Git

個人個人がワークツリーで作業をし、ローカルリポジトリにコミットし、リモートリポジトリにプッシュすることでソースコードが共有される。そしてリモートリポジトリからローカルリポジトリにプルすることで、他人が書いたコードや自分が過去に書いたコードをダウンロード(？)できる

リポジトリ：ソースコードのデータベース(Git)のこと

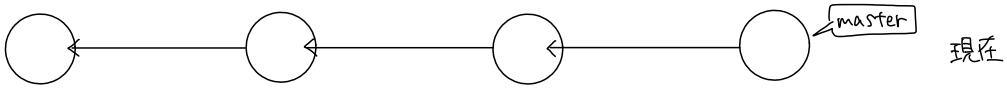
ローカルリポジトリ：個人のPCにあるデータベース(Git)

リモートリポジトリ：ネット上にある、データベース(Git)。その1つがGithub

ワークツリー：Gitを使わなくても存在するディレクトリとかファイルのこと

コミット：ワークツリーの状態をGitに登録すること。どこに登録するかは後述

登録するときに、作者や1つ前のバージョン(リビジョン)が記録される(これ大事)



上図のようにリレーみたいに繋がっている(この図をコミットグラフという：家系図みたい！)

※2つ前のバージョンは「1つ前のもう1つ前」って感じ

「コミットメッセージ」というものを書いて、なんでコミットするかを書く！(後の図参照)

でも、複数人で作業していると、AさんとBくんでは違うものを作ってて自分だけの「リレー」をしたいときがある。そのときにリレーを分岐させる。それで別々にコミットしていくとお互い邪魔しない。

ブランチ：この分岐路のこと

masterブランチ：メインのリレー、一番初めから存在している

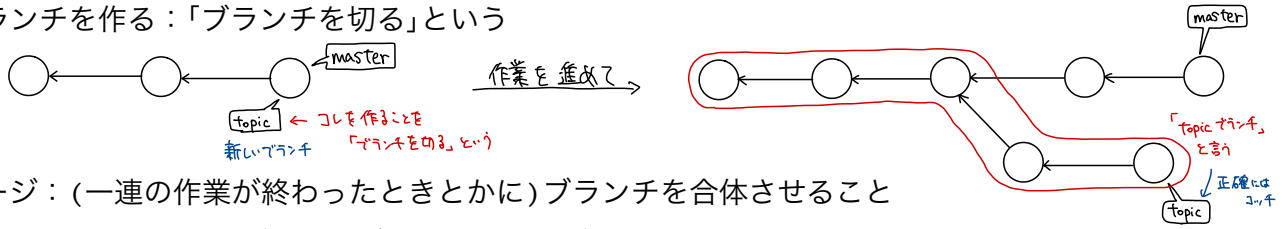
HEAD：勝手に作られていて、現在の作業から見て直前のコミットを指している(後の図参照)

※注意！

正確には、ブランチは「あるコミットを指している」だけです。つまり、「ブランチ」は「最新コミット」の別名

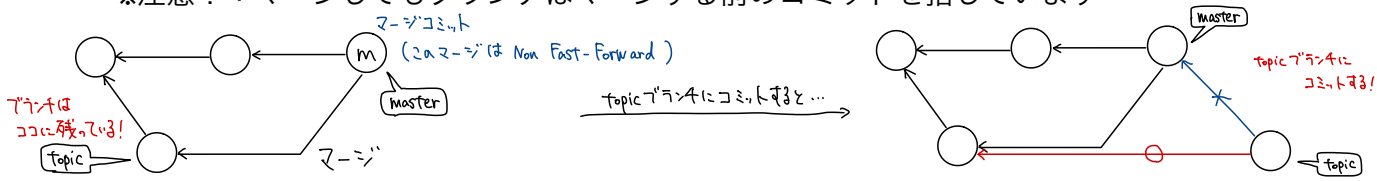
でも、普通、その指されてるコミットから連なってるコミット全てを「ブランチ」という

