SOFTWARE PRODUCT RELEASE FLOW: A 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* available at [3]. The picture below shows the main idea of GitHub flow

Date: January 4, 2025.
Key words and phrases. Software engineering, DevOps, Software release, GitHub flow, GitLab flow, Azure DevOps, Azure pipelines, Semantic versioning, GitVersion, CI/CD .

Sources: https://github.com/kolosovpetro/ReleaseFlowProposal

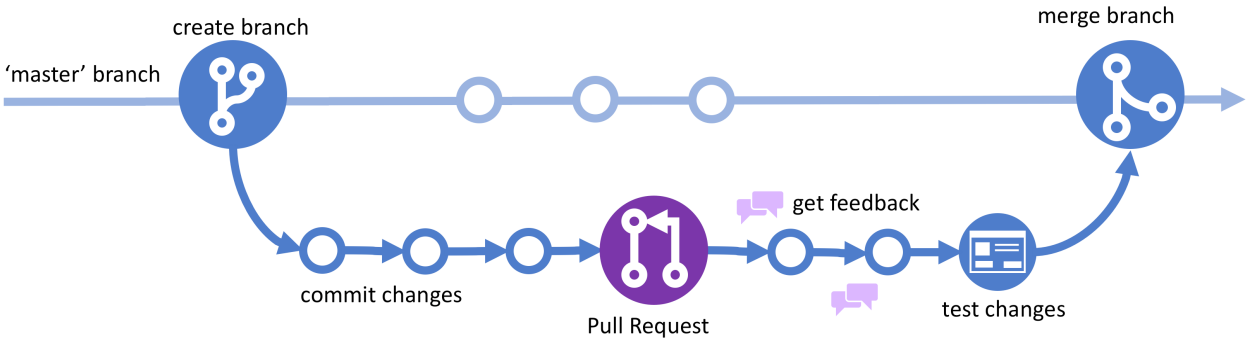

Figure 1. GitHub Flow diagram.

- **Master** – branch that contains tested, validated and verified code, ready to be released and deployed to production.
- **Feature** – branch that contains implementation of a new feature according to sprint plan. The `feature` branch is based onto `master`.
- **Bugfix** – branch that contains non-critical bug fix. The `bugfix` branch is based onto `master`, and merged back to `master` after fix is done.
- **Release/v*** – branch that contains upcoming release state of software product, serves to keep small changes and updates to `CHANGELOG` file. The `Release/v*` branch is based onto `master`. The `Release/v*` branch is considered to be long-living branch, it should not be deleted after code is deployed to production. After release is completed it is merged back to `master`.
- **Hotfix** – branch that contains critical bug fix. It is used to patch production environment and must be released as quick as possible. The `hotfix` branch is based onto the latest released `Release/v*` branch. After hotfix is released it is merged back to its base `Release/v*` and Cherry-picked [4] by `master`.

1.1. **Release process.** Having all above, assume we have initial semantic version of our application as `v1.0.0`, so that we must release upcoming version. The version `v1.0.0` has

been tested by QA team, so that release was approved by whole team. General release steps to perform are following

(1) **Code phase.** Software engineer creates 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 is not started yet, only CI.

(2) **Code phase.** After all CI checks passed, pull request reviewed by team and every comment from code review is fixed – the `feature` branch is ready to be merged into `master` branch. No CI/CD pipeline triggered by the merge.

(3) **Code phase.** Next, release engineer reviews software product changes documenting them in `CHANGELOG` file. Release engineer decides on the next Semantic Version [1] increment. For example, software product has breaking changes, then release engineer decides to increment the major part of semantic version, so that `v0.1.1 -> v1.0.0`

(4) **Code phase.** Release engineer creates new release as follows
   - Checkout to release branch: `git checkout -b release/v1.0.0`
   - Adding minor changes and `CHANGELOG` file update
   - Push release branch to remote: `git push origin release/v1.0.0`
   - Create tag: `git tag -a v1.0.0 -m "Release v1.0.0"`
   - Push tag: `git push origin v1.0.0`

(5) **Build phase.** After new `TAG` is pushed to the remote repository, the build pipeline is being triggered [5], initializing the build phase of DevOps cycle. Therefore, the code is being built, tested and specific artifacts are being created and published.

(6) **Release phase.** Release engineer validates the build artifacts, underlying infrastructure and deployment automations, ensuring smooth and reliable upcoming deployment.

(7) **Deploy phase.** There are a few deployments scheduled including the environments `DEV`, `QA`, `UAT`. Deployments to `QA` and `UAT` environments are to be approved by designated personnel, meanwhile `DEV` environment to be deployed automatically.

(8) Finally, the `Release/v*` branch is merged back to `master` after deployment is complete.

