First step for version control

Introduction for Git

s1220233 Ryoya Komatsu

こんにちは

自己紹介

- 名前　　：小松　凌也
- Slack ID　　：@michael
- 学年　　：学部３年
- 出身地　　：福島県棚倉町
- 好きな言語　　：Python(2.x)
- 趣味　　：音楽鑑賞、麻雀

目次

- 導入
- Gitって何？
- Gitの構造と仕組み
- Gitの使い方
- 演習問題
- 参考資料

導入

みなさんはGitを
どの程度理解していますか？

導入

- 「add→commit→push」ってやるとGitHubにファイルが打ち上げられる
- 言われるがまま思考停止して使っている
- 難しいし意味分からない
- コマンドとかオプション多すぎ
- めんどい

- GitHubにファイル打ち上げてどうすんの？

同感です。

導入

- Gitとは何なのかを学ぶ前に、そもそも「バージョン管理システムとは何なのか」を知る必要がある。

導入 – バージョン管理システムとは

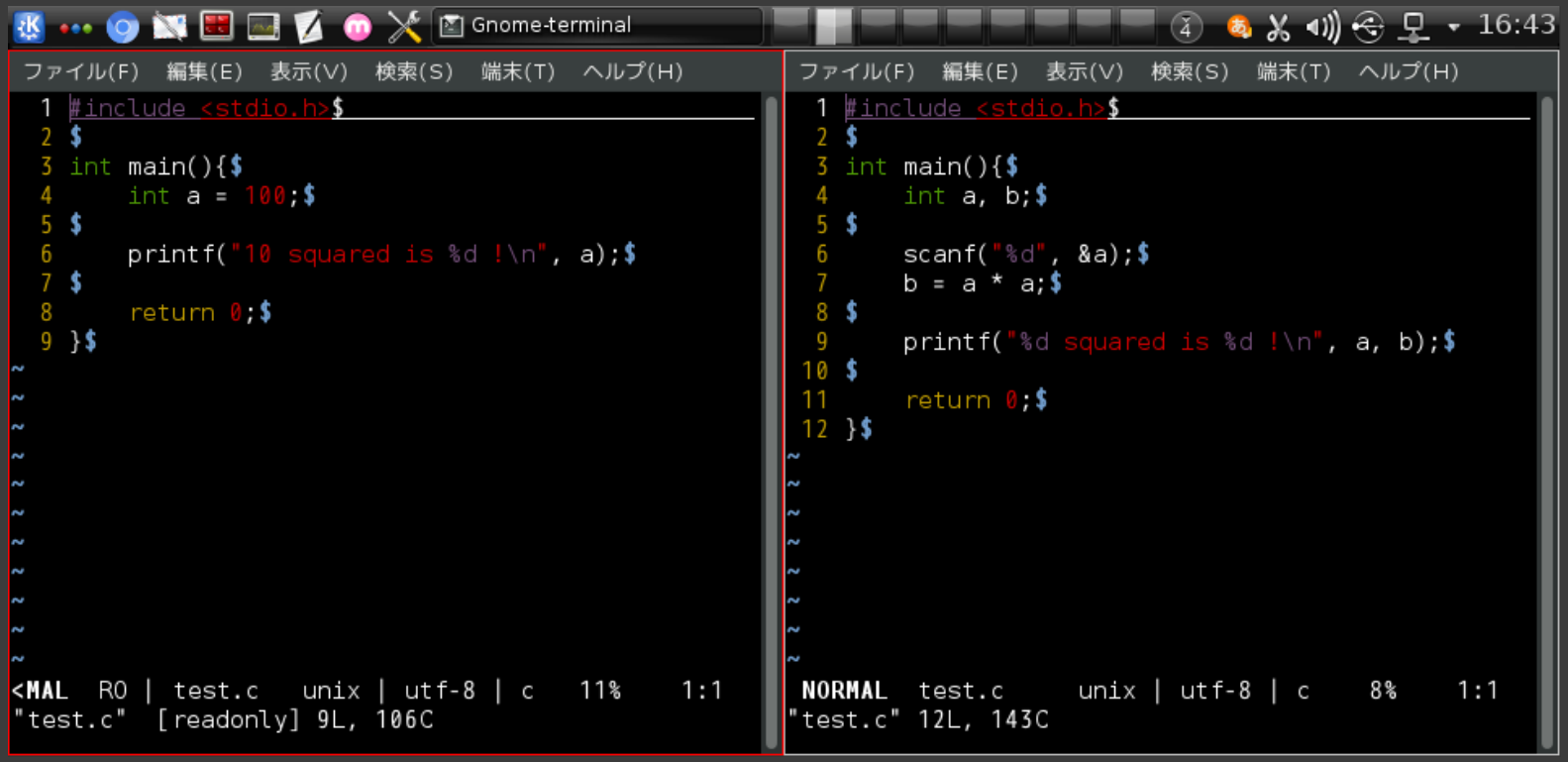
バージョン管理システム (Version Control System, VCS)

