Git 超入門

お断り

わかりやすさ優先のため厳密でない表現があります

概要

Git とは

ソースコードの

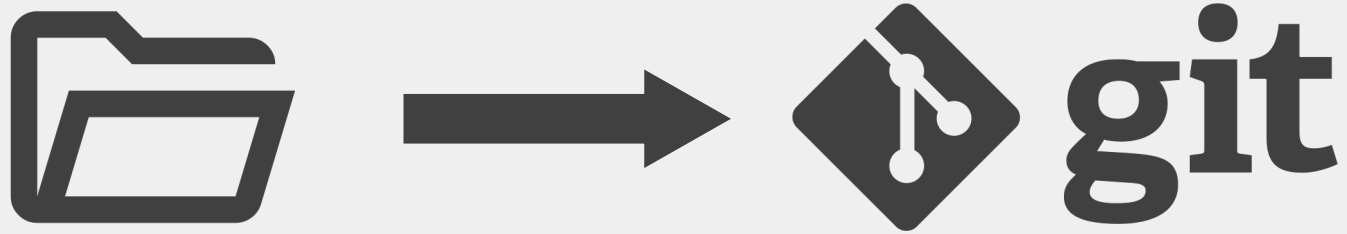
バージョン管理ツール

Gitでなにができる？

- 編集履歴の保存
- ファイル・編集履歴の共有
- 時間の巻き戻し
- 違う世界線を作成 (?)

Gitの始め方

自分で決めたディレクトリ内の管理をGitに任せる形



プロジェクトの
ディレクトリ

Gitが無いと...

バージョン管理したい
ファイルA

A.py

```
while True:
  print(“Hello, world!”)
```

Gitが無いと...

バージョン管理したい
ファイルA

A_最終版.py

中身を編集

```
# while True:
print(“Hello, world!”)
```

Gitが無いと...

バージョン管理したい
ファイルA

A_最終版_デラックス.py

さらに中身を編集

```
# while True:
print(“こんにちは!”)
```

Gitが無いと...

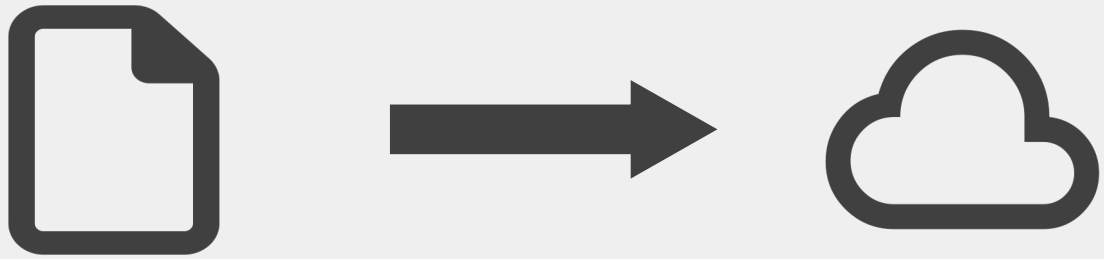
A_ver1.py
A_最終版.py
A_最終版_デラックス.py
A_最終版_ウルトラデラックス.py

Gitが無いと...

A_ver1.p
A_最終版
A_最終版
A_最終版

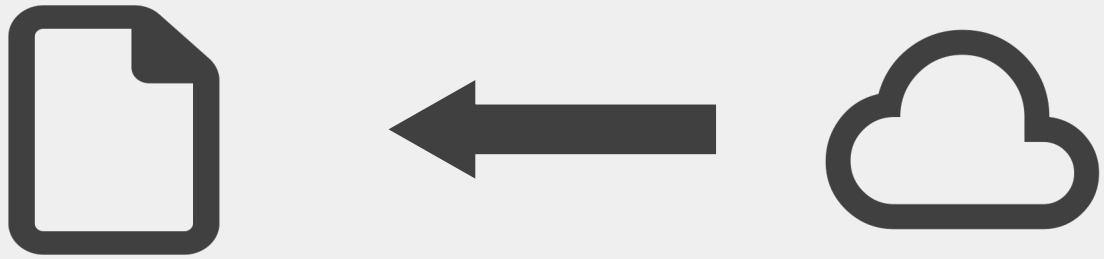
更新する度に
ファイルが増えていく
(管理や共有が困難)

