

<https://guides.github.com/>

A GitHub-folyamat megértése

A GitHub flow egy könnyű, branch alapú munkafolyamat, amely támogatja azokat a csapatokat és projekteket, ahol rendszeresen végeznek deployment-et. Ez az útmutató elmagyarázza, hogyan és miért működik a GitHub flow.

Hozzon létre egy branch-et

Amikor egy projekten dolgozik, akkor egy csomó különféle funkció vagy ötlet van folyamatban bármikor - amelyek közül néhány készen áll, mások pedig még nem. Branch létezik ennek a munkafolyamatnak a kezelésében.

Amikor létrehoz egy branch-et a projektben, olyan környezetet hoz létre, ahol új ötleteket próbálhat ki. Az ágon (branch) végrehajtott változtatások nem befolyásolják a fő ágot, így szabadon kísérletezhet és változtatásokat hajthat végre, biztonságban abban a tudatban, hogy branch addig nem egyesítik, amíg készen áll arra, hogy felülvizsgálja valaki, akivel együttműködik.

Profi tipp

A branch a Git egyik alapkoncepciója, és a teljes GitHub-folyamat ezen alapul. Csak egy szabály van: a fő ágon bármi mindig telepíthető.

Emiatt rendkívül fontos, hogy új ága a főn kívül kerüljön létrehozásra, amikor egy szolgáltatáson vagy javításon dolgozik. A branch leírónak kell lennie (pl. Refactor-authentication, user-content-cache-key, make-retina-avatars), hogy mások is láthassák, min dolgoznak.

Commit hozzáadása

Miután branch létrejött, ideje elkezdni a módosításokat. Valahányszor hozzáad, szerkeszt vagy töröl egy fájlt, commit-olja, és hozzáadja az ágához. A commit-ok hozzáadásának ez a folyamata nyomon követi az előrehaladást, amikor egy szolgáltatáságon dolgozik.

A commit-ok átlátható történelmet is készítenek munkádról, amelyet mások követhetnek, hogy megértsék, mit és miért tettél. Minden commit-hoz tartozik egy commit üzenet, amely leírja, hogy miért történt egy adott módosítás. Ezenkívül minden commit külön egységnek számít a változásban. Ez lehetővé teszi a változtatások visszavonását, ha hibát találnak, vagy ha más irányba dönt.

Profi tipp

A commit üzenetek fontosak, különösen azért, mert a Git nyomon követi a változtatásokat, majd a szerverre toláskor (push) végrehajtásként jeleníti meg őket. Világos commit üzenetek megírásával megkönnyítheti mások számára a követést és a visszajelzést.

Nyisson egy pull request-et

A Pull Requests kezdeményezi a megbeszélést az Ön commit-jairól. Mivel szorosan integrálódnak az alapul szolgáló Git-repóba, bárki láthatja, hogy pontosan milyen módosítások kerülnek egyesítésre, ha elfogadják a kérését.

A Pull Request a fejlesztési folyamat bármely pontján megnyitható: amikor kevés kódja van, vagy nincs kódja, de szeretne megosztani néhány képernyőképet vagy általános ötletet, amikor elakadt, segítségre vagy tanácsra van szüksége, vagy ha készen áll valaki hogy áttekintse a munkáját. Ha a GitHub @mention rendszerét használja a Pull Request üzenetben, visszajelzést kérhet bizonyos emberektől vagy csapatoktól, függetlenül attól, hogy a folyosón vannak, vagy tíz időzónával arrébb.

Profi tipp

A pull request kérelmek hasznosak a nyílt forráskódú projektekhez való hozzájáruláshoz és a megosztott táruk változásainak kezeléséhez. Ha Fork & Pull modellt használ, a Pull Requests lehetőséget nyújt arra, hogy értesítse a projekt karbantartóit azokról a változásokról, amelyeket figyelembe kíván venni. Ha megosztott adattár modellt használ, a Pull Requests segít megkezdeni a kód felülvizsgálatát és a javasolt változtatásokról folytatott beszélgetést, mielőtt beolvadnának a fő ágba.

Beszélje meg és ellenőrizze a kódot

A Pull Request megnyitása után a változtatásokat áttekintő személynek vagy csapatnak kérdései vagy észrevételei lehetnek. Lehet, hogy a kódolási stílus nem felel meg a projekt irányelveinek, a módosításból hiányoznak az egységtesztok, esetleg minden remekül néz ki, és a kellékek rendben vannak. A Pull Requests célja az ilyen típusú beszélgetések ösztönzése és megragadása.

Profi tipp

A Pull Request megjegyzéseket a Markdown-ban írják, így beágyazhatja a képeket és az emoji-kat, használhat előre formázott szövegblokkokat és más könnyű formázást.

Telepítés (Deploy)

A GitHub segítségével branch-ből deploy-olhatja, mielőtt egyesülne a main branch-el.

Miután a pull request-et felülvizsgálták, és az branch sikeresen teljesítette a tesztet, deploy-olhatja a módosításokat, hogy ellenőrizze azokat a production során. Ha a branch problémákat okoz, visszavonhatja azt

Különböző csapatok eltérő deployment stratégiákkal rendelkezhetnek. Egyesek számára a legjobb lehet egy speciálisan kiépített tesztelési környezetbe telepíteni. Mások számára a közvetlen production-ba állítás lehet a jobb választás a munkafolyamat többi eleme alapján.

Merge (egyesítés)

Most, hogy a production-ben ellenőrizte a változtatásokat, itt az ideje, hogy egyesítse a kódot a fő ágba.

Az egyesítés után a Pull Requests megőrzi a kód korábbi változásait. Mivel kereshetők, így bárkit visszamennek az időben, hogy megértsék, miért és hogyan született döntés.

Hello World

- Hozzon létre és használjon repository-t
- Új branch létrehozása és kezelése
- Végezzen módosításokat egy fájlban, és tegye őket a GitHub-ba, mint commit
- Nyissa meg és merge-je a pull request-eket

Mi az a GitHub?

A GitHub egy kódtárhely platform a verzióellenőrzéshez és az együttműködéshez. Ez lehetővé teszi, hogy Ön és mások bárholnan együtt dolgozhassanak a projekteken.

1. lépés: Hozzon létre egy repository-t

A repository általában egyetlen projekt megszervezésére szolgál. A repository-k mappákat és fájlokat, képeket, videókat, táblázatokat és adatkészleteket tartalmazhatnak - bármi, amire a projektnek szüksége van. Javasoljuk, hogy adjon hozzá egy README-t vagy egy fájlt, amely tartalmazza a projektre vonatkozó információkat. A GitHub megkönnyíti az egyik hozzáadását az új repository létrehozásakor. Más általános lehetőségeket is kínál, például egy licencfájlt.

A hello-world repository lehet olyan hely, ahol ötleteket, forrásokat tárolsz, vagy akár másokkal is megoszthatsz és megbeszélhetsz dolgokat.

Új repository létrehozása

- Kattintson a jobb felső sarokban az avatar vagy azonosító mellett, majd válassza az new repository lehetőséget.
- Nevezze el repository-t hello-world.
- Írjon rövid leírást.
- Válassza a repository inicializálása README-vel lehetőséget.

Owner: hubot / Repository name: hello-world ✓

Great repository names are short and memorable. Need inspiration? How about **petulant-shame**.

Description (optional): Just another repository

☒ Public: Anyone can see this repository. You choose who can commit.

☐ Private: You choose who can see and commit to this repository.

☒ Initialize this repository with a README: This will allow you to `git clone` the repository immediately. Skip this step if you have already run `git init` locally.

Add .gitignore: None | Add a license: None ⓘ

Create repository

2. lépés: Hozzon létre egy branch-et (ág)

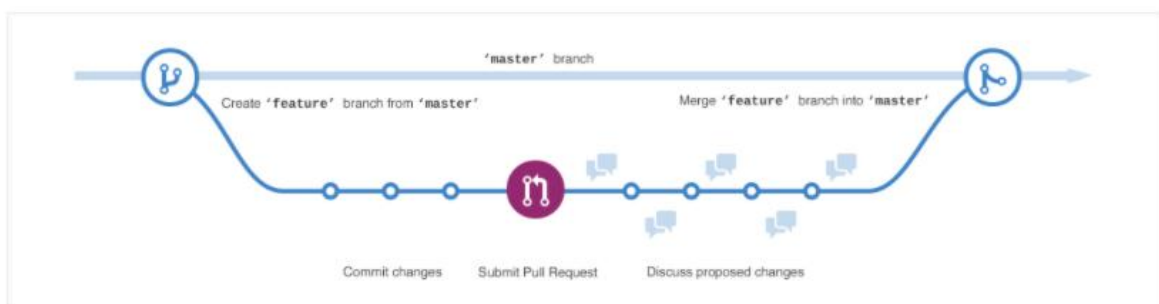
A branch a repository különböző verzióinak egyszerre történő kezelésének módja.

Alapértelmezés szerint a repository-nak van egy ága, amelynek neve main, amelyet végleges ágnak tekintenek. Elágazásokkal kísérletezünk és szerkesztünk, mielőtt elköteleznénk őket a main-el.

Amikor létrehoz egy elágazást a fő ágról, akkor másolatot vagy pillanatképet készít a főről, amint az abban az időpontban volt. Ha valaki más változtatott a fő ágon, miközben Ön a repository-n dolgozott, akkor pull-ozhatja ezeket a frissítéseket.

