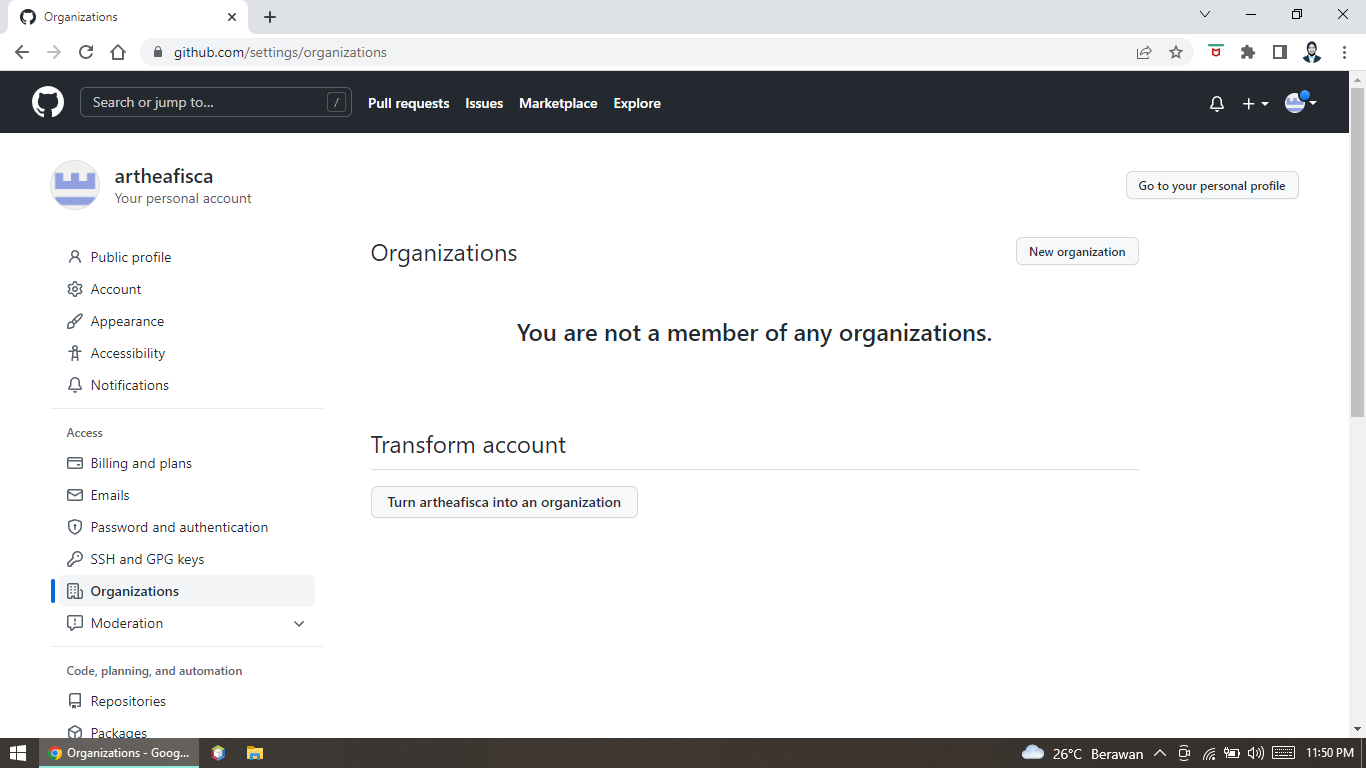
**Alat 1: Menambahkan Aggota Tim**

Biasanya ada dua cara menyiapkan Github untuk kolaborasi tim:

1. **Organizations** - Pemilik organisasi dapat membuat banyak tim dengan tingkat izin yang berbeda untuk berbagai repositori
2. **Collaborators** - Pemilik repositori dapat menambahkan kolaborator dengan akses Baca + Tulis untuk satu repositori

Organizations

Jika Anda mengawasi beberapa tim dan ingin menetapkan tingkat izin yang berbeda untuk setiap tim dengan berbagai anggota dan menambahkan setiap anggota ke repositori yang berbeda, maka Organization akan menjadi pilihan terbaik. Akun pengguna Github apa pun sudah dapat membuat Organizations gratis untuk repositori kode sumber terbuka. Untuk membuat Organization, cukup telusuri ke [halaman pengaturan organisasi Anda](https://github.com/settings/organizations):

