Git勉強会

2014/02/05

アンケート

* バリバリ使える
* 使ったことがある
* 名前は知っている
* 何それ？美味しいの？

001

目次

* バージョン管理
* Git
* 環境構築
* まずは、ローカルにて
* そして、リモートへ
* Tips

予　　告

バージョン管理

* ファイルの変更履歴を管理すること

バージョン管理

* やり方は色々ありますよね
* オレオレバージョン管理
* 人力バージョン管理ともいう

オレオレバージョン管理

その1

オレオレバージョン管理 その1

* module.java

オレオレバージョン管理 その1

* module.java
* module_backup.java

オレオレバージョン管理 その1

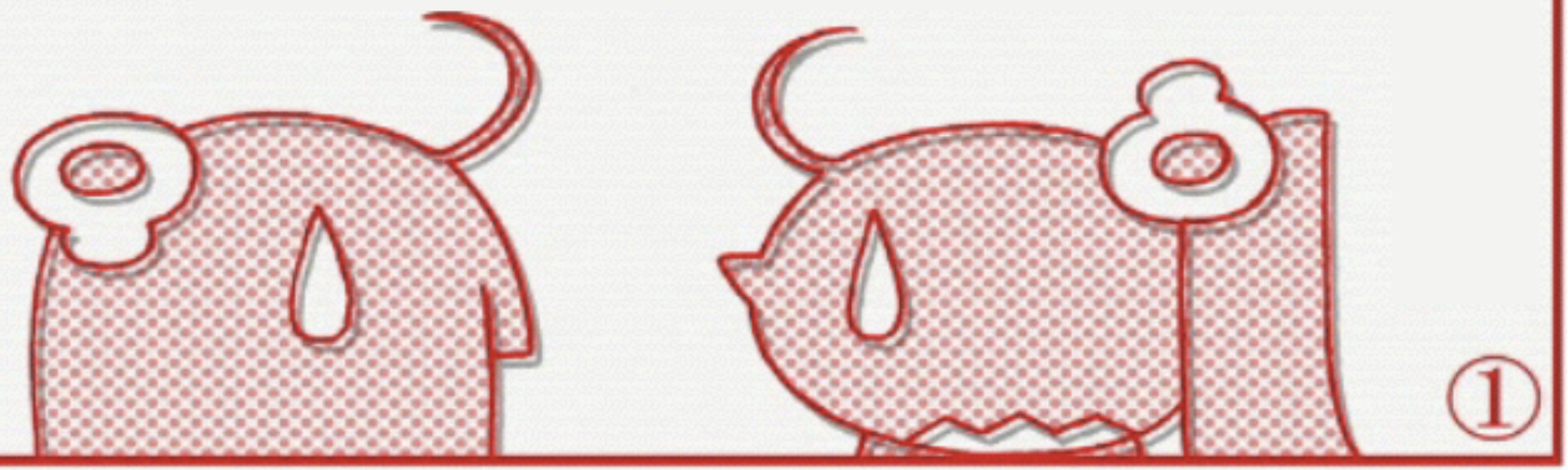
* module.java
* module_backup.java
* module_ver1.java

オレオレバージョン管理 その1

* module.java
* module_backup.java
* module_ver1.java
* module_ver2.java

豫　　告

何モシナイヨリハ…



①

オレオレバージョン管理

その2

オレオレバージョン管理 その2

```
/* 2014-02-04 追加 */
printf("hogehoge");
```

オレオレバージョン管理 その2

```
/* 2014-02-04 追加 */
/* 2014-02-05 コメントアウト */
//printf("hogehoge");
```

オレオレバージョン管理 その2

```
/* 2014-02-04 追加 */
/* 2014-02-05 コメントアウト */
//printf("hogehoge");

/* 2014-02-05 追加 ここから */
for (int i = 0; i < 10; i++) {
    printf("fugafuga");
}
/* 2014-02-05 ここまで */
```