Gitの仕組み – 図解



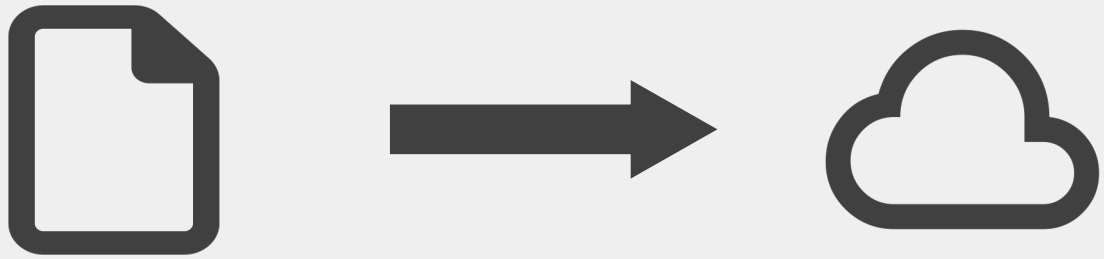
常に最新版はクラウド上

Gitの仕組み – 図解



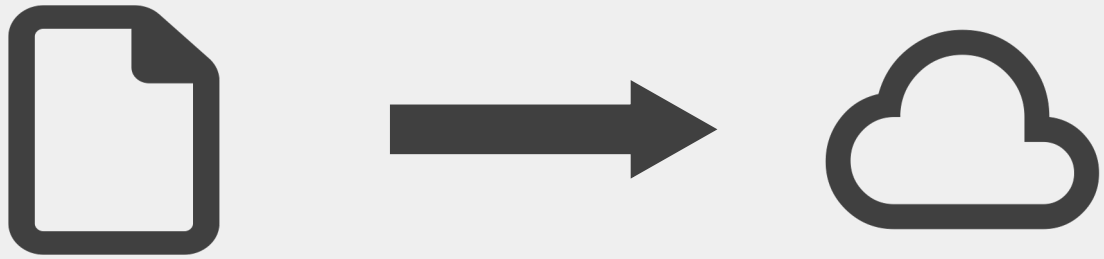
最新版をダウンロードして編集

Gitの仕組み – 図解



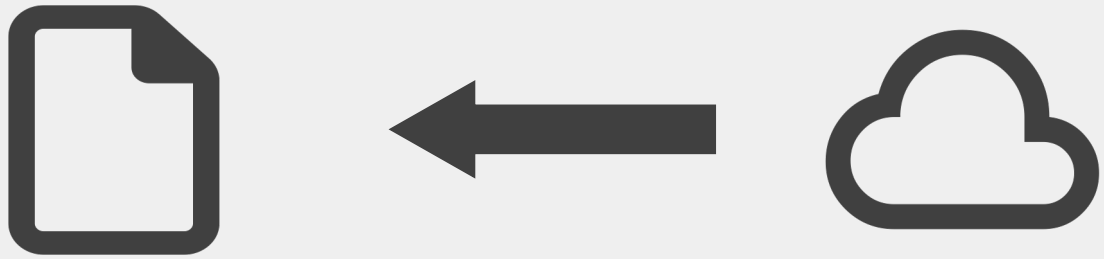
編集したらクラウドへアップロード

Gitの仕組み – 図解



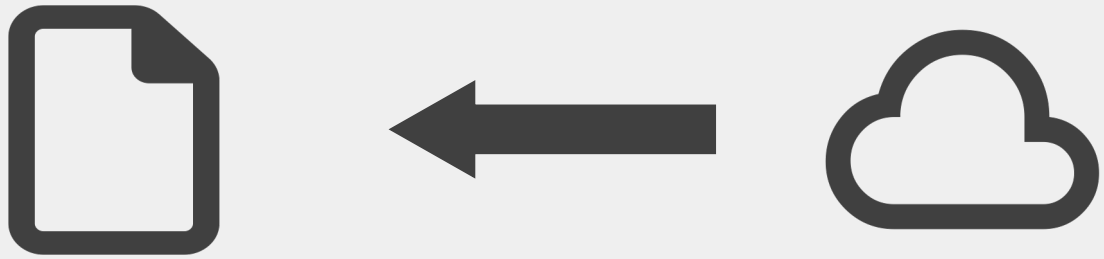
この時，前回との差分のみが送られる

Gitの仕組み – 図解



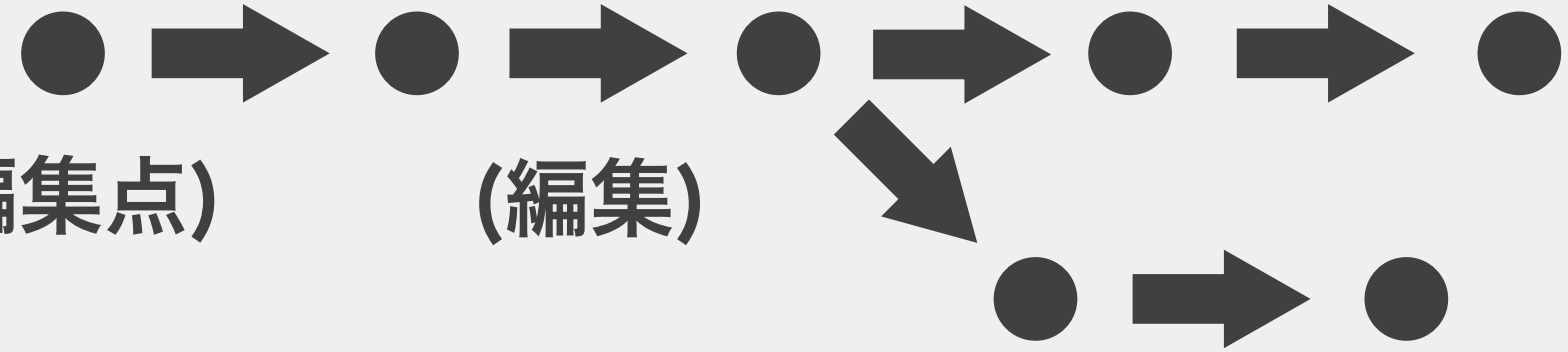
保存した任意の時点まで巻き戻せる

Gitの仕組み – 図解



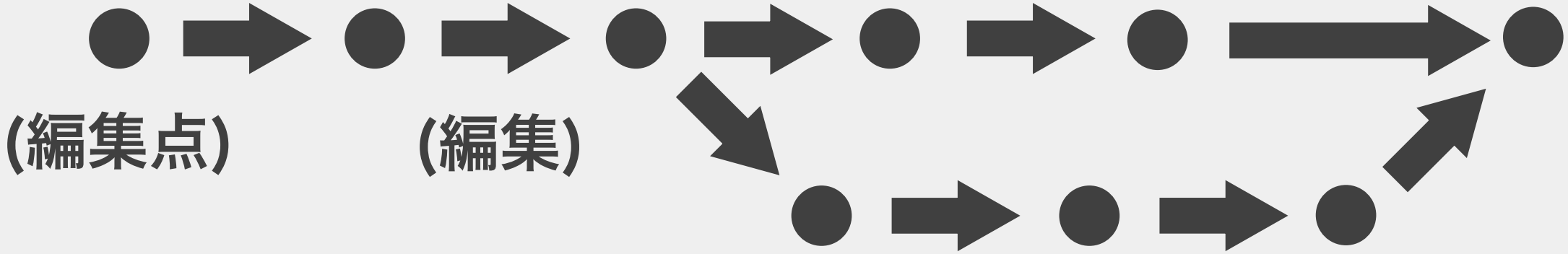
クラウド上にあるため共有が簡単

Gitの仕組み – 図解



編集履歴を分岐させることが可能
(複数の世界線を作成ができる)