- ファイルの変更履歴をバージョン情報として蓄積して、ファイルに関する5W1Hの情報群を容易に管理できるようにするための便利ツール

※5W1H … when, where, who, what, why, how

……？？？

導入 – バージョン管理システムとは



変更前の test.c と変更後の test.c

- 4行目にint bの変数宣言が追加されている
- scanfで数字を読み取り、その二乗を求める処理が追加されている
- それらを表示するprintfが少し書き換えられている

導入 – バージョン管理システムとは

- 「数字の二乗を表示する」というプログラム自体の目的は変わっていないが、変更の前後でその機能が変わっている。

→ 変更前をver1.0とすると変更後はver2.0

導入 – バージョン管理システムとは

- VCSを使うと、「いつ・誰が・どこに・どんな変更を加えたのか」を全て保存することができる。

- 変更して上書き保存してしまった場合でも、過去のバージョン情報をVCSを使って記録しておけばver2.0のtest.cをver1.0に(更にその前のバージョンにも)戻すことができる。

導入 – バージョン管理システムとは

更に…

- GitHubなどと連携すればソースコードのバックアップも同時にこなせるのでとても便利。
- 複数人でチームを組んで一つのソフトウェアを開発するときにとても便利。(今回はあまり関係ない)

結論：めっちゃ便利

Gitって何？

Gitって何？

- 分散型バージョン管理システム (Distributed Version Control System, DVCS) の一つ
- ディレクトリの中のファイルの変更履歴を保存し、VCS用の共有ウェブサービスなどと連携しソフトウェアのバージョンを管理するためのツール

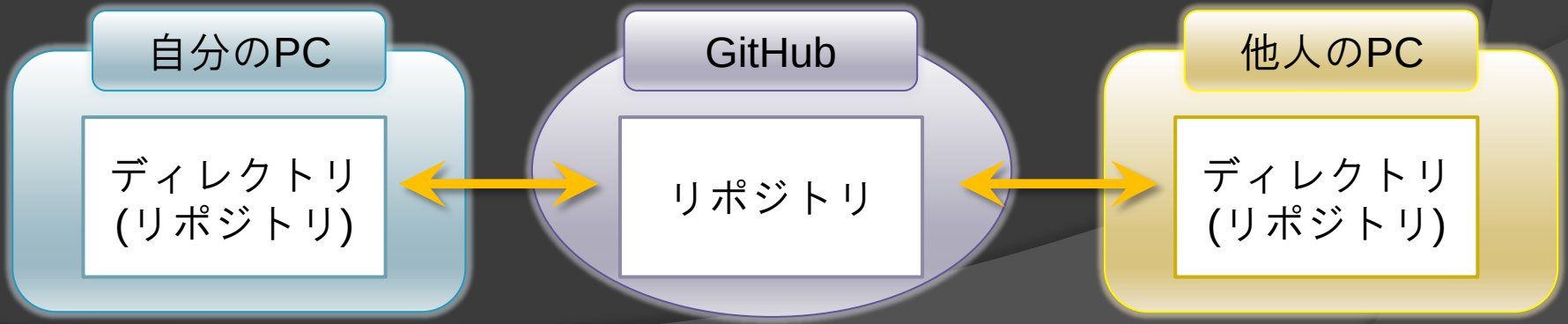
Gitって何？

- **git**（ギット）は、プログラムのソースコードなどの変更履歴を記録・追跡するための分散型バージョン管理システムである。Linuxカーネルのソースコード管理に用いるためにリーナス・トーバルズによって開発され、それ以降ほかの多くのプロジェクトで採用されている。