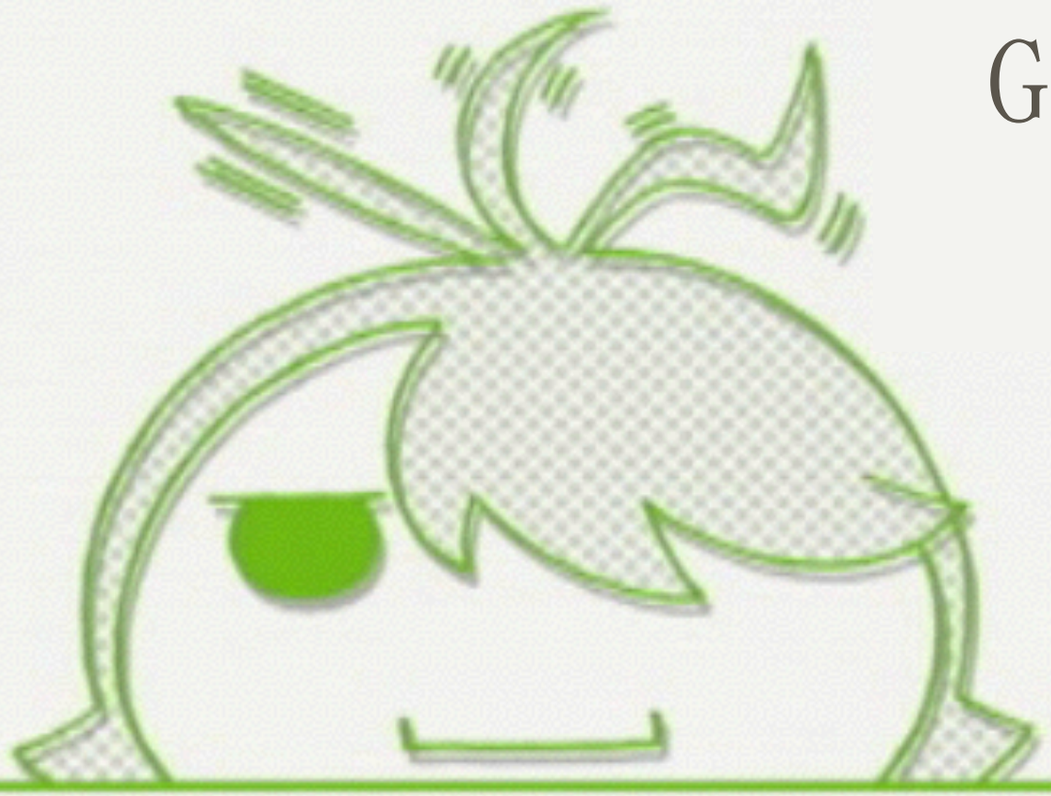
豫　　告

もうちょっと

マトモな

バージョン管理をしよう

豫　　告



Gitヲ使オウ！

①

002

目次

* バージョン管理
* Git
* 環境構築
* まずは、ローカルにて
* そして、リモートへ
* Tips

Git

* バージョン管理システムの1つ
* リポジトリで履歴を管理
* 分散型

リポジトリ

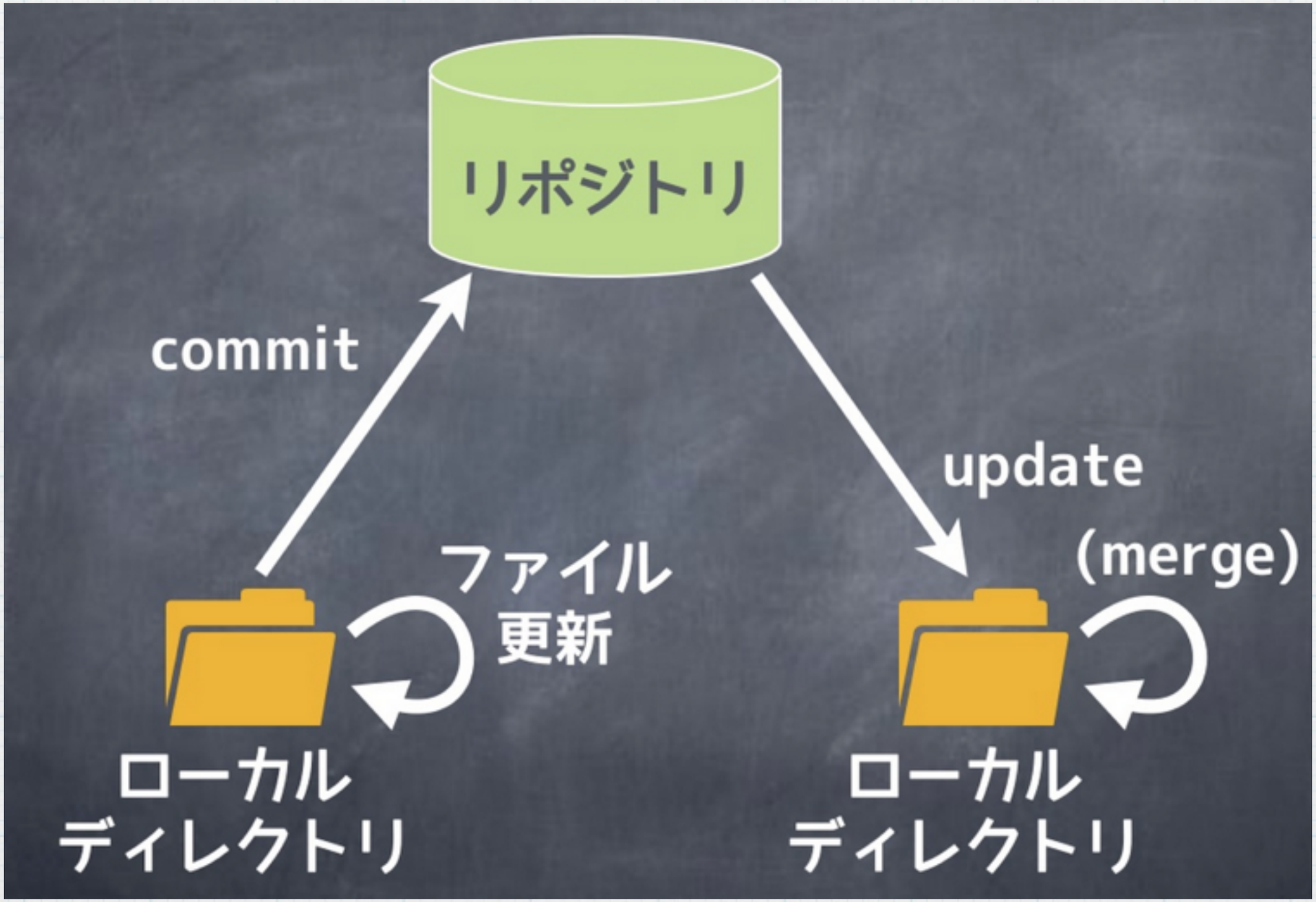
* Gitが管理する履歴情報

* 履歴そのものと考えてOK

分散型と集中型

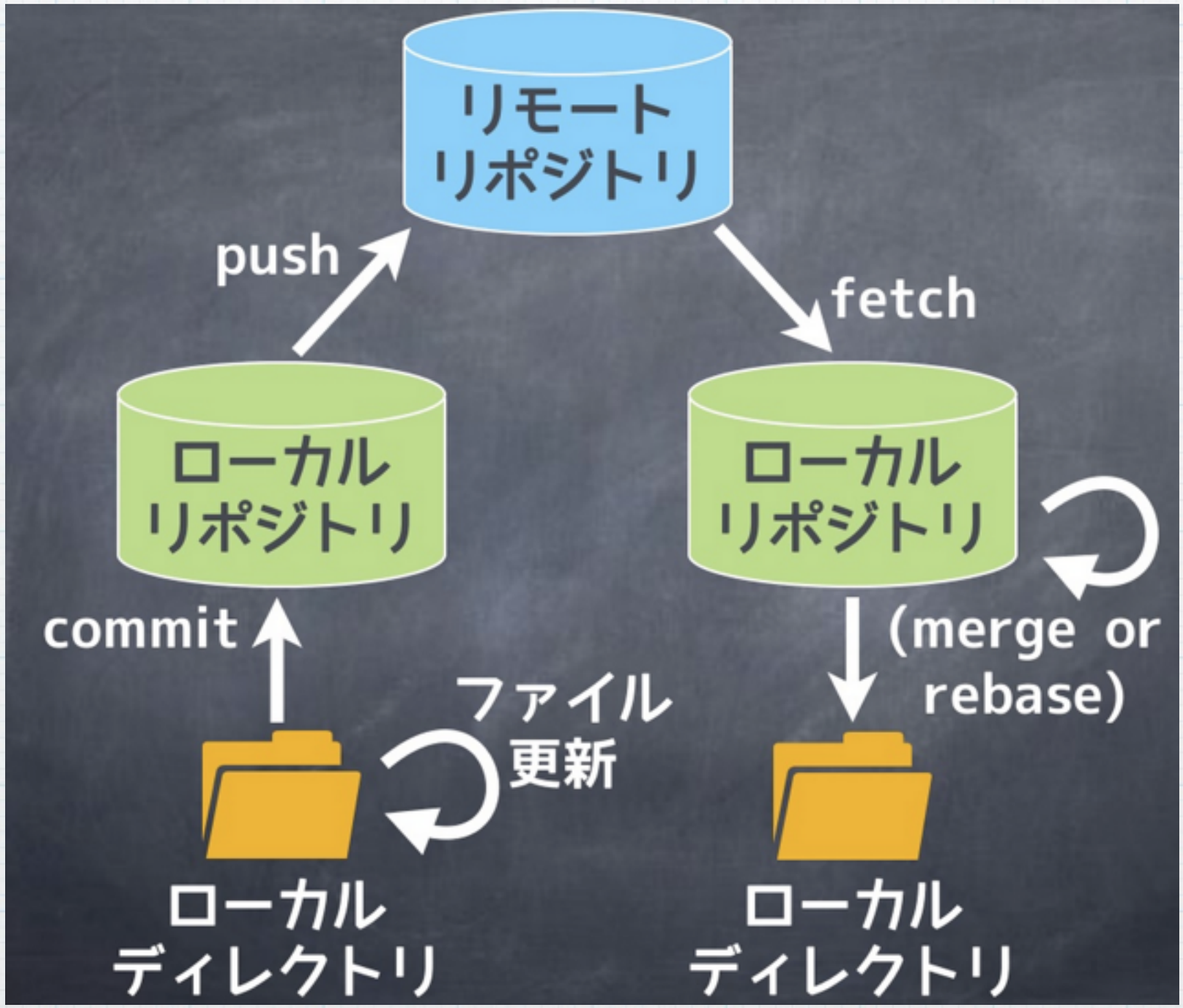
集中型	CVS Subversion
分散型	Git Mercurial

集中型(Subversion)



