



HÁSKÓLINN Í REYKJAVÍK
REYKJAVIK UNIVERSITY

Notkunarleiðbeiningar fyrir Git, GitHub og GitHub Desktop

(hvað eru mörg git í því?)

Efnisyfirlit

Inngangur.....	3
Uppsetning reikninga.....	4
GitHub.....	4
Git.....	4
GitHub Desktop.....	4
Pull request / code reviews.....	9
Branching / merging.....	9
Branching strategies.....	9
Git flow.....	10
GitHub Flow.....	11
Árekstrar.....	11
Leiðrétting á þessum árekstri með git terminal.....	11
Leiðrétting á þessum árekstri í GitHub Desktop.....	12
Annað dæmi um árekstur leiðréttan í GitHub Desktop.....	14
Algeng vandamál.....	15
Detached HEAD state.....	15
Commit án stage.....	15
Óskráðar skrár eftir pull.....	15
Grunnaðgerðir í git terminal.....	16
init.....	16
clone.....	16
status.....	16
add.....	16
commit.....	17
branch.....	17
checkout.....	17
remote.....	18
push.....	18
pull.....	18
fetch.....	18
merge.....	19
Grunnaðgerðir í GitHub Desktop.....	20
Stofna nýja geymslu.....	20
Klóna geymslu.....	22
Setja inn breytingu.....	23
Sækja breytingar.....	24
Pulla breytingar.....	24
Pusha á remote.....	25
Stofna grein.....	26
Eyða grein.....	27
Skipta um grein.....	28
Sameina grein.....	29
Stofna pull request.....	30
Samþykkja pull request.....	32
Loka/hafna pull request.....	32

Inngangur

Við þróun hugbúnaðar er mikilvægt að halda vel utan um þær skrár sem unnið er með. Ein algengasta aðferðin sem er notuð til þess er svonefnd útgáfustýring (e. version control). Útgáfustýringarkerfi eru tölvukerfi sem eru sérhæfð til þess að vinna með mismunandi útgáfur af skrám og þau hafa oftast líka góðan stuðning fyrir samvinnu.

Git er eitt slíkt kerfi. Git getur haldið utan um skrár og breytingar á þeim ásamt því að það býður upp á fjölmarga öfluga möguleika til samvinnu. Maður stofnar git geymslur (e. repository) sem innihalda allar upplýsingar um breytingar á þeim skrám sem maður er að vinna með.

GitHub er vefsíða/þjónusta þar sem fólk getur hýst git geymslurnar sínar og ýmislegt tengt þeim. GitHub Desktop er forrit fyrir Windows og MacOS sem gerir fólki kleift að vinna með GitHub verkefni á þægilegan hátt í myndrænu viðmóti.

Uppsetning reikninga

GitHub

Til að stofna GitHub reikning þá þarf að fara á <https://github.com/>, smella á “Sign up” og fylgja nokkrum einföldum skrefum. Sjá <https://docs.github.com/en/get-started/start-your-journey/creating-an-account-on-github> fyrir nánari útskýringar á því ferli.

Git

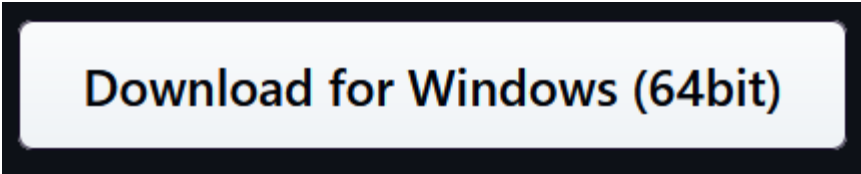
Til að nota git þarf maður að setja upp git hugbúnaðinn á tölvunni hjá sér. Nákvæma ferlið er mismunandi eftir því hvaða stýrikerfi fólk notar en hérna má nálgast frekari upplýsingar um það: <https://git-scm.com/book/en/v2/Getting-Started-Installing-Git>

Þegar uppsetningunni er lokið þá fer aðalnotkun git í raun fram í terminal gluggum þar sem maður keyrir þær git skipanir sem maður þarf. Einnig er hægt að nota ýmis hjálparforrit til að aðstoða við að keyra git skipanir eins og GitHub Desktop eða viðbætur í VSCode.

GitHub Desktop

Til að setja upp GitHub Desktop þá þarf að fara á slóðina <https://desktop.github.com/download/>. Smáatriði uppsetningarinnar eru mismunandi milli stýrikerfa. Fyrir t.d. Windows þá er ferlið eftirfarandi:

Fyrst þarf að smella á “Download for Windows (64bit)”

A rectangular button with a dark background and a light-colored border. The text "Download for Windows (64bit)" is centered in a bold, white, sans-serif font.

Svo þarf að opna skrána “GitHubDesktopSetup-x64.exe”



GitHubDesktopSetup-x64.exe

Completed — 162 MB



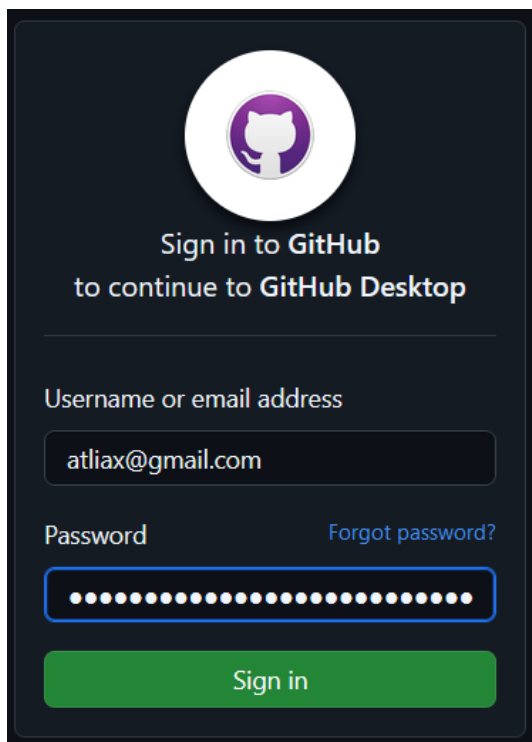
Þá ætti þessi gluggi að birtast:



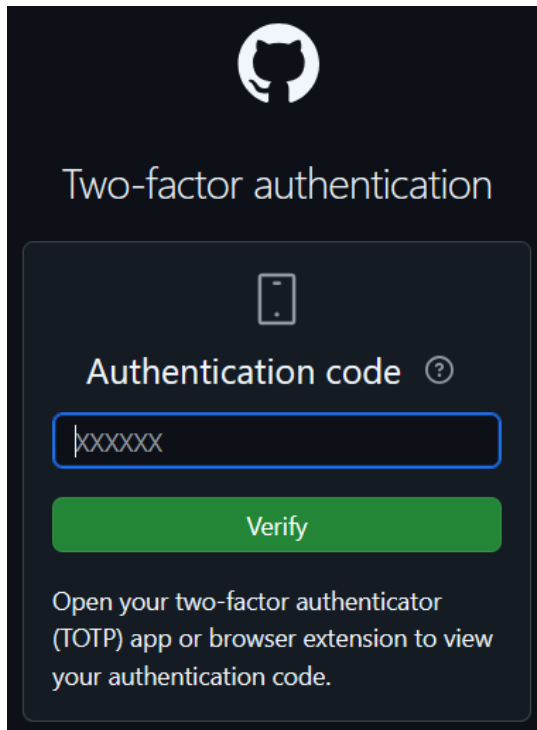
Eftir smástund ætti uppsetningin að klárast og GitHub Desktop glugginn að opnast.

Þá þarf að smella á "Sign in to GitHub.com" í GitHub Desktop glugganum.

Þá ætti að opnast vafragluggi sem biður um GitHub innskráningu:

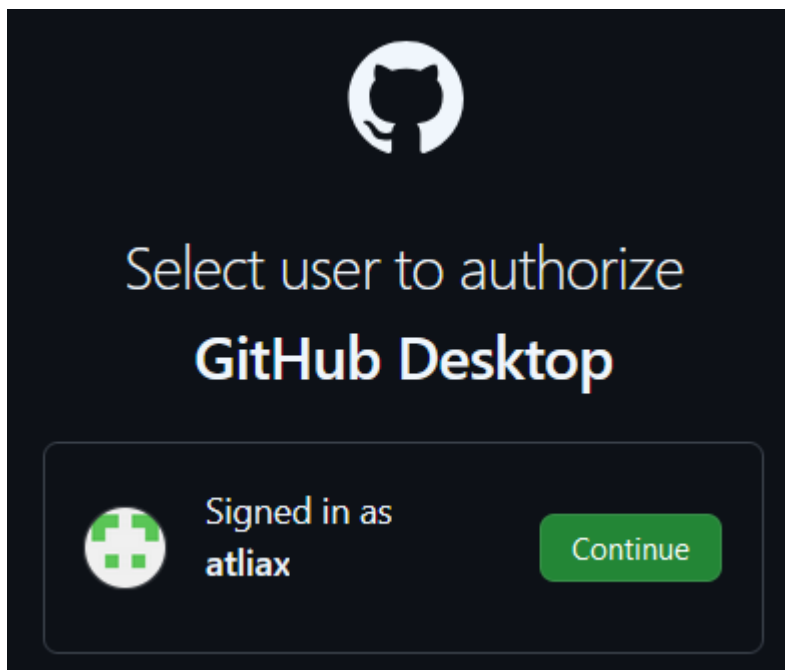
A dark-themed sign-in window for GitHub. At the top is the GitHub logo. Below it, the text says "Sign in to GitHub to continue to GitHub Desktop". There are two input fields: "Username or email address" with the value "atliax@gmail.com" and "Password" which is masked with dots. A link "Forgot password?" is next to the password field. At the bottom is a green "Sign in" button.

Ef tveggja þátta auðkenning er virk þarf að setja inn þann kóða líka:



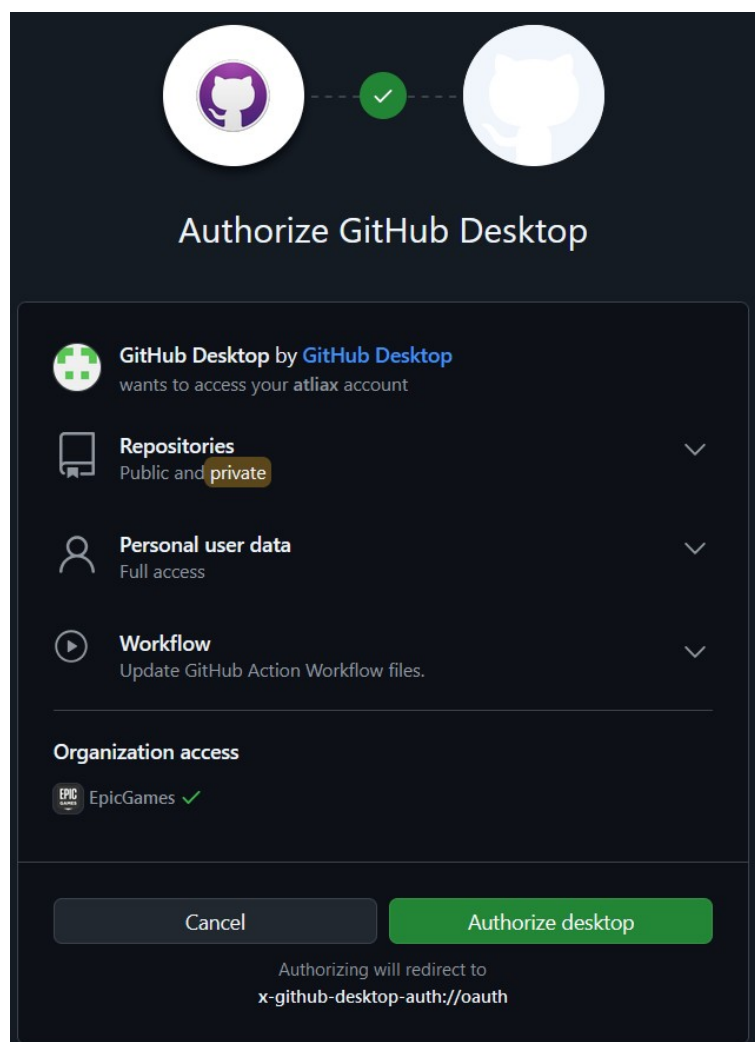
The image shows a dark-themed GitHub interface for two-factor authentication. At the top is the GitHub logo. Below it, the text "Two-factor authentication" is displayed. In the center, there is a box containing a smartphone icon, the label "Authentication code" with a help icon, a text input field with "xxxxxx" and a cursor, and a green "Verify" button. At the bottom of this box, there is instructional text: "Open your two-factor authenticator (TOTP) app or browser extension to view your authentication code."

