2025

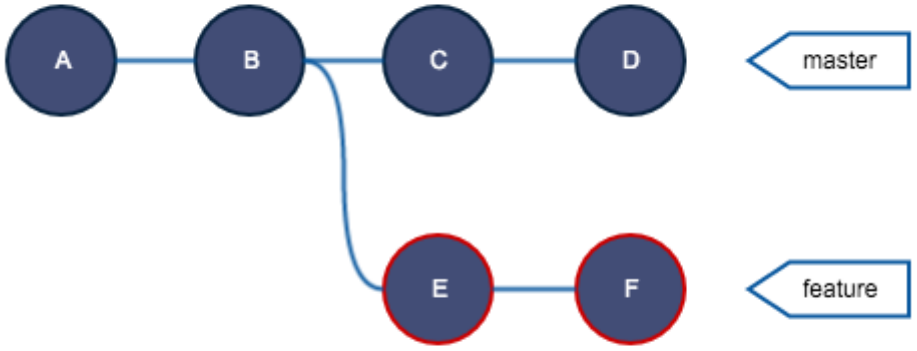
ბიზნესისა და ტექნოლოგიების უნივერსიტეტი

ვერსიონირება და უწყვეტი ინტეგრაცია

თემა მეოთხე: Rebase და შედარება შერწყმასთან (merge)

Git Rebase

Git rebase არის ბრძანება რომლის მეშვეობითაც შესაძლებელია ბრენჩის ბაზის გადაწერა. მარტივად რომ ვრთვათ ის არ ცვლის ბრენჩში დამატებულ ქომითებს მაგრამ ამატებს ან ცვლის ქომითებს სხვა ბრენჩიდან, თითქოს განშტორება შეიქმნა სხვა ქომითიდან. მაგალითად:

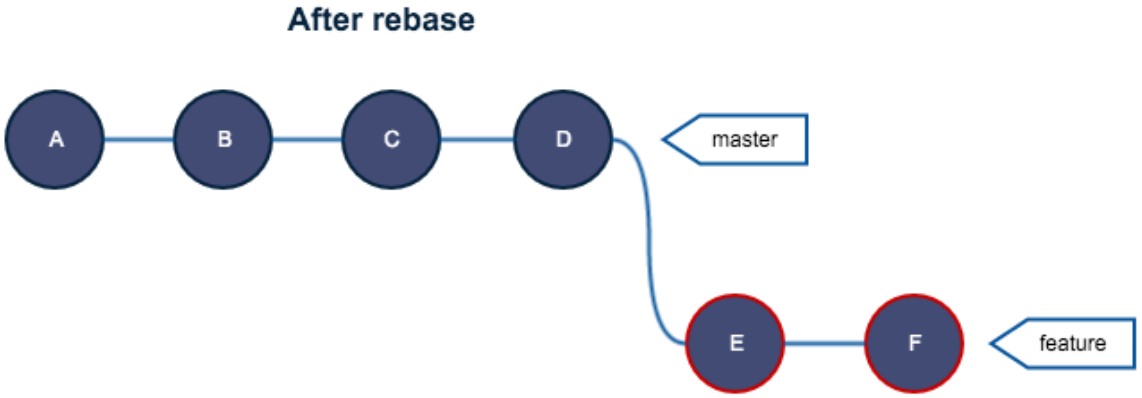


Feature ბრენჩი შეიქმნა B ქომითიდან რის შემდეგაც მასტერ შტოში კიდევ რამოდენიმე ქომითი გაკეთდა, ამ შემთხვევაში ბრენჩების დამერჯვისას დაემატება კიდევ დამატებითი გამაერთიანებელი ქომითი. Feature ბრენჩი რომ შექმნილიყო D ქომითიდან და იგივენაირად E და F ქომითები დაემატებინა, მერგი გაკეთდებოდა fast-forward სტრატეგიით და არ შექმნიდა დამატებით (გამაერთიანებელ) ქომითს. Rebase ბრძანებით შესაძლებელია უკვე არსებული feature ბრენჩის გადაკეთება თითოს ის შეიქნა D ქომითიდან.

Git checkout feature

Git rebase master

გადააკეტებს feature ბრენჩს, შექმნის ბაზად აიღებს D ქომითს და feature ბრენჩის ლოგებში აჩვენებს ჯერ მასტერ შტოს ქომითებს (D ქომითის ცათვლით) შემდეგ კი თავის ქომითებს (E და F ქომითებს).



Rebase-ის გამოყენებამდე საჭიროა სიტუაციის გააზრება რადგან მას როგორც დადებითი ასევე უარყოფითი მხარეები გააჩნია.

