Kelas GIT

Nova Adi Saputra

# **BAB I**

Pengendali Versi dan GIT

this was the right turn that sent me down a much more interesting path than the one I was on.

~ Ben Straub ~

## Pengendali Versi

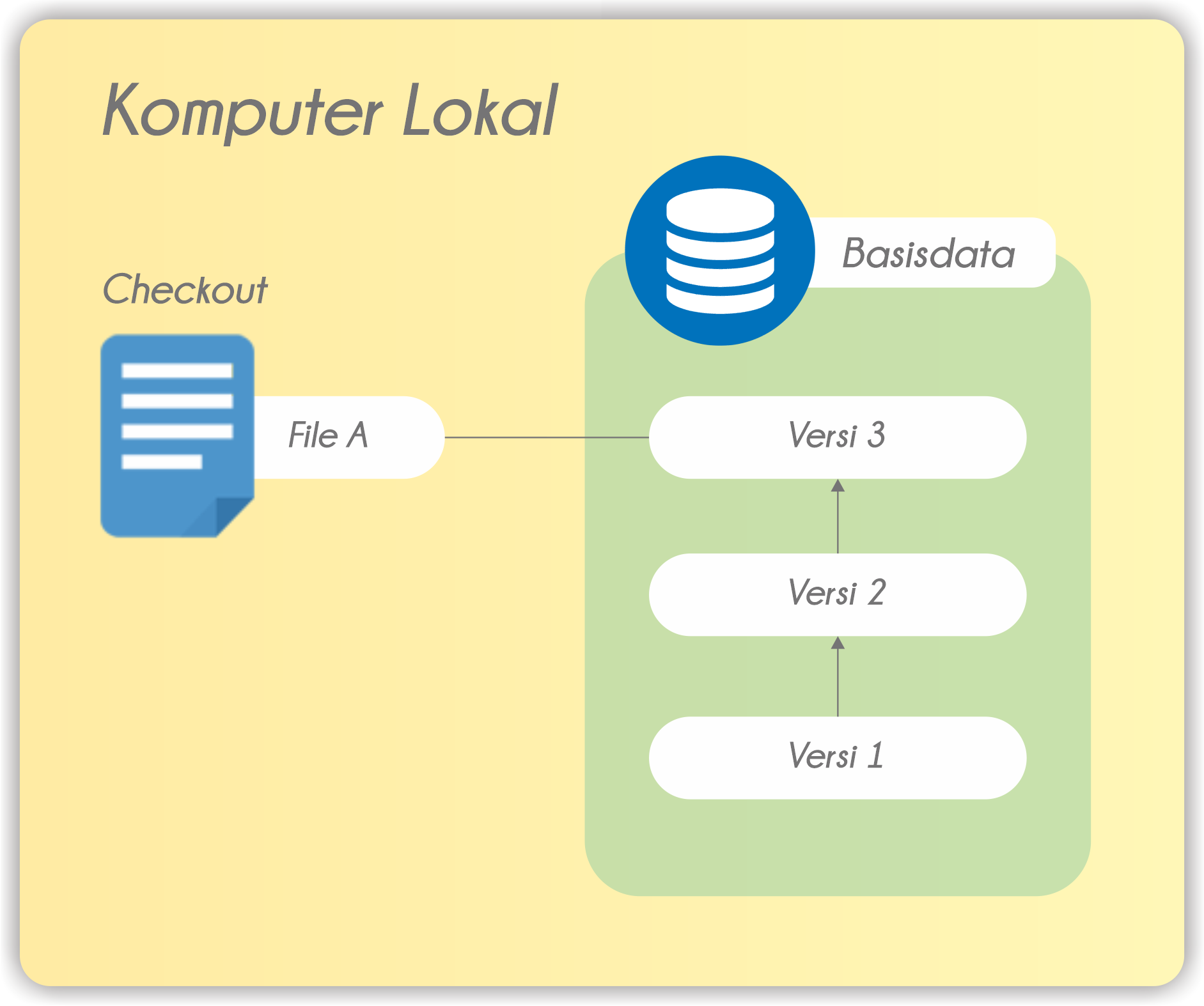
*version control* atau *Pengendali Versi* adalah suatu sistem yang merekam perubahan pada sebuah file atau kumpulan file dari waktu ke waktu sehingga kita dapat kembali ke waktu tertentu. Developer program sering menggunakan Pengendali Versi karena dirasa efisien untuk merekam perubahan pada sourcecode dan memudahkan developer untuk menandai versi perubahannya, Memudahkan editor grafis untuk mengembalikan perubahan setiap langkah dalam pengeditannya.

