Git for (litt) viderekomne

En praktisk guide til å lage rot og fikse opp i det

Imre Kerr

Recap av basics

- Start: `git init`
- Lagre arbeidet ditt: `git add` og `git commit`
- Jobb med remotes: `git push` og `git pull`
- Branching: `git switch` og `git merge`

Commit ofte

- Så fort noe er commitet, er det så å si umulig å miste det.
- Bedre å lage en rotete historikk og så rydde opp i etterkant.

Reachability

En commit er reachable, dersom den:

a) pekes på av en ref (branch eller tag), eller
b) er forelderen til en annen reachable commit

Regel: Reachable commits blir aldri slettet automatisk!

Lag en midlertidig branch før du gjør noe skummelt

```
git branch save-point-1
```

Hvis branchen din blir ødelagt:

```
git reset --hard save-point-1
```

...og alt er som før.

Omskriving av historikk

Rette på siste commit:

```
git commit --amend
```

Flytte aktiv branch til en annen commit:

```
git reset [--soft|--mixed|--hard] <commit>
```

Kopiere commit til aktiv branch (ikke omskriving, teknisk sett):

```
git cherry-pick <commit>
```

Ultimat multiverktøy:

```
git rebase --interactive [--onto <newbase>] <upstream> [<branch>]
```

Ikke skriv om historikk på delte brancher!

Ryddig commit-historikk

- Letter code review, bisect, lettere å finne når og hvorfor endringer ble gjort...
- Flytt rundt på commits, slå sammen, splitt opp

Splitte ucommitede endringer:

```
git add --patch
```

Bisect – finn når en bug ble introdusert

```
git bisect start
git bisect bad
git bisect good <kjent fungerende commit>
```

Deretter, til git sier at du er ferdig:

```
git bisect [good|bad]
```

Eller automatisk testing:

```
git bisect run <script> [<args>]
```

Best om alle commits bygger og tester grønt (annet enn den aktuelle bugen)

Alternativt:

```
git bisect skip
```

Finne unreachable commits

Historikk over commits du har vært innom:

```
git reflog
# eller:
git reflog <ref>
```

Finn hashen til commiten din, og lag en branch som peker på den:

```
git branch lost-commit a1b2c3
```

Vanskelig å lese output fra `git reflog`?

```
git log --reflog --graph --oneline
```

Finne igjen unreachable commits… på GitHub

“Repository events” – GitHubs “reflog”

```
https://api.github.com/repos/{owner}/{repo}/events
```

Finn event av type `PushEvent`, sjekk `before`-feltet.

```
https://github.com/{owner}/{repo}/tree/{hash}
```

…og lag ny branch fra dropdown.

Finne commit i historikken

`git log` har en søkefunksjon:

```
git log -S <string>
```

Kan såklart kombineres med `--reflog`.

Fjerne ting fra git-historikken

Hemmeligheter, GDPR-brudd, store filer…

På feature-branch:

- `git commit --amend`
- `git rebase --interactive`

Lengre tilbake: BFG Repo-Cleaner

```
https://rtyley.github.io/bfg-repo-cleaner/
```

⚠️ For å slette ting permanent etterpå:

```
git reflog expire --expire=now --all && git gc --prune=now --aggressive
```

❌ Ikke bruk: `git filter-branch`

BFG Repo-Cleaner, steg for steg

Alt må renses. Derfor:

Alle på teamet:

1. Commit og push (feature/midlertidig branch er fint)

En person:

2. `git fetch`, og så `git switch {branch}` for hver remote-branch
3. Kjør BFG
4. `git push --force --all`
5. `git push --force --tags`

Alle:

6. Slett repo-mappa og clone på nytt.

Videre lesning

Mitt 💡-øyeblikk:

```
http://think-like-a-git.net/
```

Dypdykk:

```
https://github.com/pluralsight/git-internals-pdf
https://git-scm.com/book/en/v2
```

Tid for oppgaver!

Oppgaver og slides:

```
https://github.com/imre-kerr-sb1/git-workshop
```

Åpne i Coder, kjør `./init.sh`. Oppgavene ligger nå i `/workspace`

- Jobb gjerne i par/små grupper
- Flere av oppgavene kan løses på flere måter. Diskuter pros/cons.