Síðan þarf að velja hvaða notanda maður er að auðkenna fyrir GitHub Desktop:

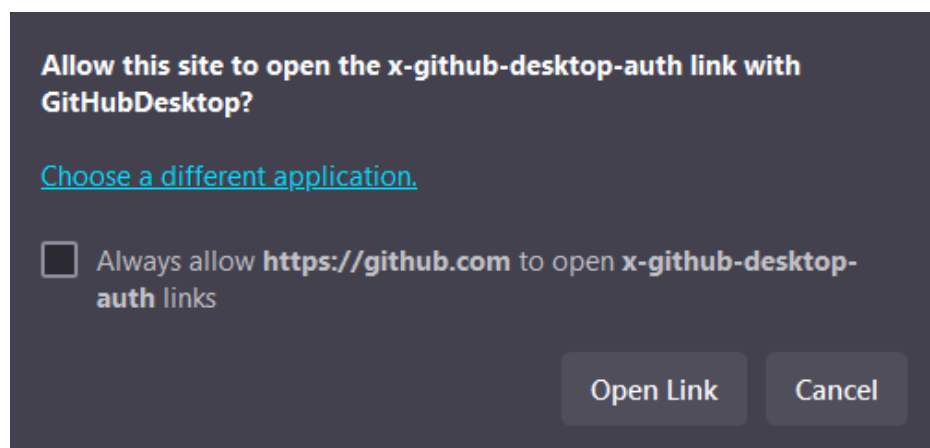


The image shows a dark-themed GitHub interface for selecting a user to authorize GitHub Desktop. At the top is the GitHub logo. Below it, the text "Select user to authorize" is displayed, followed by "GitHub Desktop" in a larger font. At the bottom, there is a box containing a user profile icon, the text "Signed in as atliax", and a green "Continue" button.

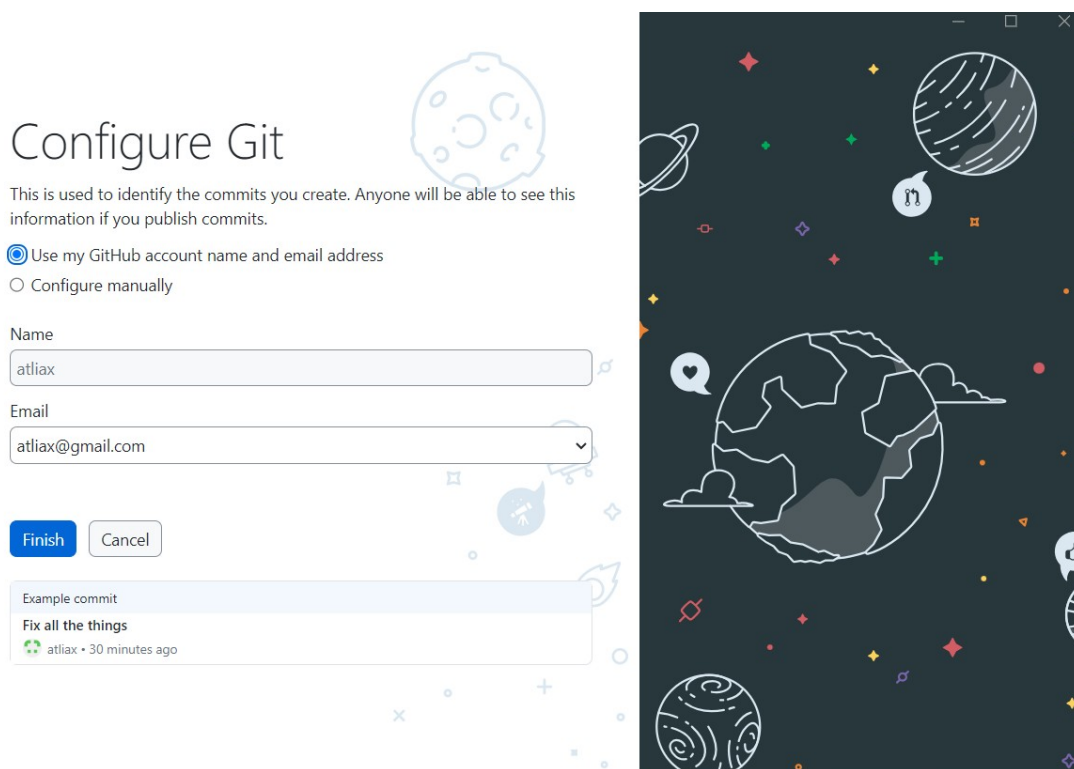
Svo þarf að samþykkja GitHub Desktop auðkenninguna fyrir notandann með því að smella á “Authorize Desktop”:



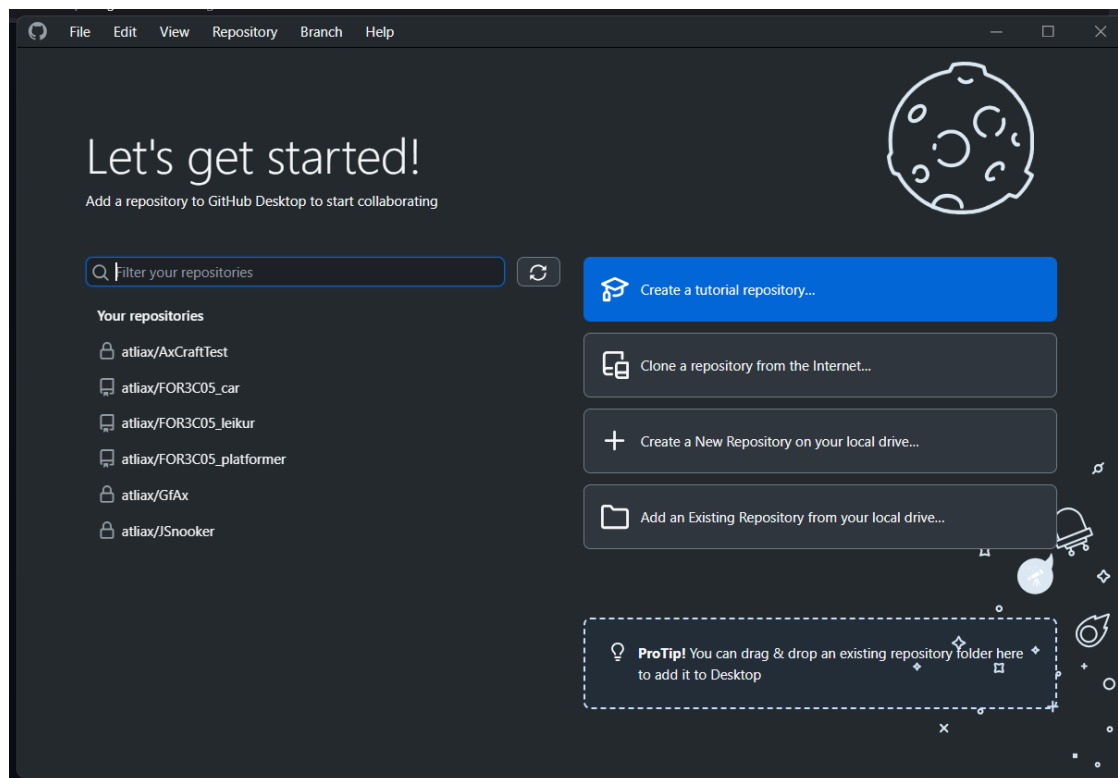
Þegar því er lokið ætti GitHub Desktop að opnast. Í einhverjum tilfellum gæti þurft að samþykkja opnunina í vafraglugganum:



Í GitHub Desktop glugganum þarf svo að stilla hvaða nafn og tölvupóstur fylgir þeim breytingum (e. commit) sem maður setur inn:



Núna ætti GitHub Desktop að vera tilbúið til notkunar og eftirfarandi viðmót að koma:



Pull request / code reviews

Þegar fólk vill framkvæma breytingar á verkefni á GitHub, hvort sem það er verkefni sem einhver annar á eða sitt hvor greinin í verkefni sem maður á sjálfur, þá er hægt að stofna svonefnt “pull request”. Þá safnar maður saman breytingum af einhverri grein sem maður er að vinna í og sendir þær sem beiðni um sameiningu.

Sá sem tekur við beiðninni getur þá farið yfir breytingarnar í beiðninni og ýmist samþykkt eða hafnað. Ef beiðnin er samþykkt þá eru breytingarnar sameinaðar yfir í greinina sem við á.

Ein algeng aðferð er að finna fyrst eitthvað verkefni á GitHub sem manni finnst maður geta hjálpað með. Síðan gerir maður “fork” af því, sem gerir fullkomið afrit af því verkefni á manns eigin GitHub reikningi. Þá getur maður unnið með “forkaða” afritið og tekið svo saman þær breytingar maður gerir, sett þær í pull request og sent það á upprunalega verkefnið. Þá getur sá sem sér um upprunalega verkefnið farið yfir breytingarnar og sameinað þær inn í upprunalega verkefnið ef hann vill taka við þeim.

Pull request geta líka verið gagnleg þegar fólk er að vinna saman í verkefni. Þá getur einn kannski séð um aðalgrein verkefnisins og hinir unnið í sínum breytingum á sérgreinum. Síðan þegar breytingarnar eru tilbúna geta hinir sent þær sem pull request á aðalgreinina.

Þegar Pull Request hefur verið stofnað geta þeir sem hafa aðgang að geymslunni á GitHub sett inn athugasemdir og þannig átt umræðu við samstarfsfólk þar sem breytingarnar eru ræddar.

Branching / merging

Það getur verið öflugt að nota greinar til að þróa nýja virkni. Þannig getur maður unnið í nýju virkninni án þess að hafa áhrif á aðalgreinina. Þannig má t.d. stofna nýja grein, vinna í nýju virkninni og sameina hana svo yfir á aðalgreinina þegar nýja virknin er tilbúin.

Branching strategies

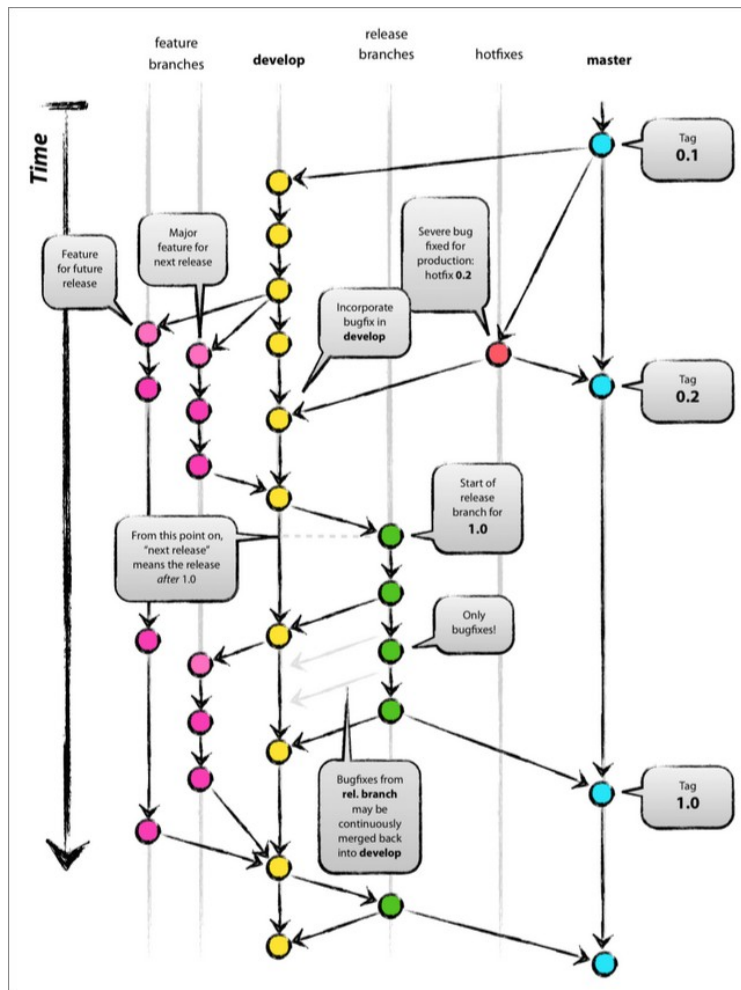
Það er búið að útfæra ýmsar sniðugar vinnuaðferðir þegar það kemur að vinnslu með greinum. Með því að vinna kerfisbundið með greinum er hægt að hafa skýrt og afmarkað ferli fyrir annað hvort nýja virkni eða nýjar útgáfur af hugbúnaðinum.

