Waylen Allen

WAP-3-107 GitHub Branch Management

Version 1.1

16th August 2024

Date	Change	Author	Description	Version
14th August 2024	Creation	Hao Nguyen	Document Creation	1.0
16th August 2024	Modify	Dr Simon Benson	Minor Edits	1.1

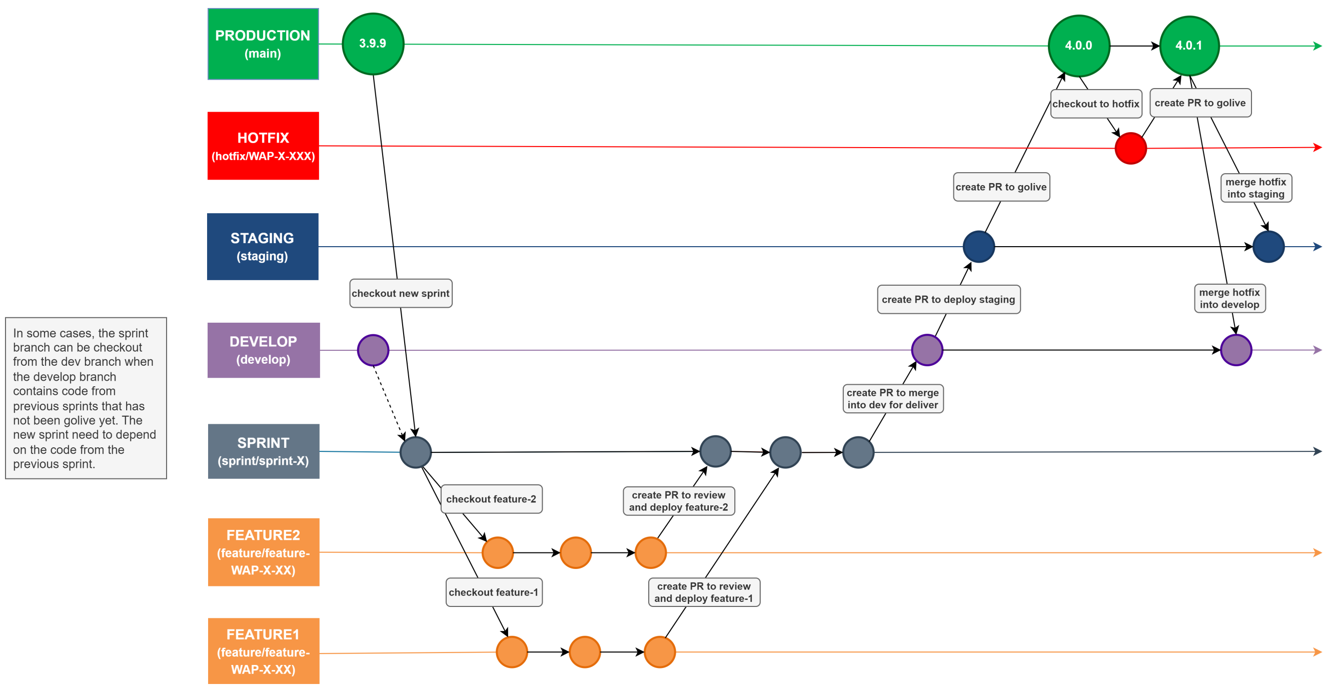
1. Scope

This document describes how to manage branches on GitHub. What branches mean, how to create branches, merge code, as well as standards for naming and code review process.

2. Details

Source code will be stored on GitHub. For a normal project, there will be 3 main environments: QA, Staging and Production. Corresponding to these 3 environments, we will have corresponding branches: develop, staging and main.

2.1 GitHub branch management


PRODUCTION (main)
HOTFIX (hotfix/WAP-X-XXX)
STAGING (staging)
DEVELOP (develop)
SPRINT (sprint/sprint-X)
FEATURE2 (feature/feature-WAP-X-XX)
FEATURE1 (feature/feature-WAP-X-XX)
3.9.9
4.0.0
4.0.1
checkout to hotfix
create PR to golive
create PR to golive
merge hotfix into staging
merge hotfix into develop
create PR to deploy staging
create PR to merge into dev for deliver
checkout new sprint
checkout feature-2
create PR to review and deploy feature-2
checkout feature-1
create PR to review and deploy feature-1

In some cases, the sprint branch can be checkout from the dev branch when the develop branch contains code from previous sprints that has not been golive yet. The new sprint need to depend on the code from the previous sprint.

(GitHub branch management)

- When starting a new sprint, by default we will create a new sprint branch from the main branch. The format of the sprint branch will correspond to the name of the sprint and will have the format *"sprint/sprint-X"* (eg. sprint/sprint-17). By default, the sprint branch will be checked out from the main branch, however in some special cases it may be checkout from other branches such as the develop branch. In case the develop branch has some required source code for the new sprint that are not on main branch yet. For example, the hardware team will soon develop a new version of Bluetooth, so they can checkout the sprint branch from the develop branch.

- When team members receive a task, they checkout a new feature branch from sprint branch with the format "*feature/[taskId]-[task-short-description]*" (eg. feature/ WAP-9-3007-forgot-password-screen). Similar to sprint branches, feature branches can also be checkout from another branch(main branch) (depend on the situation).
- After completing coding, team members will create a pull request into the sprint branch for review and then proceed with the build. Next, merge into the develop branch and staging branch to push to the environment for QA team/product owner to test and deliver. In special cases, if you want to go-live part by part by feature, you can merge the feature branch into staging and continue the process.
- When the product owner confirms to release a new product version, we will create pull request and merge from staging branch to main branch, then mark the new release version on GitHub. Sometimes, If we do not want to release everything on staging into production, we can create pull requests from sprint/feature branches into main branch.
- For issues that are required to be fixed immediately, we will create a new hotfix branch from the main branch with the format "*hotfix/[taskId]-[task-short-description]*". After fixing and testing the hotfix branch, we will merge it into main and release it. When the hotfix is deployed successful, we need to merge it back to staging and develop to make them upto date with the main branch.

2.2 Code review flow

- New pull requests must follow a format that ensures easy search by ticket:
    - Pull requests must have a title in the format "*[taskID] [task-description]*" (eg. [WAP-9-3081] Make comment for code Bluetooth).
    - Pull requests must be ensured to have Dev Lead review and resolve all comments before merging.
- Make sure the staging and main branch are protected, no one can push directly from local to these branches.
    - Developers must create a pull request to make changes.
    - Only the Dev Lead has permission to merge pull requests into these two branches.