GIT

Gambaran Umum

Materi ini akan mempelajari penggunaan version control Git sebagai sarana untuk berkolaborasi dalam membangun sebuat produk aplikasi. Selain itu, juga mempelajari Gitflow sebagai kerangka kerja yang umum digunakan di dunia industri dalam membangun aplikasi.

BINAR

Tujuan Pembelajaran

- Siswa memahami dan menggunakan Git
- Siswa memahami dan menerapkan Git Flow

Kenapa sih harus tau tentang Version Control?

VERSION CONTROL

BINAR

- Makalah Bab V
- Makalah Bab V Rev
- Makalah Bab V Rev 2
- Makalah Bab V Rev 3
- Makalah Bab V Rev 3 + DaPus
- Makalah Bab V Rev 3 + DaPus Revisi Lagi
- Makalah Bab V Rev 3 + DaPus Revisi Lagi Final
- Makalah Bab V Rev 3 + DaPus Revisi Lagi Final Fix
- Makalah Bab V Rev 3 + DaPus Revisi Lagi Final Fix Paten

Ayo.. ada yang pernah buat file dengan revisi yang sebanyak ini?

Pasti terlihat berantakan dan tidak teratur. Selain itu juga susah untuk melihat perubahan dari satu revisi dengan revisi yang lain.

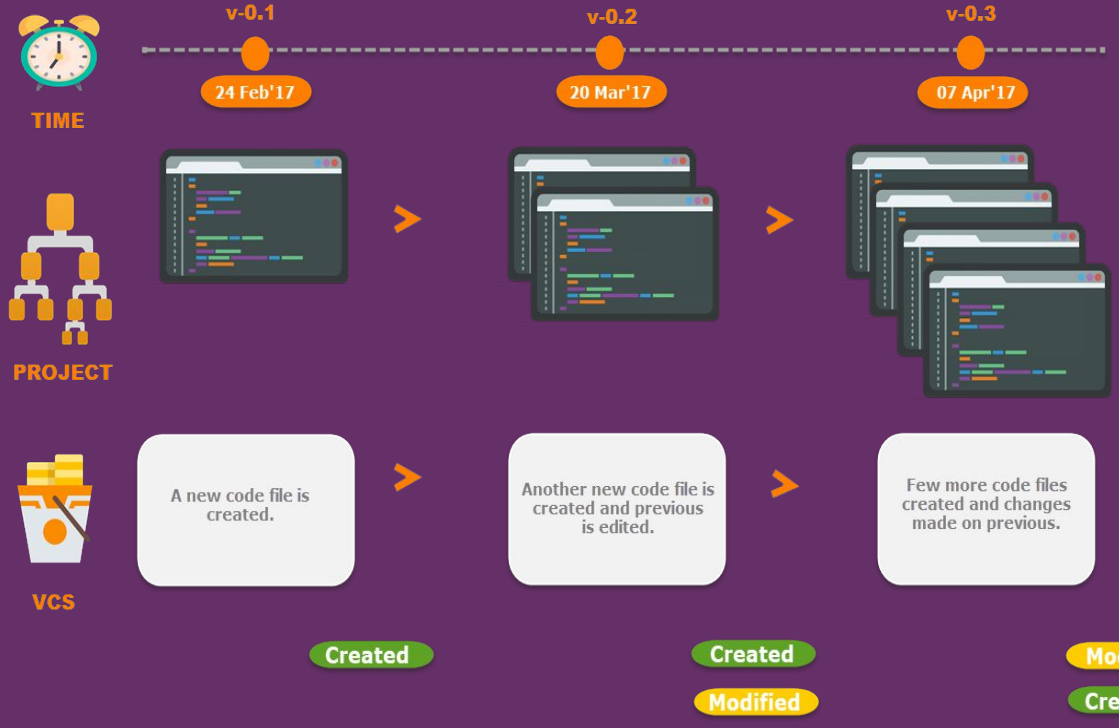
Inilah gunanya version control yang bisa menyelesaikan permasalahan seperti ini. Jadi kita hanya perlu 1 file saja!

Pada pembuatan software, seringkali terjadi beberapa perubahan selama proses pembuatan. Perubahan ini belum tentu benar dan belum tentu salah. Seringkali kita perlu menyimpan perubahan tersebut untuk melihat perubahan terhadap fitur software sebelum/setelah dilakukan perubahan. Atau dalam kasus-kasus tertentu ternyata perubahan tersebut tidak jadi diterapkan.

Oleh karena itu, version control merupakan hal yang penting, karena dengan version control kita dapat:

1. Menelusuri perubahan pada software yang sedang dikembangkan
2. Membuat beberapa versi sebagai opsi penawaran kepada customer/client
3. Mengulang atau membatalkan suatu perubahan.
4. Mengarsipkan kode-kode program sesuai histori perubahan.

BINAR



Nah kan, semakin banyak perubahan, semakin membingungkan bukan?

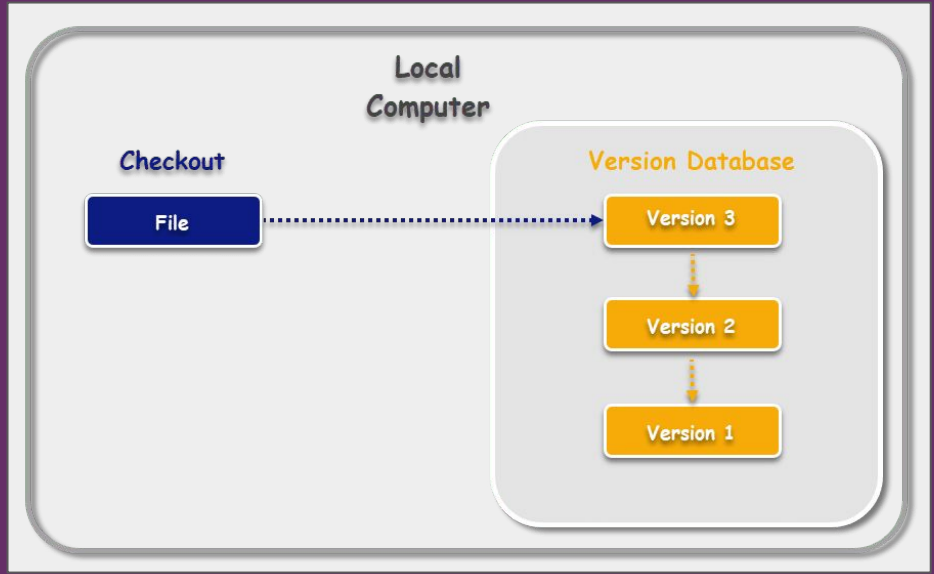
Terdapat beberapa jenis
Version Control
berdasarkan penerapannya

1. Local Version Control System

Version control yang hanya pada komputer pengguna. Berkas tidak dibagikan dengan pengguna di komputer lain.

Dalam kondisi seperti ini, kolaborasi tidak efektif karena hanya melibatkan satu pengguna pada satu komputer.

Jika tujuannya untuk menyimpan dan menelusuri perubahan yang dilakukan oleh pribadi, sudah cukup.