Ez a diagram a következőket mutatja:

- A fő ág
- Egy új ág, az úgynevezett szolgáltatás (mert ezen a fiókon „funkciómunkát” végzünk)
- Az az út, amelyet a szolgáltatás megtesz, mielőtt beolvadna a főbe



Mentett már valaha egy fájl különböző verzióit? Valami hasonló:

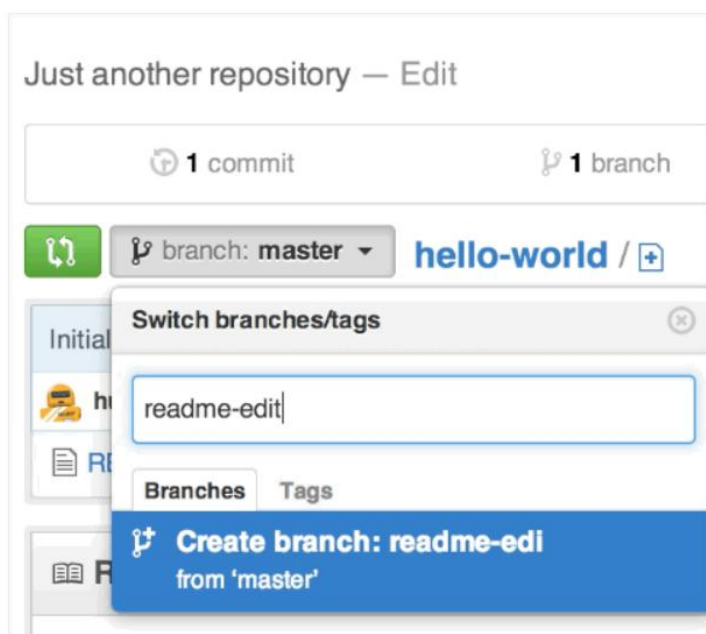
- story.txt
- story-joe-edit.txt
- story-joe-edit-reviewed.txt

Az ágak hasonló célokat érnek el a GitHub repóban.

Itt, a GitHub-nál fejlesztőink, íróink és tervezőink ágakat használnak a hibajavítások és a funkciómunkák elkülönítéséhez a fő (gyártási) ágtól. Amikor a változás kész, összeolvasztják águkat a főbe.

Új branch létrehozása

- Menj az new repository hello-world-re.
- Kattintson main feliratot tartalmazó fájllista tetején található legördülő menüre.
- Írja be a branch nevét, a readme-edits parancsot az új branch szövegmezőjébe.
- Jelölje ki a kék Create branch mezőt, vagy nyomja meg az Enter billentyűt.



Most két ága van, a fő és a readme-edits. Pontosan ugyanúgy néznek ki, de nem sokáig! Ezután hozzáadjuk a változtatásokat az új branch-hez.

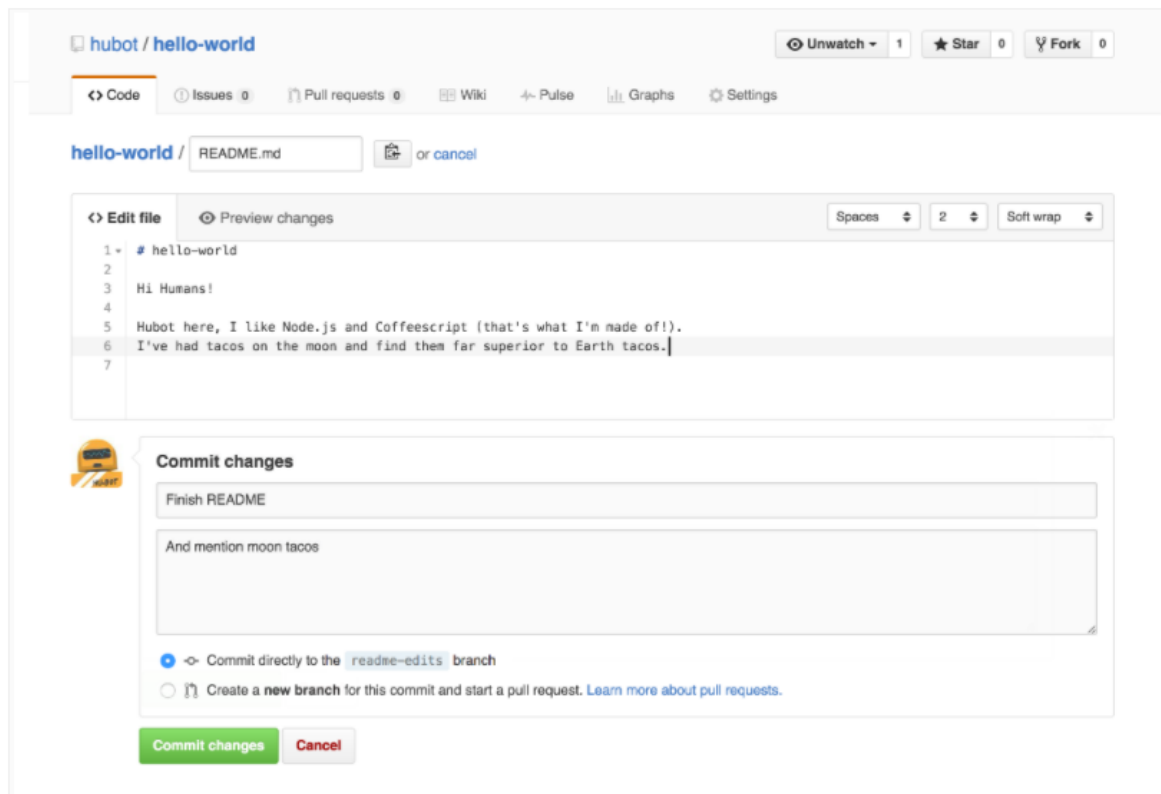
3. lépés. Végezze el és hajtsa végre a módosításokat

Bravó! Most a readme-edits ágának kódnézetében van, amely a main másolata. Végezzünk néhány módosítást.

A GitHubon a mentett módosításokat nevezzük commit-oknak. Minden commit-hoz tartozik egy commit üzenet, amely leírja, hogy miért történt egy adott módosítás. A Commit üzenetek rögzítik a változtatások előzményeit, így a többi közreműködő megértheti, hogy mit tett és miért.

Változásokat hajtson végre és commit-olja a változtatásokat

- Kattintson a README.md fájlra.
- Kattintson a ceruza ikonra a fájl nézet jobb felső sarkában a szerkesztéshez.
- A szerkesztőben írj egy kicsit róla.
- Írjon egy commit üzenetet, amely leírja a változtatásokat.
- Kattintson a Változtatások végrehajtása gombra.



Ezeket a módosításokat csak a readme-edits ág README fájlján hajtjuk végre, így most ez az ág a maintól eltérő tartalmat tartalmaz.

4. lépés: Nyisson meg a Pull kérést

Szép szerkesztések! Most, hogy változásokat hajtott végre a fő ágon kívül, megnyithat egy pull request-et.

A Pull Requests a GitHub együttműködésének középpontjában áll. Amikor pull request-et nyit meg, javasolja a változtatásokat, és azt kéri, hogy valaki nézze át, és vonja be közreműködését, és egyesítse őket fiókjába. A pull request-ek a két ág tartalmának eltéréseit vagy különbségeit mutatják. A változtatások, összeadások és kivonások zöld és piros színnel jelennek meg.

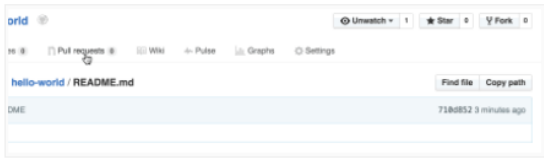
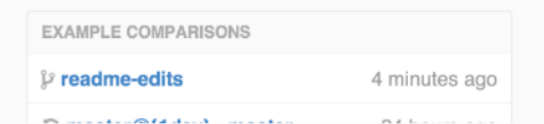
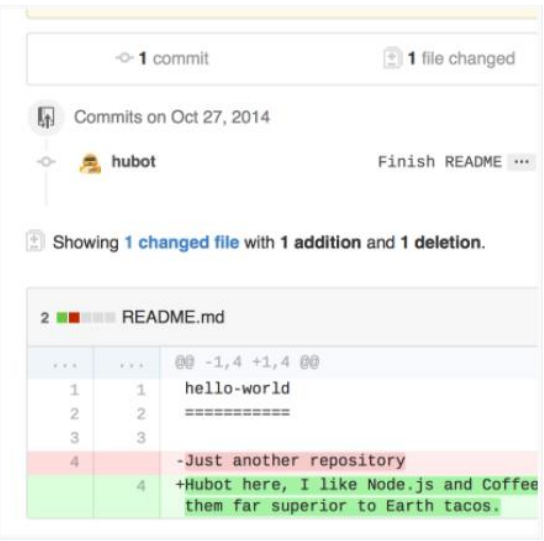
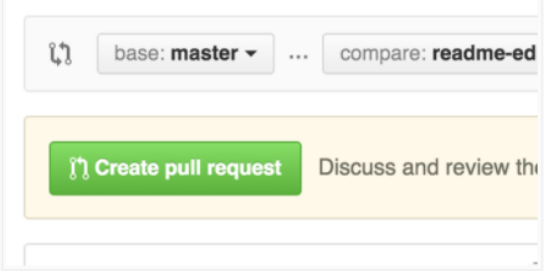
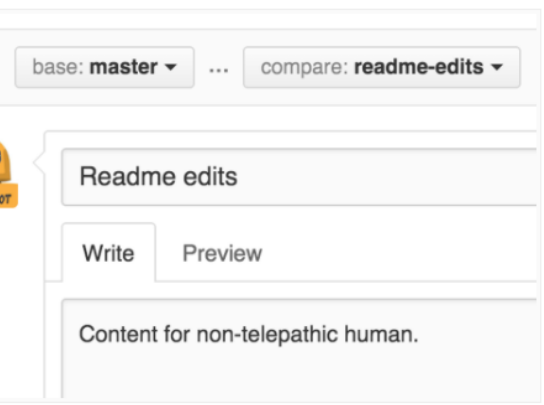
Amint commit-ot készít, megnyithatja a pull request-et és megkezdheti a beszélgetést, még a kód befejezése előtt.

A GitHub @mention rendszerének felhasználásával a pull request üzenetében visszajelzést kérhet bizonyos emberektől vagy csapatoktól, függetlenül attól, hogy a csarnokban vannak, vagy 10 időzónától távolabb.

Akár nyithat pull request-et a saját repository-jában, és egyesítheti azokat. Ez egy nagyszerű módja a GitHub-folyamat megismerésének, mielőtt nagyobb projekteken dolgozna.

Open a Pull Request for changes to the README

Click on the image for a larger version

Step	Screenshot
Kattintson a Pull Request fülre, majd a Pull Request oldalon kattintson a zöld New pull request gombra.	
Example Comparisons mezőben válassza ki a létrehozott ágot, a readme-edits szerkesztését, hogy összehasonlítsa a fővel (az eredetivel).	
Nézze át a változásokat az Compare oldalon található eltérésekben, és győződjön meg arról, hogy ezeket el szeretné-e küldeni.	
Ha meggyőződött arról, hogy ezeket a módosításokat el szeretné küldeni, kattintson a nagy zöld Create Pull Request gombra.	
Adjon címet a pull request-nek, és írjon rövid leírást a változásokról.	

