SourceTree入門

SourceTreeでGitによるバージョン管理および環境を構築します。

ホスティングサービスはGitHubを使用します。

前提条件としてGitHubのアカウントを作成しておいてください。

# SourceTreeとGitについて



Gitとは主にソースコードなどのバージョン管理を行うCUIツールです。

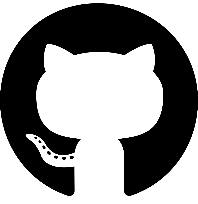
バージョン管理システムでは管理データを**リポジトリ**といいます。

Gitでは編集前のリポジトリと編集後のリポジトリを**ブランチ**という分岐点で表現します。



SourceTreeとはGitをGUIで操作可能にしたツールです。

コマンドを直接記述せず、直感的に操作可能です。



GitHubはGitによるバージョン管理に特化したホスティングサービスです。

リポジトリを保存するサーバと関連機能を提供します。

PC上のリポジトリをローカルリポジトリといい、

それに対して、サーバ上のリポジトリをリモートリポジトリといいます。

# SourceTreeの導入

1. ダウンロード

以下のサイトからインストーラをダウンロードします。

https://www.sourcetreeapp.com

サイト右上の[Download free]からダウンロードできます。



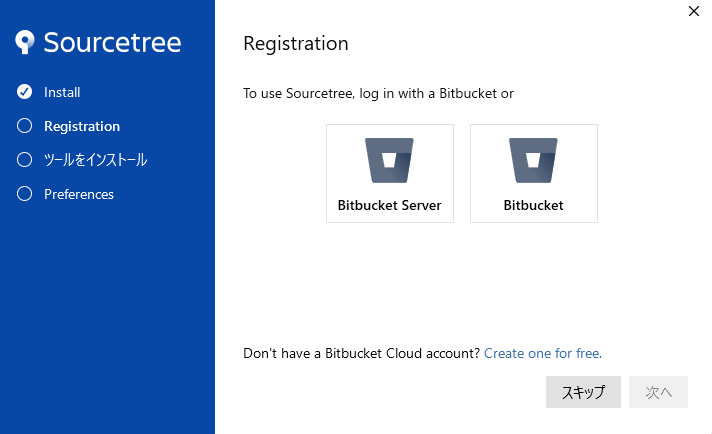
1. インストール

インストーラを起動します。

* 1. Bitbucketの設定はスキップします。

BitbucketはGit向けのホスティングサービスです。

今回はGitHubを使用するため[スキップ]します。

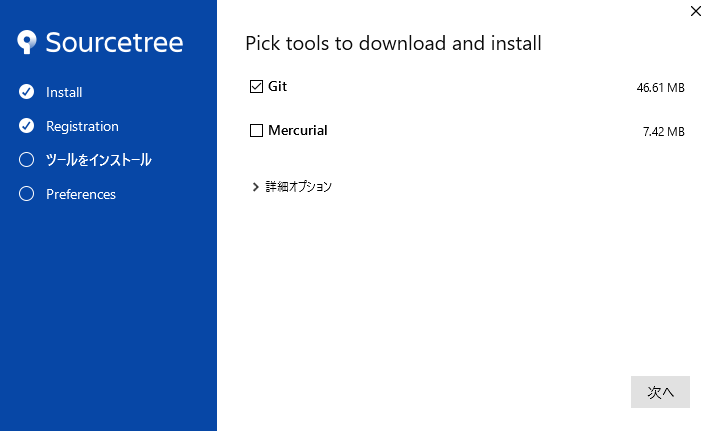


* 1. バージョン管理ツールはGitのみ導入します。

GitとMercurialを導入可能ですが、Gitのみ導入します。

[ ]Mercurial チェックボックスのチェックを外してから[次へ]で進みます。

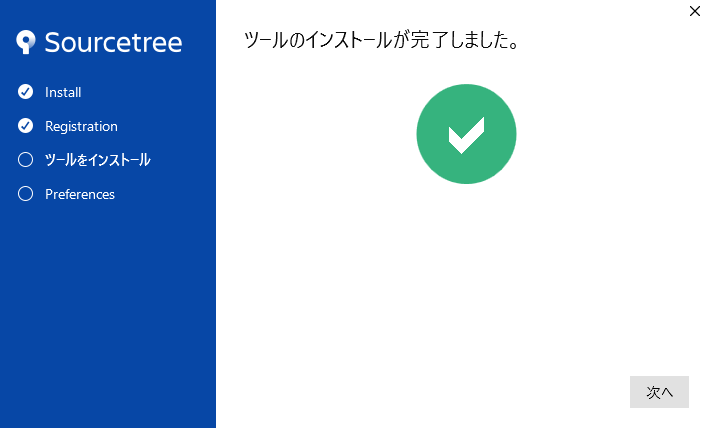
すでに導入済みの場合は、チェックボックスが非アクティブになります。



* 1. インストールされます。

インストール場所は「ユーザー\AppData\Local\SourceTree」です。

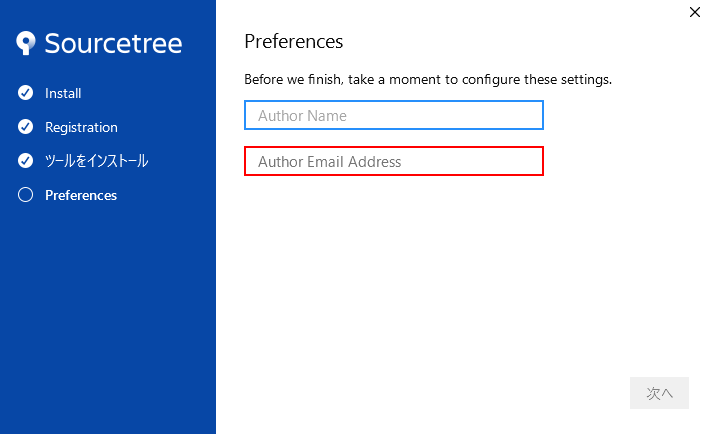
完了を確認してから[次へ]で進みます。



