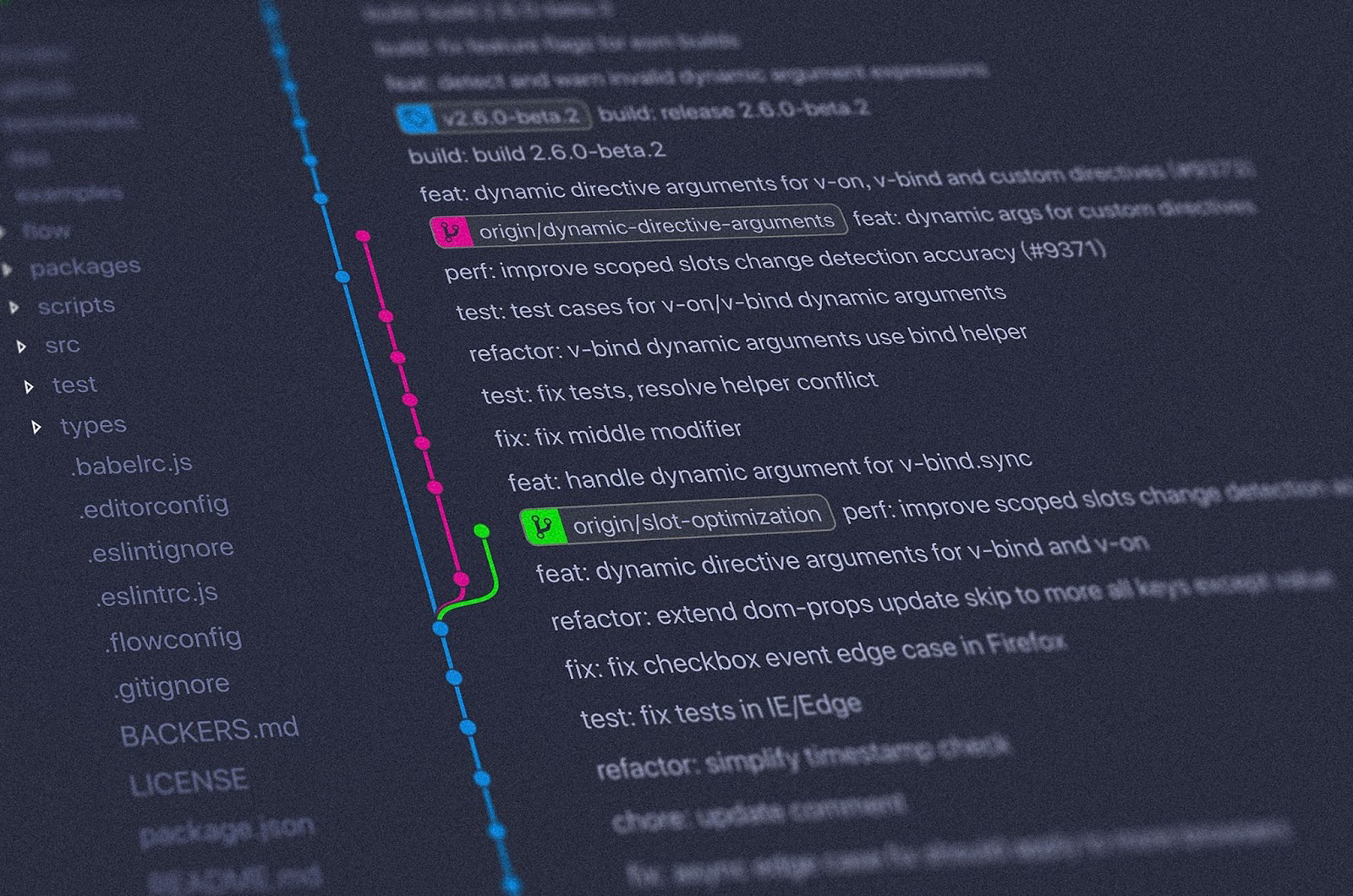
Pengantar Git dan GitHub

Di dalam dunia pengembangan aplikasi, kita akan sering mendengar istilah source code management atau yang biasa dikenal dengan version control system. Apa itu Version Control System? Mengapa sih penting untuk mempelajarinya, terlebih jika kita adalah seorang developer.

Version Control System merupakan tools untuk mengatur suatu perubahan dan konfigurasi dari suatu aplikasi, termasuk juga source code. Setiap perubahan yang dilakukan akan dicatat, sehingga memperjelas siapa yang telah melakukan perubahan tersebut. Selain itu, version control juga berfungsi sebagai backup files. Dengan begitu, akan mempermudah pencarian bug atau masalah.



Version Control System atau yang biasa disingkat dengan VCS, merupakan tools terbaik untuk berkolaborasi antar developer ketika membangun aplikasi. Perusahaan-perusahaan besar di dunia telah menggunakannya untuk mengelola produk atau aplikasi mereka. Contohnya adalah Facebook yang menggunakan Mercurial dan Twitter yang menggunakan Git. Git dan Mercurial ini adalah contoh dari Version Control System.

Lalu bagaimana cara menggunakannya? Kita bisa menggunakan Git secara free atau gratis melalui layanan online yang tersedia seperti GitHub, Bitbucket, dan Gitlab. Namun, salah satu yang cukup populer adalah GitHub. Terlebih ketika ia diakuisisi oleh Microsoft, kepopulerannya semakin menjadi-jadi. GitHub merupakan layanan hosting repository Git berbasis web yang juga memiliki banyak fitur seperti bug tracking dan task management.

Jika Anda merasa awam tentang Git dan GitHub, maka tak perlu khawatir ya. Sebab, kita akan membahas tuntas berbagai hal menarik seputar Version Control System, seperti

Git dan GitHub.

Dasar-dasar Git.

Git Branches.

Berkolaborasi dengan Tim.

GitHub sebagai Portofolio.

**Mengenal Git secara Lebih Dalam**

Pernahkah Anda mengerjakan suatu proyek, baik kecil maupun besar, dengan berkolaborasi bersama tim seperti halnya mengerjakan produk perangkat lunak? Masing-masing anggota tim akan mengerjakan setiap bagian pekerjaan, kemudian menggabungkannya untuk mendapatkan satu tujuan. Namun dalam implementasinya, komunikasi, riwayat pengerjaan, dan distribusi file proyek kerap mengalami kendala.

Pada pengerjaan secara tradisional, setiap anggota tim harus menunggu satu sama lain untuk menyelesaikan satu bagian, kemudian mengirimkannya ke anggota lain untuk melanjutkan pengerjaannya. Tentu itu akan memakan banyak waktu, bukan?