Gitの仕組み – 図解



複数の世界線の融合が可能

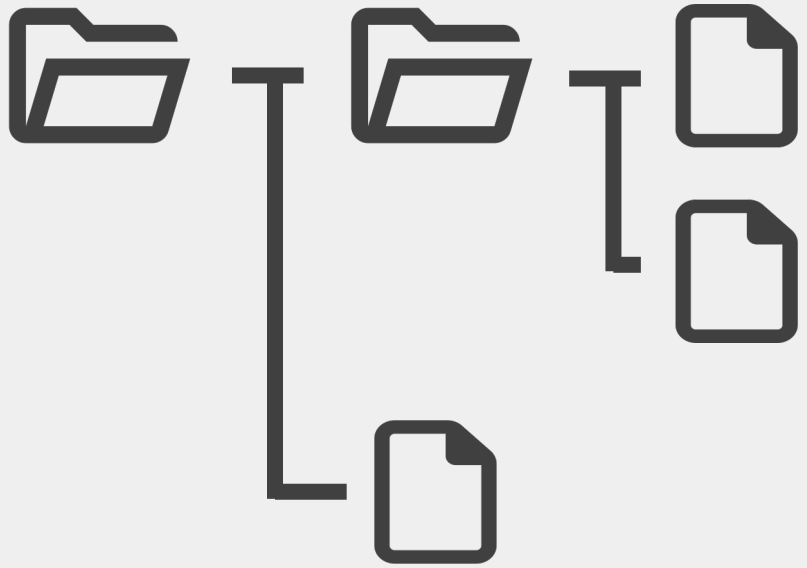
用語・用法

Git用語一覧（抜粋）

- (リモート|ローカル)リポジトリ
- クローン
- フェッチ
- コミット
- プッシュ
- プル
- ブランチ
- マージ

リポジトリとは

Gitに管理を任せているディレクトリ



ローカルリポジトリとは

手元(自分のPC)にあるGitで管理しているフォルダ

自分のPC上での編集(ローカルリポジトリ上の編集)は

自分のみに反映される

リモートリポジトリとは

クラウド上におく最新版のリポジトリ

ローカルリポジトリで編集したものをこれに反映させる

他の人と共有されている

(Google Drive上に作ったフォルダみたいなイメージ)

クローンとは

リモートリポジトリを自分のPCにダウンロードをすること

{誰か一人が作成したリポジトリを共同編集者がクローン(手元に落と)して共同開発を始める}

コミットとは

ローカルリポジトリ上で編集履歴を保存すること

コミットメッセージと共に保存する

過去に行ったコミットの時点の状態に戻すことが可能

(なるべく細かく，キリの良いタイミングで行うことが好ましい)

プッシュとは

ローカル上のコミットをまとめてリモートへ
反映させること

(これによって初めてリモートに変更が反映され他人が
自分の編集が反映できるようになる)

フェッチとは

リモートリポジトリに更新があるかどうか確認すること

プルとは

リモートの状態をローカルへ反映させること

フェッチによってリモートの更新が確認された際に行う

(ローカルリポジトリにリモートリポジトリとの
差分のみを反映する)

基本的な使われ方

1. 誰かが(リモート)リポジトリを作成
2. 共同開発者がそれをクローン
3. それぞれが作業を行う

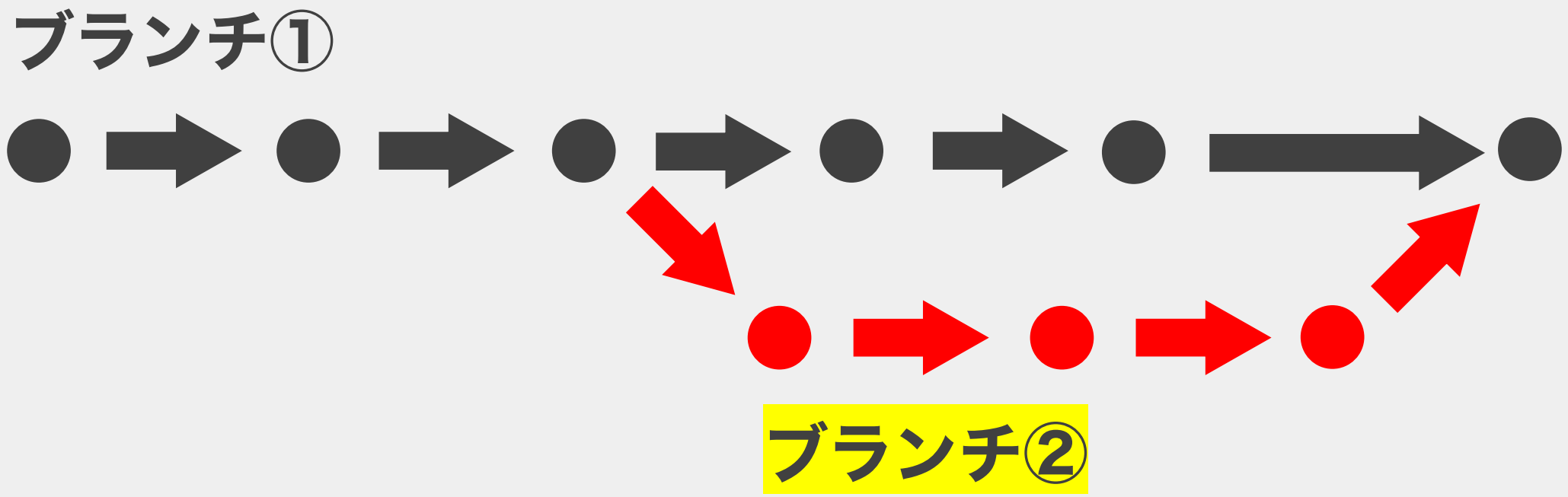
基本的な開発の流れ

1. ローカルリポジトリで作業 (今まで通り)
2. 編集履歴を保存 (コミット)
3. リモートリポジトリに反映 (プッシュ)
4. 他人の編集した最新版をローカルに反映 (プル)