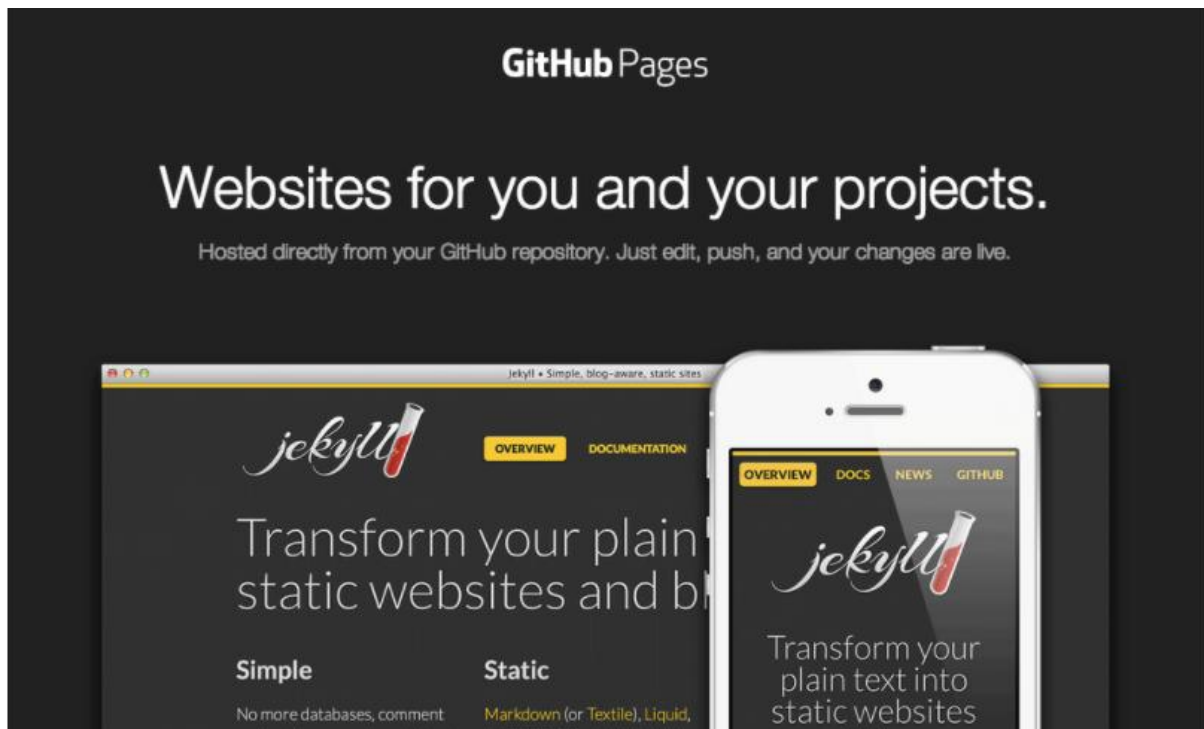
5. lépés: Egyesítse (merge) a Pull request-et

Ebben az utolsó lépésben itt az ideje, hogy összefogja a változtatásokat - a readme-edits ág összevonása a fő ágba.

- Kattintson a zöld Merge pull request relem gombra a változtatások man-be egyesítéséhez.
- Kattintson az Confirm merge gombra.
- Folytassa, és törölje az ágot, mivel a módosítások be lettek építve, a lila mezőben kattintson a Delete branch gombra.

Az első lépések a GitHub oldalakkal

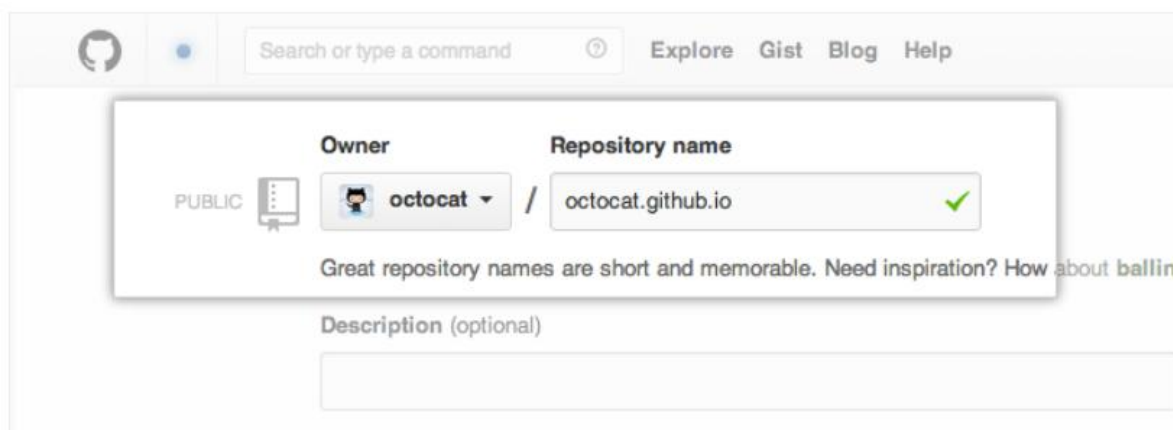
A GitHub oldalak nyilvános weboldalak, amelyeket a GitHubon keresztül tárolnak és egyszerűen közzétesznek. A leggyorsabb módja annak, hogy elinduljon, ha a Jekyll Theme Chooser-t használja egy előre elkészített téma betöltésére. Ezután távolról módosíthatja a GitHub Oldalak tartalmát és stílusát az interneten keresztül vagy lokálisan a számítógépén.



Készítse el webhelyét



Miután bejelentkezett, létrehozza az új repository-t a kezdéshez.

Az új repository képernyőn külön nevet kell adnia ennek a repository-nak a webhely létrehozásához.



Search or type a command ⓘ Explore Gist Blog Help

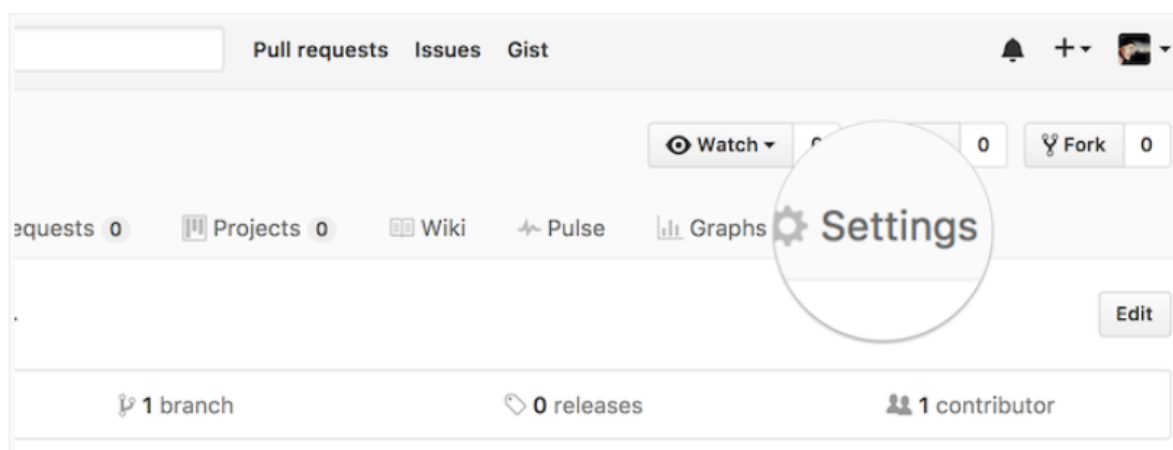
Owner **Repository name**

PUBLIC   octocat / octocat.github.io ✓

Great repository names are short and memorable. Need inspiration? How about ballin-

Description (optional)

Webhelyének fájljai a felhasználónév.github.io nevű adattárban fognak élni (ahol a „felhasználónév” a tényleges GitHub felhasználói név). A webhely beállításának megkezdéséhez meg kell nyitnia a Beállítások fület



Ha lefelé görget a beállítási oldalon, akkor az alján megjelenik a GitHub Pages szakasz. Kattintson a Choose a theme gombra a webhely létrehozásának megkezdéséhez.

GitHub Pages

Your site is ready to be published at <https://octocat.github.io/>.

GitHub Pages is designed to host your personal, organization, or project pages from a GitHub repository.

Source

Your GitHub Pages site is currently being built from the `master` branch. [Learn more.](#)

master branch ▾

Save

User pages must be built from the `master` branch.

Theme chooser

Select a theme to build your site with a Jekyll theme. [Learn more.](#)