Oleh karena itu, harus ada solusi untuk menyelesaikan permasalahan tersebut.  Teknologi untuk melakukan manajemen tim serta pengerjaan proyek pun harus dihadirkan, salah satunya adalah **Git** yang dapat memberikan kemudahan pada tim untuk melakukan komunikasi saat proses pengerjaan proyek, melihat riwayat perubahan pengerjaan, serta menuntaskan pekerjaan tanpa harus saling menunggu satu sama lain.



Pernahkah Anda mengerjakan tugas penulisan misalnya tugas akhir dan membuat versi revisi ke dalam folder, kemudian memberikan nama revisi 1, revisi 2, dan seterusnya, lalu menyimpannya di komputer lokal atau penyimpanan cloud? Jika iya, pastinya itu akan memakan ruang yang banyak untuk sekadar menyimpan hasil revisi tugas akhir.

Selain itu, jika ada perubahan di setiap revisi, Anda tidak akan dapat melihat riwayat perubahannya. Anda juga tidak akan bisa mengembalikan perubahan yang terjadi pada dua hari yang lalu. Lebih parahnya lagi, Anda tidak akan mendapatkan kembali file yang sudah terhapus ataupun hilang. Tentu saja itu semua akan membuat masalah besar, bukan?



Oleh karena itu, Anda dapat menggunakan Git sebagai solusi untuk menyimpan semua revisi Anda ke dalam penyimpanan yang lebih aman. Anda bisa melihat semua revisi yang telah Anda lakukan, bahkan mengembalikan setiap file yang kemungkinan terhapus. Intinya, riwayat setiap perubahan file akan dapat Anda lihat. Keren, kan? Kemudahan tersebut dapat ditemukan pada teknologi Git. Nah, sekarang muncul pertanyaan, apa sebenarnya Git itu?

Git adalah perangkat lunak yang disediakan secara open source (kode sumber terbuka). Git bertujuan untuk mendukung kolaborasi antar anggota tim serta mengontrol setiap versi perubahan dalam sebuah pekerjaan pengembangan aplikasi ataupun pada bidang lainnya. Git pertama kali dibangun oleh pengembang kernel Linux, yaitu *Linus Torvalds* pada tahun 2005.

Sejarah Git dimulai dari pengembangan kernel Linux yang merupakan proyek perangkat lunak open source dalam lingkup yang sangat besar. Dalam kurun waktu tahun 1991-2002, pengembangan kernel Linux dihabiskan dengan membuat patch dan berkas terarsip untuk melakukan pemeliharaan source code. Pada tahun 2002, proyek kernel Linux mulai menggunakan **Distributed version control systems (DVCS)** yang paten bernama BitKeeper.

Pada tahun 2005, hubungan antara komunitas yang mengembangkan kernel Linux dan perusahaan komersial yang mengembangkan BitKeeper terputus, sehingga komunitas pengembang kernel Linux pun tidak dapat memanfaatkan BitKeeper secara gratis. Hal ini mendesak komunitas pengembangan Linux (khususnya Linus Torvalds sang pencipta Linux) untuk mengembangkan **Version Control System (VCS)** mereka sendiri berdasarkan pengalaman yang telah mereka dapatkan ketika menggunakan BitKeeper. Beberapa kriteria dari sistem baru tersebut adalah sebagai berikut:

1. Kecepatan
2. Rancangan yang sederhana
3. Mendukung alur kerja terdistribusi yang menyerupai BitKeeper
4. Dukungan yang kuat untuk pengembangan non-linier (ribuan cabang paralel)
5. Benar-benar tersebar dan memiliki perlindungan yang sangat kuat terhadap kerusakan data, baik yang tidak disengaja maupun yang berniat buruk
6. Mampu menangani proyek besar seperti Linux secara efisien (kecepatan dan ukuran data)

Kriteria-kriteria tersebut tidak bisa dipenuhi oleh semua sistem version control yang tersedia pada saat itu. Sehingga setelah rilis pengembangan Linux versi 2.6.12-rc2, Linus Torvalds mulai mengembangkan sistem version control sendiri.

Pengembangan Git dimulai pada 3 April 2005, Linus Torvalds mengumumkan proyeknya pada 6 April 2005, kemudian Git mampu melakukan *self-hosting* pada keesokan harinya. Penggabungan (*merge*) beberapa cabang pertama kali dilakukan pada 18 April 2005. Linus Torvalds mencapai sasaran performanya pada 29 April 2005, Git sudah tercatat mengirimkan patch ke root kernel Linux dengan kecepatan 6,7 patch per detik. Pada 16 Juni 2005, Git mengeluarkan rilis kernel Linux versi 2.6.12. Linus Torvalds menyerahkan pemeliharaan Git pada 26 Juli 2005 kepada Junio Hamano, seorang kontributor besar dalam proyek. Hamano bertanggung jawab atas rilis 1.0 pada 21 Desember 2005 dan terus menjadi pemelihara utama proyek Git tersebut.



Sejak kelahirannya pada 2005, Git telah berevolusi dan berkembang sehingga menjadi platform yang mudah digunakan dan memiliki kualitas yang sangat baik. Git sangat cepat dan efisien pada proyek-proyek besar. Git juga memiliki sistem percabangan yang hebat untuk pengembangan non-linear.

Karakteristik kinerja Git sangat kuat jika dibandingkan dengan banyak alternatif version control system lainnya. Semua proses untuk version control, baik perubahan baru, percabangan, penggabungan, maupun perbandingan versi baru dengan sebelumnya telah dioptimalkan.

Ketahuilah! Algoritma yang diimplementasikan di dalam Git memanfaatkan pengetahuan mendalam tentang atribut umum dari tree source code dan bagaimana mereka biasanya dimodifikasi dari waktu ke waktu.

Dengan memanfaatkan Git, kita bisa berkolaborasi dalam tim. Jadi, kita tidak perlu lagi mengerjakan pengembangan sistem ataupun hal lainnya yang berbasis text dengan cara antrean alias menunggu anggota tim pertama selesai mengerjakan kemudian mengirimkan hasil pekerjaan kepada anggota berikutnya untuk melanjutkannya. Tentunya itu akan banyak memakan waktu dan juga tidak efektif.