ブランチを作る：「ブランチを切る」という

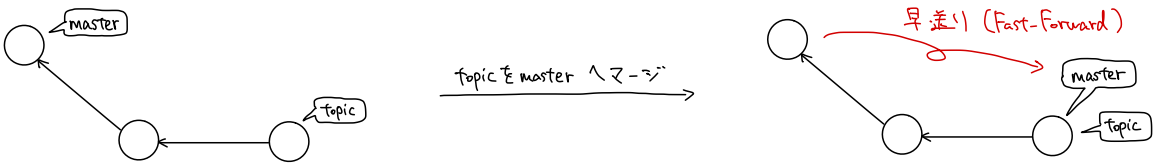


マージ：(一連の作業が終わったときとかに)ブランチを合体させること

※注意！：マージしてもブランチはマージする前のコミットを指しています



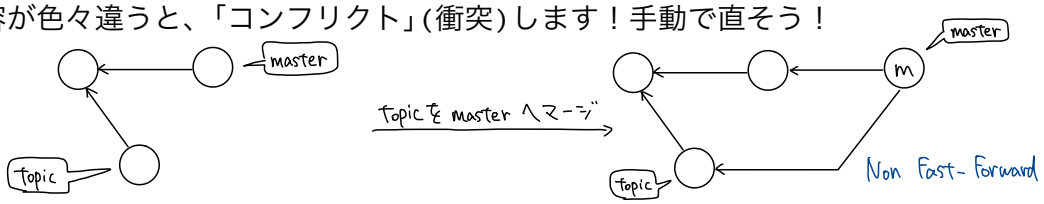
① Fast-Forward：枝分かれしたけど元のやつ何も変わってないじゃん！進めればいいね！



② Non Fasr-Forward：枝分かれして色々変わっちゃったからくっつける！

新しくできるコミットを「マージコミット」という

内容が色々違うと、「コンフリクト」(衝突)します！手動で直そう！



※注意！

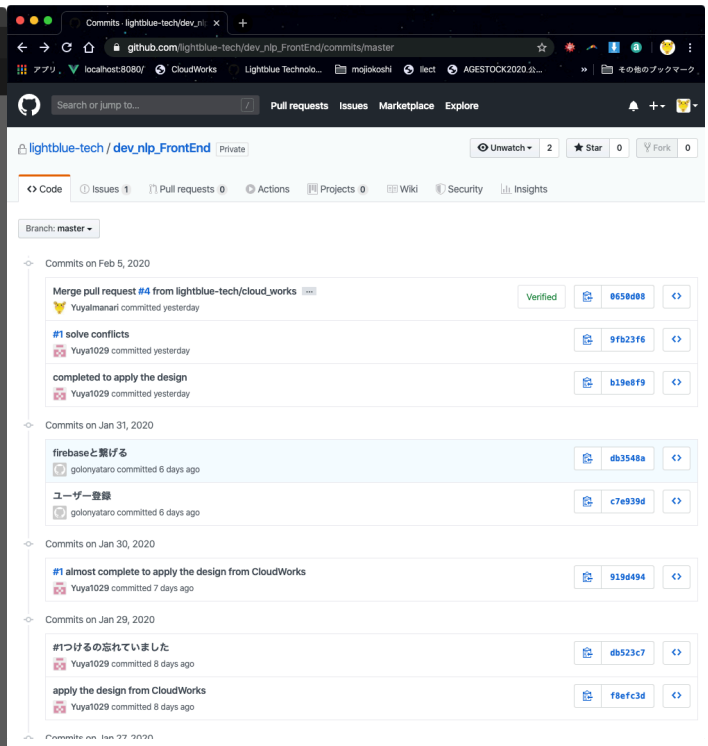
Fast-Forwardはブランチをマージしたという事実がコミットグラフ(家系図)に残らない！

→ブランチのマージが取り消ししにくい

※超重要！！！！

もう1種類「リベース」というものがあるけど、グッチャグチャになるから絶対使っちゃダメ！

[コミットグラフ・コミットメッセージ]

