

Git 仕組み 入門

2013 02 23

岡山 Git 勉強会

自己紹介

火村 智彦

<http://eiel.info/>

Twitter
eielh

Github
eiel

Emacs
Haskell
Ruby
Gentoo
貳寺

目的

目的

目的

目的

- **Git の仕組みを**

目的

- **Git の仕組みを**
 - **なんとなく理解する**

目的

- **Git の仕組みを**
 - **なんとなく理解する**
- **特に**

目的

- **Git の仕組みを**
 - **なんとなく理解する**
- **特に**
 - **Git の Object について**

中身を知ると

中身を知ると

中身を知ると

- 応用力がつく

中身を知ると

- 応用力がつく
- トラブルに強くなる

対象者

対象者

対象者

- 一応 Git を利用してる人

参考文献

参考文献

参考文献

- Pro git 9章

参考文献

- **Pro git 9章**

- <http://git-scm.com/book/ja/Git%E3%81%AE%E5%86%85%E5%81%B4>

参考文献

- **Pro git 9章**

- <http://git-scm.com/book/ja/Git%E3%81%AE%E5%86%85%E5%81%B4>

- **man 7 gittutorial-2**

参考文献

- **Pro git 9章**

- <http://git-scm.com/book/ja/Git%E3%81%AE%E5%86%85%E5%81%B4>

- **man 7 gittutorial-2**

- http://cdn8.atwikiimg.com/git_jp/pub/git-manual-jp/Documentation/gittutorial-2.html

アジェンダ

アジェンダ

アジェンダ

- そもそも Git ってなんだっけ

アジェンダ

- そもそも Git ってなんだっけ
- Git Object

アジェンダ

- そもそも Git ってなんだっけ
- Git Object
- 考えてみよう - 他の機能とか

そもそも Git ってなんだっけ

バージョン管理システム

バージョン管理システム

バージョン管理システム

- 履歴が辿れる

バージョン管理システム

- 履歴が辿れる
- 変更内容を確認できる

バージョン管理システム

- 履歴が辿れる
- 変更内容を確認できる
- 以前の状態に戻れる

履歴が辿れる

- いつ、どうして変更したんだっけ

変更内容を確認できる

- **どんな変更をしたんだっけ**

以前の状態に戻る

- やっぱり前のほうが...