* 1. ユーザー名とメールアドレスを入力します。

入力したユーザー名はコミット時の実行者として情報が載ります。

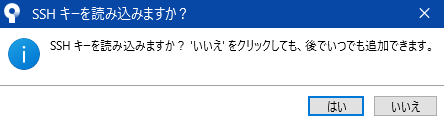
メールアドレスはGitHubアカウントを作成したときに使用したメールアドレスを入力してください。



* 1. SSHキーは設定しません。

GitHubはHTTPSによる通信を推奨しているため、SSHでは通信しません。

[いいえ]で閉じます。



1. アクセストークン

アクセストークンとはGitHubとSourceTreeをHTTPSで通信するための認証キーです。

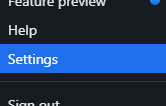
GitHubにサインインします。

* 1. ユーザーページの右上のユーザーアイコンをクリックし、メニューを開きます。



赤い枠で囲われた部分がアイコンです。

* 1. メニューから「Settings」を開きます。

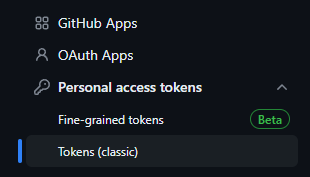


* 1. 左の項目から「Developer settings」を選択します。

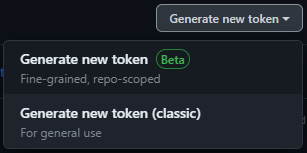


* 1. 左の項目から「Personal access tokens」を開き、「Token(classic)」を選択します。

「Token(classic)」は現行のトークン設定ですが、いずれ「Fine-grained tokens」に移行されるようです。



* 1. 右の[Generate new token]を開き「Generate new token (classic)」で新しいアクセストークンを生成します。



* 1. このアクセストークンを使用したアクセスのアクセス権を設定します。
     1. 「Note」でアクセストークン名を設定します。



* + 1. 「Expiration」で有効期限を設定します。

有効期限が切れると設定しなおさなければなりません。

「No expiration」で無期限を設定できます。



* + 1. 「Select scopes」でアクセス制限をかけます。

各設定の詳細は以下に載っています。

https://docs.github.com/ja/apps/oauth-apps/building-oauth-apps/scopes-for-oauth-apps#available-scopes