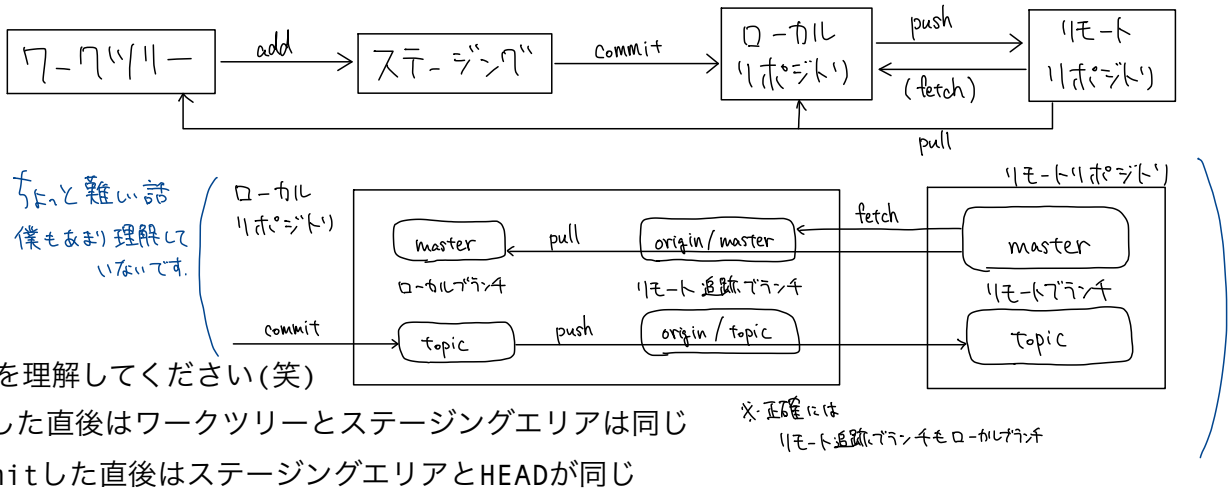


▲ Githubで見たコミット(masterブランチのみ)

◀ VSCodeのGit History(拡張機能)

ブランチについてわからないなら以下URL

https://www.slideshare.net/kotas/git-15276118
https://qiita.com/jesus_isao/items/2a0495c973a4c911c2cc



上図を理解してください(笑)

addした直後はワークツリーとステージングエリアは同じ

commitした直後はステージングエリアとHEADが同じ

▼ 初回だけ ▼

一番最初にGitを初期化　　$ git init

※”initialize”の略

ー>”Initialized empty Git repository in パス“と表示される

プッシュ先のリモートリポジトリを登録　　$ git remote add origin <URL>

リモートリポジトリをクローン　　$ git clone <URL>

▲ 初回だけ ▲

※参考URL：https://prog-8.com/docs/git-env

※Windows用：https://prog-8.com/docs/git-env-win

※参考URL：https://qiita.com/devzip8/items/28ac253ea295ad6c2b73

▼ 毎回 ▼

共有するファイルを選択(ステージングに上げる)　$ git add ファイル名

ー>全てのファイルをステージングエリアに上げる　$ git add .

ステージしたファイルを取り消す　　$ git reset ファイル名

ステージングエリアの状態を確認する　　$ git status

※Changes to be comittedはHEADの状態とステージングエリアの状態の差

コミット(記録)する　　$ git commit -m “コミットメッセージ”