Git flow

Ein vinsæl aðferð til að vinna með greinar heitir “git flow”. Aðferðin gengur í aðalatriðum út á það að hafa tvær aðalgreinar, eina “master” grein og eina “develop” grein. Master greinin er þá sú grein forritsins þar sem hver breyting á þeirri grein er tilbúin útgáfa af forritinu, t.d. v0.1 eða v0.2. Develop greinin mætti segja að sé þá í rauninni aðalgrein forritsins þar sem öll þróun á því fer fram. Öll ný virkni forritsins er svo útfærð með því að gera hliðargreinar af develop greininni. Þegar virknin er tilbúin eru virknigreinar sameinaðar aftur á develop greinina og þá er hægt að eyða þeim.

Síðan þegar ný útgáfa af forritinu er nálægt því að vera tilbúin er hægt að gera “release” grein sem hliðargrein af develop greininni. Á þeirri grein má bara laga villur, ekki þróa nýja virkni. Þegar útgáfan er tilbúin er svo hægt að sameina release greinina inn á master greinina og gefa út nýja útgáfu af forritinu. Á því stigi er release greinin líka sameinuð inn á develop greinina og þá er hægt að eyða release greininni. Ef alvarlega villur koma fram á master greininni er líka hægt að búa til hotfix grein út frá henni og laga þær. Þegar því er lokið þarf að sameina hotfix greinina yfir á bæði master og develop.

Hægt er að sjá dæmi um þetta ferli á eftirfarandi mynd:



Höfundur myndar: Vincent Driessen (Creative Commons BY-SA)

GitHub Flow

GitHub Flow er aðferð sem er aðeins einfaldari í notkun en git flow. Í GitHub flow eru ekki hafðar develop, hotfix eða release greinar. Þar er notuð ein aðalgrein t.d. main eða master sem er leitast eftir að sé alltaf tilbúin til útgáfu. Þegar bæta á við einhverri virkni er svo stofnuð ný grein þar sem sú virkni er útfærð.

Þegar virknin á nýju greininni er tilbúin eru breytingarnar teknar saman í Pull Request yfir á master greinina. Eftir að það er búið að sameina Pull Requestið er hægt að eyða virknigreininni.

Árekstrar

Þegar tveir eða fleiri vinna saman í sama kóðanum þá geta orðið árekstrar þegar það kemur að því að sameina breytingar. Þetta á sérstaklega við ef fleiri en einn vinnur í sömu skrá. Dæmi um árekstur er ef skjal er búið til og einhver texti settur í hana. Svo er búið til önnur grein og settur einhver annar texti í skrána. Svo er skipt aftur á fyrri greinina og bætt meiri texta við upprunalega textann. Þá er kominn árekstur ef maður reynir að sameina nýju greinina yfir á fyrri greinina.

Leiðrétting á þessum árekstri með git terminal

Ef maður keyrir git merge skipunina kemur eftirfarandi villa:

```
CONFLICT (content): Merge conflict in [filename]  
Automatic merge failed; fix conflicts and then commit the result.
```

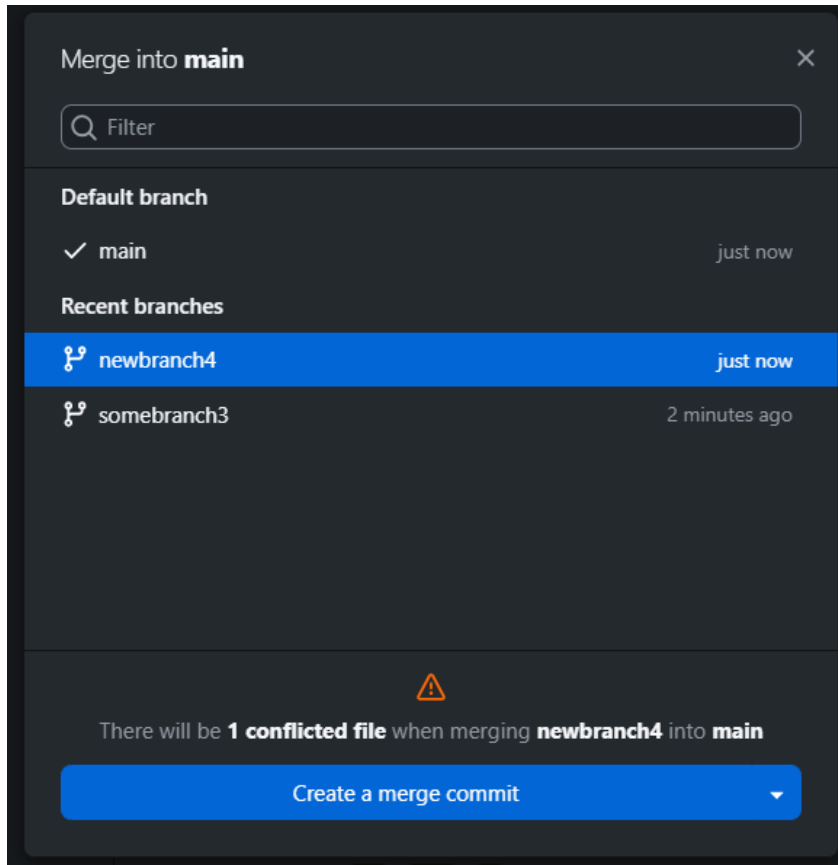
Þarna þarf að laga áreksturinn til að geta haldið áfram. Ef maður skoðar skrána þá lítur hún út einhvern veginn svona:

```
<<<<<< HEAD  
fyrsti textinn  
textinn sem var bætt við í fyrri grein  
=====  
nýr texti úr seinni grein  
>>>>>> seinni_grein
```

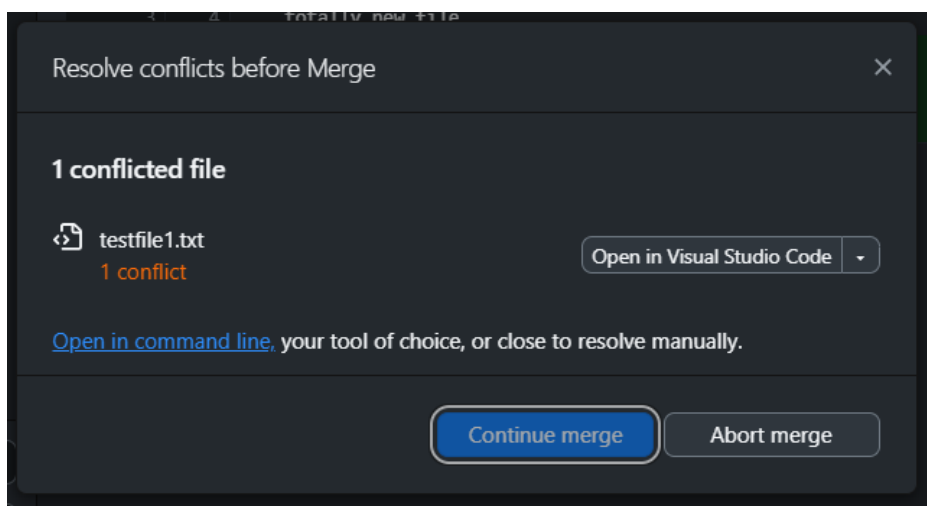
Þarna er innihaldið úr skránni af fyrri greinin á milli <<<<<<HEAD línunnar og ===== striksins. Innihaldið úr skránni af seinni greininni er svo á milli ===== striksins og >>>>>> seinni_grein. Þarna þarf að fara í gegnum skrána og leiðrétta handvirkt áður en það er hægt að commita því og klára sameininguna.

Leiðrétting á þessum árekstri í GitHub Desktop

Fyrst koma eftirfarandi villuboð í sameiningarglugganum:



Ef maður smellir svo á “Create a merge commit” kemur upp gluggi sem aðstoðar mann í að finna út úr árekstrinum:



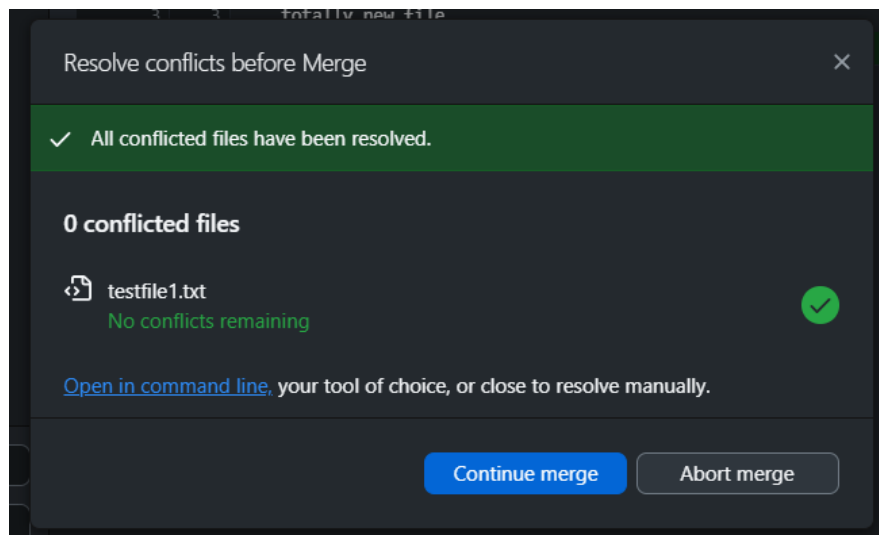
Þarna er t.d. hægt að smella á “Open in Visual Studio Code” (eða öðrum textabreytforritum). VS Code getur verið gott til þess þar sem það hjálpar manni með því að litakóða árekstrana gefur takka sem er hægt að nota til að leiðrétta áreksturinn:

```
Accept Current Change | Accept Incoming Change | Accept Both Changes | Compare Changes
<<<<<<< HEAD (Current Change)
some contents
this is modified <---
totally new file
=====
replaced everything
>>>>>>> newbranch4 (Incoming Change)
```

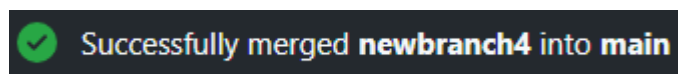
Ef smellt er á t.d. “Accept Both Changes” þá sameinar VS Code allar breytingarnar:

```
ers > Moses Feinberg > Documents > GitHub > testrepo1 >
some contents
this is modified <---
totally new file
replaced everything
```

Þá getur maður vistað skrána, skipt aftur yfir í GitHub Desktop og smellt á “Continue merge” til að klára sameininguna:

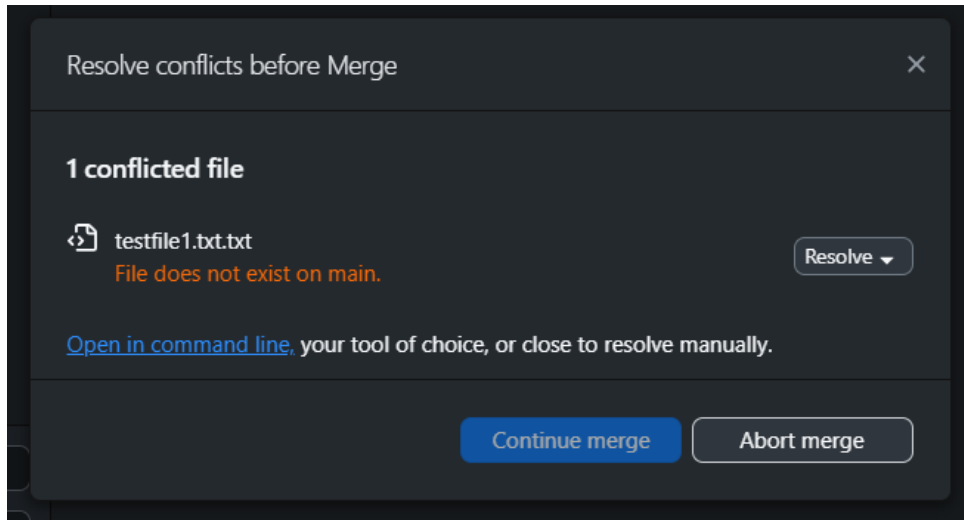


Ef allt fór vel ætti þessi borði að koma efst:

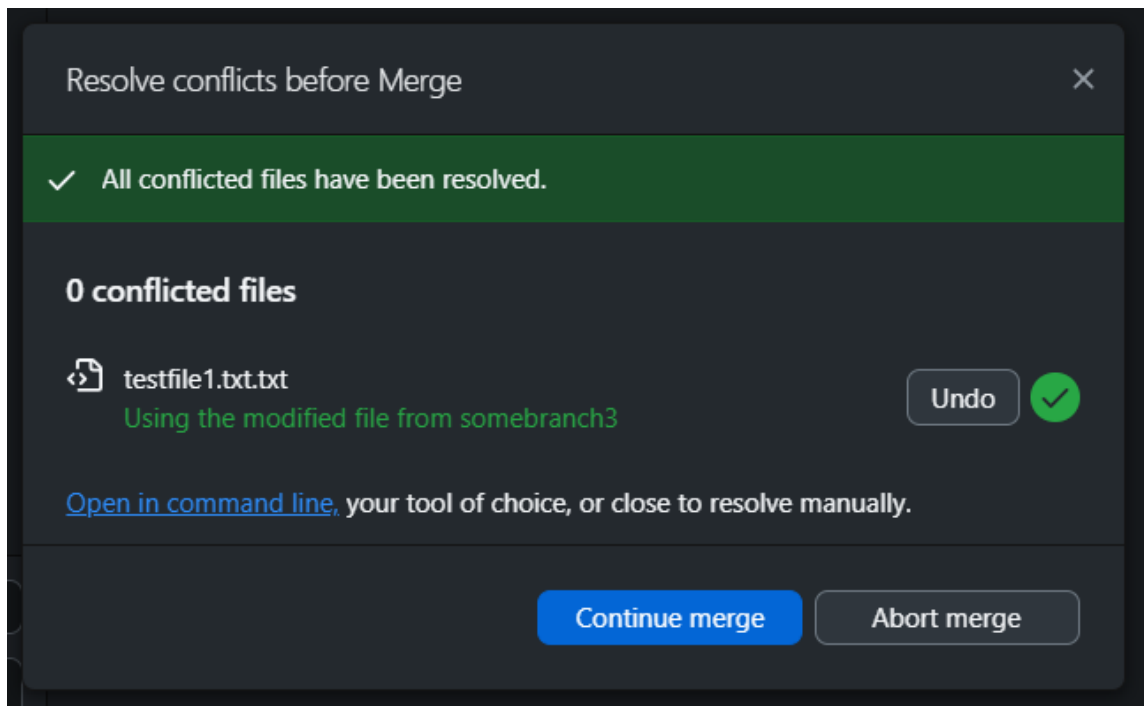


Annað dæmi um árekstur leiðréttan í GitHub Desktop

Önnur gerð af árekstri getur komið upp þegar skrá er endurskírð á einni greini og henni svo breytt. Ef maður setur svo inn breytingu í útgáfuna sem var til á hinn greininni með gamla skráarheitinu kvartar Git yfir því að skráin sé ekki til á fyrri greininni:



Þarna er hægt að velja nokkra möguleika í “resolve” fellivalinu. Ef maður velur t.d. að nota skrána af seinni greininni (með upprunalega skráarnafninu) þá er hægt að halda sameiningunni áfram með því að smella á “Continue merge”:



Þá verða til tvær útgáfur af skránni. Annars vegar skráin af aðalgreininni með nýja nafninu og líka skráin af hinn greininni með gamla nafninu.

Algeng vandamál

Ýmis vandamál geta komið upp við notkun á Git og langoftast (eða alltaf) er hægt að leita á Google og finna aðra sem hafa lent í sama vandamáli. Þannig er nánast alltaf hægt að finna lausnir á hverju því sem kemur upp. Þó er tilefni til að nefna nokkra hluti hérna.

Detached HEAD state

Ef maður keyrir checkout skipun á breytingu en ekki grein þá getur maður endað í því sem Git kallar “Detached HEAD state” þar sem HEAD bendillinn bendir ekki lengur á neina ákveðna grein. Það er hægt að koma sér til baka með því að keyra checkout á einhverja grein eða stofna nýja grein á þeim stað sem HEAD er með því að keyra **git checkout -b nafnágrein**.

Commit án stage

Ef maður reynir að setja inn breytingu án þess að setja breyttu skrána á “staging” svæðið fyrst þá neitar Git að setja inn breytinguna. Þetta er vegna þess að Git tekur ekki afstöðu til þess hvaða skrá maður vill setja inn breytingu á fyrir mann. Maður gæti verið að vinna í mörgum skráum í einu en viljað setja inn breytingu á hluta þeirra án þess að setja inn breytingar á hinum. Þess vegna vill Git að maður keyri fyrst add skipunina til að merkja við þær skrár sem maður ætlar að skrá breytingar á. Hægt er að nota -a flaggið á commit skipuninni til að setja sjálfkrafa allar skrárnar á staging svæðið ef maður ætlaði að gera það. GitHub Desktop er þó byggt þannig að þetta vandamál kemur ekki fram vegna þess að sjálfvalda leiðin þar er að setja inn allar breytingar nema maður velji annað.

Óskráðar skrár eftir pull

Ef maður reynir að keyra pull og það ætla að koma skrár með því sem myndu skrifa yfir skrár sem Git geymslan á drifinu hjá manni er ekki að halda utan um, þá neitar Git að gera það. Þá er hægt að geyma breytingarnar sem maður á eftir að skrá, framkvæma pull skipunina og sækja breytingarnar aftur með eftirfarandi skipunum:

git stash

git pull

git stash pop

Grunnaðgerðir í git terminal

init

Init aðgerðin býr til tóma geymslu sem maður getur svo bætt skráum í.

Dæmi:
git init

clone

Clone aðgerðin í git tekur við slóð á git geymslu og gerir afrit af henni sem maður getur svo notað til að vinna með skrárnar sem viðkomandi geymsla inniheldur. Þetta er í raun grunnaðgerðin sem þarf nánast alltaf að byrja á ef maður vill taka þátt í verkefni sem er haldið utan um með git.

Dæmi: git clone <https://github.com/ohmyzsh/ohmyzsh.git>

status

Með status skipuninni er hægt að sjá hvaða skráum hefur verið breytt og hvaða skrár, ef einhverjar, eru komnar inn á “staging” svæðið.

Dæmi:
git status

add

Þegar maður er búinn að gera breytingar á skráum, þá er hægt að nota status skipunina til að sjá hvaða skráum hefur verið breytt. Síðan er hægt að nota add skipunina til að bæta skráum inn á “staging” svæði til að hægt sé að skrá inn breytingarnar á þeim með commit skipuninni.

Dæmi 1:
git add .

Tekur allar skrár í öllum möppum í geymslunni sem hafa breytingar og setur þær í staging. Með öðrum orðum, gera allar breytingar tilbúna fyrir commit.

Dæmi 2:
git add docs/README

Tekur bara skrána README inni í docs möppunni og setur hana í staging. Það getur verið nytsamlegt að keyra add bara á ákveðnum skráum ef maður vill t.d. setja inn litlar afmarkaðar breytingar í staðinn fyrir að setja allar breytingar inn í einu stóru commit sem getur leitt til ruglings.

commit

Þegar maður er búinn að keyra clone skipunina, þá getur maður breytt skráum eftir þörfum og fært breytingarnar inn í geymsluna með commit skipuninni. Þar þarf að fylgja með stutt lýsing á viðkomandi breytingu/viðbót. Maður notar “-m” parameterinn til að skilgreina lýsinguna. Þær skrár sem commit skipunin skráir breytingar á eru þær sem búið er að setja inn á “staging” svæðið.

Dæmi:

```
git commit -m "Uppfærði API key fyrir Instagram"
```

branch

Í git geymslum er hægt að skilgreina mismunandi greinar sem gerir manni kleift að gera breytingar á hliðargrein án þess að fíka í því sem er á aðalgreininni. Þannig gæti maður t.d. verið með master eða main grein sem er með aðalútgáfuna af kóðanum og stofnað svo nýja grein til að útfæra einhverja nýja virkni. Svo þegar nýja virknin er tilbúin er hægt að sameina hliðargreinina á aðalgreinina.

Branch skipunina er bæði hægt að nota til að birta þær greinar sem eru til og til að stofna nýjar greinar.

Dæmi 1:

```
git branch new-branch
```

Stofnar nýja grein með nafninu “new-branch”.

Dæmi 2:

```
git branch
```

Birtir lista yfir þær greinar sem eru til í geymslunni.

checkout

Checkout skipunina er hægt að nota til að skipta vinnuumhverfinu yfir á aðra grein. T.d. þegar maður er búinn að stofna nýja grein með branch skipuninni myndi maður keyra checkout skipunina til að byrja að vinna á nýju greininni.

Einnig er hægt að nota checkout skipunina til að búa til nýja grein um leið og maður skiptir yfir á hana, sjá dæmi 2 hér að neðan.

Dæmi 1:

```
git checkout new-branch
```

Dæmi 2:

```
git checkout -b new-branch master
```

Býr til nýja grein sem heitir “new-branch” út frá “master” greininni og skiptir yfir á hana.

remote

Þegar maður vinnur með git geymslur þá er algengt að hafa skráð einhvern stað sem “remote” útgáfu af geymslunni. Það gæti t.d. verið á GitHub, GitLab eða einhverjum sambærilegum hýsingarstað. Þar er þá geymt afrit af geymslunni sem fleiri en einn aðili geta haft skráð sem remote hjá sér. Þá vinnur fólk með gögnin á drifinu hjá sér og sækir svo og sendir í remote geymsluna eftir þörfum.

Remote skipunina er hægt að nota til að bæta við nýjum remote geymslum, skoða þær remote geymslur sem eru þegar skilgreindar og breyta þeim.

Dæmi:

```
git remote add origin https://github.com/atliax/T-113-VLN1\_hopur2.git
```

Bætir geymslunni “T-113-VLN1_hopur2” frá GitHub notandanum “atliax” sem remote með nafninu “origin”.

push

Þegar maður er búinn að gera einhverjar breytingar á skrá in geymslunni sem maður bjó til með clone skipuninni þá getur maður notað push skipunina til að ýta breytingunum upp í það remote sem maður vill nota t.d. til að leyfa öðrum að fá breytingarnar. Gott dæmi er þegar maður er að vinna með GitHub þá sækir maður geymsluna með clone, breytir einhverju og pushar svo upp í GitHub svo hinir sem maður er að vinna með geti fengið nýju breytingarnar.

Dæmi:

```
git push origin master
```

Sendir “master” greinina upp í það remote sem maður hefur skilgreint undir nafninu “origin”.

pull

Svipað og push nema í hina áttina. Samsagt notað til að sækja nýjustu breytingarnar úr remote geymslu. Pull er sambærilegt við að keyra fyrst fetch og svo merge skipanirnar.

Dæmi:

```
git pull origin master
```

Sækir “master” greinina úr því remote sem maður hefur skilgreint undir nafninu “origin” og sameinar það inn á þá grein sem maður er með virka.

fetch

Fetch er hægt að nota til að sækja þær breytingar sem eru til í remote en eru ekki til staðar í geymslunni sem maður er að vinna með á staðbundna drifinu hjá manni. T.d. ef einhver stofnar nýja grein og pushar henni í remote, þá er hægt að nota fetch til að sækja þessa nýju grein án þess að fíkta í þeirri grein sem maður er sjálfur að vinna með.