この4つの繰り返し

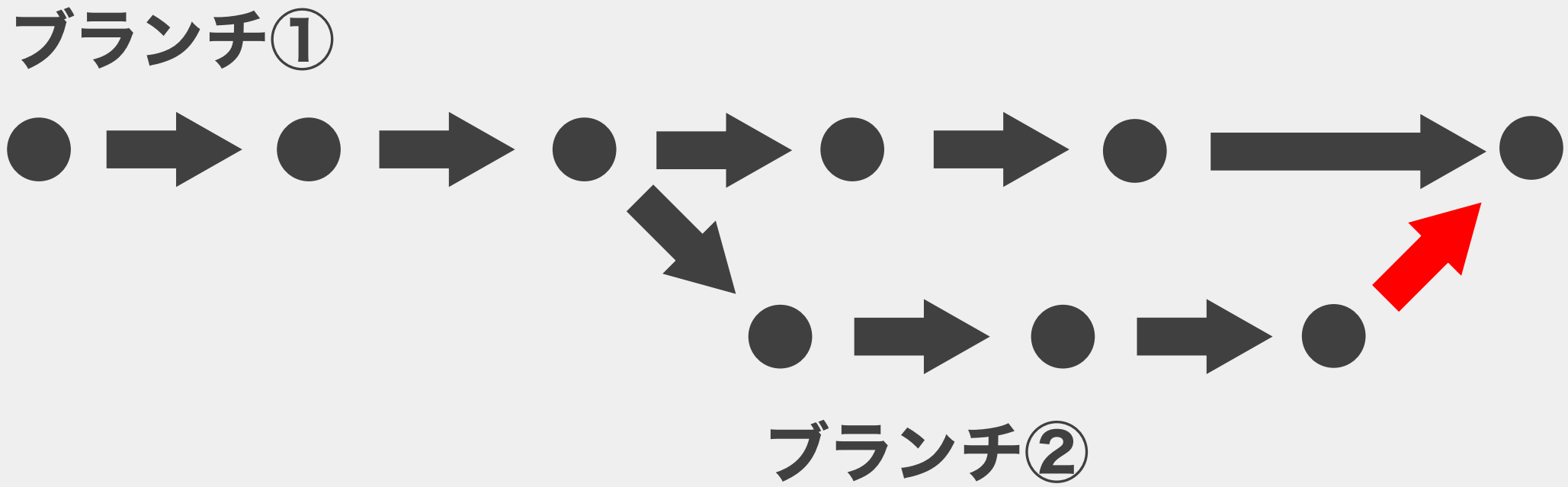
ブランチとは

今の状態を丸々複製した違う世界線



マージとは

別々の世界線を融合させること

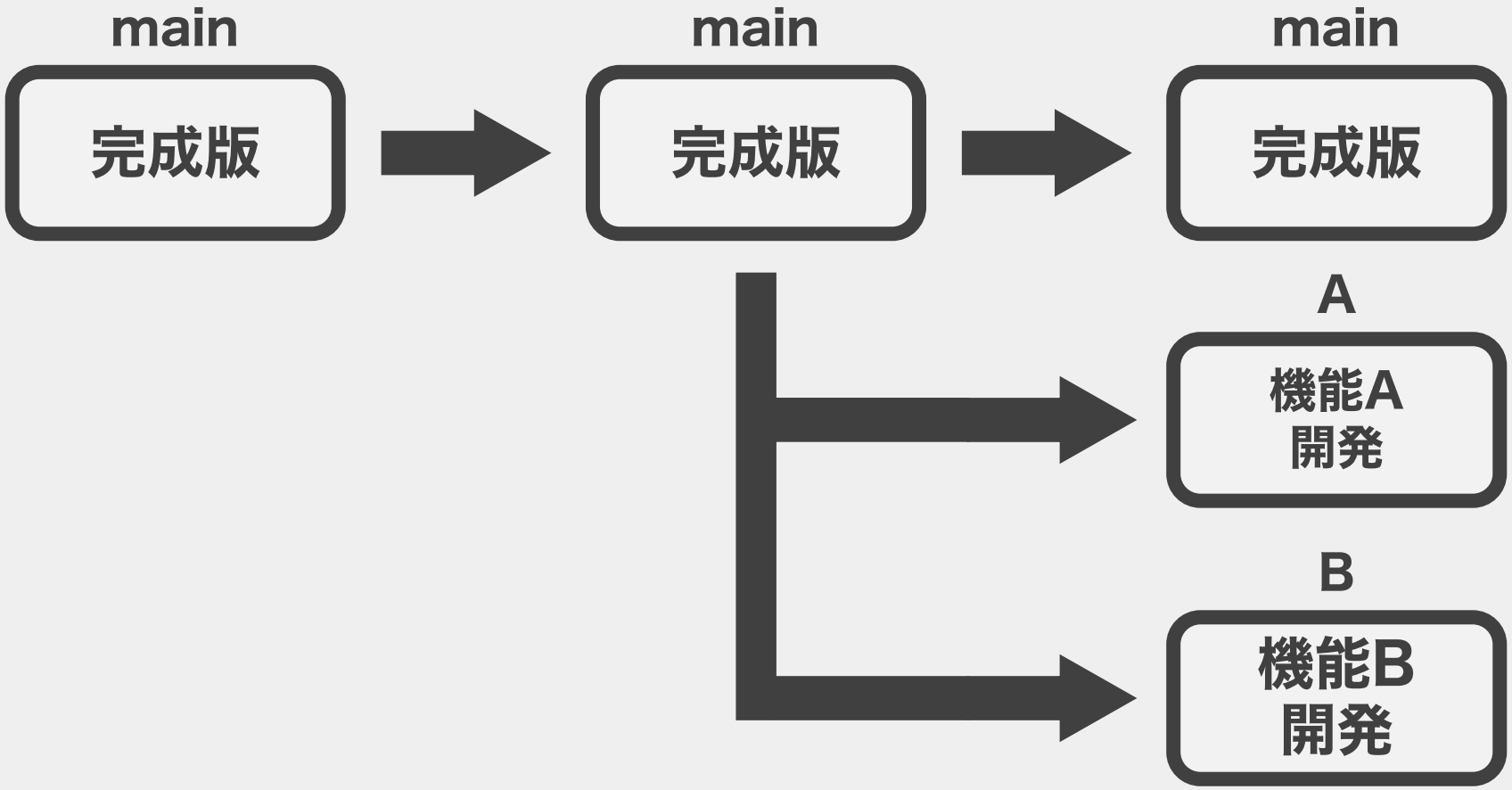


ブランチがあると何が嬉しいか

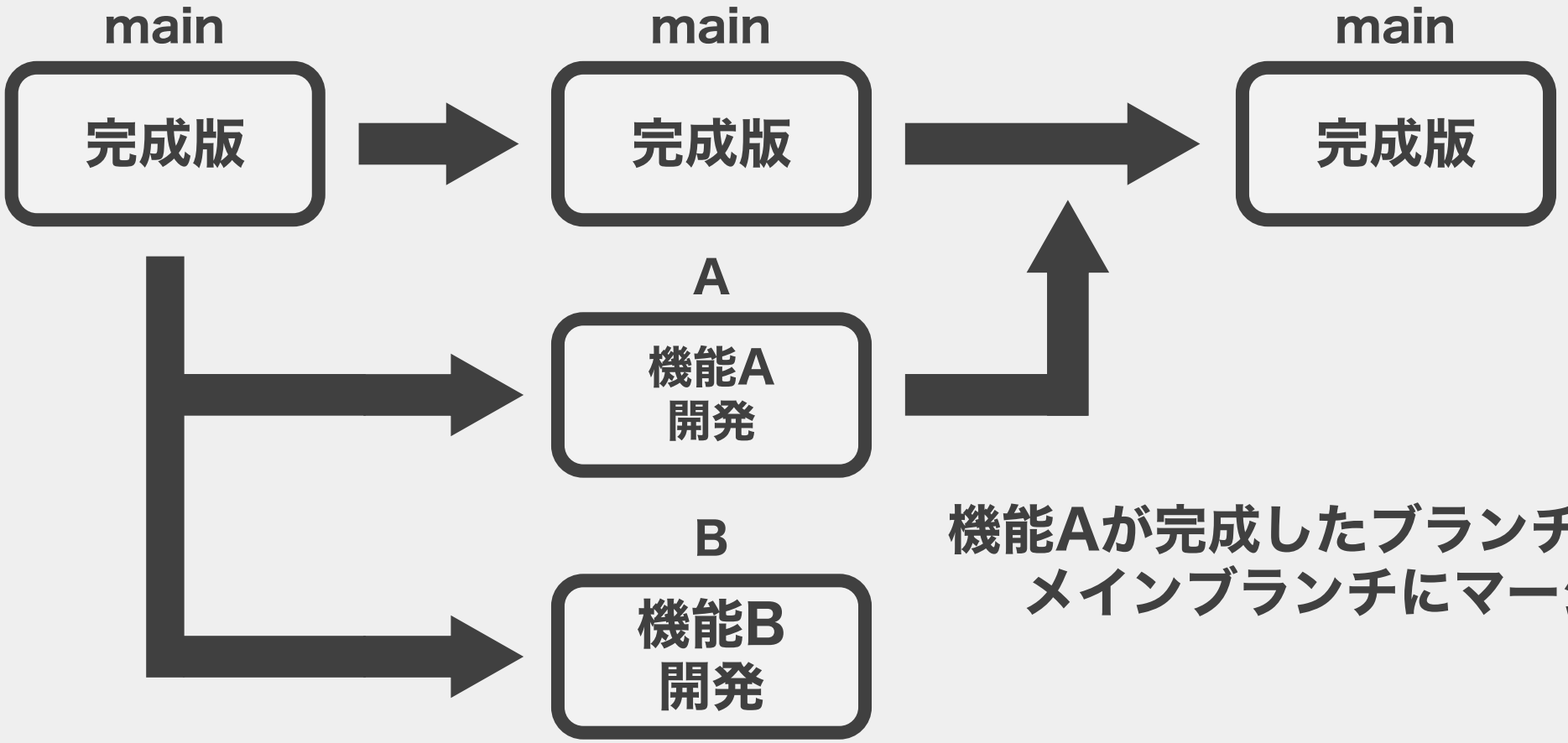
- プロジェクト本体に影響を与えずに作業ができる
- 目的ごとに並行して作業ができる

  → 共同作業が用意になる
- 不具合が起きた時の保険になる

ブランチの利用例



ブランチの利用例


main
完成版
main
完成版
main
完成版
A
機能A
開発
B
機能B
開発
機能Aが完成したブランチAを
メインブランチにマージ

ブランチの使われ方

大抵の場合においてデフォルトブランチ(main)は

『動くもの』にするため，ブランチを切って(作成して)