プッシュ(アップロード)する　　$ git push origin ブランチ名

プル(ダウンロード)する　　$ git pull origin ブランチ名

※参考URL：https://qiita.com/sayama0402/items/9afbb519d97327b9f05c

ブランチを作成する　　$ git branch ブランチ名

ブランチを確認する　　$ git branch

ブランチを切り替える　　$ git checkout ブランチ名

ー>ブランチの作成と切り替え　　$ git checkout -b ブランチ名

マージ済みローカルブランチを削除する　　$ git branch -d ブランチ名

ローカルブランチを削除する(マージ前も可)　　$ git branch -D ブランチ名

リモートブランチを削除する　　$ git push --delete origin ブランチ名

もしくは　　$ git push origin :ブランチ名

他人が削除したブランチを反映する　　$ git fetch -p

※参考URL：https://backlog.com/ja/git-tutorial/stepup/07/

ブランチ名を変更する　　$ git branch -m hoge fuga

変更したブランチは普通にpushする

※参考URL：https://qiita.com/hogeta_/items/e47dfb0cf88270ef2802

当然ブランチを切り替えかえる前にcommitしろと怒られる！　　※ ▼“save”以下は省略可

ー>変更を退避する　　$ git stash save “メッセージ”

退避した作業の一覧を見る　　$ git stash list

退避した作業をブランチに戻す　　$ git stash apply stash@{番号}

※stash名を指定しないと、stash@{0}をブランチに戻す

※addしていた変更もaddされていないものとして戻される、防ぎたいなら“--index”オプション

退避した作業を消す　　$ git stash drop stash@{番号}

退避した作業を戻し消す　　$ git stash pop stash@{番号}

※参考URL：https://qiita.com/chihiro/items/f373873d5c2dfbd03250

マージするときは･･･まずマージされる(受け入れる側の)ブランチに移動する

ー>ブランチをマージする　　$ git merge 取り込まれるブランチ名

衝突したら･･･修正する

ー>ステージングに上げて　　$ git add .

ー>コミットする　　$ git commit -m “コミットメッセージ”

エラー：fatal: The current branch branch-a has no upstream branch.

ー>上流ブランチがない！

ー>pushで上流ブランチに追加する　　$ git push --set-upstream origin ブランチ名

※参考URL：https://backlog.com/ja/git-tutorial/stepup/12/

※参考URL：https://qiita.com/ponsuke0531/items/410735b544795506fdc5

自分が変更した内容を把握する　　$ git diff

コミット履歴を確認する　　$ git log

コミット履歴を簡潔に表示する　　$ git log --oneline

コミットグラフを表示する　　$ git log --graph --oneline

変更内容も表示する　　$ git log -p

※表示内容が多い時の表示モードは「↑↓キー」で移動できて「Qキー」で終了できる

特定のコミットからブランチを切りたいとき　　$ git checkout -b ブランチ名 コミット名

※コミット名はリビジョン(頭文字7文字くらい)でも、コミット名でも、HEADを使って指定してもよい

※HEADを使う時･･･HEAD^：HEADの1つ前 / HEAD~3やHEAD^^^：HEADの3つ前

ステージングエリアとHEADをもとに戻す(addを取り消したいときに使う)

　　$ git reset --mixed 戻るコミット

※<戻るコミット>は指定しないとHEAD ※—mixedはなくてもOK

コミットだけを取り消したいとき　　$ git reset —soft 戻るコミット

ワークツリーを含め全て戻す(本当にやらかしたときに自分の作業を開始前に戻す)

　　$ git reset --hard 戻るコミット

※ただ昔の状況を確認したいだけなら　　$ git checkout 戻るコミット

履歴を残したままコミットを戻す(逆向きのコミット)　　$ git revert 戻るコミット

※参考URL：https://qiita.com/kmagai/items/6b4bfe3fddb00769aec4

▼ 初回だけ ▼

一番最初にGitを初期化　　$ git init

※”initialize”の略

ー>”Initialized empty Git repository in パス“と表示される

プッシュ先のリモートリポジトリを登録　　$ git remote add origin <URL>

リモートリポジトリをクローン　　$ git clone <URL>

▲ 初回だけ ▲

※参考URL：https://prog-8.com/docs/git-env

※Windows用：https://prog-8.com/docs/git-env-win

※参考URL：https://qiita.com/devzip8/items/28ac253ea295ad6c2b73

▼ 毎回 ▼

共有するファイルを選択(ステージングに上げる)　$ git add ファイル名

ー>全てのファイルをステージングエリアに上げる　$ git add .