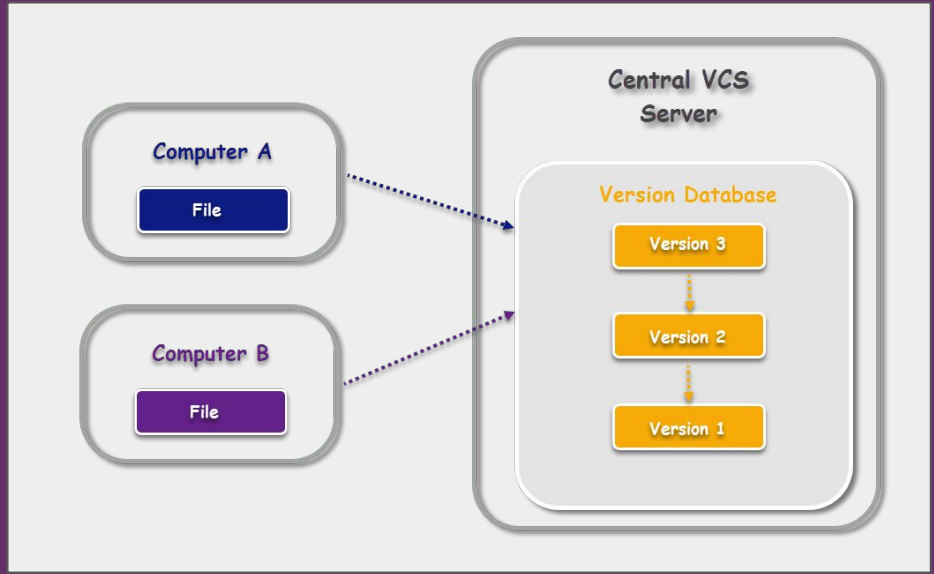
BINAR

2. Central Version Control System

Central version control bisa menghubungkan dua komputer atau lebih untuk berkolaborasi.

Biasanya hal seperti ini digunakan pada jaringan lokal yang tidak terlalu banyak memiliki komputer client.

Dengan jenis version control ini, memungkinkan kita untuk berkolaborasi dengan pengguna lain di komputer yang berbeda dalam satu jaringan. Namun hal ini tidak memiliki portabilitas yang baik. Dikarenakan berkas tidak bisa diakses dimana saja, namun harus melalui jaringan lokal.



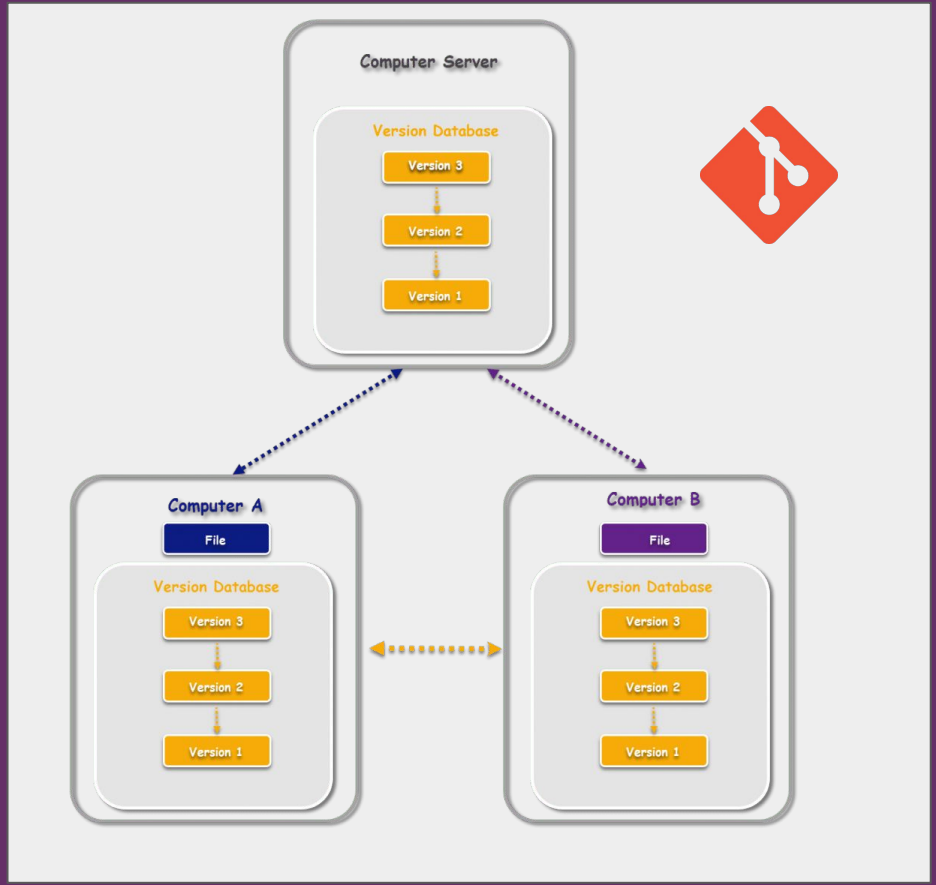
BINAR

3. Distributed Version Control System

Distributed version control mengharuskan pengguna untuk mempunyai berkas version control pada masing-masing komputer.

Antar komputer dihubungkan oleh server yang menangani version control secara keseluruhan. Server ini biasanya ada di cloud atau internet sehingga setiap pengguna bisa mengakses berkas dimana saja asalkan terhubung dan memiliki akses ke server.

Jenis version control inilah yang akan kita pelajari selanjutnya. Kita akan menggunakan Git sebagai version control.



Memang apa sih Git itu?

Merupakan salah satu perangkat version control system yang paling banyak digunakan. Memiliki fitur-fitur:

1. Version Control yang terdistribusi
2. Operasi bersifat atomik (berhasil seluruhnya atau gagal seluruhnya)
3. Semua file disimpan pada folder .git
4. Data model yang mendukung integritas data
5. Staging area / index untuk mereview aplikasi sebelum ditayangkan untuk umum

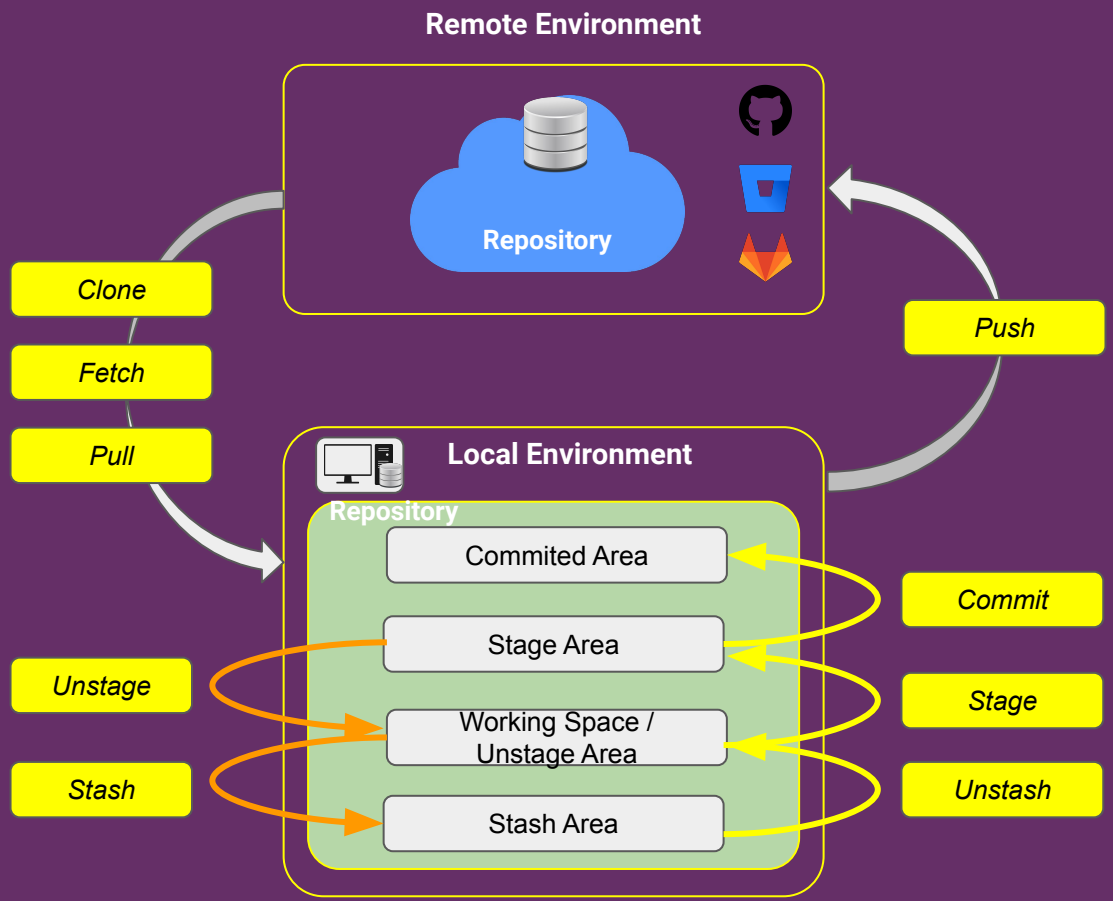
Lalu apakah ada perangkat version control selain Git?

