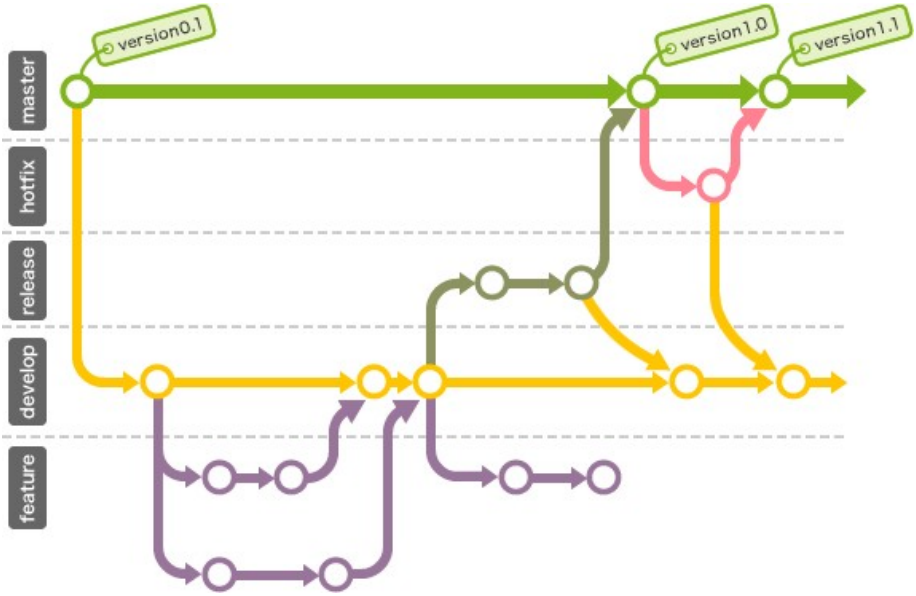
Git／JIRA 運用ルール

https://backlog.com/ja/git-tutorial/stepup/05/

に乗っ取り、以下 5 つのブランチで運用

- main
- develop
- feature
- release
- hotfix



◆ルール

- 基本的な流れは、develop→release→main と develop に反映
- develop は Dev 環境、release は STG 環境、main は PROD 環境で動作確認
- develop で機能追加やバグ修正は feature ブランチを切って対応
- main で緊急バグ対応は hotfix ブランチを切って対応（hotfix→main と develop に反映）
- release から main にマージする際のコミットにタグをつけてバージョン管理。その後、release ブランチは削除する。
- コミットメッセージはわかりやすく簡潔に

◆受け入れ手順（JIRA アジャイルルール）

JIRA スプリントにおいて

「進行中」＝チケットが着手する場合、必ず進捗を進行中に更新してもらう（**開発担当者の作業**）

「開発/解決済み」＝開発者で Dev 環境で動作確認して問題ないことを確認している状態（**開発担当者の作業**）

「リリースまち」スプリントを終了後に、次回のリリース対象チケットをピックアップして、リリースまちの進捗に更新する（**出荷判定者のタスク**）

「完了」＝STG環境で動作確認して問題なしの状態（**出荷判定者のタスク**）

https://qiita.com/ren0826jam/items/8eb8f59c13dbba7863ad

◆ブランチ命名規則

- main ブランチ：main
- develop ブランチ：develop
- feature ブランチ：feature/[チケット番号・機能名・修正名]
- release ブランチ：release/[バージョン番号]
- hotfix ブランチ：hotfix/[修正内容]

◆バージョン

Ver1.2.5＝メジャーバージョン番号.マイナーバージョン番号.パッチ番号

1. メジャーバージョン番号（Major Version Number）：大きな変更があった場合にインクリメントします。例えば、大幅な機能の追加やアーキテクチャの変更、非互換性のある変更があった場合にインクリメントします。
2. マイナーバージョン番号（Minor Version Number）：機能の追加や改善、バグ修正などの小さな変更があった場合にインクリメントします。新しい機能を追加した場合は、その機能がどのようなレベルの機能かを考慮して、マイナーバージョン番号をインクリメントすることもあります。
3. パッチ番号（Patch Number）：バグ修正やセキュリティアップデートなどの小さな修正があった場合にインクリメントします。パッチ番号は必ずしも必要ではありませんが、必要に応じて使用することができます。

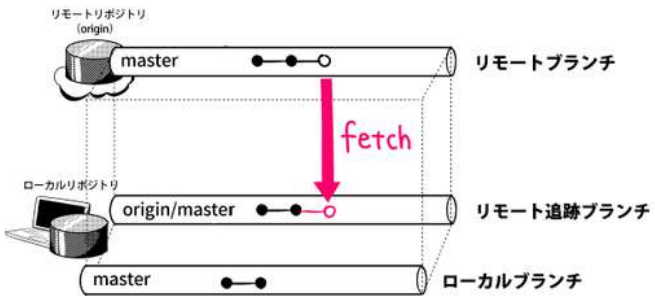
◆JIRA ルール

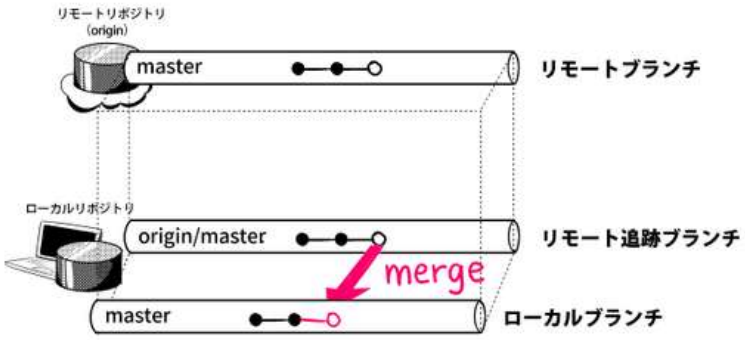
- JIRA チケットの「時間の管理」のところに、開発にかかった時間を書くこと。実際にかかった時間を記録する。
    - 例）14:00-18:00 で開発作業、15:00-16:00 は別打ち合わせなら、3h を記録
- JIRA チケットにブランチとプルリクを紐づけること

◆その他 Git で覚えておく概念

慣れないうちは SourceTree を使ってください。以下全て、SourceTree で簡単に実現できます。

- 過去に戻って新規ブランチを作成、作業をやり直す：**過去のコミットにチェックアウト→ブランチ作成**
- 過去のコミットを打ち消す：**リバート**
- feature ブランチで作業進めている間にリモートの develop ブランチが更新された：**pull してローカルでリベース**
    - **※GitHub にプッシュしたコミットをリベースするのは絶対に NG！！プッシュできないが、強制プッシュすると GitHub 履歴が壊れる**
    - **→→プッシュしていないローカルの変更にはリベース、プッシュした後はマージを使うこと！！→**https://qiita.com/riku929hr/items/15415d34ee5fc412c126
    - 更新のあった develop を feature にマージする手順（安全）
        1. ローカルで develop にチェックアウト
        2. プルをして develop を最新にする
        3. 取り込む側のブランチ(feature)に移動
        4. develop をマージ
        5. プッシュをする
- 複数のコミットを一つにまとめたい：**インタラクティブリベースモードにしてスカッシュ**
- pull で実際やっていること：リモートから origin/main というリモート追跡ブランチをローカルに作成している。その後、ローカルの main ブランチにマージしている。（origin はリモートリポジトリのデフォルト名）





- 直近のコミットメッセージを修正：**amend**
- 緊急案件が来た。作業中の未コミットのファイルを一時退避したい：**スタッシュ**
- 別ブランチから特定コミットのみ取り込む：**チェリーピック**