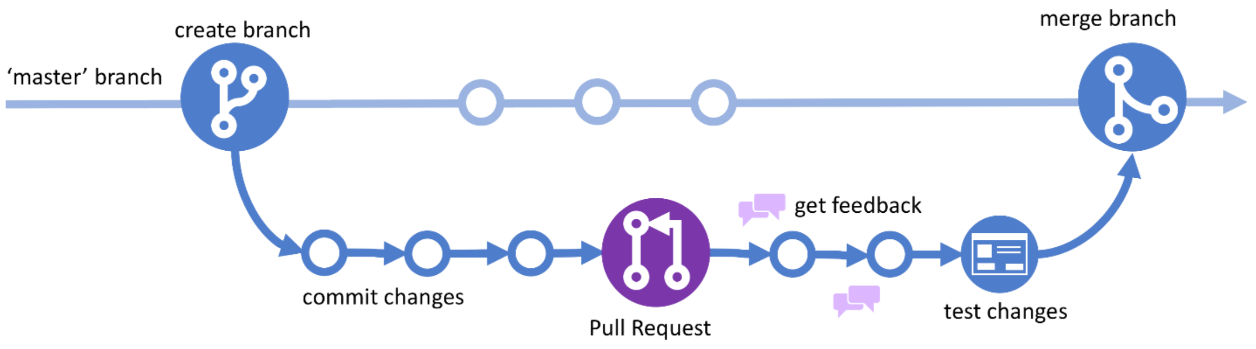
Release flow proposal

Release flow is a set of steps to perform to release upcoming version of software product. Main aim of this document is to present simple and working model of software release using semantic versioning, azure pipelines, and mainline development (also known as GitHub flow). Current document is motivated by Microsoft's "Adopt a Git branching strategy". Let's briefly describe how GitHub flow works, see picture below.



Picture 1. GitHub flow diagram

Generally, we have following kinds of branches:

1. **Master** – is the branch we release code from and then deploy it to various environments like DEV, QA or UAT. Each state of master branch is deployable.
2. **Feature** – feature branch that created to implement new feature according to sprint plan. Feature branch is branched from the master's HEAD.
3. **Bugfix** – same as feature branch in general, branched out of master's HEAD and contains not critical bug fix.

4. **Release/v*** – branched from master and ready to be released, CD pipeline itself is triggered by push tag event from this branch. <u>It is long-living branch used to support particular release.</u>
5. **Hotfix** – branch is created from latest release/v* branch. This branch contains hotfix that should be deployed as soon as possible. In case of hotfix branch, release to be done directly from it. It means that after hotfix is finished new tag with patch increment to be created and pushed, triggering CI/CD and deployment of hotfix to production.

Having all above, assume we have current initial semantic version on our main branch as v0.1.0 and we must release upcoming version, our steps to perform are:

1. Create pull request from recent feature branch to master branch, this pull request triggers Continuous Integration (CI) to start, CI runs tests, code quality checks etc., but deployment won't be started yet, only CI.
2. After all checks passed, pull request reviewed by team and every comment from code review is fixed – pull request to be merged from Feature branch to Master branch. No CI/CD pipeline triggered by the merge.
3. Next, responsible person reviews changes (using change log or git compare, it doesn't matter, but change log is preferable solution), responsible person makes decision which part of semantic version to increment and then creates and pushes tag to remote repository respectively. For example, it is decided that minor version should be incremented then our version becomes v0.1.0 -> v0.2.0, so responsible person creates release as follows:
    - Checkout to branch release/v0.2.0 from Master
    - git tag -a v0.2.0 -m "Release v0.2.0"

- git push origin v0.2.0

4. After step 3 CI/CD process is triggered by tag push event, after build there are three deployments scheduled: DEV, QA, UAT. Environments QA and UAT are to be approved by designated personnel before deployment starts, DEV to be deployed without approvement.

Sources

- SemVer.org
- GitVersion: Mainline development mode
- Microsoft's "Adopt a Git branching strategy"
- Azure pipelines trigger types