus. Zu Hause hast du inzwischen einen Commit c4 gemacht und nach GitHub gepusht.



   (a) Zeichne den Graphen in der Schule nach `git fetch`. Was meldet `git status`?
   (b) Zeichne den Graphen danach nach `git pull`. Welche Art von Merge findet statt?
   (c) Was ist der Unterschied zwischen `fetch` und `pull`?

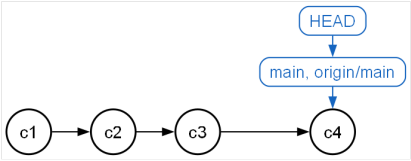
Lösung:

(a) `origin/main` zeigt auf c4, `main` bleibt bei c3:



`git status`: Your branch is behind 'origin/main' by 1 commit (der lokale Branch liegt einen Commit zurück).

(b) `main` rückt auf c4 vor, ein Fast-Forward (kein neuer Commit):



(c) `fetch` holt nur die Commits und bewegt `origin/main`. `pull` macht danach noch einen Merge in den eigenen Branch, das Arbeitsverzeichnis ändert sich.

3. (5 Punkte) Du arbeitest zu Hause und in der Schule am selben Repository. Beide hatten den Stand c1, c2. Zu Hause hast du einen Commit h1 gemacht und gepusht. In der Schule machst du, ohne vorher zu pullen, einen Commit s1 und gibst `git push` ein. Git antwortet:

```
! [rejected]        main -> main (fetch first)
```

   (a) Warum lehnt Git den Push ab?

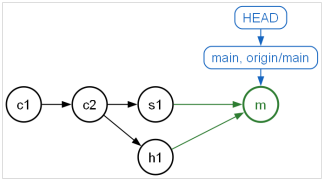
   (b) Was musst du tun? Zeichne den Graphen in der Schule, nachdem du es getan hast.

   (c) Welche Regel vermeidet das Problem?

Lösung:

(a) Auf GitHub liegt h1, das die Schule nicht hat. Würde GitHub den Branch auf s1 setzen, ginge h1 verloren. Ein Push wird nur angenommen, wenn er ein Fast-Forward ist.

(b) `git pull` (holt h1 und erzeugt einen Merge-Commit m aus s1 und h1, bei Konflikten muss man sie vorher lösen), danach `git push`.



(c) Vor der Arbeit `git pull`, nach der Arbeit `git push`.

4. (4 Punkte) Anna hat ein Repository mit den Branches `main` und `grafik` auf GitHub. Ben klont es mit `git clone`.

   (a) Welche Branches hat Ben danach lokal? Was zeigt `git branch -a`?

   (b) Ben möchte am Branch `grafik` weiterarbeiten. Welcher Befehl?

Lösung:

(a) Lokal nur `main`. Geholt wurden aber alle Commits, `git branch -a` zeigt zusätzlich die Remote-Tracking-Branches `remotes/origin/main` und `remotes/origin/grafik` (und `remotes/origin/HEAD`).

(b) `git switch grafik`: Git legt einen lokalen Branch `grafik` an, der mit `origin/grafik` verknüpft ist.

5. (4 Punkte) `git branch -vv` zeigt (nach einem `git fetch`):

```
* feature e4f5a6b [origin/feature: ahead 1] d2
  main    a1b2c3d [origin/main: behind 2] c2
```

   (a) Was bedeuten `ahead 1` und `behind 2`?

   (b) Welchen Befehl brauchst du jeweils, damit lokaler Branch und GitHub wieder übereinstimmen?

Lösung:

(a) `feature` hat einen Commit, der noch nicht auf GitHub ist (`ahead 1`). Auf GitHub hat `main` zwei Commits, die lokal fehlen (`behind 2`); jemand anders hat gepusht.

(b) Auf `feature`: `git push`. Auf `main`: `git pull` (hier ein Fast-Forward, weil `main` lokal keine eigenen neuen Commits hat).

6. (5 Punkte) (a) Was ist ein Pull Request? Warum ändert man `main` in einem Team nur über Pull Requests?

   (b) Was ist der Unterschied zwischen einem Collaborator und jemandem, der zu einem fremden Open-Source-Projekt beitragen will?

   (c) Beschreibe die Schritte, mit denen du eine Verbesserung zu einem fremden Projekt auf GitHub beisteuerst.

Lösung:

(a) Ein Pull Request ist der Vorschlag, die Commits eines Branches in einen anderen (meist `main`) zu übernehmen. Andere können die Änderungen ansehen, kommentieren und annehmen oder ablehnen. So wird Code geprüft, bevor er in `main` kommt, und `main` bleibt lauffähig.

(b) Ein Collaborator hat Schreibrechte im Repository, er kann Branches direkt dorthin pushen. Bei einem fremden Projekt hat man keine Schreibrechte.

(c) Auf GitHub einen Fork anlegen (eigene Kopie), den Fork klonen, einen Branch anlegen, Änderungen committen, in den eigenen Fork pushen, auf GitHub einen Pull Request an das Original-Repo stellen. Die Betreuer des Projekts entscheiden, ob sie ihn übernehmen.