Dæmi:

```
git fetch
```

merge

Merge er notað til að sameina tvær eða fleiri breytingarsögur. Það tekur þá þær breytingar sem hafa verið gerðar á þeirri grein sem maður biður um að sé sameinuð og gerir tilraun til að samhæfa þær við greinina sem er virk. Merge stoppar ef það rekst á breytingu sem er ekki hægt að samhæfa sjálfvirkt.

Dæmi 1:

`git merge new-branch`

Ef við gefum okkur að greinin “master” sé virk, þá tekur þetta dæmi breytingarnar sem hafa orðið til á “new-branch” greinni frá því hún varð aðskilin frá “master” greinni og gerir tilraun til að samhæfa þær á “master” greinina.

Dæmi 2:

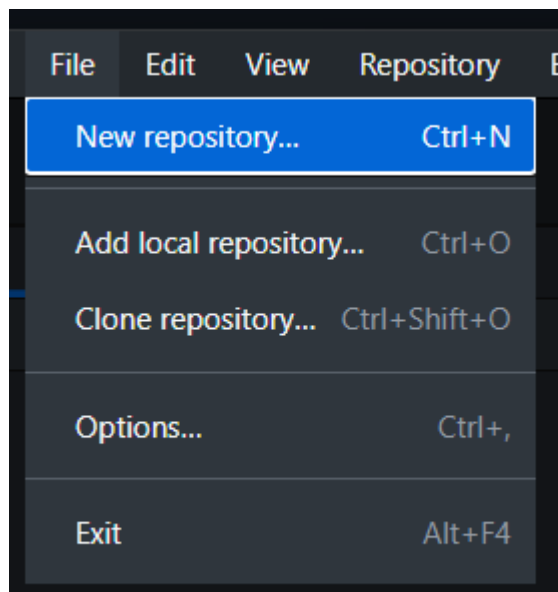
`git merge --abort`

Ef fyrri merge skipun náði ekki að klára þá er hægt að nota --abort til að reyna að endurskapa aðstæðurnar sem voru til staðar áður en sú merge skipun var keyrð.

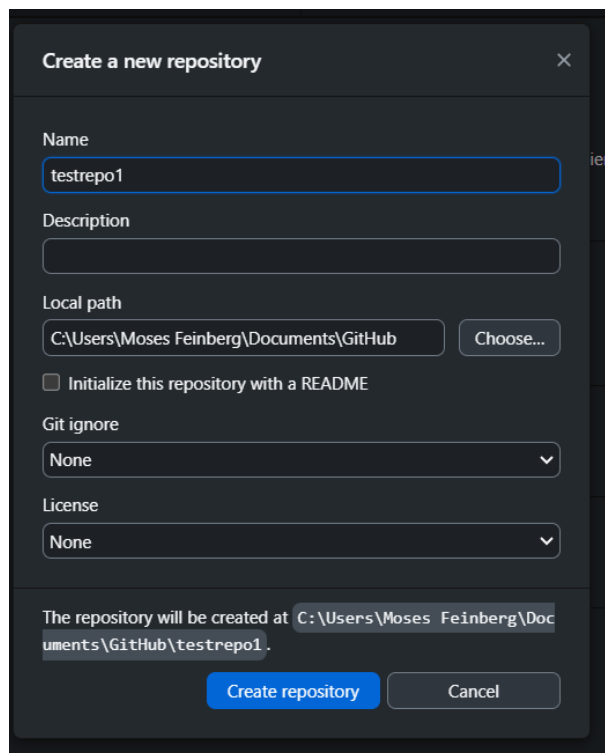
Grunnaðgerðir í GitHub Desktop

Stofna nýja geymslu

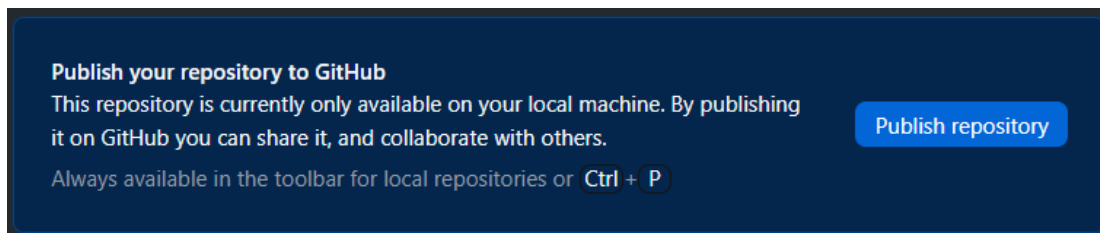
Til að stofna nýja geymslu þá er hægt að fara í File → New repository eða ýta á Ctrl+N.



Þá opnast þessi gluggi þar sem maður stillir grunnatriði nýju geymslunnar:



Svo smellir maður á “Create repository”. Þá verður til tóm geymsla sem er einungis til á tölvunni hjá manni sjálfum. Maður hefur þá möguleikann á því að setja hana inn á GitHub með því að smella á “Publish repository” í aðalglugganum:

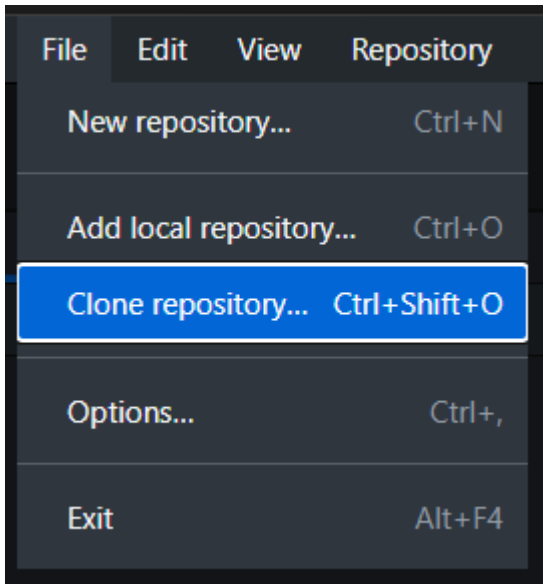


Þá opnast gluggi sem gerir manni kleift að velja stillingar fyrir GitHub geymsluna:

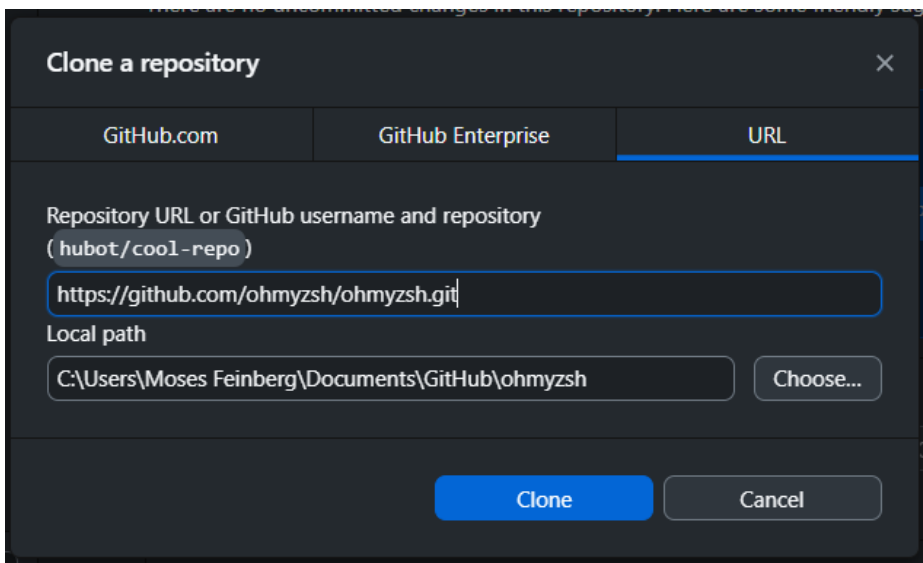
Þegar maður er búinn að slá inn þær upplýsingar sem maður vill þá smellir maður aftur á “Publish repository” og þá stofnast geymslan á GitHub.

Klóna geymslu

Til að klóna geymslu er hægt að fara í File → Clone repository eða ýta á Ctrl+Shift+O.



Svo velur maður annað hvort “GitHub.com” ef maður vill sækja geymslu á GitHub reikninginn hjá sjálfum manni eða “URL” ef maður vill sækja geymslu sem einhver annar á:

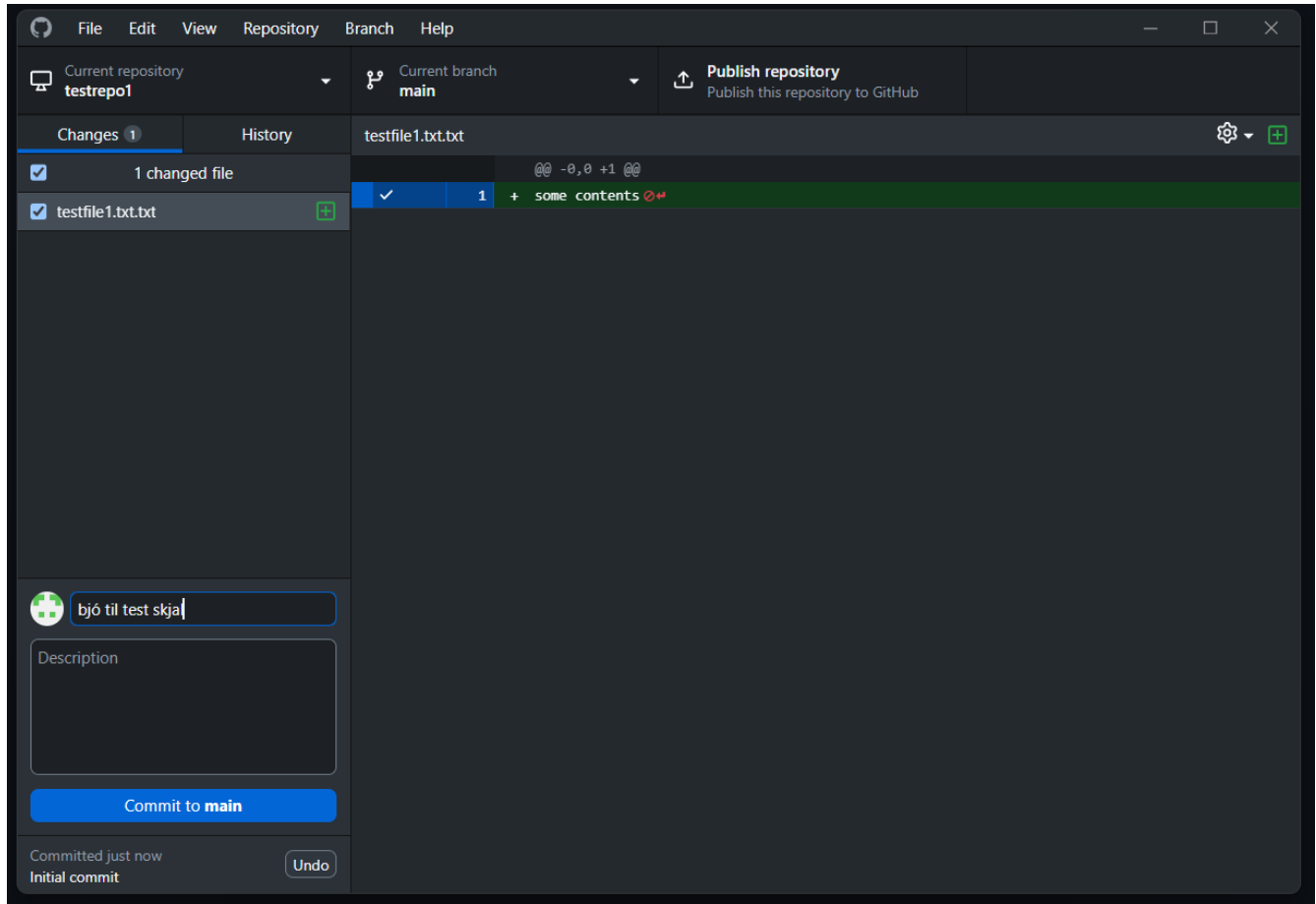


Síðan smellir maður annað hvort á geymsluna í listanum (fyrir GitHub.com) eða límir inn slóð á geymsluna (fyrir URL) og smellir á “Clone”.