Choose a theme

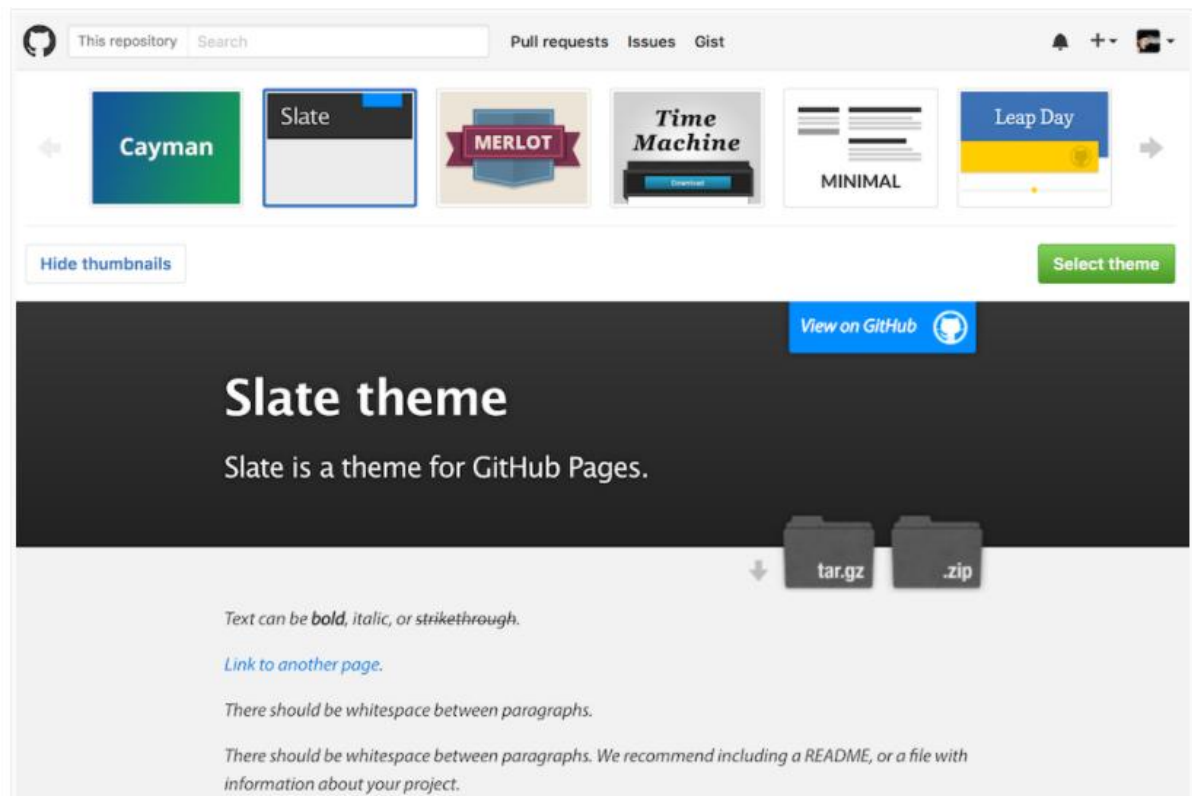
Custom domain

Custom domains allow you to serve your site from a domain other than `octotester-sj.github.io`. [Learn more.](#)

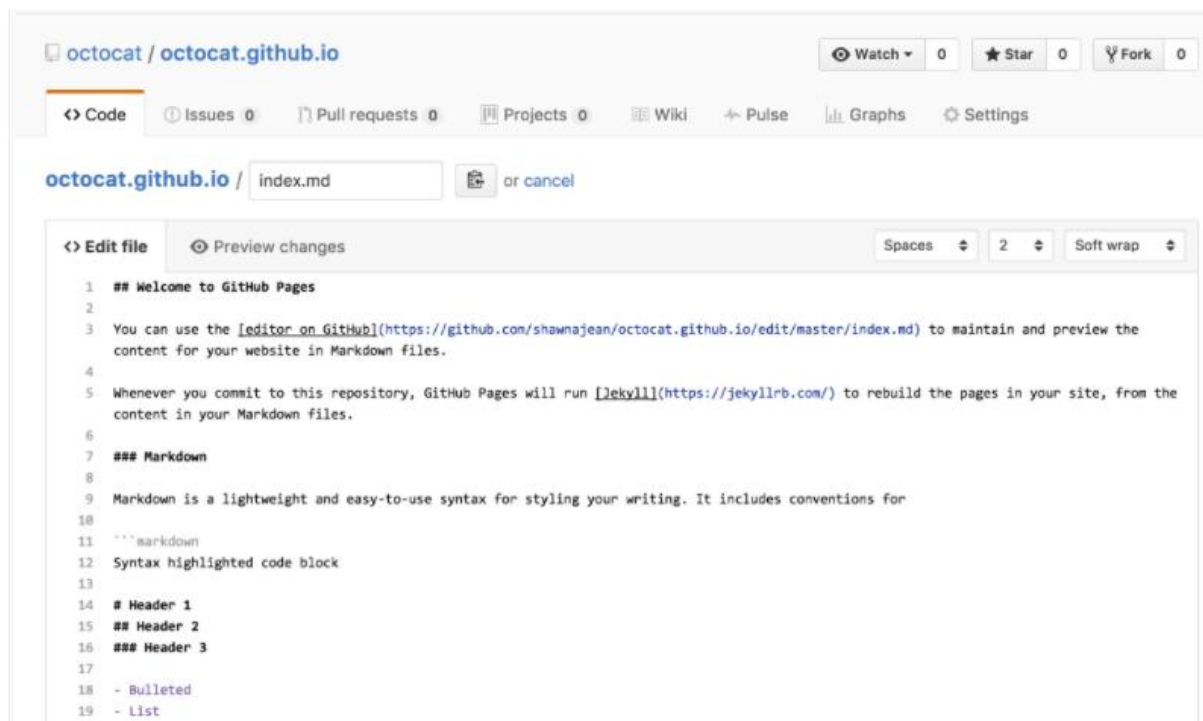
Save

☒ **Enforce HTTPS** — Required for your site because you are using the default domain (`octotester-sj.github.io`)
HTTPS provides a layer of encryption that prevents others from snooping on or tampering with traffic to your site.
When HTTPS is enforced, your site will only be served over HTTPS. [Learn more.](#)

Miután rákattintott a gombra, a Theme Chooser-hez irányítja. Számos témaopciót láthat az oldal tetején. Kattintson a képekre a témák előnézetének megtekintéséhez. Miután kiválasztott egyet, kattintson a jobb oldali Choose a theme gombra a továbblépéshez. Könnyű később megváltoztatni a témát, ezért ha nem biztos benne, akkor most csak válasszon egyet.



Itt írhatja meg saját tartalmát (egyelőre megtarthatja az alapértelmezett tartalmat, ha szeretné).




A szerkesztés befejezése után görgessen le az oldal aljára, és kattintson a Commit changes gombra.

file.

34

35 ### Support or Contact



Commit changes

Add content to new Pages site

Add an optional extended description...

☒ Commit directly to the `master` branch.

☐ Create a new branch for this commit and start a pull request. [Le](#)

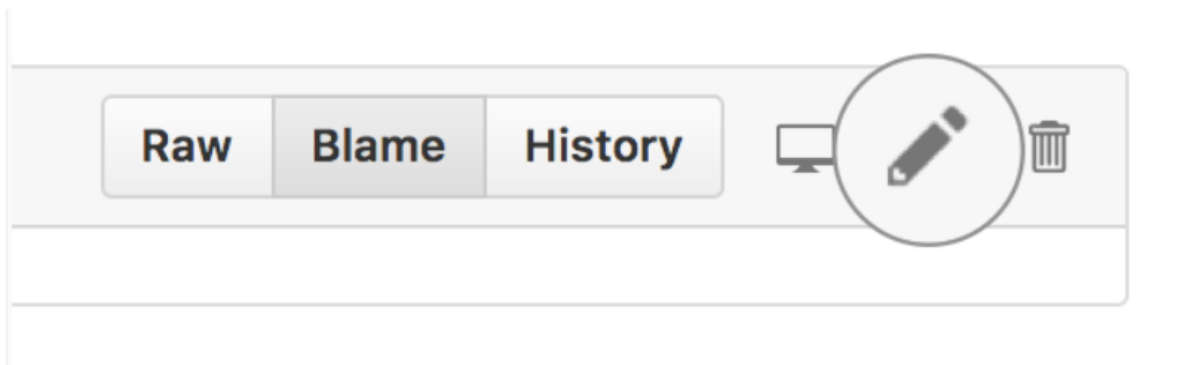
Commit changes

Cancel

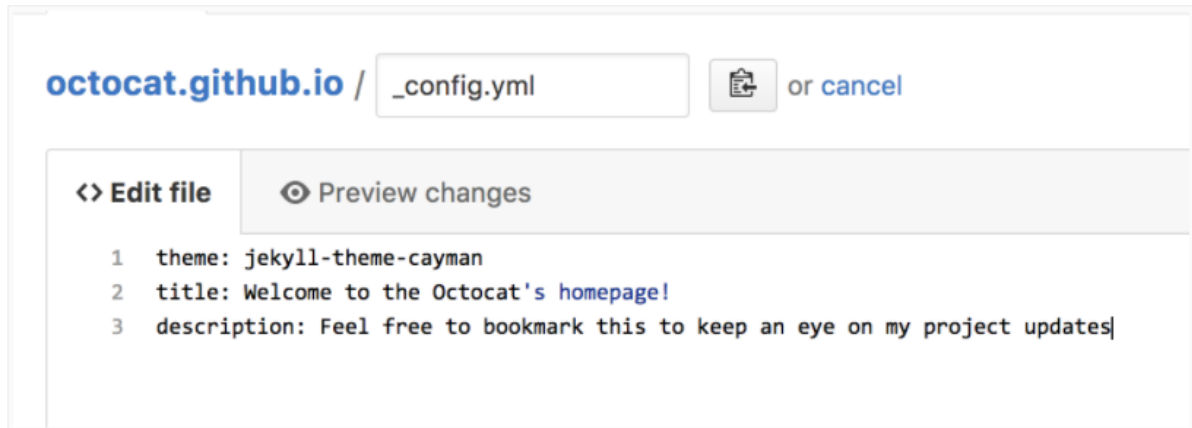
A GitHub mindent megtesz annak érdekében, hogy a látogatókat a felhasználónév.github.io webhelyre irányítsa, hogy megtekinthesse új webhelyét. Ez akár 10 percet is igénybe vehet. Egy idő elteltével megnyithat egy új fület a böngészőjében, hogy felkeresse webhelyét!

Változtatások

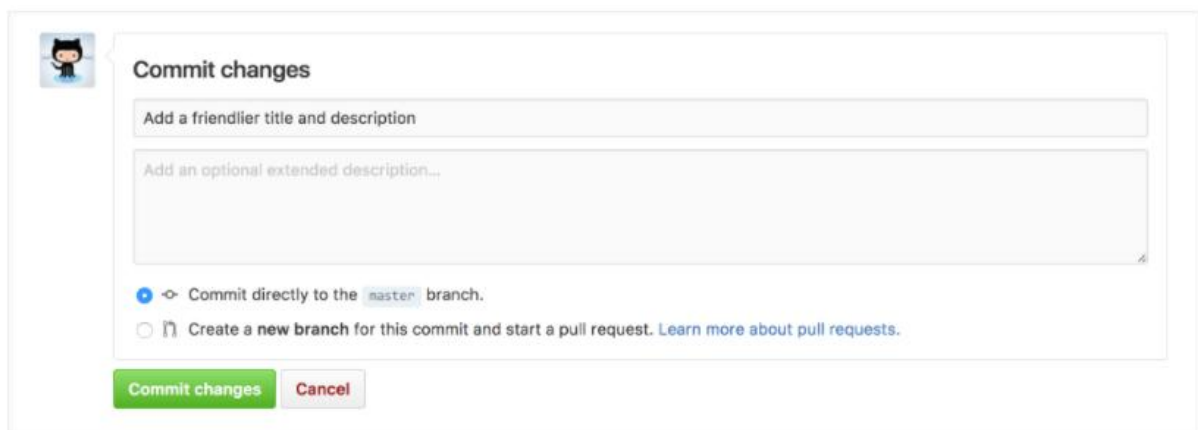
Az első dolgok közé tartozik az, hogy eltávolítja az indexlap alapértelmezett címét, és hozzáad egy barátságosabb üzenetet. Mivel ez egy nagyon gyors változás - és az első -, az alapértelmezett ágon: main. Tekintse meg a `_config.yml` fájlt a Code fülön navigálva. A ceruza ikonra kattintva szerkesztheti a fájlt.