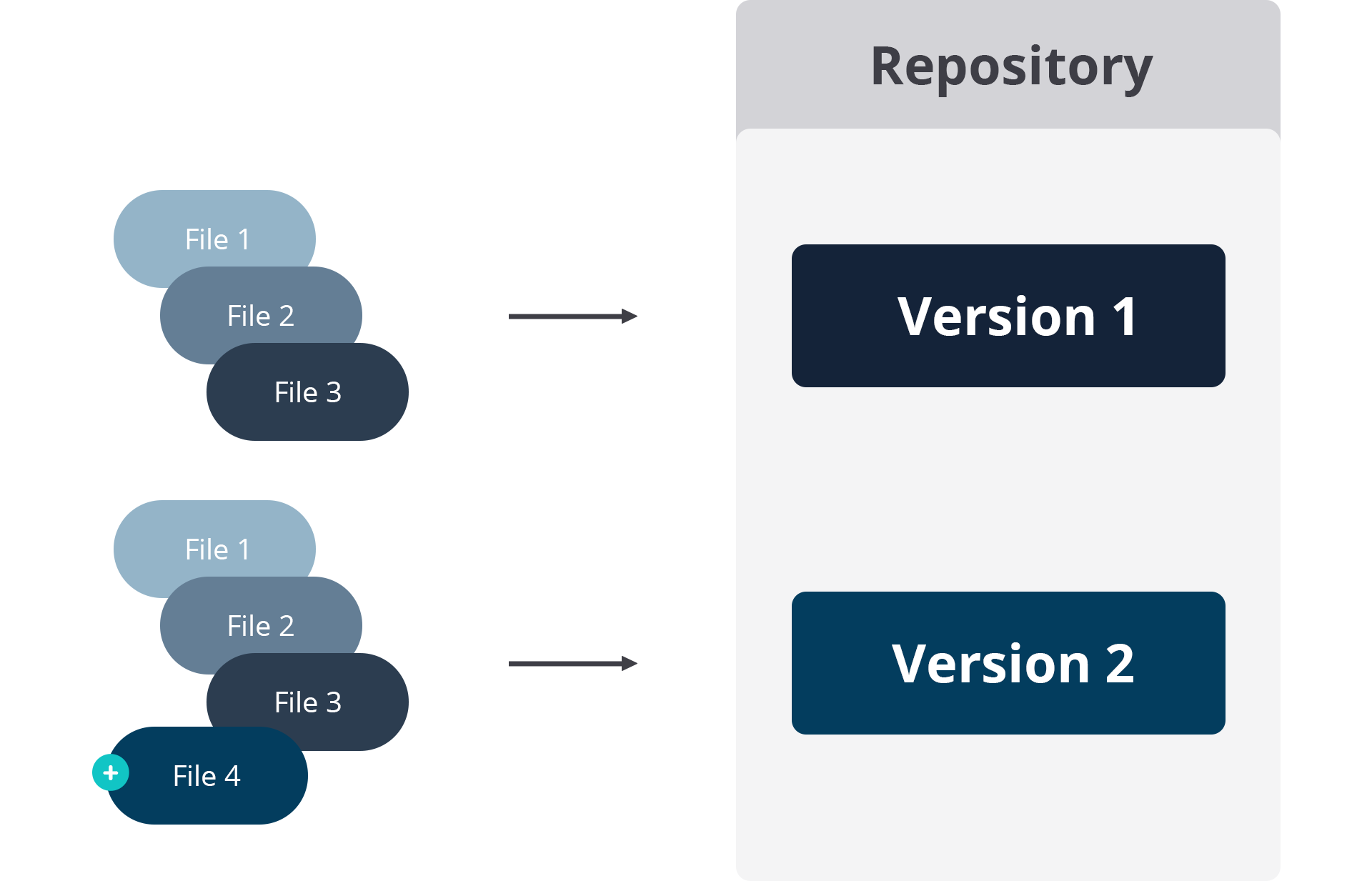
Pada implementasinya, saat menggunakan Git, kita akan dihadapkan dengan berbagai pengaturan awal yang diperlukan untuk membangun sebuah repository Git server sebagai ruang untuk menyimpan proyek yang akan dibangun.

Pada tahap awal mengenal Git, terdapat beberapa istilah yang akan sering Anda temui. Beberapa istilah ini akan lebih diperdalam pada modul selanjutnya. Berikut ini istilah-istilah yang terdapat dalam Git:

**Git Repository**

Git Repository (Repositori Git) merupakan wadah penyimpanan kumpulan file dari berbagai versi proyek yang berbeda. Kumpulan file ini diimpor melalui repository ke server lokal pengguna untuk pembaruan dan modifikasi lebih lanjut dalam konten file.

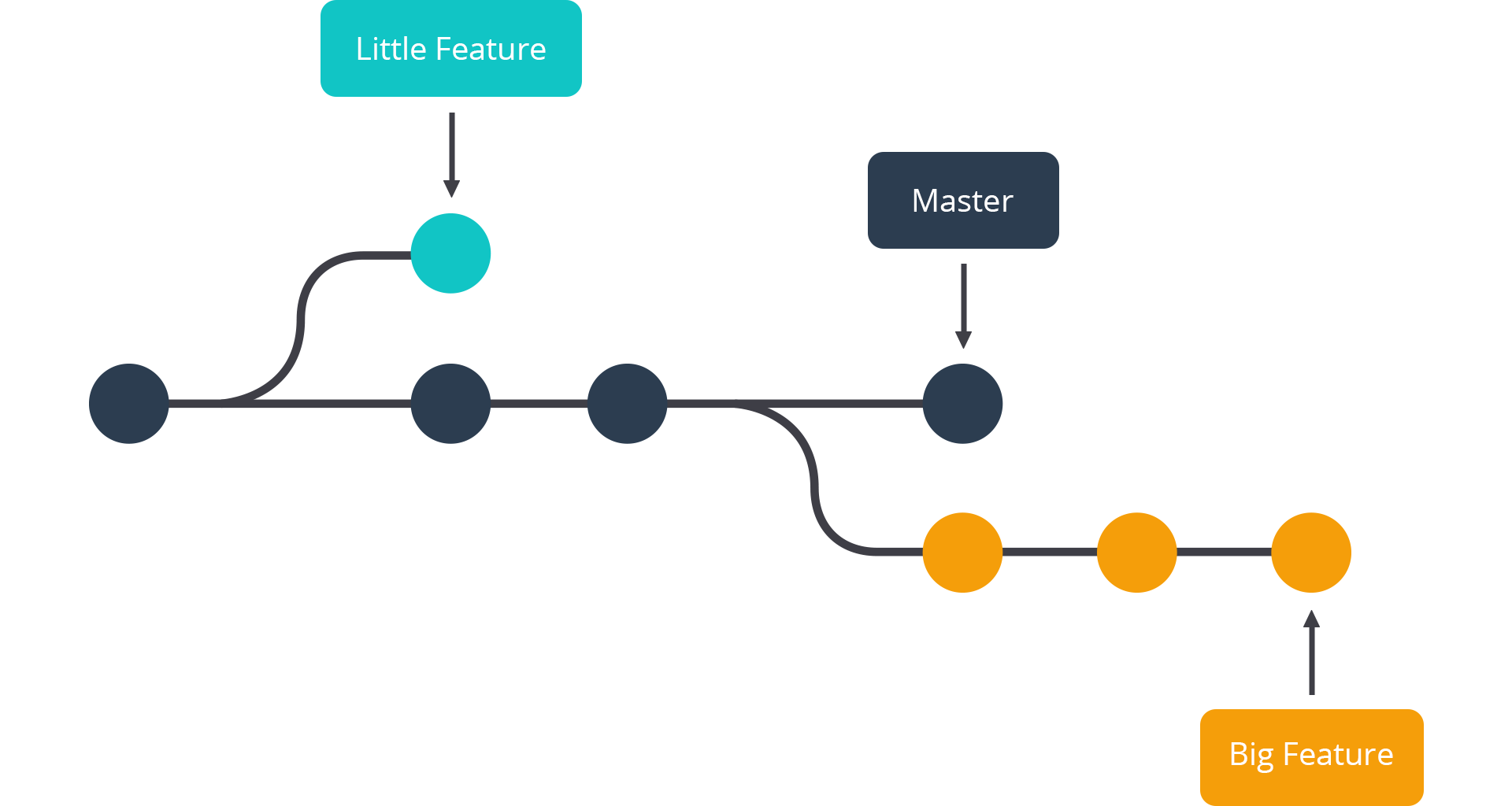
VCS (Version Control System) atau Sistem Kontrol Versi digunakan untuk membuat versi dan menyimpannya di tempat tertentu yang disebut sebagai repository. Proses menyalin konten dari Repositori Git menggunakan bantuan berbagai alat Git disebut sebagai kloning (**clone**). Setelah proses kloning selesai, pengguna mendapatkan repository lengkap di mesin lokalnya. Pada posisi ini, Git akan mengasumsikan bahwa berbagai hal yang akan dikerjakan oleh pengguna terjadi pada repository lokal. Pengguna juga dapat membuat repository baru atau menghapus repository yang ada.