* + 1. [Generate token]で作成します。



* + 1. 作成したアクセストークンを保存します。

アクセストークンは以下のように表示されます。



白い隠しの位置に文字の羅列が表示されます。

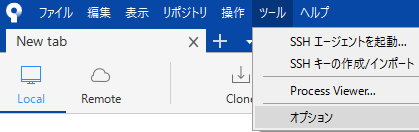
右のアイコンはコピーボタンです。

後ほどSourceTreeで設定するので保存しておきます。

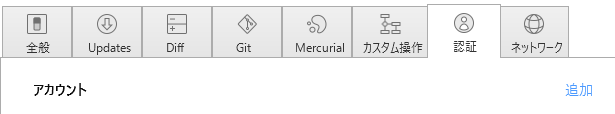
## SourceTreeにGitHubアカウントを登録

SourceTreeを開きます。

メニューの「ツール」から「オプション」を開きます。



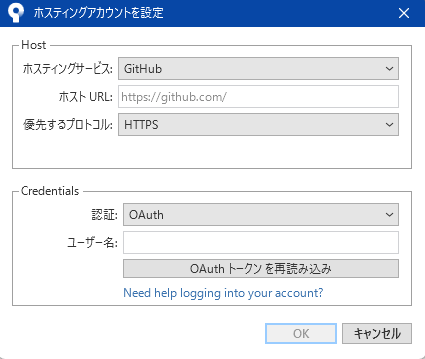
「認証」の「追加」をクリックします。



設定します。

ホスティングサービスは「GitHub」、優先するプロトコルは「HTTPS」、認証は「OAuth」

以上を設定して[OAuthトークンを再読み込み]でユーザーを読み込みます。



自動でブラウザが開きますが、引き続きSourceTreeで作業を行います。

ユーザー名がGitHubアカウント名になっている場合成功です。

[OK]でアカウントを追加します。

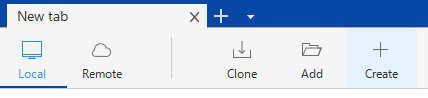
## リポジトリを作成

ローカルリポジトリを作成してリモートリポジトリに反映させます。

1. ローカルリポジトリを作成します。

New tabの「Create」をクリックします。

タブが開いていない場合「+」で追加できます。



「保存先のパス:」に対象のディレクトリパスを設定します。

[参照]で直接指定することもできます。

[作成]で対象のディレクトリのリポジトリを作成します。

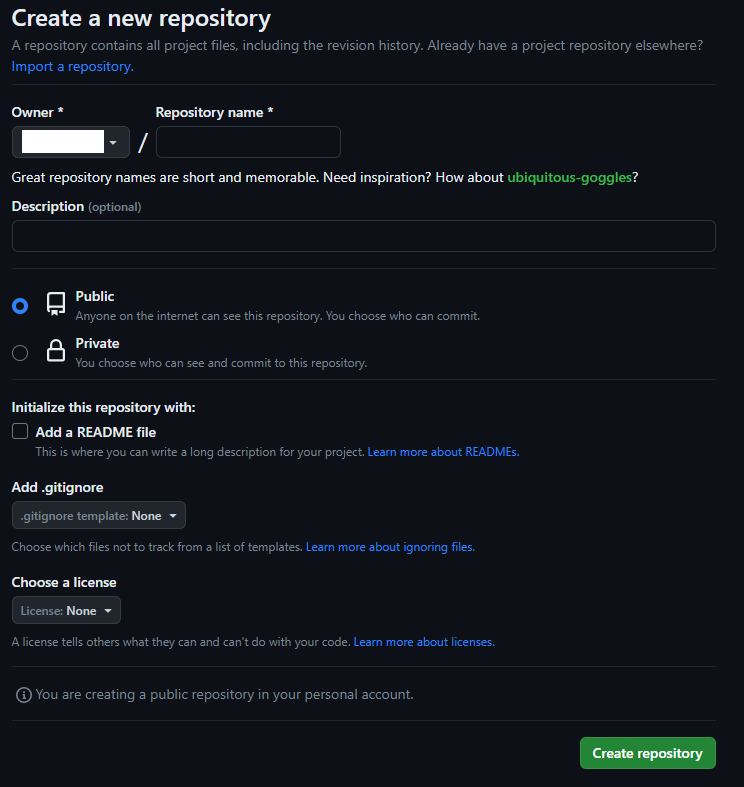


1. GitHubにリモートリポジトリを作成します。

GitHubのユーザーページを開きます。

「Create repository」または、「New」からリポジトリ作成画面を開きます。





「Repository name」にリポジトリの名前を入力します。

「Description」にリポジトリの説明を入力します。

「〇Public / 〇Private」でリポジトリを第三者にも公開するか設定します。