Setja inn breytingu

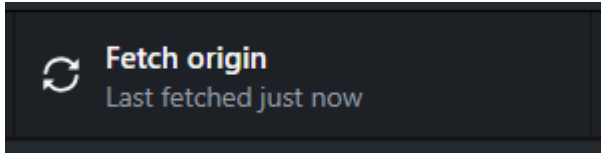
Þegar maður hefur unnið í einhverri skrá í verkefninu þá ætti GitHub Desktop að skynja að það sé búið að breyta einhverju í geymslunni. Þá getur maður farið í “Changes” flipann í aðalglugganum og séð yfirlit yfir þær skrár sem hafa verið gerðar breytingar á. Með því að smella á hverja skrá er hægt að sjá nánar hverju var breytt í henni.

Neðan við skráalistann er svo hægt að slá inn lýsingu á breytingunum og setja breytinguna inn á þá grein sem er virk með því að smella á “Commit to [grein]” takkann neðst.



Sækja breytingar

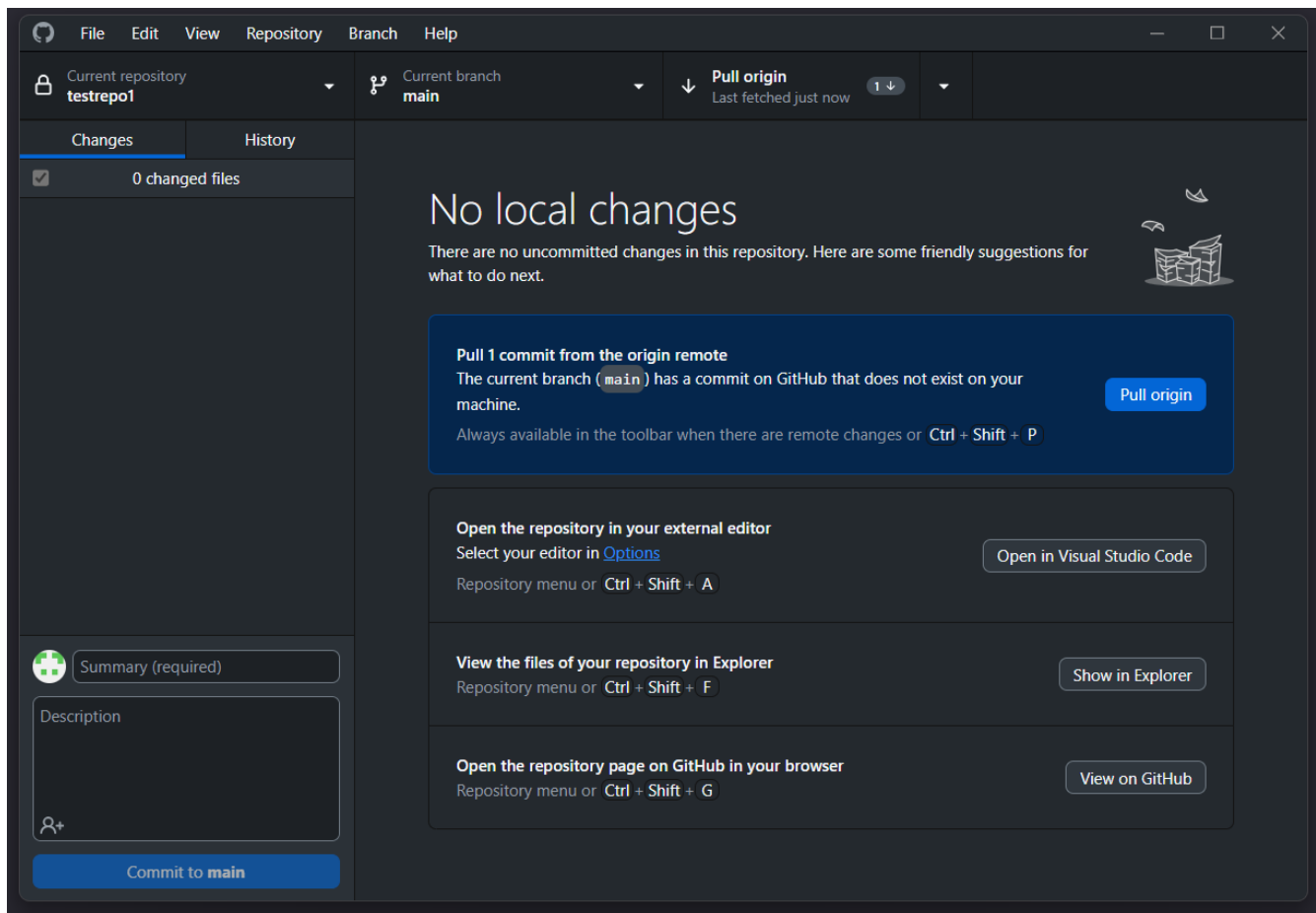
Áður en maður setur inn breytingar þá getur verið gott að sækja fyrst breytingar sem aðrir hafa mögulega gert á remote geymslunni. Það er hægt að gera með því að smella á “Fetch” takkann efst í GitHub Desktop glugganum.



Þessi aðgerð sækir bara þær breytingar sem eru til en sameinar þær ekki inn í skrárnar fyrr en maður gerir “pull”.

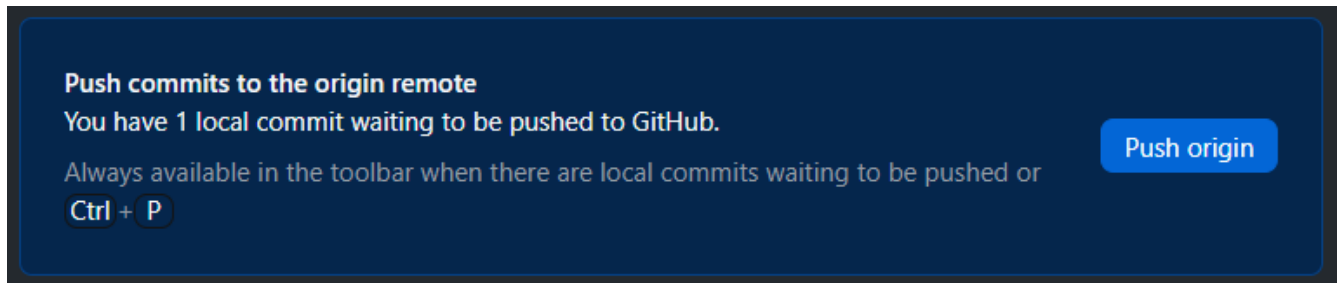
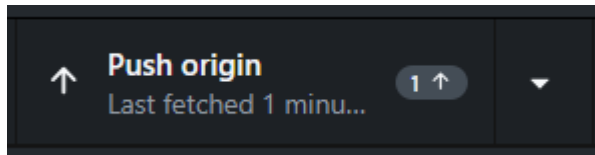
Pulla breytingar

Þegar maður er búinn að sækja breytingar með “fetch” skipuninni þá þarf að gera “pull” til þess að breytingarnar færist inn í skrárnar í geymslunni hjá manni. Það er hægt að gera með því að ýta á “Pull” takka, annað hvort í verkefnastikunni efst eða hægra megin í aðalglugga forritsins.



Pusha á remote

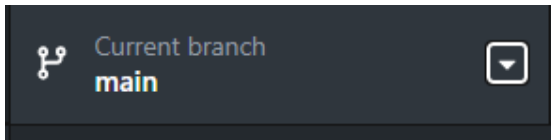
Þegar breyting hefur verið gerð og sett inn í geymsluna á drifinu hjá manni þá getur maður sent uppfærðu geymsluna í remote geymsluna með því að velja “Push” annað hvort í verkefnastikunni efst eða vinstra megin í aðalglugganum.



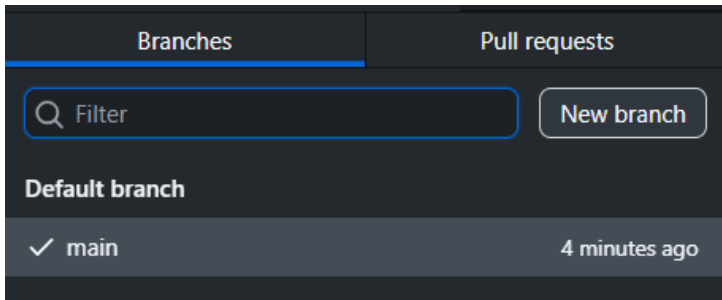
Þegar því er lokið þá eru breytingarnar komnar inn í remote geymsluna sem gerir samstarfsfólki kleift að sækja þær með því að gera fetch/pull hjá sér.

Stofna grein

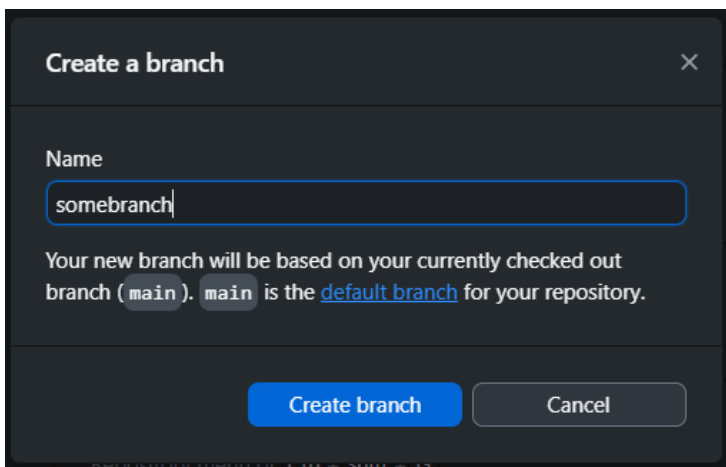
Til að stofna nýja grein er hægt að smella á “Current branch” takkan hægra megin:



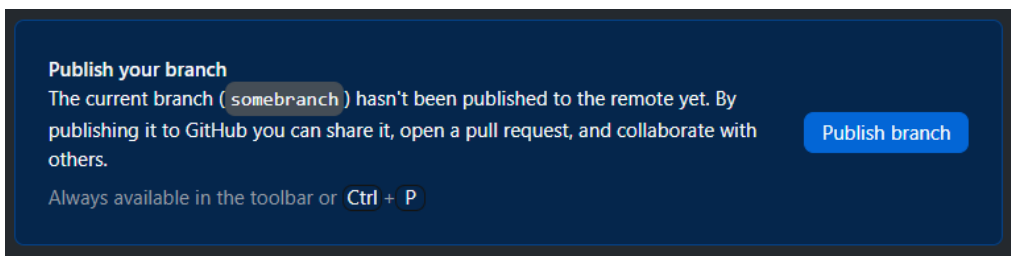
Þá opnast listi yfir þær greinar sem eru þegar til í geymslunni. Þar smellir maður á “New branch” takkann:



Þá opnast gluggi þar sem maður slær inn nafnið á nýju greininni og ýtir á “Create branch”:

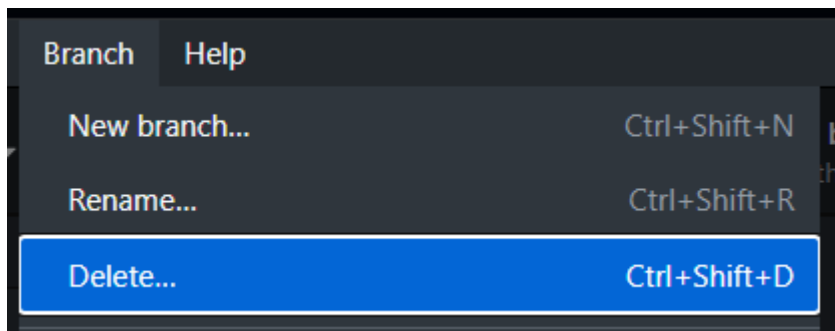


Þá verður til grein í geymslunni á drifinu hjá manni sem maður getur unnið með. GitHub Desktop skptir sjálfkrafa á hana þegar hún er stofnuð. Þegar maður er tilbúinn til þess þá getur maður valið “Publish branch” til að senda greinina í remote geymsluna en það er ekki nauðsynlegt, stundum gæti maður viljað vinna með einhverjar breytingar í grein sem er bara til á drifinu hjá manni sjálfum.