Jelenleg webhelyének nincs meghatározva címe, ezért visszatérünk a repó nevére. Ennek megváltoztatásához hozzáadom a „cím: Üdvözljük az Octocat honlapján!” Sort. ehhez a fájlhoz. Nyugodtan tegye ugyanezt, kivéve a saját felhasználónevét. E cím alatt hozzáadhat egy üzenetet az oldal céljáról, és leírhatja, mit szeretne tenni az embereknek, amíg itt vannak. Az enyémet a következőre állítom: "Nyugodtan jelölje meg ezt könyvjelzővel, hogy szemmel tarthassa a projekt frissítéseimet"



Miután végzett ezzel a kis változtatással, görgessen az oldal aljára a második commit végrehajtásához. Két helye van, hogy írjon erről a változásról: egy tárgy és egy kiterjesztett leírás. A kiterjesztett leírás nem kötelező, ezért hagyjunk egy leíró üzenetet a témában. Ha végzett, kattintson a Commit changes gombra, és a frissítései néhány másodpercen belül elérhetővé válnak!



Git kézikönyv

Mi a verziókezelő rendszer?

A verziókezelő rendszer vagy a VCS nyomon követi a változások történetét, miközben az emberek és a csapatok együtt dolgoznak a projektekben. A projekt fejlődésével a csapatok futtathatnak tesztet, kijavíthatják a hibákat, és új kódot adhatnak hozzá abban a bizalomban, hogy bármelyik verzió bármikor helyreállítható. A fejlesztők áttekinthetik a projekt előzményeit, és megtudhatják: Milyen változtatásokat hajtottak végre? Ki hajtotta végre a változtatásokat? Mikor történtek a változtatások? Miért volt szükség változtatásokra?

Mi az elosztott verziókezelő rendszer?

A Git egy példa egy elosztott verziókezelő rendszerre (DVCS), amelyet általában nyílt forráskódú és kereskedelmi szoftverek fejlesztésére használnak. A DVCS-k teljes hozzáférést biztosítanak a projekt minden fájljához, ágához és iterációjához, és minden felhasználó számára hozzáférést biztosítanak a változások teljes és önálló előzményéhez. Az egykor népszerű központosított verzióvezérlő rendszerektől eltérően az olyan DVCS-knek, mint a Git, nincs szükségük állandó kapcsolatra egy központi adattárral. A fejlesztők bárhol dolgozhatnak és aszinkron módon működhetnek együtt, bármilyen időzónából. Verzióellenőrzés nélkül a csapat tagjaira redundáns feladatok, lassabb ütemtervek és egyetlen projekt több példánya vonatkozik. A felesleges munka kiküszöbölése érdekében a Git és más VCS-k minden résztvevőnek egységes és következetes képet adnak a projektről, felszínre hozva a már folyamatban lévő munkát. A változások átlátható történetének megismerése, ki hajtotta végre őket, és hogyan járulnak hozzá a projekt fejlesztéséhez, segít a csapattagoknak az önálló munkavégzés során.

Miért Git?

A legfrissebb Stack Overflow fejlesztői felmérés szerint a fejlesztők több mint 70 százaléka használja a Git-et, ezzel a világon a leggyakrabban használt VCS. A Git-t gyakran használják nyílt forráskódú és kereskedelmi szoftverek fejlesztésére is, jelentős előnyökkel járva magánszemélyek, csapatok és vállalkozások számára. A Git segítségével a fejlesztők egy helyen láthatják a projektek változásainak, döntéseinek és előrehaladásának teljes ütemtervét. Attól a pillanattól kezdve, hogy hozzáférnek a projekt történetéhez, a fejlesztők megvan az összes olyan kontextus, amelyre szüksége van ahhoz, hogy megértse és elkezdhesse a hozzájárulást. A fejlesztők minden időzónában dolgoznak. A Git-hez hasonló DVCS esetén az együttműködés bármikor megtörténhet a forráskód integritásának fenntartása mellett. Az ágak segítségével a fejlesztők biztonságosan javasolhatják a gyártási kód módosítását. A Git-et használó vállalkozások lebontják a kommunikációs akadályokat a csapatok között, és koncentrálnak arra, hogy a legjobb munkát végezzék. Ráadásul a Git lehetővé teszi a szakértők összehangolását az üzleti életben, hogy együttműködjenek a nagyobb projekteknél.

Mi az a repository?

A repository vagy a Git projekt felöleli a projekthez társított fájlok és mappák teljes gyűjteményét, az egyes fájlok felülvizsgálati előzményeivel együtt. A fájl-előzmények pillanatfelvételekként jelennek meg az úgynevezett commit-ok között, a commit-ok pedig összekapcsolt lista-kapcsolatokként léteznek, és több fejlesztési sorba szervezhetők, ágaknak hívva. Mivel a Git egy DVCS, a repository-k önálló egységek, és bárki, aki a repository másolatát birtokolja, hozzáférhet a teljes kódalaphoz és annak történetéhez. A parancssor vagy más, egyszerűen használható interfészek használatával a git-repository lehetővé teszi az alábbiakat is: interakció az előzményekkel, klónozás, ágak létrehozása, commit, merge, a változatok összehasonlítása a kód verziói között és még sok más.

A repository-kban történő munkavégzés fenntartja és védi a fejlesztési projekteket. A fejlesztőket arra ösztönzik, hogy javítsák ki a hibákat, vagy hozzanak létre új szolgáltatásokat, félelem nélkül a fő fejlesztési erőfeszítések kisiklásától. A Git ezt megkönnyíti a branch-ek használatával: könnyű hivatkozások a commit-okra, amelyek könnyen létrehozhatók és megszüntethetők, amikor már nincs rá szükség. A GitHubhoz hasonló platformokon keresztül a Git több lehetőséget kínál a projektek átláthatóságára és együttműködésére. A nyilvános repository-k segítik a csapatokat a lehető legjobb végtermék elkészítésében.

Basic Git parancsok

A Git használatához a fejlesztők speciális parancsokat használnak a kód másolásához, létrehozásához, módosításához és kombinálásához. Ezeket a parancsokat közvetlenül a parancssorból vagy egy olyan alkalmazás segítségével hajthatjuk végre, mint a GitHub Desktop vagy a Git Kraken. Íme néhány általános parancs a Git használatához:

- `git init` inicializálja a vadonatúj Git-repository-t, és megkezdje a meglévő repository nyomon követését. Egy rejtett almappát ad hozzá a meglévő könyvtárba, amely a verziókezeléshez szükséges belső adatstruktúrát tartalmazza.
- `git clone` létrehoz egy helyi másolatot egy projektről, amely már távolról is létezik. A klón tartalmazza a projekt összes fájlját, előzményeit és ágait.
- `git add` A Git követi a fejlesztő kódbázisának változásait, de a változtatásokról szakaszra kell készíteni és pillanatfelvételt kell készíteni, hogy bekerüljenek a projekt történelmébe. Ez a parancs staging-et hajt végre, ennek a kétlépcsős folyamatnak az első részét. Minden változás a következő pillanatkép és a projekt történetének része lesz. A külön staging és commit a fejlesztésnek teljes ellenőrzést biztosít a projektjük története felett, anélkül, hogy megváltoztatnák a kódolás és a működés módját.
- `git commit` elmenti a pillanatképet a projektelőzményekbe, és befejezi a változáskövetési folyamatot. Röviden, a commit úgy működik, mint egy fénykép..
- `git status` változások követése untracked, modified, or staged.
- `git branch` branch-eket mutat, amelyen lokálisan dolgozunk
- `git merge` egyesíti a fejlődési vonalakat (branch). Ezt a parancsot általában két külön ágon végrehajtott változtatások kombinálására használják. Például, ha egyes funkcióágakból a főágakba kívánja változtatni a telepítést.
- `git pull` a helyi fejlesztési vonalat a távoli partner frissítéseivel. A fejlesztők akkor használják ezt a parancsot, ha egy csapattársa commit-olt egy távoli ágon, és ezeket a változásokat szeretnék tükrözni a helyi környezetben.
- `git push` frissíti a távoli repository-t minden olyan helyileg végrehajtott commit-al, amelyet egy branch-en hajtottak végre.

GitHub és a parancssor

Example: Contribute to an existing repository

```
# download a repository on GitHub.com to our machine
git clone https://github.com/me/repo.git

# change into the `repo` directory
cd repo

# create a new branch to store any new changes
```

```
git branch my-branch

# switch to that branch (line of development)
git checkout my-branch

# make changes, for example, edit `file1.md` and `file2.md` using the text
editor

# stage the changed files
git add file1.md file2.md

# take a snapshot of the staging area (anything that's been added)
git commit -m "my snapshot"

# push changes to github
git push --set-upstream origin my-branch
```

Example: Start a new repository and publish it to GitHub

```
# create a new directory, and initialize it with git-specific functions
git init my-repo

# change into the `my-repo` directory
cd my-repo

# create the first file in the project
touch README.md

# git isn't aware of the file, stage it
git add README.md

# take a snapshot of the staging area
git commit -m "add README to initial commit"

# provide the path for the repository you created on github
git remote add origin https://github.com/YOUR-USERNAME/YOUR-REPOSITORY.git

# push changes to github
git push --set-upstream origin main
```

Example: contribute to an existing branch on GitHub

```
# assumption: a project called `repo` already exists on the machine, and a
new branch has been pushed to GitHub.com since the last time changes were
made locally

# change into the `repo` directory
cd repo

# update all remote tracking branches, and the currently checked out branch
git pull

# change into the existing branch called `feature-a`
git checkout feature-a
```



```
# make changes, for example, edit `file1.md` using the text editor

# stage the changed file
git add file1.md

# take a snapshot of the staging area
git commit -m "edit file1"

# push changes to github
git push
```

Az együttműködésen alapuló fejlesztés modelljei

Két fő módja van a GitHub együttműködésének:

Shared repository

Fork and pull

Megosztott adattár használatával az egyéneket és a csapatokat kifejezetten olvasó, író vagy rendszergazdai hozzáféréssel rendelkező közreműködőként jelölik meg. Ez az egyszerű engedélystruktúra olyan funkciókkal kombinálva, mint a védett ágak és a Marketplace, segít a csapatoknak a gyors előrehaladásban, amikor átvesszik a GitHub-ot.