Sebuah Git repository memungkinkan Anda melakukan berbagai operasi di atasnya untuk membuat versi yang berbeda dari proyek. Operasi ini termasuk penambahan file, membuat repository baru, melakukan tindakan, atau menghapus repository. Modifikasi ini akan menghasilkan pembuatan versi proyek yang berbeda.

**Git Branch**

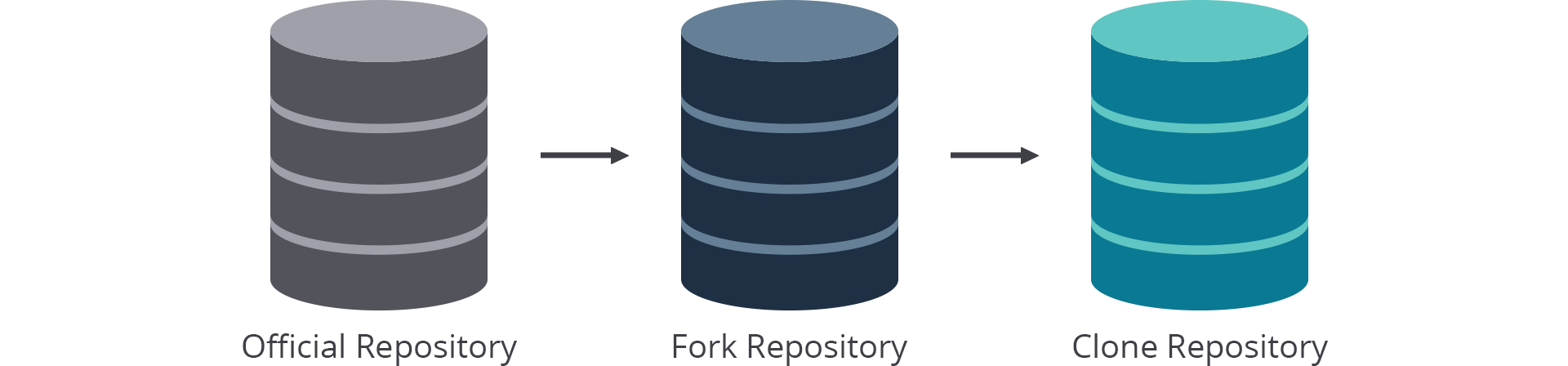
Git Branch merupakan fitur yang dapat digunakan untuk membuat sebuah percabangan untuk versi baru dari proyek Anda. Penggunaan Git Branch ini sangat berguna pada saat Anda melakukan eksperimen dengan proyek Anda. Dengan begitu, Anda tidak akan mengubah apa pun di dalam proyek utama.



Pada Git branch terdapat cabang utama (master) sebagai induk dari pekerjaan. Selain itu, terdapat juga percabangan baru yang dapat Anda visualisasikan sebagai penambahan fitur, ada fitur kecil dan fitur besar.

