GitHub

GitHubのインストール

GitHubの準備

バージョン確認：

$ git —version

Ubuntuの場合のインストール：

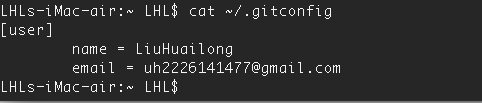
$ sudo apt-get install git-core

GitHubの初期設定：

$ git config —global user.name ‘xxxx’

$ git config —global user.email ‘[xxxxx@xxx.com](mailto:xxxxx@xxx.com)'

GitHubの設定確認：

$ cat ~/.gitconfig

リポジトリの作成

--------リポジトリとは：Gitがファイルの履歴を保存している場所です。

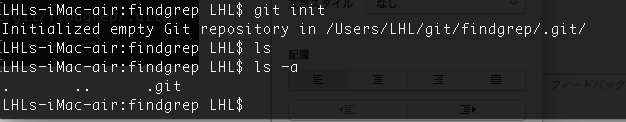
まず初めに、作業を行うための新しいディレクトリを作成します。ディレクトリ名はなんでもいいですが、ここでは「findgrep」という名前にします。

新しいディレクトリを作成：

$ mkdir -p ~/git/findgrep

初期化（.gitファイルの作成）：

$ cd ~/git/findgrep

$ git init

git initを実行すると、.gitというディレクトリが作成されます。

この.gitディレクトリはGitのリポジトリの実体です。

！！！.gitファイルを削除すると大変なことが起こります。

（自分の小宇宙が滅びるぐらい）！！！

ワークツリー

———-リポジトリの内容をファイルとして展開する場所

ワークツリーでは普通のファイルを編集したり、追加・削除したりするところです。

リポジトリにファイルを追加

（以下では全部~/git/findgrepディレクトリに実行する）

バージョン管理するファイルを作成する：

$ touch findgrep.sh

$ chmod 755 findgrep.sh

$ vi findgrep.sh

findgrep.shファイルに以下のコードを書く

#!/bin/bash

pattern=$1

find . -type f | xargs grep -nH “$pattern”

このファイルをGitのリポジトリに履歴として追加してみましょう。

リポジトリに追加するには：

仮の変更

git add

git addは：

どのファイルをリポジトリに履歴として追加するのかを指定するコマンドです。

git commit

git commitは：

実際にリポジトリにファイルの変更履歴を追加する

本物の変更

今の時点では、findgrep.shファイルまだバージョン管理の対象になっていないため。

リポジトリに履歴として追加する：

$ git add findgrep.sh

続いて、git commitを実行して、実際の履歴に反映する

なお、git commitする際には、次のように-mオプションを指定して、今回の修正に対するメッセージを入力する：

$ git commit -m ‘findgrep.sh新規作成’

差分の表示と再コミット

これから、別の修正を行ってもう一度commitしてみましょう。

findgrep.shファイルを開いて、内容を次のように変更。

#!/bin/bash

pattern=$1

directory=$2

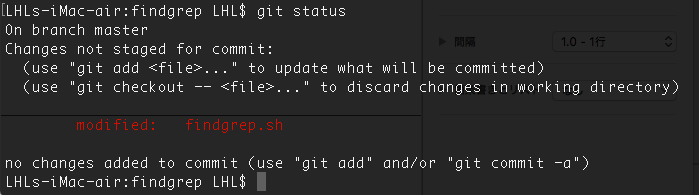
if [ -z "$directory" ]; then

directory='.'

fi

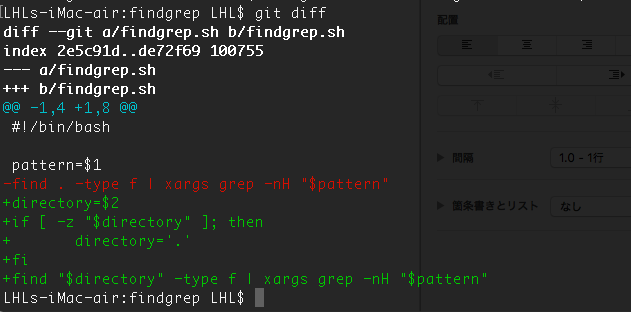
find "$directory" -type f | xargs grep -nH "$pattern"