http://www.slideshare.net/matsukaz/git-17499005

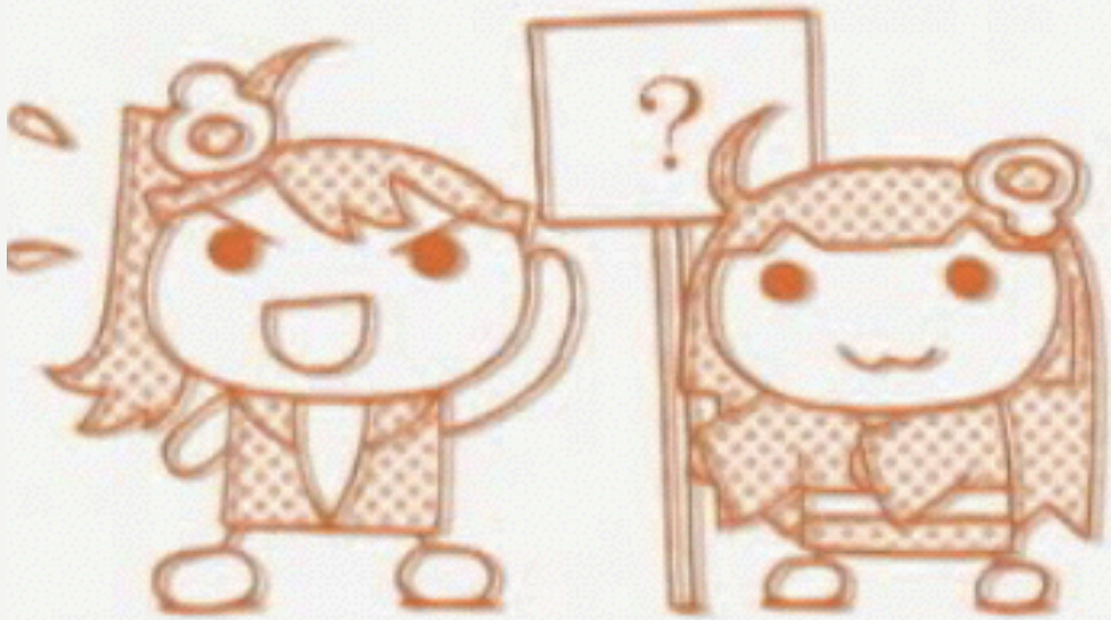
分散型(Git)