Tentu saja ada dong!.. Ini dia version control selain Git...

SUBVERSION

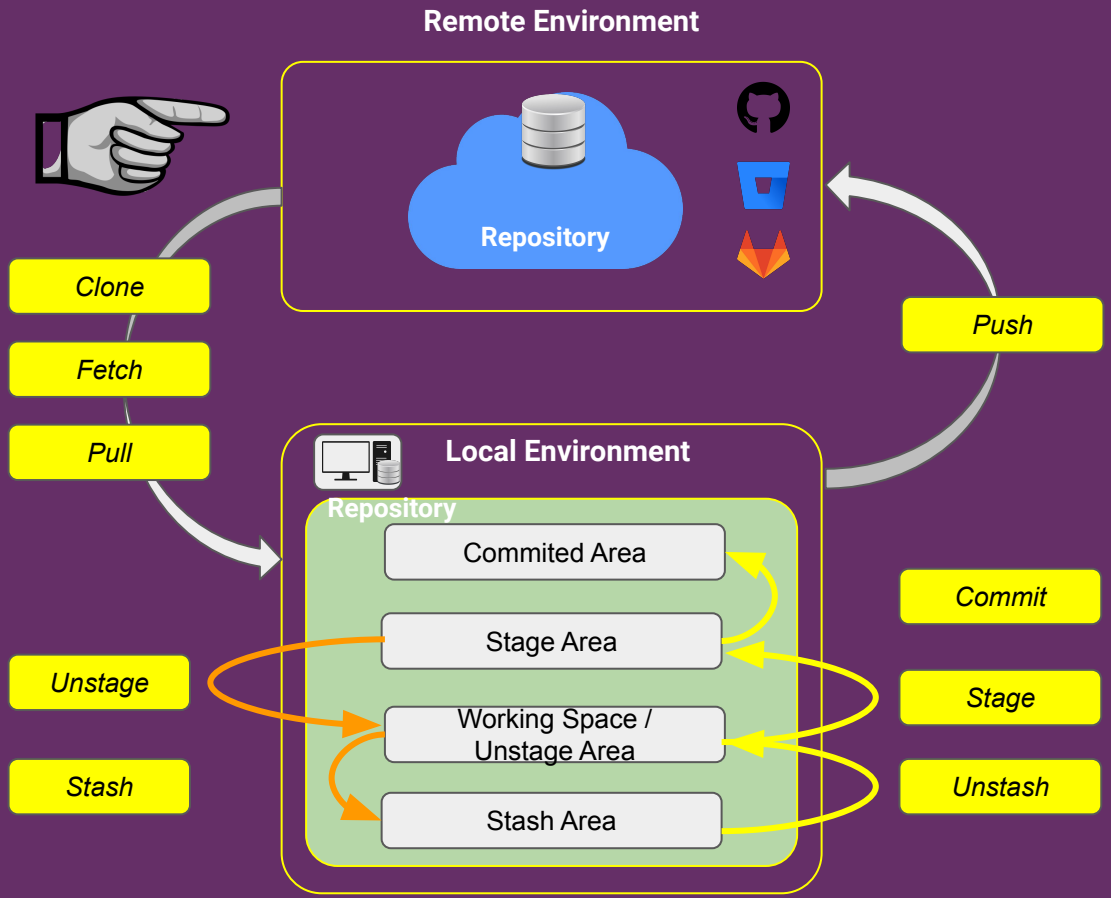
Google Cloud
Source Repository

mercurial



Secara keseluruhan, seperti inilah ekosistem dalam Git version control

Mari kita bahas!

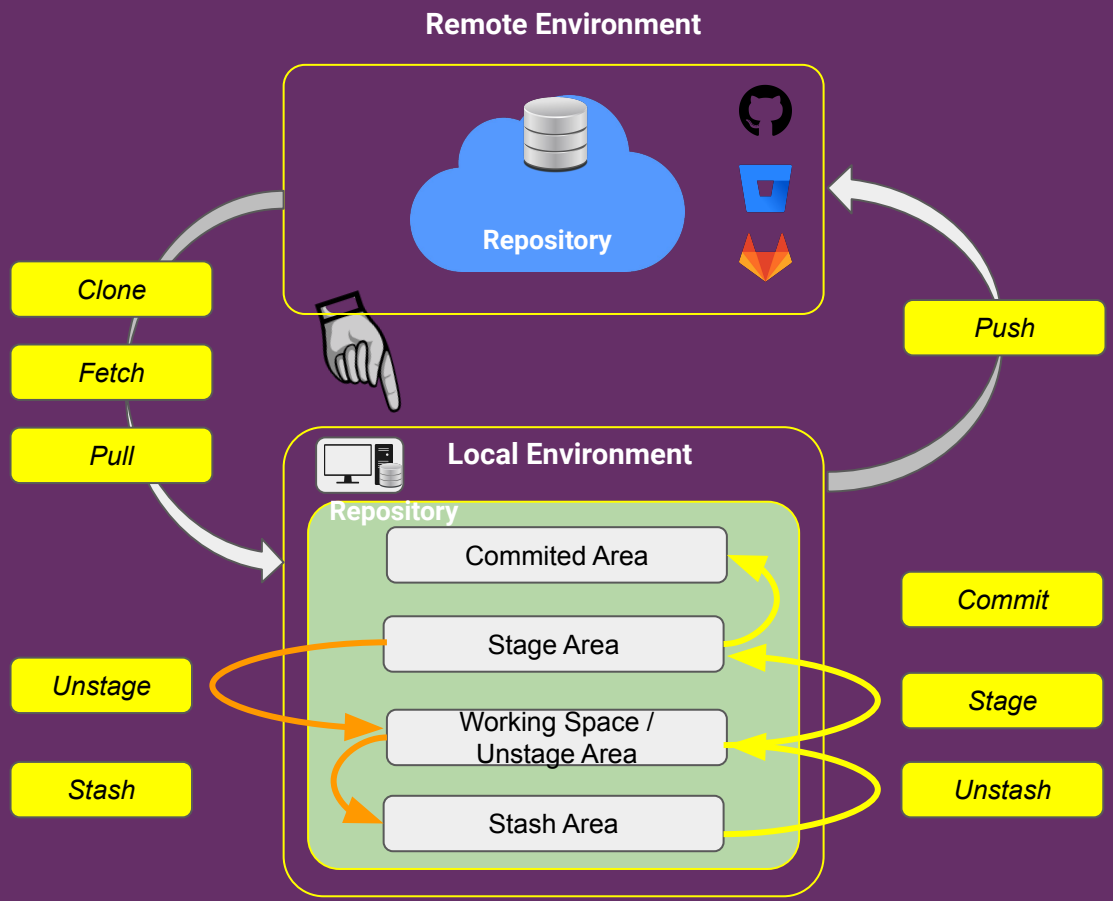


Remote Environment :

Adalah penyedia layanan Version Control yang ada di internet. Seperti Github, Gitlab, Bitbucket, dll.

