

git rebase

ta kontrollen over historien

Innhold

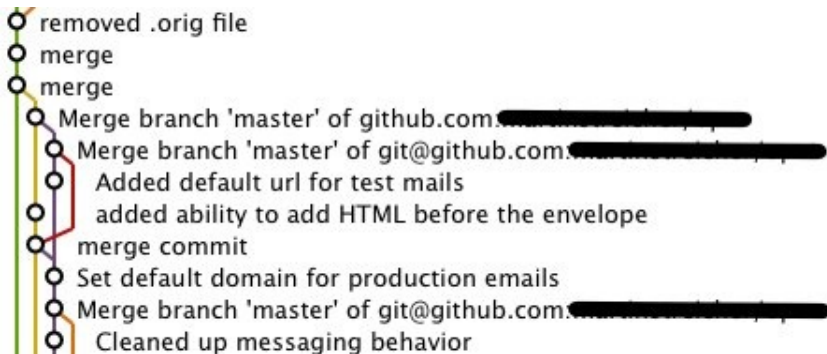
- ▶ Hva er problemet?
- ▶ Hva er galt med `merge`?
- ▶ Avmystifisering av `rebase`.
- ▶ En enkel arbeidsflyt.
- ▶ Lek med `--interactive`.

Motivasjon

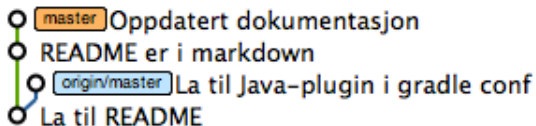
Merge made by the 'recursive' strategy.

Motivasjon

Merge made by the 'recursive' strategy.



Et vanlig scenario

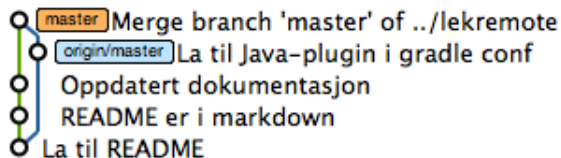


Et vanlig scenario

```
$ git pull origin master
```

Et vanlig scenario

```
$ git pull origin master
```



Hva gjør egentlig `pull`?

```
$ git pull origin master
```

...er det samme som ...

```
$ git fetch  
$ git merge origin/master
```


Hva er galt med `merge`?

- ▶ Historien blir grusom å lese
- ▶ Verktøy som `git bisect` blir forvirret
- ▶ *Semantisk* riktig?

Hva er galt med `merge`?

- ▶ Historien blir grusom å lese
- ▶ Verktøy som `git bisect` blir forvirret
- ▶ *Semantisk riktig?*

Hva er galt med `merge`?

- ▶ Historien blir grusom å lese
- ▶ Verktøy som `git bisect` blir forvirret
- ▶ *Semantisk* riktig?

Løsningen

Vi vil simulere seriell utvikling ved hjelp av `rebase`!

```
man git-rebase
```

```
git-rebase - Forward-port local commits to the updated upstream head
```

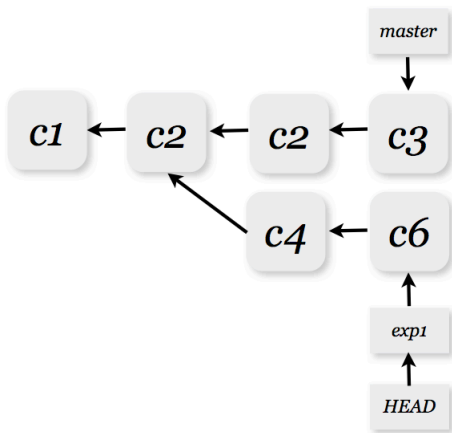
```
man git-rebase
```

```
git-rebase - Forward-port local commits to the updated upstream head
```

At du sa??

Hva er rebase?

“Ta mine nyeste commits, og legg dem oppå det treet der.”

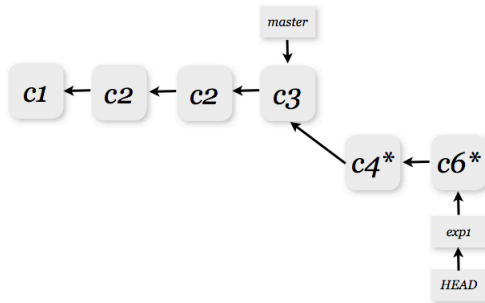
Hva er `rebase`?

Hva er rebase?

```
$ git rebase master
```

Hva er `rebase`?

```
$ git rebase master
```



Hva utgjør en commit?

```
$ git cat-file -p HEAD
tree 9724058020812154185e7feeed04cb0a8def91d0
parent 6a72980c6e7010c4687ba65721314d2404047442
author Pål Ruud <ruudud@gmail.com> 1346090316 +0200
committer Pål Ruud <ruudud@gmail.com> 1346090316 +0200
```

La til Java-plugin i gradle conf

Farlig?

Om du rebaser (les: endrer) en commit som ligger sentralt, vil andre som har basert arbeid på denne committen bli svært forvirret.

Farlig?

Om du rebaser (les: endrer) en commit som ligger sentralt, vil andre som har basert arbeid på denne committen bli svært forvirret.

SÅ IKKE GJØR DET!!1

Tilbake til det første eksempelet ...

```
$ git pull --rebase
```

```
First, rewinding head to replay your work on top of it...
```

```
Applying: README er i markdown
```

```
Applying: Oppdatert dokumentasjon
```

Tilbake til det første eksempelet ...

```
$ git pull --rebase
```

```
First, rewinding head to replay your work on top of it...
```

```
Applying: README er i markdown
```

```
Applying: Oppdatert dokumentasjon
```

- master Oppdatert dokumentasjon
- README er i markdown
- origin/master La til Java-plugin i gradle conf
- La til README

Hva gjør egentlig `pull`?

```
$ git pull --rebase origin master
```

...er det samme som ...

```
$ git fetch  
$ git rebase origin/master
```


For den vågale ...

```
[branch]
```

```
autosetuprebase = always
```

~/.gitconfig

Enkel arbeidsflyt basert på rebase

1. Lag ny branch basert på **master**
2. Lag fine, ryddige commits som bare gjør én ting
3. Med gjevne mellomrom og før sammenslåing, dra ned endringer og rebase på **master**
4. Bytt til **master** og merge inn feature branch.

Enkel arbeidsflyt basert på `rebase`

1. Lag ny branch basert på **master**
2. Lag fine, ryddige commits som bare gjør én ting
3. Med gjevne mellomrom og før sammenslåing, dra ned endringer og rebase på **master**
4. Bytt til **master** og merge inn feature branch.

Enkel arbeidsflyt basert på rebase

1. Lag ny branch basert på **master**
2. Lag fine, ryddige commits som bare gjør én ting
3. Med gjevne mellomrom og før sammenslåing, dra ned endringer og rebase på **master**
4. Bytt til **master** og merge inn feature branch.

Enkel arbeidsflyt basert på `rebase`

1. Lag ny branch basert på **master**
2. Lag fine, ryddige commits som bare gjør én ting
3. Med gjevne mellomrom og før sammenslåing, dra ned endringer og rebase på **master**
4. Bytt til **master** og merge inn feature branch.

Mer komplisert arbeidsflyt basert på rebase

Scenario: fem utviklere må samarbeide om en feature.

1. Lag ny branch basert på **master**
2. Lag fine, ryddige commits som bare gjør én ting
3. Før du deler koder din, dra ned endringer og rebase på **origin/ny-branch**
4. Om det går lang tid (dårlig tegn), bli enige om å få branchen oppdatert med endringer fra master
5. Før feature skal tilbake til master: kommunisere at én får oppgaven med å rydde i historien.
6. Bytt til **master** og merge inn feature branch.

Mer komplisert arbeidsflyt basert på rebase

Scenario: fem utviklere må samarbeide om en feature.

1. Lag ny branch basert på **master**
2. Lag fine, ryddige commits som bare gjør én ting
3. Før du deler koder din, dra ned endringer og rebase på **origin/ny-branch**
4. Om det går lang tid (dårlig tegn), bli enige om å få branchen oppdatert med endringer fra master
5. Før feature skal tilbake til master: kommunisere at én får oppgaven med å rydde i historien.
6. Bytt til **master** og merge inn feature branch.

Mer komplisert arbeidsflyt basert på rebase

Scenario: fem utviklere må samarbeide om en feature.

1. Lag ny branch basert på **master**
2. Lag fine, ryddige commits som bare gjør én ting
3. Før du deler koder din, dra ned endringer og rebase på **origin/ny-branch**
4. Om det går lang tid (dårlig tegn), bli enige om å få branchen oppdatert med endringer fra master
5. Før feature skal tilbake til master: kommunisere at én får oppgaven med å rydde i historien.
6. Bytt til **master** og merge inn feature branch.

Mer komplisert arbeidsflyt basert på rebase

Scenario: fem utviklere må samarbeide om en feature.

1. Lag ny branch basert på **master**
2. Lag fine, ryddige commits som bare gjør én ting
3. Før du deler koder din, dra ned endringer og rebase på **origin/ny-branch**
4. Om det går lang tid (dårlig tegn), bli enige om å få branchen oppdatert med endringer fra master
5. Før feature skal tilbake til master: kommunisere at én får oppgaven med å rydde i historien.
6. Bytt til **master** og merge inn feature branch.

Mer komplisert arbeidsflyt basert på rebase

Scenario: fem utviklere må samarbeide om en feature.

1. Lag ny branch basert på **master**
2. Lag fine, ryddige commits som bare gjør én ting
3. Før du deler koder din, dra ned endringer og rebase på **origin/ny-branch**
4. Om det går lang tid (dårlig tegn), bli enige om å få branchen oppdatert med endringer fra master
5. Før feature skal tilbake til master: kommunisere at én får oppgaven med å rydde i historien.
6. Bytt til **master** og merge inn feature branch.

Mer komplisert arbeidsflyt basert på `rebase`

Scenario: fem utviklere må samarbeide om en feature.

1. Lag ny branch basert på **master**
2. Lag fine, ryddige commits som bare gjør én ting
3. Før du deler koder din, dra ned endringer og rebase på **origin/ny-branch**
4. Om det går lang tid (dårlig tegn), bli enige om å få branchen oppdatert med endringer fra master
5. Før feature skal tilbake til master: kommunisere at én får oppgaven med å rydde i historien.
6. Bytt til **master** og merge inn feature branch.