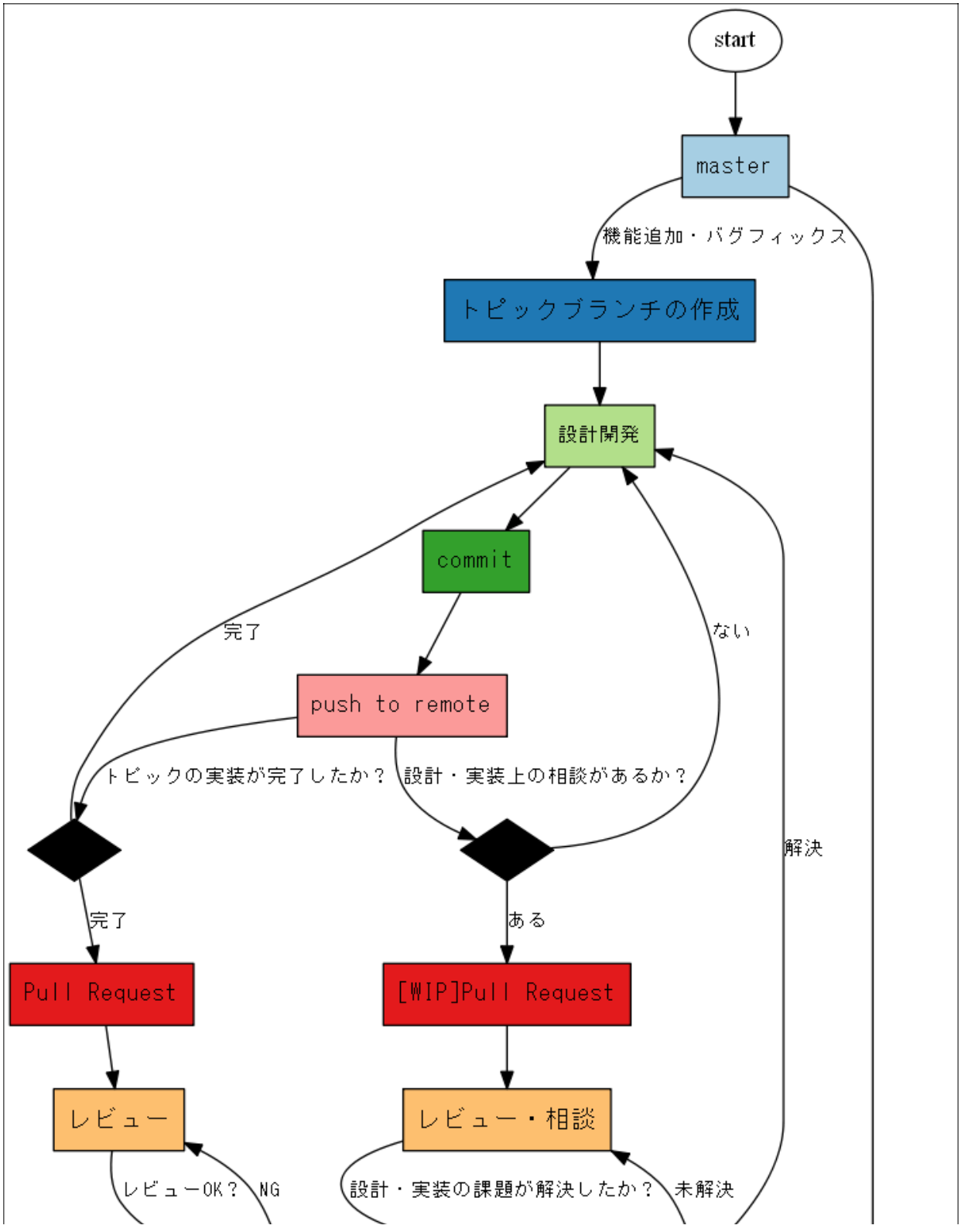
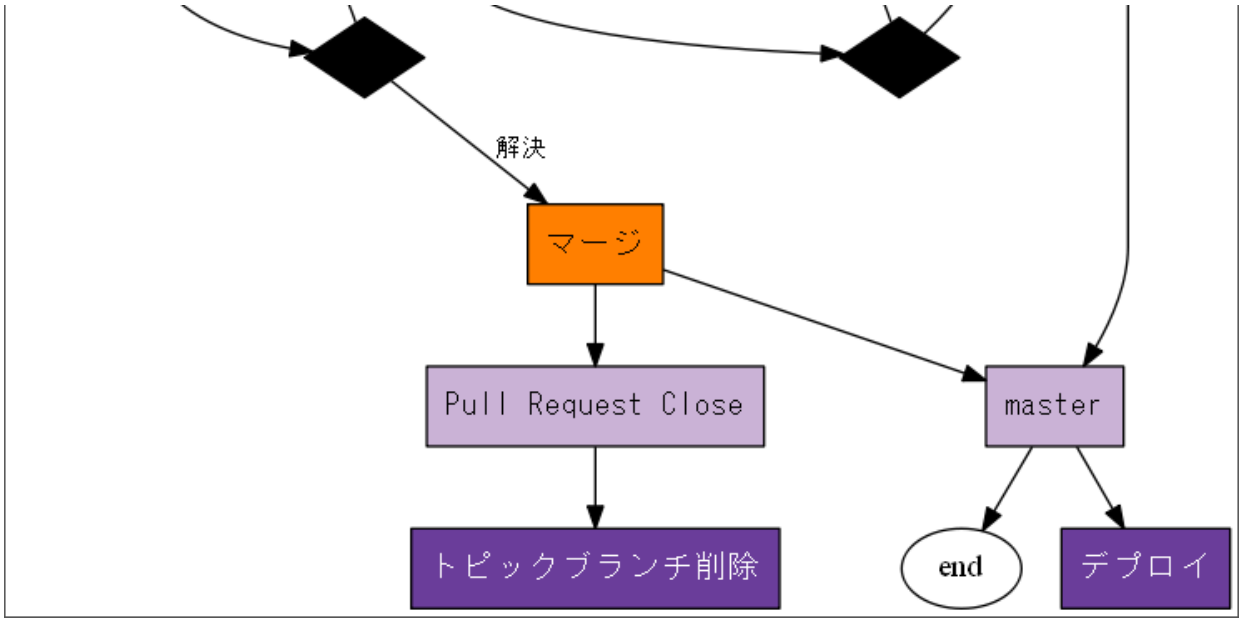
Github Flowについて　素人にもわかるようにまとめる