Pada remote environment, kita bisa menyimpan repository / project aplikasi kita.

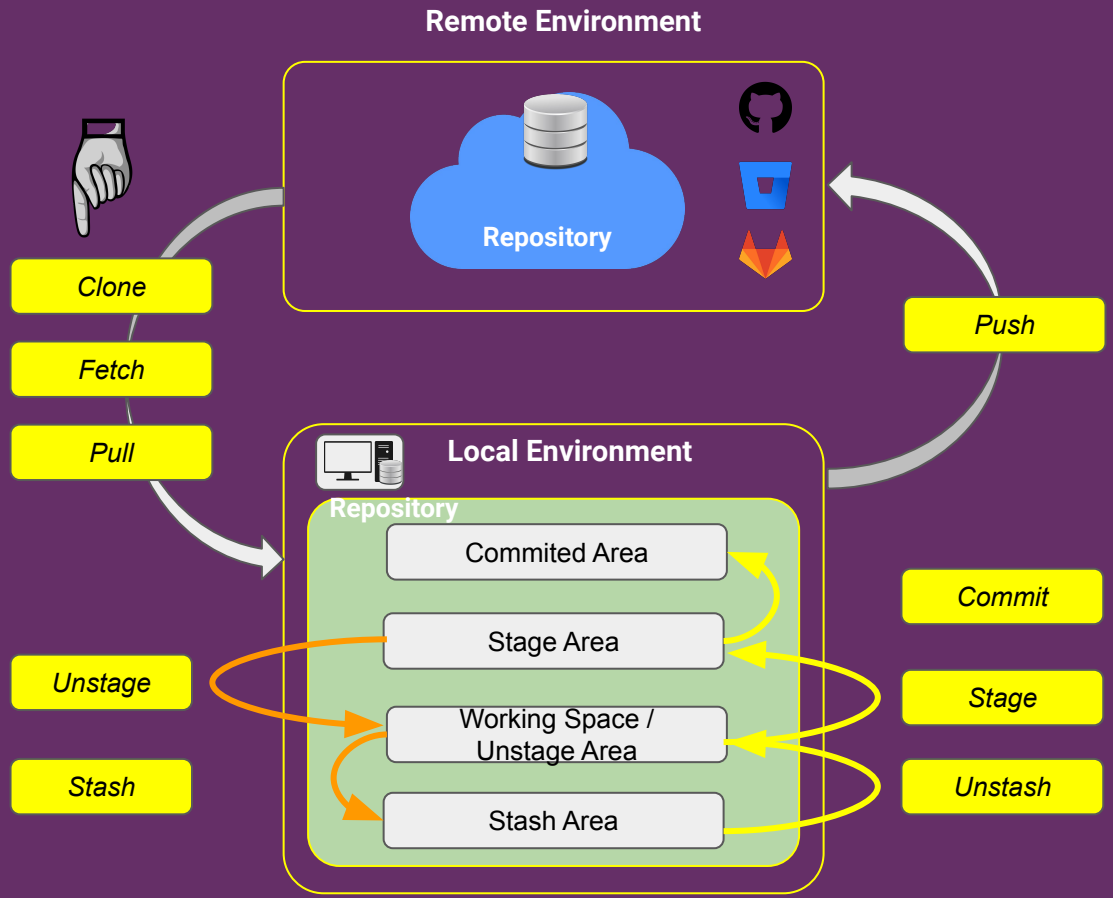
Karena itulah Remote Repository dapat kita akses dari mana saja asalkan kita terhubung ke internet.



Local Environment :

Adalah lingkungan komputer Local yang sedang dipakai. Pada local environment kita bisa terdapat banyak repository (project).

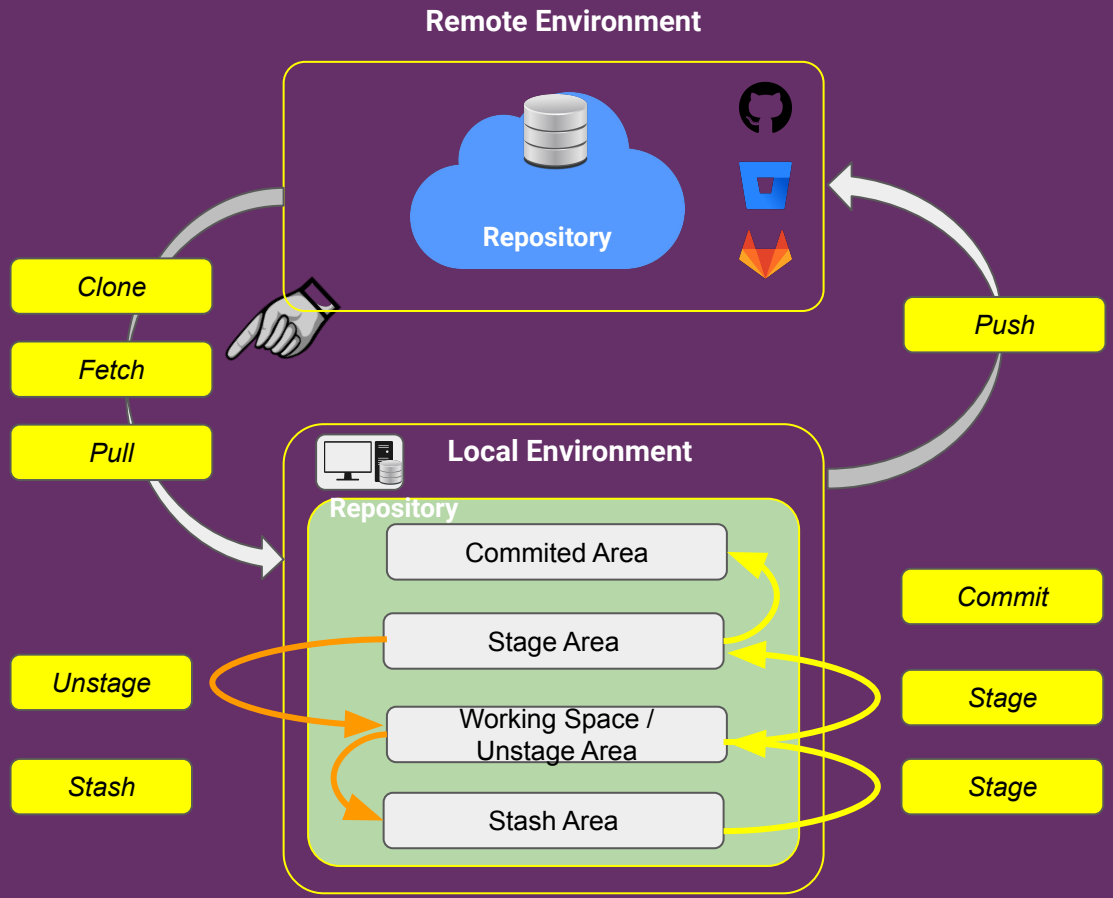
Karena itulah kita dapat melakukan berbagai aksi Git terhadap Repository yang ada pada Local Environment tanpa terhubung ke Internet.