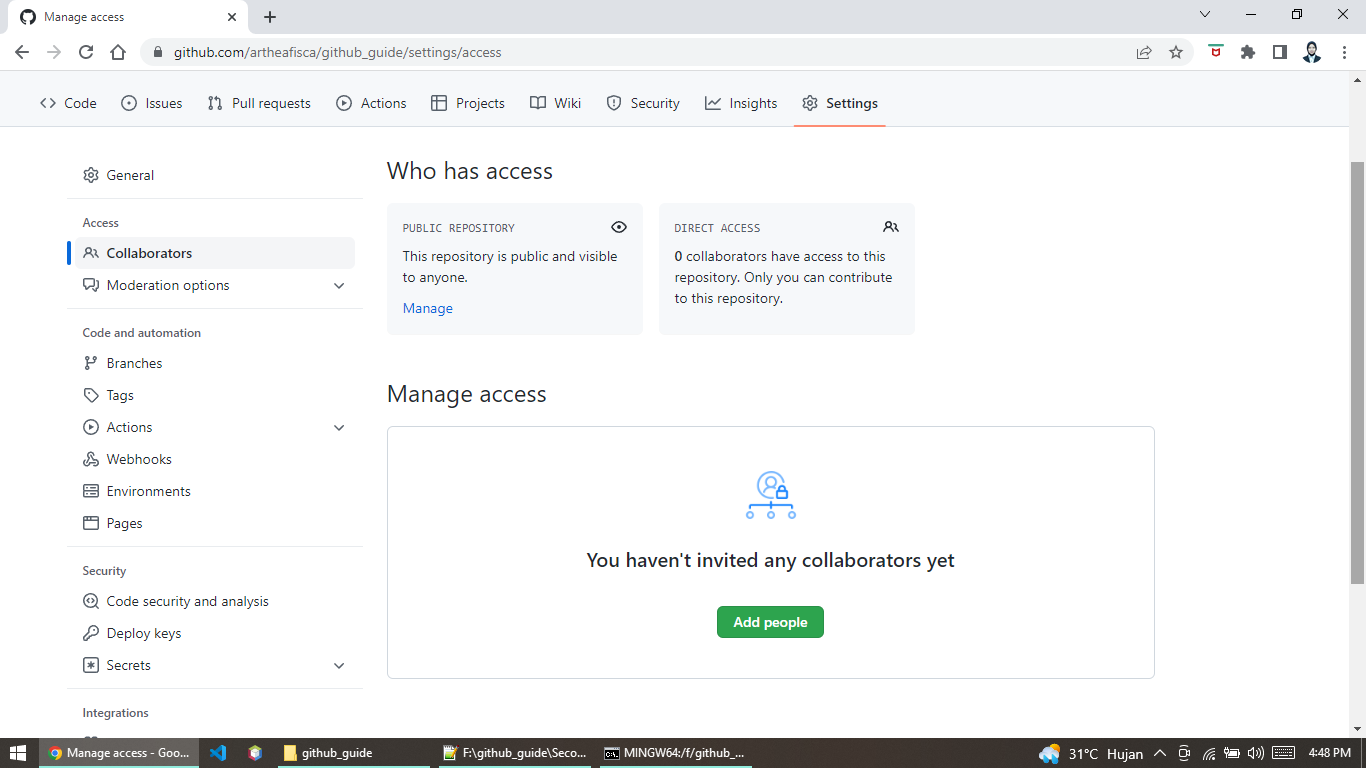


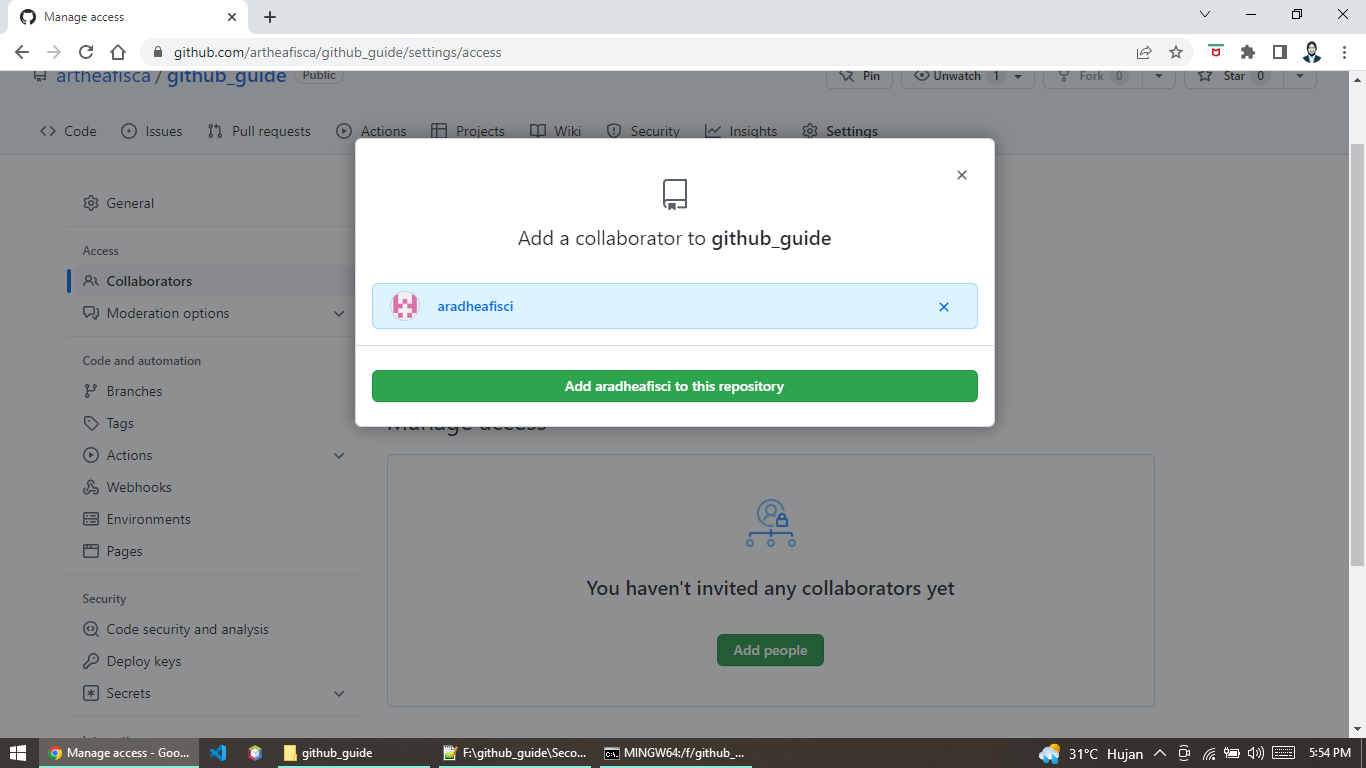
Untuk mengakses halaman tim untuk Organization Anda, Anda cukup pergi ke http://github.com/organizations/[organization-name]/teams untuk melihatnya atau bahkan mengunjungi https://github.com/organizations/[organization-name]/teams/new untuk membuat tim baru dengan anggota dari 3 tingkat izin yang berbeda seperti:

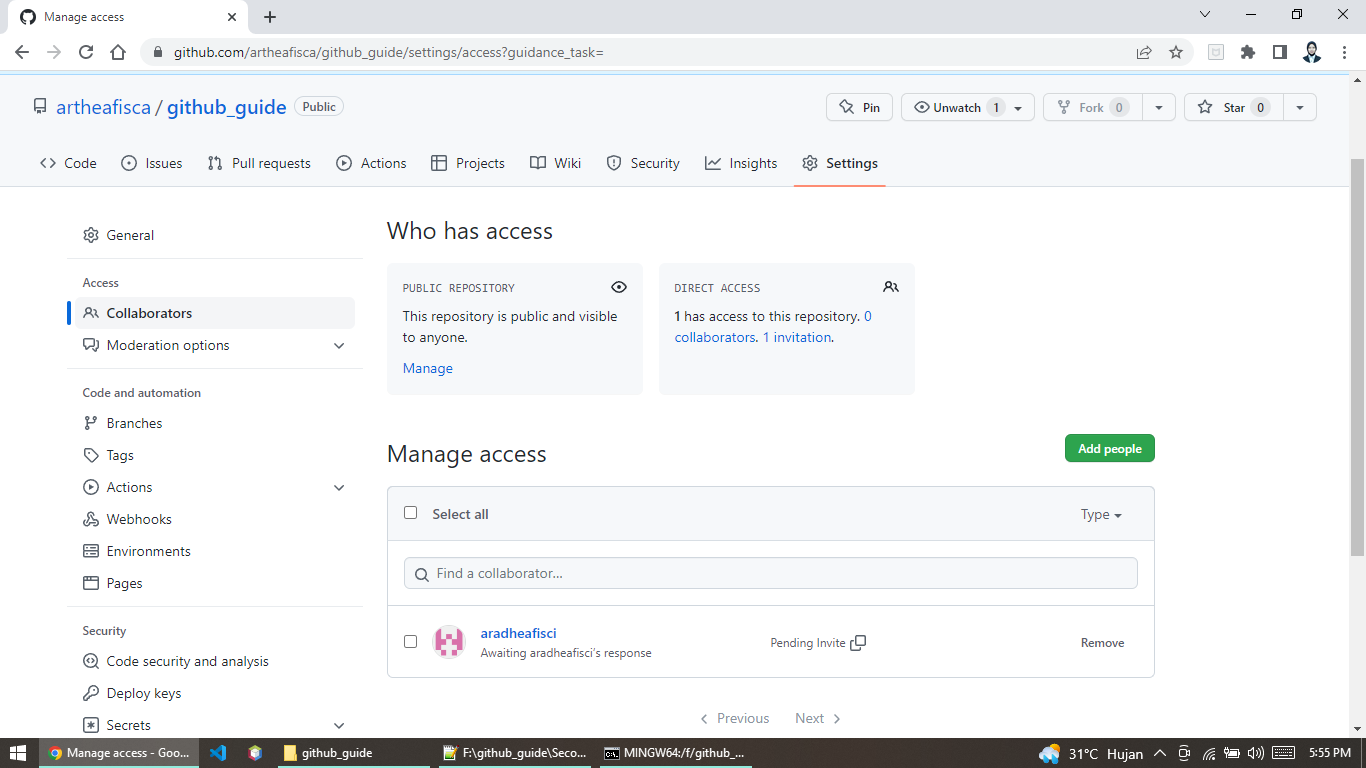
1. **Pull Only:**[**Fetch and Merge**](https://www.kernel.org/pub/software/scm/git/docs/git-pull.html)dengan repositori lain atau salinan lokal. Akses hanya baca.
2. **Push and Pull:** (1) bersama dengan [pembaruan](https://www.kernel.org/pub/software/scm/git/docs/git-push.html) reparasi jarak jauh. Baca + Akses tulis.
3. **Push, Pull & Administrative:** (1), (2) bersama dengan hak atas info penagihan, membuat tim serta membatalkan akun Organization. Baca + Tulis + akses Admin