http://www.slideshare.net/matsukaz/git-17499005

豫　　告

ドッチガ
良インダ？

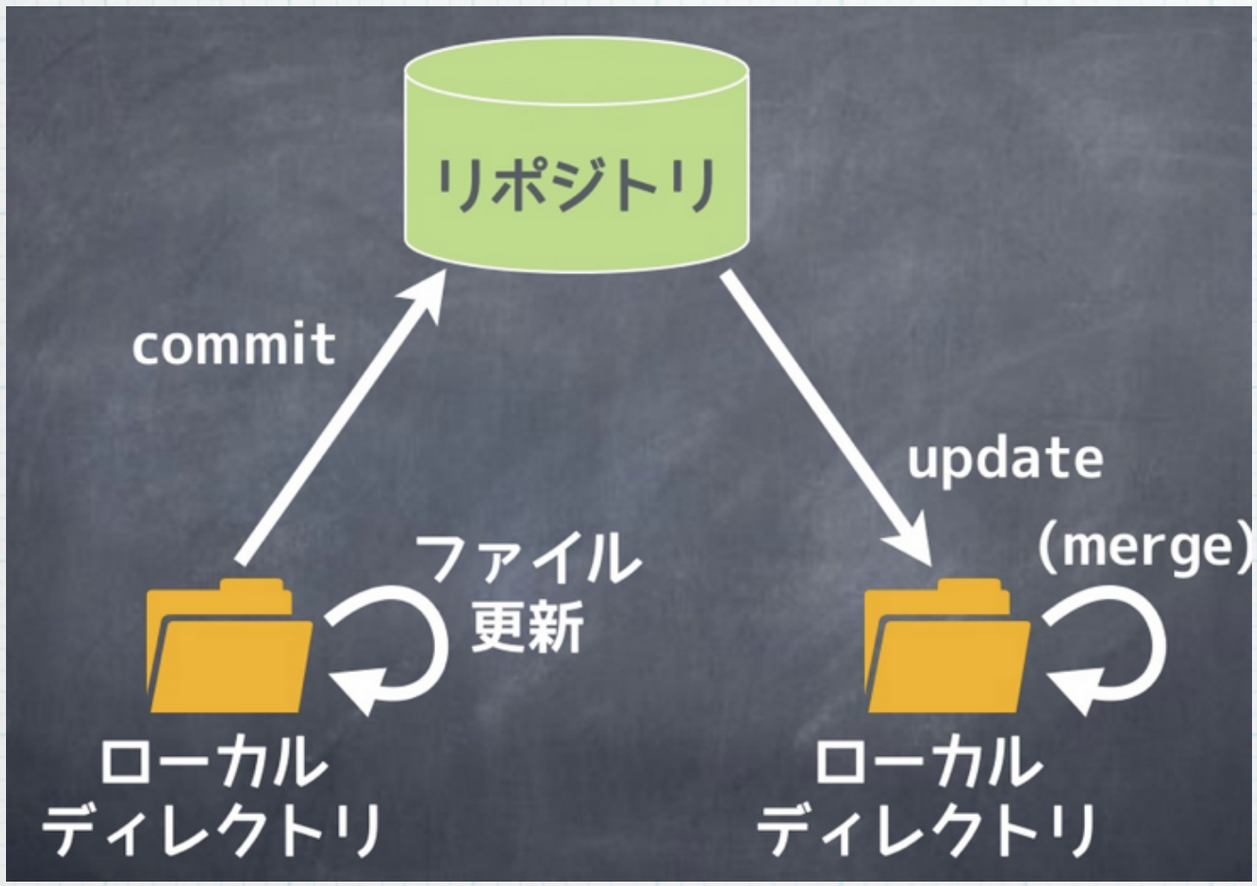


②

NISIOISIN

* どちらかが全てにおいて優れているわけではない
* ただ、最近はGitが流行っている

集中型

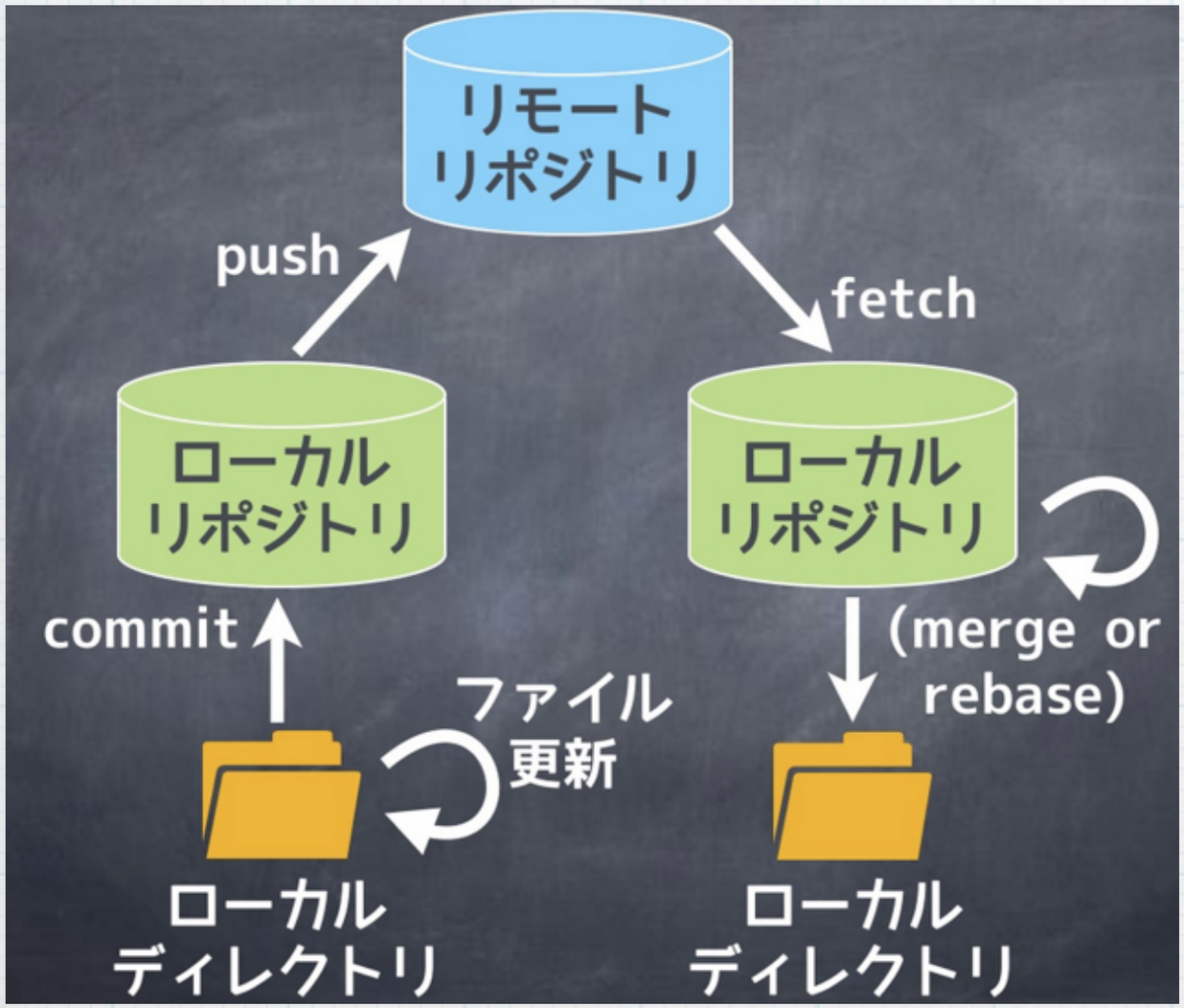


commit=
共有リポジトリ更新
↓
一度コミットすると、
取り消しが不可能

commit, updateは
オンライン必須

http://www.slideshare.net/matsukaz/git-17499005

分散型



commit=
ローカルリポジトリ更新
↓
黒歴史の隠蔽が可能

基本的には
オフラインで動作可能

http://www.slideshare.net/matsukaz/git-17499005

“新しい用語が次々と出てきて
混乱してきましたか？”