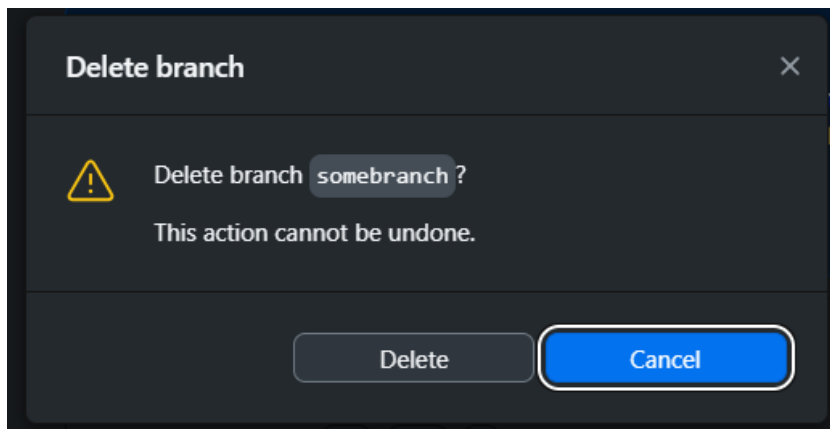


Eyða grein

Til að eyða grein sem er orðin óþörf er hægt að skipta á hana og fara svo í Branch → Delete eða ýta á Ctrl+Shift+D:



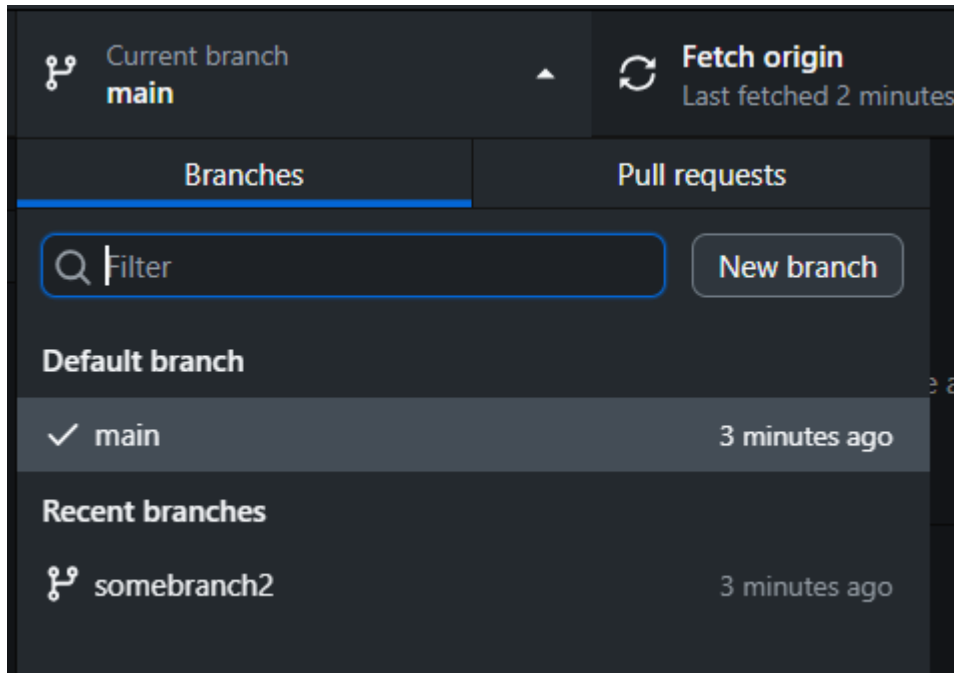
Þá opnast staðfestingargluggi þar sem maður þarf að smella á “Delete” til að greinin eyðist:



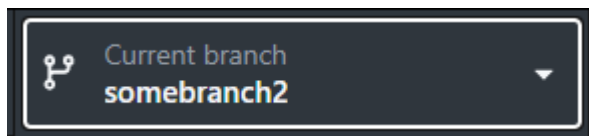
Hafa þarf í huga að þessi aðgerð eyðir eingöngu greininni af geymslunni sem er á drifinu hjá manni. Ef búið var að “pusha” greininni í remote geymslu þá er hún ennþá til staðar í henni og það þarf að eyða henni sérstaklega þar ef þörf er á því.

Skipta um grein

Til að skipta yfir á aðra grein þá smellir maður á “Current branch” takkann vinstra megin í viðmótinu og smellir svo á þá grein sem maður vill nota úr listanum sem kemur upp:

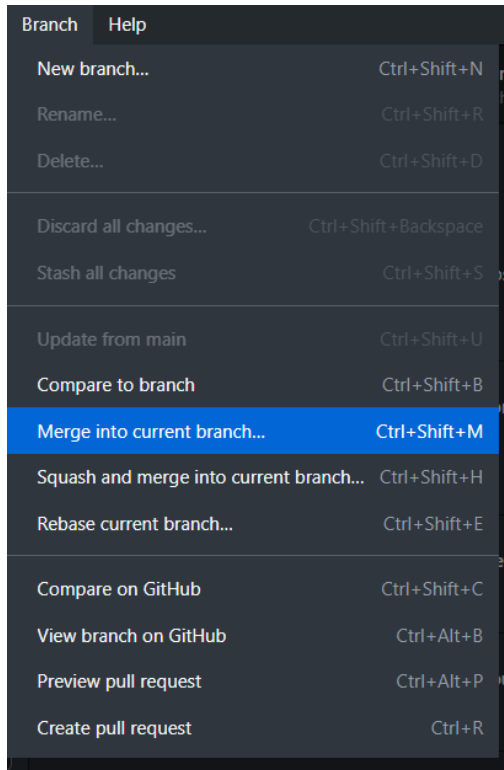


Þá skiptir GitHub Desktop yfir á þá grein og nafnið undir “Current branch” uppfærist:

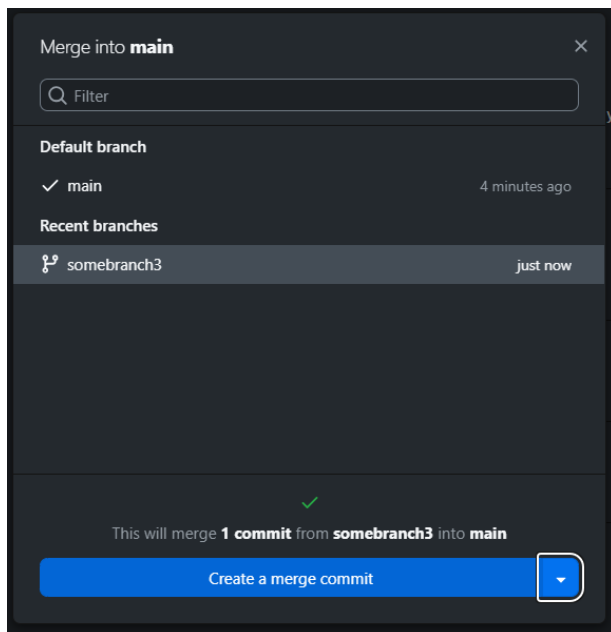


Sameina grein

Til að sameina einhverja grein inn á aðra grein þá þarf fyrst að skipta yfir á greinina sem á að taka við breytingunum. Svo velur maður Branch → Merge into current branch eða ýta á Ctrl+Shift+M.



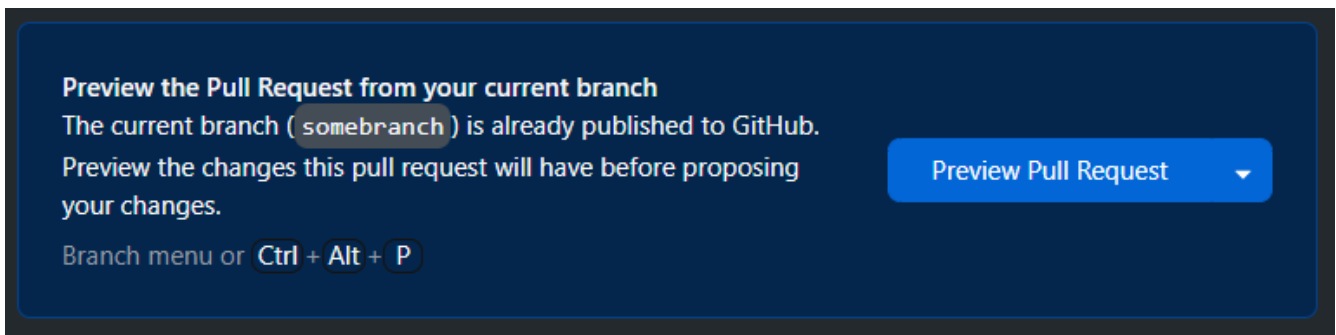
Síðan smellir maður á greinina sem inniheldur breytingarnar sem maður vill sameina og smellir á “Create a merge commit”:



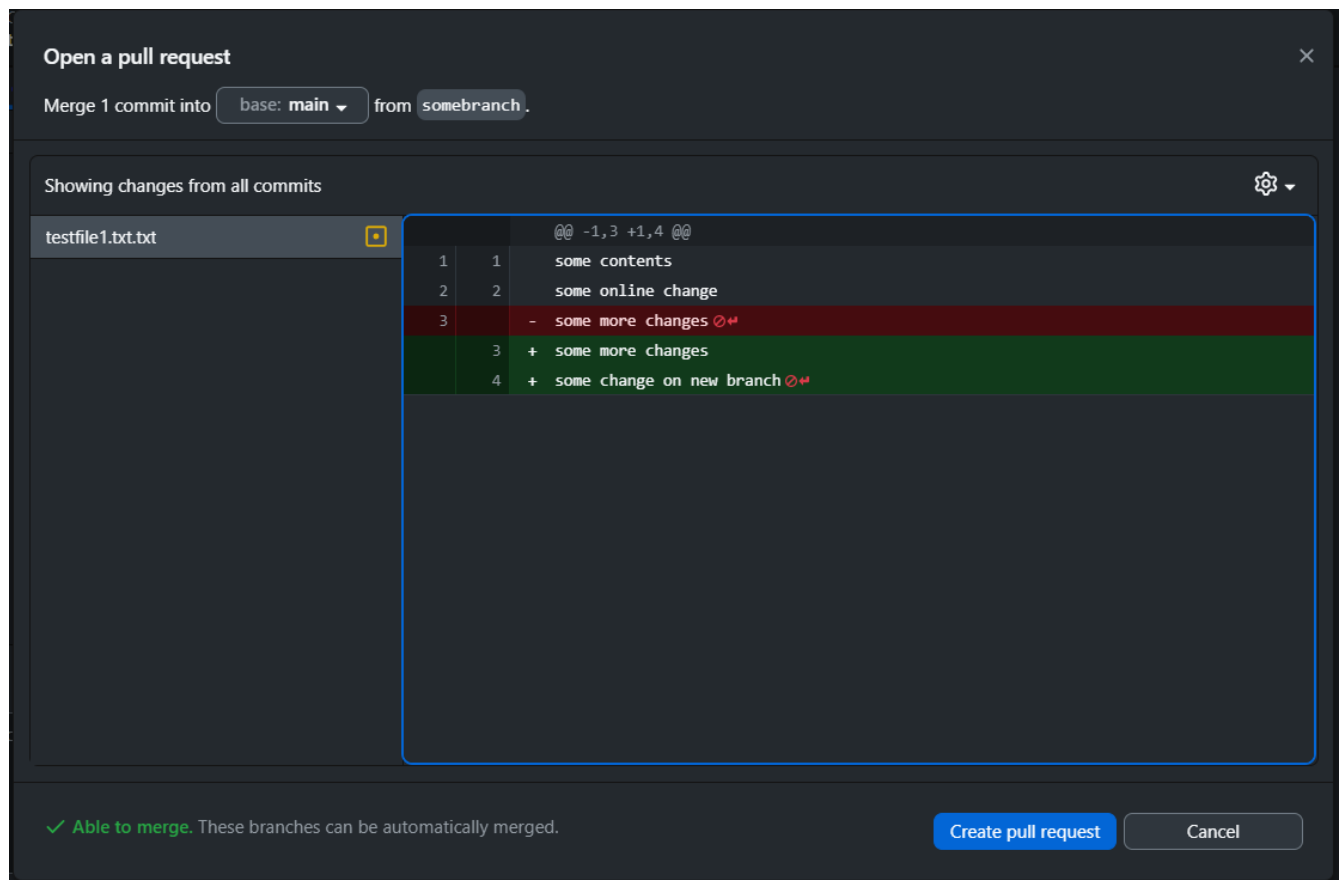
Stofna pull request

Þegar búið er að stofna nýja grein og færa inn breytingar á hana þá er hægt að senda hana inn á GitHub geymsluna með því að smella á “Publish branch”. Þegar greinin er komin á GitHub þá er hægt að stofna Pull Request til að biðja um að breytingarnar á greininni séu sameinaðar inn í aðra grein eins og t.d. master eða main.