**Collaborators**

Collaborators digunakan untuk memberikan akses **Read + Write access** ke satu repositori yang dimiliki oleh akun pribadi. Untuk [**menambahkan Collaborators**](https://help.github.com/articles/how-do-i-add-a-collaborator), (akun pribadi Github lainnya), cukup buka https://github.com/[username]/[repo-name]/settings/collaboration:







Setelah itu selesai, masing-masing Collaborator kemudian akan melihat perubahan dalam status akses pada halaman repositori. Setelah kita memiliki akses Write ke repositori, kita dapat melakukan git clone, bekerja pada perubahan, membuat git pull untuk mengambil dan menggabungkan setiap perubahan dalam repositori jarak jauh dan akhirnya git push, untuk memperbarui repositori jarak jauh dengan perubahan kita sendiri.

**Alat 2: Pull Requests**

[Pull Requests](https://help.github.com/articles/using-pull-requests) adalah cara yang mengagumkan untuk berkontribusi pada repositori secara mandiri dengan fork itu. Pada akhirnya, jika kita mau, kita dapat mengirim permintaan tarik ke pemilik repositori untuk menggabungkan perubahan kode kita. Pull request itu sendiri dapat memicu diskusi untuk kualitas kode, fitur atau bahkan strategi umum.

Sekarang mari kita melalui langkah-langkah dasar untuk [pull request](https://help.github.com/articles/using-pull-requests).

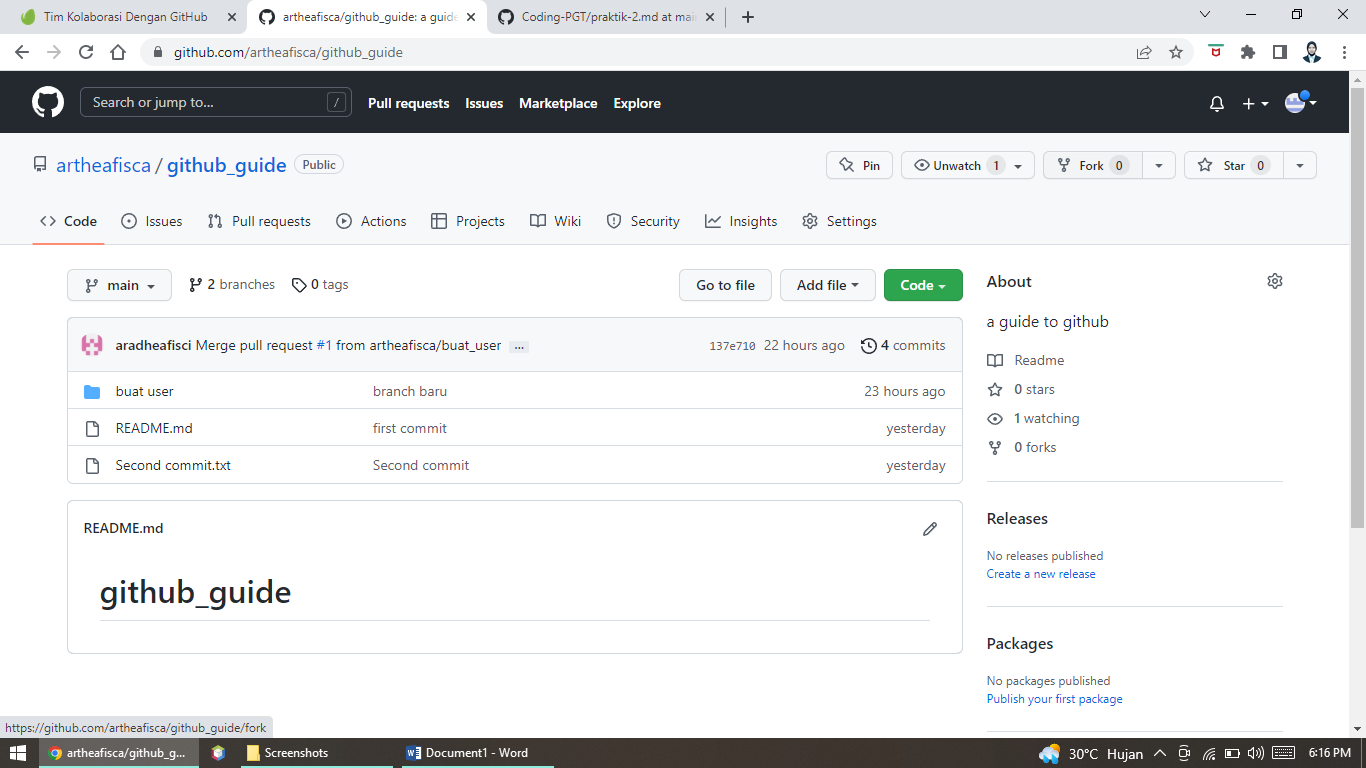
**Memulai Pull Request**

Ada dua model pull request di Github:

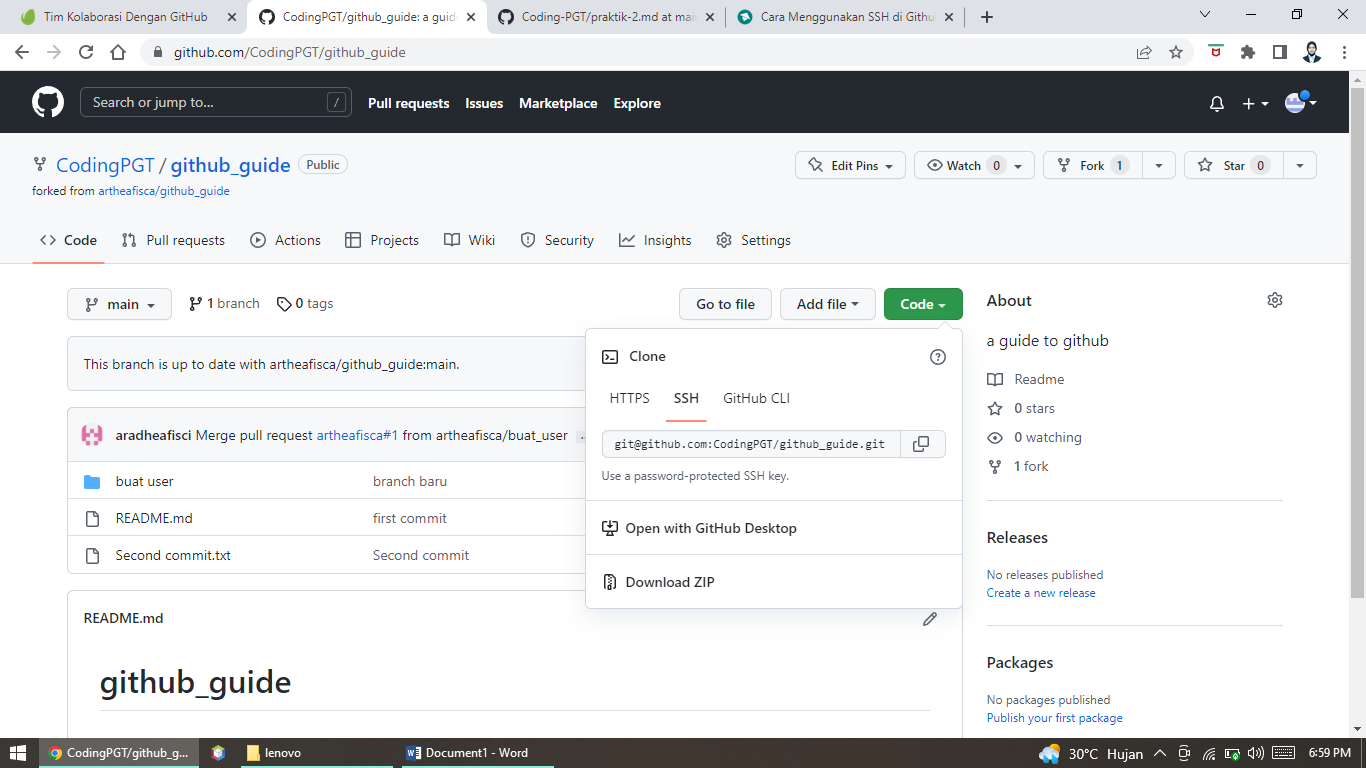
1. **Fork & Pull Model** – digunakan di repositori public yang tidak memiliki akses push
2. **Share Repository Model** – digunakan dalam repositori pribadi yang kita miliki akses push. Fork tidak diperlukan adalah case ini.

Di sini kita melihat alur kerja antara dua pengguna (repo-owner dan forked-repo-owner) untuk model Fork and Pull:

1. Identifikasi Github Repository yang ingin Anda kontribusikan, dan klik tombol “Fork” untuk membuat tiruan dari repository di akun Github Anda sendiri.



1. Ini akan membuat Salinan persis dari repositori di akun Anda sendiri.



1. [Pilih URL SSH](https://help.github.com/articles/why-is-git-always-asking-for-my-password)sehingga akan meminta kata kunci SSH Anda, bukan nama pengguna dan kata sandi setiap kali Anda git push atau git pull. Selanjutnya, kita akan mengkloning repositori ini ke komputer local.

$ git clone [ssh-url] [folder-name]

$ cd [folder-name]

1. Secara umum, kita akan membuat cabang git baru untuk setiap fitur baru. Ini adalah praktik yang baik karena di masa depan jika kita memperbarui cabang lebih lanjut setelah beberapa diskusi, [**pull request akan diperbarui secara otomatis**](https://stackoverflow.com/questions/9790448/how-to-update-a-pull-request). Mari buat cabang baru untuk membuat perubahan yang sangat sederhana untuk mengubah file: readme.md

$ git checkout -b [new-feature]

1. Setelah membuat tambahan yang relevan untuk membangun fitur baru, kami hanya akan melakukan perubahan baru dan checkout ke cabang master git:

$ git add .

$ git commit -m "information added in readme"

$ git checkout master

1. Pada titik ini, kami akan mendorong cabang ke repositori jarak jauh. Untuk ini pertama-tama kita akan memeriksa nama cabang dengan fitur baru serta alias repositori jarak jauh git. Kemudian kita akan mendorong perubahan menggunakan git Push [git-remote-alias] [branch-name].

$ git branch

\* master

readme

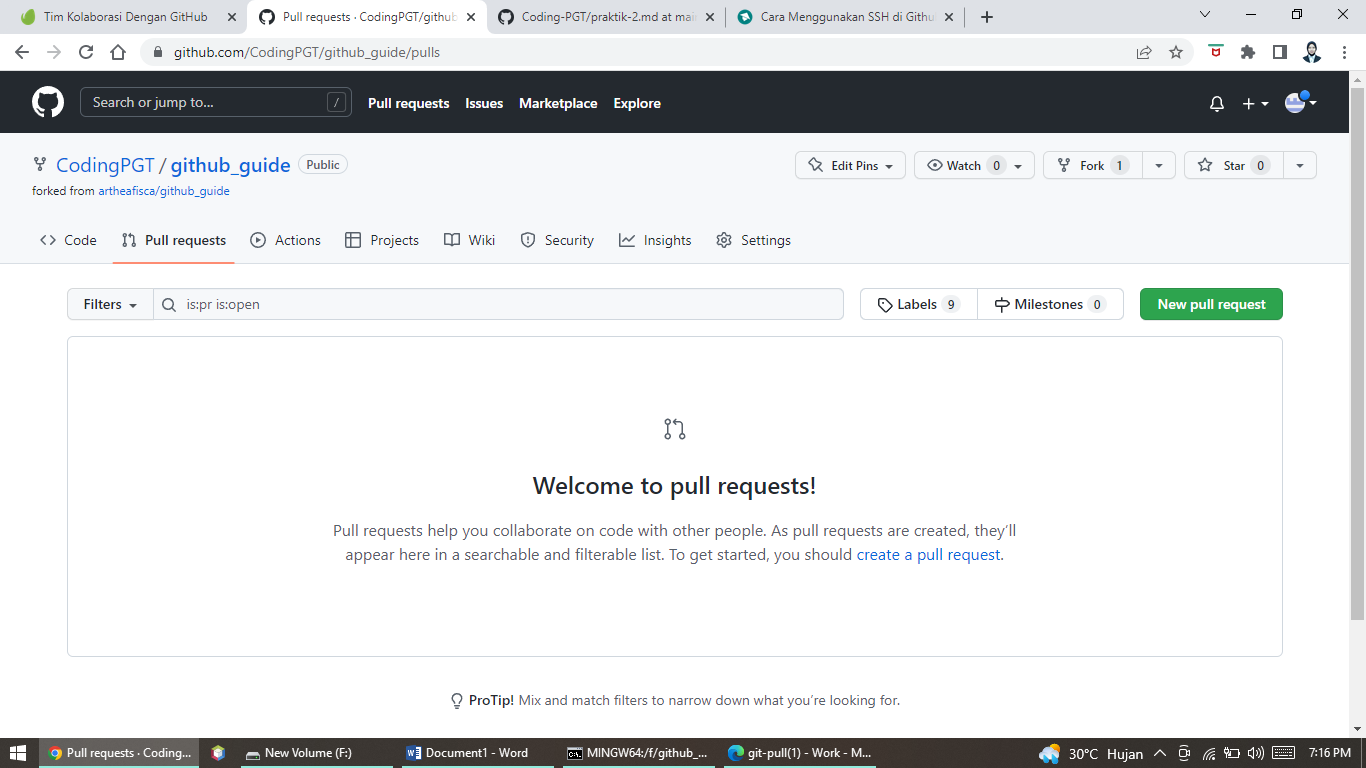
$ git remote -v

origin git@github.com:[forked-repo-owner-username]/[repo-name].git (fetch)

origin git@github.com:[forked-repo-owner-username]/[repo-name].git (push)

$ git push origin readme

1. Di halaman Github repositori bercabang kami, kami akan mengubah ke cabang dengan fitur baru dan kemudian menekan tombol "Pull Request".