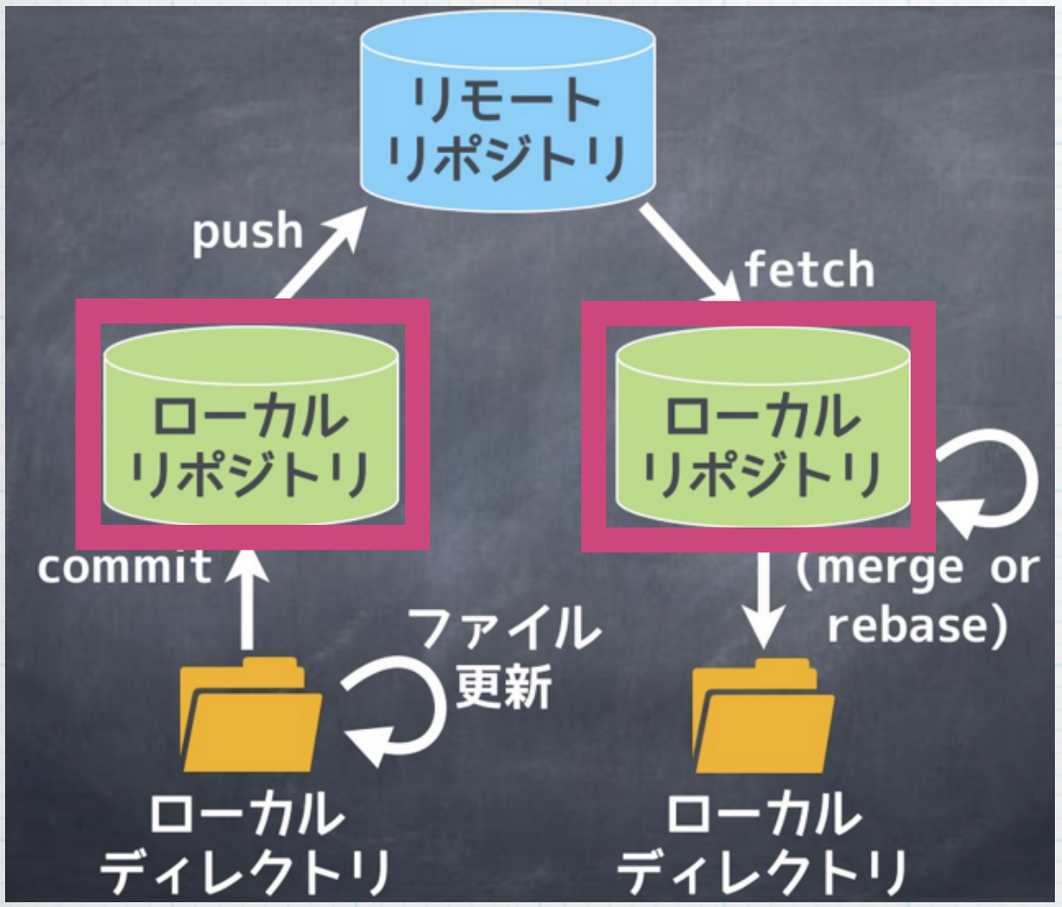
ここで一旦

用語を整理

重要用語一覧

* リポジトリ
* commit/push
* fetch/merge

リポジトリ

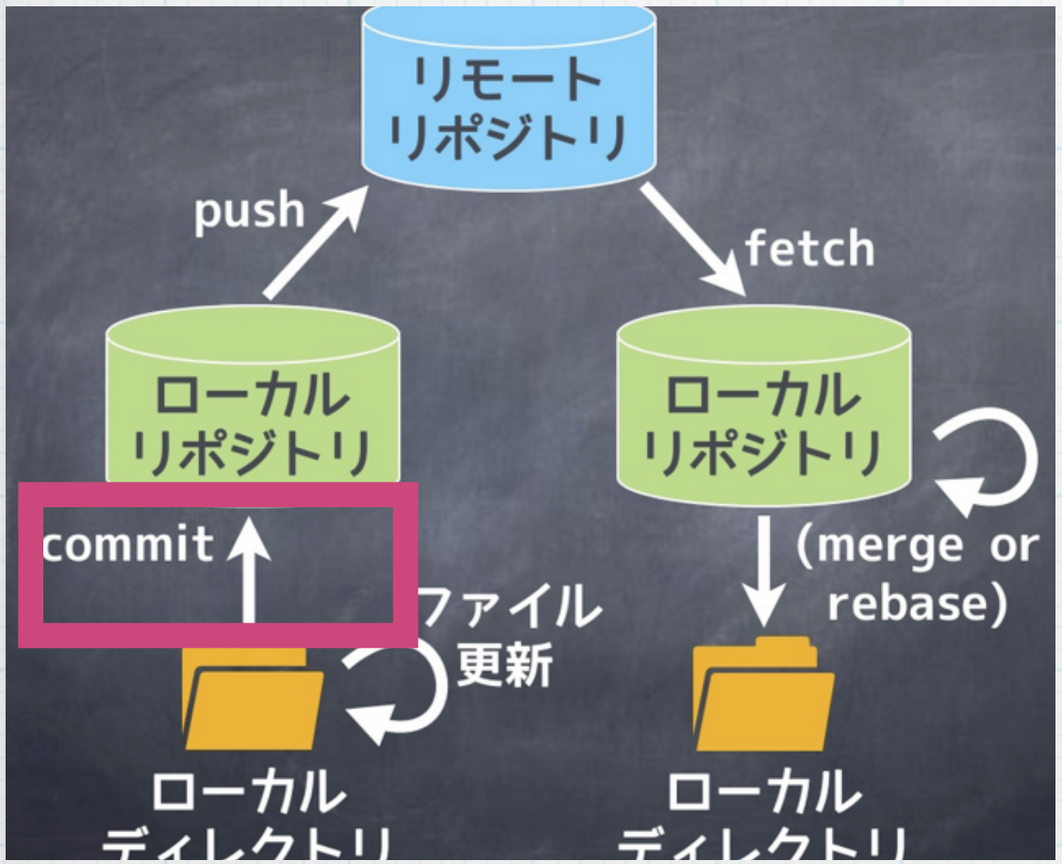


* Gitが管理する履歴情報

* 履歴そのものと考えてOK

http://www.slideshare.net/matsukaz/git-17499005

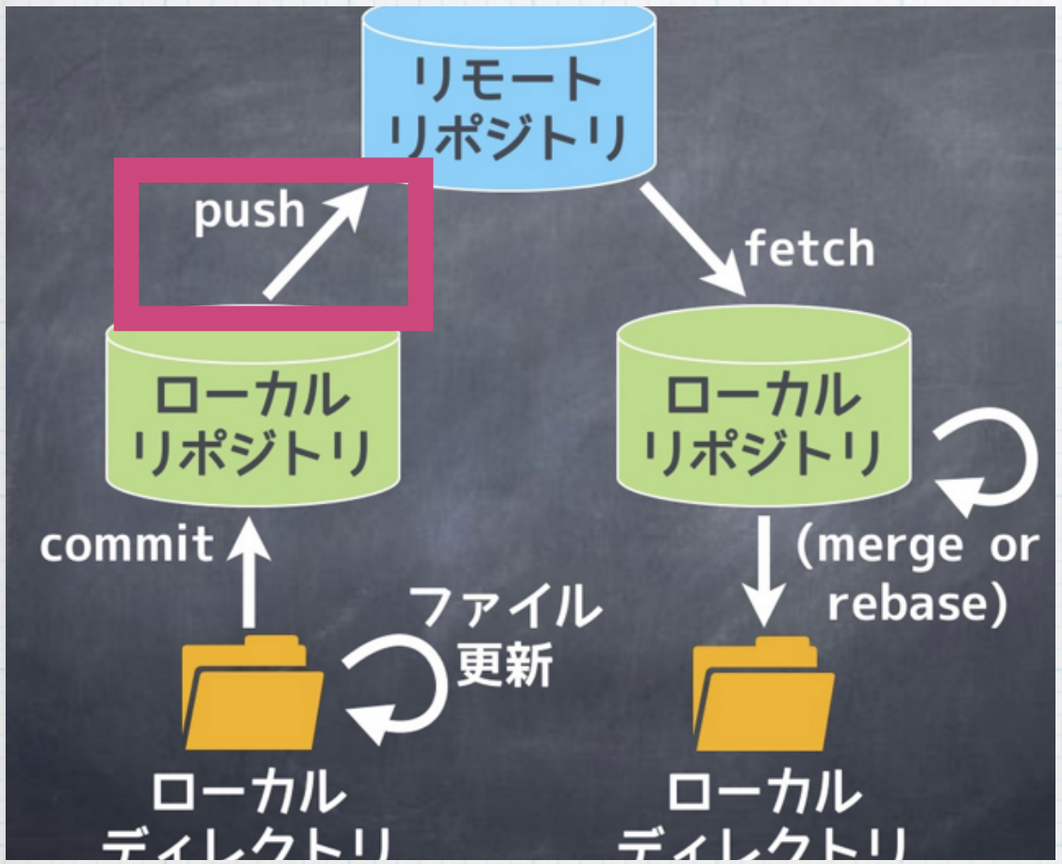
Commit



* ファイルの変更をローカルリポジトリに記録
* あくまでローカルに記録

http://www.slideshare.net/matsukaz/git-17499005

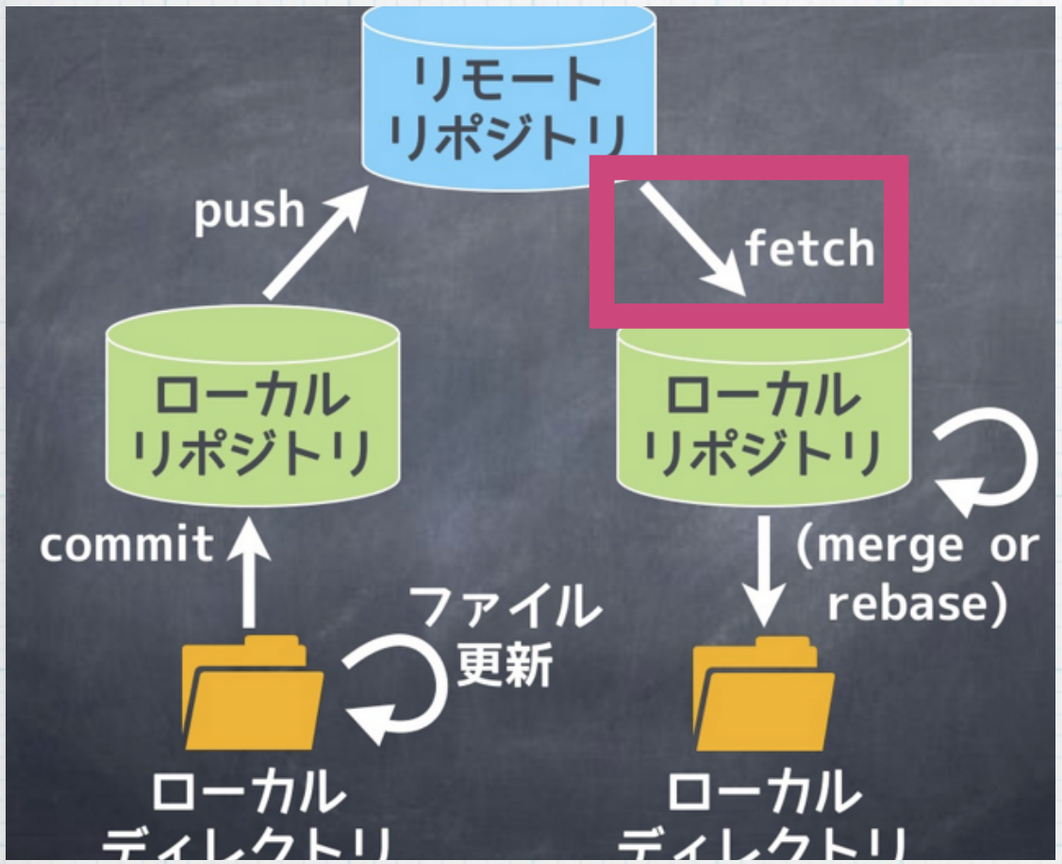
Push



* ローカルの変更をリモートに送る

http://www.slideshare.net/matsukaz/git-17499005

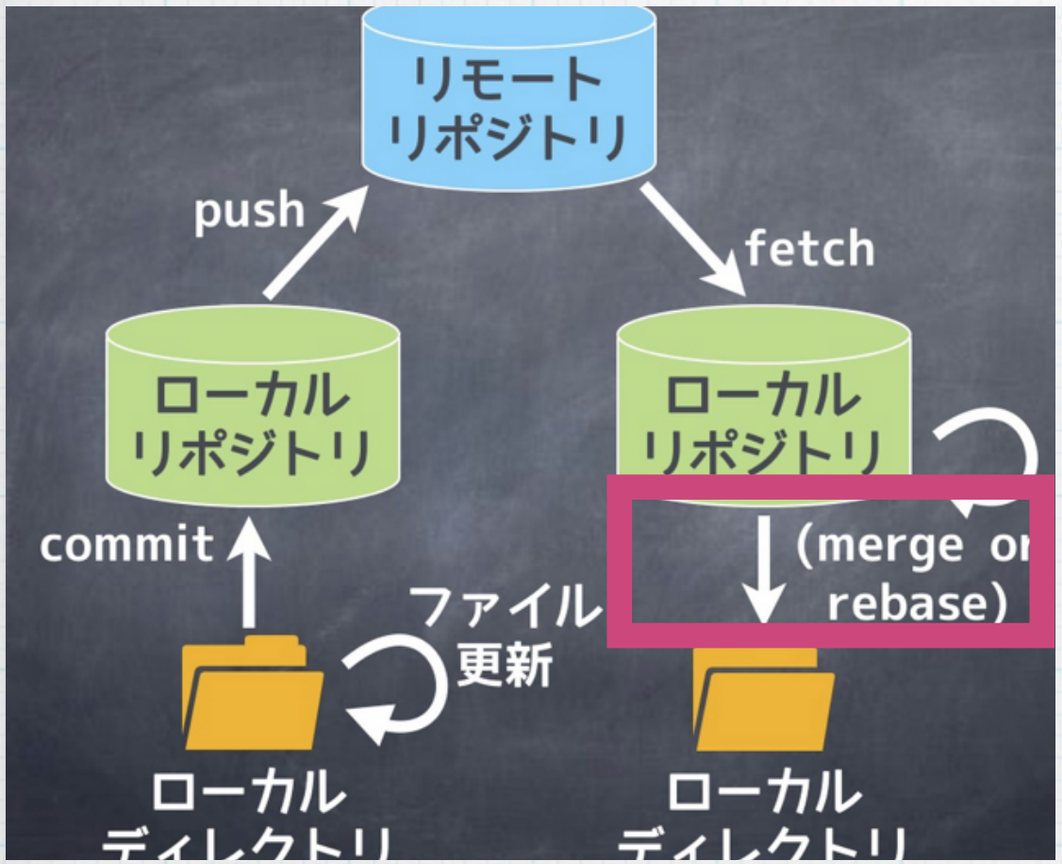
Fetch



* リモートの変更を取得
* 取得するだけで、ローカルディレクトリには反映させない

http://www.slideshare.net/matsukaz/git-17499005

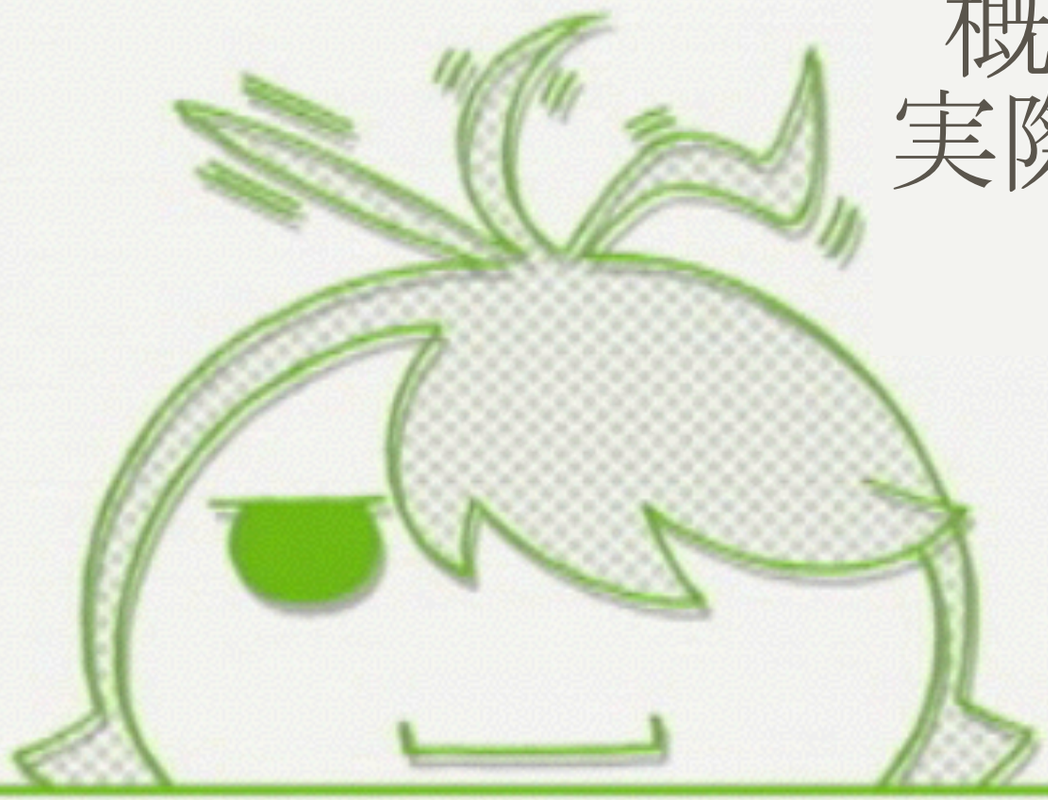
Merge



* リモートからfetchした変更をローカルディレクトリに反映させる