「□Add a README file」で「README.md」ファイルを追加するか設定します。

READMEとはこのリポジトリの扱い方などを記したファイルです。

「Add .gitignore」で変更の「.gitignore」ファイルのテンプレートを選択します。

.gitignoreとはGitによる変更の追跡をしないファイルを設定するファイルです。

「Choose a license」でライセンスを選択します。

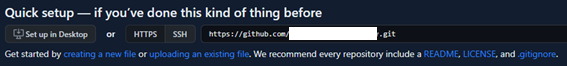
オープンではない独自のライセンスは後程手動で追加してください。

[Create repository]で作成します。

1. ローカルリポジトリとリモートリポジトリを関連付けます。

リポジトリを作成するリポジトリページが開かれます。

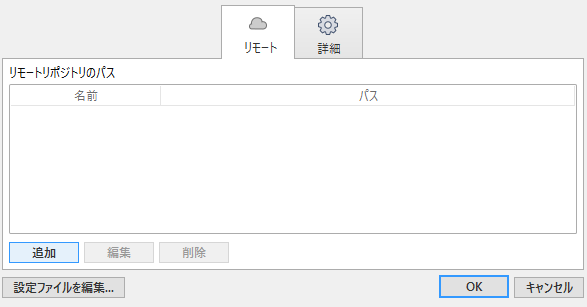
「<https://github.com/>ユーザーネーム/リポジトリ名.git」がリポジトリのURLです。



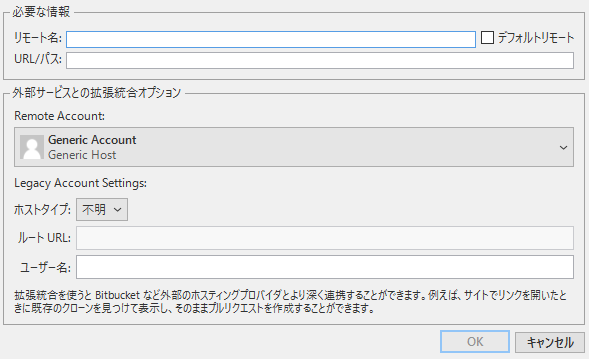
SourceTreeのタブで「リモート」からリモート設定画面を開きますます。



リモート設定画面のリモートタブの「追加」でリモートリポジトリと関連付けます。



関連付けるリモートリポジトリの設定を入力します。



「リモート名:」はリポジトリの名前を入力します。

「URL/パス:」にリモートリポジトリのURLを入力します。

「Remote Account」で自身のアカウントを選択します。

この画面で[OK]、さらにまどった先で[OK]をクリックして関連付けます。

1. ローカルリポジトリの内容を反映させます。

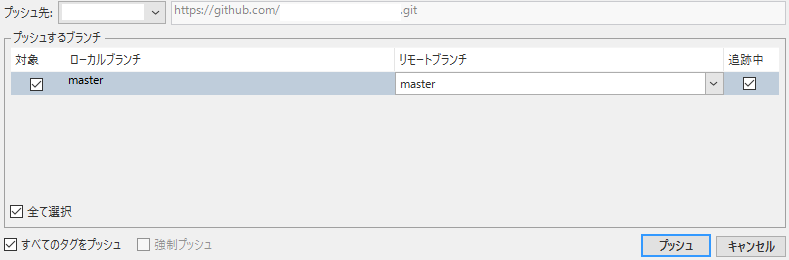
タブの「プッシュ」をクリックしてプッシュ設定画面を開きます。



「対象」チェックボックスにチェックを入れることで追跡するブランチを設定します。

「リモートブランチ」に関連付けるリモートのブランチを設定します。

[プッシュ]ボタンでリモートに反映させます。



1. はじめてのプッシュでは設定することがあります。

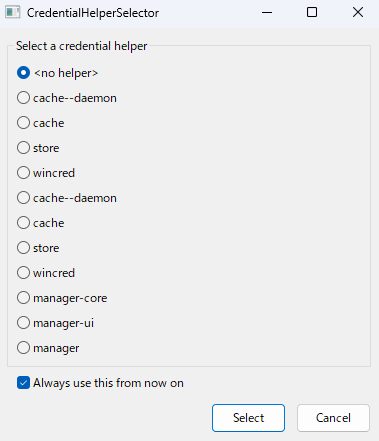
まず、CredentialHelperSelectorは「<no helper>」を選択します。

