GIT 勉強会

第 3 回だよ

3.1 ブランチとは

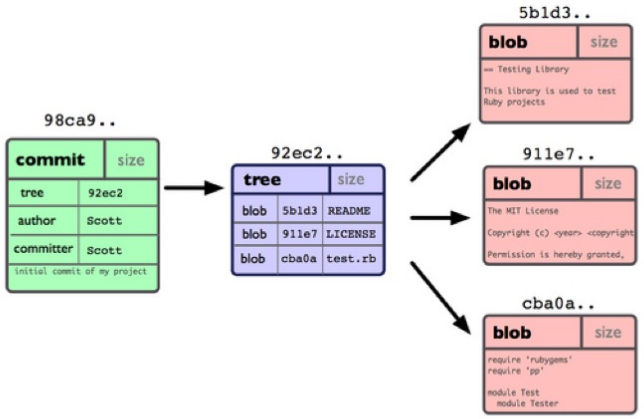


図 3.1: ひとつのコミットをあらわすリポジトリ上のデータ

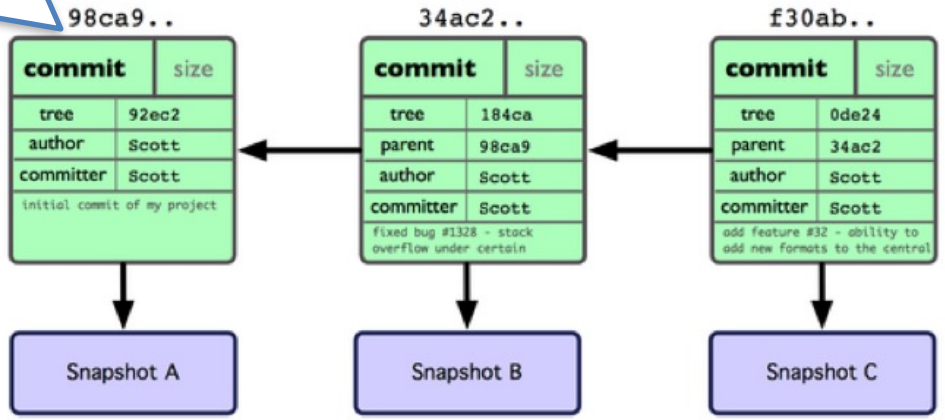


図 3.2: 複数のコミットに対応する Git オブジェクト

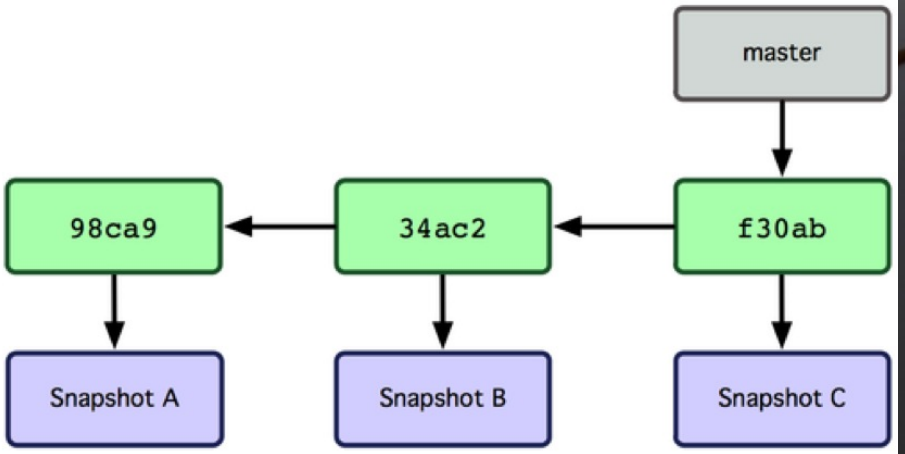


図 3.3: コミットデータの歴史を指すブランチ

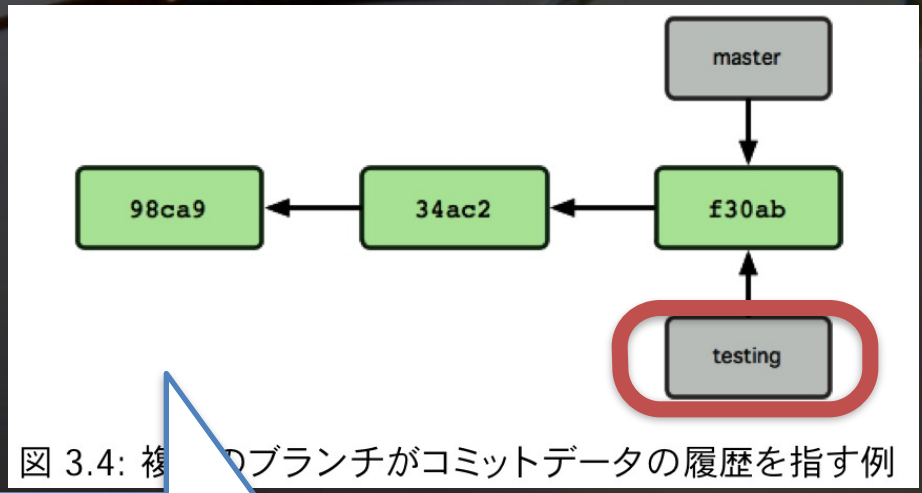


図 3.4: 複 のブランチがコミットデータの履歴を指す例

Branch コマンドで
ブランチを作成

$ git **branch** testing