**Git Fork**

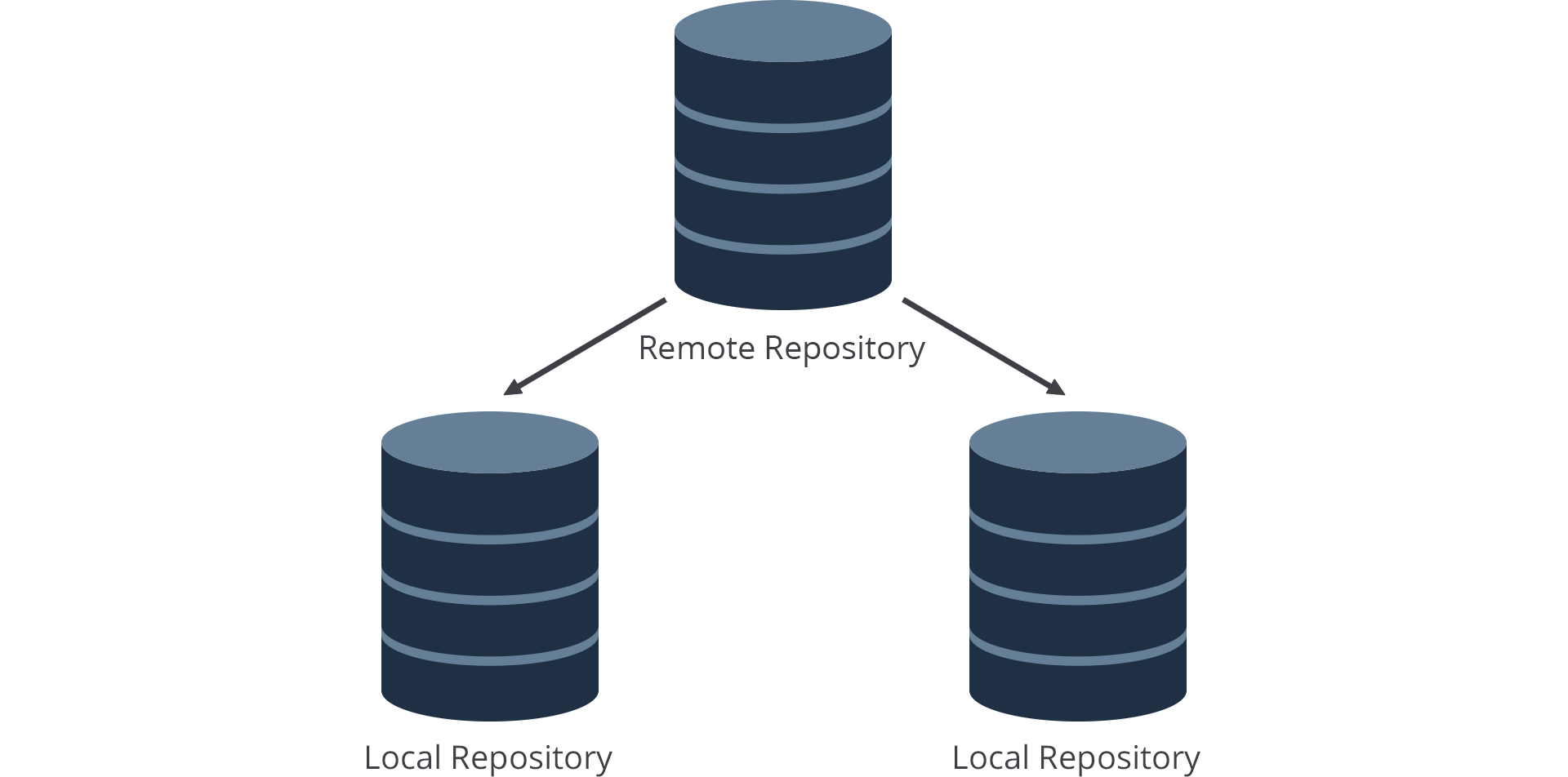
Fork merupakan salinan dari sebuah repository orang lain yang tersimpan di dalam sebuah repository Git. Forking repository memungkinkan Anda untuk bereksperimen dengan bebas dan membuat perubahan tanpa memengaruhi repository utamanya.



Alur kerja fork dimulai dengan menemukan repository publik asli yang tersimpan dalam server. Namun ketika pengembang ingin memulai mengerjakan proyek, mereka tidak langsung mengkloning repository resmi. Sebagai gantinya, mereka mengambil repository resmi untuk membuat salinannya di server. Salinan baru ini bersifat sebagai repository publik pribadi mereka. Tidak ada pengembang lain yang diizinkan untuk mengirim perubahan pada repository tersebut. Namun, mereka dapat mendapatkan perubahan dari repository tersebut. Selain bersifat publik, kita juga dapat menyimpan salinan repository menjadi private. Sehingga pengembang lain tidak akan bisa melihat perubahan yang telah kita lakukan.

**Git Clone**

Git Clone hampir sama dengan Git Fork, keduanya sama-sama melakukan salinan pada repository. Yang berbeda adalah saat melakukan git fork, repository akan disimpan berada di dalam server. Sedangkan untuk git clone, hasil salinannya disimpan dalam penyimpanan lokal. Perintah Git Clone adalah cara yang paling umum bagi pengguna untuk mendapatkan salinan tulisan. Melakukan Clone hanya dilakukan satu kali, yaitu saat akan mengerjakan satu proyek yang telah tersimpan di dalam Git repository. Kemudian pengembang menyalin proyek yang tersimpan di dalam repository untuk diletakkan di penyimpanan lokal. Setelah pengembang mendapatkan salinan pekerjaan, semua operasi dan kolaborasi kontrol versi dikelola melalui repository lokal.

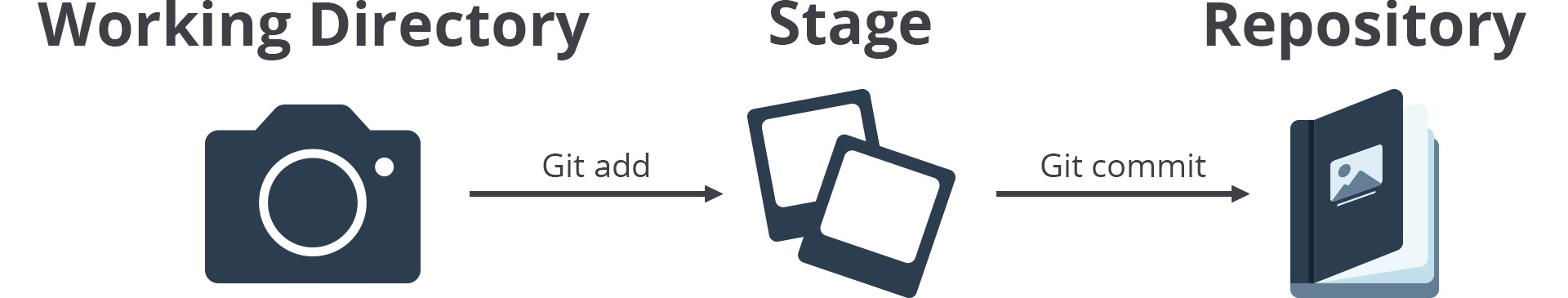


Penggunaan clone paling umum digunakan untuk mengambil satu repository yang ada dan membuat clone dari repository tersebut ke direktori baru di lokasi lain. Repository asli dapat ditemukan di sistem file lokal atau pada protokol yang didukung untuk dapat diakses melalui jarak jauh.