ユーザー名とパスワードを自動入力するヘルパーを使用するための選択です。

SourceTreeはヘルパーなしで自動入力してくれるのでここでは設定しません。

「[ ] Always this use from now on」チェックボックスにチェックを入れて、一回きりではなく今後もヘルパーを使わないように設定できます。

[Select]で確定します。

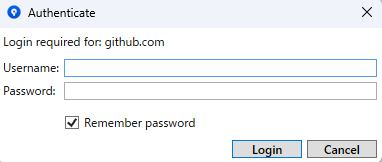


次にアカウント名とアクセストークンを入力します。

UsernameにGitHubのアカウント名を入力します。

Passwordに**アクセストークン**を入力します。

[Login]で設定完了です。



元の画面に戻ったらプッシュ完了です。

以後は特に入力なしでプッシュが可能です。

# Gitでの管理方法

管理の流れです。

サーバ

リモートリポジトリ

リモート

ブランチ

フェッチ

リモートブランチを

ローカル上にコピーします

プッシュ

ローカルブランチを

リモートに上げます

PC

ローカルリポジトリ

リモート追跡

ブランチ

ローカル

ブランチ

マージ

コピーされたブランチと

ローカルブランチを統合します

ブランチの内容は反映されます

ファイル

コミット

変更を確定させます

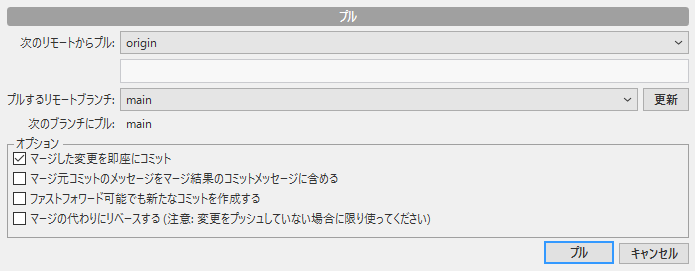
## リモートからローカルへ反映

リモートブランチをローカルブランチに反映させるにはプルを行います。

プルは「プル」ボタンで実行可能で、フェッチとマージを一度に行います。



「プル」ボタンでプル操作の設定画面を開きます。



「プルするリモートブランチ」で反映するブランチを選択できます。

[プル]をクリックして実行します。

## 変更内容を確定

SourceTreeは対象のプロジェクトを監視し、変更が加わると変更内容を表示します。

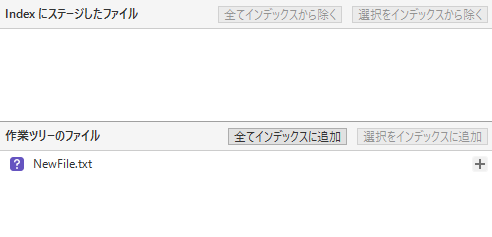


変更内容は確定を行うまではリポジトリに反映されません。

変更を確定させる処理をコミットといい、「コミット」ボタンで設定画面を表示させます。

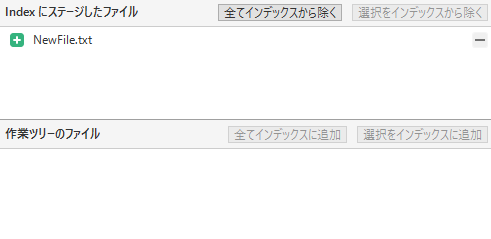


コミットさせる前にどのファイルの変更を確定させるかを設定します。



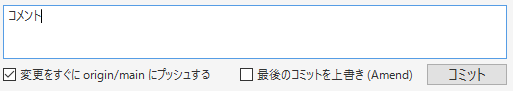
変更が加わったファイルは「作業ツリーのファイル」に表示されます。

変更を確定させたいファイルの[+]をクリックする、または、[全てインデクスに追加]で全てのファイルを確定対象に設定できます。



確定対象のファイルは「Indexにステージしたファイル」に移動します。

ファイルの[-]をクリックする、または、[全てインデクスから除く]で全てのファイルを確定対象から外すことができます。



[コミット]ボタンで確定させます。

