github

開発開始時

リポジトリのデータ更新を確認、取得する

git pull (origin)

ブランチを切る(すでに作成済みの場合は、移動する)

ブランチの名称は feature/XXXとなる。

git branch feature/【ブランチ名】

※feature/XXX(XXXは任意の文字列)という名称のブランチを作成する

例:feature/admin ←これがブランチ名称

admin ←これがブランチ名

ブランチを指定したい場合は、ブランチ名称で行う！

ブランチの移動

git checkout feature/【ブランチ名】

・コーディングやテストを行い、ファイルの作成・変更する

開発途中～終了時

追加の前にpullしてデータを取得する(この際、データが更新されている場合がある)

git pull origin 【ブランチ名称】

追加・変更したファイルをリポジトリのファイルに入れる

git add 【追加ファイル名】で認識させる(すべての追加ファイルの場合は -Aを入れる)

branchがfeature/【ブランチ名】なことを確認する。(水色で表示されている)

ローカルリポジトリにコミットする。

git commit -m “[コミット種別] #プルリクエスト番号 要約”

例：git commit -m “[add] #1 〇〇機能の実装”

(commitごとに何をやったかわかりやすくするため、毎回記述する)

●コミット種別

fix：バグ修正

add：新規（ファイル）機能追加

update：機能修正（バグではない）

remove：削除（ファイル）

リモートリポジトリへpush(絶対にmainにpushしない！)

git push origin feature/【ブランチ名】

開発完了後

Rull Requestを出す(完成した場合のみ)

タイトルを [ブランチ名]　要約

内容(Write)には、概要とmerge受け入れ要件を書く

Create pull requestを押す

☆チームで同じブランチで作業する場合

共通の機能の作業を行う場合は、ブランチ名を同じにして同じブランチでpush、pullしあうのが望ましい(お互いに別の箇所を開発しながら進められ、問題があればすぐに戻れるようにするため)

[作業の流れ]

・１人がブランチを作成、addしてcommit、pushを行う

git branch feature/【ブランチ名】

git checkout feature/【ブランチ名】

git add -A

(git status)

git commit -m “[コミット種別] #プルリクエスト番号 要約”

git push origin feature/【ブランチ名】

・その他のメンバーは、git pullでブランチを取得

git pull origin feature/【ブランチ名】

・checkoutでそのブランチに移動して開発を行う

git checkout feature/【ブランチ名】

[case1:誰かに更新データを渡したいとき]

・ブランチが正しいか確認する(右側の青色の文字がブランチ名称)

・渡す側は、git pullで更新されているか確認

git pull feature/【ブランチ名】

・渡す側は、更新データをローカルリポジトリ内のフォルダに追加、git addする

データをローカルリポジトリ内のフォルダに追加

git add フォルダorファイル名(ファイル名の場合は拡張子も付ける)

・commitでコメントを必ず書き、pushする。

git commit -m “[コミット種別] #プルリクエスト番号 要約”

・受け取り側のメンバーが、pullをする

git pull origin feature/【ブランチ名】

[不要リスト]

ブランチを指定してデータを取得する場合(まず使わない)

git clone -b ブランチ名 リポジトリurl([git@github.com:YT41-EquipmentSystem/YT41-EquipmentSystem.git](mailto:git@github.com:YT41-EquipmentSystem/YT41-EquipmentSystem.git))

×変更を取ってくる(pull)

git fetch --all または　git fetch origin　←リモートの変更を取得

git merge origin 【ブランチ名称】　←他者の変更とは別に自分の変更を追加し合流させる

git rebase origin 【ブランチ名称】　←他者の変更後に自分の変更を追加