そこで開発作業を行う

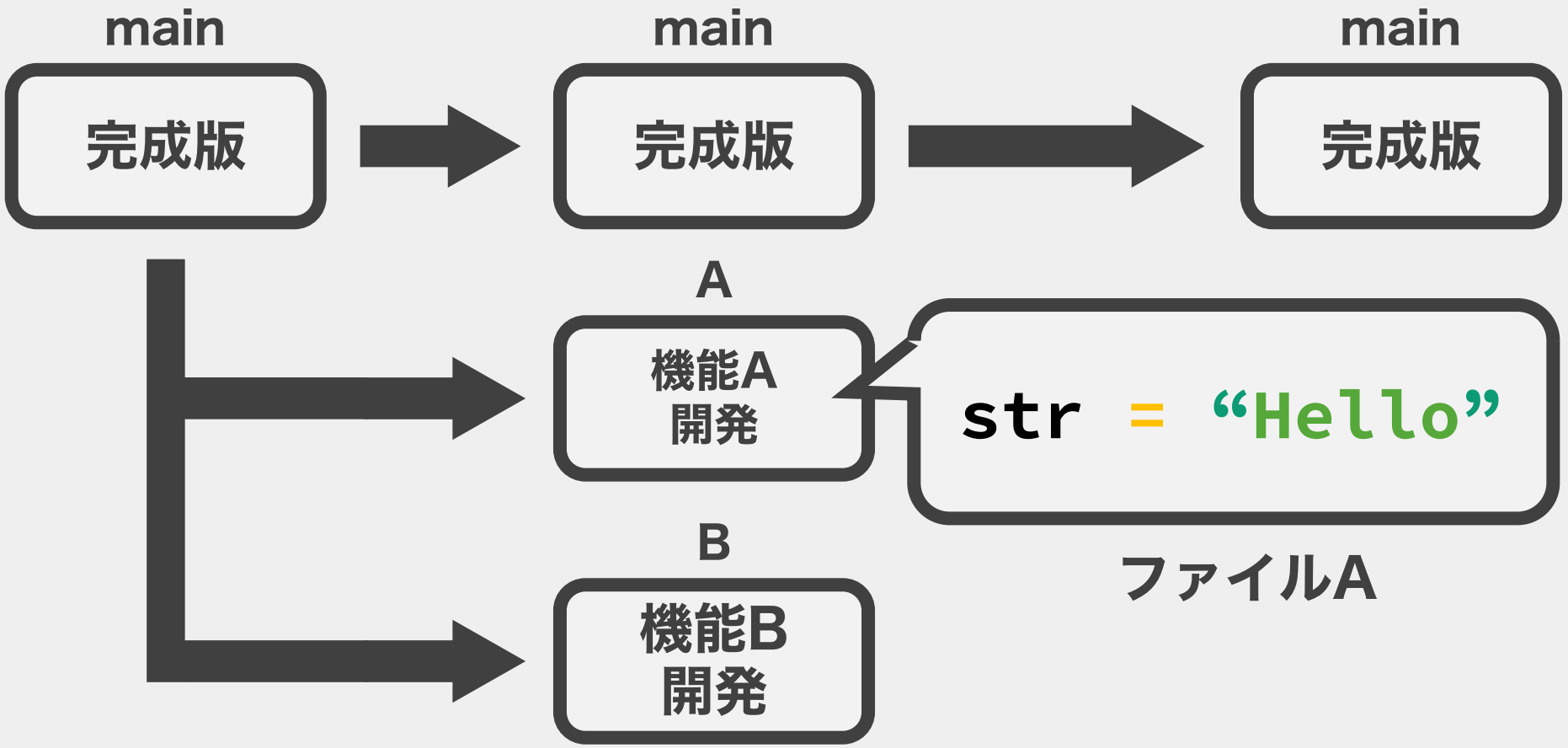
機能・修正などの単位でブランチを切って作業をする

コンフリクトとは

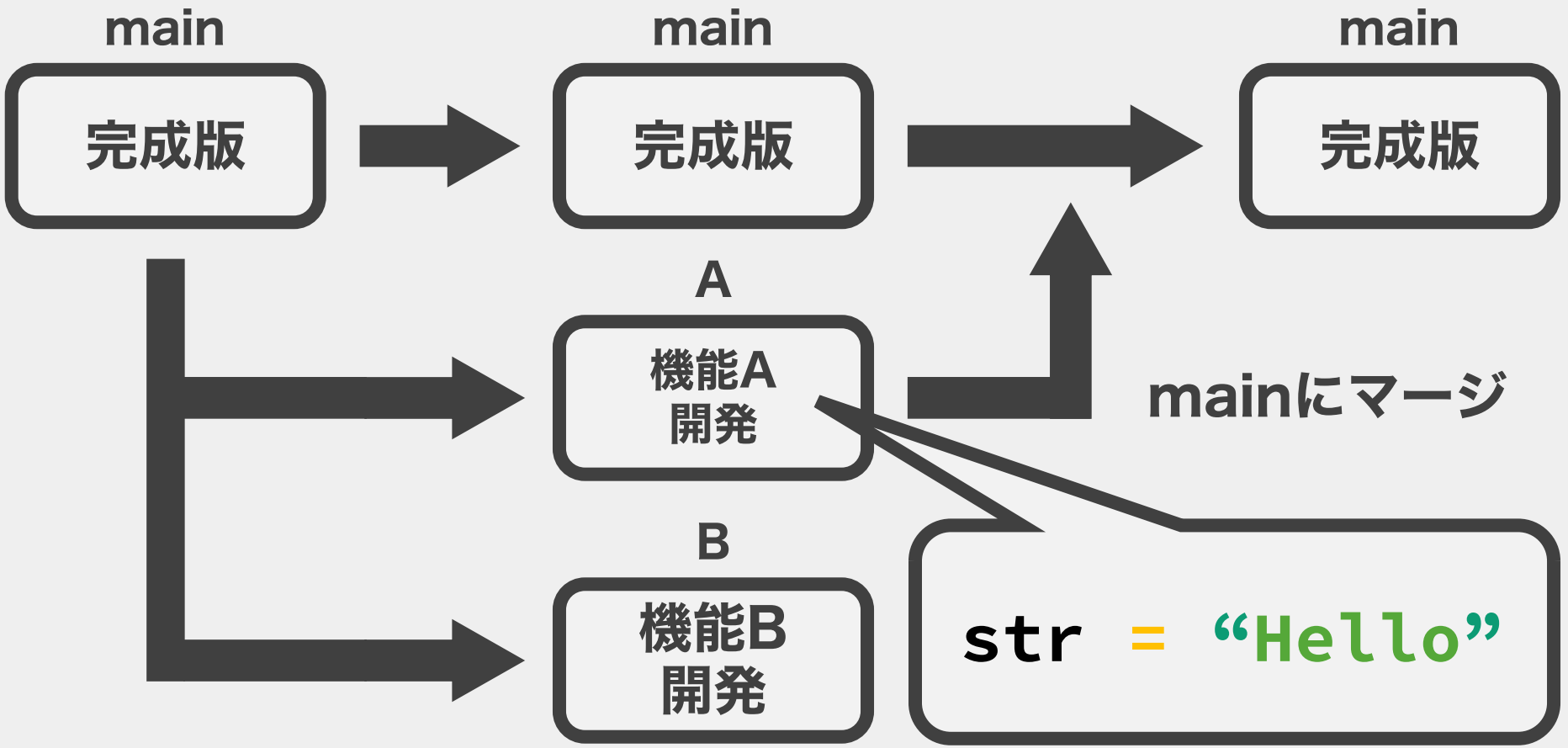
基本的にマージは自動で行われるが，同じブランチから派生した複数ブランチで同じファイルが編集されるとコンフリクト(衝突)が発生する

