DaVita MCOE Branching Strategy

Purpose

This guide presents a recommended approach for managing git branches during development, testing and release of mobile applications.

master - what is currently IN production (we need to define what production means) - never pushed directly into - merged from a qa/release branch using pull request

development - what is currently DONE in a development sprint - merged from feature branches at end of sprint - merge party where pull requests are reviewed / accepted / modified / etc

qa / [release] - qa branch for integration testing of development - created straight from development branch after a sprint - pull requests allowed during sprint but can't be closed until merge party - any pull requests reviewed must also be considered for development branch merging as well - if release ready for production, a pull request to master is performed - once pulled into master, the master branch is tagged and the qa branch is killed during merge party

hotfix / [release] - branch to track hot fix that needs to be made in production - this branch can be pushed directly into and qa tested from - once ready, a pull request to master is performed - once pulled into master, the master branch is tagged and the hot fix branch is killed during merge party

[kind] / [owner] / [description] - feature branches where pushes can be made to directly - owner is the team member responsible for the branch - SCRUM testing can be done from this branch - at end of sprint, pull request to development branch is made (or qa_ branch as appropriate) - branch must die after merge party unless it is not merged and if team agrees it can stay - kind should reflect the branch where the feature/fix will be merged TO. The kind should be one of the following: dev, qa, hotfix - kind may also refect an experimental branch that will never be directly merged into another branch. These branches should begin with the kind of experiment

examples * development * dev/house/super_cool_new_feature_us1221 * dev/boris/another_cool_feature_us1432 * qa/release_1_1 * hotfix/release_1_0_1 * qa/boris/defect_fix_1234 * hotfix/house/fix_oops_bug * experiment/house/evil_genius_stuff_here

Reference

This guide is similar to the gitflow approach from this article.

https://www.atlassian.com/git/workflows#!workflow-gitflow