[Dari Remote ke Local]

Clone :

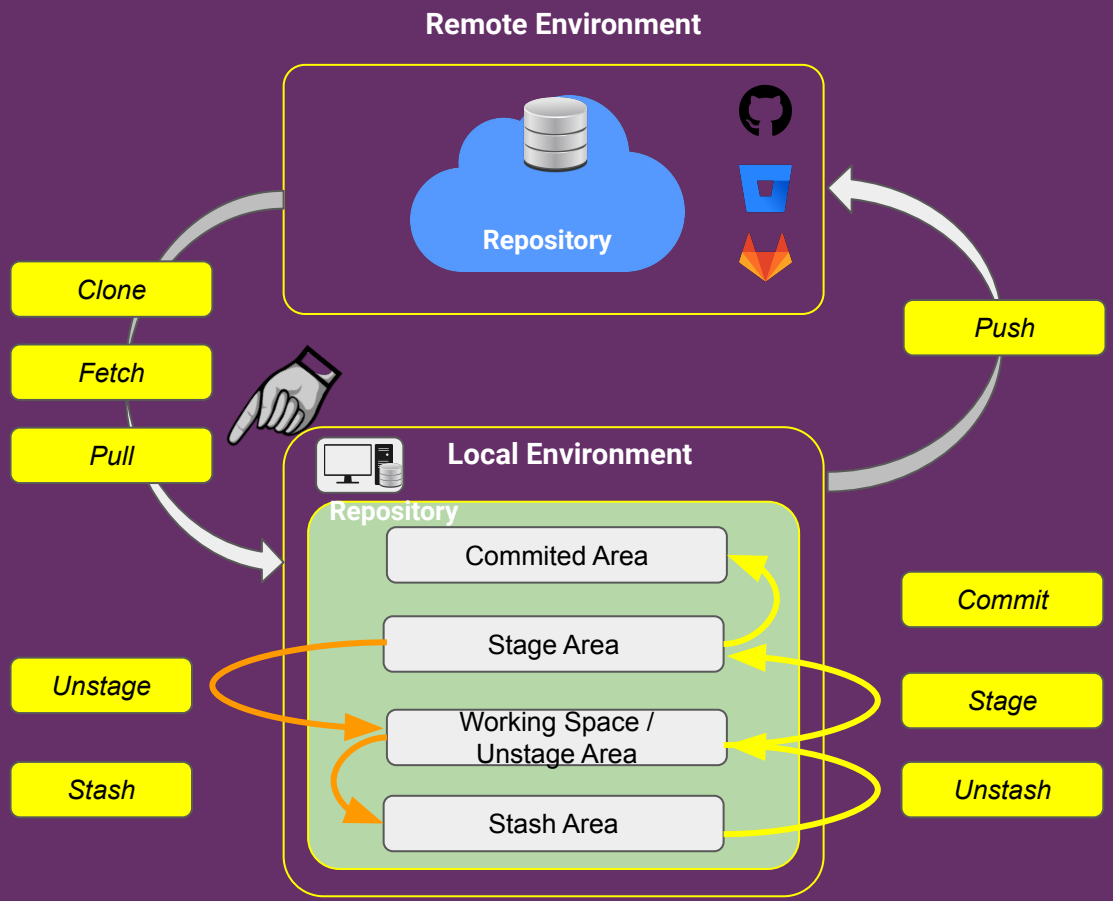
Perintah clone adalah perintah untuk mereplikasi Repository kita yang ada di Remote ke Local sehingga kita bisa melakukan berbagai perintah lain seperti fetch, pull, dan push.



[Dari Remote ke Local]

Fetch :

Adalah perintah untuk mendapatkan hal-hal apa saja yang berubah pada Remote Repository. Namun perubahan yang didapatkan tidak akan berpengaruh pada repository local kita hingga kita melakukan perintah pull.



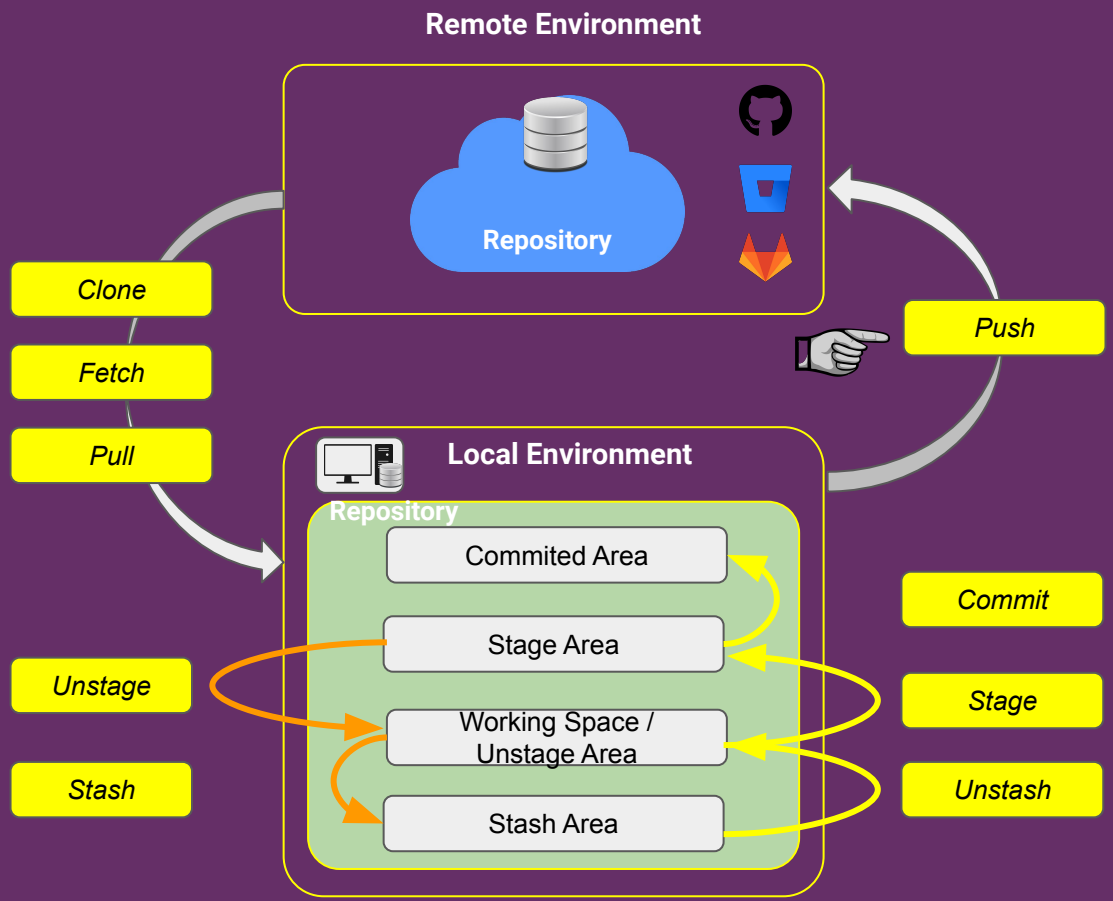
[Dari Remote ke Local] Pull :

Adalah perintah untuk ‘menarik’ berbagai macam perubahan yang terjadi di Remote.