この場合，マージ時に手動でコンフリクトを解消する必要がある

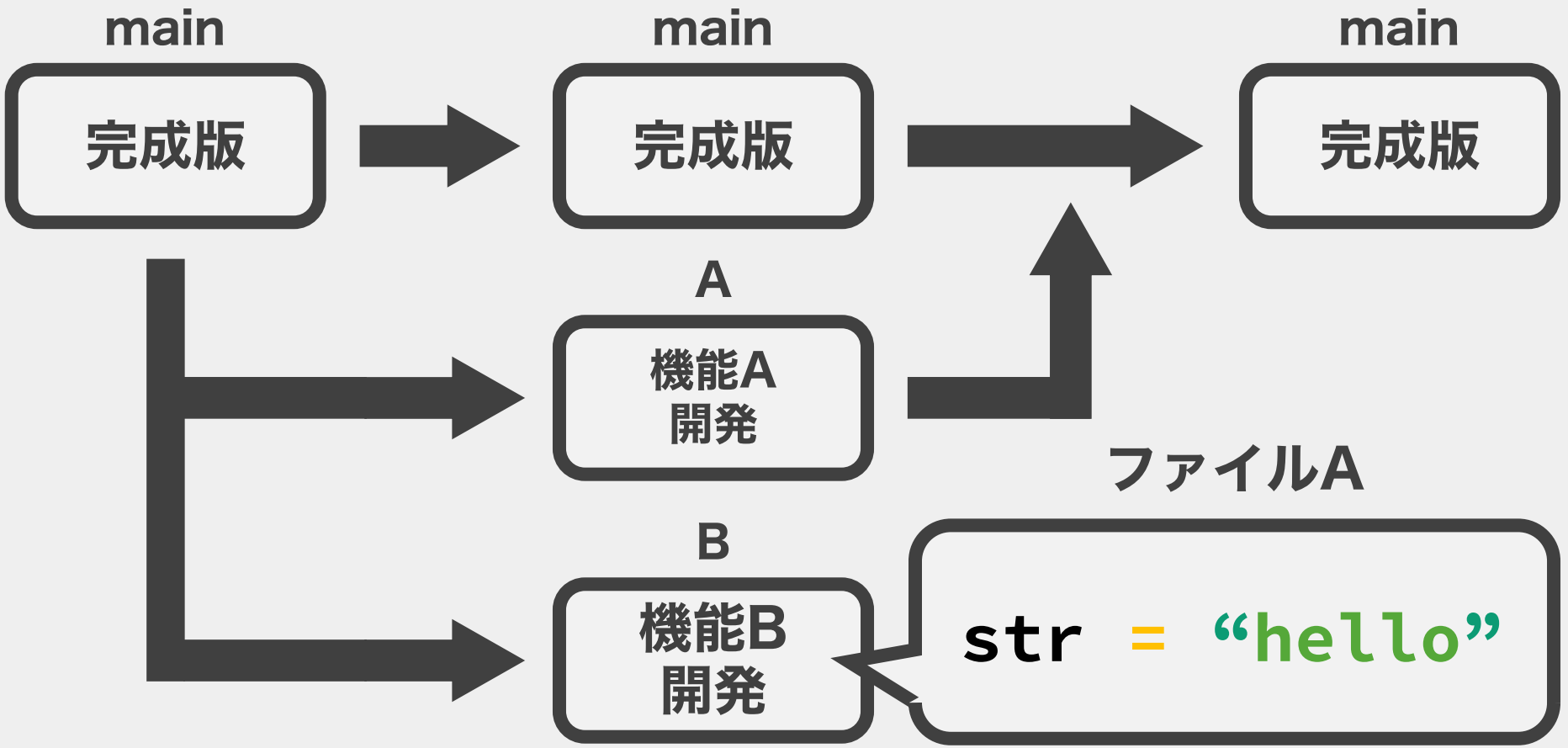
コンフリクトの例



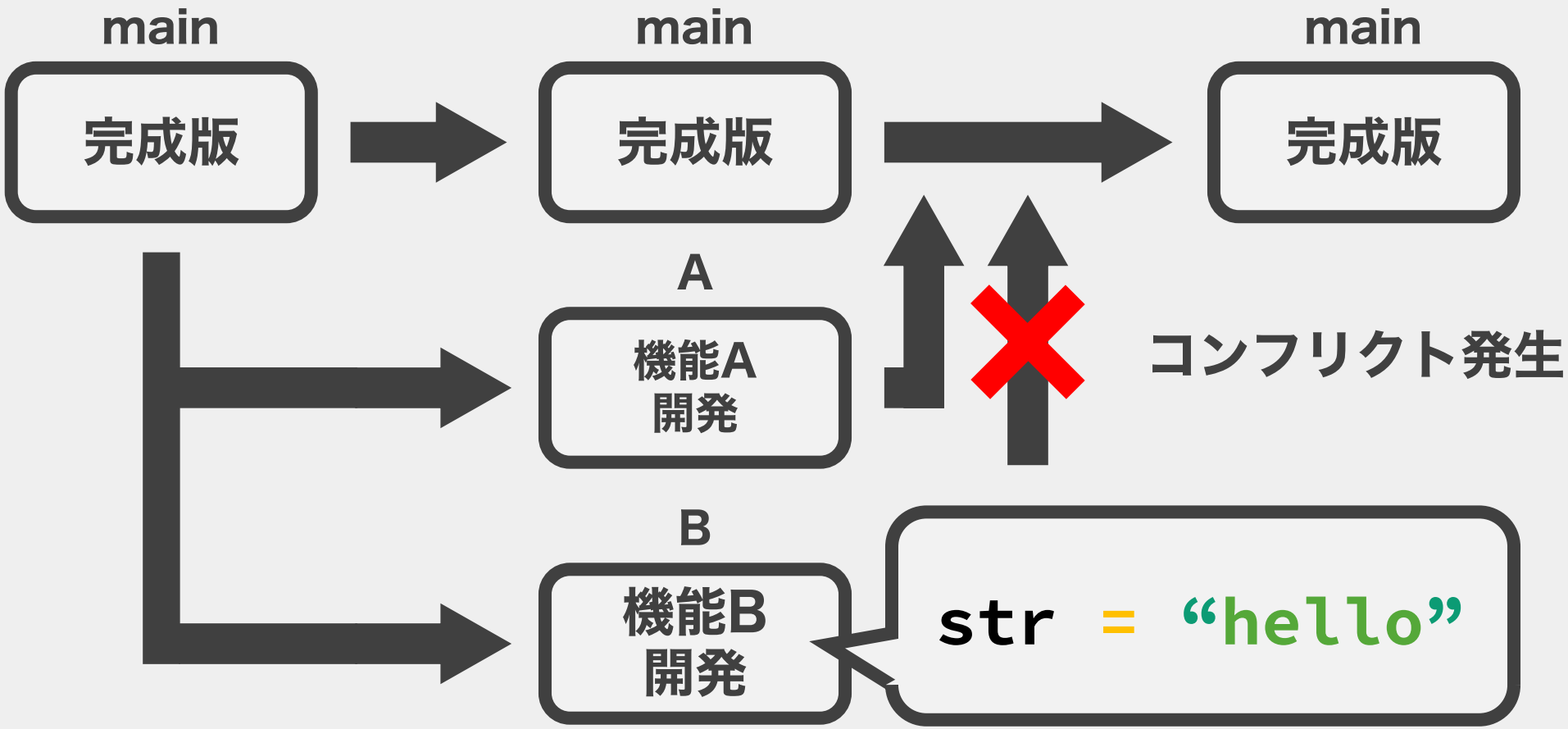
コンフリクトの例



コンフリクトの例



コンフリクトの例



コンフリクトの例

このような状況が起こると，マージ時にコンフリクトが起きている行は以下のようになる

ファイルA

```
<<<<<<< HEAD
str = “hello”
=======
str = “Hello”
>>>>>>> main
```

コンフリクト箇所のみ抜粋

コンフリクトの例

これが表示されたらこの箇所の最終的に反映されたい

部分のみを残し，コミット・プッシュ・再マージをする

ファイルA

```
<<<<<<< HEAD
str = “hello”
=======
str = “Hello”
>>>>>>> main
```

要らない行を削除

コンフリクト箇所のみ抜粋

プルリクエスト(PR)とは

自分が編集しリモートへプッシュした内容について

共同編集者にプル・検証してもらうためのリクエスト

他のブランチに自分の一連の作業をマージする提案のこと

プルリクエスト(PR)とは

プルリクエストでは他の共同開発者にコードのレビューを求めることができる

この仕組みによって他の共同開発者に変更点をまとめて伝え，議論することが容易になる

ignoreファイルとは

Gitで管理したいディレクトリ下にGitで管理する
必要のないファイルやディレクトリがある場合は
.gitignoreというファイルに記述することで
Gitでの管理から除外することができる

(指定にはワイルドカードが使用可能)

ignoreファイルとは

GitHub等のサービスでリポジトリをホストする場合,
リポジトリの設定によっては赤の他人からコードや
コミット履歴等を閲覧することが可能
→ パスワードやトークン等の秘密情報をコミットや
　　プッシュするのはとても危険なため除外する

gitignoreで除外する物の例

- Buildディレクトリ (各々の環境に合わせたビルド)
- .DS_Store (macOSのディレクトリ情報ファイル)
- ENVファイル (DBパスワード等)
- tempファイル (一時ファイル)
- logファイル (実行ログ)

gitignoreで除外する物の例

機密情報が

書かれたファイル

gitignoreの書き方

```