  - Wikipediaより引用

Gitって何？

- GitHubやBitBucketなどのVCS用共有ウェブサービスなどと連携することによって、自分の手元にあるプロジェクトの開発ディレクトリを瞬時にチームメンバーと共有することができ、進捗状況の同期や復元を容易に行うことができる。



Gitって何？

大切なこと

- Gitはバージョン管理用のツール、GitHubはそれを利用したサービスの一つです。
  別にGitとGitHubが２つで一つだというわけではなく、それぞれ独立しているものだということをしっかり覚えておきましょう。

Gitって何？

演習問題

1. Gitを利用した共有ウェブサービスの名前を一つ以上答えよ。

Gitの構造と仕組み

Gitの構造と仕組み

Gitを学ぶ上で重要なキーワード

- ワーキングディレクトリ
- リポジトリ(ローカル、リモート)
- ステージングエリア

Gitの構造と仕組み

- ワーキングディレクトリ

  ユーザーがファイル内容の編集や削除などの作業をする場所。ワーキングディレクトリ内のファイルの状態は常にGitによって監視されているが、ユーザーがワーキングディレクトリの存在を特に意識する必要はない。
  (見かけは普通のディレクトリと一緒)

Gitの構造と仕組み

- ローカルリポジトリ、リモートリポジトリ

  変更履歴の情報を保存しておくための場所をリポジトリという。Gitは分散型のバージョン管理システムなので、ワーキングディレクトリ内(**ローカル**)と共有サービス上(**リモート**)にリポジトリを持つ。

Gitの構造と仕組み

- ローカルリポジトリ、リモートリポジトリ

  ローカルリポジトリはワーキングディレクトリ内に隠されているが、変更記録の保存などは全てGitが見えない所で勝手に行ってくれるので特に気にする必要はない。

Gitの構造と仕組み

- ステージングエリア

  変更をリポジトリに記録するときに、どのファイルの変更を記録するか区別するためのスペース。

Gitの構造と仕組み

- ステージングエリア

  ワーキングディレクトリのファイルを変更しただけではリポジトリに変更が保存されない。

  ↓

  変更したファイルをステージングエリアに追加し、これから変更を記録することを予めGitに知らせておく。

Gitの構造と仕組み

Gitの基本フロー

- 「ワーキングディレクトリ内」のファイルを編集する
- ↓
- 編集したファイルを「ステージングエリア」に追加する
- ↓
- 「ローカルリポジトリ」にコミットする
- ↓
- 「リモートリポジトリ」にプッシュする

Gitの構造と仕組み

ローカル(自分のPC)環境上の構成

1. ワーキングディレクトリ
2. ローカルリポジトリ
3. (ステージングエリア)

