Exercise 4.9 – Team BDF Branching Strategy

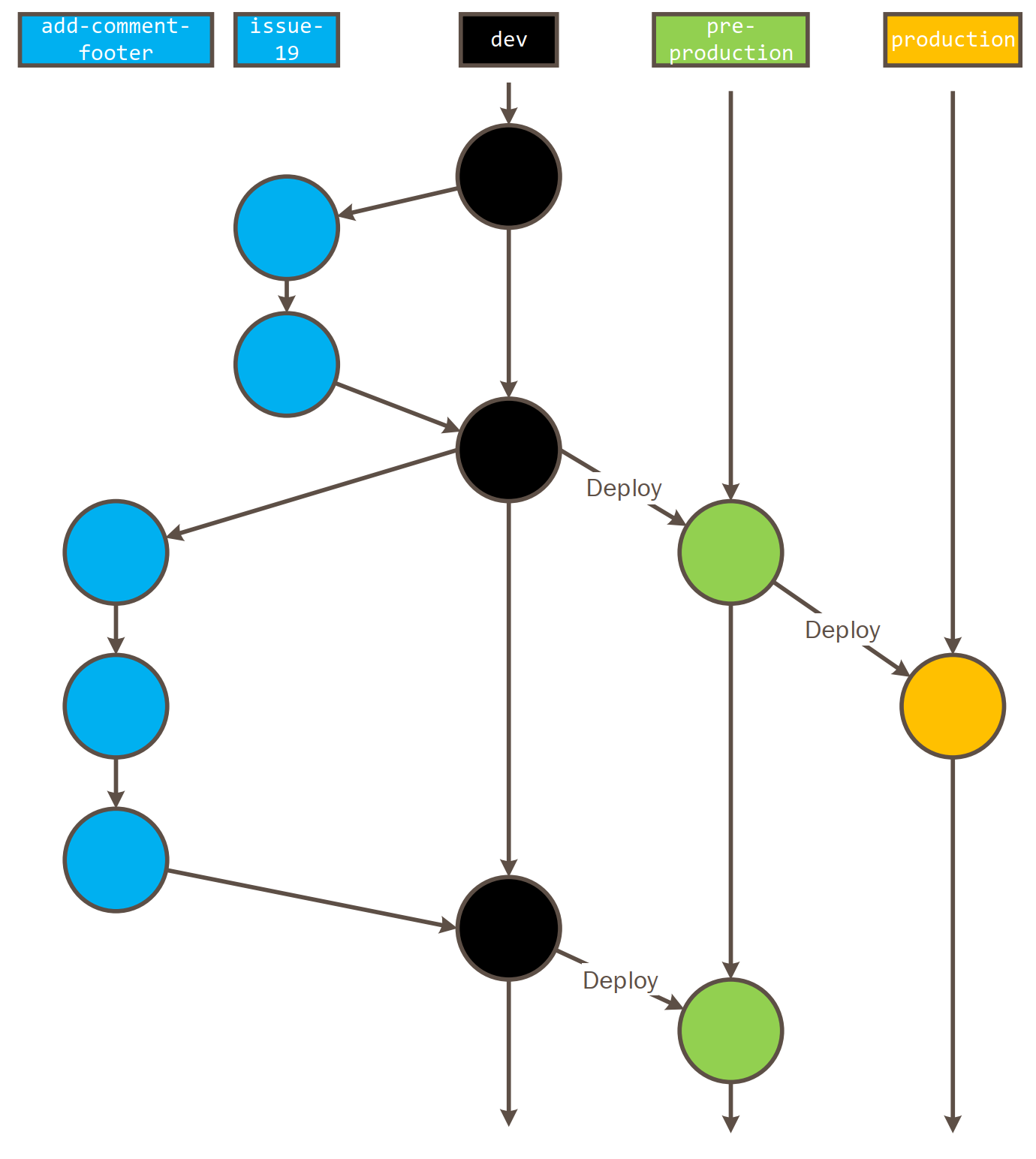
Team BDF has chosen to implement a slightly modified version of the GitLab Flow branching strategy [1]. We feel that this strategy works best for our relatively small team, and the type of project that we aim to develop. Most of the other branching strategies we looked at, including Git flow, Trunk Based Development, Dictator and Lieutenants Workflow, and Dekray’s Scalable Git Model are more complex than what we need, and thus will likely not result in enough benefit to compensate for their relatively high overhead. Our project will execute as an always-running server application, and thus will ideally have a 24/7 “production” environment. As a result, this project will benefit from smaller incremental changes as opposed to having large releases, which the GitLab Flow branching strategy easily accommodates.

The main modification to GitLab Flow that we will make is changing the name of the main flow branch from master to dev. We are making this change to reduce confusion that might arise from any preconceived ideas about what a master branch typically has. Specifically, we do not feel ready to commit to having a fully stable product at the end of every feature branch, particularly towards the beginning of the project. Feature branches will coincide with issues raised within GitHub. For any change or feature added a feature branch will be opened to address those specific changes. We also plan on implementing environment branches (see next paragraph), but not release branches.

GitLab Flow is flexible in that it allows any (reasonable) number of environment branches alongside the main flow branch (dev in our implementation). We plan to have two environment branches: pre-production and production. Ideally, we will have two actual environments (likely running on the same hardware or virtual machine) that are watching either the pre-production branch or the production branch for any commits. When a change is detected, the environment will automatically be updated with the state of the repository in the (pre-) production branch. The pre-production and production environments will be very similar in setup, but the pre-production branch will have additional restrictions that limit the negative effects of any faults (e.g. posting only in a community that is dedicated for testing). We plan on releasing to production roughly once a week, resulting in a “deployment day” (likely to be every Friday).

The GitLab Flow branching strategy also suggests some process guidelines that work well with the GitLab Flow strategy. We plan to implement most of these, with an emphasis on “intermediate results” and integrating bugfixes to main flow branch (dev) followed by cherry-picking those fixes into (pre-) production. The former will help prevent work from stagnating and not being shared with the rest of the team by having us open PRs whenever a feature branch has been open for a while without it being completed. However, we plan to extend GitLab’s suggestion of a “few hours” to 24 hours to accommodate for student schedules. The other emphasis on cherry-picking will help prevent regressions while also ensuring that production gets the latest bug fixes even if work had already progressed in the main flow branch.

**Example workflow with Team BDF’s branching strategy:**



# References

|  |  |
| --- | --- |
| [1] | S. Sijbrandij, "GitLab Flow," 29 September 2014. [Online]. Available: https://about.gitlab.com/2014/09/29/gitlab-flow/. |