**CI/CD Pipeline Workflow and Git Branching Strategy**

This document gives the high level overview of CI/CD pipeline work flow and git branching strategy for smooth release of ongoing features and hotfix releases.

**Pain points:**

1. Slow feature release cycle
2. Hotfixes have higher priorities,
3. QA wants only specific changes on the QA environment for testing (feature/hotfix/bugfix)
4. UAT blocked because of stakeholders are taking long time
5. PR (pull request) merging taking long time
6. Increasing testing cycle and lot of back and forth between multiple teams
7. No ownership of controlling the releases.
8. Rollback strategy

**Proposed solution branching strategy:**

1. master branch (always locked, merge via PRs only) -
2. Long living trunk branch that holds the latest production copy.
3. This branch should be locked for any direct merging after the QA signed off the current release.
4. Only changes related to hotfix and bug fixes in the current release cycle are allowed to be merged on this branch.
5. Developer is responsible for merging changes from master to dev branch related to bugs and hotfixes.

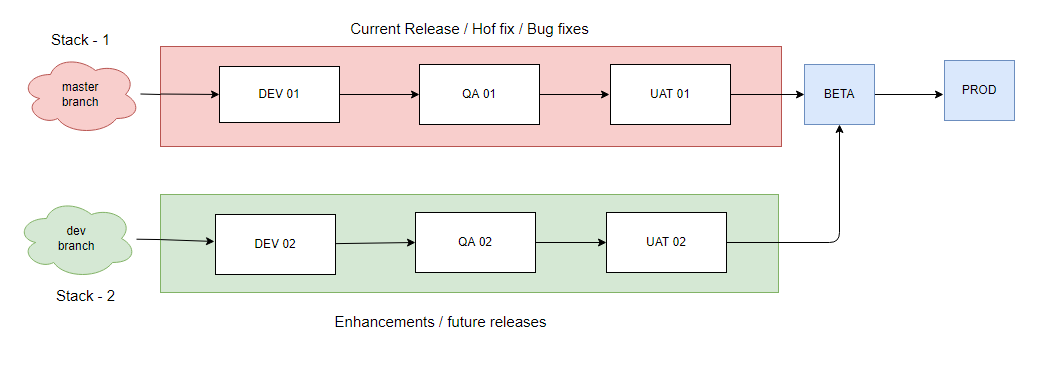
2. dev(always locked and no direct merge, merge via PRs only)/ feature branch -

1. Any features that should go to production on next releases will be added in this branch.
2. Developer is responsible for merging the hotfix or bugfix changes to the dev from the master branch.

**Note:** Once the QA signs off the features, the build will move to the UAT environment. UAT will have all the feature changes in the current release. Based on the release cycle the build will be moved to beta and prod.

Each environment will have its own infrastructure like database, sns, sqs etc.

The below diagram gives the overview of the branching strategy discussed above.



**Stack 1:**

1. Master branch will have the latest code changes related to the current release cycle.
2. Once QA has signed off the current release in the QA environment, artefacts/build from QA environment will be moved to UAT environment.
3. Once QA is signed off, the master branch will be locked and not allowed to add any changes.
4. From UAT based on the release timelines, the build moved to beta and prod environments.

**Stack 2:**

1. dev/future branch will have the new features for the next release.
2. Developer has to merge the changes associated with hotfix and bug fixes in the current release.

**Note:** Stack 1 and Stack 2 are two different workflows added for understanding the branching strategy.

CI/CD requirement:

1. Master and develop branch should be locked
2. Master and develop branch should be updated via only MR or PR
3. MR (Merge Request) or PR can’t be merged without Approval, so at least one approval should be required (you may keep one user as super user in case of any urgent fix required to merge and deploy on prod/staging)