Perubahan meliputi perubahan file, penghapusan file, maupun penambahan file.

Sebelum melakukan perintah pull, kondisi local repository harus bersih dari perubahan. Hal ini berarti segala perubahan harus dilakukan commit ataupun tidak ada perubahan dengan melakukan *stash*.

Setelah melakukan perintah pull, akan terjadi conflict jika perubahan yang ada dalam commit local kita berbeda dengan perubahan yang ada pada commit remote.



[Dari Local ke Remote]

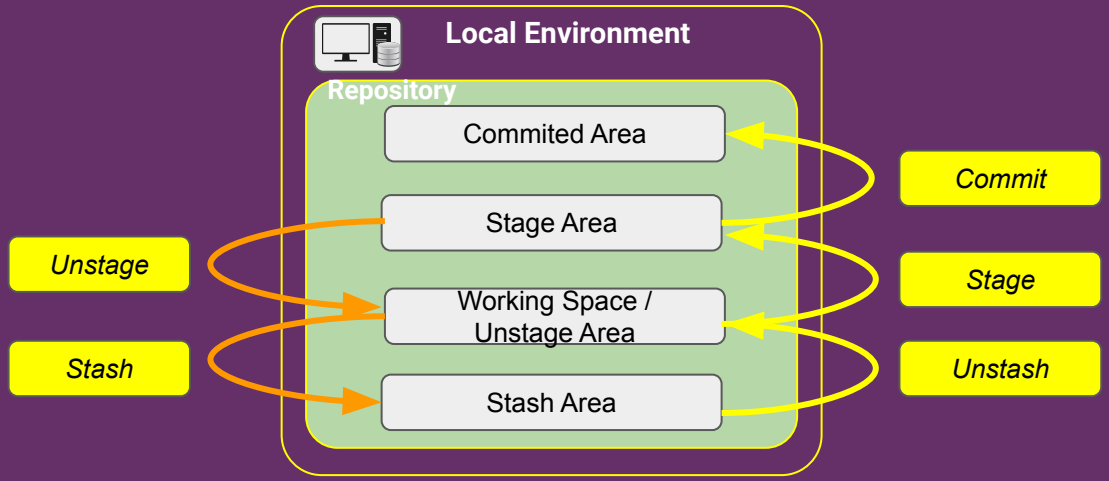
Push :

Adalah kebalikan dari perintah pull. Jika perintah pull menarik perubahan dari remote ke local, jika push mendorong perubahan dari local ke remote.

Semua perubahan file di local, pembuatan branch, dll akan dikirimkan ke remote.

Nah, sudah tahu kan operasi git apa saja yang bisa dilakukan dari Local Environment ke Remote Environment maupun sebaliknya?

Sekarang kita pelajari operasi Git yang terjadi
Hanya dalam Local Environment!



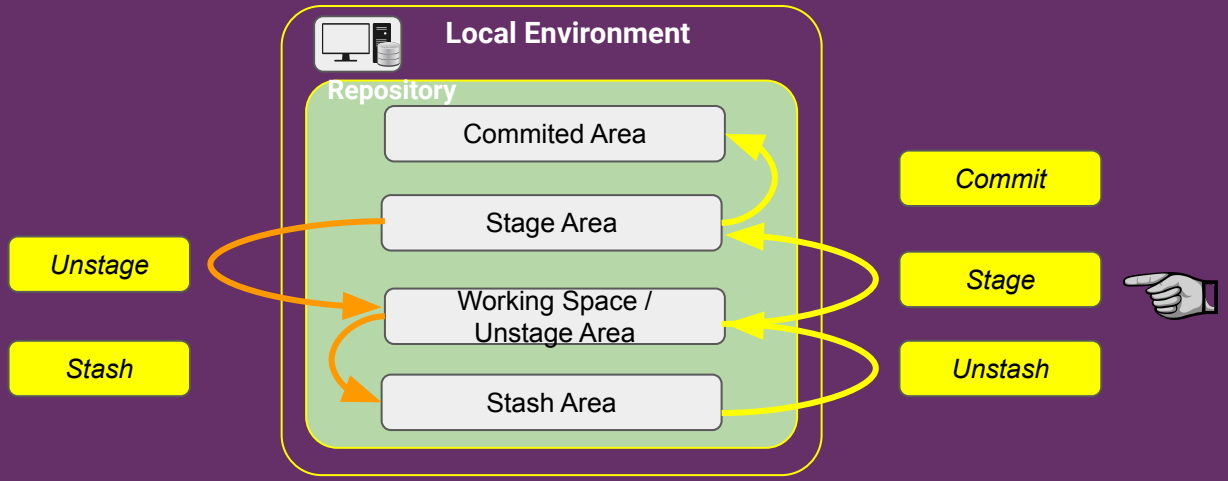
Eits! Sebelum mengetahui perintahnya, kudu tahu dulu tentang tempat-tempatnya nih!

Commited Area : Area tempat commit dikumpulkan. Commit adalah kumpulan perubahan yang direkam.

Stage Area : Area tempat file-file kita diawasi perubahannya oleh Git.

Working Space / Unstage Area : Tempat file yang tidak terawasi / belum masuk ke Stage Area.

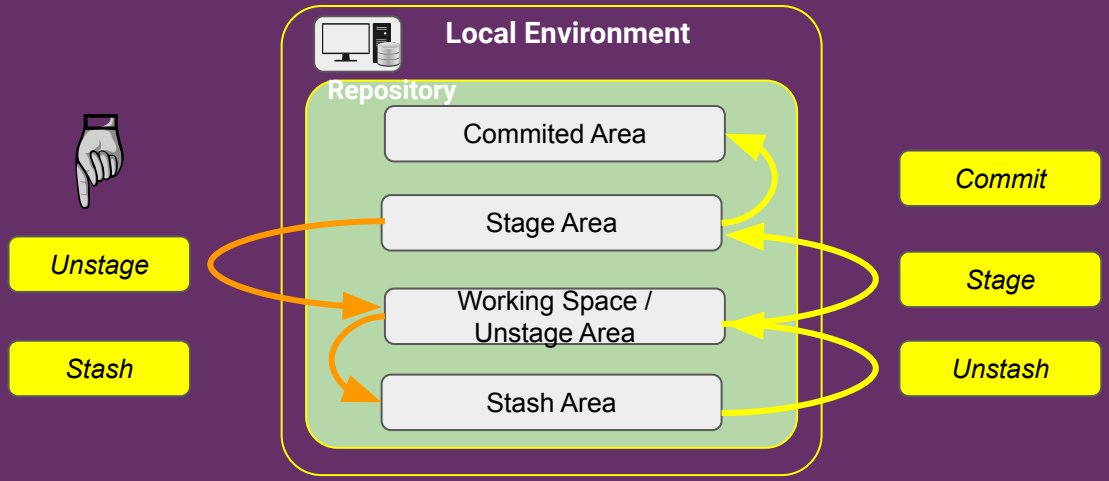
Stash Area : Tempat berkumpulnya stash. Stash adalah perubahan yang disimpan tanpa melakukan commit.