git status

——-現在のワークツリーの状態を表示するためのコマンド

6行目に「modified:」と表示したのはファイルの差分

git diff

——-文字どおり、変更した内容をチェックする

こういうの変更のチェックをお勧めします。

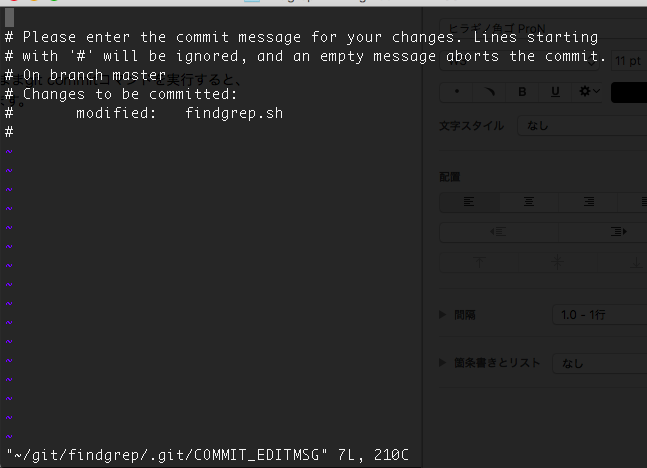
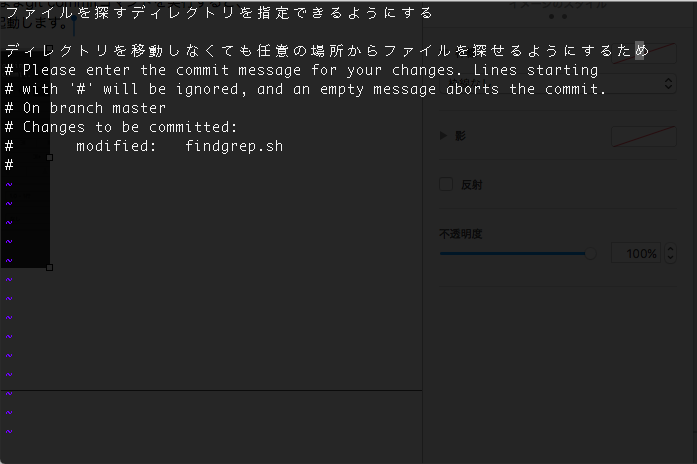
Gitでは新規作成でも編集でも手順は変わりません。