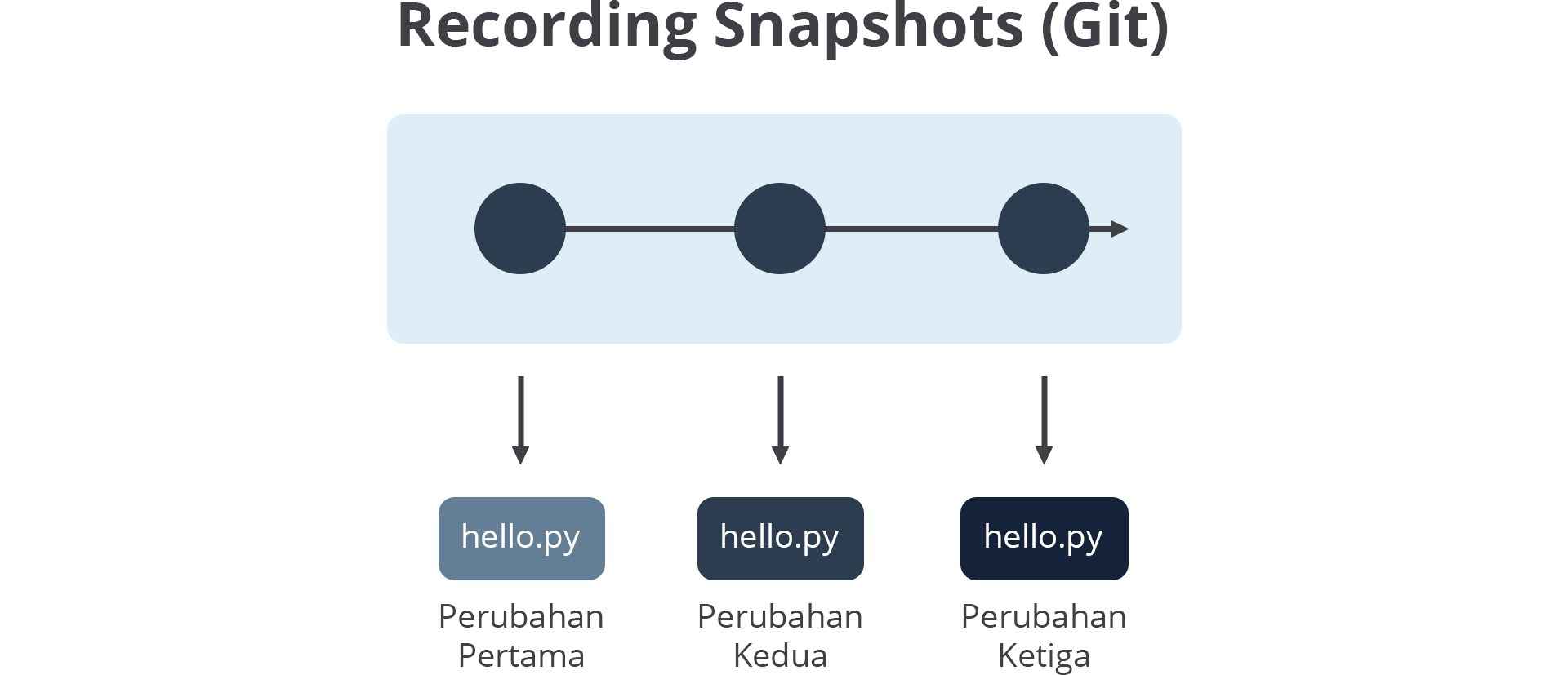
Untuk memudahkan, cloning secara otomatis membuat koneksi jarak jauh (*remote*) yang disebut dengan “**origin**” yang mengacu dari repositori lokal ke repository asli (di server). Ini membuatnya sangat mudah untuk berinteraksi dengan repository pusat.

**Git Commit**

Commit merupakan cuplikan perubahan (snapshot) dari repository Anda pada waktu tertentu. Dengan begitu, setiap pekerjaan yang telah selesai, dapat disimpan ke dalam repository Anda.



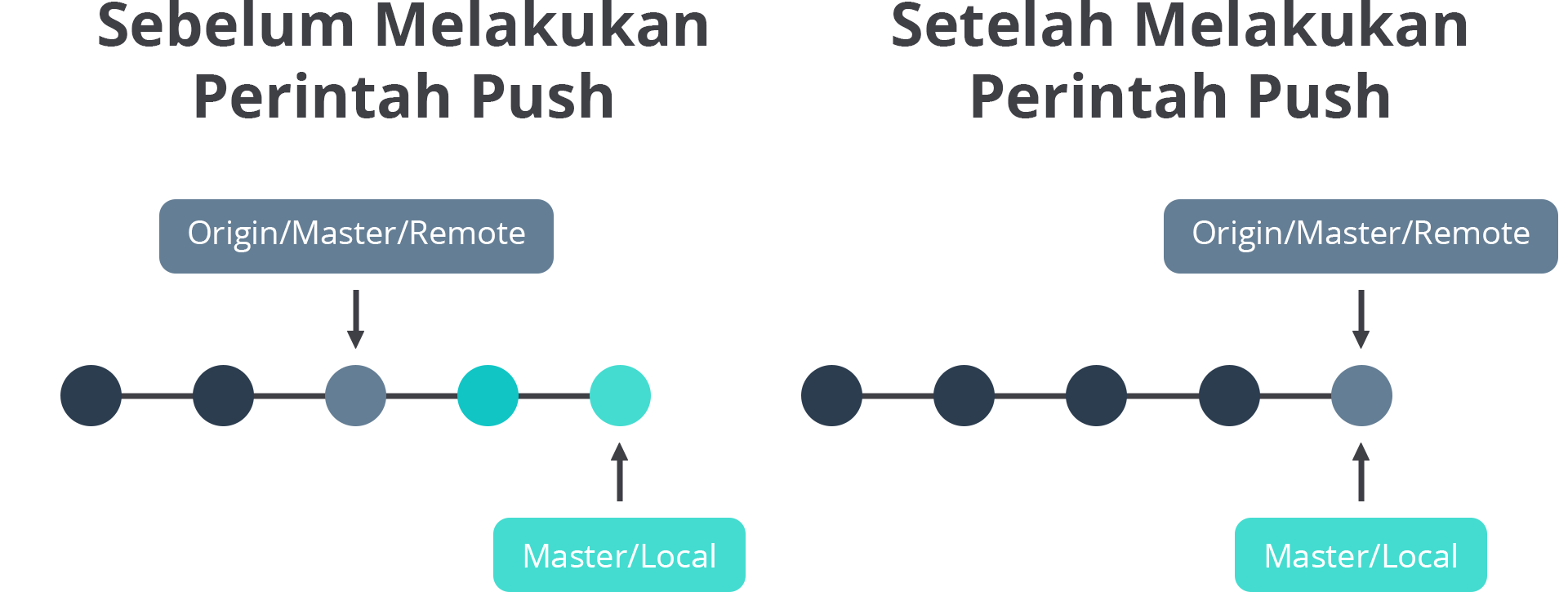
Ketika Anda menggunakan antarmuka GitHub, Anda dapat secara langsung menggunakan perintah git commit untuk menyimpan setiap perubahan yang dilakukan. Namun, ketika Anda menggunakan penyimpanan lokal, Anda harus mengoperasikan [Git add](https://www.atlassian.com/git/tutorials/saving-changes) sebelum memanggil git commit. Git add berguna untuk menambahkan file yang diubah ke dalam daftar antrean perubahan. Kedua perintah ini, baik Git commit maupun Git add, merupakan dua perintah yang sering digunakan saat Anda menggunakan Git dengan penyimpanan lokal.