http://www.slideshare.net/matsukaz/git-17499005

豫　　告



概要ハ分カッタ？
実際二使ッテミヨウ

①

003

目次

* バージョン管理
* Git
* 環境構築
* まずは、ローカルにて
* そして、リモートへ
* Tips

Mac

* App StoreからXcodeをインストール
* Xcodeからコマンドラインツールをインストール

Windows

* Cygwinを使います
* 情科貸与PCであれば、既にCygwinでGitが使えるはず

004

目次

* バージョン管理
* Git
* 環境構築
* まずは、ローカルにて
* そして、リモートへ
* Tips

準備

* バージョン管理の対象となるプロジェクトを作り、リポジトリの初期化を行います

* 今回は適当にディレクトリを作成

* Cygwinを起動
* mkdir hoge
* cd hoge/
* git init

* git config —-global user.email “Githubに登録したメールアドレス”

* git config —-global user.name “自分の名前”

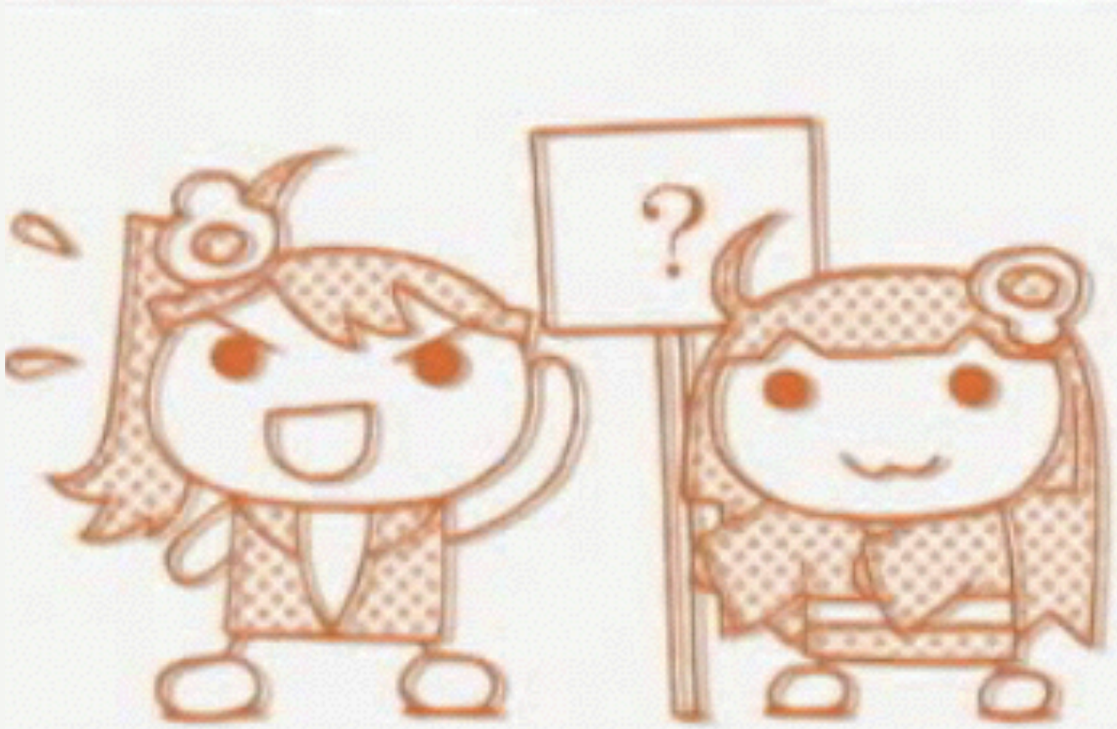
* git config --global color.ui auto

初コミット

* touch hoge.txt
* git add .
* git commit -m “Initial commit”

