IT UNIVERSITY OF COPENHAGEN

DevOps, Software Evolution and Software Maintenance

COURSE CODE: BSDSESM1KU

BACHELOR IN SOFTWARE DEVELOPMENT

DevOps: ITU-MiniTwit

GROUP R — RHODODEVDRON

IT UNIVERSITY OF COPENHAGEN

Name	Email
Adrian Valdemar Borup	adbo@itu.dk
Albert Rise Nielsen	albn@itu.dk
Joachim Alexander Borup	aljb@itu.dk
Thomas Wolgast Rørbech	thwr@itu.dk

May 17, 2022

Contents

1 Process Perspective

1.1 Organisation of repositories

The group uses GitHub to manage repositories. On GitHub, we have a dedicated organization that owns all of the project repositories. This way, we can keep related repositories together and reuse permission settings on GitHub. Currently, we have the following repositories:

1. `Devops-2022-Group-R/itu-minitwit`, which is the main project repository. Here, we have the backend web server which handles all business logic.
2. `Devops-2022-Group-R/itu-minitwit-frontend`, containing our frontend web-app.
3. `Devops-2022-Group-R/flag-tool`, which is a rewrite of the original flag tool given with the project template. The tool has been rewritten and moved to a separate repository.
4. `Devops-2022-Group-R/bump-tool`, which is a small tool to help with finding the next version bump based on a pull request's tag (major/minor/patch) and the project's current version.

The general philosophy has been to separate parts that can exist alone with a single responsibility. This way, all issues, pull requests, and releases are related to the topic of the repository. In a monolithic repository, you have to put in more effort to specify which part of the repository is relevant for a PR or issue. This also allows for easier and more simple CI/CD pipelines, and it becomes more obvious what a release changes. One example of this separation is the frontend web application, which is completely separated from the backend, with Kubernetes specs and CI/CD pipelines in its own repository.

However, we have not been as good at following this philosophy as we would like to. For example, our monitoring is a separate entity from our web server, but all the monitoring configuration is stored in the `itu-minitwit` repository [1] [2]. Instead, we should have moved this to a separate repository because it does not have anything to do with how the web server operates.

The same goes for the LaTeX files that this report consists of. We would have liked to have a separate repository for this in order to keep Git history clean and CI/CD more separate, but in this case it is a project requirement to have it in the main repository.

1.2 Git branching strategy

We have applied trunk-based development, meaning we branch out from the main branch, have a branch that lives shortly until the changes are implemented, and then merge it directly into the main branch. This keeps unrelated changes separate and makes PRs easier to review and merge. It's worth noting that we do not utilise release branches from trunk-based development, because we release continuously for each merge into the main branch.

2 References

[1] Group R. *Configuration files for ITU-MiniTwit monitoring*. Accessed: May 17 2022. Apr. 2022. URL: https://github.com/Devops-2022-Group-R/itu-minitwit/tree/master/monitoring/grafana/provisioning.

[2] Group R. *Kubernetes specs for ITU-MiniTwit monitoring*. Accessed: May 17 2022. Apr. 2022. URL: https://github.com/Devops-2022-Group-R/itu-minitwit/tree/master/.infrastructure/kubernetes/monitoring.

[3] Gene Kim. *The DevOPS Handbook - How to Create World-Class Agility, Reliability, and Security in Technology Organizations*. It Revolution Press, 2016. ISBN: 9781942788003.