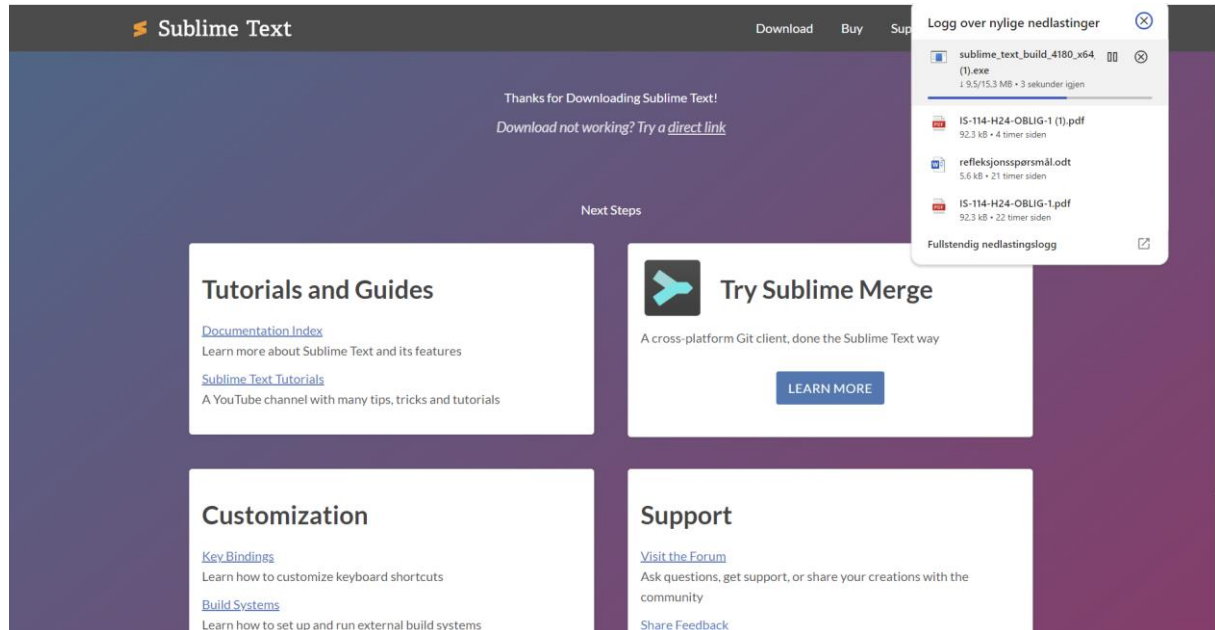


Hva kan være argumenter imot å bruke IDE (Integrated Development Environment) som Visual Studio Code eller IntelliJ hvis man er nybegynner i systemutvikling?

Sammenlignet med andre verktøy kan Ide være vanskelig å bruke for nybegynnere. programmeringsverktøyet er nyttig, men kan fremstå som komplekst og overveldende, noe som kan føre til at ineffektivitet og lengere læringsprosess enn nødvendig

Dokumentasjon

Lastet ned verktøy sublime



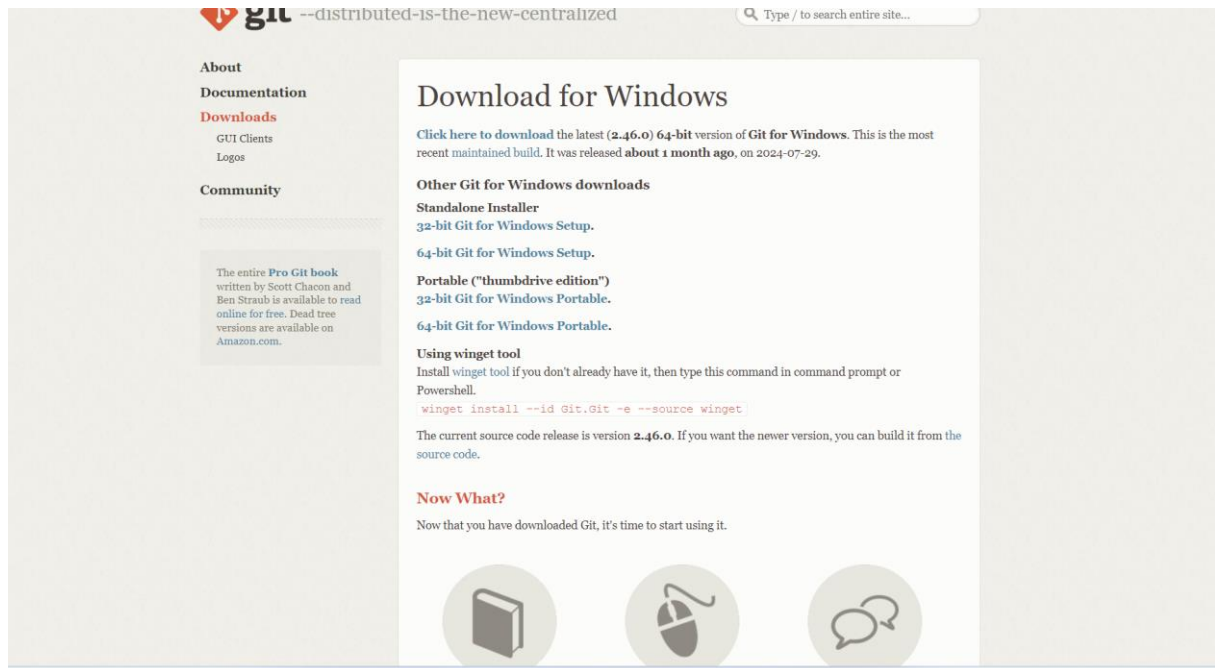
Hva er Git og hvorfor trenger man å bruke det? Hvorfor er det ikke anbefalt å installere en GUI (Graphical User Interface) klients applikasjon for Git?

Git er et program som hjelper brukere å lagre, endre filer og oppbevare filer. Ved bruke av git er det lettere å holde oversikten over utvikling og samarbeidsprosesser.