*Sistem Pengendali Versi (SPV)* atau *Version Control System (VCS)* adalah hal yang patut untuk dicoba. VCS dapat mengembalikan file yang diinginkan kembali ke keadaan sebelumnya, mengembalikan seluruh file atau bahkan sebuah proyek ke keadaan sebelumnya, membandingkan perubahan yang terjadi dari waktu ke waktu, memeriksa siapa yang telah melakukan modifikasi yang dapat menyebabkan sebuah permasalahan terjadi, siapa yang mengidentifikasi masalah dan kapan, dan masih banyak lagi. Menggunakan VCS dapat diartikan bahwa jika kita mengalami permasalahan atau kehilangan suatu file, kita dapat dengan mudah menangani atau mengembalikannya. Selain itu, kita dapat mendapatkan semua keuntungan tersebut dengan mudah.

### Sistem Pengendali Versi Lokal

Banyak orang memilih metode Pengendali Versi dengan cara menyalin file ke direktori lain dengan cap waktu berbeda. Metode tersebut sangat umum karena mudah untuk diimplementasikan, tetapi memiliki banyak kekurangan dan potensi kerusakan yang besar. Dengan menggunakan metode tersebut nantinya akan mudah melupakan direktori mana yang sedang dikerjakan dan terkadang secara tidak sengaja mengedit file didirektori yang salah atau bahkan menyalin file yang tidak sesuai. Untuk menangani kasus seperti ini, dahulu Programmers mengembangkan SPV local yang menggunakan basisdata sederhana untuk menyimpan segala perubahan yang terjadi pada file menggunakan Pengendali Revisi.

Salah satu dari alat SPV terkemuka adalah sebuah system yang disebut RCS (Revisions Control System) atau Sistem Pengendali Revisi (SPR), yang mana pada saat ini masih didistribusikan dengan banyak perangkat computer. SPR bekerja dengan menyimpan kumpulan bagian kecil perubahan (patch) pada format tertentu didalam penyimpanan, patch tersebut kemudia dapat dibuat ulang dengan sebuah file pada suatu waktu tertentu dengan menambahkan bagian-bagian yang diubah.

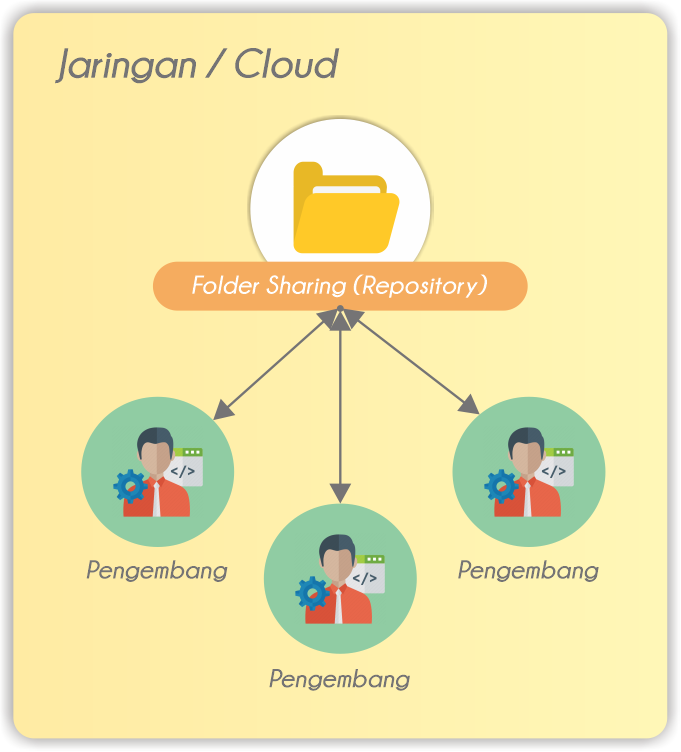


### Sistem Pengendali Versi Terpusat

Masalah besar berikutnya adalah dimana kita akan membuat proyek yang dikembangkan oleh banyak orang dengan system yang berbeda. Untuk menangani masalah ini, dikembangkanlah *SPV Terpusat*. Sistem ini memiliki sebuah server yang didalamnya termuat seluruh file yang di versikan, dan jumlah dari klien yang mengambil atau mengedit file dari tempat pusat tersebut. Selama bertahun-tahun SPV ini menjadi standar Pengendali Versi.