- ステージングエリアは概念のようなものなので、実際にユーザーから可視的であるわけではない。

Gitの構造と仕組み

ローカル(自分のPC)環境上の構成

- リポジトリに対してファイルの変更を保存する操作自体は、ローカル環境のみで完結している。

→ ローカルリポジトリへのコミットの度にリモートリポジトリにプッシュする必要はなく、複数のコミットを重ねた任意のタイミングでプッシュすることができる。

ちょっと難しい話

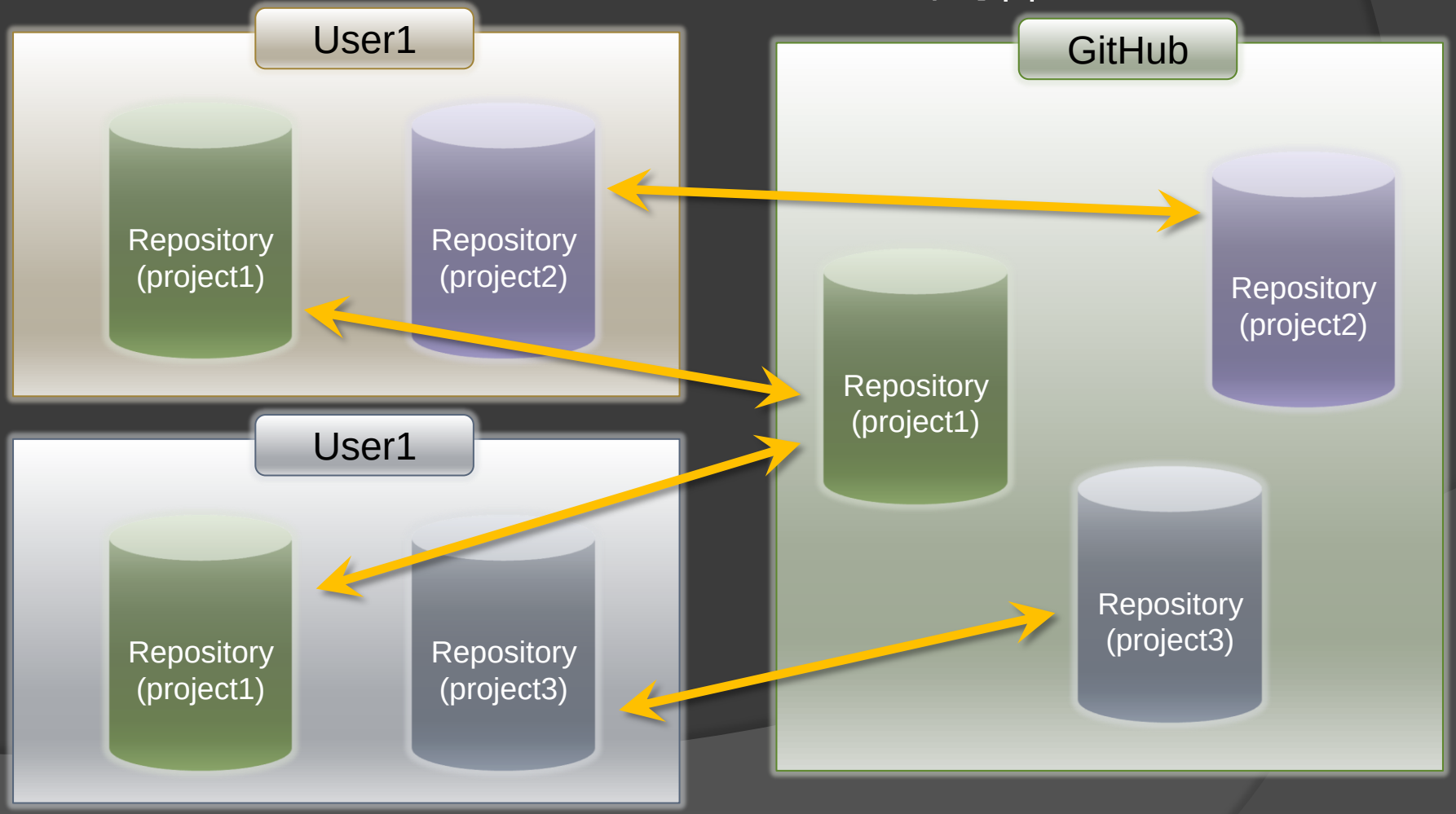
Gitの構造と仕組み

ローカルとリモートの関係

- （GitHubなどを利用している場合）リモートリポジトリは一つのプロジェクトにつきGitHub上に一つ置かれる。
- ローカルリポジトリは、プロジェクトに参加している各ユーザーのマシン上に置かれる。