大まかな流れは

- masterブランチから新しくブランチを切る
- 変更を加える
- commitする

  変更を加えコミットする際は、コミット１つ１つのサイズを小さくし、それぞれコミットの意図を伝わるようにする。
- pushする

  これのコミットを定期的にpushすることでバックアップにもなり、開発者同士のコミュニケーションにもなる。
- プルリクエストを出す

  masterブランチへマージする直前だけでなく、適宜フィードバックを貰いつつ修正していくべき。
- レビューしてもらう
- 問題なければマージする(してもらう)
- リモートブランチを削除する

以上のように、基本的には特定の作業をするブランチを作成するだけなので、作業を始めてからデプロイするまでの工数が少なくシンプルで学習コストが低い。したがって、多くの開発者の迅速な作業を可能にして、小さな変更などにも柔軟に対応できる。

これのコミットを定期的にpushすることでバックアップにもなり、開発者同士のコミュニケーションにもなる。この工数であれば開発をしてプルリクエストを送るということを容易にでき、レビューやテストが重要視される昨今にマッチしたモデルである。

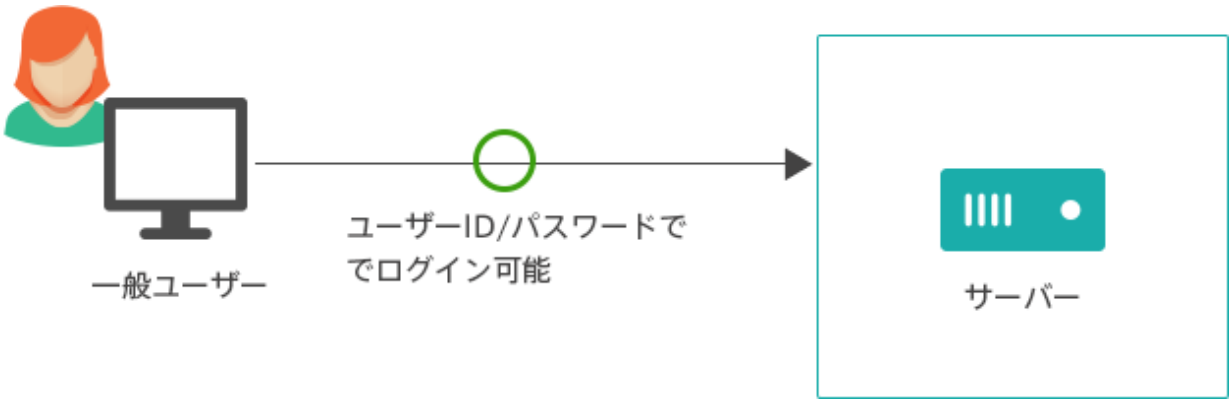
http://komaken.me/blog/2013/09/09/git-flow%E3%81%A8github-flow%E3%81%96%E3%81%A3%E3%81%8F%E3%82%8A%E3%81%BE%E3%81%A8%E3%82%81/