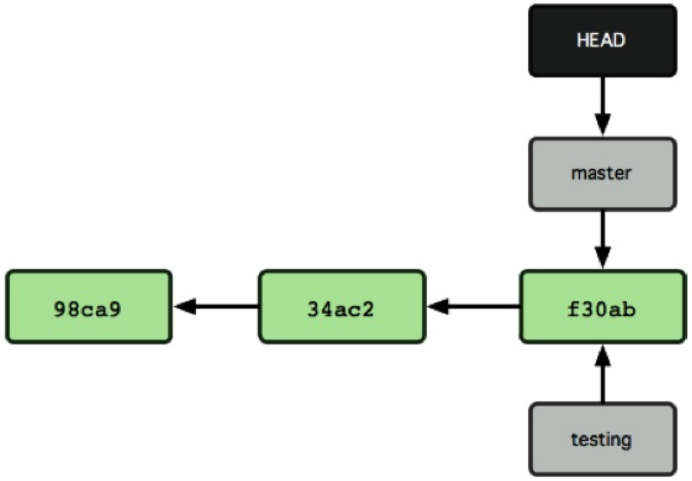


図 3.5: 現在作業中のブランチを指す HEAD

checkout コマンドで
ブランチの向き先を切り替える

$ git **checkout** testing

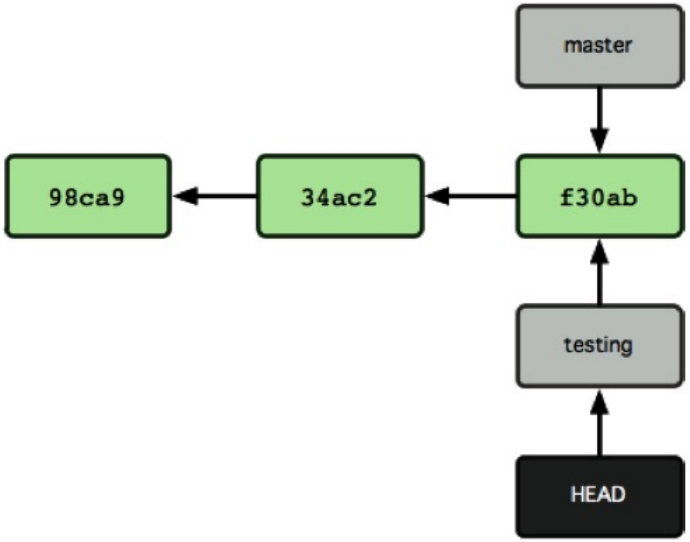


図 3.6: ブランチを切り替えると、HEAD の指す先が移動する

コミットをすると、ブランチの先にあるリポジトリが更新される

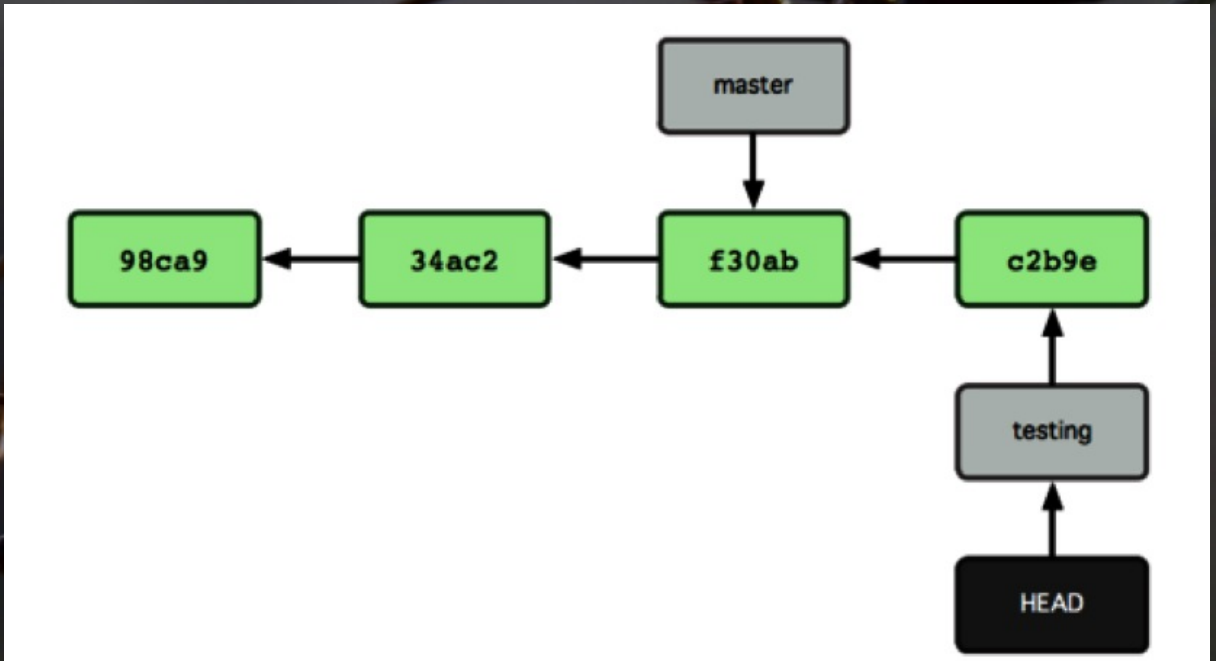


図 3.7: HEAD が指すブランチが、コミットによって移動する

3.2 ブランチとマージの基本

branchって、

実際のPJではどのように活用できるの？

実際の業務の流れに沿って