豫　　告

チョット待ッタ
Addッテ何ダ？



②

NISIOISIN

Commit/Add

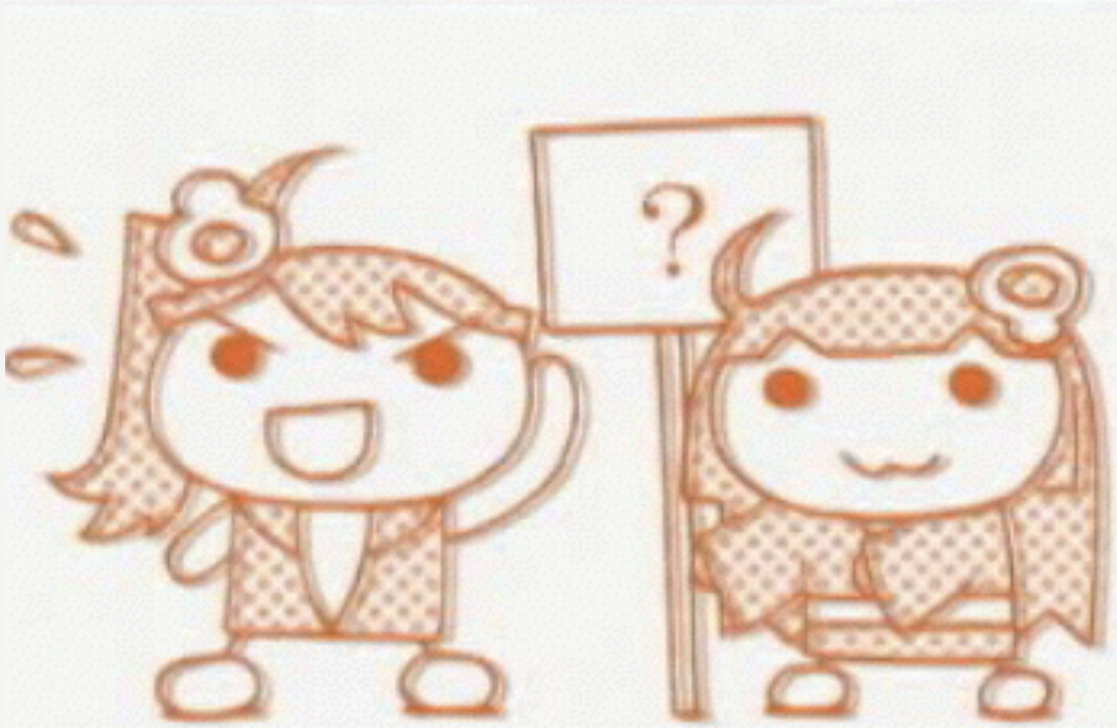
* ファイルを変更しただけでは、そのファイルはCommit対象とならない

* 変更したファイルをAddすることで始めてCommit対象となる

* 「git add .」はカレントディレクトリの全てのファイルをCommit対象にする

豫　　告

Commitノ後ノ
変ナノハ？



②

NISIOISIN

コミットメッセージ

* コミット内容を要約したもの
* 出来るだけ分かりやすく、簡潔に
* これがしっかり書けると格好いい

さっきは

単純にファイルを

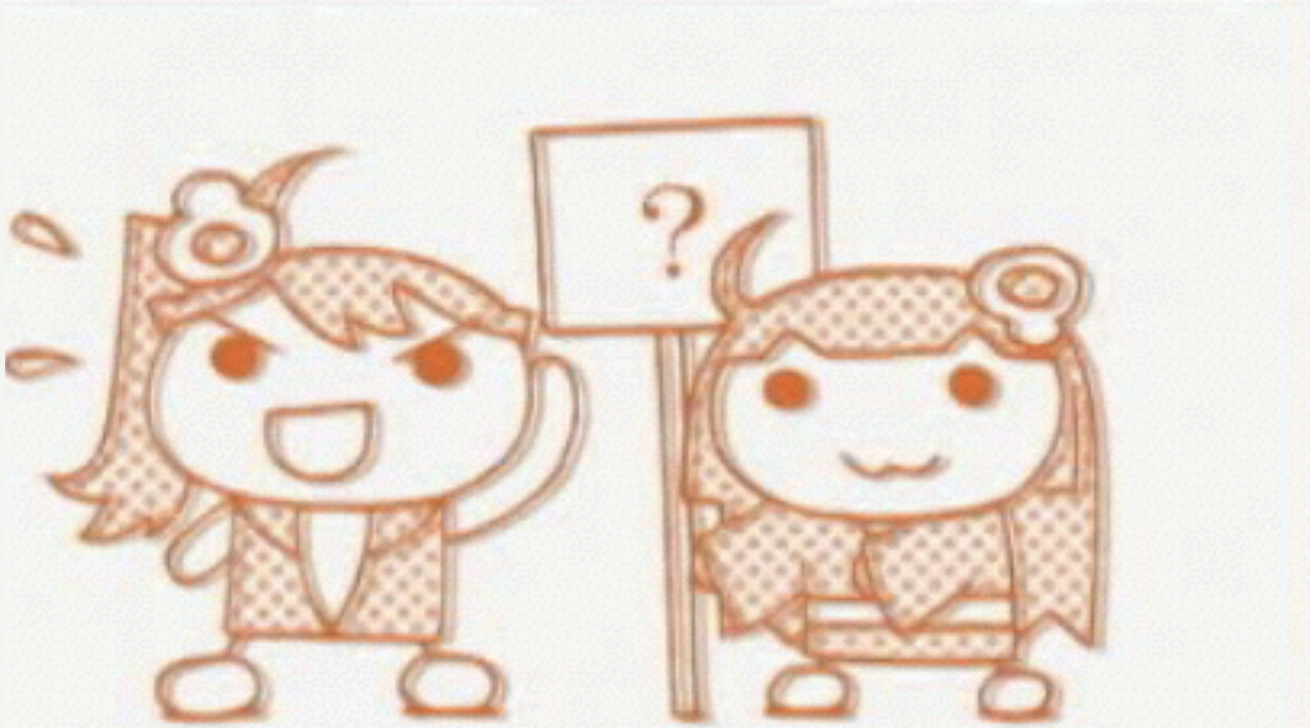
追加しただけ

内容を編集してみる

* 適当にファイルを編集
* `git add .`
* `git commit -m “Edit hoge.txt”`

豫　　告

履歴ヲ確認シタイ



②

NISIOISIN

履歴確認

* `git log`
* これでコミット履歴を確認できます

バージョン管理っぽい

ことをしてみよう

以前の状態に戻す

* 「git reset」というコマンドを使います
* git logで戻りたいコミットのハッシュをコピーします
* git reset コミットのハッシュ
* 実例