Gitの構造と仕組み

ローカルとリモートの関係



Gitの構造と仕組み

ローカルとリモートの関係

- オリジナルはリモートリポジトリ（中央リポジトリ）であり、各ユーザーの持つローカルリポジトリはそのリモートリポジトリのコピーであるという考え方

Gitの構造と仕組み

Gitのライフサイクル

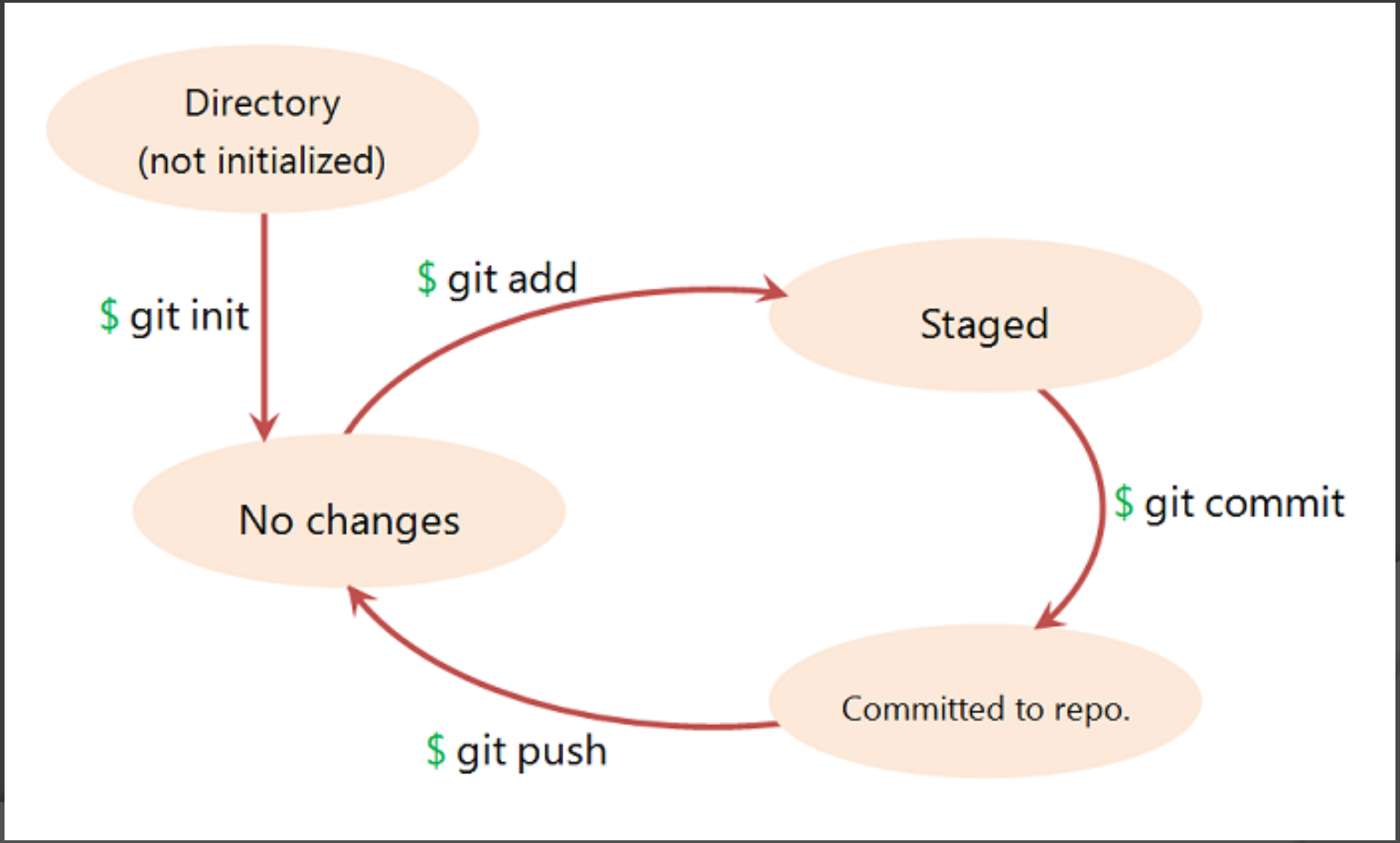
- Gitで管理されているディレクトリは、

  add – commit – push

  を一つのサイクルとして状態が循環的に変化していく。
  慣れないうちは、上のサイクルを一連のコマンドの塊として覚えてしまっても良い。

Gitの構造と仕組み

Gitのライフサイクル



Gitの構造と仕組み

Gitのライフサイクル

- 課題や演習問題の提出は、「リモートリポジトリへのプッシュ」を提出したものとみなします。

  - コードは完成したけどaddしてない
  - addしたけどcommitしてない
  - ローカルにcommitしたけどリモートにpushしてない

  などといったことがないように、
  きちんと add – commit – push しましょう。

Gitの構造と仕組み

演習問題

1. Gitを学ぶ上で重要な概念は３つあるが、それぞれの説明として適当なものを選べ。

1. ワーキングディレクトリ
2. リポジトリ
3. ステージングエリア

a. 変更履歴を保存する所
b. 変更したファイルを保存するかどうか区別する所
c. 実際にファイルの編集や削除を行う所

Gitの構造と仕組み

演習問題

1. Gitのライフサイクルを構成する1つのサイクルに含まれる3つのGitコマンドを列挙せよ。

2. ワーキングディレクトリ内のファイルの変更を直接記録(commit)するためのリポジトリを答えよ。(ローカル or リモート)