ここで、もう一度コミットしましょう。

$ git add findgrep.sh

$ git commit

今回では-mオプションを指定せず、そのままgit commitコマンドを実行すると、

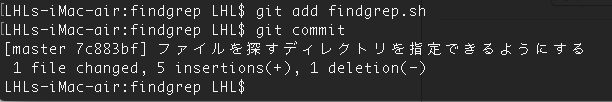
特に設定しない場合は、Vimエディタが起動します。

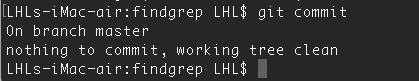
1行目 ：変更の概要

2行目 ：空行

3行目 ：詳細メッセージ

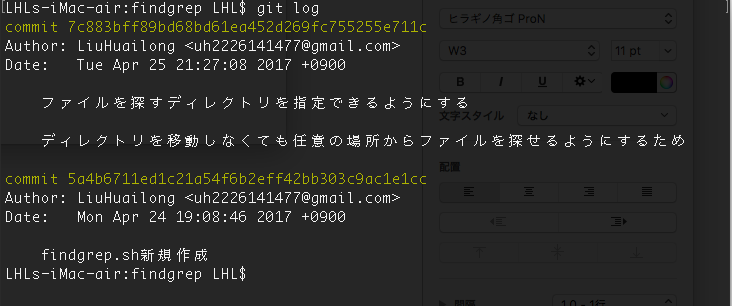
こんなコメントの習慣を慣れましょう！！！

（:wqでVimを終了する）

変更がない場合はgit commitを実行したら、

このようなメッセージが表示されます。

git log

——-変更の履歴を確認する

今までは２回commitを行ったため、↑のように２つcommitが表示しています。

git log -pオプションを指定すると、commitごとの差分も合わせて表示されます。

（各自やってみましょう！！）

$ git log -p

git diff 7c883bfのような特定のcommitとの違いを確認することができます。

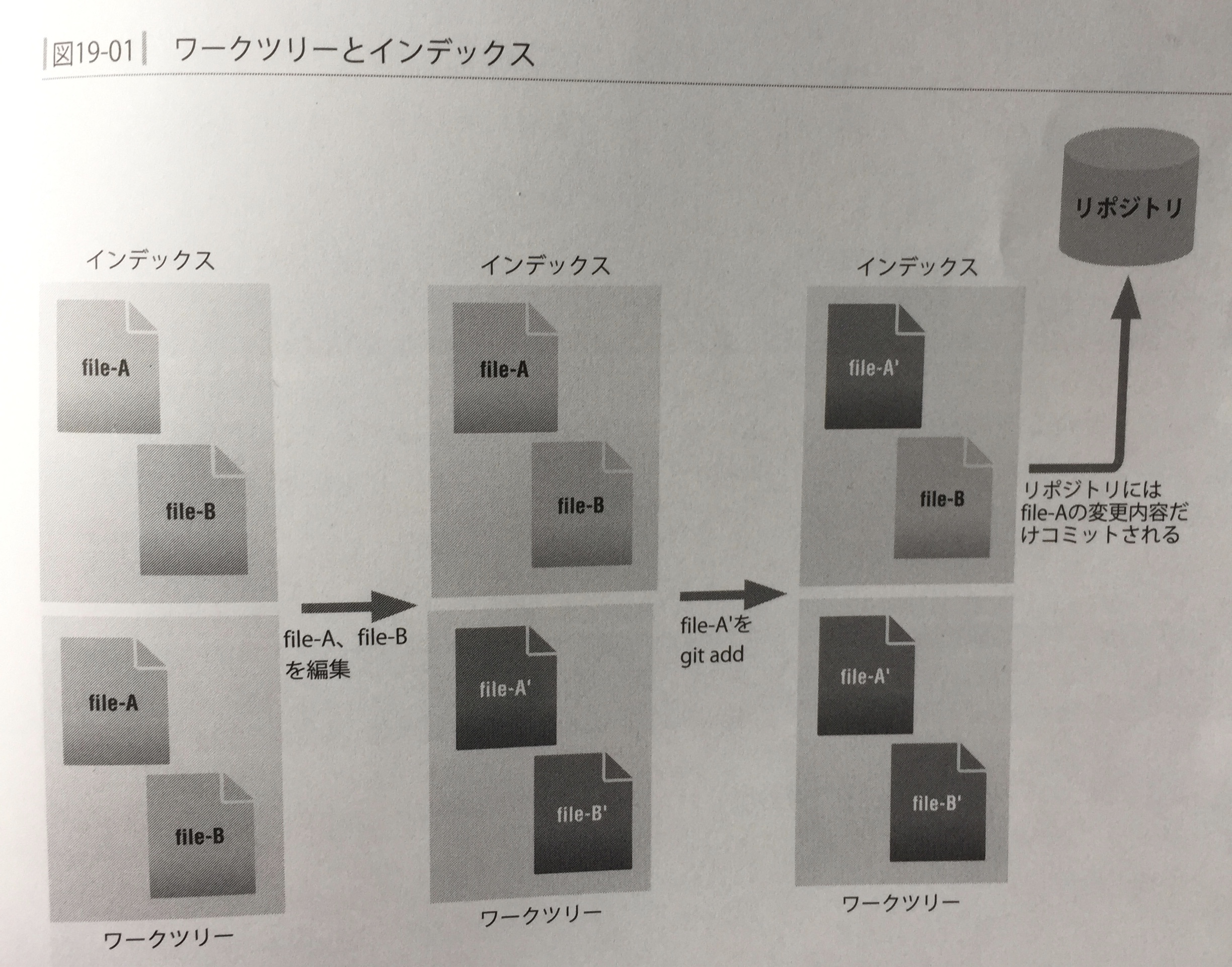
（各自やってみましょう！！）

ワークツリーとインデックス

Gitでは、ワークツリーにあるファイルを直接コミットしてリポジトリに反映させるのではなく、いったんインデックスと呼ばれる領域に配置する

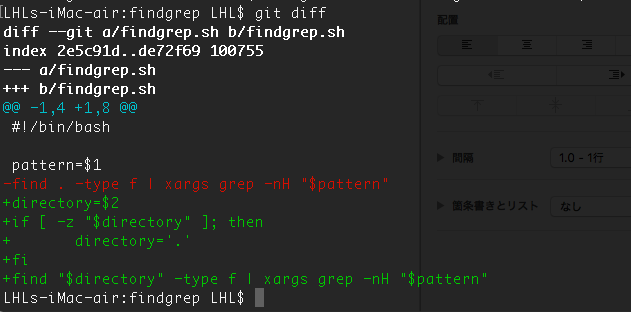
インデックスとは

コミットの前にワークツリーのファイルを一時的に配置するための領域です。そして、ワークツリーからインデックスにファイルを登録するためのコマンドがgit addです。



git commitコマンドを実行するとインデックスに登録されている内容がリポジトリに登録されます。

ワークツリー、インデックス、リポジトリと三つの領域の違いをはっきりさせるために、git diffコマンドでそれぞれの差分を確認してみましょう。

まず、git diffコマンドは、特に何も指定しなければワークツリーとインデックスの差分を表示します。

ここで、「abc.txt」というファイルを追加し、git addをしてみたいと思います。

$ git touch abc.txt

$ vi abc.txt

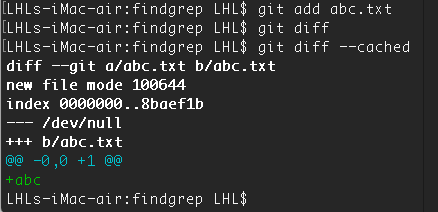
適当な”abc”を入れて保存します。

$ git add abc.txt

$ git diff

この時点で、git addコマンドを実行しても差分がないため何も表示されません。

インデックスとリポジトリの差分を表示する：

$ git diff —cached

別の言い方すると、git diff —cachedコマンドでは、次にgit commitコマンドを実行した時にリポジトリにコミットされる差分が表示されます。git addしてからgit commitするとき、実際にコミットされる内容を事前に確認したい時にこのコマンドを使います。