このとき、ボタン上の枠にコメントを残すことで、何を変更させたかをわかりやすくします。

「[ ]変更をすぐに…にプッシュする」にチェックを入れることでコミットと同時にプッシュまで行います。

成功するとブランチのツリーに変更が表示されます。



失敗した場合エラー内容が表示されます。

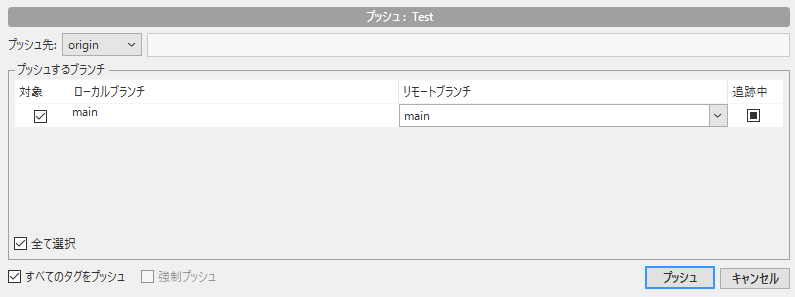
## ローカルからリモートへ反映

ローカルブランチをリモートブランチに反映させる操作はプッシュといいます。

プッシュはコミット時に「[ ]変更をすぐに…にプッシュする」にチェックを入れる以外に「プッシュ」ボタンをクリックして手動で行うこともできます。



プッシュボタンで設定画面を開きます。



「リモートブランチ」で反映させるブランチを設定できます。

[プッシュ]をクリックして実行します。

# 100MBを超えるファイル

## ファイルのプッシュ

プッシュする際、10MBを超えるサイズのファイルは警告が発生します。

さらに、GitHubは100MBを超えるサイズのファイルは通常プッシュできません。

## 100MBを超えるファイルの管理

100MBを超えるサイズのファイルは分割、もしくは、他の方法で管理します。

他の方法で管理する場合は .gitignoreで対象のファイルを追跡しないようにしましょう。

チームアカウントによるGoogleドライブの使用、また期限付きでのアップロードならば以下のサイトを利用できます。

ギガファイル便 <https://gigafile.nu>

# リポジトリの共有

複数人で1つのリポジトリを編集できるようにします。

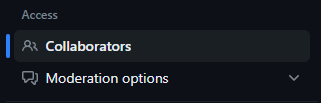
## オーナー設定

リポジトリのオーナー側の設定です。

GitHubでリポジトリのページから「Setting」を開きます。



左の項目から「Collaborators」を開きます。

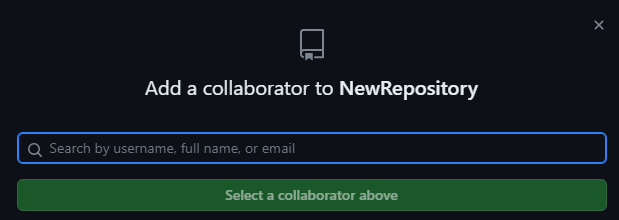


[Add people]を押してメンバー登録画面を開きます。



メンバー登録画面でメンバーネームを検索します。

[Select a collaborator above]を押して登録します。



## メンバー設定

メンバー側の設定です。

先にオーナー側でメンバー登録をしておきます。

SourceTreeを開きます。

新しいタブを開き、「Clone」を選択します。



リポジトリをURLで検索します。

このとき「github」の前にアクセストークンと「@」を付けます。

このアクセストークンはオーナーが用意したものです。



クローン位置を指定して[クローン]をクリックしてクローンします。

# ブランチとマージ

ブランチとは分岐のことで、Gitでは編集の分岐点です。

複数人で１つのリポジトリを編集する際は、並列に実行できる処理をブランチで分けます。

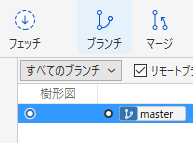