Git はどのように実現したか

Git はどのように実現したか

Git はどのように実現したか

- 履歴にすべてを保存した

Git はどのように実現したか

- 履歴にすべてを保存した
 - ファイルの内容

Git はどのように実現したか

- 履歴にすべてを保存した
 - ファイルの内容
 - ディレクトリの構造

Git はどのように実現したか

- 履歴にすべてを保存した
 - ファイルの内容
 - ディレクトリの構造
 - ファイルの一覧

Git はどのように実現したか

- 履歴にすべてを保存した
 - ファイルの内容
 - ディレクトリの構造
 - ファイルの一覧
- 履歴の情報

Git はどのように実現したか

- 履歴にすべてを保存した
 - ファイルの内容
 - ディレクトリの構造
 - ファイルの一覧
- 履歴の情報

Git はどのように実現したか

- 履歴にすべてを保存した 復元できる
 - ファイルの内容
 - ディレクトリの構造
 - ファイルの一覧
- 履歴の情報

履歴の情報

履歴の情報

履歴の情報

- 変更した人、時間

履歴の情報

- 変更した人、時間
- ひとつ前の履歴

履歴の情報

- 変更した人、時間
- ひとつ前の履歴
- 変更した理由

履歴の情報

- 変更した人、時間
- **ひとつ前の履歴**
- 変更した理由

履歴の情報

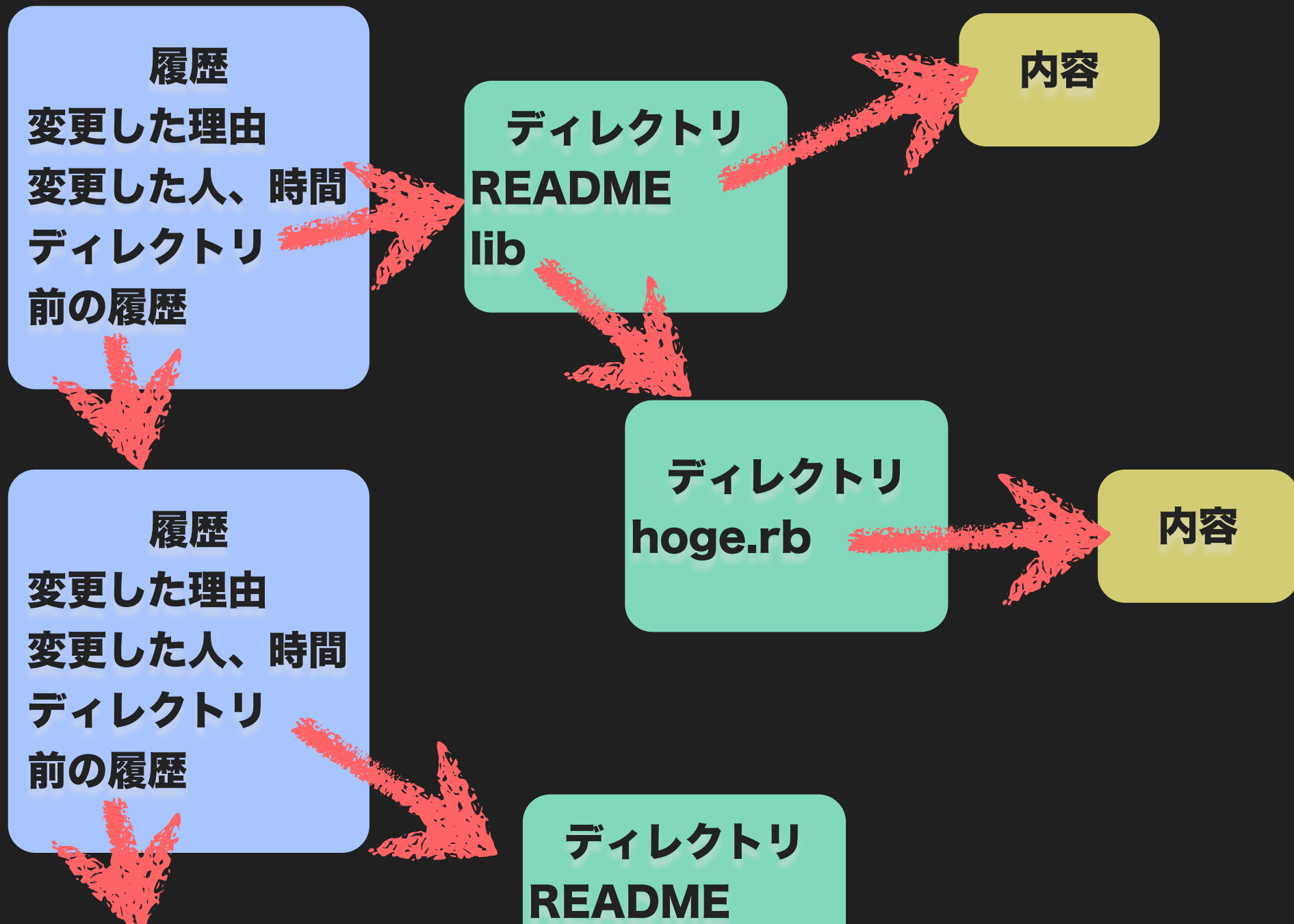
- 変更した人、時間

履歴を辿れる

- ひとつ前の履歴

- 変更した理由

Git はどのように実現したか



まとめ

まとめ

まとめ

- **Git はバージョン管理システム**

まとめ

- **Git はバージョン管理システム**
- **バージョン管理システムを作るには**

まとめ

- **Git はバージョン管理システム**
- **バージョン管理システムを作るには**
 - **ファイル**

まとめ

- **Git はバージョン管理システム**
- **バージョン管理システムを作るには**
 - **ファイル**
 - **ディレクトリ**

まとめ

- **Git はバージョン管理システム**
- **バージョン管理システムを作るには**
 - **ファイル**
 - **ディレクトリ**
 - **履歴**