ステージしたファイルを取り消す　　$ git reset ファイル名

ステージングエリアの状態を確認する　　$ git status

※Changes to be comittedはHEADの状態とステージングエリアの状態の差

コミット(記録)する　　$ git commit -m “コミットメッセージ”

プッシュ(アップロード)する　　$ git push origin ブランチ名

プル(ダウンロード)する　　$ git pull origin ブランチ名

※参考URL：https://qiita.com/sayama0402/items/9afbb519d97327b9f05c

ブランチを作成する　　$ git branch ブランチ名

ブランチを確認する　　$ git branch

ブランチを切り替える　　$ git checkout ブランチ名

ー>ブランチの作成と切り替え　　$ git checkout -b ブランチ名

マージ済みローカルブランチを削除する　　$ git branch -d ブランチ名

ローカルブランチを削除する(マージ前も可)　　$ git branch -D ブランチ名

リモートブランチを削除する　　$ git push --delete origin ブランチ名

もしくは　　$ git push origin :ブランチ名

他人が削除したブランチを反映する　　$ git fetch -p

※参考URL：https://backlog.com/ja/git-tutorial/stepup/07/

ブランチ名を変更する　　$ git branch -m hoge fuga

変更したブランチは普通にpushする

※参考URL：https://qiita.com/hogeta_/items/e47dfb0cf88270ef2802

当然ブランチを切り替えかえる前にcommitしろと怒られる！　　※ ▼“save”以下は省略可

ー>変更を退避する　　$ git stash save “メッセージ”

退避した作業の一覧を見る　　$ git stash list

退避した作業をブランチに戻す　　$ git stash apply stash@{番号}

※stash名を指定しないと、stash@{0}をブランチに戻す

※addしていた変更もaddされていないものとして戻される、防ぎたいなら“--index”オプション

退避した作業を消す　　$ git stash drop stash@{番号}

退避した作業を戻し消す　　$ git stash pop stash@{番号}

※参考URL：https://qiita.com/chihiro/items/f373873d5c2dfbd03250

マージするときは･･･まずマージされる(受け入れる側の)ブランチに移動する

ー>ブランチをマージする　　$ git merge 取り込まれるブランチ名

衝突したら･･･修正する

ー>ステージングに上げて　　$ git add .

ー>コミットする　　$ git commit -m “コミットメッセージ”

エラー：fatal: The current branch branch-a has no upstream branch.

ー>上流ブランチがない！

ー>pushで上流ブランチに追加する　　$ git push --set-upstream origin ブランチ名

※参考URL：https://backlog.com/ja/git-tutorial/stepup/12/

※参考URL：https://qiita.com/ponsuke0531/items/410735b544795506fdc5

自分が変更した内容を把握する　　$ git diff

コミット履歴を確認する　　$ git log

コミット履歴を簡潔に表示する　　$ git log --oneline

コミットグラフを表示する　　$ git log --graph --oneline

変更内容も表示する　　$ git log -p

※表示内容が多い時の表示モードは「↑↓キー」で移動できて「Qキー」で終了できる

特定のコミットからブランチを切りたいとき　　$ git checkout -b ブランチ名 コミット名

※コミット名はリビジョン(頭文字7文字くらい)でも、コミット名でも、HEADを使って指定してもよい

※HEADを使う時･･･HEAD^：HEADの1つ前 / HEAD~3やHEAD^^^：HEADの3つ前

ステージングエリアとHEADをもとに戻す(addを取り消したいときに使う)

$ git reset --mixed 戻るコミット

※<戻るコミット>は指定しないとHEAD ※—mixedはなくてもOK

コミットだけを取り消したいとき　　$ git reset —soft 戻るコミット

ワークツリーを含め全て戻す(本当にやらかしたときに自分の作業を開始前に戻す)

$ git reset --hard 戻るコミット

※ただ昔の状況を確認したいだけなら　　$ git checkout 戻るコミット

履歴を残したままコミットを戻す(逆向きのコミット)　　$ git revert 戻るコミット