Det anbefales å bruke git kommandolinjen i istedenfor Gui klient for nybegynnere fordi kommandolinjen er lettere forståelig og gir en dypere forståelse av hvordan git fungerer. Gui er kan være mer utfordrende for nybegynnere fordi den viser ikke nyttige konsepter som er viktig for tidlig læring.

Dokumentasjon

Lastet ned Glt



Hvilke grunnleggende kommandoer kan brukes på kommandolinjen for å kunne se lene, som er lagret på egen datamaskin? Hvordan finner jeg de samme lene i Terminal/GitBash som jeg ser i applikasjonen med grask grensesnitt for fremvisning av mappe- og l- struktur på min datamaskin? Hvorfor er det anbefalt å lære å bruke Git-kommandoer, som kan brukes på kommandolinjen?

For å navigere i git må man bruke kommandolinjen. Det er 3 grunnleggende måter å navigere i kommandolinjen på. Ls brukes for å liste filene i mappen. Pwd brukes for å vise gjeldene mappe og cd brukes for å bytte mappe.

Dokumentasjon

Utførte de tre grunnleggende kommandoene

```

danie@PC MINGW64 ~
$ ls
AndrMask@
AppData/
Contacts/
Cookies@
Daniel.github.io/
DanielDaniel/
Desktop/
Documents/
Downloads/
Favorites/
Links/
'Lokale innstillinger'@
Maler@
'Mine dokumenter'@
Music/
NTUSER.DAT
NTUSER.DAT{01a8171e-2e7f-11ef-8190-14ac606eda94}.TM.b1f
NTUSER.DAT{01a8171e-2e7f-11ef-8190-14ac606eda94}.TM.Container00000000000000000000
1. regtrans-ms
NTUSER.DAT{01a8171e-2e7f-11ef-8190-14ac606eda94}.TM.Container00000000000000000000
2. regtrans-ms
OneDrive/
Pictures/
Programdata@
Recent@
'Saved Games'/
Searches/
SendTo@
Skriver@
Start-meny@
Videos/
daniel/
ntuser.dat.LOG1
ntuser.dat.LOG2
ntuser.ini

danie@PC MINGW64 ~
$ pwd
/c/Users/danie

danie@PC MINGW64 ~
$ cd daniel

danie@PC MINGW64 ~/daniel (main)
$

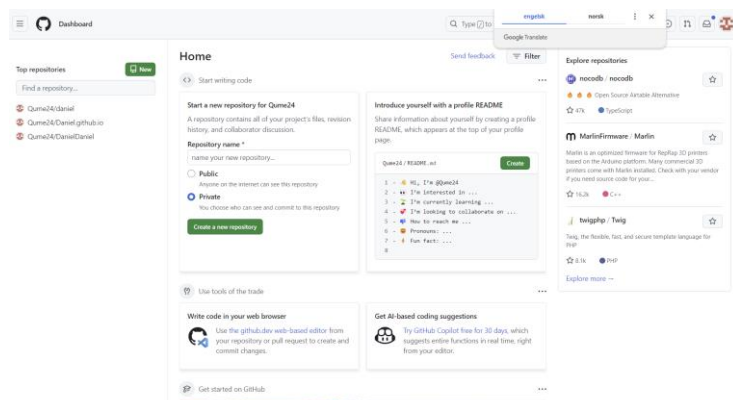
```

Hva karakteriserer et godt brukernavn på Github?

En bra bruker navn kan kjennetegne noe som er enkelt å huske, men samtidig profesjonelt. Bruker navnet bør også være noe som er gjenkjennelig for alle og ikke noe som er personlig for deg.

Dokumentasjon

Lagde konto på Git hub

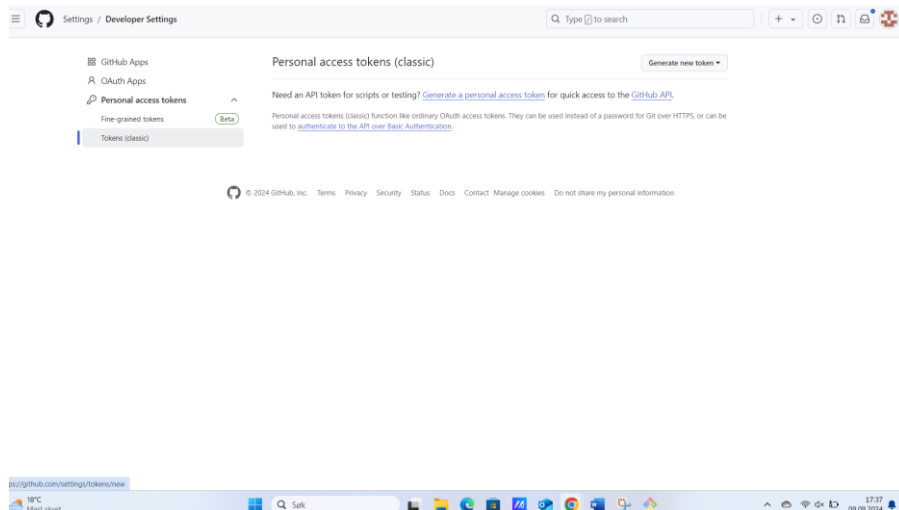


Hvorfor trenger man en slik "token"? Kan man bruke andre måter å autentisere seg på Github enn "token"

Personal Access token er som en bekreftelse på å tilgang til git ressurser på vegne av brukeren. De finnes også mange andre måter for autentisering på git hub for eksempel ssh- nøkler. Dette er en av de mer populære måtene for å bruke personal Access token.

Dokumentasjon

Lagde en personal Access token



Hva er en repository? Hvorfor skal et repositorynavn være meningsbærende? Hvorfor velge en lisens? Hva er formålet med README-l og hvorfor er det anbefalt å ha det i hver repository? Hva betyr det at repository-en er på Github? Hva betyr det at repository har en gren (på engelsk branch) som heter main

Hva er repository

En repository er en oppbevaring av alle filer tilknyttet et prosjekt. Med navn som er mer meningsfylt vil være lettere for andre brukere å forstå både innholdet og hva prosjektet handler om. For at andre skal kunne endre og bruke koden er det nødvendig, med en lisens. Hovedformålet med readme er å informere hva selve prosjektet handler om. Den informerer også om instruksjoner til andre brukere. Det betyr i bunn og grunn at den er tilgjengelig for alle på nettet. Main grenen er når den endelige versjonen av prosjektet er ferdigstilt.

Dokumentasjon

Lagde en repository

Create a new repository
A repository contains all project files, including the revision history. Already have a project repository elsewhere?
[Import a repository.](#)

Required fields are marked with an asterisk (*).

Owner: Qume24 / Repository name: Daniel
Your new repository will be created as Daniel-
The repository name can only contain ASCII letters, digits, and the characters -, ., and _.

Great repository names are short and memorable. Need inspiration? How about [ideal-parakeet](#)?

Description (optional):

☒ **Public**
Anyone on the internet can see this repository. You choose who can commit.

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

Initialize this repository with:
☒ **Add a README file**
This is where you can write a long description for your project. [Learn more about READMEs.](#)

Add .gitignore
Choose which files not to track from a list of templates. [Learn more about creating files.](#)

Choose a license
License: None

Hvorfor skal man kopiere repository-en fra Github til egen datamaskin? Hvordan velger man et egnet sted å kopiere repository-en til på egen datamaskin? Hvilke ler og eventuelt mapper som blir kopiert fra Github til egen datam0061skin med kommandoen for kloning?’

Når man kloner repository kan man jobbe lokalt med et prosjekt, samtidig gjøre endringer og synkronisere dem med Github. Når man skal velge et eget sted å kopiere repository bør man velge en mappe, som er godt organisert. En mappe til prosjekter for eksempel.

Dokumentasjon

Klonet fra repository

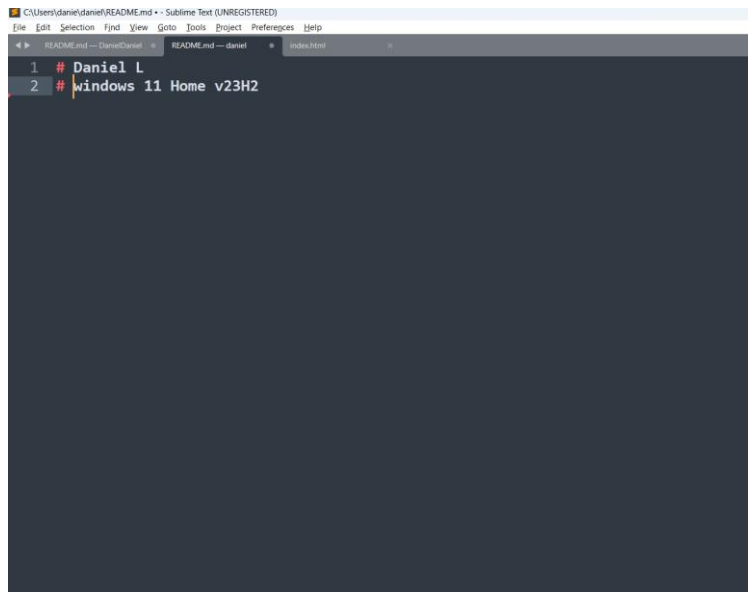
```
danie@Pc MINGW64 ~  
$ cd DanielDaniel  
  
danie@Pc MINGW64 ~/DanielDaniel (main)  
$ git clone https://github.com/Qume24/DanielDaniel.git  
Cloning into 'DanielDaniel'...  
remote: Enumerating objects: 3, done.  
remote: Counting objects: 100% (3/3), done.  
remote: Total 3 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)  
Receiving objects: 100% (3/3), done.  
  
danie@Pc MINGW64 ~/DanielDaniel (main)  
$ cd DanielDaniel  
  
danie@Pc MINGW64 ~/DanielDaniel/DanielDaniel (main)  
$ git status  
On branch main  
Your branch is up to date with 'origin/main'.  
  
nothing to commit, working tree clean  
  
danie@Pc MINGW64 ~/DanielDaniel/DanielDaniel (main)  
$
```

Er den eneste måten å gjøre endringene på i lene i en mappe administrert av Git å åpne egen applikasjon for tekstredigering? Kunne jeg endret andre type ler på samme måte, som, for eksempel, bilde- eller videoer? Begrunne svar.

Dersom man ønsker å gjøre endringer på Git hub via nettleseren er dette fullt mulig, men det er mer partisk å bruke et lokalt tekstredigeringsverktøy. Det er også mulig og med bilder eller video

Dokumentasjon

Skriver navn på operativsystemet



Hvordan finner man hva er gjeldende mappe på kommandolinjen? Hvorfor trenger man flere operasjoner for å registrere endringene i Git versjonskontrollsystemet? Kan man klonе og kopiere endringene tilbake for en repository man ikke selv eier (dvs. repository-er som andre har laget ved hjelp av sine kontoer på Github)? Hvorfor er det nyttig å kopiere endringene gjort i en l på egen datamaskin til Github?

Pwd brukes for å navigere frem til gjeldene mappe. Man trenger flere operasjoner fordi det gir kontroll og versionshåndtering over endringer som skal legges. Det er mulig å klonе endinger tilbake selv om ikke eier det. Får å klonе et repository må man skrive en kommando pull. Det er smart å laste opp endringer fra din lokale maskin til git fordi da får man en sikkerhets kopi og arbeides sitt.

Dokumentasjon

Utøver ulike kommandoer

```
C:\Users\daniel> git status
fatal: not a git repository (or any of the parent directories): .git

daniel@Pc MINGW64 ~
$ cd daniel

daniel@Pc MINGW64 ~/daniel (main)
$ git status
On branch main
Your branch is up to date with 'origin/main'.

nothing to commit, working tree clean

daniel@Pc MINGW64 ~/daniel (main)
$ git status
On branch main
Your branch is up to date with 'origin/main'.

Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   README.md

no changes added to commit (use "git add" and/or "git commit -a")

daniel@Pc MINGW64 ~/daniel (main)
$ git add .

daniel@Pc MINGW64 ~/daniel (main)
$ git commit -m"changing user name"
[main b3b9500] changing user name
1 file changed, 1 insertion(+), 1 deletion(-)

daniel@Pc MINGW64 ~/daniel (main)
$ git push
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 16 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 340 bytes | 340.00 KiB/s, done.
Total 3 (delta 1), reused 0 (delta 0), pack-reused 0 (from 0)
remote: Resolving deltas: 100% (1/1), completed with 1 local object.
To https://github.com/Quime24/daniel.git
   a0a7488..b3b9500  main -> main

daniel@Pc MINGW64 ~/daniel (main)
$
```

Hvilke argumenter kan være imot å gjøre endringene i filene direkte i web-grensesnittet til Github?

Det kan være mindre effektivt å gjøre endringene på filer direkte i web enn på Git hub fordi når du tester endringer kan det øke risikoen får feil, spesielt når man jobber med større filer.

Dokumentasjon

Skriver inn navn på operativsystemet

```
C:\Users\daniel\daniel\README.md - Sublime Text (UNREGISTERED)
File Edit Selection Find View Goto Tools Project Preferences Help

1 # Daniel L
2 # windows 11 Home v23H2
```

Hvilken respons genererer "push"-kommandoen? Hva er grunnen til en slik respons?

Når man utfører kommandoen Git Push får man en melding som informerer om at endringene er vellykket lastet opp, dersom man ikke har noen konflikter. Grunnen til en slik respons er at Git sjekker om den lokale repository er synkronisert som den skal med GIT HUB

Dokumentasjon

Legger til Version nummeret på git

```
1 # Daniel L
2 # windows 11 Home v23H2
3 # Git version 2.46.0.windows.1
4 #
```

Skriver inn kommandoer for å lagre endringene i Git hub.com

```
daniel@PC-MINGW64 ~/daniel (main)
$ git status
On branch main
Your branch is up to date with 'origin/main'.

nothing to commit, working tree clean

daniel@PC-MINGW64 ~/daniel (main)
$ git status
On branch main
Your branch is up to date with 'origin/main'.

Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   README.md

no changes added to commit (use "git add" and/or "git commit -a")

daniel@PC-MINGW64 ~/daniel (main)
$ git add .

daniel@PC-MINGW64 ~/daniel (main)
$ git commit -m "adding Git version"
[main 3df2bd1] adding Git version
1 file changed, 1 insertion(+), 1 deletion(-)

daniel@PC-MINGW64 ~/daniel (main)
$ git push
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 16 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 344 bytes | 344.00 KiB/s, done.
Total 3 (delta 1), reused 0 (delta 0), pack-reused 0 (from 0)
remote: Resolving deltas: 100% (1/1), completed with 1 local object.
To https://github.com/Queme24/daniel.git
   b3b9500..3df2bd1  main -> main

daniel@PC-MINGW64 ~/daniel (main)
$ git status
On branch main
Your branch is up to date with 'origin/main'.

nothing to commit, working tree clean

daniel@PC-MINGW64 ~/daniel (main)
```

Hvilke kommandoer brukes for sammenslåing? Hvorfor må man redigere filen manuelt?

For sammenslåing bruker man Git pull og deretter bruker Git merge. Git pull brukes for å hente siste endringer, mens git merge slår sammen endringene. Man må redigere manuelt når det oppstår konflikter. For eksempel hvis flere har gjort redigerings endringer i en fil.

Hva er formålet med Github Pages?

Git pages hjelper utviklere med å lage nettsider. Dette kan brukes til prosjekter eller for eksempel Portfolio.

Dokumentasjon

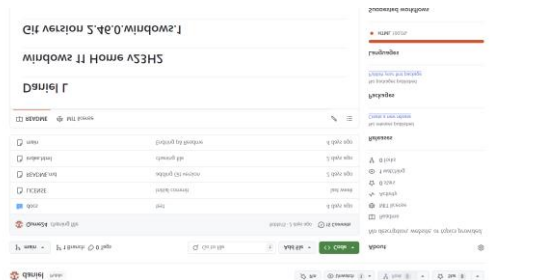
Får ikke tilgang til pages lenger innstillingene lenger. Må ha domain, men navnene jeg oppgir fungerer ikke

Hvorfor er det hensiktsmessig å gjøre endringene i den lokale repository-en og ikke i Github direkte?

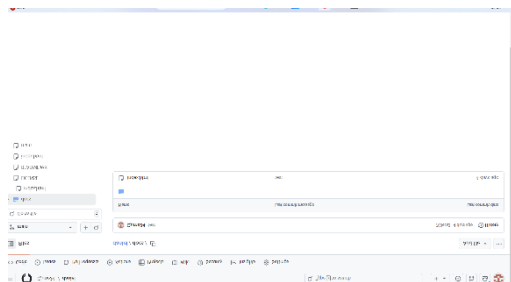
Når man gjør endringer lokalt, kan dette bidra med å redusere risikoen for feil. Ulike tester og endringer kan utøves før man laster opp lokalt i motsetning til å laste opp på GIT hub direkte

Dokumentasjon

Lagde docs mappe.



Lagt inn [index.html](#)



Hvilken type kode er denne? Hvordan og hvor utfør denne koden?

Det er en så kalt HTML kode som hovedsakelig brukes for å organisere innholdet på en nettside. Koden utøves på en nettleser.

Dokumentasjon

Lagt inn koden i [index.html](#) på sublime og lagret

```

1 <!DOCTYPE html>
2 <html>
3 <head>
4 <meta charset="utf-8">
5 <meta name="viewport" content="width=device-width, initial-scale=1">
6 <title>(Daniel [Gundersen]* Lelienhof)</title>
7 </head>
8 <body>
9 <h1>Min IS-114 portfolio</h1>
10 <p>Her skriver jeg portfolio git checkout main
11 </body>
12 </html>

```

pusher det inn i git hub. Com

Hva betyr det at en webside er tilgjengelig på Internett?

Det betyr at alle har tilgang til å se og bruke innholdet som de ønsker

Dokumentasjon

Lagrer endringene og bruker push kommando

```

1 git checkout main
2 git pull
3 git status
4 git add .
5 git commit -m "Her skriver jeg portfolio git checkout main"
6 git push
7
8 git checkout main
9 git pull
10 git status
11 git add .
12 git commit -m "Her skriver jeg portfolio git checkout main"
13 git push
14
15 git checkout main
16 git pull
17 git status
18 git add .
19 git commit -m "Her skriver jeg portfolio git checkout main"
20 git push
21
22 git checkout main
23 git pull
24 git status
25 git add .
26 git commit -m "Her skriver jeg portfolio git checkout main"
27 git push
28
29 git checkout main
30 git pull
31 git status
32 git add .
33 git commit -m "Her skriver jeg portfolio git checkout main"
34 git push
35
36 git checkout main
37 git pull
38 git status
39 git add .
40 git commit -m "Her skriver jeg portfolio git checkout main"
41 git push
42
43 git checkout main
44 git pull
45 git status
46 git add .
47 git commit -m "Her skriver jeg portfolio git checkout main"
48 git push
49
50 git checkout main
51 git pull
52 git status
53 git add .
54 git commit -m "Her skriver jeg portfolio git checkout main"
55 git push
56
57 git checkout main
58 git pull
59 git status
60 git add .
61 git commit -m "Her skriver jeg portfolio git checkout main"
62 git push
63
64 git checkout main
65 git pull
66 git status
67 git add .
68 git commit -m "Her skriver jeg portfolio git checkout main"
69 git push
70
71 git checkout main
72 git pull
73 git status
74 git add .
75 git commit -m "Her skriver jeg portfolio git checkout main"
76 git push
77
78 git checkout main
79 git pull
80 git status
81 git add .
82 git commit -m "Her skriver jeg portfolio git checkout main"
83 git push
84
85 git checkout main
86 git pull
87 git status
88 git add .
89 git commit -m "Her skriver jeg portfolio git checkout main"
90 git push
91
92 git checkout main
93 git pull
94 git status
95 git add .
96 git commit -m "Her skriver jeg portfolio git checkout main"
97 git push
98
99 git checkout main
100 git pull
101 git status
102 git add .
103 git commit -m "Her skriver jeg portfolio git checkout main"
104 git push
105
106 git checkout main
107 git pull
108 git status
109 git add .
110 git commit -m "Her skriver jeg portfolio git checkout main"
111 git push
112
113 git checkout main
114 git pull
115 git status
116 git add .
117 git commit -m "Her skriver jeg portfolio git checkout main"
118 git push
119
120 git checkout main
121 git pull
122 git status
123 git add .
124 git commit -m "Her skriver jeg portfolio git checkout main"
125 git push
126
127 git checkout main
128 git pull
129 git status
130 git add .
131 git commit -m "Her skriver jeg portfolio git checkout main"
132 git push
133
134 git checkout main
135 git pull
136 git status
137 git add .
138 git commit -m "Her skriver jeg portfolio git checkout main"
139 git push
140
141 git checkout main
142 git pull
143 git status
144 git add .
145 git commit -m "Her skriver jeg portfolio git checkout main"
146 git push
147
148 git checkout main
149 git pull
150 git status
151 git add .
152 git commit -m "Her skriver jeg portfolio git checkout main"
153 git push
154
155 git checkout main
156 git pull
157 git status
158 git add .
159 git commit -m "Her skriver jeg portfolio git checkout main"
160 git push
161
162 git checkout main
163 git pull
164 git status
165 git add .
166 git commit -m "Her skriver jeg portfolio git checkout main"
167 git push
168
169 git checkout main
170 git pull
171 git status
172 git add .
173 git commit -m "Her skriver jeg portfolio git checkout main"
174 git push
175
176 git checkout main
177 git pull
178 git status
179 git add .
180 git commit -m "Her skriver jeg portfolio git checkout main"
181 git push
182
183 git checkout main
184 git pull
185 git status
186 git add .
187 git commit -m "Her skriver jeg portfolio git checkout main"
188 git push
189
190 git checkout main
191 git pull
192 git status
193 git add .
194 git commit -m "Her skriver jeg portfolio git checkout main"
195 git push
196
197 git checkout main
198 git pull
199 git status
200 git add .
201 git commit -m "Her skriver jeg portfolio git checkout main"
202 git push
203
204 git checkout main
205 git pull
206 git status
207 git add .
208 git commit -m "Her skriver jeg portfolio git checkout main"
209 git push
210
211 git checkout main
212 git pull
213 git status
214 git add .
215 git commit -m "Her skriver jeg portfolio git checkout main"
216 git push
217
218 git checkout main
219 git pull
220 git status
221 git add .
222 git commit -m "Her skriver jeg portfolio git checkout main"
223 git push
224
225 git checkout main
226 git pull
227 git status
228 git add .
229 git commit -m "Her skriver jeg portfolio git checkout main"
230 git push
231
232 git checkout main
233 git pull
234 git status
235 git add .
236 git commit -m "Her skriver jeg portfolio git checkout main"
237 git push
238
239 git checkout main
240 git pull
241 git status
242 git add .
243 git commit -m "Her skriver jeg portfolio git checkout main"
244 git push
245
246 git checkout main
247 git pull
248 git status
249 git add .
250 git commit -m "Her skriver jeg portfolio git checkout main"
251 git push
252
253 git checkout main
254 git pull
255 git status
256 git add .
257 git commit -m "Her skriver jeg portfolio git checkout main"
258 git push
259
260 git checkout main
261 git pull
262 git status
263 git add .
264 git commit -m "Her skriver jeg portfolio git checkout main"
265 git push
266
267 git checkout main
268 git pull
269 git status
270 git add .
271 git commit -m "Her skriver jeg portfolio git checkout main"
272 git push
273
274 git checkout main
275 git pull
276 git status
277 git add .
278 git commit -m "Her skriver jeg portfolio git checkout main"
279 git push
280
281 git checkout main
282 git pull
283 git status
284 git add .
285 git commit -m "Her skriver jeg portfolio git checkout main"
286 git push
287
288 git checkout main
289 git pull
290 git status
291 git add .
292 git commit -m "Her skriver jeg portfolio git checkout main"
293 git push
294
295 git checkout main
296 git pull
297 git status
298 git add .
299 git commit -m "Her skriver jeg portfolio git checkout main"
300 git push
301
302 git checkout main
303 git pull
304 git status
305 git add .
306 git commit -m "Her skriver jeg portfolio git checkout main"
307 git push
308
309 git checkout main
310 git pull
311 git status
312 git add .
313 git commit -m "Her skriver jeg portfolio git checkout main"
314 git push
315
316 git checkout main
317 git pull
318 git status
319 git add .
320 git commit -m "Her skriver jeg portfolio git checkout main"
321 git push
322
323 git checkout main
324 git pull
325 git status
326 git add .
327 git commit -m "Her skriver jeg portfolio git checkout main"
328 git push
329
330 git checkout main
331 git pull
332 git status
333 git add .
334 git commit -m "Her skriver jeg portfolio git checkout main"
335 git push
336
337 git checkout main
338 git pull
339 git status
340 git add .
341 git commit -m "Her skriver jeg portfolio git checkout main"
342 git push
343
344 git checkout main
345 git pull
346 git status
347 git add .
348 git commit -m "Her skriver jeg portfolio git checkout main"
349 git push
350
351 git checkout main
352 git pull
353 git status
354 git add .
355 git commit -m "Her skriver jeg portfolio git checkout main"
356 git push
357
358 git checkout main
359 git pull
360 git status
361 git add .
362 git commit -m "Her skriver jeg portfolio git checkout main"
363 git push
364
365 git checkout main
366 git pull
367 git status
368 git add .
369 git commit -m "Her skriver jeg portfolio git checkout main"
370 git push
371
372 git checkout main
373 git pull
374 git status
375 git add .
376 git commit -m "Her skriver jeg portfolio git checkout main"
377 git push
378
379 git checkout main
380 git pull
381 git status
382 git add .
383 git commit -m "Her skriver jeg portfolio git checkout main"
384 git push
385
386 git checkout main
387 git pull
388 git status
389 git add .
390 git commit -m "Her skriver jeg portfolio git checkout main"
391 git push
392
393 git checkout main
394 git pull
395 git status
396 git add .
397 git commit -m "Her skriver jeg portfolio git checkout main"
398 git push
399
400 git checkout main
401 git pull
402 git status
403 git add .
404 git commit -m "Her skriver jeg portfolio git checkout main"
405 git push
406
407 git checkout main
408 git pull
409 git status
410 git add .
411 git commit -m "Her skriver jeg portfolio git checkout main"
412 git push
413
414 git checkout main
415 git pull
416 git status
417 git add .
418 git commit -m "Her skriver jeg portfolio git checkout main"
419 git push
420
421 git checkout main
422 git pull
423 git status
424 git add .
425 git commit -m "Her skriver jeg portfolio git checkout main"
426 git push
427
428 git checkout main
429 git pull
430 git status
431 git add .
432 git commit -m "Her skriver jeg portfolio git checkout main"
433 git push
434
435 git checkout main
436 git pull
437 git status
438 git add .
439 git commit -m "Her skriver jeg portfolio git checkout main"
440 git push
441
442 git checkout main
443 git pull
444 git status
445 git add .
446 git commit -m "Her skriver jeg portfolio git checkout main"
447 git push
448
449 git checkout main
450 git pull
451 git status
452 git add .
453 git commit -m "Her skriver jeg portfolio git checkout main"
454 git push
455
456 git checkout main
457 git pull
458 git status
459 git add .
460 git commit -m "Her skriver jeg portfolio git checkout main"
461 git push
462
463 git checkout main
464 git pull
465 git status
466 git add .
467 git commit -m "Her skriver jeg portfolio git checkout main"
468 git push
469
470 git checkout main
471 git pull
472 git status
473 git add .
474 git commit -m "Her skriver jeg portfolio git checkout main"
475 git push
476
477 git checkout main
478 git pull
479 git status
480 git add .
481 git commit -m "Her skriver jeg portfolio git checkout main"
482 git push
483
484 git checkout main
485 git pull
486 git status
487 git add .
488 git commit -m "Her skriver jeg portfolio git checkout main"
489 git push
490
491 git checkout main
492 git pull
493 git status
494 git add .
495 git commit -m "Her skriver jeg portfolio git checkout main"
496 git push
497
498 git checkout main
499 git pull
500 git status
501 git add .
502 git commit -m "Her skriver jeg portfolio git checkout main"
503 git push
504
505 git checkout main
506 git pull
507 git status
508 git add .
509 git commit -m "Her skriver jeg portfolio git checkout main"
510 git push
511
512 git checkout main
513 git pull
514 git status
515 git add .
516 git commit -m "Her skriver jeg portfolio git checkout main"
517 git push
518
519 git checkout main
520 git pull
521 git status
522 git add .
523 git commit -m "Her skriver jeg portfolio git checkout main"
524 git push
525
526 git checkout main
527 git pull
528 git status
529 git add .
530 git commit -m "Her skriver jeg portfolio git checkout main"
531 git push
532
533 git checkout main
534 git pull
535 git status
536 git add .
537 git commit -m "Her skriver jeg portfolio git checkout main"
538 git push
539
540 git checkout main
541 git pull
542 git status
543 git add .
544 git commit -m "Her skriver jeg portfolio git checkout main"
545 git push
546
547 git checkout main
548 git pull
549 git status
550 git add .
551 git commit -m "Her skriver jeg portfolio git checkout main"
552 git push
553
554 git checkout main
555 git pull
556 git status
557 git add .
558 git commit -m "Her skriver jeg portfolio git checkout main"
559 git push
560
561 git checkout main
562 git pull
563 git status
564 git add .
565 git commit -m "Her skriver jeg portfolio git checkout main"
566 git push
567
568 git checkout main
569 git pull
570 git status
571 git add .
572 git commit -m "Her skriver jeg portfolio git checkout main"
573 git push
574
575 git checkout main
576 git pull
577 git status
578 git add .
579 git commit -m "Her skriver jeg portfolio git checkout main"
580 git push
581
582 git checkout main
583 git pull
584 git status
585 git add .
586 git commit -m "Her skriver jeg portfolio git checkout main"
587 git push
588
589 git checkout main
590 git pull
591 git status
592 git add .
593 git commit -m "Her skriver jeg portfolio git checkout main"
594 git push
595
596 git checkout main
597 git pull
598 git status
599 git add .
600 git commit -m "Her skriver jeg portfolio git checkout main"
601 git push
602
603 git checkout main
604 git pull
605 git status
606 git add .
607 git commit -m "Her skriver jeg portfolio git checkout main"
608 git push
609
610 git checkout main
611 git pull
612 git status
613 git add .
614 git commit -m "Her skriver jeg portfolio git checkout main"
615 git push
616
617 git checkout main
618 git pull
619 git status
620 git add .
621 git commit -m "Her skriver jeg portfolio git checkout main"
622 git push
623
624 git checkout main
625 git pull
626 git status
627 git add .
628 git commit -m "Her skriver jeg portfolio git checkout main"
629 git push
630
631 git checkout main
632 git pull
633 git status
634 git add .
635 git commit -m "Her skriver jeg portfolio git checkout main"
636 git push
637
638 git checkout main
639 git pull
640 git status
641 git add .
642 git commit -m "Her skriver jeg portfolio git checkout main"
643 git push
644
645 git checkout main
646 git pull
647 git status
648 git add .
649 git commit -m "Her skriver jeg portfolio git checkout main"
650 git push
651
652 git checkout main
653 git pull
654 git status
655 git add .
656 git commit -m "Her skriver jeg portfolio git checkout main"
657 git push
658
659 git checkout main
660 git pull
661 git status
662 git add .
663 git commit -m "Her skriver jeg portfolio git checkout main"
664 git push
665
666 git checkout main
667 git pull
668 git status
669 git add .
670 git commit -m "Her skriver jeg portfolio git checkout main"
671 git push
672
673 git checkout main
674 git pull
675 git status
676 git add .
677 git commit -m "Her skriver jeg portfolio git checkout main"
678 git push
679
680 git checkout main
681 git pull
682 git status
683 git add .
684 git commit -m "Her skriver jeg portfolio git checkout main"
685 git push
686
687 git checkout main
688 git pull
689 git status
690 git add .
691 git commit -m "Her skriver jeg portfolio git checkout main"
692 git push
693
694 git checkout main
695 git pull
696 git status
697 git add .
698 git commit -m "Her skriver jeg portfolio git checkout main"
699 git push
700
701 git checkout main
702 git pull
703 git status
704 git add .
705 git commit -m "Her skriver jeg portfolio git checkout main"
706 git push
707
708 git checkout main
709 git pull
710 git status
711 git add .
712 git commit -m "Her skriver jeg portfolio git checkout main"
713 git push
714
715 git checkout main
716 git pull
717 git status
718 git add .
719 git commit -m "Her skriver jeg portfolio git checkout main"
720 git push
721
722 git checkout main
723 git pull
724 git status
725 git add .
726 git commit -m "Her skriver jeg portfolio git checkout main"
727 git push
728
729 git checkout main
730 git pull
731 git status
732 git add .
733 git commit -m "Her skriver jeg portfolio git checkout main"
734 git push
735
736 git checkout main
737 git pull
738 git status
739 git add .
740 git commit -m "Her skriver jeg portfolio git checkout main"
741 git push
742
743 git checkout main
744 git pull
745 git status
746 git add .
747 git commit -m "Her skriver jeg portfolio git checkout main"
748 git push
749
750 git checkout main
751 git pull
752 git status
753 git add .
754 git commit -m "Her skriver jeg portfolio git checkout main"
755 git push
756
757 git checkout main
758 git pull
759 git status
760 git add .
761 git commit -m "Her skriver jeg portfolio git checkout main"
762 git push
763
764 git checkout main
765 git pull
766 git status
767 git add .
768 git commit -m "Her skriver jeg portfolio git checkout main"
769 git push
770
771 git checkout main
772 git pull
773 git status
774 git add .
775 git commit -m "Her skriver jeg portfolio git checkout main"
776 git push
777
778 git checkout main
779 git pull
780 git status
781 git add .
782 git commit -m "Her skriver jeg portfolio git checkout main"
783 git push
784
785 git checkout main
786 git pull
787 git status
788 git add .
789 git commit -m "Her skriver jeg portfolio git checkout main"
790 git push
791
792 git checkout main
793 git pull
794 git status
795 git add .
796 git commit -m "Her skriver jeg portfolio git checkout main"
797 git push
798
799 git checkout main
800 git pull
801 git status
802 git add .
803 git commit -m "Her skriver jeg portfolio git checkout main"
804 git push
805
806 git checkout main
807 git pull
808 git status
809 git add .
810 git commit -m "Her skriver jeg portfolio git checkout main"
811 git push
812
813 git checkout main
814 git pull
815 git status
816 git add .
817 git commit -m "Her skriver jeg portfolio git checkout main"
818 git push
819
820 git checkout main
821 git pull
822 git status
823 git add .
824 git commit -m "Her skriver jeg portfolio git checkout main"
825 git push
826
827 git checkout main
828 git pull
829 git status
830 git add .
831 git commit -m "Her skriver jeg portfolio git checkout main"
832 git push
833
834 git checkout main
835 git pull
836 git status
837 git add .
838 git commit -m "Her skriver jeg portfolio git checkout main"
839 git push
840
841 git checkout main
842 git pull
843 git status
844 git add .
845 git commit -m "Her skriver jeg portfolio git checkout main"
846 git push
847
848 git checkout main
849 git pull
850 git status
851 git add .
852 git commit -m "Her skriver jeg portfolio git checkout main"
853 git push
854
855 git checkout main
856 git pull
857 git status
858 git add .
859 git commit -m "Her skriver jeg portfolio git checkout main"
860 git push
861
862 git checkout main
863 git pull
864 git status
865 git add .
866 git commit -m "Her skriver jeg portfolio git checkout main"
867 git push
868
869 git checkout main
870 git pull
871 git status
872 git add .
873 git commit -m "Her skriver jeg portfolio git checkout main"
874 git push
875
876 git checkout main
877 git pull
878 git status
879 git add .
880 git commit -m "Her skriver jeg portfolio git checkout main"
881 git push
882
883 git checkout main
884 git pull
885 git status
886 git add .
887 git commit -m "Her skriver jeg portfolio git checkout main"
888 git push
889
890 git checkout main
891 git pull
892 git status
893 git add .
894 git commit -m "Her skriver jeg portfolio git checkout main"
895 git push
896
897 git checkout main
898 git pull
899 git status
900 git add .
901 git commit -m "Her skriver jeg portfolio git checkout main"
902 git push
903
904 git checkout main
905 git pull
906 git status
907 git add .
908 git commit -m "Her skriver jeg portfolio git checkout main"
909 git push
910
911 git checkout main
912 git pull
913 git status
914 git add .
915 git commit -m "Her skriver jeg portfolio git checkout main"
916 git push
917
918 git checkout main
919 git pull
920 git status
921 git add .
922 git commit -m "Her skriver jeg portfolio git checkout main"
923 git push
924
925 git checkout main
926 git pull
927 git status
928 git add .
929 git commit -m "Her skriver jeg portfolio git checkout main"
930 git push
931
932 git checkout main
933 git pull
934 git status
935 git add .
936 git commit -m "Her skriver jeg portfolio git checkout main"
937 git push
938
939 git checkout main
940 git pull
941 git status
942 git add .
943 git commit -m "Her skriver jeg portfolio git checkout main"
944 git push
945
946 git checkout main
947 git pull
948 git status
949 git add .
950 git commit -m "Her skriver jeg portfolio git checkout main"
951 git push
952
953 git checkout main
954 git pull
955 git status
956 git add .
957 git commit -m "Her skriver jeg portfolio git checkout main"
958 git push
959
960 git checkout main
961 git pull
962 git status
963 git add .
964 git commit -m "Her skriver jeg portfolio git checkout main"
965 git push
966
967 git checkout main
968 git pull
969 git status
970 git add .
971 git commit -m "Her skriver jeg portfolio git checkout main"
972 git push
973
974 git checkout main
975 git pull
976 git status
977 git add .
978 git commit -m "Her skriver jeg portfolio git checkout main"
979 git push
980
981 git checkout main
982 git pull
983 git status
984 git add .
985 git commit -m "Her skriver jeg portfolio git checkout main"
986 git push
987
988 git checkout main
989 git pull
990 git status
991 git add .
992 git commit -m "Her skriver jeg portfolio git checkout main"
993 git push
994
995 git checkout main
996 git pull
997 git status
998 git add .
999 git commit -m "Her skriver jeg portfolio git checkout main"
1000 git push

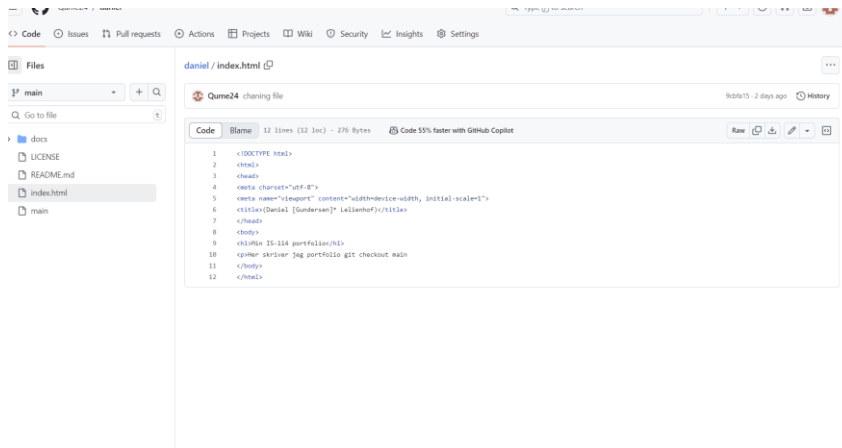
```