1. Setelah mengirimkan permintaan tarik, itu akan langsung membawa kita ke halaman permintaan tarik repositori asli. Kami akan melihat permintaan tarik kami, baik sebagai masalah baru maupun permintaan tarik baru.
2. Setelah diskusi, mungkin pemilik repositori bercabang mungkin ingin menambahkan perubahan pada fitur baru. Dalam hal ini, kami akan checkout ke cabang yang sama di mesin lokal kami, melakukan itu, dan mendorongnya kembali ke Github. Ketika kami mengunjungi halaman permintaan tarik dari repositori asli, itu akan diperbarui secara otomatis!

**Penggabungan Pull Request**

Jika Anda adalah pemilik repositori asli, ada dua cara untuk menggabungkan permintaan tarik yang masuk:

1. Menggabungkan langsung di Github: Jika kita menggabungkan langsung di Github, pastikan tidak ada konflik dan siap untuk digabung ke cabang master. Pemilik repositori asli cukup mengklik tombol hijau "Gabungkan Permintaan Tarik" untuk melakukannya.
2. Penggabungan di mesin lokal kami: Di lain waktu, mungkin ada konflik penggabungan, dan setelah mengklik tombol "info", Github akan memiliki instruksi yang jelas tentang bagaimana kami dapat menggabungkan cabang di mesin lokal kami dengan menarik perubahan dari cabang kontributor.

**Alat 3: pelacak Bug**

Pull Request adalah cara yang luar biasa untuk berkontribusi ke repositori secara mandiri dengan melakukan forking.

Di Github, pusat untuk semua pelacakan bug adalah Masalah. Meskipun mereka terutama untuk pelacakan bug, juga berguna untuk menggunakan Masalah dengan cara berikut:

1. Bug: Hal-hal yang jelas rusak dan perlu diperbaiki
2. Fitur: Ide-ide baru yang keren dan keren untuk diterapkan.
3. To do list: Daftar periksa item yang harus diselesaikan.

Mari kita jelajahi beberapa fitur dari Issues:

1. Label: Mereka adalah kategori berwarna untuk setiap edisi. Mereka sangat membantu untuk menyaring masalah yang sesuai.
2. Milestones: Mereka adalah kategori tanggal yang dapat dikaitkan dengan setiap masalah dan berguna untuk mengidentifikasi masalah apa yang perlu dikerjakan untuk rilis berikutnya. Juga karena Milestones terhubung ke masalah, secara otomatis memperbarui bilah kemajuan setelah menutup setiap masalah terkait.
3. Pencarian: Pencarian lengkapi otomatis untuk masalah dan pencapaian.
4. Penugasan: Setiap masalah dapat ditugaskan ke orang yang bertanggung jawab untuk memperbaiki masalah tersebut. Ini adalah fitur lain yang berguna untuk melihat apa yang harus kita kerjakan.
5. Tutup otomatis: Komit pesan dengan Perbaikan/Tetap atau Tutup/Tutup/Tutup #[nomor-masalah] akan secara otomatis menutup masalah.

$ git add .

$ git commit -m "corrected url. fixes #2"

$ git push origin master

1. Sebutan: Siapa pun juga dapat meninggalkan catatan dengan hanya menunjukkan #[nomor-masalah] di pesan mereka. Karena nomor masalah memiliki hyperlink, ini membuatnya sangat mudah untuk menyebutkan masalah terkait selama diskusi.

**Alat 4: Analisis**

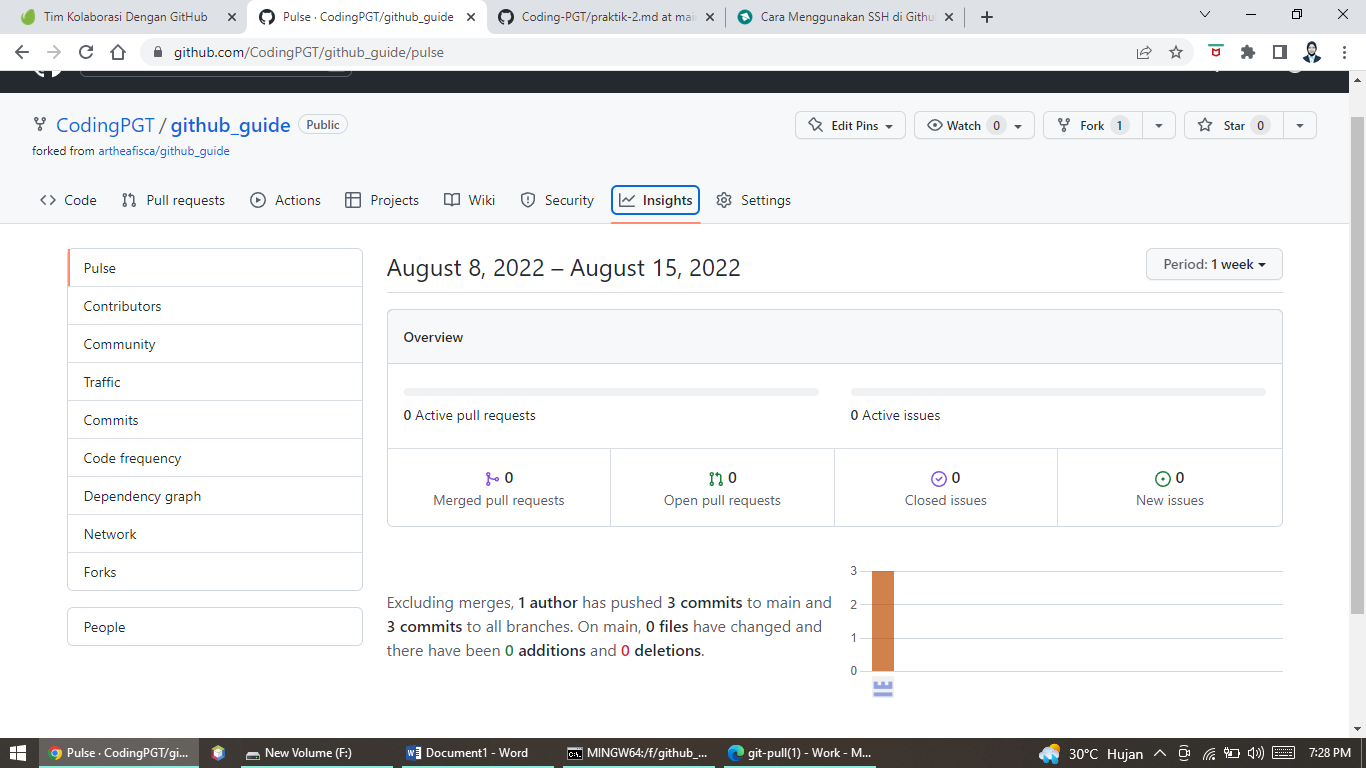
Jelas bahwa kami dapat dengan erat memasangkan daftar tugas kami dan pembaruan untuk komit kode kami.