Setiap *commit*yang dilakukan dapat menggambarkan aktivitas apa saja yang terjadi selama proyek dikerjakan. Setiap tim dapat melakukan commit terhadap file dalam satu proyek secara bersamaan tanpa harus menunggu anggota tim lain melakukannya.

**Git Push**

Perintah *Git push*merupakan tindakan lanjutan dari perintah *Git commit*. *Git push*mengunggah semua commit cabang (branch commit) pada repository lokal ke repository server. Secara default, *Git push* hanya memperbarui cabang yang sesuai pada remote. Sehingga jika Anda sedang berada di branch master atau melakukan check out ke master dengan perintah Git push, hanya branch master-lah yang akan diperbarui.



**Penggunaan Git push**

Setelah membuat dan melakukan perubahan secara lokal, Anda dapat membagikannya dengan repository di server menggunakan Git push. Mendorong perubahan ke server akan membuat commit Anda dapat diakses oleh orang lain yang mungkin berkolaborasi dengan Anda. Ini juga akan memperbarui permintaan dengan cabang yang sedang Anda kerjakan.

Ketika Anda menggunakan antarmuka GitHub, setiap commit yang dilakukan akan tersimpan secara langsung di repository. Sehingga Anda tidak perlu menjalankan perintah git push dan git pull.

Namun, ketika Anda menggunakan penyimpanan lokal, penting untuk menjalankan perintah [Git pull](https://github.com/git-guides/git-pull) sebelum meng-upload (git push) perubahan baru ke server. Dengan melakukan git pull, repository lokal akan diperbarui sesuai dengan repository remote saat ini. Sehingga ketika ada kontributor lain yang melakukan perubahan pada repository server, repository lokal ikut diperbarui.

**Rangkuman Git dan Github**

Bagaimana, apakah pembelajaran yang kami jabarkan dapat dipahami dengan baik? Pada modul ini kita telah mempelajari bagaimana sejarah serta penggunaan Git dan GitHub. Oleh karena itu, mari kita ulas secara singkat apa yang telah kita pelajari.

* Pada materi Git kita mempelajari sejarah berdirinya platform Git, bagaimana **Linus Torvalds** bersama tim membangun platform tersebut pada tahun 2005. Git dibangun dengan tujuan agar dapat melakukan kolaborasi antar tim serta bagaimana proyek dapat dikontrol dengan baik. Untuk bisa berkolaborasi, bisa dengan membangun proyek menjadi open source atau pilihan lainnya menyetel kolaborasi dengan orang lain.
* Kemudian, kita juga telah belajar bagaimana dapat memahami perintah dan istilah yang ada pada Git, seperti **Git Repository**, **Git Branch**, **Git Fork**, **Git Clone**, **Git Commit**, dan **Git Push**.

| **Perintah** | **Fungsi** |
| --- | --- |
| Git Repository | Media penyimpanan file proyek di dalam Git server. |
| Git Branch | Percabangan untuk versi baru dari proyek repository. |
| Git Fork | Penyalinan dari repository orang/organisasi lain, lalu menyimpannya di dalam repository Git sendiri. |
| Git Clone | Mengambil sebuah repository dan menyimpannya pada direktori lokal. |
| Git Commit | Cuplikan perubahan (snapshot) dari repository Anda pada waktu tertentu. |
| Git Push | Mengirim hasil dari perubahan file yang dilakukan ke dalam repository server. |
| Git Pull | Menarik atau mengambil *source code* dari repository server ke lokal.. |