実はresetには

2種類あります

Soft/Hardリセット

* Softリセットはコミットを無かったことにするだけで、編集内容は残る

* Hardリセットはコミットを無かったことにし、編集内容も消し去る

まあ、実はHardリセットしても復元出来るけどね…
黒歴史を完全に消し去ることは出来ないのです

豫　　告

コミットメッセージ
ヲ間違エタ…

* `git commit –-amend`
* コミットメッセージを修正
* これで修正完了

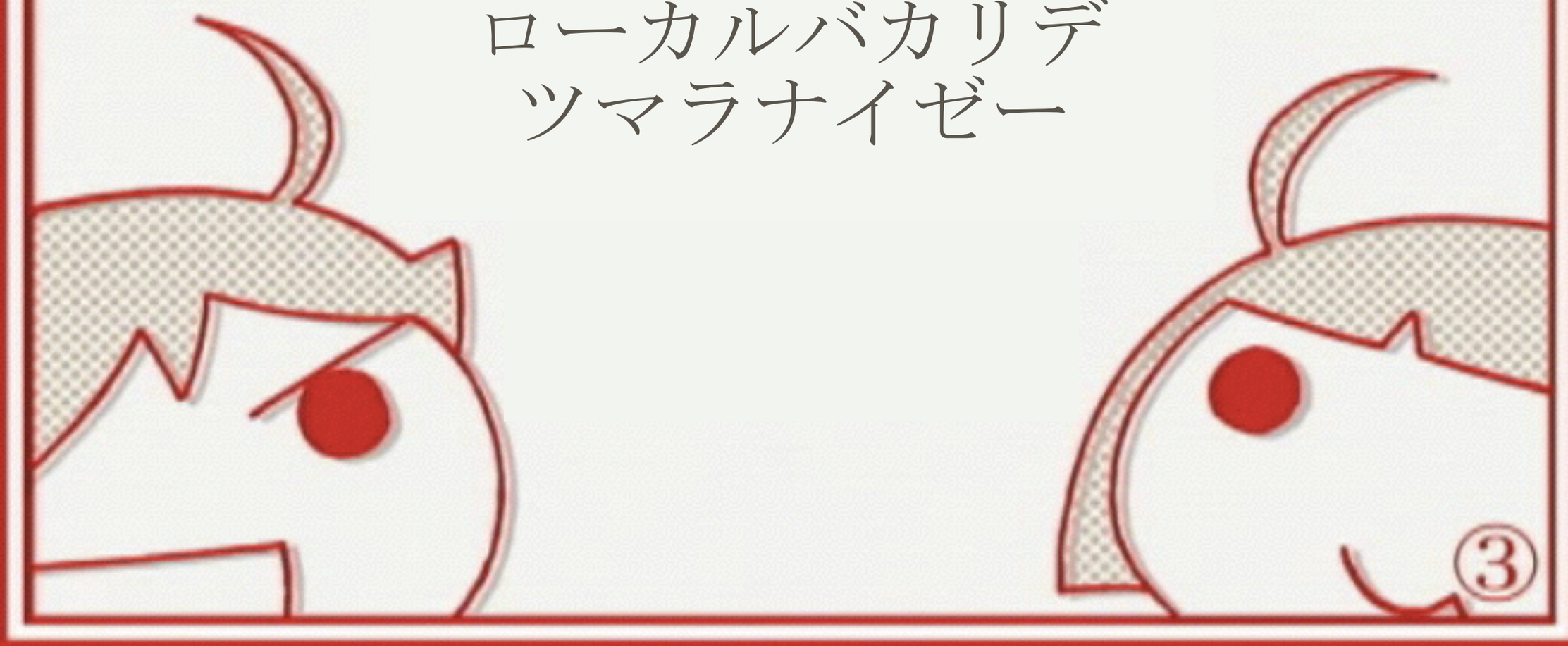
“エディタから抜けられなくなった？”

* vi/vimはモードという概念を持つ
* a/i/o: 編集モード
* ESC: コマンドモード
* コマンドモードで「:wq」で終了
* Emacsにしたい人はconfigを編集

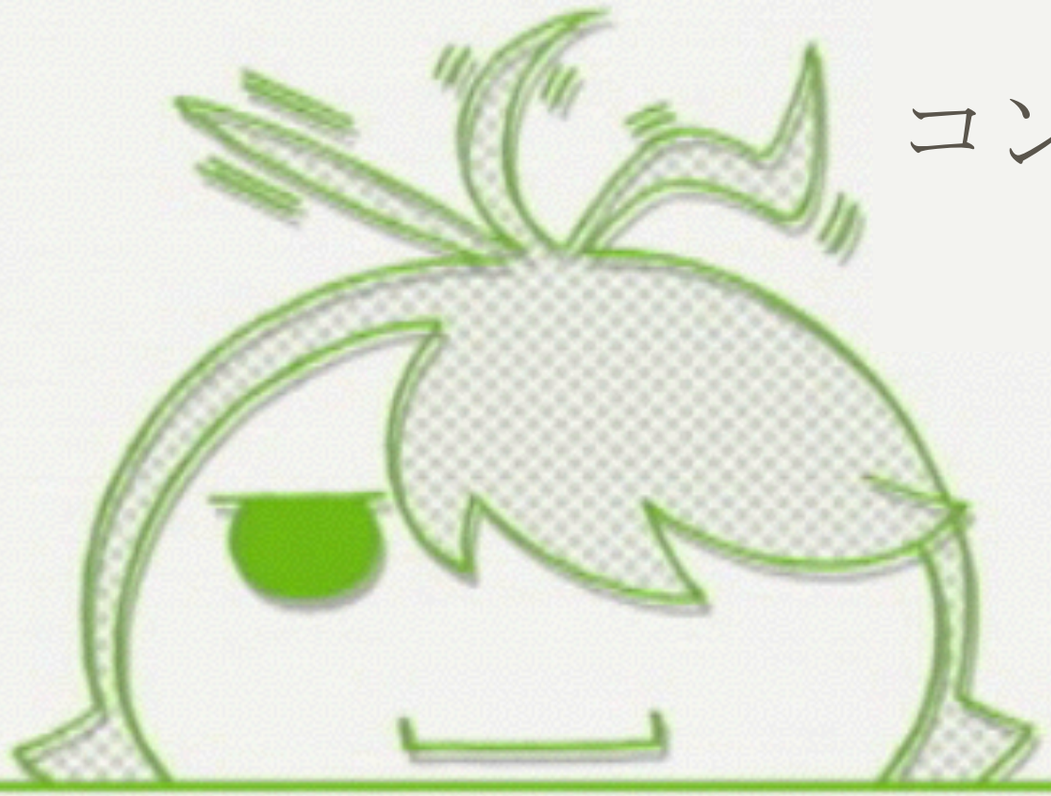
目次

* バージョン管理
* Git
* 環境構築
* まずは、ローカルにて
* そして、リモートへ
* Tips

豫　　告



豫　　告



Gitノ真ノ力ハ
コンナモンジャナイゼ

①

リモート連携してこそ

Git

Github

* Gitリポジトリのホスティングサービス
* ソーシャルコーディング
* 他には、Bitbucket, Backlogなど
* まずは、アカウントを取得しましょう

Clone

* リモートリポジトリをコピーして、ローカルに保存すること

* git clone git@github.com:autohn/GitExercise.git

練習

* 自分の名前を付けたファイルを作成し、適当に編集し、コミットしてください

* 簡単ですね

ついに

リモート連携

* `git push origin master`
* 上記のコマンドは、リモートの`origin`リポジトリの`master`ブランチに自分の変更を送るという意味

先のコマンドで
リモートリポジトリに
皆さんの変更が反映されました

ブラウザから

確認してみる

他の人の変更を
自分のリポジトリに
取り込んでみる

* git fetch
* git merge origin/master
* まずは、リモートの変更をfetch、
  次に取得した変更をローカルに
  mergeします
* さっき説明した手順です

* 実は、fetchとmergeを同時に行なってくれる便利なコマンドがあります

* git pull

先のコマンドで
他の人の変更が
自分のリポジトリに
取り込まれたはず

予告

全員ガ同ジファイルヲ
編集シタラドウナル？

変更が衝突します

コンフリクト(変更の衝突)

* gitはファイルをロックしません
* 集中型では、厳格にロックするものもある？
* コンフリクトはあんまり起こらないだろうというスタンス
* コンフリクトしたら手動で修正
* 実演

目次

* バージョン管理
* Git
* 環境構築
* まずは、ローカルにて
* そして、リモートへ
* Tips

なんとなく
Gitの使い方が
分かりましたか？

最後にちょっと
便利な小技を
紹介します

git status

* 作業ツリーとローカルリポジトリの変更点が確認出来ます

* 実演

git diff

* diffが見れてこそバージョン管理
* 作業ツリーとローカルリポジトリの差分が確認出来ます
* diffはGUIツール使った方が見やすいかも
* 実演

git stash

* 作業ツリーの状態を一時保存
* 保存: `git stash`
* 復元: `git stash pop`
* 実演