Branching Strategy

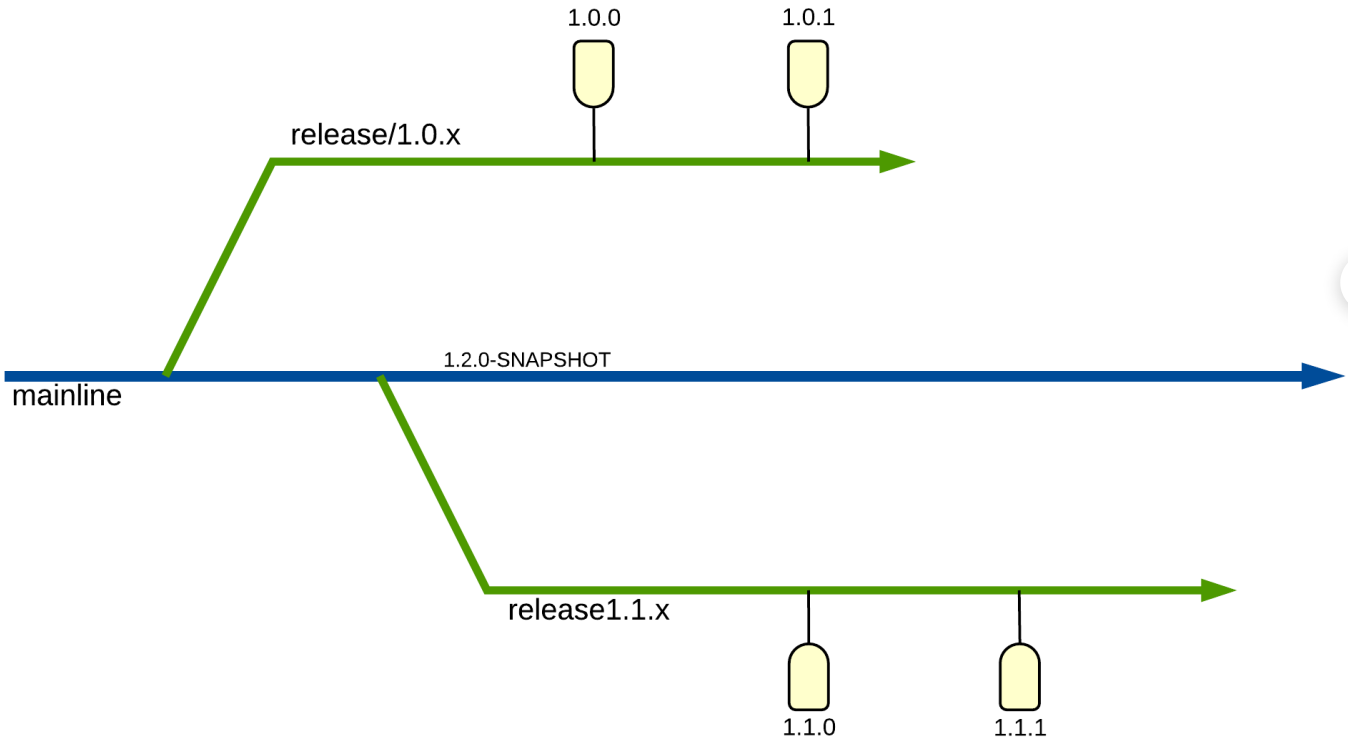
Edit New page

Jump to bottom

Zander Rommelaere edited this page yesterday · 6 revisions

Branching Strategy

The branching strategy that we will be using is more or less a simplified version of the GitFlow strategy.



1. Master Branch

- Exclusively contain fully completed and tested deliverables.
- Deployed to the Production environment.

2. Release Branch

- Contain completed and tested work of the current deliverable, once the deliverable is finished, will be merged into the master branch through a Pull Request (attached with a Git issue).

- Serve as the starting point for all pull requests and topic branches.
- Code freeze occurs on the Friday before the deliverable deadline; no new merges are accepted on the release branch thereafter.
- Get deployed to Testing Environment.
- Release branch naming convention:

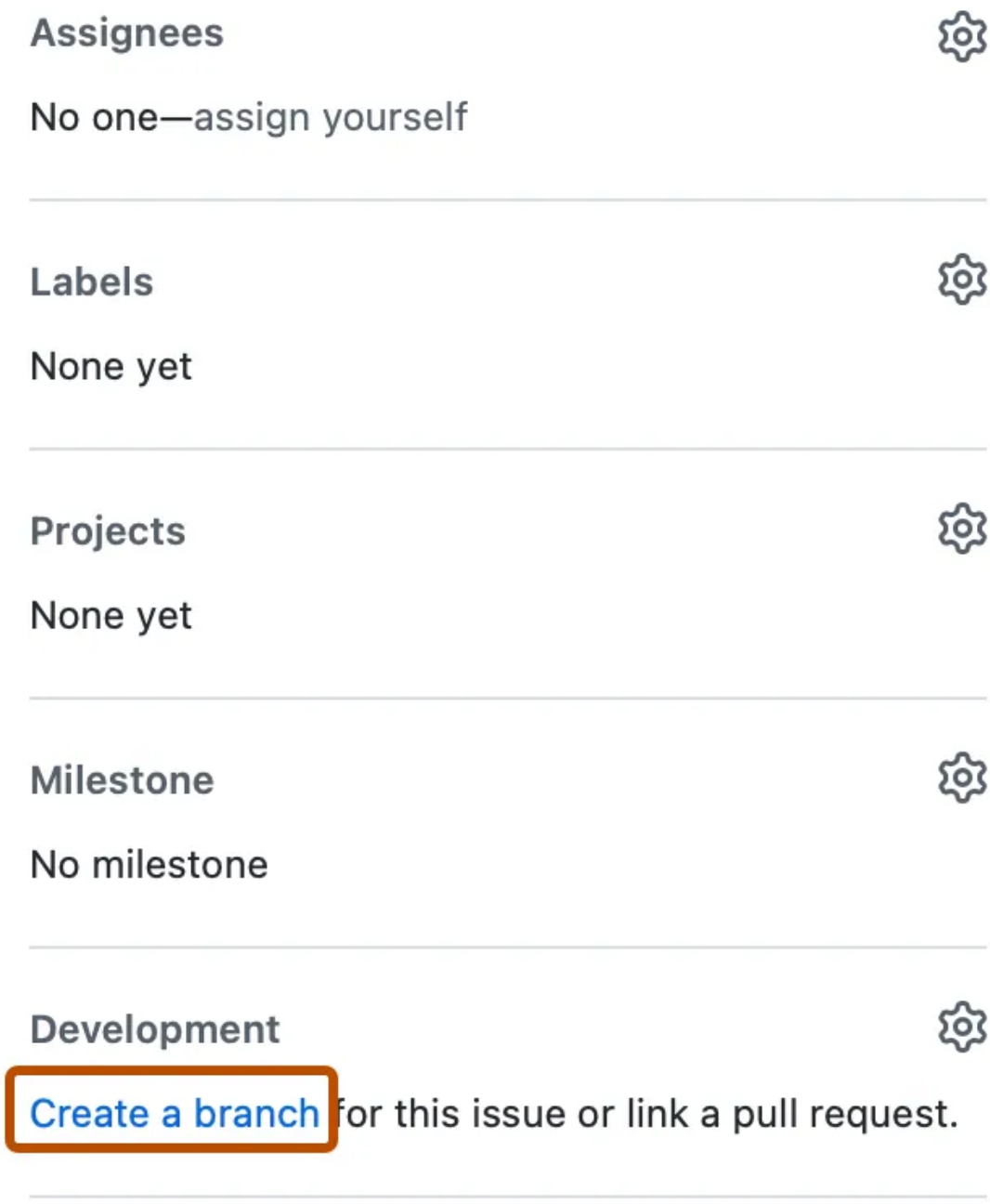
```
release/<deliverableNumber>
```

3. Individual/Topic Branches

- Contain work-in-progress features and bug fixes.
- Based off the current release branch.
- Upon completion, should be merged into the release branch
- **Topic branch naming convention**:

```
#<issueNumber>-name-of-the-branch
```

Note: Branches CANNOT be linked automatically to the issue by prefixing branch name with #<issueNumber>. To link a branch to an issue, please choose 'Create the branch' under 'Development' in the right sidebar of the iss



- **Commit message convention**:

```
#<issueNumber> [Action] Short summary (50 chars or less, this is the subject)

More detailed explanatory text, if necessary. Maximum 80 characters to wrap it.

- Hyphen for bullet points.
```

Note: #<issueNumber> will automatically be linked the issue.

- Good Examples of Commit Messages:

```
feat: improve performance with lazy load implementation for images
chore: update npm dependency to latest version
Fix bug preventing users from submitting the subscribe form
Update incorrect client phone number within footer body per client request
```

- Bad Examples of Commit Messages:

```
fixed bug on landing page
Changed style
address review comment
oops
I think I fixed it this time?
[empty commit messages]
```

4. How to deal with bugs

- If bugs are **found mid-release**, bug fixes should be merged in the release branch.
- If bugs are **found after the problematic release branch (where the bug is introduced) is merged**, there are 3 techniques that you can consider:

1. If the bug fix is **minor** and we currently **have no release branch**.

    -> Merge the bug fix into the problematic release branch.

    -> Re-merge the release branch into the master branch.

2. If the bug fix is **minor** and we currently **have a new release branch** (different from the one that introduces the bug):

    -> Merge the bug fix into the current release branch.

3. If the bug is **critical** and/or **contains a substantial amount of code changes**:

    -> Merge the bug fix into the problematic release branch

    -> Create a new branch off the master branch

    -> Cherry-pick the bug fix from the problematic release branch to the new branch

    -> Merge the new branch into master

5. Development Environments

There will be two distinct environments that will have unique Firebase deployments:

- Testing Environment: Built from the release branch.
- Production Environment: Built from the master branch.

6. GitHub Actions/Build Pipeline

The flow should be as follows:

1. Run base actions (e.g. build the application, run check style, security scanning, tests, etc).
2. If PR is into release, deploy to the testing environment.
3. If the sprint is completed and ready, merge it into the production environment.