Gitの使い方

Gitの使い方

- Gitのコマンドは非常に多いが、主に使われるものはそれほど多くない。

Gitの使い方

とりあえず今覚えるべきなのはこれ

- git init
- git clone
- git branch
- git checkout
- git add
- git commit
- git push

Gitの使い方

覚えておくと便利なのがこれ

- git pull
- git merge
- git reset
- git revert
- git status
- git diff
- git log

Gitの使い方

git init

- 普通のディレクトリを、gitで管理可能なワーキングディレクトリとして初期化するためのコマンド。

  git clone(後述)を使わない限りは、ディレクトリ一つに対して必ず最初に一回実行する必要がある。

Gitの使い方

git init

- コマンド実行例

```
$ cd ~
$ mkdir git_intro
$ cd git_intro
$ ls -la
# ディレクトリがまだ空っぽであることを確認する
$ git init
$ ls -la
# .git というディレクトリが生成されていることを確認する
```

Gitの使い方

git init

- .git ディレクトリの中には、リモートリポジトリに関する情報やローカルリポジトリそのものが保存されている。

  誤って .git を消してしまうとコミット情報や他のブランチ(後述)のデータなどが全て吹き飛ぶので注意。

Gitの使い方

git clone

- 共有ウェブサービス上にあるリモートリポジトリをローカル環境に複製するためのコマンド。

Gitの使い方

git clone

- 間違えてローカルリポジトリを消した場合
- 他の人が作ったソースコードを自分の環境にコピーしたい場合
- 複数のユーザーやマシンでリポジトリ内のソースコードを管理したい場合

などに用いる。(割とよく使う)

Gitの使い方

git clone

- コマンド実行例

```
$ cd ~
$ git clone git@github.com:SCCP2016/git_intro_sample.git
$ ls
# git_intro_sample が存在することを確認する
$ cd git_intro_sample
$ ls -la
# README.md と .git が存在することを確認する

$ cd ~; rm -rf git_intro_sample
$ cd git_intro
```

Gitの使い方

git add

- ワーキングディレクトリ内で変更・追加されたファイルをステージングエリアに追加するためのコマンド。

  これを実行しないとリポジトリに変更を保存できない。

Gitの使い方

演習問題

1. 実行するとターミナル上に “This is a pen.” と表示するrubyのプログラム(pen.rb)を記述してgit_introディレクトリ内に保存せよ。

2. pen.rb をステージングエリアに追加するコマンドを実行せよ。

Gitの使い方

git add

- git status コマンドを使うと、ファイルの変更追跡情報を確認できる。
  - 新しくワーキングディレクトリにファイルを追加した場合
    → untracked files…
  - 直前のコミットからファイルに変更などを加えた場合
    → changes not staged for commit…
  - ステージングエリアにファイルが追加された場合
    → changes to be committed…

Gitの使い方

git add

- ファイルを変更・新しく追加した場合は必ず忘れずに git add するようにしましょう。

Gitの使い方

git commit

- ステージングエリアに追加されたファイルの変更をローカルリポジトリに保存するためのコマンド。

  コミット時にはメッセージを付与する必要がある。

Gitの使い方

git commit

- コマンド実行例

```
$ git commit -m “first commit”
$ git status
# nothing to commit と表示される
$ git log
# コミットの履歴を確認できる
```

Gitの使い方

git push

- ローカルリポジトリの内容をリモートリポジトリに反映するためのコマンド。

Gitの使い方

git push

- コマンド実行例

```
$ git push
# no configured push destination と表示される
$ git remote add origin REPOSITORY_URL
# REPOSITORY_URL には自分のGitHubリポジトリのURLを入れる
$ git push -u origin master
# pushに失敗する
$ git pull
$ git push
# 正しくpushできるはず
```

Gitの使い方

git push – 失敗する原因

- リモートリポジトリが設定されていない
  → git remote add
- リモートとローカルの歴史(コミット履歴)に矛盾がある
  → git pull

最後に

- Gitの使い方を覚えておくと、将来就職が有利になるようなことがあるかもしれません。

- 学んでいる間に数多くのエラーに衝突すると思われますが、英語を忌避せずにエラーを読んだりググったりして理解を深めるようにしましょう。