公開鍵認証について素人にまとめるようにまとめる

暗号や認証の技術を利用して、安全にリモートコンピュータと通信するためのルールであるSSHの認証方式

に「パスワード認証方式」と「公開鍵認証方式」の２つがある。

一般ユーザーはパスワードがわかれば誰でもログインできてしまう。



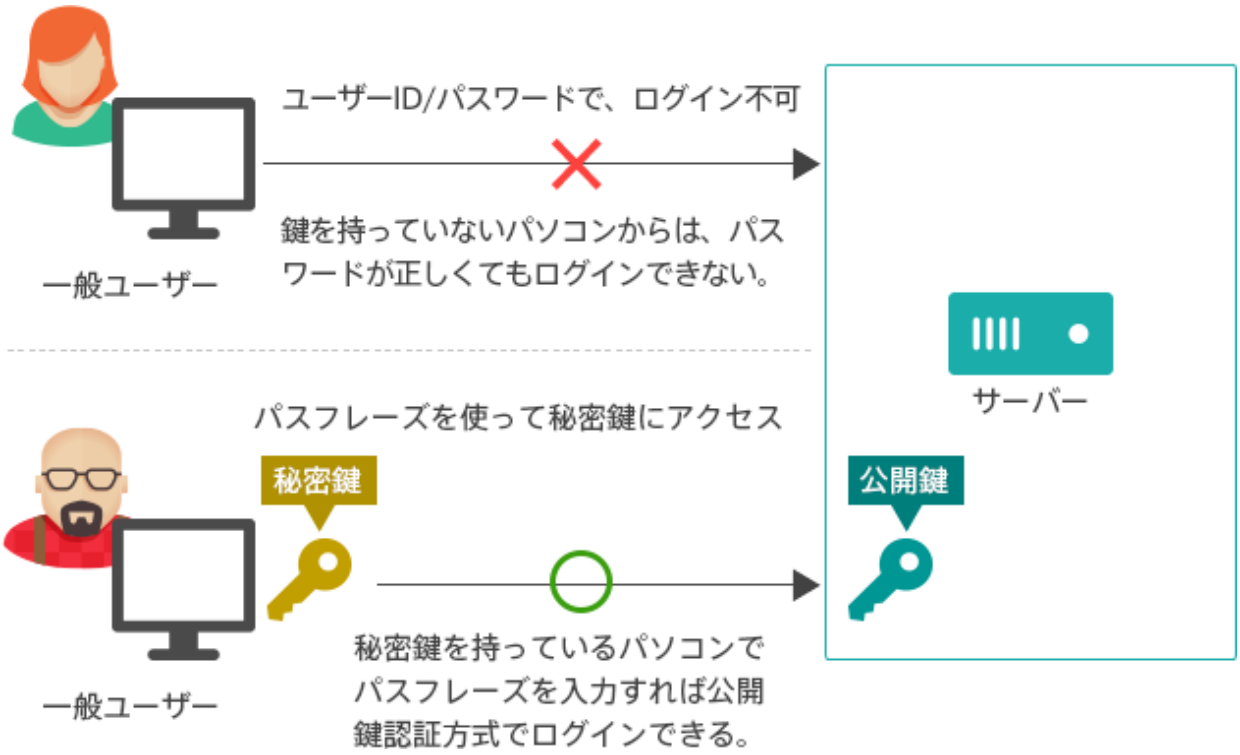
これを公開鍵認証を使って「鍵を持っているパソコンからのみ、一般ユーザーによるログインを許可する」という方式に変更できる。これによってパスワードの総当たり攻撃や、第３者に不正にログインをここ見られても鍵が盗まれない限り破られることはない。

公開鍵認証でログインするには鍵が２つ必要で、パソコン側に置く秘密鍵ファイルとサーバー側に置く

公開鍵ファイルがログインに必要となる。

公開鍵認証ではsshのログインの時に使ったパスワードの代わりにパスフレーズを使用する。

サーバーへログインするパスワードと似たようなもので、自分が決めた文字列を設定できるが、これはサーバーへログインするためのものではなく、秘密鍵にアクセスするための秘密の文字列。



公開鍵暗号方式で暗号化と複合化(暗号化されたデータを元に戻し、読める状態に戻すこと)を行う場合は

- 公開鍵で暗号化したデータは秘密鍵でしか複合できない
- 秘密鍵で暗号化したデータは公開鍵でしか複合できない

以上のような特徴がある。公開されている鍵は公開鍵なのでデータの送信者も悪意のある第三者も取得できる鍵は公開鍵だけ。そして公開鍵暗号方式の特徴として最初に記載した通り公開鍵で暗号化したデータは秘密鍵でしか復号化できない。その為、公開鍵を使って暗号化されたデータを悪意のある第三者が盗み見したとしても、持っている公開鍵では復号化することができない。このように公開鍵は誰に知られても問題ないので、送信側に公開鍵を渡す時に誰かに盗み見られる心配をする必要が無く単に公開しておけばいいだけ。

https://milestone-of-se.nesuke.com/sv-advanced/digicert/public-private-key/

その他　今回の授業で学んだことを記述

　一人で淡々とコードを書き進めるより、誰かと一緒ににペアプログラミングしたりフィードバックを貰いながら進めたことでそれまで知らなかった知見を発見できた。