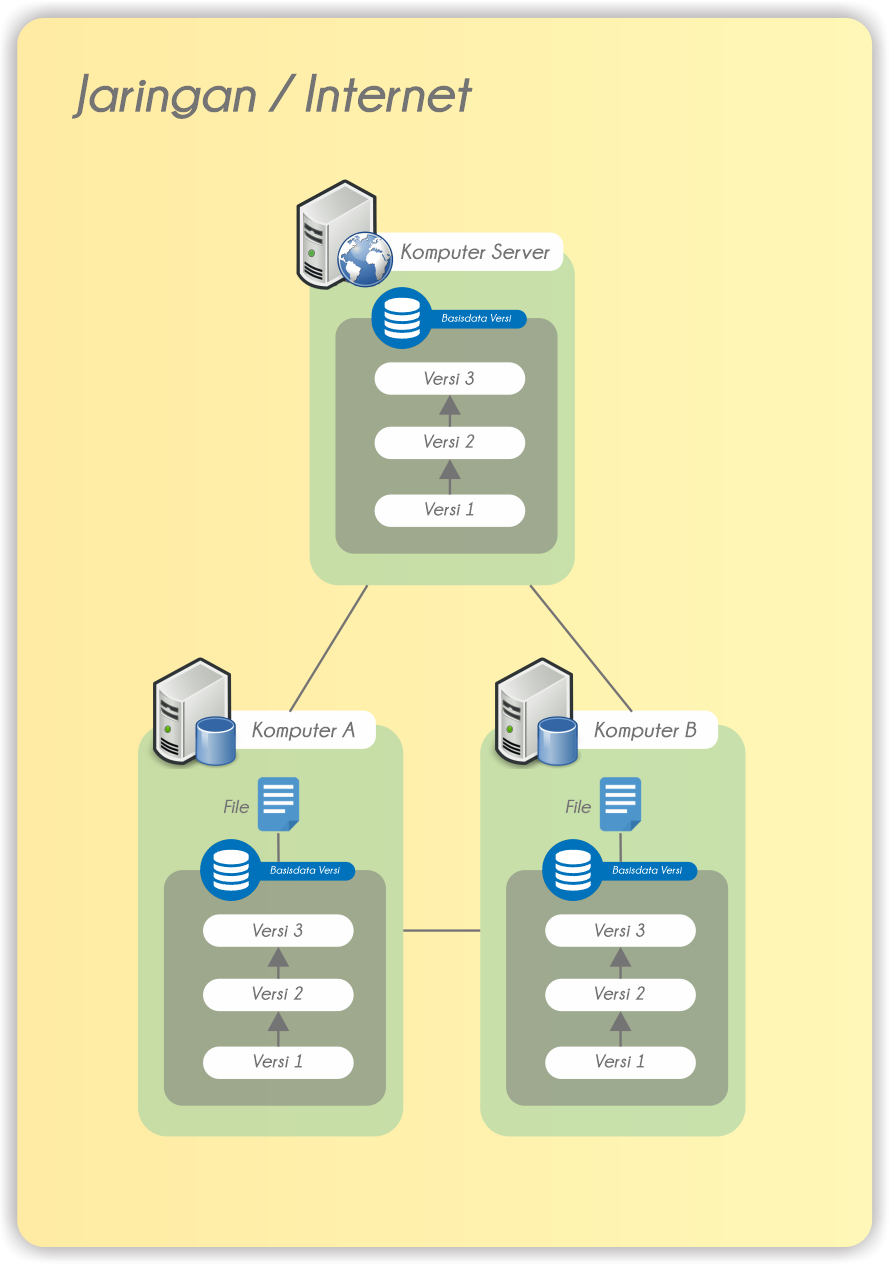
Sistem ini menawarkan banyak keuntungan, terutama dibandingkan dengan SPV lokal. Sebagai contoh, semua orang mengetahui pada tingkatan mana orang lain sedang lakukan dalam proyek tersebut. Administrator memiliki akses yang baik untuk mengendalikan siapa dan apa yang dapat dilakukan, dan memanajemen SPV Terpusat jauh lebih mudah dibandingkan memanajemen SPV dengan basisdata local pada setiap klien.

