aCompiler

GIT Best Practices With AFTER technique

GIT Best Practices Summary

Hey what's up, everybody? Rajeev Bera here.

And in this worksheet, I'm going to show you Summary of **GIT Best Practices** Also I will show all time favorite **AFTER technique**.

Let's get started.

#1. Atomic commit

#2. Commit early, Commit often

#3. Do not commit generated files

#4. Do not commit dependencies

#5. Do not commit local configuration files

#6. Do not commit half (broken) code

#7. Test your changes before committing

#8. Write useful commit messages

#9. Review your changes before committing

#10. Use the reference number with your commit

#11. Refer a commit by its hash

#12. VCS does not substitute for a good backup

#13. Choose a workflow

#14. Enforce standards

#15. Integrate with external tools

#16. Use pull request

#17. For one concern - one pull request

#18. Do not delay with pull requests

#19. Complete your work before a pull request

#20. Comments with your pull request

#21. Use branches

#22. Branch naming convention

#23. Delete stale branches

#24. Keep your branches up to date

#25. Do not rewrite history

#26. Protect master

#27. Test before you push

#28. Define code owners

#29. Use GIT Ignore file

#30. Use rebase

#31. Keep your repositories healthy

#32. Use diff

#33. Stash name

#34. Tag your releases

#35. Don’t mix “refactoring” with a new feature

Now, I will show all-time favorite **AFTER technique**.

It will boost your productivity up to 80%

A - Atomic Commits

F - Frequent Commits

T - Test before push

E - Enforce standards

R - Refactoring is not a feature.