Git Object

Git Object

Git Object

Git Object

- Git のデータベース

Git Object の種類

Git Object の種類

Git Object の種類

- **blob**

Git Object の種類

- **blob**
- **tree**

Git Object の種類

- **blob**
- **tree**
- **commit**

Git Object の種類

- **blob**
- **tree**
- **commit**
- **tag**

Git Object の種類

- **blob**
- **tree**
- **commit**
- **tag**
 - **今回は省略**

blob

blob

blob

- ファイルの内容

blob

- **ファイルの内容**
- **それ以上でもそれ以下でもない**

tree

tree

tree

- ディレクトリ構造

tree

- ディレクトリ構造
- blob と treeの一覧

commit

commit

commit

- 履歴そのもの

commit

- 履歴そのもの
- ひとつ前の commit を知ってる

commit

- 履歴そのもの
- ひとつ前の commit を知ってる
- 誰がいつ作ったものか知ってる

commit

- 履歴そのもの
- ひとつ前の commit を知ってる
- 誰がいつ作ったものか知ってる
- ルートディレクトリを知ってる

Object のフォーマット

名前

名前

名前

- 中身を **SHA-1** でハッシュ化したもの

中身

中身

中身

- ヘッダ

中身

- ヘッダ
 - オブジェクトの種類

中身

- ヘッダ
 - オブジェクトの種類
 - ファイルサイズ

中身

- ヘッダ
 - オブジェクトの種類
 - ファイルサイズ
- ボディ

中身

- ヘッダ
 - オブジェクトの種類
 - ファイルサイズ
- ボディ
 - 内容

中身

- ヘッダ
 - オブジェクトの種類
 - ファイルサイズ
- ボディ
 - 内容
- ヘッダとボディの区切り NUL

中身

- ヘッダとボディ が zlib で圧縮

オブジェクトの保存場所

オブジェクトの保存場所

オブジェクトの保存場所

- **.git/objects**

具体的に

サンプルリポジトリ

```
$ git clone git@github.com:ei-el/git-object-sample.git  
$ cd git-object-sample
```

コマンドとか資料は以下に用意してます。

<https://github.com/eiel/okagit-object>

ファイル構成

```
$ tree
```

```
├── README.md
├── doc
│   └── commands.org
└── object.org
```

```
1 directory, 3 files
```

オブジェクトの保存場所

- **.git/objects**

.git/objects

```
$ tree .git/objects
```

```
.git/objects/
```

```
├── info
```

```
└── pack
```

```
    ├── pack-68c839f0766ad945f74b6bc1785d7b952c07996b.idx
```

```
    └── pack-68c839f0766ad945f74b6bc1785d7b952c07996b.pack
```

```
2 directories, 2 files
```

pack はオブジェクトをまとめたもの

今回は無視します

.git/objects

```
$ tree .git/objects
.git/objects/
├── info
└── pack
    ├── pack-68c839f0766ad945f74b6bc1785d7b952c07996b.idx
    └── pack-68c839f0766ad945f74b6bc1785d7b952c07996b.pack
```

2 directories, 2 files

とりあえずコミットしてみる

とりあえずコミットしてみる

```
$ echo "READMEを上書き" > README.md
```

```
$ git commit -a -m 'READMEを上書きしました'
```

```
$ cat README.md
```

```
READMEを上書き
```


.git/objects

```
$ tree .git/objects/
.git/objects/
├── 04
│   └── fb0db5f68f3e917ba90aacb09db78199bda6a7
├── a0
│   └── d74d48d61f95a874e30f8fb71bbd68506d0f6e
├── fe
│   └── ba9138249c3c6fcd9435fc38a6a29193abb76c
├── info
└── pack
    ├── pack-68c839f0766ad945f74b6bc1785d7b952c07996b.idx
    └── pack-68c839f0766ad945f74b6bc1785d7b952c07996b.pack
```