Namun, system ini juga memiliki beberapa kelemahan yang serius. Masalah yang paling jelas terjadi pada SPV Terpusat adalah Ketika server mengalami kerusakan dan dimatikan selama satu jam katakanlah, maka pada saat itu tidak ada yang dapat mengerjakan apapun dan tidak ada yang dapat menyimpan perubahan versi pada apapaun yang sedang dikerjakan. Jika penyimpanan pada basisdata sentral mengalami kerusakan atau bahkan *corrupted*, dan datanya belum sempat di backup, maka anda akan kehilangan segalanya. Seluruh jejak proyek akan hilang, kecuali apapun yang telah tersimpan pada penyimpanan local setiap orang yang berkolaborasi. SPV local juga memiliki masalah yang sama, Ketika anda hanya memiliki jejak pada satu tempat saja, maka resikonya anda akan kehilangan segalanya.



### Sistem Pengendali Versi Terdistribusi

Di sinilah Sistem Pengendali Versi Terdistribusi (SPV Terdistribusi) hadir. Dalam SPV Terdistribusi (seperti Git, Mercurial, Bazaar atau Darcs), klien tidak hanya dapat melihat snapshot terbaru dari suatu file; akan tetapi, mereka sepenuhnya menyalin repositorinya, termasuk riwayat lengkapnya. Jadi, jika ada server yang mengalami kerusakan, dan sistem ini berkolaborasi melalui server tersebut, setiap repositori yang telah disalin oleh klien dapat disalin kembali ke server untuk mengembalikannya. Setiap salinan benar-benar merupakan cadangan penuh dari semua data yang ada.



Selain itu, sistem ini mendukunf dengan cukup baik untuk memiliki beberapa *remote* repositori yang dapat saling bekerja sama, sehingga Anda dapat berkolaborasi dengan tim yang berbeda dengan cara yang berbeda dalam waktu yang bersamaan di proyek yang sama. Ini memungkinkan Anda untuk memanajemen beberapa jenis alur kerja yang tidak dapat dilakukan dalam SPV terpusat, seperti model yang *hierarkis*.

## Sejarah Singkat Git

Seperti kebanyakan hal hebal yang ada dalam hidup, Git dimulai dari kesederhanaan kreatifitas dan berbagai macam kontroversi. *Kernel Linux* adalah proyek perangkat lunak sumber terbuka dengan cakupan yang cukup besar. Pada tahun-tahun awal pengembangan kernel Linux (1991–2002), perubahan pada perangkat lunak diedarkan menggunakan *patch* dan file yang diarsipkan. Pada tahun 2002, proyek kernel Linux mulai menggunakan SPV Terdistribusi komersial yang disebut *BitKeeper*.

Pada tahun 2005, hubungan antara komunitas yang mengembangkan kernel Linux dan perusahaan komersial yang mengembangkan BitKeeper mengalami permasalahan, dan status alat bebas biaya telah dicabut. Hal ini mendorong komunitas pengembangan Linux (dan khususnya Linus Torvalds, pencipta Linux) untuk mengembangkan alat mereka sendiri berdasarkan beberapa pelajaran yang dipelajari saat menggunakan BitKeeper. Beberapa tujuan dari sistem baru tersebut adalah sebagai berikut:

* Kecepatan
* Desain yang sederhana
* Dukungan kuat untuk pengembangan non-linier (ribuan cabang paralel)
* Terdistribusi penuh
* Mampu menangani proyek besar seperti kernel Linux secara efisien (kecepatan dan ukuran data)

Sejak lahir pada tahun 2005, Git telah berkembang dan matang agar mudah digunakan namun tetap mempertahankan kualitas bawaannya. Sistem Ini memiliki kecepatan luar biasa, sangat efisien dengan proyek-proyek besar, dan memiliki sistem percabangan yang luar biasa untuk pengembangan non-linier (Git Branching).

### Apa itu Git ?

Jadi, apa itu Git secara singkat? Ini adalah bagian yang penting untuk dipahami, karena jika anda mengerti apa Git dan dasar-dasar cara kerjanya, kemudian menggunakan Git secara efektif mungkin akan jauh lebih mudah untukmu. Saat Anda mempelajari Git, cobalah untuk menjernihkan pikiran Anda dari hal-hal yang mungkin Anda ketahui tentang VCS lain, seperti seperti CVS, Subversion atau Perforce — melakukannya akan membantu Anda menghindari kebingungan halus saat menggunakan alat ini.

Meskipun antarmuka pengguna Git cukup mirip dengan VCS lain ini, Git menyimpan dan memikirkannya

informasi dengan cara yang sangat berbeda, dan memahami perbedaan ini akan membantu Anda menghindari

menjadi bingung saat menggunakannya.

## Instalasi Git

## Konfigurasi Git