Ada dua alat yang memberikan wawasan tentang repositori - Grafik dan Jaringan. Grafik Github memberikan wawasan tentang kolaborator dan komitmen di balik setiap repositori kode, sementara Github Network menyediakan visualisasi pada setiap kontributor dan komitmen mereka di semua repositori bercabang. Analisis dan grafik ini menjadi sangat kuat, terutama saat bekerja dalam tim.

**Grafik**

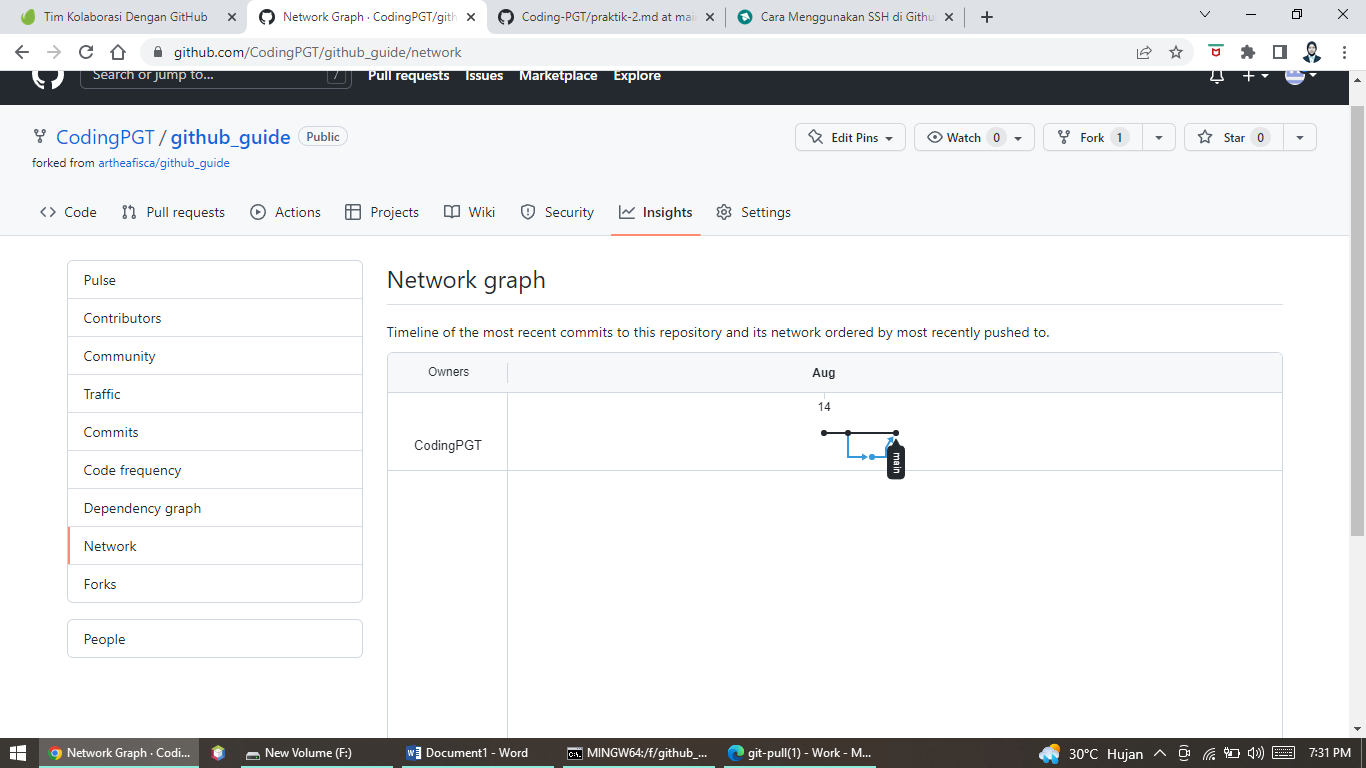
Grafik memberikan analisis terperinci seperti:

1. **Kontributor**: Siapa saja kontributornya? Dan berapa banyak baris kode yang mereka tambahkan atau hapus?
2. **Aktivitas Komit**: Minggu mana komit terjadi dalam setahun terakhir?
3. **Frekuensi Kode**: Berapa banyak baris kode yang dilakukan di seluruh siklus hidup proyek?
4. **Punchcard**: Pada jam berapa biasanya commit dilakukan?



**Jaringan**

[Github Network](https://github.blog/2008-04-10-say-hello-to-the-network-graph-visualizer/) adalah alat yang ampuh yang memungkinkan Anda melihat komitmen setiap kontributor dan bagaimana mereka terkait satu sama lain. Ketika kita melihat visualisator secara keseluruhan, kita melihat setiap komit pada setiap cabang dari setiap repositori yang dimiliki jaringan. Wawasan!



**Alat 5: Manajemen Proyek**

Sementara Masalah Github memiliki kemampuan manajemen proyek dengan Masalah dan Tonggak, beberapa tim mungkin lebih memilih alat lain karena fitur lain atau alur kerja yang ada. Di bagian ini, kita akan melihat bagaimana kita dapat menghubungkan Github dengan dua alat manajemen proyek populer lainnya - Trello dan Pivotal Tracker. Dengan kait layanan Github, kami dapat mengotomatiskan tugas pembaruan dengan komitmen, masalah, dan banyak aktivitas lainnya. Otomatisasi ini membantu tidak hanya menghemat waktu, tetapi juga meningkatkan akurasi pembaruan untuk setiap tim pengembangan perangkat lunak.

**Github dan Trello**

Trello menyediakan cara sederhana dan visual untuk mengelola tugas. Menggunakan metodologi Pengembangan Perangkat Lunak Agile, kartu Trello dapat meniru Papan Kanban virtual sederhana. Sebagai contoh, kami akan secara otomatis membuat kartu Trello setiap kali Permintaan Tarik dibuat menggunakan Kait Layanan Github. Mari kita pergi melalui langkah-langkah!

1. Buka akun di Trello jika Anda belum memilikinya dan buat Trello Board baru.
2. Buka repositori Github > Pengaturan > Kait Layanan > Trello
3. Dapatkan TOKEN di bawah Instal Nites #1 dengan hyperlink yang disediakan untuk otentikasi.
4. Di bawah Instal Catatan #2, gunakan URL yang diberikan untuk menghasilkan struktur berformat json yang memberi kita id daftar untuk setiap kartu Trello. BOARDID adalah bagian dari URL ketika kami mengunjungi board di [https://trello.com/board/[BOARD-NAME]/[BOARDID]](https://trello.com/board/%5BBOARD-NAME%5D/%5BBOARDID%5D). TOKEN dapat dibuat dengan hyperlink yang diberikan di bawah Install Notes #1.
5. kembali ke kait layanan Github, masukkan id daftar dan token. Centang Aktif, Uji Kait, dan kami siap untuk mendapatkan pembaruan otomatis setiap kali ada Permintaan Tarik.
6. Saat berikutnya ada Pull Request, kartu Trello Pull Request akan otomatis memiliki item baru!

**Github dan Pivotal Tracker**

[**Pivotal Tracker**](https://www.pivotaltracker.com/) adalah alat manajemen proyek ringan ringan lainnya di mana perencanaan berdasarkan cerita memungkinkan tim untuk berkolaborasi dengan mudah dengan segera bereaksi terhadap berbagai perubahan dan kemajuan proyek. Berdasarkan kemajuan tim saat ini, itu juga dapat membuat bagan untuk menganalisis kecepatan tim, iterasi yang membakar, melepaskan burn-down, dll. Dalam contoh singkat ini, kita akan secara otomatis menyampaikan kisah dengan menautkannya ke komitmen Github!