5 directories, 5 files

オブジェクトが3つできました

04fb0db5f68f3e917ba90aacb09db78199bda6a7
- commit **人によって違う**

a0d74d48d61f95a874e30f8fb71bbd68506d0f6e
- blob **README**

feba9138249c3c6fcd9435fc38a6a29193abb76c
- tree

ファイルの中身を見てみよう

zlib ライブラリで圧縮されてるので...

```
ruby -rzlib -e 'puts Zlib.inflate(ARGF.read)'
```

展開して中身を見る

```
$ cat .git/objects/a0/d74d48d61f95a874e30f8fb71bbd68506d0f6e | \  
    ruby -rzlib -e 'puts Zlib.inflate(ARGF.read)'  
blob 19READMEを上書き
```

ただのテキストデータ

中身

- ヘッダ

- オブジェクトの種類
- ファイルサイズ

- ボディ

- 内容
- ヘッダとボディの区切り NUL

ヘッダ

ボディ

blob 19READMEを上書き

中身

- ヘッダ
 - オブジェクトの種類
 - ファイルサイズ
- ボディ
 - 内容
- ヘッダとボディの区切り NUL

種類 サイズ

blob 19

中身

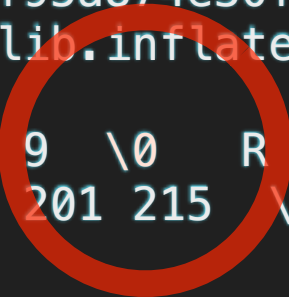
- ヘッダ
 - オブジェクトの種類
 - ファイルサイズ
- ボディ
 - 内容
- ヘッダとボディの区切り NUL

区切りは NUL

```
$ cat .git/objects/a0/d74d48d61f95a874e30f8fb71bbd68506d0f6e | \
    ruby -rzlib -e 'puts Zlib.inflate(ARGF.read)' | \
    od -c
0000000  b   l   o   b           1   9   \0   R   E   A   D   M   E   202
0000020 222       ? 212   233   ?   201 215   \n
0000033
```

区切りは NUL

```
$ cat .git/objects/a0/d74d48d61f95a874e30f8fb71bbd68506d0f6e | \  
  ruby -rzlib -e 'puts Zlib.inflate(ARGF.read)' | \  
  od -c  
0000000  b  l  o  b      1  9  \0  R  E  A  D  M  E  202  
0000020 222      ? 212  233  ? 201 215  \n  
0000033
```



名前

- 中身を **SHA-1** でハッシュ化したもの

名前は 中身を SHA でハッシュしたもの

```
$ cat .git/objects/a0/d74d48d61f95a874e30f8fb71bbd68506d0f6e | \
  ruby -rzlib -e 'puts Zlib.inflate(ARGF.read)' | \
  sha1sum
a0d74d48d61f95a874e30f8fb71bbd68506d0f6e
```

名前は 中身を SHA でハッシュしたもの

```
$ cat .git/objects/a0/d74d48d61f95a874e30f8fb71bbd68506d0f6e | \
  ruby -rzlib -e 'puts Zlib.inflate(ARGF.read)' | \
  sha1sum
a0d74d48d61f95a874e30f8fb71bbd68506d0f6e
```

名前は 中身を SHA でハッシュしたもの

```
$ cat .git/objects/a0/d74d48d61f95a874e30f8fb71bbd68506d0f6e | \
  ruby -rzlib -e 'puts Zlib.inflate(ARGF.read)' | \
  sha1sum
a0d74d48d61f95a874e30f8fb71bbd68506d0f6e
```

