Întrebări și Răspunsuri despre Git și GitHub

# Întrebări și Răspunsuri despre Git

## Ce este Git și de ce este important în dezvoltarea software-ului?

Git este un sistem de control al versiunilor distribuit, open-source, care permite urmărirea modificărilor aduse fișierelor și coordonarea lucrului între mai mulți dezvoltatori. Este important deoarece asigură că fiecare modificare este documentată, permite colaborarea eficientă și oferă posibilitatea de a reveni la versiunile anterioare ale codului.

## Cum initializezi un nou repository Git?

Comanda `git init` inițializează un nou repository Git în directorul curent.

## Ce este un commit în Git?

Un commit este o înregistrare a modificărilor făcute într-un repository. Fiecare commit are un mesaj descriptiv și un identificator unic (hash) care îl diferențiază de alte commit-uri.

## Cum adaugi fișiere în zona de staging?

Comanda `git add [fisier]` adaugă fișierele specificate în zona de staging, pregătindu-le pentru următorul commit.

## Cum creezi un commit cu un mesaj?

Comanda `git commit -m "mesaj"` creează un commit cu fișierele din zona de staging și atașează un mesaj descriptiv.

## Ce este un branch și de ce este util?

Un branch este o ramură de dezvoltare care permite lucrul izolat pe funcționalități sau fix-uri. Este util deoarece permite dezvoltarea paralelă și menținerea stabilității codului în branch-ul principal.

## Cum creezi un nou branch?

Comanda `git branch [nume-branch]` creează un nou branch.

## Cum comuți între branch-uri?

Comanda `git checkout [nume-branch]` comută pe branch-ul specificat.

## Ce este un merge și cum se face?

Un merge este procesul de combinare a modificărilor dintr-un branch în altul. Se face folosind comanda `git merge [nume-branch]`.

## Ce este un conflict de merge și cum se rezolvă?

Un conflict de merge apare atunci când două modificări contradictorii sunt combinate. Se rezolvă manual, editând fișierele conflictuale și apoi realizând un nou commit.

## Cum poți vizualiza istoricul commit-urilor?

Comanda `git log` afișează istoricul commit-urilor.

## Ce este rebase și când ar trebui utilizat?

Rebase este procesul de mutare a unei serii de commit-uri pe un nou punct de bază. Ar trebui utilizat pentru a crea un istoric mai liniar și mai curat al commit-urilor.

## Cum salvezi modificările curente fără a le comite, pentru a lucra la altceva?

Comanda `git stash` salvează temporar modificările curente și curăță zona de lucru pentru a permite lucrul la altceva.

## Cum poți anula modificările dintr-un fișier?

Comanda `git checkout -- [fisier]` anulează modificările necomise dintr-un fișier.

## Cum poți reveni la un commit anterior?

Comanda `git checkout [commit-hash]` permite revenirea la un commit anterior.

# Întrebări și Răspunsuri despre GitHub

## Ce este GitHub și cum diferă de Git?

GitHub este o platformă de găzduire pentru proiecte Git, care oferă funcționalități suplimentare pentru colaborare și gestionarea proiectelor, cum ar fi pull requests, issues, și integrare continuă. Git este un sistem de control al versiunilor, în timp ce GitHub este o platformă care utilizează Git și adaugă funcționalități suplimentare.

## Cum creezi un repository pe GitHub?

Pe GitHub, te conectezi la contul tău, mergi la secțiunea "Repositories", și apeși pe butonul "New". Completezi detaliile repository-ului și apeși pe butonul "Create repository".

## Cum clonezi un repository GitHub local?

Comanda `git clone [URL]` clonează un repository GitHub local.

## Ce este un pull request și care este scopul său?

Un pull request este o cerere de îmbinare a modificărilor dintr-un branch (sau fork) într-un alt branch. Scopul său este de a facilita revizuirea codului și colaborarea între dezvoltatori.

## Cum contribui la un proiect open-source pe GitHub?

Pașii tipici sunt: fork-ul repository-ului, clonarea acestuia local, crearea unui nou branch, realizarea modificărilor, commit-urile, push-ul modificărilor și crearea unui pull request.

## Ce sunt GitHub Issues și cum sunt utilizate?

GitHub Issues sunt utilizate pentru a gestiona task-urile, bug-urile și cerințele pentru proiect. Ele permit urmărirea problemelor și colaborarea pentru rezolvarea lor.

## Cum configurezi GitHub Actions pentru CI/CD?

Creezi un fișier YAML în directorul `.github/workflows` din repository-ul tău, unde definești pașii pentru build, testare și deploy.

## Ce este un fork și când ar trebui utilizat?

Un fork este o copie a unui repository existent, care permite dezvoltarea independentă. Este utilizat de obicei pentru a contribui la proiecte open-source sau pentru a începe o dezvoltare paralelă.

## Cum poți gestiona accesul și permisiunile în repository-urile GitHub?

Prin setările repository-ului, poți invita colaboratori și poți gestiona permisiunile acestora (read, write, admin).

## Ce este GitHub Pages și cum îl folosești pentru a publica un site web?

GitHub Pages este un serviciu gratuit de găzduire pentru site-uri web statice. Publici un site web prin crearea unui branch `gh-pages` în repository-ul tău și adăugarea fișierelor HTML, CSS, și JavaScript.

## Cum protejezi branch-urile importante în GitHub?

Poți configura protecția branch-urilor prin setările repository-ului, unde poți impune reguli pentru commit-uri, merge-uri și review-uri.

## Ce este un webhook în contextul GitHub și cum se utilizează?

Un webhook este un callback HTTP care este declanșat de anumite evenimente în repository (de exemplu, push-uri). Este utilizat pentru a integra GitHub cu alte servicii și aplicații.

## Cum gestionezi conflictele de merge în cadrul unui pull request pe GitHub?

Conflictele de merge pot fi gestionate prin revizuirea fișierelor conflictuale în interfața GitHub sau local, rezolvarea conflictelor și realizarea unui nou commit pentru a actualiza pull request-ul.

## Ce sunt GitHub Discussions și cum sunt utilizate?

GitHub Discussions sunt un forum pentru comunicare și colaborare în cadrul unui proiect. Ele permit echipelor să discute idei, să răspundă la întrebări și să facă anunțuri.

## Cum poți automatiza procesul de code review folosind GitHub?

Poți utiliza GitHub Actions sau alte instrumente de automatizare pentru a configura verificări automate ale codului (linting, teste), asigurându-te că pull request-urile respectă standardele înainte de review-ul manual.