Rebase-ის უპირატესობებია:

- მკაფიო ქომითების ისტორია
- გამარტივებული ქომითების ლოგები
- არ საჭიროებს მერჯ ქომითს

Rebase-ის რისკებია:

- გაზიარებულ ბრენჩზე რებეისი სინქრონიზაციის რისკს შეიცავს
- რებეისის დროს შესაძლებელია კონფლიქტი წარმოიქმნას რომელიც განსხვავებულ ბრძანებებს საჭიროებს მერჯ კონფლიქტისგან

Rebase კონფლიქტის დროს კონფლიქტი სტანდარტულად უნდა გადაიჭრას, ახალი ფაილის add ბრძანებით უნდა დაემატოს და შემდეგ rebase-ის გასაგრძელებლად გამოიყენება ბრძანება:

git rebase --continue

ხოლო rebase-ის გასაუქმებლად:

git rebase --abort

interactive rebase

გარდა ჩვეულებრივი rebase-is არსებობს interactive rebase. ჩვეულებრივი rebase გარდაქმნის ერთი შტოს ისტორიას სხვა შტოზე დაყრდნობით, ხოლო interactive rebase გარდაქმნის მინდინარე შტოს ისტორიას. interactive rebase-ის ბრძანება:

git rebase -i HEAD~(ქომითების რაოდენობა)

რადგან interactive rebase მუშაობს ერთ შტოზე საჭიროა **HEAD** პარამეტრის გამოყენებით მითითება თუ რომელ ვერსიას მიუბრუნდეს. ამ ბრძანების შემდეგ Git გახსნის ტექსტურ რედქტოს სადაც ჩამოთვლილი იქნება ყველა ქომითი რომელიც გაკეთდა ჩვენს მიერ მითითებული **HEAD**-ის შემდეგ.

```
1 pick e1bce7b Tweaking some things
2 pick 401f965 WIP
3 pick a5a177f Changes to contact form
4 pick 40238cf Fixes
5
6 # Rebase 5e06f48..40238cf onto 5e06f48 (4 commands)
7 #
8 # Commands:
9 # p, pick = use commit
10 # r, reword = use commit, but edit the commit message
11 # e, edit = use commit, but stop for amending
12 # s, squash = use commit, but meld into previous commit
13 # f, fixup = like "squash", but discard this commit's log message
14 # x, exec = run command (the rest of the line) using shell
15 # d, drop = remove commit
16 #
17 # These lines can be re-ordered; they are executed from top to bottom.
18 #
19 # If you remove a line here THAT COMMIT WILL BE LOST.
20 #
21 # However, if you remove everything, the rebase will be aborted.
22 #
23 # Note that empty commits are commented out
```

Pick-ით გამოყოფილია მითითებული ქომითები, თავისი კომენტარებით. **Commands**-ის ქვევით ჩამოთვლილია ძირითად ბრძანებები და მათი მნიშვნელობები. ბრძანების გამოსაყენებლად **pick** სიტყვა უნდა ჩანაცვლდეს შესაბამისი ბრძანებით, მაგალითად **edit** ან **squash**.

- **Pick** - ბრძანება ტოვებს ქომითს ისე როგორც არის, ტექსტურ რედაქტორში გამოიყენება ნაგულისხმევი მნიშვნელობით.
- **Reword** - ცვლის ქომითის კომენტარს.
- **Edit** - ცვლის ქომითის შიგთავსს. გამოყენების შემდეგ აბრუნებს აღნიშნული ქომითის ვერსიას, რედაქტირების და შეცვლის შესაძლებლობით. რის შემდეგაც საჭიროებს ფაილის დამატებას შეცვლას და რებეისზე დაბრუნებას:

  git add .
  git commit --amend
  git rebase --continue

- **Squash** - აერთიანებს რამოდენიმე ქომითს და მათ ნაცვლად ერთს ტოვებს ისტორიაში. **Squash**-ის გამოყენება საჭიროებს **pick** ბრძანებას, მაგალითად:

  pick b2c3d4e Add file2
  squash c3d4e5f Add file 1

  აღნიშნული ბრძანებით **b2c3d4e** , **c3d4e5f** ქომითები გაერთიანდება.

- **Fixup** - აერთიანებს რამოდენიმე ქომითს, თუმცა არ ინახავს fixup-ის შემდეგ მიწერილი ქომითის ცვლილებას, მაგალითად:

  pick b2c3d4e Add file2
  fixupc3d4e5f Add file 1

  აღნიშნული ბრძანებით **b2c3d4e** , **c3d4e5f** ქომითები გაერთიანდება. თუმცა დარცება მხოლოდ **b2c3d4e** ქომითის ცვლილებები.

- **Drop** - შლის ქომითს ქომითების ისტორიიდან.

interactive rebase ისევე როგორც ჩვეულებრივი **rebase** საჭიროებს დიდ ყურადღებას, რადგან ის მუშაობს ქომიტების ისტორიასთან.