Nyílt forráskódú projektek vagy olyan projektek esetében, amelyekhez bárki hozzájárulhat, az egyéni engedélyek kezelése kihívást jelenthet, de a fork and pull modell lehetővé teszi, hogy mindenki, aki megtekintheti a projektet, hozzájárulhat. A fork a fejlesztő személyes repository-jában lévő projekt másolata. Minden fejlesztő teljes mértékben ellenőrzi a fork-ját, és szabadon alkalmazhat egy javítást vagy új funkciót. A forkban végzett munkákat vagy külön tartják, vagy pull kérelem alapján visszavezetik az eredeti projektre. Ott a karbantartók áttekinthetik a javasolt módosításokat a merge előtt.

Be Social

Ha egyre többen csatlakoznak a GitHubhoz, és minden nap hozzáadnak projekteket, nehéz lehet lépést tartani mindegyikkel. Ez azonban szórakoztató és egyszerű lehet, ha követi a felhasználókat vagy megnézi a repository-kat, egyszerűen csak csillaggal jelzi érdeklődését irántuk, vagy az Explore segítségével új embereket és projekteket keres.

Kövessen egy barátot

A GitHub egyik nagyszerű tulajdonsága, hogy meg tudja nézni, hogy más emberek min dolgoznak és kivel állnak kapcsolatban. Ha követesz valakit, értesítést kapsz a GitHub tevékenységéről. A barátok követése segít új projekteket és új embereket találni, akikkel közösen érdeklődhet. Megtekintheti, hogy egyes ismerősei mit érdekelnek, ha megnézi a személyes irányítópultot, vagy megnézheti a GitHub közösségben zajló eseményeket a Felfedezés oldalon.

Watch a Project's Repository

Egy bizonyos ponton érdemes lehet naprakész lenni egy adott projekttel. Ez hasonló egy személy követéséhez, azzal a különbséggel, hogy a hangsúly csak a projekt adattárának eseményeire szűkül. A beállítások megadásával választhatja, hogy e repository

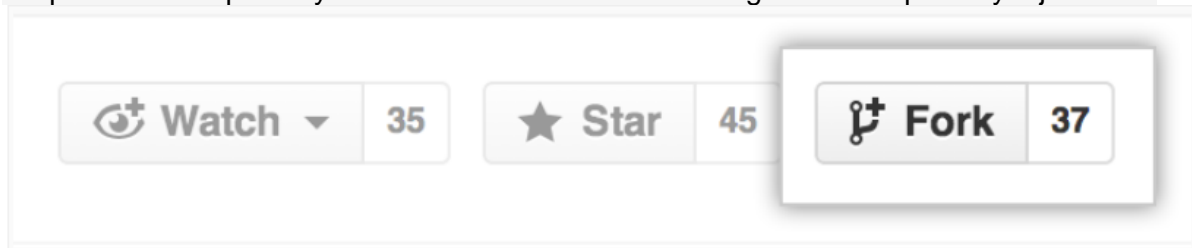
értesítéseket küldjön e-mailben, vagy az interneten nézze meg. Tipikus értesítések lehetnek a Pull Requesthez vagy egy kérdéshez fűzött megjegyzések, vagy csak egy megjegyzés a táron belül.

Forking Projects

Miután egy ideig egyedül használta a GitHub-ot, előfordulhat, hogy szeretne hozzájárulni valaki más projektjéhez. Vagy talán valaki projektjét szeretné használni a sajátjának kiindulópontjaként. Ez a folyamat elágazásként ismert.

A „fork” létrehozásával valaki más projektjének személyes másolata készül. A fork-ok egyfajta hídként működnek az eredeti repository és az Ön személyes másolata között. Nyújthat be kéréseket, hogy mások projektjei jobbak legyenek, ha felajánlja a változtatásokat az eredeti projektig. A fork a GitHub közösségi kódolásának középpontjában áll.

A Spoon-Knife repository fork-olásához kattintson a Fork gombra a repository fejlécében.



Klónozza a fork-ot

Sikeresen fork-olt a Spoon-Knife repository-n, de eddig csak a GitHubon létezik. Ahhoz, hogy dolgozni tudjon a projekten, klónoznia kell a számítógépére. A Spoon-Knife fork-on lépjen át a jobb oldali sávba, és kattintson a Klónozás vagy a Letöltés gombra. Hogy klónozol, rajtad múlik. Néhány opció klónozás a parancssorral vagy a GitHub Desktop használatával.

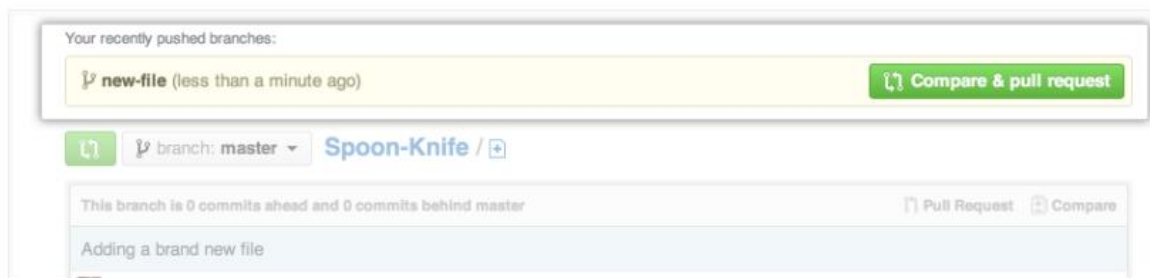
Változások végrehajtása és push-olása

Menjen előre, és végezzen néhány módosítást a projekten a kedvenc szövegszerkesztőjével, például az Atom segítségével. Megváltoztathatja például az index.html szövegét a GitHub felhasználónevének hozzáadásához. Ha készen áll a változások benyújtására, akkor commit-oljon.

Most lényegében azt mondtad Gitnek: „Oké, pillanatképet készítettem a változásaimról!” Folytathatja a további módosításokat, és több commit pillanatképet készíthet.

Making a Pull Request

Végre készen áll arra, hogy javaslatot tegyen a fő projekt módosítására! Ez az utolsó lépés egy másik project-jének elkészítésében, és vitathatatlanul a legfontosabb. Ha olyan változtatást hajtott végre, amelyről úgy érzi, hogy az egész közösség javát szolgálná,.



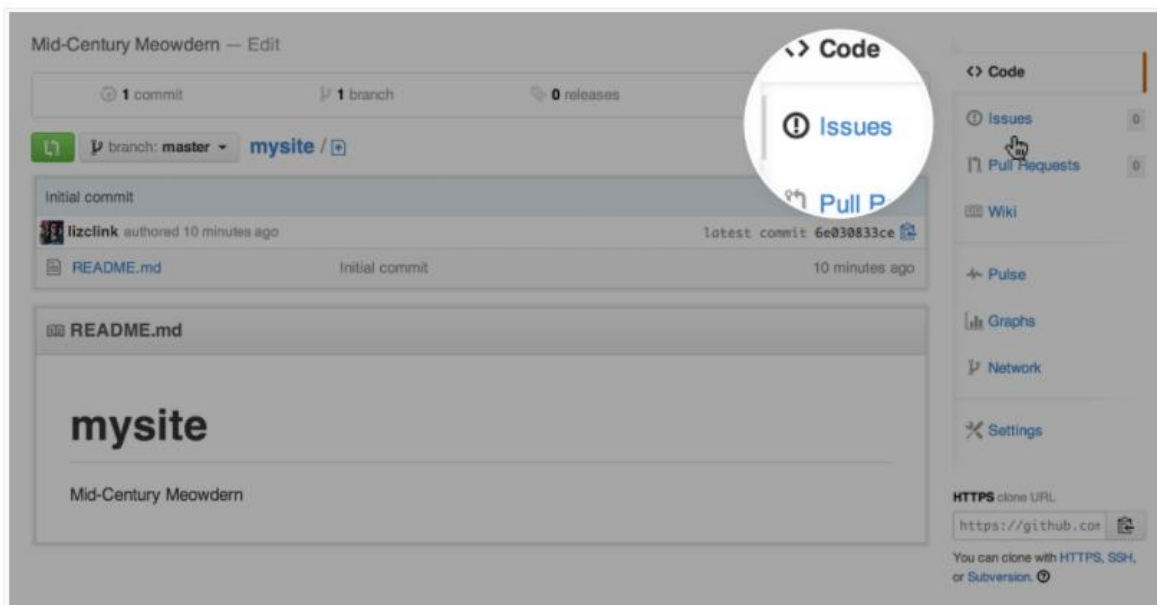
Az Compare and Pull Request elemre kattintva egy beszélgetési oldalra kerül, ahol megadhat címet és opcionális leírást. Fontos, hogy minél több hasznos információt és indoklást adjon meg arra vonatkozóan, hogy miért is adja meg először ezt a Pull Request-et. A projekt tulajdonosának meg kell tudnia állapítani, hogy a változtatás mindenki számára hasznos-e, mint gondolja.



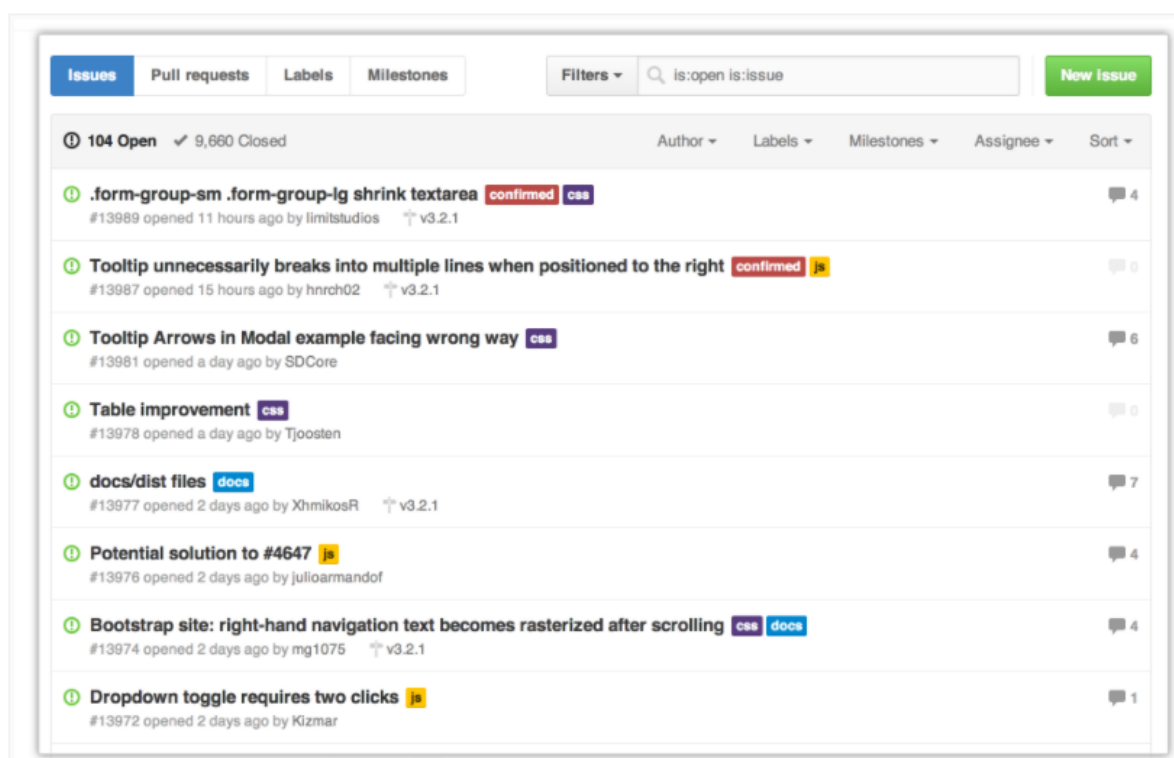
A Pull Requests egy vita terület. Ebben az esetben az Octocat nagyon elfoglalt, és valószínűleg nem fogja egyesíteni a változtatásokat. Más projektek esetén ne sértődjön meg, ha a projekt tulajdonosa elutasítja a Pull Request-et, vagy további információt kér arról, hogy miért készült. Még az is lehet, hogy a projekt tulajdonosa úgy dönt, hogy nem egyesíti a pull kérést, és ez teljesen rendben van.

Mastering Issues

Az issue-k nagyszerű módon nyomon követhetik a projektek feladatait, fejlesztéseit és hibáit. Olyanok, mint az e-mailek, csakhogy megoszthatók és megvitathatók a csapat többi tagjával. A legtöbb szoftverprojekt rendelkezik valamilyen hibakeresővel. A GitHub nyomkövetője Issues névre hallgat, és minden adattárban saját szakasza van.



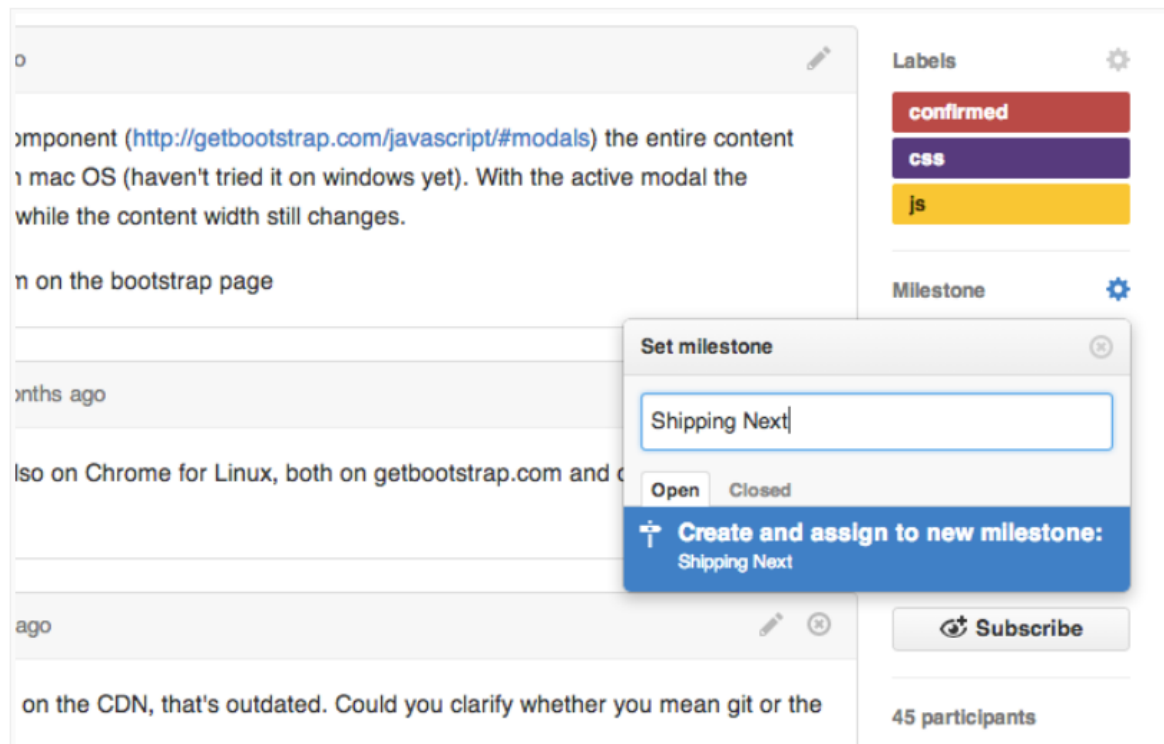
For example, let's take a look at [Bootstrap's Issues section](#):



A GitHub issue tracking azért különleges, mert az együttműködésre (collaboration), a referenciákra és a kiváló szövegformázásra összpontosítunk. A GitHub tipikus kérdése kissé így néz ki:

Ha nem látja a szerkesztés gombokat, az az oka, hogy nincs engedélye a probléma szerkesztésére. Megkérheti a repository tulajdonosát, hogy vegyen fel Önt együttműködőként a hozzáférés megszerzéséhez.

Milestones



A mérföldkövek olyan kérdések csoportjai, amelyek megfelelnek egy projektnek, funkciónak vagy időszaknak. Az emberek sokféleképpen használják őket a szoftverfejlesztésben.

Néhány példa a GitHub mérföldköveire:

Béta indítás - Fájlhibák, amelyeket meg kell javítani, mielőtt elindíthatja a projekt bétáját.

Remek módszer arra, hogy megbizonyosodjon arról, hogy nem hiányzik semmi.

Októberi sprint - olyan fájlok, amelyekkel októberben szeretne foglalkozni. Remek lehetőség arra, hogy erőfeszítéseit összpontosítsa, amikor sok a tennivaló.

Újratervezés - A projekt újratervezésével kapcsolatos fájlok. Remek módszer ötletek gyűjtésére, hogy min dolgozzon.

Címkék

A címkék kiválóan alkalmasak a különféle típusú kérdések rendezésére. A kiadásoknak annyi címkéje lehet, amennyit csak akar, és egyszerre egy vagy több címke alapján szűrhet.

The screenshot shows a list of five GitHub issues. Each issue has a title, a status icon (green circle with an exclamation mark), labels (js, css, confirmed, feature), the user who opened it, the time since it was opened, the number of comments, and an issue number.

- Open modal is shifting body content to the left** (js, css, confirmed) #9855. Opened by mat0r 3 months ago, 62 comments.
- Navbar issues** (js, css, confirmed) #11243. Opened by Nugrata a month ago, 36 comments.
- Add support of extra styling class on collapse event** (js, feature, css) #11350. Opened by zlogaschr 22 days ago, 24 comments.
- Redundant responsive utility styles** (css) #11214. Opened by AlexYursha a month ago, 24 comments.
- Select tag not properly styled on stock android browser** (css, confirmed) #11055. Opened by ADmad a month ago, 19 comments.

Assignees

Mindegyik számnak lehet engedélyezettje - egy személy, aki felelős a kérdés továbbhaladásáért. Az engedélyezettet ugyanúgy választják ki a mérföldkövek, mint a kiadás tetején található szürke sávon keresztül.

Notifications, @mentions, and References

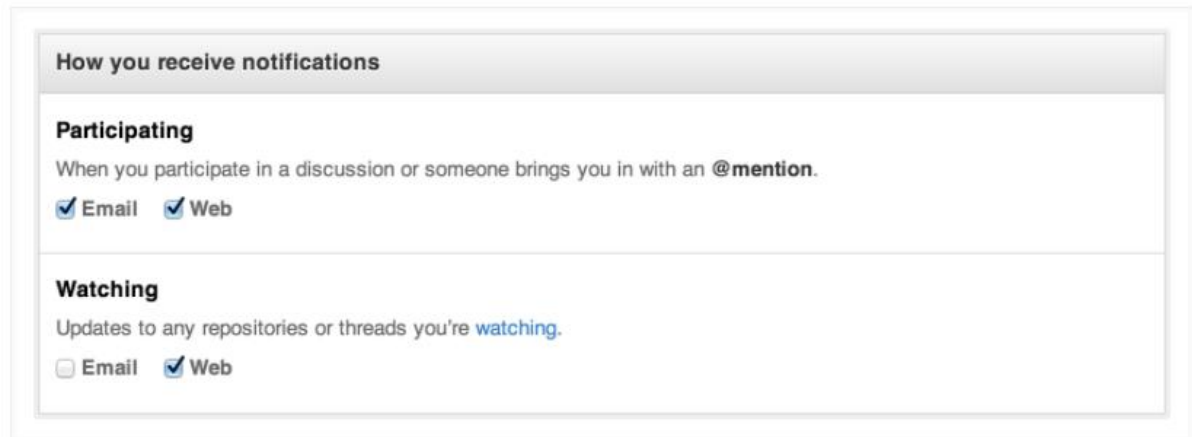
A Issue-k belsejében található @mentions és references használatával értesítheti a többi GitHub felhasználót és csapatot, és összekapcsolhatja a problémákat egymással. Ezek rugalmas módot kínálnak a megfelelő emberek bevonására a problémák hatékony megoldására, és könnyen megtanulhatók és használhatók. A GitHub összes szövegmezőjén dolgoznak - a GitHub Flavored Markdown nevű szövegformázási szintaxisunk részét képezik.

The screenshot shows the GitHub 'New Issue' form. At the top, there is a yellow banner that says 'Please review the [guidelines for contributing](#) to this repository.' Below this is the GitHub logo and a text input field with the placeholder 'This is an example of a new issue'. There are two tabs: 'Write' and 'Preview'. The 'Preview' tab is selected, showing a preview of the issue content. The preview text is: 'Text fields on GitHub are parsed with Markdown. This means we can @mention people like @githubstudent or @octocat to reference them into the issue. You can also cross-reference other issues and pull requests with the number of those issues and pull requests like #14020. You can also create tasks list to monitor the progress of smaller tasks as well: - [] #23 - [] #142 - [] #140. Attach images by dragging & dropping, [selecting them](#), or pasting from the clipboard.' At the bottom right, there is a green button that says 'Submit new issue'.

Értesítések

Az értesítések a GitHub módja annak, hogy naprakészen tartsa problémáit. Használhatja őket az repository-k új issue-jainak megismerésére, vagy csak annak megismerésére, hogy valakinek szüksége van-e az Ön közreműködésére egy issue előrehaladásához.

Az értesítések kétféleképpen fogadhatók: e-mailben és az interneten keresztül. A beállításokban konfigurálhatja az értesítések fogadásának módját. Ha sok értesítést szeretne kapni, javasoljuk, hogy kapjon webes és e-mailes értesítéseket.



The screenshot shows the 'How you receive notifications' settings in GitHub. It is divided into two sections: 'Participating' and 'Watching'. Under 'Participating', there is a description: 'When you participate in a discussion or someone brings you in with an @mention.' Below this, there are two checkboxes: 'Email' (checked) and 'Web' (checked). Under 'Watching', there is a description: 'Updates to any repositories or threads you're watching.' Below this, there are two checkboxes: 'Email' (unchecked) and 'Web' (checked).

@mentions

@mentions más GitHub-felhasználókat hivatkozhatunk a GitHub-issue-n belül. Az issue leírásában vagy bármilyen megjegyzésében adja meg egy másik GitHub felhasználó @ felhasználónevét, hogy értesítést küldhessen nekik. Ez nagyon hasonlóan működik, mint ahogy a Twitter használja a @issue-t

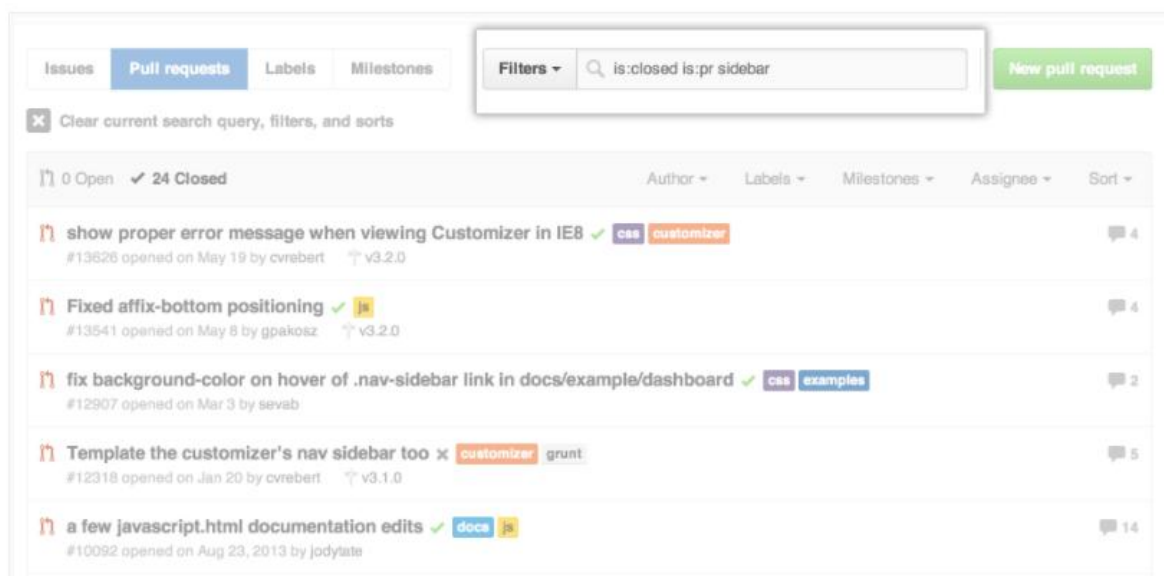
Hivatkozások

Gyakran előfordul, hogy a problémák más kérdésektől függenek, vagy legalábbis kapcsolódnak hozzájuk, és szeretné összekapcsolni a kettőt. A kérdésekre hivatkozhat, ha beír egy hashtaget és a issue számát.

Hey @kneath, I think the problem started in #42

Keresés

Minden oldal tetején található egy keresőmező, amely lehetővé teszi az issue-k közötti keresést.



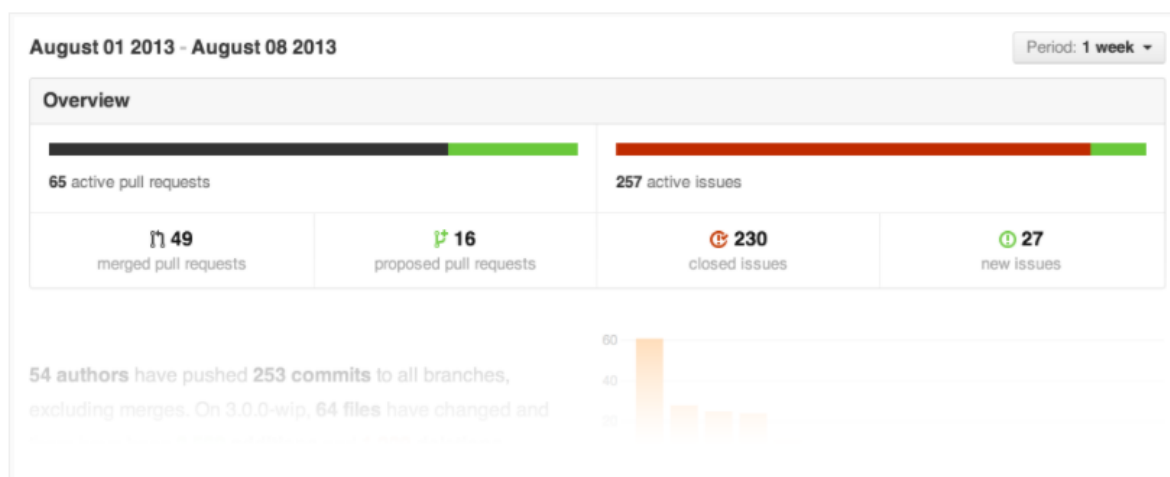
The Issue Dashboard

Ha számos projekt szélesebb listáját keresi számos projektben, akkor ez egy remek eszköz lehet. Az dashboard nagyon hasonlóan működik, mint az issue szakasz, de másként gyűjti az issue-kat:

- Az összes a repository-ban,
- amelyek tulajdonában vannak
- és amelyeken együttműködik
- Önhöz rendelt
- Ön által létrehozott

Pulse

Minden repository alatt található egy Pulse nevű szakasz - A Pulse pillanatkép mindenről, ami a repository-ban történt az elmúlt héten (vagy nap, vagy elmúlt 3 hónap stb.).



Markdown elsajátítása

A Markdown a webes szövegstílus módja. Te irányítod a dokumentum megjelenítését; a szavak félkövér vagy dőlt formázása, képek hozzáadása és listák létrehozása csak néhány

dolog, amit a Markdown segítségével megtehetünk. Többnyire a Markdown csak normál szöveg, néhány nem ábécés karakterrel, például # vagy * betűvel.

A Markdownt a GitHub-on használhatja a legtöbb helyen:

- Gists
- Comments in Issues and Pull Requests
- .Md vagy .markdown kiterjesztésű fájlok

Példák:

Text

```
It's very easy to make some words bold and other words italic with
Markdown. You can even [link to Google!] (http://google.com)
```

Listák:

```
Sometimes you want numbered lists:
```

- ```
1. One
2. Two
3. Three
```

```
Sometimes you want bullet points:
```

- ```
* Start a line with a star
* Profit!
```

```
Alternatively,
```

- ```
- Dashes work just as well
- And if you have sub points, put two spaces before the dash or star:
 - Like this
 - And this
```

### Képek:

```
If you want to embed images, this is how you do it:
```

```
![Image of Yaktocat] (https://octodex.github.com/images/yaktocat.png)
```

### Header and quotes:

```
Structured documents
```

Sometimes it's useful to have different levels of headings to structure your documents. Start lines with a `#` to create headings. Multiple `##` in a row denote smaller heading sizes.

```
This is a third-tier heading
```

You can use one `#` all the way up to `#####` six for different heading sizes.

If you'd like to quote someone, use the `>` character before the line:

```
> Coffee. The finest organic suspension ever devised... I beat the Borg with
it.
> - Captain Janeway
```

## Codes:

There are many different ways to style code with GitHub's markdown. If you have inline code blocks, wrap them in backticks: `var example = true`. If you've got a longer block of code, you can indent with four spaces:

```
 if (isAwesome){
 return true
 }
```

GitHub also supports something called code fencing, which allows for multiple lines without indentation:

```
```  
if (isAwesome){  
    return true  
}  
```
```

And if you'd like to use syntax highlighting, include the language:

```
```javascript  
if (isAwesome){  
    return true  
}  
```
```

### Extrák:

GitHub supports many extras in Markdown that help you reference and link to people. If you ever want to direct a comment at someone, you can prefix their name with an @ symbol: Hey @kneath – love your sweater!

But I have to admit, tasks lists are my favorite:

- [x] This is a complete item
- [ ] This is an incomplete item

When you include a task list in the first comment of an Issue, you will see a helpful progress bar in your list of issues. It works in Pull Requests, too!

And, of course emoji!