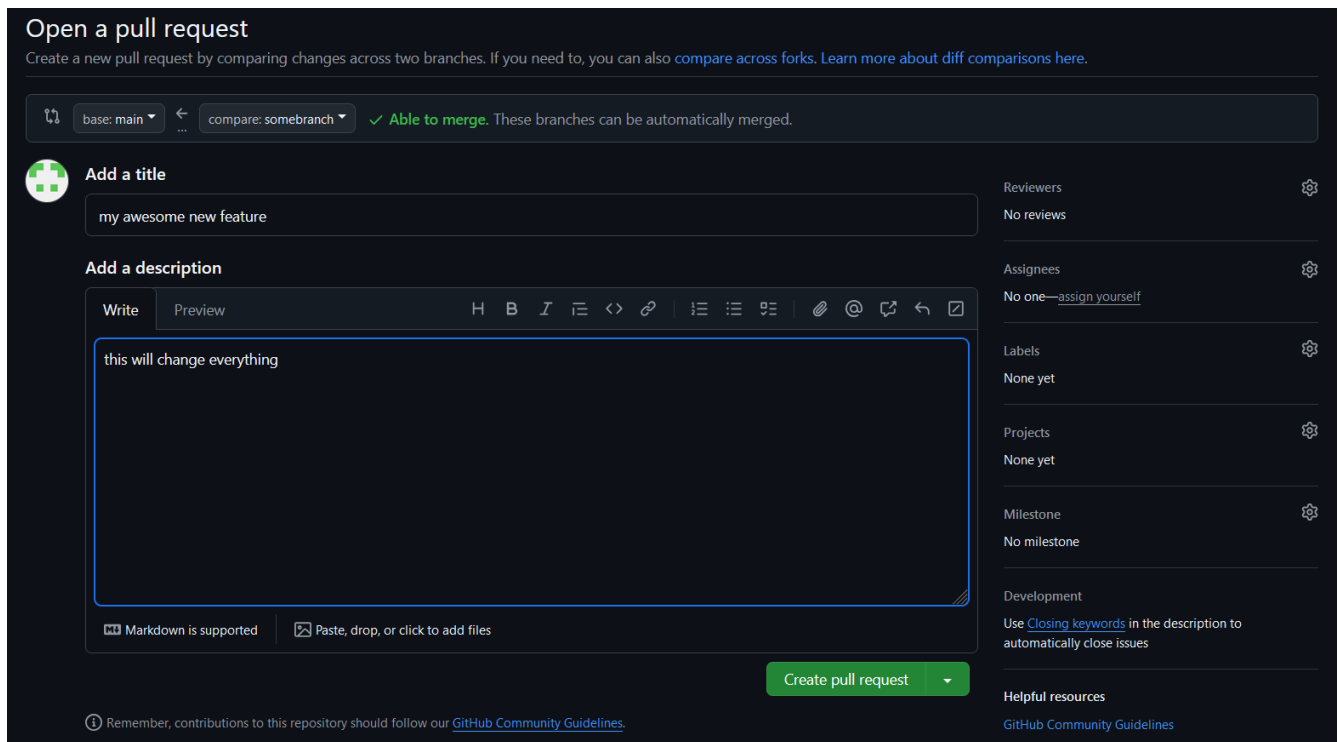
Í GitHub Desktop er hægt að stofna Pull Request með því að smella á “Preview Pull Request” takkann hægra megin í aðalglugga forritsins:



Þá opnast gluggi þar sem maður sér yfirlit yfir breytingarnar:

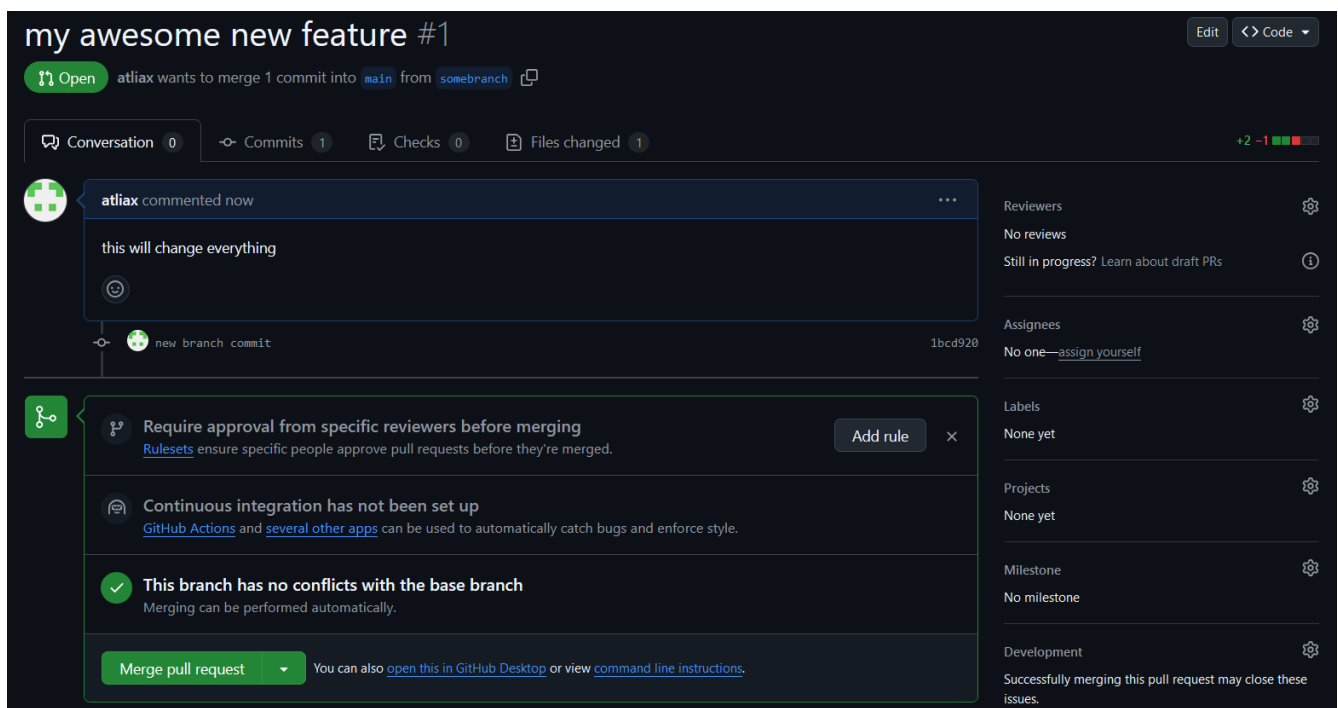


Þá smellir maður á “Create Pull Request” sem sendir mann yfir í vafraglugga þar sem GitHub opnast og maður slær inn titil, lýsingu og smellir á “Create Pull Request”:



The screenshot shows the GitHub interface for creating a pull request. At the top, it says "Open a pull request" and provides a brief explanation. Below this, there's a section for selecting the base branch (main) and the compare branch (somebranch), with a green checkmark indicating they are "Able to merge". The main form has two sections: "Add a title" and "Add a description". The title field contains "my awesome new feature". The description field contains "this will change everything". To the right of the form, there are several settings: Reviewers (No reviews), Assignees (No one—assign yourself), Labels (None yet), Projects (None yet), Milestone (No milestone), and Development (Use Closing keywords in the description to automatically close issues). At the bottom right, there's a green button labeled "Create pull request".

Þá verður til Pull Request á GitHub fyrir þessar breytingar þar sem fólk getur átt umræðu um breytingarnar og eigandi verkefnisins getur samþykkt eða hafnað beiðninni.

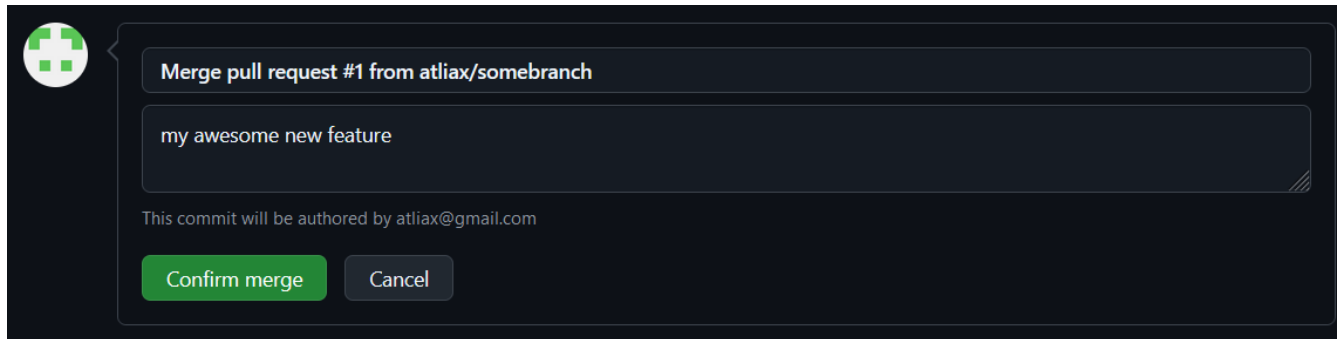


The screenshot shows the GitHub pull request page for "my awesome new feature #1". At the top, it says "my awesome new feature #1" and "atliax wants to merge 1 commit into main from somebranch". Below this, there's a section for "Conversation" with 0 comments, "Commits" with 1 commit, "Checks" with 0 checks, and "Files changed" with 1 file changed. The main content area shows a comment from "atliax" saying "this will change everything". Below the comment, there's a section for "Checks" with three items: "Require approval from specific reviewers before merging", "Continuous integration has not been set up", and "This branch has no conflicts with the base branch". At the bottom, there's a green button labeled "Merge pull request".

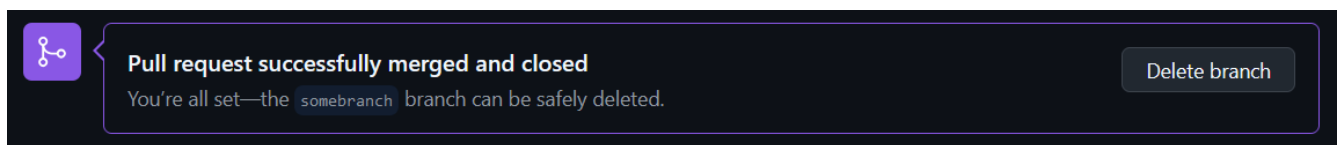
Samþykkja pull request

Þegar búið er að stofna Pull Request á GitHub getur eigandi verkefnisins farið inn í hana á GitHub og smellt á “Merge Pull Request” takkann til að samþykkja breytingarnar og sameina þær inn á þá grein sem breytingarnar eiga við (sjá öftustu mynd undir “Stofna Pull Request”).

Þá opnast staðfestingargluggi þar sem þarf að smella á “Confirm merge”:



Ef allt gekk vel ætti að koma upp eftirfarandi staðfesting:



Ef það á við er hægt að ýta á “Delete branch” þarna til að eða greininni af GitHub sem breytingarnar í Pull Requestinu komu frá.

Loka/hafna pull request

Ef þeim sem sér um GitHub verkefnið líst ekki vel á breytingarnar í einhverju Pull Request getur hann farið inn í það á GitHub og scrollað neðst þar sem athugasemdir eru settar inn og smellt á “Close pull request” til að loka/hafna því Pull Request:

