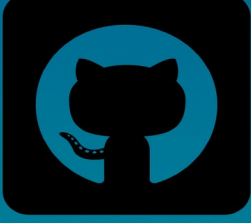


எளிய தமிழில் கிட்
(Git)



கி. முத்துராமலிங்கம்

எளிய

தமிழில் கிட்(**Git**)

கி. முத்துராமலிங்கம்

நூல் : எளிய தமிழில்
சாப்ட்வேர் டெஸ்டிங்
ஆசிரியர் :
கி.முத்துராமலிங்கம்

அட்டைப்படம்
மின்னுலாக்கம் : அ.ஷேக்
அலாவுதீன்

தமிழ் இ

சர்வீஸ்,

tamilservice17@gmail.com

வெளியீடு :
FreeTamilEbooks.com

உரிமை : CC-BY-SA

பொருளடக்கம்

எளிய தமிழில் கிட்(Git).....9

கிட் – கற்றுக்கொள்வோம் வாருங்கள்!.....18

கிட்டை விண்டோசில் நிறுவுவது எப்படி?.....	25
கிட் நிறுவி முடித்ததை எப்படிச் சோதிப்பது?..	33
கிட் ஹப்பில் நம் பெயர், மின்னஞ்சல் முகவரி சேர்ப்பது எப்படி?.....	35
நம் அடைவை(ஃபோல்டர்) கிட் அடைவாக மாற்றுவது எப்படி?.....	40
கிட் அடைவில் கோப்பு உருவாக்குவது எப்படி?	42
கிட் அடைவில் உள்ள கோப்பை கிட் மூலம் கண்காணிப்பது எப்படி?.....	45
கோப்புப் பற்றிய குறிப்பு(log) என்பது என்ன?	51
git commit எப்படிக் கொடுப்பது?.....	54

ஏற்கெனவே commit ஆன கோப்பில்
மாற்றங்கள் செய்வது எப்படி?.....57

கோப்பில் என்னென்ன
மாற்றப்பட்டிருக்கின்றன என்று எப்படிப்
பார்ப்பது?.....61

விரும்பும் பதிப்பை(வெர்ஷனை)ப் பார்ப்பது
எப்படி?.....67

‘கிட்ஹப்’பில் இருந்து
திட்டப்பணி(பிராஜெக்ட்)கள் எடுப்பது எப்படி?
.....72

முன்னேற்பாடுகள்.....74

களஞ்சியம் என்றால் என்ன என்று தெரியுமா?.76

முள்கரண்டியால் எப்படிக் களஞ்சியத்தில்
இருந்து எடுப்பது?.....78

ஃபோர்ட் செய்யப்பட்ட திட்டப்பணி(பிராஜெக்ட்)யை குளோன்(clone) செய்வது எப்படி?.....	82
புதிய கிளை ஆரம்பம்.....	85
புதிய கிளை(பிராஞ்ச்)யை உருவாக்குவது எப்படி?.....	86
புதிய கிளைக்குத் தாவுவது எப்படி?.....	88
கோப்பைப் புதிய கிளையில் திறந்து மாற்றுவது?.....	89
git push படிக்கும் நேரம் வந்து விட்டது!....	91
சுருக்க வழி:.....	93
முக்கிய குறிப்பு:.....	94
கதை எப்படிப் போகிறது?.....	96

வேறொருவர் களஞ்சியத்தை(ரெப்போ)
முள்கரண்டி(ஃபோர்க்) மூலம் எடுக்க.....98

Pull Request கட்டளையைப் பார்ப்போமா? 104

Rebase கட்டளை.....108

Rebase என்றால் என்ன?.....108

‘கிட்’டில் SSH Key எப்படி உருவாக்குவது?. 116

துணை நின்ற தளங்கள்.....119

எளிய தமிழில் கிட்(Git)

சாப்ட்வேர்துறை என்றால் ஜாவா, டாட்நெட், பைத்தான், லினக்ஸ், ஆண்டிராய்டு – இப்படித்தானே சொல்லிக் கேள்விப்பட்டிருக்கிறோம். இதென்ன புதிதாக ‘கிட்’(Git)? என்று நீங்கள் யோசித்தால் – உங்களுக்கானது இந்தச் சின்ன புத்தகம். கிட் என்பது புரோகிராமிங் மொழியா? என்றால் ‘இல்லை’. ஓ! இது ஓர் இயங்கு தளமா(OS) என்றால் ‘அதுவும் இல்லை’. இல்லையா? அப்படியானால் இது டெஸ்டிங் கருவி(tool)யா என்றால் ‘மஹும், இல்லவே இல்லை’.

எதுவுமே இல்லையா? அப்படியானால் ‘கிட்’
என்றால் என்ன?

கிட் என்றால் என்ன என்று சொல்லி
விடுகிறேன். கிட் என்பது பதிப்புகளுக்கு
இடையே இருக்கும் வேற்றுமைகளைக்
களைந்து ஒருமைப்படுத்தும் ஒரு கருவி –
அவ்வளவு தான்! “இப்படிச் சொல்வதற்கு,
சொல்லாமலே இருக்கலாம். ஒன்றுமே
புரியவில்லை” என்று நீங்கள் நினைக்கிறீர்களா?
உங்கள் நினைப்புச் சரி தான்! மேல் சொன்ன
வரியைக்(பதிப்புகளுக்கு இடையே இருக்கும்
வேற்றுமைகளைக் களைந்து ஒருமைப்படுத்தும்
ஒரு கருவி) கொஞ்சம் விளக்கமாகப் பார்த்து
விடுவோம்.

முதலில் பதிப்பு என்றால் என்ன
என்று பார்ப்போம். ஒரு
மென்பொருள்(அதாங்க, சாப்ட்வேர்)

வெளியாகிறது என்று வைத்துக் கொள்ளுங்கள்.
நம் எல்லோருக்கும் தெரிந்த ஆண்டிராய்டில்
வெர்ஷன் என்று ஒவ்வொரு இனிப்புப் பெயரை
வைத்துக் கொள்கிறார்கள் அல்லவா? அது தான்
பதிப்பு! ஆங்கிலத்தின் வெர்ஷன்(version)!

இப்படி ஒரு பதிப்பை ஒவ்வொரு
மென்பொருளும் வெளியிடும் அல்லவா?
அந்தப் பதிப்புக்கு ஒரே ஒரு மென்பொறியாளரா
உட்கார்ந்து வேலை செய்திருப்பார்? இல்லை
தானே! பல மென்பொறியாளர்கள் சேர்ந்து
உருவாக்கும் ஒரு கூட்டுப் படைப்புத் தான்
மென்பொருள் – சரி தானே! இப்படிப்பட்ட
மென்பொருளை ஒவ்வொரு
மென்பொறியாளரும் எங்கே
உருவாக்கியிருப்பார்? “இதென்ன கேள்வி –
அவரவர் கணினியில் தான்!” என்கிறீர்களா? சரி
தான் நீங்கள் சொல்வது!

அது சரி! இப்போது இன்னொரு
கேள்வி – இப்படி – அவரவர் கணினியில்
உருவாக்கப்பட்டிருக்கும் நிரல்(அதாங்க,
புரோகிராம்)களை எல்லாம் ஒரிடத்தில் சேர்க்க
வேண்டாமா? “அதற்குத் தான், கூகுள் டிரைவ் –
போல ஏதாவது ஒரிடத்தில் எல்லோருடைய
நிரலையும் வைத்துக் கொள்ளலாமே!” என்று
நீங்கள் நினைத்தால் – இப்போதும் நீங்கள்
சொல்வது சரி தான்! ஆனால் திரும்பவும் இதில்
ஒரு கேள்வி வந்து விடுகிறது.

சில நேரங்களில் ஒரே
கோப்பில்(ஃபைல்) ஒன்றுக்கும் அதிகமான
பொறியாளர்கள் வேலை செய்ய வேண்டியது
வருமே! அப்போது ஒருவர் செய்யும்
மாற்றங்களை இன்னொருவர் எப்படித் தெரிந்து
கொள்வார்?

ஒரு கதை பேசுவோமா? நாம்
எல்லோரும் சேர்ந்து இரண்டு நாள் இன்பச்

சுற்றுலா போகலாம் என்று திட்டமிடுவதாக
வைத்துக் கொள்ளுங்கள். இன்பச் சுற்றுலாவுக்கு
வரும் நண்பர்கள் அனைவரையும் – கூகுள்
டிரைவ் பக்கம் ஒன்றைக் கொடுத்து – இதில்

1) எந்த ஊருக்குப் போகலாம்

2) எப்போது போகலாம்

3) எப்படிப் போகலாம்

என்று பதியச் சொல்லி விடலாம்.
முதல் நண்பர் அவருடைய திட்டத்தைப் பதிந்து
விடுகிறார்; பதிந்து விட்டு சின்ன வேலையாக
வெளியே போய் விடுகிறார். இரண்டாவது
நண்பர், முதல் நண்பரின் திட்டத்தில் எப்போது
போகலாம் என்பதில் சின்னச் சின்ன
திருத்தங்களைச் செய்கிறார். இப்போது
மூன்றாவது நண்பர் – அதே இடத்தில் 'எப்படிப்
போகலாம்' என்பதில் மாற்றங்கள் செய்கிறார்.
இப்படிப்பட்ட சூழலில் – முதல் நண்பர்-

நம்முடைய திட்டம் தான், கூகுள் டிரைவில் இருக்கிறதே! எப்போது வேண்டுமானாலும் பார்த்துக் கொள்ளலாம் – என்று தானே நினைத்துக் கொண்டிருப்பார்? ஆனால், அவருடைய திட்டமோ, பல மாற்றங்களுக்கு ஆளாகியிருக்கும் அல்லவா?

இது போன்ற சூழலில்,

1) ஒவ்வொருவர் பதிவதையும் ஒரு பதிப்பாக எடுத்து வைத்து

அ. என்ன பதிகிறார்

ஆ. எப்போது பதிகிறார்

என்பதைக் குறித்து வைத்துக்
கொண்டு

2) இன்னொருவர் வந்து முன்னவரின் கருத்தை மாற்றினால்

அ. இரண்டாமவர் – என்ன
சேர்த்திருக்கிறார், என்ன நீக்கியிருக்கிறார்
என்பதையும்

ஆ. எப்போது மாற்றினார்
என்பதையும்

குறித்து அதையும் ஒரு பதிப்பாக
வைத்துக் கொண்டால் எவ்வளவு வசதியாக
இருக்கும்? முதலில் பதிந்ததே
வேண்டுமானாலும் சரி, இரண்டாமவர் சேர்த்தது
மட்டும் வேண்டும் என்றாலும் சரி, எப்படி
வேண்டுமானாலும் – வேண்டும் பதிப்பை
எடுத்துக் கொள்ளலாம் அல்லவா? இதைத் தான்
பதிப்புக் கட்டுப்பாடு (version control) என்று
சொல்கிறார்கள்.

மென்பொருள் துறையில்
இப்படிப்பட்ட பதிப்புக் கட்டுப்பாட்டுக்

கருவிகள்(Version Control Systems) பல இருக்கின்றன. அதில் குறிப்பிடத்தக்க ஒன்று தான் – கிட்(Git) ஆகும். அப்படி என்ன 'குறிப்பிடத்தக்க விடயங்கள்' கிட்டில் இருக்கின்றன என்கிறீர்களா?

- நாம் எல்லோருக்கும் பிடித்த ஒன்று – கட்டற்ற மென்பொருள்
- இலவசம் – எவ்வளவு கோப்புகள் வேண்டுமானாலும் வைத்துக் கொள்ளலாம்.
- வேகமானது
- மிகவும் பாதுகாப்பானது
- சோதித்துப் பார்ப்பதற்குத் தனி மேடை

இப்படிப் பல விடயங்கள் கிட்டைத் தனியொருவனாகத் தூக்கி நிறுத்துகின்றன. இப்படிப்பட்ட ஒரு நல்ல மென்பொருளை

உருவாக்கியவர் யார்? லினக்சை உருவாக்கிய
அதே லினஸ் டோர்வால்ஸ் தான்! சரி,
இப்போது கிட் படிக்கப் போகலாமா?

கிட் – கற்றுக்கொள்வோம் வாருங்கள்!

- 1) முதலில் github.com இல் கணக்குத் தொடங்குங்கள். <https://github.com> க்குப் போய் Sign Up என்பதைச் சொடுக்கவும்.

[https://github.com/join?
source=header-home](https://github.com/join?source=header-home)



Join GitHub

The best way to design, build, and ship software.



Step 1:
Set up your account



Step 2:
Choose your subscription

Create your personal account

Username *

This will be your username. You can add the name of your organization later.

Email address *

We'll occasionally send updates about your account to this inbox. We'll never share your email address with anyone.

Password *

2) மேலே உள்ள விவரங்களைக் கொடுத்த
பிறகு, கீழுள்ள பக்கம் வரும்.

Welcome to GitHub

You're a few steps away from building better software. @panumaitalk

 **Completed**
Set up your account

 **Step 2:**
Choose your subscription

 **Step 3:**
Personalize your experience

Choose your subscription

With tens of millions of developers and the world's largest open source community, there's no wrong choice.



Free

The basics of GitHub for every developer

\$0
per month

Includes:

- ✓ Unlimited public and private repositories
- ✓ 3 collaborators for private repositories
- ✓ Issues and pull requests
- ✓ Project management



Pro

The basics for developers with advanced requirements

\$7
per month
(new \$12.95)

Includes:

- ✓ Unlimited public and private repositories
- ✓ Unlimited collaborators
- ✓ Issues and bug tracking
- ✓ Project management
- ✓ Advanced tools and insights

Are you a student? Get access to the best developer tools for free with the [GitHub Student Developer Pack](#).

☐ Help me set up an organization now

Organizations are separate from personal accounts and are best suited for businesses who need to manage permissions for many employees. [Learn more about organizations](#)

☒ Send me updates on GitHub news, offers, and events

Subscribe anytime in your email preferences. [Learn more](#)

[Continue](#)

3) மேல் உள்ள (Step 2) படத்தில் 'Continue' ஐச் சொடுக்கவும். இப்போது கீழுள்ள பக்கம் வரும்.

Search or jump to... Pull requests Issues Marketplace Explore

Welcome to GitHub

You'll find endless opportunities to learn, code, and create. @psumailtalk.

✓ Completed
Set up a personal account

Step 2
Choose your subscription

Step 3
Tailor your experience

What is your level of programming experience?

☐ None—I don't program at all
☐ New to programming
☐ Somewhat experienced
☐ Very experienced

What do you plan to use GitHub for? (Select up to 3)

☐ Learning to code
☐ Learning Git and GitHub
☐ Meet a project (repository)
☐ Creating a website with GitHub Pages
☐ Collaborating with my team
☐ Finding a project to contribute to
☐ School work / School-related project
☐ The GitHub API
☐ Don't know yet
☐ Other (please specify)

What are you interested in?

What languages, frameworks, industries, or design areas you're interested in?
e.g. 3D, object components, AI

[Continue](#) Skip this step

- 4) இந்தப் பக்கத்தில் உங்களுக்குப் பொருந்தும் தகவல்களைக் கொடுங்கள்.
- 5) பிறகு பச்சை நிற 'Submit' பொத்தானை அழுத்துங்கள்.
- 6) மின்னஞ்சல் முகவரியை உறுதிப்படுத்த, கீழுள்ளவாறு தோன்றும்.



Please verify your email address

Before you can contribute on GitHub, we need you to verify your email address.

An email containing verification instructions was sent to ` @gmail.com.

Didn't get the email? [Resend verification email](#) or [change your email settings](#).

- 7) உங்கள் மின்னஞ்சல் முகவரிக்குப் போய் git இல் இருந்து வந்திருக்கும் மின்னஞ்சலை உறுதிப்படுத்துங்கள். ஒருவேளை inbox இல் இல்லையெனில் Spam ஐயும் பாருங்கள்.

8) இப்போது கிட்ஹப் தளத்தின் முகப்புப் பக்கத்தில் உங்கள் பயனர் பெயர், கடவுச்சொல்லைக் கேட்பார்கள்.



Sign in to GitHub

Username or email address

Password

[Forgot password?](#)

Sign in

கிட்டை விண்டோசில்
நிறுவுவது எப்படி?

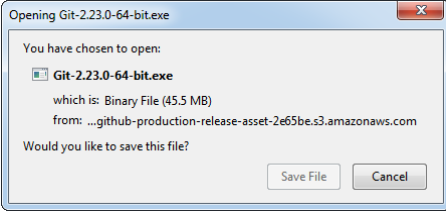
<https://git-scm.com/downloads>

பக்கத்திற்குப் போங்கள்.



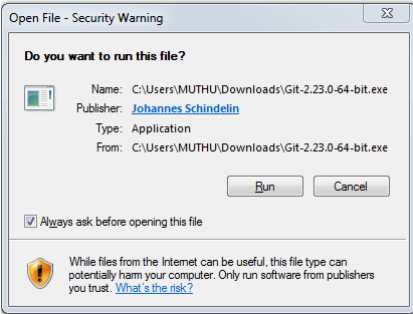
Downloads தலைப்பின் கீழ், Windows
என்பதைச் சொடுக்குங்கள்.

கீழ் உள்ளது போல் ஒரு மேல் மீட்புப் பெட்டி
(பாப் அப்) வரும்.

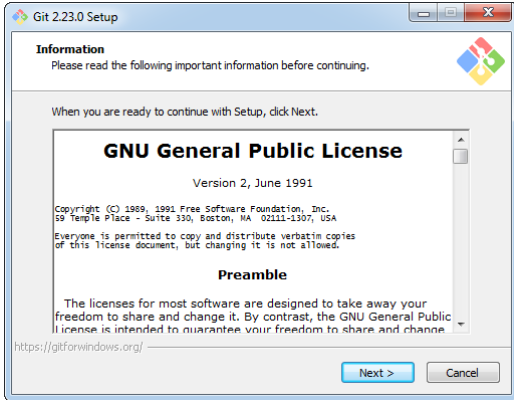


அதில் 'Save File' ஐச் சொடுக்கிப் பதிவிறக்கிக் கொள்ளுங்கள்.

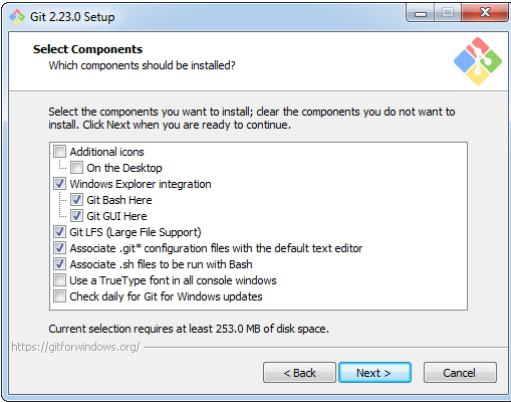
பதிவிறக்கிய கிட் கோப்பை நிறுவுங்கள்.



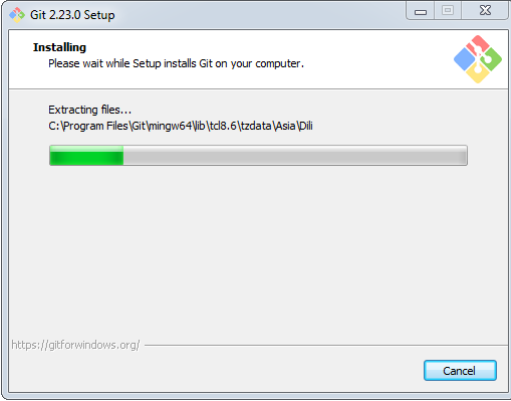
‘Next’ ஐச் சொடுக்குங்கள்.



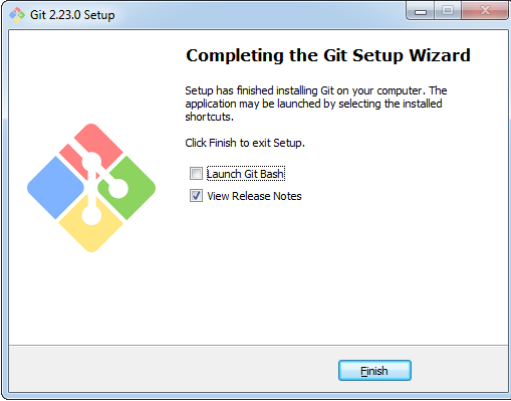
வேறு எந்த மாற்றங்களும் செய்ய வேண்டியதில்லை. ‘Next’ ஐச் சொடுக்கிக் கொண்டே போகலாம்.



பிறகு, கீழ் உள்ளது போல, கிட்டின் நிறுவல்
தொடங்கும்.



கடைசியாக, கீழ் உள்ளது போல் திரை
தோன்றும். அதில் 'Finish' ஐச் சொடுக்குங்கள்.



இப்போது கிட்டின் வெளியீட்டுக் குறிப்புகள் (ரிலீஸ் நோட்ஸ்) உலாவியில் காட்டப்படும். அதைப் பற்றிப் பெரிதாகக் கவலைப்பட வேண்டியதில்லை.

Git for Windows v2.23.0 Release Notes

Latest update: August 17th 2019

Introduction

These release notes describe issues specific to the Git for Windows release. The release notes covering the history of the core git commands can be found [in the Git project](#).

See <http://git-scm.com/> for further details about Git including ports to other operating systems. Git for Windows is hosted at <https://gitforwindows.org/>.

கிட் நிறுவி முடித்ததை எப்படிச் சோதிப்பது?

- 1) Start பொத்தானை அழுத்தி, வரும் மேல் மீட்புப் பெட்டியில், 'Search programs and Files' பெட்டியில் 'git bash' என அச்சிடுங்கள்.
- 2) Git bash பெட்டி வரும். அதைச் சொடுக்குங்கள்.
- 3) இப்போது வரும் git bash கருப்புத் திரையில் git --version என அச்சிடுங்கள்.

MINGW64:/c/Users/MUTHU

```
MUTHU@MUTHU-PC MINGW64 ~  
$ git --version  
git version 2.23.0.windows.1
```

- 4) நீங்கள் நிறுவிய கிட்டின் பதிப்பு என்ன என்பது அந்தக் கருப்புத் திரையில் காண்பிக்கப்படும்.
- 5) சரியாக, கிட் நிறுவப்படவில்லை எனில், கீழ் உள்ளது போல் காண்பிக்கப்படும்.

```
-bash: git: command not found
```

```
'git' is not recognized as an internal or external command, operable program or batch file.
```

- 6) ஒரு வேளை ஐந்தாம் படியில் உள்ளது போல், பிழைச் செய்தி வந்தால் – நீங்கள் கிட்டை மீண்டும் நிறுவ வேண்டும்.
- 7) இப்போது, Start -> git எனத் தட்டச்சிடுங்கள்.
- 8) காட்டப்படும் நிரல்களில் git bash ஐத் தேர்ந்து கொள்ளுங்கள்.

கிட் ஹப்பில் நம் பெயர், மின்னஞ்சல் முகவரி சேர்ப்பது எப்படி?

- 1) நாம் ஏற்கெனவே <https://github.com> இல் தொடங்கிய கணக்கின் பெயர், மின்னஞ்சல் முகவரியை – நம்முடைய கணினியில் உள்ள கிட்டுக்குத் தெரியப் படுத்த வேண்டும். அப்பொழுது தான், இங்கிருந்து நாம் எழுதும் நிரல்களை <https://github.com> க்கு அனுப்பிச் சேமித்து உறுதிப்படுத்த (Commit) முடியும்.
- 2) அதற்காக, நாம் நிறுவிய கிட்ஹப்பில் நம்முடைய

பெயரையும்
மின்னஞ்சல் முகவரியையும் எப்படிச்
சேர்ப்பது எனப் பார்ப்போம்.

3) Git bash இல்

```
git config --global user.name "Your Name"  
git config --global user.email  
"youremail@domain.com"
```

ஆகிய இரண்டையும் கொடுங்கள்.

```
MUTHU@MUTHU-PC MINGW64 ~  
$ git config --global user.name "Muthuramalingam"  
MUTHU@MUTHU-PC MINGW64 ~  
$ git config --global user.email "muthu1809@gmail.com"
```

4) இப்போது, git config --list

கொடுத்து உங்கள் பெயர்,
மின்னஞ்சல் முகவரி தகவல்கள்
ஏற்றுக்கொள்ளப்பட்டிருக்கிறதா என
உறுதிப்படுத்தலாம்.

5) இப்போது உங்கள் இயங்குதளத்தில் எந்த வட்டுக்கு(டிரைவ்)ப் போக

வேண்டும் – (எங்கே போதுமான அளவு நினைவகம் வைத்திருக்கிறீர்கள் என்பதைப் பொறுத்து) என்பதை முடிவு செய்து கொள்ளுங்கள்.

6) இந்த எடுத்துக்காட்டில் நாம் E:\ வட்டிற்குப் போகப் போகிறோம்.

7) Git bash இல் cd E: என்று தட்டச்சிடுங்கள்.

```
MUTHU@MUTHU-PC MINGW64 ~  
$ cd E:
```

8) நீங்கள் விரும்பிய வட்டிற்குப் போன பிறகு, அதை உறுதிப்படுத்த pwd என்று தட்டச்சிடுங்கள். (pwd – என்பது present working directory என்பதன் சுருக்கம்).

```
MUTHU@MUTHU-PC MINGW64 /e  
$ pwd  
/e
```

- 9) இப்போது அந்த வட்டில் ஒரு அடைவை(ஃபோல்டர்) உருவாக்க, mkdir எனத்C தட்டச்சிட்டு, தொடர்ந்து அடைவின் பெயரைக் கொடுங்கள்.

mkdir foldername

- 10) நாம் இங்கு gitLearning என்னும் பெயரில் அடைவு ஒன்றை உருவாக்கியிருக்கிறோம்.

```
MUTHU@MUTHU-PC MINGW64 /e
$ mkdir gitLearning
```

- 11) இப்போது நாம் உருவாக்கியுள்ள gitLearning அடைவுக்கு நகர வேண்டும். அப்படி நகர்வதற்கு cd என்று கொடுத்து அடைவின் பெயரைக் கொடுத்தால் போதும்.

(cd என்பது change directory என்பதன் சுருக்கம்). இங்கு நாம் cd gitLearning என்று கொடுத்திருக்கிறோம்.

```
MUTHU@MUTHU-PC MINGW64 /e
$ cd gitLearning

MUTHU@MUTHU-PC MINGW64 /e/gitLearning
$
```

நம் அடைவை(ஃபோல்டர்) கிட் அடைவாக மாற்றுவது எப்படி?

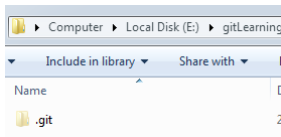
இப்போது, இந்த அடைவை கிட் அடைவாக மாற்ற வேண்டும். அதாவது, நம்முடைய கணினியில் உள்ள இந்த அடைவில் நாம் என்ன மாற்றம் செய்தாலும் அது github.com இல் உள்ள நம்முடைய தொகுப்பில் (ரெப்பாசிட்டரியில்) மாற வேண்டும். அதற்கு,

1) git init என்று தட்டச்சிடுங்கள்.

```
MUTHU@MUTHU-PC MINGW64 /e/gitLearning
$ git init
Initialized empty Git repository in E:/gitLearning/.git/
```

2) உடனே, மேலே இருப்பது போல், Initialized empty Git repository in என வரும்.

3) இப்போது அந்த அடைவைத் திறந்து பார்த்தால் கீழ் உள்ளது போல, .git அடைவு(ஃபோல்டர்) ஒன்று உங்கள் அடைவு(ஃபோல்டர்)க்கு உள்ளே உருவாகியிருக்கும்.



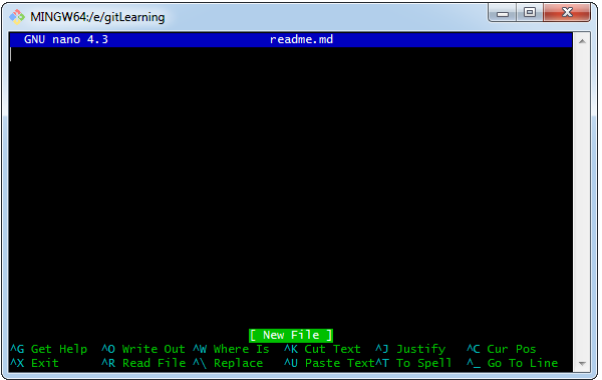
கிட் அடைவில் கோப்பு உருவாக்குவது எப்படி?

- 1) இப்போது உங்கள் அடைவில் ஏதாவது ஒரு கோப்பை(ஃபைல்) உருவாக்க வேண்டும் என்றால் nano filename.md கொடுங்கள். அதாவது,

nano readme.md எனக் கொடுத்தால் readme என்னும் பெயரில் கோப்பு உருவாகும். (nano - என்பது ஒருவகை text editor. இதே போல் vim அல்லது notepad++ ஆகியவற்றையும் பயன்படுத்தலாம்).

(md என்பது mark down என்பதன் சுருக்கம். இது ஒரு கோப்பு வகை (text file format).

- 2) இப்போது கீழே உள்ளது போல, திரை காண்பிக்கப்படும்.



The screenshot shows a terminal window with the title 'MINGW64:/e/gitLearning'. Inside the terminal, the GNU nano 4.3 editor is running. The editor's interface includes a top bar with 'GNU nano 4.3' and 'readme.md'. The main area is a black terminal window. At the bottom, a status bar lists various keyboard shortcuts. A green box highlights the 'New File' option in the status bar.

- 3) அந்தக் கோப்பில் என்ன வேண்டுமானாலும் தட்டச்சிடுங்கள்.
- 4) தட்டச்சிட்ட பிறகு, கோப்பில் இருந்து வெளியேற, ctrl+x ஐ அழுத்துங்கள்.
- 5) இப்போது சேமிக்கவா? எனத் திரையில் கேட்கப்படும். அதற்கு Y ஐ அழுத்துங்கள்.

6) பிறகு File to write readme.md எனக்
காண்பிக்கப்பட்டால் என்டர்
பொத்தானை அழுத்துங்கள்.

கிட் அடைவில் உள்ள கோப்பை கிட் மூலம் கண்காணிப்பது எப்படி?

- 1) இப்போது நீங்கள் – உங்கள் கணினியில்
சேமித்த readme.md ஐ கிட்ஹப்
கண்காணிக்கிறதா என்று பார்ப்பதற்கு,
git status எனத் தட்டச்சிடுங்கள்.

```
MUTHU@MUTHU-PC MINGW64 /e/gitLearning (master)
$ git status
On branch master

No commits yet

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        readme.md

nothing added to commit but untracked files present (use "git add" to track)
```

- 2) இப்போது கிட் பேஷ் திரையில் மேல்
உள்ளது போல்,

On branch master

No commits yet

Untracked files:

(use "git add <file>..." to include in what will be committed)

readme.md

nothing added to commit but untracked files present (use "git add" to track)

என்று தோன்றும்.

3) அதாவது 'readme.md' கோப்பு இன்னும் 'கிட்'டால் கண்காணிக்கப்படவில்லை.

4) இந்தக் கோப்பு, 'கிட்'டால் கவனிக்கப்படவேண்டும் எனில், அதை 'கிட்'உடன் சேர்க்க வேண்டும். அதற்கான கிட் கட்டளை

git add readme.md என்பதாகும்.

- 5) இப்போது திரையில் நீங்கள் இப்படிச்
கீழ் உள்ளதைப் போல, பார்க்கலாம்.

```
MUTHU@MUTHU-PC MINGW64 /e/gitLearning (master)
$ git add readme.md
warning: LF will be replaced by CRLF in readme.md.
The file will have its original line endings in your working directory
```

- 6) இப்போது திரும்பவும் git status
கொடுத்துப் பாருங்கள். கீழே உள்ள
திரை தோன்றும்.

```
MUTHU@MUTHU-PC MINGW64 /e/gitLearning (master)
$ git status
On branch master

No commits yet

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)
    new file:   readme.md
```

- 7) மேல் உள்ள திரையில் Changes to be
committed: என்று காண்பிக்கப்படுகிறது
அல்லவா? அதாவது, 'கிட்' –
readme.md என்னும் கோப்பைக் (git
add கட்டளை மூலம்) கவனிக்கத்
தொடங்கிவிட்டது. ஆனால் இன்னும்

அதன் உள்ளடக்கத்தைப்
உறுதிப்படுத்த(commit)வில்லை.

இந்த இடத்தில் git add கட்டளைக்கும் git commit கட்டளைக்கும் உள்ள வேறுபாட்டைப் புரிந்து கொள்ள வேண்டும்.

git add - git commit என்ன தான் வித்தியாசம்?

git add என்பது 'கிட்'டில் சேர்க்க வேண்டிய கோப்புப் பட்டியலில் (index) ஒரு கோப்பைச் சேர்த்துக் கொள்வது. "இந்தக் கோப்பு - பின்னர் 'கிட்'டில் சேர்க்கப்பட வேண்டிய கோப்பு. எனவே, இந்தக் கோப்பைக் கவனித்து வைத்துக் கொள்' என 'கிட்'ட்டம் சொல்வது தான் git add.

அதே நேரம், git commit என்பது சேர்க்க வேண்டிய கோப்புப் பட்டியலில் (அதாவது add ஆன) உள்ள கோப்புகளை - கோப்பு, இந்தக் கோப்பு எத்தனையாவது பதிப்பு (வெர்ஷன்), கோப்புப் பற்றிய குறிப்பு(log) ஆகிய தகவல்களுடன் உங்கள் கணினியிலேயே (நீங்கள் சொன்ன இடத்திலேயே) சேமிப்பது

ஆகும். இப்படியாக, கோப்பையும் அதைப் பற்றிய பிற விவரங்களையும் உறுதிப்படுத்துவதைத் தான் 'கம்மிட்' (commit) என்று குறிப்பிடுகிறார்கள்.

கிட் கம்மிட்(git commit) எப்படிச் செய்வது?

நாம் 'கிட்' டில் சேர்த்த(add) கோப்பை உறுதிப்படுத்த கிட் கம்மிட் கட்டளையைக் கீழ் உள்ளவாறு பயன்படுத்த வேண்டும்.
git commit -m "message" என்று கொடுக்க வேண்டும். இதில் "message" என்பதில் நம் பயனுக்கு ஏற்றவாறு கொடுக்க வேண்டும்.
git add filename என்று கொடுத்து, அதன் பின்னர் git commit -m "message" என்று உறுதிப்படுத்துவதை ஒரே வரியில்

git commit -am "message" என்னும் கட்டளை மூலமாகவும் செய்யலாம்.

பதிப்பு(version) என்பதன் பயன் என்ன?

முதலில் பதிப்பு என்றால் என்ன புரிந்து கொள்வோம். பலர் சேர்ந்து வேலை செய்யும் இடத்தில், பலரும் ஒரே நேரத்தில் ஒரே கோப்பை மாற்றி(edit)யமைக்கும் சூழல் வரலாம். அப்போது யார் யார் என்னென்ன

வரிகளை மாற்றியிருக்கிறார்கள்,
நீக்கியிருக்கிறார்கள், சேர்த்திருக்கிறார்கள்
என்பதையெல்லாம் எப்படிப் பார்ப்பது?
அதற்குத் தான் ஒவ்வொரு மாற்றத்தின் போதும்
கோப்பைப் பதிப்புகளாக(version)ச் சேமித்து
வைப்பார்கள். இதை 'கிட்' டில் எப்படிச்
செய்வது? என்ன கட்டளை உதவும்?

Commit கட்டளை தான் - ஒரே கோப்பின் பல
பதிப்புகளை எடுத்து வைக்கிறது. பதிப்பு
என்பது - நீங்கள் ஒவ்வொரு முறை ஒரு
நிரலை(program) உறுதிப்படுத்தும்(commit)
செய்யும் போதும் புதிய நிரலின் ஒரு
படி(நகல்) எடுத்து வைப்பது. இப்படிச்
செய்வதன் மூலம், சில நாட்களுக்குப் பிறகு
பழைய பதிப்பு தேவைப்பட்டால் நீங்கள்
எடுத்துக் கொள்ளலாம். பதிப்பு(வெர்ஷன்)
மூலமாக - உங்கள் பழைய நிரல்(program),
மாற்றப்பட்ட புதிய நிரல்(program) என
எல்லாவற்றையும் நீங்கள் எடுத்துக் கொள்ள
முடியும்.

கோப்புப் பற்றிய குறிப்பு(log) என்பது என்ன?

நாம் ஒரு நிரலையோ(program)
கோப்பையோ(file) எத்தனை தடவை
உறுதிப்படுத்தி(commit)யிருக்கிறோம் என்று
பார்ப்பதற்கு குறிப்புகள்(logs)
பயன்படுகின்றன.

git log என்று கொடுப்பதன் மூலம் அந்தக்
குறிப்புகளை நாம் பார்க்க முடியும்.

```
MUTHU@MUTHU-PC MINGW64 /e/gitLearning (master)
$ git log
commit 7cefafa44e6e488787c169b8f656c67d2895ceb5 (HEAD -> master)
Author: Muthuramalingam <muthu1809@gmail.com>
Date: Thu Sep 26 22:04:50 2019 +0530
```

```
This is my first commit, initial version of this file
```

\$ git log

commit

7cefafa44e6e488787c169b8f656c67d2895ceb
5 (HEAD -> master)

Author: Muthuramalingam
<muthu1809@gmail.com>

Date: Thu Sep 26 22:04:50 2019 +0530

This is my first commit, initial version of
this file

commit
7cefafa44e6e488787c169b8f656c67d2895c
eb5

இங்கு என்பதில் கம்மிட் என்பதை அடுத்து
வரும் வாசிக்கமுடியாத எழுத்துக் கோவையை
கம்மிட் ஹேஷ் (commit hash) என்று
சொல்வார்கள். ஹேஷ் என்பது ஒரு
தனித்துவம் வாய்ந்த மதிப்பு. இந்த மதிப்பைக்
கொண்டே, பல முறை
உறுதிப்படுத்தலை(கம்மிட்)ச் செய்யும்
இடங்களில் தேவையான பதிப்பைப்
பெறுவார்கள்.

இந்த ஹேஷ் மதிப்பு மட்டும் தெரிந்தால்
போதும் என்று நினைத்தால்
`git log --oneline` என்னும் கட்டளையைக்
கொடுக்கலாம்.

git commit எப்படிக் கொடுப்பது?

கிட் கம்மிட் கட்டளையைக் கொடுத்து
உறுதிப்படுத்தும் போது, பொதுவாக, அந்த
உறுதிப்படுத்தலை(கம்மிட்)ப் பற்றிய
விவரங்களைச் செய்தியாக இணைத்து
உறுதிப்படுத்துவார்கள்.

1) ஒரு கோப்பை

உறுதிப்படுத்தும்(commit) போது -
என்னென்ன மாற்றங்கள் அந்தக்
கோப்பில் செய்யப்பட்ட பிறகு அந்தக்
கோப்பு உறுதிப்படுத்தப்படுகிறது என்று
சொல்வது பொது மரபு.

2) இப்போது நம்முடைய readme.md கோப்பை உறுதிப்படுத்த(commit),

git commit -m "This is my first commit, initial version of this file"
(இங்கு m என்பது message) என்று
கொடுக்கலாம்.

```
MUTHU@MUTHU-PC MINGW64 /e/gitLearning (master)
$ git commit -m "This is my first commit, initial version of this file"
[master (root-commit) 7cefafa] This is my first commit, initial version of this
file
1 file changed, 7 insertions(+)
create mode 100644 readme.md
```

\$ git commit -m "This is my first commit, initial version of this file"
[master (root-commit) 7cefafa] This is my first commit, initial version of this file
1 file changed, 7 insertions(+)
create mode 100644 readme.md

எனத் திரையில் தோன்றும்.

3) இப்போது திரும்பவும் git status
கொடுத்தால் என்ன வருகிறது என்று
பார்ப்போம்.

```
MUTHU@MUTHU-PC MINGW64 /e/gitLearning (master)
$ git status
On branch master
nothing to commit, working tree clean
```

\$ git status எனத் தட்டச்சிட்டு என்டர்
பொத்தானை அழுத்தினால்

On branch master

nothing to commit, working tree clean

என்று வரும்.

- 4) இப்போது நாம் உருவாக்கியிருக்கும்
readme.md என்பது அந்தக் கோப்பின்
முதல் பதிப்பு ஆகும்.
- 5) இப்போது நாம் அந்தக் கோப்பில் சில
மாற்றங்களைச் செய்ய விரும்புவதாக
வைத்துக் கொள்வோம். அதற்கு என்ன
செய்வது?

ஏற்கெனவே commit ஆன கோப்பில் மாற்றங்கள் செய்வது எப்படி?

- 1) முதலில் அந்தக் கோப்பைத் திறக்க வேண்டும்.
- 2) nano readme.md எனக் கட்டளை இட்டால் கோப்பு திறக்கப்படும்.
(இங்கு readme.md என்பது கோப்பின் பெயர்)
- 3) இப்போது நாம் “தமிழ் I am updating this file தமிழ்” என்று கோப்பில் தட்டச்சிட்டு விட்டு, ctrl+X கொடுப்போம்.

- 4) இந்தக் கோப்பைச் சேமிக்கவா? என்று கிட் பேஷ் கேட்கும். அதற்கு ஆமாம் (Yes) என்று சொல்லி விடுவோம்.
- 5) இப்போது கோப்பு சில மாற்றங்களுடன் சேமிக்கப்பட்டிருக்கிறது.
- 6) இப்போது மீண்டும் git status கொடுத்து கிட் – இந்தக் கோப்பை எப்படிப் பார்க்கிறது என்று பார்ப்போமா?

```
MUTHU@MUTHU-PC MINGW64 /e/gitLearning (master)
$ git status
On branch master
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")
```

\$ git status என்று கொடுத்தால்

On branch master

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")

என்று திரையில் தோன்றும். அதாவது,
“Changes not staged for commit:” மாற்றங்கள்
இன்னும் உறுதிப்படுத்த(commit)ப்படவில்லை.
அதே நேரம்

modified: readme.md என்றும் சொல்கிறது.
அதாவது, கோப்பு மாற்றப்பட்டிருக்கிறது.

கோப்பில் என்னென்ன மாற்றப்பட்டிருக்கின்றன என்று எப்படிப் பார்ப்பது?

- 1) கோப்பின் முந்தைய பதிப்பிற்கும் தற்போதைய நிலைக்கும் என்னென்ன வித்தியாசங்கள் இருக்கின்றன என்று பார்ப்பது மிகவும் எளிதானது.
- 2) `git diff` என்றொரு கட்டளை இருக்கிறது.
- 3) `git diff filename` என்று கொடுத்தால் போதுமானது. (filename – நம்முடைய கோப்பின் பெயர்).

```
$ git diff readme.md
warning: LF will be replaced by CRLF in readme.md.
The file will have its original line endings in your working
diff --git a/readme.md b/readme.md
index 96f1835..94e8ae4 100644
--- a/readme.md
+++ b/readme.md
@@ -5,3 +5,6 @@
Version control isn't just for code. It's for anything you
same time, see differences between those drafts, and even
tral git repositories.

In this tutorial you'll use Git to manage a small Markdown
us version. When you're done, you'll have a workflow you can
+
+
+தமிழ் I am updating this file தமிழ்
```

\$ git diff readme.md

warning: LF will be replaced by CRLF in
readme.md.

The file will have its original line endings in
your working directory

diff --git a/readme.md b/readme.md

index 96f1835..94e8ae4 100644

--- a/readme.md

+++ b/readme.md

@@ -5,3 +5,6 @@

Version control isn't just for code. It's for anything you want to track, including content. Using Git to manage your next writing project gives you the ability to view multiple drafts at the same time, see differences between those drafts, and even roll back to a previous version. And if you're comfortable doing so, you can then share your work with others on GitHub or other central git repositories.

In this tutorial you'll use Git to manage a small Markdown document. You'll store an initial version, commit it, make changes, view the difference between those changes, and review the previous version. When you're

done, you'll have a workflow you can apply to your own writing projects.

+

+

+தமிழ் I am updating this file தமிழ்

மேல் உள்ள கோப்பில் பச்சை நிறத்தில் உள்ள ('+' (கூட்டல்) குறியுடன் தொடங்கும்) வரிகள் முந்தைய பதிப்பில் இல்லாமல் தற்போது சேர்க்கப்பட்டவை. இதே போல், ஏதாவது வரி நீக்கப்பட்டிருந்தால் அவை இளஞ்சிவப்பு நிறத்தில் (- (கழித்தல்) குறியுடன்) காட்டப்படும்.

எல்லாம் சரி, தப்பாக உறுதிப்படுத்தலை(கம்மிட்) செய்து விட்டால் மாற்ற முடியுமா?

சில நேரங்களில், உறுதிப்படுத்தலை(commit)ச் செய்த பின்னர், மாற்றங்கள் தேவைப்படலாம். அதற்குத் தனியாக இன்னொரு முறை

உறுதிப்படுத்தலைச் செய்ய வேண்டிய தேவையில்லை. பழைய உறுதிப்படுத்தலையே மேம்படுத்தி மாற்ற முடியும். எப்படிச் செய்வது?

1) `git commit --amend -m "message"`

என்று கொடுக்க வேண்டும்.

2) இங்கு "message" என்பது

உறுதிப்படுத்தும் போது நாம் கொடுக்க விரும்பும் செய்தியாகும்.

```
$ git commit --amend -m "amended message"
[master 421c302] amended message
Date: Thu Sep 26 22:04:50 2019 +0530
1 file changed, 7 insertions(+)
create mode 100644 readme.md
```

இரண்டாவது முறை உறுதிப்படுத்துவது(கம்மிட்) இப்போது நாம் மீண்டும் `git add filename` கொடுத்து, பின்னர் உறுதிப்படுத்தல்(கம்மிட்) கட்டளையையும் கொடுத்துக் கொள்ளலாம்.

```
$ git add readme.md
warning: LF will be replaced by CRLF in readme.md.
The file will have its original line endings in your working directory
MUTHU@MUTHU-PC MINGW64 /e/gitLearning (master)
$ git commit -m "Second commit"
[master e45b396] Second commit
1 file changed, 3 insertions(+)
```

இப்போது தான் கோப்பின் குறிப்புகளை(logs)ப் பயன்படுத்தப்

போகிறோம். எப்படி? இப்போது ஒரே கோப்பை இரண்டு முறை மாற்றி நாம் உறுதிப்படுத்தி(கம்மிட்)யிருக்கிறோம்.

இந்த இரண்டு

உறுதிப்படுத்தல்(கம்மிட்)களுக்கும் இரண்டு தனித்துவமான ஹேஷ் மதிப்புகள்

உருவாகியிருக்கும். அவற்றை

`git log --oneline` கட்டளை மூலம் பார்க்க முடியும்.

```
$ git log --oneline
e45b396 (HEAD -> master) Second commit
421c302 amended message
```

இங்கு e45b396 என்பது இரண்டாவது உறுதிப்படுத்தலின்(கம்மிட்) ஹேஷ் மதிப்பு.

421c302 என்பது முதல்

உறுதிப்படுத்தலின்(கம்மிட்) ஹேஷ் மதிப்பு.

முதல் உறுதிப்படுத்தலை நாம்

மாற்றி(amend)யிருந்தது உங்களுக்கு

நினைவிருக்கும்.

விரும்பும் பதிப்பை(வெர்ஷனை)ப் பார்ப்பது எப்படி?

இப்போது நம்மிடம் ஒரே கோப்பின்(ஃபைல்)
இரண்டு பதிப்பு(வெர்ஷன்)கள் உள்ளன.
இந்த இரண்டு பதிப்புகளில் நாம் விரும்பும்
பதிப்பைப் பார்ப்பது எப்படி என்று
பார்ப்போம்.

அதற்கான கிட் கட்டளை `git show`
ஹேஷ்மதிப்பு கோப்பின் பெயர் ஆகும்.
அதாவது,

இப்போது நாம்,
`git show 421c302 readme.md` என்று
கொடுத்தால் `readme.md` கோப்பின்(ஃபைல்)
முதல் பதிப்பை(வெர்ஷன்)ப் பார்த்துவிட
முடியும். இரண்டாவது பதிப்பைப் பார்க்க
வேண்டும் என்றால் அதன் ஹேஷ் மதிப்பைக்
கொடுத்தால் போதுமானது.

```

$ git show 421c302 readme.md
commit 421c3027a900c1256defa7d1433fef6c1485e29b
Author: Muthuramalingam <muthu1809@gmail.com>
Date: Thu Sep 26 22:04:50 2019 +0530

    amended message

diff --git a/readme.md b/readme.md
new file mode 100644
index 0000000..96f1835
--- /dev/null
+++ b/readme.md
@@ -0,0 +1,7 @@
+
+
+### Introduction
+
+Version control isn't just for code. It's for anything you want to track
+same time, see differences between those drafts, and even roll back to
+tral git repositories.
+
+In this tutorial you'll use Git to manage a small Markdown document. You
+us version. When you're done, you'll have a workflow you can apply to you

```

இதைக் கோப்பின்(ஃபைல்)பார்க்க, `git show 421c302:./readme.md` என்று `./` ஐ மேல் உள்ள கட்டளையில் சேர்த்தால் போதும். கீழே உள்ளது போல, கோப்பு(ஃபைல்) வடிவத்திலேயே தெரியும்).

```

$ git show 421c302:./readme.md

### Introduction

Version control isn't just for code. It's for anything you want to track
same time, see differences between those drafts, and even roll back to
ral git repositories.

In this tutorial you'll use Git to manage a small Markdown document. You
s version. When you're done, you'll have a workflow you can apply to you

```

ஒரு கோப்(ஃபைல்)பின் பதிப்பை(வெர்ஷன்) இன்னொரு கோப்பாக(ஃபைல்) மாற்ற முடியுமா?

முடியும். இப்போது கோப்பின் இந்தப் பதிப்பை, ஒரு புதிய கோப்பாகச் சேமிக்க -
git show 421c302:./readme.md >
old_article.md என்று கொடுக்கலாம்.

இங்கு readme.md கோப்பின் பதிப்புகளில் ஒரு பதிப்பு(வெர்ஷன்) 421c302 ஐ ஹேஷ் மதிப்பாகக் கொண்டிருக்கிறது. அந்தப் பதிப்பை எனக்கு old_article.md என்னும் பெயருக்கு மாற்றி இன்னொரு புதிய கோப்பாகவே சேமித்து விடு என்று 'கிட்'டிற்கு நாம் கட்டளையிடுகிறோம் என்பது மேல் உள்ள கட்டளையின் அர்த்தம் ஆகும்.
இதுவரை நாம் பார்த்திருப்பது -

1) நம்முடைய கணினியில் கிட்டை நிறுவுவது

2) அதில் நமக்குத் தேவையான அடைவு(ஃபோல்டருக்கு)க்குப் போவது

3) அதில் ஒரு கோப்பை(ஃபைலை) உருவாக்குவது

4) அந்தக் கோப்புக்குப் பல
பதிப்பு(வெர்ஷன்)கள் உருவாக்குவது

5) அவற்றைப் பார்ப்பது

ஆகியவற்றைப் படித்திருக்கிறோம். இவை
தான் 'கிட்' டின் பயன்களா? என்று கேட்டால்
இல்லை,

1) இன்னும் வேறொருவர் கணினியில்
இருந்து திட்டப்பணி(பிராஜெக்ட்)களை
எவ்வாறு நம்முடைய கணினிக்கு 'கிட்'
மூலம் கொண்டு வருவது

2) அந்தத் திட்டப்பணியில் உள்ள
கோப்பு(ஃபைல்)களில் மாற்றங்கள்
எப்படிச் செய்வது?

3) நம்முடைய கணினியில் செய்யப்பட்ட
இந்த மாற்றங்களை எப்படி
மாற்றவர்களுக்குத் தெரியப்படுத்துவது
என்று பல்வேறு கேள்விகள்

இருக்கின்றன. அவை அனைத்தையும்
ஒவ்வொன்றாகப் பார்ப்போம்.

‘கிட்ஹப்’பில் இருந்து திட்டப்பணி(பிராஜெக்ட்)கள் எடுப்பது எப்படி?

கிட்ஹப் என்பது பல
திட்டப்பணி(பிராஜெக்ட்)கள் தொகுத்து
வைக்கப்பட்டிருக்கும் இடம் ஆகும். இந்த
இடத்தில் - ஒவ்வொரு
திட்டப்பணி(பிராஜெக்ட்)க்கும் அதன் பல
பதிப்பு(வெர்ஷன்)களுடன் இடம்
ஒதுக்கப்பட்டிருக்கும். எனவே, ஒரு
திட்டப்பணி(பிராஜெக்ட்)யின் எந்தப்
பதிப்பு(வெர்ஷன்) நமக்கு வேண்டுமோ அதை
நாம் கிட்ஹப்பில் இருந்து எடுத்துக்
கொள்ளலாம். கிட்ஹப் திறந்த மூல(ஓப்பன்
சோர்ஸ்)த் தளமாகும்; இலவசத் தளமும் கூட.
யார் வேண்டுமானாலும் கிட்ஹப்பில் இருந்து
திட்டப்பணிகளை எடுத்து வேலை செய்யலாம்.
எனவே

- 1) திட்டப்பணி(பிராஜெக்ட்)களின் பல்வேறு பதிப்பு(வெர்ஷன்)கள் கொண்ட (Version Control)
- 2) பலருக்கும் பகிர்ந்தளிக்கப்படும் (Distributed)
- 3) திறந்த மூல , இலவச இணையத்தளம் (Open Source and Free)

என 'கிட்ஹப்' பை வரையறுக்கிறார்கள் .

இப்படிப்பட்ட கிட்ஹப்பில் இருந்து நம்முடைய கணினிக்கு எப்படித் திட்டப்பணி(பிராஜெக்ட்)களைத் தரவிறக்குவது(டவுன்லோடு)? அதில் எப்படி வேலை செய்வது? பிறகு மீண்டும் எப்படி கிட்ஹப்பிற்கு ஏற்றுவது? என்பன போன்ற கேள்விகளுக்குத் தான் பதில்கள் பார்க்கப் போகிறோம் .

முன்னேற்பாடுகள்

- 1) இதைப் படிக்க நம்முடைய கணினியில் ஏற்கெனவே கிட்டை நிறுவியிருக்க(இன்ஸ்டால்) வேண்டும். (இதை நாம் ஏற்கெனவே முடித்து விட்டோம்.)
- 2) கிட்ஹப் தளத்தில் நமக்கென்று ஒரு புகுபதிகை(லாக் இன்) இருக்க வேண்டும். (இதையும் நாம் முடித்து விட்டோம்.)
- 3) கடைசியாக, எந்தத் திட்டப்பணி(பிராஜெக்ட்)க்கு வேலை செய்யப் போகிறோம் என்பது தெரிய வேண்டும். ஏனென்றால் கிட்ஹப்பில் ஆயிரக்கணக்கான திட்டப்பணி(பிராஜெக்ட்)கள்

இருக்கின்றன. அவற்றுள் நாம் எதை
எடுத்து வேலை செய்யப் போகிறோம்
என்பது தெரிய வேண்டும். அந்தத்
திட்டப்பணி(பிராஜெக்ட்)யின்
இணையப்பக்க முகவரி(URL) தெரிய
வேண்டும்.

இவை மூன்றும் தெரிந்தால் தான் நாம்
திட்டப்பணி(பிராஜெக்ட்)யில் ஈடுபட முடியும்.

களஞ்சியம் என்றால் என்ன என்று தெரியுமா?

களஞ்சியம் என்ற வார்த்தைக்குக் குவித்து வைக்கும் இடம் என்று அர்த்தம். தமிழ்நாட்டில் தஞ்சாவூரைத் 'தமிழகத்தின் நெற்களஞ்சியம்' என்று சொல்வார்கள். அதாவது, நெல் அதிகமாக விளைந்து குவிந்து கிடக்கும் இடம்.

இப்போது உங்களுக்குக் களஞ்சியம் என்றால் என்ன என்று தெரிந்திருக்கும் என்று நம்பலாம் அல்லவா? இதே வார்த்தையைத் தான் திட்டப்பணி(பிராஜெக்ட்)யின் முக்கியமான கோப்புகள்(ஃபைல்), நிரல்(புரோகிராம்)கள் அடங்கிய இடத்திற்கு வைத்திருக்கிறார்கள். இந்தக் களஞ்சியத்தை ஆங்கிலத்தில் ரெப்பாசிட்டரி(சுருக்கமாக ரெப்போ) என்று சொல்வார்கள். அதாவது, ரெப்போ என்பது திட்டப்பணி(பிராஜெக்ட்)யின் எல்லா

நிரல்(புரோகிராம்)களும் அடங்கிய ஓர்
அடைவு(ஃபோல்டர்)!

முள்கரண்டி(Fork) தெரியுமா?



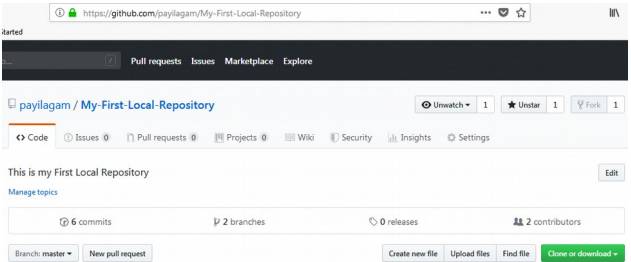
முதலில் நெல், இப்போது கரண்டியா
என்கிறீர்களா? சரிதான் உங்கள் கேள்வி!
களஞ்சியத்தில்(ரெப்போ) இருந்து அவ்வளவு
நிரல்களையும் எடுக்கக் கை போதாது
அல்லவா? அதற்குத் தான் கரண்டியைப்
பயன்படுத்துகிறோம். நம்மூர் ஆள் கிட்ஹப்
தளத்தைக் கொண்டு வந்திருந்தால் -
அகப்பையைப் பயன்படுத்தியிருப்பார்.
வெளிநாட்டு ஆள் கொண்டு வந்ததால்
முள்கரண்டி(Fork) வந்து விட்டது.

முள்கரண்டியால் எப்படித் களஞ்சியத்தில் இருந்து எடுப்பது?

- 1) முதலில் நாம் விரும்பும்
களஞ்சியத்திற்கு(ரெப்போ)ப் போக
வேண்டும். ஓர் எடுத்துக்காட்டிற்கு, நாம்
[https://github.com/payilagam/
My-First-Local-Repository](https://github.com/payilagam/My-First-Local-Repository)
ரெப்போவிற்குப் போவதாக வைத்துக்
கொள்வோம்.

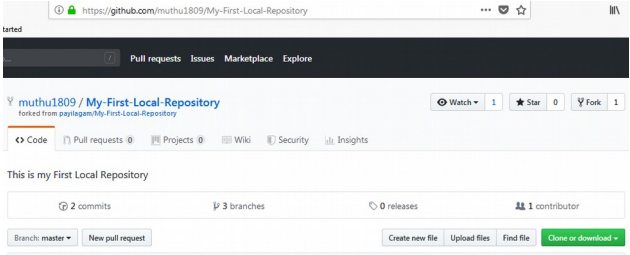
(இங்கு payilagam என்பது கிட்டஹப்பில் உள்ள
பயனர் பெயர், My-First-Local-
Repository என்பது அந்தப் பயனர்
வைத்திருக்கும் திட்டப்பணி(பிராஜெக்ட்)யின்
பெயர்)

2) அங்கே போய், அங்கிருக்கும் நிரல்களை முள்கரண்டியால்(ஃபோர்க்) நம்முடைய கணினிக்குக் கொண்டு வர வேண்டும் என்றால் – அந்தப் பக்கத்தில் இருக்கும் ஃபோர்க்(Fork) என்னும் பொத்தானை அழுத்த வேண்டும்.



3) இப்போது முள்கரண்டியால் மேல் உள்ள திட்டப்பணி(பிராஜெக்ட்)யை நாம் எடுத்த பிறகு, இப்போது இணையப்பக்கத்தின் முகவரி(URL)யைக் கவனித்துப் பாருங்கள்.

- 4) அது நம்முடைய கிட் பக்க முகவரியாக மாறியிருக்கும். இங்கு <https://github.com/muthu1809/My-First-Local-Repository> என்று மாறியிருக்கிறது.



- 5) இதன் அர்த்தம் என்னவென்றால் My-First-Local-Repository என்னும் களஞ்சியத்தின்(ரெப்போ) ஒரு நகலை நாம் நம்முடைய கிட் பக்கத்திற்கு எடுத்துக்(ஃபோர்ட்) கொண்டோம்.
- 6) இப்போது நம்முடைய கிட்ஹப் இணையத்தளத்தில் இந்தத்

திட்டப்பணி(பிராஜெக்ட்) வந்து
விட்டது.

7) அடுத்த வேலை, இந்தத்
திட்டப்பணி(பிராஜெக்ட்)யை
நம்முடைய கணினி நினைவகத்திற்குக்
கொண்டு வருவது தான்.

8) இதை இரண்டு வழிகளில் செய்யலாம்.

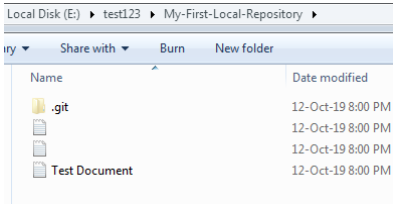
ஃபோர்க் செய்யப்பட்ட திட்டப்பணி(பிராஜெக்ட்)யை குளோன்(clone) செய்வது எப்படி?

கிட் பேஷில் git clone
<https://github.com/muthu1809/My-First-Local-Repository> என்று கட்டளை
கொடுப்பது ஒரு வகை.

clone(குளோன்) என்னும் ஆங்கில வார்த்தை
நாம் கேள்விப்பட்டது தான்! இருப்பதைப்
போலவே படி(நகல்) எடுப்பது என்பது அதன்
அர்த்தம்! இப்போது நாம் கிட்டஹப்பில் உள்ள
களஞ்சியத்தில்(ரெப்போவில்) இருந்து
நம்முடைய கணினிக்குத்
திட்டப்பணி(பிராஜெக்ட்)யைப் படி(நகல்)
எடுக்கிறோம். எனவே தான் அந்தக்
கட்டளையின் பெயர் குளோன்.

```
MUTHU@MUTHU-PC MINGW64 /e/test123
$ git clone https://github.com/muthu1809/My-First-Local-Repository
Cloning into 'My-First-Local-Repository'...
remote: Enumerating objects: 13, done.
remote: Total 13 (delta 0), reused 0 (delta 0), pack-reused 13
Unpacking objects: 100% (13/13), done.
```

- 1) இப்போது நாம், நம்முடைய கிட் அடைவை(ஃபோல்டர்)ப் போய்ப் பார்த்தால்



My-First-Local-Repository என்னும் அடைவு(ஃபோல்டர்) இருக்கும்.

- 2) இன்னொரு வகை, கீழ் உள்ள படத்தில் பச்சை நிறத்தில் உள்ள 'Clone or download' பொத்தானைச் சொடுக்குவது.

Code

Pull requests 0

Projects 0

Wiki

Security

Insights

This is my First Local Repository

2 commits

3 branches

0 releases

1 contributor

Branch: master

New pull request

Create new file

Upload files

Find file

Clone or download

புதிய கிளை ஆரம்பம்

நாம் இணையத்தில் இருந்து எடுத்திருக்கும் திட்டப்பணி(பிராஜெக்ட்) நாம் ஒருவர் செய்யும் திட்டப்பணி(பிராஜெக்ட்) இல்லை. பலரும் சேர்ந்து வேலை செய்யப்போகும் பெரிய திட்டப்பணி(பிராஜெக்ட்). இந்தத் திட்டப்பணி(பிராஜெக்ட்)யின் ஒரு படி(நகலை)யை நாம் நம்முடைய கணினிக்குக் கொண்டு வந்திருக்கிறோம். இப்போது இந்த மூலப்படி(master copy)யை அப்படியே ஒரு மாதிரி (reference) போல வைத்துக் கொள்வது நல்லது. ஏனென்றால் - நிரல் எழுதும் நண்பர்களுடன் உரையாடும் போது மாற்றங்கள் எதுவும் செய்யப்படாத மூலத்தை(master copy) வைத்து உரையாடுவது நல்லதல்லவா? நாம் விரும்பும் மாற்றங்களை இன்னொரு கிளை உருவாக்கி அதில் செய்துகொள்ளலாம்.

புதிய கிளை(பிராஞ்ச்)யை உருவாக்குவது எப்படி?

- 1) தரவிறக்கப்பட்ட (டவுன்லோடு)
களஞ்சியத்தை(ரெப்போ) அப்படியே
வைத்துக் கொள்வதை மாஸ்டர்
கிளை(பிராஞ்ச்) என்று சொல்வார்கள்.
- 2) இப்போது அந்த மாஸ்டர் கிளையில்
இருந்து நாம் இன்னொரு கிளையை
உருவாக்கிக் கொண்டால்,
 - நம்முடைய நிரல்
தொடர்பான அனைத்து
மாற்றங்களையும் அந்தக்
கிளையில் செய்து
கொள்ளலாம்.
 - நாம் செய்யும் எந்த
மாற்றமும் மாஸ்டர்
கிளையில் எந்த

மாற்றத்தையும்
ஏற்படுத்தாது.

- 3) புதிய கிளையை உருவாக்க, git branch
branch_name கட்டளையைப்
பயன்படுத்தலாம். (முதலில் அந்த
அடைவு(ஃபோல்டர்)க்கு cd கொடுத்துப்
போய் விடுங்கள்.)

```
MUTHU@MUTHU-PC MINGW64 /e/test123  
$ cd My-First-Local-Repository  
MUTHU@MUTHU-PC MINGW64 /e/test123/My-First-Local-Repository (master)  
$ git branch kilai
```

\$ git branch kilai

இங்கு kilai என்பது தான் புது
கிளை(பிராஞ்ச்)யின் பெயர்.

புதிய கிளைக்குத் தாவுவது எப்படி?

இப்போது நாம் உருவாக்கியிருக்கும் புதிய கிளை(பிராஞ்ச்)க்கு எப்படி மாறுவது? அதற்கு 'கிட்' டில் `git checkout` என்று ஒரு கட்டளை இருக்கிறது. `git checkout branchname` என்று கொடுத்தால் போதும்.

அதாவது நாம்,
`git checkout kilai` என்று கொடுக்க வேண்டும். (நம்முடைய கிளை(பிராஞ்ச்)யின் பெயர் `kilai`).

```
MUTHU@MUTHU-PC MINGW64 /e/test123/My-First-Local-Repository (master)
$ git checkout kilai
Switched to branch 'kilai'
```

```
MUTHU@MUTHU-PC MINGW64 /e/test123/My-First-Local-Repository (kilai)
```

மேல் உள்ள படத்தில் 'switched to branch 'kilai' என்றும் (kilai) என்று இருப்பதைக் கவனிக்கலாம்.

சுருக்க வழி:

புதிய கிளையை(பிராஞ்ச்) உருவாக்கி அதற்குத் தாவுவதை(செக் அவுட்) ஒரே வரியில்
`git checkout -b kilai` என்றும்
கொடுக்கலாம்.

இப்போது மீண்டும் மாஸ்டர்
கிளை(பிராஞ்ச்)க்குத் தாவ வேண்டும் என்று
ஆசைப்பட்டால், `git checkout master`
என்று கொடுக்க வேண்டும்.

எத்தனை கோப்புகள்(ஃபைல்கள்) உள்ளன
என்று எப்படிப் பார்ப்பது?

ஒரு கிளையில் எத்தனை கோப்பு(ஃபைல்)கள்
இருக்கின்றன என்று பார்ப்பதற்கு `ls` என்னும்
கட்டளையைப் பயன்படுத்தலாம். (`ls` என்பது
`list` என்பதன் சுருக்கம் ஆகும்).

```
MUTHU@MUTHU-PC MINGW64 /e/test123/My-First-Local-Repository1 (kilai)
$ ls
README.md
```

**கோப்பைப் புதிய கிளையில் திறந்து
மாற்றுவது?**

இப்போது README.md கோப்பை kilai கிளையில் திறந்து அதில் 'This is a new line added in branch called kilai' என்னும் வரியைச் சேர்க்கப் போகிறோம்.

- கோப்பைத் திறக்க nano README.md
- திறந்த கோப்பில் தேவையான வரிகளைச் சேர்த்த பிறகு ctrl+x கொடுத்து வெளிவந்து விடலாம்.
- இப்போது git add -A என்று கொடுத்துப் பாருங்கள். இதில் 'A' என்று கொடுப்பது 'All' (எல்லாமே) என்பதன் சுருக்கம்.

```
MUTHU@MUTHU-PC MINGW64 /e/test123/My-First-Local-Repository (kilai)  
$ git add -A
```

அதாவது, பல கோப்புகளை மாற்றியிருக்கும்போது அவையனைத்தையும் கிட்டில் சேர்க்க git add -A கட்டளையைக் கொடுக்கலாம். இப்போது 'கிட்' டில் இந்த மாற்றங்களை உறுதிப்படுத்த, நாம் ஏற்கெனவே படித்த git commit கட்டளையைப் பயன்படுத்த வேண்டும்.

```
MUTHU@MUTHU-PC MINGW64 /e/test123/My-First-Local-Repository (kilai)  
$ git commit -m "Added one line in new branch"  
[kilai 4952923] Added one line in new branch  
1 file changed, 1 insertion(+)  
create mode 100644 Test
```

இப்போது புதிய கிளை(பிராஞ்ச்)யில் உறுதிப்படுத்தல்(கம்மிட்) நடந்து விட்டது என்பதை உறுதிப்படுத்த, மீண்டும் git status கட்டளையைக் கொடுத்துப் பார்க்கலாம்.

```
MUTHU@MUTHU-PC MINGW64 /e/test123/My-First-Local-Repository (kilai)
$ git status
On branch kilai
nothing to commit, working tree clean
```

git push படிக்கும் நேரம் வந்து விட்டது!

இப்போது git push கட்டளையைப் படிக்கத் தொடங்கி விடலாம். இந்தக் கட்டளை எதற்குப் பயன்படும் என்கிறீர்களா? push என்றால் என்ன? தள்ளுவது! எங்கே இருந்து - எதை - எங்கே - தள்ளப் போகிறோம்? எங்கே இருந்து - நம்முடைய கணினியில் நாம் உருவாக்கிய புதிய கிளை(பிராஞ்ச்)யில் இருந்து எதை- நாம் உருவாக்கிய README.md கோப்பை.

எங்கே தள்ளப் போகிறோம் - கிட்ஹப் தளத்தில் உள்ள நம்முடைய மூலமான (origin) களஞ்சியத்திற்கு(ரெப்போ)த் தள்ளப் போகிறோம்.

git remote add கட்டளை என்ன செய்யும்?

இந்த மூன்றையும் செய்வதற்கு முன்னர் ஒரு சின்ன வேலை செய்ய வேண்டியிருக்கிறது.

கிட்டஹப் தளத்தில் நாம் பல களஞ்சியங்கள்(ரெப்போக்கள்) வைத்திருக்கலாம் அல்லவா? அவற்றுள் எந்தக் களஞ்சியம்(ரெப்போ) நமக்கு இப்போது மூல(origin)க் களஞ்சியம் என்று சொல்ல வேண்டியது நம்முடைய கடமை ஆகிறது அல்லவா? அதற்காக, 'கிட் பேஷில்'

```
git remote add moolam
```

<https://github.com/muthu1809/My-First-Local-Repository1>

அதாவது, கிட்டஹப் தளத்தில் (remote) மூலக்(origin) களஞ்சியமாக

<https://github.com/muthu1809/My-First-Local-Repository1> என்பதைச்

சேர்க்கிறோம்(add) என்பது இந்தக் கட்டளைக்கு அர்த்தம்.

இப்படிக் கொடுத்ததற்குப் பிறகு, நம்முடைய கிளையை origin கிளைக்குத் தள்ள

```
git push --set-upstream origin new-branch
```

எனும் கட்டளையைக் கொடுக்க வேண்டும். (set-upstream என்பது new-branch எனும்

நம் கணினியில் உள்ள கிளைக்கு - கிட் ஹப் தளத்தில் ஒரு களஞ்சியத்தை எடுத்து வைக்கிறது. இதன் பிறகு, நாம் தள்ளும் நிரல்கள், கோப்புகள் எல்லாமே, கிட்ஹப் தளத்தில் உள்ள இந்தக் களஞ்சியத்திற்குத் தள்ளப்படும்.

```
MUTHU@MUTHU-PC MINGW64 /e/test123/My-First-Local-Repository (kilai)
$ git push --set-upstream moolam kilai
Enumerating objects: 8, done.
Counting objects: 100% (8/8), done.
Delta compression using up to 4 threads
Compressing objects: 100% (5/5), done.
Writing objects: 100% (7/7), 710 bytes | 710.00 KiB/s, done.
Total 7 (delta 1), reused 0 (delta 0)
remote: Resolving deltas: 100% (1/1), done.
To https://github.com/muthu1809/My-First-Local-Repository
```

சுருக்க வழி:

`git push -set-upstream origin new-branch` என்னும் கட்டளையைச் சுருக்கமாக `git push -u origin new-branch` என்றும் கொடுக்கலாம். அண்மைக் காலங்களில் `-set-upstream` என்பதற்கு மாற்றாக `-set-upstream-to` என்று கொடுக்கும் வழக்கமும் நடைமுறையில் இருக்கிறது.

முக்கிய குறிப்பு:

சில சமயங்களில் நம்முடைய கணினியில் இருந்து ரிமோட் ரெப்போவிற்குத் தள்ளும்(புஷ்) போது சிக்கல்கள் வரலாம். அந்தச் சூழலில் ,
`git pull origin new-branch --allow-unrelated-histories`

```
MUTHU@MUTHU-PC MINGW64 /e/test/cloud_haiku (new-branch)
$ git pull origin new-branch --allow-unrelated-histories
From https://github.com/muthu1809/cloud_haiku
* branch                new-branch -> FETCH_HEAD
Merge made by the 'recursive' strategy.
 cloud_haiku             | 1 +
 old_article.md          | 7 ++++++
 readme.md               | 12 ++++++++
 readme_diff.diff        | 11 ++++++++
4 files changed, 31 insertions(+)
create mode 160000 cloud_haiku
create mode 100644 old_article.md
create mode 100644 readme.md
create mode 100644 readme_diff.diff
```

கட்டளையைக் கொடுத்து நம்முடைய களஞ்சியத்தை(new-branch) நிகர்படுத்தி(update)க் கொள்ள வேண்டும். அதன் பிறகு மீண்டும் தள்ள(push) முயல வேண்டும்.

இந்தக் கட்டளைக்குப் பிறகு நாம் 'கிட்ஹப்' தளத்தில் நம்முடைய களஞ்சியத்திற்கு(ரெப்போ)ப் போய்ப் பார்த்தால் நம்முடைய புதிய கிளை இருப்பதைப் பார்க்கலாம்.

Branch: master ▾ New pull request

Switch branches/tags

Find or create a branch...

Branches

Tags

kilai

✓ master

Initial commit

My-First-Local-Repository1

மேல் உள்ள பக்கத்தில் `kilai` என்பதைத் தேர்ந்தெடுத்துப் பார்க்கலாம். அப்போது இணையப்பக்கத்தின் முகவரியில் அந்தக் கிளையின் பெயரும்

<https://github.com/muthu1809/My-First-Local-Repository1/tree/kilai>

சேர்ந்திருப்பதைப் பார்க்க முடியும்.

கதை எப்படிப் போகிறது?

இன்னும் அடுத்தடுத்த கட்டளைகளைப் படிப்பதற்கு முன்பு ஒரு முறை முன்கதைச் சுருக்கத்தைப் பார்த்து விடுவது நல்லது.

1) கிட்ஹப் தளத்தில் கணக்கு உருவாக்கினோம்.

2) கிட்டை நம்முடைய கணினியில் நிறுவினோம்.

3) கிட்ஹப் தளத்தில் வேறொருவர் பக்கத்தில் இருந்த களஞ்சியத்தை(ரெப்போ) முள்கரண்டி(ஃபோர்க்) மூலம் 'கிட்'டில் நம்முடைய பக்கத்திற்குக் கொண்டு வந்தோம்.

4) அதன் பிறகு கிளை(பிராஞ்ச்)
உருவாக்கினோம்.

5) அந்தக் கிளை(பிராஞ்ச்)யில் மாற்றங்கள்
செய்தோம்.

6) அதை கிட்ஹப் தளத்தில் நம்முடைய
பக்கத்தில் உள்ள
களஞ்சியத்திற்கு(ரெப்போ)
நகர்த்தினோம்.

இவைதாம் இதுவரை நாம் செய்த வேலைகள் .

இப்போது வேறொருவரின்
களஞ்சியத்தை(ரெப்போ) நாம் எடுத்து
கிளை(பிராஞ்ச்) முளைத்து அந்தக்
கிளை(பிராஞ்ச்)யில் மாற்றங்கள்
செய்திருக்கிறோம். நாம் இப்படிப்பட்ட
மாற்றங்களைச் செய்திருக்கும் இதே நேரம் -
நாம் (வேறொருவருடைய) எந்தக்
களஞ்சியத்தை(ரெப்போ) கிட்ஹப்பில் இருந்து
எடுத்தோமோ, அதில் மாற்றங்கள்
செய்யப்பட்டிருக்கலாம் அல்லவா? அந்த

மாற்றங்கள், நம்முடைய கணினிக்கும்
தெரியவேண்டும் என்றால் என்ன செய்வது?
அதாவது இப்போது திரும்பவும்
வேறொருவரின் களஞ்சியத்தை(ரெப்போ)
முள்கரண்டி(ஃபோர்க்) மூலம் எடுக்க
வேண்டும்.

வேறொருவர் களஞ்சியத்தை(ரெப்போ) முள்கரண்டி(ஃபோர்க்) மூலம் எடுக்க

ஏற்கெனவே நாம் நம்முடைய கிட்ஹப்
களஞ்சியத்தை(ரெப்போ)
முள்கரண்டி(ஃபோர்க்) மூலம் எடுத்திருக்கிறோம்
அல்லவா? அதே போலத் தான்!

வேறொருவரின் களஞ்சியத்தில் இருந்து புதிய,
மாற்றங்கள் செய்யப்பட்ட
கோப்பு(ஃபைல்)களை நம்முடைய
களஞ்சியத்திற்குக் கொண்டுவரப் போகிறோம்.
அதாவது, மேல் இருக்கும் அவருடைய
ஓடையில் இருந்து கொஞ்சம் கொஞ்சமாக

நம்முடைய மூல(origin)க் களஞ்சியத்திற்குத் தரவுகளைக் கொண்டு வரப் போகிறோம்.

அவருடைய களஞ்சியம் மேல்

ஓடை(upstream) - நம்முடைய களஞ்சியம் தான் நம்முடைய மூலம் (origin).

இப்போது மேல் ஓடை என்பது அவருடைய களஞ்சியத்தைக் குறிக்கும் என்பதை 'கிட்' டிற்கு எப்படிச் சொல்வது?

```
git remote add melodai
```

<https://github.com/payilagam/My-First-Local-Repository1> என்று

கொடுத்தால் போதும்.

இங்கு

<https://github.com/payilagam/My-First-Local-Repository1> என்பதை

melodai எனப் பெயர் சூட்டி, 'கிட்' டிற்குச் சொல்லி விட்டோம். சரி தானே!

```
MUTHU@MUTHU-PC MINGW64 /e/test123/My-First-Local-Repository1 (kilai)
$ git remote add melodai https://github.com/payilagam/My-First-Local-Repository1
```

இப்போது மீண்டும் `git remote -v` கட்டளை கொடுத்துப் பார்ப்போம்.

```
MUTHU@MUTHU-PC MINGW64 /e/test123/My-First-Local-Repository1 (kilai)
$ git remote -v
melodai https://github.com/payilagam/My-First-Local-Repository1 (fetch)
melodai https://github.com/payilagam/My-First-Local-Repository1 (push)
moolam git@github.com:muthu1809/My-First-Local-Repository1.git (fetch)
moolam git@github.com:muthu1809/My-First-Local-Repository1.git (push)
```

இப்போது மிக எளிதாக, மேல் ஓடை(upstream)யில் இருந்து நம்முடைய மூலத்திற்கு(origin)க் கோப்புகளை எடுத்து விடலாம்! `git fetch upstream` என்று 'கிட்' டிடம் சொன்னால் போதும்.

```
MUTHU@MUTHU-PC MINGW64 /e/test123/My-First-Local-Repository1 (kilai)
$ git fetch melodai
From https://github.com/payilagam/My-First-Local-Repository1
* [new branch]      master      -> melodai/master
```

மேலே கடைசி வரியைப் பாருங்கள்.

[new branch] master -> melodai/master

payilagam பயனரின் My-First-Local-Repository1 களஞ்சியத்தின்(ரெப்போ) மாஸ்டர் கிளையில் இருந்து நம்முடைய கணினியின் melodai/master களஞ்சியத்திற்குச் சேமிக்கப் படும்.

இப்போது நம் கணினியின் மாஸ்டர் கிளைக்குப் போவோமா? (இதுவரை நாம் kilai கிளையில் இருப்பதை நினைவில் வைத்திருக்கிறீர்கள் தானே!)

இப்போது

git checkout master

என்று கட்டளையிட்டால் போதும்.

(ஒருவேளை இதில் சிக்கல் வந்தால் git checkout -f master கட்டளை கொடுத்துப் பாருங்கள். 'f' என்பது force என்பதன் சுருக்கம்).

```
MUTHU@MUTHU-PC MINGW64 /e/test123/My-First-Local-Repository1 (kilai)
$ git checkout master
Switched to branch 'master'
Your branch is up to date with 'moolam/master'.
```

நம்முடைய கணினியில் உள்ள மாஸ்டர் கிளை – இப்போது கிட்ஹப்பில் உள்ள மேல் ஓடை(அப்ஸ்டிரீம்) களஞ்சியத்திற்கு(ரெப்போ) நிகராக உள்ளதா எனப் பார்க்க வேண்டும் அல்லவா? அதாவது,

- 1) நம்முடைய கணினியில் உள்ள களஞ்சியம் தான் நமக்கு மூல(ஆரிஜின்) களஞ்சியம்(ரெப்போ).
- 2) இந்தக் களஞ்சியம் வேறொருவரின் களஞ்சியத்தில் இருந்து தானே தரவுகளை வாங்கியது. அப்போது அது தானே மேல் ஓடை!
- 3) அந்த மேல் ஓடையின் தரவுகளின் ஏதாவது மாற்றங்கள் நடந்திருந்தால், அம்மாற்றங்களை நாமும் எடுத்துக் கொள்ள வேண்டும் அல்லவா?

அதற்கான கட்டளை தான் git merge என்பதாகும்.

git merge melodai/master என்று கொடுக்க வேண்டும்.

இங்கு melodai யார்? master யார் என்பதைப்
போதுமான அளவு பேசிவிட்டோம்.

Your recently pushed branches:

✱ kilai (29 minutes ago)

[Compare & pull request](#)

Branch: kilai ▾

[New pull request](#)

[Create new file](#)

[Upload files](#)

[Find File](#)

[Clone or download ▾](#)

```
MUTHU@MUTHU-PC MINGW64 /e/test123/My-First-Local-Repository1 (master)
$ git merge melodai/master
Already up to date.
```

Pull Request கட்டளையைப் பார்ப்போமா?

Pull Request என்பது என்னவென்று தெரிய வேண்டுமே!

நாம் வேறொருவரின் களஞ்சியத்தை(ரெப்போ) எடுத்து, நம்முடைய பக்கத்திற்குக் கொண்டு வந்தோம். பிறகு, கிளை உருவாக்கி மாற்றங்கள் செய்திருக்கிறோம். இந்த மாற்றங்களை ஏற்றுக் கொள்ளச் சொல்ல வேண்டுமே! அதாவது, 'நான் செய்த மாற்றங்களை இழுத்துக் கொள்ளுங்கள்' என்று சொல்ல வேண்டுமே! அது தான் Pull Request! (pull என்றாலே இழுப்பது தானே!)

இப்போது

<https://github.com/muthu1809/My-First-Local-Repository1/tree/kilai>

பக்கத்திற்குப் போவோம். அங்கு,

Branch: kilai ▼

New pull request

என்று இருக்கும். இதில் 'New pull request' என்பதைச் சொடுக்குங்கள். அதில் நீங்கள் விரும்பும் விவரங்களை உள்ளிட்டு 'Create pull request' ஐச் சொடுக்குங்கள்.

Open a pull request

Create a new pull request by comparing changes across two branches. If you need to, you can also [compare across forks](#).



base repository: **payilagam/My-First-Local-Repo...** **base: master** **head repository: muthu1809/My-First-Local-Re...** **compare: kilai**

✓ Able to merge. These branches can be automatically merged.



Added one line in new branch

Write

Preview

AA B i « > 🔗 ☰ ☷ ☹ @ 📌 ↶

Leave a comment

Attach files by dragging & dropping, selecting or pasting them.

☒ Allow edits from maintainers. [Learn more](#)

Create pull request

payilagam / My-First-Local-Repository1

Watch 1

Star 0

Fork 1

Code

Issues 0

Pull requests 1

Projects 0

Wiki

Security

Insights

Added one line in new branch #1

Edit

Open muthu1809 wants to merge 1 commit into **payilagam:master** from **muthu1809:kilai**

Conversation 0

Commits 1

Checks 0

Files changed 1

+1 -1



muthu1809 commented 4 minutes ago

+1 -1

முதல் Pull Request

Added one line in new branch

d43a755

Reviewers

No reviews

Assignees

No one assigned

Labels

None yet

Add more commits by pushing to the **kilai** branch on **muthu1809/My-First-Local-Repository1**.

Open a pull request

Create a new pull request by comparing changes across two branches. If you need to, you can also [compare across forks](#).

 base: master  compare: kilai  **Able to merge.** These branches can be automatically merged.



Added two lines

Write

Preview

AA

B

i

“

<>

🔗

☰

☷

☰

@

📄

↶

Leave a comment

di

Attach files by dragging & dropping, selecting or pasting them. 

Create pull request 

Rebase கட்டளை:

Pull Request தெரிந்த பிறகு, நமக்குத் தெரிய வேண்டிய முதன்மையான கிட் கட்டளை Rebase.

Rebase என்றால் என்ன?

ஒரு திட்டப்பணி என்பது பலரும் சேர்ந்து வேலை செய்யும் இடம். அதில் மூலத் திட்டப்பணி, மேலோடை திட்டப்பணி, நம்முடைய கிளை திட்டப்பணி ஆகிய மூன்றிற்கும் இடையே பல வித்தியாசங்கள் இருக்க வாய்ப்புகள் உண்டு. அப்படி வித்தியாசங்கள் இருக்கும் போது ஒருவர் செய்யும் மாற்றம், மற்றவர்களின் நிரலில் தவறுகளை ஏற்படுத்தும் நிலை வரலாம். இது போன்ற சூழலில், நம்முடைய மாற்றத்தை மூலத் திட்டப்பணியிலோ, மேலோடை திட்டப்பணியிலோ செய்வது சிக்கலை ஏற்படுத்தும். எனவே, நாம் செய்யும்

மாற்றத்தை ஏற்க கிட் மறுத்துவிடும்.
(Merging நடக்காது).

Output

CONFLICT (content): Merge conflict in
README.md

Automatic merge failed; fix conflicts
and then commit the result.

இதைத் தவிர்க்க, மூலத்
திட்டப்பணி(பிராஜெக்ட் ரெப்போ)யையும்
நம்முடைய திட்டப்பணியையும் ஒன்று போல்
மாற்றுவது தான் ரீபேஸ் (Rebase) ஆகும்.
படிகள்:

- 1) முதலில் கிட் பேஷில் - நம்முடைய
களஞ்சியத்திற்குப் போக வேண்டும்.
- 2) அங்கிருந்து மேலோடை களஞ்சியத்தில்
இருந்து எல்லாத் தகவல்களையும்
எடுத்துக் கொள்ள வேண்டும்.
அதற்கான கட்டளை

git fetch melodai

```
MUTHU@MUTHU-PC MINGW64 /e/test123/My-First-Local-Repository1 (master)  
$ git fetch melodai
```

நாம், நம்முடைய களஞ்சியத்தில்(ரெப்போ) பல மாற்றங்கள் செய்திருப்போம். அவற்றைப் பல முறை 'கிட்' டுடன் சேர்த்து (add), உறுதிப்படுத்தி(commit)யிருப்போம். இப்போது மேலோடை களஞ்சியத்தை(ரெப்போ)யும் நம்முடைய கிளை களஞ்சியத்தையும் ஒன்றாக்க(ரீபேஸ்)ப் போகிறோம். எனவே, நம்முடைய பழைய உறுதிப்படுத்தல்(கம்மிட்)கள் எல்லாவற்றையும் சேர்த்து ஒன்றாக்க விரும்பினால் - ஒன்றாக்கிக் கொள்ளலாம். (தேவையில்லாத கம்மிட்கள் இருந்தால் அவற்றை நீக்கவும் முடியும்.). அதற்கு முதலில் git log கட்டளையைக் கொடுத்து எத்தனை கம்மிட்கள் இருக்கின்றன, என்னென்ன கம்மிட்கள் என்று பார்த்துக் கொள்வோமா?

```
MUTHU@MUTHU-PC MINGW64 /e/test123/My-First-Local-Repository1 (master)
$ git log
commit f785f0d09b97908e5067d31fc2e254b756b1b5ad (HEAD -> master, origin/master,
origin/HEAD, moolam/master, melodai/master)
Author: Payilagam <payilagamchennai@gmail.com>
Date: Sun Oct 13 09:27:59 2019 +0530

Initial commit
```

மேலே ஒரே ஒரு உறுதிப்படுத்தல்(கம்மிட்) தான் காட்டப்படுகிறது. ஆனால், பெரிய பெரிய திட்டப்பணிகளில் பல உறுதிப்படுத்தல்(கம்மிட்)கள் காட்டப்படும். அதுவும், பல்வேறு பயனர்கள் செய்த

உறுதிப்படுத்தல்(கம்மிட்)களாக அவை இருக்கும். அது போன்ற சூழலில், ஒரு குறிப்பிட்ட பயனர் செய்த உறுதிப்படுத்தல்(கம்மிட்) எது என்று கண்டுபிடிக்க,
git log --author=பயனர்-பெயர்
என்று கொடுத்தால் போதும். அந்தப் பயனரின் உறுதிப்படுத்தல்(கம்மிட்) மட்டும் காண்பிக்கப்படும்.

கிளையில் நாம் செய்த மாற்றம் என்ன?

மாஸ்டர் கிளைக்கும் நம்முடைய கிளைக்கும் இடையே இருக்கும் வித்தியாசங்கள் என்னென்ன என்று எப்படிக் கண்டுபிடிப்பது? மிக எளிது! கிளை களஞ்சியத்தில் நாம் மாற்றங்களைச் செய்திருக்கும் போது உறுதிப்படுத்தல்(கம்மிட்) செய்திருப்போம் இல்லையா! அந்த உறுதிப்படுத்தலுக்கான கிட் ஹேஷ் மதிப்பு தெரிந்தால் போதும். அது சரி! அந்த ஹேஷ் மதிப்பை எப்படித் தெரிந்து கொள்வது?

git merge-base kilai master

என்று கொடுத்தால் போதும். ஹேஷ் மதிப்பு என்ன என்று தெரிந்து விடும்.

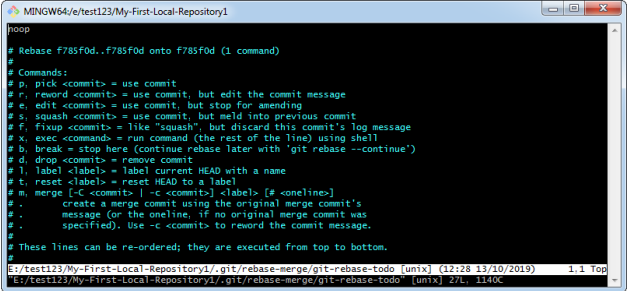
```
MUTHU@MUTHU-PC MINGW64 /e/test123/My-First-Local-Repository1 (master)
$ git merge-base kilai master
f785f0d09b97908e5067d31fc2e254b756b1b5ad
```

இப்போது, அந்த ஹேஷ் மதிப்பை எடுத்து ரீபேஸ் கட்டளைக்குப் பயன்படுத்திக் கொள்ளலாம்.

`git rebase -i`

`f785f0d09b97908e5067d31fc2e254b756b1b5ad` என்று கொடுத்தால் போதும்.

கீழே தெரிவது போல, பல கிட்டளைகளுடன் கூடிய கோப்பு திறக்கும்.



```
MINGW64: e:/test123/My-First-Local-Repository1
hoop
# Rebase f785f0d..f785f0d onto f785f0d (1 command)
# Commands:
# p, pick <commit> = use commit
# r, reword <commit> = use commit, but edit the commit message
# e, edit <commit> = use commit, but stop for amending
# s, squash <commit> = use commit, but meld into previous commit
# f, fixup <commit> = like "squash", but discard this commit's log message
# x, exec <command> = run command (the rest of the line) using shell
# b, break = stop here (continue rebase later with 'git rebase --continue')
# d, drop <commit> = remove commit
# l, label <label> = label current HEAD with a name
# t, reset <label> = reset HEAD to a label
# m, merge [-C <commit> | -c <commit>] <label> [# <oneline>]
# . create a merge commit using the original merge commit's
# message (or the oneline, if no original merge commit was
# specified). Use -c <commit> to reword the commit message.
#
# These lines can be re-ordered; they are executed from top to bottom.
#
E:/test123/My-First-Local-Repository1/.git/rebase-merge/git-rebase-todo [unix] (12:28 13/10/2019) 1,1 Top
E:/test123/My-First-Local-Repository1/.git/rebase-merge/git-rebase-todo" [unix] 27L, 1140C
```

இதில் உள்ள `pick`, `reword`, `squash` ஆகிய கட்டளைகளைப் பயன்படுத்தி, பழைய உறுதிப்படுத்தல்(கம்மிட்)களை ஒன்றாக்குவது, மாற்றுவது ஆகியவற்றைச் செய்ய முடியும். நம்முடைய எடுத்துக்காட்டில் அப்படிப்பட்ட தேவை எதுவும் இல்லை எனவே, `esc`

பொத்தானை அழுத்தி, பிறகு, :q கொடுத்து கோப்பை விட்டு வெளியேறிவிடலாம். இப்போது கீழ் உள்ளது போல, Successfully rebased and updated refs/heads/master என்று திரையில் தோன்றும்.

```
MUTHU@MUTHU-PC MINGW64 /e/test123/My-First-Local-Repository1 (master)
$ git rebase -i f785f0d09b97908e5067d31fc2e254b756b1b5ad
Successfully rebased and updated refs/heads/master.
```

இப்போது ஏதாவது சிக்கல்கள் / வேறுபாடுகள்(conflicts) இருந்தால் கிட்ட நமக்குத் தெரியப்படுத்தும். வேறுபாடுகள் இல்லை என்றால், git rebase --continue கொடுத்து விடலாம்.

ரீபேஸ் கட்டளை முடிந்த பிறகு, நம்முடைய கிளைக்குப் போக வேண்டும் (நினைவு இருக்கிறதா - git checkout கட்டளை!). அதற்குப் பிறகு, git push -f கட்டளையைக் கொடுத்து விடலாம். இங்கு -f என்பது force - அதாவது என்ன ஆனாலும் சரி, என்னுடைய களஞ்சியத்தைத் தள்ளுகிறேன் என்பது அர்த்தம்! நாம் தான், நம்முடைய கட்டளையை ரீபேஸ் செய்துவிட்டோமே - அதனால் தான் விசையோடு(force) தள்ளுகிறோம்.

Pull Request க்குப் பிறகு என்ன செய்வது?

Pull request என்பது நம்முடைய மாற்றங்களை மூலக் களஞ்சியத்தில் (ரெப்போ) இழுத்துக் கொள்ள வேண்டுவது! இந்த வேண்டுகோளை ஏற்றுக் கொண்டு விட்டால் - அதற்குப் பிறகு நம்முடைய கணினியில் உள்ள களஞ்சியத்தை மூலக் களஞ்சியத்திற்கு ஏற்ப நிகர்ப்படுத்த (update) வேண்டும் அல்லவா?

1) முதலில் git checkout master கொடுக்க வேண்டும்.

```
MUTHU@MUTHU-PC MINGW64 /e/test123/My-First-Local-Repository1 (kilai)
$ git checkout master
Switched to branch 'master'
Your branch is up to date with 'moolam/master'.
```

2) பிறகு, git pull --rebase melodai master கொடுக்க வேண்டும்.

```
MUTHU@MUTHU-PC MINGW64 /e/test123/My-First-Local-Repository1 (master)
$ git pull --rebase melodai master
remote: Enumerating objects: 1, done.
remote: Counting objects: 100% (1/1), done.
remote: Total 1 (delta 0), reused 0 (delta 0), pack-reused 0
Unpacking objects: 100% (1/1), done.
From https://github.com/payilagam/My-First-Local-Repository1
 * branch      master      -> FETCH_HEAD
  f785f0d..c6889a2 master   -> melodai/master
Updating f785f0d..c6889a2
Fast-forward
 README.md | 2 +-
 1 file changed, 1 insertion(+), 1 deletion(-)
Current branch master is up to date.
```

3) கடைசியாக, git push -f moolam master கொடுக்க வேண்டும்.

4) இதன் பிறகு, நம்முடைய கிளையை நீக்கி விடலாம். அதற்கு,

`git branch -d kilai` என்று கொடுத்தால் போதும்.

```
MUTHU@MUTHU-PC MINGW64 /e/test123/My-First-Local-Repository1 (master)
$ git branch -d kilai
Deleted branch kilai (was d43a755).
```

5) இப்படிச் கணினியில் உள்ள கிளை களஞ்சியத்தை நீக்கிய பிறகு, கிட் ஹப்பில் உள்ள மூலக் களஞ்சியத்தையும் நீக்கி விடலாம்.

`git push moolam --delete kilai`

‘கிட்’டில் SSH Key எப்படி உருவாக்குவது?

SSH என்பது ஒரு பயனர் மிகவும் பாதுகாப்பாக கிட் தளத்தில் உள்ள தம்முடைய களஞ்சியத்தை அணுகப் பயன்படும்.

1. கிட் பேஷ் திறந்து கொள்ளுங்கள்.
2. கீழே இருப்பது போல, தட்டச்சிடுங்கள்.
இதில் உங்கள் மின்னஞ்சல் முகவரியை மாற்றி விடுங்கள்.

```
$ ssh-keygen -t rsa -b 4096 -C  
your\_email@example.com
```

3. பிறகு தேவைப்பட்டால் கடவுச்சொல் கொடுக்கலாம். இல்லாவிட்டால் அதை வெறுமனே விட்டு விடலாம்.

```
MUTHU@MUTHU-PC MINGW64 ~  
$ ssh-keygen -t rsa -b 4096 -C "muthu1809@gmail.com"  
Generating public/private rsa key pair.  
Enter file in which to save the key (/c/Users/MUTHU/.ssh/id_rsa):  
Enter passphrase (empty for no passphrase):  
Enter same passphrase again:  
Your identification has been saved in /c/Users/MUTHU/.ssh/id_rsa.  
Your public key has been saved in /c/Users/MUTHU/.ssh/id_rsa.pub.  
The key fingerprint is:  
SHA256:Gt71K5j+Qd9n/BN2vCDjUntzh2hhKHawxBP/3JvikZQ muthu1809@gmail.com  
The key's randomart image is:  
+---[RSA 4096]-----+  
|          ..o  |  
|         .  E  |  
|        . o . . |  
|       = . o .. |  
|      ..S+o+...o |  
|     . ++oo+=o.+o |  
|    o..=+. =o+.B |  
|     o..+o+. =o |  
|    ...ooo o + |  
+-----[SHA256]-----+
```

```
MUTHU@MUTHU-PC MINGW64 ~  
$ eval $(ssh-agent -s)  
Agent pid 1789
```

```
MUTHU@MUTHU-PC MINGW64 ~  
$ ssh-add ~/.ssh/id_rsa  
Identity added: /c/Users/MUTHU/.ssh/id_rsa (muthu1809@gmail.com)
```


துணை நின்ற தளங்கள்:

- 1) <https://git-scm.com>
- 2) <https://www.git-tower.com>
- 3) <https://www.digitalocean.com>
- 4) <https://www.techaaroorian.com>
- 5) <https://guides.github.com/activities/hello-world/>

நூல் ஆசிரியர் பற்றி:
பெயர்: கி. முத்துராமலிங்கம்
தொழில்: பயிற்றுநர், பயிலகம் பயிற்சி
நிறுவனம், வேளச்சேரி, சென்னை.

(<https://payilagam.com>)

படித்தது: பொறியியல்

பிடித்தது: தமிழ்

தொடர்புக்கு: muthu1809@gmail.com

எழுதிய இன்னொரு நூல்: [எளிய தமிழில்
சாப்ட்வேர் டெஸ்டிங்](#)