1. Buat proyek baru di Pivotal Tracker dengan kisah baru yang perlu disampaikan.
2. Pergi ke Profil > API Token (kanan dibagian bawah). Salin token API yang diberikan
3. Kembali ke Github repository > Settings > Service Hooks > Pivotal Tracker. Tempel token, centang Aktif dan klik Update Settings. Kita siap untuk secara otomatis memberikan Pivotal Tracker Stories dengan Github Commits!
4. Akhirnya kita akan melakukan perubahan dan [menambahkan id tracker ke pesan commit](http://pivotallabs.com/level-up-your-development-workflow-with-github-pivotal-tracker/) dengan format git commit -m "message [delivers #tracker\_id]"
5. Sekarang, ketika kita kembali ke Pivotal Tracker, kita akan melihat bahwa cerita telah secara otomatis disampaikan dengan tautan ke komit Github yang tepat yang menunjukkan perubahan file!

Dengan contoh Trello dan Pivotal Tracker ini, jelas bahwa kita dapat dengan erat memasangkan daftar tugas kita dan memperbarui kode kita. Ini adalah penghemat waktu yang luar biasa saat bekerja dalam tim, dan ini meningkatkan akurasi saat menautkan tugas ke komitmen yang tepat. Kabar baiknya adalah, jika Anda sudah menggunakan alat manajemen proyek lain seperti [**Asana**](https://asana.com/), [**Basecamp**](https://basecamp.com/), dan lainnya, Anda juga dapat membuat Kait Layanan dengan cara yang sama. Jika tidak adaService Hooks yang ada untuk alat manajemen proyek Anda saat ini, [**Anda bahkan dapat membuatnya**](https://github.com/github/github-services)!

**Alat 6: Continuous Integration**

[**Continuous Integration**](https://en.wikipedia.org/wiki/Continuous_integration) (CI) adalah bagian penting dari semua proyek pengembangan software yang bekerja dengan tim. CI memastikan bahwa, ketika seorang pengembang memeriksa dalam kode mereka, sebuah build otomatis (termasuk tes) mendeteksi kesalahan integrasi secepat mungkin. Ini jelas mengurangi kesalahan integrasi dan membuat iterasi cepat jauh lebih efisien. Dalam contoh ini, kita akan melihat bagaimana [**Travis CI**](https://travis-ci.org/) dapat digunakan dengan Github untuk CI untuk mendeteksi kesalahan serta merekomendasikan penggabungan ketika melewati semua tes.

**Menyiapkan Travis CI**

Kita akan menggunakan proyek "hello-world" sederhana untuk [**node.js**](https://nodejs.org/) dengan [**grunt.js**](https://gruntjs.com/) sebagai alat pembangunan untuk menyiapkan proyek Travis CI. Berikut ini file-file dalam proyek:

1. File hello.js adalah proyek node. Di sini kita dengan sengaja akan meninggalkan titik koma sehingga tidak akan lulus alat penggiling mendengus untuk linting:

var http = require('http');

http.createServer(function (req, res) {

res.writeHead(200, {'Content-Type': 'text/plain'});

res.end('Hello World in Node!\n') // without semicolon, this will not pass linting

}).listen(1337, '127.0.0.1');

console.log('Server running at <http://127.0.0.1:1337/>');

1. package.json menunjukkan dependensi:

{

"name": "hello-team",

"description": "A demo for github and travis ci for team collaboration",

"author": "name <email@email.com>",

"version": "0.0.1",

"devDependencies": {

"grunt": "~0.3.17"

},

"scripts": {

"test": "grunt travis --verbose"

}

}

1. Alat build gruntjs hanya memiliki satu tugas (linting) hanya untuk kesederhanaan:

module.exports = function(grunt) {

grunt.initConfig({

lint: {

files: ['hello.js']

}

});

grunt.registerTask('default', 'lint');

grunt.registerTask('travis', 'lint');

};

1. . travis.yml adalah file konfigurasi Travis yang akan membuat Travis menjalankan pengujian kami:

language: node\_js

node\_js:

- 0.8

1. Selanjutnya, masuk ke Travis dengan Akun Github Anda dan nyalakan kait repositori di bawah tab repositori.
2. Jika langkah sebelumnya tidak memicu build, maka kita harus mengatur service hook secara manual. Untuk mengaturnya secara manual, salin Token di bawah tab profil.
3. Kembali ke repositori Github untuk menyiapkan kait layanan Github Travis dengan token
4. untuk pertama kalinya, kita perlu melakukan git push manual untuk memicu build Travis pertama dan jika semuanya baik-baik saja, kunjungi [http://travis-ci.org/[username]/[repo-name]](http://travis-ci.org/%5Busername%5D/%5Brepo-name%5D) untuk melihat build status lulus!

**Travis CI dengan Pull Requests**

Sebelumnya, tanpa CI dalam proses permintaan tarik, langkah-langkahnya seperti ini (1) mengirim pull request (2) menggabungkan (3) pengujian untuk melihat apakah lulus/gagal. Dengan Travis CI terhubung dengan pull requests, kita akan dapat membalikkan langkah 2 dan 3, semakin cepat membuat keputusan tentang apakah atau tidak itu baik untuk bergabung dengan Travis memberi kita status dengan setiap pull requests. Mari kita lihat cara mewujudkannya.

1. Kirim Pull Request dengan membangun lewat dan Travis akan melakukan sihir untuk memberi tahu Anda bahwa itu baik untuk menggabungkan bahkan sebelum menggabungkan
2. Jika Pull Request gagal membangun, Travis juga akan memperingatkan Anda.
3. Jika kita mengklik tombol tanda merah, itu juga akan menghubungkan ke halaman travis untuk menunjukkan kepada kita rincian dari build.

Travis CI dengan Github sangat berguna untuk tim karena membangun otomatis dan pemberitahuan segera. Ini tentu membuat siklus koreksi correction cycle lebih pendek. Jika Anda menggunakan [**Jenkins**](https://jenkins-ci.org/), CI tool lain yang populer, ya Anda dapat mengatur hook layanan Github sangat mirip juga.

**Alat 7: Code Review**

Dengan setiap commit, Github memungkinkan interface yang bersih untuk komentar umum atau bahkan komentar khusus pada baris kode. Kemampuan untuk mengajukan komentar atau pertanyaan pada setiap baris kode sangat berguna dalam melakukan tinjauan kode baris demi baris. Untuk melihat komentar inline, aktifkan kotak centang di kanan atas setiap commit.

1. **Bandingkan cabang / tag / SHA1** : gunakan pola URL [https://github.com/[username]/[repo-name]/compare/[starting-SHA1]...[ending-SHA1]](https://github.com/%5Busername%5D/%5Brepo-name%5D/compare/%5Bstarting-SHA1%5D...%5Bending-SHA1%5D). Anda dapat melakukan hal yang sama dengan cabang atau tag.
2. **Bandingkan tanpa spasi** : tambahkan ?w=1 ke url perbandingan
3. **Diff** : tambahkan .diff ke URL perbandingan untuk mendapatkan informasi git diff dalam teks biasa. Ini bisa berguna untuk tujuan scripting.
4. **Patch** : tambahkan .patch ke URL perbandingan untuk mendapatkan informasi git diff yang diformat untuk pengiriman patch email.
5. **Line Linking** : Ketika kita mengklik nomor baris pada file apapun, Github akan menambahkan #line di akhir URL dan membuat seluruh warna latar belakang baris menjadi kuning. Ini rapi untuk menunjukkan garis yang tepat. Itu juga menerima rentang garis dengan menambahkan #start-end. Berikut adalah contoh-contoh dari line linking dan line range linking.