Er denne fremgangsmåten – å redigere filer direkte i grenen main – optimal? Begrunn svaret.

Det er ikke samarbeid, fordi det kan oppstå feil med versjonskontroll og samarbeid, på grunn av at man ikke har hatt muligheten til å teste endringene før de har blitt en del av hovedkoden. I disse tilfellene kan det være bedre å bruke separate grener for testing.

Dokumentasjon

Lagt inn navn og fikset følgende feil <</head> til </head>



Hva er fast-forward?’

Fast forward er en type merge som git gjør når den skal håndtere spesifikke merging av grener

Dokumentasjon

Har allerede gjort merge prosessen, glemte å ta bildet av det.

```
danie@Pc MINGW64 ~
$ git checkout main
fatal: not a git repository (or any of the parent directories): .git

danie@Pc MINGW64 ~
$ cd daniel

danie@Pc MINGW64 ~/daniel (main)
$ git checkout main
Already on 'main'
Your branch is up to date with 'origin/main'.

danie@Pc MINGW64 ~/daniel (main)
$ git merge
Already up to date.

danie@Pc MINGW64 ~/daniel (main)
$ |
```

Reflekter rundt tegningen av "commit-tre".

Tegningen commit tre er en visuell visning av historikken til prosjekter. Den viser for eksempel hvordan commits blir strukturert og hvordan grener blir sammenslått ved å merge commits. Å ha tilgang til en slik historikk gjør det enklere å finne frem til prosjekt endringer og feil søke problemer

Kilder

OpenAI. (2024). *ChatGPT* (Version 4) [Generative pre-trained transformer].
<https://chat.openai.com/>