名前は 中身を SHA でハッシュしたもの

一致

```
$ cat .git/objects/a0/d74d48d61f95a874e30f8fb71bbd68506d0f6e | \  
  ruby -rzlib -e 'puts Zlib.inflate(ARGF.read)' | \  
  sha1sum  
a0d74d48d61f95a874e30f8fb71bbd68506d0f6e
```

他の中身を見る方法

```
$ git cat-file -t a0d74  
blob
```

```
$ git cat-file -s a0d74  
19
```

```
$ git cat-file -p a0d74  
READMEを上書き
```

```
$ git show a0d74  
READMEを上書き
```

他の中身を見る方法

```
$ git cat-file -t a0d74  
blob
```

種類

```
$ git cat-file -s a0d74  
19
```

```
$ git cat-file -p a0d74  
READMEを上書き
```

```
$ git show a0d74  
READMEを上書き
```

他の中身を見る方法

```
$ git cat-file -t a0d74  
blob
```

種類

```
$ git cat-file -s a0d74  
19
```

サイズ

```
$ git cat-file -p a0d74  
READMEを上書き
```

```
$ git show a0d74  
READMEを上書き
```

他の中身を見る方法

```
$ git cat-file -t a0d74  
blob
```

種類

```
$ git cat-file -s a0d74  
19
```

サイズ

```
$ git cat-file -p a0d74  
READMEを上書き
```

中身

```
$ git show a0d74  
READMEを上書き
```


他の中身を見る方法

```
$ git cat-file -t a0d74  
blob
```

種類

```
$ git cat-file -s a0d74  
19
```

サイズ

```
$ git cat-file -p a0d74  
READMEを上書き
```

中身

```
$ git show a0d74  
READMEを上書き
```

整形された情報

補足

パックされてるオブジェクトの一覧

オブジェクトを全部見てみる

```
$ git rev-list --all --objects
04fb0db5f68f3e917ba90aacb09db78199bda6a7
7dca985cea0d6e31591b46fe2610d5538dce466d
26d033a78bb394bb0e86f8fb27a024f1df165300
e4ffe1da5832106dd1a173db4854a1f18d54f0e8
feba9138249c3c6fcd9435fc38a6a29193abb76c
a0d74d48d61f95a874e30f8fb71bbd68506d0f6e README.md
7ba05afba1830003b1c68c63f95f08440ee7a355 doc
2784fbf7b28eeebf2676ff242e6490247e5477d6 doc/
commands.org
e2c1d9639a78a7eeb2432931f3fc78ffabf6aed9 object.org
3b146a001783e7e5333f6da0560cf569c73da9b2
bbc83e8056103c2270c974e1abe2b79ac518cb6a README.md
488d7f8b03c0874df7e9829cfa96a318fe4a4248
```

オブジェクトの詳細

commit

ふたつ前のcommit をみてみます

ふたつ前のcommit

```
$ git cat-file -p 7dca985cea0d6e31591b46fe2610d5538dce466d
tree 3b146a001783e7e5333f6da0560cf569c73da9b2
parent 26d033a78bb394bb0e86f8fb27a024f1df165300
author Tomohiko Himura <eiel.hal@gmail.com> 1361346494 +0900
committer Tomohiko Himura <eiel.hal@gmail.com> 1361347251 +0900
```

三番目のコミット

commit

- 履歴そのもの
- ひとつ前の commit を知ってる
- 誰がいつ作ったものか知ってる
- ルートディレクトリを知ってる

ふたつ前のcommit

```
$ git cat-file -p 7dca985cea0d6e31591b46fe2610d5538dce466d  
tree 3b146a001783e7e5333f6da0560cf569c73da9b2  
parent 26d033a78bb394bb0e86f8fb27a024f1df165300  
author Tomohiko Himura <eiel.hal@gmail.com> 1361346494 +0900  
committer Tomohiko Himura <eiel.hal@gmail.com> 1361347251 +0900
```