こうすることで、互いの作業が影響し合わないようにします。

最後に複数のブランチをマージすることで、１つのブランチに統合します。

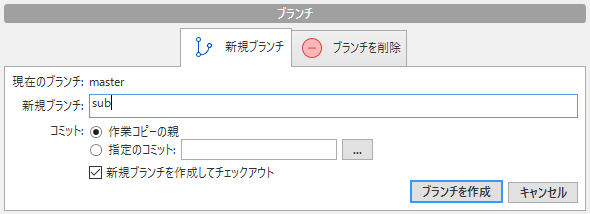
## 分割

SourceTreeでmasterブランチを分割します。

「ブランチ」をクリックしてブランチ画面を開きます。



「新規ブランチ」で分岐するブランチの名前を付けます。



[ブランチを作成]をクリックして作成します。

それぞれのブランチで変更が加わると分岐が発生します。

現在のブランチは白い点で表されます。

ダブルクリックで現在のブランチから移動できます。



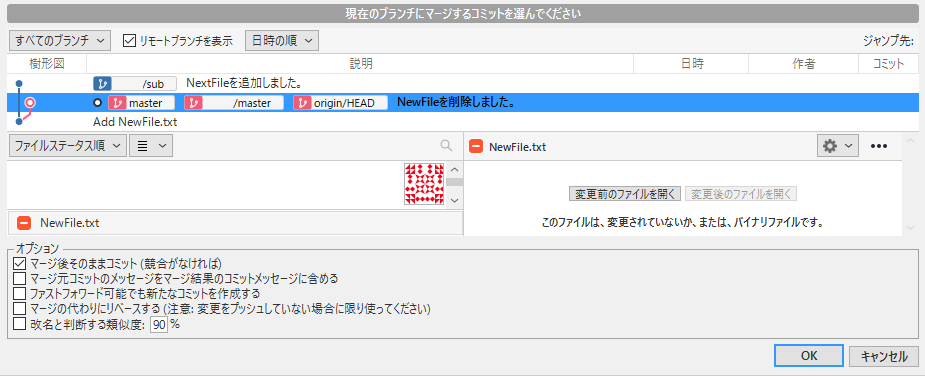
## 統合

masterブランチとsubブランチを統合します。

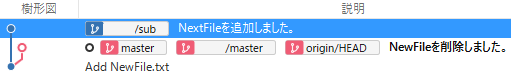
「マージ」をクリックしてマージ画面を開きます。



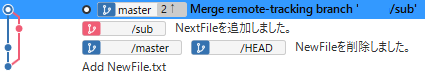
「樹形図」からマージ先のブランチを選択します。



現在のブランチ(白い点)に選択先(青く選択された)ブランチを統合します。

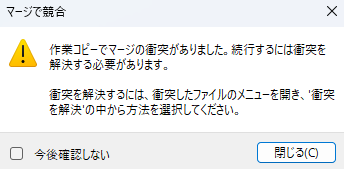


[OK]をクリックして統合します。



## 競合

マージでは同じファイルに対して別の変更が加わった際、競合が起きます。



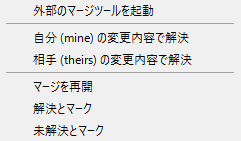
「作業ツリーのファイル」や「Indexにステージしたファイル」などの一覧から探します。

競合が起きたファイルを右クリックして、「競合を解決」から方法を選びます。



この場で解決できる方法は「自分」(master) 優先または「相手」(sub) 優先での解決です。

基本は再度ファイルを編集し直すことをおすすめします。

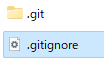


解決するとマージはされますが、コミットは別で行います。

# .gitignore

「.gitignore」ファイルはGitで追跡しない(管理しない)ファイルを指定するファイルです。

対象のフォルダの直下に配置されます。



.gitignoreには追跡しないファイルやディレクトリを指定します。

特定の名前のファイル「File.txt」を追跡しない場合は以下のように記述します。

File.txt

.gitignoreと同じディレクトリの「File.txt」の場合。

/File.txt

.gitignoreと同じディレクトリにある「Dir」ディレクトリ内の「File.txt」の場合。

/Dir/File.txt

任意のファイル名で特定の拡張子「.a」のみ追跡しない場合は「\*」を使用します。

\*.a

特定の名前のディレクトリ「Dir」以下のファイル全てを追跡しない場合。

Dir/