説明していきます！

3.2 ブランチとマージの基本

①

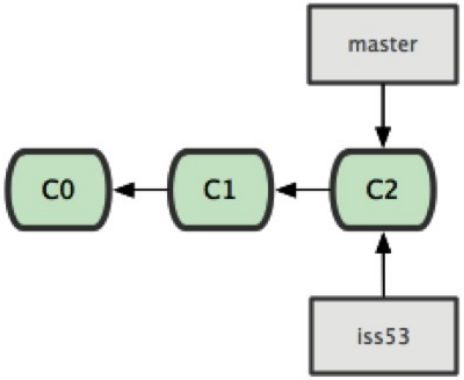


図 3.11: 新たなブランチポインタの作成

マスター(master)から
branch(iss53)を切る

$ git checkout -b iss53
(branch作成とcommit作成の
ショートカット)

②

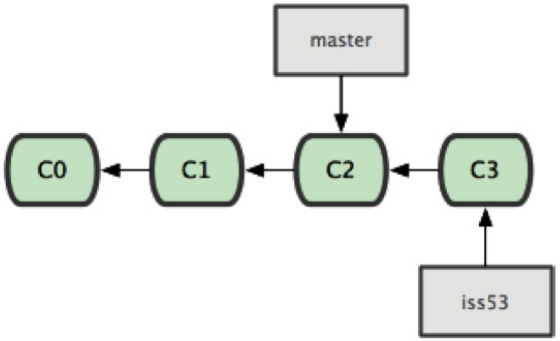


図 3.1

作業後にコミットする

⇒iss53が更新されていく

masterに問題発生！！

3.2 ブランチとマージの基本

③ branchをマスター(master)に戻す

$ git checkout master

(※作業フォルダ(iss53)をクリーンな状態にしておくこと)