Stage

Stage adalah perintah untuk menambahkan file yang sebelumnya di Unstage Area / Tidak Terawasi oleh Git ke Stage Area. Sehingga setiap perubahan yang terjadi pada file yang ada di Stage Area akan terdeteksi oleh git dan memungkinkan untuk dilakukan commit.

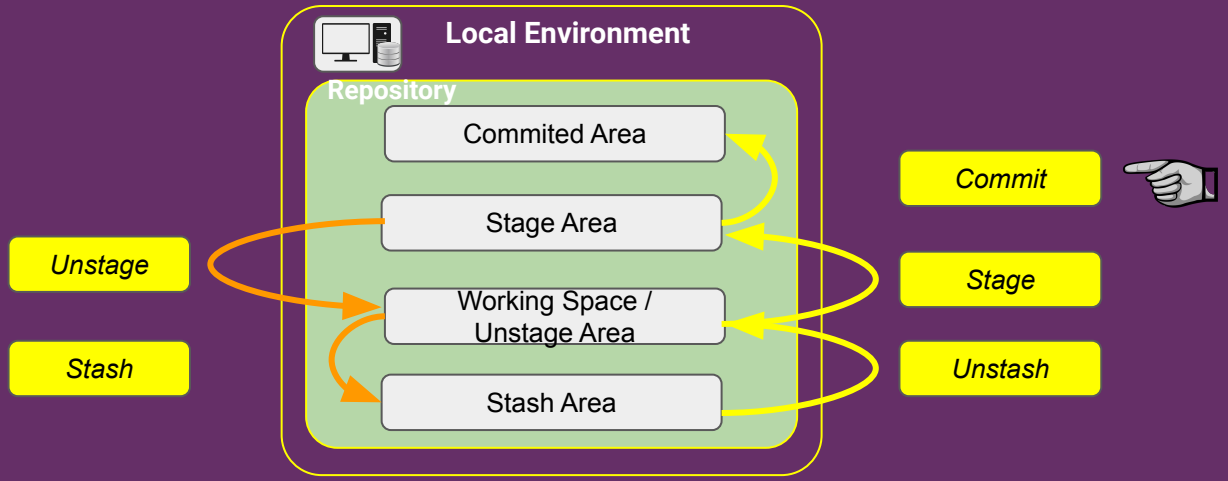
Cara mengetahui status perubahan pada Git Repository adalah dengan command : **`git status.`**
Sedangkan untuk menambahkan file ke Stage Area, bida dengan perintah **`git add`**



Unstage

Unstage adalah perintah untuk menghapus file dari Stage Area dan memindahkannya ke Working Space / Unstage Area. Dengan melakukan hal ini, file yang telah di unstage tidak akan diawasi oleh Git dan tidak dimungkinkan dilakukan commit. Karena commit hanya bisa dilakukan terhadap file yang ada pada Stage Area .

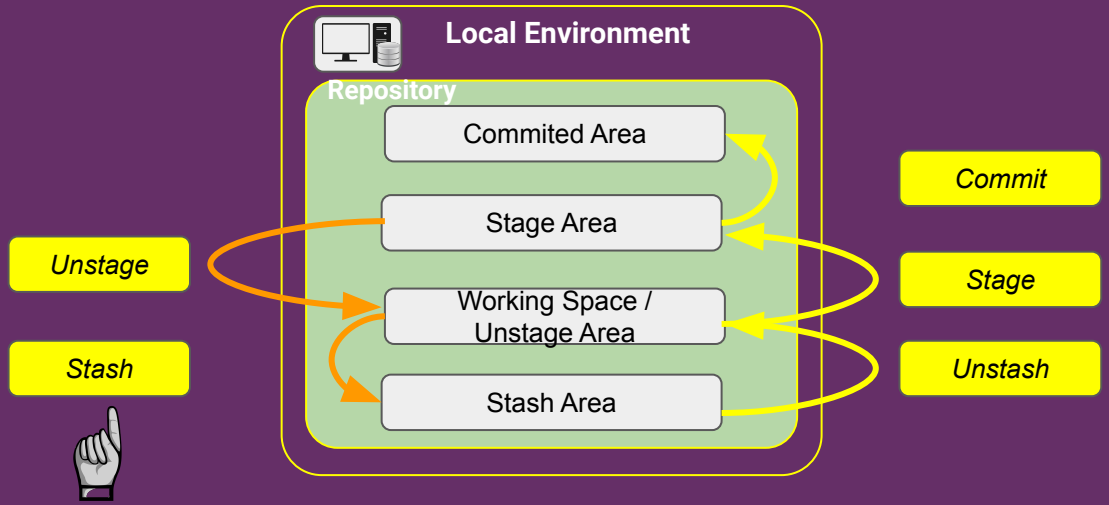
Untuk melakukan unstage bisa menggunakan perintah `git reset`



Commit

Commit adalah perintah untuk menyimpan rekaman perubahan yang terjadi pada file yang ada di Stage Area. Sebelum melakukan commit, harus dipastikan kondisi pada Working Space sudah bersih dari perubahan. Untuk membersihkannya bisa menggunakan perintah commit maupun stash.

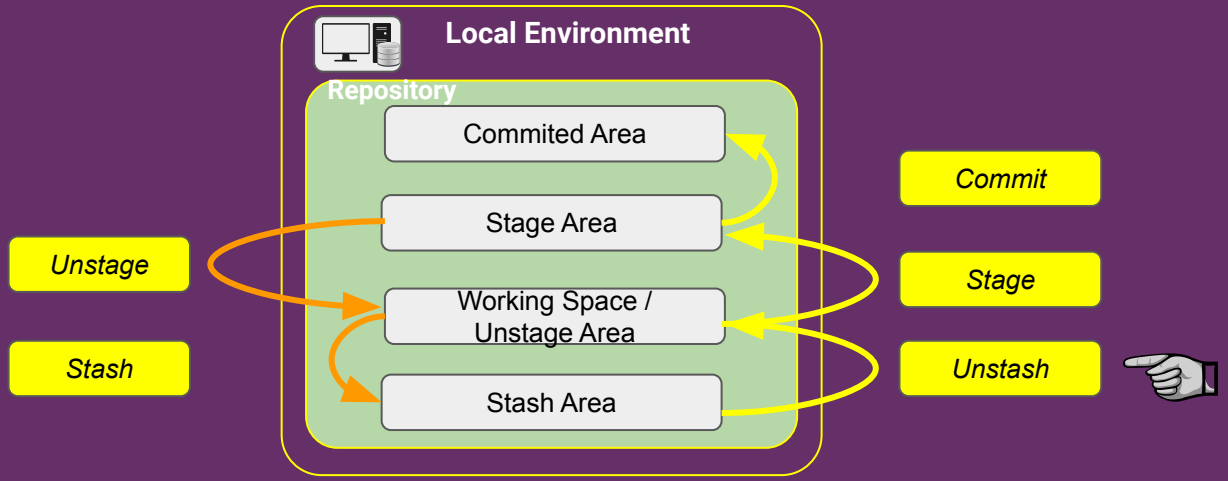
Untuk melakukan commit bisa menggunakan perintah `git commit -m “Pesan Commit”`



Stash

Stash adalah perintah untuk menyimpan rekaman perubahan yang terjadi pada Working Space / Unstage Area. Hal ini berbeda dengan commit karena perintah stash akan tersimpan ke stash area. Kita melakukan stash jika sedang membangun code dengan progress yang nanggung. Code yang dibuat masih jauh dari kata selesai dan berfungsi, namun terlalu sayang untuk dihapus atau diundo.