# macのディレクトリ管理用隠しファイル(.DS_Store)を除外
.DS_Store

# .exeのファイルを除外
*.exe

# buildディレクトリを除外
build/

# packageディレクトリの.tsファイルを除外
package/**/*.ts
```

GitHubについて

GitHubとは

世界最大のリポジトリホスティングサービス

リモートリポジトリをこのサービス上に保存する

同様のサービスにはGitLab，Bitbucket,
Assembla等がある

(Google Driveのような存在)

Gitの運用について

本来GitはCUIツールであるがコマンドを覚えるのが大変な上，実際の使用イメージが掴みづらいため多数のGUIツールが提供されている

```
$ git commit -m “commit message”
$ git push -u origin master
```

といったように本来はコマンドで使う

GUIツール

今回はGitHub Desktopを用いてGitを扱うこととする

GitHub Desktopについて

GitHub社が公式に提供しているGUIツール

前述したGitの操作をGUIから行うためのソフトウェア

詳細な使い方については解説記事を参照されたい

【入門】Github Desktopとは？インストール・使い方

https://www.kagoya.jp/howto/it-glossary/develop/githubdesktop/

参考 / 引用

- サル先生のGit入門〜バージョン管理を使いこなそう〜 | Backlog
  https://backlog.com/ja/git-tutorial/
- 【Git】オレならこう説明する！Git初心者への用語説明 | Qiita
  https://qiita.com/nnahito/items/e546b27f73e7be131d4e
- 初心者のためのやさしいGit | Zenn
  https://zenn.dev/getgotgoto/articles/506bcfbcd55149
- gitignoreの書き方チートシート2021年版 | WiseLoan Engineer Blog
  https://leadingtech.co.jp/wiseloan/gitignore/
- 【入門】Github Desktopとは？インストール・使い方 | カゴヤのサーバー研究室
  https://www.kagoya.jp/howto/it-glossary/develop/githubdesktop/