* Selanjutnya, pada materi GitHub kita telah mempelajari bagaimana menggunakan **Platform Website GitHub**. GitHub merupakan sebuah perusahaan layanan hosting repository Git berbasis **Cloud.**Dengan menggunakan GitHub, berkolaborasi antar anggota tim menjadi lebih mudah. GitHub adalah *tools* yang sangat populer karena mudah digunakan.
* Terdapat menu utama pada saat mengakses halaman GitHub yaitu, **Explore Repository**, **Sign up**, dan **Sign in**. Selain itu, ada juga fitur-fitur pendukung pada GitHub seperti:
  + **New Repository**, digunakan membuat repository baru dengan jenis *public* & *private*.
  + **Import Repository**, digunakan untuk mengambil/menyalin file proyek dari vendor lain seperti Subversion atau Mercurial ke dalam repository baru pada GitHub. Proses impor dilakukan dengan menggunakan *link URL*.
  + **Gist**, digunakan untuk berbagi potongan kode, catatan, melakukan daftar, dan lainnya. Seperti halnya repositori, Anda dapat menyetel dan menyimpan Intisari Anda sebagai rahasia atau publik.
  + **GitHub Organization**, digunakan untuk berkolaborasi bersama pada suatu organisasi misalnya untuk bisnis dengan sejumlah kelebihan misalnya banyak proyek sekaligus. Pemilik dan administrator dapat mengelola hak akses anggota ke suatu data atau proyek dengan fitur keamanan dan administratif yang canggih.
  + **GitHub Project**,digunakan untuk mengatur dan memprioritaskan pekerjaan. Anda dapat membuat papan proyek (*project board*) untuk mengerjakan fitur tertentu, peta alur yang komprehensif (*comprehensive roadmaps*), atau bahkan merilis daftar periksa (*release checklists*). Dengan papan proyek, Anda memiliki fleksibilitas untuk membuat alur kerja khusus yang sesuai dengan kebutuhan Anda.
* Selain itu, terdapat beberapa latihan yang telah kita lakukan dalam materi Git dan GitHub. Berikut di antaranya:
  + **Latihan Membuat Akun GitHub**  
    Berikut beberapa tahapan yang bisa Anda lakukan untuk membuat akun GitHub:
    - Mengakses alamat[https://github.com/signup](https://github.com/signup" \t "_blank).
    - Melengkapi data-data yang diperlukan.
    - Melakukan verifikasi pendaftaran menggunakan kode unik yang dikirimkan GitHub melalui email.
    - Menambahkan dan memverifikasi email cadangan untuk mengelola akun GitHub (opsional).
  + **Latihan Mengeksplorasi Proyek GitHub**  
    Berikut beberapa tahapan yang bisa Anda lakukan untuk mengeksplorasi proyek GitHub:
    - Melakukan pencarian proyek GitHub dengan menggunakan fitur Search. Pencarian proyek GitHub bertujuan untuk mendapatkan referensi repository saat mengerjakan proyek sendiri.
    - Menganalisis hasil pencarian tersebut seperti, jumlah pengunjung yang memonitor (*watch*), memberikan bintang (*star)*, dan menyalin proyek repository ke repository pribadi (fork).
    - Memberikan bintang hasil pencarian proyek repository agar dapat mengikuti perkembangannya. GitHub memiliki fitur stars (bintang) yang memudahkan Anda untuk menemukan repository atau topik yang Anda cari sebelumnya.
    - Melihat kontributor dan kolaborator dari sebuah proyek repository. Setiap proyek repository memiliki kolaborator dan kontributor.
  + **Latihan Mengenal GitHub Dashboard**Berikut beberapa tahapan yang bisa Anda lakukan untuk mengenal GitHub Dashboard:
    - Melihat semua aktifitas yang dilakukan pada halaman GitHub dashboard, seperti mengikuti proyek repository, melakukan perubahan pada proyek GitHub, atau memberikan komentar/tanggapan di dalam repository.
    - Melihat bagian toolbar pada halaman GitHub dashboard. Dalam toolbar terdapat beberapa menu seperti:
      * **Pull request** untuk melihat semua aktivitas mengenai pekerjaan yang telah selesai dan tersimpan di dalam GitHub.
      * **Issues** untuk melihat semua tanggapan atau komentar mengenai permasalahan yang kita dapat saat menggunakan repository milik orang lain.
      * **Marketplace** untuk mencari dan memasang fitur *tools* pendukung yang disediakan oleh GitHub. Tools dalam marketplace akan membantu Anda dalam mengoptimalkan penggunaan GitHub di dalam proyek.
      * **Explore** untuk melihat berbagai informasi atau berita terkini mengenai repository, topik, dan *trending* di GitHub.
      * **Settings** untuk melakukan perubahan profil seperti username, foto, serta informasi lainnya.
* Setiap proyek repository memiliki kolaborator dan kontributor. Seorang **kolaborator** merupakan anggota tim yang memiliki akses dalam sebuah repositori proyek (*project repository*). Akses ini tentunya bermacam-macam ya, ada yang dibatasi pada pekerjaan-pekerjaan tertentu, ada pula yang dapat melakukan semua pekerjaan dalam sebuah repository.
* Ketika Anda membuat sebuah repository menggunakan akun personal, maka secara otomatis akan menjadi seseorang owner repository dan memiliki akses penuh terhadap repository tersebut. Di sisi lain, ketika Anda mengundang orang lain menjadi kolaborator dalam repository tersebut, maka mereka akan dapat melakukan *pull*(*read*) dan *push*(*write*) pada repository tersebut.
* Namun, jika Anda menggunakan akun organisasi, GitHub menyediakan 5 tingkatan level (role) yang berbeda untuk seorang kolaborator, yakni read, triage, write, maintain, dan admin.
  + Seorang kolaborator dengan tingkatan **read**, biasanya hanya memiliki akses untuk membuka repository. Level ini direkomendasikan untuk orang non-coding (tim lain) yang ingin melihat atau berdiskusi mengenai proyek repository.
  + Seorang kolaborator dengan tingkatan **write**, biasanya digunakan oleh seorang developer atau pengembang untuk membuka, melakukan clone, dan melakukan push pada repository tersebut. Ia juga dapat mengelola issues dan pull request yang masuk.
  + Seorang kolabolator dengan tingkatan **triage**, biasanya digunakan oleh maintainer yang hanya dapat mengeloa issues dan pull request yang ada masuk ke dalam repository.
  + Seorang kolaborator dengan tingkatan **maintain**, biasanya digunakan oleh seorang project manager yang dapat mengelola beberapa pengaturan dalam repository. Ia juga dapat melakukan berbagai hal seseorang dengan tingkatan read.
  + Seorang kolaborator dengan tingkatan **admin**, biasanya digunakan oleh seseorang yang butuh mengakses secara penuh kepada repository. Ia dapat melakukan berbagai hal, termasuk mengundang kolaborator lain untuk bergabung dalam repository.
* Seorang kolaborator juga dapat berperan sebagai kontributor. Apa itu kontributor? **Kontributor** adalah siapa saja yang dapat melakukan commit atau menyimpan cuplikan perubahan ke dalam repository (*git push*). Akan tetapi, tidak semua kontributor dapat melakukan *commit*secara langsung. Ini disebabkan karena mereka tidak memiliki akses untuk melakukan push ke dalam repository tersebut (atau tidak berperan sebagai kolaborator). Sehingga mereka perlu melakukan *pull request* untuk melakukan commit pada repository orang lain.
* Di dalam GitHub terdapat sistem notifikasi untuk memberikan pembaruan informasi kepada Anda. Secara umum, Anda akan otomatis berlangganan pemberitahuan bila Anda memiliki:
  + Fitur **Watch** pada repository atau tim yang Anda ikuti dalam keadaan aktif
  + Menerima tanggapan **Pull request** atau membuat **issues**.
  + Melakukan **Pull request**, **issue**, atau membuat percakapan diskusi bersama tim.
  + Berlangganan secara manual dengan mengeklik **Watch** atau **Subscribe**.
  + Saat teman atau orang lain melakukan **@mentioned** terhadap nama GitHub Anda.
  + Mengubah status, seperti dengan menutup masalah (**issues**) atau melakukan penggabungan pull request.
  + Anda menjadi anggota tim dan mendapatkan **@mentioned** oleh satu tim Anda.
* Setiap notifikasi yang masuk akan terlihat pada[Inbox Notifications](https://github.com/notifications). Anda dapat menghentikan notifikasi dengan cara melakukan **Unsubscribe**pada menu **Manage Notification**.
* Selain mendapatkan notifikasi masuk langsung di dalam GitHub Dashboard, Anda juga akan mendapatkan notifikasi melalui email yang Anda gunakan sebagai akun GitHub. Pengaturan notifikasi email dapat dilakukan pada menu[Setting Notifications](https://github.com/settings/notifications).

Dengan mengikuti pembelajaran Git dan GitHub ini, Anda diharapkan dapat mencoba mengimplementasikan semua materi ke dalam proyek yang sesungguhnya, serta dapat berkolaborasi bersama tim dengan menggunakan GitHub. Semangat!