Entire release process is shown on the picture below

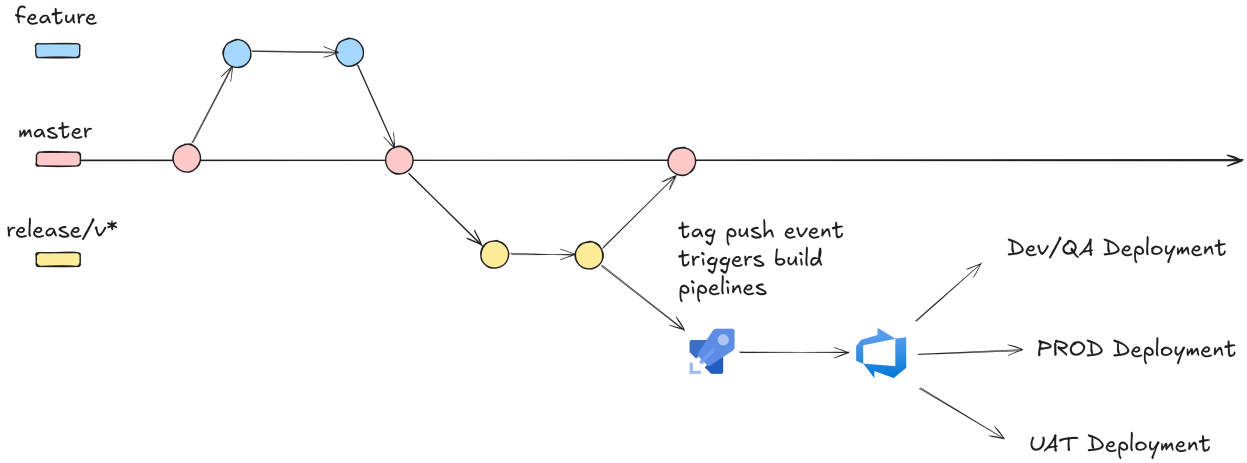

Figure 2. Release flow diagram.

1.2. **Hotfix strategy.** Assume that our current released version of software product is `v1.0.0` and there is a critical bug appears. In order to release a hotfix the following set of steps to be performed

(1) Hotfix to be assigned to a software engineer.
(2) Software engineer creates a new branch `hotfix/*` from the latest `Release/v*` branch.
(3) Software engineer fixes the bug in the `hotfix/*` branch.
(4) Software engineer creates a pull request `hotfix/* -> release/v*`.
(5) The pull request `hotfix/* -> release/v*` is reviewed by team and merged.
(6) Release engineer creates a new tag `v1.0.1` from the latest `release/v*` branch.
(7) Hotfix deployment process is started after new tag is pushed.
(8) Hotfix is deployed to production.
(9) Release engineer creates a pull request `release/v* -> master` that contains the hotfix.
(10) The pull request `release/v* -> master` is reviewed by team and merged.

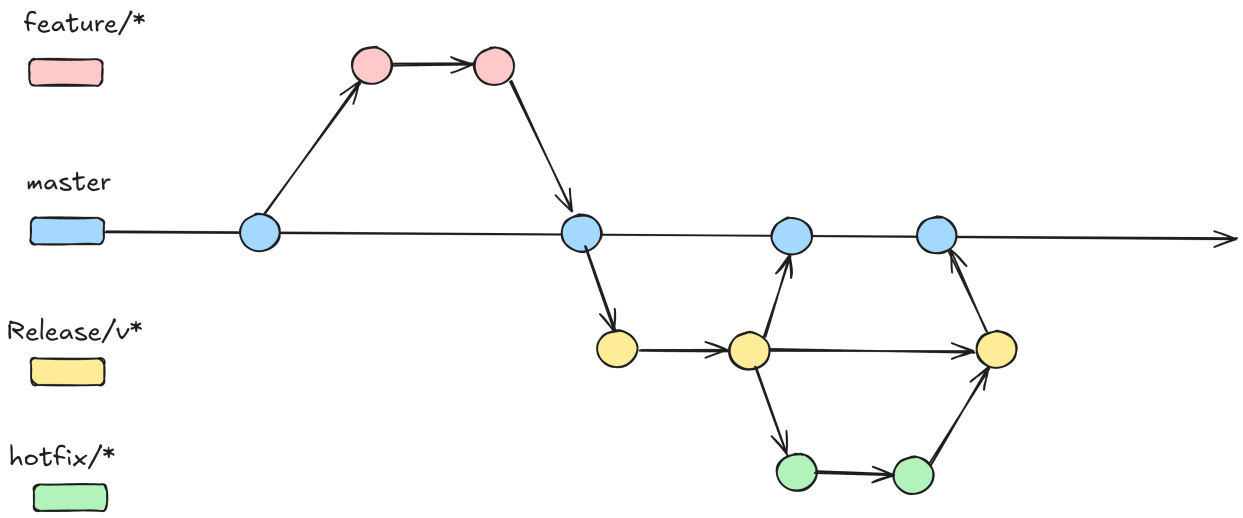

Figure 3. Hotfix diagram.

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, 2023. https://semver.org/.

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

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

[4] Atlasian Docs. Git Cherry Pick, 2023. https://www.atlassian.com/git/tutorials/cherry-pick.

[5] Microsoft Documentation. Build Azure Repos Git or TFS Git repositories, 2023. https://learn.microsoft.com/en-us/azure/devops/pipelines/repos/azure-repos-git.

Version: `Local-0.1.0`

Software Developer, DevOps Engineer

Email address: `kolosovp94@gmail.com`

URL: `https://kolosovpetro.github.io`