⇒ローカルファイルの削除・追加・変更が自動的に行われる

④

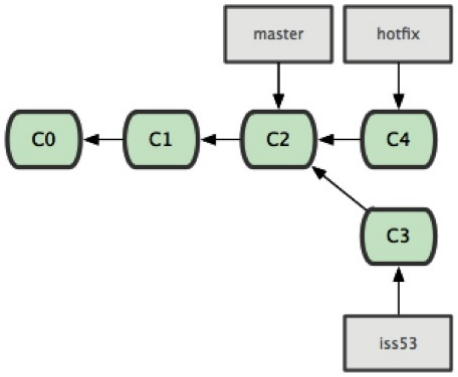


図 3.13: master ブランチから新たに作成した hotfix ブランチ

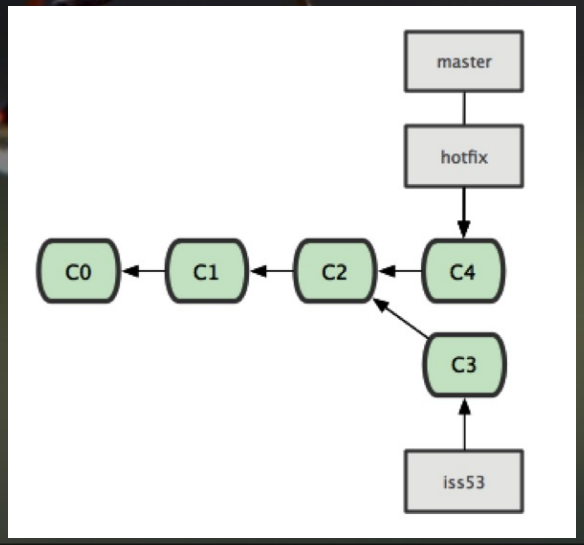
緊急対策用branchを作成する

$ git checkout -b hotfix

3.2 ブランチとマージの基本

⑤ 資源を修正後、hotfixをmasterにマージする

```
$ git checkout master
$ git merge hotfix
Updating f42c576..3a0874c
Fast forward
README | 1 -
1 files changed, 0 insertions(+), 1 p(-)
```



※Fast forward
marge先が分岐しておらず
ポインタを進めるだけでいい

3.2 ブランチとマージの基本

⑥ masterと同じになったhotfixを削除する

$ git branch -d hotfix

⑦

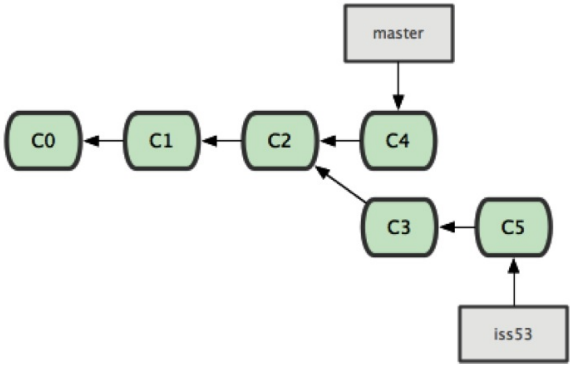


図 3.15: iss53 ブランチは独立して進めることができる

iss53に戻り作業を進める

※hotfixの内容はiss53には反映されていない

3.2 ブランチとマージの基本

⑧ 資源を修正後、hotfixをmasterにマージする

```
$ git checkout master
$ git merge iss53
Merge made by recursive.
README | 1 +
1 files changed, 1 insertions(+), 0 deletions(-) git checkout master
```