Untuk melakukan stash bisa menggunakan perintah `git stash`

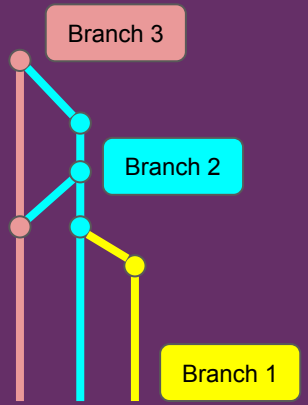


Unstash

Unstash adalah perintah untuk menerapkan kembali code nanggung yang kita simpan di Stash Area dengan perintah stash sebelumnya.
Proses unstash biasa disebut dengan proses Apply Stash.

Untuk melakukan stash bisa menggunakan perintah **`git stash pop`** **atau** **`git stash apply`**

Branch



Branch jika diterjemahkan langsung ke bahasa indonesia artinya Cabang. Branch inilah komponen utama pada version control. Apakah kita sudah menyadari, jika membuat suatu repository, setidaknya ada satu branch yang dibuat, biasanya bernama branch **Master.**

Fungsi branch sangat beragam. Contoh sederhana manfaat branch adalah kita bisa beralih code dengan sangat mudah hanya dalam sekejap!.

Sebagai contoh, pada file Main.java terdapat code untuk menampilkan teks salam pengenal. Dengan adanya branch, kita bisa berganti-ganti salam pengenal dengan sangat mudahnya!

GITFLOW WORKFLOW

Gitflow Workflow adalah cara kerja kita menggunakan Git. Gitflow bekerja berdasarkan branch yang sudah ditentukan.

GITFLOW WORKFLOW

Branch Master :
Branch induk tempat berbagai perubahan aplikasi terjadi. Pada branch ini merekam perubahan yang terjadi walaupun tidak untuk dirilis. Hal inilah yang membedakan dengan branch release.

Branch Hotfix :
Branch Hotfix akan digunakan jika ternyata ada bug ataupun kesalahan pada versi aplikasi yang ada pada branch master, dan bug / kesalahan ini cukup penting dan diperlukan untuk segera diselesaikan.

GITFLOW WORKFLOW

Branch Release :
Sesuai namanya, branch ini berfungsi jika terdapat pembaruan ataupun penambahan fitur pada aplikasi lalu dirilis.

Branch Develop :
Pada branch develop, para pengembang aplikasi melakukan pengembangan tahap-tahap awal dan sangat rentan terjadinya kesalahan. Pada branch ini juga terjadi berbagai macam percobaan.

GITFLOW WORKFLOW

Branch Feature :
Setiap fitur baru harus berada di cabang ini. Tapi, branch feature ini tidak bercabang dari master , feature cabang menggunakan develop sebagai cabang induknya. Ketika suatu fitur selesai, ia akan bergabung kembali ke dalam pengembangan . Fitur tidak boleh berinteraksi langsung dengan master .

BINAR

Git Cheat Sheet

For more awesome cheat sheets visit rebellabs.org!

REBELLABS by ZEROTURNAROUND

Create a Repository

From scratch -- Create a new local repository
```
$ git init [project name]
```

Download from an existing repository
```
$ git clone my_url
```

Observe your Repository

List new or modified files not yet committed
```
$ git status
```

Show the changes to files not yet staged
```
$ git diff
```

Show the changes to staged files
```
$ git diff --cached
```

Show all staged and unstaged file changes
```
$ git diff HEAD
```

Show the changes between two commit ids
```
$ git diff commit1 commit2
```

List the change dates and authors for a file
```
$ git blame [file]
```

Show the file changes for a commit id and/or file
```
$ git show [commit]:[file]
```

Show full change history
```
$ git log
```

Show change history for file/directory including diffs
```
$ git log -p [file/directory]
```

Working with Branches

List all local branches
```
$ git branch
```

List all branches, local and remote
```
$ git branch -av
```

Switch to a branch, my_branch, and update working directory
```
$ git checkout my_branch
```

Create a new branch called new_branch
```
$ git branch new_branch
```

Delete the branch called my_branch
```
$ git branch -d my_branch
```

Merge branch_a into branch_b
```
$ git checkout branch_b
$ git merge branch_a
```

Tag the current commit
```
$ git tag my_tag
```

Make a change

Stages the file, ready for commit
```
$ git add [file]
```

Stage all changed files, ready for commit
```
$ git add .
```

Commit all staged files to versioned history
```
$ git commit -m "commit message"
```

Commit all your tracked files to versioned history
```
$ git commit -am "commit message"
```

Unstages file, keeping the file changes
```
$ git reset [file]
```

Revert everything to the last commit
```
$ git reset --hard
```

Synchronize

Get the latest changes from origin (no merge)
```
$ git fetch
```

Fetch the latest changes from origin and merge
```
$ git pull
```

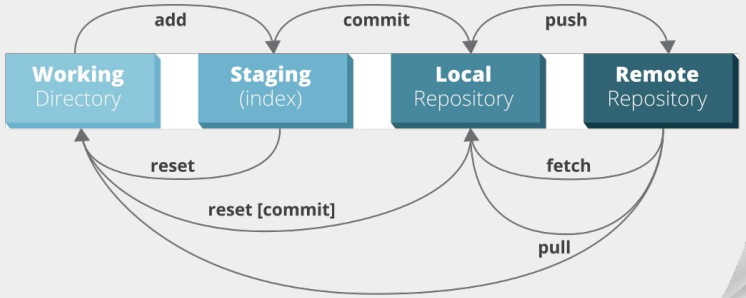
Fetch the latest changes from origin and rebase
```
$ git pull --rebase
```

Push local changes to the origin
```
$ git push
```

Finally!

When in doubt, use git help
```
$ git command --help
```

Or visit https://training.github.com/ for official GitHub training.



BROUGHT TO YOU BY JRebel

GIT CHEAT SHEET

Berikut adalah kumpulan syntax yang ada pada Git!

BINAR

Version Control
Perangkat lunak yang membantu mengatur dan menelusuri perubahan yang terjadi pada project / repository dan untuk berkolaborasi.

Git
Adalah salah satu Perangkat Version Control yang populer. Penyedia remote Git ada Github, Gitlab, Bitbucket, dll.

Jenis Version Control
Jenis version control ada 3, Local VCS, Central VCS, dan Distributed VCS.

Environment Git
Environment Git ada Local Environment dan Remote Environment.

Operasi Git Antara Local dan Remote Environment
Arah Remote-Local : Clone, Fetch, Pull. **Arah Local-Remote** : Push.

Operasi Git yang terjadi di Local
Beberapa operasi Git yang bisa terjadi pada Local adalah Commit, Stage, Unstage, Stash, dan Unstash (Apply Stash).

TUGAS

Kita ingin mendesain web perusahaan (ecommerce), tapi karena dikejar deadline jadi kita memutuskan untuk berkolaborasi dengan teman-teman.

Pada dasarnya web perusahaan ini akan memiliki menu: Home, Product and Services, FAQ, About, dan Contact. Jadi, kita perlu membuat 5 branch fitur.

Silakan gunakan alur kerja git, dan buat permintaan gabungan atau tarik permintaan untuk menguasai atau mengembangkan branch dari branch fitur.

BINAR

Referensi

https://medium.com/@patrickporto/4-branching-workflows-for-git-30d0aaee7bf
https://www.udacity.com/course/version-control-with-git--ud123
https://try.github.io