**Alat 8: Documenting**

Di bagian ini, kita akan mengeksplorasi dua metode dokumentasi:

1. **Formal Documentation**: Github Wiki untuk membuat dokumentasi untuk kode sumber
2. **Informal Documentation**: [**Github Hubot**](https://github.com/github/hubot) untuk mendokumentasikan diskusi di antara anggota tim dan mengotomatiskan pengambilan informasi dengan berinteraksi dengan bot obrolan yang menyenangkan!
3. **Mentions, Shortcuts & Emoji**

**GitHub Wiki**

Wiki Github dapat dibuat dengan masing-masing repositori, dan ini bisa sangat berguna untuk menempatkan kedua kode sumber dan dokumentasi di bawah repositori yang sama. Untuk membuat Wiki, cukup akses tab Wiki di header utama dan Anda diatur untuk membuat halaman dengan informasi. Wiki sendiri memiliki versi sendiri, dan datanya dapat dikloning ke mesin lokal untuk pembaruan atau bahkan akses offline.

Satu hal yang saya anggap sangat berguna adalah mengintegrasikan Github Wiki ke dalam proyek kode sumber utama sehingga saya tidak perlu mempertahankan dua proyek git yang terpisah. Untuk melakukan ini, saya menambahkan repo Wiki git sebagai [**submodule**](https://git-scm.com/book/ch6-6.html) dari cabang master. Jika Anda menggunakan Travis CI atau CI lainnya, pastikan bahwa alat build [**mengabaikan submodul wiki**](https://github.com/travis-ci/travis-build/pull/46). Untuk file Travis CI .travis.yml, tambahkan yang berikut:

git:

submodules: false

kemudian tambahkan wiki submitul git ke repositori ke kode utama:

$ git submodule add git@github.com:[username]/[repo-name].wiki.git

Cloning into 'hello-team.wiki'...

remote: Counting objects: 6, done.

remote: Compressing objects: 100% (3/3), done.

remote: Total 6 (delta 0), reused 0 (delta 0)

Receiving objects: 100% (6/6), done.

$ git add .

$ git commit -m "added wiki as submodule"

$ git push origin master

Sekarang Wiki akan tampil rapi sebagai submodule dalam proyek kode sumber utama.

**GitHub Hubot**

*Hubot, singkatnya, dapat sangat menambah banyak kesenangan dalam mendokumentasikan dan memberi tahu diskusi tim tentang komitmen penting.*

[**Hubot**](https://github.com/github/hubot) hanyalah sebuah obrolan bot yang dapat mengambil informasi atau memberikan pemberitahuan ketika terhubung ke github commit, masalah atau kegiatan. Dalam sebuah tim yang berusaha mengurangi rapat secara signifikan atau bahkan menghilangkannya sama sekali, Hubot dengan antarmuka obrolan di antara anggota tim membantu mendokumentasikan setiap diskusi. Ini tentu mempromosikan waktu kerja yang fleksibel, karena tidak ada yang hadir pada saat yang sama untuk diskusi berlangsung. Peringatan: Hubot sangat membuat ketagihan!

Dengan ini, mari mulai dengan mengatur Hubot yang dihosting di [**Heroku**](https://www.heroku.com/) dan sebagai bot dengan antarmuka obrolan [**Campfire**](http://campfirenow.com/)! Untuk Heroku dan Campfire, ada versi gratis untuk memulai.

1. Kita akan menggunakan [**versi Github's Campfire dari Hubot**](https://github.com/github/hubot). Jika Anda ingin, Anda dapat memeriksa ad[**aptor untuk obrolan lain**](https://github.com/github/hubot/wiki) seperti Skype, IRC, Gtalk, dll.
2. Buka akun Campfire baru hanya untuk Hubot dan akun ini akan membuat ruang obrolan baru yang akan diundang orang lain nantinya.
3. Terapkan Hubot ke Heroku dengan [**instruksi yang diberikan**](https://github.com/github/hubot/wiki/Deploying-Hubot-onto-Heroku) di Hubot Wiki. Jangan khawatir jika url aplikasi heroku memberikan Cannot GET / karena [**tidak ada yang didapatkan**](https://github.com/github/hubot/issues/286) secara default.
4. Dari akun Campfire Hubot, undang dirimu sendiri. Sekarang, masuk ke akun Campfire Anda sendiri dan jalankan Hubot help. Ini akan memberi Anda semua perintah yang tersedia untuk hubot.
5. Cobalah, seperti hubot ship it atau hubot map me CERN.
6. Selanjutnya, kita akan menambahkan skrip Hubot. Ada [**banyak tersedia**](https://github.com/github/hubot-scripts/tree/master/src/scripts) dengan [**ilustrasi perintah**](https://hubot-script-catalog.herokuapp.com/).
7. Sebagai contoh, kita akan menambahkan skrip [**github-commit**](https://github.com/github/hubot-scripts/blob/master/src/scripts/github-commits.coffee) sehingga setiap kali ada commit, Hubot akan memberi tahu di ruang obrolan. Masukkan file github-commits.coffee di dalam folder scripts.
8. Perbarui file package.json dengan dependensi yang relevan seperti yang diperintahkan di atas setiap file skrip hubot di bawah komentar.
9. Menyebarkan perubahan ke Heroku sekali lagi dengan git push heroku master.
10. Arahkan ke repositori di Github yang pemberitahuan commitnya akan ditampilkan dalam obrolan. Tambahkan di hook web di bawah pengaturan repo. Dalam kasus tulisan "github-commit", webhook akan menjadi [HUBOT\_URL]:[PORT]/hubot/gh-commits?room=[ROOM\_ID]
11. Saat berikutnya repositori memiliki komit, Hubot akan mengobrol dan mengatakannya!

Checkout [**Gitub lain terkait skrip Hubot**](https://github.com/github/hubot-scripts), atau jika Anda ingin menulis satu ada [**tutorial keren tentang itu**](https://net.tutsplus.com/tutorials/javascript-ajax/writing-hubot-plugins-with-coffeescript/) juga! Hubot, singkatnya, dapat sangat menambah banyak kesenangan dalam mendokumentasikan dan memberi tahu diskusi tim tentang komitmen penting, masalah dan kegiatan yang terjadi dengan repositori Github. Cobalah!

Sebagai catatan terakhir untuk bekerja dengan tim di Github, berikut beberapa kiat produktivitas:

1. **Mentions** - Di area teks apa pun, kita dapat menyebutkan pengguna github lain dengan @user dan pengguna akan diberi tahu tentang komentar tersebut
2. **Shortcuts Keys** - Tekan [shift] + ? untuk mengakses tombol pintas Github di halaman mana pun.
3. **Emoji** - Menggunakan [**contekan Emoji**](http://www.emoji-cheat-sheet.com/), Github textareas juga mendukung penyisipan ikon. Pergi ke depan dan bersenang-senang dengan teman tim Anda!