自動的にmarge先を探してくれる！

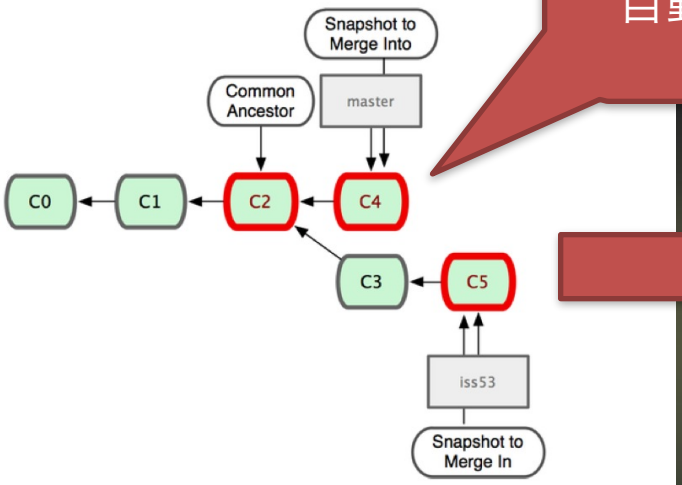


図 3.16: Git が共通の先祖を自動的に見つけ、ブランチのマージに使用する

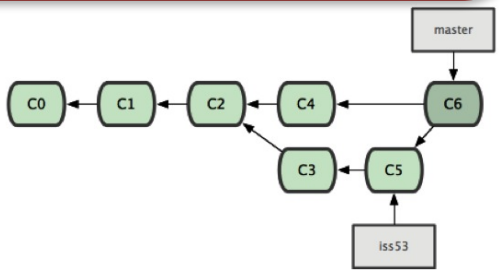


図 3.17: マージ作業の結果から、Git が自動的に新しいコミットオブジェクトを作成する

3.2 ブランチとマージの基本

同じファイルを変更していた場合、

マージは成功するの？

master とbranchのどっちが反映されるの？

マージは失敗するけど、
どっちを反映させるかは簡単に選べるよ！

3.2 ブランチとマージの基本

同じ資源に対して変更を加えていた場合、

margeの際に下記のようなコンフリクトが発生する

※marge commitは作成されない

```
$ git merge iss53
Auto-merging index.html
CONFLICT (content): Merge conflict in index.html
Automatic merge failed; fix conflicts and then commit the result.
```

merge に失敗しました。

3.2 ブランチとマージの基本

どのファイルが原因で競合しているかを調べる

$ git **status**

```
index.html: needs merge
# On branch master
# Changes not staged for commit:
# (use "git add <file>..." to update what will be committed)
# (use "git checkout -- <file>..." to discard changes in working
directory)
#
# unmerged: index.html
#
```

原因のファイル

3.2 ブランチとマージの基本

競合が発生しているファイルには印がつく！

```
<<<<<<< HEAD:index.html
<div id="footer">
contact : email.support@github.com
</div>
=======
<div id="footer">
please contact us at support@github.com
</div>
>>>>>>> iss53:index.html
```

HEAD

iss53

3.2 ブランチとマージの基本

競合を解決して、<<<<や====や>>>>を消すよ

```
<<<<<<<< HEAD:index.html
<div id="footer">
contact : email.support@github.com
</div>
========
<div id="footer">
please contact us at support@github.com
</div>
>>>>>>>> iss53:index.html
```

3.2 ブランチとマージの基本

競合を解決して、<<<<や====や>>>>を消すよ

```
<div id="footer">
please contact us at email.support@github.com
</div>
```

競合を解決したら

$ git add

※ステージングされたファイルは競合が解消されていると認識される。

3.2 ブランチとマージの基本

addした後に $ git **status** すると

競合が解決されたか確認することが出来る

$ git **commit** でmargeを完了させる

<<結果>>

```
Merge branch 'iss53'
Conflicts:
index.html
#
# It looks like you may be committing a MERGE.
# If this is not correct, please remove the file
# .git/MERGE_HEAD
# and try again.
#
```

3.2 ブランチとマージの基本

グラフィカルにmergeを解決したい場合は

$ git **mergetool** を使う

(※デフォルトのdiffツール起動)

<<結果>>

```
merge tool candidates: kdiff3 tkdiff xxdiff meld gvimdiff
opendiff emerge vimdiff
Merging the files: index.htm

Normal merge conflict for 'i
{local}: modified
{remote}: modified
Hit return to start merge re
```

デフォルト以外のdiffツールを
使いたい場合は
この中から選んで打ち込む

3.2 ブランチとマージの基本

$ git **branch** を使うことで
現在のブランチ一覧を取得することが出来る

<<結果>>

```
  iss53
* master
  testing
```

現在参照しているブランチには
「*」マークがつく

3.2 ブランチとマージの基本

$ git **branch -v** を使うことで
各ブランチの直近のコミットを
参照することが出来る

<<結果>>

```
  iss53 93b412c fix javascript issue
* master 7a98805 Merge branch 'iss53'
  testing 782fd34 add scott to the author list in the readmes
```

3.2 ブランチとマージの基本

ブランチがマージ済みかどうか
調べることも出来る

マージ済みブランチを見る：$ git **--merged**
未マージブランチを見る　：$ git **--no-merged**

3.2 ブランチとマージの基本

マージされていないブランチは

$ git **branch -d** testing　では

削除することが出来ない

-D をオプションとして
使えば
強制的に削除出来るよ

<<結果>>

error: The branch 'testing' is not an ancestor of your current HEAD.
If you are sure you want to delete it, run 'git branch -D testing'.

ありがとうございました！

次回は「3.4 ブランチでの作業の流れ」から
お願いします！