RELEASE FLOW PROPOSAL

PETRO KOLOSOV

Abstract. In this document software product release process is proposed and discussed.

Contents

1. Introduction

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 [1], azure pipelines, and mainline development [2]. Mainline development is also known as GitHub flow. Current document is motivated by Microsoft's "Adopt a Git branching strategy", see [3]. Below picture shows main idea of GitHub flow itself

Date: January 28, 2023.

Key words and phrases. Software engineering, DevOps, Software release, GitHub flow, GitLab flow, Azure DevOps, Azure pipelines, Semantic versioning, GitVersion, CI/CD .

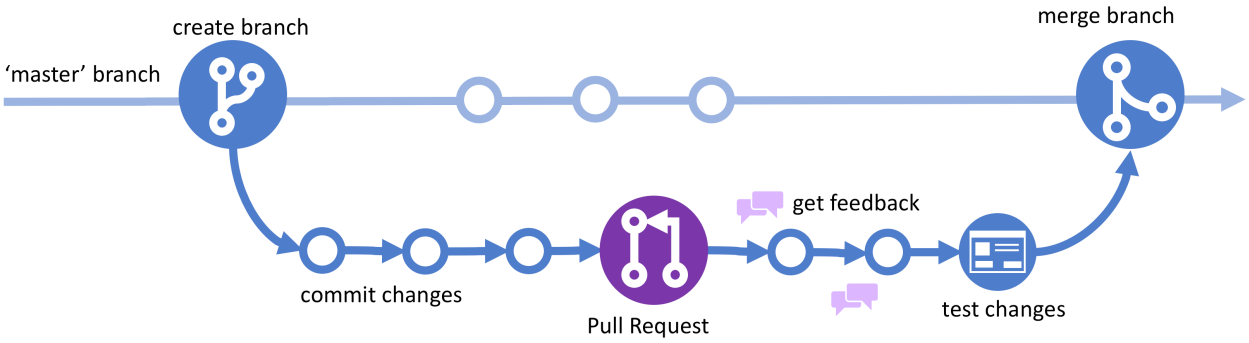

Figure 1. GitHub Flow diagram.

- **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.
- **Feature** – feature branch that created to implement new feature according to sprint plan. Feature branch is branched from the master's HEAD.
- **Bugfix** – same as feature branch in general, branched out of master's HEAD and contains not critical bug fix.
- **Release/v*** – branched from master and ready to be released, CD pipeline itself is triggered by push tag event from this branch. It is long-living branch used to support particular release.
- **Hotfix** – branch is created from the latest deployed unit where error occurs, for example `release/v*` branch. Hotfixes are not stick to releases, they may be necessary in any deployed unit like DEV, QA environments, depends on where error occurs. 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.

1.1. **Release process.** 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 push origin release/v0.2.0`
    - `git tag -a v0.2.0 -m "Release v0.2.0"`
    - `git push origin v0.2.0`
(4) After that CI/CD pipeline is triggered by tag push event [4], 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 manual approve.
(5) Finally, `master` branch cherry-picks from `release/v0.2.0` after deployment is complete.

Entire process is shown on picture below

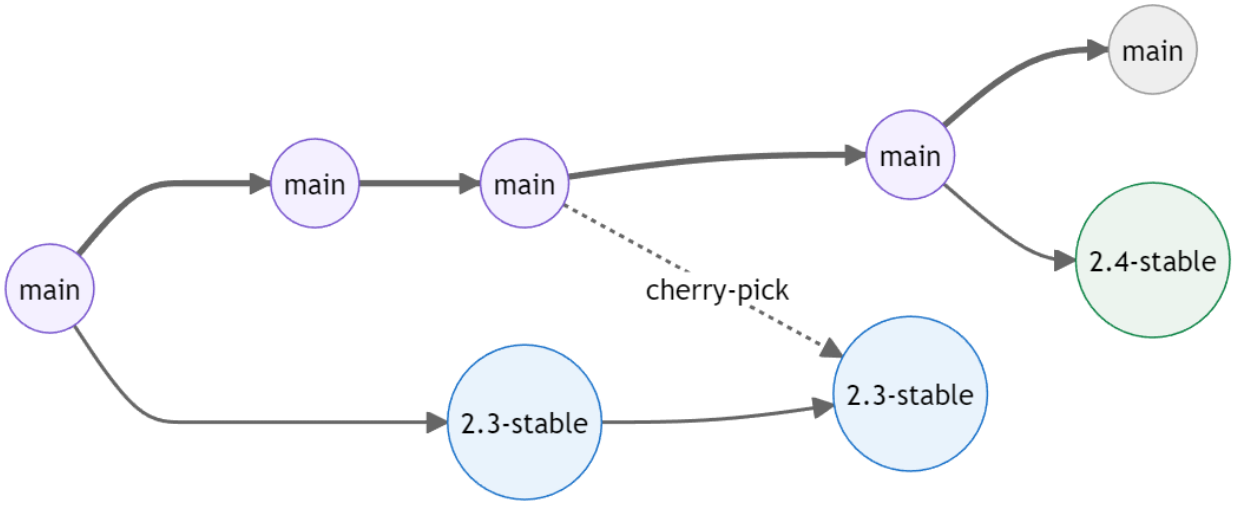

Figure 2. GitLab Flow diagram [5].

2. Conclusions

In this document software product release process is proposed and discussed. Few useful GIT commands worth to remember:

- `git tag -a v0.1.0 -m "my version 0.1.0"`
- `git tag -d <tag_name>`
- `git push origin <tag_name>`
- `git push --delete origin <tag_name>`

References

[1] Semantic Versioning docs. Semantic Versioning 2.0.0. Available electronically at https://semver.org/, 2023.

[2] GitVersion docs. Mainline Development. Available electronically at https://gitversion.net/docs/reference/modes/mainline, 2023.

[3] Microsoft documentation. Adopt a Git branching strategy. Available electronically at https://learn.microsoft.com/en-us/azure/devops/repos/git/git-branching-guidance?view=azure-devops, 2022.

[4] Microsoft documentation. Build Azure Repos Git or TFS Git repositories. Available electronically at https://learn.microsoft.com/en-us/azure/devops/pipelines/repos/azure-repos-git?view=azure-devops&tabs=yaml, 2023.

[5] GitLab docs. Introduction to GitLab Flow. Available electronically at https://docs.gitlab.com/ee/topics/gitlab_flow.html, 2023.

Email address: kolosovp94@gmail.com

URL: https://kolosovpetro.github.io