三番目のコミット

commit

- 履歴そのもの
- ひとつ前の commit を知ってる
- 誰がいつ作ったものか知ってる
- ルートディレクトリを知ってる

ふたつ前のcommit

```
$ git cat-file -p 7dca985cea0d6e31591b46fe2610d5538dce466d  
tree 3b146a001783e7e5333f6da0560cf569c73da9b2  
parent 26d033a78bb394bb0e86f8fb27a024f1df165300  
author Tomohiko Himura <eiel.hal@gmail.com> 1361346494 +0900  
committer Tomohiko Himura <eiel.hal@gmail.com> 1361347251 +0900
```

三番目のコミット

commit

- 履歴そのもの
- ひとつ前の commit を知ってる
- 誰がいつ作ったものか知ってる
- ルートディレクトリを知ってる

ふたつ前のcommit

```
$ git cat-file -p 7dca985cea0d6e31591b46fe2610d5538dce466d  
tree 3b146a001783e7e5333f6da0560cf569c73da9b2  
parent 26d033a78bb394bb0e86f8fb27a024f1df165300  
author Tomohiko Himura <eiel.hal@gmail.com> 1361346494 +0900  
committer Tomohiko Himura <eiel.hal@gmail.com> 1361347251 +0900
```

三番目のコミット

tree

tree

```
$ git cat-file -p feba913
100644 blob a0d74d48d61f95a874e30f8fb71bbd68506d0f6e README.md
040000 tree 7ba05afba1830003b1c68c63f95f08440ee7a355 doc
100644 blob e2c1d9639a78a7eeb2432931f3fc78ffabf6aed9 object.org
```

mode 種類 sha ファイル名

tree

```
$ git cat-file -p feba913
```

100644	blob	a0d74d48d61f95a874e30f8fb71bbd68506d0f6e	README.md
040000	tree	7ba05afba1830003b1c68c63f95f08440ee7a355	doc
100644	blob	e2c1d9639a78a7eeb2432931f3fc78ffabf6aed9	object.org

mode

種類

sha

ファイル名

ファイル構成

```
$ tree
```

```
├── README.md
├── doc
│   └── commands.org
└── object.org
```

```
1 directory, 3 files
```

tree

```
$ git cat-file -p feba913
```

```
100644 blob a0d74d48d61f95a874e30f8fb71bbd68506d0f6e    README.md
040000 tree 7ba05afba1830003b1c68c63f95f08440ee7a355    doc
100644 blob e2c1d9639a78a7eeb2432931f3fc78ffabf6aed9    object.org
```

ファイル構成

```
$ tree
```

```
├── README.md
├── doc
│   └── commands.org
└── object.org
```

```
1 directory, 3 files
```

tree

```
$ git cat-file -p feba913
```

```
100644 blob a0d74d48d61f95a874e30f8fb71bbd68506d0f6e    README.md
040000 tree 7ba05afba1830003b1c68c63f95f08440ee7a355    doc
100644 blob e2c1d9639a78a7eeb2432931f3fc78ffabf6aed9    object.org
```

**mode については
man 2 stat
st_mode で検索**

blob

blob は もう見たからいいや

まとめ

まとめ

まとめ

- **Git は バージョン管理システム**

まとめ

- Git は バージョン管理システム
- バージョン管理システムを作るには

まとめ

- **Git は バージョン管理システム**
- **バージョン管理システムを作るには**
 - **ファイル - blob**

まとめ

- **Git は バージョン管理システム**
- **バージョン管理システムを作るには**
 - **ファイル - blob**
 - **ディレクトリ - tree**

まとめ

- **Git は バージョン管理システム**
- **バージョン管理システムを作るには**
 - **ファイル - blob**
 - **ディレクトリ - tree**
 - **履歴 - commit**

考えてみよう

**Git の 機能について
実装方法を
想像してみよう**

答えはかかないけど

前の状態にもどす

差分をとる

ブランチ

git add したければ復元できるかも

ファイルの移動の検知

git clone とか fetch

**rebase したり
コミットログをかきかえると
コミットの sha が変わる**

ご清聴ありがとうございます